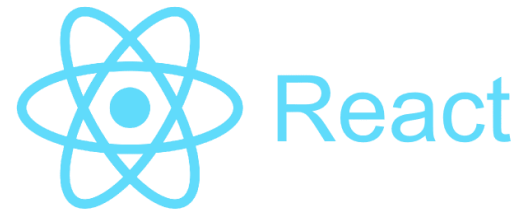




Microsoft Partner
Silver Learning



React Essential

Состояние и жизненный цикл



ITVVDN
IT VIDEO DEVELOPERS NETWORK

React Essential

Introduction

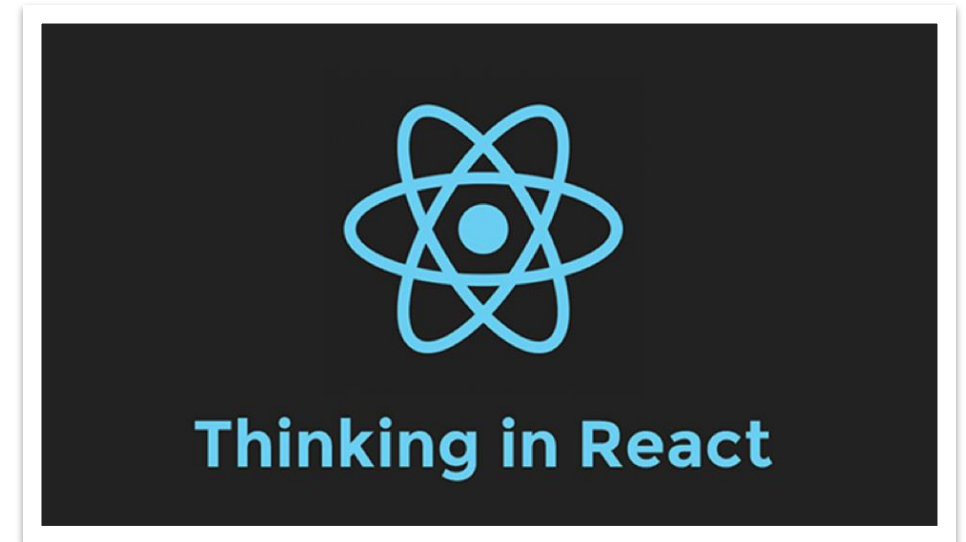


Муляк Дмитрий

Front-end developer at GlobalTechMakers

 dmitriymuliak

 dmitriymuliak



Состояние и жизненный цикл

Состояние и жизненный цикл

1. Функциональный компонент
2. Классовый компонент
3. Преобразование функционального компонента в классový
4. Добавление методов жизненного цикла
5. Знакомство с Redux

React Essential

Функциональный компонент

Проще всего объявить React-компонент как функцию:

```
function Welcome(props) {  
  return <h1>Привет, {props.name}</h1>;  
}
```

Эта функция — компонент, потому что она получает данные в одном объекте («пропсы») в качестве параметра и возвращает React-элемент. Мы будем называть такие компоненты «функциональными», так как они буквально являются функциями.

React Essential

Классовый компонент

Ещё компоненты можно определять как [классы ES6](#):

```
class Welcome extends React.Component {  
  render() {  
    return <h1>Привет, {this.props.name}</h1>;  
  }  
}
```

Функциональным и классовым компонентам доступны дополнительные возможности, о которых мы поговорим в [следующих главах](#).

React Essential

Как отрендерить компонент

Пока что мы только встречали React-элементы, представляющие собой DOM-теги:

```
const element = <div />;
```

Но элементы могут описывать и наши собственные компоненты:

```
const element = <Welcome name="Алиса" />;
```

Когда React встречает подобный элемент, он собирает все JSX-атрибуты и дочерние элементы в один объект и передаёт их нашему компоненту. Такие объекты называются «пропсы» (props). Например, этот компонент выведет «Привет, Алиса» на страницу:

```
function Welcome(props) {  
  return <h1>Привет, {props.name}</h1>;  
}  
  
const element = <Welcome name="Алиса" />;  
ReactDOM.render(  
  element,  
  document.getElementById('root')  
);
```

