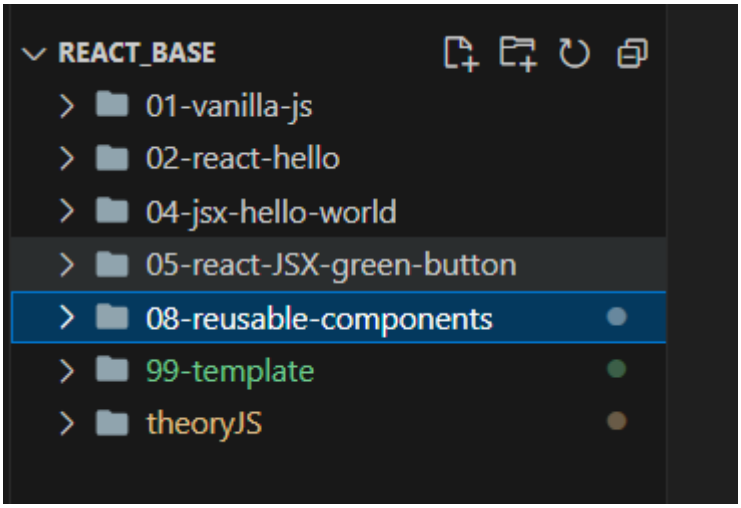
Однократное и повторное использование компонентов



08-reusable-components

Однократное использование компонента MyComponent

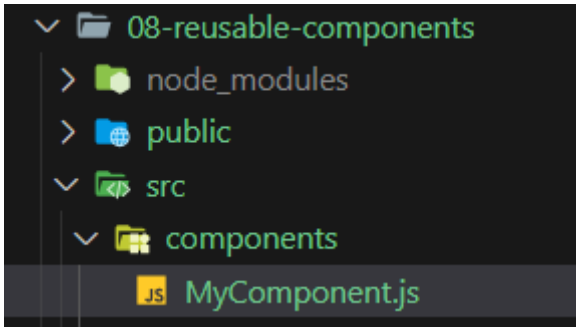
App.js U

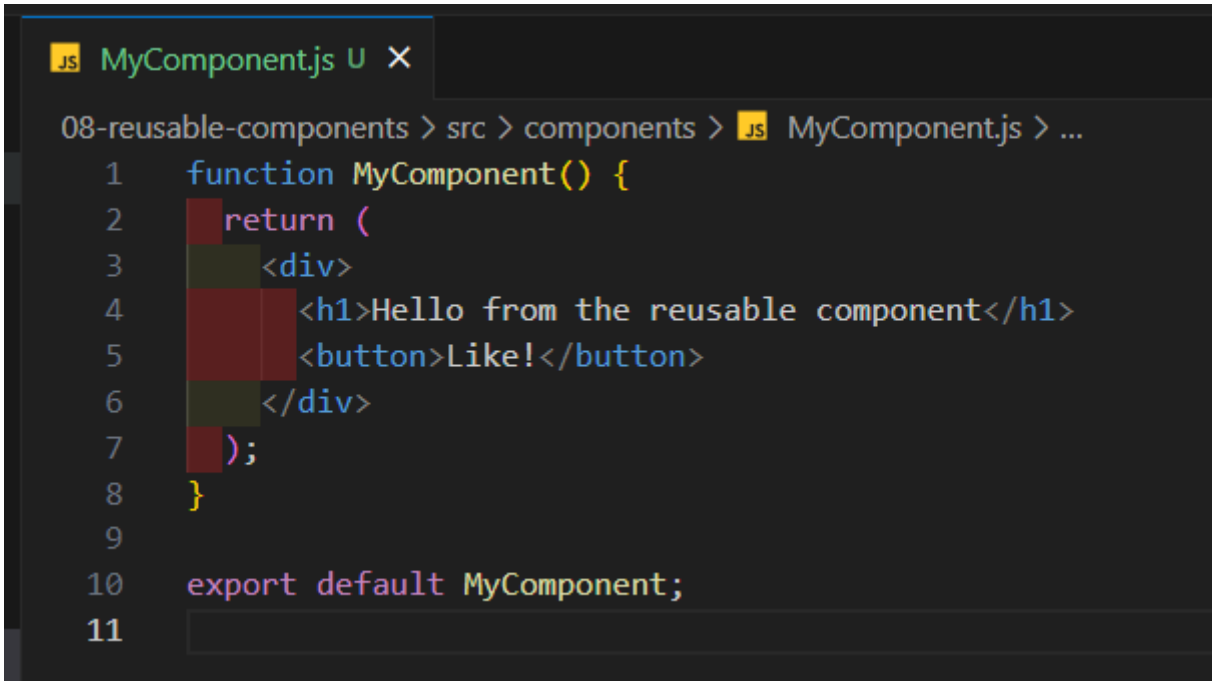
08-reusable-components > src > App.js > ...

```
1   import './App.css';
2
3   function MyComponent() {
4     return (
5       <div>
6         <h1>Hello from the reusable component</h1>
7         <button>Like!</button>
8       </div>
9     );
10  }
11
12  function App() {
13    return (
14      <div className="App">
15        <MyComponent />
16      </div>
17    );
18  }
19
20  export default App;
```

