МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение

высшего образования

«Санкт-Петербургский национальный исследовательский университет

информационных технологий, механики и оптики»

**Факультет программной инженерии и компьютерной техники**

**Направление (специальность) ⎯ 09.04.02 Информационные системы и технологии**

**Специализация ⎯ Веб-технологии**

**Дисциплина — Проектирование и анализ языков веб-решений**

**Курсовой проект (работа)**

**ТЕМА: Исследование оптимизации веб-ресурсов с использованием клиентских и серверных веб-технологий**

ВЫПОЛНИЛ

Студент группы P41622 \_\_\_\_\_\_\_\_\_\_\_\_ Харионовская П.С.

№  группы подпись, дата ФИО

ПРОВЕРИЛ     \_\_\_\_\_\_\_\_\_\_\_\_ Государев И.Б.

ученая степень, должность подпись, дата ФИО

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 3](#_Toc10201486)

[1 ТЕНДЕНЦИЯ ИСПОЛЬЗОВАНИЯ JAVASCRIPT В СЕТИ 4](#_Toc10201487)

[2 СЕРВЕРНЫЙ И КЛИЕНТСКИЙ РЕНДЕРИНГ 6](#_Toc10201488)

[3 СРАВНЕНИЕ SSR И CSR 8](#_Toc10201489)

[4 КАК GOOGLE ОБРАБАТЫВАЕТ РЕНДЕРИНГ JAVASCRIPT 10](#_Toc10201490)

[5 ВЛИЯНИЕ SSR И CSR НА SEO 13](#_Toc10201491)

[ЗАКЛЮЧЕНИЕ 15](#_Toc10201492)

[СПИСОК ЛИТЕРАТУРЫ 17](#_Toc10201493)

# Введение

Цель работы — Исследовать оптимизацию веб-ресурсов с использованием клиентских и серверных веб-технологий.

Для выполнения обозначенной цели были поставлены следующие задачи:

1. Исследование тенденция использования JavaScript в сети;
2. Исследования понятий и принципов работы серверного и клиентского рендеринга;
3. Сравнение серверного и клиентского рендеринга, выделение преимуществ и недостатков;
4. Исследование Google технологии рендеринга Java Script;
5. Исследование влияния SSR и CSR для SEO.

# 1 тенденция Использования javascript в сети

С ростом популярности фреймворков JavaScript, таких как Angular, React и Vue.js, разработчики могут создавать более эффективные веб-страницы, а также обеспечивать быструю загрузку пользовательских интерфейсов. Однако, если не спланировать это тщательно, этот скачок в эффективности может привести к существенной цене за SEO.

В прошлом JS преимущественно использовался для добавления различных уровней взаимодействия к веб-странице. Это было достигнуто путем использования библиотек JS, таких как JQuery. Поскольку JS просто манипулировал содержимым HTML, уже присутствующим в исходном коде, возникла небольшая проблема с поисковыми системами, обнаружившими и проиндексировавшими содержимое. Тем не менее, с некоторыми из JS-структур и клиентским рендерингом сегодня исходный код веб-страницы становится практически пустым, а контент обрабатывается исключительно на стороне клиента.

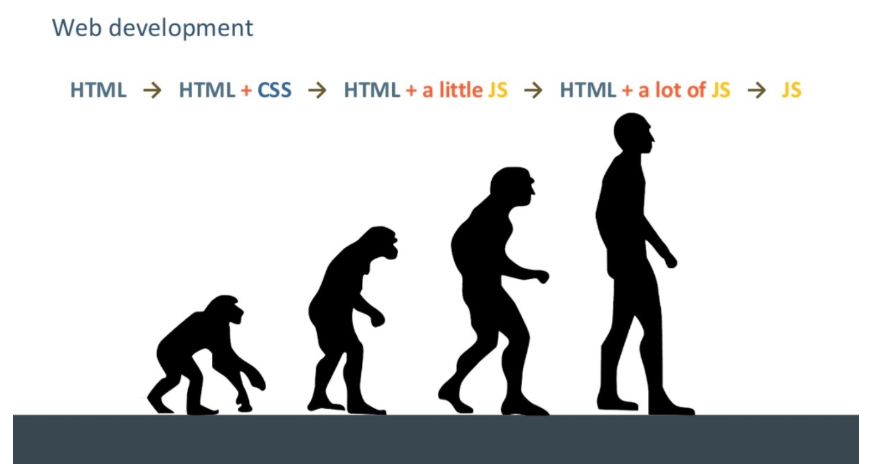


Рисунок 1 - Развитие использования JavaScript в сети [1]

Это усложняет работу поисковых систем. Сегодня Google является единственной поисковой системой, которая в некоторой степени справляется с этим. Ни одна из других поисковых систем не обладает возможностями рендеринга JS, которые сейчас используются Google. Это означает, что даже если ваш контент, использующий рендеринг на стороне клиента, может быть проиндексирован Google, он, вероятно, не будет проиндексирован другими поисковыми системами [1].

