



Лекция 6

SRA

Мартин Комитски



ЦРИТО
Центр Развития
ИТ-Образования



План на сегодня

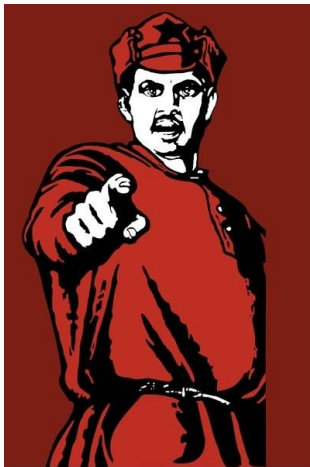


- Введение
- Теория
 - MPA
 - SPA
 - SPA vs. MPA
 - SSR
 - SEO
 - Fetch
 - Short polling
 - Long polling
 - Websocket
 - SSE
 - History API
 - React-router

Минутка бюрократии



- Внимание
- Отметки о посещении занятий
- Обратная связь о лекциях



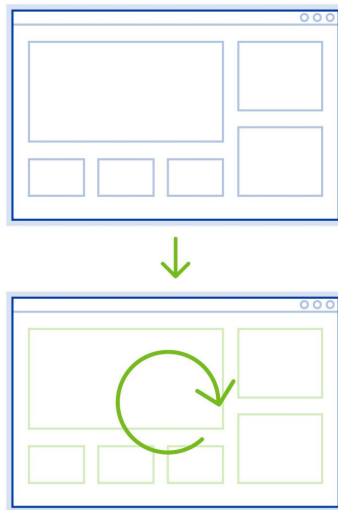


Как было раньше

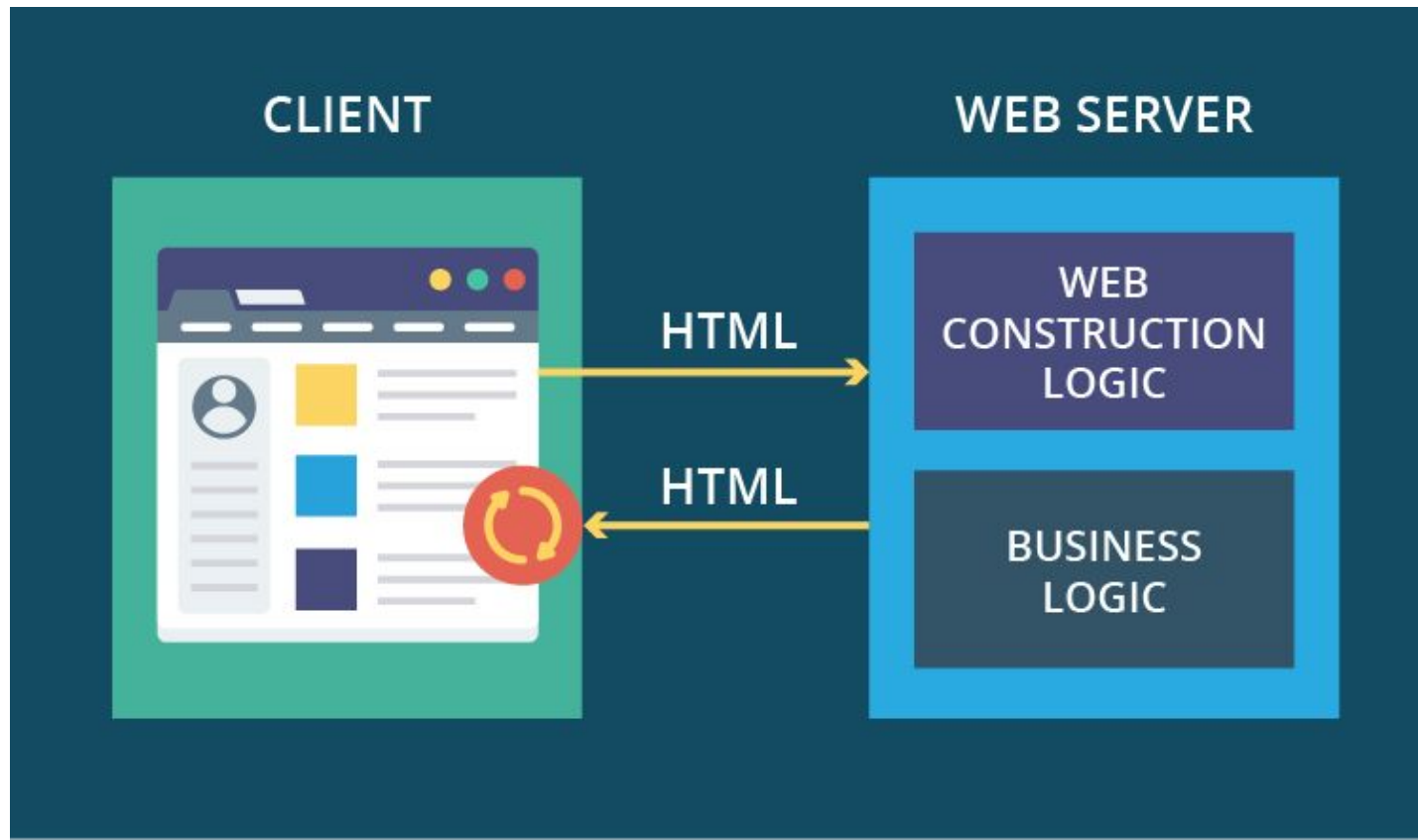


MPA (Multiple Page Application) - многостраничное приложение (*website*).

