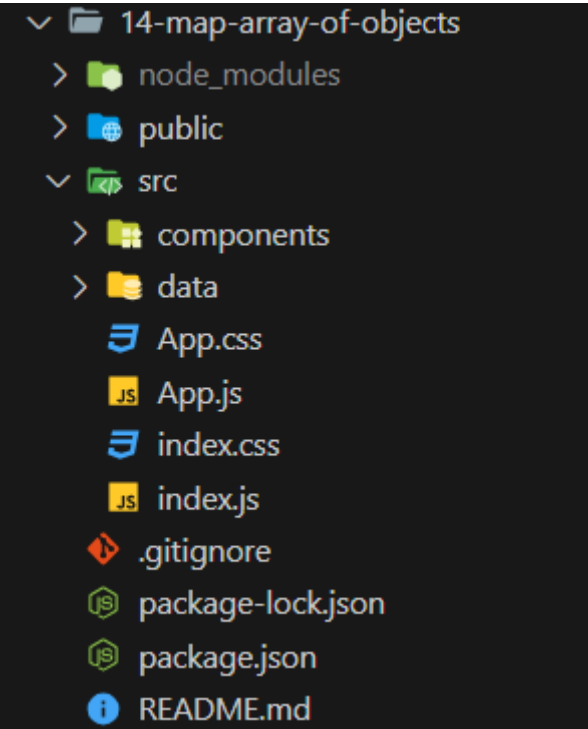


Проект с итерацией по массиву объектов



<div>200 × 100</div> <div>Addy Mostyn amostyn0@smugmug.com</div>	<div>200 × 100</div> <div>Berry Brammar bbrammar1@about.com</div>	<div>200 × 100</div> <div>Gavan Crigin gcrigin2@bandcamp.com</div>	<div>200 × 100</div> <div>Karrie De L'Isle kdelisle3@auda.org.au</div>
<div>200 × 100</div> <div>Jammie Concannon jconcannon6@cloudflare.com</div>	<div>200 × 100</div> <div>Forrest Shawley fshawley7@digg.com</div>	<div>200 × 100</div> <div>Katie Laraway klaraway8@uol.com.br</div>	<div>200 × 100</div> <div>Izzy Munnings imunnings9@ocn.ne.jp</div>

1 этап

```
JS person.js JS persons.js JS App.js M X
14-map-array-of-objects > src > JS App.js > App > persons.map() callback
1  import './App.css';
2  import persons from './data/persons';
3  import Person from './components/person';
4
5  function App() {
6    return (
7      <div className="App">
8        {persons.map((person) => {
9          const { id, firstName, lastName, email, img } = person;
10         return (
11           <Person
12             id={id}
13             firstName={firstName}
14             lastName={lastName}
15             email={email}
16             img={img}
17           />
18         );
19       })}
20     </div>
21   );
22 }
23
24 export default App;
```

```
JS person.js X JS App.js JS Counter.js
14-map-array-of-objects > src > components > JS person.js > ...
1  function Person(props) {
2    console.log(props);
3    return <div></div>;
4  }
5
6  export default Person;
7  |
```

14-map-array-of-objects > src > data > JS persons.js > default

```
1  const persons = [  
2    {  
3      id: 1,  
4      firstName: 'Addy',  
5      lastName: 'Mostyn',  
6      email: 'amostyn0@smugmug.com',  
7      img: 'http://dummyimage.com/200x100.png/5fa2dd/ffffff',  
8    },  
9    {  
10     id: 2,  
11     firstName: 'Berry',  
12     lastName: 'Brammar',  
13     email: 'bbrammar1@about.com',  
14     img: 'http://dummyimage.com/200x100.png/ff4444/ffffff',  
15   },  
16   {  
17     id: 3,  
18     firstName: 'Gavan',  
19     lastName: 'Crigin',  
20     email: 'gcrigin2@bandcamp.com',
```

Внимание!!! Содержимое файла data/persons.js сгенерировано на сайте <https://mockaroo.com/> и содержится в файле MOCK_DATA.json

2 этап

Что бы устранить избыточность кода в компоненте Person используем оператор spread (...) и укажем уникальный ключ key, в качестве значения которого возьмем person.id

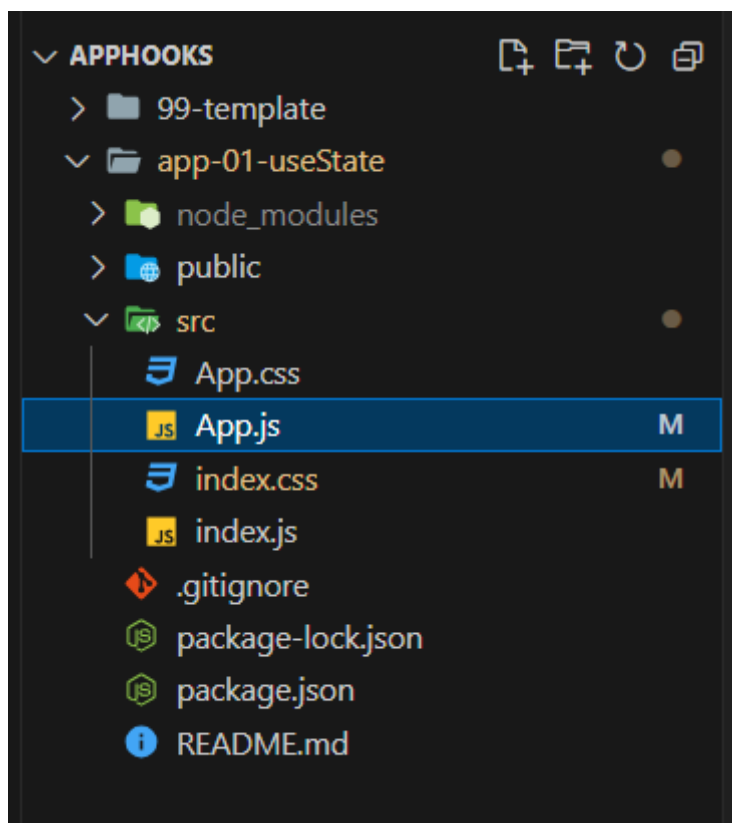
```
JS Person.js JS persons.js JS App.js M X
14-map-array-of-objects > src > JS App.js > App > persons.map() callback
1  import './App.css';
2  import persons from './data/persons';
3  import Person from './components/Person';
4
5  function App() {
6    return (
7      <div className="App">
8        {persons.map((person) => {
9          return <Person key={person.id} {...person} />;
10        })}
11      </div>
12    );
13  }
14
15  export default App;
```

Сформируем блок JSX-кода на основе полученных свойств

```
JS Person.js M X JS persons.js JS App.js M
14-map-array-of-objects > src > components > JS Person.js > Person
1  function Person(props) {
2    const { firstName, lastName, email, img } = props;
3    return (
4      <div>
5        <img src={img} />
6        <h3>
7          {firstName} {lastName}
8        </h3>
9        <h4>{email}</h4>
10      </div>
11    );
12  }
13
14  export default Person;
15
```

Самостоятельно добавить стили для получения вышеуказанного результата

Хук useState (функциональный и классовый компоненты)



localhost:3000

React App

Hello 3

Start aditing to see some magic happen!

