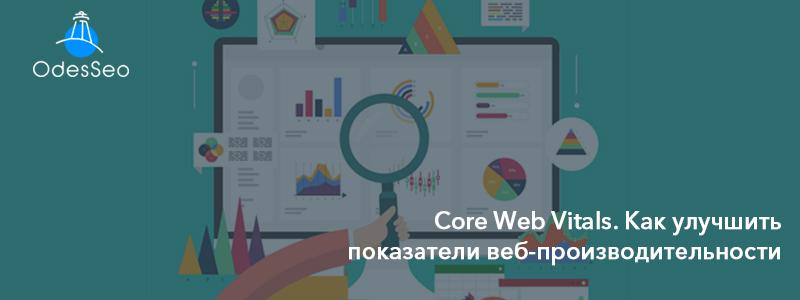
# **<h1>Core Web Vitals. Как улучшить показатели веб-производительности</h1>**

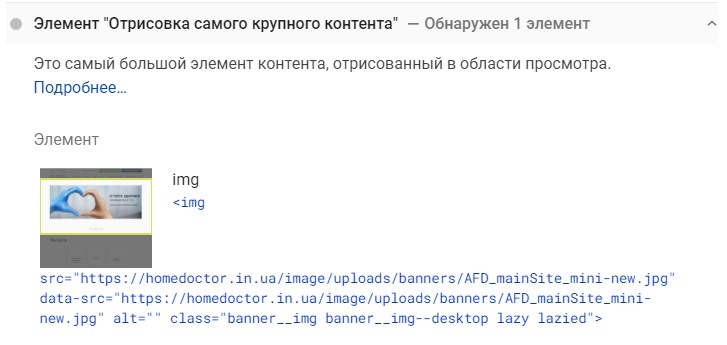


***Значение атрибута alt для изображения: Как улучшить показатели Core Web Vitals***

[В прошлой статье](https://odesseo.com.ua/core-web-vitals-noviy-kriteriy-ranzhirovaniya-google-i-kak-teper-kontrolirovat-web-proizvoditelnost/) мы рассмотрели основные 3 показателя нового критерия ранжирования Google - Core Web Vitals, предназначенных для измерения пользовательского опыта работы с веб-сайтом. Нововведения начнут действовать уже в июне 2021 года. В данной статье я расскажу, что мы делаем для улучшения веб-производительности во время разработки.

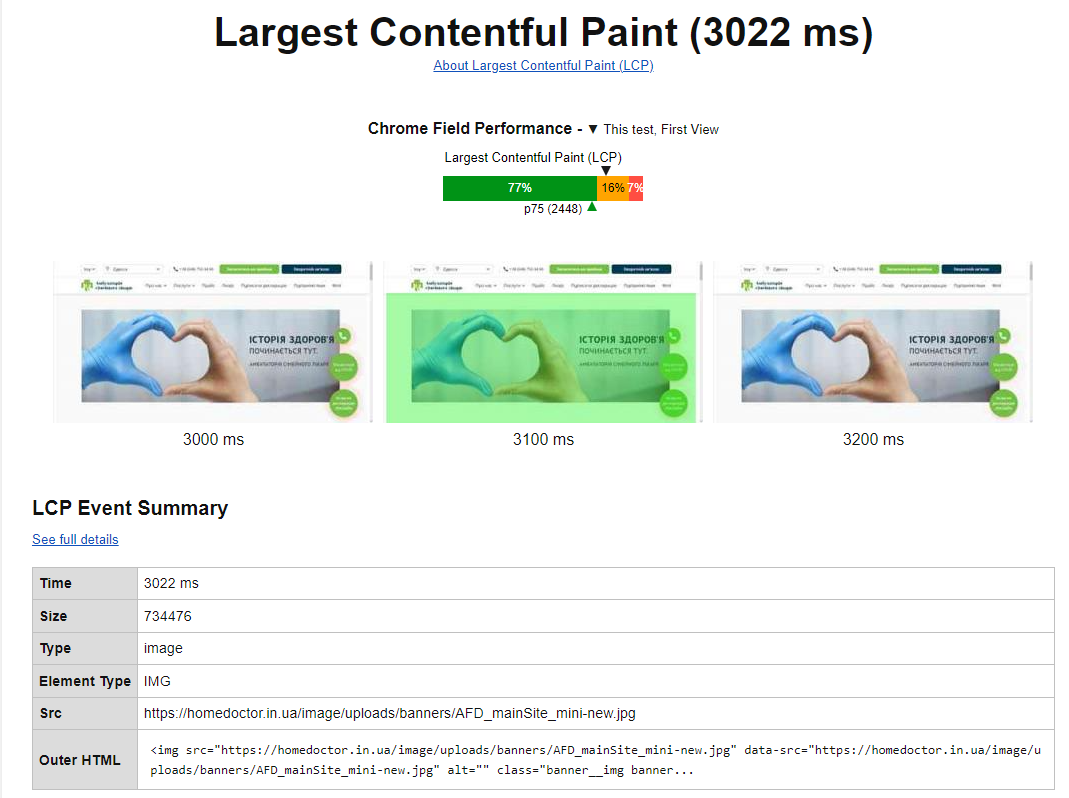
## **<h2>LARGEST CONTENTFUL PAINT (LCP)</h2>**

Данный показатель измеряет, как быстро пользователь получает самый тяжелый элемент контента, отрисованный на странице.



***Значение атрибута alt для изображения: Отрисовка самого крупного контента в Pagespeed Insight***

*“Отрисовка самого крупного контента” в разделе “Диагностика” в* [*Pagespeed Insight*](https://developers.google.com/speed/pagespeed/insights/)



***Значение атрибута alt для изображения: Самый крупный элемент страницы на WebPageTest***

*Информация про самый крупный элемент страницы на* [*WebPageTest*](https://www.webpagetest.org/)

**Советы по улучшению данного показателя**

1. Оптимизация серверных операций

* Проверьте, насколько “быстрым” является ваш хостинг-провайдер
* Оптимизируйте запросы к базе данных
* Используйте CDN для вашего сайта

1. Настройке кэширование статических ресурсов для уменьшения количества дорогостоящих серверных вычислений. Рассмотрите кэширование элементов, которые редко изменяются: изображения, PDF документы, возможно CSS и JS файлы.
2. Настройте подключение к сторонним ресурсам заблаговременно.

Запросы сервера к сторонним ресурсам также могут повлиять на LCP, особенно если они необходимы для отображения важного содержимого на странице. Используйте rel = "preconnect", чтобы сообщить браузеру, что ваша страница намеревается установить соединение как можно скорее.

<link rel="preconnect" href="https://example.com" />

Вы также можете использовать dns-prefetch для более быстрого поиска DNS.

<link rel="dns-prefetch" href="https://example.com" />

Несмотря на то, что оба метода работают по-разному, рассмотрите возможность использования dns-prefetch в качестве альтернативы для браузеров, которые не поддерживают предварительное подключение.

<head>

…

<link rel="preconnect" href="https://example.com" />

<link rel="dns-prefetch" href="https://example.com" />

</head>

*\*Уделите данному пункту особое внимание, так как он важен как для LCP, так и для FID.*

1. Оптимизация блокирующих поток CSS и Javascript.

Скрипты и стили являются ресурсами, блокирующими рендеринг и следовательно, ухудшающие показатель LCP. Отложите любой некритический JavaScript и CSS, чтобы ускорить загрузку основного содержимого вашей веб-страницы.

