

React入门和实战

前端 蔡超

简介

最简例子

JSX

组件

事件

状态

稍复杂的例子

简介

2014 最火爆的前端技术

2014 最火爆的前端技术 之一



出品



~~Need legal assistance~~
with your California
craft beer? www.craftbeerlaw.com

Bigpipe

组件化

推荐视频



天蝎上身 迈凯伦12C声浪 ▶ 1.5万



震惊了! 油门/刹车较劲 ▶ 4.1万



悲催 警车意外车祸合集 ▶ 5.6万



宝马M4加速对比 奔驰C63 ▶ 1.3万

[进入视频频道 >](#)

Just the UI

只专注与MVC中的V，M和C交给其他框架去实现

Virtual DOM

使用Virtual DOM实现了超高的性能

Data flow

单向数据流，区别于其他库框架的双向绑定概念，更简单

易开发

易维护

模块化

服务端

最简例子

最简例子

React.render方法

```
React.render(  
  <div>Hello!</div>,  
  document.body  
);
```

```
React.render(  
  <div>Hello World!</div>,  
  document.getElementById('result')  
);
```

第二个参数不能少，父节点

看例子：1hello.html, 2append.html

JSX

JSX

JSX是一个XML语法的预处理器。使用React时可以不使用JSX，但JSX已经基本上成为标配了。由于是XML，所以是大小写敏感的！

Live JSX Editor

```
var HelloMessage = React.createClass({
  render: function() {
    return <div>Hello {this.props.name}</div>;
  }
});

React.render(<HelloMessage name="John" />, mountNode);
```

使用了JSX

```
var HelloMessage = React.createClass({displayName: "HelloMessage",
  render: function() {
    return React.createElement("div", null, "Hello ", this.props.name);
  }
});

React.render(React.createElement(HelloMessage, {name: "John"}),
mountNode);
```

未使用JSX

JSX在线编译器

JSX

```
<div className="red">Children Text</div>  
<MyCounter count={3 + 5} />
```

```
var gameScores = {  
  player1: 2,  
  player2: 5  
};
```

```
<DashboardUnit data-index="2">  
  <h1>Scores</h1>  
  <Scoreboard className="results" scores={gameScores} />  
</DashboardUnit>
```

就像XML一样，JSX的标签包括一个标签名，若干属性，还有子节点。
双引号包起来的是字符串，花括号包起来的是JS表达式。

组件 Componets

组件 Componets

核心

组件 Componets

核心

使用`React.createClass`来创建一个组件，创建组件只有一个要求，暨需要实现`render`方法。该方法定义组件将被怎样渲染。

```
// 这里定义了组件
var MessageComponent = React.createClass({
  render: function() {
    return (
      <div>{this.props.message}</div>
    );
  }
});

// 这里使用了组件（渲染到body）
React.render(
  <MessageComponent message="Hello!" />,
  document.body
);
```

组件 Componets

组件的属性

这些属性，在组件被render后，可以使用`this.props`来直接访问到。

```
// 这里定义了组件
var MessageComponent = React.createClass({
  render: function() {
    return (
      <div>{this.props.message}</div>
    );
  }
});

// 这里使用了组件（渲染到body）
React.render(
  <MessageComponent message="Hello!" />,
  document.body
);
```

看例子：[4props.html](#)

事件 Events

事件 Events

首先，创建个组件，后我们要使用行内事件处理器（inline event handlers）进行事件处理。众所周知，onclick是个很差的事件处理方案，但是在React中并不是这样。

React自动帮你完成了事件函数的bind(this)工作哦 💕

```
var BannerAd = React.createClass({  
  onBannerClick: function(evt) {  
    // codez to make the moneys  
  },  
  
  render: function() {  
    return <div onClick={this.onBannerClick}  
    >Click Me!</div>;  
  }  
});
```

由于JSX是基于XML的，所以大小写敏感，要注意哦 😊

看例子： [5event.html](#)

状态 State

状态 State

React中引入的一个新概念，区别于属性和事件，理解起来稍难。

状态(**state**)与属性(**props**)最大的区别在于：状态是组件内部且被组件自行修改的，而属性是可以通过外部注入或者修改的。

看例子：[6state.html](#)

状态 State

getInitialState

该接口返回组件状态的初始化值，键-值对象类型。

```
getInitialState: function() {  
  return {  
    clicks: 0  
  };  
}
```

this.state

访问一个组件的状态，使用**this.state**，就像使用**this.props**一样。

看例子：[6state.html](#)

状态 State

this.setState

更新一个组件的状态，传入一个键值组合。

```
this.setState({  
  clicks: this.state.clicks + 1  
})
```

当组件的一个状态变化时，渲染器将使用新的状态值与UI重新渲染组件。

这是React实现的核心。

看例子：[6state.html](#)

稍复杂的例子

稍复杂的例子

组件的组合 看例子: [7page.html](#)

列表（循环） 看例子: [8list.html](#)

Q&A

下载本教程

<https://github.com/hkongm/ReactGuide>