Увеличить

Уменьшить

Функциональный компонент и его подключение

```
JS App.js M X
app-01-useState > src > JS App.js > ...
1 import { useState } from 'react';
2
3 export default function App() {
4   const [item, setItem] = useState(1);
5   const incrementItem = () => setItem(item + 1);
6   const decreaseItem = () => setItem(item - 1);
7
8   return (
9     <div className="App">
10      <h1>Hello {item}</h1>
11      <h2>Start adding to see some magic happen!</h2>
12      <button onClick={incrementItem}>Увеличить</button>
13      <button onClick={decreaseItem}>Уменьшить</button>
14    </div>
15  );
16 }
17
```

```
JS App.js M JS index.js M
app-01-useState > src > JS index.js > ...
1 import React from 'react';
2 import ReactDOM from 'react-dom/client';
3 import './index.css';
4 import App from './App';
5
6 const root = ReactDOM.createRoot(document.
  getElementById('root'));
7 root.render(<App />);
8
```

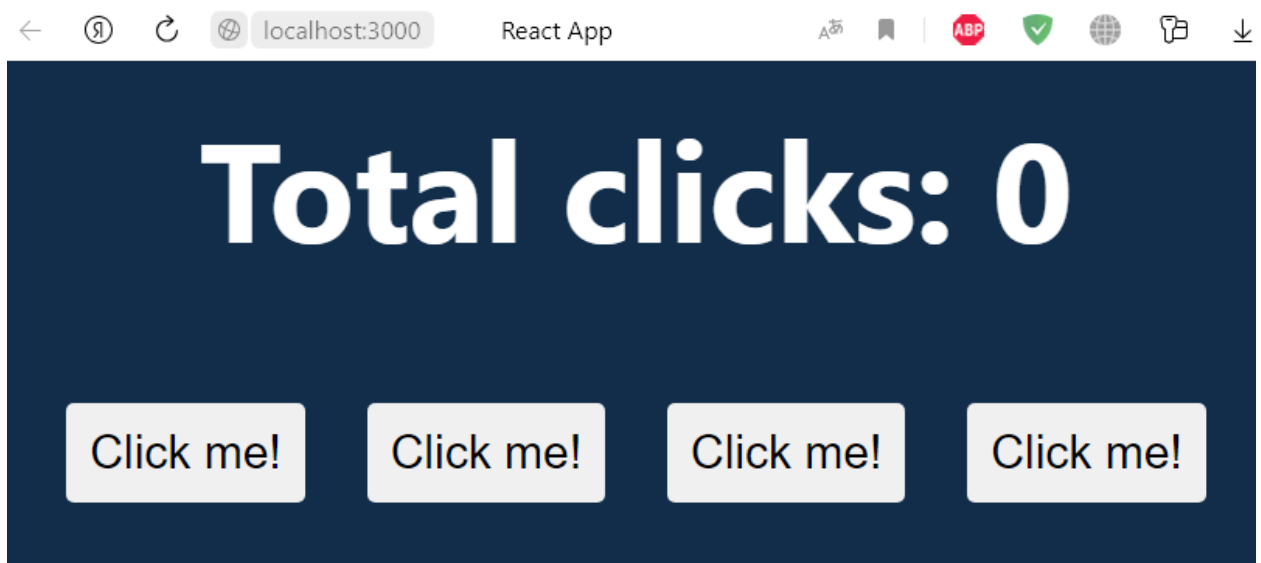
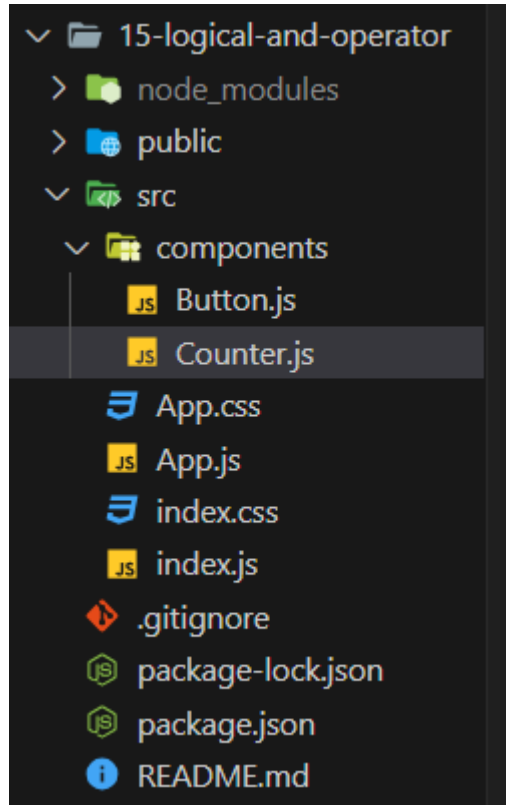
Классовый компонент и его подключение