* Минимизируйте размер CSS и JS. Для облегчения чтения файлы CSS и JS могут содержать такие символы, как интервалы, отступы или комментарии. Все эти символы не нужны браузеру, и минимизация этих файлов гарантирует, что они будут удалены. Используйте сборщики модулей или инструменты сборки, такие как [Webpack](https://webpack.js.org/), [Gulp](https://gulpjs.com/) или [Rollup](https://rollupjs.org/guide/) для минимизации файлов при каждой сборке.
* Отложите загрузку некритического CSS. Используйте предварительную загрузку важных файлов стилей:

<link rel="preload" as="style" href="styles.css">

* [Используйте медиазапросы, чтобы сократить время вывода страницы на экран](https://developers.google.com/web/fundamentals/performance/critical-rendering-path/render-blocking-css):

<link href="styles.css" rel="stylesheet" media="(min-width: 1920px)">

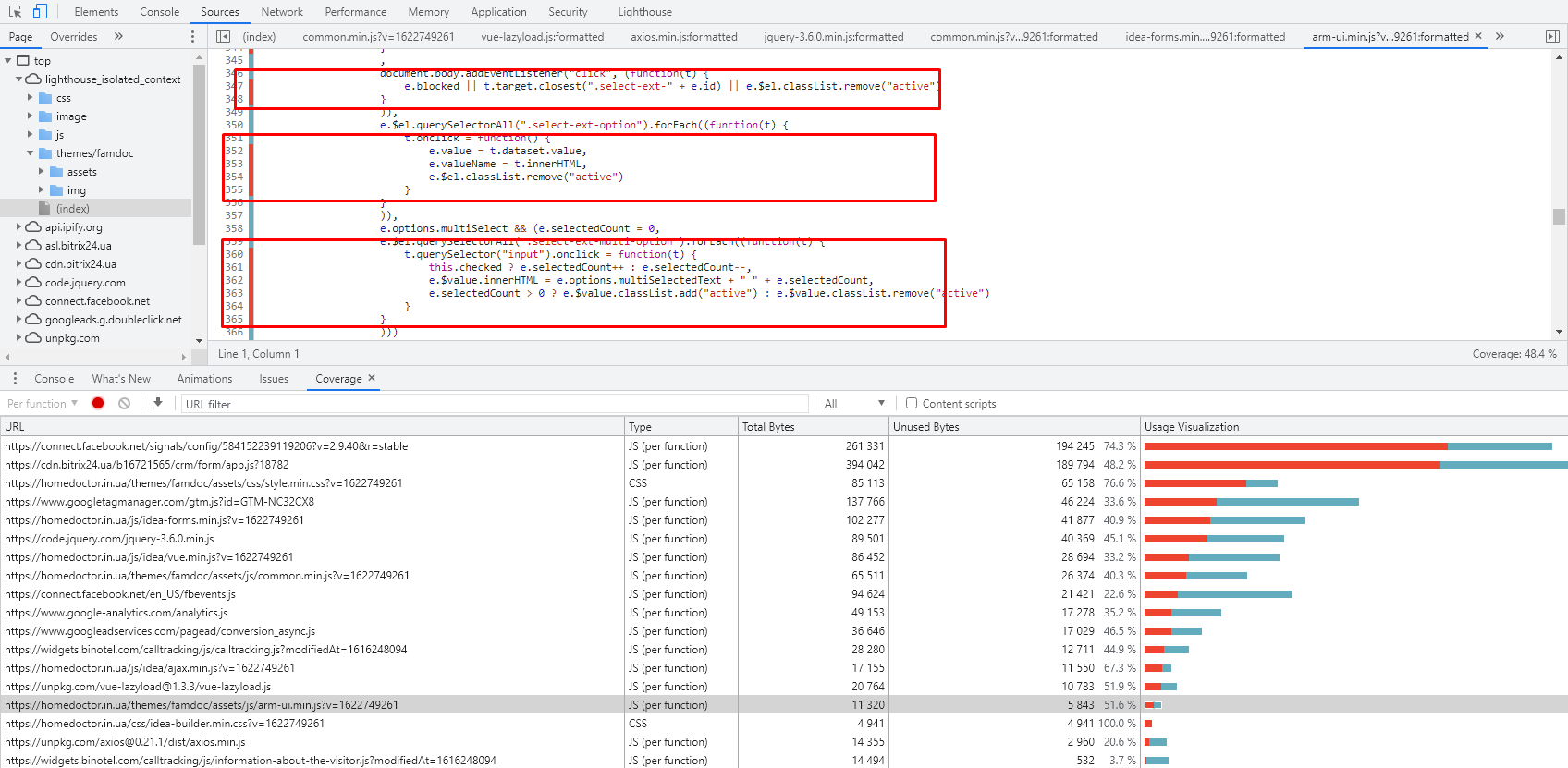
* Заинлайните критический CSS.



***Значение атрибута alt для изображения: Инлайнинг критического CSS***

*“Инлайнинг” критического CSS*

* Удалите неиспользуемый CSS и JS или переместите в отдельные файлы в том случае, если на разных страницах используется различный код. Используйте вкладку [Coverage](https://developers.google.com/web/tools/chrome-devtools/coverage) в Chrome DevTools для отладки.



***Значение атрибута alt для изображения: Вкладка Coverage в ChromeDevTools***

*Вкладка “Coverage” в ChromeDevTools для отладки кода*

*\*Уделите данному пункту особое внимание, так как он важен как для LCP, так и для FID.*

1. Заблаговременная загрузка ключевого изображения.

Зачастую, самым важным и самым большим элементом для измерения LCP являются “hero” изображения в верхней части страницы. Для улучшения данной метрики настройте [предзагрузку](https://web.dev/preload-responsive-images/) данных изображений.

<link rel="preload" as="image" href="image.jpg">

**Важно!** Предзагрузкой нужно пользоваться аккуратно. Пропускная способность соединения на первом этапе загрузки страницы невелика и предзагрузка нескольких картинок может повлиять на загрузку других важных ресурсов. Убедитесь, что ресурсы расположены в правильном порядке, чтобы не привести к ухудшению других метрик, когда другие ресурсы тоже считаются важными (критический CSS, JS, шрифты). Подробнее о цене предзагрузки можно узнать [тут](https://docs.google.com/document/d/1ZEi-XXhpajrnq8oqs5SiW-CXR3jMc20jWIzN5QRy1QA/view).

Как пример, хотелось бы отметить такое частое явление как “карусель” баннеров в верхней части страницы. Настройка предварительной загрузки всех изображений внутри нее может негативно сказаться на LCP, вместо этого, настройте предзагрузку первого изображения, остальные же загружайте “лениво”(о чем будет сказано ниже).

1. Использование отзывчивых изображений с атрибутом srcset и их предварительная загрузка.

Ознакомьтесь с возможностью [предварительной загрузки адаптивных изображений](https://web.dev/preload-responsive-images/#imagesrcset-and-imagesizes). Благодаря атрибутам imagesrcset и imagesizes, добавленных в <link>, вы можете предзагрузить адаптивные картинки:

<link rel="preload" as="image" href="your-image.jpg"

imagesrcset="your-image\_575px.jpg 575w,

your-image\_991px.jpg 991w,

your-image\_1920px.jpg 1920w"

imagesizes="50vw">

В разметке же код должен выглядеть примерно таким образом:

<img src="your-image.jpg"