# 2 серверный и клиентский рендеринг

Рендеринг на стороне сервера (SSR)  - традиционный метод рендеринга, при котором ресурсы веб страницы размещаются на сервере. При запросе страницы, HTML-код доставляется в браузер и обрабатывается, загружается JS и CSS, и конечный рендер отображается пользователю или боту.

В традиционном подходе браузер или робот Googlebot получает HTML-код, полностью описывающий страницу. Копия контента уже есть - браузер или робот Google должен загрузить CSS и «нарисовать» контент на экране. Обычно у поисковых систем не возникает проблем с отображаемым на сервере контентом.

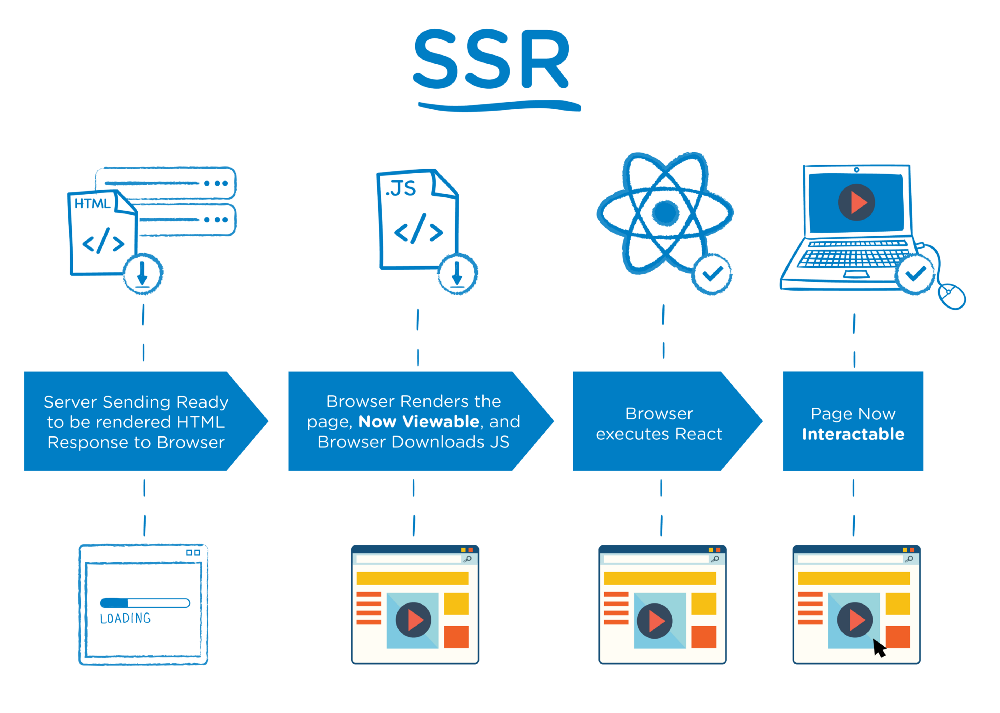


Рисунок 2 – Принцип реализации серверного рендеринга [2]

Рендеринг на стороне клиента (CSR) - новый метод рендеринга, основанный на JS, выполняемом на стороне клиента (в браузере) через инфраструктуру JavaScript. Фактически, на такой странице либо вовсе нет данных, подходящих для анализа и индексации, либо их очень мало. Затем в дело вступают JavaScript-механизмы, которые асинхронно загружают данные с сервера и обновляют страницу (изменяя DOM) [1].

