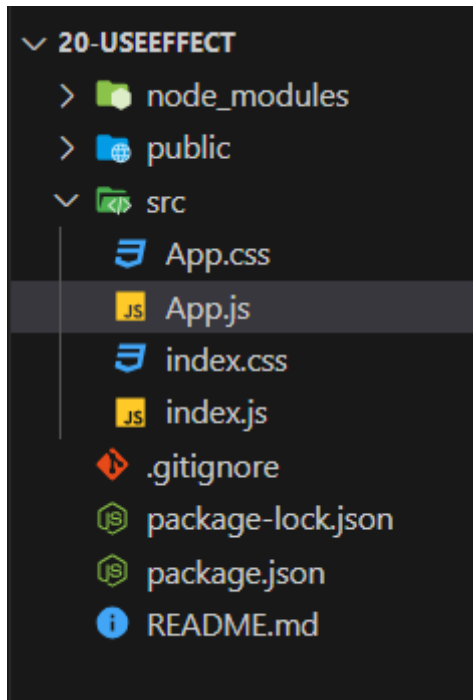
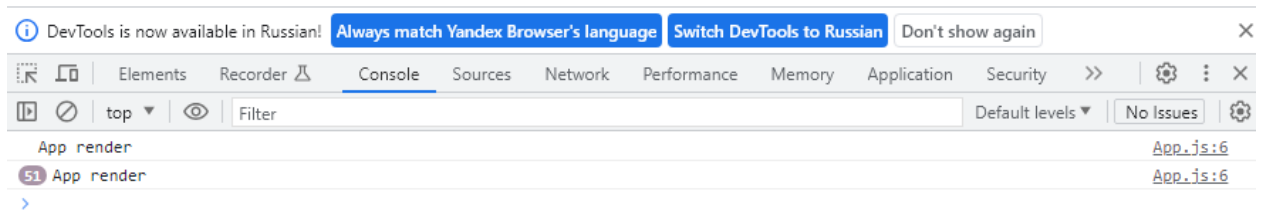


Проект по использованию fetch в компонентах

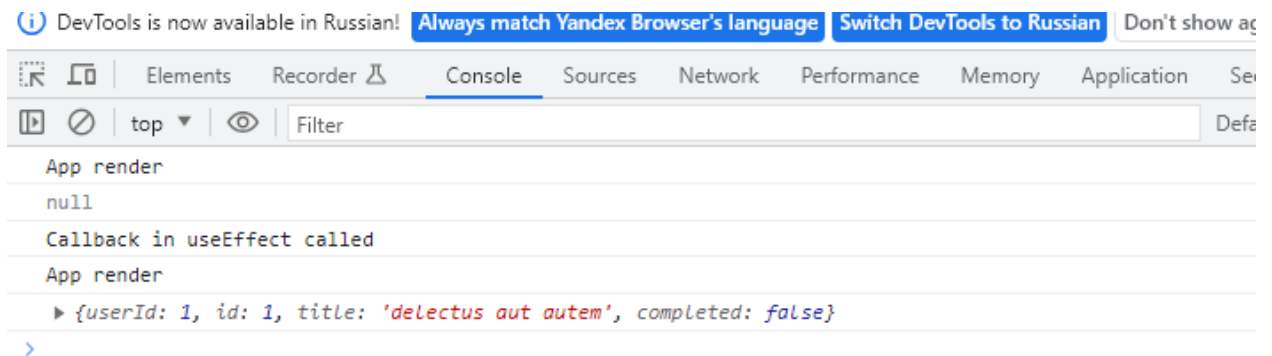


```
2  import './App.css';
3
4  function App() {
5    fetch('https://jsonplaceholder.typicode.com/todos/1')
6      .then((response) => response.json())
7      .then((json) => console.log(json));
8
9    return <div className="App"></div>;
10 }
11
12 export default App;
```

```
JS App.js ×
src > JS 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
```



```
src > JS 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
```

