Анализ и разработка алгоритмов, оптимизирующих компоновку html документа для браузеров

С. В. Жуков, О. А. Ковалева, С. В. Ковалев

Тамбовский государственный университет имени Г. Р. Державина, Тамбов, Россия

Аннотация. В статье приведен обзор метрик сервиса Google PageSpeed Insights (на базе Lighthouse 10 версии), позволяющих определить скорость загрузки веб страницы. На основании приведенных метрик собраны и проанализированы методы оптимизации html контента. Комплексное применение приведенных методов, а также корректная конфигурация сервера, способного быстро отдавать сгенерированный код, позволят получать высокую оценку скорости загрузки страницы на основании рассмотренных ранее метрик. Приведена авторская реализация алгоритмов, способных автоматически изменять компоновку контента html документа, для увеличения скорости отображения содержимого web страницы.

Ключевые слова: оптимизация, html, веб страница, скорость загрузки, PageSpeed Insights, компоновка веб контента.

DOI 10.14357/20718632230312

Введение

Количество сайтов в сети интернет неуклонно растет. Для того чтобы выбрать из миллиардов страниц те, которые будут наилучшим способом подходить под запрос пользователя, поисковые системы анализируют множество факторов. Скорость загрузки веб ресурса является одним из них. Ресурсы, предоставляющие контент быстрее, имеют большие шансы занять лидирующие позиции при выдаче в поисковых системах. Кроме этого, оперативный отклик сайта на запрос уменьшит вероятность того, что пользователь покинет ресурс, не дождавшись его загрузки, что в свою очередь делает скорость загрузки страницы также важным аспектом для улучшения поведенческих факторов [1].

На сегодняшний день остается актуальной задача оптимизации веб контента для быстрой передачи пользователям. Для ее решения разрабатываются новые методы кеширования страниц, настраиваются и балансируются по нагрузкам серверы, обрабатывающие запросы, и проводятся работы по оптимизации компоновки контента на страницах сайтов. Последний процесс является одним из важных факторов, определяющих скорость загрузки страницы. Методы его оптимизации будут актуальны для ресурсов, работающих на разных технологических платформах.

Таким образом, целью данной работы является увеличение скорости процесса загрузки и отображения веб контента в современных браузерах.

Для достижения цели требуется решить следующие задачи:

- рассмотреть метрики, которые позволяют измерить скорость загрузки страницы;

- провести анализ методов, позволяющих ускорить загрузку веб страницы;
- разработать алгоритмы для автоматической компоновки html документа в соответствии с оптимальным расположением ресурсов для обеспечения высокой скорости загрузки контента.

1. Обзор основных метрик PageSpeed Insights

Для оценки скорости загрузки сайта существует несколько инструментов. Чаще всего используются веб-сервисы, такие как GTmetrix, WebPageTest или Google PageSpeed Insights (далее PageSpeed). Они предоставляют «синтетическую» оценку скорости загрузки страницы. То есть, их оценка получена исходя только из замеров определенных метрик и не отображает реального опыта взаимодействия пользователя с сайтом. Каждый сервис использует свой набор метрик для оценки сайта и предоставляет разные возможности для их анализа. Оценка в большинстве случаев будет меняться от теста к тесту даже без изменений контента на сайте из-за того, что на доставку контента влияет не только его компоновка, но и среда передачи данных. Поэтому нет ничего критичного в том, что сайт не набирает максимальный бал в тестах. Но стремится к нему важно, так как именно этот синтетический бал позволит понять с какими проблемами скорее всего столкнутся пользователи.

Для анализа скорости загрузки сайта можно использовать сразу несколько инструментов, что позволит видеть более полную картину работы ресурса. При оптимизации web-ресурса часто используется сервис PageSpeed от Google [2]. Выбор этого сервиса обусловлен тем, что он показывает обобщенные балы основных метрик на основе данных, полученных от реальных пользователей, и эти данные учитываются при ранжировании страниц в поисковой системе Google. Таким образом, он позволяет буквально определить, что необходимо улучшить для более эффективного продвижения в поисковой выдаче. Всего PageSpeed использует шесть основных метрик:

First Contentful Paint (FPC) - время с момента начала загрузки страницы до момента, когда какая-либо часть содержимого страницы отобразится на экране. [3]

Speed Index (SI) – насколько быстро контент визуально отображается во время загрузки страницы. [4]

Total Blocking Time (TBT) – общее количество времени между первой отрисовкой контента и временем до интерактивности страницы, когда выполнялась задача, длительностью более 50 миллисекунд. [5]

Largest Contentful Paint (LCP) - время рендеринга самого большого изображения или текстового блока, видимого в области просмотра, отсчитанное от момента начала загрузки страницы. [6]

Cumulative Layout Shift (CLS) – показатель по временному окну с максимальными оценками смещения макета для каждого неожиданного смещения макета, которые происходят в течение всего времени отрисовки страницы. [7]

Метрики измеряются при помощи Google Lighthouse – сервиса с открытым исходным кодом для анализа качества веб страниц. Итоговый бал рассчитывается исходя из того, на сколько хорошо страница показала себя в каждой из перечисленных метрик. Их влияние на итоговую оценку не одинаковое. На Рис. 1 представлен процент влияния каждой метрики на итоговую оценку.

Для мобильных и десктопных устройств процент влияния метрики на итоговую оценку одинаковый, но границы того, какой результат метрики считать приемлемым, отличается в связи с тем, что мобильные устройства в большинстве своем уступают по техническим характеристикам среднему персональному компьютеру. Оптимизация контента и кода для них требует большего внимания. Так, например, одна и та же страница будет быстрее начинать отображаться на персональном компьютере, чем на смартфоне

Рис. 1. Процент влияния метрики на итоговый бал скорости загрузки страницы

из-за того, что процессору компьютера потребуется меньше времени для расчета данных страницы.

