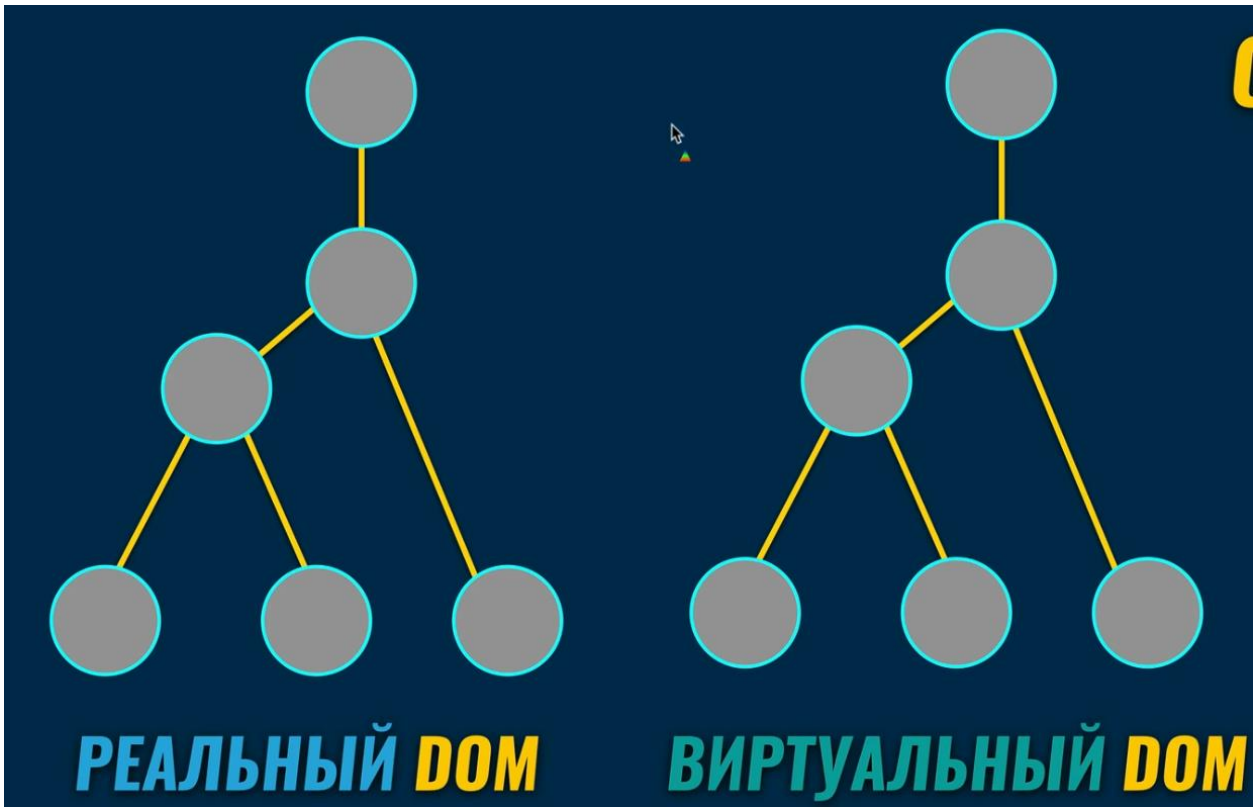
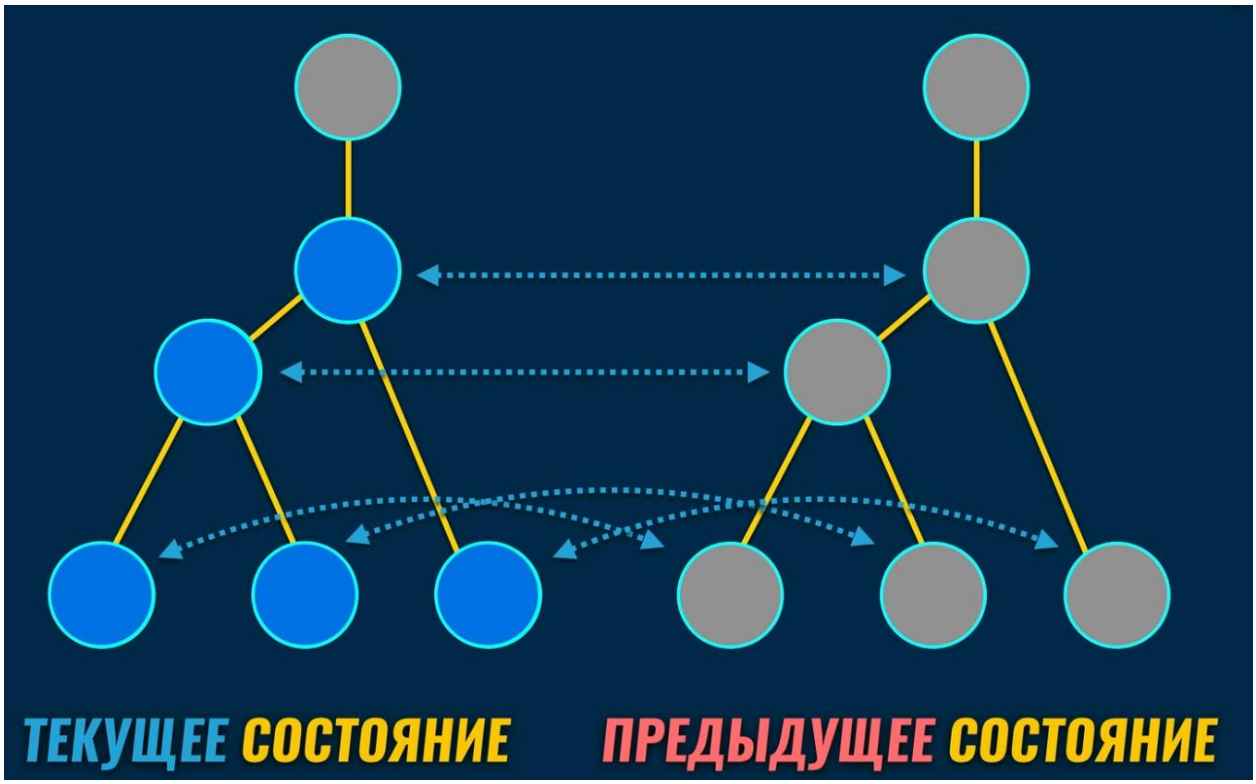
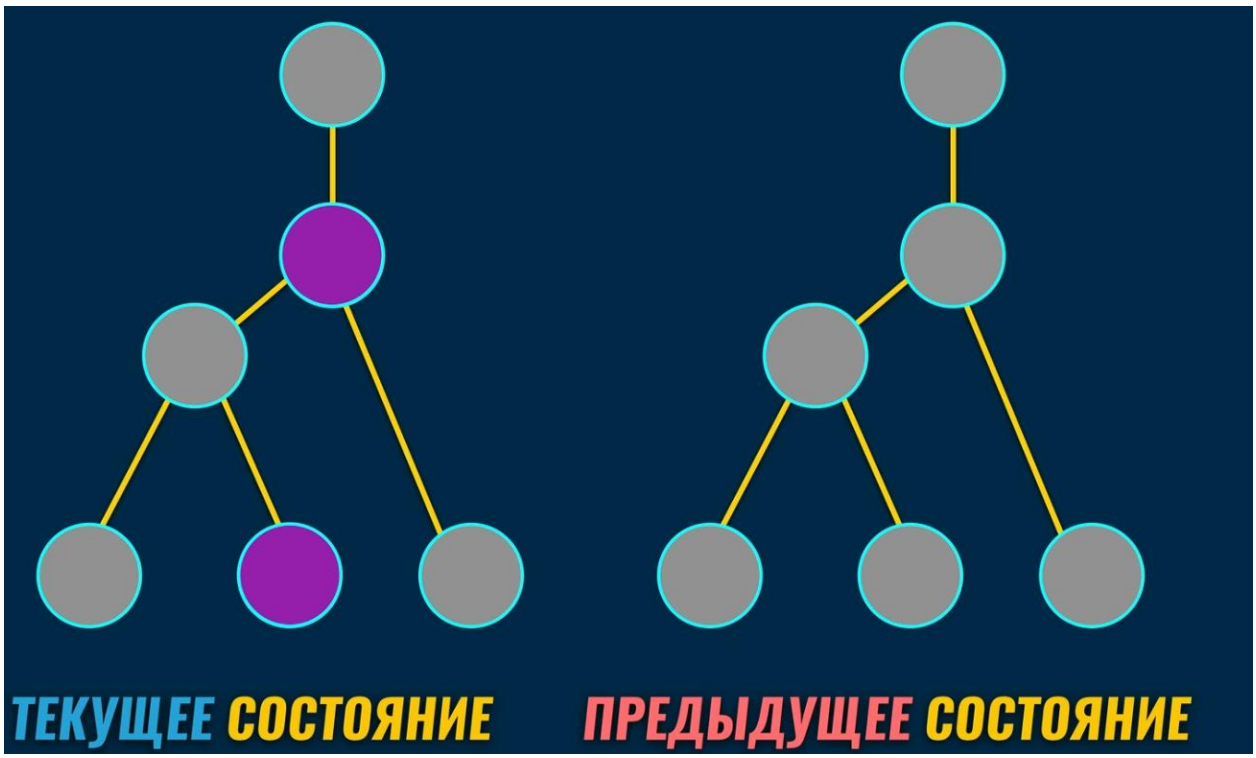