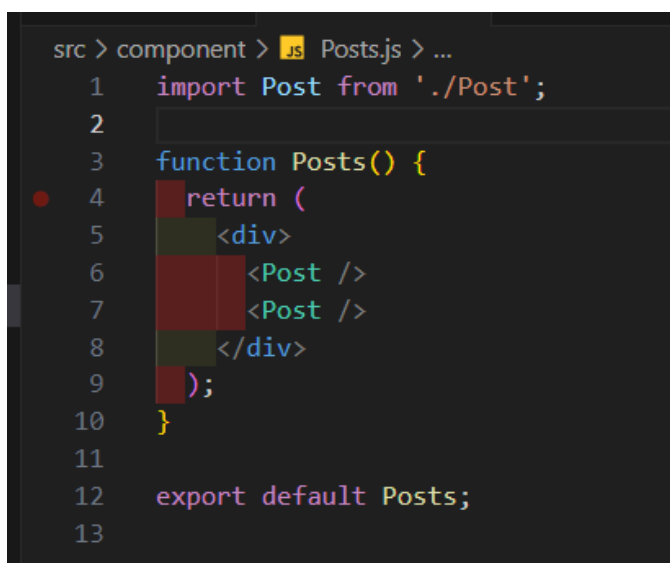
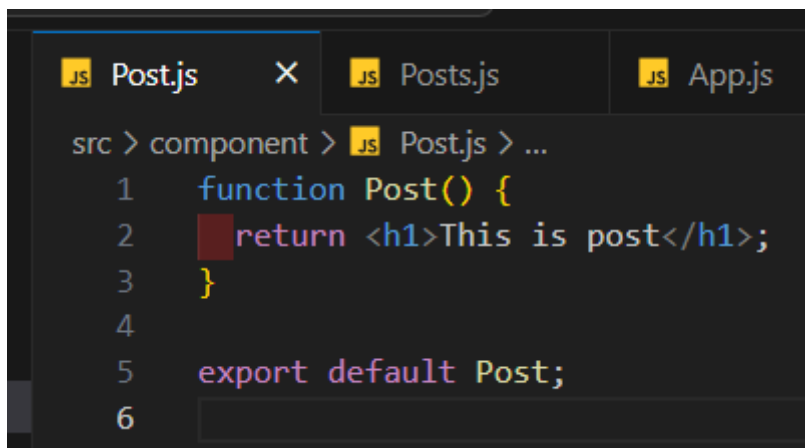
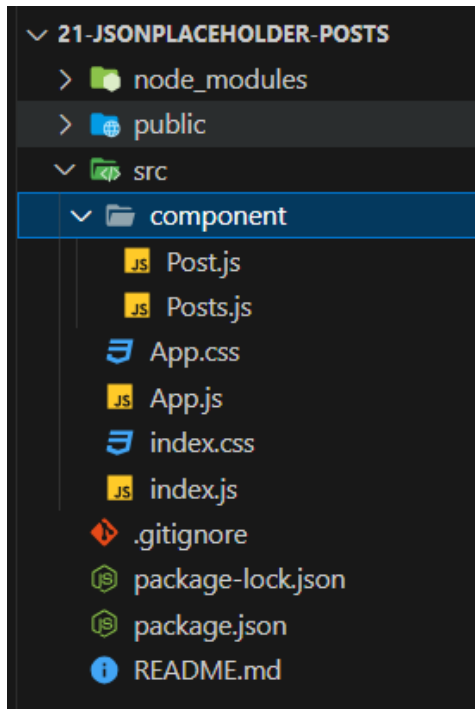


[todo] снова запустит бесконечный цикл обращений к серверу

Выведем данные с удаленного сервера в интерфейс

```
JS App.js ×
src > JS 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





localhost:3000

React App



This is post

This is post

Получение массива постов через API