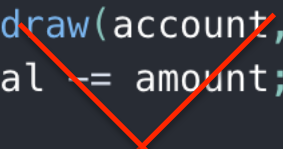
React Essential

Пропсы можно только читать

```
function sum(a, b) {  
  return a + b;  
}
```

Компонент никогда не должен что-то записывать в свои пропсы — вне зависимости от того, [функциональный он или классовый](#).

```
function withdraw(account, amount) {  
  account.total -= amount;  
}
```



Такие функции называют [«ЧИСТЫМИ»](#), потому что они не меняют свои входные данные и предсказуемо возвращают один и тот же результат для одинаковых аргументов.

React-компоненты обязаны вести себя как чистые функции по отношению к своим пропсам.

Преобразование функционального компонента в классовый

1. Создаём [ES6-класс](#) с таким же именем, указываем `React.Component` в качестве родительского класса
2. Добавим в класс пустой метод `render()`
3. Перенесём тело функции в метод `render()`
4. Заменяем `props` на `this.props` в теле `render()`
5. Удалим оставшееся пустое объявление функции

```
function Clock(props) {  
  return (  
    <div>  
      <h1>Привет, мир!</h1>  
      <h2>Сейчас {props.date.toLocaleTimeString()}</h2>  
    </div>  
  );  
}
```

```
class Clock extends React.Component {  
  render() {  
    return (  
      <div>  
        <h1>Привет, мир!</h1>  
        <h2>Сейчас {this.props.date.toLocaleTimeString()}</h2>  
      </div>  
    );  
  }  
}
```

React Essential

Добавим внутреннее состояние в класс

Переместим date из пропсов в состояние в три этапа:

1. Заменим `this.props.date` на `this.state.date` в методе `render()`.
2. Добавим [конструктор класса](#), в котором укажем начальное состояние в переменной `this.state`.
3. Удалим проп `date` из элемента `<Clock />`:

```
class Clock extends React.Component {  
  render() {  
    return (  
      <div>  
        <h1>Привет, мир!</h1>  
        <h2>Сейчас {this.props.date.toLocaleTimeString()}</h2>  
      </div>  
    );  
  }  
}
```

```
class Clock extends React.Component {  
  constructor(props) {  
    super(props);  
    this.state = {date: new Date()};  
  }  
  
  render() {  
    return (  
      <div>  
        <h1>Привет, мир!</h1>  
        <h2>Сейчас {this.state.date.toLocaleTimeString()}</h2>  
      </div>  
    );  
  }  
}
```

React Essential

Добавление методов жизненного цикла

Методы, которые помогают управлять нашими компонентами, называются «методами жизненного цикла» (lifecycle methods).

Первоначальный рендеринг компонента в DOM называется «монтирование» (mounting).

Каждый раз, когда DOM-узел, созданный компонентом, удаляется, происходит «размонтирование» (unmounting).

<https://ru.reactjs.org/docs/react-component>.

```
class Clock extends React.Component {
  constructor(props) {
    super(props);
    this.state = {date: new Date()};
  }

  componentDidMount() {
  }

  componentWillUnmount() {
  }

  render() {
    return (
      <div>
        <h1>Привет, мир!</h1>
        <h2>Сейчас {this.state.date.toLocaleTimeString()}.</h2>
      </div>
    );
  }
}
```

React Essential

Знакомство с Redux

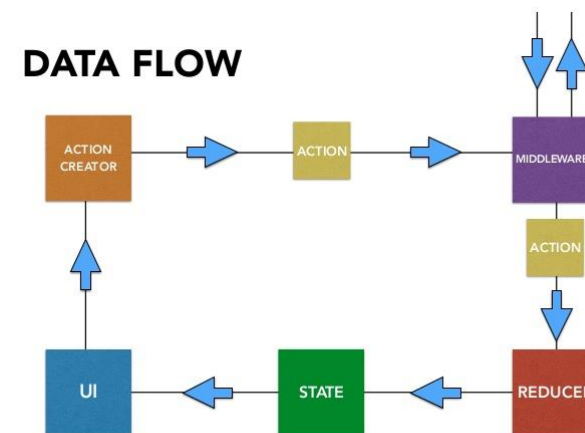
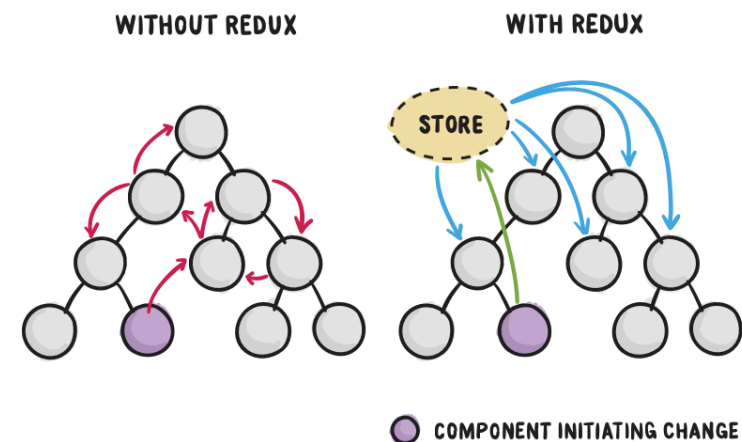
Redux - это шаблон для управления состоянием приложения. Библиотека управления состоянием для приложений, написанных на JavaScript.

Когда имеет смысл использовать Redux:

- У вас есть обоснованные объемы данных, меняющихся со временем.
- Вам нужен один источник информации для вашего состояния.
- Вы приходите к выводу, что сохранять все ваше состояние в компоненте верхнего уровня уже недостаточно.

<https://redux.js.org/>.

<https://rajdee.gitbooks.io/redux-in-russian/content/>



Проверка знаний

TestProvider.com



Проверьте как Вы усвоили данный материал на TestProvider.com

TestProvider – это online сервис проверки знаний по информационным технологиям. С его помощью Вы можете оценить Ваш уровень и выявить слабые места. Он будет полезен как в процессе изучения технологии, так и для общей оценки знаний IT специалиста.

Успешное прохождение финального тестирования позволит Вам получить соответствующий Сертификат.

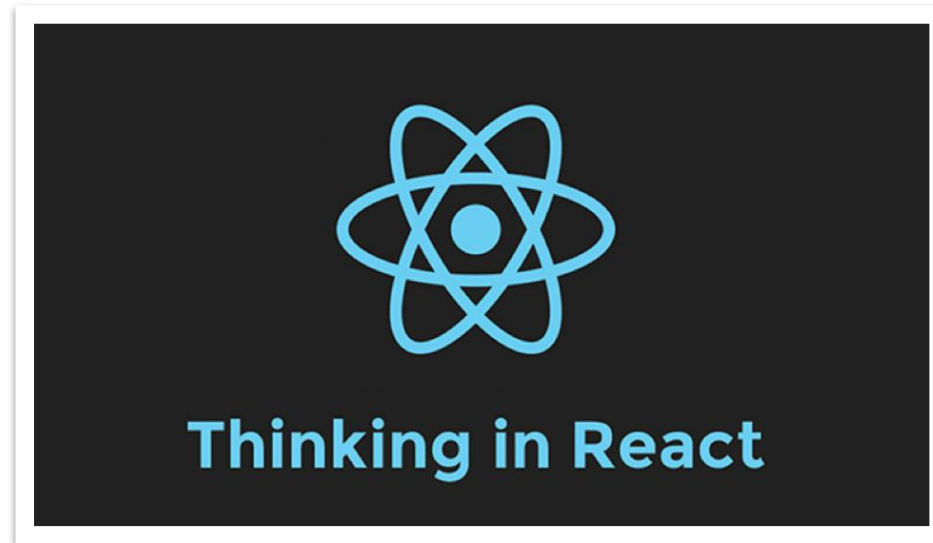
React Essential

Спасибо за внимание! До новых встреч!



Муляк Дмитрий

Front-end Developer at GTM



Информационный видеосервис для разработчиков программного обеспечения