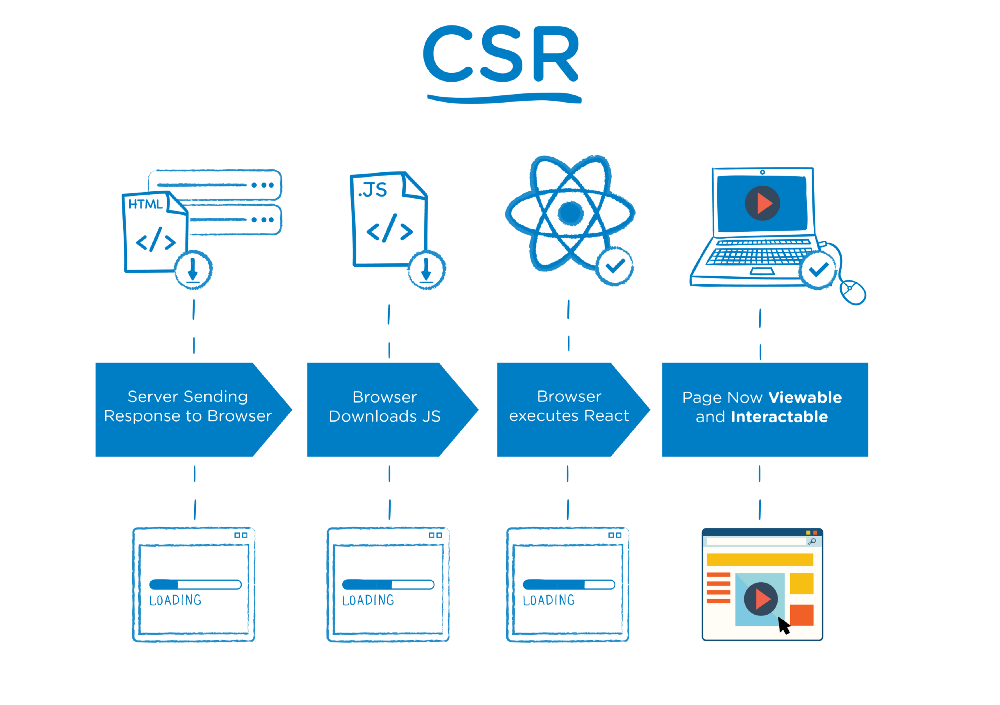


Рисунок 3 – Принцип реализации клиентского рендеринга [2]

# 3 СРАВНЕНИЕ SSR и CSR

Рендеринг на стороне сервера является наиболее распространенным способом отображения информации на экране. Он работает путем преобразования файлов HTML на сервере в полезную информацию для браузера.

При каждом посещении веб-сайта браузер отправляет запрос на сервер, который содержит содержимое веб-сайта. Запрос обычно занимает всего несколько миллисекунд, но зависит от множества факторов:

* Скорость интернета пользователя, делающего запрос;
* Сколько активных пользователей получают доступ к сайту в данный момент;
* Физическое расположение сервера;
* Насколько оптимизированы страницы для скорости;
* И др.

После завершения обработки запроса браузер возвращает полностью отображенный HTML-код и отображает его на экране. Если затем возникает необходимость посетить другую страницу на веб-сайте, браузер еще раз сделает новый запрос информации. Так будет происходить каждый раз, когда пользователь посещает страницу, на которой у браузера нет кэшированной версии. Не имеет значения, если на новой странице есть только несколько элементов, которые отличаются от текущей страницы, браузер запросит всю новую страницу и будет перерисовывать все с начала [3].

Рендеринг на стороне клиента, с другой стороны, медленнее при первоначальном запросе, потому что он совершает несколько обращений к серверу. Тем не менее, как только эти запросы завершены, CSR выполняет быструю работу с JS-фреймворком.

Метафора Адама Зернера, описывающая разницу между SSR и CSR: «With server-side rendering, whenever you want to see a new web page, you have to go out and get it, this is analogous to you driving over to the super market every time you want to eat. With client-side rendering, you go to the super market once and spend 45 minutes walking around buying a bunch of food for the month. Then, whenever you want to eat, you just open the fridge» [4].