```
JS Post.js JS Posts.js X JS App.js
src > component > JS Posts.js > Posts
1 import { useState, useEffect } from 'react';
2 import Post from './Post';
3
4 function Posts() {
5   const [posts, 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;
```

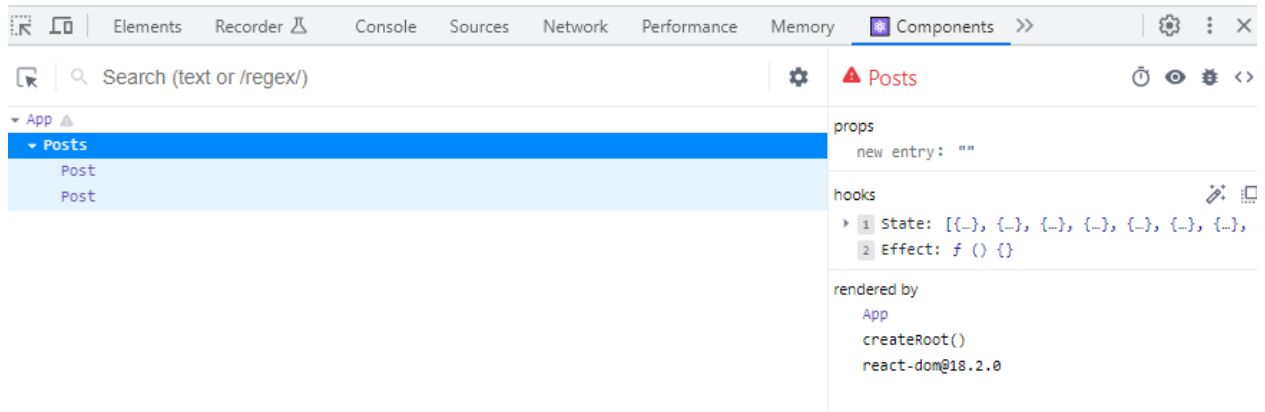
localhost:3000

React App



This is post

This is post



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

Отображение массива постов в интерфейсе

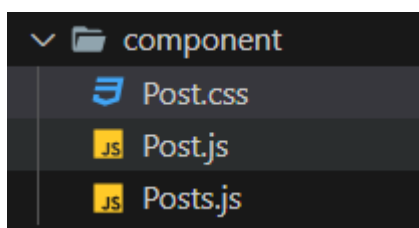
```
JS Post.js JS Posts.js X JS App.js
src > component > JS Posts.js > Posts
1 import { useState, useEffect } from 'react';
2 import Post from './Post';
3
4 function Posts() {
5   const [posts, setPosts] = useState([]);
6   useEffect(() => {
7     fetch('https://jsonplaceholder.typicode.com/posts').then((res) =>
8     res
9     .json()
10    .then((posts) => {
11      console.log(posts);
12      setPosts(posts);
13    })
14    .catch((error) => console.log(error.message))
15  );
16 }, []);
17
18 return (
19   <div>
20     {posts.map((post) => (
21       <Post {...post} />
22     ))}
23   </div>
24 );
25 }
26
27 export default Posts;
```

```
JS Post.js X JS Posts.js JS App.js
src > component > JS Post.js > Post
1 function Post(props) {
2   console.log(props);
3   return <h1>This is post</h1>;
4 }
5
6 export default Post;
7
```


Выведем посты в видимую часть

```
JS Posts.js X
21-jsonplaceholder-posts > src > component > JS Posts.js > Posts > useEffect() 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')
8        .then((res) => res.json())
9        .then((posts) => setPosts(posts))
10       .catch((error) => console.log(error.message));
11    }, []);
12
13    return (
14      <div>
15        {postas.map((post) => (
16          <Post key={post.id} {...post} />
17        ))}
18      </div>
19    );
20  }
21
22  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

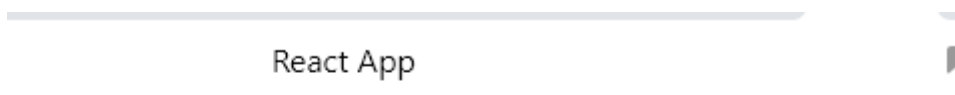
**ea molestias quasi exercitationem
repellat qui ipsa sit aut**

Добавим в наш проект улучшения:

1. Обработка ошибок при связи с сервером

```
JS Posts.js M X
21-jsonplaceholder-posts > src > component > JS Posts.js > Posts
1  import { useState, useEffect } from 'react';
2  import Post from './Post';
3
4  function Posts() {
5    const [posts, setPosts] = useState([]);
6    const [error, setError] = useState('');
7
8    useEffect(() => {
9      fetch('https://jsonplaceholder.typicode.com/posts')
10        .then((res) => res.json())
11        .then((posts) => setPosts(posts))
12        .catch((error) => setError(error.message));
13    }, []);
14
15    if (error) {
16      return <h1>Error: {error}</h1>;
17    }
18
19    return (
20      <div>
21        {posts.map((post) => (
22          <Post key={post.id} {...post} />
23        ))}
24      </div>
25    );
26  }
27
28  export default Posts;
```

Вызовем ошибку и убедимся, что catch реагирует на ошибку:



2. Добавление индикации загрузки данных

```
21-jsonplaceholder-posts > src > component > JS Posts.js > Posts
1  import { useState, useEffect } from 'react';
2  import Post from './Post';
3
4  function Posts() {
5    const [postas, setPosts] = useState([]);
6    const [error, setError] = useState('');
7    const [isLoading, setIsLoading] = useState(true);
8
9    useEffect(() => {
10      fetch('https://jsonplaceholder.typicode.com/posts')
11        .then((res) => res.json())
12        .then((posts) => setPosts(posts))
13        .catch((error) => setError(error.message))
14        .finally(() => setIsLoading(false));
15    }, []);
16
17    if (isLoading) {
18      return <h1>Loading...</h1>;
19    }
20
21    if (error) {
22      return <h1>Error: {error}</h1>;
23    }
24
25    return (
26      <div>
27        {postas.map((post) => (
28          <Post key={post.id} {...post} />
29        ))}
30      </div>
31    );
32  }
33
34  export default Posts;
```

3. Перенос индикатора загрузки на место постов

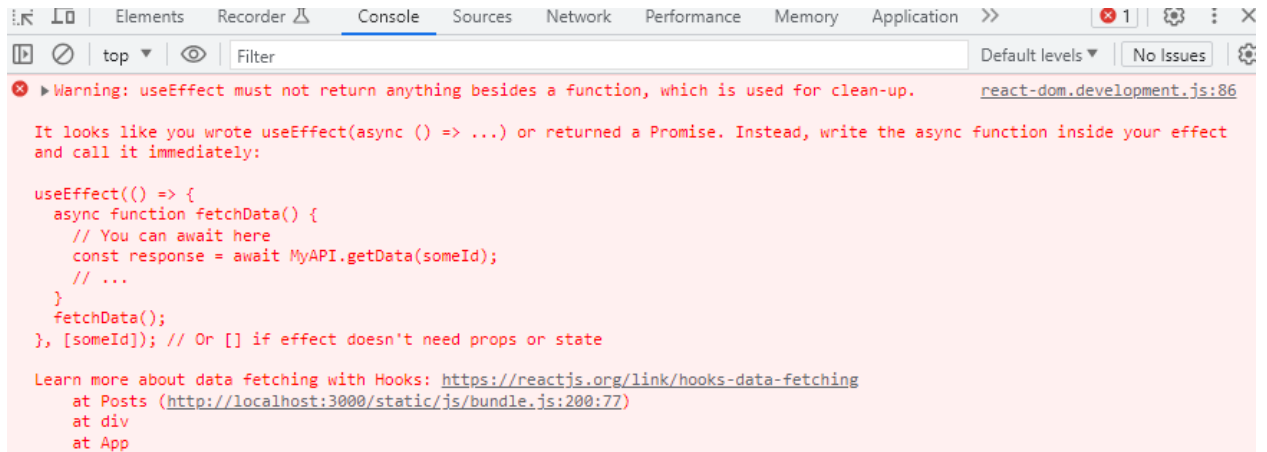
```
JS Posts.js M X
21-jsonplaceholder-posts > src > component > JS Posts.js > Posts > useEffect() callback
1  import { useState, useEffect } from 'react';
2  import Post from './Post';
3
4  const API_URL = 'https://jsonplaceholder.typicode.com/posts';
5
6  function Posts() {
7    const [posts, setPosts] = useState([]);
8    const [error, setError] = useState('');
9    const [isLoading, setIsLoading] = useState(true);
10
11    useEffect(() => {
12      fetch(API_URL)
13        .then((res) => res.json())
14        .then((posts) => setPosts(posts))
15        .catch((error) => setError(error.message))
16        .finally(() => setIsLoading(false));
17    }, []);
18
19    if (error) {
20      return <h1>Error: {error}</h1>;
21    }
22
23    return (
24      <>
25        <h1>Posts</h1>
26        <hr />
27        {isLoading ? (
28          <h1>Loading...</h1>
29        ) : (
30          posts.map((post) => <Post key={post.id} {...post} />)
31        )}
32      </>
33    );
34  }
35
36  export default Posts;
37
```

Проект с async await в useEffect

(на основе проекта 21-jsonplaceholder-posts)

```
JS Posts.js U X
22-useEffect-async-await > src > component > JS Posts.js > Posts
1  import { useState, useEffect } from 'react';
2  import Post from '../Post';
3
4  const API_URL = 'https://jsonplaceholder.typicode.com/posts';
5
6  function Posts() {
7    const [postas, setPostas] = useState([]);
8    const [error, setError] = useState('');
9    const [isLoading, setIsLoading] = useState(true);
10
11    useEffect(async () => {
12      try {
13        const res = await fetch(API_URL);
14        const postas = await res.json();
15        setPostas(postas);
16      } catch (error) {
17        setError(error.message);
18      }
19      setIsLoading(false);
20    }, []);
21
22    if (error) {
23      return <h1>Error: {error}</h1>;
24    }
25
26    return (
27      <>
28        <h1>Posts</h1>
29        <hr />
30        {isLoading ? (
31          <h1>Loading...</h1>
32        ) : (
33          postas.map((post) => <Post key={post.id} {...post} />)
34        )}
35      </>
36    );
37  }
38
39  export default Posts;
40
```

