

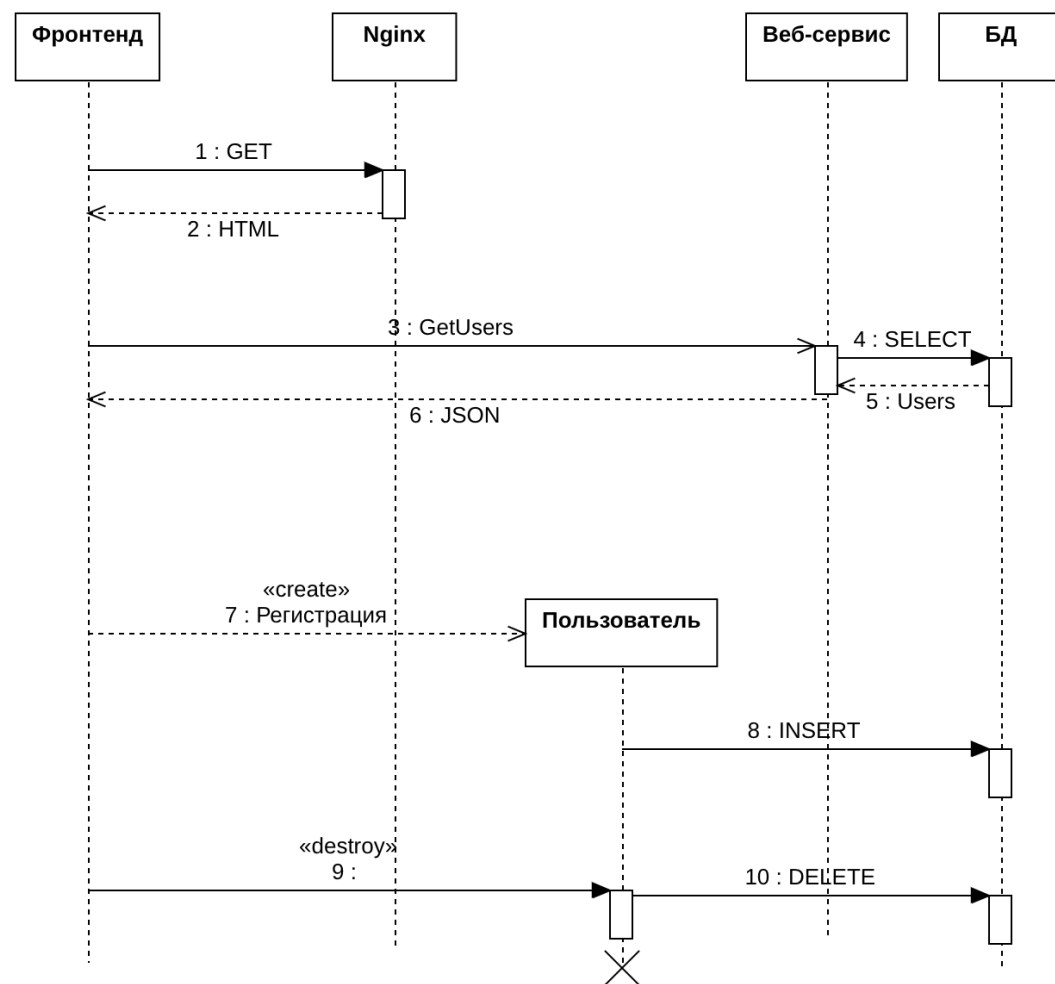
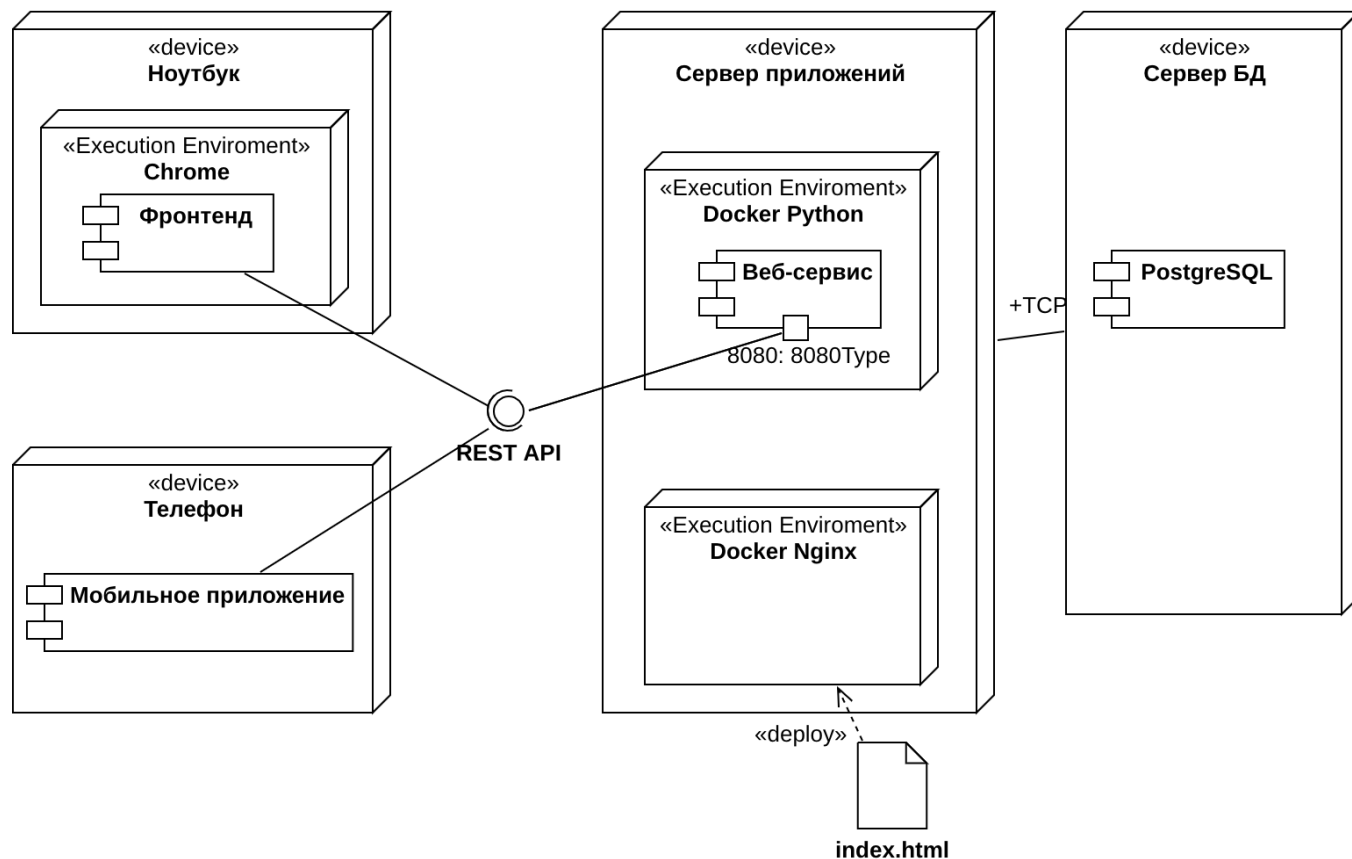
# Лекция 10