Самостоятельно: выполнить повторное (многократное) использование компонента MyComponent

Вынесем компонент в отдельный файл



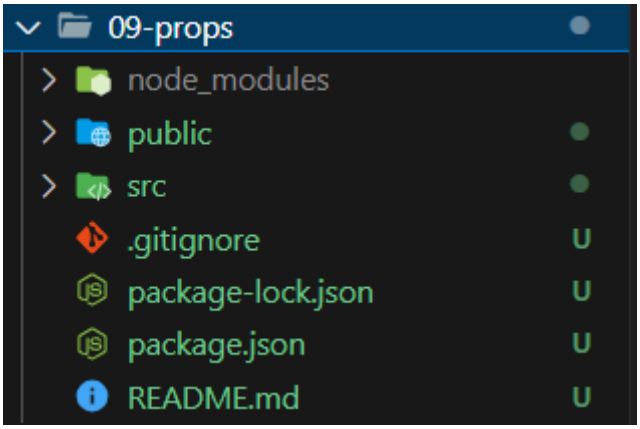


Самостоятельно:

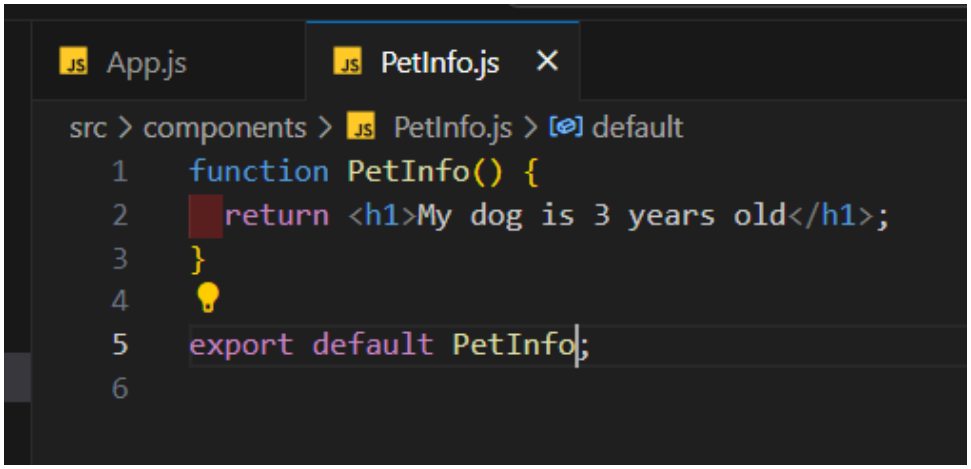
1. выполнить подключение компонента MyComponent (import)
2. создать еще один компонент (например, OtherComponent.js) и выполнить его подключение

Передача свойств компоненту

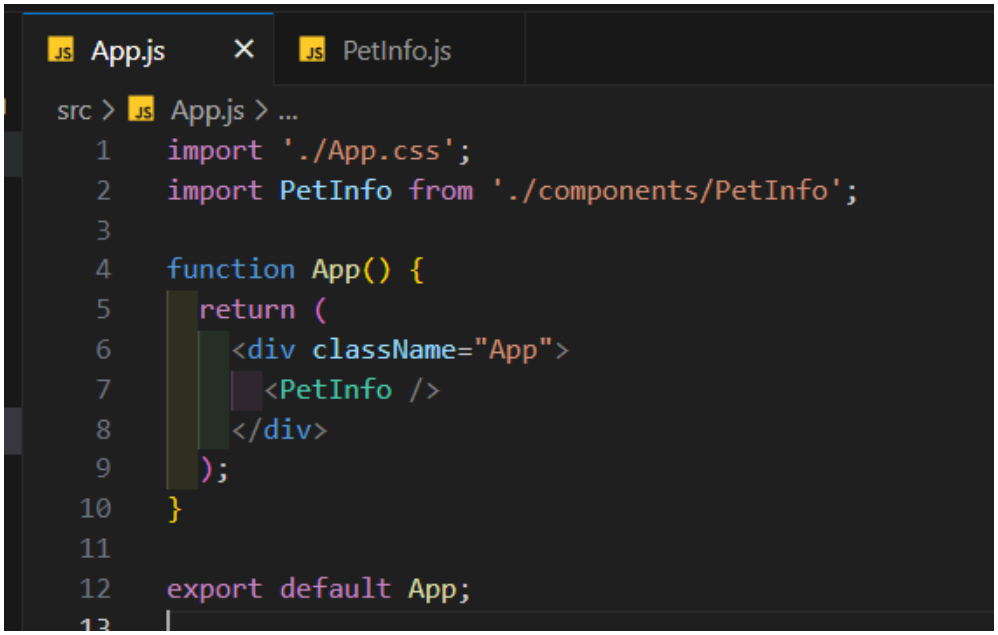
09-props



Создадим компонент PetInfo



Выполним его подключение



Теперь через props передадим в компонент свойство animal

```
<div className="App">
  <PetInfo animal="cat" />
</div>
```

