Научная статья
УДК 004 DOI: 10.51623/23132027_122_130

ОБЗОР МЕТОДОВ ОПТИМИЗАЦИИ РЕНДЕРИНГА ПРИЛОЖЕНИЯ, РАЗРАБОТАННОГО С ПОМОЩЬЮ КЛИЕНТСКИХ ФРЕЙМВОРКОВ

Марина Владимировна Агафонова
Университет ИТМО, Санкт-Петербург, Россия,
agafonova_mv@niuitmo.ru

Аннотация. Нередко возникает ситуация, когда у начинающих разработчиков возникают проблемы с производительностью приложения, которое состоит из множества сложных элементов и содержит большое количество данных. В такие моменты возникает вопрос, каким именно способом можно улучшить приложение. Данная статья будет полезна для начинающих специалистов в качестве ознакомления с популярными методами оптимизации рендеринга страниц приложения, реализованного с помощью клиентских фреймворков таких, как ReactJS, Angular, VueJS. Описаны ситуации, при которых можно использовать те или иные методы и выделены наиболее популярные способы оптимизации и вариации их реализации.

Ключевые слова: веб-разработка, оптимизация рендеринга, JavaScript, клиентские фреймворки

Для цитирования: Агафонова М. В. Обзор методов оптимизации рендеринга приложения, реализованного с помощью клиентских фреймворков // Современное образование: традиции и инновации. 2022. №. С. 130.

Original article

OVERVIEW OF OPTIMIZATION METHODS OF RENDERING APPLICATION, DEVELOPED WITH CLIENT-SIDE FRAMEWORKS

Marina V. Agafonova
ITMO University, Saint Petersburg, Russia,
agafonova_mv@niuitmo.ru

Abstract. It's quite common for beginner developers to experience performance issues with an application that consists of many complex elements and contains a large amount of data. In such situations, the question arises of how exactly the application can be improved. This article will be useful for beginners as an introduction to popular methods for optimizing page rendering of the application implemented using client-side frameworks such as ReactJS, Angular, VueJS. Conditions for using specific methods are given for all methods, that are described in this article, and the most popular optimization methods with their variations of implementation are highlighted.

Keywords: web-development, rendering optimization, JavaScript, client-side frameworks

For citation: Agafonova M.V. Overview of optimization methods of rendering application, developed with client-side frameworks // Modern education: traditions and innovations. 2022. no. 1. P. 130.

В настоящее время разработка веб-приложений набирает большую популярность и спрос. Один из ключевых факторов для успешного приложения – это пользовательский

интерфейс (UI – user interface). Более того, UI должен быть как хорошо разработан визуально (привлекательный современный, а также интуитивный дизайн), так и профессионально сделан с хорошими показателями по производительности.

Под хорошими показателями по производительности подразумевается время загрузки страницы или обновление отдельных компонентов. По данным опроса Unbounce, 70% пользователей с меньшей вероятностью совершат покупку (и маловероятно, что вернутся в будущем), если время загрузки дольше, чем ожидалось [1]. Помимо этого, существуют иные проблемы из-за медленной загрузки страницы.

Если рассматривать отображение сайта в поисковых системах, то скорость загрузки сайта – это один из факторов ранжирования. Компания Google объявила об этом ещё в 2010 году, с годами значимость показателя «page speed» только росла. Если рассматривать то, как быстродействие влияет на SEO (англ. Search Engine Optimization), то можно выделить следующее:

1) Поисковые системы напрямую замедляют скорость загрузки, повышая приоритет быстрых сайтов.

2) Поисковые системы следят за поведением пользователей: если страницы реже посещают и чаще закрывают – рейтинг страницы понижается.

На сегодняшний день создано множество сервисов для проверки скорости веб-сайта. Самым популярным является PageSpeed Insights от компании Google. Данный сервис замеряет скорость загрузки на ПК и мобильных устройствах, предлагает свои рекомендации по ускорению работы веб-приложения. Но его рекомендации являются довольно поверхностными, которые не дадут сильно прироста в быстродействии сайта.