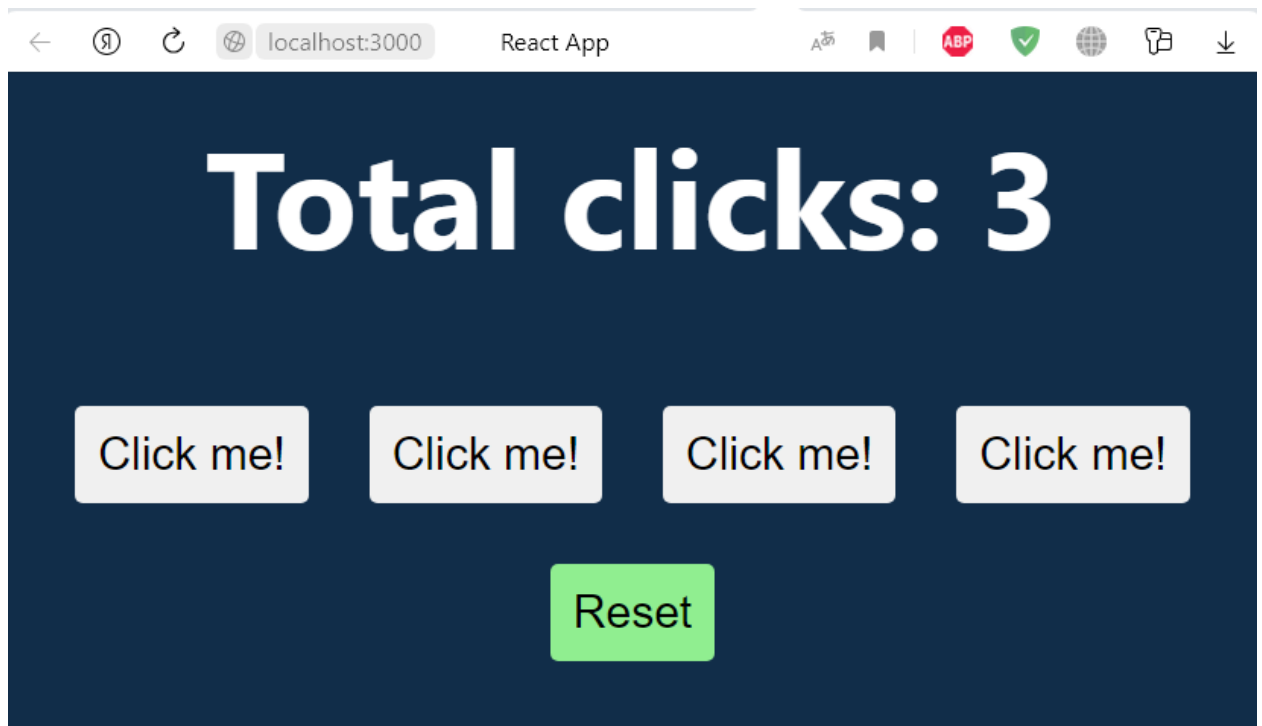
```
JS App.js M X JS index.js
app-01-useState > src > JS App.js > ...
1 import { useState } from 'react';
2 import React from 'react';
3
4 > export default function App() { ...
17 }
18
19 class AppClassUseState extends React.Component {
20   state = {
21     item: 1,
22   };
23   render() {
24     const { item } = this.state;
25     return (
26       <div className="App">
27         <h1>Hello {item}</h1>
28         <h2>Start adding to see some magic happen!</h2>
29         <button onClick={this.incrementItem}>Увеличить</button>
30         <button onClick={this.decreaseItem}>Уменьшить</button>
31       </div>
32     );
33   }
34   incrementItem = () =>
35     this.setState((state) => {
36       return { item: state.item + 1 };
37     });
38   decreaseItem = () =>
39     this.setState((state) => {
40       return { item: state.item - 1 };
41     });
42 }
```

```
JS App.js M JS index.js X
app-01-useState > src > JS index.js > ...
1 import React from 'react';
2 import ReactDOM from 'react-dom/client';
3 import './index.css';
4 import App from './App';
5 import AppClassUseState from './App';
6
7 const root = ReactDOM.createRoot(document.getElementById('root'));
8 root.render(<AppClassUseState />);
9
```

Самостоятельно вынести в отдельные компоненты оба способа реализации useState

Проект с оператором И в коде JSX (на основе 12-state-via-props)





```
<html lang="en">
  <head> ... </head>
  <body>
    <div id="root">
      <div class="App">
        <h1> ... </h1>
        <button>Click me!</button>
        <button>Click me!</button>
        <button>Click me!</button>
        <button>Click me!</button>
        <div == $0
          <button style="background-color: lightgreen;">Reset</button>
        </div>
      </div>
    </div>
  </div>
```

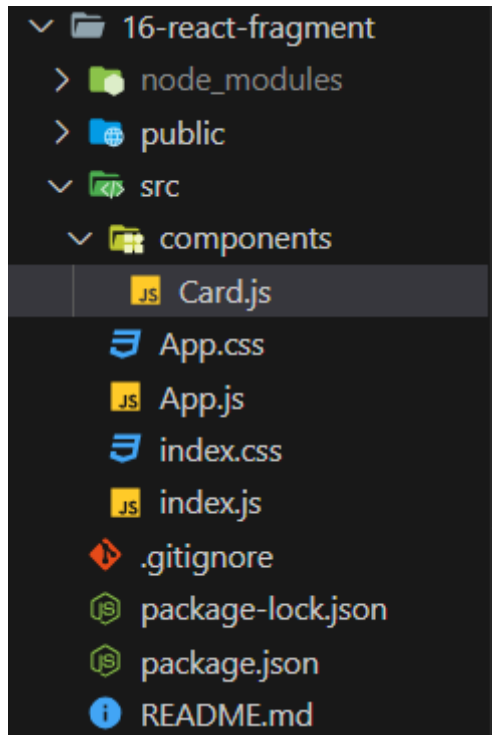
JS App.js



15-logical-and-operator > src > JS App.js > ...

```
1  import { useState } from 'react';
2  import './App.css';
3  import Button from './components/Button';
4  import Counter from './components/Counter';
5
6  function App() {
7    const [count, setCount] = useState(0);
8    const incrementCount = () => {
9      setCount(count + 1);
10   };
11   const resetCount = () => {
12     setCount(0);
13   };
14
15   return (
16     <div className="App">
17       <Counter countProps={count} />
18       <Button onClick={incrementCount} />
19       <Button onClick={incrementCount} />
20       <Button onClick={incrementCount} />
21       <Button onClick={incrementCount} />
22       {count > 0 && (
23         <div>
24           <button
25             style={{ backgroundColor: 'lightgreen' }}
26             onClick={resetCount}
27           >
28             Reset
29           </button>
30         </div>
31       )}
32     </div>
33   );
34 }
35
36 export default App;
37
```

Проект по использованию React Fragment



Student

I am studying a course on React

Like!

```
<!DOCTYPE html>
<html lang="en">
  <head> </head>
  <body>
    <div id="root">
      <div class="App">
        <div == $0
          <h1>Student</h1>
          <h3>I am studying a course on React</h3>
          <button>Like!</button>
        </div>
      </div>
    </div>
  </body>
</html>
```

```
JS Card.js × JS App.js
16-react-fragment > src > components > JS Card.js > ...
1  const Card = () => {
2    return (
3      <div>
4        <h1>Student</h1>
5        <h3>I am studying a course on React</h3>
6        <button>Like!</button>
7      </div>
8    );
9  };
10
11  export default Card;
12
```

Добавим в проект React Fragment

```
JS Card.js × JS App.js
16-react-fragment > src > components > JS Card.js > [🔗] Card
1  const Card = () => {
2    return (
3      <>
4        <h1>Student</h1>
5        <h3>I am studying a course on React</h3>
6        <button>Like!</button>
7      </>
8    );
9  };
10
11  export default Card;
12
```

