УДК 681.3

Кожантаев Тимур Рафаэлевич, магистрант,
Ошский технологический университет

СОВРЕМЕННЫЕ СПОСОБЫ ОПТИМИЗАЦИИ ПРОИЗВОДИТЕЛЬНОСТИ ВЕБ-САЙТОВ

В статье освещены современные способы, которые могут помочь ускорить работу веб-сайтов, показаны и описаны примеры их использования, с помощью интернет инструментов протестирована эффективность некоторых методов.

Ключевые слова: Веб-сайт, сервер, DOM (Интерфейс программирования приложений), HTML (Стандартизированный язык разметки документов), CSS (Каскадные таблицы стилей), Javascript или JS (Язык программирования), HTTP (Протокол передачи гипертекста).

Kozhantaev Timur Rafaelevish graduate student,
Osh Technological University

MODERN WAYS TO OPTIMIZE WEBSITE PERFORMANCE

The article highlights the ways, that can help with website speed and performance optimization, the examples of usage are shown and descripted, also the efficiency of some ways are tested with internet free tools.

Key words: website, DOM (Document Object Model), HTML (Hyper Text Markup Language), Javascript or JS (computer programming language), CSS (Cascading Style Sheets), HTTP (Hypertext Transfer Protocol).

Кожантаев Тимур Рафаэлевич, магистрант,
Ош технологиялык университети

ВЕБСАЙТТАРДЫН ИШМЕРДҮҮЛҮГҮН ОПТИМДАШТЫРУУНУН ЗАМАНДЫ ЖОЛДАРЫ

Макалада веб-сайттарды тездетүүгө жардам бере турган заманбап ыкмалар баса белгиленет, аларды колдонуунун мисалдары көрсөтүлөт жана сүрөттөлөт, ошондой эле Интернет куралдарын колдонуу менен кээ бир ыкмалардын натыйжалуулугу текшерилген.

Ачкыч сөздөр: Веб-сайт, сервер, DOM (Application Programming Interface), HTML (Standardized Document Markup Language), CSS (Cascading Style Sheets), Javascript же JS (Programming Language), HTTP (Hypertext Transfer Protocol).

Введение. По статистике половина интернет-пользователей закрывает сайт, если он загружается более 3 секунд. Почти 80% пользователей не возвращаются, если сайт работает плохо, даже если контент более информативен, чем на альтернативных ресурсах. В большинстве случаев пользователь отдает предпочтение интернет ресурсам с хорошей производительностью и скоростью загрузки.

Очевидно, что скорость загрузки существенно влияет на успех бизнеса. В данной статье рассматриваются полезные способы, которые помогут увеличить скорость загрузки веб-сайтов.

Существует много способов повысить скорость веб-сайта, и в этой статье будет рассмотрены основные актуальные стратегии и лучшие практики на сегодняшний день. Реализация даже 1-2 описанных подходов должна помочь веб-сайту работать быстрее.

Цель исследования. Основной целью исследования являются определение и тестирование современных способов оптимизации производительности веб-сайтов и веб-приложений. Исползуемые спороспособы улучшения работы веб-ресурсов несут практическую ценность для применения в учебных и коммерческих разработках.

Материалы и методы исследования. При реализации данного иследования предворительно были собраны и протестированы 27 методов оптимизации работы веб-ресурсов. Далее по следующим критериям: простота применения, эффективность применения и наличие побочных эффектов были выбраны лучшие способы.

Метод откладывания загрузки внешнего файла JavaScript в процессе обращения к странице веб-сайта — один из наиболее важных и сложных вопросов. Отложенная загрузка внешних скриптов, позволяет странице сайта загрузиться намного быстрее, нежели загрузка страницы последовательно, строчка за строчкой исходного кода. Для web-сайтов с относительно несложной архитектурой скорее всего будет достаточно поместить вызывающий скрипт в конец раздела<body>, и дело будет решено. Однако в сложных проектах дело обстоит иначе. Приведенное на (рис.1) решение позволит загружать внешний скрипт только после полной загрузки страницы и не вызовет ошибки «Defer loading of Javascript» в инструментах для веб-разработки от Google.

```
16
17    <script type="text/javascript">
18      function downloadJSAtOnload() {
19        let element = document.createElement("script");
20        element.src = "script.js";
21        document.body.appendChild(element);
22      }
23      if (window.addEventListener)
24        window.addEventListener("load", downloadJSAtOnload, false);
25      else if (window.attachEvent)
26        window.attachEvent("onload", downloadJSAtOnload);
27      else window.onload = downloadJSAtOnload;
28    </script>
```