## Ајах

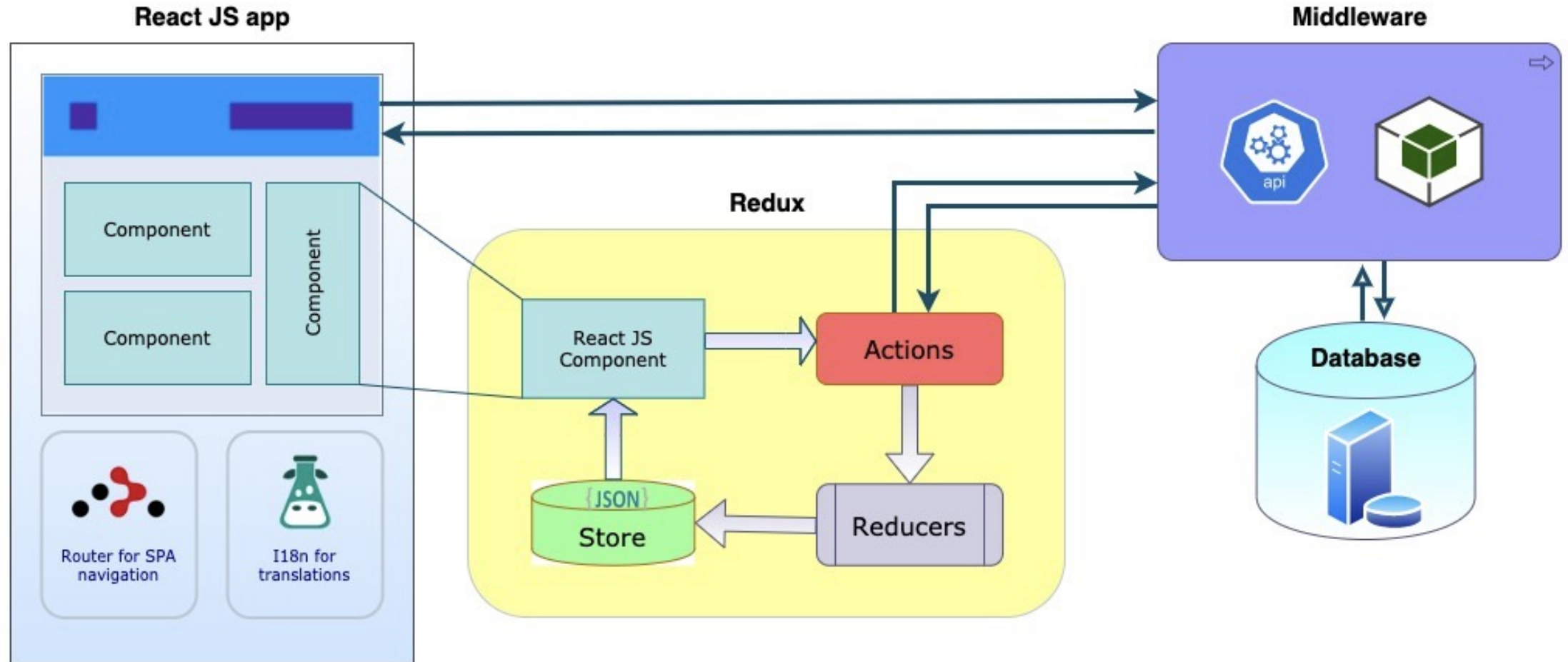
Разработка интернет приложений

Канев Антон Игоревич

# Трехзвенная архитектура. AJAX

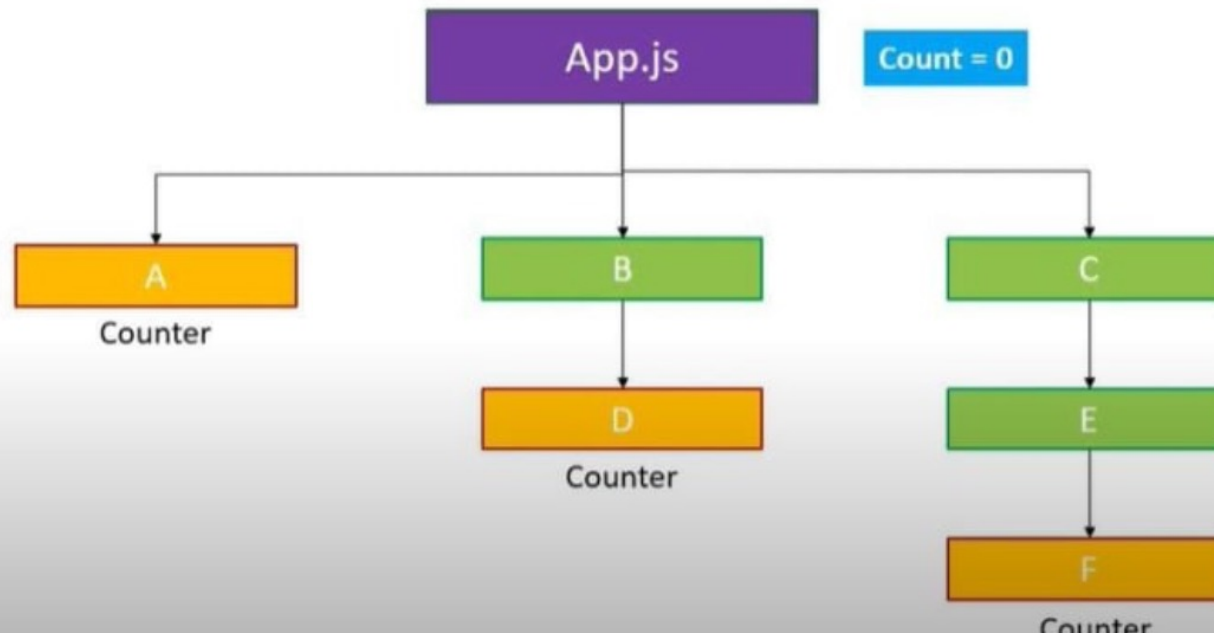


# React

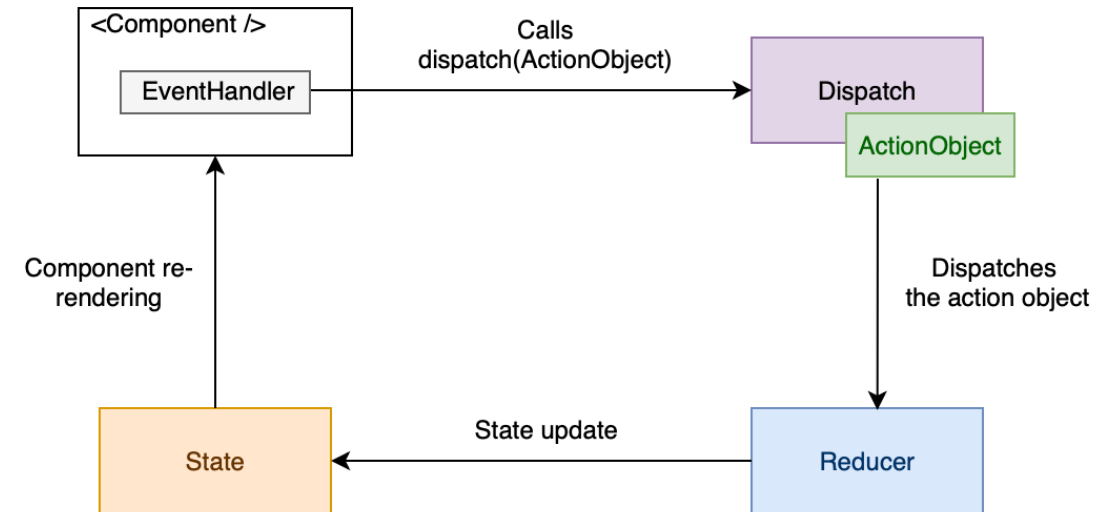


# useReducer + useContext

## useReducer with useContext



## useReducer()



# Middleware

- В наиболее общем случае, термин *middleware* часто используют для обозначения инфраструктуры: веб-серверов, серверов приложений, мониторов транзакций, программного обеспечения сервисных шин.

# Redux-middleware

- **Мидлвары** (middlewares) — это функции, которые последовательно вызываются в процессе обновления данных в хранилище.

Мидлвары используются в задачах:

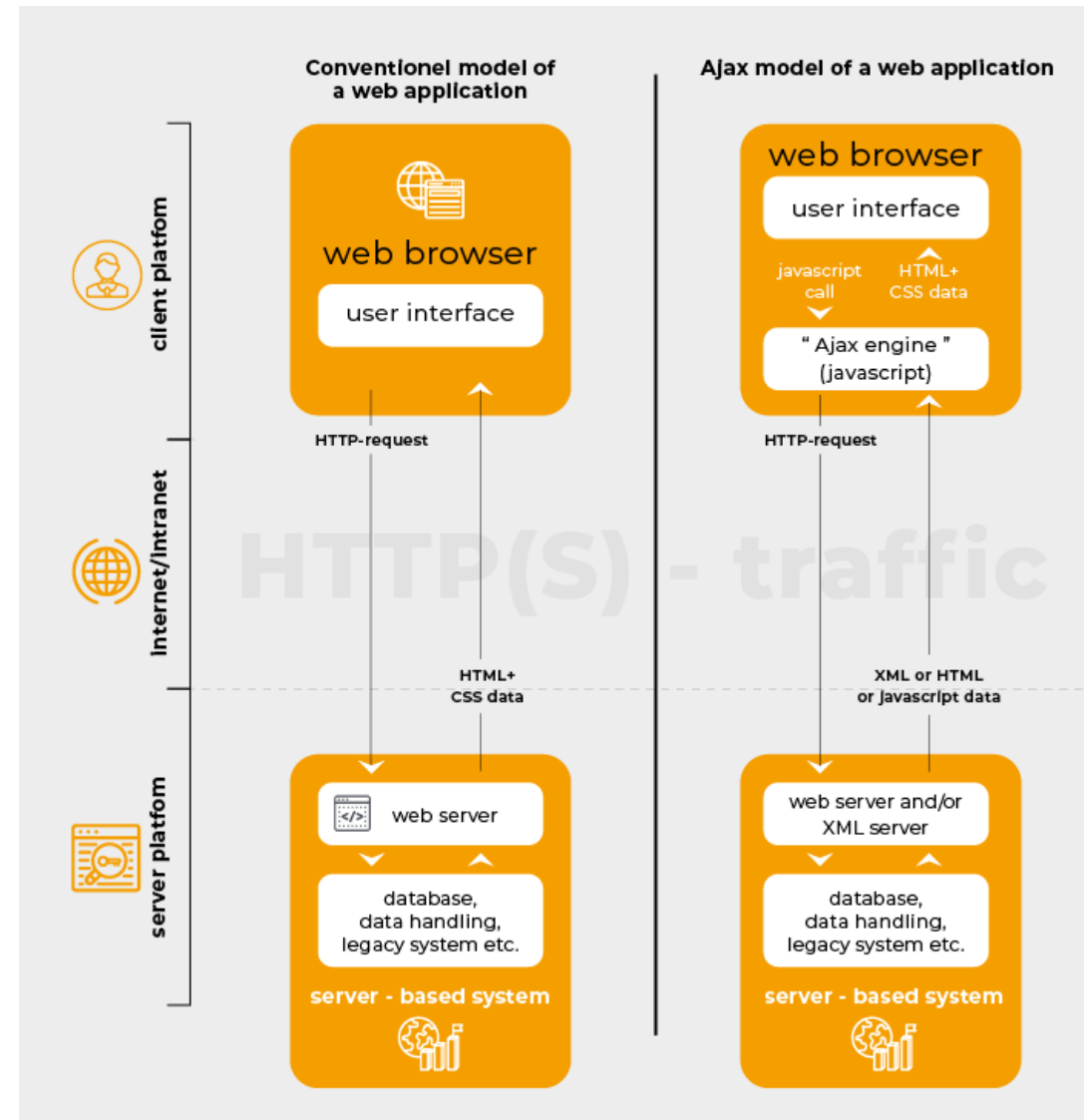
- Логирование
- Оповещение об ошибках
- Работа с асинхронным API
- Маршрутизация

```
const logger = store => next => action => {  
  let result;  
  console.groupCollapsed("dispatching", action.type);  
  console.log("prev state", store.getState());  
  console.log("action", action);  
  result = next(action);  
  console.log("next state", store.getState());  
  console.groupEnd();  
  return result;  
};
```

```
const middleware = applyMiddleware(logger);  
const store = createStore(reducers, middleware);
```