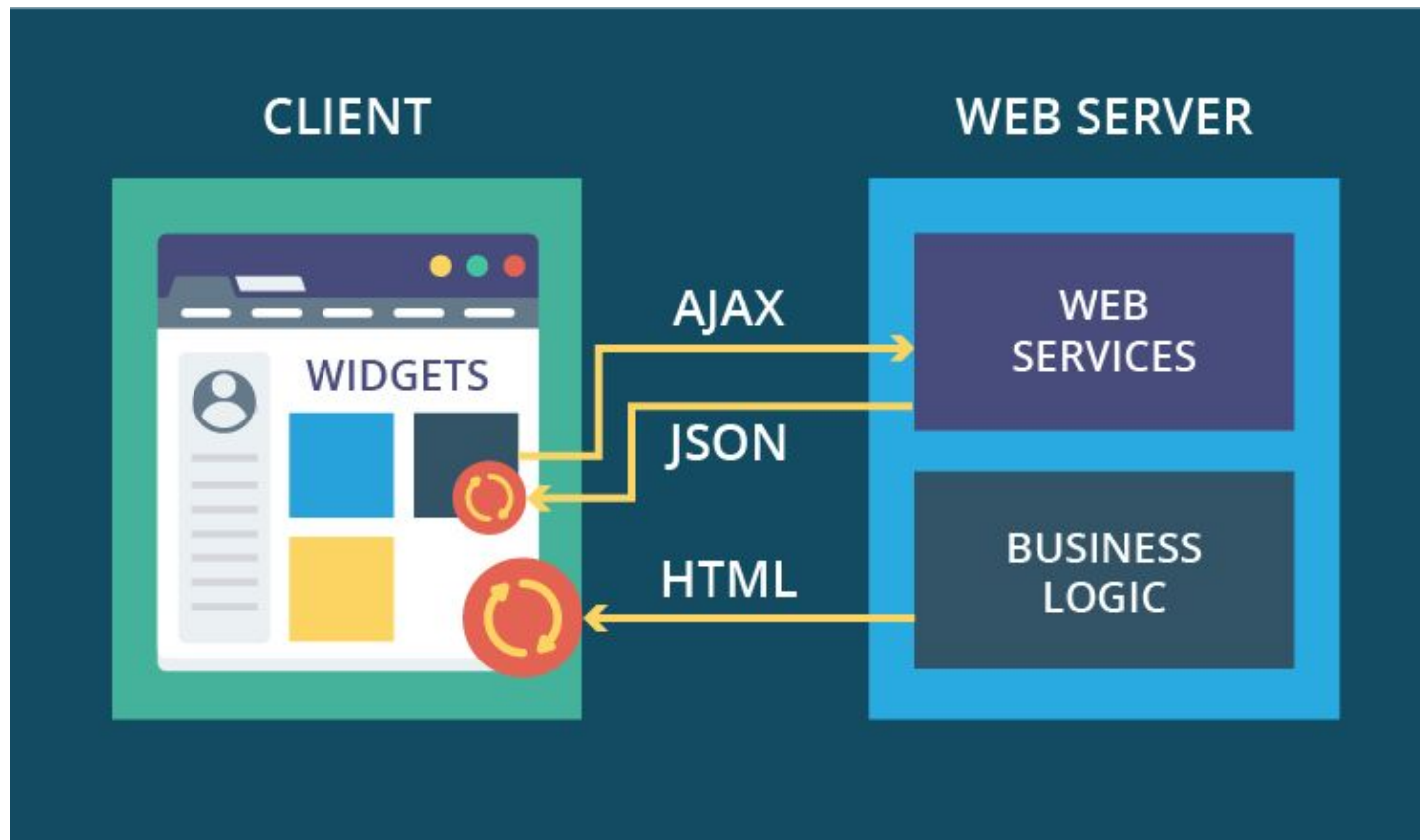
TRADITIONAL PAGE



МРА. Архитектура старого HTML сайта



МРА. Архитектура чуть менее старого сайта с виджетами



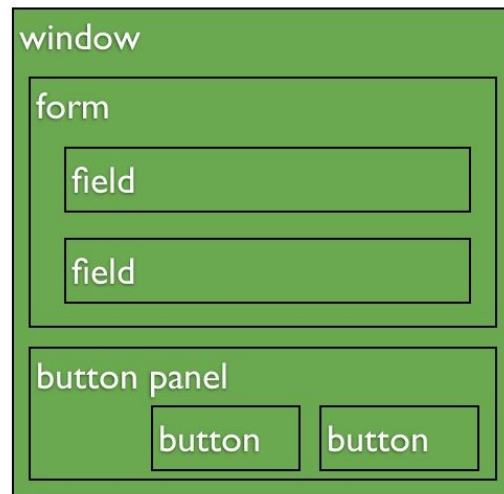
МРА. Архитектура. CMS



МРА. Подход к построению верстки. Монолит



Монолитный подход:



МРА. Плюсы и минусы



Плюсы

- Лучший выбор для простого старта
- Безопасность (сокрытие опасных и важных данных)
- SEO из коробки
- Могут работать без JS
- Стоимость разработки
- Множество готовых решений

Минусы

- Скорость работы приложения
- Скорость разработки
- Сложность
- Монолитность
- Обслуживание и обновление
- Высокая связность бекенда с фронтендом
- Легко начать делать плохие решения и костыли

МРА. Примеры фреймворков и CMS



Фреймворки и CMS, которые работают по принципу МРА

- Wordpress
- Joomla
- Drupal
- Django CMS
- Bitrix



VS



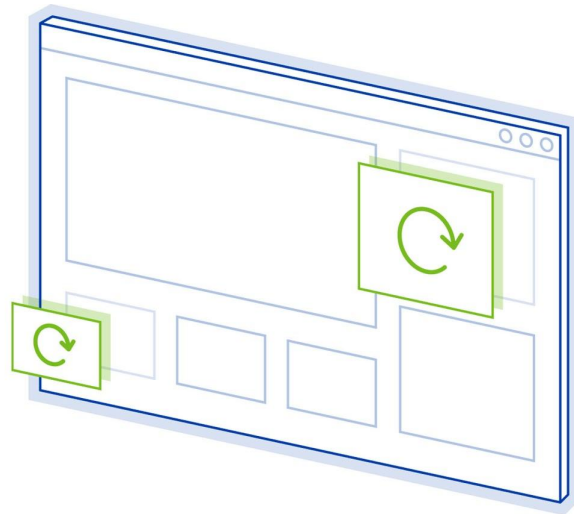
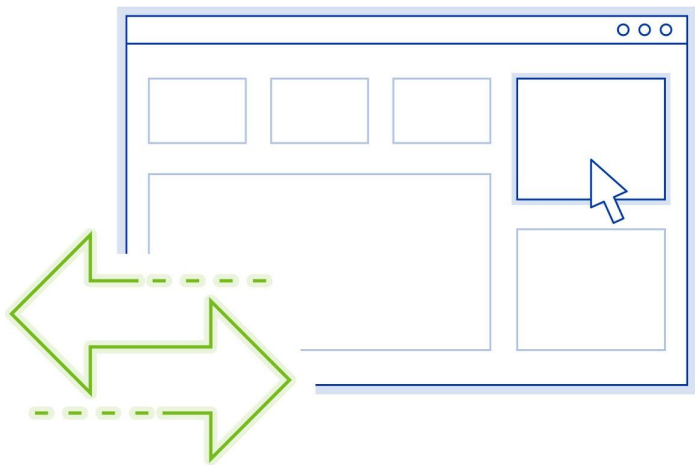