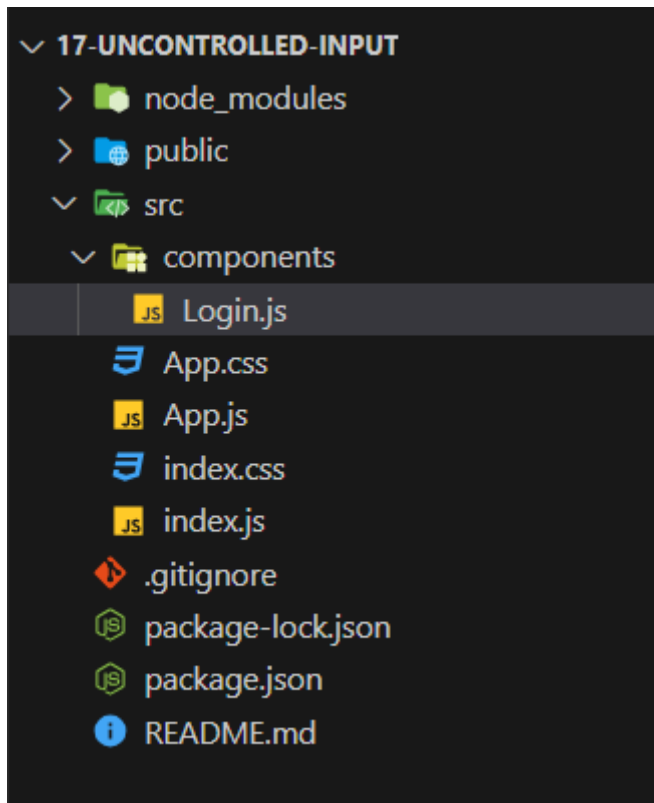
```
<!DOCTYPE html>
...<html lang="en"> == $0
  <head>...</head>
  <body>
    <div id="root">
      <div class="App">
        <h1>Student</h1>
        <h3>I am studying a course on React</h3>
        <button>Like!</button>
      </div>
    </div>
  </body>
</html>
```

Проект по неконтролируемым полям ввода

Поля ввода в React делятся на **неконтролируемые** и **контролируемые**.

Неконтролируемые поля называются так потому что их значения ни где не сохраняются.

Из проекта 12-state-via-props перенести файлы App.css и index.css



Login Form

Username:

Password:

Login

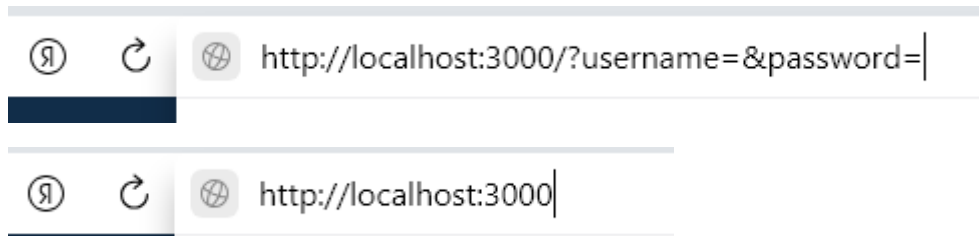
```
JS Login.js X JS App.js
src > components > JS Login.js > Login
1  function Login() {
2      return (
3          <>
4              <h1>Login Form</h1>
5              <form>
6                  <label>
7                      Username: <input type="text"
8                          name="username" />
9                  </label>
10                 <label>
11                     Password: <input type="password"
12                         name="password" />
13                 </label>
14                 <button type="submit">Login</button>
15             </form>
16         </>
17     );
18 }
19
20 export default Login;
```

Сделаем так, чтобы значения введенные в Username и Password выводились в консоль и в alert.

Сначала внесем следующие изменения


```
JS Login.js x
src > components > JS Login.js > ...
1  function Login() {
2      function handelFormSubmit(event) {}
3      return (
4          <>
5              <h1>Login Form</h1>
6              <form onSubmit={handelFormSubmit}>
7                  <label>
8                      Username: <input type="text" name="username" />
9                  </label>
10                 <label>
11                     Password: <input type="password" name="password" />
12                 </label>
13                 <button type="submit">Login</button>
14             </form>
15         </>
16     );
17 }
18
19 export default Login;
```

Обновим страницу и в адресной строке уберем

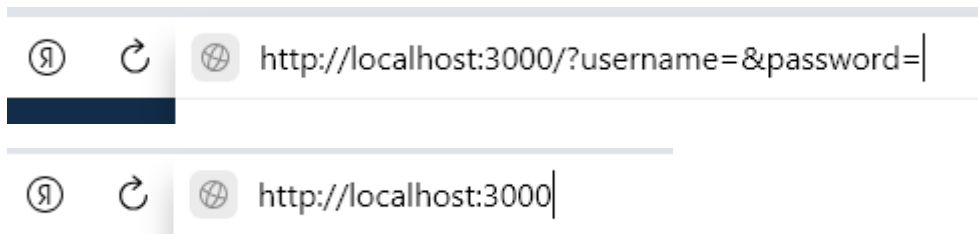


Внесем информацию в поля и нажмем Login. Снова в адресной строке видна информация о переходе на новую страницу (браузер выполнил обновление страницы). Такое поведение недопустимо с точки зрения React, т.к. в данном случае не поддерживается концепция SPA.

Что бы предотвратить такое поведение внесем следующие изменения:

```
JS Login.js x
src > components > JS Login.js > Login > handelFormSubmit
1 function Login() {
2   function handelFormSubmit(event) {
3     event.preventDefault();
4   }
5   return (
6     <>
7       <h1>Login Form</h1>
8       <form onSubmit={handelFormSubmit}>
9         <label>
10           Username: <input type="text" name="username" />
11         </label>
12         <label>
13           Password: <input type="password" name="password" />
14         </label>
15         <button type="submit">Login</button>
16       </form>
17     </>
18   );
19 }
20
21 export default Login;
```

Обновим страницу, в адресной строке уберем



Обновим страницу и убедимся что перенаправления на другую страницу больше не происходит.