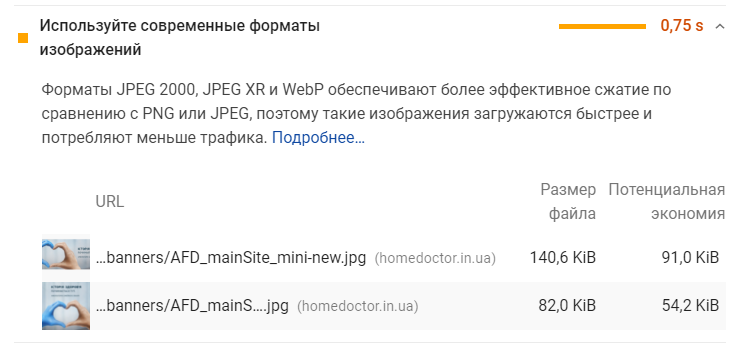
srcset=" your-image\_575px.jpg 575w,

your-image\_991px.jpg 991w,

your-image\_1920px.jpg 1920w"

sizes="50vw" loading="lazy" decoding="async" alt="">

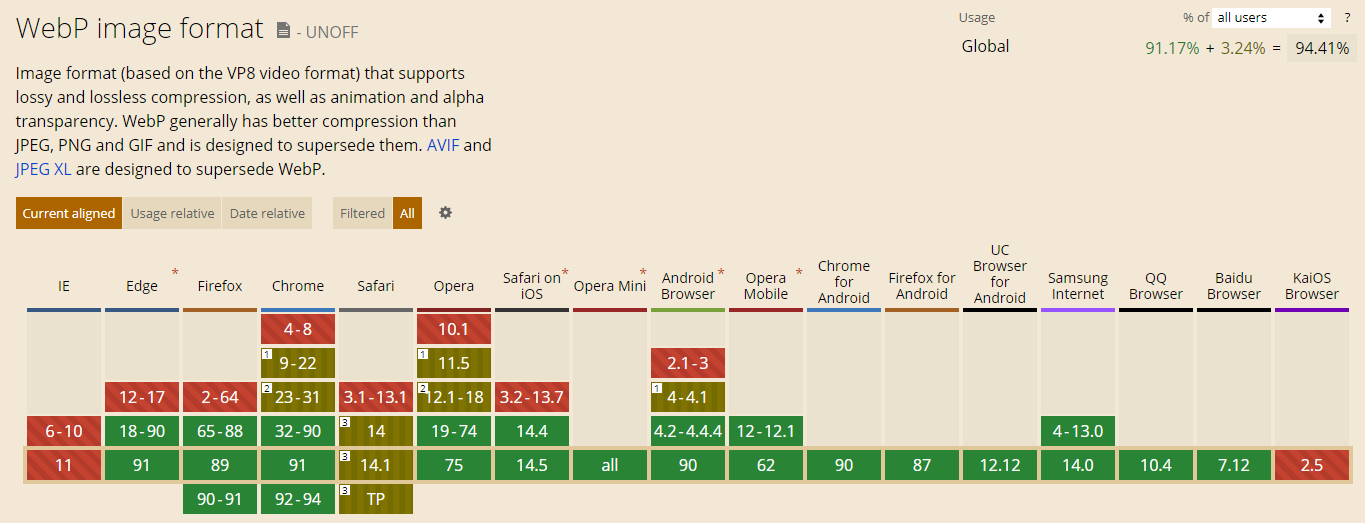
1. Использование эффективных современных форматов изображений.



***Значение атрибута alt для изображения: Рекомендации по оптимизации изображений в Pagespeed Insight***

*Рекомендации по оптимизации изображений в Pagespeed Insight*

Размер изображений относительно нового формата “WebP” обычно меньше аналогов в формате JPEG и PNG на 25–35%. Очевидным будет сказать, что это уменьшает влияние изображений на LCP и положительно сказывается на показателях загрузки. Поддержка данного формата уже довольно высока.



***Значение атрибута alt для изображения: Поддержка формата WebP в различных браузерах***

*Поддержка формата “WebP” в различных браузерах*

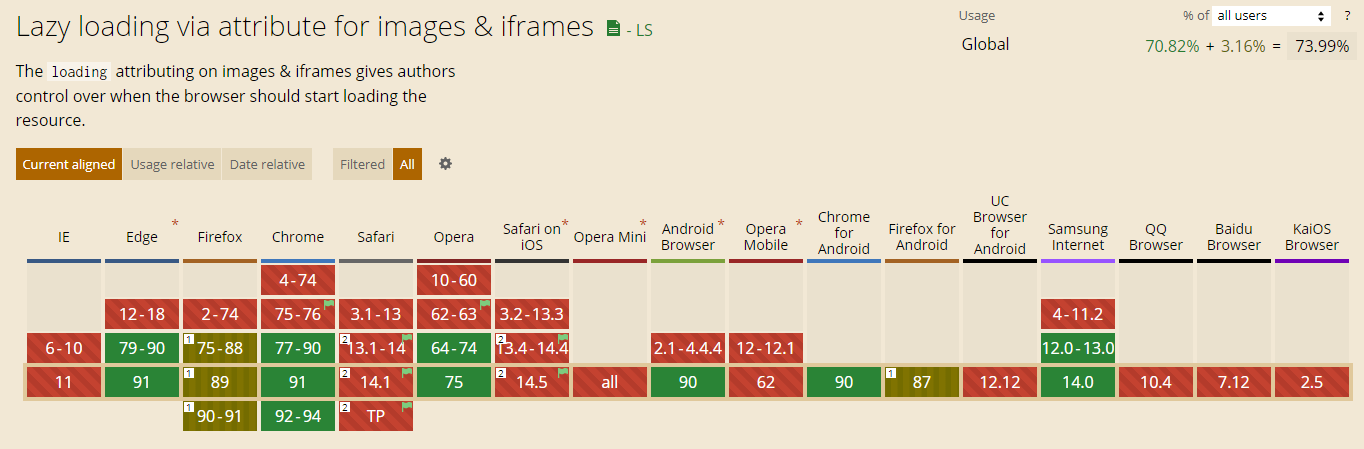
1. Избегайте(при возможности) использования изображений в высоком разрешении.

Зачастую, желание предоставить пользователю изображения в наилучшем разрешении лишь вредит. [Как упоминалось в предыдущей статье](https://odesseo.com.ua/core-web-vitals-noviy-kriteriy-ranzhirovaniya-google-i-kak-teper-kontrolirovat-web-proizvoditelnost/) LCP сильно зависит от состояния сети и вычислительной мощности используемых устройств и высока вероятность, что [пользователи сайта будут использовать устройства с меньшей мощностью, чем вы думаете](https://infrequently.org/2021/03/the-performance-inequality-gap/). Именно поэтому, если возможно, старайтесь не использовать изображения в высоком разрешении, а также не забывайте их сжимать(о чем будет сказано ниже).

1. Используйте [ленивую загрузку](https://web.dev/lazy-loading-images/) для изображений вне зоны видимости.

Используя [“loading” атрибут](https://web.dev/native-lazy-loading/) в <img>, мы можем контролировать поведение загрузки изображения. loading="lazy" - отложенная загрузка изображений до тех пор, пока они не достигнут видимой области просмотра. loading="eager" загружает изображения сразу, независимо от их видимости в области просмотра. eager является значением по умолчанию, поэтому его не нужно явно добавлять (просто используйте <img> для обычной загрузки).

Поддержка данного атрибута все еще находится в переходном состоянии.



***Значение атрибута alt для изображения: Поддержка свойства изображений loading в различных браузерах***

*Поддержка свойства “loading” для изображений в различных браузерах*

В связи с этим, со своей стороны, для отложенной загрузки изображений мы используем JavaScript, чтобы проверить, находятся ли они в области просмотра. Современные браузеры предлагают производительный и эффективный способ выполнения работы по проверке видимости элементов с помощью [Intersection Observer API](https://developers.google.com/web/updates/2016/04/intersectionobserver).

1. Сжатие изображений.

Для оптимизации форматов JPG и PNG попробуйте данные сервисы: <https://imageoptim.com/online> , <https://compressor.io/>, <https://tinypng.com/>, <https://squoosh.app/>, для сжатия SVG - решение Google-разработчика [Jake Archibald](https://twitter.com/jaffathecake) - <https://jakearchibald.github.io/svgomg/>.

1. Предзагрузка шрифтов.

Иногда бывает, что самым крупным элементом страницы может быть элемент текста, который блокирует поток. Чаще всего браузеры не отображают текст, написанный пользовательским шрифтом, до тех пор, пока файл шрифта не будет загружен.

Настройте их предварительную загрузку, чтобы процесс рендеринга текста прошел быстрее.

<link rel="preload" href="[/frontend/fonts/Circe-Bold.woff2](http://yanko.odesseo.com.ua/frontend/fonts/Circe-Bold.woff2)" as="font" type="font/woff2" crossorigin="anonymous">

Кроме того, в нашей доступности есть свойство CSS “font-display”, которое можно использовать, чтобы указать браузеру отображать текст другим шрифтом (из цепочки font-family) во время загрузки файла шрифта.