2. Оптимальная компоновка данных

Рассмотрев метрики, оценивающие скорость загрузки страницы, можно выделить ряд качеств, которыми должен обладать документ для получения высокого бала производительности. Документ должен как можно быстрее начать отображаться (в том числе частично) в окне пользователя, за минимально возможное время полностью показать самый крупный элемент страницы и начать реагировать на действия пользователя, при этом оперативно подгружать остальной контент страницы, не сдвигая уже отрисованный макет.

Для того, чтобы страницы веб-ресурса соответствовали данным характеристикам, следует проверить корректно ли настроен веб сервер и соответствует ли следующим критериям: страницы и ресурсы отдаются с 200 откликом, цепочка сертификатов ssl полностью задана (при наличии), использует современную версию HTTP протокола (не ниже HTTP/2), для статических ресурсов задано адекватное время кеширования, передаваемые данные сжимаются, ресурсов сервера достаточно для обеспечения отдачи страницы после запроса в течении 200-300 миллисекунд. Последний пункт может быть выполнен как подбором подходящего оборудования, так и путем оптимизации кода веб-сайта. Так, например, разработчики продуктов «1С-Битрикс» приводят результаты замеров производительности своей системы на разных серверах [8]. Низкая оценка ресурсов сервера коррелирует с увеличением времени, затрачиваемым на формирование страниц. В оценку производительности входят такие показатели, как количество файловых операций в секунду и количество операций в секунду, выполняемых процессором. Эти показатели зависят от серверного оборудования. В тоже время код сайта также имеет большое влияние на скорость формирования страницы. Например, в системе «1С-Битрикс» выборка товаров по id из базы данных за один запрос будет работать быстрее, чем генерация множества запросов на выборку товаров по одному [9].

После оптимизации работы серверной части следует уделить особое внимание компоновке данных на странице. Один и тот же контент будет иметь различные показатели скорости загрузки в зависимости от того, как он представлен на странице, а визуально не отличимые блоки могут иметь различную структуру html-тегов и, соответственно, отличающийся набор стилей и скриптов. Рассмотрим общие методы оптимальной компоновки данных.

Сокращение элементов dom дерева – заключается в уменьшении количества html-тегов, используемых для отображения страницы. Это позволит устройству быстрее просчитать и отобразить контент, а также даст меньшую нагрузку при работе js-кода. Если макет страницы уже был разработан, следует проанализировать его и избавиться от элементов, которые могут быть заменены с помочью css-стилей (например, псевдоэлемент after может заменить несколько узлов для стилизации переключателей спойлеров или слайдеров), а также элементов, не задействованных в стилизации контента (такие, как правило, остаются в макете после корректировки шаблона).

Сжатие статических ресурсов - оптимизация размера изображений, минификация стилей и скриптов. Это позволит сократить не только время, требуемое для анализа файлов, но и уменьшит нагрузку на сеть. Некоторые CMS позволяют использовать этот подход прямо из коробки. Например, структура компонентов для БУС (Битрикс управление сайтом) подталкивает разработчика к такой организации кода, а также дает возможность использовать общие ресурсы для нескольких компонентов.

Если файл стилей велик, имеет смысл использовать метод **прогрессивной загрузки стилей**, при этом основной файл стилей будет разделен на несколько частей так, чтобы перед каждым новым блоком подключались стили, относящиеся непосредственно к этому блоку. Если данный метод технически не выполним, можно применить альтернативный метод решения проблемы - использование **критических стилей**, встроенных в страницу. Это позволит уменьшить количество запросов к серверу, но будет увеличивать вес страницы. Какой из двух рассмотренных выше методов возможно приме-

нить и какой будет показывать наилучший результат определяется на конкретном проекте. Если целевая аудитория вашего ресурса в основном использует старые версии браузеров, не поддерживающие постепенную отрисовку страницы (например, Chrome меньше 69 версии), использование прогрессивной загрузки стилей приведет к увеличению времени отображения белого экрана (времени до того, как будет отрисован первый контент из-за ожидания загрузки большого числа отдельных файлов стилей, расположенных равномерно по странице), и в этом случае следует отдать предпочтение второму варианту [10].

Подключаемые файлы стилей следует проверить на наличие избыточных правил и правил, замедляющих отрисовку страницы, используя метод **очистки и оптимизации CSS-кода.** Следует убрать правила @import в файлах css и сделать подключение стилей отдельно через тег link в основном документе. Это позволит загружать стили параллельно. Если в стилях подключаются сторонние шрифты, нужно разрешить браузеру отрисовку текста до загрузки файла шрифта. Сам файл шрифта можно предварительно загрузить. Это ускорит отрисовку контента и позволит избежать создания цепочек критических запросов. Один файл стилей можно разбить на несколько по медиазапросам, создав отдельные таблицы стилей для каждой группы устройств (мобильные, планшеты, мониторы, принтеры (версия для печати)). Так можно сократить количество избыточных стилей и уменьшить нагрузку на сеть. [11]

Устранение избыточного кода. Для страницы должны загружаться только ресурсы, требующиеся для отображения контента на странице (а не общий пакет ресурсов для всего сайта). Нужно использовать минимально необходимый набор скриптов на страницах. Так, например, если шаблон использует несколько вариантов библиотек для реализации слайдера, имеет смысл выбрать только одну из них, отдав предпочтение той, которая будет минимально влиять на смещение макета до полного исполнения сценария, требует меньше зависимостей, менее требовательна к ресурсам устройства. Оставшиеся блоки можно переверстать с применением только выбранной библиотеки.

