

Классовые компоненты

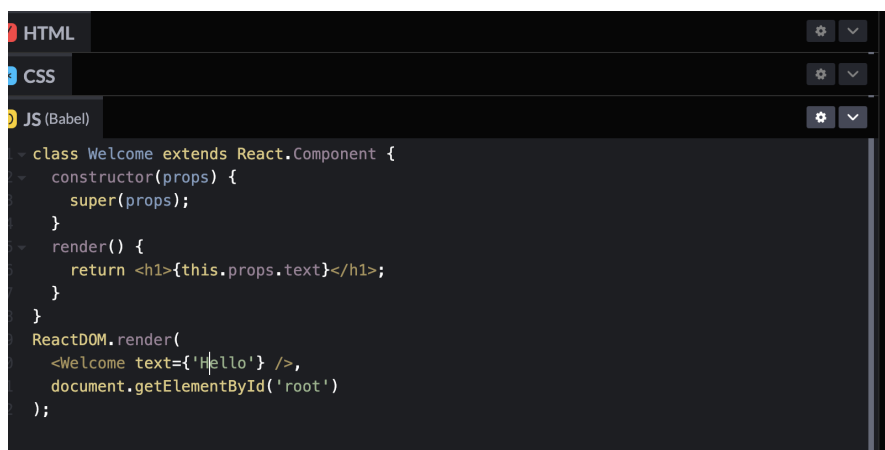
Чтобы определить классový компонент, необходимо отнаследоваться от `React.Component`. Единственный обязательный метод в подклассе `React.Component` — `render()`.

```
class Welcome extends React.Component {  
  render() {  
    return <h1>Hello</h1>;  
  }  
}
```

constructor()

Вы можете не использовать конструктор в React-компоненте, если не определяете состояние или не привязываете методы.

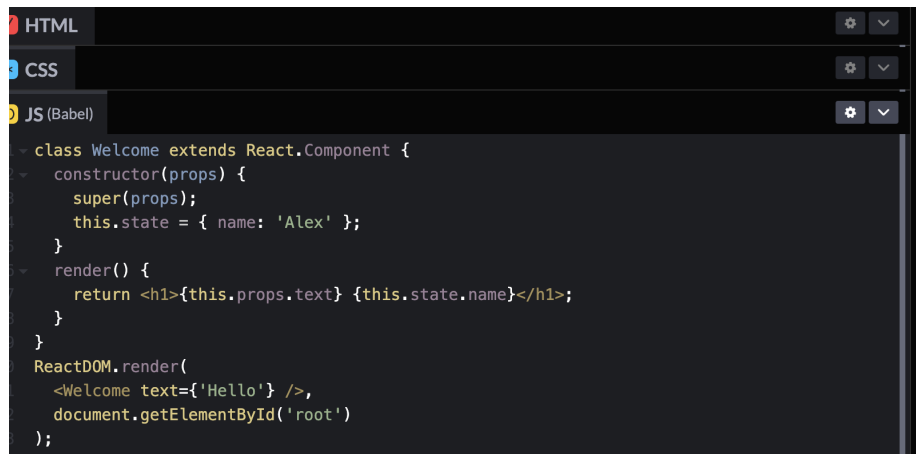
Конструктор компонента React вызывается до того, как компонент будет примонтирован. В начале конструктора необходимо вызывать `super(props)`. Если это не сделать, `this.props` не будет определён. Это может привести к багам.



```
HTML  
CSS  
JS (Babel)  
class Welcome extends React.Component {  
  constructor(props) {  
    super(props);  
  }  
  render() {  
    return <h1>{this.props.text}</h1>;  
  }  
}  
ReactDOM.render(  
  <Welcome text={'Hello'} />,  
  document.getElementById('root')  
);
```

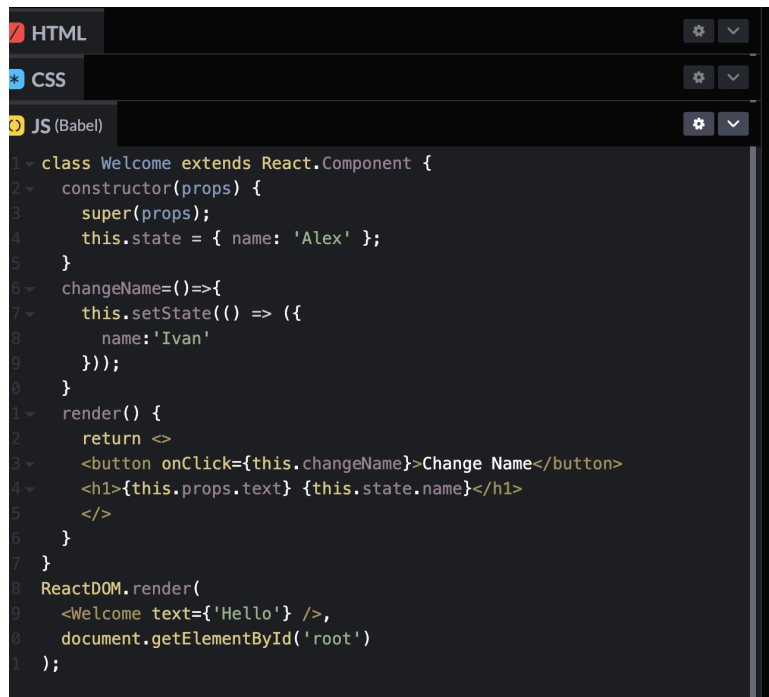
Hello

Конструктор — единственное место, где можно напрямую изменять `this.state`. В остальных методах необходимо использовать `this.setState()`.



```
HTML
CSS
JS (Babel)
class Welcome extends React.Component {
  constructor(props) {
    super(props);
    this.state = { name: 'Alex' };
  }
  render() {
    return <h1>{this.props.text} {this.state.name}</h1>;
  }
}
ReactDOM.render(
  <Welcome text={'Hello'} />,
  document.getElementById('root')
);
```

Hello Alex



```
HTML
CSS
JS (Babel)
class Welcome extends React.Component {
  constructor(props) {
    super(props);
    this.state = { name: 'Alex' };
  }
  changeName={() => {
    this.setState(() => ({
      name: 'Ivan'
    }));
  }}
  render() {
    return <
      <button onClick={this.changeName}>Change Name</button>
      <h1>{this.props.text} {this.state.name}</h1>
    </>
  }
}
ReactDOM.render(
  <Welcome text={'Hello'} />,
  document.getElementById('root')
);
```

Change Name
Hello Ivan

Жизненный цикл компонента

Каждый компонент имеет несколько методов жизненного цикла.

Переопределение такого метода позволяет выполнять код на конкретном этапе процесса жизненного цикла. Далее на странице **полужирным шрифтом** выделены самые распространённые методы жизненного цикла.

Монтирование

При создании экземпляра компонента и его вставке в DOM, следующие методы вызываются в установленном порядке:

- **constructor()**
- **render()**
- **componentDidMount()**

Добавление метода жизненного цикла в класс

Метод `componentDidMount()` запускается после того, как компонент отрендерился.