Теперь выведем в консоль информацию из полей формы

Login Form

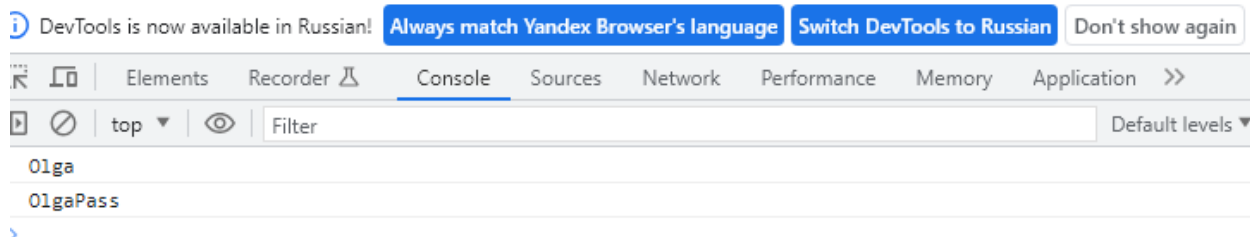
Username:

Olga

Password:

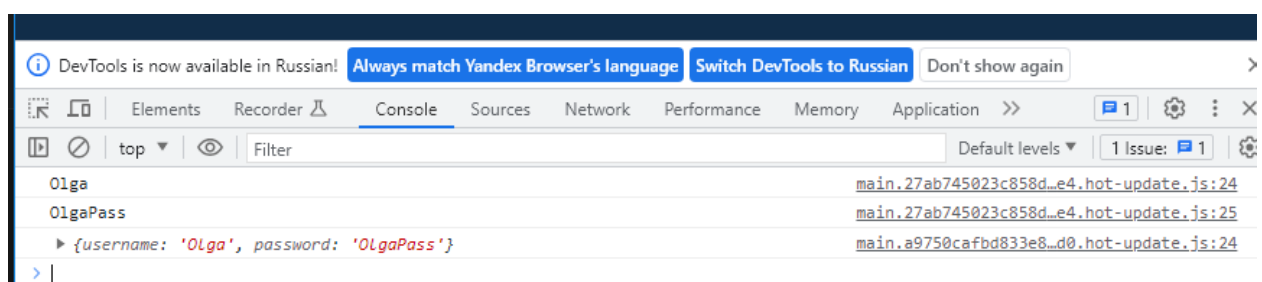
.....

Login



Выведем в консоль информацию из полей формы в виде объекта

```
JS Login.js x
src > components > JS Login.js > Login > handelFormSubmit
1 function Login() {
2   function handelFormSubmit(event) {
3     event.preventDefault();
4     console.log({
5       username: event.target.username.value,
6       password: event.target.password.value,
7     });
8   }
9   return (
10    <>
11    <h1>Login Form</h1>
12    <form onSubmit={handelFormSubmit}>
13      <label>
14        Username: <input type="text" name="username" />
15      </label>
16      <label>
17        Password: <input type="password" name="password" />
18      </label>
19      <button type="submit">Login</button>
20    </form>
21    </>
22  );
23 }
24
25 export default Login;
```



Теперь выведем объект при помощи alert на экран

```
JS Login.js x
src > components > JS Login.js > Login > handelFormSubmit
1 function Login() {
2   function handelFormSubmit(event) {
3     event.preventDefault();
4     const userData = {
5       username: event.target.username.value,
6       password: event.target.password.value,
7     };
8
9     console.log(userData);
10    alert(userData);
11  }
12  return (
13    <>
14      <h1>Login Form</h1>
15      <form onSubmit={handelFormSubmit}>
16        <label>
17          Username: <input type="text" name="username" />
18        </label>
19        <label>
20          Password: <input type="password" name="password" />
21        </label>
22        <button type="submit">Login</button>
23      </form>
24    </>
25  );
26 }
27
28 export default Login;
29
```

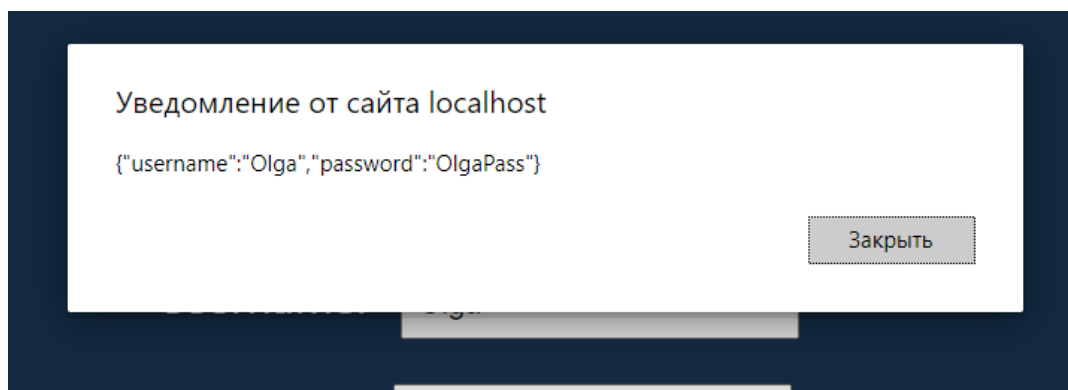
Уведомление от сайта localhost

[object Object]

Заккрыть

Видим, что alert не может отобразить значение переменной, которая указывает на объект. Для этого выполним конвертацию:

```
JS Login.js X
src > components > JS Login.js > ...
1  function Login() {
2    function handelFormSubmit(event) {
3      event.preventDefault();
4      const userData = {
5        username: event.target.username.value,
6        password: event.target.password.value,
7      };
8
9      console.log(userData);
10     alert(JSON.stringify(userData));
11   }
12   return (
13     <>
14     <h1>Login Form</h1>
15     <form onSubmit={handelFormSubmit}>
16       <label>
17         Username: <input type="text" name="username" />
18       </label>
19       <label>
20         Password: <input type="password" name="password" />
21       </label>
22       <button type="submit">Login</button>
23     </form>
24   </>
25 );
26 }
27
28 export default Login;
29
```



Неконтролируемые поля ни где не сохраняются, и мы берем эти значения из реального DOM только после нажатия на Login (т.е. после Submit). Таким образом получается, что в React мы ни где не отслеживаем, что пользователь вводит в полях формы, т.е. внешний вид приложения не соответствует состоянию приложения React. Это не рекомендуется делать в React, поэтому используются контролируемые поля.