А в компоненте PetInfo примем это свойство

App.js | PetInfo.js

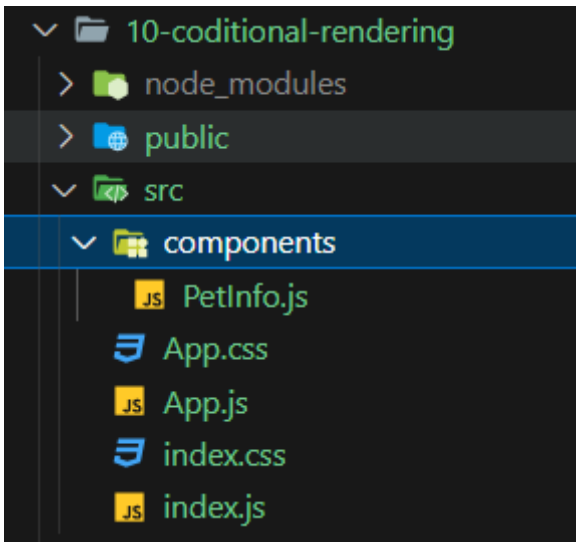
src > components > PetInfo.js > ...

```
1  function PetInfo(props) {
2    return <h1>My {props.animal} is 3 years old</h1>;
3  }
4
5  export default PetInfo;
6
```

Самостоятельно:

1. повторно использовать этот же компонент для dog возрастом 5 лет
2. выполнить деструктуризацию свойств