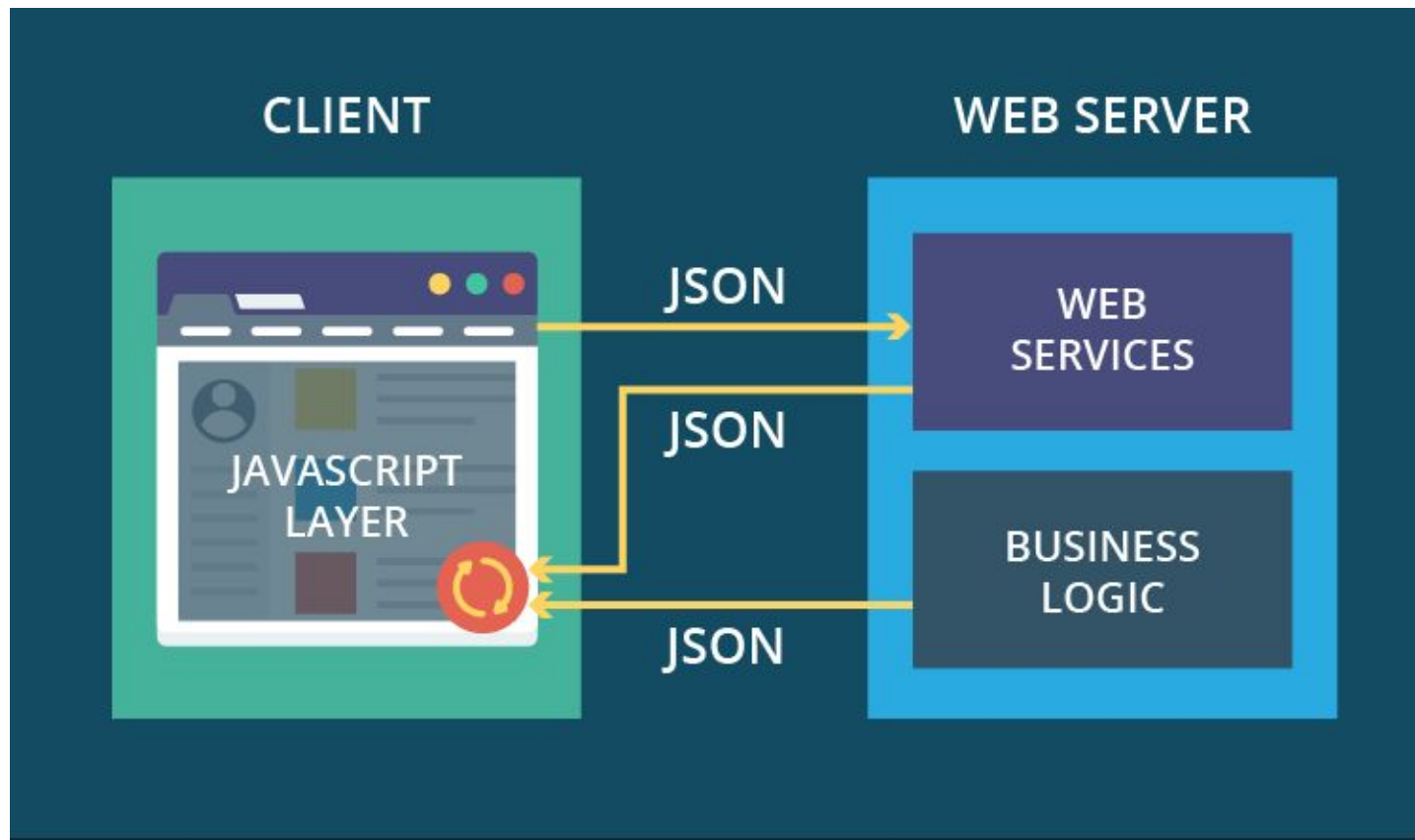
Как стало сейчас



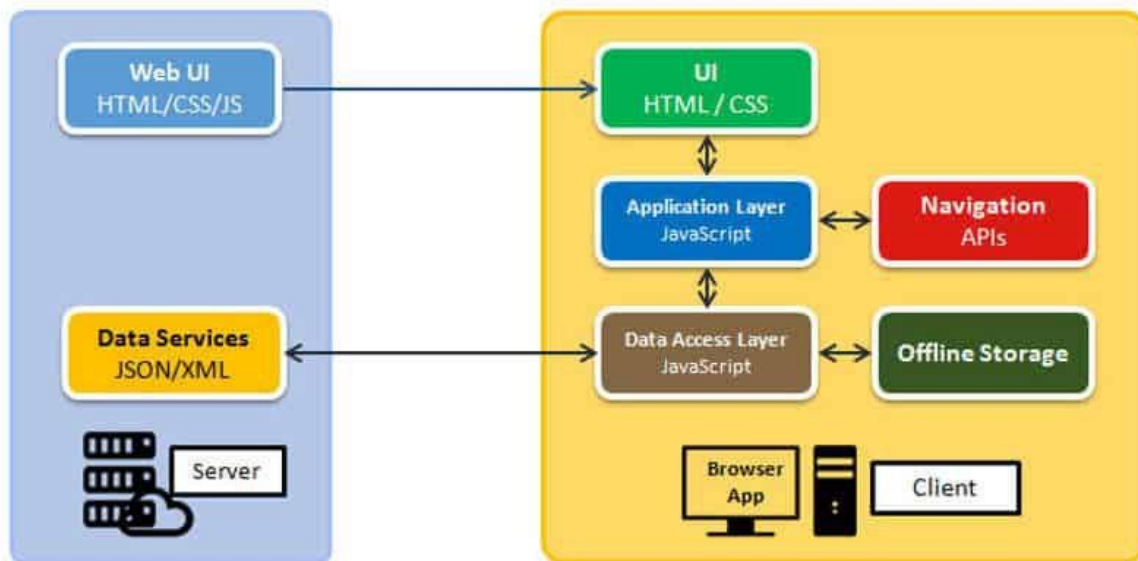


SPA (Single Page Application) - *одностраничное приложение (website).*





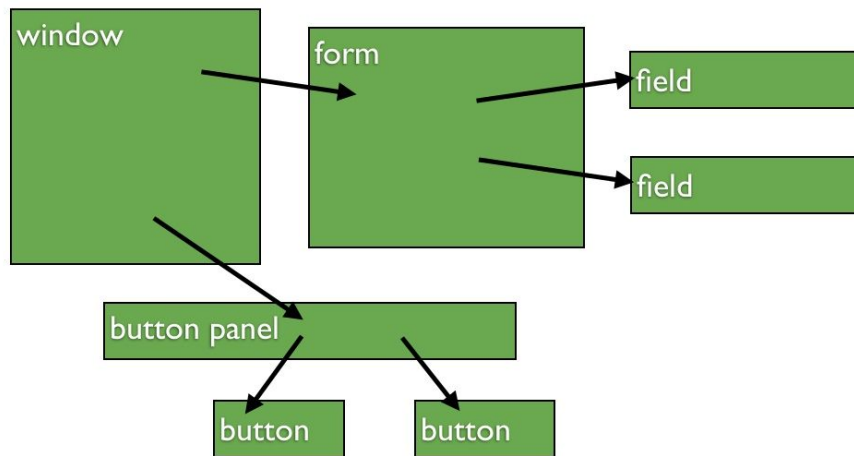
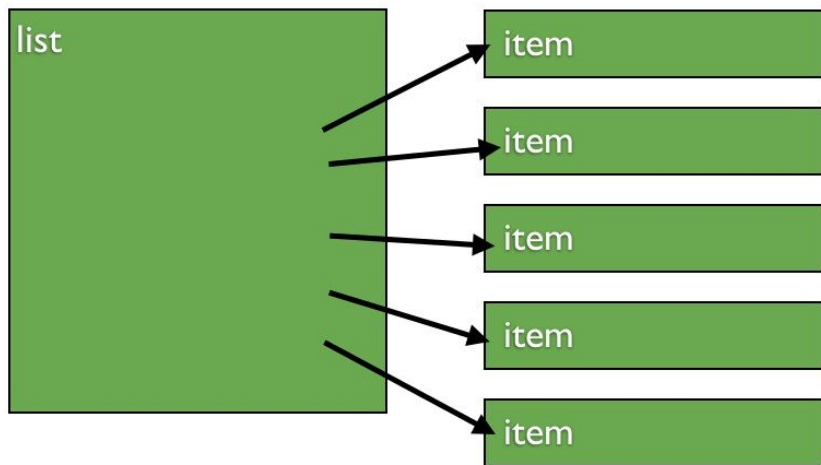
SPA. Архитектура



SPA. Подход к построению верстки. Компоненты



Компонентный подход:





- Компоненты могут быть достаточно сложны внутри, но они должны быть просты для использования снаружи
- Компонентом может быть вообще всё что угодно, что выполняет какую-то функцию в вашем приложении

Виды компонентов:

- **Dummy-компоненты** — компоненты, которые либо вообще не содержат никакой логики (чисто визуальные компоненты), либо содержат логику, которая глубоко инкапсулирована внутри компонента
