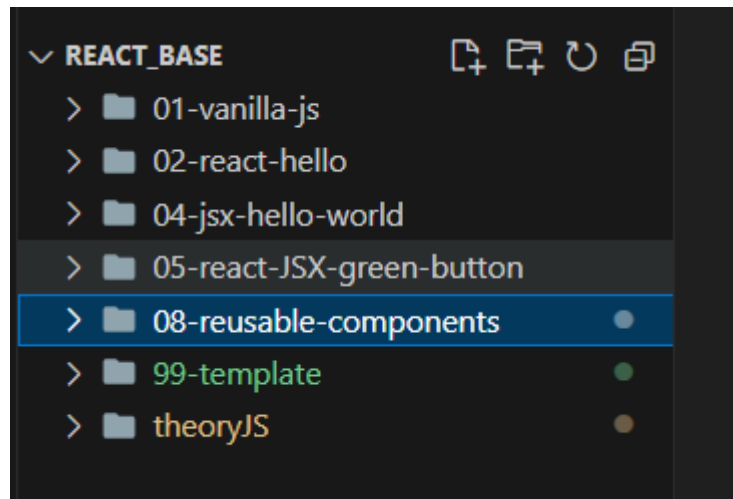


## Однократное и повторное использование компонентов



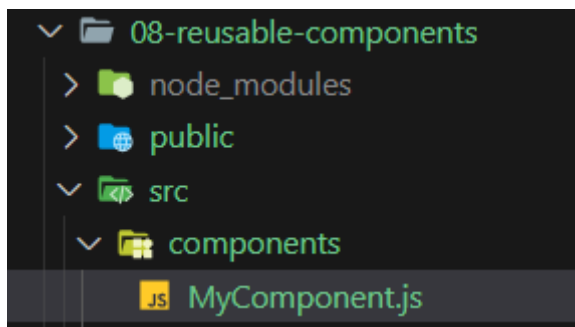
### 08-reusable-components

Однократное использование компонента MyComponent

```
JS App.js U X
08-reusable-components > src > JS 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



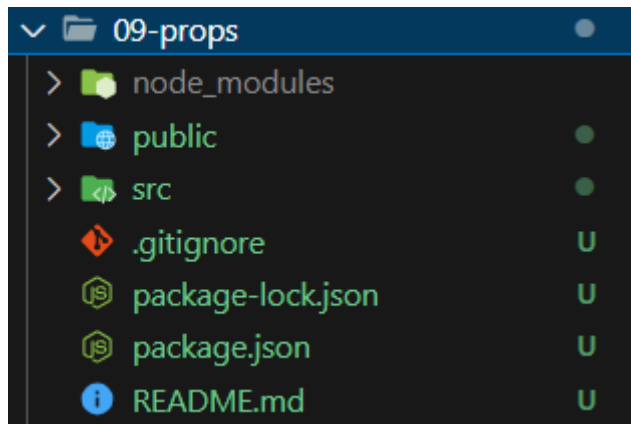
```
JS MyComponent.js U X
08-reusable-components > src > components > JS MyComponent.js > ...
1  function MyComponent() {
2      return (
3          <div>
4              <h1>Hello from the reusable component</h1>
5              <button>Like!</button>