Протокол HTTP/2 и технология cache partitioning (технология, поддерживаемая многими браузерами, обеспечивающая изоляцию файлов кэша ресурса для каждого веб-ресурса) сводят на нет необходимость использования CDN серверов. Однако если от их использования технически невозможно отказаться или имеются ресурсы, которые обязательно следует загружать со стороннего сервера, можно применить метод **предварительного подключение к сторонним ресурсам**. Технически он реализуется добавлением тегов <link rel= "preconnect"> и <link rel= "dns-prefetch"> с адресом стороннего ресурса. Не следует злоупотреблять данными тегами, подключаясь к ресурсам, которые потенциально могут быть не запрошены (например, если сторонний ресурс находится в блоке с отложенной загрузкой, до которого пользователь может не дойти, перейдя на другую страницу). [12]

Каждый из рассмотренных методов влияет на метрики PageSpeed. В Табл. 1 приводится сравнение методов по степени влияния на метрики, где «+» означает значительное влияние, а «-» отсутствие влияния или опосредованное влияние (например, LCP всегда срабатывает после FPC, следовательно, все что влияет на FPC косвенно влияет и на LCP).

Существует еще ряд методов, позволяющих сократить время ожидания загрузки страницы. Их

Табл. 1. Сравнение методов оптимизации компоновки контента html по влиянию на основные метрики PageSpeed (на базе Lighthouse 10 версии)

	FPC	**SI**	**TBT**	**LCP**	**CLS**
Сокращение элементов dom дерева	-	-	+	-	-
Сжатие статических ресурсов	+	-	-	+	-
Прогрессивная загрузки стилей	+	-	-	-	+
Использование критических стилей	+	-	-	-	+
Очистка и оптимизация CSS кода	+	-	-	+	-
Устранение избыточного кода	+	-	-	+	-
Предварительное подключение к сторонним ресурсам	+	-	-	-	-

можно разделить на две условные группы. Первая группа методов использует специальную организацию разметки и требует написания своих вспомогательных блоков кода для обеспечения отложенной загрузки «тяжелого» контента (высококачественных изображений большого веса и размера, сложных виджетов, требующих подзагрузки стилей и скриптов). Так как современные сайты — это технически сложные системы, как правило, работающие на основе систем управления контентом или специализированных фреймворках, то применение этих методов требует автоматизации. В случае с первой группой методов будет полезно вынести специально подготовленное решение с правильной разметкой контента в отдельные переиспользуемые контейнеры (их название и вид будет различным в зависимости от того, на чем работает конкретный ресурс, например, в modx роль такого контейнера могут выполнять чанки и снипеты).

Для применения методов из второй группы можно использовать заранее подготовленные скрипты для автоматического приведения структуры документа к такому виду, чтобы документ не имел ресурсов, блокирующих отрисовку страницы. Использование таких скриптов становится особенно актуальным при невозможности изначально подготовить страницу в соответствии с желаемой структурой из-за технических ограничений движка сайта и/или из-за необходимости использования не оптимизированных сторонних решений. Так, например, не все плагины для CMS WordPress используют api системы для добавления ресурсов на страницу (вроде функции wp_enqueue_style для регистрации и подключения стилей), что не позволяет их автоматически обрабатывать плагинам, занимающимся оптимизацией контента.

3. Разметка контента, замедляющего рендеринг страницы

Для быстрой отрисовки страницы браузеру важно как можно скорее отобразить все ее составляющие части. Если одна из частей страницы будет требовать загрузки файла большого объема или повлечет дозагрузку дополнительных ресурсов, отрисовка страницы будет приостановлена. Чтобы избежать этого поведения, нужно использовать ряд методов, обеспечивающих оптимальную загрузку контента.

Самая часто встречающаяся задача — это размещение на сайте изображений. При этом файлы изображений часто могут иметь большой вес, могут генерироваться для конкретного пользователя и иметь долгий ответ сервера, могут находиться на других серверах, а также быть самым крупным объектом на первом экране. Это делает изображения одним из главных элементов, оптимизацией которого следует заняться в любом проекте. Для реализации такой оптимизации существует множество отдельных модулей для разных систем, но их совмещение и применение уникальны для каждого проекта.

Прежде всего, следует выбрать подходящий формат для изображения. Предпочтительно выбрать тот, который, сохраняя максимальную близость к оригинальному изображению, даст наименьший вес файла. Обычно хорошим выбором для статичных растровых изображений являются современные форматы WebP и AVIF. Для небольших векторных изображений исходного формата svg можно рассмотреть возможность встроить их прямо в тело страницы. Анимированные изображения большого размера можно конвертировать в видео формат MPEG-4/WebM. Вне зависимости от того какой формат был выбран, изображение нужно сжать одним из современных способов (например, через онлайн сервис вроде squoosh.app).

Несмотря на то, что современные форматы поддерживаются почти всеми широко распространенными браузерами (в последних версиях) [13, 14], нужно ориентироваться на программы, используемые целевой аудиторией конкретного ресурса. Если их программы не могут отображать эти форматы, следует дублировать изображения в более распространенных форматах jpeg (для изображений с непрозрачным фоном) и png (для изображений с прозрачным фоном).

Нужно понимать, что изображение может отображаться на устройствах разного размера. Будет нецелесообразно использовать для мобильных устройств изображения размером в несколько раз превосходящие самые крупные дисплеи. Для того, чтобы предоставить пользователям изображение, подходящие по размерам под используемое ими устройство, следует генерировать несколько вариантов

изображения разного размера и подключать их через тег source с атрибутами sizes и srcset. Для того чтобы изображение не ломало макет до своей полной загрузки, следует указывать также атрибуты height и width, что позволит браузеру правильно вписать изображение в страницу.