- **Smart-компоненты** — компоненты, которые управляют множеством других компонентов, содержат в себе бизнес-логику и хранят какое-то состояние

SPA. Плюсы и минусы



Плюсы

- Скорость работы приложения
- Экономия трафика
- Возможности кеширования
- Возможность работы оффлайн
- Простота отладки
- Низкая связность бекенда с фронтендом
- Возможность сделать SPA мобильным приложением
- Кроссплатформенность (отличающиеся части в зависимости от UA)

Минусы

- Отсутствие SEO
- Нужно самому имитировать историю переходов
- Нужно сильнее следить за безопасностью и изначальными данными
- Разработка
- Возможность утечек памяти на клиенте

SPA vs. MPA

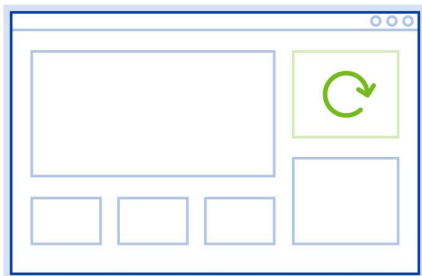
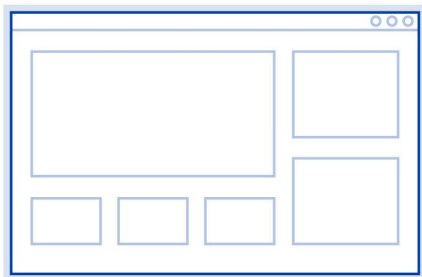


SPA vs. MPA



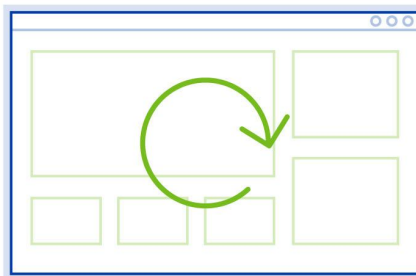
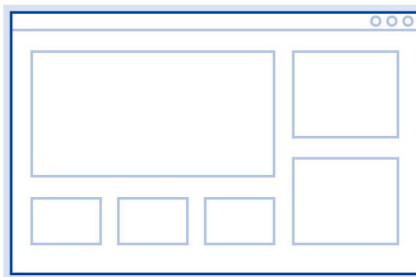
Пример SPA + SSR