Рис. 1 Метод загрузки рекомендуемый Google

Данный код «сообщает» браузеру о необходимости дождаться завершения полной загрузки страницы и последующего исполнения скрипта script.js.

Метод асинхронной загрузки JavaScript: атрибуты Defer и Async. Асинхронная загрузка JavaScript — это разновидность загрузки кода в многопоточном режиме позволяющая оптимизировать время загрузки веб-сайта

При синхронной загрузке когда браузер находит строку кода <script src = “script.js”></script> создание моделей DOM and CSSOM останавливается на время загрузки кода Javascript (рис.2). По этой причине в коммерческих проектах ссылка на внешний JS файл помещается после основного кода HTML.

Для понимания специфики и различий асинхронной и синхронной загрузки, ниже рассмотрено несколько примеров.

```
<html>
<head>
<script src="main.js"></script>
</head>

<body>
Данный текст не отобразится на странице пока не загрузится  внешний файл JS
</body>
</html>
```

Рис. 2 Пример размещения ссылки на JS файл в тэге <head>

Чтобы решить проблему можно добавить атрибут async в <script asyns></script>, чтобы создание модели DOM происходило параллельно и не прерывалось во время загрузки и выполнения JavaScript.

Однако следует быть осторожным, если JS код должен выполнять некоторые манипуляции с HTML или CSS, или если вы загружаете скрипт в строгом порядке.
Например, если на веб-сайте используется популярный bxSlider и для подключения библиотеки jQuery используется CDN внешнего интернет источника. Чтобы подключить bxSlider, вы можете добавить этот код в HTML на (рис.3)

```
<html>
<head>
  <!-- jQuery библиотека  -->
<script src="//ajax.googleapis.com/ajax/libs/jquery/1.8.2/jquery.min.js"></script>
<!-- bxSlider  файл Javascript  -->
<script src="/js/jquery.bxslider.min.js"></script>
<!-- bxSlider файл CSS  -->
<link href="/lib/jquery.bxslider.css" rel="stylesheet">
</head>
```

Рис. 3 Подключение bxSlider.

Как видно на (рис.3), jQuery загружается из Google CDN, при этом код bxSlider является локальным. Если мы добавим атрибут async в строку, которая включает jQuery, это может привести к ошибкам со слайдером, так как файл jquery.bxslider.min.js загружается до jquery.min.js. В этом случае порядок очень важен, поэтому хорошим решением будет использование тэга defer.

В случае если браузер увидит тег defer с кодом JavaScript, загрузка моделей DOM и CSSOM не остановится. Все скрипты с атрибутом defer будут загружены и запущены сразу после завершения создания моделей DOM и CSSOM. Скрипты будут загружены в том порядке, в котором вы расположены коде

```
<script src="1.js" defer></script>
<script src="2.js" defer></script>
```

Рис. 4 Порядок загрузки JS файлов относительно расположения

На (рис.4), выше, скрипт 2.js загрузится только после файла 1.js.

Атрибуты defer и async применяются только на для внешних файлов JS.Если использовать данные атрибуты для внутренних скриптов, то они будут проигнорированы.