Одной из причин медленного рендеринга страницы может являться неверный выбор архитектурного стиля для веб-приложения. Такой случай рассматривается в работе Р.А. Торопкина, Я.В. Зиновьева, Н.С. Рассказова, М.А. Митрохина в 2020 году [2]. Из наиболее популярных методов авторы выделили такие способы оптимизации, как уменьшение объема загружаемых данных, сохранение части информации на устройстве пользователя или кэширования, «визуальная» оптимизация веб-страницы.

Также рассматривается способ вычислений на стороне клиента. Чем больше сторонних библиотек используется в проекте, тем больше информации нужно передать на пользовательское устройство. Но эту проблему решают такие инструментов для сборки, как gulp, Webpack. Они позволяют указывать среду, минимизировать исходные JS-файлы, использовать различные плагины для оптимизации сборки, позволяющие формировать файлы формата «.gz».

Помимо этого, авторы сравнивают SPA и MPA решения, т.к. принцип их работы значительно различается: в случае с MPA отправляются запросы при каждом изменении на странице, что приводит к ее принудительному обновлению, тогда как в SPA используется динамическое обновление DOM-дерева, что позволяет исключить перезагрузку веб-страницы.

Отмечается, что в последнее время становится популярной «отложенная загрузка» – это оптимизация загрузки медиафайлов и компонентов, некритичных для отображения веб-страницы и взаимодействия с интерфейсом. Такая загрузка больше всего подходит для тех элементов, которые располагаются за линией видимости пользователя и отображаются только после прокрутки. Подобная загрузка уже реализована во многих клиентских фреймворках, таких как Vue.js, Angular и React.

Важно помнить и о формате файлов, т.к. более экономичные форматы позволяют сократить объем передаваемого трафика. В качестве примера приводится формат WebP, который предоставляет возможность экономить до 34% при конвертировании картинки из формата JPG и 45% для формата PNG [3]. Однако, стоит учитывать, что только у 79,2% пользователей присутствует поддержка данного формата.

Для снижения нагрузки на устройство пользователя можно использовать серверный рендеринг. Его суть заключается в том, что вся нагрузка по отображению

страницы ложится на сервер, а ресурсы устройства клиента освобождаются для иных процессов [4].

Одним из самых популярных способов для оптимизации работы приложения является кеширования, и авторы статьи описывают одну из вариаций – использование «сервис-воркеров». Реализация такого способа заключается в том, что сайт разделяется на две части: со статическим и динамическим контентом. При повторной загрузке скачивается только часть с уникальным контентом и объединяется со статическим [5].

Более продвинутое кеширование и оптимизация веб-приложения с помощью технологии SWR описано в работе И.В. Князева. Технология SWR предоставляет возможность с минимальным количеством кода обрабатывать уже обновленный контент. Это является хорошим компромиссом между эффективностью веб-приложения и пользовательским опытом [6].

В другой работе И.В. Князева описывается такой способ оптимизации, как серверный рендеринг, который упоминался в работе Р.А. Торопкина, Я.В. Зиновьева, Н.С. Рассказова, М.А. Митрохина, но не был применен для рассматриваемого проекта. И.В. Князев в своей статье проанализировал продвинутые функции NextJS, реализацию самого кода, миграцию, настройку и развертывание приложения с использованием современного server side rendering (рендеринга на стороне сервера).

Как отмечает автор, в JS-фреймворках в основном используют рендеринг на стороне клиента (CSR – client side rendering). В таком случае сервер отправляет на страницу почти пустой файл HTML, за которым следуют все JS-файлы в одном пакете, который позже должен быть обработан в браузере пользователя для рендеринга DOM-дерева. На таких страницах начальная скорость загрузки низкая, поэтому пользователь видит пустой экран, пока не будет выполнен весь код JavaScript и не завершатся запросы к API. Но такой сценарий касается только первоначальной загрузки страницы. Последующие скорости загрузки будут быстрыми, т.к. дальнейшие изменения потребуют только обновления соответствующих разделов DOM-дерева.

SSR подход решает проблему начальной скорости загрузки в CSR, т.к. сервер извлекает информацию из БД и отправляет клиенту подготовленный файл HTML. Вторым преимуществом SSR перед CSR является поисковая оптимизация (SEO). Поскольку клиенту приходит уже подготовленная страница с соответствующими метаданными, алгоритмы поисковой системы могут легко классифицировать эти страницы, что приводит к более высокому поисковому рейтингу веб-сайта. Если рассматривать CSR, то в этом случае для заполнения методанных должен выполниться код JavaScript, и такое выполнение может занять более 300-400 мс. В результате такой задержки алгоритм обрабатывает пустую страницу, что негативно влияет на SEO показатели.

Использование исключительно одного из приведенных способов рендеринга страниц не будет гарантированно верным для конкретного приложения. Идеальным решением представляется создание гибридного приложения, которое использует преимущества обоих методов для оптимизации доступности и взаимодействия с пользователем. Возможность создания такого гибрида предоставляет Next.js [7].

В работе О.В. Бородина и В.А. Егунова рассматривает способ оптимизации, который заключается в многопоточности с использованием API Web-workers. Особенность WebWorkers API состоит в том, что веб-воркеры – это потоки, принадлежащие браузеру, которые можно использовать для выполнения JS-кода без блокировки цикла событий [8]. Использование воркеров наиболее подходит для таких ситуаций, как работа с файлами, шифрование, предварительная загрузка данных, проверка правописания, рендеринг трёхмерных сцен.

Авторы приводят измерения по времени выполнения операции по обратке изображения с помощью различного количества потоков. Если рассматривать ситуацию с 500 изображениями, то было достигнуто ускорение в три раза при использовании четырех

потоков (воркеров) вместе с основным потоком, по сравнению с использованием только основного потока. На основе полученных результатов авторы выделили следующие достоинства и недостатки технологии.

Достоинства web workers:

1) поток пользовательского интерфейса свободен и может выполнять отрисовку и служебные задачи вовремя;

2) скорость выполнения вычислений увеличивается, но только для тех случаев, которые можно выполнять параллельно;

3) данная технология имеет хорошую поддержку и удобный интерфейс взаимодействия, кроссбраузерность и кроссплатформенность, не нуждается в настройке и установке.

Недостатки web workers:

1) накладные расходы на пересылку и сериализацию данных, невозможность передачи данных воркеру по ссылке;

2) некоторые алгоритмы могут потребовать некоторого видоизменения для учета всех тонкостей;

3) некоторые данные могут потребовать конвертацию из-за потери контекста и отсутствия доступа к некоторым нужным api [9].

Как и в работе Р.А. Торопкина, Я.В. Зиновьева, Н.С. Рассказова, М.А. Митрохина, в статье зарубежных ученых V. Gowda и S. Rangaswamy было отмечено, что неоптимальные архитектурные решения для конкретного проекта могут повлиять на эффективность работы веб-приложения [10]. В их работе, посвященной устранению ограничений ReactJS, отмечается такая проблематичная ситуация, когда при каждом клике мыши на строку таблицы, вся таблицы перерисовывалась, что приводило в итоге к низким показателям производительности. Авторы отмечают, что данную проблему со строками таблицы можно решить изоляцией каждой строки или столбцы таблицы в качестве независимого компонента. В результате эти компоненты будут отслеживать и реагировать на события независимо друг от друга. Таким образом, каждый раз, когда происходит событие, необходимо повторно отобразить только одну строку или столбец вместо повторного отображения всей таблицы.

Довольно глобальным способом оптимизации является архитектурный стиль приложения. S.K. Mukhiya и H.K. Hung упоминают в своей статье работу J. Nielsen, который описывал сервисно-ориентированную архитектуру (SOA – service-oriented architecture) [11]. Данная архитектура помогает достичь масштабируемости, улучшить пользовательский отклик, сократить задержки и организовать модель программирования. Но такой архитектурный стиль требует, чтобы все стандарты веб-сервисов перенимали одни и те же технические стандарты. Такую архитектуру можно совмещать с другими методами оптимизации, что положительно скажется на производительности приложения.

Заключение. Одним из самых популярных способов оптимизации является кеширование. Существует множество вариантов реализации данного способа: использование «сервис-воркеров», более продвинутое кеширование с помощью технологии SWR. Вторым из часто упоминаемых способов оптимизации является многопоточность. В рассмотренных материалах многопоточность реализовывалась с помощью веб-воркеров. Важным аспектом является архитектурный стиль проекта. Неверно подобранный паттерн может сильно сказаться на производительности приложения, необходимо понимать принципы работы SPA и MPA.

Список источников

1. Зачем и как проверять скорость загрузки сайта? [Электронный ресурс] URL: https://habr.com/ru/post/507746/ (дата обращения: 07.11.2021).

2. Торопкин Р.А., Зиновьев Я.В., Рассказов Н.С., Митрохин М.А. Технологии оптимизации работы сайта на примере аналитической системы публикационной активности пензенского государственного университета // Модели, системы, сети в экономике, технике, природе и обществе. – 2020. – № 4 (36). – С. 71–78. URL: ehttps://cyberleninka.ru/article/n/tehnologii-optimizatsii-raboty-sayta-na-primere-analiticheskoy-sistemy-publikatsionnoy-aktivnosti-penzenskogo-gosudarstvennogo (дата обращения 09.11.2021).

3. How to Make Your Website Load Faster With WebP Images. [Электронный ресурс] URL: https://techstacker.com/load-website-faster-with-webp-images/mhzdwjfko9eb4fbep/ (дата обращения: 10.11.2021).

4. Start Performance Budgeting. [Электронный ресурс] URL: https://medium.com/@addyosmani/start-performance-budgeting-dabde04cf6a3 (дата обращения: 10.11.2021).

5. Как ускорить загрузку сайта и улучшить поведенческие факторы. [Электронный ресурс] URL: https://ru.megaindex.com/blog/service-workers-site-speed (дата обращения: 11.11.2021).

6. Князев И.В. Продвинутое кеширование и оптимизация веб-приложений с помощью технологии SWR // Sciences of Europe. – 2021. – № 73 (1). – С. 47-49. URL: https://cyberleninka.ru/article/n/prodvinutoe-keshirovanie-i-optimizatsiya-veb-prilozheniy-s-pomoschyu-tehnologii-swr (дата обращения: 17.11.2021).

7. Князев И.В. Анализ работы приложения с использованием server-side rendering: миграция, настройка и развертывание приложения Next.js // Sciences of Europe. – 2021. – № 76 (1). – С. 71-74. URL: https://cyberleninka.ru/article/n/analiz-raboty-prilozheniya-s-ispolzovaniem-server-side-rendering-migratsiya-nastroyka-i-razvertyvanie-prilozheniya-next-js (дата обращения: 25.11.2021).

8. Background processing using web workers. [Электронный ресурс] URL: https://angular.io/guide/web-worker (дата обращения: 29.11.2021).

9. Бородин О.В., Егунов В.А. Многопоточная обработка изображений с использованием API Web-workers // CASPIAN JOURNAL: Control and High Technologies. – 2021. – № 3 (55). – С. 33-46. URL: https://cyberleninka.ru/article/n/mnogopotochnaya-obrabotka-izobrazheniy-s-ispolzovaniem-api-web-workers (дата обращения: 3.12.2021).

10. Gowda V., Rangaswamy S. Addressing the Limitations of React JS // International Research Journal of Engineering and Technology (IRJET). 2020. № 7(4). С. 5065-5068. URL: https://www.irjet.net/archives/V7/i4/IRJET-V7I4966.pdf (дата обращения: 9.12.2021).

11. Mukhiya S.K., Hung H.K. An Architectural Style for Single Page Scalable Modern Web Application // International Journal of Recent Research Aspects. – 2018. – № 5(4). – С. 6-13. URL: https://www.researchgate.net/publication/335259756_An_Architectural_Style_for_Single_Page_Scalable_Modern_Web_Application (дата обращения: 17.12.2021)

References

1. Why and how to check website loading speed? [Electronic resource] URL: https://habr.com/ru/post/507746/ (date of access: 11/07/2021).