Если изображение не находится на первом экране, его загрузку можно отложить, как и его расшифровку. Для этого используются атрибуты loading="lazy" и decoding="async". Чтобы убрать пустоту на месте еще не загруженного изображения, на случай если пользователь перешел к области его отображения до того, как браузер успел загрузить изображение, можно использовать размытую заглушку. Обычно это маленький svg файл, встроенный в html документ, дающий общее представление о картинке, которая будет загружена позже. [15]

Кроме изображений значительное влияние на скорость загрузки страницы могут оказывать виджеты со сторонних ресурсов, такие как яндекс карты, чат Jivo или виде из YouTube. Для оптимизации загрузки таких ресурсов можно применять два метода. Первый - для блоков, которые должны быть встроены в контент страницы (например, видео в тексте статьи). Для них используются фасады. Это статические элементы, которые похожи на реальные встроенные элементы с других ресурсов, но при этом не обладающие функциональностью реальных элементов и, как следствие, не замедляющие загрузку страницы. При наведении курсора на такой элемент происходит предварительное подключение к сторонним ресурсам, а по клику непосредственно загрузка стороннего ресурса. [16]

Второй метод применяется для ресурсов, которые должны быть отображены не сразу и закреплены на определенном месте на экране (например, в правом нижнем углу) или при необходимости загрузить встроенный в контент элемент до того, как пользователь его увидит. Он заключается в начале загрузки ресурсов после первого взаимодействия пользователя с сайтом или при приближении области просмотра к заданному элементу на определенное расстояние. Вариант реализации алгоритма отложенной загрузки при помощи JS кода, предложенный авторами, приведен на Рис. 2.

Принцип работы скрипта заключается в том, что создается и тут же выполняется функция,

```
<script type="text/javascript">
  (function deferredCode()
  {
    if (typeof window.deferredCodePrepared == "undefined")
    {
      window.deferredCodePrepared = true;
      window.addEventListener('scroll',     deferredCode, {passive: true});
      window.addEventListener('touchstart', deferredCode);
      window.addEventListener('click',      deferredCode);
      window.addEventListener('keydown',    deferredCode);
      window.deferredCodeTimer = setTimeout(deferredCode ,10000);
    }
    else
    {
      window.removeEventListener('scroll',     deferredCode, false);
      window.removeEventListener('touchstart', deferredCode, false);
      window.removeEventListener('click',      deferredCode, false);
      window.removeEventListener('keydown',    deferredCode, false);
      clearTimeout(window.deferredCodeTimer);

      let yandexRating = document.getElementById('js--xxxxxxxxxx');
      if (yandexRating != null)
      {
          yandexRating.src = "https://yandex.ru/sprav/widget/rating-badge/xxxxxxxxxx";
      };
    }
  })();
</script>
```

Рис. 2. Отложенная загрузка сторонних ресурсов (при взаимодействии со страницей) – алгоритм на языке программирования JavaScript

которая проверяет была ли она выполнена ранее, в зависимости от чего выполняет один из двух блоков кода. Если ранее функция не выполнялась, то она назначается как функция-обработчик для событий взаимодействия пользователя со страницей (скрол, прикосновение, клик, нажатие клавиши), а также назначается на таймер, срабатывающий через некоторое заданное время. Второе срабатывание функции возможно только при наступлении одного из событий, на которое она была назначена. При этом, функция отвязывается от всех событий, таймер отменяется, и выполняется некоторый код, исполнение которого и требовалось отложить. В примере выполняется установка значения атрибута src в iframe для загрузки виджета со стороннего сайта.

4. Алгоритмы автоматической перекомпоновки контента

Как уже было упомянуто ранее, для того чтобы обеспечить быструю отрисовку контента на странице, необходимо устранить ресурсы, блокирующие эту отрисовку. Наиболее распространенные ресурсы, блокирующие отрисовку, js и css файлы, подключенные в области тега head. Для того чтобы устранить блокировку, достаточно перенести загрузку этих ресурсов после основных блоков с контентом страницы. Для некоторых скриптов также допустима отложенная загрузка через указание атрибутов defer и async в теге script. [17] В отличие от контента, который формируется в теле страницы, изменить контент в разделе head бывает невозможно вручную, не внося изменения в сторонние модули сайта. Однако такое решение задачи будет ненадежным, так как после каждого обновления стороннего решения правки придется вносить повторно.

В случае если нет возможности повлиять на расположение ресурсов на странице непосредственно в момент ее формирования, можно изменить расположение ресурсов перед отправкой контента пользователю. Для этого используется буфер, который накапливает вывод html документа и модифицируется до отправки данных в браузер. Такой код может быть повторно использован в различных системах управления контентом.

Для решения типовых задач оптимизации скорости загрузки веб страницы на языке программирования PHP авторами предложены алгоритмы, меняющие компоновку html документа. Рассмотрим разработанные алгоритмы, увеличивающие показатель First Contentful Paint. Первый позволит устранить блокировку рендеринга скриптами. Пример реализации этого алгоритма представлен на Рис. 3. Блок-схема на Рис. 4 объясняет принцип его работы.

```php
<?php
$html = getHtml();
$marker = '<!-- scripts -->';
if (strpos($html, $marker))
{
  if (preg_match('/<head([^>]*)>(.*)<\/head>/s', $html, $match))
  {
    $head = $match[2];
    $scripts = '';
    $regExp = '/<script([^>]*)src=([^>]+)><\/script>/';
    if (preg_match_all($regExp, $head, $matches, PREG_SET_ORDER)>0)
    {
      foreach ($matches as $item)
      {
        $html = str_replace($item[0], '', $html);
        $scripts .= $item[0];
      }
    }
    $html = str_replace($marker, $scripts, $html);
  }
}
setHtml($html);
?>
```

Рис. 3. Автоматический перенос скриптов из раздела head – алгоритм на языке программирования php