6          </div>
7      );
8  }
9
10 export default MyComponent;
11
```

Самостоятельно:

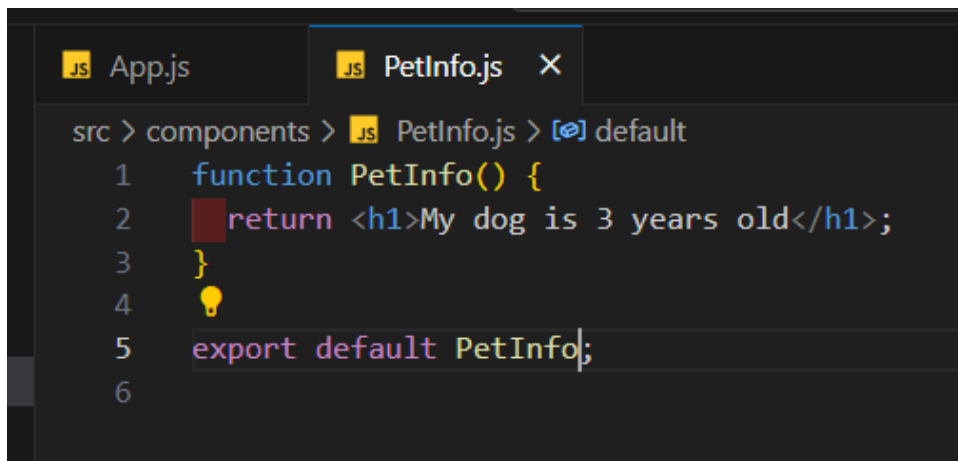
1. выполнить подключение компонента MyComponent (import)
2. создать еще один компонент (например, OtherComponent.js) и выполнить его подключение

## Передача свойств компоненту

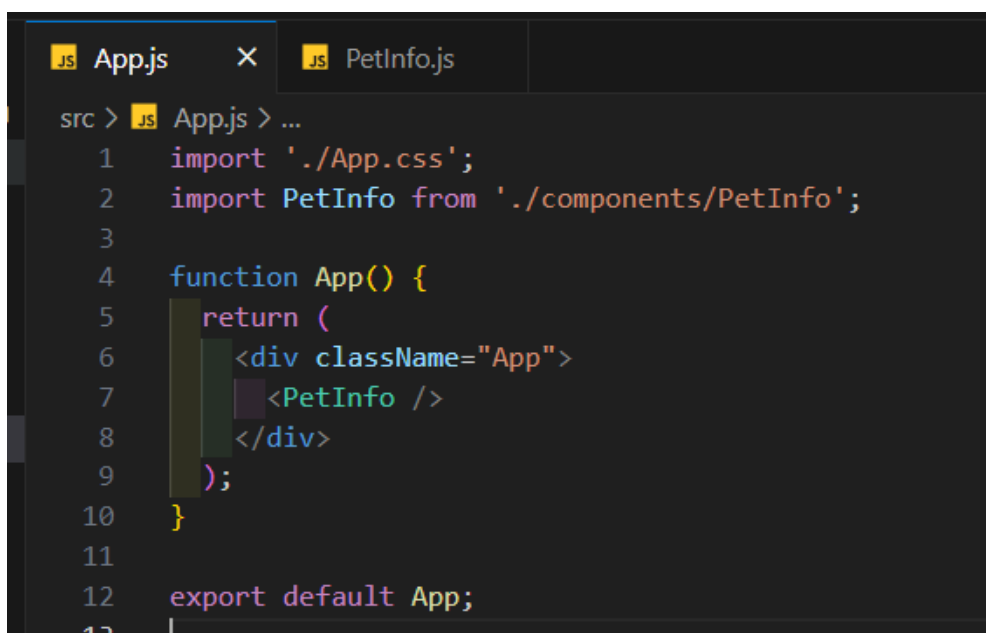
09-props



Создадим компонент PetInfo



Выполним его подключение



Теперь через props передадим в компонент свойство animal

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

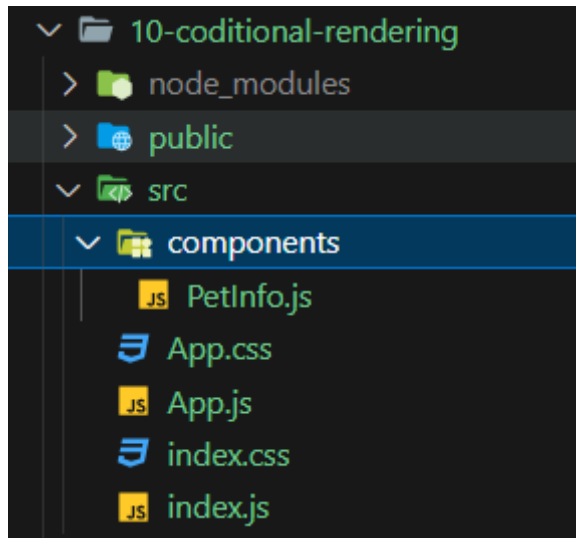
А в компоненте PetInfo примем это свойство