Проект по контролируемым полям ввода (на основе 17-uncontrolled-input)

Что бы сделать поля формы контролируемыми необходимо добавить им состояние.

```
JS Login.js U X
18-controlled-input > src > components > JS Login.js > Login > handelFormSubmit
1  import { useState } from 'react';
2  function Login() {
3    const [user, setUser] = useState('');
4    const [password, setPassword] = useState('');
5    function handelFormSubmit(event) {
6      event.preventDefault();
7      const userData = {
8        username: event.target.username.value,
9        password: event.target.password.value,
10     };
11     // console.log(userData);
12     // alert(JSON.stringify(userData));
13   }
14   return (
15     <>
16     <h1>Login Form</h1>
17     <form onSubmit={handelFormSubmit}>
18     <label>
19       Username:
20       <input type="text" value={user} name="username" />
21     </label>
22     <label>
23       Password:
24       <input type="password" value={password}
25         name="password" />
26     </label>
27     <button type="submit">Login</button>
28   </form>
29   </>
30   );
31 }
32 export default Login;
```


Мы сделали поля формы контролируемыми. Форма работает, но при вводе ничего не отображается в полях. Это связано с тем, что при использовании `useState` задано значение по умолчанию – пустая строка.

```
JS Login.js U ●
18-controlled-input > src > components > JS Login.js > Login > handelFormSubmit
1  import { useState } from 'react';
2  function Login() {
3    const [user, setUser] = useState('');
4    const [password, setPassword] = useState('');
5    function handelFormSubmit(event) {
6      event.preventDefault();
7      const userData = {
8        username: event.target.username.value,
9        password: event.target.password.value,
10     };
11     console.log(userData);
12     // alert(JSON.stringify(userData));
13   }
14   return (
15     <>
16     <h1>Login Form</h1>
17     <form onSubmit={handelFormSubmit}>
18       <label>
19         Username:
20         <input
21           type="text"
22           value={user}
23           onChange={(e) => setUser(e.target.value)}
24           name="username"
25         />
26       </label>
27       <label>
28         Password:
29         <input type="password" value={password} name="password" />
30       </label>
31       <button type="submit">Login</button>
32     </form>
33   </>
34 );
35 }
```

← ↻ 🌐 localhost:3000 React App

Login Form

Username:

Password:

Login

DevTools is now available in Russian! Always match Yandex Browser's language Switch DevTools to Russian Don't show again

Elements Recorder Console Sources Network Performance Memory Components

Search (text or /regex/)

App

Login

Login

props

new entry: ""

hooks

1 State: "olga"

2 State: ""

rendered by

App

Самостоятельно добавить обработчик для поля Password

Так как теперь значения полей мы теперь берем из состояния компонента Login. Поэтому перепишем компонент следующим образом:

```
import { useState } from 'react';
function Login() {
  const [user, setUser] = useState('');
  const [password, setPassword] = useState('');
  function handelFormSubmit(event) {
    event.preventDefault();
    const userData = {
      user,
      password,
    };
    console.log(userData);
    // alert(JSON.stringify(userData));
  }
  return (
    <>
      <h1>Login Form</h1>
      <form onSubmit={handelFormSubmit}>
        <label>
          Username:
          <input
            type="text"
            value={user}
            onChange={(e) => setUser(e.target.value)}
          />
        </label>
        <label>
          Password:
          <input
            type="password"
            value={password}
            onChange={(e) => setPassword(e.target.value)}
          />
        </label>
        <button type="submit">Login</button>
      </form>
    </>
  );
};
```

Теперь компонент Login полностью контролирует значение полей формы.

Выполним оптимизацию проекта:

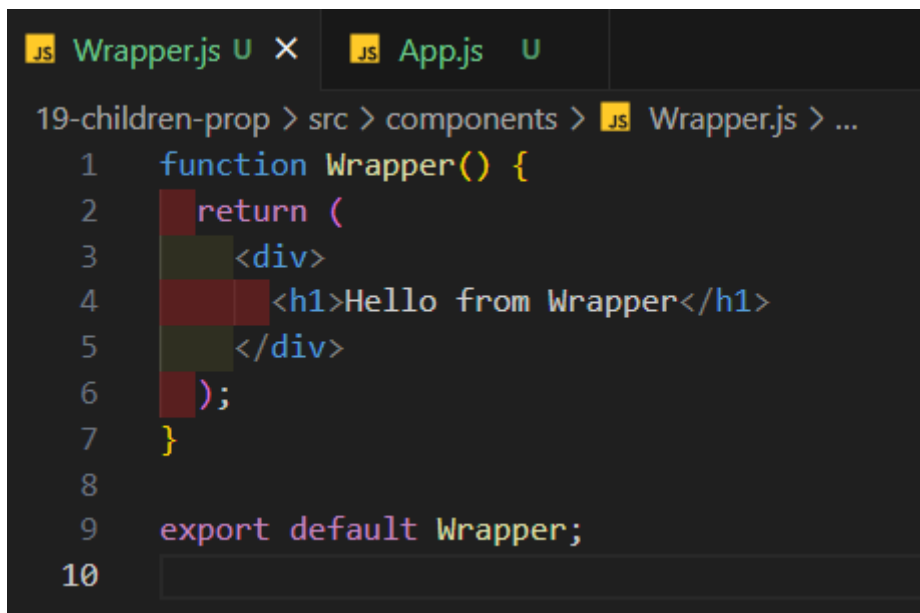
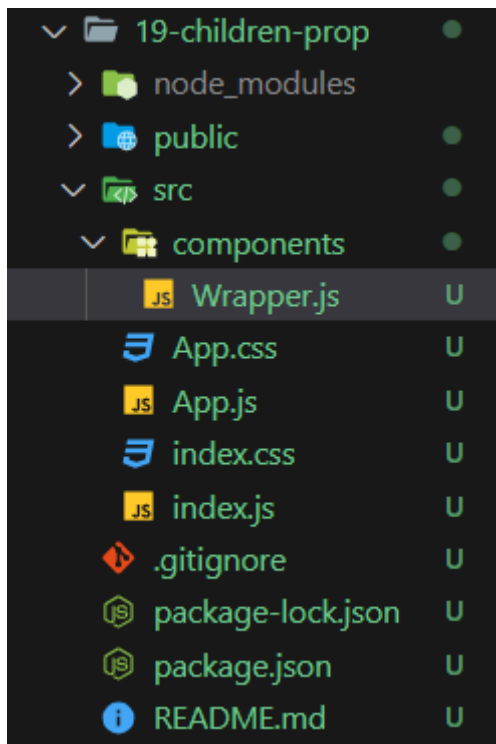
1. Избавимся от многократного вызова `useState`. Вызовем `useState` один раз и передадим объект как часть состояния компонента

```
JS Login.js M X
18-controlled-input > src > components > JS Login.js > Login
1 import { useState } from 'react';
2 function Login() {
3   const [data, setData] = useState({ user: '', password: '' });
4   function handelFormSubmit(event) {
5     event.preventDefault();
6     console.log(data);
7     // alert(JSON.stringify(data));
8   }
9   return (
10    <>
11      <h1>Login Form</h1>
12      <form onSubmit={handelFormSubmit}>
13        <label>
14          Username:
15          <input
16            type="text"
17            value={data.user}
18            onChange={(e) => setData({ ...data, user: e.target.value })}
19          />
20        </label>
21        <label>
22          Password:
23          <input
24            type="password"
25            value={data.password}
26            onChange={(e) => setData({ ...data, password: e.target.value })}
27          />
28        </label>
29        <button type="submit">Login</button>
30      </form>
31    </>
32  );
33 }
34
35 export default Login;
```

