**React**

**1 现在最热门的前端框架，毫无疑问是 React 。**

上周，基于 React 的 React Native 发布，结果一天之内，就获得了 5000 颗星，受瞩目程度可见一斑。React 起源于 Facebook 的内部项目，因为该公司对市场上所有 JavaScript MVC 框架，都不满意，就决定自己写一套，用来架设 Instagram 的网站。做出来以后，发现这套东西很好用，就在2013年5月开源了。

**2 HTML 模板**

<!DOCTYPE html>

<html>

<head>

<script src="../build/react.js"></script>

<script src="../build/react-dom.js"></script>

<script src="../build/browser.min.js"></script>

</head>

<body>

<div id="example"></div>

<script type="text/babel">

// \*\* Our code goes here! \*\*

</script>

</body>

</html>

上面代码有两个地方需要注意。

首先，最后一个 <script> 标签的 type 属性为 text/babel 。这是因为 React 独有的 JSX 语法，跟 JavaScript 不兼容。凡是使用 JSX 的地方，都要加上 type="text/babel" 。

其次，上面代码一共用了三个库： react.js 、react-dom.js 和 Browser.js ，它们必须首先加载。其中，react.js 是 React 的核心库，react-dom.js 是提供与 DOM 相关的功能，Browser.js 的作用是将 JSX 语法转为 JavaScript 语法，这一步很消耗时间，实际上线的时候，应该将它放到服务器完成。

**3 ReactDOM.render()**

ReactDOM.render 是 React 的最基本方法，用于将模板转为 HTML 语言，并插入指定的 DOM 节点。

ReactDOM.render(

<h1>Hello, world!</h1>,

document.getElementById('example')

);

**4 JSX 语法**

var names = ['Alice', 'Emily', 'Kate'];

ReactDOM.render(

<div>

{

names.map(function (name) {

return <div>Hello, {name}!</div>

})

}

</div>,

document.getElementById('example')

);

var arr = [

<h1>Hello world!</h1>,

<h2>React is awesome</h2>,

];

ReactDOM.render(

<div>{arr}</div>,

document.getElementById('example')

);

上面代码的arr变量是一个数组，结果 JSX 会把它的所有成员，添加到模板。

上面代码体现了 JSX 的基本语法规则：遇到 HTML 标签（以 < 开头），就用 HTML 规则解析；遇到代码块（以 { 开头），就用 JavaScript 规则解析。HTML 语言直接写在 JavaScript 语言之中，不加任何引号，这就是 JSX 的语法，它允许 HTML 与 JavaScript 的混写。

**5 组件**

var HelloMessage = React.createClass({

render: function() {

return <h1>Hello {this.props.name}</h1>;

}

});

ReactDOM.render(

<HelloMessage name="John" />,

document.getElementById('example')

);

React 允许将代码封装成组件（component），然后像插入普通 HTML 标签一样，在网页中插入这个组件。React.createClass 方法就用于生成一个组件类。上面代码中，变量 HelloMessage 就是一个组件类。模板插入 <HelloMessage /> 时，会自动生成 HelloMessage 的一个实例（下文的"组件"都指组件类的实例）。所有组件类都必须有自己的 render 方法，用于输出组件。

注意，组件类的第一个字母必须大写，否则会报错，比如HelloMessage不能写成helloMessage。另外，组件类只能包含一个顶层标签，否则也会报错。

**6 this.props.children**

var NotesList = React.createClass({

render: function() {

return (

<ol>

{

React.Children.map(this.props.children, function (child) {

return <li>{child}</li>;

})

}

</ol>

);

}

});

ReactDOM.render(

<NotesList>

<span>hello</span>

<span>world</span>

</NotesList>,

document.body

);

this.props 对象的属性与组件的属性一一对应，但是有一个例外，就是 this.props.children 属性。它表示组件的所有子节点

这里需要注意， this.props.children 的值有三种可能：如果当前组件没有子节点，它就是 undefined ;如果有一个子节点，数据类型是 object ；如果有多个子节点，数据类型就是 array 。所以，处理 this.props.children 的时候要小心。