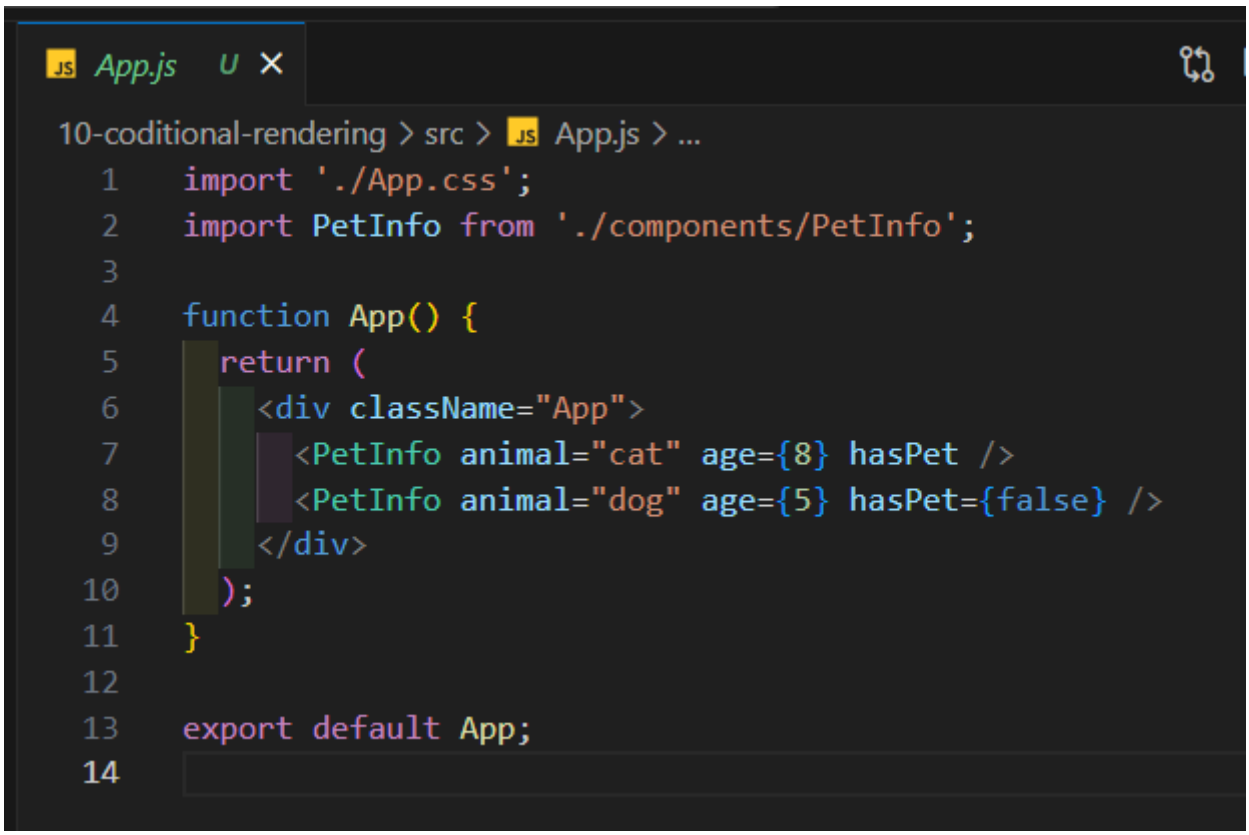
Проект по условному возврату JSX



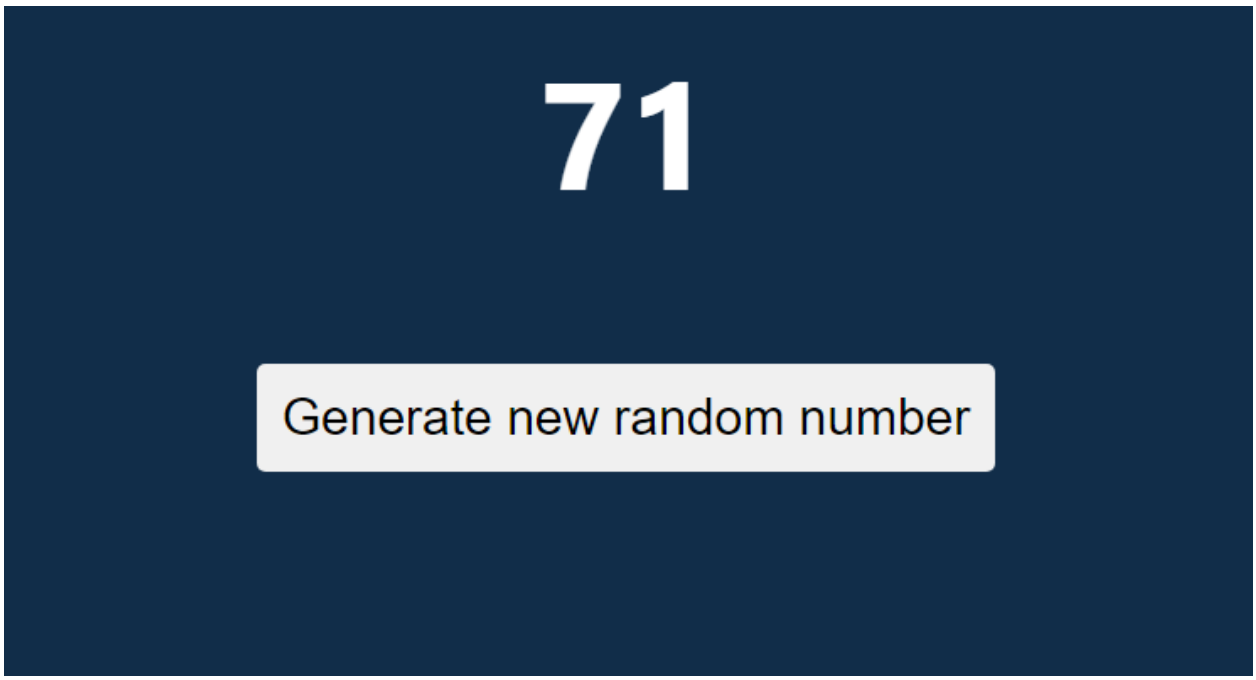
PetInfo.js U

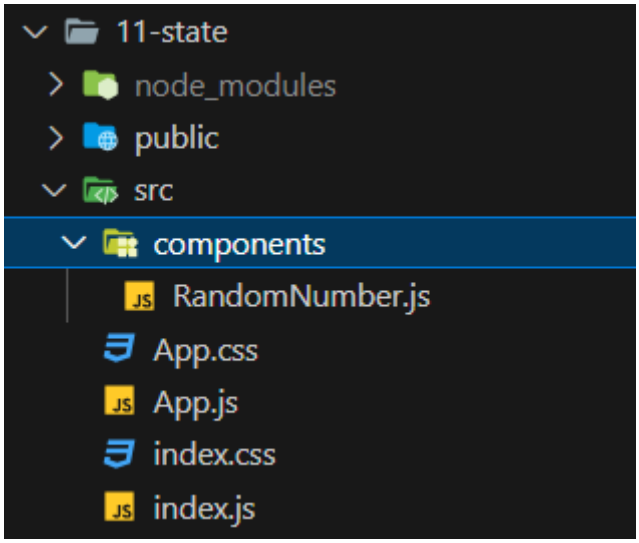
10-coditional-rendering > src > components > PetInfo.js > ...

```
1   function PetInfo({ animal, age, hasPet }) {
2     return hasPet ? (
3       <h1>
4         My {animal} is {age} years old
5       </h1>
6     ) : (
7       <h1>I dont't have an animal</h1>
8     );
9   }
10
11  export default PetInfo;
12
```