2. Уберем однотипное действие – вызовы (e) => setData

```
JS Login.js M X
18-controlled-input > src > components > JS Login.js > Login
1  import { useState } from 'react';
2  function Login() {
3    const [data, setData] = useState({ user: '', password: '' });
4    function handelFormSubmit(event) {
5      event.preventDefault();
6      console.log(data);
7      // alert(JSON.stringify(data));
8    }
9
10   function handelInputChange(e, name) {
11     setData({ ...data, [name]: e.target.value });
12   }
13   return (
14     <>
15       <h1>Login Form</h1>
16       <form onSubmit={handelFormSubmit}>
17         <label>
18           Username:
19           <input
20             type="text"
21             value={data.user}
22             onChange={(e) => handelInputChange(e, 'user')}
23           />
24         </label>
25         <label>
26           Password:
27           <input
28             type="password"
29             value={data.password}
30             onChange={(e) => handelInputChange(e, 'password')}
31           />
32         </label>
33         <button type="submit">Login</button>
34       </form>
35     </>
36   );
37 }
38
39 export default Login;
```

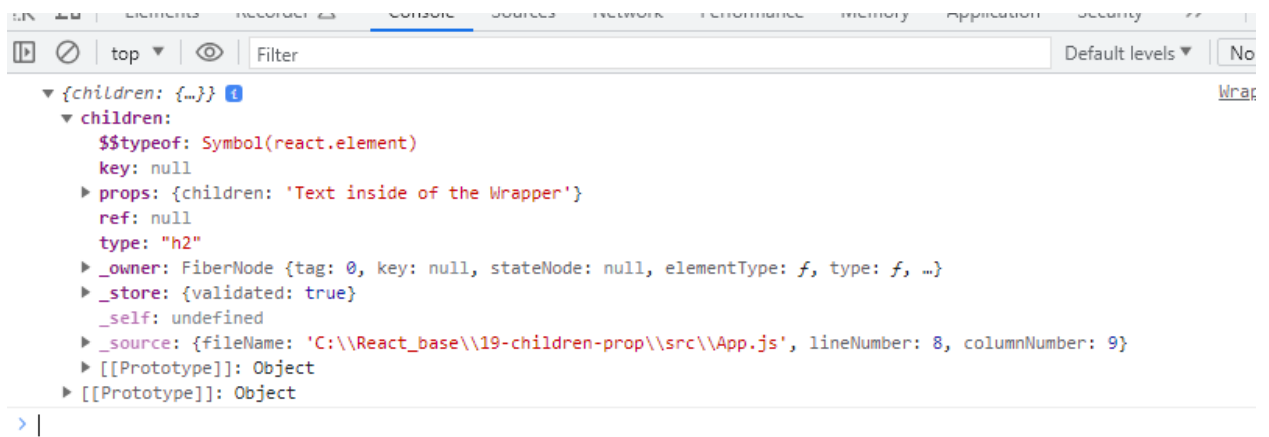
Проект с использованием свойства children



Добавим в компонент Wrapper внутренний компонент

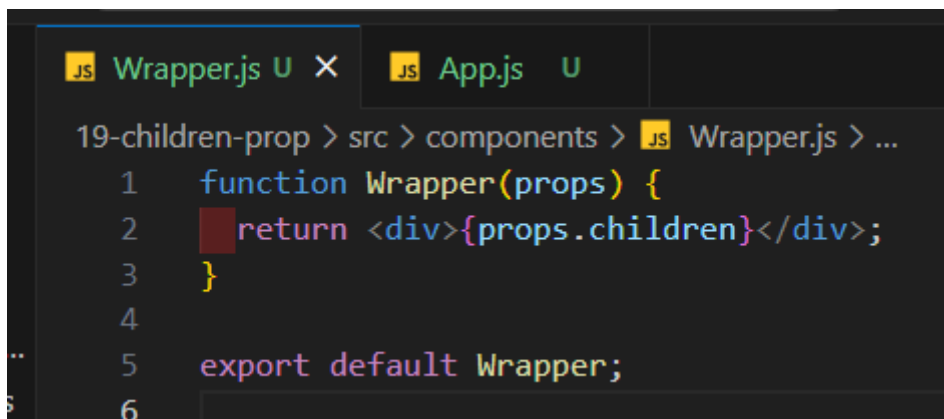
```
JS Wrapper.js U App.js U X
19-children-prop > src > JS App.js > App
1  import './App.css';
2  import Wrapper from './components/Wrapper';
3
4  function App() {
5    return (
6      <div className="App">
7        <Wrapper>
8          <h2>Text inside of the Wrapper</h2>
9        </Wrapper>
10     </div>
11   );
12 }
13
14 export default App;
15
```

```
JS Wrapper.js U X App.js U
19-children-prop > src > components > JS Wrapper.js > ...
1  function Wrapper(props) {
2    console.log(props);
3    return (
4      <div>
5        <h1>Hello from Wrapper</h1>
6      </div>
7    );
8  }
9
10 export default Wrapper;
11
```

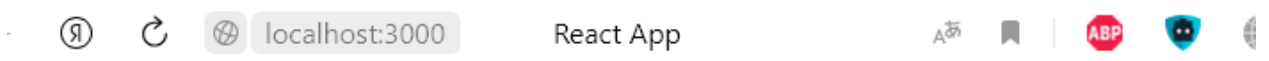


Т.е. видно, что JSX-код `<h2>Text inside of the Wrapper</h2>` передается в компонент `Wrapper` через специальное свойство `children`

Но на странице переданный компонент `<h2>Text inside of the Wrapper</h2>` не отображается. Это значит что компонент `Wrapper` не контролирует что добавляется в компонент. Внесем следующие изменения