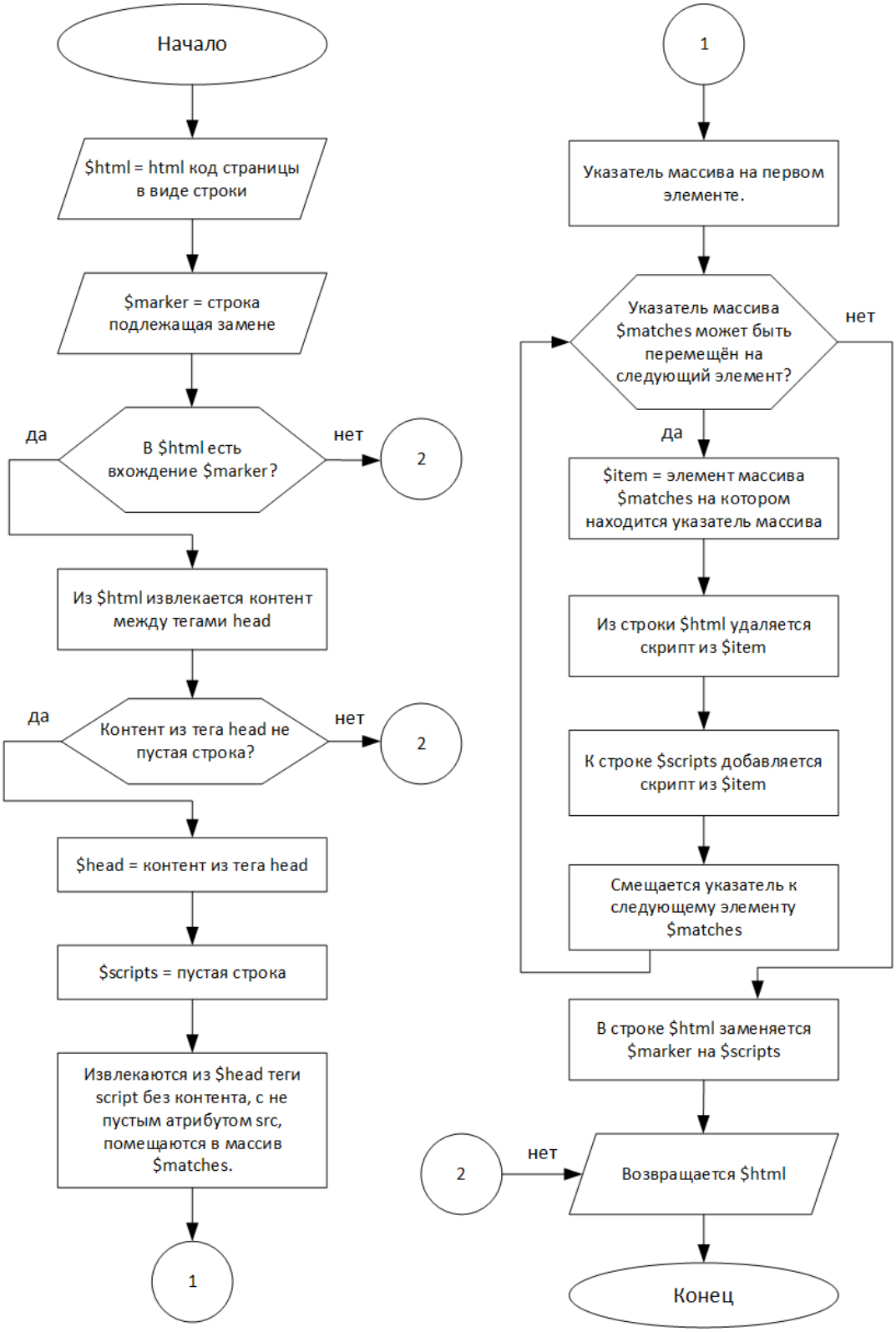

Рис. 4. Автоматический перенос скриптов из раздела head – блок схема алгоритма

Представленный алгоритм позволяет извлечь из области тега head теги со скриптами и перенести их в заданное место в html документе, указанное с помощью определенной строки. Перенос тегов может быть заменен на добавление атрибутов асинхронной или параллельной загрузки.

Второй алгоритм позволит устранить блокировку рендеринга страницы внешними стилями. Пример реализации этого алгоритма представлен на Рис. 5. Блок схема на Рис. 6 объясняет принцип его работы.

Представленный алгоритм позволяет перенести все стили из области тега head в область тега body, добавить в область тега head временные критические стили, которые будут заменены оригинальными стилями сайта после их загрузки в конце обработки страницы. Временные критически стили необходимы для того, чтобы устранить большое смещение макета в начале загрузки страницы. JavaScript-код, добавляемый этим алгоритмом, выполняется после построения dom-дерева. Его функция сводится к тому, чтобы раз в секунду проверять сколько стилей осталось незагруженными, и когда их не останется – удалить тег с критическими стилями.

```php
<?php
$html = getHtml();
$lazyStyles = '';

$regExp = '/<link([^>]*)rel=["\']stylesheet["\']([^>]*)>/';
if (preg_match_all($regExp, $html, $styles, PREG_SET_ORDER))
{
  foreach ($styles as $style)
  {
    $html = str_replace($style[0], '', $html);
    $lazyStyles .= str_replace(
      '<link',
      '<link onload="window.lazyStylesQuantity--;" ',
      $style[0]
    );
  }
}

$lazyStyles =
'<script type="text/javascript">
    window.lazyStylesQuantity = '.count($styles).';
    document.addEventListener("DOMContentLoaded", function() {
      var lazyStylesInterval = setInterval(function(){
        if (window.lazyStylesQuantity==0) {
          clearInterval(lazyStylesInterval);
          var criticalCss = document.getElementById("js--critical-css");
          criticalCss.parentNode.removeChild(criticalCss);
        };
      }, 1000);
    });
</script>'.$lazyStyles;

$criticalCss  = '<style id="js--critical-css" type="text/css">';
$criticalCss .= file_get_contents('style.css');
$criticalCss .= '</style></head>';

$html = str_replace('</body>', $lazyStyles.'</body>', $html);
$html = str_replace('</head>', $criticalCss, $html);

setHtml($html);
?>
```