SPA



Пример MPA

TRADITIONAL PAGE



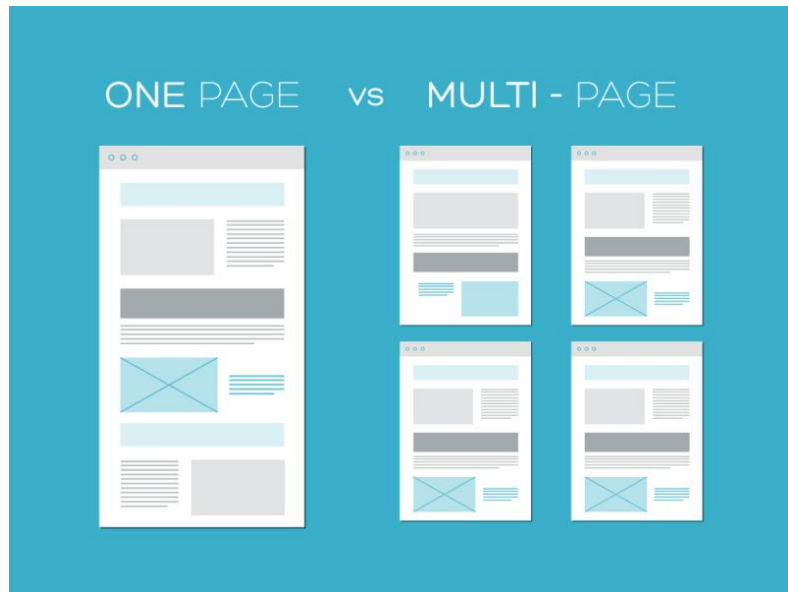
SPA vs. MPA



Что выбрать?

Нужно оценить:

- Цель проекта
- Специфику проекта
- Бюджет
- Квалификацию разработчиков
- Сроки
- Тенденции в разработке

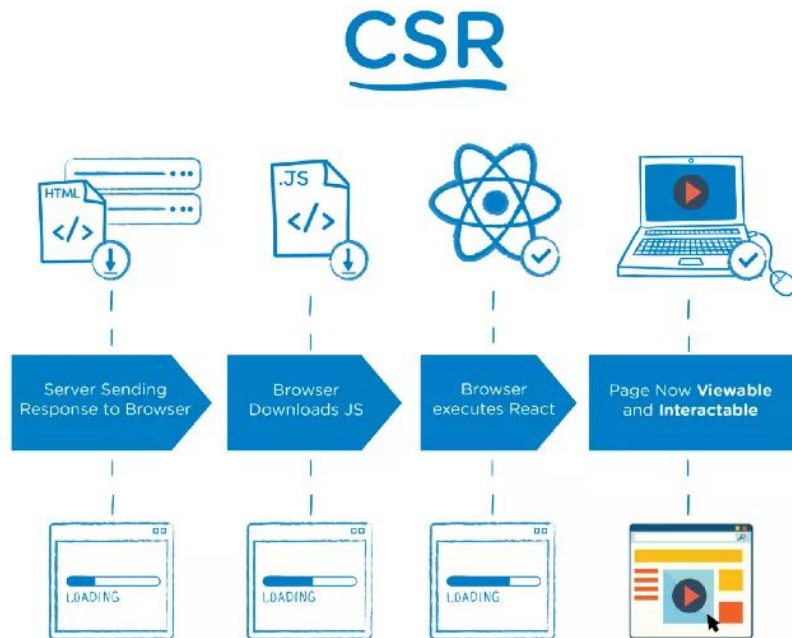




**Как становится сейчас
и как будет дальше**

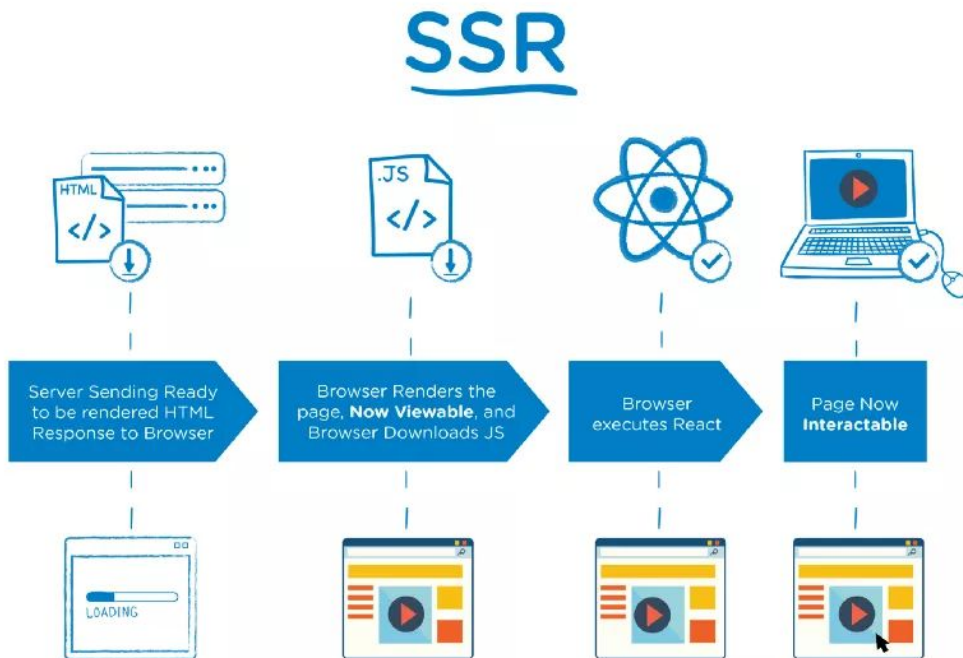


CSR (Client Side Rendering) - отрисовка динамического HTML на фронтенде.





SSR (Server Side Rendering) - отрисовка динамического HTML на бекенде.





SEO (Search Engine Optimization, поисковая оптимизация) – это всестороннее развитие и продвижение сайта для его выхода на первые позиции в результатах выдачи поисковых систем (SERPs) по выбранным запросам с целью увеличения посещаемости и дальнейшего получения дохода.



SEO. Как работает





“
Перерыв!

Препода (с)



Виды коммуникаций с сервером



Fetch API предоставляет интерфейс JavaScript для работы с запросами и ответами *HTTP*. Он также предоставляет глобальный метод ***fetch()***, который позволяет легко и логично получать ресурсы по сети асинхронно.

- ***Promise*** возвращаемый вызовом ***fetch()*** не перейдет в состояние "отклонено" из-за ответа *HTTP*, который считается ошибкой, даже если ответ *HTTP* 404 или 500. Вместо этого, он будет выполнен нормально (с значением `false` в статусе `ok`) и будет отклонён только при сбое сети или если что-то помешало запросу выполниться.
- По умолчанию, ***fetch*** не будет отправлять или получать cookie файлы с сервера, в результате чего запросы будут осуществляться без проверки подлинности, что приведёт к неаутентифицированным запросам, если сайт полагается на проверку пользовательской сессии (для отправки cookie файлов в аргументе *init options* должно быть задано значение свойства *credentials* отличное от значения по умолчанию *omit*).