@font-face {

font-family: 'Circe-Bold';

src: local('Circe-Bold'),

url('../fonts/Circe-Bold.woff2');

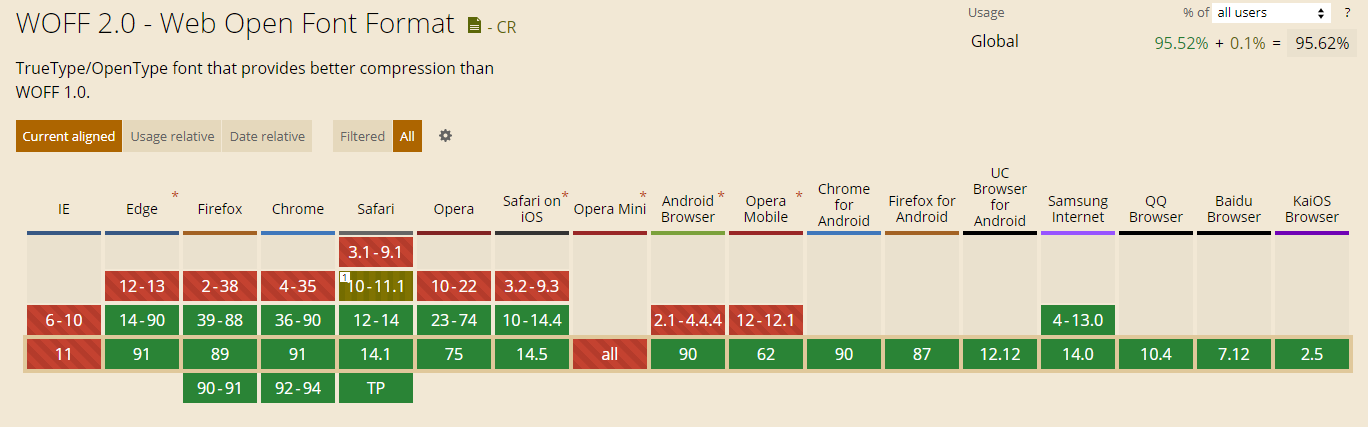
font-display: swap;

}

Если включен font-display: swap, текст отобразится на странице сразу после загрузки кода CSS.

1. Используйте современные форматы шрифтов.

Используйте форматы “woff” и “woff2”. “woff2” использует собственные алгоритмы предварительной обработки и сжатия, чтобы уменьшить размер файла примерно на 30% по сравнению с другими форматами. Уровень поддержки данного формата не оставляет никаких сомнений в необходимости его использования.



***Значение атрибута alt для изображения: Поддержка формата шрифтов woff2 в различных браузерах***

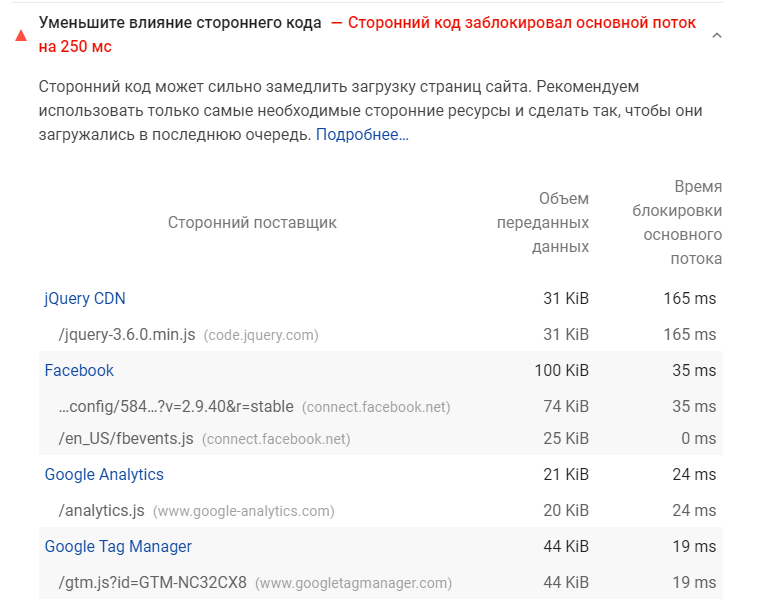
*Поддержка формата шрифтов “woff2” в различных браузерах*

**<h2>FIRST INPUT DELAY (FID)</h2>**

Данный показатель измеряет время, через которое пользователь может взаимодействовать со страницей.

**Советы по улучшению данного показателя:**

1. Уменьшите влияние сторонних ресурсов.



***Значение атрибута alt для изображения: Вкладка «Уменьшите влияние стороннего кода» в Pagespeed Insight***

*Вкладка “Уменьшите влияние стороннего кода” в разделе “Диагностика” в Pagespeed Insight*

Убедитесь, что загрузка сторонних ресурсов несет реальную ценность для вашего сайта.

Как пример приведу один из самых используемых сторонних ресурсов - библиотека jQuery. Когда-то jQuery был отличным решением, когда не было общепринятого способа обращаться к элементам DOM с помощью CSS-селектора, не было стандартного способа добавить анимацию к стилю элемента, а интерфейс [XMLHttpRequest](https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest), предложенный Internet Explorer, как и многие API, был плохо совместим с браузерами. Сейчас же нативные CSS и JS позволяют реализовать подавляющее число задач, поэтому [рассмотрите вариант отказа от jQuery](http://youmightnotneedjquery.com/).

В тех случаях, когда на сайте необходимо использовать карусель изображений, для чего многие зачастую выбирали [“OwlCarousel”](https://owlcarousel2.github.io/OwlCarousel2/) или [“Slick”](https://kenwheeler.github.io/slick/) , которые требуют наличия jQuery, сейчас мы выбираем нативные решения, например [Swiper](https://swiperjs.com/) или [“Glider.js”](https://nickpiscitelli.github.io/Glider.js/), которые имеют маленький вес и свободны от зависимостей. Для AJAX-запросов отдавайте предпочтение уже давно себя зарекомендовавшему [axios](https://github.com/axios/axios) или же нативному [fetch](https://developer.mozilla.org/ru/docs/Web/API/Fetch_API/Using_Fetch).

И все же, если сторонние ресурсы важны для вашего сайта, попробуйте оптимизировать процесс их загрузки:

* Используйте async или defer атрибуты для тега <script>

<script async src="script.js">

<script defer src="script.js">

Используйте async, если важно, чтобы скрипт запустился раньше в процессе загрузки.

Используйте defer для менее важных ресурсов. Например, видеоплеер в нижней части страницы.

* Настройте подключение к данным ресурсам заблаговременно.

Вы можете сэкономить 100–500 мс, установив ранние подключения к важным сторонним источникам. Тут вам помогут два типа подключения тега <link>: preconnect и dns-prefetch.

<link rel = "preconnect" href="<https://cdn.example.com>"> сообщает браузеру, что ваша страница намеревается установить соединение с другим источником и что вы хотите, чтобы процесс начался как можно скорее. Когда делается запрос ресурса из предварительно подключенного источника, загрузка начнется немедленно.

<link rel = "dns-prefetch> обрабатывает небольшое подмножество того, что обрабатывает <link rel =" preconnect ">. Он включает в себя поиск DNS и подтверждение TCP, а для безопасных источников - согласование TLS.

preconnect лучше всего использовать только для наиболее важных соединений; для менее важных сторонних доменов используйте <link rel = dns-prefetch>.

* Настройте ленивую загрузку данных ресурсов.

Встроенные сторонние ресурсы могут быть большим фактором снижения скорости страницы, если они построены плохо. Если они не критичны или находятся в нижней части страницы (т.е., если пользователям нужно прокручивать их для их просмотра), отложенная загрузка - хороший способ улучшить скорость страницы и отрисовку. Таким образом, пользователи быстрее получат контент главной страницы.