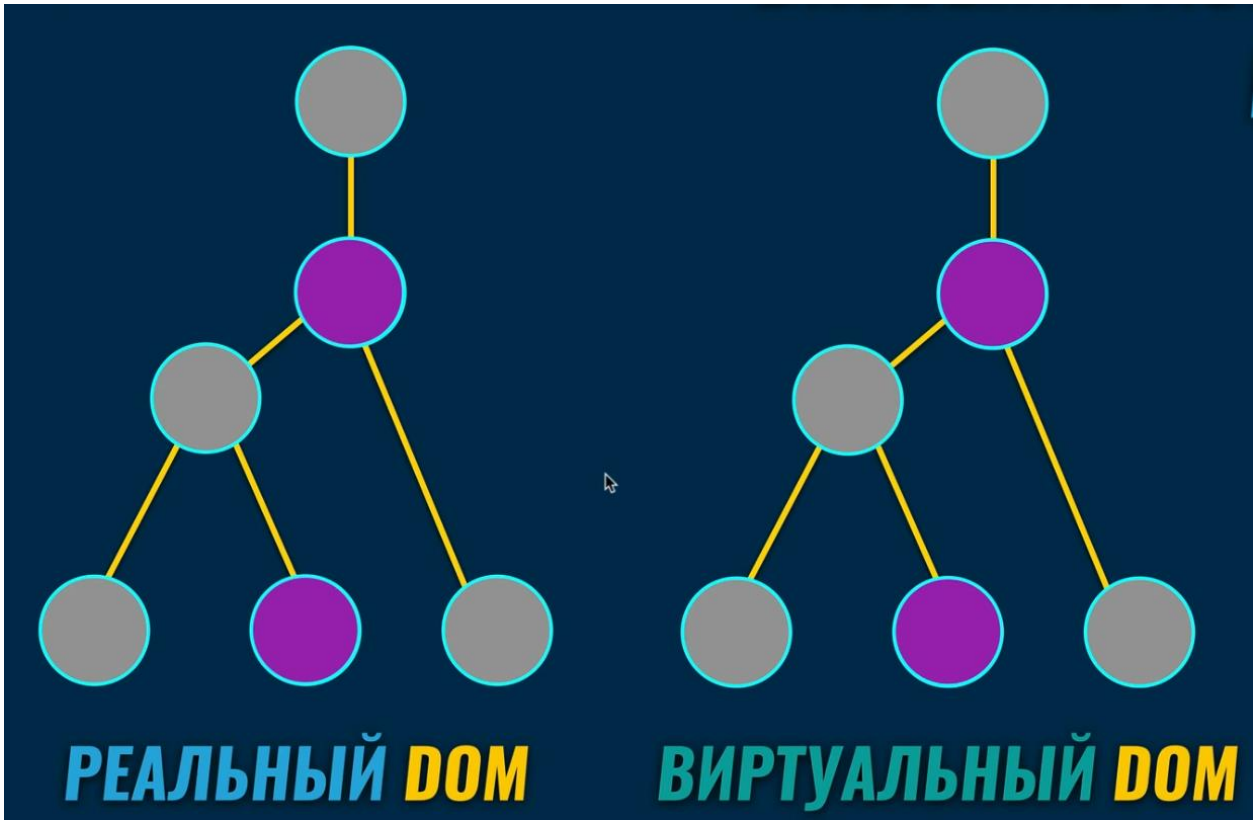
Основы React и взаимодействие с DOM