Fetch. Пример



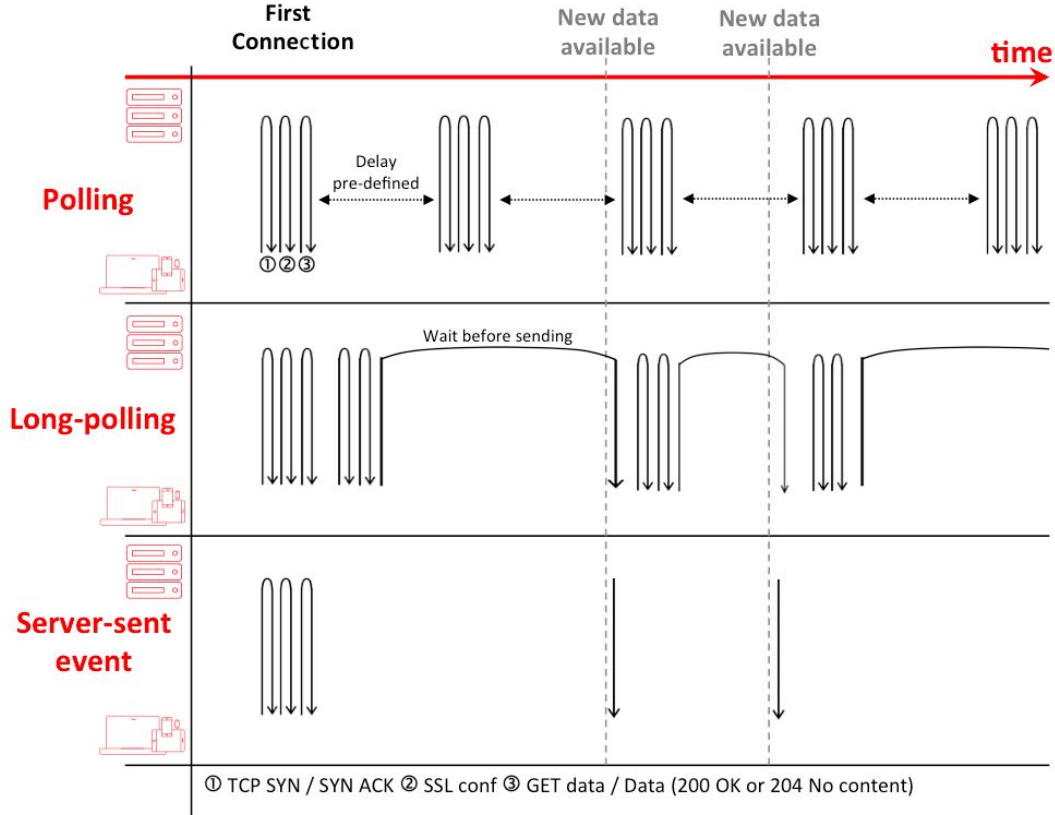
```
1. // Пример отправки POST запроса:
2.
3. postData('http://example.com/answer', {answer: 42})
4.   .then(data => console.log(JSON.stringify(data))) // JSON-строка полученная после вызова
   `response.json()`
5.   .catch(error => console.error(error));
6.
7. function postData(url = '', data = {}) {
8.   // Значения по умолчанию обозначены знаком *
9.   return fetch(url, {
10.     method: 'POST',          // *GET, POST, PUT, DELETE, etc.
11.     mode: 'cors',           // no-cors, cors, *same-origin
12.     cache: 'no-cache',      // *default, no-cache, reload, force-cache, only-if-cached
13.     credentials: 'same-origin', // include, *same-origin, omit
14.     headers: {
15.       'Content-Type': 'application/json',
16.       // 'Content-Type': 'application/x-www-form-urlencoded',
17.     },
18.     redirect: 'follow', // manual, *follow, error
19.     referrer: 'no-referrer', // no-referrer, *client
20.     body: JSON.stringify(data), // тип данных в body должен соответствовать значению
заголовка "Content-Type"
21.   })
22.   .then(response => response.json()); // парсит JSON ответ в Javascript объект
23. }
24.
```