Один из самых эффективных подходов - отложенная загрузка стороннего контента после загрузки контента главной страницы. Альтернативный подход - загружать сторонний контент только тогда, когда пользователь прокручивает страницу вниз до этого раздела страницы.

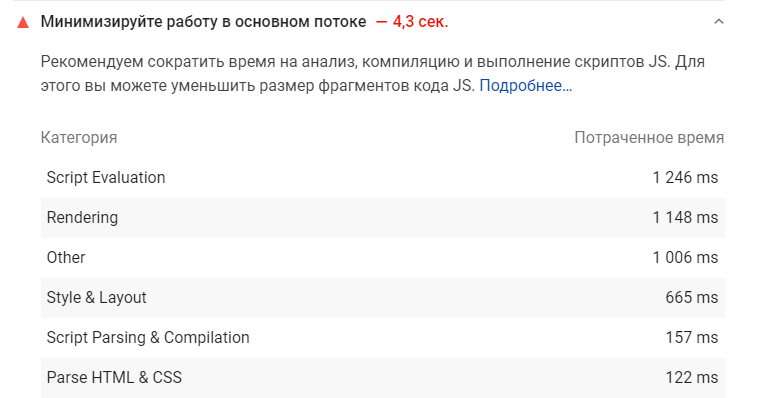
Используйте [Intersection Observer](https://developers.google.com/web/updates/2016/04/intersectionobserver) - API браузера, который эффективно определяет, когда элемент попадает или пропадает из области просмотра браузера, и его можно использовать для реализации этого метода.

1. Уменьшите время выполнения Javascript.

Данный идентичен описанным шагам в разделе LCP. Еще раз отмечу шаги:

* Минимизируйте и сжимайте Javascript.
* Удалите неиспользуемый Javascript
* Отложите выполнение некритического Javascript

1. Минимизируйте работу в основном потоке.

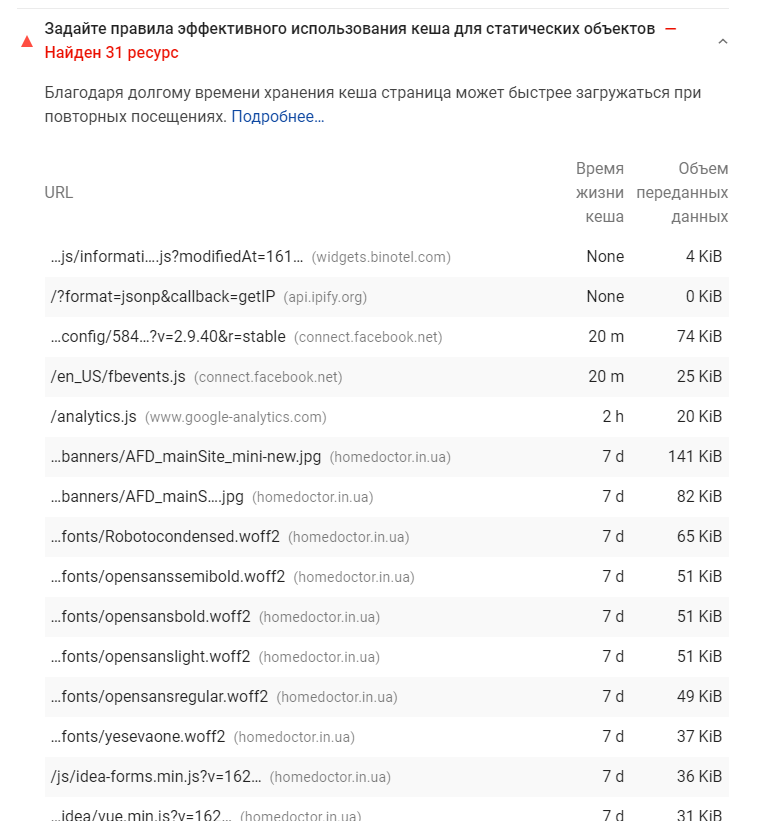


***Значение атрибута alt для изображения: Вкладка «Минимизируйте работу в основном потоке»***

*Вкладка “Минимизируйте работу в основном потоке” в разделе “Диагностика” в Pagespeed Insight*

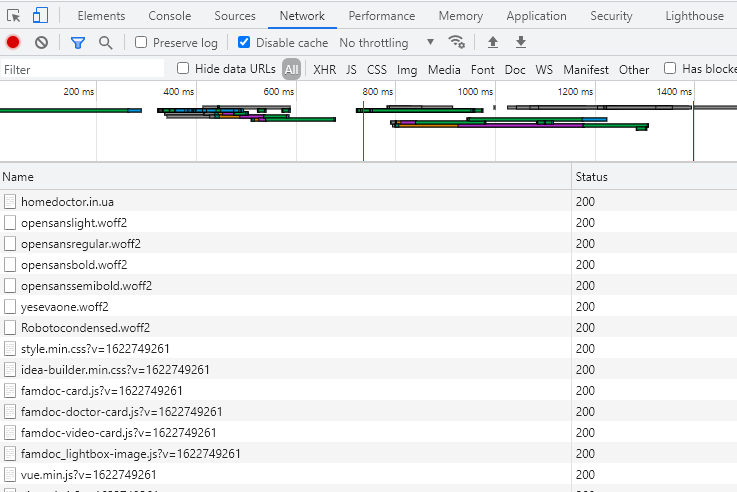
* Уменьшите количество и размер запрашиваемых файлов(как отмечалось ранее).

Немного слов о “количестве ресурсов”. Используйте вкладку “Задайте правила эффективного использования кеша для статических объектов” в “Pagespeed Insight” или вкладку “Network” в Chrome DevTools для того, чтобы посмотреть все ресурсы, которые загружены на странице:



***Значение атрибута alt для изображения: Вкладка «Задайте правила эффективного использования кеша для статических объектов»***

*Вкладка “Задайте правила эффективного использования кеша для статических объектов” в разделе “Диагностика” в Pagespeed Insight*

**

***Значение атрибута alt для изображения: Вкладка «Network» в Chrome DevTools***

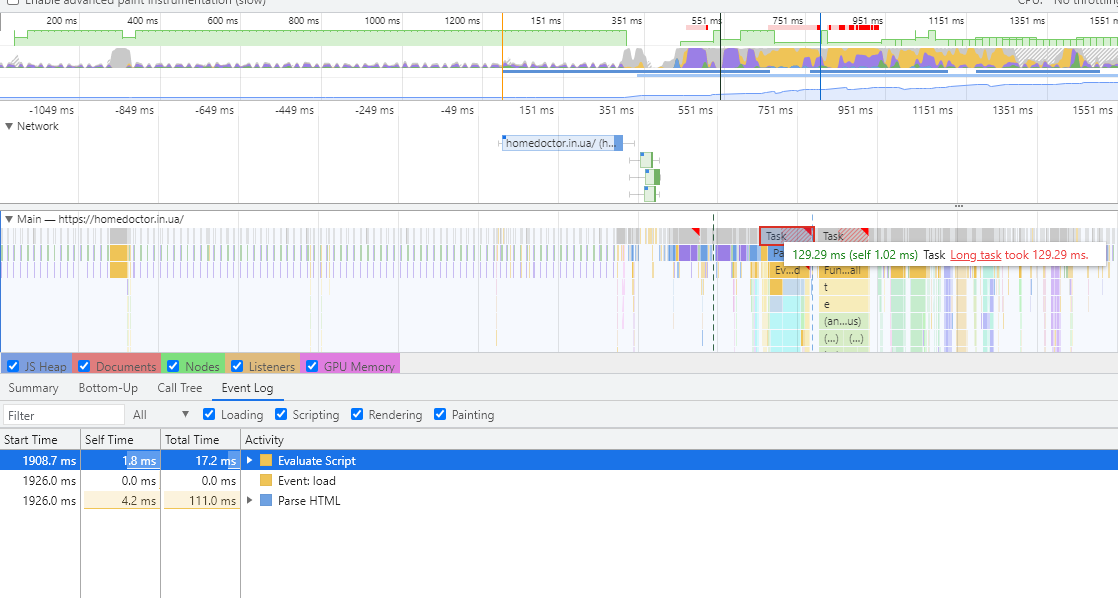
*Вкладка “Network” в разделе Chrome DevTools*

Вкратце отмечу, что мы делаем для того, чтобы уменьшить количество загружаемых ресурсов:

1. Объединяем и минимизируем Javascript (как отмечалось ранее).
2. Объединяем и минимизируем CSS (как отмечалось ранее).
3. Минимизируем использование свойства “background-image” в файлах стилей и “инлайним” графические элементы.