App.js U
10-coditional-rendering > src > App.js > ...
1 import './App.css';
2 import PetInfo from './components/PetInfo';
3
4 function App() {
5 return (
6
7 <PetInfo animal="cat" age={8} hasPet />
8 <PetInfo animal="dog" age={5} hasPet={false} />
9
10);
11 }
12
13 export default App;
14

Проект с состоянием компонента





App.css

11-state > src > App.css > button

```css
.App {
  text-align: center;
}

button {
  padding: 15px;
  font-size: 30px;
  border-radius: 5px;
  border: 0;
  margin: 20px;
  cursor: pointer;
}

h1 {
  font-size: 90px;
  margin-top: 20px;
}

```

App.js

11-state > src > App.js > ...

```jsx
import './App.css';
import RandomNumber from './components/RandomNumber';

function App() {
  return (
    <div className="App">
      <RandomNumber />
    </div>
  );
}

export default App;

```

index.css

11-state > src > index.css > ...

```css
body {
  margin: 0;
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI',
  'Roboto', 'Oxygen',
    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans',
    'Helvetica Neue',
    sans-serif;
  color: white;
  background-color: #112d49;
}

```

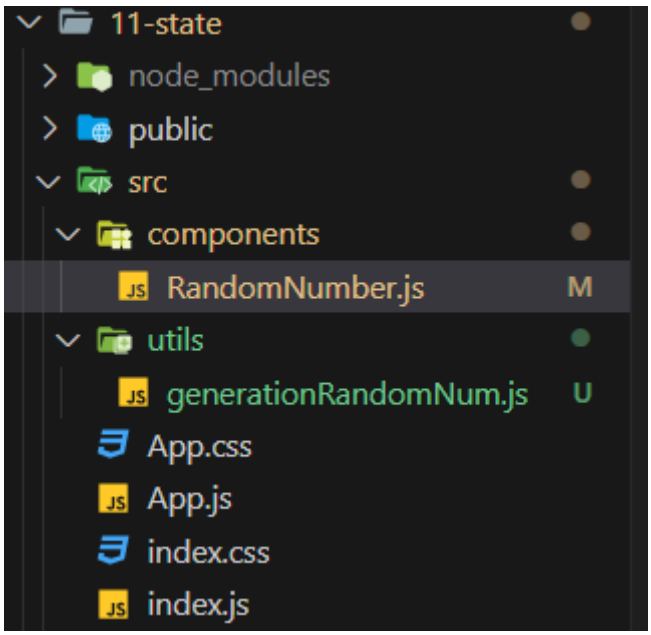
RandomNumber.js

11-state > src > components > RandomNumber.js > RandomNumber

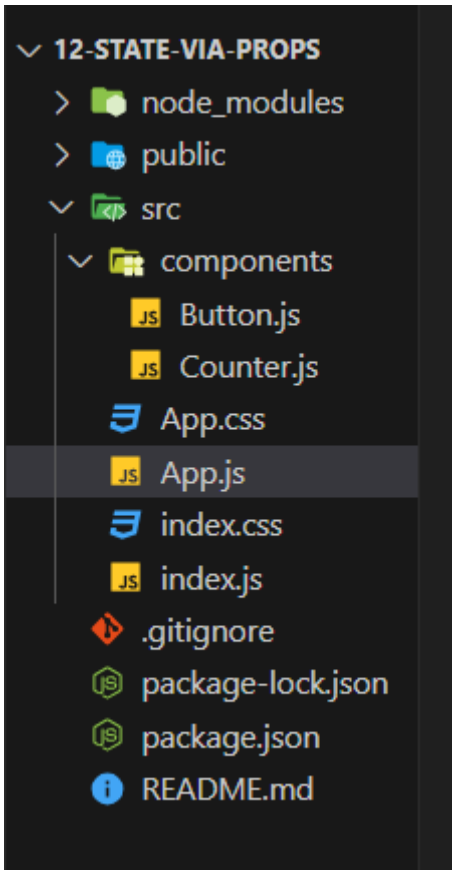
```jsx
import { useState } from 'react';

function generateRandomNum() {
  return Math.floor(Math.random() * 100);
}
function RandomNumber() {
  const [randomNum, setRandomNum] = useState(generateRandomNum());
  const changeRandomNum = () => {
    setRandomNum(generateRandomNum());
  };

  return (
    <div>
      <h1>{randomNum}</h1>
      <button onClick={changeRandomNum}>Generate new random number</button>
    </div>
  );
}

export default RandomNumber;
```