Наша функция работает, но в консоли мы видим предупреждение:



Связано это с тем, что `useEffect` может возвращать только `undefined` или другую функцию (даже асинхронную). В нашем же случае мы возвращаем асинхронную стрелочную функцию, которая всегда возвращает промис, что для `useEffect` не является правильным: когда react убирает компонент из видимой части интерфейса, то он задействует метод `Unmount`, который в свою очередь вызовет функцию после ключевого слова `return`. Например,

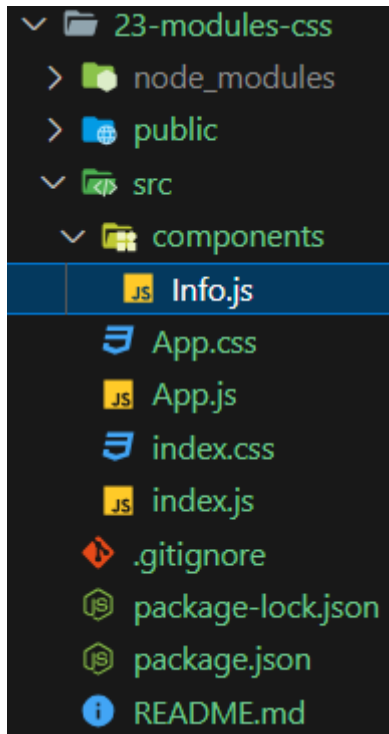
```
useEffect(() => {  
  //тело функции  
  
  return () => {  
    console.log('123');  
  };  
}, []);
```

В этом предупреждении нам предлагается внутри `useEffect` организовать другую, асинхронную, функцию:

22-useEffect-async-await > src > component > JS Posts.js > ...

```
1  import { useState, useEffect } from 'react';
2  import Post from './Post';
3
4  const API_URL = 'https://jsonplaceholder.typicode.com/posts';
5
6  function Posts() {
7    const [postas, setPosts] = useState([]);
8    const [error, setError] = useState('');
9    const [isLoading, setIsLoading] = useState(true);
10
11    useEffect(() => {
12      async function fetchData() {
13        try {
14          const res = await fetch(API_URL);
15          const posts = await res.json();
16          setPosts(posts);
17        } catch (error) {
18          setError(error.message);
19        }
20        setIsLoading(false);
21      }
22      fetchData();
23    }, []);
24
25    if (error) {
26      return <h1>Error: {error}</h1>;
27    }
28
29    return (
30      <>
31        <h1>Posts</h1>
32        <hr />
33        {isLoading ? (
34          <h1>Loading...</h1>
35        ) : (
36          postas.map((post) => <Post key={post.id} {...post} />)
37        )}
38      </>
39    );
40  }
41
42  export default Posts;
```


Проект с модулями CSS



```
23-modules-css > src > components > JS Info.js > [🔗] default
1  function Info() {
2    return (
3      <>
4      <h1>Hello from the Info component</h1>
5      <button>Click me!</button>
6      </>
7    );
8  }
9
10 export default Info
11
```

3000

React App

Aあ

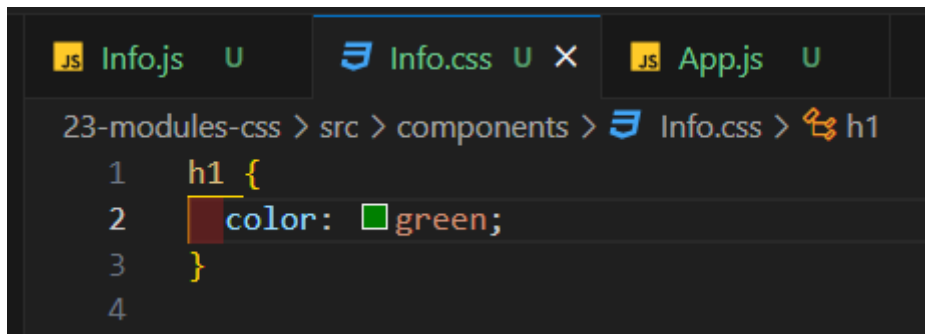
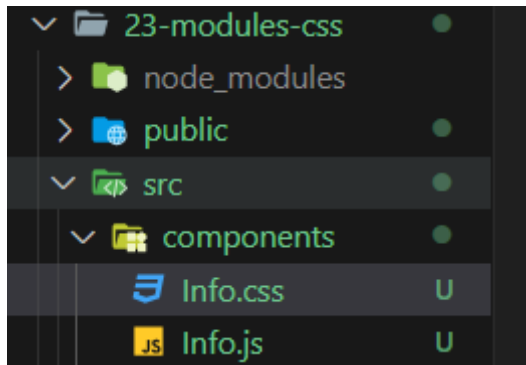
🔖