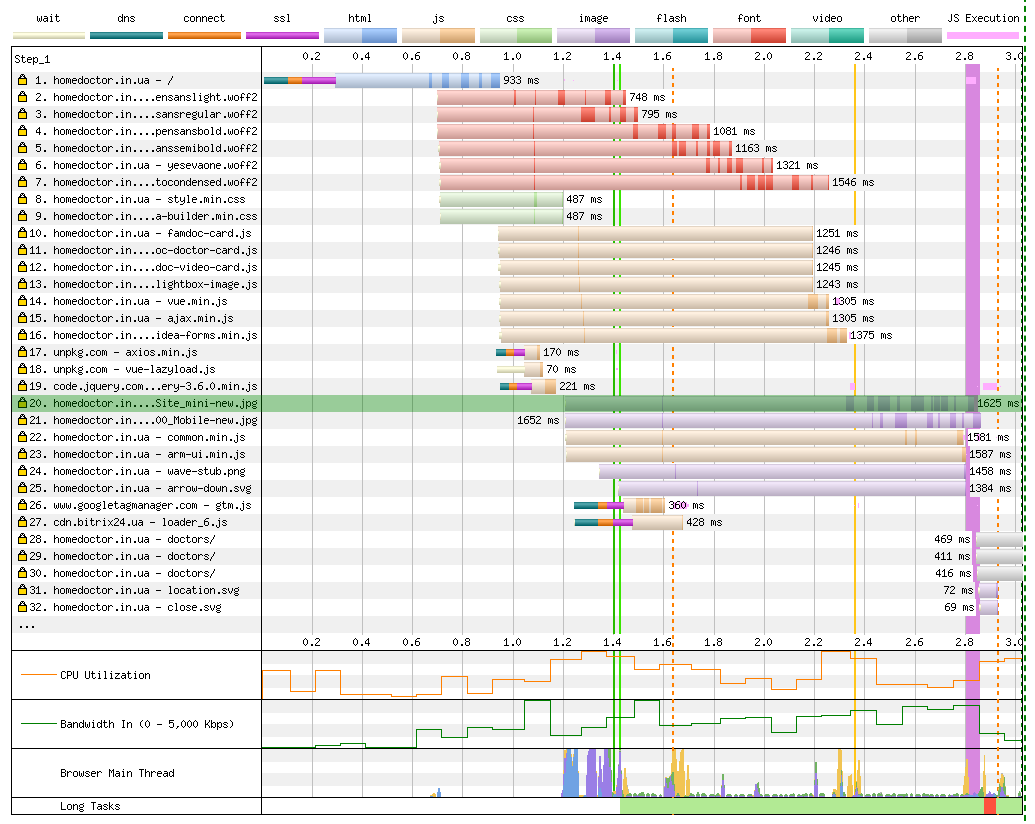
* Уменьшите влияние сторонних ресурсов(как отмечалось ранее)
* Оптимизируйте длительно-выполняющиеся скрипты

Используйте вкладку “Performance” в Chrome DevTools для поиска и отладки длительных операций. Определите, какие процессы являются наиболее затратными и по возможности оптимизируйте их.



***Значение атрибута alt для изображения: Вкладка «Performance» в Chrome DevTools***

*Вкладка “Performance” в Chrome DevTools*

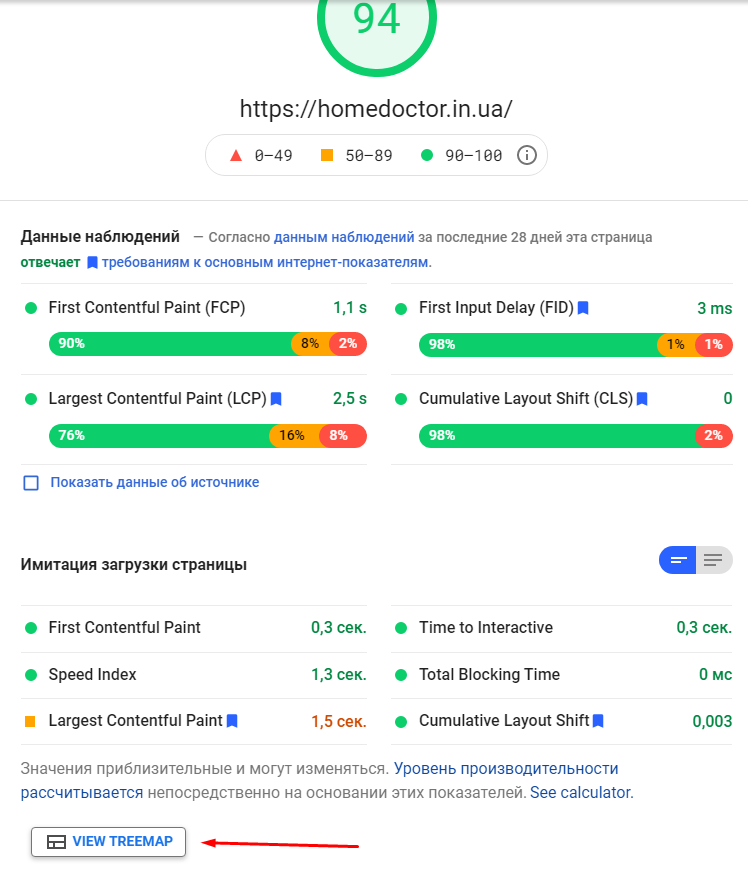


***Значение атрибута alt для изображения: Отображение «Performance» страницы***

*Отображение “Performance” страницы в* [*https://www.webpagetest.org/*](https://www.webpagetest.org/)

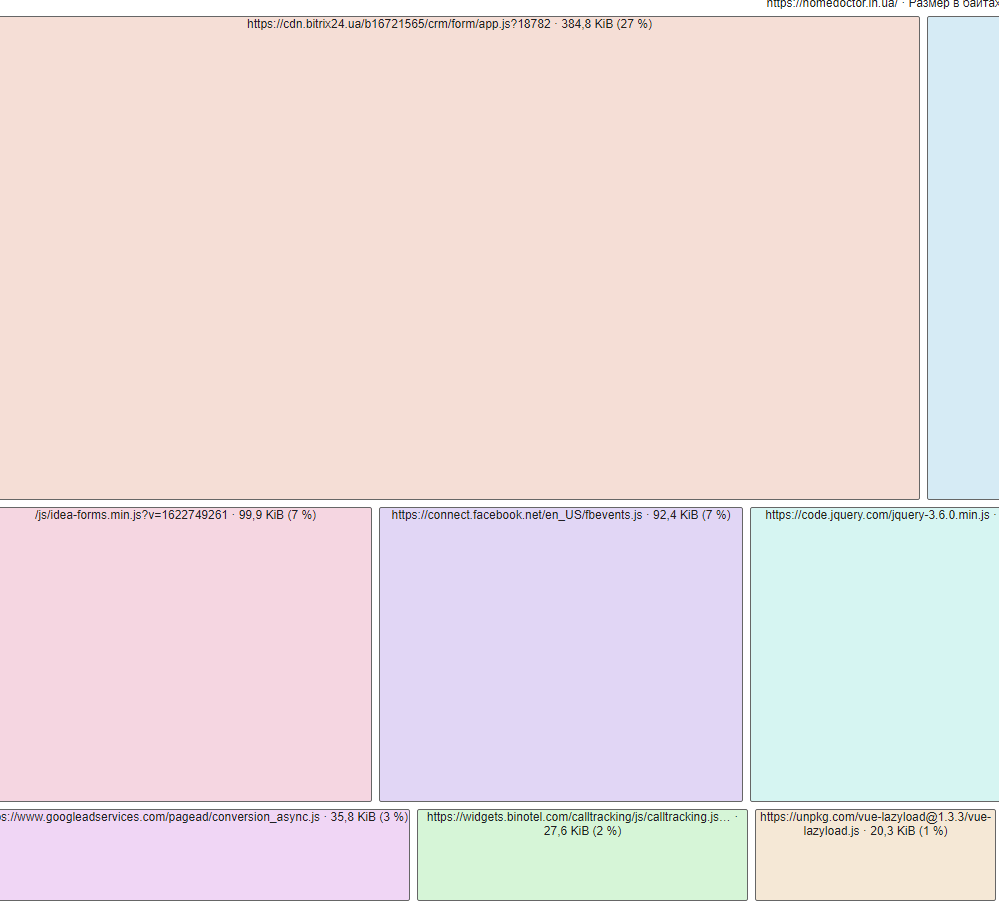
Обратите внимание на процессы, которые отмечены как “Long task” и по возможности оптимизируйте их.