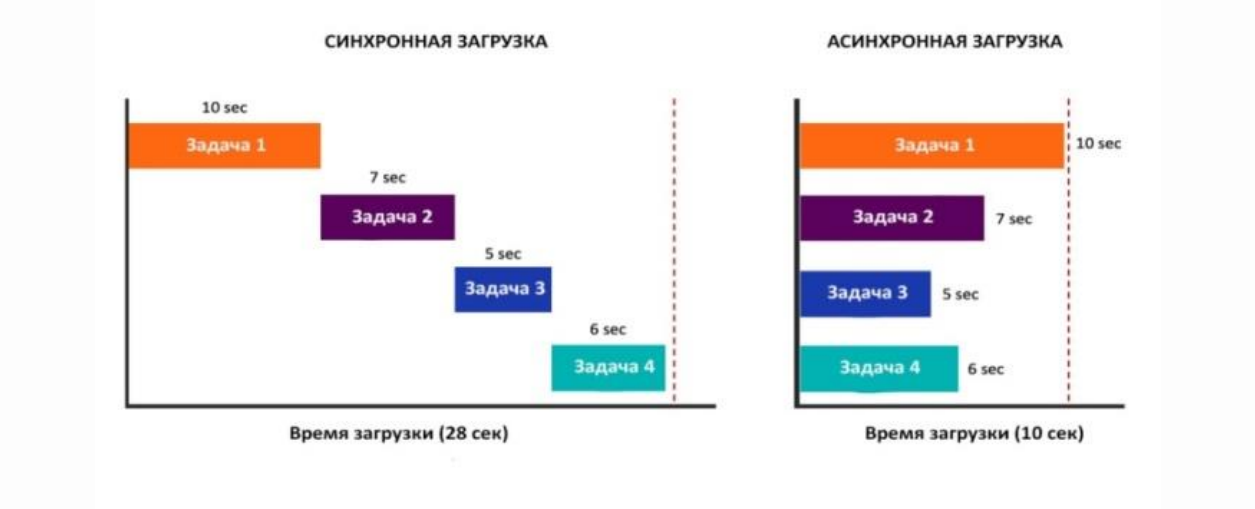

Рис. 5 Результаты исследования эффективности при асинхронной загрузке JS кода.

Замена JavaScript кода на CSS3. Старые версии CSS 1.0 и 2.0 не давали широкие возможностей для реализации определенных эффектов на веб-сайтах, поэтому было необходимо прибегать к JS для написания логики для более сложных задач по оформлению интерактивных элементов. Сегодня версия CSS 3.0 является достаточно мощным языком с большим количеством дополнительных функций, которые могут в некоторых случаях заменить Javascript. Преимуществом CSS является то, что код можно предварительно скомпилировать, поэтому CSS гораздо меньше загружается CPU чем Javascript. На (рис.6), ниже можно посмотреть, как написать код для слайдера на чистом CSS

```css
@keyframes simple-slider {
    0% {  left: 0%;}
    20% {left: 0%;}
    25% {left: -100%;}
    45% {left: -100%;}
    50% {left: -200%;}
    70% {left: -200%;}
    75% {left: -300%;}
    95% {left: -300%;}
    100% {left: -400%;}
}
body {
    margin: 0;
}
div#slider-css {
    overflow: hidden;
}
div#slider-css figure img {
    width: 20%;
    float: left;
}
div#slider-css figure {
    position: relative;
    width: 500%;
    margin: 0;
    left: 0;
    text-align: left;
    font-size: 0;
    animation: 30s simple-slider infinite;
    -webkit-animation: 30s simple-slider infinite;
}
```

Рис. 6 CSS код слайдера.

Использование протокола HTTP/2. В Основе протокола HTTP/2 лежит SPDY от Google. Это самый новый стандарт, который уже поддерживается Chrome, Firefox, Opera, Safari и другими браузерами. Основными задачами HTTP/2 протокола являются предоставление возможности клиентам/серверам выбирать протокол, реализация

совместимости с HTTP/1.1, снижение задержки загрузки – веб-страниц, а также поддержание существующих способов использования HTTP. [1]

Основная цель разработки данного протокола – это ускорение загрузки страниц. Высокая производительность достигается путем использования специальной методики – мультиплексирования потоков. Что оно собой представляет? Пакеты нескольких потоков смешиваются в одном соединении, после чего разделяются по прибытию. Помимо мультиплексирования, данный протокол будет использовать сжатие заголовков, а также расстановку приоритетов запросов. Часть представленных функций уже была использована в SPDY. [2]

Результаты тестов (рис.7), проведённых с помощью интернет ресурсов https://www.httpvshttps.com/ и https://http2.akamai.com/demo показывают однозначное преимущество использование протокола HTTP/2.