AB

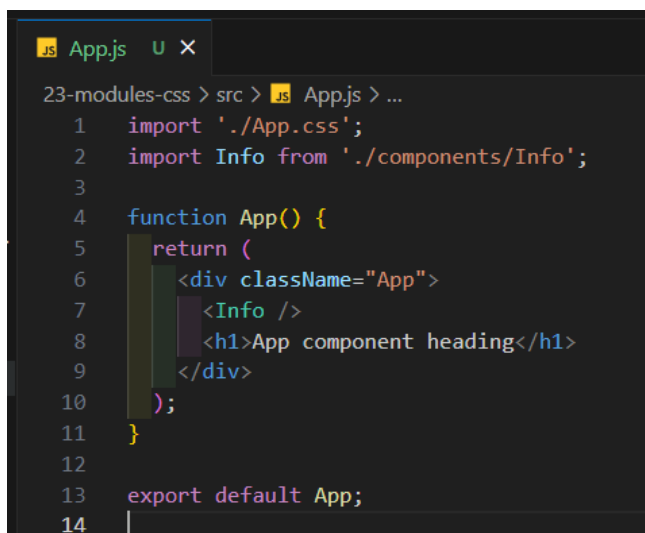
Hello from the Info component

Click me!

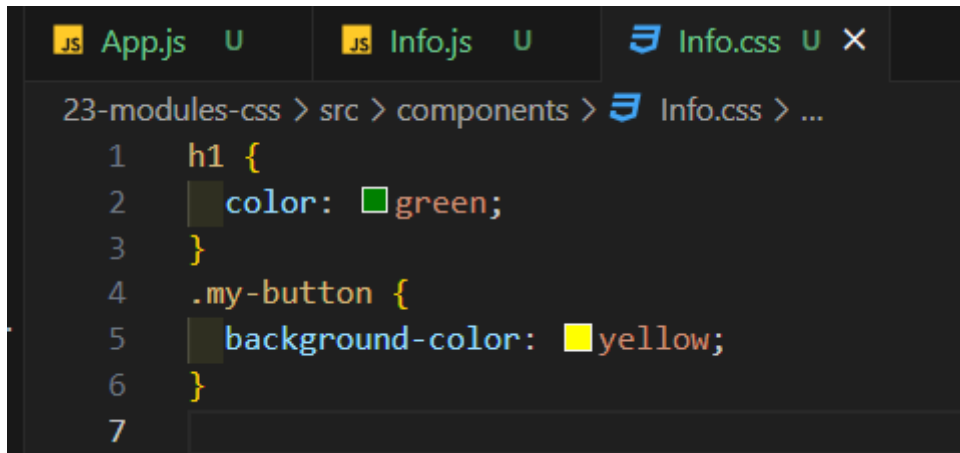
Применим стили:



Далее в файл App.js добавим еще компонент и увидим, что он тоже стал зеленого цвета

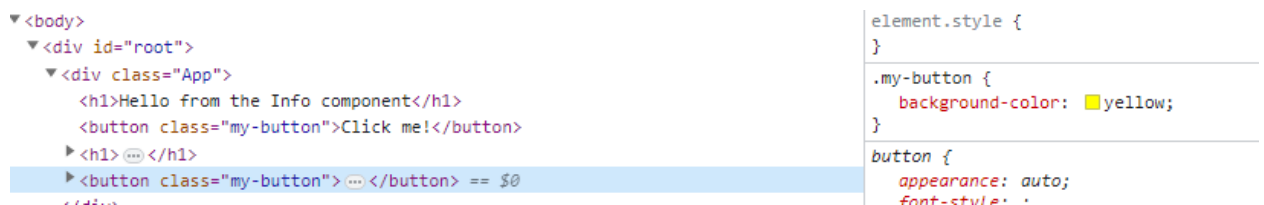


Для кнопки в компоненте Info.js назовем класс my-button



```
JS App.js U JS Info.js U Info.css U X
23-modules-css > src > components > Info.css > ...
1  h1 {
2    color: green;
3  }
4  .my-button {
5    background-color: yellow;
6  }
7
```