ТЕКУЩЕЕ СОСТОЯНИЕ
ПРЕДЫДУЩЕЕ СОСТОЯНИЕ


РЕАЛЬНЫЙ DOM
ВИРТУАЛЬНЫЙ DOM

КЛЮЧЕВЫЕ ПОНЯТИЯ В REACT

КОМПОНЕНТЫ
components

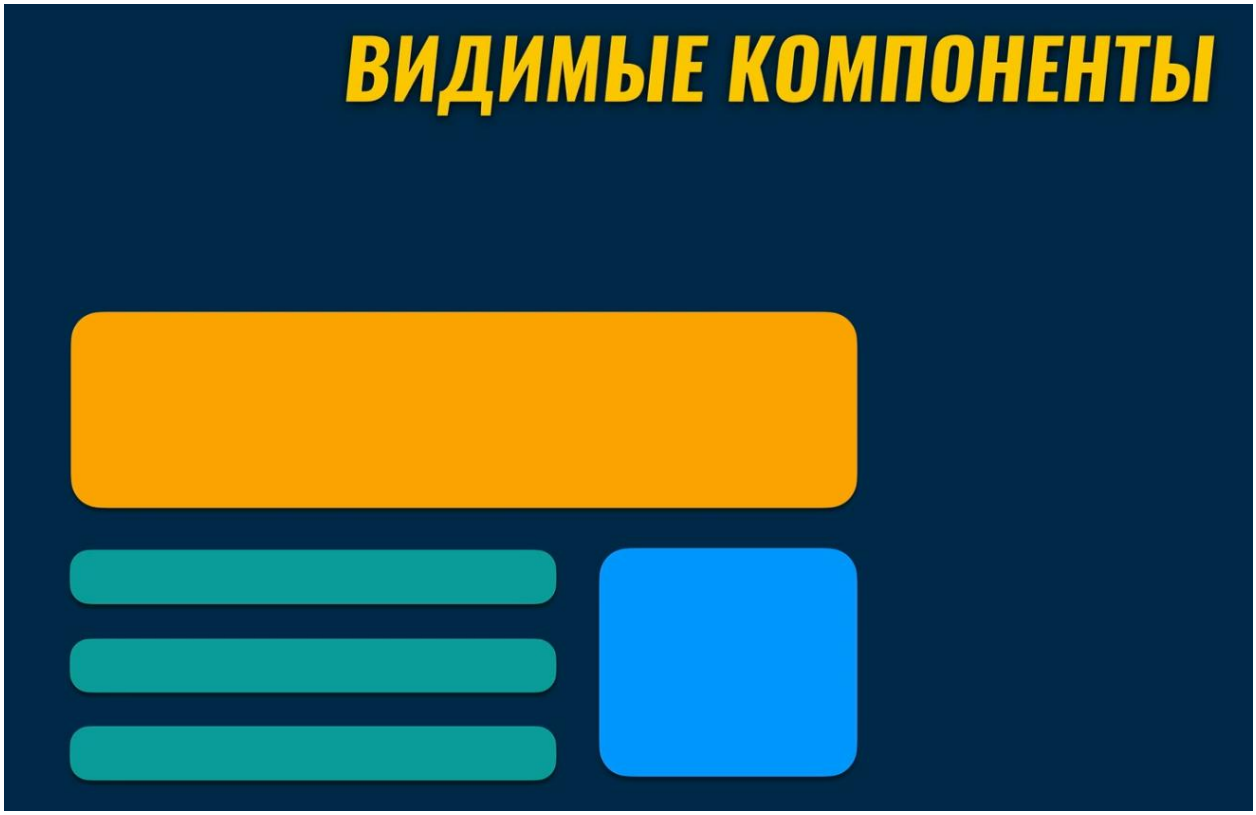
JSX
JavaScript Syntax Extension

СВОЙСТВА
props

СОСТОЯНИЕ
state

ИЕРАРХИЯ КОМПОНЕНТОВ


Корневой
компонент
Компонент,
объединяющий
схожие
компоненты


ВИДИМЫЕ КОМПОНЕНТЫ


ПЕРЕИСПОЛЬЗОВАНИЕ
КОМПОНЕНТОВ


АНАТОМИЯ КОМПОНЕНТА REACT
КОМПОНЕНТ REACT
HTML
CSS
JavaScript

ФУНКЦИОНАЛЬНЫЕ КОМПОНЕНТЫ

```
function HelloWorld() {
  return <h1>Hello World</h1>
}
```

ФУНКЦИОНАЛЬНЫЕ КОМПОНЕНТЫ

```
const HelloWorld = () => {
  return <h1>Hello World</h1>
}
```

КЛАССОВЫЕ КОМПОНЕНТЫ

```
class HelloWorld extends Component {
  render() {
    return <h1>Hello World</h1>
  }
}
```

JAVASCRIPT SYNTAX EXTENSION (JSX)

```
<Card
  style={{
    backgroundColor: `rgb(${initialColor}, ${opacity})`,
  }}
  className="m-2"
>
  <Card.Img
    variant="top"
    style={
      imageLoaded
        ? { opacity: 1, transition: 'opacity 2s ease-in-out' }
        : { opacity: 0 }
    }
    src={image}
    alt={title}
    onLoad={() => setImageLoaded(true)}
  />
</Card>
```

Похоже на CSS

Похоже на HTML

JavaScript

JSX - СИНТАКСИЧЕСКАЯ НАДСТРОЙКА НАД JS

```
React.createElement(Card, {
  style: {
    backgroundColor: `rgb(${initialColor}, ${opacity})`
  },
  className: "m-2"
}, React.createElement(Card.Img, {
  variant: "top",
  style: imageLoaded ? {
    opacity: 1,
    transition: 'opacity 2s ease-in-out'
  } : {
    opacity: 0
  },
  src: image,
  alt: title,
  onLoad: () => setImageLoaded(true)
}));
```

ПОЛЬЗОВАТЕЛЬСКИЕ И ВСТРОЕННЫЕ КОМПОНЕНТЫ В JSX

Встроенный компонент h1

```
return (
  <div>
    <h1>JSX в React</h1>
    <p>JSX похож на HTML</p>
    <Footer text={text} />
  </div>
)
```