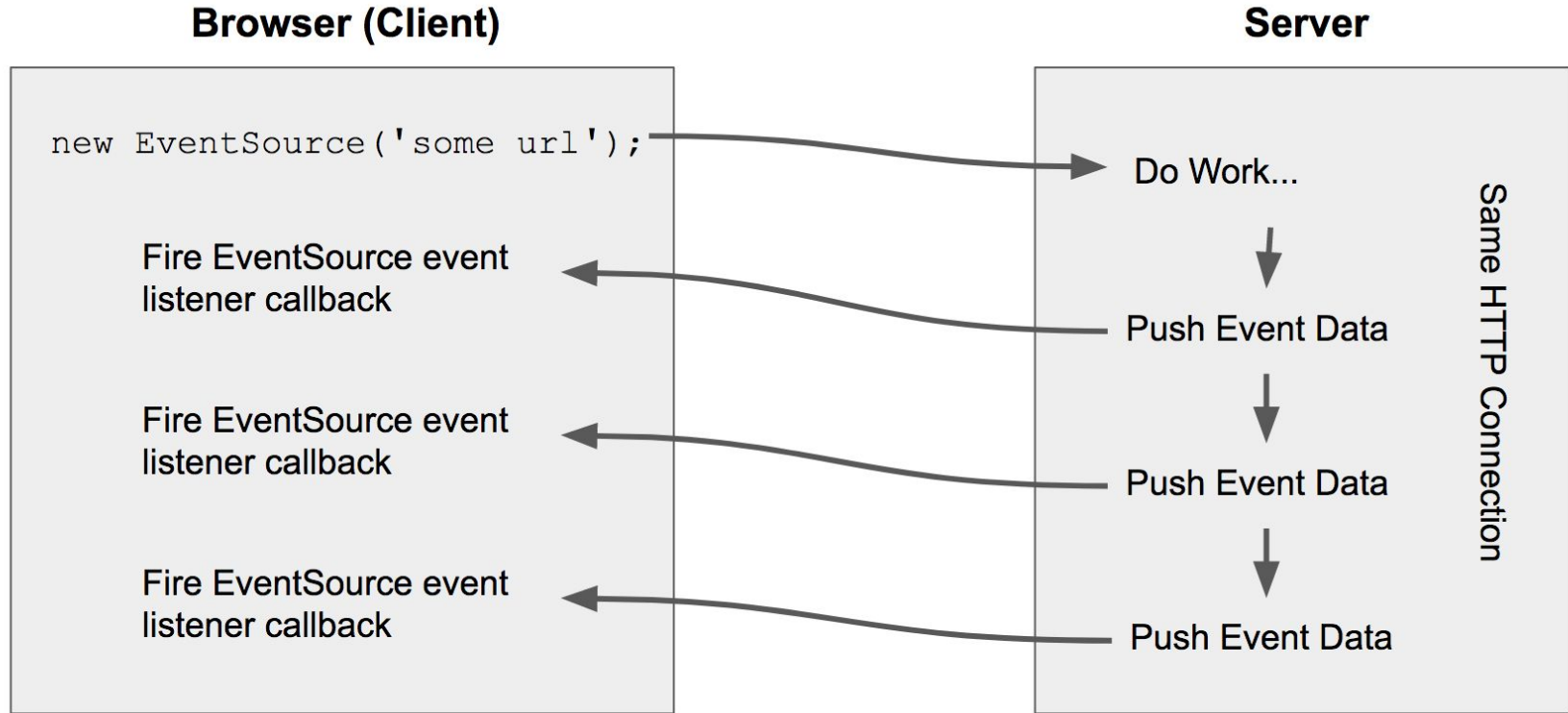
Виды коммуникаций с серверами



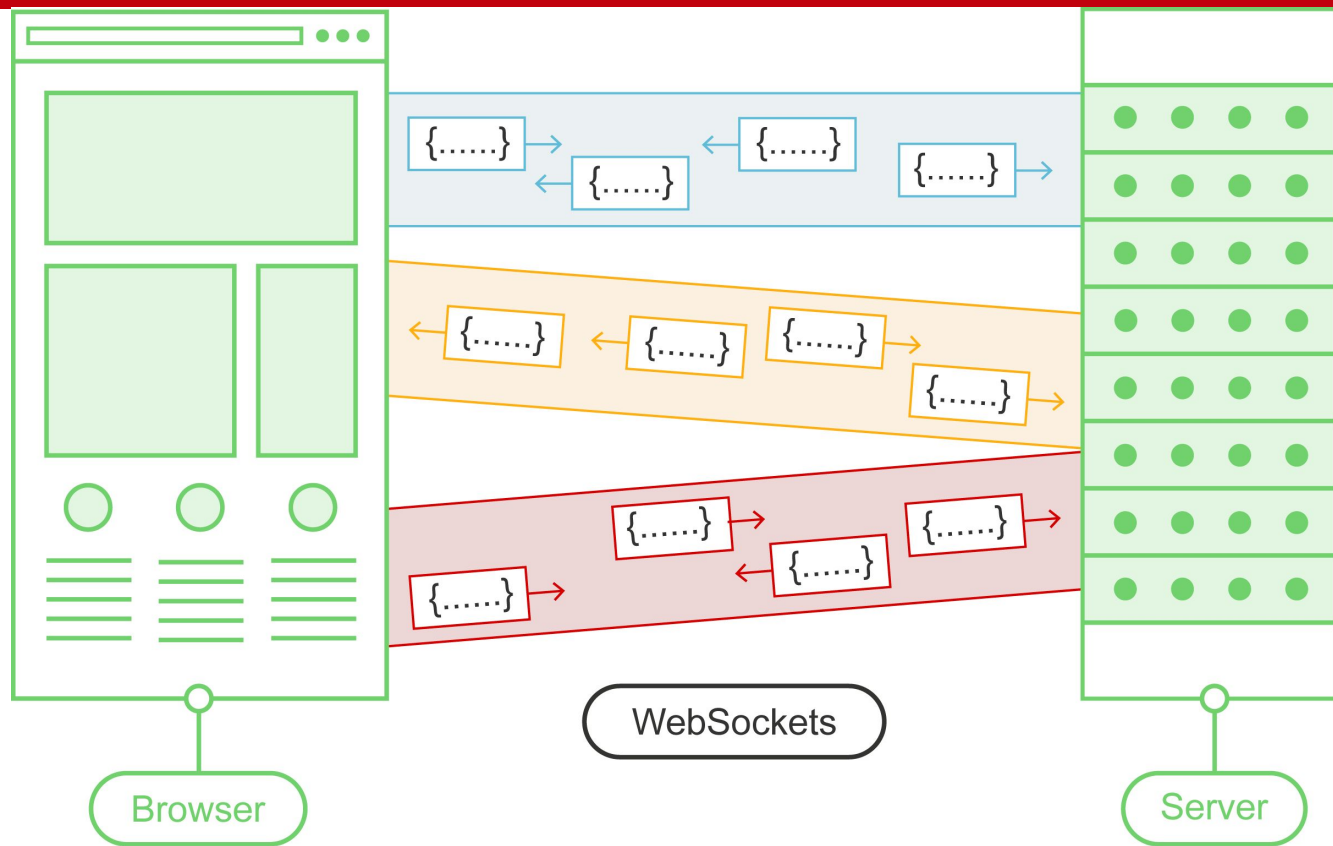
- Polling (client pull)
- Long polling (client pull)
- Websocket (server push)
- SSE (server push)