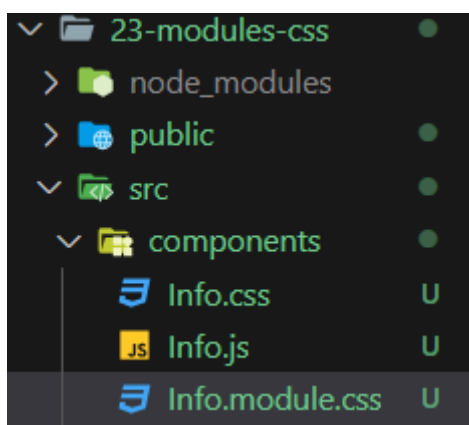
Добавим еще одну кнопку с классом my-button в App.js и убедимся, что цвет кнопки стал желтым. Однако в компонент в App.js не производился экспорт Info.css! Это значит, что свойства из файла Info.css доступны глобально для всего react-приложения. Это может привести к конфликтам правил css в крупных проектах.



```
<body>
  <div id="root">
    <div class="App">
      <h1>Hello from the Info component</h1>
      <button class="my-button">Click me!</button>
      <h1>...</h1>
      <button class="my-button">...</button> == $0
    
```

```
element.style {
}
.my-button {
  background-color: yellow;
}
button {
  appearance: auto;
  font-style: ...
}
```

Что бы не вызывать конфликты рекомендуется локализовать стили, т.е. использовать module.css

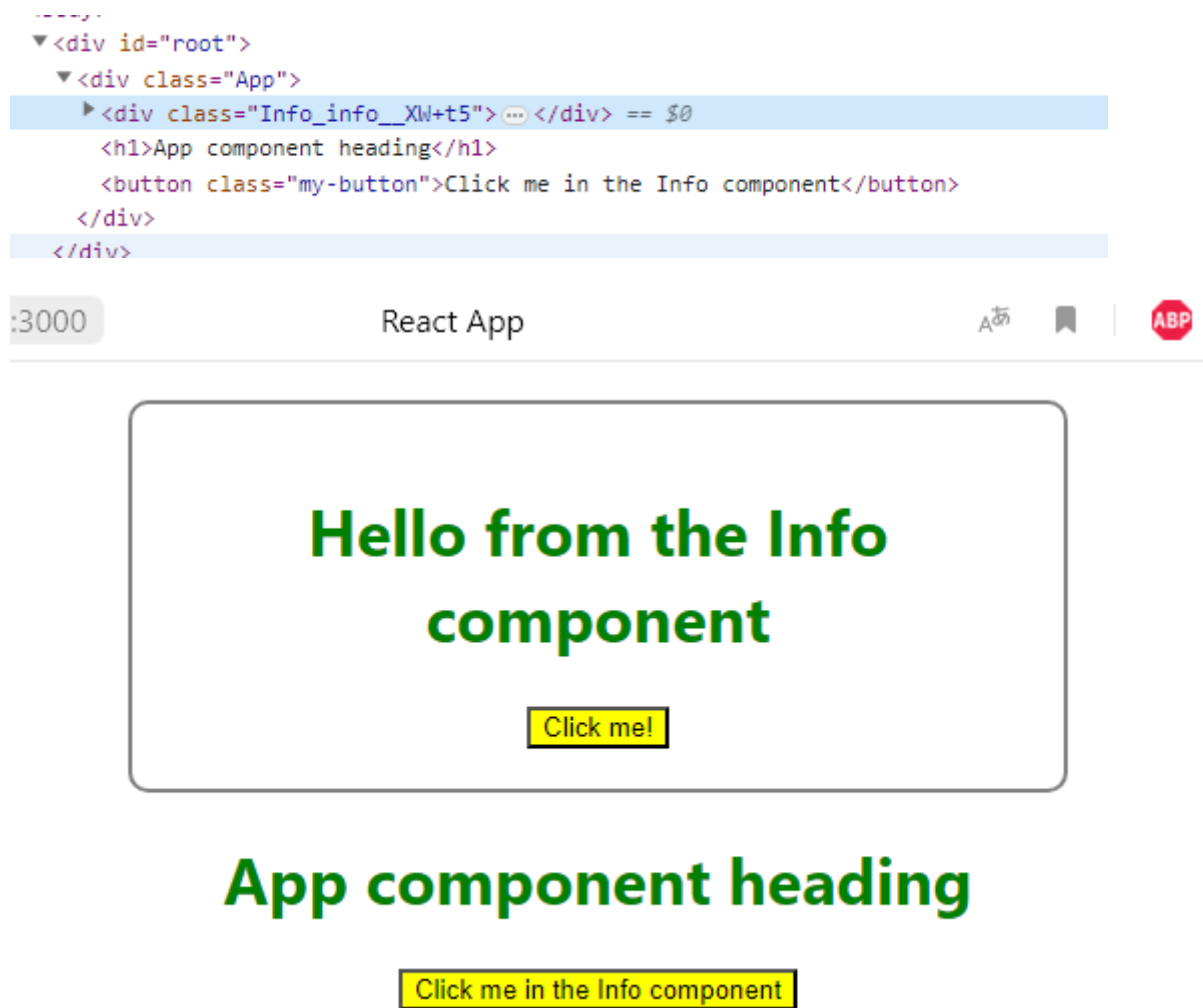


```
Info.module.css U X JS Info.js U JS App.js U Info.c
23-modules-css > src > components > Info.module.css > .info
1  .info {
2    border: 2px solid gray;
3    border-radius: 10px;
4    padding: 20px;
5    max-width: 40%;
6    margin: 20px auto;
7  }
8
9  .myOtherButton {
10   background: coral;
11 }
```

```
Elements Recorder Console Sources Network Performance Memory
top Filter
▼ {info: 'Info_info_XW+t5', myOtherButton: 'Info_myOtherButton__DPMY3'}
  info: "Info_info_XW+t5"
  myOtherButton: "Info_myOtherButton__DPMY3"
  ▶ [[Prototype]]: Object
>
```

