МЕТОДЫ ОПТИМИЗАЦИИ ПРОИЗВОДИТЕЛЬНОСТИ WEB-ПРИЛОЖЕНИЙ.
Логинова Н.В.

Логинова Наталья Владиславовна - старший инженер-программист,
Epam systems Montenegro DOO Podgorica,
г. Подгорица, Черногория

Аннотация: *статья посвящена методам оптимизации производительности web-приложений. В работе рассматриваются различные подходы, такие как минификация и обфускация кода, кэширование, оптимизация изображений и базы данных, применение CDN, разделение кода и асинхронные API, Lazy loading и Optimistic UI Updates. Также представлены статистические данные и диаграммы, демонстрирующие влияние каждого из указанных методов на производительность web-приложения. В статье расссмотрены конкретные примеры использования данных методов в крупнейших мировых корпорациях. Данная работа будет полезна разработчикам web-приложений, менеджерам проектов и всем, кто стремится улучшить производительность своих web-приложений.*
Ключевые слова: *оптимизация производительности, frontend разработка, web-приложения, UX, улучшение скорости загрузки страницы, анализ производительности.*

УДК 004.05

1. Введение

В современном цифровом мире эффективность и быстродействие web-приложений становятся ключевыми факторами успешности любого бизнеса онлайн. Время отклика, скорость загрузки страниц, объем используемых ресурсов влияют на общую эффективность приложения и удовлетворенность пользователя [11, 12]. Оптимизация web-приложений стала одной из основных задач разработчиков и администраторов, принимающих участие в создании и поддержании этих приложений.

Целью данной статьи является исследование, анализ и описание основных методов, позволяющих оптимизировать производительность web-приложений. Будут рассмотрены такие методы как минификация и обфускация кода, оптимизация изображений, использование сетей доставки контента (CDN), оптимизация баз данных, техника разделения кода (Code Splitting), применение асинхронных API, и тактики такие как "Lazy Loading" и "Optimistic UI Updates" [2, 3].

Актуальность статьи обусловлена все возрастающей ролью web-приложений в бизнесе, обучении, социальных сетях и других сферах жизни, что влечет за собой необходимость полного понимания механизмов оптимизации и способов их реализации. В данной статье мы постараемся собрать и систематизировать ключевые подходы к оптимизации производительности web-приложений [1, 9].

2. Обзор методов оптимизации производительности web-приложений

Оптимизировать производительность web-приложения — значит, улучшить его работу, сделать его быстрее и эффективнее. В этом разделе мы рассмотрим несколько основных методов оптимизации, потенциально способных улучшить web-приложение.

- Минификация и обфускация кода: Минификация кода - это процесс удаления всех ненужных символов из исходного кода без изменения его функциональности. Обфускация - это метод, который делает код трудным для понимания и изменения, в то же время сохраняя его функциональность. Эти методы уменьшают размер файлов, что может привести к ускорению загрузки страниц.
- Кэширование: Это процесс хранения данных, которые используются повторно, для уменьшения времени, необходимого для загрузки или достижения этих данных. Кэширование может применяться на нескольких уровнях: на стороне клиента (в браузере), на уровне сервера или с использованием сетей доставки контента (CDN).
- Оптимизация изображений: Часто изображения являются наиболее "тяжелыми" для загрузки ресурсами на веб-странице. Их оптимизация может включать в себя изменение размеров, смену формата, сжатие без потери качества и др.
- Использование CDN: CDN, или сети доставки контента, используются для быстрой доставки контента пользователям. они позволяют хранить кэшированные версии контента на серверах, расположенных в разных местах мира, для ускорения загрузки контента.
- Оптимизация базы данных: Эффективность работы с базой данных зависит не только от самой СУБД, но и от того, как организована работа с ней. Важно правильно налаживать процессы индексации,

проводить нормализацию и денормализацию данных, а также оптимизировать самые "тяжелые" и частые запросы.

- Code Splitting: Это метод разделения кода на отдельные пакеты, которые загружаются только при необходимости. Такой подход позволяет ускорить начальную загрузку страницы.
- Использование браузерных асинхронных API и техники Lazy Loading: С асинхронными API и техникой отложенной загрузки можно выполнить некоторые задачи в фоновом режиме, пока клиентавых API и техники Lazy Loading: С асинхронными API и техникой отложенной загрузки можно выполнить некоторые задачи в фоновом режиме, пока клиент продолжает взаимодействовать с приложением.
- Optimistic UI Updates: Эта техника предполагает немедленное отображение действий пользователя в UI, даже если эти действия еще не подтверждены сервером. Такой подход может улучшить восприятие производительности приложения пользователем.

Эти методы оптимизации обеспечивают широкий диапазон возможностей для улучшения производительности web-приложений [4, 7]. Они могут использоваться отдельно или вместе, в зависимости от конкретных потребностей приложения.

3. Практические случаи применения методов оптимизации

В применении теории к практике следует обратить внимание на успешные случаи использования методов оптимизации производительности web-приложений. Применяя методы минификации и обфускации кода, корпорация Google успешно уменьшает объем передаваемых данных и ускоряет загрузку своих сложных web-приложений, таких как Google Search и Google Maps.

В компании Facebook, где основной объем данных представляют собой изображения от пользователей, прибегают к алгоритмам сжатия изображений. По сути, это позволяет им минимизировать объем передаваемых данных без значительного снижения качества изображений. Такой подход хорошо обрабатывает большой поток данных и вместе с тем сохраняет высокое качество пользовательского опыта.

Веб-гигант Amazon ускоряет загрузку своих веб-страниц за счет использования глобальной сети CDN. С помощью CDN, содержимое сайтов Amazon быстро доставляется пользователю из любой точки мира, что ускоряет время загрузки и улучшает взаимодействие с сайтом.

Содействуя ускорению загрузки своих веб-страниц, GitHub применяет технику Code Splitting. В отличие от классических методов, эта техника позволяет загружать только нужную в данный момент функциональность, обеспечивая быстродействие и удобство использования сайта.

Netflix, сервис потокового воспроизведения видео, использует асинхронные API и технику Lazy Loading для управления большими объемами данных и улучшения пользовательского опыта при просмотре фильмов и сериалов. Это позволяет ускорить загрузку контента и сделать его доступным для просмотра быстрее, нежели это было бы при классических методах загрузки.

LinkedIn в свою очередь, применяет паттерн Optimistic UI Updates для обеспечения быстрого взаимодействия пользователя с приложением. Даже при невысокой скорости интернета, в результате оптимизации, приложение ощущается быстрым и отзывчивым.

Все указанные примеры подтверждают важность проведения исследований для определения эффективных методов оптимизации производительности конкретного web-приложения [5, 6]. Значительное улучшение производительности web-приложений, пользовательского опыта и в целом успешного функционирования онлайн-сервиса свидетельствует об эффективности применения данных методов.

4. Примеры применения методов оптимизации

Проведём сравнительный анализ времени загрузки страницы, написанного автором web-приложения, до и после применения методов оптимизации, рассматриваемых в статье. На Рисунке 1 представлена диаграмма по данным времени загрузки страницы в секундах при применении разных методов оптимизации.

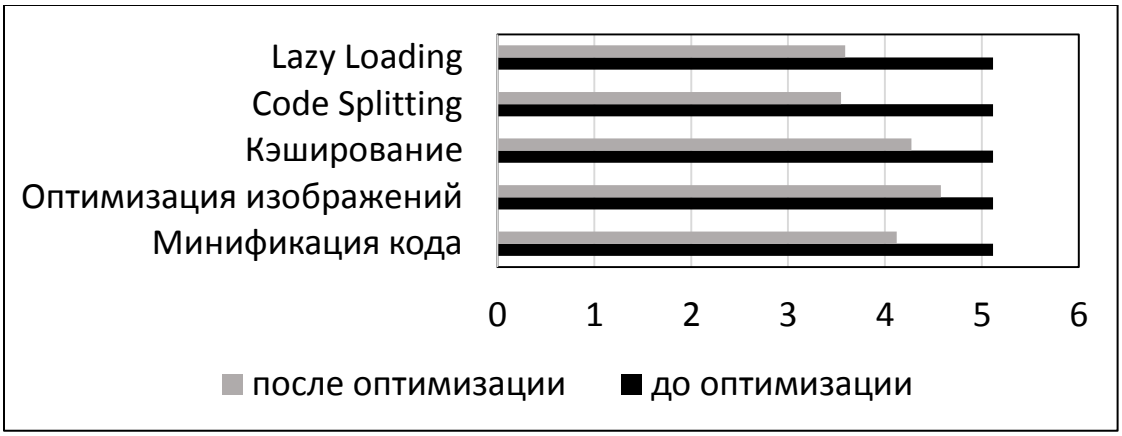

Рис. 1. Сравнение времени загрузки страницы в секундах до и после оптимизации.

Анализ данных с диаграммы показывает, что каждый из примененных методов оптимизации по отдельности вносит значительный вклад в улучшение времени загрузки страницы. Метод минификации кода позволил сократить время загрузки на примерно 19%, оптимизация изображений также урезала время загрузки примерно на 10.5%, кэширование позволяет сократить время загрузки на около 16.5%, Code Splitting и Lazy Loading улучшили время загрузки на 30.7% и 29.9% соответственно.

Далее применим группу рассмотренных методов для оптимизации времени загрузки страницы. Это завершающий шаг, который поможет нам визуализировать и оценить эффективность примененных методов. В рамках этого этапа мы проводим диагностику производительности веб-страницы, используя функционал Chrome DevTools. В результате анализа мы получаем такую информацию как время загрузки, время скриптинга, рендеринга, отрисовки, системное время и простоя.

Для наглядности их вклада в общее время создадим "Круговую диаграмму распределения времени между различными стадиями обработки веб-страницы". Это даёт возможность визуально сравнить, как применяемые методы оптимизации влияют на каждую стадию обработки веб-страницы.

На Рисунке 2 представлена диаграмма по данным распределения времени в миллисекундах между различными стадиями обработки веб-страницы до оптимизации.

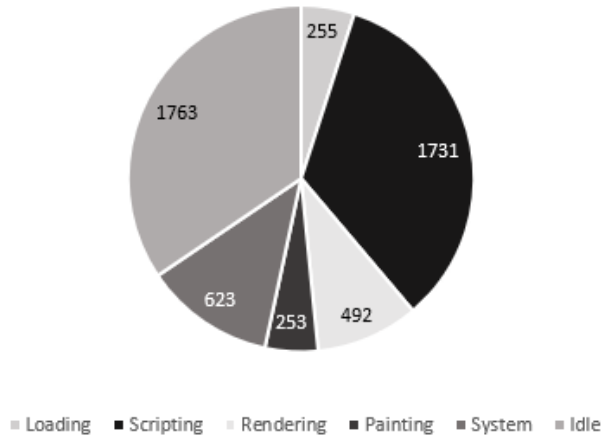

Рис. 2. Распределение времени в миллисекундах между различными стадиями обработки веб-страницы до оптимизации.

Из данных, представленных в диаграмме, видим, что наибольший вклад в общее время занимает стадия "Scripting", составляя примерно 33,8% от общего времени. "Idle" или время простоя составляет около 34,4%. Остальное время распределяется между "Loading" (5%), "Rendering" (9,6%), "Painting" (4,9%) и "System" (12,2%).

На Рисунке 3 представлена диаграмма по данным распределения времени в миллисекундах между различными стадиями обработки веб-страницы после оптимизации.

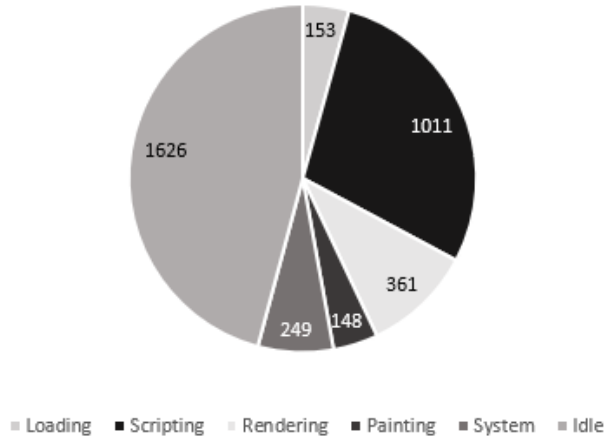

Рис. 3. Распределение времени в миллисекундах между различными стадиями обработки веб-страницы после оптимизации.

Из данных, представленных в диаграмме, видим, что время, затрачиваемое на "Scripting", снизилось до 28,5% от общего времени, а время простоя "Idle" стало составлять 45,9%. Также наблюдается уменьшение времени других этапов в процентах от общего времени: "Loading" (4,3%), "Painting" (4,2%) и "System" (7%), а время "Rendering" стало состаялять 10,2% от общего времени.

Из сравнения двух диаграмм видно, что применяемые методы оптимизации привели к снижению времени всех стадий загрузки и обработки, что говорит об общем улучшении производительности веб-страницы. Использование данных диаграмм может быть полезно для идентификации «узких мест» по производительности и помочь понять, на что нужно обратить внимание в первую очередь при проведении дальнейших оптимизаций.

Далее рассмотрим долю каждого из применённых методов оптимизации в общем улучшении производительности приложения. Рассчитаем долю каждого метода в процентах как отношение улучшения времени загрузки для этого метода к общему улучшению.

На Рисунке 4 представлена диаграмма, отображающая долю каждого из методов оптимизации в процентах в общем улучшении производительности приложения.

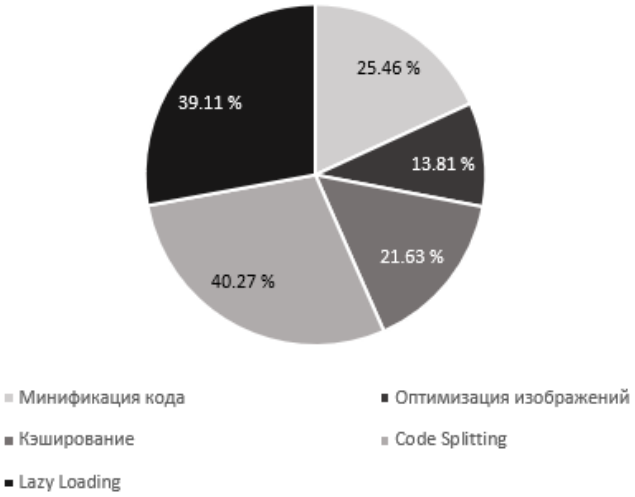

Рис. 4. Доли каждого из методов оптимизации в процентах в общем улучшении производительности приложения.

Диаграмма визуализирует вклад каждого из пяти рассмотренных методов в увеличение скорости загрузки. Очевидно, что код разделения (Code Splitting) и ленивая загрузка (Lazy Loading) имеют наибольший вклад из всех методов, с долями в 40,27% и 39,11% соответственно. Это свидетельствует о том, как эти две техники могут значительно влиять на производительность приложения. Минификация кода, является следующей по значимости техникой, способствующей улучшению производительности, составляя 25,46% от общего вклада. Оптимизация изображений и кэширование также вносят свой вклад, составляя 13,81% и 21,63% соответственно. Вместе все эти методы оптимизации влияют на уменьшение времени загрузки веб-страницы, повышая ее производительность. Обратите внимание, что сумма процентов превышает 100%, потому что методы оптимизации могут пересекаться и вместе улучшить производительность больше, чем по отдельности.

Из построенных диаграмм следует, что применённые в работе методы оптимизации показали значительное улучшение производительности веб-страницы. После применения методов оптимизации время загрузки страницы сократилось с 5117 миллисекунд до 3548 миллисекунд. Это составляет уменьшение примерно на 30%, что является заметным улучшением в производительности. Этот результат подчеркивает значение применения оптимизаций для улучшения пользовательского опыта и общей эффективности веб-страницы.

Однако важно помнить, что оптимизация веб-страницы — это процесс непрерывного улучшения. Всегда есть место для дополнительных улучшений и работы над производительностью, ведь потребности пользователей и технологии продолжают меняться.

5. Общие рекомендации и лучшие практики для оптимизации производительности web-приложений

Оптимизация производительности web-приложений — это сложный и непрерывный процесс, который требует постоянного мониторинга и регулярного применения различных методов. Прежде всего, важно отслеживать производительность вашего приложения в режиме реального времени, используя инструменты аналитики и профилирования. Это позволит выявлять проблемные места сразу же, как они появятся, и своевременно принимать меры по их устранению.

Одним из самых эффективных методов оптимизации является использование сетей доставки контента (CDN). Если ваше приложение обслуживает пользователей из различных частей мира, CDN помогут улучшить время отклика и сократить нагрузку на ваши основные сервера. CDN работают, храня копии вашего контента на серверах, расположенных ближе к местоположению пользователей, что ускоряет время его доставки.

Большая часть данных, которые отображаются в браузере пользователя, часто представляет собой код: HTML, CSS или JavaScript. Минификация и обфускация кода позволяют уменьшить его объем, делая ваше приложение более компактным и быстрее загружающимся. Не забывайте, что данные методы должны внедряться на этапе сборки вашего приложения [8].

Изображения часто являются еще одной большой частью данных, которые загружаются на веб-странице. Оптимизация изображений — это процесс, который может включать в себя изменение размеров изображения, сжатие или смену формата. Это помогает сохранить равновесие между качеством изображения и его влиянием на производительность приложения [10].

Также код вашего приложения можно организовать так, чтобы не все его части загружались сразу. Техники, такие как разделение кода (Code splitting) или отложенная загрузка (Lazy loading), позволяют вам загружать только ту функциональность, которая необходима пользователю в данный момент. Это не только сокращает время загрузки страницы, но и экономит пропускную способность сети [2].

Оптимизация взаимодействия с базой данных также играет важную роль в повышении производительности web-приложений. В данной области обычно стоит обратить внимание на SQL запросы, которые могут быть точками узкого прохода. Здесь может помочь оптимизация запросов и структуры данных, использование индексов и другие подходы.

И, наконец, оптимизация должна быть инкрементной, то есть постепенной. Она тесно связана с аналитикой производительности, ее анализом и тестированием. Нельзя просто внедрить все перечисленные методы и ожидать мгновенного улучшения - каждая оптимизация требует тестирования и анализа своего эффекта. Помните, что каждое приложение уникально, и потому та оптимизация, которая работает для одного приложения, может не работать или даже ухудшить производительность другого.

6. Заключение

Таким образом, оптимизация производительности web-приложений играет важную роль в обеспечении отличного пользовательского опыта, достижении бизнес-целей и построении успешных онлайн-сервисов.

Использование таких методов, как минификация и обфускация кода, кэширование, оптимизация изображений, применение CDN, оптимизация базы данных, Code Splitting, асинхронные API, Lazy loading и Optimistic UI Updates, может значительно повысить производительность вашего web-приложения [5, 7].

Однако стоит помнить, что оптимизация — это процесс непрерывного улучшения, требующий регулярного мониторинга, анализа и тестирования. Также важно понимать, что каждое приложение уникально, и методы, которые работают для одного приложения, могут быть не так эффективны для другого. Поэтому необходимо подходить к оптимизации индивидуально, с учетом специфики вашего приложения и его пользователей.

Учитывая успешные практические случаи применения методов оптимизации, представленные в этой статье, можно с уверенностью сказать, что правильно примененные методы оптимизации способны не только ускорить загрузку вашего web-приложения, но и существенно улучшить восприятие его пользователями. Это, в свою очередь, приведет к увеличению количества постоянных пользователей, росту продаж и успешному развитию бизнеса.

Список литературы

1. *Flanagan D.* (2011). JavaScript: The Definitive Guide: Activate Your Web Pages. O'REILLY.
2. *Srivastava A.* (2015). Web Performance Daybook Volume 2: Techniques and Tips for Optimizing Web Site Performance. O'Reilly Media.
3. *Grigorik I.* (2013). High Performance Browser Networking: What every web developer should know about networking and web performance. O'Reilly Media.
4. *Mechanic M.* (2018). Web Performance in Action: Building Fast Web Pages. Manning Publications.
5. *Myers S.* (2008). High Performance Web Sites: Essential Knowledge for Front-End Engineers. O'Reilly Media.
6. *Lindley S.* (2017). Pro Website Development and Operations: Streamlining DevOps for large-scale websites. Apress.
7. *Souders S.* (2009). Even Faster Web Sites: Performance Best Practices for Web Developers. O'Reilly Media.
8. *Farrelly E.* (2019). Javascript Performance: Optimize ES2015, jQuery, Angular, React and more. Packt Publishing.
9. *Grigorik I.* (2014). Web Page Optimization: The Pragmatic Approach. O’Reilly Media.
10. *Souder S.* (2018). Optimizing the Critical Rendering Path: Advanced Page Load Speed Techniques. O’Reilly Media.
11. *Jon Duckett.* (2014) JavaScript and JQuery: Interactive Front-End Web Development.
12. *Marijn Haverbeke*. (2018) Eloquent JavaScript.