Рис. 5. Автоматический перенос стилей из head в body, подключение временных критических стилей – алгоритм на языке программирования php и JavaScript

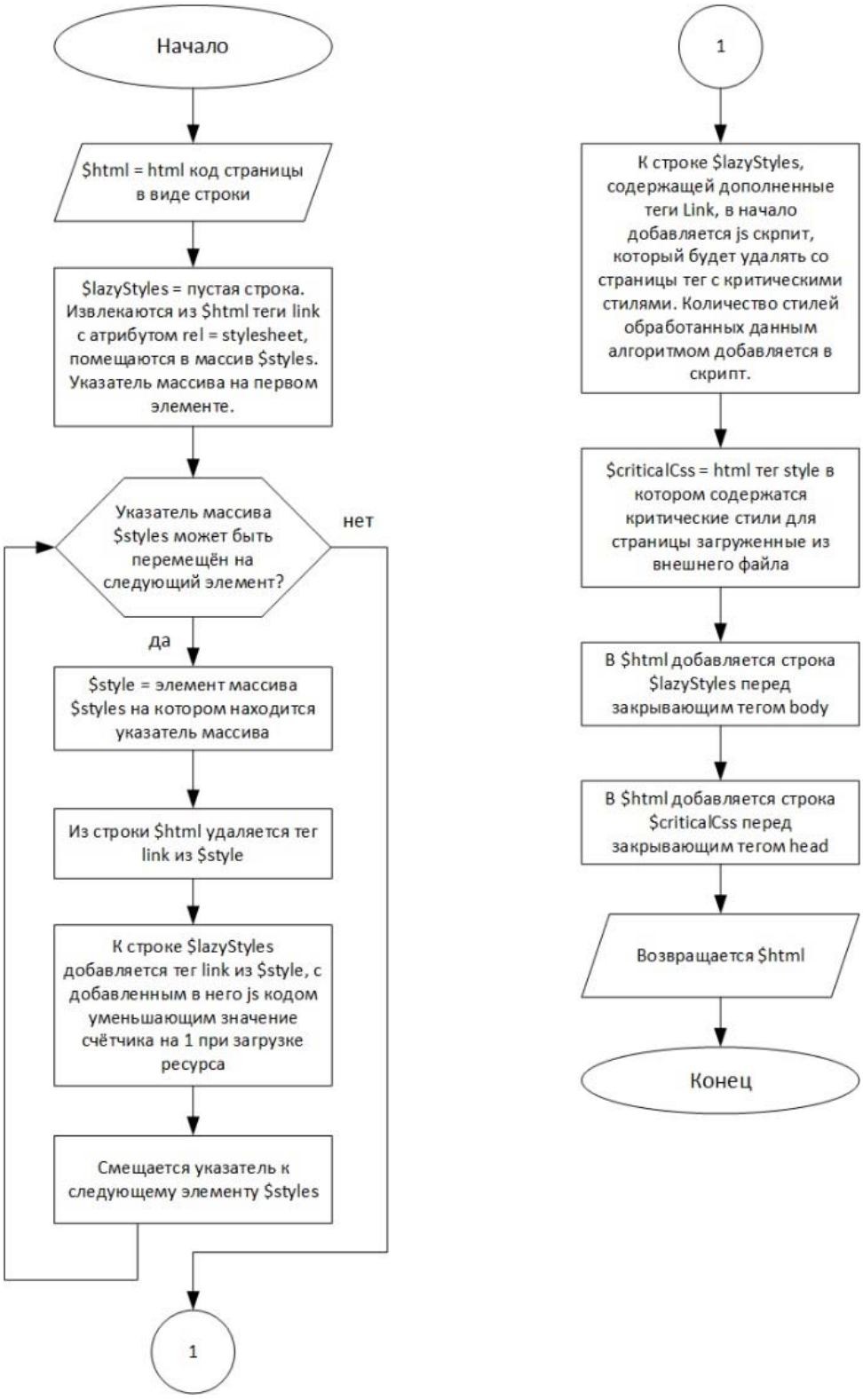

Рис. 6. Автоматический перенос стилей из head в body, подключение временных критических стилей – блок схема алгоритма

Табл. 2. Сравнение влияния разработанных алгоритмов и известных методов на скорость загрузки главной страницы сайта murkosha.ru

	Номер теста	**Платформа**	**FPC**	**SI**	**TBT**	**LCP**	**CLS**	Итоговый балл
До оптимизации	1	Mobile	9,1	11,9	1 440	21,6	0.017	30
		PC	3,2	3,5	160	6,8	0.154	47
	2	Mobile	9,1	11,8	870	22,8	0.406	16
		PC	3,2	3,4	220	6,8	0.153	44
	3	Mobile	9,1	11,3	1 230	22,3	0.017	32
		PC	3,2	3,4	290	6,9	0.152	40
После применения разработанных алгоритмов	1	Mobile	1,0	4,9	2 760	14,3	0	42
		PC	0,3	1,5	170	4,6	0.095	70
	2	Mobile	1,0	5,2	3 090	13,4	0.021	42
		PC	0,3	1,5	310	3,9	0.095	62
	3	Mobile	1,0	4,6	2 510	12,1	0.021	43
		PC	0,3	1,5	290	4,6	0.095	62
После применения известных методов	1	Mobile	1,0	5,0	2 390	8,4	0.021	43
		PC	0,3	1,5	220	3,0	0.095	72
	2	Mobile	1,0	4,1	3 050	8,6	0	44
		PC	0,3	1,6	290	2,9	0.095	68
	3	Mobile	1,0	4,3	1 490	9,8	0.021	47
		PC	0,3	1,4	380	2,7	0.095	65

Эти алгоритмы позволяют решить проблему с ресурсами, блокирующими рендеринг страницы, увеличивают показатель First Contentful Paint в PageSpeed. Влияние предложенных алгоритмов на метрики PageSpeed оценивалось экспериментально посредством применения алгоритмов на страницу существующего веб-сайта. В качестве тестовой страницы была взята главная страница сайта murkosha.ru. Контент и статические ресурсы были перемещены на тестовый сервер. Были сделаны замеры скорости веб страницы до применения алгоритмов автоматической перекомпоновки данных. Анализ страницы показал, что метрики FPC и LCP сильно завышены. Затем, к странице был применен алгоритм автоматической компоновки данных, после чего внесены правки по методам, изложенным в главах 2 и 3. Результаты замеров получившихся страниц представлены в Табл. 2.