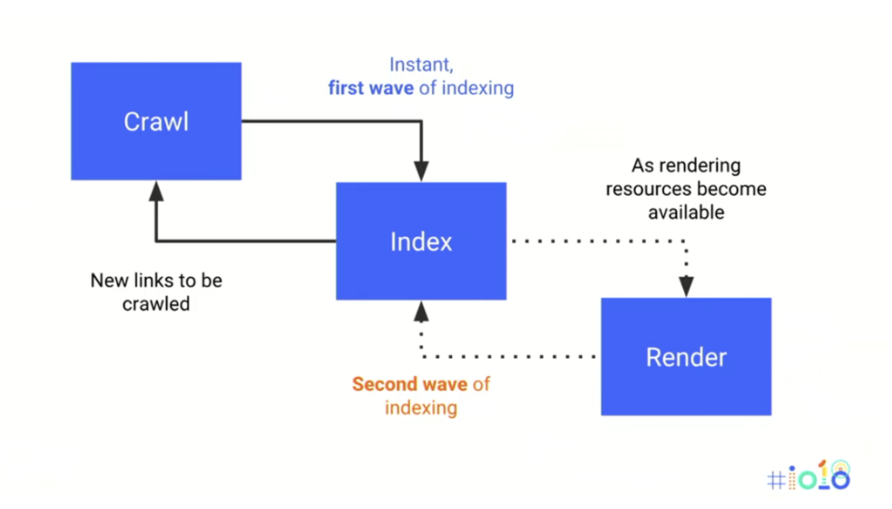
При сравнении серверного и клиентского видов рендеринга были выделены преимущества и недостатки, представленные в таблице1.

Таблица 1 – Преимущества и недостатки двух видов рендеринга

|  |  |  |
| --- | --- | --- |
|  | Плюсы | Минусы |
| Серверный рендеринг | 1. Поисковые системы могут сканировать сайт для лучшего SEO 2. Начальная загрузка страницы происходит быстрее 3. Отлично подходит для статических сайтов | 1. Частые запросы к серверу; 2. Общий медленный рендеринг страницы; 3. Полная перезагрузка страницы; 4. Низкое взаимодействие с сайтом. |
| Клиентский рендеринг | 1. Богатое взаимодействие с сайтом; 2. Быстрый рендеринг сайта после начальной загрузки; 3. Отлично подходит для веб-приложений; 4. Надежный выбор библиотек JavaScript. | 1. Низкий SEO, если не реализован правильно; 2. Начальная загрузка может потребовать больше времени; 3. В большинстве случаев требуется внешняя библиотека. |

# 4 Как Google обрабатывает рендеринг JavaScript

Google использует двухволновой процесс рендеринга и индексации JS.



Google двухволновой процесс рендеринга JavaScript [5]

Первая волна запрашивает исходный код, сканирует и индексирует любой существующий HTML и CSS, добавляет любые существующие ссылки в очередь сканирования и загружает коды ответа страницы. На отрисованном сайте на стороне клиента Google практически ничего не индексирует в исходном коде во время первой волны.

Вторая волна может произойти в течение нескольких часов или даже несколько недель спустя, Googlebot возвращается на страницу, когда доступны дополнительные ресурсы для полного рендеринга и индексирования JS контента.

Это означает, что там, где, как и в SSR, обычно нет проблемы с задержкой рендеринга Google, поскольку весь контент находится в исходном коде и индексируется во время первой волны, в CSR, где индексируемый контент раскрывается только при визуализации, эта значительная задержка может означать, что контент может не индексироваться или не появляться в результатах поиска в течение нескольких дней или даже недель [1].

Рендеринг JavaScript очень ресурсоемкий. Это влечет значительное увеличение потребления электроэнергии, мощности процессора, потенциальное снижение скорости сканирования и многое другое. Это в свою очередь приводит к значительному увеличению денежных затрат. Эта дополнительная работа по визуализации JS актуальна не только для поисковых систем, этот эффект увеличения нагрузки можно наблюдать и на различных пользовательских устройствах. Например, по словам Сэма Сакконе из Google разбор 1 МБ JS кода на Samsung Galaxy S7 выполняется за 850 мс, а Nexus 5 за 1700 мс [6].

Также, Google официально заявил, что отображение сайтов с поддержкой JavaScript в поиске Google откладывается до тех пор, пока у робота Google не появятся ресурсы для обработки этого контента. Таким образом, при прочих равных условиях для сканирования и индексации веб-сайтов для JS требуется гораздо больше времени, чем для HTML [7].