То есть перед нами объект, к свойству которого мы можем обращаться привычным образом

```
Info.module.css U JS Info.js U X JS App.js U Info.css U
23-modules-css > src > components > Info.js > ...
1  import './Info.css';
2  import styles from './Info.module.css';
3
4  console.log(styles);
5
6  function Info() {
7    return (
8      <div className={styles.info}>
9        <h1>Hello from the Info component</h1>
10       <button className="my-button">Click me!</button>
11     </div>
12   );
13 }
14
15 export default Info;
16
```



После локализации CSS-свойств в одном компоненте мы уже не можем обращаться к ним из другого компонента:

```

JS App.js  U X
23-modules-css > src > JS App.js > ...
1  import './App.css';
2  import Info from './components/Info';
3
4  function App() {
5    return (
6      <div className="App">
7        <Info />
8        <div className="info">
9          <h1>App component heading</h1>
10         <button className="my-button">Click me in the Info component</button>
11       </div>
12     </div>
13   );
14 }
15
16 export default App;
17

```

Изменений не произошло:

:3000 React App

あ



Hello from the Info component

Click me!

App component heading

Click me in the Info component

```
...
<div class="App">
  <div class="Info_info_XW+t5">
    <h1>Hello from the Info component</h1>
    <button class="my-button">Click me!</button>
  </div>
  <div class="info">...</div> == $0
</div>
...
```

Но если создать правило по селектору, то оно будет работать. Делать так не рекомендуется, лучше использовать классы!

```
JS App.js U Info.module.css U X
23-modules-css > src > components > Info.module.css > h
1  .info {
2    border: 2px solid gray;
3    border-radius: 10px;
4    padding: 20px;
5    max-width: 40%;
6    margin: 20px auto;
7  }
8
9  .myOtherButton {
10   background: coral;
11 }
12
13 h2 {
14   color: blue;
15 }
16
```

```
JS App.js U X Info.module.css U
23-modules-css > src > App.js > App
1  import './App.css';
2  import Info from './components/Info';
3
4  function App() {
5    return (
6      <div className="App">
7        <Info />
8        <div className="info">
9          <h1>App component heading</h1>
10         <h2>Heading in the App component</h2>
11         <button className="my-button">Click me in the Info component</button>
12       </div>
13     </div>
14   );
15 }
16
17 export default App;
```

**Hello from the Info
component**

Click me!

App component heading

Heading in the App component

Click me in the Info component