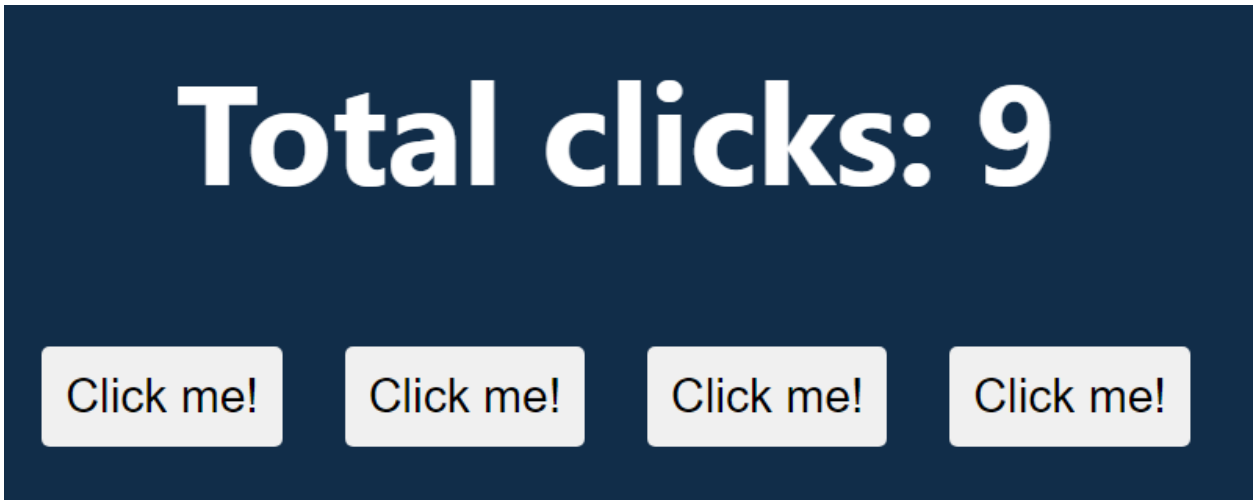
Самостоятельно:

1. Вынести функцию generateRandomNum() в отдельный файл согласно следующей структуре папок
2. Изменить диапазон для случайного числа, передав его как свойство (props) из родительского компонента. Назовите свойство maxNum.


11-state
node_modules
public
src
components
RandomNumber.js
M
utils
generationRandomNum.js
U
App.css
App.js
index.css
index.js

Проект по передаче состояния через свойства





App.js | Counter.js | Button.js

src > App.js > App

```jsx
import './App.css';
import Button from './components/Button';
import Counter from './components/Counter';

function App() {
  return (
    <div className="App">
      <Counter />
      <Button />
      <Button />
      <Button />
      <Button />
    </div>
  );
}

export default App;

```

App.js | Counter.js | Button.js

src > components > Counter.js > default

```jsx
function Counter() {
  return <h1>Total clicks: 0</h1>;
}

export default Counter;

```

App.js | Counter.js | Button.js

src > components > Button.js > default

```jsx
function Button() {
  return <button>Click me!</button>;
}

export default Button;

