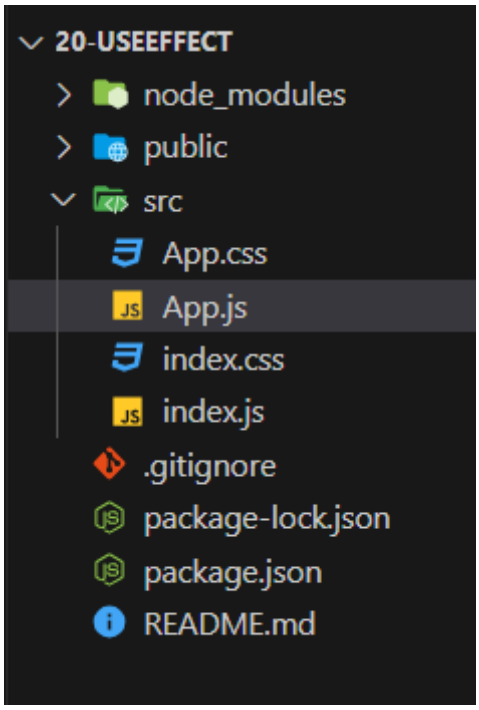
Проект по использованию fetch в компонентах

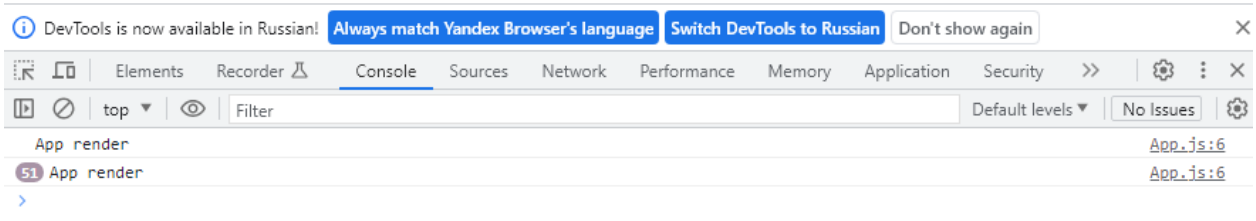



```
2   import './App.css';
3
4   function App() {
5     fetch('https://jsonplaceholder.typicode.com/todos/1')
6     .then((response) => response.json())
7     .then((json) => console.log(json));
8
9     return <div className="App"></div>;
10  }
11
12  export default App;
```

App.js

src > App.js > ...

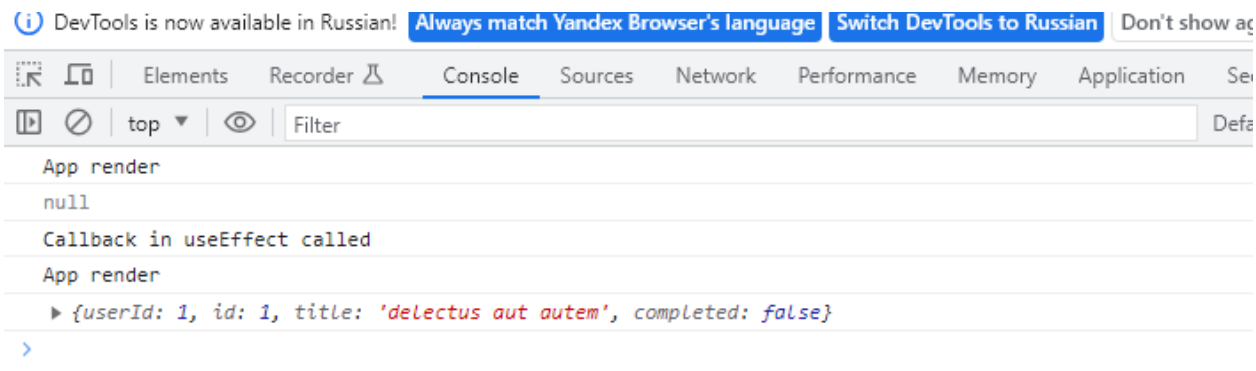
```
1  import { useState } from 'react';
2  import './App.css';
3
4  function App() {
5    const [todo, setTodo] = useState(null);
6    console.log('App render');
7
8    fetch('https://jsonplaceholder.typicode.com/todos/1')
9      .then((response) => response.json())
10     .then((json) => setTodo(json));
11
12   return <div className="App"></div>;
13 }
14
15 export default App;
16
```