Еще одной особенностью CSR является то, Googlebot использует Chrome 41 для рендеринга сайтов [8]. Этот браузер был выпущен в марте 2015 года. IT и, в частности, JavaScript за это время сильно развились. Есть много современных функций, которые недоступны для Googlebot. Некоторые из основных ограничений включают в себя:

* Chrome 41 лишь частично поддерживает современный синтаксис JavaScript ES6. Например, он не понимает новые языковые конструкции.
* Интерфейсы, вроде IndexedDB и WebSQL, отключены.
* Куки, локальные хранилища и сессионные хранилища очищаются при перезагрузке страницы [5].

HTML и JS коренным образом различаются в подходах к обработке ошибок. Одна ошибка в коде JavaScript может привести к тому, что Google не сможет отобразить страницу. Цитата Матиаса Шефера, автора онлайн-книги «Надёжный JavaScript»: «JS-парсер не отличается дружелюбием. Он совершенно нетерпим к ошибкам. Если он встречает символ, появление которого в определённом месте не ожидается, он немедленно прекращает разбор текущего скрипта и выдаёт SyntaxError. Поэтому единственный символ, находящейся не там, где нужно, единственная опечатка, может привести к полной неработоспособности скрипта» [9].

# 5 Влияние SSR и CSR на SEO

Браузер, применяемый в Google для формирования страниц сайтов, основанных на JS, может правильно обрабатывать сайты, использующие современные возможности JS, однако, разработчикам таких сайтов потребуется приложить для этого некоторые усилия. А именно, использовать полифиллы, создавать упрощённую версию сайта и выполнять транспиляцию кода в ES5.

Популярность JavaScript быстро росла и растет сейчас. Однако некоторые функции JavaScript не реализованы в старых браузерах и, как следствие, препятствуют рендерингу. Однако, это можно обойти, используя технику постепенной деградации. При реализации некоторых современных функций, которые поддерживают только несколько браузеров, нужно обеспечить использование упрощённой версии веб-сайта в других браузерах.

Выполняя анализ браузера, можно в любое время проверить, поддерживает он некую возможность или нет. Если он эту возможность не поддерживает, вместо неё, необходимо предложить то, что подходит для этого браузера. Эта замена и называется полифиллом. Кроме того, если нужно, чтобы сайт мог быть обработан поисковыми роботами Google, необходимо использовать транспиляцию JS-кода в ES5, то есть, преобразование тех конструкций JS, которые не понимает Googlebot, в конструкции, понятные ему. Например, когда транспилятор встречает выражение let x=5, он преобразует его в выражение var x=5 (эта конструкция понятна всем браузерам, в том числе и Chrome 41, который играет для особую роль) [10].

При использовании интернета, браузер (Chrome, Firefox, Opera и т. Д.) загружает все ресурсы (изображения, скрипты, таблицы стилей) и показывает визуализированный вид. Однако робот Google работает не так, как браузер. Он нацелен на сканирование всего интернета и получение только ценных ресурсов.

Интернет огромен, поэтому Google оптимизирует свои сканеры для повышения производительности. Вот почему робот Googlebot иногда не посещает все страницы, которые хотят веб-мастера. Алгоритмы Google пытаются определить, нужен ли ресурс с точки зрения рендеринга. Если нет, то, вероятно, Googlebot не получит его. Поэтому Google может не выбрать некоторые JS файлы, потому что его алгоритм решил, что в этом нет необходимости с точки зрения рендеринга или просто из-за проблем с производительностью (то есть выполнение сценария заняло слишком много времени). Хотя точное время ожидания не указано, говорится, что Google не может ждать сценарий более 5 секунд. На JavaScript SEO group, Джон Мюллер сказал: «Мы не определяем конкретное значение тайм-аута, в основном потому, что время, необходимое для извлечения ресурсов, не является детерминированным или сопоставимым с браузером (из-за кэширования, загрузки сервера и т. д.). 5 секунд - хорошая цель, я подозреваю, что многим сайтам будет сложно достичь ее» [5].

Если сайт загружается медленно, это может привести к различным последствиям, а именно:

* Посетителям сайта будет некомфортно с ним работать, они могут его покинуть;
* У Google могут возникнуть проблемы с анализом страниц;
* Это может замедлить процесс сканирования сайта. Если страницы медленны, Google может решить, что его роботы замедляют ваш сайт и снизить частоту сканирования.

# ЗАКЛЮЧЕНИЕ

В ходе данной работы была исследована тема оптимизации веб-ресурсов с использованием клиентских и серверных веб-технологий.

В работе внимание уделялось тенденциям развития JavaScript в сети,

разбору понятий и механизмов выполнения серверного и клиентского рендеринга.