Пользовательский компонент Footer

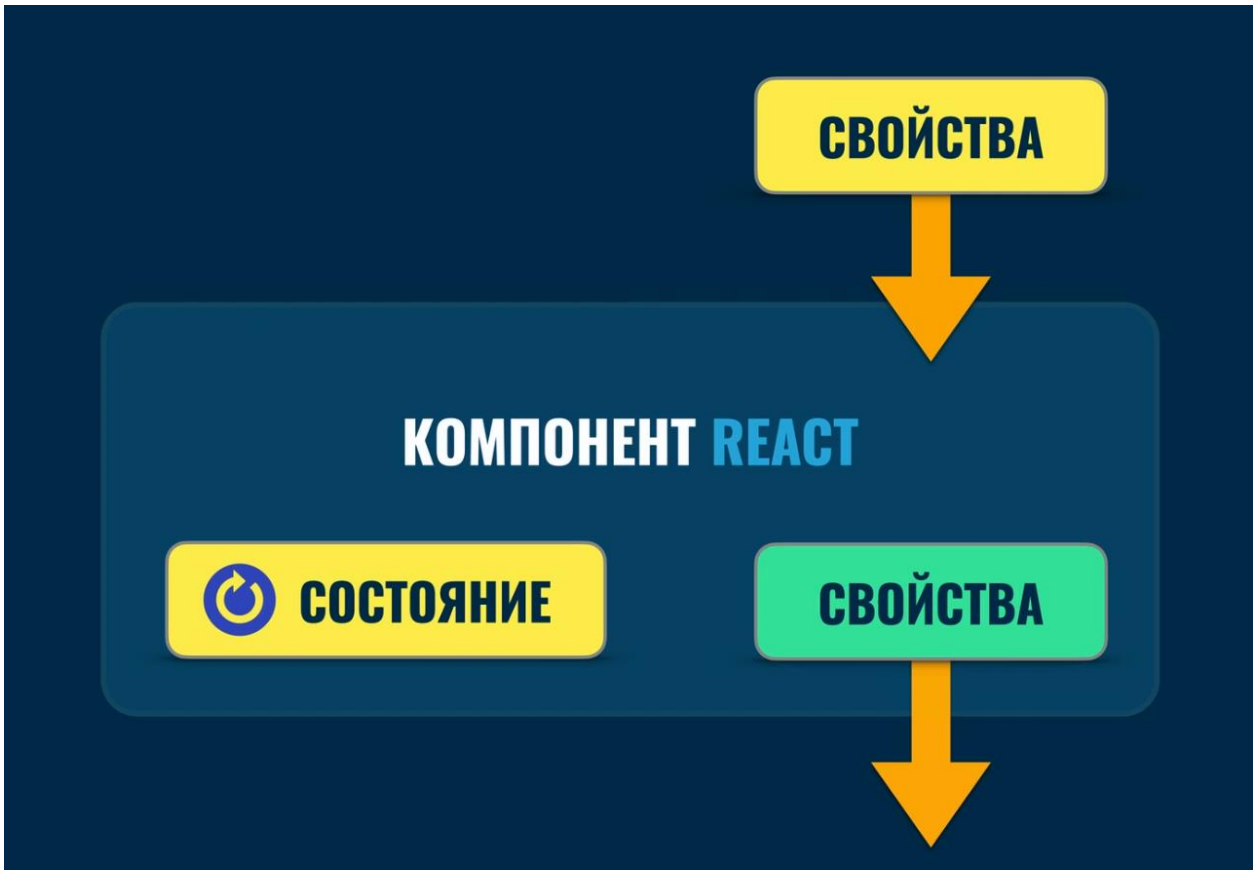
JSX ДОЛЖЕН ИМЕТЬ ОДИН КОРНЕВОЙ ЭЛЕМЕНТ

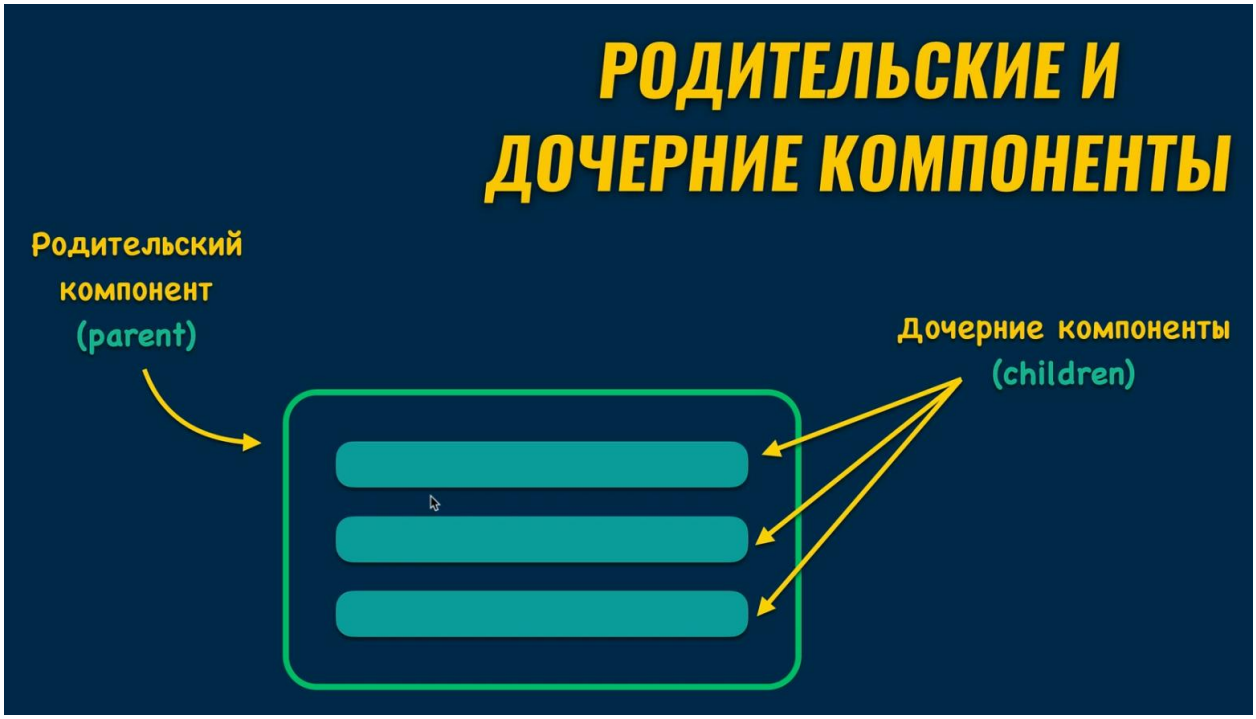
```
return (
  <div>
    <h1>JSX в React</h1>
    <p>JSX похож на HTML</p>
    <Footer text={text} />
  </div>
)
```

Валидный JSX

```
return (
  <h1>JSX в React</h1>
  <p>JSX похож на HTML</p>
)
```

Невалидный JSX


СВОЙСТВА
КОМПОНЕНТ REACT
СОСТОЯНИЕ
СВОЙСТВА

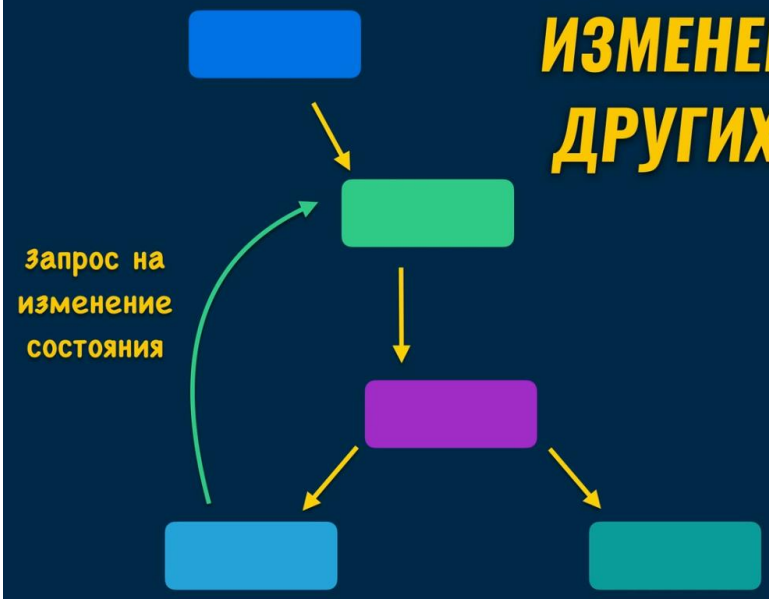

РОДИТЕЛЬСКИЕ И ДОЧЕРНИЕ КОМПОНЕНТЫ
Родительский компонент (parent)
Дочерние компоненты (children)

