Яков Осипенков

Google Tag Manager

ИЗДАНИЕ ВТОРОЕ.
БОЛЬШЕ, СОВЕРШЕННЕЕ.

ПРАКТИЧЕСКОЕ РУКОВОДСТВО

coobiQ

Яков Осипенков

Самиздат: Google Tag Manager – 2020. – 536 с.: ил. – osipenkov.ru.

Перед вами обновленная версия электронного руководства по Google Tag Manager.

Прочитав эту книгу, вы поймете тонкости и нюансы работы с диспетчером тегов Google, сможете управлять всеми тегами в едином интерфейсе, сэкономите время в процессе настройки, снизите зависимость от разработчиков при внедрении кодов сторонних сервисов на ваш сайт, улучшите производительность путем снижения времени загрузки страниц, а также получите необходимые знания для дальнейшего карьерного роста по специальности «Веб-аналитик».

Рекомендуется к прочтению владельцам бизнеса, предпринимателям, студентам, стажерам, арбитражникам, фрилансерам, менеджерам по рекламе и всем тем, кто только собирается запустить свой собственный проект в интернете и хочет узнать, как правильно работать с самым популярным инструментом по управлению тегами в мире.

12+ (В соответствии с Федеральным законом от 29 декабря 2010 г. № 436-ФЗ.)

Дата выхода: 09 сентября 2020 г.

Издание 2, дополненное

Содержание

От автора

Прошло два года с момента выхода первого издания электронной книги **Google Tag Manager для googлят (2018)**, которая стала настольной для многих интернет-маркетологов и веб-аналитиков по всему миру.

За это время только из моего блога первое издание скачали более 6 800 раз, а суммарное количество загрузок в сети, по грубым подсчетам, превысило 10 000. Я получил большое количество отзывов, предложений, слов благодарности и поддержки от своих подписчиков за систематизацию и полноту изложенной информации.

Некоторые из читателей делились своими жизненными историями. Кого-то, благодаря полученным из руководства знаниям, взяли на должность веб-аналитика, кто-то после прочтения книги вдохновился и решил сменить род деятельности, кто-то понял, что диспетчер тегов Google не такой сложный, как про него говорят, и начал применять его в своих проектах, а кто-то просто стал больше зарабатывать, потому что расширил свои профессиональные навыки.

Именно такие рассказы от аудитории побуждают меня совершенствоваться и продолжать работу, создавая все новые и новые произведения, и публиковать материалы в блоге, которые понятны многим. А еще - любопытство и собственный вклад в развитие отрасли. Мне хочется, чтобы русскоязычное сообщество маркетологов не отставало от специалистов других стран и имело доступ к той информации, которая есть у них. Для этого необходимо, чтобы каждый человек делился опытом и знаниями друг с другом, а не держал все это в секрете. Только тогда будет прогресс.

Электронное руководство, которое вы держите сейчас в руках - новее, совершеннее своего предшественника, в два раза больше по объему и содержит структурированное и последовательное изложение теоретического материала с поясняющими практическими примерами. Немалый упор в этом издании был сделан на программный код, чтобы интернет-маркетологи смогли познакомиться с основами JavaScript и заложить базис для дальнейшего изучения этого языка программирования, который расширяет возможности работы с Google Tag Manager.

Книга не была в издательстве, ее не проверял корректор, поэтому в ней могут содержаться как речевые, языковые, так и орфографические и пунктуационные ошибки. Буду признателен тем, кто в случае нахождения таковых укажет на неточности, чтобы с каждой последующей версией издания их становилось все меньше и меньше. Пишите мне на почту ya.osipenkov@icloud.com

Яков Осипенков

Выпускник МГТУ им. Баумана (2008-2014), ведущий специалист по контекстной рекламе компании ConvertMonster (2016-2017), сертифицированный специалист продуктам Яндекса и Google, менеджер по мобильному трафику в компании Kokoc (2017-2018), автор нескольких курсов по веб-аналитике и контекстной рекламе, популяризатор веб-аналитики в русскоязычном сообществе. Мои статьи можно прочитать на таких крупных порталах, как seonews.ru, searchengines.ru, webpromoexperts.net, yagla.ru, habr.com, spywords.ru, ppc.world, convertmonster.ru

Контактная информация:

- **YouTube:** youtube.com/c/YakovOsipenkov
- **Telegram:** t.me/clicksider
- **ВКонтакте:** vk.com/yakov.osipenkov
- **Facebook:** facebook.com/yakov.osipenkov
- **E-mail:** ya.osipenkov@icloud.com

Условные обозначения

Ниже приведены условные обозначения, принятые в этой книге для удобства чтения:

1. **Жирным** и *Курсивом* выделены слова, на которых автор делает акцент, усиливая тем самым значимость конкретного слова или словосочетания.

2. (см. приложение) – отсылка к **Приложению** в конце руководства. В него вынесены ссылки и материалы других авторов, которые были использованы при написании этой книги.

3. Другой шрифт текста:

```
<script>
window.dataLayer = window.dataLayer || [];
window.dataLayer.push({'event':'value'});
</script>
```

Так выделяется специальный код, который вы можете использовать в своих проектах.

4. // комментарий

Синий цвет используется в коде для комментариев и пометок.

Обновление руководства

Вы держите в руках данное электронное руководство, потому что:

1. вы приобрели его официально на сайте osipenkov.ru;
2. вы получили его в подарок к моему онлайн-курсу по Google Tag Manager;
3. вам отправил его коллега / знакомый / друг;
4. вы скачали его нелегально в сети.

Если вы относитесь к первой и второй категориям читателей, то вам доступно одно следующее обновление данного руководства совершенно бесплатно, а также доступ ко всем кодам, файлам и дополнительным материалам из этой книги. Всем, кто приобрел руководство через мой сайт, я отправлю специальную ссылку на электронную почту, указанную в заказе.

Если вы относитесь к третьей категории читателей и получили копию этой книги от друзей или коллег, вы все равно можете выразить благодарность автору и купить руководство официально по ссылке https://osipenkov.ru/product/gtm-book. Тогда вам также будет доступно одно бесплатное обновление.

Если вы относитесь к последней категории, то поздравляю! Вы – пират! Пиратство является постоянной проблемой во всех средствах массовой информации. Я всегда серьезно отношусь к защите своих трудов, хоть и распространяю часть материалов в сети бесплатно. Это руководство распространяется платно.

Если вы столкнетесь с какими-либо незаконными копиями данного руководства в интернете (не считая ознакомительного фрагмента), пожалуйста, сообщите мне URL-адрес местонахождения или название веб-сайта, чтобы я смог проверить и принять соответствующие меры. Связаться со мной можно по адресу ya.osipenkov@icloud.com. Я ценю вашу помощь в защите моей интеллектуальной собственности.

Предисловие

Эта книга познакомит вас с инструментом Google Tag Manager. Вы пройдете путь от истории возникновения систем управления тегами, через фундаментальные знания по основам JavaScript, CSS-селекторам, переменным, триггерам и тегам, которые так необходимы в процессе работы с GTM, до практических навыков по настройке различных отслеживаний с помощью диспетчера тегов Google.

Глава 1. Начало работы с диспетчером тегов Google. Глава посвящена истории возникновения систем управления тегами, регистрации аккаунта Google, повествует о принципах работы GTM, дает понимание о навыках и знаниях, необходимых для его уверенного использования при решения практических задач.

Глава 2. Интерфейс Google Tag Manager. Глава, в которой каждый элемент интерфейса Менеджера тегов Google разобран с рисунками и подробными комментариями.

Глава 3. CSS-селекторы. Отдельная глава, которая посвящена селекторам CSS. Вы узнаете о разных типах селекторов и научитесь использовать их для поиска нужных элементов на странице, чтобы потом применять их для условий активации триггеров, а также в пользовательских переменных.

Глава 4. Переменные. В главе рассматриваются типы данных, которые встречаются в JavaScript, выражения и операторы (присваивания, сравнения, арифметические, строковые, специальные и т.д.), затрагивается тема преобразования типов данных, а также разбираются все встроенные и пользовательские переменные, которые существуют в Google Tag Manager.

Глава 5. Триггеры. Глава посвящена изучению всех типов триггеров и созданию условий, при которых активируются или блокируются теги диспетчера тегов Google.

Глава 6. Теги. Из этой главы вы узнаете о самых популярных тегах, которые собирают данные о посетителях на сайте, а затем пересылают их в другие сервисы.

Глава 7. Первые настройки в Google Tag Manager. Установка счетчиков веб-аналитики, настройка показателя отказов, отслеживание глубины скроллинга, видео YouTube, 404 ошибок, загрузки файлов, настройка уникального идентификатора пользователя (Client ID), а также определение точного времени обращения пользователя – все это в главе 7.

Глава 8. Работа с формами. Глава посвящена отслеживанию отправки форм 9 различными способами, включая брошенные формы (те, которые не были отправлены).

Глава 9. Работа с элементами на странице. В этой главе разбираются некоторые HTML-теги (<input> и <select>), которые являются одними из разносторонних элементов формы, браузерное событие клавиатуры keypress, а также настройки, связанные с отслеживанием скопированного текста на странице и кликов по кнопкам.

Глава 10. Настройка e-commerce. Глава содержит материалы по настройке интернет-магазина - стандартная электронная торговля, расширенная электронная торговля (Enhanced Ecommerce), динамический ремаркетинг Google, функция User ID.

Глава 11. «Продвинутая» настройка Google Tag Manager. В главе рассмотрены темы, которые могут быть сложны для понимания начинающим интернет-маркетологам, но которые могут оказаться полезными для специалистов с определенным уровнем знаний. Среди тем главы 11: прослушивание пользовательских событий, виртуальные страницы, файлы cookie, sessionStorage и localStorage, подмена контента на сайте, определение геолокации пользователя (включая IP-адрес), настройка междоменного отслеживания, отслеживание времени сеанса пользователя, счетчик просмотренных страниц и многое другое. Напоследок автор руководства делится списком плагинов и расширений для браузеров, которые он использует в своей работе.

Глава 1

Начало работы с диспетчером тегов Google

История возникновения систем управления тегами

Система управления тегами (Tag Management System, TMS) предназначена для управления тегами отслеживания, используемыми в digital-маркетинге. Она позволяет маркетологам и аналитикам устанавливать теги на веб-сайте, в мобильном приложении и управлять ими самостоятельно, сводя к минимуму привлечение сторонних разработчиков.

Теги (**англ. Tags**, иногда *пиксель* или *маячок*) — это средства для сбора данных и их обмена между вашим веб-сайтом, мобильным приложением и различными сервисами (например, рекламными инструментами, аналитическими платформами), которые вы используете в повседневной работе. Как правило, тег представляет из себя небольшой фрагмент кода на языке JavaScript, который добавляется на отслеживаемые страницы сайта.

Когда браузер запрашивает веб-страницу с сервера сайта, содержимое страницы вместе с кодом тега возвращается браузеру, код тега выполняется и осуществляется сбор данных. Помимо этого, тег указывает браузеру как отправить полученные данные на сервер сбора данных.

Теги используются для различных целей:

- сбора данных из веб-браузеров;
- взаимодействия между различными сайтами (например, для передачи демографических данных об аудитории);
- интеграции стороннего контента (виджеты социальных сетей, чаты, формы обратного звонка и т.д.);
- настройки файлов cookie и т.д.

Теги можно разделить на две группы в зависимости от функционального назначения:

1. **основные теги (first-party tags)**, которые устанавливают файлы cookie для сбора данных в одном домене и облегчают внутренний сбор данных;
2. **сторонние теги (third-party tags)**, которые устанавливают файлы cookie для сбора данных в стороннем домене, обычно на стороннем сайте или сайте поставщика (продавца).

Например, при посещении сайта **site.ru** тег, передающий информацию в **my.site.ru**, будет считаться основным тегом (first-party tags), в то время как тег из **my.website.ru** будет сторонним (third-party tags), поскольку домен верхнего уровня **website.ru** отличается.

Сегодня сайты могут использовать больше сотни различных тегов одновременно. В связи с этим появляется большая вероятность возникновения проблем, связанных со скоростью загрузки тегов, контролем и качеством сбора данных, общей производительностью сайта и т.д. Системы управления тегами (TMS) были разработаны как раз для решения множества этих проблем и упрощения работы с тегами как для технических, так и для нетехнических специалистов.

Ниже приведен краткий обзор истории тегов и того, как они превратились из пикселя отслеживания в сложную систему сбора данных.

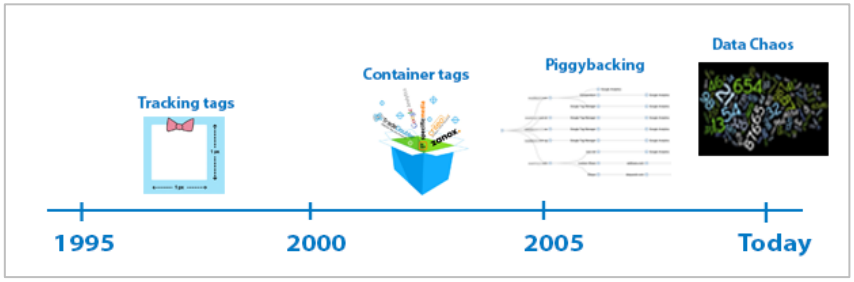


Рис. 1. Эволюция тегов

Сторонние теги (Third-Party Tags) – 90-ые

Первые теги появились примерно в 1995 году и их основной функцией было оказание помощи рекламным сервисам и аналитическим системам проведению оценки эффективности онлайн-маркетинговых кампаний для веб-сайтов. С развитием цифровых технологий все больше сторонних поставщиков услуг начали создавать собственные теги для сбора данных, требуемых для функционирования их проектов.

Все началось с простых пикселей отслеживания, которые представляли собой небольшие прозрачные изображения, невидимые посетителям сайта, которые загружались с сервера, на котором они размещены. Эти *теги-изображения* представляли собой **пиксель размером 1×1**, который отображался после осуществления определенного действия, например, отправки формы или совершения покупки. Запросы пиксельного изображения у сервера подсчитывались в целях отслеживания действий пользователей на сайте. Изменив его исходный HTML-код, пиксель можно было легко добавить на веб-страницу.

Постепенно теги превратились из простых пикселей в сложный JavaScript код, который стал собирать широкий спектр данных. Фрагмент JavaScript вставлялся в код страницы сайта для отслеживания действий пользователей. После загрузки страницы этот код выполнялся, затем он собирал большой объем данных и отправлял их в систему аналитики.

В результате упрощения реализации тегов и гибкости настройки, количество тегов резко увеличилось, а управлять тегами стало сложнее, поскольку в код сайта необходимо было добавлять коды от разных сервисов.

Тег контейнера (Tag Container) – 00-ые

Для решения проблемы добавления нескольких тегов на сайт и управления ими через единый интерфейс (в одном месте), в начале 2000-х годов крупные рекламные сети запустили **тег контейнера**. Он выполнял роль своеобразной оболочки вокруг нескольких тегов. Запуск одного тега приводил к запуску других тегов, содержащихся в этом контейнере. Контейнеры, представляющие собой механизм доставки тегов, были разработаны для упрощения добавления, изменения или удаления нескольких тегов.

Несмотря на все преимущества централизации тега контейнера, у него есть один главный недостаток. Когда на сайт добавлялся сторонний код, его производительность резко снижалась. Это стало поворотным моментом, который привел к эволюции систем управления тегами, благодаря которой стало возможно заменить все теги на сайте единым JavaScript кодом. Все замененные теги стали запускаться на базе правил, установленных пользователем в рамках системы управления тегами. В результате чего производительность сайта повысилась, а пользователь получил больший контроль над сайтом и тегами.

Теги piggybacking

При реализации нескольких тегов на сайте необходимо расширить диапазон сбора данных. Данная потребность была реализована в основном за счет увеличения расходов на онлайн-рекламу, а рекламные сети объединили теги рекламного сервера в цепочку для увеличения охвата аудитории на нескольких сайтах-издателей.

Объединение тегов в цепочку называется **piggybacking**. Принцип работы заключается в возможности одного сервера инициировать теги с другого сервера, которые затем по цепочке запускают теги третьего сервера и так далее. То есть теги запускают друг друга по цепочке. Например, на картинке ниже показано, как тег **AppNexus** использует стандартный тег **DoubleClick Floodlight** (см. приложение):

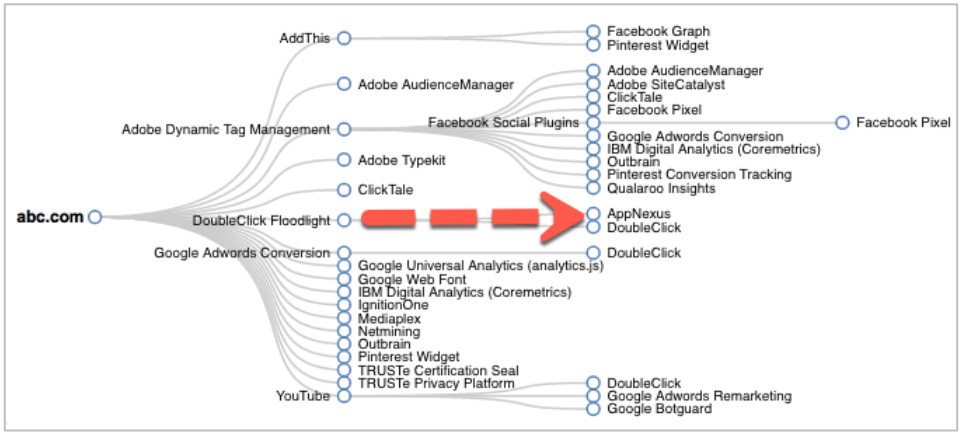


Рис. 2. Пример запуска тегов по цепочке (piggybacking)

Объединение тегов **piggybacking** расширил аудиторию сайтов, что привело к инновационным подходам к покупке и продаже рекламных объявлений, в том числе закупку трафика в реальном времени (**Real Time Bidding, RTB**) и обмен данными. Однако, это привело к хаосу данных (сегодня), так как владельцы сайтов обычно не могут контролировать теги по цепочке.

Стремительный рост маркетинговых тегов поставил на первый план проблему конфиденциальности с непреднамеренными передачами данных и утечки данных третьим сторонам в связи с тем, что владелец сайта не контролирует всю цепочку тегов. Помимо этого, затраты на инициирование тегов увеличиваются, что сказывается на производительности сайта и может привести к потере данных в результате проблем с загрузкой тегов.

Помимо этого, с помощью цепочки тегов на ваш сайт или на устройство вашего пользователя может быть добавлен вредоносный код, и получена личная информация клиента, которую он вам предоставляет.

2012: Google Tag Manager

Осенью 2012 года на саммите **eMetrics** в Бостоне компания Google анонсировала выпуск своего нового продукта – **Google Tag Manager**. По данным **similartech.com** (см. приложение) по состоянию на июль 2020 года он установлен на 5 600 000+ сайтов и является самой популярной системой по управлению тегами.

	Technology	Websites	
1	Google Tag Manager	5,634,736	Websites using Google Tag Manager
2	Adobe Dynamic Tag Management	85,518	Websites using Adobe Dynamic Tag Management
3	Tealium	46,304	Websites using Tealium
4	Yahoo Tag Manager	27,572	Websites using Yahoo Tag Manager
5	Ensighten	15,849	Websites using Ensighten
6	TagMan	567	Websites using TagMan
7	TagCommander	115	Websites using TagCommander
8	Mezzobit	59	Websites using Mezzobit

Рис. 3. Топ-8 систем управления тегами по данным similartech.com (июль 2020)

В топ-8 также входят: **Adobe Dynamic Tag Management (85,518), Tealium (46,304), Yahoo Tag Manager (27,572), Ensighten (15,849), TagMan (567), TagCommander (115)** и **Mezzobit (59)**.

Диспетчер тегов Google (менеджер тегов, тег менеджер, таг манагер, google tag, google tag manager, гугл тег), также известный как GTM (ГТМ), представляет собой бесплатный инструмент, который позволяет легко управлять тегами, размещенными на веб-сайте, в мобильном приложении (iOS, Android), а также на страницах **AMP (Accelerated Mobile Pages Project)**.

В качестве тегов могут быть:

- Код счетчика Google Analytics:

```
<!-- Global site tag (gtag.js) - Google Analytics -->
<script async src="https://www.googletagmanager.com/gtag/js?id=UA-77456218-1"></script>
<script>
  window.dataLayer = window.dataLayer || [];
  function gtag(){dataLayer.push(arguments);}
  gtag('js', new Date());

  gtag('config', 'UA-77456218-1');
</script>
```

Рис. 4. Код счетчика Google Analytics

- Код счетчика Яндекс.Метрики:

```
<!-- Yandex.Metrika counter -->
<script type="text/javascript" >
   (function(m,e,t,r,i,k,a){m[i]=m[i]||function(){(m[i].a=m[i].a||
[]).push(arguments)};
   m[i].l=1*new Date();k=e.createElement(t),a=e.getElementsByTagName(t)
[0],k.async=1,k.src=r,a.parentNode.insertBefore(k,a)})
   (window, document, "script", "https://mc.yandex.ru/metrika/tag.js",
"ym");

   ym(17776651, "init", {
        clickmap:true,
        trackLinks:true,
        accurateTrackBounce:true,
        webvisor:true,
        trackHash:true,
        ecommerce:"dataLayer"
   });
</script>
<noscript><div><img src="https://mc.yandex.ru/watch/17776651"
style="position:absolute; left:-9999px;" alt="" /></div></noscript>
<!-- /Yandex.Metrika counter -->
```

Рис. 5. Код счетчика Яндекс.Метрики

- Тег Google Рекламы:

```
<!-- Global site tag (gtag.js) - Google Ads: 940401511 -->
<script async src="https://www.googletagmanager.com/gtag/js?id=AW-940401511"></script>
<script>
  window.dataLayer = window.dataLayer || [];
  function gtag(){dataLayer.push(arguments);}
  gtag('js', new Date());

  gtag('config', 'AW-940401511');
</script>
```

Рис. 6. Тег Google Рекламы

- Facebook Pixel:

HTML

```
<!-- Facebook Pixel Code -->
<script>
  !function(f,b,e,v,n,t,s)
  {if(f.fbq)return;n=f.fbq=function(){n.callMethod?
  n.callMethod.apply(n,arguments):n.queue.push(arguments)};
  if(!f._fbq)f._fbq=n;n.push=n;n.loaded=!0;n.version='2.0';
  n.queue=[];t=b.createElement(e);t.async=!0;
  t.src=v;s=b.getElementsByTagName(e)[0];
  s.parentNode.insertBefore(t,s)}(window, document,'script',
  'https://connect.facebook.net/en_US/fbevents.js');
  fbq('init', '401069723650740');
  fbq('track', 'PageView');
</script>
<noscript><img height="1" width="1" style="display:none"
  src="https://www.facebook.com/tr?id=401069723650740&ev=PageView&noscript=1"
/></noscript>
<!-- End Facebook Pixel Code -->

```

Рис. 7. Код пикселя Facebook

- Код пикселя ВКонтакте:

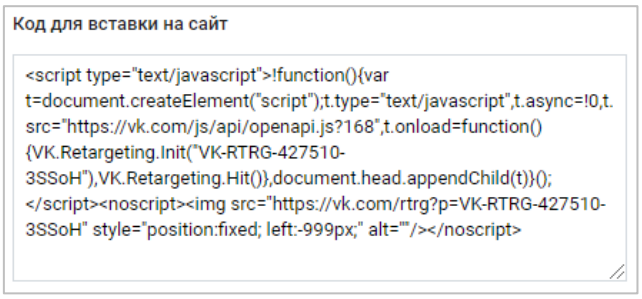


Рис. 8. Код пикселя ВКонтакте

- Код онлайн-чата JivoSite:

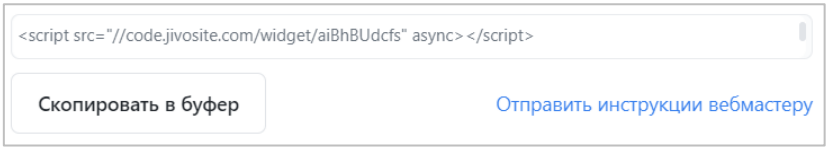


Рис. 9. Код онлайн-чата JivoSite

Сервисов, которые собирают информацию о пользователях посредством внедрения своего кода на сайт, очень много. Например, теги системы автоматизированной закупки рекламы **DoubleClick**, ретаргетинговой платформы **AdRoll**, сервисы по A/B тестированию и оптимизации конверсии **Visual Website Optimizer (VWO), Google Optimize**. В качестве дополнительного инструмента анализа аудитории владельцы сайтов используют **KISSmetrics**, **Hotjar**, **Adobe Analytics** и другие.

Сюда можно добавить установку тегов чатов, коллтрекингов, форм обратного звонка, сервисов мультиканальной и омниканальной аналитики. У каждого из них свой JavaScript код, который необходимо добавить на все отслеживаемые страницы сайта. А если ваш сайт еще является *статическим* (сайт, состоящий из неизменяемых HTML-страниц, связанных между собой ссылками), или не имеет отдельного файла с шапкой сайта (например, **header.php**), который подключается сразу ко всем страницам, то тогда все коды следует размещать вручную, дублируя их в **<head>** на каждой странице. Такая работа отнимает много времени и никому не понравится.

Принцип работы диспетчера тегов Google

Что же приходится делать владельцам сайтов, когда они хотят внедрить очередной код какого-либо сервиса к себе на сайт или внести изменения в существующий без использования Google Tag Manager? Как минимум писать разработчику ТЗ с подробными инструкциями того, куда нужно вставить код, а как максимум - внедрять его на сайт самостоятельно. При такой последовательности действий есть большая вероятность:

- самому ошибиться с внедрением различных кодов в силу незнания правил и разметки веб-страниц;
- получить от программиста перечень пунктов, которые были ему непонятны из ТЗ. В этом случае начнется игра в настольный теннис: я сделал все согласно ТЗ, но не работает. Присылайте новое ТЗ;
- сорвать все сроки и дедлайны из-за долгой обратной связи. Вытекает из предыдущего пункта.

Такой вариант внедрения можно представить в виде последовательности шагов:

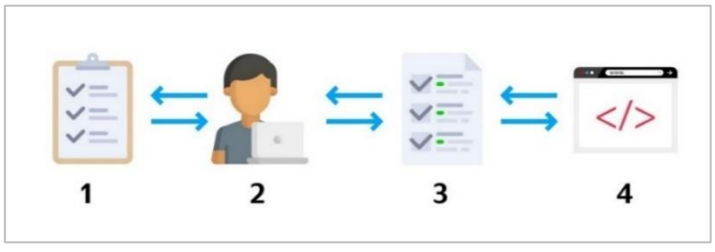


Рис. 10. Внесение изменений в код сайта без использования Google Tag Manager

1. подготавливается ТЗ из некоторого количества пунктов;
2. список передается разработчику, который просматривает его;
3. если у него нет вопросов, то он внедряет эти пункты. В противном случае мы возвращаемся на шаг 1;
4. код устанавливается на сайт. Если по каким-то причинам это было сделано некорректно, все этапы придется начинать сначала (подготавливать ТЗ, назначать тикеты программисту, писать письма, отправлять фрагменты кода и т.д.).

Согласитесь, очень долгий и трудозатратный процесс. При установке контейнера Google Tag Manager процесс сводится к следующему:

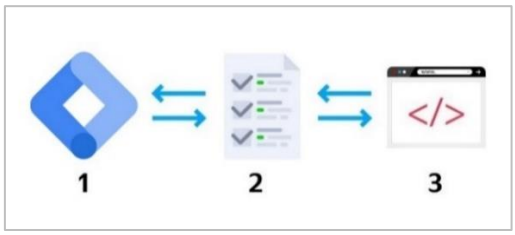


Рис. 11. Внесение изменений через Google Tag Manager

1. устанавливается код Google Tag Manager один раз;
2. внедряются изменения на сайт через рабочую область GTM без прибегания к помощи со стороны;
3. производится отладка всех процессов и публикуется рабочая версия тегов.

Когда Google выпускал свой продукт больше 8 лет назад, он хотел решить ряд задач, в числе которых:

- ускорение загрузки страниц и работоспособности сайтов путем объединения тегов в одном удобном инструменте;
- экономия времени разработчиков, маркетологов и веб-аналитиков;
- избегания дублирования и ошибок в работе тегов;
- снижение необходимости изменения исходного кода сайта при обновлении или добавлении тегов;
- завоевание доли рынка благодаря бесплатности GTM (да, без этого никуда).

В качестве недостатка Google Tag Manager, да и вообще всех диспетчеров тегов, можно отметить зависимость от объектной модели документа **(Document Object Model, DOM)** - верстки или исходного кода страниц. Поскольку все операции выполняются с привязкой к различным идентификаторам, атрибутам и классам элементов на веб-странице, то в случае их изменения, сделанные ранее настройки могут перестать работать.

Несмотря на то, что с внедрением Google Tag Manager наша зависимость от разработчиков существенно снизилась, отказаться полностью от их помощи все же не удастся. Есть ряд задач, которые по-прежнему будет необходимо решать вместе с программистом. Сюда входит:

- фиксация транзакций;
- настройка User ID;
- отслеживание динамического ремаркетинга;
- добавление пользовательских параметров и показателей;
- внедрение уровня данных (dataLayer);
- другие задачи.

Теоретически, мы можем настроить практически все через диспетчер тегов Google, используя при этом технологию извлечения данных со страниц сайта (так называемый **web scraping**). Но чтобы что-то извлекать со страницы сайта, эта информация должна присутствовать на ней. В случае ее отсутствия также придется обращаться к стороннему специалисту.

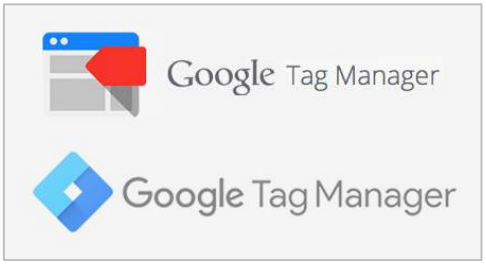


Рис. 12. Старый (сверху) и новый (внизу) логотипы Google Tag Manager

Таким образом, из преимуществ Google Tag Manager можно выделить:

- **бесплатный инструмент** – лидер рынка;
- **экономит время** - не нужно искать разработчиков, которые внедрят изменения на сайт и нет необходимости ждать последующих правок, если первоначальные были сделаны с ошибками;
- **снижает зависимость от разработчиков**;
- **позволяет управлять тегами в едином пространстве** - не придется писать дополнительный код или вносить изменения в код отслеживания, вся работа выполняется через веб-интерфейс;
- **средства предотвращения ошибок** - режим предварительного просмотра (чтобы вы могли видеть предлагаемые изменения перед их внедрением);
- **работает быстро благодаря асинхронной загрузке тегов** – одновременная (параллельная) загрузка тегов, в результате которой более медленные загружающиеся теги никак не повлияют на скорость выполнения других, более быстрых тегов.

Google Tag Manager состоит из 3 основных сущностей: *переменных (что извлекать, находить?), триггеров (когда и где?)* и *тегов (куда передавать данные?)*. С помощью **встроенных и пользовательских переменных** мы можем создавать условия для активации триггеров, находить нужную информацию на странице сайта, и извлекать ее для последующей передачи в инструменты веб-аналитики. В переменных мы можем хранить различные данные – Client ID, User ID, источник перехода, URL страницы, ID заказа, сумму транзакции и т.д. Они могут применяться как в триггерах, так и в тегах.

Триггеры – это условия активации или блокировки для тега. Они применяются для осуществления какого-либо события, происходящего на сайте. Например, мы хотим показывать определенное сообщение для пользователей, которые зашли на сайт с мобильного устройства. Для того, чтобы это сделать, необходимо создать сначала переменную, которая осуществляла бы проверку на тип устройства, а затем использовать триггер, который проверяет, зашел ли пользователь с mobile или desktop. Если с мобильного устройства, тогда триггер срабатывает и активируется тег, который показывает наше сообщение пользователю. А если нет, то триггер не срабатывает, тег не активируется, и пользователь не видит специального сообщения. Условия активации могут быть самым разными – по времени, по количеству просмотренных страниц, по отслеживанию определенного элемента на странице и т.д.

Теги — это фрагменты JavaScript кода, которые собирают данные о посетителях на сайте и в приложении, а затем пересылают их на сторонние сервисы — Google Analytics, Google Ads, Facebook, Яндекс.Метрику и т.д. У каждого тега должен быть хотя бы один триггер.

Таким образом, при использовании комбинации **переменная – триггер – тег** интернет-маркетолог может отслеживать различные события на сайте, создавать при этом необходимые правила (условия), с помощью которых необходимая информация будет передаваться в инструменты веб-аналитики.

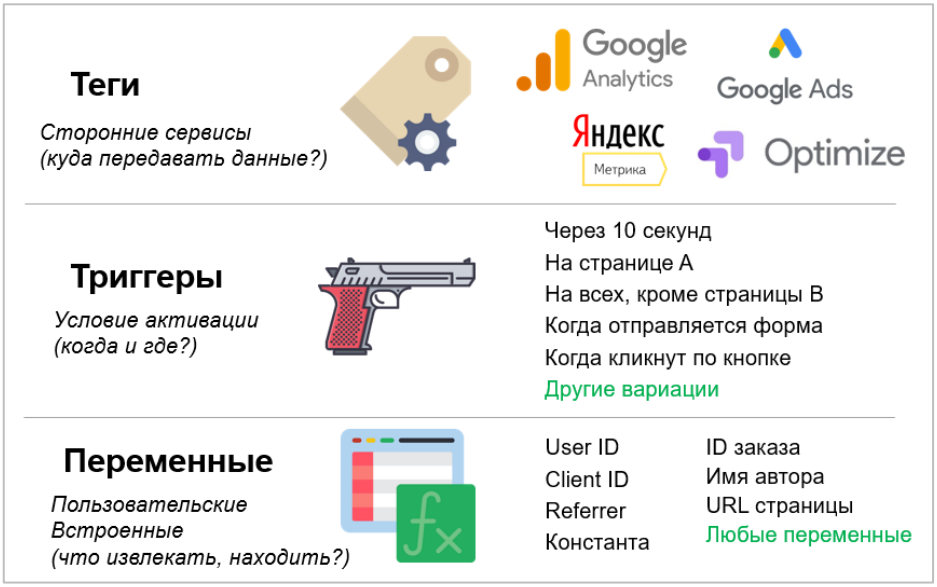


Рис. 13. Три основные сущности в Google Tag Manager

Более подробно переменные, триггеры и теги Google Tag Manager разберем в соответствующих главах.

Навыки и знания, необходимые в работе

Чтобы максимально использовать функционал системы управления тегами, желательно знать основы HTML, CSS и язык программирования JavaScript, хотя бы на уровне чтения кода. Именно в такой последовательности лучше всего начинать обучение: **1. HTML; 2. CSS; 3. JavaScript**.

Рис. 14. HTML, CSS, JavaScript и jQuery

HTML (от англ. HyperText Markup Language — язык гипертекстовой разметки) — стандартизированный язык разметки документов в интернете. Большинство веб-страниц содержат описание разметки на языке HTML (или XHTML). Язык HTML интерпретируется браузерами; полученный в результате интерпретации форматированный текст отображается на экране монитора компьютера или мобильного устройства.

Затем вы можете перейти к изучению CSS, и как использовать их для оформления HTML-документа (например, изменить шрифт и его размер, добавить границы и тени, разметить страницу в несколько столбцов, добавить анимацию и другие визуальные эффекты).

Один из самых важных механизмов CSS - **селекторы**. Это формальное описание того элемента или группы элементов, к которым применяется указанное правило стиля. В процессе работы с GTM вы будете очень часто сталкиваться с ситуацией, когда необходимо настроить отслеживание клика по какому-либо элементу на веб-странице, а у этого элемента нет идентифицирующих его атрибутов, таких как, например, **ID**, **class** или **URL**. Тогда на помощь приходят CSS-селекторы.

Рекомендуемая литература: htmlbook.ru книга Влада Мержевича HTML и CSS на примерах.

Как правило, счетчики веб-аналитики представляют собой фрагмент JavaScript кода, который размещается на отслеживаемых страницах сайта. Код Google Analytics, Google Tag Manager, Яндекс.Метрики и других сервисов – в основе всех их лежит JavaScript (JS). **Не путайте его с другим языком программирования Java!**

Работая с переменными в GTM, целесообразно разобраться с типами данных, чем они отличаются друг от друга, как происходит их преобразование, как они сравниваются между собой, какие бывают операторы, какой синтаксис используется при их написании и многое другое. На JavaScript также можно писать скрипты для Google Ads для автоматизации определенных действий.

Если вы на 80% специалист по рекламе (тот, кто больше времени настраивает) и только на 20% веб-аналитик (тот, кто больше времени работает с отчетностью, анализирует), который хочет подтянуть свои знания в работе с Google Tag Manager, полное погружение в освоение JavaScript необязательно. Достаточно понимать используемый код хотя бы на уровне чтения. Для этого хватит знаний, которые представлены в этой книге.

Не забывайте, что порог входа в нашу профессию с каждым годом растет, появляется все больше вакансий веб-аналитиков, продуктовых аналитиков, дата-сайентистов, интернет-маркетологов со знанием языков программирования (JavaScript, Python, R), предлагается множество курсов и программ обучения. Настраивать контекстную, таргетированную или любую другую рекламу уже не является какой-то сверхсложной или уникальной задачей.

Разработчик, который владеет языками программирования, также может пройти обучение и научиться запускать рекламные кампании. Но не каждый интернет-маркетолог, прошедший курсы по программированию, сможет написать программу или какой-нибудь простой обработчик. Другой вопрос, что каждый должен заниматься своим делом. А чтобы оставаться востребованным специалистом ваши навыки должны быть на порядок выше, чем у ваших оппонентов. Другими словами, при прочих равных на позицию веб-аналитика возьмут человека, который знает JavaScript. Именно поэтому так важно в 2020 году приступить к изучению одного из самых популярных языков программирования в мире.

Помимо самого JS, не будут лишними знания **jQuery**. jQuery — библиотека JavaScript, фокусирующаяся на взаимодействии JavaScript и HTML. Она помогает легко получать доступ к любому элементу DOM, обращаться к атрибутам и содержимому элементов DOM, манипулировать ими.

Рекомендуемая литература: learn.javascript.ru**,** Крис Минник и Ева Холланд «JavaScript для чайников», Марейн Хавербек «Выразительный JavaScript», Эрик Фримен и Элизабет Робсон «Изучаем программирование на JavaScript», Антон Шевчук Учебник «jQuery для начинающих».

Темы, которые также могут вызвать затруднения у начинающих пользователей, но без которых работа с Google Tag Manager будет не такой гибкой и полноценной:

- принцип работы счетчиков веб-аналитики;
- файлы cookie, localStorage, sessionStorage;
- регулярные выражения;
- извлечение данных со страницы (DOM Scraping);
- виртуальные страницы;
- уровень данных (dataLayer).

Все это мы подробнее разберем в последующих главах книги.

Регистрация аккаунта

Для того чтобы начать работу с инструментом Google Tag Manager, необходимо зарегистрировать обычную почту в Gmail.com.

Если вы когда-либо работали с другими продуктами Google и уже имеете почту, то заводить отдельную не требуется. Авторизовавшись в аккаунте Google, просто перейдите на сайт **tagmanager.google.com** или на обновленную англоязычную версию **marketingplatform.google.com/about/tag-manager**

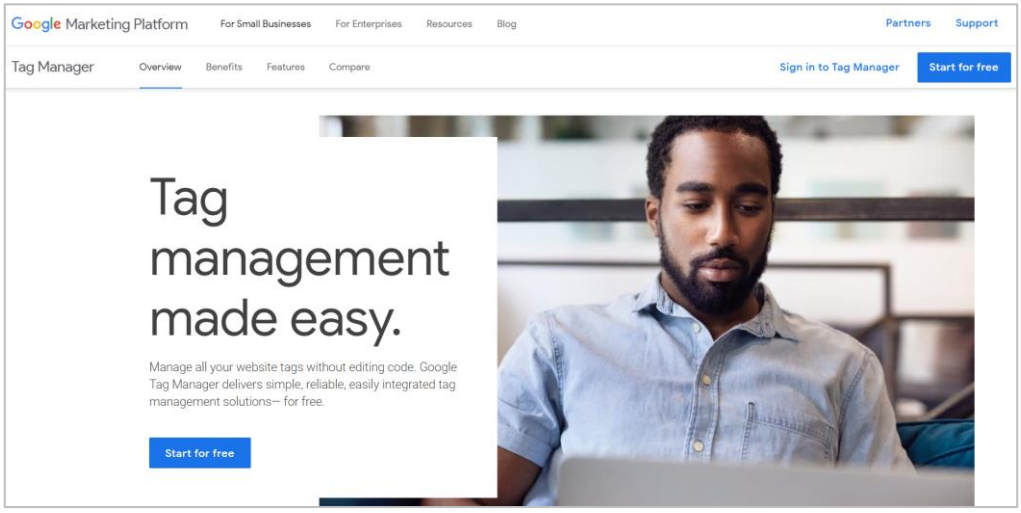


Рис. 15. Google Marketing Platform – Tag Manager

Создание контейнера GTM состоит из двух этапов:

1. ввод названия аккаунта (лучше всего вводить простое, запоминающееся название, которое вы с легкостью сможете найти, если в учетной записи этих аккаунтов станет много) и выбора страны;

Также Google просит вас разрешить передавать основные данные в свои продукты анонимно.

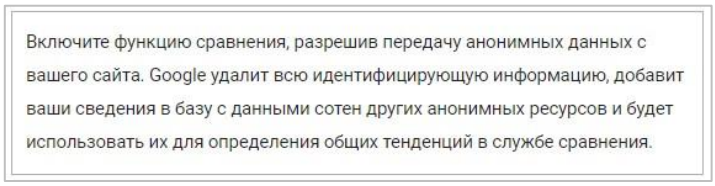


Рис. 16. Передача анонимных данных в Google и другие службы

Можете ставить галочку, можете нет – по желанию. На работоспособность и функции диспетчера тегов это не влияет.

2. ввод названия контейнера и тип использования: **Веб-сайт, Мобильные приложения на iOS / Android, AMP-страницы, Server (BETA)** - новый способ отслеживания данных на стороне сервера **(Server-Side Tracking)**, который является альтернативой привычному методу отслеживания на стороне клиента **(Client-Side Tracking)**.

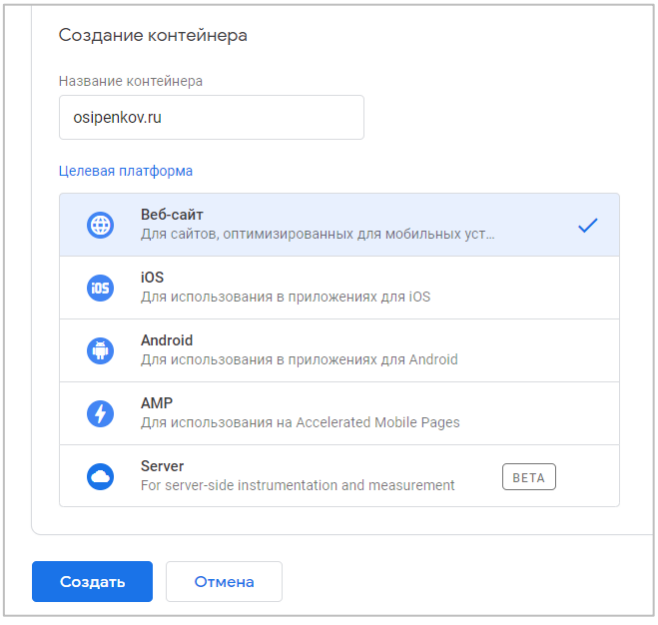


Рис. 17. Добавление нового аккаунта

За основу структуры аккаунт-контейнер вы можете взять традиционные аккаунт-ресурс из Google Analytics, где аккаунт – это верхний уровень доступа, а контейнер — ресурс из Google Analytics, то есть домен сайта или мобильное приложение. Чаще всего используют два подхода при именовании:

1. **аккаунт** – название компании, сотрудник, ответственный (например, Михаил), группа сайтов (общее название);
2. **контейнер** – конкретные сайты или мобильные приложения;

Вот как может выглядеть структура аккаунтов и контейнеров и двух разных проектов:

1. аккаунт = контейнер;
2. аккаунт – название компании, контейнеры – сайты по регионам;

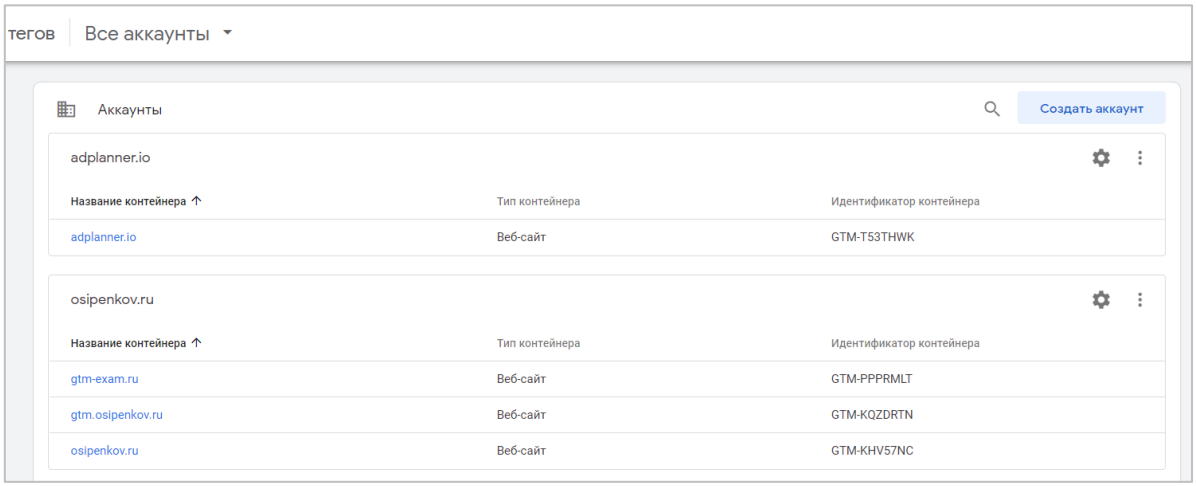


Рис. 18. Пример структуры в диспетчере тегов Google

Вы в любой момент потом можете поменять как название аккаунта, так и название контейнера. После их ввода нажмите **Создать**. Далее внимательно читаем соглашение об условиях использования диспетчера тегов Google, при желании переключив на русский язык. Если все устраивает (а другого и быть не может), то нажмите **Да** в правом верхнем углу.

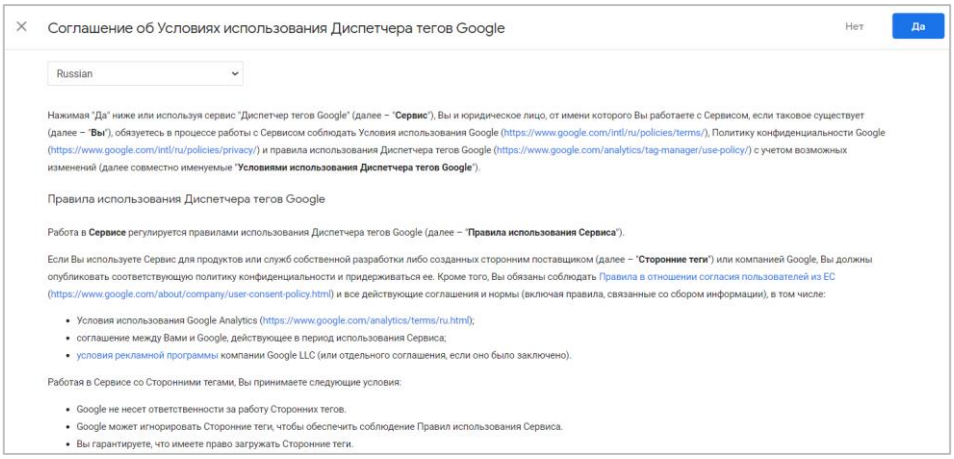


Рис. 19. Соглашение об Условиях использования Диспетчера тегов Google

Установка контейнера

После перезагрузки страницы вам станет доступен основной интерфейс GTM. Но перед полноценным использованием всех его возможностей необходимо установить Tag Manager на каждую страницу сайта в раздел **<head>**, а часть кода после открывающего тега **<body>**. На экране появится соответствующий код:

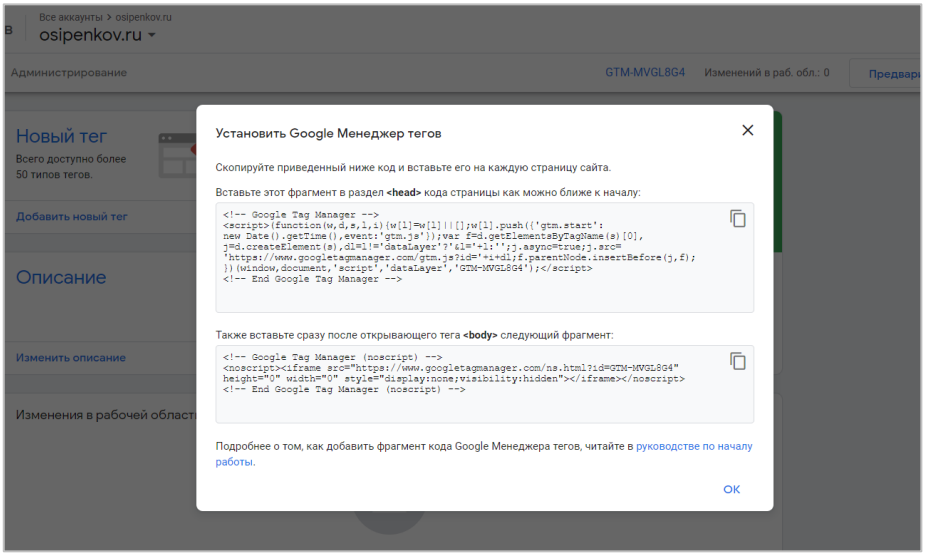


Рис. 20. Код установки Google Tag Manager

Если вы используете готовые CMS-движки, то в 99% случаев на рынке существует готовые решения, где с помощью установки дополнительного модуля или плагина можно исключить ручную вставку кода на сайт. Например, у движка интернет-магазина на CS-Cart есть готовый модуль, благодаря которому весь процесс настройки GTM сводится к вставке ID контейнера в соответствующее поле:

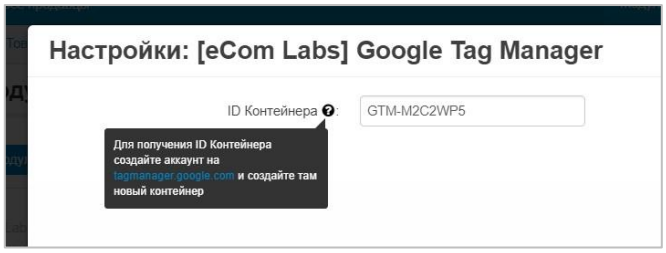


Рис. 21. Вставка ID контейнера

После добавление кода GTM не забудьте опубликовать контейнер, нажав на кнопку **Отправить — Опубликовать**.

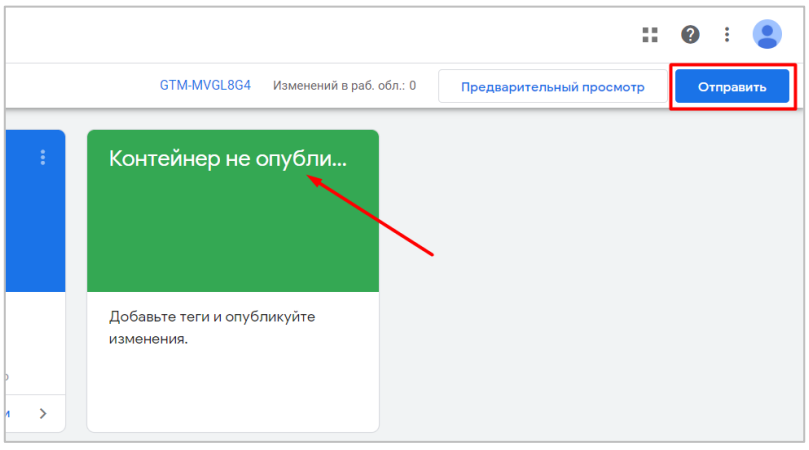


Рис. 22. Публикация контейнера

До публикации ваш контейнер будет возвращать ответ 404. В **Google Tag Assistant** это будет выглядеть так: *HTTP response code indicates tag failed to fire: Status 404. This could be due to an empty or un-published container*

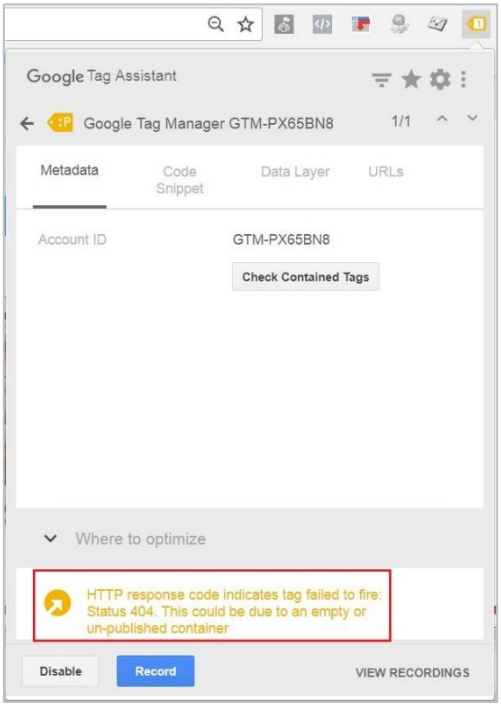


Рис. 23. Google Tag Assistant. Ошибка 404

Если все сделаете правильно, увидите опубликованную версию и зеленую иконку в GTA:

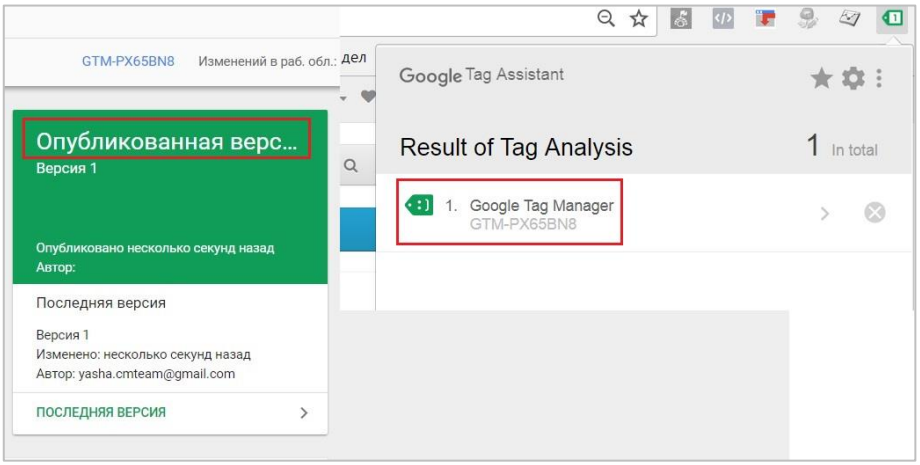


Рис. 24. Корректная установка кода контейнера на сайт

Глава 2

Интерфейс Google Tag Manager

Обзор

В процессе знакомства с любым новым для себя продуктом у пользователей часто возникает проблема привыкания к самому сервису, программе или ее отдельным функциям. Будь-то это совершенно новый интерфейс Google Ads, обновленный Google Analytics, или Google Tag Manager версии 2.

Вот так выглядит страница контейнера диспетчера тегов Google одного из аккаунтов:

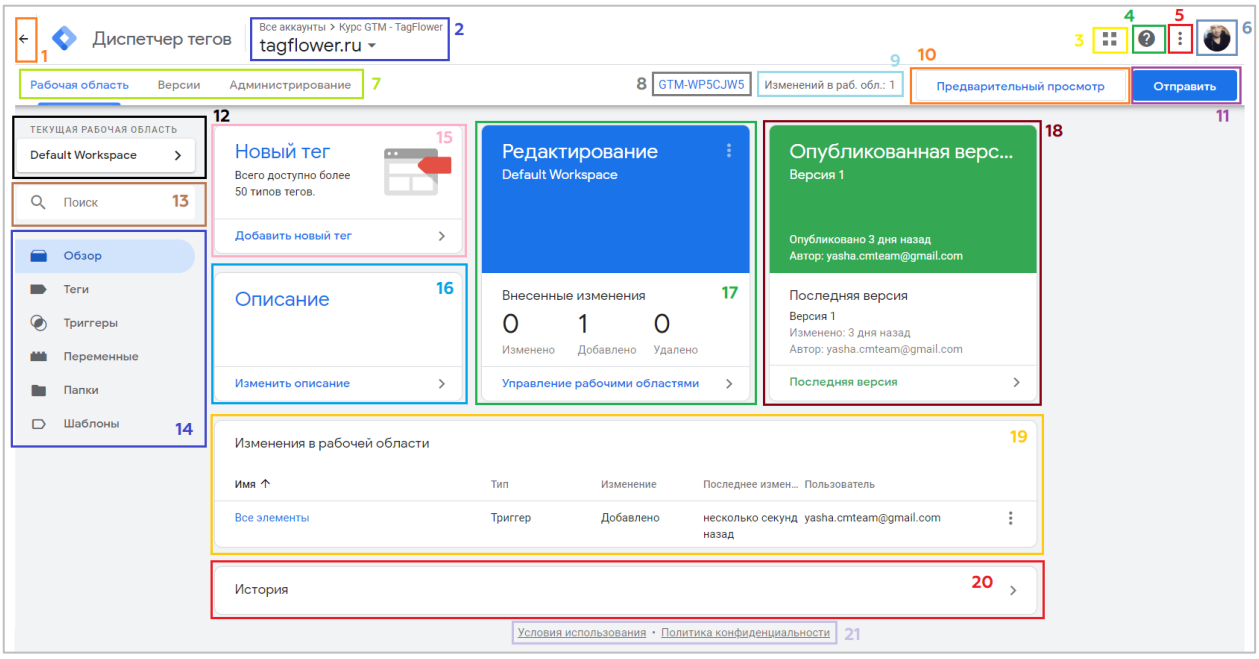


Рис. 25. Интерфейс Google Tag Manager

1. На главную страницу

При клике на данную стрелочку вы попадете на страницу **tagmanager.google.com/#/home**, где будут отображены все доступные вам контейнеры диспетчера тегов Google.

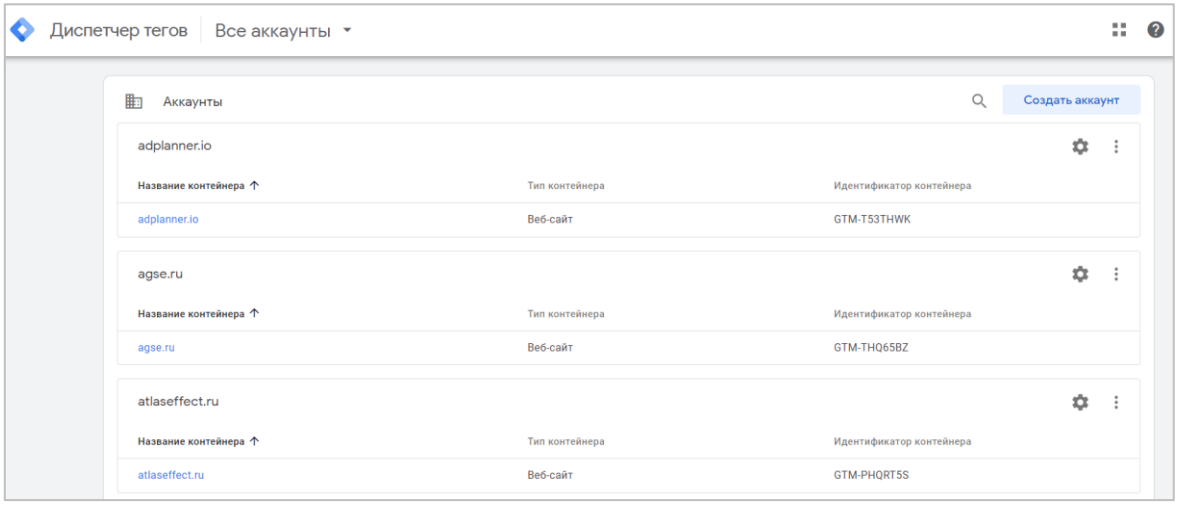


Рис. 26. Список аккаунтов и контейнеров

На главной странице возможно создание нового аккаунта, фильтра по текущим, а также доступны общие настройки по каждому конкретному контейнеру. Фильтр доступен как по названию контейнера, так и по его идентификатору. Когда вы начнете вводить последовательность букв/цифр, Google Tag Manager сразу же выдаст соответствующий результат.

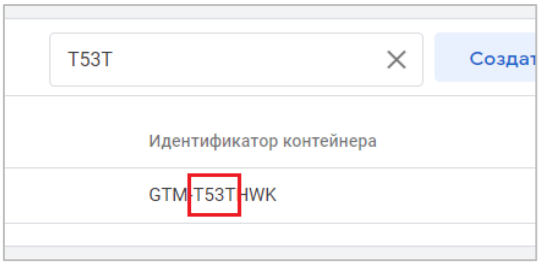


Рис. 27. Поиск по идентификатору контейнера

На главной странице в самом низу располагаются удаленные аккаунты и контейнеры. Чтобы удалить аккаунт или контейнер, необходимо обладать соответствующими правами.

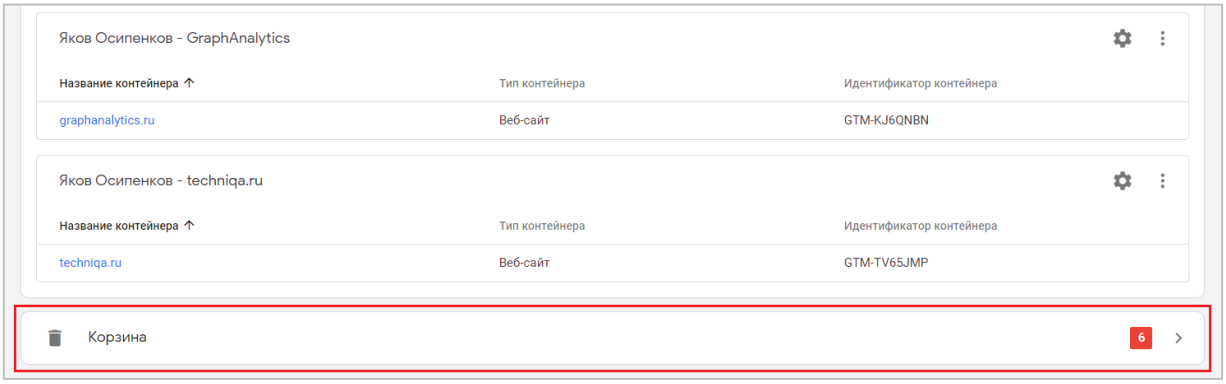


Рис. 28. Удаленные аккаунты и контейнеры в Google Tag Manager

При удалении аккаунта через **Администрирование** в GTM он будет перемещен в корзину и окончательно удален через 30 дней. Вместе ним будут удалены все контейнеры, включая все теги, триггеры и переменные. Теги из этого контейнера больше не будут активироваться при посещении вашего сайта пользователями. При удалении контейнера он также будет перемещен в корзину и окончательно удален через 30 дней.

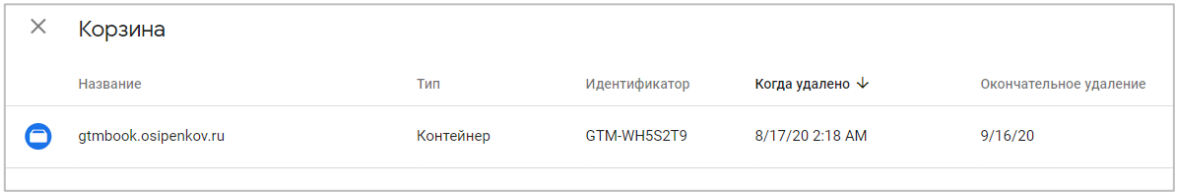


Рис. 29. Корзина

По клику на контейнер мы можем узнать точную дату удаления. По прошествии 30 дней данного контейнера в корзине уже не будет, также, как и возможности его восстановления. Если же нам понадобиться восстановить элемент из корзины, мы должны:

- выбрать его из списка;
- в открывшемся окне нажать кнопку **Восстановить**.

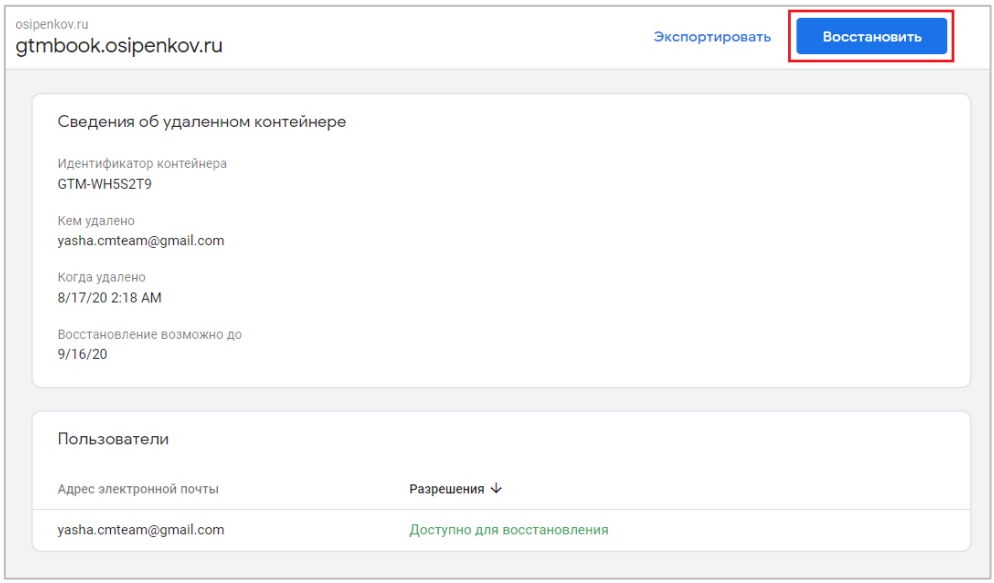


Рис. 30. Сведения об удаленном аккаунте

Еще нам будет доступна возможность экспорта контейнера и информация о пользователях, у которых есть разрешение на восстановление, а также электронная почта того человека, кто совершил операцию удаления (поле Кем удалено).

2. Аккаунты и сервисы Google

В одном меню Google объединил все свои продукты, и теперь доступ к ним стал возможен через одну панель. Если вы захотите перейти в Google Analytics, Google Data Studio или любой другой продукт, который привязан к данному аккаунту Google, просто нажмите на соответствующий значок на панели **Все аккаунты**.

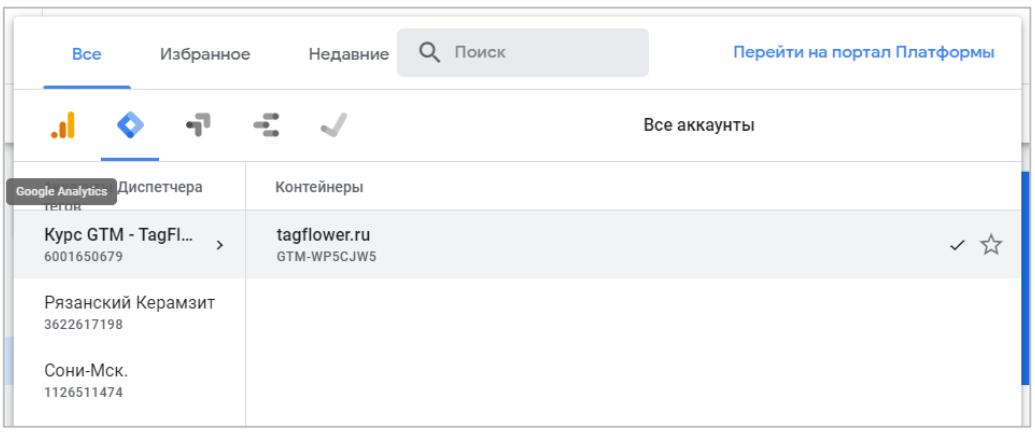


Рис. 31. Google Analytics Suite

3. Google Marketing Platform

Google Marketing Platform – это объединенная рекламно-аналитическая платформа, которая упрощает взаимодействие между маркетинговыми командами благодаря интеграции продуктов DoubleClick и Google Analytics 360 Suite.

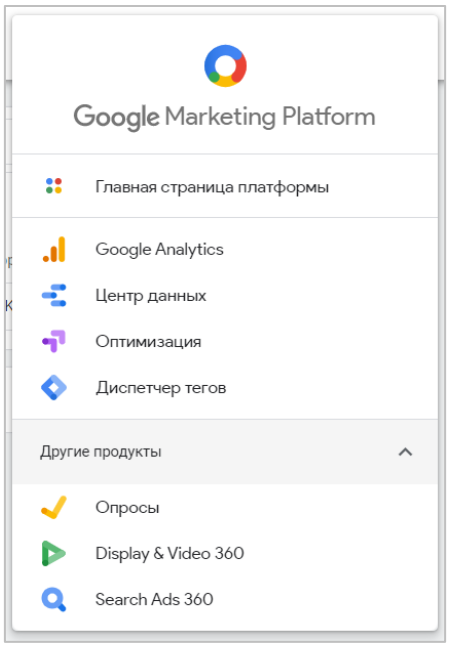


Рис. 32. Google Marketing Platform

Меню, в котором отображаются продукты Google из Google Платформы для маркетинга.

4. Справка Google Tag Manager

Доступны как все материалы о GTM, так и наиболее популярные статьи о диспетчере тегов Google.

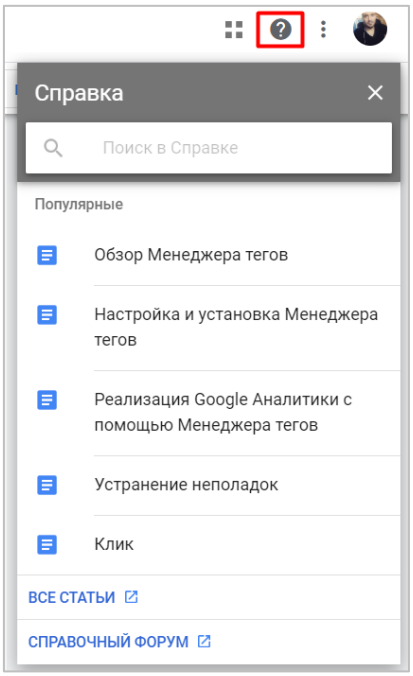


Рис. 33. Справка Google Tag Manager

Если у вас остались вопросы, вы всегда можете обратиться за помощью в официальное сообщество Google (Tag Manager Help Community), просто перейдя по последней ссылке в справочный форум.

5. Дополнительные настройки

Меню состоит из пользовательских настроек и отзыва.

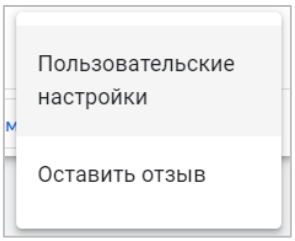


Рис. 34. Дополнительные настройки

В пользовательских настройках содержится информация по вашему роду деятельности (агентство, маркетолог, веб-разработчик), языку интерфейса Google Tag Manager и уведомлениях, связанных с новостями, анонсами новых функций, тестированию и различными предложениями от Google.

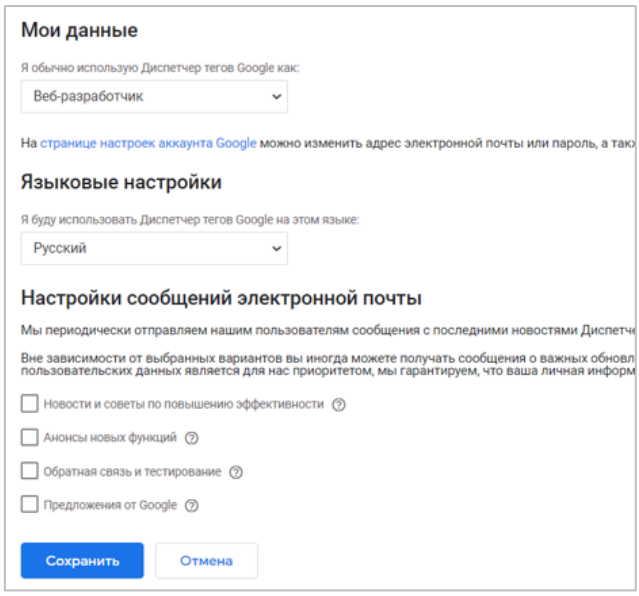


Рис. 35. Данные аккаунта

В разделе **Оставить отзыв** вы можете описать проблему работы с GTM или поделиться мнением с командой Google об этом инструменте.

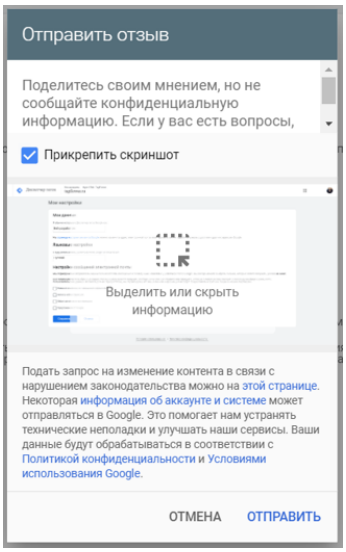


Рис. 36. Оставить отзыв

6. Аккаунты

Можно быстро переключаться между аккаунтами при условии, что вы залогинены в несколько из них.

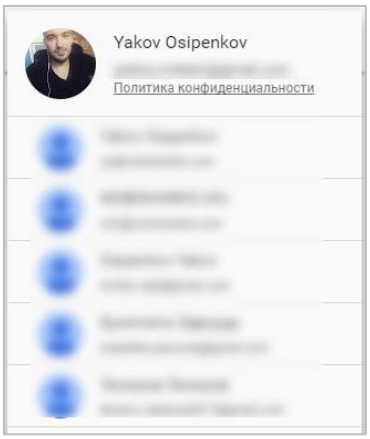


Рис. 37. Несколько аккаунтов Google

7. Горизонтальное навигационное меню

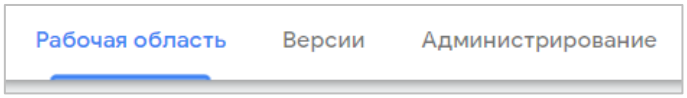


Рис. 38. Рабочая область — Версии — Администрирование

Позволяет быстро переключаться между рабочими областями, версиями и панелью администрирования аккаунта и контейнера.

По умолчанию активна первая вкладка **Рабочая область** (пространство для работы с изменениями в диспетчере тегов), но в любой момент можно перейти на **Версии** (сохраненная копия контейнера в данный момент времени) или **Администрирование**.

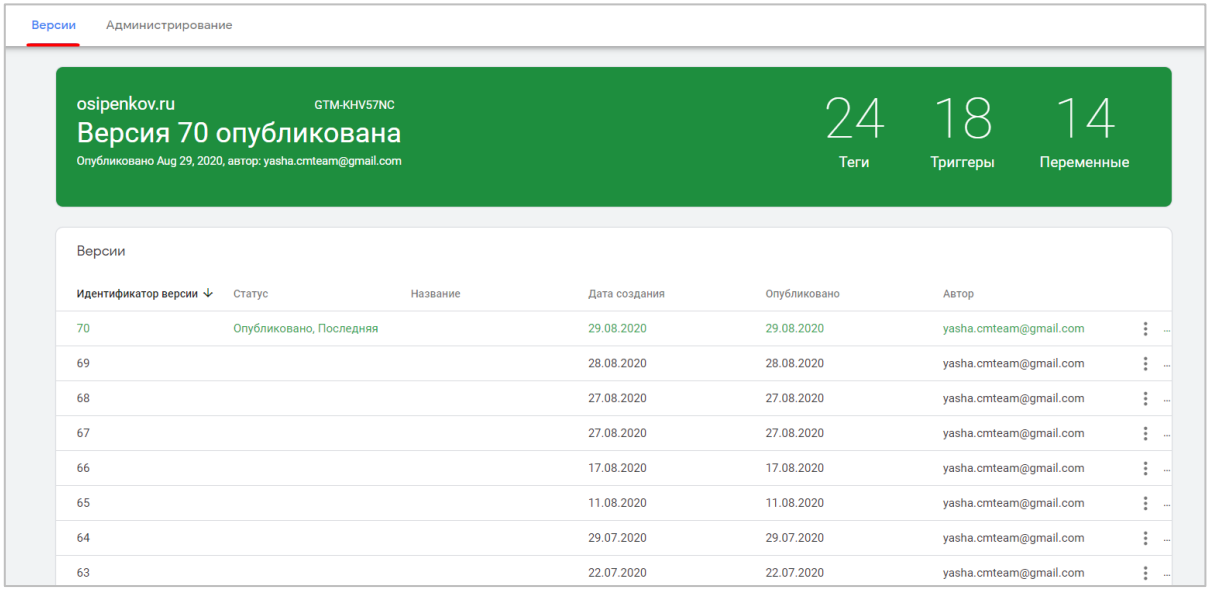


Рис. 39. Версии

На вкладке **Администрирование** можно управлять доступами, изменять настройки, на уровне аккаунта и контейнера, импортировать-экспортировать контейнеры, работать со средами и др.

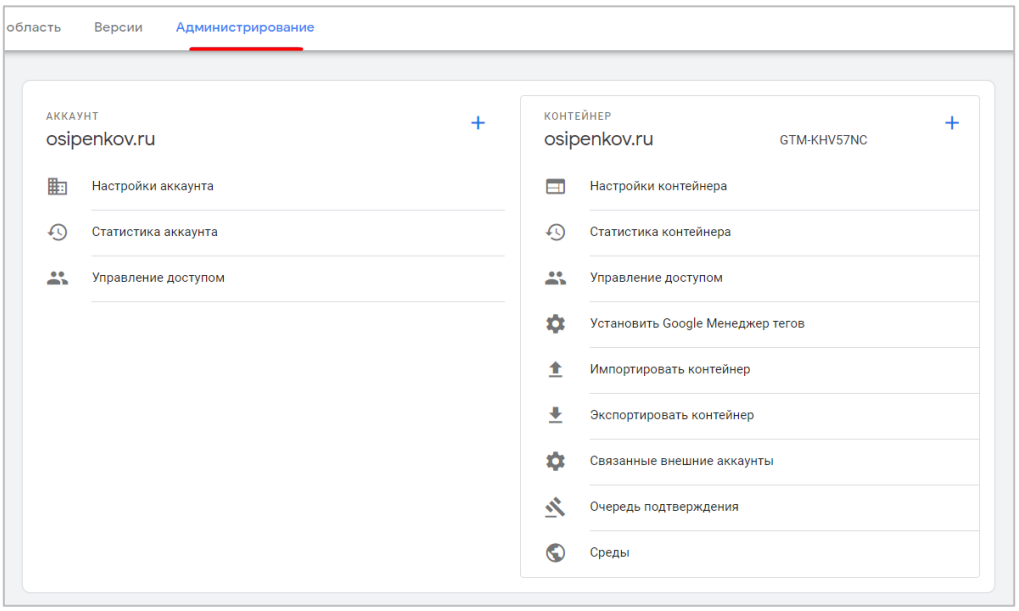


Рис. 40. Администрирование

8. Идентификатор контейнера

При клике на ID контейнера Google Tag Manager откроется окно с кодом, часть которого необходимо вставить в раздел **<head>** кода страницы как можно ближе к началу, а часть после открывающего тега **<body>** при условии, что вы этого не сделали ранее. Тот же самый код вы видели на этапе регистрации в GTM и установки контейнера.

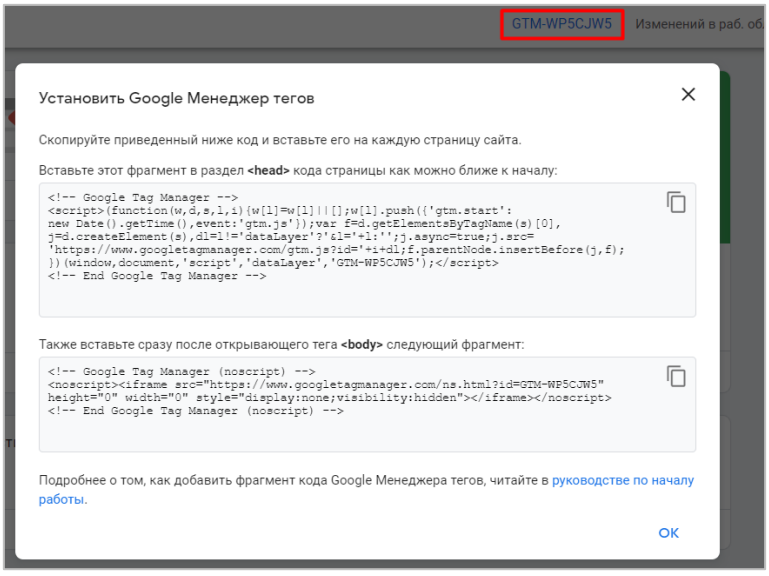


Рис. 41. Фрагменты кода Google Tag Manager

9. Изменения в рабочей области

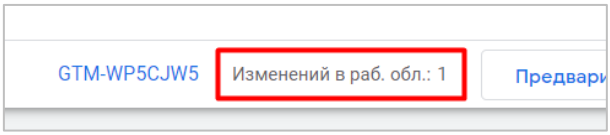


Рис. 42. Изменения в рабочей области

Счетчик отображает количество внесенных изменений в текущей рабочей области. Например, вы создали новую переменную. Изменений в рабочей области станет 1. Добавили новый триггер. Значение изменится на

2. Еще два триггера? Тогда изменений в рабочей области будет 4. Создали тег? Google Tag Manager зафиксирует еще одно изменение в рабочей области и т.д. Счетчик обнуляется после публикации версии.

10. Предварительный просмотр

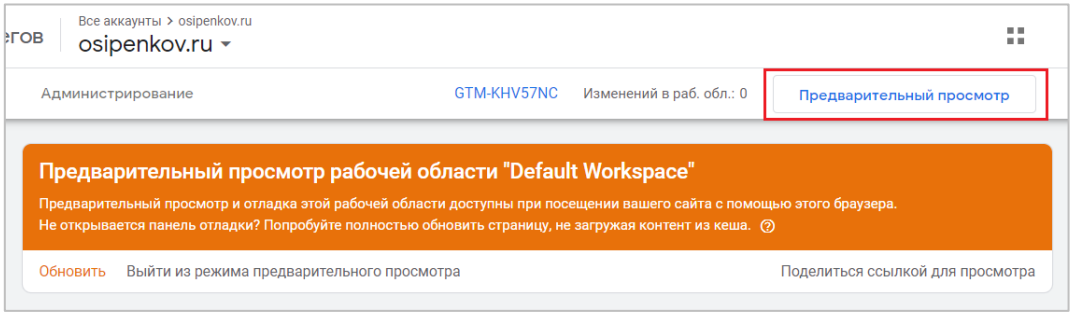


Рис. 43. Предварительный просмотр

Режим предварительного просмотра в GTM позволяет проверить работоспособность сайта с действующим контейнером. При каждом изменении конфигурации, корректировки настроек тегов, триггеров или переменных, рекомендуется проверять себя через отладку контейнера. Благодаря предварительному просмотру вы сможете избежать многих ошибок в процессе работы и узнать, какие теги активируются и в каком порядке.

Если включить режим предварительного просмотра, то на обзорной странице рабочей области вы увидите оранжевый баннер с оповещением (на изображении выше), а на сайте станет доступна панель отладки. Она будет видна только в том браузере, в котором был включен этот режим, но вы можете предоставить доступ другим пользователям.

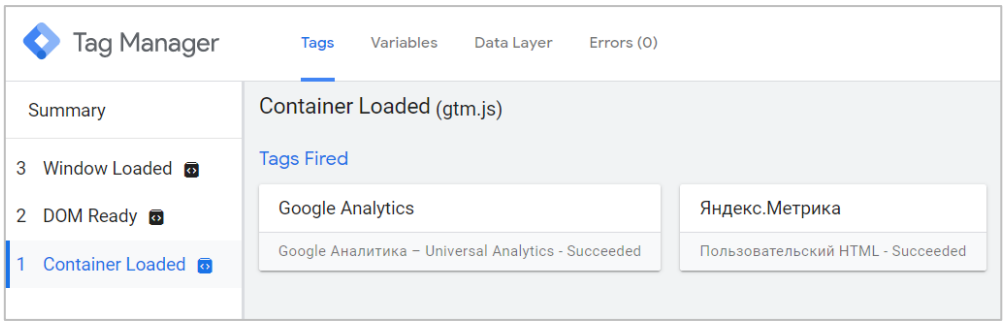


Рис. 44. Нижняя панель отладки

Сделать это можно с помощью опции **Поделиться ссылкой для просмотра**.

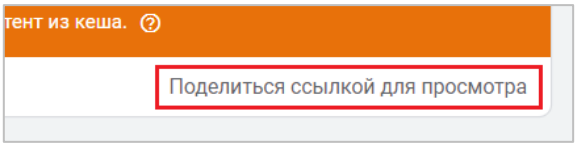


Рис. 45. Поделиться ссылкой для просмотра

Режим предварительного просмотра доступен пользователям с соответствующими правами.

11. Публикация контейнера

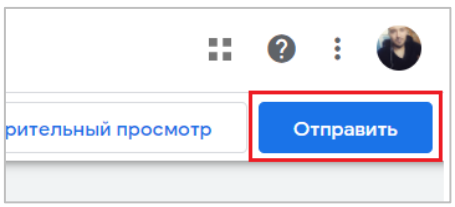


Рис. 46. Публикация контейнера

Чтобы добавить на сайт контейнер GTM впервые или отредактированную версию, его необходимо опубликовать. Сделать это можно через последовательность шагов по кнопке **Отправить - Публикация**. Опубликовать контейнер могут пользователи с соответствующими правами доступа в одну из указанных сред.

12. Рабочая область

Рабочая область – это пространство для работы с изменениями в диспетчере тегов. Их можно сравнить с представлениями в Google Analytics.

По умолчанию, после создании контейнера GTM добавляется рабочая область с названием **Default Workspace**. В соответствующем меню можно выбрать одну из доступных рабочих областей, либо же создать новую через **+** и добавить (по желанию) описание.

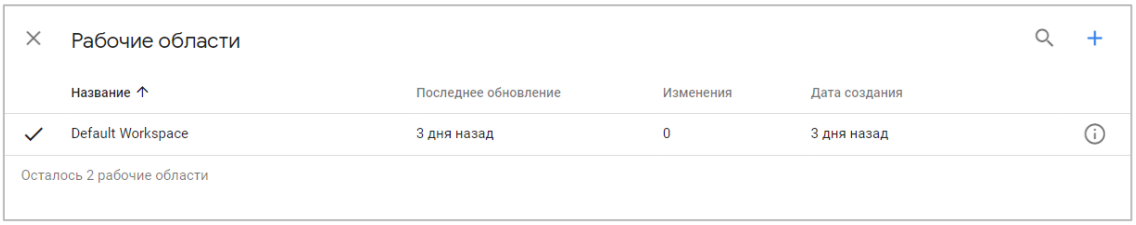


Рис. 47. Рабочие области

При выборе рабочей области доступен поиск по названию.

13. Поиск

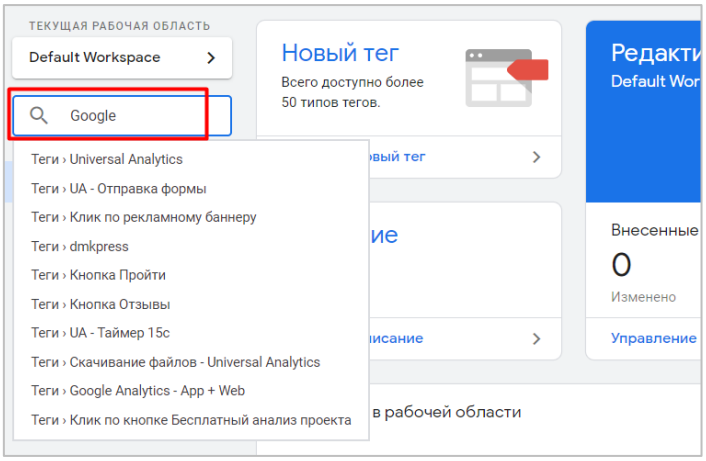


Рис. 48. Строка поиска

Функция поиска позволяет нам быстро искать информацию по тегам, триггерам и переменным в рамках выбранной рабочей области. Нажав на один из найденных элементов нас перенаправит на его настройки.

14. Вертикальное навигационное меню

Сюда входят: обзор, теги, триггеры, переменные, папки и шаблоны.

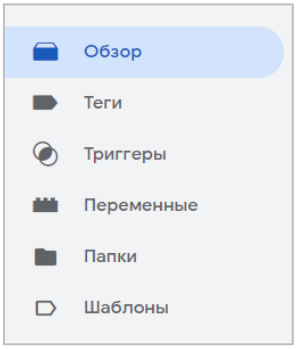


Рис. 49. Вертикальное навигационное меню

Обзор. В данный раздел входят пункты 14-20. Это общая страница, которая отображает основную информацию по текущему контейнеру GTM в виде 6 блоков (дашбордов).

Теги, триггеры, переменные, папки и шаблоны. Подробнее разберем в соответствующих главах, посвященным каждой из этих сущностей.

При переходе на каждый из этих разделов у вас есть возможность сортировки внутри таблиц по различным столбцам. Например, в тегах это можно делать по имени, типу объекта и последним изменениям.

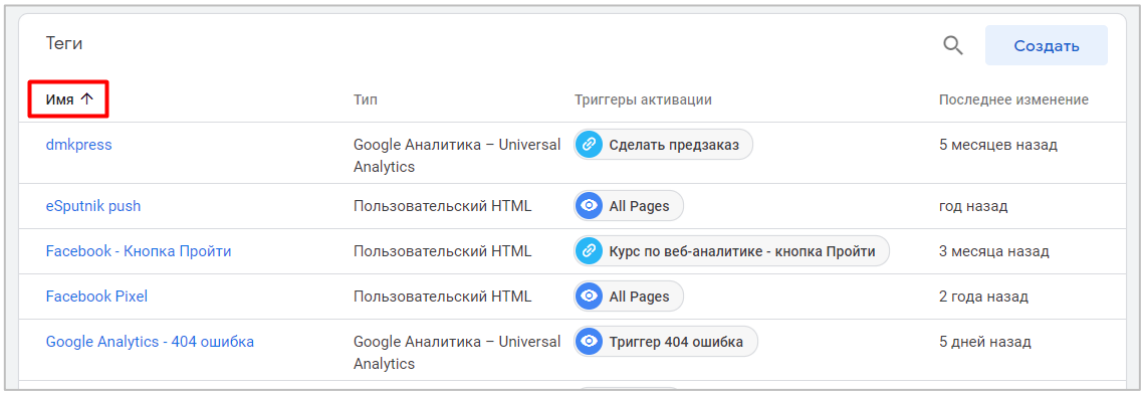


Рис. 50. Таблица тегов

В триггерах присутствует столбец, в котором выводится информация о количестве тегов, связанных с данным триггером.

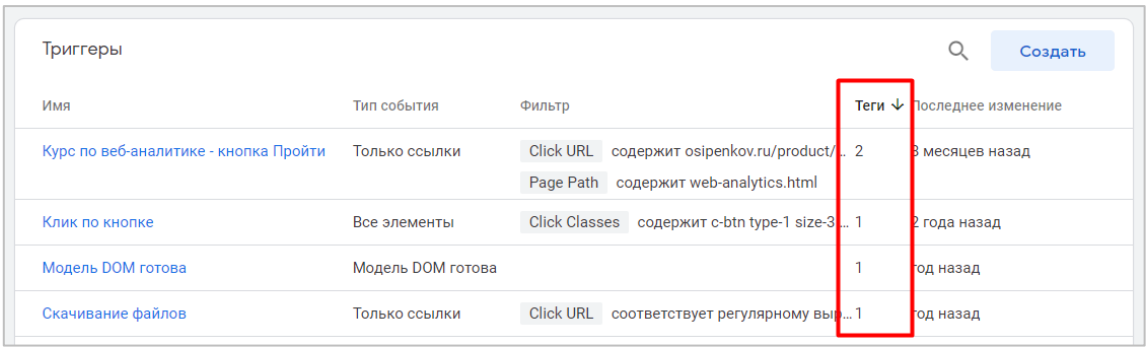


Рис. 51. Столбец Теги в разделе Триггеры

15. Блок «Новый тег»

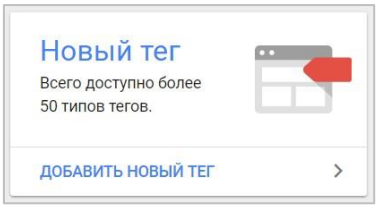


Рис. 52. Блок Новый тег

С помощью данного блока мы можем добавить новый тег в выбранную рабочую область.

16. Блок «Описание»

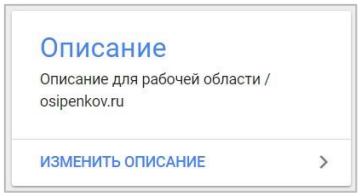


Рис. 53. Блок Описание

Предназначен для добавления описания к той области, в которой мы работаем. Чтобы вы, или человек, который будет работать с этой областью, понимали смысл всех внесенных изменений. Если вы используете различные рабочие области для сервисов Google и Яндекс, в описании к workspace можно написать: *Используется для продуктов Google.../или Яндекс.*

17. Блок «Редактирование»

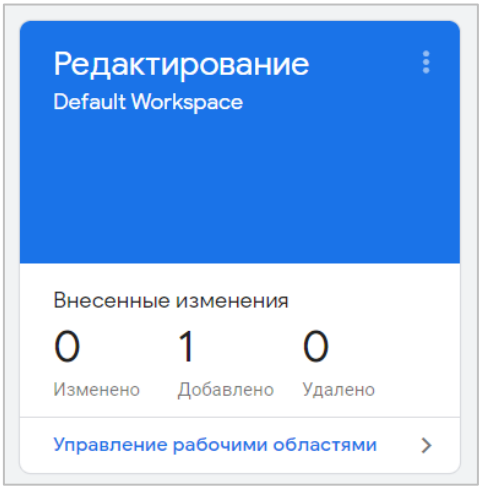


Рис. 54. Блок Редактирование

В нем отображается информация о количестве тех изменений, которые были выполнены в рабочей области перед непосредственной публикацией контейнера. Сумма изменений в этом блоке равна пункту 9 (Изменения в рабочей области).

Из раздела **Редактирование** можно перейти в управление рабочими областями (выделенное меню на изображении выше).

18. Блок «Опубликованная версия контейнера»

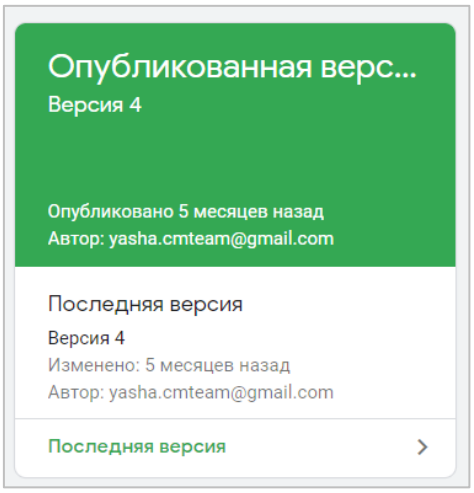


Рис. 55. Блок Опубликованная версия контейнера

Через этот блок мы можем перейти в раздел **Версии**, а также получить информацию об актуальной версии, когда она была изменена и кем.

19. Блок «Изменения в рабочей области»

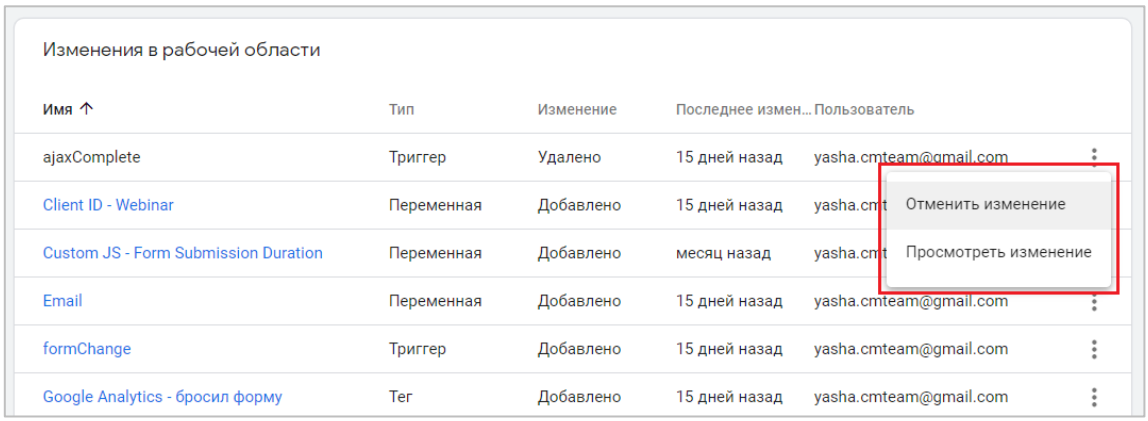


Рис. 56. Изменения в рабочей области

Здесь отображаются изменения, которые были выполнены в выбранной рабочей области с момента последней публикации контейнера Google Tag Manager.

Можно использовать сортировку по имени, типу, пользователям и изменениям, а также отменить или просмотреть изменение. При выборе **Просмотреть изменение** у нас откроется дополнительное окно с настройками. Слева – это последняя версия (то, что в данный момент опубликовано), а текущая рабочая область – это черновик нашего контейнера.

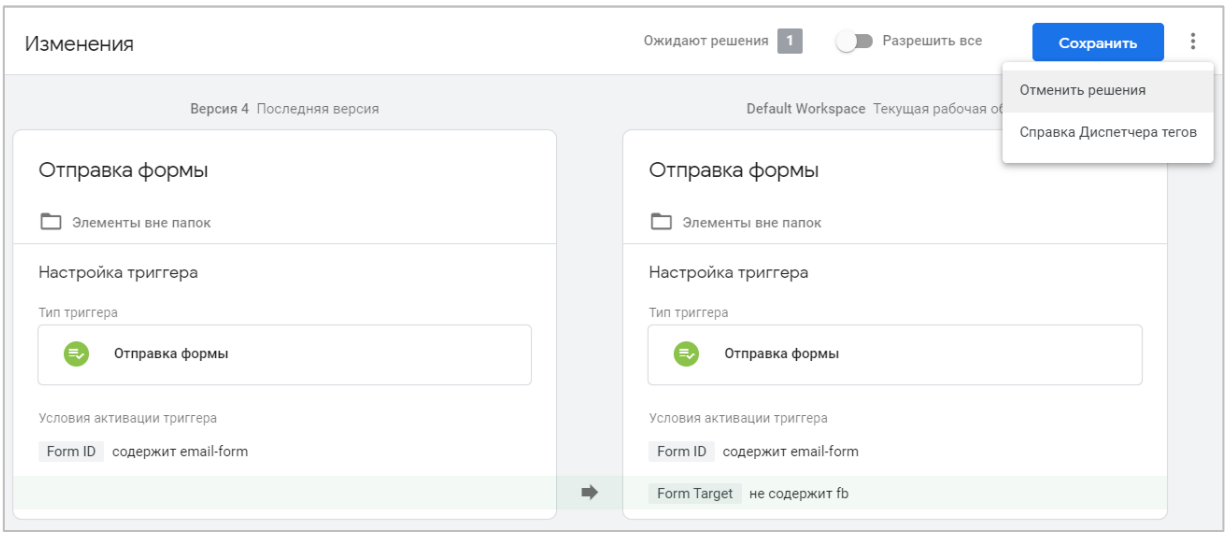


Рис. 57. Блок Изменения в рабочей области

В рассмотренном примере – это измененный триггер **Отправка формы**. Слева – опубликованная версия контейнера, справа – последние изменения, которые были внесены, но которые еще не были опубликованы. В правом верхнем углу доступна справка диспетчера тегов, а также возможность отмены решения.

20. Блок «История»

Рис. 58. Блок История

При клике на этот блок отобразится история всех изменений в выбранной рабочей области. А именно – кто сделал изменения, что было за действие (создание/изменение/удаление), какого типа использовался элемент в GTM (тег, триггер или переменная), его название и дата изменения.

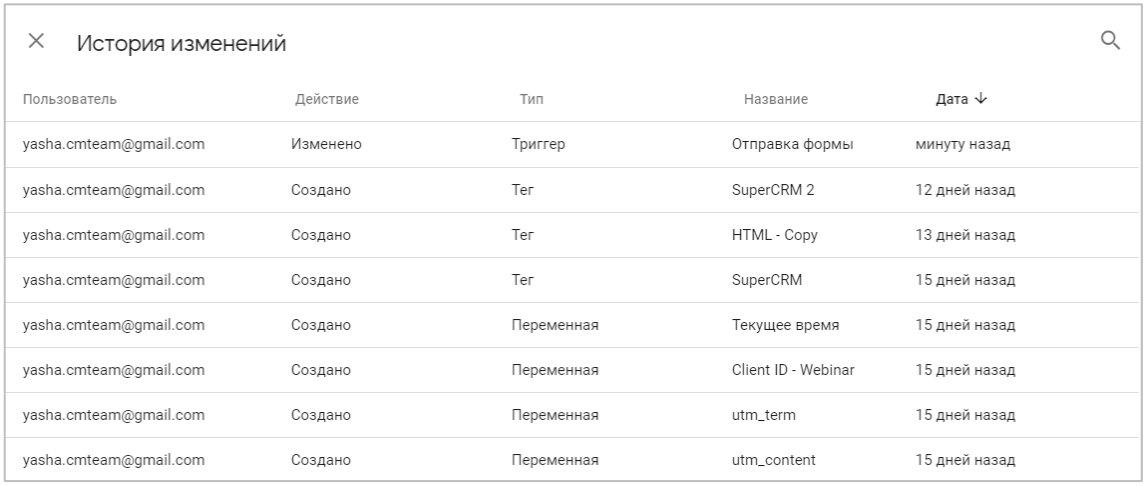


Рис. 59. История изменений

Если изменений было достаточно много, можно воспользоваться поиском в правом верхнем углу.

21. Условия использования и Политика конфиденциальности

Рис. 60. Условия использования и Политика конфиденциальности Google

Несмотря на то что Google является публичной многомиллиардной компанией и как никто другой заботится о защите персональных данных, не будет лишним хотя бы раз в своей жизни ознакомиться с материалами **Политики конфиденциальности** и **Условиях использования** их продуктов (см. приложение).

По ссылке выше вы сможете узнать:

- какую информацию собирает Google;
- как они используют эту информацию;
- как изменить свои персональные данные;
- какую информацию Google передает третьим лицам;
- какие существуют варианты доступа к данным и их обновления;
- и т.д.

Управление пользователями

В Google Tag Manager управлять пользователями и предоставлять им различные права доступа можно с помощью двух уровней.

Один из них – это **уровень аккаунта**, а другой – **уровень контейнера**. На первом уровне (аккаунта) доступы бывают двух типов:

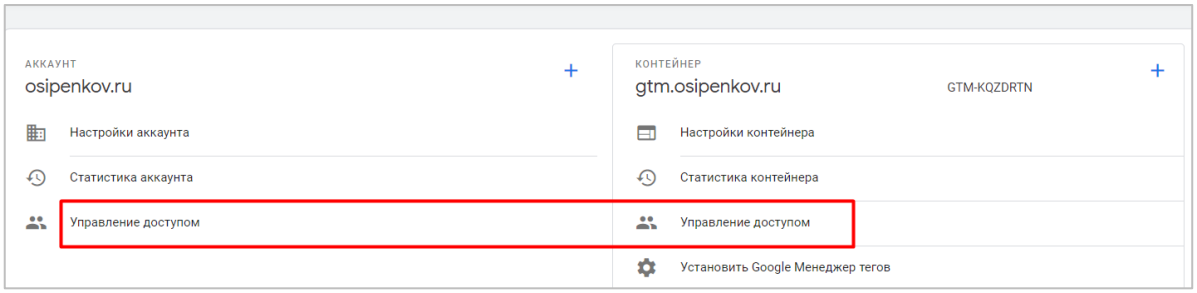


Рис. 61. Управление доступом

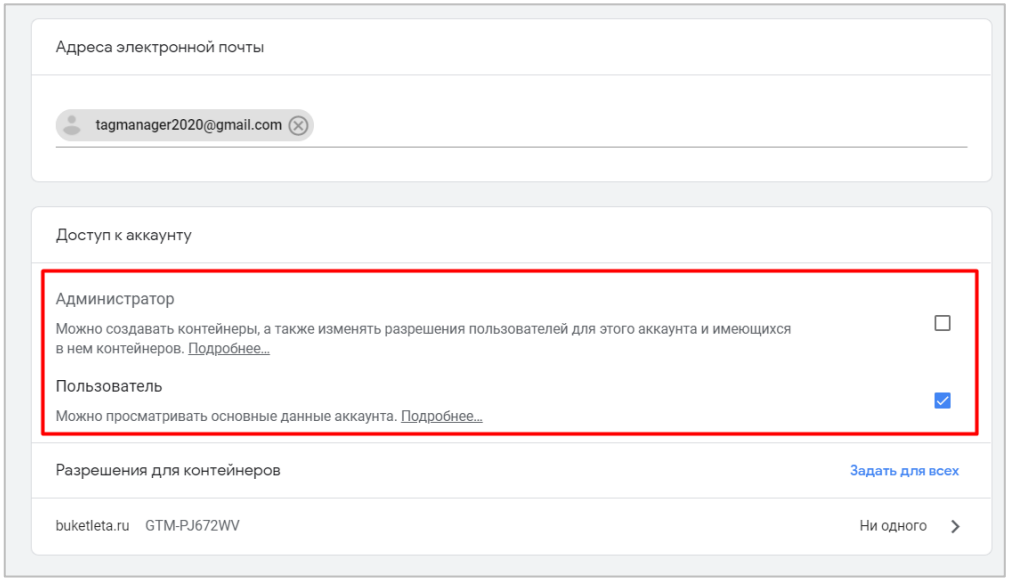


Рис. 62. Разрешения на уровне аккаунта

1. Пользователь;
2. Администратор.

Разрешение на уровне аккаунта типа **Пользователь**, к которому предоставлен данный уровень доступа, сможет **просматривать** статистику по аккаунту, список пользователей и настройки аккаунта. Разрешение на уровне аккаунта типа **Администратор** позволяет просматривать статистику, управлять списком пользователей и изменять настройки аккаунта.

На втором уровне (контейнера) возможны следующие варианты:

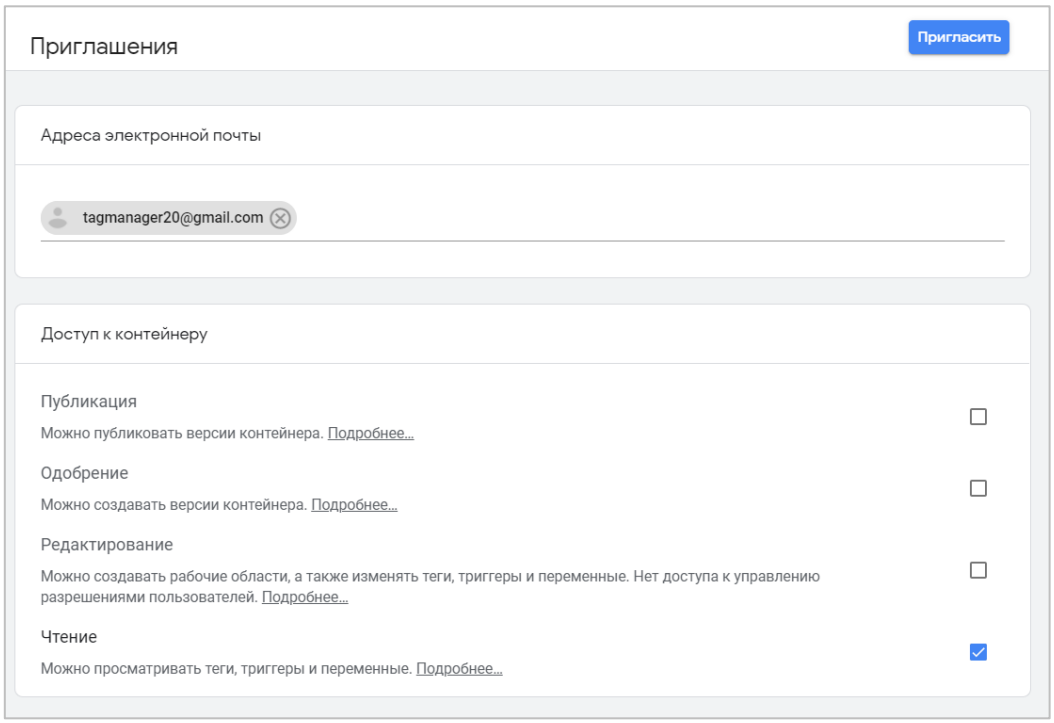


Рис. 63. Разрешения на уровне контейнера

- **Нет доступа.** Пользователь не будет видеть контейнер в аккаунте и у него нет разрешения на создание новых контейнеров для этого аккаунта

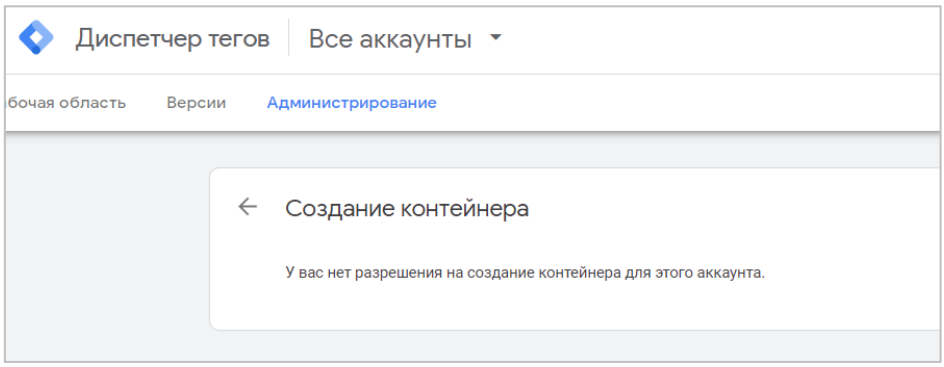


Рис. 64. Права доступа на уровне контейнера — Нет доступа

Разрешение на уровне аккаунта **Пользователь**, а на уровне контейнера **Нет доступа**.

- **Чтение.** Пользователь будет видеть контейнер в списке и сможет просматривать теги, триггеры и переменные в контейнере без возможности вносить изменения.

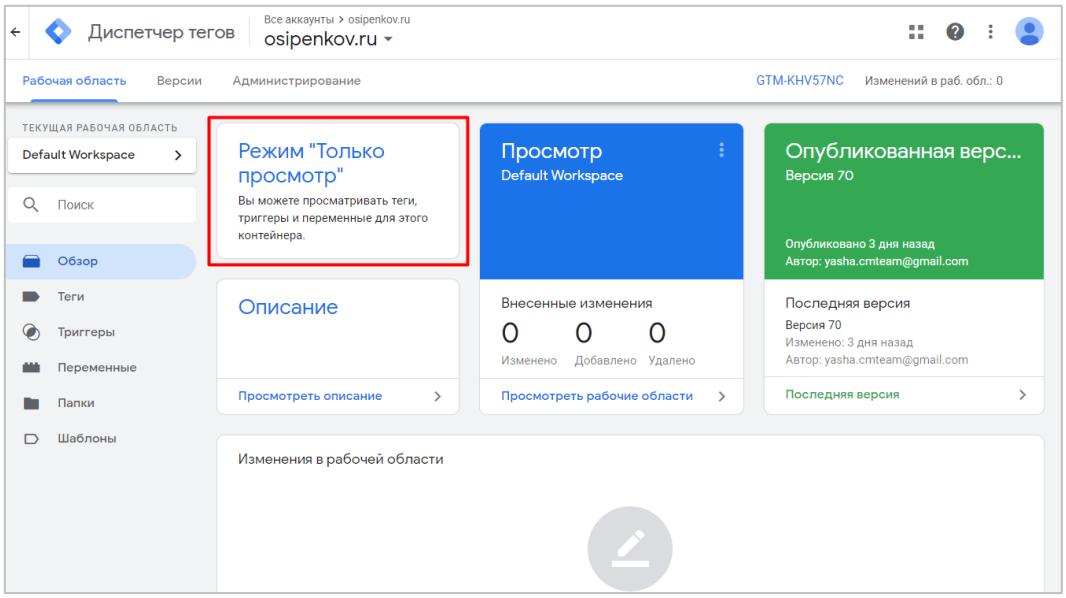


Рис. 65. Права доступа на уровне контейнера — Чтение

Кнопки **Предварительный просмотр** и **Отправить** в правом верхнем углу отсутствуют.

Однако на уровне такого доступа мы можем предварительно просматривать **Версии** и примечания к ним. Создавать версии и публиковать их возможности нет.

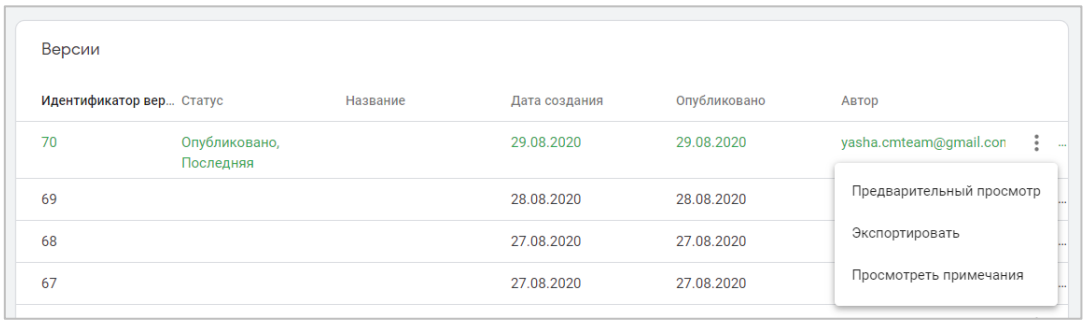


Рис. 66. Предварительный просмотр версий

Мы также можем экспортировать контейнер. Подробнее об импорт-экспорте контейнеров Google Tag Manager будет разобрано в отдельной статье.

Таким образом, разрешение на уровне контейнера **Чтение** позволяет просматривать рабочие области, настройки, теги, триггеры и переменные, статистику, пользователей, экспортировать контейнер, осуществлять предварительный просмотр версий и их примечаний.

- **Редактирование.** Пользователь имеет право создавать рабочие области и вносить изменения, но не может создавать версии и осуществлять публикацию.

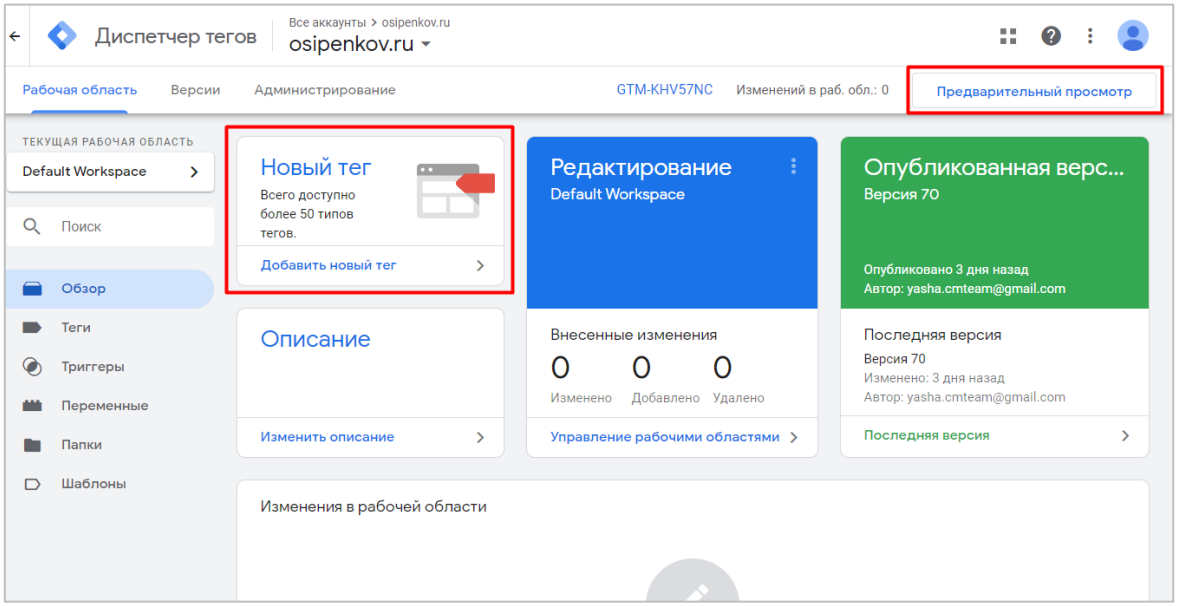


Рис. 67. Права доступа на уровне контейнера — Изменение

Доступны кнопки **Создать**, **Настроить, Предварительный просмотр.**

Таким образом, разрешение на уровне контейнера **Изменение** позволяет создавать рабочие области, просматривать настройки, статистику и пользователей, импортировать-экспортировать контейнер, осуществлять предварительный просмотр версий и их примечаний, создавать теги, триггеры и переменные.

- **Одобрение.** Пользователь имеет право создавать версии и рабочие области и может вносить изменения, но не осуществлять публикацию.

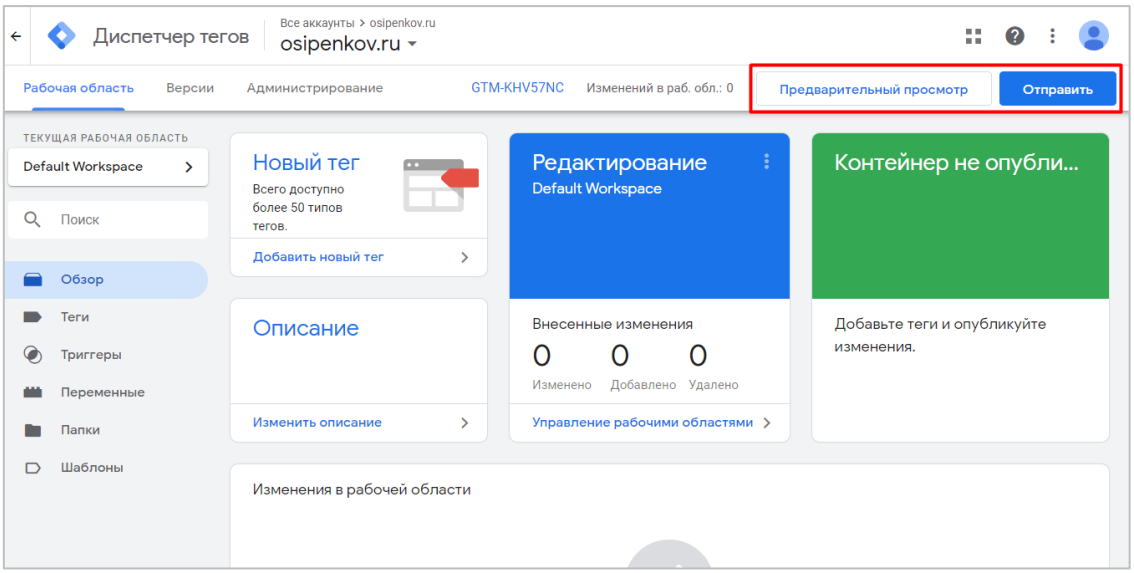


Рис. 68. Права доступа на уровне контейнера — Подтверждение

Доступны кнопки **Создать, Настроить, Предварительный просмотр** и **Отправить**.

Отличие данного доступа от **Изменение** заключается в том, что разрешение на уровне контейнера **Подтверждение** позволяет еще редактировать настройки контейнера и создавать версии.

- **Публикация.** Пользователь имеет право создавать версии и рабочие области, вносить изменения и осуществлять публикацию.

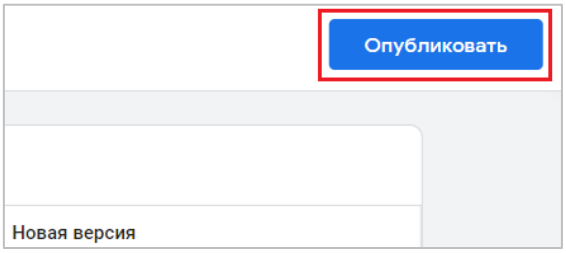


Рис. 69. Доступна публикация изменений

Доступны кнопки **Создать, Настроить, Предварительный просмотр, Отправить** и **Публикация**.

Наиболее полный доступ среди всех в Google Tag Manager с возможностью публикаций изменений в контейнере.

Чтобы предоставить или изменить права доступа на уровне аккаунта в диспетчере тегов Google:

- перейдите вкладку **Администрирование**;
- в разделе **Аккаунт** выберите **Управление доступом**;

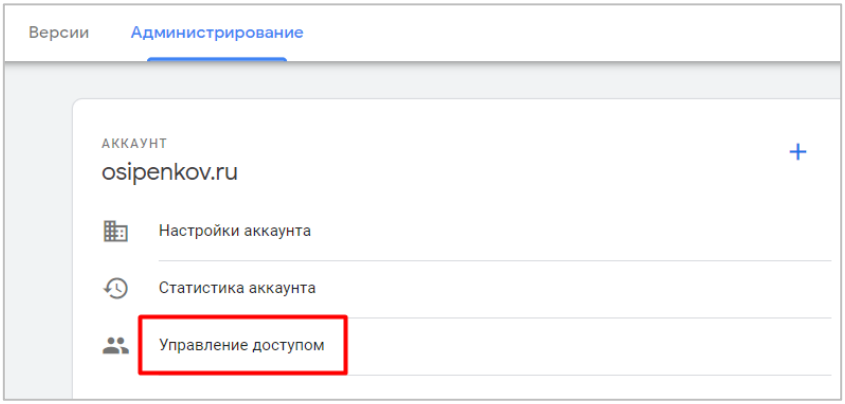


Рис. 70. Администрирование — Управление доступом

- нажмите кнопку **+** и **Добавить пользователей**;

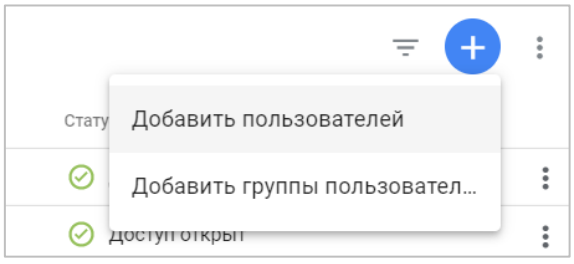


Рис. 71. Добавление пользователей

- укажите электронную почту, которая зарегистрирована в сервисах Google (аккаунт Gmail.com), разрешение на уровне аккаунта и на уровне контейнера
- нажмите кнопку **Сохранить**.

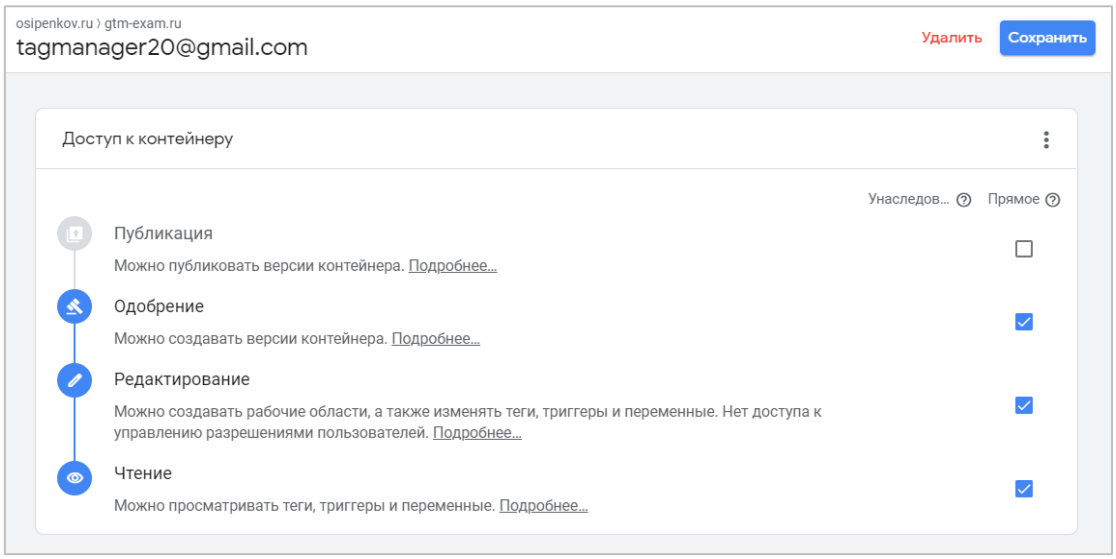


Рис. 72. Добавление нового пользователя

Пользователь добавлен в аккаунт и ему присвоен соответствующий уровень доступа. Удалить из списка на уровне аккаунта можно кликнув по его электронной почте и нажав на кнопку **Закрыть доступ**.

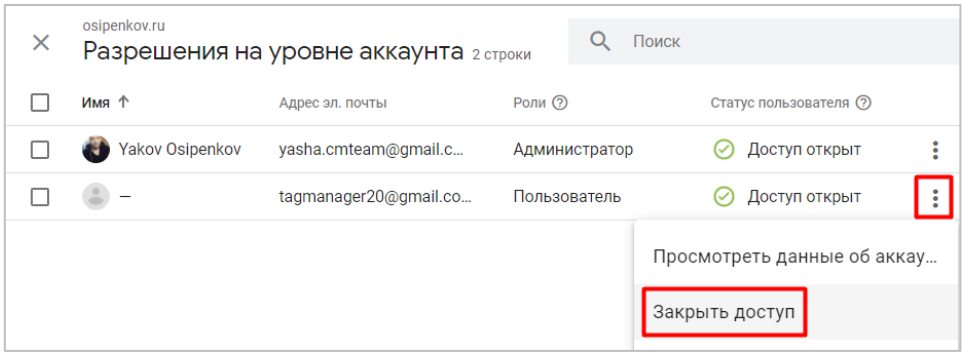


Рис. 73. Закрыть доступ к аккаунту

Удалить из конкретного контейнера можно с помощью ссылки **Удалить**:

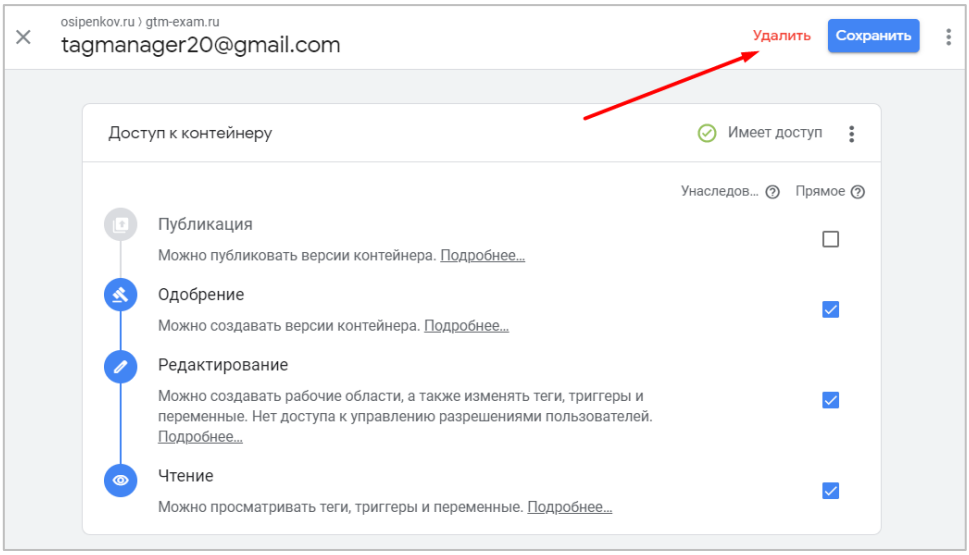


Рис. 74. Удаление доступа на уровне контейнера

Разрешения в диспетчере тегов на уровне контейнера выдаются для каждого конкретного контейнера Google Tag Manager, либо же общие доступы для всех контейнеров в аккаунте.

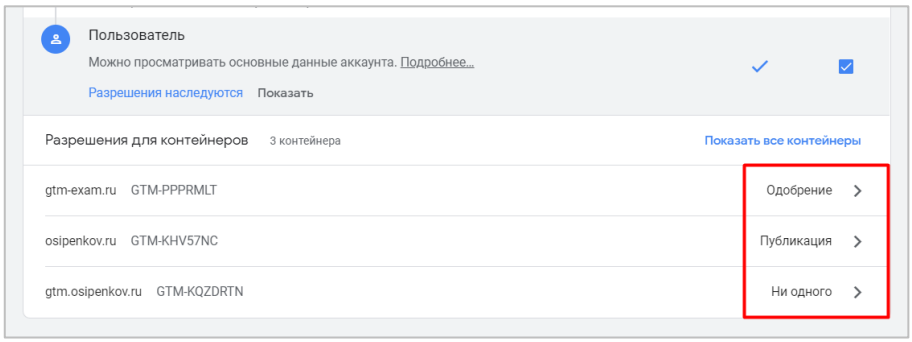


Рис. 75. Разные разрешения для разных контейнеров

Таким образом, различные уровни доступа на уровне аккаунта и контейнера в GTM предоставляются в зависимости от решаемых задач. Например, для удаленной настройки уже имеющегося контейнера или для удаленного создания и настройки нового контейнера в аккаунте. Права с уровнем доступа без возможности редактирования можно выдать для поиска причины неработоспособности контейнера, отдельных тегов и т.д.

Версии

Версия – это сохраненная копия контейнера в текущий момент времени. Делая изменения в рабочей области, вы всегда можете сохранить ее в виде версии. Благодаря этому при необходимости легко возвращаться к предыдущим изменениям. Создавать версии могут пользователи с правом на подтверждение или доступом более высокого уровня.

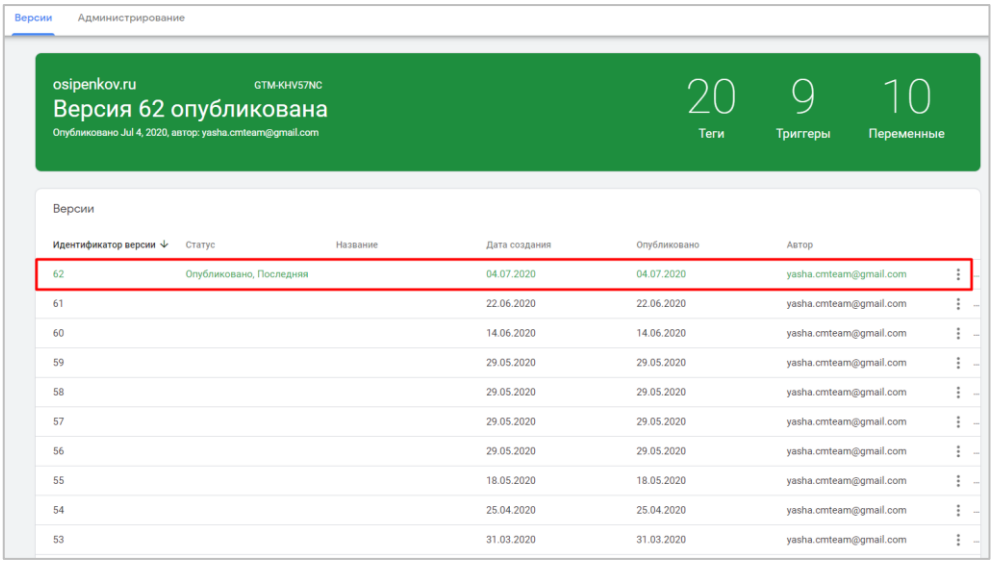


Рис. 76. Версии

Когда вы публикуете контейнер, автоматически записывается его версия. Таким образом, в вашем аккаунте будут находиться все опубликованные версии контейнера. Вы можете сохранить текущий вариант рабочей области в виде версии без публикации. Для этого во время публикации контейнера переключитесь с **Публикация и создание версии** на **Новая версия** и нажмите **Создать**.

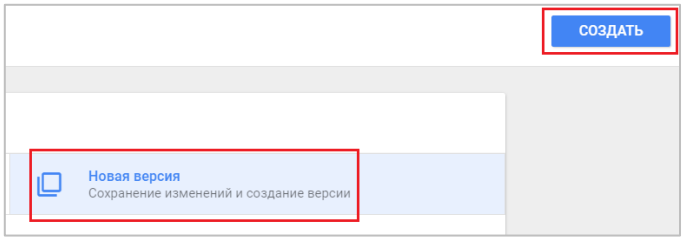


Рис. 77. Новая версия

С версиями в Google Tag Manager можно производить различные действия:

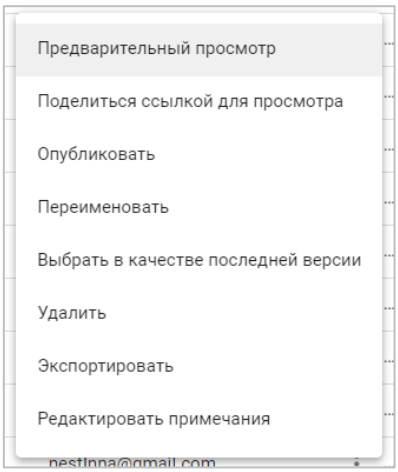


Рис. 78. Действия над версиями

- просматривать в режиме предварительного просмотра (режиме отладки);
- поделиться ссылкой для просмотра с другим человеком;
- опубликовать;
- переименовать;
- редактировать примечания (описание);
- выбрать в качестве последней версии (чтобы заменить текущую версию контейнера ранее сохраненной);
- удалить (будет удалена навсегда через 30 дней);
- экспортировать в другой контейнер (формат файла .json).

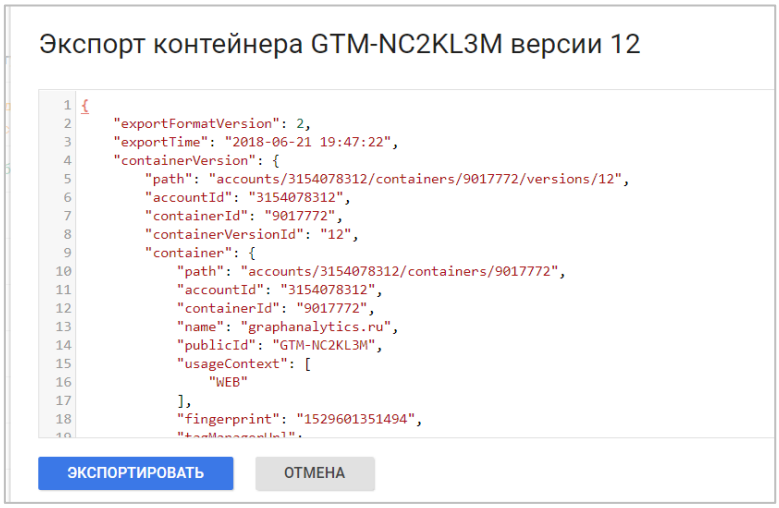


Рис. 79. Экспорт определенной версии контейнера

Диспетчер тегов сохраняет историю публикаций, так что вы всегда можете узнать, когда и кем была опубликована та или иная версия.

Администрирование

Раздел **Администрирование** разделен на два уровня:

1. Аккаунт;
2. Контейнер.

На уровне аккаунта в Менеджере тегов Google доступны настройки аккаунта, статистика аккаунта по действиям (создание, изменение, добавление тегов и т.д.) и опции управления доступами.

Администратор может разрешить вносить изменения в контейнеры только пользователям, прошедшим двухэтапную аутентификацию. Эта функция настраивается на уровне аккаунта и применяется в следующих случаях:

- создание или изменение переменных;
- создание или изменение пользовательских тегов HTML;
- изменение настроек пользователя.

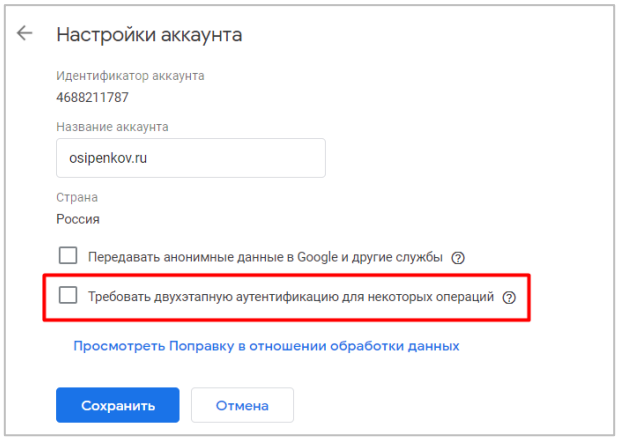


Рис. 80. Двухэтапная аутентификация для некоторых операций

На уровне контейнера в GTM доступны следующие настройки:

- настройки контейнера (название и тип контейнера);
- статистика контейнера (создание, изменение, добавление тегов и т.д.);
- управление доступами пользователей;
- установить Диспетчер тегов Google (фрагменты кодов GTM);
- импорт-экспорт контейнеров (формат файла .json);
- связывание внешних аккаунтов (например, Менеджер кампаний с контейнером GTM);
- очередь подтверждения (здесь принимаются запросы на установление связи с аккаунтом DoubleClick Campaign Manager и подтверждение перемещения тегов);
- среды (подробнее в материалах ниже);
- уведомления контейнера.

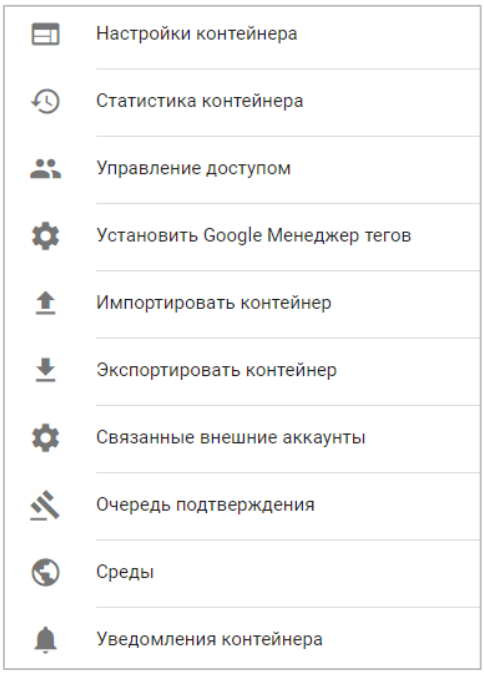


Рис. 81. Настройки на уровне контейнера

Импорт-экспорт контейнеров

Для ускорения первичной настройки и переноса данных из одного контейнера в другой в Google Tag Manager можно использовать функции импорта-экспорта.

Например, вы имеете сайт с установленным на нем контейнером GTM. Долгое время вы настраивали все теги, триггеры и переменные. В ближайших планах у вас расширение и сделать еще один сайт, другой тематики, но с той же самой структурой. Чтобы заново вручную не настраивать всю конфигурацию, в Google Tag Manager предусмотрена автоматическая выгрузка контейнера как текстового файла в формате JSON. Всего в несколько кликов вы можете перенести все настройки из одного контейнера GTM в другой.

Несомненно, в новом контейнере все же придется внести некоторые изменения – у них могут быть разные идентификаторы счетчиков, конверсий или доменные имена. Экспортированный файл можно изменить, оставив в нем только нужные триггеры, теги и переменные. Это позволяет создавать стандартные шаблоны контейнеров и делиться ими с другими пользователями Google Tag Manager. Экспортированные контейнеры можно сравнивать между собой, изменять, сохранять в системе управления версиями и импортировать обратно в GTM.

Таким образом, основное практическое применение импорт-экспорта контейнеров в GTM – это сокращение времени на выполнение большого количества однотипных действий путем дублирования конфигурации тегов, триггеров и переменных.

В качестве примера я буду переносить контейнер с настроенными тегами Google Analytics и Яндекс.Метрики. Чтобы экспортировать (выгрузить) контейнер, перейдите в **Администрирование** и выберите **Экспортировать контейнер**.

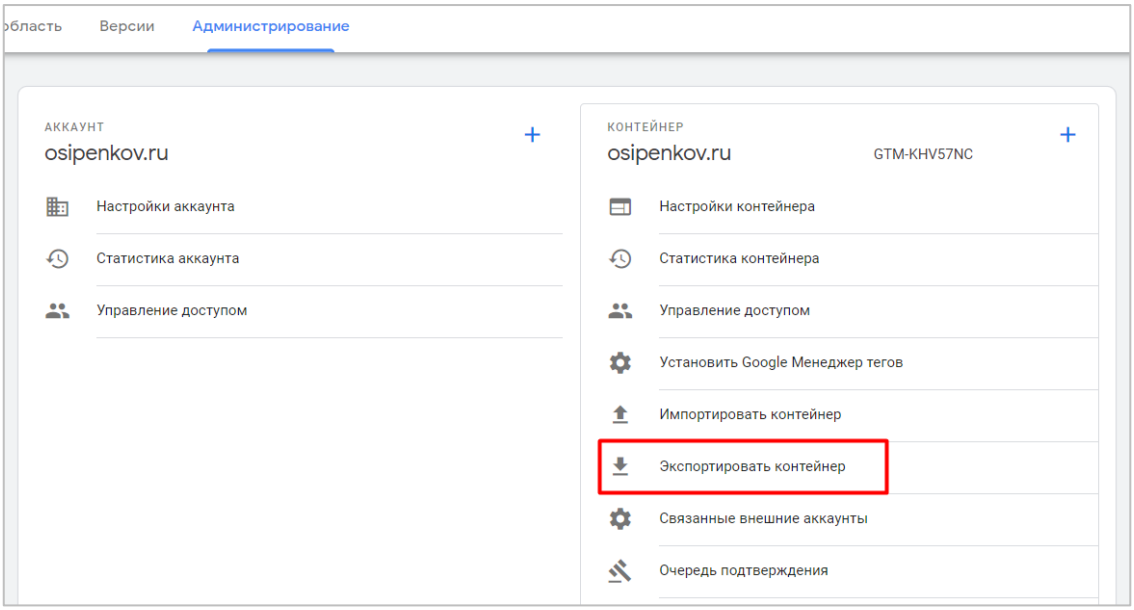


Рис. 82. Администрирование — Экспортировать контейнер

Далее выбираем версию или рабочую область. В рабочей области может содержаться несколько версий. Например, одна для тегов аналитики, другая для целей и событий, в третьей версии присутствуют изменения по электронной торговле и т.д. При выборе рабочей области экспортируются настройки всех версий. Если же вы хотите скопировать настройки только из конкретной версии, выберите только ее. В нашем случае это последняя 4 версия, которая называется Google Analytics и Яндекс.Метрика.

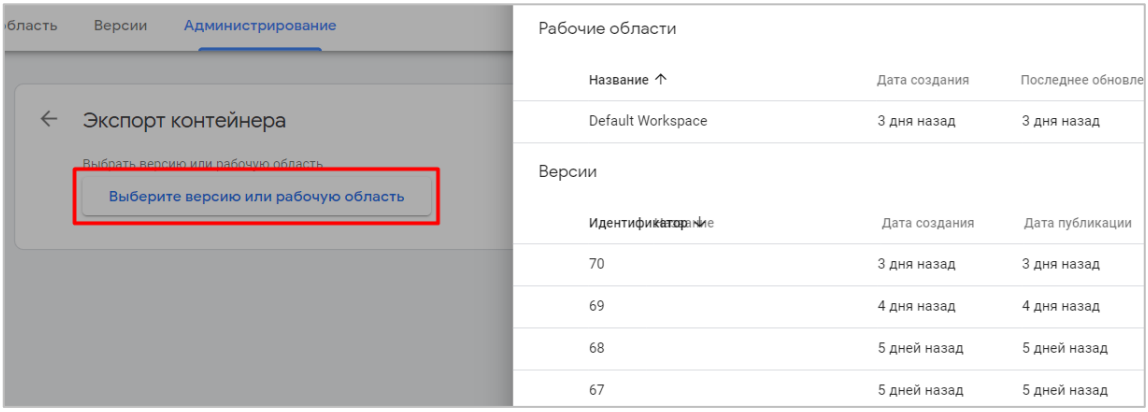


Рис. 83. Выбор версии контейнера

В следующем окне нам доступен предварительный просмотр содержимого контейнера. Если все верно, нажмите **Экспортировать**.

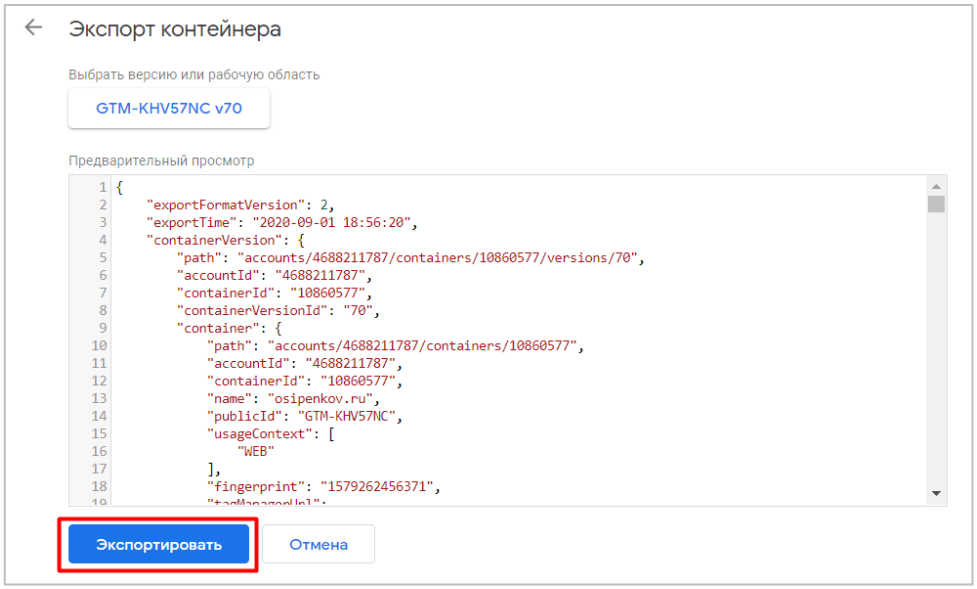


Рис. 84. Экспорт контейнера

На компьютер загрузится файл в формате JSON (формат обмена данными на JavaScript).

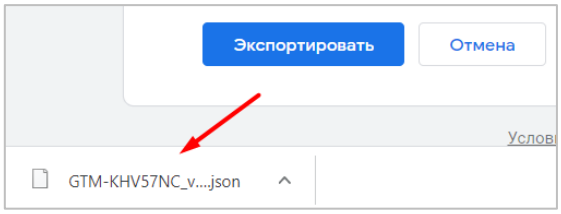


Рис. 85. Загрузка файла JSON на компьютер

Что дальше делать с этим файлом? Есть два варианта:

1. сразу же импортировать в новый контейнер;
2. сравнить данные с новым контейнером.

Рассмотрим 2 вариант. Перед тем, как загрузить наш файл в другой контейнер, мы можем выгрузить данные из второго контейнера и с помощью утилиты сравнить оба файла (экспортируемого контейнера и текущего, который будем заменять). В этом нам может помочь утилита сравнения файлов diff, которая выводит разницу между двумя файлами. Эта программа выводит построчно изменения, сделанные в файле (для текстовых файлов). Если вы используете diff, вы можете увидеть, что именно изменилось. Это способ предотвратить проскальзывание нежелательных изменений в файле.

Один из наиболее распространенных сервисов сравнения файлов **jsondiff.com**. Просто указываете два пути к файлам на компьютере и нажимаете **Compare**. Все остальное программа сделает за вас, и где надо, подсветить код.

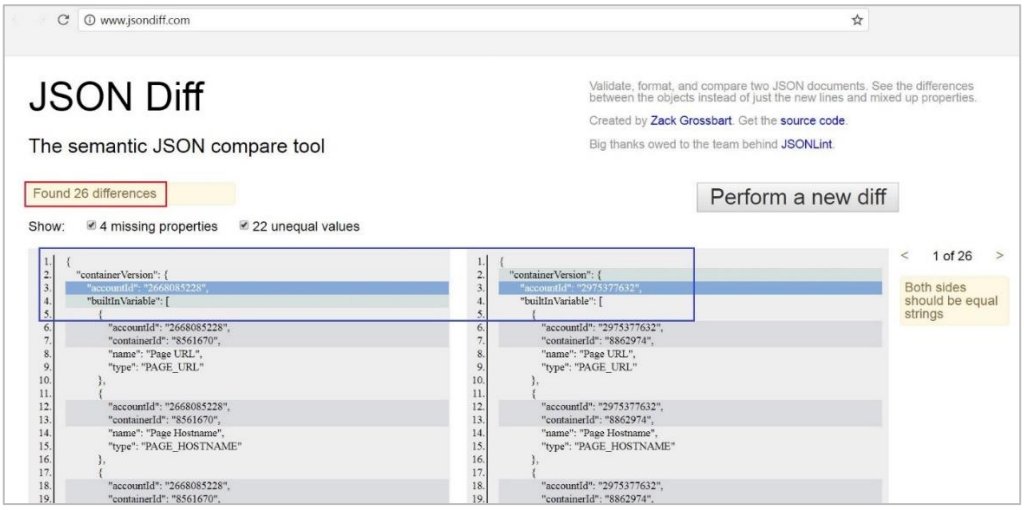


Рис. 86. JSON Diff

Утилита полезна не только при импорте-экспорте контейнеров в Google Tag Manager, но и тогда, когда требуется сравнить изменения в одном файле относительно другого. На изображении выше было найдено 26 различий, которые JSON Diff подсветил, а выделенная область синим указывает лишь на то, что два контейнера имеют разные **accountID**.

Познакомившись немного с инструментов diff, переходим к импорту нашего контейнера. Для этого переходим в контейнер, в который хотим загрузить новые настройки. Раздел **Администрирование - Импортировать контейнер**.

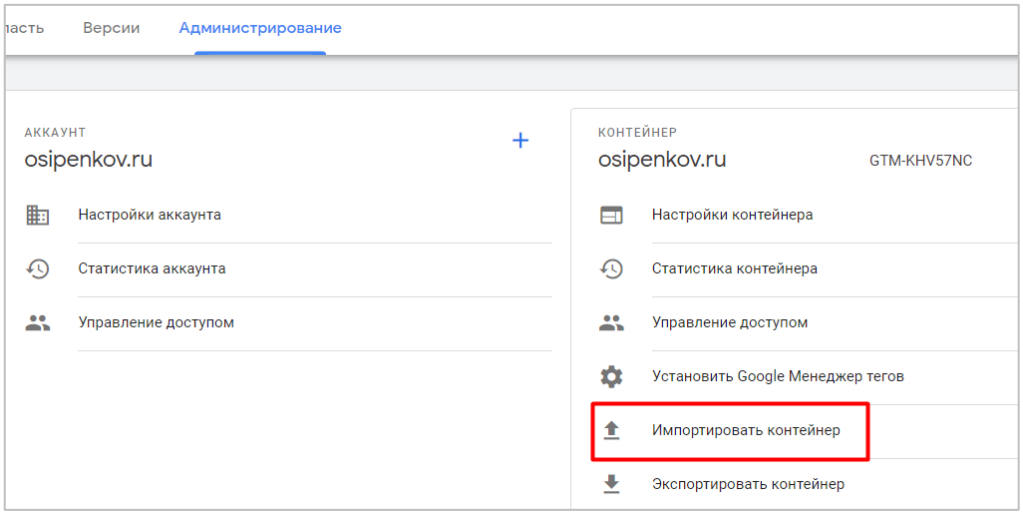


Рис. 87. Администрирование — Импортировать контейнер

Загружаем файл, который мы экспортировали, и выбираем рабочую область. Рекомендация: если у вас уже есть настройки в текущем контейнере, лучше создавать новую рабочую область. Если контейнер пустой, то можно в **Существующая**.

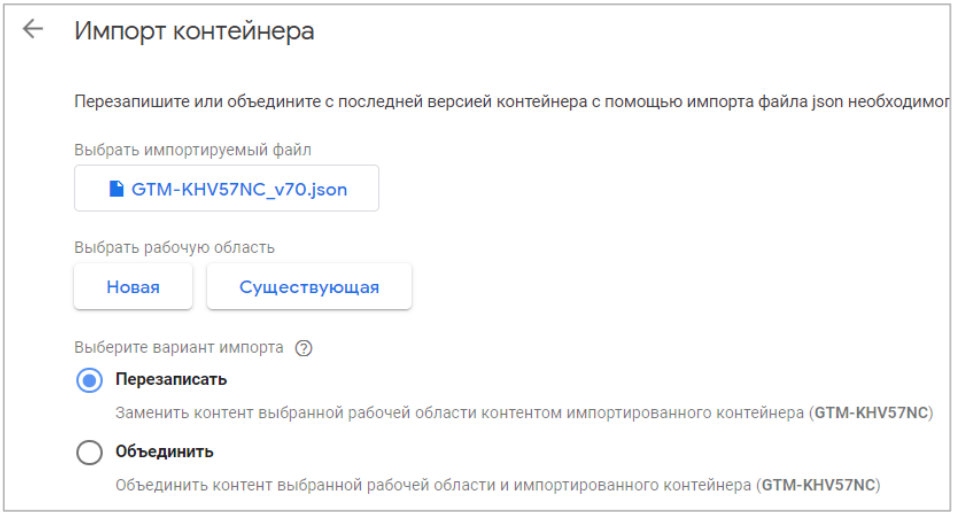


Рис. 88. Импорт контейнера

Выбираем вариант импорта:

- *перезаписать* – все старые настройки заменятся на новые;
- *объединить* – данные двух контейнеров будут объединены. При этом могут возникнуть ошибки. Чтобы это исправить, перезапишите, либо переименуйте конфликтующие теги, триггеры или переменные.

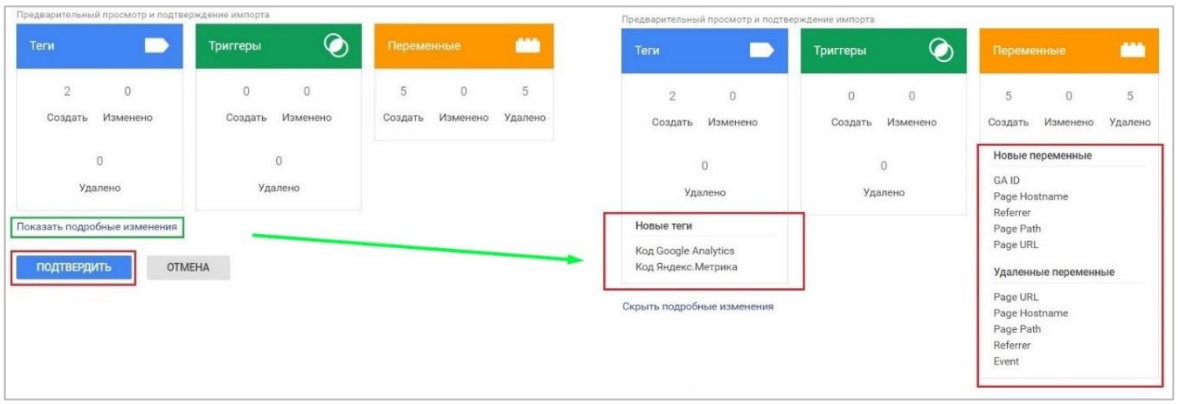


Рис. 89. Подробные изменения в контейнере

В предварительном просмотре Google Tag Manager визуально сообщит нам о том, какие теги, триггеры и переменные будут созданы, изменены или удалены. Можно посмотреть подробных список изменений, нажав на **Показать подробные изменения**. Если все сделали правильно, остается только **Подтвердить**.

На этом импорт данных в контейнер завершен. Не забудьте опубликовать контейнер, чтобы текущая конфигурация стала доступна пользователям. Если вы импортируете большое количество данных, перед публикацией обязательно проверяйте все изменения через отладчик GTM.

P.S. В интернете есть большое количество готовых JSON-файлов для Google Tag Manager. Например, компания **Bounteous (ex LunaMetrics)** на своем сайте (см. приложение) публикует различные решения с подробным описанием. Автор блога **analyticsmania.com** также собрал рецепты от разных команд и веб-аналитиков, в числе которых: **Simo Ahava, David Vallejo, Pat Grady, Daniel Carlbom** и другие.

Среды

В Google Tag Manager есть возможность создания так называемых сред, которые позволяют протестировать изменения, внесенные в контейнер, перед их публикацией на сайте. Прежде чем разобраться в этом функционале, постараемся понять, как работает контроль версий в диспетчере тегов.

Рис. 90. Пример версий

При создании тега, триггера и переменной до публикации все изменения хранятся в черновой версии. Чтобы новые данные стали доступны пользователю, версию необходимо опубликовать. Все версии контейнера хранятся на вкладке **Версии**.

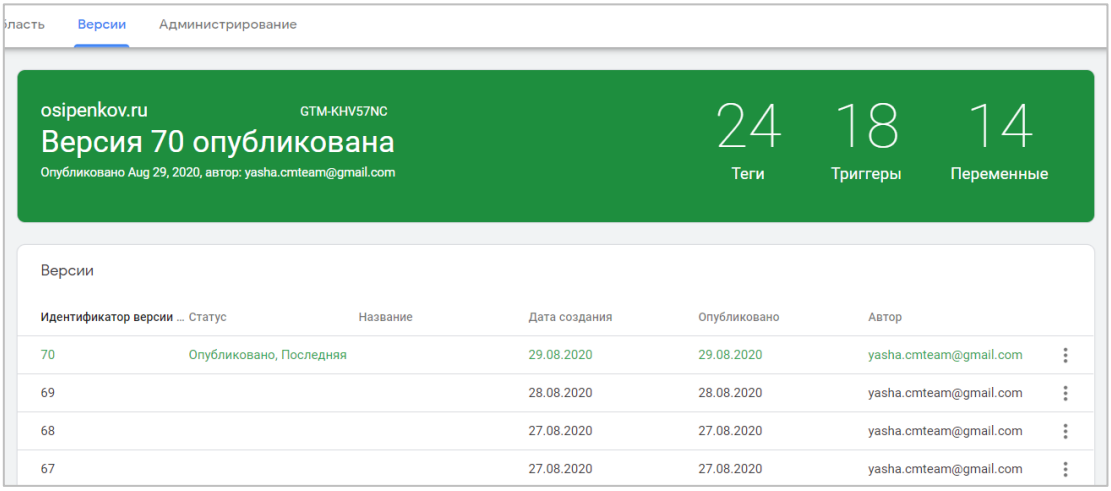


Рис. 91. Версии

Во время внесения изменений в конфигурацию отправки версии у нас есть возможность задать среду для публикации.

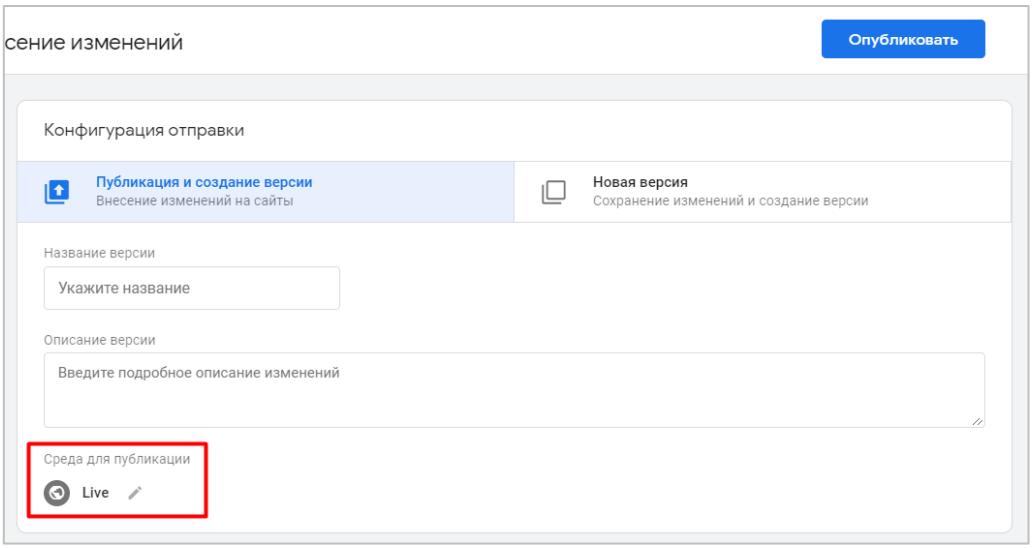


Рис. 92. Задание среды для публикации

Несмотря на то, что обычно публикуется последняя версия контейнера, Google Tag Manager также дает нам возможность переиздавать более старые версии. Эта функция как раз необходима для сред диспетчера тегов.

Когда-то GTM разрешал публиковать только одну версию контейнера. Это означало, что для каждого отдельного веб-сайта у вас был единственный фрагмент кода Google Tag Manager.

Но ситуация кардинально изменилась, когда пришло осознание того, что веб-сайты могут находиться на разных этапах разработки – от предварительной подготовки, производства, до бета-тестирования и окончательного запуска. При таком подходе все равно необходимо устанавливать теги, отслеживать статистические данные и обеспечивать должное качество работы.

Рассмотрим пример, в котором у нас есть две версии сайта:

1. тестовый вариант, над которым разработчики ведут работы (недоступен широкой публике);
2. конечный вариант, который в данный момент доступен для пользователей в сети.

Рис. 93. Пример версий сайтов

Обе версии являются копиями одного и того же сайта, и оба используют один и тот же контейнер Google Tag Manager. Но вы не будете публиковать версию тестового варианта, поскольку она сырая и над ней ведутся работы по устранению ошибок.

Благодаря средам в GTM стала доступна одновременная публикация нескольких версий контейнеров, что значительно упрощает тестирование и отладку тегов.

Рис. 94. Две среды разработки: конечная и промежуточная

В приведенном выше примере у нас есть две среды: *конечная (live, 3)* и тестовая *(промежуточная, 4)*. Конечная среда будет содержать теги, которые мы хотим запустить на сайте, а промежуточная среда будет включать в себя дополнительные теги, которые либо проходят тестирование, либо не готовы к публикации в финальной версии.

Чтобы получить доступ к средам, перейдите в раздел **Администрирование** и откройте **Среды** на уровне контейнера.

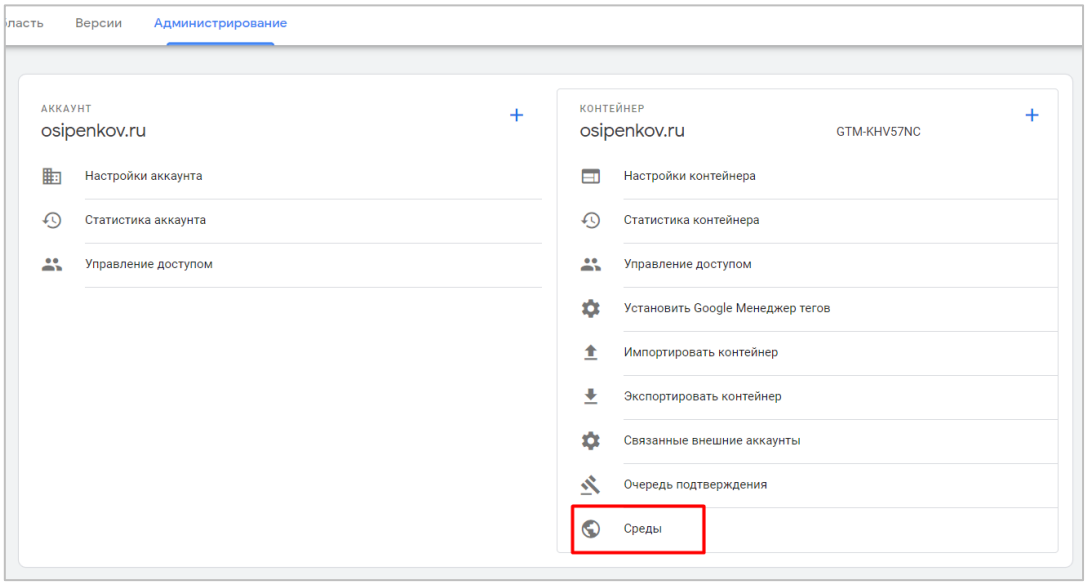


Рис. 95. Администрирование — Среды

По умолчанию в контейнере создается две среды – **Live (реальная)** и **Latest (последняя)**.

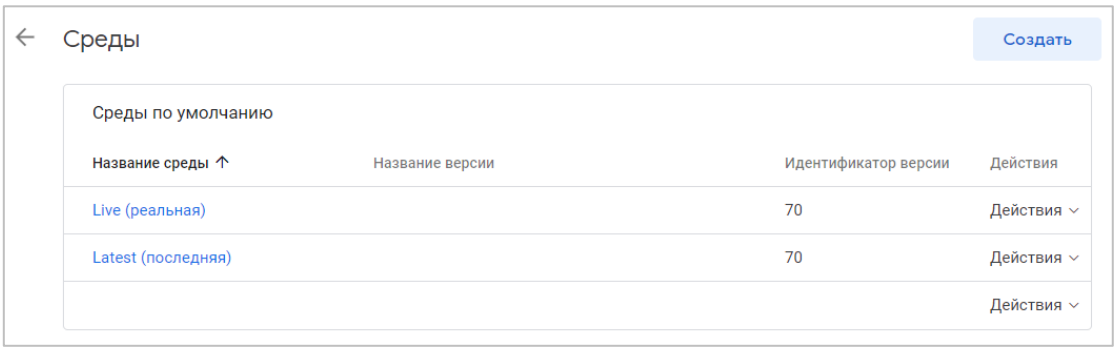


Рис. 96. Среды по умолчанию

Live (реальная) среда всегда указывает на версию контейнера, которая была опубликована, а **Latest (последняя)** среда указывает на последнюю версию контейнера, которая была создана. Однако это не обязательно та же версия, что и Live.

В Google Tag Manager можно создавать собственные среды. Для этого нажмите кнопку **Создать**. В конфигурации будет доступно 4 опции:

1. название среды;
2. описание (чтобы другие пользователи вашего контейнера GTM могли быстро определить, для чего используется среда);
3. включение отладки;
4. поле с указанием целевого URL.

Заполнив все необходимые данные, нажмите **Создать среду**.

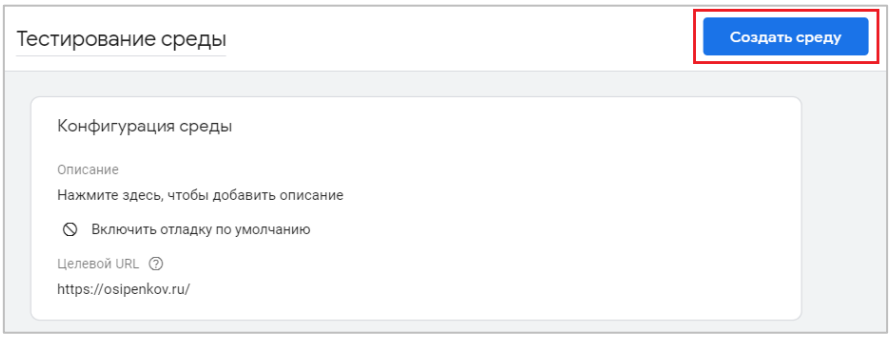


Рис. 97. Создание новой среды

Далее сообщение от Google о начале работы с новой средой.

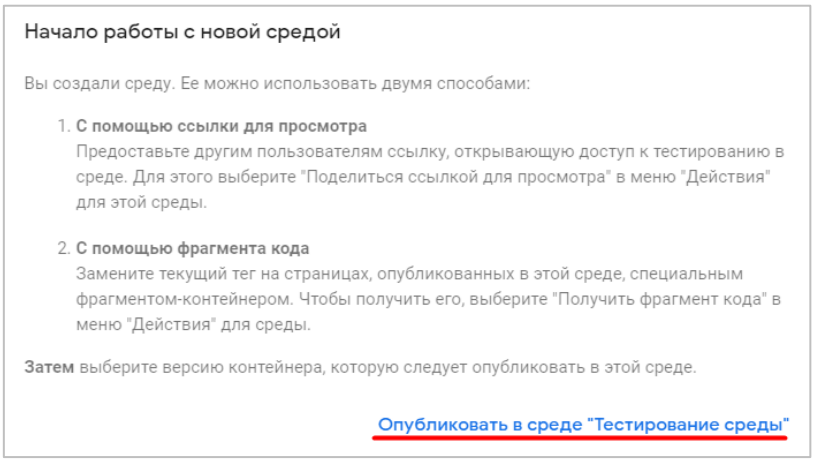


Рис. 98. Использование среды двумя способами

Вы создали среду. Ее можно использовать двумя способами:

1. **С помощью ссылки для просмотра**

Предоставьте другим пользователям ссылку, открывающую доступ к тестированию в среде. Для этого выберите **Поделиться ссылкой для просмотра** в меню **Действия** для этой среды.

2. **С помощью фрагмента кода**

Замените текущий тег на страницах, опубликованных в этой среде, специальным фрагментом-контейнером. Чтобы получить его, выберите **Получить фрагмент кода** в меню **Действия** для среды.

Примечание: не устанавливайте фрагмент контейнера среды рядом со стандартным фрагментом контейнера Google Tag Manager на том же сайте. Это может привести к дальнейшим ошибкам.

Также можно загружать среды на сайт с помощью функции предварительного просмотра. При этом браузер автоматически перейдет в режим предварительного просмотра, и вы сможете протестировать любую версию среды. Таким образом вы получаете доступ ко всем функциям тестирования без установки и внедрения фрагментов кода.

Если вы уже работали с контейнером и имеете различные версии, то система предупредит о том, что при публикации эти изменения появятся на сайте. Вы можете сначала выполнить предварительный просмотр и отладку.

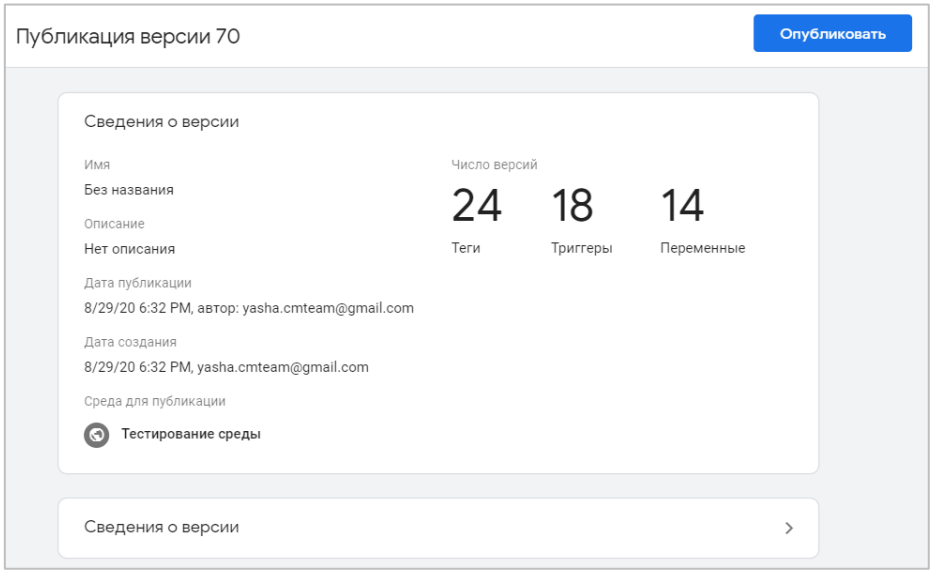


Рис. 99. Выбор версии и публикация в среде

Если же контейнер новый и не имеет различных версий, то при публикации выбранная версия будет перемещена в только что созданную среду. Нажмите **Опубликовать в среде** и возвращаемся в меню, где видим только что созданную специальную среду.

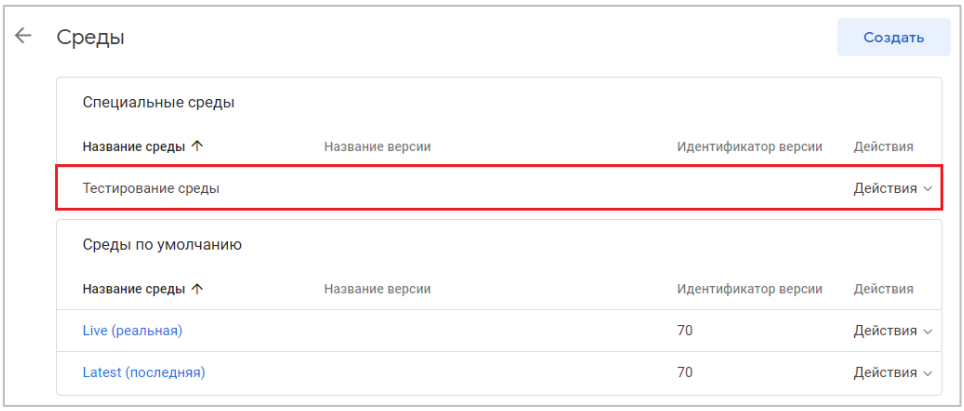


Рис. 100. Специальные среды

В разделе **Версии** появится столбец **Среды**, в котором для каждой версии будет указана среда публикации.

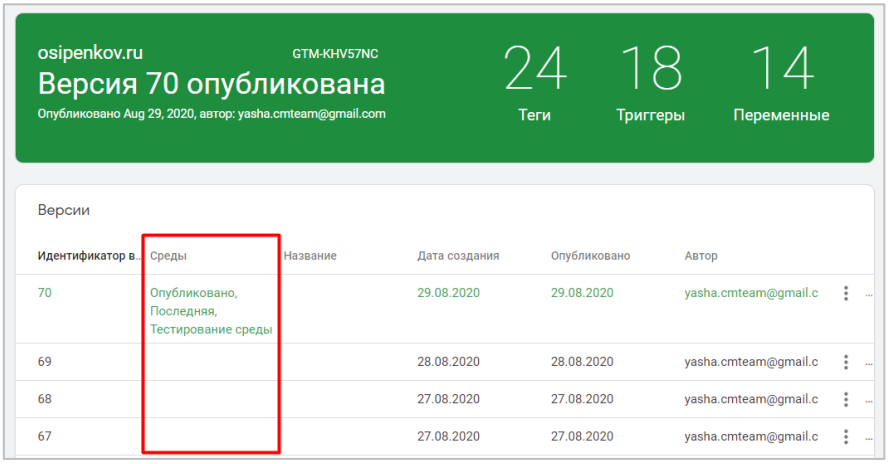


Рис. 101. Новый столбец Среды

В отличие от сред по умолчанию, которые динамически изменяются в зависимости от того, какая версия вашего контейнера GTM активна, пользовательские среды всегда будут загружать одну и ту же версию, если вы явно не измените опубликованную версию.

Вы можете задать неограниченное количество сред и публиковать любые версии контейнеров в любой из них. Действия, которые можно совершать над средами:

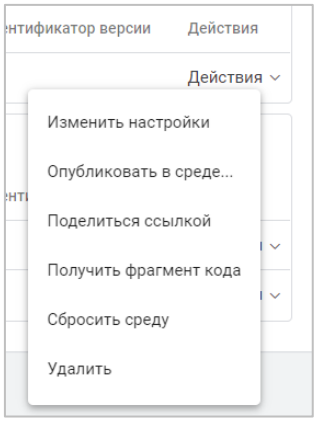


Рис. 102. Действия со средами

- изменить настройки (описание, целевой URL, режим отладки);
- опубликовать в среде (возможность смены версии контейнера, можно задать любую версию контейнера, откатиться на более раннюю);
- поделиться ссылкой (открытие доступа к просмотру и тестированию в среде);

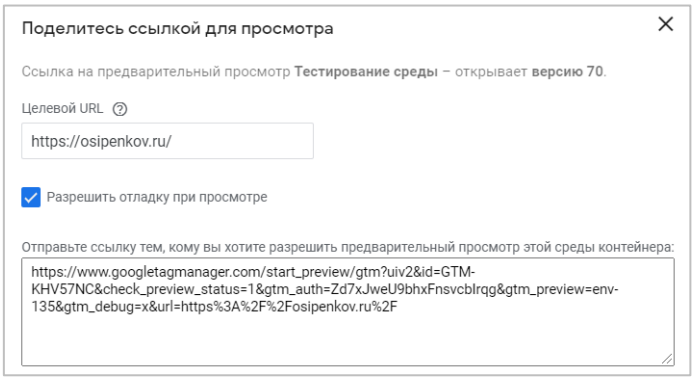


Рис. 103. Поделиться ссылкой для просмотра

- получить фрагмент кода (от основного контейнера GTM отличаются параметры **gtm_auth** и **gtm_preview**);

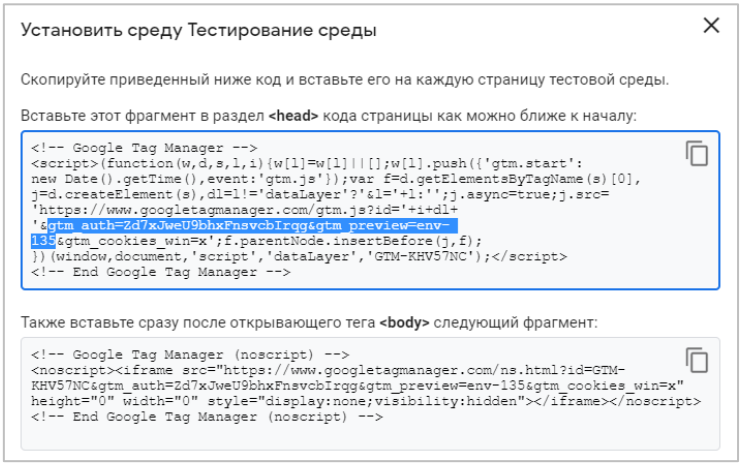


Рис. 104. Фрагмент кода для тестовой среды

- сбросить среду;

Если вы не хотите, чтобы ссылка, которой вы поделились с другим человеком, была доступна, вы можете сбросить код авторизации (параметр **gtm_auth**).

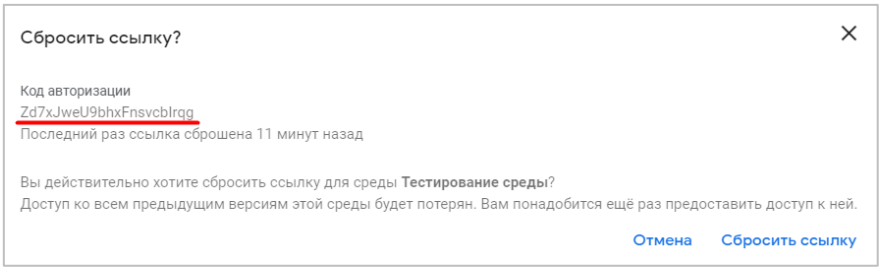


Рис. 105. Сбросить ссылку

Примечание: все установленные фрагменты кода также станут недействительны. Поэтому после сброса ссылки необходимо установить на сервер среды новый фрагмент кода контейнера Google Tag Manager.

- удалить среду;

Удаление среды приведет к потере доступа навсегда.

Как правило, среды в Google Tag Manager предназначены для работы с промежуточными версиями сайта. Если же сайт готов к полноценному запуску и никаких изменений в ближайшее время не предвидится, рекомендуется устанавливать стандартный фрагмент кода контейнера GTM.

Уведомления контейнера

В начале сентября 2020 года в диспетчере тегов Google появилась новая настройка **Уведомления контейнера**.

Подписавшись, вы будете получать уведомления на ваш адрес электронной почты о ключевых состояниях контейнера GTM, а именно когда:

- версия контейнера была опубликована;
- новая версия контейнера создана, но не была опубликована;

Чем полезны уведомления контейнера?

Во-первых, вы всегда будете знать об изменениях в текущей версии контейнера и понимать, когда последний раз производилась работа в вашем Google Tag Manager. Например, если вы поручили настройку своему подрядчику, то сообщения на почту позволят вам отслеживать текущее состояние работы и контролировать сроки ее выполнения.

А во-вторых, это просто очень удобно, особенно когда вы работаете в агентстве, являетесь руководителем отдела, группы специалистов, и ведете несколько проектов одновременно. Получая сообщения о созданных и опубликованных версиях контейнера, вы всегда будете знать, над чем ваши сотрудники работают(работали) в данный момент. И, напротив, отсутствие таких уведомлений будет косвенно свидетельствовать о том, что интернет-маркетолог не приступал к настройкам отслеживания в Google Tag Manager, не создавал новые версии и не публиковал изменения в контейнере.

Каждый пользователь диспетчера тегов Google может включить и настроить свои собственные настройки уведомлений. Сделать это можно на двух уровнях:

1. для отдельных контейнеров. Опция доступна в разделе **Администрирование – Контейнер – Уведомления контейнера;**
2. сразу для всех контейнеров, к которым у вас есть доступ. Опция доступна в разделе **Пользовательские настройки.**

Уведомления для отдельных контейнеров. Администрирование – Контейнер – Уведомления контейнера

Находится она на уровне контейнера:

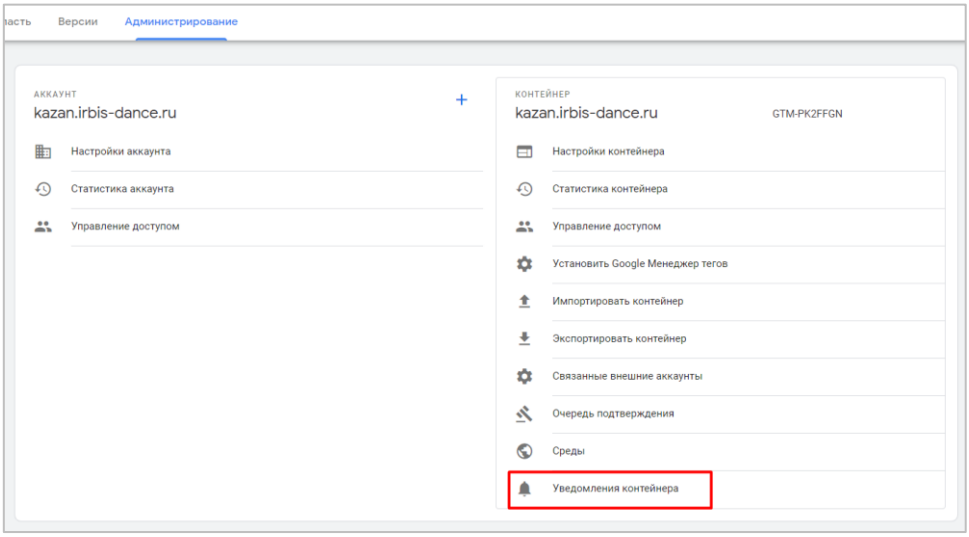


Рис. 106. Администрирование - Контейнер - Уведомления контейнера

Помимо раздела **Администрирование**, настройка доступна на главной странице диспетчера тегов:

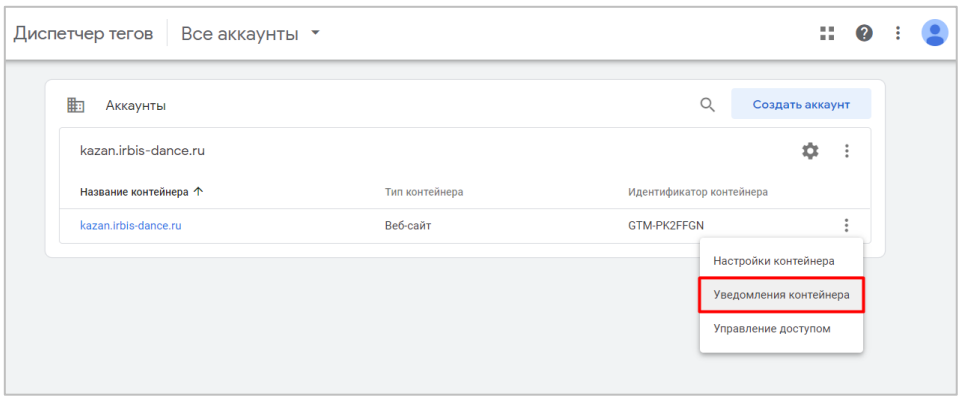


Рис. 107. Все аккаунты

Версия опубликована имеет три варианта настройки:

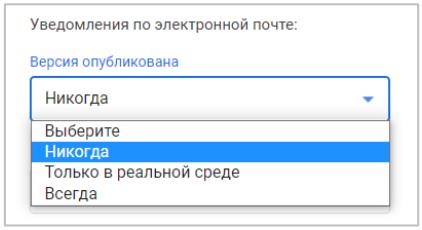


Рис. 108. Версия опубликована

- параметр **Всегда** отправляет электронное письмо всякий раз, когда версия публикуется в любой среде;
- параметр **Только в реальной среде** отправляет электронное письмо только тогда, когда версия публикуется в среде для публикации **Live**;
- параметр **Никогда** не будет отправлять электронное письмо.

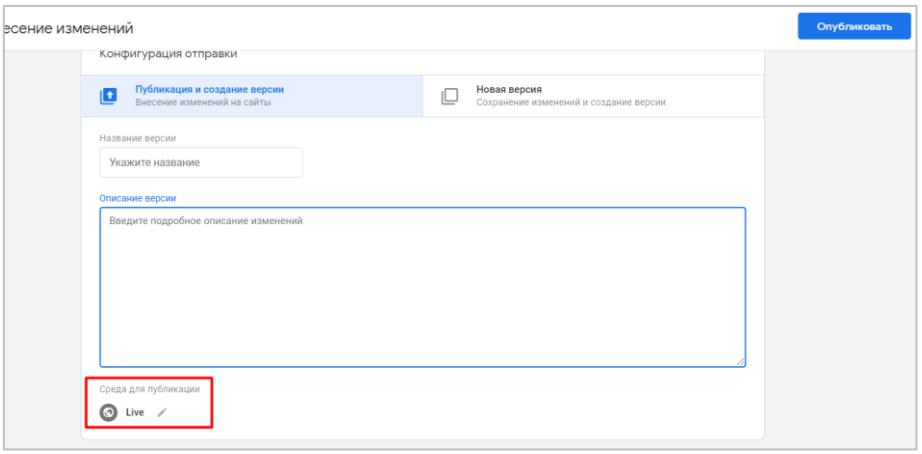


Рис. 109. Среда для публикации - Live

Новая версия контейнера создана, но она не опубликована имеет два варианта настройки:

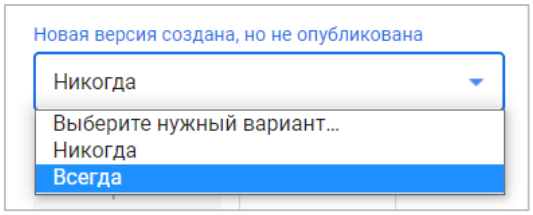


Рис. 110. Новая версия создана, но не опубликована

- параметр **Всегда** отправляет электронное письмо всякий раз, когда версия контейнера создается (но не публикуется)
- параметр **Никогда** не будет отправлять электронное письмо.

Уведомления для всех контейнеров. Пользовательские настройки

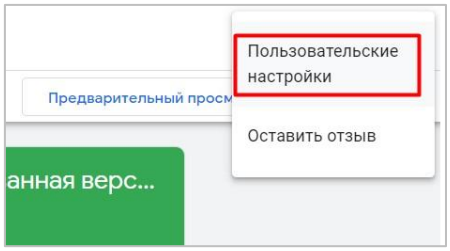


Рис. 111. Пользовательские настройки

Когда вы устанавливаете уведомления контейнера по умолчанию, они будут применяться ко всем контейнерам, к которым у вас есть доступ, пока вы не перейдете к каждому отдельному контейнеру и не измените значение параметра уведомления в нем.

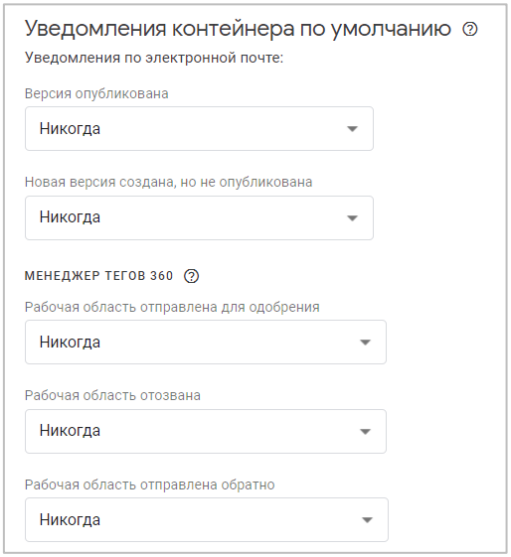


Рис. 112. Уведомления контейнера по умолчанию

Кроме уведомлений для обычного диспетчера тегов, Google добавил настройки к контейнерам в аккаунте Менеджера тегов 360 для рабочих областей. Когда:

- рабочая область отправлена для одобрения;
- рабочая область отозвана;
- рабочая область отправлена обратно.

Вот так выглядит уведомление на почте об опубликованной версии:

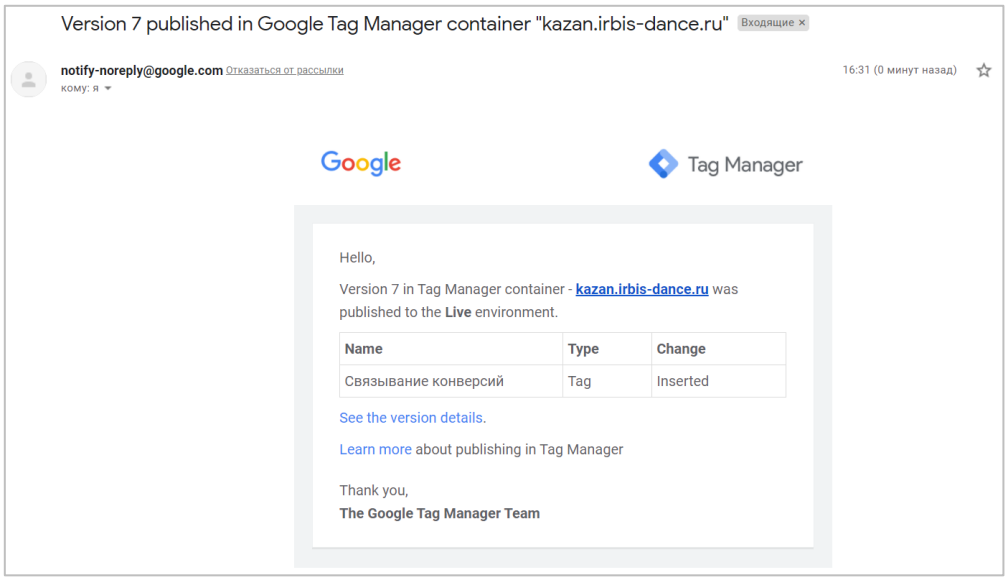


Рис. 113. Письмо на почту: Версия опубликована

А когда версия создана, но не опубликована, на почту приходит такое уведомление:

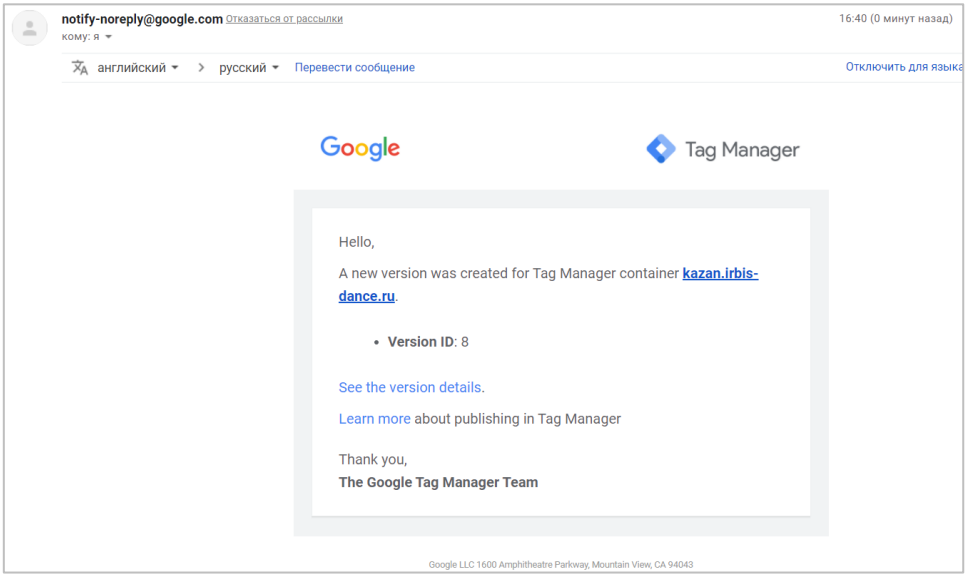


Рис. 114. Новая версия создана, но не опубликована

Вы можете отказаться от подписки на уведомления, щелкнув на ссылку **unsubscribe from this category of emails** в нижней части электронного письма с уведомлением или изменив настройки уведомлений в интерфейсе GTM.

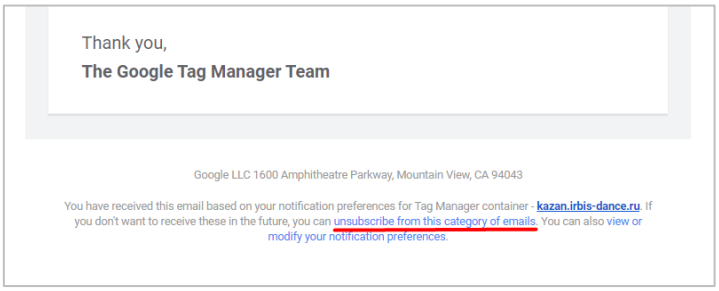


Рис. 115. Отписаться от уведомлений

Функция **Уведомления контейнера** - очень полезное нововведение, которое позволяет контролировать ключевые состояния контейнера. На момент написания данного руководства в обычной версии Google Tag Manager доступно не так много настроек. Возможно, со временем разработчики добавят более гибкие уведомления контейнера, такие как: создание, удаление переменных, триггеров, тегов и изменений прав доступов.

Папки

Еще один способ организации элементов внутри контейнера Google Tag Manager – это папки.

Со временем в Google Tag Manager накапливается большое количество тегов, триггеров и переменных. У нас есть возможность каждый объект в контейнере отнести к какой-то своей логической сущности. Иными словами, сгруппировать их таким образом, чтобы в дальнейшем было проще работать. А помогут нам в этой систематизации папки.

Группировать создаваемые внутри контейнера GTM элементы можно по-разному. Например:

- по инструментам (отдельно завести папку для Google Ads, отдельно для Google Analytics и т.д.);
- по типу решаемой задачи (отслеживание определенного события или группы событий);
- по типу выполняемой работы (отдельно завести папку для специалиста по контекстной рекламе, отдельно для человека, который занимается SEO, отдельно для веб-аналитика и т.д.);
- по проектам (для каждого микросайта или рекламной кампании);
- по подрядчикам (отдельно для агентства такого-то, организации такой-то и т.д.);

- как-то иначе.

Чтобы создать новую папку, перейдите в раздел **Папки** и выберите **Новая папка**.

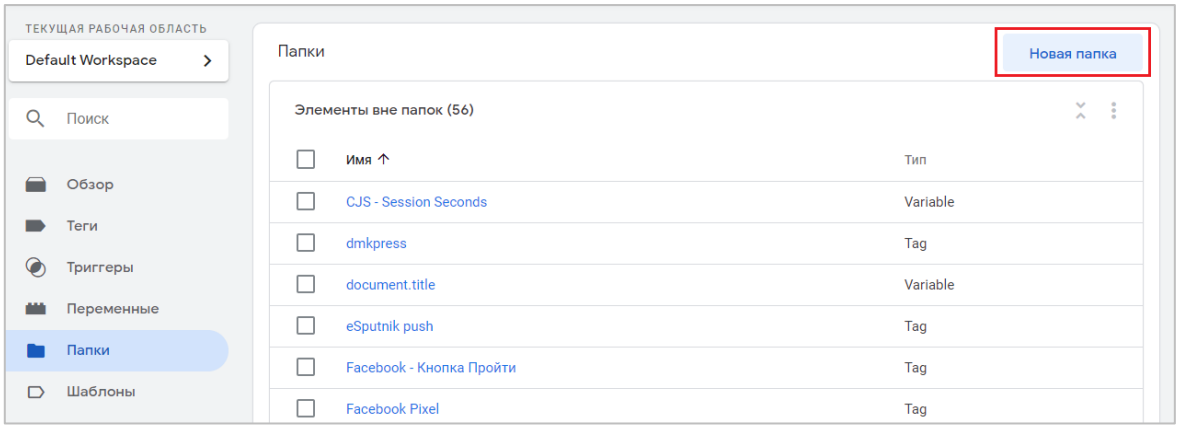


Рис. 116. Создание новой папки

Укажите ее название (например, *Папка Google Analytics*) и нажмите **Создать.** Имена папок следует задавать таким образом, чтобы из названия сразу было понятно, что находится в ней или к какой сущности принадлежат внутренние элементы.

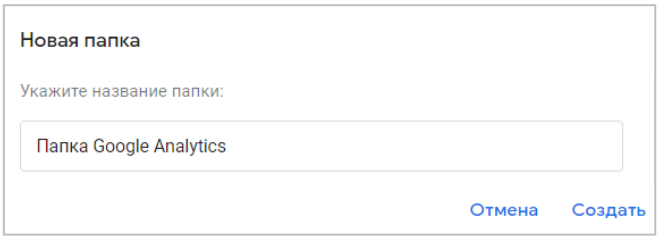


Рис. 117. Название папки

Папка создана и теперь может быть использована для группировки различных объектов контейнера GTM. Перенесем в нее элементы, относящиеся к Google Analytics. В нашем примере это переменная GA ID (идентификатор счетчика Google Analytics) и сам тег Universal Analytics.

Ставим галочки напротив этих объектов и вверху выбираем **Переместить – Папка Google Analytics**.

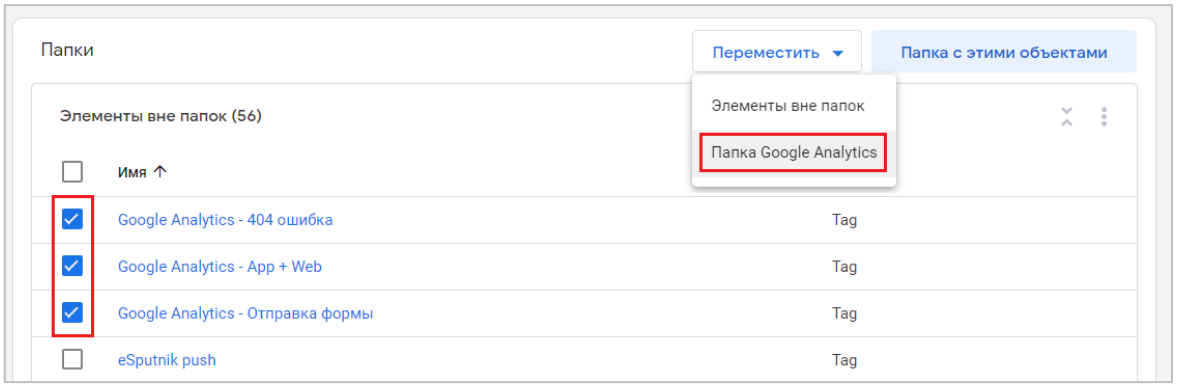


Рис. 118. Перемещение элементов в папку

После загрузки GTM выбранные элементы будут добавлены в папку.

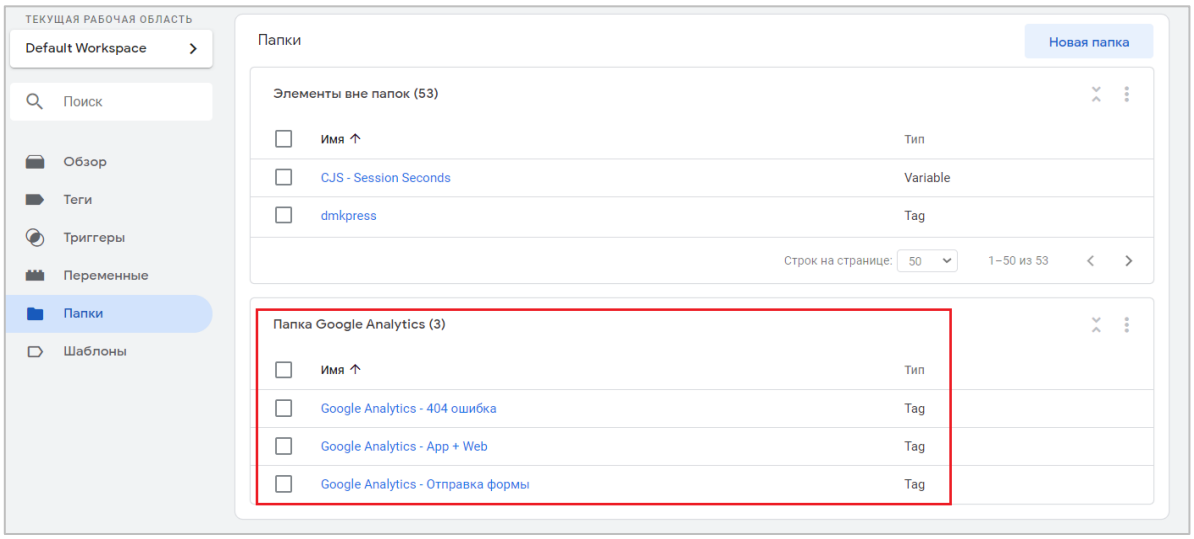


Рис. 119. Элементы в папке

Если нам необходимо исключить какой-либо объект из папки, то выбираем данный элемент и вверху таблицы выбираем **Переместить – Элементы вне папок**.

Чтобы добавить элементы в папку, не обязательно было сначала создавать новую папку, а затем перемещать их в нее. Можно было сначала выбрать необходимые объекты, а далее нажать на **Папка с этими объектами**. После чего Google Tag Manager предложит создать папку.

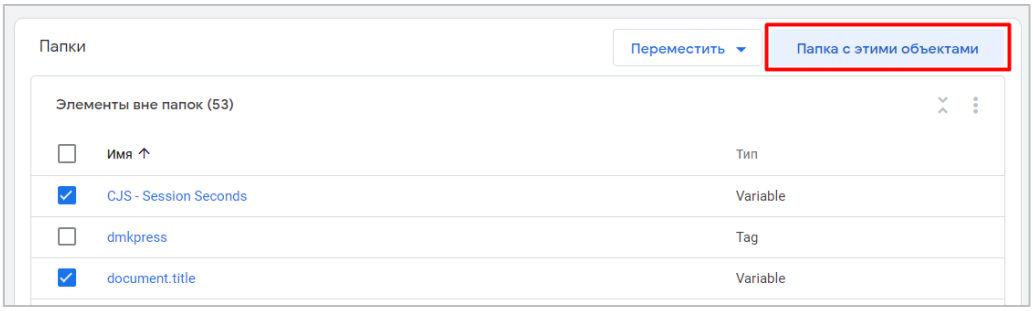


Рис. 120. Папка с этими объектами

Можно использовать оба варианта при группировке элементов в папки. Управлять папками и их содержимым можно с помощью меню.

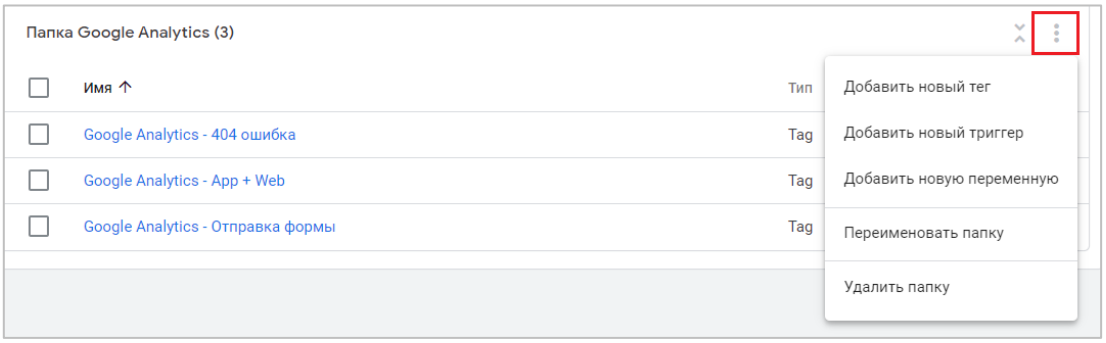


Рис. 121. Управление папками

Доступны:

- добавить новый тег;
- добавить новый триггер;
- добавить новую переменную;
- переименовать папку;
- удалить папку.

Если вы заходите удалить папку, в которой присутствуют теги, триггеры или переменные, то вы сначала должны очистить ее содержимое, сделать ее пустой, иначе GTM выдаст вам предупреждение:

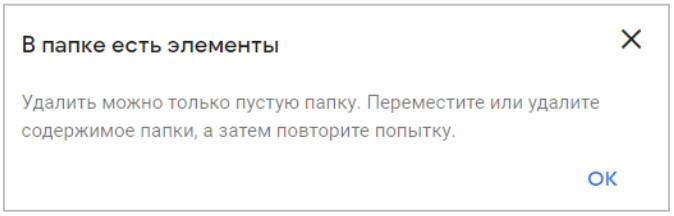


Рис. 122. Предупреждение Google Tag Manager

При создании новых объектов в GTM вы можете сразу указать конечную папку с помощью иконки:

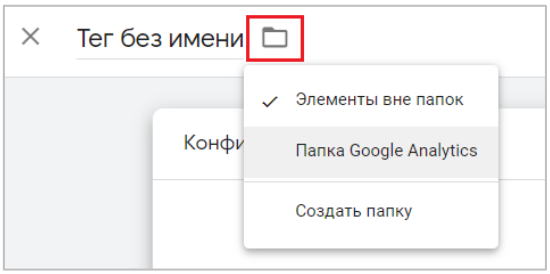


Рис. 123. Иконка папки

И последнее. После создания папок в таблицах тегов, триггеров и переменных появляется дополнительный столбец **Папка**, по которому можно сортировать элементы.

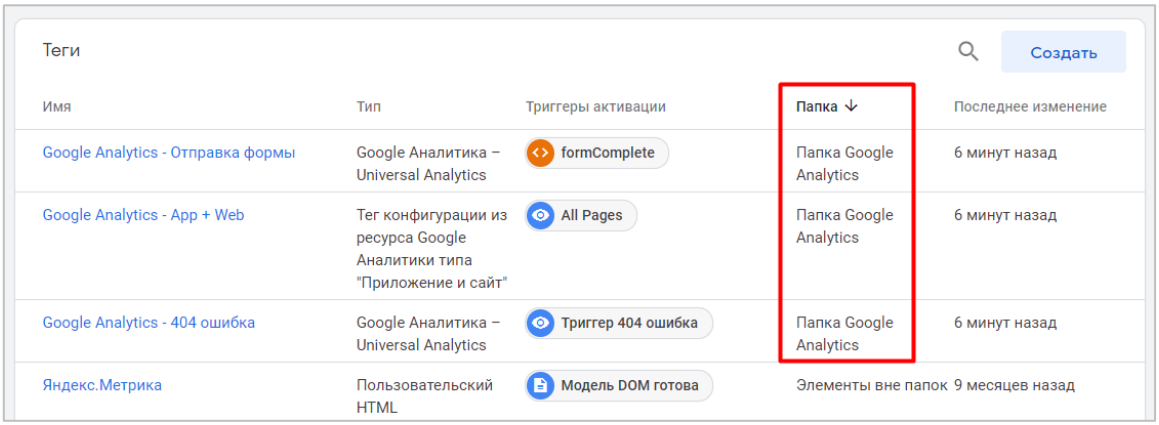


Рис. 124. Дополнительный столбец Папка

Режим предварительного просмотра

Режим предварительного просмотра позволяет проверить правильность настроек тегов Google Tag Manager перед публикацией контейнера, выявить текущие проблемы на этапе тестирования и снизить риск их возникновения в будущем. Режим предварительного просмотра в GTM еще называют режимом отладки.

Проще говоря, это этап, на котором обнаруживают, локализуют и устраняют ошибки. Почему не активируется тег, не срабатывает событие, не определяется переменная? Ответить на эти и многие другие вопросы поможет именно отладка.

Предположим, что в своем контейнере вы настроили какие-то теги, добавили триггеры, переменные, и теперь хотите проверить корректность их выполнения. Чтобы включить режим предварительного просмотра, в интерфейсе Google Tag Manager нажмите на кнопку **Предварительный просмотр**.

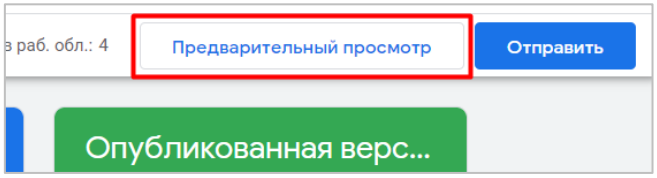


Рис. 125. Предварительный просмотр

Таким образом вы активируете режим предварительного просмотра и сверху в оранжевом прямоугольнике появится надпись:

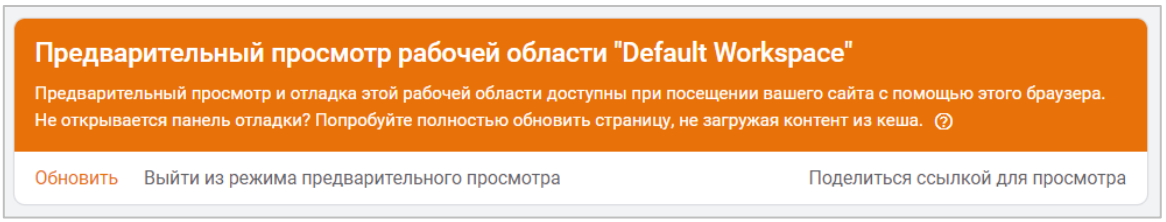


Рис. 126. Предварительный просмотр активирован

Доступны три функции:

- **Обновить** – если в процессе отладки, не выходя из режима, вы внесли изменения в тегах, триггерах или переменных, то после каждого такого изменения следует обновлять предварительный просмотр, иначе Google Tag Manager не увидит правок и будет показывать предыдущую версию контейнера;
- **Выйти из режима предварительного просмотра** – выход из режима отладки;

Перед выходом GTM переспросит у вас:

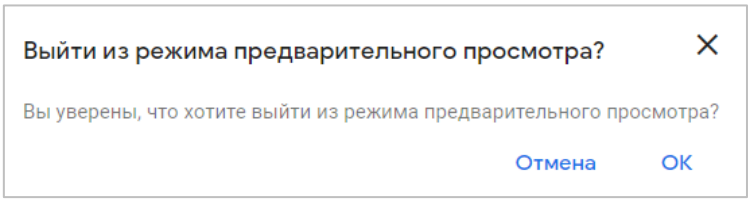


Рис. 127. Выйти из режима предварительного просмотра?

- **Поделиться ссылкой для просмотра** – отправить доступ другим людям;

Режим предварительного просмотра доступен только в том браузере, в котором был включен этот режим. Но вы можете предоставить доступ другим пользователям. Для этого необходимо нажать на кнопку **Поделиться ссылкой для просмотра**. Далее вводите целевой URL, разрешаете отладку при просмотре (gtm_debug=x), копируете ссылку и отправляете ее другому пользователю.

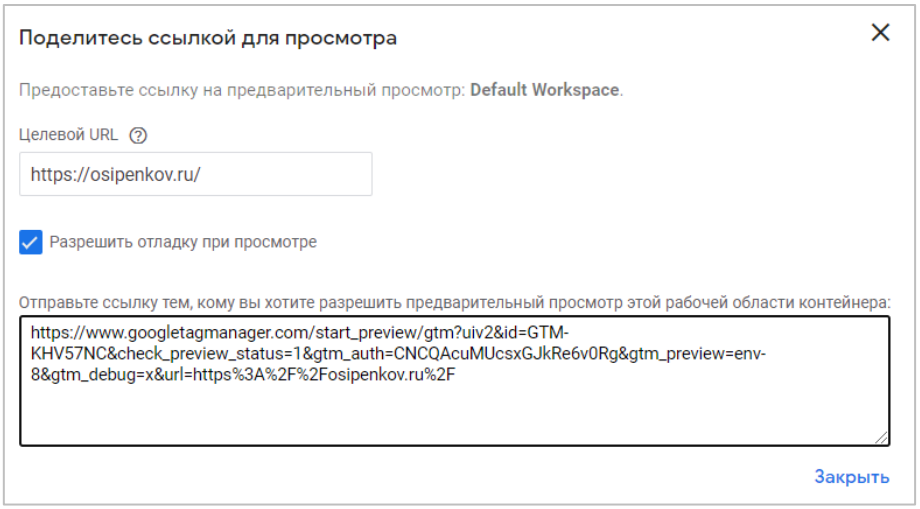


Рис. 128. Поделиться ссылкой для просмотра

Пользователь, в свою очередь, должен будет перейти по ссылке и кликнуть на целевой URL. После этого режим предварительного просмотра в его браузере будет активирован.

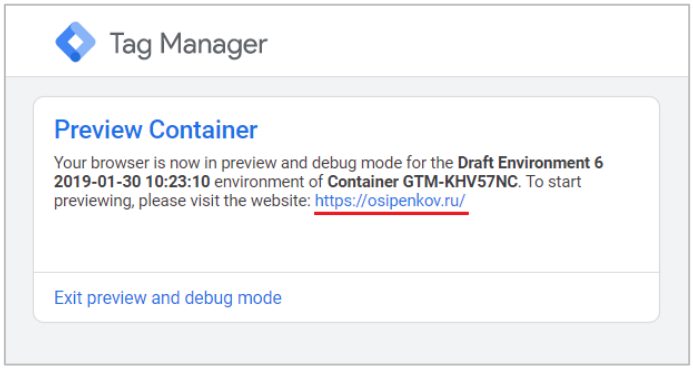


Рис. 129. Preview Container

После активации предварительного просмотра в соседней вкладке откройте ваш сайт. В нижней части окна браузера появится консоль отладки с подробной информацией о запущенных тегах, порядке их активации и собираемых данных. Эту консоль будете видеть только вы – другим посетителям сайта она недоступна.

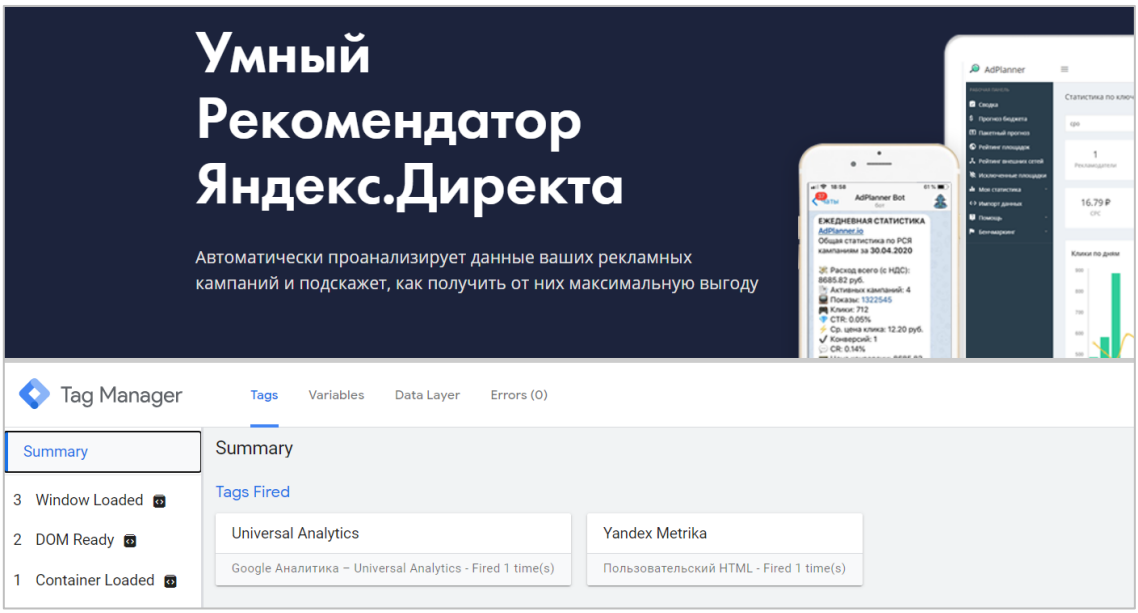


Рис. 130. Панель отладки

Если после проделанных работ вы не увидели нижнюю консоль, то это может быть связано с тем, что:

- вы используете какие-либо блокировщики, например, AdBlock или AdGuard (хотя в некоторых случаях GTM корректно работает и с ними);
- ваш антивирус блокирует всплывающие окна;
- сам браузер настроен таким образом, что воспрепятствует появлению дополнительных окон.

Если ничего из вышеперечисленного не помогло, попробуйте зайти в режим отладки с другого браузера или компьютера.

Давайте разберем полностью всю навигацию консоли отладки.

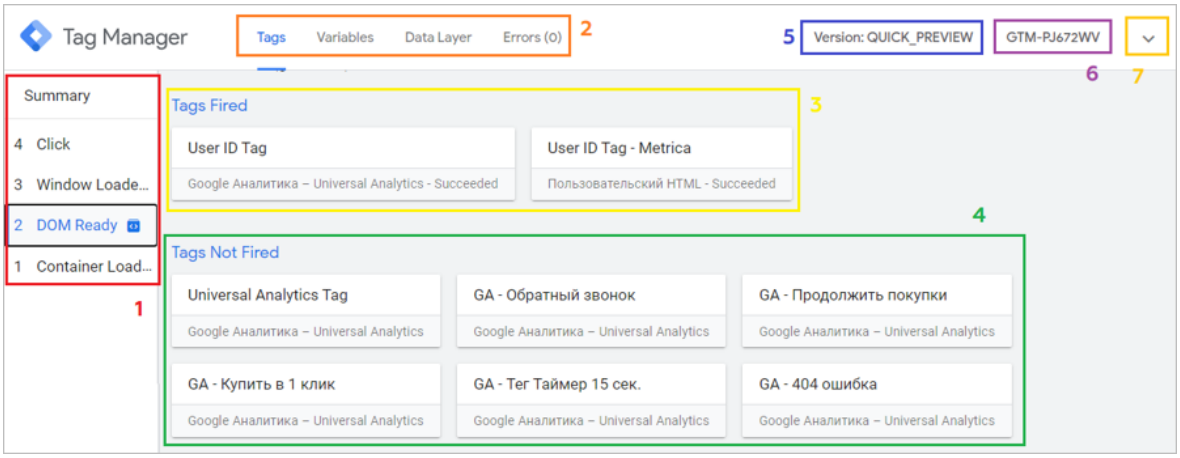


Рис. 131. Навигация режима отладки

1 - Сводка, шкала временных событий

В этом блоке отображаются все события в хронологическом порядке, которые зафиксировал Google Tag Manager. Диспетчер тегов передает на уровень данных 3 события вне зависимости от того, настроили вы что-то еще или нет. Это **gtm.js (Container Loaded), gtm.dom (DOM Ready)** и **gtm.load (Window Loaded)**. Цифра рядом с каждым событием означает порядок активации. В данном случае событие **Container Loaded (1)** было зафиксировано раньше остальных. Подробнее об этих трех событиях мы разберем в следующей главе.

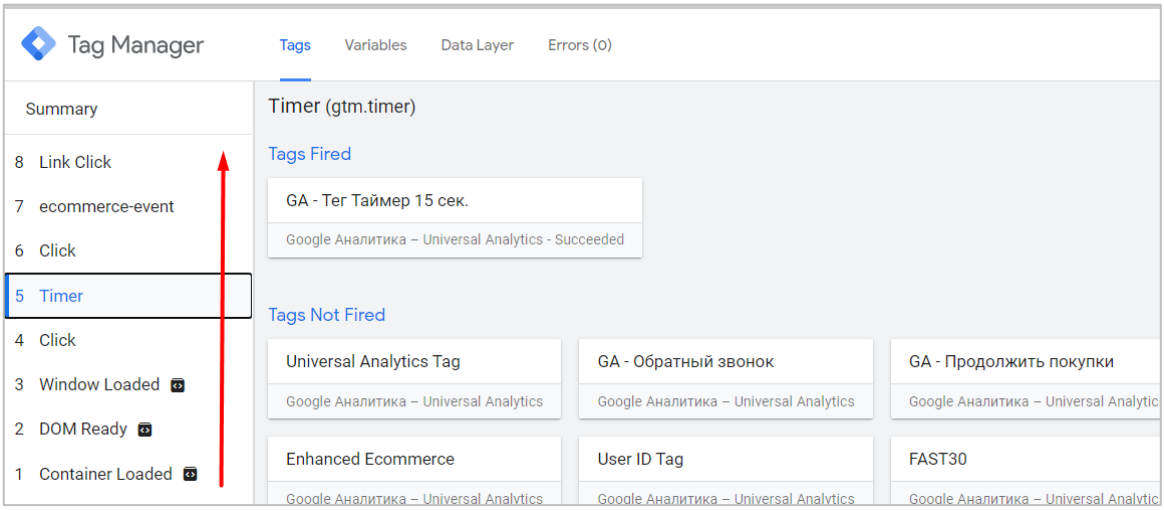


Рис. 132. Шкала событий

Если наверху выбрана вкладка **Tags (Теги)**, то мы можем увидеть какие теги были активированы по данному события, а какие нет:

- **Tags Fired On This Page** – теги, которые были активированы (цифра 3);
- **Tags Not Fired On This Page** – теги, которые не были активированы (цифра 4);

Если выбрана вкладка **Variables (Переменные)**, то мы получаем подробные сведения о переменных в выбранном событии в тот момент, когда оно произошло. GTM будет передавать те значения переменных, которые были активированы в интерфейсе, даже если они не определены (undefined).

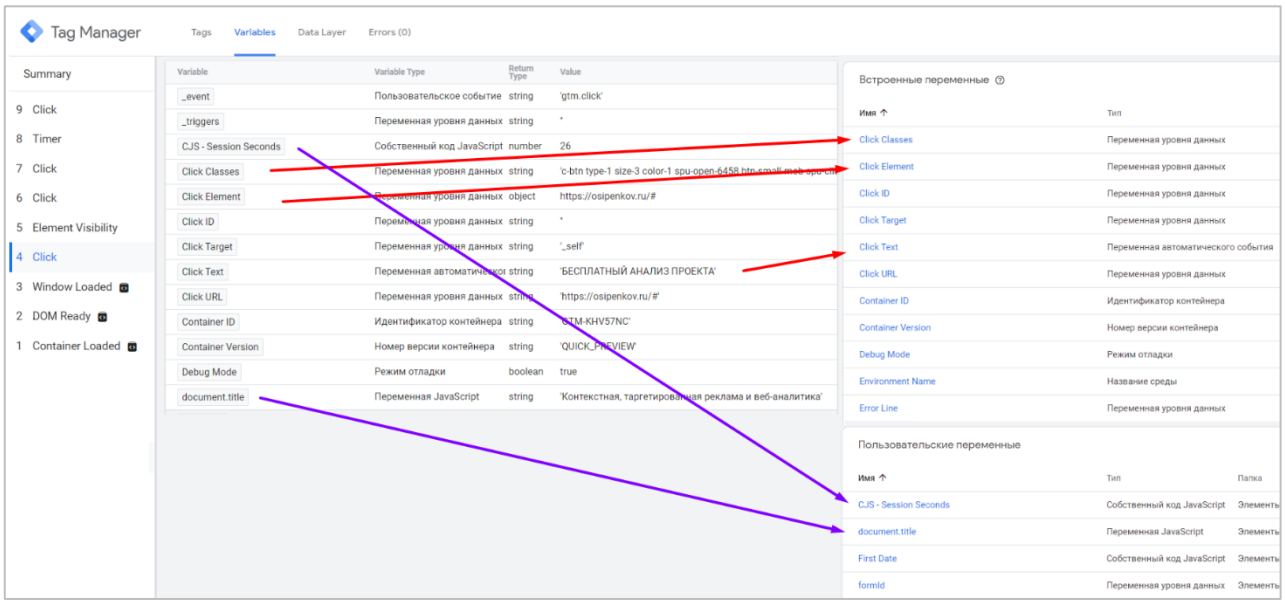


Рис. 133. Вкладка Variables

На вкладке **Data Layer (Уровень данных)** отображается информация объекта в двух состояниях:

1. в том виде, в каком она была передана в уровень данных в момент события;
2. содержимое уровня данных после того, как произошло событие.

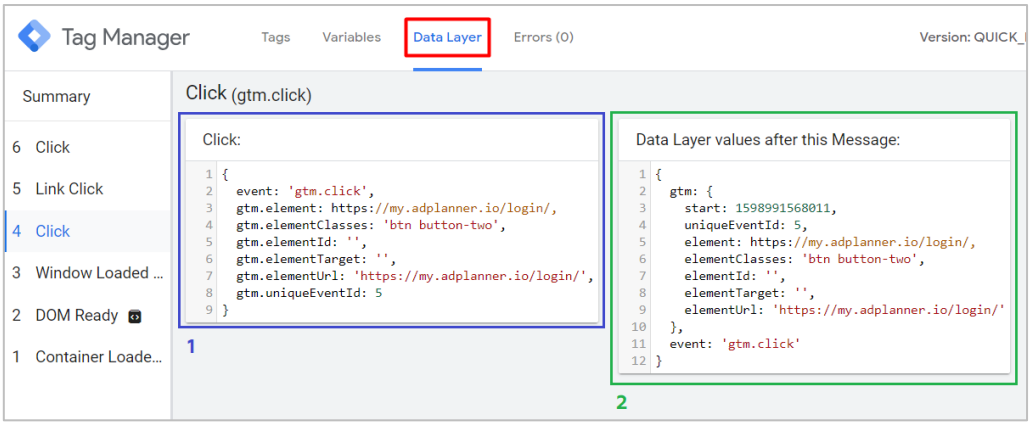


Рис. 134. Вкладка Data Layer

Верхний пункт **Summary** показывает общую информацию о работе всех тегов GTM, которые сработали на данной странице за все время работы независимо от времени их загрузки.

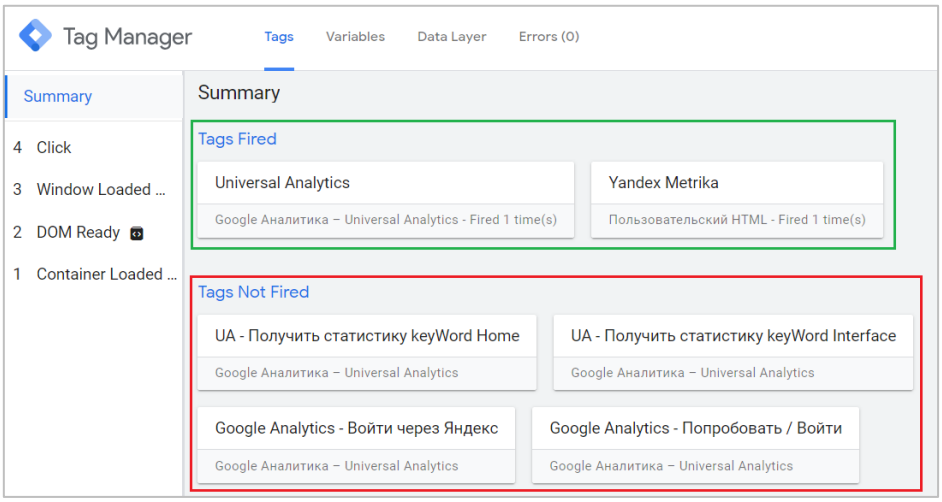


Рис. 135. Summary

Если тег сработал несколько раз, то в **Summary** рядом с этим тегом будет изменено значение **Fired .. time (s)**

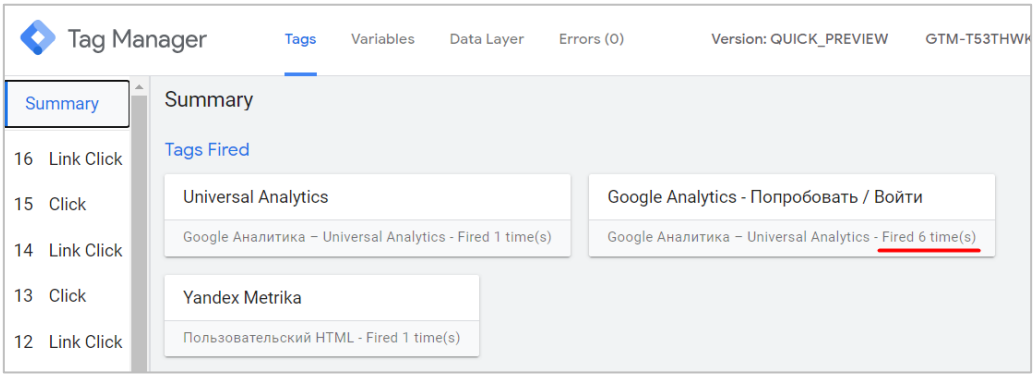


Рис. 136. Fired .. time(s)

В **Summary** информация по переменным недоступна, поскольку их значения меняются в зависимости от события.

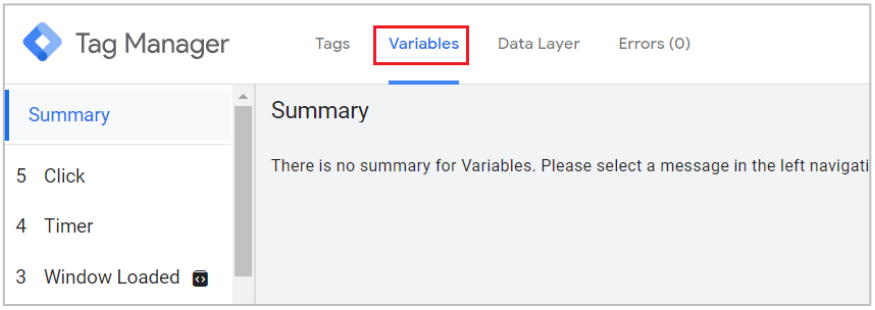


Рис. 137. Информация по переменным недоступна

Ее необходимо просматривать только тогда, когда вы выбираете конкретное событие.

В **Summary** на вкладке **Data Layer** выводится вся информация по уровню данных в хронологическом порядке – события с номерами, та информация, которая была передана по событию и общая информация об уровне данных на текущий момент, **Current values of the Data Layer** (блок выделен зеленым):

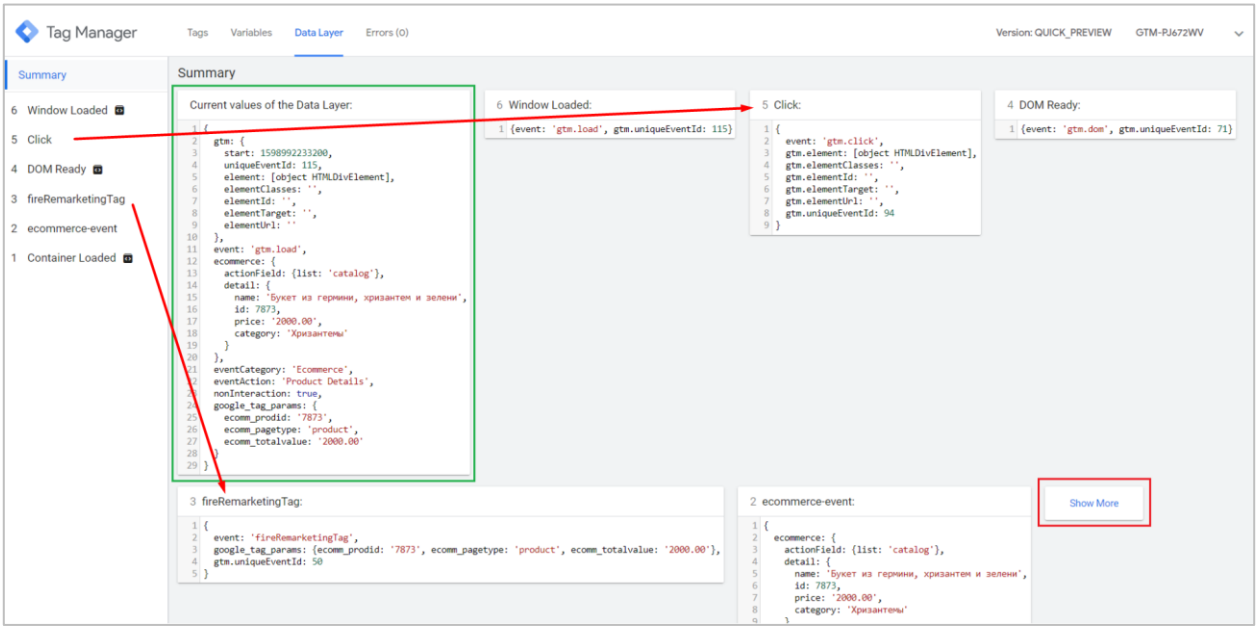


Рис. 138. Summary - Data Layer

Если событий было зафиксировано большое количество, то с помощью кнопки **Show more** можно просмотреть предыдущий список событий.

Возвращаемся назад к временной шкале событий и на вкладку **Tags**.

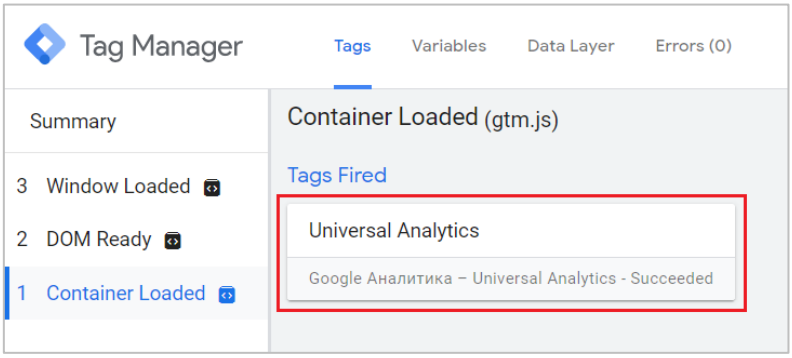


Рис. 139. Теги

Нажав на тег, можно просмотреть его свойства и значения, а также триггеры активации **(Firing Triggers)** и блокировки **(Blocking Triggers)**.

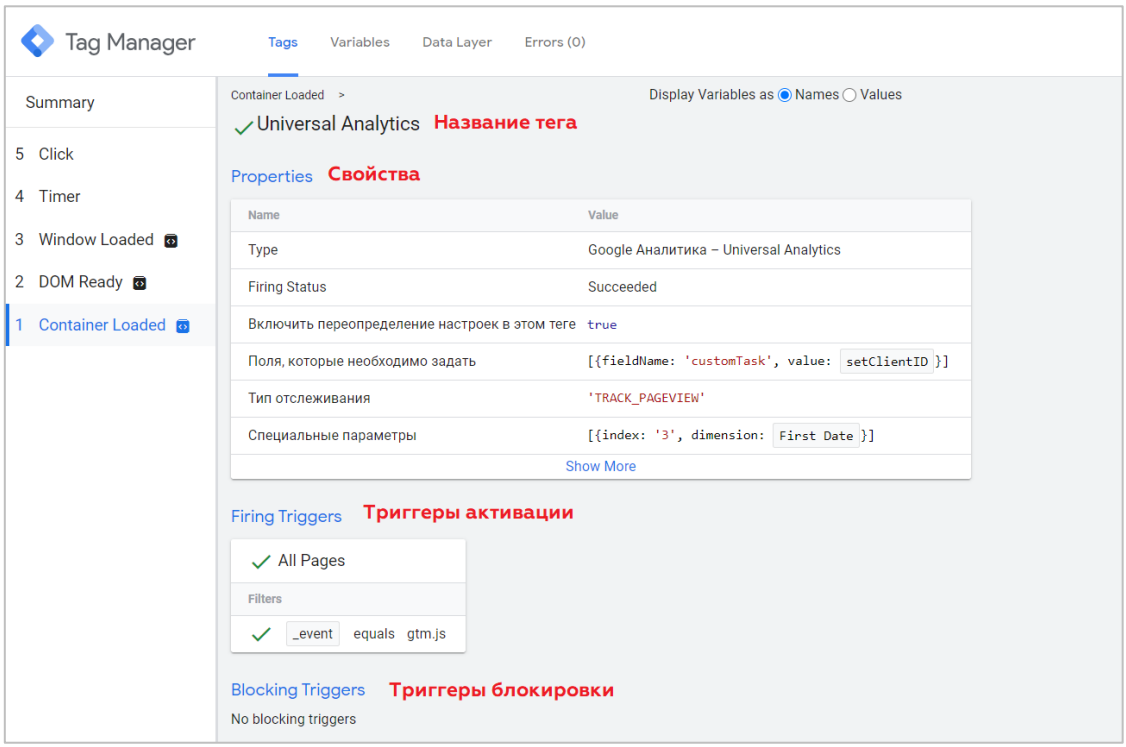


Рис. 140. Информация по тегу

Триггер активации сработал и нет (отличия):

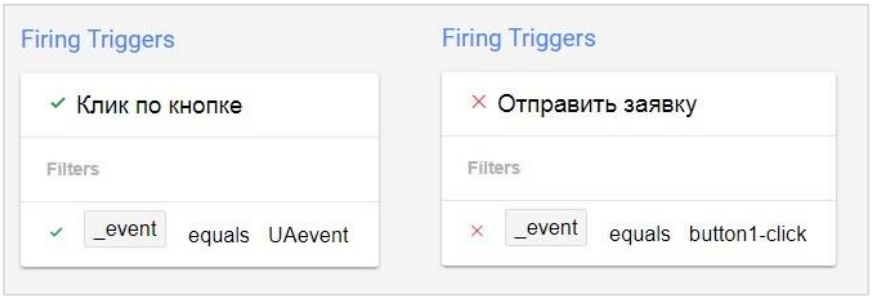


Рис. 141. Триггер активации сработал и нет

Активация триггера может состоять из нескольких условий. В этом случае какие-то отдельные условия могут сработать, а какие-то нет:

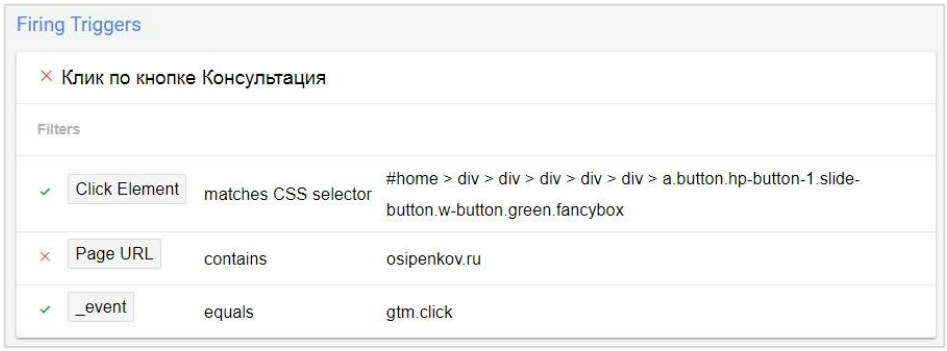


Рис. 142. Несколько условий активаций

При несоблюдении хотя бы одного условия триггер не сработает. В верхней части блока отладки на вкладке **Tags** есть настройка, которая называется **Display Variables as**.

Рис. 143. Display Variables as

Она позволяет выбрать режим представления переменных: в виде **имени (Names)** или в виде **значения (Values)** переменной в момент активации.

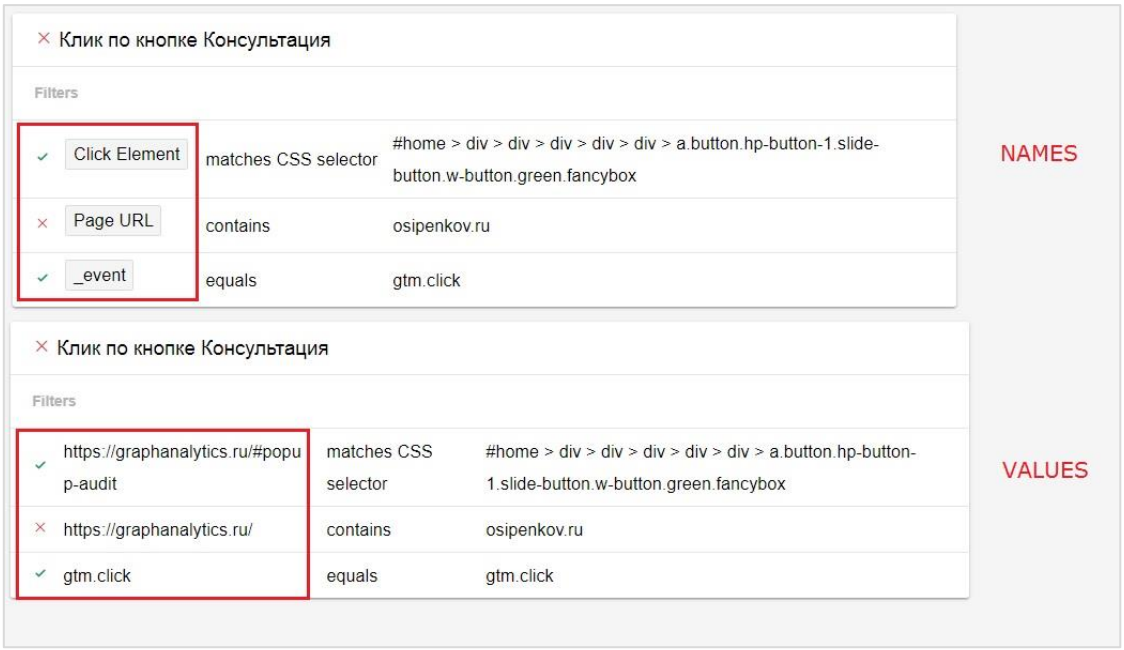


Рис. 144. Отличия NAMES от VALUES

На примере выше условия активации триггера изменились с **Click Element**, **Page URL** и **_event** (названия переменных) на значения переменных в момент активации.

Как мы уже знаем, для отслеживания различных действий пользователя в GTM предусмотрена специальная переменная **event**. Она используется внутри обработчика того или иного события. Если **event** отсутствует или не задано, то в режиме отладки мы увидим слово **Message** и надпись:

Рис. 145. Message

Если мы передаем через уровень данных пользовательское событие, то имя этого события выводится на временной шкале. Например, клик по кнопке зафиксировал пользовательское событие **Яша Осипенков**.

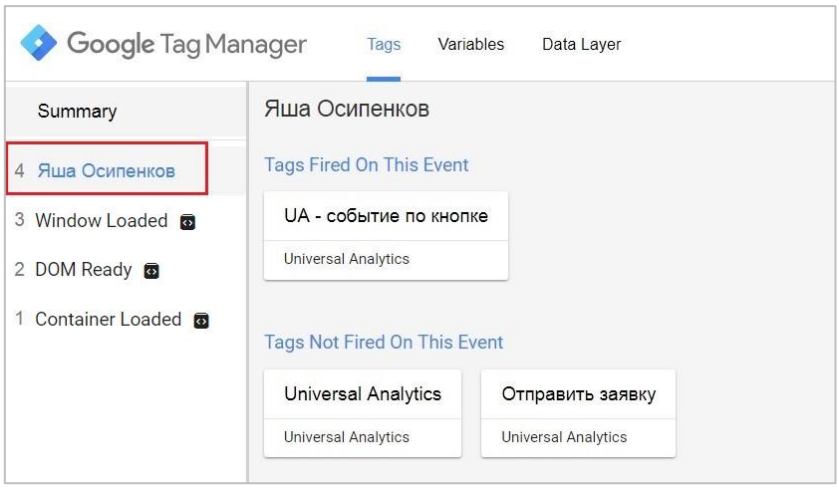


Рис. 146. Пример пользовательского события

Название может быть, как на кириллице, так и на латинице. Однако устоявшимся традиционным написанием пользовательских событий все же является латинский алфавит. А информация на уровне данных выглядит следующим образом:

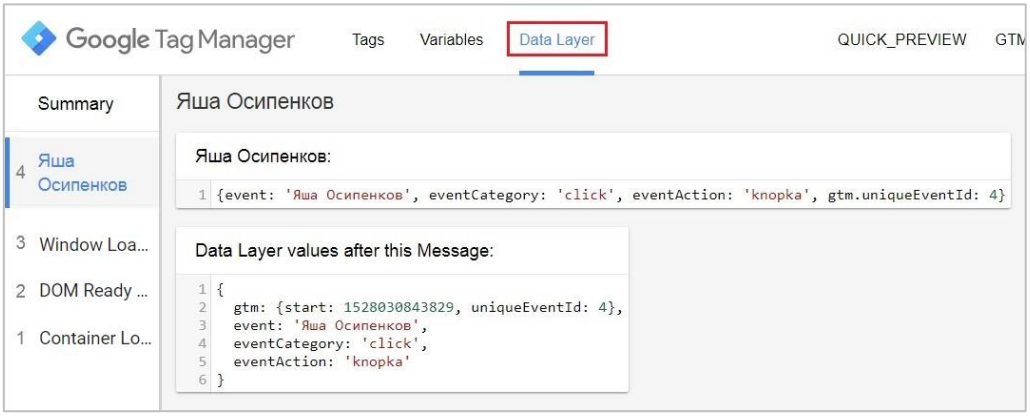


Рис. 147. Data Layer пользовательского события

5 - QUICK_PREVIEW

Рис. 148. Режим предварительного просмотра

Эта надпись свидетельствует о том, что в данный момент мы находимся в режиме предварительного просмотра (отладки) контейнера Google Tag Manager.

6 – Код контейнера

Рис. 149. Код контейнера GTM

Идентификатор контейнера диспетчера тегов Google.

7 – Сворачивание панели отладки

Рис. 150. Сворачивание панели отладки

Иногда панель отладки Google Tag Manager загораживает нижнюю часть экрана. В этом случае мы на некоторое время можем свернуть его для внесение каких-либо правок с помощью данной опции.

В свернутом состоянии панель отладки выглядит так:

Рис. 151. Debug

При клике на элемент вернется исходное состояние GTM. Помимо этого, управлять высотой панели можно с помощью ее перетягивания левой кнопкой мыши вверх-вниз.

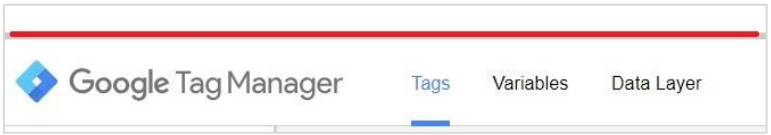


Рис. 152. Перетягивание панели по границе консоли

После того, как вы проверили корректность всех настроек в режиме отладки и внесли соответствующие изменения, их можно опубликовать. Для этого необходимо нажать на кнопку **Отправить** в правом верхнем углу интерфейса Google Tag Manager.

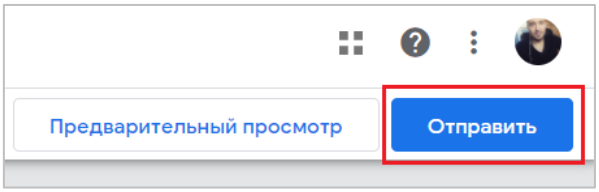


Рис. 153. Отправка контейнера после отладки

Указав название версии или создав новую, а также выбрав среду, опубликуйте контейнер.

Рекомендации: если вы проверяете корректность срабатывания событий Google Analytics, то используйте для этого группу отчетов **В режиме реального времени**.

Три события: Container Loaded, DOM Ready, Window Loaded

Каждый раз, когда загружается страница вашего сайта, в уровень данных передается информация о трех событиях: **Container Loaded (gtm.js), DOM Ready (gtm.dom)** и **Window Loaded (gtm.load).**

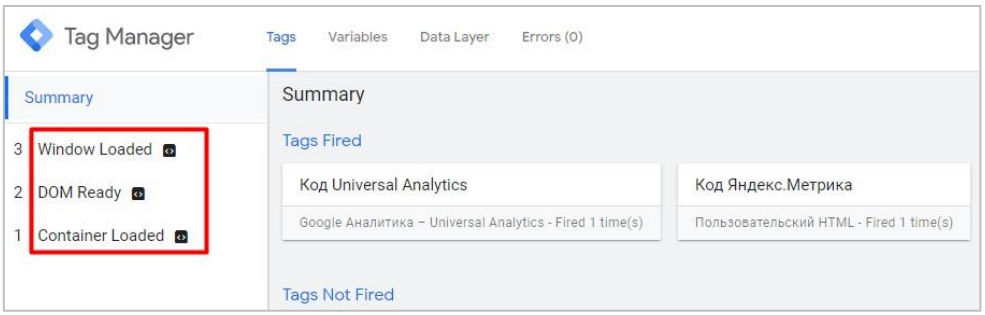


Рис. 154. Режим отладки - Три события по умолчанию

Если вы перейдете в консоль разработчика (клавиша F12 в Google Chrome) на вкладку **Console**, введете **dataLayer** и нажмете **Enter**, то увидите тот же самый результат:

Рис. 155. dataLayer в консоли разработчики

Что же означают эти события?

Жизненный цикл HTML-страницы состоит из трех важных событий: **DOMContentLoaded, load** и **beforeunload/unload**. Первое отвечает за полную загрузку HTML и построение DOM-дерева, второе за окончательную прогрузку HTML и всех внешних ресурсов (включая картинки, стили и т.д.), а последнее срабатывает, когда пользователь покидает страницу сайта.

Подробнее об этом читайте в документации **learn.javascript.ru** (см. приложение). Важно отметить, что два из трех вышеперечисленных событий по умолчанию срабатывают и в Google Tag Manager.

1. Container Loaded (gtm.js)

Это первое событие, которое запускает триггер **gtm.js**, помещается в dataLayer и срабатывает тогда, когда контейнер GTM готов к работе.

Не так давно разработчики Google переименовали данное событие в **Container Loaded**. Раньше это событие называлось **Page View (Просмотр страницы)**, что несколько вводило интернет-маркетологов в заблуждение, поскольку в Google Tag Manager по умолчанию существует триггер со схожим названием **Page View (All Pages)**, который используется при настройке отслеживания тегов на всех страницах сайта. Да и Page View меньше охарактеризовало себя как событие, поскольку правильнее было бы говорить именно о загрузке самого контейнера Google Tag Manager, а не о просмотре страницы.

Примечательно, что данное событие содержит в себе отметку времени в **gtm.start**:

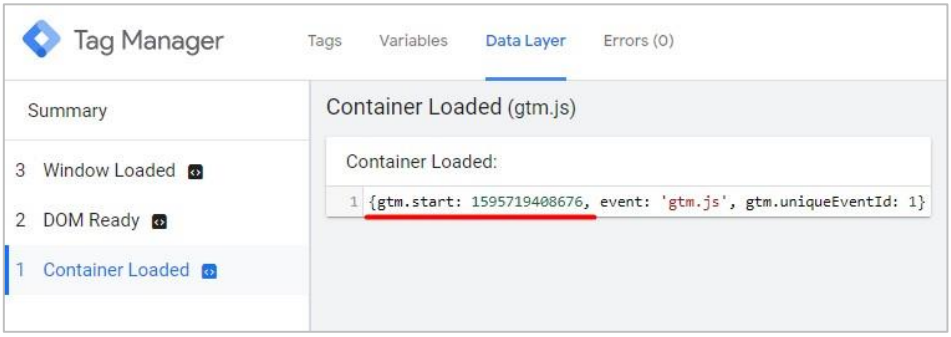


Рис. 156. gtm.js с отметкой времени в gtm.start

В дальнейшем мы можем использовать переменную уровня данных **gtm.start** для своих целей, например, для определения времени начала пользовательской сессии.

Как вы уже знаете, события в режиме отладки отображаются в хронологическом порядке по мере их запуска. Каждое из них имеет свой порядковый номер. **Container Loaded** - первое событие на представленном примере.

Примечание: некоторые события могут срабатывать ДО загрузки контейнера GTM. В этом случае Container Loaded может иметь другой порядковый номер, отличный от единицы. Например:

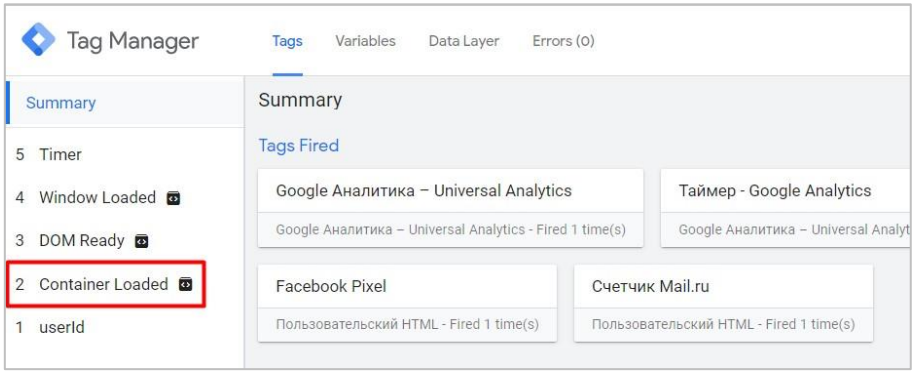


Рис. 157. Container Loaded загружается вторым по порядку

Здесь Container Loaded загружается вторым по порядку, до загрузки контейнера Google Tag Manager уже доступна информация по userId.

2. DOM Ready (gtm.dom)

Событие, которое запускает триггер **gtm.dom**, помещается в dataLayer и срабатывает, как только браузер загрузит объектную модель документа (Document Object Model). Событие происходит, когда весь HTML был полностью загружен и пройден парсером, не дожидаясь окончания загрузки таблиц стилей, изображений и фреймов.

3. Window Loaded (gtm.load)

Событие, которое запускает триггер **gtm.load**, помещается в dataLayer и срабатывает после завершения загрузки всей страницы и всех связанных ресурсов (загрузки таблиц стилей, изображений, фреймов), когда окно полностью загружено.

Разница в скорости загрузки

Для наглядности сравним время, через которое срабатывает каждое представленное событие. Воспользуемся расширением для браузера **Adswerve - dataLayer Inspector+** и вкладкой **Console**. Обновив страницу, мы увидим переданные события с временной меткой:

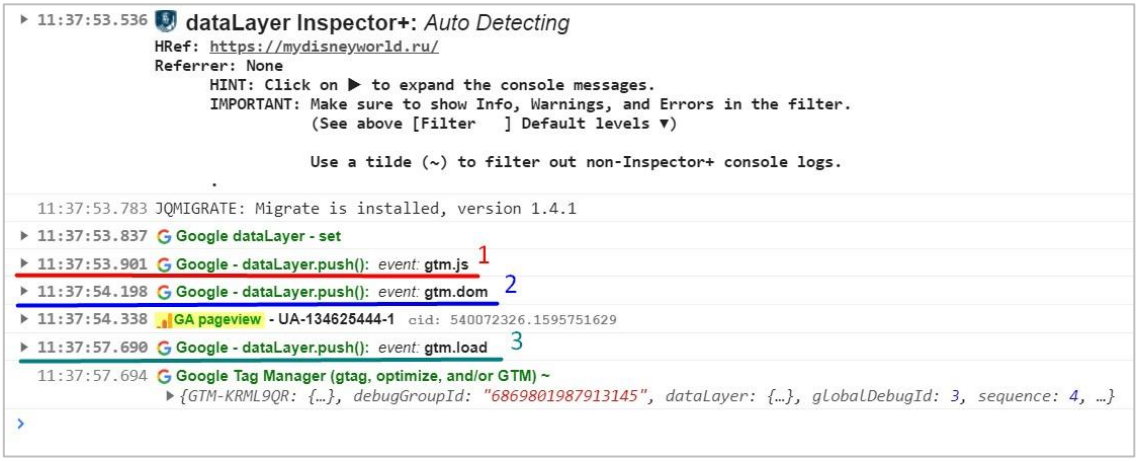


Рис. 158. Хронология с событиями и таймингом (расширение dataLayer Inspector+)

Примечание: временная метка включается в Google Chrome через **Настройки - раздел Console - галочка Show timestamps**.

На примере выше у нас событие gtm.js произошло в **11:37:53.901**, событие gtm.dom в **11:37:54.198**, а последнее событие gtm.load в **11:37:57.690**, то есть через 0,790 секунды позже, чем загрузился контейнер GTM, и на 0,492 секунды позже, чем загрузилась объектная модель документа (DOM).

Аналогично можно увидеть различия, если мы перейдем на вкладку **Network** и обновим страницу:

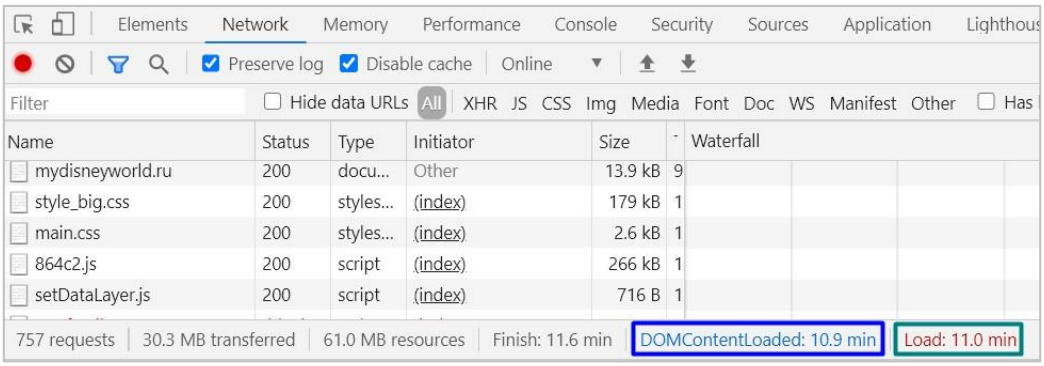


Рис. 159. Консоль разработчика - вкладка Network

Если сайт тяжелый, имеет большое количество неоптимизированных картинок, долго загружаются скрипты, вы зашли на него впервые (страница не закэширована), то разница между **DOMContentLoaded** и **Load** может быть существенной.

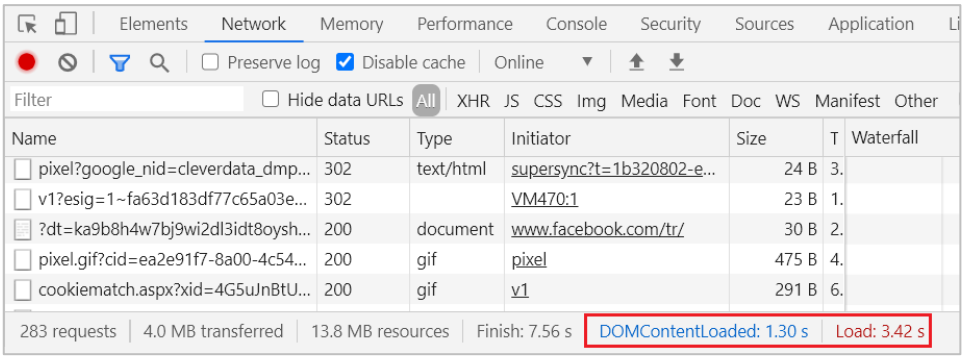


Рис. 160. Консоль разработчика - вкладка Network. Разница более 3 секунд

И вот как раз здесь возможна ситуация, при которой вы будете пытаться прикрепить слушатель к элементу, которого еще не существует. Триггер GTM сработает, но информация через тег в инструменты веб-аналитики не попадет, поскольку не будет еще определена.

Какое из трех событий выбрать в качестве триггера?

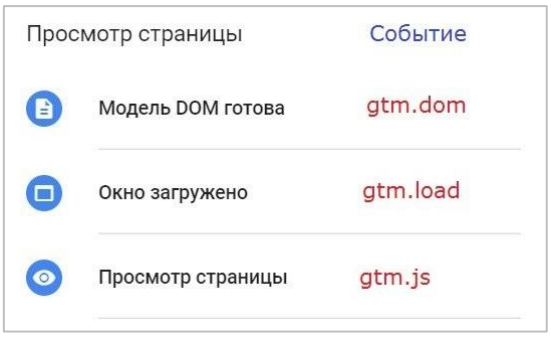


Рис. 161. Триггеры Просмотр страницы

Зависит от конкретной задачи и от момента, когда ваши данные будут готовы к отправке в инструменты аналитики:

- если информация доступна до загрузки контейнера Google Tag Manager, и вы хотите, чтобы ваши теги срабатывали в самый ранний момент, вы можете отправлять данные сразу же, используя триггеры **Пользовательское событие** с именем **gtm.js** или **Просмотр страницы (Page View)**.
- если вы хотите, чтобы ваши теги запускались после загрузки объектной модели документа, вы можете использовать триггеры **Пользовательское событие** с именем **gtm.dom** или **Модель DOM готова (DOM Ready)**.
- если вы не знаете, когда точно появляется доступ к элементу на странице, вы можете использовать триггеры **Пользовательское событие** с именем **gtm.load** или **Window Loaded (Окно загружено)**.

События для одностраничных сайтов (SPA)

Одностраничное приложение (SPA, Single Page Application) — это веб-сайт / веб-приложение, основанное на JavaScript-фреймворках, которое загружает весь контент, необходимый для навигации по сайту, при загрузке первой страницы. А когда пользователь взаимодействует со страницей, контент динамически подается через JavaScript вместо отправки запроса на сервер.

Поэтому, когда вы устанавливаете Google Tag Manager на своем SPA-сайте, первая загрузка запустит все три события **gtm.js, gtm.dom, gtm.load**. Но вы не увидите других таких событий, поскольку страница не перезагружается. Чтобы отслеживать последующие страницы в одностраничных приложениях, вам нужно использовать *виртуальные страницы*, а само отслеживание можно на основе триггера типа **Изменение в истории (gtm.historyChange)**. Подробнее про виртуальные страницы будет разобрано в следующих главах.

Глава 3

CSS-селекторы

CSS-селекторы в Google Tag Manager

В процессе настройки аналитики часто приходится отслеживать клики по определенным элементам на странице сайта. Все просто, если у отслеживаемых элементов (кнопки, ссылки, формы и т.д.) есть атрибуты **id** или **class**. Например, как здесь (инспектирование кода веб-страницы в консоли разработчика):

Рис. 162. Атрибуты id и class у элемента на сайте

В Google Tag Manager для активации тега на данный элемент остается только настроить триггер со встроенными переменными **Click Classes** и **Click ID**.

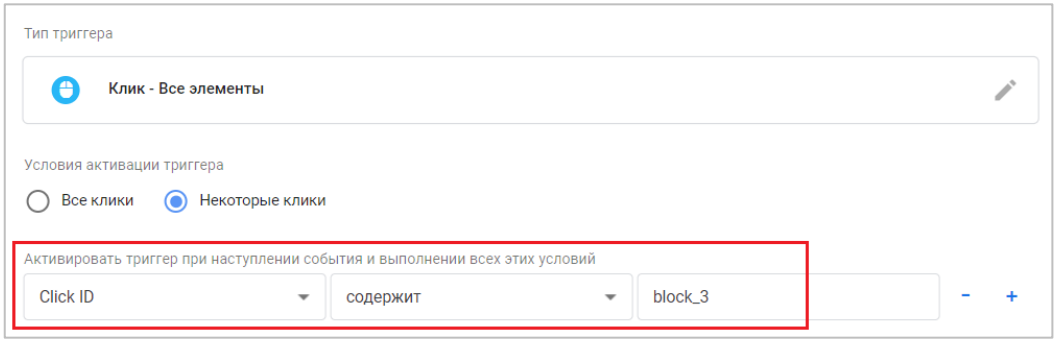


Рис. 163. Пример условия активации триггера по Click ID

Но часто бывает, что у необходимого элемента нет таких атрибутов. Тогда на помощь приходят css-селекторы.

Селектор — это часть CSS-правила, которая сообщает браузеру, к какому элементу (или элементам) веб-страницы будет применен стиль. CSS-селекторы применяются в Google Tag Manager для условий активации триггеров, а также в пользовательских переменных, таких как **Видимость элемента**, **Элемент DOM** и **Собственный код JavaScript**.

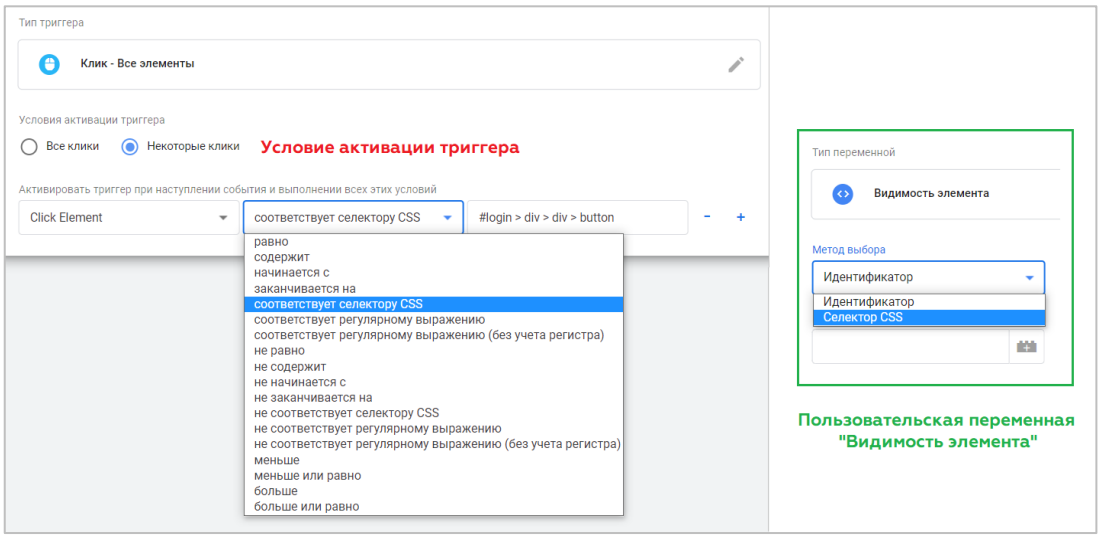


Рис. 164. CSS-селекторы в триггерах и переменных

Если атрибут нужного элемента не повторяется на странице, то рекомендуется использовать именно встроенные переменные. Но если один и тот же класс используется в нескольких элементах на странице, тогда рекомендуется использовать селекторы CSS.

Как правило, большинство настроек с использованием селекторов CSS приходится на связку **соответствует селектору CSS + Click Element** или **Form Element.**

Но прежде, чем разбираться в селекторах, рекомендую для начала ознакомиться с глобальной структурой документа в формате HTML – какие бывают элементы, заголовки, атрибуты, метаданные и т.д.

Подробнее обо всех элементах HTML читайте в руководстве разработчиков Mozilla (см. приложение). Чаще всего бы будете встречаться с такими тегами, как **<p>, <a>, , , <h1>...<h6>**.

Кроме этого, вы должны познакомиться с объектной моделью документа **(Document Object Model, DOM)**. Согласно DOM-модели, документ является иерархией, деревом. Каждый HTML-тег образует узел дерева с типом элемент. Вложенные в него теги становятся дочерними узлами.

Прежде чем в Google Tag Manager настраивать триггеры на отслеживание прокрутки, взаимодействия пользователей с видео, электронную торговлю, вам необходимо понять, что из себя представляет дерево DOM.

DOM определяет логическую структуру документа, способы доступа к элементам и их изменению. Документ может быть как HTML, так и XML. Когда DOM используется для определения логической структуры HTML-документа, значит речь идет о HTML DOM. Давайте рассмотрим пример:

HTML ▼

```
<html>
<head>
<title>Google Tag Manager для googлят</title>
<meta name="description" content="Интернет-маркетинг и веб-аналитика"/>
</head>
<body>
<h2>Руководство по управлению тегами</h2>
<p>2018</p>
<table>
<tr>
<td>Автор: Яков Осипенков</td>
</tr>
</table>
</body>
</html>
```

Рис. 165. Пример HTML-документа

Представление DOM этого документа HTML выглядит следующим образом:

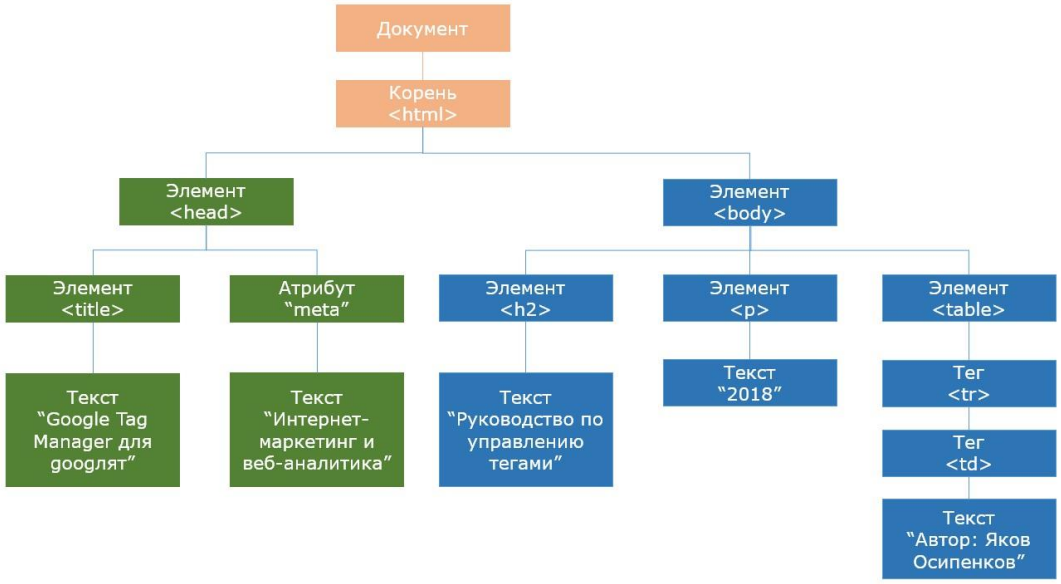


Рис. 166. Представление DOM для документа HTML

Здесь html, head, body, title, meta, h2, p, table, tr, td, Google Tag Manager для googлят, Интернет-маркетинг и веб-аналитика, 2018 и т.д. являются примерами объектов DOM. Они также называются элементами или узлами. Нижние блоки с пометкой *Текст* являются текстовыми узлами.

Чтобы отслеживать клики/события через GTM по определенным элементам на странице, необходимо понимать взаимосвязь между различными узлами документа.

DOM нужен для того, чтобы манипулировать страницей – читать информацию из HTML, создавать и изменять элементы. Получив узел, мы можем что-то сделать с ним. Все узлы в документе имеют иерархическое отношение друг к другу. Существует несколько специальных терминов для описания отношений между узлами:

- **родительский узел (parent node)** - родительским узлом по отношению к рассматриваемому объекту является узел, в который вложен рассматриваемый объект.

На примере выше по отношению к узлам *<h2>*, *<p>* и *<table>* элемент **<body>** является родительским. Для узла *<title>* родительским является узел **<head>**.

- **узлы-потомки (child node)** - узлом-потомком по отношению к рассматриваемому объекту является узел, который вложен в рассматриваемый объект. На нашей схеме по отношению к узлу *<body>* **<h2>**, **<p>** и **<table>** являются потомками. Для узла *<head>* потомками являются **<title>** и атрибут **“meta”**.
- **узлы-братья (sibling node)** – узлы, находящиеся на одинаковом уровне вложенности по отношению к их родительскому узлу. На нашей схеме узлами-братьями являются **<body>** и **<head>** по отношению к корневому *<html>*, а также **<h2>, <p>** и **<table>** по отношению к родительскому узлу *<body>*.

Самый верхний узел в DOM называется корневым. Так как объект document не является частью DOM, в нашем примере корневым является **<html>**. Он не имеет родителей и сам является родительским узлом по отношению к *<head>* и *<body>*.

В интернете в различных материалах можно встретить разные термины – предок, потомок (прямой и непрямой), сыновья, дочери, дети, ребенок, братья, сестры, правнуки и т.д. По сути, это одно и тоже. То, как вы привыкнете их называть, не имеет значения. Главное – это правильно понимать взаимосвязь между ними и уметь к ним обращаться.

Каждый объект HTML, кроме текстовых узлов, имеет свойства (атрибуты) и методы. Например:

```
<a href="#web-analytics" class="purchase-btn tariff-order fancybox"
onclick="dataLayer.push({'event':'buttonClick','eventCategory':'click',
'eventAction':'knopka'});">Весь перечень</a>
```

Рис. 167. Фрагмент кода

- **<a>** - это элемент HTML;
- **href, class** – атрибуты элемента <a>;
- **dataLayer.push ()** – это метод, который выполняется в ответ на событие **onclick** (возникает при щелчке левой кнопкой мыши на элементе, к которому добавлен атрибут onclick).

Google Tag Manager по умолчанию отслеживает все основные элементы и атрибуты, но не отслеживает кастомные (пользовательские) атрибуты.

Когда мы используем CSS-селекторы, мы находим элементы в DOM. Для управления внешним видом HTML-документа существует формальный язык CSS. CSS это аббревиатура **Cascading Style Sheets** или **Каскадные Таблицы Стилей**. Обычно CSS называют просто стили. На текущий момент существует несколько спецификаций (CSS1, CSS2, CSS2.1, CSS3, CSS4). Последняя содержит незначительные изменения, и поэтому самой популярной и часто используемой на данный момент является CSS3.

Как правило, все стили веб-сайта выносятся в отдельный файл с расширением **.css**, к которому впоследствии можно обратиться по определенному пути.

Селекторы могут быть:

- *простыми* (напрямую идентифицируют элемент или несколько элементов страницы на основании типа элемента (или его **класса (class)** или **id**.);
- *по атрибутам* (позволяют искать элементы по их атрибутам **attributes** и значениям этих атрибутов. Делятся на две категории: селекторы наличия и значения атрибута и селекторы подстроки в значении атрибута);
- *псевдоселекторами* (те, что выбирают только часть элемента или только элементы в определенном контексте. Бывают двух типов - псевдоклассы и псевдоэлементы);
- *комбинированными* (селекторы и их взаимосвязи между друг другом выражаются посредством комбинаторов);

Давайте рассмотрим каждый вид подробнее.

Простые селекторы

К ним относятся: *селекторы классов (class selectors)* и *селекторы по ID*.

Селектор класса состоит из точки (**.**), за которой следует имя класса. Имя класса — это любое значение, не содержащее пробелов в HTML-атрибуте **class**.

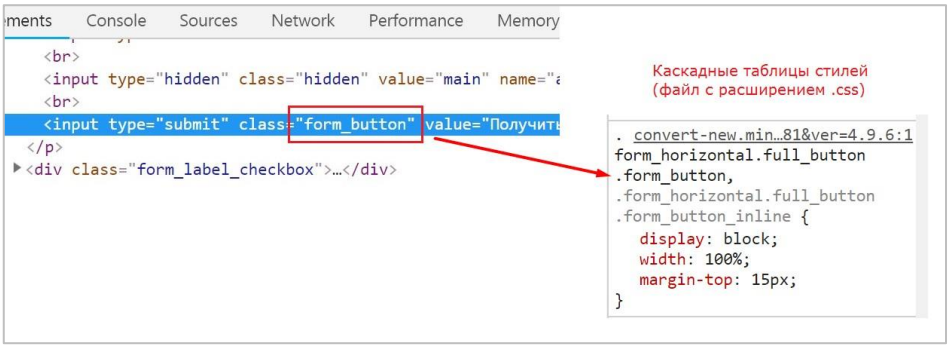


Рис. 168. Пример селектора классов

Пример **.form_button**. Это означает, что css-селектор **.class** соответствует всем элементам с классом .form_button (class="form_button").

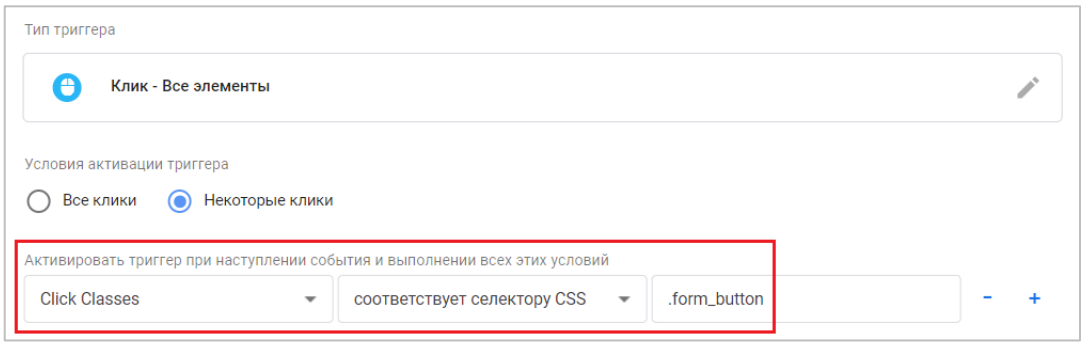


Рис. 169. Клик по элементам, соответствующим CSS-селектору класса .form_button

ID-селектор состоит из символа решетки (**#**), за которым следует ID нужного элемента. Каждый элемент может иметь уникальный идентификатор, назначаемый атрибутом **id**.

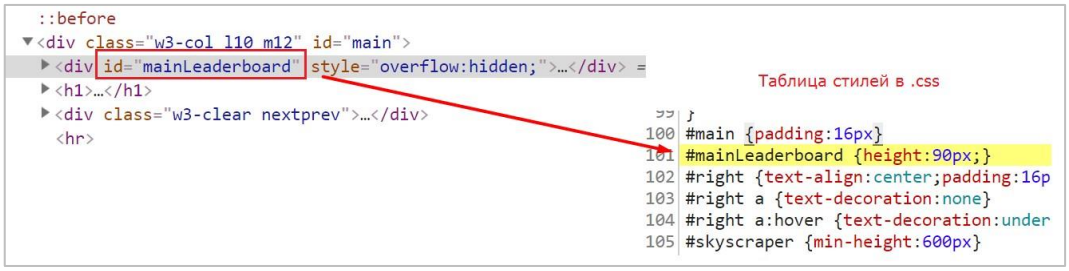


Рис. 170. Пример селектора по ID

Пример **#mainLeaderboard**. Это означает, что css-селектор **#id** соответствует всем элементам с идентификатором maiLeaderboard (id=”mainLeaderboard”).

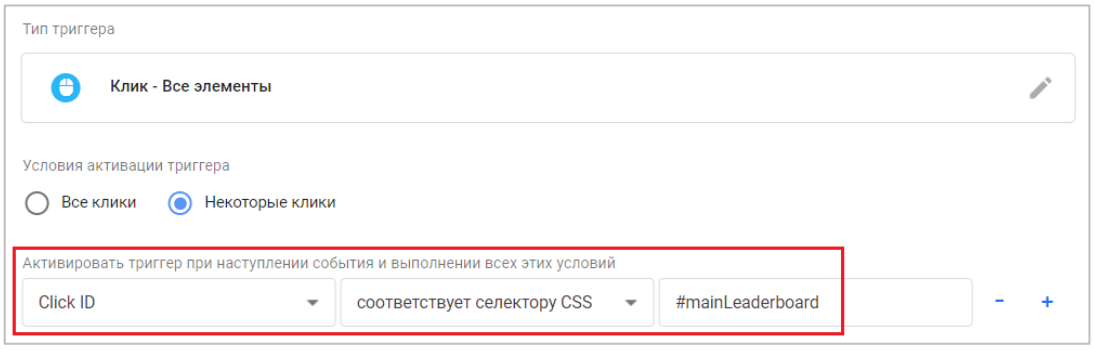


Рис. 171. Клик по элементам, соответствующим CSS-селектору с идентификатором #maiLeaderboard

Для выбора всех элементов на странице существует универсальный селектор (*). Помните этот спецсимвол в регулярных выражениях? Там он соответствует 0 или более предыдущих элементов, а здесь служит для одновременного установления одного стиля для всех элементов веб-страницы. Например, чтобы задать шрифт или начертание текста.

Рассмотрим пример, в котором нам необходимо отследить клик по любому блоку внутри блока ** **:

```
<div class="block">
<div class="firstscreen dark-1>...</div>
<div class="secondscreen dark-1>...</div>
<div class="thirdscreen dark-1>...</div>
<div class="fourthscreen dark-1>...</div>
<div class="fifthscreen dark-1>...</div>
<div class="sixthscreen dark-1>...</div>
</div>
```

Настроить в GTM триггер на клик по элементу с классом "block", используя встроенную переменную Click Classes = “block”, не получится, так как сам клик будет приходиться на один из узлов-потомков блока block. А настраивать на каждый внутренний элемент (firstscreen dark-1, secondscreen dark-1 и т.д.) не целесообразно.

По этой причине будем использовать универсальный селектор * для класса **.block**:

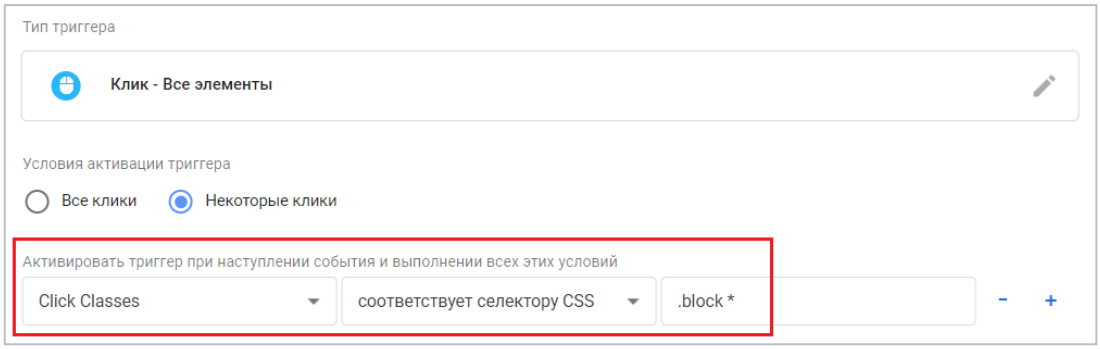


Рис. 172. Клик по элементам, соответствующим CSS-селектору класса .block и всех узлов-потомков с помощью универсального селектора *

Такая запись означает, что необходимо отслеживать клик по всем узлам-потомкам элемента с классом **block.**

В CSS есть такое понятие, как комбинаторы. Они позволяют объединить множество селекторов, что дает возможность выбирать элементы внутри других элементов, или смежные элементы.

Элемент

Если необходимо выбрать все определенные элементы на страницы, используйте конструкцию *элемент*.

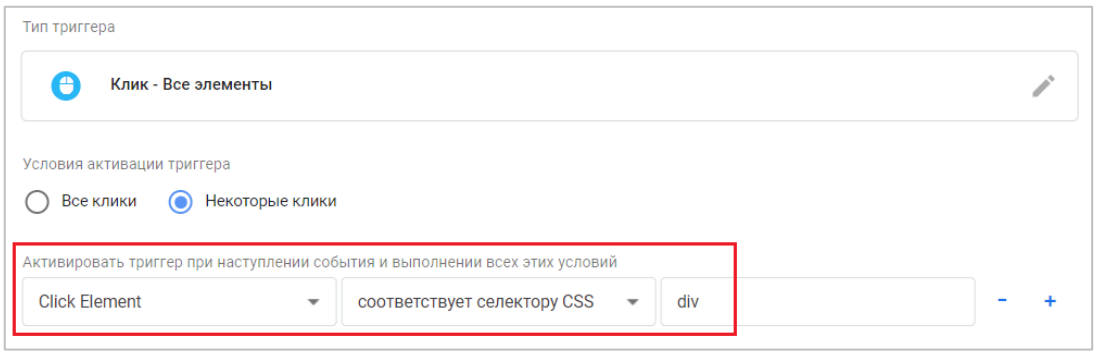


Рис. 173. Клик по элементам, соответствующим CSS-селектору по div

Триггер в GTM будет активироваться при выполнении события по всем элементам **“div”** на странице.

Элемент,Элемент

Если необходимо выбрать все элементы **“div”** и **“p”**, то используйте конструкцию *элемент, элемент*.

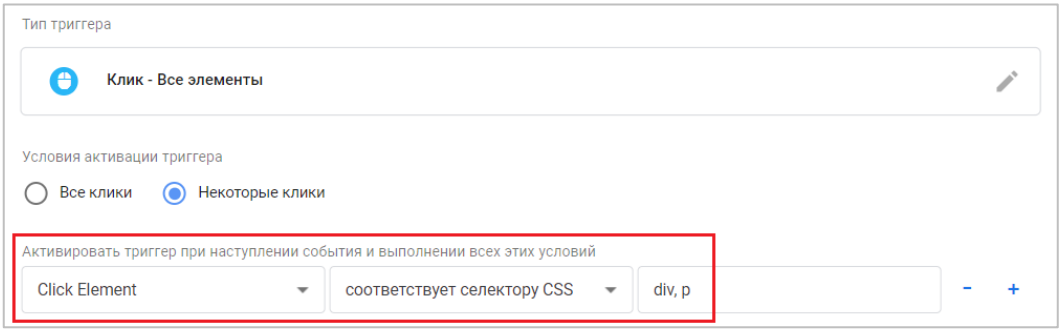


Рис. 174. Клик по элементам, соответствующим CSS-селектору по div, p

Триггер в GTM будет активироваться при выполнении события по всем элементам **“div”** и **“p”** на странице.

Элемент

Если необходимо выбрать элемент, вложенный в другой элемент, то используйте конструкцию **элемент**.

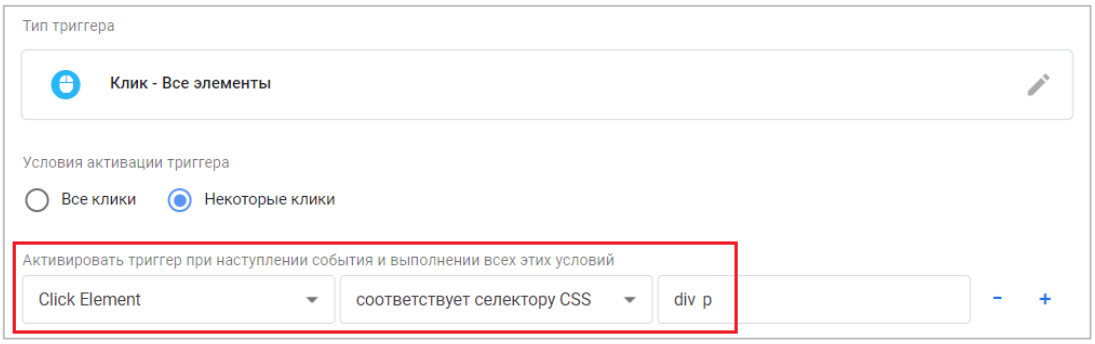


Рис. 175. Клик по элементам, соответствующим CSS-селектору div p

Триггер в GTM будет активироваться при выполнении события по всем элементам **“p”** на странице внутри элементов **”div”**. Не обязательно должен быть *родительский узел - узел потомка*, но необходимо, чтобы первый элемент был перед вторым.

Элемент>Элемент

Пример div>span. Конструкция **элемент>элемент** выбирает все дочерние элементы **“span”**, у которых родитель - элемент **”div”**.

Разберем это на конкретном примере:

```
HTML ▼
<div>
  <p>Мороз и солнце; день чудесный!
</p>
  <span>
    <span>
      <p>Еще ты дремлешь, друг прелестный -</p>
    </span>
  </span>
  <p>Мороз и солнце; день чудесный!</p>
  <span>
    <p>Еще ты дремлешь, друг прелестный -</p>
  </span>
  <p>Мороз и солнце; день чудесный!</p>
</div>
```

Рис. 176. Пример конструкции элемент>элемент

Добавляем в Google Tag Manager условие активации триггера:

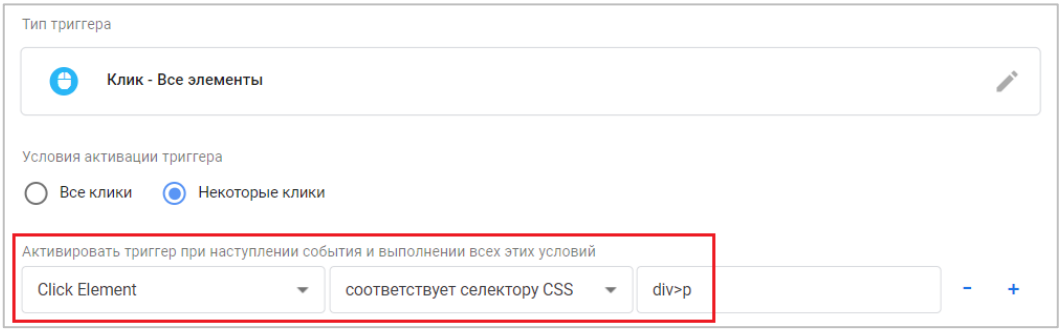


Рис. 177. Клик по элементам, соответствующим CSS-селектору div>p

Триггер в GTM будет активироваться при выполнении события по дочерним элементам **“p”** на странице внутри элемента **”div”**, то есть тех, которые на рисунке выше выделены зеленым цветом и соответствуют строке **Мороз и солнце; день чудесный!**.

Элемент+Элемент

Пример div+a. Конструкция *элемент+элемент* позволяет выбрать все элементы <a>, которые расположены сразу после закрывающегося тега .

```
<div class="first-buttons">
<a href="#popup-audit" class="button green">Консультация</a>
<a href="#price" class="button blue">Заказать рекламу</a>
</div>
<a class="policy-a fancybox" href="#policy-2">Пользовательское
соглашение</a>
```

Рис. 178. Конструкция элемент+элемент на примере div+a

Таким образом, если мы зададим условие активации триггера в GTM по Click Element и соответствию селектору CSS **“div+a”**, то тег будет активироваться по событию на странице по тегу <a>, который идет сразу после закрывающегося тега ”div” (выделено зеленым), игнорируя другие теги <a> внутри “div” (выделены красным).

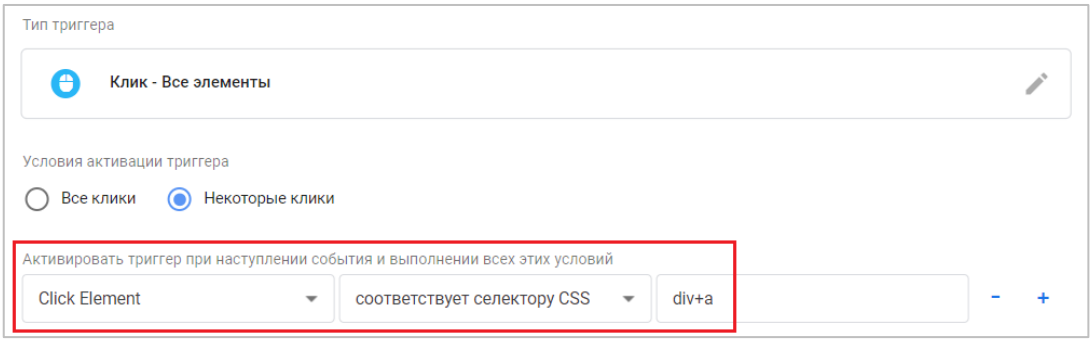


Рис. 179. Клик по элементам, соответствующим CSS-селектору div+a

Элемент~Элемент

Пример p~ol. Конструкция **элемент~элемент** позволяет выбрать все элементы **“ol”**, которым предшествует элемент **“p”**.

Селекторы по атрибутам

У элементов HTML есть атрибуты — это дополнительные значения, которые настраивают элементы или регулируют их поведение различным способом.

Существует особый вид селекторов, позволяющий искать элементы по их атрибутам и значениям. Для их записи используются квадратные скобки **[]**.

Селекторы наличия и значения атрибута

Эти селекторы выделяют элементы на основе точного значения атрибута:

- **[атрибут]** - выбирает все элементы с атрибутом **”атрибут”**, вне зависимости от его значения;

Пример [target] - выбирает все элементы на странице с атрибутом **target**.

Примечание: по умолчанию, при переходе по ссылке документ открывается в текущем окне или фрейме. При необходимости, это условие может быть изменено атрибутом target тега <a>.

- **[атрибут=”значение”]** - выбирает все элементы с атрибутом **“атрибут”**, которое имеет значение **“значение”**.

Пример [target=_blank] – выбирает все элементы с атрибутом **“target”**, который имеет значение **“_blank”**.

- **[атрибут~=”значение”]** – выбирает все элементы с атрибутом “атрибут”, в значении которого (в любом месте) встречается "значение" в виде отдельного слова. Если говорить простыми словами, то является аналогом условия *содержит*.

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Пример</title>
    <style>
      [alt~=flower] { border: 5px double red; }
    </style>
  </head>
  <body>
    <p>Картинка с атрибутом alt, содержащим слово "flower", будет иметь красную рамку.</p>
    <p>
      <img src="image1.jpg" alt="klematis flower" width="150" height="113">
      <img src="image2.gif" alt="flowers" width="224" height="162">
      <img src="image3.jpg" alt="landscape" width="160" height="120">
    </p>
  </body>
</html>
```

Рис. 180. Пример использования [атрибут~=”значение”]

Тильда **(~)** в данном селекторе является ключом для осуществления выбора элемента на основании наличия в значении атрибута нужного слова, отделенного пробелом. Если тильда будет пропущена, то получится требование к точному значению.

Селекторы подстроки в значении атрибута

Селекторы по атрибутам этого типа еще называют **Селекторы типа регулярных выражений (RegExp-like selectors),** поскольку они предлагают способ проверки условий, подобно тому, как это делают регулярные выражения.

Селекторы типа регулярных выражений:

- **[атрибут|=”значение”]** - выбирает все элементы с атрибутом “атрибут”, имеющие значение в точности равное **“значение”** или начинающееся с **“значение-“** (обратите внимание, что символ **|** - это не ошибка, он нужен для работы с языковой кодировкой.)

Значение может быть единственным словом в атрибуте, например, lang="ru" или с добавлением других слов через дефис, например, lang="ru-region".

На примере ниже для “lang|=ru” задается стиль (color:green). Результатом будет измененный цвет текста у атрибута “ru” Привет! и “ru-region” Здравствуйте. (выделено зеленым), поскольку **[атрибут|=”значение”]** подразумевает *точное* значение или же *начинающееся с*.

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Пример</title>
    <style>
      [lang|=ru] { color:green; }
    </style>
  </head>
  <body>
    <p lang="ru">Привет!</p>
    <p lang="en">Hi!</p>
    <p lang="en">Hello!</p>
    <p lang="ru-region">Здравствуйте.</p>
  </body>
</html>
```

Рис. 181. Пример использования [атрибут|=”значение”]

Далее идут три css-селектора, которые используют регулярные выражения и в их конструкции присутствуют специальные символы, такие же, как в Google Analytics при фильтрации или поиске. Это **^**, **$** и *.

- **[атрибут^=”значение”]** - выбирает каждый элемент с атрибутом **“атрибут”**, значение которого начинается с **“значение”**.

Пример a[href^="https"] – выбирает каждый элемент <a> с атрибутом **href**, значение которого начинается с **"https"**.

- **[атрибут$=”значение”]** - выбирает все элементы с атрибутом **“атрибут”**, значение которого заканчивается на **“значение”**.

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Пример</title>
    <style>
      div[class$="test"] {color: red;}
    </style>
  </head>
  <body>
    <div class="1_test">Первый элемент div.</div>
    <div class="2">Второй элемент div.</div>
    <div class="test">Третий элемент div.</div>
    <p class="test">Абзац с текстом.</p>
  </body>
</html>
```

Рис. 182. Пример использования [атрибут$=”значение”]

На примере выше для элемента div и класса с атрибутом, значение которого заканчивается на **“test”**, применяется стиль {color:red}; (задается красный цвет). Таким образом два элемента из четырех на странице будут отображены шрифтом красного цвета. Это div class =”1_test” (Первый элемент div.) и div class =”test” (Третий элемент div.). Последний тоже имеет class = “test” (Абзац с текстом), однако он входит в элемент <p>, а не div, и поэтому не будет выделен красным цветом.

Или еще один пример a[href$=".pdf"]. В этом случае вы можете отслеживать каждый элемент тега <a> с атрибутом **“href”**, значение которого оканчивается на **".pdf"**. В Google Tag Manager с помощью данного атрибута можно отслеживать скачивание файлов на сайте, клики по определенным картинкам формата **“.png”, “.jpg”, “.gif”** и т.д.

- **[атрибут*=”значение”]** - выбирает все элементы с атрибутом **“атрибут”**, в значении которого (в любом месте) встречается подстрока (в виде отдельного слова или его части) **"значение"**.

Проиллюстрируем это на следующем примере:

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Пример</title>
    <style>
      div[class*="test"] {color: blue;}
    </style>
  </head>
  <body>
    <div class="1_test">Первый элемент div.</div>
    <div class="testirovshik_2">Второй элемент div.</div>
    <div class="testing">Третий элемент div.</div>
    <p class="test">Абзац с текстом.</p>
  </body>
</html>
```

Рис. 183. Пример использования [атрибут*=”значение”]

В стиле для элемента div и класса с атрибутом, значение которого “test”, применяется стиль {color:blue}; (задается синий цвет). Таким образом будут выбраны первые три элемента вне зависимости от того, *точное* ли было вхождение условия **class*=”test”** или нет, *начиналось с* или *заканчивалось на*. Сам факт наличия значения “test” во всех трех элементах div селектора подстроки в значении атрибута позволяет нам выбрать все эти элементы. Последний вариант (Абзац с текстом) тоже имеет значение “test”, однако он не входит в элемент div, а является составляющей тега <p>, поэтому он не будет выделен синим цветом.

Псевдоклассы

Псевдоклассы — это дополнения к обычным селекторам, которые делают их еще точнее и мощнее. Они определяют динамическое состояние элементов, которое изменяется с помощью действий пользователя, а также положение в дереве документа. Примером такого состояния служит текстовая ссылка, которая меняет свой цвет при наведении на нее курсора мыши. Это реализуется с помощью псевдокласса **:hover**.

Псевдокласс добавляется к селектору с помощью символа двоеточия **: (селектор:псевдокласс).** В Google Tag Manager псевдоклассы также можно применять для активации переключателей элементов интерфейса, например, *checkbox* и *флажки (radio)*, или для отслеживания кликов по определенным элементам из выпадающего списка.

Псевдокласс – это селектор, который используется для выбора того, что не может быть выбрано с помощью других селекторов или может быть выбрано, но достаточно трудным способом. Их существует большое количество, но на практике в GTM используются некоторые. Вот одни из самых популярных:

- **:checked, :selected, :disabled, :enabled** — различные состояния элементов *input*;
- **:nth-child(n)** — позволяет отследить определенный элемент списка. Например, **“ul>li:nth-child(4)”** позволяет отследить четвертый элемент;
- **:nth-last-child(n)** — псевдокласс, противоположный предыдущему, который позволяет отследить определенный элемент списка, отчет элементов идет с конца. Например, селектор CSS **“:nth-last-child(2)”** вернет предпоследний элемент списка;
- **:not(селектор)** — псевдокласс отрицания. Выбирает все элементы, кроме того элемента, что в скобках.

Таблица псевдоклассов с примерами (см. приложение):

Псевдокласс	Пример	Описание
:link	a:link	Выбор всех не посещенных ссылок
:visited	a:visited	Выбор всех посещенных ссылок
:active	a:active	Выбор активной ссылки
:hover	a:hover	Выбор ссылки при наведении курсора мышки
:focus	input:focus	Выбор элемента <input>, который находится в фокусе
:first-child	p:first-child	Выбор каждого элемента <p>, который является первым дочерним элементом своего родителя
:lang(язык)	p:lang(ru)	Выбор каждого элемента <p> с атрибутом lang, значение которого начинается с "ru"
:first-of-type	p:first-of-type	Выбор каждого элемента <p>, который является первым из элементов <p> своего родительского элемента
:last-of-type	p:last-of-type	Выбор каждого элемента <p>, который является последним из элементов <p> своего родительского элемента
:only-of-type	p:only-of-type	Выбор каждого элемента <p>, который является единственным элементом <p> своего родительского элемента
:only-child	p:only-child	Выбор каждого элемента <p>, который является единственным дочерним элементом своего родительского элемента
:nth-child(n)	p:nth-child(2)	Выбор каждого элемента <p>, который является вторым дочерним элементом своего родительского элемента
:nth-last-child(n)	p:nth-last-child(2)	Выбор каждого элемента <p>, который является вторым дочерним элементом своего родительского элемента, считая от последнего дочернего элемента
:nth-of-type(n)	p:nth-of-type(2)	Выбор каждого элемента <p>, который является вторым дочерним элементом <p> своего родительского элемента
:nth-last-of-type(n)	p:nth-last-of-type(2)	Выбор каждого элемента <p>, который является вторым дочерним элементом <p> своего родительского элемента, считая от последнего дочернего элемента

Псевдокласс	Пример	Описание
:last-child	p:last-child	Выбор каждого элемента <p>, который является последним элементом своего родительского элемента
:root	:root	Выбор корневого элемента в документе
:empty	p:empty	Выбор каждого элемента <p>, который не содержит дочерних элементов (включая текст)
:target	:target	Выбор текущего целевого элемента на странице, то есть элемента, к которому был осуществлён переход по ссылке внутри страницы
:enabled	input:enabled	Выбор каждого включенного элемента <input>
:disabled	input:disabled	Выбор каждого выключенного элемента <input>
:checked	input:checked	Выбор элемента <input>, выбранного по умолчанию или пользователем
:not(селектор)	:not(p)	Выбор всех элементов, кроме элемента <p>

Псевдоэлементы

Помимо псевдоклассов, существуют еще и псевдоэлементы. **Псевдоэлемент** — это виртуальный элемент, который не существует в явном виде в дереве элементов документа. Они используются для выбора тех частей элемента, которые не могут быть выбраны с помощью других селекторов, а также для стилизации не всего элемента, а отдельных его частей.

Таблица псевдоэлементов с примерами:

Псевдоэлемент	Пример	Описание
::first-letter	p::first-letter	Выбор первой буквы каждого элемента <p>
::first-line	p::first-line	Выбор первой строки каждого элемента <p>
::before	p::before	Добавляет элемент с содержимым перед содержимым каждого элемента <p>
::after	p::after	Добавляет элемент с содержимым после содержимого каждого элемента <p>

Рекомендую перейти на сайт **w3schools.com** (см. приложение) и изучить два материала для лучшего понимания данной темы. Первая ссылка содержит все CSS-селекторы и псевдоклассы с примерами и описанием их работы, а вторая визуально демонстрирует, как работают различные селекторы.

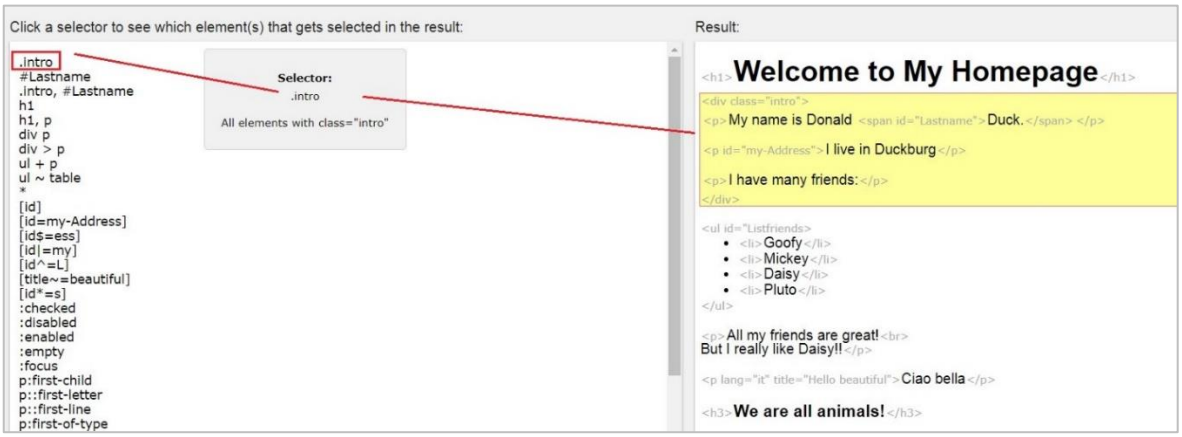


Рис. 184. Демонстрация селекторов на сайте w3schools.com

Выбрав в левой части экрана один из селекторов, справа вы увидите подсвеченный результат. Также для закрепления пройденного материала рекомендую прочитать статью по основам селекторов на примере котиков (см. приложение) и несколько разделов из справки разработчиков Mozilla (см. приложение).

Для того, чтобы правильно обращаться к элементам на странице и применять их в Google Tag Manager для отслеживания определенных событий, нужно просто глубже разобраться в теме CSS-селекторов, понять их взаимосвязь друг с другом и использовать инструменты, которые упрощают их обнаружение.

Разберем три варианта определения CSS-селекторов.

Консоль разработчика

Традиционный способ, с помощью консоли разработчика любого браузера. Я пользуюсь Google Chrome, поэтому разберем пример на нем. Выбрав отслеживаемый элемент на странице, нажмите на него правой кнопкой мыши и **Просмотреть код**.

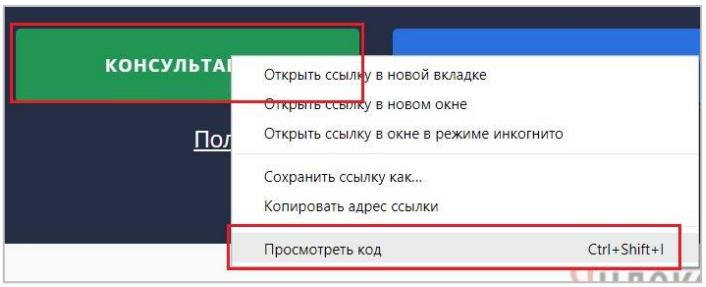


Рис. 185. Правой кнопкой мыши – Просмотреть код

После того, как мы удостоверились, что выбрали нужный элемент, в открывшемся коде страницы еще раз нажмите правой кнопкой мыши по нужному элементу. Далее **Copy - Copy selector**.

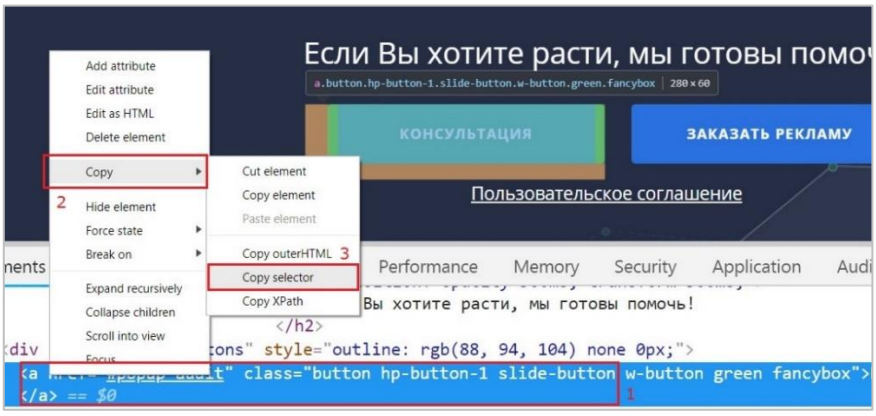


Рис. 186. Правой кнопкой мыши – Copy – Copy selector

В буфер обмена у вас скопировался селектор. Можно вставить его в любой текстовый документ и посмотреть, как он будет выглядеть. В моем примере он выглядит так:

#home > div > div > div > div > div > a.button.hp-button-1.slide-button.w-button.green.fancybox

Получилась очень длинная цепочка взаимосвязей элементов друг с другом. Несмотря на это его можно использовать в Google Tag Manager для настройки отслеживания события. Однако тот же селектор можно записать и в таком виде: *a.button.hp-button-1.slide-button.w-button.green.fancybox*, просто убрав общую часть в начале.

Селекторы в jQuery

Как мы узнали раннее, jQuery – это библиотека JavaScript, которая фокусируется на взаимодействии JavaScript, HTML, CSS и служит для облегчения работы разработчика. На данный момент она является самой распространенной библиотекой JS в мире.

В природе существуют и другие JS-библиотеки, фреймворки и инструменты, например, **React, AngularJS, Backbone.js, Ember.js** и т.д. Но в свое время именно jQuery произвела революцию в программировании клиентской части веб-приложений, введя селекторы CSS для доступа к узлам DOM-дерева, обработчики событий, анимации и AJAX-запросы.

JavaScript является фундаментальным языком и для того, чтобы создать всплывающее окно или выпадающий список, разработчику потребуется написать большое количество строк кода. В jQuery это уже все реализовано, достаточно всего лишь использовать готовую функцию.

Библиотека jQuery на сайте по умолчанию отсутствует. Для ее подключения необходимо скачать актуальную версию с официального сайта **jquery.com/download**, загрузить ее на сервер, а затем подключить с помощью фрагмента кода:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <!--Подключаем библиотеку-->
  <script src="js/jquery-3.3.1.min.js"></script>
</head>
<body>
</body>
</html>
```

Рис. 187. Подключение библиотеки jQuery через путь к файлу (версия 3.3.1)

Существует альтернативный способ подключить jQuery на страницы вашего сайта (не закачивая библиотеку на сервер). Можно подключить библиотеку, которая находится не на вашем сервере, а на серверах CDN. Существуют несколько таких хранилищ, наиболее известные и надежные из них Google CDN, Microsoft CDN, а также CDN, который организовали создатели jQuery. Код выглядит так:

```
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
```

Можно подключить jQuery и через Менеджер тегов Google. Для этого необходимо создать пользовательский HTML-тег и добавить туда фрагмент кода выше, а в качестве триггера активации выбрать **All Pages (Все страницы)**.

Чтобы проверить, установлен ли на сайте jQuery, введите код в консоли разработчика (вкладка Console) и нажмите **Enter**:

```
var msg;
if (window.jQuery) {
    msg = 'Версия jQuery: ' + jQuery.fn.jquery;
} else {
    msg = 'jQuery не установлен';
}
alert(msg);
```

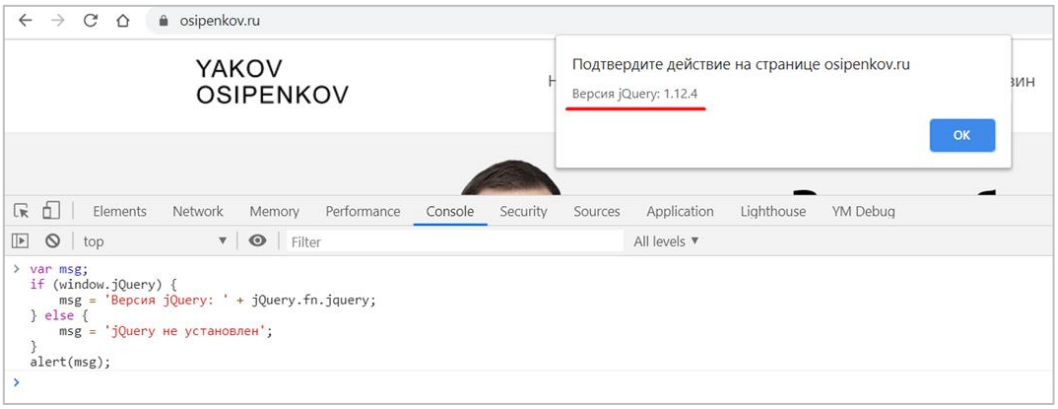


Рис. 188. Проверка установки jQuery на сайте

Селекторы в jQuery базируются на CSS селекторах (см. выше), они также используются для определения элемента или группы элементов.

Все селекторы в jQuery начинаются со знака доллара и круглых скобок **$ ()**, например, $('div'). В этом случае будет осуществлен поиск всех элементов div на странице, $('.className') найдет все элементы с классом className, $('#sidebar') найдет все элементы на странице с идентификатором sidebar и т.д.

Примечание: при использовании метасимволов (**#;&,.+*~':"!^$[]()=>|/**) в значении любого идентификатора, класса или названия атрибута, необходимо экранировать эти символы в селекторах с помощью двух обратных слэшей ****

```
$('input[value=3.5]') /* неправильно */
$('input[value=3\\.5]') /* правильно */
```

В процессе работы с Google Tag Manager jQuery позволяет удобнее решать множество задач. Задачи можно разделить на два типа:

1. взять элемент, который у нас находится на сайте, например, значение цены, название продукта, категорию продукта и т.д.;
2. отследить факт того, что у нас произошел клик по какому-либо элементу, зафиксировано событие или пользователь заполнил и отправил форму на сайте.

При настройке некоторых событий в GTM иногда возникают ситуации, когда необходимо получить дополнительную информацию, связанную с элементом, над которым произошло событие. Но эту информацию так просто нельзя получить через CSS-селекторы, поскольку она может находится в структуре других элементов, которые логически связаны с рассматриваемым элементом. В этом случае можно воспользоваться функциями библиотеки jQuery.

Давайте разберем на нескольких примерах работу библиотеки jQuery с набором элементов. Для этого я перейду на свой сайт osipenkov.ru и вызову консоль разработчика, а затем перейду на вкладку **Console**.

С помощью **$("div").css("border", "1px solid red");** выберем все элементы div на странице и обведем их в красную рамку в 1 пиксель.

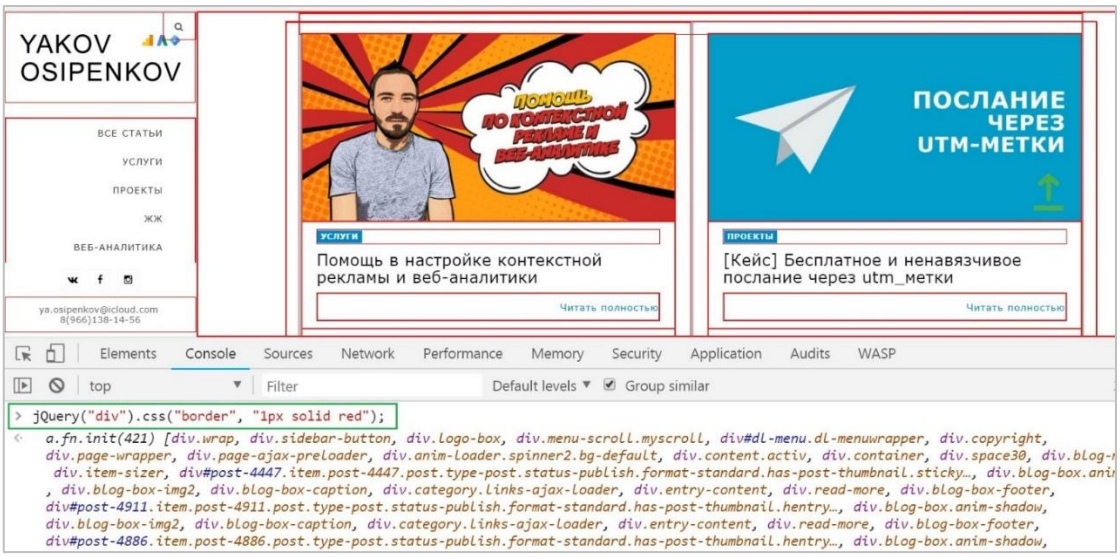


Рис. 189. Все div на странице обведены в красную рамку шириной в 1 пиксель

Примечание: мой блог работает на WordPress. В нем стандартный способ обращения к элементам через **$()** отключен (в конце файла .../wp-includes/js/jquery/jquery.js прописана строчка **jQuery.noConflict();**, которая отключает работу с элементами страницы через $.), так как другие библиотеки также могут использовать данный механизм обращения к элементам. Поэтому вместо знака доллара я использовал альтернативную конструкцию **jQuery ()**. Для изменения способа обращения на привычный $ необходимо изменить фрагмент кода. Решение проблемы ищите в интернете с пометкой *jQuery не работает в WordPress*.

Теперь давайте поработаем с формами. Перейдем на страницу **Контакты** https://osipenkov.ru/contacts/ где есть форма обратной связи.

Добавим в консоль такую строчку:

```
$("form input").css("border", "1px solid blue");
```

Данная конструкция добавит рамку всем input-ам, которые являются потомками элемента form. Это пример так называемого *parent child*, когда выбираются все элементы input, являющиеся узлами-потомками для родительского элемента form.

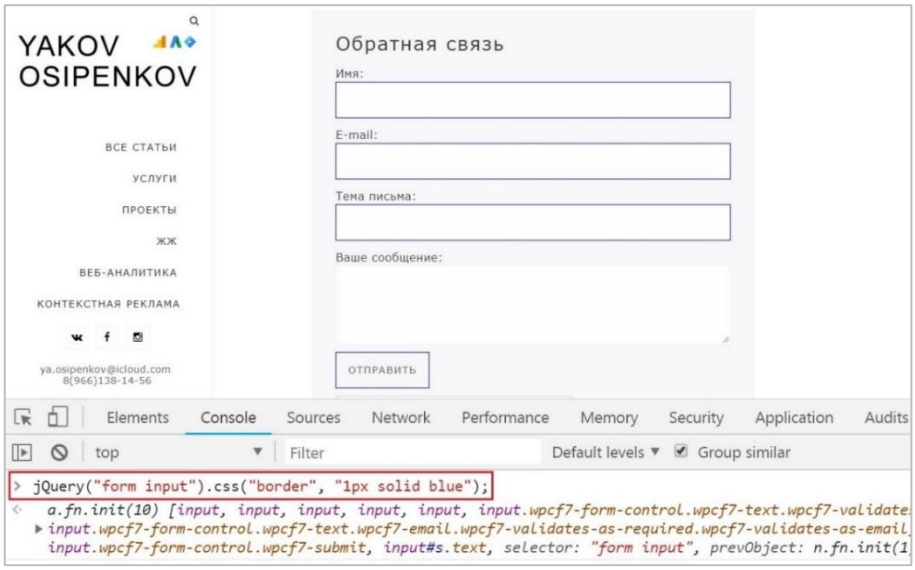


Рис. 190. Добавление синей рамки в 1 пиксель для всех input

В качестве еще одного простого примера разберем вывод значения заполненного поля Имя с **name=”your-name”**.

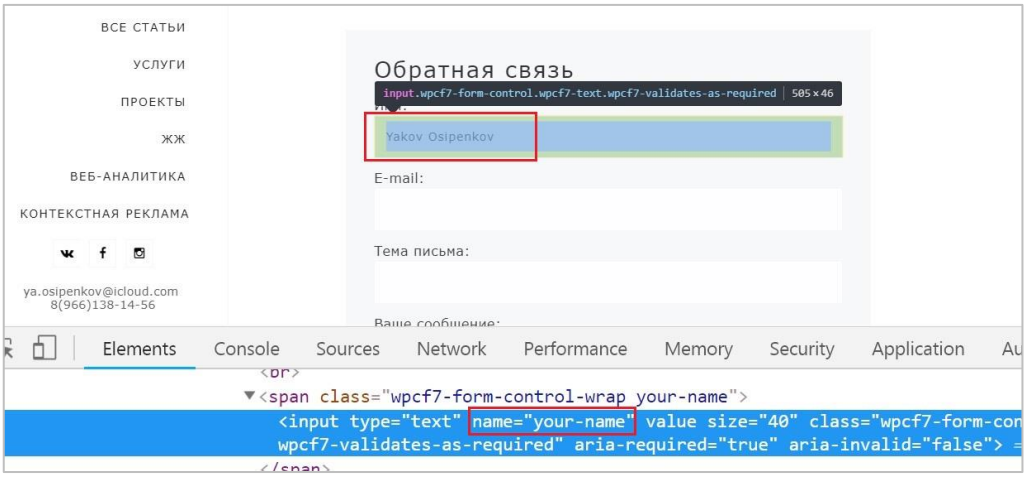


Рис. 191. Поле Имя с name=”your-name”

В консоли разработчика введите такую конструкцию:

```
$('[name="your-name"]').val();
```

Метод **.val()** позволяет получать и изменять значения элементов форм. Для элементов input это значение атрибута **“name”**, то есть в нашем случае “your-name”. Получим результат **Yakov Osipenkov**.

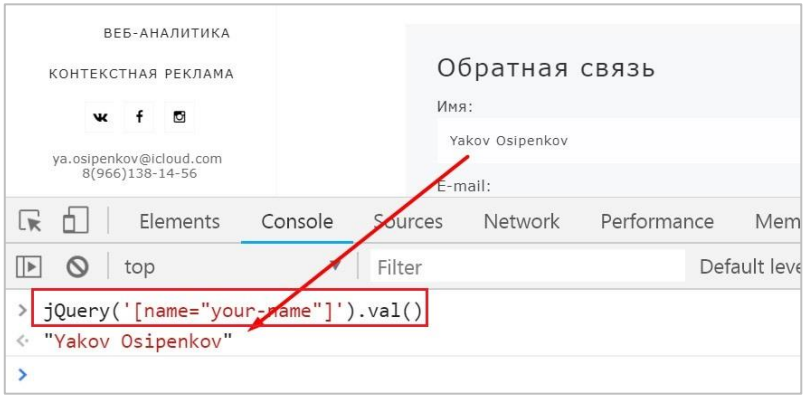


Рис. 192. Результат $('[name="your-name"]').val()

Разобранные выше примеры являются самыми простыми в освоении. При работе с Google Tag Manager вы будете использовать гораздо более сложные конструкции с различной цепочкой методов и другим набором элементов.

Таблица некоторых функций jQuery перемещения по дереву DOM (см. приложение):

Функция	Описание
.children()	Находит все дочерние элементы у выбранных элементов. При необходимости, можно указать селектор для фильтрации
.closest()	Находит ближайший, соответствующий заданному селектору элемент, из числа следующих: сам выбранный элемент, его родитель, его прародитель, и так далее, до начало дерева DOM
.find()	Находит элементы по заданному селектору, внутри выбранных элементов
.next()	Находит элементы, которые лежат непосредственно после каждого из выбранных элементов
.nextAll()	Находит элементы, которые лежат после каждого из выбранных элементов
.nextUntil()	Находит элементы, которые лежат после каждого из выбранных, но не дальше элемента, удовлетворяющего заданному селектору
.offsetParent()	Возвращает ближайшего предка с позиционированием, отличным от static (позиционирование по умолчанию)
.parent()	Находит родительские элементы у всех выбранных элементов
.parents()	Находит всех предков у выбранных элементов, т.е. не только прямых родителей, но и прародителей, прапрародителей и так далее, до начала дерева DOM
.parentsUntil()	Находит предков, как и .parents(), но прекращает поиск перед элементом, удовлетворяющим заданному селектору
.prev()	Находит элементы, которые лежат непосредственно перед каждым из выбранных элементов
.prevAll()	Находит элементы, которые лежат перед каждым из выбранных элементов
.prevUntil()	Находит элементы, которые лежат перед каждым из выбранных, но не дальше элемента, соответствующего заданному селектору
.siblings()	Находит все соседние элементы (под соседними понимаются элементы с общим родителем)

Функции фильтрации набора элементов:

Функция	Описание
.eq()	Возвращает элемент, идущий под заданным номером в наборе выбранных элементов
.filter()	Фильтрует набор выбранных элементов с помощью заданного селектора или функции
.first()	Возвращает первый элемент в наборе
.has()	Фильтрует набор выбранных элементов, оставляя те, которые имеют потомков, соответствующих селектору
.is()	Проверяет, содержится ли в наборе, хотя бы один элемент, удовлетворяющий заданному селектору
.last()	Возвращает последний элемент в наборе
.not()	Возвращает элементы, не соответствующие заданным условиям
.slice()	Возвращает элементы с индексами из определенной области (например, от 0 до 5)

Также, как и с CSS-селекторами, рекомендую перейти на сайт **w3schools.com** и изучить два материала для лучшего понимания темы селекторов jQuery (см. приложение).

На сайте **jquery-docs.ru** (см. приложение) есть перевод официальной документации API jQuery на русский язык.

После того, как вы познакомитесь с CSS-селекторами и уделите им достаточное количество времени на изучение, вы сможете выбрать любой элемент для отслеживания в Google Tag Manager на вашем сайте, даже если у него нет никаких отличительных знаков, а также по достоинству оцените всю мощь и функциональность диспетчера тегов Google.

Регулярные выражения в CSS-селекторах

В CSS-селекторах можно использовать регулярные выражения (см. приложение). Если быть точнее, то с помощью специальной конструкции (спецсимволов) вы можете задать определенное правило, которое будет распространяться на атрибут не одного элемента, а сразу на целую группу, которая попадает под заданное условие.

Давайте разберем практическую задачу. На сайте по производству мебели необходимо отслеживать клик по адресу магазина и передавать эту информацию в инструменты веб-аналитики.

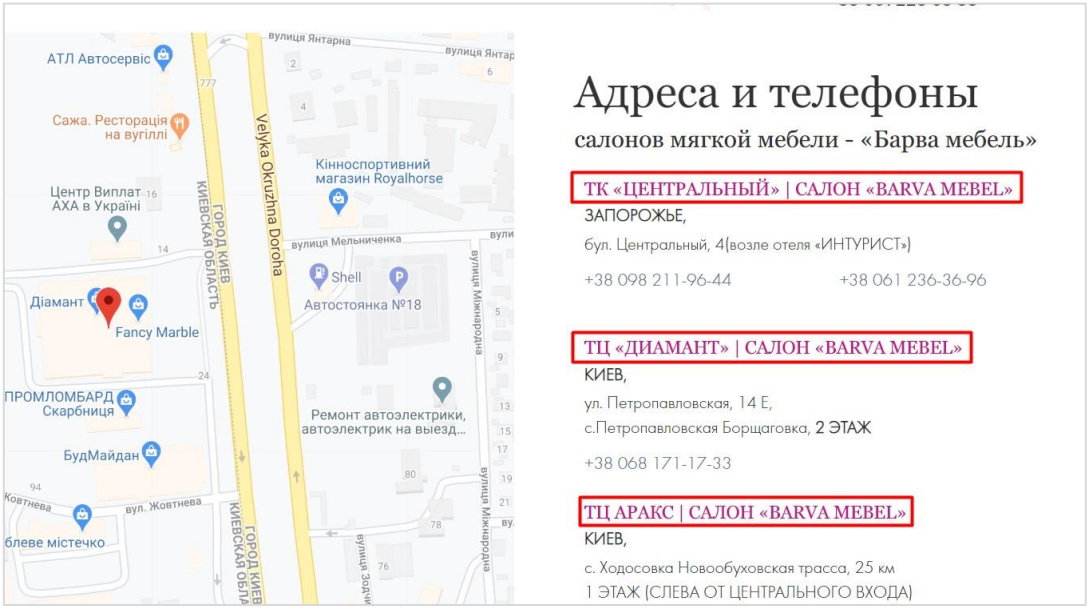


Рис. 193. Клик по адресу - Отправка события в счетчик веб-аналитики

Например, в Google Analytics, чтобы получить данные в таком виде:

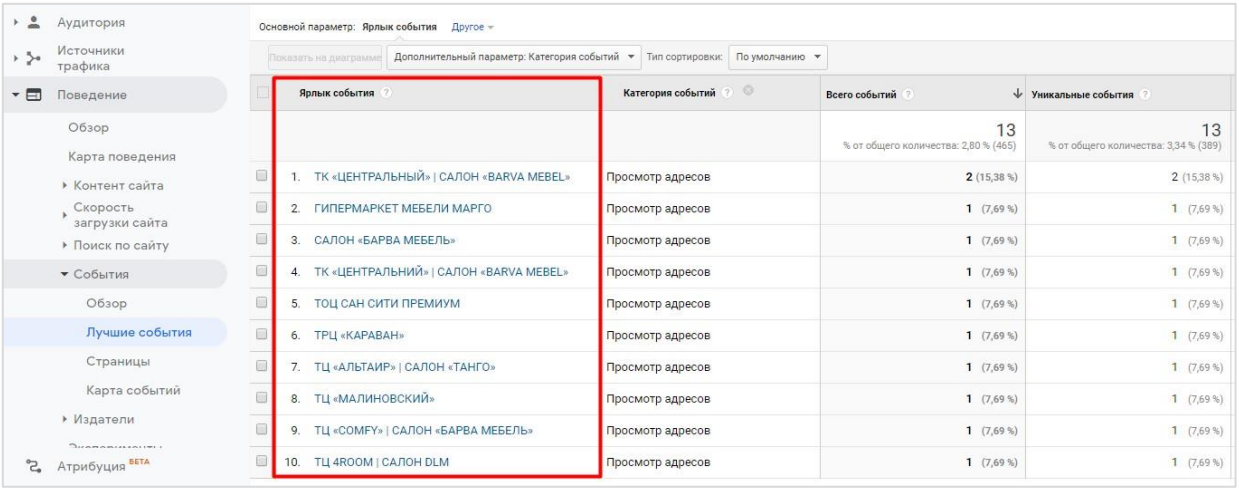


Рис. 194. События в Google Analytics

Чтобы мы делали в обычной ситуации? Определяли бы для каждого элемента свой CSS-селектор и настраивали бы триггер с конкретным условием, а потом еще и собственный тег для передачи данных в Яндекс.Метрику и Google Analytics. В результате у нас бы получилось много триггеров и много тегов. Нам это не подходит.

Если мы исследуем каждый из этих элементов, то найдем закономерность - все они имеют одинаковое начало атрибута id вида store-span-...

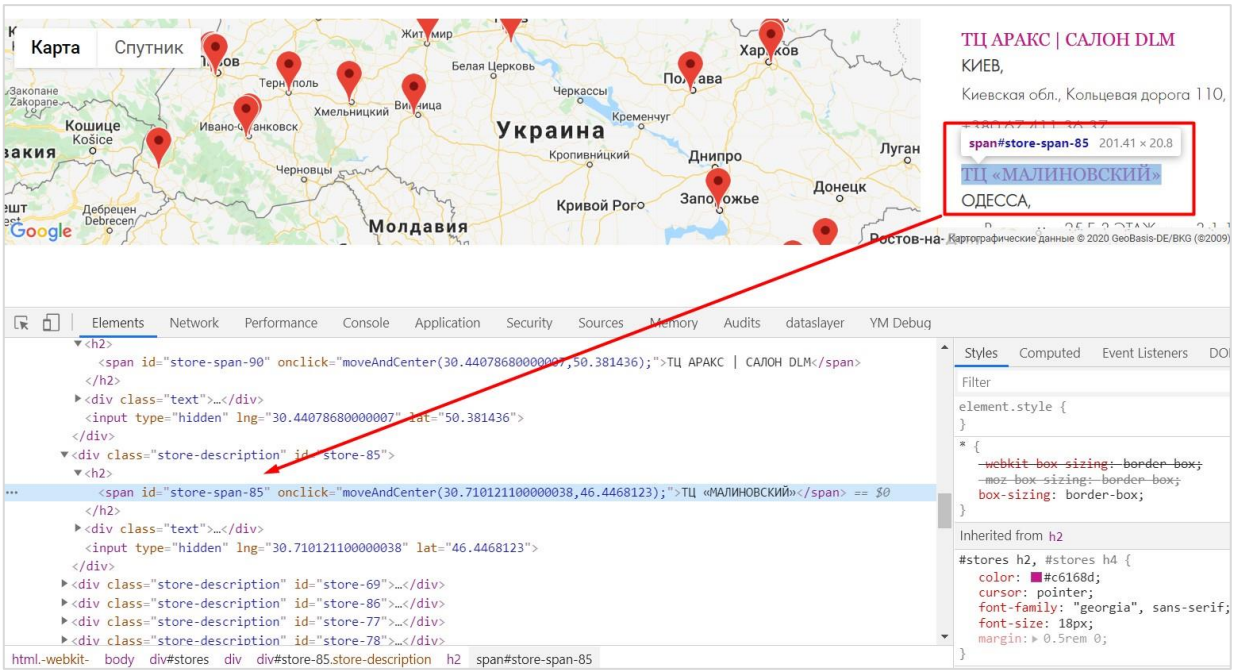


Рис. 195. У всех элементов одинаковое начало store-span-

Далее отличается только цифра: 69, 86, 77, 78, 85 и т.д. Чтобы отследить клики по всем таким адресам, нам в GTM потребуется прописать всего лишь 1 условие для триггера, используя регулярные выражения, и создать 1 тег.

У маркетологов часто встречается схожая задача - отследить клик по кнопке Купить или Добавить в корзину. Товаров может быть много, у каждого из них своя кнопка, в которой отличие только в ее идентификаторе, классе, атрибуте:

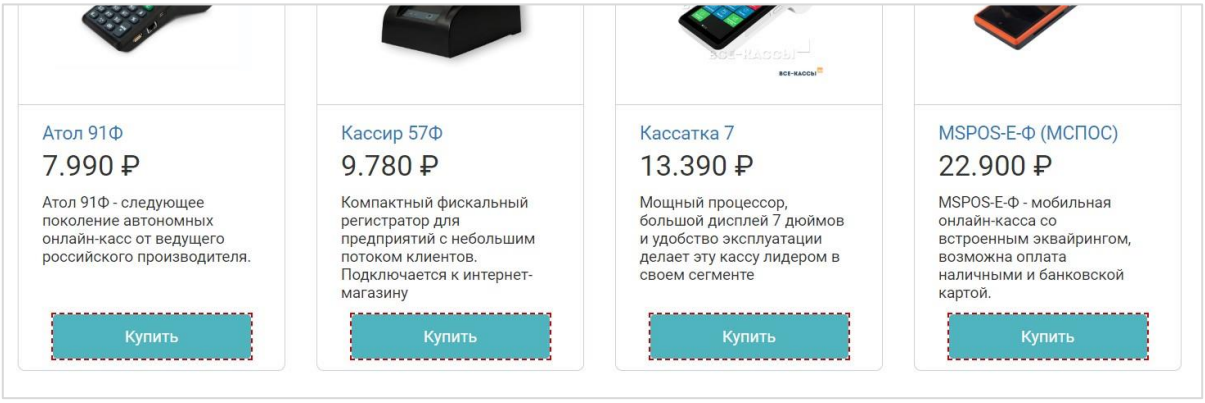


Рис. 196. Кнопка "Купить" с одинаковым атрибутом

Так или иначе, все это можно отслеживать благодаря Google Tag Manager, а прописывать условия с помощью CSS-селекторов и регулярных выражений.

В предыдущей статье про CSS-селектор я рассматривал принцип работы селекторов атрибутов. Они отбирают элементы по его наличию или значению. Посмотреть весь список можно в справочной документации Mozilla. Вот в них как раз и можно использовать регулярные выражения: ^=, $=, *=, ~=, |=

Давайте разберем на примерах.

[attribute^=value] Начинается с ^=

Атрибут начинается с value. Сам символ ^ используется для обозначения начала строки. Возьмем вышеописанный пример, где необходимо отследить клики по адресам магазинов и передать эту информацию в Google Analytics. Атрибут id начинается одинаково с store-span, поэтому мы можем написать такую конструкцию:

```
[id^=store-span]
```

С помощью CSS Selector Tester мы можем проверить какие элементы будут подсвечены на странице с данным атрибутом.

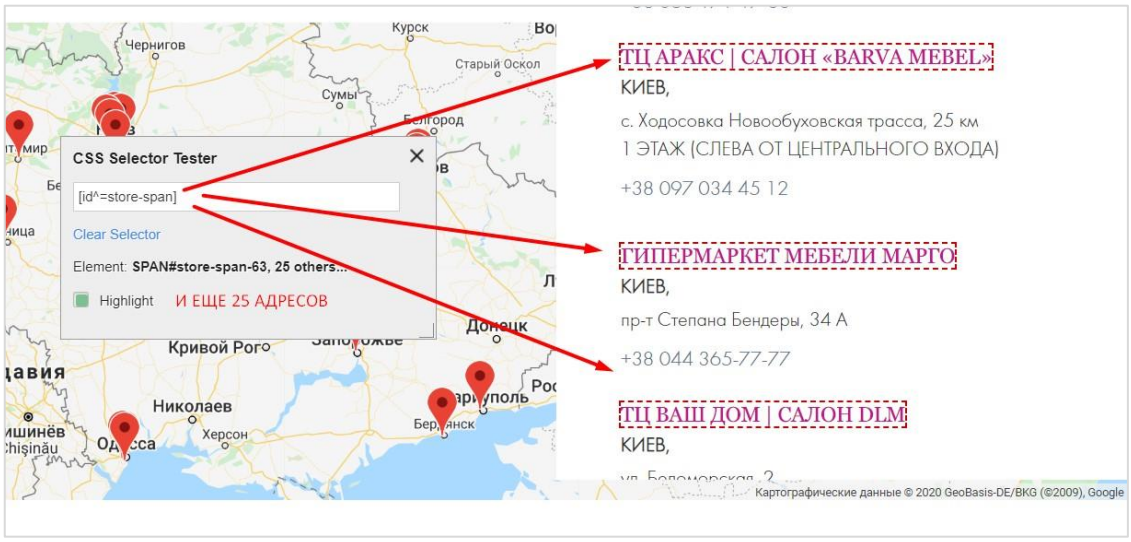


Рис. 197. [id^=store-span]

Для настройки триггера в Google Tag Manager можно воспользоваться типом Клик - Все элементы. В качестве условия активации триггера задать **Некоторые клики - Click Element - соответствует селектору CSS - [id^=store-span]**

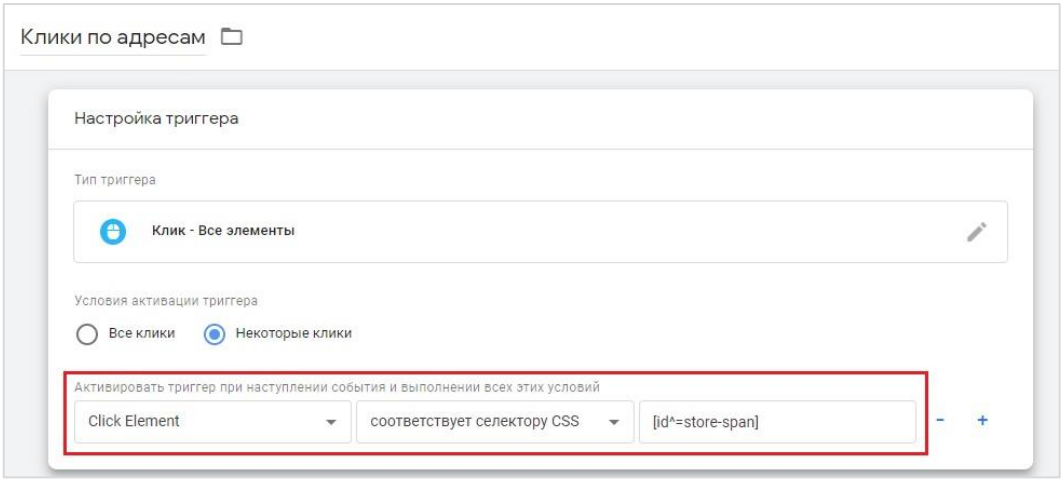


Рис. 198. Click Element - соответствует селектору CSS

Ну а далее, чтобы передать значение самого названия магазина, можно воспользовать встроенной переменной Click Text, которую нужно добавить в тег Universal Analytics с типом Событие в любое из 4 компонентов (Категория, Действие, Ярлык). Триггер активации - наш настроенный выше триггер **Клик - Все элементы**.

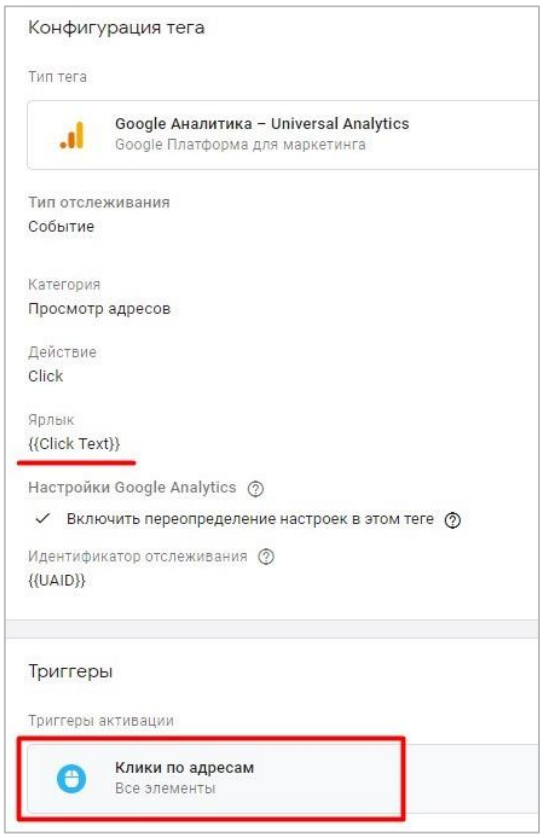


Рис. 199. Настройка тега Universal Analytics

[attribute$=value] Заканчивается на $=

Противоположный вариант, когда атрибут заканчивается на value. Символ $ используется для обозначения окончания строки. Например, активировать триггер при нажатии на ссылки, которые заканчиваются на .jpg. Конструкция такая:

```
img[src$=jpg]
```

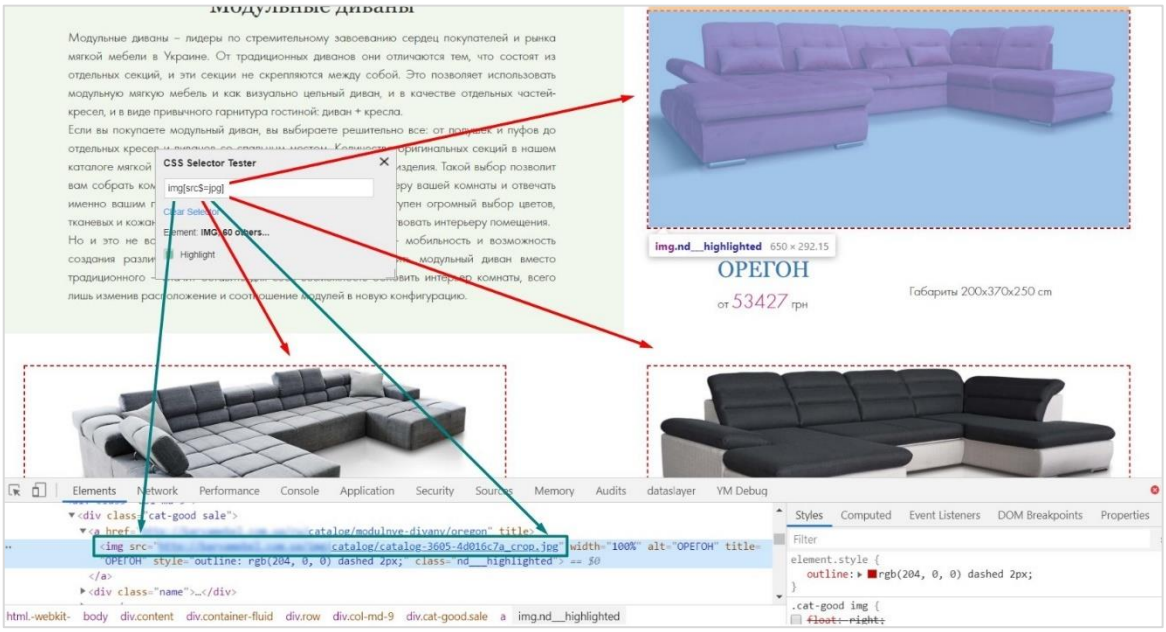


Рис. 200. img[src$=jpg]

В качестве условия активации триггера в Google Tag Manager можно задать то же самое: **Некоторые клики - Click Element - соответствует селектору CSS - img[src$=jpg]**. Либо же воспользоваться другим триггером **Клик - Только ссылки**.

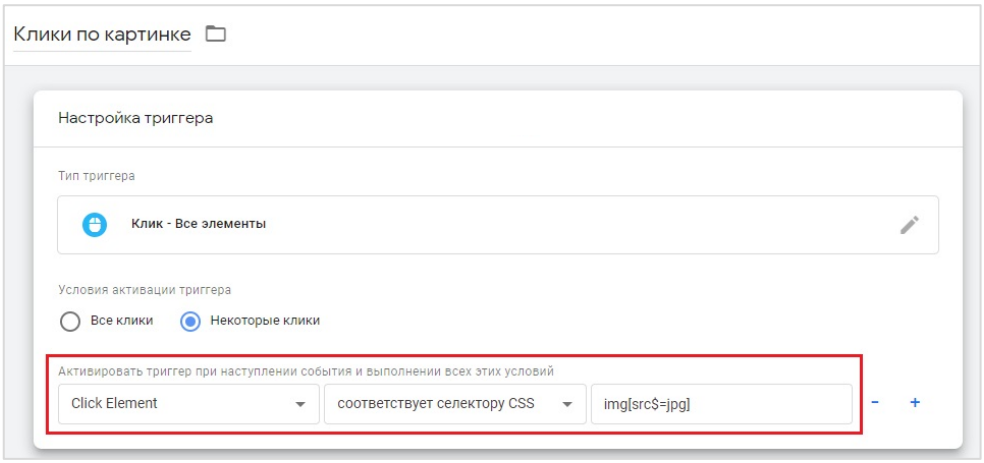


Рис. 201. Click Element - соответствует селектору CSS

[attribute*=value] Подстановочный знак *=

Атрибут содержит подстроку value и может быть равен myvalue, value2020, myvalue2020 и т.д. То есть выбираются все элементы с attribute, в значении которого (в любом месте) встречается подстрока (в виде отдельного слова или его части) value.

Таким образом, мы можем указать часть значения (без начала или конца) и при выполнении условия триггер в GTM также будет срабатывать. Приведу пример:

```
[title*=зыв]
```

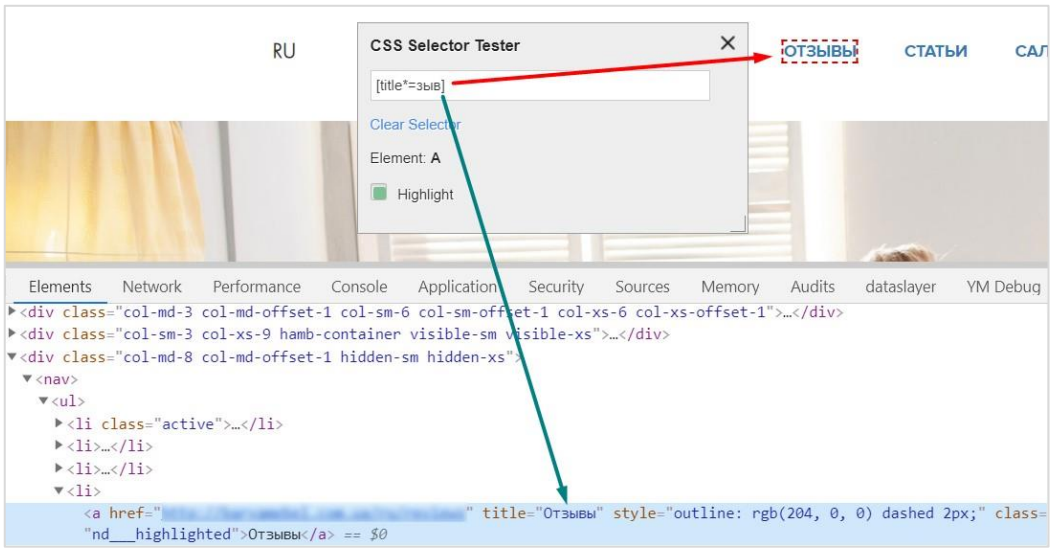


Рис. 202. [title*=зыв]

В этом случае произойдет активация триггера только в том случае, если пользователь нажмет на элемент, который содержит title со значением зыв. Я не дописал слово целиком целенаправленно, чтобы продемонстрировать, что подстановочный знак * все равно выделяет нужный элемент. А наше значение (value) находится ни в начале, ни в конце, а посередине.

В качестве условия активации триггера в Google Tag Manager можно задать аналогичную конструкцию: **Некоторые клики - Click Element - соответствует селектору CSS - [title*=зыв]**. Либо же воспользоваться другим триггером Клик - Только ссылки.

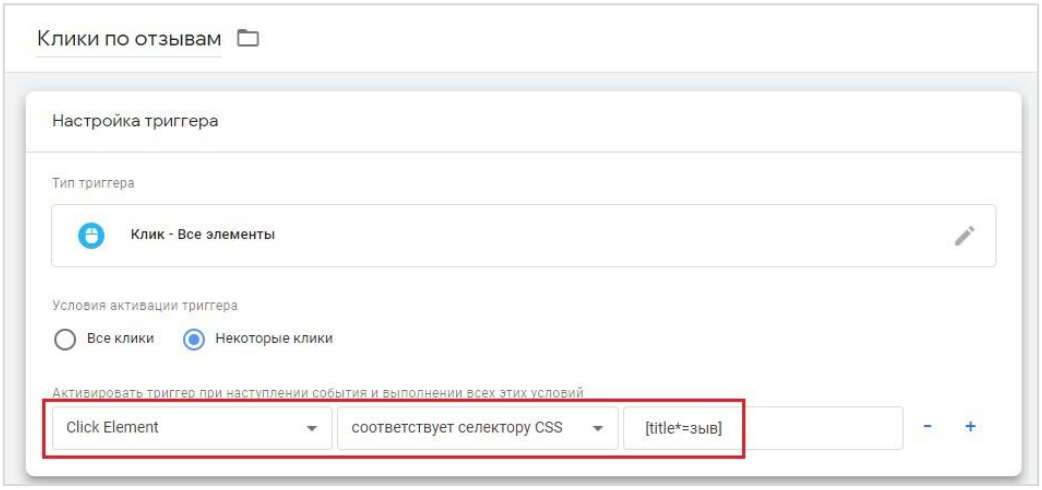


Рис. 203. Click Element - соответствует селектору CSS

Существует еще два селектора атрибутов со спецсимволами: [attribute|=value] и [attribute~=value].

[attribute|=value]

Атрибут равен value или начинается с value- (дефис следом!). Эта же конструкция подошла бы и для отслеживания кликов по адресам из предыдущего примера. Задав:

```
[id|=store-span]
```

Мы бы получили тот же самый результат:

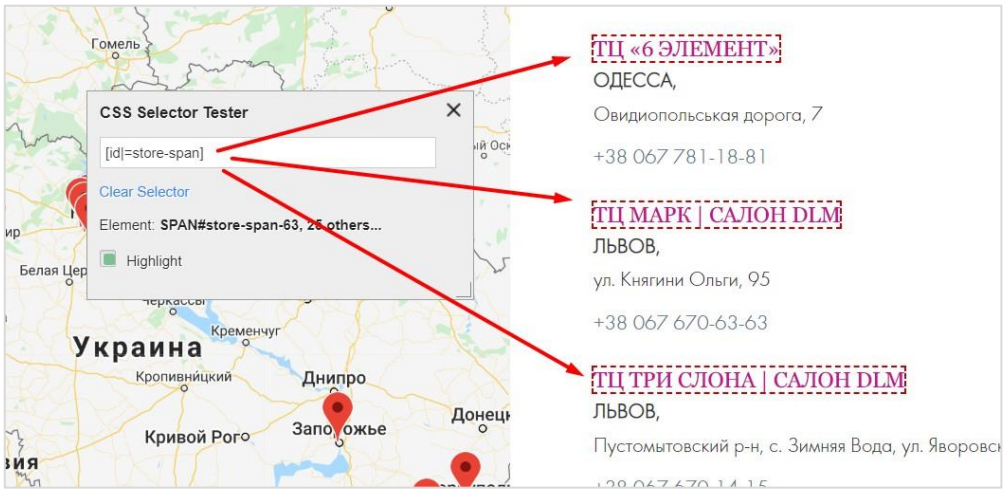


Рис. 204. [id|=store-span]

[attribute~=value]

Атрибут содержит содержит value, значением которого является набор слов, разделенных пробелами, одно из которых в точности равно value. То есть value должно идти как отдельное слово через пробел, иначе конструкция не будет работать. Например: [attribute~=value] верно для 2020 value и неверно для 2020value или 2020-value.

В качестве примера разберем набор товаров на сайте, у которых есть класс **cat-good**. У части мебели есть дополнительный класс **sale**, отделенный пробелом, а у некоторых моделей такого класса через пробел нет.

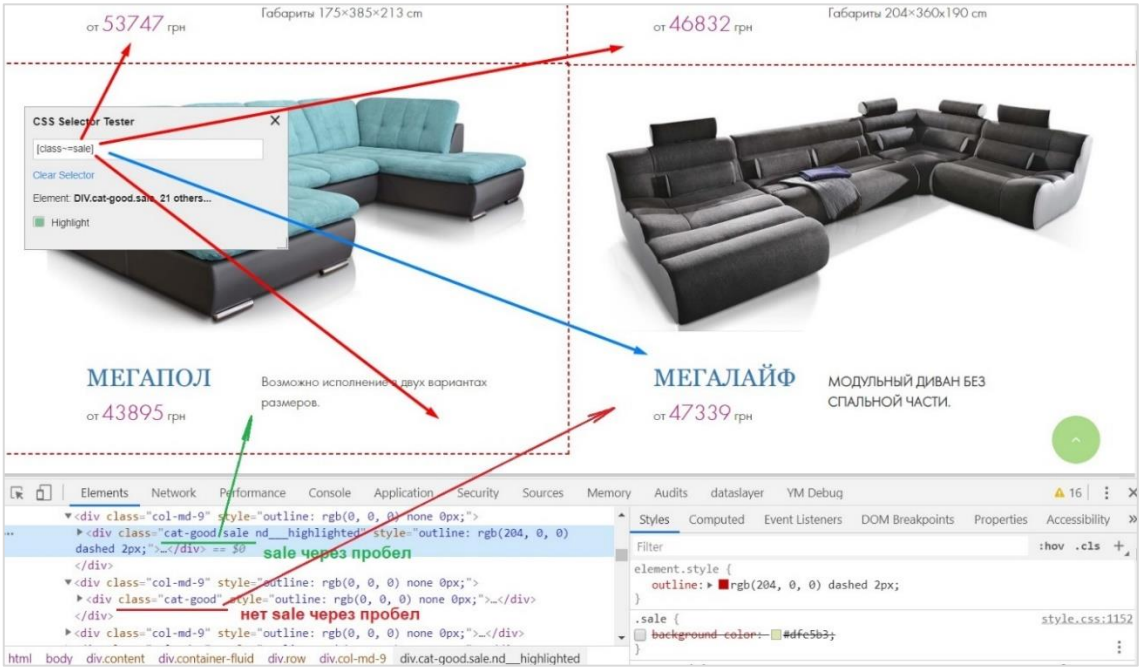


Рис. 205. sale через пробел есть и нет

Используя конструкцию:

```
[class~=sale]
```

CSS Selector Tester выделит нам странице только те товары, которые содержат в себе класс со значением sale, разделенный пробелом. Для последних двух селекторов по атрибутам условие активации триггера в GTM такое же - **Некоторые клики - Click Element - соответствует селектору CSS - CSS-селектор**.

Таким образом, конструкции CSS-селекторов с регулярными выражениями помогут вам создать наиболее гибкие условия для активации триггеров и пользовательских переменных в Google Tag Manager.

Отслеживание вложенных элементов с помощью универсального селектора (*)

Вспомните эпизоды из своей практики, когда вам необходимо было настроить отслеживание какого-либо элемента на странице. Вроде как все сделали правильно: нашли нужный элемент, скопировали его CSS-селектор, создали триггер, добавили условие активации, настроили тег. Но в режиме отладки вы все равно не видели срабатываемых событий и данные в инструменты аналитики не поступали? Почему так происходило? Возможно, дело во вложенности элементов.

Давайте разберем конкретный пример. На сайте graphanalytics.ru есть шапка (header), в которой размещены логотип и навигационное меню:

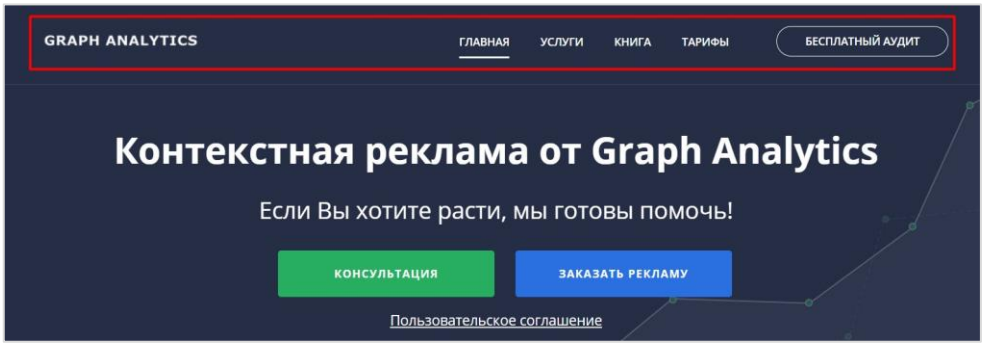


Рис. 206. Пример

Код этого блока выглядит так:

```
<header class="header" id="stricky">
<nav class="navbar navbar-default">
<div class="container">
<div class="navbar-header">
<img src="image/logo1.png" alt="">
```

Предположим, мы хотим отслеживать клики по логотипу через **div class="navbar-header"**, независимо от того, падает ли клик на элемент изображения или нет. Если мы воспользуемся триггером типа **Клики - Все элементы**, то условия, которые мы могли бы добавить в качестве активации, будут следующими:

- **Click Classes - равно - navbar-header**
- **Click Element - соответствует селектору CSS - div.navbar-header**

Если использовать расширение для браузера CSS Selector Tester, то мы можем с помощью CSS-селектора найти элемент на странице:

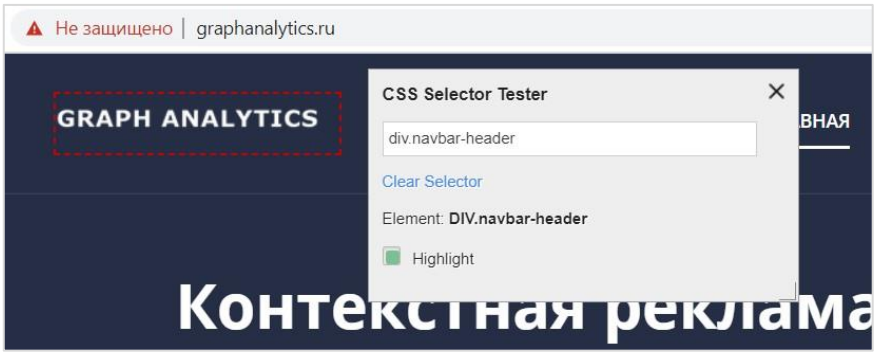


Рис. 207. Подсвеченный элемент с помощью CSS Selector Tester

Область та, условия активации триггера похожи на правду. Но это не будет работать.

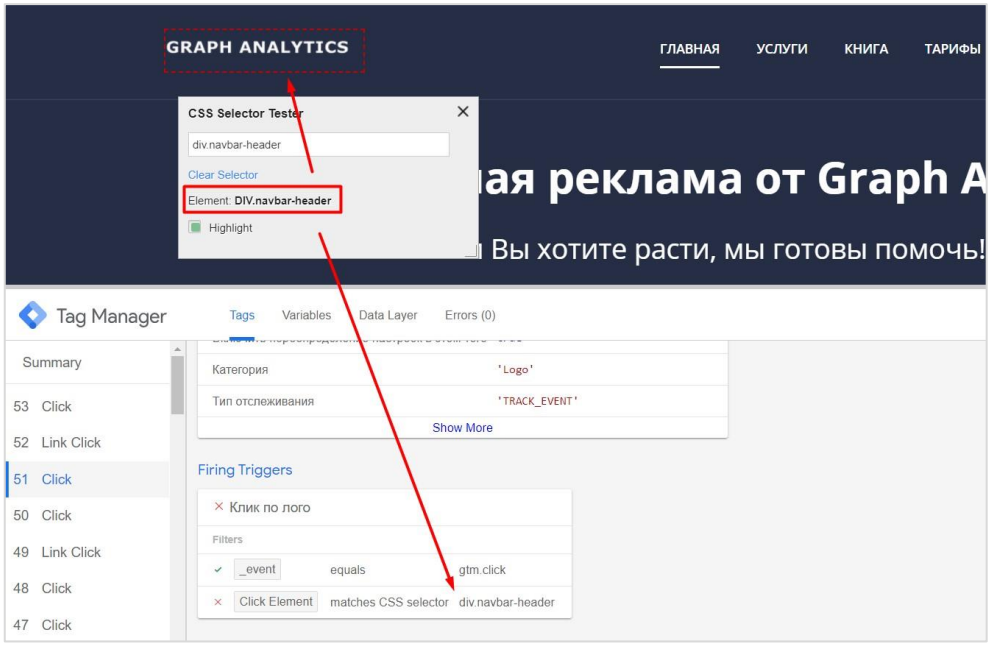


Рис. 208. Условие активации триггера не сработало

А все потому, что клик пользователя с большей вероятностью не попадет на тег , а попадет на один из вложенных элементов. Это связано как раз с элементом , который является блочным.

Когда нужно отследить клики по какому-то , то нужно прописывать селектор не только на сам , но и указывать все его вложенные элементы. А то возможна ситуация, при которой все вложенные элементы

перекроют сам и по нему невозможно будет кликнуть. Итог - событие не произойдет, триггер не сработает, тег не активируется.

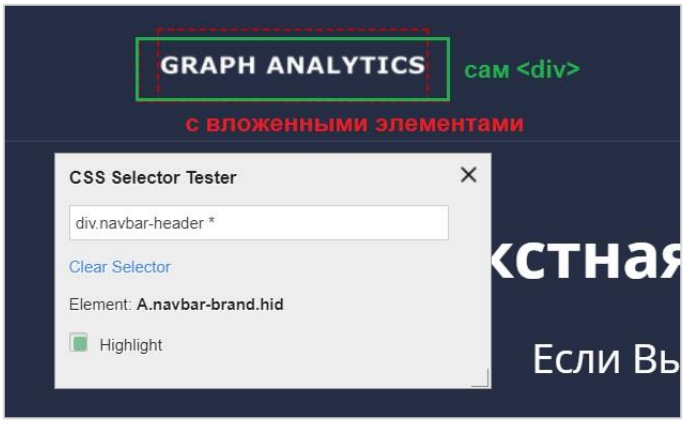


Рис. 209. Вложенный элемент перекрывает div

Как же быть? На самом деле, решение есть, и его еще в 2018 году написал **Симо Ахава (Simo Ahava)** в своем блоге (см. приложение). Здесь как раз пригодится конструкция селектора с подстановочным знаком * (wildcard), которую необходимо применять не только на элементе верхнего уровня, но и на любых вложенных элементах, которые входят в него.

Селектор, который нужно использовать: **Click Element - соответствует селектору CSS - div.navbar-header, div.navbar-header ***

Условие означает, что будут отслеживаться клики, которые попадают на **div class="navbar-header"** элемент или любой элемент, вложенный в него. В Google Tag Manager это выглядит так:

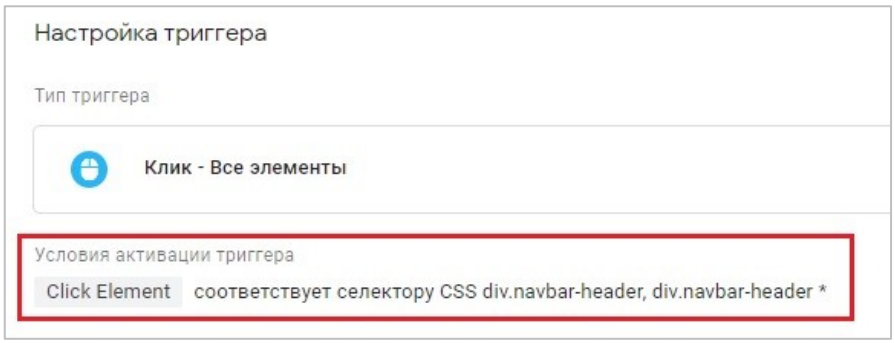


Рис. 210. Отслеживание вложенных элементов с помощью подстановочного знака *

Таким образом, вы всегда сможете отслеживать клики на соответствующем элементе, независимо от вложенной структуры HTML. Чтобы себя проверить, перейдем в режим предварительного просмотра и кликнем по логотипу на сайте.

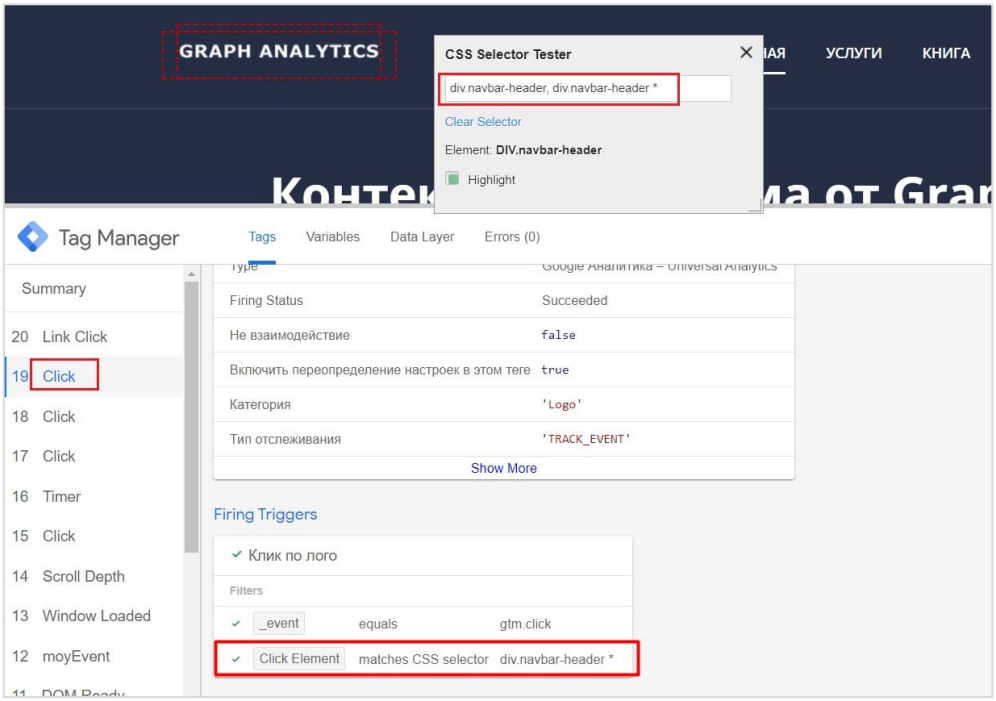


Рис. 211. Триггер сработал

Наша конструкция с подстановочным знаком * сработала!

Отслеживание событий с помощью data-атрибутов

В HTML5 для любого элемента можно использовать собственные атрибуты, начинающиеся с префикса **data-**, так называемые *атрибуты данных*.

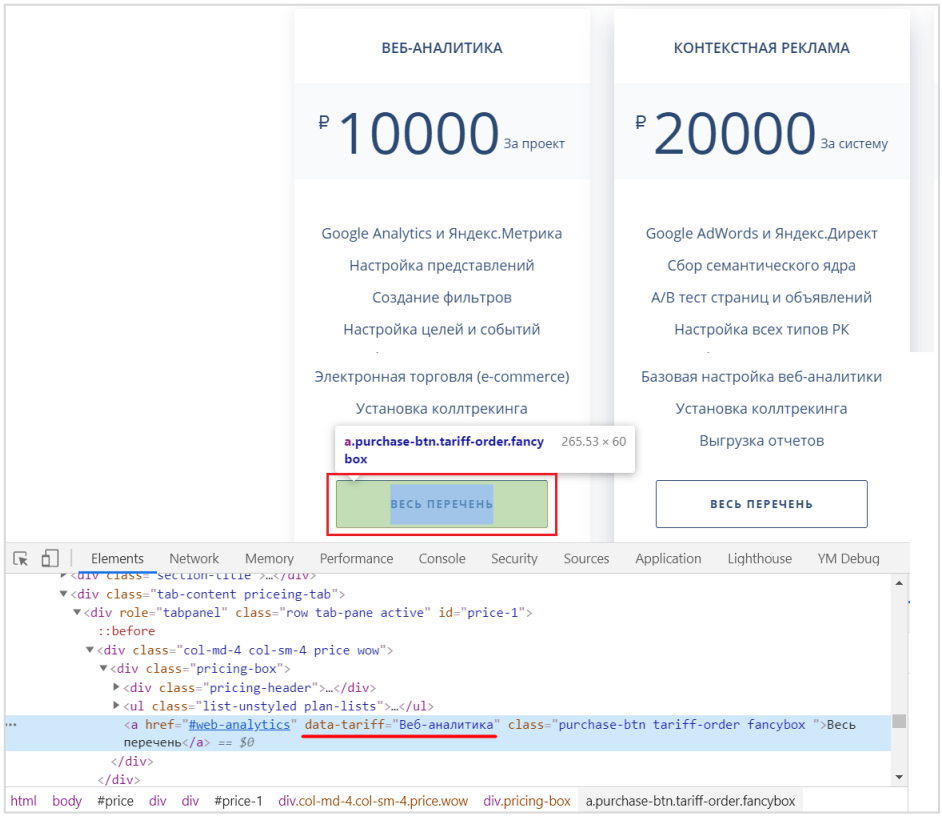


Рис. 212. Пример data-атрибута для HTML-элемента

Они позволяют хранить информацию, связанную с различными DOM-элементами, которая может помочь в работе скриптов, а также для оформления элементов через CSS. Для пользователей она может быть несущественной, а для разработчиков и веб-аналитиков упрощает жизнь.

Может произойти такое, что в процессе работы над сайтом разработчик поменяет идентификатор или класс элемента, и забудет оповестить об этом веб-аналитика. А привязавшись к данным селекторам, мы рискуем потерять часть данных, поскольку собьется настройка и все условия активации в GTM. Дата-атрибуты, как правило, не меняются при обновлении и служат гарантом корректного отслеживания. Но опять же, не всегда.

data-атрибуты всегда начинаются с приставки **data-** и заканчиваются чем-то более понятным для того, кто его создавал и использует. В имени могут содержаться латинские буквы в нижнем регистре, цифры и символы **дефис (-), двоеточие (:), подчеркивание (_)**. В моем примере выше (на рисунке) у кнопки **Весь перечень** есть data-атрибут с именем **data-tariff** и значением **Веб-аналитика**.

Элемент может содержать в себе любое количество дата-атрибутов. Например, **data-tariff (тариф), data-price (цена)** и **data-id (идентификатор услуги)**:

```
<a href="#web-analytics" data-tariff="Веб-аналитика" data-price="10000"
data-id="1" class="purchase-btn tariff-order fancybox ">Весь перечень</a>
```

В data-атрибуте могут храниться любые данные с типом string (строка) или которые можно перекодировать в строку. Преобразования типов выполняются в JavaScript. data-атрибуты в CSS можно использовать для стилизации элементов с помощью селекторов атрибутов.

Примечание: поисковые роботы не индексируют данные, содержащиеся в data-атрибутах.

Имена атрибутов трансформируются в переменные, к которым затем можно обращаться и получать значения, по следующим правилам:

- data- удаляется;
- любой дефис, идущий перед буквой, удаляется, а буква за ним становится заглавной (стиль **camelCase**);
- любые другие буквы остаются неизменными.

Например, атрибут **data-my-personal-tariff** преобразуется в переменную **myPersonalTariff**.

Чтение (извлечение) data-атрибутов

Осуществляется с помощью:

- метода **getAttribute ()**;
- **jQuery** и функции **attr ()**;
- объекта **dataset**.

Давайте разберем более подробно. В качестве примера я буду использовать тестовый проект graphanalytics.ru, на котором некоторые HTML-элементы имеют дата-атрибуты (см. рисунок выше). Для этого нам понадобится консоль разработчика (F12 - в Google Chrome, вкладка **Console**).

Пример получения всех элементов, которые присутствуют на странице с атрибутом **data-tariff**:

```
document.querySelectorAll('[data-tariff]');
```

В консоли разработчика нам вернуться все элементы с этим атрибутом:

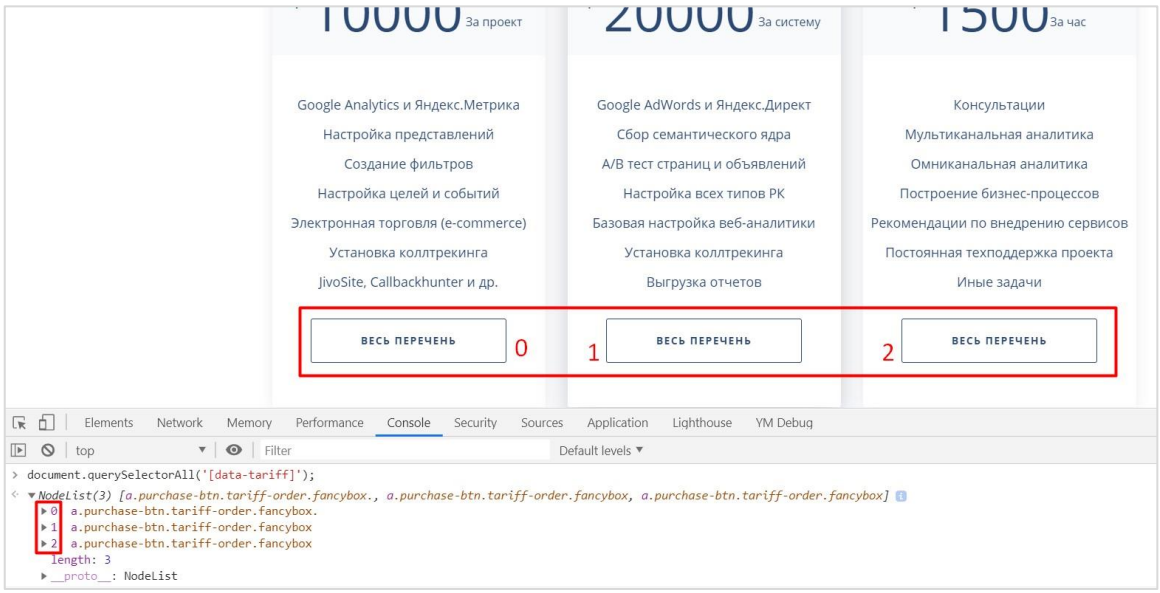


Рис. 213. Все элементы с атрибутом data-tariff

В jQuery можно использовать такое написание, команда эквивалентна вызову **document.querySelectorAll()**:

```
$$('[data-tariff]');
```

Пример получения конкретного элемента с определенным значением атрибута **data-tariff**:

```
document.querySelector('[data-tariff="Веб-аналитика"]');
```

В консоли разработчика возвращается только один элемент со значением **Веб-аналитика**:

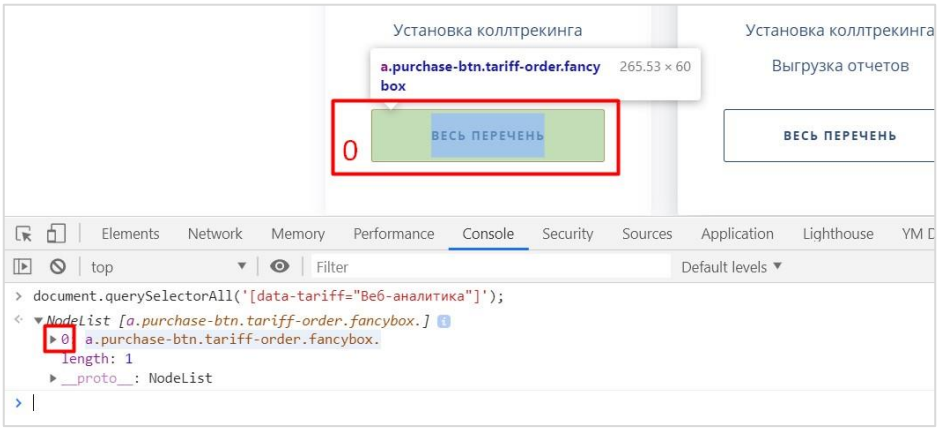


Рис. 214. Элемент с определенным значением атрибута data-tariff

В jQuery можно использовать такое написание:

```
$$('[data-tariff="Веб-аналитика"]');
```

Пример получения значения из data-атрибута с помощью метода **getAttribute** для первого элемента:

```
var myTariff = document.getElementsByClassName ('purchase-btn');
var tariff = myTariff[0].getAttribute('data-tariff');
tariff
```

Сначала с помощью переменной **myTariff** и команды **document.getElementsByClassName** возвращается значение всех элементов по имени класса **purchase-btn**. Затем с помощью переменной **tariff** получаем из атрибута **data-tariff** по **0** элементу значение **Веб-аналитика**.

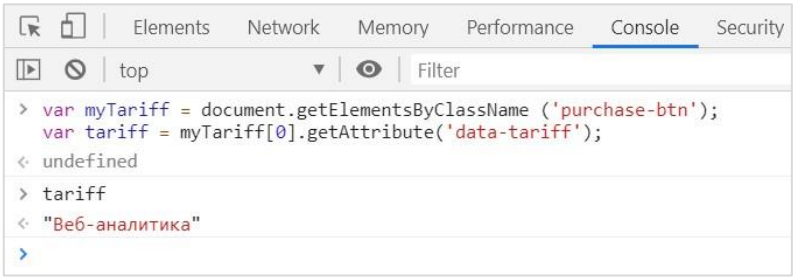


Рис. 215. Пример метода getAttribute в консоли разработчика

Для 0 элемента - **Веб-аналитика**, для 1 элемента - **Контекстная реклама**, для 2 элемента - **Консультация**. В Google Tag Manager мы могли бы создать пользовательскую переменную типа **Собственный код JavaScript** и добавить туда такую конструкцию:

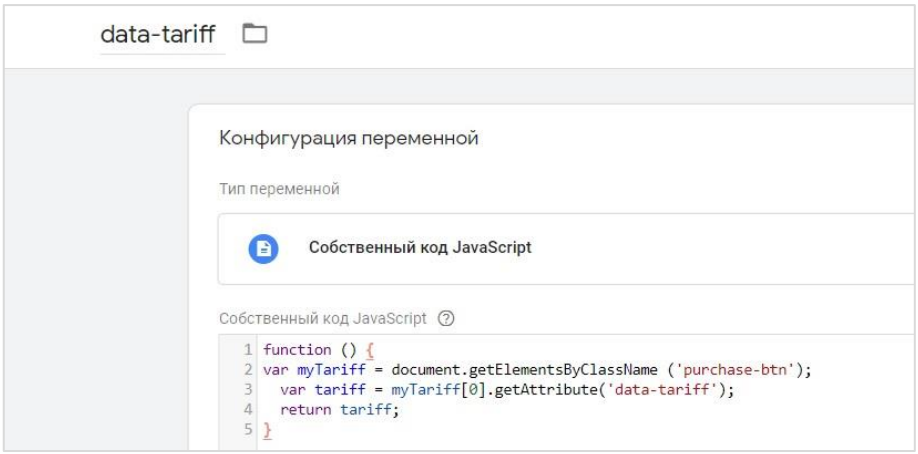


Рис. 216. Пример метода getAttribute в GTM

Чтобы извлечь значение атрибута с помощью jQuery и функции **attr ()**, можно воспользоваться следующей конструкцией (для моего примера!):

```
$('.purchase-btn').eq(0).attr('data-tariff');
```

, где метод **.eq(0)** позволяет выбрать элемент с конкретным индексом (у меня 0) из набора выбранных элементов, чтобы получить значение **Веб-аналитика**.

Можно в jQuery использовать и такую команду (метод **.data**):

```
$('.purchase-btn').eq(0).data('tariff')
```

Но есть и более простой способ чтения data-атрибутов, который мы будем использовать в Google Tag Manager. Для этого используется объект **dataset**. Чтобы получить data-атрибут, необходимо взять свойство объекта dataset и написать его с учетом правил, о которых я написал выше (без префикса data-, без дефисов и с помощью стиля camelCase). Подробнее о свойстве **dataset** читайте в документации Mozilla (см. приложение).

Я покажу пример отслеживания клика по кнопке Весь перечень. Событие будет передаваться стандартным способом со значением дата-атрибута (тарифа) в инструменты веб-аналитики с помощью соответствующих тегов. Для этого необходимо создать пользовательскую переменную типа **Переменная уровня данных** со значением, которое соответствует конструкции:

```
gtm.element.dataset.{имя атрибута по правилам с помощью стиля camelCase}
```

В результате из data-tariff в GTM я получу переменную **gtm.element.dataset.tariff**:

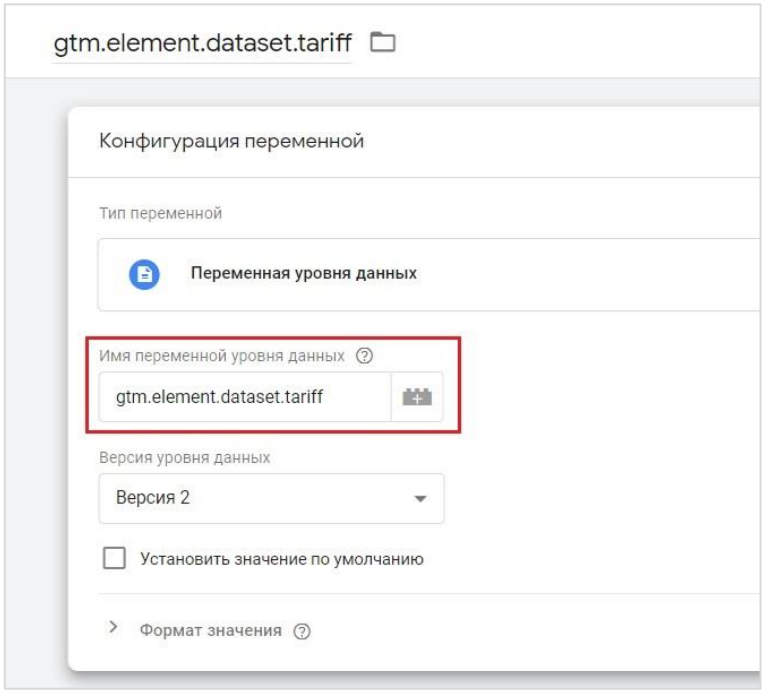


Рис. 217. Переменная уровня данных gtm.element.dataset.tariff

Далее следует создать триггер типа **Клики - Все элементы** с условием активации - **Некоторые клики - Click Element - соответствует селектору CSS - [data-tariff], [data-tariff] ***

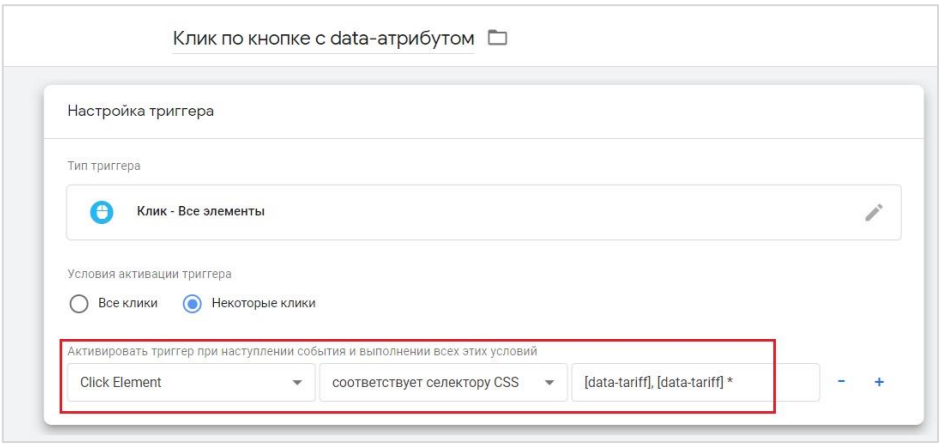


Рис. 218. Триггер активации по дата-атрибуту

Звездочка означает, что клики будут фиксироваться на любых вложенных элементах. Сохраняем триггер. Осталось создать тег **Google Аналитика - Universal Analytics** с типом отслеживания **Событие**.

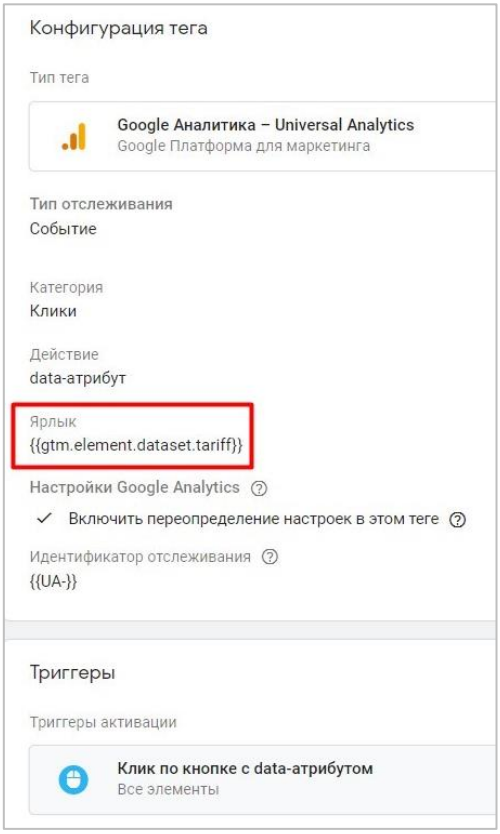


Рис. 219. Тег Google Аналитика – Universal Analytics

В качестве **Категория** и **Действие** я указал произвольные значения, а в **Ярлык** поместил переменную уровня данных **gtm.element.dataset.tariff**, которая будет передавать значение дата-атрибута элемента, по которому был клик. Сохраняем.

Проверить корректность настройки можно с помощью режима отладки GTM и отчетов В режиме реального времени Google Analytics. Кликнув несколько раз по разным кнопкам, в отчете появятся данные по нашим событиям:

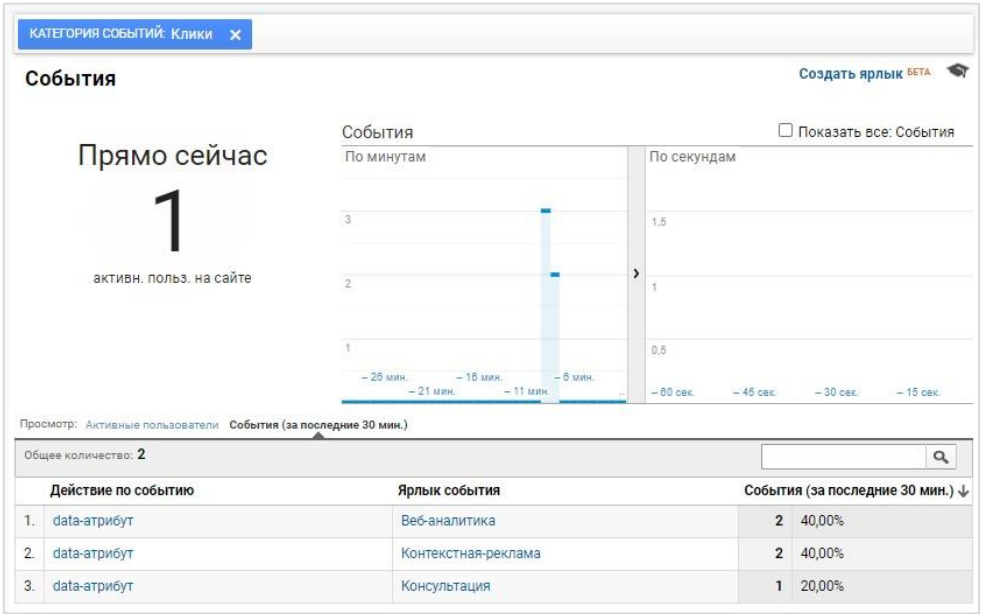


Рис. 220. События со значениями дата-атрибутов

Через некоторое время все данные по событиям появятся в отчете **Поведение — События — Лучшие события.**

Расширения для браузеров

CSS Selector Tester

Чтобы проверить правильность выбора определенного элемента для настройки триггера и его отладки в GTM, можно воспользоваться специальным расширением для браузера Google Chrome, которое называется CSS Selector Tester. Скачать его можно по ссылке (см. приложение).

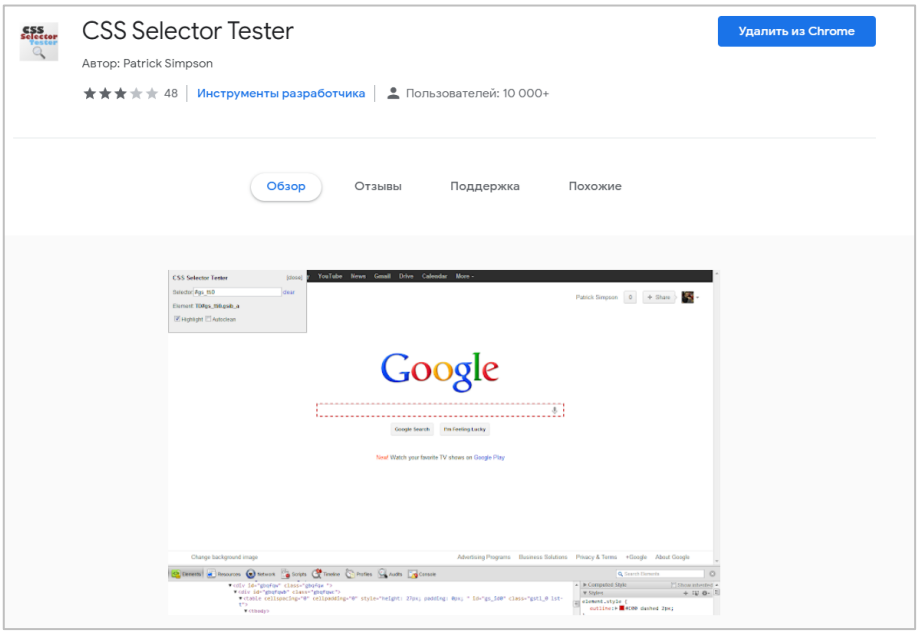


Рис. 221. CSS Selector Tester

После установки в правом верхнем углу у вас должен отобразиться новый значок. Перейдите на необходимую страницу, где хотите найти селектор и нажмите на иконку:

Рис. 222. Иконка CSS Selector Tester

У вас откроется дополнительное окно со строкой ввода селектора. Просто вставьте ваш селектор, и он подсветится на странице. В качестве примера я добавлю *a.button.hp-button-1.slide-button.w-button.green.fancybox* – укороченную версию селектора, который мы определили с помощью консоли разработчика.

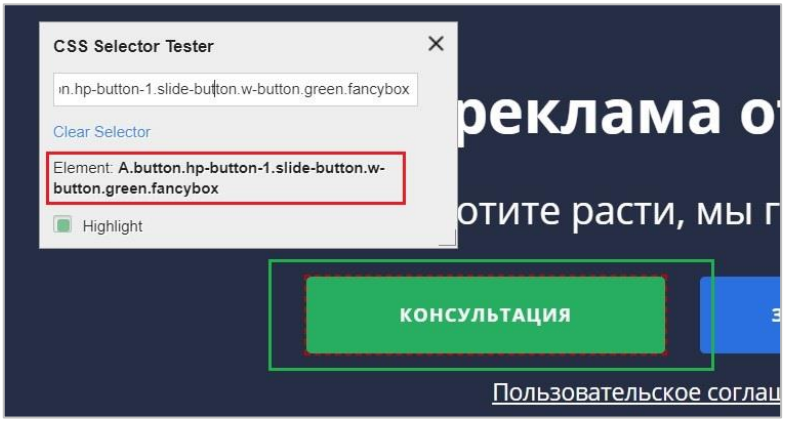


Рис. 223. Поиск селектора на странице

CSS Selector Tester нашел такой элемент на странице и выделил его границы красной пунктирной линией. Вот так он выделил все элементы div на странице:

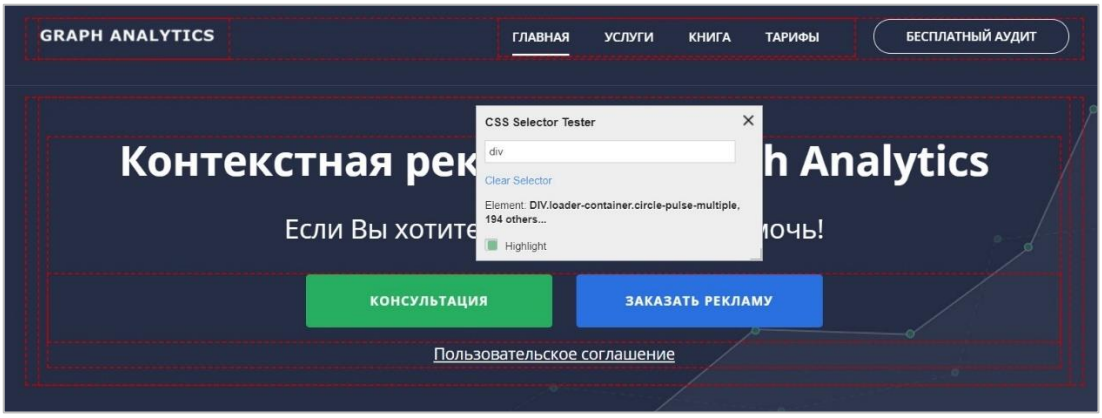


Рис. 224. Элементы div на странице

Если бы мы ввели неверный селектор, то сообщение было бы **undefined (не определено)** и элемент не был бы найден на странице.

SelectorGadget

Для тех, кто не хочет использовать в консоль разработчика, инспектировать там элементы страницы и копировать конечный вариант селектора, может воспользоваться еще одним расширениям для браузера Google Chrome - **SelectorGadget**. Оно позволяет легко определять селекторы CSS (при наведении на элемент в отдельном поле показывает его значение) и для наглядности выделяет выбранный элемент зеленым цветом, а схожие – желтым.

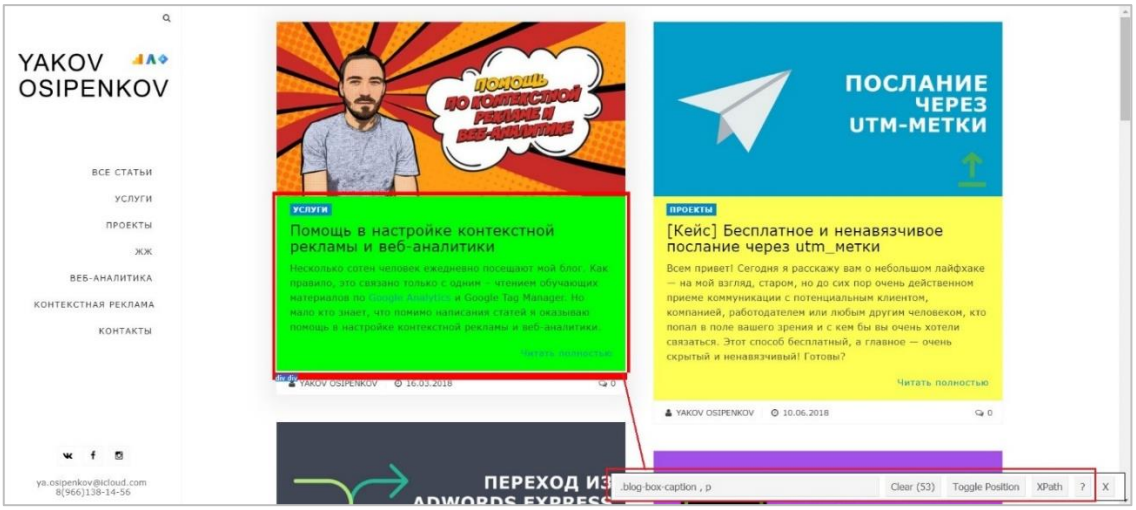


Рис. 225. Пример работы расширения SelectorGadget

Скачать расширение **SelectorGadget** можно по ссылке (см. приложение). После установки в правом верхнем углу у вас должен отобразиться новый значок.

Рис. 226. Иконка SelectorGadget

Глава 4

Переменные

Типы данных

Большинство языков программирования содержат встроенные типы данных. В JavaScript есть 8 основных типов:

1. **number** (число);
2. **bigint** (целое число произвольной длины);
3. **string** (строка);
4. **boolean** (булев, логический тип)
5. **null** (неизвестный, пустой тип);
6. **undefined** (не было присвоено значение);
7. **object** (объект);
8. **symbol** (символ);

Все вышеперечисленные типы, кроме **object**, называются *примитивами*. **Примитив** – это данные, которые не являются объектом и не имеют методов. Все типы данных в JavaScript, кроме объектов, являются *иммутабельными* (значения не могут быть модифицированы, а только перезаписаны новым полным значением). Существует еще девятый тип данных - **function** (функция), специальный случай, упрощающий определение типа для функций несмотря на то, что все функции конструктивно унаследованы от **object**.

JavaScript является *динамически типизированным* языком. Это означает, что вам не нужно определять тип переменной заранее. Он определится автоматически во время выполнения программы. Также это означает, что одну и ту же переменную можно использовать для хранения данных различных типов. Например, в один момент переменная может принять строковое значение, а в другой – числовое:

```
// Не будет ошибкой
var variable = "привет";
variable = 123456;
```

С каждым из этих типов данных вы можете встретиться при работе с Google Tag Manager. Например, в режиме предварительного просмотра на вкладке **Variables (Переменные)** отображается список ваших переменных с определенными значениями для каждого события и типами данных:

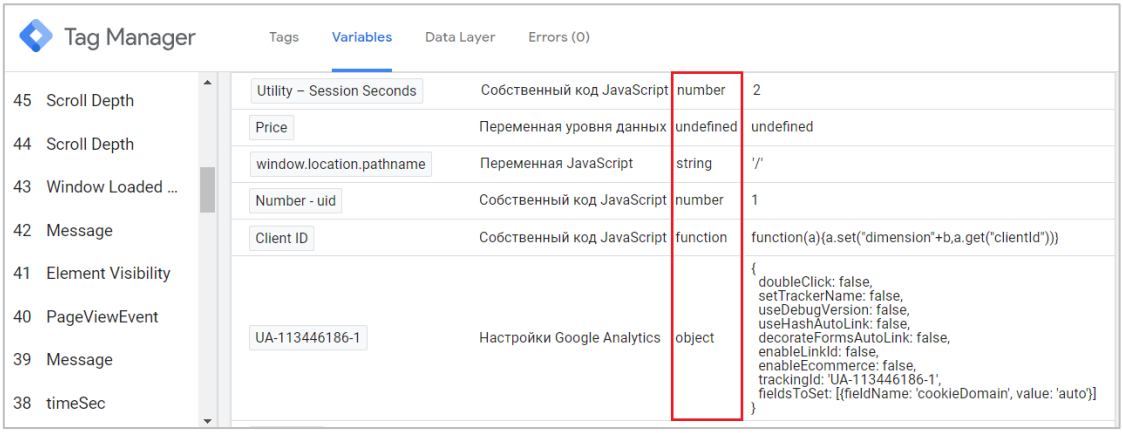


Рис. 227. Типы данных в Google Tag Manager

Рассмотрим типы данных подробнее.

Примечание: все нижеописанные примеры вы можете легко воспроизвести в своем браузере в консоли разработчика (клавиша F12 для Google Chrome) на вкладке **Console (Консоль)**.

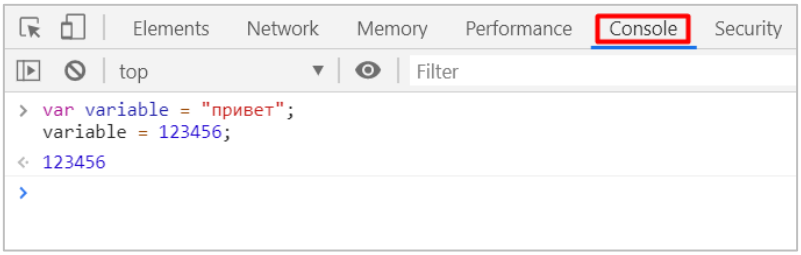


Рис. 228. Приведенные примеры можно повторить в консоли разработчика

number (число)

Тип данных **number (число)** представляется в формате 64-битного числа двойной точности с плавающей запятой.

Числа в JavaScript могут иметь две формы:

1. **целые числа**, например, 19. Мы можем использовать как положительные, так и отрицательные числа. Диапазон используемых чисел: от -2^{53} до 2^{53}
2. **дробные числа** (числа с плавающей точкой), например, 5.6675. Опять же можно использовать как положительные, так и отрицательные числа. Для чисел с плавающей точкой используется тот же диапазон: от -2^{53} до 2^{53}

В качестве разделителя между целой и дробной частями, как и во многих языках программирования, в JavaScript используется точка:

```
var n = 19;
n = 22.521;
```

Как правило, переменные с таким типом хранят в себе количественную информацию. Например: данные о стоимости товара, доставке, налоге, количестве товара, об итоговой сумме покупки и т.д. В Google Tag Manager переменные с числовым типом данных в режиме отладки подсвечиваются зеленым цветом:

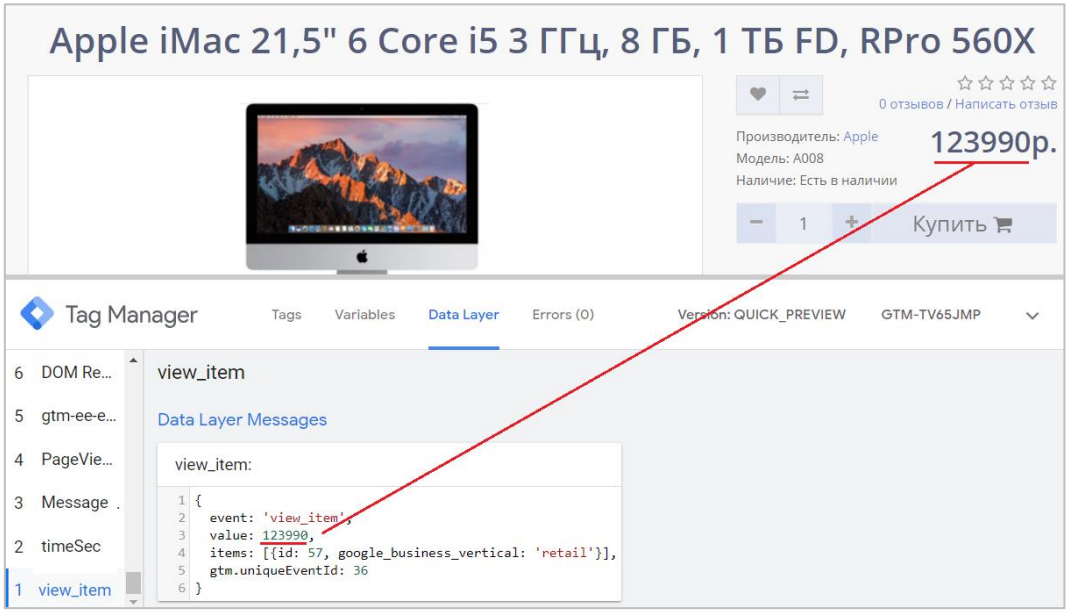


Рис. 229. Числа в диспетчере тегов в режиме отладки подсвечиваются зеленым цветом

Кроме обычных чисел, существуют так называемые *специальные числовые значения*, которые относятся к типу number: **+Infinity (положительная бесконечность)**, **-Infinity (отрицательная бесконечность)** и **NaN (Not-A-Number, не число)**.

Infinity представляет собой математическую бесконечность ∞. Получить его можно в результате деления на ноль:

```
1 / 0; // Infinity
```

Или задать явно:

```
var n = Infinity;
n = Infinity;
```

Ноль в JavaScript имеет два представления: -0 и +0. (0 это синоним +0). На практике это имеет малозаметный эффект. Например, выражение +0 === -0 является истинным. Чтобы получить -∞ (-Infinity), достаточно число разделить на -0, а чтобы получить +∞ (Infinity), необходимо число разделить на +0:

```
2 / +0; // Infinity
2 / -0; // -Infinity
```

NaN означает вычислительную ошибку. Чаще всего, это результат неправильной математической операции, проделываемой между двумя разными типами данных. Например, деление переменной со строковым значением на числовое является ошибкой:

```
"привет" / 2; // NaN
true * 5; // NaN
```

Если где-то в математическом выражении есть NaN, то результатом вычислений с его участием будет NaN.

Существует множество операций для чисел, например, **умножение *, деление /, сложение +, вычитание** - и так далее. А также оператор **typeof**, который позволяет возвращать тип аргумента. Их мы рассмотрим чуть позже.

BigInt (целое число произвольной длины)

BigInt числа дают возможность работать с целыми числами произвольной длины, когда необходимо выйти за пределы 2^{53} или -2^{53}.

Чтобы создать значение типа BigInt, необходимо добавить **n** в конец числового литерала или вызвать функцию BigInt, которая создаст число типа BigInt из переданного аргумента. Аргументом может быть число, строка и другие типы:

```
// символ "n" в конце означает, что это BigInt
const bigInt1 = 12345678901234567890123456789012345678901234567890n;
const bigInt2= BigInt("12345678901234567890123456789012345678901234567890");
```

Google Tag Manager не поддерживает этот тип данных, поскольку использует изолированный JavaScript.

string (строка)

Любые текстовые данные в JavaScript являются **строками (string)**. И вся текстовая информация должна быть заключена в кавычки:

```
var variant1 = "привет";
variant2 = 'Как дела?';
variant3 = `У меня все хорошо ${variant1}`; // У меня все хорошо привет
```

В JavaScript используется три типа кавычек:

1. **Двойные кавычки:** "Привет".
2. **Одинарные кавычки:** 'Привет'.
3. **Обратные кавычки:** `Привет`.

Двойные или одинарные кавычки являются *простыми*, между ними нет разницы в JavaScript. Единственное ограничение - тип закрывающей кавычки должен быть тот же, что и тип открывающей, то есть либо обе двойные, либо обе одинарные.

Обратные кавычки появились в стандарте ECMAScript 6 (ES6). Они позволяют встраивать выражения в строку, заключая их в **${...}** В примере выше переменная **variant3** примет значение У меня все хорошо привет, поскольку мы в ${...} использовали переменную **variant1**, которая равна слову привет. Выражение внутри ${...} вычисляется, и его результат становится частью строки.

Строковые переменные чаще всего встречаются в Google Tag Manager. В режиме отладки они подсвечиваются красным цветом:

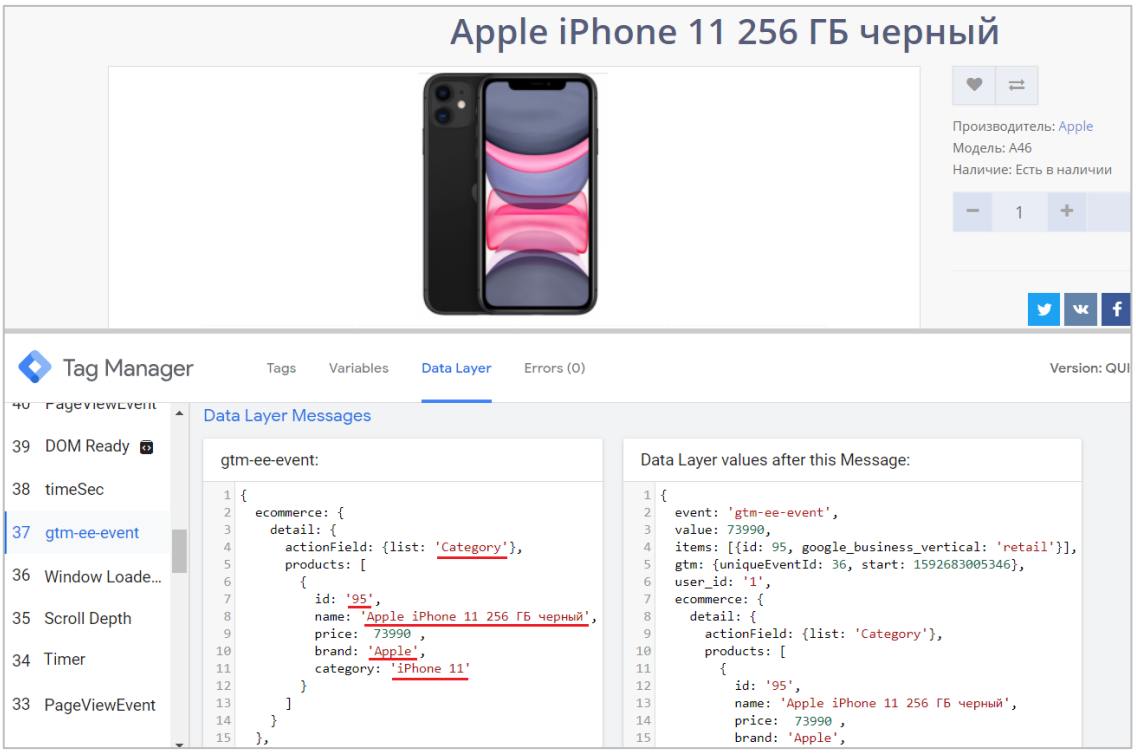


Рис. 230. Строковые данные в диспетчере тегов в режиме отладки подсвечиваются красным цветом

Все спецсимволы начинаются с обратного **слеша **

Если внутри строки встречаются кавычки, то мы их должны экранировать слешем. Например, у нас есть текст "Tag Manager – инструмент компании "Google"". Теперь экранируем кавычки:

```
var GTM1 = "Tag Manager – инструмент компании \"Google\"";
var GTM2 = "Tag Manager – инструмент компании \'Google\'"; // другой тип кавычек
```

boolean (булев, логический тип)

Булевый тип **(boolean)** представляет логическую сущность и может принимать только два состояния: **true (истина)** и **false (ложь)**.

Как правило, используется для хранения значений:

- Да: true значит да, правильно;
- Нет: false значит нет, не правильно.

В JavaScript boolean часто применяется для того, чтобы определить какие части кода выполнять (например, в операторах **if**) или повторять (например, циклы **for**). В Google Tag Manager логический тип используется в собственных переменных JavaScript и для запуска триггеров активации и блокировки. Например, можно выполнить проверку – зашел ли пользователь на сайт с мобильного устройства? Если да, то в переменную вернуть значение *true*. Если нет, то *false*.

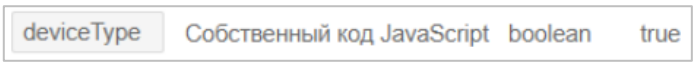


Рис. 231. Пример логического типа в Google Tag Manager

И далее уже выполнять различные команды в зависимости от полученного результата.

Булевые значения также могут быть результатом сравнений:

```
4 > 1 // true
100 == 101 // false
'56' > '52' // true
```

Подробную информацию о типах данных в JavaScript читайте на сайте javascript.ru и в официальной документации Mozilla (см. приложение)

null (неизвестный, пустой тип)

null (неизвестный, пустой тип) – это специальное значение, которое представляет собой *ничего*, *пусто* или *значение неизвестно*. Другими словами, переменная имеет некоторое неопределенное значение. Это не число, не строка, не логическое значение, но все-таки оно есть.

```
var city = null; // null
```

Если у вас есть переменная с типом **null**, и вам необходимо преобразовать ее в какое-нибудь другое значение, то это можно сделать с помощью Google Tag Manager и, например, пользовательской переменной типа **Переменная JavaScript**, выбрав в формате значения соответствующий параметр:

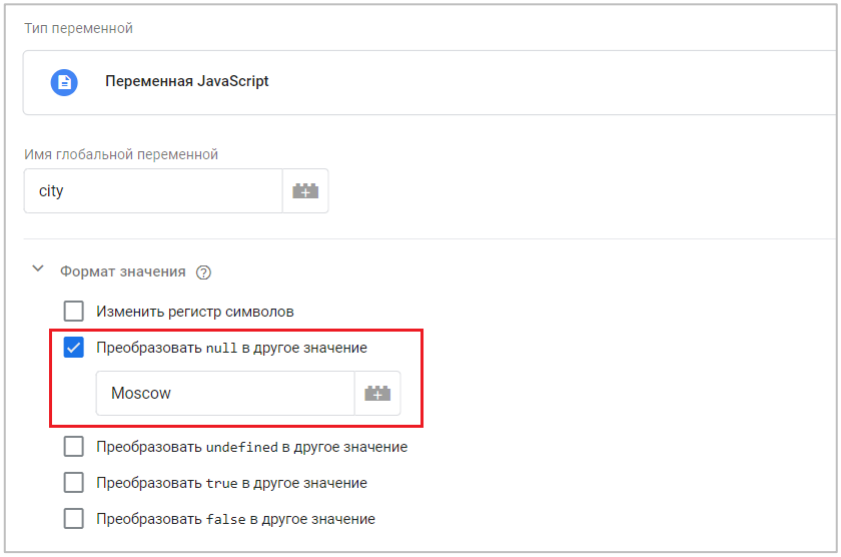


Рис. 232. Настройки формата значений в переменных Google Tag Manager

undefined (значение не было присвоено)

Если переменная объявлена, но ей не присвоено никакого значения, то ее значением будет **undefined**.

Часто возникает путаница между **null** и **undefined**. Отличие заключается в том, что у переменной, которой присвоено значение **null**, есть некоторое неопределенное значение. А у **undefined** вообще нет никакого значения, мы только ее объявили.

```
var x; // undefined
var y = undefined; // undefined
```

null, преимущественно, используется для присвоения переменной *пустого* или *неизвестного значения*, а undefined - для проверок, была ли переменная назначена.

undefined может появляться в некоторых переменных при настройке в Google Tag Manager, а также тогда, когда вы выполнили какое-нибудь действие и в режиме предварительного просмотра просматриваете значения полученных переменных.

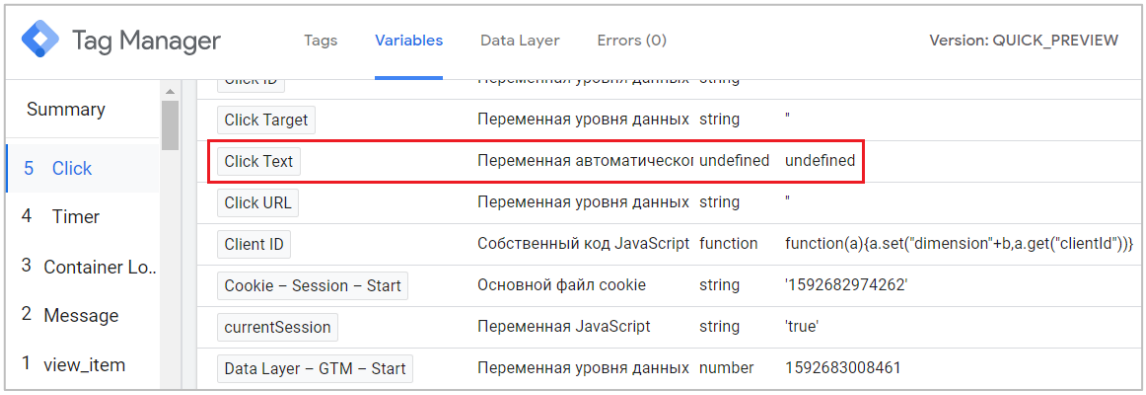


Рис. 233. Значение undefined в режиме отладки

Режим отладки GTM показывает значения переменных, которые существовали на момент совершения события. Если вы видите undefined, скорее всего, переменна на момент совершения события еще не была присвоена. Это распространенное явление, поскольку в диспетчере тегов мы настраиваем большое количество различных переменных, а значения им могут присваиваться только в момент совершения конкретного действия. В остальное время переменные будут существовать (объявлены, активированы), но значения им присвоены не будут.

Аналогично и с порядком активации событий. Например, если вы видите, что данные помещаются в уровень данных (dataLayer) после фрагмента контейнера Google Tag Manager **(Page View, он же Container Loaded,**

gtm.js), то они не будут доступны для триггеров **Просмотр страницы (All Pages)**. В этом случае вы должны использовать триггер **Модель DOM готова (DOM Ready, gtm.dom)** или **Окно загружено (Window Loaded, gtm.load)**. Подробнее это мы разберем в следующих главах.

Если у вас есть переменная с изначальном типом **undefined**, и вам необходимо преобразовать ее в какое-нибудь другое значение, то это можно сделать, например, с помощью пользовательской переменной типа **Переменная JavaScript**, выбрав в формате значения соответствующий параметр:

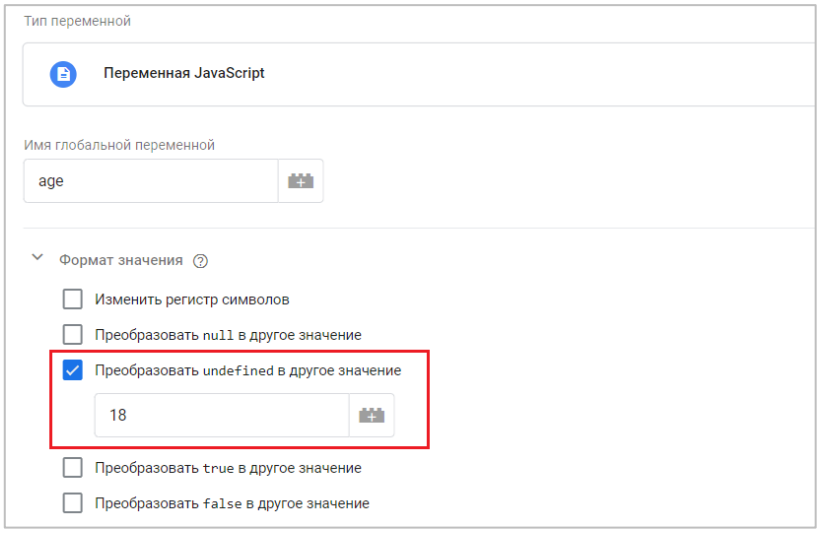


Рис. 234. Преобразование undefined в другое значение у переменной JavaScript

object (объект)

object (объект) – особенный тип данных, поскольку может использоваться для хранения набора свойств и более сложных объектов, а не только простых, примитивных значений, которые содержат в себе только что-то одно (будь то строка или число, или что-то еще).

В JavaScript объекты (ассоциативные массивы) используются очень часто. Объект может быть создан:

```
var ecommerce = new Object(); // синтаксис "конструктор объекта"
var ecommerce = {}; // с помощью фигурных скобок {…}
```

Вариант с фигурными скобками **{...}** еще называют *литералом объекта* или *литеральной нотацией*.

Свойство – это пара *ключ:значение*, где **ключ** – это строка или символ (также называемая *именем свойства*), а **значение** может быть любого типа данных. Свойства объекта также иногда называют *полями объекта*.

Например:

```
var google = {     // объект
   founder1: "Larry",  // под ключом "founder1" хранится значение "Larry"
   founder2: "Sergey",  // под ключом "founder2" хранится значение "Sergey"
   age: 1998 // под ключом "age" хранится значение 1998
};
```

В объекте **google** находятся три свойства:

1. Первое свойство с именем "founder1" и значением "Larry";
2. Второе свойство с именем "founder2" и значением "Sergey";
3. Третье свойство с именем "age" и значением 1998.

Имя свойства может состоять из нескольких слов, но тогда оно должно быть заключено в кавычки:

```
var google = {
   founder1: "Larry",
   founder2: "Sergey",
   age: 1998,
```

```
    "Number of employees": 114096 // Число сотрудников
};
```

Последнее свойство объекта может заканчиваться запятой:

```
var google = {
    founder1: "Larry",
    founder2: "Sergey",
    age: 1998,
    "Number of employees": 114096,
};
```

Это называется *висячая запятая*. Такой подход упрощает добавление, удаление и перемещение свойств, так как все строки объекта становятся одинаковыми.

Мы можем записать объект таким образом, что ключ будет иметь несколько значений. Например, вот так:

```
var google = {
    founders: ["Larry", "Sergey"],
    age: 1998,
    "Number of employees": 114096,
};
```

Получить значение свойства можно с помощью операторов точки **.** (так называемая *точечная нотация, точечная запись*) или квадратных скобок **[]** (так называемая *скобочная нотация, скобочная запись*) и кавычек (подойдет любой тип кавычек).

```
var google = {
    founders: ["Larry", "Sergey"],
    age: 1998,
    "Number of employees": 114096,
};

google.founders // ["Larry", "Sergey"]
google["founders"] // ["Larry", "Sergey"]
```

Для получения доступа к значению в массиве используются квадратные скобки, за которыми следует номер позиции. Чтобы получить значение "Sergey", нам необходимо написать так:

```
var google = {
    founders: ["Larry", "Sergey"],
    age: 1998,
    "Number of employees": 114096,
};

google.founders[1] // "Sergey"
google["founders"][1] // "Sergey"
```

В квадратных скобках идет номер позиции [1], поскольку массивы в JavaScript индексируются с нуля: первый элемент массива имеет индекс, равный 0, а индекс последнего элемента равен значению свойства массива length минус 1 (length-1).

Объекты можно помещать в другие объекты. Например:

```
var analytics = {
    tool: "Google Analytics",
    data: {
      owner: "Google",
      commercial: "Yes",
      url: "analytics.google.com",
      launched: "2005"
    }
};
```

Здесь объект **data** с 4 ключами (owner, commercial, url, launched) помещен в другой объект **analytics**. Получить значение свойства объекта внутри другого объекта можно также с помощью точечной нотации (оператора точки) или квадратных скобок.

```
var analytics = {
   tool: "Google Analytics",
   data: {
     owner: "Google",
     commercial: "Yes",
     url: "analytics.google.com",
     launched: "2005"
   }
};

analytics.data.url // " analytics.google.com"
analytics["data"]["launched"] // " 2005"
analytics.tool // " Google Analytics"
```

Используя object, вы можете создать массив с данными для отслеживания различных событий электронной торговли или динамического ремаркетинга, например: *просмотр товара, добавление/удаление товаров из корзины, покупка* и т.д.

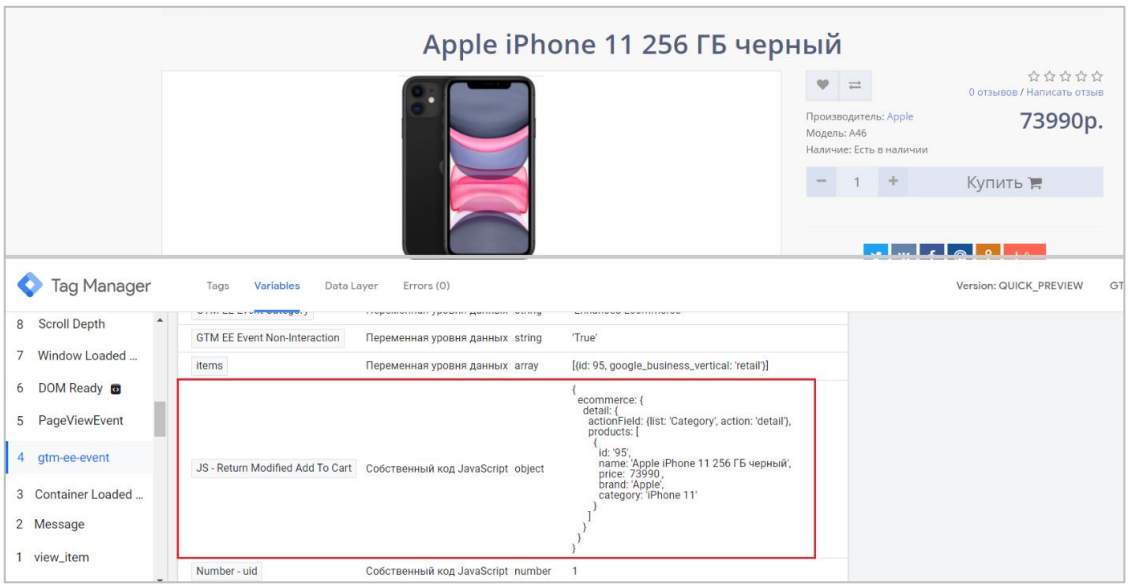


Рис. 235. Пример объекта в режиме предварительного просмотра

В JavaScript есть другие типов объектов:

- **Array** для хранения упорядоченных коллекций данных,
- **Date** для хранения информации о дате и времени,
- **Error** для хранения информации об ошибке.

Формально они не являются отдельными типами, а относятся к типу данных object. Они лишь расширяют его различными способами. Google Tag Manager также будет их корректно определять. Например, array:

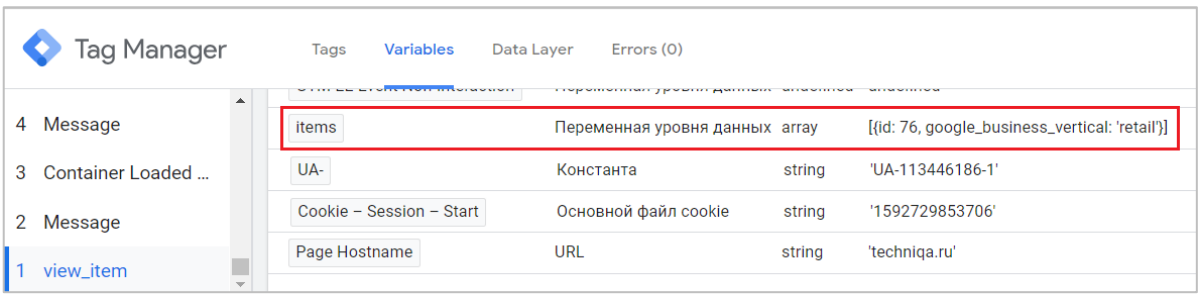


Рис. 236. Пример массива в режиме отладки

symbol (символ)

Символ (symbol) – одно из нововведений в стандарте ECMAScript 6 (ES6), примитивный тип данных, представляющий собой неизменяемый, уникальный идентификатор. Символы создаются с помощью функции **Symbol()**:

```
var mySymbol = Symbol();
```

При создании символу можно дать описание (необязательный параметр), в основном использующееся для отладки кода:

```
var mySymbol = Symbol("Описание символа");
```

Важной особенностью символа также является то, что его значение уникально. Вы можете использовать символ как имя свойства, которое гарантированно не будет повторяться с любым другим свойством. Даже, если у двух символов одинаковое имя, то это не значит, что они равны:

```
var mySymbol1 = Symbol("Описание символа");
var mySymbol2 = Symbol("Описание символа");
mySymbol1 == mySymbol2 // false
```

На примере выше два символа с одинаковым описанием, но они не равны. Символы в JavaScript используются для безопасного добавления нового функционала в программу, а также для защиты от совпадения имен с уже существующими объектами. Подробнее о данном типе данных вы можете прочитать в этой статье (см. приложение).

В Google Tag Manager я пока не встречал задач, связанных с использованием данного типа данных. Сам GTM в режиме предварительного просмотра определяет **symbol** как **object**:

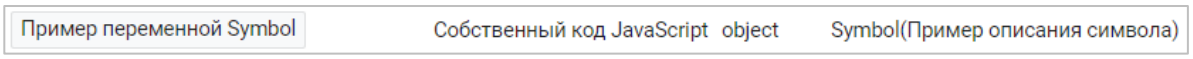


Рис. 237. Symbol в Google Tag Manager

Выражения и операторы

Выражения в JavaScript представляют собой комбинации *операндов* и *операторов*.

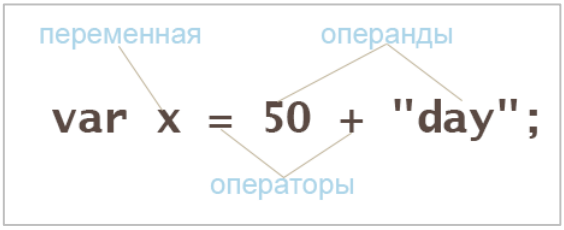


Рис. 238. Операнды и операторы

Операнды — это данные, обрабатываемые сценарием JavaScript. Операнды могут быть числами, строками, логическими величинами или объектами. Иногда операнды называют *аргументами*.

Операторы — это символы языка, выполняющие различные операции с данными. Операторы могут записываться с помощью символов пунктуации или ключевых слов.

Операции в выражениях выполняются последовательно в соответствии со значением приоритета (чем больше значение приоритета, тем он выше). Возвращаемый результат не всегда имеет значение того же типа, что и тип обрабатываемых данных. Например, в операциях сравнения участвуют операнды различных типов, но возвращаемый результат всегда будет логического типа.

В зависимости от количества операндов различают следующие типы операторов:

- **унарный** — оператор, который применяется к одному операнду;
- **бинарный** — оператор, который применяется к двум операндам;

- **тернарный** — оператор, который применяется к трем операндам (условие, за которым следует знак вопроса (?), затем выражение, которое выполняется, если условие истинно, сопровождается двоеточием (:), и, наконец, выражение для выполнить, если условие ложно. Он часто используется в качестве укороченного варианта условного оператора if).

Бинарная операция использует два операнда, один перед оператором и другой за ним:

```
// операнд1 оператор операнд2
3 + 4;
4 * 5;
```

Унарная операция использует один операнд, перед или после оператора:

```
// оператор операнд или операнд оператор
X++;
--y;
```

Простейшая форма выражения – **литерал**. Это любое значение, указанное явным образом в коде. В качестве литералов в JavaScript могут выступать числа, строки (текстовые значения), логические значения. Например: число 2012, строка "Google Tag Manager" и т.д.

Примечание: в рамках знакомства с JavaScript я привожу сокращенную информацию о переменных, типах данных, операторах, поскольку цель данной главы состоит в том, чтобы познакомить вас с основными понятиями, которые могут быть

Операторы присваивания

Оператор присваивания (=) используется для присваивания значения переменной. В результате операции операнду слева от оператора присваивания устанавливается значение, которое берется из правого операнда. Таким образом, выражение x = y означает, что x присваивается значение y.

```
var y = 5; // y присвоено значение 5
```

Оператор присваивания имеет ассоциативность справа налево, поэтому при наличии в выражении нескольких операторов присваивания они вычисляются справа налево. Благодаря этому можно написать код, присваивающий одно значение нескольким переменным:

```
var a, b, c;
a = b = c = 300; // a,b,c присвоено значение 300
```

Также существуют **составные операторы** присваивания, которые используются для сокращенного представления операций:

Оператор / Операция	Сокращенный оператор	Эквивалентно
Присваивание со сложением	x += y	x = x + y
Присваивание с вычитанием	x -= y	x = x - y
Присваивание с умножением	x *= y	x = x * y
Присваивание с делением	x /= y	x = x / y
Присваивание по модулю	x %= y	x = x % y
Присваивание с левым сдвигом	x <<= y	x = x << y
Присваивание с правым сдвигом	x >>= y	x = x >> y
Присваивание с беззнаковым сдвигом вправо	x >>>= y	x = x >>> y
Присваивание с побитовым AND	x &= y	x = x & y
Присваивание с побитовым XOR	x ^= y	x = x ^ y
Присваивание с побитовым OR	x |= y	x = x | y

Например, выражение **x += 10** эквивалентно **x = x + 10**. Составные операторы только сокращают объем кода, но не увеличивают его быстродействие.

Примеры присваиваний:

```
var a = 5; var b = 10; var c = 15;
```

```
a = b // переменной a присваивается значение b, результат равен 10
a = b = c // все значения переменных равны 15

var str = "Привет "; var num = 3;
str += "мир"; // "Привет мир", так как строки поддерживают конкатенацию
num += 2; // 5, поскольку num = num + 2

var num = 5;
num -= 2; // 3, присвоение с вычитанием

var num = 5;
num *= 5; // 25, присвоение с умножением

var num = 25;
num /= 5; // 5, присвоение с делением

var num = 5;
num %= 2 // 1, присвоение по модулю
```

Для более сложного присваивания в JavaScript есть синтаксис **деструктуризации** - это выражение, которое позволяет извлекать данные из массивов или объектов, используя синтаксис, который зеркалирует конструкторы массивов и литералы объектов:

```
var google = ["analytics", "gtm", "ads"];

// без деструктуризации
var gAnalytics   = google[0];
var gManager   = google[1];
var gAds = google[2];

// с деструктуризацией
var [gAnalytics, gManager, gAds] = google;
```

Операторы сравнения

Операторы сравнения используются для сопоставления операндов, результатом выражения может быть одно из двух значений — **true (да, верно, истина)** или **false (нет, неверно, ложь)**. Операндами могут быть не только числа, но и строки, логические значения и объекты. Однако сравнение может выполняться только для чисел (number) и строк (string), поэтому операнды, не являющиеся числами или строками, преобразуются.

Исключением из данного правила является сравнение с использованием операторов **===** и **!==**, которые производят строгое сравнение на равенство или неравенство. Эти операторы не пытаются преобразовать операнды перед их сравнением. Строки сравниваются на основании стандартного лексикографического порядка, используя Unicode-значения.

Примечание: в JavaScript существует различие между одним (=), двумя (==) и тремя знаками равно (===). Один знак равенства означает *присваивание*, два знака равно - *сравнение на равенство*, допуская преобразование типов данных, а три – *сравнение на идентичность переменных* (строгое равенство), не допуская преобразования типов данных, то есть, когда типы сравниваемых значений являются одинаковыми (string-string, number-number).

Операторы сравнения:

Оператор / Операция	Описание
== Равенство	Проверяет две величины на совпадение, допуская преобразование типов. Возвращает true, если операнды совпадают, и false, если они различны
!= Неравенство	Возвращает true, если операнды не равны
=== Идентичность	Проверяет два операнда на идентичность, руководствуясь строгим определением совпадения. Возвращает true, если операнды равны без преобразования типов
!== Неидентичность	Выполняет проверку идентичности. Возвращает true, если операнды не равны без преобразования типов
> Больше	Возвращает true, если первый операнд больше второго, в противном случае возвращает false

>= Больше или равно	Возвращает true, если первый операнд не меньше второго, в противном случае возвращает false
< Меньше	Возвращает true, если первый операнд меньше второго, в противном случае возвращает false
<= Меньше или равно	Возвращает true, если первый операнд не больше второго, в противном случае возвращает false

Примеры операторов сравнения:

```
6 == "6"; // true
6 === "6"; // false
7 != -7.0; // true
false === false; // true
1 !== true; // true
1 != true; // false, так как true преобразуется в 1
9 > -9; // true
2 >= "3"; // false
```

Стандартные операции равенства с преобразованием типов (== и !=) используют **Абстрактный Алгоритм Эквивалентного Сравнения (The Abstract Equality Comparison Algorithm)** для сравнения двух операндов. Если у операндов различные типы, то JavaScript пытается привести их к одному типу, перед тем как сравнивать их. К примеру, в выражении **8 == '8'**, строка справа конвертируется в число, и только потом сравнивается.

Операторы строгого равенства (=== и !==) также используют **Строгий Алгоритм Эквивалентного Сравнения** (см. приложение), и предназначены для сравнения операндов одного типа. Если операнды имеют разные типы, то результат операции сравнения всегда будет ложью. К примеру, выражение **9 !== '9'** будет истинным.

Строки сравниваются по алфавиту, буквы в верхнем регистре всегда меньше букв в нижнем регистре. Сравнение строк основывается на номерах символов, указанных в стандарте Unicode, где прописные буквы идут раньше, чем строчные. Например:

```
"2" + "10"; // "210"
"2" + 10; // "210"
6 + 3 + " переменных"; // "9 переменных"
"переменных " + 6 + 3; // "переменных 63"
"1" > "10"; // false
"10" <= 10; // true
"GTM" == "gtm"; // false
```

Арифметические операторы

Арифметические операции принимают в качестве *операндов* числовые значения (это может быть и литерал и переменная) и возвращают результат в виде одного числового значения. Стандартными арифметическими операциями являются:

- сложение +
- вычитание -
- умножение *
- деление /

Операции в выражениях выполняются последовательно в соответствии со значением приоритета (чем больше значение приоритета, тем он выше). Возвращаемый результат не всегда имеет значение того же типа, что и тип обрабатываемых данных. Например, в операциях сравнения участвуют операнды различных типов, но возвращаемый результат всегда будет логического типа.

Арифметические операторы предназначены для выполнения математических операций, они работают с числовыми операндами (или переменными, хранящими числовые значения), возвращая в качестве результата числовое значение. Если один из операндов является строкой, интерпретатор JavaScript попытается преобразовать его в числовой тип, а после выполнить соответствующую операцию. Если преобразование типов окажется невозможным, будет получен результат **NaN (не число).**

Сложение (+)

Оператор **сложения (+)** возвращает сумму числовых операндов или объединяет строки:

```
// number + number -> сложение
5 + 6 // 11

// boolean + number -> сложение
true + 1 // 2

// boolean + boolean -> сложение
false + false // 0

// number + string -> конкатенация
7 + "gtm" // "7gtm"

// string + boolean -> конкатенация
"gtm" + false // "gtmfalse"

// string + string -> конкатенация
"google" + "tagmanager" // "googletagmanager"
```

В программировании (=JavaScript) существует такое понятие, как **конкатенация** – это процесс объединения. Конкатенация используется для соединения значений переменных друг с другом, чаще всего строк со строками (для образования более длинных строк). Оператор конкатенации в JavaScript такой же, как и оператор сложения (+).

Оператор сложение складывает числовые операнды. Если один из операндов — строка, то результатом выражения будет строка. Но если перед строкой идут два числа, то числа будут сложены перед преобразованием в строку, поскольку операции выполняются слева направо:

```
5 + 5 + '1' ; // будет "101", а не "551"
"google" + 20 + 20 // "google2020"
20 + 20 + "google" // "40google"
```

Сложение и преобразование строк – особенность бинарного плюса **+**. Другие арифметические операторы работают только с числами и всегда преобразуют операнды в числа.

Вычитание (-)

Оператор **вычитания** выполняет вычитание второго операнда из первого и возвращает разницу.

```
18 - 3 // 15
20 - 42 // -22
"gtm" - 2 // NaN
"gtm" - "tm" // NaN
```

Деление (/)

Оператор **деления** делит первый операнд (левый, делимый) на второй (правый, делитель). Результатом деления может являться как целое, так и число с плавающей точкой.

```
18 / 3 // 6
5 / 3 // 1.6666666666666667
1 / 2 // 0.5
1.0 / 2.0 // 0.5
2.0 / 0 // Infinity
3.5 / -0 // -Infinity
```

Умножение (*)

Оператор **умножения** умножает два операнда и возвращает их произведение.

```
5 * 5 // 25
-2 * 4 // -8
Infinity * 0 // NaN
Infinity * Infinity // Infinity
"gtm" * 25 // NaN
```

Остаток от деления, деление по модулю (%)

Оператор вычисляет остаток, получаемый при целочисленном делении первого операнда (левого) на второй (правый). Возвращаемое значение всегда получает знак делимого, а не делителя. Применяется как к целым числам, так и числам с плавающей точкой.

```
11 % 5 // 1
-5 % 2 // -1
NaN % 3 // NaN
1 % 5 // 1
2 % 6 // 2
-2 % 2 // -0
2 % -2 // 0
5.5 % 2 // 1.5
```

Возведение в степень (**)

Оператор **возведения в степень** имеет два операнда. Первый операнд является основанием степени, второй операнд – показателем степени. В результате оператор возвращает основание, возведенное в указанную степень:

```
3 ** 2 // 9
3 ** 2.5 // 15.588457268119896
10 ** -1 // 0.1
NaN ** 2 // NaN
2 ** 3 ** 2 // 512
2 ** (3 ** 2) // 512
(2 ** 3) ** 2 // 64
-(2 ** 2) // -4
(-2) ** 2 // 4
```

Унарный минус (-)

Унарный минус ставится перед своим операндом и возвращает его математическое отрицание, то есть преобразует положительное число в отрицательное, и наоборот.

```
var x = 3;
y = -x; // y = -3, x = 3
```

Унарный плюс (+)

Оператор **унарный плюс** предшествует своему операнду и оценивает его, пытается преобразовать его в число. Является быстрейшим и предпочитаемым способом конвертирования чего-либо в **число (number)** потому, что он не выполняет каких-либо операций с числом. Он может конвертировать строковые представления целых и чисел с плавающей точкой, а также нестроковые значения **true, false** и **null.** Если он не может вычислить конкретное значение, то выполнится как **NaN**.

При использовании с числовым операндом он не выполняет никаких действий:

```
+4 // 4
+"5" // 5
+true // 1
```

```
+false // 0
+null // 0
```

Инкремент (++)

Оператор **++ (инкремент)** увеличивает значение своего операнда на единицу. Если значение операнда не является **числом (number)**, оператор автоматически преобразует его в число, увеличивает на единицу и возвращает результат, который присваивается обратно операнду.

Инкремент имеет две формы:

1. **постфиксную** (оператор ставится после операнда, x++);
2. **префиксную** (оператор ставится перед операндом, +xx).

Если он используется в **постфиксной форме**, то сначала возвращается исходное значение операнда, и только затем значение операнда увеличивается на единицу:

```
var x = 5;
x++;  // 5
x;     // 6
```

В **префиксной форме** оператор инкремент сразу увеличивает значение своего операнда на единицу:

```
var x = 5;
++x;  // 6
```

Декремент (--)

Оператор **-- (декремент)** работает аналогично оператору инкремент, но не увеличивает значение своего операнда, а наоборот, уменьшает его на единицу, и также имеет две формы: **постфиксную (x--)** и **префиксную (--x)**.

```
var y = 5;
y--; // 5
y; // 4

var z = 8;
--z; // 7
```

Оператор typeof ()

Оператор **typeof()** позволяет узнать к какому типу данных относится переменная. Имеет две формы написания: **typeof x** или **typeof(x)**. Возвращает строку с именем типа. Например:

```
var n = 5;
typeof(n) // "number"

var hi = "привет";
typeof hi // "string"

var big = 10n;
typeof big // "bigint"

var analytics = {
     owner: "Google",
     commercial: "Yes",
     url: "analytics.google.com",
     launched: "2005"
};
typeof(analytics) // "object"
```

```
var m = true;
typeof m // "boolean"

var id = Symbol();
typeof id // "symbol"
```

Во всех случаях оператор **typeof** будет возвращать верное значение из списка кроме **null**. В случае с **null** оператор **typeof** вернет **object**. Это неверно. Это официально признанная ошибка в **typeof**, сохраненная для совместимости. **null** не является объектом. Это специальное значение с отдельным типом.

Побитовые операторы

Побитовые операторы работают с операндами как с 32-битной последовательностью нулей и единиц и возвращают числовое значение, означающее результат операции, записанное в десятичной системе счисления. В качестве операндов рассматриваются целые числа, дробная часть операнда отбрасывается. Побитовые операции могут использоваться, например, при шифровании данных, для работы с флагами, разграничения прав доступа.

Поддерживаются следующие операторы:

Оператор / Операция	Использование	Описание
Побитовое И	a & b	Возвращает единицу в каждой битовой позиции, для которой соответствующие биты обеих операндов являются единицами
Побитовое ИЛИ	a | b	Возвращает единицу в каждой битовой позиции, для которой один из соответствующих битов или оба бита операндов являются единицами
Исключающее ИЛИ	a ^ b	Возвращает единицу в каждой битовой позиции, для которой только один из соответствующих битов операндов является единицей
Побитовое НЕ	~ a	Заменяет биты операнда на противоположные
Сдвиг влево	a << b	Сдвигает a в двоичном представлении на b бит влево, добавляя справа нули
Сдвиг вправо с переносом знака	a >> b	Сдвигает a в двоичном представлении на b бит вправо, отбрасывая сдвигаемые биты
Сдвиг вправо с заполнением нулями	a >>> b	Сдвигает a в двоичном представлении на b бит вправо, отбрасывая сдвигаемые биты и добавляя слева нули

Побитовые операторы встроены в изолированную версию JavaScript, которая используется в Google Tag Manager. Однако на практике они встречаются крайне редко. Подробнее о побитовых операторах можно прочитать на **developer.mozilla.org** (см. приложение).

Логические операторы

Логические операторы используются с булевыми (логическими) значениями. Возвращаемое ими значение также является булевым. Однако операторы **&&** и **||** фактически возвращают значение одного из операндов, поэтому, если эти операторы используются с небулевыми величинами, то возвращаемая ими величина также может быть не булевой. Чаще всего используются в условном выражении **if.**

Поддерживаются следующие операторы:

Оператор / Операция	Использование	Описание
Логическое И	&&	Возвращает true, только если оба операнда истинны. При выполнении операции сначала проверяется значение первого операнда. Если оно имеет значение false, то значение второго оператора не учитывается и результату выражения присваивается false
Логическое ИЛИ	||	Возвращает true, если хотя бы один операнд истинен, т.е. проверяет истинность как минимум одного условия
Логическое НЕ (инверсия)	!	Изменяет значение оператора на обратное - с true на false, и наоборот

Примеры логических операторов:

```
var a = true, b = false;
```

```
!a; // false
!b; // true
(2 < 3) && (3 === 3); // true
false && (3 == 4); // false
true && true; // true
n = 1 || 5; // 1, небулевая величина
m = 0 || 5; // 5, небулевая величина
```

Примерами выражений, которые могут быть преобразованы в **false** являются: **null, 0, NaN, пустая строка ("")** или **undefined**.

Строковые операторы

В дополнение к операторам сравнения, которые могут использоваться со строковыми значениями, **оператор +** позволяет объединить две строки, возвращая при этом третью строку, которая представляет собой объединение двух строк-операндов (конкатенация). Сокращенный оператор присваивания **+=** также может быть использован для объединения строк:

```
var a = "google", b = " tag", c = " manager";
a + b + c // "google tag manager"

var a = "google";
a += " tag manager"; // "google tag manager", и присваивается a
```

Специальные операторы

В JavaScript существует ряд специальных операторов, которые выполняют различные функции:

Оператор / Операция	Использование	Описание
Обращение к свойству	.	Осуществляет доступ к свойству объекта
Множественное вычисление (запятая)	,	Вычисляет несколько независимых выражений, записанных в одну строку. Чаще всего используется в цикле for
Индексация массива	[]	Осуществляет доступ к элементам массива или свойствам объекта
Вызов функции, группировка	()	Группирует операции или вызывает функцию
Определение типа данных	typeof	Унарный оператор, возвращает тип данных операнда
Проверка типа объекта	instanceof	Оператор проверяет, является ли объект экземпляром определенного класса. Левый операнд должен быть объектом, правый - должен содержать имя класса объектов. Результат будет true, если объект, указанный слева, представляет собой экземпляр класса, указанного справа, в противном случае - false
Проверка наличия свойства (оператор отношения)	in	В качестве левого операнда должна быть строка, а правым - массив или объект. Если левое значение является свойством объекта, вернется результат true.
Создание объекта	new	Оператор создает новый объект с неопределенными свойствами, затем вызывает функцию-конструктор для его инициализации (передачи параметров). Также может применяться для создания массива.
Удаление (унарный оператор)	delete	Оператор позволяет удалять свойство из объекта или элемент из массива. Возвращает true, если удаление прошло успешно, в противном случае false. При удалении элемента массива его длина не меняется.
Определение выражения без возвращаемого значения	void	Унарный оператор, отбрасывает значение операнда и возвращает undefined
Условный (тернарный) оператор	?:	Позволяет организовать простое ветвление. В выражении участвуют три операнда, первый должен быть логическим значением или преобразовываться в него, а второй и третий - любыми значениями. Если первый операнд равен true, то условное выражение примет значение второго операнда; если false - то третьего

Приоритет операторов

Приоритет операторов определяет порядок, в котором операторы выполняются. Операторы с более высоким приоритетом выполняются первыми. Приведенная ниже таблица описывает приоритет операторов от наивысшего до низшего:

Тип оператора	Операторы
Свойство объекта	. []
Вызов, создание экземпляра объекта	() new
Отрицание, Инкремент	! ~ - + ++ -- typeof void delete
Умножение, Деление	* / %
Сложение, Вычитание	+ -
Побитовый сдвиг	<< >> >>>
Сравнение, Вхождение	< <= > >= in instanceof
Равенство	== != === !==
Побитовое И	&
Исключающее ИЛИ	^
Побитовое ИЛИ	|
Логическое И	&&
Логическое ИЛИ	||
Условный (тернарный) оператор	?:
Присваивание	= += -= *= /= %= <<= >>= >>>= &= ^= |=
Запятая	,

Более подробная версия данной таблицы, содержащая ссылки и дополнительную информацию по каждому оператору, находится в справочнике JavaScript (см. приложение).

Преобразование типов данных

В JavaScript значения переменных могут легко быть преобразованы из одного типа в другой. Например, если какой-нибудь оператор ожидает получить значение определенного типа, а ему передается значение другого типа, то интерпретатор автоматически попытается выполнить преобразования к нужному типу:

```
15 + " переменных" // "15 переменных". Число неявно преобразуется в строку
'6' * '5' // 30. Строки неявно преобразуются в числа
```

Встречаются **явное** и **неявное** преобразования.

Неявное преобразование – это когда интерпретатор автоматически выполняет преобразование типов, то есть без участия программиста.

Явное преобразование – это когда преобразование выполняет сам программист. Явное преобразование еще называют *приведением типов*.

```
'6' * '5' // 30. Неявное преобразование
Number('6') * Number('5') // 30. Явное преобразование
```

Для явного преобразования в простые типы используются функции **Boolean()**, **Number()**, **String()**. При неявном преобразования интерпретатор использует те же функции, что используются для явного преобразования.

Преобразование типов для примитивов

Разберем преобразование типов для примитивов **number, string, boolean, null, undefined** и **symbol**.

В JavaScript существует 3 типа преобразования:

1. строковое преобразование (String);
2. численное преобразование (Number);
3. булевое преобразование, к логическому типу (Boolean).

Строковое преобразование

Для явного приведения значения к строке необходимо применить к нему функцию **String()**:

```
String(521) // "521"
String(-15.5) // "15.5"
String(null) // "null"
String(undefined) // "undefined"
String(true) // "true"
String(false) // "false"
```

Неявное преобразование будет вызвано бинарным оператором **+**, когда один из операндов является строкой. В этом случае + приводит к строке и другой аргумент:

```
123 + '' // "123"
"google tag manager " + 2020 // "google tag manager 2020"
```

Численное преобразование

Для явного преобразования к числу нужно применить функцию **Number()**:

```
Number(null) // 0
Number(undefined) // NaN
Number(true) // 1
Number(false) // 0
Number(" 18 ") // 18
Number("-16.78") // -16.78
Number("\n") // 0
Number(" 12x ") // NaN
Number(555) // 555
```

Неявное приведение значения к числовому типу применяется чаще, чем преобразование в строку или в логическое значение. Преобразование выполняют следующие операторы:

- Операторы сравнения >, <, <=, >=
- Побитовые операторы |, &, ^, ~
- Арифметические операторы -, +, *, /, % . Оператор + с двумя операндами не вызывает неявное преобразование к числовому типу, если хотя бы один оператор является строкой
- Унарный оператор +
- Оператор нестрогого равенства == и неравенства != . Оператор == не производит неявного преобразования в число, если оба операнда являются строками

```
Number('678') // явное преобразование
+'678' // неявное преобразование
123 != '456' // неявное преобразование
4 > '5' // неявное преобразование
5/null // неявное преобразование
true | 0 // неявное преобразование
```

Преобразование значений с помощью функции **Number()** осуществляется по следующим правилам:

- числа возвращаются без каких-либо изменений;
- **true** преобразуется в **1**;
- **false** преобразуется в **0**;
- значение **null** преобразуется в **0**;
- значение **undefined** преобразуется в **NaN**.

Преобразование строк также осуществляется по определенным правилам:

- при преобразовании строк в числа интерпретатор обрезает пробелы, а также управляющие символы (перенос строки **\n**, табуляция **\t**, вертикальная табуляция **\v** и др.), которые находятся в начале или в конце строки;
- если строка содержит только цифры с унарным оператором **-** или **+** либо без знака, она всегда преобразуется в целое десятичное число (отрицательное или положительное). Начальные нули игнорируются, например **"007"** преобразуется в **7**;
- если строка представляет собой число в шестнадцатеричном формате, она преобразуется в соответствующее целое десятичное число;
- пустая строка преобразуется в **0**;
- если строка содержит что-то отличное от предыдущих вариантов, например инкремент (декремент), она преобразуется в **NaN**.

Булевое преобразование

Для явного преобразования к булевому значению применяется функция **Boolean()**:

```
Boolean(1); // true
Boolean(0); // false
Boolean("GTM"); // true
Boolean(""); // false
Boolean(-0) // false
Boolean(NaN) // false
Boolean(null) // false
Boolean(undefined) // false
Boolean(false) // false
```

Любое значение, которое не вошло в этот список, будет преобразовано в **true**, включая объекты, функции, **Array, Date** и так далее. Символы, пустые объекты и массивы так же будут иметь значение **true**:

```
Boolean({}) // true
Boolean([]) // true
Boolean(Symbol()) // true
!!Symbol() // true
Boolean(function() {}) // true
```

Неявное преобразование происходит в логическом контексте **if (val) { ... }**, или вызывается логическими операторами **|| && !**

```
!!"0" // true
!!5 // true
!!" " // true
```

Переменные в Google Tag Manager

Переменная в Google Tag Manager — это место хранения необходимой информации, которой дано имя, и которая может принимать различные значения (вида пара ключ:значение). Имя переменной постоянно, а ее значение меняется в зависимости от того, какие действия происходят на сайте. Устаревшее название переменной в предыдущей версии диспетчера тегов Google – **макрос**.

Переменные в GTM практически ничем не отличаются от переменных в JavaScript. У каждого события на сайте есть свои переменные, в которых находится информация об этом событии. Например, когда в интернете-магазине мы нажмите на кнопку *Добавить в корзину*, в переменных этого события будут лежать значения CSS-класса этого элемента или сам текст элемента (кнопки). При клике на ссылку в переменной будет передаваться еще и URL-адрес.

Переменные могут применяться как в триггерах, так и в тегах. В триггерах они используются в качестве фильтра активации, то есть когда следует запускать тот или иной тег.

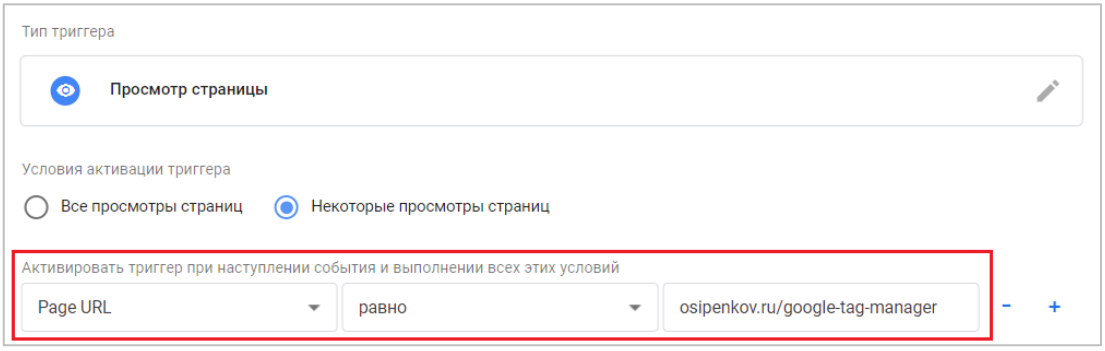


Рис. 239. Переменные в триггерах

Например, запускать триггер **Просмотр страницы** только тогда, когда переменная **Page URL** равна значению *osipenkov.ru/google-tag-manager*. Или выбрав тип триггера Клик – Все элементы, вы можете с помощью условий (переменная + фильтр + значение) активировать триггер по нажатию на определенные элементы, и т.д.

Переменные в тегах используются для получения динамических значений. Например, в Google Analytics о сумме транзакции, ID заказа, идентификаторе продукта и т.д.

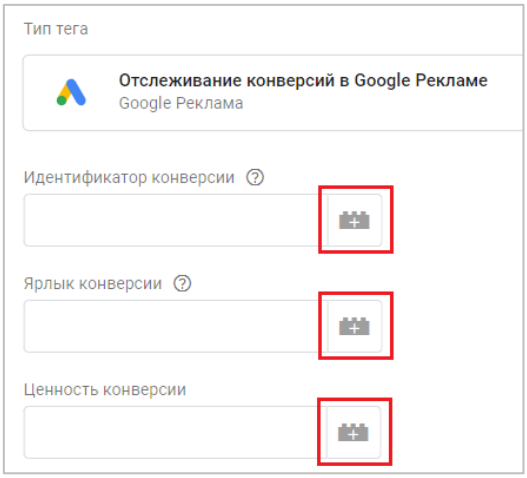


Рис. 240. Поле шаблона Переменная

В примере выше с помощью иконки (поле шаблона) **Переменная** для типа тега **Отслеживание конверсий в Google Рекламе** задаются значения для различных полей – идентификатор отслеживания, ярлык конверсии, ценность конверсии и т.д.

Также с переменными можно работать и в пользовательских HTML-тегах. Для этого нам необходимо использовать специальную конструкцию, заключенную в теги *<script></script>*. Тег **<script>** предназначен для описания скриптов и является контейнером для сценария, выполняющегося на стороне клиента, то есть в браузере пользователя. Как правило, тегом <script> подключают JavaScript.

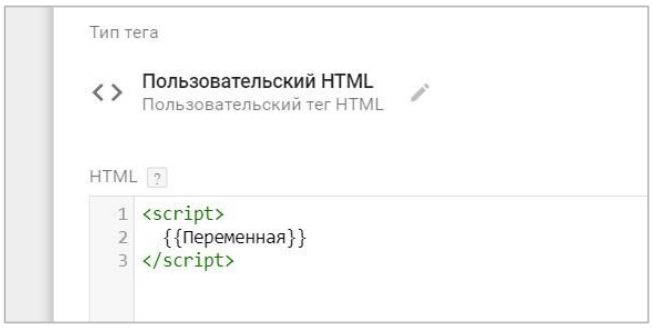


Рис. 241. Пользовательский HTML тег с переменной

Данный код работать не будет, поскольку он является лишь демонстрацией того, как в пользовательских HTML-тегах можно обращаться к переменным.

В GTM вызов функции переменной выполняется с помощью специального синтаксиса:

```
{{имя переменной}}
```

Переменные шаблона в Google Tag Manager, как и выражения, заключаются в двойные фигурные скобки **{{ }}** с обеих сторон. Данные в шаблон передаются в виде обычного объекта, переменным шаблона соответствуют свойства этого объекта.

С такой конструкцией мы с вами знакомились несколько ранее, когда устанавливали счетчики Google Analytics и Яндекс.Метрики через GTM. Там в качестве переменной выступал идентификатор отслеживания Google Analytics. Чтобы не вводить ID вручную при создании каждого тега, переменная для Universal Analytics создавалась как пользовательская. В дальнейшем, если идентификатор отслеживания GA изменится, достаточно в одном месте указать новый ID счетчика.

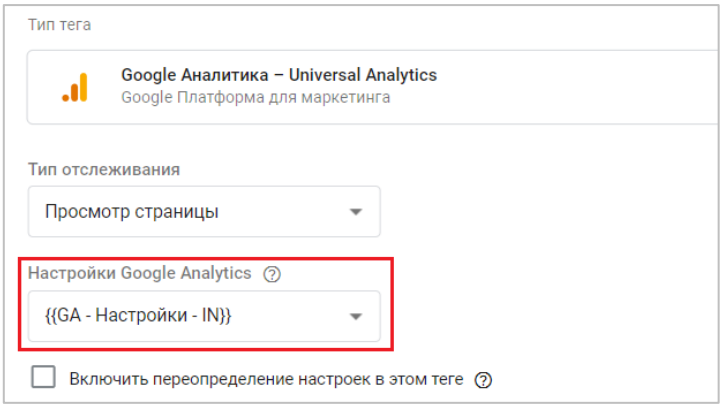


Рис. 242. Пользовательская переменная с ID счетчиком Google Analytics

Примечание: как только вы начнете использовать шаблон переменной и напишите {{, Google Tag Manager предложит вам выбрать вариант из доступных переменных.

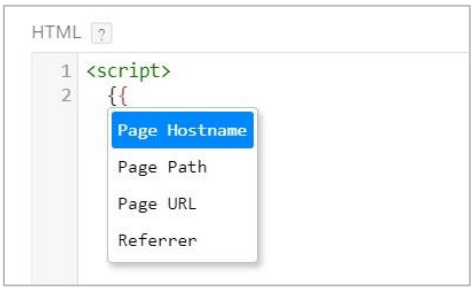


Рис. 243. Подсказка при написании

Любопытный момент: при изменении имени переменной все ссылки на переменную автоматически обновляются. Даже те, которые вы ввели вручную в пользовательских HTML-тегах и пользовательских переменных JavaScript. Это сильно экономит время, поскольку не нужно проходить каждую ссылку и переписывать код в соответствии с новым именем переменной.

Источником информации могут быть:

- уровень данных;
- переменные JavaScript;
- файлы cookie сайта;
- Document Object Model (DOM).

В Google Tag Manager переменные делятся на две категории:

1. **встроенные переменные** – набор готовых, предустановленных в системе переменных;
2. **пользовательские переменные** – самостоятельное создание с использованием существующих типов переменных.

Встроенные переменные

Встроенные переменные создаются автоматически и их нельзя изменить. Часть из них в GTM не активна по умолчанию.

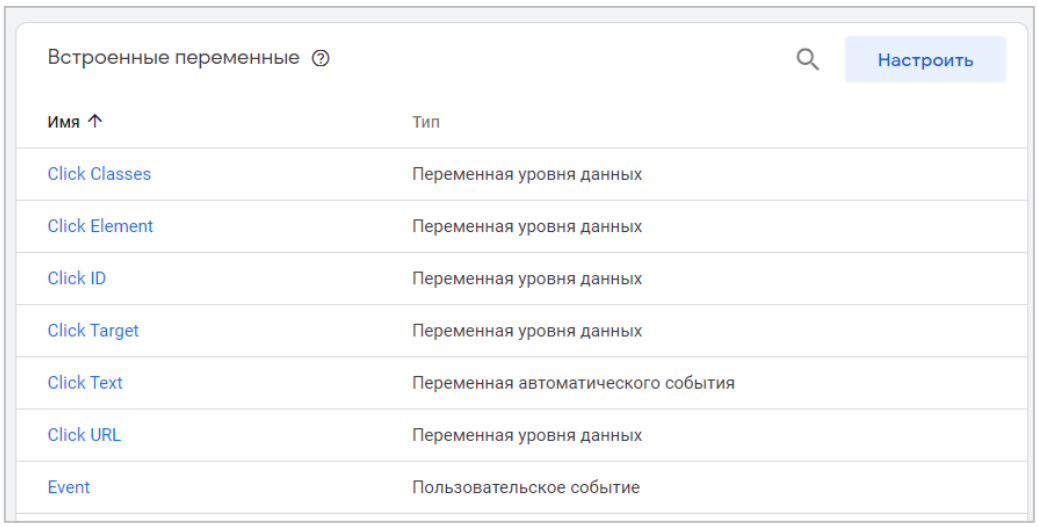


Рис. 244. Встроенные переменные

Они включаются и отключаются путем простановки галочек (чекбоксов) напротив каждой из них.

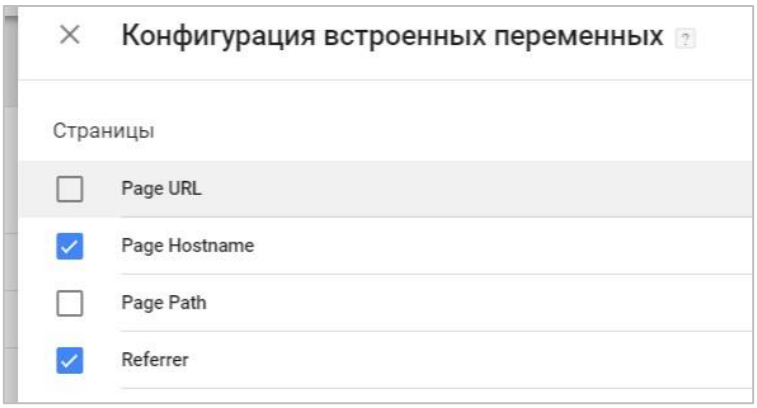


Рис. 245. Включение/Отключение встроенных переменных

Использование встроенных переменных не требует глубоких знаний в программировании и иногда позволяет заменить большое количество строк кода всего парой кликов мыши.

В Google Tag Manager на данный момент существует 9 категорий встроенных переменных:

- Страницы
- Утилиты
- Ошибки
- Клики
- Формы
- История
- Видео
- Прокрутка
- Видимость

Страницы

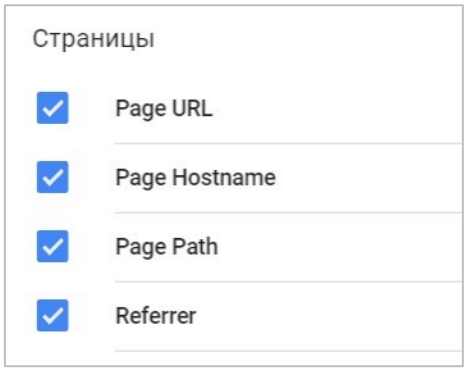


Рис. 246. Встроенные переменные типа Страницы

Наиболее часто используемые переменные. Они позволяют определить, где пользователь находится в данный момент относительно URL-структуры сайта. Страница может быть как текущая, на которой в данный момент находится посетитель сайта, так и та страница, с которой он перешел на текущую страницу (Referrer).

Page URL - переменная возвращает полный URL текущей страницы, но без хэша # (все, что после решетки). Например, если пользователь загрузил страницу *https://osipenkov.ru/analytics?parameter=true#vars*, то переменная вернет значение *https://osipenkov.ru/analytics?parameter=true*

Page Hostname - переменная возвращает имя хоста (доменное имя) в URL текущей страницы. Например, для страницы *https://osipenkov.ru/analytics?parameter=true#vars* данная переменная вернет значение osipenkov.ru.

Page Path - переменная возвращает путь к странице в текущем URL без учета GET-параметров, то есть всего того, что идет в URL после знака вопроса **?** и **&** между параметрами.

Например, на странице *https://osipenkov.ru/analytics/?parameter=true#vars* в GTM данная переменная вернет значение */analytics/*.

Referrer - полный URL перехода к текущей странице, то есть путь (ссылка) откуда пришел пользователь на текущую страницу.

Например, если пользователь со страницы *https://osipenkov.ru/google-analytics-book/* перешел на страницу *https://osipenkov.ru/google-tag-manager/*, то значение данной переменной будет *https://osipenkov.ru/google-analytics-book/*

Первые три переменные предназначены для текущей страницы, а последняя Referrer необходима для работы с той URL-страницы, с которой пользователь перешел на наш сайт. По умолчанию в Google Tag Manager все 4 встроенные переменные из категории **Страницы** активированы.

В качестве примера разберем простой переход из поисковой системы Google на сайт graphanalytics.ru. В режиме отладки GTM это будет выглядеть так:

Page Hostname	URL	string	'graphanalytics.ru'
Page Path	URL	string	'/'
Page URL	URL	string	'https://graphanalytics.ru/'
Referrer	Источник ссылки HTTP	string	'https://www.google.ru/'

Рис. 247. Пример встроенных переменных типа Страницы

Переменная **Page Hostname** вернула имя домена *graphanalytics.ru*, **Page Path** содержит путь к странице в текущем URL, что соответствует / (главная директория), **Page URL** – полный URL текущей страницы

https://graphanalytics.ru/, а переход на сайт был выполнен из поиска Google, поэтому переменная **Referrer** вернула значение *https://www.google.ru/*.

Утилиты

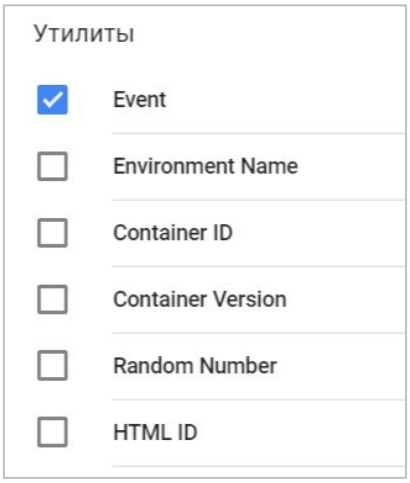


Рис. 248. Встроенные переменные типа Утилиты

Набор из 6 встроенных переменных в GTM, которые в большей степени носят служебные функции.

Event – возвращает подстроку, которая содержит тип события, произошедшее на сайте. Например, при нажатии на любой элемент возвращает **gtm.click**, на ссылку - возвращает **gtm.linkClick**, при заполнении формы - **gtm.formSubmit**, при возникновении ошибки - **gtm.pageError**.

Environment Name – возвращает название текущей среды (Реальная, Последняя или Редактирование), если запрос контейнера выполнен из функции Поделиться ссылкой для просмотра или из фрагмента кода среды.

Container ID - возвращает номер контейнера GTM. Например, GTM-NC2LK3M.

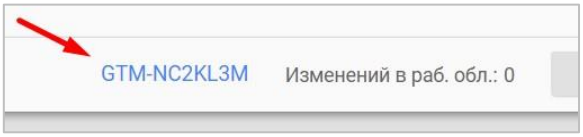


Рис. 249. Container ID

Container Version – возвращает номер версии контейнера в виде строкового значения. Например, 5.

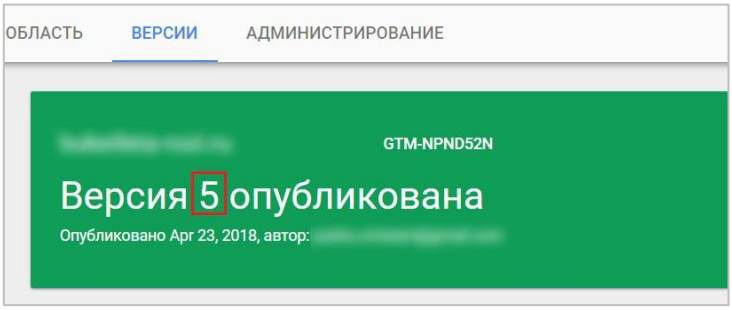


Рис. 250. Container Version

Random Number - возвращает случайное число от 0 до 2147483647. Например, 666.

HTML ID – возвращает идентификатор пользовательского HTML-тега. Используется с секвенированием (порядком активации) тегов.

Если мы с вами перейдем в отладчик GTM, то увидим такие значения переменных из категории **Утилиты**:

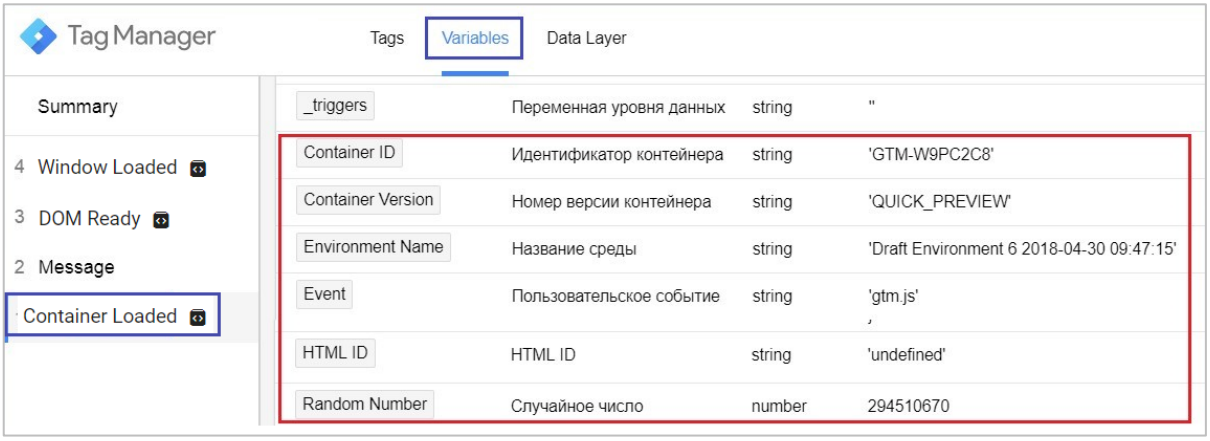


Рис. 251. Пример встроенных переменных типа Утилиты

В качестве события был выбран Page View (Просмотр страницы):

- **Container ID** – GTM-W9PC2C8 (идентификатор контейнера);
- **Container Version** – QUICK_PREVIEW (режим предварительного просмотра);
- **Environment Name** - Draft Environment 6 2018-04-30 09:47:15 (название среды);
- **Event** – gtm.js (событие просмотра страницы);
- **HTML ID** - значение не присвоено. Значение ID переменной HTML отображается только в том случае, если мы работаем с тегами типа пользовательский HTML;
- **Random Number** – 294510670 (генератор случайных чисел). При обновлении страницы и фиксации нового события данное число изменится.

Ошибки

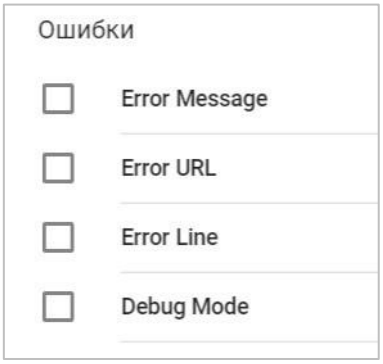


Рис. 252. Встроенные переменные типа Ошибки

Переменные из этой категории позволяют разработчикам анализировать ошибки, происходящие на сайте (анализ страниц, типы устройств, версий браузеров, разрешений экранов и т.д.). Для использования этих переменных необходимо их активировать и создать хотя бы один триггер типа **Ошибка JavaScript**.

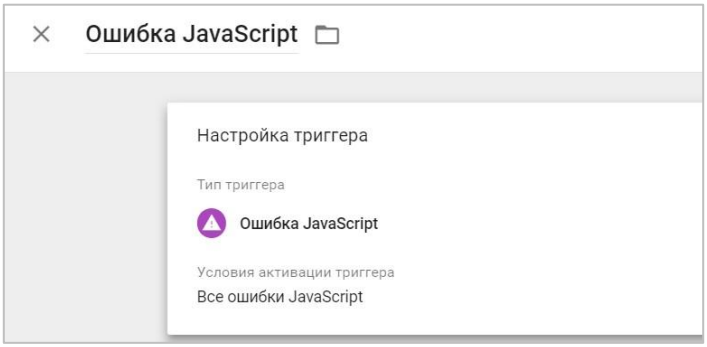


Рис. 253. Тип триггера – Ошибка JavaScript

Error Message - возвращает строку, содержащую сообщение об ошибке, отправленное с помощью триггера Ошибка JavaScript.

Error URL - возвращает строку, содержащую URL-адрес скрипта, в котором была обнаружена ошибка.

Error Line - строка файла, в которой произошла ошибка.

Debug Mode – возвращает значение true, если активирован режим отладки в Google Tag Manager.

В качестве наглядного примера создайте вынужденную ошибку в коде JS. Но перед тем, как запускать отладчик Google Tag Manager, не забываем создать новый триггер **Ошибка JavaScript** и обновить режим предварительного просмотра. Только после этого изменения вступят в силу.

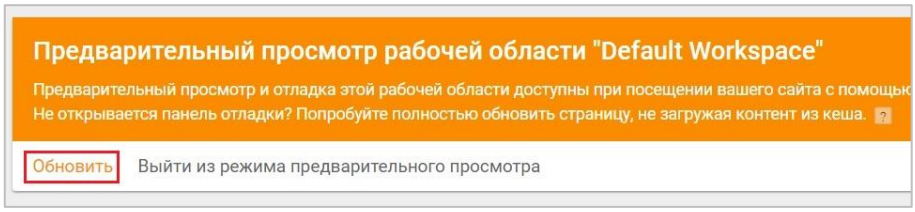


Рис. 254. Обновление режима предварительного просмотра

В отладчике GTM будет доступна следующая информация:

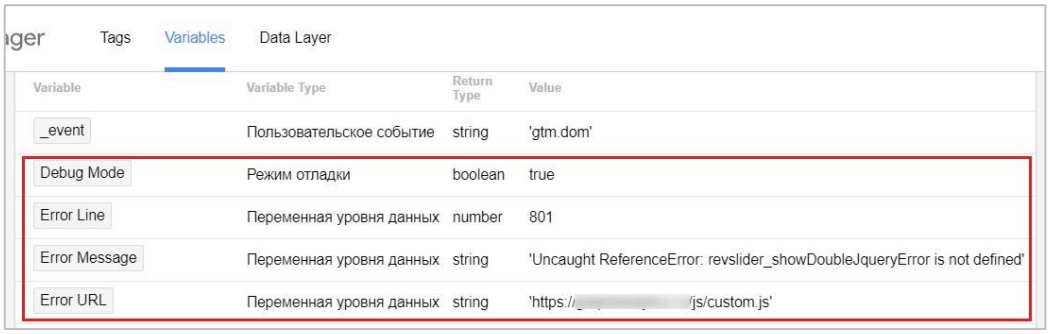


Рис. 255. Пример встроенных переменных типа Ошибки

- **Debug Mode** – true, так как у нас активирован режим отладки;
- **Error Line** – 801 строка;
- **Error Message** – сообщение об ошибке;
- **Error URL** – URL-адрес скрипта, в котором обнаружена ошибка.

В случае, если мы в JavaScript используем конструкцию **try..catch**, то Google Tag Manager не зафиксирует ошибку. Конструкция выглядит следующим образом:

```
try {
// код …
} catch (err) {
// обработка ошибки
}
```

Работает она так:

1. Выполняется код внутри блока **try**.
2. Если в нем ошибок нет, то блок **catch(err)** игнорируется, то есть выполнение доходит до конца **try** и потом прыгает через **catch**.
3. Если в нем возникнет ошибка, то выполнение **try** на ней прерывается, и управление прыгает в начало блока **catch(err)**.

При этом переменная **err** (можно выбрать и другое название) будет содержать объект ошибки с подробной информацией о произошедшем. Таким образом, при ошибке в **try** скрипт не падает, и мы получаем возможность обработать ошибку внутри **catch**.

Подробнее про **try..catch** читайте в справочнике JavaScript (см. приложение).

Клики

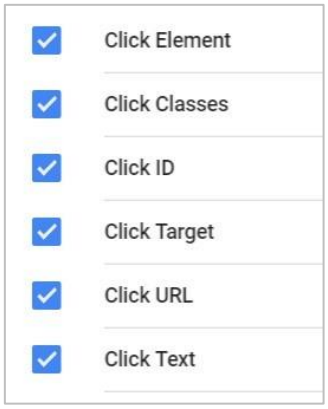


Рис. 256. Встроенные переменные типа Клики

Переменные в этой категории используются для отслеживания кликов по элементам сайта. Разделить их можно на две составляющие:

- переменные, которые возвращают значения при клике по всем элементам (**gtm.click**);
- переменные, которые возвращают значения при клике только по ссылкам (**gtm.linkClick**).

Click Element – возвращает значение для того HTML-элемента, на который кликнул пользователь и который был объектом действия пользовательского события (Event - gtm.click или gtm.linkClick). Этот объект извлекается из ключа gtm.element в dataLayer.

Click Classes – возвращает строку, содержащуюся в ключе gtm.elementClasses значении атрибута className пользовательского события по которому был выполнен клик. Аналог {Click Element}}.className

Click ID – возвращает строку, содержащуюся в ключе gtm.elementId значении атрибута ID пользовательского события по которому был выполнен клик. Аналог свойства {{Click Element}}.id

Click Target - возвращает строку, содержащуюся в ключе gtm.elementTarget значении атрибута target пользовательского события по которому был выполнен клик. Аналог свойства {{Click Element}}.target

Click URL - возвращает строку, содержащуюся в ключе gtm.elementUrl значении атрибута href пользовательского события по которому был выполнен клик. Аналог свойства {{Click Element}}.href для ссылок, и {{Click Element}}.action для форм.

Click Text - возвращает строку, содержащуюся в ключе gtm.elementText значении атрибута textContent / innerText пользовательского события по которому был выполнен клик. Аналог свойства {{Click Element}}.innerText

Для прослушивания кликов в Google Tag Manager используются встроенные триггеры из раздела **Клики**.

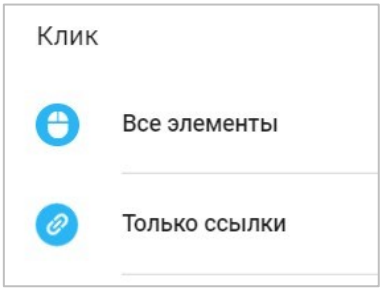


Рис. 257. Клики - Все элементы или Только ссылки

При активации триггера, запуске отладчика в Google Tag Manager и клике по любому элементу своего сайта, вы можете наблюдать такую картину (на вкладке **Variables**):

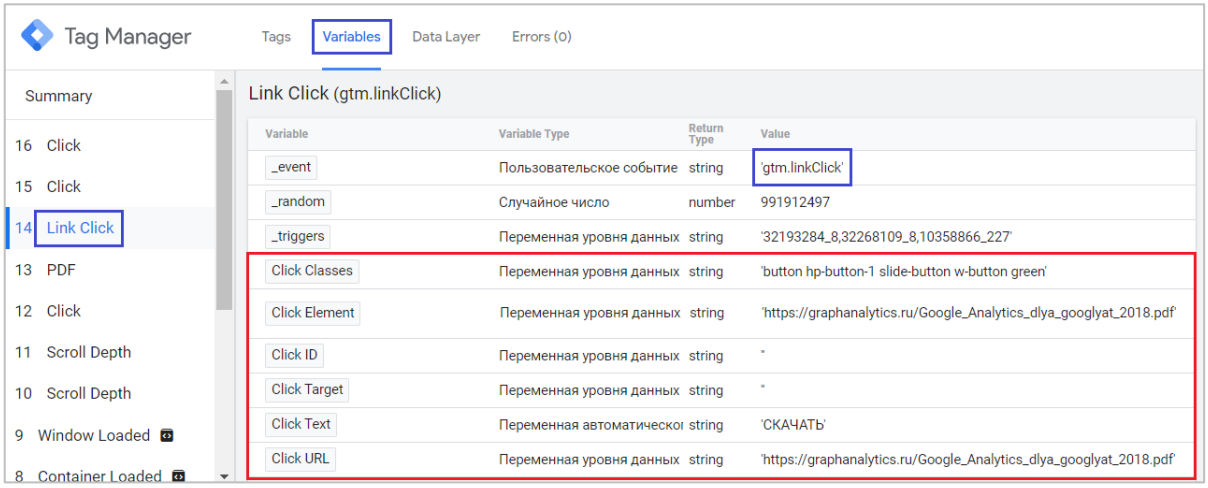


Рис. 258. Пример встроенных переменных типа Клики

Данный клик был осуществлен по кнопке СКАЧАТЬ, которая имеет ссылку для скачивания электронной книги на сайте graphanalytics.ru.

Рис. 259. Клик по кнопке Скачать

Открыв консоль разработчика (в Google Chrome F12) и подсветив нужный фрагмент кода, мы увидим, что у данного элемента присутствуют всего два атрибута – *href* и *class*.

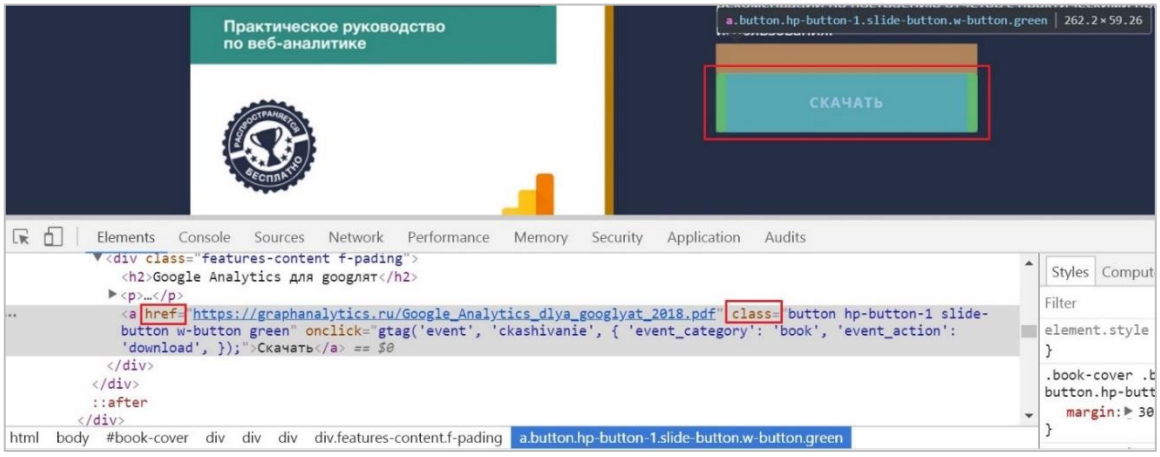


Рис. 260. Атрибуты href и class

Переменные в данном конкретном примере вернули значения:

- **_event (пользовательское событие)** – gtm.linkClick, так как клик был осуществлен по ссылке;
- **Click Element** – https://graphanalytics.ru/Google_Analytics_dlya_googlyat_2018.pdf
- **Click Classes** – button hp-button-1 slide-button w-button green (класс кнопки)

- **Click ID** – пустое значение, нет данных по атрибуту id у этого элемента
- **Click Target** – пустое значение, нет данных по атрибуту target у этого элемента
- **Click URL** - https://graphanalytics.ru/Google_Analytics_dlya_googlyat_2018.pdf
- **Click Text** - СКАЧАТЬ

На вкладке Data Layer можно увидеть, как переменные получают доступ к уровню данных и считывают ключи, которые задаются триггерами **Все элементы** и **Только ссылки**.

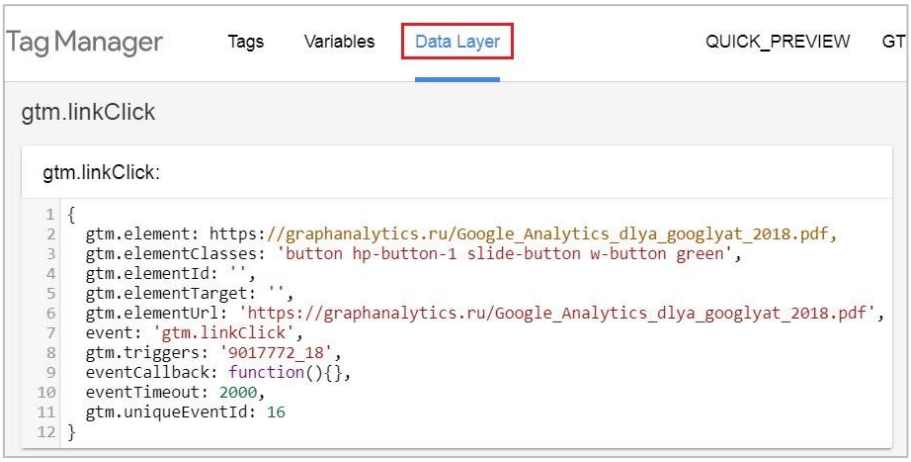


Рис. 261. Вкладка Data Layer в режиме отладки

Аналогичным образом можно посмотреть значения передаваемых переменных по пользовательскому событию gtm.click.

Формы

Схожа с категорией **Клики**, только в качестве отслеживаемых объектов используются формы.

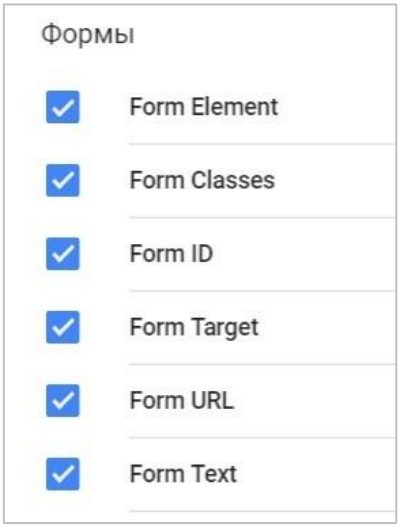


Рис. 262. Встроенные переменные типа Формы

Те же самые переменные, только вместо Click -> Form, а в качестве триггера прослушивания используется **Отправка формы**, а пользовательское событие называется **gtm.formSubmit**.

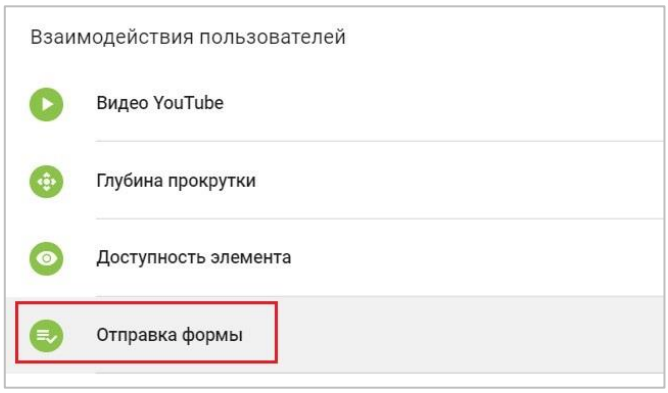


Рис. 263. Тип триггера – Отправка формы

Form Element — возвращает JS-объект для формы, которая была отправлена пользователем. Например, атрибуты class, ID, данные о родительских и дочерних элементах. Этот объект также извлекается из ключа gtm.element в dataLayer.

Form Classes — возвращает набор атрибутов class для отправленной формы. Содержится в ключе gtm.elementClasses.

Form ID — возвращает значение атрибута id для отправленной формы. Содержится в ключе gtm.elementId.

Form Target — возвращает значение атрибута target. Содержится в ключе gtm.elementTarget

Form URL — возвращает значение атрибута action для отправленной формы. Содержится в ключе gtm.elementUrl.

Данная переменная полезна, когда формам не присвоены атрибуты id и class, и они отсылаются разным обработчикам на сервере. Например, форма обратной связи может отсылать данные на /contacts.php, а форма новостной подписки на /subscribe.php.

Form Text — переменная возвращает текст, содержащийся в отправленной форме и ее потомках. Содержится в ключе gtm.elementText.

В качестве примера отправим форму на тестовом сайте graphanalytics.ru

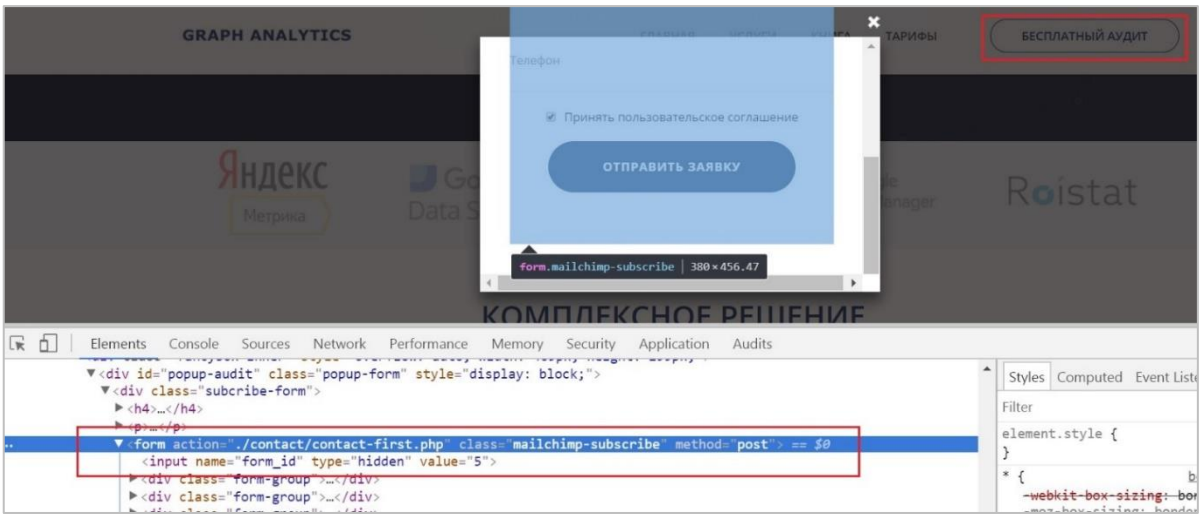


Рис. 264. Пример отправки формы

После отправки формы отладчик Google Tag Manager зафиксировал событие **gtm.formSubmit**, при переходе в которое (вкладка Variables) нам станут доступны значения переменных из этой категории:

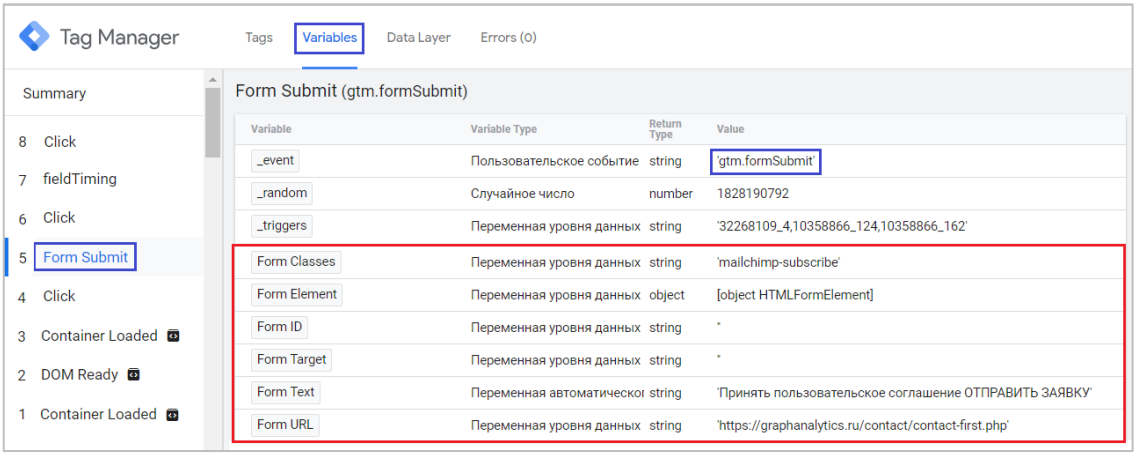


Рис. 265. Пример встроенных переменных типа Формы

- **Form Element** – [object HTMLFormElement], представляет объект HTMLFormElement, созданный на основе HTMLElement и наследующий его свойства, методы и события.
- **Form Classes** – mailchimp-subscribe
- **Form ID** - пустое значение, нет данных по атрибуту id у этого элемента
- **Form Target** - пустое значение, нет данных по атрибуту target у этого элемента
- **Form Text** – Принять пользовательское соглашение ОТПРАВИТЬ ЗАЯВКУ (весь собранный текст из данной формы)
- **Form URL** - https://graphanalytics.ru/contact/contact-first.php (обработчик, на который отправляются данные из формы)

На вкладке Data Layer можно увидеть, как переменные получают доступ к уровню данных и считывает ключи, которые задаются триггерами **Отправка формы**.

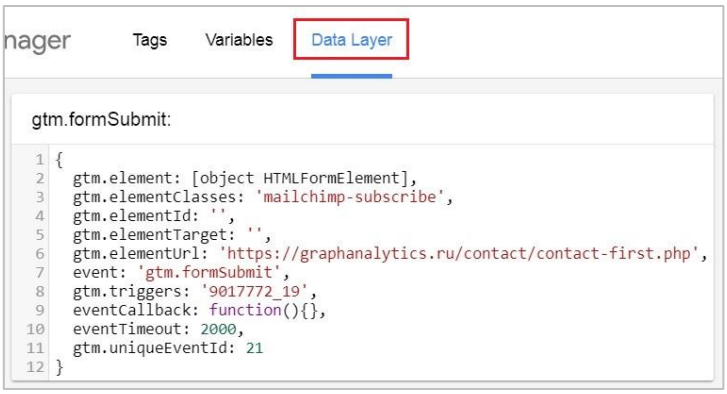


Рис. 266. gtm.formSubmit в Data Layer

История

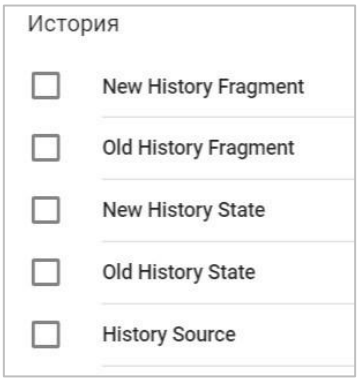


Рис. 267. Встроенные переменные типа История

История – это место, где хранится информация о том, по каким ссылкам в открытой вкладке браузера переходил посетитель. Они фиксируются, начиная с первой и заканчивая текущей. С помощью JavaScript мы можем перемещаться по данной истории и возвращаться к определенному элементу, добавлять новый элемент и изменять содержимое текущего.

Подробную информацию о добавлении и изменении записей истории можно получить в документации разработчика на сайте **developer.mozilla.org** (см. приложение).

Данная категория позволяет организовать навигацию или смоделировать поведение на сайтах, на которых контент подгружается динамически, без перезагрузки страниц (как правило, с помощью AJAX). При этом в адресной строке может меняться хэш (/#contacts, /#price, /#otzivi и т.д.).

Встроенный триггер в Google Tag Manager, который используется для прослушивания изменений в истории, называется **Изменение в истории** (событие **gtm.HistoryChange**).

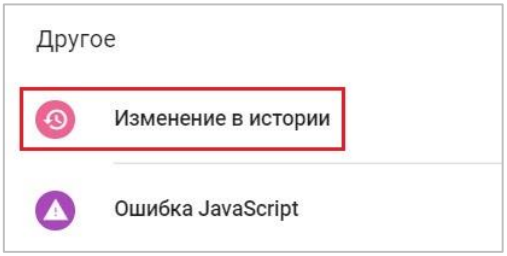


Рис. 268. Триггер – Изменение в истории

Категория **История** в GTM состоит из 5 встроенных переменных:

New History Fragment – возвращает переменную, которая содержит новое значение хэша (#) URL-сайта после совершения пользовательского события Изменение в истории. Содержится в ключе gtm.newUrlFragment.

Old History Fragment – обратное действие, возвращает переменную, которая содержит предыдущее значение хэша URL-сайта до совершения пользовательского события. Содержится в ключе gtm.oldUrlFragment.

New History State – возвращает объект, содержащий новое состояние истории после того, как произошло событие и метод **history.pushState()** был выполнен. Содержится в ключе gtm.newHistoryState.

Old History State – возвращает объект, содержащий старое состояние истории перед тем, как произошло событие и метод **history.pushState()** был выполнен. Содержится в ключе gtm.oldHistoryState.

History Source – возвращает строку-событие, которое привело к изменению объекта истории. Например, **pushState** или **popstate** (см. приложение).

Разберем на примере. Для демонстрации функциональности будем использовать меню тестового сайта.

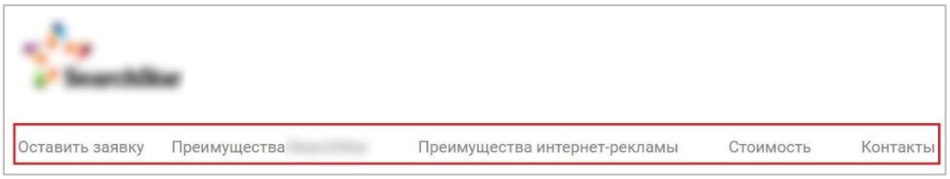


Рис. 269. Пример встроенных переменных типа История

Предположим, мой начальный путь был с ссылки раздела Стоимость. Ссылка имеет вид site.ru/#4, где #4 – хэш, так называемый якорь, разметка блоков на веб-страницах.

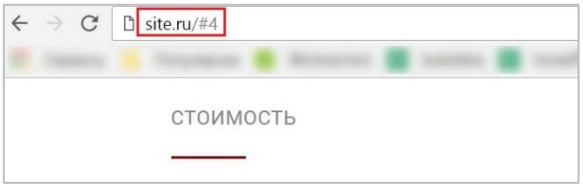


Рис. 270. Ссылка с якорем (хэшем)

Далее я перехожу на раздел Оставить заявку, ссылка которого имеет вид site.ru/#1. Как видим, произошла подмена значения в ссылке (адресной строке браузера) без перезагрузки страницы.

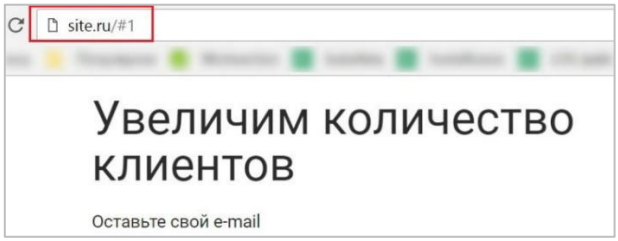


Рис. 271. Подмена значения в ссылке без перезагрузки страницы

Google Tag Manager зафиксировал событие, связанное с изменением истории.

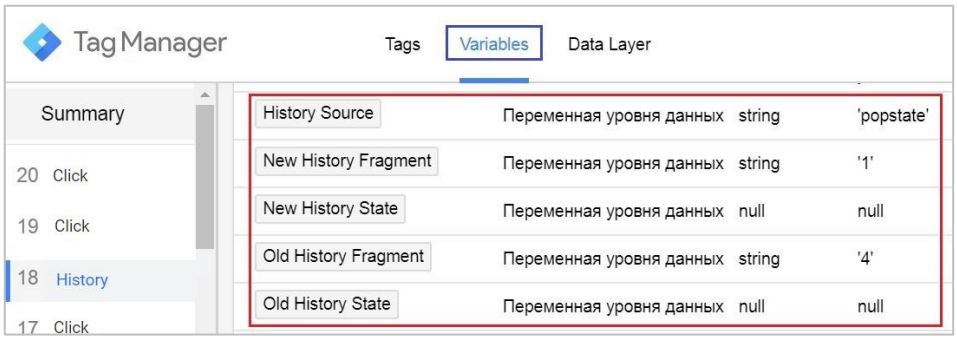


Рис. 272. Пример встроенных переменных типа История

Переменные принимают соответствующие значения:

- **History Source** – popstate, поскольку всякий раз, когда пользователь переходит к новому состоянию, происходит событие popstate, и свойство события state содержит копию объекта записи истории.

Данное значение будет и тогда, когда мы используем переходы по кнопкам Вперед - Назад в браузере.

Рис. 273. Кнопки Вперед-Назад в браузере

К тому же, браузеры работают с событием popstate по-разному. Chrome и Safari всегда вызывают popstate по окончании загрузки страницы, а Firefox не делает этого.

- **New History Fragment** – 1, так как мы осуществили переход site.ru/#4 на site.ru/#1
- **Old History Fragment** – 4
- **New History State** и **Old History State** – null, пустое значение.

На вкладке Data Layer можно увидеть, как переменные получают доступ к уровню данных и считывает ключи, которые задаются триггерами **Изменение в истории**.

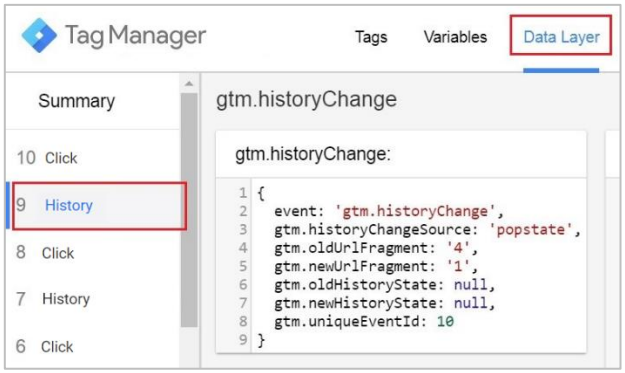


Рис. 274. gtm.historyChange в Data Layer

Видео

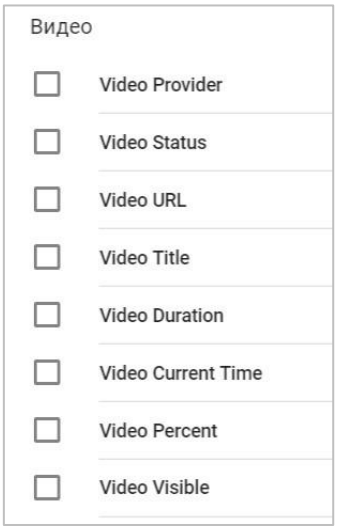


Рис. 275. Встроенные переменные типа Видео

Категория переменных, которая необходима для отслеживания видео, встроенных на сайт. Триггер в Google Tag Manager, который используется для прослушивания, называется **Видео YouTube** (событие **gtm.video**).

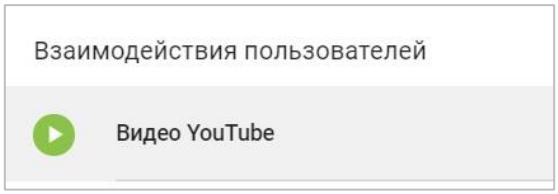


Рис. 276. Триггер – Видео YouTube

Google Tag Manager добавил 8 переменных в эту категорию:

Video Provider — возвращает значение поставщика видео. Сейчас доступен только YouTube. Содержится в ключе gtm.videoProvider.

Video Status — возвращает состояние видео в момент регистрации пользовательского события. Например, Start или Pause видео. Содержится в ключе gtm.videoStatus.

Video URL — URL-адрес, ссылка видео на YouTube. Содержится в ключе gtm.videoUrl.

Video Title — название видео. Содержится в ключе gtm.videoTitle.

Video Duration — общая продолжительность видео в секундах, выраженная целом числом. Содержится в ключе gtm.videoDuration.

Video Current Time — возвращает текущее время видео в секундах, в которое произошло пользовательское событие. Содержится в ключе gtm.videoCurrentTime.

Video Percent — возвращает значение воспроизведенного видео, выраженное в процентах целым числом (от 0 до 100), на момент, когда состоялось пользовательское событие. Содержится в ключе gtm.VideoPercent.

Video Visible — возвращает значение видимости видео в окне браузера. Если видео отображается в области просмотра, результатом будет значение true, если же в другой области (например, в нижней части страницы, на фоновой вкладке) – false. Содержится в ключе gtm.videoVisible.

Создайте триггер **Видео YouTube** и зададим ему соответствующие настройки:

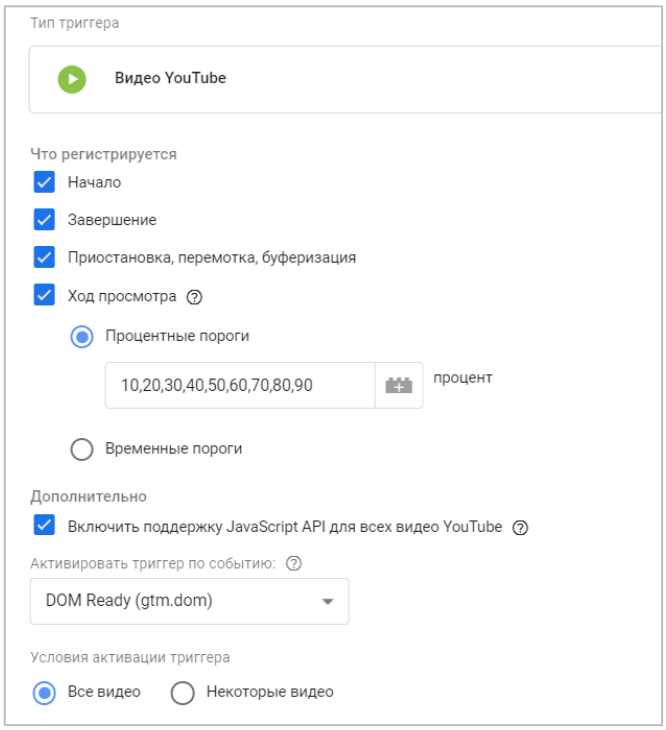


Рис. 277. Настройки триггера Видео YouTube

- **Начало (Start)** – пользователь начинает просмотр видео;
- **Завершение (Complete)** – пользователь достигает конца видео;
- **Приостановка (пауза), перемотка, буферизация (Pause, Seeking, Buffering)** – пользователь останавливает, перематывает видео или когда происходит буферизация;
- **Ход просмотра (Progress)** – пользователь проходит процентный или временный порог (время измеряется в секундах). Целые положительные числа указываются через запятую.
- **Включить поддержку JavaScript API для всех видео YouTube**;
- **Активировать триггер по событию** – возможность задать событие, когда триггер должен начать отслеживание релевантных взаимодействий видео (сразу же после загрузки контейнера gtm.js, после обработки модели DOM gtm.dom или когда все окно загружено gtm.load). По умолчанию – DOM Ready (gtm.dom).

Установив этот флажок, вы включите YouTube iFrame Player API. В результате ко всем URL видеопроигрывателя YouTube будет добавлен параметр *enablejsapi = 1* для управления проигрывателем через iframe или JavaScript.

Примечание: если при загрузке Google Tag Manager воспроизведение ролика уже началось, оно начнется сначала.

Сохраняем изменения. Обновляем режим предварительного просмотра и переходим в отладчик Google Tag Manager. На сайте перемотаем видео чуть вперед и приостановим его на кадре:

Рис. 278. Пример видео с YouTube

В отладчике будет доступна информация по всем 8 переменным:

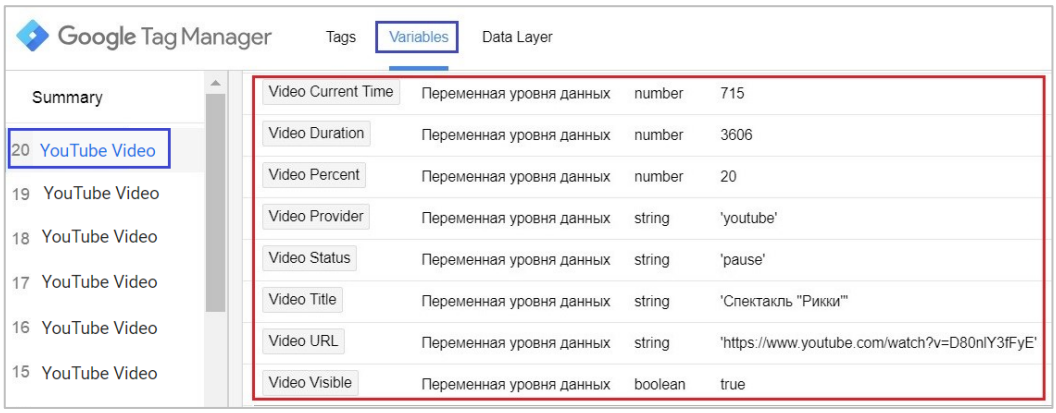


Рис. 279. Пример встроенных переменных типа Видео

- **Video Current Time** – 715 (в секундах), текущее время, в которое произошло пользовательское событие gtm.video;
- **Video Duration** – 3606 (в секундах), общая продолжительность видео;
- **Video Percent** – 20 (%, процент), процент воспроизведенного видео до момента совершения пользовательского события gtm.video;
- **Video Provider** – youtube, поставщик видео (пока единственный);
- **Video Status** – pause, так как оно приостановлено;
- **Video Title** – Спектакль Рикки, название видеоролика;
- **Video URL** – https://www.youtube.com/watch?t=1118&v=u-wkJTzEdns, ссылка на видео на YouTube;
- **Video Visible** – значение true, так как видео отображалось в видимой области экрана браузера.

На вкладке **Data Layer** можно увидеть, как переменные получают доступ к уровню данных и считывает ключи, которые задаются триггерами **Видео YouTube**.

Рис. 280. gtm.video в Data Layer

Прокрутка

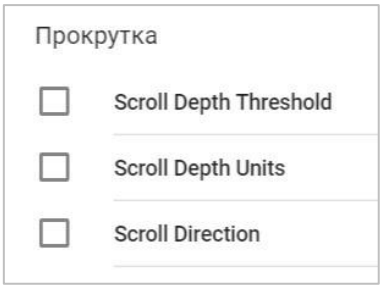


Рис. 281. Встроенные переменные типа Прокрутка

Триггер **Глубина прокрутки** включает базовые опции и позволяет отслеживать как вертикальную глубину скроллинга, так и горизонтальную.

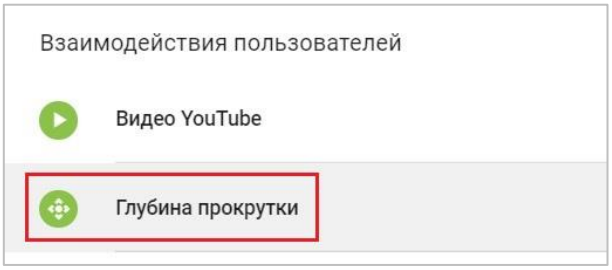


Рис. 282. Триггер – Глубина прокрутки

Доступно три переменных:

Scroll Depth Threshold - возвращает значение порога прокрутки, который был пересечен в результате пользовательского события gtm.scrollDepth. Содержится в ключе gtm.scrollThreshold.

Scroll Depth Units - переменная возвращает значение в пикселях (pixels) или процентах (percent), в зависимости от настроек триггера. Содержится в ключе gtm.scrollUnits.

Scroll Direction – переменная возвращает значение направления прокрутки (vertical или horizontal), в зависимости от настроек. Содержится в ключе gtm.scrollDirection.

Создайте триггер **Глубина прокрутки** и зададим ему соответствующие настройки (пример):

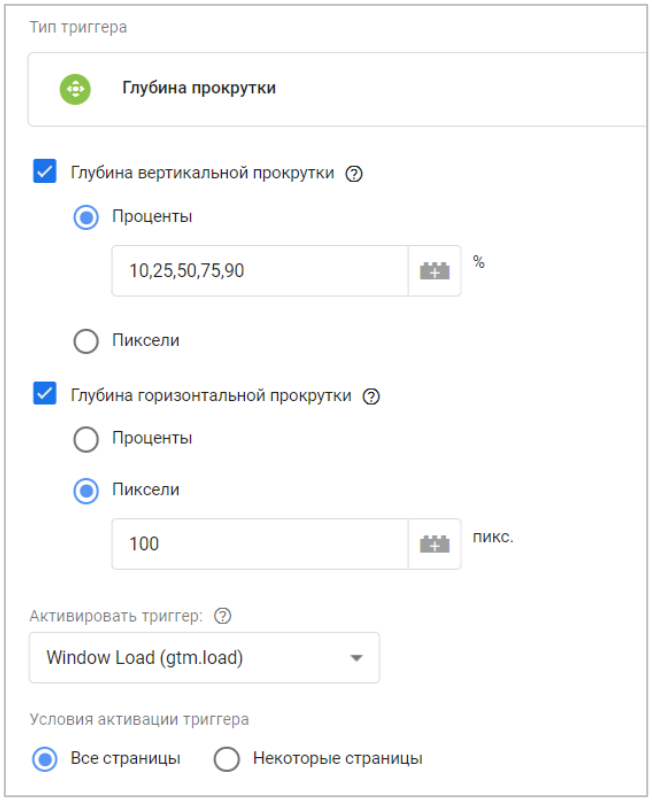


Рис. 283. Настройки триггера Глубина прокрутки

- **Глубина вертикальной прокрутки в процентах** – 10, 25, 50, 75, 90;
- **Глубина горизонтальной прокрутки в пикселях** – 100;
- **Активировать триггер** – возможность задать событие, когда триггер должен начать отслеживание релевантных взаимодействий по скроллингу (сразу же после загрузки контейнера gtm.js, после обработки модели DOM gtm.dom или когда все окно загружено gtm.load). По умолчанию – Window Load (gtm.load);
- **Условия активации триггера** – Все страницы.

Сохраняем изменения. Обновляем режим предварительного просмотра и переходим в отладчик Google Tag Manager. На сайте проскроллим чуть вниз страницы и увидим пользовательское событие gtm.scrollDepth:

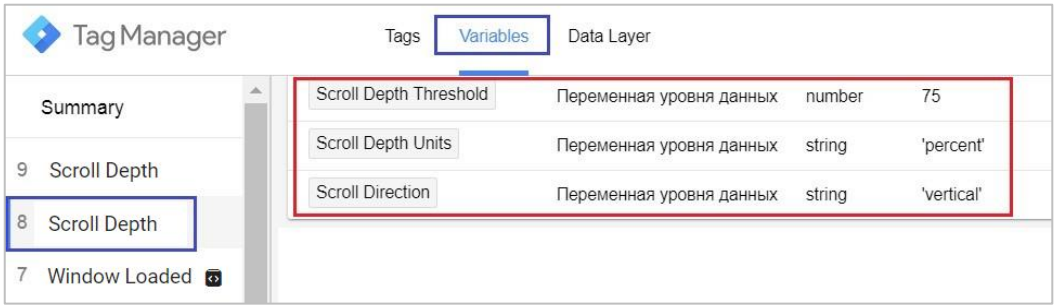


Рис. 284. Пример встроенных переменных типа Прокрутка

- **Scroll Depth Threshold** – 75% (процент прокрутки), поскольку вертикальная прокрутка у нас была задана в %;
- **Scroll Depth Units** – percent, здесь значение может быть вертикальный или горизонтальный, в зависимости от типа прокрутки, который пересек порог;
- **Scroll Direction** – vertical, тип/направление прокрутки.

А вот так выглядит горизонтальный скроллинг в пикселях (в нашем примере):

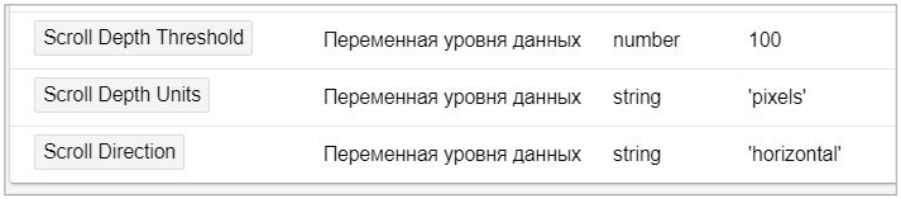


Рис. 285. Пример горизонтальной прокрутки

- **Scroll Depth Threshold** – 100;
- **Scroll Depth Units** – pixels;
- **Scroll Direction** – horizontal.

На вкладке Data Layer можно увидеть, как переменные получают доступ к уровню данных и считывает ключи, которые задаются триггерами **Глубина прокрутки**.

Рис. 286. gtm.scrollDepth в Data Layer

Отслеживать прокрутку без привязки к элементам сложно. 50% глубины прокрутки для статьи длиной 1000 символов и 50% прокрутки для статьи 20000 символов не дадут никаких данных для анализа. Также триггер не позволяет отслеживать прокрутку до конкретных HTML-элементов. В таких случаях используют другой триггер, который называется **Видимость элемента**.

Видимость

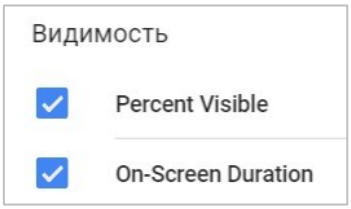


Рис. 287. Встроенные переменные типа Видимость

Переменные из этой категории помогают отслеживать данные о том, где посетитель остановился и сколько времени потратил на просмотр элементов на сайте – заголовков, текстовых абзацев, изображений и т.д. Функция появилась не так давно, но является очень полезной, так как теперь есть возможность точнее получить представление о том, как пользователи взаимодействуют с контентом сайта и какие элементы остаются без внимания.

Встроенный триггер – **Доступность элемента**, пользовательское событие **gtm.elementVisibility**.

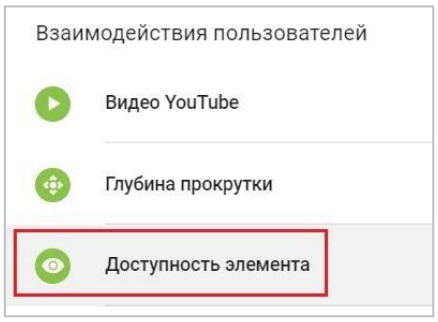


Рис. 288. Триггер – Доступность элемента

В Google Tag Manager в категории **Видимость** две переменных:

Percent Visible – возвращает числовое значение от 0 до 100, которое показывает, какой процент выбранного элемента был виден при срабатывании триггера. Содержится в ключе gtm.visibleRatio.

On-Screen Duration – возвращает числовое значение, которое показывает, как долго выбранный элемент был виден при срабатывании триггера. Содержится в ключе gtm.visibleTime.

Создайте триггер **Доступность элемента** и зададим ему соответствующие настройки (пример):

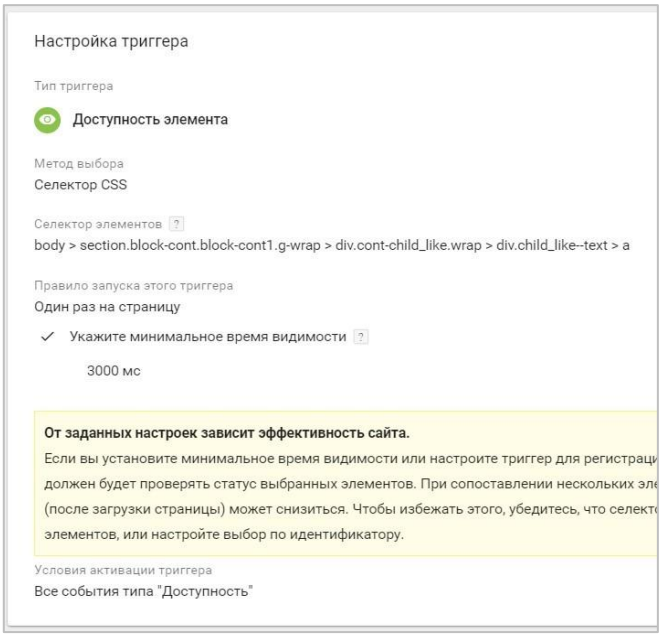


Рис. 289. Пример настройки триггера Доступность элемента

Какие-либо настройки триггеров в этой статье целенаправленно опускаются, поскольку более подробно о триггерах будет говориться в далее. Стоит отметить, что в данном примере был выбран элемент (кнопка *Записаться на бесплатное занятие*), при прокрутке которого активировалось пользовательское событие gtm.elementVisibility.

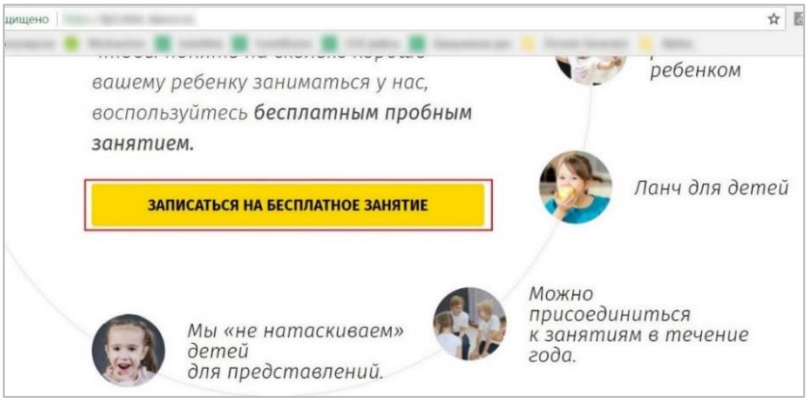


Рис. 290. Кнопка Записаться на бесплатное занятие как условие активации события

- **Метод выбора** – Селектор CSS;
- **Селектор элементов** - body > section.block-cont.block-cont1.g-wrap > div.cont-child_like.wrap > div.child_like--text > a ;
- **Правило запуска этого триггера** – Один раз на страницу;
- **Минимальное время видимости (в миллисекундах)** – 3000 (как долго выбранный элемент должен быть виден на экране для срабатывания триггера);
- **Условия активации триггера** – Все события типа Доступность.

Сохраняем изменения. Обновляем режим предварительного просмотра и переходим в отладчик Google Tag Manager. На сайте мы проскроллили до данной кнопки и через минимальное время видимости элемента произошла фиксация события gtm.elementVisibility.

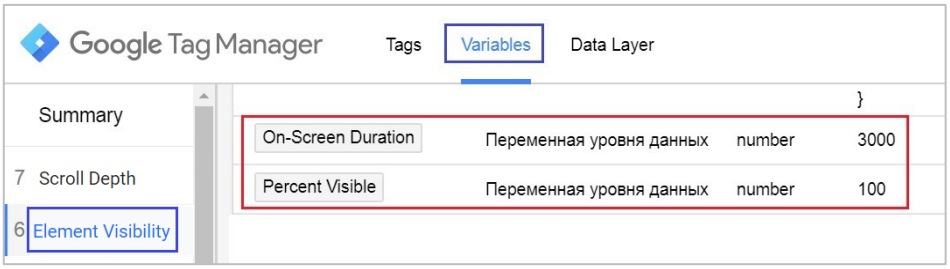


Рис. 291. Пример встроенных переменных типа Видимость

- **On-Screen Duration** – 3000 (в миллисекундах), как долго выбранный элемент был виден при срабатывании триггера;
- **Percent Visible** – 100 (в процентах), видимость элемента. В нашем примере он был виден на все 100%.

На вкладке Data Layer можно увидеть, как переменные получают доступ к уровню данных и считывает ключи, которые задаются триггерами **Доступность элемента**.

Рис. 292. gtm.elementVisibility в Data Layer

Все вышеупомянутые встроенные переменные используются для веб-контейнеров, однако еще есть большая часть переменных для контейнеров AMP, iOS, Android, и устаревших мобильных контейнеров, использующих контейнеры и SDK ниже версии 5. Их разбор вынужденно опускается в силу узкой тематики. Подробнее обо всех остальных встроенных переменных Google Tag Manager читайте в официальной справке Google.

Пользовательские переменные

Для создания пользовательской переменной перейдите в **Переменные** и нажмите кнопку **Создать**.

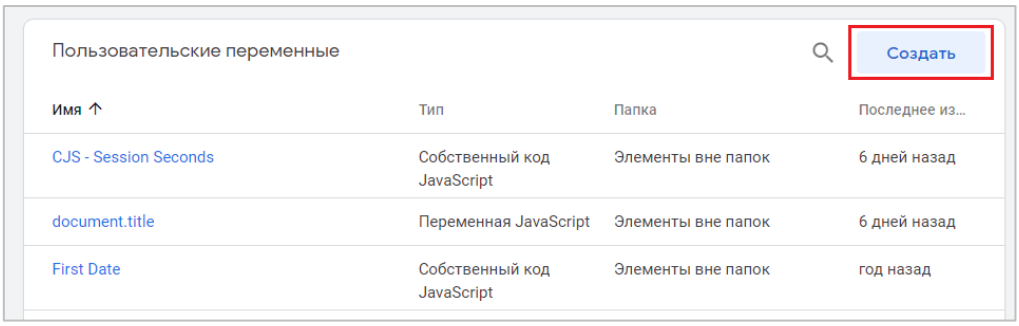


Рис. 293. Создание пользовательской переменной

При создании переменной необходимо указать ее имя. Далее нам доступно на выбор 5 категорий пользовательских переменных:

1. Навигация
2. Переменные страницы
3. Элементы страницы
4. Утилиты
5. Данные контейнера

Навигация

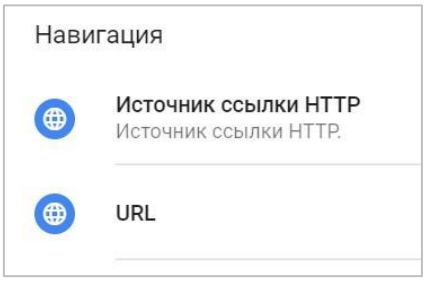


Рис. 294. Пользовательские переменные типа Навигация

С помощью переменных этой категории можно отслеживать с какой страницы к вам пришел пользователь. Или если он перемещается внутри сайта, от страницы к странице, на каждой следующей странице доступна информация о предыдущей странице пользователя.

Доступно две пользовательских переменных:

Источник ссылки HTTP

Значение извлекается из **document.referrer**. Аналог встроенной переменной (Referrer). У данной переменной есть несколько типов компонентов: полный URL, протокол, имя хоста, порт, путь, запрос и фрагмент.

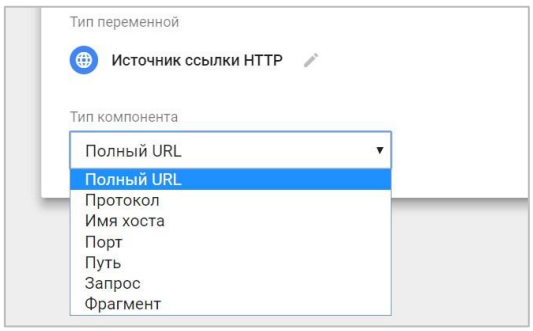


Рис. 295. Типы компонента - Источник ссылки HTTP

Разберем все типы на примере:

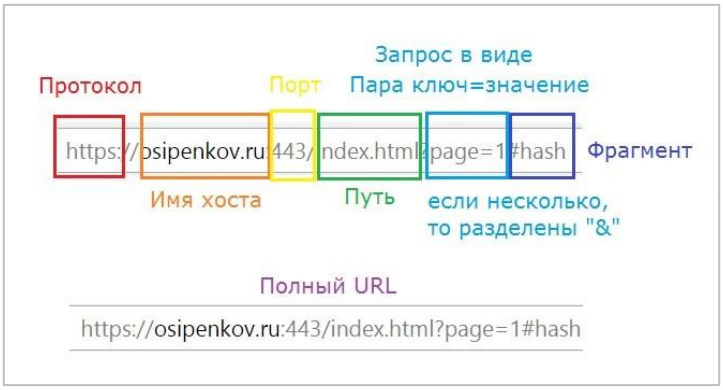


Рис. 296. Типы компонента на примере

Полный URL-адрес - возвращает полный URL-адрес без фрагмента хеширования (#). Например, https://osipenkov.ru/index.html?page=1

Протокол - возвращает протокол URL. Например, https

Имя хоста - возвращает имя хоста URL-адреса без номера порта. Например, osipenkov.ru или с www

Порт - возвращает номер порта, используемый в URL-адресе, или 80 для HTTP / или 443 для HTTPS, если URL-адрес не имеет номера порта. На примере выше число 443 вставлено лишь с целью демонстрации

Путь - возвращает только путь в URL. Например, index.html

Запрос - возвращает всю строку параметров запроса (без указания **?**) при условии, что вы не укажете ключ запроса. Если вы укажете ключ запроса, возвращается только значение этого ключа или не определено, если в URL-адресе такого ключа нет. Если помните настройку представления в Google Analytics под названием **Отслеживание поиска по сайту**, то разобраться в типе компонента запрос не составит проблем

Фрагмент - возвращает значение фрагмента URL-адреса без ведущего #. Например, hash

Вы можете создать триггер, который запускается, когда Источник ссылки HTTP не содержит ваш собственный домен. Таким образом можно отслеживать пользователей, перешедших на сайт извне, а не переходящих по страницам на сайте.

URL

Универсальная переменная, которая может быть использована для доступа к компонентам текущей URL-страницы (по умолчанию) или любого URL-адреса, возвращаемой переменной. Аналог встроенных переменных Page URL, Page Hostname, Page Path.

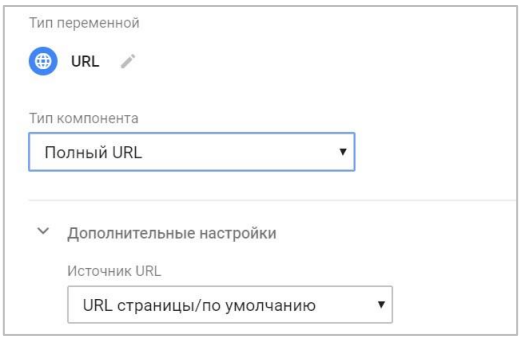


Рис. 297. Пользовательская переменная URL

Доступны те же типы компонентов, что и у **Источник ссылки HTTP**.

Переменные страницы

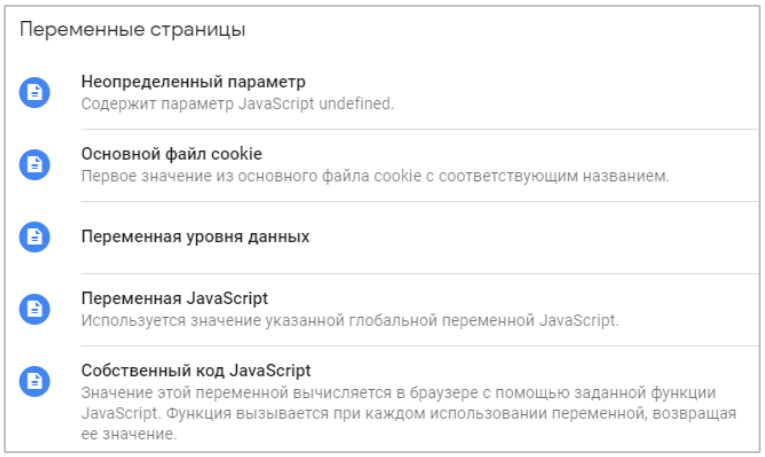


Рис. 298. Пользовательские переменные типа Переменные страницы

Неопределенный параметр

Переменная содержит параметр JavaScript **undefined**. Позволяет вернуть «значение не присвоено» вместо строки (string), то есть значение в другом типе данных.

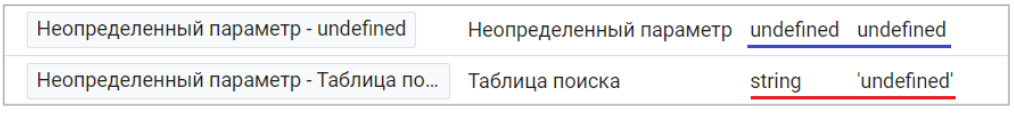


Рис. 299. Отличие типов данных

У переменной нет настроек.

Основной файл cookie (1st Party Cookie)

Возвращает значение cookie, которое доступно для текущего сайта;

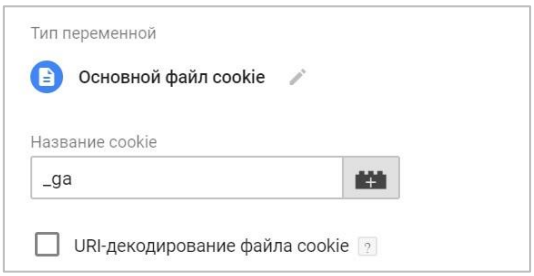


Рис. 300. Основной файл cookie

В качестве названия cookie можно задать **_ga** для различия пользователей. Таким образом при просмотре страницы сайта в отладчике Google Tag Manager будет доступно значение cookie пользователя:

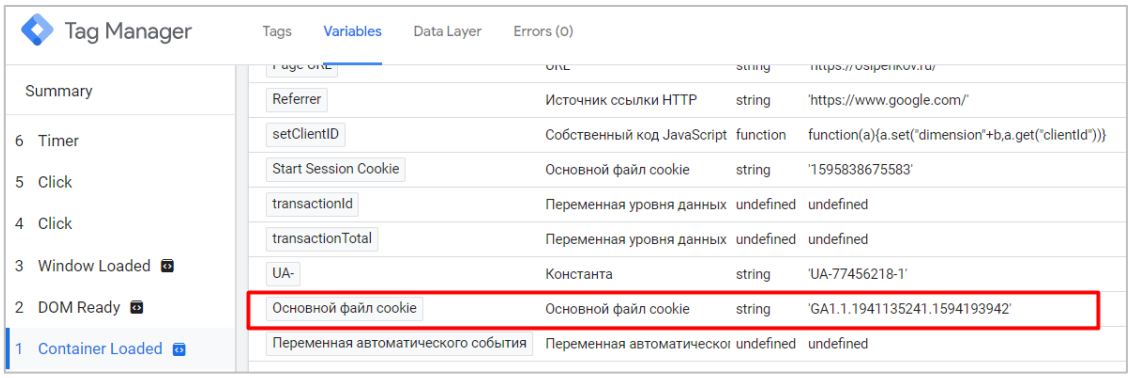


Рис. 301. Пример пользовательской переменной Основной файл cookie

URI-декодирование файла cookie. Приятной особенностью файлов cookie является то, что почти каждый браузер, поддерживающий JavaScript, обеспечивает и доступ сценариев к строкам cookie. Строки cookie представляются свойством cookie объекта Document. Это свойство доступно как для чтения, так и для записи. Когда вы присваиваете строку свойству document.cookie, браузер анализирует ее как строку файла cookie и добавляет ее в список строк cookie.

```
document.cookie = "username=yakov; expires=Friday, 01-Dec-2017 08:00:00 GMT;
path=/home";
```

Довольно часто значения полей cookie шифруются, что может оказаться проблемой при вставке их в строку cookie. Данная функция выполняет URL-кодирование и декодирование строк, которые передаются им в качестве аргументов, возвращая соответствующий результат. Если cookie возвращает значение **undefined**, то она не определена.

Переменная уровня данных

Один из наиболее часто используемых типов в GTM. Когда вы используете уровень данных (Data Layer), вы передаете пару *key:value* (ключ:значение) с помощью конструкции:

```
dataLayer.push({'var': 'value'});
```

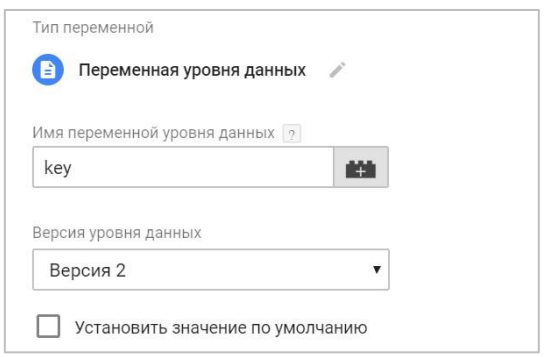


Рис. 302. Переменная уровня данных

Чтобы данные стали доступны в Google Tag Manager создается пользовательская переменная типа **Переменная уровня данных**, и в поле **Имя переменной уровня данных** указывается key.

Примечание: переменные уровня данных назначаются для страниц, а не для сеансов.

Для простых значений (строки, числа, булевы, функции) переменная будет возвращать все, что было недавно передано в ключ. Для объектов и массивов переменная возвращает результат рекурсивного слияния, где заменяются только общие ключи.

Вы можете использовать точечную нотацию для доступа к переменным ключам Data Layer, которые имеют точку в имени (например, **gtm.element**) или для доступа к свойствам объектов DOM (например, **gtm.element.dataset.name**).

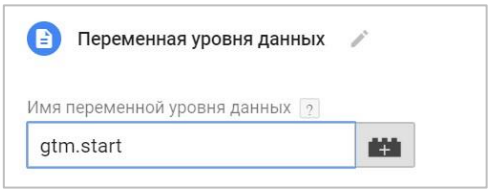


Рис. 303. Пример точечной нотации

Если вы не указали значение по умолчанию, переменная Data Layer вернет **undefined**.

Переменная JavaScript

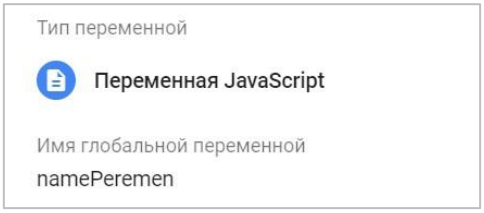


Рис. 304. Пользовательская переменная JavaScript

Принимает значение переменной JavaScript, имя которой указано в поле **Имя глобальной переменной**. Если такой переменной не существует, вернется значение **undefined**.

Например, у вас на сайте в коде страницы объявлена глобальная переменная:

```
var namePeremen = 5;
```

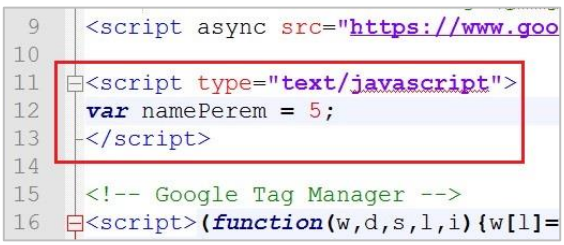


Рис. 305. Переменная namePeremen объявлена в коде сайта

В этом случае переменная JavaScript вернет значение глобальной переменной, что объявлена на странице, и ее тип данных:

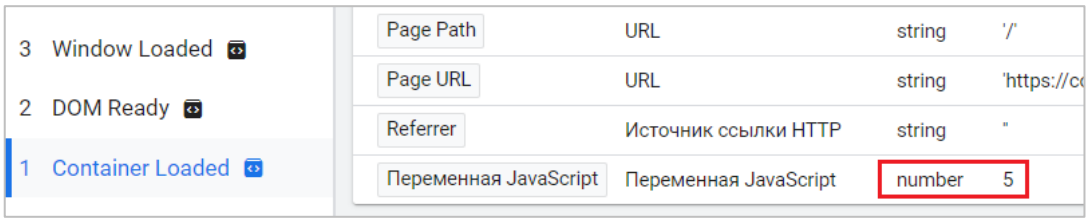


Рис. 306. Пример пользовательской переменной JavaScript

Собственный код JavaScript

Пользовательские JavaScript переменная является наиболее универсальным переменной в наборе. Вы можете использовать ее для запуска произвольного JavaScript на странице. Он создает контекст сценария, то есть вы также можете вызывать другие переменные изнутри, используя соответствующий синтаксис.

Значение этой переменной вычисляется в браузере с помощью заданной функции JavaScript. Функция вызывается при каждом использовании переменной, возвращая ее значение.

Пользовательская переменная JavaScript должна следовать нескольким правилам:

- скрипт должен быть размещен в функциональном блоке **function() { ... })**
- функция должна иметь оператор **return**
- функция должна возвращать только значение.

Если явного значения нет, функция возвращает неопределенное значение. Это может привести к нарушению нормальной работы контейнера.

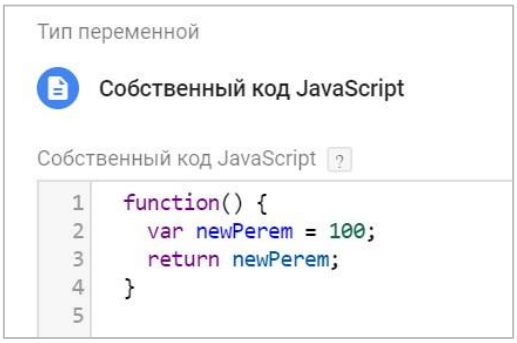


Рис. 307. Пример собственного кода JavaScript

В отладчике Google Tag Manager:

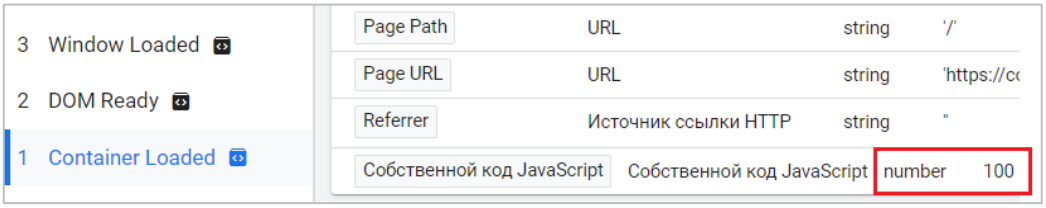


Рис. 308. Пример пользовательской переменной Собственный код JavaScript

Вы можете вернуть любую переменную или значение, даже другие функции, другие переменные GTM или ничего (return без возврата -> **undefined**, неопределенное значение).

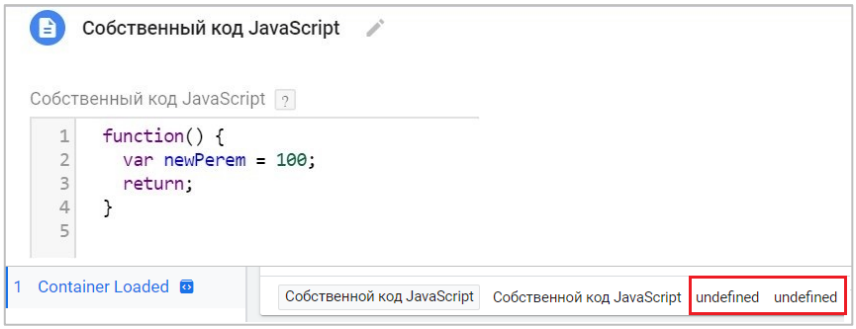


Рис. 309. Пример без возврата (неопределенное значение, undefined), return

Элементы страницы

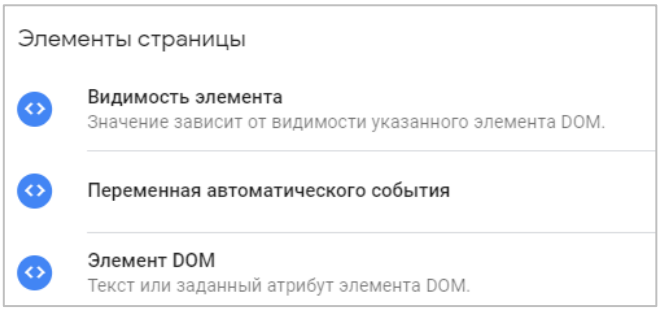


Рис. 310. Пользовательские переменные типа Элементы страницы

Следующая категория в Google Tag Manager состоит из 3 пользовательских переменных:

Видимость элемента

Позволяет вам определить какой конкретный элемент был виден в браузере пользователя. Используется с триггером **Доступность элемента**. Значение зависит от видимости указанного элемента DOM. В качестве примера зададим селектор элемента кнопки на сайте:

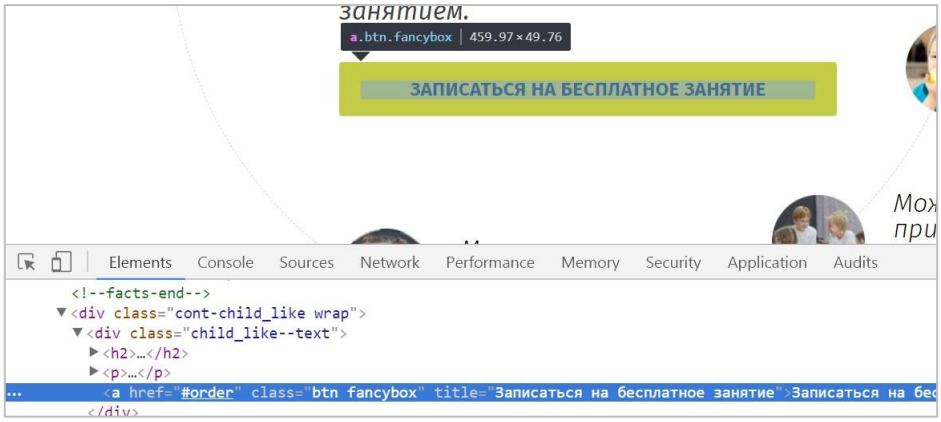


Рис. 311. Селектор элемента кнопки на сайте

Тип результата истина/ложь (true/false) и минимальный процент видимости:

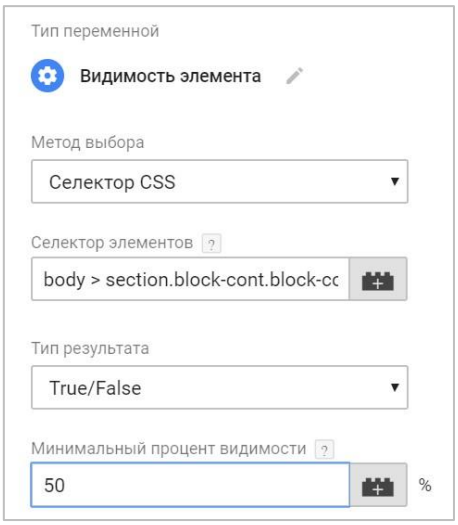


Рис. 312. Настройки в переменной Видимость элемента

В отладчике Google Tag Manager будет доступен следующий результат:

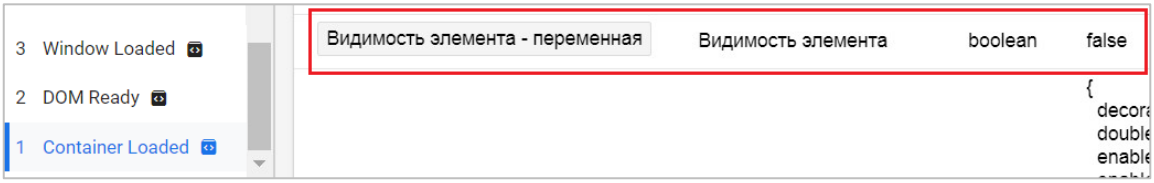


Рис. 313. Пример пользовательской переменной Видимость элемента

В случае видимости элемента в браузере при минимальном проценте видимости – *true*, в случае если элемент не был в поле зрения пользователя и не удовлетворяет проценту видимости – *false*.

Переменная автоматического события (Auto-Event Variable)

Переменная автоматического события используется для доступа к целевому элементу действия пользовательского события (клики, ошибки, отправки формы и т.д.). Когда вы создаете новую переменную Auto-Event, вам нужно указать только тот компонент целевого элемента, к которому вы хотите получить доступ.

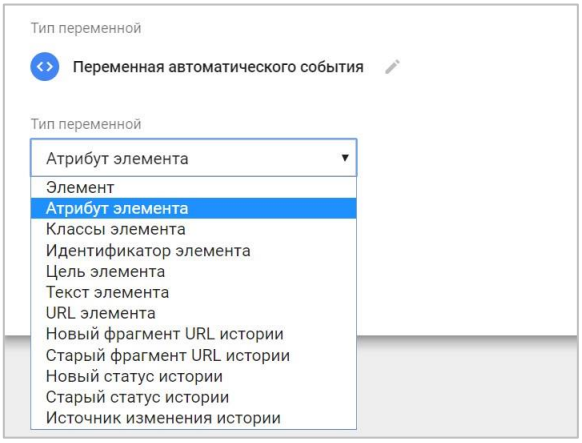


Рис. 314. Переменная автоматического события

Часть типов переменной совпадает со встроенными переменными (см. выше):

- **Элемент** = Click Element и Form Element;
- **Классы элемента** = Click Class и Form Class;
- **Идентификатор элемента** = Click ID и Form ID;
- **Цель элемента** = Click Target и Form Target;
- **Текст элемента** = Click Text и Form Text;
- **URL элемента** = Click URL и Form URL;
- **Новый фрагмент URL истории** = New History Fragment;
- **Старый фрагмент URL истории** = Old History Fragment;
- **Новый статус истории** = New History State;
- **Старый статус истории** = Old History State;
- **Источник изменения истории** = History Source.

Переменная автоматического события возвращает значение, соответствующее типу выбранного элемента. Если соответствующее автоматическое событие не было зарегистрировано, переменная возвращает значение по умолчанию (если установлено) или **undefined**.

Элемент DOM

Вы можете использовать переменную для получения текстового содержимого любого элемента DOM или для извлечения значения любого атрибута элемента DOM. Например, мы хотим получить значение текстового содержимого элемента кнопки Заказать рекламу:

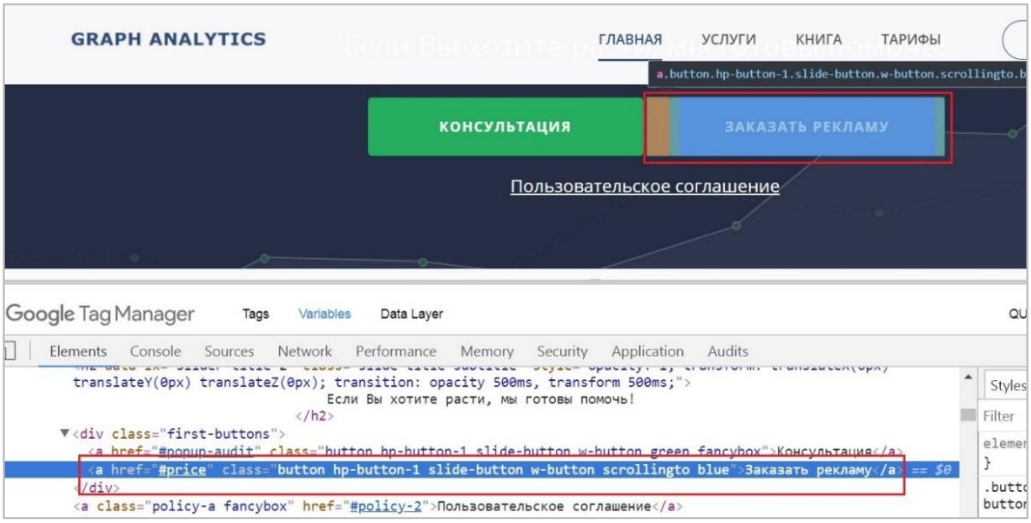


Рис. 315. Пример извлечения текстового содержимого элемента кнопки

Полный CSS-селектор у кнопки через Copy selector в браузере:

#home > div > div > div > div > div > a.button.hp-button-1.slide-button.w-button.scrollingto.blue

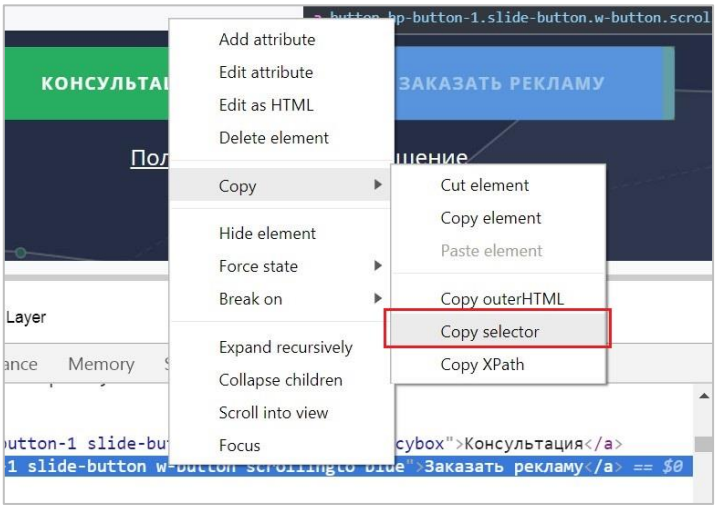


Рис. 316. Копирование селектора

Создайте пользовательскую переменную в Google Tag Manager:

Рис. 317. Пример пользовательской переменной Элемент DOM

Примечание: в данном примере CSS-селектор очень длинный и его можно сократить. Однако цель примера не в этом, так что оставим полный селектор, скопированный из консоли разработчика.

Если задано имя атрибута, переменной будет присвоено значение атрибута элемента DOM, в противном случае в качестве значения будет использоваться текст элемента DOM. В отладчике GTM:

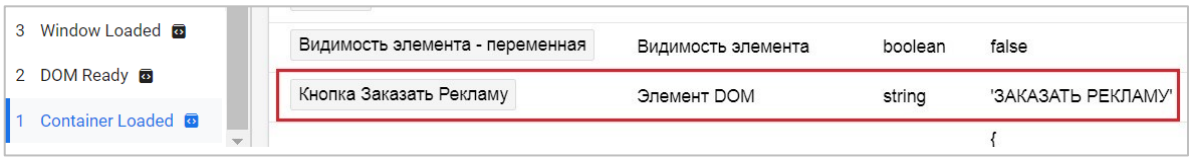


Рис. 318. Отображение значения кнопки в переменной Элемент DOM

Если элемент DOM с указанным идентификатором или CSS-селектором не найден, переменная возвращает нулевое значение (null).

Утилиты

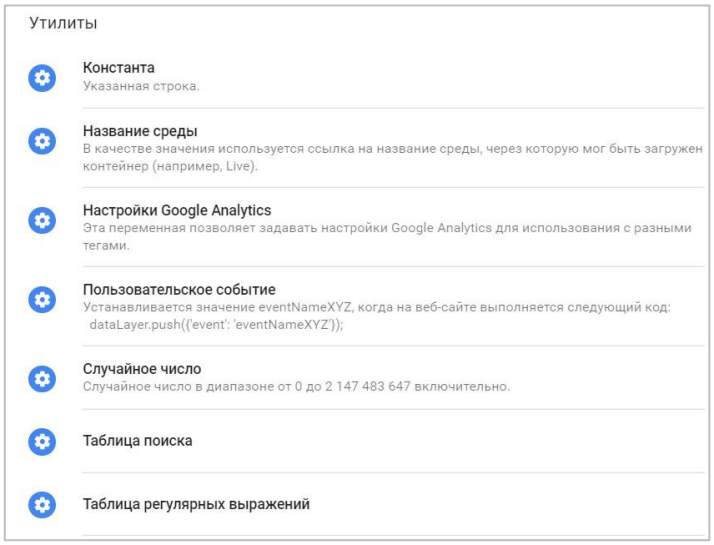


Рис. 319. Пользовательские переменные типа Утилиты

Константа

Постоянная переменная является наглядным примером того, как переменные могут быть использованы повторно. Константа будет постоянно принимать значение из поля Значение:

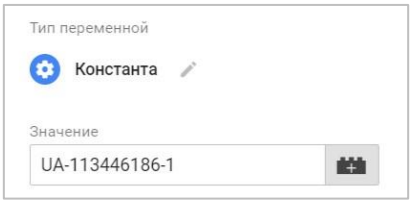


Рис. 320. Переменная Константа

Чаще всего этот тип переменной используется для указания идентификатора отслеживания Google Analytics. Сохранив константу UA один раз, вам не нужно создавать его (идентификатор) каждый раз, когда вы создаете новый тег GA.

Название среды

Аналог встроенной переменной **Environment Name**, которая возвращает название пользовательской среды.

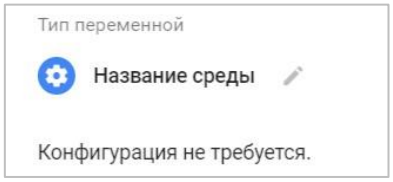


Рис. 321. Переменная Название среды

Настройки Google Analytics

Переменная возвращает набор параметров тега Universal Analytics. Используется **ТОЛЬКО** в теге Universal Analytics и для одновременной настройки нескольких тегов, например, для объединения своих пользовательских параметров (custom dimension) и полей.

Рис. 322. cookieDomain - auto

Домен cookie. По умолчанию стоит auto, также, как и при отслеживании в Google Analytics, для поля **cookieDomain** значение *auto*.

```
ga('create', 'UA-XXXXX-Y', 'auto');
```

При автоматической *auto* конфигурации домена cookieDomain библиотека *analyltics.js* самостоятельно подбирает домен для хранения файлов cookie.

Дополнительные настройки:

- Поля, которые необходимо задать – доступен раскрывающийся список с доступными полями;

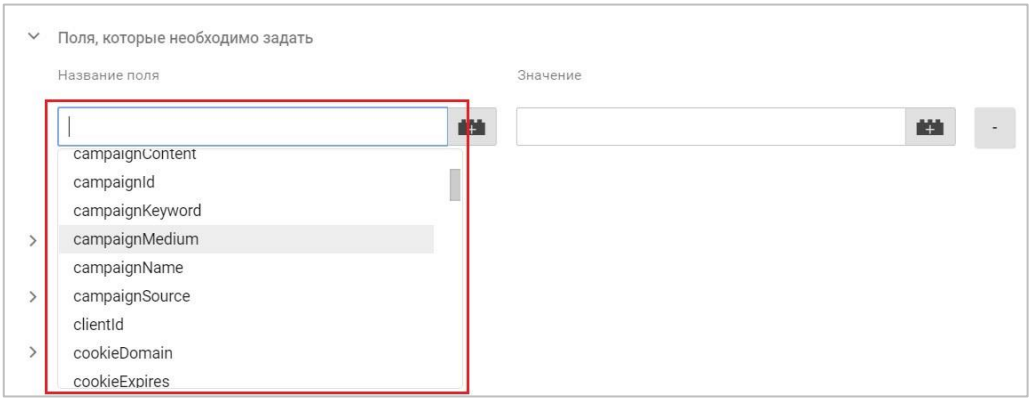


Рис. 323. Поля, которые необходимо задать

- Специальные параметры;
- Специальные показатели;
- Группы контента;
- Реклама - Включить функции для контекстно-медийной сети;
- Электронная торговля - Включить расширенные функции электронной торговли;
- Междоменное отслеживание;
- Расширенная конфигурация, в которой можно задать глобальную функцию, использовать отладочную версию и улучшенную атрибуцию ссылок.

Пользовательское событие

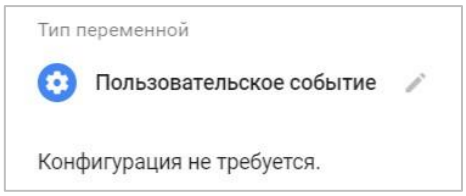


Рис. 324. Пользовательское событие

Принимает значение, равное текущему значению переменной **_event**. Аналог встроенной переменной Event (Утилиты).

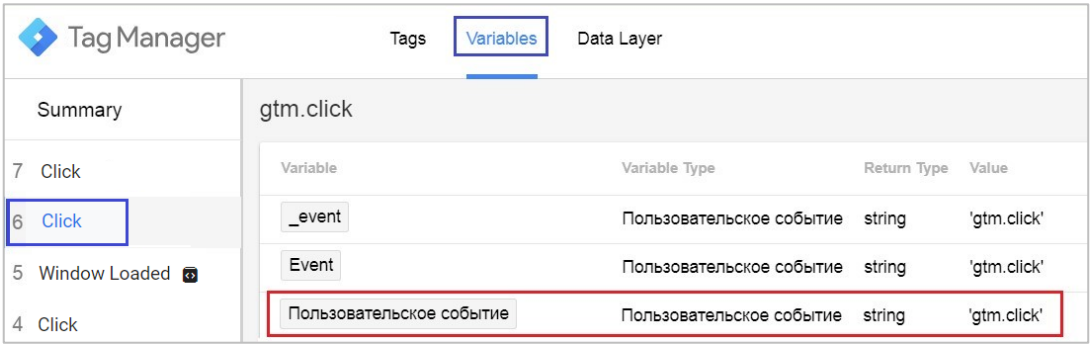


Рис. 325. Пример пользовательского события gtm.click

Случайное число

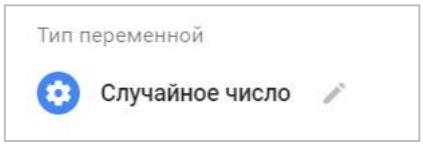


Рис. 326. Переменная Случайное число

Переменная возвращает случайное число в диапазоне от 0 до 2 147 483 647. Аналог встроенной переменной **Random Number**.

Таблица поиска

Тип переменной, который позволяет избежать конструкции **if..else** и **switch-case**.

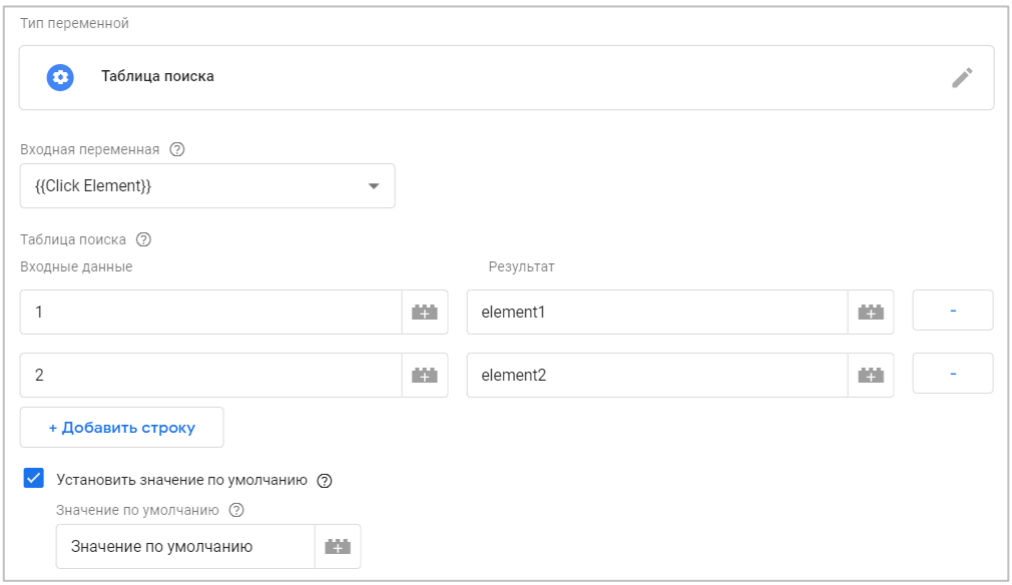


Рис. 327. Переменная Таблица поиска

В таблице поиска во входных данных задаются конкретные значения. Поиск значений всегда является точным совпадением и чувствительным к регистру.

Как работает таблица поиска?

1. Задается входная переменная. Например, **{{Click Element}}**;
2. Далее идет проверка по таблице поиска и входным данным;
3. Если переменная **{{Click Element}}** принимает одно из значений из таблицы, то входная переменная **{{Click Element}}** примет значение из поля Результат;
4. Если значение переменной **{{Click Element}}** не найдено среди таблицы поиска, то будет использовано значение по умолчанию (если указано в настройках).

Если вы не укажете значение по умолчанию, переменная вернет неопределенное значение **undefined** в случае, если совпадение не будет выполнено.

Таблица регулярных выражений

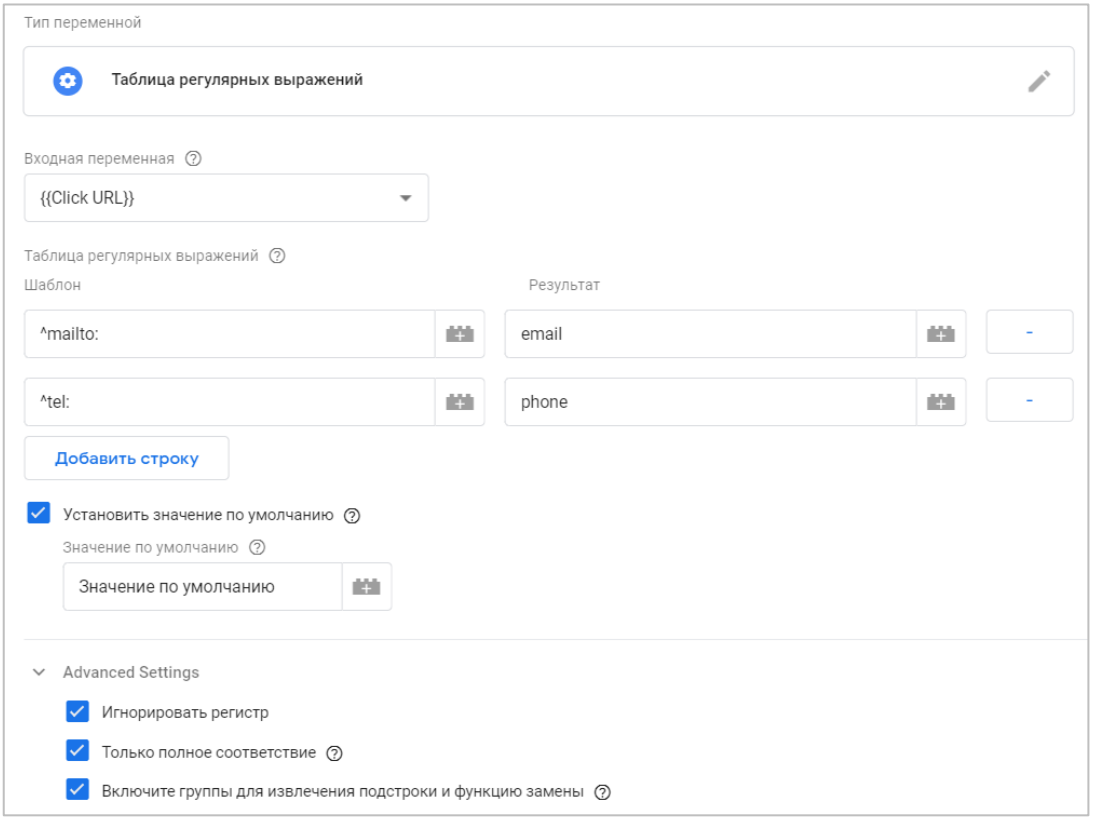


Рис. 328. Переменная Таблица регулярных выражений

Тип переменной, который похож на **Таблица поиска**, только более гибкий, поскольку здесь можно использовать регулярные выражения.

Данные контейнера

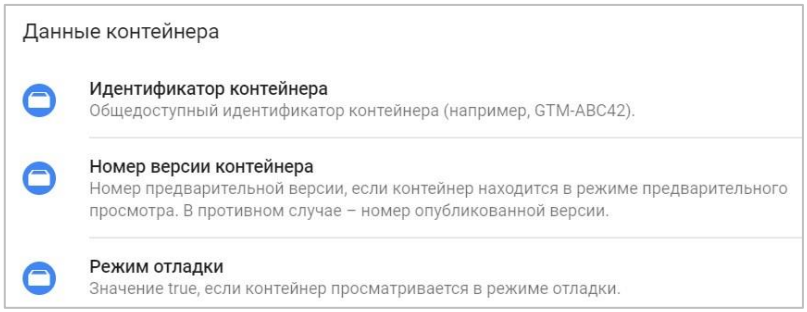


Рис. 329. Пользовательские переменные типа Данные контейнера

Три переменные из этой категории:

Идентификатор контейнера

Возвращает номер контейнера GTM. Например, GTM-NC2LK3M. Аналог встроенной переменной **Container ID**.

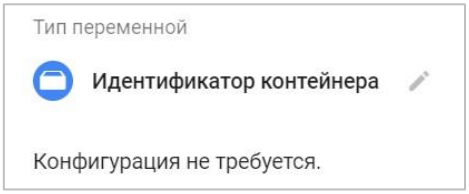


Рис. 330. Переменная Идентификатор контейнера

Номер версии контейнера

Возвращает версию контейнера. Например, 5. Аналог встроенной переменной **Container Version**.

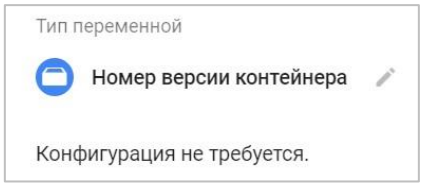


Рис. 331. Переменная Номер версии контейнера

Значение **QUICK_PREVIEW** возвращается в том случае, если вы просматриваете в режиме отладки.

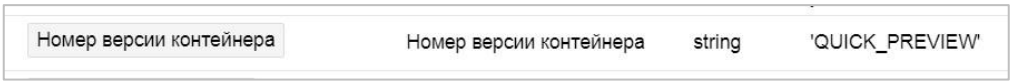


Рис. 332. QUICK_PREVIEW

Режим отладки

Возвращает значение *true*, если пользователь просматривает контейнер в режиме предварительного просмотра, и *false* - если нет. Аналог встроенной переменной **Debug Mode**.

Рис. 333. Режим отладки

С помощью этой переменной можно собрать все ваши хиты в отдельное тестовое свойство, а значение параметра **Режим отладки** в качестве пользовательского параметра, чтобы в Google Analytics создать под него отдельное представление и исключить режим отладки из статистики, отфильтровав его.

Встроенные и пользовательские переменные в GTM можно *копировать, удалять, просматривать изменения*, и у них можно *вывести примечания*. Для этого в правом верхнем углу нажмите на значок три точки.

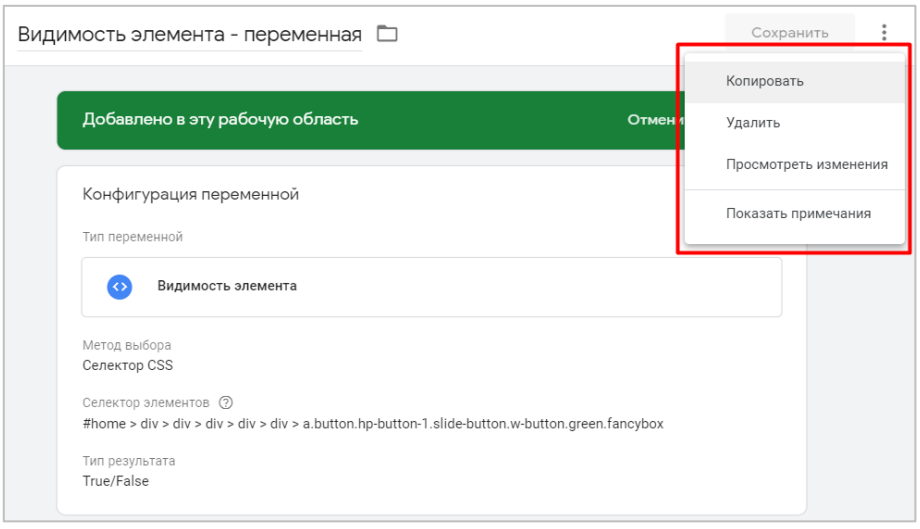


Рис. 334. Действия над переменными

В **Просмотреть изменения** можно узнать, что конкретно было изменено в переменной по сравнению с предыдущей версией, а в Показать примечания есть возможность заносить какие-то пометки, добавить описание по функциональности и т.д.

Уровень данных (dataLayer)

Уровень данных (dataLayer) – это объект (массив объектов) или переменная JavaScript, которая хранит и передает информацию с вашего сайта в Google Tag Manager. Затем эти данные могут передаваться и другим сервисам, например, в Google Analytics, Facebook, Criteo OneTag и др.

Информацию с помощью уровня данных можно передавать самую разную – от специальных параметров и показателей, данных по транзакциям (настройка расширенной электронной торговли), User ID до каких-либо пользовательских событий.

Рис. 335. Пример dataLayer

Очень часто уровень данных и **dataLayer** используют как синонимы. Это не так, поскольку уровень данных – это сам объект, с которым мы работаем и благодаря которому мы передаем данные в GTM, а **dataLayer** – всего лишь имя данного объекта. Оно в Google Tag Manager стандартное по умолчанию. Убедиться можно в этом перейдя в код контейнера:

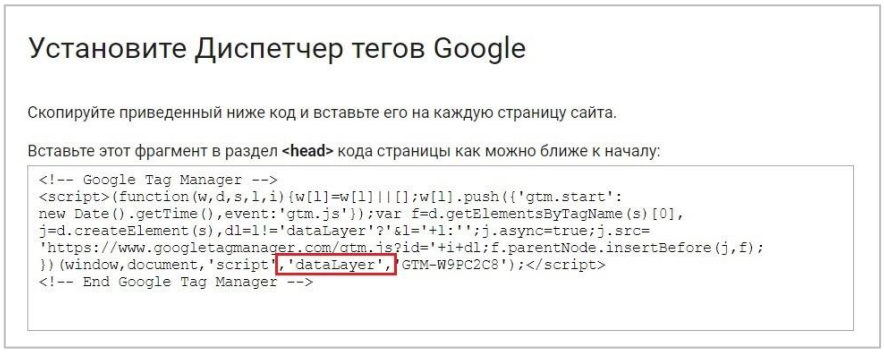


Рис. 336. Фрагмент кода контейнера GTM

Мы можем изменить данное имя и тогда будет использоваться не dataLayer, а то, которое захотим, например, такое:

```
<!-- Google Tag Manager -->
<script>(function(w,d,s,l,i){w[l]=w[l]||[];w[l].push({'gtm.start':
new Date().getTime(),event:'gtm.js'});var f=d.getElementsByTagName(s)[0],
j=d.createElement(s),dl=l!='dataLayer'?'&l='+l:'';j.async=true;j.src=
'https://www.googletagmanager.com/gtm.js?id='+i+dl;f.parentNode.insertBefore(j,f);
})(window,document,'script','urovenDannix','GTM-NC2KL3M');</script>
<!-- End Google Tag Manager -->
```

Рис. 337. Пример изменения уровня данных в коде сайта

Впоследствии все обращения к уровню данных мы должны выполнять, используя новое имя. Для того чтобы информация через уровень данных попадала в Google Tag Manager, необходимо использовать специальную конструкцию, которая задается по определенным правилам.

Все значения, которые используются в уровне данных GTM, состоят из пары *ключ:значение*. Имя ключа и значения заключаются в одинарные кавычки, между собой они разделяются двоеточием. Например:

```
dataLayer = [{'ключ1': 'значение1'}];
```

Если мы используем несколько пар **'ключ':'значение'**, то такие пары между собой разделяются запятыми. Все пары заключаются в фигурные скобки. Например:

```
dataLayer = [{'ключ1': 'значение1','ключ2': 'значение2'}];
```

Переменная dataLayer должна быть объявлена между тегами **<script></script>**. Существует несколько вариантов того, когда, каким образом и в каком месте должен быть описан уровень данных.

- **описание ДО кода контейнера GTM**. Конструкция ДО контейнера GTM выглядит так:

```
<script>
dataLayer = [{'ключ': 'значение'}];
</script>
```

К тому моменту, когда зафиксируется первое стандартное событие **Просмотр страницы (Page View)**, информация, которую мы передали с помощью данной конструкции, уже будет присутствовать в Google Tag Manager и мы с ней сможем работать.

В режиме предварительного просмотра это будет выглядеть так:

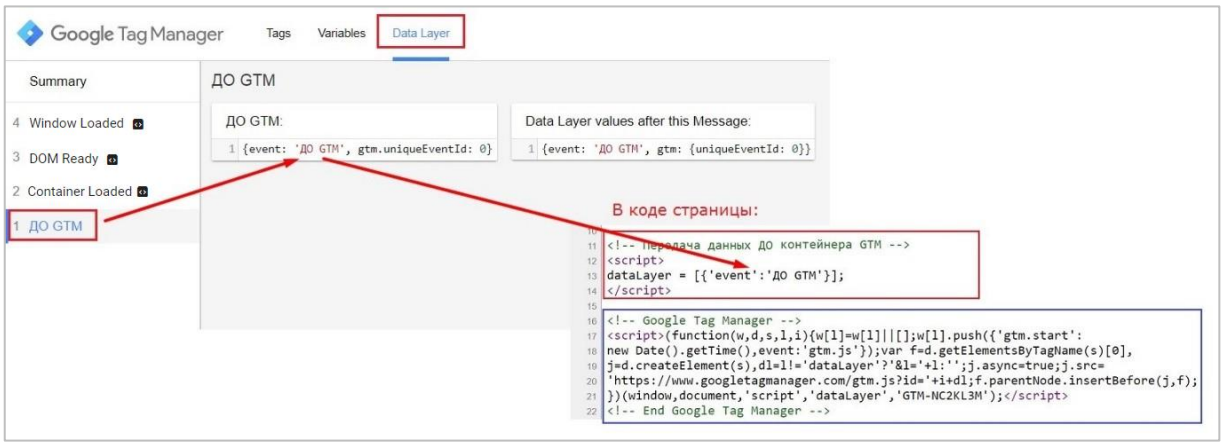


Рис. 338. Пример описания ДО кода контейнера GTM

Для отслеживания различных действий пользователя в GTM предусмотрена специальная переменная event. Event можно использовать внутри обработчика того или иного события. Наличие ключа **event** сообщает GTM, что нужно что-то сделать. Далее он действует в зависимости от содержания **event**, в данном случае выводит значение **ДО GTM**. Если **event** отсутствует или не задано, то в режиме отладки мы увидим слово **Message** и надпись:

Рис. 339. При отсутствии event будет отображаться сообщение Message

Например, при клике на кнопку, которая открывает форму с заказом услуги веб-аналитики, можно использовать конструкцию:

```
<a href="#analytics" data-tariff="Веб-аналитика" class="purchase-btn tariff-order fancybox " onclick="window.datalayer.push({'event':'button-click1', 'eventCategory' : 'click', 'eventAction' : 'knopka'});">Заказать услугу</a>
```

- **описание ПОСЛЕ кода контейнера GTM**. Конструкция ПОСЛЕ контейнера GTM выглядит так:

```
dataLayer.push ({'ключ':'значение'});
```

Отличие от описания ДО заключается в том, что здесь используется метод **push ()**, который добавляет один или более элементов в конец массива и возвращает новую длину массива. Помимо этого, квадратные скобки и равно заменились на круглые.

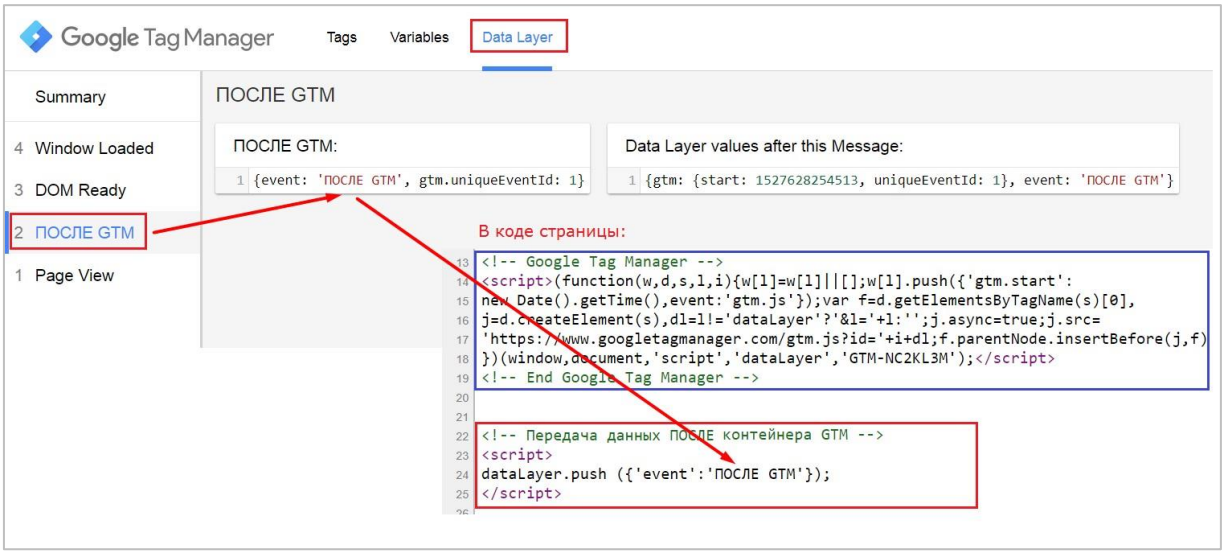


Рис. 340. Пример описания ПОСЛЕ кода контейнера GTM

Подробнее о методе **push ()** читайте на javascript.ru. Описывая данные ДО кода GTM, уровень данных (dataLayer) еще не создан самим Google Tag Manager, а если ПОСЛЕ кода GTM, то уровень данных уже есть и нужно данные в нем дополнить теми, которые вам необходимы, а не создавать новый объект.

- **универсальное описание.** Для того, чтобы упростить работу в дальнейшем и не задаваться вопросом о том, когда вызывать уровень данных (ДО или ПОСЛЕ), используется конструкция следующего вида:

```
<script>
window.dataLayer = window.dataLayer || [];
window.dataLayer.push({'event':'value'});
</script>
```

В режиме предварительного просмотра Google Tag Manager будут зафиксированы сразу два события:

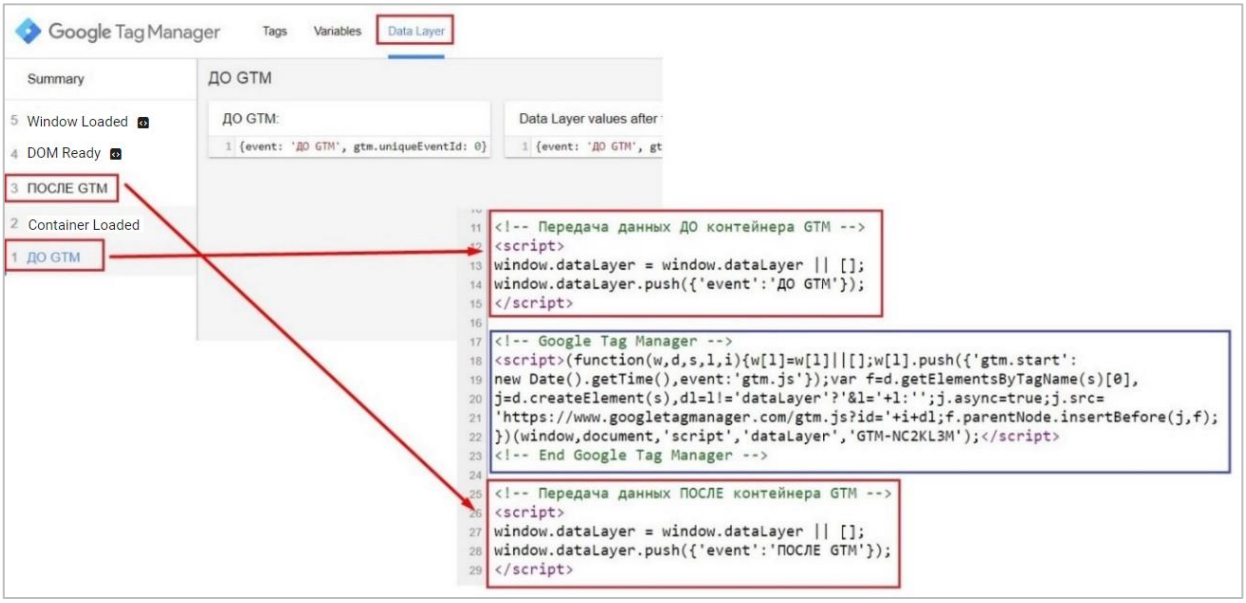


Рис. 341. Универсальное описание уровня данных

С помощью первой строки **window.dataLayer = window.dataLayer || []** мы проверяем, существует ли объект dataLayer. Если он существует, то используем его. В противном случае мы его создаем, и он у нас пустой. Второй строчкой **window.dataLayer.push({'event':'value'})** мы дополняем с помощью метода push () тот объект (массив), который был создан в первой строчке.

Объект **window** сочетает два в одном: глобальный объект JavaScript и окно браузера. При добавлении объекта **window** эта переменная уровня данных также будет доступна из другого скрипта.

Важно отметить, что переопределение переменной уровня данных с тем же именем, что и существующая переменная, приведет к тому, что существующее значение будет перезаписано новым значением.

С помощью уровня данных можно передавать пользовательские события. Давайте разберем простой пример – передать данные в Google Analytics с помощью метода push () по клику на кнопке одной из форм.

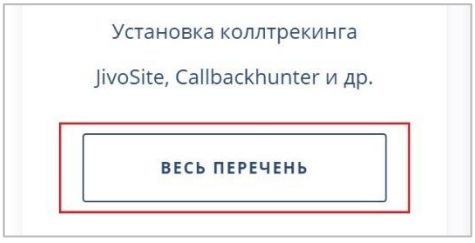


Рис. 342. Кнопка на одной из форм

Если бы вы не использовали GTM в своей работе, а только Google Analytics (код отслеживания Global Site Tag), и библиотеку *gtag.js*, то конструкция в коде страницы выглядела бы так:

```
    </ul>
    <a href="#web-analytics" data-tariff="Веб-аналитика" class="purchase-btn tariff-order fancybox"
    onclick="gtag('event', 'analitika', { 'event_category': 'click', 'event_action': 'knopka', });">
    Весь перечень</a>
</div>
```

Рис. 343. Пример кода реализации конструкцией из Google Analytics

Событие **onclick** возникает при щелчке левой кнопкой мыши на элементе, к которому добавлен атрибут **onclick**. Таким образом, когда пользователь нажимает на кнопку Весь перечень, срабатывает событие, которое передает два обязательных условия (Категория *event_category* и Действие *event_action*) в Google Analytics, тем самым фиксируя цель.

Но когда Google Analytics установлен через GTM, так отслеживать события уже не получится. И поможет нам в этом уровень данных. Данные о событии мы можем передать сначала в dataLayer, а только оттуда - в Analytics.

Добавляем уровень данных к нашему элементу:

```
    <a href="#web-analytics" data-tariff="Веб-аналитика" class="purchase-btn tariff-order fancybox "
     onclick="window.dataLayer.push({'event':'UAevent', 'eventCategory':'click', 'eventAction' :
    'knopka'});">Весь перечень</a>
</div>
```

Рис. 344. Добавление dataLayer.push () к элементу

где **'event': 'UAevent'** – пользовательское событие, а **eventCategory** и **eventAction** – все те же обязательные условия.

Крайне важно: следите за символами кавычек. Копирование из разных мест (из чужих блогов и сайтов) и кодировок недопустимо.

```
    <a href="#web-analytics" data-tariff="Веб-аналитика" class="purchase-btn tariff-order fancybox "
     onclick="window.dataLayer.push({'event':'UAevent', 'eventCategory':'click', 'eventAction' :
    'knopka'});">Весь перечень</a>
</div>
```

Разные кавычки!

Рис. 345. Используйте одинаковые кавычки

В примере выше кавычки имеют разное написание, и событие фиксироваться не будет. Во всей конструкции используйте один тип символов!

Переходим к настройкам в интерфейсе Google Tag Manager. Нам необходимо создать:

1. две переменных уровня данных (**eventCategory** и **eventAction**);
2. триггер **Пользовательское событие** со значением **UAevent**;
3. тег для передачи данных в Google Analytics с типом отслеживания **Событие** и триггером активации **UAevent**;
4. цель в Google Analytics типа **Событие** со значениями категорий/действий = click/knopka соответсвенно.

В GTM на основании передаваемых с помощью dataLayer данных можно создавать переменные и триггеры. Создайте их:

Переходим в **Переменные - Создать пользовательскую переменную**. Выбираем из категории **Переменные страницы - Переменная уровня данных**. Задаем значение eventCategory и такое же имя. Сохраняем.

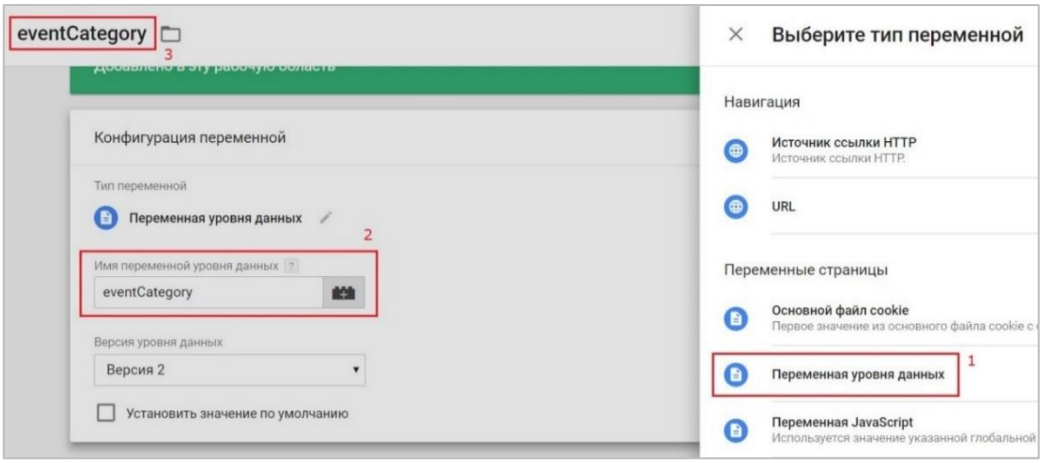


Рис. 346. Переменная уровня данных

Аналогично создаем вторую переменную уровня данных, только с именем **eventAction**. Далее создаем триггер типа Пользовательское событие с тем именем, которое указали в коде страницы у кнопки. В нашем случае – это **UAevent**. Вводим название и сохраняем триггер.

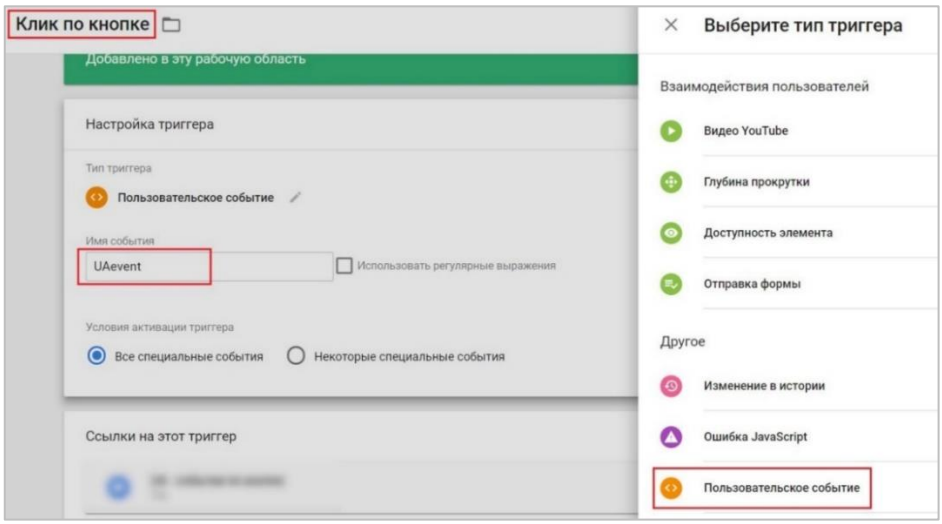


Рис. 347. Пользовательское событие

Теперь создаем тег типа **Universal Analytics** с настройками:

- Тип отслеживания – **Событие**
- Категория – **{{eventCategory}}**
- Действие – **{{eventAction}}**
- Идентификатор отслеживания – наш код отслеживания Google Analytics (можно создать переменную типа **Константа** и добавить туда идентификатор UA)
- Триггер активации – Пользовательское событие **Клик по кнопке.**

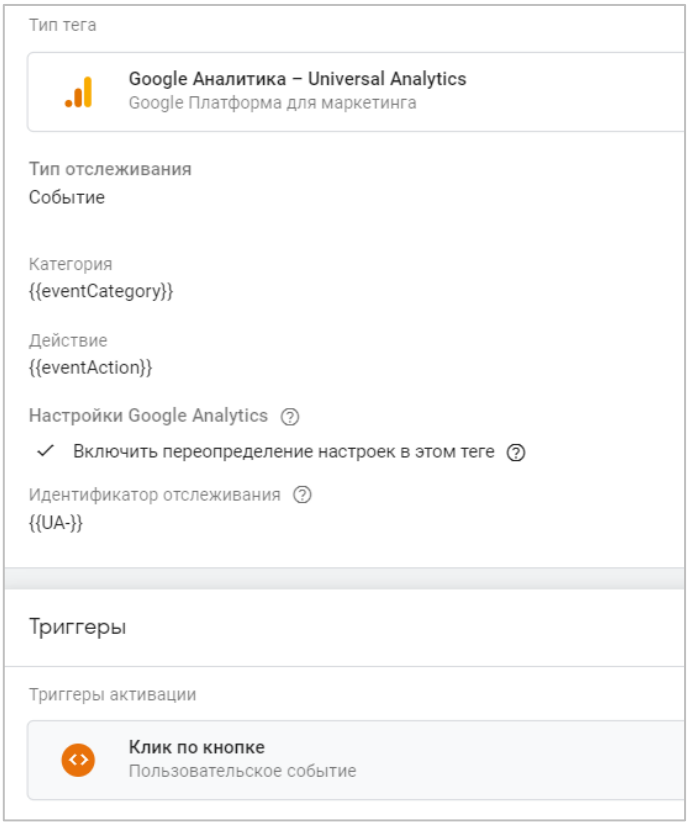


Рис. 348. Настройки тега

На последнем шаге переходим в Google Analytics и создаем цель **Событие** с условиями (те, которые указаны в коде страницы сайта):

- Категория – **click**
- Действие - **knopka**

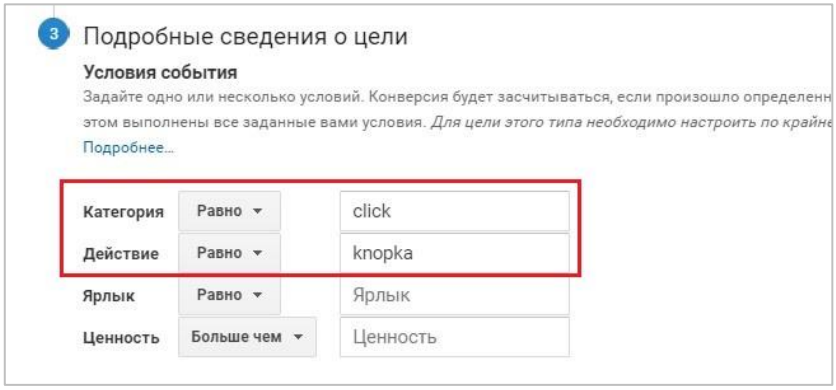


Рис. 349. Настройка события в Google Analytics

Сохраняем цель. На этом настройка передачи события через уровень данных завершена. В режиме предварительного просмотра в GTM по клику на кнопку зафиксируется событие **UAevent** и сработает наш тег.

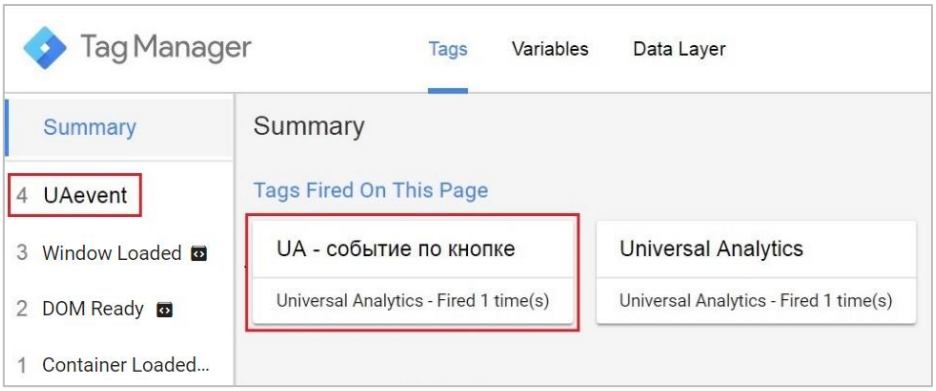


Рис. 350. Событие UAevent в режиме отладки

А с помощью отчетов **В режиме реального времени** в Google Analytics вы можете убедиться в корректности ее настройки и передачи данных.

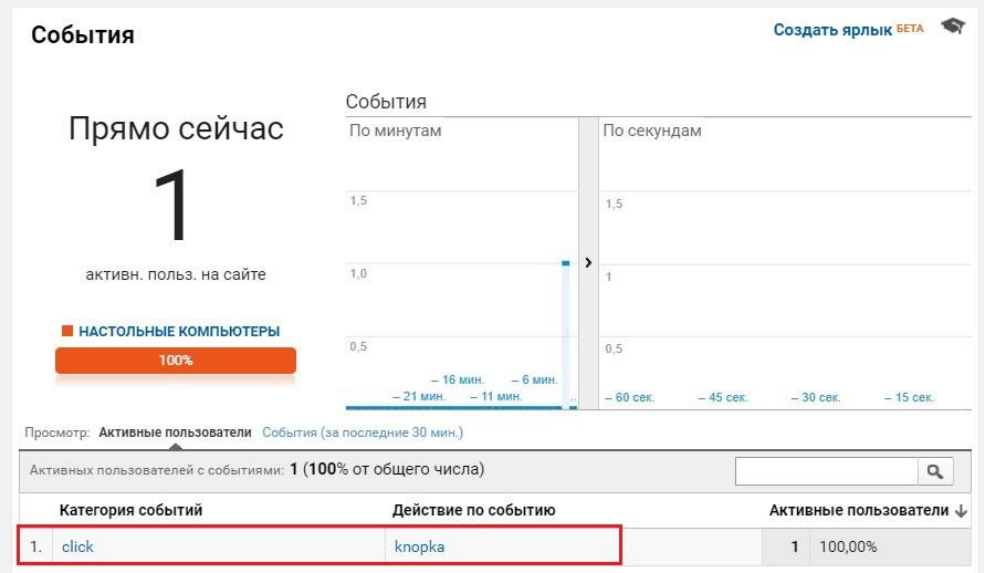


Рис. 351. Отчет В режиме реального времени

Чтобы проверить, что передается в dataLayer, можно воспользоваться консолью в панели инструментов разработчика (в Google Chrome – F12). Вводим dataLayer и нажмите клавишу **Enter**.

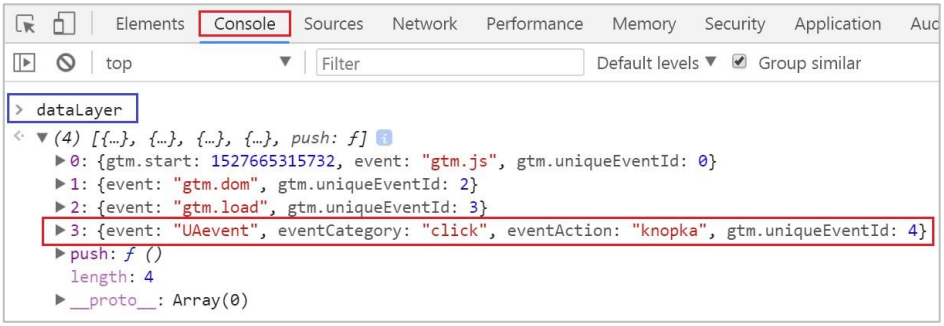


Рис. 352. Console – dataLayer – Enter

Как мы уже знаем, Google Tag Manager по умолчанию передает на уровень данных набор значений **gtm.js** (когда контейнер готов к работе), **gtm.dom** (когда готова модель DOM) и **gtm.load** (когда окно полностью загружено). Последнее по счету событие **UAevent** – это наш клик по кнопке.

Использование уровня данных полезно при отслеживании многих событий на сайте: отправки форм, данных по e-commerce, динамического ремаркетинга, функции User ID и др. Также с помощью уровня данных можно передавать сложные, вложенные структуры. Ярким примером служит передача значений о совершенных транзакциях расширенной электронной торговли. В примере ниже предполагается, что информация о продуктах, отображаемых на странице, известна в момент загрузки страницы:

```
<script>
window.dataLayer = window.dataLayer || [];
dataLayer.push({
 'ecommerce': {
   'currencyCode': 'UAH',
   'impressions': [
    {
      'name': 'Product 1',
      'id': 'ID1',
      'price': '23.5',
      'brand': 'Brand 1',
      'category': 'Category 1/Subcategory 11',
      'variant': 'Variant 1',
      'list': 'List 1',
      'position': 1
    },
    {
      'name': 'Product 2',
      'id': 'ID2',
      'price': '14',
      'brand': 'Brand 2',
      'category': 'Category 2/Subcategory 21',
      'variant': 'Variant 3',
      'list': 'List 1',
      'position': 2
    }]
 },
 'event': 'gtm-ee-event',
 'gtm-ee-event-category': 'Enhanced Ecommerce',
 'gtm-ee-event-action': 'Product Impressions',
 'gtm-ee-event-non-interaction': 'True',
});
</script>
```

Подробнее об этом читайте в официальной справке разработчиков (см. приложение).

Понимание принципов работы и передачи данных с помощью dataLayer существенно упростит вам жизнь при работе с Google Tag Manager. Если вы до конца не разбираетесь в том, как устроен уровень данных, то

реализацию dataLayer лучше всего доверить разработчику или специалисту с соответствующими техническими навыками.

Переменные с точечной нотацией

Обратиться к свойствам объекта в JavaScript можно несколькими способами: с помощью *точечной нотации (dot notation)* или *скобочной нотации (bracket notation)*. Но зачем это знать веб-аналитику? Постараемся разобраться.

Работая с GTM, вы можете столкнуться с такими задачами:

- извлечь данные какого-либо объекта из уже сформированного уровня данных (dataLayer), например, электронной торговли или динамического ремаркетинга;
- получить данные из полей форм в момент отправки заявки.

Зачем это нужно? Например, чтобы передавать эти данные в Google Analytics в качестве пользовательских метрик для дальнейшего анализа и сегментации пользователей.

Как это сделать? Чтобы вы лучше понимали, о чем пойдет речь в этой статье, рекомендую прочитать материал Отслеживание выбранного элемента из выпадающего списка в GTM.

Причем же здесь переменные и точечная нотация? Постараюсь подробно пояснить на следующих примерах. Когда мы работаем в JavaScript с массивами, мы можем хранить в них различные элементы и объекты.

Например, у нас есть такой массив, состоящий из 5 элементов с различными типами данных:

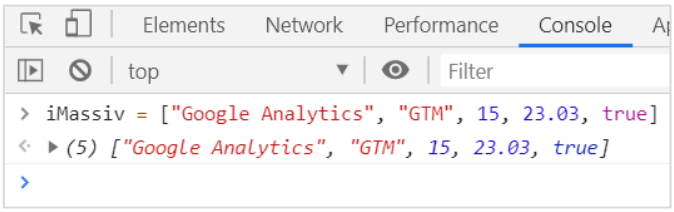


Рис. 353. Массив элементов

И мы хотим получить из iMassiv второй элемент со значением “GTM”. Для этого можно указать его номер в квадратных скобках:

```
iMassiv[1] // "GTM"
```

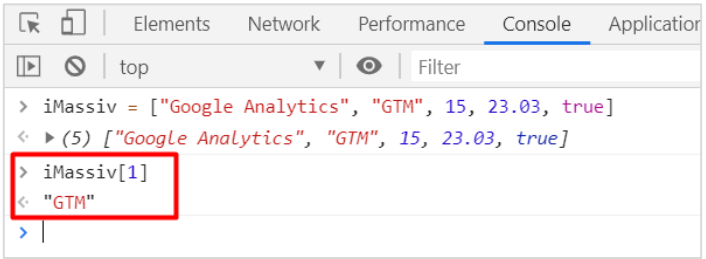


Рис. 354. Полученное значение "GTM"

Элементы массива нумеруются, начиная с 0. Поэтому GTM – 1 элемент в массиве, 0 – Google Analytics, 2 - 15, 3 – 23.03, 4 – true.

Давайте усложним пример. Вместо элементов будем работать с объектами. Они используются для хранения коллекций различных значений и более сложных сущностей. Объект может быть создан с помощью фигурных скобок **{ }** с необязательным списком свойств. Свойство — это пара *ключ: значение*, где:

- **ключ** – это строка (также называемая именем свойства),
- **значение** – что угодно.

Немного изменив массив, получим:

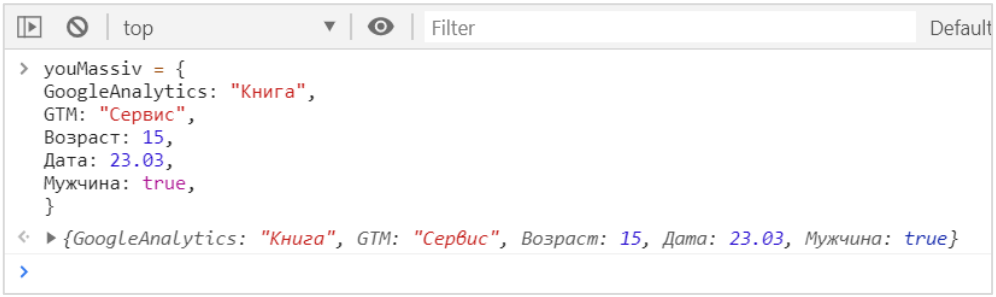


Рис. 355. Массив объектов

Напоминает конструкцию уровня данных (dataLayer)? Что-то похожее вы наверняка видели при настройке электронной торговли и динамического ремаркетинга. Вот как выглядит dataLayer для интернет-магазина с типом с объектами, передаваемыми в момент показов товаров **Показы (Impressions)**.

Рис. 356. Событие gtm-ee-event электронной торговли

Такая же вложенная структура, только пар ключ-значение гораздо больше. Для обращения к свойствам объекта используется запись через точку:

```
youMassiv.GTM // "Сервис"
```

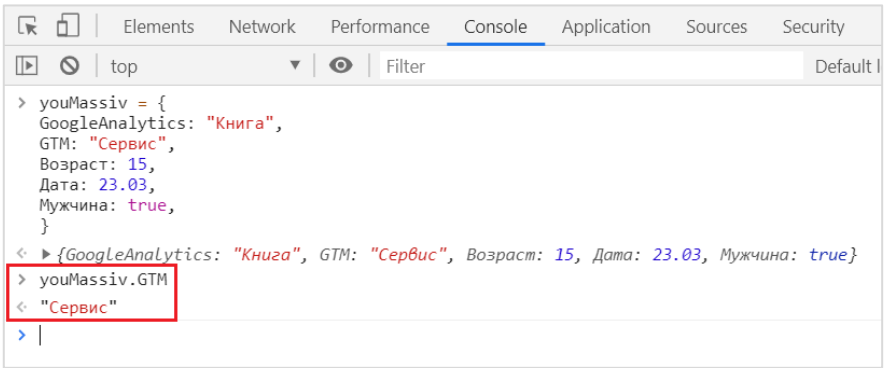


Рис. 357. Полученный результат "Сервис"

Это и есть точечная запись (**Точечная нотация, Dot notation**). Поскольку у меня единственный уровень вложенности, то используется одна точка. Но если нам нужно добавить еще одно свойство со значением внутри объекта, а потом вывести значение этого свойства, то будет использоваться уже две точки:

```
myMassiv.GTM.Сервис // "Google"
```

Рис. 358. Пример точечной нотации

Есть два варианта составления:

1. в консоли разработчика написать команду dataLayer, найти необходимое событие из списка сработанных, спуститься на нужный уровень и записать итоговое значение конкретной переменной;

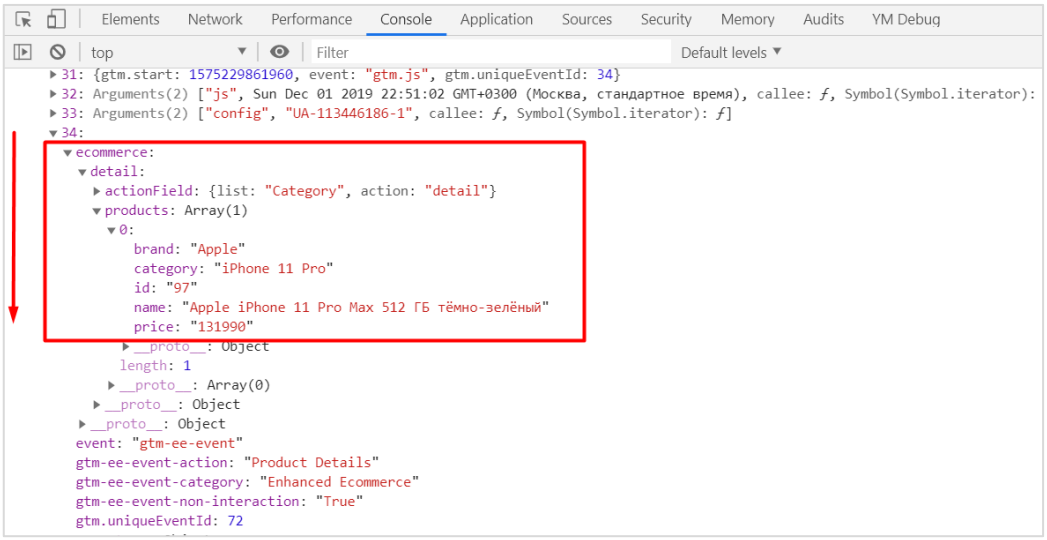


Рис. 359. Консоль разработчика

2. можно воспользоваться расширением **Datalayer Checker** для браузера Google Chrome.

Это самый простой способ отладки и проверки реализаций dataLayer без использования консоли разработчика. Datalayer Checker быстро покажет все необходимые свойства объектов в нужном формате (FLAT & JSON):

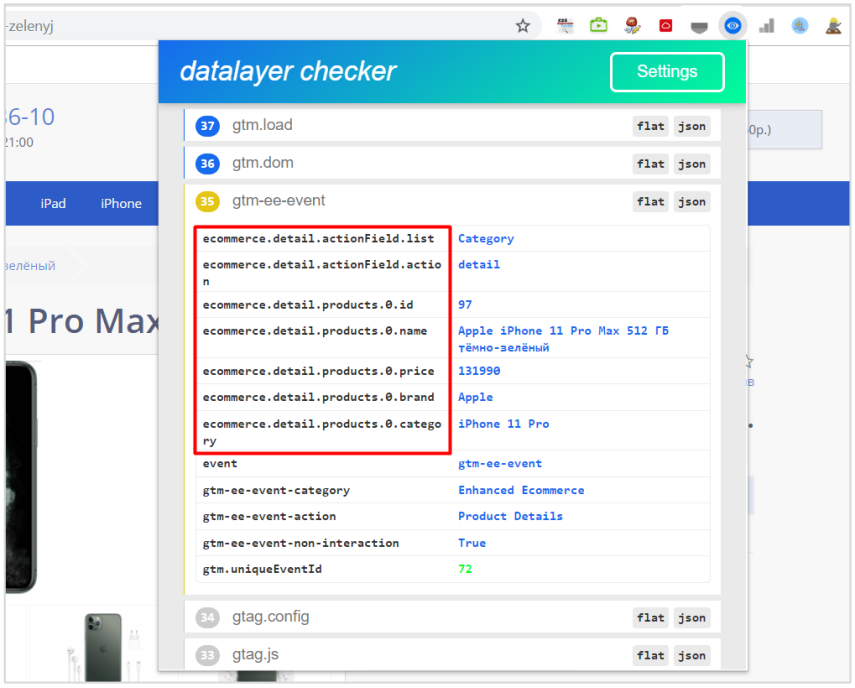


Рис. 360. Готовые переменные с точечной нотацией в Datalayer Checker

Останется только скопировать и вставить ключ в Google Tag Manager для пользовательской переменной типа **Переменная уровня данных**.

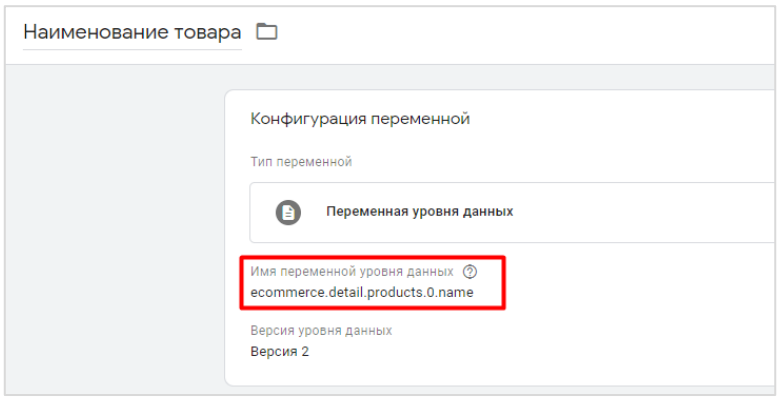


Рис. 361. Пример переменной типа Переменная уровня данных

В примере выше я хочу, чтобы в переменной выводилось наименование просматриваемого товара.

В Google Tag Manager можно определить интерпретацию точек в имени переменной. Делается это с помощью версии уровня данных:

Версия 1. Разрешить точки в названиях ключей. Например, в выражении dataLayer.push('a.b.c': 'value') название ключа будет интерпретировано как a.b.c (т. е. {'a.b.c': 'value'}).

Версия 2. Интерпретировать значения с точками как иерархические значения. В этом случае выражение dataLayer.push({'a.b.c': 'value'}) будет интерпретировано как трехуровневая иерархия: {a: {b: {c: 'value'}}}.

Поскольку в документации Google (и Яндекса тоже) речь идет о вложенной иерархии, то при настройке переменных выбирают именно версию 2.

Метрика

```
dataLayer.push({
    "ecommerce": {
        "purchase": {
            "actionField": {
                "id" : "TRX987"
            },
            "products": [
                {
                    "id": "25341",
                    "name": "Толстовка Яндекс мужская",
                    "price": 1345.26,
                    "brand": "Яндекс / Яndex",
                    "category": "Одежда/Мужская одежда/Толстовки и свитшоты",
                    "variant": "Оранжевый цвет"
                },
                {
                    "id": "25314",
                    "name": "Толстовка Яндекс женская",
                    "price": 1543.62,
                    "brand": "Яндекс / Яndex",
                    "category": "Одежда/Женская одежда/Толстовки и свитшоты",
                    "variant": "Белый цвет",
                    "quantity": 3
                }
            ]
        }
    }
});
```

Analytics

```
dataLayer.push({
  'ecommerce': {
    'purchase': {
      'actionField': {
        'id': 'T12345',
        'affiliation': 'Online Store',
        'revenue': '35.43',
        'tax':'4.90',
        'shipping': '5.99',
        'coupon': 'SUMMER_SALE'
      },
      'products': [{
        'name': 'Triblend Android T-Shirt',
        'id': '12345',
        'price': '15.25',
        'brand': 'Google',
        'category': 'Apparel',
        'variant': 'Gray',
        'quantity': 1,
        'coupon': ''
       },
       {
        'name': 'Donut Friday Scented T-Shirt',
        'id': '67890',
        'price': '33.75',
        'brand': 'Google',
        'category': 'Apparel',
        'variant': 'Black',
        'quantity': 1
       }]
    }
  }
});
```

Рис. 362. Уровень данных для Яндекс.Метрики и Google Analytics (e-commerce)

Вернемся к примеру. Результатом настроенной переменной будет правильно определенное значение:

Рис. 363. Результат извлечения

Аналогично можно извлечь значения всех других переменных (бренда, цены, категории, ID заказа и т.д.).

Отлично! Мы сделали это. Но бывают задачи более сложного уровня. Например, как отслеживание выбранного элемента из выпадающего списка в GTM (статья, что рекомендовал прочитать в самом начале). Эта информация существует в dataLayer как выбранное значение из этого списка, и для его извлечения вам нужно сначала его найти, записать путь и после передать в Google Tag Manager в качестве пользовательской переменной. А как быть с теми полями и данными, которые формируются и отправляются в момент совершения какого-либо события? Например, пользователь заполнил заявку на вашем сайте и отправил форму. dataLayer здесь не формируется, а в роли элементов являются те самые выпадающие списки, input, textarea и другие элементы форм. А отслеживать их очень хочется. Поиск вложенности и пути до переменной отнимает много времени.

На помощь приходит другое расширение браузера для Google Chrome, которое называется **GTM dataLayer Sifter**. Установить его можно по ссылке (см. приложение).

В правом верхнем углу окна браузера появится иконка с приложением. Что умеет делать это расширение? Плагин предназначен для поиска и определения точечной нотации от элемента конкретного события по отношению к другому элементу на странице, а также извлечения значения элемента DOM (например, для поля формы) на уровне данных, которое затем можно использовать в GTM в качестве переменной.

Давайте попробуем извлечь значение поля Имя в форме на сайте graphanalytics.ru в момент ее отправки. Нам необходимо передать значение, которое заполнил пользователь, в Google Analytics.

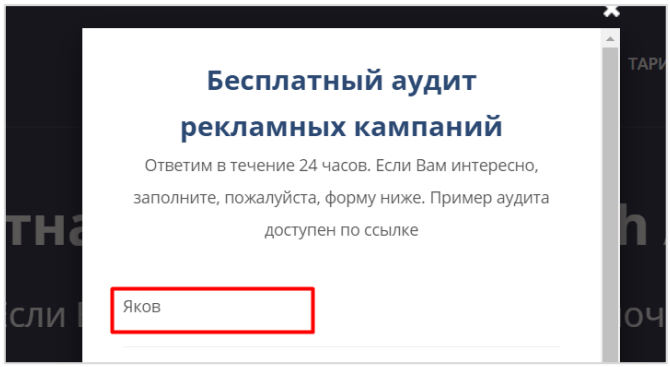


Рис. 364. Пример извлечения данных из поля Имя

В моем блоге есть статья на эту тему. Там это было реализовано оптимальным способом через пользовательскую переменную типа Собственный код JavaScript. В этом примере будем использовать готовое решение **GTM dataLayer Sifter**.

Я для этого зайду на сайт, открою форму и заполню поле Имя. Затем необходимо открыть консоль разработчика (клавиша F12 в Google Chrome). Далее заполняю все поля формы и делаю **тестовую заявку (1)**, чтобы сработало наше событие gtm.formSubmit (триггер **Отправка формы**).

После этого я перехожу на вкладку **GTM dataLayer Sifter (2)** и выделяю то событие, относительно которого хочу получить значение поля Имя. В моем случае – это **gtm.formSubmit (3)**.

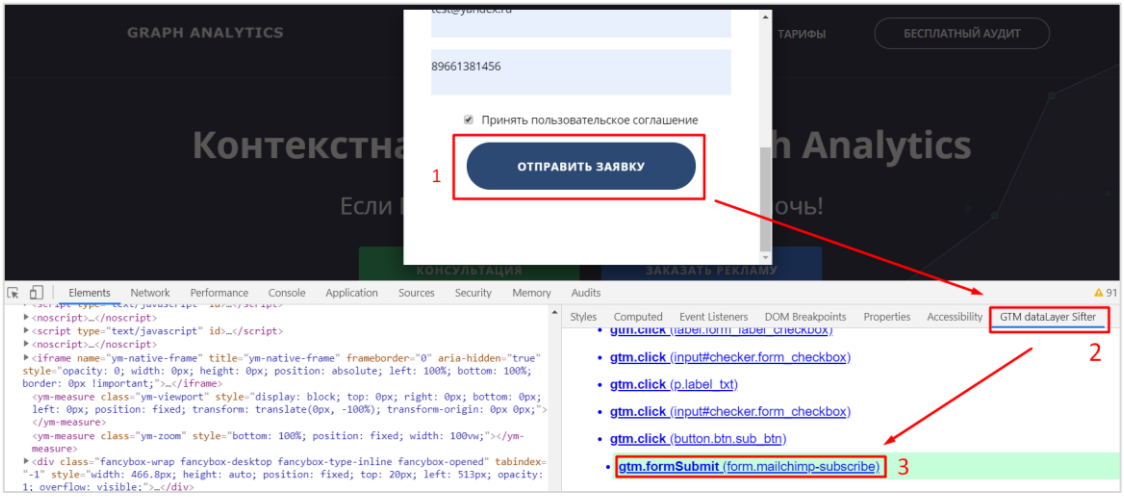


Рис. 365. Последовательность действий

После этого я могу прослушать необходимое мне поле и получить значение переменной с точечной нотацией. Для этого я перехожу на вкладку **Elements** и исследую элемент Имя (4). Выделяю его (5) и справа под выделенным событием в GTM dataLayer Sifter в графе **Results** я получаю 3 заполненных строчки:

- Path to element (Путь к элементу);
- Path to Value (Путь к значению);
- Path to Text (Путь к тексту элемента).

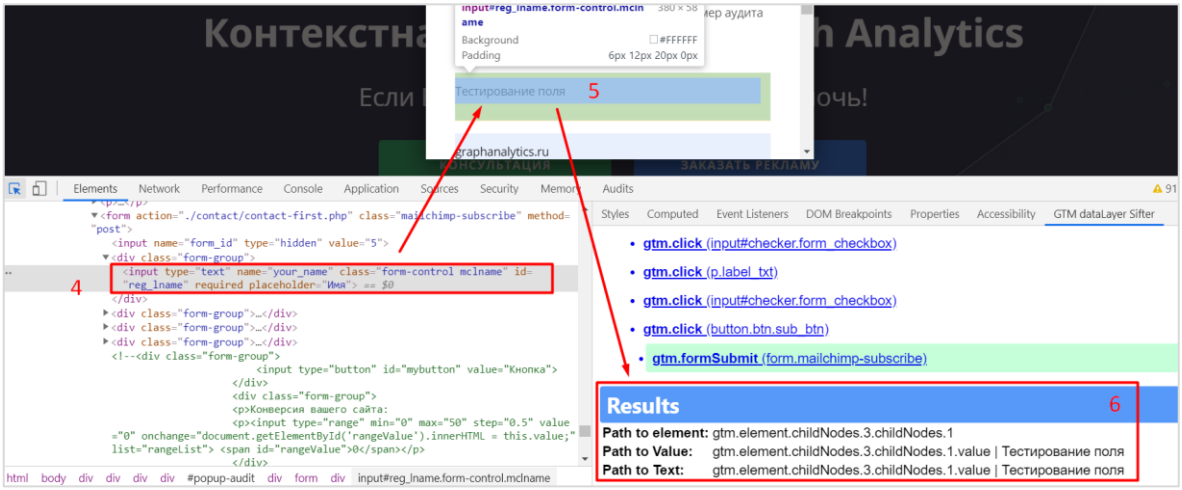


Рис. 366. GTM dataLayer Sifter - Results

Копируем то, что в строчке Path to Value. Далее переходим в Google Tag Manager и создаем переменную типа Переменная уровня данных. Вставляем туда полученное значение. Сохраняем.

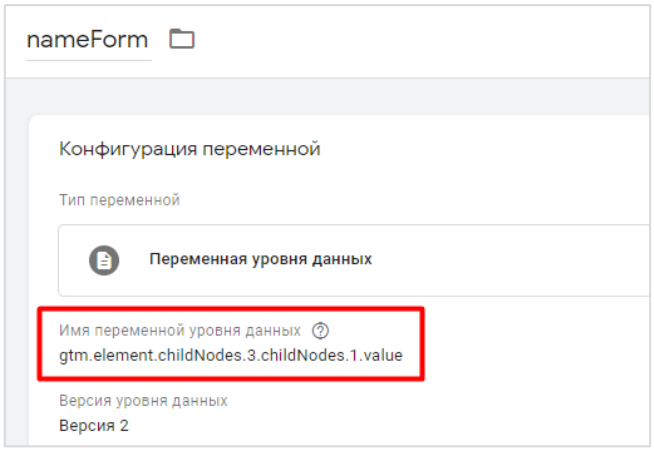


Рис. 367. Полученная переменная уровня данных

Теперь идем в наш тег и в качестве дополнительного поля в ярлыке события (пример!) я выберу эту переменную.

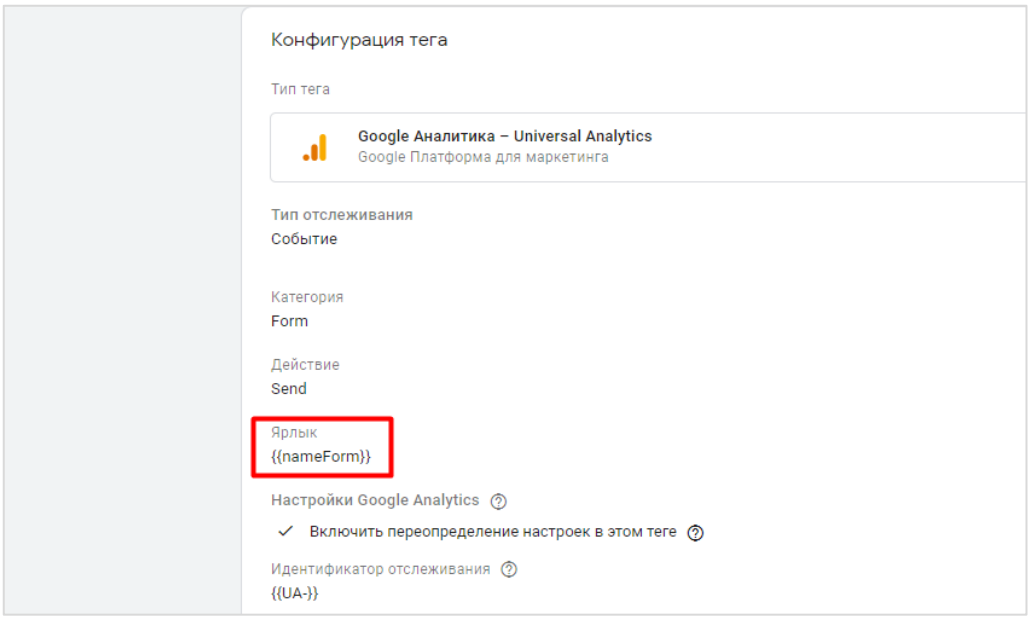


Рис. 368. Передача значения переменной в ярлыке события

Вы можете передавать значения полей в специальных параметрах и показателях. Однако не забывайте: загрузка данных, которая может идентифицировать личность пользователя, запрещена политикой Google.

Сохраняем тег. Для проверки корректности передачи данных можно воспользоваться режимом предварительного просмотра GTM.

Шаблоны переменных

В середине 2019 года в диспетчере тегов Google появилась возможность использования пользовательских шаблонов (**custom templates**).

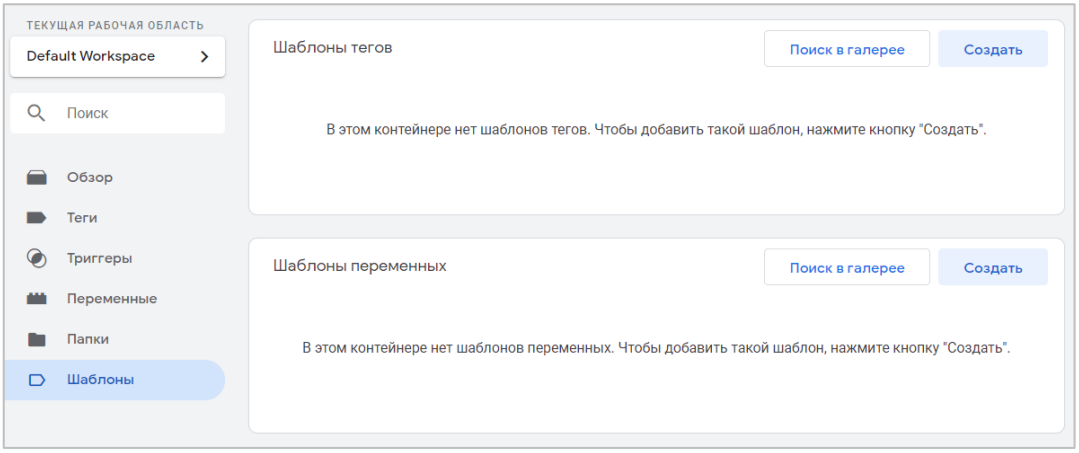


Рис. 369. Шаблоны тегов и переменных

Это набор функций, позволяющих компаниям или отдельным специалистам создавать и делиться своими собственными решениями (тегами и переменными) с другими пользователями благодаря галереи шаблонов сообщества Google Tag Manager. Пользовательские шаблоны используют **изолированный JavaScript** (см. приложение), который обеспечивает безопасное выполнение за счет ограничения и отключения некоторых функций.

Используя custom templates, создавать теги и переменные стало проще и безопаснее, чем работая с пользовательскими HTML-тегами и переменными JavaScript, поскольку для этого не требуются каких-либо знаний и навыков в области программирования. Вы просто добавляете информацию в определенные поля шаблона, активируете необходимые опции с помощью раскрывающихся списков, проставляете галочки напротив нужных настроек – и все!

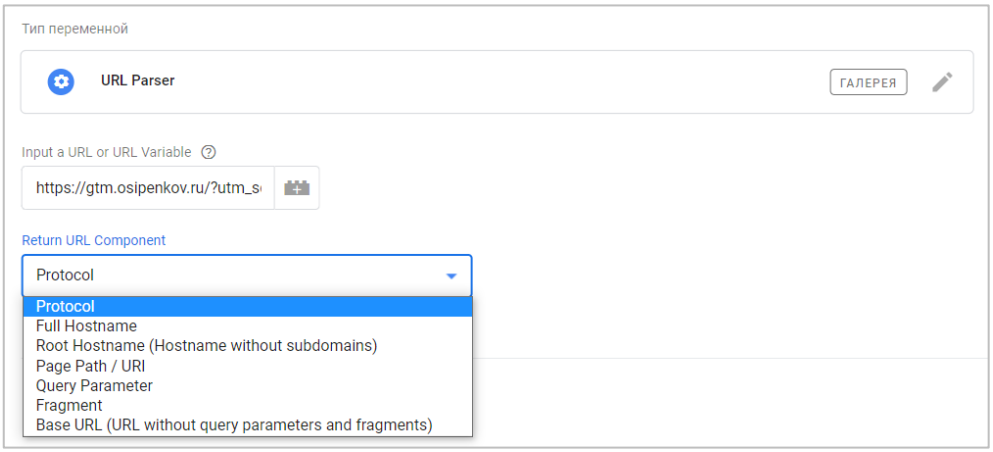


Рис. 370. Встроенные опции в шаблоне переменной URL Parser

Самым большим преимуществом использования пользовательских шаблонов является снижение рисков «положить» сайт. Когда вы используете пользовательские переменные и HTML-теги, и добавляете в них коды, скопированные с других сайтов, и о работе которых вы не имеете ни малейшего представления, очень сильно возрастают шансы что-нибудь сломать. Потому что код, который был описан в материале, мог предназначаться

конкретно для этого сайта и конкретно для этой задачи. Пользовательские шаблоны, в этом плане, надежное решение!

Минус тоже есть – это язык пользовательского шаблона. Как правило, решения сторонних разработчиков имеют английский интерфейс. Поэтому для тех, кто не обладает знаниями языка, использование шаблонов тегов или переменных может показаться сложной задачей.

Если вы хотите подробнее познакомиться с пользовательскими шаблонами, попробовать создать свое решение, то используйте пошаговое руководство (на английском языке) от Симо Ахавы (см. приложение). В нем он подробно описал весь процесс создания и настройки тегов и переменных.

Добавить переменную стороннего разработчика в свою рабочую область GTM можно двумя способами:

1. через вкладку **Шаблоны** и **Поиск в галерее**

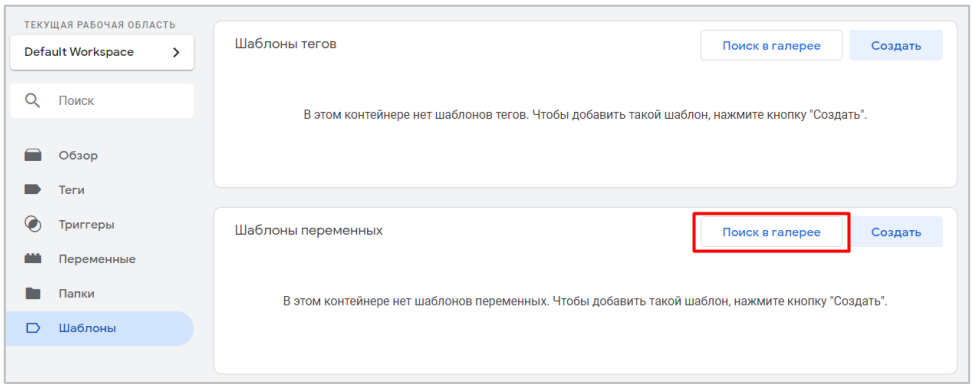


Рис. 371. Шаблоны переменных – Поиск в галерее

2. через вкладку **Переменные**, создание пользовательской переменной. В открывшемся меню необходимо кликнуть на самое верхнее уведомление:

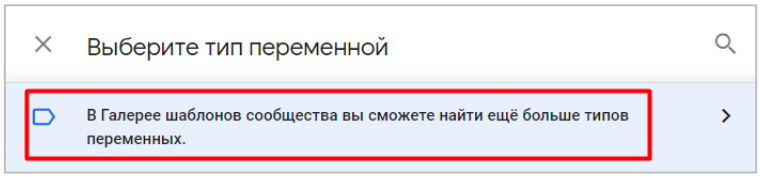


Рис. 372. Создание переменной через уведомление о галереи шаблонов

В открывшемся списке вы увидите большое количество шаблонов переменных как для сторонних сервисов, так и предназначенных для упрощения настройки различных отслеживаний.

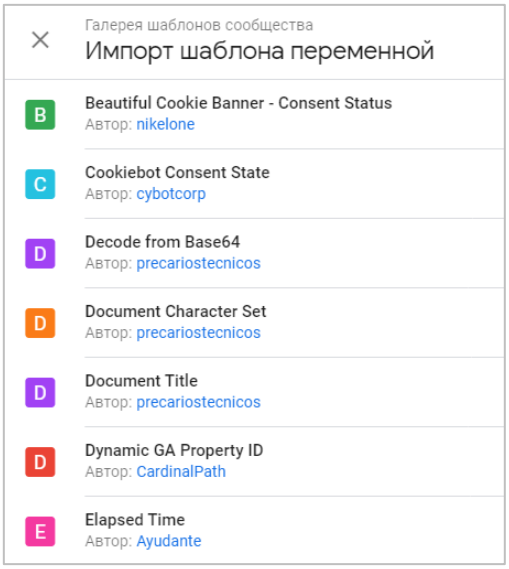


Рис. 373. Импорт шаблона переменной

После того, как вы добавите шаблон переменной в свою рабочую область, сама переменная будет отображаться в разделе **Шаблоны**, а также в списке пользовательских переменных в блоке **Пользовательские шаблоны** с соответствующим значком **ГАЛЕРЕЯ**:

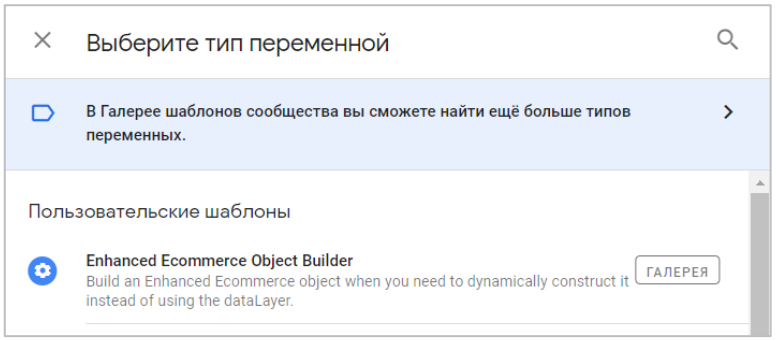


Рис. 374. Пользовательские шаблоны

Подробнее о том, как и какие шаблоны тегов и переменных можно использовать в своей работе, мы разберем в последующих главах.

Глава 5

Триггеры

Триггер в Google Tag Manager (от анг. слова trigger в значении спусковой крючок, или приводящий в действие элемент) – это условие, при котором активируется или блокируется тег. Для запуска тега должен быть хотя бы один триггер. Таким образом, нельзя создать тег без триггера. Триггер может принимать значение *true* (истина), либо *false* (ложь). Он выполняется только в том случае, когда его значение является истинным, и, если выполняются все условия триггера.

А чтобы это определить (true или false), значение переменной сравнивается с тем, которое задано в триггере. Все триггеры в Google Tag Manager связаны с событиями. Также они могут иметь дополнительные условия или фильтры. В старой версии диспетчера тегов Google триггер назывался **правилом**.

Существует два способа создания триггеров:

1. перейти на вкладку **Триггеры** и нажать на кнопку **Создать**;

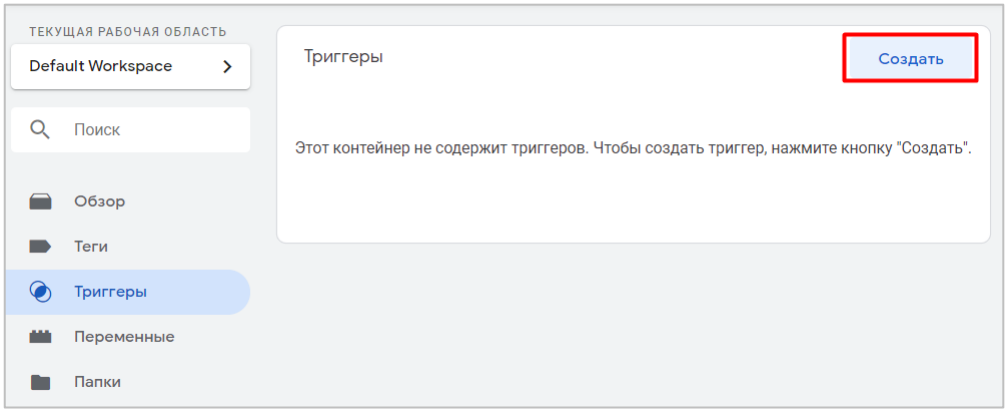


Рис. 375. Создание триггера

2. через сам тег в блоке **Триггеры** по нажатию на **+** если в нем уже есть триггеры;

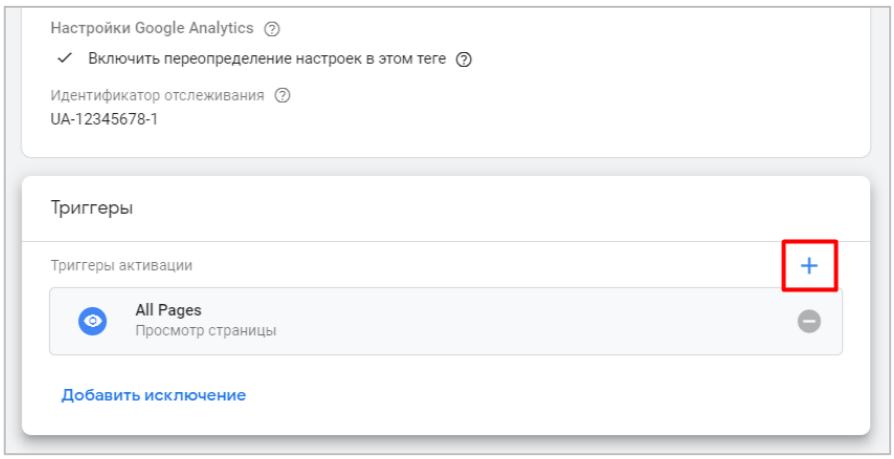


Рис. 376. Создание триггера

или же через пустую область блока, если триггеры к тегу еще не были добавлены.

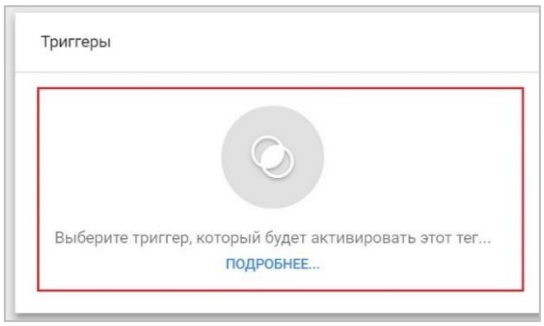


Рис. 377. Создание триггера

Тип триггера представляет собой комбинацию события GTM и типа взаимодействия (или события), которое вы хотите, чтобы GTM начал прослушивать. Они разбиты на несколько категорий:

- Просмотр страницы;
- Клик;
- Взаимодействие пользователей;
- Другое;

Разберем каждый из них последовательно. Но перед тем, как создавать новый триггер, вы должны четко осознать, какое событие GTM должно запустить ваш тег.

Просмотр страницы

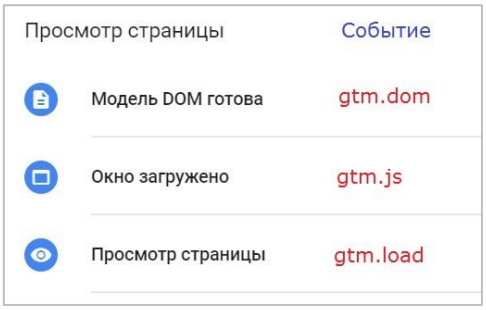


Рис. 378. Триггеры типа Просмотры страницы

- **Модель DOM готова (DOM Ready)** - передается gtm.dom событие, как только браузер загрузил объектную модель документа;
- **Окно загружено (Window Loaded)** – передается gtm.load событие после завершения загрузки всей страницы, когда окно полностью загружено;
- **Просмотр страницы (Container Loaded)** – передается gtm.js событие, когда контейнер GTM готов к работе.

Google Tag Manager передает эти три события на уровень данных по умолчанию. Убедиться в этом можно перейдя в режим предварительного просмотра.

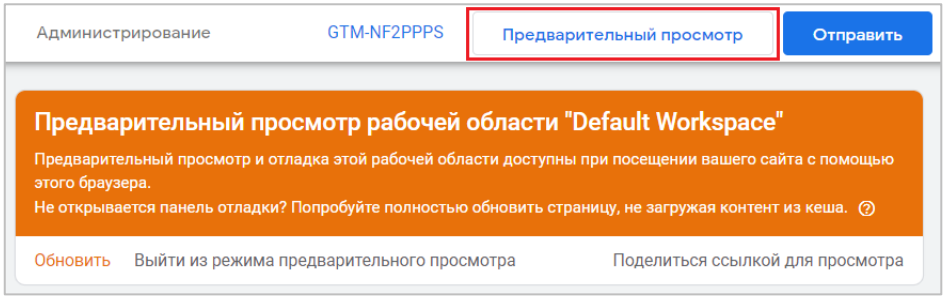


Рис. 379. Предварительный просмотр

Перейдя на сайт в режиме отладки, вы увидите:

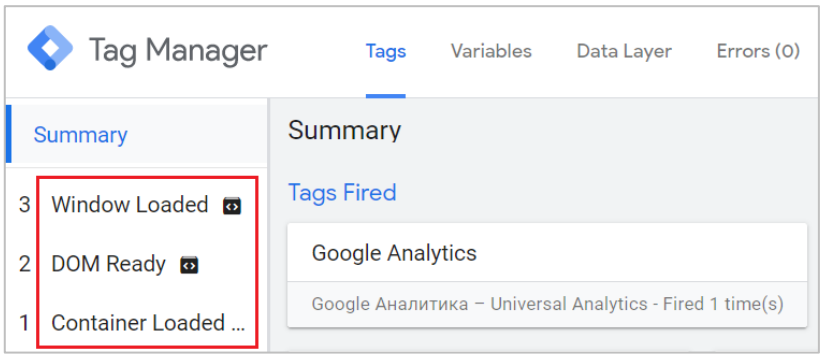


Рис. 380. Режим отладки – Три события по умолчанию

Выбрав событие, например, **Window Loaded**, и перейдя на вкладку **Variables (Переменные)**, можно просмотреть зафиксированное пользовательское событие (переменная **_event**) и его значение (value – в нашем примере gtm.load):

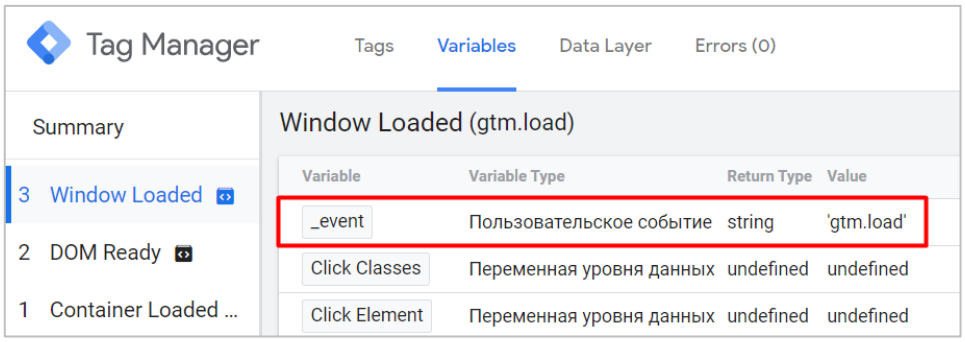


Рис. 381. Пользовательское событие

или у переменной **Пользовательское событие**:

Рис. 382. Переменная Пользовательское событие

Клик

Тип триггера фиксирует клики по ссылкам или любым кликабельным элементам сайта.

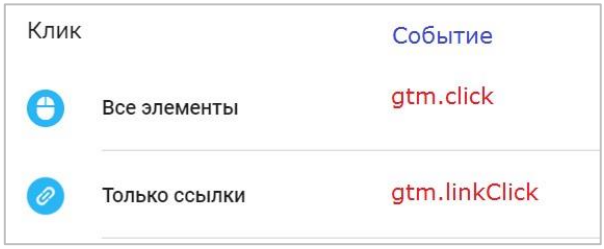


Рис. 383. Триггеры типа Клики

- **Все элементы** - передается gtm.click событие когда на странице щелкнут любой элемент;
- **Только ссылки** – передается gtm.linkClick событие при нажатии на тег <a> элемента HTML.

Например, создайте триггер на **Все элементы** с условием активации триггера **Все клики**.

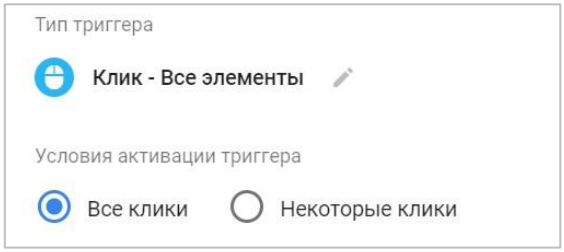


Рис. 384. Тип триггера – Клик – Все элементы

В таком случае при каждом клике по любой области на странице в режиме предварительного просмотра GTM будет фиксировать событие **Click (gtm.click)**:

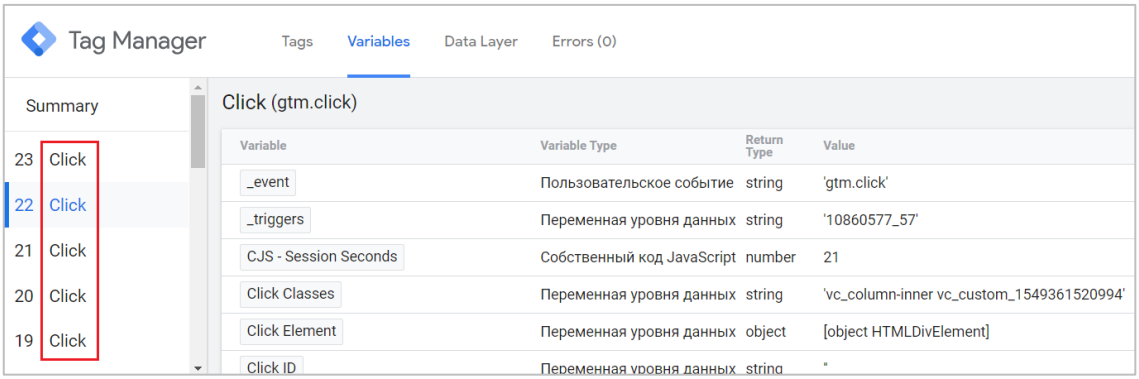


Рис. 385. Фиксация кликов в режиме отладки

То есть вне зависимости от того, куда пользователь кликнул (на кнопку, на фон, в пустую область на странице) – все события будут зафиксированы. Если же мы хотим отслеживать события только по определенным элементам, то вместо условия активации триггера **Все клики** выбираем **Некоторые клики**:

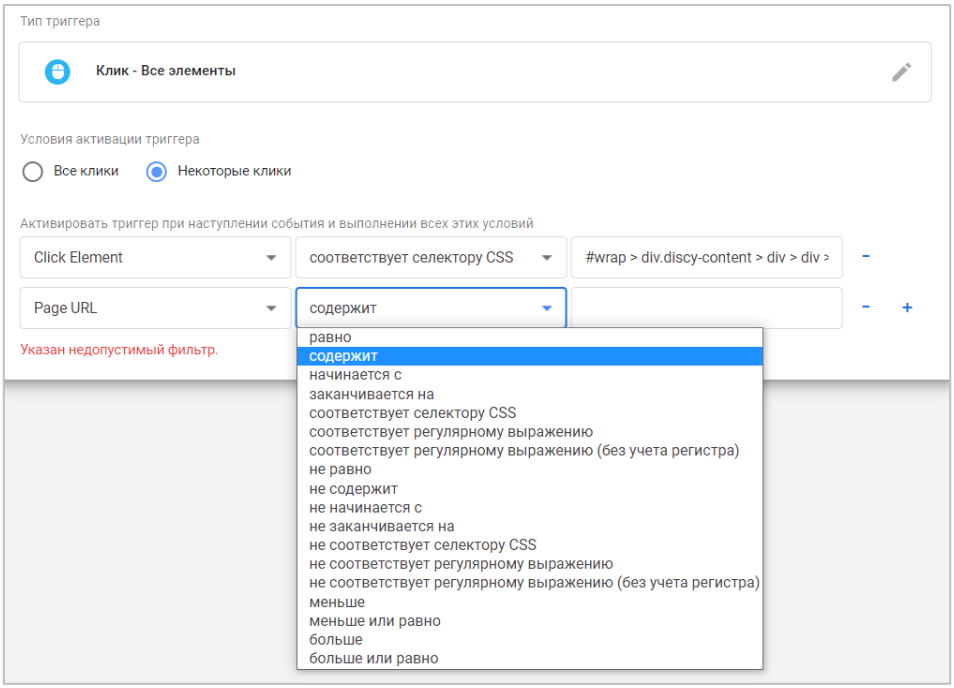


Рис. 386. Клики по определенным элементам

Нам станет доступна настройка активации триггера при наступлении события и выполнении всех условий, а именно:

1. сначала задается переменная (на примере выше – **Click Element**);
2. затем выбирается оператор из списка (равно, содержит, соответствует селектору CSS, регулярному выражению и т.д.);

3. задается само значение (на изображении выше – CSS селектор одного элемента на сайте).

При необходимости можно задать несколько фильтров с помощью значка **+**, либо же удалить ненужное с помощью **-**.

Рис. 387. Добавление условий через +/-

Условия и фильтры связаны между собой логическим **И**. То есть задав две строки, триггер будет срабатывать только тогда (событие будет считаться истинным и принимать значение true), когда одновременно соблюдаются первое **И** второе условия.

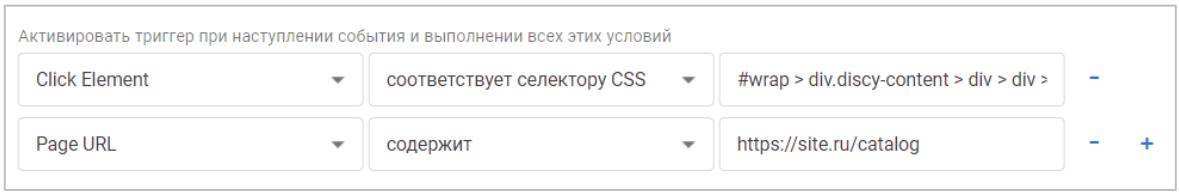


Рис. 388. Логическое И

В этом примере триггер сработает только тогда, когда пользователь кликнет по элементу с соответствующим селектором *#features1 > div > ul > li:nth...* **И** только на странице *https://site.ru/catalog*.

Важно: gtm.click — это предустановленные события GTM, они срабатывают при любом клике вне зависимости от того, был ли он осуществлен по некоторым элементам (при выборе такого) или же по всем элементам страницы. Главное, чтобы на нецелевых событиях не срабатывал тег, который активируется соответствующим триггером.

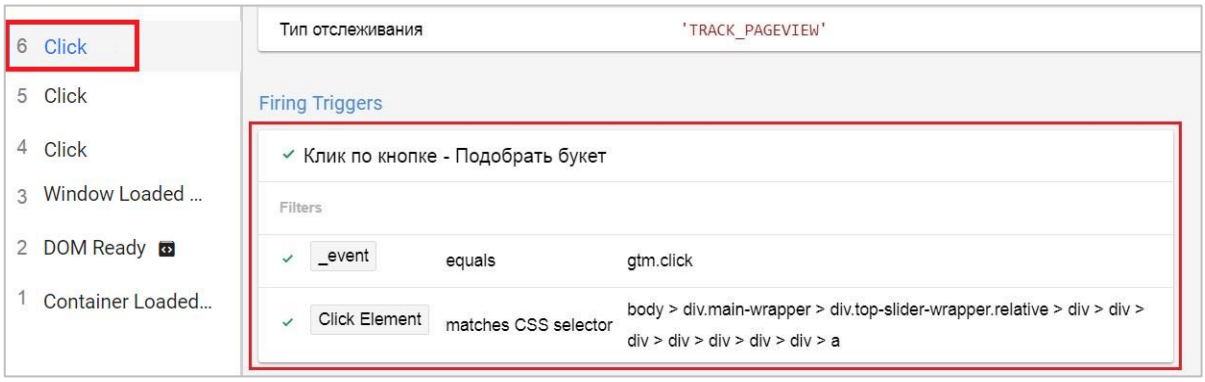


Рис. 389. Соблюдение условий активации триггера

Аналогичным образом настраиваются триггеры для **Только ссылки**, а событие в режиме отладки будет называться **gtm.linkClick**:

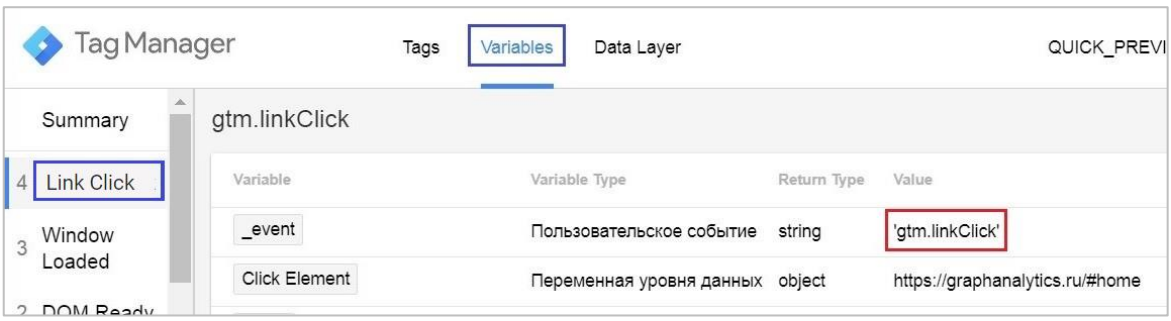


Рис. 390. Событие gtm.linkClick

Однако данный тип триггера имеет дополнительные настройки. Такие же опции появляются при создании триггера **Отправка формы**.

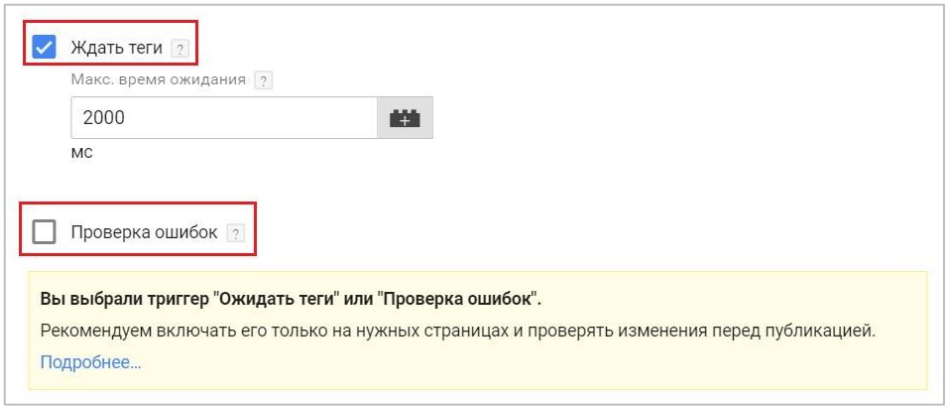


Рис. 391. Опции Ждать теги и Проверка ошибок

- **Ждать теги** – опция блокирует действие (задержка открытия ссылок или отправки формы) до активации всех тегов в контейнере GTM, то есть пока все теги не загрузятся или пока не истечет указанное время ожидания (смотря что произойдет раньше), событие зафиксировано не будет. Задается в миллисекундах (2000 мс = 2 секунды);
- **Проверка ошибок** – функция блокирует активацию тега, если для отслеживаемого элемента ссылки/формы не было выполнено действие по умолчанию (**Клик по ссылке** или **Отправка формы**). Опция отвечает за то, на каких страницах нам необходимо использовать данный триггер.

Блок **Проверка ошибок** позволяет нам указать **ГДЕ** мы хотим использовать данный триггер, а блок с условиями активации определяет **КОГДА** этому быть. В качестве примера давайте разберем условие клика по ссылке (элемент – кнопка) со значением CSS-селектора и зададим **Проверка ошибок** на главной странице сайта.

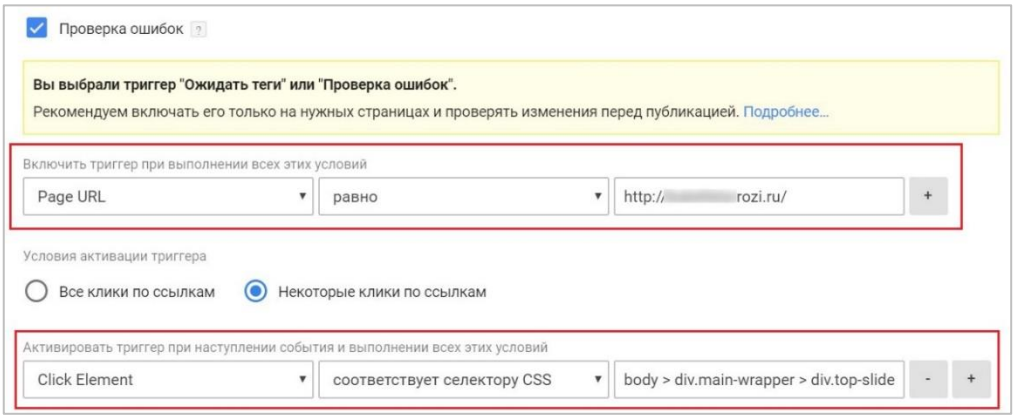


Рис. 392. КОГДА и ГДЕ

В режиме предварительного просмотра при клике на данную кнопку зафиксируется событие **Link Click (gtm.linkClick).** Но если мы поменяем условие проверки ошибок, например, на такое:

Рис. 393. Пример условия проверки ошибок

То события gtm.linkClick фиксироваться не будут, поскольку не соблюдено условие **ГДЕ**. До обновления Google Tag Manager **Проверка ошибок** и условие активации назывались **Условия включения** и **Условия активации**. Рядовой пользователь мог легко запутаться в данной терминологии.

Для триггера типа **Отправка формы** распространенным условием активации функции **Проверка ошибок** является указание только тех страниц, на которых есть формы.

Триггер можно использовать со специально встроенными переменными: Click Element, Click Classes, Click ID, Click Target, Click URL, Click Text.

Взаимодействие пользователей

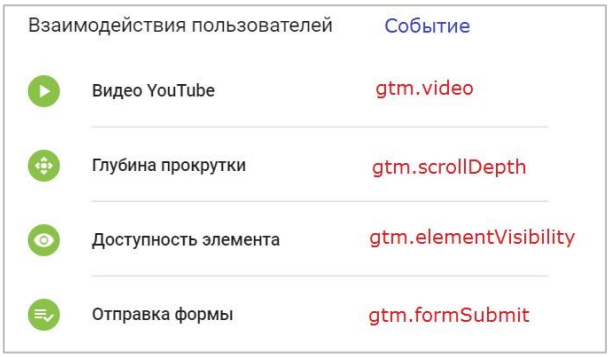


Рис. 394. Триггеры типа Взаимодействие пользователей

- **Видео YouTube** – срабатывает событие gtm.video когда пользователь просматривает видео на сайте (доступен только сервис-поставщик YouTube);
- **Глубина прокрутки** – срабатывает событие gtm.scrollDepth в том случае, когда пользователь прокручивает страницу (как горизонтально, так и вертикально);
- **Доступность элемента** - срабатывает событие gtm.elementVisibility когда элемент становится видимым на странице;
- **Отправка формы** – срабатывает тег gtm.formSubmit в случае, если форма была отправлена.

Видео YouTube

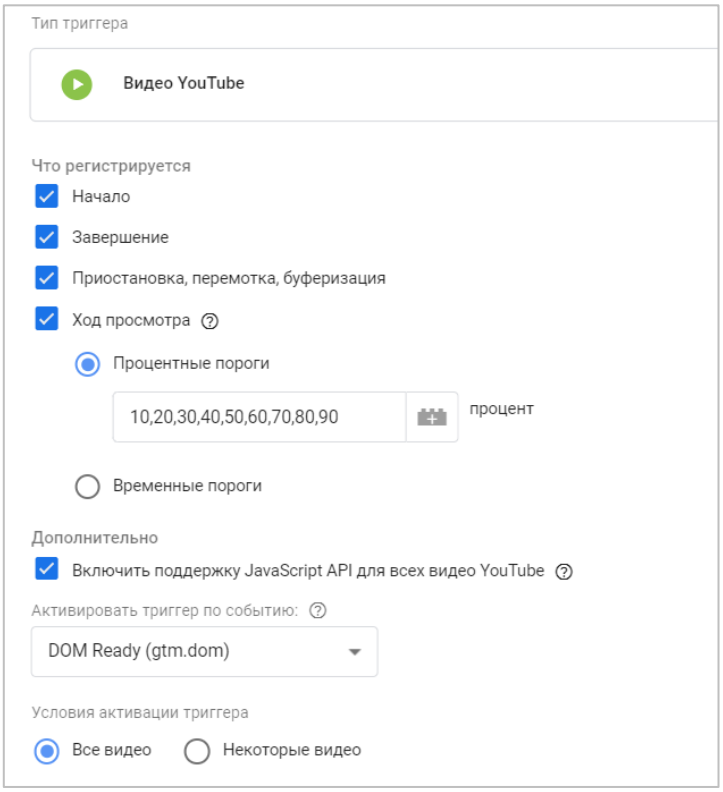


Рис. 395. Пример настройки триггера Видео YouTube

То же самое, что было описано в разделе Переменные:

- **Начало (Start)** – пользователь начинает просмотр видео;
- **Завершение (Complete)** – пользователь достигает конца видео;

- **Приостановка (пауза), перемотка, буферизация (Pause, Seeking, Buffering)** – пользователь останавливает, перематывает видео или когда происходит буферизация;
- **Ход просмотра (Progress)** – пользователь проходит процентный или временный порог (время измеряется в секундах). Целые положительные числа указываются через запятую.
- **Включить поддержку JavaScript API для всех видео YouTube**;
- **Активировать триггер по событию** – возможность задать событие, когда триггер должен начать отслеживание релевантных взаимодействий видео (сразу же после загрузки контейнера gtm.js, после обработки модели DOM gtm.dom или когда все окно загружено gtm.load). По умолчанию – DOM Ready (gtm.dom).

Установив этот флажок, вы включите YouTube iFrame Player API. В результате ко всем URL видеопроигрывателя YouTube будет добавлен параметр *enablejsapi = 1* для управления проигрывателем через iframe или JavaScript.

Как правило, данный триггер используется в связке со встроенными переменными из блока Видео: Video Provider, Video Status, Video URL, Video Title, Video Duration, Video Current Time, Video Percent, Video Visible.

Глубина прокрутки

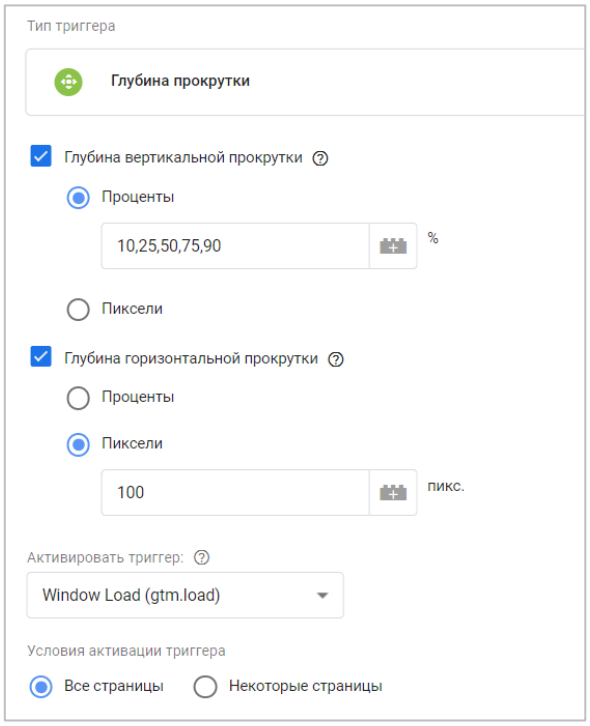


Рис. 396. Триггер Глубина прокрутки

- **Глубина вертикальной прокрутки в процентах** – 10, 25, 50, 75, 90 (пример);
- **Глубина горизонтальной прокрутки в пикселях** – 100 (пример);
- **Активировать триггер** – возможность задать событие, когда триггер должен начать отслеживание релевантных взаимодействий по скроллингу (сразу же после загрузки контейнера gtm.js, после обработки модели DOM gtm.dom или когда все окно загружено gtm.load). По умолчанию – Window Load (gtm.load);
- **Условия активации триггера** – Все страницы.

Как правило, данный триггер используется в связке со встроенными переменными из блока **Прокрутка**: Scroll Depth Threshold, Scroll Depth Units, Scroll Direction.

Доступность элемента

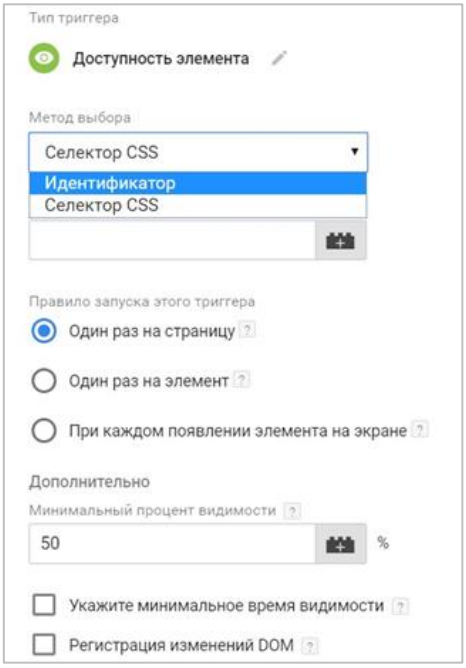


Рис. 397. Триггер Доступность элемента

Доступны следующие настройки:

Метод выбора:

- **Идентификатор (id);**

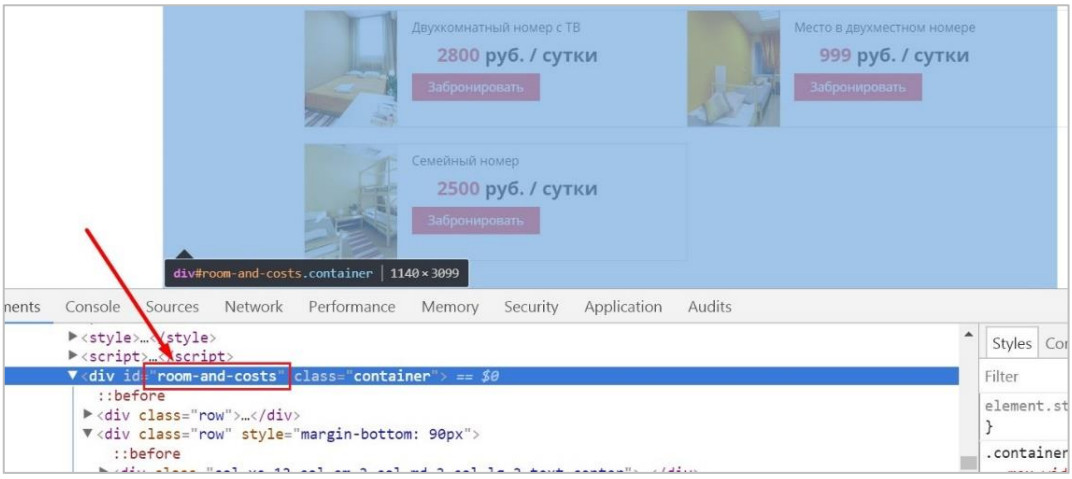


Рис. 398. Метод выбора – id

В большинстве случаев название атрибута id на странице уникальное.

- **Селектор CSS**

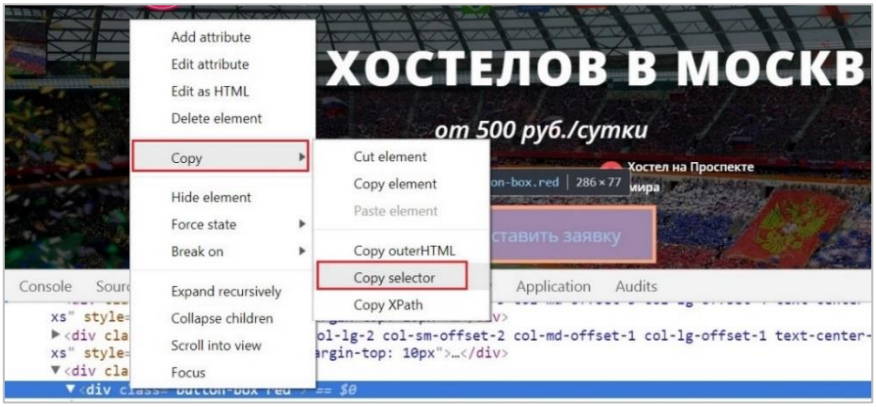


Рис. 399. Метод выбора – Селектор CSS

Применяется, если у отслеживаемого элемента нет атрибута **id**, а также если нам нужно отследить несколько элементов.

Например, у нас на сайте есть несколько форм отправки. У каждой из них есть свой уникальный id. Отслеживание всех форм через **Идентификатор** нам бы не подошло. А вот привязать метод выбора через **Селектор CSS** элементов вполне можно.

Совет: если не знаете, за что зацепиться, а нужно прослушивать конкретный элемент – цепляйтесь за его CSS-селектор. Получить его легко можно через консоль разработчика (в браузере Google Chrome вызывается через клавишу F12, см. рисунок выше).

Правило запуска этого триггера

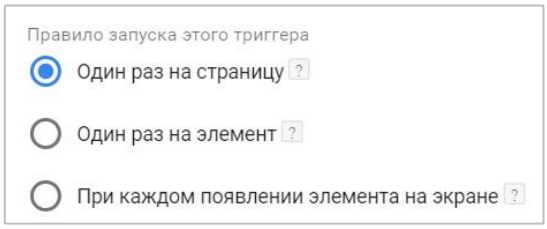


Рис. 400. Правило запуска триггера

- **Один раз на страницу** — если несколько элементов на странице сопоставляются идентификатором или CSS-селектором, этот триггер запускается только один раз, когда первый из них отображается на этой странице.

При перезагрузке страницы триггер сработает повторно;

- **Один раз на элемент** — если несколько элементов на странице сопоставляются CSS-селектором, этот триггер будет запускаться когда каждый из них будет видимым на этой странице в первый раз. Если несколько элементов на странице имеют одинаковый идентификатор, только первый согласованный элемент будет запускать этот триггер;

Если на странице несколько форм, по которым после заполнения информации всплывает одно и то же сообщение, используйте эту настройку.

- **При каждом появлении элемента на экране** — в этом случае триггер срабатывает каждый раз, когда соответствующий элемент становится видимым.

Минимальный процент видимости – какой процент выбранного элемента должен быть виден на экране для срабатывания триггера. Можно использовать собственную переменные.

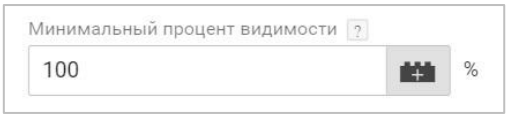


Рис. 401. Минимальный процент видимости

Укажите минимальное время видимости – как долго выбранный элемент должен быть виден на экране для срабатывания триггера. Время периодов видимости на одной странице суммируется. Например, если элемент был виден в течение 4000 мс (4 секунд), затем скрыт из видимости окна браузера и после этого снова виден в течение 2000 мс, то общее время видимости этого элемента составит 6 000 мс.

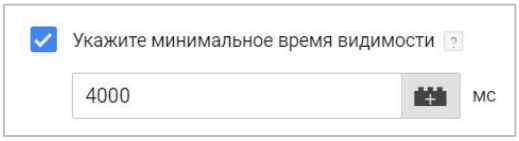


Рис. 402. Минимальное время видимости

Если галочку не ставить, то время видимости для срабатывания триггера не будет играть роли.

Регистрация изменений DOM - настройка отслеживает элементы, которых изначально нет на странице, которые подгружены динамически и могут не входить в DOM (Объектная Модель Документа). Например, всплывающие формы. Если у вас есть на сайте есть такой контент, поставьте галочку, и такие элементы будут регистрироваться.

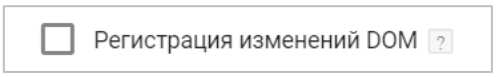


Рис. 403. Регистрация изменений DOM

Отправка формы

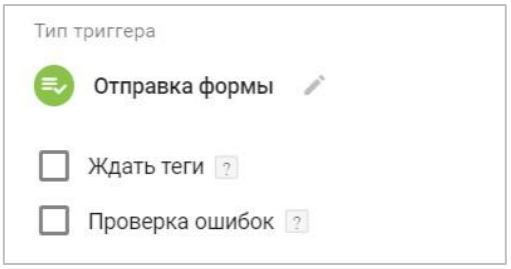


Рис. 404. Триггер Отправка формы

Триггер аналогичен настройкам другого триггера **Клик – Только ссылки**. Можно использовать со специально встроенными переменными: Form Element, Form Classes, Form Target, Form URL, Form Text.

Другое

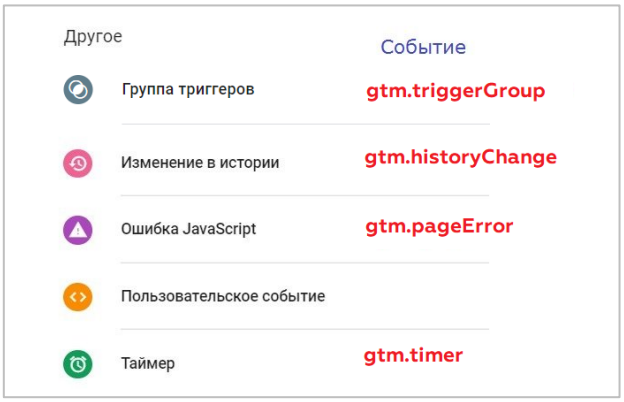


Рис. 405. Триггеры типа Другое

Группа триггеров

Относительно новый тип триггера, который был добавлен в Google Tag Manager в начале 2019 года. Он создает общую зависимость между несколькими отдельными триггерами, и позволяет не запускать тег до тех пор, пока каждый триггер в группе не сработает хотя бы один раз.

Триггер позволяет объединить несколько различных событий (других триггеров) в одно общее условие, которое в дальнейшем используется для активации сторонних тегов. Срабатывает событие gtm.triggerGroup если все выбранные триггеры были активированы хотя бы один раз на странице.

Другими словами, теперь можно настраивать более сложные условия активации, которые состоят из нескольких шагов: **пользователь сделал А, потом выполнил Б, затем совершил действие В**, которые привели к запуску группы триггеров и активации тега.

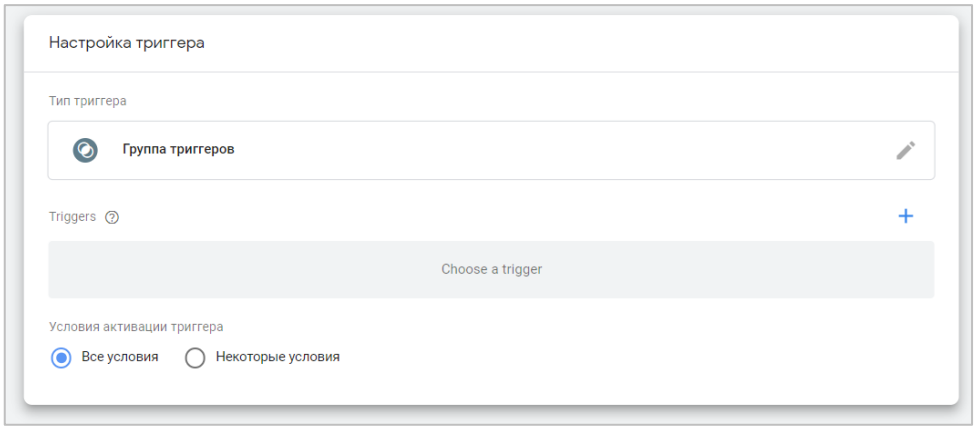


Рис. 406. Настройка группы триггеров

При настройке данного типа необходимо учитывать, что группа триггеров:

- не может использоваться для тега в качестве исключения;
- срабатывает только один раз на страницу (после перезагрузки страницы отдельные триггеры должны быть активированы повторно);
- не заменяет сгруппированные триггеры.

Если в группу добавлено несколько отдельных триггеров, то для активации группы его нужно активировать соответствующее количество раз. Также должно быть удовлетворено **условие И** (и то, и другое событие должно сработать).

Добавить все отдельные триггеры в группу можно с помощью серой области с названием **Choose a trigger** или значка **+** справа над ней.

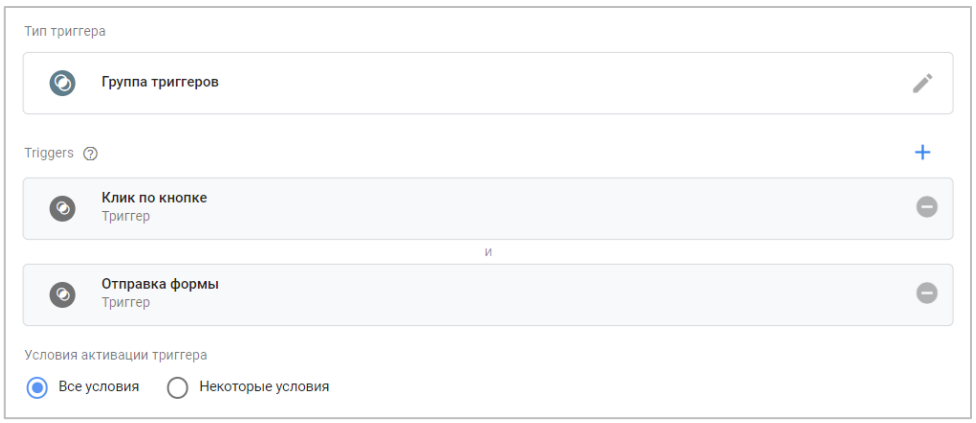


Рис. 407. Отдельные триггеры объединены условием И

Порядок, в котором добавлены отдельные триггеры в группу, значения не имеет. Необходимо только то, чтобы все входящие в группу триггеры были активированы на странице.

Применяется тогда, когда вы хотите, чтобы событие активировалось не по одному действию пользователя, а суммарно по нескольким (сумме событий). Например, при использовании комбинации следующих отдельных триггеров:

- Пользователь был на сайте X секунд (**Таймер**) и проскроллил Y % страницы (**Глубина прокрутки**);

- Пользователь был на сайте X секунд (**Таймер**), проскроллил Y % страницы (**Глубина прокрутки**), и все это было только на конкретной странице (дополнительное условие);
- Пользователь сначала нажал на кнопку (**Клик – Все элементы**), а только потом отправил заявку (**Отправка формы**);
- Пользователь ознакомился с вашим приветственным видео (**Видео YouTube**) и скачал коммерческое предложение (**Клик – Только ссылки**).

Изменение в истории

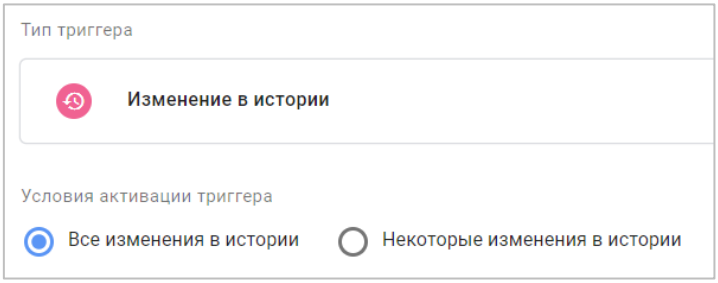


Рис. 408. Изменение в истории

Срабатывает событие gtm.historyChange если изменился фрагмент URL (хэш, #). Как правило, используется на сайтах, контент которых подгружается динамически, без перезагрузки страниц;

В GTM можно использовать со встроенными переменными: New History Fragment, Old History Fragment, New History State, Old History State, History Source.

Ошибка JavaScript

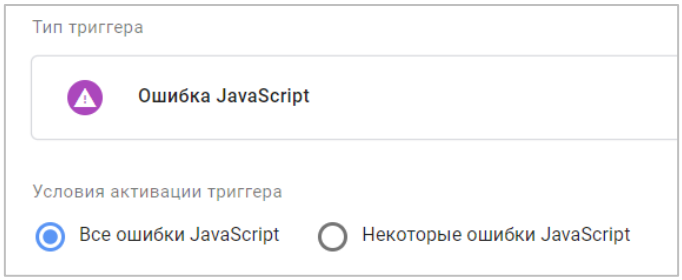


Рис. 409. Ошибка JavaScript

Срабатывает событие gtm.pageError если происходит не перехваченное исключение JavaScript (window.onError). В случае, если в JavaScript мы используем конструкцию try..catch, то GTM не зафиксирует ошибку. Также триггер не будет сигнализировать об ошибках, которые произошли до загрузки кода Диспетчера тегов. Можно использовать со встроенными переменными Google Tag Manager: Error Message, Error URL, Error Line, Debug Mode.

Пользовательское событие

Срабатывает событие event, которое отправили с помощью конструкции;

```
dataLayer.push({'event': 'event_name'});
```

Пользовательские события позволяют отслеживать те взаимодействия, для которых не подходят стандартные методы. Например, отслеживание отправки формы с переопределением события submit.

Допустим мы хотим отправлять событие на уровень данных при нажатии кнопки **Отправить заявку**. Для этого добавим код:

```
dataLayer.push({'event': 'button1-click'})
```

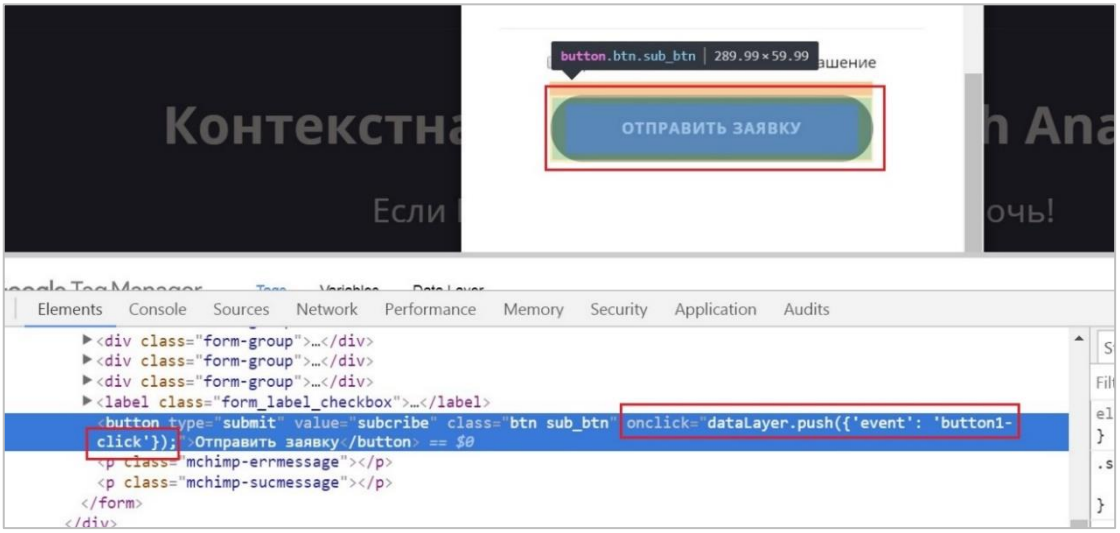


Рис. 410. dataLayer.push({'event': 'button1-click'})

где **button1-click** – название пользовательского события;

Далее необходимо создать тег, а в настройках триггера **Пользовательское событие** задать название button1-click и условие активации – **Все специальные события**. При необходимости можно использовать регулярные выражения.

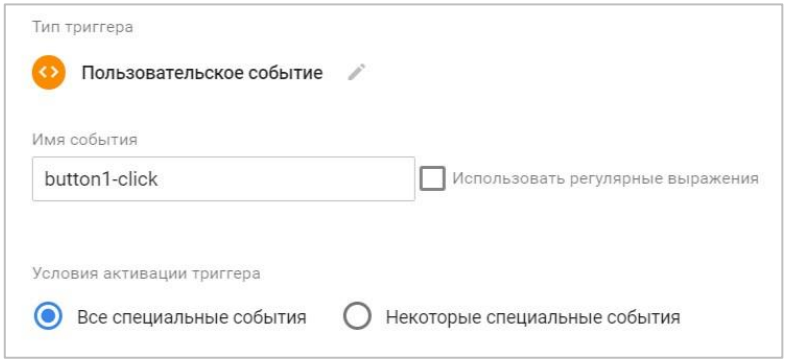


Рис. 411. Пользовательское событие button1-click

Когда пользователь нажмет кнопку, на уровень данных будет отправлено пользовательское событие со значением *button1-click*. Затем GTM определит **button1-click** как значение пользовательского события и активирует тег.

Рис. 412. Событие button1-click

В dataLayer.push() можно добавить не только одну переменную вида ключ=значение, но и целый массив. А чтобы посмотреть порядок, в котором события передаются на уровень данных, откройте консоль JavaScript в веб-браузере и введите dataLayer:

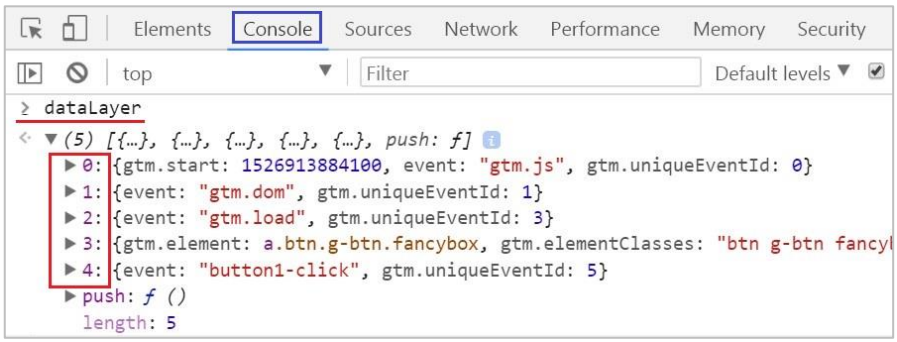


Рис. 413. Console – dataLayer

- **gtm.start** – запуск контейнера GTM;
- **gtm.dom** – модель DOM готова;
- **gtm.load** – окно загружено;
- **gtm.linkClick** – открытие формы;
- **button1-click** – клик по кнопке.

Таймер

Событие **Timer (gtm.timer)** срабатывает через заданный интервал времени (в миллисекундах). Доступны следующие настройки:

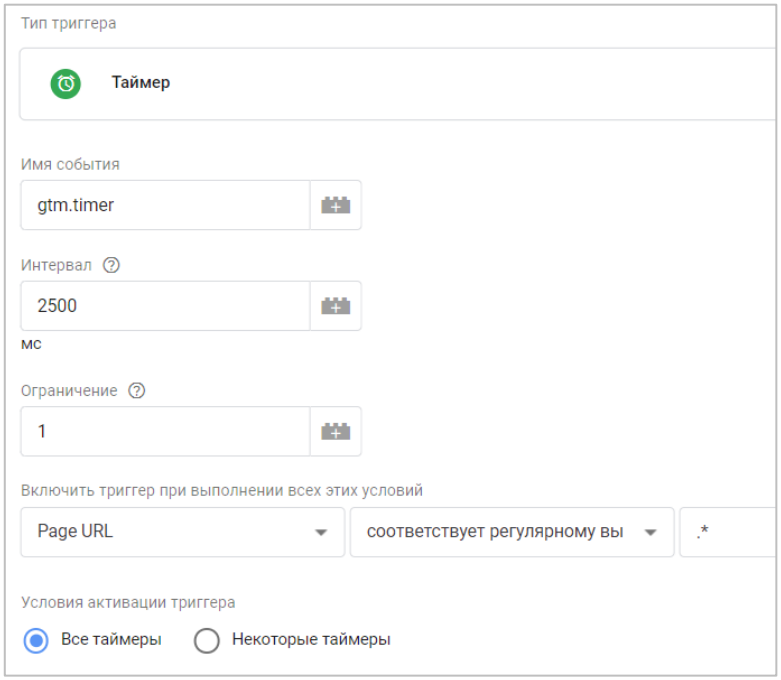


Рис. 414. Триггер Таймер

- **Имя события (по умолчанию gtm.timer)** – можем изменить на любое другое. Например, timer2018;
- **Интервал** - время, через которое должно активироваться событие. Например, через 2,5 секунды;
- **Ограничение** - максимальное число активаций события. Например, 1. Если оставить это поле пустым, событие будет активироваться до тех пор, пока пользователь не покинет страницу.

Если необходимо, чтобы таймер срабатывал на всех страницах сайта, воспользуйтесь конструкцией регулярного выражения.

Так выглядит уровень данных события **gtm.timer**:

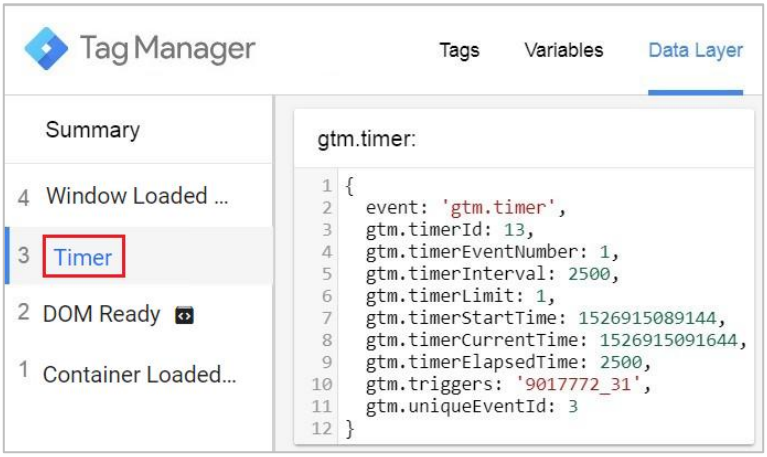


Рис. 415. gtm.timer в Data Layer

Триггеры в Google Tag Manager делятся на **триггеры активации** тегов и **триггеры блокировки** тегов.

Триггер активации — это условие, при выполнении которого срабатывает тег. Например, вы хотите отслеживать клики по определенной кнопке на сайте. Зная значения переменной, вы можете настроить триггер активации, например, по Click ID:

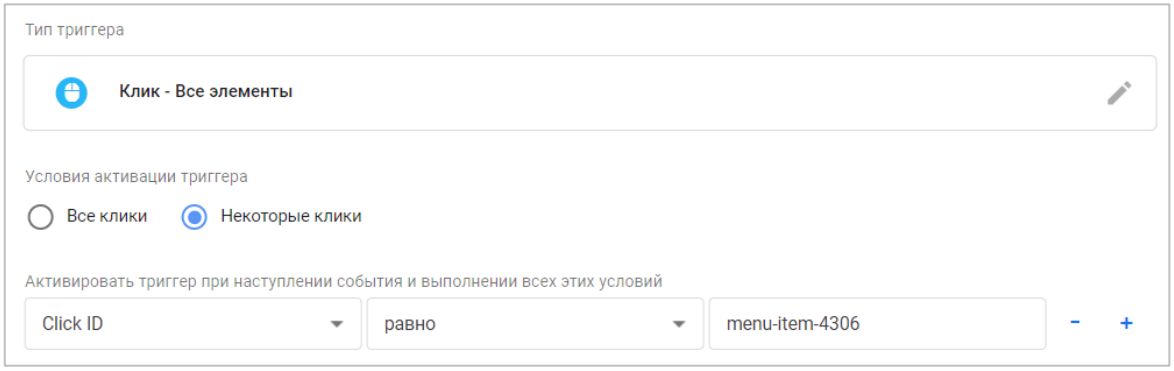


Рис. 416. Пример условия активации триггера

Таким образом, тег активируется только в том случае, когда идентификатор HTML-тега равен **menu-item-4306**.

Триггеры активации делятся на:

- *встроенные триггеры* – доступны к использованию готовые шаблоны;
- *пользовательские триггеры* – ручная самостоятельная настройка.

Триггер блокировки — это условие, при выполнении которого активация тега блокируется. Например, вы хотите активировать тег на всех страницах сайта, кроме одной конкретной. Тогда необходимо создать триггер с условием активации на нужных страницах, а далее в соответствующем теге добавить этот триггер в исключение.

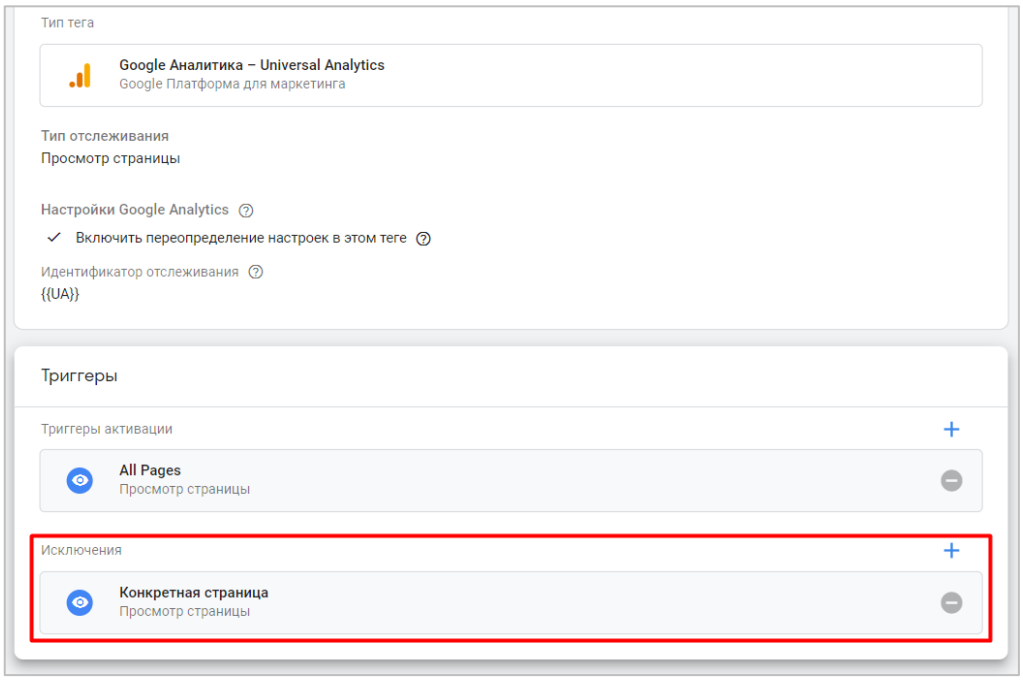


Рис. 417. Триггер блокировки или исключения

При конфликте триггеров активации и блокировки, приоритет отдается блокиратору. Добавить триггер блокировки можно с помощью кнопки **Добавить исключение** в настройках тега.

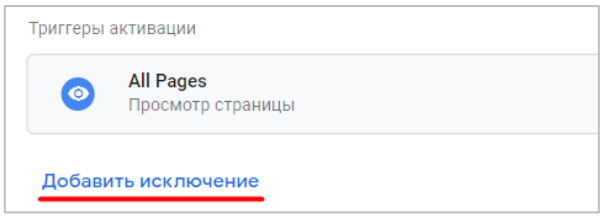


Рис. 418. Добавить исключение

В Google Tag Manager существует триггер, который добавлен по умолчанию и который нельзя удалить. Это **All Pages (Просмотр страницы)**.

Также как и переменные, триггеры в GTM можно *копировать, удалять, просматривать изменения*, и у них можно *вывести примечания*. Для этого в правом верхнем углу нажмите на значок три точки.

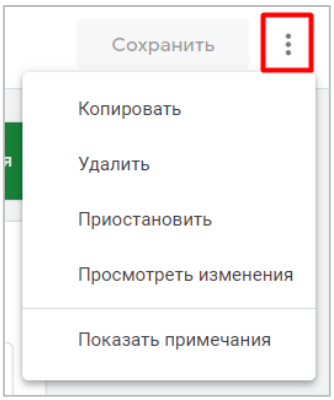


Рис. 419. Действия над триггерами

В **Просмотреть изменения** можно узнать, что конкретно было изменено в переменной по сравнению с предыдущей версией, а в **Показать примечания** есть возможность заносить какие-то пометки, добавить описание по функциональности и т.д.

После создания триггеров их следует добавить к тегам.

Глава 6

Теги

Тег в Google Tag Manager – это фрагмент JavaScript кода, который собирает данные о посетителях на сайте и в приложении, а затем пересылает их на сторонние сервисы — Google Analytics, Google Ads, Facebook, Яндекс.Метрика и т.д.

Если вы не используете Google Tag Manager, то все фрагменты кода других сервисов вам приходится вставлять вручную в исходный код страницы сайта. Это не очень удобно и отнимает много времени. Благодаря GTM все становится куда проще - исходный код менять не требуется, достаточно лишь указать в интерфейсе, какие теги вы хотите использовать и когда их нужно активировать.

В Google Tag Manager есть n-ое количество шаблонов тегов, которые облегчают их установку на сайте. Они разделены на категории:

- **рекомендуемые** – Google Аналитика - Universal Analytics, Тег конфигурации из ресурса Google Аналитики типа «Приложение и сайт», Тег события из ресурса Google Аналитики типа «Приложение и сайта», Отслеживание конверсий в Google Рекламе, Ремаркетинг в Google Рекламе, Счетчик Floodlight, Продажи Floodlight, Связывание конверсий, Google Optimize и Google Опросы;

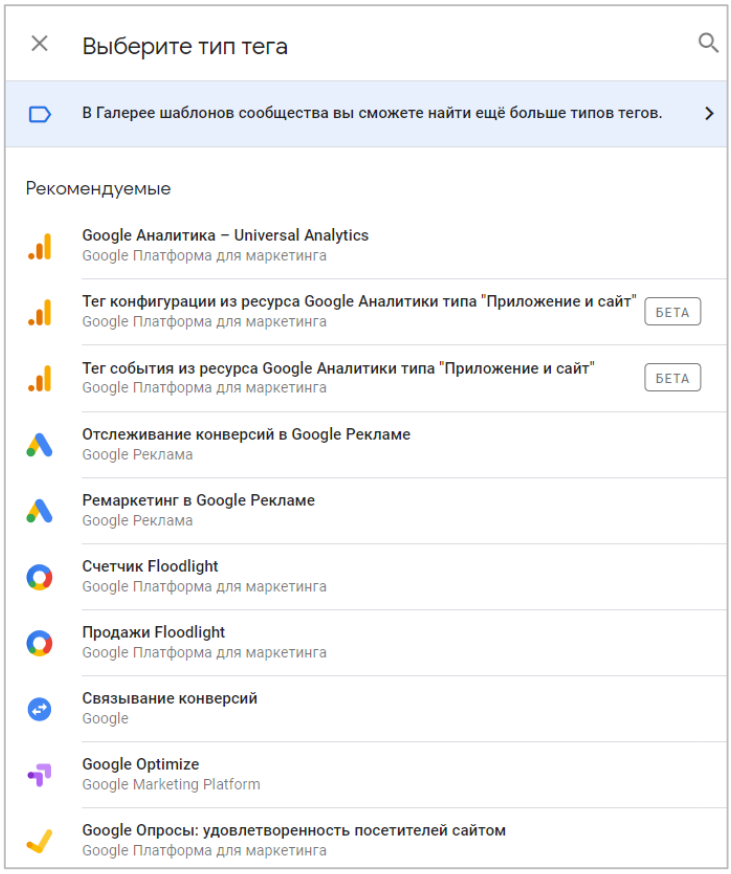


Рис. 420. Рекомендуемые теги

- **специальные** – пользовательский HTML и пользовательское изображение;

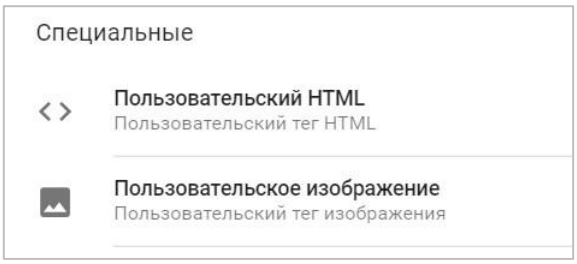


Рис. 421. Специальные теги

- **еще** – сторонние сервисы, Twitter Universal Website Tag, Adometry, Crazy Egg, comScore Unified Digital Measurement, K50 tracking tag, Foxmetrics, Hotjar Tracking Code и другие.

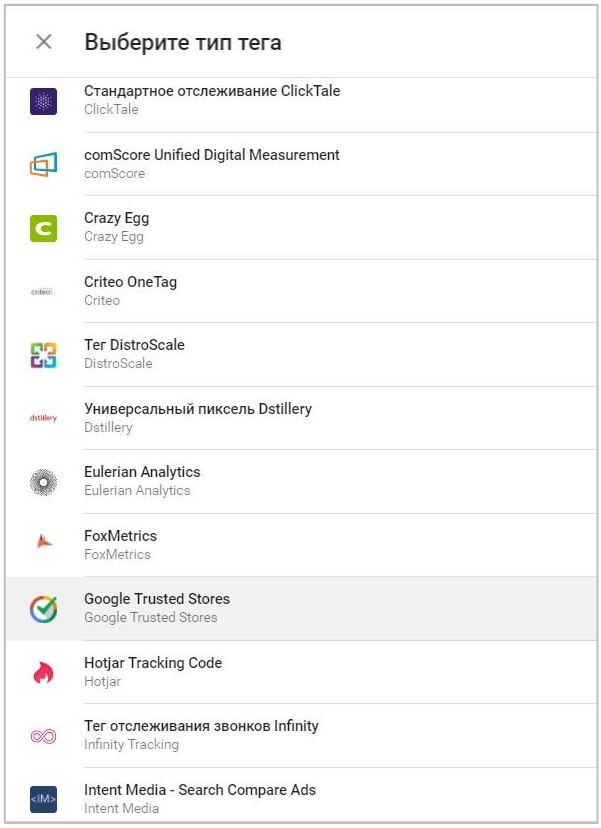


Рис. 422. Теги сторонних сервисов

Чтобы создать тег в Google Tag Manager, перейдите на вкладку меню **Теги** и нажмите кнопку **Создать**.

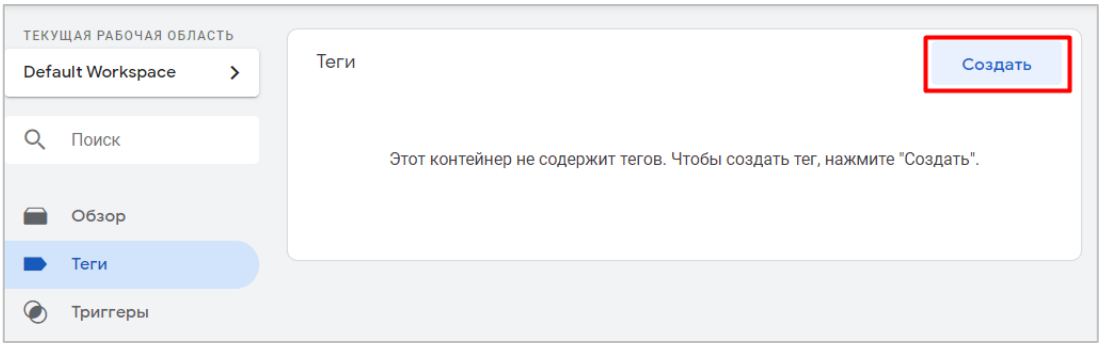


Рис. 423. Создание тега

В появившемся окне нажмите на пустое пространство конфигурации тега и выберите его тип.

Рис. 424. Выбор конфигурации тега

Рекомендуемые теги

Как правило, большинство маркетологов и веб-аналитиков работает с первыми четырьмя рекомендуемыми тегами: **Universal Analytics, Отслеживание конверсий в Google Рекламе** и **Ремаркетинг в Google Рекламе**. В зависимости от решаемых задач могут использоваться и другие.

Примечание: в 2019 году в Google Analytics появился новый тип аккаунта – **Приложение и сайт (App + Web)**. С его помощью можно изучать данные о кроссплатформенном поведении пользователей в одном аккаунте. Это очень удобно для тех владельцев бизнесов, кто имеет как веб-сайт, так и мобильное приложение и хочет отслеживать действия пользователей сразу из двух систем.

Поскольку теги этого типа в GTM в момент написания руководства находятся в режиме **БЕТА**, и сама книга освещает отслеживания только для веб-сайта, я решил не включать разбор текущих настроек.

Google Аналитика – Universal Analytics

Создадим тег **Universal Analytics** и разберем его настройки:

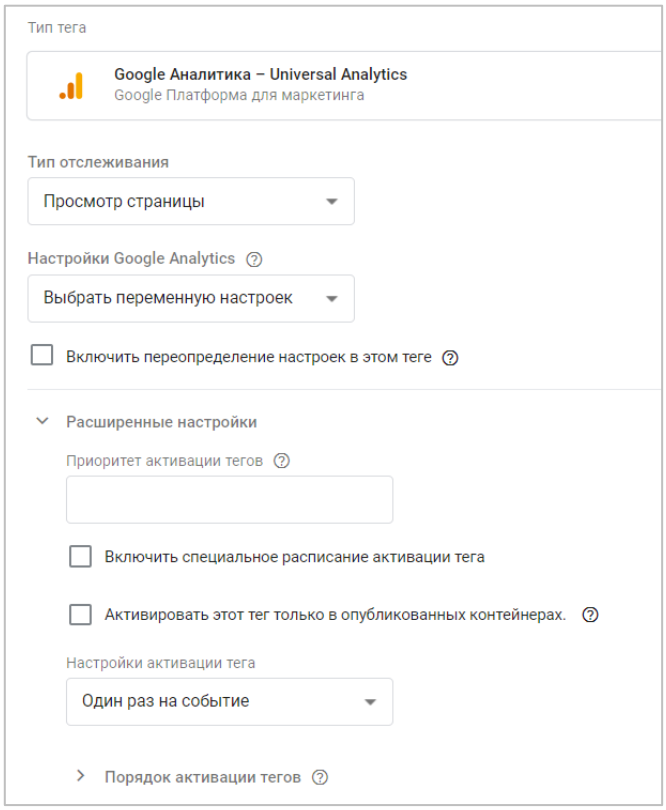


Рис. 425. Тип тега Google Аналитика - Universal Analytics

В зависимости от выбранного в теге типа отслеживания дополнительные поля настроек будут меняться:

Тип отслеживания	Поля, которые необходимо задать	Спец-ные параметры	Спец-ные показатели	Группы контента	Электронная торговля	Реклама	Междоменное отслеживание	Расширенная конфигурация
Просмотр страницы	+	+	+	+	+	+	+	+
Событие	+	+	+	+	+	+	-	+
Транзакция	+	+	+	+	-	+	-	+
Социальные сети	+	+	+	+	-	+	-	+
Время	+	+	+	+	-	+	-	+
Внешний вид ссылки	+	+	+	+	-	+	-	+
Изменить внешний вид формы	+	+	+	+	-	+	-	+

Для простого отслеживания Universal Analytics необходимо добавить один тег и выбрать тип отслеживания **Просмотр страницы**. Если же вас интересуют события или транзакции электронной торговли, потребуется добавить и другие теги.

Тип отслеживания – Просмотр страницы (Page View)

Выбирается в том случае, когда вас интересует, какие страницы просматривают посетители на вашем сайте.

Для того, чтобы информация из GTM передавалась в Google Analytics, необходимо в тег добавить идентификатор отслеживания GA. Сделать это можно несколькими способами:

1. указать переменную в опции **Настройки Google Analytics**;

При такой настройке следует добавить идентификатор отслеживания Google Analytics (например, UA-113446186-1) в пользовательскую переменную типа **Настройки Google Analytics**, предварительно создав ее.

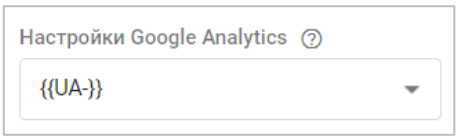


Рис. 426. Настройки Google Analytics

2. поставить галочку **Включить переопределение настроек в этом теге** и вручную добавить идентификатор отслеживания Google Analytics, или же использовать пользовательскую переменную типа **Константа**.

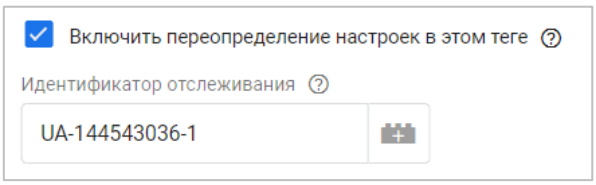


Рис. 427. Включить переопределение настроек в этом теге

При активации данного способа у тега появляются дополнительные поля переопределения настроек, в которых мы можем изменять настройки счетчика Google Analytics через интерфейс Google Tag Manager. Она служит для настройки конфигураций Google Analytics, общих для нескольких тегов. Когда вы создаете тег Google Analytics с помощью Universal Analytics, Диспетчер предложит вам выбрать или создать переменную настроек Google Analytics. Большинство пользователей начнут работу с переменной настроек, содержащей только идентификатор отслеживания Google Analytics. Эту переменную можно будет использовать в любых дополнительных конфигурациях тегов с тем же идентификатором отслеживания Google Analytics (при этом вам не понадобится повторно его указывать).

- **Поля, которые необходимо задать** – доступен раскрывающийся список с полями, в которых задаются пара *ключ:значение*;

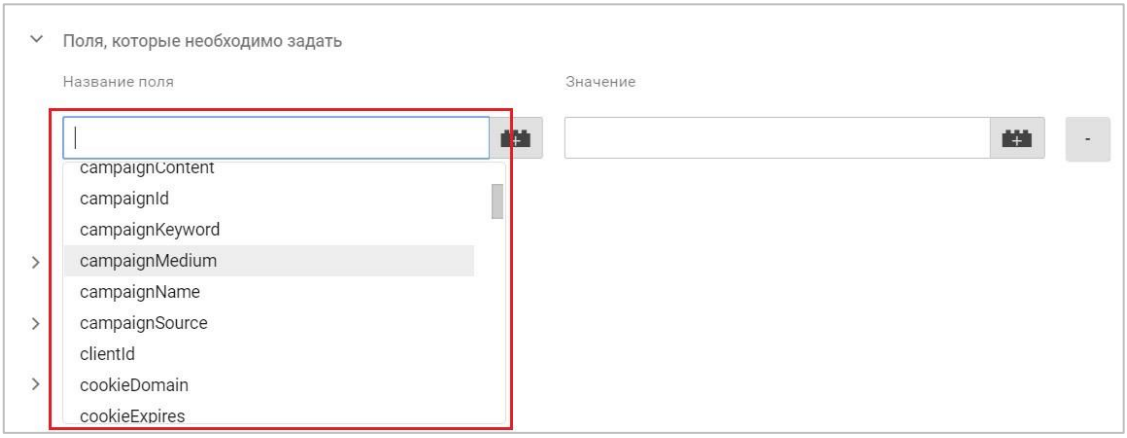


Рис. 428. Поля, которые необходимо задать

Например, вы можете добавить поле для кроссплатформенного отслеживания (функция User ID) или для отслеживания поддоменов. При настройке электронной коммерции в **Fields to Set** также можно передавать значения по товарам – идентификатор (ecomm_prodid), тип страницы (ecomm_pagetype) или общую ценность товаров (ecomm_totalvalue), а также многое другое.

- Специальные параметры;
- Специальные показатели;
- Группы контента;
- Реклама – Включить функции для контекстно-медийной сети;

Если вас интересуют отчеты по демографии и интересам, ремаркетинг Google Analytics и интеграция с DoubleClick Campaign Manager (DCM), задайте значение **True**. Включить эту функцию можно непосредственно через сам Google Analytics (на уровне ресурса **Отслеживание – Сбор данных – Ремаркетинг - Вкл.**)

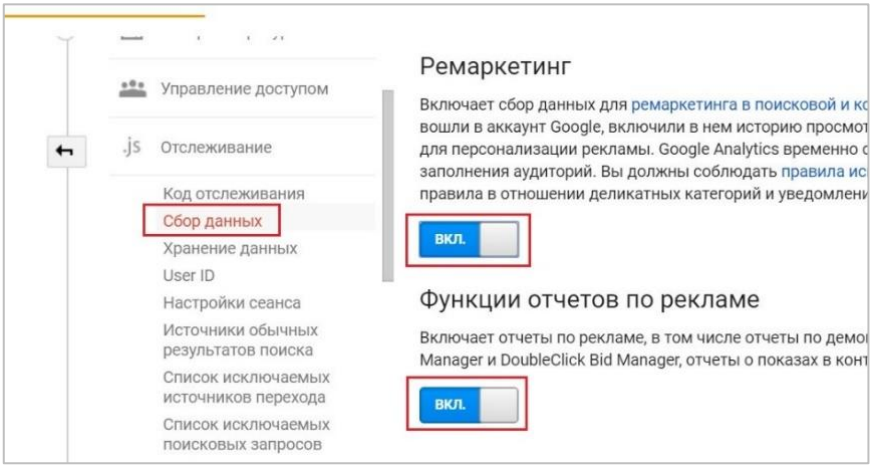


Рис. 429. Включение ремаркетинга и функций отчетов по рекламе в Google Analytics

- Электронная торговля - Включить расширенные функции электронной торговли;

При значении **True** вы сможете узнавать, когда пользователи добавляли товары в корзину, переходили к оформлению покупки и завершали ее, а также какие сегменты покупателей не совершают заказ. В Google Analytics настройка задается на уровне представления в разделе **Настройки электронной торговли**.

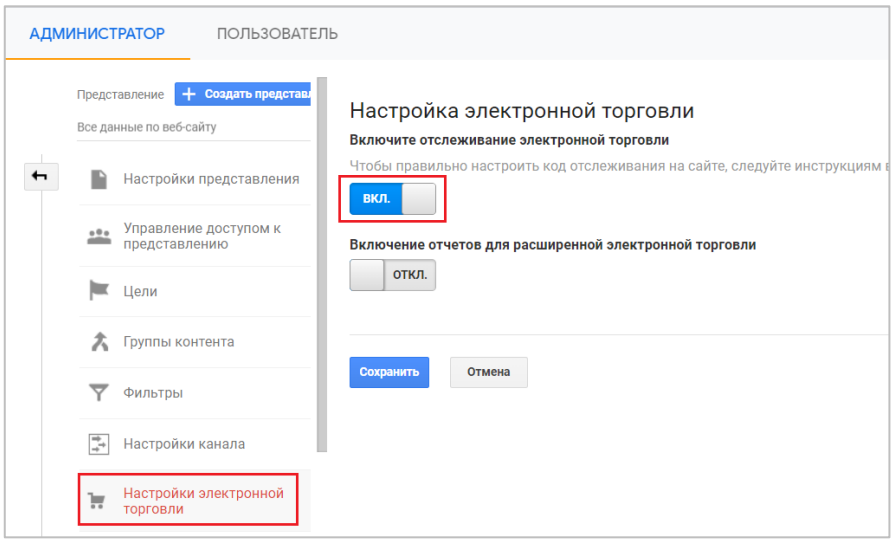


Рис. 430. Включение отслеживание электронной торговли

Примечание: просто включив эту настройку, данные по электронной коммерции передаваться не будут. Настройка Enhanced Ecommerce куда более сложный процесс и требует определенного времени и навыков.

- **Междоменное отслеживание;**

Для корректного сбора данных можно настроить междоменное отслеживание. Тогда сеанс с переходом между вашими разными сайтами, например, domain.com, domain.ru, domain.org, будет засчитан как один сеанс, а время сеанса будет определено как общее время пребывания на всех посещенных ресурсах.

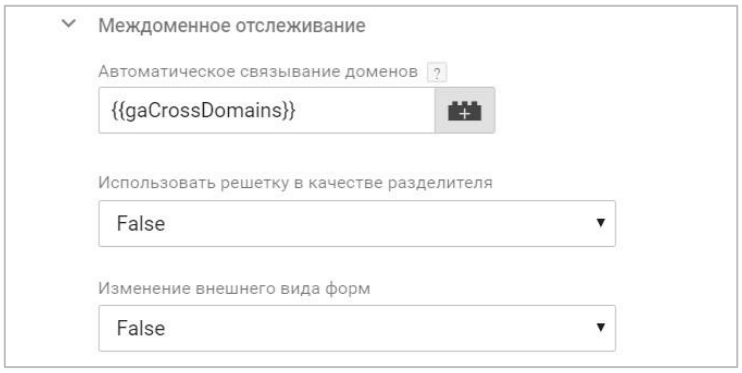


Рис. 431. Междоменное отслеживание

Автоматическое связывание доменов. Перечислите через запятую домены, для которых следует использовать общий идентификатор клиента, либо же создайте переменную типа Константа и задайте все домены там;

Использовать в качестве разделителя решетку. Позволяет использовать решетку (#) вместо знака вопроса (?) для добавления значений cookie в URL;

Изменение внешнего вида форм. Позволяет добавить информацию отслеживания к действию с формой.

- **Расширенная конфигурация**, в которой можно задать глобальную функцию, использовать отладочную версию и улучшенную атрибуцию ссылок. Настройки в этой категории используются крайне редко и предназначены для опытных пользователей.

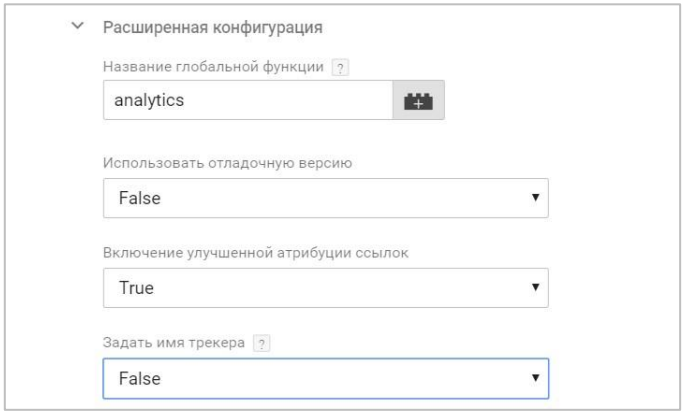


Рис. 432. Расширенная конфигурация

Название глобальной функции. Позволяет переименовать глобальную функцию, используемую тегом Universal Analytics.

В некоторых случаях при добавлении элементов библиотеки *analytics.js* на страницу переменная **ga** может быть уже занята. Эту проблему можно решить путем переименования глобального объекта **ga**. Например, чтобы переименовать объект **ga** в analytics, нужно изменить код отслеживания следующим образом:

```
<!-- Google Analytics -->
<script>
(function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){
(i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new Date();a=s.createElement(o),
m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,m)
})(window,document,'script','//www.google-analytics.com/analytics.js','analytics')
          Было ga
analytics('create', 'UA-XXXXX-Y', 'auto');
analytics('send', 'pageview');
</script>
<!-- End Google Analytics -->
```

Рис. 433. Изменение переменной ga в коде отслеживания Google Analytics

Подробнее о переименовании глобальной функции (объектов ga) читайте в официальной справке разработчиков Google (см. приложение).

Использовать отладочную версию. Этот вариант библиотеки *analytics.js* при выполнении создает подробные записи в консоли JavaScript.

Они содержат сведения об успешно выполненных командах, а также предупреждения и сообщения об ошибках в коде отслеживания. Кроме того, с их помощью можно получить подробную информацию о каждом обращении, отправленном в Google Analytics. Это позволит проверить, какие именно данные отслеживаются.

Чтобы включить отладочную версию библиотеки *analytics.js*, измените URL в коде отслеживания JavaScript с *https://www.google-analytics.com/analytics.js* на *https://www.google-analytics.com/analytics_debug.js*

```
(function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){
(i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new Date();a=s.createElement(o),
m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,m)
})(window,document,'script','https://www.google-analytics.com/analytics_debug.js','ga');

ga('create', 'UA-XXXXX-Y', 'auto');
ga('send', 'pageview');
```

Рис. 434. Включение отладочной версии через код отслеживания

Включение улучшенной атрибуции ссылок. Позволяет повысить точность отчета **Статистика страницы**. Когда страница содержит несколько ссылок на один URL, к каждой ссылке добавляется уникальный идентификатор элемента.

В Google Analytics эта функция находится на уровне ресурса в разделе **Настройки ресурса**.

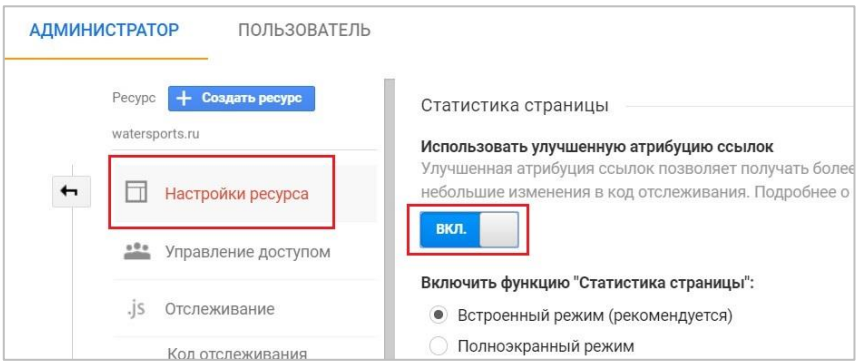


Рис. 435. Использование улучшенной атрибуции ссылок в Google Analytics

Задать имя трекера. Позволяет присвоить название объекту трекера. Не рекомендуется использовать в GTM, так как указав уже существующее имя трекера, это может привести к искажению данных. Хотя присвоение имени позволяет согласовать работу трекеров, созданных в Google Tag Manager с уже существующим кодом Google Analytics, вместо этого лучше перейти на *dataLayer*.

С помощью функции переопределения можно сделать так, чтобы тег Google Analytics использовал все те же настройки, которые были заданы в переменной настроек Google Analytics, но данные передавались на другой идентификатор отслеживания.

Подробнее о том, какие теги Google Analytics и дополнительные настройки поддерживает GTM, читайте в официальной справке Google (см. приложение).

Расширенные настройки

Приоритет активации тегов - целое число (положительное или отрицательное), определяющее порядок запуска тегов. Чем больше данное число, тем раньше будет активирован тег при истинности одного из триггеров, связанных с данным тегом. По умолчанию значение "0", и все теги начинают выполняться одновременно независимо от того, закончилось ли выполнение предыдущего тега. Например, тег с приоритетом "3" будет запускаться до тегов с приоритетами 1 и 2.

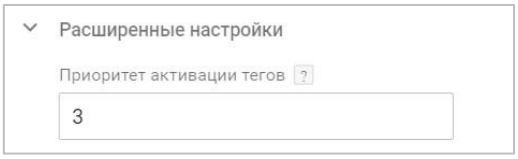


Рис. 436. Приоритет активации тегов

Включить специальное расписание активации тега. Если вы знаете, что некоторый тег понадобится вам на сайте в определенный период времени (например, при проведении рекламной акции вы отслеживаете определенные действия пользователей на сайте) просто активируйте данную настройку. Вы можете задать дату начала (время начала) и дату окончания (время окончания). Также дополнительно необходимо указать часовой пояс. Дата и время окончания не могут предшествовать дате и времени начала.

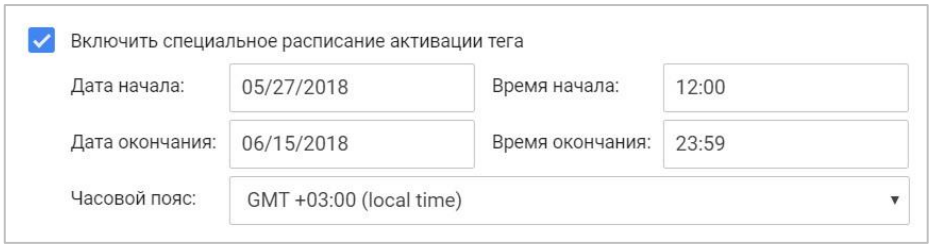


Рис. 437. Включение специального расписания активации тега

Активировать этот тег только в опубликованных контейнерах. Настройка позволяет исключать запуск тегов в определенных ситуациях, например, в режиме предварительного просмотра.

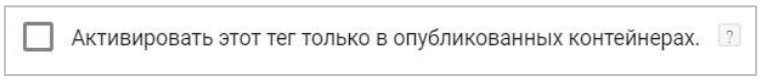


Рис. 438. Активировать тег только в опубликованных контейнерах

Когда вы находитесь в режиме отладки, как правило, вы контейнер не опубликовывайте до тех пор, пока не убедитесь в корректности работы всех настроек. При активации данной настройки тег не будет активироваться, поскольку он не опубликован. Чтобы результаты тестирования были более точными, не включайте эту функцию.

Настройки активации тега

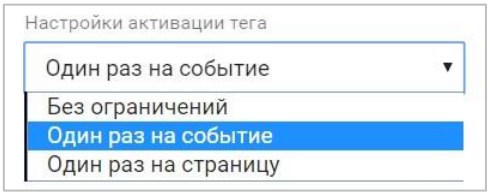


Рис. 439. Настройки активации тега

Без ограничений. Тег активируется при каждом срабатывании триггера. Этот вариант используется, только если задан порядок активации тегов.

Один раз на событие (указывается по умолчанию). Тег активируется только один раз при выполнении определенного события. Этот вариант полезен для передачи информации на уровень данных. Информация передается один раз, поэтому и тег срабатывает один раз.

Один раз на страницу. Тег активируется один раз при загрузке страницы. Этот вариант применяется, когда необходимо активировать какой-то пользовательский скрипт JS, который мы подгрузили на страницу с помощью пользовательского HTML-тега.

Например, активации тега **Один раз на страницу** полезна, когда мы хотим отследить прокрутку страницы. Пользователь дошел до какого определенного элемента один раз, и мы это действие отследили. При последующих прокрутах туда-сюда нам уже это не так важно.

Порядок активации тегов

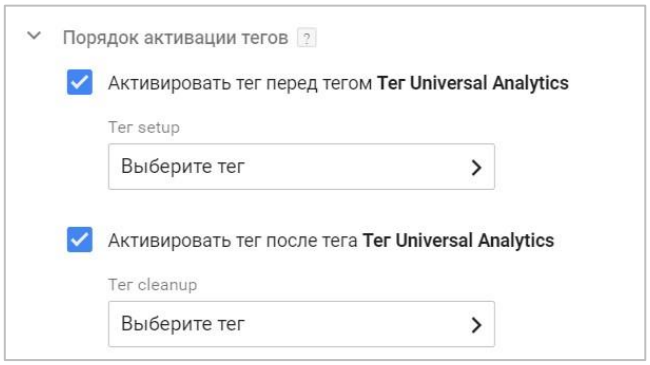


Рис. 440. Порядок активации тегов

В Google Tag Manager есть возможность задать тег, который будет активирован перед текущим тегом и тег, который будет активирован после выполнения текущего тега.

После выбора тега у нас появляется настройка, которая позволяет указать параметр.

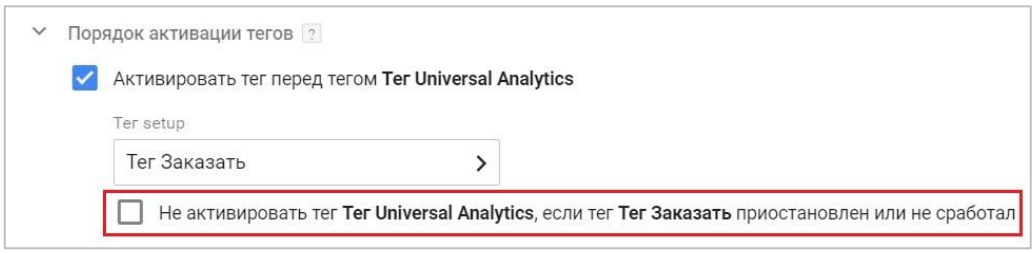


Рис. 441. Порядок активации тегов

Если выбираем активировать тег перед текущим, то мы можем указать, что данный тег, в котором мы сейчас находимся не должен быть активирован, если тег, который задан перед ним, приостановлен или не сработал.

Если мы выбираем тег, который должен быть активирован после данного тега, в котором мы сейчас находим и выполняем настройку, то поставив галочку мы указываем, что тег, который выбран из списка, не должен быть активирован, если тег, в котором мы сейчас находимся, приостановлен или не сработал.

В качестве простого примера разберем:

1. клик по кнопке **Подобрать букет** на сайте;

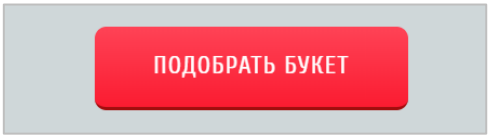


Рис. 442. Пример отслеживания клика по кнопке

2. вывод с помощью пользовательского HTML-тега сообщения в консоли браузера Hello, World. Конструкция имеет такой вид:

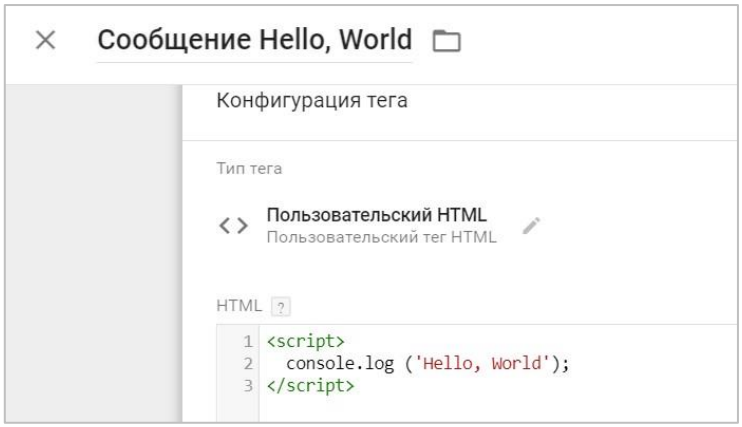


Рис. 443. Вывод сообщения с помощью пользовательского HTML-тега

В качестве порядка активации тегов зададим следующие значения:

- в теге – Подобрать букет активируем настройку запуска тега перед текущим, выбрав наш пользовательский HTML-тег Сообщение Hello, World

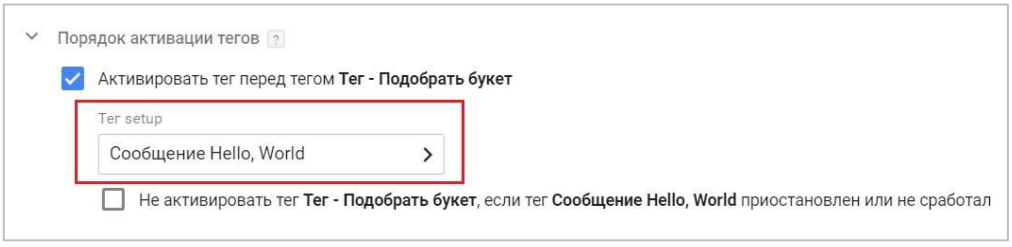


Рис. 444. Активация тега перед Тег – Подобрать букет

Таким образом, у нас сообщение в консоли Hello, World должно появиться до активации тега клика по кнопке Подобрать букет.

У тега **Сообщение Hello, World** не заданы триггеры для активации, но есть информация о том, что данный тег будет активироваться непосредственно перед тегом **Подобрать букет**.

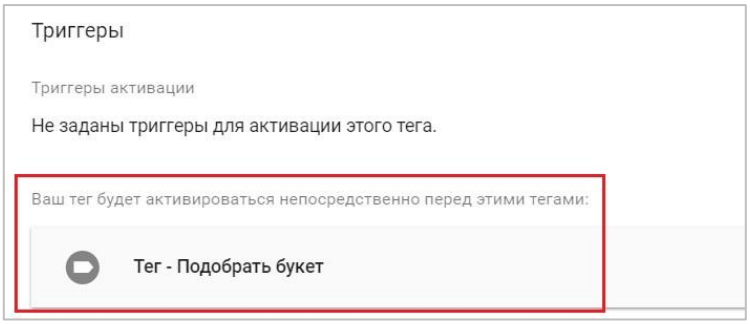


Рис. 445. Активация тега перед другими тегами

Перейдя в режим отладки Google Tag Manager и кликнув по кнопку, мы увидим, что по событию **gtm.click** активировалось два тега, несмотря на то, что у нас в настройках задан только один тег – это **Подобрать букет**.

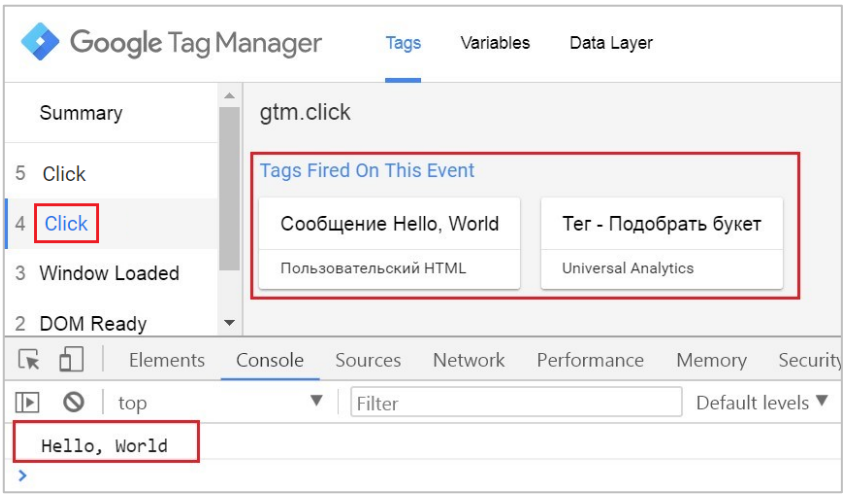


Рис. 446. Пример активации тегов

Данные теги связаны между собой последовательностью активации. Аналогичным образом настраивается порядок активации ПОСЛЕ.

Разберем другие типы отслеживания.

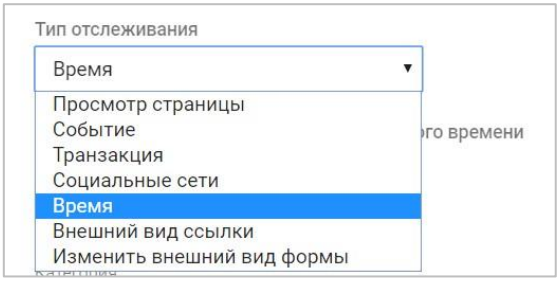


Рис. 447. Типы отслеживания

Тип отслеживания – Событие

В этом типе отслеживания задаются значения, которые использовались при настройке событий в Google Analytics.

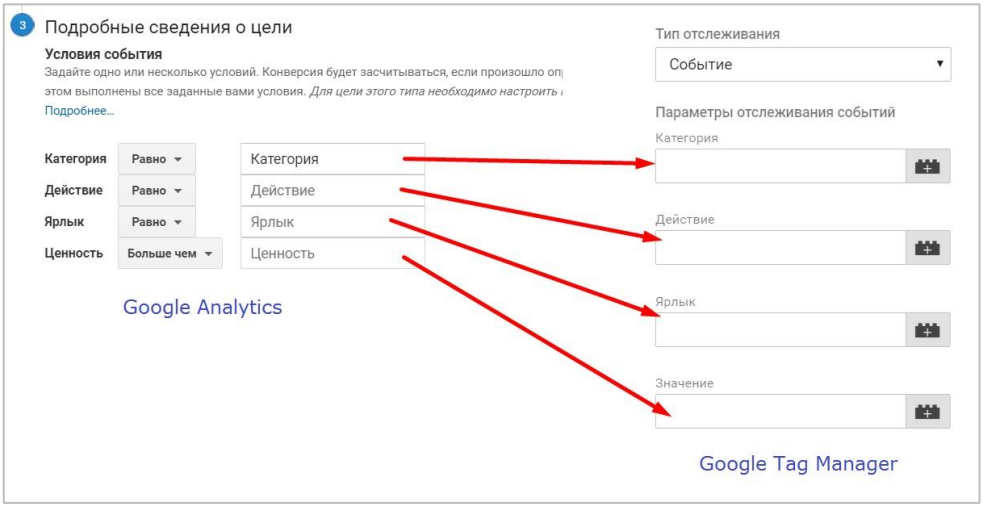


Рис. 448. Тип отслеживания – Событие

Опция **Не взаимодействие**. Если вы хотите, чтобы отправка события влияла на ваш показатель отказов, то необходимо установить значение *False* для этого параметра. В противном случае – устанавливайте значение *True*.

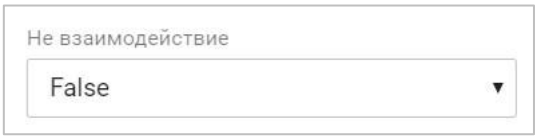


Рис. 449. Опция Не взаимодействие

Все остальные настройки идентичны вышеописанным в типе отслеживания **Просмотр страницы**.

Тип отслеживания – Транзакция

После того, как будет настроена передача данных о покупке, необходимо настроить тег, который будет передавать данные о транзакции в Google Analytics. Для этого и используется тип отслеживания – **Транзакция**.

Важно: включение отслеживания электронной торговли и создание тега с данным типом не позволит вам отслеживать данные о покупках пользователей. Для того, чтобы Google Tag Manager начал передавать сведения в Google Analytics обо всех транзакциях (идентификаторе товара, стоимости товара, наименовании и т.д.), необходимо использовать уровень данных (dataLayer), а также определенную конструкцию для передачи соответствующих параметров.

Тип отслеживания – Социальные сети

С помощью данного типа можно отслеживать социальные взаимодействия пользователей с сайтом, например, лайки/репосты Facebook, ВКонтакте, Twitter и т.д.

Тип отслеживания – Время

Отслеживание событий, срабатывающих по таймеру, может использоваться, если у вас есть страница (например, для просмотра видео), на которой пользователи могут долго оставаться, не запуская событий. Поскольку сеансы Google Analytics по умолчанию прекращаются через 30 минут, вам может понадобиться реализовать пользовательское событие, чтобы активность возобновлялась через какой-то другой промежуток времени.

Как правило, тип отслеживания **Время** в GTM используется в связке с триггером **Таймер**, а для передачи данных в Google Analytics задаются параметры отслеживания пользовательского времени: переменная, категория, значение, ярлык.

Тип отслеживания – Внешний вид ссылки

Данный тип предназначен для отслеживания исходящих/внешних ссылок, то есть выходов (ухода) пользователей с сайта или загрузку файлов на странице (брошюр, прайс-листов, купонов и т.д.).

В качестве дополнительной настройки доступен параметр внешнего вида ссылки # (решетки) в качестве разделителя.

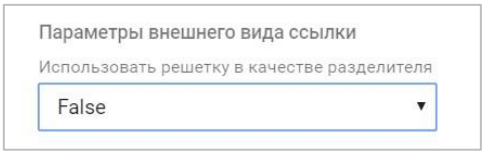


Рис. 450. Параметры внешнего вида ссылки

Тип отслеживания – Изменить внешний вид формы

Этот тип отслеживания позволяет добавить информацию отслеживания к действию с формой. Например, чтобы отслеживать формы по доменам, информация об отслеживании должна быть добавлена к URL-адресу назначения формы при отправке формы. URL-адрес назначения формы — это URL-адрес в атрибуте действия формы:

<form action = "https://www.site.ru">

Отсюда URL-адрес назначения формы называют действием с формой, а добавление информации отслеживания к действию с формой называют изменением внешнего вида формы.

В качестве дополнительной настройки также доступен параметр внешнего вида ссылки # (решетки) в качестве разделителя.

Отслеживание конверсий в Google Рекламе

Рекомендуемый тег в Google Tag Manager предназначен для отслеживания действий пользователя на сайте и передачи информации в Google Ads. Например, когда пользователь оформляет заказ на сайте, подписывается на рассылку, звонит в вашу компанию, скачивает приложение и т.д. Все эти действия называются конверсиями.

После создания конверсии в Google Рекламе необходимо настроить тег.

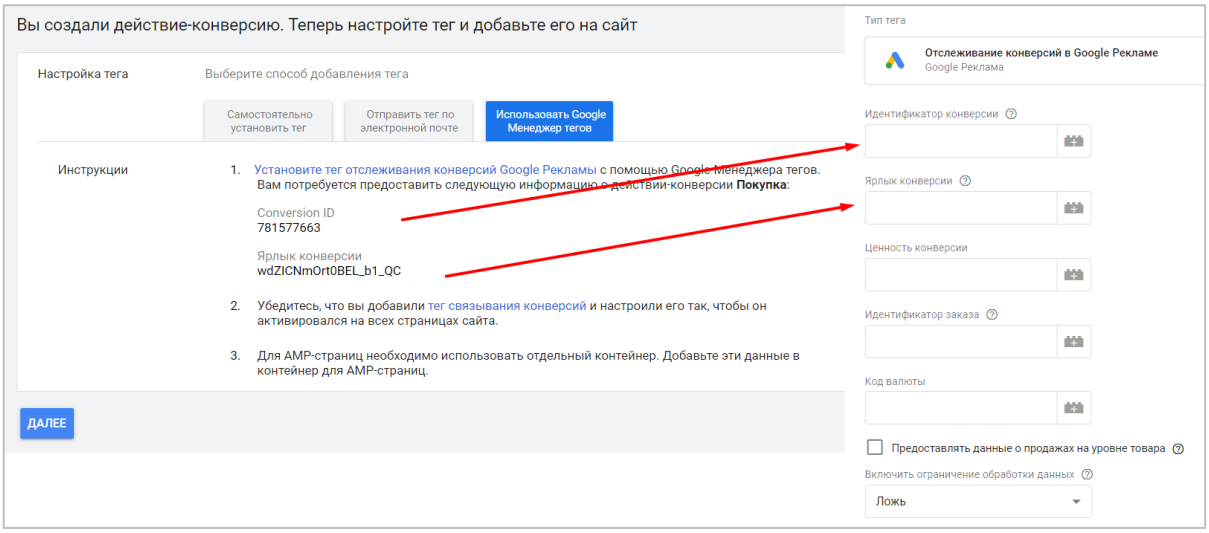


Рис. 451. Отслеживание конверсий в Google Рекламе

Добавляем в тег **Отслеживание конверсий в Google Рекламе**:

- **Идентификатор конверсии (Conversion ID);**

- **Ярлык конверсии.**

Остальные значения заполнять не обязательно.

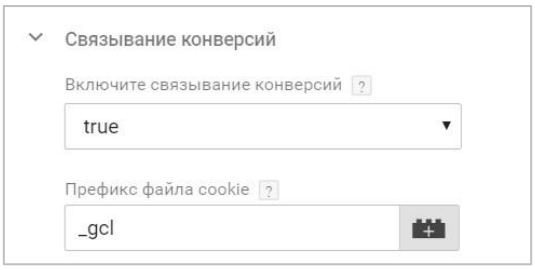


Рис. 452. Связывание конверсий

По умолчанию в опциях **Связывание конверсий** включено связывание конверсий (true) и задан префикс файла cookie **_gcl**.

На втором этапе нужно добавить тег связывания конверсий и настроить его так, чтобы он активировался на всех страницах сайта.

Связывание конверсий

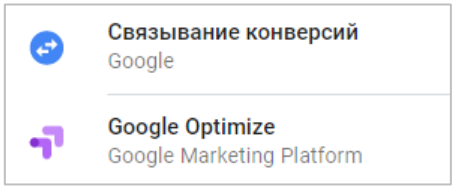


Рис. 453. Тег Связывание конверсий

При нажатии на объявление URL целевой страницы обычно содержит информацию об этом клике. Когда посетитель выполняет нужное вам действие (например, активируя тег отслеживания конверсий в Google Рекламе), эта информация используется для связывания произошедшей конверсии с кликом, который привел пользователя на сайт.

Тег связывания конверсий автоматически считывает информацию о клике из URL целевой страницы и сохраняет эти данные в собственные файлы cookie в вашем домене.

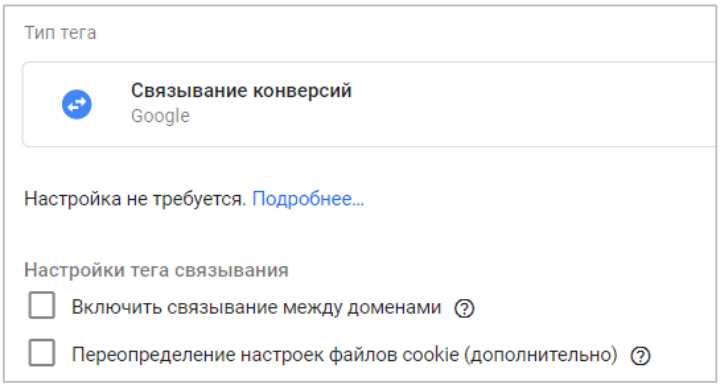


Рис. 454. Настройки тега Связывание конверсий

В качестве активации тега используется триггер **Все страницы**. В большинстве случаев для работы тега связывания конверсий будет достаточно базовой настройки. Однако вы можете переопределить значения полей **Префикс имени (Name prefix), Домен (Domain)** и **Путь (Path)** через настройку **Переопределение настроек файлов cookie (дополнительно)**:

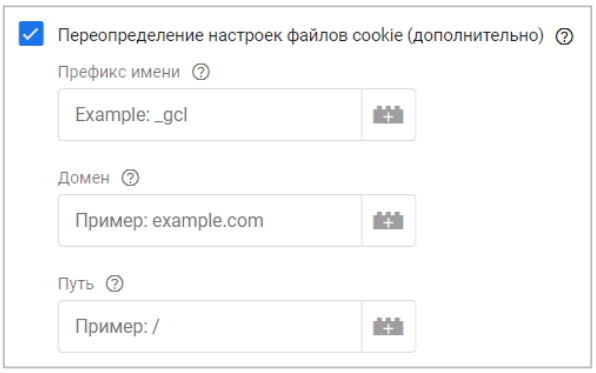


Рис. 455. Переопределение настроек файлов cookie

Имя. Префикс, используемый как часть имен cookie, по умолчанию – _gcl. Например, если вы измените имя на _gcl2, тег связывания конверсий установит файлы cookie под названием _gcl2_aw и _gcl2_dc. При изменении префикса все теги, считывающие информацию о кликах из этих файлов cookie (например, теги отслеживания конверсий в Google Рекламе), нужно будет перенастроить на использование такого же префикса.

Домен. Домен, в котором устанавливаются собственные файлы cookie. По умолчанию тег связывания конверсий использует домен самого высокого уровня. Например, если адрес вашего сайта – blog.site.ru, тег связывания конверсий будет использовать домен site.ru. Это поле следует задавать, только если вы хотите указать для файлов cookie домен более низкого уровня.

Путь. Путь для файлов cookie. По умолчанию используется корневой уровень (/). Это поле следует задавать, только если вам нужно указать для файлов cookie подкаталог домена.

Специальные теги Google Tag Manager

1. Пользовательский HTML
2. Пользовательское изображение

Пользовательский HTML

У тегов, которых нет встроенных шаблонов в GTM, используется категория **Специальные теги**. К ним относятся, например, коды Яндекс.Метрики, Facebook Pixel, Ретаргетинг ВК и другие.

Пользовательский тег HTML – это код стороннего сервиса, который должен быть заключен внутри тегов **<script></script>**. Вот так выглядит пользовательский HTML тег счетчика Яндекс.Метрики:

Рис. 456. Пользовательский HTML-тег, Яндекс.Метрика

Вы также можете включить вызовы функции **document.write()** в тегах, установив соответствующий флажок.

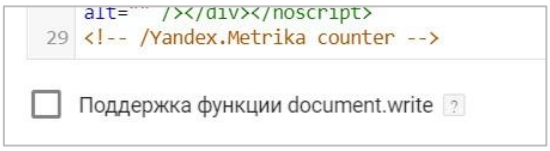


Рис. 457. Поддержка функции document.write

document.write() – метод добавления текста к документу. Он работает только пока HTML-страница находится в процессе загрузки и дописывает текст в текущее место HTML еще до того, как браузер построит из него DOM.

Также в пользовательский HTML тег можно вставлять встроенные и пользовательские переменные. Для этого используют конструкцию двойных фигурных скобок **{{myVariable}}**.

Рис. 458. Вставка переменных в HTML-тег

Пользовательское изображение

Помогает отслеживать поведение пользователей там, где не работают остальные теги (например, в браузерах с отключенным выполнением скриптов). В конце ссылки добавляется невидимый пиксель изображения с URL-адресом определенного формата со специальными параметрами. По ним происходит передача данных.

Universal Analytics, Пользовательский HTML и многие другие теги работают на основе JavaScript. Если они загружаются на странице, на которой отключена поддержка JavaScript, то теги не сработают.

В связи с этим разработчики Google предусмотрели возможность загрузки через тег **<iframe>**. А он, в свою очередь, загружается частью контейнера **<noscript>** и показывает свое содержимое, если браузер не поддерживает работу со скриптами или их поддержка отключена пользователем. В остальных случаях браузер игнорирует этот тег и все, что располагается внутри него.

Именно для этих целей и нужна вторая часть кода контейнера Google Tag Manager, которую Google просит разместить после открывающего тега **<body>**:

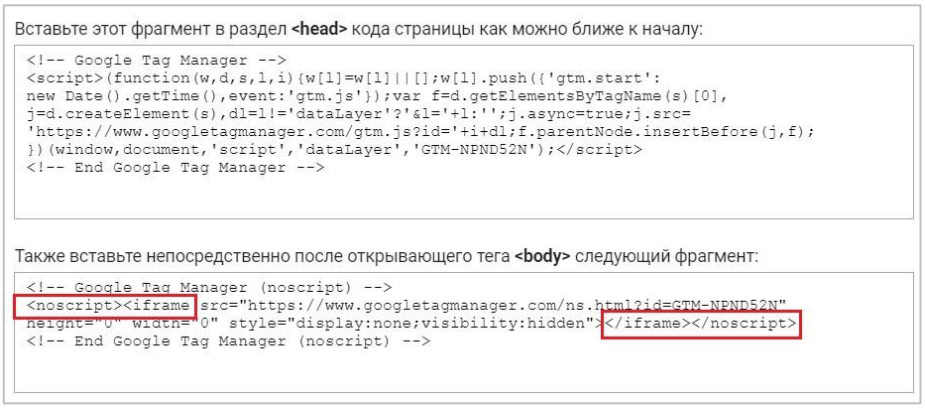


Рис. 459. Фрагменты кода контейнера GTM

Наиболее частое применение тега **Пользовательское изображение** в GTM — отслеживание посещений для пикселя Facebook. У этого типа тега всего две настройки:

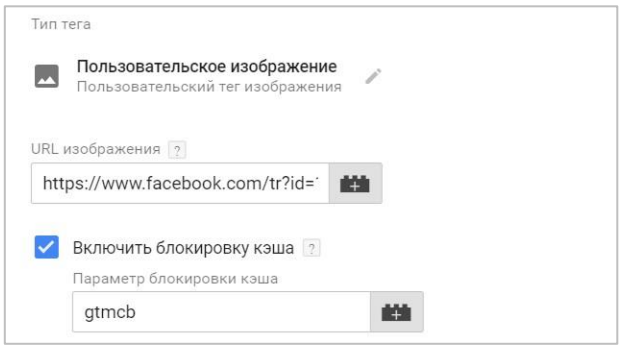


Рис. 460. Настройки пользовательского изображения

- **URL изображения** – поле, где содержится адрес изображения со всеми параметрами, которые нужно передать;
- **Включить блокировку кэша (параметр gtmcb по умолчанию)** – при включении обращения будут регистрироваться в том числе с браузеров с кэшированным изображением.

Знаете ли вы почему сайты загружаются так быстро? Одной из причин является *кэширование*. Первый раз, когда вы заходите на новый сайт, все статические элементы (шрифты, изображений, стили, скрипты и т.д.) загружаются во временную папку, которая будет хранить все эти файлы и в последующих заходах каждый раз показывать на соответствующих страницах. Периодически браузер проверяет, не обновились ли эти файлы на стороне сервера, и при необходимости загружает их заново.

С точки зрения пользователя – это удобно. Нет необходимости ждать очередной полной загрузки страницы. С точки зрения интернет-маркетологов и веб-аналитиков – это не очень хорошо, поскольку на странице данные не поменялись, файлы загрузились один раз во временную папку пользователя и больше отследить к ним обращения возможности нет.

Если отключить галочку параметра **Включить блокировку кэша**, то тег сработает только при первой загрузке сайта, так как изображение будет загружено во временное хранилище. А при включении опции Google Tag Manager добавит параметр gtmcb к URL изображения со случайным значением. Например, так:

site.ru/image.jpeg?gtmcb=1713862799

Поскольку значение в URL всегда разное, браузеру придется каждый раз скачивать изображение. А раз это будет происходить каждый раз, то мы сможем отслеживать это событие постоянно. Поэтому этот параметр лучше всегда держать включенным.

В категории тегов **Еще** содержатся шаблоны к сервисам, которые не так популярны у нас, но которые часто используются в мире. Это и **comScore Unified Digital Measurement**, и тег **Adroll Smart Pixel**, и **Criteo One Tag**, и **Hotjar Tracking Code**, и другие. Руководства по настройке какого-либо тега из этой категории вы найдете на сайте поставщика.

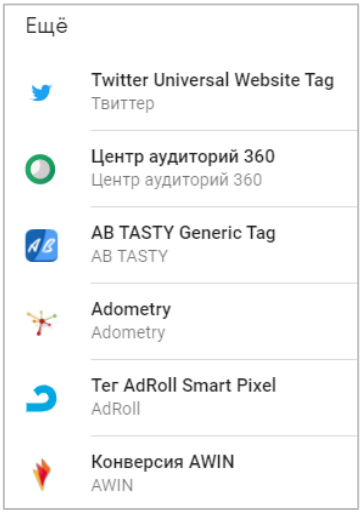


Рис. 461. Теги сторонних сервисов

Обычно теги активируются при загрузке страницы или в ответ на какое-либо действие на ней. В Google Tag Manager вы устанавливаете триггеры, определяющие, когда должны активироваться теги. Например, триггер **Все страницы** будет запускать тег на всех страницах при загрузке сайта.

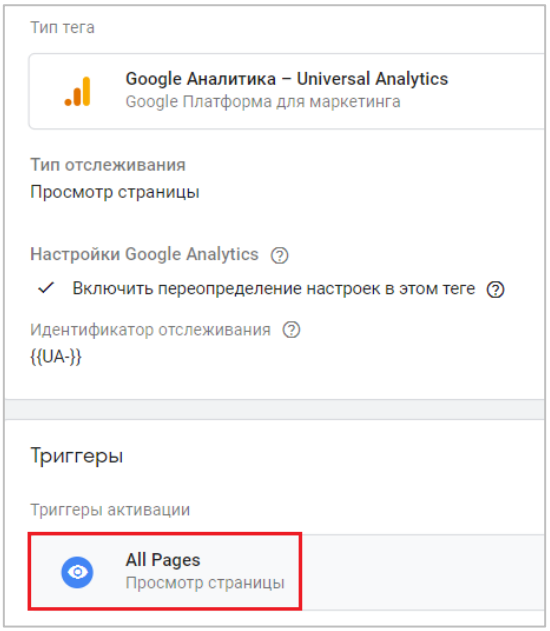


Рис. 462. Триггер активации – All Pages

Для тегов можно задавать как триггеры активации, так и триггеры блокировки (исключения).

Шаблоны тегов

Как вы уже знаете, в диспетчере тегов Google существует возможность использования пользовательских шаблонов (**Custom Templates**).

С их помощью можно создавать собственные теги и переменные, которые ваши коллеги будут использовать наряду со встроенными шаблонами GTM. Добавить тег стороннего разработчика в свою рабочую область можно двумя способами:

1. через вкладку **Шаблоны** и **Поиск в галерее**

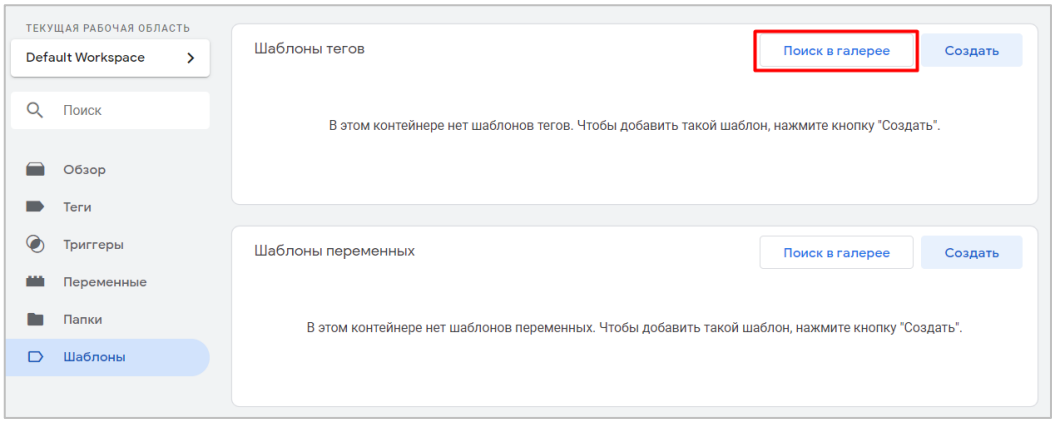


Рис. 463. Шаблоны тегов – Поиск в галерее

2. через вкладку **Теги**, создание тега. В открывшемся меню необходимо кликнуть на самое верхнее уведомление:

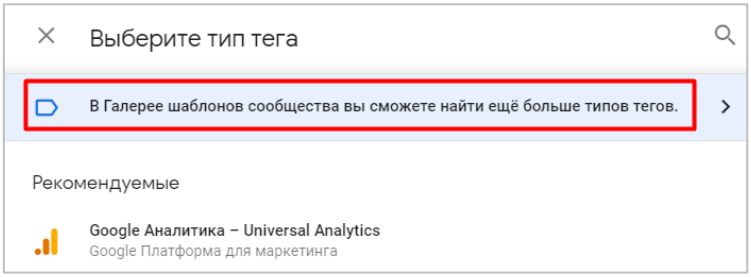


Рис. 464. Создание через уведомление о галереи шаблонов

В открывшемся списке вы увидите большое количество шаблонов тегов как для сторонних сервисов, так и предназначенных для упрощения настройки различных отслеживаний.

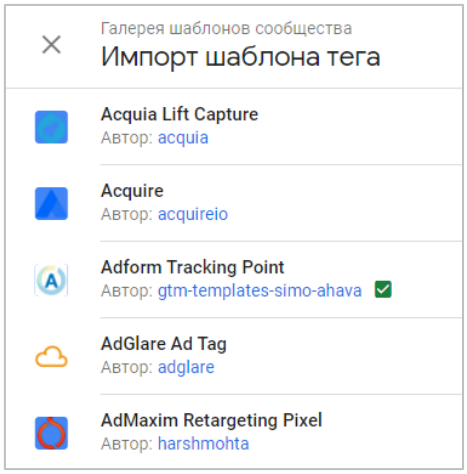


Рис. 465. Импорт шаблона тега

Например, в пользовательских тегах есть готовое решение для установки счетчика Яндекс.Метрики:

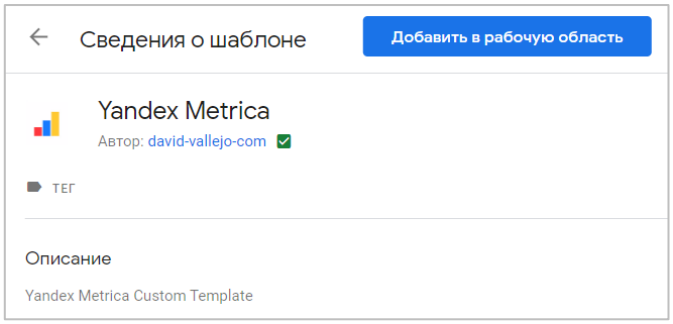


Рис. 466. Шаблон тега Yandex Metrica

Подробнее о том, как с его помощью можно установить счетчик Яндекс.Метрики через Google Tag Manager, будет разобрано в главе, посвященной первым настройкам диспетчера тегов.

После того, как вы добавите шаблон тега в свою рабочую область, сам тег будет отображаться в разделе **Шаблоны**, а также в списке тегов в блоке **Специальные** с соответствующим значком **ГАЛЕРЕЯ**:

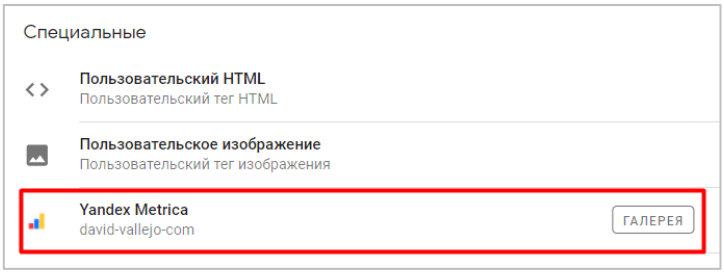


Рис. 467. Шаблон тега в разделе Специальные

Глава 7

Первые настройки в Google Tag Manager

Установка счетчиков веб-аналитики

После регистрации и установки контейнера Google Tag Manager на сайт произведем установку счетчиков веб-аналитики: Google Analytics, Яндекс.Метрика, myTarget, Facebook и ВКонтакте.

Установка кода Google Analytics

Получив идентификатор отслеживания в Google Analytics (UA-XXXXXXXXX-Y), создайте переменную для экономии времени и избежание ошибок в дальнейшем. Для этого перейдите в **Переменные — Создать**.

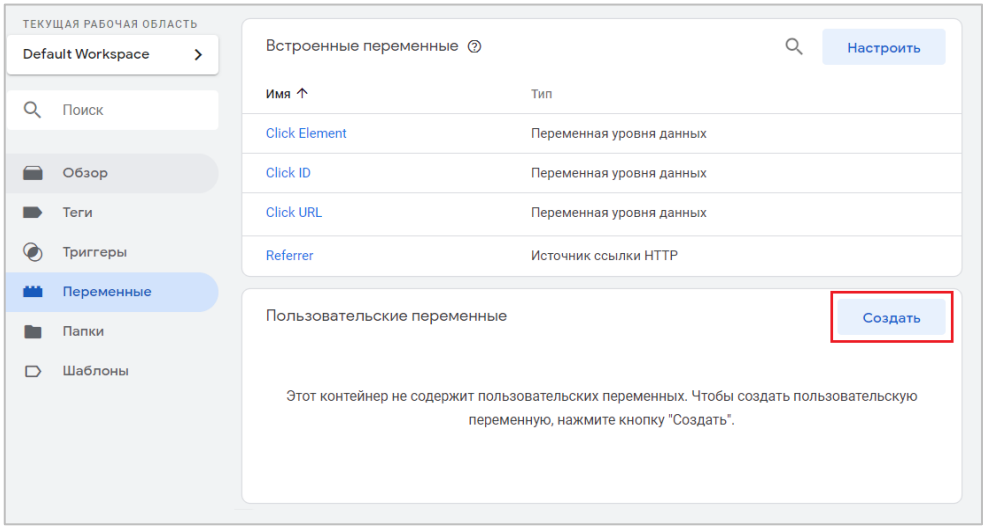


Рис. 468. Создание пользовательской переменной

Выберите тип переменной – **Настройки Google Analytics**. Эта переменная позволяет задавать настройки Google Analytics для использования с разными тегами.

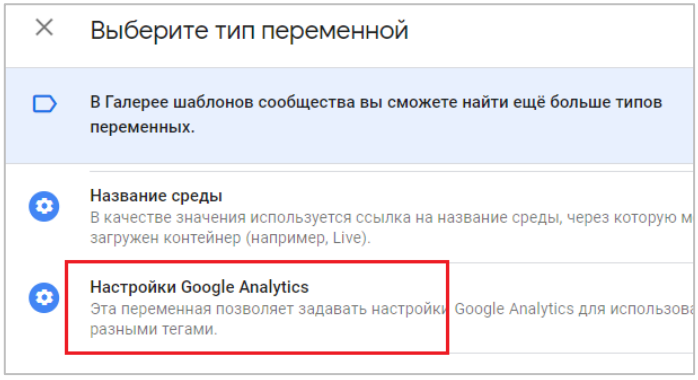


Рис. 469. Тип переменной — Настройки Google Analytics

Введите название переменной, идентификатор отслеживания (код Google Analytics) и нажмите **Сохранить**.

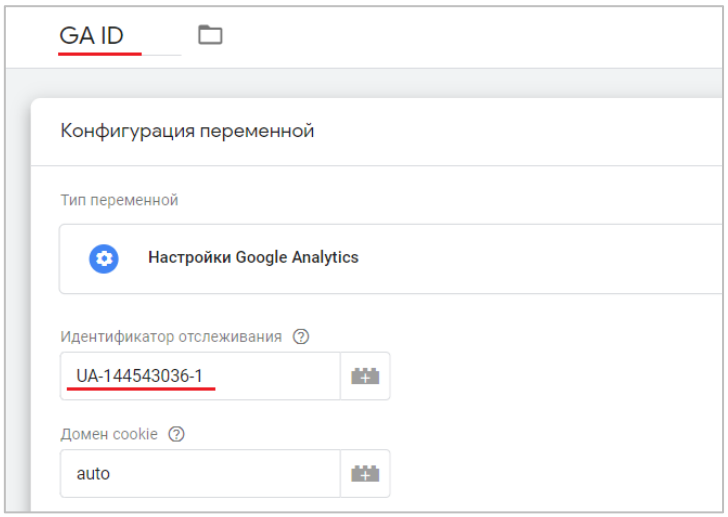


Рис. 470. Идентификатор отслеживания и Название переменной

Теперь нет необходимости постоянно копировать полный код Analytics. С этого момента у нас есть пользовательская переменная, которую можно использовать при внедрении тега Google Analytics.

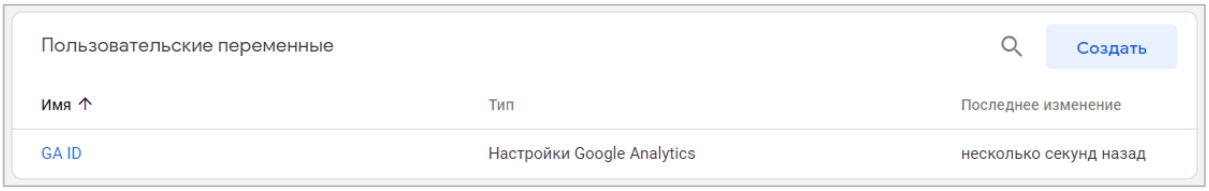


Рис. 471. Пользовательская переменная Настройки Google Analytics

Перейдите в меню **Теги** и создайте новый тег.

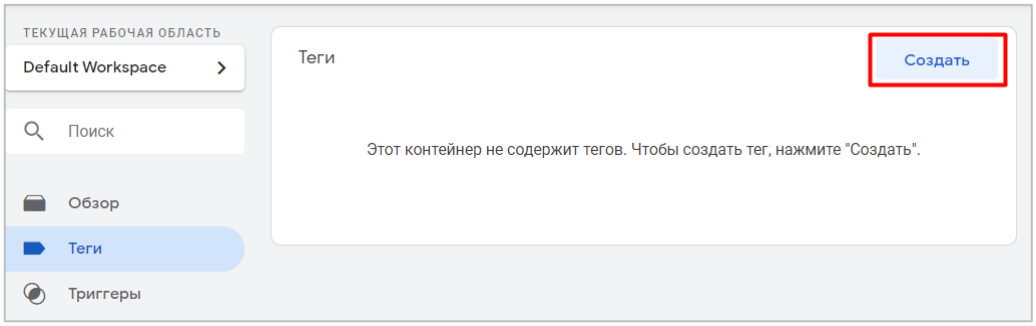


Рис. 472. Создание нового тега Google Analytics

Конфигурация тега – **Google Аналитика – Universal Analytics**.

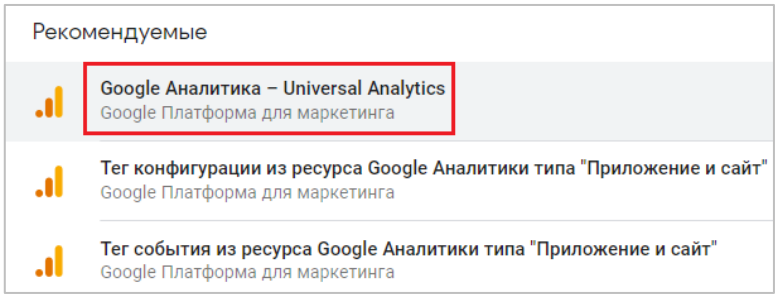


Рис. 473. Рекомендуемый тип тега Google Аналитика - Universal Analytics

Настройки:

- **Тип отслеживания** – Просмотр страницы;
- **Настройки Google Analytics** – Выбираем нашу переменную GA ID (автоматически заключается в двойные фигурные скобки, как шаблон);

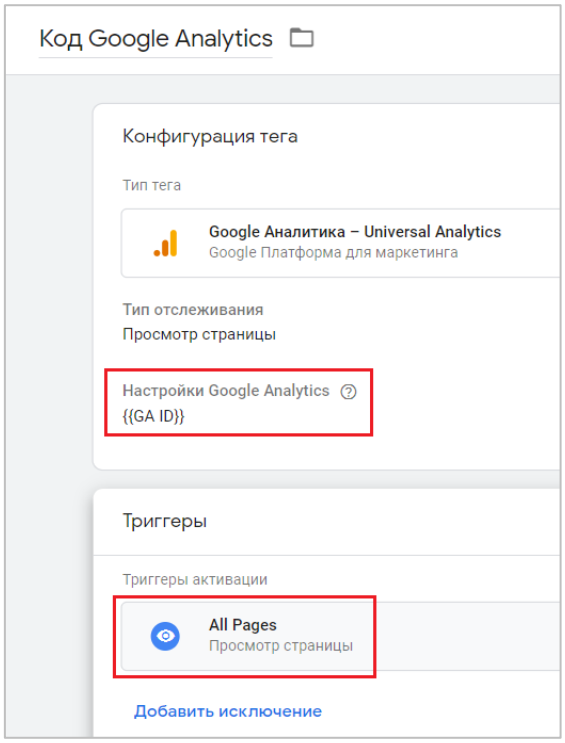


Рис. 474. Конфигурация тега

- **Триггер / Условие активации тега** – All Pages (на всех страницах).

Введите название тега и сохраните его. На этом установка кода отслеживания Google Analytics на сайт через Google Tag Manager завершена.

Проверьте корректность настройки кода отслеживания GA традиционным способом через сам Google Analytics без использования предварительного просмотра в отладчике. Для этого в правом верхнем углу нажмите **Отправить — Опубликовать** и **Далее**. Новая версия контейнера опубликована!

Теперь просто перейдите в интерфейс Google Analytics. В отчетах **В режиме реального времени** вы должны будете увидеть ваш заход. Для наглядности пометьте свой переход с помощью utm-меток. Можно перейти на различные разделы и посмотреть, фиксируются ли переходы по всем страницам сайта или нет.

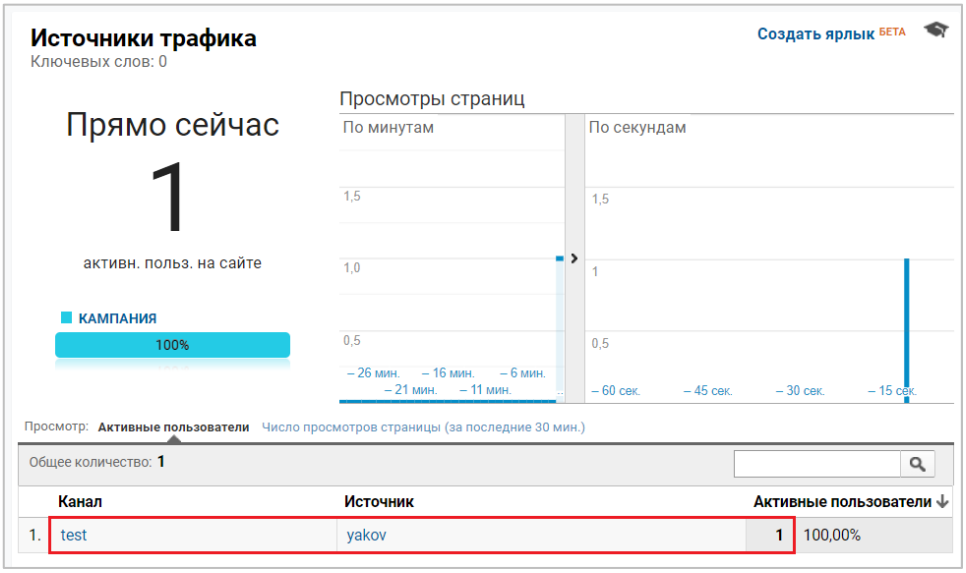


Рис. 475. Проверка установки кода Google Analytics в режиме реального времени

Данные передаются, тег настроен корректно. Проверить фиксацию данных также можно с помощью режима отладки Google Tag Manager или специальных браузерных расширений.

Установка счетчика Яндекс.Метрики

Перед установкой кода Метрики необходимо создать счетчик в аккаунте Яндекса и получить код отслеживания. Аналогично Google Analytics создайте новый тег. Только вместо рекомендуемого типа тега выберите специальный – **Пользовательский HTML**.

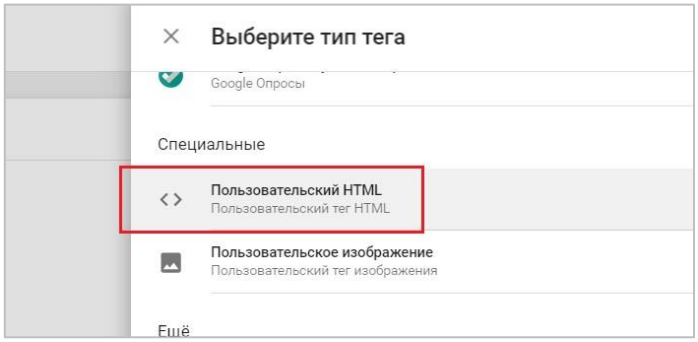


Рис. 476. Тип тега — Пользовательский HTML

Скопируйте код отслеживания из интерфейса Яндекс.Метрика и вставьте его в поле HTML. Не забудьте про условие активации тега –триггер **All Pages (Все страницы).** Нажмите **Сохранить**.

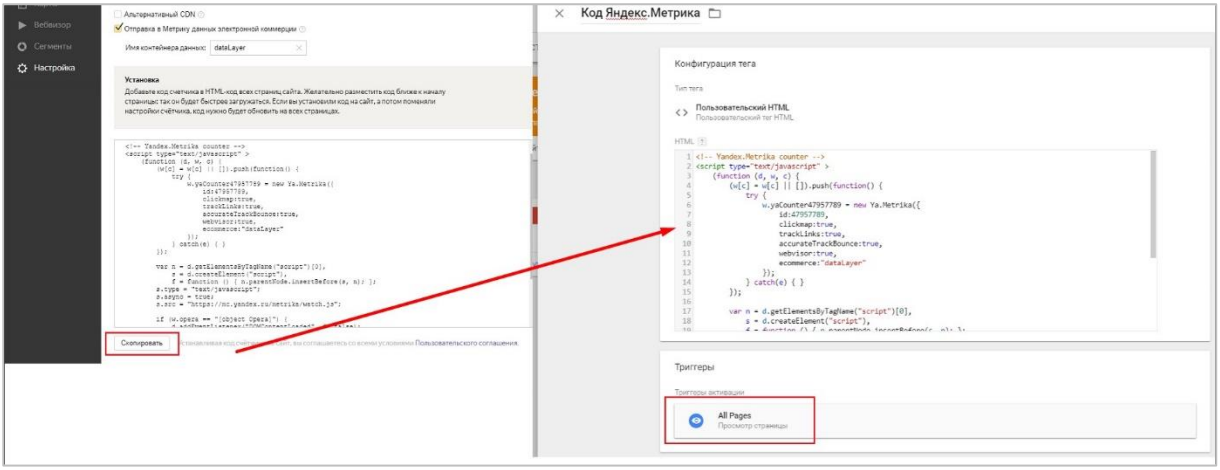


Рис. 477. Копирование кода из Метрики в Пользовательский HTML

В результате настроек вы получите два тега и два кода счетчика веб-аналитики.

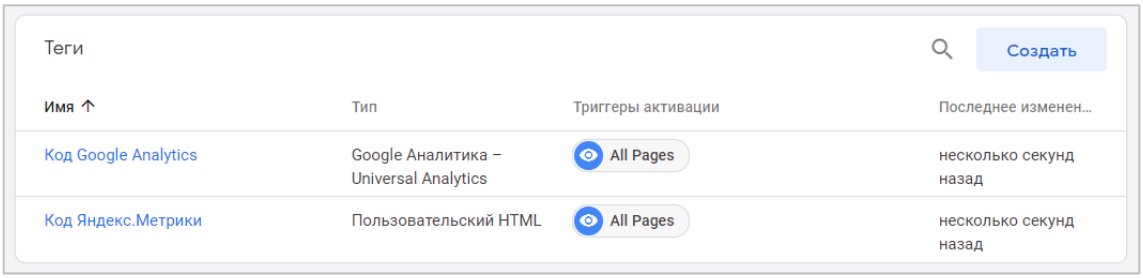


Рис. 478. Теги двух кодов счетчиков веб-аналитики

Чтобы данные передавались в Яндекс.Метрику, осталось только опубликовать новую версию контейнера на сайт. Действия прежние - в правом верхнем углу нажмите **Отправить — Опубликовать** и **Далее**. Новая версия контейнера опубликована.

Самый простой способ проверить корректность настройки счетчика Метрики – это обновить статус самого счетчика в списке всех счетчиков учетной записи.

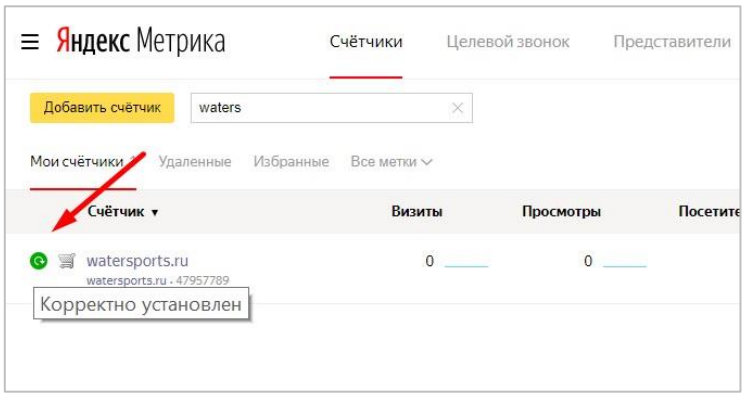


Рис. 479. Счетчик Яндекс.Метрики установлен корректно

Если вместо желтого или красного значка появился зеленый, значит он корректно установлен.

Есть еще один способ проверки установки кода счетчика – это добавление к адресу страницы параметра **_ym_debug** со значением **1**. В моем случае это **https://watersports.ru/?_ym_debug=1**

Далее необходимо вызвать консоль разработчика (в браузере Google Chrome кнопка F12) и перейти на вкладку **Console**. Если код установлен правильно, в консоли вы увидите номер вашего счетчика и данные, которые отправляет код.

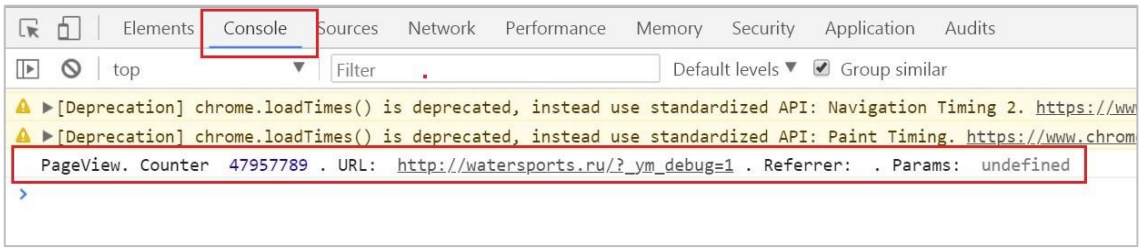


Рис. 480. Данные о просмотре страниц передаются

Установка счетчика Яндекс.Метрики с помощью шаблона тега

С появлением возможности создания пользовательских шаблонов в Google Tag Manager стал доступен дополнительный способ настройки Яндекс.Метрики.

Дэвид Вальехо (David Vallejo), автор популярного блога **thyngster.com** и расширения **Yandex Metrica Debugger**, создал шаблон тега Яндекс.Метрики и разместил его в галереи шаблонов сообщества. Тег является неофициальным, никакого отношения команда Яндекса к нему не имеет. Однако он может быть полезен интернет-маркетологам благодаря простой настройке. Давайте разберем процесс настройки Яндекс.Метрики с помощью решения Дэвида.

Для того, чтобы воспользоваться готовым шаблоном для Яндекс.Метрики, необходимо перейти в раздел **Теги** и создать новый тег. При выборе тега в самом верху перейдите по **В Галерее шаблонов сообщества вы сможете найти ещё больше типов тегов...**

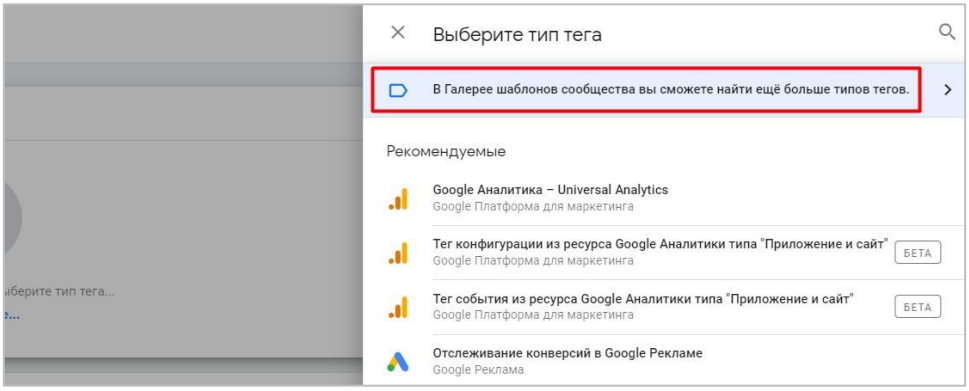


Рис. 481. Галерея шаблонов сообщества

Среди всех представленных тегов сторонних разработчиков найдите тег **Yandex Metrica** от David Vallejo:

Рис. 482. Тег Yandex Metrica

Аналогично можно было бы поступить, перейдя в раздел **Шаблон - Шаблоны тегов - Поиск в галерее**:

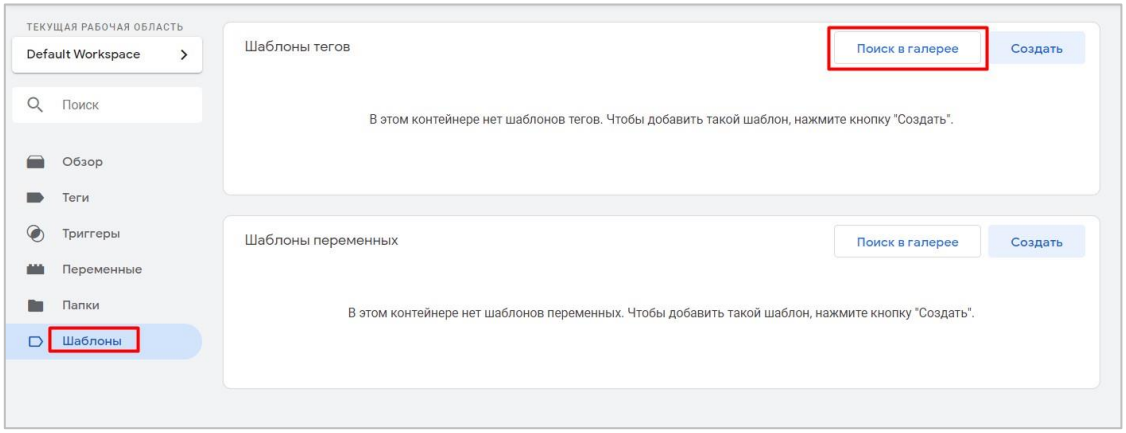


Рис. 483. Шаблоны - Поиск в галерее

Выбрав его, добавьте в свою рабочую область:

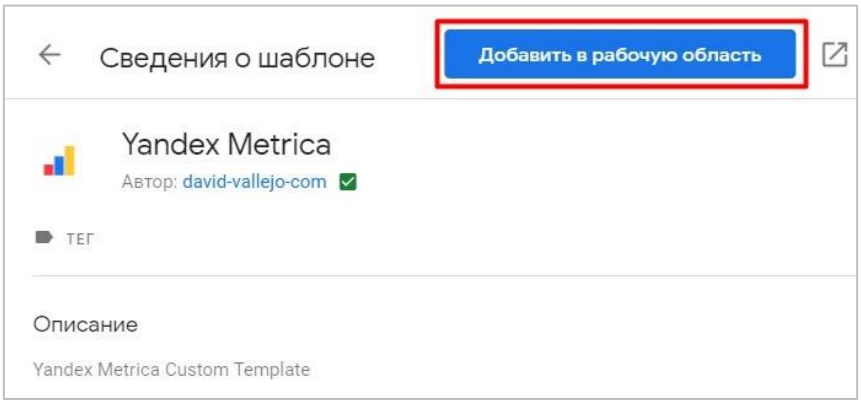


Рис. 484. Добавление тега в рабочую область

Примечание: все добавленные в рабочую область теги в дальнейшем будут доступны в разделе **Специальные** с меткой Галерея:

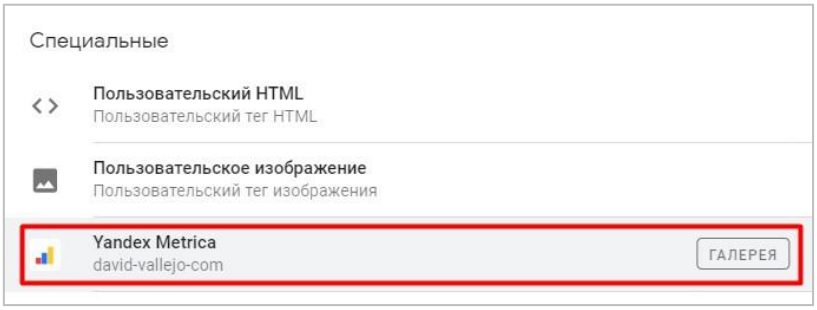


Рис. 485. Шаблоны тегов отображаются в специальных тегах

Подтвердите действие, нажав кнопку **Добавить**. После этого вам откроется меню с настройками тега:

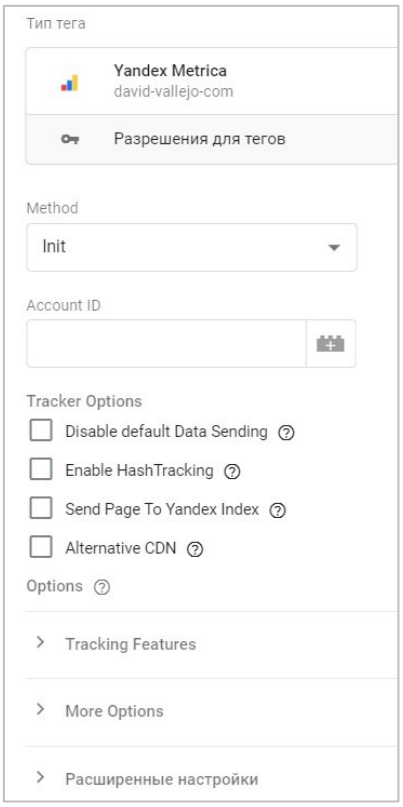


Рис. 486. Настройки Яндекс.Метрики в теге

Сам тег имеет английский язык интерфейса, что может усложнить первичную настройку для интернет-маркетолога. Дэвид в своем блоге написал краткое руководство (см. приложение) по его использованию (на английском языке), которое я адаптировал на русский язык.

На текущий момент тег Яндекс.Метрики поддерживает следующие методы (**Method**):

- **Init** - инициализация счетчика (для запуска Яндекс.Метрики);
- **Outbound Link** - отправка информации о переходе по внешней ссылке (extLink);
- **File Download** - отправка информации о загрузке файла (file);
- **Reach Goal** - достижение цели;
- **Hit** - отправка данных вручную о просмотрах для AJAX- и Flash-сайтов;
- **Session Parameters** - передача пользовательских параметров (params) в отчет **Параметры визитов;**
- **User Parameters** - передача пользовательских параметров (userParams) в отчет **Параметры посетителей**;
- **User ID** - передача идентификатора посетителя, заданного владельцем сайта (setUserID);
- **notBounce** - передача информации о том, что визит пользователя не является отказом.

Подробнее о каждом методе читайте в официальной справке Яндекса (см. приложение).

Поле **Account ID** необходимо для задания номера счетчика Яндекс.Метрики. Вы можете использовать пользовательскую переменную типа **Константа**, в которую добавьте значение счетчика, а затем добавьте его в соответствующее поле тега. В результате получится:

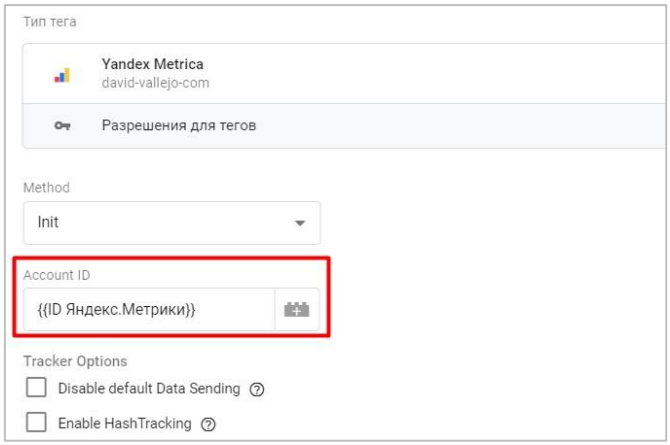


Рис. 487. Идентификатор счетчика Яндекс.Метрики в качестве константы

Далее идут настройки отслеживания (**Tracker Options**), которые можно включить/отключить. Они похожи на те, что представлены в самом интерфейсе Яндекс.Метрики, но отличаются некоторыми настройками, которые отдельно добавил Дэвид.

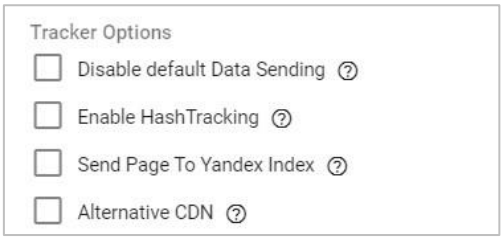


Рис. 488. Tracker Options

- **Enabling/Disabling default Data Sending** - за настройку отвечает параметр **defer**, который позволяет включать/отключать автоматическую отправку данных о просмотрах. Полезно в работе с SPA-страницами;
- **Enabling Hash Tracking** - отслеживание хеша в адресной строке браузера. Значение в Яндекс.Метрике по умолчанию - **false**;
- **Disabling sending pages to Yandex Index** - за настройку отвечает параметр **ut**, который запрещать отправку информации о существовании страницы поисковым роботам Яндекса. Использование данного параметра не гарантирует, что страница вообще не будет проиндексирована. Информация о существовании страницы может быть получена роботом из других источников, например, из Яндекс.Вебмастера.
- **Alternative CDN** - Альтернативный CDN.

Следующим разделом идут опции **Options - Tracking Features** (рекомендуемые отслеживания):

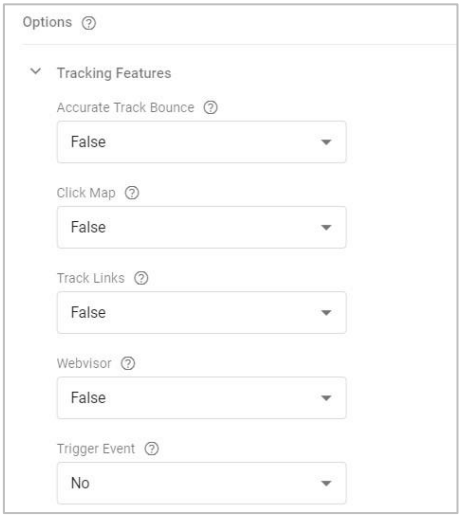


Рис. 489. Опции отслеживания Яндекс.Метрики

- **Accurate Track Bounce** - точный показатель отказов, который в Яндекс.Метрике имеет значение по умолчанию **true**.

Параметр может принимать значения:

- **true** - включить точный показатель отказов, событие о неотказе засчитывается через 15000 мс (15 с);
- **false** - не включать точный показатель отказов;
- **<N> (целое число)** - включить точный показатель отказов, событие о неотказе засчитывается через <N> мс

По умолчанию в этом теге стоит **False**. Нам нужно поменять его на **True**. Либо же мы можем задать любое другое значение, указав собственную переменную GTM.

- **Click Map** - карта кликов (**False** - выключить, **True** - включить). Значение в Яндекс.Метрике по умолчанию - **true**;
- **Track Links** - отслеживание переходов по внешним ссылкам (**False** - выключить, **True** - включить). Значение в Яндекс.Метрике по умолчанию - **true**;
- **Webvisor** - использование Вебвизора (**False** - выключить, **True** - включить). Значение в Яндекс.Метрике по умолчанию - **false**;
- **Trigger Event** - проверки готовности счетчика Яндекс.Метрики (**Yes** - да, **No** - нет) и запуск события с именем "yacounter" + counterID + "inited". Значение в Яндекс.Метрике по умолчанию - **false**;

В разделе **More Options** есть еще несколько настроек:

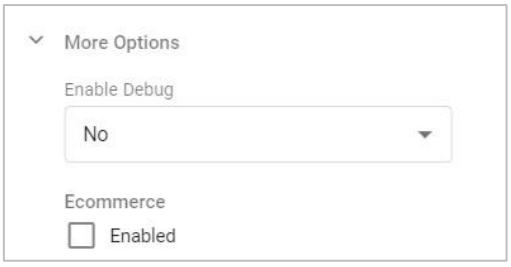


Рис. 490. More Options

- **Enable Debug** - режим отладки (**Yes** - включен, **No** - выключен). Его можно включить из тега инициализации без необходимости настройки параметра **QueryString**. Эта настройка является альтернативой включения режима отладки, когда к адресу страницы в конце дописывается параметр **_ym_debug=1**, чтобы проверить корректность передачи данных, в том числе и по целям Яндекс.Метрики;
- **Ecommerce** - позволяет отслеживать взаимодействие посетителей с товарами сайта. Чтобы статистика начала собираться, включите эту опцию, введите имя контейнера данных (по умолчанию **dataLayer**) и настройте на сайте передачу данных об электронной торговле.

После добавления всех настроек в тег Яндекс.Метрики не забудьте опубликовать обновленный контейнер Google Tag Manager.

Создание и управление тегом Google Аналитика - Universal Analytics

Бывало ли у вас такое, что при настройке через Google Tag Manager происходит активация всех переменных, триггеров и тегов, однако данные в Google Analytics почему-то не передаются? Причем это возникает при настройке абсолютно разных опций: пользовательских метрик, электронной торговли, динамического ремаркетинга, функции User ID и т.д. Вроде бы и тег Google Аналитика - Universal Analytics настроен верно, и счетчик подключен, и триггер срабатывает, в режиме отладки тоже все передается (показывает статус **Still running**), но данных в Аналитике нет?

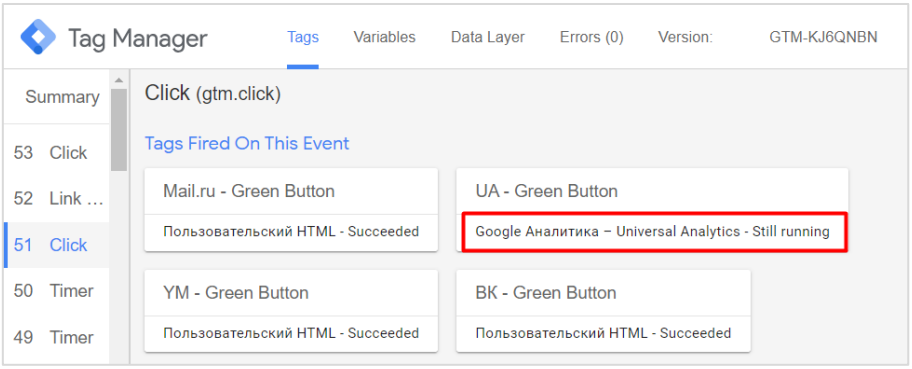


Рис. 491. Тег Universal Analytics активируется, но данных в Google Analytics нет

Уверен, что многие сталкивались с такой проблемой. В 90% случаев это происходит потому, что веб-аналитик не в том поле переопределяет идентификатор отслеживания Google Analytics. Другими словами, для корректного отслеживания всех действий важно понимать отличие переменной типа **Настройки Google Analytics** и типа **Константа**. С помощью этих двух переменных можно настроить 4 различных типа комбинаций тега Google Аналитика - Universal Analytics:

1. с помощью переменной **Настройки Google Analytics**;
2. с помощью переменной **Константа**;
3. комбинация первых двух + переопределение настройки в теге и перенастройка полей тега;
4. настройка тега Google Аналитика - Universal Analytics без каких-либо изначально созданных переменных.

Разберем каждый вариант подробнее.

1. Переменная Настройки Google Analytics

Эта переменная появилась в середине 2017 года и является пользовательской в Google Tag Manager.

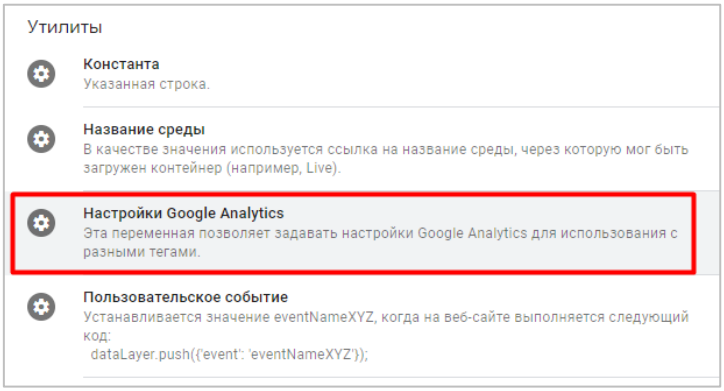


Рис. 492. Переменная Настройки Google Analytics

Она используется тогда, когда у вас есть n-ое количество предопределенных параметров (например, везде в тегах передается одна и та же метрика, пути к страницам, настройки междоменного отслеживания, анонимизация IP-адреса и т.д.), и вы не хотите каждый раз при создании нового тега Universal Analytics вводить все эти настройки. Все это можно указать один раз в переменной типа **Настройки Google Analytics** и затем размножать на большое количество других тегов.

Когда такой переменной не было, веб-аналитики при изменении какого-либо параметра внутри тега должны были заходить в каждый из них и менять настройки вручную. Что очень сильно раздражало и тормозило работу. С появлением такого типа переменной настройка тегов существенно упростилась.

В переменной **Настройки Google Analytics** нужно задать идентификатор отслеживания счетчика Google Analytics (1), а также выбрать **Дополнительные настройки** и указать те, которые являются одинаковыми и могут быть одновременно использованы в разных тегах (2).

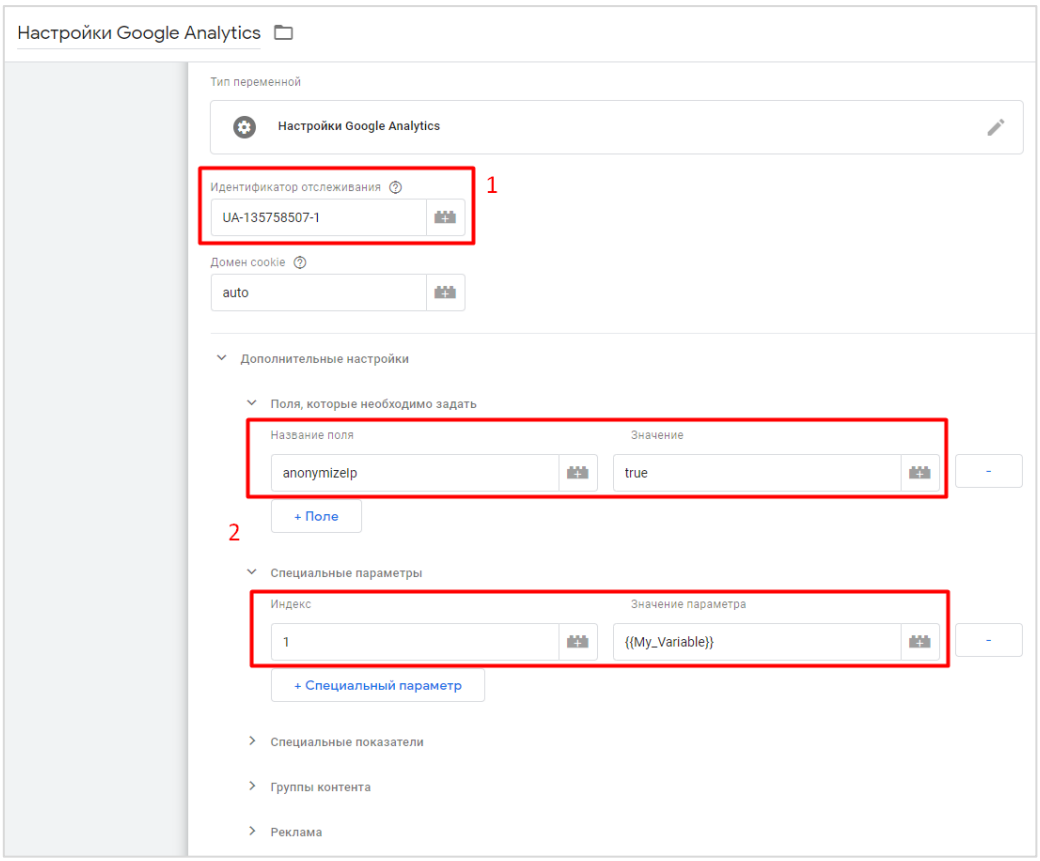


Рис. 493. Общие настройки для всех тегов Universal Analytics

Домен cookie. По умолчанию стоит *auto*, также, как и при отслеживании в Google Analytics, для поля **cookieDomain** значение *auto*. Если у вас на сайте нет других тегов Google Analytics, установленных с помощью *analytics.js* или GTM, оставьте значение *auto*. Если у вас есть другие теги, убедитесь, что используется одинаковое значение домена cookie.

После сохранения переменной ее можно использовать в теге Universal Analytics.

Важно: для этой переменной существует свое собственное поле, которое называется **Настройки Google Analytics**. Именно в него вы вставляете свою переменную.

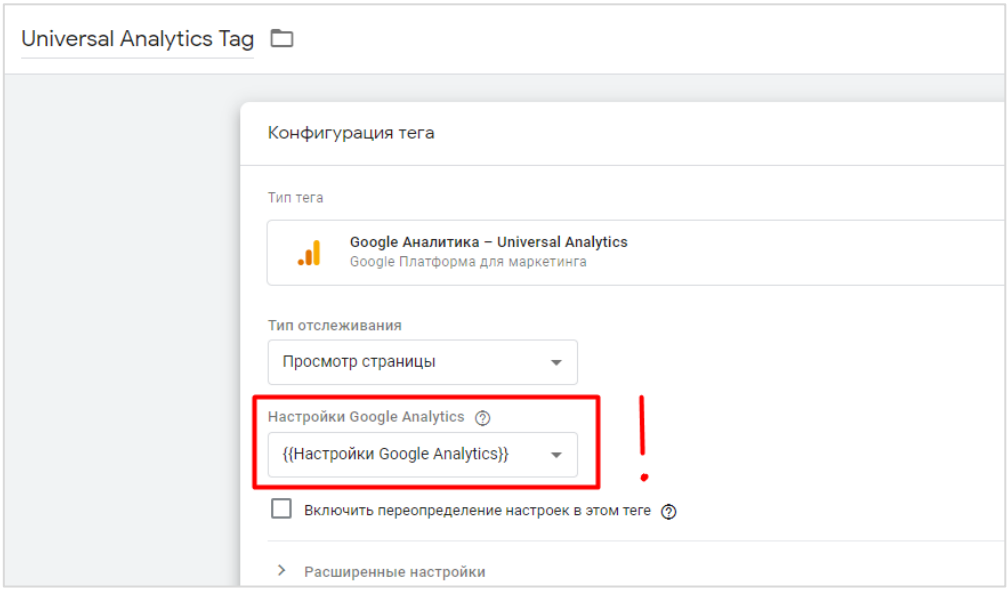


Рис. 494. Собственное поле для переменной Настройки Google Analytics

Для переменной типа **Настройки Google Analytics** используйте поле **Настройки Google Analytics**! При такой конфигурации тега доступны **Расширенные настройки**, но нет полей с дополнительными настройками. Галочка **Включить переопределение настроек в этом теге**, как правило, используется во втором способе с другой переменной.

2. Переменная Константа

В Google Tag Manager она выполняет функцию неизменяемой переменной, то есть будет постоянно принимать значение из соответствующего поля:

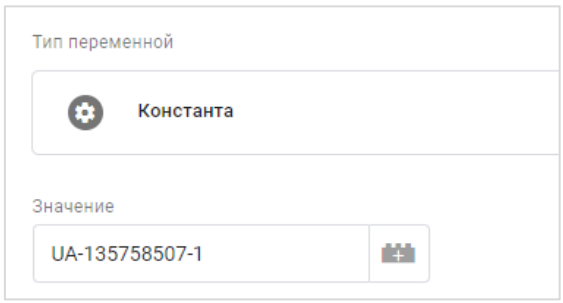


Рис. 495. Переменная Константа

Чаще всего ее используют именно для указания счетчика Google Analytics. Константу можно использовать и в другой переменной типа **Настройки Google Analytics**:

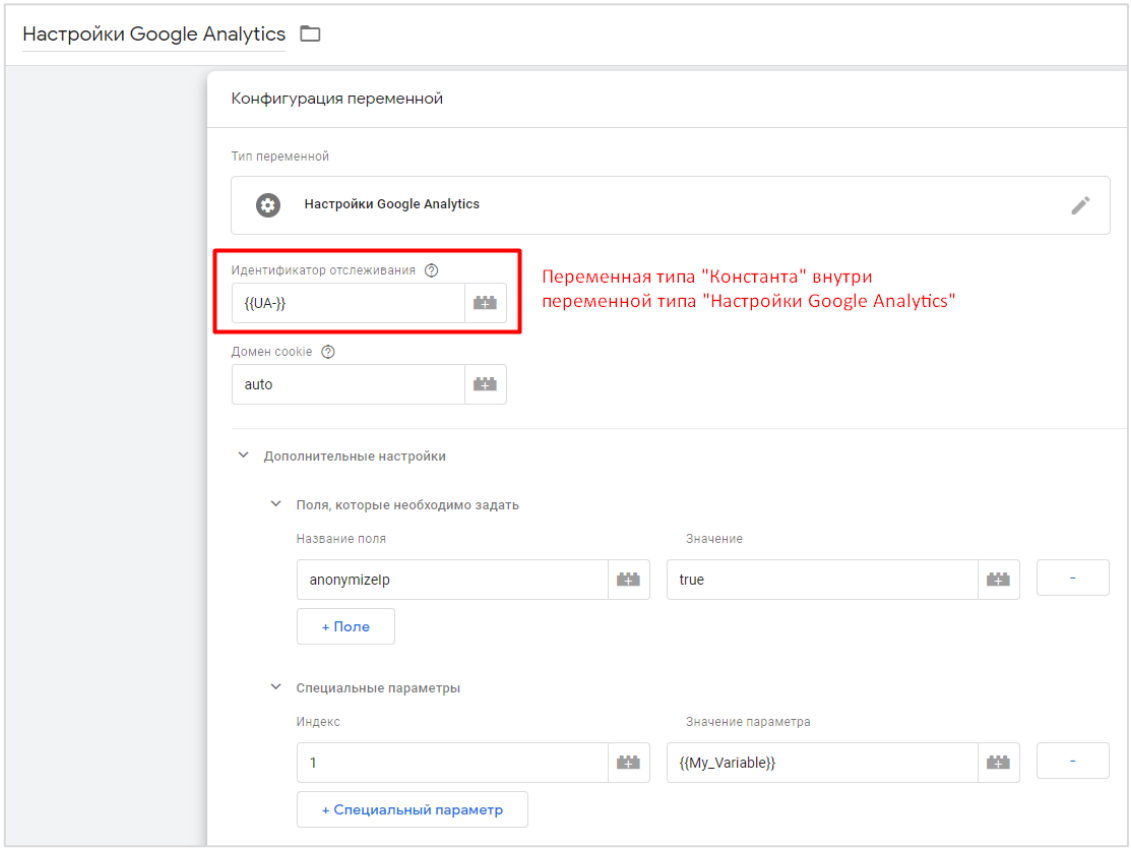


Рис. 496. Переменная внутри переменной

После сохранения переменной ее можно использовать в теге Universal Analytics.

Важно: для этой переменной существует своя последовательность действий:

- ставится галочка **Включить переопределение настроек в этом теге (1)**;
- указывается **идентификатор отслеживания с типом Константа (2)**.

Поле **Настройки Google Analytics** игнорируется.

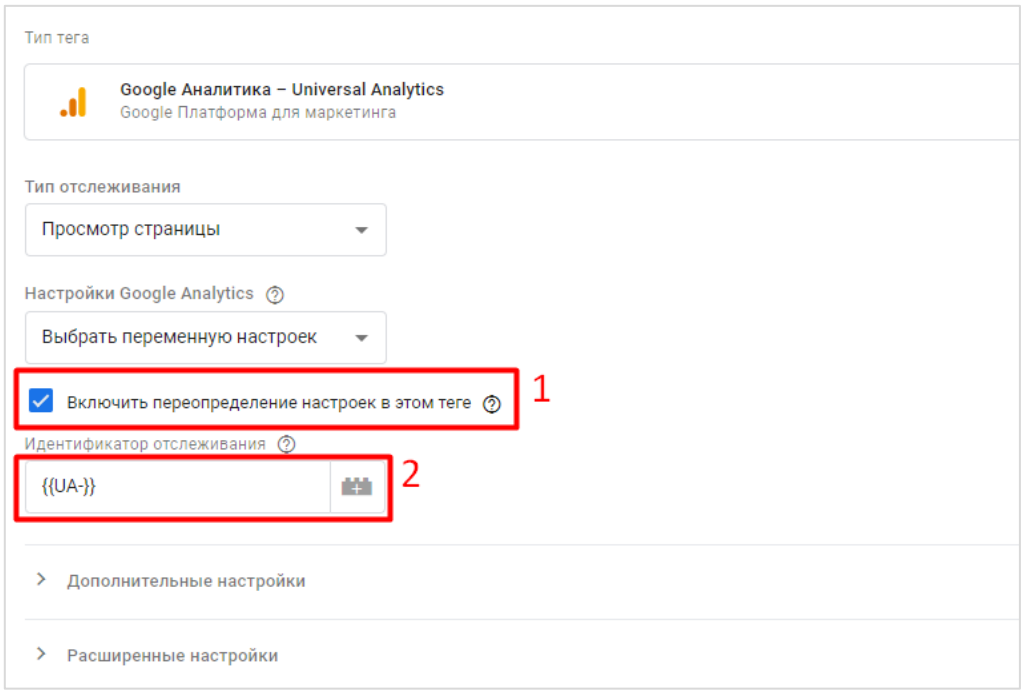


Рис. 497. Переопределение настроек и добавление идентификатора отслеживания

При такой конфигурации тега доступны и **Дополнительные настройки**, и **Расширенные настройки**. Внутри тега вы сможете задать все те параметры, которые вам необходимы.

Если вы создали переменную типа **Настройки Google Analytics**, но при этом желаете добавить еще некоторые дополнительные настройки в одном из тегов, вы можете использовать комбинацию этих двух способов.

3. Комбинация первых двух способов + переопределение настроек в теге и перенастройка полей тега

В этом случае вы добавляете переменную типа **Настройки Google Analytics** в собственное поле, ставите галочку **Включить переопределение настроек в этом теге**, но поле с идентификатором отслеживания оставляете пустым.

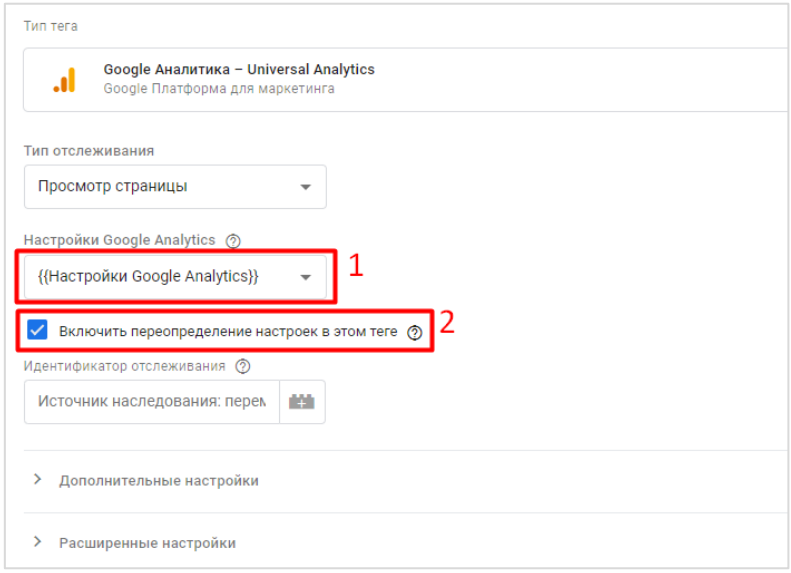


Рис. 498. Комбинация двух способов

При такой конфигурации тега доступны и **Дополнительные настройки**, и **Расширенные настройки**. Внутри тега вы сможете задать все те параметры, которые вам необходимы.

Именно здесь чаще всего и возникает ошибка у веб-аналитиков, которые настраивают тег Universal Analytics. Чтобы отредактировать **Дополнительные настройки**, они ставят галочку переопределения, а в поле **Идентификатор отслеживания** добавляют переменную типа **Настройки Google Analytics**. Этого делать нельзя!

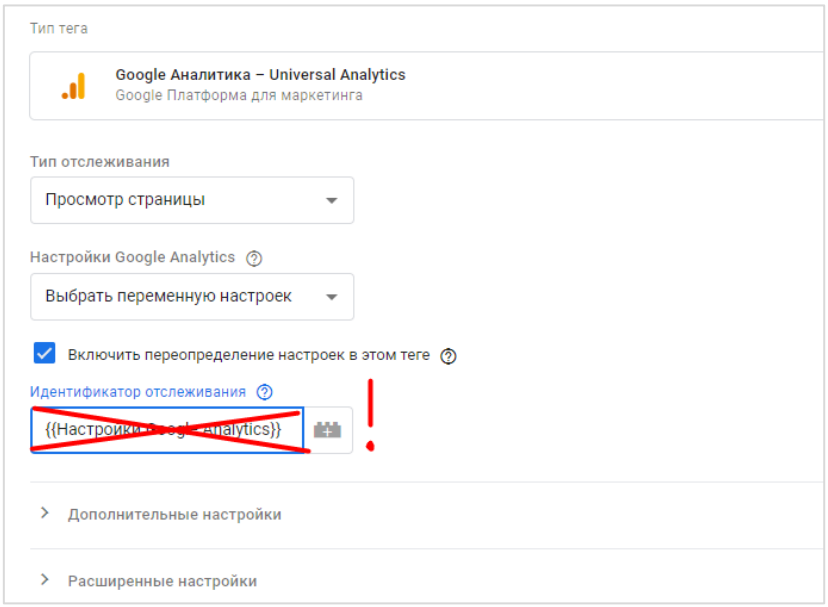


Рис. 499. Настройки Google Analytics не вставляются в поле Идентификатор отслеживания

Казалось бы, маленькая деталь, но именно такая настройка не передает данные в аккаунт Google Analytics. А все потому, что переменная типа **Настройки Google Analytics** была добавлена в другое поле с именем **Идентификатор отслеживания**. В него можно вставлять или переменную типа **Константа**, или же просто код счетчика GA. Собственно, в этом и заключается 4 способ.

4. Настройка тега Universal Analytics без каких-либо изначально созданных переменных

Для реализации этого способа никаких переменных можно не создавать. Вы просто добавляете новые тег типа Universal Analytics, а в поле **Идентификатор отслеживания** (с галочкой **Включить переопределение настроек в этом теге**) вставляете вручную свой код счетчика Google Analytics.

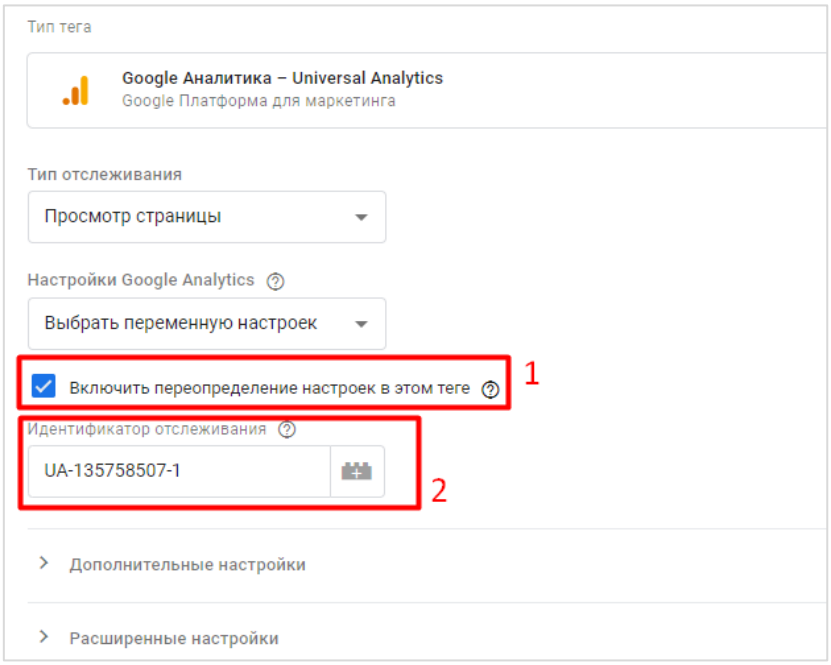


Рис. 500. Добавление идентификатора отслеживания напрямую

Настройка показателя отказов

Показатель отказов в Google Analytics — сеанс с просмотром только одной страницы. Даже если вы находились на странице несколько минут, но не совершили никаких взаимодействий, а просто читали информацию и затем вышли, будет засчитан отказ.

Разберем несколько примеров. На сайте было 3 пользователя:

1. Пользователь зашел на сайт. Данные о просмотре страницы отправились в Google Analytics. Пользователь ознакомился с информацией и покинул сайт, никуда не перейдя дальше. Analytics не зарегистрировал больше никаких событий и поэтому для пользователя показатель отказов равен 100%;
2. Пользователь зашел на сайт. Данные о просмотре страницы отправились в Google Analytics. Затем пользователь перешел на вторую страницу. Данные о просмотре другой страницы также отправились в отчеты GA. Таким образом в рамках одной сессии в GA было отправлено два обращения, поэтому для этого пользователя показатель отказов равен 0%;
3. Пользователь зашел на сайт. Данные о просмотре страницы отправились в Google Analytics. Далее пользователь добавил товар в корзину (это действие мы отслеживаем в GA как событие). В результате событие сработало и данные отправились в Analytics. После добавления в корзину пользователь покинул сайт. Получается пользователь был на одной странице и отправил два хита в инструмент веб-аналитики. У такого пользователя показатель отказов будет равен 0%.

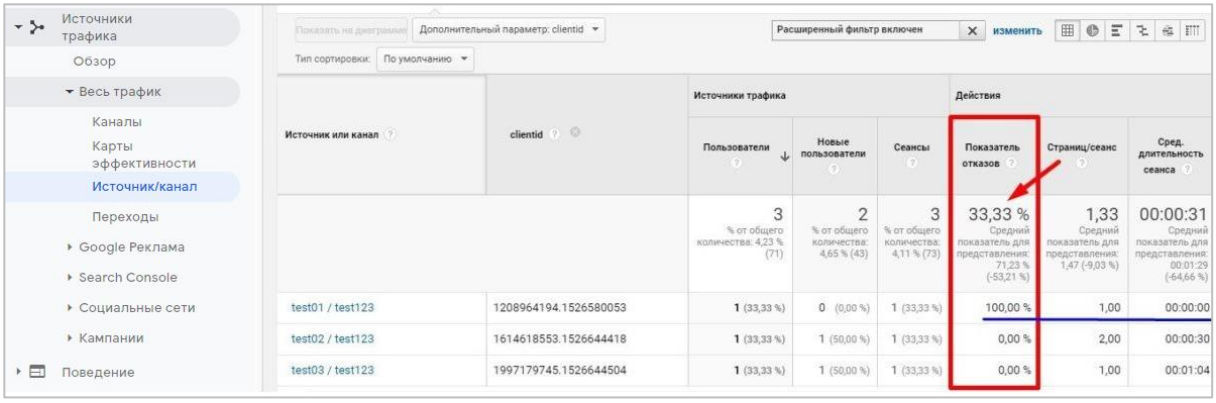


Рис. 501. Общий показатель отказов для всех трех пользователей - 33%

Построим отчет в Google Analytics. На примере выше общий показатель отказов для всех трех пользователей составляет 33%. И с точки зрения уравнивания пользователей в плане поведенческих факторов это не совсем правильно. Например, пользователь, который перешел на сайт и внимательно изучал страницу сайта (например, 10 мин.), не сделал ни одного действия, которое отправило данные в GA, и вышел из сайта - показатель отказа будет равен 100% и средняя длительность сеанса будет 00:00:00. И другой пример: пользователь перешел на главную страницу сайта, пробыл там 10 секунд, перешел на вторую страницу, просмотрел информацию в течение еще 10 секунд и затем покинул сайт. Показатель отказов в этом случае будет равен 0%.

Есть существенное отличие при подсчете этого показателя в Google Analytics и Яндекс.Метрика. В Метрике отказом считается посещение, в котором пользователь просмотрел всего одну страницу и посвятил ее просмотру менее 15 секунд. Во всех остальных случаях отказа не будет, даже если пользователь покинул страницу через 17 или 45 секунд после захода на сайт.

Из-за двойного условия показатель в Яндекс.Метрике ниже, чем в Google Analytics. Для того чтобы сделать его одинаковым в двух системах веб-аналитики, достаточно отправить данные в Google Analytics как событие через 15 секунд после того, как пользователь перешел на сайт.

Настроить «точный показатель отказов» в Google Analytics можно путем добавления дополнительного фрагмента кода вручную в код счетчика Google Analytics. В зависимости от библиотеки, которую вы используете (gtag.js или analytics.js), конструкция будет отличаться. Подробнее о том, как это сделать без использования Google Tag Manager, читайте в этом материале (см. приложение).

Для настройки показатель отказов через диспетчер тегов Google необходимо воспользоваться триггером **Таймер** со следующими настройками:

- **Имя события:** gtm.timer
- **Интервал:** 15000 (15000 миллисекунд = 15 секунд)
- **Ограничение:** 1 (количество активаций триггера на странице)
- **Включить триггер при выполнении всех этих условий:** Page URL соответствует регулярному выражению .* (означает, что триггер будет запускаться на всех страницах сайта)
- **Активация триггера** – Все таймеры

В Google Tag Manager это выглядит так:

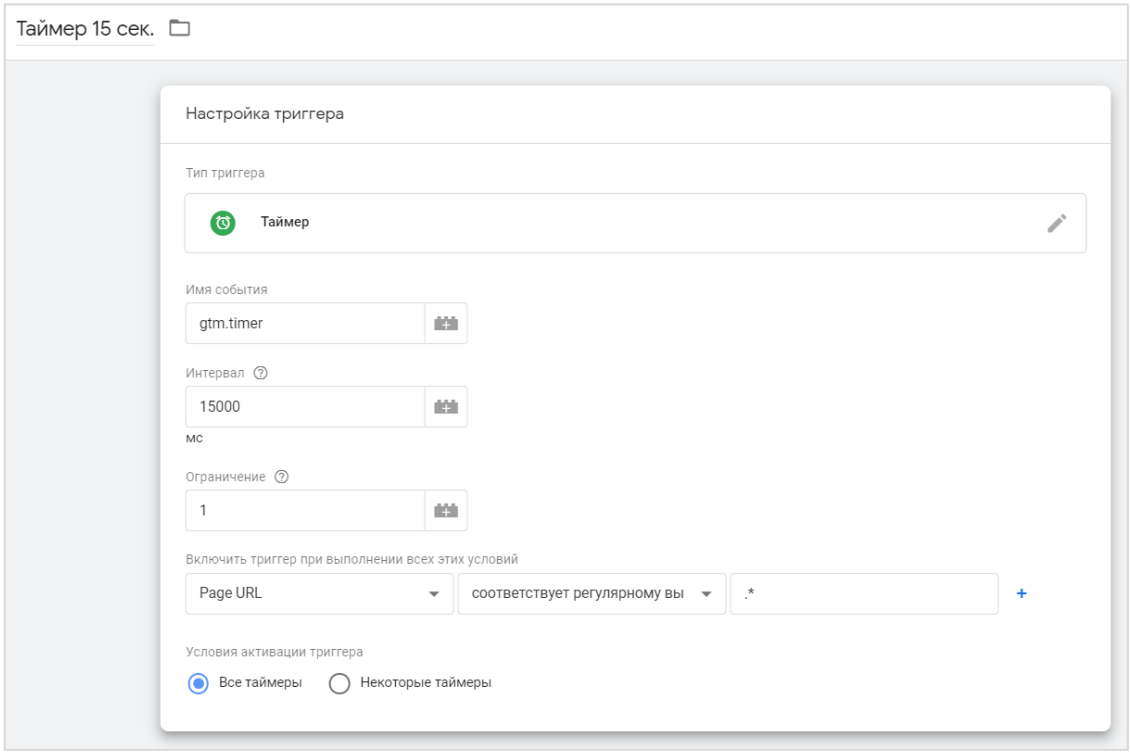


Рис. 502. Настройки триггера Таймер

Осталось создать тег типа **Google Аналитика – Universal Analytics** с типом отслеживания **Событие**, который будет иметь такие настройки:

- **Тип отслеживания** - Событие
- **Категория** – Таймер
- **Действие** – 15 сек.
- **Триггер активации** –Таймер, который мы создали шагом ранее

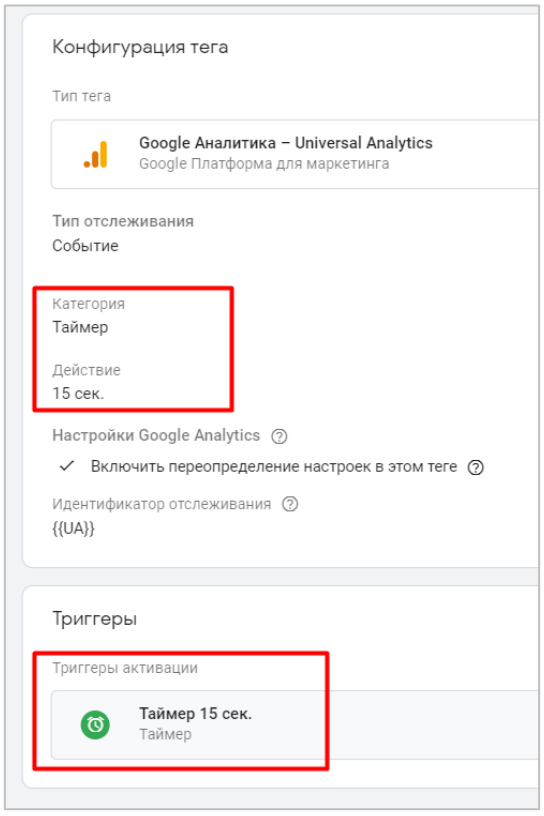


Рис. 503. Настройки тега Google Аналитика – Universal Analytics

Категория и Действие могут быть другими, заданы произвольно. Вы также можете использовать Ярлык и Ценность для передачи дополнительной информации в Google Analytics.

Для проверки фиксации данного события через 15 секунд используйте группу отчетов "В режиме реального времени". Посетив сайт, перейдите в раздел "События".

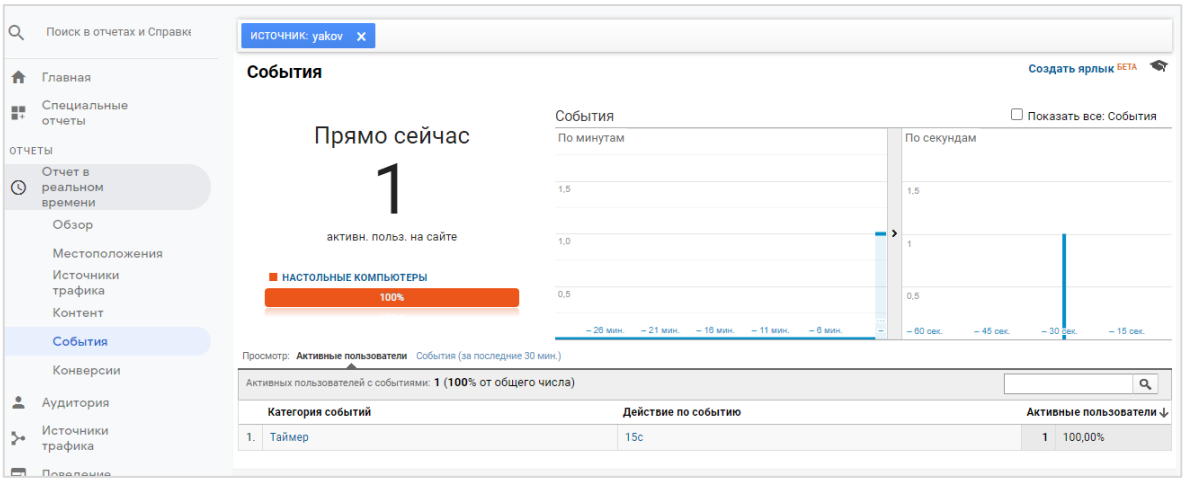


Рис. 504. Проверка события с помощью отчета В режиме реального времени

Настройка глубины прокрутки

Поведенческие характеристики являются важным инструментом для оценки вовлечения пользователей в контент вашего сайта. Среди них можно выделить несколько наиболее важных, на которые следует обращать внимание в первую очередь. Это *показатель отказов, средняя длительность сеанса, страниц/сеанс*.

Про первый показатель вы уже знаете из предыдущей главы.

Показатель **Средняя длительность сеанса** - отношение суммарной длительности пребывания пользователей на сайте на количество сеансов.

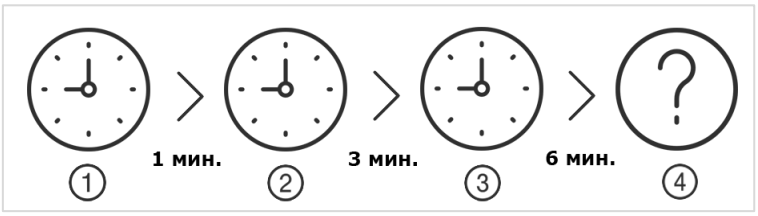


Рис. 505. Пример подсчета длительности сеанса

Для понимания подсчета этой величины разберем наглядный пример одного сеанса на примере просмотра 4 страниц:

1. пользователь зашел на сайт в 15:00 и просматривал первую страницу. В этот момент открылся новый сеанс;
2. в 15:01 он перешел на следующую страницу. Длительность его сеанса на сайте составляет 1 минуту;
3. в 15:04 система аналитики зафиксировала переход на другую страницу. Длительность сеанса составляет 4 минуты;
4. еще через 6 минут (в 15:10) он перешел в другой подраздел сайта, побыл там какое-то время и вышел. Может 1 минуту, может 20. Неизвестно.

Google Analytics не получил никаких данных о взаимодействиях на последнем шаге, а следовательно, мы не знаем, сколько времени пользователь провел на последней странице. Поэтому сеанс будет длиться с 15:00 до 15:10 и его продолжительность составит 10 минут = 600 секунд.

Длительность сеанса рассчитывается без учета пользователя после последнего взаимодействия. При обработке данных стоит не забывать про погрешность измерений, поскольку 10 минут – это величина, которая передалась бы кодом отслеживания Google Analytics в отчеты. На самом деле, пользователь мог открыть страницу и перейти на другую вкладку браузера, просматривая в этот момент не наш сайт.

Показатель **Средняя длительность сеанса** наряду с **Страниц/сеанс** учитывает вовлеченность пользователя. Если время взаимодействия с сайтом возросло по сравнению с предыдущим периодом, значит пользователи стали больше времени проводить на ресурсе – им нравится новый дизайн, навигация, контент, функционал и прочее.

Показатель **Страниц/сеанс** – отношение числа просмотров страниц к количеству сеансов за отчетный период. Повторные просмотры одной страницы также учитываются.

Рис. 506. Глубина просмотра в Google Analytics

Данная величина характеризует вовлеченность пользователя на вашем сайте - насколько удобна постраничная навигация и интересен ли сам контент. Его норма напрямую зависит от типа сайта. Блоги, информационные порталы, интернет-магазины, как правило, имеют большее значение **Страниц/сеанс** по сравнению с другими сайтами, поскольку в своей структуре содержат большее количество страниц, товаров, статей.

В случае же с одностраничным сайтом глубина просмотра может составлять 1.0 – 1.5. Такие сайты состоят всего из нескольких страниц:

- главная страница сайта;
- так называемая Страница благодарности, Страница спасибо, которую пользователь видит после заполнения формы на сайте.

Ситуация, когда пользователь, перейдя из поисковой выдачи на сайт, не возвращается обратно в поиск, является наилучшей для Google и Яндекса, и подходит под многие типы веб-сайтов. Для оценки вовлеченности можно использовать еще один критерий – **глубина прокрутки (глубина скроллинга, глубина пролистывания, % скроллинга)**. Будь то это новостной сайт, блог, или LP со множеством экранов – глубина просмотра поможет вам оценить эффективность посадочной страницы. Вы всегда будете знать какой % скроллинга совершил пользователь, дошел ли он до того экрана/информации, которую вы хотели ему показать. И если нет, то произвести изменения в структуре контента.

Так уж довелось, что для решения этой задачи все чаще используют Google Tag Manager, поскольку в него уже заложен готовый функционал. Последовательность настройки глубины прокрутки такая:

- активация встроенных переменных;
- настройка триггера;
- создание тегов для Google Analytics.

Давайте разберем каждый из этапов более подробно.

1. Активация встроенных переменных

Для этого необходимо перейти в раздел **Переменные** и активировать встроенные переменные типа **Прокрутка**.

Рис. 507. Встроенные переменные типа "Прокрутка"

Переменные:

- **Scroll Depth Threshold** — возвращает значение порога прокрутки. В зависимости от настроек будет либо в процентах (%), либо в пикселях (px);
- **Scroll Depth Units** — переменная возвращает значение в пикселях (pixels) или процентах (percent), в зависимости от настроек триггера;
- **Scroll Direction** – переменная возвращает значение направления прокрутки (vertical или horizontal), в зависимости от настроек.

2. Настройка триггера

После активации переменных переходим на вкладку **Триггеры** и создаем триггер типа **Глубина прокрутки**:

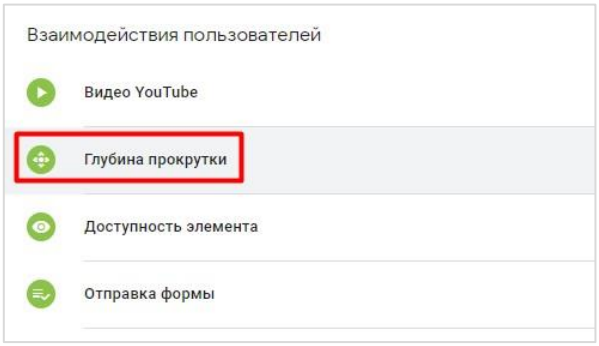


Рис. 508. Триггер Глубина прокрутки

Доступны следующие виды настроек:

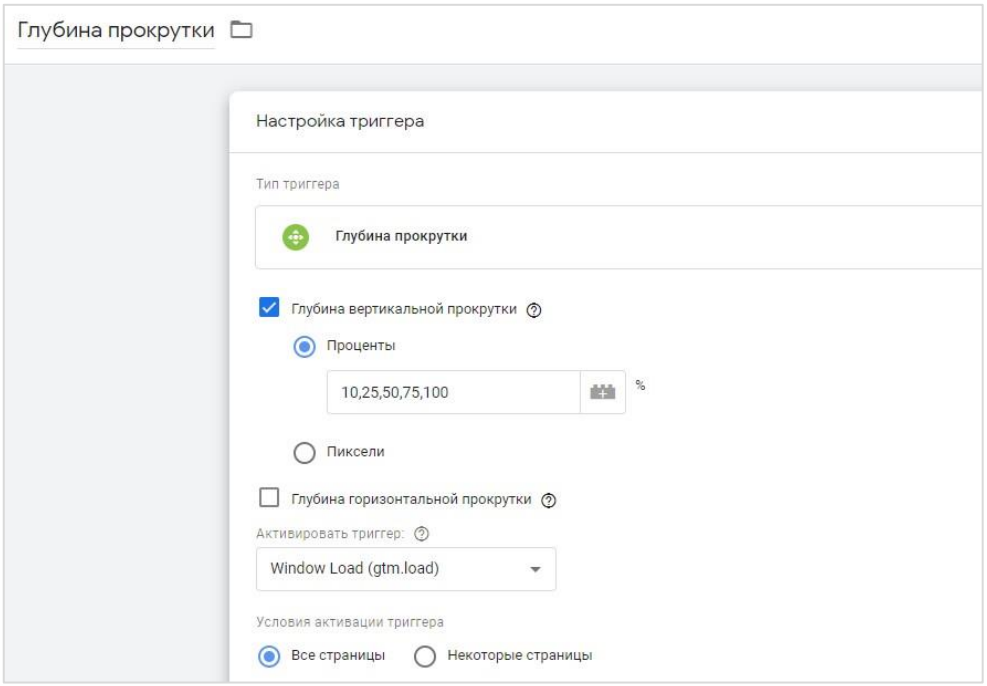


Рис. 509. Настройка триггера

Можно выбрать тип прокрутки (вертикальная или горизонтальная), а также задать пороги активации (проценты или пиксели). Условия активации триггера можно задать как на все страницы, так и на некоторые, а начать отслеживание выбранных взаимодействий по событию gtm.js (после загрузки страницы), gtm.dom (когда модель DOM готова) или по gtm.load (когда загрузится весь первоначальный контент страницы). Если вы не знаете, какое именно событие выбрать, оставьте по умолчанию Window Load.

В качестве примера я выберу наиболее популярный вариант настройки для сайтов - отслеживание вертикальной прокрутки на всех страницах с порогами 10, 25, 50, 75, 100 (см. рисунок выше).

3. Создание тегов

Перейдите на вкладке **Теги** и создайте тег типа **Google Аналитика - Universal Analytics** с типом отслеживания **Событие**:

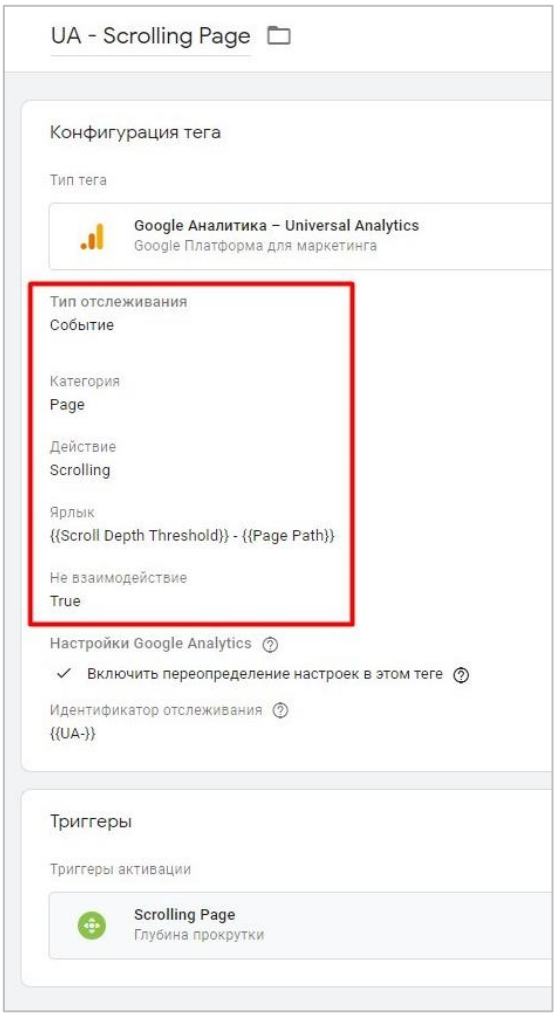


Рис. 510. Тег Google Аналитика – Universal Analytics

В **Категория** и **Действие** вы можете добавить произвольные значения. В Ярлык добавляем встроенную переменную **Scroll Depth Threshold** (ее можно добавить и в Категорию и Действие, кому как удобнее), в которой возвращается значение порога прокрутки (в процентах). В конце можно добавить % или любой другой текст (например, слово "процентов"), чтобы не забыть единицу измерения переменной.

В **Ярлык** также можно добавить и другие переменные. Я передаю вместе с порогом прокрутки URL-страницы, на которой совершено событие. Эта информация хранится в переменной **Page Path**. Таким образом, в Google Analytics я получу конструкцию вида 10% - /event-google-analytics. В настройке **Не взаимодействие** указываем **True** в том случае, если вы не хотите, чтобы порог глубины прокрутки в 10% влиял на показатель отказов.

Если вы хотите влиять на настройку **Не взаимодействие** и в зависимости от достижения определенного порога прокрутки событие влияло/не влияло на показатель отказов, нужно сделать еще один шаг и создать пользовательскую переменную типа **Таблица поиска**.

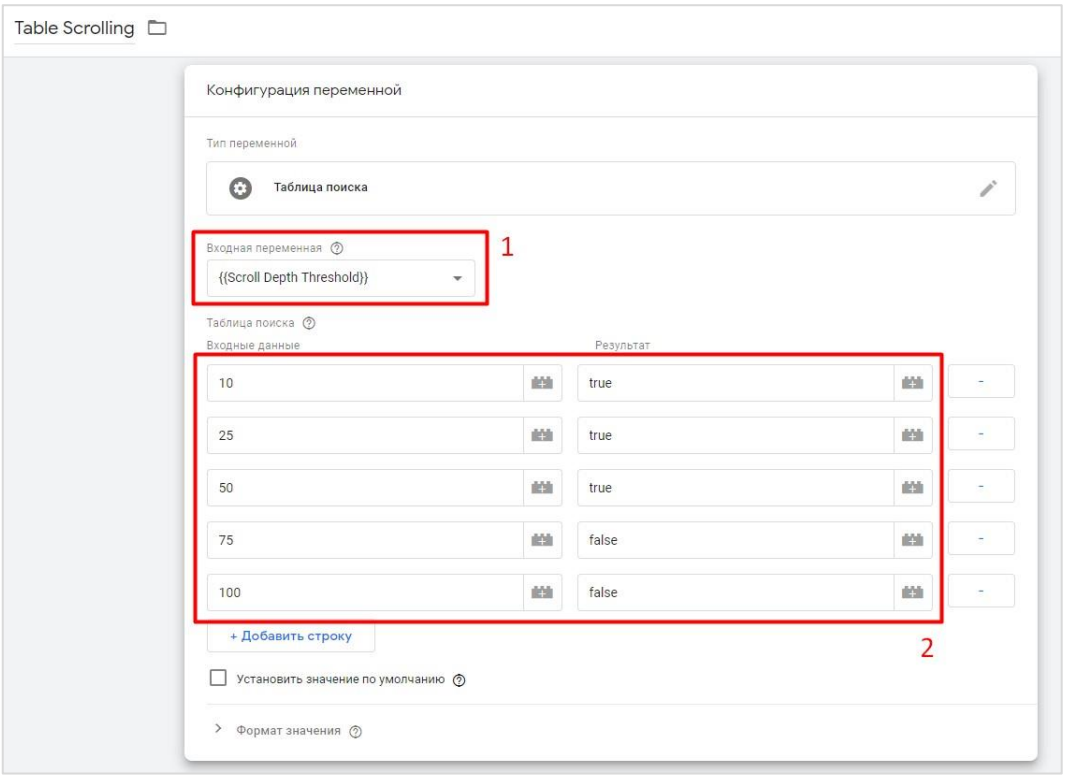


Рис. 511. Таблица поиска для глубины прокрутки

В таблице поиска во входных данных задаются конкретные значения. Поиск значений всегда является точным совпадением и чувствительным к регистру.

Как работает таблица поиска?

1. Задается входная переменная. В данном примере — это **{{Scroll Depth Threshold}}**;
2. Далее идет проверка по таблице поиска и входным данным;
3. Если переменная **{{Scroll Depth Threshold}}** принимает одно из значений из таблицы, то входная переменная **{{Scroll Depth Threshold}}** примет значение из поля Результат;
4. Если значение переменной **{{Scroll Depth Threshold}}** не найдено среди таблицы поиска, то будет использовано значение по умолчанию (если указано в настройках).

В примере с глубиной прокрутки - когда переменная **{{Scroll Depth Threshold}}** принимает значение 10, 25 или 50, то результат в переменной Не взаимодействие будет true (не будет влиять на показатель отказов). В случае с порогами 75 и 100 Не взаимодействие будет false (будет влиять на показатель отказов).

Все, что осталось сделать, это добавить в тег **Google Аналитика - Universal Analytics** в поле **Не взаимодействие** нашу таблицу поиска.

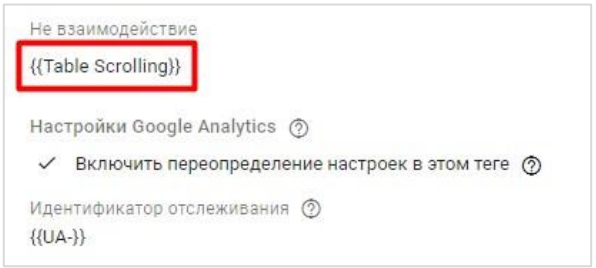


Рис. 512. Таблица поиска в поле Не взаимодействие

Триггер активации для данного тега - **Глубина прокрутки**. Сохраняем изменения.

Иногда требуется кастомизировать отслеживание глубины прокрутки. Например, использовать разные пороги прокруток на различных страницах (см. приложение).

Проверить корректность сбора данных и отслеживание можно с помощью режима отладки GTM и отчетов **В режиме реального времени** в Google Analytics.

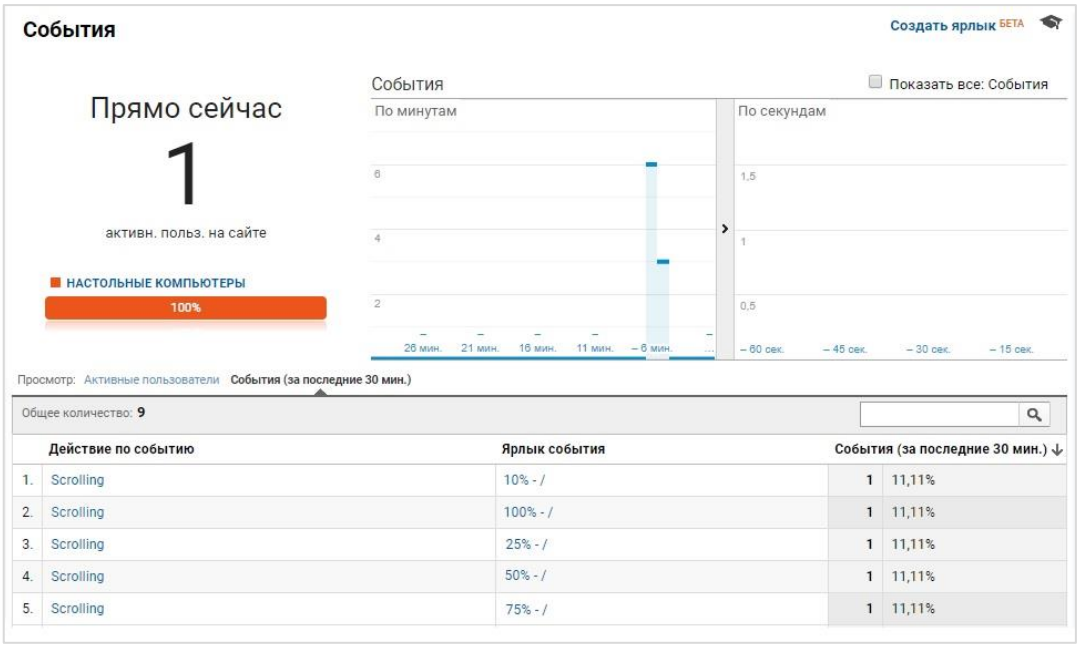


Рис. 513. Отчет В режиме реального времени

Отслеживание видео YouTube

Видеоформаты стремительно набирают популярность. Instagram, YouTube, Snapchat, TikTok, прямые эфиры - все это стало частью нашей жизни. Потребление видеоконтента продолжает расти в геометрической прогрессии. Увеличивается и количество устройств просмотра.

В связи с этим владельцы бизнесов стараются использовать видеоформат при продвижении собственных товаров и услуг. Кто-то создает промо-ролик, где рассказывает о своем продукте и его преимуществах, а кто-то заводит отдельный YouTube-канал, чтобы максимально долго удерживать контакт с целевой аудиторией.

Несомненно, контент такого типа очень сильно повышает вероятность покупки именно у вас, поскольку многие пользователи перед совершением оплаты предпочитают увидеть продукт вживую. YouTube — это как раз простое и эффективное решение донесения информации о товаре/услуге до конечного покупателя. А оставляя комментарии под видео, зрители формируют мнение, которое может служить мощным мотивирующим фактором при принятии решения относительно покупки не только для себя, но и для других пользователей.

Вспомните ролики с распаковкой товаров у Aliexpress, Joom, когда за коротенькое 15-20 секундное видео под красивую музыку показывают вам ускоренную распаковку какой-нибудь посылки. Или, наоборот, 10-15 минутные подробные видеообзоры конкретных товаров, которые заказывают производители у разных блогеров. А уж смотреть на распаковку игрушек на камеру от детей - вообще прелесть! К слову, это один из самых популярных жанров в детском сегменте YouTube.

Все это подтверждает тренд на видео. В последнее время я все чаще и чаще замечаю интернет-магазины, у которых к товарам добавлены не только изображения, но и видео.

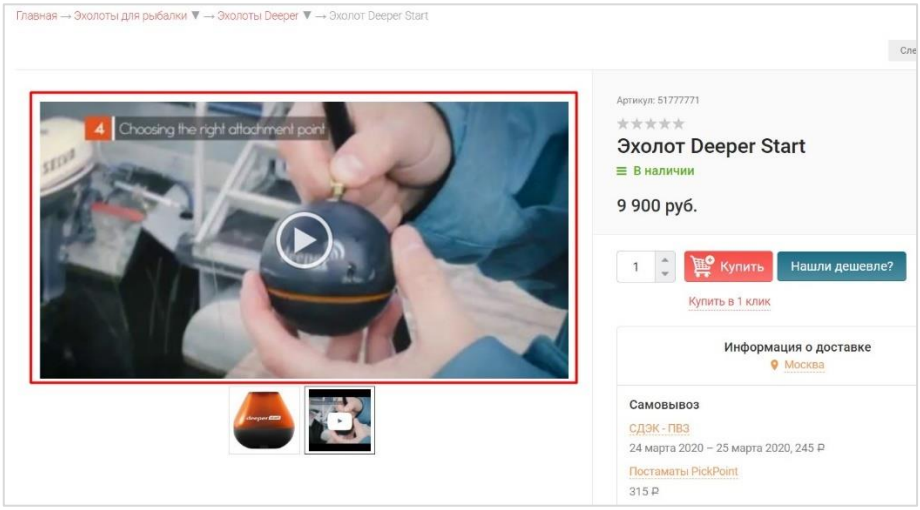


Рис. 514. Пример видеообзора товара в интернет-магазине

Такой подход позволяет не только повысить интерес пользователя к данному продукту, но и может служить триггером к увеличению конверсии. Еще запуск видео на сайте способствует повышению основных поведенческих факторов (длительность сеанса, снижение показателя отказов), а как следствие, улучшению ранжирования в органической выдаче.

Видео можно сделать самостоятельно (если есть навыки и желание), а можно заказать у специалиста. Однако прежде, чем это делать, можно загрузить простое, пробное и посмотреть как пользователи на него реагируют, настроив отслеживание с помощью Google Tag Manager.

Сделать это с появлением в 2017 году встроенных переменных и триггера стало совсем просто. До этого маркетологам и веб-аналитикам приходилось как-то выкручиваться: писать собственный скрипт для прослушивания событий, либо использовать готовый, любезно предоставленный на github каким-нибудь энтузиастом.

Давайте пройдемся по всем настройкам. Для начала на вкладке **Переменные** активируйте все встроенные переменные из раздела **Видео**.

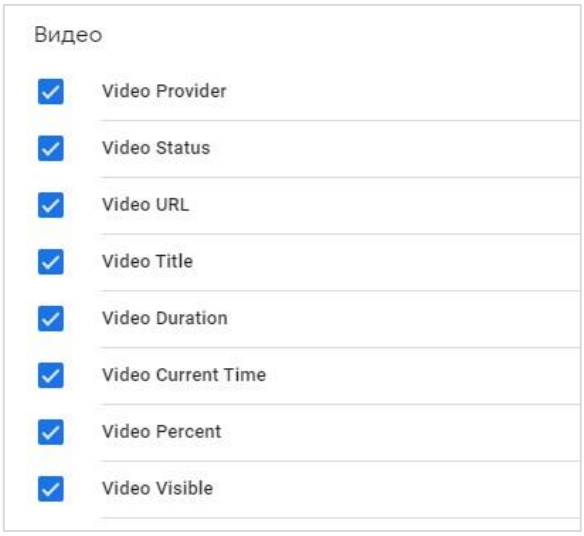


Рис. 515. Переменные типа Видео

Затем перейдите на вкладку **Триггеры** и создайте триггер типа **Видео YouTube**.

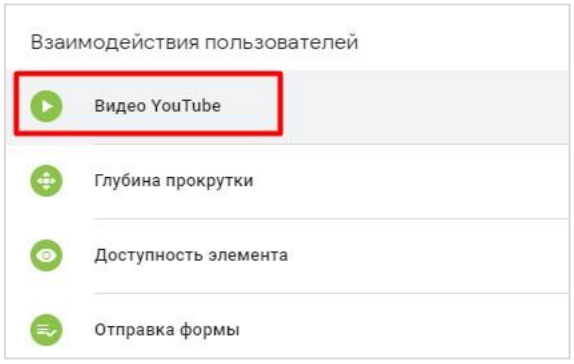


Рис. 516. Триггер Видео YouTube

Google Tag Manager добавил 8 переменных в эту категорию:

1. **Video Provider** — название платформы видео (YouTube);
2. **Video Status** — статус видео в момент регистрации события. Может быть: **Start, Complete, Pause, Seek, Buffering, Progress**;
3. **Video URL** — URL-адрес, ссылку на видео YouTube (https://www.youtube.com/watch?v=...);
4. **Video Title** — название видео;
5. **Video Duration** — общая продолжительность видео (в секундах);
6. **Video Current Time** — текущее время видео (в секундах), в которое произошло событие;
7. **Video Percent** — значение воспроизведенного видео (в процентах) на момент, когда сработало событие;
8. **Video Visible** — значение видимости видео в окне браузера. Если видео отображается в области просмотра, результатом будет значение true, если же в другой области (например, в нижней части страницы, на фоновой вкладке) – false.

В настройках триггера отмечаем то, что хотим отслеживать:

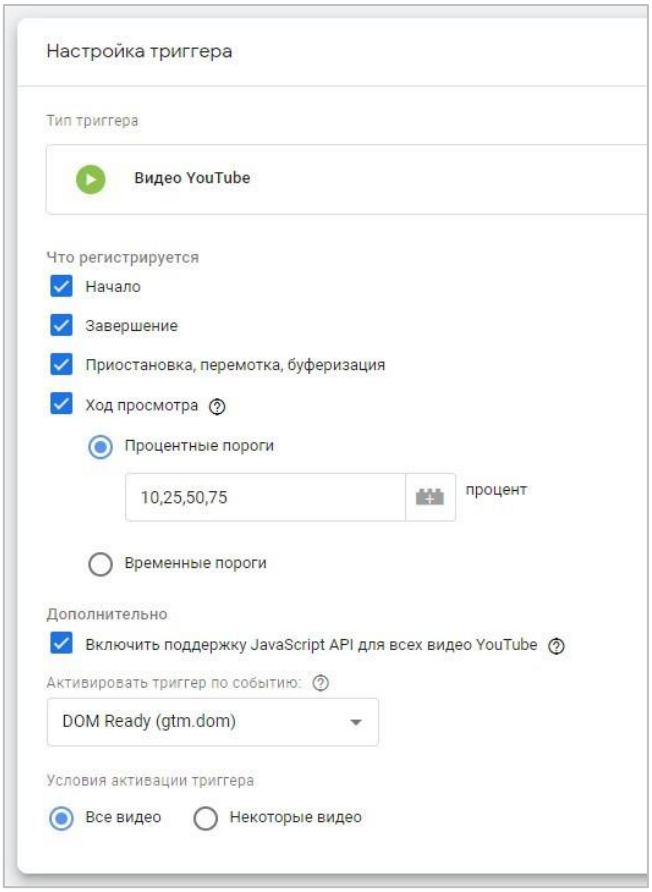


Рис. 517. Настройки триггера

Триггер имеет встроенные события:

- **Начало (Start)** – пользователь начинает просмотр видео;
- **Завершение (Complete)** – пользователь достигает конца видео;
- **Приостановка (пауза), перемотка, буферизация (Pause, Seeking, Buffering)** – пользователь останавливает, перематывает видео или когда происходит буферизация;
- **Ход просмотра (Progress)** – пользователь проходит процентный (заданный в поле) или временный порог (время измеряется в секундах). Целые положительные числа указываются через запятую, как на рисунке выше.
- **Дополнительно - Включить поддержку JavaScript API для всех видео YouTube**. Установив этот флажок, вы включите YouTube iFrame Player API. В результате ко всем URL видеопроигрывателя YouTube будет добавлен параметр enablejsapi = 1 для управления проигрывателем через iframe или JavaScript.
- **Активировать триггер по событию:** DOM Ready (gtm.dom). Можно выбрать один из 3 способов начала отслеживания - **Container Load (gtm.js)** - после загрузки страницы, DOM Ready (gtm.dom) - после обработки модели DOM и Window Load (gtm.load) - после загрузки контента страницы.

Подробнее о параметрах проигрывателя YouTube и IFrame Player API читайте в официальной документации Google (см. приложение).

Примечание: ход просмотра относится ко всей длине видео. Поэтому когда вы настроите триггер на процентные пороги (например, на 25%, 50%, 75%), или на временные пороги (например, на 15, 30, 60, 120 секунд), он сработает и в случае перемотки пользователем на соответствующую метку. Другими словами: условия непрерывного просмотра видео здесь нет.

В качестве триггера активации можете задать все видео или некоторые. Сохраните настройки триггера. Для того, чтобы передавать как можно большее количество информации в Google Analytics одним тегом, необходимо создать пользовательскую переменную типа **Таблица поиска**. Принцип работы переменной схож с конструкциями **if..else** и **switch-case**. Ее конфигурация будет выглядеть так:

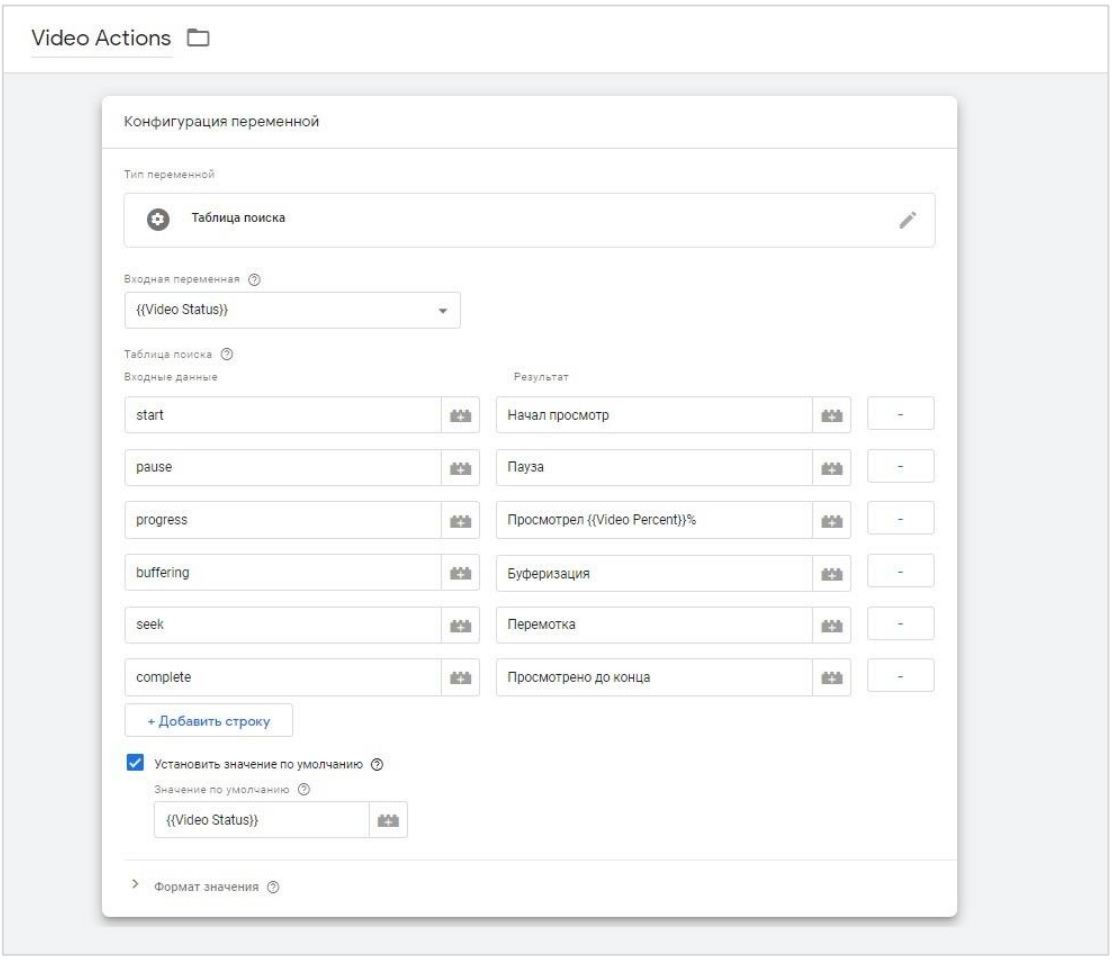


Рис. 518. Таблица поиска

Как работает таблица поиска?

1. Задается входная переменная. В этом примере - **{{Video Status}}**;
2. Далее идет проверка по таблице поиска и входным данным;
3. Если переменная **{{Video Status}}** принимает одно из значений из таблицы (start, pause, progress и т.д.), то входная переменная **{{Video Status}}** примет значение из поля **Результат**. В этих полях вы прописываете названия по своему усмотрению;
4. Если значение переменной **{{Video Status}}** не найдено среди таблицы поиска, то будет использовано значение по умолчанию (если указано в настройках). В этом примере - **{{Video Status}}.**

Примечание: напротив значения progress рекомендую прописать:

- название (например, **Просмотрел**);
- переменную процентов просмотра **{{Video Percent}}**;
- значок процента **%**.

Вместо таблицы поиска можно использовать переменную типа Собственный код JavaScript, которая выглядит так:

```
function() {
  var status = {{Video Status}};
  switch (status) {
    case 'start':
      return 'Начал просмотр';
    case 'pause':
      return 'Пауза';
    case 'buffering':
      return 'Буферизация';
    case 'progress':
      return 'Просмотрел ' + {{Video Percent}} + '%';
    case 'complete':
      return 'Просмотрено до конца';
  }
}
```

Сохраните настройки переменной. Она понадобится нам для следующего шага: настройки тега **Google Аналитика - Universal Analytics** с типом отслеживания **Событие**.

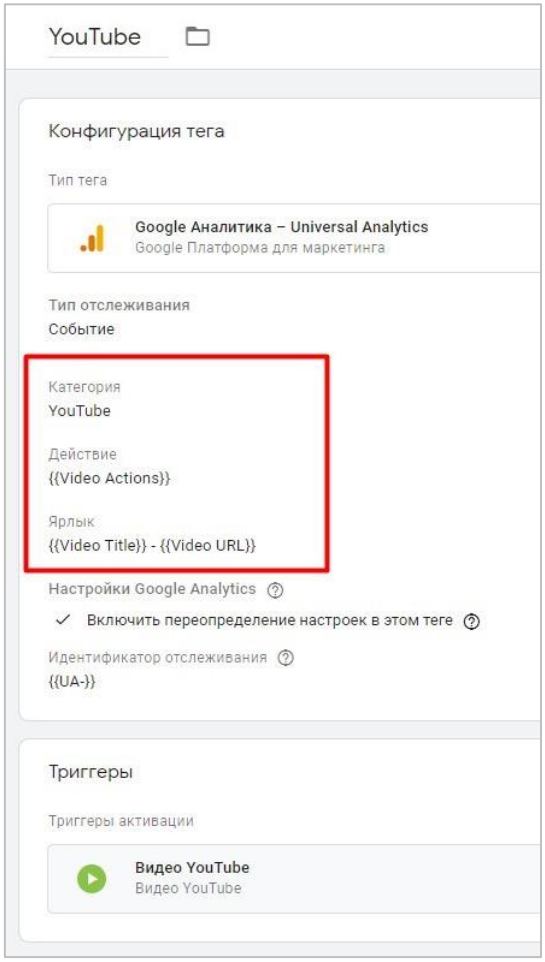


Рис. 519. Тег Google Аналитика - Universal Analytics

В качестве **Категории** можете задать произвольное значение. Я использую YouTube. В поле **Действие** добавьте созданную на предыдущем шаге таблицу поиска {{Video Actions}}. В **Ярлыке** события рекомендую передавать данные о названии видео **{{Video Title}}** и его URL **{{Video URL}}**, разделив их между собой тире "-". Триггер активации - **Видео YouTube**.

Перед публикацией тега следует проверить корректность его настройки с помощью режима отладки.

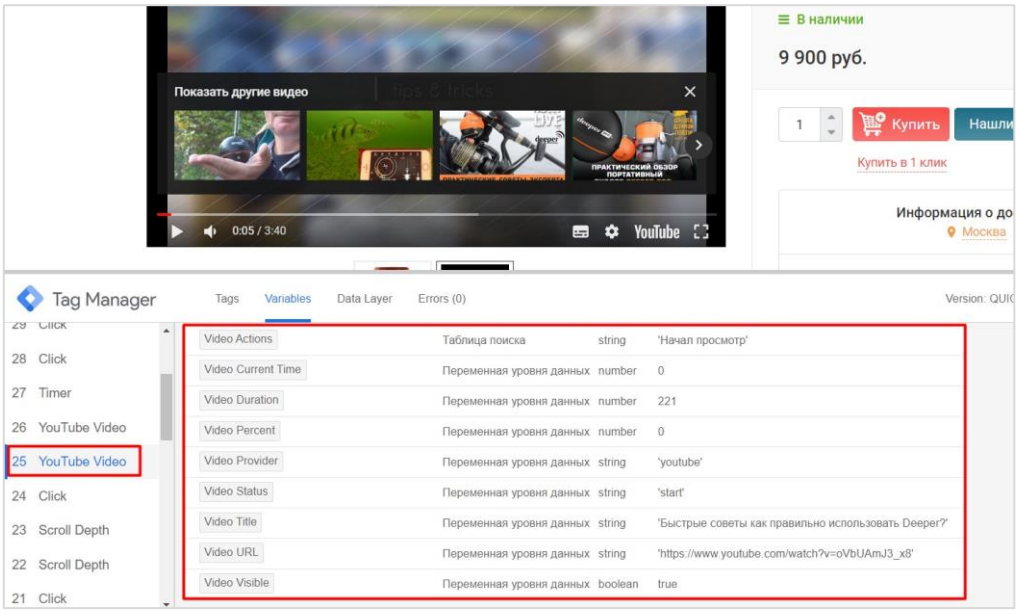


Рис. 520. Проверка в режиме предварительного просмотра

В режиме предварительного просмотра будет доступна информация по всем 8 переменным:

1. **Video Provider** — название платформы: youtube;
2. **Video Status** — статус видео: start;
3. **Video URL** — ссылка на видео YouTube: https://www.youtube.com/watch?v=oVbUAmJ3_x8;
4. **Video Title** — название видео: Быстрые советы как правильно использовать Deeper?;
5. **Video Duration** — продолжительность видео (в секундах): 221;
6. **Video Current Time** — текущее время видео (в секундах), в которое произошло событие: 0;
7. **Video Percent** — значение воспроизведенного видео (в процентах) на момент, когда сработало событие: 0;
8. **Video Visible** — значение видимости видео в окне браузера: true.

Переменная таблицы поиска **{{Video Actions}}** приняла удобочитаемое значение **Начал просмотр**, которое вы указали в выходных данных напротив стандартного **start**. На вкладке **Data Layer** можно увидеть, как переменные получают доступ к уровню данных и считывает ключи, которые задаются триггерами **Видео YouTube**.

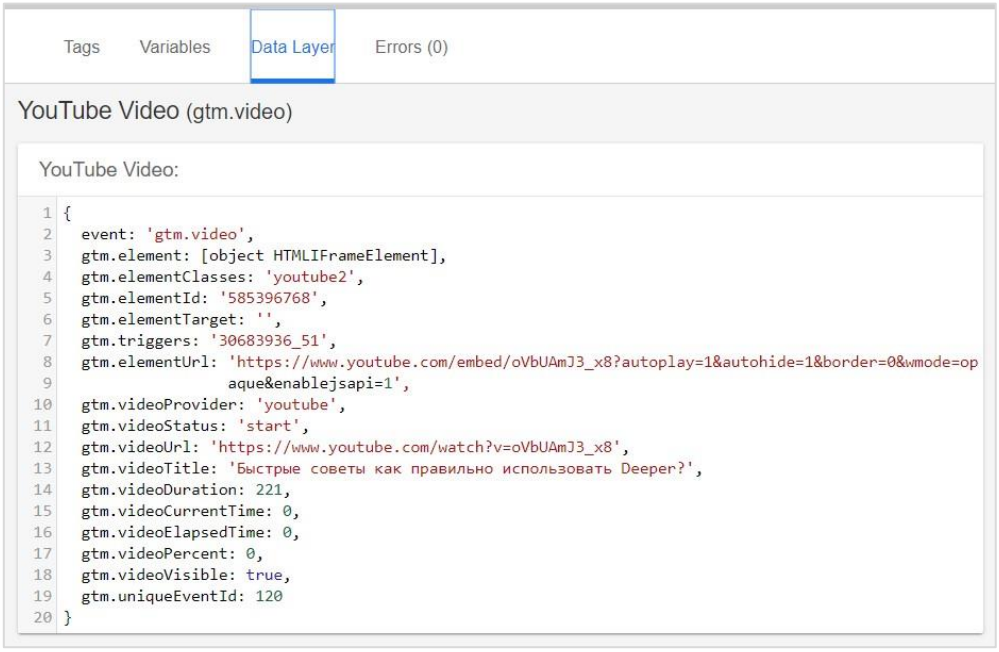


Рис. 521. Вкладка Data Layer

При правильной настройке в отчетах Google Analytics **В режиме реального времени** будут отображаться наши отслеживаемые события:

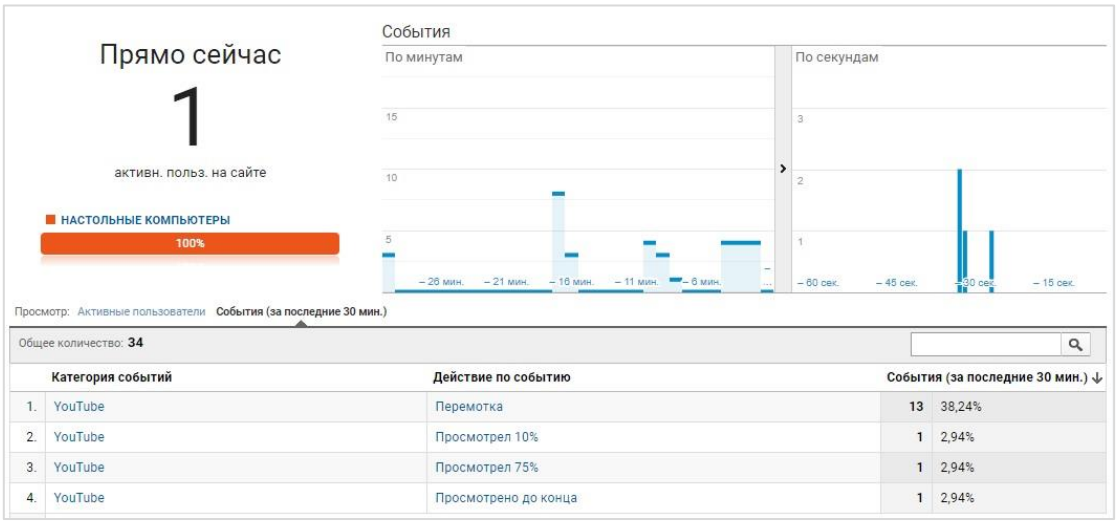


Рис. 522. Отчет В режиме реального времени

Посмотреть на все события можно в разделе **Поведение – События – Лучшие события**. Выбрав **Ярлык события** в качестве основного или дополнительного параметра, вы увидите название видео и URL-адрес.

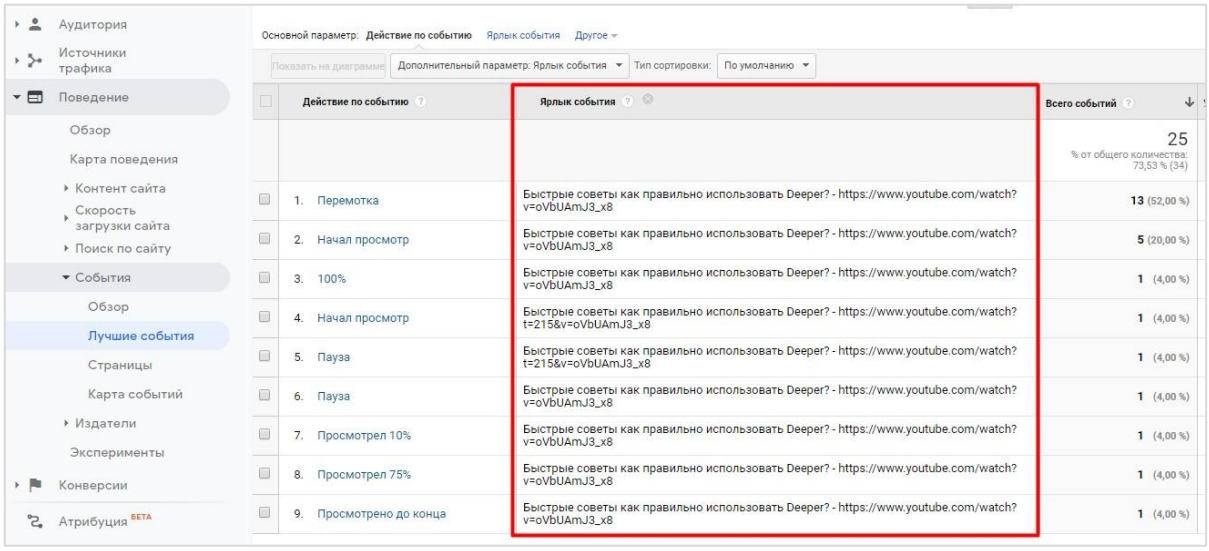


Рис. 523. Ярлык события в качестве дополнительного параметра с названием видео и URL

Иногда с помощью данного способа видео YouTube не отслеживаются и события не передаются. Это может быть связано с тем, что ваш сайт загружает контент динамически (например, во всплывающем окне) или у видео есть изображение, на которое нужно нажать перед тем, как видео начнет проигрываться.

Такое происходит не очень часто. Но если вы обнаружите, что видео не отслеживаются, попробуйте добавить следующую строчку кода в тег типа Пользовательский тег HTML. Триггер активации - **All Pages (Все страницы).**

```
<script src="https://www.youtube.com/iframe_api"></script>
```

В Google Tag Manager:

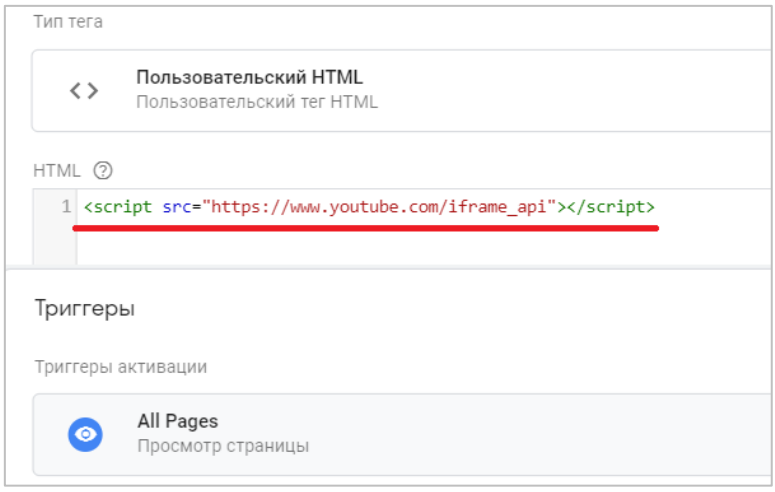


Рис. 524. Пользовательский тег HTML

Отслеживание загрузки файлов

В отличие от Яндекс.Метрики, в Google Analytics нет встроенного функционала отслеживания скачиваемых с вашего сайта файлов. Для этого необходимо размечать каждую ссылку соответствующим кодом события. А можно воспользоваться GTM и за 3 действия настроить отслеживание всех загружаемых файлов.

Начнем с Метрики. Чтобы определить, сколько раз посетители попытались (!) скачать с вашего сайта файлы (музыку, видео, документы и т.д.), нет необходимости в какой-либо настройки. Яндекс сам собирает статистику по скачиванию файлов, по умолчанию поддерживая большое количество расширений.

Для этого можно воспользоваться стандартным отчетом **Загрузки файлов**:

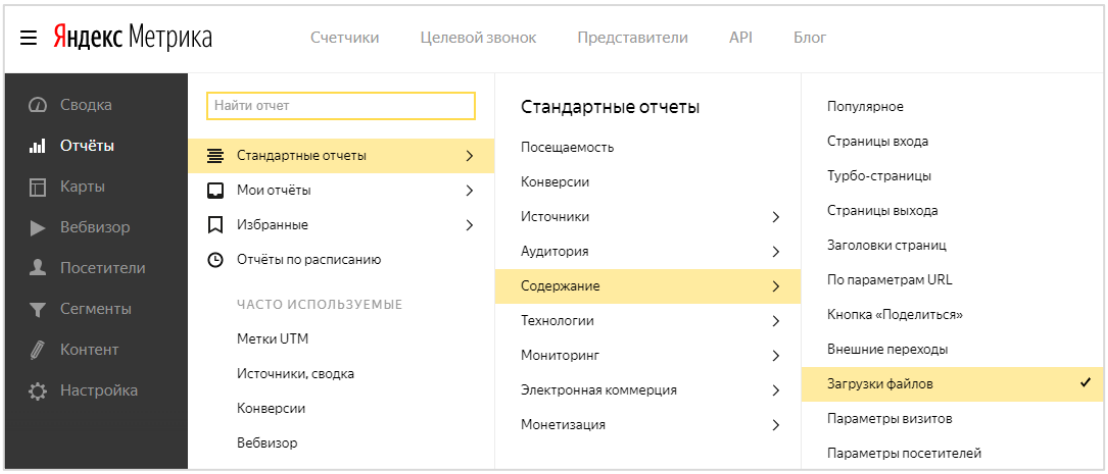


Рис. 525. Отчеты - Стандартные отчеты - Содержание - Загрузки файлов

В нем содержатся данные по взаимодействиям пользователей с файлами, которые размещены на вашем сайте. Это могут файлы разного размера и формата. Например:

- музыка (.mp3);
- документы (.docx, .rtf);
- отчетность, таблицы (.xlsx);
- видео (.mp4, .mpeg, .mov);
- установочные файлы (.exe);
- фотографии (.jpg, .jpeg, .png);
- книги (.djvu, .pdf);
- и т.д.

Если кто-то из посетителей захочет скачать что-то подобное на вашем сайте, Метрика зафиксирует это событие. По умолчанию поддерживается следующий список расширений: *3GP, 7Z, AAC, AC3, ACS, AI, AVI, APE, APK, ASF, BMP, BZ2, CAB, CDR, CRC32, CSS, CSV, CUE, DIVX, DMG, DJV, DJVU, DOC, DOCX, DOCM, DOCB, EMF, EPS, EXE, FLA, FLAC, FLV, GIF, GZ, TGZ, ISO, JPG, JPE, JPEG, JS, M3U, 3U8, M4A, M4V, MD5, MKV, MP3, MP4, MPG, MPEG, MOV, MSI, ODS, OGG, OGM, OGV, PDF, PHPS, PNG, PPT, PPTX, PPTM, PPTB, PSD, RAR, RSS, RTF, SEA, SIT, SFV, SHA1, SVG, SWF, TAR, TIF, TIFF, TORRENT, TS, TXT, VOB, WAV, WAVE, WEBM, WMA, WMV, WMF, XLS, XLSX, XLSM, XLSB,XPI,ZIP,GZIP*.

Вы можете передать в отчеты и другие расширения. Подробнее читайте в официальной справке Яндекса (см. приложение).

На рисунке ниже представлен пример отчета **Загрузки файлов** по сайту моего курса по веб-аналитике, на котором размещена программа обучения в формате **.pdf**:

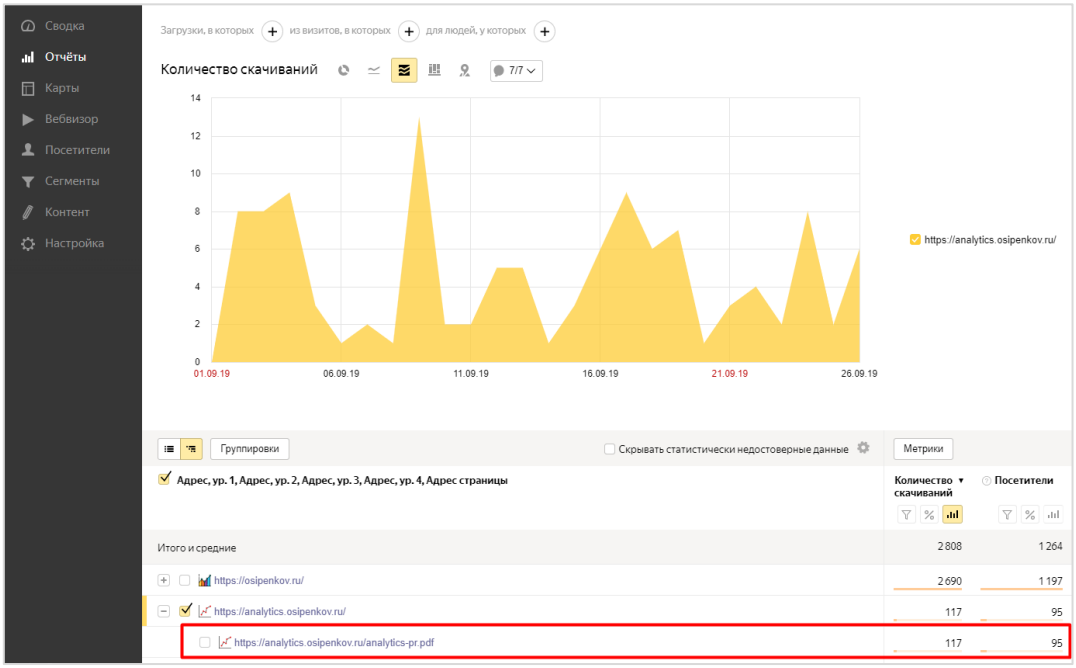


Рис. 526. Отчет Количество скачиваний

95 пользователей 117 раз переходило на ссылку с файлом **analytics-pr.pdf**. Данные в отчете сгруппированы по адресу, по которому расположен файл для скачивания. В сводку Метрики также можно добавить виджет со статистикой по скачиваниям, выбрав при этом нужную сегментацию (например, по странице загрузки).

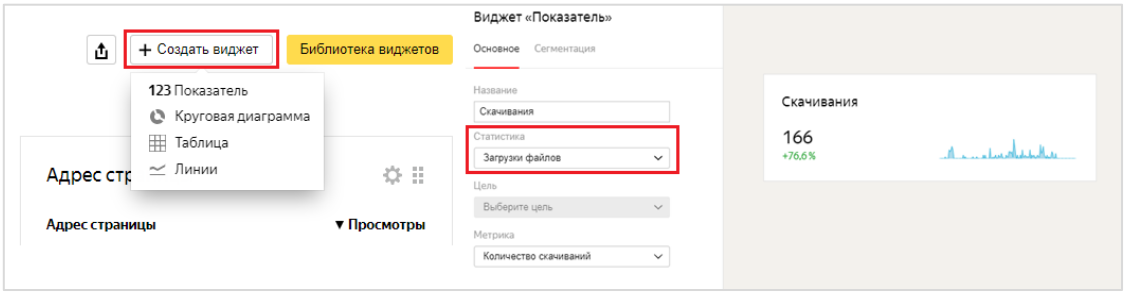


Рис. 527. Настройка виджета

Если Метрика умеет отслеживать загрузку файлов по умолчанию, то Google Analytics этим похвастаться не может. И чтобы зафиксировать взаимодействие пользователя с каким-либо файлом на вашем сайте, необходимо в код данного элемента добавить специальную конструкцию события.

Разберем тот же пример с analytics.osipenkov.ru. Нужно настроить отслеживание загрузки файла этого блока.

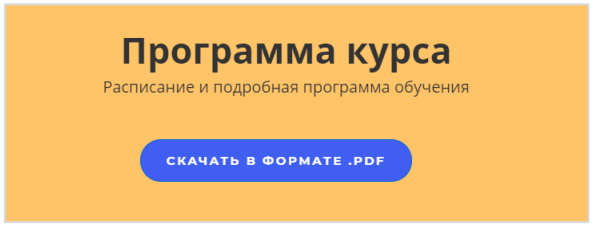


Рис. 528. Отслеживание скачивания файла

Чтобы настроить отслеживание загрузки файлов с помощью Google Tag Manager, необходимо:

- включить 1 встроенную переменную;
- настроить 1 триггер активации;
- создать 1 тег.

Переходим на вкладку **Переменные** и включаем встроенную переменную **Click URL**.

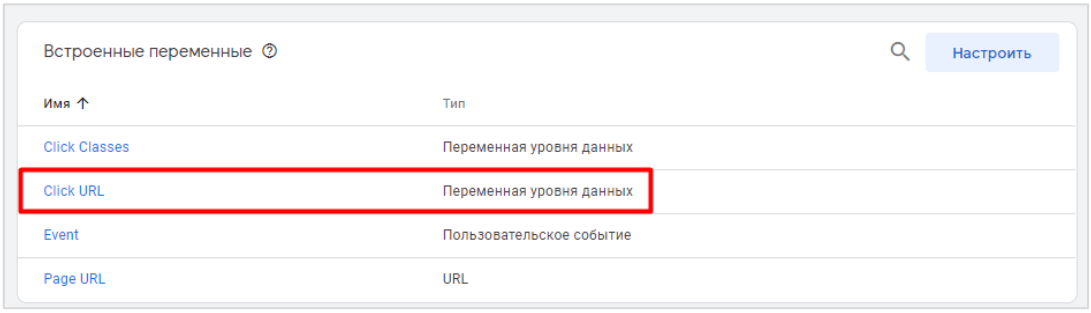


Рис. 529. Переменная Click URL

Переходим на вкладку **Триггер** и выбираем **Тип триггера – Клик – Только ссылки**. Добавляем такие настройки:

- Условие активации триггера – **Некоторые клики по ссылкам**
- Активировать триггер по Click URL – соответствует регулярному выражению

Регулярное выражение **\.(pdf|xlsx|xls|doc|txt|png|docx|ppt|pptx)$**

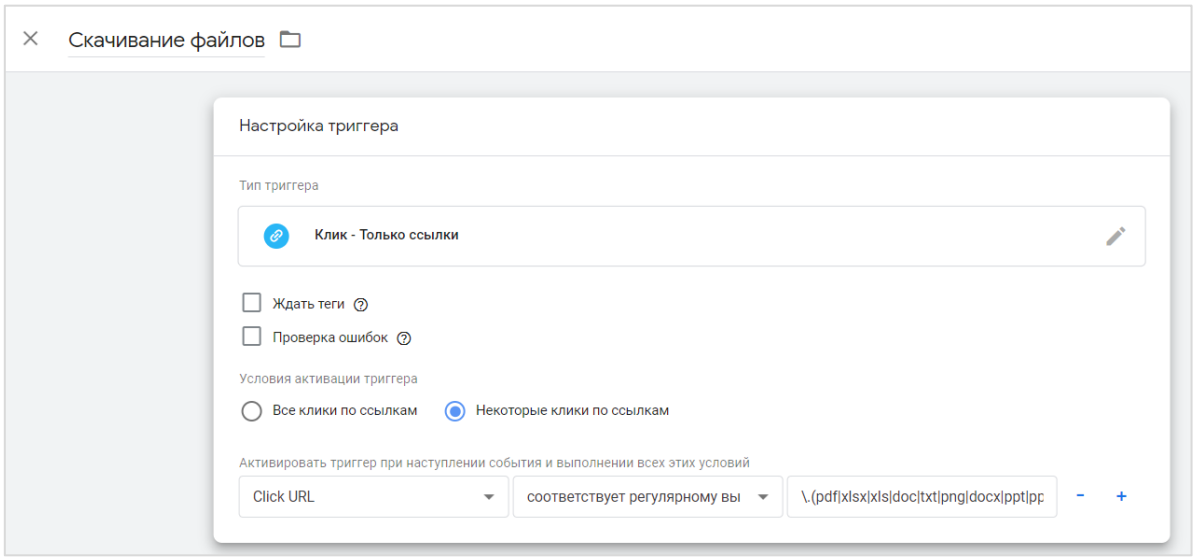


Рис. 530. Настройка триггера активации

В зависимости от типов файлов, которые размещены у вас на сайте, вы можете использовать разное количество расширений. Например, взять полный список всех расширений из Метрики и добавить его в триггер. В результате получите что-то подобное:

```
\.(3gp|7z|aac|ac3|acs|ai|avi|ape|apk|asf|bmp|bz2|cab|cdr|crc32|css|csv|cue|divx|dmg|djv|djvu|doc|docx|docm|docb|emf|eps|exe|fla|flac|flv|gif|gz|tgz|iso|jpg|jpe|jpeg|js|m3u|3u8|m4a|m4v|md5|mkv|mp3|mp4|mpg|mpeg|mov|msi|ods|ogg|ogm|ogv|pdf|phps|png|ppt|pptx|pptm|pptb|psd|rar|rss|rtf|sea|sit|sfv|sha1|svg|swf|tar|tif|tiff|torrent|ts|txt|vob|wav|wave|webm|wma|wmv|wmf|xls|xlsx|xlsm|xlsb,xpi,zip,gzip)$
```

Сохраняем триггер. Переходим на вкладку **Теги** и создаем тег типа **Google Аналитика – Universal Analytics**. Выбираем такие настройки:

- Тип отслеживания – **Событие**;
- Категория – произвольная (в моем примере **Файл**);
- Действие – произвольное (в моем примере **Скачивание**);
- Ярлык – **{{Click URL}}** (здесь будет отображаться путь к загружаемому файлу).

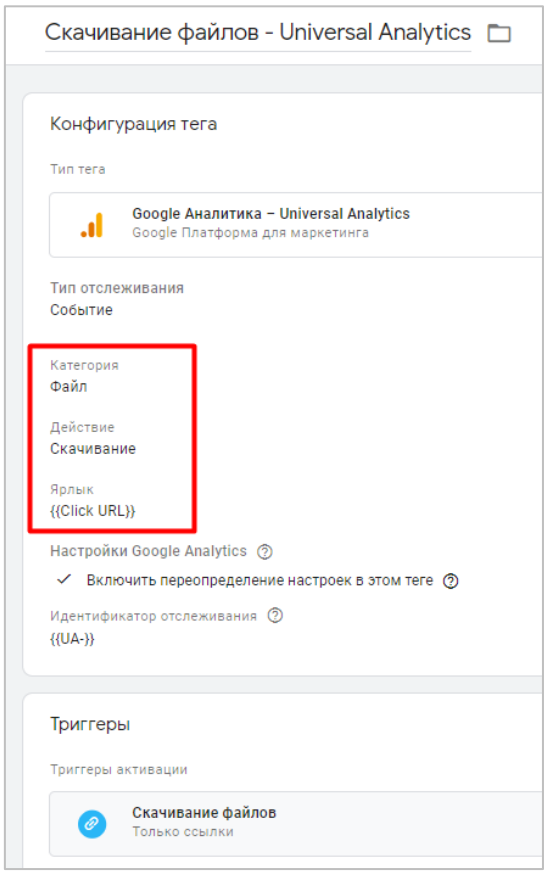


Рис. 531. Настройка тега

Это один из способов передачи данных в Analytics в качестве события. Вы можете вместо Категории и Действия использовать другие значения. Например, переменную **{{Page URL}}**, которая будет передавать URL страницы с файлом. И т.д.

Сохраняем тег. Публикуем обновленный контейнер. Чтобы проверить корректность отслеживания, воспользуемся режимом предварительного просмотра GTM. Кликнув по кнопке на сайте, сработал триггер активации, а с ним и тег.

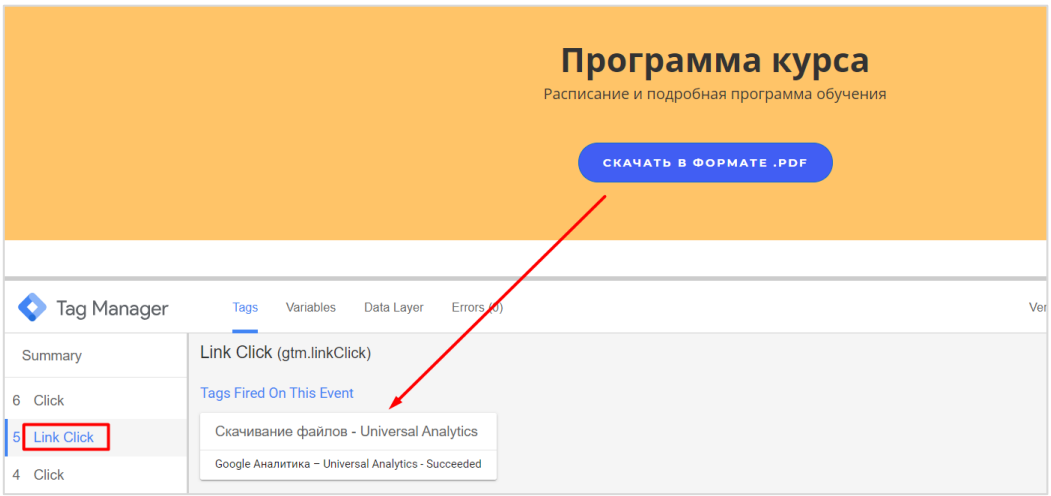


Рис. 532. Проверка отслеживания в режиме отладки

Информация также передалась в Google Analytics. В режиме реального времени видим наши Категорию, Действие и Ярлык события, в котором передался путь до загружаемого файла:

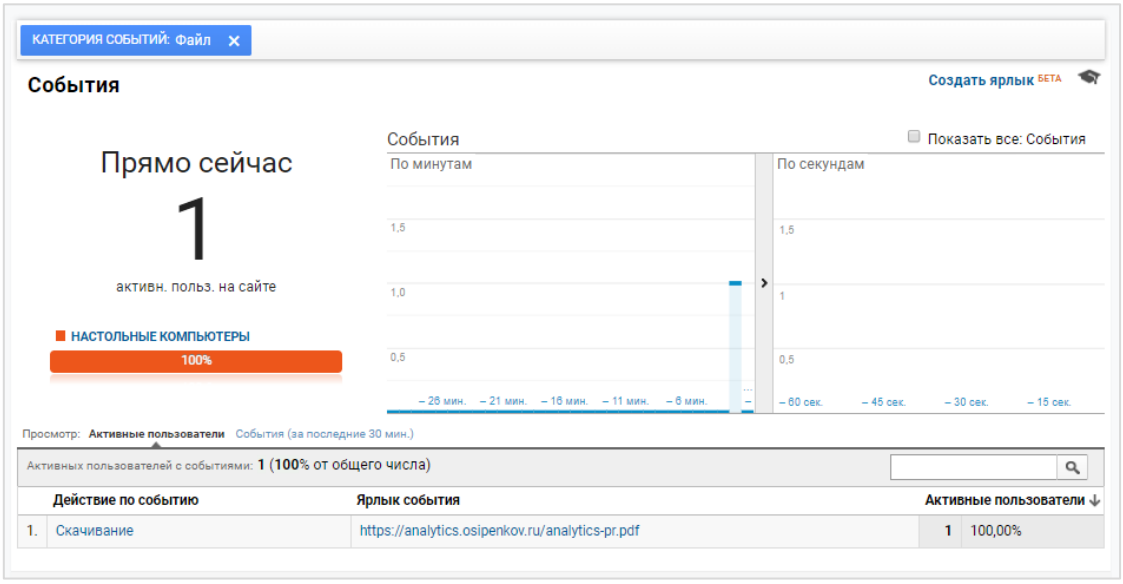


Рис. 533. Отчет В режиме реального времени

Данные по событиям будут также доступны в отчете **Поведение – События – Лучшие события**. Либо можно воспользоваться специальным отчетом с типом **Простая таблица**, добавив Категорию, Действие и Ярлык в качестве параметров.

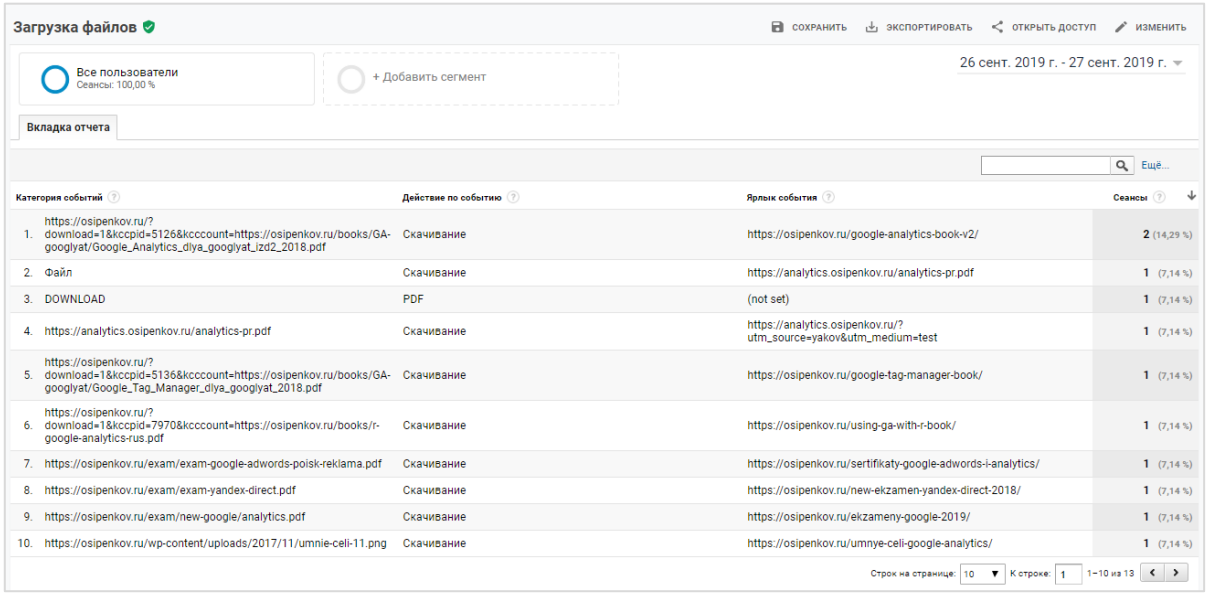


Рис. 534. Пример специального отчета в Google Analytics

Отслеживание 404 ошибок

Если на своем сайте вы часто видите 404 ошибка или Not Found (не найдено), то это важный звоночек к разбору данной проблемы. В этой статье разберем причины появления битых ссылок (URL) и способы их быстрого обнаружения с помощью Google Analytics.

Рис. 535. Пример сообщения 404 ошибки

Сообщение говорит нам о том, что сервер не может найти данные согласно запросу. Это может быть связано с тем, что пользователем целенаправленно был введен ошибочный адрес страницы, или он использует некорректную ссылку, а также, если ранее эта страница была доступна, а теперь нет. Например, пользователь удалил свою учетную запись.

Так или иначе это может пагубно сказаться на ранжировании нашего сайта в поисковых системах и при ведении рекламных кампаний (Google Ads, Яндекс.Директ, Facebook и т.д.), поскольку запоздалое реагирование на битую ссылку может привести к потере n-ого количества денег.

Ошибка 404 может появится как в случае каких-то внутренних изменений (работы разработчиков), так и внешних (вы или посетитель допустили опечатку в ссылке, неправильно разметили рекламные кампании и т.д.). Не забывайте, что страницы, которые были доступны на запуске и которые Google и Яндекс пропустили (посчитали активными), в какой-то момент могут таковыми перестать быть. А вы будете продолжать откручивать рекламный бюджет на эти страницы. Это не есть хорошо.

Такие ошибки могут привести и к ухудшению имиджа вашей компании, а также к негативной реакции пользователей на ваш сайт. Представьте себе, что человек ищет конкретную информацию, и на вашем сайте она есть. Но когда пользователь переходит по ссылке – ему выдает 404 ошибку. Потенциальный клиент недоволен, злой уходит с сайта и совершает покупку у вашего конкурента.

Поэтому так важно отслеживать переходы на такие страницы и реагировать на факт перехода на страницу 404. Еще раз резюмируем перечень возможных причин ошибки Not Found:

- URL введен неправильно;
- битые ссылки;
- адрес страницы был изменен. В этом случае поможет 301 редирект, который указывает роботам ПС, что страница перемещена по новому адресу, а старый адрес следует считать устаревшим. Ссылочный вес старого адреса будет передан новому URL;
- страница удалена или прекращено существование всего сайта.

Задача сводится к двум составляющим:

1. отследить 404 ошибки (разбираем в этой статье);
2. исправить эти неработающие ссылки, перенаправив людей на новый целевой URL.

404 ошибки можно отслеживать не только с помощью Analytics. Для этого даже лучше подходит инструмент Google Search Console, который обнаруживает неисправности сканерами Google.

В связи с этим существуют некоторые ограничения: эти ошибки регистрируются роботом Googlebot, то есть не обязательно просматриваются пользователями. Поэтому вы не можете видеть, как это влияет на общие сеансы пользователей, и вы не можете включать их в свои отчеты аналитики.

Чтобы настроить отслеживание 404 ошибок с помощью Google Tag Manager, нам необходимо выполнить 4 пункта:

1. определить название (title) страницы;
2. создать переменную JavaScript;
3. создать триггер;
4. создать тег;

1. Определение названия страницы с 404 ошибкой

Самый простой способ – вызвать искусственно 404 ошибку на вашем сайте, добавив в url любую абракадабру в конце.

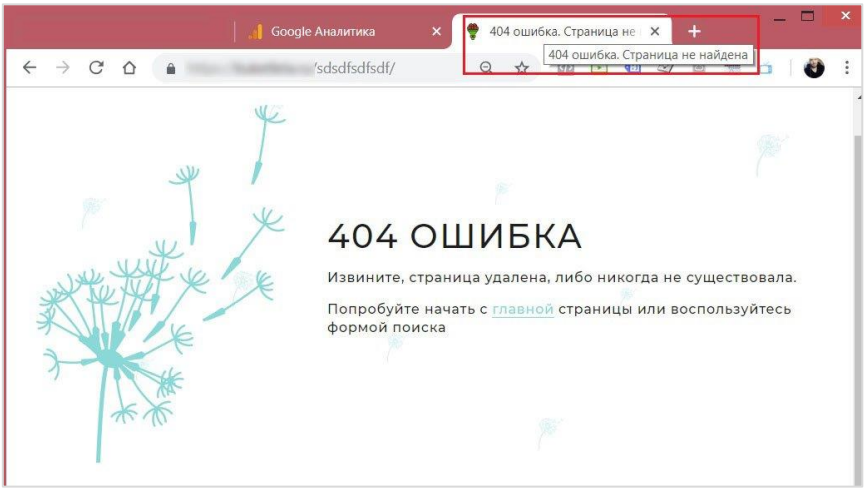


Рис. 536. Пример страницы с сообщением о 404 ошибке

Title страницы 404 в моем примере: **404 ошибка. Страница не найдена**. У вас может быть другое название или не быть его вообще. В таком случае обратитесь к разработчику, чтобы он в шаблоне 404 ошибки сделал статичный вывод данного заголовка.

2. Создание переменной JavaScript

Для записи и возврата значения по заголовку той страницы, которую пользователь посетил, можно воспользоваться пользовательской переменной GTM.

В Google Tag Manager заходим в Переменные, в пользовательских переменных нажмите **СОЗДАТЬ** и выбираем **Переменная JavaScript**.

- **Название** - document.title
- **Тип** - Переменная Javascript
- **Имя глобальной переменной** - document.title

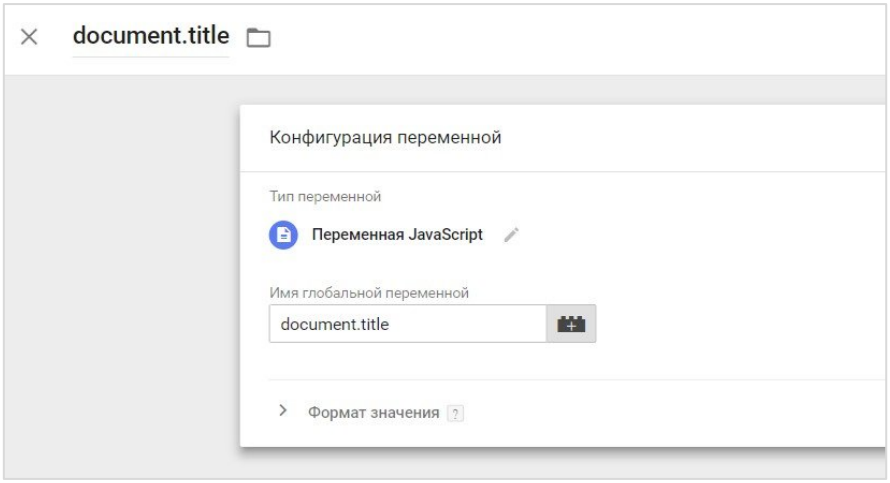


Рис. 537. Переменная JavaScript - document.title

Сохраняем переменную. Свойство **document.title** получает или задает текущий заголовок документа. Вы можете протестировать эту конструкцию, перейдя на любой сайт, любую вкладку и открыв консоль разработчика. Введите **document.title** и нажмите Enter:

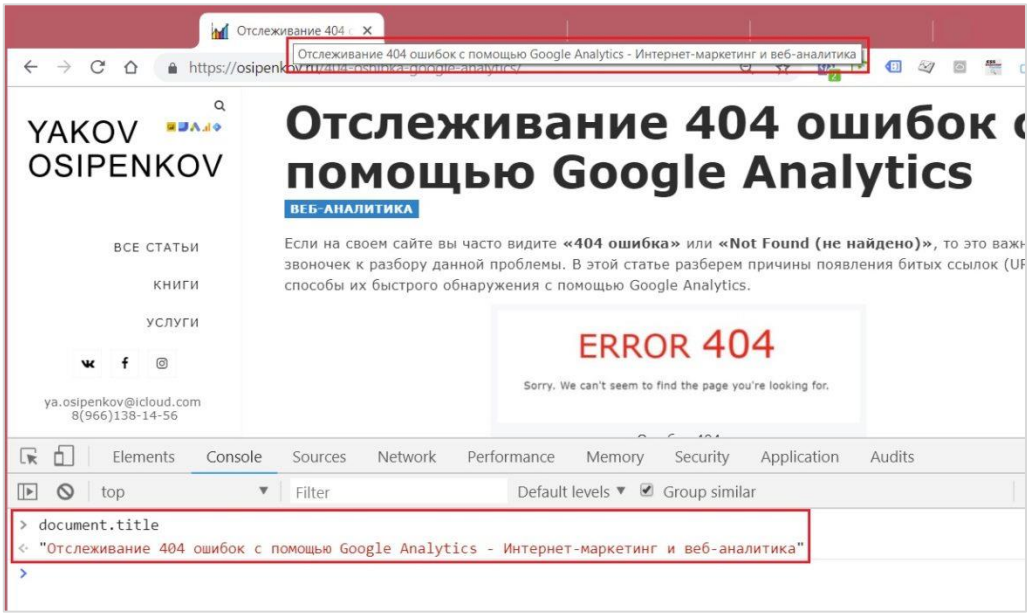


Рис. 538. Вызов document.title в консоли разработчика

3. Создание триггера

Далее необходимо создать триггер, который бы активировался не на всех страницах, а когда заголовок страницы содержит **404 ошибка. Страница не найдена**.

В GTM заходим в Триггеры, нажмите **СОЗДАТЬ** и выбираем **Просмотр страницы**.

- **Название** – Триггер 404 ошибка
- **Тип триггера** – Просмотр страницы
- **Условия активации триггера** - Некоторые просмотры страниц

Выбираем нашу переменную **document.title**, условие **содержит** и добавляем часть или полностью **Страница не найдена**.

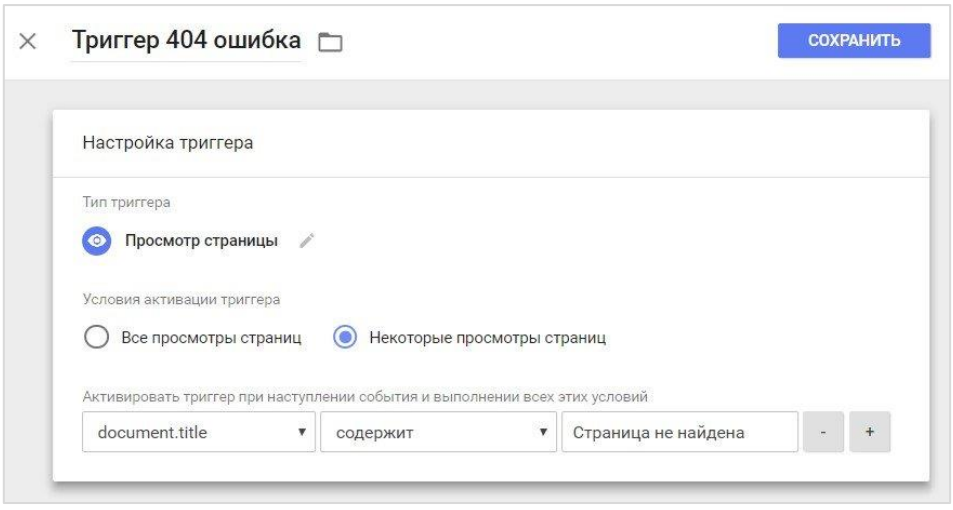


Рис. 539. Триггер активации 404 ошибки

Сохраняем триггер.

4. Создание тега

На последнем шаге настройки нам следует создать тег, который будет активироваться по триггеру **Триггер 404 ошибка** и передавать данные в Google Analytics как события, с категорией, действием и ярлыком.

В Google Tag Manager переходим в раздел Теги, нажмите **СОЗДАТЬ** и выбираем в качестве конфигурации тип тега **Google Аналитика – Universal Analytics**.

Настройки:

- **Тип отслеживания** – Событие
- **Категория** – 404 ошибка (произвольно)
- **Действие** - {{Page URL}} или {{Page Path}} (если хотите весь путь, то первый вариант, если часть URL, то Page Path)
- **Ярлык** – {{Referrer}} (переменная, в которой будет значение страницы, с которой перешел пользователь, то есть предыдущая)
- **Не взаимодействие** – True (чтобы действие не влияло на показатель отказов)

В случае, если у вас до этого была создана переменная с типом **Константа** для идентификатора Google Analytics (чтобы не вводить каждый раз UA- счетчика), включаем переопределение настроек и выбираем нашу переменную. У меня это {{UA-}}. В качестве триггера активации выбираем Триггер 404 ошибка.

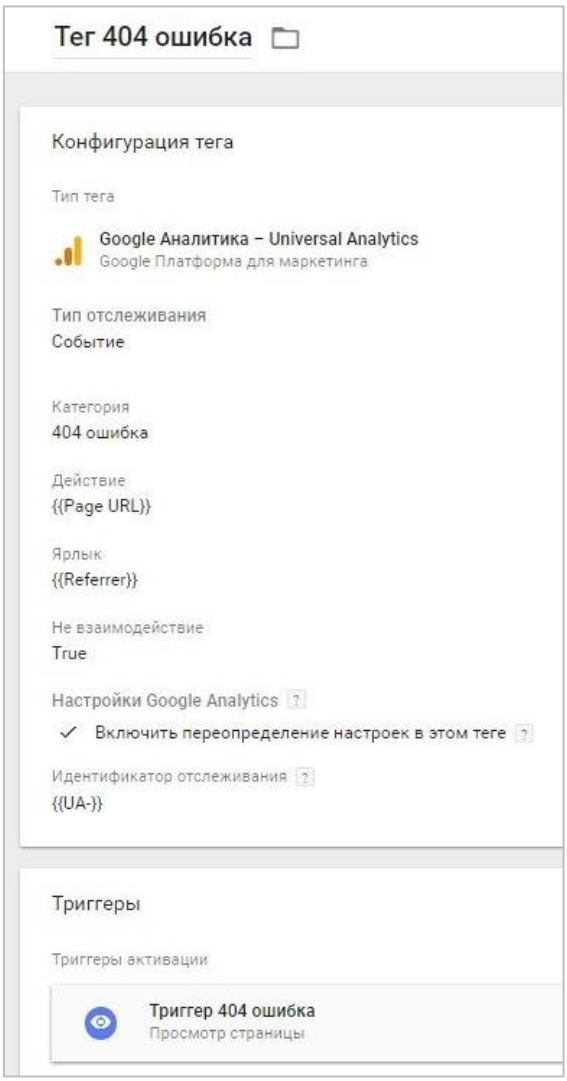


Рис. 540. Настройка тега

Сохраняем тег. На этом настройка отслеживания 404 ошибок с помощью Google Tag Manager завершена.

Хорошим тоном будет проверка корректности передачи данных с помощью режима отладки. Перейдем на страницу с 404 ошибкой. Если все настроено верно, то **Тег 404 ошибка** будет активирован и находиться в **Tags Fired On This Page**.

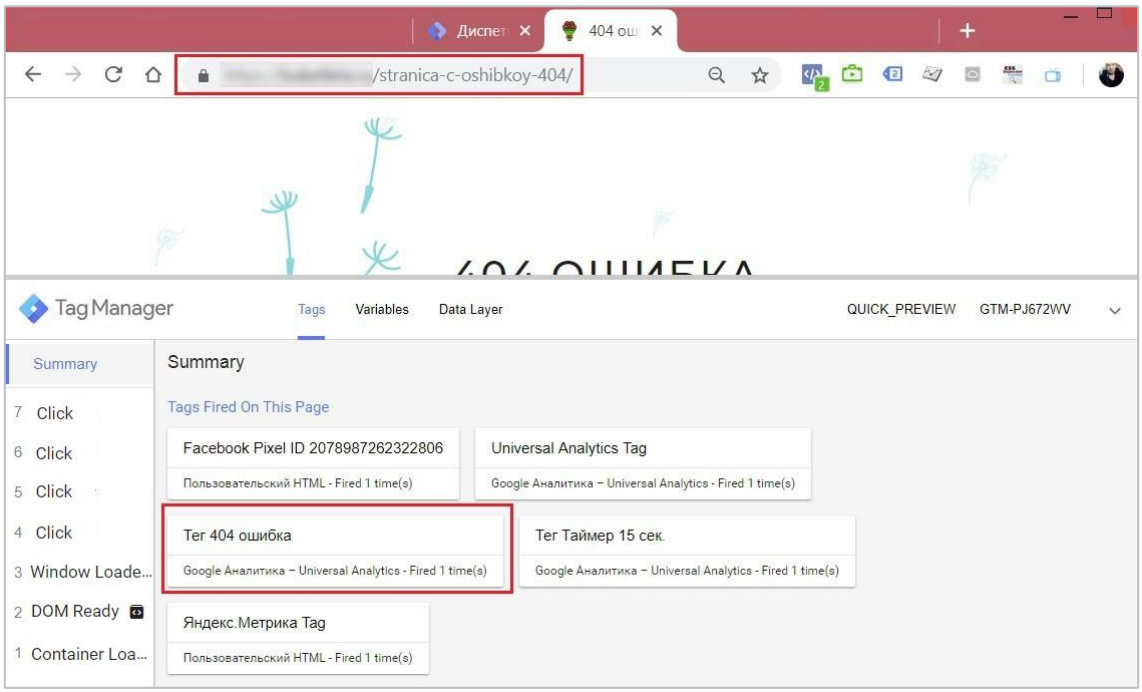


Рис. 541. Проверка тега в режиме предварительного просмотра

Раскроем его и увидим данные, которые в нем возвращаются - наши Категория, Действие, Ярлык.

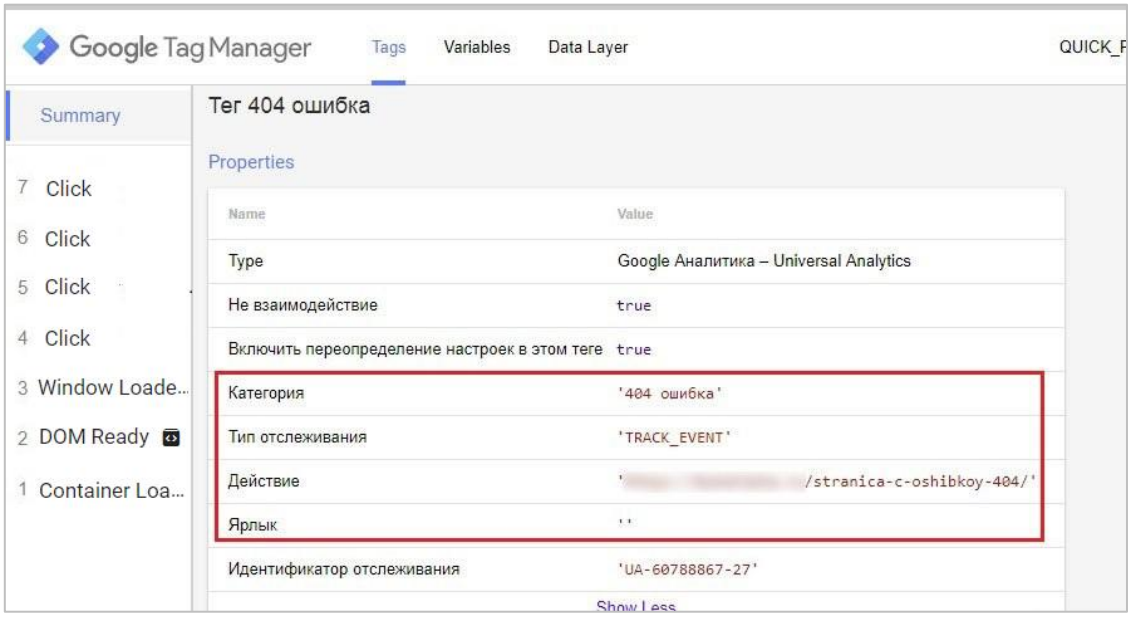


Рис. 542. Данные, которые передаются в теге

Перейдем в Google Analytics и в отчетах **В режиме реального времени** на вкладке **События** увидим наше событие.

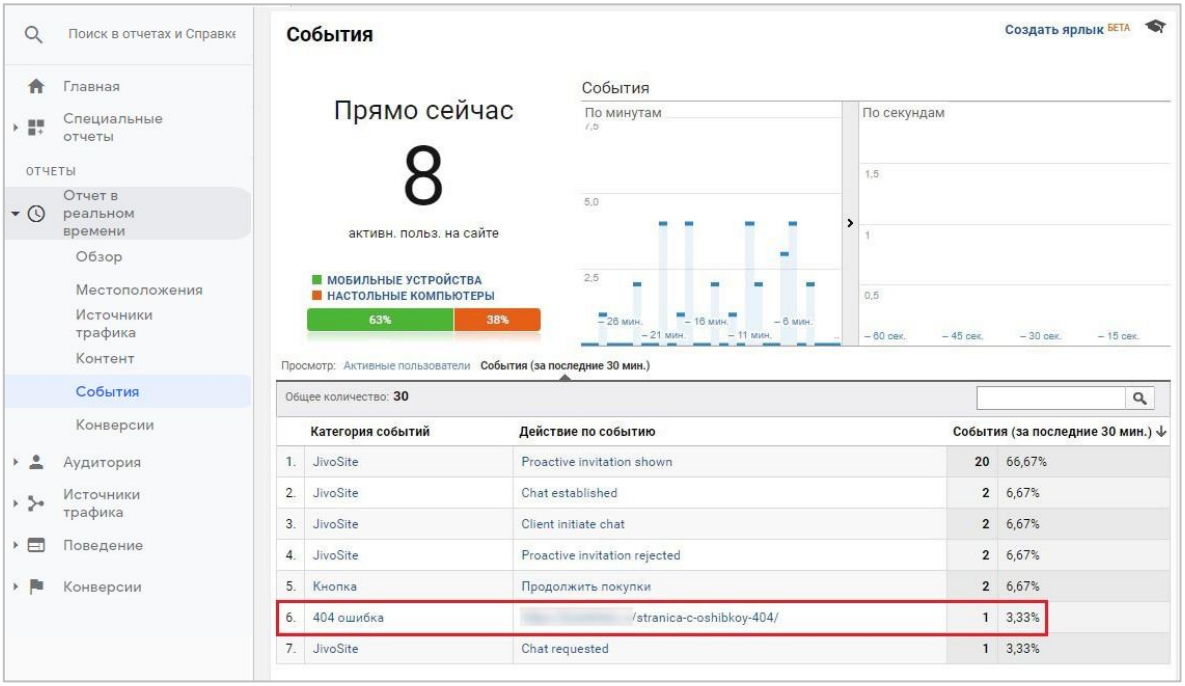


Рис. 543. Отчет В режиме реального времени

Данные по событиям также можно посмотреть в отчетах **Поведение – События – Лучшие события**, или же построив собственный отчет.

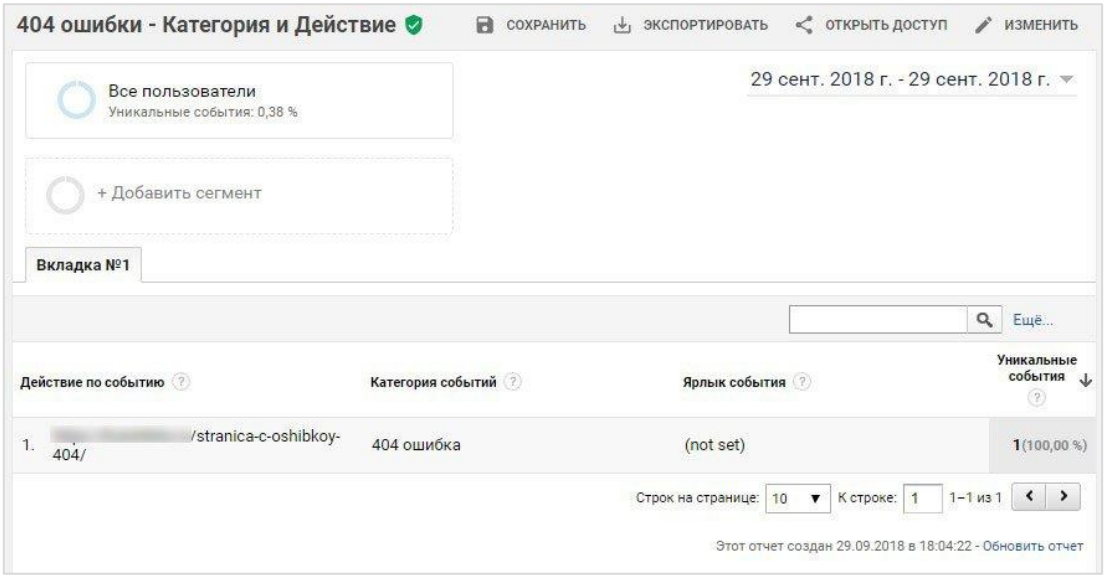


Рис. 544. Пример специального отчета в Google Analytics

Отслеживание уникального идентификатора пользователя (Client ID, cid)

Как вы уже знаете из предыдущей главы про файлы cookie, идентификация посетителей сайта и связывания его действий в единый профиль происходит в счетчиках веб-аналитики с помощью уникального идентификатора пользователя (Client ID, cid). В Google Analytics доступна история обращений для каждого Client ID в отчете **Статистика по пользователям** (раздел **Аудитория**).

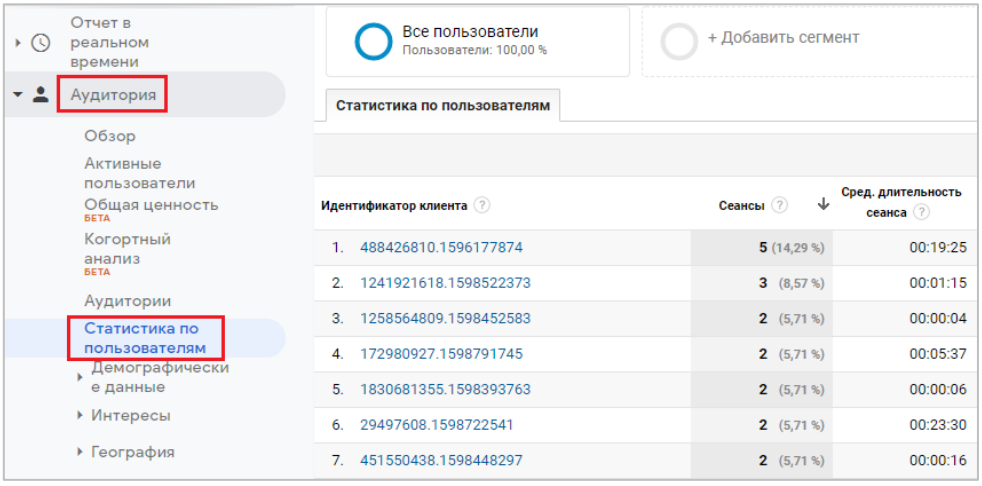


Рис. 545. Отчет Статистика по пользователям

Историю пользователя, его переходы по страницам на сайте можно посмотреть, провалившись внутрь идентификатора клиента:

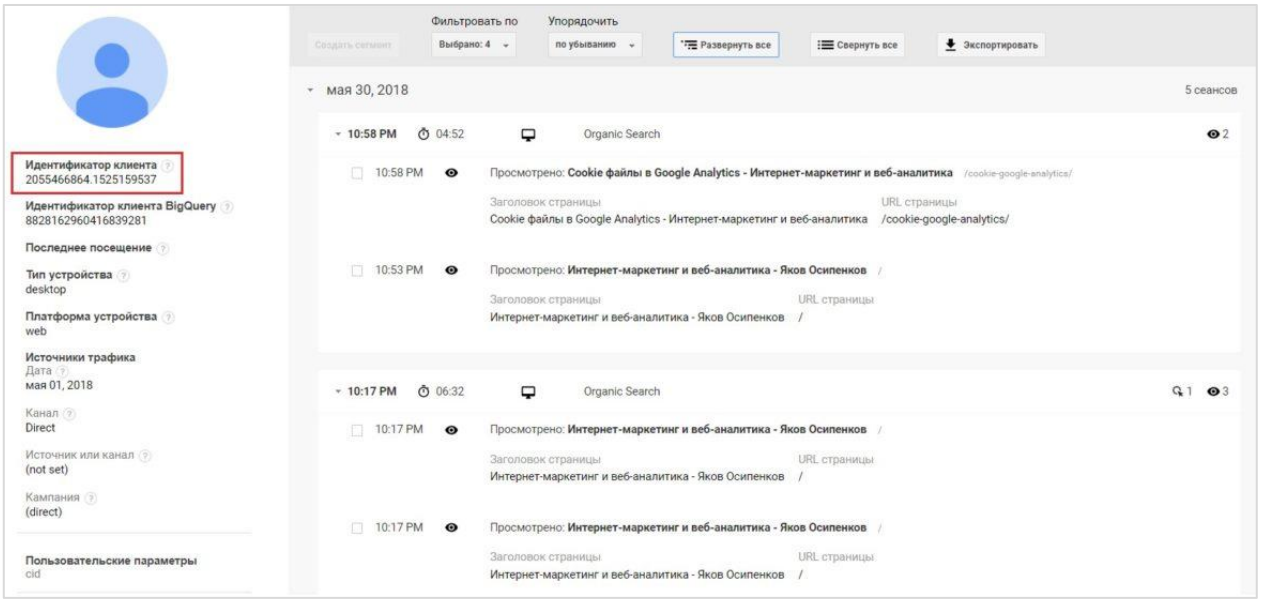


Рис. 546. Карточка одного из пользователей сайта

Отчет **Статистика пользователей** ограничен своей функциональностью. Например, идентификатор пользователя нельзя сгруппировать с другими параметрами, оперативно посмотреть в разрезе устройств, источников или каналов, местоположению, страницам перехода и другим срезам. Поэтому Client ID в Google Analytics, как правило, передают в качестве специального параметра.

С помощью Client ID можно интегрировать Google Analytics и CRM, сводить данные, отслеживать обращения в offline и строить различные пути взаимодействия пользователя с сайтом. Давайте разберем 3 способа передачи уникального идентификатора пользователя в Google Analytics через Google Tag Manager:

1. с помощью функции **customTask**;
2. с помощью метода **get()** Google Analytics;
3. из cookie **_ga** Google Analytics;

+ бонус: если вы добавляли тег Google Analytics через пользовательский HTML (gtag.js).

Способ №1 – customTask (см. приложение)

Задания – расширенная функция, которая позволяет настраивать проверку, создание и отправку запросов Measurement Protocol с помощью *analytics.js*. По умолчанию сама ничего не делает. Получить доступ к заданиям или заменить их можно с помощью стандартных методов счетчика **get** и **set**.

Последовательность действий:

- создаем пользовательский параметр в Google Analytics с областью действий **Пользователь**

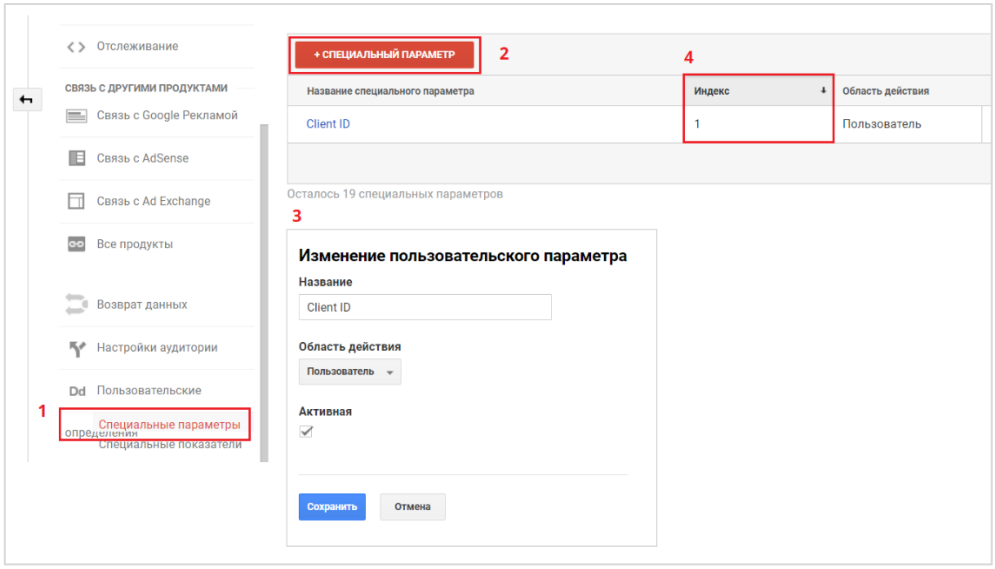


Рис. 547. Создание пользовательского параметра

Запоминаем **Индекс** параметра. В этом примере **1**.

- переходим в Google Tag Manager и создаем пользовательскую переменную типа **Собственный код JavaScript**. Вставляем следующий код и заменяем цифру 5 в переменной **customDimensionIndex** на собственный индекс (см. выше). Название переменной GTM задаем произвольное:

```
function() {
  var customDimensionIndex = 5;
  return function(model) {
    model.set('dimension' + customDimensionIndex, model.get('clientId'));
  }
}
```

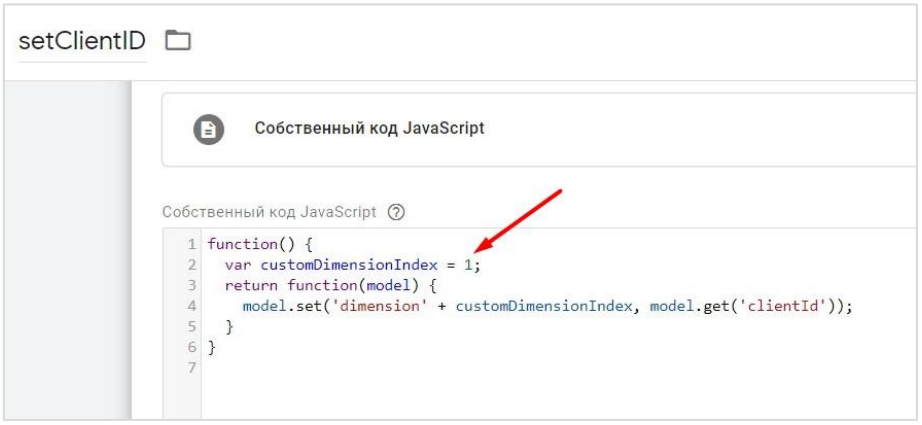


Рис. 548. Замените индекс в переменной customDimensionIndex на свой

- в настройках существующего тега Universal Analytics в дополнительных настройках добавьте поле с именем **customTask** (пропишите вручную) и значением вашей переменной, созданной на предыдущем шаге. Должно получиться так:

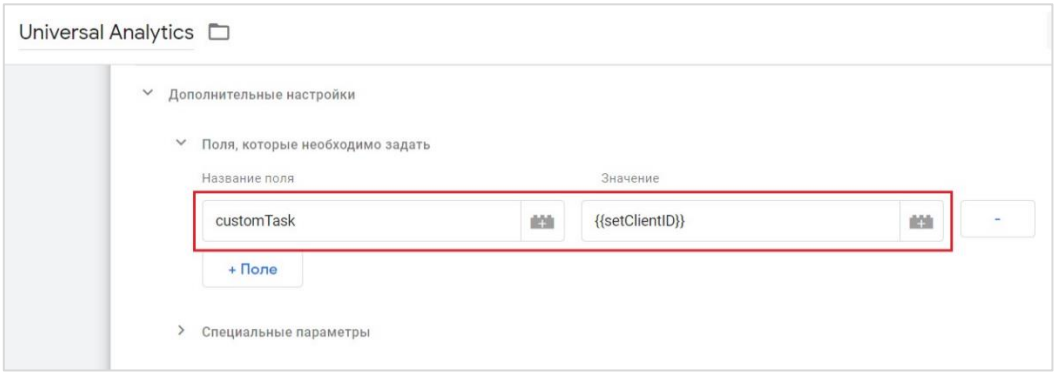


Рис. 549. Поля, которые необходимо задать – customTask

Опубликуйте контейнер с изменениями. Проверить корректность передачи Client ID можно с помощью консоли разработчика или специальные плагины / расширения для GTM и Google Analytics.

Проверка через консоль разработчика:

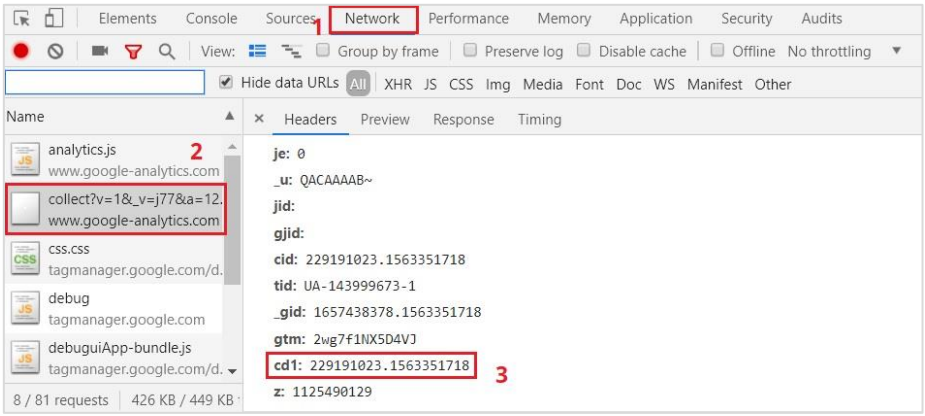
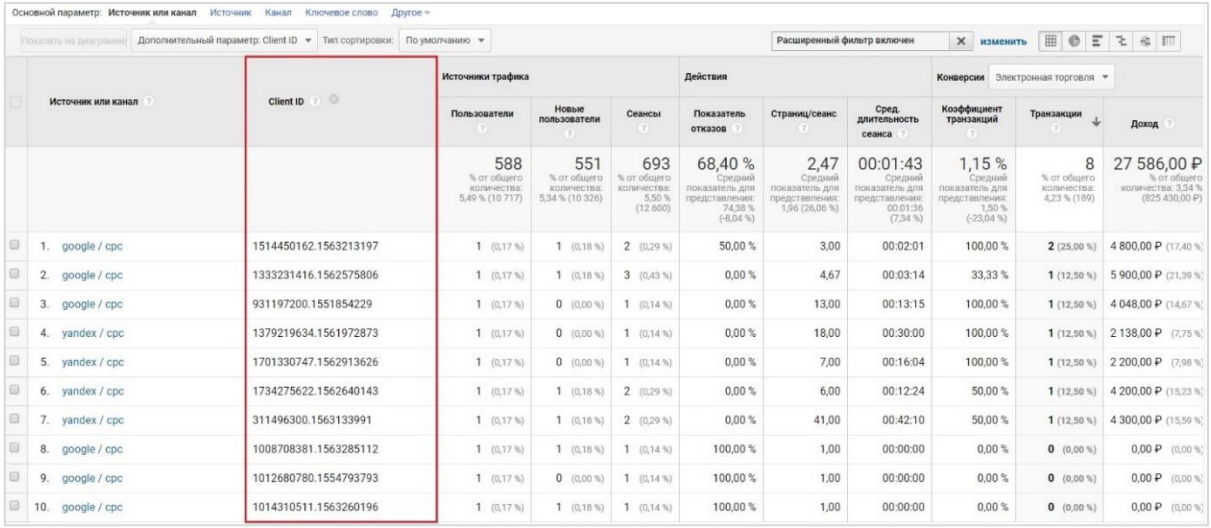


Рис. 550. Проверка через консоль разработчика – collect

Если все хорошо, то через некоторое время данные по Client ID начнут поступать в отчеты Google Analytics. Параметр Client ID в отчетах можно использовать в качестве основного, дополнительного, и в собственных отчетах.



	Источник или канал	Client ID	Пользователи	Новые пользователи	Сеансы	Показатель отказов	Страниц/сеанс	Сред. длительность сеанса	Коэффициент транзакций	Транзакции	Доход
			588	551	693	68,40 %	2,47	00:01:43	1,15 %	8	27 586,00 ₽
1.	google / cpc	1514450162.1563213197	1	1	2	50,00 %	3,00	00:02:01	100,00 %	2	4 800,00 ₽
2.	google / cpc	1333231416.1562575806	1	1	3	0,00 %	4,67	00:03:14	33,33 %	1	5 900,00 ₽
3.	google / cpc	931197200.1551854229	1	0	1	0,00 %	13,00	00:13:15	100,00 %	1	4 048,00 ₽
4.	yandex / cpc	1379219634.1561972873	1	0	1	0,00 %	18,00	00:30:00	100,00 %	1	2 138,00 ₽
5.	yandex / cpc	1701330747.1562913626	1	0	1	0,00 %	7,00	00:16:04	100,00 %	1	2 200,00 ₽
6.	yandex / cpc	1734275622.1562640143	1	1	2	0,00 %	6,00	00:12:24	50,00 %	1	4 200,00 ₽
7.	yandex / cpc	311496300.1563133991	1	1	2	0,00 %	41,00	00:42:10	50,00 %	1	4 300,00 ₽
8.	google / cpc	1008708381.1563285112	1	1	1	100,00 %	1,00	00:00:00	0,00 %	0	0,00 ₽
9.	google / cpc	1012680780.1554793793	1	0	1	100,00 %	1,00	00:00:00	0,00 %	0	0,00 ₽
10.	google / cpc	1014310511.1563260196	1	1	1	100,00 %	1,00	00:00:00	0,00 %	0	0,00 ₽

Рис. 551. Отчет Google Analytics с дополнительным параметром Client ID

Этот способ настройки является наиболее распространенным, поскольку, по наблюдениям специалистов, имеет наибольший процент прометки трафика, а также наименьшую нагрузку. Client ID отправляется с начальным просмотром страницы и не создает дополнительных хитов. Я рекомендую использовать именно этот вариант передачи Client ID.

Способ №2 - с помощью метода get()

Этот способ получения идентификатора клиента рекомендует Google в справке разработчика (см. приложение). Что нужно сделать? Также создать пользовательский параметр Google Analytics (как и в способе 1), в Google Tag Manager создать пользовательскую переменную типа **Собственный код JavaScript** и вставить туда следующий код (см. приложение):

```
function() {
  try {
    var tracker = ga.getAll()[0];
    return tracker.get('clientId');
  } catch(e) {
    console.log("Ошибка получения Client ID");
  }
}
```

В GTM это выглядит так:

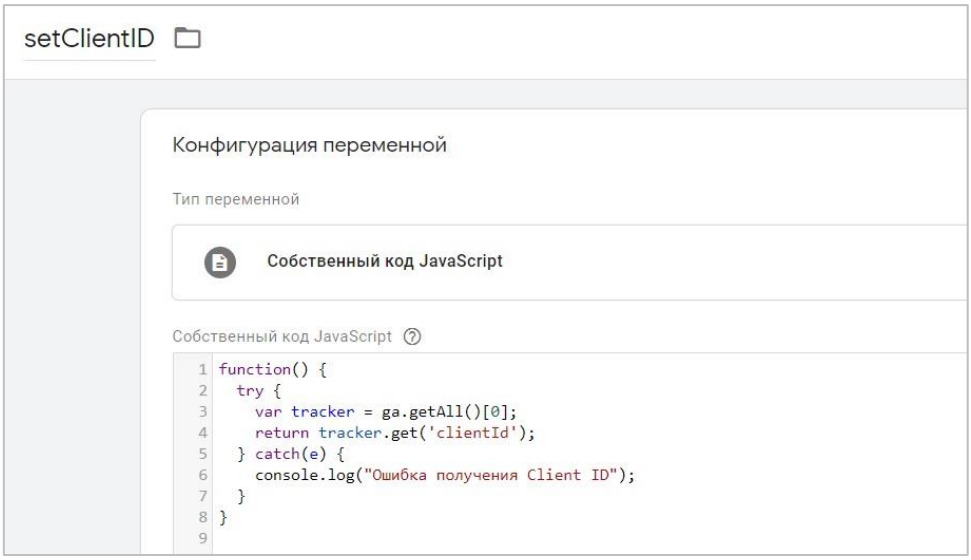


Рис. 552. Получение Client ID с помощью метода get()

С помощью него вы получите трекер Google Analytics, а затем через него методом **get()** получите из трекера нужно значение Client ID. **get()** - доступный метод объекта **Tracker**, который получает значение поля, хранящегося в счетчике. Это и есть **clientId**.

Результат в режиме отладки:

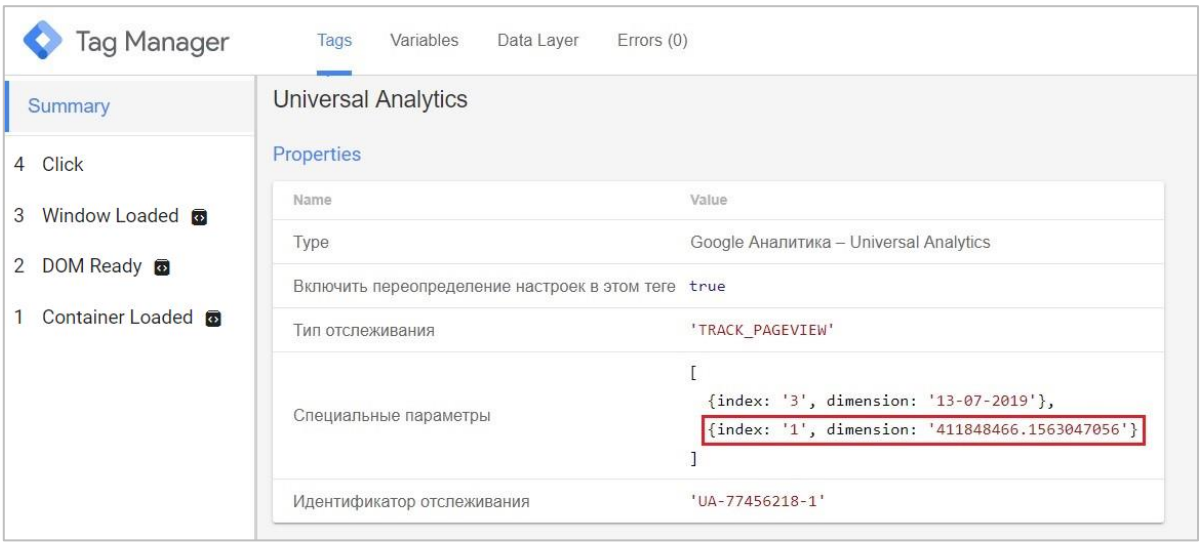


Рис. 553. Данные в режиме отладки

Client ID отправляется с начальным просмотром страницы в момент загрузки контейнера. Однако из такого подхода есть вероятность получить некоторую часть трафика непомеченной. В этом случае рекомендуется отправлять Client ID отдельным тегом Universal Analytics с типом отслеживания **Событие** и триггером активации **Окно загружено (gtm.load).**

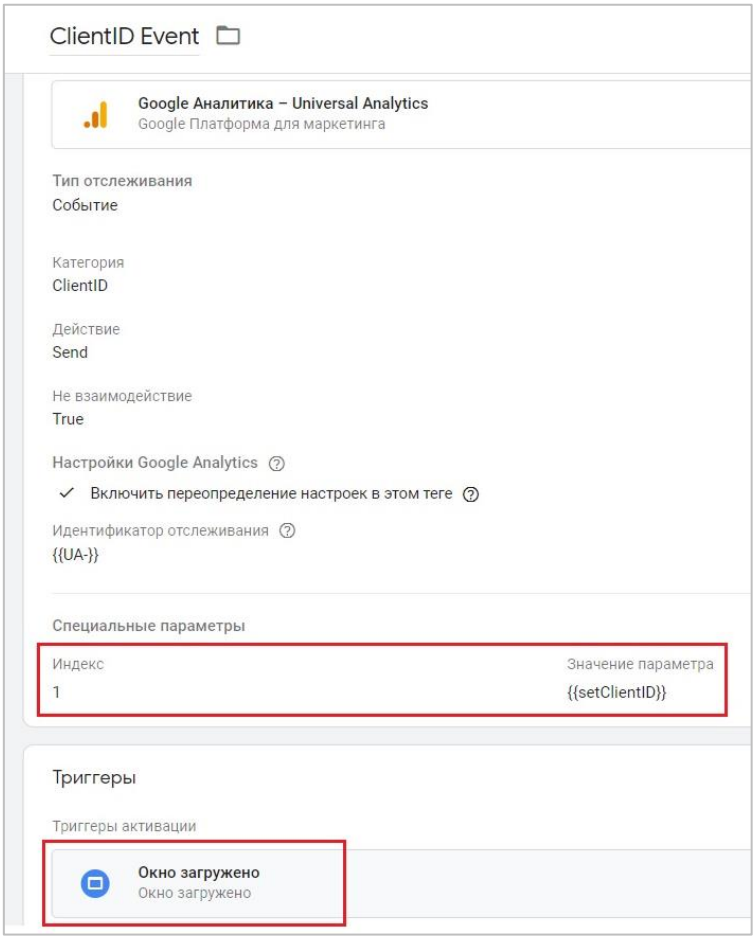


Рис. 554. Тег Google Аналитика - Universal Analytics с типом отслеживания Событие

Минус такого подхода – лишние обращения в Google Analytics.

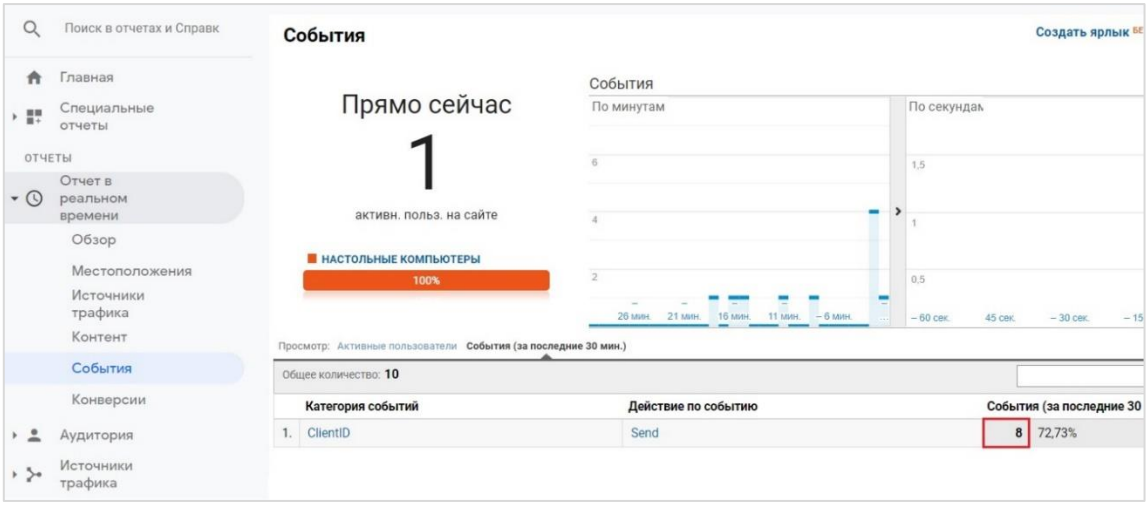


Рис. 555. Лишние хиты в Google Analytics

Способ №3 - из cookie _ga Google Analytics

Для этого необходимо создать пользовательскую переменную типа **Основной файл cookie (1st Party Cookie)** со значением **_ga**.

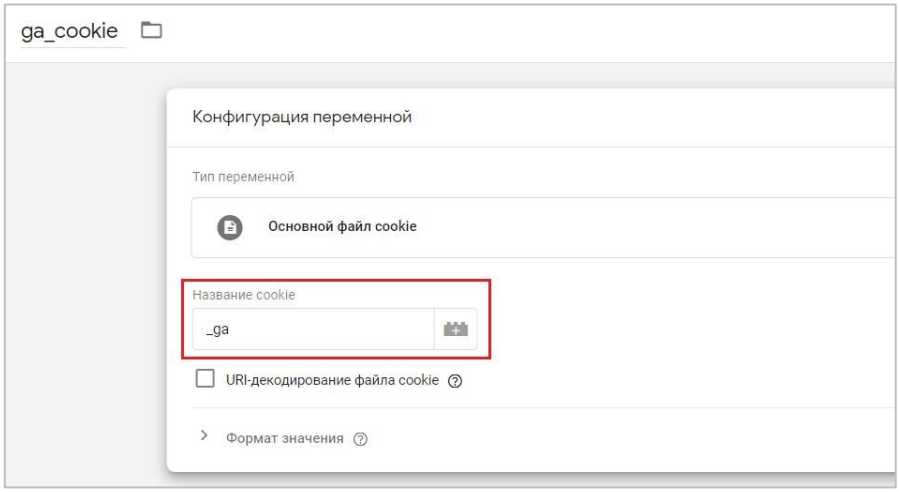


Рис. 556. Основной файл cookie _ga

Она возвращает значение cookie, которое доступно для текущего сайта. По умолчанию в Analytics – это **_ga**.

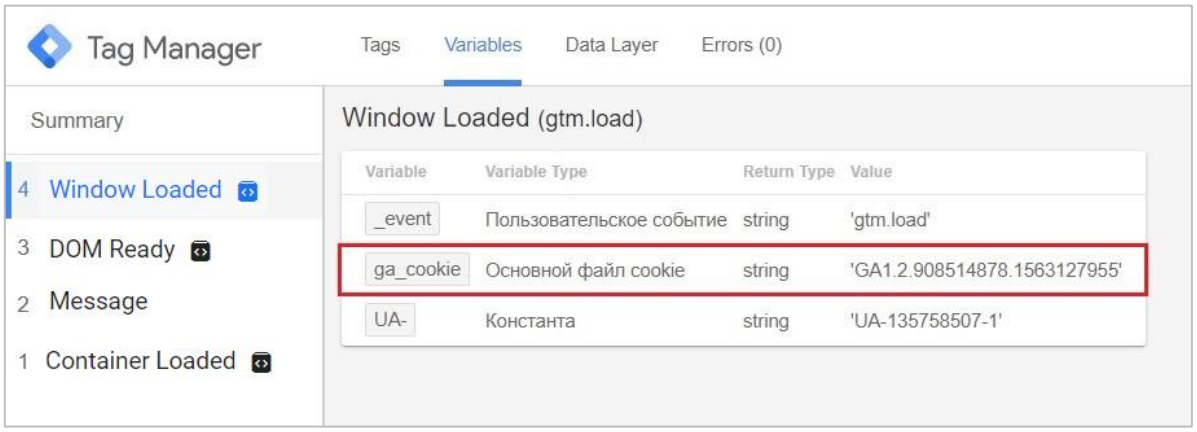


Рис. 557. Полная кука с префиксом GA.1.2.

Как видно из рисунка выше, основной файл cookie выводит полное значение, включая номер GA версии и уровень домена. Нам же нужна только последняя часть - **908514878.1563127955**.

В Google Tag Manager создаем пользовательскую переменную типа **Собственный код JavaScript** и вставить туда следующий код:

```
function() {
  try {
    var cookie = {{ga_cookie}}.split(".");
    return cookie[2] + "." + cookie[3];
  } catch(e) {
    console.log("Файл cookie Universal Analytics не найден ");
  }
}
```

где **ga_cookie** – название вашей переменной **Основной файл cookie** (см. шаг выше).

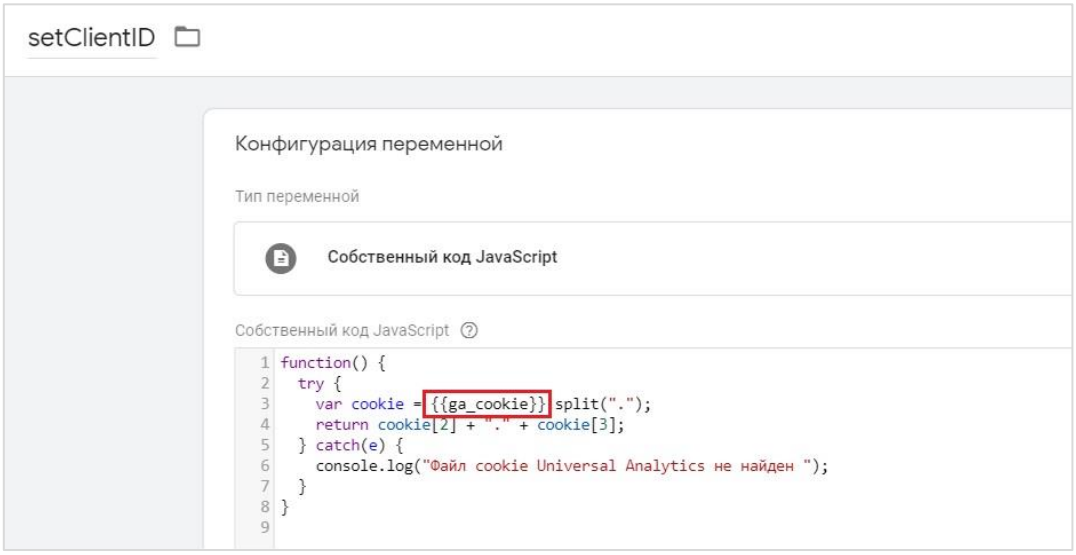


Рис. 558. Собственный код JavaScript

Этот код JavaScript анализирует строку и возвращает третий (идентификатор клиента) и четвертый (временная метка) элементы значения cookie (**908514878.1563127955**), которые создают полный Client ID пользователя.

Все дальнейшие действия аналогичны. Создаете пользовательский параметр, запоминаете индекс, и затем добавляете данные в тег Universal Analytics:

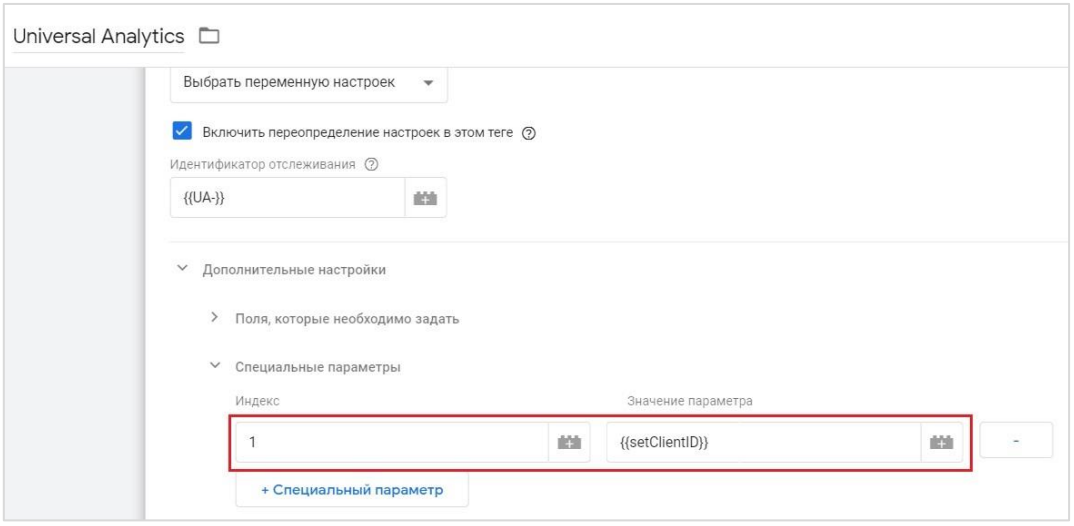


Рис. 559. Добавление специального параметра в тег

Бонус: если код Google Analytics установлен через пользовательский HTML

Не все веб-аналитики используют готовый тег Universal Analytics. Некоторые добавляют код через **Пользовательский HTML**, чтобы в работе использовать команды из библиотеки gtag.js. Для этого используется параметр **custom_map** (см. приложение), а конечный код выглядит следующим образом:

```
  <!-- Global site tag (gtag.js) - Google Analytics -->
  <script  async  src="https://www.googletagmanager.com/gtag/js?id='UA-XXXXXXXX-
X"></script>
  <script>
    window.dataLayer = window.dataLayer || [];
    function gtag(){dataLayer.push(arguments);}
```

```
    gtag('js', new Date());
    gtag('config', 'UA-XXXXXXXX-X', {
      'custom_map': {
          'dimensionN': 'clientId'
      }
      });
</script>
```

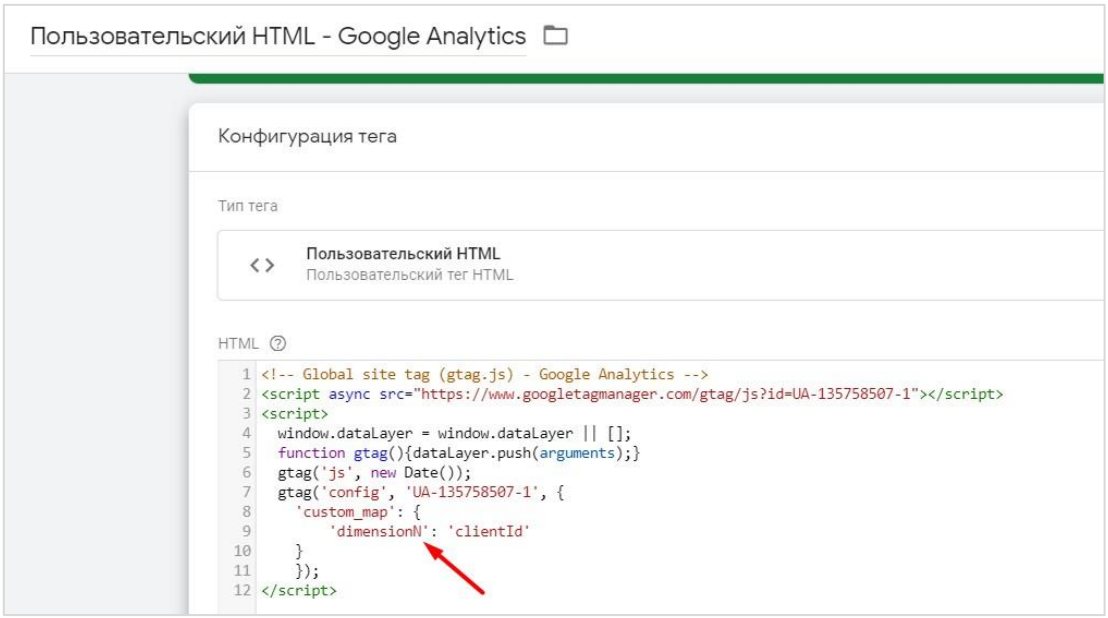


Рис. 560. Пользовательский HTML тег - Google Analytics

, где вместо **N** – ваш индекс пользовательского параметра, а **UA-XXXXXXXX-X** – ваш № счетчика.

Таким образом, мы с вами познакомились с несколькими способами получения Client ID и с передачей его значения в Google Analytics с помощью Google Tag Manager. Если происходящее событие имеет два и более различающихся решения, ведущих к одному ответу, то верным будет самое простое решение, которое требует меньше неизвестных.

Общий тег событий

Чтобы на сайте отслеживать нужную информацию и передавать ее в инструменты веб-аналитики, нам приходится настраивать большое количество различных переменных, триггеров и тегов. Встречаются события, для которых следует создавать много разных сущностей. Но чаще такие события отличаются только названием события и полями тегов (Категория, Действие, Ярлык события).

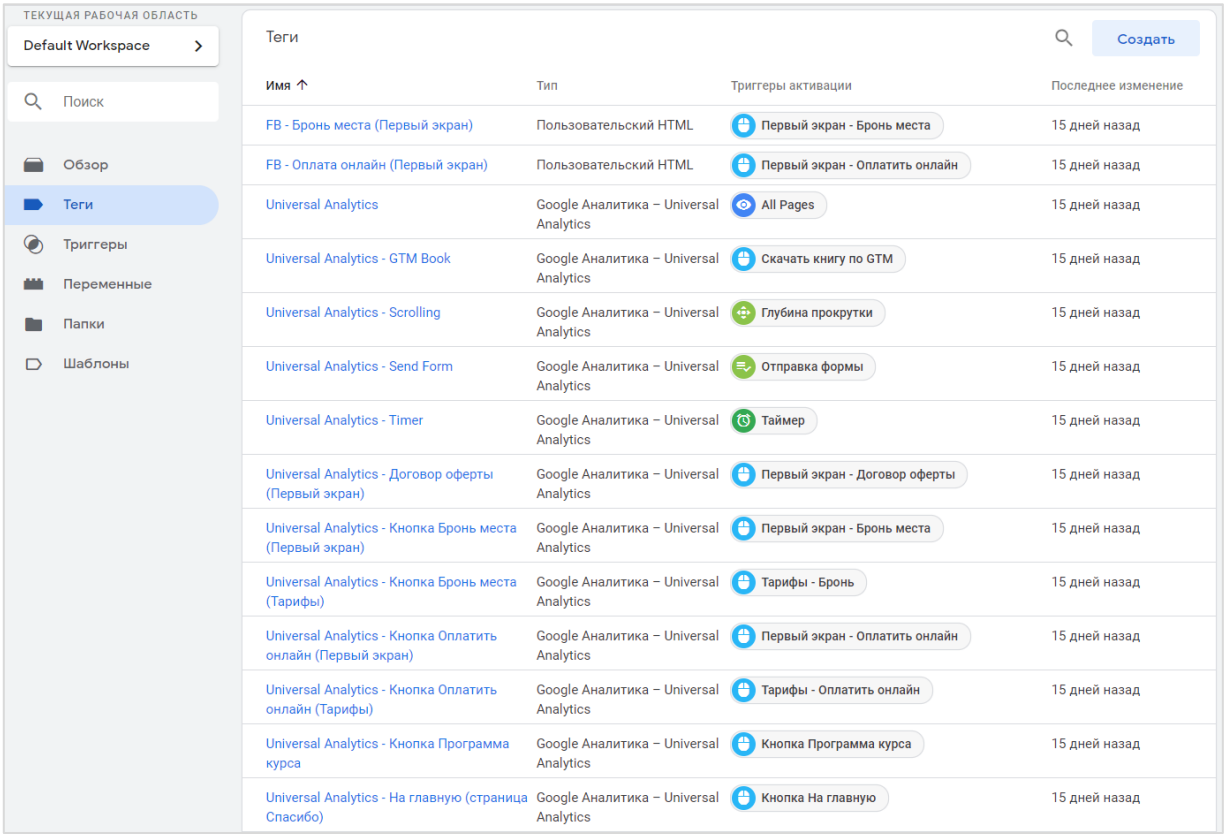


Рис. 561. Под каждое событие на сайте - свой тег Universal Analytics

Получается, что под каждое событие на сайте вы создаете свой собственный тег **Google Аналитика - Universal Analytics** с типом отслеживания **Событие**. А чем больше у нас тегов, тем сложнее становится управлять аккаунтом Google Tag Manager.

Для упрощения работы с контейнером можно воспользоваться универсальным способом отслеживания событий с помощью конструкции dataLayer (уровень данных), используя общий синтаксис:

```
window.dataLayer = window.dataLayer || [];
dataLayer.push({
  'event': 'mainEvent',
  'category': 'eventCategory',
  'action': 'eventAction',
  'label': 'eventLabel',
  'value': 'eventValue'
});
```

, где:

- **'event': 'mainEvent'** - неизменное название для наших событий (может быть другим);
- **'category': 'eventCategory'** - категория события (может быть другим);
- **'action': 'eventAction'** - действие события (может быть другим);
- **'label': 'eventLabel'** - ярлык события (может быть другим);
- **'value': 'eventValue'** - ценность события (может быть другим).

Код такого вида ваш разработчик должен добавить на все элементы, которые вы хотите отслеживать. Причем 4 переменных будут иметь для каждого отслеживаемого элемента на сайте свои значения. К примеру, давайте настроим общий тег для сайта graphanalytics.ru на отслеживание клика по 3 кнопкам:

- Консультация;
- Заказать рекламу;
- Скачать книгу.

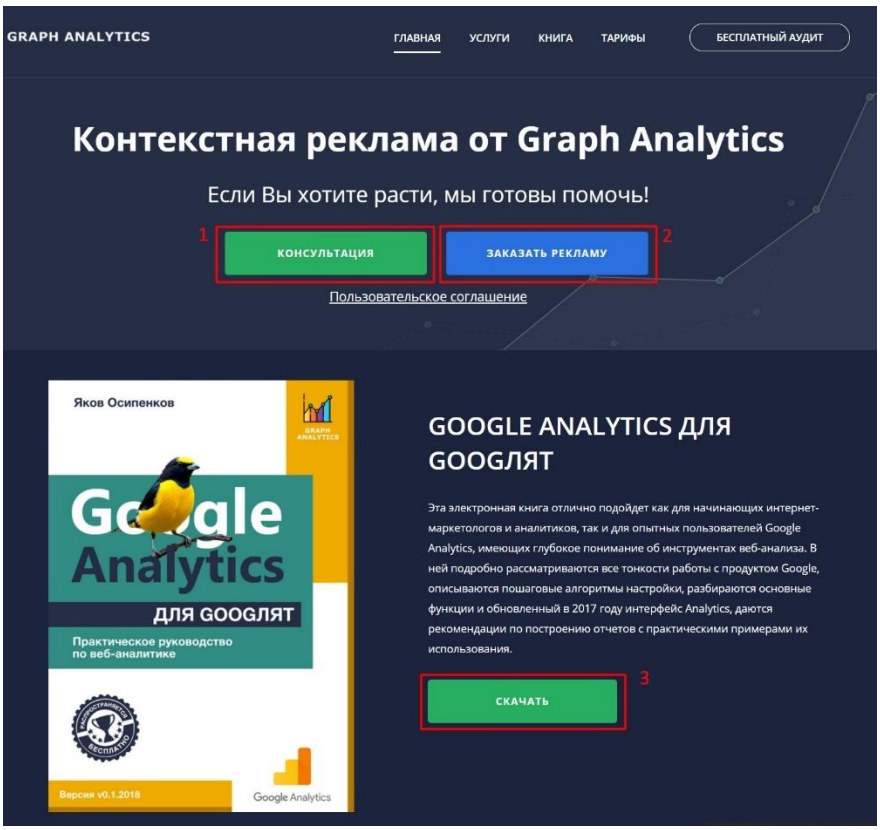


Рис. 562. Настройка трех кнопок одним общим тегом

Примечание: ярлык события и ценность события являются необязательными полями, поэтому их можно не назначать. Но поскольку переменные уровня данных связаны вместе с моделью данных GTM, рекомендуется все же помечать как **undefined** поля, которые вы не используете (см. приложение).

Для моего примера коды на кнопках будут выглядеть так (событие onclick, клик):

```
                    <h1 data-ix="slider-title" class="slide-title" style="opacity: 1; transform: translateX(0px) translateY(0px)
                    translateZ(0px); transition: opacity 500ms, transform 500ms;">
                        Контекстная реклама от Graph Analytics
                    </h1>
                    <h2 data-ix="slider-title-2" class="slide-title subtitle" style="opacity: 1; transform: translateX(0px)
                    translateY(0px) translateZ(0px); transition: opacity 500ms, transform 500ms;">
                        Если Вы хотите расти, мы готовы помочь!
                    </h2>
                    <div class="first-buttons">
                        <a href="#popup-audit" class="button hp-button-1 slide-button w-button green fancybox" onclick=
                        "window.dataLayer = window.dataLayer || [];
dataLayer.push({
  'event': 'mainEvent',
  'category': 'greenButton',
  'action': 'click',
  'label': 'undefined',
  'value' : 'undefined'
});">Консультация</a>
                        <a href="#price" class="button hp-button-1 slide-button w-button scrollingto blue" onclick="window.dataLayer =
                        window.dataLayer || [];
dataLayer.push({
  'event': 'mainEvent',
  'category': 'blueButton',
  'action': 'click',
  'label': '1view',
  'value' : '100'
});">Заказать рекламу</a>
                    </div>
                    <a class="policy-a fancybox" href="#policy-2">Пользовательское соглашение</a>
```

Кнопка 1: Консультация

Кнопка 2: Заказать рекламу

Рис. 563. Коды кнопок Консультация и Заказать рекламу

Пример кода для зеленой кнопки Консультация:

```
window.dataLayer = window.dataLayer || [];
dataLayer.push({
  'event': 'mainEvent',
  'category': 'greenButton',
  'action': 'click',
  'label': 'undefined',
  'value' : 'undefined'
});
```

Примечательно, что для одной кнопки я указал значения ярлыка и ценности, а для другой оставил их не присвоенными (undefined). Я это сделал осознанно, чтобы показать разницу. Аналогичный код и для третьей кнопки СКАЧАТЬ, только в категории и действии написаны другие значения:

```
и обновленный в 2017 году интерфейс Analytics, дается
рекомендации по построению отчетов с практическими примерами
их использования.</p>
<a href="
https://graphanalytics.ru/Google_Analytics_dlya_googlyat_2018.
pdf" onclick="window.dataLayer = window.dataLayer || [];
dataLayer.push({
  'event': 'mainEvent',
  'category': 'downloadBook',     Кнопка 3: Скачать
  'action': 'click',
  'label': 'GA',
  'value' : '50'
});" class="button hp-button-1 slide-button w-button green">Скачать</a>
                    </div>
                </div>
            </div>
```

Рис. 564. Код для кнопки Скачать

Событие **'event'** для всех кнопок одинаковое - **mainEvent**. Название этого события нам пригодится в дальнейшем при настройке триггера активации.

Все вышеописанное касается работы разработчика. Теперь настало наше время. Для того, чтобы извлечь информацию из уровня данных, нам необходимо создать 4 пользовательских переменных типа Переменная уровня данных с соответствующими названиями: category, action, label, value (с такими же именами, как указано в коде).

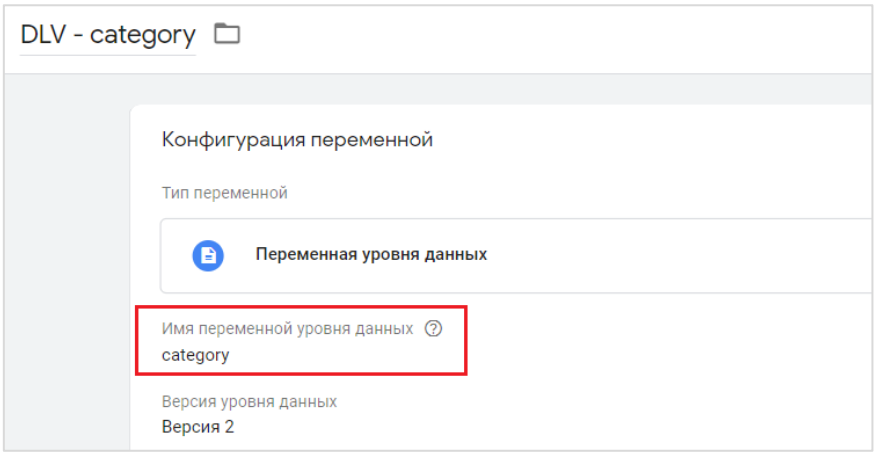


Рис. 565. Переменная уровня данных

Повторяем с оставшимися 3 переменными. В результате получаем 4 пользовательские переменные:

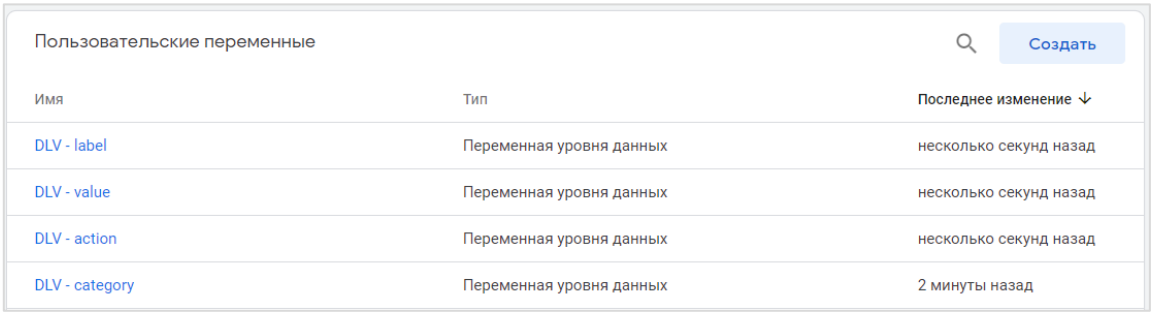


Рис. 566. Четыре переменных уровня данных

На следующем шаге создаем триггер типа **Пользовательское событие** с названием события **mainEvent**. Оно общее для всех наших кнопок, что как раз и упрощает отслеживание.

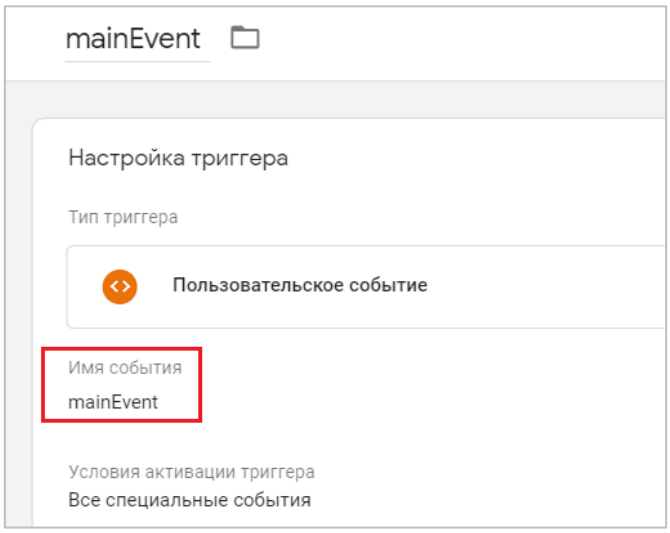


Рис. 567. Пользовательское событие mainEvent

На последнем шаге создается один тег типа **Google Аналитика - Universal Analytics** с типом отслеживания **Событие**, где в качестве 4 компонентов используются наши переменные уровня данных (см. выше). А триггер активации - пользовательское событие **mainEvent**.

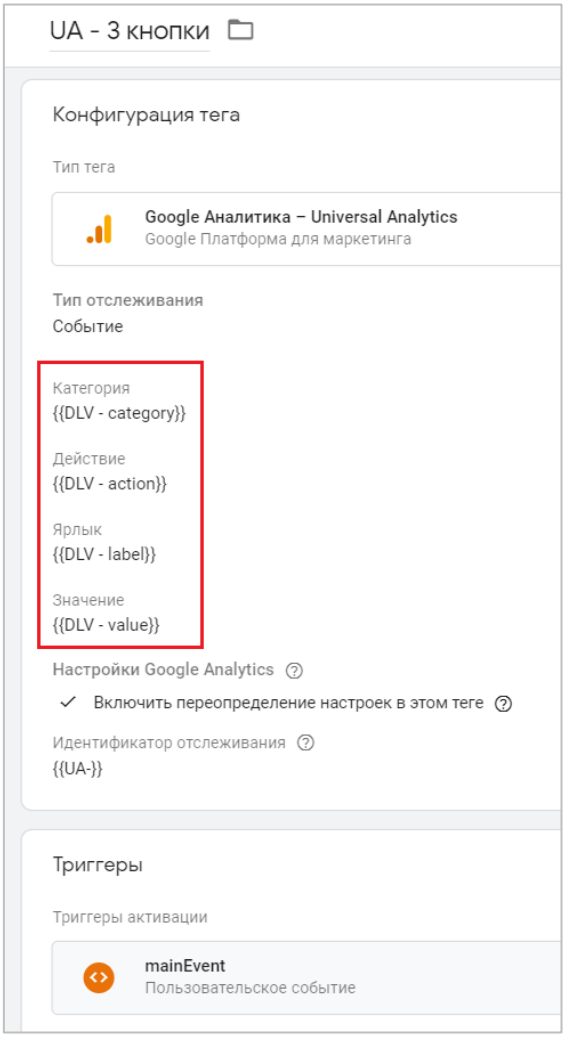


Рис. 568. Общий тег событий

Сохраняем настройки. Все, что осталось сделать, это проверить корректность отслеживания наших событий. Для этого включаем режим предварительного просмотра GTM и отчеты **В режиме реального времени** в Google Analytics. Кликаем поочередно по 3 кнопкам, которые отслеживаем.

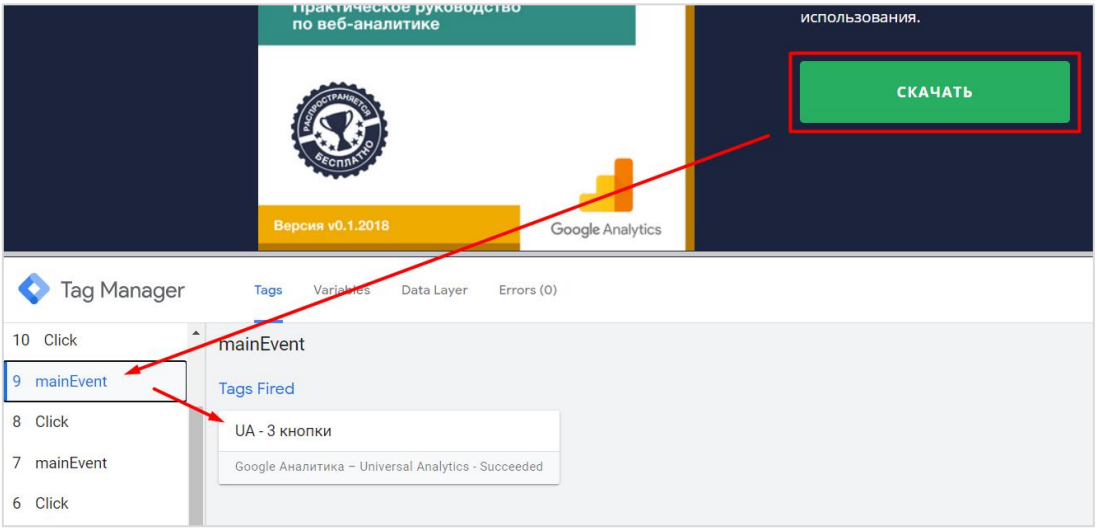


Рис. 569. Событие срабатывает, тег активируется

При клике на отслеживаемый элемент событие **mainEvent** и тег Google Аналитики срабатывает для всех 3 кнопок. А если перейти на вкладку **Data Layer** с наведенным событием mainEvent, то можно увидеть, какая информация передается на уровне данных. Для кнопки **СКАЧАТЬ** — это:

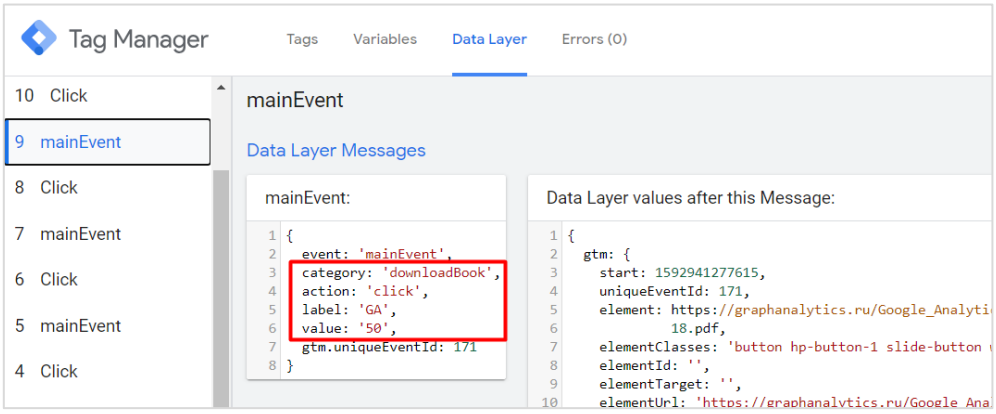


Рис. 570. Уровень данных для кнопки Скачать

, где **category, action, label, value** - то, что прописывал разработчик в коде кнопки. В Google Analytics информация также корректно передалась:

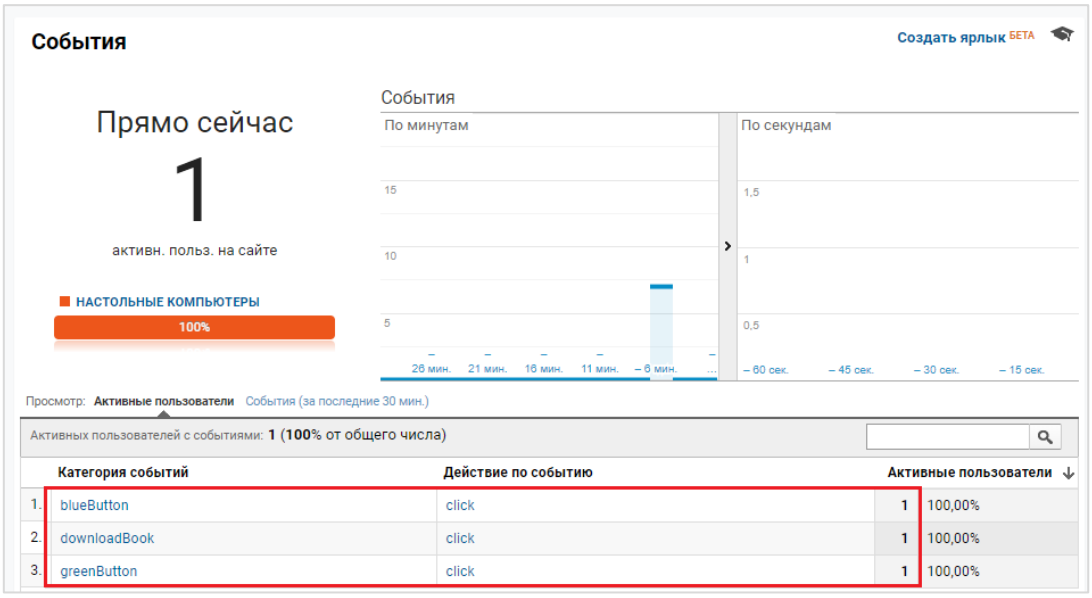


Рис. 571. Отчет В режиме реального времени

Таким образом, для отслеживания трех кнопок нам понадобилось всего 4 переменных, 1 триггер и 1 тег. А в обычной настройке, скорее всего, вы бы воспользовались 3 отдельными триггерами и 3 тегами. А если бы у нас было 10 кнопок? Все равно достаточно 4 переменных, 1 триггера и 1 тега. Через некоторое время все данные по событиям появятся в отчете **Поведение — События — Лучшие события**.

С помощью общего тега событий вы можете существенно сократить настройку триггеров и тегов в Google Tag Manager. Для этого нужно только попросить разработчика поместить информацию о событиях в уровень данных, оставив название события без изменений. Таким способом, традиционно, настраивается расширенная электронная торговля (Enhanced Ecommerce) по известному в русском сообществе руководству от компании Neatpeak. Подробнее настройка будет разобрана в соответствующей главе.

Точное время обращения (Hit Timestamp)

Вам необходимо определить точное время, когда пользователь на сайте просмотрел страницу, совершил событие, создал транзакцию? Сопоставить данные с логами сервера или сравнить информацию в Google Analytics со сторонними сервисами? Вам поможет пользовательский параметр **Hit Timestamp**.

По умолчанию в стандартных отчетах Google Analytics нет детальной статистики по секундам и общей дате совершения какого-либо хита. Разве, что выбрать час или минуты, отдельно дату или время суток.

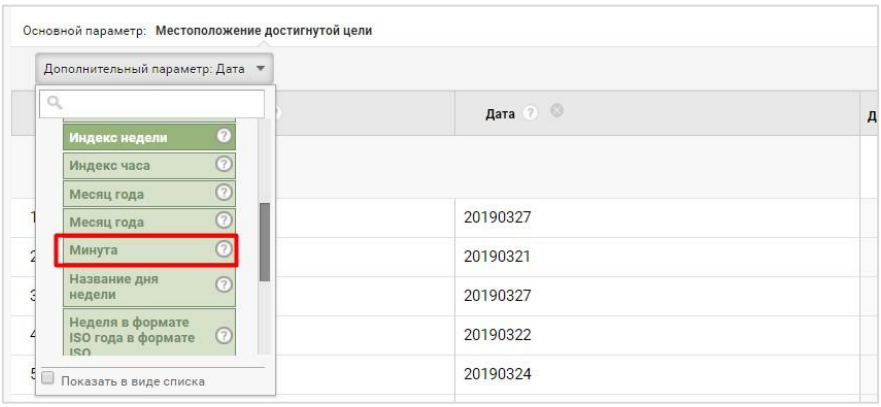
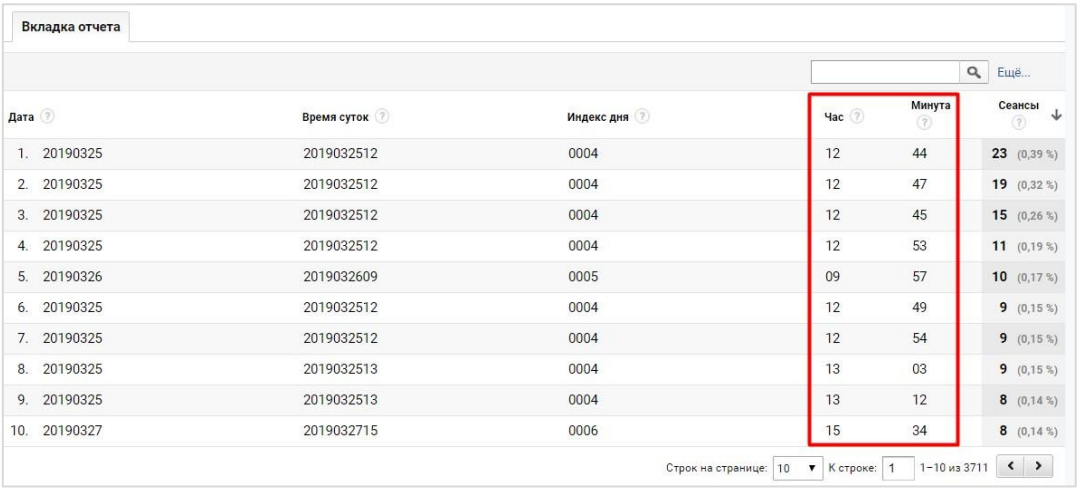


Рис. 572. Параметры группы Время

Можно попробовать создать специальный отчет, в котором выбрать все предложенные временные метрики. И тогда получится примерно следующее:



	Дата	Время суток	Индекс дня	Час	Минута	Сеансы
1.	20190325	2019032512	0004	12	44	23 (0,39 %)
2.	20190325	2019032512	0004	12	47	19 (0,32 %)
3.	20190325	2019032512	0004	12	45	15 (0,26 %)
4.	20190325	2019032512	0004	12	53	11 (0,19 %)
5.	20190326	2019032609	0005	09	57	10 (0,17 %)
6.	20190325	2019032512	0004	12	49	9 (0,15 %)
7.	20190325	2019032512	0004	12	54	9 (0,15 %)
8.	20190325	2019032513	0004	13	03	9 (0,15 %)
9.	20190325	2019032513	0004	13	12	8 (0,14 %)
10.	20190327	2019032715	0006	15	34	8 (0,14 %)

Рис. 573. Пример специального отчета в Google Analytics

При должном желании можно сопоставить дату и время с каким-либо обращением посетителя на сайте. Но поскольку пользователь может посетить несколько страниц в течение минуты, вы не можете получить точный

порядок последовательности их просмотра. Но можно сделать по-другому – создать пользовательский параметр, в котором будет представлена вся необходимая информация. Например, так:

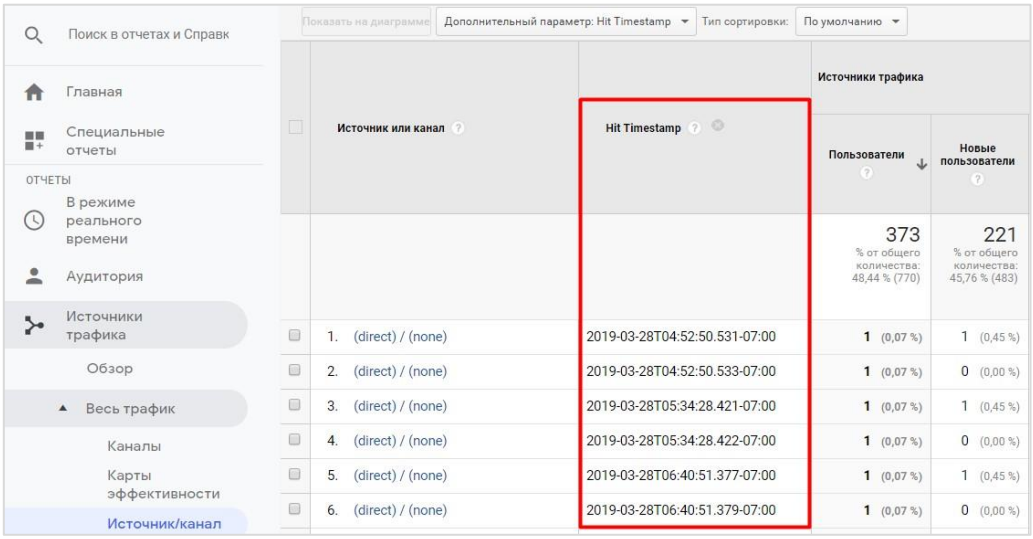


Рис. 574. Пользовательский параметр Hit Timestamp

В свое время Симо Ахава написал подробный материал (см. приложение) на тему улучшения сбора данных путем создания 4 специальных параметров в Google Analytics. Воспользуемся его JavaScript-кодом, чуть модифицировав формат вывода данных под себя. Результат получится таким:

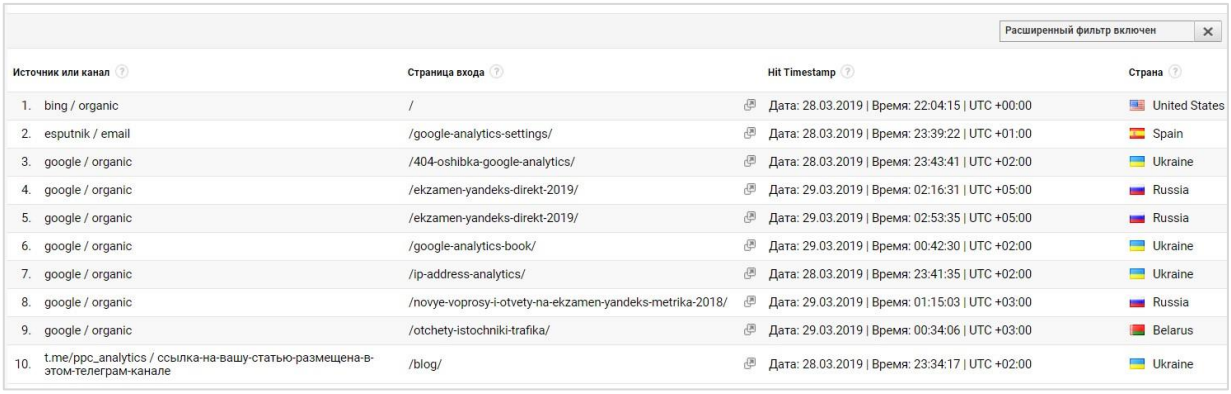


Рис. 575. Специальный отчет с привычным Hit Timestamp

Hit Timestamp - пользовательский параметр области действия **Хит (Hit)**, который показывает точное время совершения просмотра страницы, события, транзакции в формате **YYYY-MM-DDThh:mm:ss** со смещением часового пояса.

Разберем настройки пользовательского параметра через Google Tag Manager и напрямую. Для начала в Google Analytics создаем пользовательский параметр с именем **Hit Timestamp** и областью действия **Hit.** Запоминаем **Индекс** параметра. В нашем случае это 3.

В Google Tag Manager переходим на вкладку **Переменные** и создаем пользовательскую переменную JavaScript с кодом, которая будет возвращать значение в формате string (строка):

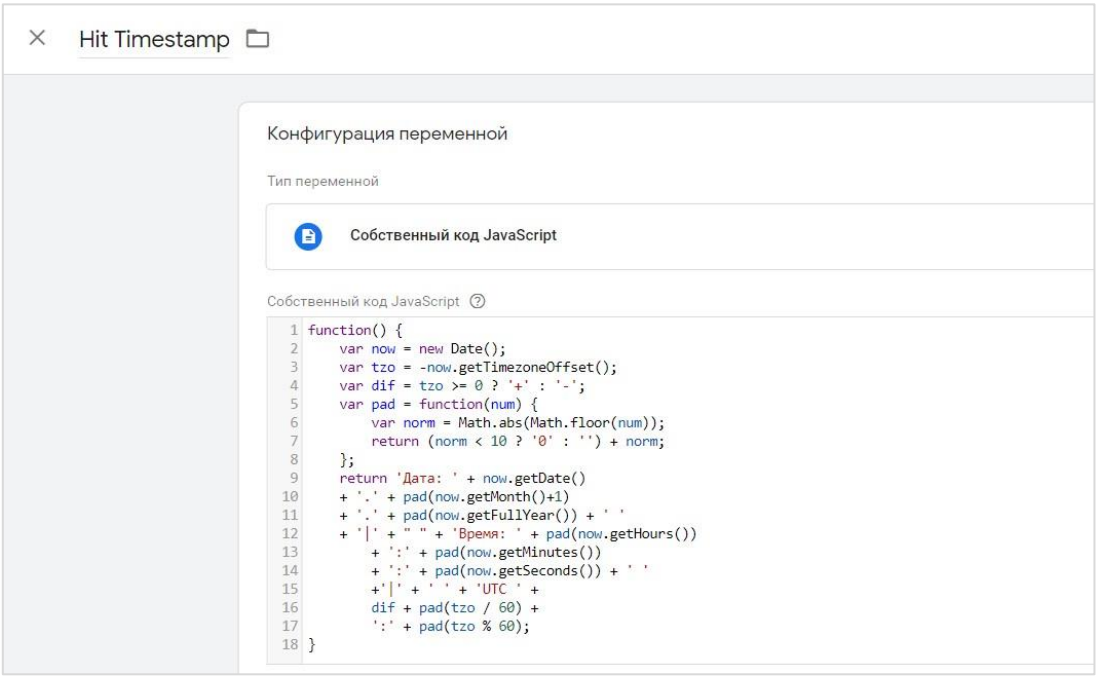


Рис. 576. Собственный код JavaScript

Сам код:

```
function() {
    var now = new Date();
    var tzo = -now.getTimezoneOffset();
    var dif = tzo >= 0 ? '+' : '-';
    var pad = function(num) {
        var norm = Math.abs(Math.floor(num));
        return (norm < 10 ? '0' : '') + norm;
    };
    return 'Дата: ' + now.getDate()
    + '.' + pad(now.getMonth()+1)
    + '.' + pad(now.getFullYear()) + ' '
    + '|' + " " + 'Время: ' + pad(now.getHours())
        + ':' + pad(now.getMinutes())
        + ':' + pad(now.getSeconds()) + ' '
        +'|' + ' ' + 'UTC ' +
        dif + pad(tzo / 60) +
        ':' + pad(tzo % 60);
}
```

Далее в основном теге Google Analytics в дополнительных настройках, связанных со специальными параметрами, для отслеживания просмотра страниц следует добавить наш индекс (см. в Google Analytics и рисунок выше) и значение параметра (наша пользовательская переменная).

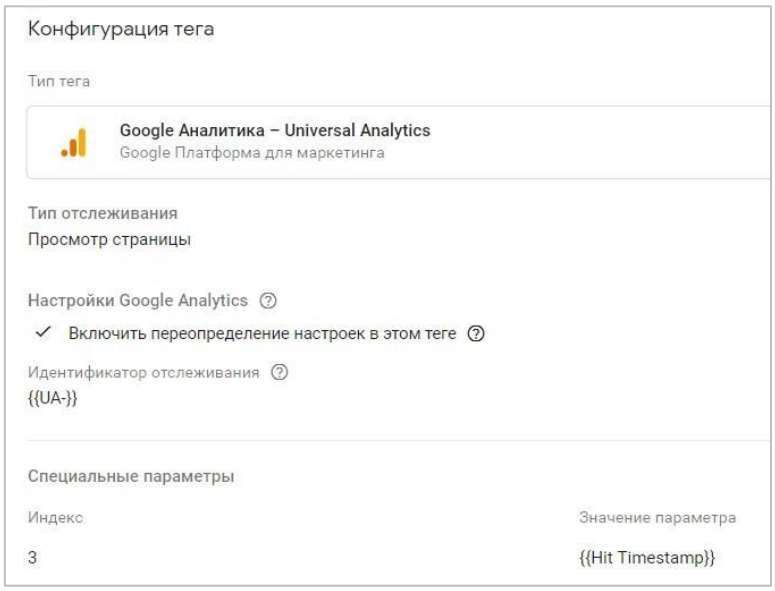


Рис. 577. Конфигурация тега со специальным параметром

Если вы хотите отслеживать только определенные события на сайте (или транзакции), добавьте его в соответствующий тег в GTM. Проверить корректность передачи данных можно с помощью режима предварительного просмотра:

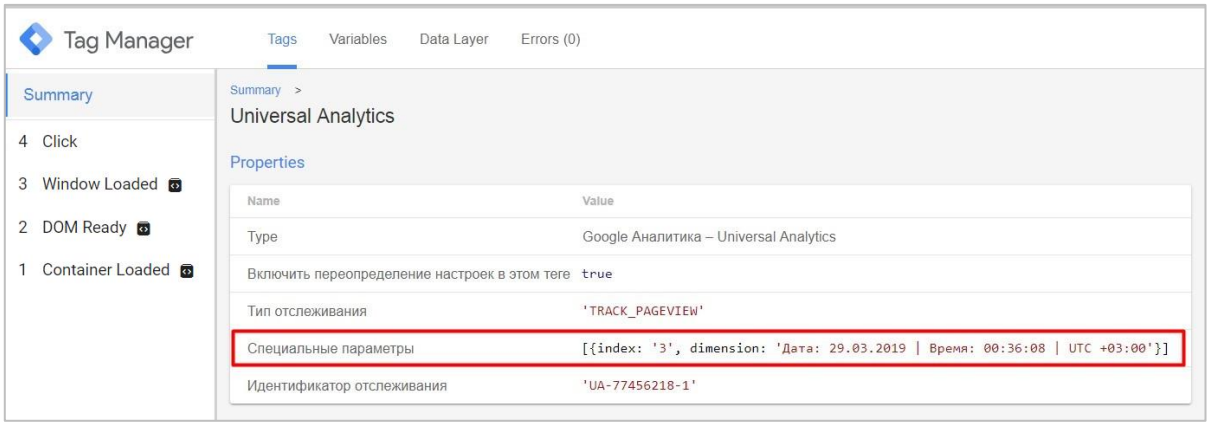


Рис. 578. Режим отладки

После этого в отчетах Google Analytics вы можете применять пользовательский параметр в качестве дополнительного.

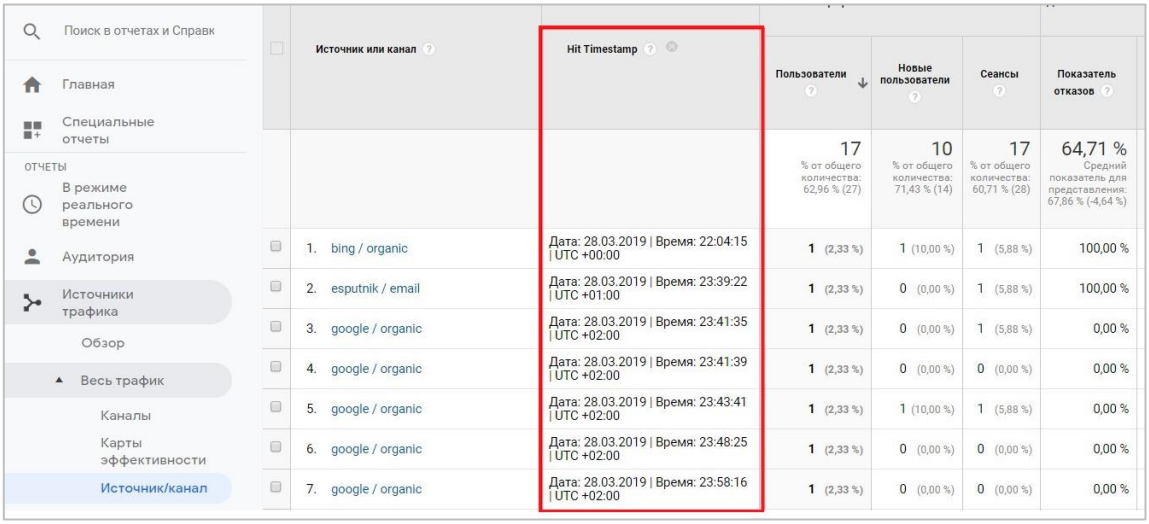
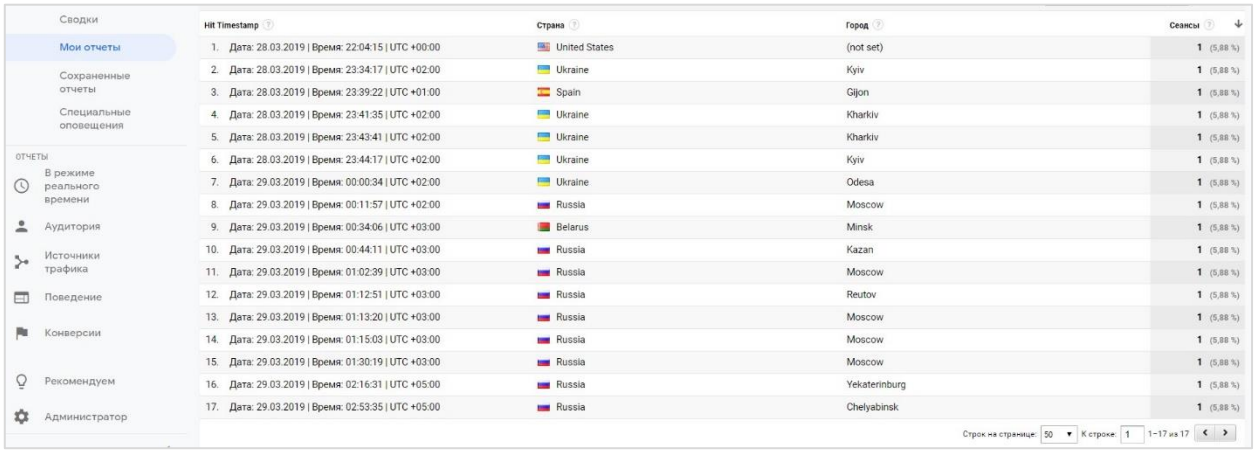


Рис. 579. Дополнительный параметр в отчете

Также можно также построить специальный отчет, чтобы сопоставить данные всемирного координированного времени (UTC) с местоположением пользователя и убедиться в подсчете значений.



	Hit Timestamp	Страна	Город	Сеансы
1.	Дата: 28.03.2019 | Время: 22:04:15 | UTC +00:00	United States	(not set)	1 (5,88 %)
2.	Дата: 28.03.2019 | Время: 23:34:17 | UTC +02:00	Ukraine	Kyiv	1 (5,88 %)
3.	Дата: 28.03.2019 | Время: 23:39:22 | UTC +01:00	Spain	Gijon	1 (5,88 %)
4.	Дата: 28.03.2019 | Время: 23:41:35 | UTC +02:00	Ukraine	Kharkiv	1 (5,88 %)
5.	Дата: 28.03.2019 | Время: 23:43:41 | UTC +02:00	Ukraine	Kharkiv	1 (5,88 %)
6.	Дата: 28.03.2019 | Время: 23:44:17 | UTC +02:00	Ukraine	Kyiv	1 (5,88 %)
7.	Дата: 29.03.2019 | Время: 00:00:34 | UTC +02:00	Ukraine	Odesa	1 (5,88 %)
8.	Дата: 29.03.2019 | Время: 00:11:57 | UTC +02:00	Russia	Moscow	1 (5,88 %)
9.	Дата: 29.03.2019 | Время: 00:34:06 | UTC +03:00	Belarus	Minsk	1 (5,88 %)
10.	Дата: 29.03.2019 | Время: 00:44:11 | UTC +03:00	Russia	Kazan	1 (5,88 %)
11.	Дата: 29.03.2019 | Время: 01:02:39 | UTC +03:00	Russia	Moscow	1 (5,88 %)
12.	Дата: 29.03.2019 | Время: 01:12:51 | UTC +03:00	Russia	Reutov	1 (5,88 %)
13.	Дата: 29.03.2019 | Время: 01:13:20 | UTC +03:00	Russia	Moscow	1 (5,88 %)
14.	Дата: 29.03.2019 | Время: 01:15:03 | UTC +03:00	Russia	Moscow	1 (5,88 %)
15.	Дата: 29.03.2019 | Время: 01:30:19 | UTC +03:00	Russia	Moscow	1 (5,88 %)
16.	Дата: 29.03.2019 | Время: 02:16:31 | UTC +05:00	Russia	Yekaterinburg	1 (5,88 %)
17.	Дата: 29.03.2019 | Время: 02:53:35 | UTC +05:00	Russia	Chelyabinsk	1 (5,88 %)

Строк на странице: 50 К строке: 1 1–17 из 17

Рис. 580. Специальный отчет с Hit Timestamp

Например, UTC +03:00 – Russia (Moscow), а UTC +02:00 – Ukraine (Kyiv).

Глава 8

Работа с формами

Второй по значимости настройкой (а может быть и первой) у интернет-маркетологов является отслеживание различных форм на сайте. Обращение, заполненное и отправленное с контактной информацией пользователя, как правило, говорит об интересе к вашему товару или услуге, что, в свою очередь, может в дальнейшем привести к реальным продажам. Поэтому крайне важно правильно настроить сбор и передачу данных об этом событии в инструменты веб-аналитики.

В этой главе мы рассмотрим несколько способов отслеживания отправки форм с помощью Google Tag Manager, а именно:

1. с помощью отдельной страницы;
2. с помощью триггера **Отправка формы**;
3. с помощью триггера **Доступность элемента**;
4. с помощью триггера **Пользовательское событие** и уровня данных (dataLayer);
5. с помощью прослушивателя автоматических событий;
6. с помощью универсального кода для форм на **AJAX**;
7. с помощью **DOM Scraping**;
8. с помощью виртуальных страниц;
9. брошенные формы.

Примечание: в зависимости от того, как у вас на сайте реализована отправка формы, вы выбираете для себя наиболее оптимальный вариант отслеживания.

Разберем каждый способ более подробно.

1. Отслеживание отправки формы с помощью отдельной страницы

Суть этого способа заключается в том, что после отправки формы пользователя перенаправляет на отдельную страницу с уникальным URL-адресом. Еще ее называют страницей **Спасибо**.

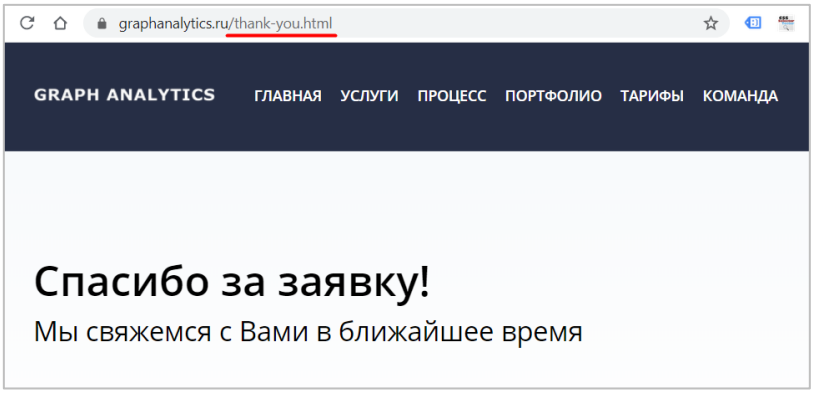


Рис. 581. Отслеживание формы с помощью отдельной страницы

В этом случае нет необходимости использовать переменные, триггеры и теги Google Tag Manager. Достаточно просто в Яндек.Метрике и Google Analytics настроить цель на посещение этой страницы:

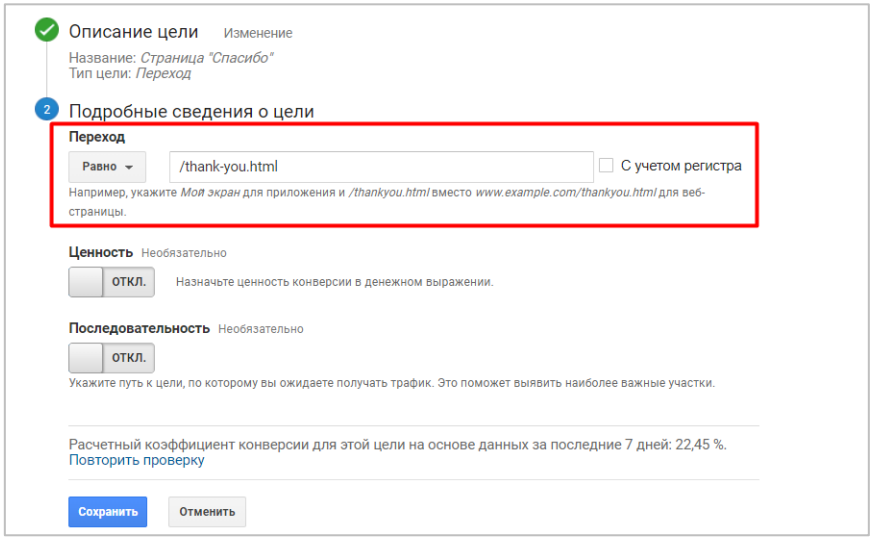


Рис. 582. Настройка цели на целевую страницу

Но если вы все же захотите настроить отслеживание как событие через GTM, то можете воспользоваться триггером **Просмотр страницы** с условием активации на конкретной странице (странице Спасибо):

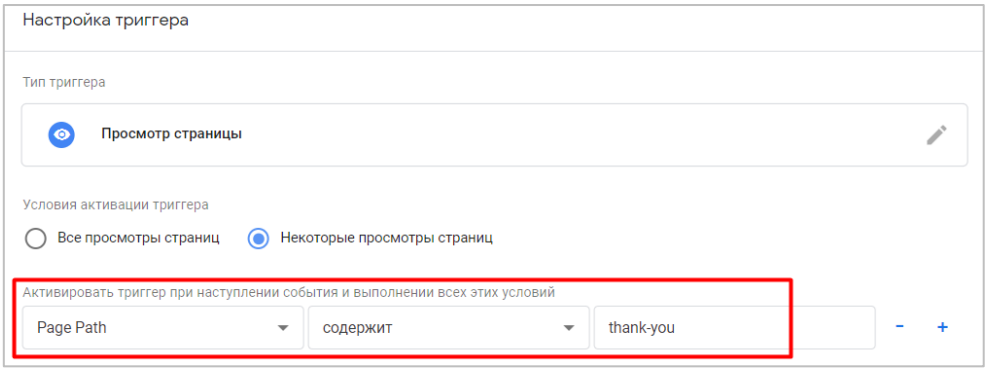


Рис. 583. Триггер Просмотр страницы

В моем примере – это **Page Path содержит thank-you**, поскольку URL-страницы имеет адрес **/thank-you.html**. А тег, который используется для передачи данных в Google Analytics, имеет вид:

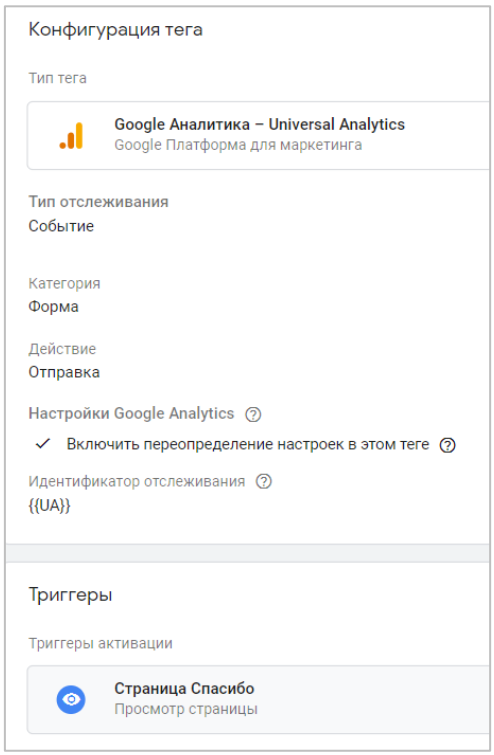


Рис. 584. Настройка тега

Через некоторое время все данные по событиям появятся в Google Analytics отчете **Поведение — События — Лучшие события**. Если вы планируете отслеживать отправку формы не только как события, но и как цель-события (конверсии), настройте цель с типом **Событие**:

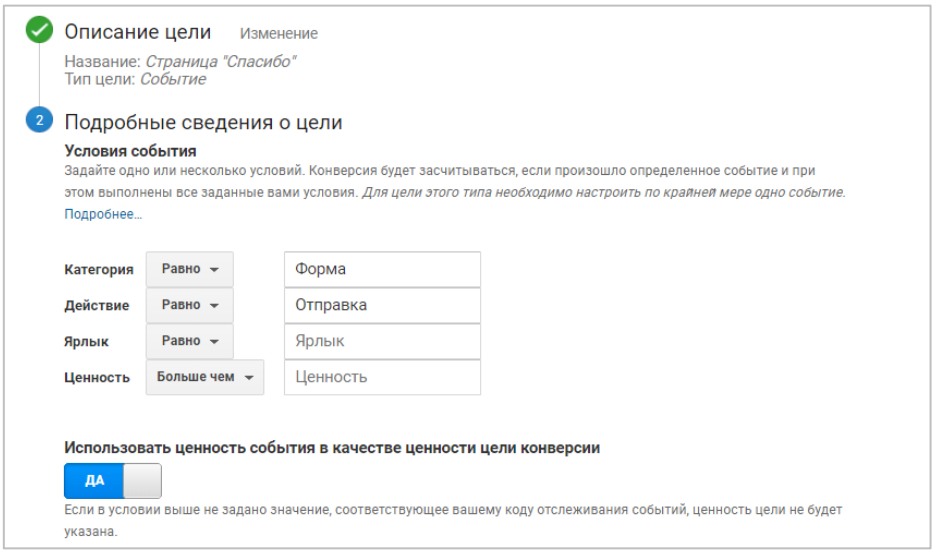


Рис. 585. Настройка цели-события в Google Analytics

Данный способ является наиболее простым в отслеживании.

2. Отслеживание отправки формы с помощью триггера Отправка формы

Если на вашем сайте есть форма с тегом **<form>** и кнопкой, которые имеют синтаксис:

```
<form action="URL">
  ...
<input type="submit">
</form>
```

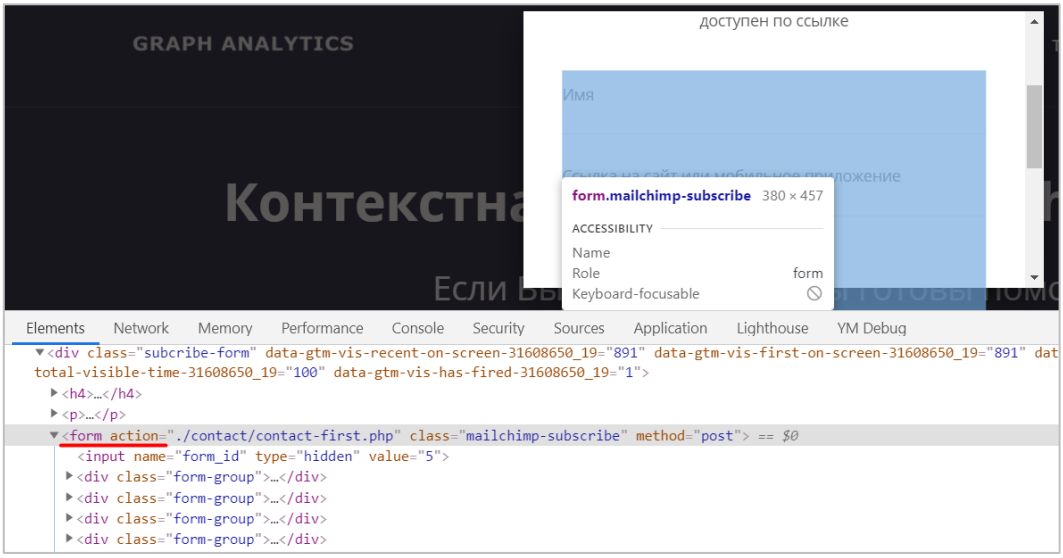


Рис. 586. Тег <form>

то вы можете попробовать использовать стандартный триггер **Отправка формы**. Но перед тем, как это сделать, необходимо активировать встроенные переменные типа **Формы**, которые позволят вам настроить корректное условие активации:

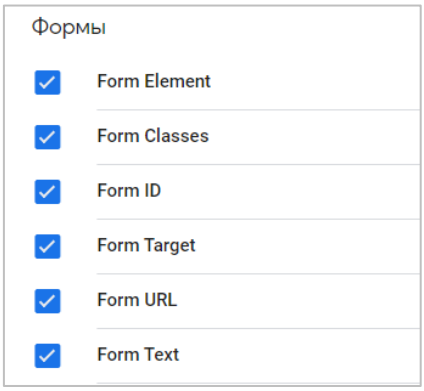


Рис. 587. Активация встроенных переменных

Затем создайте триггер типа **Отправка формы**. О том, что означают настройки данного триггера (**Ждать теги** и **Проверка ошибок**), читайте в соответствующей главе, посвященной триггерам Google Tag Manager. Пока зададим условие активации на все формы и сохраним изменения.

Чтобы проверить, работает ли данный триггер применительно к вашей форме, включите режим предварительного просмотра, а затем отправьте форму на сайте. В качестве примера я это сделаю на сайте **gtm.osipenkov.ru**.

Если событие **Form Submit** появилось в консоли предварительного просмотра, то вы можете использовать стандартный прослушиватель для своей формы. Воспользуйтесь значениями активированных переменных, чтобы узнать, есть ли у данной формы идентификатор (id) или класс (class), и активировать триггер только для конкретной формы, а не всех, которые есть на сайте. Для этого кликните на событие **Form Submit** и перейдите на вкладку **Variables**:

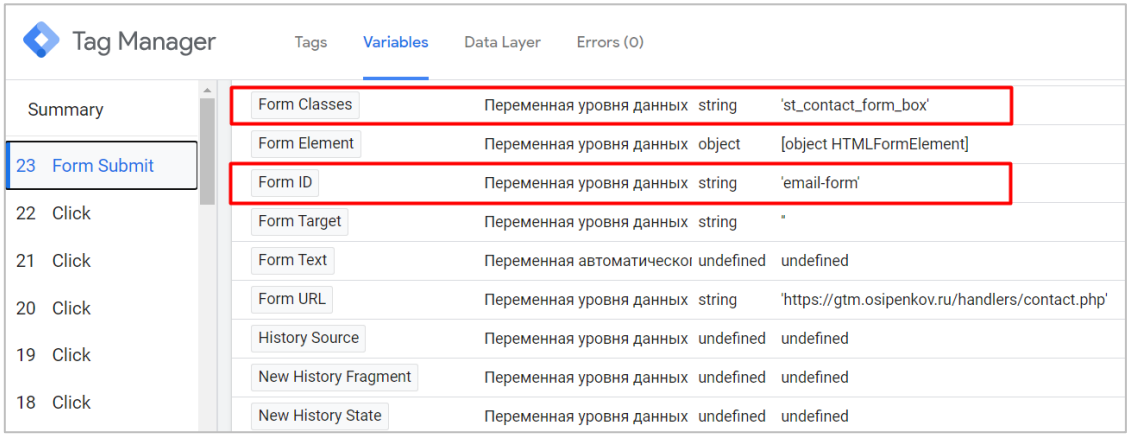


Рис. 588. Идентификатор и класс формы

Как видим из рисунка выше, у данной формы есть и то, и то. Поэтому мы можем указать в качестве условия активации триггера **Form Classes содержит st_contact_form_box** или **Form ID содержит email-form**.

Например, если у вас на сайте есть несколько форм с одинаковыми значениями класса или идентификатора, вы можете добавить для триггера дополнительное условие, например, отслеживать отправку формы только на странице **Контакты**:

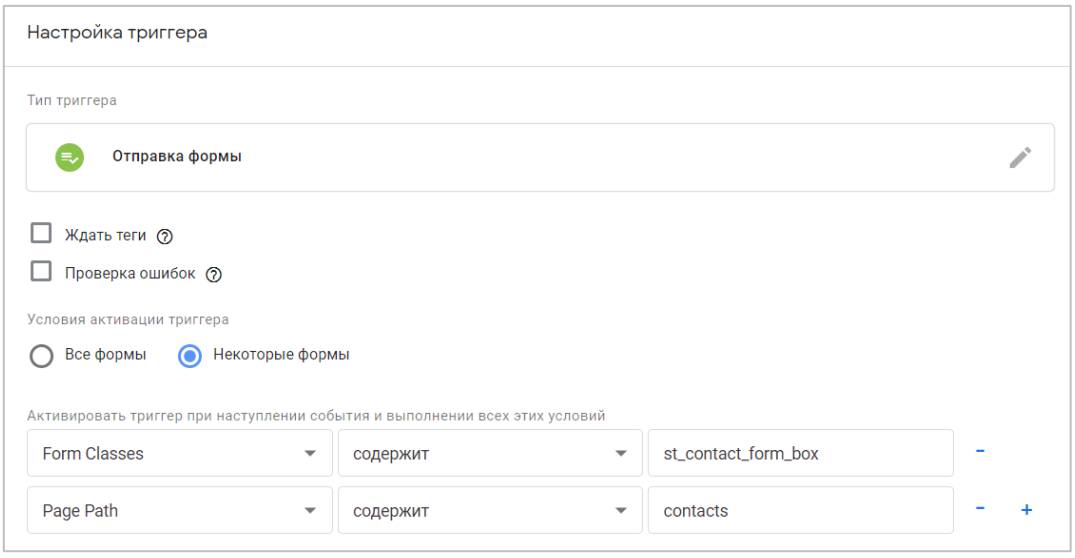


Рис. 589. Настройки триггера Отправка формы

Сохраним настройки триггера. Чтобы передать информацию об успешной отправке формы в Google Analytics необходимо создать тег **Google Аналитика – Universal Analytics** с типом отслеживания **Событие**, задать соответствующие настройки **Категории** и **Действия**, и в качестве триггера активации выбрать созданный триггер типа **Отправка формы**. Например, так:

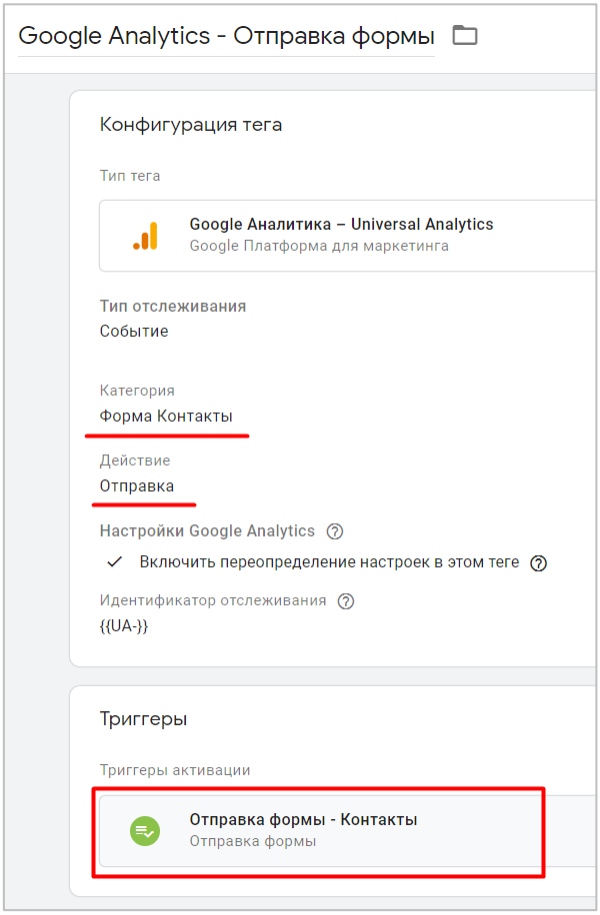


Рис. 590. Настройка тега

Такой способ отслеживания подойдет не всем сайтам, поскольку Google Tag Manager с помощью триггера **Отправка формы** определяет далеко не все отправки формы. Во многом это зависит от специфики проекта.

Form Submit на каждой странице

Иногда вы можете наблюдать следующее: в режиме предварительного просмотра GTM события **Form Submit** фиксируются, тег на отправку формы срабатывает, и данные в инструменты веб-аналитики передаются даже тогда, когда пользователь не заполнял форму. Или вы просто обновляете страницу и, когда страница загружается, видите событие **gtm.formSubmit**.

Как правило, такие события возникают из-за установленного Facebook Pixel (пикселя Facebook) при выполнении двух условий:

1. триггер **Отправка формы** создан в Google Tag Manager;
2. на сайте установлен Facebook Pixel;

Даже если страница не содержит никакой формы, вы все равно можете увидеть это событие:

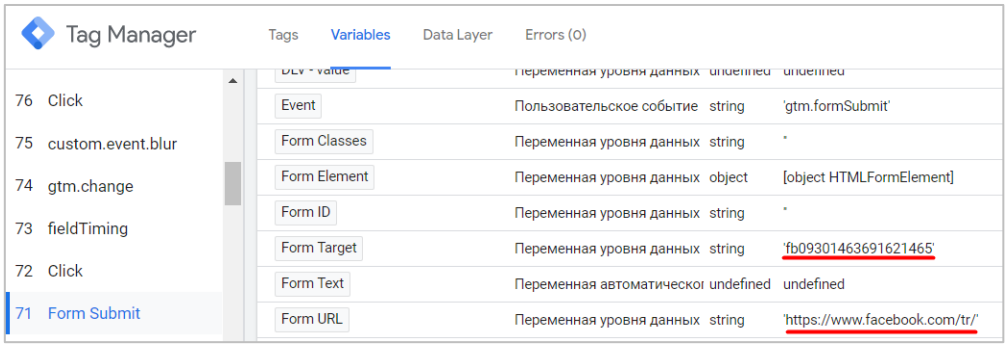


Рис. 591. Событие Form Submit от Facebook Pixel

Так Facebook Pixel отправляет данные на серверы Facebook (через другой домен для наибольшей безопасности) посредством отправки формы. Это особенность работы самого пикселя. А триггер **Отправка формы (gtm.formSubmit)** просто отображает эти представления в консоли предварительного просмотра Google Tag Manager, потому он предназначен для отображения событий отправки формы.

Причем событие может быть не одно, а иметь несколько Form Submit подряд. Активировав встроенные переменные для форм (Form ID, Form Element, Form Target и т.д.) и перейдя на вкладку **Variables** (см. рисунок выше), можно легко распознать такие события от Facebook. Если в Form URL или Form Target будет что-то, что напоминает или содержит **facebook, fb**, то это значит, чтобы событие было вызвано именно пикселем фейсбука.

Вы можете отличить обычную отправку формы от пиксельной в Facebook, посмотрев на уровень данных (вкладка **Data Layer**):

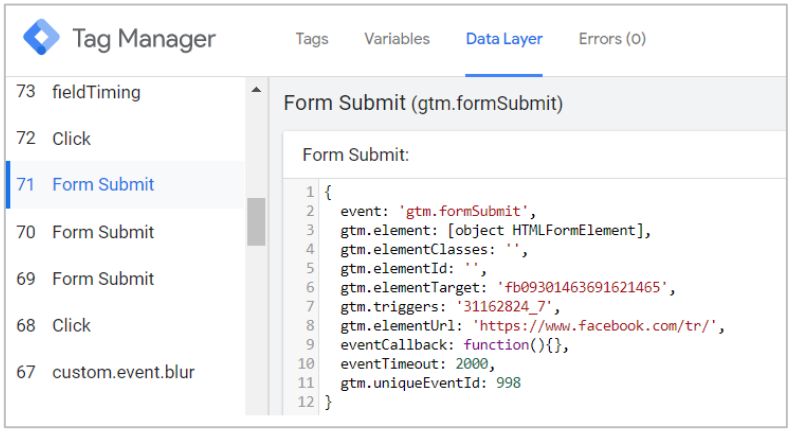


Рис. 592. Вкладка Data Layer – событие для Facebook

Переменная уровня данных **gtm.elementTarget** содержит признак Facebook - **fb093...**. Аналогично и с **gtm.elementUrl**, в которой фигурирует URL **https://www.facebook.com/tr/**. Для обычной отправки формы (стандартного триггера GTM) уровень данных будет отличаться:

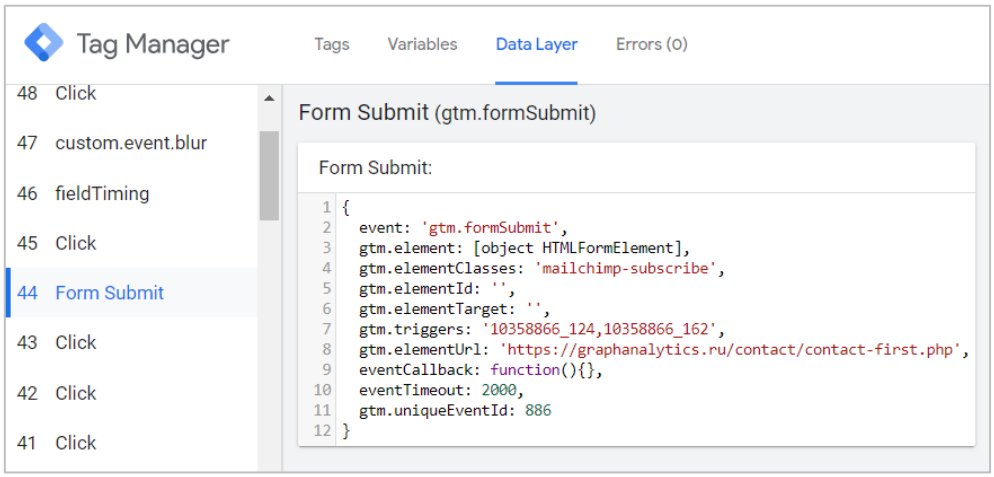


Рис. 593. Вкладка Data Layer – стандартное событие

Как видно из рисунка выше, «реальная» форма не содержит ничего, что связано с Facebook, а в переменной уровня данных **gtm.elementUrl** отображается ссылка на обработчик формы.

Что можно сделать в данном случае? Не так много. Стараться игнорировать в режиме отладки данные события. А чтобы быть на 100% уверенным, что при отправке формы не активируется тег и данные не отправляются в инструменты веб-аналитики (Google Analytics, Яндекс.Метрику) дважды, можно для триггера добавить фильтр **Form URL не содержит facebook.com/tr**:

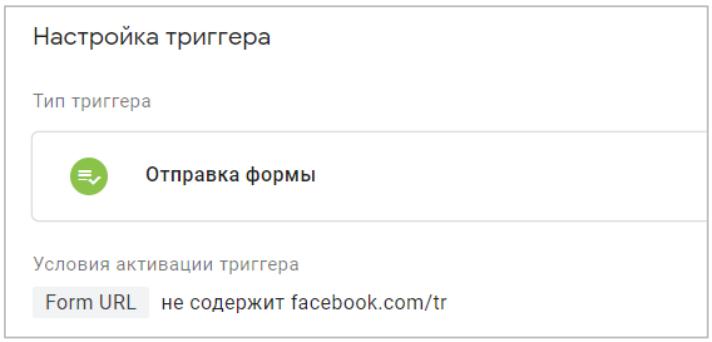


Рис. 594. Form URL не содержит facebook.com/tr

3. Отслеживание отправки формы с помощью триггера Доступность элемента

С помощью триггера **Доступность элемента** вы можете отслеживать определенные элементы на странице, которые появляются в видимой области экрана браузера пользователя. Например, когда посетитель сайта просматривает страницу и доходит до определенного блока контента. В этот момент вы можете активировать триггер и передать информацию о совершенном событии в счетчики аналитики.

Тот же самый метод может быть применен и к формам, когда после успешной отправки обращения появляется некоторое сообщение, например, «**Спасибо! Данные успешно отправлены»**:

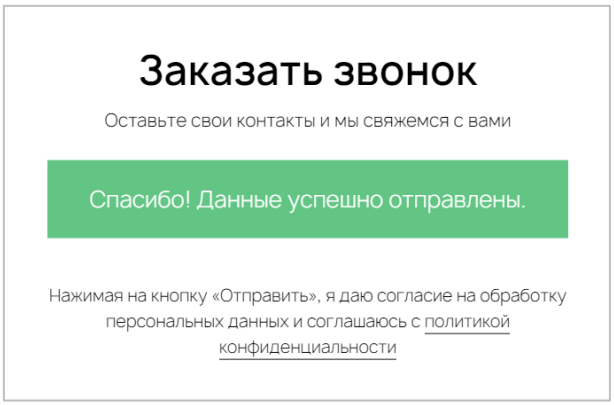


Рис. 595. Пример формы

Чтобы настроить отслеживание отправки с помощью данного способа, для начала требуется проверить, а появляется ли такое сообщение после успешной отправки формы. Для этого необходимо создать тестовое обращение.

Как только вы убедились в этом, не закрывая окна, проинспектируйте элемент, чтобы узнать его CSS-селектор. Для этого наведите курсор мыши на текст сообщения и нажмите правой кнопкой – **Просмотреть код**:

Рис. 596. Определение селектора элемента

После этого откроется консоль разработчика на вкладке **Elements** (браузер Google Chrome) с подсвеченным элементом. Из рисунка выше видно, что у данного элемента есть класс **js-successbox t-form__successbox t-text t-text_md**, который мы можем применить при настройке триггера в Google Tag Manager.

Примечание: элемент может иметь/не иметь идентификатор (#id) или класс (.class). В этом случае вы можете привязаться к его CSS-селектору. Подробно о селекторах было разобрано в соответствующей главе.

Вернемся в Google Tag Manager и создайте триггер типа **Доступность элемента**. В качестве метода выбора будем использовать **Селектор CSS**, поскольку у данного элемента нет идентификатора. Как вы уже знаете, в CSS перед каждым классом ставится точка (.), а пробелы между несколькими классами также заменяются на точки.

Первая настройка для моего примера будет выглядеть так:

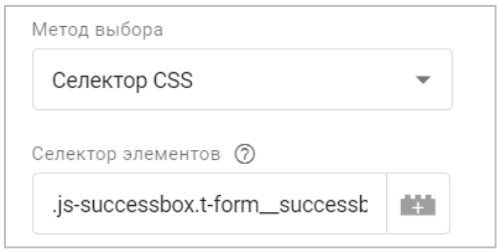


Рис. 597. Настройка триггера Доступность элемента

Правило запуска этого триггера оставим без изменений - **Один раз на страницу**. Настройку **Минимальный процент видимости** также не будем менять – 50%. Главное – это не забыть поставить галочку напротив **Регистрация изменений DOM**, поскольку изначально данного элемента нет в структуре DOM-дерева, и сообщение об успешной отправки формы появляется на экране только после ее отправки.

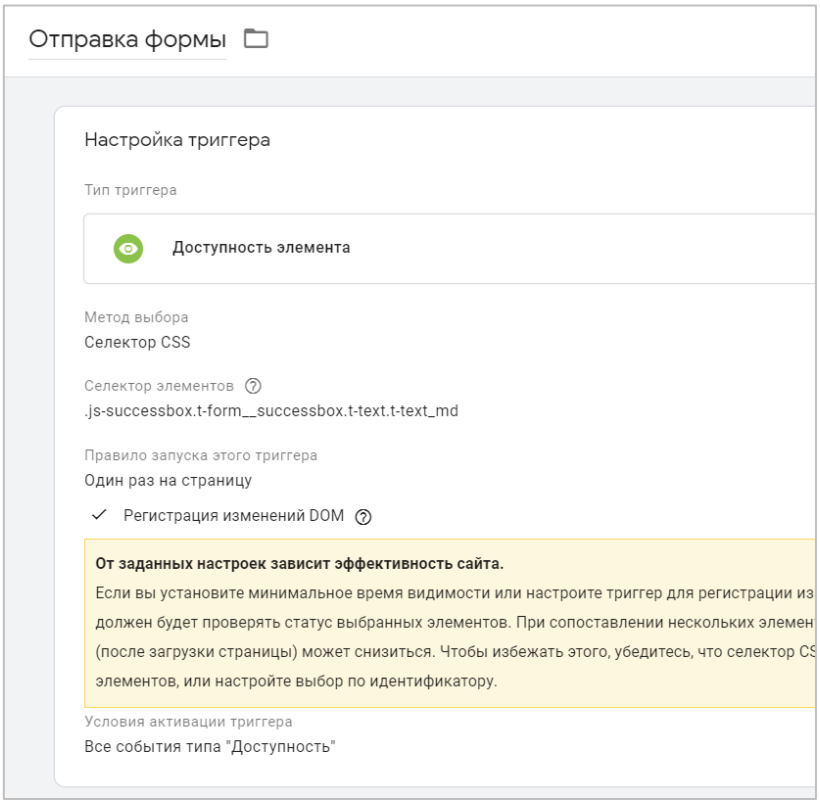


Рис. 598. Доступность элемента

Сохраним настройки триггера. Чтобы проверить срабатывание события после отправки формы воспользуемся режимом отладки GTM и повторно создайте тестовое обращение.

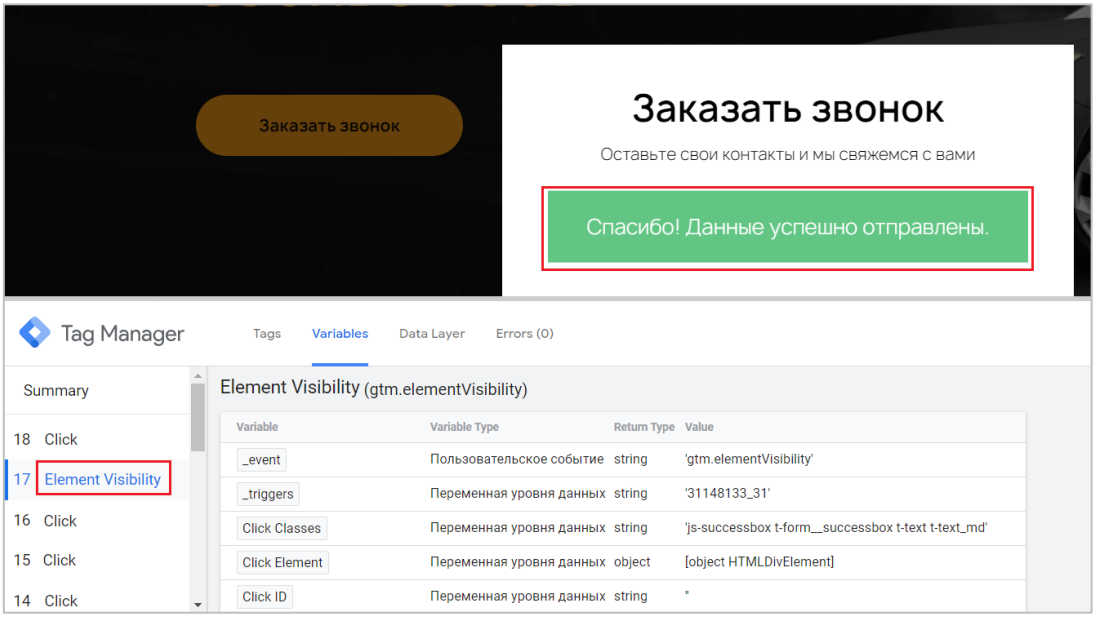


Рис. 599. Событие Element Visibility

Видим, что событие **Element Visibility (gtm.elementVisibility)** сработало после успешной отправки формы. Теперь вы можете использовать этот триггер в различных тегах для передачи информации в инструменты веб-аналитики (Яндекс.Метрику, Google Analytics, Facebook Pixel и др.). Например, как событие в Google Analytics:

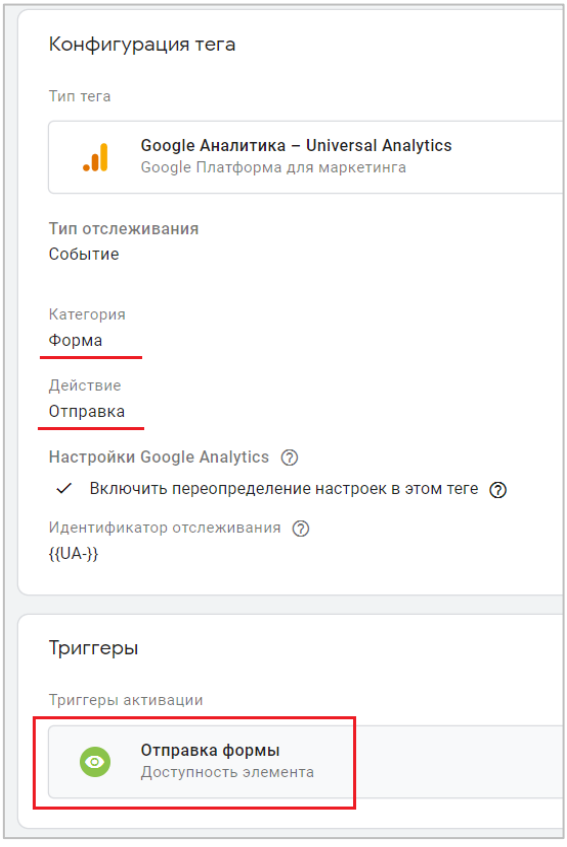


Рис. 600. Настройка тега

4. Отслеживание отправки формы с помощью триггера Пользовательское событие и уровня данных

Данный способ отслеживания отправки формы является не самым простым и требует от интернет-маркетолога коммуникации с разработчиком, поскольку для его реализации необходимо внедрение дополнительного кода на сайт.

Если стандартный триггер **Отправка формы** может не работать с вашей формой, то отслеживание отправки формы с помощью триггера **Пользовательское событие** и уровня данных 100% вызовет нужное событие (при условии, что настройка верна!). Вы можете попросить разработчика внедрить конструкцию **dataLayer.push ()** в функцию обратного вызова, которая вызывается при успешной отправке формы.

Пример кода, который вы можете отправить программисту, выглядит так:

```
window.dataLayer = window.dataLayer || [];
  window.dataLayer.push ({
   'event': 'formSuccess',
   'formName': 'Название формы',
});
```

, где после **event** может и вовсе не быть дополнительных переменных. В моем примере есть **formName**, в которую вы можете попросить разработчика добавить название формы, при условии, что на сайте их несколько, и они имеют разное назначение.

Таким образом, используя переменную уровня данных, вы сможете передавать информацию в аналитику не только об успешной отправке формы, но и ее название, чтобы всегда знать, какую конкретно форму на сайте отправил пользователь, а также ряд других дополнительных параметров. Потом статистику по каждой заполненной форме можно будет сравнить между собой в отчетах Google Analytics и Яндекс.Метрики. Кроме formName можно передавать и другие данные.

Здесь уже зависит от вашего желания и конкретной задачи. Например, на моем сайте (osipenkov.ru) есть 5 различных форм:

1. бесплатный анализ проекта;
2. запись на вебинар;
3. скачать бесплатные книги;
4. подписка на рассылку новостей;
5. связаться по форме «Контакты»;

Я могу использовать дополнительную переменную для передачи этой информации в уровне данных. У вас этого может не быть. Перед постановкой задания разработчику вам нужно выписать на листке бумаге все возможные варианты параметры передачи данных, которые вы хотите, чтобы отправлялись в dataLayer при успешной отправке формы. Я в примере привел минимальный фрагмент кода, который удовлетворяет условию отслеживания с помощью триггера **Пользовательское событие** и уровня данных. Можно использовать и более упрощенную конструкцию, без formName:

```
window.dataLayer = window.dataLayer || [];
window.dataLayer.push ({'event': 'formSuccess'});
```

Последовательность отслеживания следующая:

- определяетесь с именем события, которое в дальнейшем вы будете использовать в триггере GTM типа **Пользовательское событие** (в примере выше – это **formSuccess**);

- определяетесь с количеством дополнительных данных/переменных, которые вы хотите передавать в dataLayer;

- записываете итоговый код для разработчика (за аналог берете код в примере выше);

- отправляете разработчику текст такого вида: *просьба внедрить конструкцию dataLayer.push () в функцию обратного вызова (код прилагаю к ТЗ), которая вызывается при успешной отправке формы со следующими данными* (ниже перечисляете все параметры, которые необходимо передать вместе с формой, чтобы программист вас понял);

Ожидайте, пока разработчик выполнит ваше ТЗ и добавит код на все отслеживаемые формы. На это требуется время. После внедрения вы должны протестировать сформированный уровень данных. Для этого в Google Tag Manager также воспользуйтесь режимом предварительного просмотра., перейдите на сайт и отправьте тестовую форму. В результате одним из событий должно идти с вашим названием:

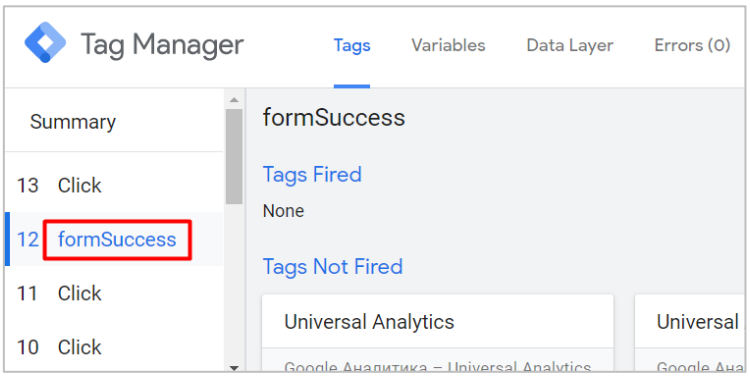


Рис. 601. Событие formSuccess

Посмотрите на вкладке **Data Layer** вашего события все ли данные правильно переданы в уровень данных:

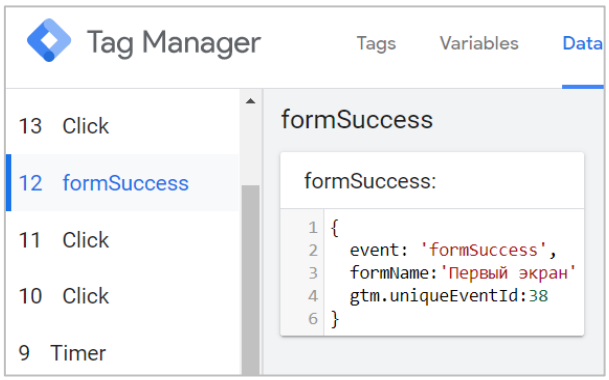


Рис. 602. Уровень данных события

Проверить следует также и не успешную отправку формы. В этом случае оставьте некоторые поля формы пустыми и попробуйте оправить форму вновь. Если вы и в этом случае увидите событие, то попросите разработчика внести изменения. Вполне вероятно, что он сформировал уровень данных на клик по кнопке или же не учел валидацию формы.

Теперь в Google Tag Manager создайте триггер типа **Пользовательское событие** с именем, которое вы назначили. В моем примере это **formSuccess**:

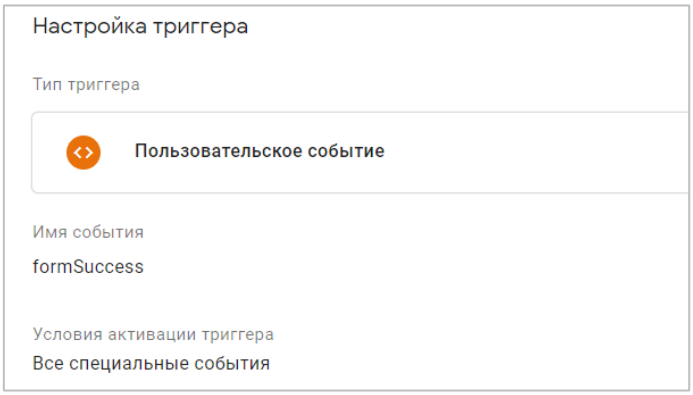


Рис. 603. Триггер типа Пользовательское событие

Поскольку в нашем примере в уровне данных еще есть дополнительные переменные, которые мы передаем (речь о **formName**), то можно создать пользовательскую переменную типа **Переменная уровня данных** для извлечения значения из нее:

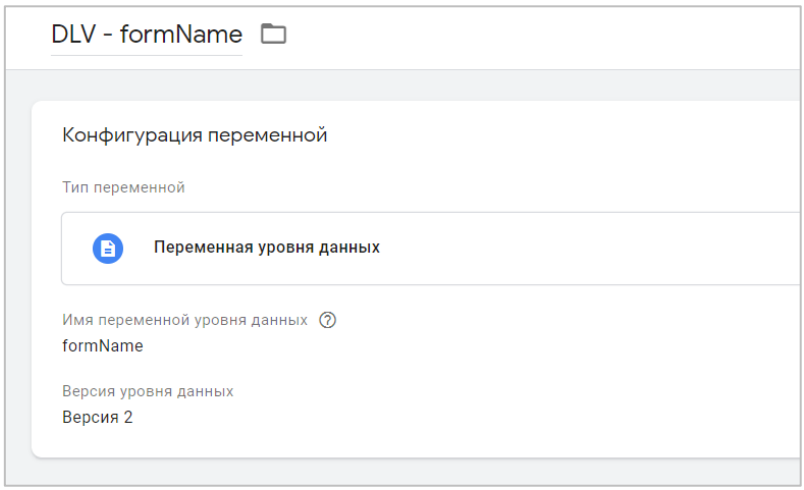


Рис. 604. Переменная уровня данных

Сохраните настройки. На заключительном шаге создайте тег **Google Аналитика – Universal Analytics** с типом отслеживания **Событие** с соответствующими настройками для передачи информации об успешной отправки формы в Google Analytics. Пример:

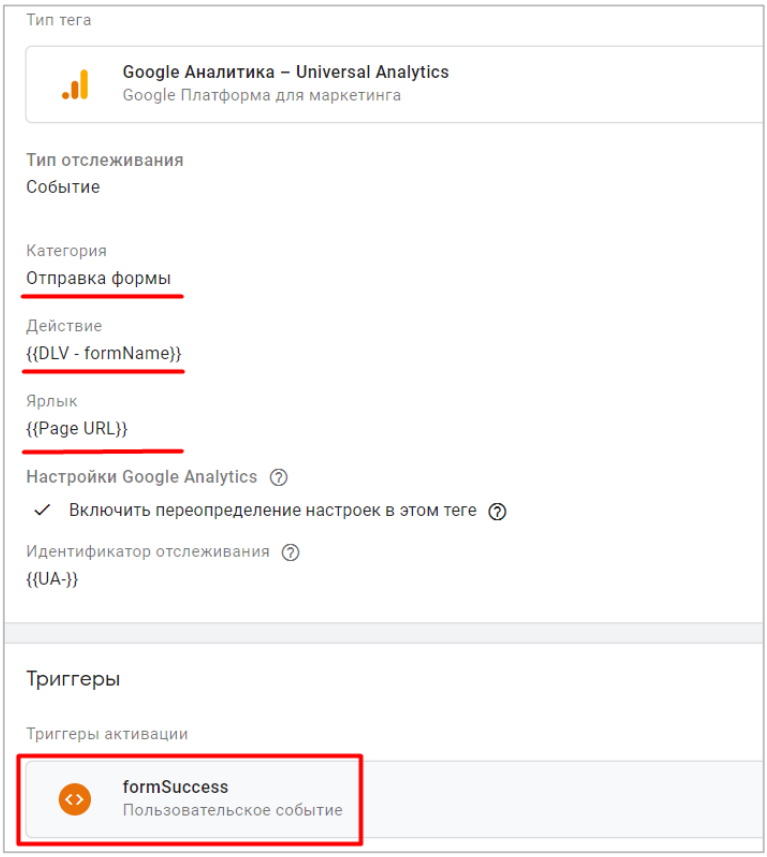


Рис. 605. Настройка тега

В **Категории** передайте произвольное название, в **Действии** я укажите пользовательскую переменную **DLV – formName**, которая передает название формы, а в **Ярлыке** события можно передать URL-адрес страницы, на которой произошло событие отправки формы. Конфигурация тега может быть иной. Главное, чтобы при отслеживании отправки формы с помощью уровня данных и пользовательского события наш триггер был добавлен в качестве условия активации тега, то есть триггер активации – **Пользовательское событие**, которое было создано на предыдущем шаге.

5. Отслеживание отправки формы с помощью прослушивателя автоматических событий

Прослушивание автоматических событий — это очень полезная функция в JavaScript, которая позволяет отслеживать определенные взаимодействия пользователя на страницах сайта. При срабатывании запускается событие, которому назначается обработчик (функция, которая отвечает за реакцию на действие пользователя), а само событие помещается в уровень данных (dataLayer). Это событие в дальнейшем можно использовать в качестве триггера активации в Google Tag Manager. Кроме этого, в момент срабатывания события, уровень данных может содержать дополнительную информацию о самом объекте. Помните предыдущий пример, где в dataLayer вместе с отправкой формы передавалась информация по и названию формы?

По умолчанию диспетчер тегов Google предоставляет встроенный прослушиватель (триггер **Отправка формы**). Но в большинстве настроек он не подходит или не может быть использован. Тогда на помощь приходит прослушиватель автоматических событий. Их еще называют *прослушивателями пользовательских* событий. Подробнее эту тему затронем в последней главе этого руководства. Здесь же я хочу разобрать один из самых распространенных примеров такого отслеживания.

Если вы используете готовый CMS-движок (OpenCart, WordPress, 1С-Битрикс), то, как правило, для них существует ряд готовых решений (модулей, плагинов), которые вы можете приобрести или скачать бесплатно. Лично я на своем сайте (osipenkov.ru, WordPress) для отправки форм использую плагин **Contact Form 7**. Разработчики этого решения заранее предусмотрели возможность отслеживания отправки форм с помощью прослушивателя автоматических событий, описав подробную настройку в своей документации для Google Analytics (см. приложение). Ниже я опишу способ отслеживания отправки форм **Contact Form 7** с помощью Google Tag Manager.

WordPress - самый популярный CMS-движок в мире. Благодаря тому, что он бесплатный, быстро устанавливается, имеет одно из самых больших сообществ разработчиков, понятную и простую административную панель, постоянно обновляется, для него можно скачать или купить шаблоны на любой вкус и цвет, он является №1 среди всех CMS. Сейчас каждый третий сайт планеты работает на WordPress. К слову, мой сайт osipenkov.ru тоже сделан на WP.

Одной из главных причин почему WordPress так популярен в последние время, является наличие огромного количества плагинов для любых задач. Есть менее известные решения, а есть определенный список плагинов (топ), которые устанавливают практически все владельцы сайтов. Среди них бесплатный плагин создания контактных форм Contact Form 7 (более 5 миллионов установок).

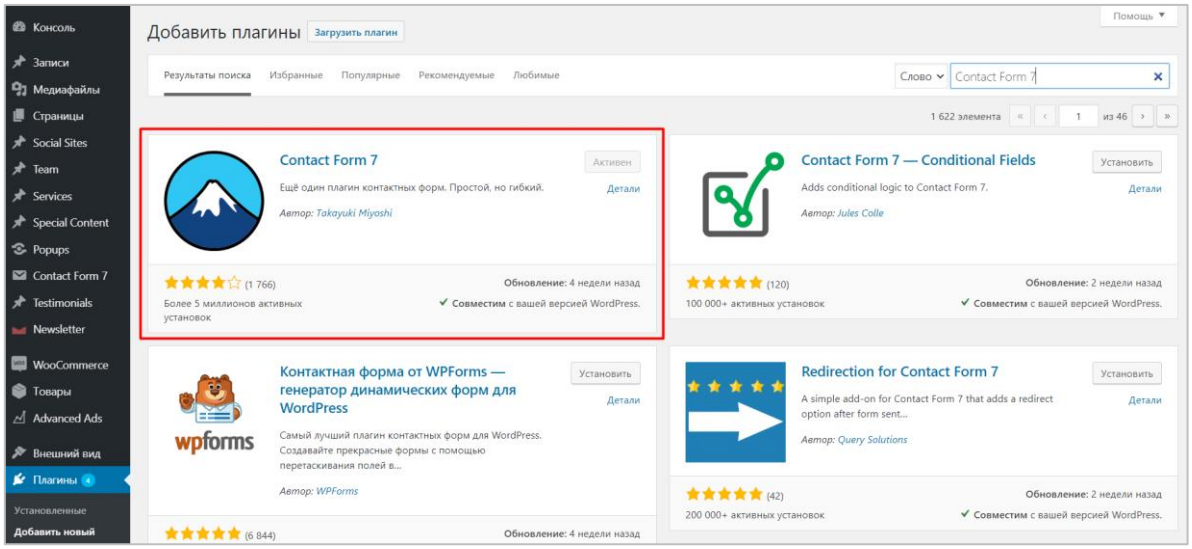


Рис. 606. Плагин Contact Form 7 для WordPress

Я целенаправленно опускаю процесс установки и настройки данного плагина в WordPress в этом руководстве по понятным причинам (читайте официальную документацию по установке) и сразу перейду к настройке отслеживания отправки форм с помощью GTM.

Для того, чтобы настроить отслеживание с помощью Google Tag Manager, необходимо:

- создать 2 тега;
- создать 1 триггер;
- создать несколько переменных.

Давайте разберемся подробнее.

Последовательность действий

Для начала создайте тег типа **Пользовательский HTML** тег со следующим кодом:

```
<script>
document.addEventListener('wpcf7mailsent', function( event ) {
dataLayer.push({
'event' : 'formComplete',
'formId' : event.detail.contactFormId
});
}, false );
</script>
```

Мы хотим отслеживать событие отправки формы в том случае, когда форма была успешно отправлена. Contact Form 7 запускает пользовательское событие **wpcf7mailsent**, когда успешно отправляет форму и почту адресату, поэтому создается обработчик с функций, внутри которой с помощью уровня данных (dataLayer) методом **.push ()** отправляется нужное нам событие **formComplete** с переменной **formId**, в которой передается идентификатор контактной формы.

Список пользовательских событий, которые предопределены (заложены) в плагин Contact Form 7:

- **wpcf7invalid** - срабатывает, когда форма успешно была отправлена на сервер, но почта не была отправлена, потому что были поля с недопустимым вводом;
- **wpcf7spam** - срабатывает, когда форма успешно была отправлена на сервер, но почта не была отправлена, поскольку обнаружена возможная активность спама;
- **wpcf7mailsent** - срабатывает, когда форма успешно была отправлена на сервер и почта отправлена адресату;
- **wpcf7mailfailed** - срабатывает, когда форма успешно была отправлена на сервер, но отправить почту не удалось. Это может быть вследствии того, что на хостинге не работает почтовый сервер;
- **wpcf7submit** - срабатывает, когда форма успешно была отправлена на сервер, независимо от других результатов. Нажали кнопку "Отправить" - сработало это действие.

Именно **wpcf7mailsent**, согласно справке плагина, отвечает за успешную отправку формы. Подробнее читайте в официальной документации.

Свойства объекта события **event**, которые вы можете использовать в своем обработчике:

- **detail.contactFormId** - идентификатор контактной формы;
- **detail.pluginVersion** - версия плагина Contact Form 7;
- **detail.contactFormLocale** - код локали контактной формы;
- **detail.unitTag** - блок-тег контактной формы;
- **detail.containerPostId** - идентификатор поста, в котором находится контактная форма.

Нам достаточно переменной **detail.contactFormId**. Например, вот так выглядят идентификаторы моих форм:

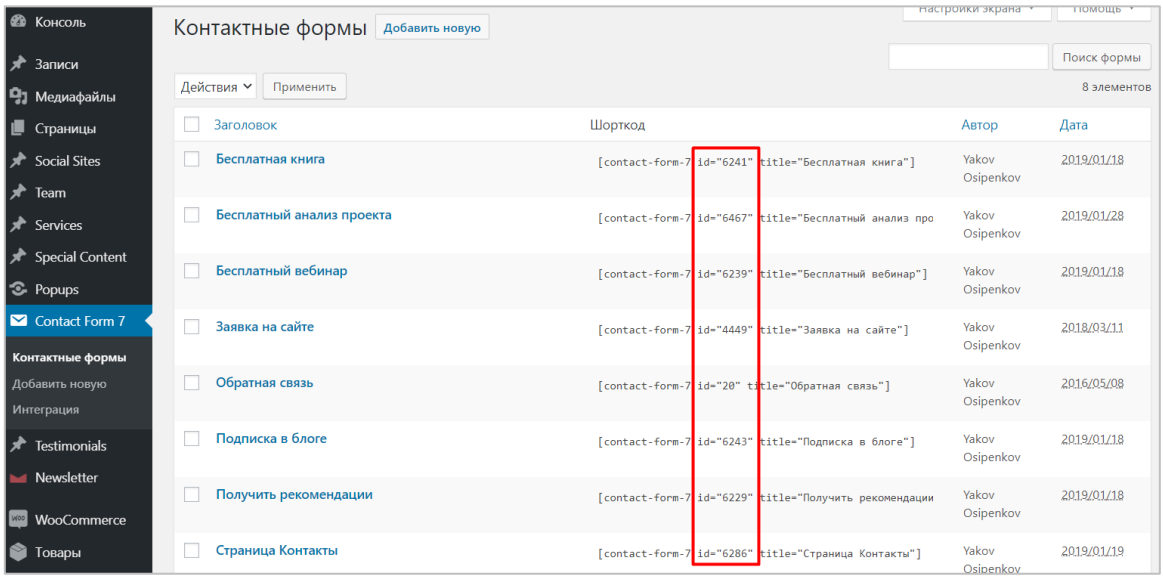


Рис. 607. Идентификаторы форм моего блога (osipenkov.ru)

В GTM это будет выглядеть так:

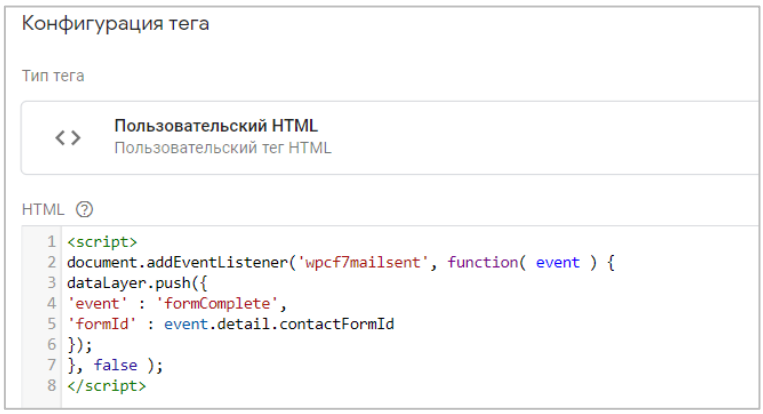


Рис. 608. Прослушиватель автоматического события wpc7mailsent

Триггер активации - **All Pages (Все страницы)**. Сохраняем изменения. На следующем этапе нам необходимо создать триггер типа **Пользовательское событие** с именем **formComplete**:

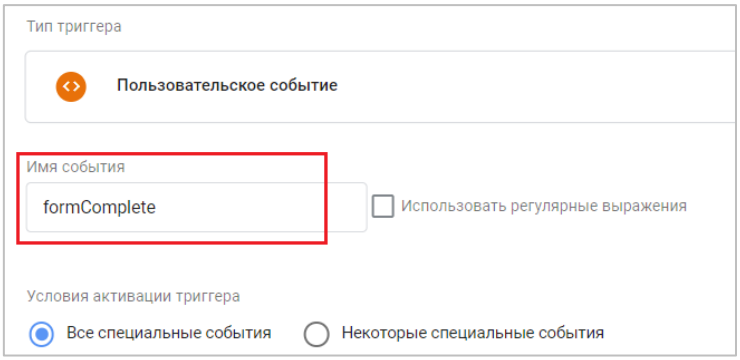


Рис. 609. Триггер Пользовательское событие

Теперь создайте пользовательскую переменную типа **Переменная уровня данных** с именем **formId**, чтобы получить значение идентификатора отправленной формы:

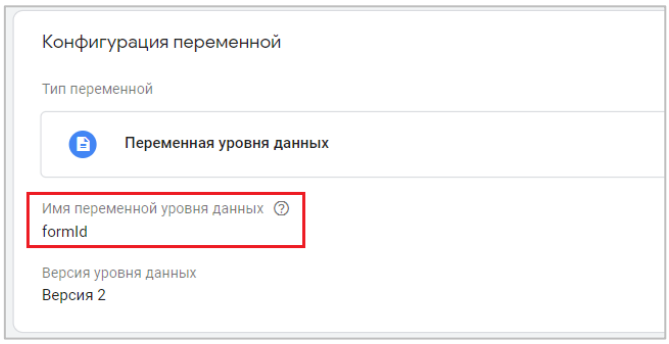


Рис. 610. Переменная уровня данных

Все, что осталось сделать — это создать тег **Google Аналитика - Universal Analytics** с типом **Событие**, в котором следует задать **Категорию, Действие** и **Ярлык** (по желанию) события:

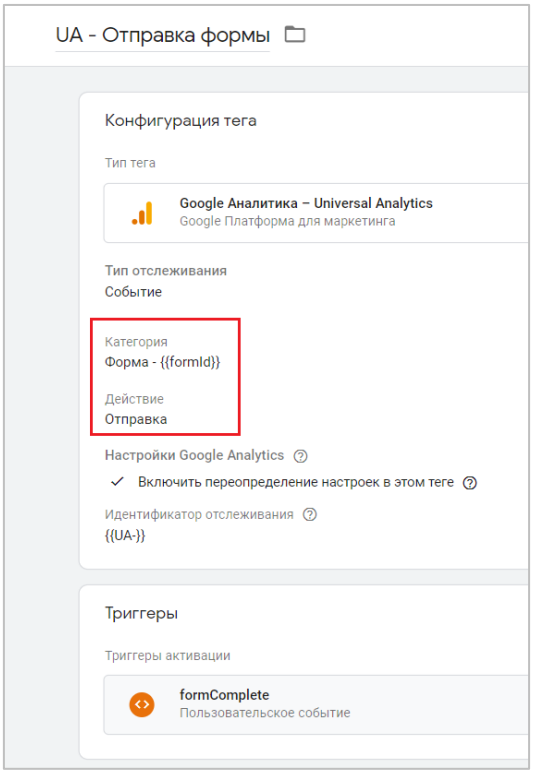


Рис. 611. Настройка тега

Триггер активации - наше пользовательское событие **formComplete**. Я в примере передаю значение идентификатора формы (formId) в Категории события, используя конструкцию **Форма - {{formId}}**. Вы можете передать как-то иначе.

Сохраняем изменения. В режиме отладки GTM можно проверить корректность настройки отслеживания отправки формы Contact Form 7. После отправки формы сработает наше событие **formComplete**:

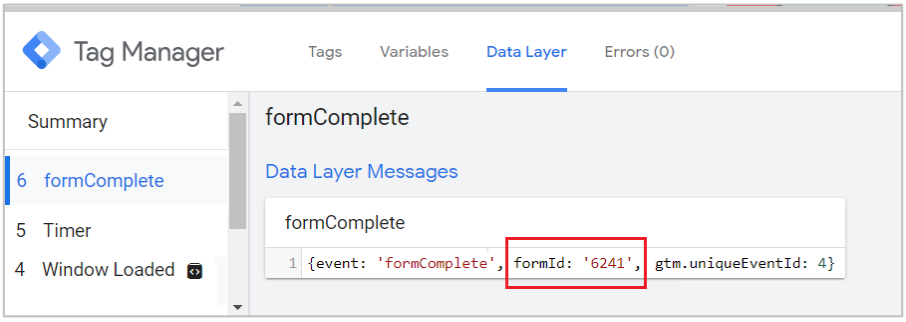


Рис. 612. Данные в режиме предварительного просмотра

Открыв вкладку **Data Layer**, увидим, как передалось событие, а вместе с ним и значение идентификатора формы **6241** в переменной **formId**.

В режиме реального времени Google Analytics:

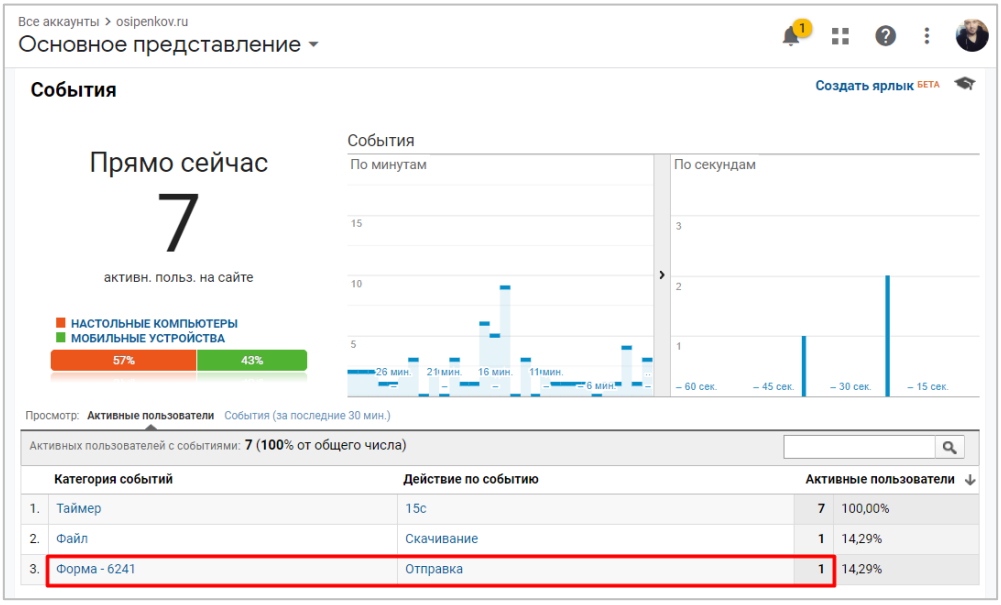


Рис. 613. Отчет В режиме реального времени

Поскольку мы настроили отслеживание отправки форм как событие, то через некоторое время все данные по событиям появятся в отчете **Поведение — События — Лучшие события.**

Примечание: если вы хотите, чтобы в Google Analytics отправлялась информация не об идентификаторе формы, а ее название, то вы можете переопределить итоговые значения с помощью пользовательской переменной **Таблица поиска**. И получить такие данные:

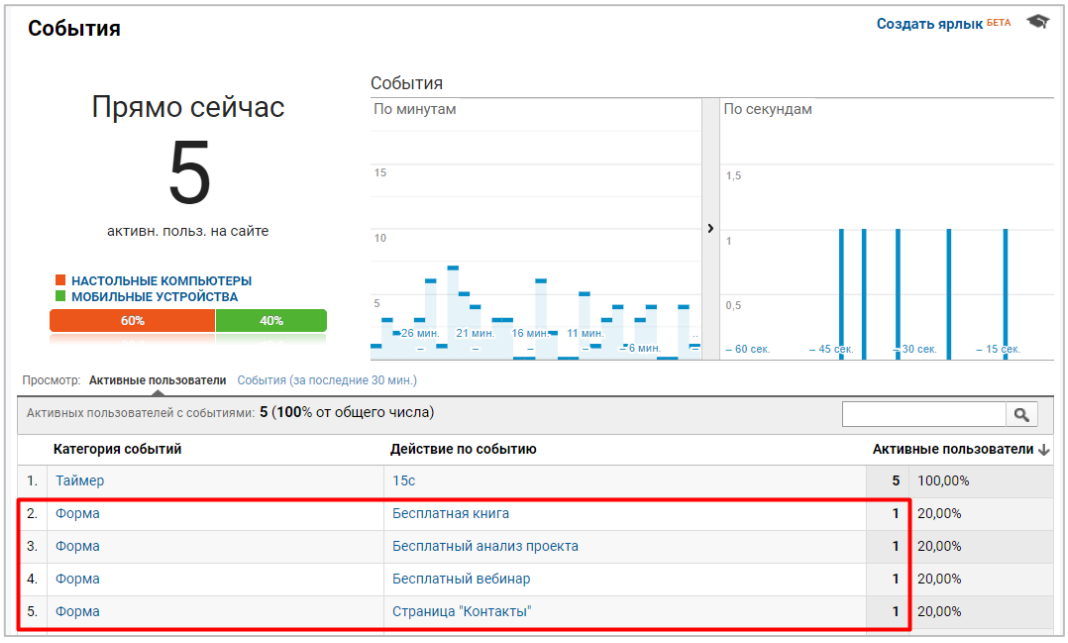


Рис. 614. Удобочитаемый вид форм (с помощью переменной Таблица поиска)

Таким образом, если вы используете готовые решения от сторонних разработчиков, первым делом попробуйте найти официальную документацию. Как правило, в ней написаны все возможные способы отслеживания. В моем примере я использовал самый популярный плагин в мире для WordPress. И, конечно же, его разработчики постарались и написали подробную инструкцию по отслеживанию отправки форм с помощью прослушивателя автоматических событий.

6. Отслеживание отправки формы с помощью универсального кода для форм на AJAX

Если у вас не получается настроить отслеживание с помощью стандартного триггера **Отправка формы**, а после отправки формы вас не перенаправляет на другую страницу, и сама форма просто обновляется, без перезагрузки страницы, то, скорее всего, на сайте используется технология AJAX.

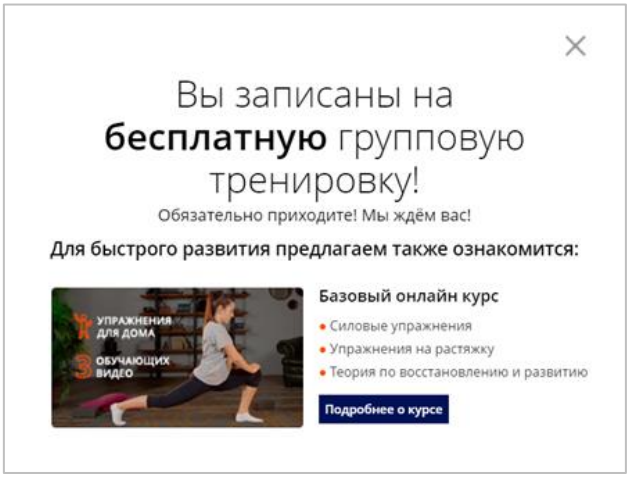


Рис. 615. Пример формы

AJAX (аббревиатура от Asynchronous JavaScript and XML) – это технология взаимодействия с сервером без перезагрузки страницы. Поскольку не требуется каждый раз обновлять страницу целиком, повышается скорость работы с сайтом и удобство его использования (см. приложение).

Компания **Bounteous (ex LunaMetrics)** в своем блоге в 2015 года опубликовала бесплатный прослушиватель AJAX событий для диспетчера тегов Google (см. приложение), который прекрасно работает и сейчас для большинства сайтов на этой технологии и который передает информацию на уровень данных.

Все, что требуется сделать, это скопировать код с их страницы и вставить его в тег типа **Пользовательский HTML** с условием активации триггера на всех страницах сайта (All Pages).

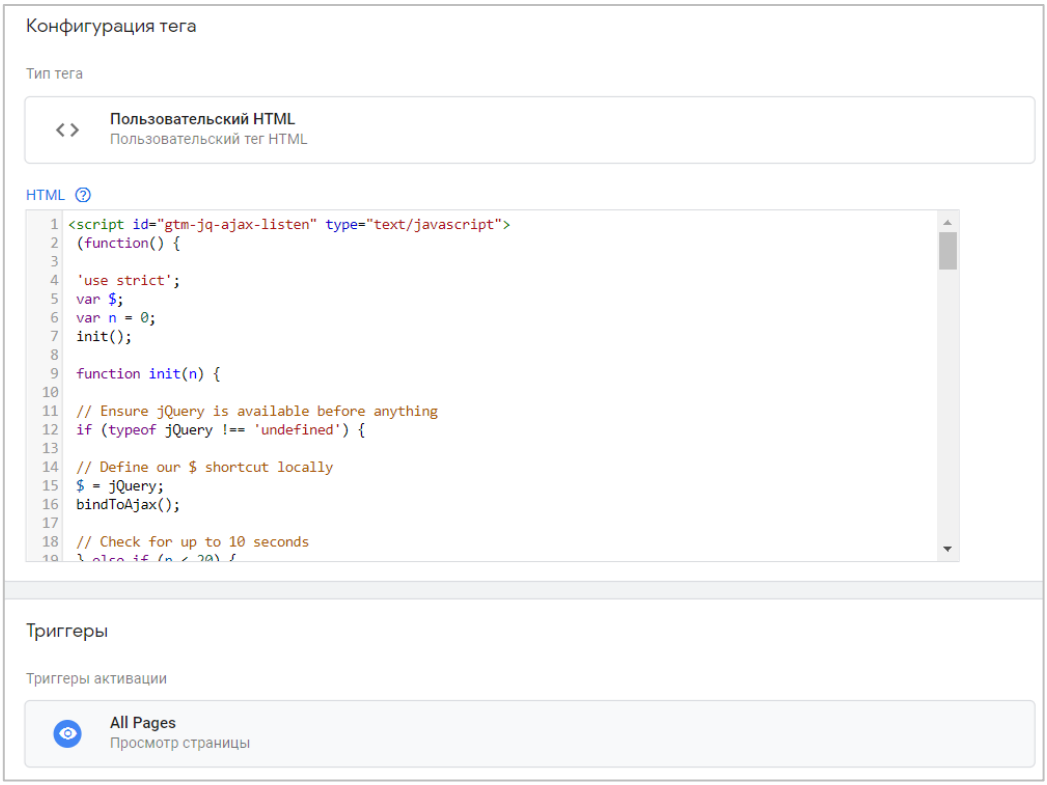


Рис. 616. Универсальный код для AJAX форм (Bounteous)

Сохраните изменения. Тег будет прослушивать все **$.ajax** запросы, включая **$.post, $.getJSON, $.getScript** и другие. Чтобы проверить корректность выполнения, в Google Tag Manager перейдите в режим предварительного просмотра и отправьте тестовую заявку.

Если в консоли появилось событие **ajaxComplete**, то значит форма использует AJAX. Если нет, значит этот код вам не подойдет, и вам необходимо использовать любой другой из представленных в этой главе способов отслеживания отправки формы.

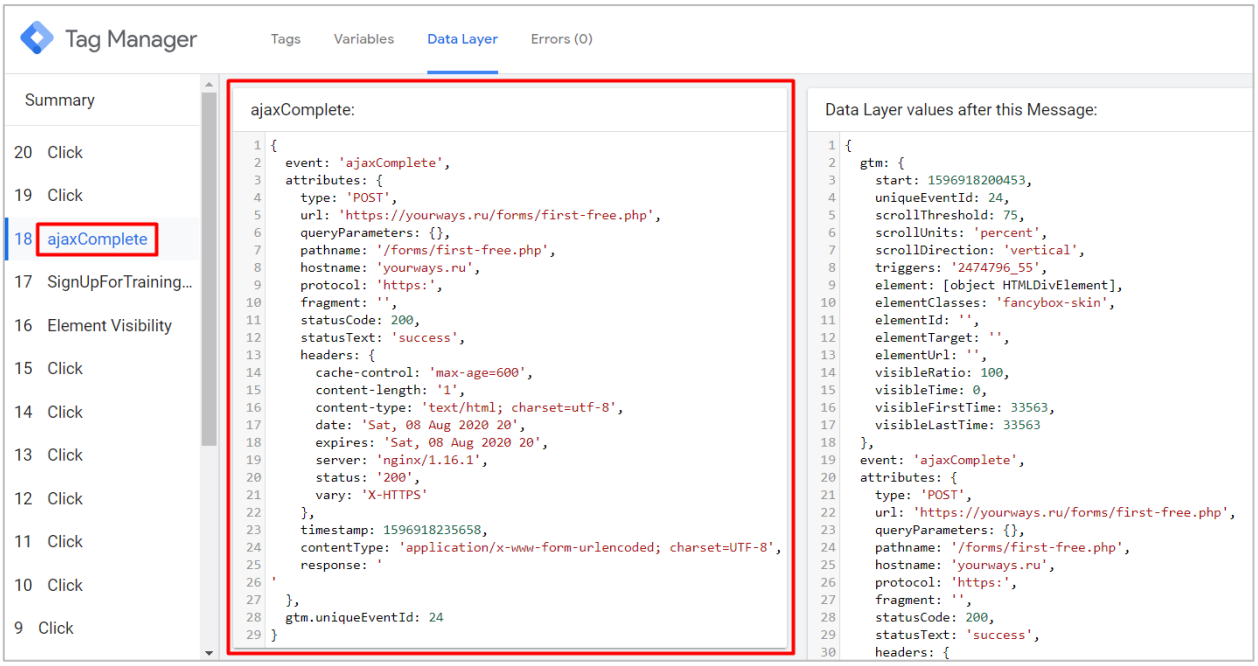


Рис. 617. Уровень данных события ajaxComplete

Выбрав событие ajaxComplete, перейдите на вкладку **Data Layer**. Здесь отображаются данные, которые были переданы на уровень данных после успешной отправки формы. Каждую строку в dataLayer можно использовать в качестве переменной уровня данных в GTM. Но перед этим следует определить, какая из представленной информации свидетельствует об успешной отправке формы. Чаще всего в этом коде используют значения из двух переменных:

1. **statusText;**
2. **response.**

К сожалению, в данном примере никакой информации в **response** нет, поэтому я буду использовать значение **success** из переменной **statusText**. Но если бы она была, вы могли бы использовать сообщение, которое отобразилось в **message** в качестве триггера активации. Пример:

```
    contentType: 'application/x-www-form-urlencoded; charset=UTF-8',
    response: {
      success: true,
      message: 'Подписка на рассылку оформлена. Проверьте почту для подтверждения подписки.'
    }
  },
```

Рис. 618. Текстовое сообщение в message

Возвращаемся к нашему примеру. Чтобы настроить триггер, нам сначала следует создать пользовательскую переменную типа **Переменная уровня данных**. Чтобы правильно написать имя переменной уровня данных, перейдите в консоль разработчика браузера на вкладку Console, введите команду **dataLayer** и нажмите **Enter**. Сделать это нужно сразу после отправки тестовой заявки, пока уровень данных заполнен событиями.

Рис. 619. Команда dataLayer в консоли разработчика

Ниже вы увидите перечень событий в хронологическом порядке, точно такой же, какой представлен в режиме отладки GTM. Найдите событие **ajaxComplete** и раскройте его на всех уровнях с помощью стрелочки:

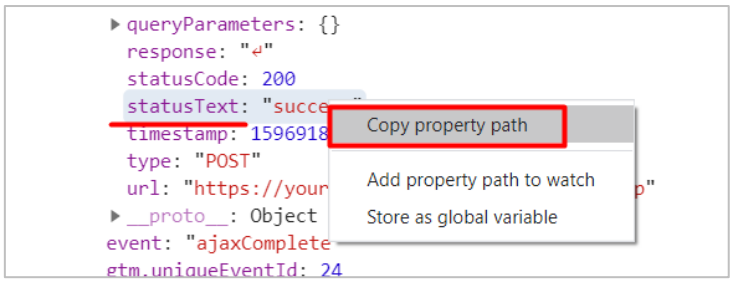


Рис. 620. Копирование пути к переменной

Найдите ту переменную, которую хотите использовать в качестве переменной уровня данных. Как я написал ранее, я буду использовать **statusText** со значением **success**. Нажмите на переменную правой кнопкой мыши и выберите пункт меню **Copy property path**. Вставьте полученное значение в переменную GTM и удалите все лишнее перед словом **attributes**, чтобы получилось так:

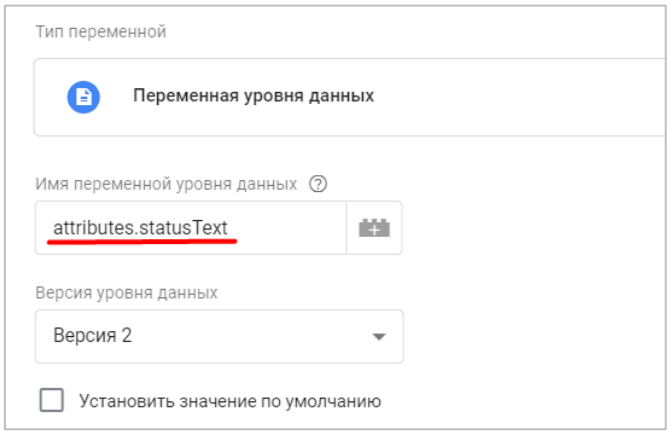


Рис. 621. Переменная уровня данных

Каждый уровень пути должен быть разделен точкой (точечная нотация). Если вы будете использовать **message**, то имя переменной уровня данных в диспетчере тегов будет **attributes.response.message**, поскольку **message** находится внутри **response**, а response внутри **attributes**.

После этого создайте триггер типа **Пользовательское событие** с такими настройками:

- Имя события – **ajaxComplete**;

- Условие активации триггера – **attributes.statusText содержит success**.

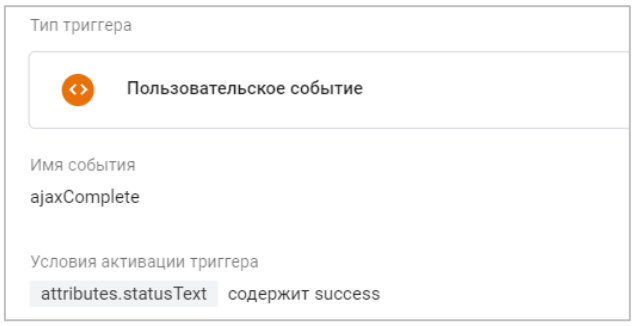


Рис. 622. Триггер Пользовательское событие с условием активации

Триггер будет срабатывать в случае, если в отправленной форме переменная уровня данных примет значение, содержащее **success**. Для переменной **attributes.response.message** триггер тот же самый, только условие активации будет примерно следующим:

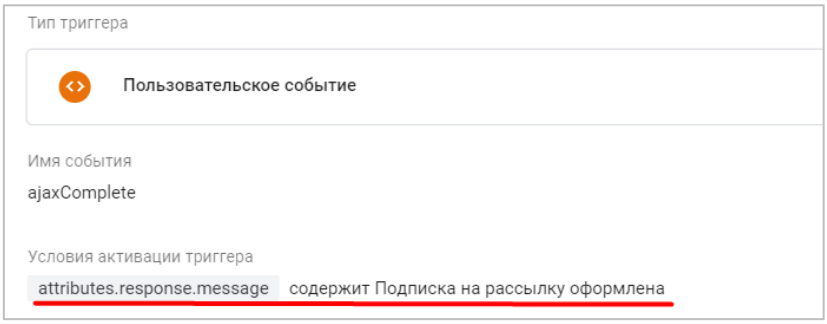


Рис. 623. Пример триггера с attributes.response.message

Условие **содержит** будет для каждого свое, в зависимости от того текста, который отображается на вашем сайте после успешной отправки формы и определяется данным прослушивателем в переменной **message**.

Сохраните триггер. Добавьте тег **Google Аналитика – Universal Analytics** с типом отслеживания – **Событие**, в котором задайте собственные настройки и используйте триггер активации ajaxComplete, созданный на предыдущем шаге:

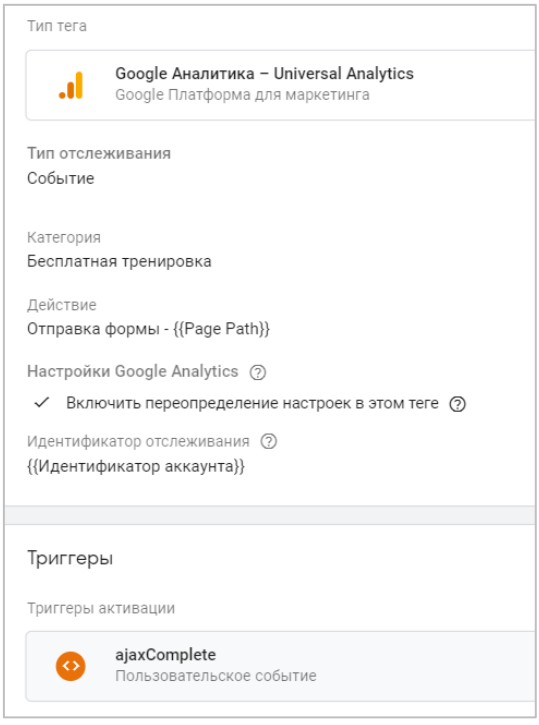


Рис. 624. Настройка тега

Поскольку мы настроили отслеживание отправки форм как событие, то через некоторое время все данные по событиям появятся в отчете **Поведение — События — Лучшие события.**

7. Отслеживание отправки формы с помощью DOM Scraping

Технология получения веб-данных путем извлечения их со страниц сайтов по-английски называется **DOM Scraping (Web scraping)**. Этим мы и будем заниматься.

В таком способе извлечения данных со страницы сайт есть один существенный недостаток - все, что вы будете делать, завязано на объектной модели документа (DOM). И при изменении каких-либо классов, идентификаторов отслеживаемых элементов на сайте есть вероятность потерять текущие настройки. Именно поэтому лучше выполнять отслеживание отправки формы или любых других нестандартных событий на сайте с помощью функций и привлечения разработчика, а не извлекая данные с помощью DOM-скрапинга.

Но такой метод отслеживания тоже существует и давайте разберем его. Например, на сайте после отправки заявки страница перезагрузилась, отображаемый контент поменялся, но URL-страницы не изменился, то есть произошло перенаправление на ту же самую страницу. Такое бывает, но встречается на практике реже (обычно наоборот – страница не перезагружается, а URL меняется). В этом случае нам как раз поможет встроенная переменная **Элемент DOM.**

Для демонстрации такого способа отслеживания я создал форму с двумя полями: Имя и E-mail.

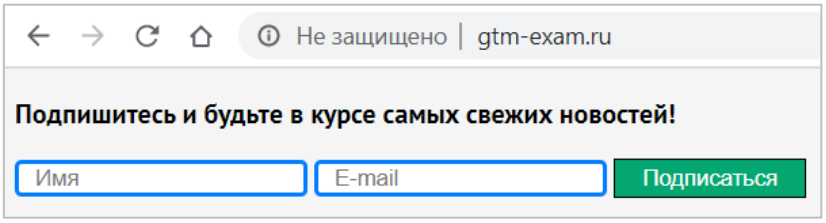


Рис. 625. Пример формы

После заполнения полей пользователь нажимает кнопку **Подписаться**, происходит перезагрузка страницы, URL-адрес остается прежним, а контент изменяется и появляется сообщение об успешной подписке:

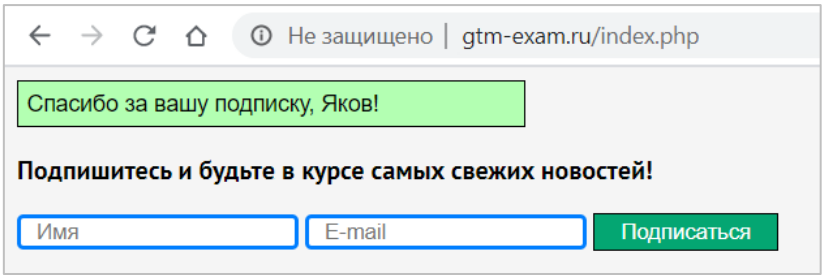


Рис. 626. URL не изменился, а содержимое поменялось

Для отслеживания отправки такой формы вы можете использовать пользовательскую переменную типа **Элемент DOM** и извлечь значение элемента с текстом **Спасибо за вашу подписку...**

Чтобы настроить переменную, необходимо сначала проинспектировать данный элемент и узнать его селектор. Откройте консоль разработчика, в которой будет отображен наш подсвеченный элемент в структуре DOM-дерева, нажмите на него правой кнопке и выберите **Copy – Copy selector**:

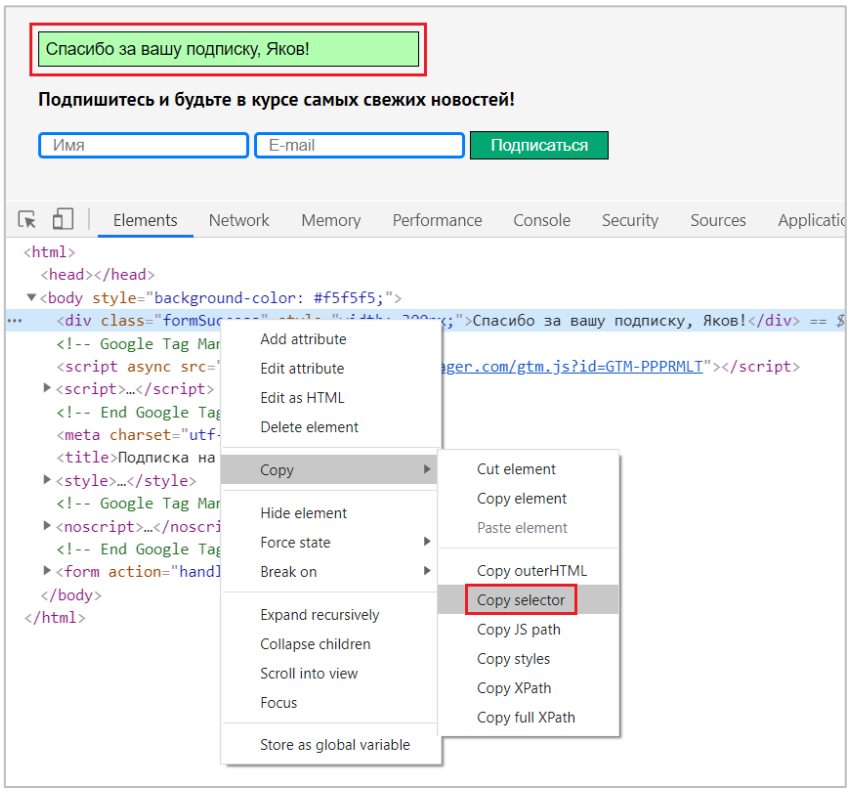


Рис. 627. Копирование селектора

В Google Tag Manager создайте пользовательскую переменную типа **Элемент DOM**, в которой будут следующие настройки:

- Метод выбора – **Селектор CSS**;
- Селектор элементов – **скопированный селектор**.

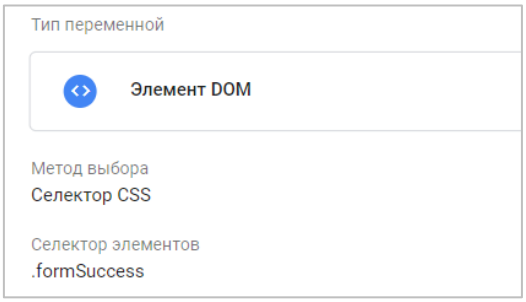


Рис. 628. Переменная Элемент DOM

Отправив тестовое обращение, поверьте корректность извлечения данных с помощью режима предварительного просмотра. Если вы указали верный селектор CSS и выбрали событие **Container Loaded**, то напротив имени вашей переменной отобразится текст сообщения:

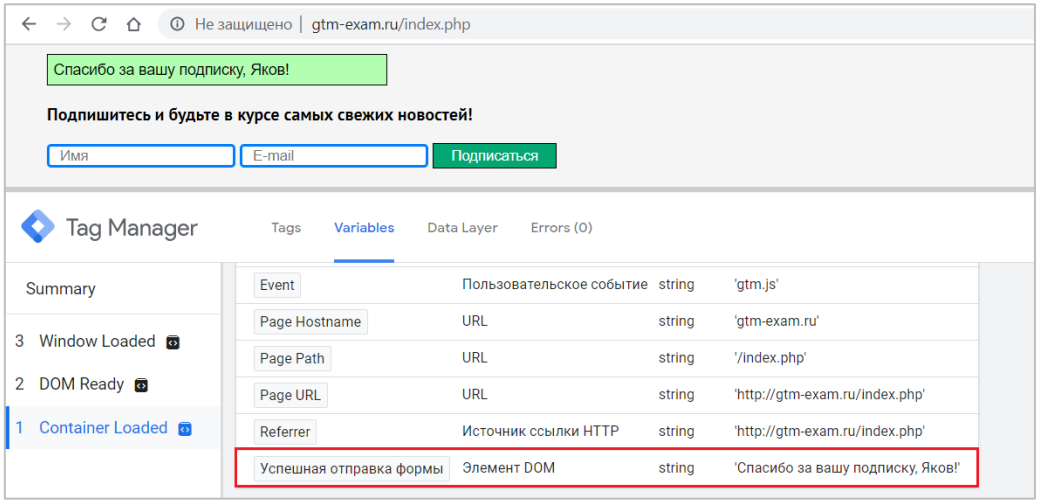


Рис. 629. Проверка в режиме предварительного просмотра

Теперь нам нужно создать триггер, который зависит от нашей новой переменной DOM. Я буду использовать тип триггера **Просмотр страницы** с условием срабатывания **Некоторые просмотры страниц – переменная Элемент DOM – содержит – Спасибо за вашу подписку.**

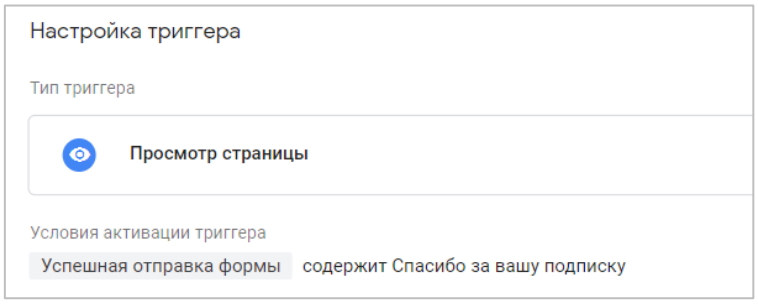


Рис. 630. Условие активации триггера Просмотр страницы

Таким образом, триггер будет срабатывать только на тех страницах, на которых посетителям отображается сообщение об успешной подписке. Остается только создать тег **Google Аналитика – Universal Analytics** с типом отслеживания **Событие**, а в качестве триггера активации задать **Просмотр страницы**.

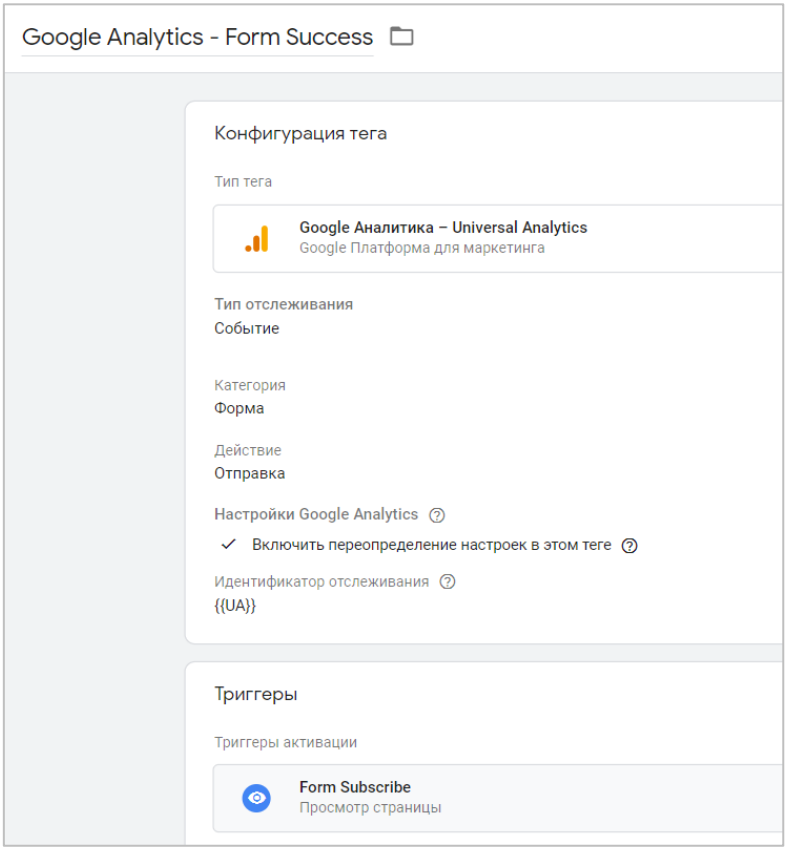


Рис. 631. Настройка тега

Если вы все сделали правильно, то после отправки формы в режиме отладки должен активироваться тег:

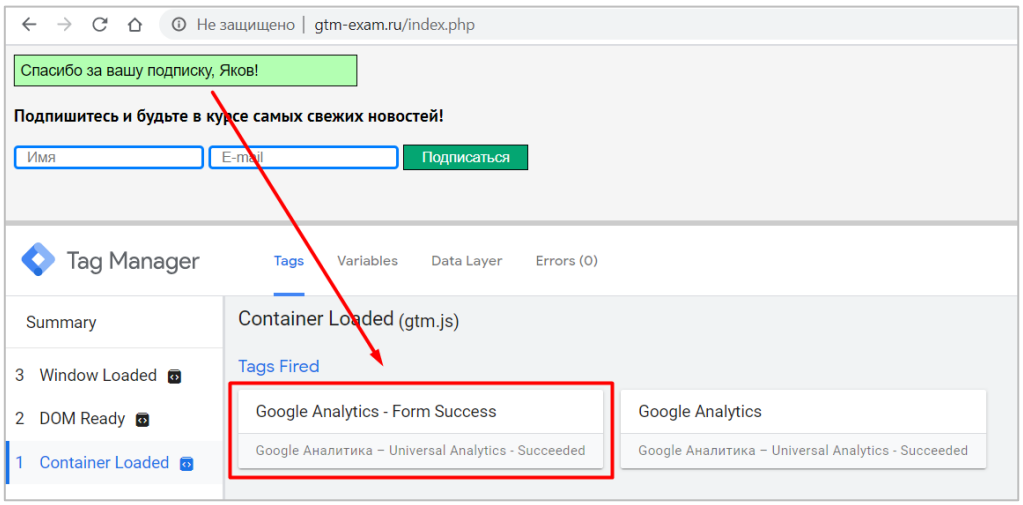


Рис. 632. Активация тега

А поскольку тег имеет тип отслеживание **Событие**, то данные в Google Analytics будут отображаться в отчете **Поведение - События - Лучшие события**.

8. Отслеживание отправки формы с помощью виртуальных страниц

По умолчанию, просмотр страницы в инструменты веб-аналитики отправляется каждый раз, когда вы:

- обновляете веб-страницу (на которой установлен счетчик веб-аналитики);
- переходите на новую страницу (на которой установлен счетчик веб-аналитики).

И в одном и другом случае у страницы есть URL-адрес, который вы видите в адресной строке браузера. Однако, когда дело доходит до одностраничных веб-сайтов (Single Page Application, SPA), такое отслеживание перестает работать.

Одностраничное приложение Single Page Application (SPA) - веб-сайт или веб-приложение, в котором все данные, необходимые для переходов между разделами сайта, загружаются вместе с первой страницей, а весь контент загружается динамически, без обновления страницы и/или без перехода на другую веб-страницу, следовательно, просмотра страницы не происходит, и данные в инструменты веб-аналитики не отправляются.

Как правило, при таком отслеживании используется все та же технология AJAX. Наиболее распространенным примером SPA являются сайты на конструкторе Tilda. Тильда передает данные в системы веб-аналитики на основе просмотра страниц. Когда пользователь совершает какое-либо действие на странице, создается **виртуальная страница**. Подробнее о том, как работают виртуальные страницы, читайте в соответствующей главе.

Мы же разберем практический пример отслеживания отправки формы с помощью данной технологии для сайта на конструкторе Tilda. Есть сайт, на котором добавлено n-ое количество форм, в том числе и на первом экране:

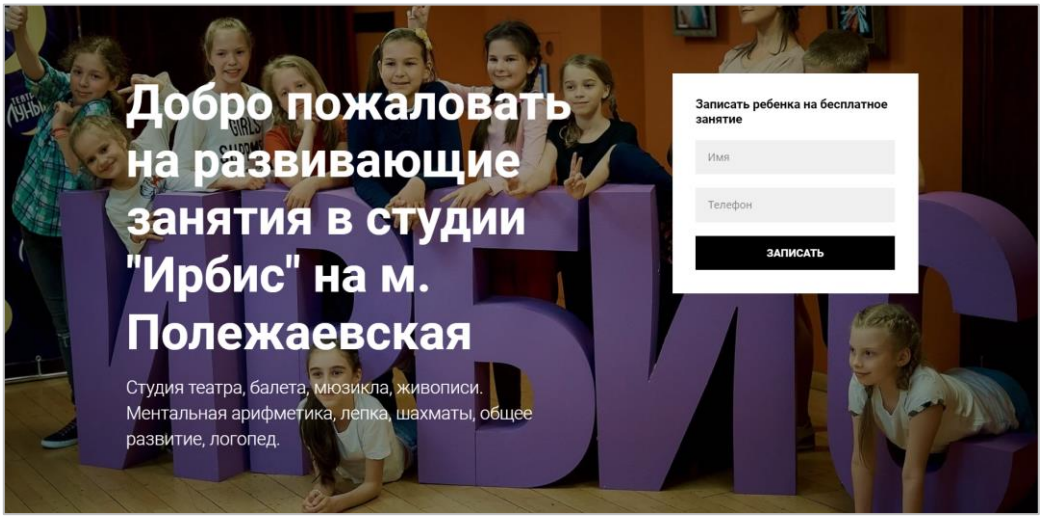


Рис. 633. Пример формы (сайт на Тильде)

Существует два подхода к анализу статистики:

1. через создание целей и фиксирование событий;
2. через просмотр страниц.

Когда пользователь совершает какое-либо действие на странице: открывает попап, нажимает на кнопку или заполняет форму, создается виртуальная страница. Данные о том, сколько человек «посетили» эту виртуальную страницу (другими словами, открыли попап или нажали на кнопку) всегда доступны в системах аналитики без дополнительных настроек.

Тильда передает данные в системы аналитики на основе просмотра страниц. Этот подход хорош тем, что данные сохраняются без вашего участия. На их основе вы можете создавать цели и смотреть статистку с того момента, как был установлен счетчик. В любой момент можно цель поменять, данные не пропадут и все равно будут доступны.

Данные о том, что пользователь заполнил форму и отправил информацию, поступают в системы аналитики автоматически. Адрес виртуальной страницы можно посмотреть в настройках блока: попапа, блока с кнопкой или формой. Пример того, как выглядит адреса виртуальных страниц:

- **tilda/popup/rec31654896/opened** — для попапа;
- **tilda/click/rec31742916/button1** — для клика на кнопку;
- **tilda/form31751802/submitted** — для заполнения формы.

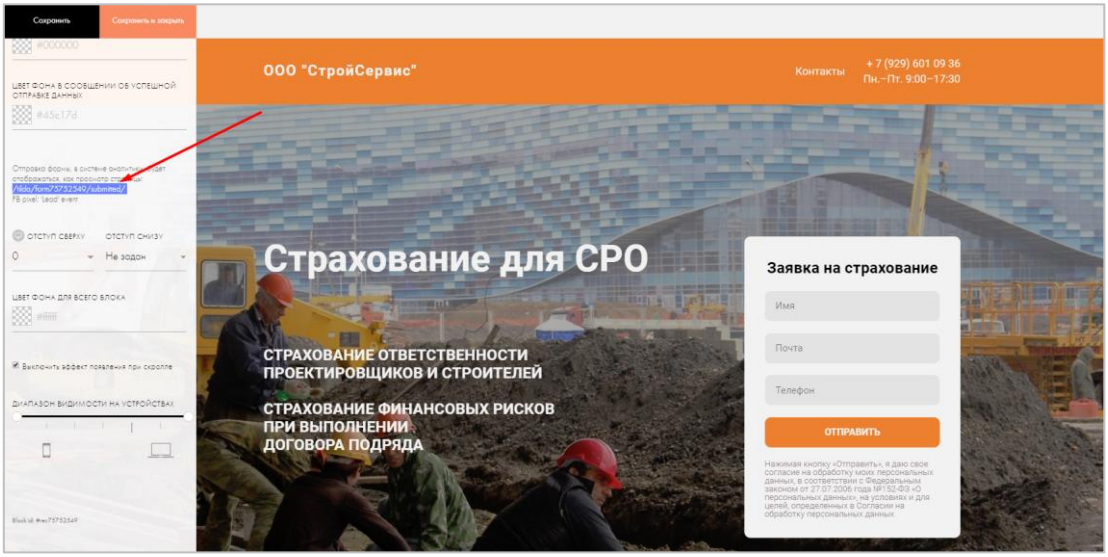


Рис. 634. Адрес виртуальной страницы отображается в настройках блока

Об этом подробно написано в справочном центре сервиса (см. приложение).

Если вы устанавливаете идентификаторы отслеживания Google Analytics и Яндекс.Метрики через интерфейс Тильды, то просмотры страниц создают автоматически при выполнении пользователями определенных событий. В случае установки кода Google Tag Manager необходимо произвести дополнительные настройки, поскольку в GTM виртуальные страницы по умолчанию не передаются. Вы можете воспользоваться этим материалом (см. приложение) и настроить отслеживание отправки формы с помощью виртуальных страниц.

Сначала необходимо создать три пользовательских переменных с типом **Переменная уровня данных**. Их имена **eventAction**, **referer** и **title** соответственно.

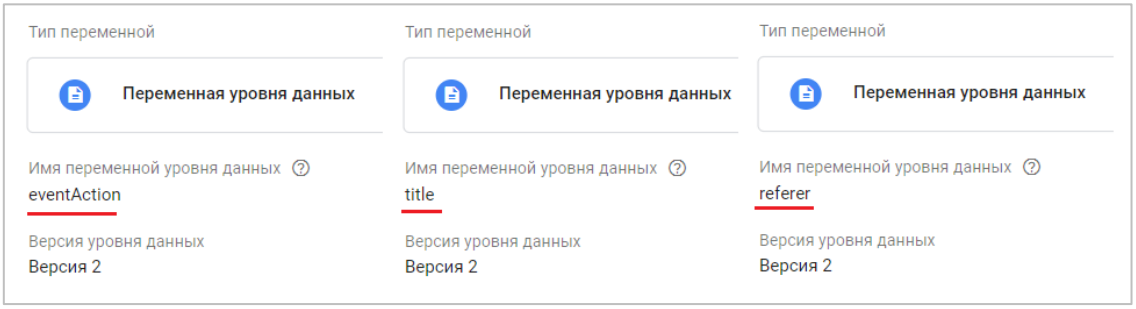


Рис. 635. Три переменных уровня данных

Переменная **eventAction** отвечает за адрес виртуальной страницы, а именно за формирование уникальной ссылки отправки формы. **title** – это заголовок формы, который предопределен в Тильде, а **referer** содержит URL исходной страницы, с которой был осуществлен переход на текущую страницу.

Затем создайте триггер с типом **Пользовательское событие** и именем **pageView**:

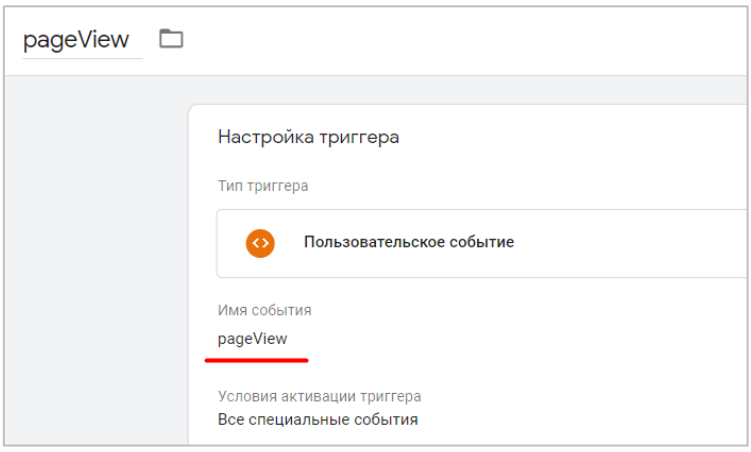


Рис. 636. Триггер Пользовательское событие

На завершающем этапе добавьте тег Google Аналитика - Universal Analytics с типом отслеживания **Просмотр страницы** и следующими настройками:

Поля, которые необходимо задать, название поля:

- page – **{{eventAction}}**
- title – **{{title}}**
- referrer – **{{referer}}**

Триггер активации – пользовательское событие **pageView**

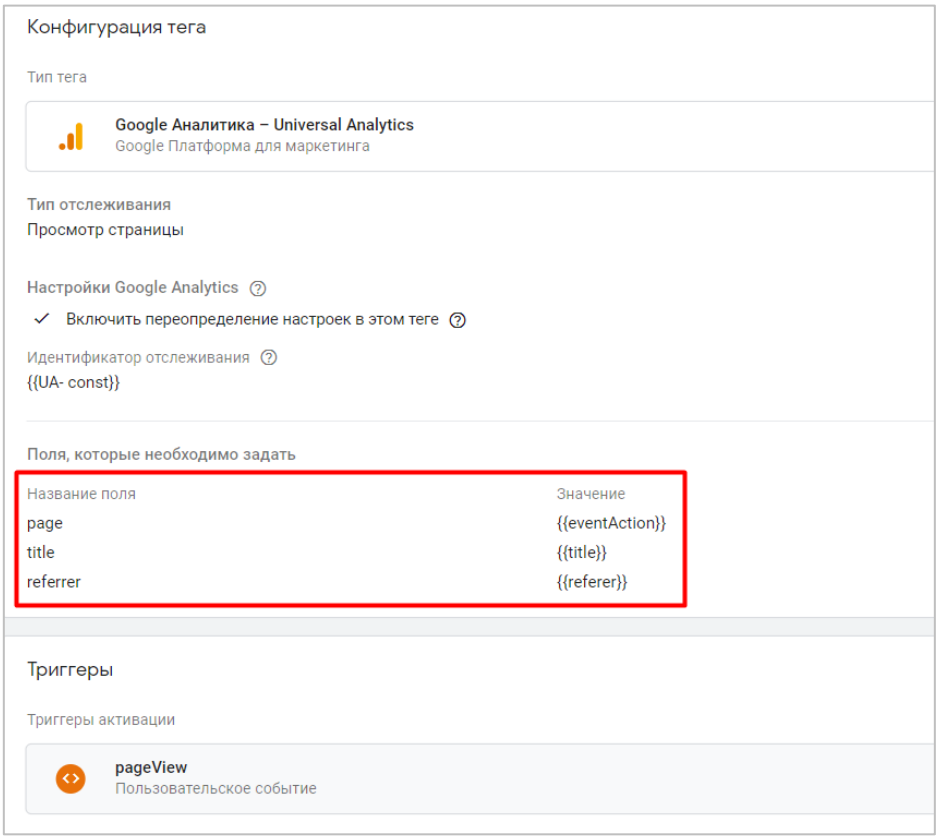


Рис. 637. Настройка тега

Сохраните изменения. Таким образом, когда пользователь отправит заявку на сайте, сработает событие **pageView**, а с помощью тега Google Analytics и поля **page** вы передадите в аналитику просмотр страницы с заголовком и реферера.

Если вы хотите передать такую же информацию в Яндекс.Метрику, то создайте тег типа **Пользовательский HTML** со следующим кодом:

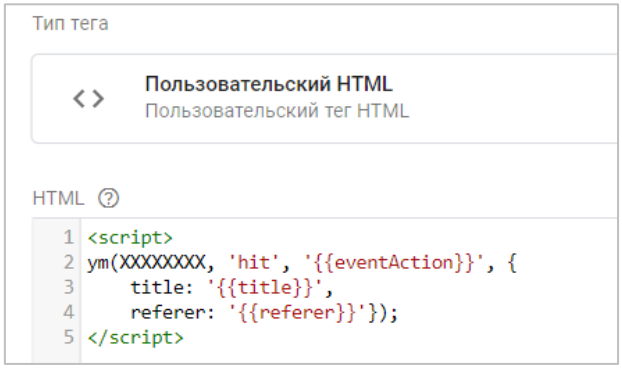


Рис. 638. Тег для Яндекс.Метрики

где **XXXXXXXX** – идентификатор счетчика Яндекс.Метрики. а конструкция **{{ }}** ссылается на три наших переменных уровня данных, которые были созданы на шаге ранее.

Подробнее об отправке данных о просмотре страницы в Яндекс.Метрику и методе **hit** читайте в официальной документации Яндекса (см. приложение).

Выполнить проверку настройки можно с помощью режима отладки Google Tag Manager и отправки тестового обращения. Если вы увидите событие **pageView** и активированный тег Google Analytics, это значит, что данные успешно отправлены в аналитику, а на вкладке Data Layer будет отображена информация по событию и трем нашим переменным:

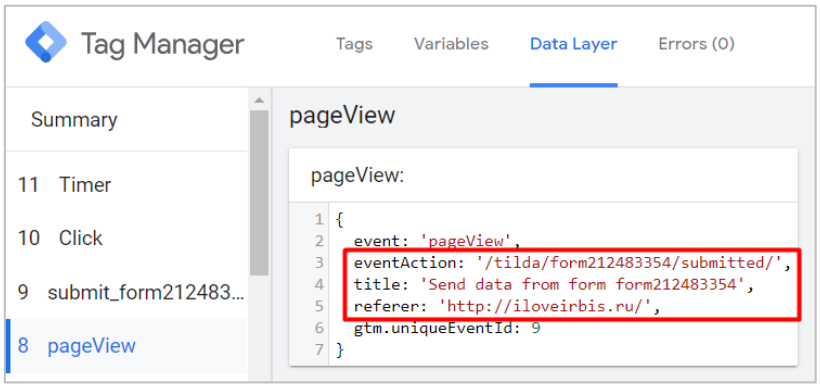


Рис. 639. Данные уровня данных при отправке формы

9. Отслеживание данных брошенных форм

Брошенная форма — это когда пользователь частично или полностью заполняет форму, но не отправляет ее нам по какой-либо причине: отвлекли, передумал, нечаянно закрыл вкладку браузера и не знает, как вернуться назад и т.д. С помощью Google Tag Manager вы можете отслеживать и такие формы.

Примечание: *все действия, которые описаны в этом материале, носят исключительно образовательный характер. Автор не призывает отслеживать персональные данные пользователей (телефон, e-mail и т.д.) и передавать их в инструменты веб-аналитики.*

В качестве примера я опишу процесс отслеживания данных формы на моем сайте **gtm.osipenkov.ru**. Сама форма содержит три поля: имя, e-mail и комментарий пользователя.

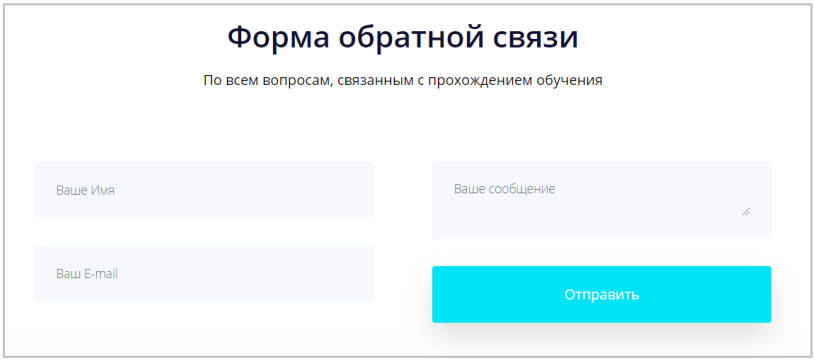


Рис. 640. Пример формы

Наша задача – передать информацию по всем трем полям в веб-аналитику даже в том случае, если пользователь не нажмет на кнопку Отправить.

Реализация завязана на браузерном событии **change**, которое происходит по окончании изменения значения элемента формы, когда это изменение зафиксировано. Для текстовых элементов это означает, что событие произойдет не при каждом вводе, а при потере фокуса. Например, пока вы набираете что-то в текстовом поле ниже – события нет. Но как только вы уведете фокус на другой элемент, например, нажмете кнопку – произойдет событие. Другими словами, пользователь сначала ввел данные в одно поле, потом перешел на другое. В этот момент вы в Google Analytics отправляете данные со значением этого поля. И так по всем полям.

Для того, чтобы передавать значения полей в Google Analytics, нам сначала необходимо эту информацию извлечь, сохранить в переменных. Для этого создайте в Google Tag Manager пользовательскую переменную типа **Собственный код JavaScript** с таким кодом:

```
function(){
  var vashePole = $('CSS-селектор').val();
  return vashePole;
}
```

где, вместо CSS-селектор вы вставляете селектор из собственного поля.

В GTM для поля **Имя** из моей формы это выглядит так:

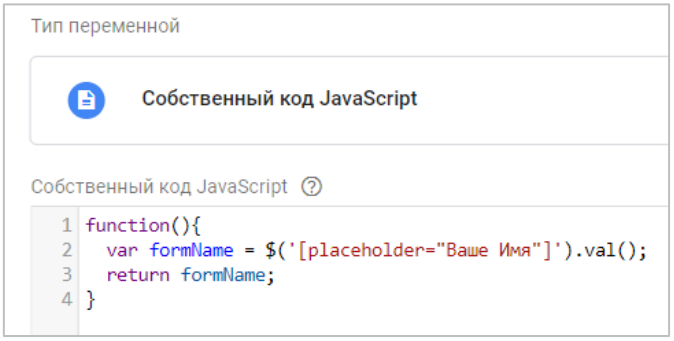


Рис. 641. Код для извлечения данных из текстового поля

Имя переменной **vashePole** может быть любым. Лично я называю их по имени самого поля в форме на сайте. Для **Имя – formName**, для **E-mail – formMail**, для комментария пользователя – **formMessage**.

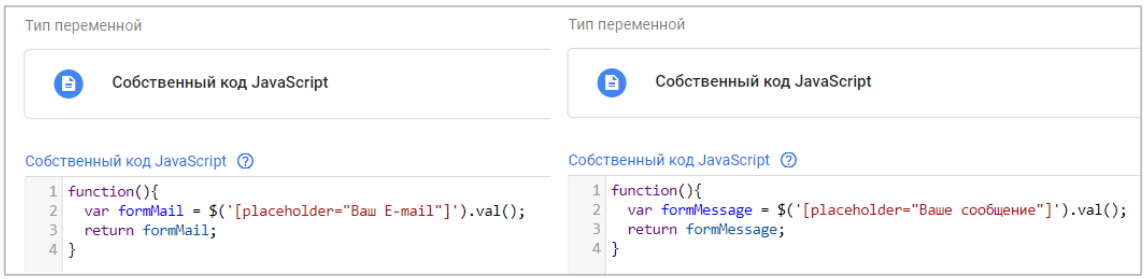


Рис. 642. Собственный код JavaScript

Чтобы узнать CSS-селектор конкретного поля, откройте в браузере консоль разработчика (клавиша F12 для Google Chrome). Выделите поле, которое хотите отследить. Нажмите на него правой кнопкой мыши и выберите **Copy – Copy selector**.

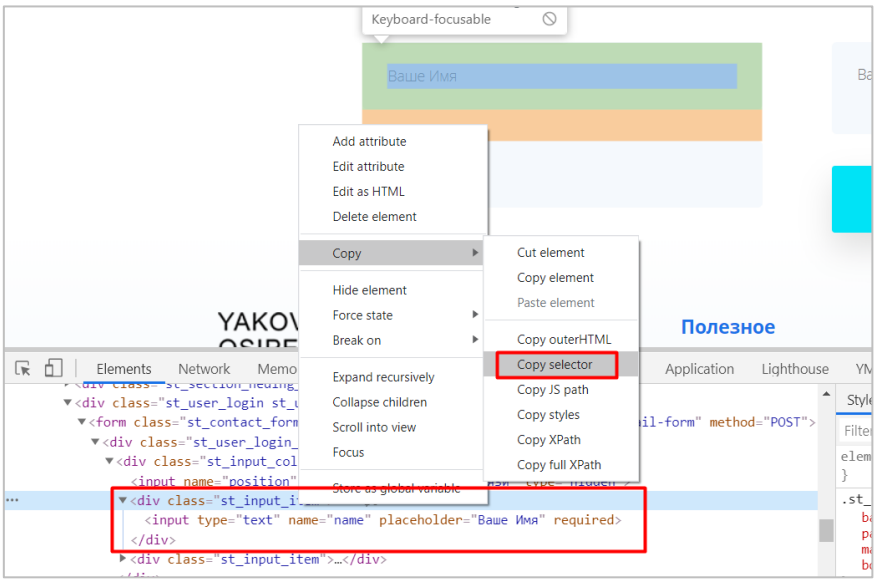


Рис. 643. Копирование селектора поля

После извлечения данных полей форм создайте тег типа **Пользовательский HTML** со следующими настройками:

```
<script>
(function() {
   var formSelector = 'CSS-селектор';
   document.querySelector(formSelector).addEventListener('change', function() {
      dataLayer.push ({
         'event':'formChange',
      })
   });
})();
</script>
```

где, вместо CSS-селектор вы вставляете селектор собственной формы.

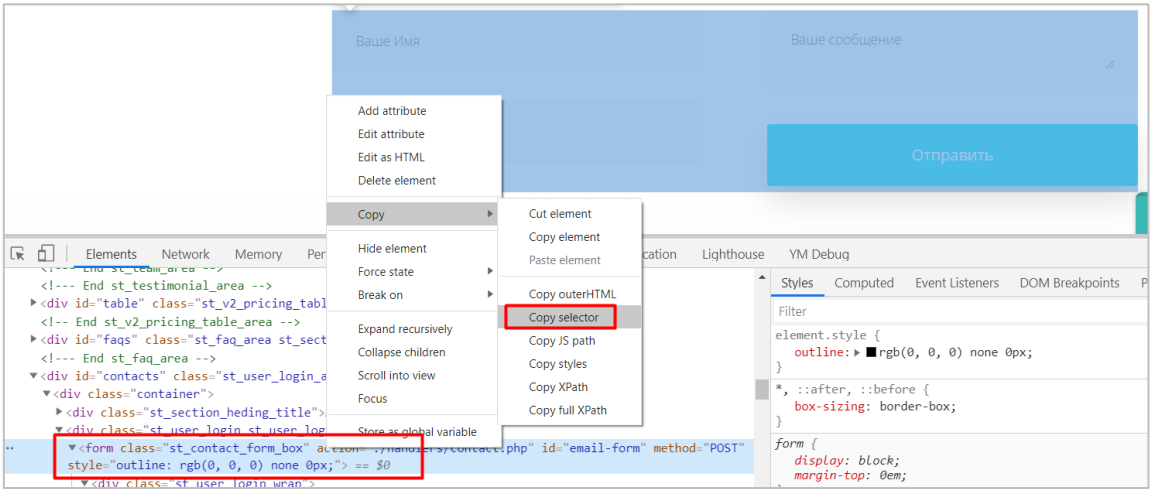


Рис. 644. Копирование селектора формы

Для моей формы код в Google Tag Manager будет выглядеть вот так:

Рис. 645. Код для тега Пользовательский HTML

Триггер активации – **Все страницы (All Pages)**. Добавьте триггер типа **Пользовательское событие** с именем **formChange**:

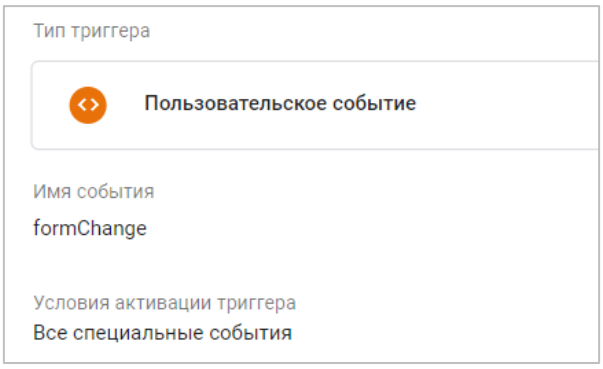


Рис. 646. Триггер Пользовательское событие

На заключительном шаге создайте тег **Google Аналитика – Universal Analytics** с типом отслеживания **Событие**. В качестве условия активации выберите пользовательское событие **formChange**.

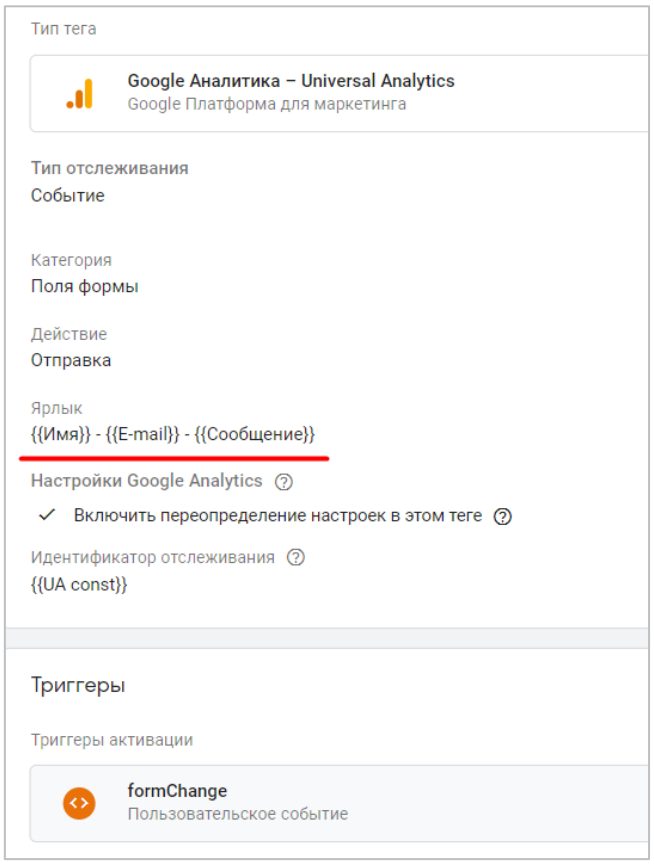


Рис. 647. Настройка тега

На рисунке выше приведен пример, в котором три переменных, созданных на предыдущих шагах, передаются в Google Analytics в поле **Ярлык**. Сохраните полученные настройки.

Проверьте передачу данных и активацию триггера в режиме предварительного просмотра Google Tag Manager. Начните заполнять форму: введите данные в поле Имя и переведите курсор в следующее поле E-mail. В этот момент должно сработать событие **formChange** и активироваться тег, в котором передастся значение поле Имя в Google Analytics.

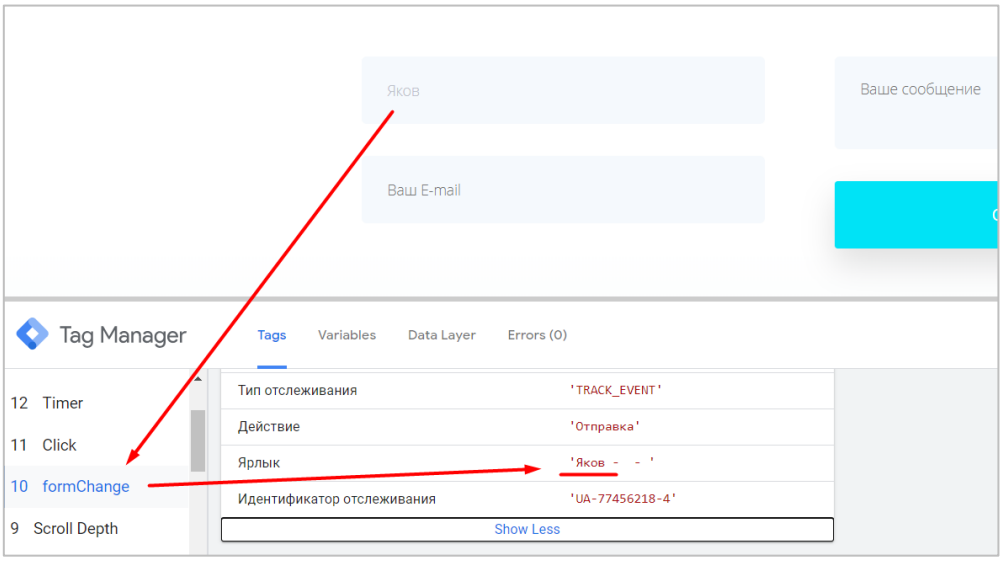


Рис. 648. Событие formChange в режиме отладки

В отчете Google Analytics **В режиме реального времени** вы увидите значение первого поля:

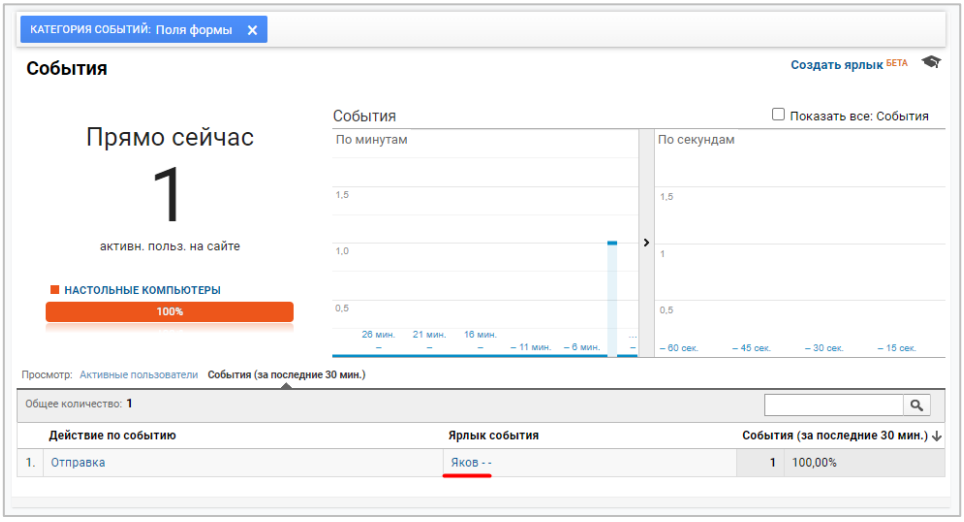


Рис. 649. Отчет В режиме реального времени – Заполнено одно поле

Если пользователь заполнит и второе поле, тогда в аналитику передастся еще одно событие **formChange**, но уже с двумя переменными:

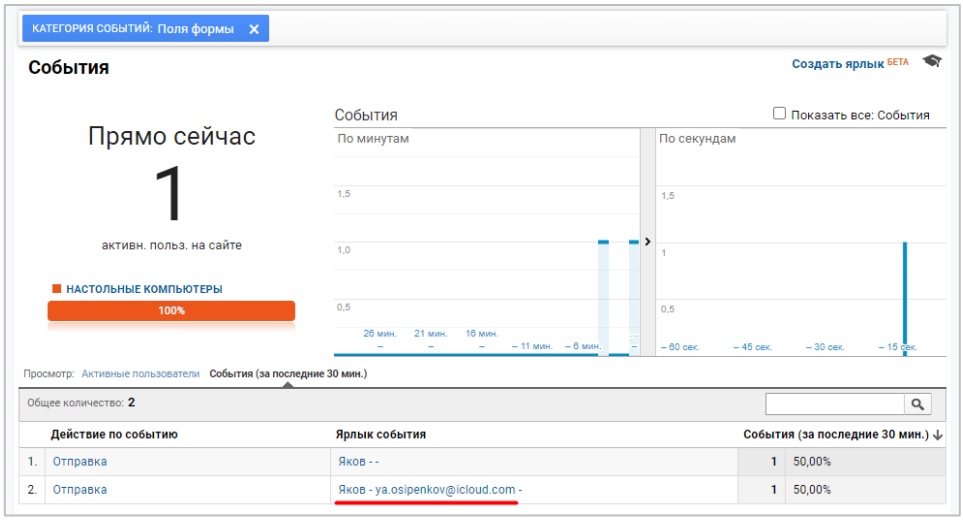


Рис. 650. Отчет В режиме реального времени – Заполнены два поля

Если пользователь заполнит все три поля, тогда в аналитику передастся еще одно событие **formChange**, но уже с тремя переменными:

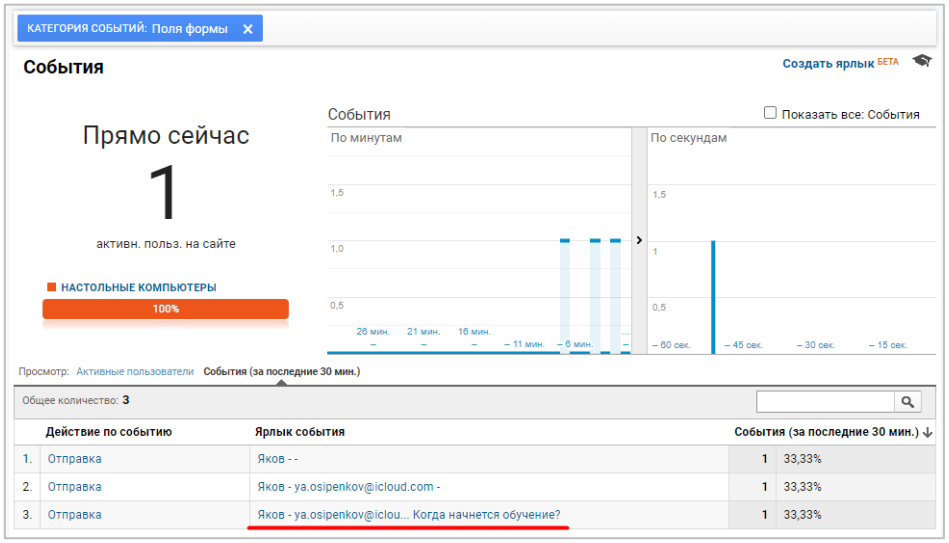


Рис. 651. Отчет В режиме реального времени – Заполнены три поля

Таким образом, вы можете отправлять в Google Analytics данные из полей форм тех пользователей, кто начал ее заполнять, но по какой-либо причине не сделал этого до конца или не отправил ее вам. Поскольку вы отправляете данные как события, то увидеть их можно в отчете **Поведение - События - Лучшие события**.

Основная проблема с отслеживанием форм в Google Tag Manager — отсутствие единого стандарта создания форм. Разработчик может сделать форму на HTML, а может в AJAX. В каких-то случаях будет достаточно встроенного триггера **Отправка формы**, а где-то поможет только уровень данных и пользовательское событие.

В этой главе мы с вами разобрали 9 различных способов отслеживания отправки формы с помощью Google Tag Manager и передачу этой информации в инструменты веб-аналитики, включая брошенные формы. В зависимости от метода реализации отправки формы на вашем сайте, используйте один из предложенных вариантов отслеживания.

Глава 9

Работа с элементами на странице

Тег <input>

В HTML существует тег **<input>**, который является одним из разносторонних элементов формы и позволяет создавать разные элементы интерфейса и обеспечить взаимодействие с пользователем. Он предназначен для создания текстовых полей, различных кнопок, переключателей и флажков.

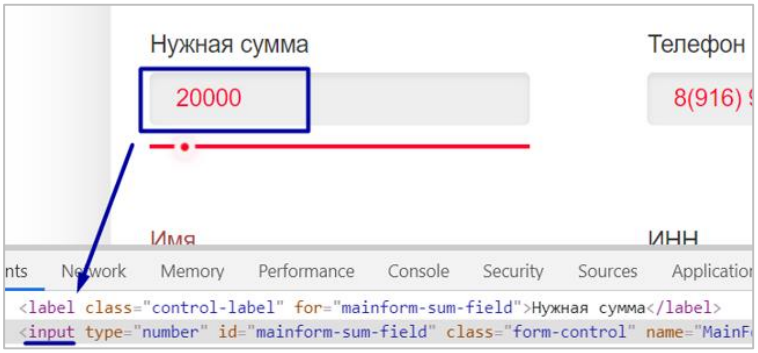


Рис. 652. Тег <input>

Атрибут type

Основной атрибут тега <input>, определяющий вид элемента — **type**. Он сообщает браузеру, к какому типу относится элемент формы.

В HTML5 добавлено еще более десятка новых элементов (см. приложение).

Тип	Описание	Вид
button	Кнопка.	Кнопка
checkbox	Флажки. Позволяют выбрать более одного варианта из предложенных.	Пиво Чай Кофе
file	Поле для ввода имени файла, который пересылается на сервер.	Выберите файл Файл не выбран
hidden	Скрытое поле. Оно никак не отображается на веб-странице.	
image	Поле с изображением. При нажатии на рисунок данные формы отправляются на сервер.	Отправить
password	Обычное текстовое поле, но отличается от него тем, что все символы показываются звездочками. Предназначено для того, чтобы никто не подглядел вводимый пароль.	•••••••••
radio	Переключатели. Используются, когда следует выбрать один вариант из нескольких предложенных.	Пиво Чай Кофе
reset	Кнопка для возвращения данных формы в первоначальное значение.	Сбросить
submit	Кнопка для отправки данных формы на сервер.	Отправить
text	Текстовое поле. Предназначено для ввода символов с помощью клавиатуры.	

Рис. 653. Атрибут type

Примеры: текстовое поле (text), поле с паролем (password), переключатель (radio), флажок (checkbox), скрытое поле (hidden), кнопка (button), кнопка для отправки формы (submit), кнопка для очистки формы (reset), поле для отправки файла (file) и кнопка с изображением (image).

Наиболее распространенными элементами, которые вы можете встретить в форме, являются следующие атрибуты **type**:

checkbox

Этот атрибут определяет, помечен ли заранее такой элемент формы, как флажок или переключатель. Имеет синтаксис:

<input type="checkbox" checked>

checked означает предварительное выделение переключателя

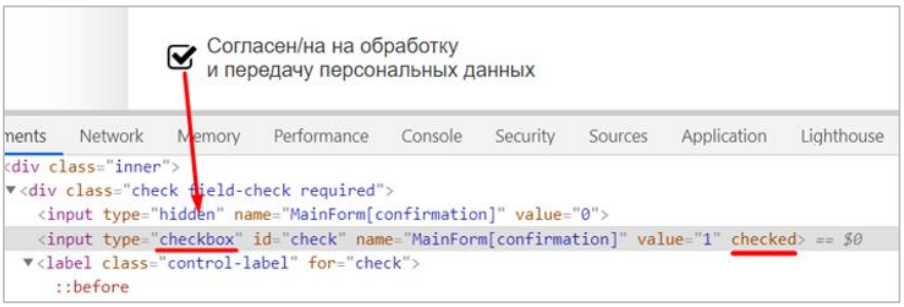


Рис. 654. Атрибут checkbox

radio

Переключатели (жарг. радиокнопки) используют, когда необходимо выбрать один единственный вариант из нескольких предложенных. Имеет синтаксис:

<input type="radio" checked>

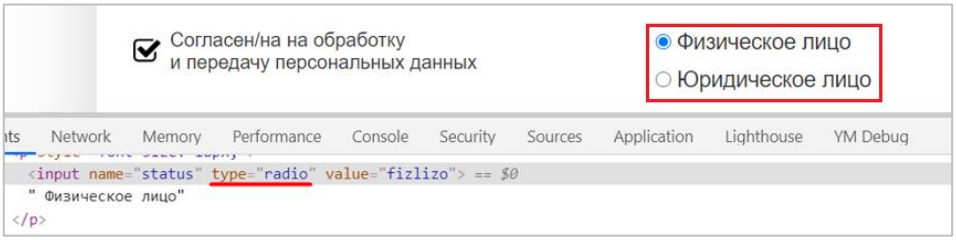


Рис. 655. Атрибут radio

reset

Кнопка для возвращения данных формы в первоначальное значение. Имеет синтаксис:

<input type ="reset">

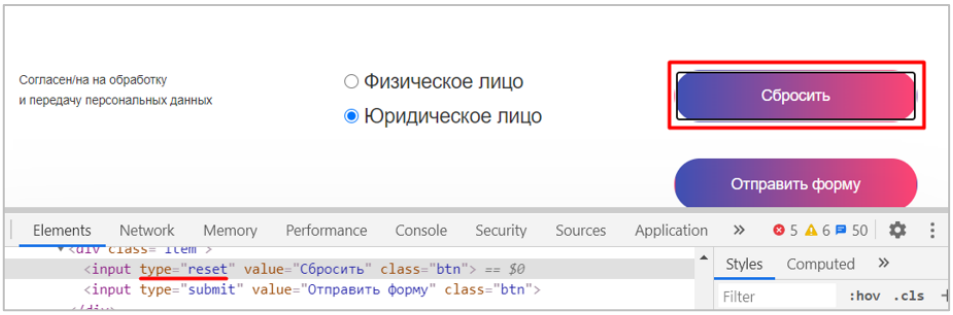


Рис. 656. Атрибут reset

Сбрасываются значения во всех полях формы.

number

Отвечает за ввод чисел в поле формы. Имеет синтаксис:

<input type ="number">

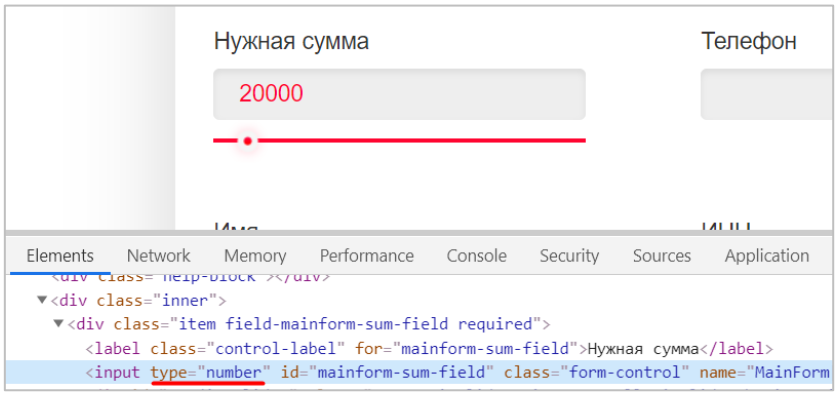


Рис. 657. Атрибут number

range

Ползунок для выбора чисел в указанном диапазоне. Имеет синтаксис:

<input type ="range">

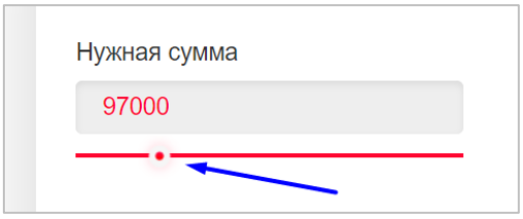


Рис. 658. Атрибут range

text

Текстовое поле. Предназначено для ввода символов с помощью клавиатуры. Имеет синтаксис:

<input type ="text">

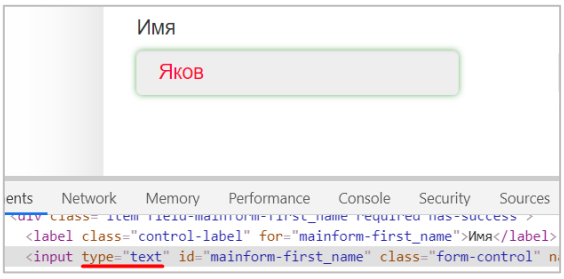


Рис. 659. Атрибут text

Тег <select>

Позволяет создать элемент интерфейса в виде раскрывающегося списка, а также список с одним или множественным выбором. Бывают списки **множественного выбора** и **раскрывающиеся** (см. приложение).

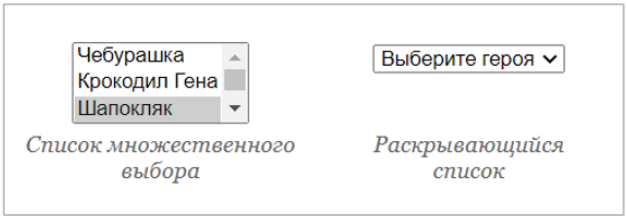


Рис. 660. Тег <select>

Конечный вид зависит от использования атрибута size тега **<select>**, который устанавливает высоту списка. Ширина списка определяется самым широким текстом, указанным в теге **<option>**, а также может изменяться с помощью стилей. Каждый пункт создается с помощью тега **<option>**, который должен быть вложен в контейнер **<select>**.

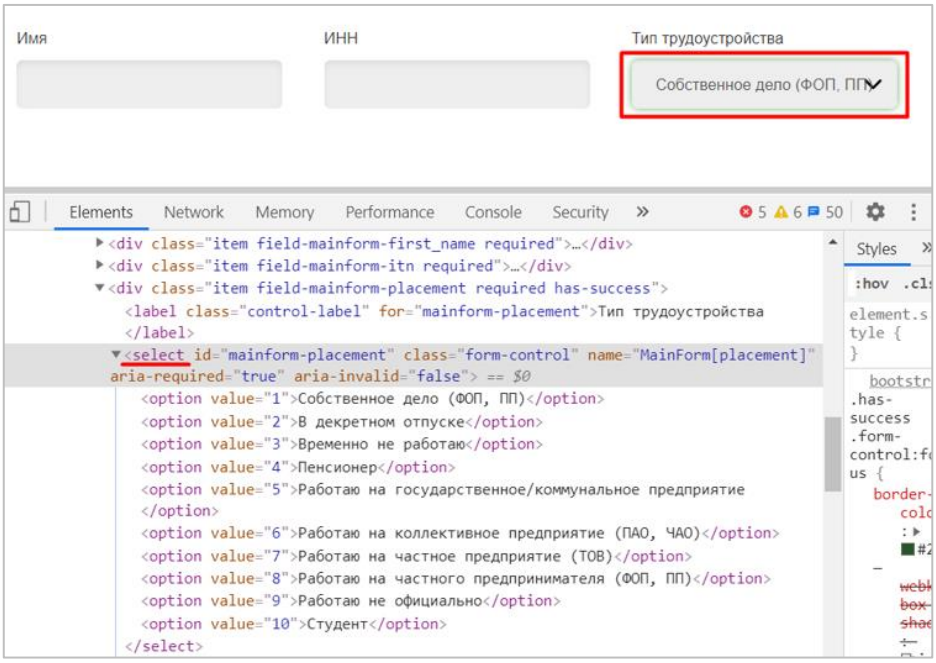


Рис. 661. Тег <select>

Теперь давайте разберем отслеживание каждого элемента подробнее.

Отслеживание чекбоксов

Согласны ли вы с пользовательским соглашением? Отметьте галочкой правильный ответ. Отправляя форму, вы соглашаетесь получать электронные письма с новостями нашей компании. Как правило, все эти сценарии реализуются с помощью флажка, галочки, или как его чаще всего называют, чекбоксом (checkbox), который пользователь ставит перед отправкой данных.

В качестве примера разберем форму на сайте graphanalytics.ru, где после заполнения полей необходимо **Принять пользовательское соглашение**, поставив галочку.

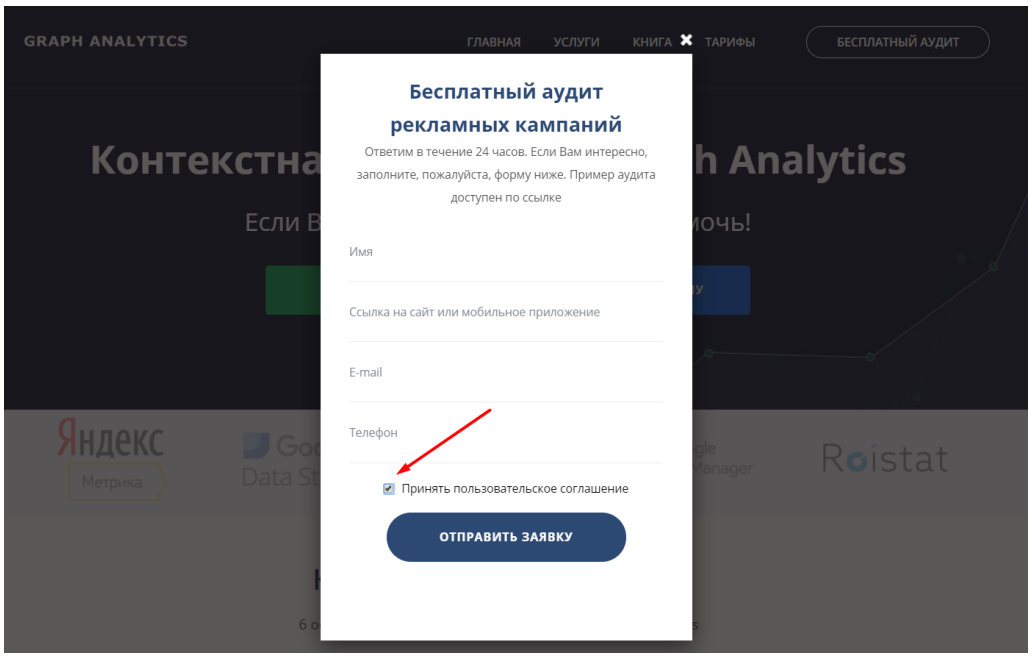


Рис. 662. Пример отслеживания

По сути, у такого элемента есть два состояния:

- галочка стоит (true);
- галочка не стоит (false).

Третьего не дано. Отслеживается достаточно просто. В Google Tag Manager создайте пользовательскую переменную типа **Собственный код JavaScript**, в которую вставим этот кусок кода:

```
function checked() {
if($('#checker').prop('checked')) {
return true;
} else {
return false;
}
}
```

В GTM это выглядит так:

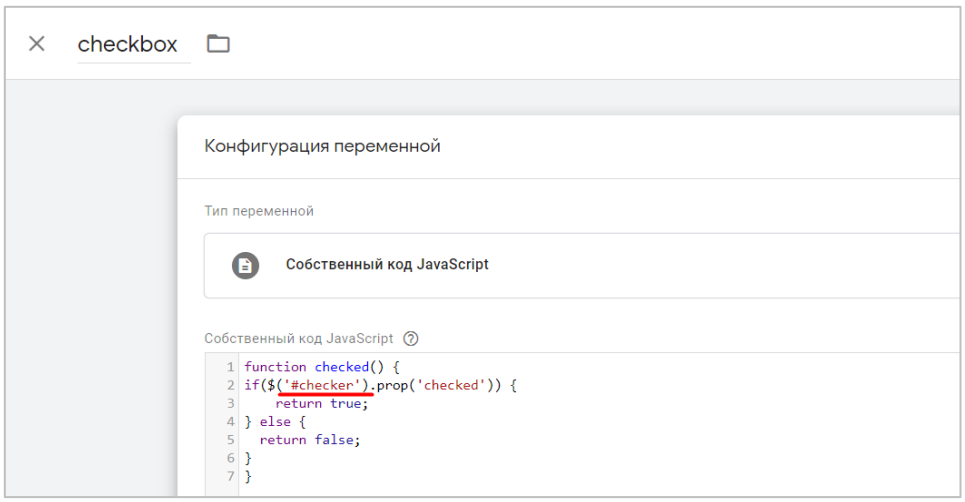


Рис. 663. Переменная Собственный код JavaScript

Вместо **#checker** вы подставляете значение собственного идентификатора элемента. Чтобы его найти, перейдите в консоль разработчика (клавиша F12 в Google Chrome) и выделите тот элемент, который хотите отслеживать. В моем случае чекбокс имеет **id = 'checker'**:

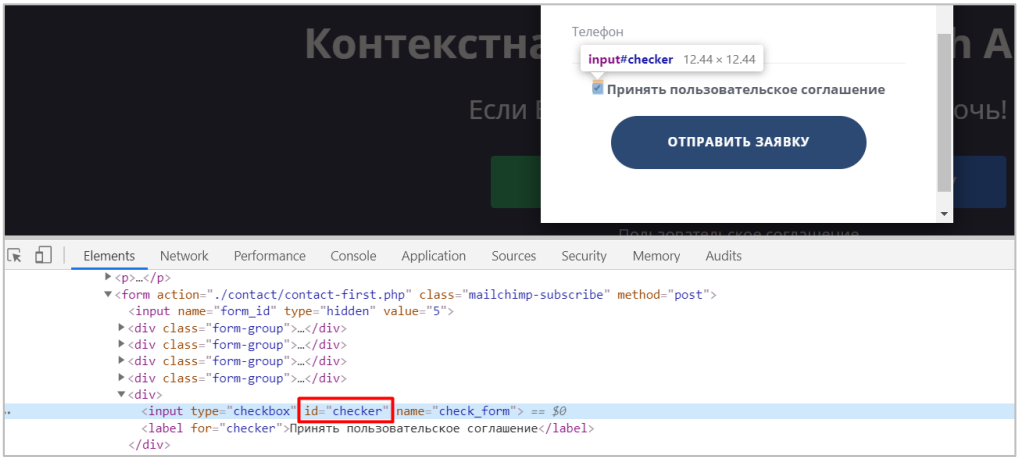


Рис. 664. Идентификатор (id) элемента - checker

Это и есть идентификатор элемента. Аналогично работает код:

```
function() {
 var checkbox = document.getElementById("checker");
 return checkbox.checked || false;
}
```

Если у вашего элемента нет идентификатора, то вы можете скопировать его CSS-селектор и вставить в тоже место.

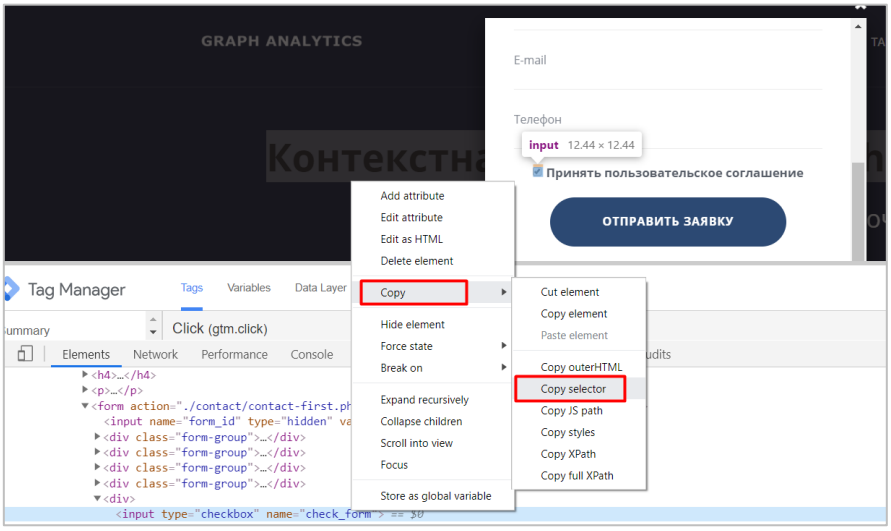


Рис. 665. Копирование CSS-селектора

В моем примере я получаю код такого вида:

```
function checked() {
if($('#popup-audit > div > form > div:nth-child(6) > input[type=checkbox]')
.prop('checked')) {
return true;
} else {
return false;
}
}
```

Что делают эти коды? Функционал одинаковый – если чекбокс стоит (галочка есть), то в переменную возвращается значение *true*, если не стоит, то *false*. Сохраняем пользовательскую переменную.

Теперь необходимо создать триггер активации, который срабатывал бы только в том случае, когда чекбокс стоит, то есть принимало значение *true*. Поскольку галочку пользователь ставит с помощью клика мыши, то

создаем триггер типа **Клик – Все элементы**. Выбираем условие активации триггера – Некоторые клики и активируем при условии **checkbox – содержит (или равно) – true**

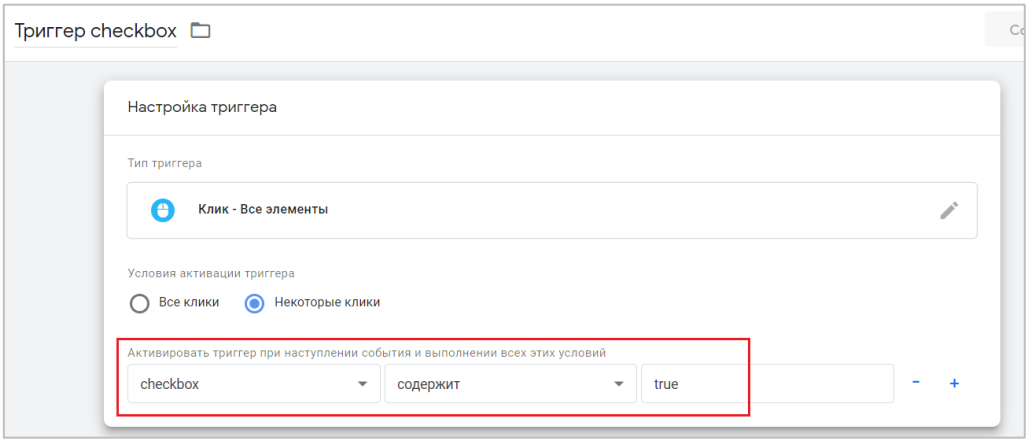


Рис. 666. Условие активации триггера

Триггер будет срабатывать только тогда, когда наша пользовательская переменная будет иметь значение *true*, то есть когда чекбокс с галочкой. Здесь могут разные условие. Например, у вас несколько форм и каждая из них срабатывает на отдельной странице. Тогда в качестве дополнительного условия вы можете задать **{{Page URL}} содержит Страница** и т.д. У меня в примере форма одна, поэтому я просто сохраняю триггер.

Осталось создать тег, который будет передавать эту информацию в системы аналитики. Тег типа **Google Аналитика - Universal Analytics** с типом **Событие** и следующими настройками:

- **Категория** – checkbox (произвольно)
- **Действие** – active (произвольное)

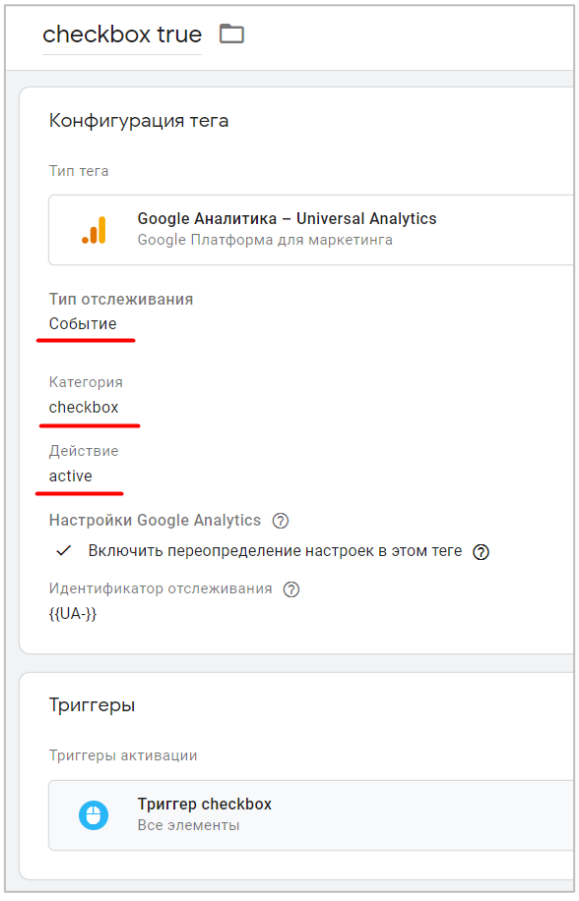


Рис. 667. Тег Google Аналитика - Universal Analytics с типом Событие

Чтобы проверить корректность настройки, воспользуемся режимом предварительного просмотра. Когда мы ставим галочку, активируется наш тег.

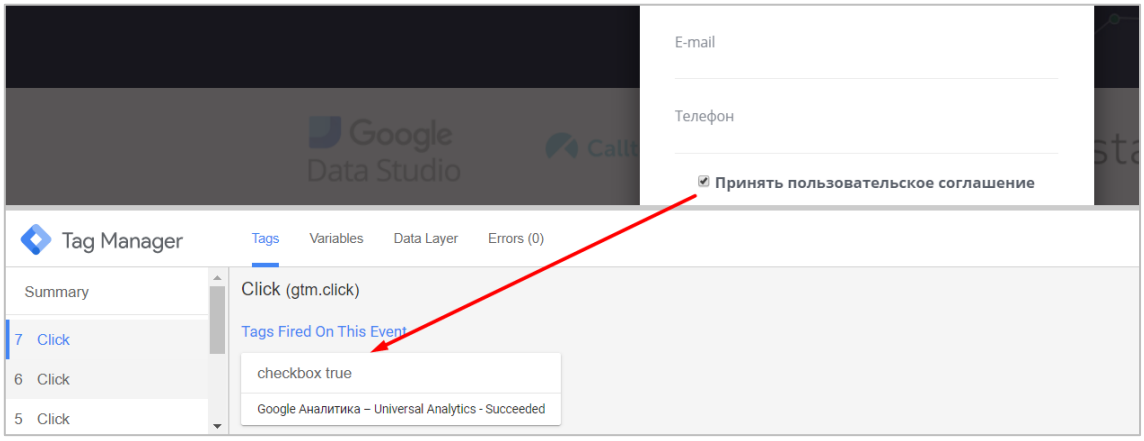


Рис. 668. Активация тега

Данные по событию можно посмотреть в Google Analytics в режиме реального времени:

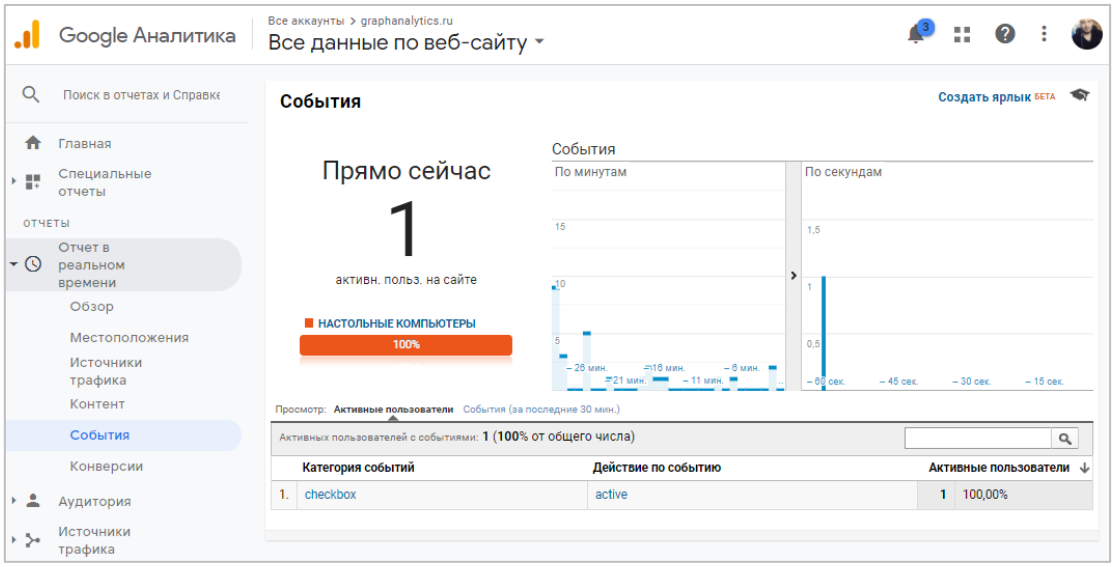


Рис. 669. Отчет В режиме реального времени

Все события доступны в отчете **Поведение - События - Лучшие события**.

Как еще можно проверить, что чекбокс принимает значения true или false? Откройте консоль разработчика, вкладку **Console** и вставьте следующий код:

```
$('#popup-audit > div > form > div:nth-child(6) > input[type=checkbox]')
.prop('checked')
```

вместо **#popup-audit > div > form > div:nth-child(6) > input[type=checkbox]** необходимо вставить свой CSS-селектор.

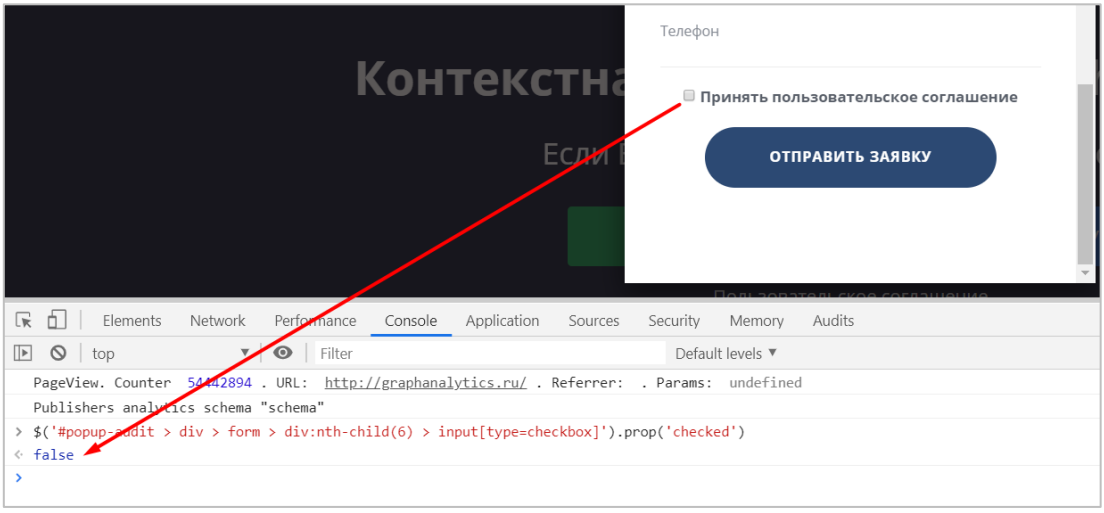


Рис. 670. Проверка, false (чекбокс не установлен)

Галочки нет, нам вернулось значение *false*. Чекбокс активен, значение *true*:

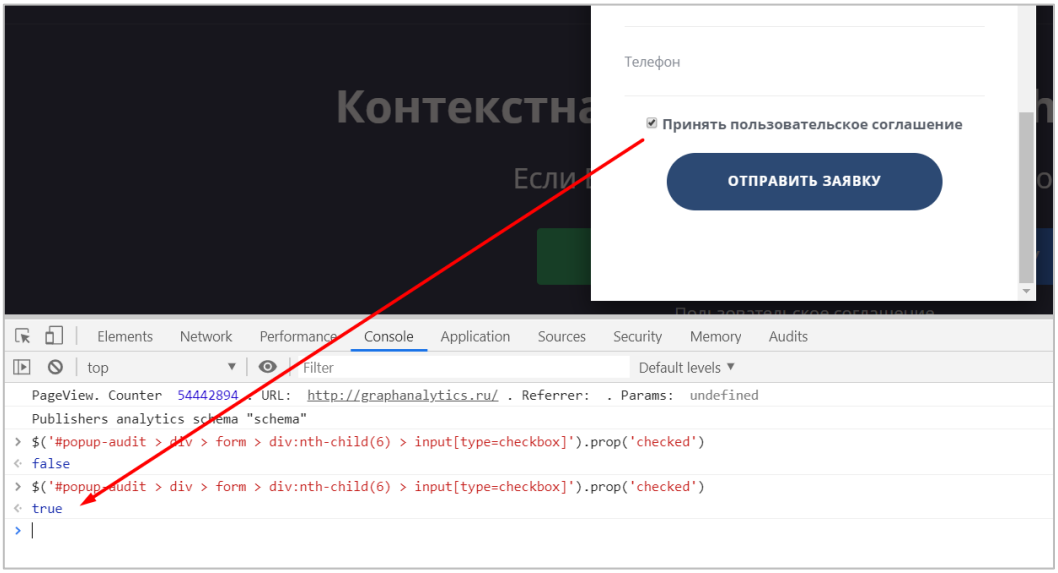


Рис. 671. Проверка, true (чекбокс установлен)

Есть и другие варианты отслеживания активации чекбоксов в Google Tag Manager. И с помощью **querySelector()**, и **getElementsByTagName** по n-ому элементу, и через свойства элемента DOM (**previousSibling.innerText || previousSibling.textContent**). Симо Ахава в своей статье (см. приложение) разбирает данные примеры и приводит плюсы и минусы таких подходов. На мой взгляд, самый оптимальный способ отслеживания тот, который описан в этой главе. 1 переменная, 1 триггер и 1 тег. А код, который вы будете использовать в качестве своего решения, выбираете вы самостоятельно.

Отслеживание состояний ползунка

Представленный в спецификации HTML5 ползунок выбора диапазона (**range**) является одним из типов у элемента **input**. Он позволяет осуществить выбор какого-то числового значения в указанном диапазоне.

Рис. 672. Пример элемента "ползунок"

Мы можем передвигать ползунок вправо или влево, чтобы выбрать нужное значение в рамках заданного диапазона. Такой элемент часто используют в автомобильном, финансовом, банковском и других секторах - на сайте размещается калькулятор или форма с N количеством полей, где одним из них и является **<input type='range'>** с возможностью выбора определенной суммы из представленного диапазона.

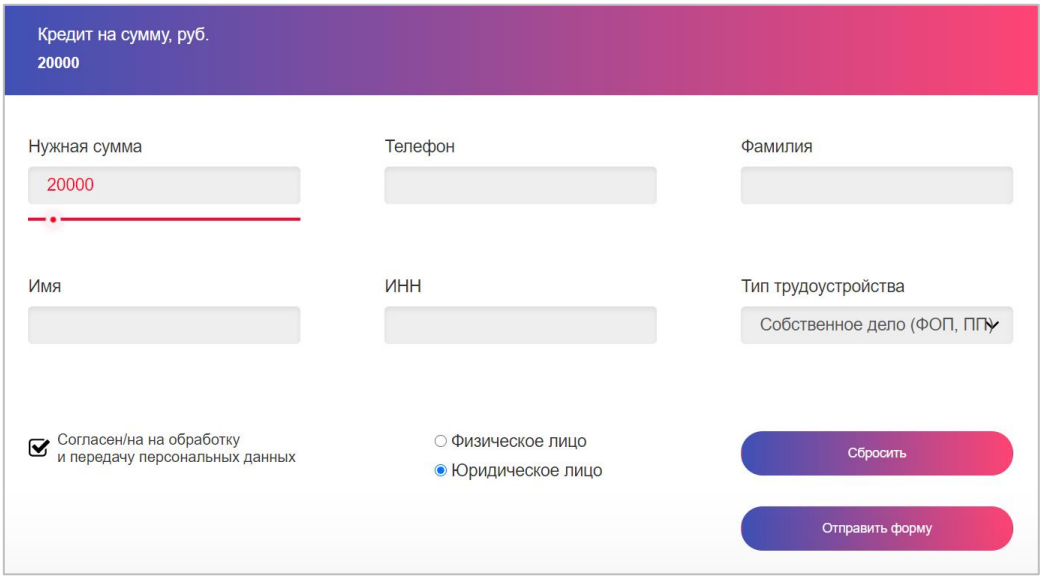


Рис. 673. Пример формы с выбором суммы с помощью ползунка

В качестве примера я покажу как отслеживать состояние ползунка с помощью Google Tag Manager на сайте **graphanalytics.ru**. На нем размещена форма, в которой пользователю необходимо выбрать значение конверсии сайта с помощью ползунка. Итоговое число мы передадим в Google Analytics.

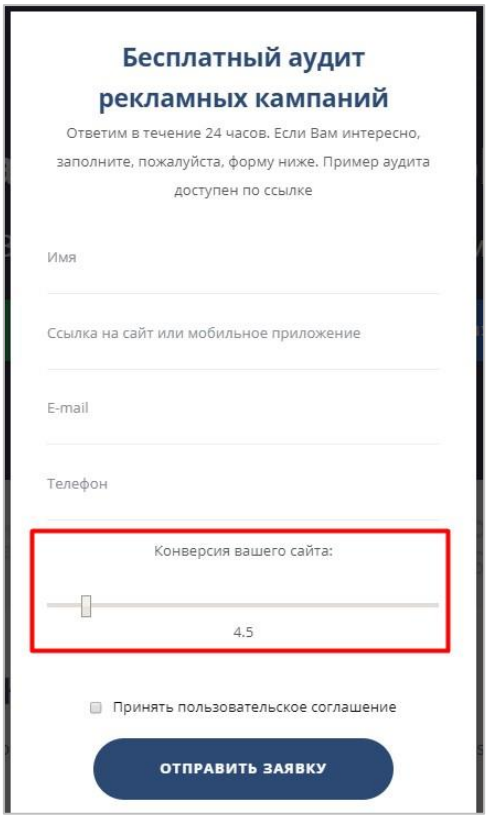


Рис. 674. Пример отслеживания

Последовательность действий следующая:

- создать пользовательскую переменную;
- создать тег прослушивания событий;

- создать пользовательскую переменную для прослушивания;
- создать пользовательское событие;
- создать теги для передачи данных в аналитику.

Создание пользовательской переменной

Для этого переходим в раздел **Переменные** и создаем переменную типа **Собственый код JavaScript**:

```
function (){
var cr = document.querySelector('#rangeValue').innerText;
  return cr;
}
```

В Google Tag Manager это будет выглядеть так:

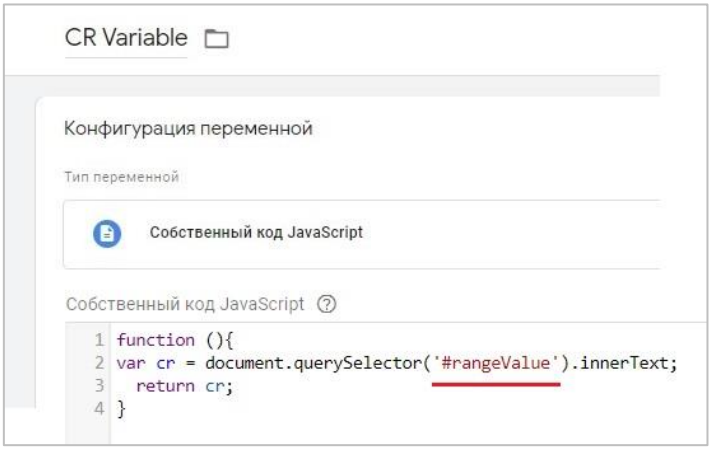


Рис. 675. Переменная Собственный код JavaScript

Эта переменная, которая возвращает значение из селектора элемента **#rangeValue** в виде числа.

Примечание: при настройке вам необходимо изменить подчеркнутый красным CSS-селектор на свой.

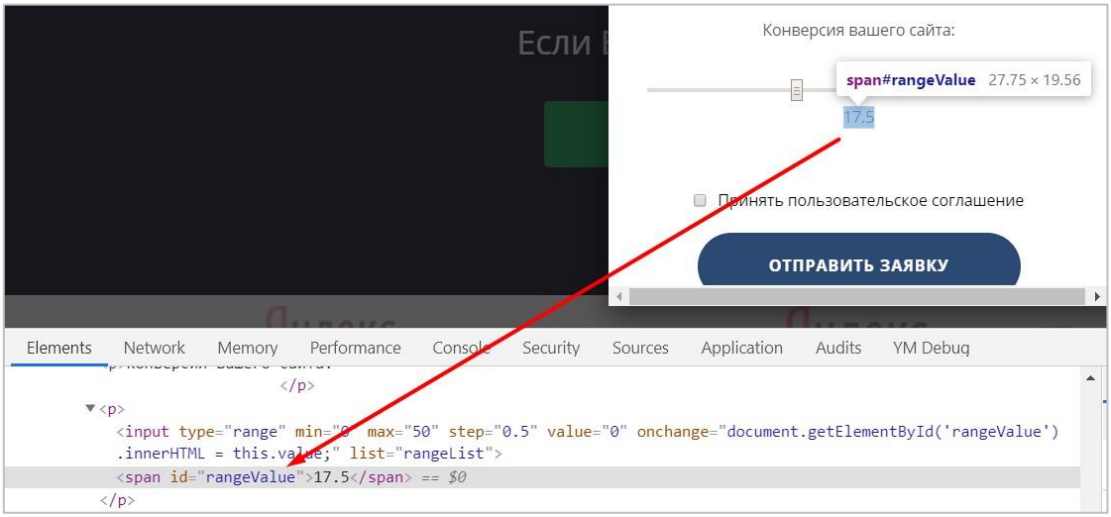


Рис. 676. Идентификатор (id) rangeValue

Сохраняем переменную. Это не единственный способ определения переменной. Для кого-то может подойти такая конструкция:

```
function (){
var cr = document.getElementById('ID').value;
  return cr;
}
```

Принцип такой же, только возвращается значение элемента по его идентификатору.

Создание тега прослушивания событий

В главе этого руководства, посвященной продвинутым настройкам Google Tag Manager, о прослушивании пользовательских событий в Google Tag Manager. Коды, размещенные в том материале, пригодятся нам для решения этой задачи.

В разделе **Переменные** создаем еще одну переменную типа **Собственый код JavaScript**:

```
function() {
return function(e) {
window.dataLayer.push({
'event': 'gtm.'+e.type,
'gtm.element': e.target,
'gtm.elementClasses': e.target.className || '',
'gtm.elementId': e.target.id || '',
'gtm.elementTarget': e.target.target || '',
'gtm.elementUrl': e.target.href || e.target.action || '',
'gtm.originalEvent': e
});
}
}
```

В GTM это выглядит так:

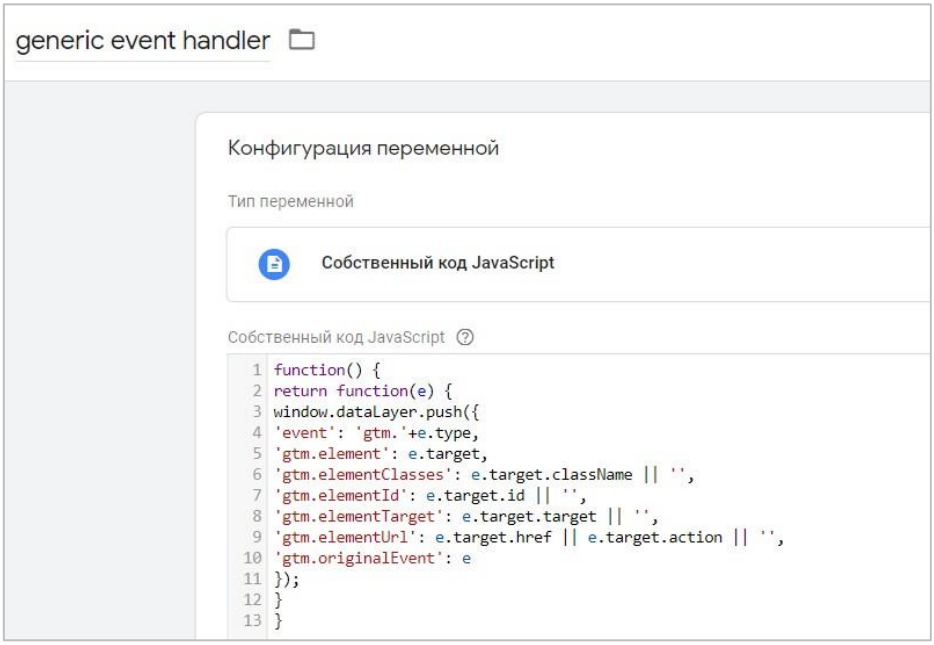


Рис. 677. Переменная Собственный код JavaScript (generic event handler)

Вводим название переменной **generic event handler** (или любое другое) и сохраняем изменения. Теперь переходим в раздел **Теги** и создаем тег типа **Пользовательский HTML** со следующим кодом:

```
<script>
(function() {
var eventType = "change"; // Измените на тот тип события, который хотите
прослушать
if (document.addEventListener) {
document.addEventListener(eventType, {{generic event handler}}, false);
} else if (document.attachEvent) {
document.attachEvent('on' + eventType, {{generic event handler}});
}
})();
</script>
```

В Google Tag Manager тег выглядит следующим образом:

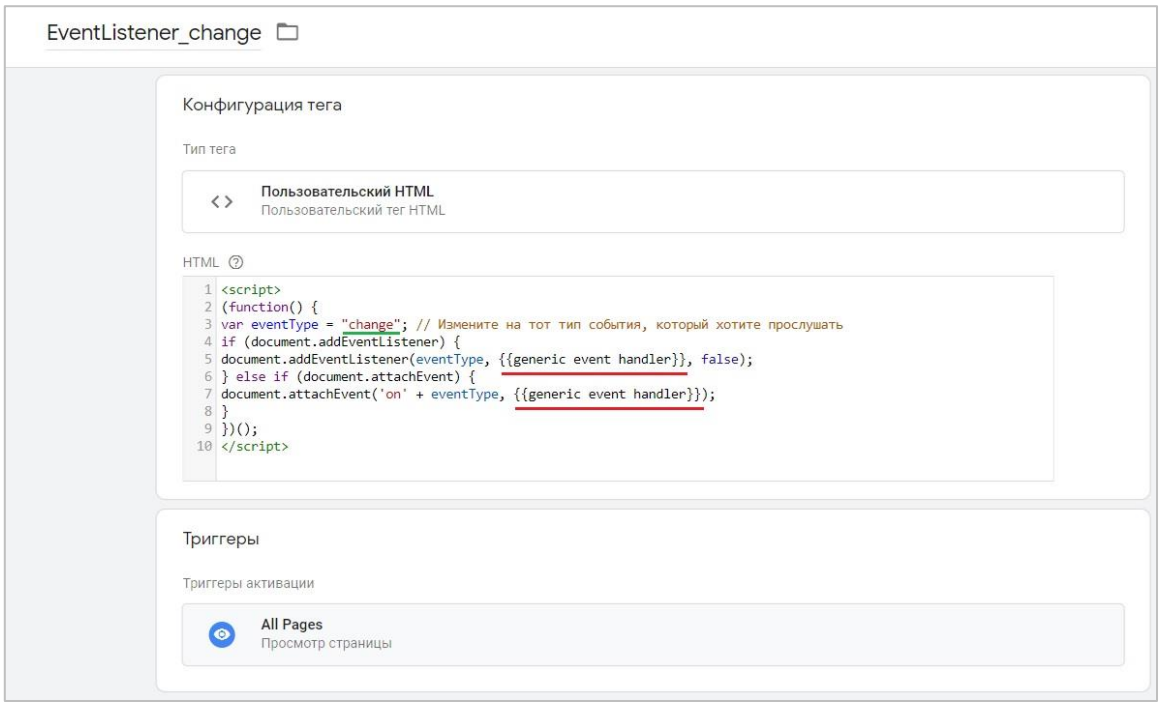


Рис. 678. Отслеживание события change

Триггер активации - **Все страницы (All Pages)**.

Хочется отметить событие **change**. Его особенность заключается в том, что оно вызывается после того, как пользователь отпустил ползунок. Событие не позволяет отследить все манипуляции, которые производит пользователь, пока выбирает нужное значение. В этом случае можно использовать событие **oninput**, которое создается и в промежуточных состояниях, пока пользователь двигает ползунок.

Примечание: при настройке следует обратить внимание на переменную **generic event handler**. Если вы изменяли название на предыдущем шаге, то и здесь необходимо указать собственное значение.

Создание тега прослушивания событий

В разделе **Триггеры** создаем событие типа **Пользовательское событие** с именем события **gtm.change**:

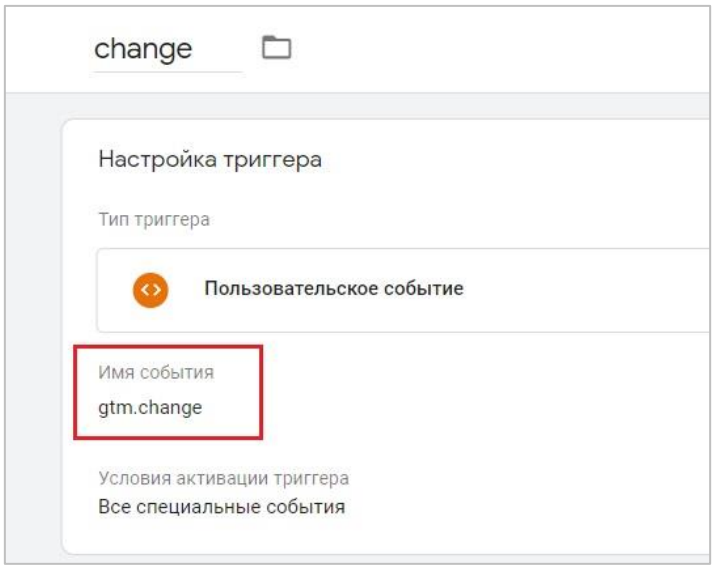


Рис. 679. Пользовательское событие gtm.change

Создание тега Google Аналитика - Universal Analytics

Все, что осталось, создать тег для передачи данных в Google Analytics. Переходим на вкладку **Теги** и создаем тег типа **Google Аналитика - Universal Analytics**

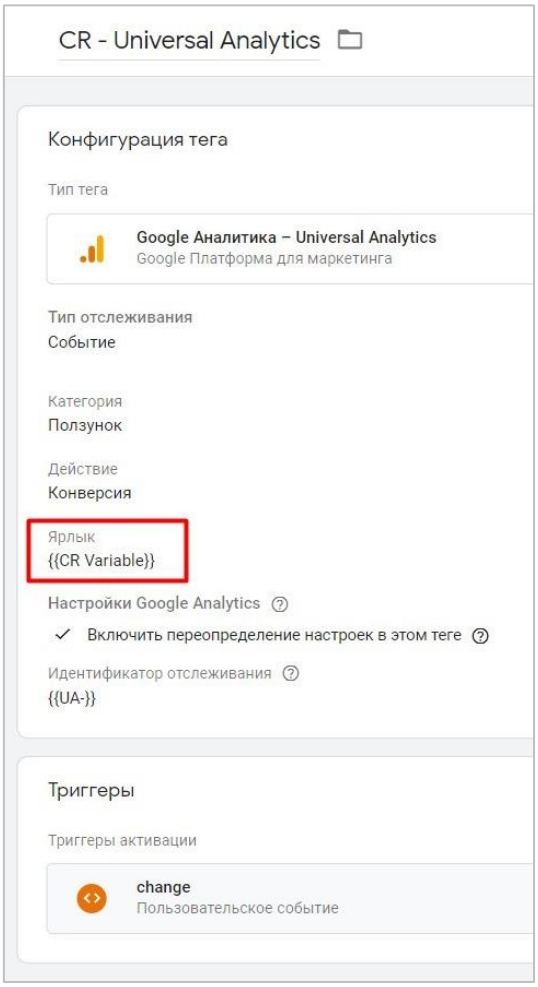


Рис. 680. Настройка тега

Категорию и **Действие** события можно задать произвольные. В **Ярлыке** я добавил передачу значение переменной **CR Variable**, которую настраивали на первом шаге, и которая будет передавать выбранное пользователем значение конверсии сайта.

Сохраняем настройки. Проверить корректность работы можно с помощью режима отладки:

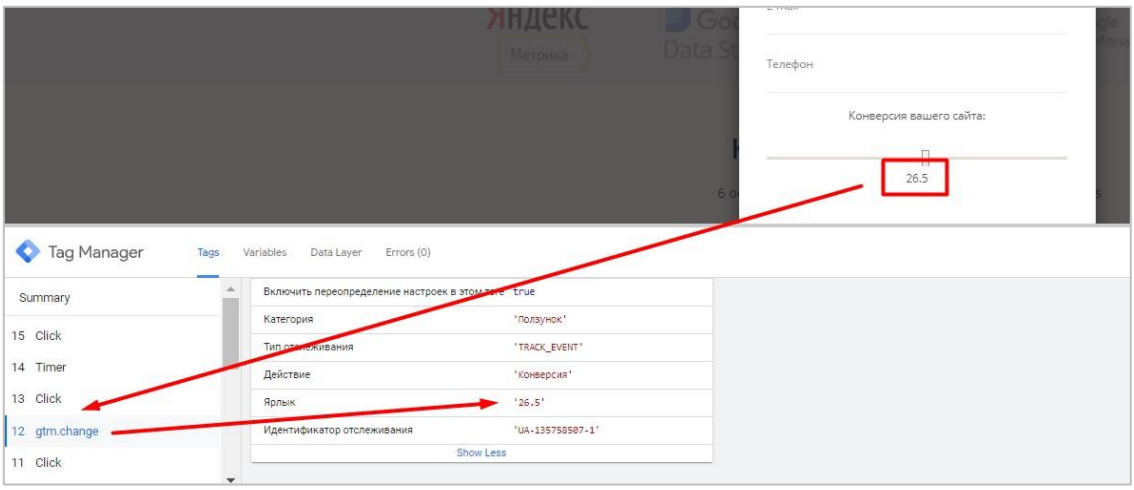


Рис. 681. Режим предварительного просмотра

Как видим, при изменении значения на ползунке в GTM срабатывает наше событие gtm.change, а оно, в свою очередь, активирует тег Universal Analytics и передает в показатель **Ярлык события** значение в формате строке (string). Если вам необходимо передавать числовое значение (number), в возвращаемой переменной используйте функцию **parseInt**:

```
function (){
var cr = document.querySelector('#rangeValue').innerText;
return parseInt(cr);
}
```

В отчете **В режиме реального времени** в Google Analytics события также отображаются:

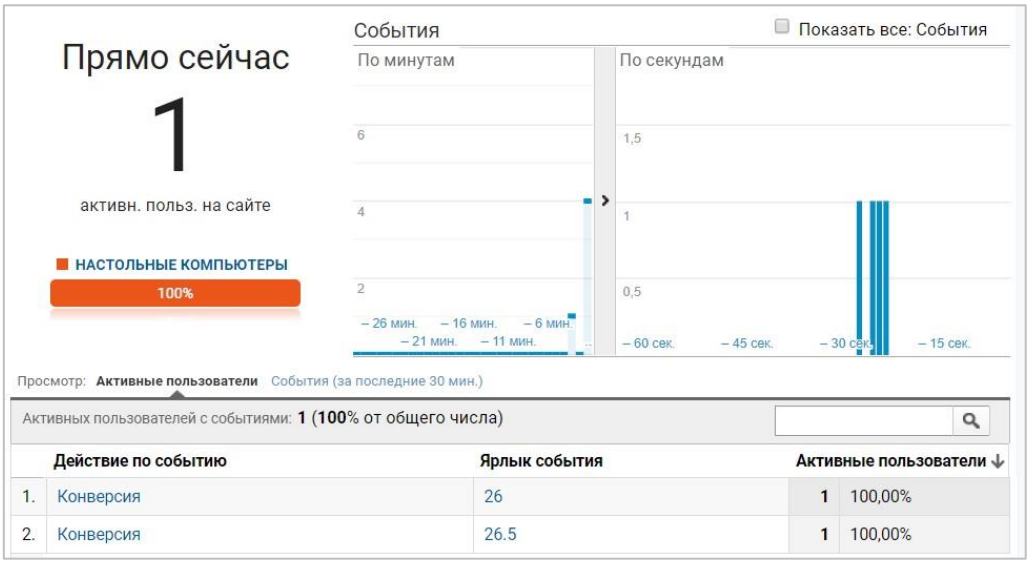


Рис. 682. Отчет В режиме реального времени

Через некоторое время все данные по событиям появятся в отчете **Поведение — События — Лучшие события**.

Отслеживание выбранного элемента из выпадающего списка

У вас на сайте есть форма, в которой присутствует выпадающий список? И вам хотелось бы фиксировать, что посетители выбирают в раскрывающемся поле при ее отправке? Давайте разберем пример отслеживания с помощью GTM и переменной уровня данных.

В качестве примера, по традиции, буду разбирать это на graphanalytics.ru. Я добавил дополнительное поле раскрывающегося списка (тег **<select>**), в котором пользователю при заказе бесплатного аудита необходимо указать текущий бюджет на рекламу.

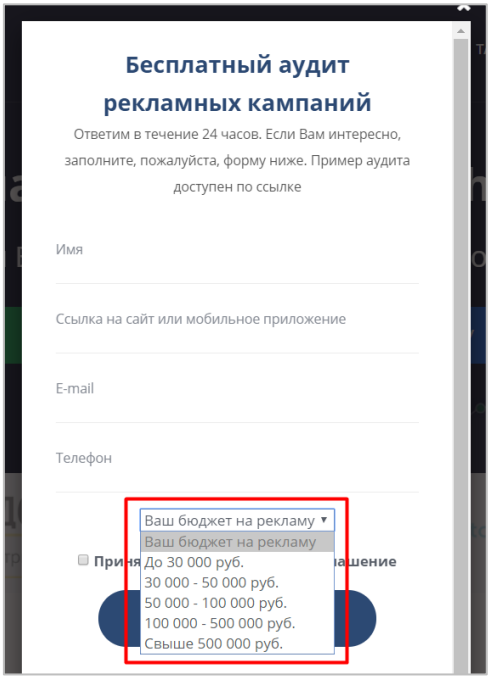


Рис. 683. Пример выпадающего списка

Именно эти данные мы и хотим отслеживать и передавать в Google Analytics в момент отправки формы.

Есть различные варианты решения этой задачи. Симо Ахава в своем блоге (см. приложение) предлагает несколько решений этой задачи:

- с помощью пользовательского тега HTML;
- с помощью пользовательской переменной типа **Собственный код JavaScript**;

В них идет обращение к списку с помощью идентификатора элемента (id) или CSS-селектора. Я воспользуюсь переменной уровня данных, чтобы извлечь значение выбранного элемента списка.

Для этого сделайте три простых шага:

1. Перейдите на сайт;
2. Откройте консоль разработчика (Клавиша F12, вкладка **Console** в Google Chrome);
3. Сделайте тестовую заявку на сайте.

После этого в консоли введите **dataLayer** и нажмите **Enter**. Появится список все прослушанных событий (от 0 до 3).

Рис. 684. dataLayer в консоли разработчика

Как вы уже знаете, Google Tag Manager по умолчанию передает на уровень данных набор значений **gtm.js** (когда GTM готов к работе), **gtm.dom** (когда готова модель DOM) и **gtm.load** (когда окно полностью загружено). В dataLayer – это элементы 0-2. Последний 3 – это событие **gtm.formSubmit** – отправка нашей формы со всеми значениями. Его мы и будем исследовать.

Примечание: чтобы в консоли в dataLayer у вас появилось событие отправки формы (**gtm.formSubmit)**, вы изначально должны настроить в GTM такое условие активации.

Нажмите на треугольник рядом с событием, чтобы увидеть, из чего оно состоит.

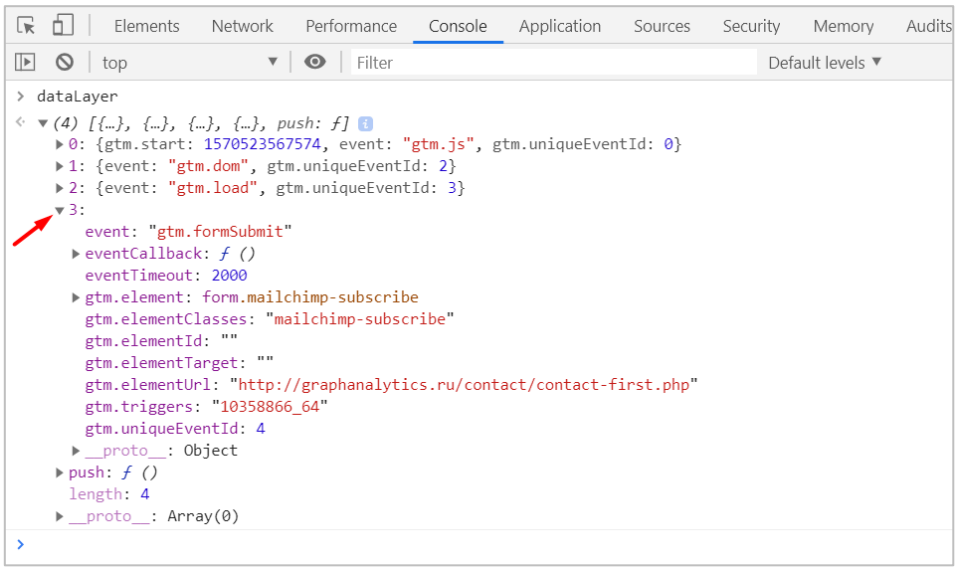


Рис. 685. Откройте событие gtm.formSubmit (индекс 3)

Нас интересует переменная **gtm.element**, поэтому нажмите еще раз на треугольник и провалитесь еще на один уровень ниже.

Рис. 686. Выберите gtm.element

В переменной представлены все поля нашей формы: Имя, Email, Телефон и т.д. 5 в списке идет **select**, наш выпадающий список. Проделаем ту же самую процедуру в третий раз. Раскрываем внутрь элемент с помощью треугольника:

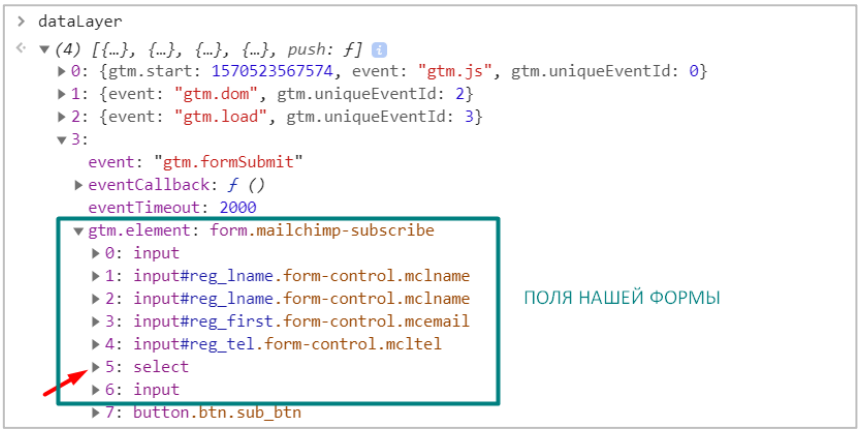


Рис. 687. Поле с индексом 5 - наш выпадающий список select

На этом уровне представлено большое количество атрибутов и свойств данного элемента. Нас интересует **value** (значение). Скроллим почти в самый низ (потому что они расположены по алфавиту) и находим *value*. В ней сохранилось выбранное значение из нашей тестовой заявки.

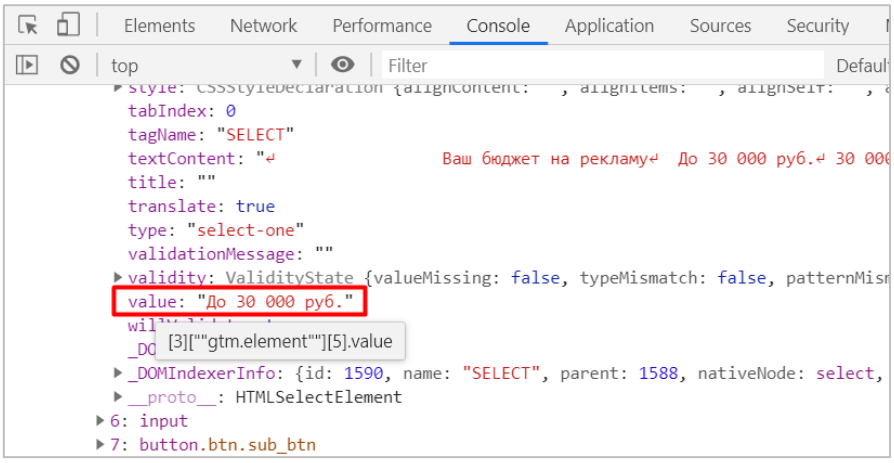


Рис. 688. Атрибут value - то, что нам нужно

Теперь нам необходимо сформировать нашу переменную уровня данных. Вспоминаем: мы раскрывали элемент с индексом 5, а его значение хранится в свойстве объекта **value**.

В Google Tag Manager можно использовать точечную нотацию для доступа к переменным ключам dataLayer, которые имеют точку в имени (например, **gtm.element**) или для доступа к свойствам объектов DOM (например, **gtm.element.dataset.name**). В моем примере переменная уровня данных будет выглядит так: **gtm.element.5.value**.

Возвращаемся в GTM и производим соответствующие настройки. Создаем пользовательскую переменную типа **Переменная уровня данных** с именем переменной **gtm.element.5.value**

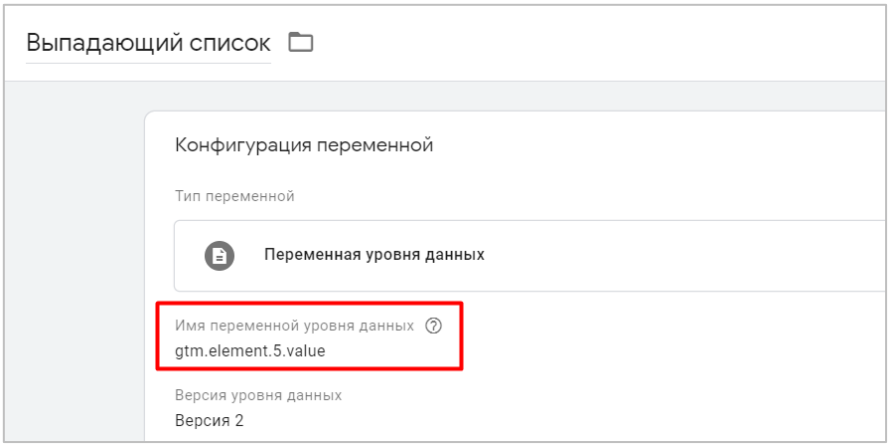


Рис. 689. Переменная уровня данных gtm.element.5.value

Сохраняем переменную и обновляем тег с отправкой формы. Я буду передавать в Google Analytics событие отправки формы, а значение из выпадающего списка помещу в **Ярлык события**.

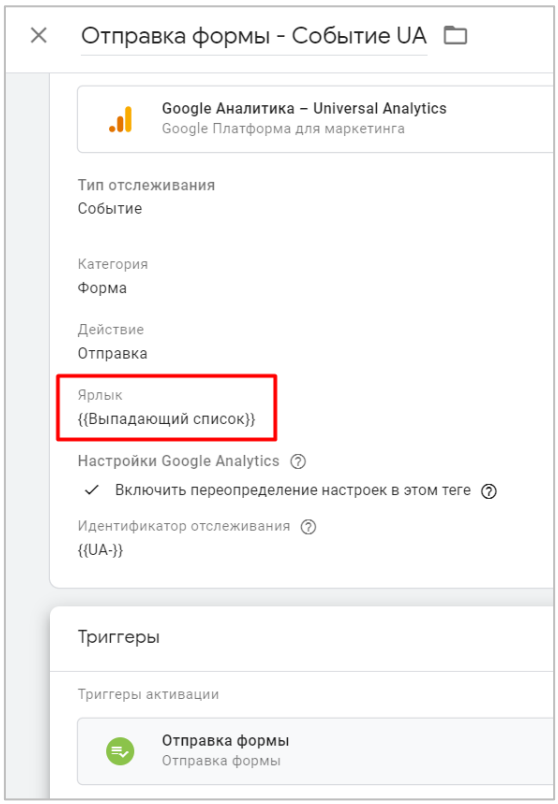


Рис. 690. Настройка тега

Сохраняем тег. Теперь с помощью режима предварительного просмотра мы можем проверить корректность настройки. Снова делаем тестовую заявку.

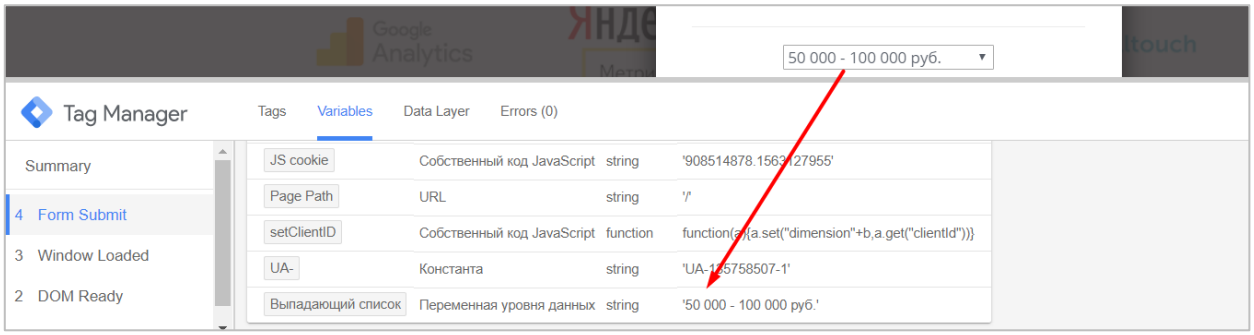


Рис. 691. Проверка в режиме отладки

Мы видим, что наша переменная заполнена правильно. Вы также можете проверить это в отчете Google Analytics **В режиме реального времени**:

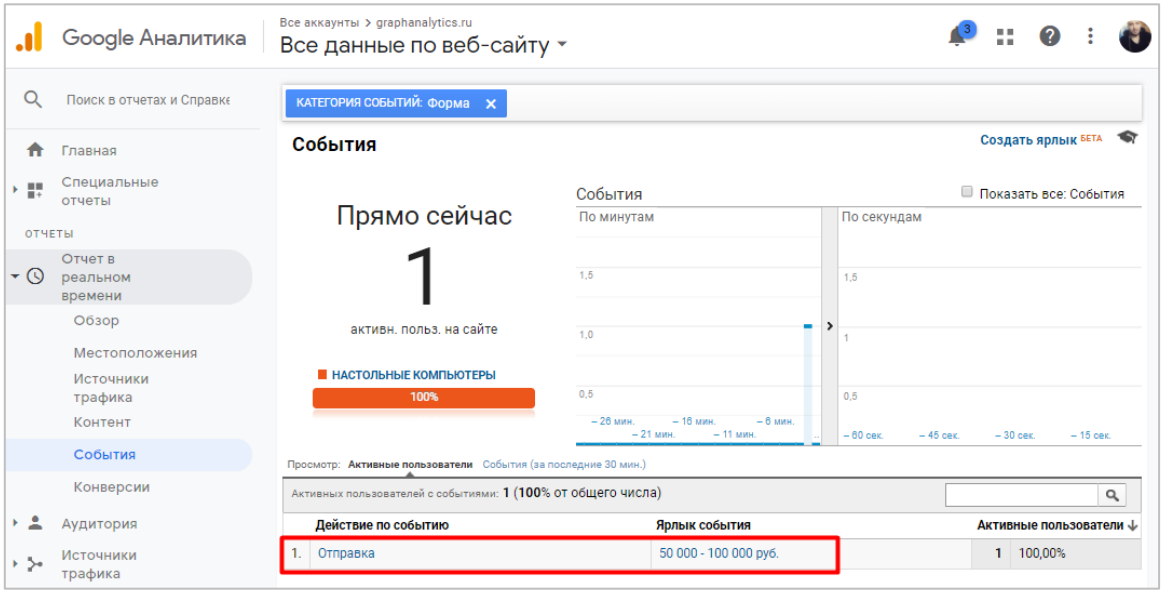


Рис. 692. Отчет В режиме реального времени

Все отслеживаемые события доступны в отчете **Поведение — События — Лучшие события**.

Существует альтернативный вариант отслеживания выпадающего списка. Как правило, у таких элементов присутствует собственный идентификатор (id), поэтому есть возможность зацепиться с помощью метода **document.getElementById**. Весь код для переменной типа **Собственный код JavaScript** выглядит так:

```
function () {
var s = document.getElementById("mainform-placement");
var selNum = s.options[s.selectedIndex].innerText;
return selNum;
}
```

Свойство **selectedOptions** возвращает индекс первого выбранного **<option>** элемента. Значение -1 указывает, что ни один элемент не выбран, а свойство **innerText** позволяет получать текстовое содержимое элемента.

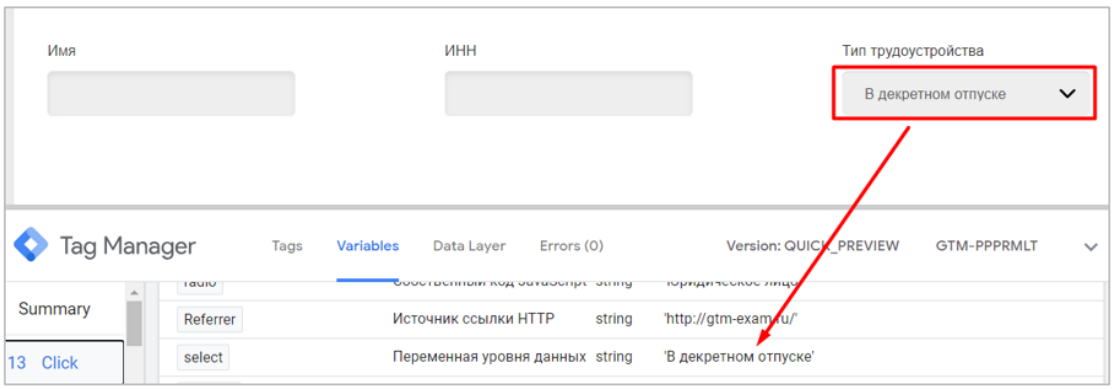


Рис. 693. Проверка в режиме предварительного просмотра

Отслеживание события клавиатуры keypress

В 90% случаев пользователи используют для совершений действий на сайте компьютерную мышь - когда отправляют форму, кликают по кнопке **Отправить**, когда совершают поиск, нажимают на кнопку **Найти** и т.д. В момент клика по данному элементу в GTM срабатывает событие **Click (gtm.click)**, благодаря которому мы можем настроить тег и передать эту информацию в инструменты веб-аналитики:

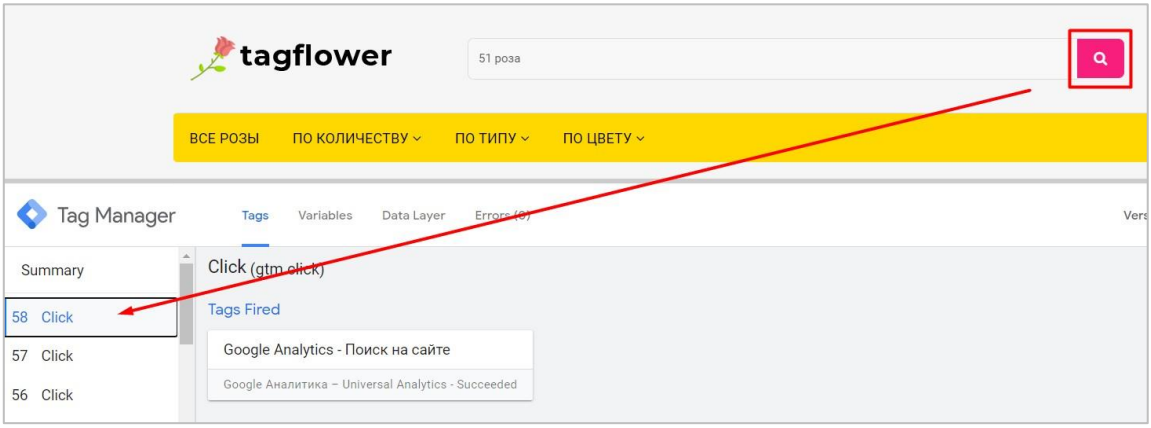


Рис. 694. Пример клика по поиску

Но что делать, когда пользователь вместо клика левой кнопкой мыши по элементу нажимает на клавиатуре **Enter**? В этом случае триггер Click не сработает, и данные о том, что пользователь совершил это действие, не отправятся в аналитику. На помощь придет браузерное событие клавиатуры **keypress**, которое срабатывает когда нажимается клавиша клавиатуры, которая создает символ. Это могут быть буквы, цифры, знаки пунктуации и т.д. Есть клавиши, которые не создают символы, например, **Alt, Shift, Ctrl**. Это клавиши модификаторы, они выполняют роль не ввода, а модификации объектов.

Для решения этой задачи необходимо создать тег типа **Пользовательский HTML** со следующим кодом:

```
<script>
document.querySelector('#search').addEventListener('keypress', function (e) {
if (e.key === 'Enter') {
dataLayer.push({"event": "enterClick"});
}
});
</script>
```

В GTM это выглядит так:

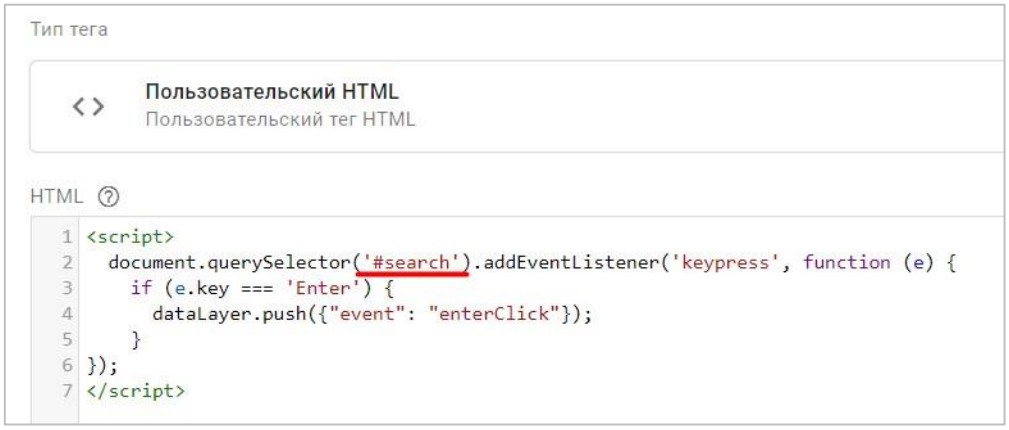


Рис. 695. Тег Пользовательский HTML

, где вместо **#search** необходимо добавить селектор того элемента, на котором хотите отследить нажатие клавиши **Enter**. Для моего примера это идентификатор (id) search:

Рис. 696. Идентификатор #search для строки поиска

Условие активации тега - **All Pages (Все страницы)**, поскольку в моем примере блок с поиском имеет сквозное размещение в шапке сайта.

Разберем код подробнее:

- **document.querySelector('CSS-селектор');** - метод возвращает первый элемент документа, который соответствует указанному селектору или группе селекторов;
- **addEventListener('keypress', function (e) {if (e.key === 'Enter') {dataLayer.push({"event": "enterClick"});** - создается метод addEventListener с событием keypress и функцией обработчика, в которой мы добавляем пользовательское событие **enterClick** в уровне данных (dataLayer) при условии, что строго была нажата клавиша Enter;
- **enterClick** - произвольное название события. Можете задать какой пожелаете.

Другими словами: при нажатии пользователем клавиши Enter внутри/на элемента/е search, будет срабатывать пользовательское событие **enterClick**, которое мы сможем использовать в качестве триггера GTM.

Сохраняем тег. Теперь настраиваем триггер типа **Пользовательское событие**. В нем указываем имя нашего события. В моем примере - **enterClick**:

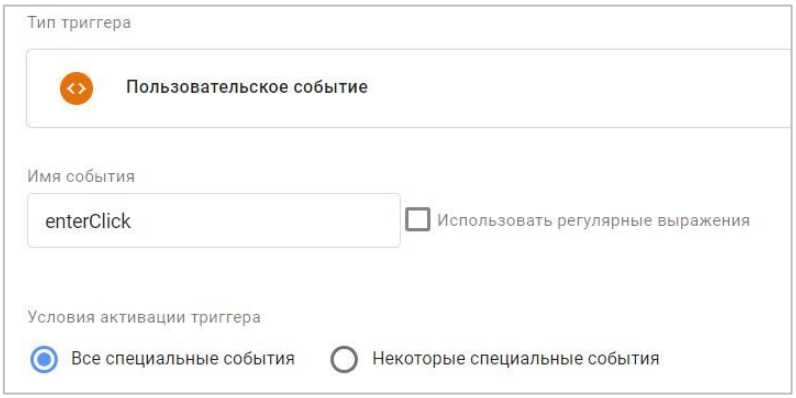


Рис. 697. Пользовательское событие enterClick

Сохраняем. Остается только добавить тег(и) для инструментов веб-аналитики. Для передачи этой информации в Google Analytics достаточно создать тег типа **Google Аналитика - Universal Analytics** с типом отслеживания **Событие**.

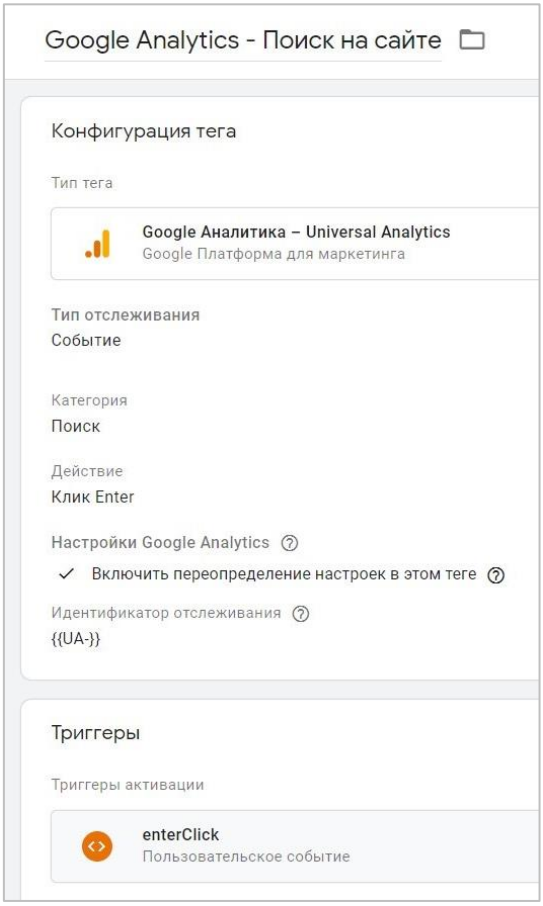


Рис. 698. Настройки тега Google Аналитика – Universal Analytics

В поля **Категория** и **Действие** вписываете произвольные значения. В качестве триггера активации добавляем наше **Пользовательское событие**, созданное на предыдущем шаге. Сохраняем настройки.

Проверить корректность отслеживания нажатия клавиши **Enter** с помощью режима предварительного просмотра Google Tag Manager будет крайне сложно, потому что после совершения события страница перезагрузиться и вся шкала событий обновиться. В этом случае мы можем воспользоваться расширением для браузера Google Chrome **Adswerve - dataLayer Inspector+** и консолью разработчика, в которой, при условии проставленной галочки **Preserve log**, будет сохранена вся история совершенных событий, и вы сможете увидеть какие события произошли до перезагрузки страницы.

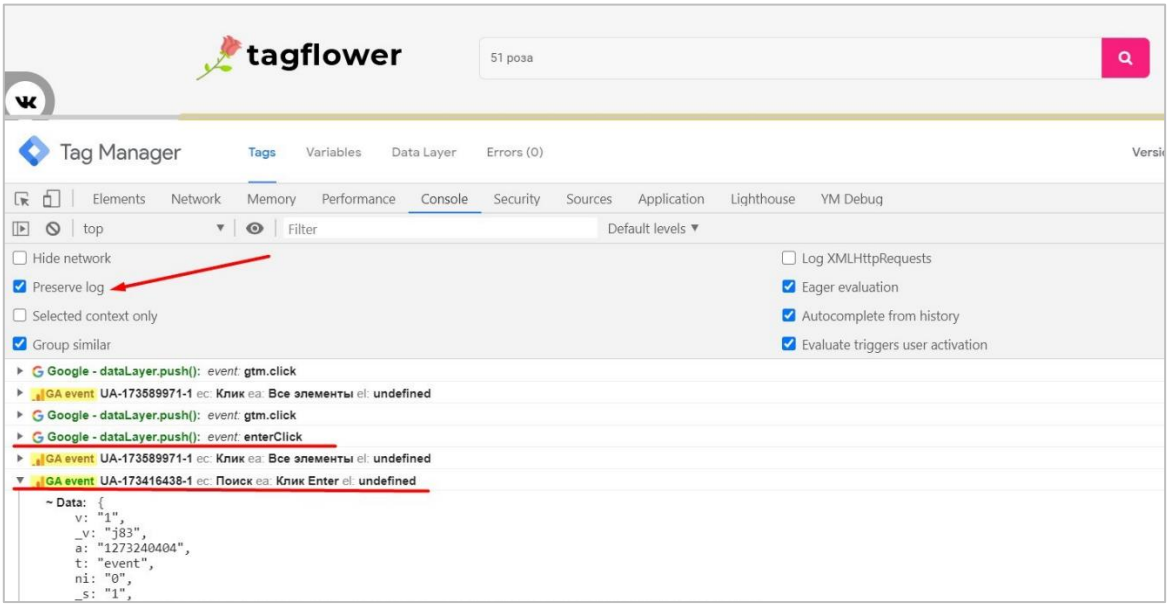


Рис. 699. Совершенные события enterClick и отправка данных в Google Analytics (галочка Preserve log)

Как видно из рисунка, после нажатия клавиши Enter было совершено 2 действия:

1. сработало событие **enterClick;**
2. активировался тег Google Analytics, который передал информацию в отчеты сервиса.

Данные в отчетах Google Analytics через некоторое время будут отображаться в разделе **Поведение – События – Лучшие события**, а **В режиме реального времени** вы можете увидеть данные по событиям мгновенно:

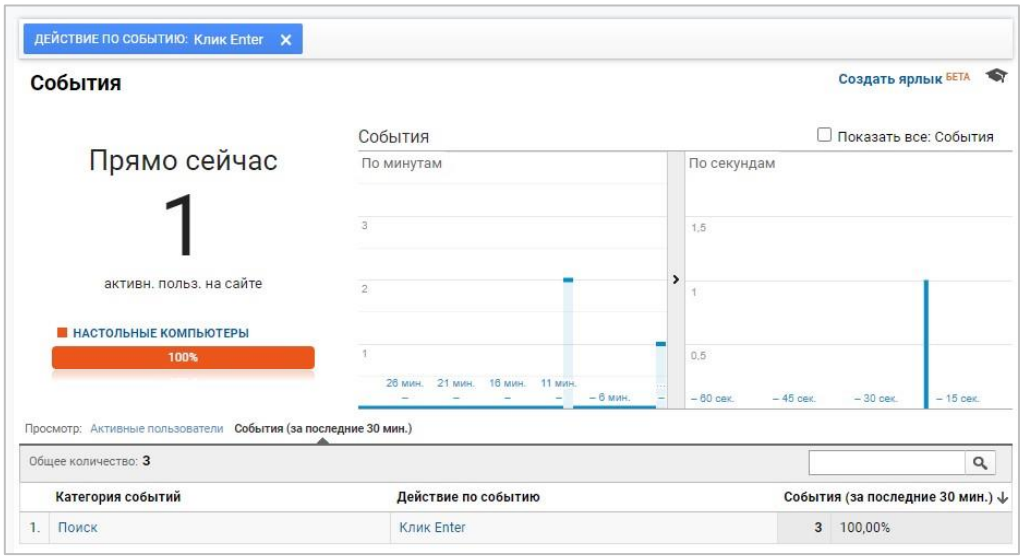


Рис. 700. Отчет В режиме реального времени

Отслеживание кликов с помощью dataLayer

При настройке Google Tag Manager иногда попадаются элементы, клики по которым нельзя просто так отследить, поскольку у них может не быть ни класса (.class), ни идентификатора (#id). А если мы перейдем в режим предварительного просмотра, то увидим там одно из таких значений:

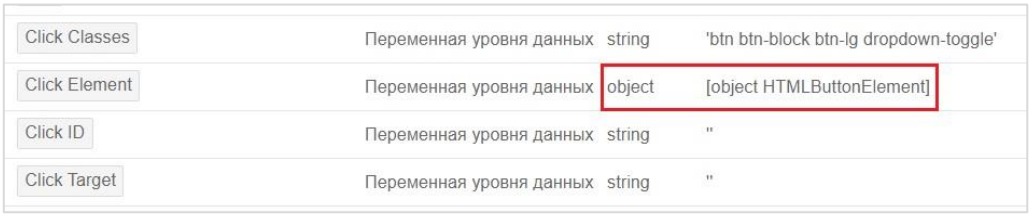


Рис. 701. Пример значения переменной [object HTMLButtonElement]

Почему так происходит? Давайте разберем пример отслеживания клика по кнопке **Товары** на моем демонстрационном сайте **techniqa.ru**:

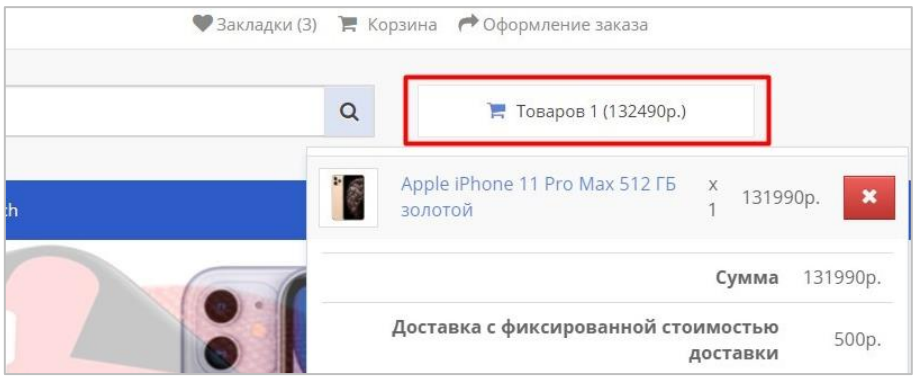


Рис. 702. Пример отслеживания кнопки Товары

Казалось бы, обычная задача и кнопка. Перед тем, как настроить тег и триггер для нее, мы можем воспользоваться триггером **Клик - Все элементы** и режимом отладки Google Tag Manager, чтобы определить атрибуты этого элемента.

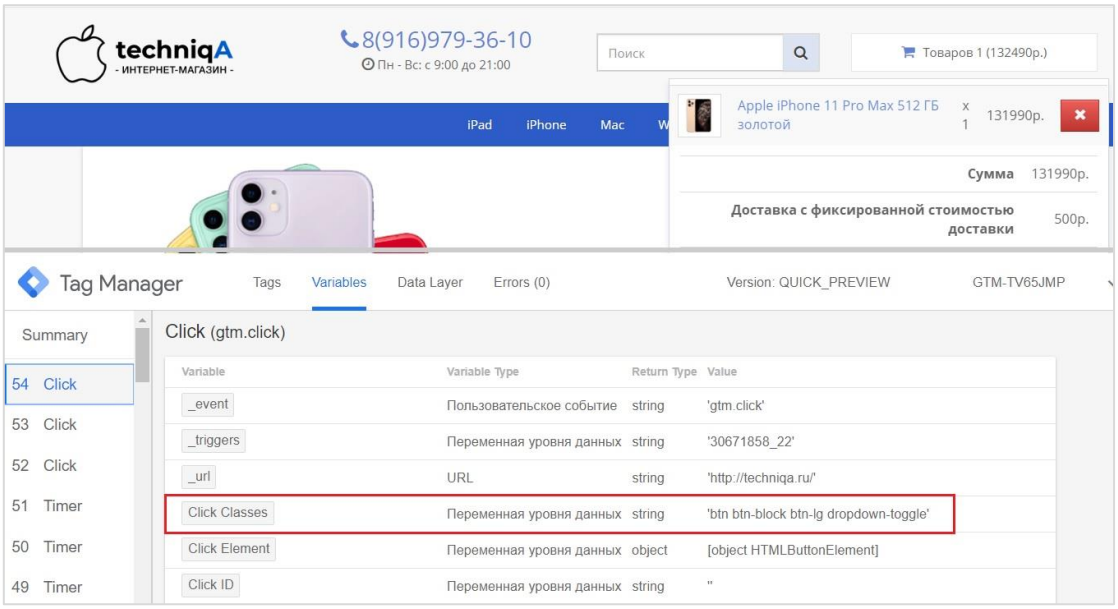


Рис. 703. Элемент имеет Click Classes

После клика мы видим, что у элемента есть **Click Classes**. Мы также можем определить CSS-селектор этой кнопки. Для этого отправляемся в консоль разработчика и инспектируем элемент, а затем выделив его, вызываем контекстное меню и копируем селектор.

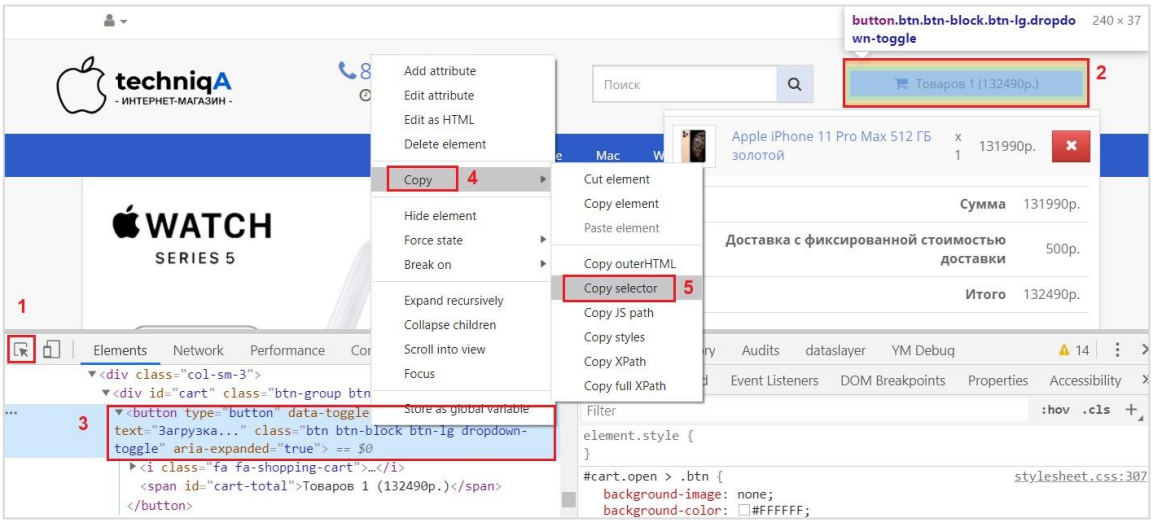


Рис. 704. Копирование селектора элемента

Получим CSS-селектор #cart > button . Чтобы проверить, что мы выбрали нужный элемент, воспользуемся расширением **CSS Selector Tester.**

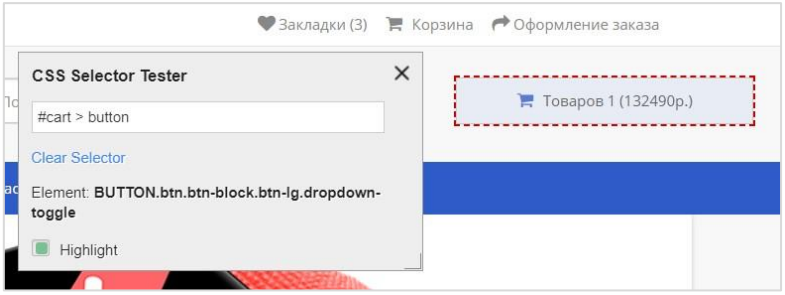


Рис. 705. Проверка селектора с помощью CSS Selector Tester

Элемент подсветился в нужной области, значит мы можем использовать его селектор при настройке триггера в GTM. Соответственно, справедлив и класс элемента, который мы также можем проверить с помощью расширения браузера:

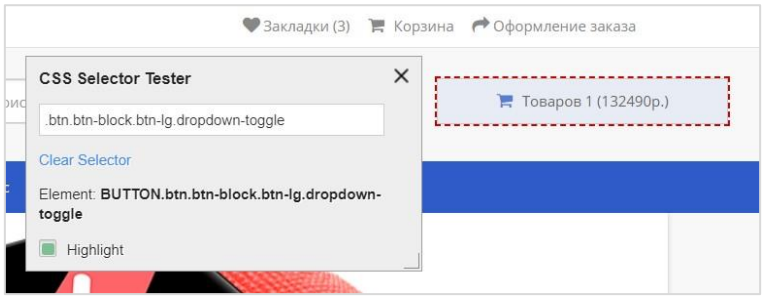


Рис. 706. Проверка класса элемента с помощью CSS Selector Tester

В результате мы получили 2 варианта отслеживания данной кнопки:

1. **Click Classes равно .btn.btn-block.btn-lg.dropdown-toggle**
2. **Click Element соответствует селектору CSS #cart > button**

Сразу скажу, что настроив любым из этих способов, наш тег и триггер могут срабатывать через раз или вовсе не срабатывать. Почему так? Все дело во вложенности. Как вы уже знаете из главы ранее, некоторые элементы содержат вложенную структуру. И когда нам кажется, что мы нашли тот самый верный атрибут элемента для отслеживания, на деле оказывается, что это не так.

Например, внутри кнопки **Товары** есть еще и маленькая иконка с корзиной, которая имеет свой класс и селектор. А что будет, если пользователь кликнет по ней? Тогда триггер и тег не сработают, и событие не будет передано в инструменты аналитики. Именно поэтому рекомендуется использовать **подстановочный знак (*, wildcard)** для точного определения. Условие **#cart > button, #cart > button *** для триггера позволило бы нам отследить все клики по данному элементу.

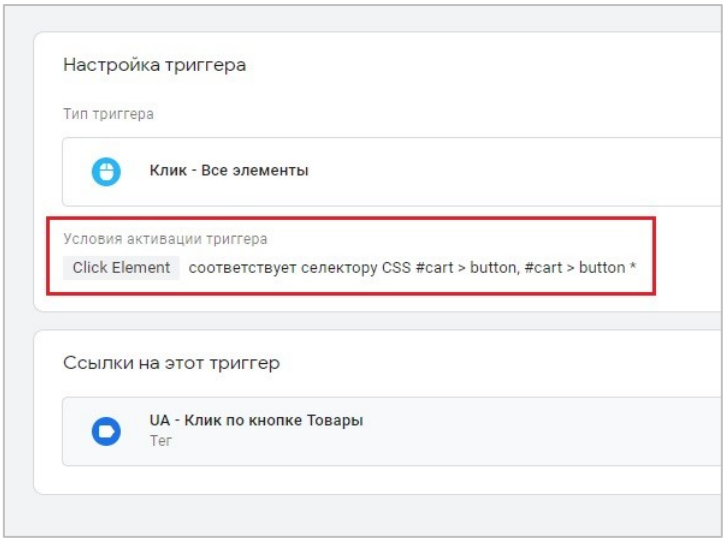


Рис. 707. Корректное условие активации триггера

Подстановочный знак очень сильно помогает. Но это еще не все способы отслеживания кликов в Google Tag Manager. Давайте вернемся к нашему примеру с кнопкой **Товары** в самом начале. В режиме отладки напротив встроенной переменной **Click Element** отображаются объекты **[object HTMLButtonElement]** и **[object HTMLSpanElement]**. Что это такое?

Чтобы понять, о чем Google Tag Manager пытается нам сообщить, необходимо вернуться к определению DOM. **Document Object Model (DOM)** — это объектная модель документа, древовидная структура, содержащая элементы, из которых состоит страница сайта. У всех на сайте есть элемент **body [HTMLBodyElement]**, неотъемлемая часть HTML-разметки, или, например, один или несколько элементов **div [HTMLDivElement]**. Обработчик кликов GTM всегда возвращает элемент, по которому кликнул текущий пользователь. Типов

объектов (их еще называют интерфейсы веб API) большое количество. Подробнее с ними вы можете ознакомиться в официальной документации Mozilla. (см. приложение).

Элементы DOM являются объектами. Их трудно представить в текстовом виде, поскольку они имеют свойства и методы, а также являются производными от родительских объектов и могут иметь дочерние узлы. Поэтому браузер пытается сообщить вам, какой тип объекта возвращается.

В этом случае мы можем использовать отслеживание кликов с помощью уровня данных (dataLayer):

```
<script>
var elem = document.querySelectorAll('CSS-селектор');
for (var i = 0; i < elem.length; i++) {
  elem[i].addEventListener("click", function(){
   dataLayer.push({"event": "customClick"});
});
}
</script>
```

Разберем код подробнее:

- **document.querySelectorAll('CSS-селектор');** - метод возвращает статический список, содержащий все найденные элементы документа, которые соответствуют указанному селектору (как определить - см. выше). В моем примере — это #cart > button . А **elem** — это переменная, которую мы объявляем и присваиваем ей это значение;
- **for (var i = 0; i < elem.length; i++)** - создаем цикл, которым будем искать все элементы в документе, которые соответствуют заданному селектору. Запускается столько раз, сколько есть элементов, не больше их количества (length);
- **elem[i].addEventListener('click', function(){ dataLayer.push({'event': 'customClick'});** - для каждого элемента создаем метод **addEventListener** с событием **click** и функцией обработчика, в которой мы прописываем пользовательское событие **customClick** в уровне данных;
- **customClick** - произвольное название события. Можете задать какой пожелаете.

Другими словами: при клике на любой элемент, содержащий заданный селектор, будет срабатывать пользовательское событие **customClick**, которое мы сможем использовать в качестве триггера GTM.

Код необходимо добавить в тег типа **Пользовательский HTML**. Триггер активации - **All Pages (Все страницы)**.

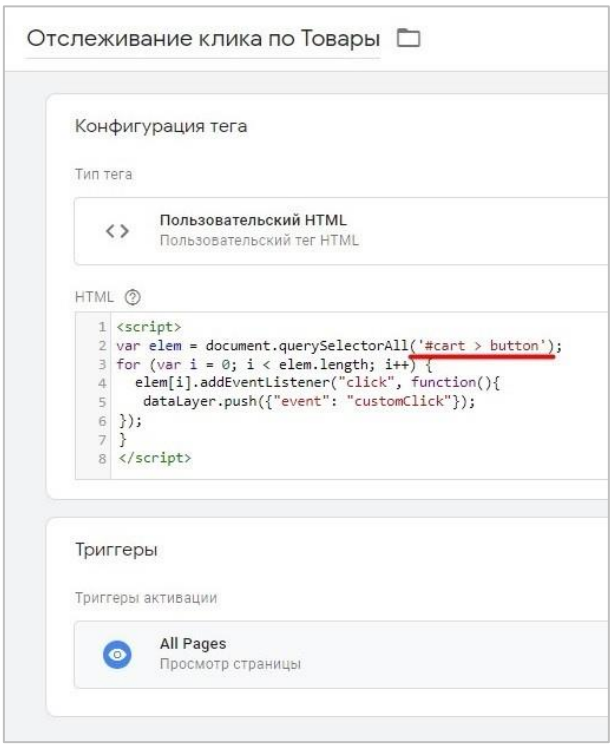


Рис. 708. Замените значение подчеркнутого селектора на собственное

Сохраняем тег. Теперь настраиваем триггер типа **Пользовательское событие**. В нем указываем имя нашего события. В моем примере - **customClick**:

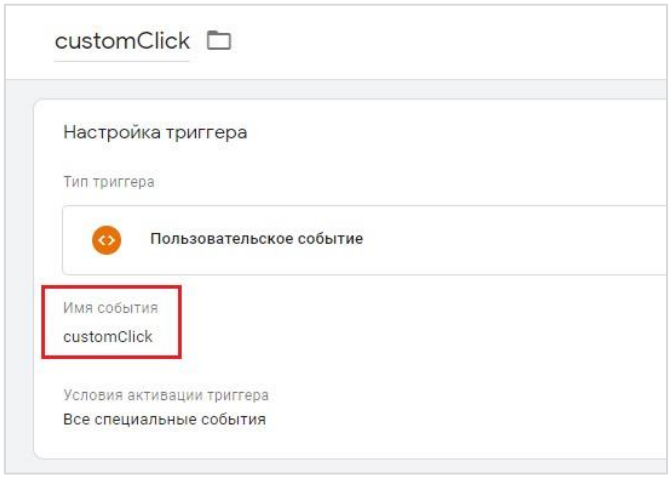


Рис. 709. Триггер Пользовательское событие customClick

Сохраняем. Остается только добавить тег(и) для инструментов веб-аналитики. Для передачи этой информации в Google Analytics достаточно создать тег типа **Google Аналитика - Universal Analytics** с типом отслеживания **Событие**.

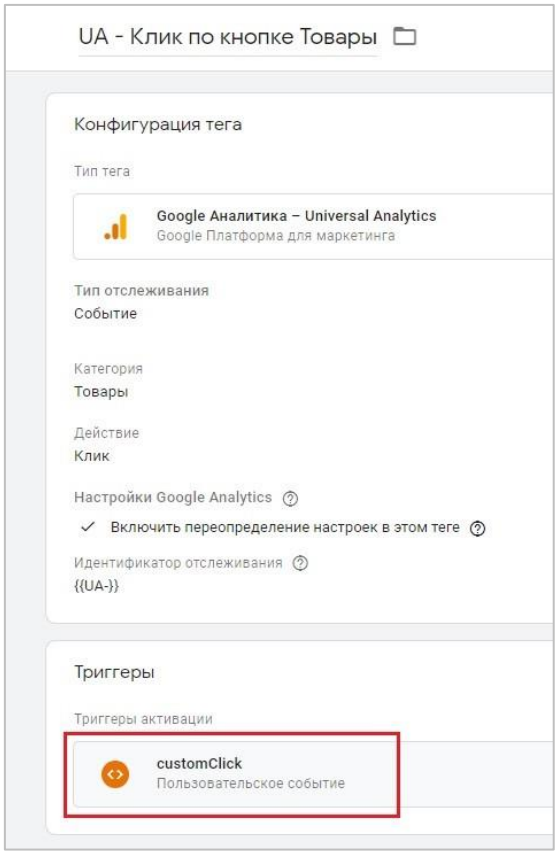


Рис. 710. Настройка тега

В поля **Категория** и **Действие** вписываете произвольные значения. Подробнее про события в Google Analytics читайте в этом материале (см. приложение). В качестве триггера активации добавляем наше **Пользовательское событие**, созданное на предыдущем шаге. Сохраняем настройки.

Проверить корректность отслеживания клика можно также с помощью режима отладки GTM. Запускаем Debug Mode и кликаем по нашему элементу.

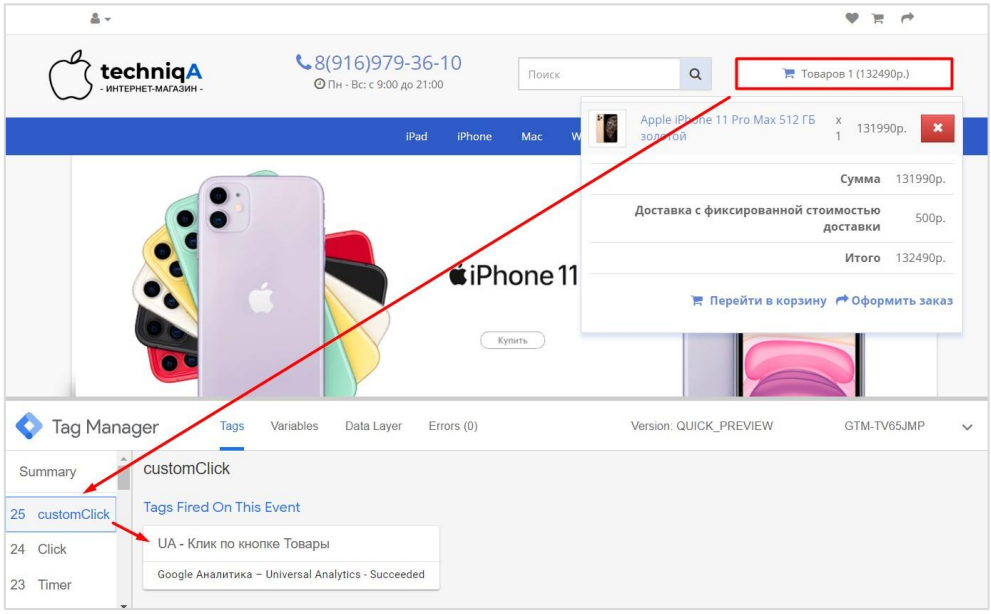


Рис. 711. Корректная настройка с активацией тега Google Analytics

Наше пользовательское событие срабатывает везде, где произошел клик, заданный CSS-селектору. Тег Google Аналитика активируется, в отчетах Google Analytics **В режиме реального времени** отображается наше событие:

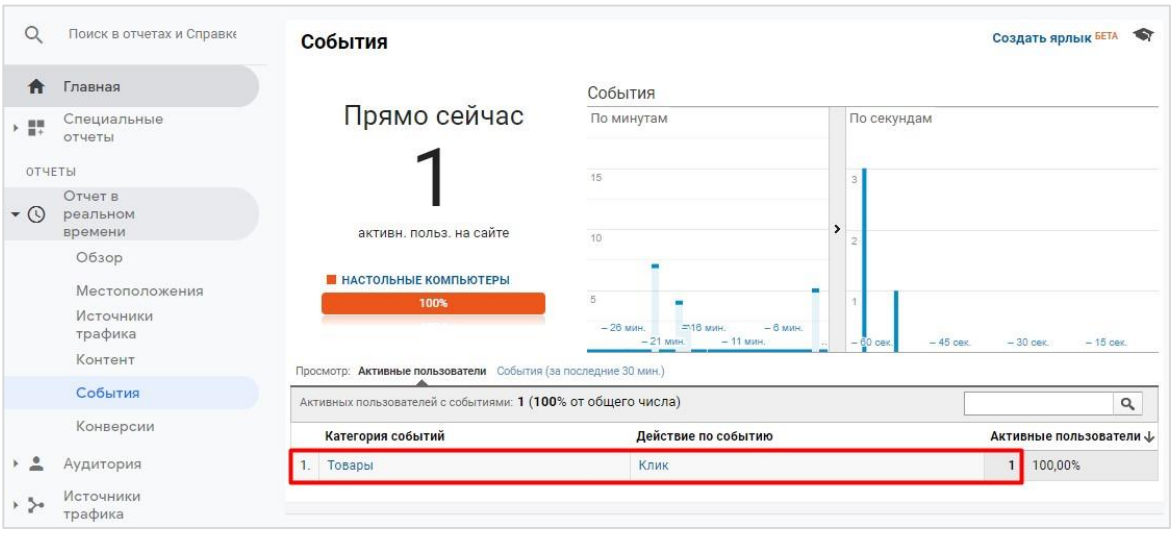


Рис. 712. Отчет В режиме реального времени

Данные в отчетах Google Analytics будут отображаться в разделе **Поведение – События – Лучшие события**.

Отслеживание скопированного текста

Разберем конкретную задачу: необходимо отправлять событие в Google Analytics каждый раз, когда пользователь копировал рефферальную ссылку на сайте:

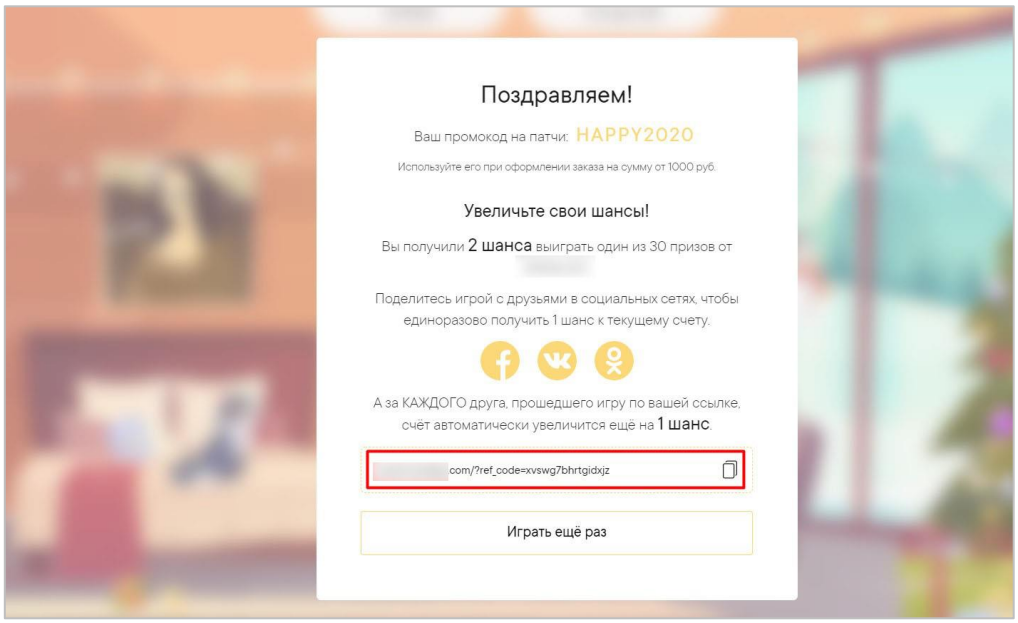


Рис. 713. Задача - активировать триггер на копирование рефферальной ссылки

Проблема заключалась еще и в том, что никак не удавалось настроить триггер активации, поскольку у данного элемента не было ни класса (class), ни идентификатора (id).

Рис. 714. Элемент без опознавательных знаков

В этом случае рекомендуется настраивать прослушивание событий с помощью метода **addEventListener**, о котором я писал ранее.

Для решения задачи воспользуемся кодом **Даниэля Карлбома (Daniel Carlbom)** из его публикации_(см. приложение), который необходимо добавить через Google Tag Manager в тег типа **Пользовательский HTML** тег.

```
<script>
function getSelectionText() {
  var text = '';
  if (window.getSelection) {
    text = window.getSelection().toString();
  } else if (document.selection && document.selection.type != 'Control') {
    text = document.selection.createRange().text;
  }
  return text;
}
document.addEventListener('copy', function(e){
  dataLayer.push({
    'event': 'textCopied',
    'clipboardText': getSelectionText(),
    'clipboardLength': getSelectionText().length
  });
});
</script>
```

В GTM это выглядит так:

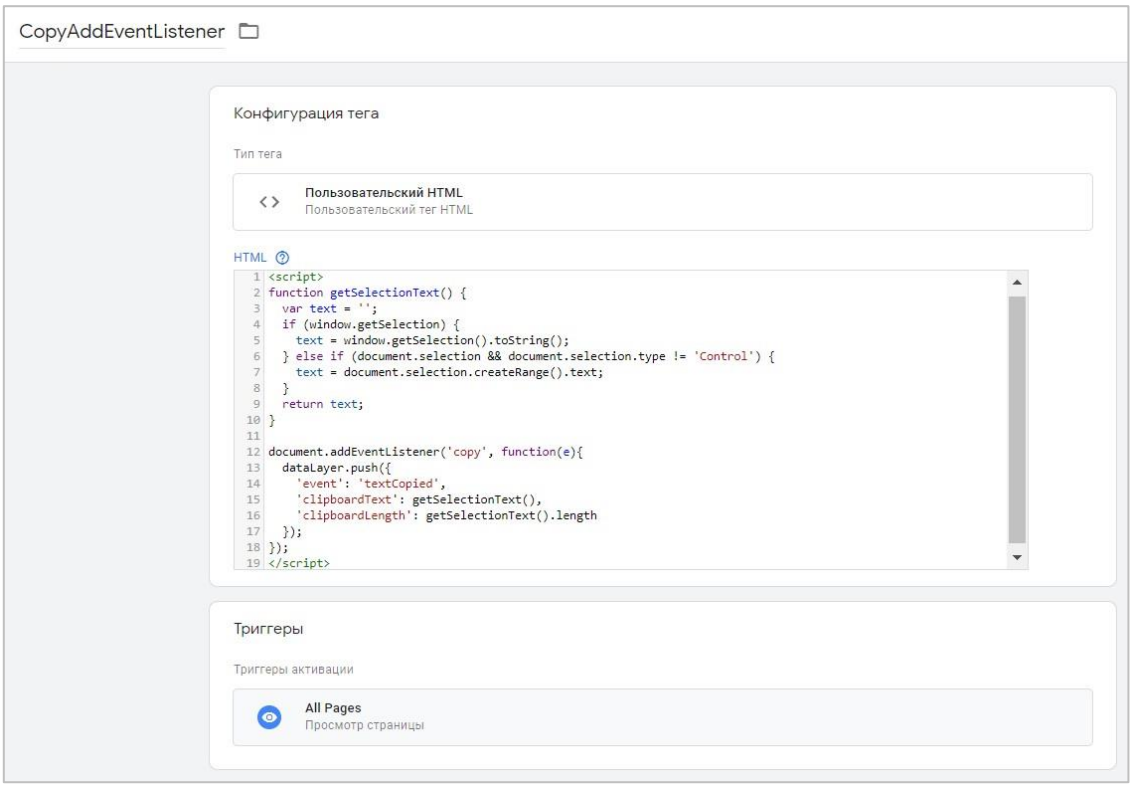


Рис. 715. Код addEventListener в Google Tag Manager

Триггер активации – **Все страницы (All Pages)**. Функция **getSelectionText** возвращает текст, который находится в буфере обмена. Метод **addEventListener** регистрирует обработчик события. В нашем случае – это **copy**. Событие срабатывает тогда, когда выделение было скопировано в буфер обмена. Затем полученную информацию мы помещаем в уровень данных (dataLayer), в которой есть 3 пары *ключ:значение*. Это:

- **event** – название пользовательского события (textCopied);
- **clipboardText** – фактическое содержимое скопированного текста;
- **clipboardLength** – длина скопированного текста (количество символов);

Теперь мы можем использовать все это при активации триггера и отправке значений в инструменты аналитики и т.д. Чтобы извлечь значение скопированного текста, создайте переменную типа **Переменная уровня данных** со значением **clipboardText:**

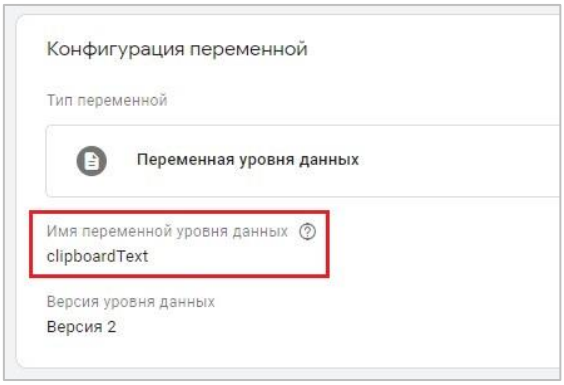


Рис. 716. Переменная уровня данных clipboardText

Для отслеживания количества символов создайте еще одну переменную со значением **clipboardLength**:

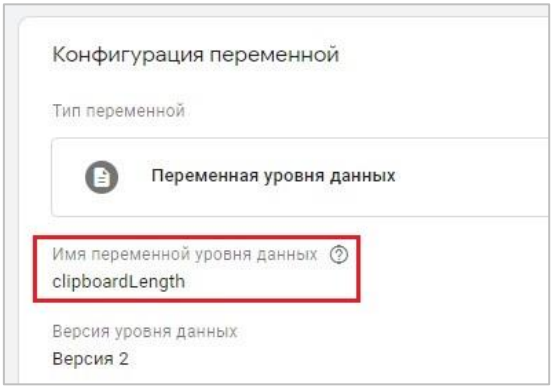


Рис. 717. Переменная уровня данных clipboardLength

Теперь необходимо создать триггер, который запускал бы пользовательское событие **textCopied:**

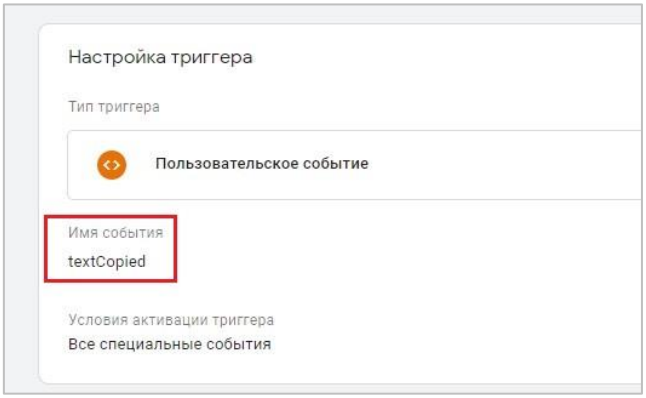


Рис. 718. Триггер Пользовательское событие

Все, что осталось сделать, это создать тег отправки полученных данных в Google Analytics. Для этого используется тег типа **Google Аналитика – Universal Analytics** с типом отслеживания **Событие**.

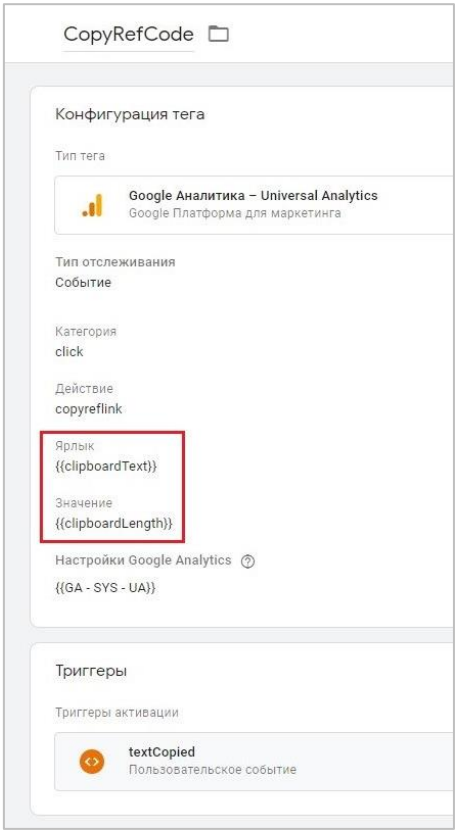


Рис. 719. Настройка тега

Триггер активации - пользовательское событие **textCopied**. В качестве **Категории** и **Действия** можно указать произвольные значения, в **Ярлык** я поместил переменную со скопированным текстом, а в **Значение** добавил длину текста. Можно также добавить и путь (Page Path), чтобы знать, с какой страницы пользователь копировал текст.

Сохранив все настройки, можно проверить корректность их работы с помощью режима отладки. При клике на значок копирования срабатывает событие **textCopied**:

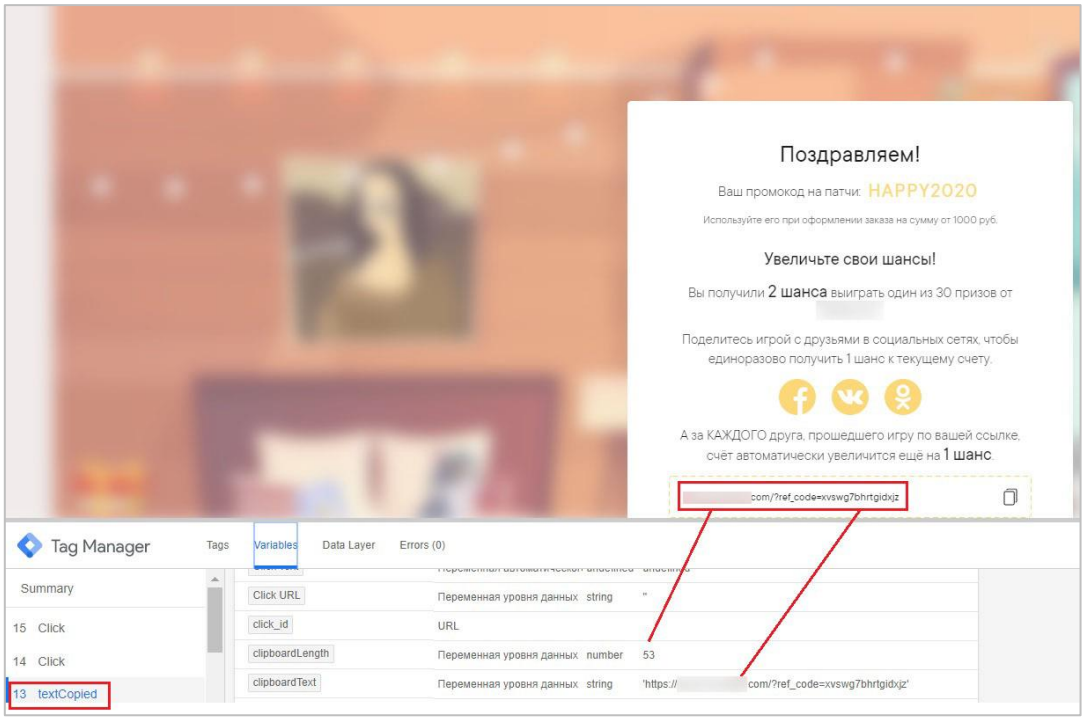


Рис. 720. Проверка события в режиме отладки

В созданные переменные информация передается. Тег Google Analytics активируется, в dataLayer содержимое тоже присутствует:

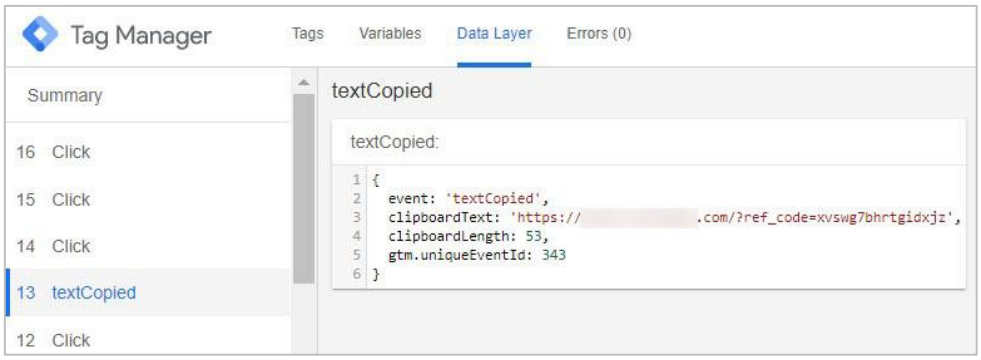


Рис. 721. Уровень данных

С помощью данного решения можно отслеживать копирование любого текста на странице, не только e-mail адреса. Например, на той же самой странице есть промокод, который мы также можем отследить. При копировании его пользователям будем срабатывать аналогичное событие, а в переменную **clipboardText** помещаться значение этого промокода:

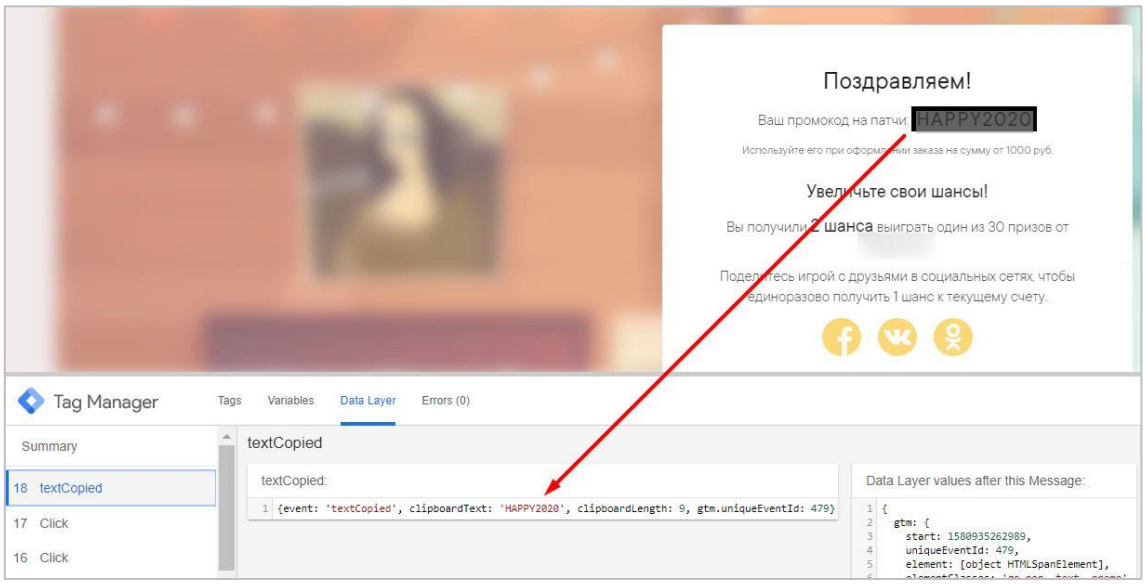


Рис. 722. Копирование промокода на странице

Данные в отчетах Google Analytics будут отображаться в разделе **Поведение – События – Лучшие события**.

Глава 10

Настройка e-commerce

Настройка стандартной электронной торговли

Когда речь заходит о настройке нестандартных событий, динамического ремаркетинга, User ID, электронной торговли или других не менее сложных процессов, у интернет-маркетолога возникает целый ряд вопросов. В этой публикации я постараюсь ответить на большую часть из них, чтобы у вас появилось окончательное понимание того, как все же настраивается электронная торговля для Google Analytics и Яндекс.Метрики.

Если вы никогда не слышали об электронной торговле, рекомендую прочитать эту статью (см. приложение) всего e-commerce настраивают для интернет-магазинов, чтобы отслеживать количество транзакций и доход от продажи товаров с привязкой к различным источникам трафика. В этой публикации (см. приложение) я описал способ отслеживания и для обычной посадочной страницы (LP, Landing Page).

Типы электронной торговли

В Google Analytics существует два основных типа электронной торговли:

- **стандартная электронная торговля (Standard Ecommerce);**
- **расширенная электронная торговля (Enhanced Ecommerce, EE).**

Стандартная электронная торговля:

- появилась раньше (в библиотеке *ga.js*), чем расширенная (*analytics.js*);
- ограничена количеством предоставляемых отчетов;
- не такая гибкая по сравнению с EE, но легче в настройке.

Список отчетов, доступных в стандартной версии:

- **Обзор:** данные по основным показателям – доход, транзакции, средняя стоимость заказа, коэффициент транзакции и т.д.
- **Эффективность товаров:** данные по к каждому отдельному товару – доход, количество покупок, средняя цена, идентификатор продукта, категория продукта и т.д.
- **Эффективность продаж:** доход с разбивкой по ID заказа и дополнительным показателям – налоги, стоимость доставки, сумма возврата и т.д.

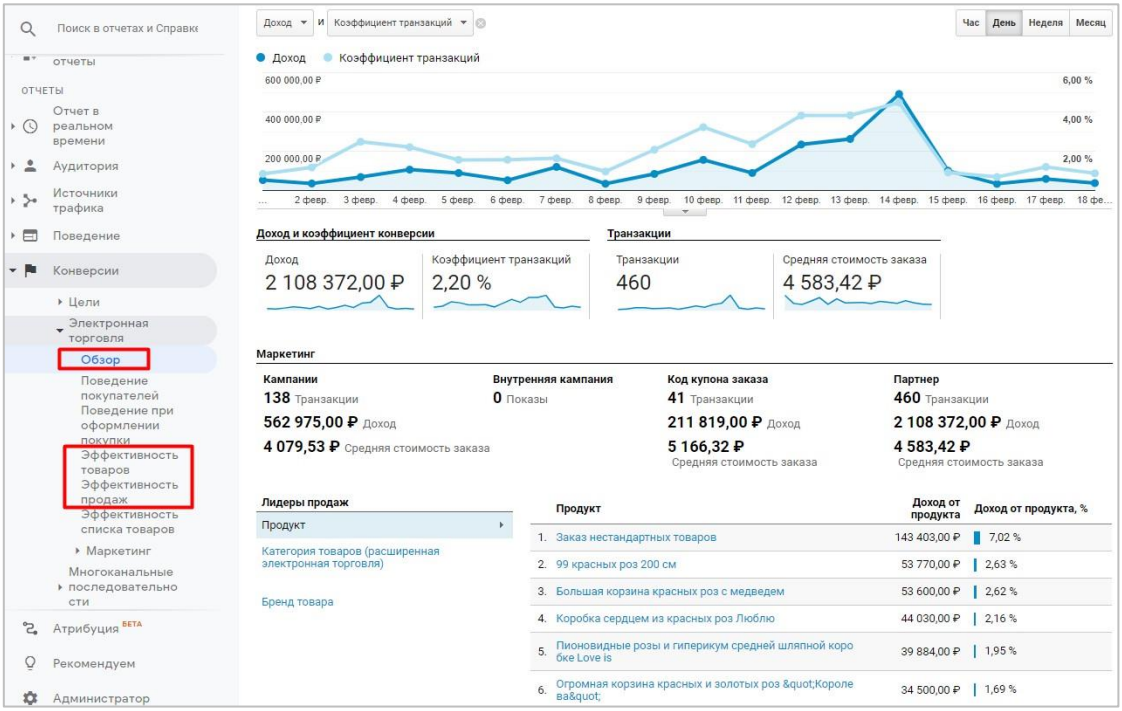


Рис. 723. Список доступных отчетов в стандартной электронной торговле

Если для расширенной электронной торговли доступно до 9 уровней отслеживания действий (показы товаров, клики по товарам, показы сведений о товарах, показы промоакций, клики по промоакциям, оформление покупки, покупки, возвраты), то для работы стандартной электронной торговли достаточно настроить **ТОЛЬКО** покупки (транзакции).

Какой тип электронной торговли выбрать?

Если вы хотите видеть только данные о продажах, понимать, насколько эффективно продается каждый товар, какие источники трафика генерируют продажи, то достаточно настроить *стандартную электронную торговлю*. Если вы хотите видеть не только покупки, но и пути пользователей от этапа к этапу на вашем сайте (какое количество пользователей добавило товар в корзину, сколько из них перешло на шаг оформления заказа, а какая часть из этой дошедших совершила покупку), строить по ним сегменты, запускать на них ремаркетинговые кампании, то тогда вам следует выбрать *расширенную электронную торговлю*.

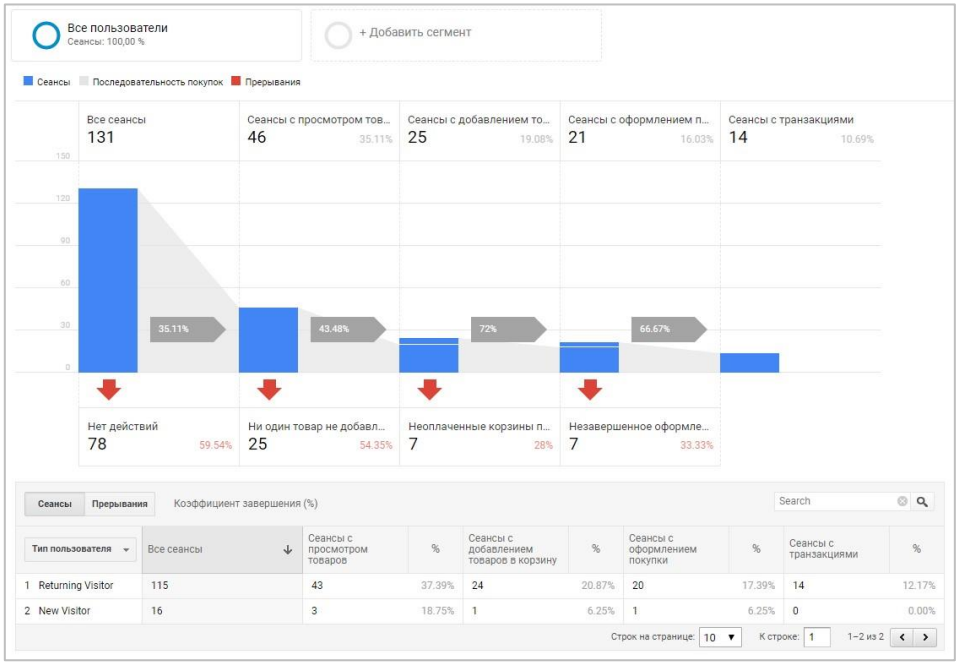


Рис. 724. Отчет Поведение покупателей (Enhanced Ecommerce)

Внедрить функционал расширенной электронной торговли гораздо сложнее, для настройки требуется больше времени и действий разработчика. Поэтому при выборе типа электронной торговли не стоит забывать и об экономической составляющей.

По сравнению с Enhanced Ecommerce, стандартная электронная торговля не требует такого большого количества времени, чтобы успешно внедрить отслеживание продаж.

Как настроить электронную торговлю?

Для внедрения на сайт стандартной электронной торговли можно использовать два метода настройки:

1. через код отслеживания;
2. с помощью Google Tag Manager.

Ниже будет рассмотрен второй способ (через Google Tag Manager), поскольку он является наиболее простым и не всегда требует сторонней помощи.

Кто настраивает электронную торговлю?

Обычно настройку электронной торговли руководство поручает интернет-маркетологу, который ведет и рекламные кампании, и имеет доступы к счетчикам веб-аналитики (Google Analytics и Яндекс.Метрики). Однако это не всегда так. Настройка любого типа электронной торговли подразумевает под собой работу с кодом, который необходимо разместить на странице оформленного заказа (в случае стандартной реализации) или на нескольких (при настройке расширенной электронной торговли). Такие коды вы увидите и в документации Google, и в любой статье в интернете, которая посвящена этой теме.

Вот один из примеров отслеживания транзакций для библиотеки gtag.js, событие **purchase (покупка):**

```
gtag('event', 'purchase', {
  "transaction_id": "24.031608523954162",
  "affiliation": "Google online store",
  "value": 23.07,
  "currency": "USD",
  "tax": 1.24,
  "shipping": 0,
  "items": [
    {
      "id": "P12345",
      "name": "Android Warhol T-Shirt",
      "list_name": "Search Results",
      "brand": "Google",
      "category": "Apparel/T-Shirts",
      "variant": "Black",
      "list_position": 1,
      "quantity": 2,
      "price": '2.0'
    },
    {
      "id": "P67890",
      "name": "Flame challenge TShirt",
      "list_name": "Search Results",
      "brand": "MyBrand",
      "category": "Apparel/T-Shirts",
      "variant": "Red",
      "list_position": 2,
      "quantity": 1,
      "price": '3.0'
    }
  ]
});
```

Этот код нельзя просто взять из документации и вставить к себе на сайт. Чтобы данные корректно передавались в Google Analytics, необходимо для каждой переменной (id, name, list_name, brand, price и т.д.)

выводить собственное значение. Причем извлечение данных для интернет-магазинов на различных CMS-платформах может быть реализовано по-разному. Для 1С-Битрикс своя, для OpenCart своя, для WordPress своя и т.д.

Поэтому когда говорят о настройке электронной торговли, то подразумевается следующее - человек, который настраивает электронную торговлю, должен понимать как:

- сформировать массив с данными;
- сделать это на конкретном движке интернет-магазина;
- извлечь эти данные и передать в инструменты веб-аналитики.

Не всегда маркетолог умеет работать с кодом. Даже не каждый разработчик, хорошо разбирающийся в каком-то одном движке, согласиться выполнять эту задачу для интернет-магазина на другой платформе. Поэтому настройку электронной торговли (стандартной или расширенной) необходимо выполнять как минимум в паре: *маркетологу* и *разработчику*. Первый подробно и понятно составляет техническое задание для второго, разработчик его внедряет, а в конце маркетолог проверяет корректность сделанной работы.

Но реальность такова, что заказчик не имеет в штате разработчика под эту задачу. Тогда снова есть несколько вариантов:

1. купить для своей CMS-платформы готовое решение (модуль/плагин);
2. нанять удаленного специалиста, который за $ настроит электронную торговлю;
3. настроить маркетологу своими силами через Google Tag Manager;

Почему в интернете нет конкретного руководства по настройке?

Во-первых, все интернет-магазины индивидуальны. У кого-то сайт на конструкторе, к которому может не быть доступа к файлам на изменения, у кого-то коробочное решение, кто-то пишет движок с нуля и т.д. Во-вторых, даже в рамках одного интернет-магазина настройку электронной торговли можно реализовать по-разному (напрямую добавляя код отслеживания или через Google Tag Manager), формировать массив данных через уровень данных (dataLayer), или использовать собственный код JavaScript. В-третьих, описать все способы настройки довольно сложно. Получается очень много различных комбинаций.

Настроив электронную торговлю для Google Analytics, она будет работать и для Яндекс.Метрики?

Да, если настройка в обоих случаях реализована с помощью контейнера данных (dataLayer). В официальной документации Яндекса об этом написано следующее (см. приложение):

Имя контейнера данных и структура вкладываемых в него Ecommerce-объектов соответствует аналогичным сущностям в Google Analytics Enhanced Ecommerce. Это означает, что если вы уже настроили отправку данных в Google Analytics Enhanced Ecommerce, в том числе через Global Site Tag, и включили Ecommerce в Яндекс.Метрике, последняя также начнет собирать данные.

Хочу отметить, что в справке речь идет о расширенной электронной торговли. А мы в этой главе говорим о Standard Ecommerce.

Как настроить стандартную электронную торговлю?

Перейдем к пошаговому плану настройки стандартной электронной торговли с помощью Google Tag Manager. Пошаговый алгоритм:

1. включить отчеты электронной торговли в Google Analytics;
2. отправить данные о транзакции на уровень данных;
3. отправить данные о совершенной транзакции в Google Analytics (через Google Tag Manager);
4. проверить корректность передачи данных.

Разберем каждый пункт более подробно.

1. Включить отчеты электронной торговли в Google Analytics

В настройках Google Analytics, на уровне представления, перейдите в раздел Настройки электронной торговли и Включите отслеживание электронной торговли.

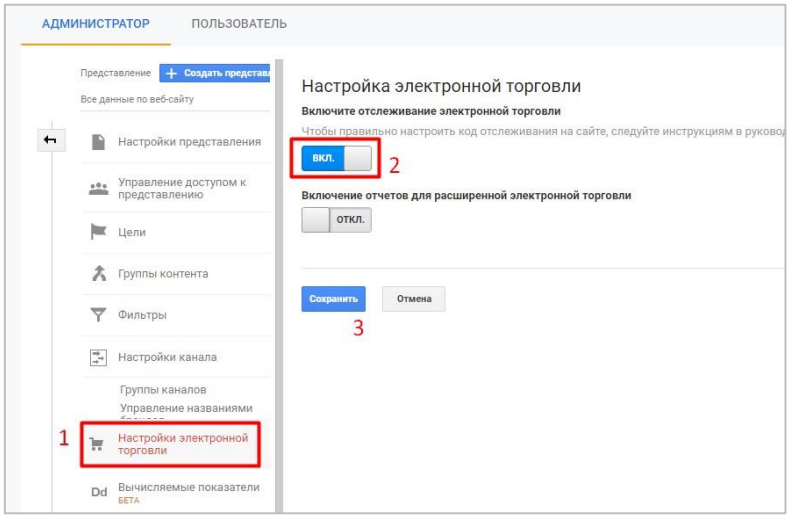


Рис. 725. Включение отслеживания электронной торговли

Сохраните настройки.

Отправить данные о транзакции на уровень данных

Чтобы настроить отслеживание электронной торговли Google Analytics с помощью диспетчера тегов Google, необходимо передать данные о транзакции на уровень данных. Затем в Google Tag Manager мы создаем правила, при которых покупка считается успешной, а после с помощью тега Universal Analytics передаем сведения о транзакции из уровня данных в Google Analytics.

О том, как настроить отслеживание электронной торговли, написано в официальной справке Google (см. приложение). Нет ничего лучше, чем следовать оригинальному источнику. Все сведения о транзакциях должны передаваться через уровень данных.

Пример кода выглядит следующим образом:

```
<script>
window.dataLayer = window.dataLayer || []
dataLayer.push({
   'transactionId': '1234',
   'transactionAffiliation': 'Acme Clothing',
   'transactionTotal': 38.26,
   'transactionTax': 1.29,
   'transactionShipping': 5,
   'transactionProducts': [{
       'sku': 'DD44',
       'name': 'T-Shirt',
       'category': 'Apparel',
       'price': 11.99,
       'quantity': 1
   },{
       'sku': 'AA1243544',
       'name': 'Socks',
       'category': 'Apparel',
       'price': 9.99,
       'quantity': 2
   }]
});
</script>
```

Существует два типа данных со своими обязательными переменными и типом:

1. Данные о транзакции (**transactionId, transactionTotal**);

Данные о транзакции

Имя переменной	Описание	Тип
transactionId (обязательная переменная)	Уникальный идентификатор транзакции	Строка
transactionAffiliation (необязательная переменная)	Партнер или магазин	Строка
transactionTotal (обязательная переменная)	Общая сумма транзакции	Число
transactionShipping (необязательная переменная)	Стоимость доставки для данной транзакции	Число
transactionTax (необязательная переменная)	Сумма налога для данной транзакции	Число
transactionProducts (необязательная переменная)	Список приобретенных товарных единиц для данной транзакции	Массив объектов товаров

Рис. 726. Обязательные переменные - transactionId, transactionTotal

2. Данные о товарах (**name, sku, price, quantity**);

Данные о товарах

Имя переменной	Описание	Тип
name (обязательная переменная)	Название товара	Строка
sku (обязательная переменная)	Идентификатор товара	Строка
category (необязательная переменная)	Категория товара	Строка
price (обязательная переменная)	Цена за единицу товара	Число
quantity (обязательная переменная)	Количество единиц товара	Число

Рис. 727. Обязательные переменные - name, sku, price, quantity

Важно:

- имена переменных (**transactionId, transactionTotal, transactionShipping** и т.д.) должны называть точно также, как написано в документации Google;
- у каждой переменной - свой тип данных. Где-то строка, где-то число. Это важно, поскольку данные о транзакциях можно передавать не только в Google Analytics, но и в другие инструменты веб-аналитики, например, в Facebook. При некорректном типе переменной возникнет ошибка;
- вся структура кода должны быть идентична тому, который представлен в справке Google. Вы можете не использовать необязательные параметры, но вы не можете переименовать **transactionId** в **purchaseId**;
- пример выше — это всего лишь пример. Соответственно, значения всех параметров должны динамически заменяться функцией (-ями), которые напишет разработчик. Простое копирование этого кода ни к чему не приведет. Ведь транзакции 1234 (на примере выше) в вашем случае может вообще не быть, а сумма покупки изменяется от заказа к заказу;
- собственные переменные не поддерживаются в коде. Тот, кто формирует уровень данных, должен точно следовать руководству и использовать те переменные, которые есть в документации;
- не все переменные, представленные в коде выше, являются обязательными. Можно сократить их количество, оставив только обязательные.

Таким образом, для формирования уровня данных для стандартной электронной торговли можно пойти 2 путями:

1. заполнить и передать в инструменты веб-аналитики только 2 обязательных переменных о самой транзакции (уникальный идентификатор транзакции transactionId и общую сумму транзакции transactionTotal);
2. заполнить и передать в инструменты веб-аналитики 2 обязательных переменных о транзакции и + 4 о товарах в заказе. В этом случае добавляются к **transactionId** и **transactionTotal -> name, sku, price, quantity**. Всего 6 переменных.

Сколько данных и какие следует передавать в Google Analytics решаете самостоятельно. Однако, чем больше данных вы передаете, тем больше вы увидите информации в отчетах по электронной торговле.

Google Tag Manager не поддерживает PHP, поэтому если ваш разработчик пришлет вам сформированный уровень данных такого вида:

```
<script>
window.dataLayer = window.dataLayer || []
dataLayer.push({
    'transactionId': '<?=$orderID?>',
    'transactionTotal': <?=$summAll?>,
    'transactionProducts': [{
        'sku': '<?=$arItem['PRODUCT_ID']?>',
        'name': '<?=htmlspecialchars($arItem['NAME'])?>',
        'category': '<?=htmlspecialchars($arItem['SECTIONS_NAME'])?>',
        'price': <?=round($arItem['PRICE'])?>,
        'quantity': <?=$arItem['QUANTITY']?>
    }]
});
</script>

window.dataLayer = window.dataLayer || []
dataLayer.push({
 'transactionId': '1234',
 'transactionAffiliation'
```

И вы захотите вставить его в тег типа **Пользовательский HTML**, то либо вы получите ошибку компилятора, либо вам просто выведется необработанный PHP-код в браузер, а не выполнится сам код. Если вы хотите увидеть результат работы скрипта на PHP, то вам нужно будет запустить этот скрипт на собственном сервере и получить результаты через JavaScript. Вот здесь как раз и пригодятся знания работы с Google Tag Manager.

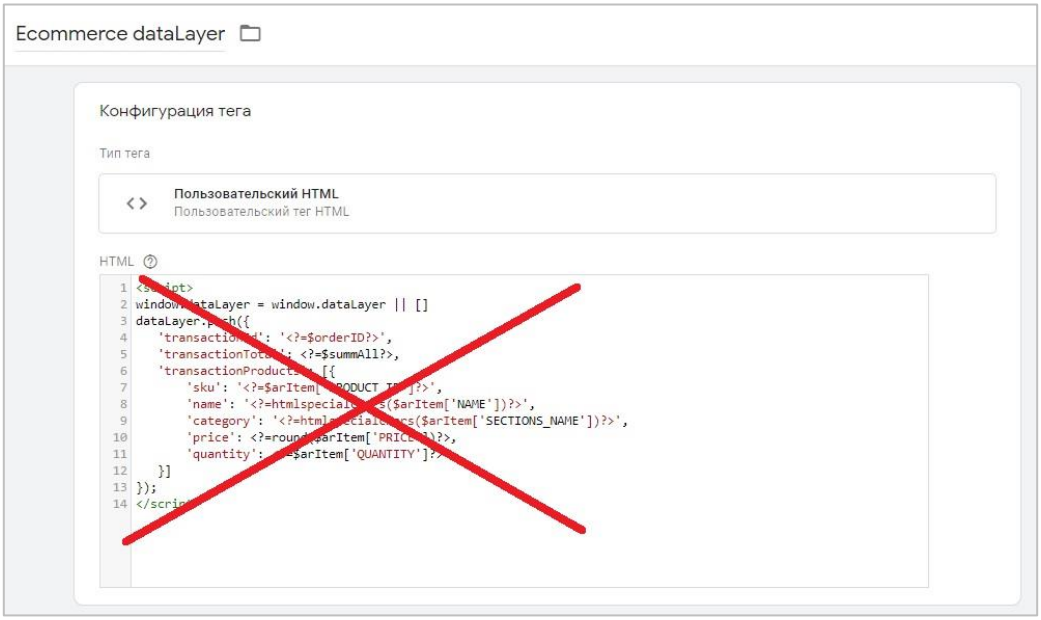


Рис. 728. Некорректное формирование уровня данных в GTM посредством функций PHP

Google Tag Manager и DOM Scraping

Но что, если мы не хотим или не можем обратиться за помощью к разработчику? Тогда нам нужно формировать уровень данных самостоятельно. Сделать это можно с помощью Google Tag Manager. Технология получения веб-данных путем извлечения их со страниц сайтов по-английски называется **DOM Scraping (Web scraping)**. Этим мы и будем заниматься.

В таком способе извлечения данных со страницы сайт есть один существенный недостаток - все, что мы делаем, завязано на объектной модели документа (DOM). И при изменении каких-либо классов,

идентификаторов отслеживаемых элементов на сайте есть большая вероятность потерять все настройки. Именно поэтому лучше выполнять отслеживание электронной торговли или любых других нестандартных событий на сайте с помощью функций и привлечения разработчика, а не извлекая данные с помощью скрапинга.

В качестве примера я буду использовать один из интернет-магазинов по продажи товаров для взрослых на платформе inSales и следующие инструменты:

- расширение **GTM Variable Builder** для браузера Google Chrome, которое позволяет создавать пользовательские переменные типа **Собственный код JavaScript** и извлекать значения из элементов сайта всего за пару кликов;
- специальный тег из Галереи шаблонов, который называется **dataLayer Builder + Standard Ecommerce**.

inSales не поддерживает PHP. Но в настройках есть возможность добавления собственного JavaScript-кода:

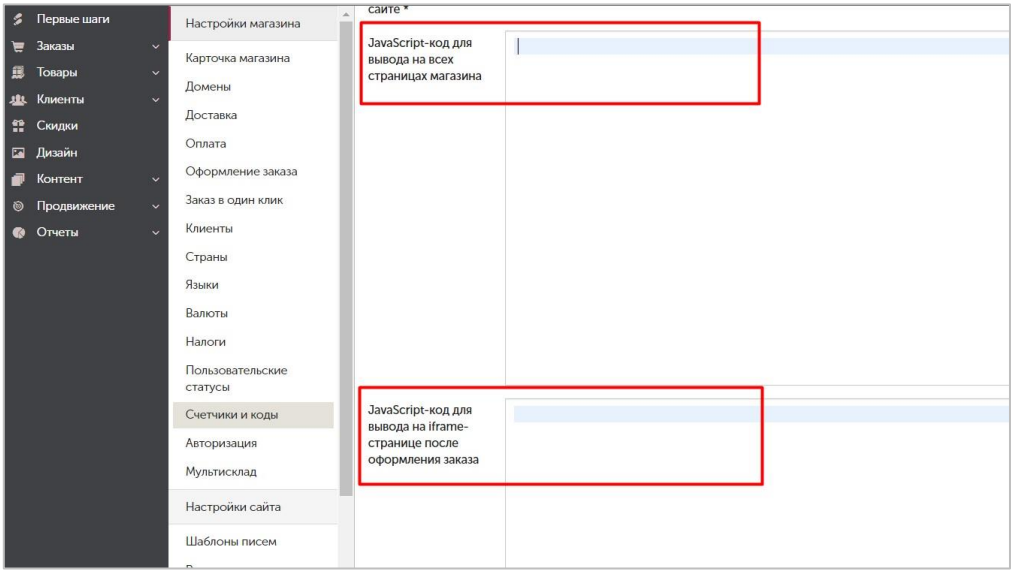


Рис. 729. Возможность добавления собственного JavaScript-кода на страницах сайта

Я не буду использовать эти поля, а сформирую уровень данных через Google Tag Manager. Давайте приступим к извлечению обязательных переменных для настройки стандартной электронной торговли.

Важно: *значения всех извлекаемых со страницы переменных должны быть доступны на странице "Спасибо" во время отправки тега. Если на странице не выводится идентификатор транзакции, общая сумма покупки или какие-либо другие обязательные переменные, которые участвуют в формировании уровня данных (данные о товаре - name, sku, price, quantity), то все равно необходимо будет прибегнуть к помощи разработчика. Для скрапинга важно, чтобы извлекаемые данные находились на нужной нам странице. В данном случае, это страница успешной покупки ("Спасибо"). Если этих данных нет, то и нечего извлекать.*

В моем примере страница успешно оформленного заказа выглядит так:

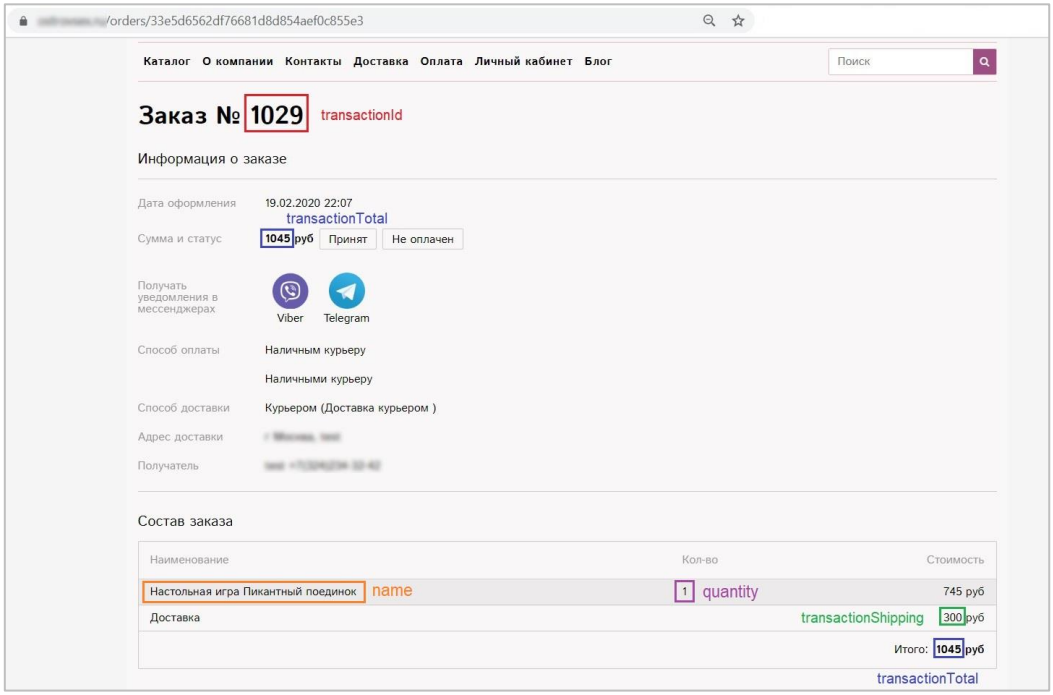


Рис. 730. Страница "Спасибо"

С этой страницы мы можем извлечь данные о переменных **transactionId** и **transactionTotal** (на примере выводится в двух местах), стоимости доставки **transactionShipping**, хоть она и не является обязательной, а также наименование товара **name** и количество **quantity**. К сожалению, для корректного формирования уровня данных 2 обязательных переменных из 4 для информации о товарах недостаточно. Поэтому есть два выбора:

1. нужно просить разработчика изменить эту страницу и добавить недостающие **sku** и **price**;
2. передавать в уровне данных только **transactionId** и **transactionTotal** данные о транзакции без каких-либо других переменных.

Использование расширения GTM Variable Builder

Предположим, мы выбрали второй вариант. Чтобы извлечь данные по этим переменным, воспользуемся расширением **GTM Variable Builder**. Выделите нужный фрагмент на странице для извлечения данных, нажмите на иконку расширения GTM Variable Builder в правом верхнем углу браузера, и на вкладке **Console** получите результат:

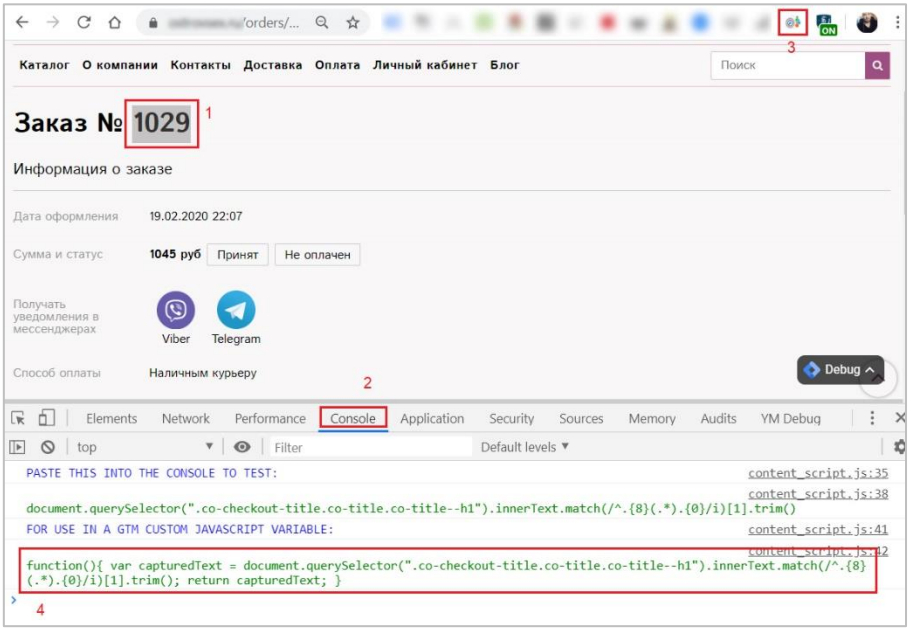


Рис. 731. Пример извлечения данных для transactionId

Скопируйте код из строчки **FOR USE IN A GTM CUSTOM JAVASCRIPT VARIABLE** и вставьте его в переменную типа **Собственный код JavaScript**.

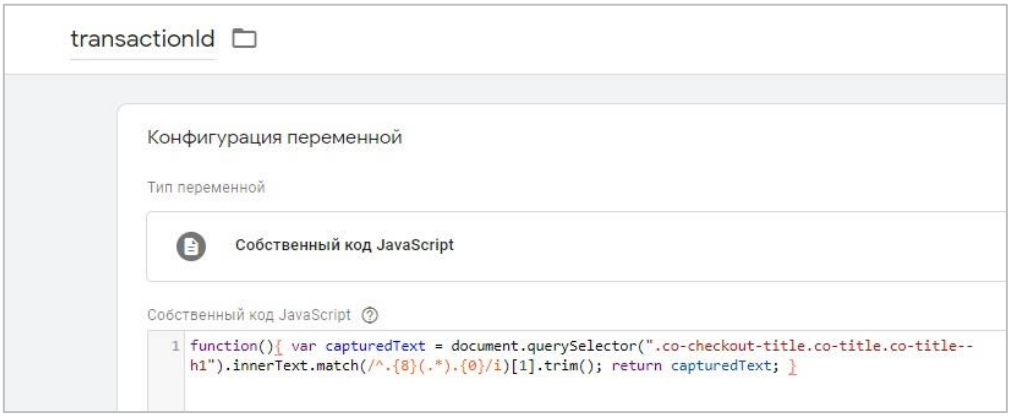


Рис. 732. Переменная Собственный код JavaScript

Сохраните переменную. Аналогично проделайте со всеми остальными переменными. Не забудьте, что общая сумма транзакции (**transactionTotal**) должна быть числом. Если вы не хотите использовать расширение **GTM Variable Builder**, то тогда следует искать на странице "Спасибо" у конкретного элемента его CSS-селектор, копировать его, при необходимости проводить манипуляции с этим элементом (преобразование типа данных, обрезание части текста и т.д.), а после добавить в пользовательскую переменную типа **Собственный код JavaScript**.

В Google Tag Manager в режиме предварительного просмотра на вкладке **Variables** напротив конкретного события отображаются данные. Значения переменных с типом **number (строка)** выводятся без кавычек, а с типом **string (строка)** в кавычках. Как в случае с **transactionId**:

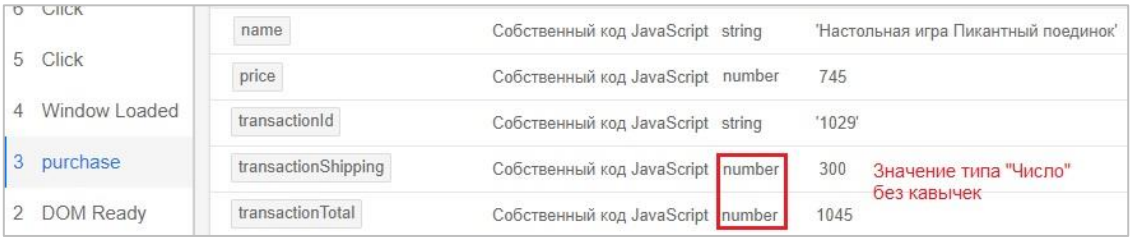


Рис. 733. Типы переменных

На вкладке **Data Layer** переменные с числовым значением выделяются зеленым цветом, а со строковым - красным.

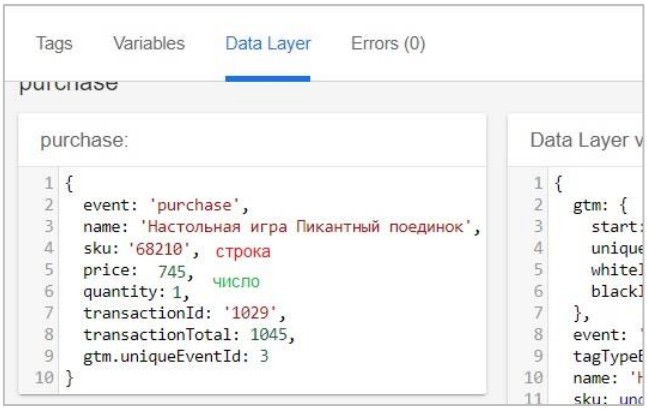


Рис. 734. Разный цвет типа данных для строк и чисел в отладке

Примечание: извлекать числовые переменные (например, стоимость доставки или общую сумма транзакции) с помощью GTM Variable Builder можно не целиком, а лишь часть (без руб.). Просто выделите фрагмент и расширение само допишет нужное регулярное выражение.

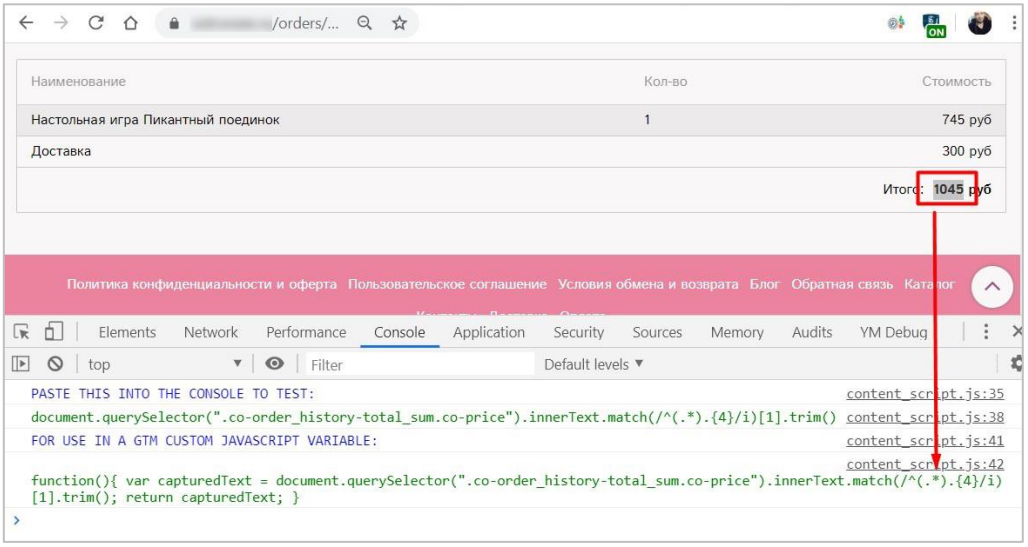


Рис. 735. Извлечение суммы транзакции

Данные о товарах извлекать чуть сложнее, поскольку в одной транзакции может быть несколько товаров. Здесь нам также необходим CSS-селектор товара, который мы извлечем со страницы "Спасибо". Если вы используете расширение GTM Variable Builder, то он сам сформирует код для переменной, и вместо **document.querySelector** будет использовать **document.querySelectorAll**, потому что нужно захватить все значения продукта, а не только первого.

Если вы будете извлекать данные вручную, воспользуйтесь кодом ниже:

```
function(){
var items = document.querySelectorAll("CSS-селектор");
var itemsLength = items.length;
var myProducts = []
for (i = 0; i < itemsLength; i++) {
    myProducts.push(items[i].innerText);
}
  return myProducts;
}
```

, где в поле **"CSS-селектор"** вставляете то значение, которое присутствует на вашем сайте.

- **document.querySelectorAll** - возвращает статический (не динамический) NodeList, содержащий все найденные элементы документа, которые соответствуют указанному селектору.
- **document.querySelector** - возвращает первый элемент (Element) документа, который соответствует указанному селектору или группе селекторов.

Поскольку в заказе может быть несколько товаров (2, 3, 5, 10 и т.д.), то **document.querySelector** (как было в примерах ранее) нам не подойдет, поскольку он добавит в массив только первый элемент. Чтобы в dataLayer поместились все товары, которые купил пользователь, используем **document.querySelectorAll**.

Для наглядности приведу пример извлечение **quantity** с помощью GTM Variable Builder на странице "Спасибо":

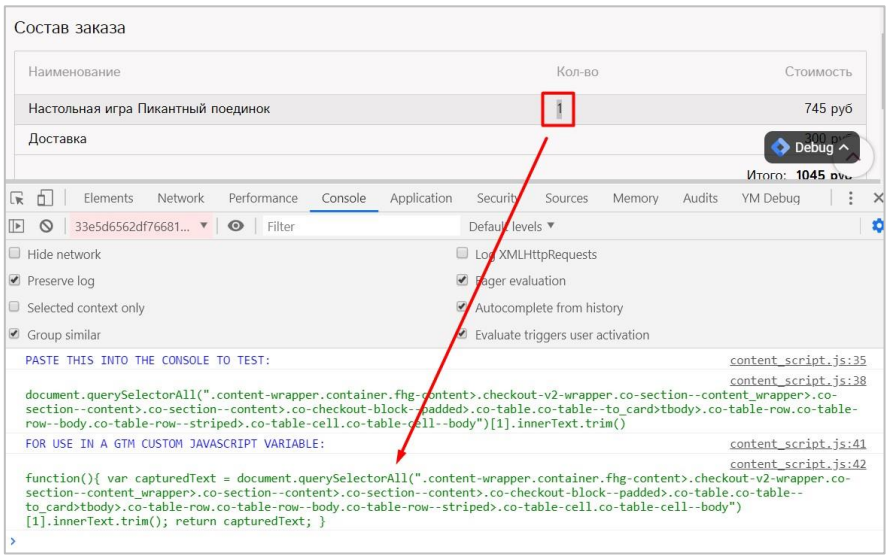


Рис. 736. Извлечение данных по переменной quantity со страницы "Спасибо"

Как видим, GTM Variable Builder вернул нам значение с **document.querySelectorAll**. А это значит, что в dataLayer передастся корректная информация.

Примечание: используйте точку в качестве десятичного разделителя в числе, а не запятую. При передаче числового значения запятая может быть неправильно определена.

Итак, что же у нас получилось? На примере моего интернет-магазина на странице "Спасибо" я могу извлечь 2 обязательные переменные для данных о транзакции (**transactionId** и **transactionTotal**), 1 необязательную (**transactionShipping**) и 2 из 4 для формирования данных о товаре (**name** и **quantity**). На странице успешного оформления покупки нет только ID товара (**sku**) и цены за единицу товара (**price**). В этом случае я могу отправить только 2 переменные в уровень данных, а затем эту информацию в Google Analytics.

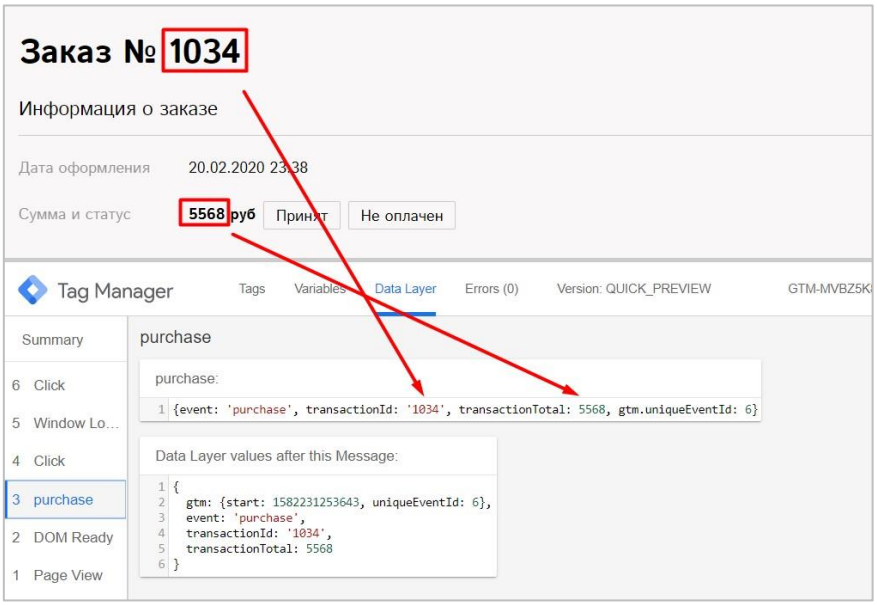


Рис. 737. Данные на уровне данных (Data Layer)

В результате в отчете **Транзакции** в Google Analytics получим данные:

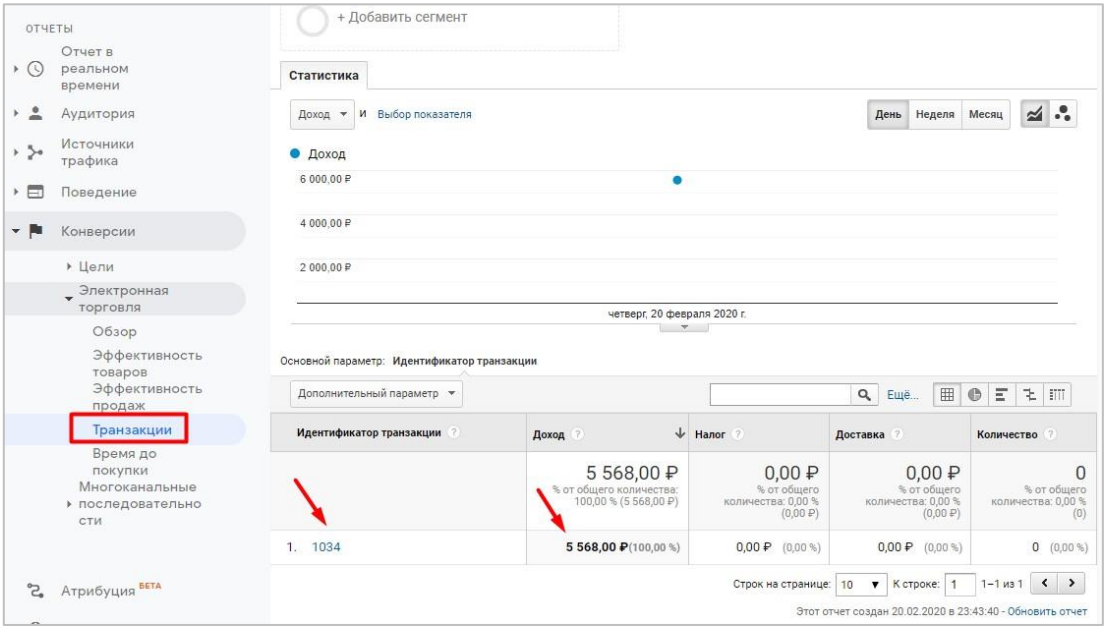


Рис. 738. Отчет Транзакции в Google Analytics

А сам уровень данных в Google Tag Manager будет иметь вид:

Рис. 739. Уровень данных с 2 обязательными переменными

Таким образом, настроить самую простую реализацию уровня данных для стандартной электронной торговли и передачей этой информации в Google Analytics можно всего лишь с помощью 2 обязательных переменных. Если вам необходимо передавать и данные о товаре, то сформируйте другой уровень данных, содержащий +4 обязательных переменных (**name, sku, price, quantity**). Но для начала извлеките с помощью GTM значения всех обязательных переменных.

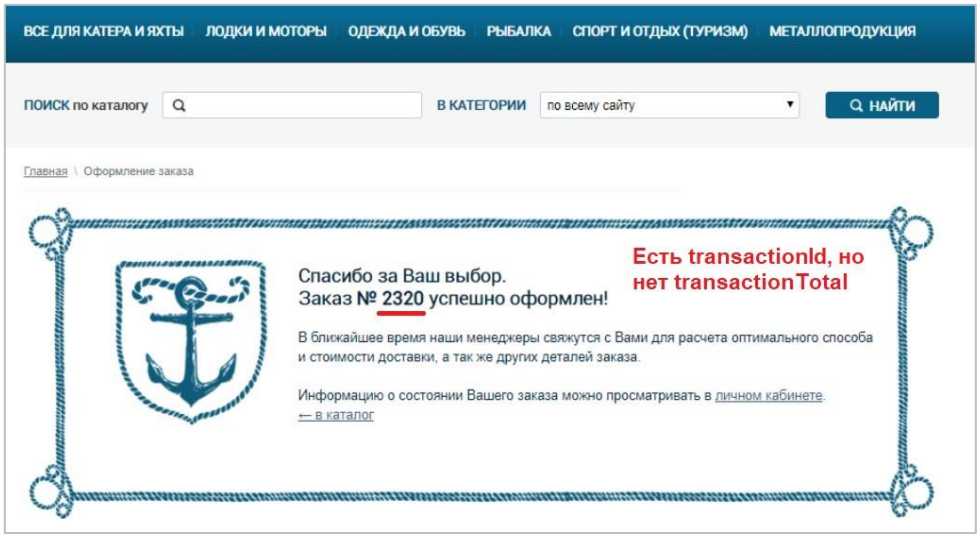


Рис. 740. Пример страницы "Спасибо", на которой нет transactionTotal

Если у вас также, как и у меня в этом примере, нет каких-то данных, вы можете пропустить эти переменные и сформировать уровень данных без них:

```
<script>
window.dataLayer = window.dataLayer || [];
dataLayer.push({
 'event': 'purchase',
 'transactionId': '{{transactionId}}',
 'transactionTotal': {{transactionTotal}},
 'transactionShipping': 0,
 'transactionProducts': [{
   'sku': '',
   'name': '{{name}}',
   'price': '',
   'quantity': {{quantity}}
}]
  });
</script>
```

Просто в Google Analytics будет попадать не вся информация о товарах, а в недостающих значения в отчетах будет написано (not set).

Сформировать такой уровень данных можно несколькими способами:

1. создать тег типа **Пользовательский HTML**;
2. использовать специальный тег из Галереи шаблонов **dataLayer Builder + Standard Ecommerce**.

Разберем два способа настройки.

Создание пользовательского HTML тега

Перейдите в раздел **Теги** и создайте тег типа **Пользовательский HTML**. Вставьте в него следующую конструкцию:

```
<script>
window.dataLayer = window.dataLayer || [];
dataLayer.push({
 'event': 'purchase',
 'transactionId': '{{transactionId}}',
 'transactionTotal': {{transactionTotal}},
 'transactionShipping': {{transactionShipping}},
 'transactionProducts': [{
   'sku': '{{sku}}',
   'name': '{{name}}',
   'price': {{price}},
   'quantity': {{quantity}}
}]
  });
</script>
```

В Google Tag Manager это будет выглядеть так:

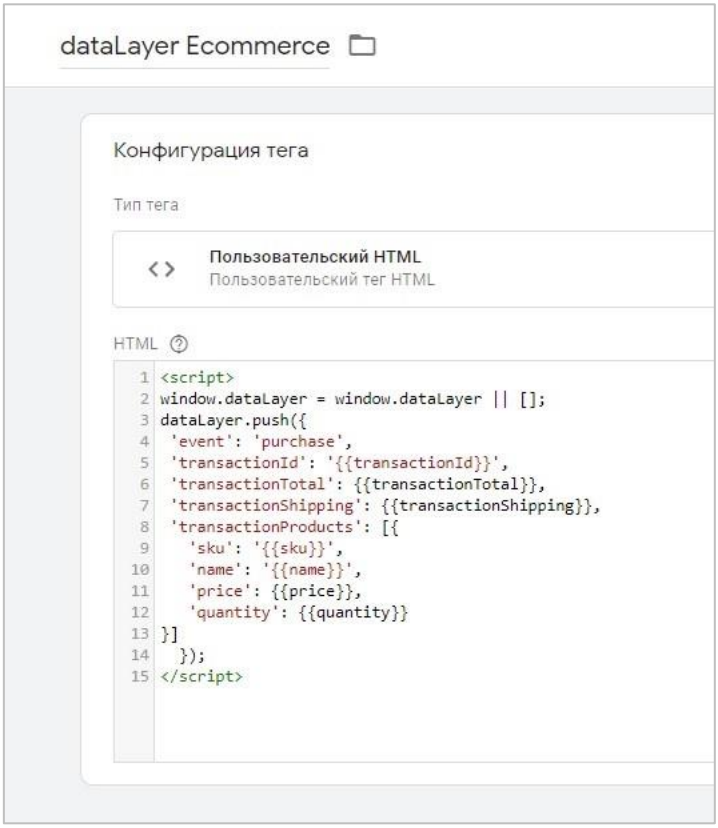


Рис. 741. Сформированный уровень данных для электронной торговли

Каждой обязательной переменной присваивается значение переменной, которое мы создали на предыдущих шагах. С помощью двойных фигурных скобок **{{ }}** мы можем обратиться к любой переменной GTM.

Такой код работает в случае, если у вас 1 товар. Но если их несколько, то данные о товарах в уровне данных будут собраны не совсем корректно:

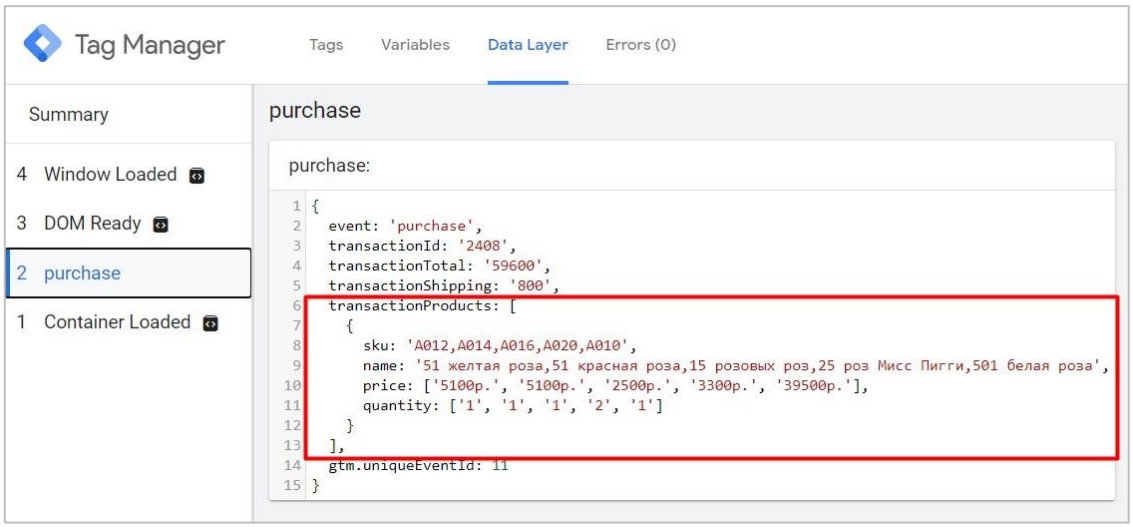


Рис. 742. Некорректный массив данных о товарах

Необходимо воспользоваться следующим кодом, который перебирает в цикле все товары в заказе и далее формирует верное свойство объекта **transactionProducts**:

```
<script>
var ecomProducts = [];
for (var i = 0; i < {{sku}}.length; i++) {
ecomProducts.push({
'sku': {{sku}}[i],
```

```
'name': {{name}}[i],
'category': '',
'price': {{price}}[i],
'quantity': {{quantity}}[i]
});
}
window.dataLayer = window.dataLayer || [];
dataLayer.push({
'event': 'purchase',
'transactionId': {{transactionId}},
'transactionAffiliation': '',
'transactionTotal': {{transactionTotal}},
'transactionTax': 0,
'transactionShipping': {{transactionShipping}},
'transactionProducts': ecomProducts
});
</script>
```

Тогда при срабатывании события покупки (purchase) массив данных будет сформирован верно:

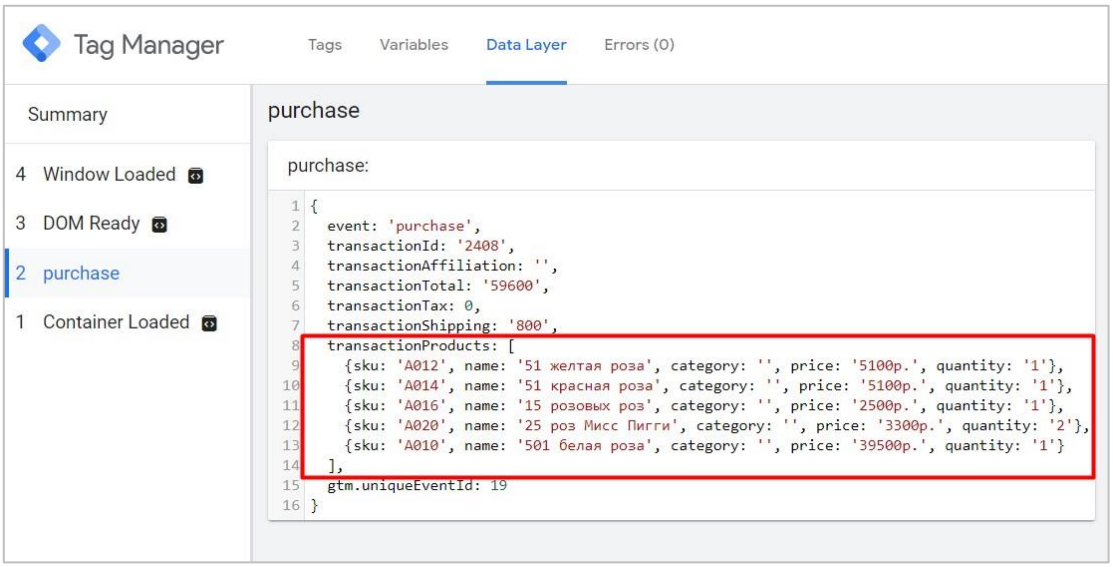


Рис. 743. Корректный массив данных о товарах

Использование специального тега из Галереи шаблонов

Вы можете добавить в свою рабочую область шаблон тега для стандартной электронной торговли, который называется **dataLayer Builder + Standard Ecommerce и dataLayer Builder + Enhanced Ecommerce** (Автор - Mikeulrich75).

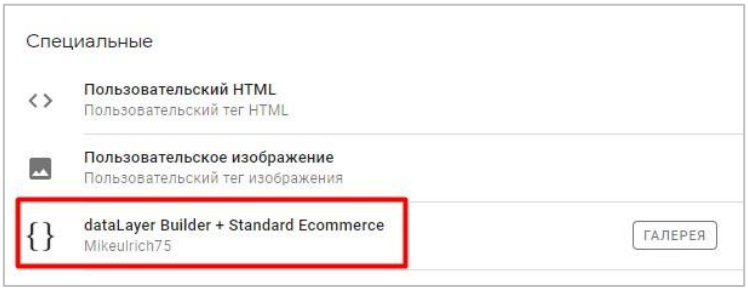


Рис. 744. Тег dataLayer Builder + Standard Ecommerce

Тег позволяет упростить формирование уровня данных. Больше не нужно создавать пользовательский HTML тег, вручную прописывать переменные, следить за синтаксисом кода (исправлять кавычки, запятые, скобки и т.д.). Все это можно сделать в простом и удобном интерфейсе тега. Он чем-то напоминает пользовательскую переменную **Таблица поиска** в Google Tag Manager.

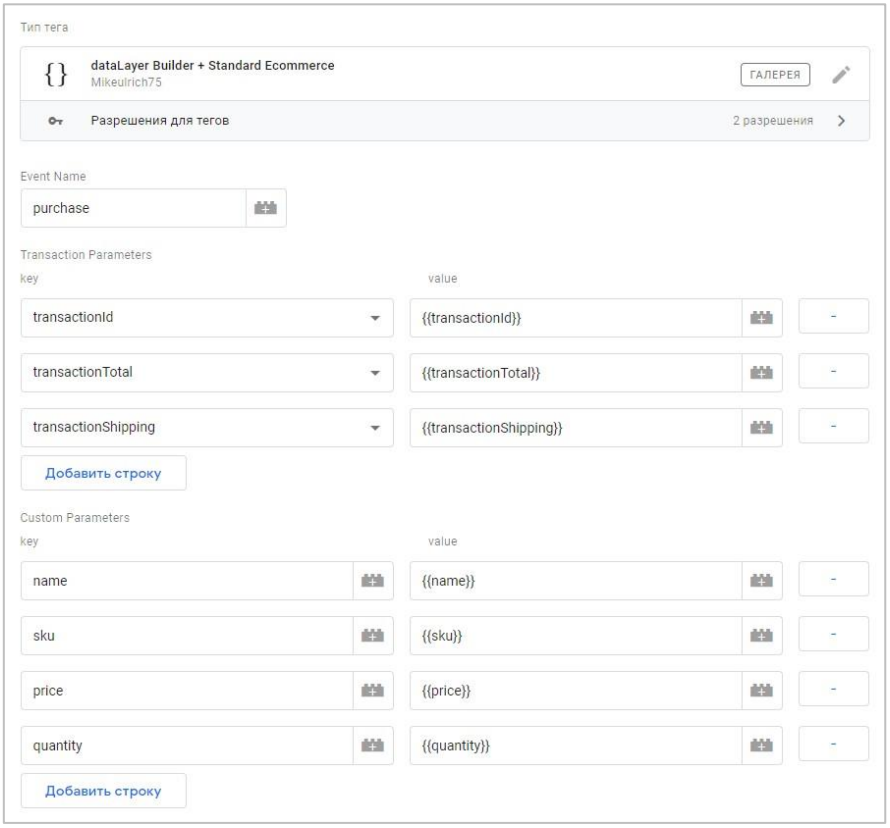


Рис. 745. Настройки тега

В поле **Event Name** введите название пользовательского события. Это **purchase**. В параметрах транзакции всего два обязательных ключа со своими значениями — это **transactionId** и **transactionTotal**. Но у меня еще был **transactionShipping**. В поле **value** я добавляю созданные переменные (см. выше). Аналогично проделываем и с параметрами товара - **name, sku, price, quantity**.

Таким образом, с появлением Галереи шаблонов GTM и готового тега стало проще формировать уровень данных для электронной торговли. Но можно использовать и первый вариант настройки, как вам будет удобнее.

В приведенном выше коде мы помещаем событие GTM с именем **purchase** на уровень данных. Для того, чтобы событие срабатывало и данные отправлялись в Google Analytics, необходимо:

1. создать пользовательское событие **purchase**;
2. создать тег Universal Analytics с типом отслеживания **Транзакция**.

Но перед тем, как это сделать, давайте разберем различные способы размещения уровня данных на странице сайта. Чтобы узнать, где метод dataLayer.push () должен быть добавлен, нужно подумать о том, что происходит после успешной покупки?

Покупателя перенаправляет на отдельную страницу "Спасибо" с уникальным URL? Или страница остается прежней, URL не изменяется, а пользователю на экране отображается только сообщение об успешной транзакции?

Если клиент перенаправляется на страницу "Спасибо", и вы используете уровень данных без события, то код электронной торговли должны быть размещен над кодом контейнера Google Tag Manager.

Что значит dataLayer.push () без события?

А теперь хотелось бы показать вам одну тонкую деталь, которые многие могли бы не заметить, но которая играет важную роль при отслеживании электронной торговли. И это строчка кода с 'event': ' purchase' и без нее. В официальной документации и в самом начале этой статьи я приводил такой код:

```
<script>
window.dataLayer = window.dataLayer || []
dataLayer.push({
   'transactionId': '1234',
   'transactionAffiliation': 'Acme Clothing',
   'transactionTotal': 38.26,
   'transactionTax': 1.29,
   'transactionShipping': 5,
   'transactionProducts': [{
       'sku': 'DD44',
       'name': 'T-Shirt',
       'category': 'Apparel',
       'price': 11.99,
       'quantity': 1
   },{
       'sku': 'AA1243544',
       'name': 'Socks',
       'category': 'Apparel',
       'price': 9.99,
       'quantity': 2
   }]
});
</script>
```

Рис. 746. Код из документации Google

А затем в Google Tag Manager настраивал такой:

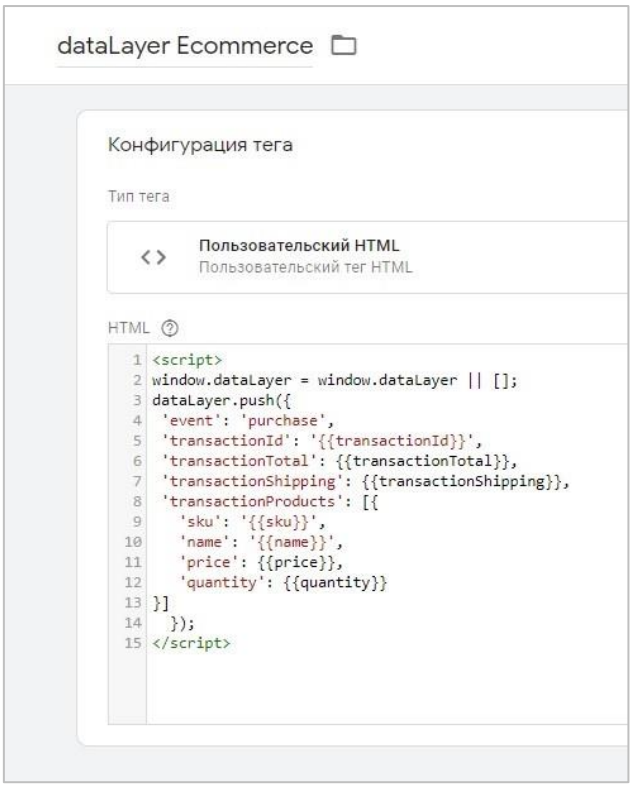


Рис. 747. Сформированный уровень данных для электронной торговли

Казалось бы, разница в одной строчке 'event': ' purchase' после dataLayer.push, но она все меняет. В первом случае (без event) важен порядок размещения кода (ДО или ПОСЛЕ контейнера GTM), а во втором случае уже нет, поскольку это классический формат события Google Analytics, где есть Категория, Действие, Ярлык, Ценность. Просто выглядит чуть сложнее. Но если присмотреться и записать код в одну строчку, то становится очень похоже:

dataLayer.push('event', 'purchase', {'transactionId':'{{transactionId}}', ...});

на конструкцию обычного отслеживаемого события, например, для библиотеки gtag.js:

gtag('event', <action>, { 'event_category': <category>, 'event_label': <label>, 'value': <value>});

Что это значит? Если в dataLayer отсутствует строка с event, то мы не сможем создать триггер типа Пользовательское событие и активировать его в тот момент, когда нам это необходимо.

Способ №1. Код dataLayer.push () добавляется ДО контейнера GTM (на странице «Спасибо»)

Возвращаемся к размещению кода dataLayer.push () над кодом контейнера GTM в случае, если пользователь перенаправляется на страницу "Спасибо" с отдельным URL. Почему так? Потому что тогда данные о транзакции электронной торговли уже будут доступны для Google Tag Manager, когда он начнет загружаться. А чем раньше эти данные будут переданы на уровень данных, тем быстрее мы сможем отправить эти данные в Google Analytics.

Поскольку этот dataLayer.push не имеет параметра event (как я написал выше), в режиме отладки он будет отображаться словом Message.

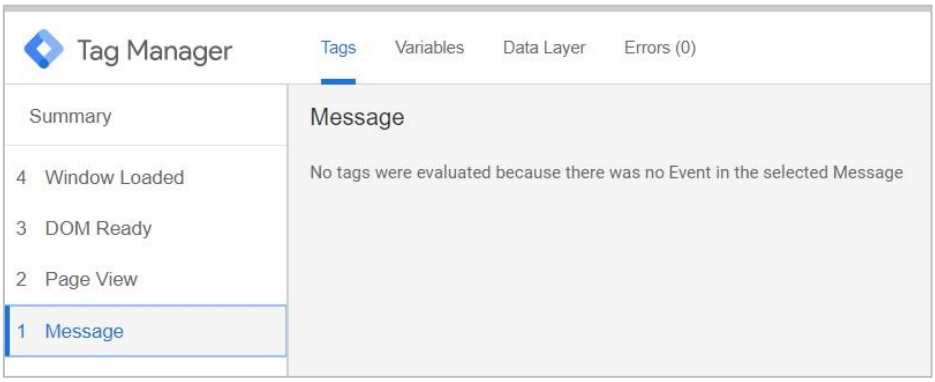


Рис. 748. Код расположен ДО контейнера GTM

Message мы не сможем использовать в качестве триггера в GTM. Поэтому самый ранний момент, когда мы можем отправить данные транзакции в Google Analytics — это событие просмотра страницы (Page View).

Вы прекрасно знаете, что каждый раз, когда на сайте загружается страница, и вы включаете режим предварительного просмотра GTM, там отображаются 3 события: **Просмотр страницы (Container Loaded), Модель DOM готова (DOM Ready)** и **Окно загружено (Window Loaded)**. Если разработчик на сайте разместит код электронной торговли с dataLayer.push до контейнера Google Tag Manager, то вы сможете отправить данные о транзакции в Google Analytics с событием Container Loaded.

Способ №2. Код dataLayer.push () добавляется ПОСЛЕ контейнера GTM (на странице «Спасибо»)

Поскольку этот dataLayer.push также не имеет параметра event, в режиме отладки он также будет отображаться как **Message**.

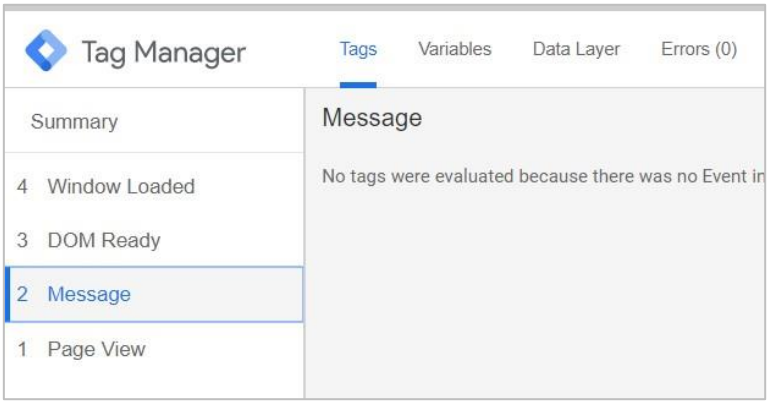


Рис. 749. Код расположен ПОСЛЕ контейнера GTM

Однако на этот раз dataLayer.push был завершен чуть позже, потому что он расположен ниже контейнера Google Tag Manager. Теперь с событием Page View мы не сможем получить доступ к данным. Остается использовать только два других события: **Модель DOM готова (DOM Ready)** или **Окно загружено (Window Loaded)**. Чаще всего при настройке используют именно второе событие DOM Ready.

Способ №3. Код dataLayer активируется на странице с помощью события

В этом способе как раз и используется дополнительный параметр события **(event)**, значением которого является покупка **(purchase)**.

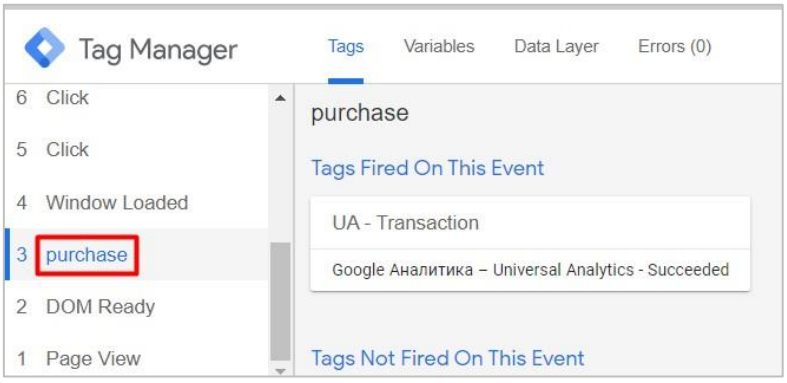


Рис. 750. Событие purchase

Такой способ чаще всего используется веб-аналитиками, поскольку с помощью пользовательского события мы можем настроить активацию триггера и тега как угодно.

И вот теперь мы снова возвращаемся к нашему примеру и части, когда для того, чтобы событие срабатывало и данные отправлялись в Google Analytics, необходимо:

1. создать пользовательское событие **purchase**;
2. создать тег Universal Analytics с типом отслеживания **Транзакция**.

Чтобы создать пользовательское событие, перейдите на вкладку **Триггеры** и создайте триггер типа **Пользовательское событие** с именем события **purchase**:

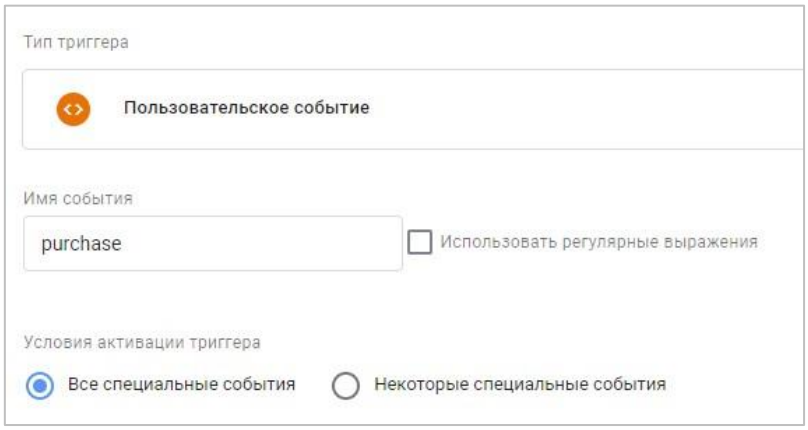


Рис. 751. Пользовательское событие purchase

Далее перейдите на вкладку **Теги** и создайте тег Google Аналитика - Universal Analytics с типом отслеживания **Транзакция**.

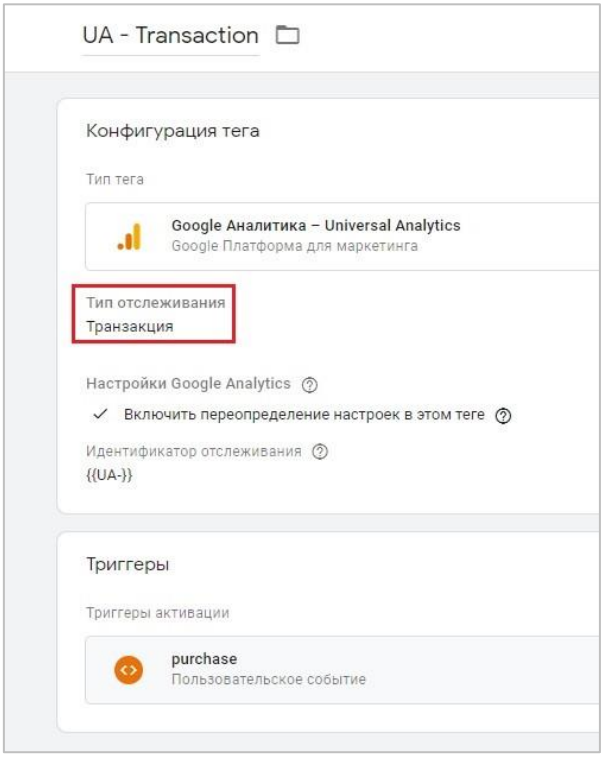


Рис. 752. Настройка тега

В качестве триггера активации укажите пользовательское событие **purchase**. Сохраните настройки.

И последнее, что осталось выяснить — это триггер активации для уровня данных, который формируется в теге типа **Пользовательский HTML** тег. Какой он? При условии, что вы используете третий способ отслеживания (код с событием **'event':'purchase'**), не имеет значения, что выбрать. Отличие будет в том, когда будут передаваться данные - с **Container Loaded**, с **DOM Ready** или с **Window Loaded**. Они все равно передадутся.

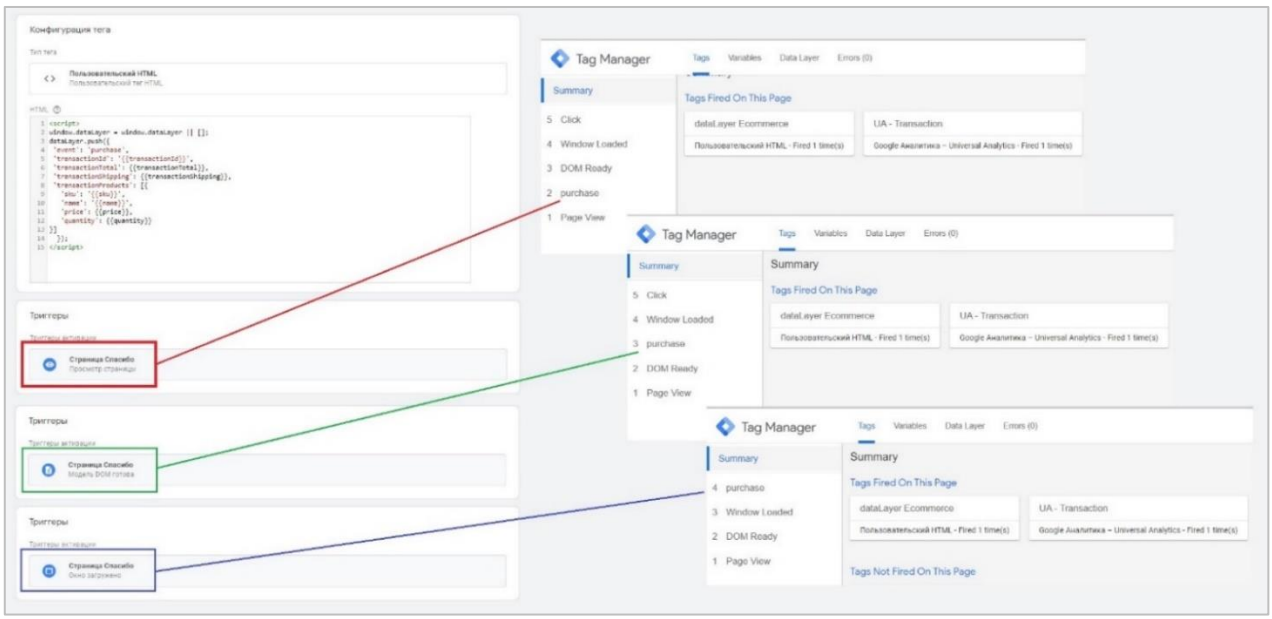


Рис. 753. Различные способы активации тега

В качестве условия активации триггера можно указать конкретно страницу "Спасибо". У меня в примере это реализовано через заголовок страницы, который на странице успешного заказа имеет title **Заказ №**:

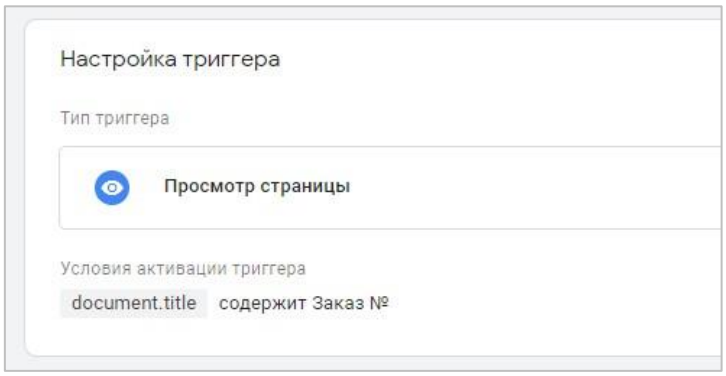


Рис. 754. Условие активации триггера - на странице "Спасибо"

Все, что осталось сделать, это проверить корректность формирования уровня данных и отправки информации о транзакции в Google Analytics. Для этого можно воспользоваться расширением GA Debugger.

Включите расширение, откройте вкладку Console в браузере и совершите транзакцию. Вы увидите много информации, появляющейся в консоли. GA Debugger проверяет веб-сайт и выводит всю информацию, связанную с Google Analytics. Нас интересует **hitType - transaction**:

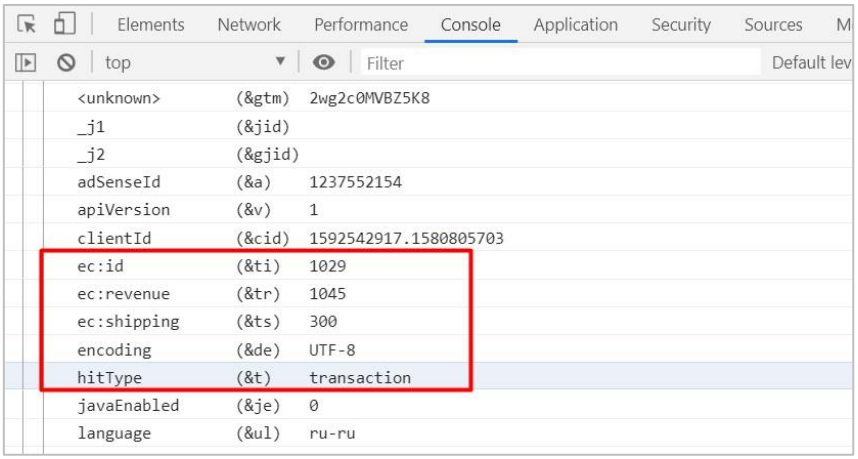


Рис. 755. hitType - transaction

Это покупка, которая была отправлена в Google Analytics:

- **ec:id** - идентификатор транзакции;
- **ec:revenue** - общая сумма покупки (в Google Analytics - Доход);
- **ec:shipping** – стоимость доставки.

Если вы передаете информацию о купленных товарах, то прокрутите консоль внизу. Вы должны увидеть такие данные:

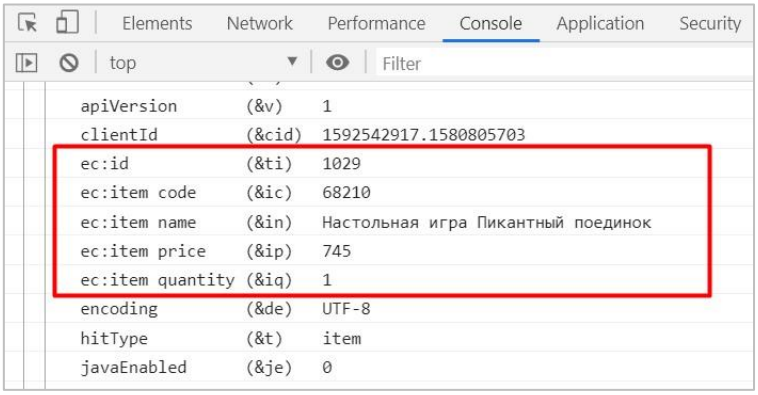


Рис. 756. Данные о товаре в GA Debugger

Просмотрите всю консоль на предмет ошибок. Если они есть, исправьте самостоятельно или проконсультируйтесь с разработчиком. Если ошибок нет, то на этом настройка стандартной электронной торговли с помощью Google Tag Manager завершена.

Данные по отслеживанию доступны в отчетах Google Analytics в разделе **Конверсии - Электронная торговля**:

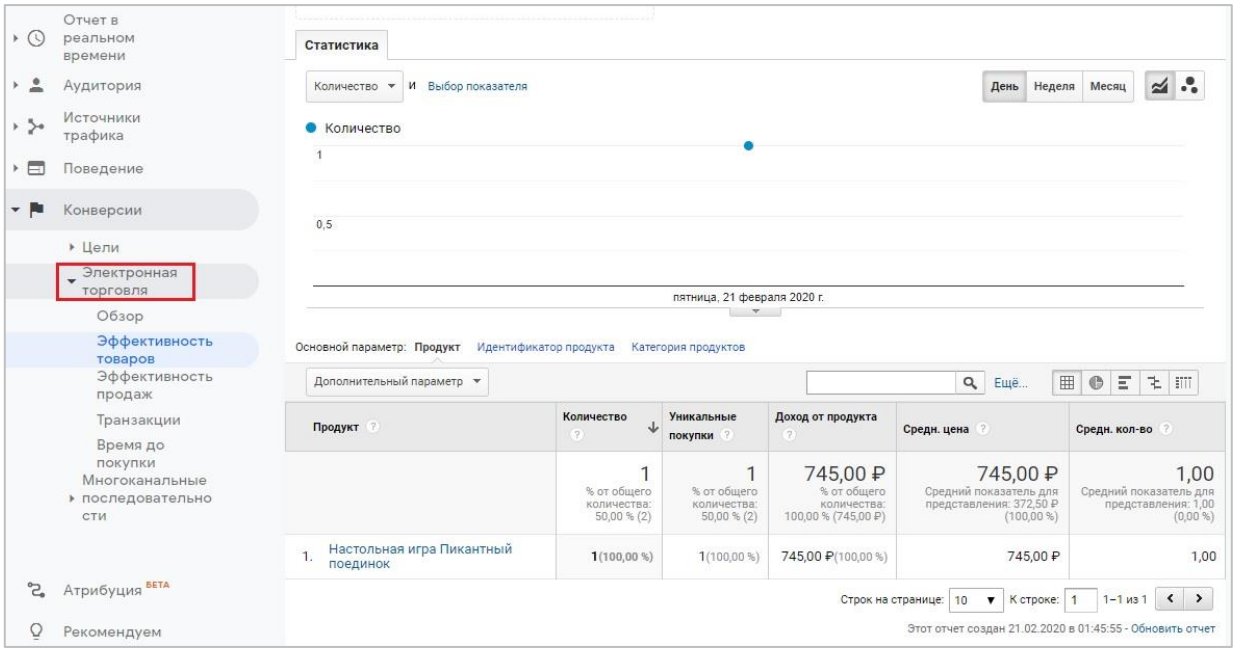


Рис. 757. Отчеты по электронной торговле

Итоги

Несмотря на то, что стандартная электронная торговля настраивается проще, чем расширенная, по причине меньшего количества отслеживаемых событий, она все равно может быть сложна для начинающих.

В этой главе мы с вами:

- досконально разобрали процесс настройки без привлечения разработчика и код стандартной электронной торговли;
- узнали про обязательные и необязательные переменные для уровня данных;
- поработали с технологией DOM Scraping;
- научились извлекать данные со страницы напрямую и с помощью GTM Variable Builder;
- познакомились с инструментами, которые упрощают работу - GTM Variable Builder, специальный тег dataLayer Builder + Standard Ecommerce, GA Debugger;
- освоили 3 способа размещения уровня данных на странице сайта;
- передали всю информацию в Google Analytics.

Как я упомянул ранее, не рекомендуется использовать извлечение информации со страницы сайта для формирования уровня данных, поскольку любое изменение CSS-селекторов элементов со стороны верстальщика приведет к сбою в отслеживании. Но в то же время, настройка стандартной электронной торговли с помощью DOM Scraping без привлечения разработчика демонстрирует всю мощь диспетчера тегов Google.

Настройка расширенной электронной торговли

Как вы уже знаете, в отличие от стандартной электронной торговли, где настраивается только событие **purchase (Покупки)**, для расширенной электронной торговли (Enhanced Ecommerce) доступно до 9 уровней отслеживания действий:

1. показы товаров (impressions);
2. клики по товарам (click);

3. показы сведений о товарах (detail);
4. добавление товаров в корзину и удаление их из корзины (add, remove);
5. показы промоакций (promoView);
6. клики по промоакциям (promoClick);
7. оформление покупки (checkout);
8. покупки (purchase);
9. возвраты (refund).

В официальной документации Google (см. приложение) подробно описывается процесс настройки расширенной электронной торговли с использованием уровня данных (dataLayer) и Google Tag Manager, а также приведены примеры кодов, которые вам, как интернет-маркетологу, необходимо адаптировать для своего проекта и добавить в техническое задание (ТЗ), чтобы разработчику было легче выполнить поставленную задачу.

Но перед тем, как выполнять настройку электронной торговли и упростить себе работу, необходимо понять некоторые постулаты:

- настройка электронной торговли подразумевает внедрение дополнительного кода на сайт, следовательно, знаний в программировании, а также умение работать с конкретном движком сайта;
- работа по настройке Enhanced Ecommerce - командная. Интернет-маркетолог (веб-аналитик) отвечает за составление технического задания для разработчика, формирование уровня данных в определенных местах на сайте и за конечную проверку, а разработчик отвечает за внедрение кода согласно вашему ТЗ. Причем разработчик может быть толковым, но не разбираться в конкретном движке вашего сайта;
- для всех популярных CMS-движков существуют готовые решения (плагины, модули) для электронной торговли, которые существенно упрощают процесс настройки и сокращают финансовые затраты;
- уровень данных и структура объектов в Яндекс.Метрике соответствует аналогичным сущностям в Google Analytics Enhanced Ecommerce. Другими словами, настроив один раз для Google Analytics, в Яндекс.Метрике тоже будут доступны отчеты по электронной торговле.

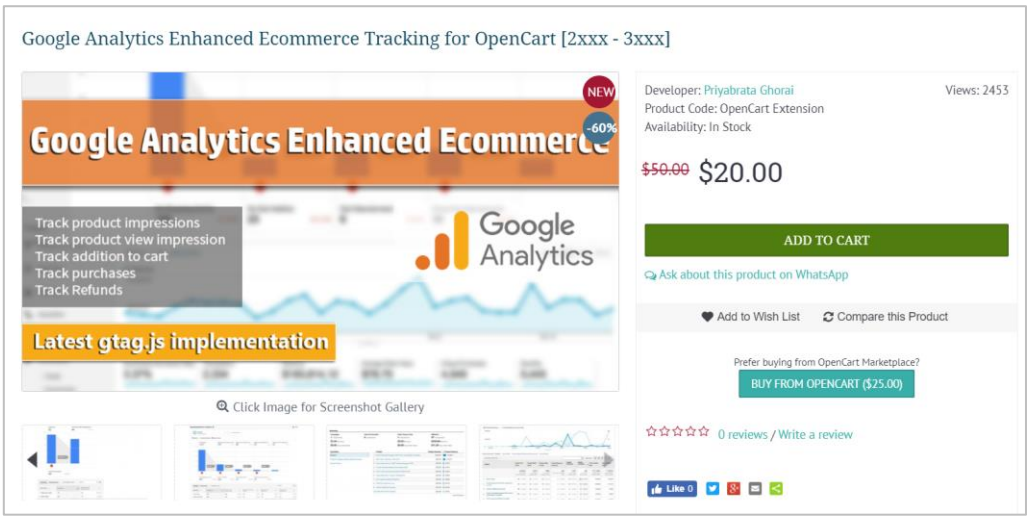


Рис. 758. Пример готового модуля для интернет-магазина на OpenCart

В этой главе я не буду подробно описывать процесс настройки расширенной электронной торговли, поскольку отслеживания действий для каждого из 9 доступных событий схожи, а сами примеры кодов займут в руководстве большое количество места. Мне бы хотелось описать последовательность действий интернет-маркетолога, которую необходимо выполнить для настройки Enhanced Ecommerce, а также поделиться с вами примером технического задания для разработчика, которое я использую в своей работе.

Вам понадобится три источника информации (см. приложение):

- официальная документация Google;
- тестовый интернет-магазин, на котором размещены фрагменты кодов установки;
- универсальное руководство по настройке расширенной электронной торговли от Netpeak;

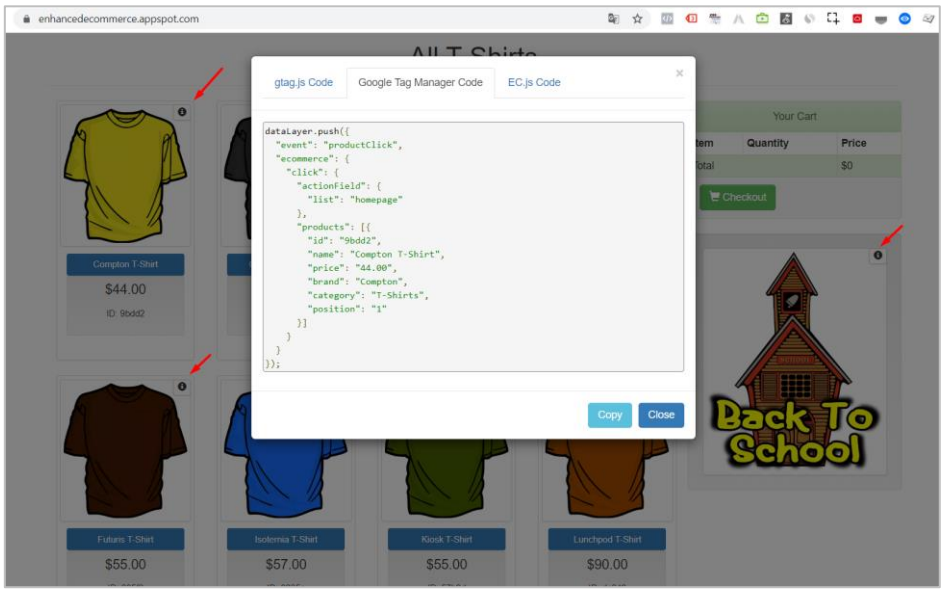


Рис. 759. Тестовый интернет-магазин enhancedecommerce.appspot.com

Последовательность действий выглядит следующим образом:

- определяете библиотеку Google Analytics (**analytics.js, gtag.js** и т.д);
- определяете движок интернет-магазина (1С-Битрикс, OpenCart, Magento, PrestaShop и т.д.). Не забывайте, что электронную торговлю можно настроить не только на интернет-магазин, но и на любой другой сайт;
- задаете себе вопрос: *«Купить готовый модуль или привлечь к задаче разработчика»*?;
- если есть возможность поручить задачу разработчику, то отправляете ему справочные материалы и техническое задание (ТЗ) с просьбой установить коды и сформировать уровни данных на соотвествующих страницах сайта;
- параллельно выполняете настройки в Google Analytics и Google Tag Manager: включаете отслеживание электронной торговли, задаете шаги последовательности (если есть), создаете переменные, триггеры и теги;
- после выполнения работы программистом проверяете корректность передачи данных с помощью режима предварительного просмотра GTM или любого другого инструмента отладки.

Схематично всю работу можно представить так:

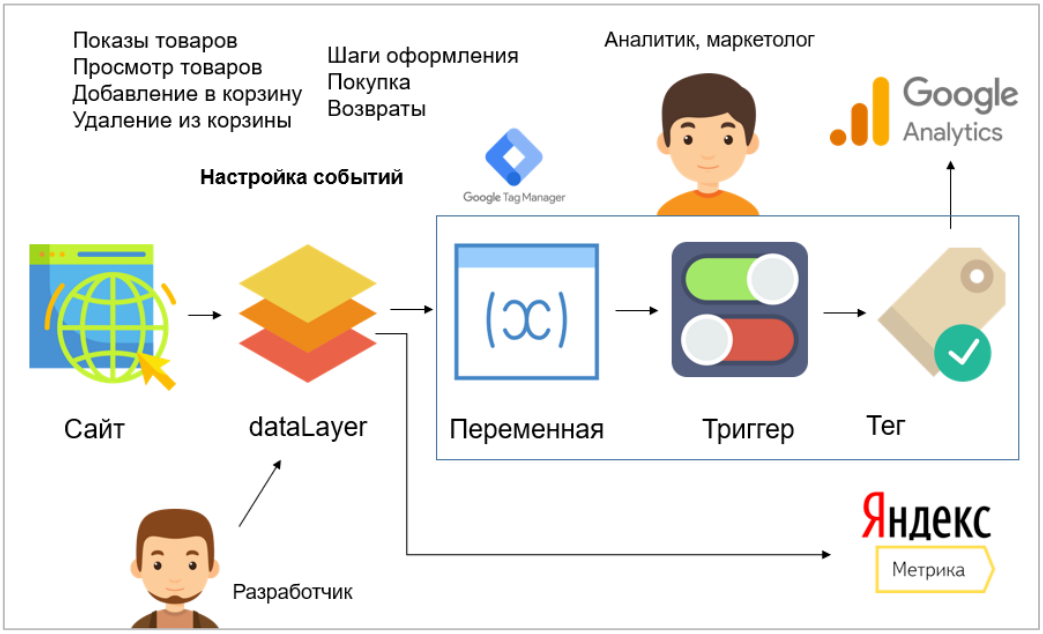


Рис. 760. Командная работа

Разработчик на сайте формирует уровень данных и настраивает те события, которые вы отобразили в ТЗ. В Google Analytics вы включаете отслеживание отчетов электронной торговли и с помощью диспетчера тегов Google создаете необходимые сущности (переменные, триггеры и теги) для передачи данных о заказах в инструменты веб-аналитики. И когда ваша работа, и работа программиста будет завершена, вы проверяете корректность передачи данных в Яндекс.Метрику, Google Analytics и другие аналитические сервисы. Если что-то не работает, совместными усилиями ищете причину и устраняете ошибки.

Пример технического задания по настройке расширенной электронной торговли для одного из моих проектов доступен по ссылке (см. приложение).

Сложность заключается в том, что в Enhanced Ecommerce много разных событий, и для полного отслеживания электронной торговли их нужно настроить на большее количество элементов сайта. Например, событие добавления товара в корзину (add) может срабатывать не только в карточке товара, но еще и по кнопкам по кнопке **Купить** на главной странице, в каталоге, и на других страницах или разделах, например, в блоке рекомендуемых товаров. Получается, что одно событие **addToCart** нужно добавить как минимум в три разных места на сайте.

Пример кода для этого события:

```
window.dataLayer = window.dataLayer || [];
dataLayer.push ({
  "ecommerce": {
    "currencyCode": "RUB",
    "add": {
      "products": [{
         "id": "{ID товара}",
        "name": "{Название товара}",
        "price": "{Цена товара}",
        "brand": "{Производитель}",
        "category": "{Категория товара}",
        "quantity": {Количество товара}
      }]
    }
  }
'event': 'gtm-ee-event',
'gtm-ee-event-category': 'Enhanced Ecommerce',
'gtm-ee-event-action': 'Product Add Cart',
'gtm-ee-event-non-interaction': 'False',
});
```

Уровень данных формируется, когда пользователь кликает по кнопке **Купить** конкретного товара на главной странице или **Купить** на карточке самого товара.

Переменные в фигурных скобках **{ }** – динамические, то есть в них подставляются разные значения в зависимости от действий пользователя в конкретный момент времени. Выбрал один товар – подставились одни данные, выбрал другой – другие данные и т.д. Вот как выглядит сформированный dataLayer для интернет-магазина на OpenCart, которые разработчик передает на странице успешного заказа (событие **purchase**):

```
        <div class="pull-right"><a href="{{ continue }}" class="bt
      </div>
      {{ content_bottom }}</div>
    {{ column_right }}</div>
</div>
{{ footer }}

{% if order_id %}
  <script type="text/javascript">
    window.dataLayer = window.dataLayer || [];
    dataLayer.push({
     'ecommerce': {
       'currencyCode': 'RUB',
       'purchase': {
       'actionField': {
           'id': "{{ order_id }}",
           'revenue': "{{ total }}",
           'shipping': "500",
           },
           'products': [
             {% for product in products %}
               {
                 "id": "{{ product.product_id }}",
                 "name": "{{ product.name }}",
                 "price": "{{ product.pr }}",
                 "brand": "{{ product.manufacturer }}",
                 "category": "{{ product.category }}",
                 "quantity": "{{ product.quantity }}"
               },
             {% endfor %}
             ]
         }
     },
     'event': 'gtm-ee-event',
     'gtm-ee-event-category': 'Enhanced Ecommerce',
     'gtm-ee-event-action': 'Purchase',
     'gtm-ee-event-non-interaction': 'False',
    });
  </script>
{% endif %}
```

Рис. 761. Уровень данных на странице успешного заказа (purchase)

Для другого проекта и CMS-движка сайта код выделенных переменных будет другим. Да, мы можем использовать GTM и DOM Scraping. Но как я писал ранее, не рекомендуется использовать извлечение информации со страницы сайта для формирования уровня данных, поскольку любое изменение CSS-селекторов элементов со стороны верстальщика приведет к сбою в отслеживании. Особенно для расширенной электронной торговли, где необходимо сформировать не один уровень данных.

Интернет-маркетолог не отвечает за формирование данных на стороне сервера. За это отвечает программист. А вот извлечь значения из этих данных с помощью Google Tag Manager он в состоянии. Чтобы это сделать, необходимо:

- включить dataLayer в Яндекс.Метрике и расширенную электронную торговлю в Google Analytics (+ шаги последовательности)
- создать 3 пользовательские переменные;
- добавить 1 триггер активации;
- настроить 1 тег для Google Analytics.

Включить опцию электронной торговли в Яндекс.Метрике можно в настройках счетчика:

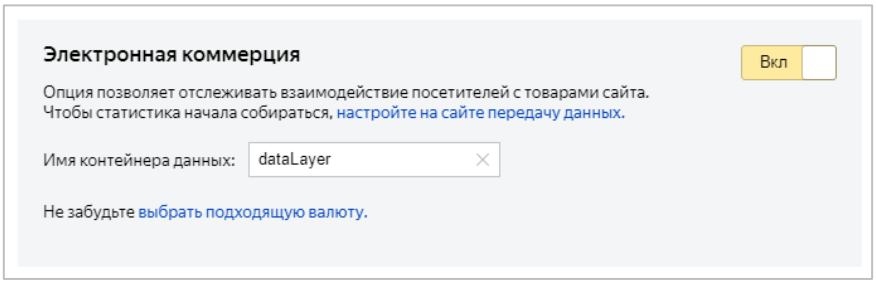


Рис. 762. Включение электронной торговли в Яндекс.Метрике

Имя контейнера данных **dataLayer** оставьте без изменений.

В Google Analytics на уровне представления включите отслеживание электронной торговли, и при необходимости, задайте шаги последовательности (step):

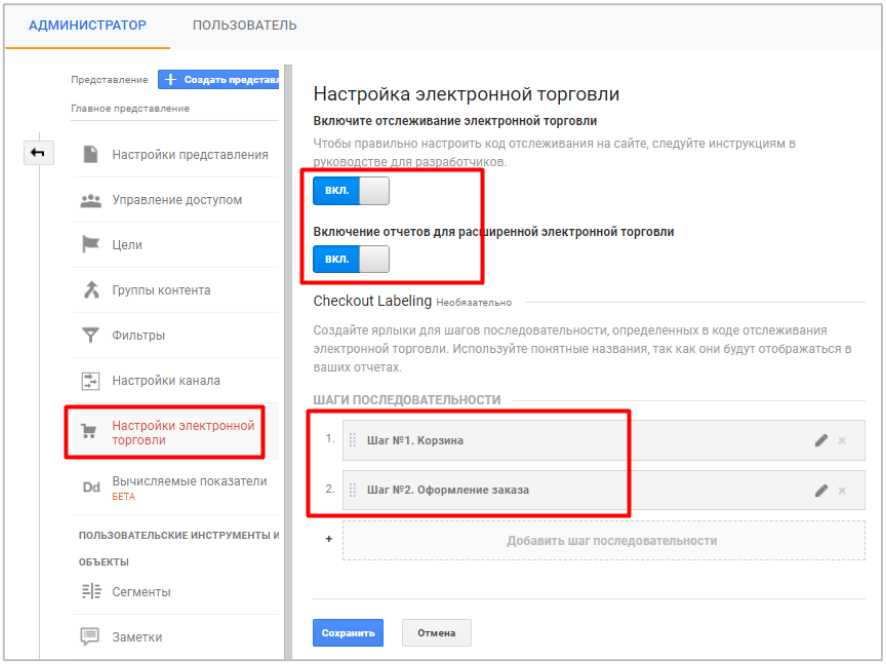


Рис. 763. Включение отчетов для расширенной электронной торговли + шаги последовательности

В Google Tag Manager создайте три переменных типа **Переменная уровня данных**: **gtm-ee-event-action, gtm-ee-event-category** и **gtm-ee-event-non-interaction**.

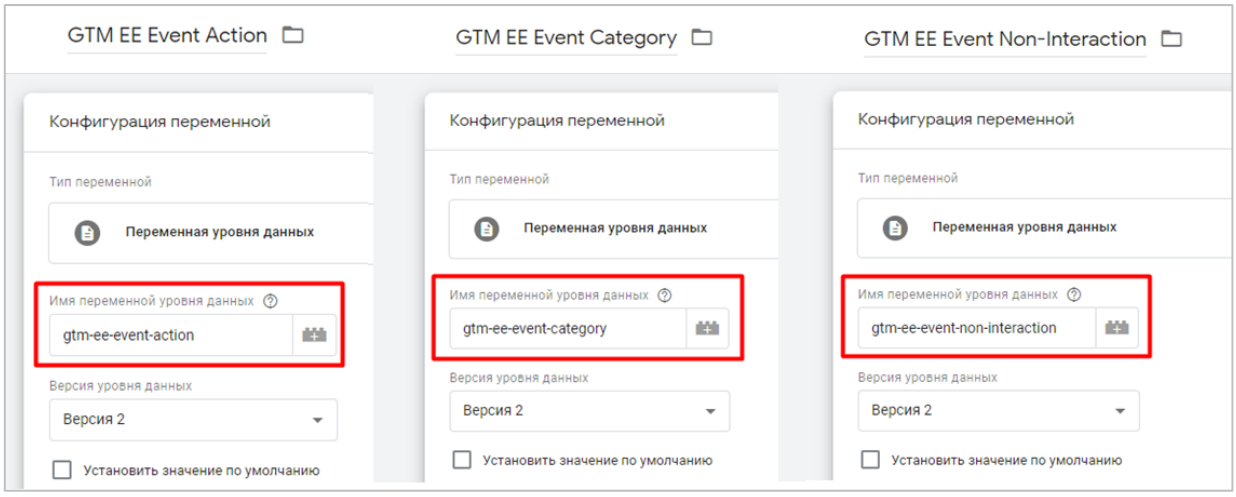


Рис. 764. Переменные уровня данных

Добавьте триггер типа Пользовательское событие с именем **gtm-ee-event**:

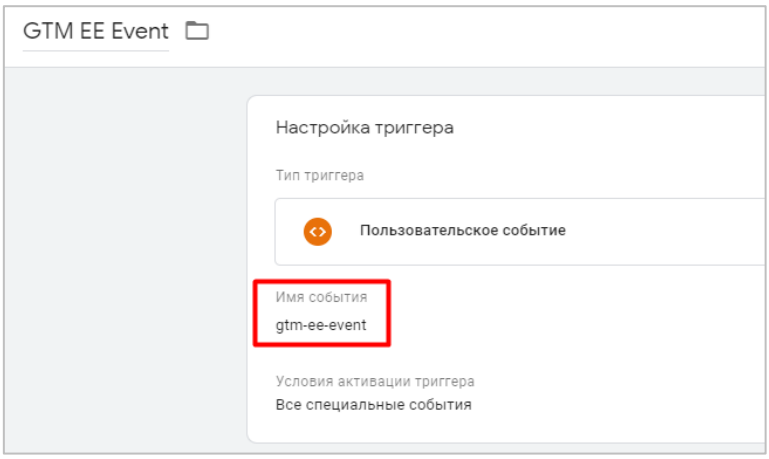


Рис. 765. Пользовательское событие gtm-ee-event

На заключительном шаге создайте тег **Google Аналитика – Universal Analytics** с типом отслеживания **Событие** со следующими настройками:

- **Категория** - {{GTM EE Event Category}}
- **Действие** - {{GTM EE Event Action}}
- **Не взаимодействие** - {{GTM EE Event Non-Interaction}}
- Включить переопределение настроек в этом теге (указать идентификатор отслеживания GA)
- Дополнительные настройки – Электронная торговля – Включить расширенные функции электронной торговли – True - Использовать уровень данных
- **Триггер активации** – GTM EE Event

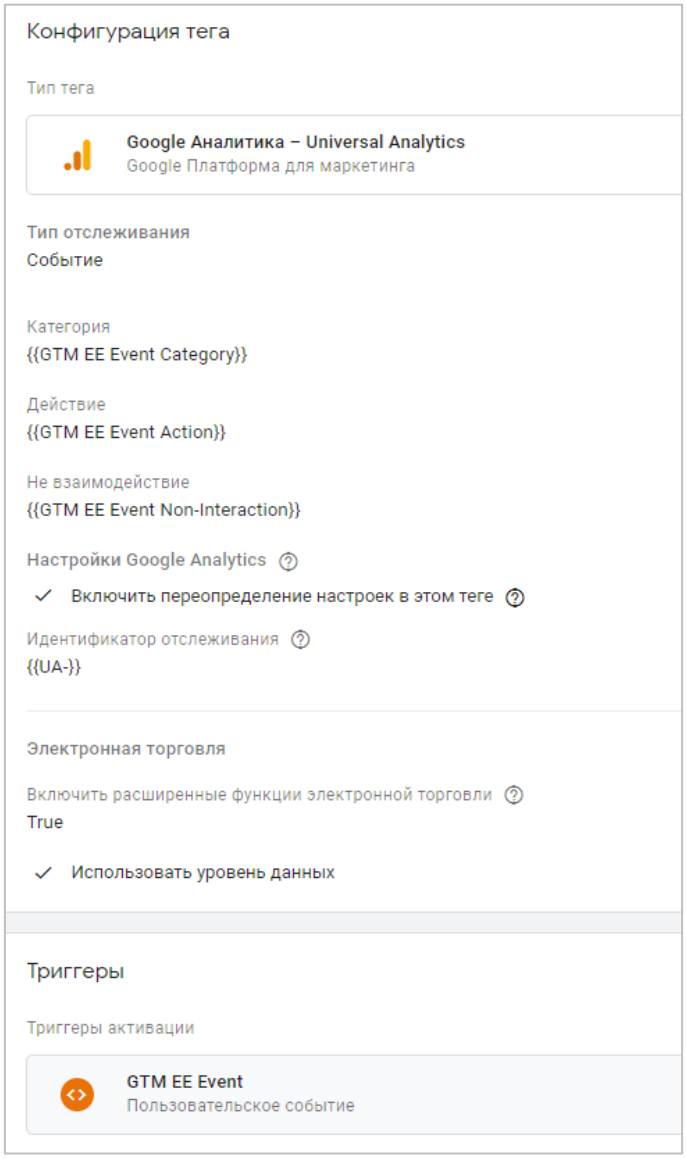


Рис. 766. Настройка тега Enhanced Ecommerce

Почему так называются переменные и триггер (gtm-ee-...)?

Все дело в общем **теге** событий, с помощью которого настраивается расширенная электронная торговля. Руководство, которое опубликовала компания Neatpeak в своем блоге (см. приложение), стало стандартом настройки Enhanced Ecommerce для всего русскоязычного сообщества.

Пример конструкции общего тега событий события **purchase (покупка)** на странице успешно оформленного заказа выглядит следующим образом:

```
{% if order_id %}
  <script type="text/javascript">
    window.dataLayer = window.dataLayer || [];
    dataLayer.push({
     'ecommerce': {
       'currencyCode': 'RUB',
       'purchase': {
       'actionField': {
           'id': "{{ order_id }}",
           'revenue': "{{ total }}",
           'shipping': "500",
           },
           'products': [
             {% for product in products %}
               {
                 "id": "{{ product.product_id }}",
                 "name": "{{ product.name }}",
                 "price": "{{ product.pr }}",
                 "brand": "{{ product.manufacturer }}",
                 "category": "{{ product.category }}",
                 "quantity": "{{ product.quantity }}"
               },
             {% endfor %}
             ]
       }
     },
     'event': 'gtm-ee-event',
     'gtm-ee-event-category': 'Enhanced Ecommerce',
     'gtm-ee-event-action': 'Purchase',
     'gtm-ee-event-non-interaction': 'False',
    });
  </script>
{% endif %}
```

Рис. 767. Общий тег событий

Из предыдущей главы вы узнали, что для упрощения работы с контейнером и для сокращения количества настраиваемых триггеров и тегов можно использовать универсальный способ отслеживания, задав везде одно и тоже имя события, а в качестве остальных переменных (**Категорие**, **Действие** и **Не взаимодействие**) передавать разные значения внутри переменных уровнях данных. Например, на рисунке выше для события покупки (**purchase**) принимаются следующие значения:

- event - gtm-ee-event;
- gtm-ee-event-category: Enhanced Ecommerce;
- gtm-ee-event-action: Purchase;
- gtm-ee-event-non-interaction: False.

А для события показа товаров (**impressions**) при такой конструкции можно использовать уже другие, но оставить название события прежним:

- event - gtm-ee-event;
- gtm-ee-event-category: Enhanced Ecommerce;
- gtm-ee-event-action: Product Impressions;
- gtm-ee-event-non-interaction: True.

Таким образом, благодаря общему тегу и переменным уровня данных нужно будет создать меньшее количество переменных, триггеров и тегов.

Пример: на странице успешно оформленного заказа в режиме предварительного просмотра GTM на шкале событий должно срабатывать **gtm-ee-event**, а на вкладке **Data Layer** уровень данных для конкретного события (add, remove, impressions, purchase и т.д.) с корректными значениями свойств объекта (передается идентификатор товара, название, цена, категория, общая сумма покупки и т.д.)

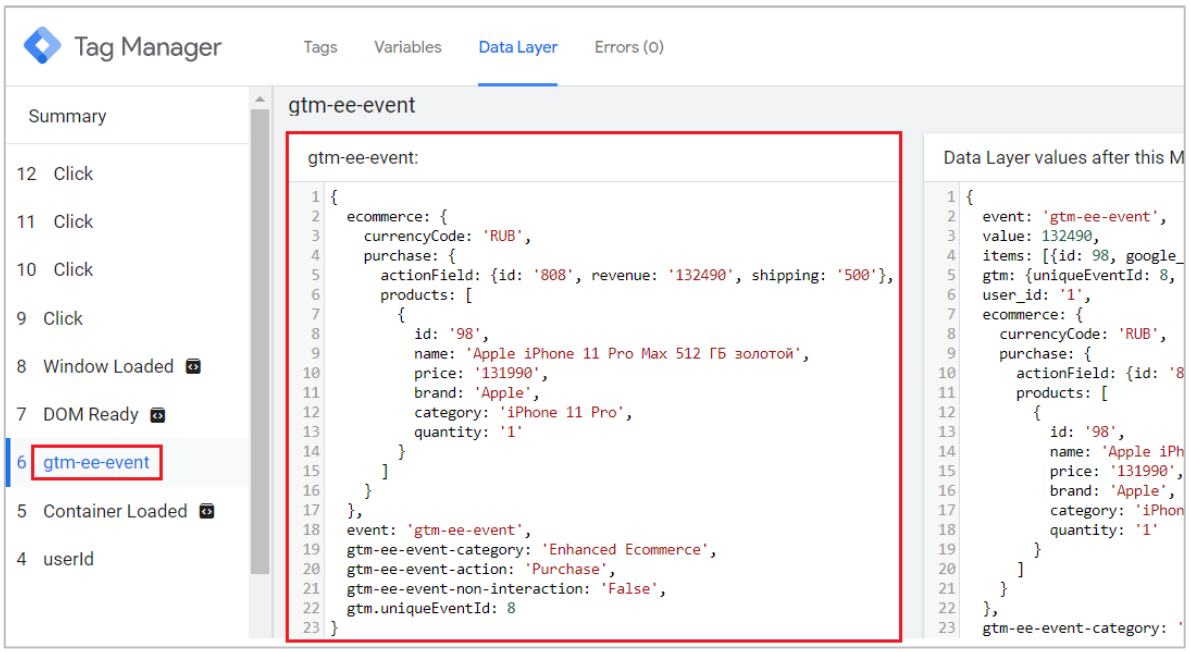


Рис. 768. Событие gtm-ee-event, уровень данных для purchase (покупка)

Примечание: для настройки расширенной электронной торговли необязательно создавать все 9 отслеживамых действий. Вы можете определить для себя наиболее важные события и попросить разработчика под них сформировать уровни данных на соответствующих страницах и элементах сайта.

Поскольку действия электронной торговли настраиваются как события, то через некоторое время все данные по событиям появятся в отчете Google Analytics **Поведение — События — Лучшие события**, в разделе **Конверсии – Электронная торговля**, и в других отчетах при выборе конверсии **Электронная торговля**. В Яндекс.Метрике данные по заказам будут доступны в разделе **Стандартные отчеты – Электронная коммерция.**

Если у вас есть несколько шагов по воронке последовательности, то в отчете вы увидите визуализации последовательности перехода от одного шага к другому:

Настройка User ID

Для связывания разных устройств одного пользователя в Google Analytics существует функция **User ID**. Она позволяет объединить различные сеансы и действия во время этих сеансов с уникальным идентификатором. Это отслеживание еще называют *кросс-девайсным*.

Например, сначала человек зашел к нам на сайт из дома с персонального компьютера, потом в течение дня на работе посещал интернет-магазин с телефона, а вечером по приезду домой оформил покупку с ноутбука.

Как мы уже с вами узнали раннее, сбор данных о посетителях на сайте с помощью Analytics основан на файлах cookie, когда данные по каждому пользователю принадлежат только одной конфигурации браузера и устройства (1 конкретный браузер - 1 конкретное устройство = 1 куки файл). И отследить поведение посетителя с нескольких девайсов, связав их в одного уникального пользователя, при классическом подходе не представляется возможным.

С помощью кросс-девайс отслеживания (функции **User ID**) мы можем связать всего его посещения воедино, при условии, что пользователь на всех трех устройствах был идентифицирован нашей CRM-системой или CMS-движком. Это позволит оценить какие посетители конвертируются лучше всего: те, кто используют только десктопы, только мобильные или оба типа устройств.

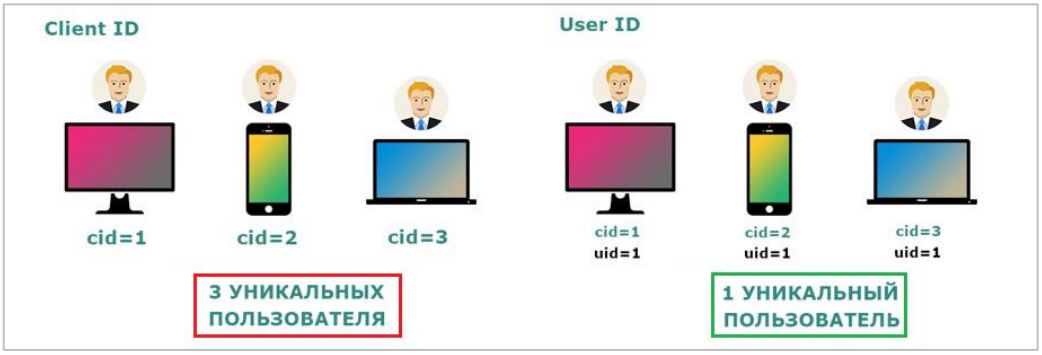


Рис. Данные о просмотре страниц передаются

Рис. 769. Схема работы Client ID и User ID

В основе этой модели лежит идентификация пользователя через его профиль, личный кабинет. Мы, как владельцы ресурса, побуждаем человека зарегистрироваться на нашем сайте, создать свой профиль, а все дальнейшие действия (просмотры страниц, оформление покупок и т.д.) просим выполнять через свою учетную запись. Таким образом, когда пользователь заходит на сайт, его данные сохраняются в базе и отправляются на сервер, чтобы в дальнейшем использовать их для сопоставления заходов на различных устройствах и браузерах.

Классический пример такого подхода – социальная сеть. Когда вы впервые зарегистрировались, например, ВКонтакте, вам присвоили уникальный ID учетной записи.

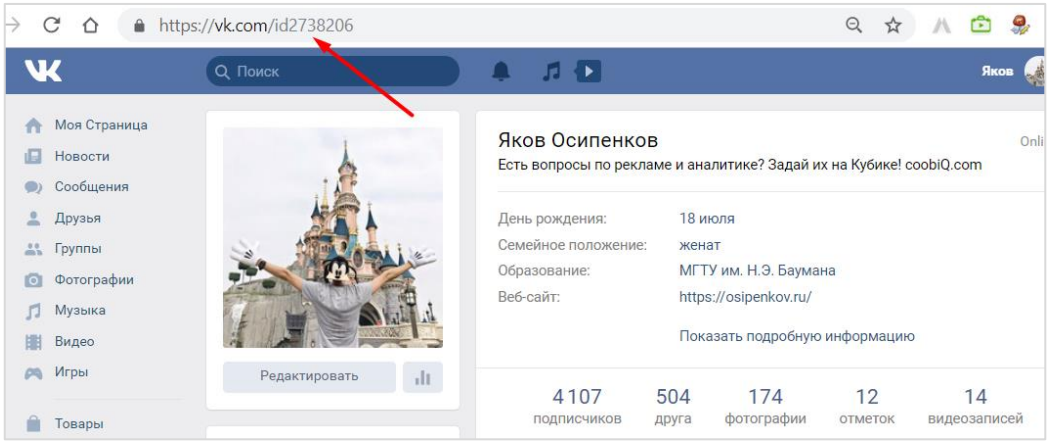


Рис. 770. ID профиля ВКонтакте – наглядный пример User ID

С этого момента, чтобы вы не делали (смотрели видео, искали группы, добавляли в друзья, писали сообщения) – все эти действия будут связаны с вашим профилем. А поскольку вы регистрировались через телефон или e-mail (да еще и вводили все данные в настройках), у администрации социальной сети есть полный набор ваших данных. А если еще и включена функция геолокации, то они не только знают о вас все, но и могут отследить ваше местонахождение в любой момент времени. И уже абсолютно не имеет никакого значения, с какого устройства вы заходите и с какого браузера, главное – вы под своей учетной записью.

Если у вас нет возможности в рамках своего проекта определить пользователя путем его идентификации через учетную запись или авторизацию через социальные сети (это не предусмотрено функционалом сайта), то и **User ID** настраивать большого смысла нет. Системы веб-аналитики сами умеют отслеживать взаимодействия пользователей на различных устройствах с помощью встроенных алгоритмов.

В 2018 году Google добавил функцию **Google Signals**, которая позволяет на основе агрегированных данных о пользователях, включивших **Персонализацию рекламы**, создавать общую модель поведения на нескольких устройствах. При этом учитываются данные о пользователях, а не о сеансах. Для создания модели поведения не требуется представления User ID. А в середине 2017 года Яндекс запустил свою бета версию отчета **Кросс-девайс**, в котором отображается статистика по конверсиям клиентов, которые заходили на сайт с нескольких устройств.

В отличие от **Client ID**, который задается на стороне системы, значение **User ID** мы назначаем и передаем в Google Analytics самостоятельно. Он должен быть сгенерирован на сайте системой аутентификации пользователей с помощь логина, под которым он заходит. Отсюда вывод: то, что представлено в справке разработчика в виде примеров кода при полном копировании на свой сайт работать не будет!

Чтобы активировать **User ID** в Google Analytics, необходимо:

- включить эту функцию;
- создать новое представление User ID;
- добавить специальный параметр.

Переходим в Analytics и в колонке *Ресурс* выбираем **Отслеживание – User-ID**:

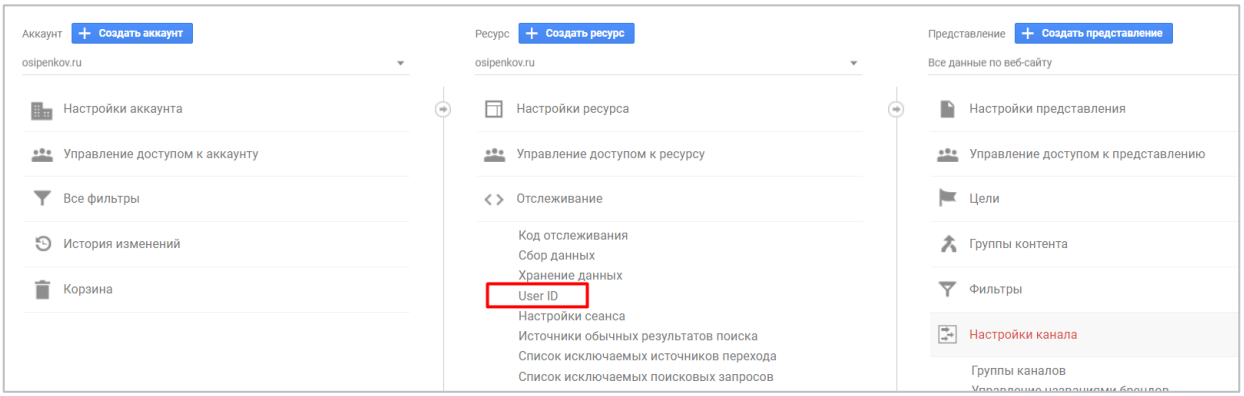


Рис. 771. Ресурс – Отслеживание – User-ID

Принимаем правила в отношении User-ID, переключаем ползунок в позицию **«Вкл.»** и нажмите *Далее*:

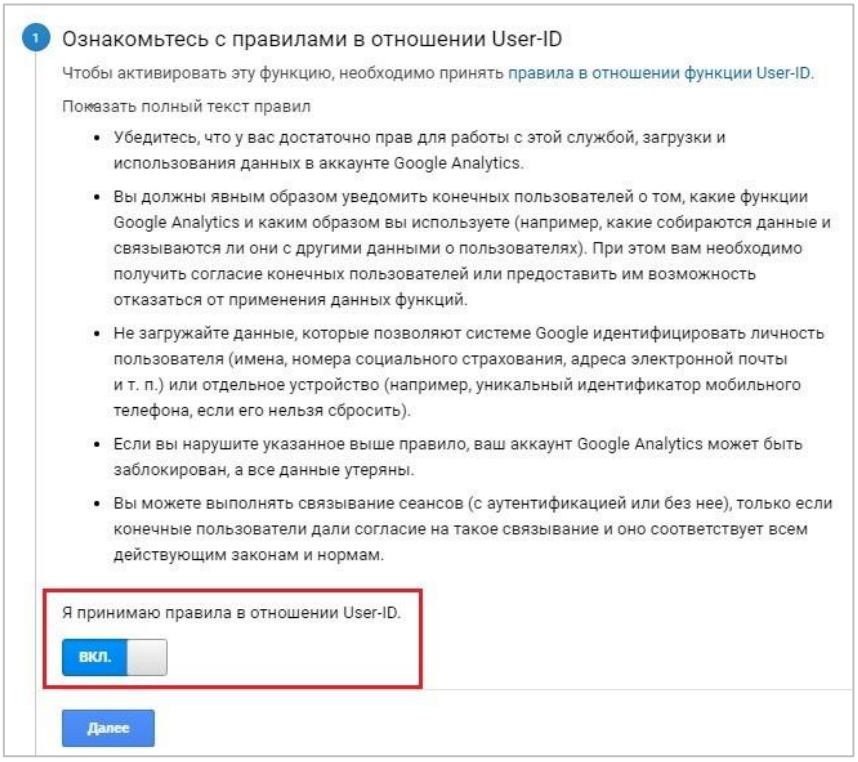


Рис. 772. Правила в отношении User-ID

На следующем шаге приведен пример классического кода для Universal Analytics (библиотеки *analytics.js*) и глобального тега Global Site Tag (библиотеки *gtag.js*).

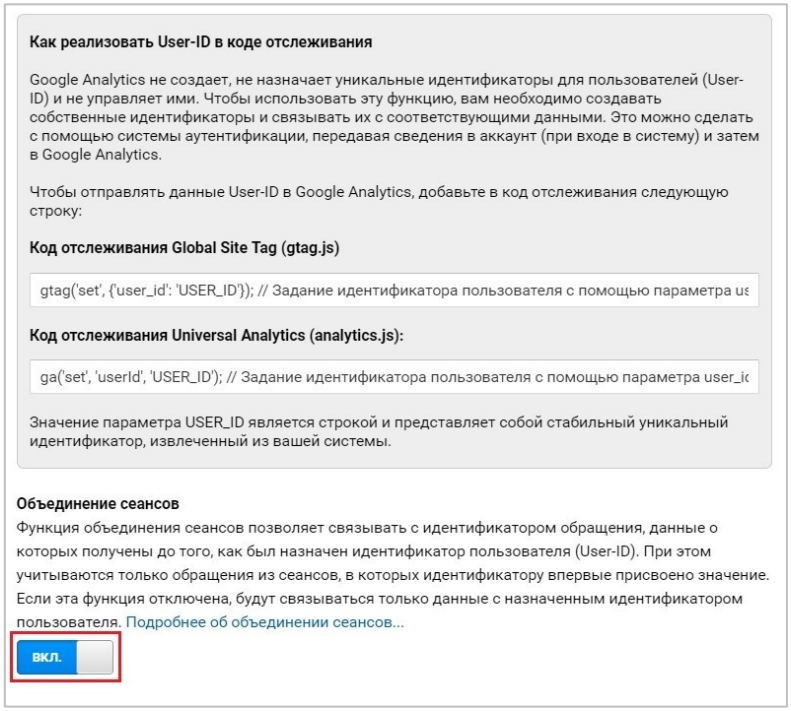


Рис. 773. Примеры кодов отслеживания Universal Analytics и Global Site Tag

На последнем шаге создаем новое представление User ID, введя его название и указав часовой пояс.

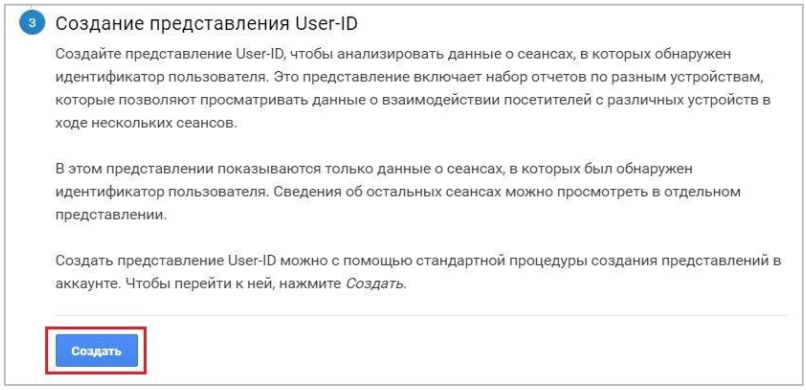


Рис. 774. Создание представления User-ID

В этом представлении будут отображаться данные о сеансах, в которых был обнаружен идентификатор пользователя.

Передачу собственных uid с вашего сайта можно осуществить несколькими способами:

- с помощью кодов отслеживания *gtag.js* или *analytics.js;*
- с помощью Google Tag Manager.

Веб-аналитика, а если быть точнее, технические настройки тем и хороши, что у них все два состояния - либо работает, либо нет (0 и 1, true или false). Поэтому если вы все настроили, все корректно собирается, не возникло никаких проблем с отслеживанием, то подойдет любое решение, удовлетворяющее начальным условиям о сборе.

Это относится и к User ID. Например, можно попробовать извлечь значение идентификатора пользователя с помощью переменной **Элемент DOM**, или из cookie, при условии, что такая кука установлена на сайте и в нее передается значение User ID пользователя. А можно привлечь разработчика, чтобы он написал функцию, которая бы возвращала ID текущего авторизованного пользователя.

Существуют разные варианты отслеживания. Все зависит от веб-сайта, с которым вы работаете, команды, возможностей вашего заказчика и ряда других факторов. Описанный ниже способ является наиболее предпочтительным, универсальным и надежным.

Попросите разработчика передать идентификатор пользователя на уровень данных, взяв за основу нижеприведенный код:

```
<script type="text/javascript">
window.dataLayer = window.dataLayer || [];
window.dataLayer.push({'user_id':'{ID пользователя на сайте}'});
</script>
```

, где **ID пользователя на сайте** означает функцию, возвращающую ID залогинненого пользователя.

Желательно, чтобы этот код был размещен над контейнером Google Tag Manager, чтобы его можно было использовать с триггером **Просмотр страницы**. Вот так выглядит проверка (залогинен ли пользователь?) для интернет-магазина на платформе OpenCart:

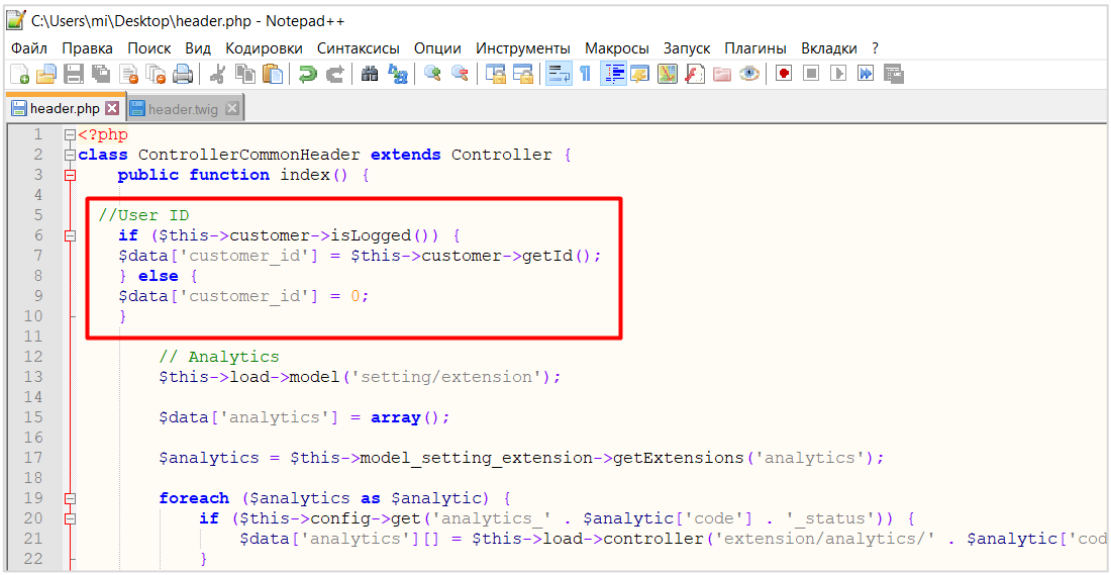


Рис. 775. Пример проверки авторизации на сайте

А сам уровень данных выглядит вот так:

```
<script type="text/javascript">
window.dataLayer = window.dataLayer || [];
window.dataLayer.push(
{'event':'userId',
'user_id':'{{customer_id}}'
});
</script>
```

```
<script type="text/javascript">
window.dataLayer = window.dataLayer || [];
window.dataLayer.push({
'event':'userId',
'user_id':'{{customer_id}}'
});
</script>
```

Рис. 776. Пример уровня данных для User ID

где в **{{customer_id}}** передается идентификатор авторизованного пользователя.

Пример технического задания по настройке User ID для одного из моих проектов доступен по ссылке (см. приложение).

Схематично всю работу можно представить так:

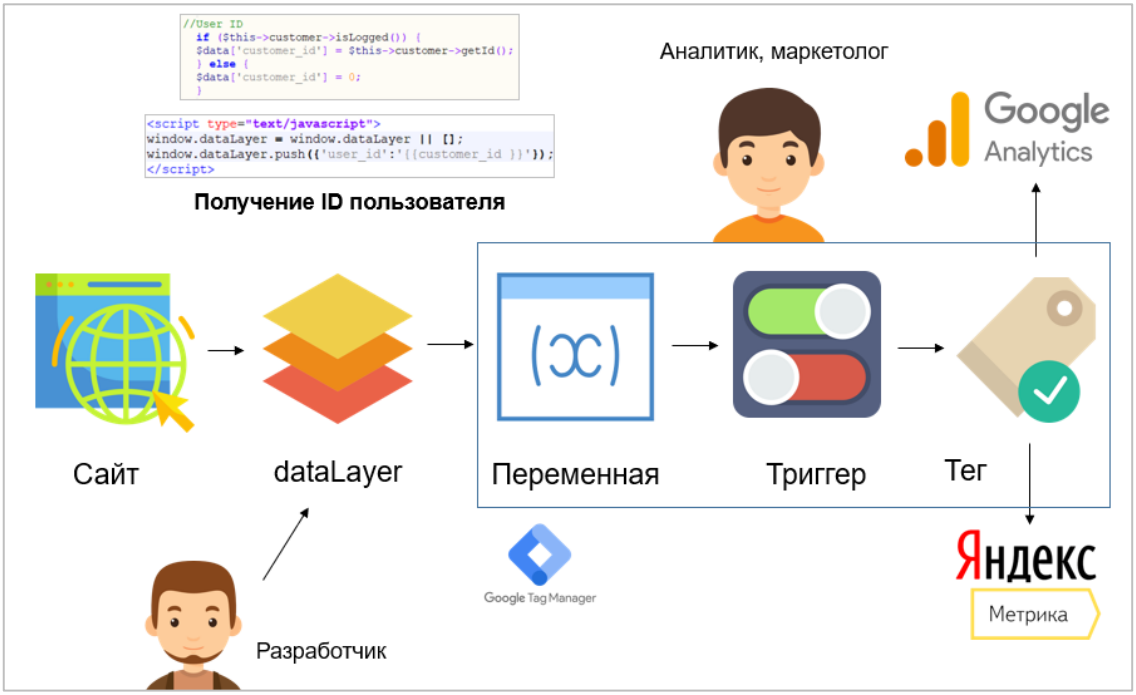


Рис. 777. Командная работа

Разработчик на сайте формирует уровень данных, который вы отобразили в ТЗ. В Google Analytics вы включаете создаете новое представление User-ID и с помощью диспетчера тегов Google создаете необходимые сущности (переменные, триггеры и теги) для передачи данных об идентификаторе пользователя. И когда ваша работа, и работа программиста будет завершена, вы проверяете корректность передачи данных в Яндекс.Метрику, Google Analytics и другие аналитические сервисы. Если что-то не работает, совместными усилиями ищете причину и устраняете ошибки.

После внедрения кода разработчиком проверьте передачу данных, авторизовавшись на сайте. Если все хорошо, в режиме отладки GTM вы увидите срабатывающее событие **userId** и значение идентификатора пользователя на сайте (user_id - 290) в переменной уровня данных на вкладке **Data Layer**:

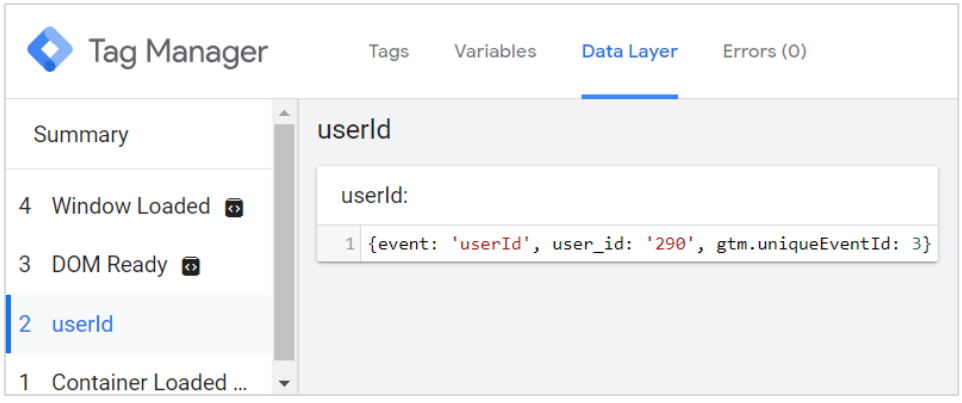


Рис. 778. Проверка в режиме предварительного просмотра

На рисунке выше срабатывает событие **userId** после загрузки контейнера **Container Loaded (gtm.js)**. Это означает, что в если мы будем использовать стандартный тег Google Analytics с триггером **All Pages (Все страницы)**, то переменная **userId** не будет еще определена, и вместо значения ID пользователя ей будет присвоено **undefined**, а значение User ID не будет передано в Google Analytics:

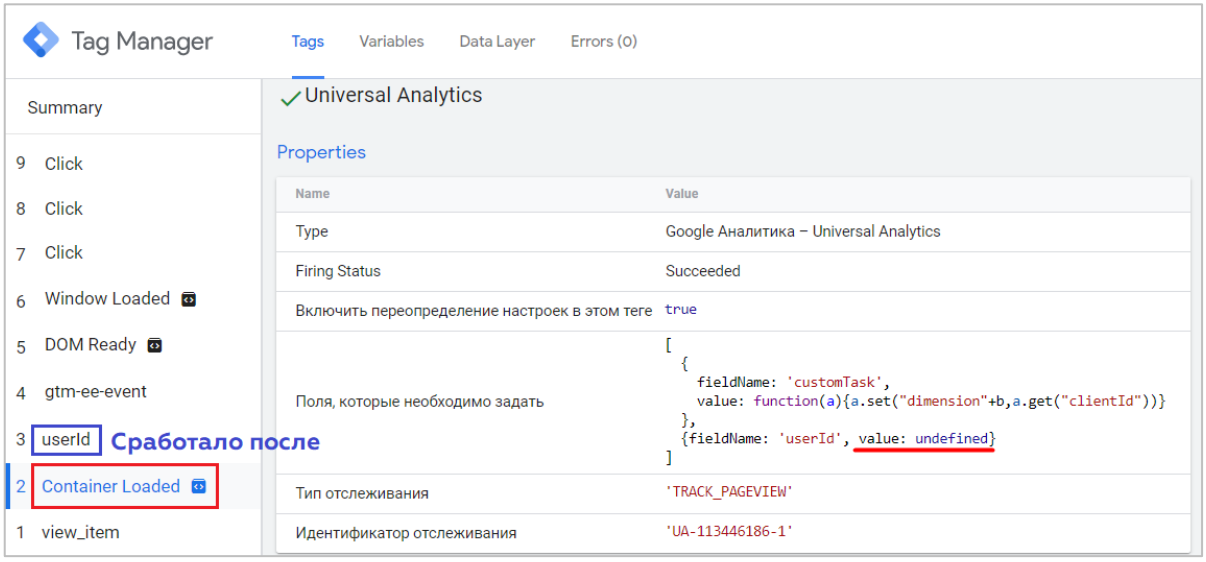


Рис. 779. userId сработал после Container Loaded, поэтому переменная еще не определена (undefined)

Тогда для запуска тега вы можете воспользоваться триггером **Модель DOM готова** или **Окно загружено**, либо же попросить разработчика разместить код с User ID над контейнером Google Tag Manager, как можно ближе к тегу **<head>**. В этом случае событие **userId** сработает до загрузки контейнера **Container Loaded** и в поле, которое необходимо задать, передастся идентификатор пользователя, который он имеет на нашем сайте:

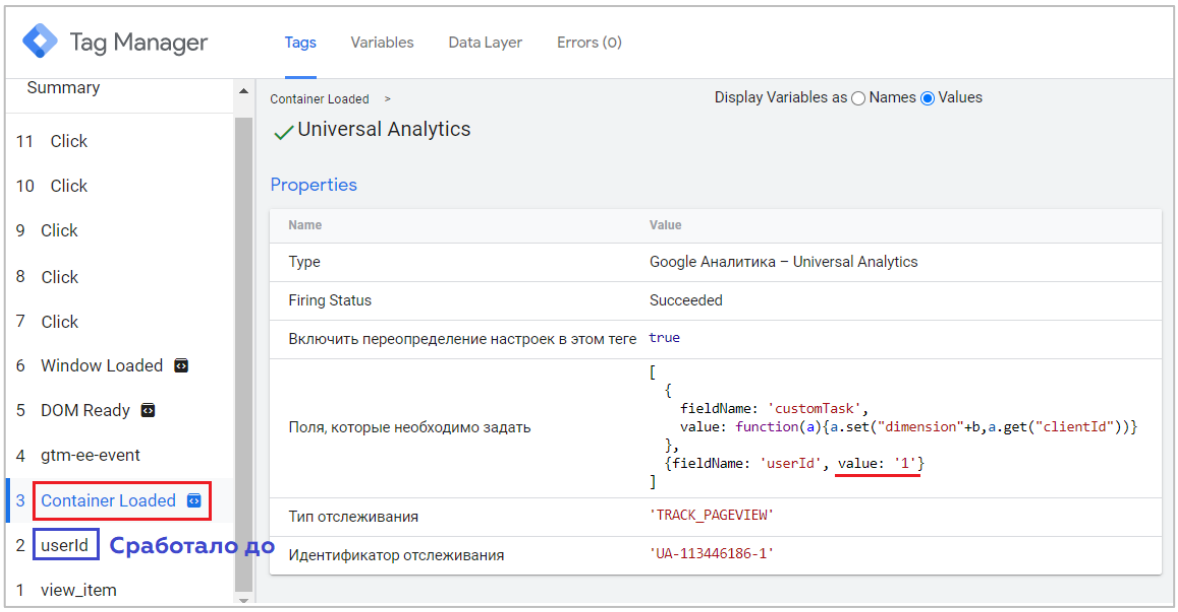


Рис. 780. userId сработал до Container Loaded, поэтому переменная определена и имеет корректное значение

Для того, чтобы это реализовать, сначала в диспетчере тегов Google следует создать переменную уровня данных. Имя переменной вводите такое же, как и в коде уровня данных, который вы отправляли разработчику (не путать с названием события!):

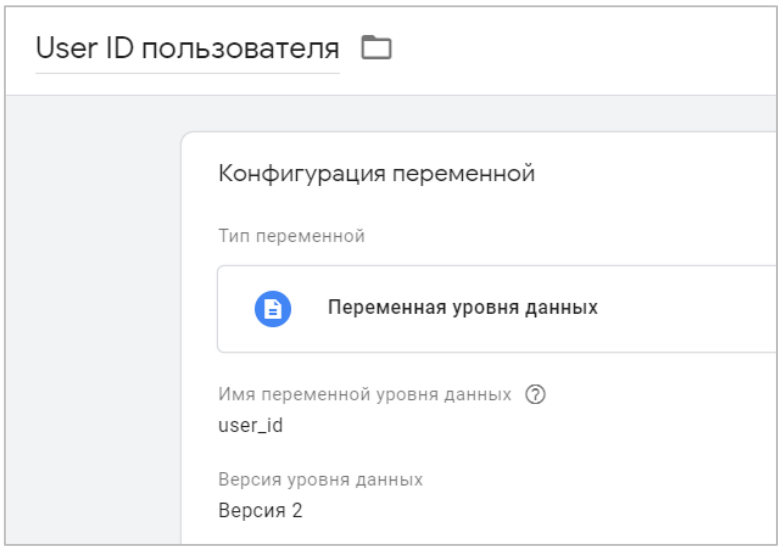


Рис. 781. Пример переменной уровня данных

Не забывайте про правила использования функции User ID в Google Analytics, а именно:

- у вас должно быть достаточно прав для работы с этим сервисом, в том числе все необходимые разрешения от владельцев прав на информацию, передаваемую с помощью вашей платформы, и правообладателей соответствующего аккаунта Google Аналитики.
- вы должны надлежащим образом уведомлять конечных пользователей о том, какие функции Google Аналитики и каким образом вы используете, в том числе о том, какие данные вы собираете и связываете ли вы их с другими доступными вам сведениями о конечных пользователях. Необходимо получить согласие конечных пользователей или предоставить им возможность отказаться от применения данных функций.
- не загружайте данные, позволяющие идентифицировать личность пользователя (имена, номера социального страхования, адреса электронной почты и т. п.) или отдельное устройство (например, уникальный идентификатор мобильного телефона, если его нельзя сбросить).
- вы можете связывать сеансы с аутентификацией и без нее только в том случае, если конечные пользователи дали согласие на такое связывание, и оно соответствует всем действующим законам и нормам.

Нарушение этих правил может привести к прекращению действия вашего аккаунта Google Аналитики и потере данных. Подробнее о правилах использования читайте в официальной документации Google (см. приложение).

Другими словами: если на ваш сайт заходят пользователи из стран, входящих в Европейскй союз, и вы планируете хранить их идентификаторы в файле cookie и отслеживать даже неаутентифицированных пользователей с идентификатором, то вам необходимо получить их согласие (General Data Protection Regulation, GDPR), а также обсудить все отслеживания с юридическим отделом и действующим законодательством тех стран, жители которых посещают ваш сайт.

Теперь необходимо передать User ID при каждом обращении Google Analytics. Чтобы не обновлять каждый тег вручную, откройте переменную типа **Настройки Google Analytics** и в разделе **Дополнительные настройки – Поля, которые необходимо** задайте **userId** и укажите в **Название поля – userId**, а в **Значение** добавьте переменную уровню данных, которую создали на предыдущем шаге:

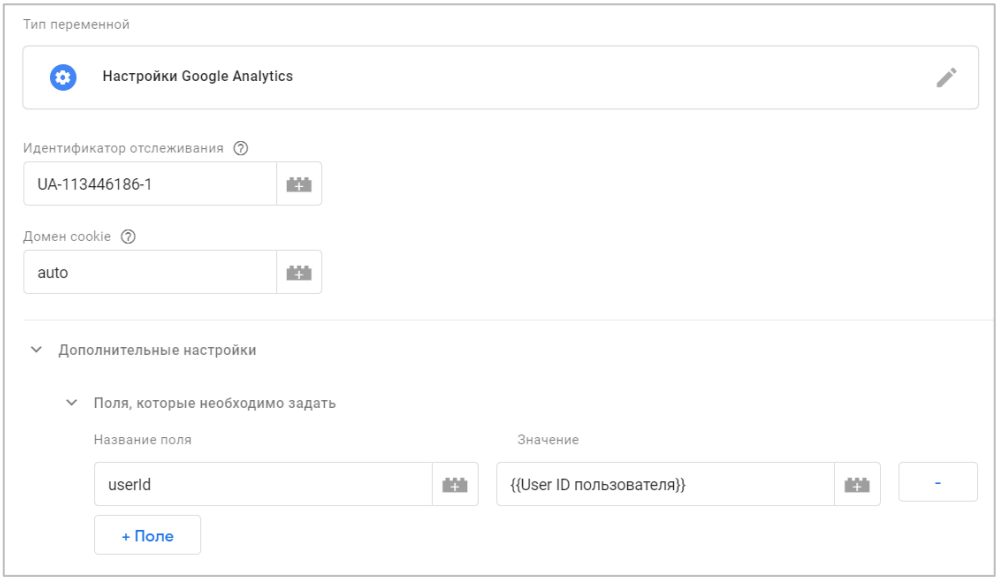


Рис. 782. Поля, которые необходимо задать - userId

Так нужно делать для того, чтобы видеть все обращения (события, просмотры страниц и т.д.) в отдельном представлении Google Analytics (User-ID). Поэтому мы настраиваем идентификатор пользователя Google Analytics на уровне переменной **Настройки Google Analytics**, а не только в теге просмотра страницы Google Analytics. В этом случае было бы достаточно этого:

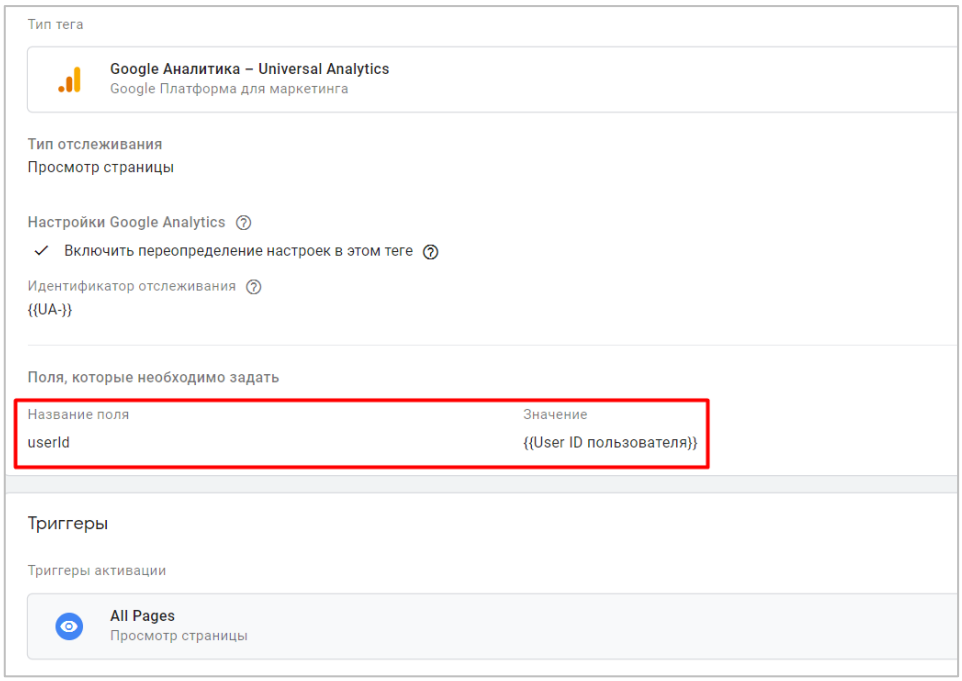


Рис. 783. Настройка тега

Можно пойти еще дальше и передать User ID в Google Analytics в качестве специального параметра. Для этого перейдите на уровне ресурса в раздел **Пользовательские определения – Специальные параметры** и нажмите **+ СПЕЦИАЛЬНЫЙ ПАРАМЕТР**:

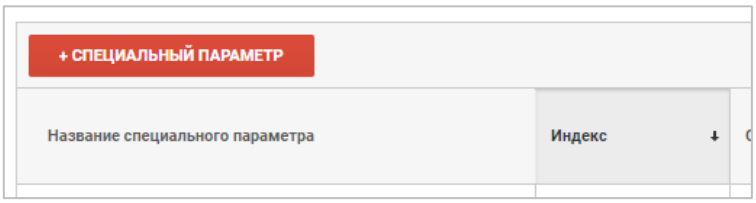


Рис. 784. Создание специального параметра

Введите название пользовательского параметра, выберите **Область действия – Пользователь**, оставьте галочку **Активная** и нажмите **Создать**:

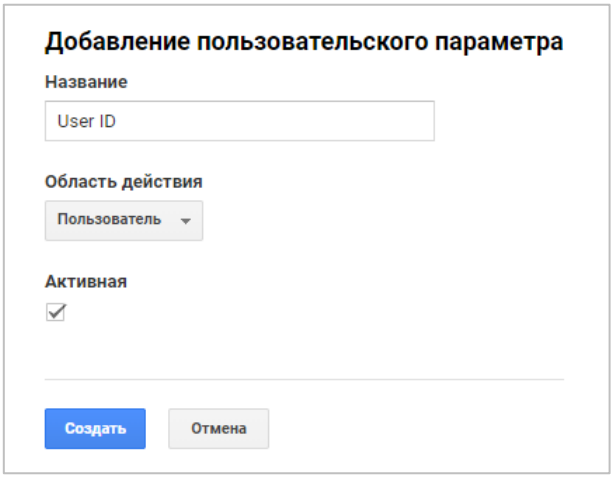


Рис. 785. Настройка специального параметра User ID

Запомните индекс пользовательского параметра (в моем примере – 1):

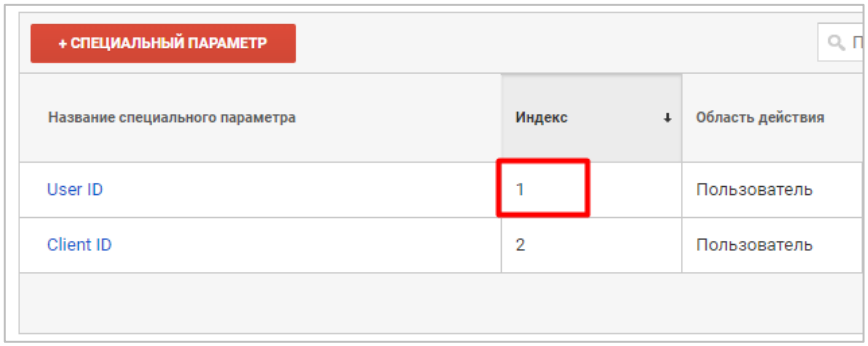


Рис. 786. Индекс специального параметра

Вернитесь в тег Google Analytics / переменную **Настройки Google Analytics** и в разделе **Дополнительные настройки – Специальные параметры** задайте **Индекс** и укажите **Значение параметра**:

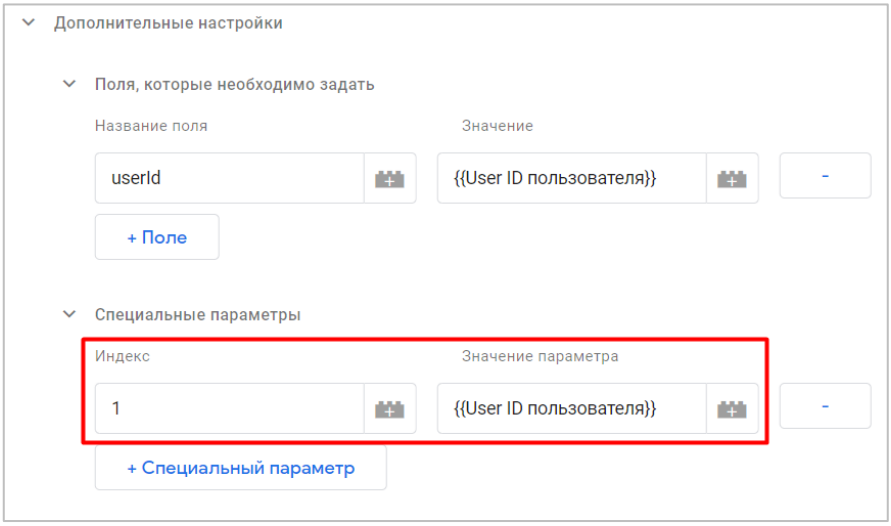


Рис. 787. Передача специального параметра в теге или переменной Настройки Google Analytics

Сохраните настройки. Проверить корректность передачи данных можно с помощью расширений для браузера Google Chrome Google Tag Assistant, GA Debugger и других.

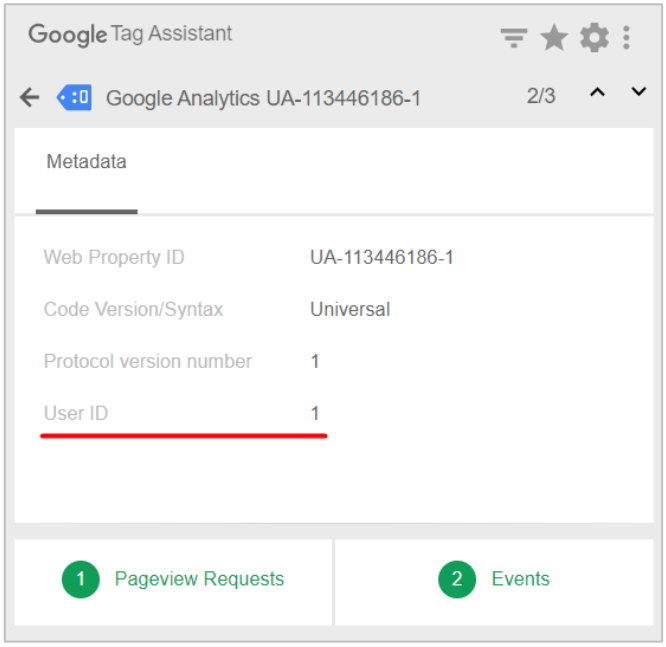


Рис. 788. Определение User ID в Google Tag Assistant

А кликнув на **Pageview Requests** и перейдя на вкладку **Custom Metrics**, мы можем посмотреть, верно ли передается специальный параметр, который мы создали:

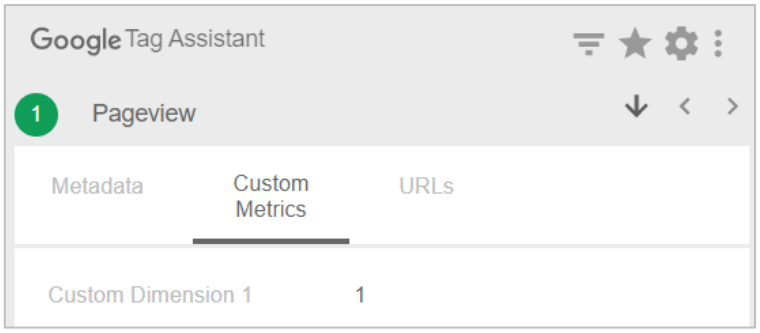


Рис. 789. Вкладка Custom Metrics

Для отправки идентификатора пользователя в Яндекс.Метрику используйте метод **userParams**, который позволяет передавать произвольные параметры посетителей сайта, и параметр **UserID**. Пример конструкции выглядит так:

```
ym(XXXXXX, 'userParams', {UserID: {{User ID пользователя}}});
```

, где **XXXXXX** – номер счетчика Яндекс.Метрики, а **User ID пользователями** – переменная уровня данных, созданная на предыдущем шаге.

В Google Tag Manager:

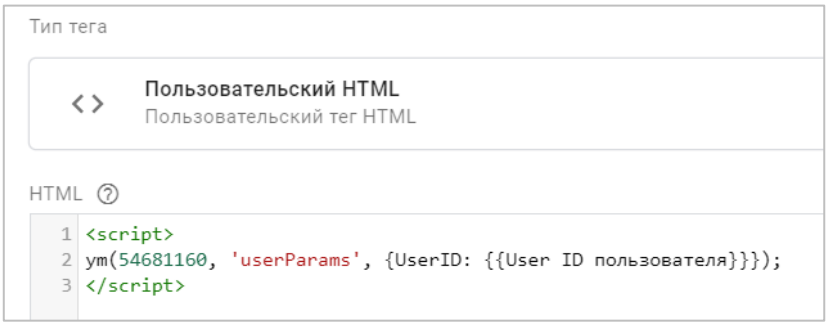


Рис. 790. Передача User ID в Яндекс.Метрику

Данные по User ID доступны в отчете **Параметры посетителей**:

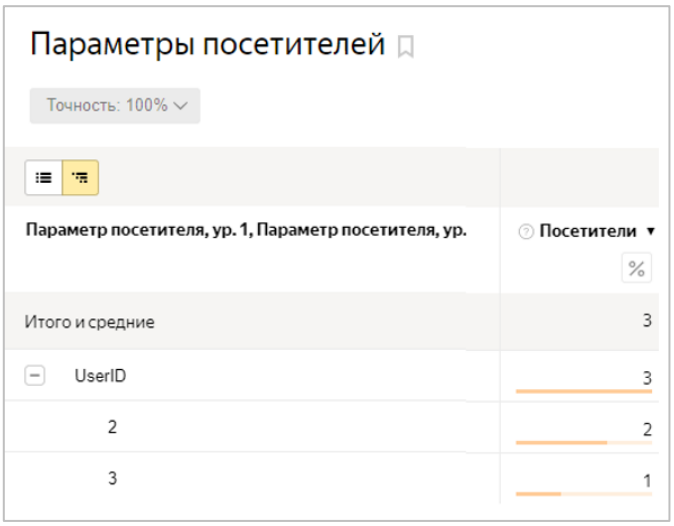


Рис. 791. Отчет Параметры посетителей

А также в карточке конкретного пользователя раздела **Посетители**:

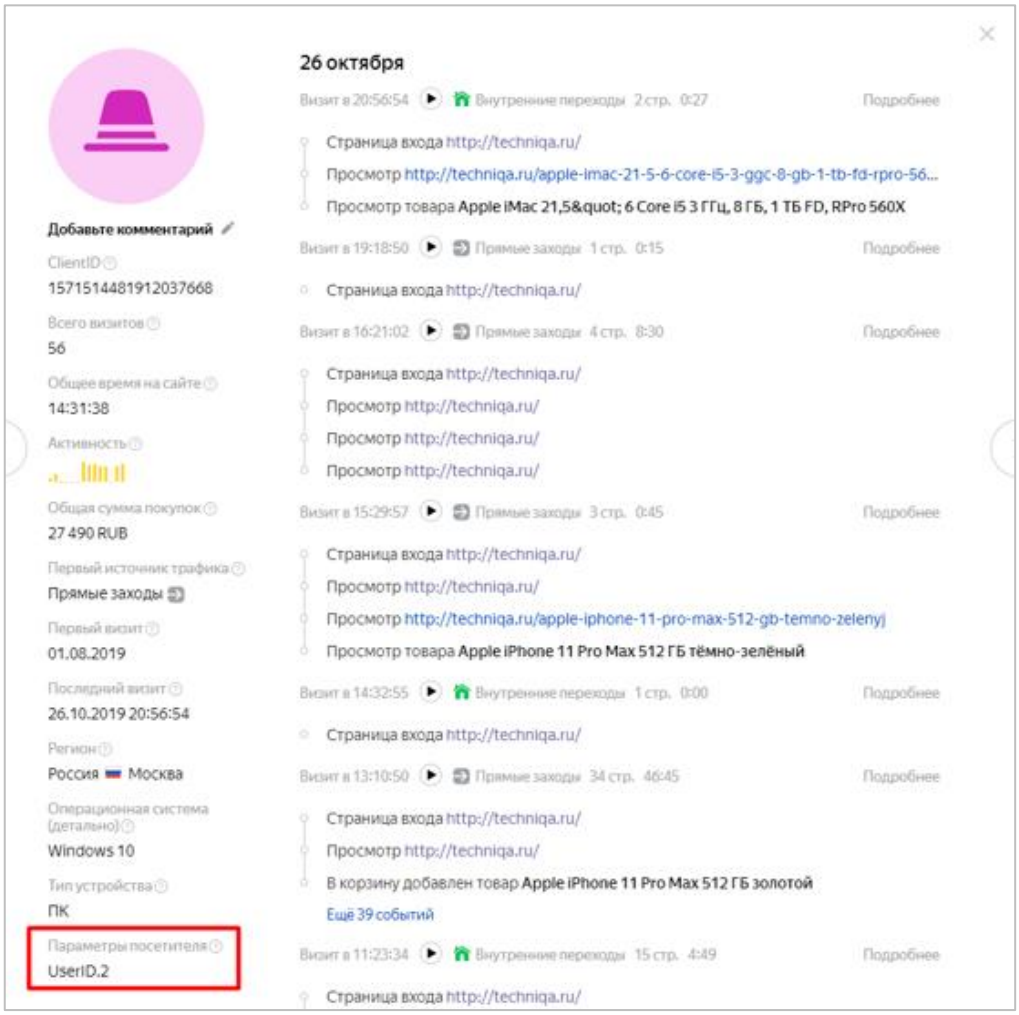


Рис. 792. Карточка пользователя в разделе Посетители

Таким образом, вы можете использовать несколько представлений (одно – User-ID, а другое – обычное) для отслеживания действий пользователей с привязкой к их идентификатору. А с созданным специальным параметром появляется возможность сопоставить обращения, покупки человека с его реальными данными, которые собираются в вашей CRM-системе.

Основной параметр: Источник или канал | Источник | Канал | Ключевое слово | Другое

Показать на диаграмме | Дополнительный параметр: User ID | Тип сортировки: По умолчанию | Ещё...

	Источник или канал	User ID	Источники трафика			Действия			Конверсии: Электронная торговля		
			Пользователи	Новые пользователи	Сеансы	Показатель отказов	Страниц/сеанс	Сред. длительность сеанса	Коэффициент транзакций	Транзакции	Доход
			68 % от общего количества: 2,28 % (2 976)	29 % от общего количества: 1,03 % (2 803)	269 % от общего количества: 7,48 % (3 597)	14,87 % Средний показатель для представления: 51,38 % (-71,06 %)	6,85 Средний показатель для представления: 2,13 (221,59 %)	00:11:19 Средний показатель для представления: 00:01:53 (499,93 %)	24,16 % Средний показатель для представления: 2,36 % (922,55 %)	65 % от общего количества: 76,47 % (85)	285 030,00 ₽ % от общего количества: 76,37 % (373 225,00 ₽)
1.	mail.ru / organic	357	1 (1,32 %)	0 (0,00 %)	34 (12,64 %)	5,88 %	7,74	00:16:58	29,41 %	10 (15,38 %)	38 272,00 ₽ (13,43 %)
2.	(direct) / (none)	6299	1 (1,32 %)	0 (0,00 %)	22 (8,18 %)	31,82 %	4,68	00:09:27	27,27 %	6 (9,23 %)	36 700,00 ₽ (12,88 %)
3.	(direct) / (none)	43050	1 (1,32 %)	0 (0,00 %)	2 (0,74 %)	50,00 %	5,50	00:18:45	150,00 %	3 (4,62 %)	9 150,00 ₽ (3,21 %)
4.	eLama-yandex / cpc	43059	1 (1,32 %)	1 (3,45 %)	1 (0,37 %)	0,00 %	29,00	00:32:13	200,00 %	2 (3,08 %)	7 050,00 ₽ (2,47 %)
5.	google / organic	42788	1 (1,32 %)	0 (0,00 %)	5 (1,86 %)	0,00 %	5,20	00:08:42	40,00 %	2 (3,08 %)	8 500,00 ₽ (2,98 %)
6.	yandex / organic	39253	1 (1,32 %)	0 (0,00 %)	1 (0,37 %)	0,00 %	37,00	00:54:03	200,00 %	2 (3,08 %)	13 100,00 ₽ (4,60 %)
7.	yandex / organic	43113	1 (1,32 %)	1 (3,45 %)	2 (0,74 %)	50,00 %	2,50	00:01:33	100,00 %	2 (3,08 %)	5 250,00 ₽ (1,84 %)
8.	(direct) / (none)	29989	1 (1,32 %)	1 (3,45 %)	1 (0,37 %)	0,00 %	26,00	00:19:55	100,00 %	1 (1,54 %)	5 600,00 ₽ (1,96 %)
9.	(direct) / (none)	42239	1 (1,32 %)	1 (3,45 %)	1 (0,37 %)	0,00 %	5,00	00:10:59	100,00 %	1 (1,54 %)	1 900,00 ₽ (0,67 %)
10.	(direct) / (none)	43032	1 (1,32 %)	0 (0,00 %)	2 (0,74 %)	50,00 %	3,00	00:09:39	50,00 %	1 (1,54 %)	3 800,00 ₽ (1,33 %)

Рис. 793. User ID в отчете Google Analytics как дополнительный параметр

Как? Элементарно! Вы можете передавать ID заявки или ID транзакции с помощью настроенной электронной торговли, а также User ID. Получается связка **User ID – ID заявки/транзакции**. А данные, которые пользователь отправил вместе с заявкой/транзакцией (это может быть телефон, e-mail, контактный адрес и другая информация), попадают к вам в CRM-систему. Получается еще одна связка **ID заявки/транзакции – Персональные данные пользователя**.

В результате, мы можем соединить две связки с помощью общего параметра **ID заявки/транзакции** и проанализировать действия пользователя на сайте с помощью функции *кросс-девайс отслеживания*. Создание таких централизованных систем – одна из основных задач при построении сквозной аналитики на базе Client ID / User ID. Но это уже совсем другая история!

Настройка динамического ремаркетинга (Google Ads)

Динамический ремаркетинг Google позволяет показывать вашей аудитории рекламу с учетом истории покупок, посещений сайта и демографических данных.

Рис. 794. Принцип работы динамического ремаркетинга

Различные платформы предоставляют его как один из видов рекламы. myTarget, Facebook, Google Ads, Яндекс – все они имеют схожий функционал.

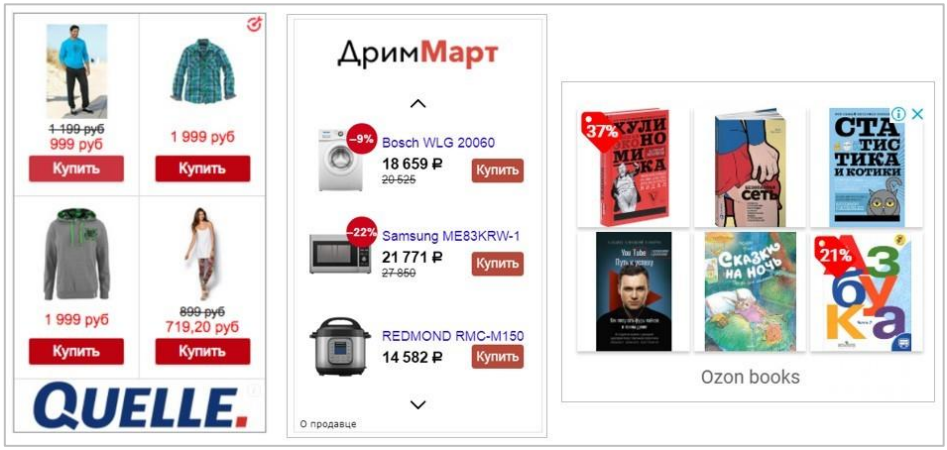


Рис. 795. Примеры объявлений динамического ремаркетинга различных рекламных систем

По официальным данным Google:

- 96% посетителей закрывают сайт, не выполнив конверсий;
- 70% посетителей уходят, не оплатив товары в корзине;
- 49% посетителей обычно просматривают от 2 до 4 сайтов прежде, чем совершить покупку;

Одного посещения пользователя вашего сайта недостаточно для совершения конверсии. Использование динамического ремаркетинга помогает увеличить количество конверсий на 200-300%, а также снизить цену за конверсию на 15-40% за счет персонализации предложения.

Настройка динамического ремаркетинга тесно связана с таким понятием, как **фид данных, фид товаров**. **Фид (feed)** – это файл, в котором содержится подробная информация о каждом товаре в интернет-магазине (название, стоимость, скидка, наличие, id и другие атрибуты).

Преимущества использования фидов:

- всегда актуальная информация о наличии и стоимости товара;
- экономия времени;
- использование изображений на поиске.

Основные формата фида – **xml, txt** (значения в столбцах разделены табуляцией) и **yml** (для РФ – формат Яндекс.Маркета). О поддерживаемых форматах файла читайте в официальной справке Google (см. приложение).

Фид, как правило, расположен по уникальному URL-адрес на вашем сайте, ссылку на который вы можете добавить как один из источников данных в Google Merchant Center для показа объявлений в Google Рекламе:

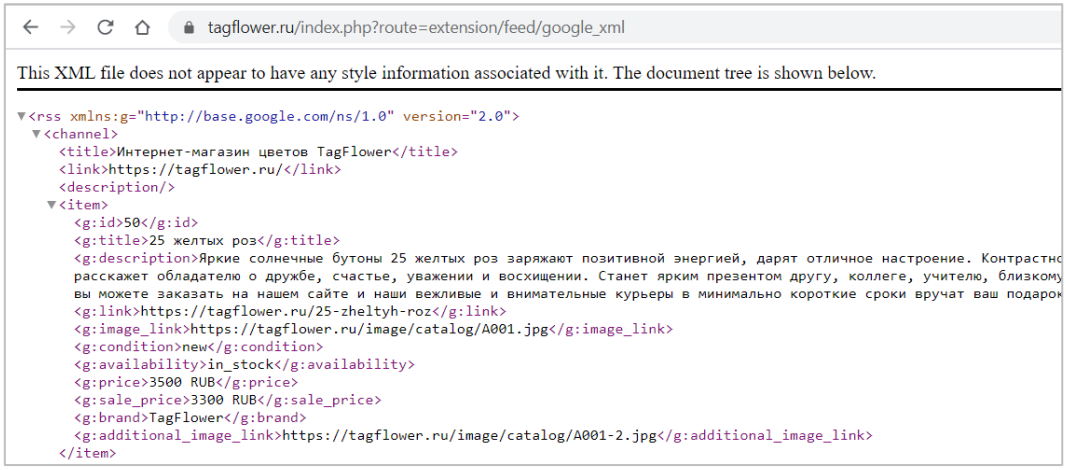


Рис. 796. Пример фида

Существует несколько способов настроить фид:

- настройка через коммерческие данные, таблицы (фиды), которые загружаются в Google Рекламу;
- при помощи Google Merchant Center (только для розничной торговли).

Также Google позволяет создавать автоматические фиды на основе данных с сайта, которые собираются с помощью сканирования. Ими можно управлять и пользоваться в разных программах Merchant Center. Принцип работы автоматических фидов подробно разобран в документации (см. приложение).

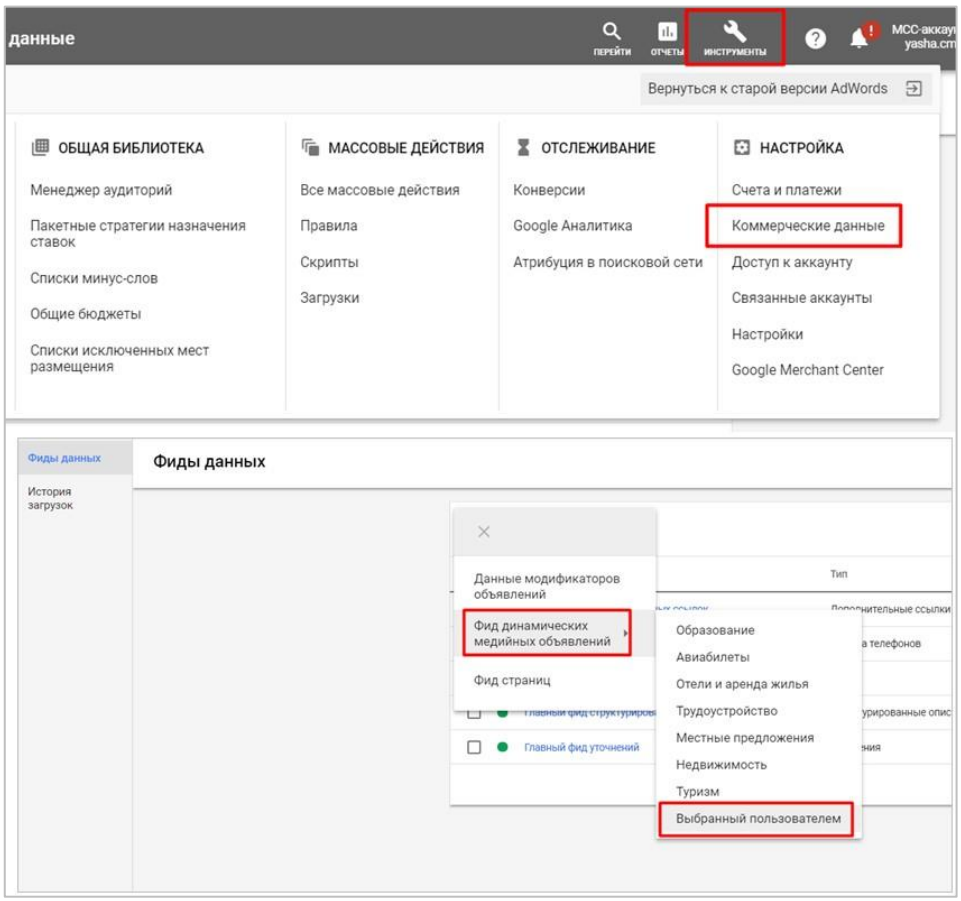


Рис. 797. Коммерческие данные

Загрузите фид с информацией о товарах и услугах, которые вы хотите рекламировать (например, добавьте описания и цены). Чтобы отформатировать файл, используйте шаблон специального фида (формат файла .csv).

В случае с Google Merchant Center вы можете выбрать программу **Динамический ремаркетинг**, а далее просто указать ссылку, по которой будет происходить синхронизация товаров и их обновление в определенное время.

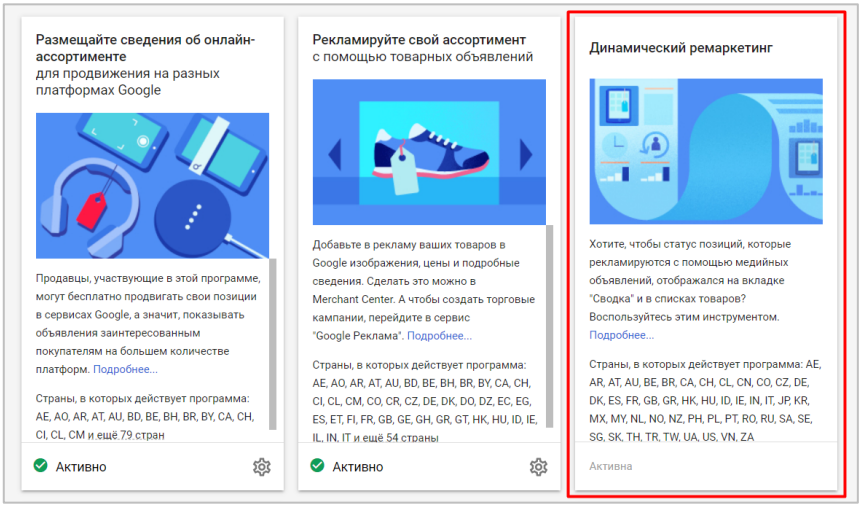


Рис. 798. Программа динамического ремаркетинга в Google Merchant Center

Процесс связывания аккаунтов Google Ads и Merchant Center я целенаправленно опускаю в этом руководстве, так как он имеет к Google Tag Manager второстепенное отношение. Аналогично и с созданием фида товаров, поскольку там есть целый ряд тонкостей и нюансов, которые выходят за рамки данной книги.

После создания фида данных у вас есть несколько вариантов настройки динамического ремаркетинга:

1. с помощью Google Analytics и сооветствующей разметки на сайте;
2. с помощью Google Tag Manager, но без уровня данных;
3. с помощью уровня данных, Google Tag Manager и тега ремаркетинга Google Рекламы.

Первый способ детально разобран в моем блоге (см. приложение). Рассмотрим процесс настройки динамического ремаркетинга для розничной торговли с помощью диспетчера тегов Google и уровня данных (вариант №3).

Описанный ниже алгоритм основан на документации Google и их рекомендациях. Материалы для справки, которые взяты за основу: **Динамический ремаркетинг в Google Рекламе** и **Как настроить события и параметры ремаркетинга с учетом рода деятельности вашей компании** (см. приложение).

После связывания аккаунтов Google Ads и Merchant Center необходимо произвести следующие настройки:

- создать тег Google Рекламы;
- настроить события и параметры ремаркетинга на сайте;
- создать переменные в Google Tag Manager;
- создать триггер активации;
- создать тег ремаркетинга в Google Рекламе;
- создать аудитории в интерфейсе Google Ads.

Создание тега Google Рекламы

В аккаунте Google Рекламы перейдите в **Инструменты и Настройки – Менеджер аудиторий – Источники аудиторий** и настройте **Тег Google Рекламы**:

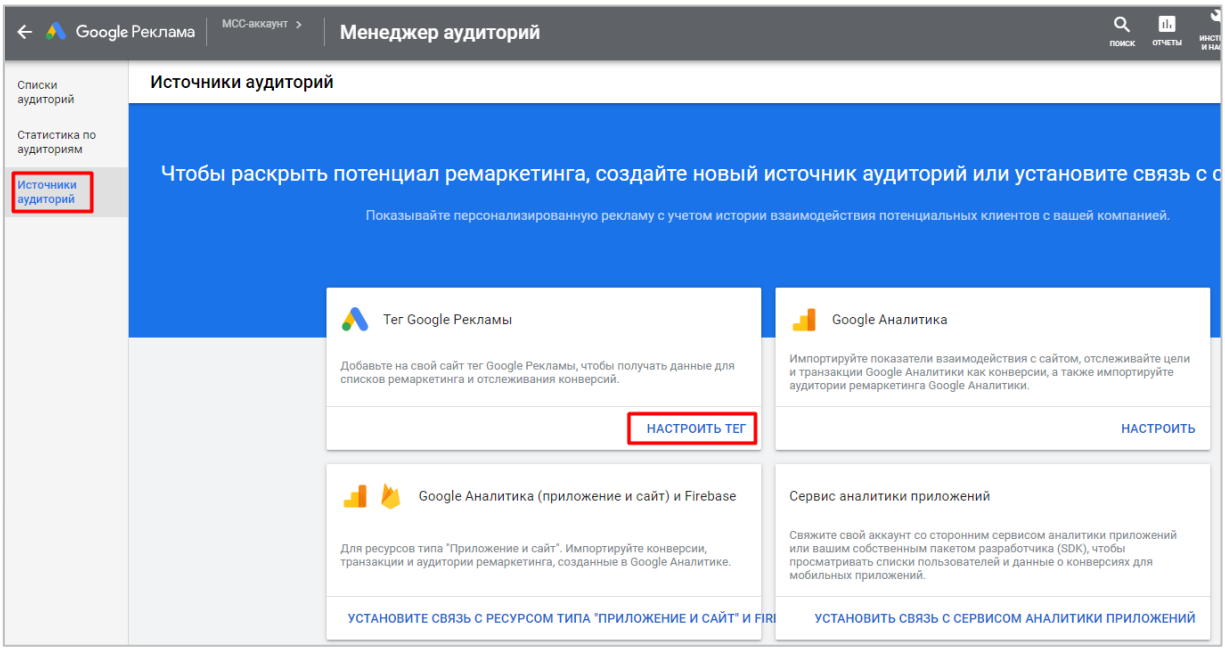


Рис. 799. Создание тега Google Рекламы

Поскольку мы создаем тег для динамического ремаркетинга, то в разделе **Ремаркетинг** укажите тип данных – **Собирать данные об определенных действиях, которые пользователи совершают на вашем сайте, чтобы показывать персонализированную рекламу**, в блоке **Вид деятельности** поставьте галочку напротив **Розничная торговля** (напоминаю, что мы разбираем пример настройки для интернет-магазина!).

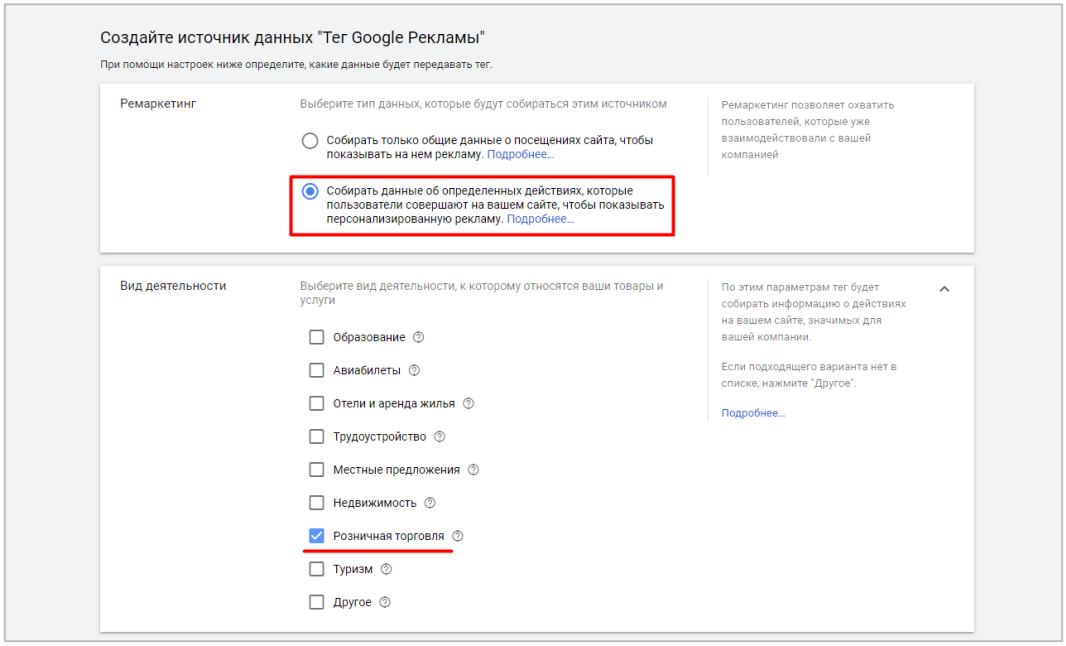


Рис. 800. Настройка тега Google Рекламы

Нажав продолжить, вам предложат несколько способов добавления тега Google Рекламы. Выберите **Использовать Google Менеджер тегов**.

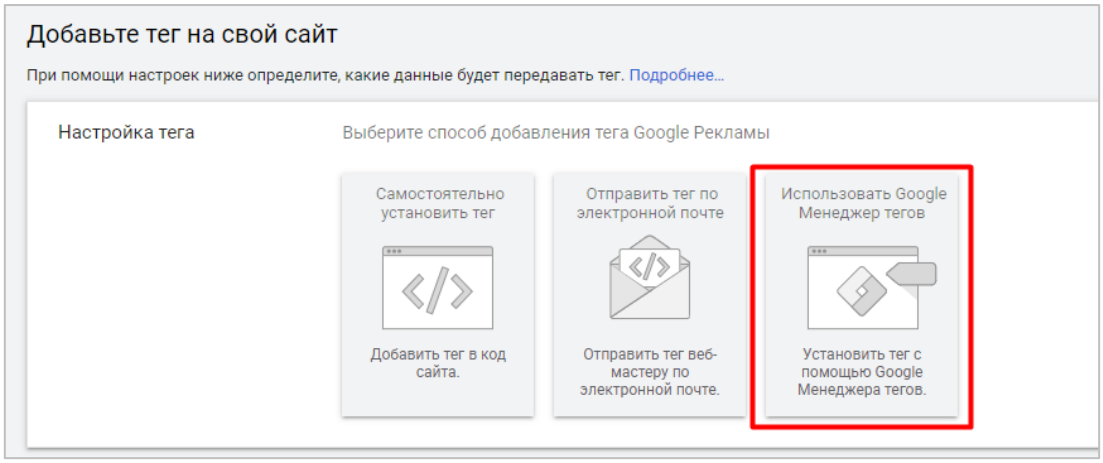


Рис. 801. Добавление тега с помощью Google Менеджера тегов

Сохраните идентификатор конверсии (**Conversion ID**). Он нам понадобится при создании тега ремаркетинга в диспетчере тегов.

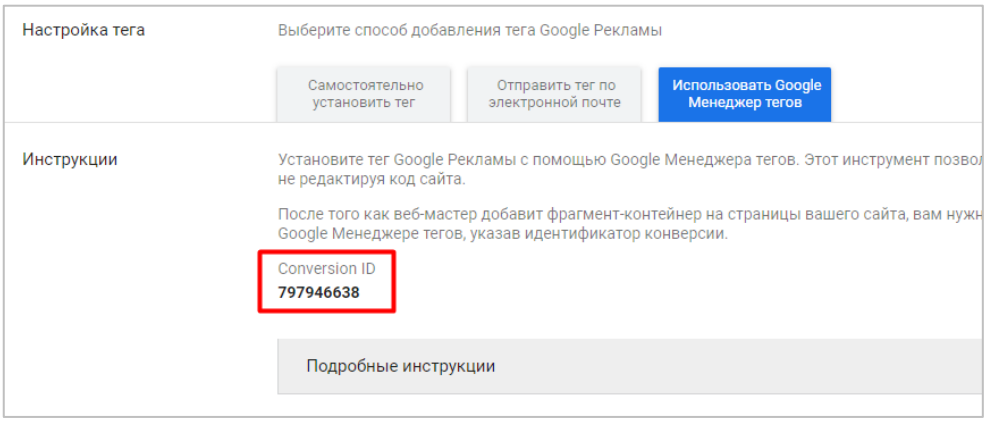


Рис. 802. Conversion ID

Настройка события и параметров ремаркетинга на сайте

Теперь настало время создать техническое задание (ТЗ) и поручить разработчику настроить события, параметры ремаркетинга, а также сформировать уровень данных, который должен соответствовать последнему обновлению Google (сентябрь 2019).

События ремаркетинга – это действия пользователей на вашем сайте, которые нужно учитывать при показе персонализированных объявлениях.

События состоят из:

- название события;
- параметры события;

```
<script>
gtag('event','view_item', {
  'value': 998.55,
  'items': [
    {
      'id': 1234,
      'google_business_vertical': 'retail'
    },
    {
      'id': 45678,
      'google_business_vertical': 'retail'
    }
  ]
});
</script>
```

Рис. 803. Пример кода события view_item

Название события – это строка, описывающая событие. С ее помощью Google Реклама определяет, в какой из автоматически созданных списков добавить пользователя.

Для розничной торговли доступны следующие названия событий:

- **view_search_results** - пользователь посетил страницу результатов поиска;
- **view_item_list** - пользователь посетил страницу категории;
- **view_item** - пользователь посетил страницу товара;
- **add_to_cart** - пользователь добавил товар в корзину;
- **purchase** - пользователь совершил покупку.

Название события	Описание
view_search_results	Это событие означает, что пользователь посетил страницу результатов поиска.
view_item_list	Это событие означает, что пользователь посетил страницу категории.
view_item	Это событие означает, что пользователь посетил страницу товара.
add_to_cart	Это событие означает, что пользователь добавил товар в корзину.
purchase	Это событие означает, что пользователь совершил покупку.

Рис. 804. Названия событий

Параметры события – это объект JavaScript, содержащий данные об отслеживаемом событии. Он обязательно должен иметь параметр **items**, содержащий один или несколько объектов **item**. В этих объектах хранятся сведения о товарах или услугах, в отношении которых пользователь совершил определенное действие.

Параметры, которые необходимо передавать в каждом теге событий:

- **items** (обязательный параметр);
- **id** – ID товара, который соответствует идентификатору в фиде данных;
- **value** – ценность (цена) товара;

- **google_business_vertical** – retail (тип фида).

Имя параметра	Описание	Обязательно ли указывать
id	Уникальный идентификатор товара. Значение этого параметра должно соответствовать одному из следующих атрибутов фида товаров в аккаунте Google Merchant Center: id, item_group_id, display_ads_id.	Обязательно
google_business_vertical	Определяет тип фида, в котором будет выполняться поиск данных. Этот параметр должен иметь значение retail.	Рекомендуется

Рис. 805. Параметры событий

Параметр **google_business_vertical** является рекомендованным, но необязательным параметром. Основную функцию несет уникальный идентификатор товара, который должен соответствовать любому из следующих трех атрибутов в фиде Google Merchant Center: **id, item_group_id** или **display_ads_id**. Он позволяет показывать пользователям рекламу именно тех товаров, сведения о которых они просматривали.

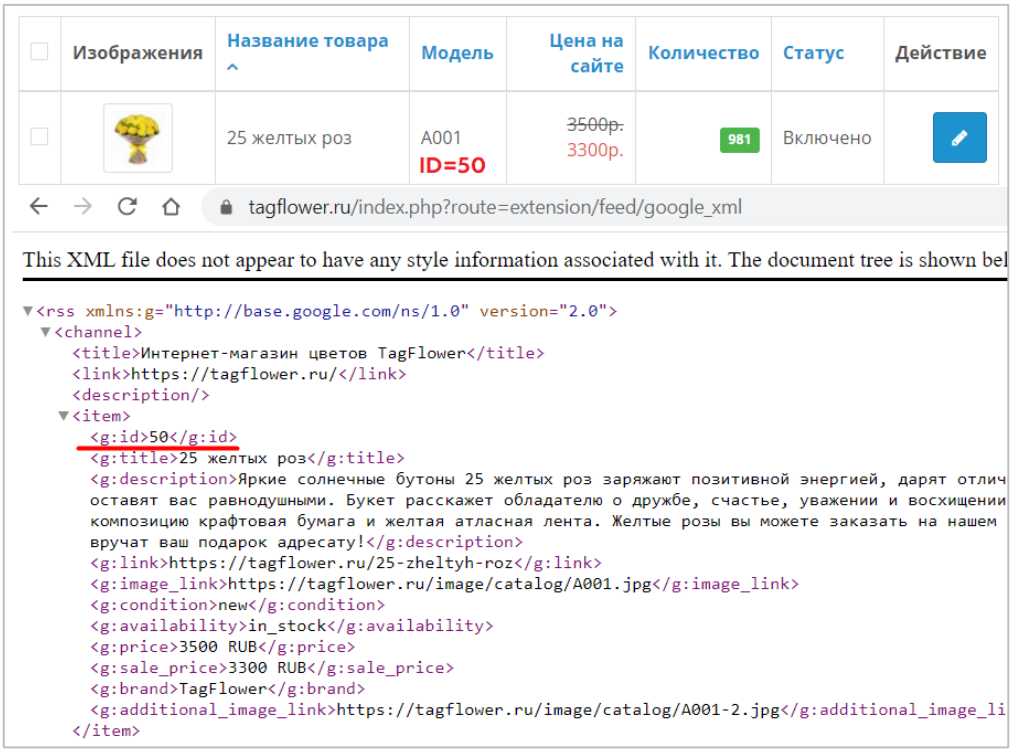


Рис. 806. Пример товара в фиде данных и в базе интернет-магазина, который имеет уникальный id

Когда пользователь переходит на сайт, срабатывают события **view_search_results, view_item_list, view_item, add_to_cart** и **purchase**, внутри которых передаются параметры события **id, item_group_id** или **display_ads_id**, а вместе с ними записывается информация по товарам, привязанная к конкретному пользователю. Таким образом, когда рекламодатель создаст в Google Ads рекламную кампанию на динамический ремаркетинг, пользователю начнет показываться персонализированная реклама в зависимости от просмотренных товаров и совершенных событий на сайте.

Именно эти события с обязательными параметрами необходимо добавить в код вашего сайта на соответствующие страницы. Пример технического задания вы можете скачать по ссылке (см. приложение).

Например, для страницы успешно отправленного заказа (событие purchase) ТЗ для разработчика будет выглядеть так: *на странице успешного оформления заказа просьба сформировать dataLayer и передать событие purchase, которое срабатывает, когда пользователь совершил покупку и в котором будут следующие данные:*

```
window.dataLayer = window.dataLayer || [];
dataLayer.push({
'event': 'purchase',
'value': {{Цена товара, переменная – число}},
'items': [
{
'id': {{ID товара}},
'google_business_vertical': 'retail'
},
]
});
```

Аналогично для всех других событий **view_search_results, view_item_list, view_item** и **add_to_cart.** Схематично всю работу можно представить так:

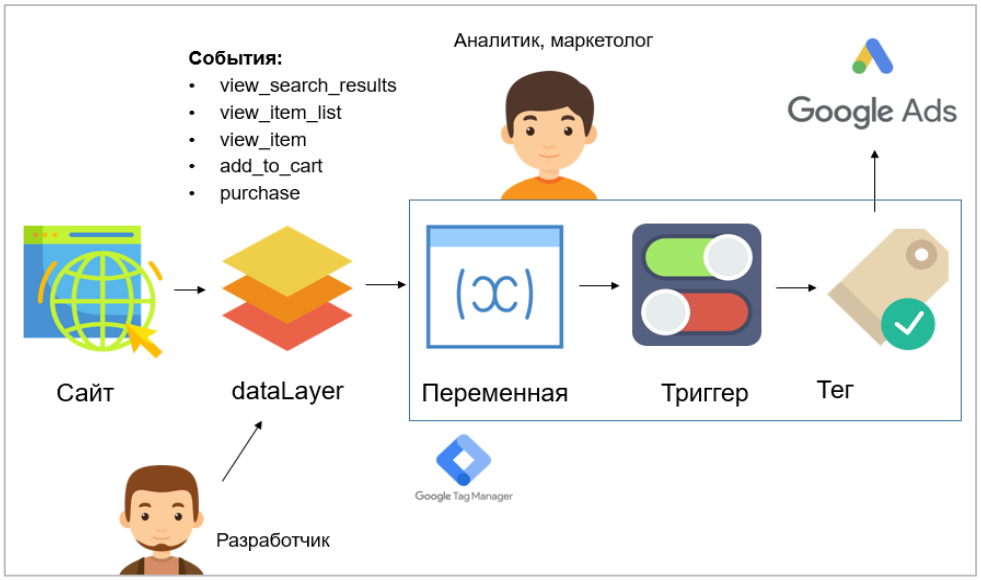


Рис. 807. Командная работа

Все точно также, как и с настройкой User ID и электронной торговлей. Разработчик на сайте формирует уровень данных на события динамического ремаркетинга, которые вы отобразили в техническом задании. Затем в Google Tag Manager вы создаете необходимые сущности (переменные, триггеры и теги) для передачи данных о совершенных событиях в Google Рекламу. И когда ваша работа, и работа программиста будет завершена, вы проверяете корректность всех настроек. Если что-то не работает, совместными усилиями ищете причину и устраняете ошибки.

Вот как выглядит сформированный dataLayer для интернет-магазина на OpenCart, который разработчик передает в зависимости от события (универсальный):

```
if ($data['google_page']) {
    $data['google_output'] .= '<script type="text/javascript">'."\n";
    $data['google_output'] .=   'dataLayer = window.dataLayer || [];'."\n";
    $data['google_output'] .=   'dataLayer.push({ '."\n";
    $data['google_output'] .= 'event: "' . $data['google_page'] .'",'."\n";
    $data['google_output'] .= "'value': ". $data['totalvalue'] . ",\n";
if(isset($data['google_ids']) && count($data['google_ids']) > 0){
    $data['google_output'] .= "'items': [\n";
foreach($data['google_ids'] as $item) {
    $data['google_output'] .= "{\n";
    $data['google_output'] .= "'id': " . $item . ",\n";
    $data['google_output'] .= "'google_business_vertical': 'retail'\n";
    $data['google_output'] .= "},\n";
}
    $data['google_output'] .= "]\n";
}
    $data['google_output'] .= '});'."\n</script>\n";
}
}
```

Рис. 808. Пример кода для формирования уровня данных (интернет-магазин на OpenCart)

Само событие **event**, помещенное в переменную **$data['google_page']** подставляется в зависимости от страницы сайта, на которой оно совершается. И там не менее сложный код, чем представленный выше. Для другого проекта и CMS-движка сайта код выделенных переменных будет другим. И реализация может быть иной.

Как и в случае с электронной торговлей и функцией User ID мы можем использовать Google Tag Manager и DOM Scraping (извлечение данных со страницы, например, с помощью переменной **Элемент DOM**). И снова я не рекомендую так делать, поскольку любое изменение CSS-селекторов элементов со стороны программиста приведет к сбою в отслеживании. Лучше доверить эту задачу разработчику, и тогда вы обезопасите себя от критических ошибок.

Чтобы проверить сформированные уровни данных, перейдите в режим отладки Google Tag Manager. Последовательно проверьте все события - **view_search_results, view_item_list, view_item, add_to_cart** и **purchase**. Например, на странице успешной оформленного заказа для события purchase уровень данных для интернет-магазина (розничная торговля) будет выглядеть так:

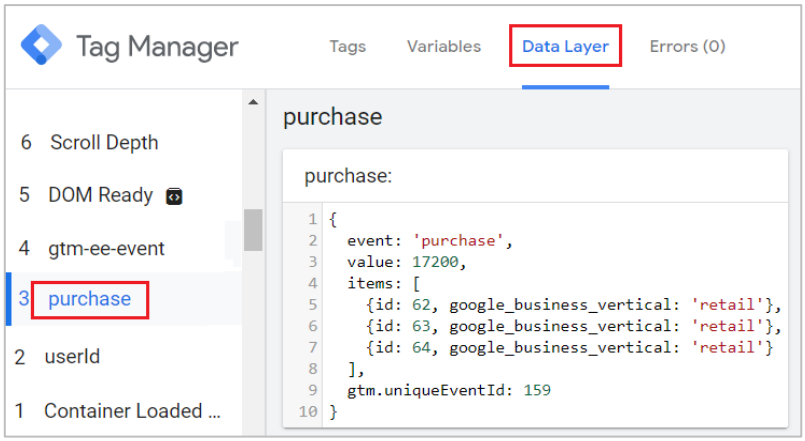


Рис. 809. Уровень данных события purchase для динамического ремаркетинга

, где:

- **value** – общая сумма заказа;
- **id** – идентификаторы товаров (62, 63 и 64);
- **google_business_vertical** – тип фида (retail – для розничной торговли).

После того, как вы проверили настроенные события, переходим к настройке переменных, триггеров и тегов.

Создание переменных

В Google Tag Manager создайте три переменных типа **Переменная уровня данных**:

- items;
- value;
- items.0.google_business_vertical.

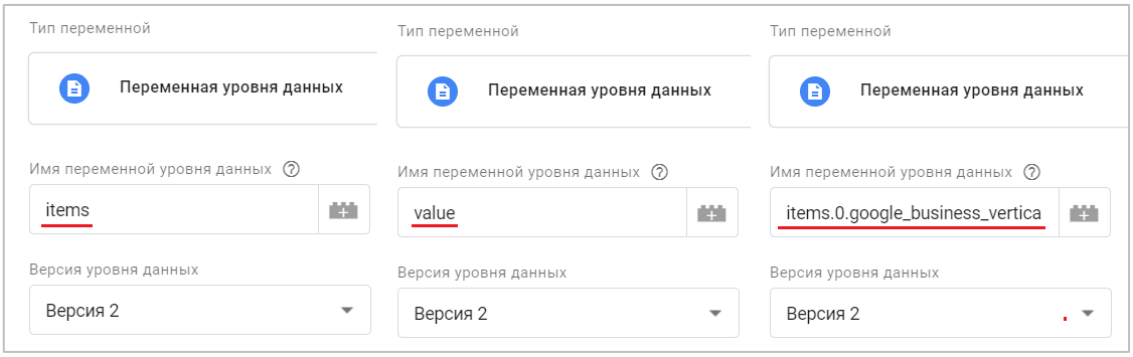


Рис. 810. Три переменных уровня данных

Последняя переменная **items.0.google_business_vertical** имеет индекс 0 и содержит точечную нотацию (см. главу, посвященную переменным), поскольку при активации события формируется массив данных, в котором может быть не один товар, который пользователь просматривал, добавил в корзину или купил. И чтобы в переменной было извлечено корректное значение (везде одинаковое – **retail**), необходимо использовать индекс. Самое простое и универсальное – это 0 индекс.

Чтобы упростить составление переменных, используйте расширение для браузера **Datalayer Checker**:

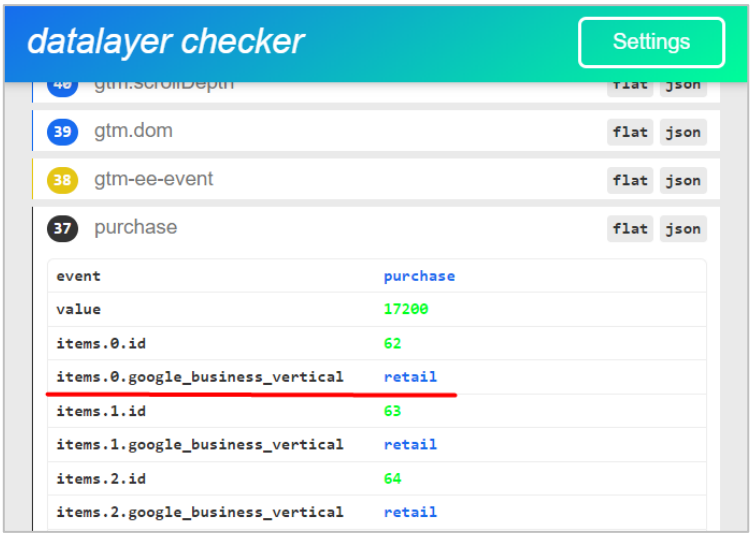


Рис. 811. Расширение Datalayer Checker

Затем активируйте встроенные переменную **Event** из раздела **Утилиты**:

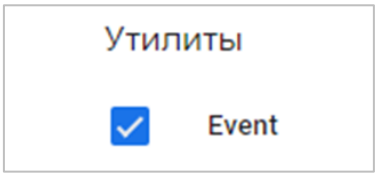


Рис. 812. Переменная Event

Создание триггера

Создайте триггер типа **Пользовательское событие** с регулярным выражением и перечислите через символ **| (оператор ИЛИ)** все ваши события:

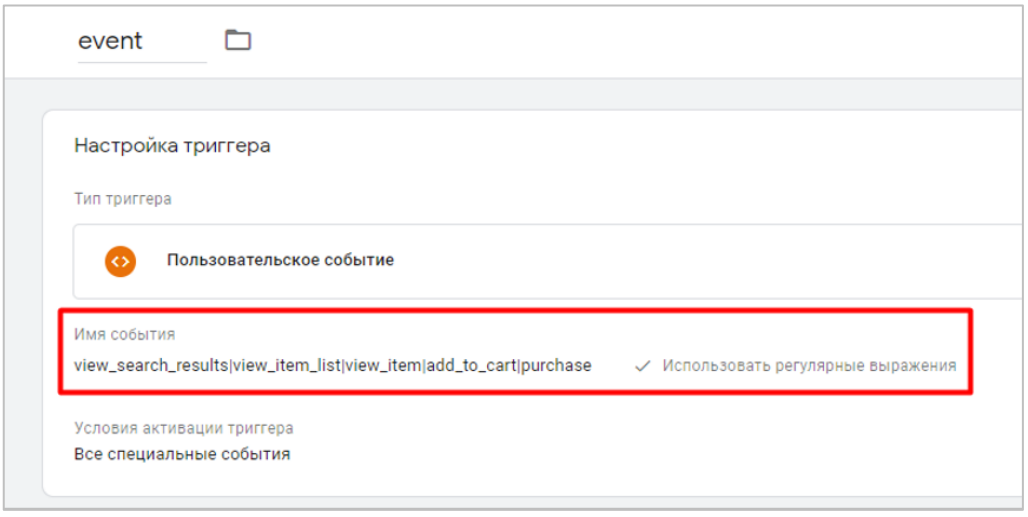


Рис. 813. Триггер активации с использованием регулярного выражения

Создание тега ремаркетинга в Google Рекламе

Добавьте тег **Ремаркетинг в Google Рекламе** со следующими настройками:

- **Идентификатор конверсии** – Conversion ID, который вы получили при создании тега в интерфейсе Google Рекламы;
- **Галочка** - Send dynamic remarketing event data;
- **Event Name** – наша встроенная переменная (Event);
- **Event Value** – наша переменная уровня данных (value);
- **События** – наша переменная уровнях данных (items);
- **Пользовательские параметры** – Нет;
- **Триггер активации** – пользовательское событие, созданное на предыдущему шаге.

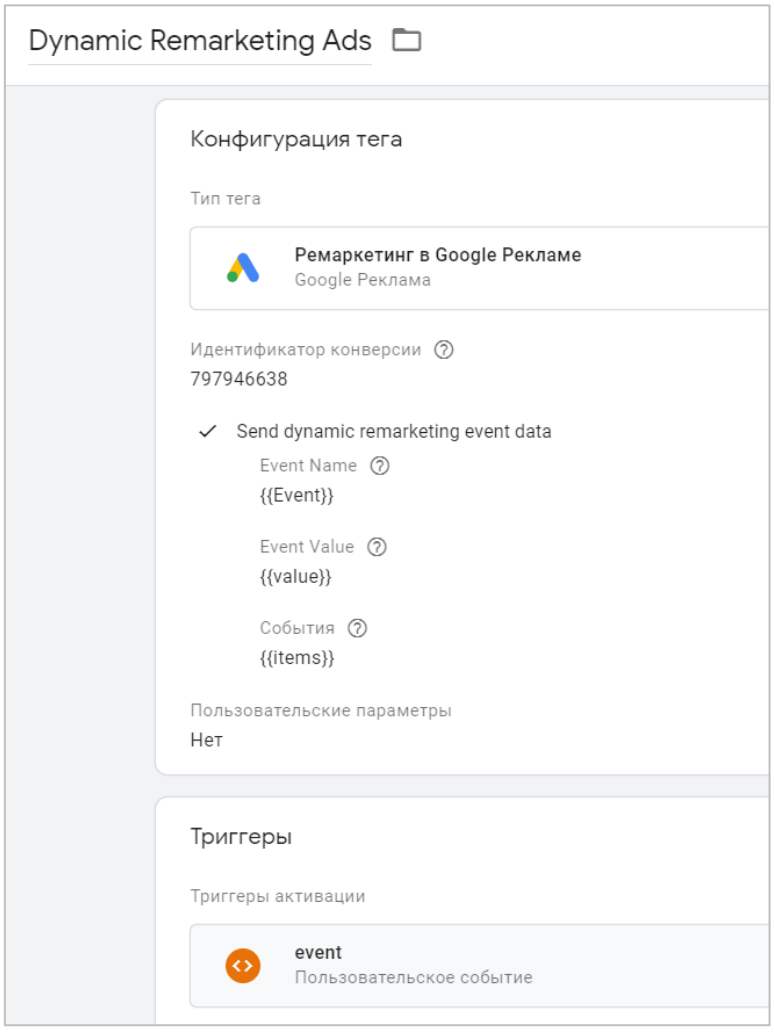


Рис. 814. Настройки тега Ремаркетинг в Google Рекламе

Сохраните тег. Окончательно проверить все настройки можно с помощью режима предварительного просмотра. Совершите поочередно все события из вашего списка и посмотрите, срабатывает ли тег ремаркетинга Google Рекламы и передает ли он данные в Google Ads.

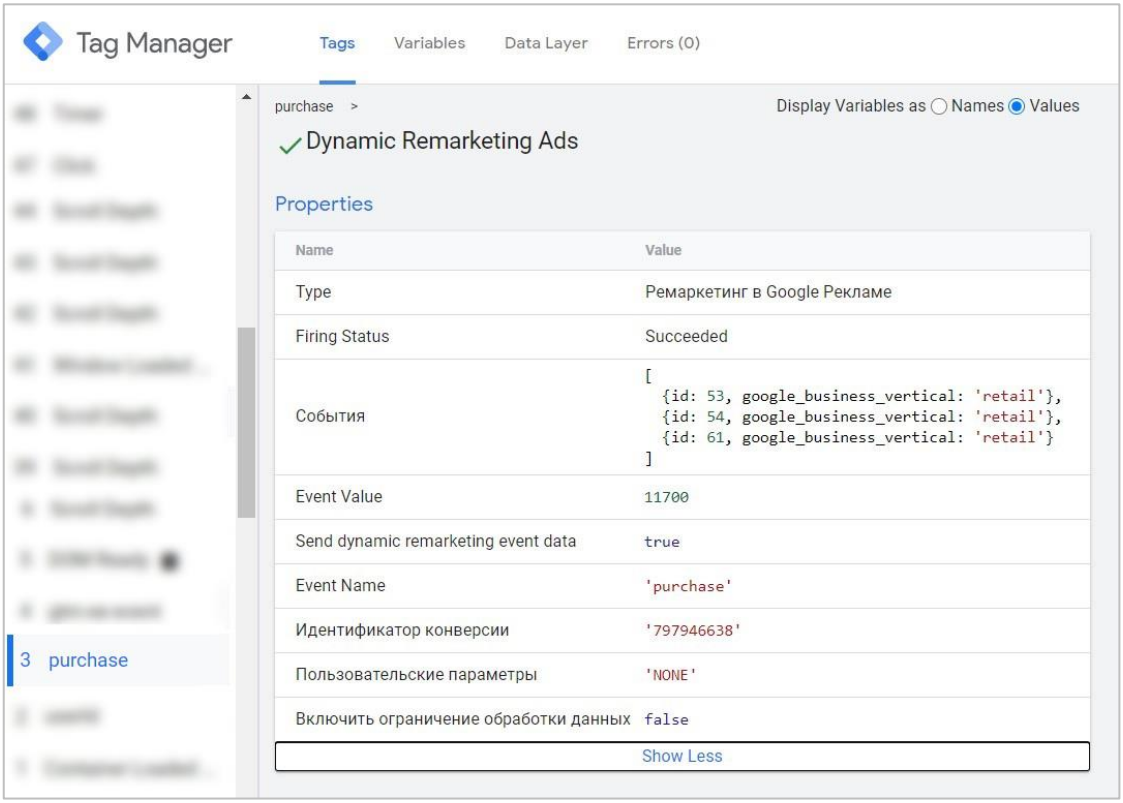


Рис. 815. Проверка работы тега в режиме отладки

На рисунке выше при срабатывании события **purchase** активируется тег **Ремаркетинг в Google Рекламе** с идентификатором конверсии, который передают в Google Ads параметры события: идентификаторы товаров, их ценность и тип фида.

Создание аудиторий в интерфейсе Google Ads

На завершающем шаге настройки перейдите в своем аккаунте Google Рекламы в **Инструменты и Настройки – Менеджер аудиторий – Источники аудиторий.** Через некоторое время в созданный тег Google Рекламы начнут поступать данные о различных событиях. Чтобы посмотреть, что Google уже собрал, нажмите на **Подробности**:

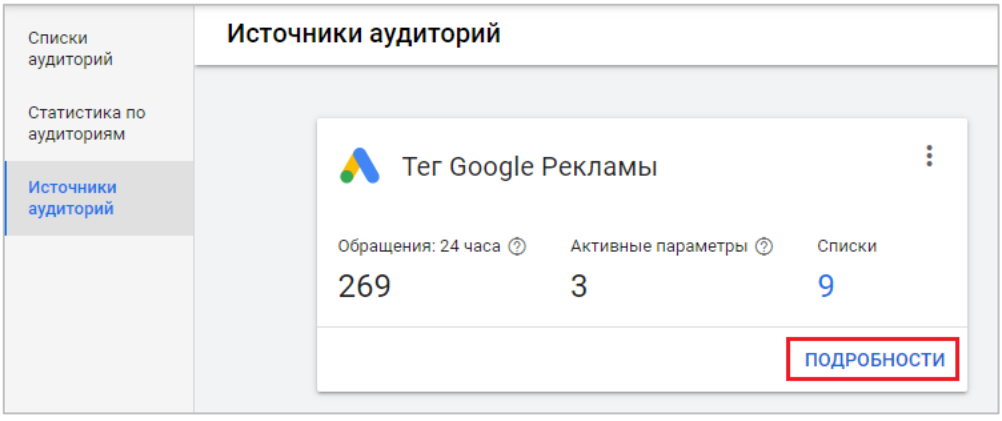


Рис. 816. Данные в теге Google Рекламы

Внутри тега будет отображена статистика по всем обращениям, включая все параметры, которые необходимы для показа персонализированной рекламы посетителям вашего сайта, включая **event, google_business_vertical, id** и **value**:

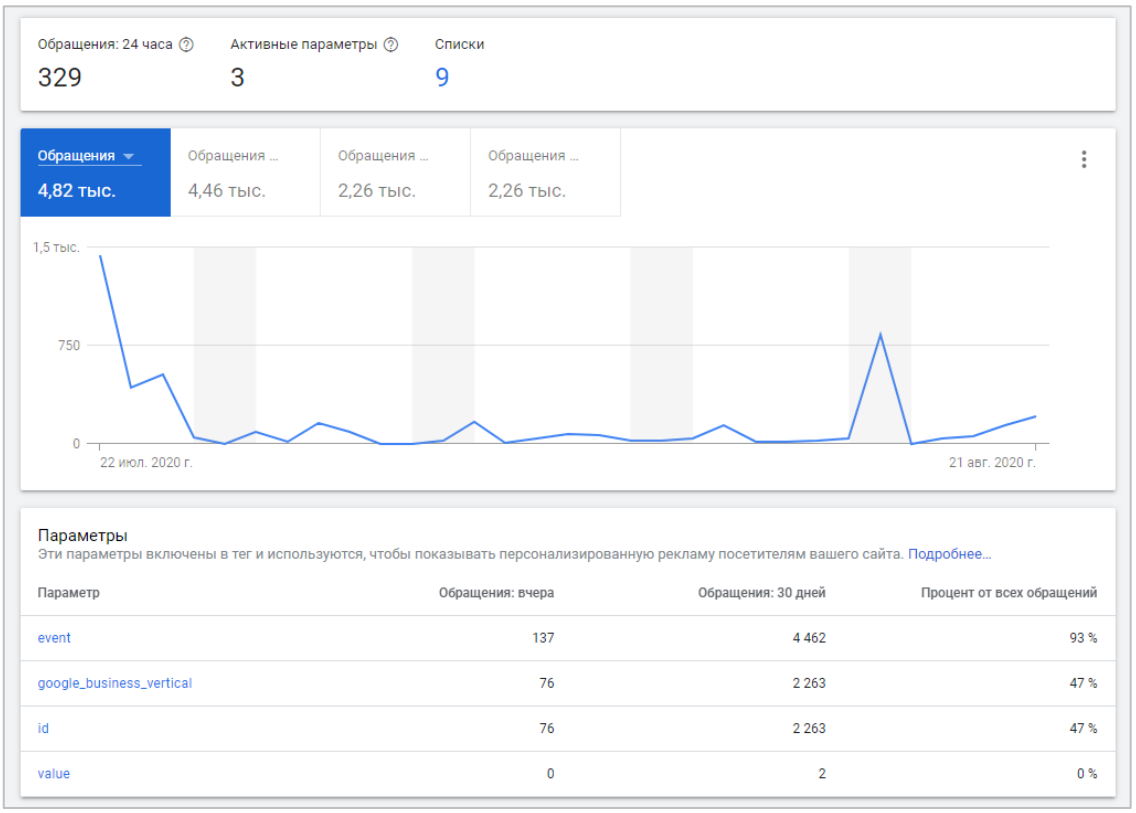


Рис. 817. Данные по настроенным событиям динамического ремаркетинга

Примечание: данные по событиям в тег Google Рекламы могут попадать с запозданием. Иногда задержка составляет 5-7 дней. Все зависит от количества посетителей сайта и количестве событий, которые они совершают во время своих сеансов.

Чтобы создать аудиторию, перейдите в меню **Списки аудиторий – + – Посетители сайта**. В условии посещенной страницы вы можете выбрать не только URL, но и указать событие,

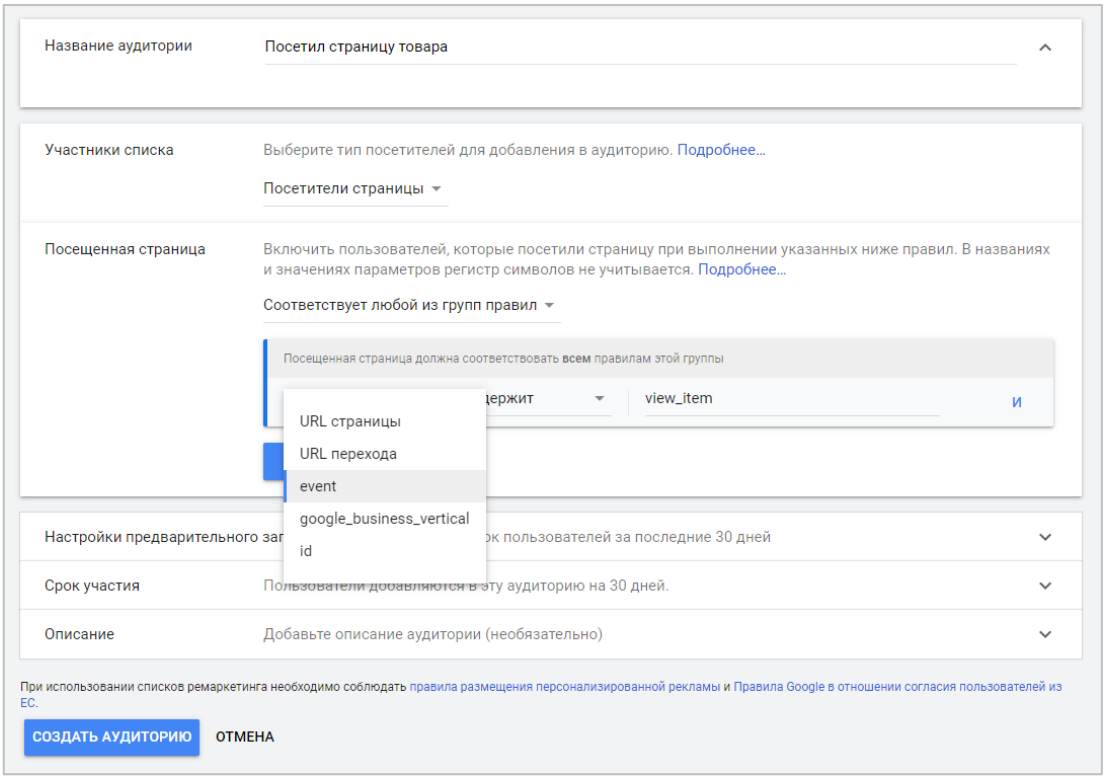
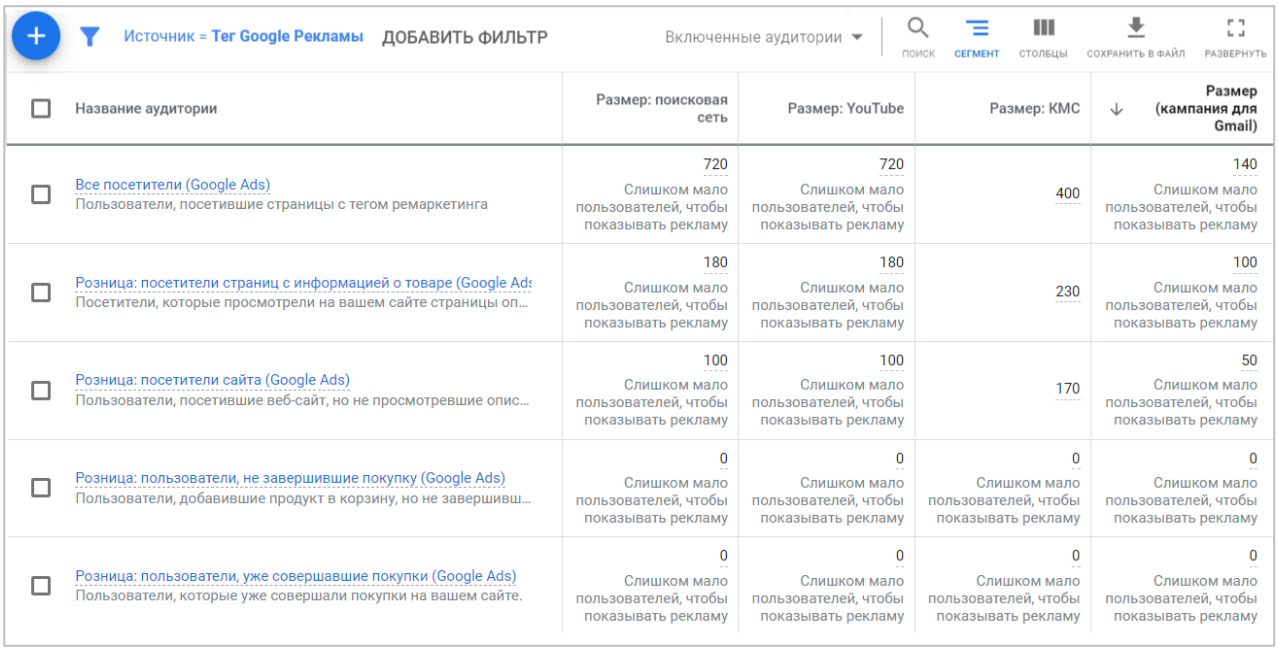


Рис. 818. Создание аудиторий в Google Рекламе

Также некоторые списки аудиторий в Google Ads создаются и пополняются автоматически. Например:

- **Все посетители (Google Ads)** - пользователи, посетившие страницы с тегом ремаркетинга;
- **Розница: посетители страниц с информацией о товаре (Google Ads)** - посетители, которые просмотрели на вашем сайте страницы определенных товаров, но ничего не добавили в корзину;
- **Розница: посетители сайта (Google Ads)** - пользователи, посетившие веб-сайт, но не просмотревшие описание какого-либо конкретного продукта;
- **Розница: пользователи, не завершившие покупку (Google Ads)** - пользователи, добавившие продукт в корзину, но не завершившие процесс покупки;
- **Розница: пользователи, уже совершавшие покупки (Google Ads)** - пользователи, которые уже совершали покупки на вашем сайте.



Название аудитории	Размер: поисковая сеть	Размер: YouTube	Размер: КМС	Размер (кампания для Gmail)
Все посетители (Google Ads) Пользователи, посетившие страницы с тегом ремаркетинга	720 Слишком мало пользователей, чтобы показывать рекламу	720 Слишком мало пользователей, чтобы показывать рекламу	400 Слишком мало пользователей, чтобы показывать рекламу	140 Слишком мало пользователей, чтобы показывать рекламу
Розница: посетители страниц с информацией о товаре (Google Ads Посетители, которые просмотрели на вашем сайте страницы оп...	180 Слишком мало пользователей, чтобы показывать рекламу	180 Слишком мало пользователей, чтобы показывать рекламу	230 Слишком мало пользователей, чтобы показывать рекламу	100 Слишком мало пользователей, чтобы показывать рекламу
Розница: посетители сайта (Google Ads) Пользователи, посетившие веб-сайт, но не просмотревшие опис...	100 Слишком мало пользователей, чтобы показывать рекламу	100 Слишком мало пользователей, чтобы показывать рекламу	170 Слишком мало пользователей, чтобы показывать рекламу	50 Слишком мало пользователей, чтобы показывать рекламу
Розница: пользователи, не завершившие покупку (Google Ads) Пользователи, добавившие продукт в корзину, но не завершивш...	0 Слишком мало пользователей, чтобы показывать рекламу	0 Слишком мало пользователей, чтобы показывать рекламу	0 Слишком мало пользователей, чтобы показывать рекламу	0 Слишком мало пользователей, чтобы показывать рекламу
Розница: пользователи, уже совершавшие покупки (Google Ads) Пользователи, которые уже совершали покупки на вашем сайте.	0 Слишком мало пользователей, чтобы показывать рекламу	0 Слишком мало пользователей, чтобы показывать рекламу	0 Слишком мало пользователей, чтобы показывать рекламу	0 Слишком мало пользователей, чтобы показывать рекламу

Рис. 819. Автоматически созданные аудитории

Примечание: ориентировочный размер списка вычисляется на основе данных за последние 30 дней. Точность оценки зависит от многих факторов, в том числе от настроек списка и от времени с момента добавления тега.

Глава 11

«Продвинутая» работа с Google Tag Manager

Прослушивание пользовательских событий

В JavaScript есть такое понятие, как «событие». Событие – это сигнал о том, что что-то произошло в браузере. Браузерных событий большое количество. Самые распространенные в интернет-маркетинге – это **click** (когда кликнули на какой-то элемент левой кнопкой мыши), **submit** (когда пользователь отправил форму *<form>*). Данные по ним мы отправляем в Яндекс.Метрику и Google Analytics, а также настраиваем цели-события.

В Google Tag Manager есть соответствующие триггеры для их отслеживания. Один называется **Клик – все элементы (gtm.click)**, а второй - **Отправка формы (gtm.formSubmit)**. Со временем появились и другие:

- Клики – Только ссылки (gtm.linkClick);
- Видео YouTube (gtm.video);
- Глубина прокрутки (gtm.scrollDepth);
- Доступность элемента (gtm.elementVisibility);
- Изменение в истории (gtm.historyChange);
- Таймер (gtm.timer);
- и другие.

Вспоминаем, что по умолчанию Google Tag Manager фиксирует три события.

1. событие **Container Loaded (gtm.js)** происходит сразу же после загрузки страницы;
2. событие **DOM Ready (gtm.dom)** происходит, когда модель DOM готова к обработке;
3. событие **Window Load (gtm.load)** происходит после загрузки всего первоначального контента страницы.

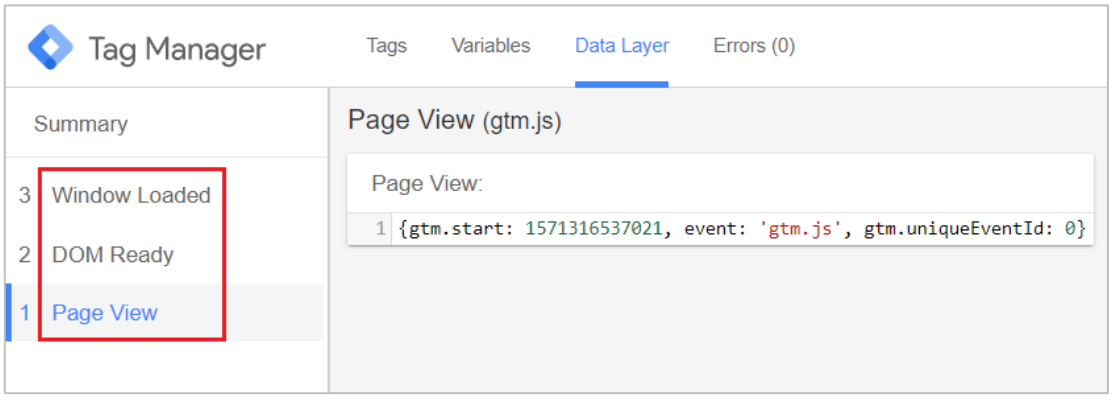


Рис. 820. Режим предварительного просмотра

Однако браузерных событий гораздо больше, чем заложено в GTM. Не всегда спасает и **Переменная автоматического события (Auto-Event)**, которая, по сути, совпадает со встроенными переменными.

События в JavaScript разделяются на системные: загрузка (**load**) и выгрузка (**unload**) страницы, события мыши (**click, mouseover, mousemove** и т.д.) и события клавиатуры (**keypress, keydown, keyup**).

Приведу список самых часто используемых DOM-событий (см. приложение):

События мыши:

- **click** – происходит, когда кликнули на элемент левой кнопкой мыши (на устройствах с сенсорными экранами оно происходит при касании);
- **contextmenu** – происходит, когда кликнули на элемент правой кнопкой мыши;
- **mouseover / mouseout** – когда мышь наводится на / покидает элемент;
- **mousedown / mouseup** – когда нажали / отжали кнопку мыши на элементе;
- **mousemove** – при движении мыши.

События на элементах управления:

- **submit** – пользователь отправил форму <form>;
- **focus** – пользователь фокусируется на элементе, например нажимает на <input>.

Клавиатурные события:

- **keydown** и **keyup** – когда пользователь нажимает / отпускает клавишу.
- **События документа:**
- **DOMContentLoaded** – когда HTML загружен и обработан, DOM документа полностью построен и доступен.

CSS events:

- **transitionend** – когда CSS-анимация завершена.

Чтобы отличить событие от свойства объекта, перед ним традиционно пишется приставка **on**. *onsubmit, onclick, onchange, onfocus, onblur* и т.д. Например, если мы выберем какой-либо элемент у себя на сайте (у меня это зеленая кнопка **«Консультация»**), и откроем консоль разработчика (клавиша F12 в Google Chrome), то на вкладке **Properties** сможем увидеть список всех свойств для данного элемента.

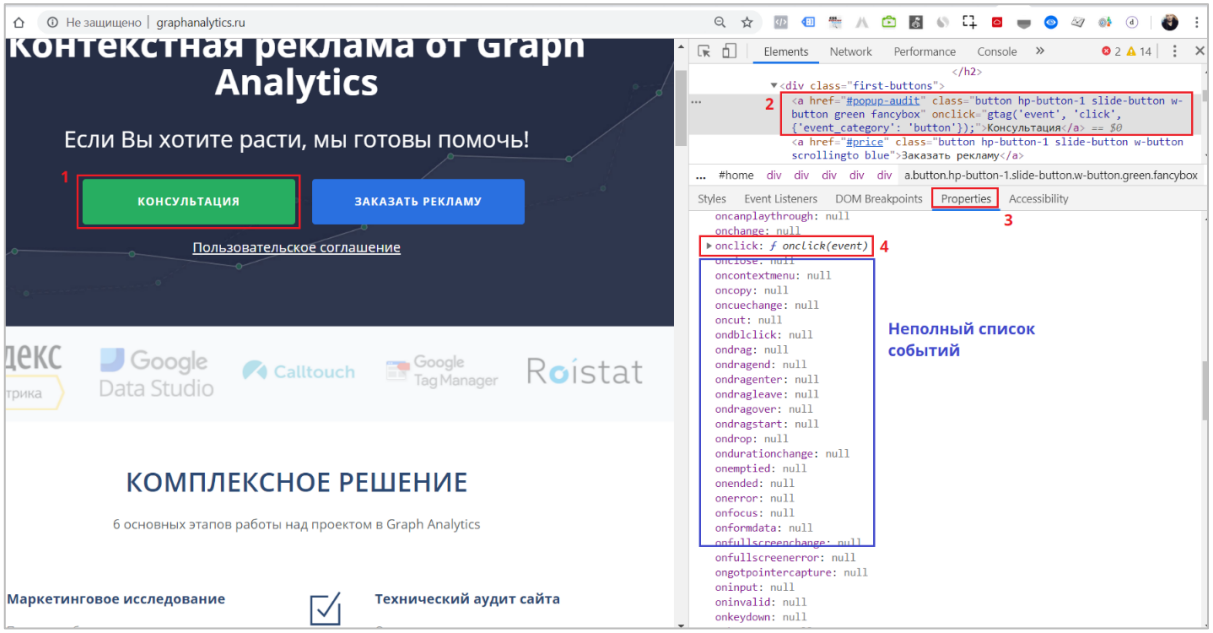


Рис. 821. Вкладка Properties в браузере

На моей кнопке в коде сайта есть событие **onclick** (клик по кнопке, с обработчиком событий для Google Analytics). Это и отображается в **Properties** (цифра 3) в соответствующем свойстве **onclick** (цифра 4) элемента.

Событию можно назначить обработчик, то есть функцию, которая сработает, как только событие произошло. В моем примере — это **gtag('event', 'click', {'event_category': 'button'});**

Обработчик события - функция, которая сработает, как только событие произошло. Именно благодаря обработчикам JavaScript-код может реагировать на действия пользователя. В них указывается тип события и цель.

Тип события - строка, определяющая тип действия, вызвавшего событие. Например, событие **blur** возникает когда элемент теряет фокус, т.е. курсор покидает этот элемент, а **change** происходит, когда содержимое элемента формы, выделения или выбора изменяется (для элементов <*input*>, <*select*>, и <*textarea*>).

Объект события - объект, связанный с определенным событием и содержащий информацию об этом событии. Объекты событий передаются функции обработчика события в виде аргумента. Все объекты событий имеют свойство **type (тип события)** и свойство **target (цель события)**.

Цель события - объект, в котором возникло событие или с которым это событие связано. Например, событие **load** объекта Window или событие **click** элемента <*button*>.

Назначить обработчик события можно несколькими способами:

- в виде атрибута элемента HTML;
- использование свойства DOM-объекта;
- с помощью addEventListener().

Использование атрибута HTML

Способ, который чаще всего показывают в статьях и который интернет-маркетологи используют для прослушивания событий вручную, добавляя обработчик прямо в разметке, в атрибуте, который называется **on<событие>**. Например:

```
  <input type="button" onclick="gtag('event', 'click', {'event_category':
'button'});" value="Кнопка">
```

При клике мышкой по кнопке выполнится код, указанный в атрибуте **onclick**. Этот код передает событие в Google Analytics с действием **click** и категорией **button** (библиотека gtag.js).

Атрибут HTML-тега не чувствителен к регистру, поэтому **ONCLICK** будет работать так же, как **onClick** и **onCLICK**... Но, как правило, атрибуты пишут в нижнем регистре: **onclick**.

Использование свойства DOM-объекта

Можно назначать обработчик, используя свойство DOM-элемента **on<событие>**.

```
  <input type="button" id="button" value="Кнопка">

  <script>
  button.onclick = function() {gtag('event', 'click', {'event_category':
'button'});};
  </script>
```

Два примера кода работают одинаково. При клике мышкой по кнопке выполнится код и в Analytics отправится та же информация по событию.

Регистр DOM-свойства имеет значение. Используйте **button.onclick**, а не **button.ONCLICK**, потому что DOM-свойства чувствительны к регистру.

Так как у элемента DOM может быть только одно свойство с именем onclick, то назначить более одного обработчика так нельзя. Но чтобы обойти это ограничение, можно воспользоваться методом **addEventListener** или **attachEvent**.

С помощью addEventListener()

Синтаксис добавления обработчика:

```
  element.addEventListener(event, handler[, options]);
```

, где:

- **event** - имя события, например "click";
- **handler** - ссылка на функцию-обработчик;
- **options** - дополнительный объект со свойствами.

Таким образом, наша запись будет выглядеть так:

```
<input type="button" id="button" value="Кнопка">

<script>
button.onclick = function handler() {gtag('event', 'click', {'event_category':
'button'});};
button.addEventListener ('click', handler);
</script>
```

Подробнее обо всех способах читайте в этой статье (см. приложение), а про **addEventListener** разобрано в видео (см. приложение):

Для удаления обработчика следует использовать **removeEventListener**:

```
element.removeEventListener(event, handler[, options]);
```

Итак, мы с вами узнали, что браузерных событий очень много и отслеживать их можно различными способами. Плюс они не все по умолчанию прослушиваются в Google Tag Manager. Для этого в самом GTM необходимо произвести дополнительные настройки. Этим мы сейчас и займемся.

Чтобы создать универсальный прослушиватель всех событий, необходимо создать:

- пользовательский тег HTML для каждого типа события, который вы хотите отслеживать (например, change, blur, focus и т.д.)
- переменную типа **Собственный код JavaScript**, которая возвращает функцию-обработчик, которая помещает событие в dataLayer;

Создайте пользовательскую переменную типа **Собственный код JavaScript** и вставьте следующий фрагмент кода (см. приложение):

```
function() {
return function(e) {
window.dataLayer.push({
'event': 'gtm.'+e.type,
'gtm.element': e.target,
'gtm.elementClasses': e.target.className || '',
'gtm.elementId': e.target.id || '',
'gtm.elementTarget': e.target.target || '',
'gtm.elementUrl': e.target.href || e.target.action || '',
'gtm.originalEvent': e
});
}
}
```

В Google Tag Manager это будет выглядеть так:

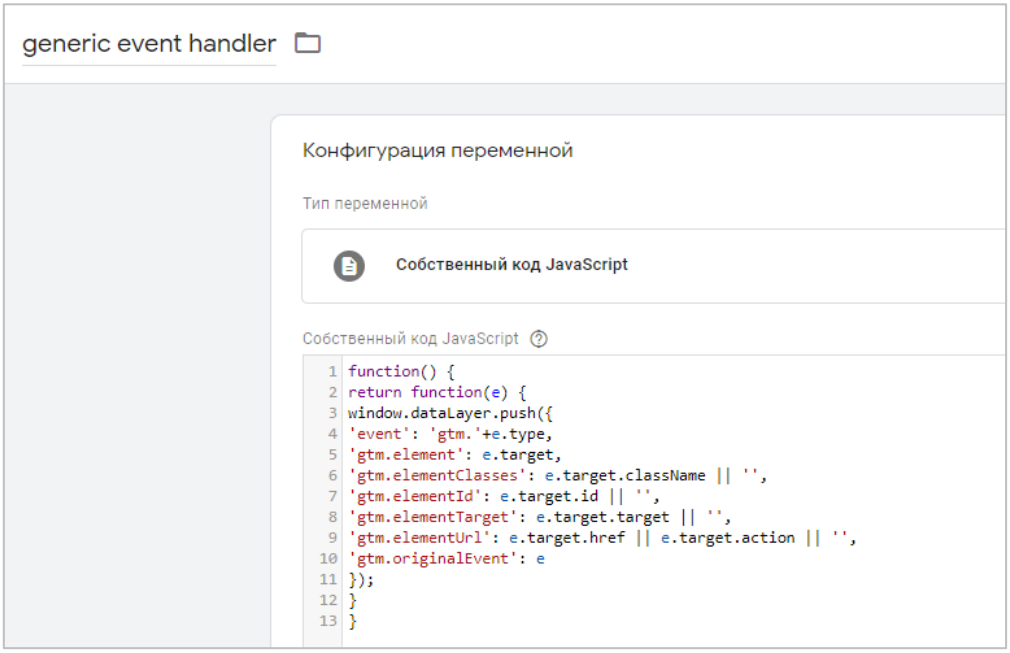


Рис. 822. Переменная Собственный код JavaScript

Вводим название переменной **generic event handler** (или любое другое) и сохраняем изменения. Рассмотрим данный код подробно.

- **return function(e) { ... }** - возвращение результата. Обязательным условием использования данной переменной типа **Собственный код JavaScript** является то, что она должна возвращать результат. Если переменная не будет возвращать функцию, а будет запускать сам код, вы будете отправлять пустые события каждый раз, когда тег прослушивания будет записываться в шаблон страницы.
- **window.dataLayer.push({ ... });** - триггерное событие event помещается в dataLayer (уровень данных);
- **'event': 'event.'+e.type** - берет свойство **(type)** из объекта события, который представляет собой строку, и помещает значение в переменную уровня данных event. Как мы уже разобрались, типы событий могут быть разными – *change, blur, click, submit* и т.д.;
- **'gtm.element': e.target** – помещает элемент, с которым произошло событие в переменную уровня данных **element**;
- **'gtm.elementClasses': e.target.className || ' '** - помещает класс **(class)** цели события в переменную **gtm.elementClasses**. Если в объекте HTML нет класса, вместо него вставляется пустая строка;
- **'gtm.elementId': e.target.id || ' '** - помещает идентификатор **(id)** цели события в переменную **gtm.elementId**. Если в HTML-объекте нет идентификатора, вместо него вставляется пустая строка;
- **'gtm.elementTarget': e.target.target || ' '** – помещает цель **(target)** цели события в переменную **elementTarget**. Если в объекте HTML отсутствует целевой атрибут, вместо него вставляется пустая строка;
- **'gtm.elementUrl': e.target.href || e.target.action || ' '**, помещает атрибут **(href)** или **(action)** цели события в переменную **elementUrl**. Если в HTML-объекте нет таких атрибутов, вместо них вставляется пустая строка;
- **'gtm.originalEvent': e** – доступ к исходному событию.

Теперь нам необходимо создать тег типа **Пользовательский HTML тег** и вставить в него следующий код:

```
<script>
(function() {
var eventType = "change";
if (document.addEventListener) {
document.addEventListener(eventType, {{generic event handler}}, false);
} else if (document.attachEvent) {
document.attachEvent('on' + eventType, {{generic event handler}});
}
})();
</script>
```

В Google Tag Manager это выглядит так:

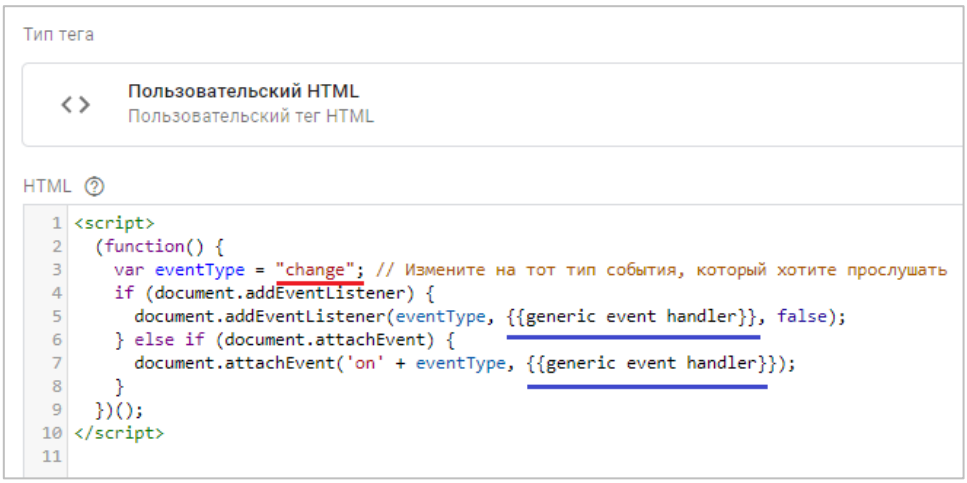


Рис. 823. Пользовательский HTML тег

Триггер активации – **All Pages (Просмотр страницы), gtm.js** в случае, если вы добавляете прослушиватель к узлу документа или **DOM Ready (Модель DOM готова), gtm.dom** когда прослушиваете конкретный элемент HTML.

Красным я выделил тип события, который вы можете заменить на тот, который хотите прослушать (без приставки **on**), а синим пометил места, в которых следует указать в двойных фигурных скобках название переменной типа Собственный код JavaScript с предыдущего шага.

Сохраняем изменения. Когда на странице активируется **Пользовательский HTML** тег, он присоединяет выбранный слушатель к узлу документа. Затем, когда событие происходит, функция обработчика события выполняется с объектом события в качестве его параметра. Этот объект события затем анализируется и помещается в dataLayer с набором свойств, к которым вы можете получить доступ с помощью переменной уровня данных.

В качестве примера я покажу, как можно отследить событие **ondblclick**, которое срабатывает только по двойному щелчку мыши по выбранном элементу. Все, что необходимо сделать, это изменить переменную **eventType** в Пользовательском HTML теге (пишем без on):

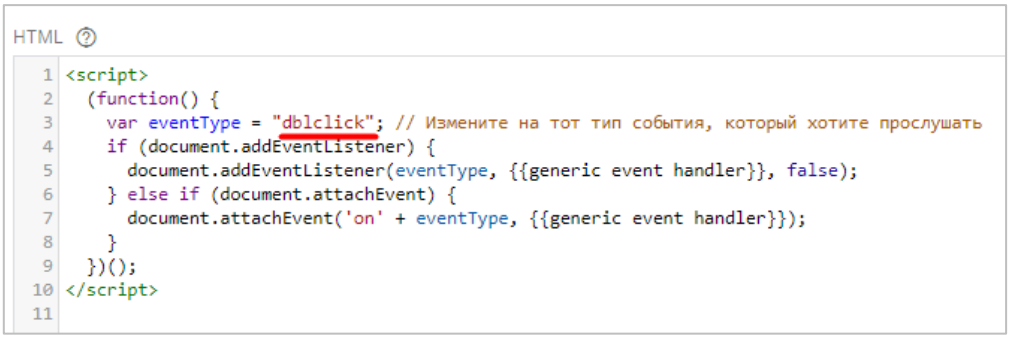


Рис. 824. Пример отслеживания события dblclick

Сохраняем. Затем переходим в режим отладки и кликаем двойным щелчком мыши по любому элементу.

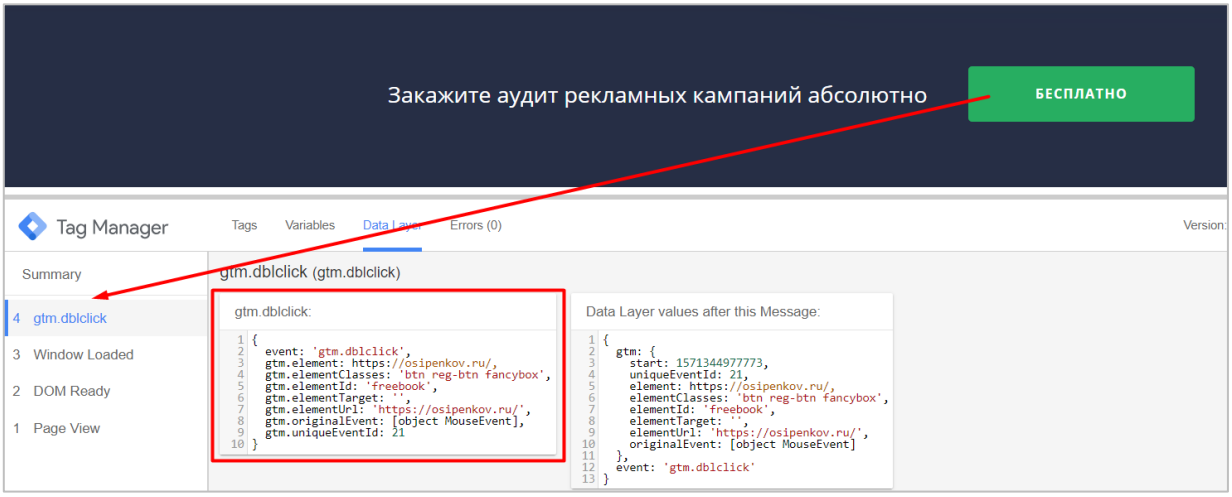


Рис. 825. Уровень данных события gtm.dblclick

Двойной клик по кнопке активировал прослушивание события **gtm.dblclick**. А на уровне Data Layer мы можем видеть все атрибуты данного объекта (класс, id, url и т.д.).

И напоследок, давайте передадим в качестве события данные о двойном щелчке в Google Analytics.

Для это необходимо:

1. создать триггер типа **Пользовательское событие** с именем **gtm.dblclick**

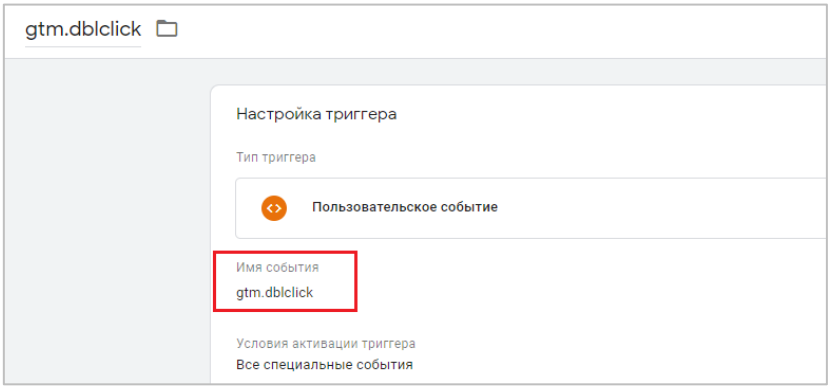


Рис. 826. Пользовательское событие gtm.dblclick

Примечание: вы можете выбрать не все специальные события, а какие-то конкретные, расположенные на определенных страницах, задав соответствующее условие активации.

2. создать тег типа Google Аналитика – Universal Analytics, указав в нем соответствующие данные.

- Тип отслеживания – **Событие**;
- Категория – **Произвольно**;
- Действие – **Произвольно**;
- Триггер активации – наше пользовательское событие **gtm.dblclick**.

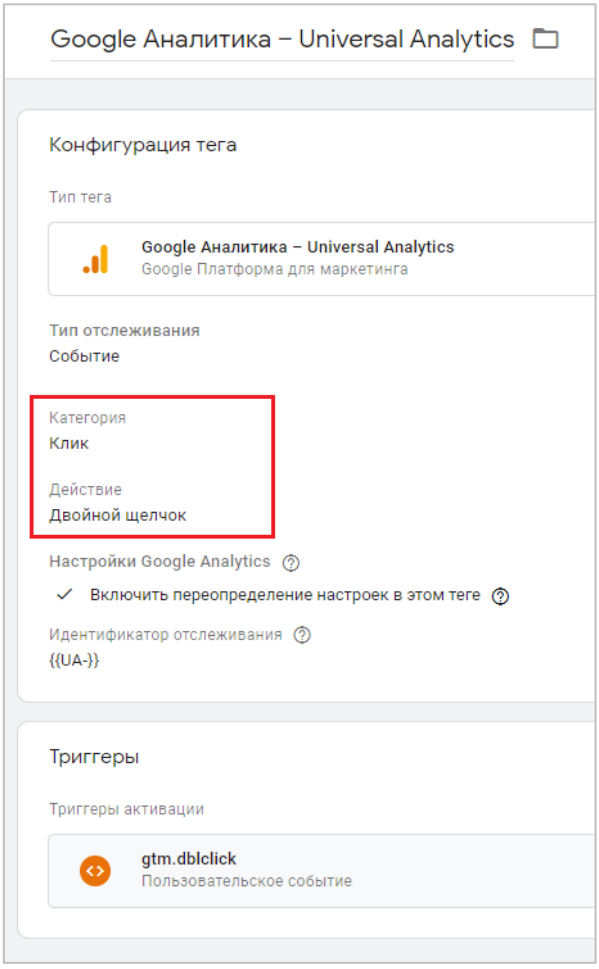


Рис. 827. Настройка тега Google Аналитика - Universal Analytics

Проверяем с помощью режима предварительного просмотра. При двойном щелчке мыши активируется тег:

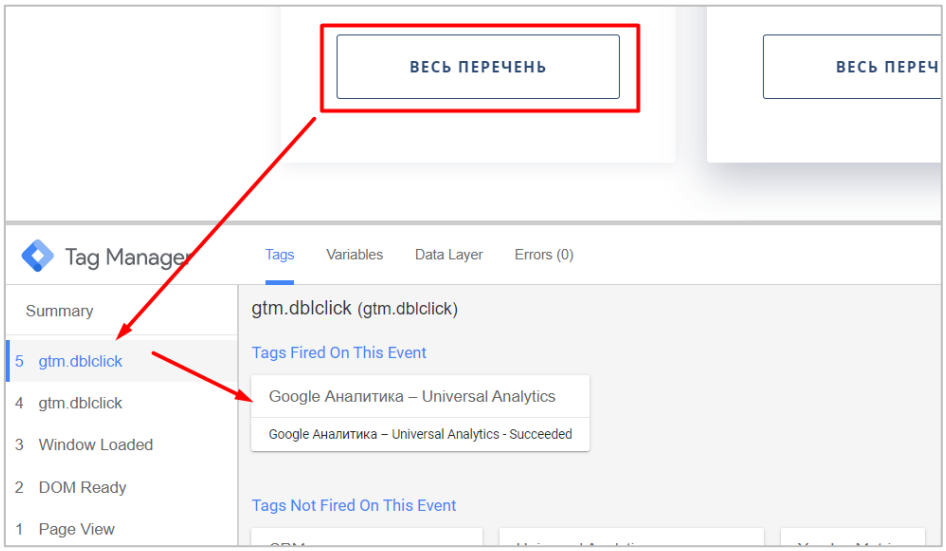


Рис. 828. Активация тега при двойном щелчке мыши

Он отправляет данные о событии в Google Analytics:

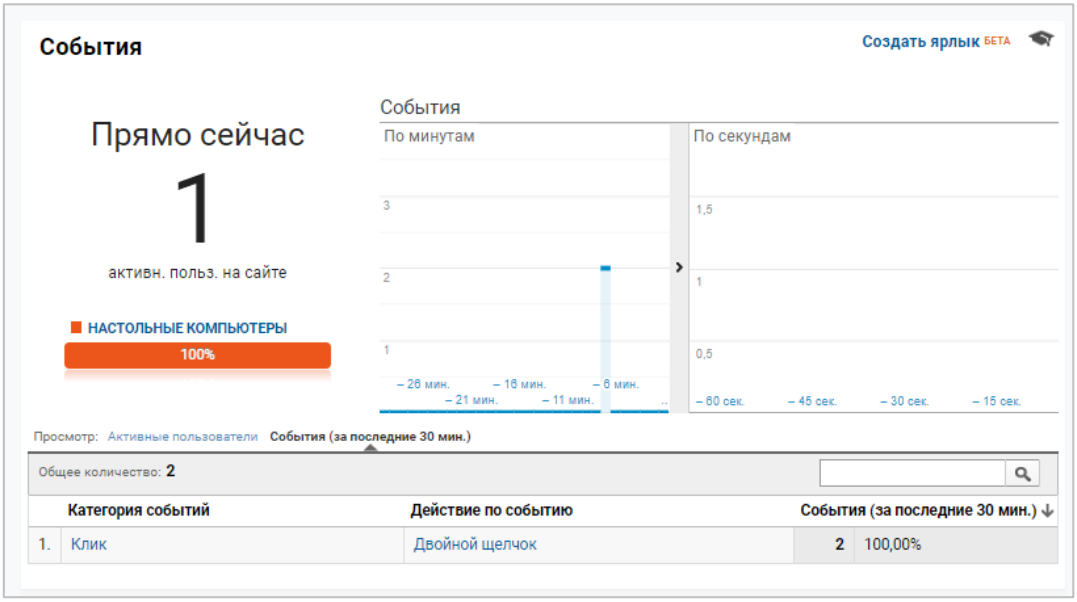


Рис. 829. Отчет В режиме реального времени

Виртуальные страницы

По умолчанию, просмотр страницы в аналитику отправляется каждый раз, когда вы:

- **обновляете веб-страницу** (на которой установлен счетчик веб-аналитики);
- **переходите на новую страницу** (на которой установлен счетчик веб-аналитики).

И в одном и другом случае у страницы есть URL-адрес, который вы видите в адресной строке браузера. И при переходе от страницы к странице он меняется. Например, на моем тестовом сайте **graphanalytics.ru** после заполнения формы на бесплатный аудит рекламных кампаний пользователя перенаправляет на страницу "Спасибо", которая имеет уникальный URL-адрес graphanalytics.ru**/thank-you.html**:

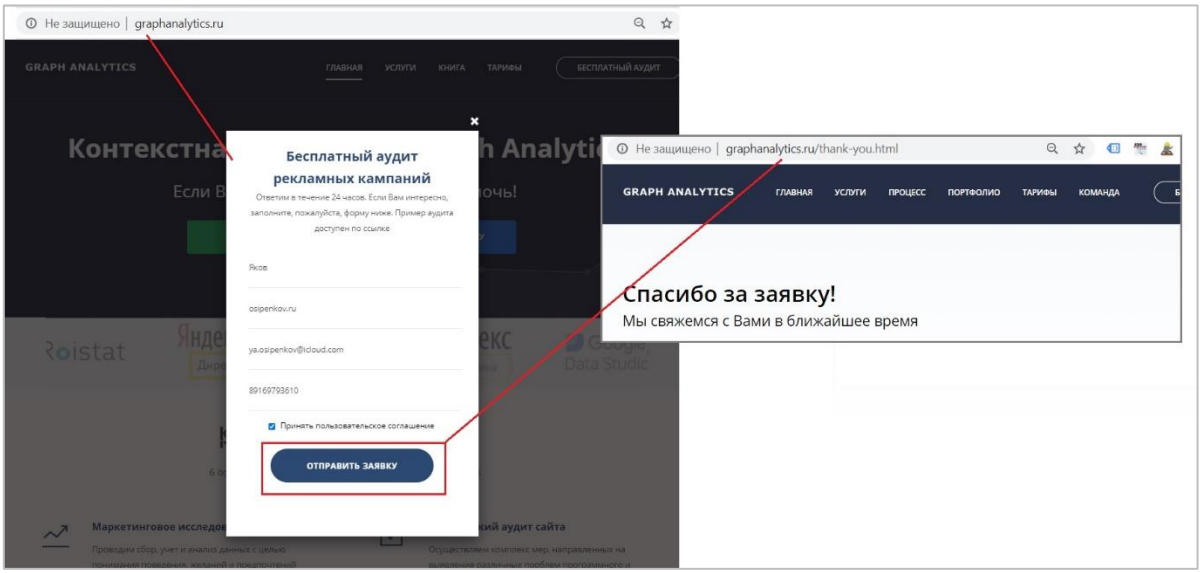


Рис. 830. Пример отправки формы с двумя разными URL-адресами

И при такой реализации все очень просто - мы можем в аналитике настроить цель на просмотр этой страницы, а также в отчетах по страницам видеть статистику по переходам:

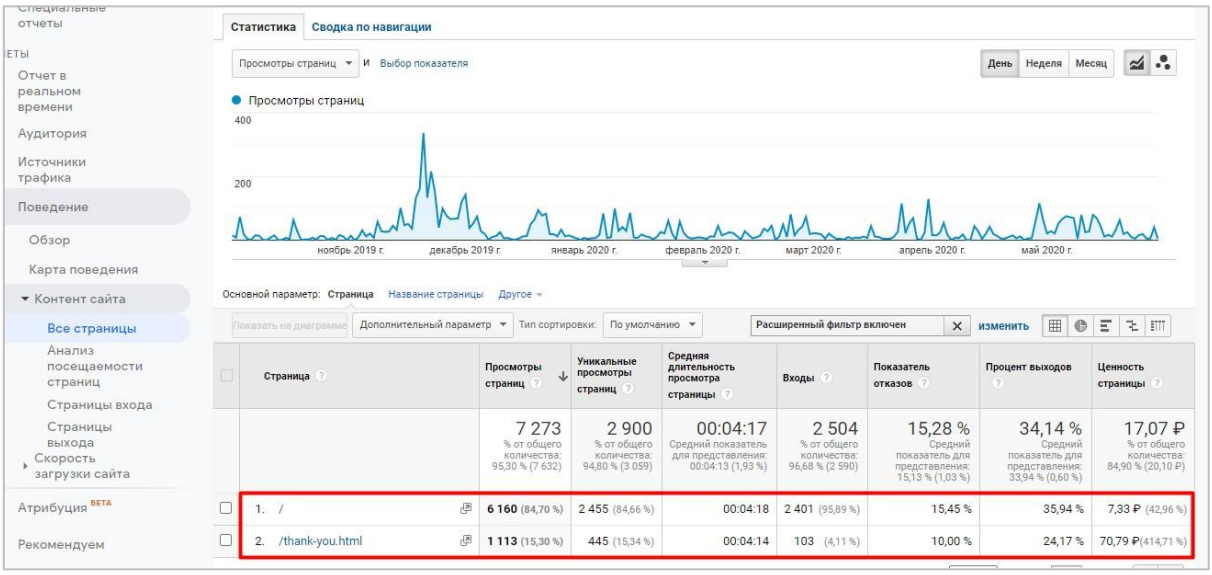


Рис. 831. Статистика по страницам

Но когда дело доходит до одностраничных веб-сайтов, такое отслеживание перестает работать. Одностраничное приложение **Single Page Application (SPA)** - веб-сайт или веб-приложение, в котором все данные, необходимые для переходов между разделами сайта, загружаются вместе с первой страницей, а весь контент загружается динамически. Такое приложение может обновлять URL в адресной строке браузера, имитируя переходы между страницами, однако запрос другой полной страницы при этом не выполняется. Поэтому отслеживание действий на странице по мере загрузки нового контента необходимо реализовывать вручную. Это статья также отвечает на вопрос: *как отследить просмотры страниц для Single Page Application (SPA)?*

Независимо от того, что пользователь делает на сайте, всегда отслеживается **ТОЛЬКО один, первый** просмотр страницы. Например:

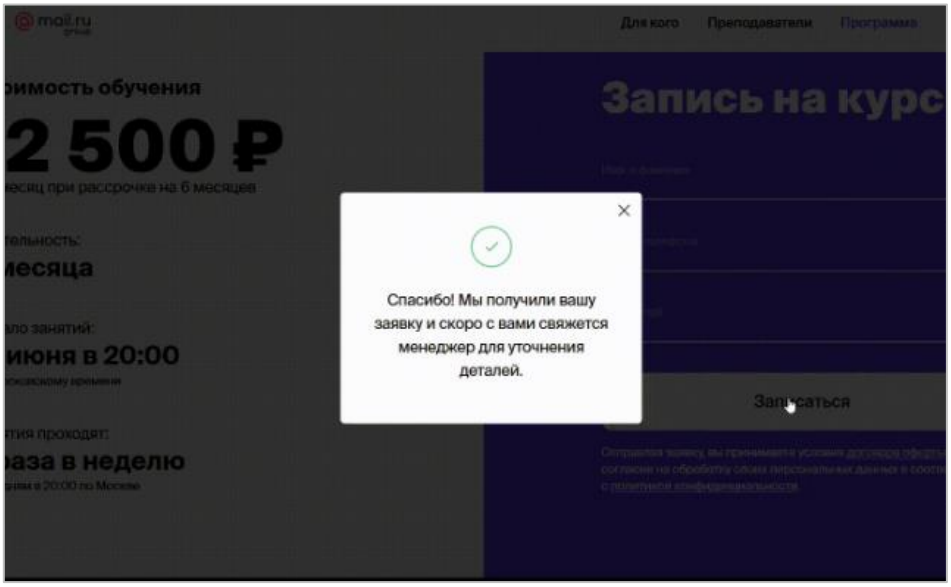


Рис. 832. Пример отправки формы без перезагрузки страницы (без изменения URL)

Контент (на примере выше - окошко благодарности) загружается без обновления страницы и/или без перехода на другую веб-страницу, следовательно, просмотра страницы не происходит, и данные в инструменты веб-аналитики не отправляются.

Как правило, такие сайты используют технологию AJAX для загрузки нового контента на странице. Одним из наиболее ярких примеров является конструктор веб-сайтов **Tilda** (и не только она). Тильда передает данные в системы аналитики на основе просмотра страниц. Когда пользователь совершает какое-либо действие на странице, создается **виртуальная страница**. Данные о том, сколько человек *«посетили»* эту виртуальную

страницу, всегда доступны в системах аналитики без дополнительных настроек. Например, в Google Analytics достижение цели будет выглядеть так:

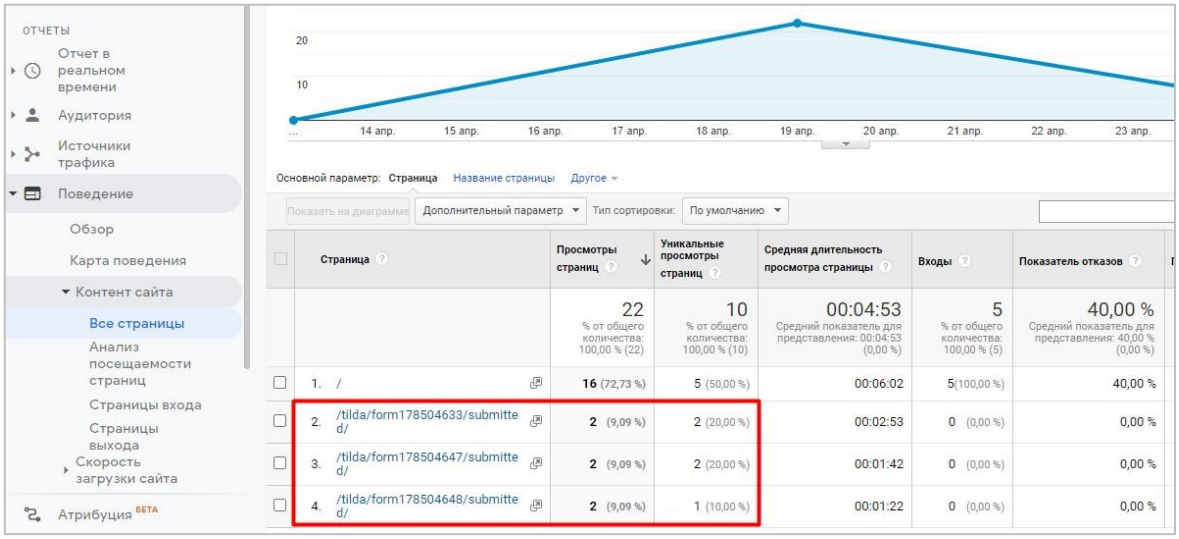


Рис. 833. Достижение целей как виртуальные просмотры страниц (Tilda)

В Тильде заложены определенные настройки, которые позволяют отслеживать открытие попапов, клики по кнопкам и отправку форм. Адрес виртуальной страницы можно посмотреть в настройках блока:

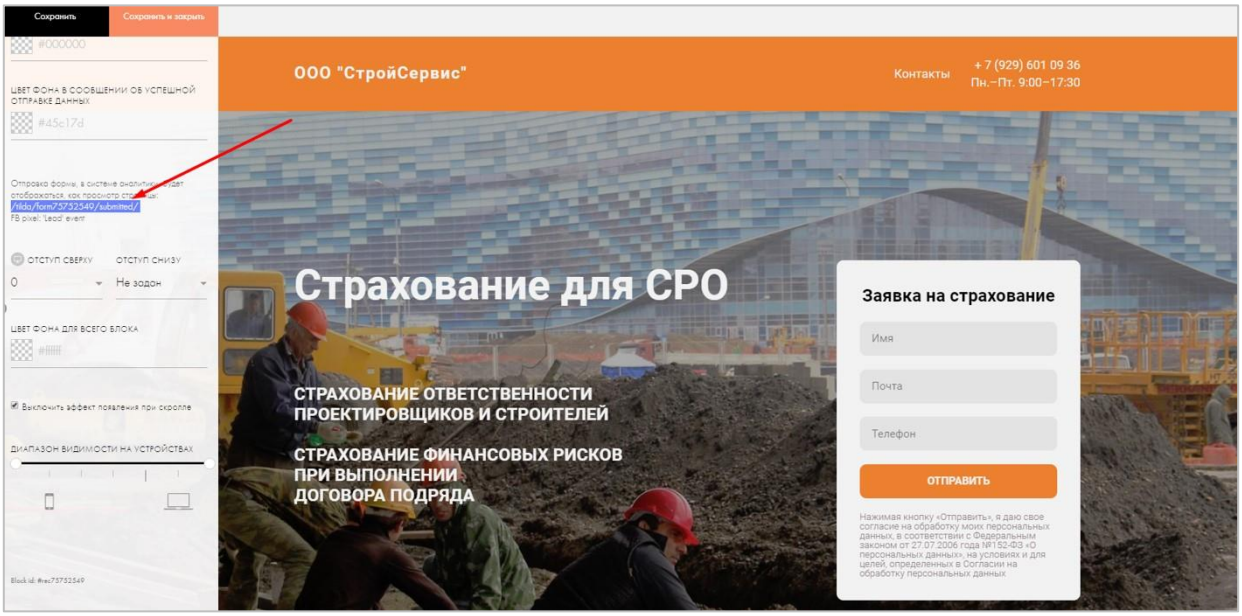


Рис. 834. Пример адреса виртуальной страница проекта в настройках блока (отправка формы)

Примеры того, как выглядят адреса виртуальных страниц в Tilda:

- **tilda/popup/rec********/opened** - для попапа;
- **tilda/click/rec********/button** - для клика на кнопку;
- **tilda/form********/submitted** - для заполнения формы.

где ******** - уникальный номер для каждого блока.

Вот мы и подошли к самому определению **виртуальных страниц**. Я разделю его на две части:

1. отправка обращения к странице в инструменты веб-аналитики (Google Analytics, Яндекс.Метрику и др.) без ее перезагрузки;
2. передача информации в инструменты веб-аналитики о просмотре страницы, хотя на самом деле этой страницы не существует.

Как вы знаете, в событиях конечных URL-адресов и целевых страниц тоже нет. И работая с кнопками или формами на сайте, мы вынуждены использовать виртуальные страницы. Благодаря виртуальным просмотрам страниц мы можем настраивать различные взаимодействия пользователя с контентом сайта, например, отслеживать:

- клики по кнопкам;
- открытие попапов и других всплывающих элементов;
- отправку форм;
- табов;
- заполнение полей в формах;
- страницы с технологией AJAX;
- и др.

С помощью виртуальных страниц возможно создавать визуализацию пути посетителя сайта и различные воронки:

Рис. 835. Визуализация последовательности заполнения полей формы

В дальнейшем, на виртуальные страницы можно настраивать обычные цели типа **Целевая страница** (для Google Analytics) и **Посещение страниц** (для Яндекс.Метрики), а также составные цели.

Виртуальные страницы в Google Tag Manager можно отслеживать несколькими способами:

- с помощью условия активации и поля **page**, которое необходимо задать в дополнительных настройках тега;
- с помощью триггера **Изменение в истории** и соответствующих переменных;
- с помощью уровня данных (dataLayer);
- с помощью тега типа **Пользовательский HTML тег** (для Яндекс.Метрики);

Давайте разберем каждый способ.

Настройка с помощью условия активации и поля page, которое необходимо задать

Классический способ, который позволяет отслеживать не только переходы со страницы на страницу, но и определенные элементы на сайте - поля формы, клик по кнопке, табы и т.д. В качестве примера я разберу отслеживание переключение шагов (табов) на сайте graphanalytics.ru и настрою это действие с помощью виртуальных страниц.

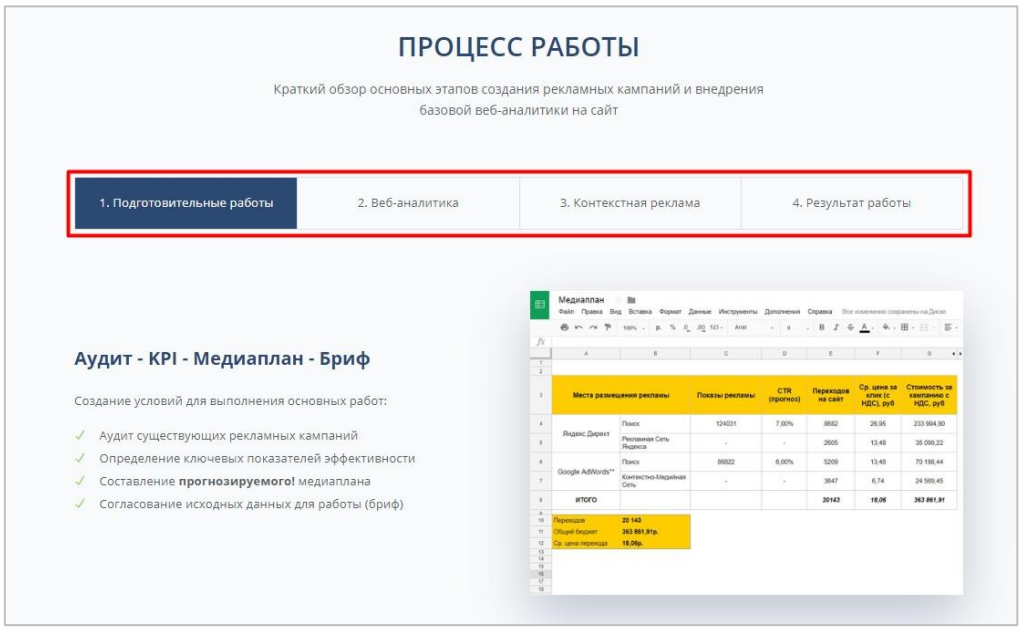


Рис. 836. Отслеживание табов (tab)

При клике на определенный шаг внизу меняется контент (текст + картинка), страница не перезагружается. Чтобы отследить переходы между вкладками и передать эту информацию в инструменты веб-аналитики как просмотр виртуальной страницы, необходимо:

- активировать встроенные переменные типа **Клики** - **Click Element, Click URL, Click Text** и т.д.;
- добавить триггер активации;
- создать теги для передачи данных в аналитику.

Отслеживать я планирую клики по элементам с атрибутом **href**, поскольку у каждого есть ссылка в виде якоря **#**:

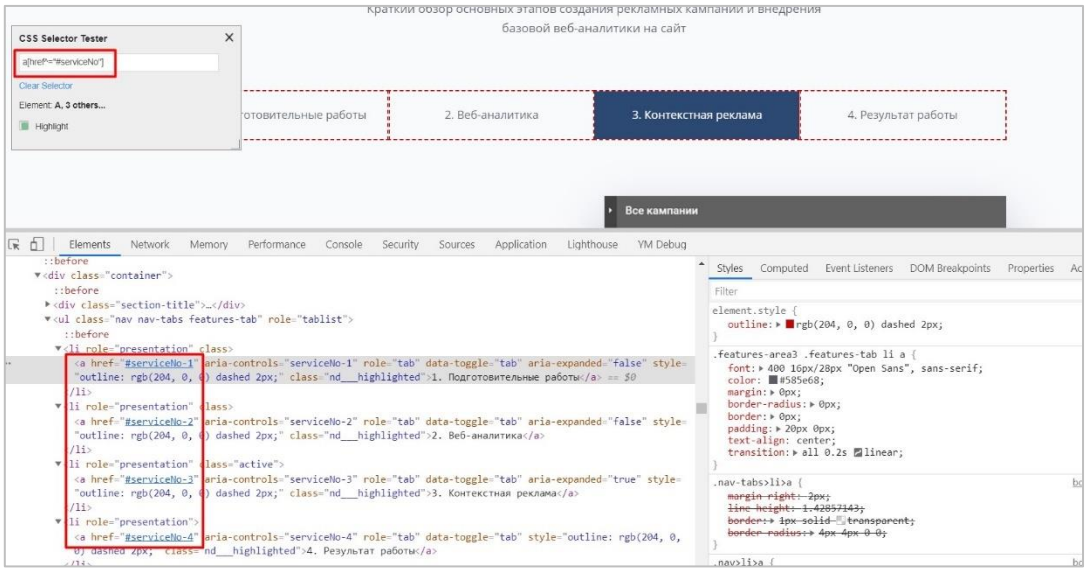


Рис. 837. Якори #

В качестве триггера я выберу **Клики - Все элементы**, а условие активации **Click Element - соответствует селектору CSS - a[href^="#serviceNo"]**

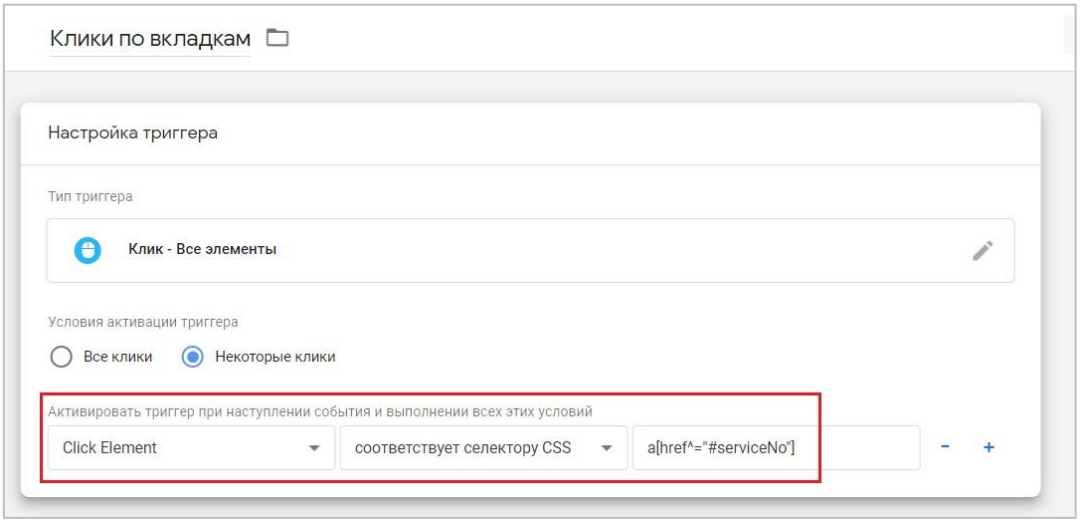


Рис. 838. Триггер активации по вкладкам (табам)

a[href^="#serviceNo"] – выбирает каждый элемент **<a>** с атрибутом **href**, значение которого начинается с **#serviceNo**. Теперь создайте переменную типа **Таблица поиска**, которая понадобится нам для отображения значения шага в пути виртуальной страницы. Выглядеть она будет так:

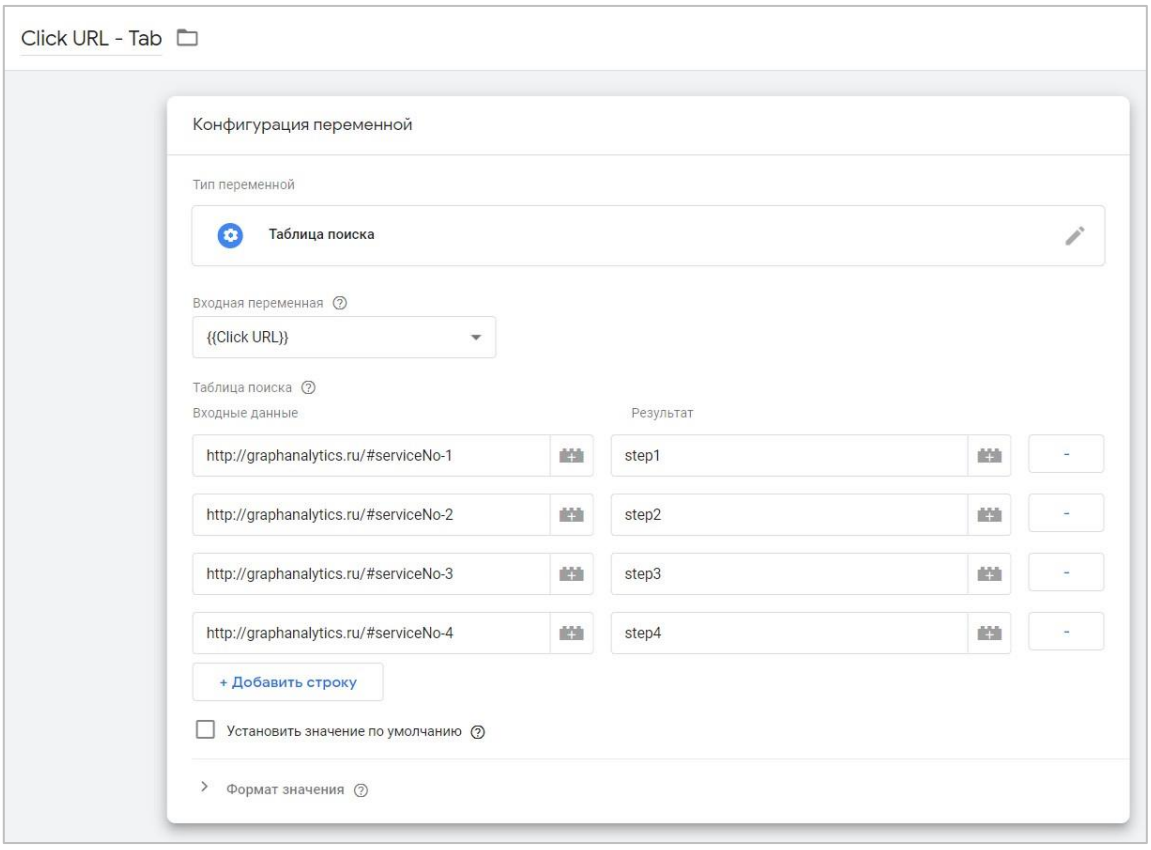


Рис. 839. Переменная Таблица поиска

Логика простая - если пользователь кликает по ссылке с serviceNo-1, то результатом будет **step1**, если serviceNo-2, то **step2** и т.д.

Все, что осталось сделать, это создать тег типа **Google Аналитика - Universal Analytics** с типом **Просмотр страницы**. В дополнительных настройках к тегу в поле **Поля, которые необходимо задать** следует указать **page** и адрес виртуальной страницы, который заполняется произвольно. Я воспользуюсь созданной на предыдущем шаге переменной **Таблица поиска** и добавлю значение URL так:

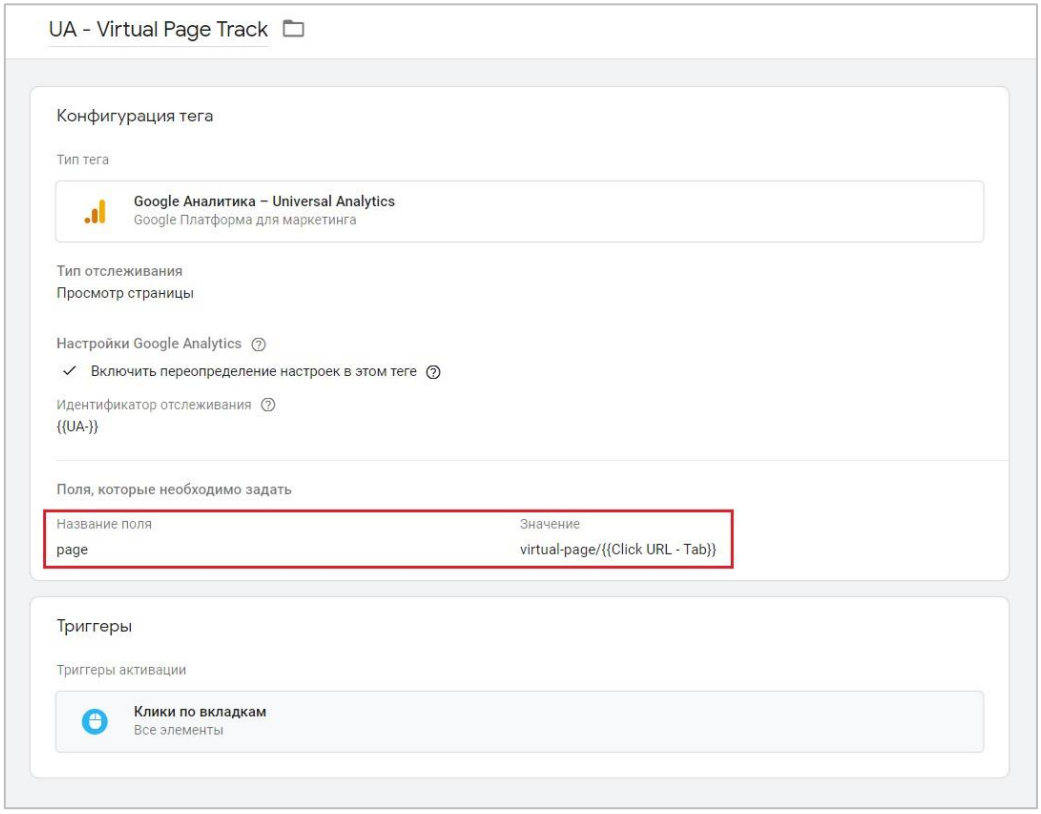


Рис. 840. Тег с передачей виртуальной страницы в Google Analytics

Триггер активации - **Клик по элементам**, созданный на предыдущем шаге. Это означает, что при клике на любую из 4 вкладок в Google Analytics будет отправляться просмотр виртуальной страницы с URL **/virtual-page/step1-4** шагом, в зависимости от конкретной вкладки. Сохраняем изменения.

Для Яндекс.Метрики необходимо создать тег типа **Пользовательский HTML тег** и вставить следующий код:

```
<script>
ym(XXXXXX, 'hit', '/virtualpage/{{Click-URL - Tab}}');
</script>
```

, где **XXXXXX** - номер вашего счетчика Яндекс.Метрики.

В GTM это выглядит так:

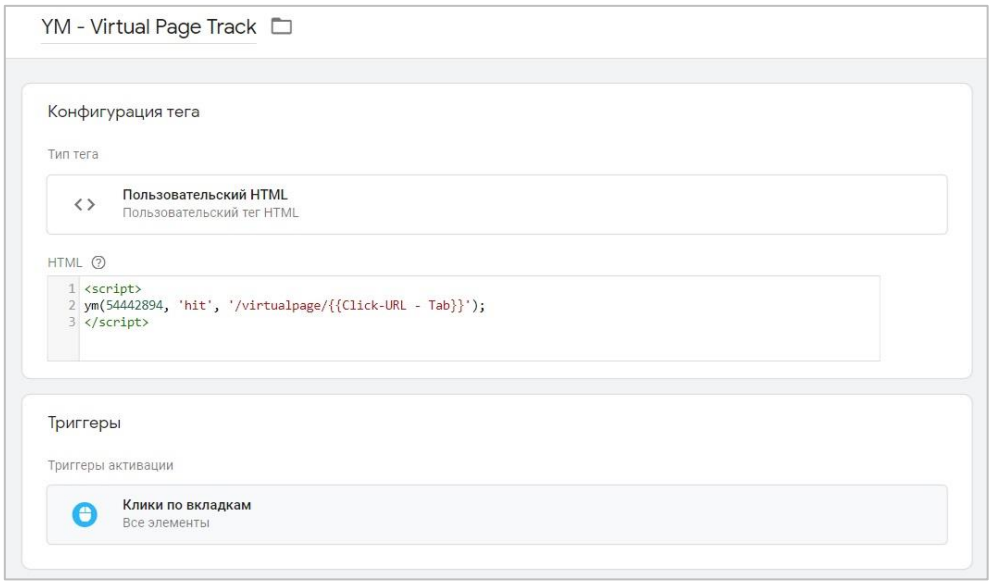


Рис. 841. Тег для Яндекс.Метрики

Триггер активации - **Клик по элементам**, созданный на предыдущем шаге. Сохраняем изменения. Проверить корректность настройки можно с помощью режима отладки GTM. Я перешел по вкладкам несколько раз туда-сюда. В отчетах Google Analytics **В режиме реального времени** на вкладке **Контент** я мог наблюдать такую картину:

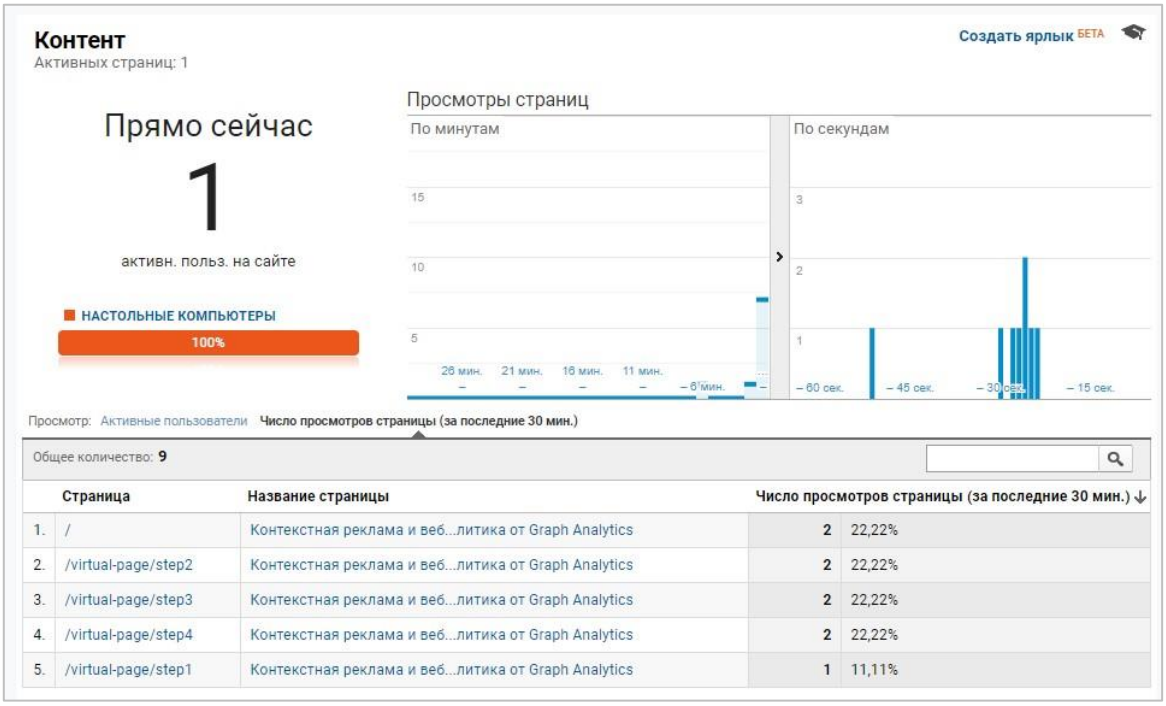


Рис. 842. Отчет В режиме реального времени

Как видим, данные успешно передаются. Шаг (/step1-4) из таблицы поиска корректно подставляется в конец URL. Далее информацию по виртуальным страницам можно найти в стандартных отчетах **Поведение - Контент сайта - Все страницы**. В Яндекс.Метрике статистика через некоторое время отобразится в отчете **Содержание - Популярное**:

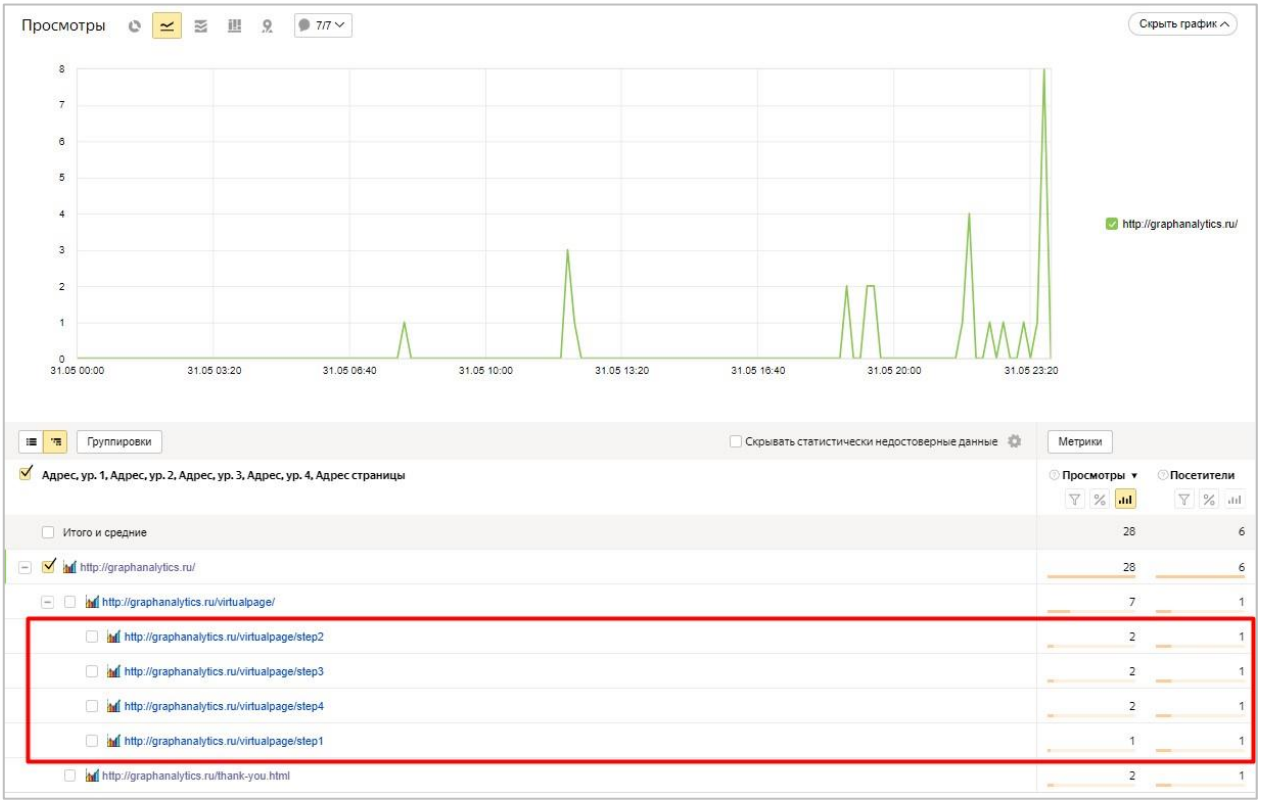


Рис. 843. Виртуальные просмотры страниц в Яндекс.Метрике

Это был первый и классический способ отслеживания виртуальных страниц с помощью Google Tag Manager. Давайте перейдем к следующему.

Настройка с помощью триггера Изменение в истории и переменных

При условии активации триггера **Изменение в истории** срабатывает событие **gtm.historyChange**, если изменился фрагмент URL (хеш, #). Но это не всегда хеш. Динамическое окончание в URL может выглядеть как на обычном веб-сайте, например, **/home**, **/contacts** и т.д.

Как правило, используется на сайтах, контент которых подгружается динамически, без перезагрузки страниц. В GTM можно использовать со встроенными переменными:

- **New History Fragment** – возвращает переменную, которая содержит новое значение хеша (#) URL-сайта после совершения пользовательского события **Изменение в истории**. Содержится в ключе **gtm.newUrlFragment**.
- **Old History Fragment** – обратное действие, возвращает переменную, которая содержит предыдущее значение хеша URL-сайта до совершения пользовательского события. Содержится в ключе **gtm.oldUrlFragment**.
- **New History State** – возвращает объект, содержащий новое состояние истории после того, как произошло событие и метод **history.pushState()** был выполнен. Содержится в ключе **gtm.newHistoryState**.
- **Old History State** – возвращает объект, содержащий старое состояние истории перед тем, как произошло событие и метод **history.pushState()** был выполнен. Содержится в ключе **gtm.oldHistoryState**.
- **History Source** – возвращает строку-событие, которое привело к изменению объекта истории. Например, **pushState** или **popstate**.

Примечание: если при изменении URL-адреса используется хеш (#), Google Analytics по умолчанию его не перехватит и не отобразит в своих отчетах.

Чтобы использовать этот способ, нужно убедиться в том, что на вашем сайте контент подгружается динамически и после кликов на определенные элементы страницы в адресной строке браузера появляются дополнительные метки. Самый простой способ проверить - активировать все встроенные переменные типа **История** и триггер **Изменение в истории**.

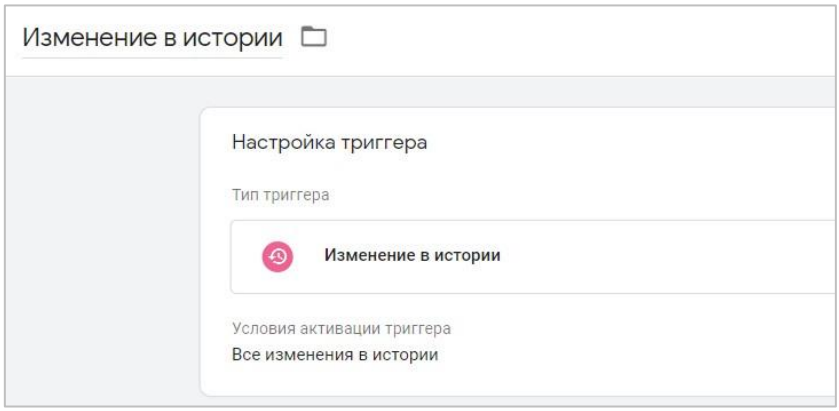


Рис. 844. Триггер Изменение в истории

Примечание: если контент на сайте подгружается динамически, то при переходе со страницы на страницу режим предварительного просмотра не будет перезагружаться. Это один из признаков одностраничного сайта. Если после перехода с одной страницы на другую Debug Mode на какое-то время скрывается и появляется вновь, но уже в новом состоянии, это свидетельствует о том, что у вас, вероятнее всего, не **Single Page Application (SPA)**.

На сайте **graphanalytics.ru** можно привести пример отслеживания кликов по меню навигации, поскольку клик по любому из заголовков отправляет пользователя на соответствующий блок на странице без перезагрузки.

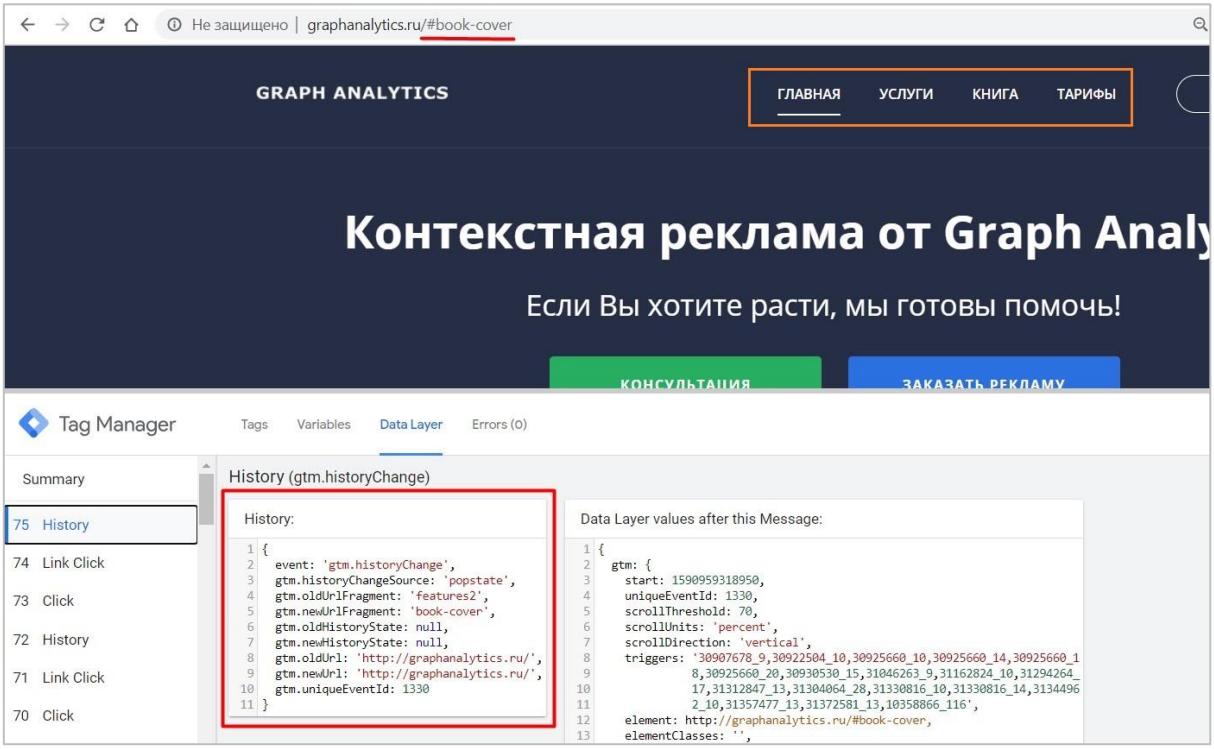


Рис. 845. Пример использования триггера

После клика по одному из элементов в шапке меню, в хронологии Debug Mode отобразилось событие **History**. А на вкладке **Data Layer** мы можем наблюдать значения всех переменных уровня данных:

Переменные принимают соответствующие значения:

- **History Source** – **popstate**, поскольку всякий раз, когда пользователь переходит к новому состоянию, происходит событие popstate, и свойство события state содержит копию объекта записи истории.

Данное значение будет и тогда, когда мы используем переходы по кнопкам **Вперед — Назад** в браузере. К тому же, браузеры работают с событием popstate по-разному. Chrome и Safari всегда вызывают popstate по окончании загрузки страницы, а Firefox не делает этого.

- **New History Fragment (gtm.newUrlFragment)** – **book-cover**, так как мы осуществили переход с graphanalytics.ru/#features2 на graphanalytics.ru/#book-cover (активная ссылка на рисунке выше);
- **Old History Fragment (gtm.oldUrlFragment)** – **features2**;
- **New History State** и **Old History State (gtm.newHistoryState и gtm.oldHistoryState)** – **null**, пустое значение.

Чтобы передать в теге виртуальный просмотр страницы с той частью, которую мы определили, нам необходимо создать все тот же тег **Google Аналитика - Universal Analytics** с типом **Просмотр страницы**. В дополнительных настройках к тегу в поле **Поля, которые необходимо задать** следует указать **page** и адрес виртуальной страницы (по аналогии со способом №1). Только вместо таблицы поиска, в значении виртуального просмотра страницы мы будем использовать встроенную переменную **New History Fragment:**

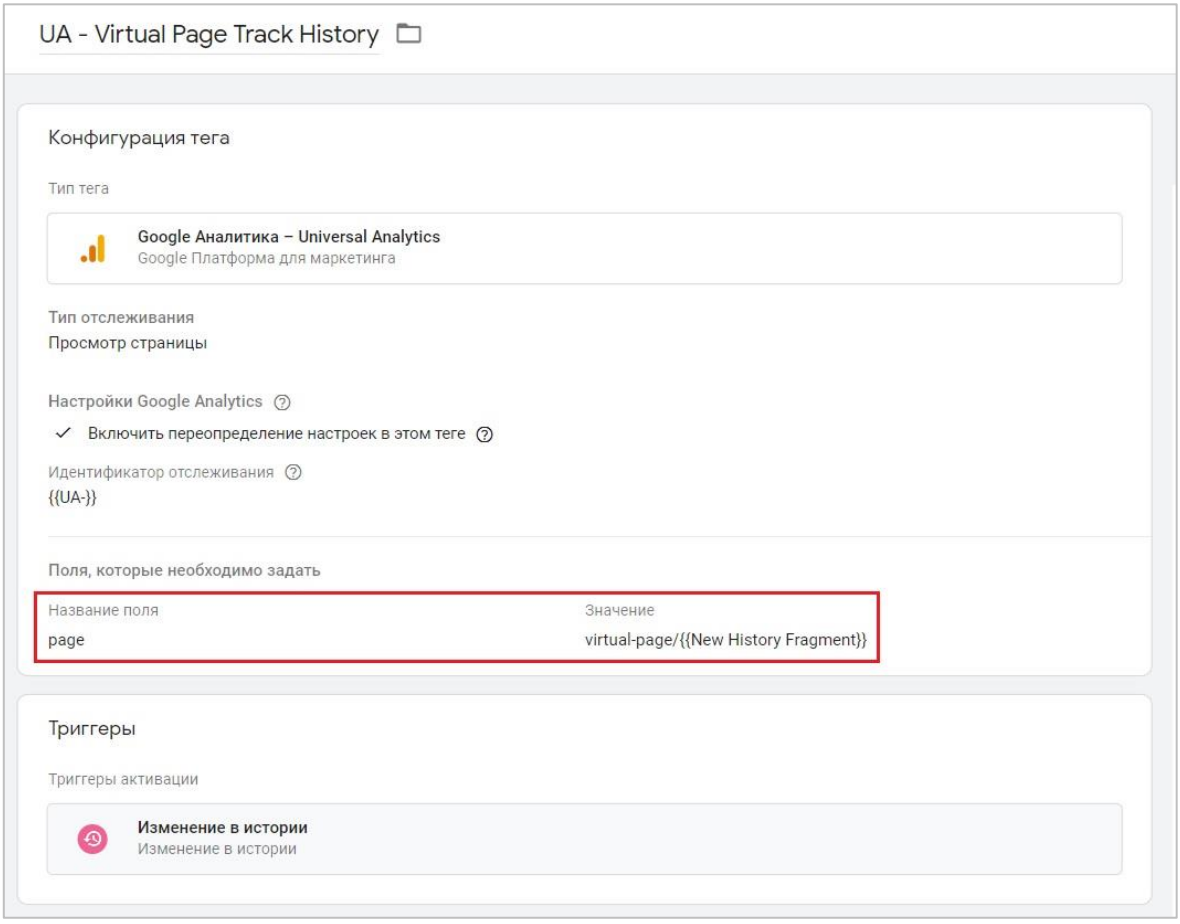


Рис. 846. Тег с передачей виртуальной страницы в Google Analytics

Триггер активации - **Изменение в истории**. Сохраняем изменения. Я перешел по вкладкам меню несколько раз туда-сюда. В режиме реального времени в Google Analytics информация отобразилась:

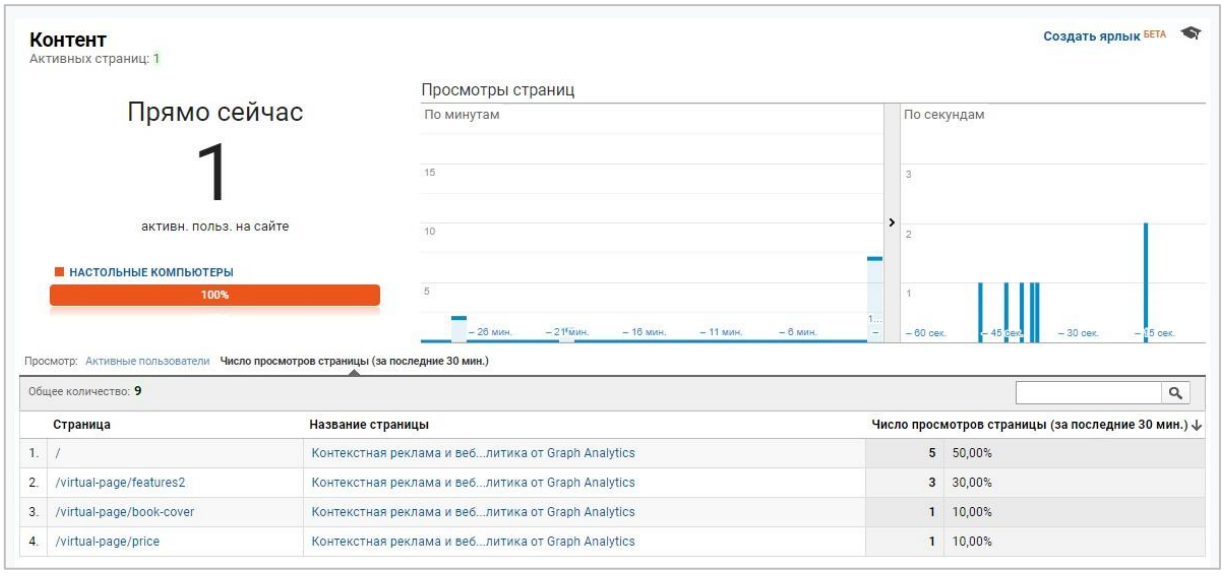


Рис. 847. Отчет В режиме реального времени

В Яндекс.Метрике тег типа **Пользовательский HTML тег** будет выглядеть так:

```
<script>
ym(XXXXXX, 'hit', '/virtualpage/{{New History Fragment}}');
</script>
```

, где **XXXXXX** - номер вашего счетчика Яндекс.Метрики.

В GTM:

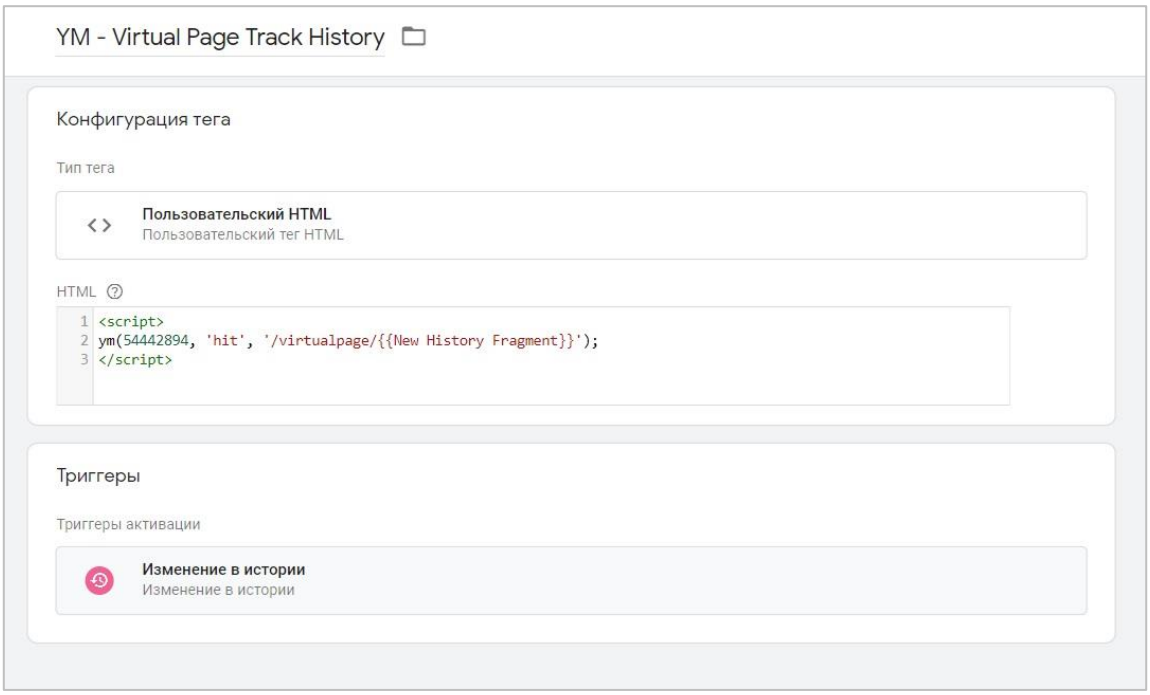


Рис. 848. Пользовательский HTML тег для Яндекс.Метрики

Данные в отчете Яндекс.Метрики:

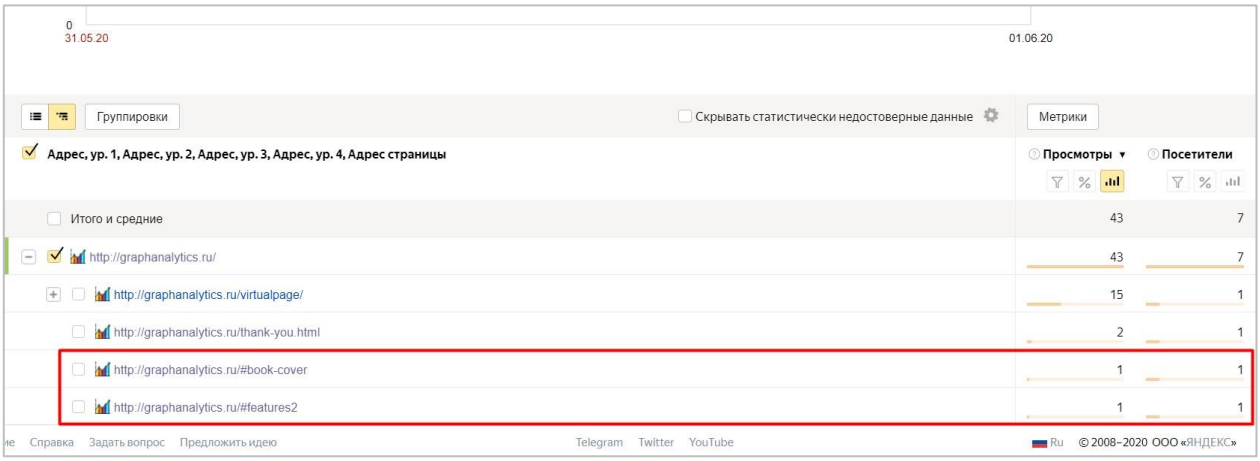


Рис. 849. Данные в отчете Метрики

Примечание: в настройках Яндекс.Метрики есть опция **Отслеживание хеша в адресной строке браузера**, которая позволяет корректно собирать данные о просмотре страниц, даже если сама страница при этом не перезагружалась, а на ней лишь обновилась часть URL после хеша.

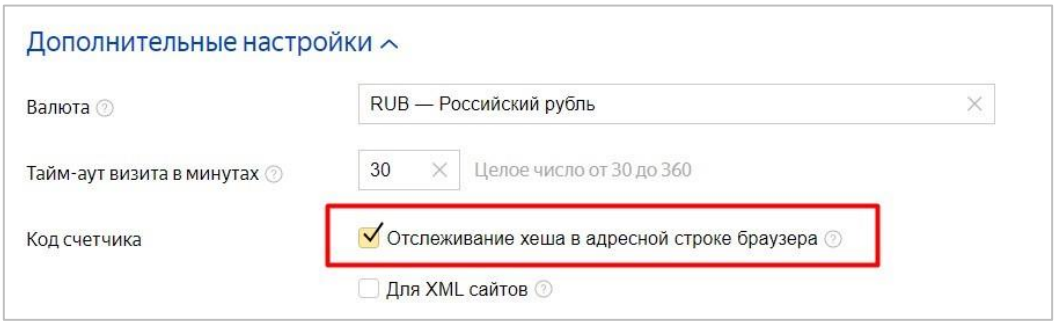


Рис. 850. Отслеживание хеша в адресной строке браузера в Яндекс.Метрике

Подробнее об этом читайте в блоге Яндекс.Метрики (см. приложение). Это был второй способ отслеживания виртуальных страниц с помощью Google Tag Manager.

Настройка с помощью уровня данных (dataLayer)

Если по какой-то причине триггер **Изменение в истории** не работает, есть еще один способ отслеживания одностраничного веб-сайта/приложения с помощью диспетчера тегов Google. Попросите разработчика добавить код **dataLayer.push** на те элементы страницы сайта, которые вы хотите отслеживать и передавать в качестве виртуальных страниц, когда пользователь перемещается между ними.

Пример кода, который разработчик может использовать:

```
<script>
 window.dataLayer = window.dataLayer || [];
 window.dataLayer.push({
 'event': 'Pageview',
 'pagePath': '/virtual-page/', // путь к виртуальной странице
 'pageTitle': 'title' // название страницы (Title)
 });
</script>
```

Примечание: переменные **pagePath** и **pageTitle** должны динамически изменяться на URL-адрес и заголовок страницы, которую в данный момент времени просматривает пользователь.

Каждый раз, когда пользователь заходит в определенный раздел вашей страницы, разработчик должен активировать этот JavaScript-код. Чтобы извлечь значения из этих переменных, необходимо создать пользовательские переменные типа **Переменная уровня данных** со следующими значениями:

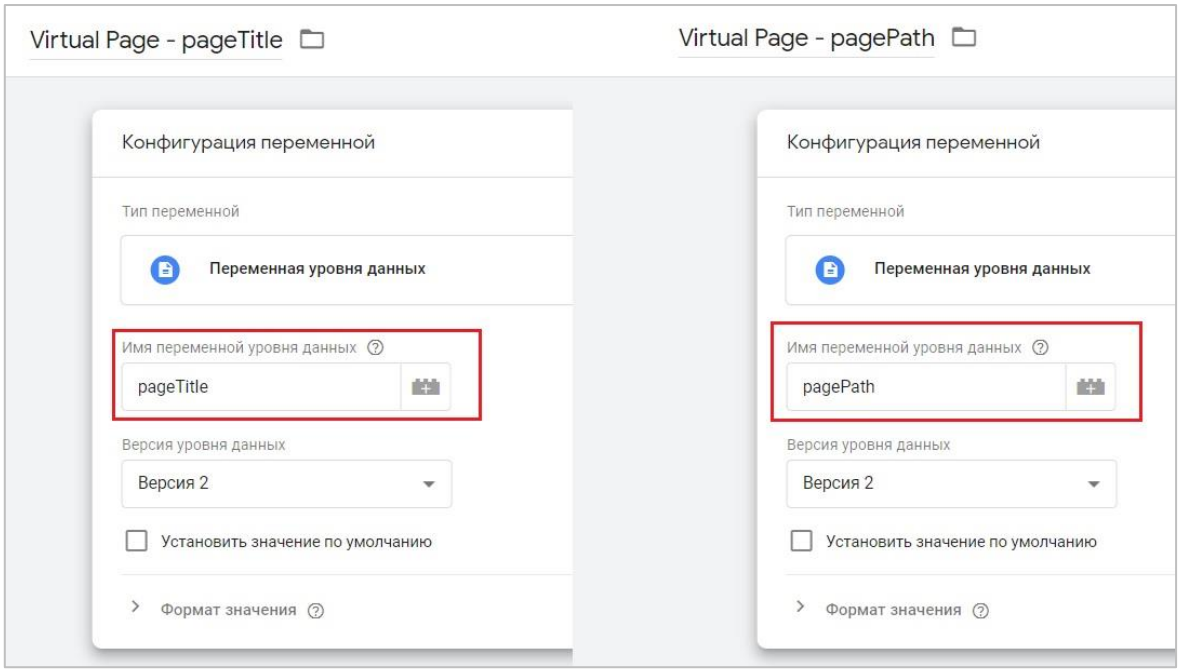


Рис. 851. Переменные уровня данных pagePath и pageTitle

Вы также должны создать триггер типа **Пользовательское событие** с именем **Pageview** (как в примере кода):

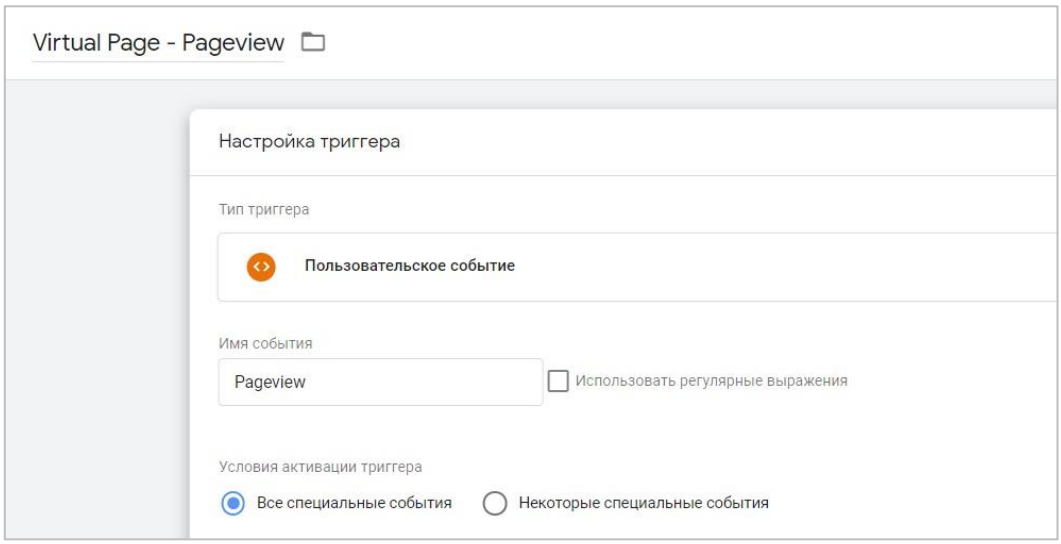


Рис. 852. Пользовательское событие Pageview

В теге **Google Аналитика - Universal Analytics** с типом **Просмотр страницы** в дополнительных настройках к тегу в поле **Поля, которые необходимо задать** теперь следует указать не только **page**, но и **title**, чтобы передать в отчеты Google Analytics название заголовка страницы:

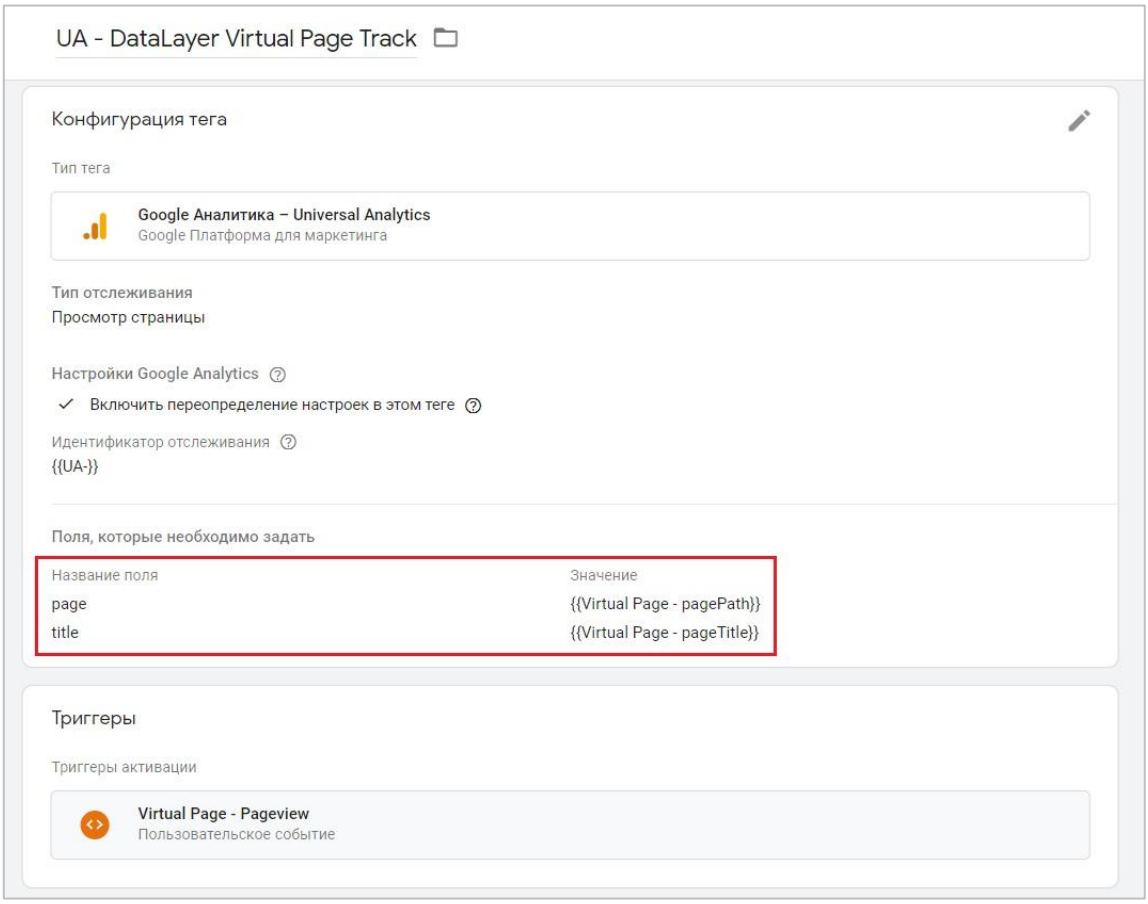


Рис. 853. Настройка тега Google Аналитика – Universal Analytics

Триггер активации - **Пользовательское событие**. Сохраняем изменения.

Также как и в предыдущих способах отслеживания, статистика по виртуальным страницам будет доступна в Google Analytics в стандартных отчетах **Поведение - Контент сайта - Все страницы**, а в Яндекс.Метрике в отчете **Содержание - Популярное.**

Настройка цели в Google Analytics

Чтобы отслеживать виртуальные просмотры страниц как конверсии, необходимо настроить цель в Google Analytics с типом **Целевая страница**, задавав URL. Например, для моего примера с табами (см. выше) это можно сделать так:

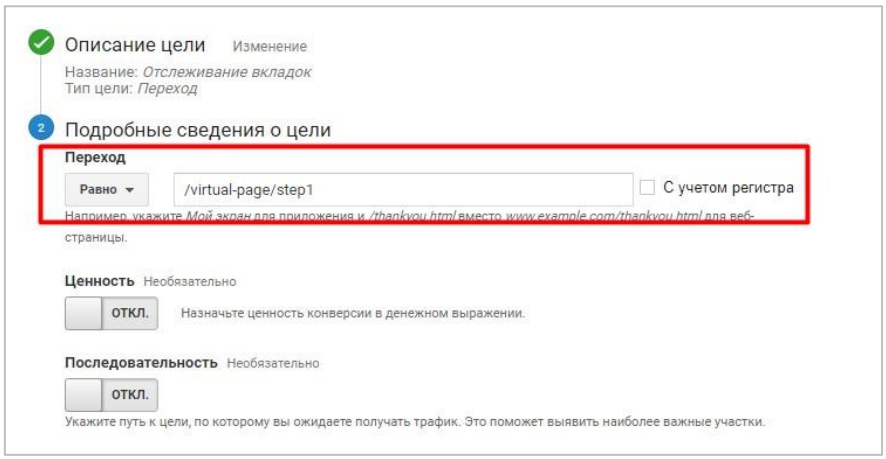


Рис. 854. Настройка цели для виртуальной страницы в Google Analytics

Для отслеживания всех шагов (step1-step4) можно задать условие с помощью регулярного выражения:

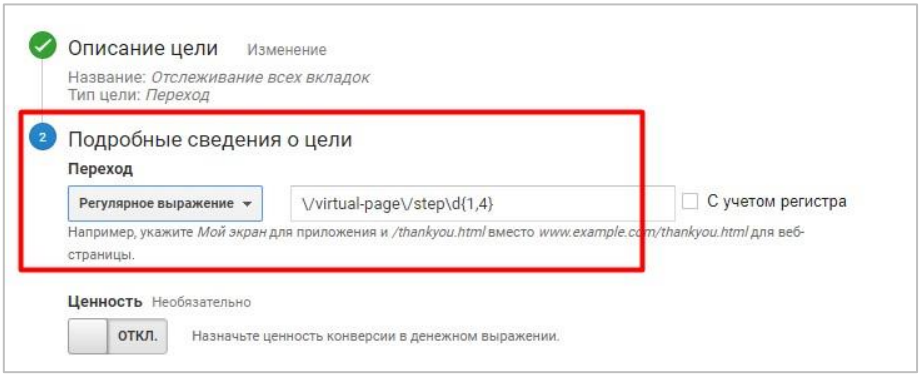


Рис. 855. Настройка с помощью регулярного выражения

где **\d{1,4}** - работает в диапазоне чисел от 1 до 4, как в нашем примере с шагами step1-step4.

Настройка цели в Яндекс.Метрике

Добавьте новую цель типа **Посещение страницы** и укажите URL-адрес виртуальной страницы с условием **url: содержит**. В моем примере для одного из шагов:

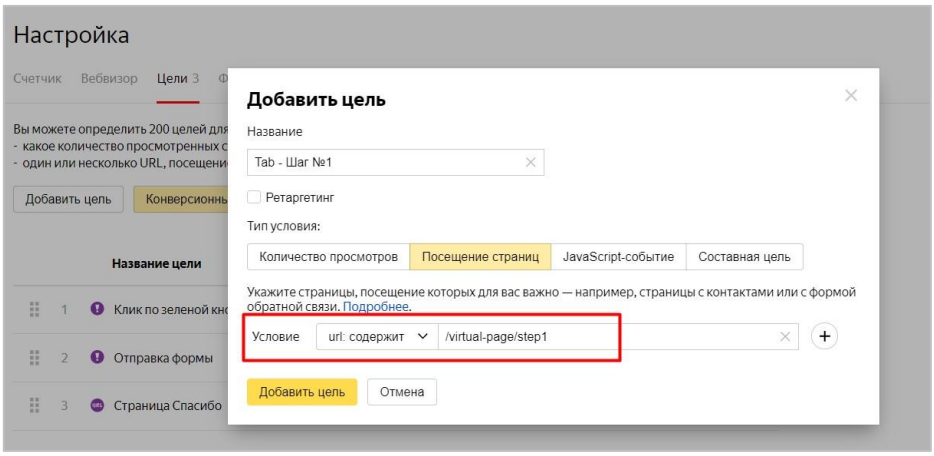


Рис. 856. Настройка цели для виртуальной страницы в Яндекс.Метрике

Визуализация последовательности заполнения полей формы

Предположим, у нас на сайте есть форма, в которой N-ое количество полей, и мы хотим отслеживать статистику не только по количеству отправленных форм (достижению цели), но еще по тому, какие поля заполняют пользователи. И все эти последовательности визуализировать. Для этого нам понадобится 2 тега, 3 переменных и 1 триггер.

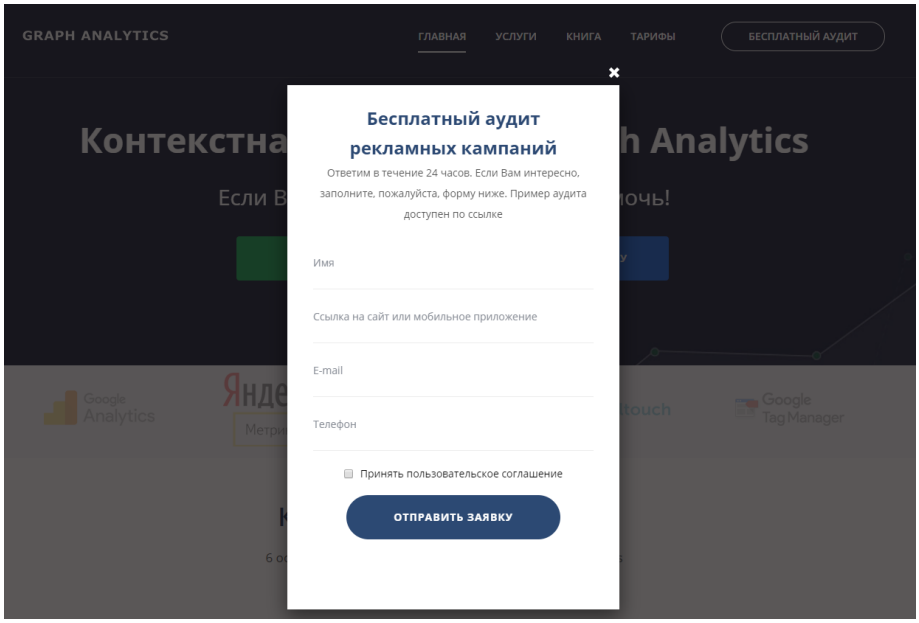


Рис. 857. Пример формы и полей на сайте (graphanalytics.ru)

Перейдите в диспетчер тегов Google и создайте **Пользовательский HTML** тег. Вставьте нижеописанный код:

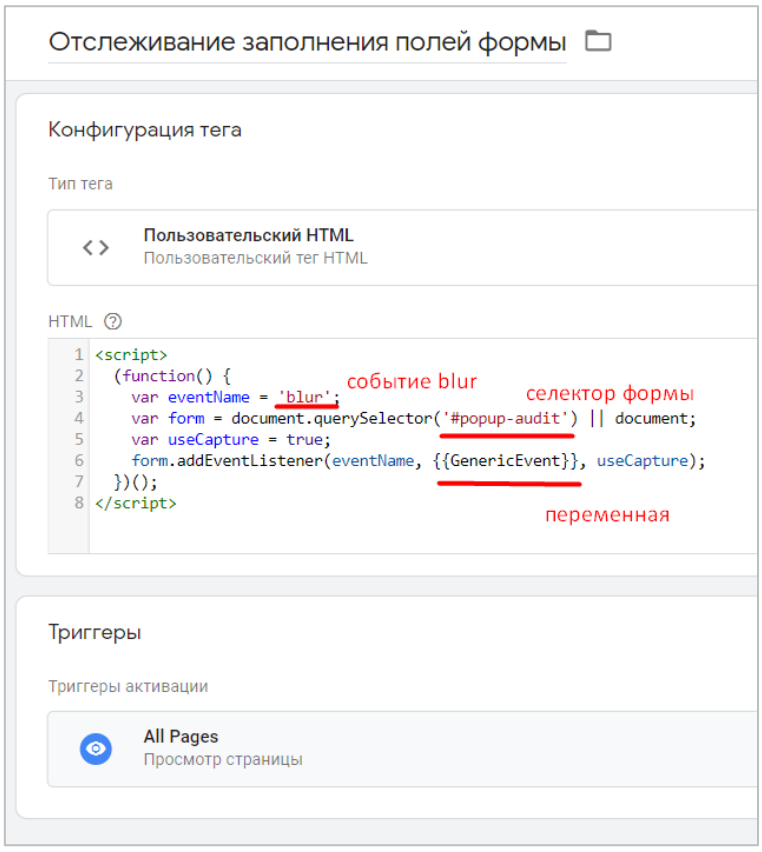


Рис. 858. Пользовательский HTML тег – событие blur

- событие **blur/onblur** – это браузерное стандартное событие типа onclick. Возникает, когда элемент теряет фокус, т.е. курсор покидает этот элемент. Таким образом мы можем отслеживать перемещение пользователя между полями.
- **#popup-audit** – это селектор формы с бесплатным аудитом
- **{{GenericEvent}}** – объект, который фиксирует различные значения нашего события

Далее создайте переменную **{{GenericEvent}}** со следующим кодом (представлен в главе **Прослушивание пользовательских событий**):

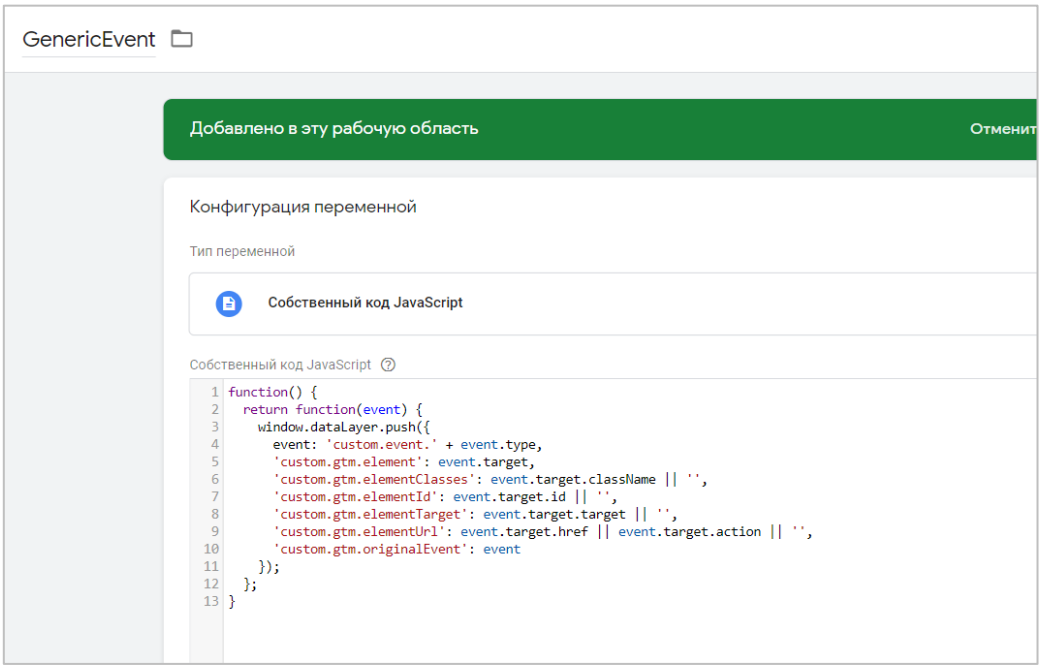


Рис. 859. Переменная Собственный код JavaScript

На следующем шаге добавьте триггер типа **Пользовательское событие** с именем **custom.event.blur**, поскольку выше сверху конструкций *'event': 'custom.event' + event.type*:

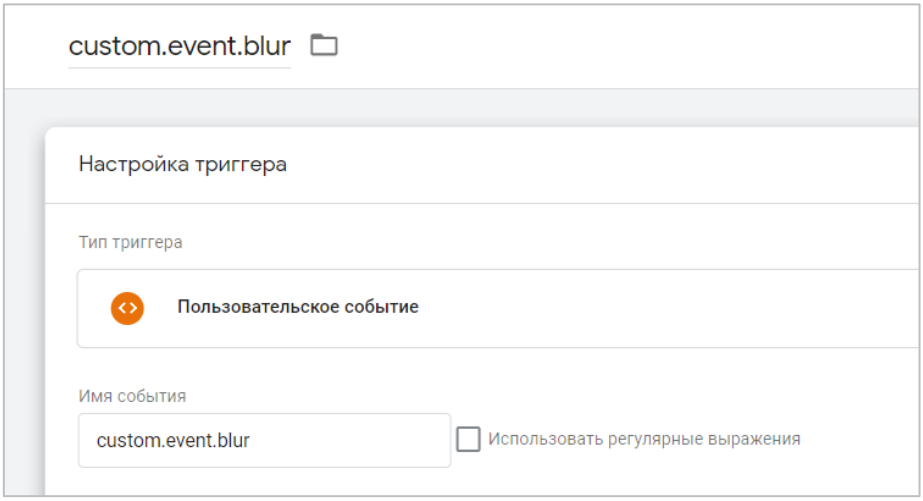


Рис. 860. Триггер Пользовательское событие

Еще нам нужна переменная уровня данных для создания шагов виртуальной страницы, которая извлекает Id элемента, на который пользователи кликают:

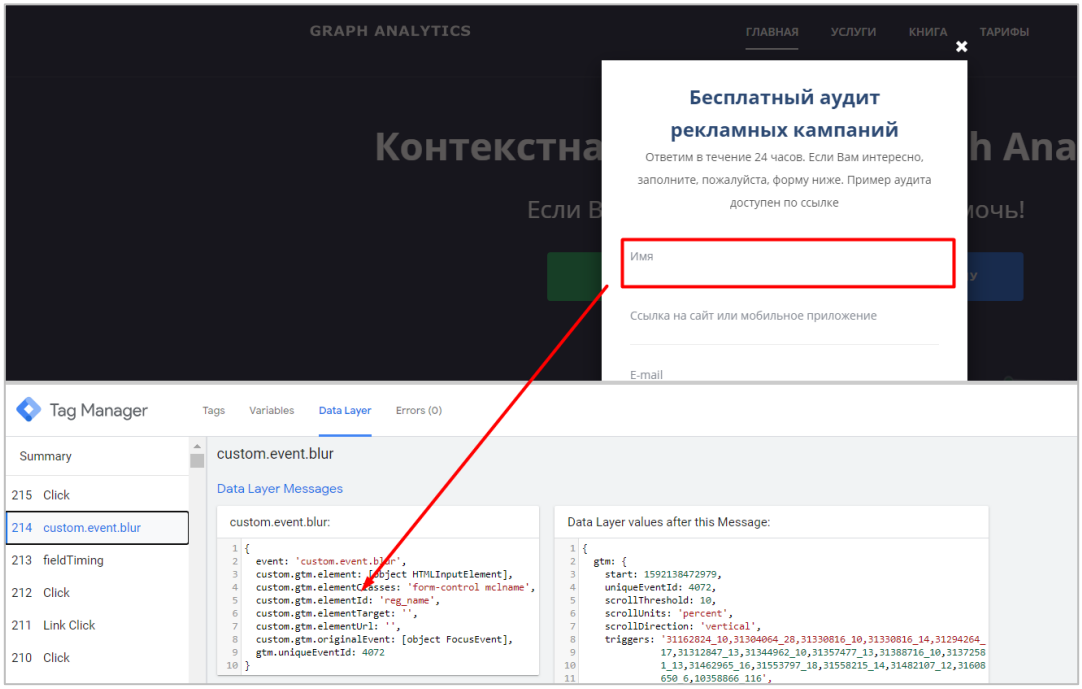


Рис. 861. Уровень данных события

Для всех это переменная уровня данных **custom.gtm.elementId**. При клике на поле **Имя** будет значение **reg_name**, на **Ссылка на сайт или мобильное приложение** - **reg_site** и т.д. Поэтому создаем ее:

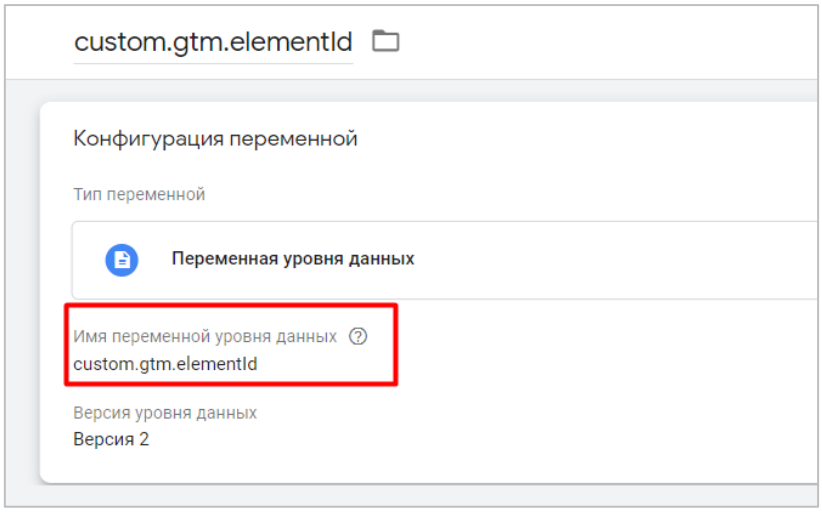


Рис. 862. Переменная уровня данных

Теперь осталось создать переменную типа **Таблица поиска**, и ввести значения для шагов нашей последовательности виртуальных страниц:

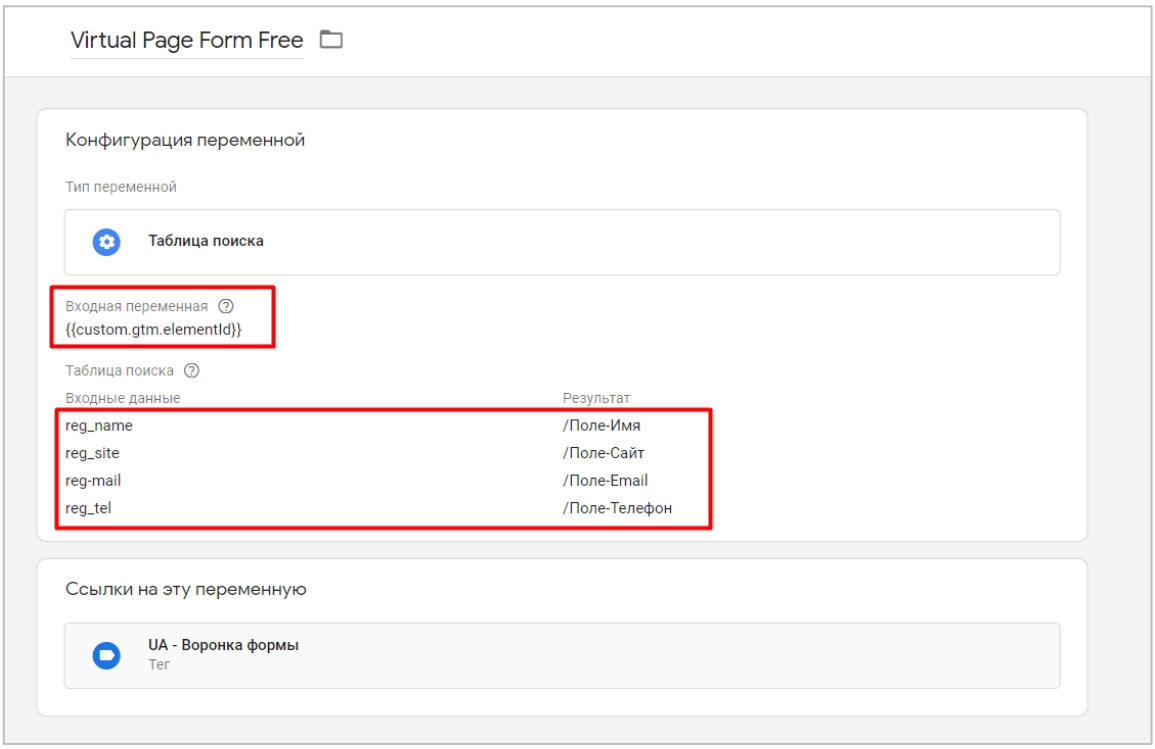


Рис. 863. Переменная Таблица поиска

Логика простая – если **custom.gtm.elementId** = reg_name, то будет значение **/Поле-Имя**, если **reg_site - /Поле-Сайт** и т.д. Их мы и будем передавать в теге как виртуальные страницы.

Теперь осталось добавить стандартный тег **Universal Analytics** с типом отслеживания **Просмотр страницы**, а в расширенных настройках добавить **Поля, которые необходимо задать - page** и нашу таблицу поиска:

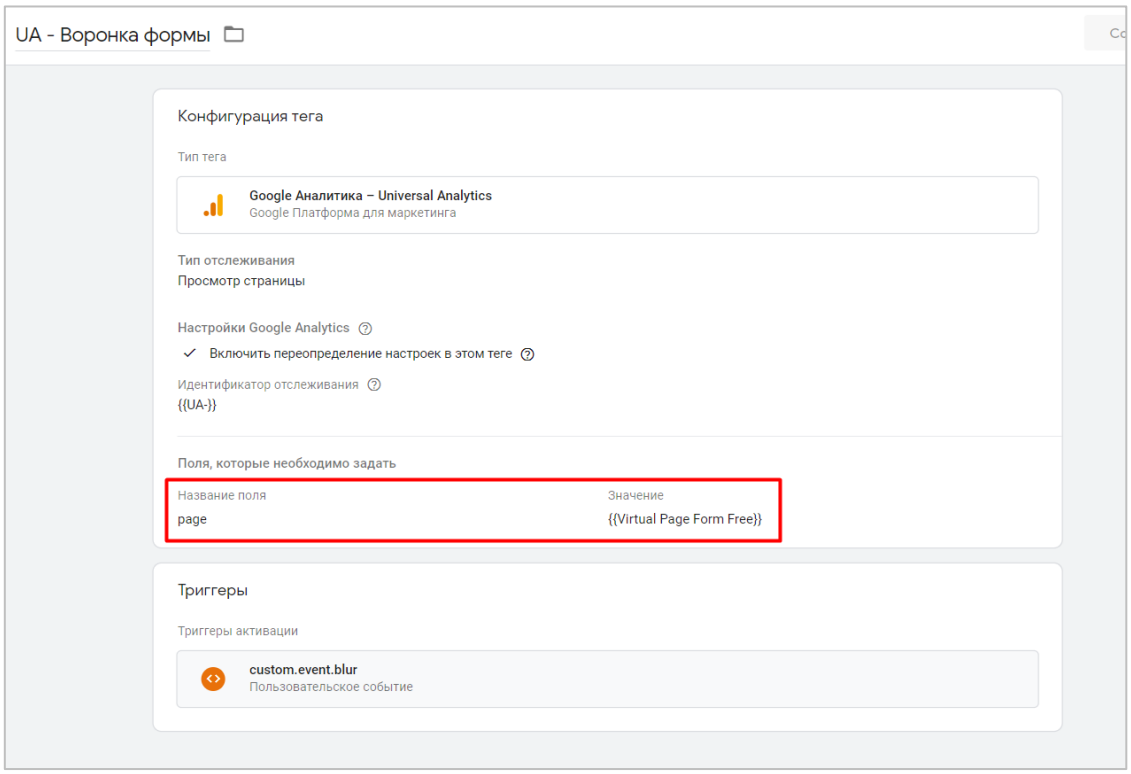


Рис. 864. Настройка тега Google Аналитика – Universal Analytics

Триггер активации – пользовательское событие **custom.event.blur**. После заполнения полей формы в режиме реального времени GA на вкладке **Контент**:

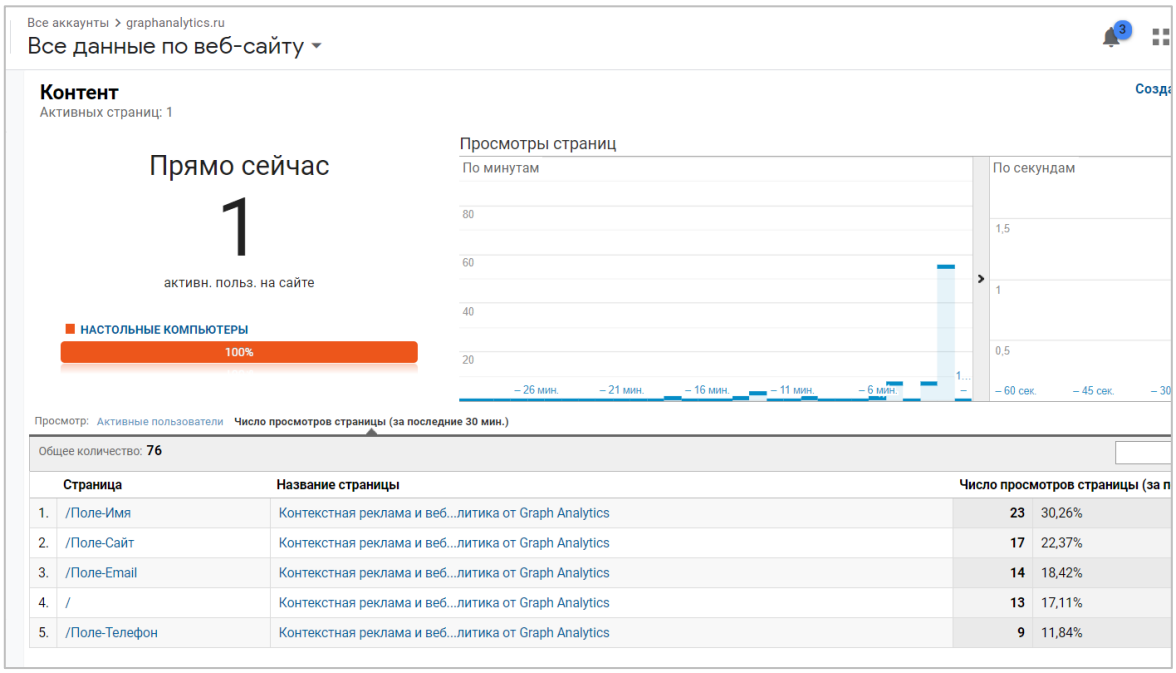


Рис. 865. Отчет В режиме реального времени

Чтобы построить визуализацию, необходимо создать цель с целевой страницей и последовательностью:

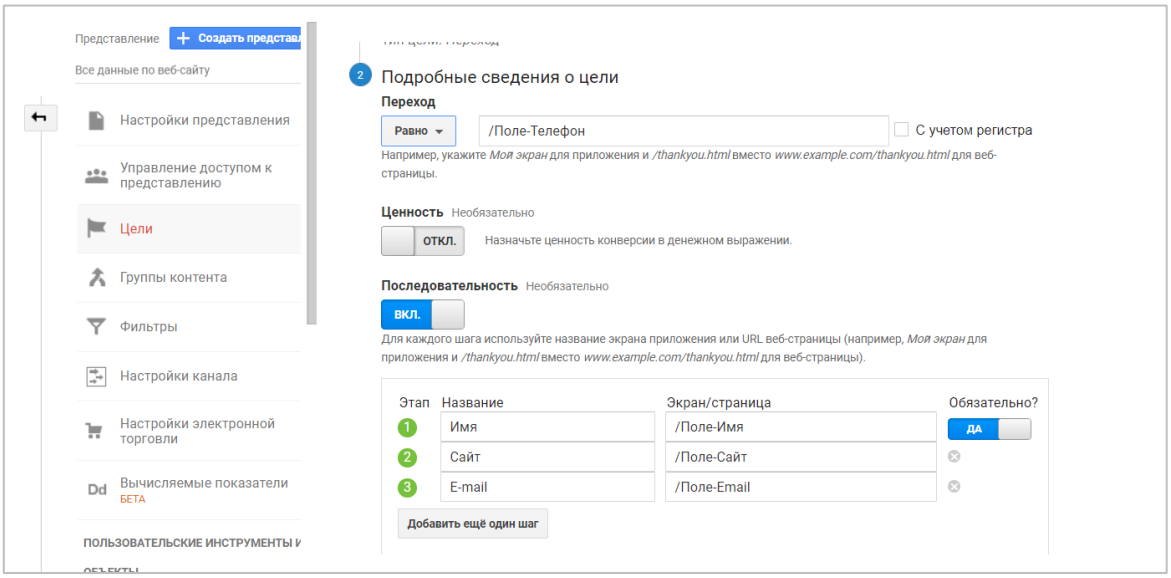


Рис. 866. Создание составной цели в Google Analytics

Последним шагом можно было добавить и чекбокс с галочкой условием пользовательского соглашения. Но в рамках примера не стал этого делать.

Наш итоговый отчет в Google Analytics: **Конверсии – Цели – Визуализация последовательности**.

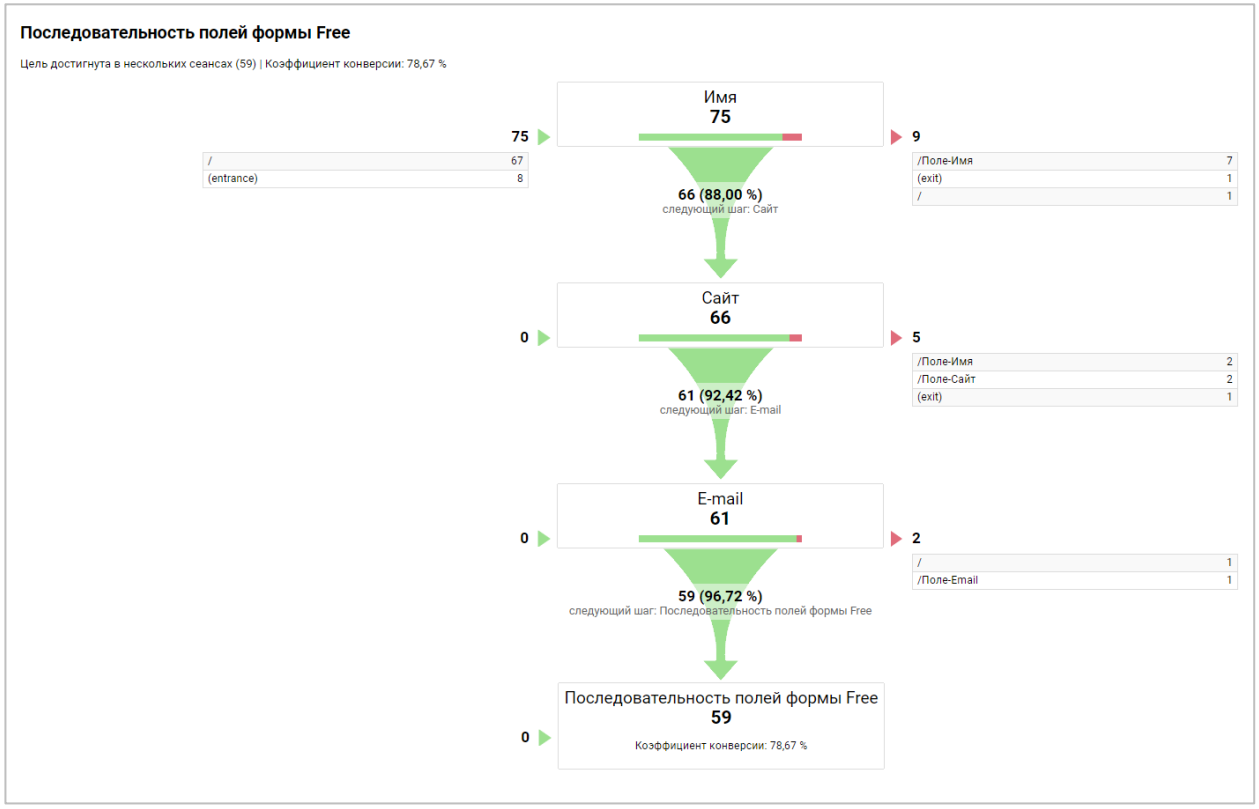


Рис. 867. Визуализация последовательности

Файлы cookie

Куки (Cookie) – небольшой фрагмент данных, который хранится на компьютере пользователя. Файлы cookie имеют множество особенностей и тонкостей в использовании. В этой статье подробнее остановимся на их применении в Google Tag Manager.

Что можно делать с помощью cookie?

Поскольку cookie — это просто текстовая информация с ограничением по размеру до 4Кбайт, то они сами по себе ничего делать не могут. Однако сервер может считывать содержащуюся в куки информацию и на ее основе совершать те или иные действия. Cookie применяются при сборе данных для инструментов веб-аналитики, персонализации рекламы и контента на сайте, для авторизации пользователей, для хранения информации о предпочтениях пользователей на сайте, местоположении, языке интерфейса и во многих других задачах.

Читайте вместе с этой публикацией (см. приложение):

- cookie файлы в Google Analytics;
- о том, какие файлы cookie использует Google;
- временные файлы, устанавливаемые Яндекс.Метрикой.

С точки зрения веб-аналитики основное назначение файлов cookie – идентификация пользователей с помощью уникального идентификатора **(Client ID, cid)**, который создается для каждого посетителя сайта. Client ID также можно использовать между различными сервисами (например, CRM-системой и счетчиком) в качестве ключа объединения при построении сквозной аналитики.

Существует два типа файлов cookie: *основные* и *сторонние*.

1. **основной файл cookie (first-party, 1st party)** – это файл, который создается одним доменом веб-сайта. Посетитель запрашивает его когда вводит URL в адресную строку браузера или выполняет переход по ссылке. Только этот сайт их может прочитать и определить, посещаете ли вы его не в первый раз. Это функция обеспечения безопасности встроена во все браузеры;

2. **сторонний файл cookie (third-party)** – это файл, который создается другими сайтами, размещающими свой контент.

В последние годы появилась тенденция к постепенной блокировке кук различными браузерами по умолчанию. Это затрудняет идентификацию пользователя в интернете. Например, разработчики браузера Firefox в 2019 году выпустили версию (69), в которой блокируется слежение за пользователями для показа навязчивой рекламы. Функция называется **Enhanced Tracking Protection**. Или Google, который 3 февраля 2020 года в своем блоге (см. приложение) анонсировал новые правила отслеживания действий пользователей. Браузер будет блокировать cookie-файлы, но только если у них не прописаны параметры для подтверждения безопасности. А к 2022 году команда Google Chrome планируют полностью запретить отслеживание действий пользователей по файлам cookie.

В конце концов, это приведет к тому, что рекламодатели не смогут персонализировать рекламу на различных площадках. Сейчас до конца не ясно, что будет являться альтернативой cookie в ближайшем будущем и какое следующее решение станет стандартом для отрасли. А пока разработчики полностью не запретили работу с cookie-файлами, ими можно и нужно пользоваться.

Благодаря связке cookie и Google Tag Manager вы сможете создавать более сложные триггеры и условия активации тегов. Например:

- пользователь просмотрел N страниц -> **показать баннер с уникальным предложением**;
- пользователь совершил M сеансов (визитов) на ваш сайт -> **предложить подписаться на рассылку**;
- пользователь был на сайте X раз с длительностью Y секунд -> **предложить пройти опрос**;
- пользователь был на сайте Z дней назад после первого посещения -> **сделать подарок, скидку;**
- пользователю был показан уникальный промокод, который теперь привязан конкретно к нему;
- пользователь не совершил транзакцию на сайте **-> показать баннер с акцией**, совершил -> **не показываем всплывающее окно**;
- пользователь перешел на сайт по партнерской ссылке, и значение идентификатора партнера сохранено в куки;
- не передавать дубли транзакций/конверсий пользователя в веб-аналитику;
- персонализировать контент на сайте для пользователя(ей);
- многое другое.

Вариантов настройки великое множество. Все ограничивается конкретными задачами.

Куки бывают:

- **постоянные cookie (persistent cookies)** – те, которые остаются доступными после закрытия браузера и повторного его открытия (например, если поставить галочку **«Запомнить меня»** при вводе логина и пароля);
- **сеансовые cookie (temporary cookies, session cookies)** – те, которые сохраняются только на протяжении посещения посетителя на сайте.

Над файлами cookie можно проводить различные операции - **установка, чтение, удаление**. Чтобы реализовать любой из сценариев, описанных выше, необходимо для начала установить файл cookie.

Установка cookie (Set-Cookie)

У куки есть ряд настроек, которые должны быть установлены. Они указываются после пары ключ=значение и отделены друг от друга ; (точкой с запятой). Куки можно создавать через JavaScript при помощи свойства **document.cookie**. Пример:

```
  document.cookie = "name=cookieName; path=/; domain=site.ru; expires=Tue, 19
Jan 2038 03:14:07 GMT; secure"
```

Каждая пара представляет собой отдельное куки:

- name – имя для cookie;
- cookieName – пример значения имени для cookie.

Не допускается использование двоеточия, запятой и пробела;

path - URL-префикс пути, по которому куки будут доступны для страниц.

Чаще всего указывают **path=/** , чтобы куки были доступны на всех страницах сайта. Если куки установлены с **path=/category**, то они будут доступны и на подкатегориях, например, **/category/products**, **/category/products/item** и т.д. По умолчанию значением является текущая директория, из которой куки устанавливаются в браузер.

- **domain** - домен, на котором доступны наши куки.

По умолчанию доступно лишь тому домену, который его установил. То есть куки, которые были установлены сайтом **site1.ru**, не будут доступны на сайте **site2.ru**. Опция **domain** позволяет использовать куки и для поддоменов;

- **site.ru** - значение домена для куки;
- **expires / max-age** - срок жизни куки.

Если не установлен ни один из этих параметров, то куки удаляются при закрытии браузера (сеансовые куки). Чтобы они стали постоянными (постоянные куки), остались доступными после закрытия браузера и повторного его открытия, необходимо установить значение опций **expires** или **max-age**.

expires

Дата истечения срока действия куки, когда браузер удалит его автоматически. При установке expires=Sun, 12 Apr 2020 13:41:56 GMT куки будут удалены **12 апреля 2020 года в 13:41:56** по Гринвичу.

Если мы установим в **expires** прошедшую дату, то срок действия куки истекает прямо сейчас.

Примечание: использование **expires** не гарантирует сохранность куки в течение заданного периода времени, поскольку клиент (браузер) может удалить запись из-за нехватки выделенного места или каких-либо других причин.

max-age

Альтернатива **expires**. Определяет срок действия куки в секундах с текущего момента. При записи max-age=7200 куки будут удалены через 2 часа. Если задан ноль или отрицательное значение, то срок действия куки истекает прямо сейчас.

- **Tue, 19 Jan 2038 03:14:07 GMT** - пример значения хранения куки (может быть другим).

Предпочитаемый формат согласно спецификации **RFC 7234** (см. приложение)**.**

- **secure** - параметр указывает браузеру, что куки должны передаваться на сервер только по защищенному https-соединению (HTTPS-протоколу).

По умолчанию куки, установленные сайтом **http://site.ru**, также будут доступны на сайте **https://site.ru** и наоборот. Они опираются на доменное имя и не обращают внимания на протоколы. Но если для сайта **https://site.ru** будут установлены cookie со значением **secure**, то они не будут доступны на том же сайте **http://site.ru** с протоколом HTTP.

Примечание: начиная с Chrome 52 и Firefox 52, незащищенные сайты (http:) не могут создавать куки с флагом **secure**.

Подробнее про куки и document.cookie читайте на **learn.javascript.ru** (см. приложение).

Мы с вами разобрали основные атрибуты установки куки. Чтобы установить куки в Google Tag Manager, нужно создать тег типа **Пользовательский HTML**. Он будет содержат код Javascript, который определяет имя файла cookie, срок жизни и т.д.

```
  <script>
  var cookieName  = "myCookie"; // название куки
  var cookieValue = "cookieValue"; // значение имени для cookie
  var cookiePath  = "/"; // URL-префикс пути, по которому куки будут доступны
для страниц
  var expirationTime = 2592000; // 1 месяц в секундах
```

```
  expirationTime = expirationTime * 1000;  // конвертация секунд в миллисекунды
  var date = new Date(); // создание объекта Date
  var dateTimeNow = date.getTime(); // получение текущего времени в
миллисекундах с 1 января 1970 года (Unix)
  date.setTime(dateTimeNow + expirationTime); // установка срока жизни куки
(текущее время + 1 месяц)
  var expirationTime = date.toUTCString(); // преобразование миллисекунд в
строку времени (формат UTC)
  document.cookie = cookieName+"="+cookieValue+"; expires="+expirationTime+";
path="+cookiePath; // установка cookie
  </script>
```

В этом коде можно отредактировать:

- **cookieName** - имя для куки. Придумывайте такое название, чтобы сразу было понятно, за что эта кука отвечает;
- **cookieValue** - значение для куки. Может быть числом, текстом, логическом типом (true или false);
- **cookiePath** - путь, внутри которого будет доступ к куки. Значение / устанавливает куку на всех страницах сайта;
- **expirationTime** - срок жизни куки. В примере выше - 30 дней в секундах (60*60*24*30).

В Google Tag Manager это выглядит так:

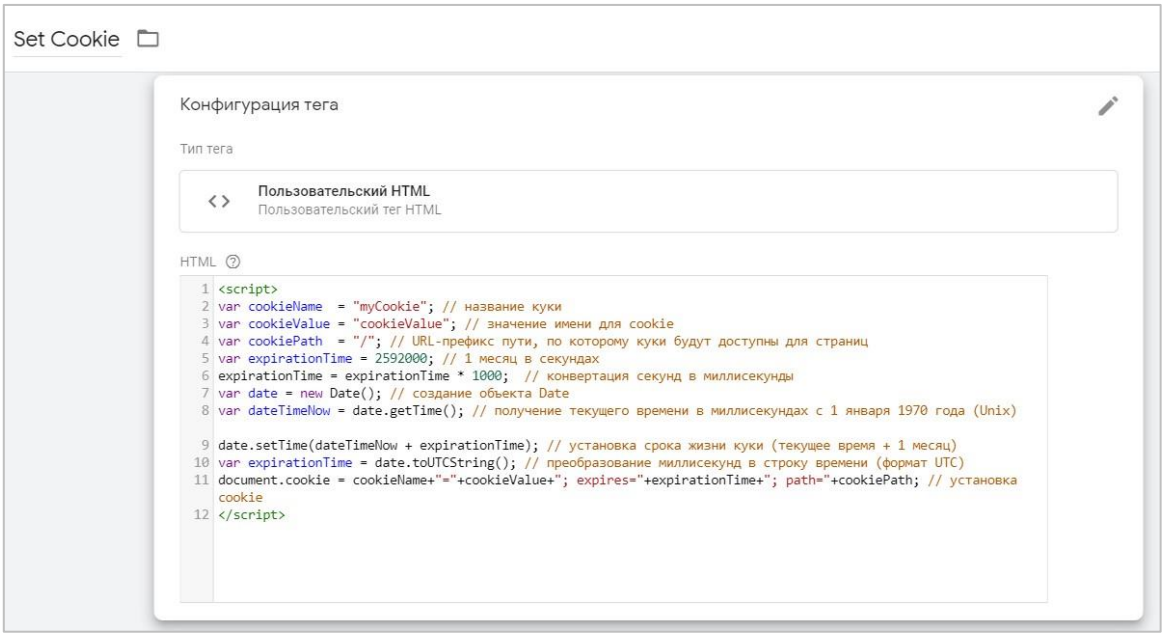


Рис. 868. Установка cookie в диспетчере тегов

Сохраните настройки тега. Теперь можно проверить установку куки с помощью консоли разработчика. Зайдите на ваш сайт, нажмите клавишу F12 (для Google Chrome). В открывшемся окне перейдите на вкладку **Application** и выберете в меню **Cookies** и домен вашего сайта:

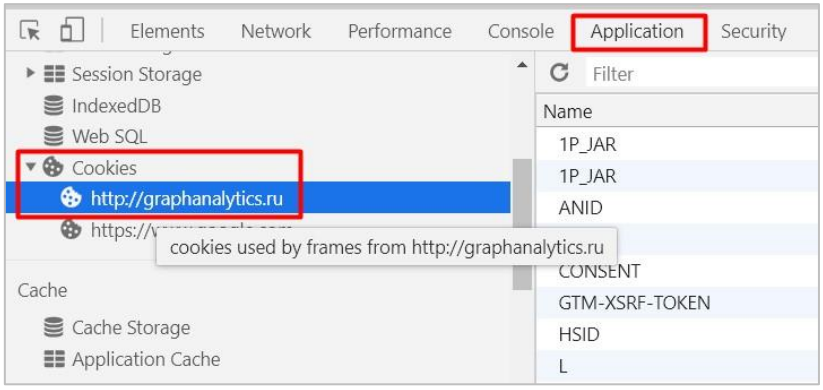


Рис. 869. Application - Cookies - Сайт

Справа появится таблица, в которой отобразятся все ваши куки (основные и сторонних сервисов). Воспользовавшись фильтром, мы сможем найти установленную на предыдущем шаге куку. В моем примере — это **myCookie** со значением **cookieValue** и сроком жизни 30 дней:

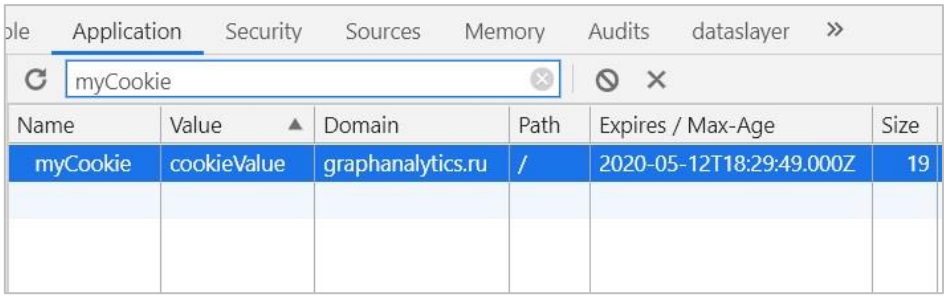


Рис. 870. Установленная с помощью GTM кука

Чтение cookie (Get Cookie)

Теперь нам необходимо научиться читать куки с помощью Google Tag Manager и извлекать их значения в переменные. Делается это с помощью пользовательской переменной типа **Основной файл cookie (1st Party Cookie)**. Все, что необходимо указать, это название cookie (name). Для моего примера переменная будет выглядеть так:

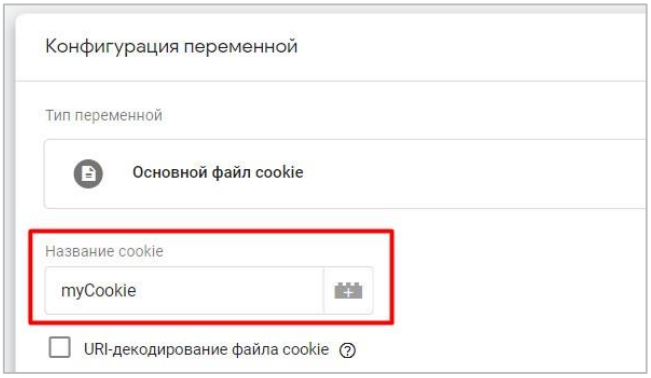


Рис. 871. Переменная Основной файл cookie

Проверить корректность извлечения значения из файла cookie с именем **myCookie** можно с помощью режима предварительного просмотра. Если все сделали правильно и не ошиблись в названии куки, то на вкладке **Variables** вы увидите переменную с установленным значением:

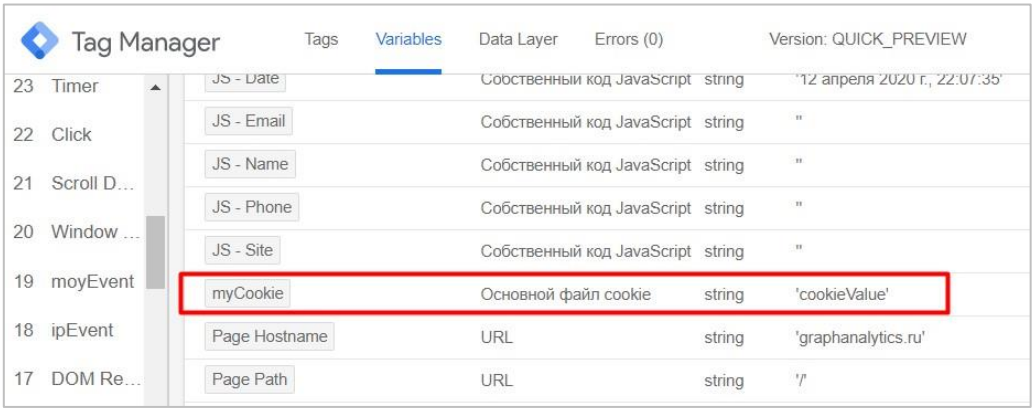


Рис. 872. Значение cookieValue в переменной myCookie

Аналогично можно извлекать значения любых других кук нашего домена. Например, **_ga** от Google Analytics или **_ym_uid** от Яндекс.Метрики:

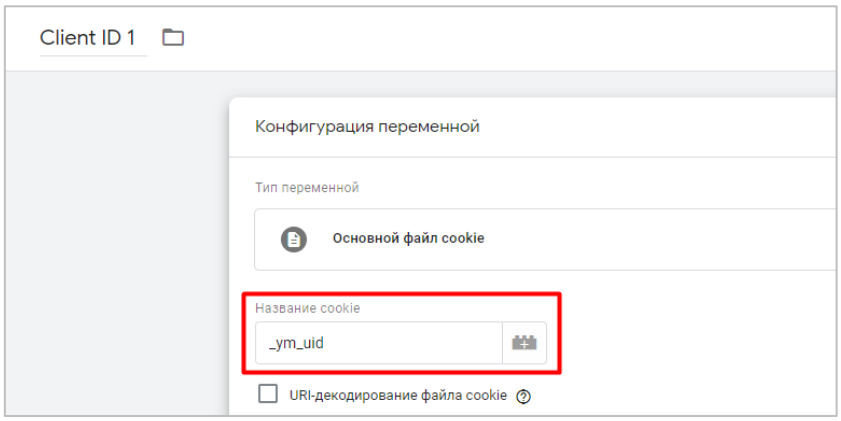


Рис. 873. Название cookie _ym_uid (Яндекс.Метрика)

Это были простые примеры чтения кук. Но вы можете использовать установку и чтение cookie для создания более сложных условий активации, которые затем будете использовать в триггерах и тегах GTM.

Примечание: не все основные файлы cookie (1st Party Cookie) можно извлекать. Например, перейдя на всю ту же вкладку **Application** в консоли разработчика и выбрав ваш домен в меню **Cookies**, вы увидите все файлы куки, которые вы можете использовать в Google Tag Manager. Различные скрипты и другие функции на вашем сайте устанавливают эти куки для браузеров ваших посетителей. Однако не все из них могут быть доступны с помощью JavaScript в вашем браузере (включая GTM).

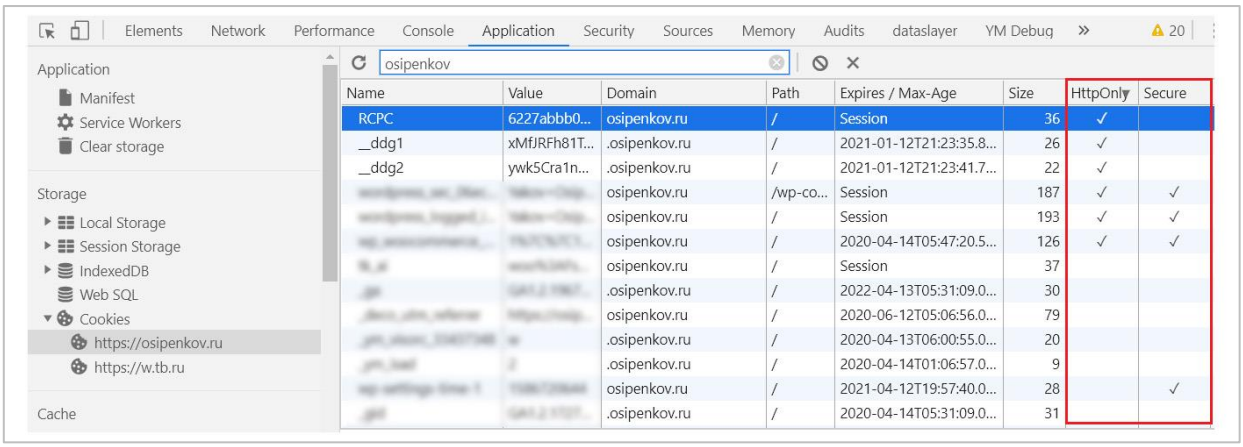


Рис. 874. Куки с галочкой HttpOnly и Secure

Если куки имеет флажок в столбцах **HttpOnly** или **Secure**, GTM не сможет получить доступ к его значению. Это означает, что если вы добавите название cookie в пользовательскую переменную, то GTM вернет **undefined**:

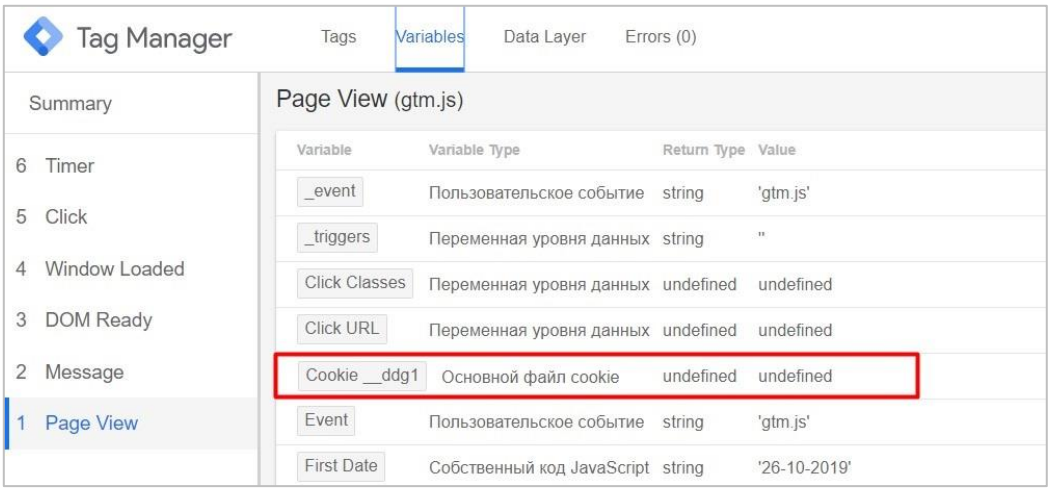


Рис. 875. undefined для cookie __ddg1

Удаление cookie (Delete Cookie)

Если вы хотите удалить определенный файл куки, то вам нужно установить срок его жизни за прошедшую дату, например, 1 января 2019 года. Такие куки не могут существовать, поэтому срок их действия немедленно истекает.

Самое простое решение:

```
<script>
document.cookie = "myCookie=; expires=Thu, 01 Jan 1970 00:00:00 UTC; path=/;";
</script>
```

Скрипт, который можно использовать:

```
<script>
function delete_cookie(name) {
var expires = new Date(0).toUTCString();
var domain = location.hostname.replace(/^www\./i, "");
document.cookie = name + "=; expires=" + expires + "; path=/; domain=." +
domain;
}
delete_cookie("myCookie"); // название куки, который вы хотите удалить
</script>
```

Замените **myCookie** на имя куки, который вы хотите удалить. Этот код также должен быть реализован в Google Tag Manager с помощью тега типа **Пользовательский HTML** и запускаться каждый раз, когда вам необходимо удалить куки.

В Google Tag Manager это выглядит так:

Рис. 876. Удаление куки

Таким образом, вы можете сохранять необходимую информацию в куки конкретных пользователей и с помощью триггеров активировать теги только для той аудитории, которая попадает под определенные критерии. А чтобы найти идеи того, какие данные можно сохранять в куки, просто зайдите на сайты крупных компаний или своих конкурентов, откройте консоль разработчика, перейдите на вкладку **Application - Cookies** и посмотрите на перечень установленных куки. Например, **mvideo.ru** в куки передает тип устройства, **ozon.ru** - значение User ID, а **beru.ru** (маркетплейс от Сбербанка и Яндекса) синхронизирует ваши данные с Яндекс.Паспорт и другими сервисами Яндекса:

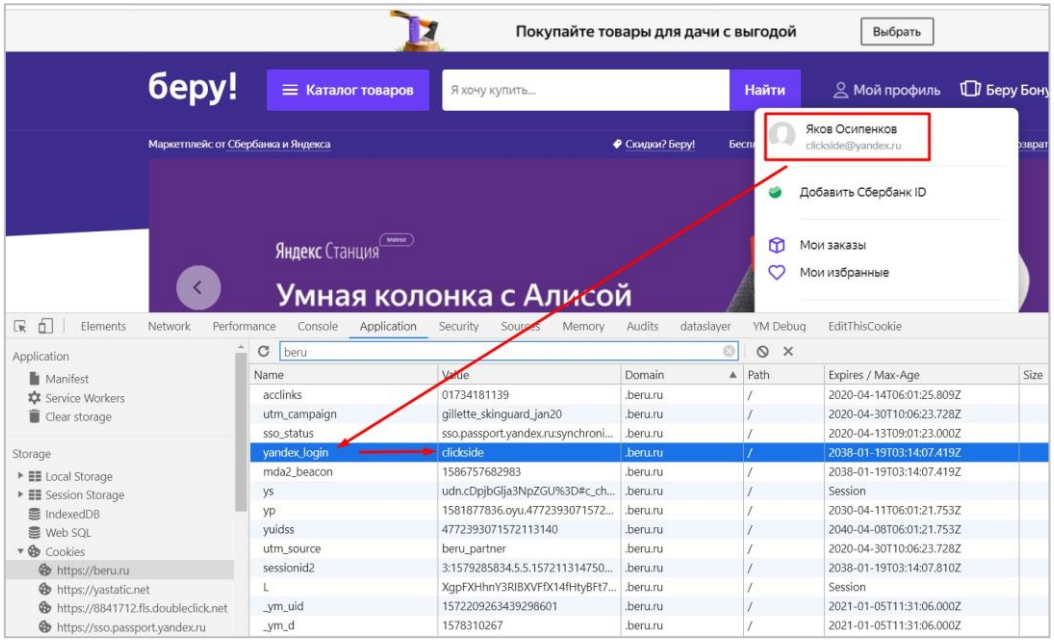


Рис. 877. Синхронизация с Yandex

Таким образом, компания знает все ваши данные, которые вы добавили в учетную запись Яндекса, включая логин почты, контактный телефон, пол, возраст. Часть из них на beru.ru автоматически подставляется на странице оформления заказа при заполнении данных.

В качестве альтернативы консоли разработчика и вкладки **Application** для просмотра кук можно использовать расширение для браузера Google Chrome - **EditThisCookie**. С его помощью вы можете добавлять, удалять, изменять, искать, защищать и блокировать cookie. Просто установите его, перейдите на нужную страницу на сайте и откройте расширение со значком печенья.

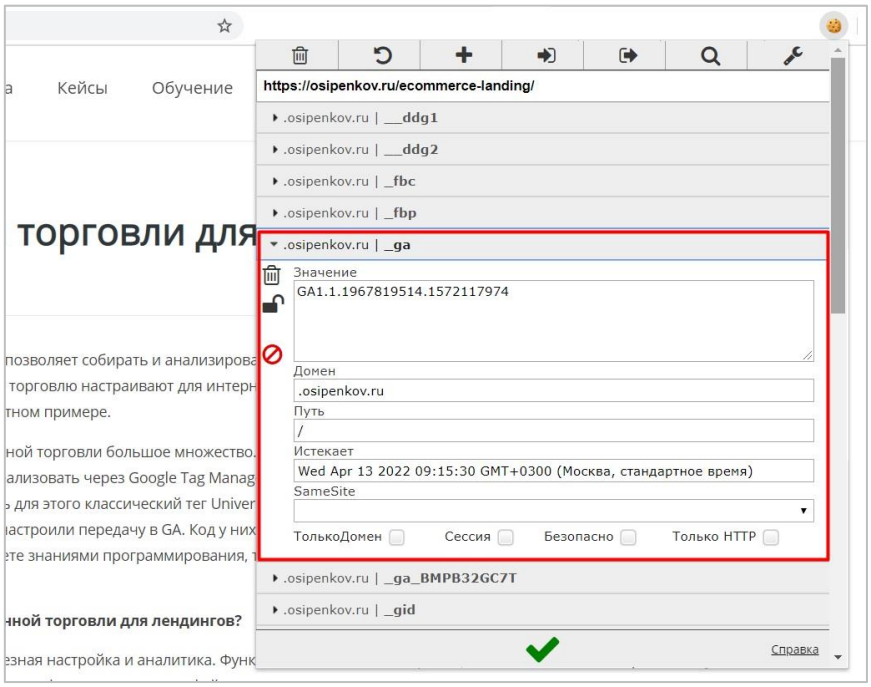


Рис. 878. Отображение кук в EditThisCookie

В нем вы увидите список всех доступных куки на текущей странице. Развернув нужную, вы сможете просмотреть данные о ней - значение, какому домену принадлежит, путь, срок жизни, есть ли параметр secure и т.д.

В связи с некоторыми ограничениями, с которыми веб-аналитики сталкиваются при работе с cookie, другим решением является использование локальных хранилищ – **sessionStorage** и **localStorage**.

sessionStorage и localStorage

Самая большая проблема при использовании cookie в качестве локального хранилища заключается в том, что:

- в каждом из них можно храниться максимум 4 Кбайт данных (этого может быть недостаточно для работы веб-приложения);
- cookie должны отправляться в обоих направлениях при каждой перезагрузке страницы, что может существенно замедлять работу веб-сайта;
- куки также включаются в каждый HTTP-запрос при передаче данных через Интернет в незашифрованном виде.

С появлением HTML5 мы получили доступ к более объемному веб-хранилищу (между 5 и 10 МБайт на каждый домен), которое сохраняет информацию между загрузками страницы и посещениями сайта (даже после выключения и включения компьютера). Кроме того, данные локального хранилища не отправляются на сервер при каждой загрузке страницы, в связи с чем ускоряется работа веб-приложения.

Веб-хранилище (Web Storage, DOM-хранилище) — это набор механизмов, которые используются для хранения данных локально в веб-браузере пользователя в виде пары ключ/значение. Другими словами, это специально отведенное место в браузере (похоже на небольшую базу данных), в котором мы можем работать с данными - записывать, читать и удалять их.

Существуют два основных типа веб-хранилища:

1. сессионное хранилище (sessionStorage);
2. локальное хранилище (localStorage);

Оба типа являются свойствами глобального объекта браузера (window). К ним можно обращаться как **window.sessionStorage** (или **sessionStorage**) и **window.localStorage** (или **localStorage**) соответственно. В них можно хранить только строковые данные для ключей и их значений.

sessionStorage (сессионное хранилище, хранилище сессии)

Сессионное хранилище — это объект **sessionStorage**, который позволяет хранить данные для каждого домена, доступные на протяжении сессии, пока открыт браузер, и даже в том случае, если была осуществлена перезагрузка страницы. Однако закрытие вкладки браузера, самого браузера или перезагрузка компьютера приводит к удалению данных из sessionStorage.

Объект sessionStorage используется гораздо реже, чем localStorage.

localStorage (локальное хранилище)

Локальное хранилище — это объект **localStorage**, который позволяет хранить данные для каждого домена в течение неограниченного времени (после закрытия вкладки браузера, самого браузера, перезагрузки компьютера), а точнее пока пользователь самостоятельно не удалит данные.

Каждый домен имеет доступ к своему хранилищу данных localStorage. Например, localStorage, используемый для, **https://osipenkov.ru** является отдельным от localStorage, используемым для **https://coobiq.com**. Субдомены (поддомены) и различные протоколы HTTP (HTTP и HTTPS) имеют независимые друг от друга хранилища данных. Например, localStorage **https://gtm.osipenkov.ru** используется полностью отдельно от **https://osipenkov.ru**. Точно так же localStorage **https:**//osipenkov.ru используется отдельно от **http:**//osipenkov.ru.

Некоторые браузеры блокируют localStorage в режиме инкогнито. localStorage работает даже с отключенными cookie.

Примечание: не храните в localStorage конфиденциальную информацию и личные данные пользователей, включая имя, фамилию, дату рождения, пароли, номера кредитных карт, номер телефона, e-mail и многое другое.

Что можно хранить и для чего использовать?

Хранение данных в локальном хранилище очень распространено. Классическим примером использования локального хранилища является веб-приложение типа **Ежедневник**, когда пользователь составляет список дел на день, и по мере выполнения удаляет их из списка. Для выполнения этой задачи не нужен сервер, подойдет localStorage. Кликнув по задаче из списка, которое человек уже сделал, она будет удаляться с локального хранилища и, следовательно, показываться пользователю больше не будет. Локальное хранилище часто используется для сохранения настроек пользователя на сайте (выбор темы оформления, вид отображения информации), чтобы при следующем заходе эти данные уже были применены к его профилю и повторно не вводились.

Поскольку мы работаем с данными, то можем сохранять в sessionStorage и localStorage все, что связано с идентификацией пользователя - User-Agent, местоположение, IP-адрес, уникальный идентификатор пользователя (Client ID), значение сгенерированного промокода от виджета на сайте, источник перехода, список товаров, добавленных в корзину, пользовательские данные при неудачной попытке отправки на сервер и другие параметры.

Посмотреть, что из себя представляют sessionStorage и localStorage, какие данные там хранятся у разных сайтов, можно в консоли разработчика на вкладе **Application** (для браузера Google Chrome).

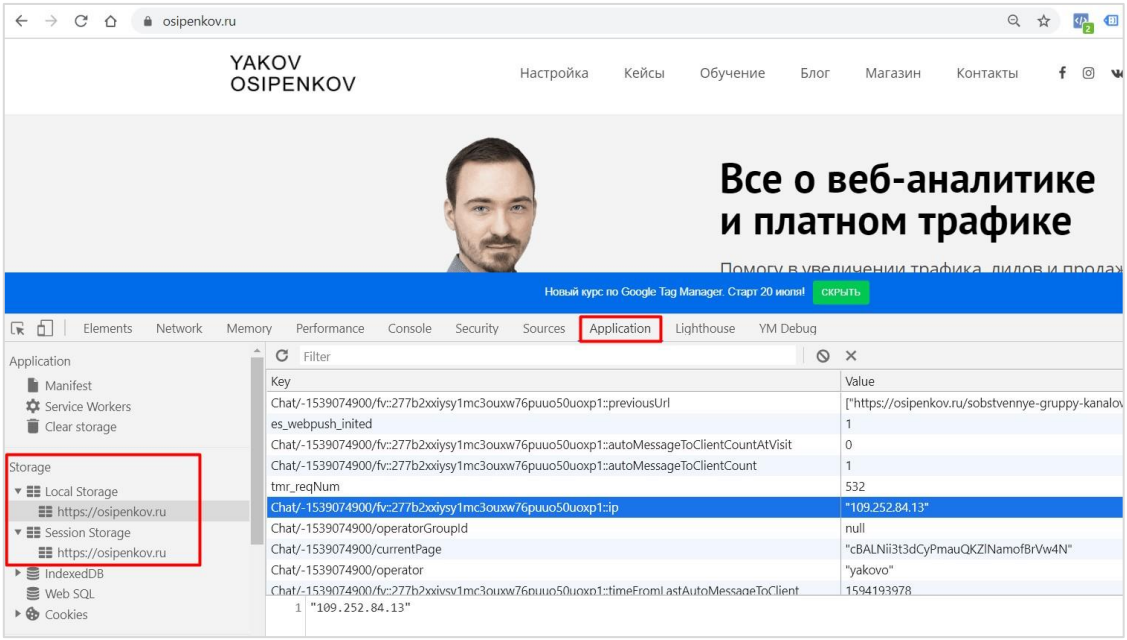


Рис. 879. Веб-хранилище (вкладка Application)

Либо использовать команды sessionStorage и localStorage на вкладке **Console**:

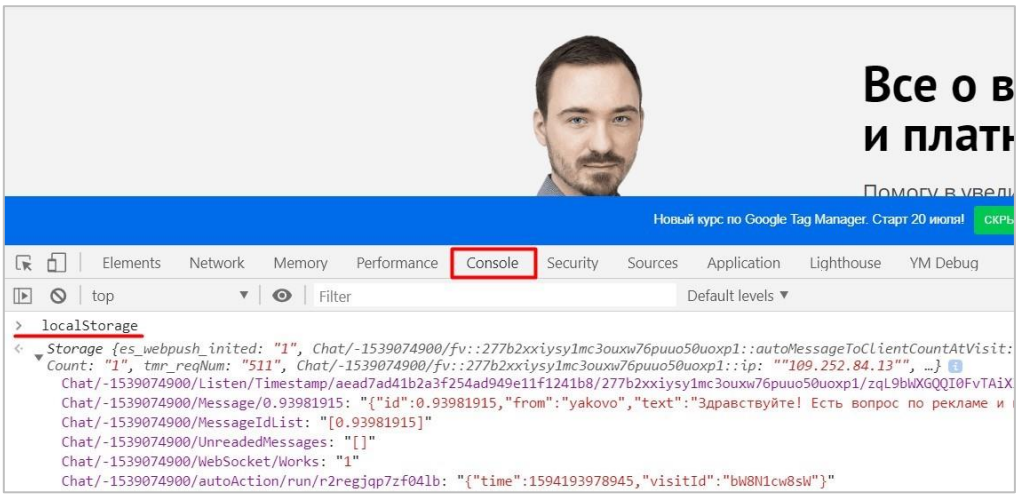


Рис. 880. Команда sessionStorage или localStorage (вкладка Console)

Как видим из примера выше, на моем сайте в localStorage хранится достаточно много данных. Большая часть из них — это установленные ключ/значение от онлайн консультанта Talk-Me. Сервис записывает в локальное хранилище данные о приветственном сообщении, времени совершения события, регионе, IP-адресе пользователя, источнике перехода, о последней просмотренной странице и многом другом. Вся эта информация по каждому пользователю потом предоставляется в виде краткого набора данных. Например, вот так выглядит информация по одному из чатов в мобильном приложении Talk-Me:

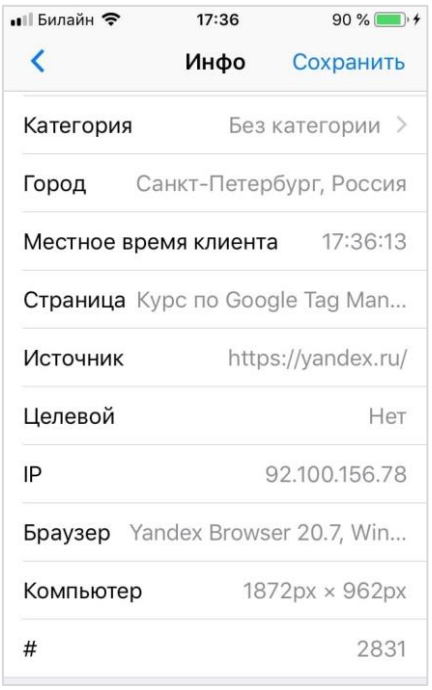


Рис. 881. Карточка пользователя по одному из чатов

sessionStorage vs localStorage vs cookies

Нагляднее всего продемонстрировать отличия между sessionStorage, localStorage и cookies с помощью таблицы:

	Cookies	sessionStorage	localStorage
Размер	4 Кбайт	5 Мбайт	до 10 Мбайт
Поддержка браузера	HTML4/HTML5	HTML5	HTML5
Отправка при запросе	Да	Нет	Нет
Срок жизни	Настраивается вручную	При закрытии вкладки браузера	Не удаляется
Место хранения	Браузер и сервер	Только в браузере	Только в браузере
Доступность	Из любого окна	Из того же окна	Из любого окна

- **Размер:** куки ограничены 4 Кбайт, sessionStorage - 5 МБайт, а localStorage - от 5 до 10 Мбайт;
- **Поддержка браузера:** cookies были еще в стандарте HTML4, в то время как sessionStorage и localStorage появились в HTML5;
- **Отправка при запросе:** можно передать куки при отправке запроса, а sessionStorage и localStorage нет;
- **Срок жизни:** для файлов cookie мы можем задать период жизни, данные в sessionStorage удаляются после закрытия вкладки браузера, а в localStorage они не удаляются "никогда";
- **Место хранения:** куки хранятся и в браузере, и на сервере, а sessionStorage и localStorage только локально в браузере пользователя;
- **Доступность:** данные о пользователе в cookies и localStorage доступны из любого окна, при условии, что вы осуществляется заход под теми же данными (например, авторизуетесь на сайте), в то время как sessionStorage работает только из того же окна, а изменение вкладки браузера приведет к удалению информации.

Помимо этого, согласно правилам обработки персональных данных, установленных Генеральным регламентом ЕС о защите персональных данных (или **GDPR — General Data Protection Regulation**), владелец сайта должен получать разрешение от пользователей на использование файлов cookie. Для локального хранилища этого не требуется.

Как вы уже знаете, сейчас в части браузеров используется технология "умной защиты от слежения", которая предоставляет отчеты о заблокированных трекерах (в том числе и счетчиках веб-аналитики), тем самым препятствуется слежение за пользователями и показ им персонализированной рекламы с помощью файлов cookie. Хоть мы и можем устанавливать срок жизни куки, их время все равно зависит от настроек самих браузеров. Например, система интеллектуального отслеживания **(Intelligent Tracking Prevention, ITP 2.2)** Apple ограничивает в Safari использование основных файлов cookie (first-party) до 1 дня. В результате файлы cookie, установленные рекламными сервисами, например, Facebook, Google или Яндексом, для измерения трафика сайта и атрибуции рекламы, будут удалены через 24 часа. И если человек нажимает на рекламное объявление в пятницу, а затем решает отложить покупку в выходные, то в понедельник у нас уже не будет cookie, чтобы показать ему рекламу и напомнить о приобретении. Остается только надеется на то, что человек запомнил наш сайт и вернется сам.

Чтобы отслеживать пользователей больше этого времени, можно использовать localStorage. Но и тут разочарование относительно пользователей Safari. В последних обновлениях разработчики добавили ограничение в 7 дней, то есть срок жизни данных в локальном хранилище теперь ограничен. Когда пользователь просматривает сайт через Safari, счетчик становится равен 7 дням. В течение этого времени он должен повторно вернуться на сайт, чтобы мы смогли использовать информацию из локального хранилища. В противном случае данные по истечении этого времени будут удалены из localStorage.

Так или иначе, sessionStorage и localStorage активно используются разработчиками при разработке веб-приложений, а интернет-маркетологами как альтернатива файлам cookie. В блоге Симо Ахавы есть статья (см. приложение), которая посвящена хранению Client ID в localStorage для Google Analytics.

Методы и свойства

Объекты хранилища sessionStorage и localStorage предоставляют одинаковые методы и свойства:

- **setItem(key, value)** – сохранить пару ключ/значение;
- **getItem(key)** – получить данные по ключу key;
- **removeItem(key)** – удалить данные с ключом key;
- **clear()** – удалить все;
- **key(index)** – получить ключ на заданной позиции;
- **length** – количество элементов в хранилище.

Запись данных в хранилище

Функция **setItem(key, value)** принимает ключ в качестве первого аргумента и значение в качестве второго аргумента. Как упоминалось ранее, все данные должны быть строками. Примеры установки:

```
// ключ "google", а значение "Analytics"
sessionStorage.setItem("google", "Analytics");
// ключ "google", а значение "Tag Manager"
localStorage.setItem ("google", "Tag Manager");
```

Попробовать сохранить пару ключ/значение можно в консоли разработчика на вкладке **Console**:

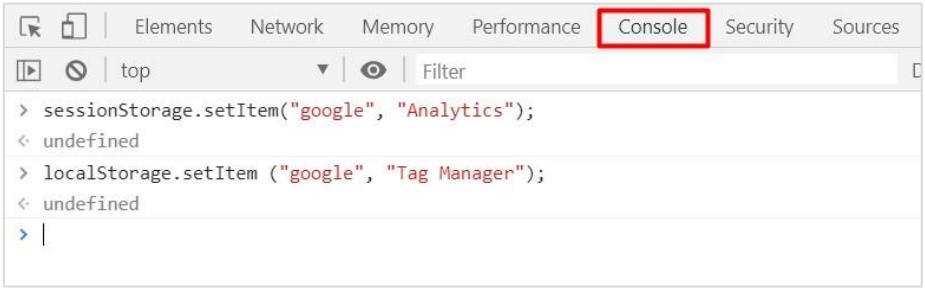


Рис. 882. Сохранение ключа и значения в sessionStorage и localStorage

Перейдя на вкладку **Application**, мы увидим записанные в хранилище данные:

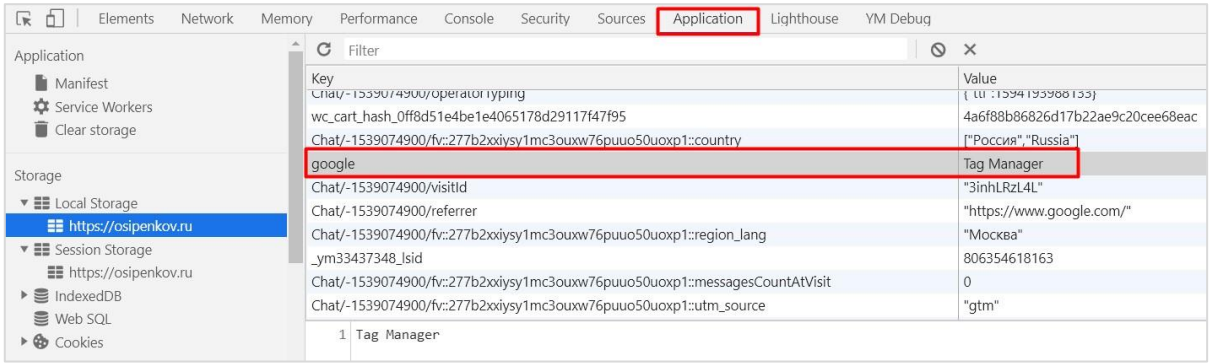


Рис. 883. Вкладка Application с установленными значениями sessionStorage и localStorage

Если мы закроем и откроем вкладку вновь, то данные в localStorage сохранятся, а в sessionStorage удалятся.

Записать данные в хранилище можно несколькими способами:

```
localStorage.setItem("key", "value");
localStorage.key = "value";
localStorage["key"] = "value";
```

Аналогично и для sessionStorage.

Как правило, интернет-маркетологи самостоятельно не записывают данные в sessionStorage или localStorage с помощью setItem. Это задача больше относится к разработчикам. Мы чаще извлекаем какие-то данные из переменных при помощи getItem, используя диспетчер тегов Google. Но знать о том, как сохранять (записывать) пару ключ/значение будет полезным.

Получение данных из хранилища

Функция **getItem(key)** берет ключ, который использовался при сохранении данных в качестве аргумента. Чтобы извлечь значение из ключей, которые мы сохранили, необходимо написать следующее:

```
// задается ключ "google", а отобразится значение "Analytics"
sessionStorage.getItem("google");
// задается ключ "google", а отобразится значение "Tag Manager"
localStorage.getItem ("google");
```

В консоли разработчика получим:

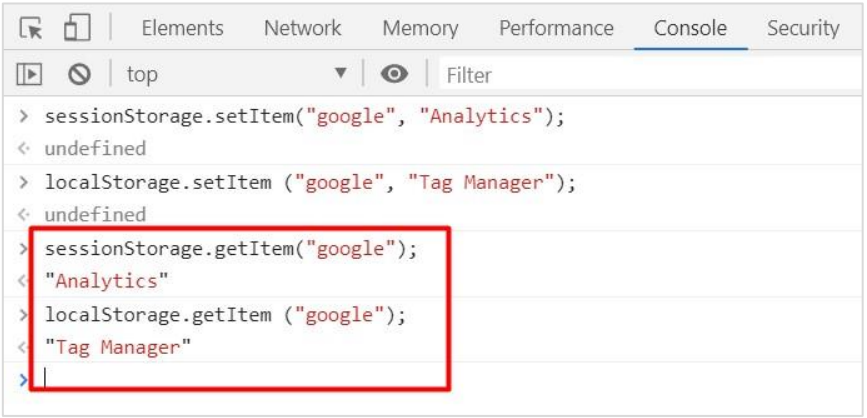


Рис. 884. *Извлечение данных из sessionStorage и localStorage*

Считать данные из хранилища также можно несколькими способами:

```
localStorage.getItem("key");
localStorage.key;
```

```
localStorage["key"];
```

Аналогично и для sessionStorage.

Чтобы извлечь данные из sessionStorage и localStorage с помощью Google Tag Manager, необходимо создать пользовательскую переменную типа **Собственный код JavaScript** со следующим кодом для sessionStorage:

```
function() {
  var sStorage = sessionStorage.getItem('key');
  return sStorage;
}
```

и для localStorage:

```
function() {
  var lStorage = localStorage.getItem('key');
  return lStorage;
}
```

, где **key** - ключ, который использовался при сохранении данных в качестве аргумента. Как я уже говорил, он чаще задается разработчиками, а интернет-маркетолог лишь извлекает данные из хранилища.

Пример переменной в Google Tag Manager:

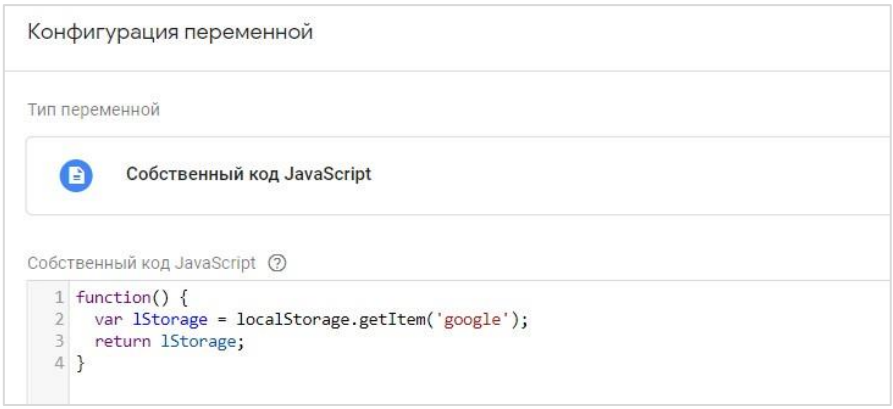


Рис. 885. Пример переменной localStorage

Если все сделано правильно, в режиме предварительного просмотра на вкладке **Variables** можно увидеть значение нашего ключа "google" в формате строки (string):

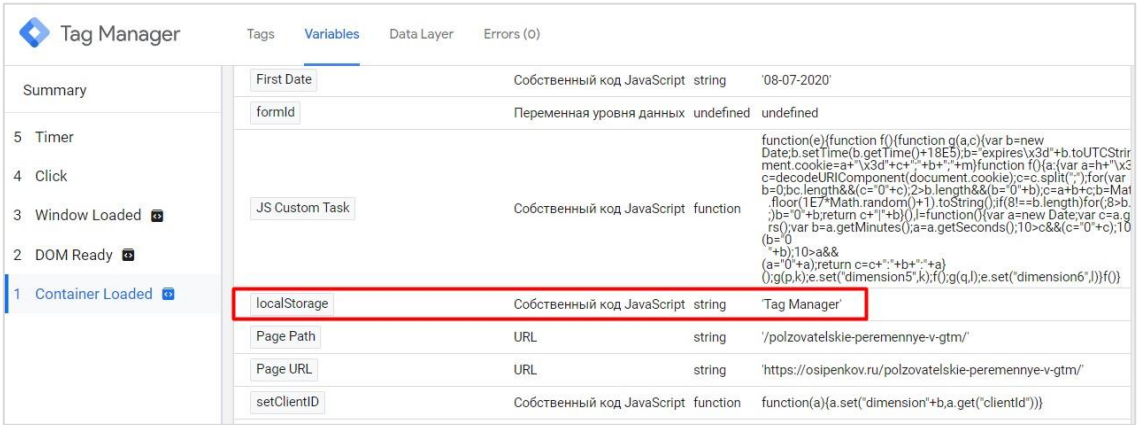


Рис. 886. Значение localStorage в режиме отладки

Удаление данных из хранилища

Имеет всего один синтаксис **removeItem(key)**:

```
localStorage.removeItem("key");
sessionStorage.revomeItem("key");
```

Удаление всех данных из хранилища

В любой момент вы можете удалить все записи командой **.clear()**:

```
localStorage.clear();
sessionStorage.clear();
```

Количество записей в хранилище

Для того, чтобы получить общее количество записей в хранилище, необходимо использовать стандартное свойство **length:**

```
localStorage.length
sessionStorage.length
```

Получение всех записей хранилища

Получить все данные, записанные в хранилище, можно с помощью цикла:

```
for (var i = 0; i < localStorage.length; i++) {
  var key = localStorage.key(i);
  console.log(key + ' = ' + localStorage[key]);
}
```

Аналогично для sessionStorage, только в коде нужно заменить local на session.

Хранение объектов

Как мы уже знаем, в localStorage для ключей и значений можно использовать только строковый тип данных. Если мы попытаемся сохранить любой другой тип данных, он преобразует его в строку. Давайте разберем небольшой пример и создайте объект с несколькими переменными внутри:

```
var company = {
firstName: "Larry",
lastName: "Page",
age: 47
};

localStorage.setItem("google", company);
```

Объект **company** содержит в себе несколько переменных (firstName, lastName, age) с разными типами данных. Записав данные в хранилище, в консоли разработчика мы получим значение **[object Object]**:

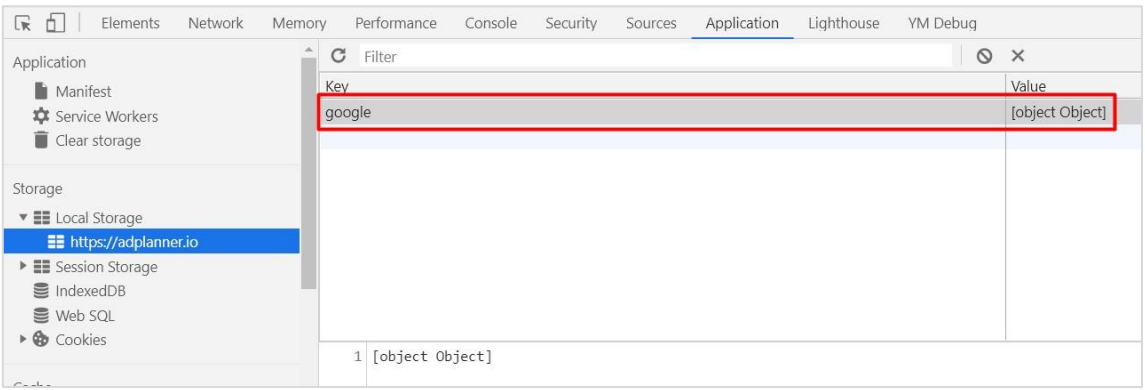


Рис. 887. Конвертация объекта JavaScript в строку приводит к [object Object]

Чтобы правильно хранить объекты JavaScript в localStorage, нужно сначала преобразовать наш объект в JSON строку. Для этого используется функция функцию **JSON.stringify()**. Таким образом вместо **localStorage.setItem("google", company);** нужно написать:

```
localStorage.setItem("google", JSON.stringify(company));
```

Тогда значение в **Value** изменится и наш объект будет корректно отображаться:

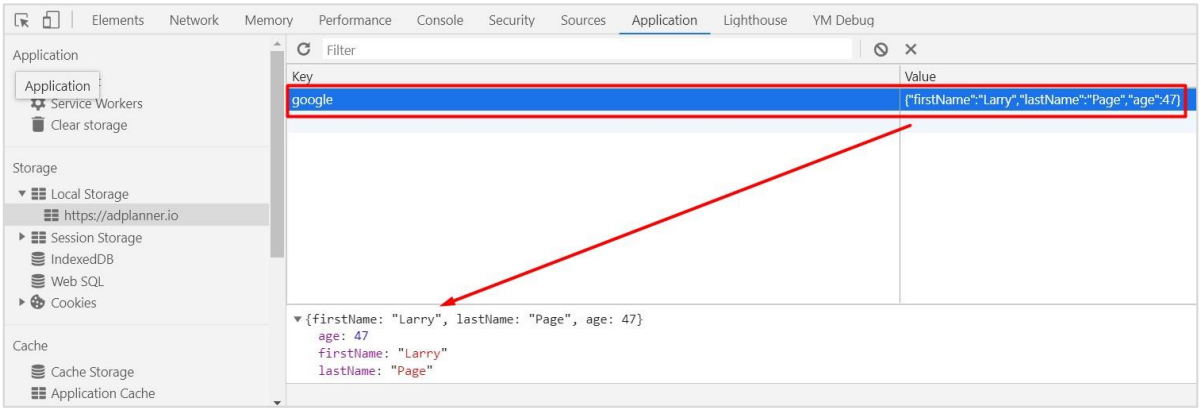


Рис. 888. Использование JSON.stringify()

Если вы извлечем данные с помощью getItem и посмотрим на тип, то увидим, что он является строкой (string):

```
// "{"firstName":"Larry","lastName":"Page" ,"age":47}"
localStorage.getItem("google");
typeof "google" // "string"
```

Строковые значения перебрать в цикле не получится. Для этого нам нужно преобразовать строку localStorage обратно в объект с помощью функции **JSON.parse()**:

```
JSON.parse(localStorage.getItem("google"));
```

Теперь мы можем работать с объектом и извлекать из него определенные значения. Например, давайте выведем значение из ключа **lastName**:

```
JSON.parse(localStorage.getItem("google")).lastName
"Page"
```

В Google Tag Manager используйте следующий код для получения определенного значения из массива с помощью переменной типа **Собственный код JavaScript**:

```
function() {
  var object = JSON.parse(localStorage.getItem('google'));
  return object.lastName;
}
```

, где **google** - ваш ключ, а **.lastName** - конкретная переменная внутри массива, значение которой хотите извлечь.

Счетчик просмотренных страниц

Если вам необходимо запустить триггер, тег после просмотра пользователем определенного количества страниц (1,2,3...N страниц и т.д.), вы можете использовать один из нижеописанных способов, основанных на работе с файлами cookie.

Данная задача является очень распространенной, поскольку позволяет посчитать для каждого пользователя количество просмотренных страниц в течение определенного времени, и, в зависимости от этого, настроить для него персональное предложение. Например, если пользователь просмотрел на сайте 5 страниц, то показать ему всплывающее окно с подпиской на новости, 10 - баннер со скидкой/подарком и т.д.

И поскольку это не так сложно, как кажется на первый взгляд, перейдем сразу к настройке. Способы, описанные в этой статье, уже были разобраны в интернете задолго до меня. В русскоязычном сообществе я пока не видел решений в GTM на основе файлов cookie. Есть материал (см. приложение) за 2015 год от Ивана Иванова, автора блога **prometriki.ru**, но он немного устарел. Предлагаю вам обновленный вариант настройки счетчика просмотренных страниц с помощью Google Tag Manager.

Важно: из-за различий в работе браузеров представленные ниже способы и решения подсчета могут отслеживаться по-разному.

Решение №1. ProMetriki

Создайте тег типа **Пользовательский HTML** со следующим кодом:

```
<script>
    var counter = getCookie('counter');
    if ( counter != null ){
        counter++;
    }
    else {
        counter = 1;
    }
    setCookie('counter', counter);
</script>
```

В Google Tag Manager это будет выглядеть так:

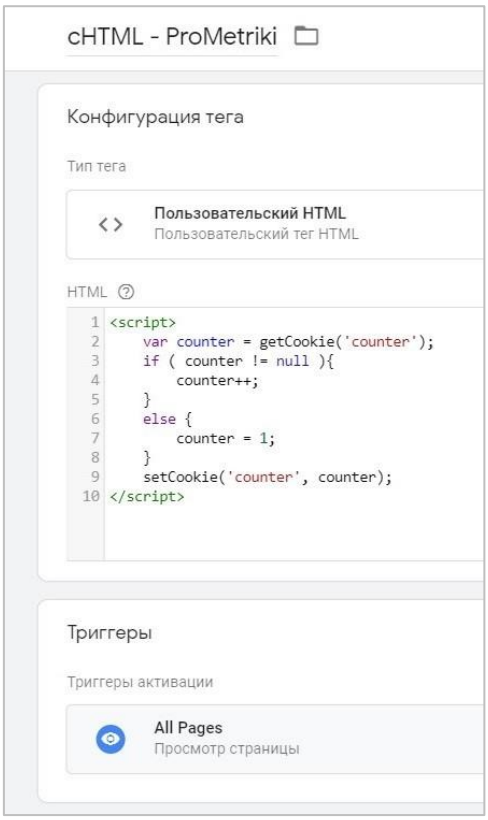


Рис. 889. Скрипт от prometriki.ru

Триггер активации - **Все страницы (All Pages)**.

Сначала с помощью команды **getCookie** мы получаем значение cookie. Затем идет проверка - если оно не установлено, то присваивается значение **1**, а если установлено, то счетчик увеличивает значение на единицу (**+1**). По окончании мы устанавливаем обновленное значение куки с помощью **setCookie**. Поскольку в setCookie не задан третий параметр, то куки являются *сеансовыми* и будут существовать пока пользователь не закроет браузер. Чтобы получать значение из **counter** с момента последнего открытия страницы сайта в течение

определенного времени, необходимо установить дополнительное значение. Добавим его чуть позже в другом способе.

Чтобы получить значение из **counter**, необходимо создать пользовательскую переменную типа **Основной файл cookie (1st Party Cookie)** с названием **counter**:

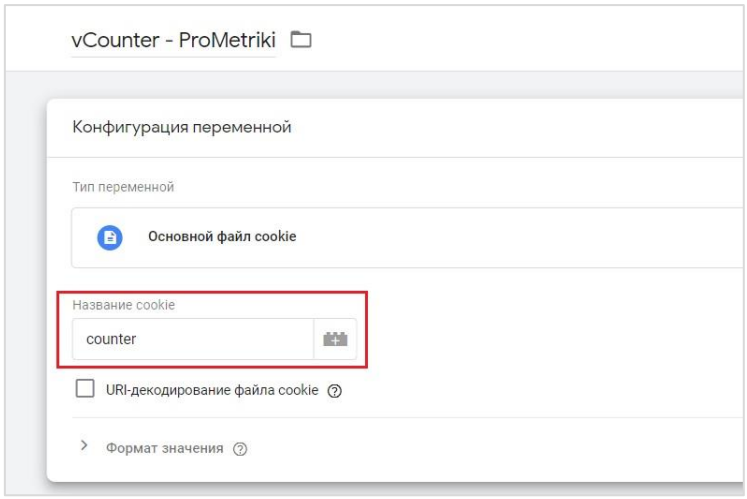


Рис. 890. Переменная Основной файл cookie

Сохраняем настройки. Теперь необходимо настроить триггер активации. Для этого можно воспользоваться типом **Окно загружено** и задать условие, в результате которого будет активироваться триггер. Например, если пользователь просмотрит **больше или равно 3** страницам:

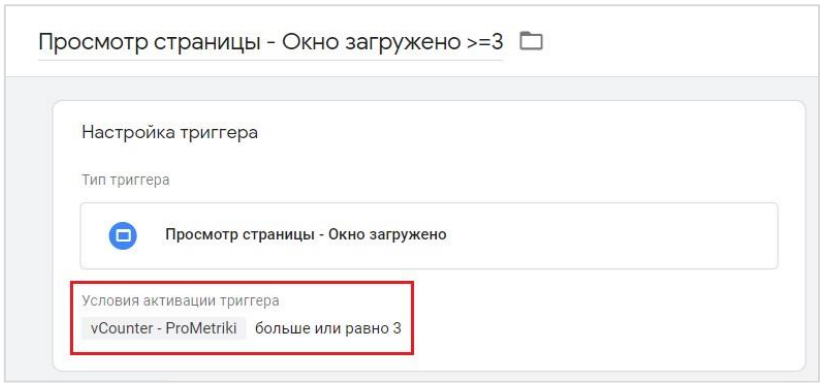


Рис. 891. Настройка триггера

На последнем шаге вы можете создать тег **Universal Analytics** и передать данные в Google Analytics, а можете использовать триггер для запуска другого кода, например, который активирует показ всплывающего окна для пользователя.

Решение №2. gtmscripts.com

Действия по настройке аналогичны предыдущему примеру. Создайте тег типа **Пользовательский тег HTML** со следующим кодом (см. приложение):

```
<script>
  (function() {
  if (typeof {{PageNumber}} === 'undefined') {
var cookieName  = "PageNumber";
var cookieValue = 0;
var cookiePath  = "/";
var expirationTime = 1800; // Срок жизни в секундах, 30 минут (60*30)
expirationTime = expirationTime * 1000;
var date = new Date();
var dateTimeNow = date.getTime();
date.setTime(dateTimeNow + expirationTime);
```

```
   var expirationTime = date.toUTCString();
   document.cookie = cookieName+"="+cookieValue+"; expires="+expirationTime+";
path="+cookiePath;
     }

   function readCookie(name) {
   var nameEQ = name + "=";
   var ca = document.cookie.split(';');
   for (var i = 0; i < ca.length; i++) {
   var c = ca[i];
   while (c.charAt(0) == ' ') c = c.substring(1, c.length);
   if (c.indexOf(nameEQ) == 0) return c.substring(nameEQ.length, c.length);
   }
   return null;
   }

   var cookieName  = "PageNumber";
   var cookieValue = 1 + parseInt(readCookie("PageNumber"));
   var cookiePath  = "/";
   var expirationTime = 1800; // Срок жизни в секундах, 30 минут (60*30)
   expirationTime = expirationTime * 1000;
   var date = new Date();
   var dateTimeNow = date.getTime();
   date.setTime(dateTimeNow + expirationTime);
   var expirationTime = date.toUTCString();
   document.cookie = cookieName+"="+cookieValue+"; expires="+expirationTime+";
path="+cookiePath;
    })();
   </script>

   <script>
   dataLayer.push({'event':'PageViewEvent'});
   </script>
```

В Google Tag Manager это выглядит так:

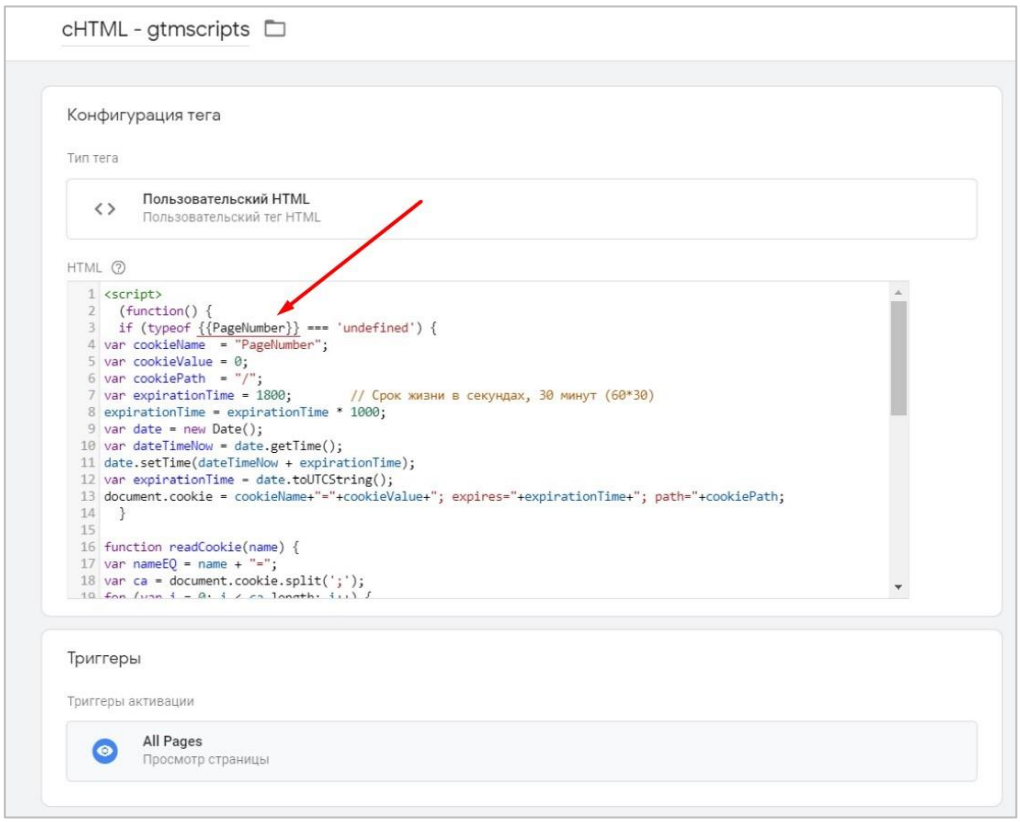


Рис. 892. Скрипт от gtmscripts.com

Триггер активации - **Все страницы (All Pages)**.

В этом коде есть два существенных отличия:

- переменная **{{Page Number}}**, на которую ссылаются (выделена стрелкой на рисунке). Здесь необходимо использовать ту переменную, которую мы создайте далее с именем **PageNumber;**
- дополнительная строчка кода с методом **dataLayer.push({'event':'PageViewEvent'})**, которая вызывает событие **PageViewEvent** (название может быть произвольным) каждый раз после просмотра страницы. Оно нам пригодится позже при настройке триггера.

В примере срок жизни куки составляет 1800 секунд (30 минут). По умолчанию сеанс в Google Analytics и Яндекс.Метрике заканчивается через 30 минут бездействия.

Необходимо создать пользовательскую переменную типа **Основной файл cookie (1st Party Cookie)** с названием cookie **PageNumber**:

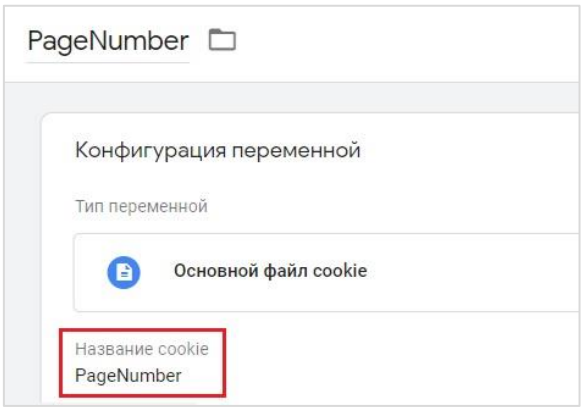


Рис. 893. Переменная Основной файл cookie

Если вы планируете ее переименовать, то и в коде выше также следует внести изменения. Я не рекомендую этого делать, чтобы не сбить корректность работы скрипта. Просто сохраните изменения.

Корректность настройки можно проверить с помощью режима отладки и консоли разработчика. В режиме предварительного просмотра появится наше событие **PageViewEvent**, а на вкладке **Application - Cookies** отобразится значение куки **PageNumber:**

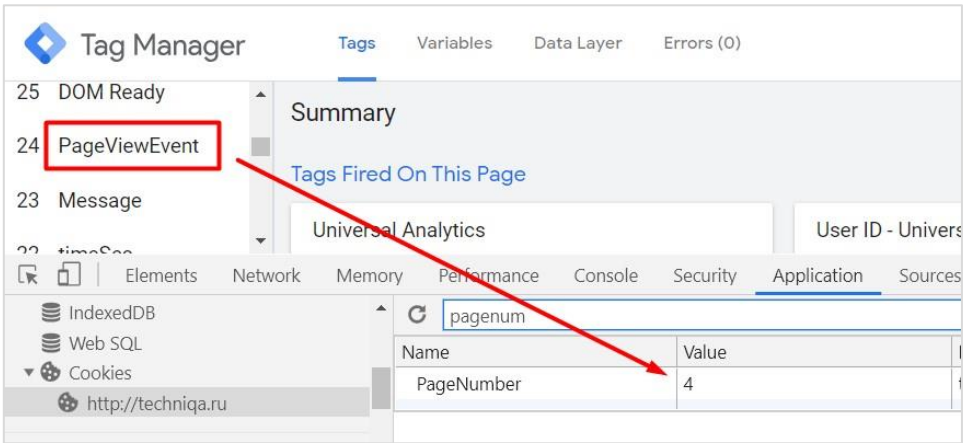


Рис. 894. Событие PageViewEvent и значение куки в консоли разработчика

Значение куки (число 4) мы можем посмотреть не только в консоли разработчика, но и на вкладке **Variables** напротив **PageNumber**:

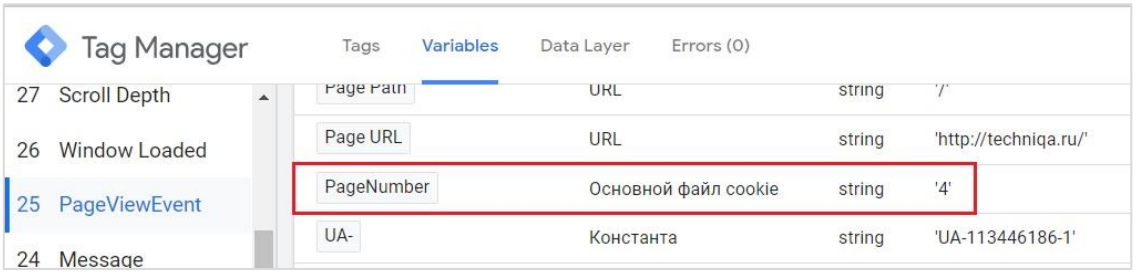


Рис. 895. Значение PageNumber на вкладке Variables

Теперь можем создать триггер активации типа **Пользовательское событие** с именем **PageViewEvent** для активации других кодов и тегов, если количество просмотров страниц у пользователя, например, будет **больше 3**:

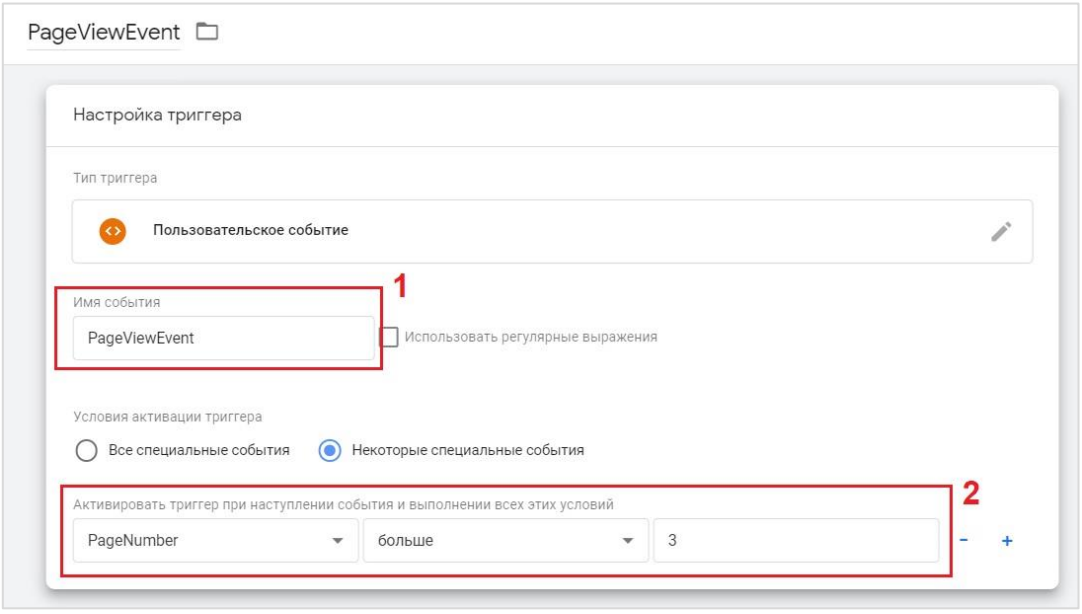


Рис. 896. Триггер сработает, если количество просмотров страниц будет больше 3

Решение №3. analyticsmania.com

Самый простой способ настройки счетчика с помощью cookie и GTM. Все, что нужно сделать, это:

- перейти на страницу (см. приложение) блога **Analytics Mania**;
- скачать готовый файл JSON с настроенными переменными, триггерами и тегами;
- импортировать файл к себе в контейнер GTM;
- настроить любой тег, который вы хотите, триггеру **Window Loaded - 3rd Page View.** Это решение устанавливает cookie и создает только триггер. Вам придется создать собственный тег для передачи данных в инструменты веб-аналитики вручную.
- проверить корректность всех настроек с помощью режима предварительного просмотра. Если все хорошо, просто опубликуйте контейнер GTM с изменениями;
- изменить срок жизни куки (необязательно), который в этом скрипте составляет 30 дней. Если вы хотите изменить срок действия, замените число 30 на собственное значение в теге **cHTML - Set Cookie - Pageview Counter**;

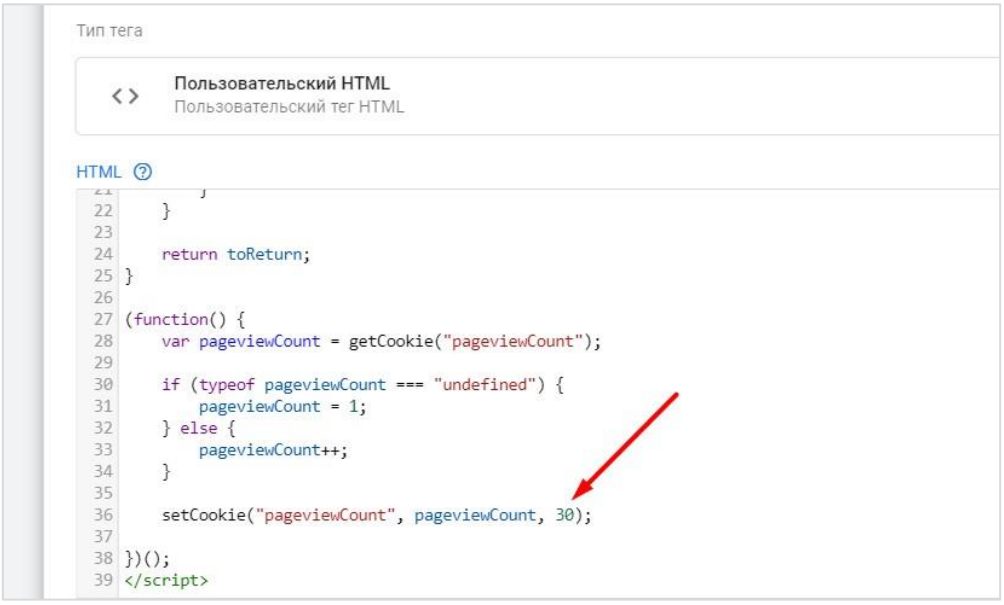


Рис. 897. Изменение срока жизни куки

- изменить правило активации триггера (необязательно), которое по умолчанию составляет **Количество просмотренных страниц больше или равно 3**:

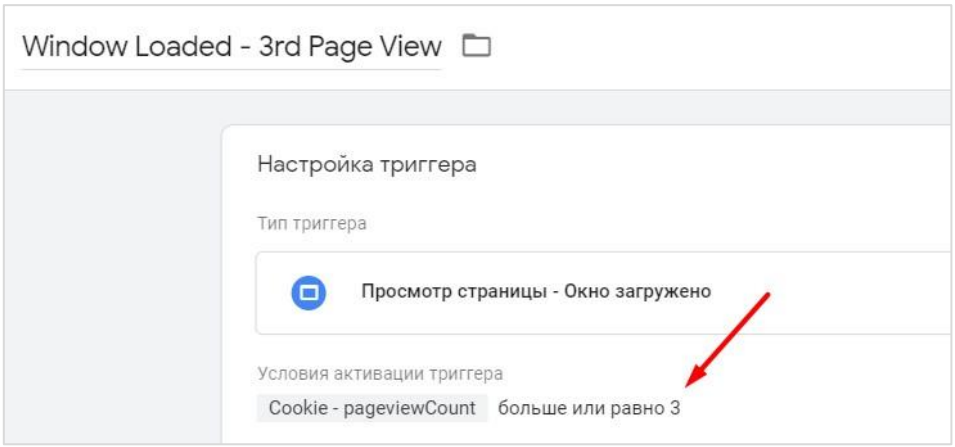


Рис. 898. Настройка триггера

Сохраните изменения. Как и в примерах выше, вы можете создать тег Universal Analytics и передать данные в Google Analytics, а можете использовать триггер для запуска другого кода, например, показа баннера.

Если вы хотите, чтобы триггер срабатывал не каждый раз, когда значение в куки будет больше или равно 3, а только один раз, когда оно равно определенному значению (например, просмотр 5 страниц), то просто измените условие активации триггера с **больше или равно 5** на условие **равно 5**. Аналогично можно сделать и для пользовательского события **PageViewEvent**.

Отслеживание времени сеанса пользователей

Как вы уже знаете, в основе принципа работы счетчиков веб-аналитики лежат различные уровни организации данных. Еще их называют *областями действия*:

- хит (hit);
- сессия (session);
- пользователь (user).

Хит (обращение, hit) – взаимодействие пользователями с сайтом, в результате которого данные отправляются в аналитику.

Хитами могут быть просмотры страницы, просмотры экрана, события, транзакции и т.д. При заходе посетителя на сайт сразу отправляется первый хит – просмотр страницы (pageview).

Сессия (сеанс, session, визит) – последовательность хитов (взаимодействий) пользователя с сайтом за определенный промежуток времени. По умолчанию длительность сеанса (время ожидания сеанса) составляет 30 минут.

Пользователь (посетитель, user) – совокупность сеансов, которые совершаются с одного и того же браузера и имеют один файл cookie. Фактически, пользователь – это уникальный куки файл (уникальный идентификатор отслеживания, он же Client ID) за определенный период времени.

Рис. 899. Области действия

Примечание: в Google Analytics есть четвертая область действий – **Товар.**

Из хитов строятся сессии, которые привязываются к определенному пользователю, который имеет свою куку и уникальный идентификатор отслеживания. Все обращения, которые пользователь выполняет на сайте, имеют привязку по времени. Чуть ранее в руководстве мы познакомились со способом отслеживания точного времени каждого обращения с помощью пользовательского параметра **Hit Timestamp**.

Большинство отчетов в Google Analytics строятся на основе сеансов и тому, что в них происходило. Хоть они и содержат данные об общей продолжительности сессии, отчеты о конкретных событиях не имеют такой информации. По статистике мы никак не сможем понять, сколько прошло времени от захода пользователя на сайт до момента совершения нужного нам события, например, отправки формы или покупки. Понадобилось посетителю 2 минуты, 20 секунд или 15 минут на принятие решения? Подлинно неизвестно.

С помощью настройки отслеживания времени сеанса мы сможем узнать точное время совершения любого события на сайте. Для этого потребуется определить два значения (время начала сеанса пользователя и текущее время совершения события), а затем посчитать разницу между этими двумя временными интервалами.

Начнем с создания трех переменных.

Перейдите в раздел **Переменные** и создайте пользовательскую переменную типа **Переменная уровня данных** с именем **gtm.start**:

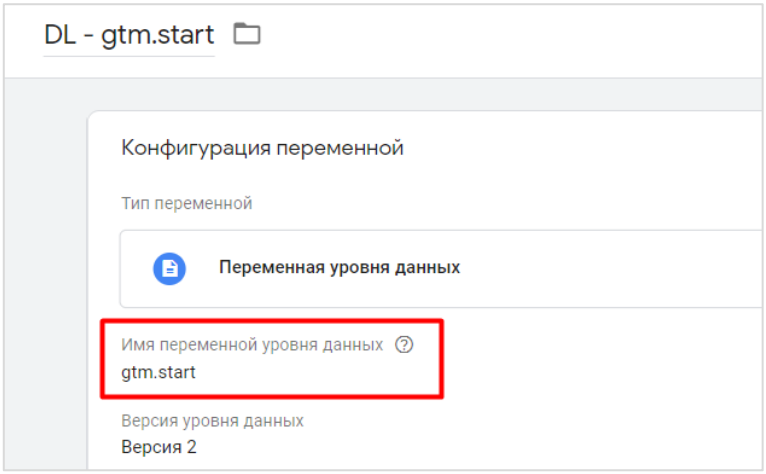


Рис. 900. Переменная уровня данных gtm.start

Название можно указать произвольное. В моем примере **DL - gtm.start**. Когда скрипт Google Tag Manager загружается на каждой странице, он заносит временную метку в dataLayer в **gtm.start**:

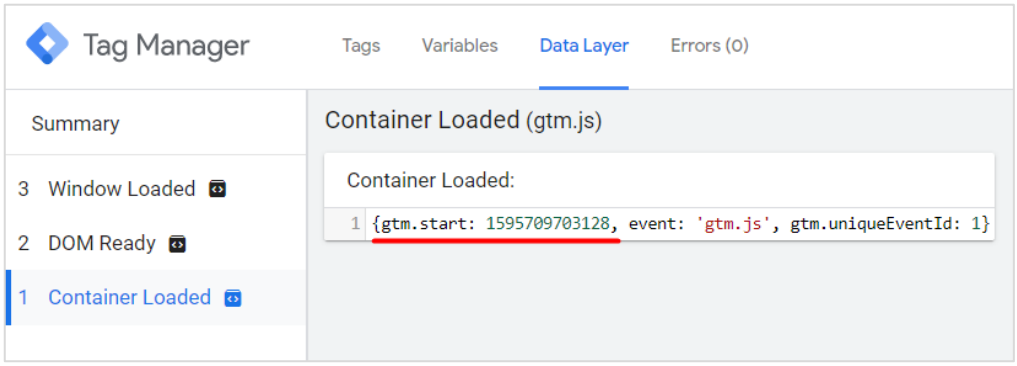


Рис. 901. Временная метка в режиме отладки

Мы можем получить доступ к этой отметке времени, используя переменную уровня данных, которую мы создали:

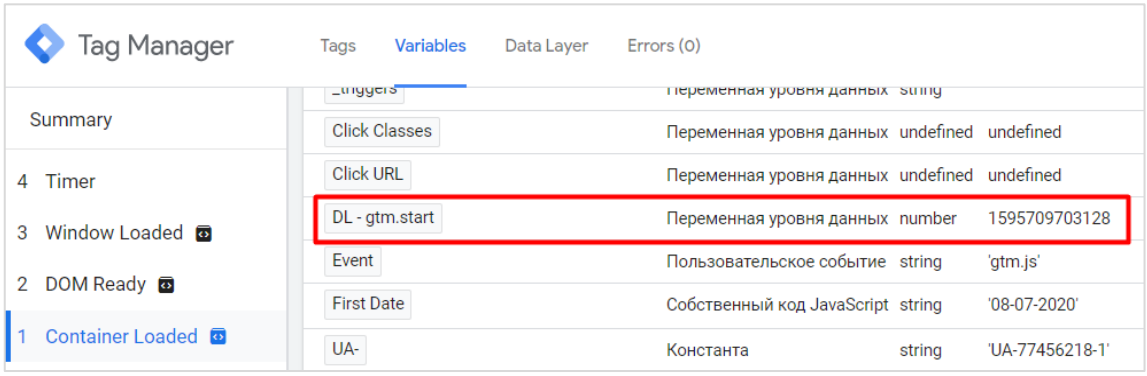


Рис. 902. Начальное значение, установленное Google Tag Manager

Переменная **DL - gtm.start** извлекает текущее значение начала сеанса.

Теперь создайте переменную типа **Основной файл cookie** с именем **gtm-session-start**:

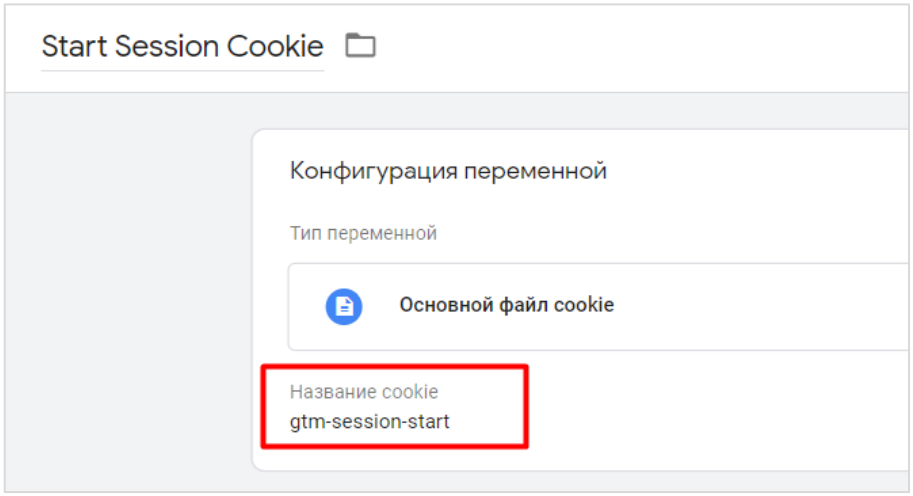


Рис. 903. Переменная Основной файл cookie

Название можно указать произвольное. В моем примере **Start Session Cookie**. Эта переменная используется для получения значения cookie из браузера пользователя. Она будет использоваться в другой переменной, которую мы создайте на следующем шаге.

Теперь следует создать переменную типа **Собственный код JavaScript** с именем **CJS – Session Seconds**, в которую нужно добавить этот код (см. приложение):

```
function(){
    var time;
    try {
        var cookieStart = {{Start Session Cookie}};
        if(typeof cookieStart !== 'undefined' && cookieStart > 0) {
            var date = new Date();
            time = Math.round((date.getTime() - cookieStart)/1000);
        }
    } catch(e) {
    }
    return time;
}
```

, где в объявлении переменной var cookieStart после знака равно необходимо указать собственную переменную **Start Session Cookie** в фигурных скобках. В Google Tag Manager это будет выглядеть так:

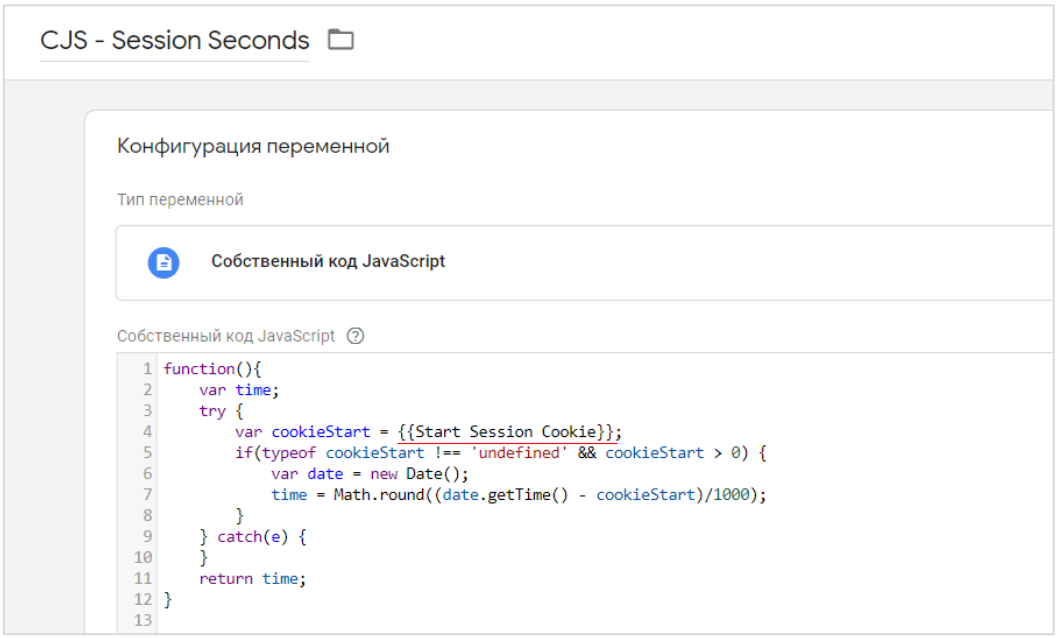


Рис. 904. Переменная Собственный код JavaScript

Данная переменная JavaScript получает время начала сохраненного сеанса из файла cookie браузера пользователя и вычитает его из текущего времени. Это значение - время, прошедшее с начала сеанса. Переменная будет постоянно обновляться при каждом вызове или использовании в тегах и триггерах, обеспечивая текущее время возврата.

В коде есть отсылка к переменной **Start Session Cookie**, которая будет возвращать время начала сеанса пользователя. Если это значение определено и имеет значение больше нуля, функция получит текущее время, и вычтет время начала сеансов. Результатом будет округленное число после деления на 1000, чтобы получить время в секундах, а не в миллисекундах.

Теперь, когда все переменные созданы, нам необходимо настроить пользовательский тег HTML, который будет отвечать за настройку времени начала сеанса в файле cookie браузера пользователя. Цель этого тега - сначала установить правильное время сеанса, а также убедиться, что оно не перезаписывается, когда пользователь перемещается по сайту, от страницы к странице.

Для создания тега перейдите на вкладку **Теги** и создайте тег типа **Пользовательский HTML**. Код для копирования и вставки:

```
<script>
    try {
        (function(){
            var current = {{Start Session Cookie}};
            var cookieValue;
            if (typeof(current) == "undefined") {
```

```
                cookieValue = {{DL - gtm.start}};
            } else {
                cookieValue = current;
            }

            var date = new Date();
            date.setTime(date.getTime()+(30*60*1000));
            var cookieName = 'gtm-session-start';
            var cookieExpires = date.toGMTString();
            var cookiePath = '/';

            document.cookie = cookieName + '=' + cookieValue + '; Expires=' +
cookieExpires + '; Path=' + cookiePath;
        })();
    } catch (e) {
    }
</script>
```

В Google Tag Manager это будет выглядеть так:

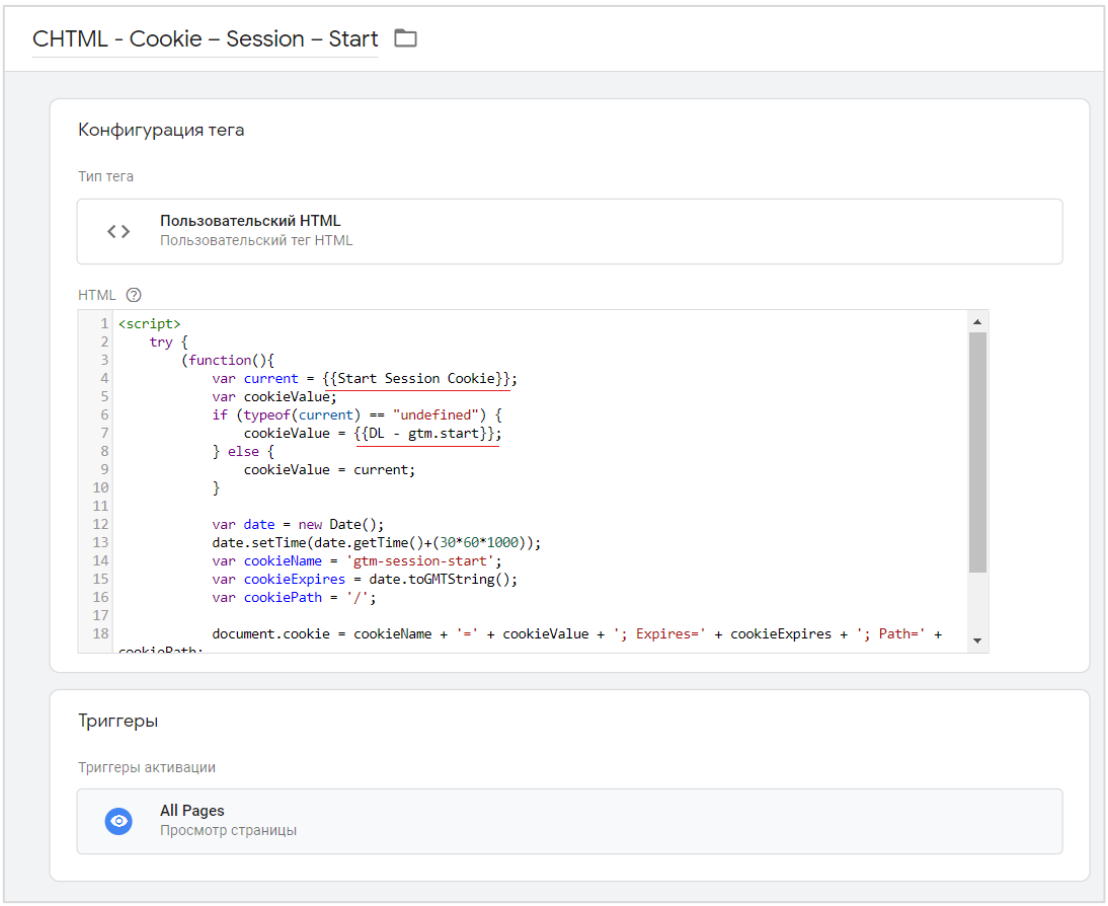


Рис. 905. Пользовательский HTML тег

Триггер активации – **Все страницы (All Pages).**

Данный код ссылается на две переменных, которые мы создали ранее: **Start Session Cookie** и **DL - gtm.start**. Сначала он попытается получить время начала сеанса из файла cookie браузера **Start Session Cookie**. Если в настоящее время значение не определено (undefined), то предполагается, что это начало сеанса, и мы получаем время начала сеанса из Google Tag Manager (**gtm.start**). Такая проверка позволяет сохранить время сеанса (если оно установлено) и данное значение не переопределяется (обнуляется) при переходе пользователя по сайту.

После того, как текущая переменная времени была установлена, скрипт создаст файл cookie. Он запросит текущую дату и добавит к ней 30 минут. Это то же самое, что значение по умолчанию для времени ожидания сеанса Google Analytics.

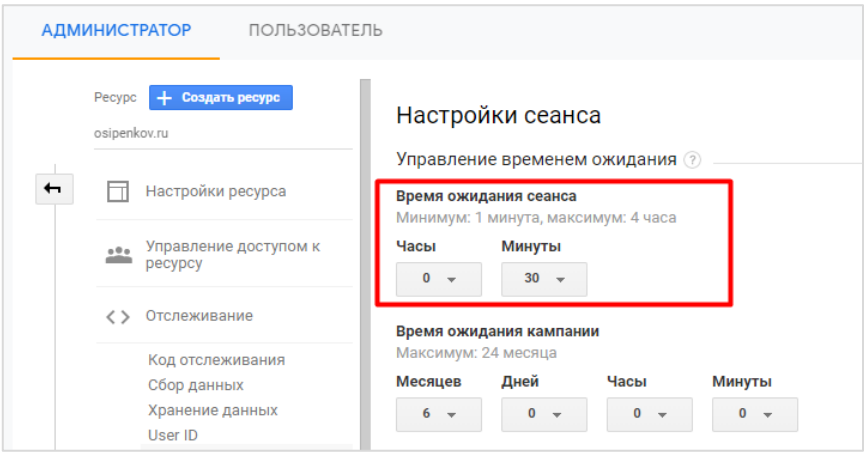


Рис. 906. Настройки сеанса - Время ожидания сеанса

Примечание: если вы обновили этот параметр в Google Analytics (**Ресурс – Отслеживание – Настройки сеанса**), установите в коде аналогичное значение.

Затем файлу присваивается имя, соответствующее переменной типа Основной файл cookie **(gtm-session-start)**, которую мы уже настроили. Последняя строка устанавливает cookie в браузере со всеми значениями, которые мы определили.

Поскольку файл cookie создается заново при каждом просмотре страницы, срок его действия постоянно обновляется. Это необходимо для поддержания актуального времени начала сеанса и для будущих посещений пользователем веб-сайта, чтобы обеспечить его перезапуск, когда это необходимо.

На этом настройка отслеживания времени сеанса завершена. Теперь мы можем использовать переменную **CJS – Session Seconds** для создания условий активации триггеров, в тегах, и для отслеживания различных взаимодействий пользователей на вашем сайте. Например, когда пользователь кликает ссылку, начинает смотреть видео, оставляет заявку и т.д. Либо использовать триггер с каким-нибудь условием, завязанным на длительности сеанса. Например, активировать пользовательское событие и показать всплывающее окно через 30 секунд после начала сеанса.

Для завершения настройки отслеживания мы можем создать тег типа **Google Analytics – Universal Analytics** и передать эту информацию в отчеты:

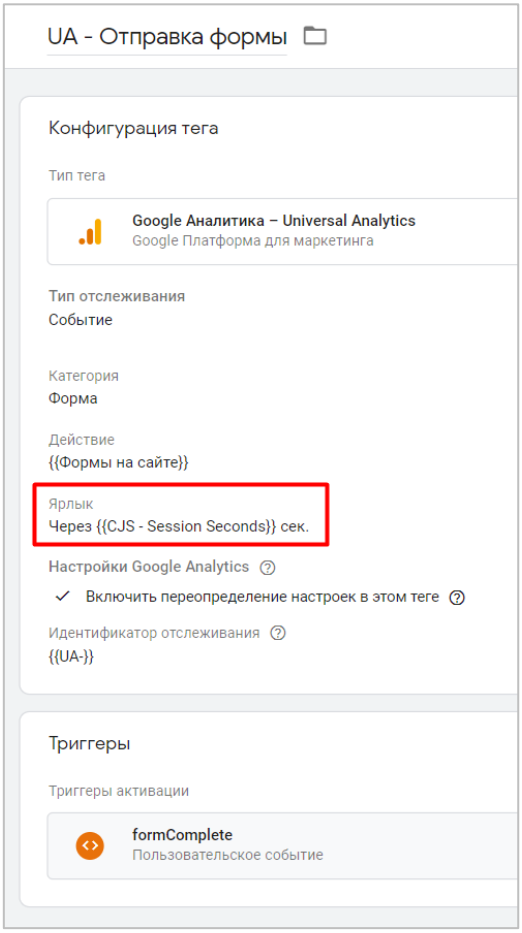


Рис. 907. Настройка тега Google Аналитика – Universal Analytics

В качестве примера я использую тег с типом **Событие** и триггер активации на отправку формы, а в ярлыке события передаю значение переменной **CJS – Session Seconds**, в которой хранится итоговое время длительности сеанса в секундах. Таким образом, я могу узнать, через какое время после начала сеанса пользователь отправил заявку.

Проверить корректность передачи данных можно с помощью режима отладки в GTM и отчетов **В режиме реального времени** в Google Analytics:

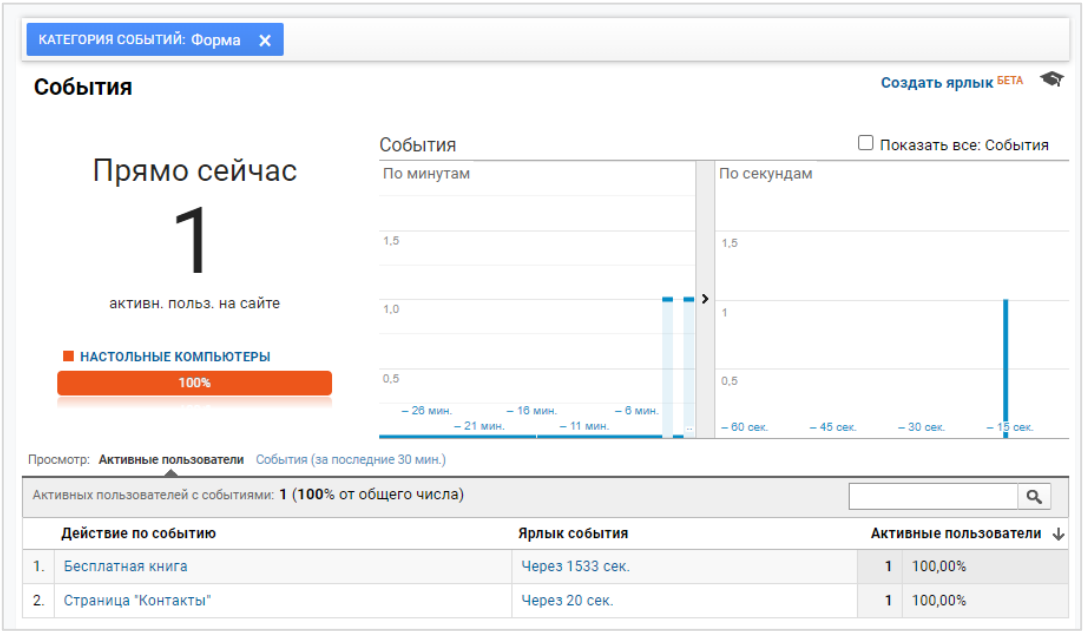


Рис. 908. Отчет В режиме реального времени

Через некоторое время все данные по событиям появятся в отчете **Поведение — События — Лучшие события**.

Предотвращение дублей транзакций/конверсий

Настроили отслеживание электронной торговли и стали замечать в Google Analytics дубли транзакций? Пользователи несколько раз отправляют заявку, тем самым искажают статистику? Давайте попробуем исправить эту ситуацию с помощью Google Tag Manager.

Повторяющиеся транзакции/конверсии означают, что в инструментах аналитики несколько раз регистрируется одно и тоже обращение. Эта проблема может возникать по нескольким причинам:

- пользователь обновляет страницу;
- пользователь через некоторое время возвращается на эту страницу из закладок, по электронной почте, прямому заходу и т.д., чаще всего - в новом сеансе;
- пользователь возвращается на страницу с помощью кнопок в браузере **Вперед - Назад**;

Наиболее распространенная причина, по которой происходят повторяющиеся обращения, заключается в том, что событие о транзакции/конверсии отправляется в Google Analytics каждый раз при загрузке страницы с благодарностью. Есть пользователи, которые сохраняют эту страницу и возвращаются, чтобы проверить идентификатор заказа или номер заявки. А некоторые компании отправляют в электронном письме URL с подтверждением заказа.

Чтобы узнать, есть ли у вас в аналитике повторные транзакции, создайте в Google Analytics специальный отчет с двумя метриками: **Параметр - Идентификатор транзакции, Показатель - Транзакции**:

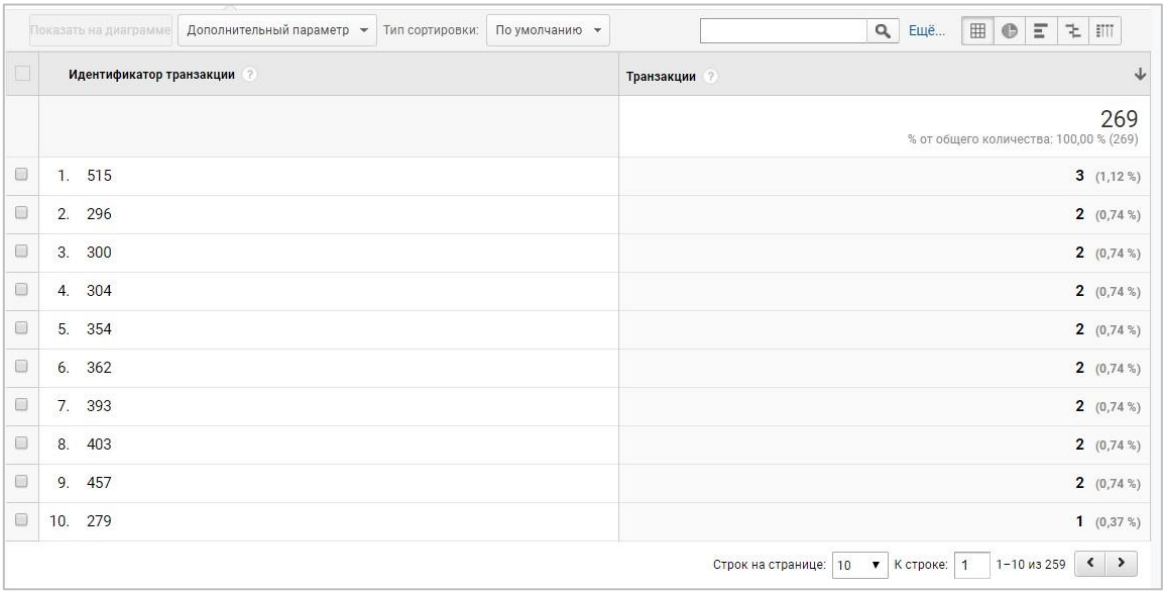


Рис. 909. Пример специального отчета

Обратите внимание на два числа - **общее количество транзакций** в отчете и **общее количество строк в отчете**. Если они совпадают и для каждого идентификатора транзакции показана только одна транзакция (1), это значит, что за выбранный период у вас не было повторных транзакций.

Если есть строчки, где вместо 1 стоит другое число (как в примере выше), то это явный признак дублей. Аналитика несовершенна, и поэтому не стоит сразу бить тревогу, искать виновных и бежать исправлять. Необходимо оценить масштаб проблемы. Сравните количество транзакций с количеством строк в таблице. Если ваш коэффициент дублирования транзакций составляет более 10-15%, тогда задумайтесь об исправлении ситуации. Если ниже, то просто понаблюдайте за изменениями во времени. Вполне вероятно, что это просто статистическая погрешность и редкие сбои в отслеживании.

При большом количестве трафика повторных обращений может быть значительная доля. А искажение данных затрудняет их анализ, особенно в e-commerce, где с увеличением количества транзакций также

увеличивается и доход, и изменяются показатели, непосредственно связанные с транзакциями, например, цена за транзакцию и коэффициент транзакции. Впоследствии это приведет к неверным выводам.

Дубли встречаются не только в интернет-магазинах, но и у обычных сайтов, чаще всего - на все тех же страницах благодарности, на которые перенаправляются пользователи после успешной отправки формы. Существует несколько способов решения проблемы, не связанных с работой маркетолога. Понадобится разработчик, который сможет на стороне сервера осуществить проверку информации о заказе пользователя после его оформления. Когда человек оставляет заказ на сайте, данные о транзакции сохраняются в базе данных. Далее идет проверка - если в базе ID транзакции нет, данные отправляются в Google Analytics - если ID транзакции уже существует, то ничего не отправляется. Еще можно реализовать так - после успешного оформления заказа перенаправлять пользователя на другую страницу. А при каждой его попытке вернуться на страницу с благодарностью - снова делать редирект. Но этот способ менее предпочтительный, чем описанный выше.

Если говорить о работе интернет-маркетолога, то он тоже может отследить дубли транзакций/конверсий, чтобы запретить повторную передачу данных в Google Analytics. Способов существует большое количество. Есть реализация для расширенной электронной торговли от Симо Ахавы **с помощью customTask**, универсальное решение от Дэвида Вальехо**,** один из первых способов отслеживания от Bounteous и дополненный от команды Renta, в некоторых случаях можно использовать Measurement Protocol (см. приложение).

Я хочу разобрать в этой статье отслеживание как повторяющихся транзакций для интернет-магазина, так и дубли конверсий (для посадочной страницы) с помощью 3 переменных, 2 триггеров и 1 тега, используя cookie и GTM. Способ подойдет для разных типов сайтов.

Дубли транзакций для интернет-магазинов

Пример: на сайте установлена расширенная электронная торговля, которая настроена через Google Tag Manager. Для отслеживания действий используется уровень данных (dataLayer), на странице с успешно оформленным заказом отправляется действие **purchase** согласно документации Google (см. приложение):

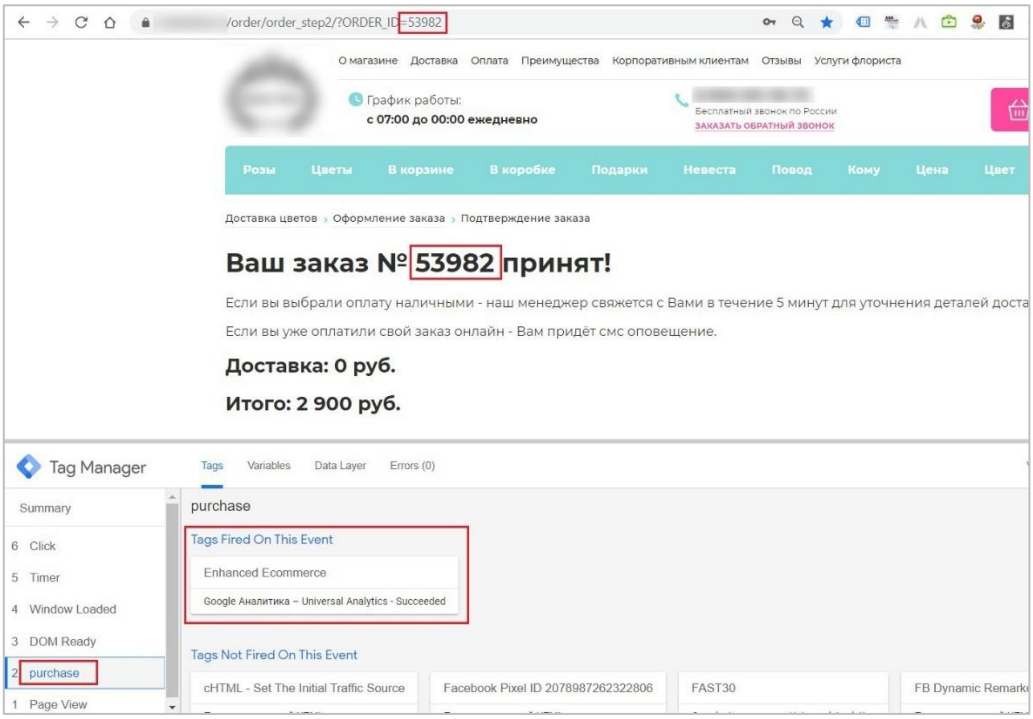
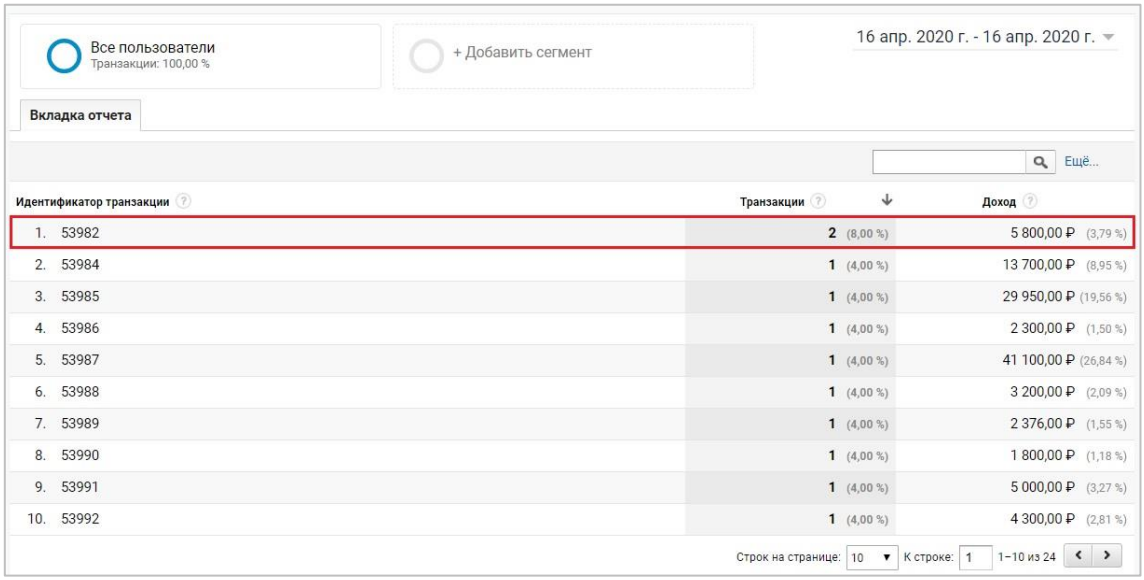


Рис. 910. Пример заказа на сайте

Тег активировался, событие успешно отправлено в Google Analytics. Я обновил страницу несколько раз. Событие не срабатывало, тег не активировался. Это свидетельствует о том, что после совершения транзакции данные о заказе обнуляются в этой же сессии. Таким образом, при обновлении страницы повторного события не происходит, а значит, тег не активируется и не передает данные в Google Analytics

Но стоит дождаться нового сеанса, скопировать полный URL-адрес **https://site.ru/order/order_step2/?ORDER_ID=53982** в адресную строку браузера и нажать Enter, как мы снова отправим событие об успешной транзакции с тем же ID. А в отчетах Google Analytics увидим дубль транзакции:



Идентификатор транзакции	Транзакции	Доход
1. 53982	2 (8,00 %)	5 800,00 ₽ (3,79 %)
2. 53984	1 (4,00 %)	13 700,00 ₽ (8,95 %)
3. 53985	1 (4,00 %)	29 950,00 ₽ (19,56 %)
4. 53986	1 (4,00 %)	2 300,00 ₽ (1,50 %)
5. 53987	1 (4,00 %)	41 100,00 ₽ (26,84 %)
6. 53988	1 (4,00 %)	3 200,00 ₽ (2,09 %)
7. 53989	1 (4,00 %)	2 376,00 ₽ (1,55 %)
8. 53990	1 (4,00 %)	1 800,00 ₽ (1,18 %)
9. 53991	1 (4,00 %)	5 000,00 ₽ (3,27 %)
10. 53992	1 (4,00 %)	4 300,00 ₽ (2,81 %)

Рис. 911. Дубль транзакции

Что же делать?

Алгоритм настройки в Google Tag Manager следующий:

- создаем переменную, которая будет извлекать со страницы благодарности ID транзакции;
- создаем переменную, которая будет извлекать значение из куки;
- создаем переменную, которая сопоставляет значения двух переменных и выдает результат *true* или *false*;
- создаем тег, который будет устанавливать значение для cookie при удовлетворении условия;
- используем существующий триггер purchase, настроенный для отслеживания успешной отправки заказа;
- создаем триггер-блокировки, который добавим в исключения;

Рассмотрим настройку более подробно.

Создание переменной с ID транзакции (Переменная уровня данных)

Поскольку dataLayer у нас уже сформирован, то остается только извлечь значение ID транзакции с помощью пользовательской переменной типа **Переменная уровня данных**. Это проще всего сделать с помощью расширения для браузера **Datalayer Checker**. Оно отобразит конечный массив с данными на странице покупки. ID транзакции хранится в переменной id, а вся переменная (формат flat) выглядит как **ecommerce.purchase.actionField.id**.

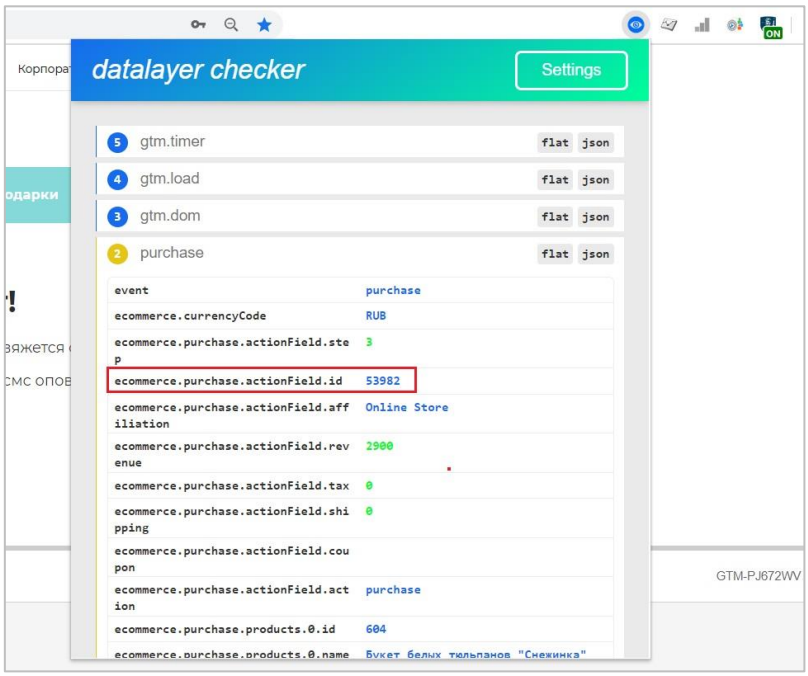


Рис. 912. Извлечение данных из переменной ID транзакции с помощью Datalayer Checker

В GTM:

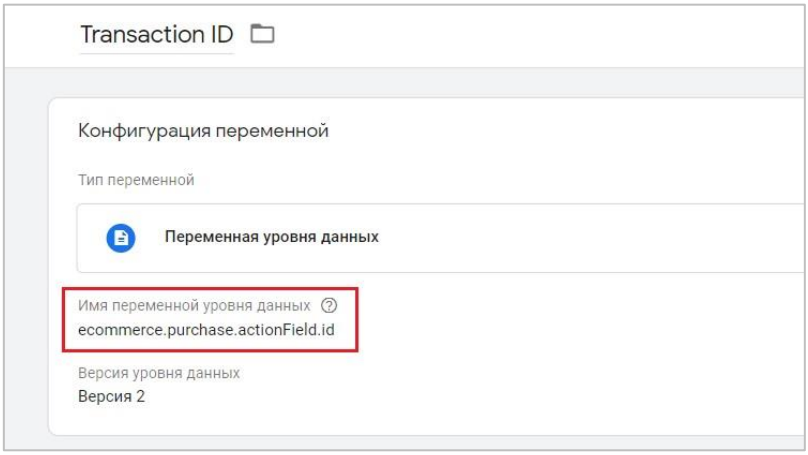


Рис. 913. Извлечение ID транзакции

Создание переменной Основной файл cookie (1st Party Cookie)

Переменная типа **Основной файл cookie**. Она будет читать значение из куки. Введите произвольное имя, например, **orderCookie**:

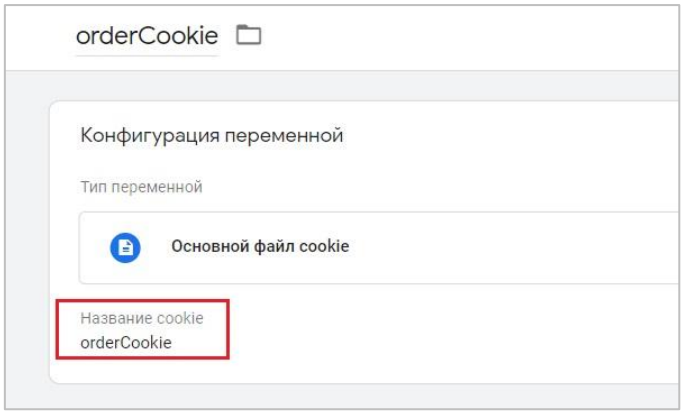


Рис. 914. Переменная Основной файл cookie

3. Создание переменной (Собственный код JavaScript)

Переменная типа **Собственный код JavaScript**. Код ее выглядит так:

```
function(){
  if( {{orderCookie}} === {{Transaction ID}} )
  {
    return false
  }
  else {
    return true
  }
}
```

В Google Tag Manager:

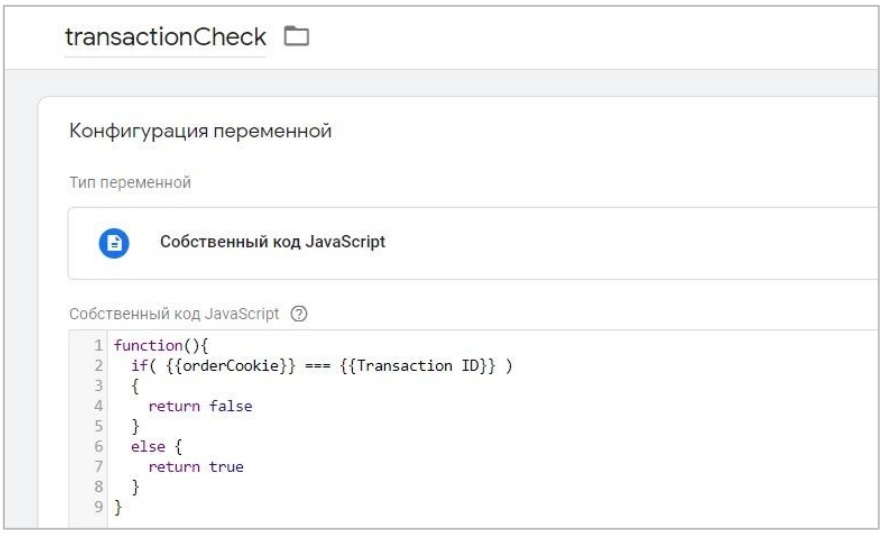


Рис. 915. Переменная Собственный код JavaScript

Переменная возвращает **false**, если значение транзакции в куки совпадает с текущим ID транзакции, и возвращает **true**, если не совпадает.

Создание тега (Пользовательский HTML)

Приведенный ниже код с помощью функции **setCookie** устанавливает cookie с именем **orderCookie** и значением переменной **Transaction ID**, при условии, что переменная **transactionCheck**, настроенная шагом ранее, вернула значение **true**.

```
<script>
function setCookie(name, value, expires) {
    document.cookie = name + "=" + escape(value) + "; path=/" + ((expires ===
null) ? "" : "; expires=" + expires.toGMTString());
}
var exp = new Date();
exp.setTime(exp.getTime() + (1000 * 60 * 60 * 24 * 30));
  if({{transactionCheck}}===true){
      setCookie("orderCookie",{{Transaction ID}},exp);
  }
</script>
```

В функции есть три параметра:

- **name** - имя для куки;
- **value** - значение для куки;
- **expires** - срок жизни куки;

В Google Tag Manager это выглядит так:

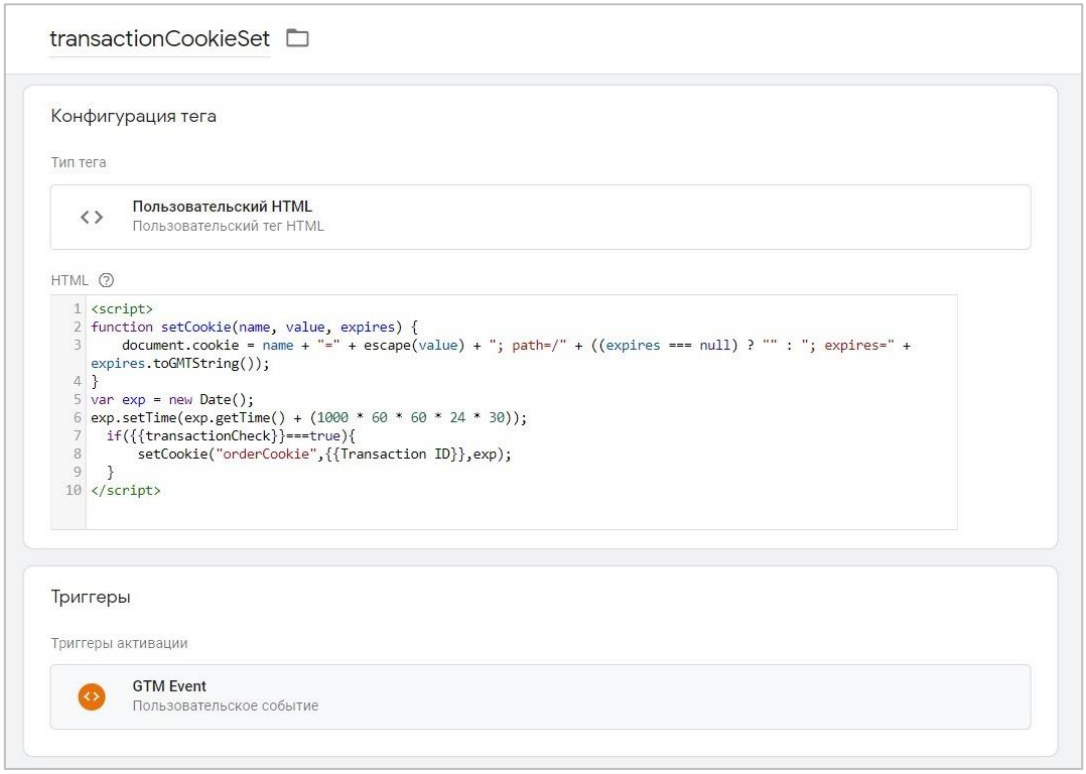


Рис. 916. Пользовательский тег HTML

Существующий триггер успешной покупки

В теге в качестве условия активации используем существующий триггер purchase, настроенный в электронной торговле для отслеживания успешной отправки заказа. В моем примере — это триггер типа **Пользовательское событие**.

Создание триггера блокировки

Необходимо скопировать триггер успешной покупки и добавить к нему условие, которое блокировало бы срабатывание тега транзакции дважды. Таким условием будет **transactionCheck - содержит - false**:

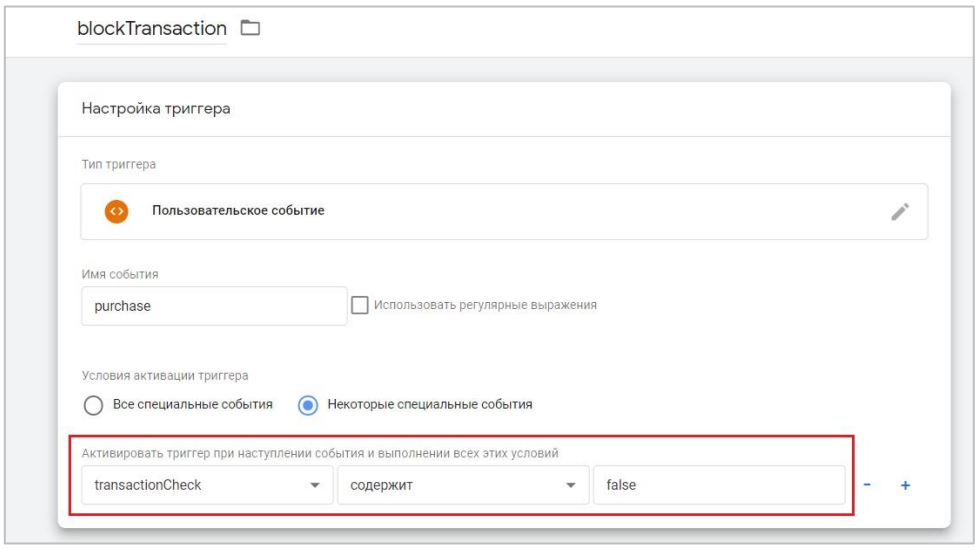


Рис. 917. Триггер блокировки

Тег транзакции не будет запускаться, если cookie уже содержит идентификатор текущей транзакции. Теперь необходимо добавить триггер блокировки к тегу транзакции, который отправляет данные в Google Analytics, в качестве исключения.

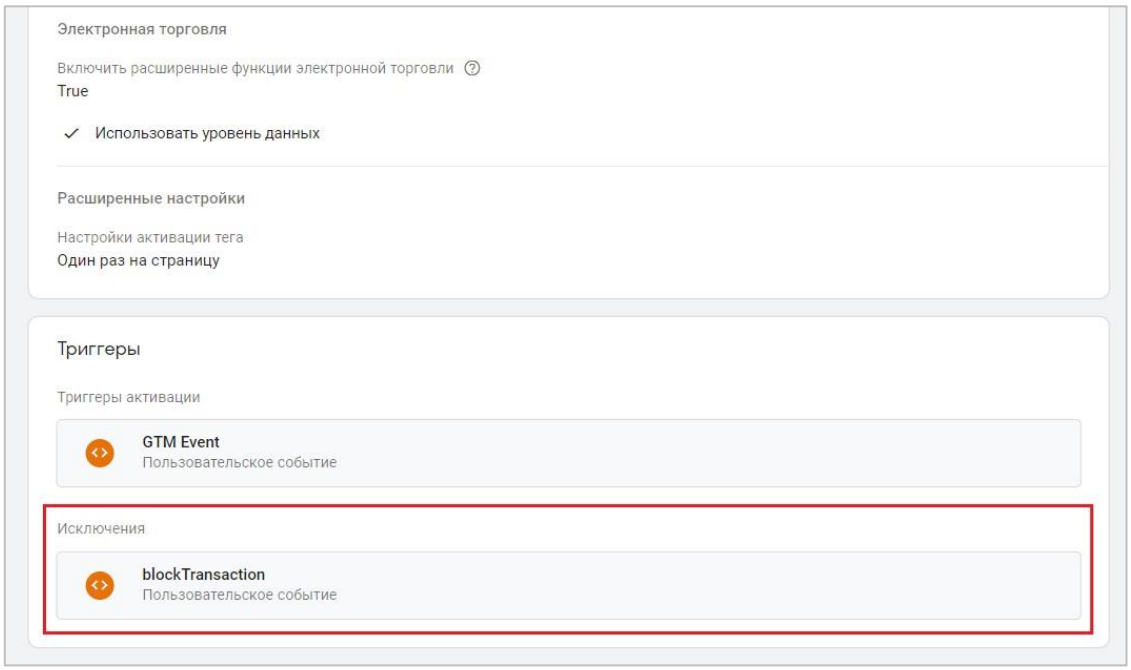


Рис. 918. Добавление триггера блокировки к тегу транзакции

Сохраняем все изменения. В режиме отладки Google Tag Manager мы можем проверить корректность настройки и посмотреть, будут ли передаваться дубль транзакции в аналитику.

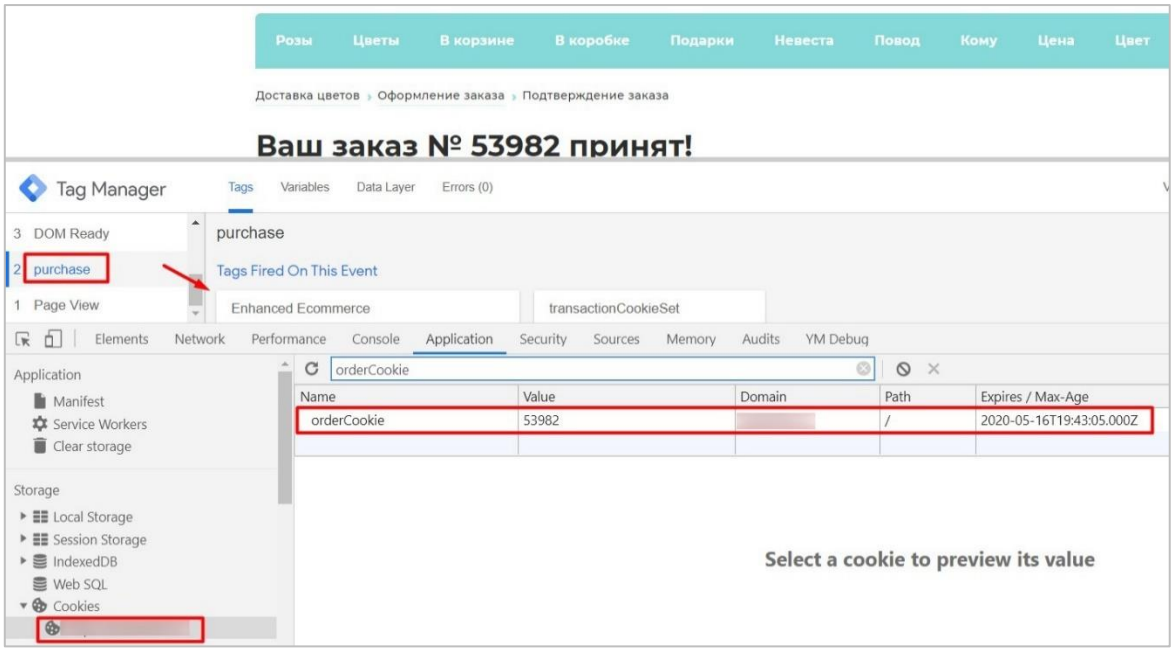


Рис. 919. Установка куки orderCookie со значением ID транзакции

Поскольку мы только-только настроили данные условия и куки **orderCookie** со значением текущей транзакции еще не было, тег транзакции сработал и событие снова передалось в Google Analytics. Но теперь, когда вы будете перезагружать страницу, или зайдете на нее еще раз в другом сеансе, событие purchase хоть и будет срабатывать, тег транзакции активироваться не будет, данные в Google Analytics не отправятся, поскольку куки **orderCookie** и **Transaction ID** совпадают, а это говорит о том, что транзакция уже была и это дубль.

Дубли конверсий для сайтов

Пример: мой тестовый сайт **graphanalytics.ru**. На нем есть форма с бесплатным аудитом и заказом услуг по настройке веб-аналитики, контекстной рекламы и индивидуальных консультаций. После заполнения и ее

отправки пользователя перенаправляет на страницу "Спасибо". У этой страницы есть конечный URL https://graphanalytics.ru/thank-you.html

Если бы мы хотели настроить отслеживание цели в Google Analytics, то просто бы создали цель на посещение этой страницы. Однако при попадании того же пользователя, но в другом сеансе, статистика будет искажаться. Достижение цели будет (+1), а новой заявки нет, поскольку пользователь не заполнял форму повторно. В результате, когда мы будем считать расход на рекламу и количество достигнутых целей, то получим некорректные данные как по стоимости конверсии, так и по конверсии сайта.

Именно и по этой причине я рекомендую использовать Google Tag Manager и настройку каких-либо взаимодействий пользователя с вашим сайтом через события, а не просмотры страниц, хоть и то, и то является хитами. Отслеживая события через GTM, вы можете более гибко задавать условия активации тегов, нежели к просмотрам страниц.

Я не буду снова описывать процесс настройки отслеживания дублей, поскольку он аналогичен. Отмечу лишь то, что я добавил в качестве отдельного тега типа **Пользовательский HTML** строчку кода с dataLayer.push, которая активирует событие **myLead** каждый раз на странице **/thank-you.html**.

Рис. 920. Активация события myLead на странице /thank-you

Логика простая - если человек попал на эту страницу, значит он оставил заявку на сайте. А настройка в GTM через dataLayer.push упрощает отслеживание дублей и становится схожа с настройкой отслеживания транзакций из предыдущего примера.

Поскольку на моей странице нет никакого идентификатора заявки или номера обращения (как в случае с ID транзакции), я могу задать произвольное значение с самого начала. Например, использовать константу со значением **success**.

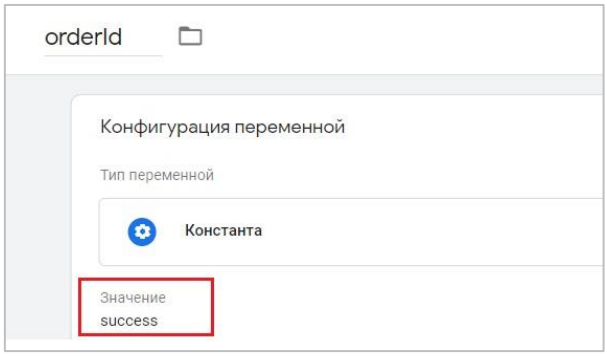


Рис. 921. Переменная Константа

Если у вас на странице благодарности присутствует идентификатор обращения, то извлеките это значение в пользовательскую переменную типа **Элемент DOM**.

Остальное все то же самое. Создается переменная типа **Основной файл cookie**, затем переменная типа **Собственный код JavaScript**, после тег типа **Пользовательский HTML**, затем триггер (в моем примере - **Пользовательское событие myLead**) и триггер блокировки, который добавляется в тег Universal Analytics, отвечающий за отправку данных в Google Analytics.

Все! Теперь событие будет передаваться в Google Analytics только один раз в течение установленного вами времени срока жизни куки, или пока сам пользователь не почистит данные. На него можно настроить цель-событие (конверсию), которая будет точнее отслеживать количество достигнутых целей. В завершении хотелось бы отметить, что приведенный способ отслеживания поможет снизить количество дублей транзакций и конверсий в Google Analytics. Но избавится от них на 100% все же не удастся.

Получение даты первого посещения пользователя

Дата первого посещения (сеанса) – это дата первого зарегистрированного взаимодействия пользователя с вашим контентом. Фактически, это тот момент, когда пользователю в Google Analytics был присвоен уникальный идентификатор пользователя (_ga). Чем же так полезен этот параметр?

С помощью данной метрики группируются когорты в отчете **Когортный анализ**. Благодаря ей мы можем проанализировать в разрезе заданной когорты любой из 14 доступных в Analytics показателей.

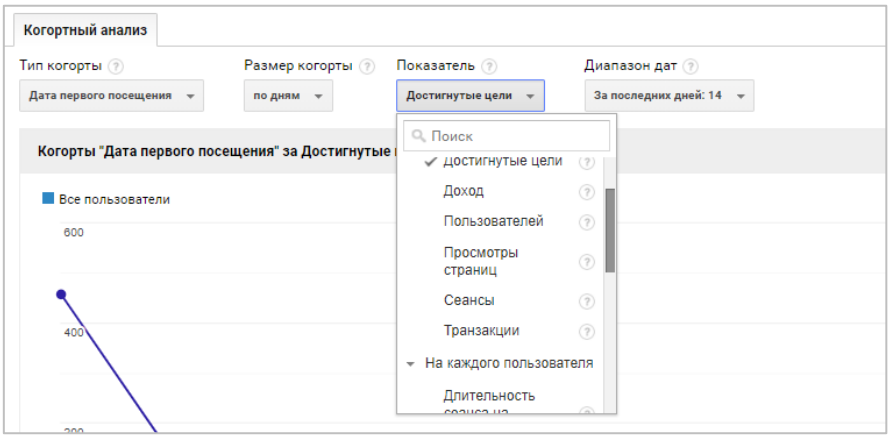


Рис. 922. Доступные показатели в когортном анализе

Когортный анализ используется при анализе LTV или удержание клиентов, рекламных кампаниях, когда необходимо отслеживать окупаемость рекламных каналов, в аналитике мобильных приложений (так называемый **«Retention Rate %»**), для SaaS – проектов, когда дата регистрации и использование пробного периода не равна дате покупки платной подписки, в том же e-commerce и многих других областях.

Разберем классический случай. У вас есть интернет-магазин по продаже какой-либо продукции, и вы вкладываете деньги в рекламу, допустим, Яндекс.Директ и Google Ads. Вы понимаете, что человек может купить у вас не сразу, а спустя несколько дней. Но смотря в разрезе каждого дня вы видите, что стоимость ежедневных затрат на рекламу больше дохода от продаж. И вы начинаете паниковать.

Давайте построим таблицу, в которой будет видно, как пользователи регистрировали/посещали ваш сайт впервые, и через какое время они делали у вас первую покупку и на какую сумму. Например, у нас получились такие данные по месяцу работ:

Дата регистрации на сайте	Дата 1 покупки	Доход
01.07.2018	05.07.2018	4000₽
01.07.2018	06.07.2018	12000₽
01.07.2018	01.07.2018	4545₽
02.07.2018	02.07.2018	2342₽
02.07.2018	02.07.2018	2341₽
02.07.2018	31.07.2018	43577₽
02.07.2018	14.08.2018	2346₽
02.07.2018	12.07.2018	123₽
03.07.2018	05.07.2018	8816₽
03.07.2018	05.07.2018	8373₽
04.07.2018	05.07.2018	7929₽
04.07.2018	05.07.2018	7486₽
04.07.2018	05.07.2018	7043₽
04.07.2018	05.07.2018	6599₽
04.07.2018	05.07.2018	6156₽
04.07.2018	10.07.2018	5713₽
04.07.2018	10.07.2018	5270₽
04.07.2018	10.07.2018	4826₽
05.07.2018	17.07.2018	4383₽
05.07.2018	17.07.2018	3940₽

Рис. 923. Дата регистрации и дата 1 покупки

Из таблицы видно, что есть люди, которые впервые зашли на сайт 1 июля (это и есть **дата первого сеанса**), но некоторые из них купили только 5 и 6 июля (не все покупают в тот же день). 2 июля два человека купила сразу же, один – 31 июля, другой – 12 июля, а один аж в следующем месяце – 14 августа. И т.д.

Теперь мы можем построить зависимость дата регистрации на сайте от даты 1 покупки. Сделаем это с помощью сводной таблицы, и разложим метрики по строкам и столбцам. Получим:

Сумма по полю Доход	Дата 1 покупки																					
	июл																	авг		сен	окт	Общий итог
Дата регистрации на сайте	01.июл	02.июл	05.июл	06.июл	08.июл	10.июл	12.июл	17.июл	21.июл	22.июл	23.июл	24.июл	25.июл	26.июл	29.июл	30.июл	31.июл	14.авг	27.авг	01.сен	31.окт	
01.07.2018	4545		4000	12000																		20545
02.07.2018		4683					123										43577	2346				50729
03.07.2018			17189																			17188,5
04.07.2018			35214			15809																51022
05.07.2018			8272,4		1723			8323	2166	3496,3		1279,8					2166,4					27426,9
06.07.2018																	1723,1					1723,1
07.07.2018												1279,8										1279,8
08.07.2018												2609,7										2609,7
09.07.2018												2166,4										2166,4
10.07.2018															1723,1							1723,1
11.07.2018																1279,8						1279,8
12.07.2018																8372,6						8372,6
13.07.2018																7929,3						7929,3
14.07.2018																	7486					7486
15.07.2018															7042,7							7042,7
16.07.2018																6599,4						6599,4
17.07.2018																6156,1						6156,1
18.07.2018																5712,8						5712,8
19.07.2018																	5269,5					5269,5
20.07.2018									4826	8372,6	7929											21128,1
21.07.2018												7486										7486
22.07.2018													7043									7042,7
23.07.2018														6599,4								6599,4
24.07.2018														6156,1								6156,1
25.07.2018																	5712,8					5712,8
26.07.2018														11869	18468		31223,8					61561
27.07.2018																7929,3						7929,3
28.07.2018																7486						7486
29.07.2018																	7042,7					7042,7
30.07.2018																	18468,3					18468,3
31.07.2018																			19698	4776	3446	27920,3
Общий итог	4545	4683	64674	12000	1723	15809	123	8323	6993	11869	7929	14822	7043	24624	27234	51465	122670	2346	19698	4776	3446	416794,4

Рис. 924. Сводная таблица в Excel

Теперь разберем одну когорту – людей, которые зашли на наш сайт с рекламного трафика 1 июля:

Сумма по полю Доход	Дата 1 покупки																					
	июл																	авг		сен	окт	Общий итог
Дата регистрации на сайте	01.июл	02.июл	05.июл	06.июл	08.июл	10.июл	12.июл	17.июл	21.июл	22.июл	23.июл	24.июл	25.июл	26.июл	29.июл	30.июл	31.июл	14.авг	27.авг	01.сен	31.окт	
01.07.2018	4545		4000	12000																		20545
02.07.2018		4683					123										43577	2346				50729
03.07.2018			17189																			17188,5

Рис. 925. Когорта - 1 июля

Всего это когорта принесла нам дохода на общую сумму 20 545 руб. Однако если бы вы смотрели в разрезе одного дня (Доход/Расход на рекламу), вы могли бы видеть убыток. Но благодаря когортному анализу вы можете оценить потенциальный вклад пользователей в долгосрочной перспективе.

Если кратко, то так и строится когортный анализ. Однако в Google Analytics нельзя анализировать какую-то одну цель. Для построения когорт доступен общий показатель **Достигнутые цели**.

Также дату первого посещения можно использовать при создании сегментов и анализе эффективности рекламных кампаний.

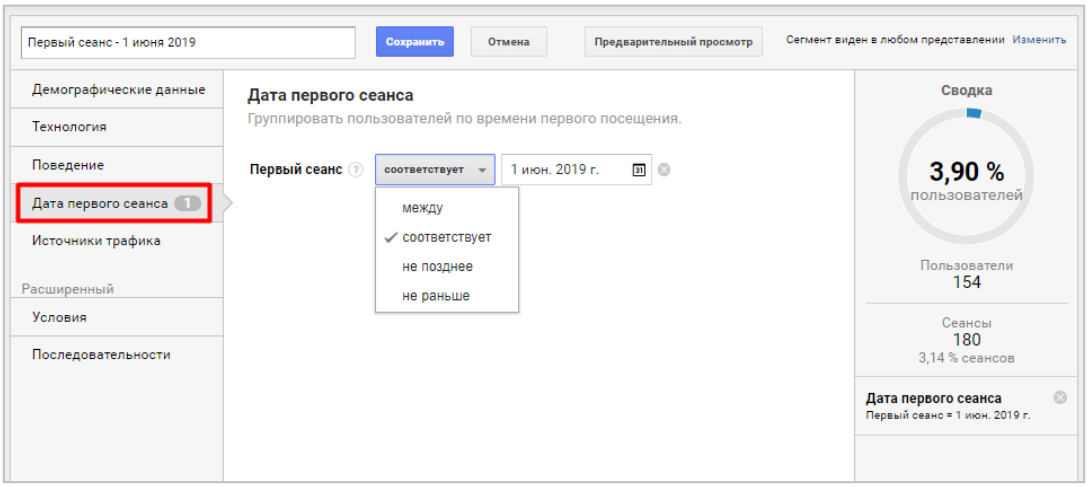


Рис. 926. Сегмент - Дата первого сеанса

В специальных и стандартных отчетах нельзя добавлять дату первого посещения в качестве параметра. Но почему бы не передавать ее в Google Analytics в качества пользовательского параметра? Просто извлечь дату создания файла cookie из **_ga**, преобразовать ее в понятный формат ДД-ММ-ГГГГ, а затем использовать в любых измерениях?

Кроме этого, мы сможем легко сопоставить дату первого посещения пользователя даже в том случае, если затем в представлении наложили фильтры на дублирующие источники трафика.

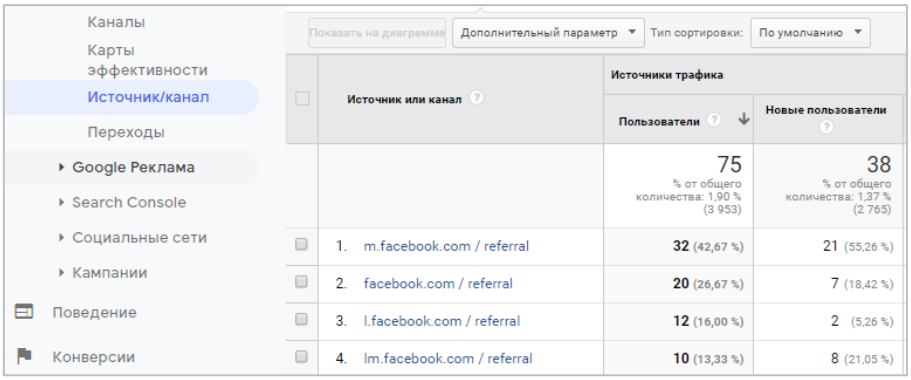


Рис. 927. Задубливание источников трафика

Извлекать дату первого посещения из cookie будем с помощью Google Tag Manager. Последовательность действий следующая:

- создать пользовательский параметр с областью действия **Пользователь**;
- запомнить индекс пользовательского параметра;

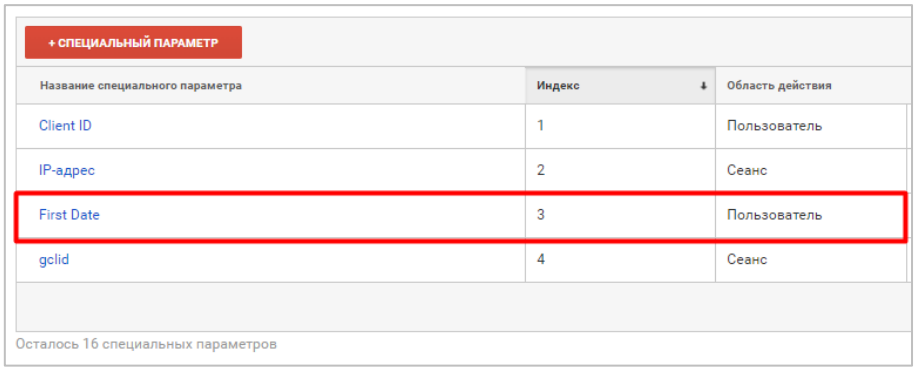


Рис. 928. Индекс специального параметра

- перейти в GTM и создать пользовательскую переменную типа **Собственный код JavaScript**
- вставить код и сохранить изменения:

```
  function(){
  var regex = new RegExp("_ga=([^;]+)");
  var value = regex.exec(document.cookie);
  var cookieCreationDate = (value != null) ? new
Date(value[1].split('.')[3]*1000) : undefined;

  if(typeof(cookieCreationDate)!=="undefined")
  {

  var year = cookieCreationDate.getFullYear().toString();
  var month = (cookieCreationDate.getMonth()+1).toString();
  var day = cookieCreationDate.getDate().toString();
  cookieCreationDate = (day[1]?day:"0"+day[0]) + "-" +
(month[1]?month:"0"+month[0]) + "-" + year;
  }

  return cookieCreationDate;
  }
```

Код в Google Tag Manager:

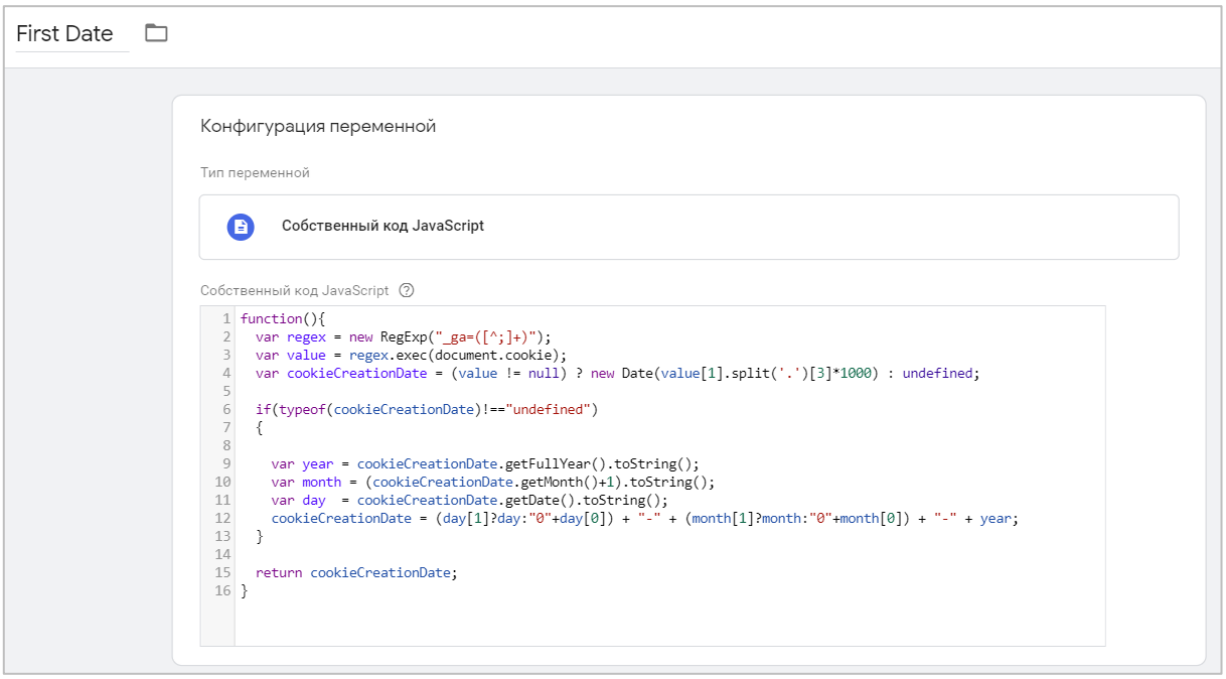


Рис. 929. Переменная Собственный код JavaScript

- перейти в тег Universal Analytics и в дополнительных настройках, в специальных параметрах добавить индекс и значение параметра нашей пользовательской переменной;

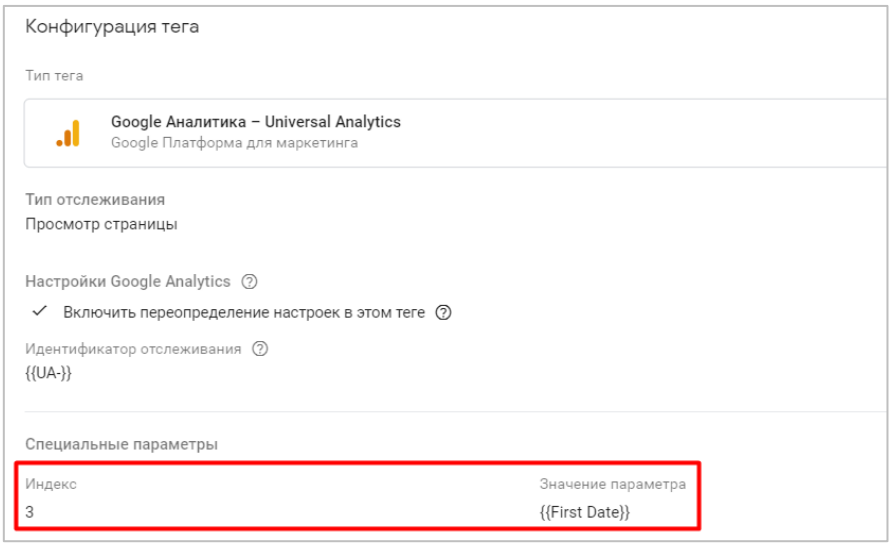


Рис. 930. Добавление специального параметра

На этом все! Перед публикацией изменений можно проверить корректность передачи данных в режиме отладки.

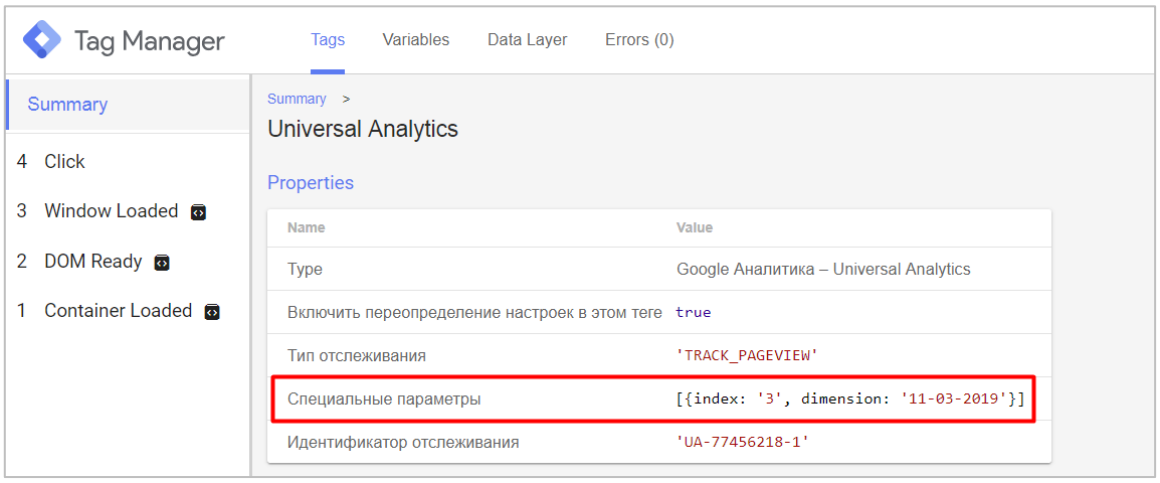


Рис. 931. Проверка в режиме отладки

В моем блоге дата первого посещения (когда был создан Client ID) числится 11 марта 2019 года. Но так ли это?

Проверить очень просто. В консоли разработчика браузера вводим команду **document.cookie**, затем копируем уникальный идентификатор пользователя:

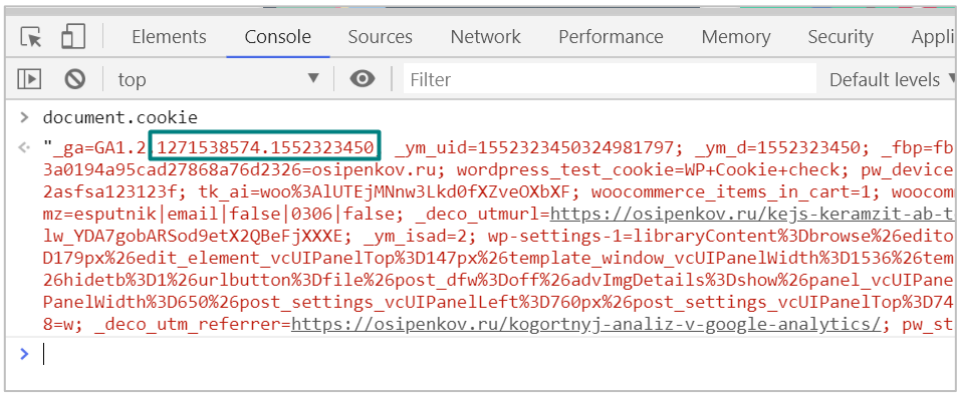


Рис. 932. Копирование идентификатора клиента

Далее идем в Google Analytics в отчет **Статистика по пользователям**. В правом углу в поиске вставляем наш идентификатор и переходим внутрь.

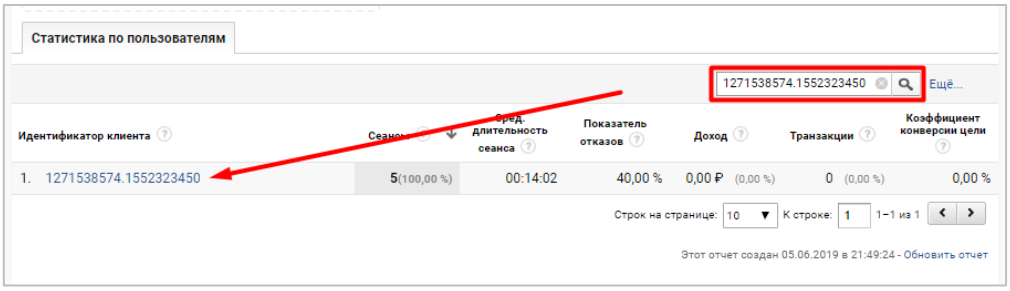


Рис. 933. Поиск записи по идентификатору клиента

Слева будет отображена основная информация по пользователю – идентификатор клиента, источник, канал, кампания, тип устройства, платформа и т.д. Под графой **Источник** будет отображена **Дата**. Это и есть дата моего первого посещения osipenkov.ru из данного браузера.

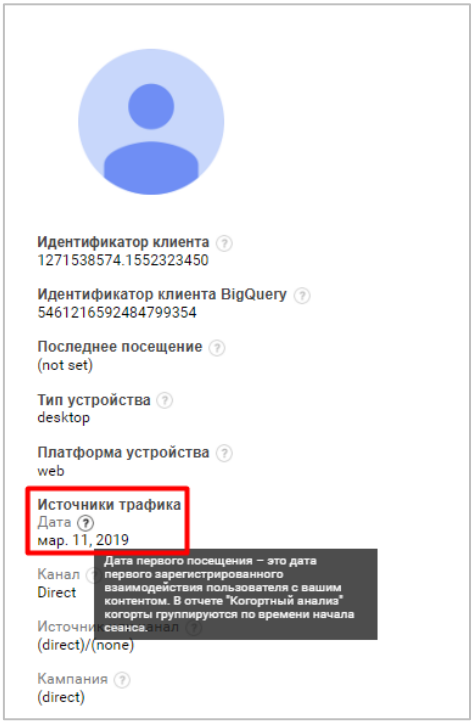


Рис. 934. Карточка пользователя

Она полностью совпадает с той информацией, что мы видели в режиме предварительного просмотра в Google Tag Manager. Теперь можно перейти в любой отчет и добавить пользовательский параметр в качестве дополнительного.

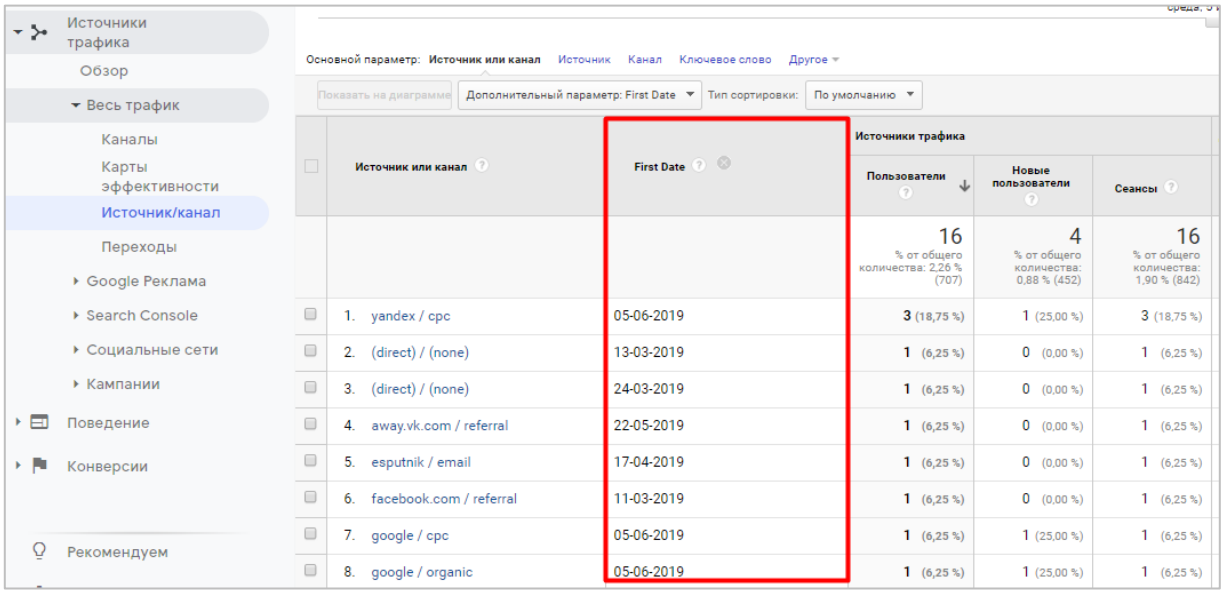


Рис. 935. Пример отчета с дополнительным параметром First Date

Или же создать свой собственный. Например, из рисунка ниже видно, что пользователь выполнил конверсию и заказал бесплатный анализ проекта 5 июня 2019 года, а дата первого посещения моего блога была 13 марта. Прошло 2,5 месяца прежде, чем этот человек решился оставить заявку на аудит.

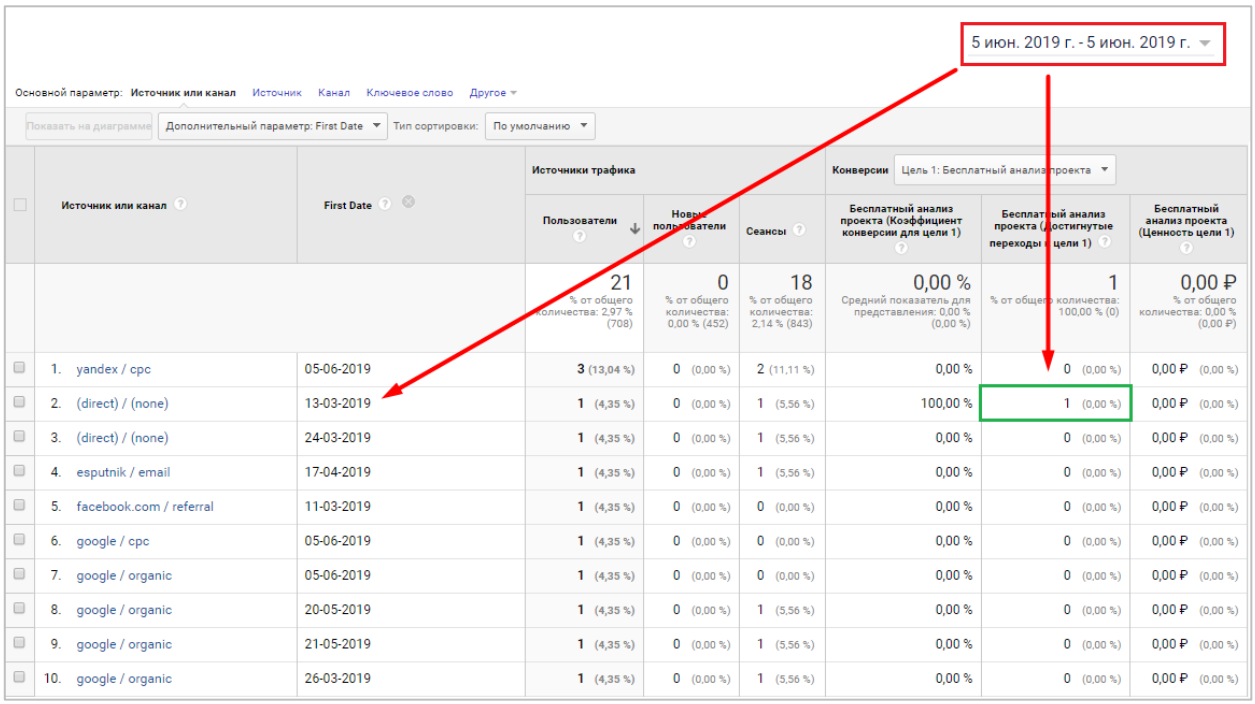


Рис. 936. Анализ в разрезе конверсий

Таким образом, дата первого посещения, извлеченная из cookie с помощью GTM, и добавленная в кастомный параметр Google Analytics, дает нам возможность строить и анализировать когорты по каждой из конверсий, а также проще определять временной период принятия решения о покупке потенциального клиента.

Отслеживание «кликов ярости» (Rage Clicks)

Клики ярости (Rage Clicks) — это клики пользователей, которые многократно щелкают, нажимают на элемент(ы) на вашем сайте или в мобильном приложении, потому что им что-то не нравится, или они не могут понять, что делать дальше на странице/экране. В этой статье познакомимся со способом отслеживания яростных кликов с помощью GTM.

Rage Clicks — это все равно, что бить беспорядочно по клавиатуре, когда пользователь чем-то разочарован. Клики ярости сигнализируют о том, что ваш сайт не отреагировал так, как хотел или думал ваш клиент. Например, у вас текст подчеркнут на сайте, но не имеет ссылки. А пользователь думает, что, кликнув по нему, он перейдет на другую страницу. Такие клики еще называют *мертвыми (Death Clicks)*, поскольку они ни к чему не приводят.

Рис. 937. Пользователь думает, что это ссылка, и кликает-кликает-кликает

Или при клике на миниатюру изображения человек предполагает, что она откроется в полном размере. Но этого не происходит.

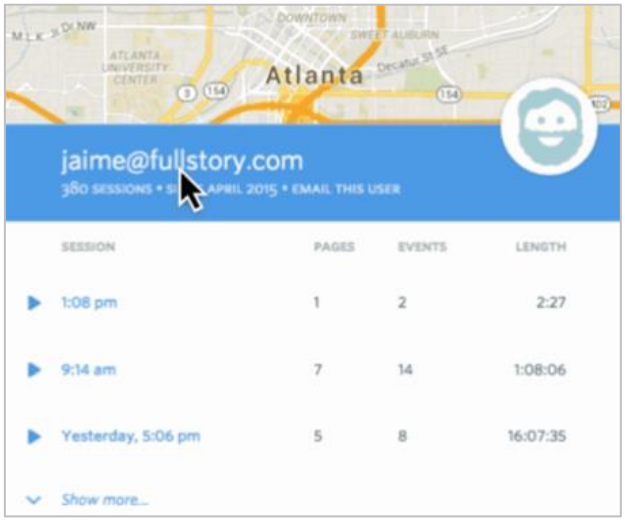


Рис. 938. Клики по картинке и почте - пример с сайта fullstory.com

Аналогично с электронной почтой. Пользователь думает, что его перенаправит в почтовый мессенджер, но e-mail на сайте не является кликабельным, а всего лишь текст. Иногда видео могут не загружаться сразу. Терпение быстро кончается, и мы начинаем чаще нажимать на кнопку мыши.

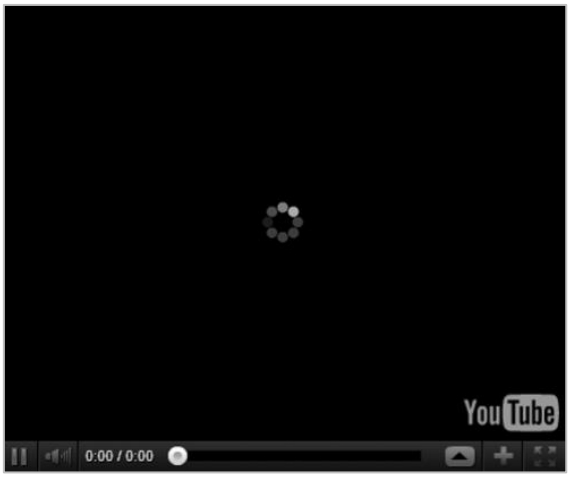


Рис. 939. Бесконечная загрузка видео - пример с сайта fullstory.com

Но самое страшное для нас (маркетологов и владельцев бизнеса), когда пользователь хочет оставить заявку или совершить заказ, нажимает на кнопку **Отправить** и... ничего не происходит.

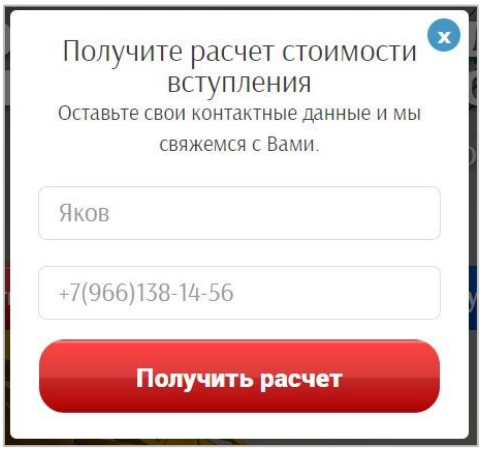


Рис. 940. Кликаем-кликаем, а форма не отправляется и не отправляется

Разработчик не сознается, что недавно вносил изменения на сайте и не проверил отправку формы. А мы, кроме как посмотреть Вебвизор в Яндекс.Метрике, больше никаких зацепок не имеем. И то не всегда. Итог - теряем деньги.

Упоминая о **Rage Clicks / Death Clicks**, хотелось бы еще сказать пару слов об **Error Clicks** и **Thrashed Cursor**.

Клики ошибок (Error Clicks) - клики, которые возникают, например, когда пользователь заполняет форму, и что-то указывает неверно (формат почтового индекса, номер кредитной карты, электронную почту и т.д.). Ему выскакивает сообщение об ошибке, но он не понимает, что оно значит, поэтому продолжает нажимать на кнопку **Отправить**, **Далее** и т.д. А если так и не разберется, то просто покинет сайт.

Побитый курсор (Thrashed Cursor) - термин, означающий хаотичное, беспорядочное перемещение мыши. Движение мыши пользователя, например, по кругу, может свидетельствовать о том, что человек сбит с толку, потерян или ожидает загрузки страницы.

Отслеживание таких действий является частью UX-аналитики. На рынке существуют решения, способные отслеживать все вышеперечисленное. Одним из самых популярных является сервис **fullstory.com**.

Чтобы настроить отслеживание кликов ярости в GTM, необходимо:

- создать тег типа **Пользовательский HTML** и вставить в него код (см. приложение):

```
<script>
  if ( typeof(jQuery) === 'function' ) {
    jQuery(document).ready(function($) {
        jQuery.fn.extend({
            getPath: function() {
                var path, node = this;
                while (node.length) {
                    var realNode = node[0],
                        name = realNode.localName;
                    if (!name) break;
                    name = name.toLowerCase();
                    var parent = node.parent();
                    var sameTagSiblings = parent.children(name);
                    if (sameTagSiblings.length > 1) {
                        var allSiblings = parent.children();
                        var index = allSiblings.index(realNode) + 1;
                        if (index > 1) {
                            name += ':nth-child(' + index + ')';
                        }
                    }
                    path = name + (path ? '>' + path : '');
                    node = parent;
                }
                return path;
            }
        });
        // Number of rage clicks
        var no_of_clicks = 3;
        //Time interval - 3 for 3 secs, 4 for secs and likewise
        var time = 2;
        var click_events = [];
        //internal variables
        var possible_click = 3;
        var radius = 100;
        function detectXClicks(count, interval) {
            var last = click_events.length - 1;
            var time_diff = (click_events[last].time.getTime() -
click_events[last - count + 1].time.getTime()) / 1000;
            //returns false if it event period is longer than 5 sec
            if (time_diff > interval) return null;
            //check click distance
            var max_distance = 0;
            for (i = last - count + 1; i < last; i++) {
```

```
                    for (j = i + 1; j <= last; j++) {
                        var distance =
Math.round(Math.sqrt(Math.pow(click_events[i].event.clientX -
click_events[j].event.clientX, 2) + Math.pow(click_events[i].event.clientY -
click_events[j].event.clientY, 2)));
                        if (distance > max_distance) max_distance = distance;
                        if (distance > radius) return null;
                    }
                }
                var result = {
                    count: count,
                    max_distance: max_distance,
                    time_diff: time_diff
                }
                return result;
            }
            function removeUsedClickPoints(count) {
                click_events.splice(click_events.length - count, count);
            }
            $("body").click(function(event) {
                click_events.push({
                    event: event,
                    time: new Date()
                });
                //remain only required number of click events and remove left of
them.
                if (click_events.length > possible_click) {
                    click_events.splice(0, click_events.length - possible_click);
                }
                //detect 3 click in 5 sec
                if (click_events.length >= 3) {
                    var result = detectXClicks(no_of_clicks, time);
                    if (result != null) {
                        var path = $(event.target).getPath();
                        //console.log('Rage Click: ' + JSON.stringify(result));
                        dataLayer.push({
                            'event': 'rage_click',
                            'rc_element': path,
                            'rc_count': result.count,
                            'rc_max_distance': result.max_distance,
                            'rc_time_diff': result.time_diff
                        });
                        removeUsedClickPoints(3);
                    }
                }
            });
        });
    }
    </script>
```

В Google Tag Manager это выглядит так. Триггер активации - **All Pages (Все страницы)**:

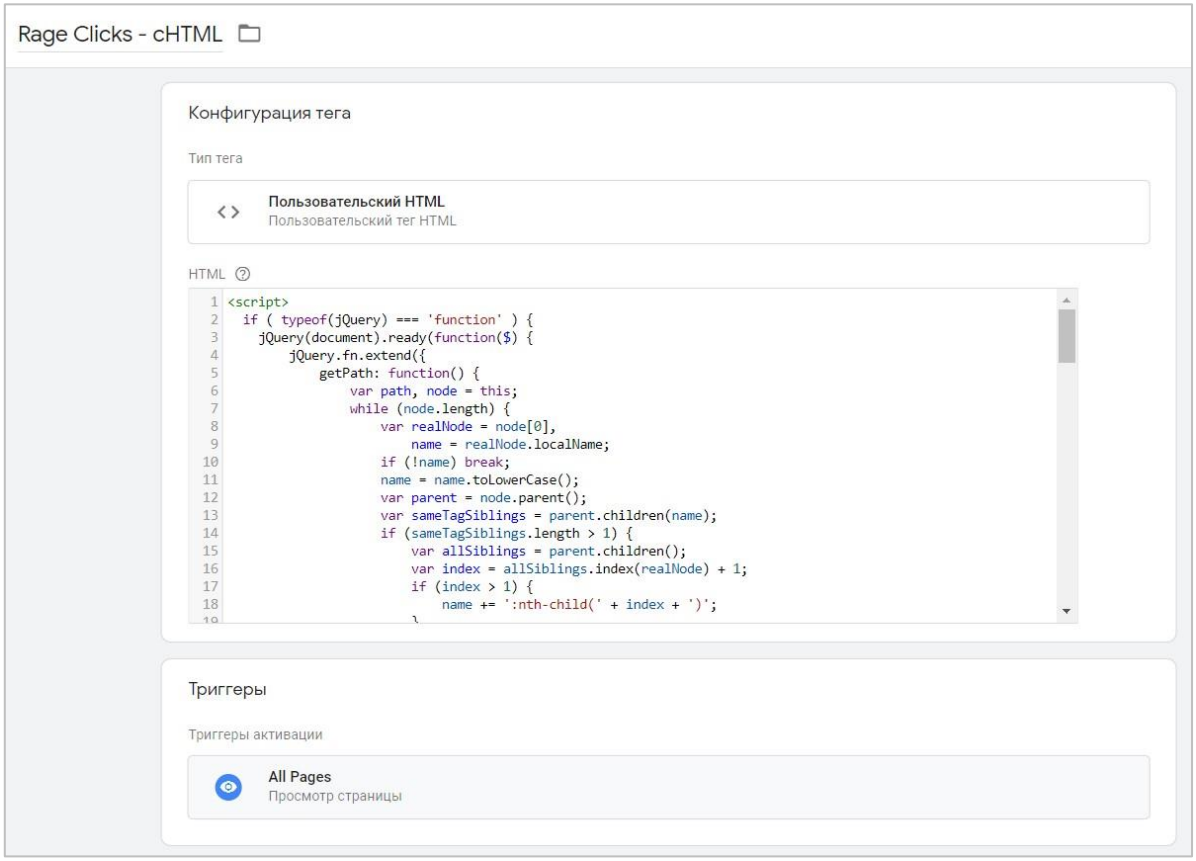


Рис. 941. Код Rage Clicks

Примечание: на отслеживаемых страницах должен быть загружен jQuery.

Этот тег отвечает за обнаружение кликов ярости и отправку пользовательского события **rage_click** в dataLayer. Есть несколько переменных, которые вы можете изменить в зависимости от задач:

- **no_of_clicks** - сколько кликов нужно, чтобы считать это кликом ярости. По умолчанию - **3**;
- **time** - за какой период времени должны произойти все щелчки. По умолчанию - **2 секунды, 2**;
- **radius** - в каком радиусе должны происходить клики. По умолчанию - **100 пикселей, 100**.

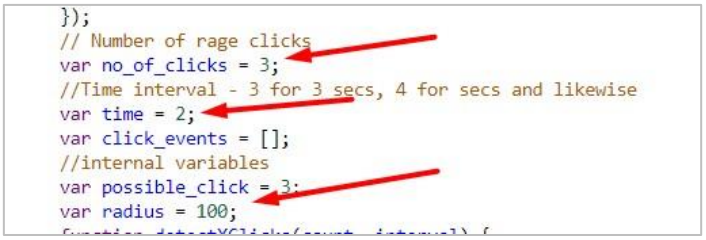


Рис. 942. Изменение значений переменных в скрипте

Следующие переменные помещаются в уровень данных при обнаружении кликов ярости:

- **event** - rage_click, пользовательское событие, которое мы можем использовать в качестве триггера активации;
- **rc_element** - селектор элемента, по которому произошло нажатие;
- **rc_count** - количество кликов;
- **rc_max_distance** - максимальное расстояние между кликами;
- **rc_time_diff** - период от первого до последнего клика;

```
dataLayer.push({
    'event': 'rage_click',
    'rc_element': path,
    'rc_count': result.count,
    'rc_max_distance': result.max_distance,
    'rc_time_diff': result.time_diff
});
```

Рис. 943. Уровень данных (dataLayer)

Автор рекомендует не передавать все переменные в инструменты аналитики, а использовать только те, которые необходимы. Для этого создайте пользовательские переменные типа **Переменная уровня данных** с соответствующим именем. Например, для **rc_element** она будет выглядеть так:

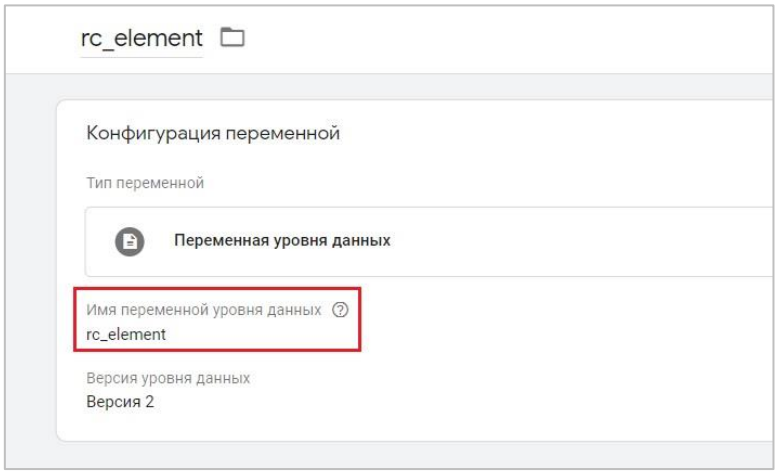


Рис. 944. Переменная уровня данных

Аналогично можно проделать с другими переменными: **rc_count**, **rc_max_distance** и **rc_time_diff**.

- создайте триггер типа **Пользовательское событие** с именем **rage_click**:

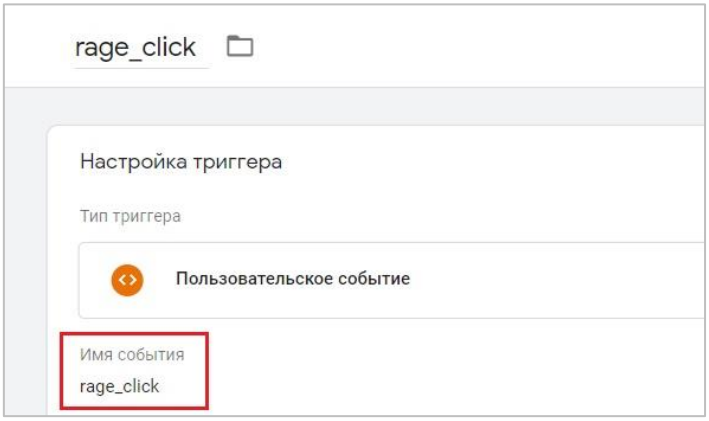


Рис. 945. Триггер Пользовательское событие

- создайте тег типа **Google Аналитика - Universal Analytics** с типом **Событие**

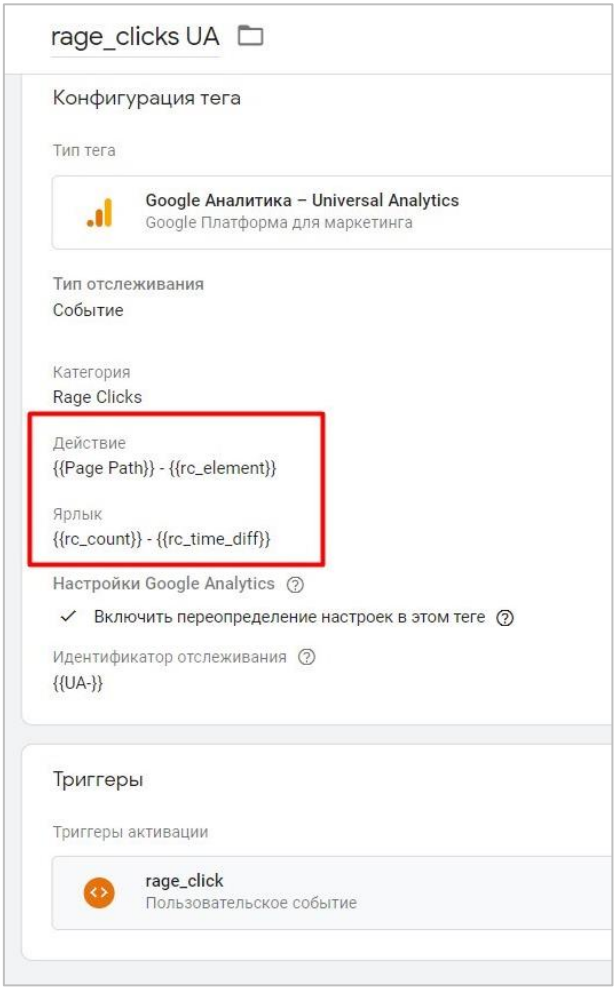


Рис. 946. Настройка тега Google Аналитика – Universal Analytics

В этом примере в **Категории** события я использовал название **Rage Clicks**, в **Действие** - встроенную переменную **Page Path**, чтобы знать, на какой странице бы совершен клик ярости, и **rc_element**, чтобы определить селектор элемента, по которому произошло нажатие. В **Ярлыке** события я добавил через дефис дополнительные переменные **rc_count** и **rc_time_diff**, чтобы передавать в Google Analytics количество кликов и время от первого до последнего клика.

Условие активации - триггер пользовательского события **rage_click**. Сохраните настройки тега.

Проверить корректность передачи информации в Google Analytics можно с помощью режима предварительного просмотра.

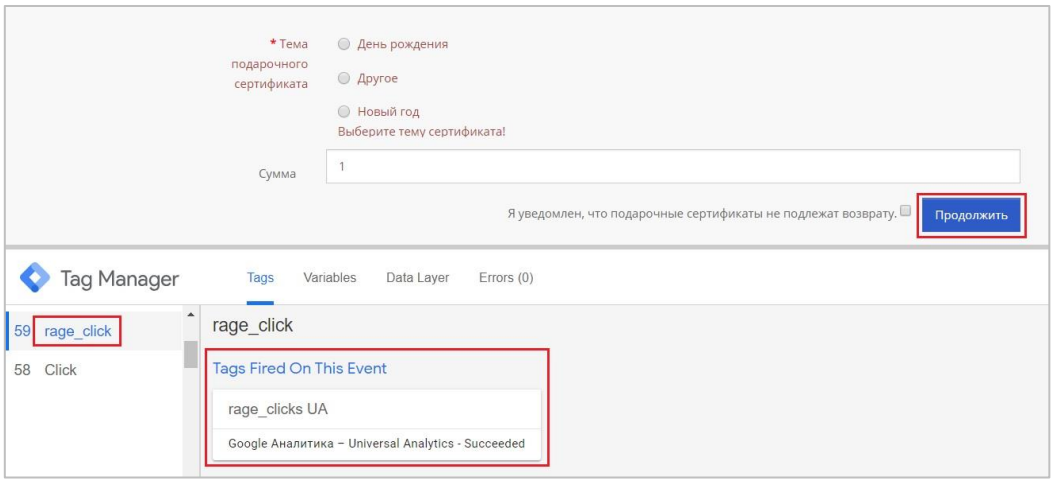


Рис. 947. Проверка события в режиме предварительного просмотра

В примере выше я кликал по кнопке **Продолжить** не заполнив обязательные поля формы. Как видим, пользовательское событие **rage_click** сработало, а тег Universal Analytics был активирован. В отчете **В режиме реального времени** можно увидеть переданные данные:

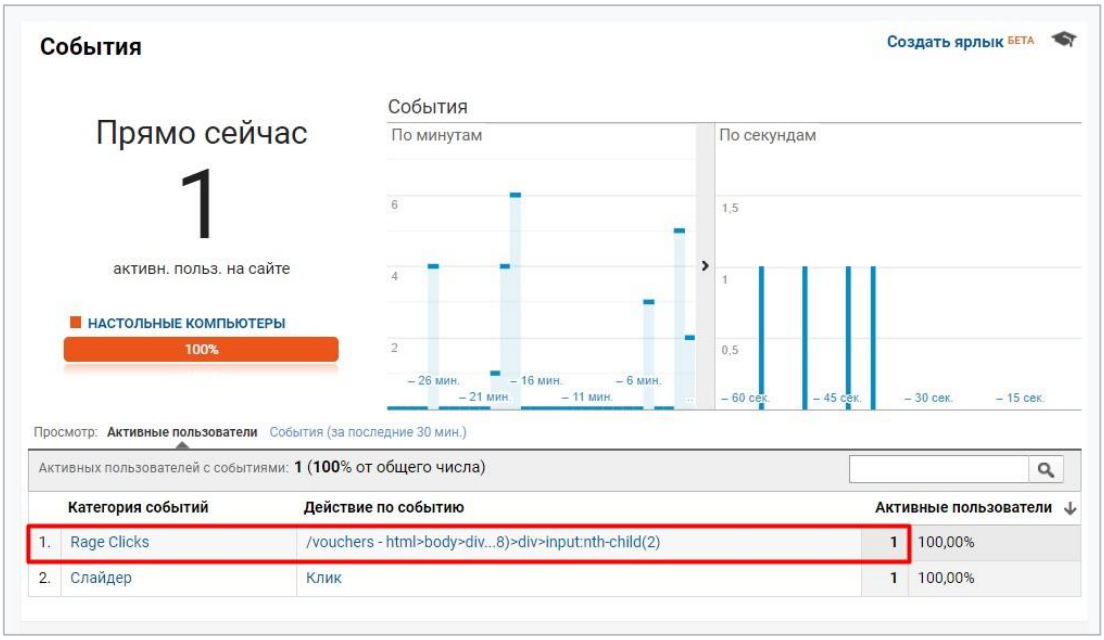


Рис. 948. Отчет В режиме реального времени

Что делать дальше? Посмотреть клики ярости можно в отчете **Поведение – События – Лучшие события**.

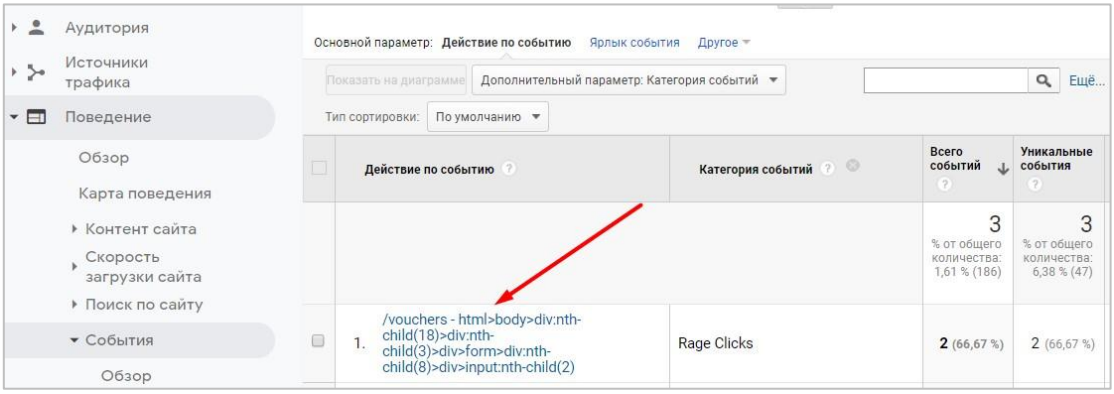


Рис. 949. Отчет по событиям

Затем можно скопировать селектор и перейти на ту страницу, на которой пользователь совершил клик ярости. В моем примере это **/vouchers**. Воспользовавшись расширением для браузера **CSS Selector Tester**, найдем наш элемент на странице:

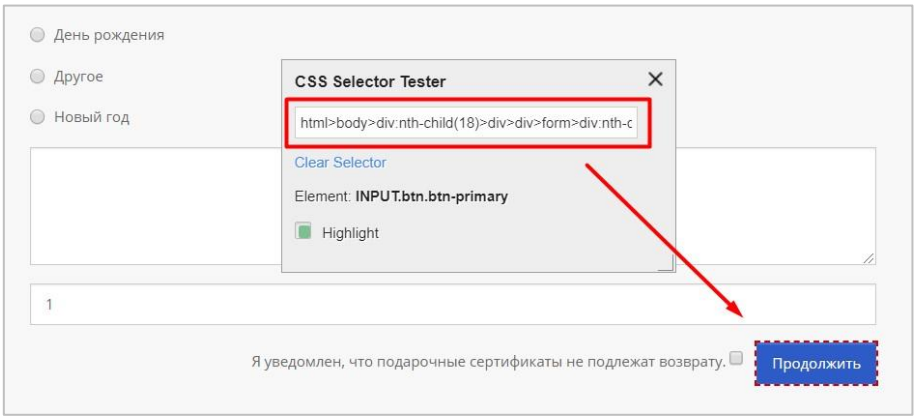


Рис. 950. Найденный элемент на странице

Элемент на странице найден. Теперь вы точно знаете, что вызвало проблемы у пользователей!

Отслеживание мобильных пользователей

Сейчас практически у каждого сайта есть адаптивная/мобильная версия, которая отличается по виду в зависимости от типа устройства – десктоп, мобильный телефон, планшет. На каждой из этих версий могут быть расположены различные элементы. Например, в мобильной версии развернутое меню может превратиться в бургер:

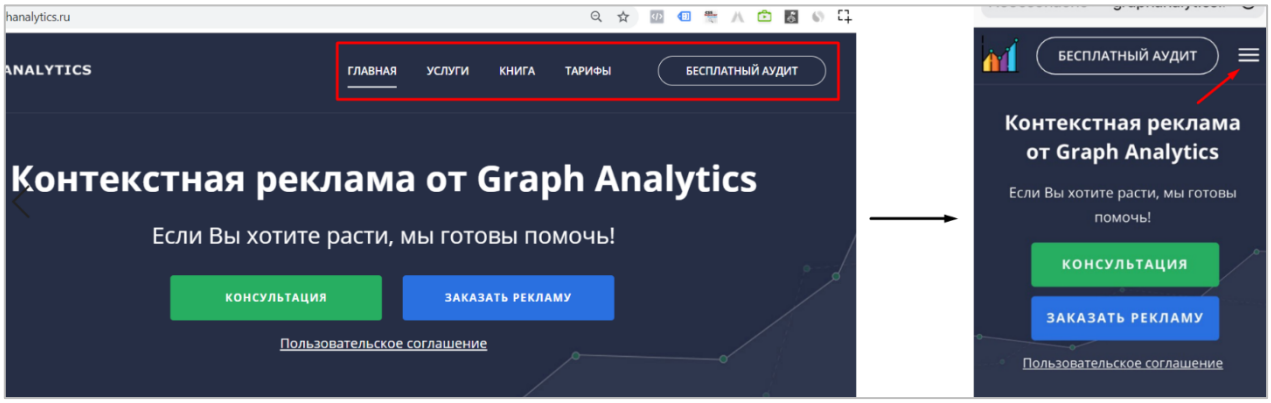


Рис. 951. Пример отображения сайта на компьютере и телефоне

Или какие-то кнопки активны на десктопной версии, а на планшете они скрыты. В общем, вариантов много. В конечном виде настольная версия сайта может значительно отличаться от портативных девайсов. Это усложняет отслеживание событий таких объектов.

А что делать, если нам необходимо отслеживать определенные действия на телефонах, чтобы потом таким пользователям показывать уникальное предложение? Если мы не определим тип устройства, с которого был осуществлен переход к нам на сайт, то и условие активации (триггер) мы никак не запустим. Запустим, конечно, силами разработчиков, но не собственными.

А чтобы это настроить самостоятельно с помощью GTM, необходимо создать пользовательскую переменную, которая отличала бы мобильных пользователей от настольных компьютеров. Сделать это можно различными способами. Я рассмотрю 2:

1. с помощью JavaScript кода, который определяет тип устройства в зависимости от полученной ширины экрана;
2. с помощью JavaScript кода, который возвращает булево значение *true (истина)* или *false (ложь)*.

Разберем более подробно.

Способ №1

Создайте пользовательскую переменную типа **Собственный код JavaScript** и вставьте туда следующий код:

```
  function () {
    var width = window.innerWidth,
      screenType;
    if (width <= 768) { screenType = "mobile"; } else if (width > 768 && width
<= 992) {
      screenType = "tablet";
    } else {
      screenType = "desktop";
    }
    return screenType;
  }
```

В Google Tag Manager это выглядит так:

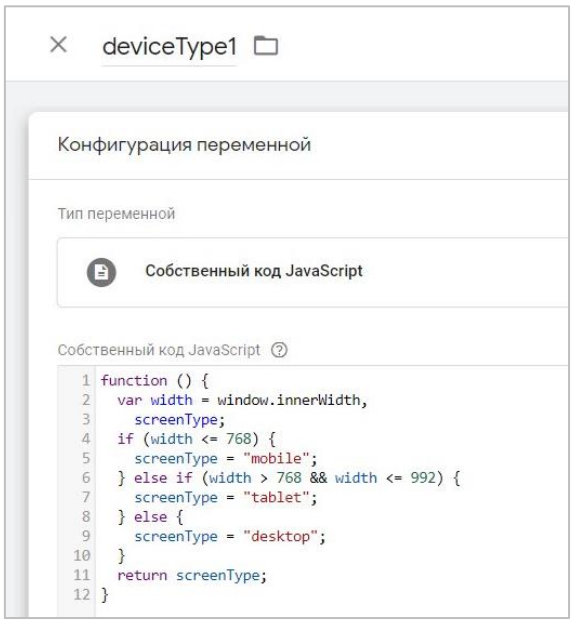


Рис. 952. Переменная deviceType1

Комментарий по коду: мы объявляем переменную **width**, в которую заносим данные по объекту **window**, с помощью которого сможем получить размеры рабочей области окна браузера. За это отвечает свойство **innerWidth**. И объявляем переменную **screenType**.

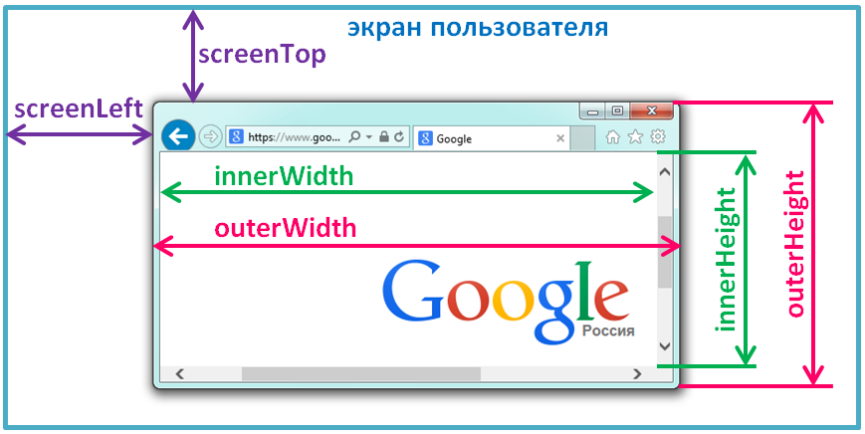
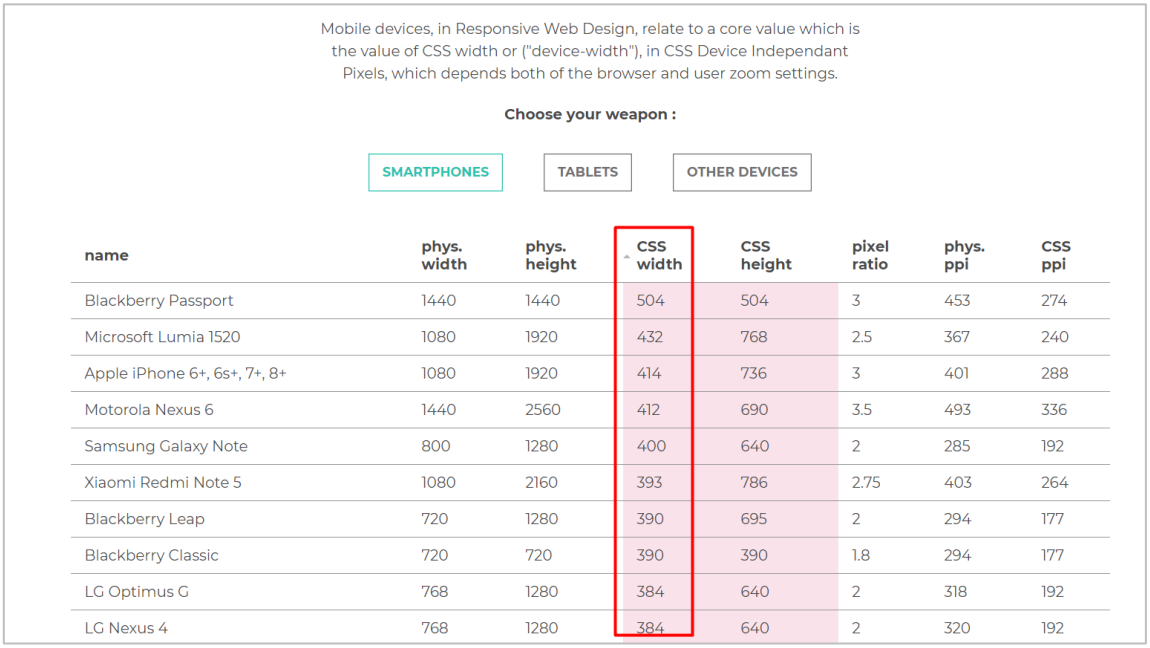


Рис. 953. innerWidth

Подробнее про объект window, свойства innerWidth и outerWidth и читайте в этом материале (см. приложение).

Затем запускаем условие (цикл), при котором происходит сопоставление размеров окна браузера пользователя. Если его ширина меньше или равна **768** пикселям, то в переменную screenType заносится **mobile**. В противном случае спускаемся на проверку следующего условия – если размеры рабочей области окна браузера больше **768** пикселей и меньше или равно **992**, то в screenType заносится **tablet**. И в противном случае если ни одно из этих условий не сработало, то значит это **desktop** (ПК).

Почему 768 и 992? Все дело в размерах экранов устройств. Чтобы лучше понять, перейдите на сайт **mydevice.io** и в таблице **Smartphones** отсортируйте по убыванию столбец **CSS width.**



Mobile devices, in Responsive Web Design, relate to a core value which is the value of CSS width or ("device-width"), in CSS Device Independant Pixels, which depends both of the browser and user zoom settings.

Choose your weapon :

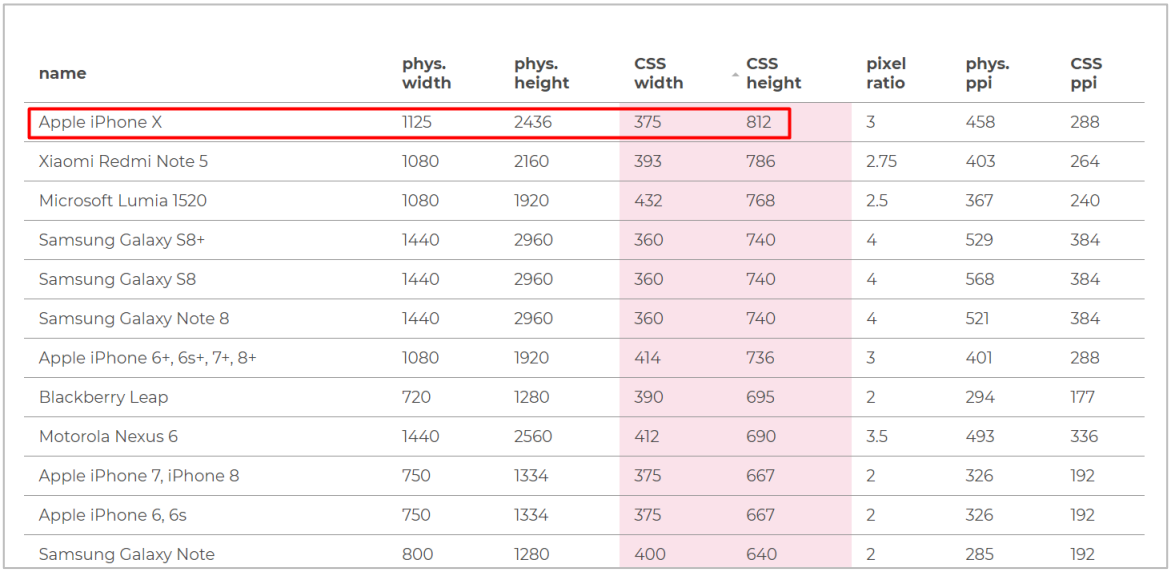
SMARTPHONES TABLETS OTHER DEVICES

name	phys. width	phys. height	CSS width	CSS height	pixel ratio	phys. ppi	CSS ppi
Blackberry Passport	1440	1440	504	504	3	453	274
Microsoft Lumia 1520	1080	1920	432	768	2.5	367	240
Apple iPhone 6+, 6s+, 7+, 8+	1080	1920	414	736	3	401	288
Motorola Nexus 6	1440	2560	412	690	3.5	493	336
Samsung Galaxy Note	800	1280	400	640	2	285	192
Xiaomi Redmi Note 5	1080	2160	393	786	2.75	403	264
Blackberry Leap	720	1280	390	695	2	294	177
Blackberry Classic	720	720	390	390	1.8	294	177
LG Optimus G	768	1280	384	640	2	318	192
LG Nexus 4	768	1280	384	640	2	320	192

Рис. 954. Таблица с разрешениями экранов (Smartphones)

В данных таблицах представлены наиболее распространенных модели устройств. Вы также можете посмотреть фактическую ширину и высота экрана для планшетов, переключившись выше на вкладку **Tablets**.

Как видим, значение не превышает **504** пикселей для Blackebrry Passport. Но это для портретной (вертикальной) ориентации, когда пользователь устройство держит вертикально. Но есть еще и ландшафтная ориентация (столбец CSS height), которую тоже надо учитывать. Именно поэтому в коде выставлено значение **768**. Хотя сейчас у того же Apple iPhone X это значение составляет **812** пикселей.



name	phys. width	phys. height	CSS width	CSS height	pixel ratio	phys. ppi	CSS ppi
Apple iPhone X	1125	2436	375	812	3	458	288
Xiaomi Redmi Note 5	1080	2160	393	786	2.75	403	264
Microsoft Lumia 1520	1080	1920	432	768	2.5	367	240
Samsung Galaxy S8+	1440	2960	360	740	4	529	384
Samsung Galaxy S8	1440	2960	360	740	4	568	384
Samsung Galaxy Note 8	1440	2960	360	740	4	521	384
Apple iPhone 6+, 6s+, 7+, 8+	1080	1920	414	736	3	401	288
Blackberry Leap	720	1280	390	695	2	294	177
Motorola Nexus 6	1440	2560	412	690	3.5	493	336
Apple iPhone 7, iPhone 8	750	1334	375	667	2	326	192
Apple iPhone 6, 6s	750	1334	375	667	2	326	192
Samsung Galaxy Note	800	1280	400	640	2	285	192

Рис. 955. Пример разрешения экрана Apple iPhone X

Примечание: в 2001 году в HTML4 и CSS2 была введена поддержка аппаратно-зависимых таблиц стилей, позволившая создавать стили и таблицы стилей для определенных типов устройств.

Из документации Mozilla (см. приложение):

Медиазапросы (Media Queries, @media) *используются в тех случаях, когда нужно применить разные css стили, для разных устройств по типу отображения (например: для принтера, монитора или смартфона), а также конкретных характеристик устройства (например: ширины окна просмотра браузера), или внешней среды (например: внешнее освещение). Учитывая огромное количество подключаемых к интернету устройств,*

медиазапросы являются очень важным инструментов при создании веб-сайтов и приложений, которые будут правильно работать на всех доступных устройствах, которые есть у Ваших пользователей.

Я рекомендую ориентироваться на следующие медиазапросы (см. приложение):

```
##Device = Desktops
##Screen = 1281px и выше
##Device = Laptops, Desktops
##Screen = B/w 1025px to 1280px
##Device = Tablets, Ipads (portrait)
##Screen = B/w 768px to 1024px
##Device = Tablets, Ipads (landscape)
##Screen = B/w 768px to 1024px
##Device = Low Resolution Tablets, Mobiles (Landscape)
##Screen = B/w 481px to 767px
##Device = Most of the Smartphones Mobiles (Portrait)
##Screen = B/w 320px to 479px
```

Как видим, самые популярные разрешения смартфонов в портретной ориентации лежат в пределах 320 – 479 пикселей, у планшетов в пределах 768 – 1024 пикселей (в нашем примере я поставил 992).

С каждым годом количество новых моделей становится больше, размеры экранов увеличиваются. Если мы перейдем в Google Analytics и построим специальный отчет, то можем наблюдать множество различных разрешений экранов и моделей мобильных устройств.

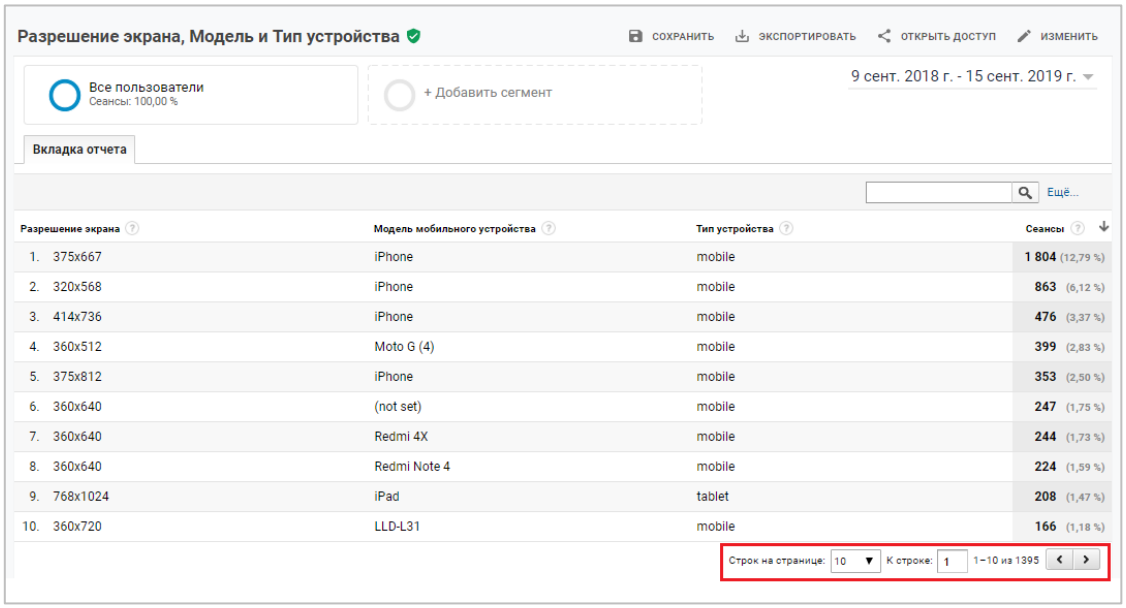


Рис. 956. Пример специального отчета

С помощью данного скрипта не всегда возможно со 100% вероятностью верно определить тип устройства пользователя. В этом и заключаются основные минусы способа №1:

- установка ограничений по ширине и высоте попиксельно – какие цифры считать правильными?
- сложно учесть ориентацию пользователя в момент просмотра сайта (держит он девайс горизонтально или вертикально?).

Как проверить? Для этого достаточно в GTM запустить режим отладки и посмотреть, какое значение передается в нашу переменную.

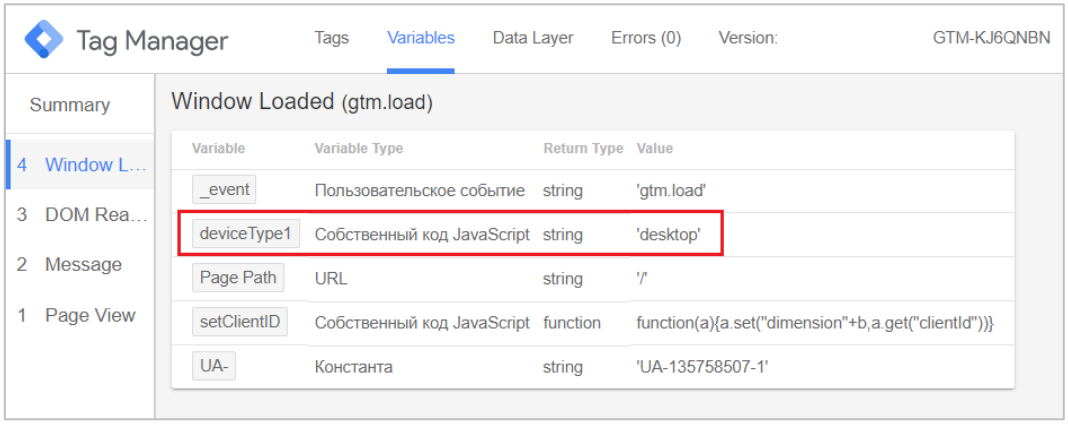


Рис. 957. Проверка способа №1 в режиме отладки

Я зашел на сайт с персонального компьютера, тип устройства он определил корректно (**desktop**). С помощью консоли разработчика и функции **Toggle Device Toolbar** я также могу протестировать адаптивный дизайн сайта путем имитации разных размеров и разрешений экрана, выбрав нужный из списка. Перезагрузив страницу, в переменной **deviceType1** я получу значение **mobile**.

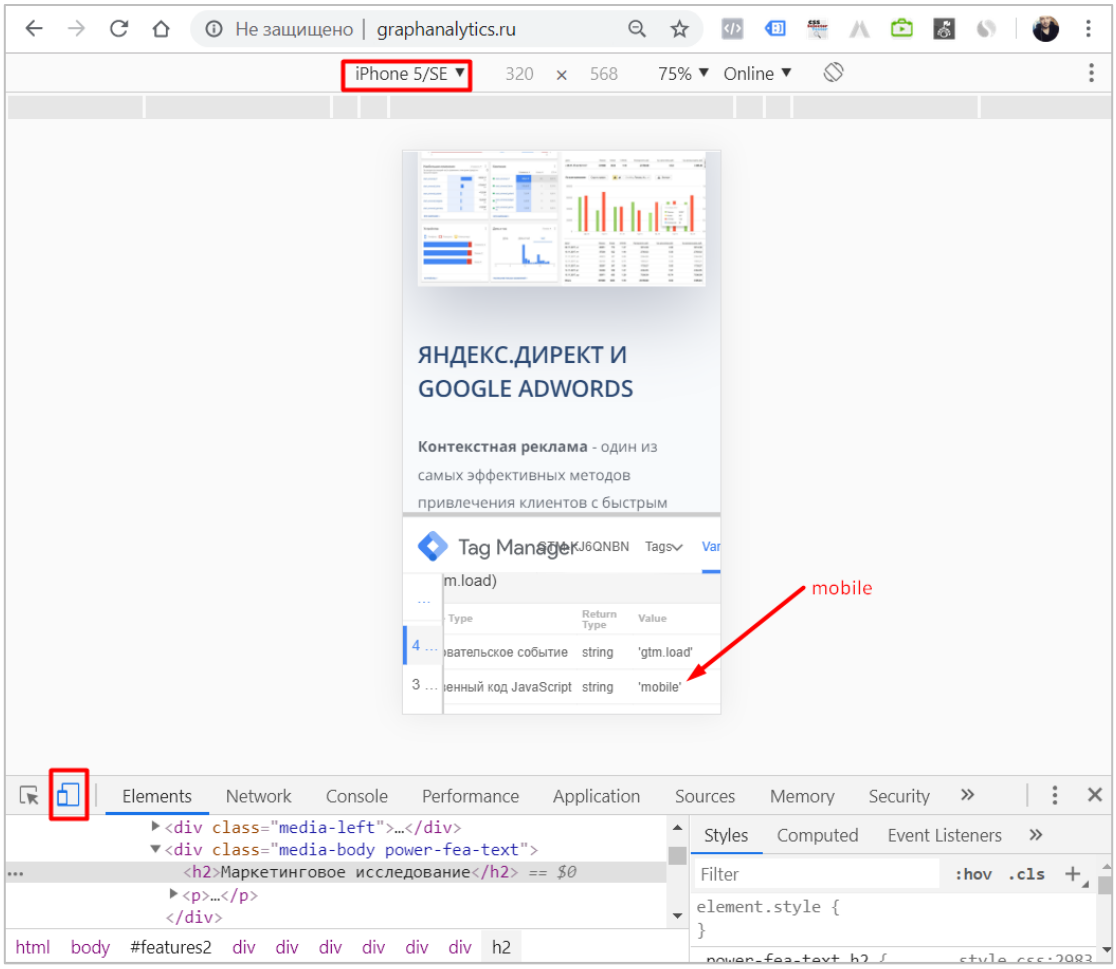


Рис. 958. Toggle Device Toolbar

Эту пользовательскую переменную можно использовать в триггерах и тегах для любых условий активации (ограничивается вашей фантазией и задачами), и передавать эту информации в счетчики аналитики. Кроме этого, вы на этапе трекинга таким образом можете исключить данные, которые не планируете отправлять в Analytics. Ярким примером реализации служит клик по номеру телефона на мобильных устройствах.

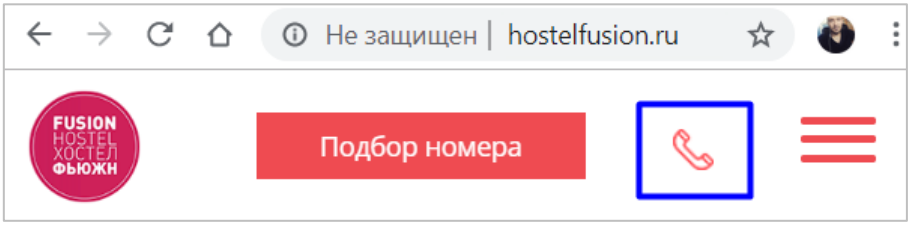


Рис. 959. Клик по телефонной трубке (номеру телефона) на мобильных устройствах

На десктопе в принципе невозможно после такого клика дозвониться (если только не установлена специальная программа для звонков) по телефону, указанному на сайте. А с мобильного устройства – это потенциальное обращение (лид, конверсия). Но люди могут кликать с компьютера, тем самым искажать статистику. А если данное событие вы используете как конверсию при оптимизации рекламных кампаний Google Ads, то отслеживание всех кликов заставит алгоритмы Google работать не лучшим образом.

Способ №2

Этот способ наиболее точный, но он возвращает не тип устройства, а булево значение **true** (мобильное устройство, истина) или **false** (нет, ложь).

Чтобы воспользоваться этим методом, в GTM также создайте пользовательскую переменную типа **Собственный код JavaScript** со следующим кодом:

```
  function () {
  var isMobile = false; // initiate as false
  // device detection
  if(/(android|bb\d+|meego).+mobile|avantgo|bada\/|blackberry|blazer|compal|elai
ne|fennec|hiptop|iemobile|ip(hone|od)|ipad|iris|kindle|Android|Silk|lge
|maemo|midp|mmp|netfront|opera m(ob|in)i|palm(
os)?|phone|p(ixi|re)\/|plucker|pocket|psp|series(4|6)0|symbian|treo|up\.(browser|
link)|vodafone|wap|windows (ce|phone)|xda|xiino/i.test(navigator.userAgent)
      || /1207|6310|6590|3gso|4thp|50[1-6]i|770s|802s|a wa|abac|ac(er|oo|s\-
)|ai(ko|rn)|al(av|ca|co)|amoi|an(ex|ny|yw)|aptu|ar(ch|go)|as(te|us)|attw|au(di|\-
m|r |s )|avan|be(ck|ll|nq)|bi(lb|rd)|bl(ac|az)|br(e|v)w|bumb|bw\-
(n|u)|c55\/|capi|ccwa|cdm\-|cell|chtm|cldc|cmd\-
|co(mp|nd)|craw|da(it|ll|ng)|dbte|dc\-s|devi|dica|dmob|do(c|p)o|ds(12|\-
d)|el(49|ai)|em(l2|ul)|er(ic|k0)|esl8|ez([4-7]0|os|wa|ze)|fetc|fly(\-|_)|g1
u|g560|gene|gf\-5|g\-mo|go(\.w|od)|gr(ad|un)|haie|hcit|hd\-(m|p|t)|hei\-
|hi(pt|ta)|hp( i|ip)|hs\-c|ht(c(\-| |_|a|g|p|s|t)|tp)|hu(aw|tc)|i\-
(20|go|ma)|i230|iac( |\-
|\/)|ibro|idea|ig01|ikom|im1k|inno|ipaq|iris|ja(t|v)a|jbro|jemu|jigs|kddi|keji|kg
t( |\/)|klon|kpt |kwc\-|kyo(c|k)|le(no|xi)|lg( g|\/(k|l|u)|50|54|\-[a-
w])|libw|lynx|m1\-w|m3ga|m50\/|ma(te|ui|xo)|mc(01|21|ca)|m\-
cr|me(rc|ri)|mi(o8|oa|ts)|mmef|mo(01|02|bi|de|do|t(\-| |o|v)|zz)|mt(50|p1|v
)|mwbp|mywa|n10[0-2]|n20[2-3]|n30(0|2)|n50(0|2|5)|n7(0(0|1)|10)|ne((c|m)\-
|on|tf|wf|wg|wt)|nok(6|i)|nzph|o2im|op(ti|wv)|oran|owg1|p800|pan(a|d|t)|pdxg|pg(1
3|\-([1-8]|c))|phil|pire|pl(ay|uc)|pn\-2|po(ck|rt|se)|prox|psio|pt\-g|qa\-
a|qc(07|12|21|32|60|\-[2-7]|i\-
)|qtek|r380|r600|raks|rim9|ro(ve|zo)|s55\/|sa(ge|ma|mm|ms|ny|va)|sc(01|h\-|oo|p\-
)|sdk\/|se(c(\-|0|1)|47|mc|nd|ri)|sgh\-|shar|sie(\-|m)|sk\-
0|sl(45|id)|sm(al|ar|b3|it|t5)|so(ft|ny)|sp(01|h\-|v\-|v
)|sy(01|mb)|t2(18|50)|t6(00|10|18)|ta(gt|lk)|tcl\-|tdg\-|tel(i|m)|tim\-|t\-
mo|to(pl|sh)|ts(70|m\-|m3|m5)|tx\-
9|up(\.b|g1|si)|utst|v400|v750|veri|vi(rg|te)|vk(40|5[0-3]|\-
v)|vm40|voda|vulc|vx(52|53|60|61|70|80|81|83|85|98)|w3c(\-| )|webc|whit|wi(g
|nc|nw)|wmlb|wonu|x700|yas\-|your|zeto|zte\-
/i.test(navigator.userAgent.substr(0,4))) {
      isMobile = true;
  }
    return isMobile;
  }
```

В Google Tag Manager выглядит так:

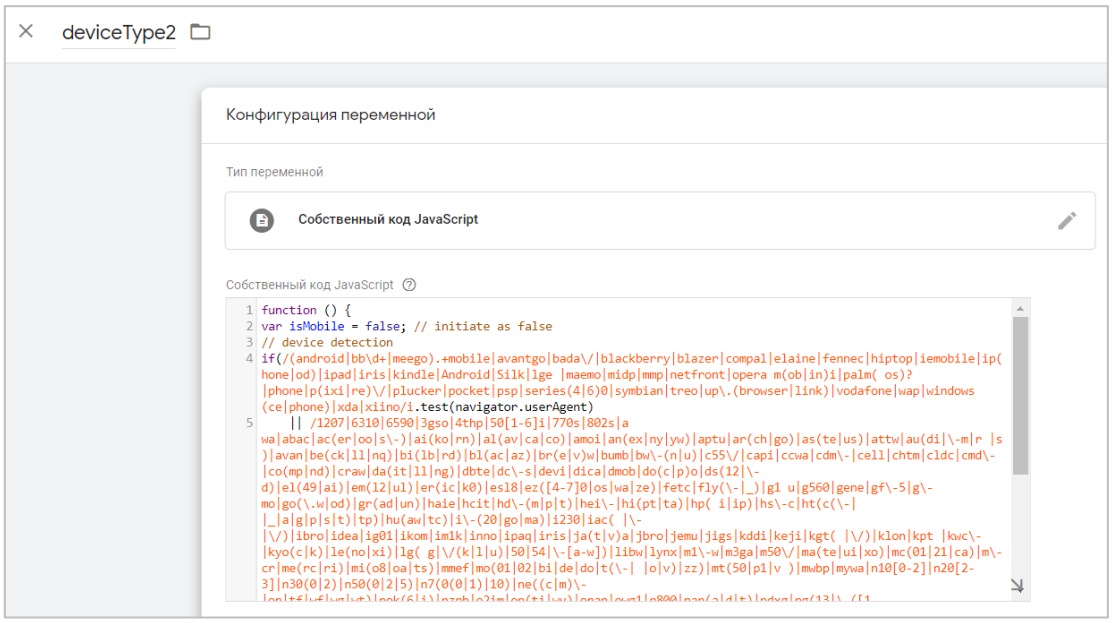


Рис. 960. Переменная Собственный код JavaScript - deviceType2

Сохраните переменную. Перейдите в режим предварительного просмотра. Так как я захожу на сайт с компьютера, то мне возвращается значение **false**.

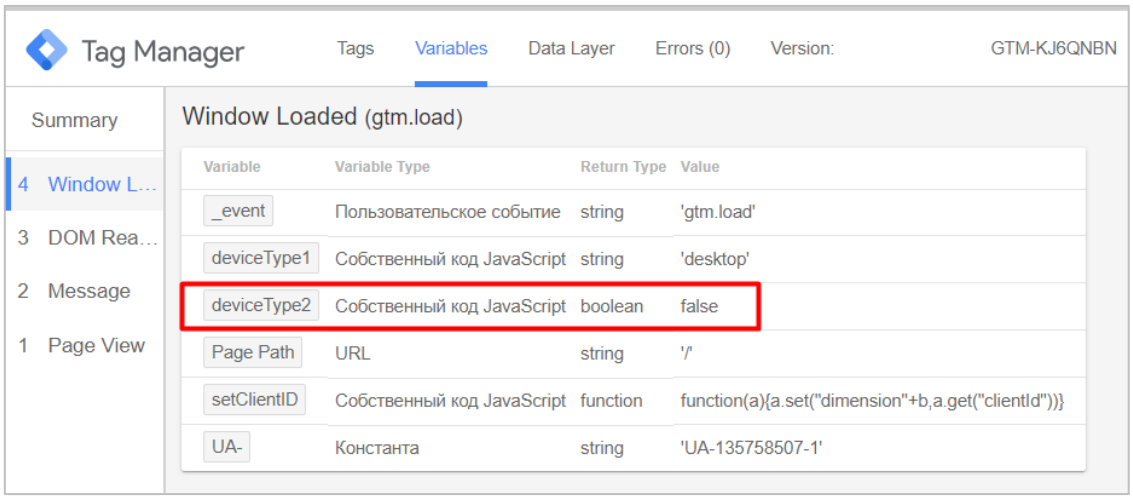


Рис. 961. Проверка способа №2 в режиме отладки

Поскольку Google Analytics и данный скрипт определяет устройство через строку **User-Agent**, можно воспользоваться расширением **User-Agent Switcher** (для Google Chrome), чтобы изменить это значение.

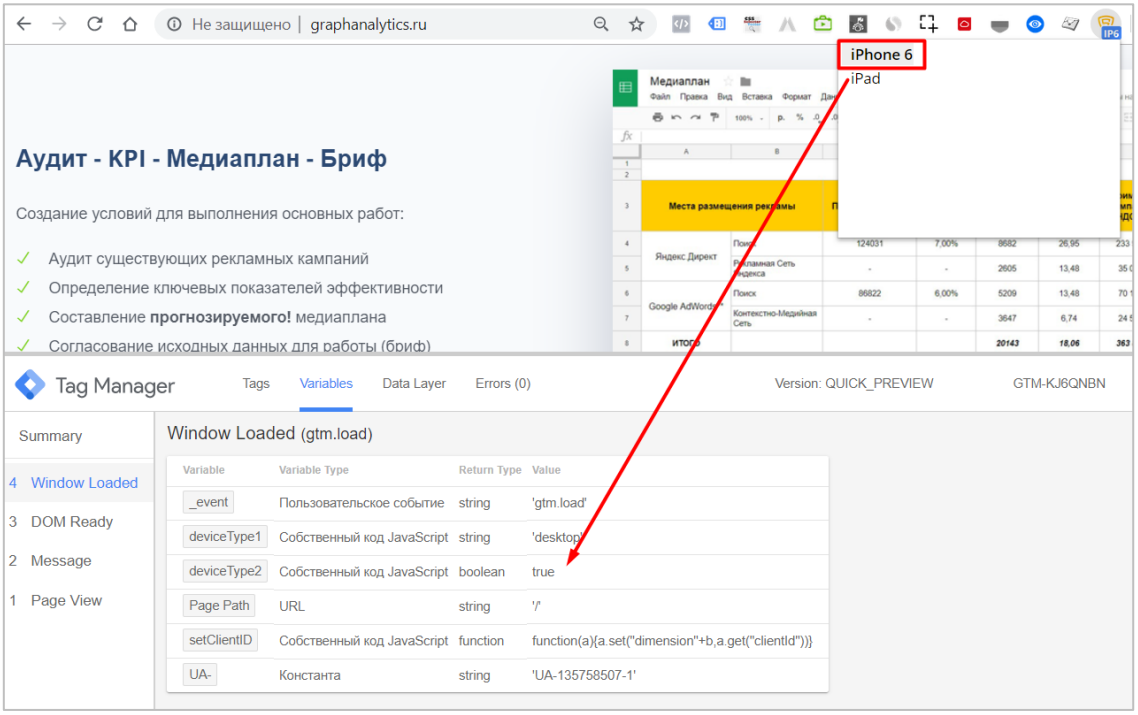


Рис. 962. Расширение User-Agent Switcher

В примере я выбрал устройство iPhone 6. Как видим на рисунке выше, переменная **deviceType2 (Способ №2)** определила по User-Agent, вернув значение **true**, а **deviceType1 (Способ №1)** определил как **desktop**, поскольку размер экрана более 992 пикселей по ширине. Моя рекомендация: использовать способ №2.

Напоследок в качестве примера привожу настройки триггера, который активируется при каждой загрузке страницы, когда пользователь просматривает ваш сайт с помощью мобильного устройства.

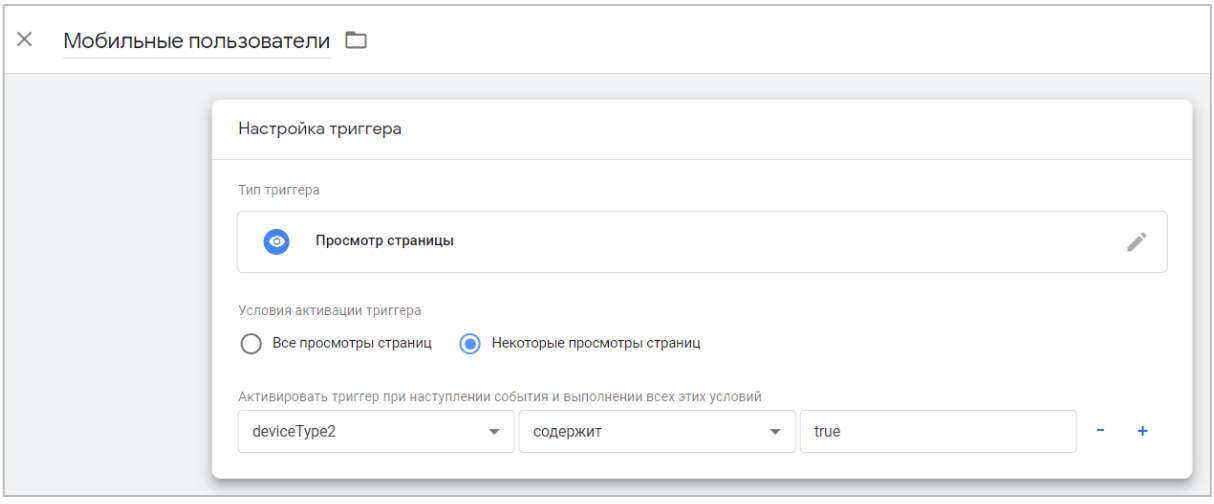


Рис. 963. Триггер, который срабатывает только для мобильных пользователей

Определение геолокации пользователя, включая IP-адрес

Хотите получить точную информацию о местоположении пользователя и передать эти данные в инструменты аналитики? IP-адрес, страну, город, часовой пояс, географические координаты (широту и долготу), интернет-провайдера, валюту, текущее время, имя хоста и многое другое? Встречайте, сервис **ipgeolocation.io**.

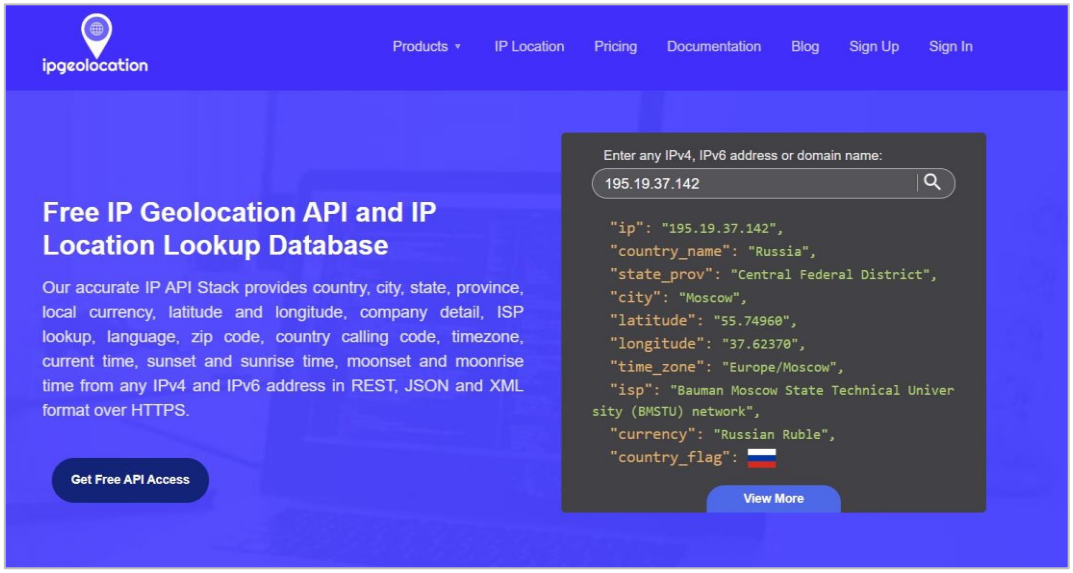


Рис. 964. Сервис ipgeolocation.io

Он позволяет получать все вышеперечисленные данные и предоставляет 3 продукта API:

1. **IP геолокации**;
2. **астрономический API** (синхронизация времени восхода, захода солнца, восхода луны, захода луны, азимута солнца, азимута луны, высоты солнца, высоты луны, расстояния от Земли до Луны и расстояния от Земли до широты и долготы, а также адреса IPv4 или IPv6);
3. **API часовых поясов** (информация о времени и дате, текущее время, дата в различных форматах, неделя, месяц, год, время в формате Unix, смещение UTC / GMT и время перехода на летнее время из названия часового пояса, любой адрес IPv4 или IPv6 или координаты геолокации в формате REST, JSON и XML через защищенное соединение HTTPS).

С помощью сервиса **ipgeolocation.io** и тега **IP Geolocation API**, который разработал **Симо Ахава** и который добавляется в рабочую область из галереи решений, можно легко получить IP-адрес и другие данные о пользователе в dataLayer.

Что для этого нужно сделать? Зарегистрируйтесь по ссылке **https://ipgeolocation.io/signup.html** для получения ключа API:

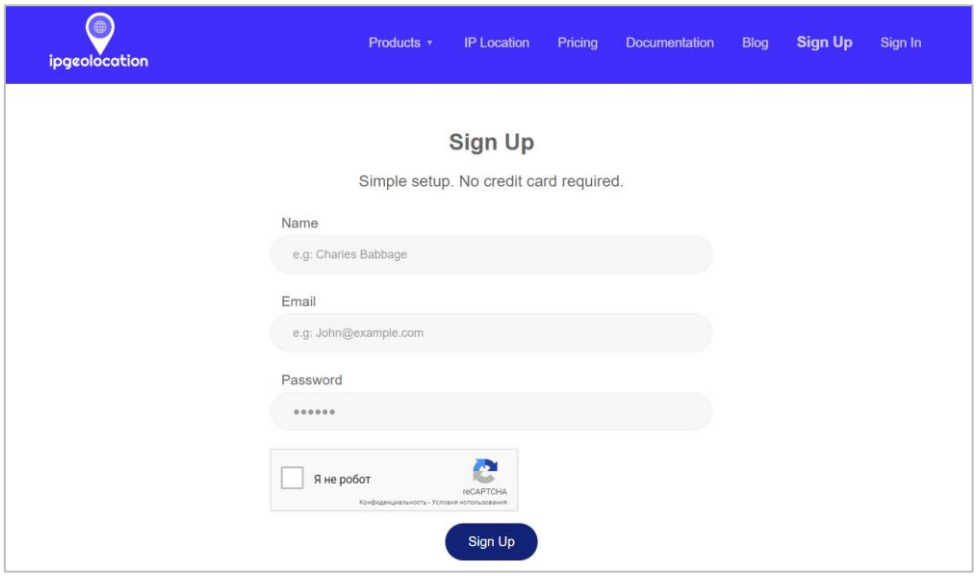


Рис. 965. Регистрация на сайте

Бесплатный тариф предназначен для некоммерческого использования и имеет ограничение в 1000 запросов/день. После регистрации в личном кабинете необходимо скопировать **API ключ (Your API Key)**, который вы вставите в тег Симо.

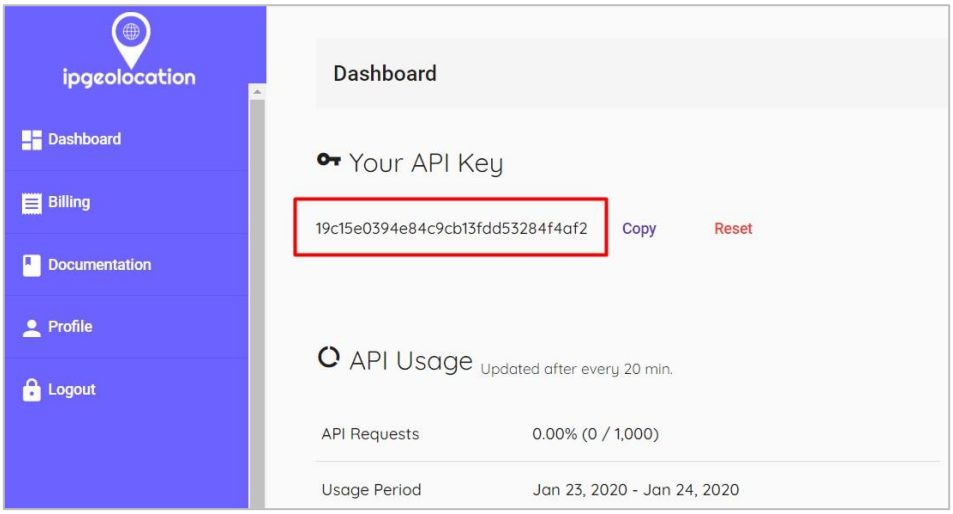


Рис. 966. Копирование ключа API

Далее перейдите в Google Tag Manager и добавьте тег **IP Geolocation API** в свою рабочую область.

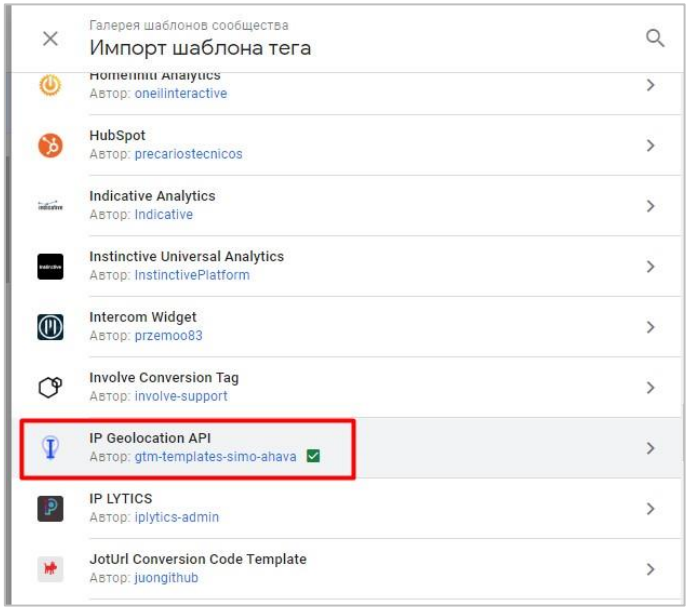


Рис. 967. Галерея шаблонов сообщества Google Tag Manager

Добавьте свой ключ API в соответствующее поле:

Рис. 968. Ваш ключ API Key

Если вы выбрали платный тариф, вы можете добавить источник (например **www.osipenkov.ru**) в список источников запроса (**Request Origins**). В этом случае добавлять ключ API в поле **API Key** не нужно.

В теге есть следующие настройки: **Data Layer Settings, IP Address Settings, Other Settings.**

Data Layer Settings

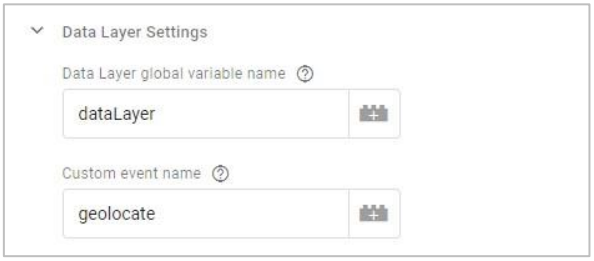


Рис. 969. Data Layer Settings

Если вы используете имя уровня данных отличное от dataLayer, то укажите его в поле глобальной переменной. В поле **Custom event name** можно указать значение ключа события (event) в объекте dataLayer, который содержит данные о геолокации и IP-адресе. По умолчанию оно содержит **geolocate**, но вы можете задать любое название события.

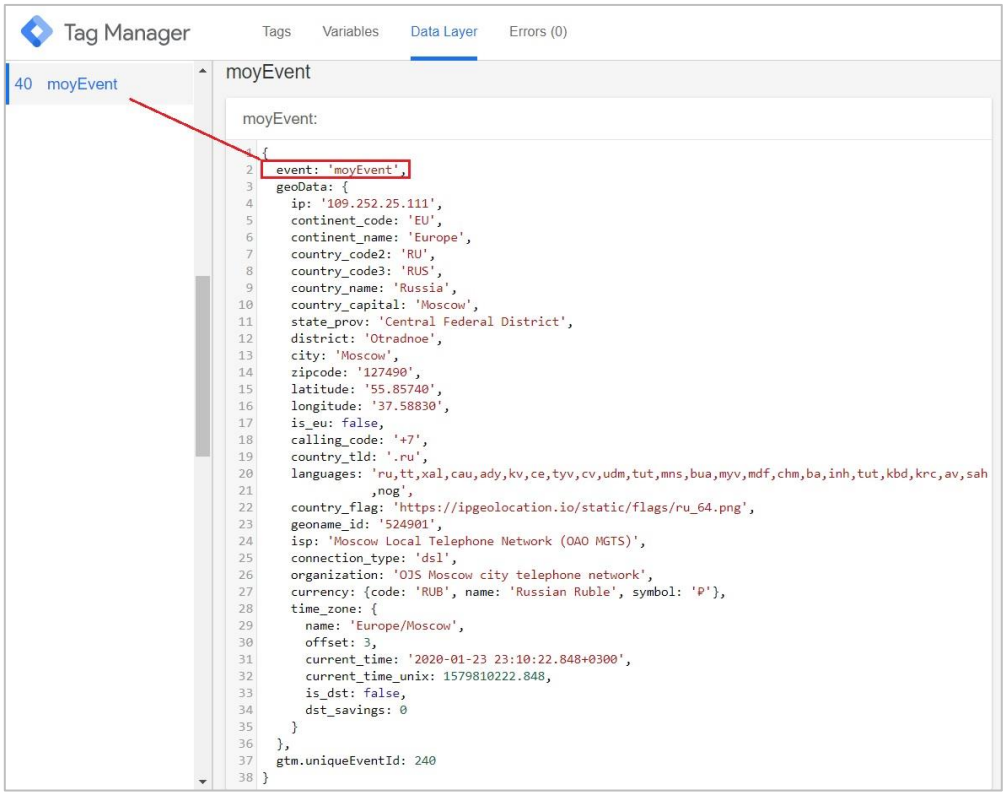


Рис. 970. Пример имени события (moyEvent)

IP Address Settings

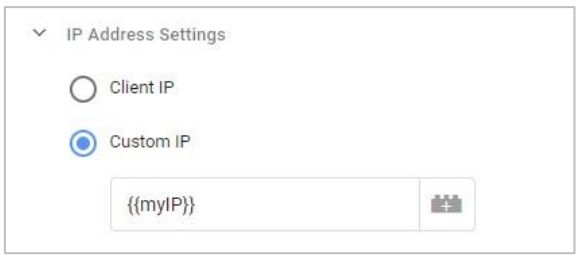


Рис. 971. IP Address Settings

Доступно две настройки: **Client IP** (тот, который определит ipgeolocation.io) и **Custom IP** (можете использовать свой вариант, полезно в том случае, когда вы определяете IP-адрес каким-то другим способом, например, с помощью программного кода).

Other Settings

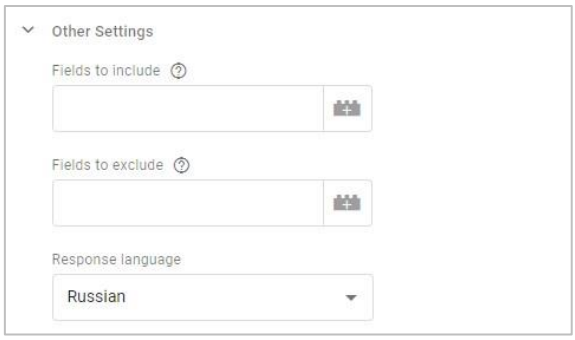


Рис. 972. Другие настройки тега

В других настройках содержатся поля:

- **Fields to include** (перечень включенных полей, которые попадут в конечный dataLayer);
- **Fields to exclude** (перечень полей, которые требуется исключить из dataLayer);
- **Response language** (язык ответа, некоторые значения местоположений будут переведены с английского языка на тот, который выберите из списка).

Рис. 973. Перевод некоторых ответов на выбранный язык (Response language)

Важно: при изменении языка в настройке **Response language** на **Russian** может не срабатывать триггер Пользовательское событие и данные в веб-аналитику не будут передаваться. Чтобы исправить это, оставьте по умолчанию язык **English**.

Например, если в поле **Fields to include** добавить **zipcode**, то в dataLayer останется только почтовый индекс. Все остальные данные будут исключены. Напротив, если в поле **Fields to exclude** ввести **zipcode**, то в dataLayer отобразится все, кроме почтового индекса. Если оба поля оставить пустыми, то в dataLayer попадут все данные, которые отдает сервис **ipgeolocation.io**.

После заполнения всех настроек вы можете добавить любой триггер к тегу. Например, **Все страницы (All Pages)**, либо же какое-то определенное правило. Вы также сможете запускать тег по триггеру пользовательского события, который установили в настройках тега. В моем примере это **moyEvent**, а по умолчанию **geolocate**.

Примечание: тег опирается на библиотеку jQuery. Если ваш сайт еще не использует его, шаблон автоматически загрузит уменьшенную библиотеку jQuery вместе с API SDK.

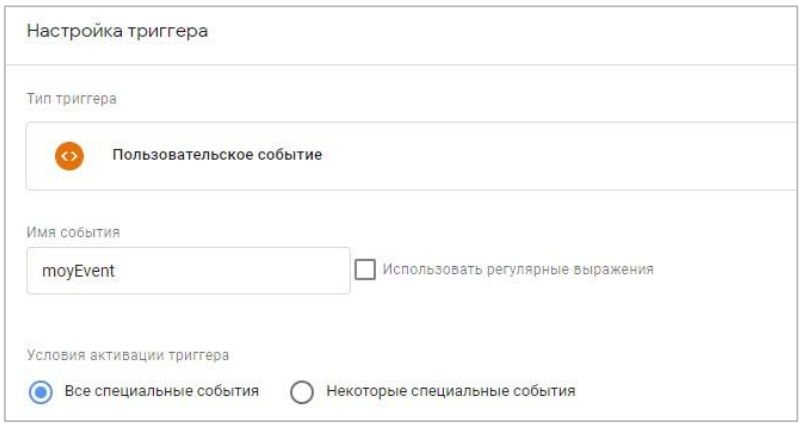


Рис. 974. Триггер Пользовательское событие

Чтобы извлечь данные из dataLayer и отправить их в инструменты аналитики (Google Analytics и Яндекс.Метрику), используйте переменную типа **Переменная уровня данных**. Для **района (district)** переменная будет выглядеть так:

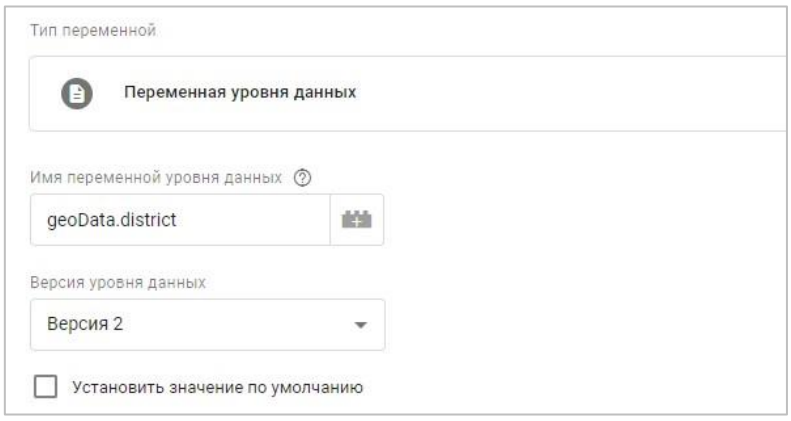


Рис. 975. Переменная уровня данных

Аналогично с другими параметрами. Проверить корректность извлечения информации можно в режиме предварительного просмотра GTM:

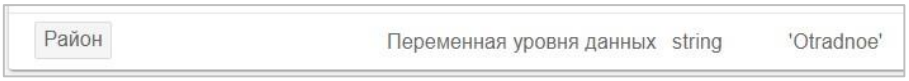


Рис. 976. Данные по району в переменной уровня данных

Далее эту информацию можно передать в Google Analytics с помощью пользовательского параметра.

ipgeolocation.io - полезный инструмент в работе интернет-маркетолога для определения местоположения пользователя и использования этой информации для своих нужд. А с помощью API ключа и тега Симо для Google Tag Manager настроить отслеживание 30+ параметров с передачей данных в инструменты аналитики можно всего за несколько минут.

Отслеживание исчезающих элементов

Три способа, которые позволяют отслеживать исчезающие элементы с экрана и настраивать их с помощью Google Tag Manager.

У некоторых проектов может возникнуть проблема извлечения элементов DOM у формы, которая исчезает практически сразу же после заполнения заявки. Окно с текстом **«Спасибо! Данные успешно отправлены»** появляется на 1-2 секунды, а затем сразу же пропадает.

В такой ситуации мы никак не сможем успеть исследовать элемент, выделить его и скопировать нужный CSS-селектор, чтобы затем применить его в каком-нибудь триггере. Бывают случаи, когда элемент исчезает при отведении от него фокуса. Например, такое происходит у раскрывающихся списков, или подсказок на сайте. Что же тогда делать?

Оказывается, решение есть! И даже не одно. Я «загуглил» и нашел 3 способа:

1. использование функции **setTimeout(function(){debugger;}, interval)** в консоли разработчика;
2. клавиша F8 в консоли разработчика, которая отвечает за остановку выполнения скриптов;
3. отключение выполнения JavaScript с помощью настройки в консоли разработчика.

Разберем каждый из этих способов.

1. setTimeout(function(){debugger;}, interval)

setTimeout() вызывает функцию или выполняет некоторый код после указанной задержки (в миллисекундах). Объединив его с оператором **debugger**, можно активировать режим отладки в консоли, когда автоматически скрываемый элемент виден, и не беспокоится о том, что он может исчезнуть. А **interval** – это и есть задержка в миллисекундах, по истечении которой будет выполнена функция. Если аргумент не задан, то значение равно 0.

Чтобы использовать этот способ, откройте консоль разработчика (клавиша F12 в Google Chrome), выберите вкладку **Console** и в строке введите:

```
setTimeout(function(){debugger;}, 3000)
```

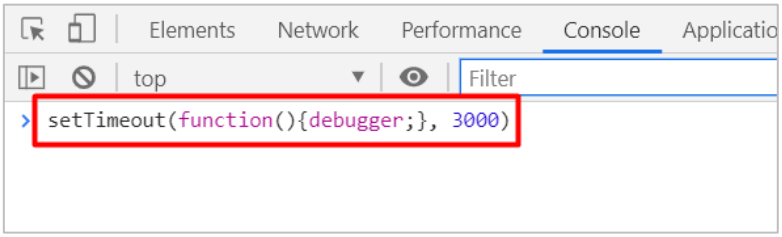


Рис. 977. Функция setTimeout

Нажмите **Enter**. В окне браузера наведите курсор мыши на нужный элемент. Через заданный промежуток времени выполнение JavaScript прекратится. То есть у вас есть ровно 3 секунды (в этом примере), чтобы выделить нужный элемент. После этого вы можете исследовать его как вам захочется. Можно перейти на вкладку **Elements**, найти CSS-селектор и добавить его в GTM.

Для отмены действия можно просто перезагрузить страницу.

2. Клавиша F8

Все то же самое, только данная команда упрощает нам остановку выполнения скриптов. Если в первом случае нам необходимо ждать указанное количество времени, то с помощью F8 мы запускаем **debugger** мгновенно. Этот способ наиболее предпочтительный, поскольку угадать, когда попап от успешно заполненный формы появится на экране (как у моего ученика на курсе) заранее не представляется возможным.

Для этого также нужно перейти в консоль разработчика, потом на вкладку **Sources** и когда отслеживаемый элемент появится на экране, нажать на клавишу **F8**.

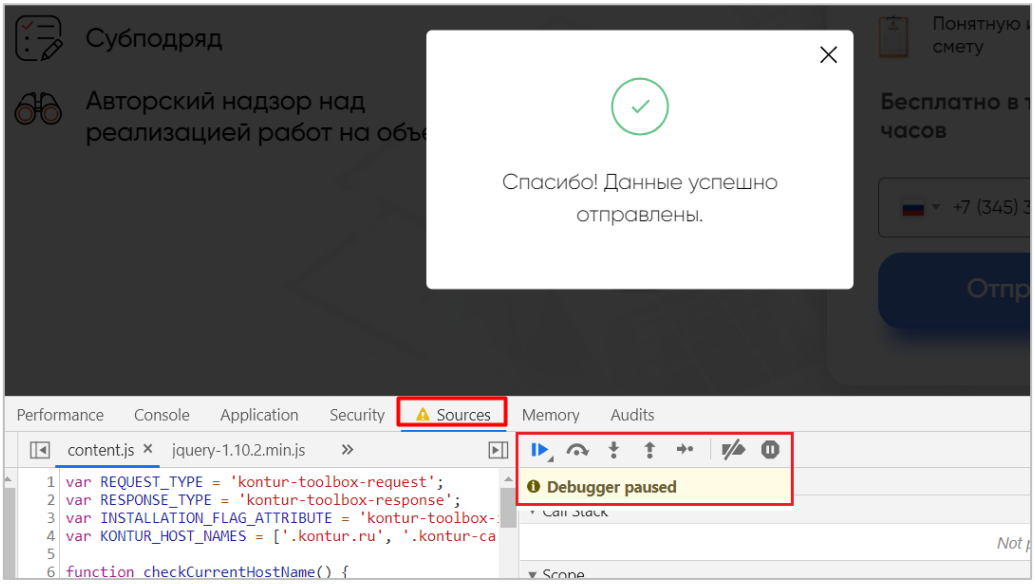


Рис. 978. Клавиша F8

Debugger перейдет в состоянии паузы. Вы можете делать в этот момент все, что захотите. Для отмены действия можно просто перезагрузить страницу.

3. С помощью отключения JavaScript

Способ похож на предыдущий, только вместо клавиши F8 -> простановка галочки. Для того, чтобы им воспользоваться, перейдите в консоль разработчика, затем справа нажмите на иконку с тремя точками – **Settings** (или клавишу F1). Найдите настройку **Disable JavaScript**.

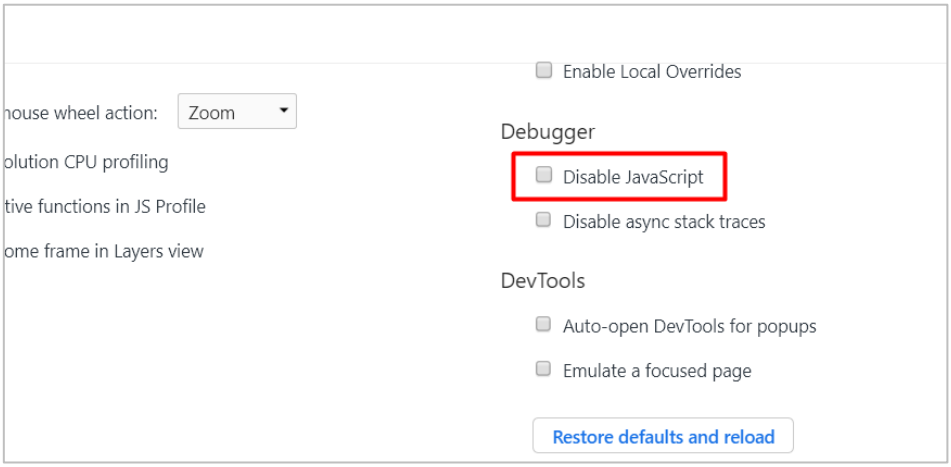


Рис. 979. Настройка Disable JavaScript

Пока там не должно стоять галочки. Теперь вам необходимо сделать так, чтобы исследуемый элемент появился на экране. Как только это наступит, вы должны успеть поставить галочку рядом с **Disable JavaScript**. Этим вы остановите дальнейшее выполнение скрипта формы, и элемент не пропадает с экрана. Как и в 2 других способах, вы можете делать в этот момент все, что захотите. После завершения проверки элемента снова включите JavaScript.

Отслеживание диалоговых окон alert ()

Сайты, на которых после заполнения заявки открывается диалоговое окно об успешной отправке формы, еще существуют! В этой статье разберем как можно отслеживать такие окна с помощью Google Tag Manager.

Команда **alert ()** выводит на экран окно с сообщением, приостанавливает выполнение сценария и дальнейшее взаимодействие пользователя со страницей до тех пор, пока это окно не закроется. В рассматриваемом примере это делается с помощью кнопки **OK**:

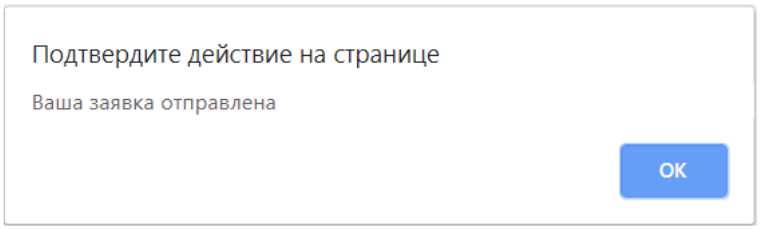


Рис. 980. Диалоговое окно alert ()

У таких окон нет ни классов, ни идентификаторов, ни CSS-селекторов. К этому объекту таким образом нельзя достучаться. Но можно использовать готовое решение – заместитель (Proxy pattern), который позволит отслеживать такие модальные окна.

Чтобы это сделать, необходимо в Google Tag Manager вставить в тег типа **Пользовательский HTML** следующий код (см. приложение):

```
<script>
if(!(document.all && !document.addEventListener))
{
 if(!window.proxied_alert)
 {
   window.proxied_alert = window.alert;
   window.alert = function() {
     var message = (!arguments[0]) ? 'null': arguments[0];
     dataLayer.push({'event': 'alert_showed','alert_message': message});
     return proxied_alert.apply(this, arguments);
   }
 }
}
</script>
```

В GTM это выглядит так:

Рис. 981. Пользовательский HTML

Триггер активации – **All Pages (Все страницы)**. Можно задать и другое условие. Например, активировать тег только на тех страницах, где появляются такие окна.

К сожалению, это не будет работать для Internet Explorer ниже 9 версии, поскольку в них alert не функция (function), а объект (object). Поэтому в самом начале кода добавлена проверка для браузеров до IE9.

С помощью такой конструкции мы передаем информацию в dataLayer. Теперь нам необходимо создать триггер типа **Пользовательское событие** - **alert_showed** и **alert_message**. При этом можно использовать регулярные выражения:

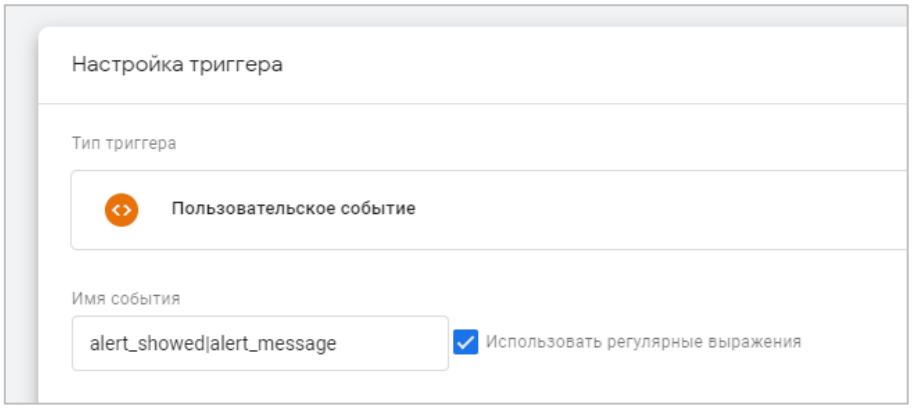


Рис. 982. Триггер Пользовательское событие

Сохраняем. Также можно создать соответствующие теги для отправки данных в Google Analytics и Яндекс.Метрику.

Теперь при заполнении формы будет открываться всплывающее окно, срабатывать событие и активироваться тег. В режиме предварительного просмотра GTM это выглядит так:

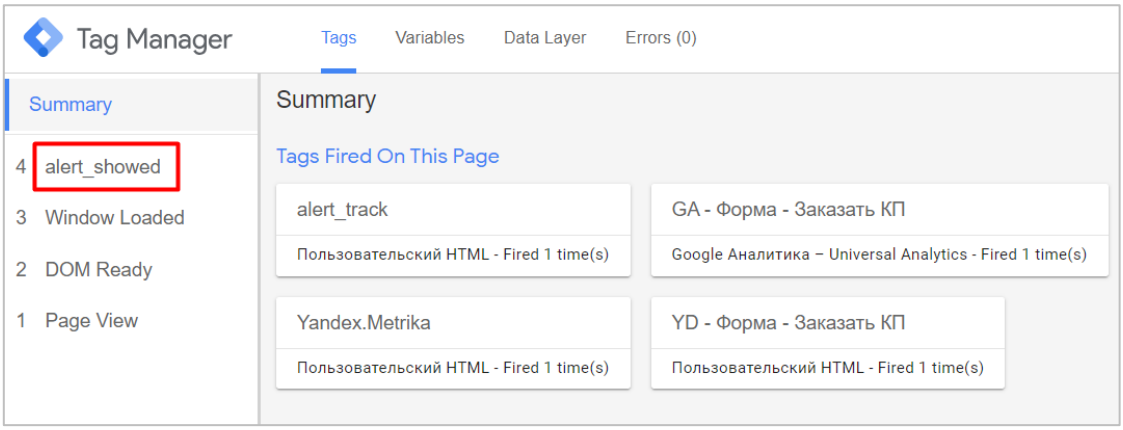


Рис. 983. Проверка в режиме отладки

В **alert_message** передалось и сообщение, которое пользователь видит в диалоговом окне после отправки формы:

Рис. 984. Уровень данных события alert_message

Эта информация может пригодиться, поскольку бывают такие сайты, на которых нет проверки валидации заполненной формы, и тогда событие **alert_showed** будет фиксироваться каждый раз, когда пользователь просто будет нажимать на кнопку Отправить. Однако клик по кнопке не равноценен отправленной заявке.

В этом случае **alert_message** в Google Tag Manager можно использовать пользовательскую переменную типа **Переменная уровня данных**.

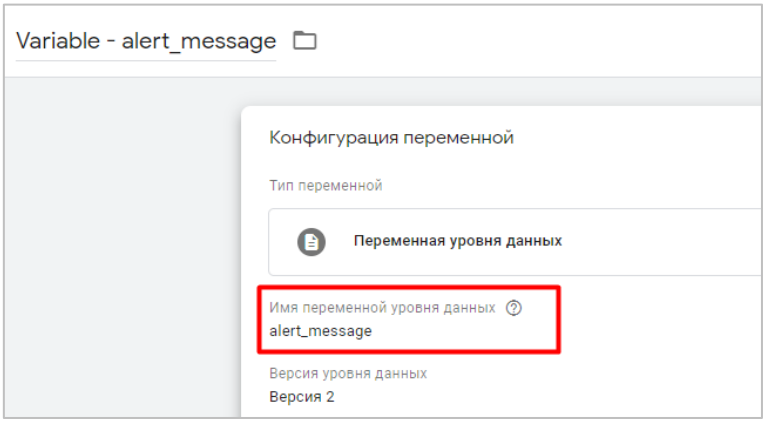


Рис. 985. Переменная уровня данных

И тогда наш триггер нужно немного изменить, поскольку мы будем фиксировать не все события, а только те, в которых выводится сообщение об успешной отправке заявки.

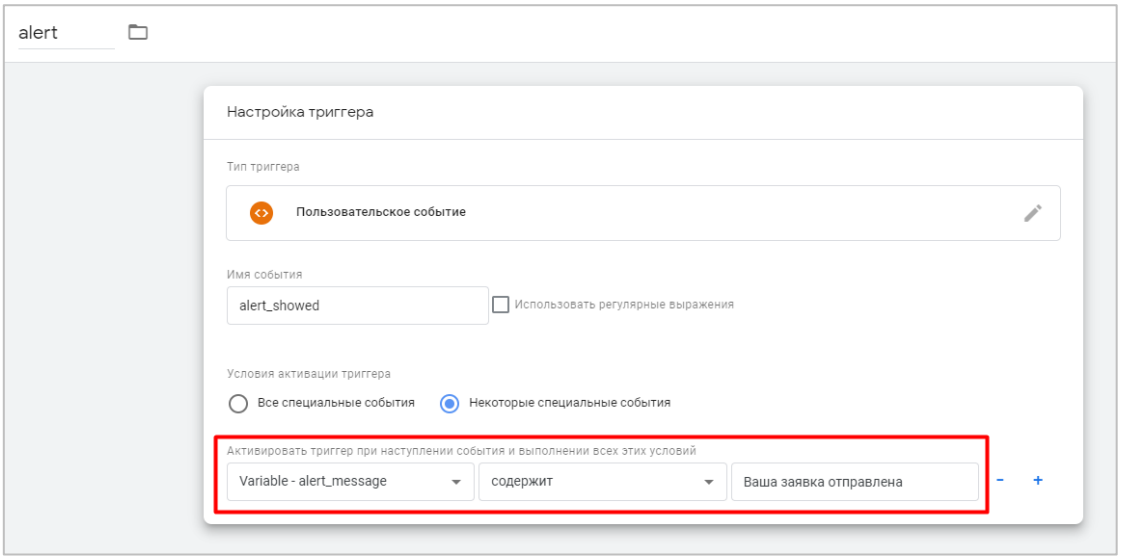


Рис. 986. Отслеживание некоторых специальных событий

Сохраняем изменения и публикуем обновленный контейнер GTM.

Настройка междоменного отслеживания

Как вы уже знаете, когда пользователь переходит на сайт, у него создается файл cookie и присваивается уникальный идентификатор (Client ID), который хранится в браузере пользователя:

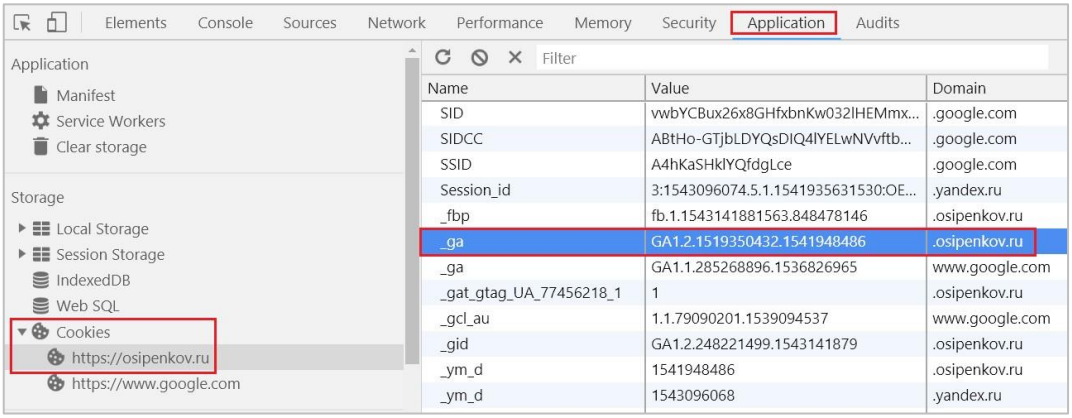


Рис. 987. Файл cookie -_ga (Google Analytics)

Но бывают ситуации, когда мы рекламируем один сайт (к примеру, домен *siteA.ru*), на который пользователи переходят по рекламе, а затем они отправляются на второй сайт (к примеру, домен *siteB.ru*) и совершают конверсию.

В результате при стандартных настройках Google Analytics зафиксируется два сеанса:

1. один на сайте siteA.ru;
2. второй на сайте siteB.ru.

Также конверсия будет закреплена по источнику **siteA.ru / referral**, и все основные показатели будут некорректными. По такой статистике нельзя сделать никаких выводов и предпринять действий. Чтобы данные отображались правильно, нам нужно настроить междоменное отслеживание.

Междоменное отслеживание – функция, позволяющая регистрировать посещение нескольких сайтов в Google Analytics как один сеанс. То есть при переходе с *siteA.ru* на *siteB.ru* будет регистрироваться один пользователь и одна сессия.

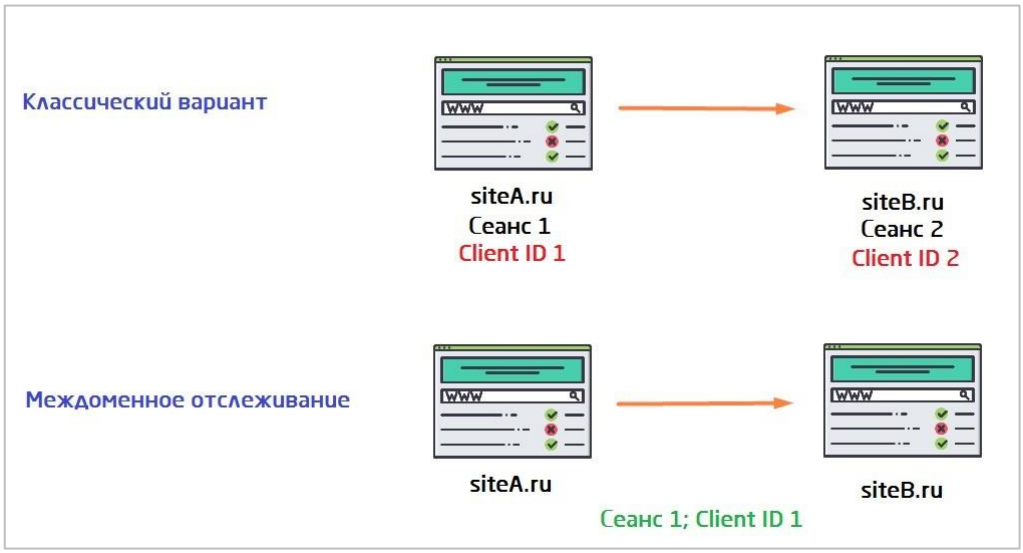


Рис. 988. Принцип работы междоменного отслеживания

Перейдем к настройке в Google Tag Manager. Есть несколько вариантов:

- если вы используете пользовательскую переменную типа **Настройки Google Analytics**, то добавьте нижеперечисленные настройки в нее;
- если вы используете переменную типа **Константа**, тогда настройки, о которых пойдет речь ниже, необходимо внести в сам тег Google Analytics.

Если у вас уже настроен тег об отправке данных в Google Analytics, вам нужно изменить его конфигурацию.

Дополнительные настройки - **Поля, которые необходимо задать**:

- Название поля - **allowLinker**,
- Значение – **true**;

Междоменное отслеживание:

- Автоматическое связывание доменов - **siteA.ru, siteB.ru** (введите домены через запятую, которые хотите отслеживать).

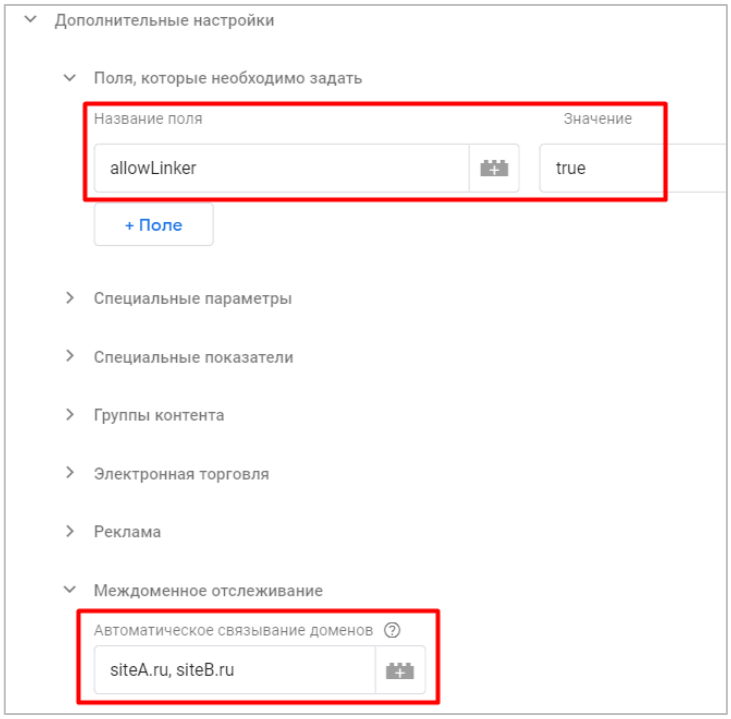


Рис. 989. Настройки междоменного отслеживания

Необязательные настройки:

- Использовать решетку в качестве разделителя – **True** (если добавляете _ga через параметр);
- Изменение внешнего вида форм – **True** (если есть форма, при отправке которой пользователя перенаправляет на второй сайт).

Сохраните изменения. На следующем шаге перейдите в Google Analytics на уровне ресурса в **Отслеживание – Список исключаемых источников перехода** и добавьте список исключаемых источников перехода. По умолчанию в настройках уже создан один источник (ваш домен). Нужно добавить для второго домена.

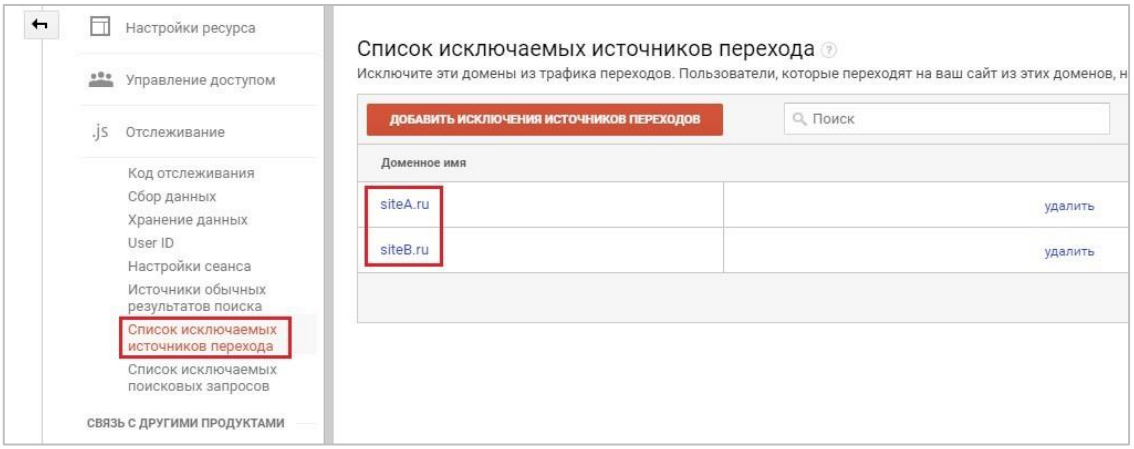


Рис. 990. Список исключаемых источников перехода

Если этого не сделать, в отчетах Google Analytics 1 пользователь при переходе с сайта на сайт будет фиксироваться как 2 сеанса, а нам нужно 1 пользователь = 1 сеанс.

После этого осуществите проверку, перейдя с одного сайта на другой. В адресной строке браузера вы должны увидеть передачу Client ID, которая начинается с _ga:

Рис. 991. Параметр связывания

Междоменное отслеживание выполняется путем передачи уникального идентификатора клиента между исходным и целевым доменом. Эта процедура выполняется в два этапа:

1. в исходном домене проверяется наличие идентификатора клиента во всех URL, указывающих на целевой домен.
2. в целевом домене проверяется наличие идентификатора клиента в URL, по которому перешел пользователь.

Настройка связывания реализует этот механизм путем добавления параметра связывания в URL-адреса, указывающие на целевой домен. Этот параметр содержит идентификатор клиента, а также закодированные текущую временную метку и метаданные браузера, которые позволяют избежать проблем с передачей URL между пользователями. Именно значение **true** в параметре **allowLinker** осуществляет проверку параметров связывания.

Для анализа данных в интерфейсе Google Analytics по страницам с названиями доменов нужно создать отдельное представление с фильтром. Добавить его можно через **Фильтры – Добавить фильтр**.

- Название фильтра – **Произвольное**;
- Тип фильтра – **Пользовательский** и **Расширенный**;
- Поле A -> Извлечь A - **Имя хоста (.*)** – извлечение имени хоста с помощью регулярного выражения;
- Поле B -> Извлечь B - **URI запроса (.*)** - извлечение URI запроса с помощью регулярного выражения;
- Вывод -> Конструктор - **URI запроса ($A1$B1)** – связывание полей A и B, где $ - метасимвол регулярного выражения.

Поставьте галочки напротив строк:

- Поле A обязательно для заполнения;
- Перезаписать поле вывода.

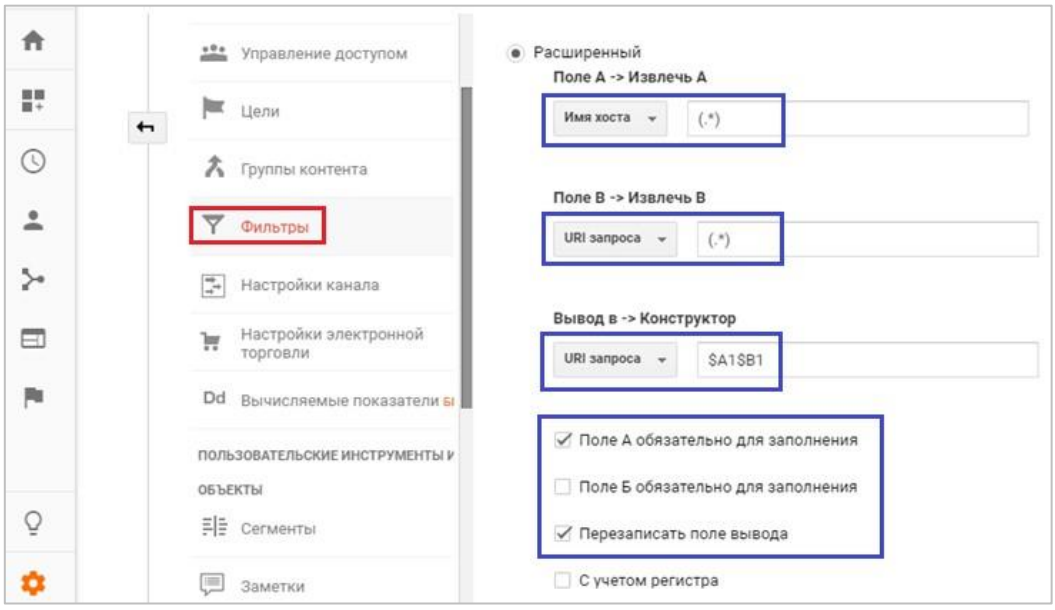


Рис. 992. Настройки фильтра Google Analytics

Сохраните фильтр. В результате вы сможете видеть статистику по страницам с названием домена.

	Страница	Источник или канал	Просмотры страниц	Уникальные просмотры страниц	Средняя длительность просмотра страницы	Входы	Показатель отказов
			32 % от общего количества: 100,00 % (32)	30 % от общего количества: 100,00 % (30)	00:17:01 Средний показатель для представления: 00:17:01 (0,00 %)	30 % от общего количества: 100,00 % (30)	76,67 % Средний показатель для представления: 76,67 % (0,00 %)
	A						
	1. siteA.ru/page1	yakov / test	18 (56,25 %)	18 (60,00 %)	00:00:00	18 (60,00 %)	100,00 %
	2. siteA.ru	yakov / test	7 (21,88 %)	6 (20,00 %)	00:00:15	6 (20,00 %)	50,00 %
	3. siteB.ru B	yakov / test	5 (15,62 %)	4 (13,33 %)	00:31:46	4 (13,33 %)	50,00 %

Рис. 993. Отчет Google Analytics по страницам

Подмена контента на сайте

Подмену контента на сайте (заголовка, подзаголовка, телефона, акции, метро и т.д.) в зависимости от условия в параметре URL (источника, рекламного объявления, запроса, гео и т.д.) можно осуществлять различными способами. Например, на базе *.php* (если знакомы с этим языком или есть разработчик) или с помощью сторонних сервисов, таких как **YAGLA, Adfor** и другие (если есть лишние $). В этом материале мы разберем *бесплатный* способ подмены заголовка на сайте с помощью Google Tag Manager.

Пример. На сайте на первом экране есть заголовок:

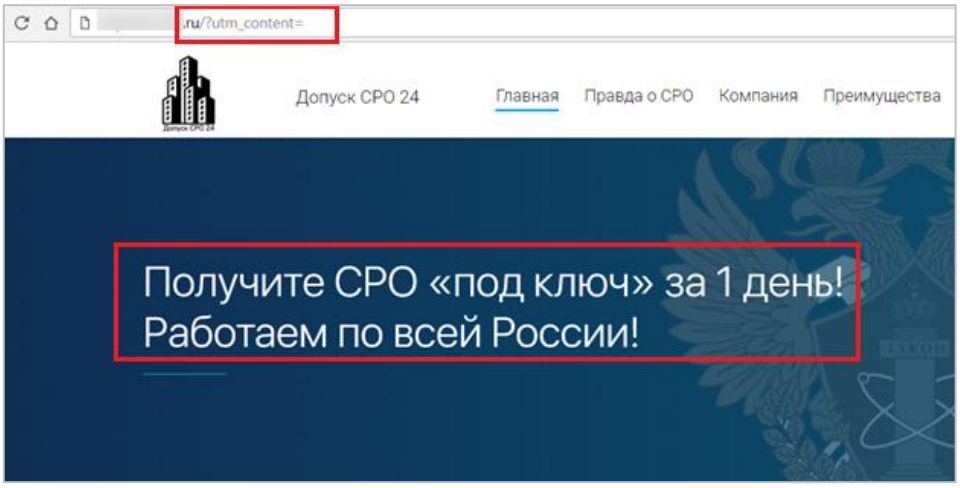


Рис. 994. Получите СРО под ключ за 1 день! Работаем по всей России!

Нам необходимо изменять этот заголовок в зависимости от параметра URL. В качестве параметра возьмем метку **utm_content** (можно взять любую другую, хоть нестандартную, например, *m_change*), в которую будем помещать значения других заголовков. Чтобы проще и быстрее реализовать задуманное, лучше всего подготовить заранее отдельную таблицу с метками и их значениями. Вот как это может выглядеть в Excel:

utm_content	Заголовок на сайте	Конечный URL
zag1	Получите СРО «под ключ» за 111 дней! Работаем по всей России!	site.ru/?utm_content=zag1
zag2	Получите СРО «под ключ» за 222 дня! Работаем по всей России!	site.ru/?utm_content=zag2
zag3	Получите СРО «под ключ» за 666 дней! Работаем по всей России!	site.ru/?utm_content=zag3
zag4	Никогда не получите СРО! Мы не работаем в принципе!	site.ru/?utm_content=zag4

Например, вы сделали пост в социальной сети или разметили рекламные кампании, разбили свой товар или услугу на категории (мебель -> один заголовок, кухни -> другой заголовок и т.д.). При переходе по одной из ссылок (столбец *Конечный URL*) пользователь будет видеть подмененный заголовок (из столбца *Заголовок на сайте*).

Ссылка обязательна должна содержать параметр запроса, иначе подмена не произойдет. Давайте перейдем к настройке в GTM. Последовательность действий:

- Создайте пользовательскую переменную типа **URL**;
- Название – **paramURL** (любое произвольное);
- Тип компонента – Запрос;
- Ключ запроса – **utm_content** (тот, который вы определили для себя, может быть другим);

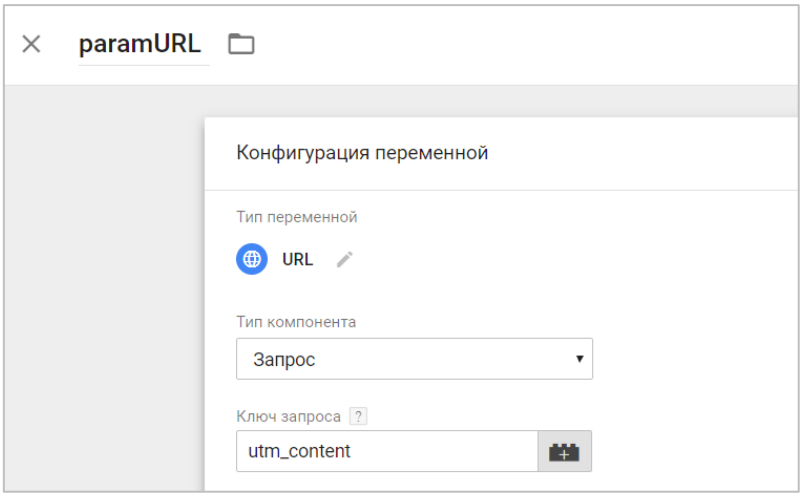


Рис. 995. Пользовательская переменная URL

Далее создайте еще одну пользовательскую переменную **Таблица поиска** из типа **Утилиты**.

- Название – **podmeniZag** (может быть любым);
- Входная переменная – **{{paramURL}}** (та, которую создали на шаге выше);
- Таблица поиска – в каждую из строк в поле **Входные данные** введите атрибут ключа запроса utm_content (см. таблицу выше), а в поле **Результат** введите значение параметра **utm_content**, то есть ваши подмененные заголовки;
- Установите значение по умолчанию (галочка) – добавьте исходный заголовок.

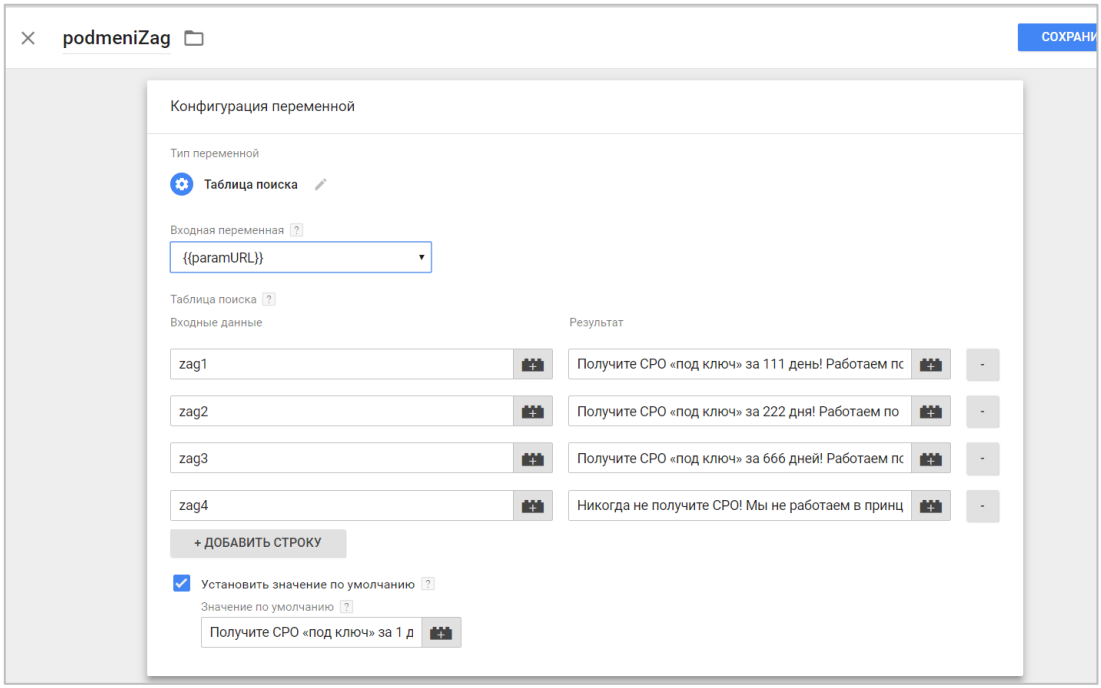


Рис. 996. Переменная Таблица поиска с нашими значениями

После этого создайте триггер типа **Просмотр страницы**, который будет активироваться только тогда, когда значение переменной utm_content определено. В противном случае если в ссылке нет параметра запроса, и оно принимает значение **undefined (не определено)**, то контент подменяться не будет, и тег не сработает.

- Название – **undefined** (может быть любым);
- Тип триггера – Просмотр страницы;
- Условие активации триггера – Некоторые просмотры страниц;

Активация триггера при наступлении события и выполнения всех этих условий: **paramURL не равно undefined**.

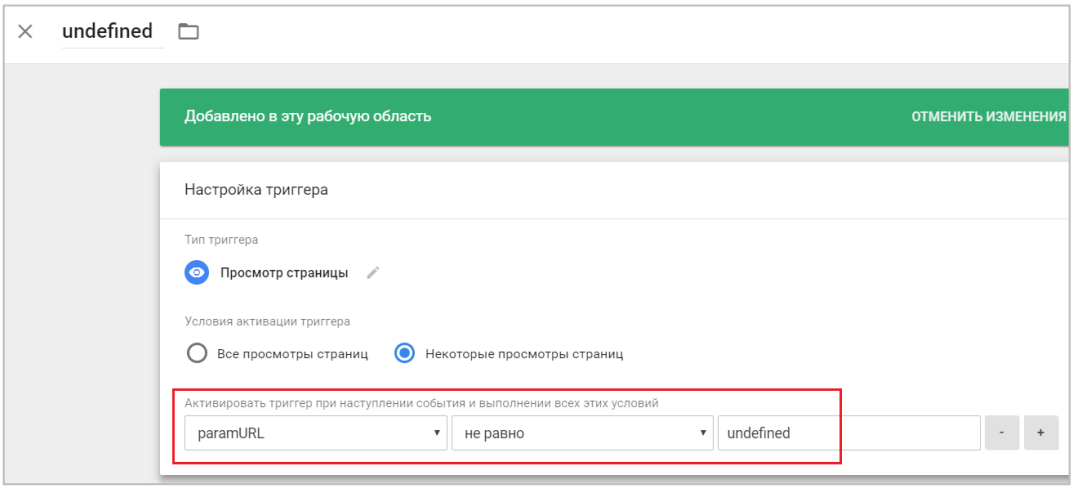


Рис. 997. paramURL не равно undefined

Сохраните триггер. Теперь осталось создать тег, который будет срабатывать при заданном условии активации триггера и подменять контент на сайте с помощью специального скрипта на значение табличной переменной.

- Создайте тег типа **Пользовательский HTML**;
- Название – **Tag – podmenaZag** (может быть любым);

Что же добавлять в текстовое поле? Какой фрагмент кода? Перед реализацией этого шага необходимо определить атрибуты заголовка, на основании которого вы будете производить изменения. Именно от этого будет зависеть реализация пользовательского HTML в Google Tag Manager.

Попробуем разобраться на конкретном примере. Перейдем на сайт и откроем консоль разработчика, чтобы посмотреть фрагмент кода нашего заголовка.

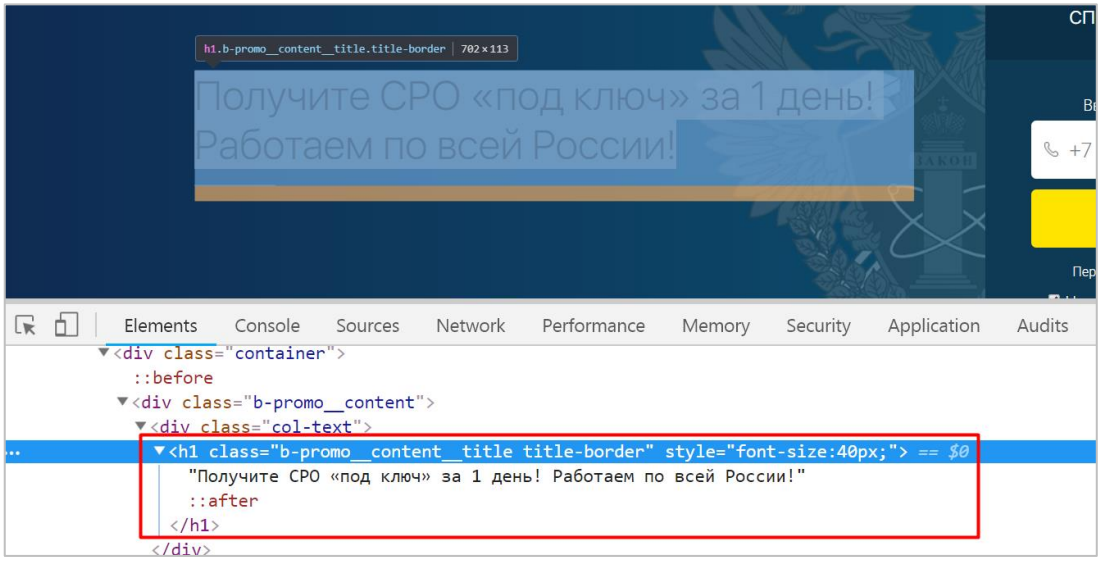


Рис. 998. Фрагмент кода заголовка в консоли разработчика

Если бы у данного заголовка h1 был атрибут **id** со значением, например, **zagolovok** (может быть другим), то фрагмент кода, который нужно вставить тег, выглядел бы так:

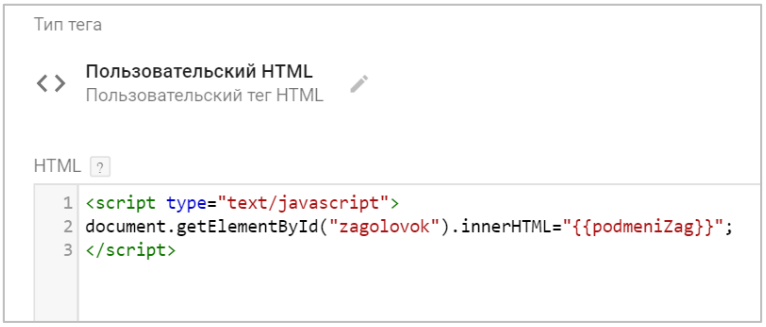


Рис. 999. Фрагмент JS-кода в случае, если бы у нас был атрибут id

В данном случае конструкция:

```
document.getElementById("zagolovok").innerHTML= "{{podmeniZag}}";
```

изменяет содержимое HTML-элемента h1 с **id = zagolovok** на новое из переменной **podmeniZag** (нашей таблицы поиска). Но у нас нет атрибута id, а есть **class**. Тогда необходимо использовать другую конструкцию для подмены. Вот пример реализации:

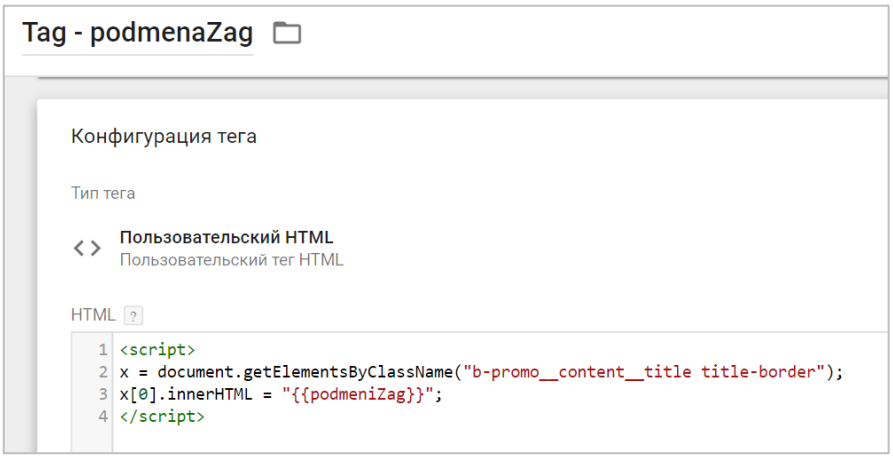


Рис. 1000. Фрагмент кода реализации через class

Сначала мы объявляем переменную x, которой присваиваем значение всех элементов, которые имеют заданные имена классов. Сам метод **getElementsByClassName ()** возвращает коллекцию дочерних элементов элемента с указанным именем класса.

В данном случае класс **b-promo__content__title.title-border** – это атрибут заголовка h1 в нашем примере. К узлам можно обращаться по номерам индексов. Индекс начинается с 0, поэтому в нашем примере x[0], то есть у 1 дочернего элемента изменяем HTML, подставляя пользовательскую переменную **podmeniZag** таблицы поиска.

Есть еще проще вариант. Использовать библиотеку jQuery. Тогда конструкция будет иметь вид:

Рис. 1001. Используем jQuery и функцию .html()

Функция **.html (newHTML)** в jQuery заменяет содержимое всех выбранных элементов на **newHTML**. В нашем примере она заменяет содержимое значение заголовка на переменную **podmeniZag**, которая содержит значения в таблице поиска.

Есть и такой вариант. Использовать метод **querySelector()**. Он возвращает первый элемент, который соответствует указанному CSS-селектору в документе.

Рис. 1002. Пользовательский HTML

Все зависит от конкретной задачи и способе реализации на вашем сайте. Потренироваться с вышеописанными методами можно здесь (см. приложение):

В теге мы выбираем условие активации триггера **undefined** и сохраняем тег.

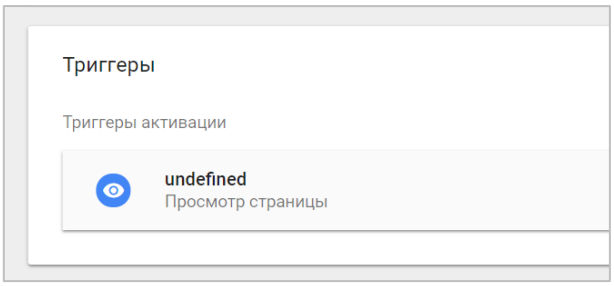


Рис. 1003. Условие активации тега — undefined

Проверяем корректность работы в режиме предварительного просмотра.

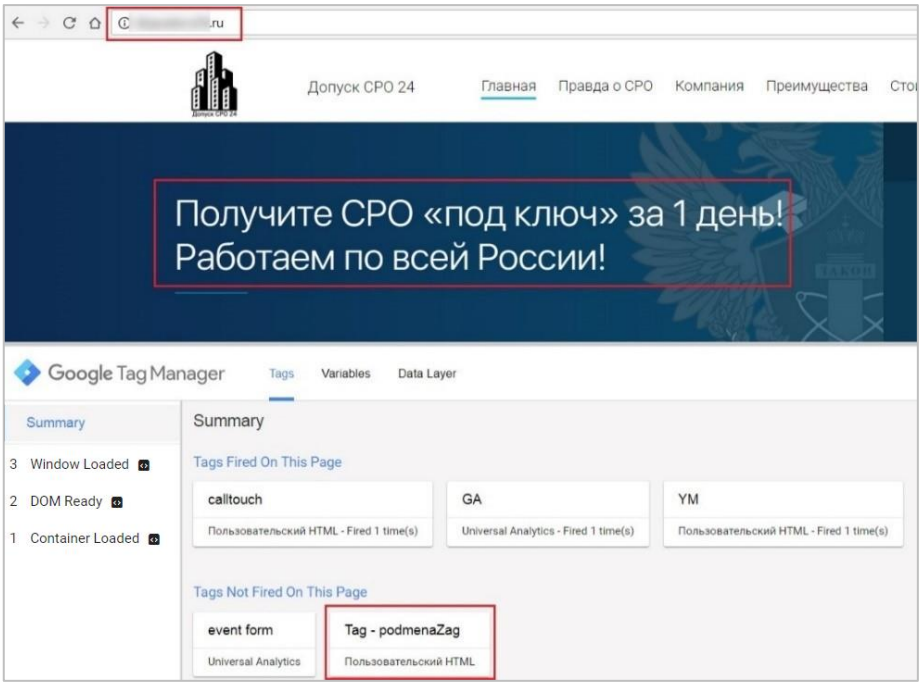


Рис. 1004. Тег не сработал, переменной utm_content нет

Если мы переходим без метки в адресной строке, то тег не активируется и заголовок не подменяется. Все правильно. Давайте добавьте к нашему url метку **utm_content=zag3**.

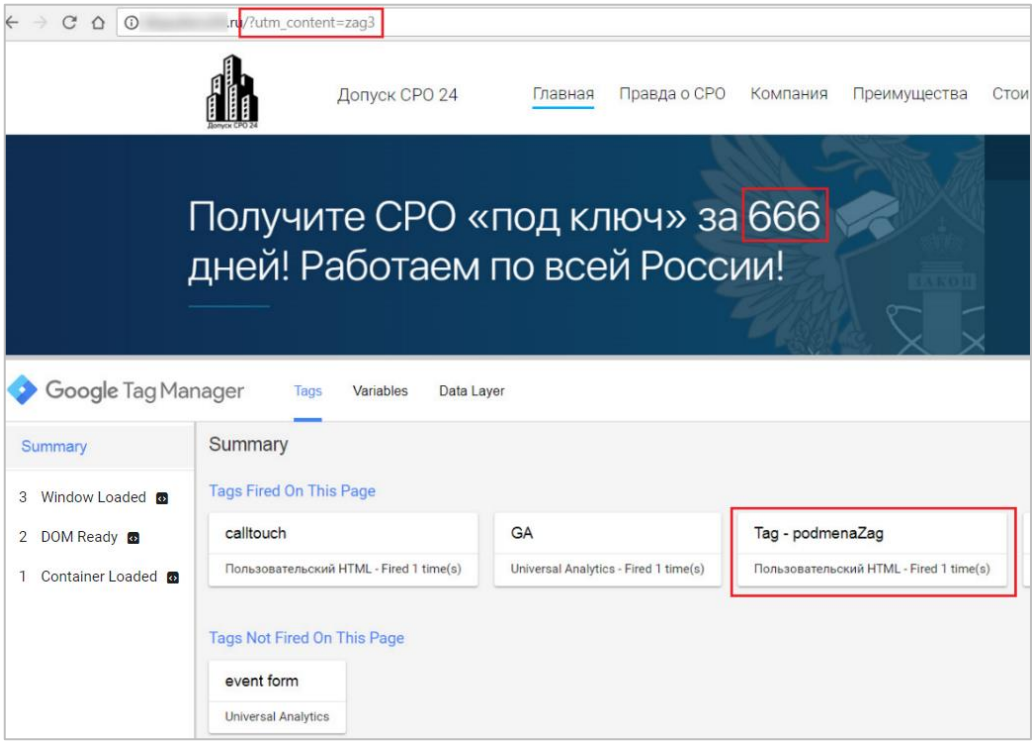


Рис. 1005. Заголовок 3: Получите СРО «под ключ» за 666 дней! Работаем по всей России!

Тег с меткой сработал, значение заголовка поменялось. Убедимся еще один раз. Теперь введем **utm_content=zag4**.

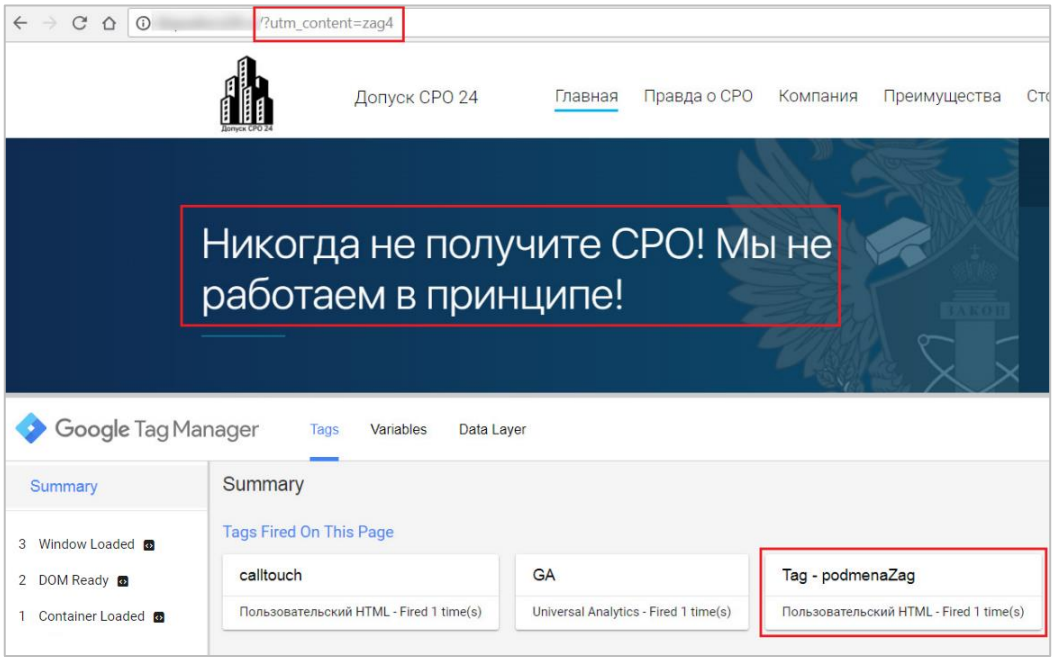


Рис. 1006. Заголовок 4: Никогда не получите СРО! Мы не работаем в принципе!

Все работает корректно. Публикуем контейнер GTM и радуемся подменам.

С помощью GTM вы можете настраивать правила, при котором на сайте будут подменяться не только заголовки, но и подзаголовки, номера телефонов в зависимости от источника переходов, изображения и многое другое.

Однако все же лучший способ реализации динамического контента на сайте – это серверная сторона (через тот же .php), а не браузерная, поскольку различные манипуляции с DOM могут привести к замедлению загрузки страницы у пользователя, да так, что он увидит эту подмену на сайте.

Допустимо использование такого приема в небольшом проекте (лендинге, несколько страничном сайте) и на малом объеме трафика (до 50-100 посетителей в день). Но если ежедневное количество пользователей гораздо больше, на сайте много заголовков и подмен, то рекомендуется реализовать все это через backend.

Плагины и расширения для браузеров

Для упрощения работы с Google Tag Manager, Google Analytics и улучшения производительности интернет-маркетолог/веб-аналитик в своей практике может использовать дополнительные инструменты - плагины и различные расширения для браузеров. Давайте разберем несколько самых популярных помощников.

Google Analytics Debugger

Расширение для браузера. Является надстройкой для консоли разработчика, в которую Debugger отправляет без изменения кода отслеживания информацию о том, что передается в Google Analytics, а также данные обо всех ошибках и сбоях. С помощью расширения можно не только проверить собственные страницы, но и узнать, как Analytics работает на других сайтах.

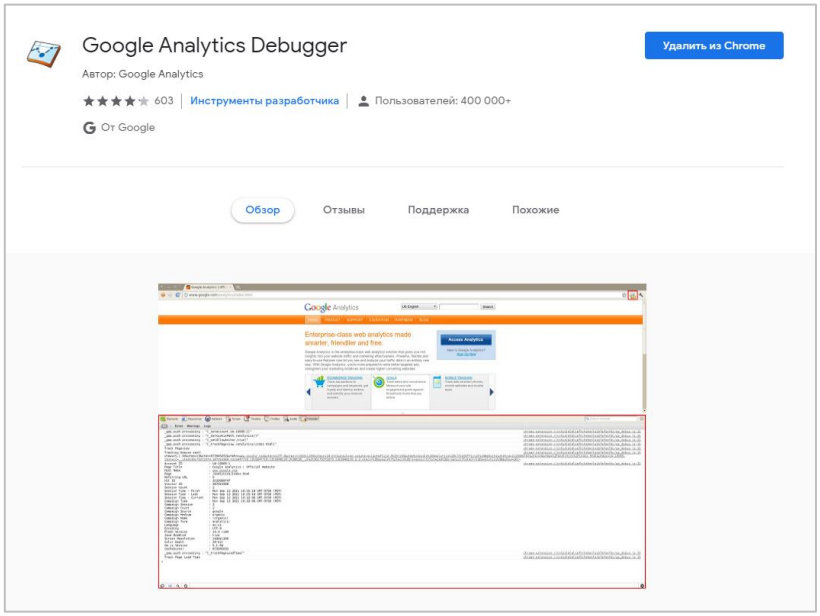


Рис. 1007. Google Analytics Debugger

GA Debugger часто используют при решении различного рода задач. Например, при настройке целей и электронной торговли, когда необходимо фиксировать передаваемые данные. Скачать расширение для браузера Google Chrome можно по ссылке (см. приложение)

Google Tag Assistant

Универсальный помощник. Расширение для браузера Google Chrome, которое позволяет записывать последовательность действий, выполняемых пользователями, поддерживает анализ тегов во многих продуктах Google — Ads, DoubleClick, Tag Manager. Он также помогает отслеживать правильность установки кода, находить и исправлять проблемы, которые могут привести к искажению данных в ресурсах Google Analytics.

Обладает функцией записи **(Google Tag Assistant Recordings, GTAR)**, которая может регистрировать теги, события и взаимодействия для любой последовательности посещенных страниц или сайтов, и тех страниц, которые открывал пользователь, даже если часть из них находится за пределами основного домена.

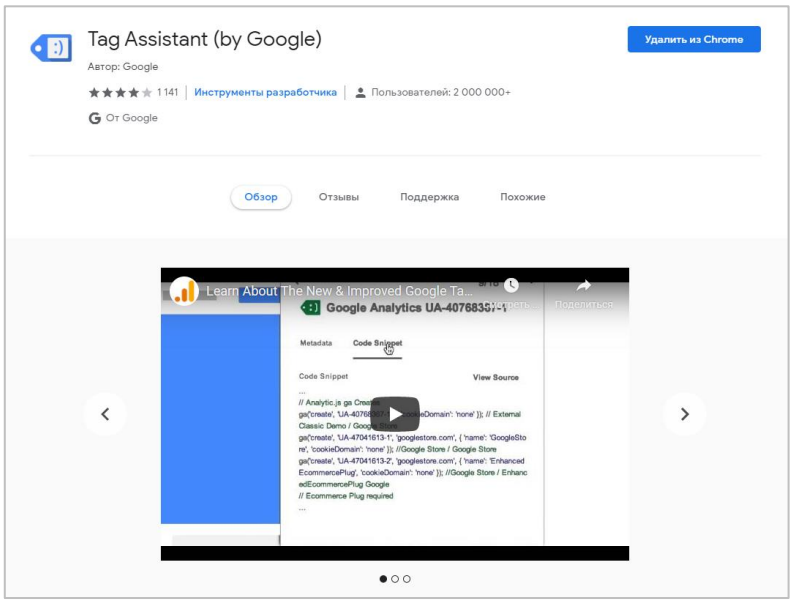


Рис. 1008. Google Tag Assistant

В моем блоге есть целая статья на эту тему (см. приложение). Скачать расширение для браузера Google Chrome можно по ссылке (см. приложение).

Web Analytics Solution Profiler (WASP)

Еще одно расширение для браузера, которое позволяет обнаружить теги любого типа, их взаимосвязь между собой и визуализировать полученные данные.

WASP запоминает состояние dataLayer после того, как вы покинули страницу и перешли на следующую. Этот функционал является большим плюсом, поскольку в режиме отладки GTM не позволяет видеть, что произошло на предыдущей странице.

Другие особенности:

- обнаружение всех типов тегов, включая те, которые отправляются через POST;
- SEO информация о странице (URL, title, description, keywords, cookie и др.);
- возможность скрытия и блокировки тега;
- декодирование информации для Google Analytics и других сервисов;
- удобная визуализация, которая показывает отношения между всеми тегами с помощью связей в виде стрелочек.

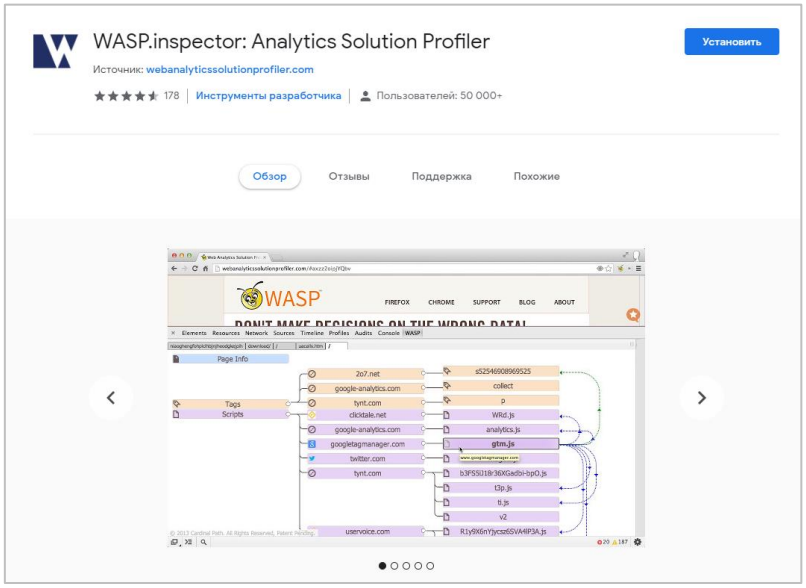


Рис. 1009. WASP.inspector

Скачать расширение можно по ссылке (см. приложение).

Tag Manager Injector

С 2019 года платное расширение (139 руб./год), которое позволяет начать работу с Google Tag Manager до того, как код контейнера будет размещен на страницах сайта. Это бывает особенно полезным, когда требуется начать работу над установкой GTM, но доступы по какой-либо причине заказчик или его команда оперативно предоставить не могут.

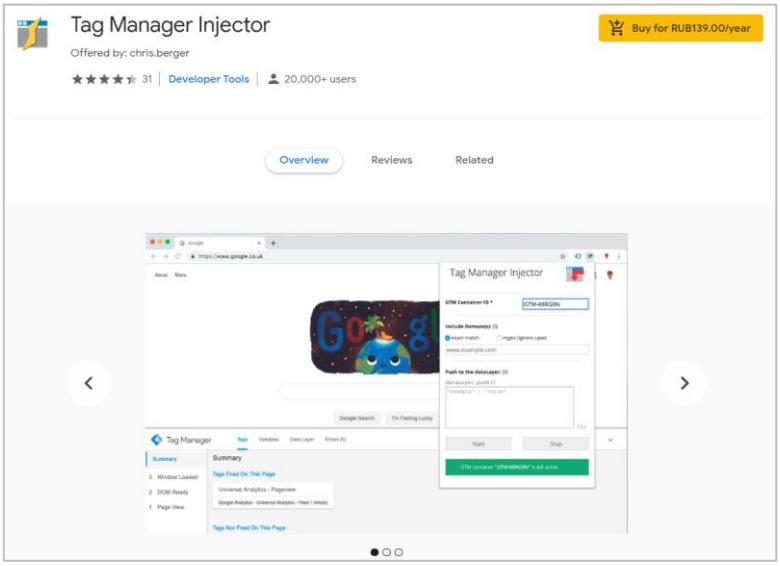


Рис. 1010. Tag Manager Injector

Скачать расширение для браузера Google Chrome можно по ссылке (см. приложение). После этого в правой верхней части появится новая иконка, при клике на которую открывается окно с настройками Tag Manager Injector:

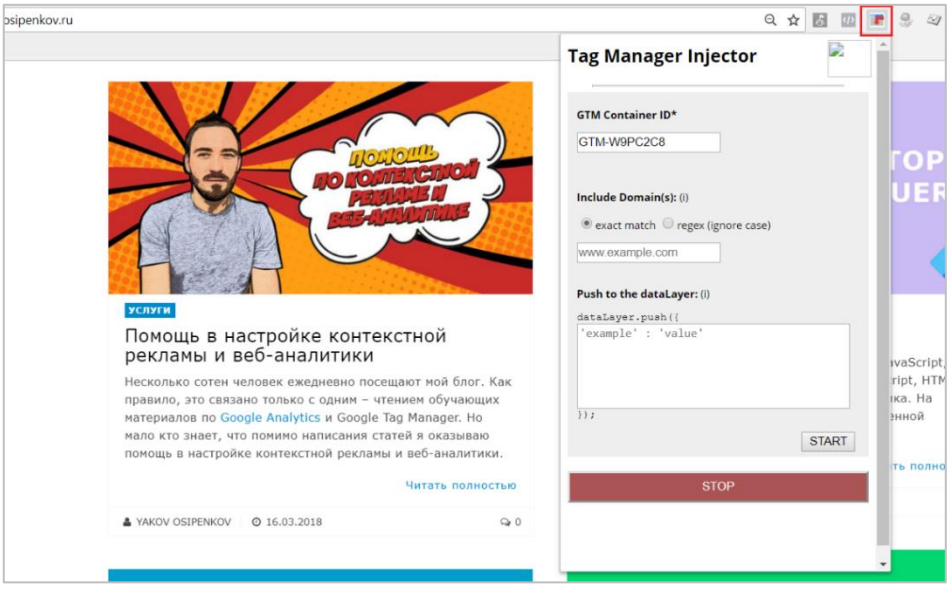


Рис. 1011. Настройки Tag Manager Injector

Затем необходимо прописать идентификатор контейнера GTM без каких-либо пробелов (поле **GTM Container ID**), а также домен (поле **Include Domain(s)**), на котором данный контейнер будет использоваться и нажать на кнопку **START**:

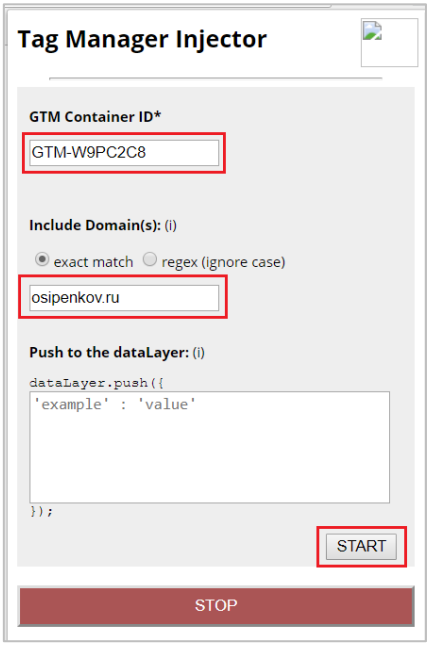


Рис. 1012. Активация Tag Manager Injector

Настройки TMI позволяют добавлять фрагмент GTM только на определенные страницы сайта. Для этого нужно переключить поле **exact match** на **regex (ignore case)** и указать те страницы, к которым вы хотели бы применить изменения. Без помощи регулярных выражений здесь не обойтись.

В результате произойдет перезагрузка текущей страницы, Tag Manager Injector напишет, что GTM контейнер активирован, а внизу страницы появится панель отладки:

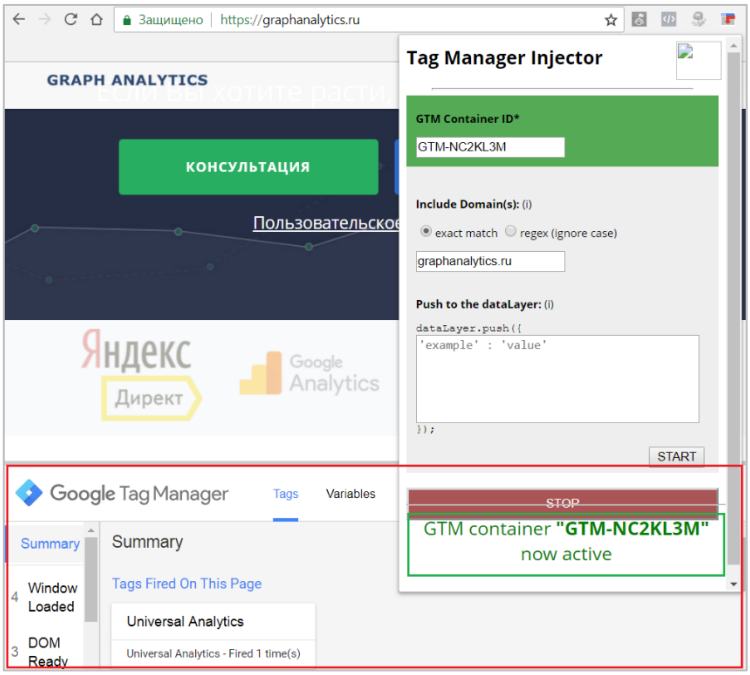


Рис. 1013. Tag Manager Injector активирован

Теперь можете выполнять настройки в контейнере, запускать предварительный просмотр и все ваши изменения будут доступны на сайте, доменное имя которого указано в настройках плагина.

Tag Manager Injector позволяет имитировать выполнение кода, передающего информацию в уровень данных. Для этого в соответствующее поле введите информацию в виде пары *ключ:значение*.

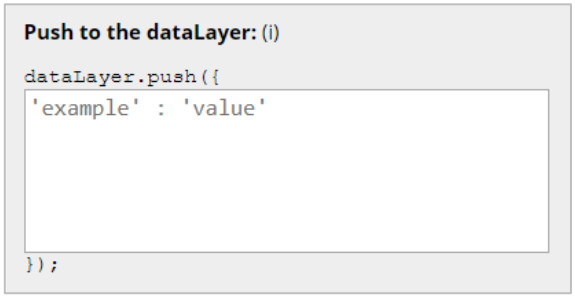


Рис. 1014. dataLayer.push

После завершения всех настроек отключите расширение с помощью кнопки **STOP**.

Рис. 1015. Остановка работы Tag Manager Injector

Важно: плагин искусственно устанавливает GTM - но только для вас, в вашем браузере. Поделиться информацией с другим человеком вы не сможете. Также описанные выше действия не заменяют собой процесс внедрения Google Tag Manager путем размещения его кода на всех страницах сайта. Если вы хотите, чтобы GTM корректно работал, необходимо добавить два фрагмента кода на отслеживаемые страницы сайта согласно руководству Google.

Tag Manager Injector может быть использован в режиме инкогнито. Существует бесплатный аналог данного расширения под названием **Injector**.

GTM Sonar

Расширение для браузера разработал эксперт по Google Tag Manager и Google Analytics, автор самого популярного блога в мире на эти темы, **Симо Ахава**.

Его творение, GTM Sonar (прошлое название **GTM Auto-Event Listener Debugger**) позволяет блокировать переход на другую страницу, но при этом формирует работу тегов таким образом, как будто вы совершили переход. Данные по действиям (кликам по элементам, по ссылкам или по формам) сохраняются в массив с именем **debugDL**, содержимое которого можно посмотреть в консоли разработчика.

Скачать расширение для браузера Google Chrome можно по ссылке (см. приложение).

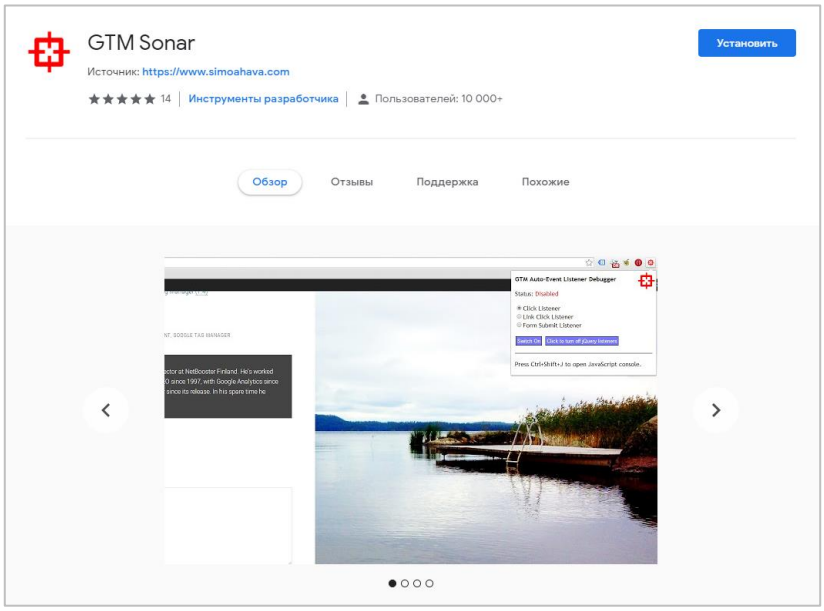


Рис. 1016. GTM Sonar

В правом верхнем углу на панели вы увидите иконку в виде красного прицела:

Рис. 1017. Иконка и настройки GTM Sonar

По умолчанию иконка красная и статус **Выключен (Disabled)**. Чтобы включить GTM Sonar, необходимо выбрать тип отслеживания: **Click Listener, Link Click Listener** или **Form Submit Listener** и нажать **Switch On**.

Если сопоставлять настройки с триггерами GTM, то это будет выглядеть так:

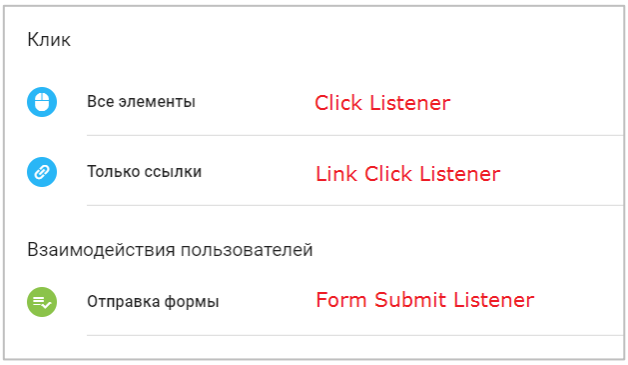


Рис. 1018. Сопоставление триггеров в GTM и функций GTM Sonar

Разберем пример взаимодействия пользователей с отправкой формы. Поскольку мы будем отслеживать форму, то в GTM Sonar выбираем **Form Submit Listener** и нажмите **Switch On**. Чтобы исключить отслеживание кликов с помощью jQuery, нажмите **Click to turn off jQuery listeners**.

В результате вы получите зеленый прицел и статус **Включен (Enabled)**:

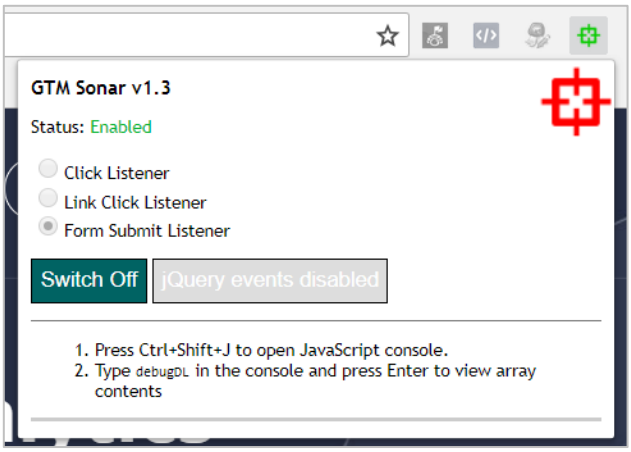


Рис. 1019. Активация GTM Sonar

Когда вы включаете GTM Sonar, происходит следующее:

- все действия по умолчанию, которые были осуществлены кликами мыши по странице, приостанавливаются;
- когда вы кликаете на элемент, информация о данном действии сохраняется в массиве **debugDL** в том же формате, что и в GTM.

Чтобы продемонстрировать это на конкретном примере, я сделаю тестовую заявку:

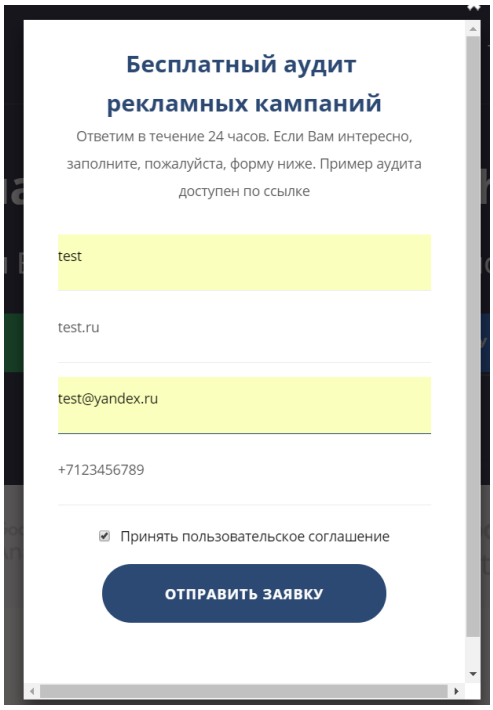


Рис. 1020. Пример тестовой заявки на сайте

Нажав на кнопку отправить, я получу уведомление о совершенном событии. Когда элемент добавлен в debugDL, счетчик в иконке браузера начинает свою работу, и данный счетчик показывает количество объектов в массиве **debugDL**. Появилась циферка 1:

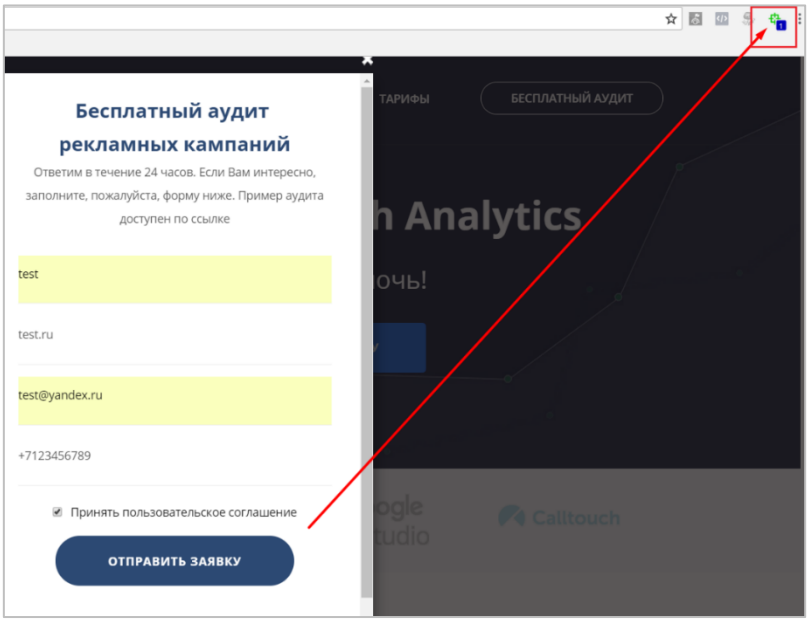


Рис. 1021. Количество объектов в массиве

Хотя в действительности я должен был увидеть вот это окно:

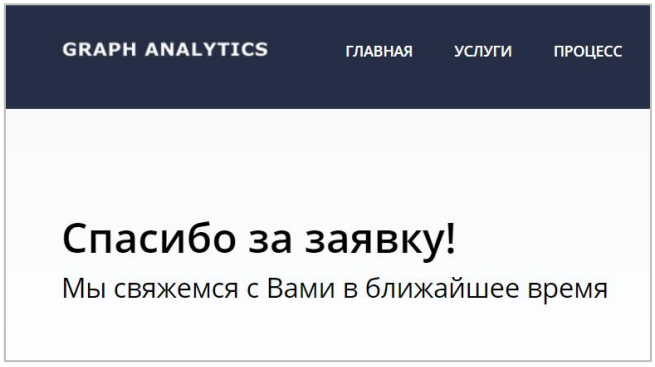


Рис. 1022. Это страница Спасибо должна была быть после отправки заявки

Для чего мы избегаем клики на странице? Дело в том, что при отладке было бы лучше, если бы вы находились на той же самой странице, не переключаясь на другую. Остановка действий по умолчанию предотвращает работу ссылок. По причине того, что **debugDL** — это объект в текущем документе, он будет работать только на той страничке, на которой вы находитесь.

Когда вы выключаете debugger, действия по умолчанию становятся активными. DebugDL массив не будет очищен, и вы можете наблюдать за содержимым массива в течение всего времени, пока будете находиться на данной странице.

Чтобы перейти к событию, откройте консоль разработчика на вкладке **Console**. Введите команду **debugDL** и нажмите **Enter**. Вы увидите свой массив с объектом:

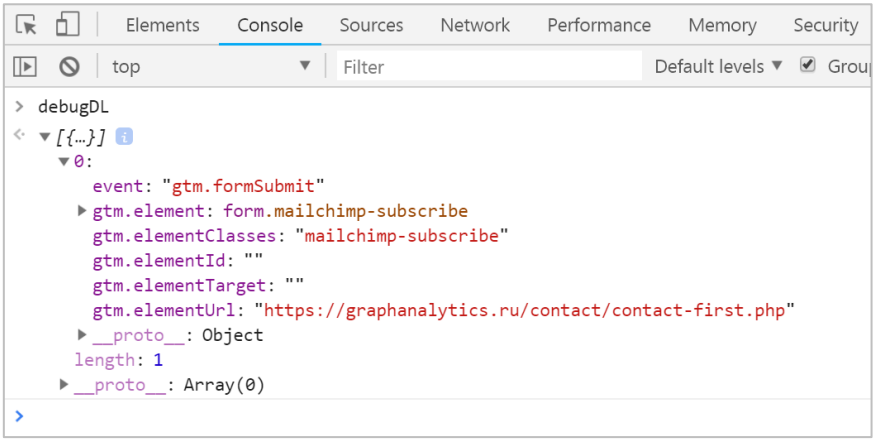


Рис. 1023. Содержимое debugDL после отправки формы

Теперь эту информацию можно использовать для отслеживания формы и передачи данных в GTM и Google Analytics.

Другие инструменты

Безусловно, есть еще целый ряд других полезных плагинов, дополнений и расширений для браузеров, например, **dataslayer, Datalayer Checker, Adswerve - dataLayer Inspector+, Da Vinci Tools** (см. приложение). Примеры с использованием некоторых из этих расширений были разобраны в некоторых главах руководства.

Injector

Данное расширение является бесплатной альтернативой Tag Manager Injector (TMI).

Что делает это расширение?

Как и Tag Manager Injector, **Injector** позволяет развернуть Google Tag Manager на любом сайте и проверить корректность настройки переменных, триггеров и тегов еще до того, как сам код контейнера будет размещен на страницах сайта. Причем сделать это можно с любым сайтом, не только со своим.

Когда можно использовать?

- вы можете не дожидаться, пока разработчик или заказчик установят код GTM на сайт согласно вашему заданию, а сразу приступить к работе;
- когда у заказчика уже есть настроенный Google Tag Manager, а вам нужно просто его донастроить (подправить). Но доступы оперативно предоставить он не может.

В этом случае варианта 2:

1. вы можете воспользоваться парсером контейнеров GTM SPY (см. в следующей главе) и выгрузить контейнер заказчика, импортировать его к себе, донастроить, выгрузить в JSON и загрузить обратно заказчику с объединением данных;
2. просто создать временный контейнер GTM для собственных настроек, протестировать его с помощью Tag Manager Injector или Injector, а затем также экспортировать и после предоставления доступов заказчиком загрузить в его контейнер новую конфигурацию с объединением данных.

Расширение Injector позволяет создавать собственные фрагменты кода для внедрения на любой веб-сайт. Injector поставляется с редактором для написания фрагментов кода, которые могут быть скриптами (JavaScript и CoffeeScript) или стилями (CSS, LESS).

Разберем пример развертывания контейнера GTM на сайте **leadjesus.ru**. На нем не установлен Google Tag Manager. Чтобы развернуть контейнер, необходимо:

1. Установить расширение по ссылке (см. приложение)

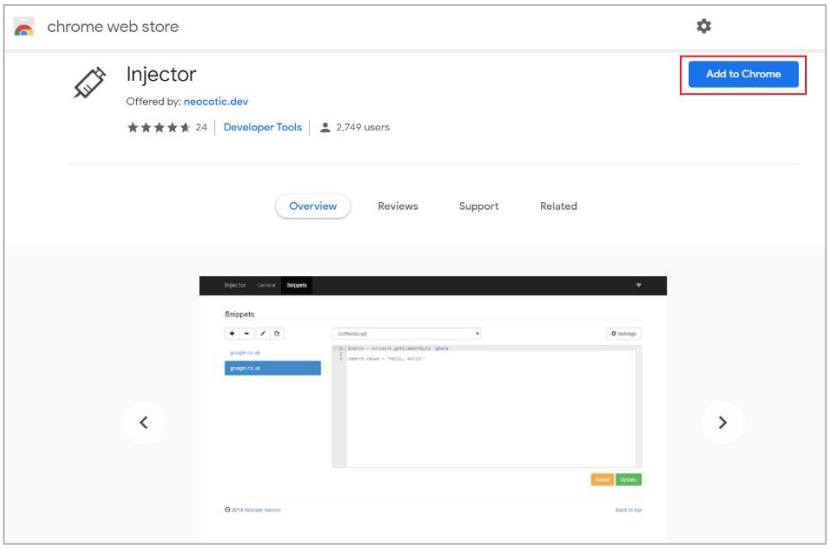


Рис. 1024. Injector

В правом верхнем углу браузера появится новая иконка. Активируйте расширение, нажав на значок.

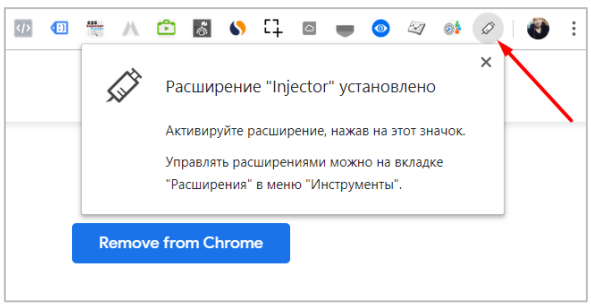


Рис. 1025. Активация расширения

Вас перенаправит на страницу расширения Injector. В окне инжектора нажмите на значок **«плюс» (+)**. Во всплывающем окне введите имя домена (не полный URL-адрес, например, leadjesus.ru) и нажмите кнопку **Create (Создать)**.

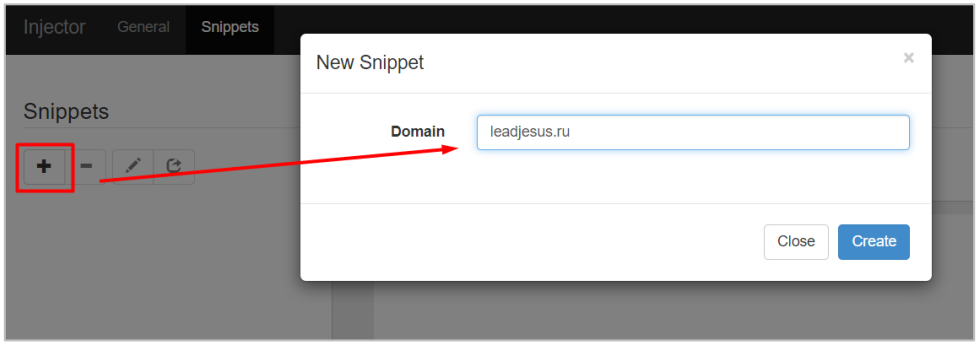


Рис. 1026. Создание проекта

На данном сайте не установлен Google Tag Manager. Переходим в интерфейс GTM и создаем аккаунт и контейнер для этого проекта. Как видим, изначально контейнер не опубликован.

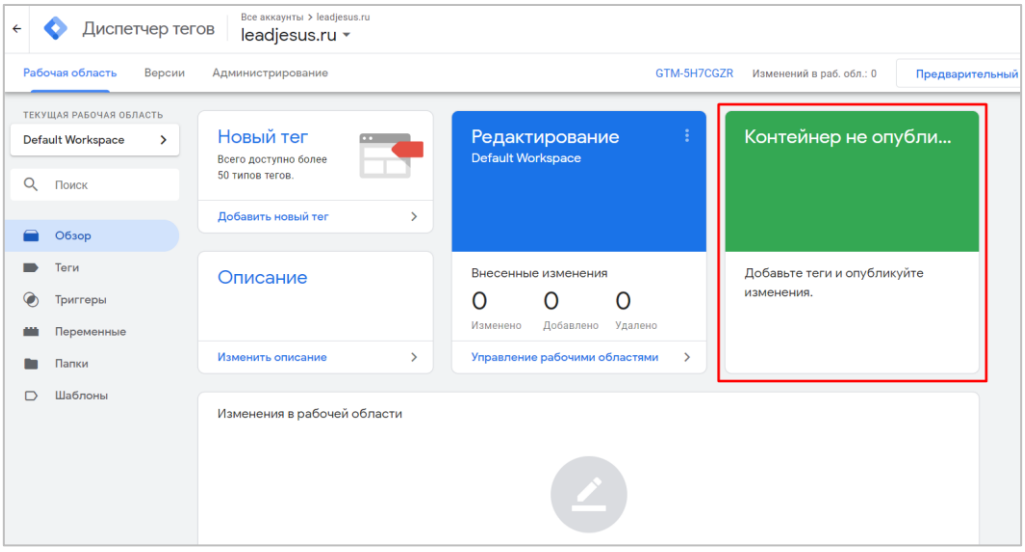


Рис. 1027. Создание контейнера

Нам нужен код GTM. Нажмите на идентификатор контейнера и копируем первый код.

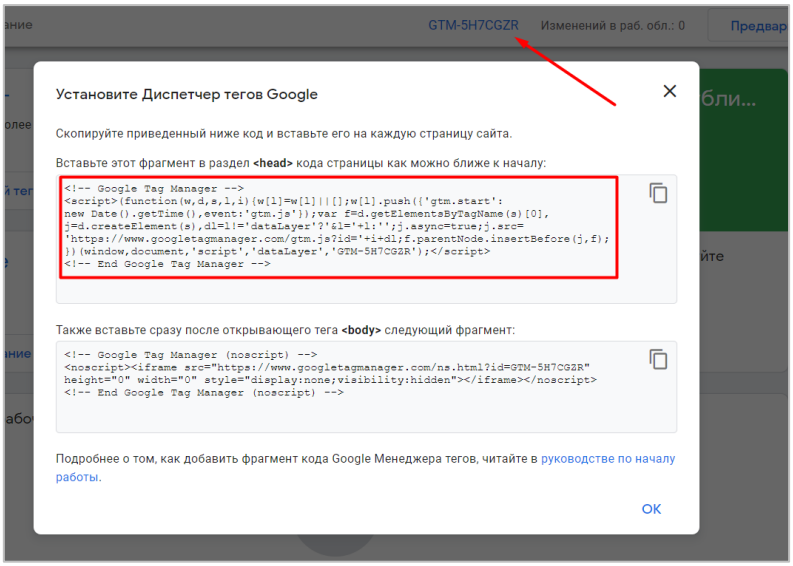


Рис. 1028. Копирование кода

Возвращаемся во вкладку нашего проекта Injector. Вставляем код в редактор, но удаляем теги **<script></script>** и **<!-- Google Tag Manager --> <!-- End Google Tag Manager -->**. Выглядеть должно так:

Рис. 1029. Добавление кода Google Tag Manager в Injector

Сохраняем настройки (кнопка **Save**). Теперь вы можете настроить переменные, триггеры и теги в Google Tag Manager, а после, активировав режим предварительного просмотра, и обновив страницу сайта, увидеть, какие из них работают и в какой момент.

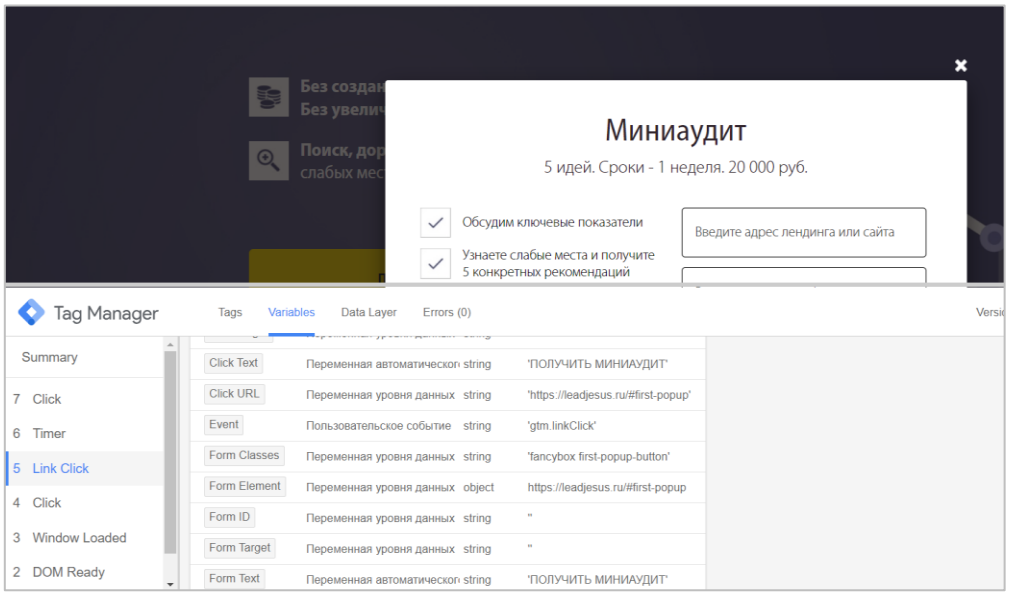


Рис. 1030. Проверка в режиме отладки

Как видите, у нас срабатывает триггер на таймер, отслеживаются клики по элементам и по ссылкам, а также определяются встроенные переменные. Хотя контейнер GTM не опубликован.

С помощью данного расширения можно развернуть код Google Tag Manager для тестирования на любом сайте без доступов к контейнеру. Всеми любимый Tag Manager Injector хоть и стал платным, но стоимость его использования ничтожна мала с тем, какую пользу это расширение приносит. Вы можете продолжать использовать TMI, приобрести подписку на год за 139 руб., а можете развернуть контейнер GTM с помощью расширения Injector. Альтернатива есть всегда. Выбор за вами!

GTM Variable Builder

Как часто вы хотели извлечь какую-либо информацию со своего сайта, но не могли этого сделать, поскольку незнакомы с элементами DOM и регулярными выражениями? Например, цену товара (но без знака валюты), его наименование, идентификатор транзакции, стоимость доставки? Или число найденных результатов на странице поиска? А может быть все переменные, которые необходимы для формирования уровня данных (dataLayer) динамического ремаркетинга или в электронной торговле?

Расширение для браузера **GTM Variable Builder** позволяет создавать пользовательские переменные типа **Собственный код JavaScript** и извлекать значения элементов сайта всего за 1 клик.

Чтобы скачать расширение, перейдите по ссылке (см. приложение) и нажмите **Установить:**

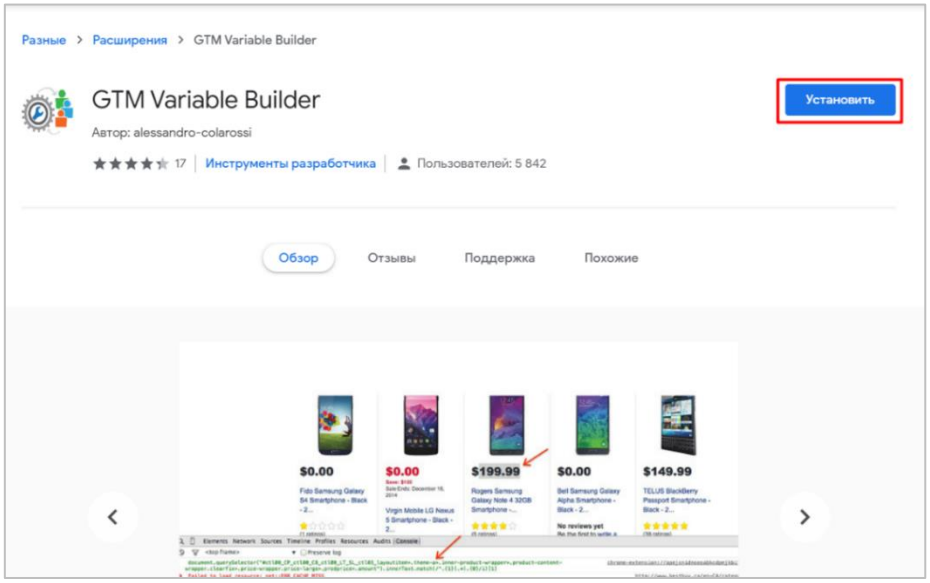


Рис. 1031. Расширение GTM Variable Builder

В правом верхнем углу появится иконка расширения. Далее следует перейти на сайт, открыть консоль разработчика (клавиша F12 в Google Chrome), выбрать вкладку **Console**.

Затем выделить нужный элемент на странице и нажать на иконку **GTM Variable Builder**.

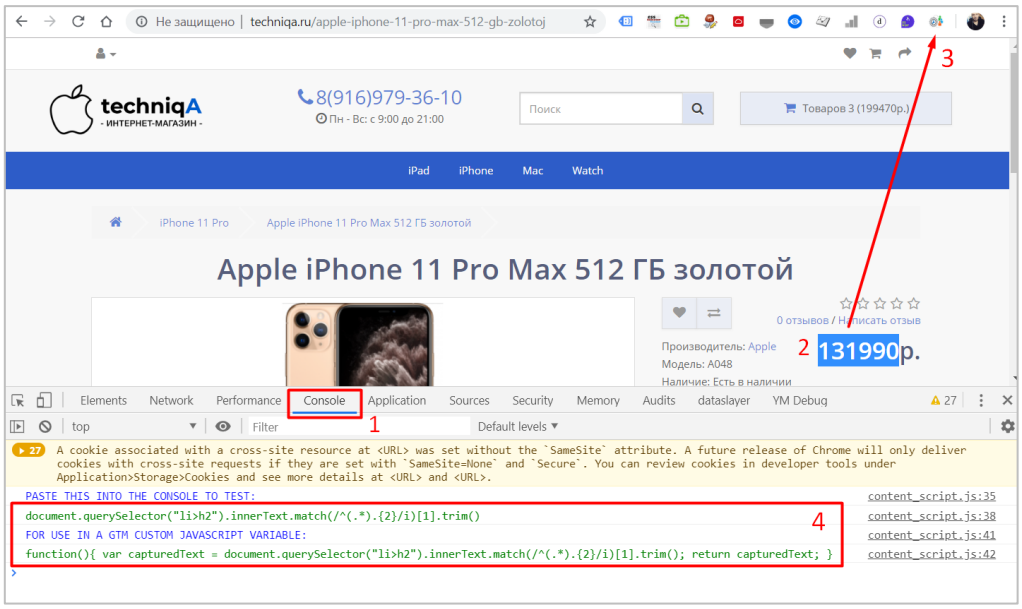


Рис. 1032. Код для переменной Google Tag Manager

В консоли появится две строчки:

- **PASTE THIS INTO THE CONSOLE TO TEST** – код для проверки нашей переменной;
- **FOR USE IN A GTM CUSTOM JAVASCRIPT VARIABLE** – код, который нужно скопировать и вставить в Google Tag Manager в переменную типа **Собственный код JavaScript**.

Чтобы проверить код переменной, скопируйте первую строчку, вставьте ее в консоль и нажмите **Enter**:

Рис. 1033. Проверка кода в консоли разработчика

Должен вернуться тот же результат. Если это так, копируйте вторую строчку с **function () {** и вставьте данный код в переменную GTM. Это будет выглядеть так:

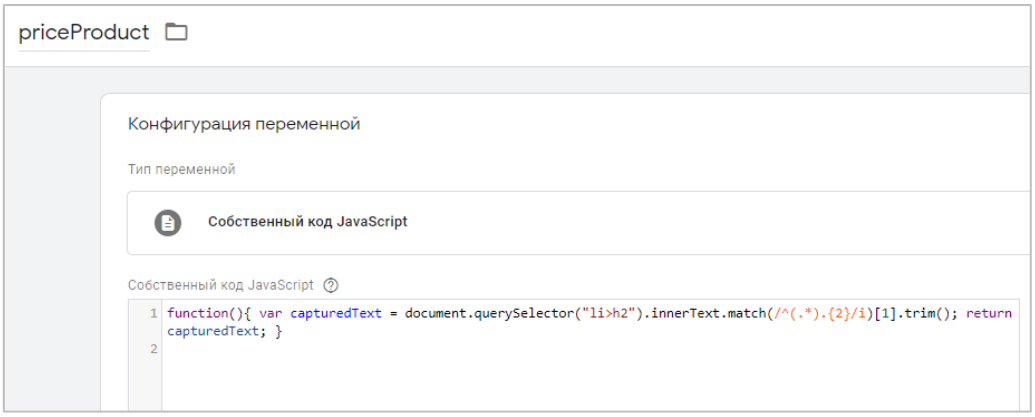


Рис. 1034. Переменная Собственный код JavaScript

Сохраняем нашу переменную. Чтобы проверить, действительно ли в данную переменную подставляется полученное значение, воспользуемся режимом предварительного просмотра Google Tag Manager. Перейдем на вкладку **Variables** и увидим значение нашей переменной:

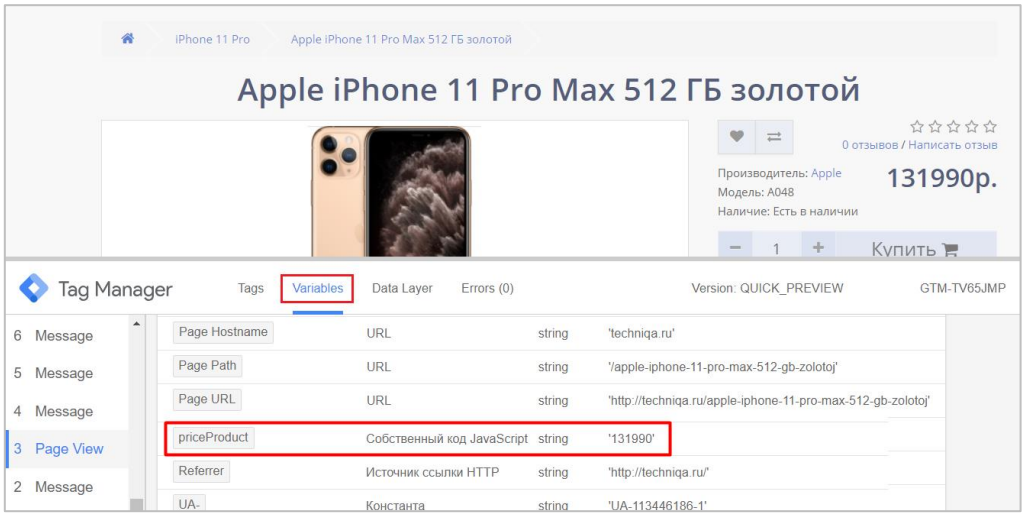


Рис. 1035. Значение переменной в режиме предварительного просмотра

Все работает. Благодаря GTM Variable Builder вы получили значение переменной без валюты (значка р.), которое автоматически создало код с регулярным выражением для нашей переменной. На других страницах переменная будет принимать значение соответствующего товара.

Расширение GTM Variable Builder очень сильно упрощают работу интернет-маркетологам с GTM при составлении собственных переменных, особенно тем, кто имеет ограниченные знания в JavaScript.

GTM SPY - парсер опубликованных контейнеров

Потеряли доступ к своему аккаунту, на котором зарегистрирован контейнер GTM? Хотите посмотреть, как настроен диспетчер тегов Google у конкурентов, чтобы найти идеи для отслеживаний на своем сайте? Встречайте, **gtmspy.com**! Сервис распарсит любой опубликованный контейнер Google Tag Manager на теги, триггеры и переменные и даст возможность скачать его в формате JSON, чтобы потом вы смогли импортировать его в свой GTM.

С помощью GTM SPY делается это очень просто - в поле **Start by entering a URL or GTM ID** необходимо вставить ссылку на сайт или **ID контейнера** Google Tag Manager и нажать на кнопку **Lookup Container**.

Рис. 1036. Введите URL сайта или ID контейнера GTM

Самый простой способ узнать ID чужого (своего) контейнера — это воспользоваться расширением Google Tag Assistant:

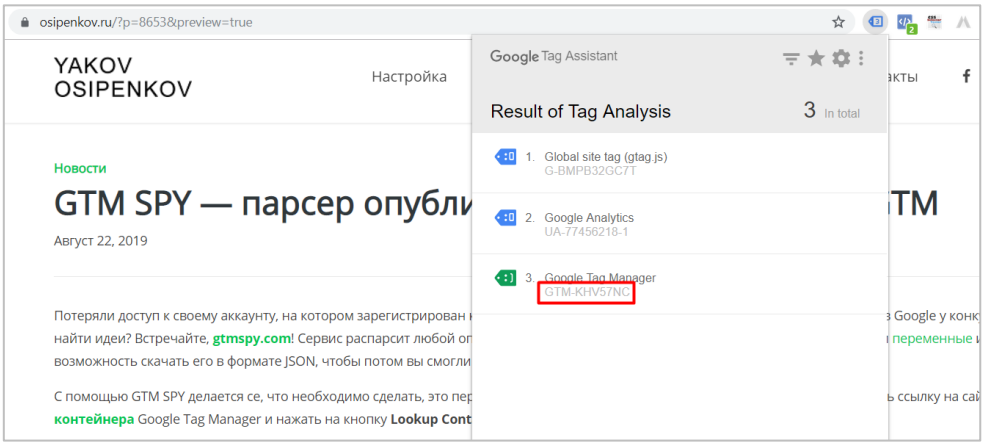


Рис. 1037. Google Tag Assistant

После нажатия кнопки начнется генерация данных:

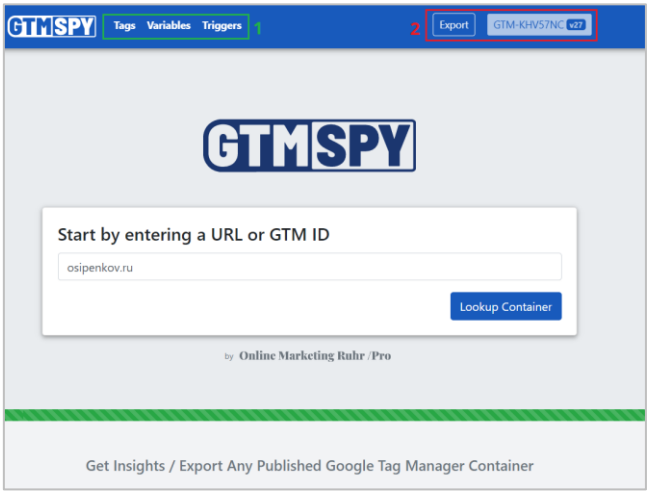


Рис. 1038. Парсер GTM SPY

Результатом будет являться:

1. вкладки **Tags, Variables, Triggers**, в которых будут отображаться все найденные сущности:

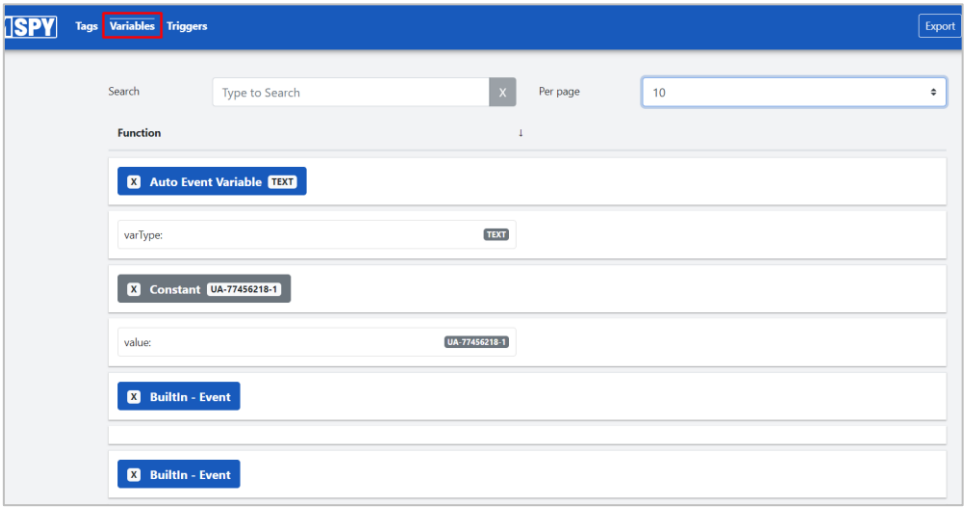


Рис. 1039. Примеры переменных (Variables)

2. последняя опубликованная версия контейнера (в моем примере v27) с возможностью скачивания (экспорта) в формате JSON с последующей загрузкой в собственный контейнер Google Tag Manager.

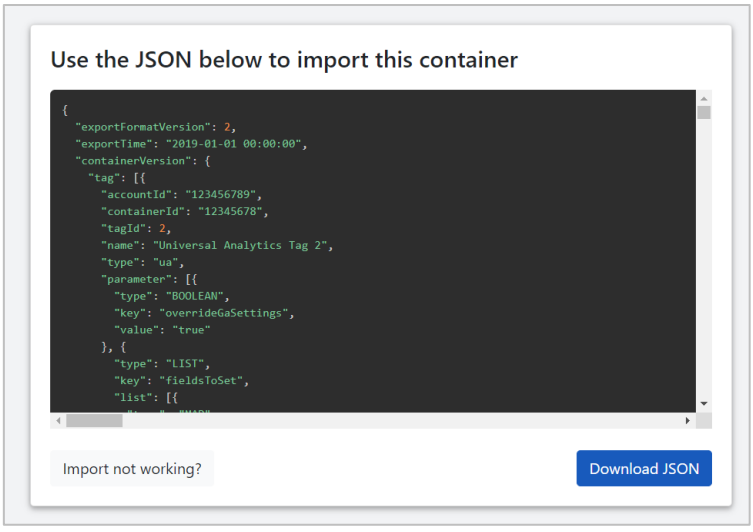


Рис. 1040. Скачать (Download) данные в формате JSON

Чтобы это сделать, перейдите на вкладку **Администрирование - Импортировать контейнер**.

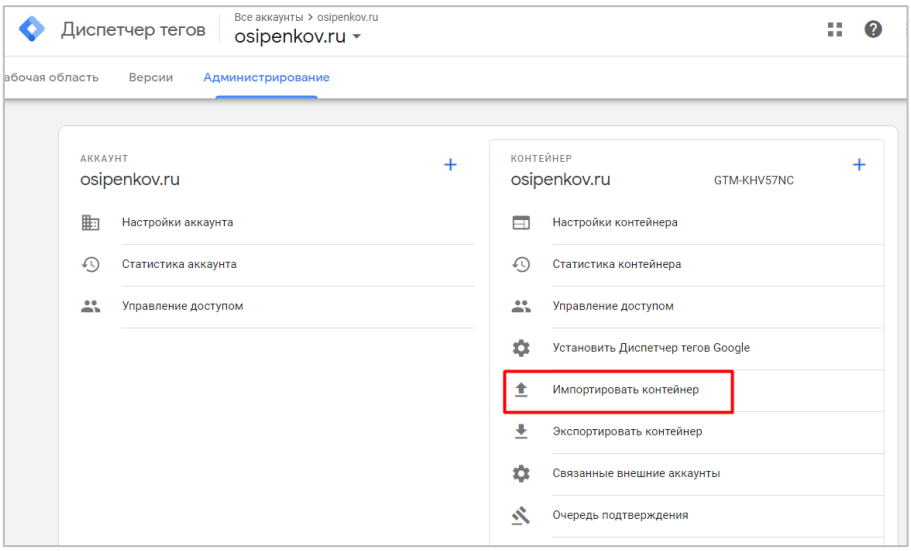


Рис. 1041. Импорт контейнера

Далее следует указать настройки импорта. Выбираем рабочую область и вариант импорта:

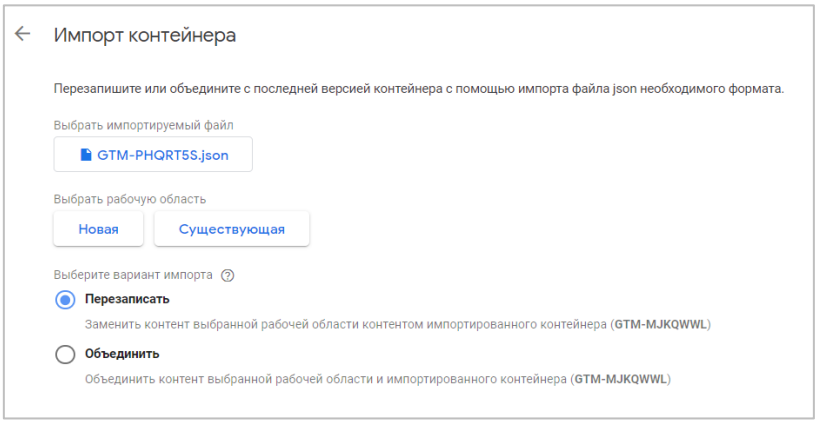


Рис. 1042. Настройки импорта контейнера

- **Перезаписать** - все данные в рабочей области будут заменены контентом из импортированного контейнера;
- **Объединить** - данные из рабочей области будут объединены с контентом из импортированного контейнера (ничего не будет заменено или удалено).

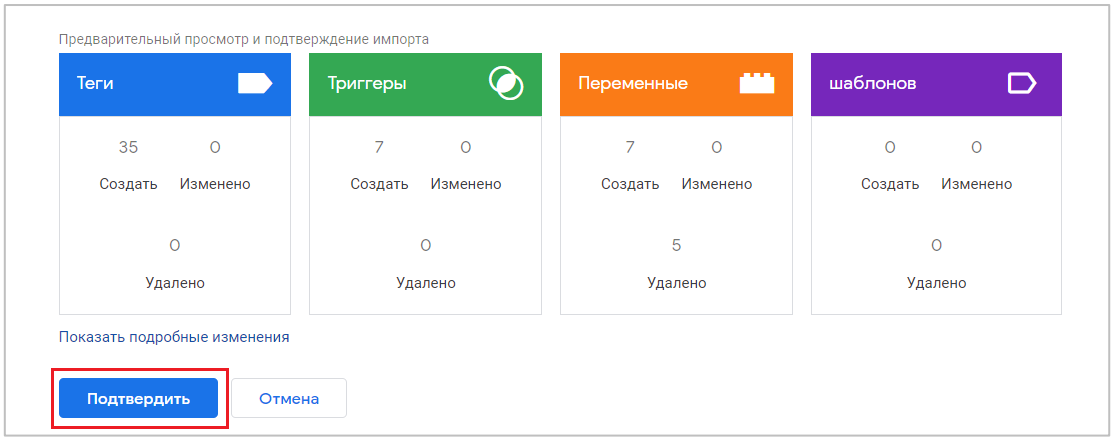


Рис. 1043. Предварительный просмотр перед импортом

Перед импортом нам еще раз покажут все сущности (теги, триггеры, переменные, шаблоны), которые будут импортированы в наш контейнер. Если все ок, нажмите **Подтвердить**.

На этом процессе импорта завершен. Все теги, триггеры, переменные и шаблоны появятся в выбранной рабочей области. Но есть одно примечание - имена всех сущностей будет не такими, как прежде или нам том сайте, с которого вы скачали контейнер. Это связано и с конфиденциальностью данных, и с самой генерацией JSON. Поэтому не пугайтесь, когда вы скачаете свой же контейнер и загрузите JSON в тестовый аккаунт. Я сделал это и увидел следующее:

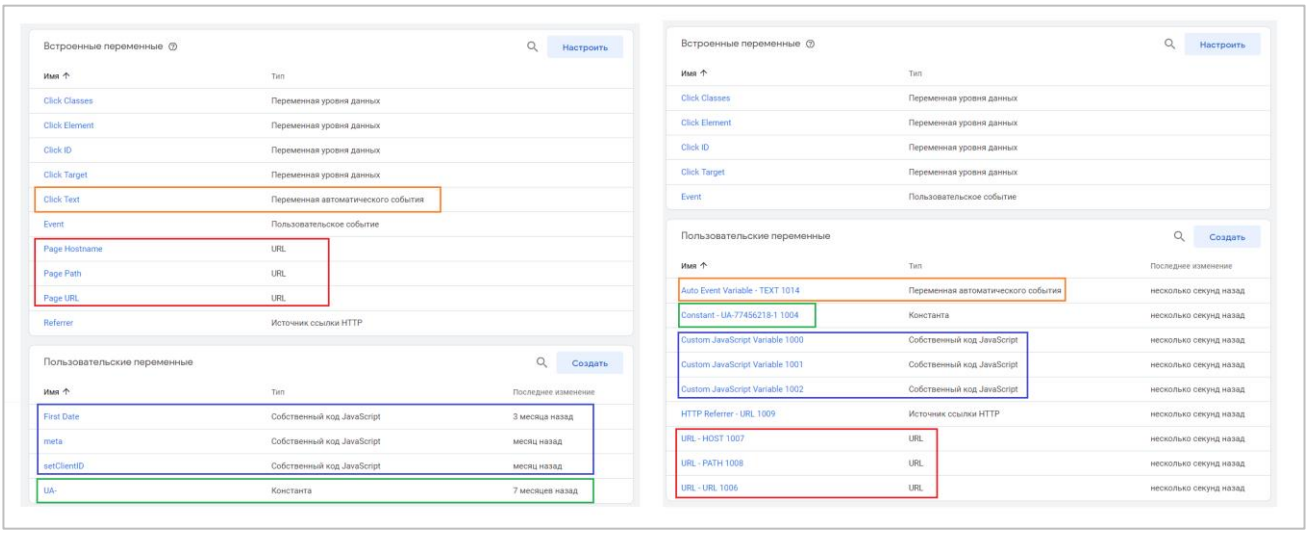


Рис. 1044. Сопоставление настроек собственного контейнера и импортированного

После импорта изменились названия всех объектов, некоторые встроенные переменные стали пользовательскими. В триггерах автоматически подгружаются базовые, в том числе и на просмотр страницы.

В тегах все аналогично. Их названия соответствуют типу выбранной конфигурации. Если это **Пользовательский HTML** тег, то его имя будет **Custom HTML Tag** + цифра в зависимости от количества тегов в аккаунте. Если это тип **Google Аналитика - Universal Analytics**, то имя тега **Universal Analytics Tag** + порядковый номер:

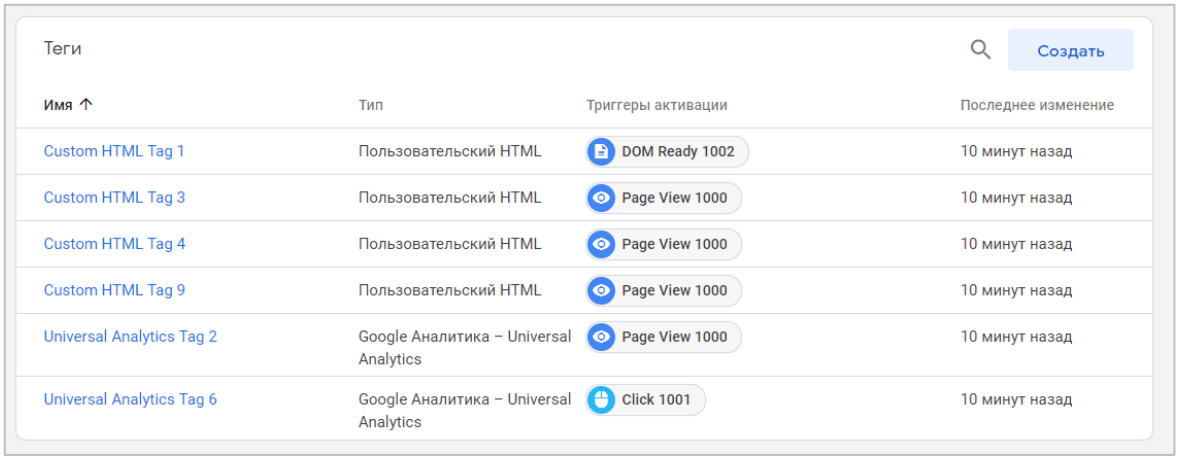


Рис. 1045. Примеры импортированных тегов

После всех изменений "под себя" опубликуйте новую конфигурацию. Вот и все!

GTM SPY - полезный инструмент при поиске новых идей на сторонних сайтах, а также незаменимый помощник в случае, если вы забыли доступ к своему аккаунту GTM. Просто скачайте файл в формате JSON и импортируйте в новый контейнер GTM. Все, что останется сделать — это переименовать объекты.

В заключение

На страницах этой книги мы с вами познакомились с большим количеством тем и настроек отслеживания, взяли управление тегов сторонних сервисов в собственные руки, снизили зависимость от разработчиков, и попутно, сами того не замечая, разрушили несколько мифов о диспетчере тегов Google. А именно:

- Google Tag Manager заменяет всех разработчиков / программистов;
- Google Tag Manager слишком сложен для интернет-маркетолога;
- Google Tag Manager небезопасен;
- Внедрение Google Tag Manager дорого и сложно;
- Google Tag Manager замедляет работу сайта.

Надеюсь, что эта книга гораздо лучше и информативнее предыдущей версии. Я открыт для сотрудничества и интересных предложений. Вы всегда можете написать мне на почту ya.osipenkov@icloud.com, через форму на сайте **osipenkov.ru** или в социальных сетях:

- ✓ YouTube: youtube.com/c/YakovOsipenkov
- ✓ ВКонтакте: vk.com/yakov.osipenkov;
- ✓ Facebook: facebook.com/yakov.osipenkov;
- ✓ Telegram: @YakovOsipenkov;
- ✓ Instagram: instagram.com/yakov.osipenkov.

Спасибо!

Отзывы читателей

Обратная связь от читателей всегда приветствуется. Дайте мне знать, что вы думаете об этой книге, что вам понравилось или не понравилось. Оставить отзыв можно несколькими способами:

- публично на странице https://www.facebook.com/pg/osipenkov.ru/reviews/ ;
- публично на странице https://vk.com/topic-115921590_42237485 ;
- написав мне на электронную почту ya.osipenkov@icloud.com с упоминанием названия книги в теме сообщения.

Если вам понравилась книга, просьба максимально распространить информацию о ее существовании. Так вы поддержите мои начинания и дадите мощный толчок для выпуска последующих руководств по веб-аналитике.

Обучение, курсы, книги

У вас остались вопросы по настройке диспетчера тегов Google?

Кроме YouTube-канала, на котором я публикую материалы по веб-аналитике, и блога **osipenkov.ru**, вы можете пройти мои онлайн-курсы, подобрав для себя оптимальную программу обучения на сайте edu.osipenkov.ru, а также написать по вопросу индивидуального обучения.

Для начинающих интернет-маркетологов и веб-аналитиков у меня разработаны различные программы обучения:

- по контекстной рекламе (в системах: Яндекс.Директ и Google Ads);
- по веб-аналитике (в сервисах: Microsoft Excel, Google Tag Manager, Яндекс.Метрика, Google Analytics, Google Data Studio, Microsoft Power BI).

Не забывайте про самообразование и совершенствование своих навыков. Изучайте веб-аналитику по моим бесплатным электронным руководствам. Среди них - как и собственные труды, так и переведенные книги с английского языка других авторов.

Подробнее по ссылке: osipenkov.ru/knigi

Приложение

Страница 13. Piggybacking

https://www.ensighten.com/blog/protect-your-brand-from-tag-piggybacking-risks/

Страница 14. Статистика similartech.com

https://www.similartech.com/categories/tag-management

Страница 37. Политики конфиденциальности и Условия использования Google

https://policies.google.com/privacy?hl=ru

Страница 49. Готовые решения для Google Tag Manager от Bounteous (ex LunaMetrics)

https://www.bounteous.com/insights/?category=resources--downloads/google-recipes

Страница 72. Жизненный цикл страницы

https://learn.javascript.ru/onload-ondomcontentloaded

Страница 77. Элементы HTML

https://developer.mozilla.org/ru/docs/Web/HTML/Element

Страница 86. Таблица псевдоклассов

https://puzzleweb.ru/css/selectors.php

Страница 87. Справочник по селекторам CSS

https://www.w3schools.com/cssref/css_selectors.asp

Страница 87. Примеры селекторов

https://www.w3schools.com/cssref/trysel.asp

Страница 87. Основы CSS-селекторов на примере котиков

https://frontender.info/basic-css-selectors-explained-with-cats/

Страница 87. Mozilla: CSS selectors

https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Selectors

Страница 92. Работа с набором элементов (jQuery)

http://jquery.page2page.ru/index.php5/

Страница 92. Селекторы jQuery

https://www.w3schools.com/jQuery/jquery_selectors.asp

Страница 92. Примеры jQuery

https://www.w3schools.com/jQuery/trysel.asp

Страница 93. Русская документация jQuery

https://jquery-docs.ru/

Страница 93. Регулярные выражения в Google Analytics

https://osipenkov.ru/regexp-google-analytics/

Страница 101. Отслеживание вложенных элементов

https://www.simoahava.com/analytics/use-wildcard-css-selectors-with-all-elements-triggers/

Страница 105. HTMLElement.dataset

https://developer.mozilla.org/ru/docs/Web/API/HTMLElement/dataset

Страница 108. Расширение CSS Selector Tester

https://chrome.google.com/webstore/detail/css-selector-tester/bbklnaodgoocmcdejoalmbjihhdkbfon

Страница 109. Расширение SelectorGadget

https://chrome.google.com/webstore/detail/selectorgadget/mhjhnkcfbdhnjickkkdbjoemdmbfginb

Страница 114. Типы данных в JavaScript

https://learn.javascript.ru/types

https://developer.mozilla.org/ru/docs/Web/JavaScript/Data_structures

Страница 119. Символы, итераторы и другие концепции JavaScript

https://proglib.io/p/simple-js-concepts/

Страница 122. Строгий Алгоритм Эквивалентного Сравнения

https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Operators/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80%D1%8B_%D1%81%D1%80%D0%B0%D0%B2%D0%BD%D0%B5%D0%BD%D0%B8%D1%8F

Страница 126. Побитовые операторы

https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Operators/Bitwise_Operators

Страница 128. Приоритет операторов

https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Operators/Operator_Precedence

Страница 138. Обработка ошибок try..catch

https://learn.javascript.ru/try-catch

Страница 143. History API

https://developer.mozilla.org/ru/docs/Web/API/History_API

Страница 143. Метод history.pushState()

https://developer.mozilla.org/ru/docs/Web/API/History/pushState

Страница 173. Руководство по расширенной электронной торговле для разработчиков (Universal Analytics)

https://developers.google.com/tag-manager/enhanced-ecommerce

Страница 178. Расширение GTM dataLayer Sifter

https://chrome.google.com/webstore/detail/gtm-datalayer-sifter/loaechmlaojdaklndmhapgkdnldjadbo?hl=ru

Страница 179. Изолированный JavaScript в пользовательских шаблонах GTM

https://developers.google.com/tag-manager/templates/sandboxed-javascript

Страница 182. Про пользовательские шаблоны

https://www.simoahava.com/analytics/custom-templates-guide-for-google-tag-manager/

Страница 207. Как переименовать объект ga

https://developers.google.com/analytics/devguides/collection/analyticsjs/renaming-the-ga-object

Страница 208. Теги и имя трекера

https://support.google.com/tagmanager/answer/2574372?hl=ru#TrackerName

Страница 226. Руководство по шаблону тега Yandex Metrica

https://www.thyngster.com/google-tag-manager-custom-template-yandex-metrica

Страница 227. Справочник методов Яндекс.Метрики

https://yandex.ru/support/metrica/objects/method-reference.html

Страница 235. Показатель отказов в Google Analytics

https://osipenkov.ru/pokazatel-otkazov-v-google-analytics/

Страница 242. Пользовательская настройка глубины прокрутки

https://www.simoahava.com/analytics/customize-scroll-depth-trigger/

Страница 245. Параметры проигрывателя YouTube

https://developers.google.com/youtube/player_parameters?hl=ru

Страница 250. Отслеживание загрузки файлов с заданными расширениями в Яндекс.Метрике

https://yandex.ru/support/metrica/objects/addfileextension.html

Страница 261. Про customTask

https://developers.google.com/analytics/devguides/collection/analyticsjs/tasks?hl=ru

http://www.adventum.ru/blog/customtask-how-to/

https://www.simoahava.com/analytics/customtask-the-guide/#how-tasks-work

Страница 264. Получает значение поля, хранящегося в счетчике

https://developers.google.com/analytics/devguides/collection/analyticsjs/tracker-object-reference

Страница 264. Код переменной для получения Client ID

https://www.simoahava.com/analytics/macro-magic-google-tag-manager/#7

Страница 267. Параметр custom_map

https://developers.google.com/analytics/devguides/collection/gtagjs/custom-dims-mets?hl=ru

Страница 270. Почему необходимо промечать неиспользуемые поля undefined?

https://www.simoahava.com/analytics/google-tag-manager-data-model/

Страница 275. Создание 4 специальных параметров

https://www.simoahava.com/analytics/improve-data-collection-with-four-custom-dimensions/

Страница 292. Отправка формы в Google Analytics для Contact Form 7 (WordPress)

https://contactform7.com/tracking-form-submissions-with-google-analytics/

Страница 296. AJAX для новичков

https://habr.com/ru/post/14246/

Страница 297. Бесплатный прослушиватель событий AJAX от Bounteous (ex LunaMetrics)

https://www.bounteous.com/insights/2015/08/27/ajax-event-listener-google-tag-manager/?ns=l

Страница 306. Как отправлять данные о достижении цели в Tilda

http://help-ru.tilda.ws/statistics-goal

Страница 306. Отслеживание отправки формы в GTM

https://tildatricks.com/gtm_events

Страница 308. Отправка данных о просмотре (hit) в Яндекс.Метрику

https://yandex.ru/support/metrica/objects/hit.html

Страница 315. Тег <input>

http://htmlbook.ru/html/input

Страница 317. Тег <select>

http://htmlbook.ru/html/select

Страница 323. Отслеживание чекбоксов

https://www.simoahava.com/analytics/form-tracking-google-tag-manager/

Страница 330. Отслеживание выпадающего списка

https://www.simoahava.com/gtm-tips/track-selection-drop-list/

https://www.simoahava.com/analytics/track-form-engagement-with-google-tag-manager/#8-capture-selected-drop-down-list-item-value

Страница 341. Интерфейсы веб API

https://developer.mozilla.org/ru/docs/Web/API

Страница 342. События Google Analytics

https://osipenkov.ru/event-google-analytics/

Страница 344. Код для отслеживания скопированного текста

https://dcarlbom.com/google-tag-manager/event-tracking-gtm-when-visitor-copies-text-from-page/

Страница 349. Электронная торговля в Google Analytics

https://osipenkov.ru/elektronnaya-torgovlya/

Страница 349. Настройка e-commerce для посадочной страницы

https://osipenkov.ru/ecommerce-landing/

Страница 352. Как подключить электронную коммерцию в Яндекс.Метрике

https://yandex.ru/support/metrica/data/e-commerce.html

Страница 353. Отслеживание электронной торговли с помощью Google Analytics

https://support.google.com/tagmanager/answer/6107169

Страница 372. Настройка расширенной электронной торговли с помощью Google Tag Manager

https://developers.google.com/tag-manager/enhanced-ecommerce?hl=ru

Страница 372. Справочные материалы для настройки Enhanced Ecommerce

https://developers.google.com/tag-manager/enhanced-ecommerce?hl=ru

https://enhancedecommerce.appspot.com/

Страница 374. Пример технического задания по настройке расширенной электронной торговли

https://osipenkov.ru/books/gtmbook/tz_ecommerce.pdf

Страница 377. Почему называются переменные и триггер gtm-ee-... ?

https://netpeak.net/ru/blog/kak-nastroit-rasshirennuyu-elektronnuyu-torgovlyu-s-pomoshch-yu-google-tag-manager/

Страница 383. Пример технического задания по настройке User ID

https://osipenkov.ru/books/gtmbook/tz_userid.pdf

Страница 383. Правила использования Measurement Protocol, SDK и User ID

https://developers.google.com/analytics/devguides/collection/protocol/policy

Страница 392. Поддерживаемые форматы файлов фидов данных

https://support.google.com/merchants/answer/160567?hl=ru

Страница 393. Принцип работы автоматических фидов

https://support.google.com/merchants/answer/7538732?hl=ru

Страница 394. Динамический ремаркетинг

https://osipenkov.ru/dinamicheskij-remarketing/

https://support.google.com/google-ads/answer/7305793?hl=ru

https://support.google.com/tagmanager/answer/6106009?hl=ru

Страница 397. Пример технического задания по настройке динамического ремаркетинга

https://osipenkov.ru/books/gtmbook/tz_dinrem.pdf

Страница 406. Введение в браузерные события

https://learn.javascript.ru/introduction-browser-events

Страница 408. Обработчики событий

https://learn.javascript.ru/introduction-browser-events#obrabotchiki-sobytiy

Страница 408. Про addEventListener

https://www.youtube.com/watch?v=EZPWupfomss

Страница 408. Код переменной для пользовательских событий

https://www.simoahava.com/analytics/custom-event-listeners-gtm/#gref

Страница 425. Отслеживание хеша в адресной строке браузера

https://yandex.ru/blog/metrika-club/4791

Страница 433. Файлы cookie в Google Analytics

https://osipenkov.ru/cookie-google-analytics/

Страница 433. Типы файлов cookie, используемые Google

https://policies.google.com/technologies/types?hl=ru

Страница 433. Временные файлы, устанавливаемые Яндекс.Метрикой

https://yandex.ru/support/metrica/general/cookie-usage.html

Страница 434. Новые правила отслеживания действий пользователей Google (от 3 февраля 2020 г.)

https://blog.chromium.org/2020/02/samesite-cookie-changes-in-february.html

Страница 435. Спецификация RFC 7234

https://tools.ietf.org/html/rfc7234#section-5.3

Страница 435. Куки, document.cookie

https://learn.javascript.ru/cookie

Страница 455. Хранение Client ID в localStorage для Google Analytics

https://www.simoahava.com/analytics/use-localstorage-client-id-persistence-google-analytics/

Страница 459. Счетчик просмотренных страниц. Способ №1

https://prometriki.ru/otslegivanie-nestandartnih-celey-v-google-analytics-s-pomoshu-google-tag-manager/

Страница 450. Счетчик просмотренных страниц. Способ №2

http://www.gtmscripts.com/uncategorized/pagecounter/

Страница 453. Счетчик просмотренных страниц. Способ №3

https://www.analyticsmania.com/google-tag-manager-recipes/3rd-page-view/

Страница 456. Код для переменной

https://datarunsdeep.com.au/blog/tracking-session-time-with-google-tag-manager

Страница 462. Отслеживание дублей транзакций

https://www.simoahava.com/analytics/prevent-google-analytics-duplicate-transactions-with-customtask/

https://www.thyngster.com/the-definitive-approach-for-preventing-duplicate-transactions-on-google-analytics-universal-customtask

https://www.bounteous.com/insights/2020/09/02/duplicate-transactions-google-analytics-and-app-web/

https://renta.im/blog/duplicate-transactions/

Страница 462. Measurement Protocol

https://osipenkov.ru/measurement-protocol/

Страница 462. Код события purchase

https://developers.google.com/tag-manager/enhanced-ecommerce#purchases

Страница 477. Отслеживание "кликов ярости" (Rage Clicks)

https://gist.github.com/silversillu/52ad72eaee5450ace862f39e52f6b953

Страница 484. JavaScript - объект window: свойства innerWidth, outerWidth и др.

https://itchief.ru/lessons/javascript/javascript-window-object-innerwidth-outerwidth

Страница 485. Медиазапросы

https://developer.mozilla.org/ru/docs/Web/CSS/Media_Queries/Using_media_queries

Страница 486. Рекомендованные значения медиазапросов

https://developer.mozilla.org/ru/docs/Web/CSS/Media_Queries/Using_media_queries

Страница 498. Отслеживание модальных окон

https://www.thyngster.com/track-alert-pop-ups-automatically-in-google-tag-manager

Страница 508. Метод getElementsByClassName()

https://www.w3schools.com/jsref/met_element_getelementsbyclassname.asp

Страница 508. Свойство innerHTML

https://www.w3schools.com/jsref/prop_html_innerhtml.asp

Страница 508. Свойство innerText

https://www.w3schools.com/jsref/prop_node_innertext.asp

Страница 510. Расширение Google Analytics Debugger

https://chrome.google.com/webstore/detail/google-analytics-debugger/jnkmfdileelhofjcijamephohjechhna?hl

Страница 511. Расширение Google Tag Assistant

https://chrome.google.com/webstore/detail/tag-assistant-by-google/kejbdjndbnbjgmefkgdddjlbokphdefk/

Страница 512. Расширение Web Analytics Solution Profiler (WASP)

https://chrome.google.com/webstore/detail/waspinspector-analytics-s/niaoghengfohplclhbjnjheodgkejpih

Страница 512. Расширение Tag Manager Injector

https://chrome.google.com/webstore/detail/tag-manager-injector/ooninanccdmjbcmghimhdfpeklpmlllg?hl=en-GB

Страница 514. Расширение GTM Sonar

https://chrome.google.com/webstore/detail/gtm-sonar/iiihoahkpncaheicjfemhjkkfamcahcd?hl=ru

Страница 517. Расширение dataslayer

https://chrome.google.com/webstore/detail/dataslayer/ikbablmmjldhamhcldjjigniffkkjgpo

Страница 517. Расширение Adswerve - dataLayer Inspector+

https://chrome.google.com/webstore/detail/adswerve-datalayer-inspec/kmcbdogdandhihllalknlcjfpdjcleom

Страница 517. Расширение Da Vinci Tools

https://chrome.google.com/webstore/detail/da-vinci-tools/pekljbkpgnpphbkgjbfgiiclemodfpen/related?hl=ru

Страница 517. Расширение Injector

https://chrome.google.com/webstore/detail/injector/bfdonckegflhbiamlmidciapolfccmmb

Страница 520. Расширение GTM Variable Builder

https://chrome.google.com/webstore/detail/injector/bfdonckegflhbiamlmidciapolfccmmb