2. R.A. Toropkin, Ya.V. Zunov'ev, N.S. Rasskazov, M.A. Mitrokhin. Site optimization technologies on the example of the analytical system of publishing activity of Penza State University // Models, systems, networks in economics, technology, nature and society. – 2020. – No. 4(36). – P. 71-78. URL: https://cyberleninka.ru/article/n/tehnologii-optimizatsii-raboty-sayta-na-primere-analiticheskoy-sistemy-publikatsionnoy-aktivnosti-penzenskogo-gosudarstvennogo (date of access: 11/09/2021).

3. How to Make Your Website Load Faster With WebP Images. [Electronic resource] URL: https://techstacker.com/load-website-faster-with-webp-images/mhzdwjfko9eb4fbep/ (date of access: 11/10/2021).
4. Start Performance Budgeting. [Electronic resource] URL: https://medium.com/@addyosmani/start-performance-budgeting-dabde04cf6a3 (date of access: 11/10/2021).
5. How to speed up site loading and improve behavioral factors. [Electronic resource] URL: https://ru.megaindex.com/blog/service-workers-site-speed (date of access: 11/11/2021).
6. I.V. Kniazev. The advanced web applications caching and optimization using SWR // Sciences of Europe. – 2021. – No. 73(1). – P. 47-49. URL: https://cyberleninka.ru/article/n/prodvinutoe-keshirovanie-i-optimizatsiya-veb-prilozheniy-s-pomoschyu-tehnologii-swr (date of access: 11/17/2021).
7. I.V. Kniazev. Analyzing application performance using server-side rendering: migrating, configuring and deploying the Next.JS application // Sciences of Europe. – 2021. – No. 76(1). – P. 71-74. URL: https://cyberleninka.ru/article/n/analiz-raboty-prilozheniya-s-ispolzovaniem-server-side-rendering-migratsiya-nastroyka-i-razvertyvanie-prilozheniya-next-js (date of access: 11/25/2021).
8. Background processing using web workers. [Electronic resource] URL: https://angular.io/guide/web-worker (date of access: 11/29/2021).
9. O.V. Borodin, V.A. Egunov. Multi-threaded fronted image processing using Web-workers API // CASPIAN JOURNAL: Control and High Technologies. – 2021. – No. 3(55). – P. 33-46. URL: https://cyberleninka.ru/article/n/mnogopotochnaya-obrabotka-izobrazheniy-s-ispolzovaniem-api-web-workers (date of access: 12/3/2021).
10. Gowda V., Rangaswamy S. Addressing the Limitations of React JS // International Research Journal of Engineering and Technology (IRJET). – 2020. – No. 7(4). – P. 5065-5068. URL: https://www.irjet.net/archives/V7/i4/IRJET-V7I4966.pdf (date of access: 12/9/2021).
11. Mukhiya S.K., Hung H.K. An Architectural Style for Single Page Scalable Modern Web Application // International Journal of Recent Research Aspects. – 2018. – No. 5(4). – P. 6-13. URL: https://www.researchgate.net/publication/335259756_An_Architectural_Style_for_Single_Page_Scalable_Modern_Web_Application (date of access: 12/17/2021)

Статья поступила в редакцию 20.02.2022, одобрена после рецензирования 21.02.2022, принята к публикации 21.02.2021.
The article was submitted 20.02.2022; approved after reviewing 21.02.2022; accepted for publication 21.02.2022.

Научная статья
УДК 004.457 *DOI: 10.51623/23132027_122_135*

ИСПОЛЬЗОВАНИЕ ТЕХНОЛОГИЙ DOCKER И KUBERNETES ДЛЯ ПОСТРОЕНИЯ ИНФРАСТРУКТУРЫ СЛОЖНЫХ СИСТЕМ

Манойло Виталий Евгеньевич
Университет ИТМО Санкт-Петербург, Россия
vitalii.manoilo3@gmail.com

Аннотация. В данной статье рассматриваются проблемы построения инфраструктуры веб приложений, с использованием микросервисной архитектуры, а также проблемы сопровождения подобных систем на примере систем, используемых для образования, а также организации деятельности образовательных учреждений. Также