```

2

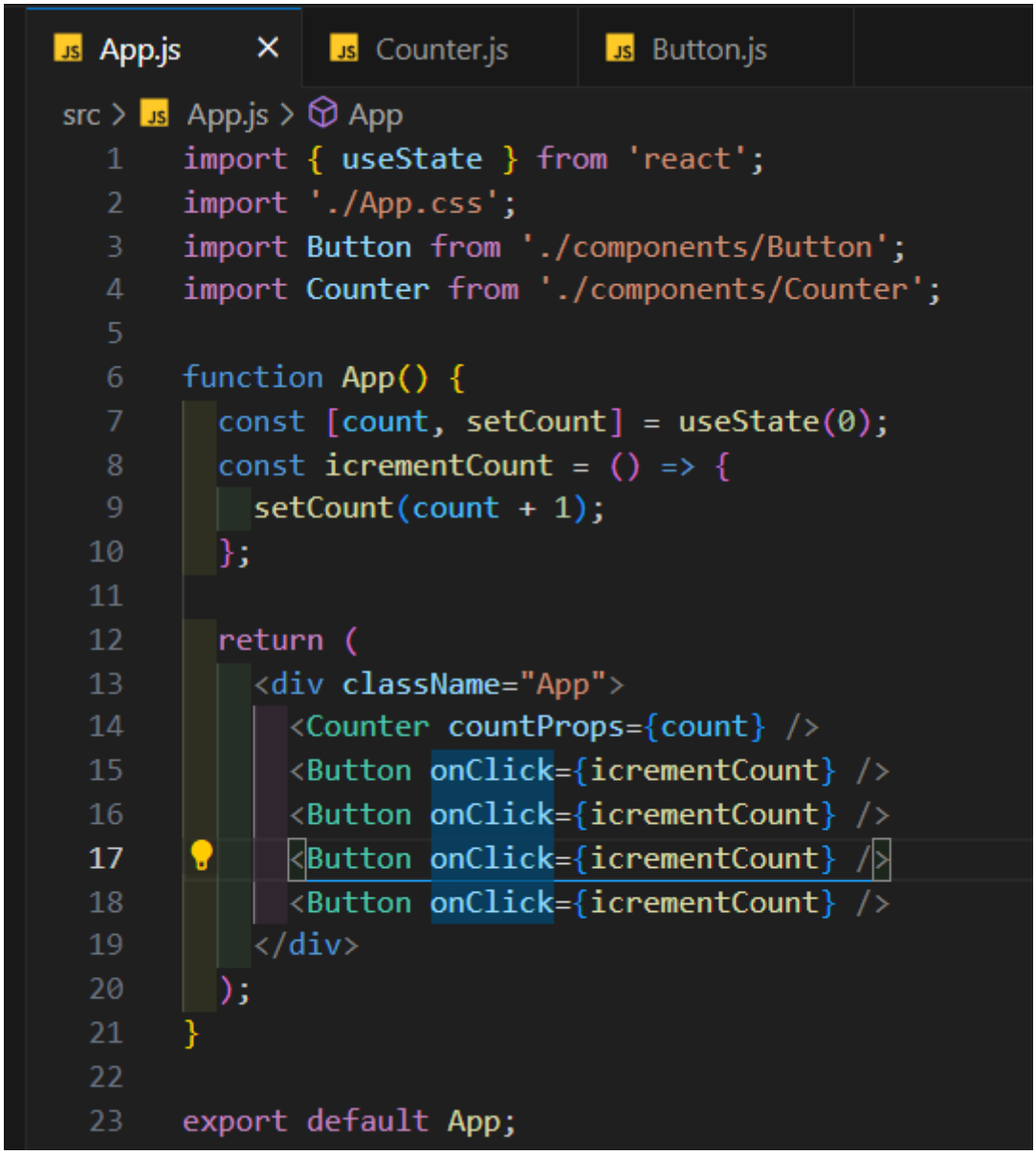
App.js

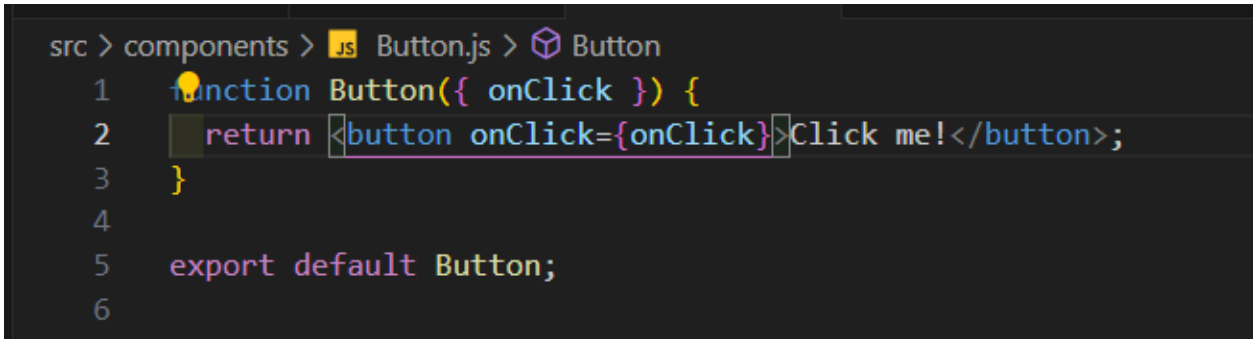
```
src > App.js > App
1   import { useState } from 'react';
2   import './App.css';
3   import Button from './components/Button';
4   import Counter from './components/Counter';
5
6   function App() {
7     const [count, setCount] = useState(0);
8
9     return (
10      <div className="App">
11        <Counter countProps={count} />
12        <Button countProps={count} onClick={setCount} />
13        <Button countProps={count} onClick={setCount} />
14        <Button countProps={count} onClick={setCount} />
15        <Button countProps={count} onClick={setCount} />
16      </div>
17    );
18  }
19
20  export default App;
21
```

Button.js

```
src > components > Button.js > Button
1   function Button({ countProps, onClick }) {
2     return <button onClick={() => onClick(countProps + 1)}>Click me!</button>;
3   }
4
5   export default Button;
6
```