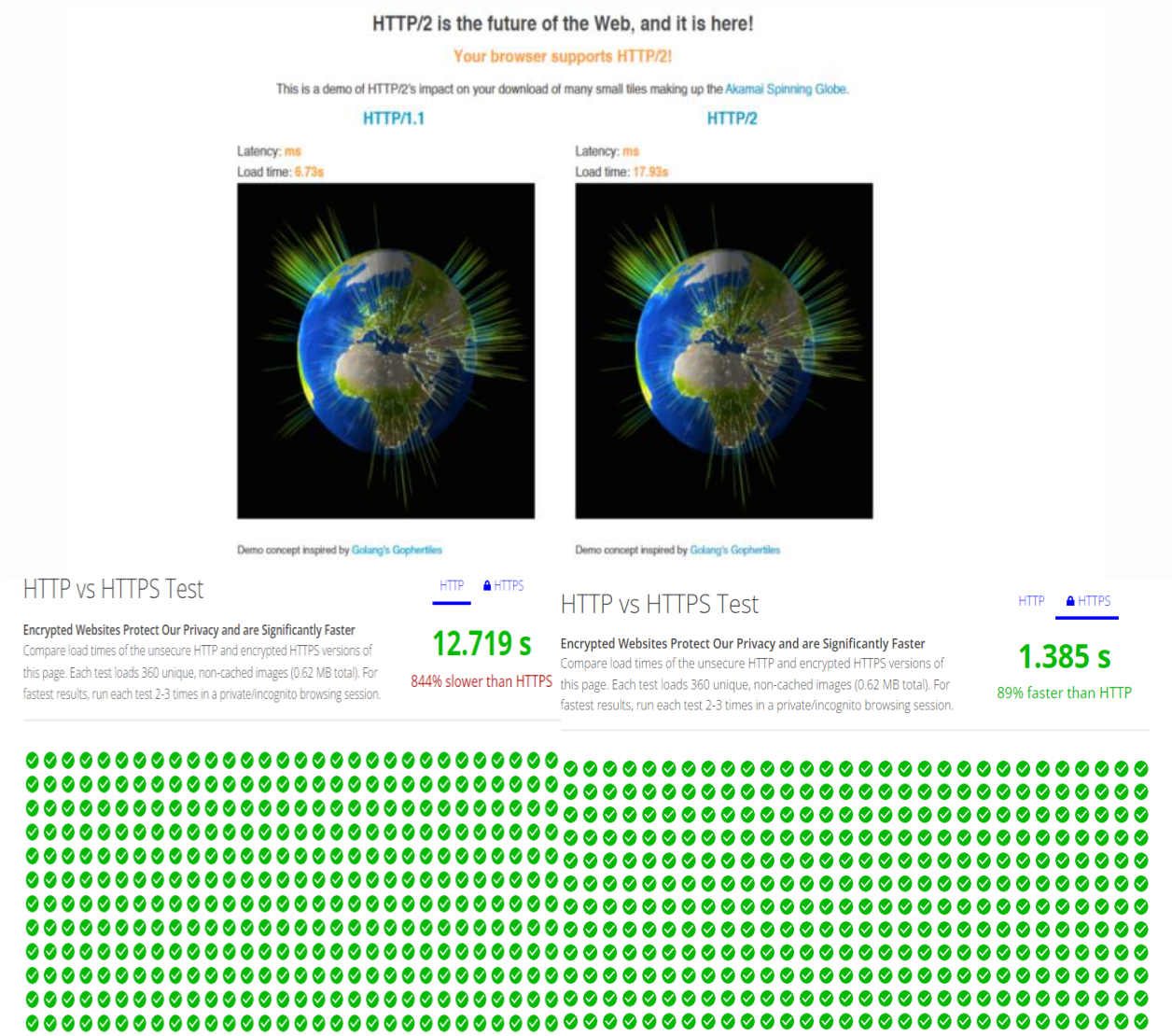

Рис. 7 Сравнение скорости передачи данных при использовании протоколов HTTP и HTTP/2.

Результаты исследования. Результаты применения описанных в статье методов оптимизации работы веб-сайта по отдельности приведены на рисунках 5 и 7. Таким образом скорость рендеринга при асинхронной увеличилась более чем на 64%. В случае использование протокола HTTP/2 вместо HTTP выросла 8.4 раза. Общие же показатели по результатам применения всех описанных методов улучшения производительности одного и того же веб-приложения показаны на (рис 8).

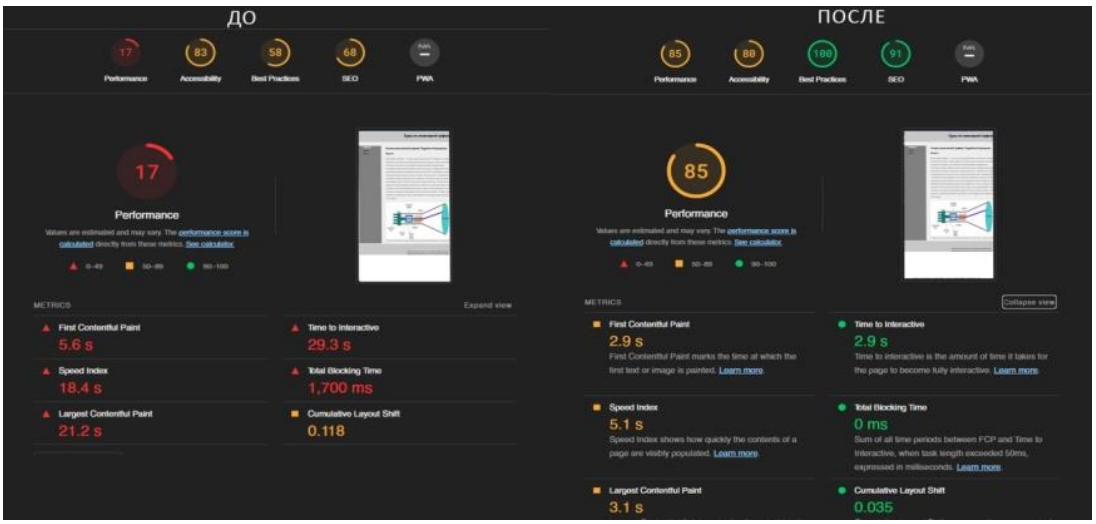

Рис. 8 Общие показатели эффективности предлагаемых методов оптимизации веб-приложения.

Выводы. Помимо представленных в статье способов оптимизации кода, существует множество других методов, технологий для увеличения производительности веб-сайтов. Важно понимать, что для каждого случая определяющее значение имеет индивидуальный подбор методов и инструментов, в зависимости от функционала, структуры и сложности веб-сайта. Так для простых информационных веб-сайтов наверняка будет достаточным расположить ссылки на внешние файлы в правильном порядке, в то время как для сайтов со сложной структурой и большой базой данных рекомендуется использовать более продвинутые, современные техники. Однако явное улучшение показателей при тестировании веб-приложения приводят к выводу о обязательном использовании описанных в статье универсальных методов для большинства веб-ресурсов.

Литература:

1. Асинхронный Javascript // Сайт веб-документации MDN - https://developer.mozilla.org – 2023. – 02 февраля.
2. Производительность CSS и JavaScript Сайт веб-документации MDN - https://developer.mozilla.org – 2023. – 05 февраля.
3. **Николай М.Р.** Методы клиентской оптимизации веб-страниц [Текст] // Бином. Лаборатория знаний, Интернет-университет информационных технологий. 2009. – с. 129.
4. Ilya Grigorik High-Performance Browser Networking [Текст] // O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472. 2013. – с. 241-242.