Polling/Long polling/SSE





Websocket





History API

History API

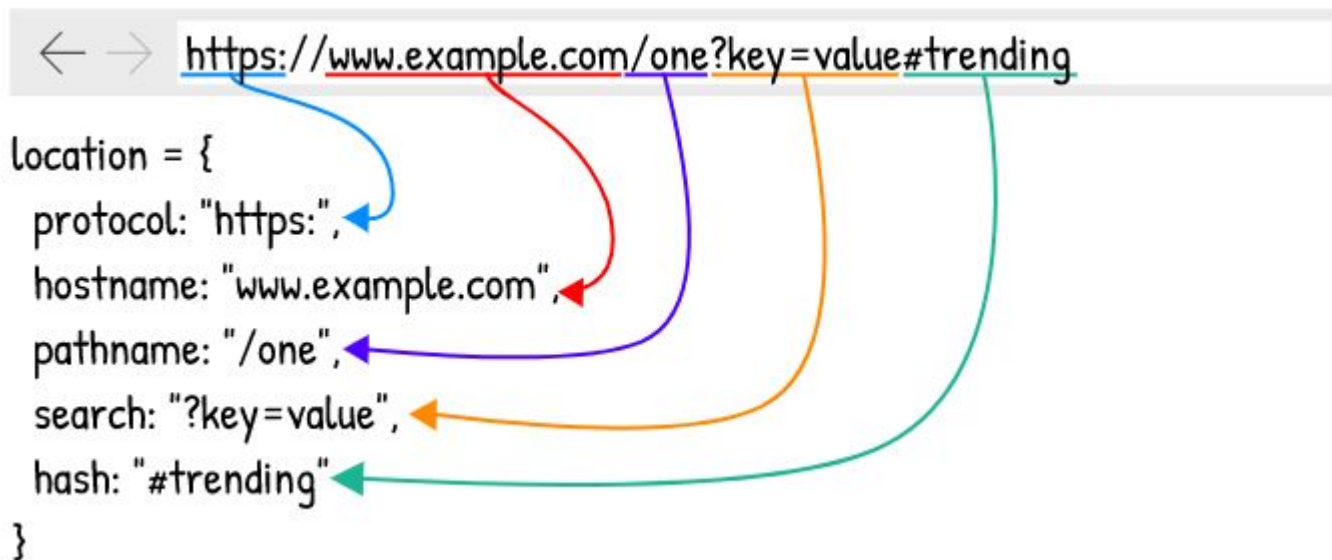


History API — браузерное API, позволяет манипулировать историей браузера в пределах сессии, а именно историей о посещённых страницах в пределах вкладки или фрейма, загруженного внутри страницы. Позволяет перемещаться по истории переходов, а также управлять содержимым адресной строки браузера

History API



Свойства объекта **window.location**



History API



Методы объекта **window.history**

```
window.history.back();
```

```
window.history.forward();
```

```
window.history.go();
```

```
window.history.pushState();
```

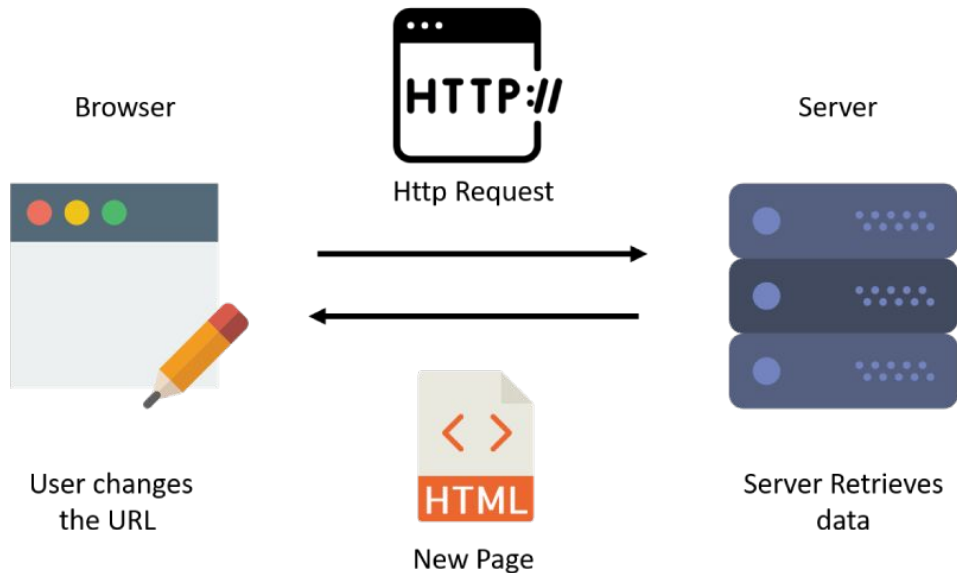
```
window.history.replaceState();
```

react-router



[react-router](#) - это библиотека, включающая в себя набор навигационных компонентов и логику, которая следит за состоянием **HistoryAPI**. С её помощью можно построить всю маршрутизацию **react** приложения.

Server-side маршрутизация:



react-router



```
1. import React from 'react'; import { BrowserRouter as Router, Switch, Route, Link } from 'react-router-dom';
2. export default function App() {
3.   return (
4.     <Router>
5.       <div>
6.         <nav>
7.           <ul>
8.             <li>
9.               <Link to="/">Home</Link>
10.            </li>
11.            <li>
12.              <Link to="/about">About</Link>
13.            </li>
14.            <li>
15.              <Link to="/users">Users</Link>
16.            </li>
17.          </ul>
18.        </nav>
19.        { /* A <Switch> looks through its children <Route>s and renders the first one that matches the current URL. */ }
20.        <Switch>
21.          <Route path="/about">
22.            <About /> // function About() { return <h2>About</h2>; }
23.          </Route>
24.          <Route path="/users">
25.            <Users /> // function Users() { return <h2>Users</h2>; }
26.          </Route>
27.          <Route path="/">
28.            <Home /> // function Home() { return <h2>Home</h2>; }
29.          </Route>
30.        </Switch>
31.      </div>
32.    </Router>
33.  );
34. }
35.
```

SPA. Полезные ссылки



- [polling-vs-sse-vs-websocket-how-to-choose-the-right-one](#)
- [next-js-vs-create-react-app](#)
- [single-page-apps-or-multiple-page-apps-whats-better-for-web-development](#)
- [single-page-application-vs-multiple-page-application](#)
- [single-page-applications](#)
- [enlightenment-of-single-page-website-single-page-wordpress-themes.html](#)
- [single-page-application-vs-multi-page-application](#)
- [Что такое SEO](#)
- [History API](#)
- [Working with History API](#)
- [React Router](#)
- [https://medium.com/webbdev/react-9cf527ed63a7](#)
- [Fetch API](#)
- [Using Fetch](#)

Домашнее задание № 6



1. Закрепить знания про SPA
2. Изучить [react-router](#)
3. Настроить клиентскую маршрутизацию для своих страниц
4. Сверстать страницу [профиля](#)

Срок сдачи

25 ноября



Спасибо за внимание!