КОМПОНЕНТ НЕ ДОЛЖЕН ИЗМЕНЯТЬ СОБСТВЕННЫЕ СВОЙСТВА

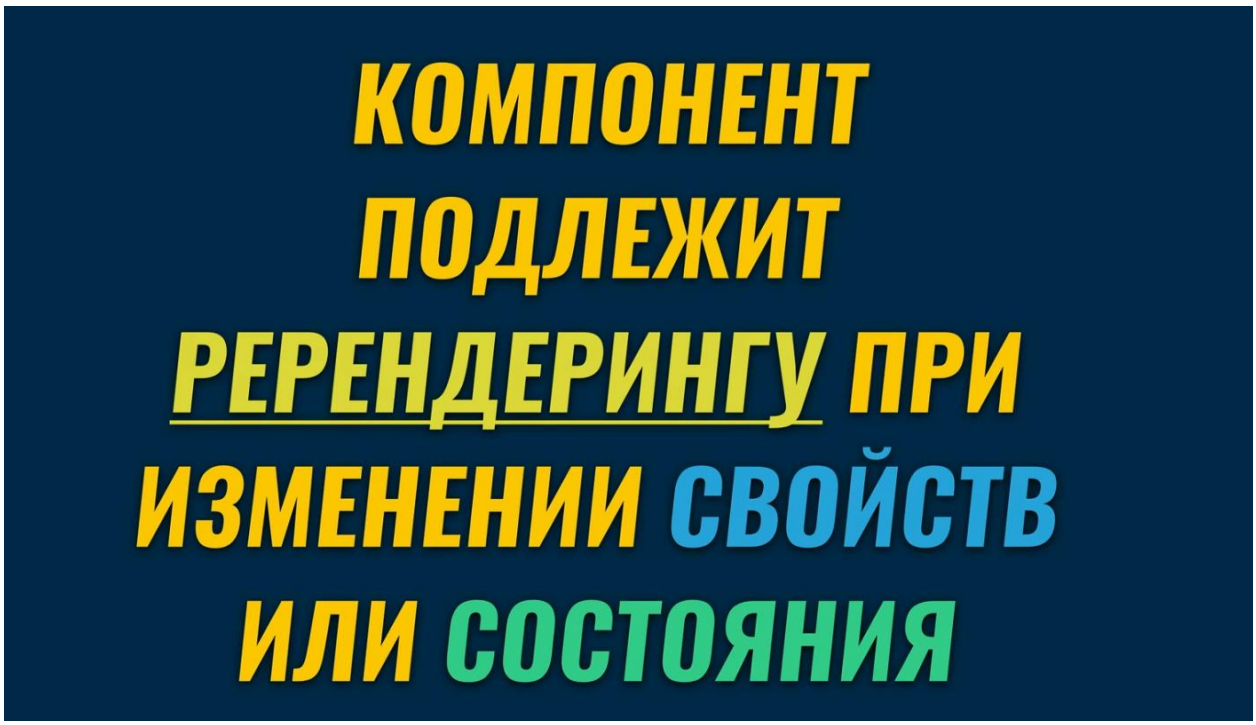
КОМПОНЕНТ МОЖЕТ ИЗМЕНЯТЬ СОБСТВЕННОЕ СОСТОЯНИЕ

КОМПОНЕНТ НЕ МОЖЕТ ИЗМЕНЯТЬ СОСТОЯНИЕ ДРУГИХ КОМПОНЕНТОВ

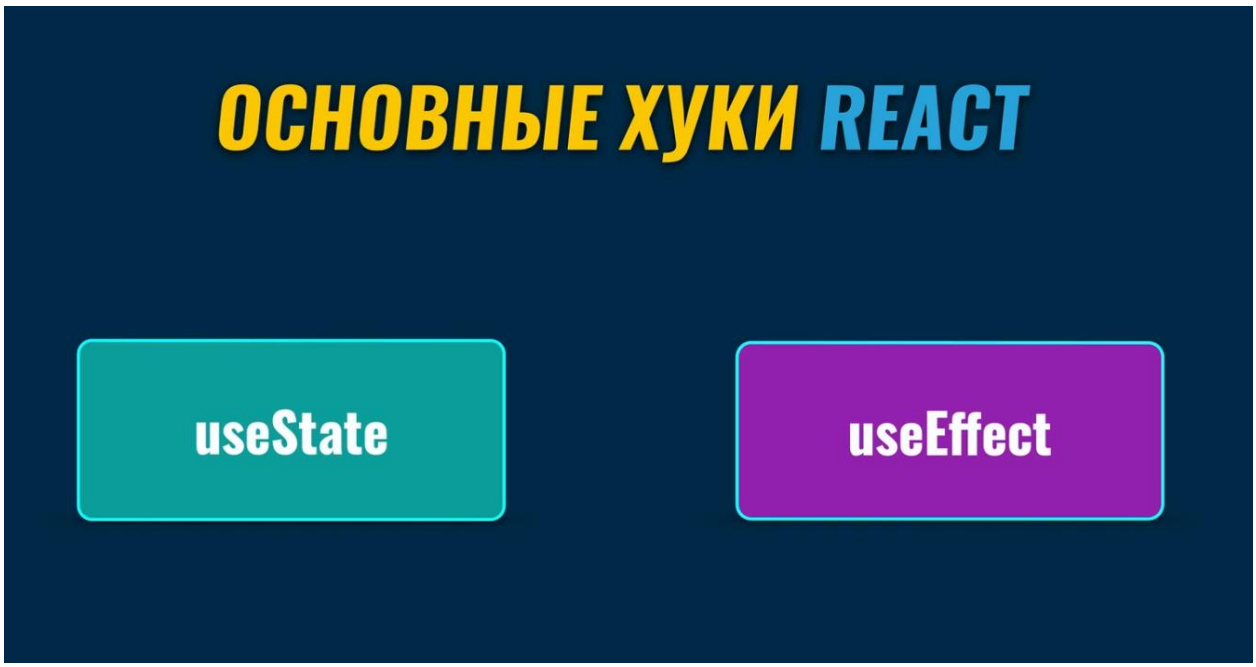
НО МОЖНО ВЛИЯТЬ НА ИЗМЕНЕНИЕ СОСТОЯНИЯ ДРУГИХ КОМПОНЕНТОВ



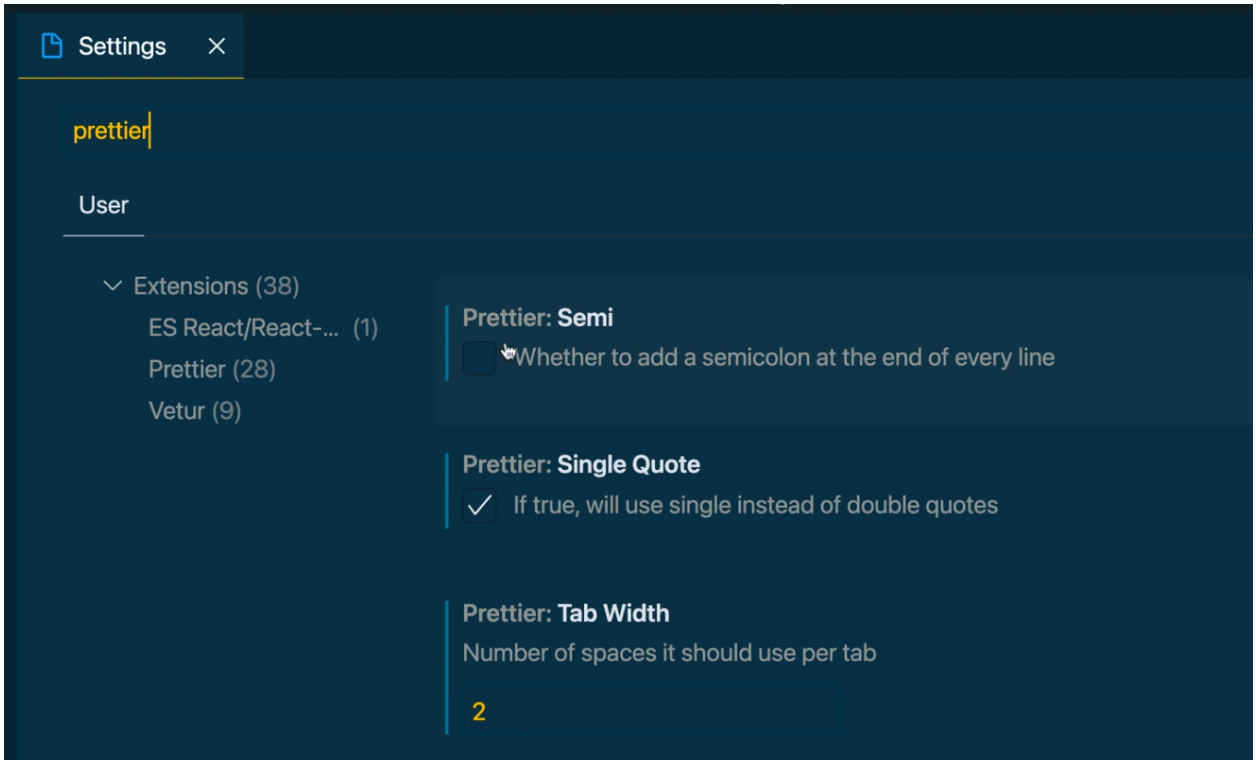
МОЖНО ПЕРЕДАВАТЬ ЧАСТЬ СВОИХ СВОЙСТВ И СОСТОЯНИЯ ДОЧЕРНИМ КОМПОНЕНТАМ В ВИДЕ СВОЙСТВ



Сам React этого не делает. Это выполняет другая библиотека: ReactDom – для Веб-приложений и React Native (RN) – для мобильных приложений


ОСНОВНЫЕ ХУКИ REACT
useState
useEffect

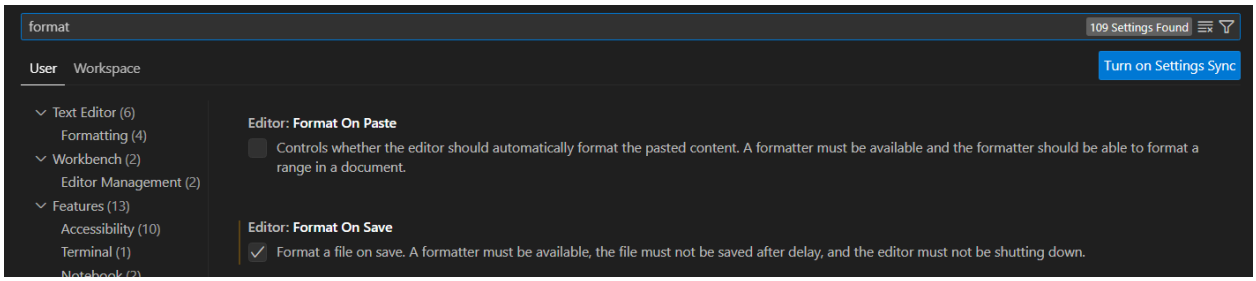
Настройка Prettier



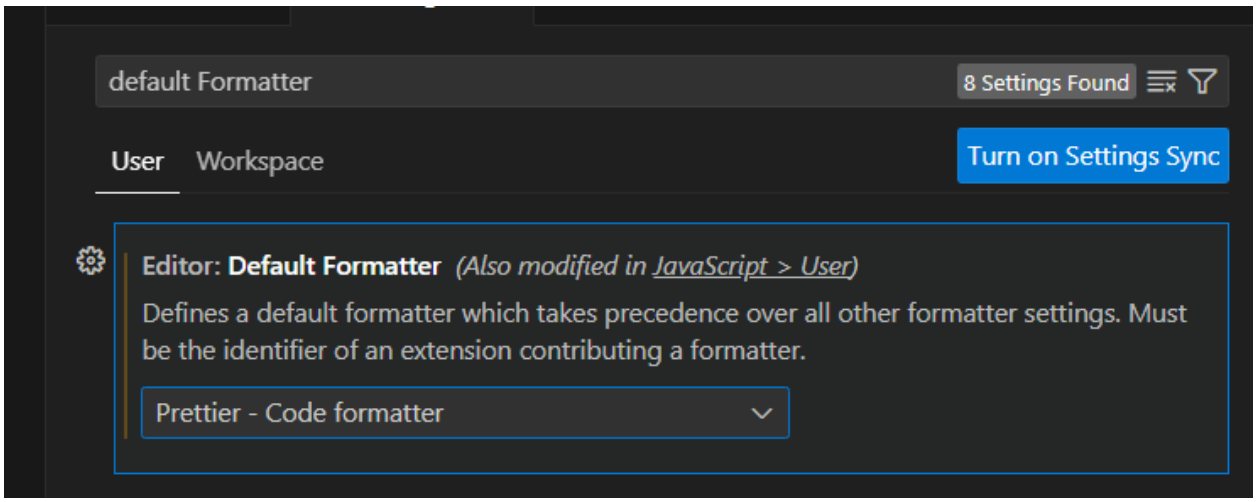
Semi – автоматической выставление «;» в конце строки кода

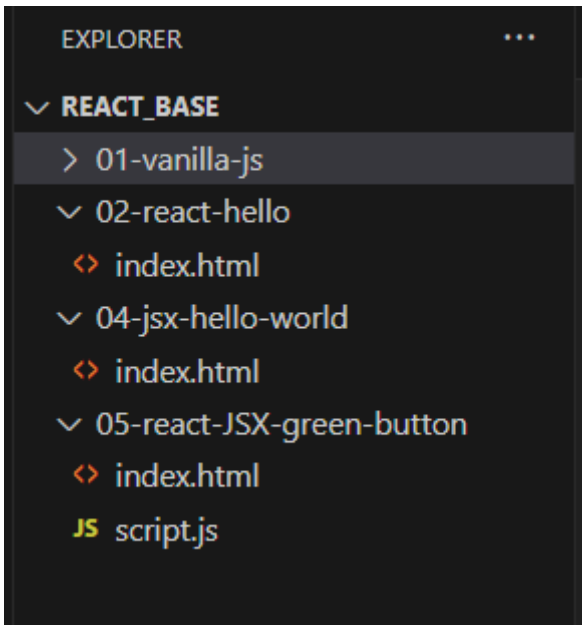
Single Quote – использование одинарных кавычек

Tab Width - табуляция



Format on Save – применить автоматическое форматирование кода при сохранении


default Formatter
8 Settings Found
User
Workspace
Turn on Settings Sync
Editor: Default Formatter (Also modified in JavaScript > User)
Defines a default formatter which takes precedence over all other formatter settings. Must be the identifier of an extension contributing a formatter.
Prettier - Code formatter

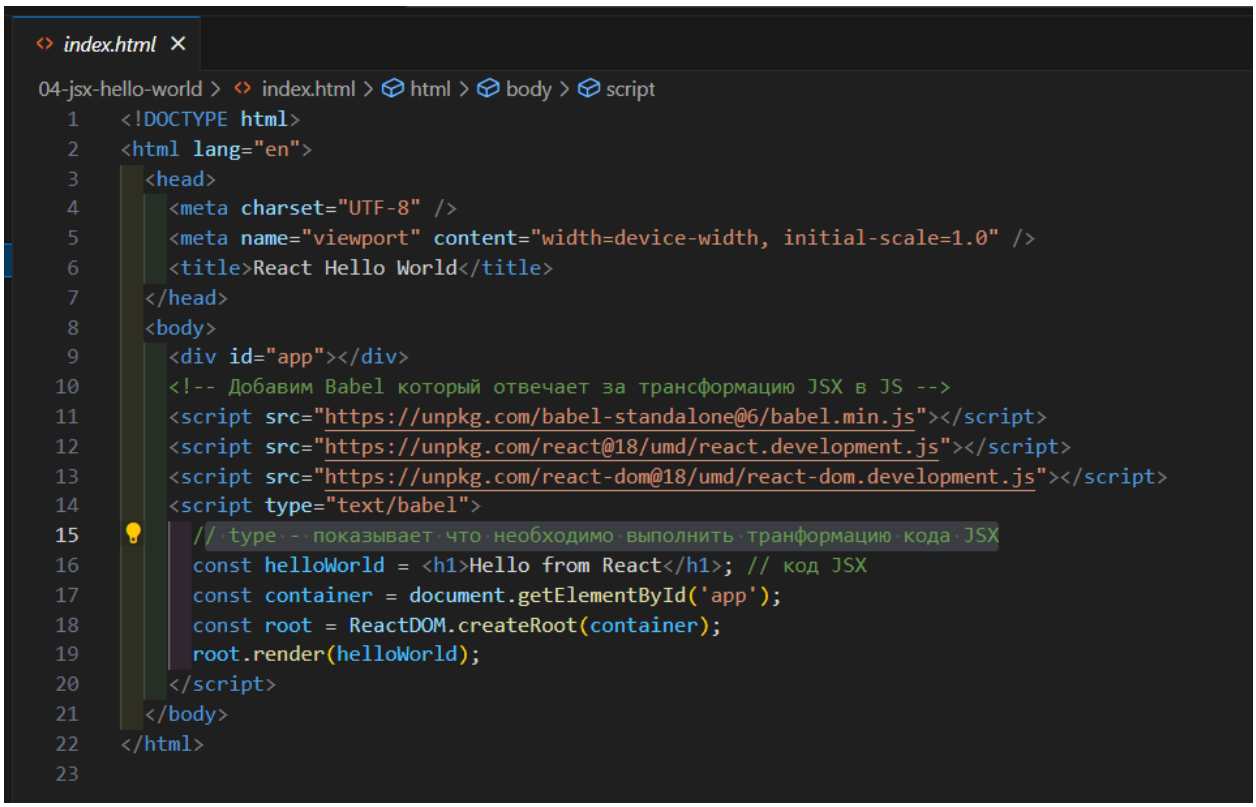


index.html

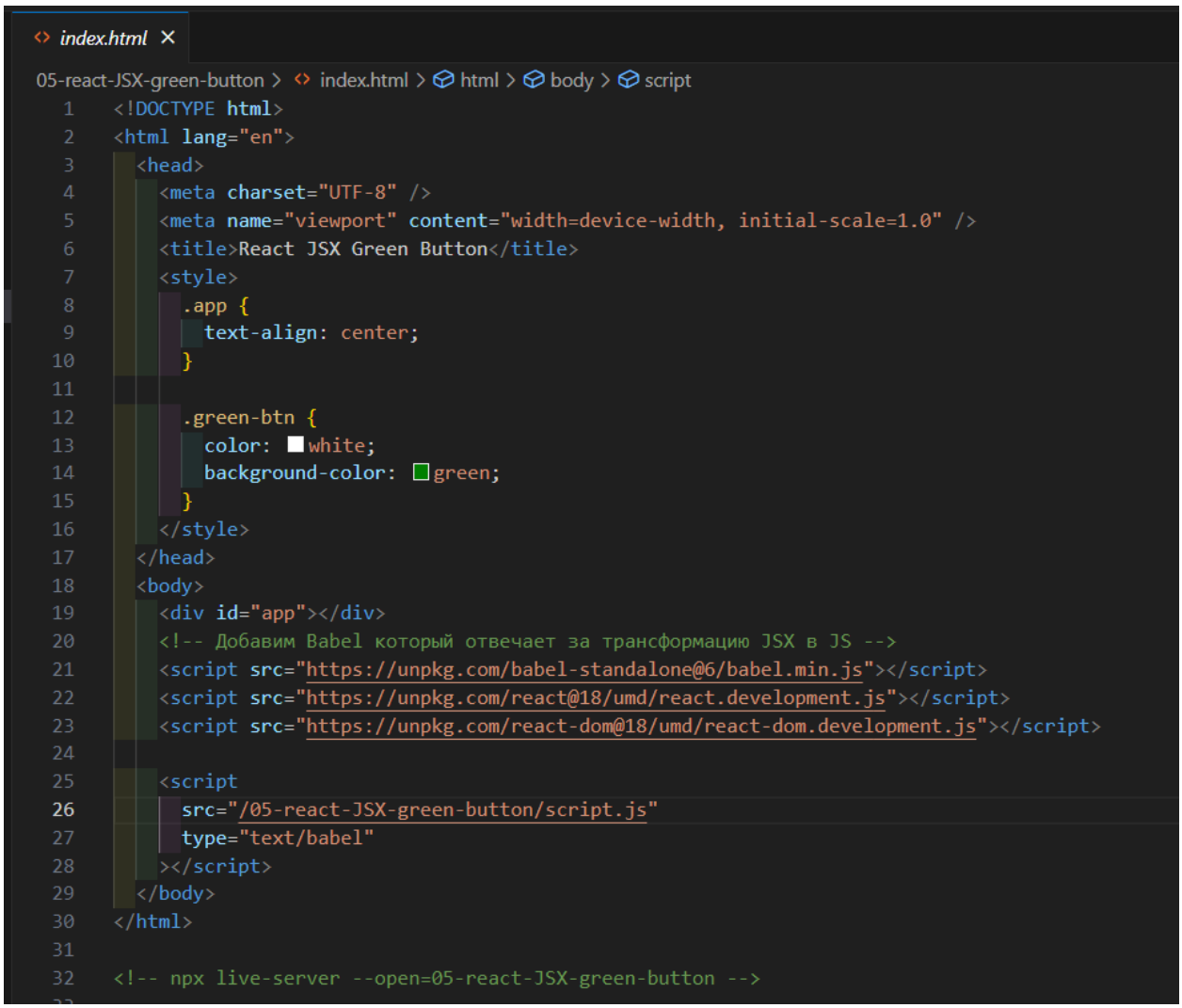
02-react-hello > index.html > ...

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>React Hello World</title>
  </head>
  <body>
    <!-- <div id="app"></div> -->
    <script src="https://unpkg.com/react@18/umd/react.development.js"></script>
    <script src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>
    <script>
      const helloWorld = React.createElement('h1', null, 'Hello from React');
      console.log(helloWorld);
      // const container = document.getElementById('app');
      // const root = ReactDOM.createRoot(container);
      // root.render(helloWorld);
    </script>
  </body>
</html>

```


index.html
04-jsx-hello-world > index.html > html > body > script
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>React Hello World</title>
</head>
<body>

<!-- Добавим Babel который отвечает за трансформацию JSX в JS -->
<script src="https://unpkg.com/babel-standalone@6/babel.min.js"></script>
<script src="https://unpkg.com/react@18/umd/react.development.js"></script>
<script src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>
<script type="text/babel">
// type - показывает что необходимо выполнить транформацию кода JSX
const helloWorld = <h1>Hello from React</h1>; // код JSX
const container = document.getElementById('app');
const root = ReactDOM.createRoot(container);
root.render(helloWorld);
</script>
</body>
</html>



index.html

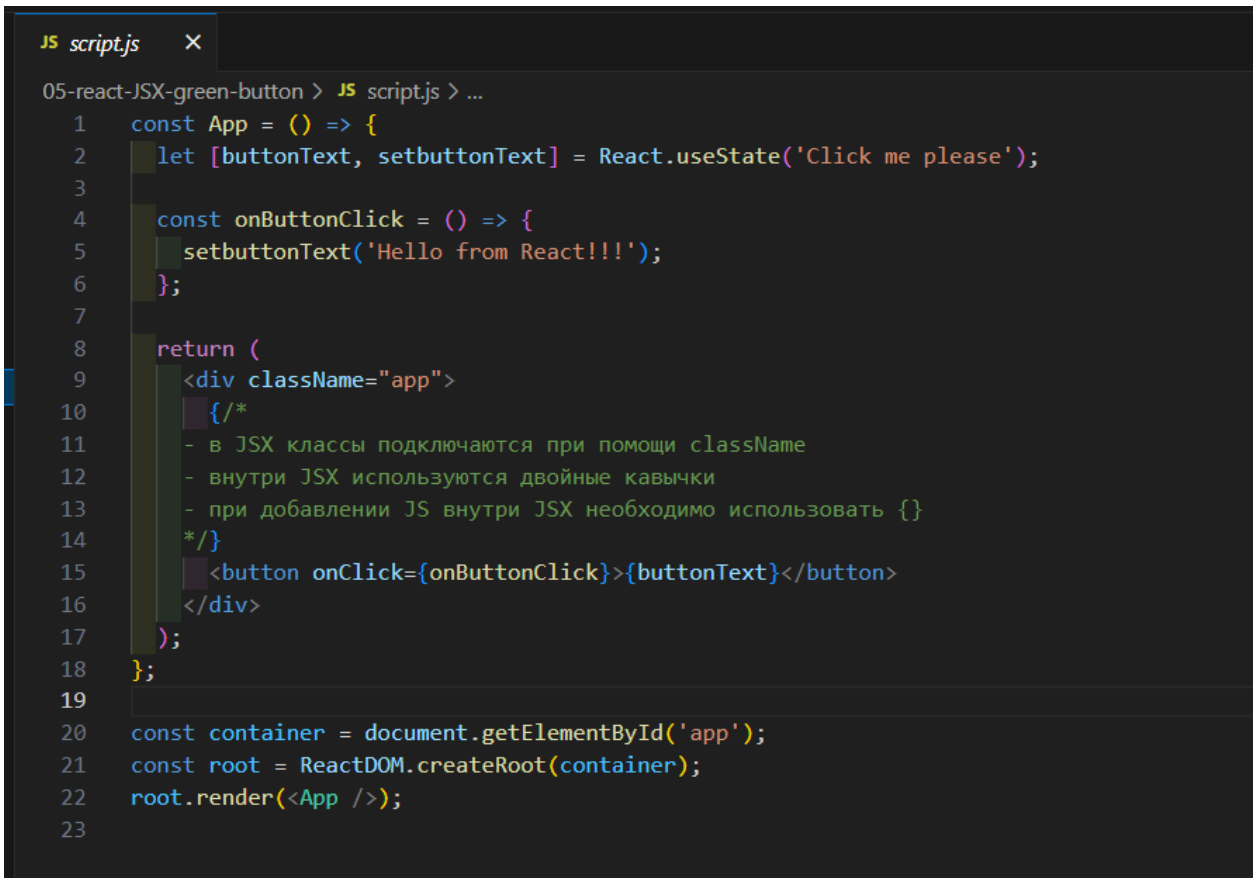
05-react-JSX-green-button > index.html > html > body > script

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>React JSX Green Button</title>
    <style>
      .app {
        text-align: center;
      }

      .green-btn {
        color: white;
        background-color: green;
      }
    </style>
  </head>
  <body>
    <div id="app"></div>
    <!-- Добавим Babel который отвечает за трансформацию JSX в JS -->
    <script src="https://unpkg.com/babel-standalone@6/babel.min.js"></script>
    <script src="https://unpkg.com/react@18/umd/react.development.js"></script>
    <script src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>

    <script
      src="/05-react-JSX-green-button/script.js"
      type="text/babel"
    ></script>
  </body>
</html>

<!-- npx live-server --open=05-react-JSX-green-button -->
```


script.js
05-react-JSX-green-button > script.js > ...
1 const App = () => {
2 let [buttonText, setbuttonText] = React.useState('Click me please');
3
4 const onButtonClick = () => {
5 setbuttonText('Hello from React!!!');
6 };
7
8 return (
9
10 {/*
11 - в JSX классы подключаются при помощи className
12 - внутри JSX используются двойные кавычки
13 - при добавлении JS внутри JSX необходимо использовать {}
14 */}
15 <button onClick={onButtonClick}>{buttonText}</button>
16
17);
18 };
19
20 const container = document.getElementById('app');
21 const root = ReactDOM.createRoot(container);
22 root.render(<App />);
23