App.js

src > App.js > App > useEffect() callback

```
1  import { useState } from 'react';
2  import './App.css';
3  import { useEffect } from 'react';
4
5  function App() {
6    const [todo, setTodo] = useState(null);
7    useEffect(() => {
8      console.log('Callback in useEffect called');
9      fetch('https://jsonplaceholder.typicode.com/todos/1')
10       .then((response) => response.json())
11       .then((json) => setTodo(json));
12   }, []);
13   console.log('App render');
14   console.log(todo);
15
16   return <div className="App"></div>;
17 }
18
19 export default App;
20
```



[todo] снова запустит бесконечный цикл обращений к серверу

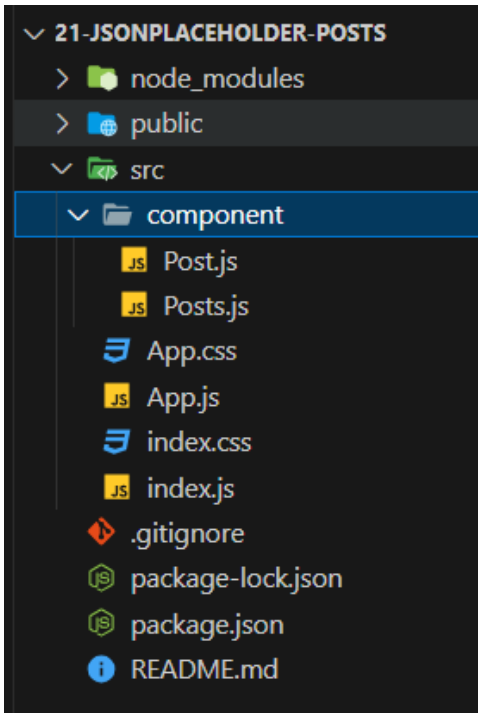
Выведем данные с удаленного сервера в интерфейс

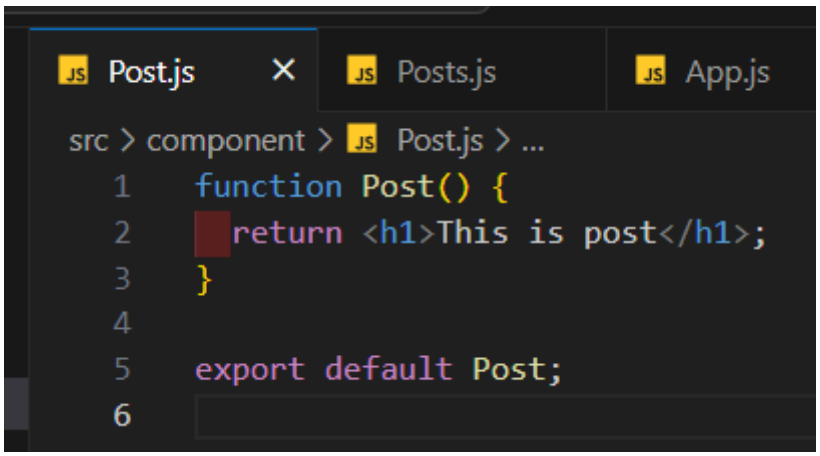
App.js

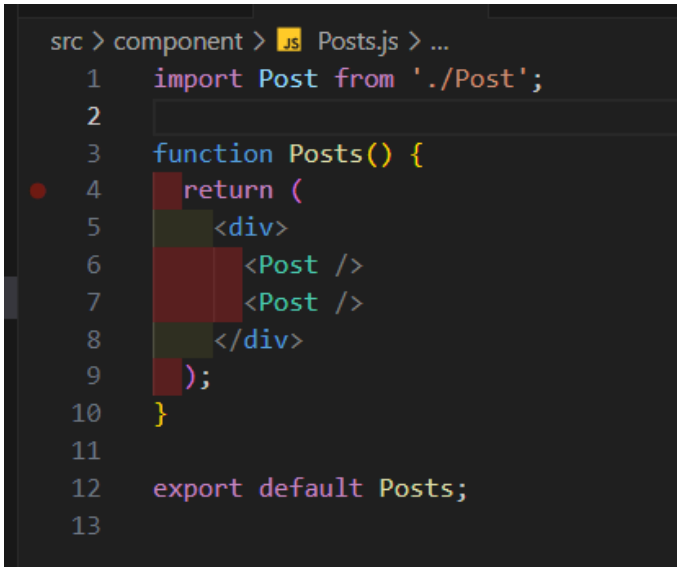
src > App.js > App > useEffect() callback