Еще одним способом поиска проблем является новая функция “Pagespeed Insight” - “View treemap”, которая позволяет вам увидеть, что находится в вашем JavaScript, найти зависимости и процент использования в коде.



***Значение атрибута alt для изображения: Вкладка «View Treemap» в Pagespeed Insight***

*Вкладка “View Treemap” в Pagespeed Insight*

**

***Значение атрибута alt для изображения: View Treemap***

*“View Treemap”*

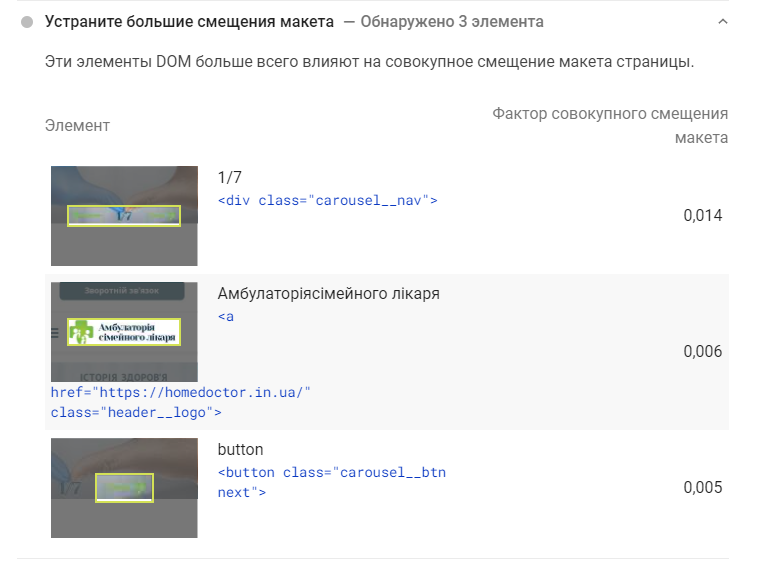
* Загружайте Javascript асинхронно(как отмечалось ранее)
* Используйте методы [“прогрессивного улучшения”](https://htmlacademy.ru/blog/boost/frontend/short-9), чтобы пользовательский интерфейс можно было использовать до запуска JavaScript

# [Ограничьте объем и сложности вычисления стилей](https://developers.google.com/web/fundamentals/performance/rendering/reduce-the-scope-and-complexity-of-style-calculations)

# [Не используйте большие сложные макеты и избегайте их подтормаживания](https://developers.google.com/web/fundamentals/performance/rendering/avoid-large-complex-layouts-and-layout-thrashing)

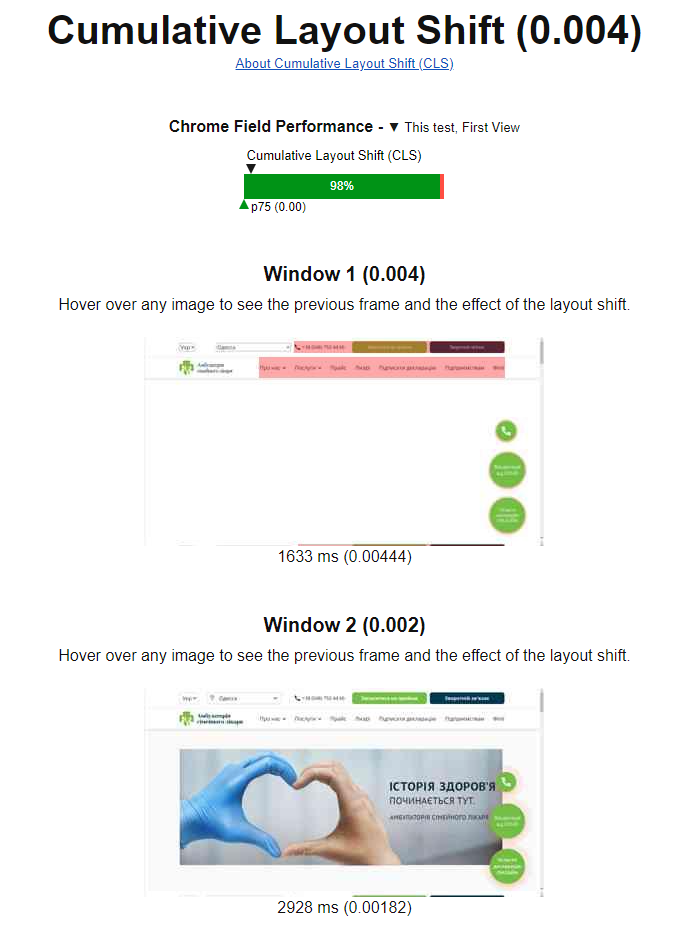
**<h2>Cumulative Layout Shift</h2>**

Данный показатель предназначен для измерения визуальной стабильности страницы и отслеживает — насколько контент страницы сдвигается, когда новые элементы отрисовываются на своих местах страницы.

****

***Значение атрибута alt для изображения: Вкладка «Устраните большие смещения макета»***

*Вкладка “Устраните большие смещения макета” в разделе “Диагностика” в Pagespeed Insight*

****

***Значение атрибута alt для изображения: Информация о смещениях в макете сайта***

*Информация о смещениях в макете сайта на* [*https://www.webpagetest.org/*](https://www.webpagetest.org/)

**Советы по улучшению данного показателя:**

1. Резервирование места для изображений, видео и iframe элементов путем указания размеров.

Чтобы [уменьшить](https://web.dev/optimize-cls/#images-without-dimensions) сдвиг раскладки, вызванный загрузкой ресурсов без заданных размеров, задайте картинкам и видео атрибуты width и height. Это помогает браузеру выделить достаточное место на странице, пока элементы грузятся.