React 提供一个工具方法 React.Children 来处理 this.props.children 。我们可以用 React.Children.map 来遍历子节点，而不用担心 this.props.children 的数据类型是 undefined 还是 object。更多的 React.Children 的方法

**7 PropTypes**

var MyTitle = React.createClass({

propTypes: {

title: React.PropTypes.string.isRequired,

},

render: function() {

return <h1> {this.props.title} </h1>;

}

});

上面的Mytitle组件有一个title属性。PropTypes 告诉 React，这个 title 属性是必须的，而且它的值必须是字符串。现在，我们设置 title 属性的值是一个数值。

var data = 12;

ReactDOM.render(

<MyTitle title={data} />,

document.body

);

这样一来，title属性就通不过验证了。控制台会显示一行错误信息。

Warning: Failed propType: Invalid prop `title` of type `number` supplied to `MyTitle`, expec

**8 获取真实的DOM节点**

var MyComponent = React.createClass({

handleClick: function() {

this.refs.myTextInput.focus();

},

render: function() {

return (

<div>

<input type="text" ref="myTextInput" />

<input type="button" value="Focus the text input" onClick={this.handleClick} />

</div>

);

}

});

ReactDOM.render(

<MyComponent />,

document.getElementById('example')

);

组件并不是真实的 DOM 节点，而是存在于内存之中的一种数据结构，叫做虚拟 DOM （virtual DOM）。只有当它插入文档以后，才会变成真实的 DOM 。根据 React 的设计，所有的 DOM 变动，都先在虚拟 DOM 上发生，然后再将实际发生变动的部分，反映在真实 DOM上，这种算法叫做 DOM diff ，它可以极大提高网页的性能表现。

但是，有时需要从组件获取真实 DOM 的节点，这时就要用到 ref 属性

上面代码中，组件 MyComponent 的子节点有一个文本输入框，用于获取用户的输入。这时就必须获取真实的 DOM 节点，虚拟 DOM 是拿不到用户输入的。为了做到这一点，文本输入框必须有一个 ref 属性，然后 this.refs.[refName] 就会返回这个真实的 DOM 节点。

需要注意的是，由于 this.refs.[refName] 属性获取的是真实 DOM ，所以必须等到虚拟 DOM 插入文档以后，才能使用这个属性，否则会报错。上面代码中，通过为组件指定 Click 事件的回调函数，确保了只有等到真实 DOM 发生 Click 事件之后，才会读取 this.refs.[refName] 属性。

React 组件支持很多事件，除了 Click 事件以外，还有 KeyDown 、Copy、Scroll 等

**9 this.state状态机**

var LikeButton = React.createClass({

getInitialState: function() {

return {liked: false};

},

handleClick: function(event) {

this.setState({liked: !this.state.liked});

},

render: function() {

var text = this.state.liked ? 'like' : 'haven\'t liked';

return (

<p onClick={this.handleClick}>

You {text} this. Click to toggle.

</p>

);

}

});

ReactDOM.render(

<LikeButton />,

document.getElementById('example')

);

组件免不了要与用户互动，React 的一大创新，就是将组件看成是一个状态机，一开始有一个初始状态，然后用户互动，导致状态变化，从而触发重新渲染 UI

上面代码是一个 LikeButton 组件，它的 getInitialState 方法用于定义初始状态，也就是一个对象，这个对象可以通过 this.state 属性读取。当用户点击组件，导致状态变化，this.setState 方法就修改状态值，每次修改以后，自动调用 this.render 方法，再次渲染组件。

由于 this.props 和 this.state 都用于描述组件的特性，可能会产生混淆。一个简单的区分方法是，this.props 表示那些一旦定义，就不再改变的特性，而 this.state 是会随着用户互动而产生变化的特性。

**10 表单**

var Input = React.createClass({

getInitialState: function() {

return {value: 'Hello!'};

},

handleChange: function(event) {

this.setState({value: event.target.value});