После автоматической компоновки данных метрики FPC и LCP значительно уменьшились, что существенно повлияло на оценку в лучшую сторону (прирост около 10 баллов). После ручной корректировки контента по методам, рассмотренным в главах 2 и 3, прирост скорости был невелик (около 2 баллов по сравнению с результатами автоматической компоновки). Это связанно с тем, что метрика FPC для мобильных устройств уже достигла минимально возможного значения, а на десктопе была очень близка к минимально возможному значению. Следовательно, LCP получала прирост по большей части из-за правок непосредственно на нее влияющих, и снижение этого показателя было невелико.

Таким образом, разработанные алгоритмы автоматической компоновки данных показали свою эффективность при оптимизации страницы. Комплексное применение всех методов оптимизации позволяет добиться более высоких результатов, чем использование исключительно методов автоматической компоновки данных.

Заключение

На сегодняшний день существует немало инструментов, позволяющих измерить скорость загрузки сайта. Один из наиболее популярных сервисов PageSpeed компании Google. Для оценки скорости загрузки сайта используются ряд метрик, определяемых сервисом. Так, упомянутый ранее PageSpeed на базе Lighthouse 10 измеряет: First Contentful Paint (время первой отрисовки контента), Speed Index (скорость загрузки контента), Total Blocking Time (общие время блокировки взаимодействия со страницей), Largest Contentful Paint (время отрисовки самого крупного блока), Cumulative Layout Shift (общее смещение макета).

На основании этих метрик выставляется синтетическая оценка скорости загрузки страницы. Для разработчиков и владельцев сайтов важно поддерживать эту оценку на высоком уровне, так как низкая оценка служит маркером проблем в работе сайта и понижает привлекательность ресурса в глазах пользователей и поисковых систем.

Авторами проанализированы методы, позволяющие поддерживать скорость загрузки страницы на высоком уровне, и предложены рекомендации компоновки веб-контента:

- использовать минимально необходимое количество dom-узлов в документе;
- подключать минифицированные версии стилей и скриптов;
- стили не должны создавать цепочки критических запросов и блокировать рендеринг страницы до загрузки шрифтов;
- файл стилей желательно разбивать на несколько по медиа-запросам;
- нужно подключать только необходимые скрипты;
- если какой-либо ресурс будет гарантированно загружен со стороннего сервиса, нужно предварительно подключится к этому сервису;
- картинки следует предоставлять в современных форматах, в размерах под разные разрешения мониторов, с разрешением отложенной загрузки и рендеринга для тех, которые находятся за пределами первого экрана;
- загрузку данных со сторонних сервисов можно отложить до первого взаимодействия со страницей или с элементом, а также до приближения зоны просмотра к элементу, требующему данные со сторонних ресурсов.

Кроме рекомендаций, при составлении контента, авторами предложены алгоритмы, которые будут автоматически менять компоновку страницы, увеличивая скорость ее отрисовки. Алгоритмы позволяют решить наиболее распространенную при оптимизации страницы задачу – устранение блокировки рендеринга стилями и скриптами.

Литература

1. Как скорость и доступность сайта влияют на seo. URL: https:/айри.рф/blog/как-скорость-доступность-влияют-seo/ (дата обращения: 28.02.2023).
2. Ситников А. С. 12 сервисов проверки скорости загрузки сайта. URL: https://site-ok.ua/blog/12-servisov-proverki-skorosti-zagruzki-sayta (дата обращения: 08.05.2023).
3. Walton P. Первая отрисовка контента (FCP). URL: https://web.dev/fcp/ (дата обращения: 28.02.2023).
4. Speed Index. URL: https://developer.chrome.com/docs/lighthouse/performance/speed-index/ (дата обращения: 28.02.2023).
5. Walton P. Общее время блокировки (TBT). URL: https://web.dev/tbt/ (дата обращения: 28.02.2023).
6. Walton P. Скорость загрузки основного контента (LCP). URL: https://web.dev/lcp/ (дата обращения: 28.02.2023).
7. Walton P., Mihajlija M. Совокупное смещение макета (CLS). URL: https://web.dev/cls/ (дата обращения: 28.02.2023).
8. Хостинг для продуктов «1С-Битрикс»». URL: https://www.1c-bitrix.ru/products/cms/hosting.php (дата обращения: 08.05.2023).
9. Оптимизация запросов к БД. URL: https://dev.1c-bitrix.ru/learning/course/index.php?COURSE_ID=32&LESSON_ID=3594 (дата обращения: 08.05.2023).
10. CSS and Network Performance. URL: https://csswizardry.com/2018/11/css-and-network-performance/ (дата обращения: 08.05.2023).
11. Мыльников К. Улучшаем производительность сайта с помощью CSS. URL: https://habr.com/ru/company/usetech/blog/718200/ (дата обращения: 28.02.2023).
12. Akulov I. Preload, prefetch and other <link> tags. URL: https://3perf.com/blog/link-rels/ (дата обращения: 28.02.2023).
13. WebP image format. URL: https://caniuse.com/webp (дата обращения: 28.02.2023).
14. AVIF image format. URL: https://caniuse.com/avif (дата обращения: 28.02.2023).
15. Maximally optimizing image loading for the web. URL: https://www.industrialempathy.com/posts/image-optimizations/ (дата обращения: 28.02.2023).
16. Lazy load third-party resources with facades. URL: https://developer.chrome.com/docs/lighthouse/performance/third-party-facades/ (дата обращения: 28.02.2023).
17. Скрипты: async, defer. URL: https://learn.javascript.ru/script-async-defer (дата обращения: 28.02.2023).