<img src="image.jpg" alt="..."

width="500" height="500">

Если ваше изображение должно быть отзывчивым, задайте ему в стилях свойство “width” со значением “100%”, а height со значением “auto”. Тем самым мы может удовлетворить требования браузера и контролировать размеры изображения на медиа-запросах:

img {

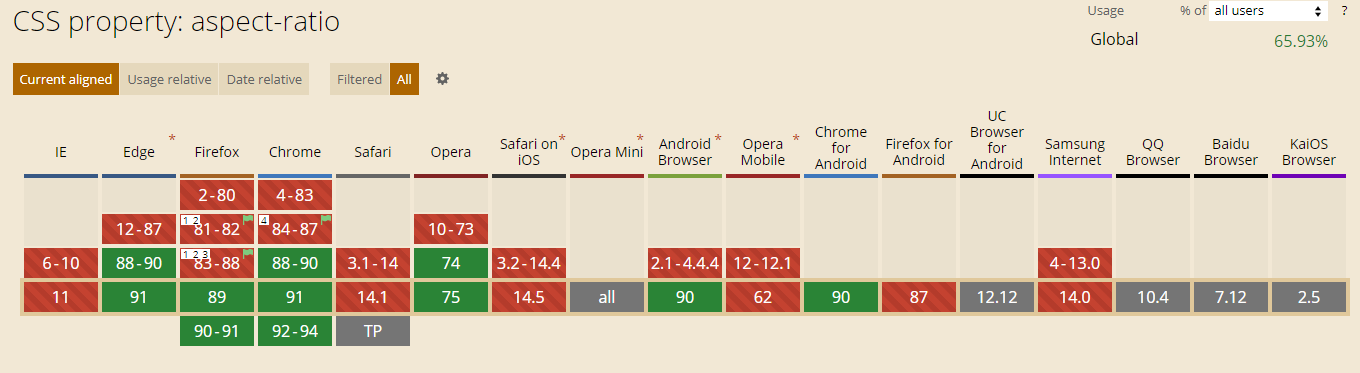
width: 100%;

height: auto;

}

Читайте статью «[Setting Height And Width On Images Is Important Again](https://www.smashingmagazine.com/2020/03/setting-height-width-images-important-again/)» для лучшего понимания важности указания размеров и соотношения сторон для изображений.

1. Если возможно, используйте CSS свойство [aspect-ratio](https://developer.mozilla.org/Web/CSS/aspect-ratio), которое рассчитывается путем соотношения сторон, что в результате зарезервирует место для элемента. Поддержка данного свойства находится в переходном состоянии, однако сообщество полагает на него большие надежды.

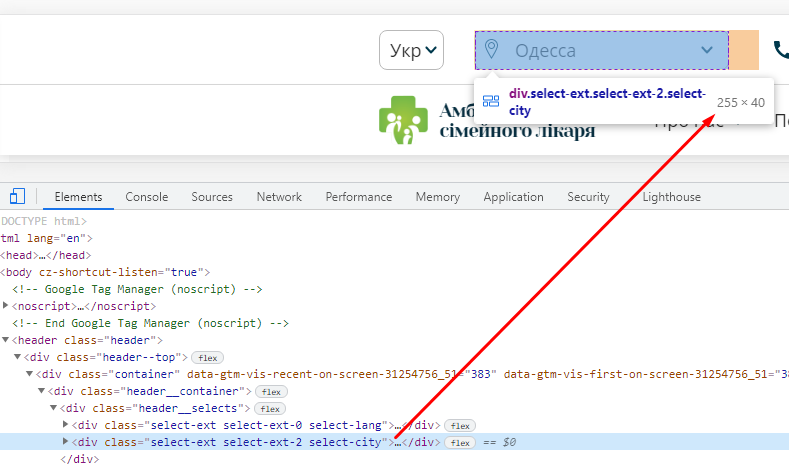


***Значение атрибута alt для изображения: Поддержка CSS свойства aspect-ratio в различных браузерах***

*Поддержка CSS свойства “aspect-ratio” в различных браузерах*

1. Настройте предварительную загрузку шрифтов и используйте CSS свойство font-display: swap, что в итоге предотвратит смещения текста в макете(как отмечалось ранее).
2. Постарайтесь избегать динамической вставки элементов, в особенности в верхней части страницы, чтобы не повлиять на процесс взаимодействия пользователя со страницей.

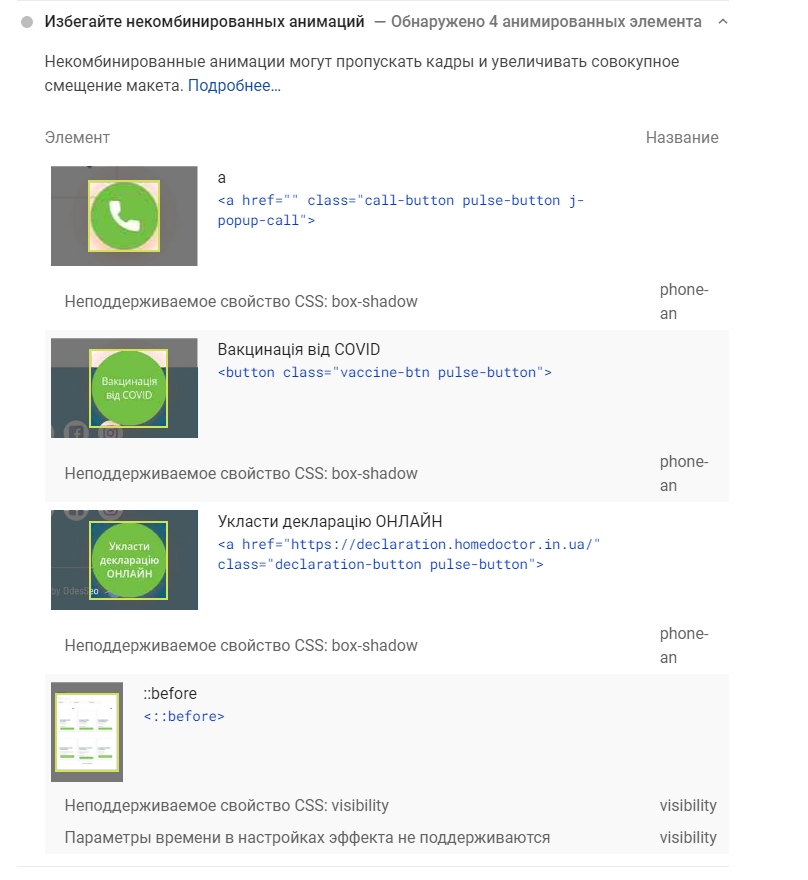
Мы часто используем свое решение для генерации интерактивных компонентов: попапов, дропдаунов, селектов, табов, аккордеонов, других различных виджетов. В процессе инициализации многие из них используют динамическую вставку разметки. Решение проблемы сдвига контента в данном случае решается очевидным способом - указанием атрибутов width и height:



***Значение атрибута alt для изображения: Указание размеров для динамических элементов улучшает показатель CLS***

*Указание размеров для динамических элементов улучшает показатель CLS*

1. Избегайте раздельных анимаций.

**

***Значение атрибута alt для изображения: Вкладка «Избегайте некомбинированных анимаций»***

*Вкладка “Избегайте некомбинированных анимаций” в разделе “Диагностика” в Pagespeed Insight*

Избегайте использования свойств “box-shadow” и “filter”, которые браузер должен пересчитывать каждый раз, когда прорисовывает элемент. Анимации, не объединенные в общий композитный слой рендеринга, могут дёргаться на слабых устройствах, если исполнение сложных JavaScript-тасков занимает главный поток. Такие анимации могут вызывать и сдвиги раскладки. Если Chrome обнаруживает, что анимация не может быть выделена в отдельный слой, он сообщает об этом в DevTools. Это позволяет составить список всех элементов, для которых анимация не была композитной и выяснить причину. Вы можете найти эту информацию в отчете «[Avoid non-composited animations](https://web.dev/non-composited-animations/)».

**<h2>Вывод</h2>**

Надеюсь, данная статья поможет вам улучшить ключевые показатели нового критерия ранжирования Google - Core Web Vitals и быть готовым к изменениям, которые грядут в вебе.

Вопрос улучшения веб-производительности вызвал большой ажиотаж в сообществах и конечно же, преследует благую цель. [Команда Chrome постоянно занимается доработкой и исправлением данных критериев](https://groups.google.com/a/chromium.org/g/chrome-ux-report/c/uI5m8Zuh2v8?pli=1) и несмотря на то, что в любой момент возможны “breaking changes”, основа данных критериев скорее всего останется нетронутой, поэтому показателями веб-производительности необходимо заняться прямо сейчас.

**Метатеги:**

***Title:***

Как улучшить показатели Core Web Vitals

**Description:**

Как улучшить показатели веб-производительности Core Web Vitals. Читайте подробные рекомендации в новой статье от специалистов OdesSeo!