В третьей части было выполнено сравнение SSR и CSR, а также выделены преимущества и недостатки каждого из методов.

Был исследован процесс рендеринга JavaScript и определены основные моменты, усложняющие процедуру рендеринга.

В последней части было изучено влияние SSR и CSR на SEO на основе чего, был сделан ряд выводов:

* Алгоритмы Google пытаются определить ценность ресурса с точки зрения целесообразности обработки этого ресурса для формирования страницы. Если ресурс не представляет ценности, Googlebot, возможно, не станет его загружать, поэтому стоит стремиться к тому, чтобы обеспечить скорость выполнения скриптов, потому что Googlebot не станет ждать результатов выполнения скрипта, включая этап загрузки, более 5 секунд.
* Необходимо помнить о том, что Googlebot использует Chrome 41 для рендеринга сайтов. Этот браузер был выпущен в марте 2015 года, таким образом есть много современных функций, которые недоступны для Googlebot, поэтому необходимо использовать технику постепенной деградации.
* Если Google не может сформировать страницу, используя JavaScript, он может решить проиндексировать изначально загруженный HTML. Это может очень плохо сказаться на одностраничных приложениях, так как Google, при таком подходе, проиндексирует пустую страницу.
* Чтобы получить лучшее от обоих видов рендеринга можно использовать рендеринг на стороне сервера для загрузки первой страницы, а рендеринг на стороне клиента для всех последующих загрузок страницы.
* Важно помнить, что SEO с учётом особенностей JavaScript выполняется на базе традиционной поисковой оптимизации. Невозможно хорошо оптимизировать сайт с учётом особенностей JS, не добившись достойной оптимизации в простом понимании.

# СПИСОК ЛИТЕРАТУРЫ

1. Benjamin Burkholder. JavaScript SEO: Server Side Rendering vs. Client Side Rendering [Электронный ресурс]: Medium. URL: <https://medium.com/@benjburkholder/javascript-seo-server-side-rendering-vs-client-side-rendering-bc06b8ca2383> (дата обращения 24.05.2019).
2. Alex Grigoryan. The Benefits of Server Side Rendering Over Client Side Rendering [Электронный ресурс]: Medium. URL: <https://medium.com/walmartlabs/the-benefits-of-server-side-rendering-over-client-side-rendering-5d07ff2cefe8> (дата обращения 24.05.2019).
3. Juan Vega. Client-side vs. server-side rendering: why it’s not all black and white [Электронный ресурс]: Freecodecamp. URL: <https://www.freecodecamp.org/news/what-exactly-is-client-side-rendering-and-hows-it-different-from-server-side-rendering-bd5c786b340d/> (дата обращения 24.05.2019).
4. Adam Zerner. Client-side rendering vs. server-side rendering [Электронный ресурс]: Medium. URL: <https://medium.com/@adamzerner/client-side-rendering-vs-server-side-rendering-a32d2cf3bfcc> (дата обращения 24.05.2019).
5. Tomasz Rudzki. The Ultimate Guide to JavaScript SEO [Электронный ресурс]: Onely. URL: <https://www.onely.com/blog/ultimate-guide-javascript-seo/> (дата обращения 24.05.2019).
6. Planning for Performance: PRPL (Chrome Dev Summit 2016) SEO [Электронный ресурс]: YouTube. URL: <https://www.youtube.com/watch?v=RWLzUnESylc> (дата обращения 24.05.2019).
7. Google I/O 2018 Summary - Day 3 [Электронный ресурс]: Auth0. URL: <https://auth0.com/blog/googleio-summary-day-three/> (дата обращения 24.05.2019).
8. 5. Tomasz Rudzki. Chrome 41: The Key to Successful Website Rendering [Электронный ресурс]: Onely. URL: <https://www.onely.com/blog/chrome-41-key-to-website-rendering/> (дата обращения 24.05.2019).
9. Martin Abegglen. Robust Client-Side JavaScript [Электронный ресурс]: Molily. URL: <https://molily.de/robust-javascript/> (дата обращения 24.05.2019).
10. David Gilbertson. Polyfills: everything you ever wanted to know, or maybe a bit less [Электронный ресурс]: Hackernoon. URL: <https://hackernoon.com/polyfills-everything-you-ever-wanted-to-know-or-maybe-a-bit-less-7c8de164e423> (дата обращения 24.05.2019).