```
JS App.js  JS PetInfo.js X  
src > components > JS 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



```
JS PetInfo.js U X
10-coditional-rendering > src > components > JS 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
```

JS App.js U X



10-conditional-rendering > src > JS App.js > ...

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

# 71

Generate new random number

- 11-state
  - node\_modules
  - public
  - src
    - components
      - RandomNumber.js
      - App.css
      - App.js
      - index.css
      - index.js

```
App.css X
11-state > src > App.css > button
1  .App {
2    text-align: center;
3  }
4
5  button {
6    padding: 15px;
7    font-size: 30px;
8    border-radius: 5px;
9    border: 0;
10   margin: 20px;
11   cursor: pointer;
12 }
13
14 h1 {
15   font-size: 90px;
16   margin-top: 20px;
17 }
18
```

```
App.js X
11-state > src > App.js > ...
1  import './App.css';
2  import RandomNumber from './components/RandomNumber';
3
4  function App() {
5    return (
6      <div className="App">
7        <RandomNumber />
8      </div>
9    );
10 }
11
12 export default App;
13
```

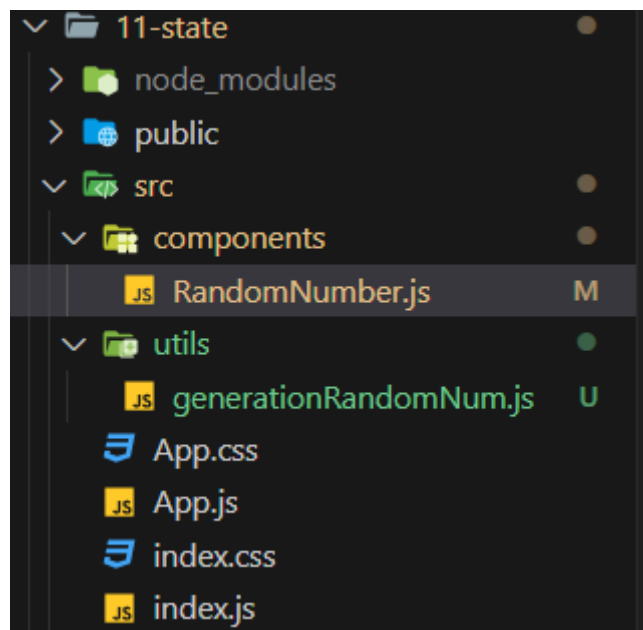


```
index.css ×
11-state > src > index.css > ...
1  body {
2    margin: 0;
3    font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI',
4      'Roboto', 'Oxygen',
5      'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans',
6      'Helvetica Neue',
7      sans-serif;
8    color: white;
9    background-color: #112d49;
```