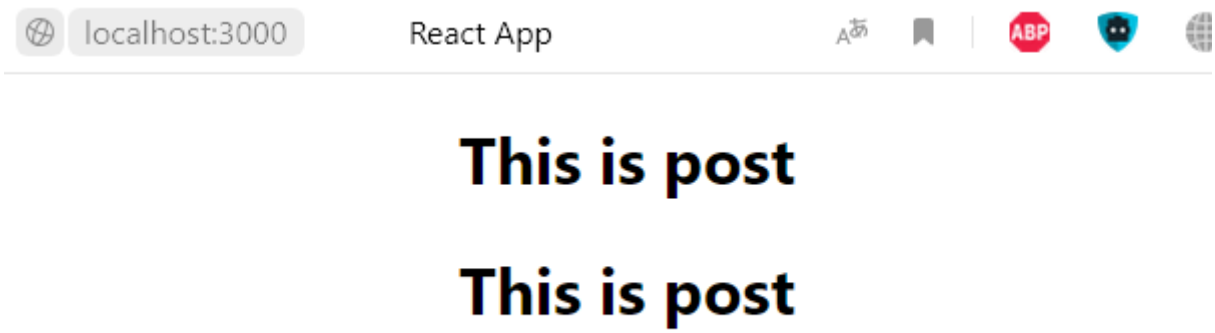
```
1  import { useState } from 'react';
2  import './App.css';
3  import { useEffect } from 'react';
4
5  function App() {
6    const [todo, setTodo] = useState(null);
7    useEffect(() => {
8      console.log('Callback in useEffect called');
9      fetch('https://jsonplaceholder.typicode.com/todos/3')
10       .then((response) => response.json())
11       .then((json) => setTodo(json));
12   }, []);
13   console.log('App render');
14   console.log(todo);
15
16   return <div className="App">{todo && <h1>{todo.title}</h1>}</
     div>;
17 }
18
19 export default App;
```

Проект с массивом постов



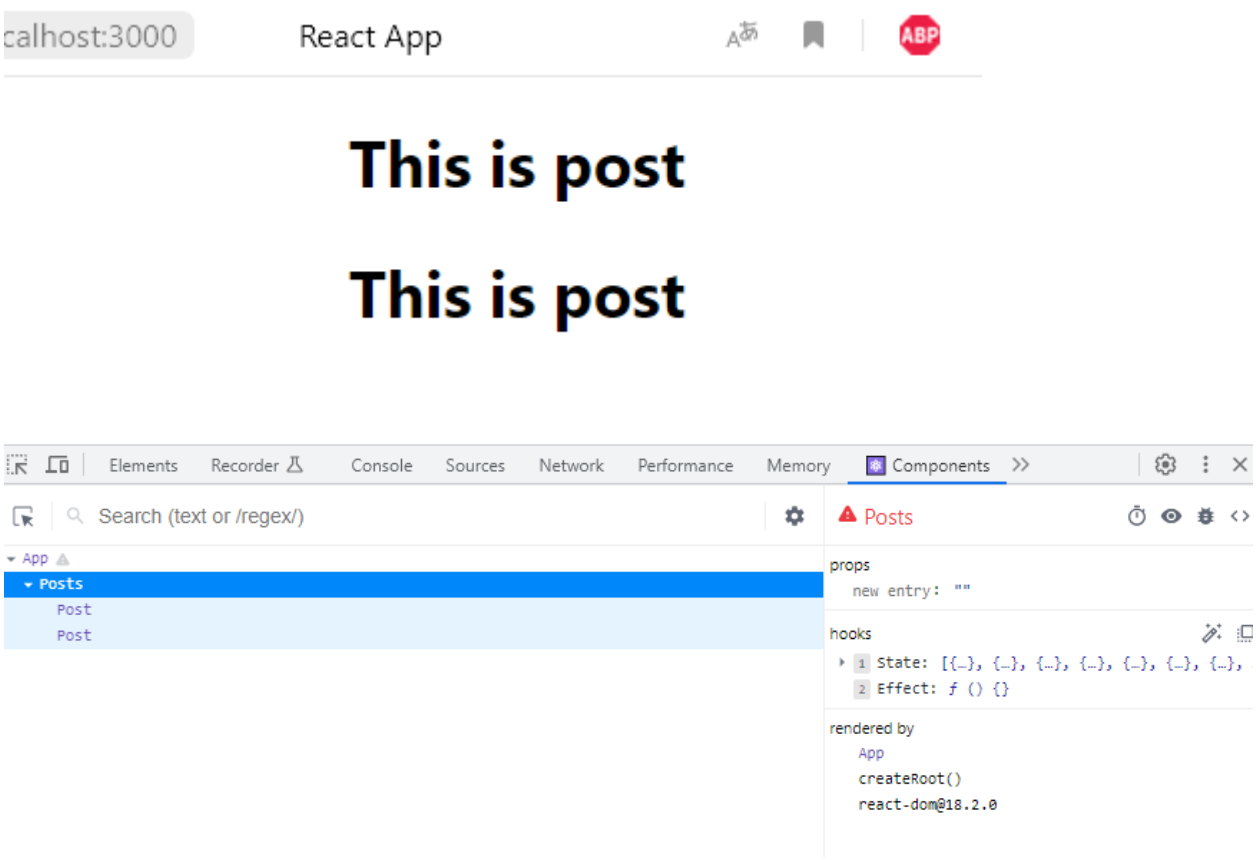






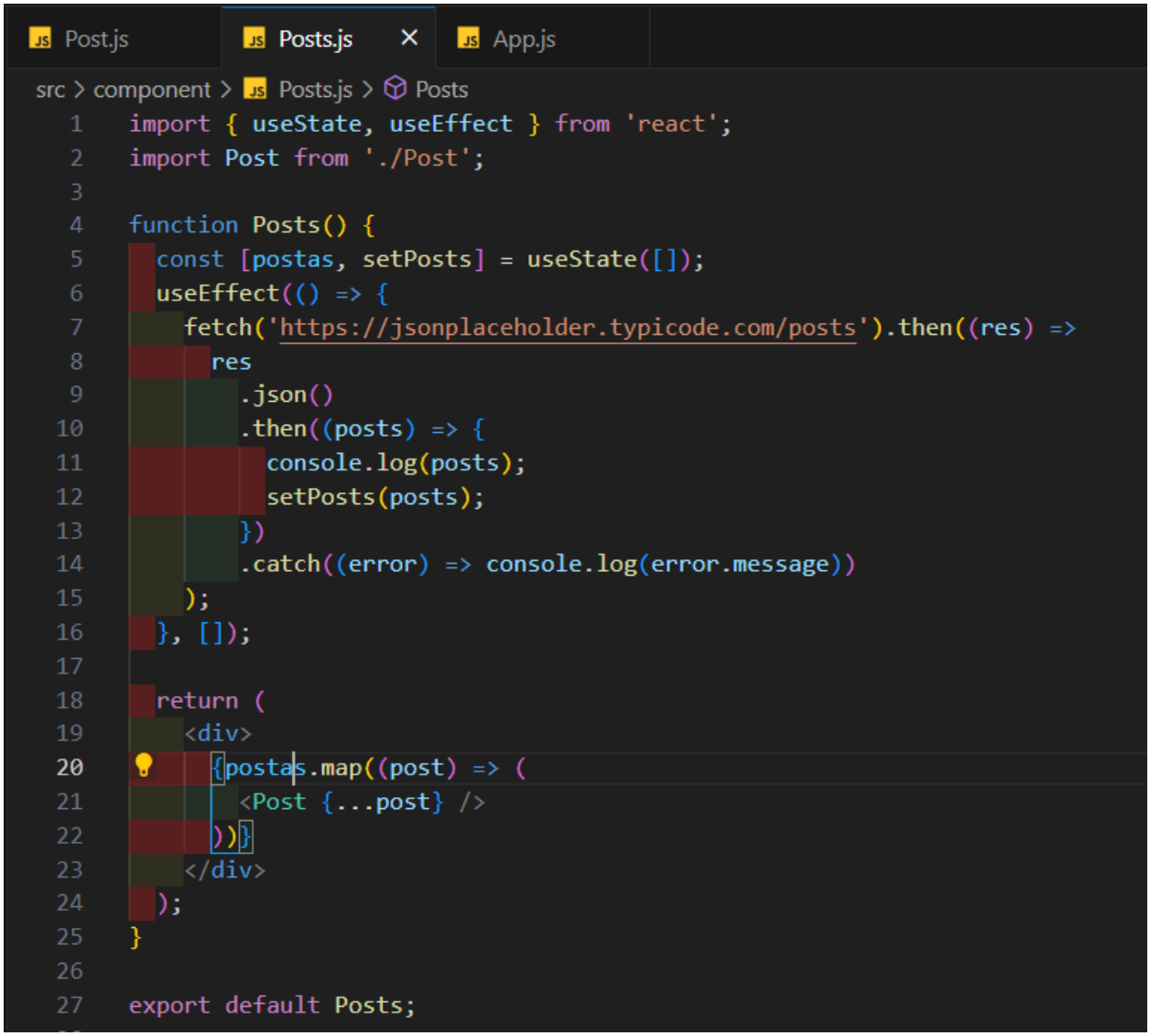
Получение массива постов через API

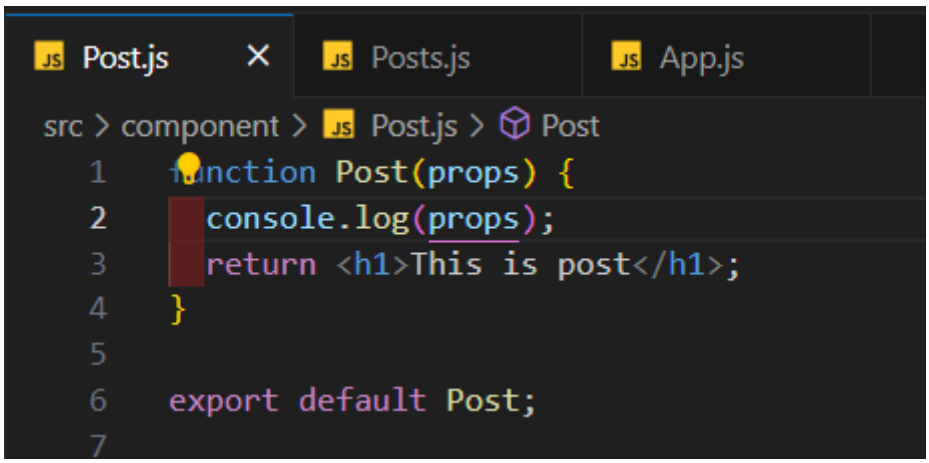
```
Post.js | Posts.js | App.js
src > component > Posts.js > Posts
1   import { useState, useEffect } from 'react';
2   import Post from './Post';
3
4   function Posts() {
5     const [postas, setPosts] = useState([]);
6     useEffect(() => {
7       fetch('https://jsonplaceholder.typicode.com/posts').then((res) =>
8         res.json().then((posts) => {
9           console.log(posts);
10          setPosts(posts);
11        })
12    );
13    }, []);
14
15    return (
16      <div>
17        <Post />
18        <Post />
19      </div>
20    );
21  }
22
23  export default Posts;
```



Самостоятельно добавить обработчик ошибок, вызвать ошибку и проверить состояние компонента Posts.

Отображение массива постов в интерфейсе





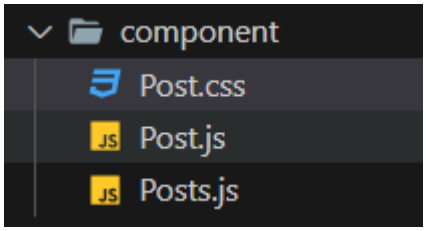
Выведем посты в видимую часть

Post.js | Posts.js | App.js

src > component > Posts.js > Posts > postas.map() callback

```
1   import { useState, useEffect } from 'react';
2   import Post from './Post';
3
4   function Posts() {
5     const [postas, setPosts] = useState([]);
6     useEffect(() => {
7       fetch('https://jsonplaceholder.typicode.com/posts').then((res) =>
8         res
9           .json()
10          .then((posts) => setPosts(posts))
11          .catch((error) => console.log(error.message))
12      );
13    }, []);
14
15    return (
16      <div>
17        {postas.map((post) => (
18          <Post key={post.id} {...post} />
19        ))}
20      </div>
21    );
22  }
23
24  export default Posts;
```

Самостоятельно в файл Post.js передать props, выполнить его деструктуризацию. Добавить стили и получить вид



1

sunt aut facere repellat provident occaecati excepturi optio reprehenderit

quia et suscipit suscipit recusandae consequuntur expedita et cum reprehenderit molestiae ut ut quas totam nostrum rerum est autem sunt rem eveniet architecto

User ID:1

2

qui est esse

est rerum tempore vitae sequi sint nihil reprehenderit dolor beatae ea dolores neque fugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis qui aperiam non debitis possimus qui neque nisi nulla

User ID:1

3

ea molestias quasi exercitationem repellat qui ipsa sit aut