Теперь видим текст



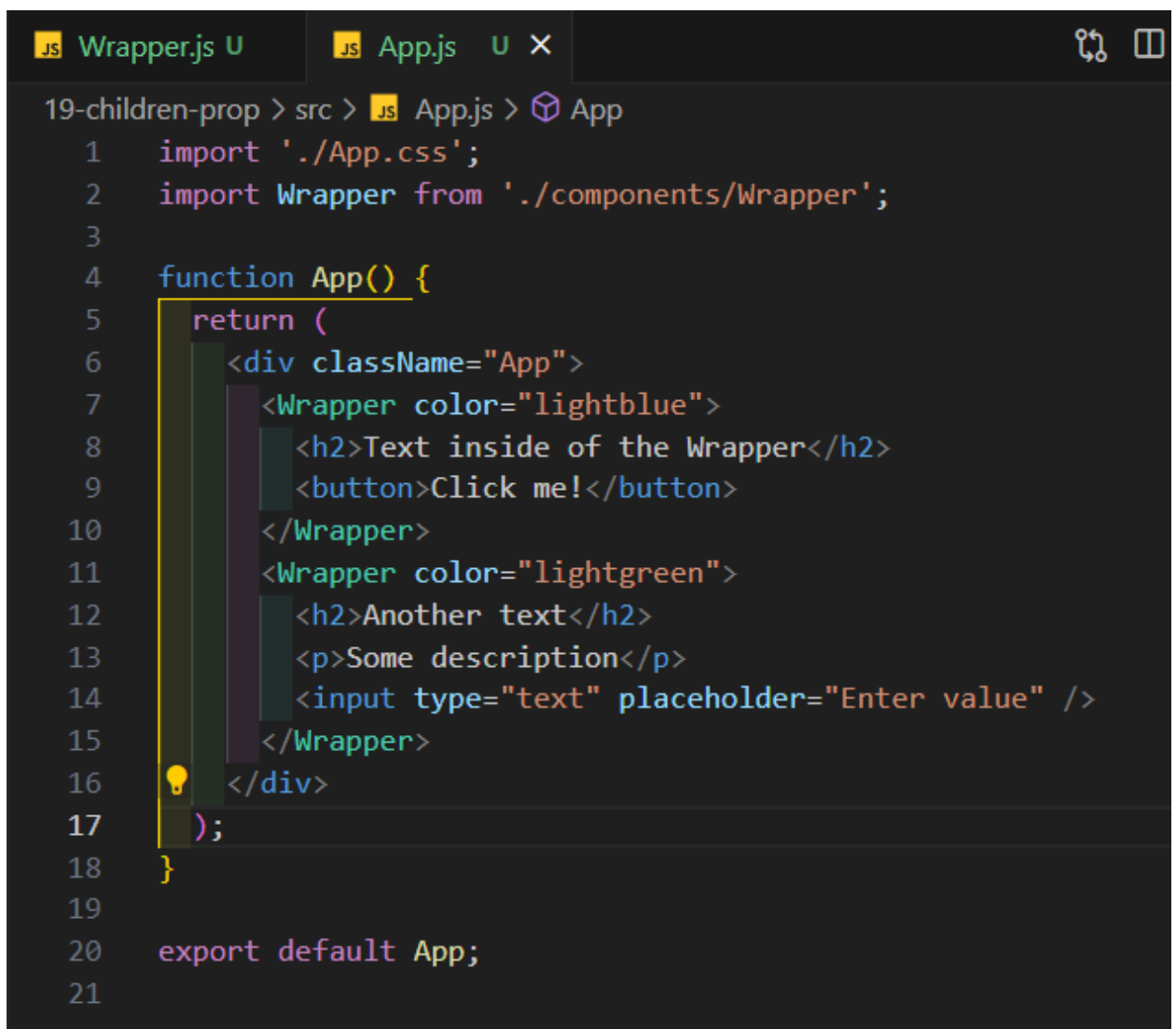
Text inside of the Wrapper


```

<!DOCTYPE html>
<html lang="en">
  <head>...</head>
  <body>
    <div id="root">
      <div class="App">
        <div> == $0
          <h2>Text inside of the Wrapper</h2>
          <button>Click me!</button>
        </div>
      </div>
    </div>
  </body>
</html>

```

Добавим дополнительные свойства для компонента Wrapper



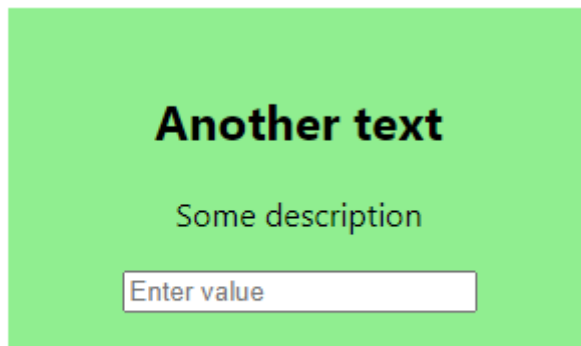
```

JS Wrapper.js U    JS App.js U X
19-children-prop > src > JS App.js > App
1  import './App.css';
2  import Wrapper from './components/Wrapper';
3
4  function App() {
5    return (
6      <div className="App">
7        <Wrapper color="lightblue">
8          <h2>Text inside of the Wrapper</h2>
9          <button>Click me!</button>
10         </Wrapper>
11        <Wrapper color="lightgreen">
12          <h2>Another text</h2>
13          <p>Some description</p>
14          <input type="text" placeholder="Enter value" />
15        </Wrapper>
16      </div>
17    );
18  }
19
20  export default App;
21

```

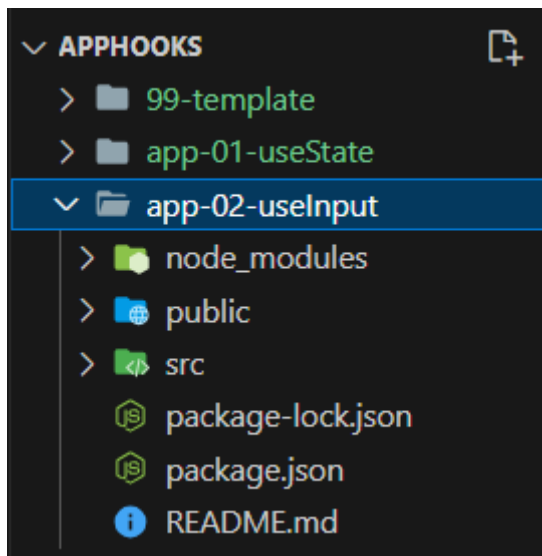
Самостоятельно:

1. Выполнить стилевое оформление до вида



2. Добавить два компонента Wrapper с произвольными содержимым и стилями.

Создание собственного хука useInput



Hello

JS App.js X

app-02-useInput > src > JS App.js > App

```
1  import { useState } from 'react';
2  import React from 'react';
3
4  const useInput = (initialValue) => {
5    const [value, setValue] = useState(initialValue);
6    const onChange = (event) => {
7      console.log(event.target);
8    };
9    return { value, onChange };
10
11  export default function App() {
12    const name = useInput('Mr. ');
13
14    return (
15      <div className="App">
16        <h1>Hello</h1>
17        <input placeholder="Имя" {...name} />
18      </div>
19    );
20  }
21
```