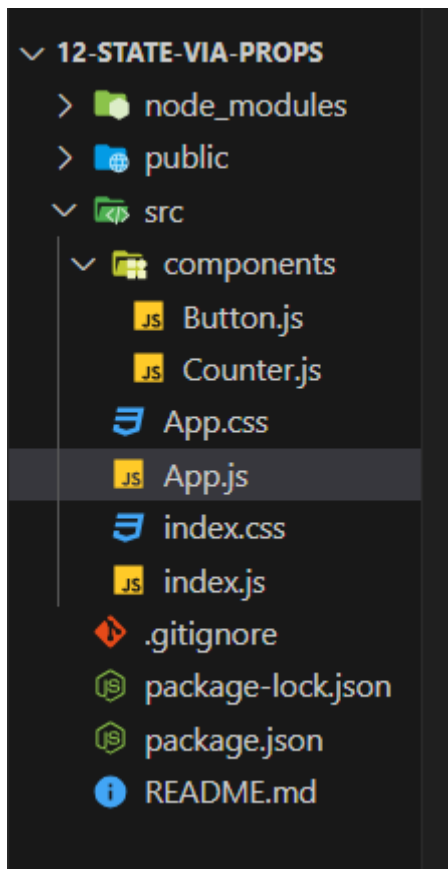
```
RandomNumber.js ×
11-state > src > components > RandomNumber.js > RandomNumber
1  import { useState } from 'react';
2
3  function generateRandomNum() {
4    return Math.floor(Math.random() * 100);
5  }
6  function RandomNumber() {
7    const [randomNum, setRandomNum] = useState(generateRandomNum());
8    const changeRandomNum = () => {
9      setRandomNum(generateRandomNum());
10   };
11
12   return (
13     <div>
14       <h1>{randomNum}</h1>
15       <button onClick={changeRandomNum}>Generate new random number</button>
16     </div>
17   );
18 }
19
20 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`.



## Проект по передаче состояния через свойства



# Total clicks: 9

Click me!

Click me!

Click me!

Click me!

```
JS App.js × JS Counter.js JS Button.js
src > JS App.js > App
1 import './App.css';
2 import Button from './components/Button';
3 import Counter from './components/Counter';
4
5 function App() {
6   return (
7     <div className="App">
8       <Counter />
9       <Button />
10      <Button />
11      <Button />
12      <Button />
13    </div>
14  );
15 }
16
17 export default App;
18
```

```
JS App.js JS Counter.js × JS Button.js
src > components > JS Counter.js > default
1 function Counter() {
2   return <h1>Total clicks: 0</h1>;
3 }
4
5 export default Counter;
6
```

```
JS App.js JS Counter.js JS Button.js ×
src > components > JS Button.js > default
1 function Button() {
2   return <button>Click me!</button>;
3 }
4
5 export default Button;
6
```

```

src > JS 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

```

```

src > components > JS 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

```

```

13-map > src > components > JS Counter.js > Counter
1  function Counter({ countProps }) {
2    return <h1>Total clicks: {countProps}</h1>;
3  }
4
5  export default Counter;
6

```

## Изолирование состояния в родительском компоненте

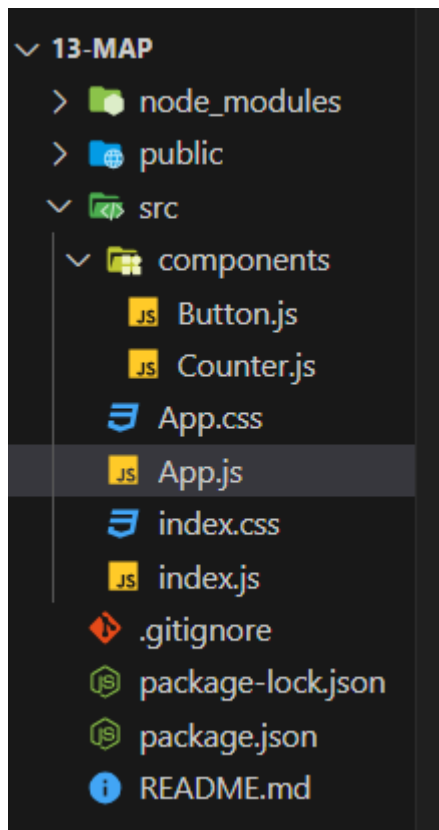
```
src > JS App.js > App
1  import { useState } from 'react';
2  import './App.css';
3  import Button from './components/Button';
4  import Counter from './components/Counter';
5
6  function App() {
7    const [count, setCount] = useState(0);
8    const incrementCount = () => {
9      setCount(count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

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

## 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)



# Total clicks: 4

Click me!

Click me please!

Hit me!

Press me!

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

Самостоятельно вывести в браузер еще две кнопки:

