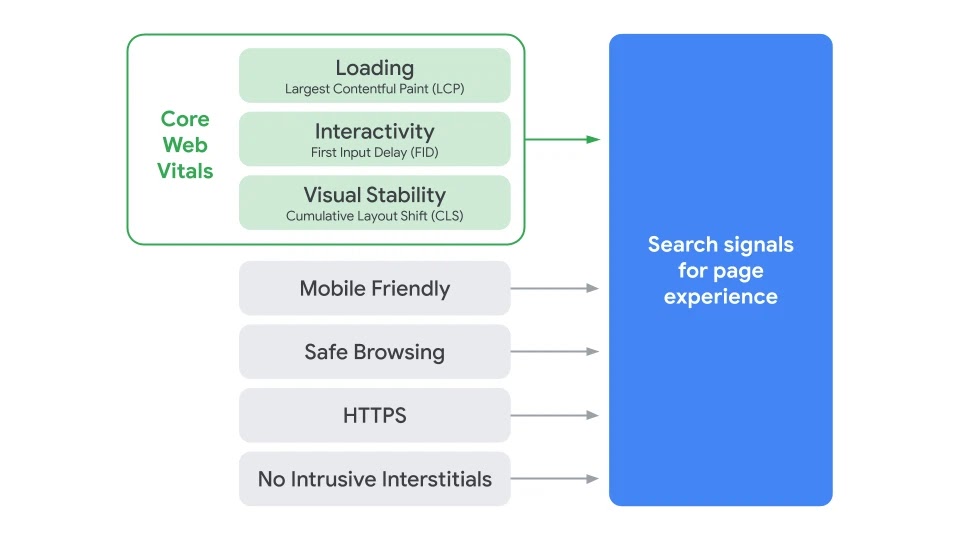
# 

# **<h1>Core Web Vitals - новый критерий ранжирования Google и как теперь контролировать веб-производительность</h1>**

Google объявил, что с июня 2021 года они начнут рассматривать [“Page Experience” как часть ранжирования в поиске](https://developers.google.com/search/blog/2020/11/timing-for-page-experience), измеряемого набором показателей, называемых Core Web Vitals.



***Значение атрибута alt для изображения: Сигналы, которые отражают удобство страницы***

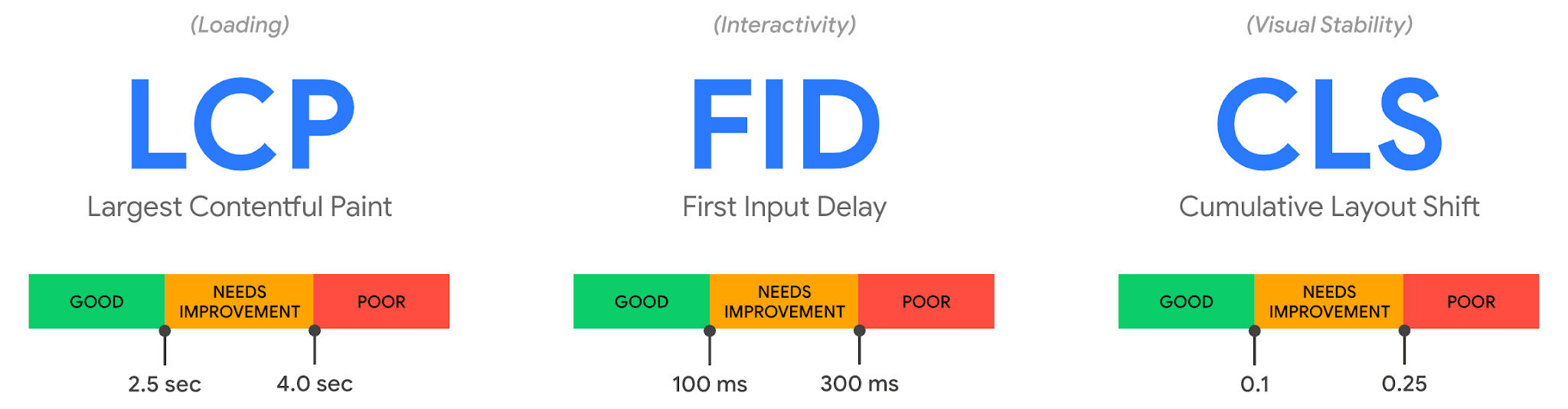
*Core Web Vitals как фактор, влияющий на ранжирование страницы в поиске Google*

Как видим на рисунке выше [оптимизация для мобильных устройств](https://webmasters.googleblog.com/2015/02/finding-more-mobile-friendly-search.html), [безопасный просмотр](https://webmasters.googleblog.com/2016/09/more-safe-browsing-help-for-webmasters.html), [обеспечение безопасности с помощью протокола HTTPS](https://webmasters.googleblog.com/2016/11/heres-to-more-https-on-web.html) и [соблюдение правил в отношении навязчивых межстраничных объявлений](https://webmasters.googleblog.com/2016/08/helping-users-easily-access-content-on.html) по-прежнему играют важную роль в выдаче.

Однако, в данной статье мы заострим внимание именно на Core Web Vitals.

Данный критерий является довольно сложным, и хотя в данный момент для нас доступно множество инструментов, раскрывающих Core Web Vitals, существует немало важных концепций и тонкостей, которые необходимо понять. Даже такие инструменты Google, как [PageSpeed ​​Insights](https://developers.google.com/speed/pagespeed/insights/) и [отчет Core Web Vitals в Google Search Console](https://support.google.com/webmasters/answer/9205520) дают запутанную информацию. Давайте разбираться.

**<h2>Что такое “Core Web Vitals”?</h2>**

[Core Web Vitals](https://web.dev/vitals/) - это набор из трех показателей, предназначенных для измерения пользовательского опыта работы с веб-сайтом. Именно они формируют вывод - насколько быстро будет открываться сайт.

***Значение атрибута alt для изображения: 3 главные метрики Core Web Vitals***

*3 главные метрики Core Web Vitals*

1. Largest Contentful Paint - показатель загрузки. Для обеспечения удобства работы пользователей, LCP должен сработать в течение первых 2,5 секунд после того, как страница начала загружаться.

#### First Input Delay (FID) - показатель интерактивности. Для обеспечения удобства работы пользователей, показатель FID должен быть меньше или равен 0.1 секунды.

#### Cumulative Layout Shift (CLS) - показатель визуальной стабильности. Для обеспечения удобства работы пользователей, показатель CLS должен быть меньше или равен 0.1.

В целом, результаты по 3-м показателям должны быть в пределах зеленых зон. За пределами зеленой зоны значения показателей Core Web Vital [могут привести к разному ранжированию страницы](https://support.google.com/webmasters/thread/104436075/core-web-vitals-page-experience-faqs-updated-march-2021). Рассмотрим каждый из критериев подробнее.

**<h3>1. LARGEST CONTENTFUL PAINT (LCP)</h3>**

Этот показатель, вероятно, самый простой для понимания. Он измеряет, как быстро пользователь получает самый тяжелый элемент контента, отрисованный на странице.

Зачастую этими элементами являются:

* тег <video>,
* тег <img>,
* тег <image> внутри <svg>,
* элемент с CSS-свойством “background-image” и атрибутом “url”,
* атрибут “poster“ в теге <video>,
* блокирующий поток элемент, содержащий текст.

Тот факт, что это [самый крупный элемент](https://web.dev/lcp/#what-elements-are-considered) на странице, свидетельствует о том, что в то же время это самый важный элемент. LCP является относительно новым критерием. Ранее мы в первую очередь заостряли свое внимание на одноименном First Contentful Paint (FCP), но в данный момент LCP рассматривается как более эффективный показатель отрисовки контента, который пользователь захочет увидеть в первую очередь.

LCP измеряет производительность загрузки и является прокси для всех старых метрик производительности, которые мы ранее использовали (например, время до получения первого байта от сервера (TTFB), время загрузки DOM, время начала рендеринга, индекс скорости). Он охватывает далеко не всю информацию, которая содержится в этих показателях по отдельности, но при этом представляет собой единую, более понятную метрику, которая дает хорошее представление о загрузке страницы.

#### **<h3>2. FIRST INPUT DELAY (FID)</h3>**

Этот показатель измеряет время, через которое пользователь может взаимодействовать со страницей. Например: щелчком по ссылке или кнопке и моментом, когда браузер начнет обрабатывать этот щелчок. Он нужен для измерения интерактивности страницы.

Представьте: пользователь видит весь контент загруженным, но при этом страница длительное время не отвечает на его действия. Пользователь остается разочарованным и покидает сайт(скорее всего навсегда), а вы упускаете потенциального клиента.

Важным моментом является то, что этот показатель нельзя смоделировать, поскольку он действительно зависит от того, когда пользователь фактически нажимает на интерактивный элемент или иным образом взаимодействует со страницей, а затем от того, сколько времени потребуется для выполнения действия.

Основной причиной таких проблем обычно являются:

* блокирующие отрисовку страницы CSS-стили и Javascript-скрипты,
* длительные задачи в основной потоке(зачастую Javascript-скрипты).

При изучении данного пункта необходимо также обратить внимание на критерии “Total Blocking Time” (TBT) и “Time to Interactive” (TTI), которые тесно связаны с FID.

#### **<h3>3. CUMULATIVE LAYOUT SHIFT (CLS)</h3>**

Очень интересный показатель, который предназначен для измерения визуальной стабильности страницы и отслеживает - насколько контент страницы сдвигается (“прыгает”), когда новые элементы отрисовываются на своих местах страницы. Каждый из нас хоть раз, но встречался с ситуацией, когда при прочтении статьи текст сдвигался по мере загрузки изображений, рекламы и другого контента.

Это довольно неприятно и раздражает пользователей. Еще хуже, когда кнопка, которую вы собирались нажать, внезапно перемещается, и вместо этого вы нажимаете другую кнопку или еще хуже ссылку, которая отправит вас на другую страницу! CLS пытается учесть эти сдвиги в макете.

Причинами сдвигов макета обычно являются:

* отсутствие размеров изображений, видео и элементов iframe,
* веб-шрифты,
* инициализация плагина или библиотеки,
* вставка рекламных объявлений, cookie-уведомления, подписки на рассылку новостей,
* динамически вставляемые после сетевого запроса DOM элементы.

### **<h2>Синтетические инструменты vs. реальные показатели</h2>**

Один из ключевых моментов для понимания Core Web Vitals заключается в том, что они основаны на реальных показателях пользователей (RUM). Google использует анонимные данные пользователей Chrome и делает их доступными в отчете о пользовательском опыте Chrome (CrUX). Эти данные они используют для измерения трех метрик, рассмотренных выше и на их основе формирует ранжирование. Данные CrUX доступны в ряде инструментов, в том числе в Google Search Console вашего сайта.

Тот факт, что используются данные RUM, является важным отличием, потому что некоторые из этих показателей доступны в синтетических инструментах веб-производительности, таких как [Lighthouse](https://developers.google.com/web/tools/lighthouse), которые в прошлом были основным продуктом мониторинга веб-производительности. Эти инструменты запускают загрузку страниц в смоделированных сетях и устройствах, а затем сообщают вам показатели тестового запуска.

Таким образом, если вы запускаете Lighthouse на мощном компьютере и получаете отличный результат по всем критериям, рассмотренным выше - это с большой долей вероятности не отразит то, что пользователи испытывают в реальном мире и что Google будет использовать для измерения опыта пользователей вашего сайта.

LCP сильно зависит от состояния сети и вычислительной мощности используемых устройств и высока вероятность, что [пользователи вашего сайта будут использовать устройства с меньшей мощностью, чем вы думаете](https://infrequently.org/2021/03/the-performance-inequality-gap/). Однако, стоит заметить, что часто реальный показатель загрузки сайта на мобильном устройстве лучше, чем его тестирование с помощью того же Lighthouse. При этом, в данный момент [в Github репозитории Lighthouse ведется обсуждение по изменению(ужесточению) мобильных настроек](https://github.com/GoogleChrome/lighthouse/issues/12235).

Точно так же FID часто зависит от скорости работы процессора и того, как устройство обрабатывает весь контент, который мы ему отправляем - будь то изображения для обработки, элементы для макета на странице или, конечно же, JavaScript.

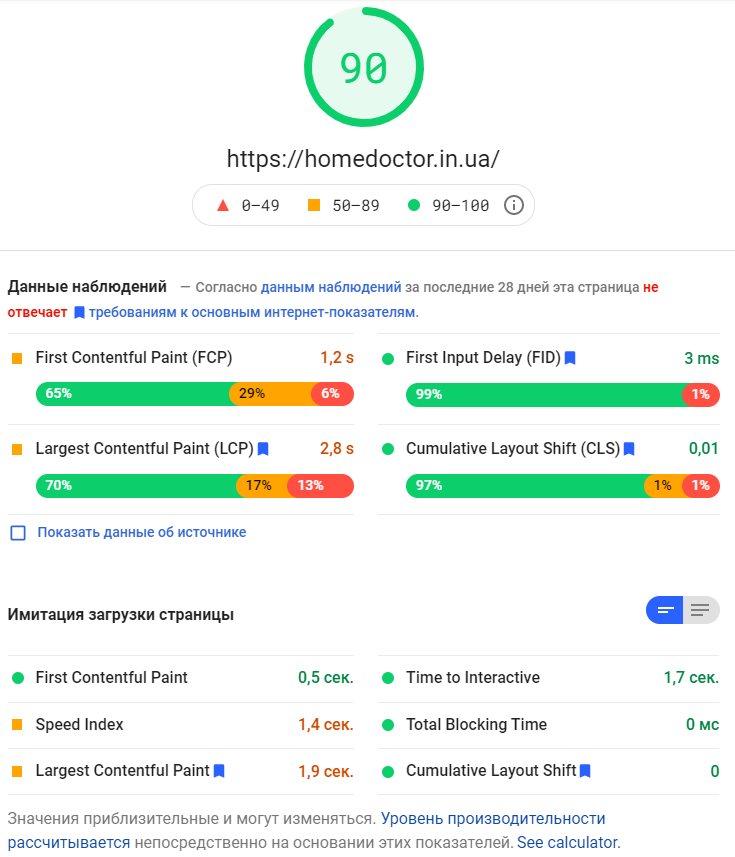
Теоретически, CLS легче измерить с помощью инструментов тестирования, поскольку он менее подвержен изменениям в сети и не так зависит от оборудовании. Однако, важно отметить несколько важных соображений, которые изначально могут быть неочевидными:

* CLS измеряется на протяжении всего времени использования страницы, в отличии от синтетических инструментов Google, о которых мы поговорим позже. Это вызывает большую путаницу, когда смоделированные данные загрузки страницы имеют очень низкий CLS, но оценка CLS намного выше из-за CLS, вызванного прокруткой или другими изменениями после начальной загрузки, которую обычно измеряют инструменты тестирования.
* CLS может зависеть от размера окна браузера - обычно такие инструменты, как PageSpeed ​​Insights, измеряют производительность для мобильных и десктопных устройств. Однако, мобильные устройства имеют разный размер экрана, а десктопные часто намного больших размеров, чем установлено в инструментах тестирования.
* Пользователи видят на веб-страницах разные вещи. Cookie-баннеры, настраиваемый контент типа рекламных акций и блокировщиков рекламы. Все они влияют на то, какой контент отображается, и на то, какой CLS пользователи могут испытать на своих устройствах.
* [CLS все еще находится в процессе развития](https://web.dev/evolving-cls/), и команда Chrome занимается исправлением «невидимых» сдвигов и прочих критериев, которые не должны учитываться в CLS. Это означает, что вы можете видеть разные значения CLS в зависимости от того, какая версия Chrome запущена.

**<h2>Как ранее формировались показатели веб-производительности</h2>**

Еще одна путаница заключается в том, что эти метрики являются новыми и отличаются от тех, которые мы традиционно использовали в прошлом для измерения веб-производительности.

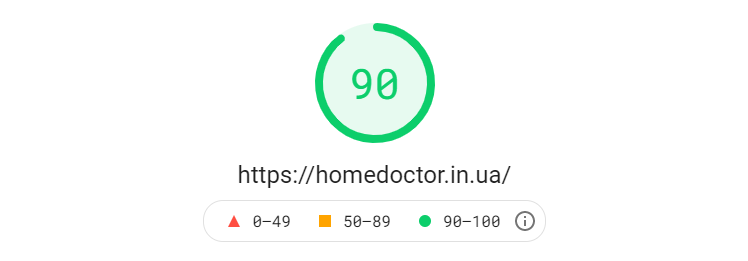
Самым популярным бесплатным инструментом онлайн-аудита является [PageSpeed ​​Insights](https://developers.google.com/speed/pagespeed/insights/). Просто введите URL-адрес, по которому вы хотите провести аудит, и нажмите «Анализировать», и через несколько секунд вам будут представлены две вкладки (одна для мобильных устройств и одна для десктопных), которые содержат большой объем информации:

****

***Значение атрибута alt для изображения: Показатели производительности сайта PageSpeed Insights***

*Показатели производительности сайта на PageSpeed Insights*

Наверху находится рейтинг производительности Lighthouse.



***Значение атрибута alt для изображения: Общая оценка производительности сайта PageSpeed Insights***

*Общая оценка производительности сайта на PageSpeed Insights*

Lighthouse хорошо известен в сообществах веб-производительности и часто упоминается как ключевой показатель, к которому нужно стремиться и который суммирует показатели многих критериев в понятный всем формат - числовое значение из 100. Частично, итоговый результат совпадает с показателями Core Web Vitals, но в отличии от Core Web Vitals, представляет более широкий набор показателей.

В настоящий момент [оценку производительности Lighthouse составляют шесть показателей](https://web.dev/performance-scoring/), включая Core Web Vitals, а также некоторые другие:

* First Contentful Paint (FCP)
* Speed Index (SI)
* Largest Contentful Paint (LCP)
* Time to Interactive (TTI)
* Total Blocking Time (TBT)
* Cumulative Layout Shift (CLS)

Чтобы усложнить задачу, [каждый из этих шести показателей оценивается по-разному](https://googlechrome.github.io/lighthouse/scorecalc/), и CLS, несмотря на то, что является одним из основных, в настоящее время составляет всего 5% от оценки производительности Lighthouse. Все это означает, что вы можете получить очень высокую оценку производительности Lighthouse, окрашенную в зеленый цвет, и думать, что ваш веб-сайт в порядке, но при всем при этом не пройти порог Core Web Vitals. Поэтому, возможно, сейчас вам потребуется направить свои усилия именно на 3 основных показателя.

Перейдем к блоку “Данные наблюдений” и получим еще одну путаницу:

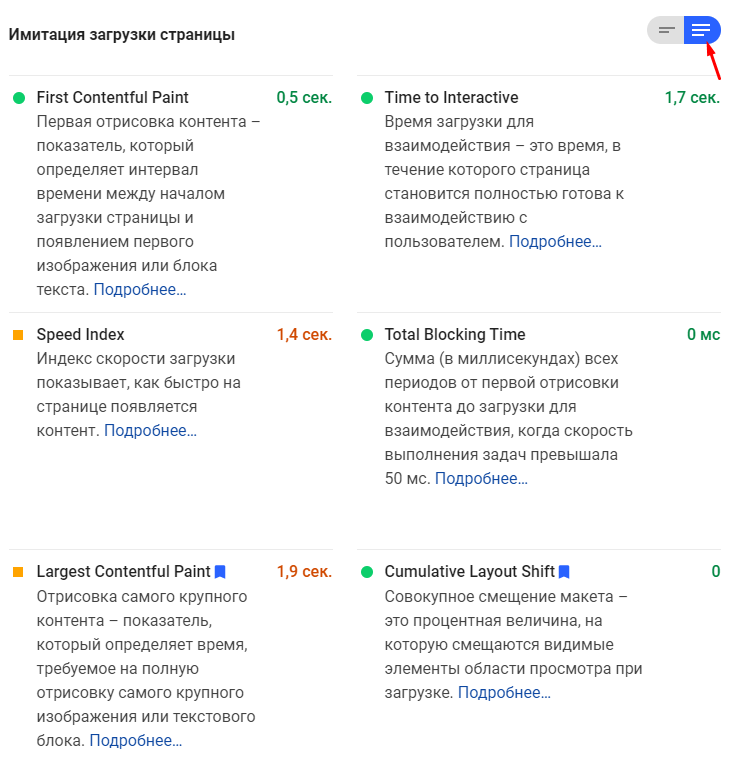


***Значение атрибута alt для изображения: Данные наблюдений сайта PageSpeed Insights***

*“Данные наблюдений” сайта на PageSpeed Insights*

First Contentful Paint отображается в данном блоке вместе с другими тремя Core Web Vitals, несмотря на то, что не является их частью. FCP всегда являлся одним из ключевых критериев и, вероятно, таким и останется. К данному разделу мы еще вернемся.

После идет блок “Имитация загрузки страницы”, где мы видим уже шесть показателей, которые составляют оценку производительности. Если вы нажмете на переключатель в правом верхнем углу данного блока, то получите немного больше информации об этих показателях:



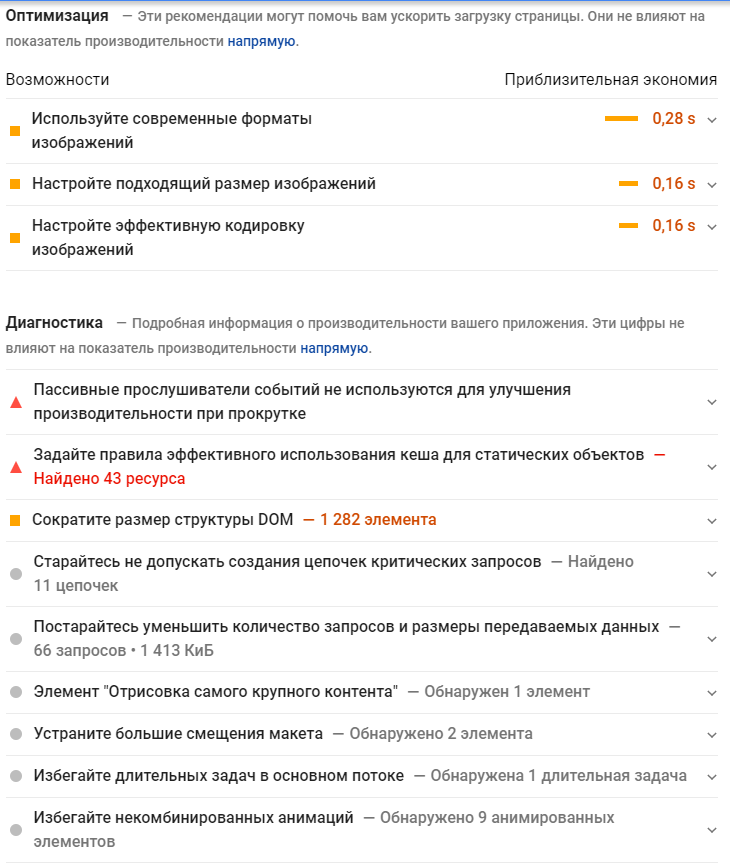
***Значение атрибута alt для изображения: Имитация загрузки страницы PageSpeed Insights***

*“Имитация загрузки страницы” на PageSpeed Insights*

Как видите, сюда включены LCP и CLS, и, поскольку они являются частью Core Web Vitals, имеют синюю этикетку, выделяясь как особо важные. PageSpeed ​​Insights также предоставляет [полезную ссылку на калькулятор](https://googlechrome.github.io/lighthouse/scorecalc/#FCP=540&SI=1396&LCP=1878&TTI=1745&TBT=1&CLS=0&FCI=1682&FMP=540), чтобы увидеть влияние этих оценок на общий балл и позволяет настраивать их, чтобы увидеть, как улучшение каждого показателя повлияет на результат.

Lighthouse также выполняет около 50 других проверок дополнительных возможностей и диагностики. По словам Google, они не влияют ни на оценку, ни на Core Web Vitals, но могут использоваться веб-разработчиками для повышения производительности сайта. От себя хотел бы добавить, что часто исправление данных рекомендаций на общую оценку все же влияют.

Они отображаются в PageSpeed ​​Insights ниже:



***Значение атрибута alt для изображения: Дополнительные рекомендации PageSpeed Insights***

*Дополнительные рекомендации PageSpeed Insights*

Несмотря на то, что Google трактует их как предложениях по повышению производительности, а не как конкретные проблемы, которые необходимо решить, я бы рекомендовал обращать на них внимание.

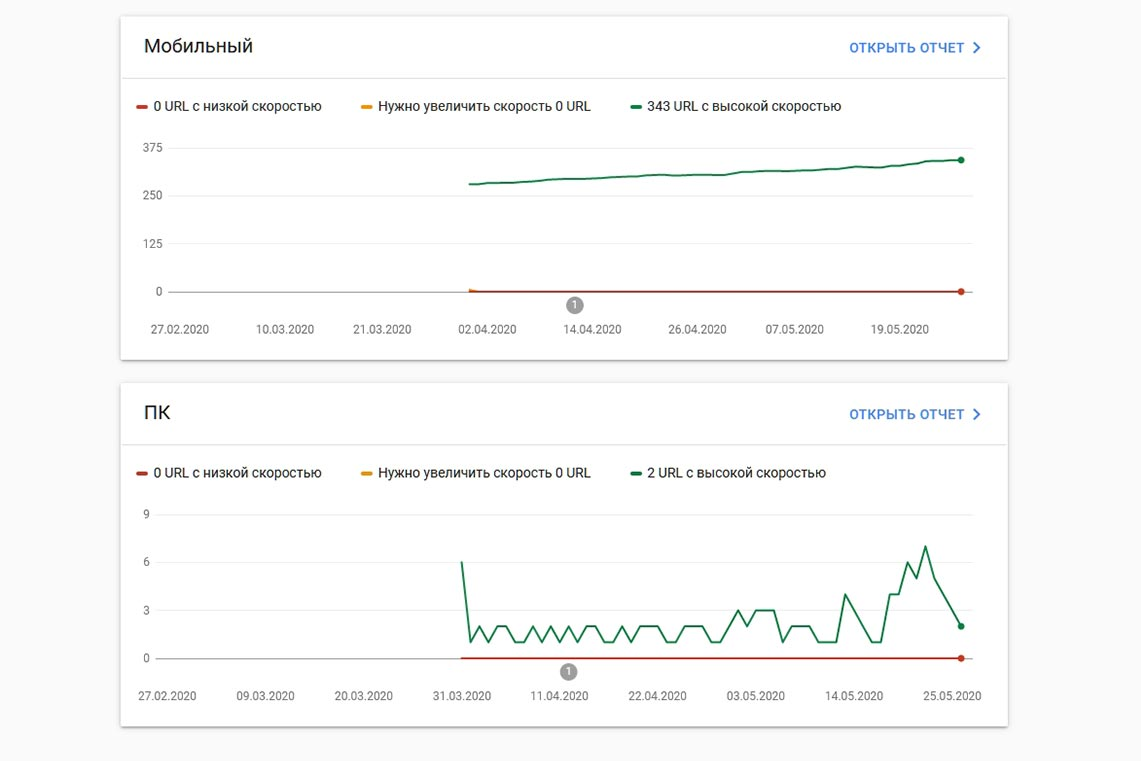
Диагностика покажет вам элемент LCP и сдвиги, которые повлияли на вашу оценку CLS, что является очень полезной информацией при оптимизации для Core Web Vitals.

Подводя небольшой итог, отмечу, что хотя в прошлом сторонники веб-производительности могли в значительной степени сосредоточиться на оценках и аудитах Lighthouse, в ближайшее время внимание будет направлено на три основные показателя Core Web Vital.

**<h2>Просмотр Core Web Vitals вашего сайта</h2>**

Самый простой способ быстро взглянуть на Core Web Vitals для отдельного URL-адреса - ввести URL-адрес в PageSpeed ​​Insights, как описывалось ранее. Однако, чтобы увидеть, как Google видит Core Web Vitals для всего вашего сайта, необходимо получить доступ к [Google Search Console](https://search.google.com/search-console/welcome). Для этого вам понадобится учетная запись Google, а затем [подтвердить свое право собственности на сайт](https://support.google.com/webmasters/answer/9008080?hl=ru).

Отчет Core Web Vitals в Google Search Console дает вам сводную информацию о том, насколько ваш сайт соответствовал критериям Core Web Vitals за последние 90 дней:



***Значение атрибута alt для изображения: Отчет Core Web Vitals в GSC***

*Отчет Core Web Vitals в Google Search Console*

В идеале, для того, чтобы пройти критерии Core Web Vitals, все ваши страницы должны быть отмечены зеленым цветом. Несмотря на то, что оранжевая линия является хорошим индикатором того, что вы близки к удаче, на самом деле, для получения всех преимуществ учитывается только зеленая.

Изначально Google [применяет рейтинг Core Web Vitals только к мобильным устройствам](https://support.google.com/webmasters/thread/86521401/core-web-vitals-page-experience-faqs?hl=en&msgid=86521401), но, на самом деле, реализация подобных алгоритмов для десктопных устройств - вопрос времени, поэтому не игнорируйте их оптимизацию.

Щелкнув на один из отчетов, вы получите более подробную информацию о том, какие из основных показателей не проходят тест, а затем и рассмотреть проблемы конкретных URL. Нажмите на URL-адрес, чтобы запустить PageSpeed ​​Insights для быстрого аудита производительности определенного URL-адреса (включая отображение данных о Core Web Vitals для этой страницы). После, устраняете выявленные проблемы, повторно запускаете PageSpeed ​​Insights для того, чтобы убедиться, что синтетические показатели верны, а затем переходите к следующей странице.

Однако, после кропотливой работы, вы, возможно, будете разочарованы тем, что отчет Core Web Vitals не изменился, а PageSpeed ​​Insights упорно продолжает показывать URL и сайт в целом как неисправные. Почему?

**<h2>Отчет о пользовательском опыте Chrome (CrUX)</h2>**

Причина, по которой Web Vitals обновляется довольно медленно заключается в том, что данные формируются на основе последних 21-28 дней [пользовательского опыта Chrome (CrUX)](https://developers.google.com/web/tools/chrome-user-experience-report) - именно столько времени достаточно для устранения разного рода отклонений и отображения более точной производительности вашего сайта.

Как уже отмечалось ранее, показатели производительности очень чувствительны к сети и устройствам, поэтому для сглаживания отклонений требуется некоторое время, чтобы получить реальную картину того, как ваш сайт работает для большинства пользователей. При этом, обратная сторона заключается в том, что данные очень медленно обновляются, создавая проблему обратной связи между исправлениями проблем и отображением изменений.

**<h3>Дополнительная информация о данных CrUX</h3>**

CrUX - это инструмент Chrome. Пользователи iOS устройств и других браузеров (десктопная версия Safari, Firefox, Opera, Edge и т.д.) никаким образом не влияют на данные CrUX, соответственно и на показатели Core Web Vitals вашего сайта.

И все же, несмотря на то, что подавляющее большинство пользователей используют именно Chrome, в большинстве случаев проблемы с производительностью, которые отмечают инструменты Google, соответствует показателям других браузеров.

Используемая версия Chrome также будет иметь влияние на показатели, поскольку все критерии все еще находятся в стадии развития и изменений([в частности, CLS](https://chromium.googlesource.com/chromium/src/+/main/docs/speed/metrics_changelog/cls.md)).

CrUX включает в себя все страницы, в том числе те, которые обычно не отображаются в поиске Google(например с мета-тегом <meta name="robots" content="noindex">). Теперь такие страницы, скорее всего, не будут включены в результаты поиска Google, соответственно не будут влиять на ранжирование, несмотря на то, что в CrUX вы все еще будете их наблюдать.

CrUX основан на просмотрах страниц. Это означает, что ваши самые популярные страницы будут иметь большое влияние на показатели. Некоторые страницы будут появляться или исчезать из CrUX в тот момент, когда будут достигать пороговых значений.

Теперь мы знаем, каким образом собираются и анализируются данные, но актуальным становится вопрос - как ускорить обновление наших изменений во время разработки. Ждать 21-28 дней, чтобы увидеть влияние данных CrUX - только для того, чтобы понять, что ваших исправлений было недостаточно?

**<h3>Получение мгновенных результатов об изменениях в CRuX</h3>**

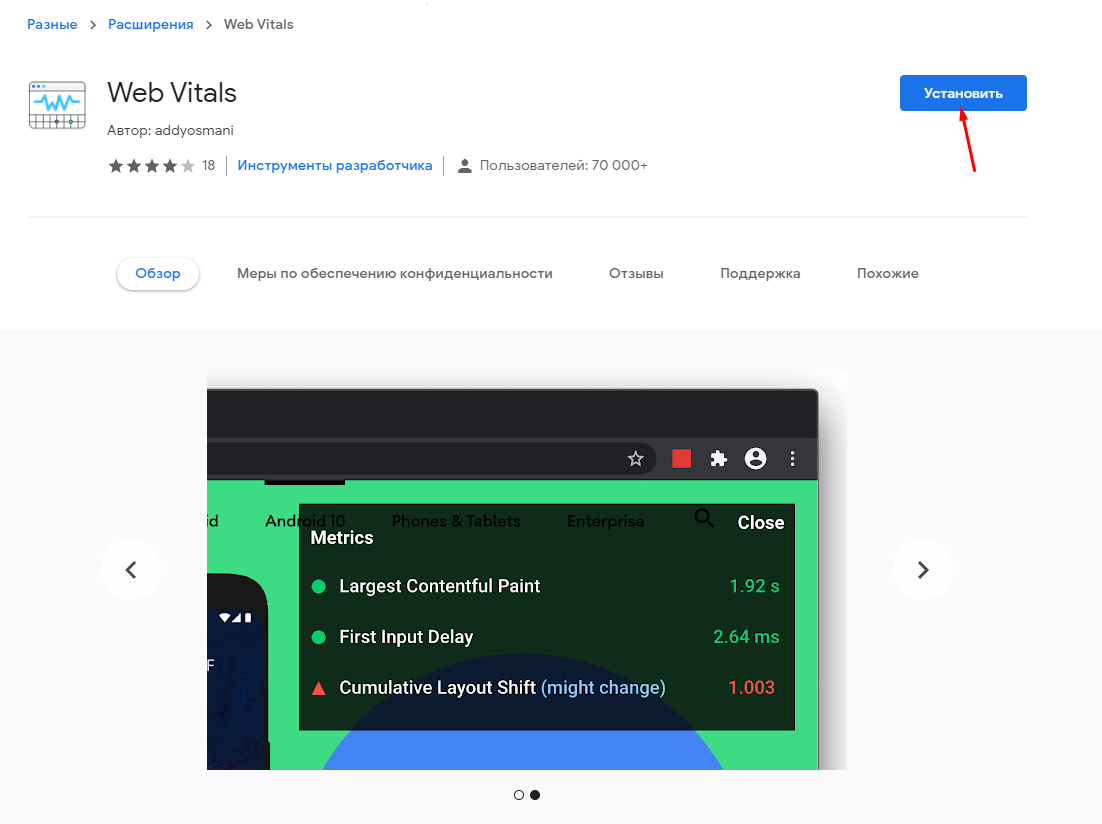
На рынке существует [несколько отличных коммерческих продуктов RUM](https://sematext.com/blog/real-user-monitoring-tools/), которые дают возможность запрашивать данные гораздо глубже и с гораздо большей частотой, чем CrUX.

Однако, в данной статье мы рассмотрим 2 других инструмента - расширение для браузера Chrome [Web Vitals Chrome Extension](https://github.com/GoogleChrome/web-vitals-extension)и Javascript-библиотека для фронтенд-разработчиков [web-vitals](https://github.com/GoogleChrome/web-vitals).

**<h4>Расширение Web Vitals Chrome Extension</h4>**

Это довольно простой для понимания инструмент. Давайте рассмотрим процесс его установки, а затем получим результаты одного из примеров.

1. Переходим на [страницу расширения в веб-магазине Google](https://chrome.google.com/webstore/detail/web-vitals/ahfhijdlegdabablpippeagghigmibma).
2. Устанавливаем расширение, нажав на кнопку “Установить” в правом верхнем углу.

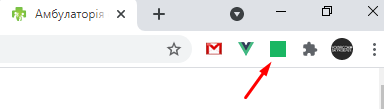


***Значение атрибута alt для изображения: Расширение Web Vitals Chrome Extension***

*Расширение Web Vitals Chrome Extension*

Спустя пару секунд расширение будет установлено в вашем браузере Chrome.

Отображается расширение в виде квадрата,



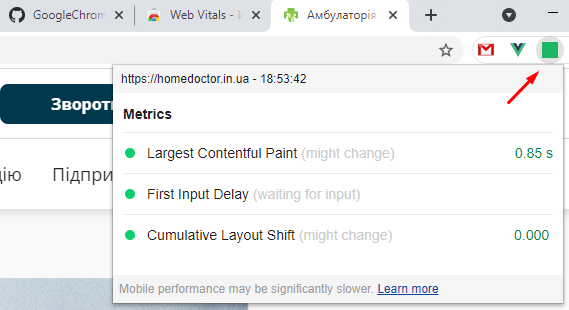
***Значение атрибута alt для изображения: Отображение Web Vitals Chrome Extension***

*Расширение Web Vitals Chrome Extension*

который может быть 3х цветов:

1. Зеленый - хорошие показатели Core Web Vitals.
2. Красный - плохие показатели Core Web Vitals.
3. Серый - показатели недоступны.

Нажав на иконку расширения, мы можем получить более точные данные по конкретным метрикам:



***Значение атрибута alt для изображения: Данные по метрикам в Web Vitals Chrome Extension***

*Расширение Web Vitals Chrome Extension*

**<h4>Javascript-библиотека web-vitals</h4>**

Это минималистичная (1кб) библиотека, которая создана для фронтенд-мониторинга. По моему мнению, в данный момент этот инструмент является самым эффективным, потому что позволяет отследить показатели за короткие промежутки времени с очень высокой точностью.

Полное описание по установке и использованию, а также API данной библиотеки можно посмотреть [тут](https://github.com/GoogleChrome/web-vitals).

Я же опишу ее главные функции. Итак, приступим:

**<h5>Установка и загрузка библиотеки с помощью npm</h5>**

**npm install web-vitals**

Существует две разные версии библиотеки web-vitals («стандартная» версия и версия «base + polyfill»), и то, как вы загружаете библиотеку, зависит от того, какую версию вы хотите использовать.

Чтобы узнать о различиях версий, посмотрите [какая сборка подходит вам](https://github.com/GoogleChrome/web-vitals#which-bundle-is-right-for-you).

1. **“Стандартная” версия**

Для загрузки импортируйте модули из пакета web-vitals в код своего приложения:

**import {getLCP, getFID, getCLS} from 'web-vitals';**

**getCLS(console.log);**

**getFID(console.log);**

**getLCP(console.log);**

1. **"Базовая версия с полифиллом"**

Загрузка версии "base + polyfill" - это двухэтапный процесс:

Во-первых, в коде вашего приложения импортируйте «базовую» сборку, вместо «стандартной». Для этого достаточно изменить путь на ‘web-vitals/base’:

**import {getLCP, getFID, getCLS} from 'web-vitals/base';**

Затем вставьте код из dist/polyfill.js, который находится в папке установленной библиотеки в <head> своих страниц. Этот шаг важен, поскольку "базовая" сборка не сработает, если код полифилла не будет добавлен.

**<!DOCTYPE html>**

**<html>**

**<head>**

**<script>**

**// Код из `dist/polyfill.js`**

**</script>**

**</head>**

**<body>**

**...**

**</body>**

**</html>**

**<h5>Установка и загрузка библиотеки с помощью CDN</h5>**

Важно отметить, что авторы рекомендуют устанавливать библиотеку через npm и интегрировать сборку в ваш код. Однако, если вы не используете npm, подключение web-vitals с помощью CDN все еще допускается.

Примеры, которые будут показаны ниже, отображают загрузку web-vitals вне зависимости, нацелены ли вы на браузеры, построенные на кодовой базе Chromium(используя “стандартную версию”) или не на ней (используя "Базовую версию с полифиллом"):

**Загрузка “стандартной версии” (используя <script type=”module”></script>):**

**<script type="module">**

**import {getCLS, getFID, getLCP} from 'https://unpkg.com/web-vitals?module';**

**getCLS(console.log);**

**getFID(console.log);**

**getLCP(console.log);**

**</script>**

**Загрузка “стандартной версии” (используя тег <script></script>):**

**<script>**

**(function() {**

**var script = document.createElement('script');**

**script.src = 'https://unpkg.com/web-vitals';**

**script.onload = function() {**

**// Когда вы используете `web-vitals` используя классический вид тега script,**

**// все публичные методы будут доступны в глобальном объекте `webVitals`**

**webVitals.getCLS(console.log);**

**webVitals.getFID(console.log);**

**webVitals.getLCP(console.log);**

**}**

**document.head.appendChild(script);**

**}())**

**</script>**

**Загрузка “базовой версии с полифиллом” (используя тег <script></script>):**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<script>**

**// Вставьте код из `https://unpkg.com/web-vitals/dist/polyfill.js` тут.**

**</script>**

**</head>**

**<body>**

**...**

**<!-- Загрузите UMD версию “базовой сборки”. -->**

**<script>**

**(function() {**

**var script = document.createElement('script');**

**script.src = 'https://unpkg.com/web-vitals';**

**script.onload = function() {**

**// Когда вы используете `web-vitals` используя классический вид тега script,**

**// все публичные методы будут доступны в глобальном объекте `webVitals`**

**webVitals.getCLS(console.log);**

**webVitals.getFID(console.log);**

**webVitals.getLCP(console.log);**

**}**

**document.head.appendChild(script);**

**}())**

**</script>**

**</body>**

**</html>**

**<h5>Использование</h5>**

Каждая метрика Web Vitals представлена ​​в виде отдельной функции, которая принимает коллбэк “onReport”, который будет вызываться каждый раз, когда значение метрики доступно и готово к отправке.

В следующем примере измеряется каждая из метрик Core Web Vitals и как только значение для отчета получено - выводится в консоль.

(Примеры ниже используют "стандартную" сборку, но они также будут работать с версией полифилла).

import {getCLS, getFID, getLCP} from 'web-vitals';

getCLS(console.log);

getFID(console.log);

getLCP(console.log);

Если значения не выводятся сразу в консоль, попробуйте перезагрузить страницу или переключитесь между вкладками.

Кроме того, в некоторых случаях данный коллбэк может никогда не вызваться:

* FID не будет получен, если пользователь не будет взаимодействовать со страницей.
* FCP, FID и LCP не будут получены, если страница загружалась в фоновом режиме.

В некоторых случаях данный коллбэк может быть вызван более одного раза:

* CLS будет получен каждый раз когда [“состояние видимости” страницы меняется на “скрытое”](https://developers.google.com/web/updates/2018/07/page-lifecycle-api#advice-hidden)
* FCP, FID, и LCP будут получены после восстановления страницы из кеша

**Получение значения при каждом изменении**

В большинстве случаев, коллбэк “onReport” вам понадобится лишь тогда, когда значение метрики будет доступно. Однако, у вас есть возможность получать значение при каждом изменении(например, при каждом сдвиге макета), установив необязательный второй аргумент

reportAllChanges в значение true.

Данный прием может быть полезен для отладки, но в большинстве случаев reportAllChanges

вам не понадобится.

import {getCLS} from 'web-vitals';

// Выводит в консоль значение CLS при каждом изменении.

getCLS(console.log, true);

**Получение только дельты изменений**

Некоторые инструменты аналитики позволяют обновлять значение метрики даже после того, как вы уже отправили ее сервер (перезаписывая ранее отправленное значение с тем же id).

Однако, существуют инструменты, которые не допускают этого, поэтому вместо передачи нового значения вам необходимо сообщить только дельту (разницу между текущим значением и последним сообщенным значением). Затем вы можете вычислить общее значение, суммируя все дельты показателей, отправленные с одним и тем же ID.

В следующем примере показано, как использовать свойства id и delta:

import {getCLS, getFID, getLCP} from 'web-vitals';

function logDelta({name, id, delta}) {

console.log(`${name} matching ID ${id} changed by ${delta}`);

}

getCLS(logDelta);

getFID(logDelta);

getLCP(logDelta);

**Отправка результатов в конечную точку аналитики**

В следующем примере измеряется каждая из метрик Core Web Vitals и отправляется по гипотетическому адресу аналитики, как только каждая из них будет готова к отправке.

Функция sendToAnalytics() использует метод [navigator.sendBeacon()](https://developer.mozilla.org/ru/docs/Web/API/Navigator/sendBeacon) (если он доступен), в противном случае возвращается к [fetch() API](https://developer.mozilla.org/ru/docs/Web/API/Fetch_API).

import {getCLS, getFID, getLCP} from 'web-vitals';

function sendToAnalytics(metric) {

const body = JSON.stringify({[metric.name]: metric.value});

// Используется `navigator.sendBeacon()` (если доступен), в противном случае `fetch()`.

(navigator.sendBeacon && navigator.sendBeacon('/analytics', body)) ||

fetch('/analytics', {body, method: 'POST', keepalive: true});

}

getCLS(sendToAnalytics);

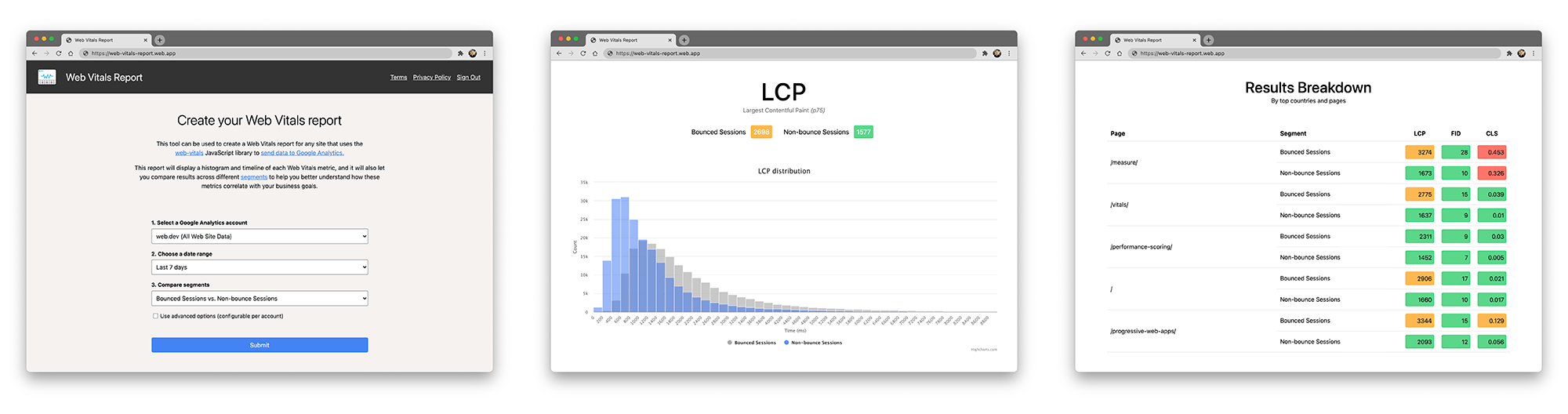
getFID(sendToAnalytics);

getLCP(sendToAnalytics);

**Отправка результатов в Google Analytics**

В том случае, если вы установите уникальное значение параметра (в данном случае id метрики, как будет показано в примерах ниже) для каждого экземпляра метрики, которую вы отправляете в Google Analytics, вы можете создать отчет, используя [Google Analytics Reporting API](https://developers.google.com/analytics/devguides/reporting) и любую библиотеку визуализации данных на ваш выбор.

Например, отчет [Web Vitals Report](https://github.com/GoogleChromeLabs/web-vitals-report) - бесплатный инструмент с открытым исходным кодом, который вы можете использовать для создания визуализаций данных Web Vitals, которые вы отправили в Google Analytics.



***Значение атрибута alt для изображения: Web Vitals Report***

*Web Vitals Report*

Чтобы использовать Web Vitals Report (или создавать свои собственные пользовательские отчеты с помощью API), вам необходимо отправить свои данные в Google Analytics, следуя одному из примеров, описанных ниже:

**Использование** **analytics.js**

import {getCLS, getFID, getLCP} from 'web-vitals';

function sendToGoogleAnalytics({name, delta, id}) {

// Предполагает, что глобальная функция `ga()` существует, подробнее тут:

// https://developers.google.com/analytics/devguides/collection/analyticsjs

ga('send', 'event', {

eventCategory: 'Web Vitals',

eventAction: name,

// Параметр`id` будет уникальным для текущей загрузки страницы.

// При отправке нескольких значений с одной и той же страницы(например CLS),

// Google Analytics может вычислить результат путем группировки параметров ID

//(важно: требуется ключ `eventLabel`, который будет специальным параметром)

eventLabel: id,

// Показатели Google Analytics должны быть целыми числами.

// Для CLS значение сначала умножается на 1000 для лучшей точности

// (важно: при необходимости увеличьте множитель для еще большей точности).

eventValue: Math.round(name === 'CLS' ? delta \* 1000 : delta),

// Используйте ключ nonInteraction, чтобы не повлиять на “показатель отказов”.

nonInteraction: true,

// Используйте `sendBeacon()` если ваш браузер поддерживает данный метод.

transport: 'beacon'

});

}

getCLS(sendToGoogleAnalytics);

getFID(sendToGoogleAnalytics);

getLCP(sendToGoogleAnalytics);

**Использование** **gtag.js (Universal Analytics)**

import {getCLS, getFID, getLCP} from 'web-vitals';

function sendToGoogleAnalytics({name, delta, id}) {

// Предполагает, что глобальная функция `gtag()` существует, подробнее тут:

// https://developers.google.com/analytics/devguides/collection/gtagjs

gtag('event', name, {

event\_category: 'Web Vitals',

// Параметр`id` будет уникальным для текущей загрузки страницы.

// При отправке нескольких значений с одной и той же страницы(например CLS),

// Google Analytics может вычислить результат путем группировки параметров ID

//(важно: требуется ключ `eventLabel`, который будет специальным параметром)

event\_label: id,

// Показатели Google Analytics должны быть целыми числами.

// Для CLS значение сначала умножается на 1000 для лучшей точности

// (важно: при необходимости увеличьте множитель для еще большей точности).

value: Math.round(name === 'CLS' ? delta \* 1000 : delta),

// Используйте ключ non\_interaction, чтобы не повлиять на “показатель отказов”.

non\_interaction: true

});

}

getCLS(sendToGoogleAnalytics);

getFID(sendToGoogleAnalytics);

getLCP(sendToGoogleAnalytics);

**Использование** **gtag.js (Google Analytics 4)**

Google Analytics 4 предоставляет новую модель событий, позволяющую настраивать параметры вместо фиксированных категорий, действий и меток. Он также поддерживает нецелочисленные значения, что упрощает измерение показателей Web Vitals по сравнению с предыдущими версиями.

import {getCLS, getFID, getLCP} from 'web-vitals';

function sendToGoogleAnalytics({name, delta, value, id}) {

// Предполагает, что глобальная функция `gtag()` существует, подробнее тут:

// https://developers.google.com/analytics/devguides/collection/ga4

gtag('event', name, {

// Встроенные параметры:

value: delta, // Используйте `delta`, следовательно значение может суммироваться.

// Кастомные параметры:

metric\_id: id, // Требуется для агрегирования событий.

metric\_value: value, // Optional.

metric\_delta: delta, // Optional.

});

}

getCLS(sendToGoogleAnalytics);

getFID(sendToGoogleAnalytics);

getLCP(sendToGoogleAnalytics);

**Отправка результатов в Google Tag Manager**

Рекомендуемый способ измерения показателей Web Vitals с помощью Google Tag Manager - использование [настраиваемого тега шаблона Core Web Vitals](https://www.simoahava.com/custom-templates/core-web-vitals/).

Полную инструкцию по установке и использованию читайте [тут](https://www.simoahava.com/analytics/track-core-web-vitals-in-ga4-with-google-tag-manager/).

**<h2>Вывод</h2>**

Google имеет большую власть в области веба. Компания не всегда действовала из лучших побуждений, но Core Web Vitals преследует благородную цель. Теперь у нас есть инструменты, которые измеряют показатели производительности страницы на основе реального опыта. Они помогут разработчикам сконцентрироваться на улучшениях, которые принесут наибольшую пользу пользователям клиентов.

Core Web Vitals - это набор ключевых показателей, предназначенных для представления пользовательского опыта просмотра веб-страниц и сейчас нам надо заострить свое внимание

на данных CrUX. Причина, по которой мы так долго зависели от синтетических данных, заключается в том, что данные RUM содержали слишком большие отклонения. Шаги, которые предпринимает CrUX, чтобы уменьшить их количество, действительно помогают обеспечить более стабильное представление, но за счет этого затрудняют просмотр последних изменений.

Javascript-библиотека “web-vitals” в свою очередь помогает разработчикам существенно повысить скорость и эффективность контроля веб-производительности.

Надеюсь, эта статья помогла вам лучше понять 3 важных критерия Core Web Vitals. Ну а в следующей я расскажу, что со своей стороны мы делаем, чтобы достичь наилучших показателей Core Web Vitals.

**Метатеги:**

***Title:***

Core Web Vitals: подробней о новом критерии ранжирования Google

**Description:**

Подробно о новом критерии оценки сайтов — Core Web Vitals в новой статье от OdesSeo. Что из себя представляет CWV, какие именно показатели он учитывает, критерии и способы их оценки.