Изолирование состояния в родительском компоненте



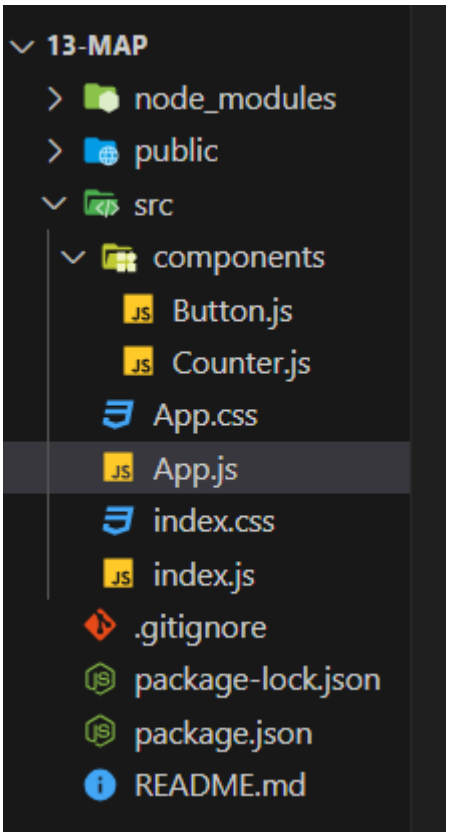


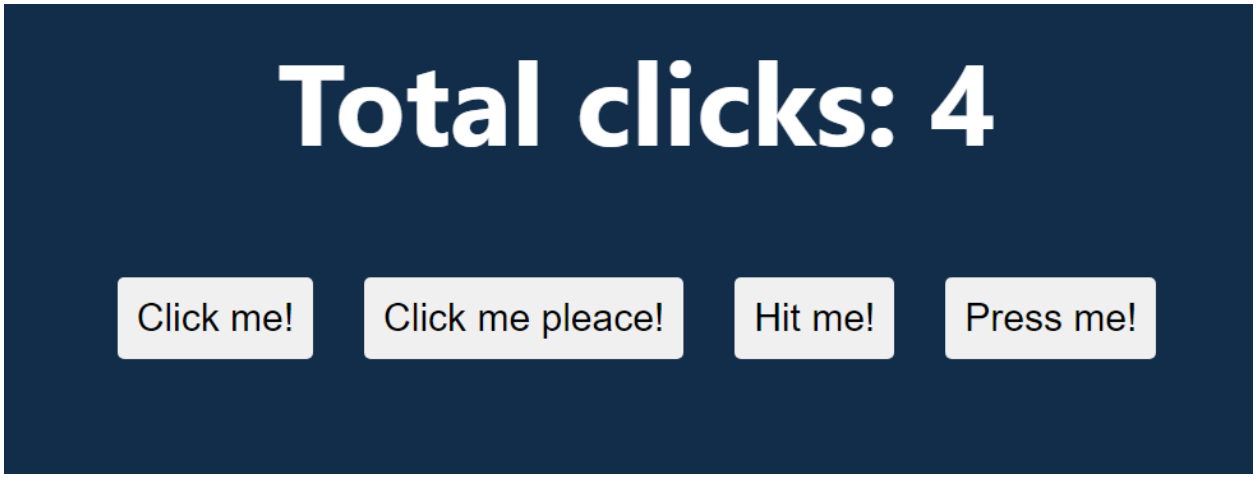
Очередность рендеринга компонент

React extension Google (https://chrome.google.com/webstore/detail/react-developer-tools/fmkadmapgofadopljbjfkapdkoienihi/related)

Отображение данных из массивов

(на основе 12-state-via-props)





Сначала передадим вручную свойство text для каждого компонента

App.js Button.js

src > App.js > App

```
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 icrementCount = () => {
9      setCount(count + 1);
10   };
11
12   return (
13     <div className="App">
14       <Counter countProps={count} />
15       <Button onClick={icrementCount} text="Click me!" />
16       <Button onClick={icrementCount} text="Click me pleace!" />
17       <Button onClick={icrementCount} text="Hit me!" />
18       <Button onClick={icrementCount} text="Press me!" />
19     </div>
20   );
21 }
22
23 export default App;
24
```

App.js Button.js

src > components > Button.js > Button

```
1  function Button({ onClick, text }) {
2    return <button onClick={onClick}>{text}</button>;
3  }
4
5  export default Button;
6
```

Затем вынесем все значения свойств в отдельный массив и в качестве значения свойств компонентов передадим обращение к элементам этого массива:

App.js Button.js

src > App.js > App

```
1  import { useState } from 'react';
2  import './App.css';
3  import Button from './components/Button';
4  import Counter from './components/Counter';
5
6  const text = ['Click me!', 'Click me pleace!', 'Hit me!', 'Press me!'];
7  function App() {
8    const [count, setCount] = useState(0);
9    const icrementCount = () => {
10     setCount(count + 1);
11   };
12
13   return (
14     <div className="App">
15       <Counter countProps={count} />
16       {text.map((text) => {
17         return <Button onClick={icrementCount} text={text} />;
18       })}
19     </div>
20   );
21 }
22
23 export default App;
24
```

Самостоятельно вывести в браузер еще две кнопки: