РАЗРАБОТКА ВЕБ-САЙТОВ*, REACTJS*, JAVASCRIPT*

React на ES6+

ПЕРЕВОД

olegshilov 9 июля 2015 в 16:30 62,3k

Оригинал: Steven Luscher

Это перевод поста Steven Luscher опубликованного в блоге Babel. Steven работает в Facebook над Relay – JavaScript фрэймворком для создания приложений с использованием React и GraphQL.

> За этот год, в процессе реорганизации Instagram Web, мы насладились использованием ряда особенностей ES6+, при написании наших React компонентов. Позвольте мне остановиться на тех моментах, когда новые возможности языка могут повлиять на то как вы пишите React приложения, и сделают этот процесс легче и веселее, чем когда-либо.

Классы

До сих пор наиболее заметным из видимых изменений в том, как мы пишем наши React компоненты, используя ES6+ является то, что мы решили использовать синтаксис определения класса. Вместо того чтобы использовать метод *React.createClass* для определения компонента, мы можем определить настоящий ES6 класс, который расширяет класс *React.Component*:

```
class Photo extends React.Component {
  render() {
    return <img alt={this.props.caption} src={this.props.src} />;
  }
}
```

Вы сразу же вы заметите небольшое различие — вам становится доступным более лаконичный синтаксис определения класса:

```
// The ES5 way
var Photo = React.createClass({
  handleDoubleTap: function(e) { … },
  render: function() { … },
});
```

```
// The ES6+ way
class Photo extends React.Component {
  handleDoubleTap(e) { … }
  render() { … }
}
```

Стоит отметить, что мы отбросили две скобки и завершающую точку с запятой, а для каждого объявленного метода мы опускаем двоеточие, ключевое слово *function* и запятую.

Все методы жизненного цикла компонента, кроме одного могут быть определены, как можно было бы ожидать, с использованием нового синтаксиса определения классов. Конструктор класса в настоящее время выступает в роли ранее используемого метода *componentWillMount*:

```
// The ES5 way
var EmbedModal = React.createClass({
```

```
  componentWillMount: function() { … },
});
```

```
// The ES6+ way
class EmbedModal extends React.Component {
  constructor(props) {
    super(props);
    // Operations usually carried out in componentWillMount go here
  }
}
```

Инициализаторы свойств

В мире классов ES6+, типы свойств и значения по умолчанию могут существовать как статические свойства этого класса. Эти переменные, а также начальное состояние компонента, могут быть определены с использованием ES7 инициализаторов свойств:

```
// The ES5 way
var Video = React.createClass({
  getDefaultProps: function() {
    return {
      autoPlay: false,
      maxLoops: 10,
    };
  },
  getInitialState: function() {
    return {
      loopsRemaining: this.props.maxLoops,
    };
  },
  propTypes: {
    autoPlay: React.PropTypes.bool.isRequired,
    maxLoops: React.PropTypes.number.isRequired,
    posterFrameSrc: React.PropTypes.string.isRequired,
    videoSrc: React.PropTypes.string.isRequired,
```

```
  },
});
```

```
// The ES6+ way
class Video extends React.Component {
  static defaultProps = {
    autoPlay: false,
    maxLoops: 10,
  }
  static propTypes = {
    autoPlay: React.PropTypes.bool.isRequired,
    maxLoops: React.PropTypes.number.isRequired,
    posterFrameSrc: React.PropTypes.string.isRequired,
    videoSrc: React.PropTypes.string.isRequired,
  }
  state = {
    loopsRemaining: this.props.maxLoops,
  }
}
```

Инициализаторы свойств ES7 работают внутри конструктора класса, где *this* относится к текущему экземпляру класса перед его созданием. Благодаря этому, начальное состояние компонента может зависить от *this.props*. Примечательно, что мы больше не должны определять значения *props* по умолчанию и начальное состояние объекта в терминах *getter* функции.

Arrow функции

Метод *React.createClass* используется для выполнения некоторых дополнительных работ по привязке к методам экземпляра компонента, чтобы убедиться, что внутри них, ключевое слово *this*

будет относиться к экземпляру компонента.

```
// Autobinding, brought to you by React.createClass
var PostInfo = React.createClass({
  handleOptionsButtonClick: function(e) {
    // Here, 'this' refers to the component instance.
    this.setState({showOptionsModal: true});
  },
});
```

Так как мы не связаны использованием метода *React.createClass*, при определении компонентов синтаксисом классов ES6+, казалось бы, что нам нужно вручную привязать методы экземпляра, туда где мы хотим их использовать:

```
// Manually bind, wherever you need to
class PostInfo extends React.Component {
  constructor(props) {
    super(props);
    // Manually bind this method to the component instance...
    this.handleOptionsButtonClick =
this.handleOptionsButtonClick.bind(this);
  }
  handleOptionsButtonClick(e) {
    // ...to ensure that 'this' refers to the component instance
here.
    this.setState({showOptionsModal: true});
  }
}
```

К счастью, путем объединения двух возможностей ES6+ – arrow функции и инициализаторы свойств – отказ от привязки к экземпляру компонента становится очень легким:

```
class PostInfo extends React.Component {
  handleOptionsButtonClick = (e) => {
```

```
    this.setState({showOptionsModal: true});
  }
}
```

Тело ES6 arrow функций использует то же лексическое *this* как и код, который её окружает. Это позволяет нам достичь желаемого результата из-за того, что ES7 инициализаторы свойств находятся в области видимости. Загляните под капот, чтобы понять почему это работает.

Динамические имена свойств и шаблоные строки

Одним из усовершенствований литералов объектов включает в себя возможность назначать производные имена свойствам. Изначально мы могли бы сделать что-то подобное для установки некоторой части состояния компонента:

```
var Form = React.createClass({
  onChange: function(inputName, e) {
    var stateToSet = {};
    stateToSet[inputName + 'Value'] = e.target.value;
    this.setState(stateToSet);
  },
});
```

Теперь у нас есть возможность создавать объекты, в которых имена свойств определяются выражением JavaScript во время выполнения. Здесь мы используем шаблонные строки, для того что-бы определить, какое свойство установить в состоянии

компонента:

```
class Form extends React.Component {
  onChange(inputName, e) {
    this.setState({
      [`${inputName}Value`]: e.target.value,
    });
  }
}
```

Деструктуризация и распространение атрибутов

Часто при создании компонентов, мы могли бы передать большинство свойств родительского компонента к дочернему компоненту, но не все из них. В сочетании ES6+ деструктурирования и распространения атрибутов JSX, это становится возможным без плясок с бубном:

```
class AutoloadingPostsGrid extends React.Component {
  render() {
    var {
      className,
      ...others,  // contains all properties of this.props except
for className
    } = this.props;
    return (
      <div className={className}>
        <PostsGrid {...others} />
        <button onClick={this.handleLoadMoreClick}>Load
more</button>
      </div>
    );
  }
}
```

Так же мы можем сочетать JSX распространение атрибутов с регулярными атрибутами, пользуясь простым правилом приоритета для реализации переопределения значений и установки значений атрибута по умолчанию. Этот элемент получит переопределеное значение атрибута *className*, даже если свойство *className* существует в *this.props*:

```
<div {...this.props} className="override">
  …
</div>
```

Атрибут *className* этого элемента по умолчанию имеет значение «base», если не существует свойства *className* в *this.props* чтобы переопределить его:

```
<div className="base" {...this.props}>
  …
</div>
```

Спасибо за прочтение

Я надеюсь что вам, так же как нам, понравится использовать возможности языка ES6+ для написания React кода. Спасибо моим коллегам за их вклад в эту статью, и особая благодарность команде Babel за то что они делают будущее доступным для всех нас, уже сегодня.

Проголосовать:

+17

Поделиться:

Сохранить:

Комментарии (15)

Похожие публикации

Frontend на CodeFest: React, Javascript и лучшие практики

3

sereje4kin • 27 февраля в 11:19

Hyperapp для беженцев с React/Redux

38

ПЕРЕВОД

MrCheater • 26 февраля в 08:51

Шаблоны проектирования в React

4

ПЕРЕВОД

ru_vds • 16 февраля в 11:35

Популярное за сутки

Яндекс открывает Алису для всех разработчиков. Платформа Яндекс.Диалоги (бета) 69

BarakAdama • вчера в 10:52

Почему следует игнорировать истории основателей успешных стартапов 20

ПЕРЕВОД

m1rko • вчера в 10:44

Как получить телефон (почти) любой красотки в Москве, или интересная особенность MT_FREE 24

ИЗ ПЕСОЧНИЦЫ

cab404 • вчера в 20:27

Java и Project Reactor 10

zealot_and_frenzy • вчера в 10:56

Пользовательские агрегатные и оконные функции в PostgreSQL и Oracle 6

erogov • вчера в 12:46

Лучшее на Geektimes

Как фермеры Дикого Запада организовали телефонную сеть на колючей проволоке

31

NAGru • вчера в 10:10

Энтузиаст сделал новую материнскую плату для ThinkPad X200s

49

alizar • вчера в 15:32

Кто-то посылает секс-игрушки с Amazon незнакомцам. Amazon не знает, как их остановить

85

Pochtoycom • вчера в 13:06

Илон Маск продолжает убеждать в необходимости создания колонии людей на Марсе

139

marks • вчера в 14:19

Дела шпионские (часть 1)

16

TashaFridrih • вчера в 13:16