# AJAX

- **AJAX**, Ajax (Asynchronous Javascript and XML — «асинхронный JavaScript и XML») — подход к построению интерактивных пользовательских интерфейсов веб-приложений, заключающийся в «фоновом» обмене данными браузера с веб-сервером.
- В результате при обновлении данных веб-страница не перезагружается полностью, и веб-приложения становятся быстрее и удобнее.
- **JSON-RPC** (JavaScript Object Notation Remote Procedure Call — JSON-вызов удалённых процедур) — протокол удалённого вызова процедур, использующий JSON для кодирования сообщений.



# XMLHttpRequest

- **XMLHttpRequest** (XMLHTTP, XHR) — API, доступный в скриптовых языках браузеров, таких как JavaScript.
- Использует запросы HTTP или HTTPS напрямую к веб-серверу и загружает данные ответа сервера напрямую в вызывающий скрипт.

- Информация может передаваться в любом текстовом формате, например, в XML, HTML или JSON. Позволяет осуществлять HTTP-запросы к серверу без перезагрузки страницы.

```
var http_request = new XMLHttpRequest();
http_request.onreadystatechange = function () {
    if (http_request.readyState !== 4)
        return;

    if (http_request.status !== 200)
        throw new Error('request was defeated');

    do_something_with_object(JSON.parse(http_request.responseText));
    http_request = null;
};
http_request.open("GET", url, true);
http_request.send(null);
```

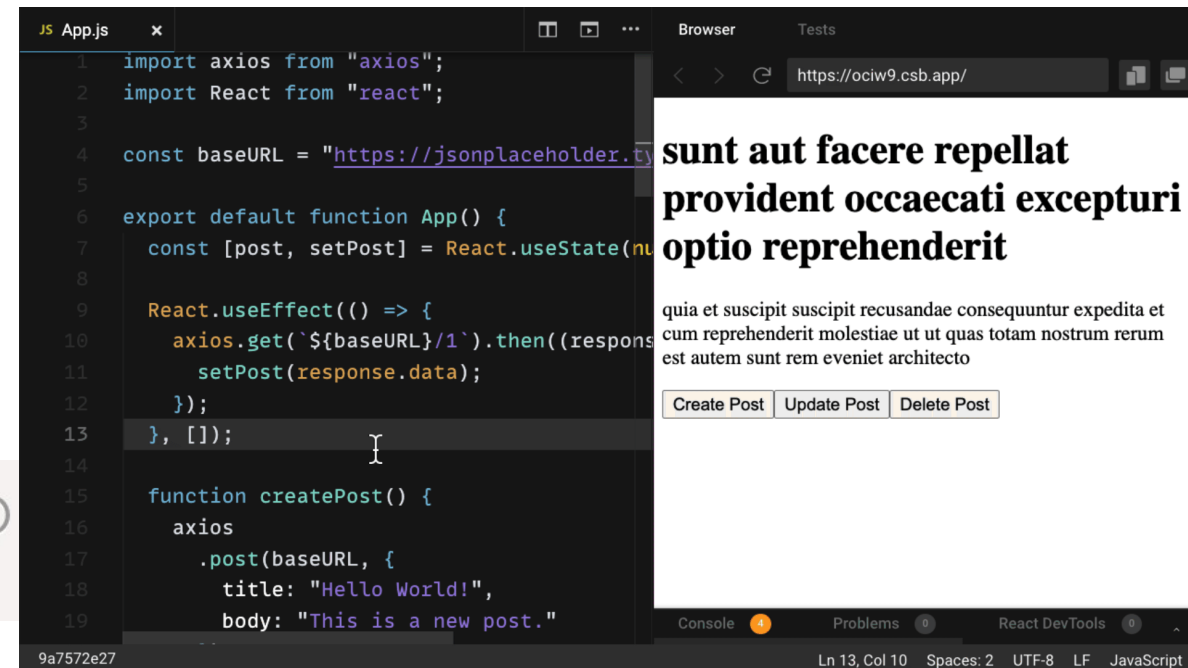


# Axios vs fetch

- Fetch — нативный низкоуровневый JavaScript интерфейс для выполнения HTTP-запросов с использованием обещаний через глобальный метод `fetch()`.
- Axios — JavaScript-библиотека, основанная на обещаниях, для выполнения HTTP-запросов.

```
fetch('https://jsonplaceholder.typicode.com/posts/1')  
  .then(response => response.json())  
  .then(json => console.log(json))
```

```
axios.get('https://jsonplaceholder.typicode.com/posts/1')  
  .then(response => console.log(response));
```



# Обработка ошибок

- Axios обрабатывает ошибки логично.
- Если сервер вернул ответ с HTTP статусом ошибки (например 404 или 500), то обещание будет отвергнуто.

```
fetch(url)
  .then(response => {
    return response.json().then(data => {
      if (response.ok) {
        return data;
      } else {
        return Promise.reject({status: response.status, data});
      }
    });
  })
  .then(result => console.log('success:', result))
  .catch(error => console.log('error:', error));
```

```
axios.get('https://jsonplaceholder.typicode.com/posts/1')
  .then(response => console.log(response));
```

# POST

- С axios всё просто, а с fetch уже не так:
- JSON обязан быть преобразован в строку, а заголовок Content-Type должен указывать, что отправляются JSON данные,
- иначе сервер будет рассматривать их как строку.

```
axios.post('/user', {  
  firstName: 'Fred',  
  lastName: 'Flintstone'  
});
```

```
fetch('/user', {  
  method: 'POST',  
  headers: {  
    'Content-Type': 'application/json'  
  },  
  body: JSON.stringify({  
    firstName: 'Fred',  
    lastName: 'Flintstone'  
  })  
});
```

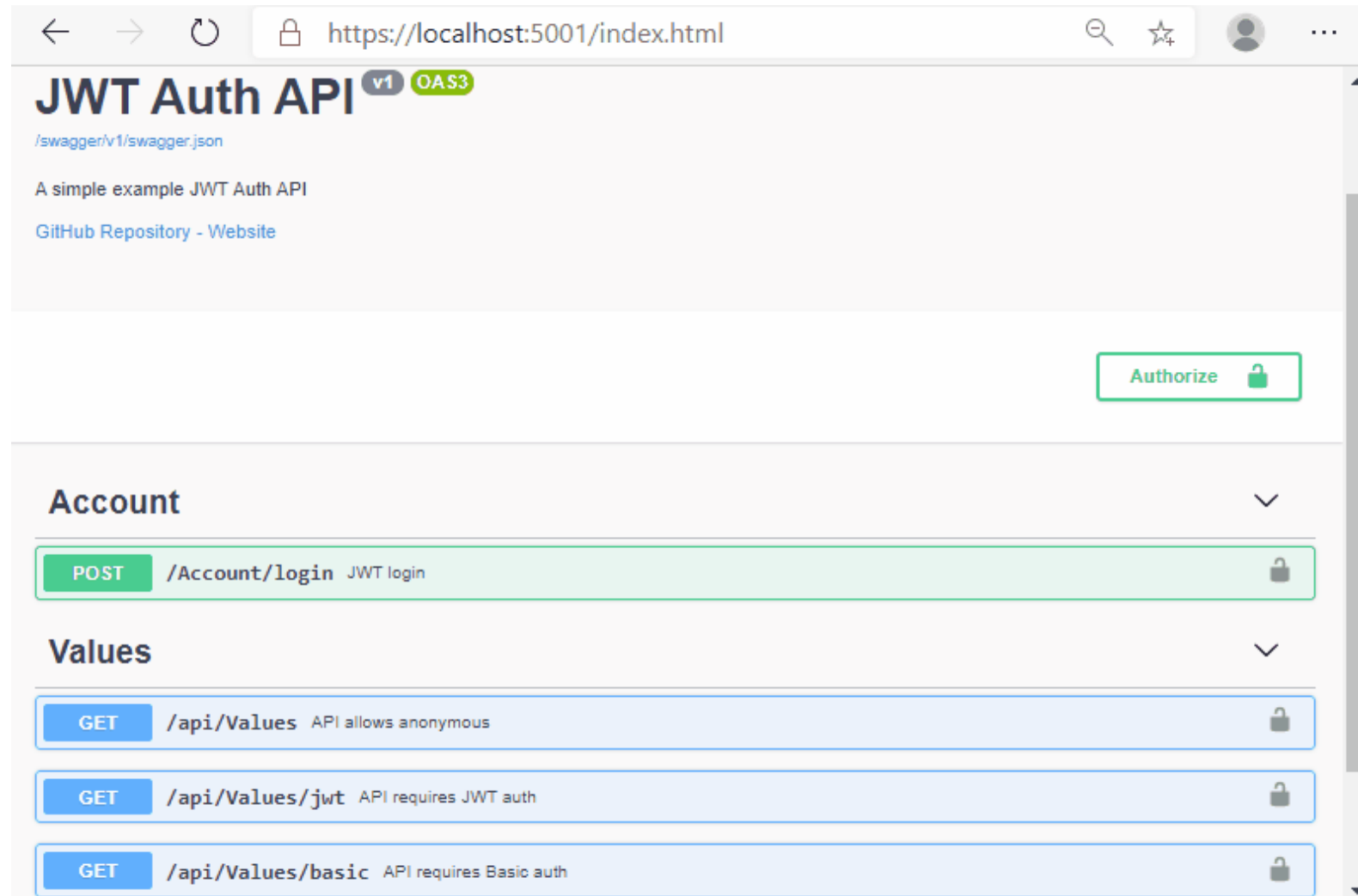
# Базовые значения для запросов

- fetch это явный API, вы ничего не получаете, если об этом не просите.
- Если используется аутентификация, основанная на сохранении сессии пользователя, то надо явно указывать куку.
- Если сервер расположен на поддомене, то надо явно прописывать CORS.
- Эти опции надо прописывать для всех вызовов сервера и у fetch нет механизма для установки значений по-умолчанию, а у axios есть.

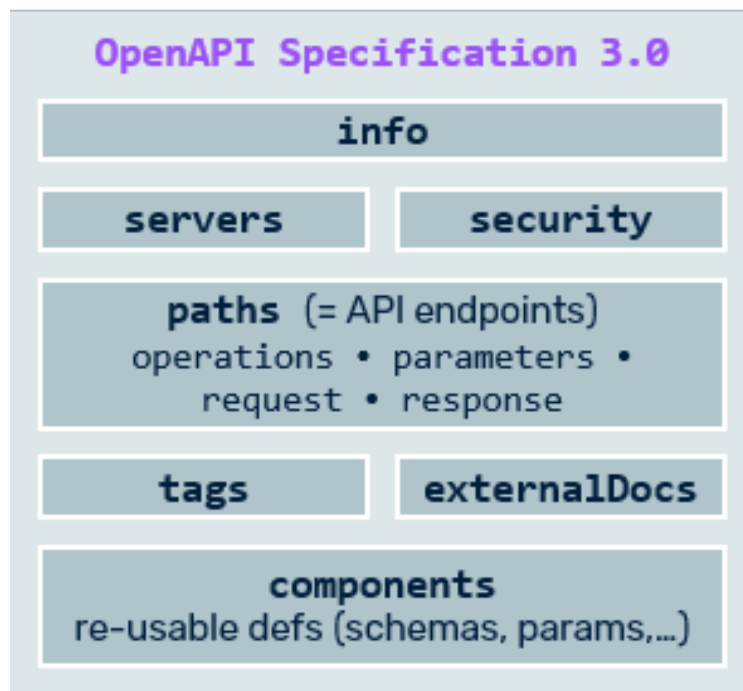
```
axios.defaults.baseURL = 'https://api.example.com';  
axios.defaults.headers.common['Accept'] = 'application/json';  
axios.defaults.headers.post['Content-Type'] = 'application/json';
```

# Swagger

- Swagger – это фреймворк для спецификации RESTful API.
- Его прелесть заключается в том, что он дает возможность интерактивно просматривать спецификацию
- и отправлять запросы – так называемый Swagger UI



# OpenAPI



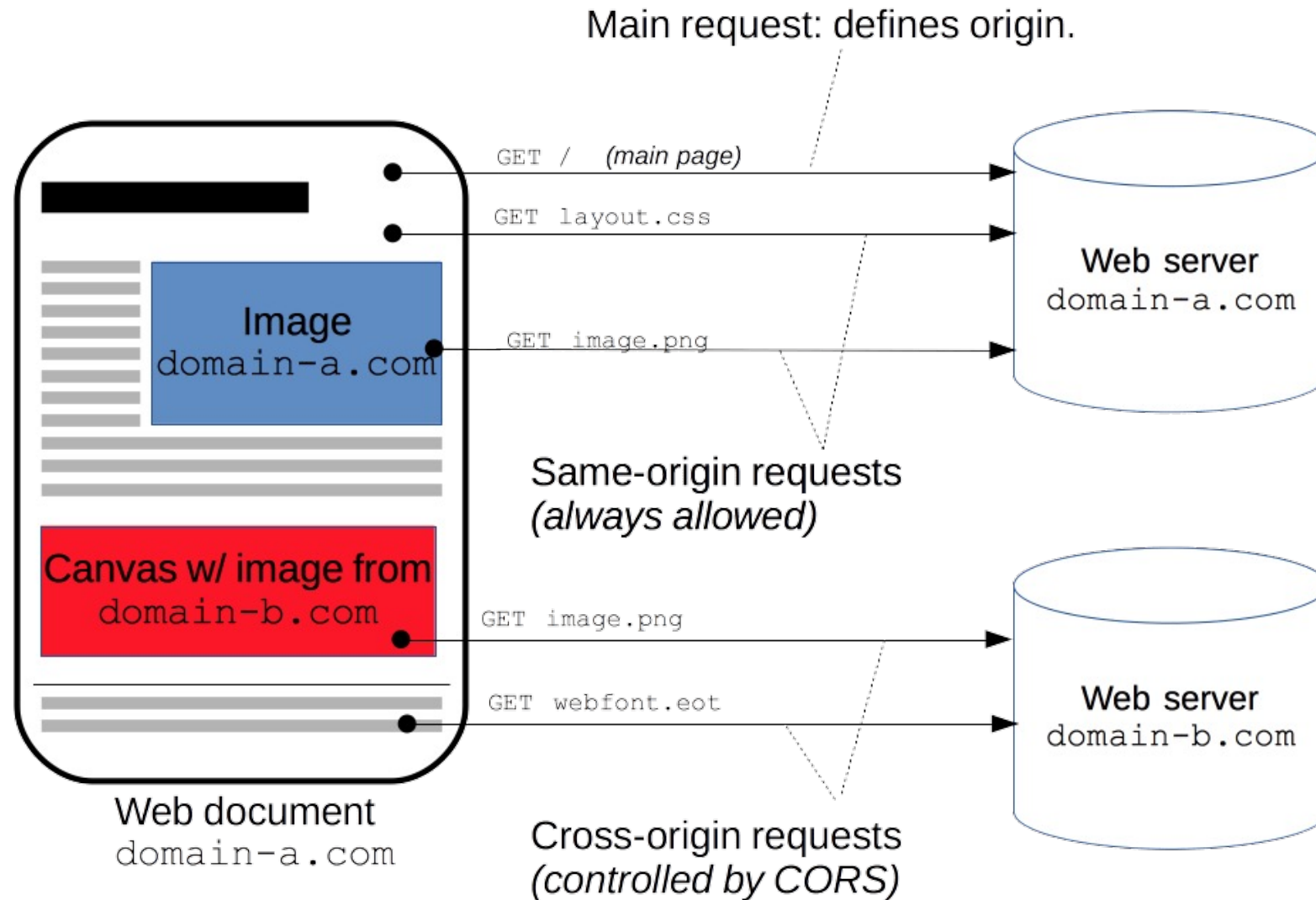
serialized in either  
JSON or YAML

HTTP, OAuth2, JWT<sup>3</sup>



■ synchronous calls  
■ call backs

- **The OpenAPI Specification** (изначально известная как *Swagger Specification*)
- формализованная спецификация и экосистема множества инструментов, предоставляющая интерфейс между front-end системами, кодом библиотек низкого уровня и коммерческими решениями в виде API.
- Вместе с тем, спецификация построена таким образом, что не зависит от языков программирования, и удобна в использовании как человеком, так и машиной

# Cors



# MUI

<p>Small <u>Medium</u> Large</p> <div> ADD TO CART</div> <p>Button</p>	<p><u>Standard</u> Filled Outlined</p> <div> Check out this alert!</div> <p>Alert</p>	<p><u>Outlined</u> Standard Filled</p> <div><p>Username</p><p>Ultraviolet</p></div> <p>Text Field</p>						
<p>CLICK TO OPEN</p> <p>Menu</p>	<table><thead><tr><th>Dessert</th><th>Calories</th></tr></thead><tbody><tr><td>Frozen yoghurt</td><td>109</td></tr><tr><td>Cupcake</td><td>305</td></tr></tbody></table> <p>Table</p>	Dessert	Calories	Frozen yoghurt	109	Cupcake	305	<p><b>Want to see more?</b></p> <p>Check out the docs for details of the complete library.</p> <p><a href="#">Learn more</a> &gt;</p>
Dessert	Calories							
Frozen yoghurt	109							
Cupcake	305							



# Figma

