УДК 681.3

Кожантаев Тимур Рафаэлевич, магистрант,
Ошский технологический университет

ВИДЫ РЕНДЕРИНГА В ВЕБ-ПРОГРАММИРОВАНИИ

В статье освещены виды рендеринга веб-страниц, показаны и описаны примеры использования, проанализированы преимущества и недостатки каждого из способов. Статья помогает принять верное решения по выбору вида рендеринга.

Ключевые слова: веб-сайт, рендеринг, фреймворк, веб-приложение, сервер, DOM (Интерфейс программирования приложений), HTML (Стандартизированный язык разметки документов), Javascript или JS (Язык программирования), CRS (рендеринг на стороне веб-клиента), SSR (рендеринг на стороне сервера).

Кожантаев Тимур Рафаэлевич, магистрант,
Ош технологиялык университети

ВЕБ ПРОГРАММАЛАРДА РЕНДИНГТИН ТҮРЛӨРҮ

Макалада веб-баракчаларды көрсөтүүнүн түрлөрү баса белгиленет, колдонуу мисалдары көрсөтүлөт жана сүрөттөлөт, ар бир ыкманын артыкчылыктары жана кемчиликтери талданат. Макала көрсөтүү түрүн тандоо боюнча туура чечим чыгарууга жардам берет.

Ачкыч сөздөр: веб-сайт, рендеринг, фреймворк, веб-тиркеме, сервер, DOM (Колдонмо программалоо интерфейси), HTML (Стандартташтырылган документ белгилөө тили), Javascript же JS (Программалоо тили), CRS (Web Client Side Rendering), SSR (Server Side Rendering) .

Kozhantaev Timur Rafaelevish, graduate student,
Osh Technological University

RENDERING TYPES ON WEB DEVELOPMENT

The article highlights the rendering types, show and descripts the examples, analyzes the advantages and disadvantages of each rendering types, that allows to make the right decision of the necessary type for the certain website.

Key words: website, rendering, framework, web application, server, DOM (Document Object Model), HTML (Hyper Text Markup Language), Javascript or JS (computer programming language), CSR (Client Side Rendering), SSR (Server Side Rendering).

Введение. Современные веб-сайты имеют сложный функционал и зачастую работают как веб-приложения. Можно отправлять сообщения, совершать покупки, регистрировать сделки и многое другое. Отчетливо отслеживается тенденция к большему взаимодействию со стороны пользователей. При наплыве пользователей, сервер может не справляться с одновременной обработкой большого количества запросов.

В недалеком прошлом, веб-сайты представляли собой страницы с информацией, с отсутствием или минимальным количеством интерактивных элементов и функций. HTML, CSS, и JavaScript являлись стандартными путями для форматирования

элементов, стилизации и написания логики. Для того чтобы отобразить веб-страницу, нужно было просто отправить HTML документ на сервер.

Однако для большинства современных, в частности коммерческих веб-сайтов и приложений, SSR является неэффективным решением. Компании хотят использовать динамический контент и правильно его отображать на страницах, чтобы все элементы загружались быстро и плавно, с высокой скоростью, имели красивый пользовательский интерфейс и получали оптимизацию поисковых систем уже в результатах поиска. И это не удивительно, так как основным инструментом продаж для большинства компаний являются интернет площадки.

Тем не менее в некоторых случая использование SSR является более целесообразным. В этой статья мы рассмотрим CSR и SSR, их преимущества и недостатки, в каких случаях следует использовать тот или иной подход.

Цель исследования. Основной целью исследования являются определение и выявление плюсов и минусов видов рендеринга в зависимости от типа веб-ресурса. Подведение итогов анализа и формирование вывода..

Материалы и методы исследования. Рендеринг на стороне клиента – это рендеринг приложения в браузере, обычно с использованием DOM (CSR). CSR стал популярным после появления фреймворков JS таких как Angular, React, Vue.js и т. д. В веб-приложении с использованием CSR JavaScript управляет всем, что отображается на странице. Как правило, вместо загрузки всего контента с помощью HTML-документов, подключается файл JavaScript для обработки динамической архитектуры загрузки веб-сайта. Вот что происходит, когда контент загружается на стороне клиента.

• Пользователь отправляет запрос на доступ к веб-контенту в браузере, используя адрес веб-сайта (ссылку).

• Сервер передает статические файлы (CSS и HTML) браузеру клиента при первом запросе на веб-сайт.

• Браузер клиента сначала загружает содержимое HTML, а затем код JavaScript. Файлы HTML связывают файлы JavaScript. Этот процесс загрузки происходит, когда сайт еще не виден пользователю.

• После того, как браузер загрузит JavaScript, содержимое динамически генерируется в браузере.

• Веб-контент становится видимым, когда клиент перемещается по веб-сайту и взаимодействует с ним.

Этот процесс означает, что первичная загрузка контента затягивается. Тем не менее после загрузки, работа веб-сайта должна быть гладкой и быстрой, и для динамического получения контента потребуются только API вызовы.

CSR является хорошим решением для одностраничных приложений (SPA). В SPA каждая страница уже есть в клиентском браузере, и сервер обслуживает только один HTML-документ.

```html
<html lang="en">
    <head>
        <meta charset="UTF-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <title>CSR</title>
        <script src="https://unpkg.com/vue@3.0.0"></script>
    </head>
    <body>
        <div id="app">
            <p>{{ title }} - {{ name}}, {{ age }} years old</p>

            <button v-on:click="age++">Увеличить возраст</button>
            <button @click="age--">Уменьшить возраст</button>
            <div @click="changeTitle('Oathbringer')">Заменить навание заголовка</div>
        </div>
        <script src="app.js"></script>
    </body>
</html>
```

Рис. 1 Пример CSR с использованием фреймворка Vue.js.

После загрузки HTML, структурой DOM веб-сайта в браузере будет управлять JS Фреймворк, например, React. Каждый элемент будет загружаться с помощью данных ранее полученных API фреймворка. Как только первичная загрузка закончится, переключение между вкладками, например, или перезагрузка страницы будут очень быстрыми.

Когда файл отправляется в браузер, веб-страница представляет собой пустой HTML-документ. Обычно видимая часть страницы не отображается.

Соответственно Vue.js получит полный контроль над страницей. То есть контент будет отображаться с помощью JS. [1]

```
const app = Vue.createApp({
    data() {
        return {
            title: "Harry Potter and the Half-Blood Prince",
            name: "J. K. Rowling",
            age: 56,
        };
    },
    methods: {
        changeTitle(title) {
            // this.title = 'Words of Radiance'
            this.title = title;
        },
    },
});
app.mount("#app");
```

Рис. 2 Пример создания приложения с помощью Vue JS

После обновления страницы содержимое веб-страницы будет видно. Фреймфорк отобразит все необходимые компоненты.

Если кликнуть на какой-либо элемент с назначенным обработчиком событий, то маршрутизация между различными элементами будет происходит в браузере, а не на сервере. Можно сказать, что JS перехватывает запросы, а Vue.js обрабатывает их в браузере. Веб-сайт работает быстро и плавно [2].

Рендеринг на стороне сервера - рендеринг клиентского или универсального приложения в HTML на сервере (SSR). В SSR рендеринг выполняется сервером. Пользователь запрашивает сервер; сервер обрабатывает HTML, CSS и JavaScript и доставляет полностью загруженную страницу в браузер.

В отличие CSR, каждый последующий раз, когда пользователь совершает переход на другую страницу веб-сайта, процесс рендеринга повторяется. Браузер делает запросы к серверу для каждого перехода и каждого запроса.

Очевидно, что недостатками SSR являются потребность в больших ресурсах и задержка доставки контента пользователю. Это увеличивает время загрузки страницы по сравнению с SPA. Это связано с тем, что серверу приходится многократно отображать динамический контент.

Когда много пользователей пытаются получить доступ к ресурсам на сайте, сервер может не успевать обрабатывать запросы, в этом случае могут возникнуть ошибки подключения.

Представим, что HTML файл ниже размещается на сервере в качестве главной страницы.

```html
<!DOCTYPE html>
<html lang="en">
<head>
<title>Домашняя страница</title>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
    <a href="blog.html">Мой блог</a>
    <h1>Мой новый веб-сайт</h1>
    <p>Это домашняя страница</p>
</html>
```

Рис. 3 Пример базовой страницы HTML

Веб-клиент отправляет запрос серверу. Сервер возвращает ответ с запрошенным содержимым, в данном случае это ссылка, заголовок и текст.

Далее пользователь кликает на ссылку и загружается вкладка «Влог» как показано на (рис.4) ниже.

```html
<!DOCTYPE html>
<html>
<head>
    <title>Страница блога</title>
    <meta name="description" content="My awesome blog">
    <meta name="keywords" content="design blog, web blog">
</head>
<body>
    <a href="index.html">Назад на домашнюю страницу</a>
    <h3>Блог №1</h3>
    <small>Опубликовано 08.02.2022</small>
    <p></p>
</head>
<body>
</html>
```

Рис. 4 Пример отдельной страницы HTML одного веб-сайта

Вся страница «Влог» будет загружена с новым содержанием. Затем пользователь возвращается на домашнюю страницу. Сервер снова начнет загружать страницу заново, несмотря на то, что ранее она была загружена. Потребность в больших ресурсах сервера является основным и значимым недостатком SSR.

Результаты исследования. Результаты исследования будут определены посредством обозначения особенностей каждого описанного вида рендеринга – их плюсов и минусов.

Вид Рендеринга CSR

Плюсы:

Веб-сайты CRS работают очень быстро и плавно. Первая загрузка может быть медленной, однако после рендеринга всех элементов, другие запросы страницы выполняются практически мгновенно.

Быстрая навигация по веб-сайту.

Отлично подходит для веб-приложений.

Минусы:

В связи относительно долгой первичной загрузкой компонентов сайта, задержка может отразится на качестве взаимодействия с пользователями. Пользователь может не дождаться полной загрузки и покинуть сайт.

Одним из самых больших недостатков CSR является негативное влияние на поисковую систему веб-сайта. CSR использует JavaScript, это замедляет обработку поисковых запросов.

Алгоритмы поисковых систем сначала сканируют и индексируют HTML-файл. Поэтому JS контент может быть не обработан во время индексации, это в свою очередь приведет к частичному индексированию и повлияет на SEO.

Запрос при инициализации загружает страницы, стили, шаблоны и т.д., сильно нагружая технику пользователя.

Вид Рендеринга SSR

Плюсы:Когда дело доходит до поисковой оптимизации оптимальным вариантом будет использование SSR. Каждая страница отображается на сервере веб-сайта независимо. В качестве примера можно рассмотреть сайт-блог. Каждый пост в блоге представляет собой независимую страницу и загружается с сервера независимо. Таким образом, можно определять метатеги на основе содержимого страницы, что соответственно оптимизирует работу поисковой системы.

SSR позволяет сфокусировать контент страницы и сделать его релевантным для поисковых систем в социальных сетях. Таким образом, роботы Google, Yandex и других поисковых систем смогут учитывать производительность веб-страницы для повышения рейтинга ресурса.

Оборудование пользователя будет меньше нагружено — поскольку пользователю уже приходит ответ от сервера и нужно будет только вывести эту информацию, ничего не нужно будет рендерить на стороне клиента, браузере или ПК пользователя.

Минусы: при переходе на новую страницу сервер получает новые запросы, что приводит к чрезмерному потреблению его памяти и мощности.

TTFB (time to first byte) будет медленнее. Это одна из метрик производительности веб-страницы (см подробнее на стр), которая описывает время, которое прошло с момента отправления браузером запроса страницей до получения первого байта информации от сервера. Вместо того чтобы отправить практически пустой HTML документ со ссылками на Java Script (как в случае с отрисовкой на стороне клиента), серверу нужно будет какое-то время для подготовки HTML.

Может увеличится стоимости хостинга веб-сайта, так как при SSR увеличивается поток данных между сервером и браузер.

Заключение. Вид рендеринга CSR подходит, в случае если веб-сайт регулярно посещают большое количество пользователей. Например, популярные онлайн магазины, браузерные игры, социальные сети. Также если у приложения сложный пользовательский интерфейс и много динамического контента, который незначительно зависит от SEO.

Если же для веб-ресурса важен рейтинг в поисковых системах, то рекомендуется использовать SSR. При использовании SSR каждая страницы веб-сайта обслуживается сервером отдельно, соответственно можно обозначить мета теги максимально подходящие к контенту. В отличие от CSR, SSR имеет относительно низкую общую скорость рендеренга контента. Поэтому этот вариант обработки запросов подходит веб-сайтам, где нет потребности обработки запросов от тысяч пользователей.

Литература:

1. Хэнчетт Эрик, Листоун Бенджамин Vue.js в действии [Текст] // Издание на русском языке ООО Издательство «Питер» 2019. – с. 36
2. Ilya Grigorik High-Performance Browser Networking [Текст]// O’Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472. 2013. – с. 236.
3. Jason Miller и Addy Osmani Рендеринг в интернете // Сайт Google - https://developers.google.com – 2023. – 09 февраля**.**