Жуков Станислав Владимирович. Тамбовский государственный университет имени Г.Р. Державина, Тамбов, Россия. Аспирант. Область научных интересов: информационные технологии, математическое и компьютерное моделирование динамических систем. E-mail: i@coder-stas.ru

Ковалева Ольга Александровна Тамбовский государственный университет имени Г.Р. Державина, Тамбов, Россия. Профессор кафедры математического моделирования и информационных технологий, доктор технических наук, доцент. Область научных интересов: информационные технологии, математическое и компьютерное моделирование динамических систем. E-mail: solomina-oa@yandex.ru

Ковалев Сергей Владимирович Тамбовский государственный университет имени Г.Р. Державина, Тамбов, Россия. Профессор кафедры математического моделирования и информационных технологий, доктор технических наук, доцент. Область научных интересов: информационные технологии, математическое и компьютерное моделирование динамических систем. E-mail: sseedd@mail.ru

Analysis and Development of Algorithms that Optimize the Layout of the html Document for Browsers

S. V. Zhukov, O. A. Kovalev, S. V. Kovalev

Derzhavin Tambov State University, Tambov, Russia

Abstract. The article provides an overview of the metrics of the Google PageSpeed Insights service (based on Lighthouse 10 version) that made it possible to determine the speed of loading a web page. Based on the above metrics, methods for marking up html content are collected and analyzed. The complex application of the above methods, as well as the correct configuration of the server that can quickly send the generated code, will allow you to get a high estimate of the page loading speed based on the previously discussed metrics. The author's implementation of algorithms that can automatically change the html document to match the scheme of optimal data layout for fast display of page content is given.
Keywords: html, content layout, web page, optimization, loading speed.

DOI 10.14357/20718632230312

References

1. Kak skorost' i dostupnost' sayta vliyayut na seo [How site speed and accessibility affect SEO]. Available at: https:/айри.рф/blog/как-скорость-доступность-влияют-seo/ (accessed February 28, 2023).
2. Sitnikov A.S. 12 servisov proverki skorosti zagruzki sayta [12 services for checking website loading speed]. URL: https://site-ok.ua/blog/12-сервисов-проверки-скорости-загрузки-сайта (accessed May 8, 2023).
3. Walton P. Pervaya otrisovka kontenta (FCP) [First Content Render (FCP)]. Available at: https://web.dev/fcp/ (accessed February 28, 2023).
4. Speed Index. Available at: https://developer.chrome.com/docs/lighthouse/performance/speed-index/ (дата обращения: 28.02.2023).
5. Walton P. Obshcheye vremya blokirovki (TBT) [Total Block Time (TBT)]. Available at: https://web.dev/tbt/ (accessed February 28, 2023).
6. Walton P. Skorost' zagruzki osnovnogo kontenta (LCP) [Main content download speed (LCP)]. Available at: https://web.dev/lcp/ (accessed February 28, 2023).
7. Walton P., Mihajlija M. Sovokupnoye smeshcheniye maketa (CLS) [Cumulative Layout Offset (CLS)]. Available at: https://web.dev/cls/ (accessed February 28, 2023).
8. Khosting dlya produktov «1S-Bitriks» [Hosting for 1C-Bitrix products]. URL: https://www.1c-bitrix.ru/products/cms/hosting.php (accessed May 8, 2023).
9. Database query optimization [Database query optimization]. URL: https://dev.1c-bitrix.ru/learning/course/index.php?COURSE_ID=32&LESSON_ID=3594 (accessed May 8, 2023).
10. CSS and Network Performance. URL: https://csswizardry.com/2018/11/css-and-network-performance/ (accessed May 8, 2023).
11. Myl'nikov K. Uluchshayem proizvoditel'nost' sayta s pomoshch'yu CSS [Improving website performance with CSS]. Available at: https://habr.com/ru/company/usetech/blog/718200/ (accessed February 28, 2023).
12. Akulov I. Preload, prefetch and other <link> tags. Available at: https://3perf.com/blog/link-rels/ (accessed February 28, 2023).
13. WebP image format. Available at: https://caniuse.com/webp (accessed February 28, 2023).
14. AVIF image format. Available at: https://caniuse.com/avif (accessed February 28, 2023).
15. Maximally optimizing image loading for the web. Available at: https://www.industrialempathy.com/posts/image-optimizations/ (accessed February 28, 2023).
16. Lazy load third-party resources with facades. Available at: https://developer.chrome.com/docs/lighthouse/performance/third-party-facades/ (accessed February 28, 2023).
17. Skripty: async, defer [Scripts: async, defer]. Available at: https://learn.javascript.ru/script-async-defer (accessed February 28, 2023).

Zhukov Stanislav V. PhD student, Derzhavin Tambov State University, 33, Internatsionalnaya St., 392000, Tambov, Russia. E-mail: i@coder-stas.ru

Kovaleva Olga A. Dr. Sc., Tech., Professor, Derzhavin Tambov State University, 33, Internatsionalnaya St., 392000, Tambov, Russia. E -mail: solomina-oa@yandex.ru

Kovalev Sergey V. Dr. Sc., Tech., Professor, Derzhavin Tambov State University, 33, Internatsionalnaya St., 392000, Tambov, Russia. E -mail: sseedd@mail.ru