},

render: function () {

var value = this.state.value;

return (

<div>

<input type="text" value={value} onChange={this.handleChange} />

<p>{value}</p>

</div>

);

}

});

ReactDOM.render(<Input/>, document.body);

上面代码中，文本输入框的值，不能用 this.props.value 读取，而要定义一个 onChange 事件的回调函数，通过 event.target.value 读取用户输入的值。textarea 元素、select元素、radio元素都属于这种情况

**11 组件的生命周期**

组件的生命周期分成三个状态：

Mounting：已插入真实 DOM

Updating：正在被重新渲染

Unmounting：已移出真实 DOM

React 为每个状态都提供了两种处理函数，will 函数在进入状态之前调用，did 函数在进入状态之后调用，三种状态共计五种处理函数。

componentWillMount()

componentDidMount()

componentWillUpdate(object nextProps, object nextState)

componentDidUpdate(object prevProps, object prevState)

componentWillUnmount()

此外，React 还提供两种特殊状态的处理函数。

componentWillReceiveProps(object nextProps)：已加载组件收到新的参数时调用

shouldComponentUpdate(object nextProps, object nextState)：组件判断是否重新渲染时调用

var Hello = React.createClass({

getInitialState: function () {

return {

opacity: 1.0

};

},

componentDidMount: function () {

this.timer = setInterval(function () {

var opacity = this.state.opacity;

opacity -= .05;

if (opacity < 0.1) {

opacity = 1.0;

}

this.setState({

opacity: opacity

});

}.bind(this), 100);

},

render: function () {

return (

<div style={{opacity: this.state.opacity}}>

Hello {this.props.name}

</div>

);

}

});

ReactDOM.render(

<Hello name="world"/>,

document.body

);

**12 Ajax**

var UserGist = React.createClass({

getInitialState: function() {

return {

username: '',

lastGistUrl: ''

};

},

componentDidMount: function() {

$.get(this.props.source, function(result) {

var lastGist = result[0];

if (this.isMounted()) {

this.setState({

username: lastGist.owner.login,

lastGistUrl: lastGist.html\_url

});

}

}.bind(this));

},

render: function() {

return (

<div>

{this.state.username}'s last gist is

<a href={this.state.lastGistUrl}>here</a>.

</div>

);

}

});

ReactDOM.render(

<UserGist source="https://api.github.com/users/octocat/gists" />,

document.body

);

上面代码使用 jQuery 完成 Ajax 请求，这是为了便于说明。React 本身没有任何依赖，完全可以不用jQuery，而使用其他库。

**13 优点**

1）.声明式设计 −React采用声明范式，可以轻松描述应用。

2）.高效 −React通过对DOM的模拟，最大限度地减少与DOM的交互。

3）.灵活 −React可以与已知的库或框架很好地配合。

4）.JSX − JSX 是 JavaScript 语法的扩展。React 开发不一定使用 JSX ，但我们建议使用它。

5）.组件 − 通过 React 构建组件，使得代码更加容易得到复用，能够很好的应用在大项目的开发中。

6）.单向响应的数据流 − React 实现了单向响应的数据流，从而减少了重复代码，这也是它为什么比传统数据绑定更简单。

7） 能够实现服务器端的渲染，便于搜索引擎优化。这一点要比Backbone, Angular 1.x和Ember早期强

8） 能够很好的和现有的代码结合。React只是MVC中的View层，对于其他的部分并没有硬性要求。意味着很公司在选择用Angular全部重构和用React部分重构的时候，选择了React部分重构

9） 因为一切都是component，所以代码更加模块化，重用代码更容易

10） 学起来非常容易，几个小时就可以入门

11） 因为强调只从this.props和this.state生成HTML，写起来bug比较少

12）、React速度很快：它并不直接对DOM进行操作，引入了一个叫做虚拟DOM的概念，安插在javascript逻辑和实际的DOM之间，性能好。

13）、跨浏览器兼容：虚拟DOM帮助我们解决了跨浏览器问题，它为我们提供了标准化的API，甚至在IE8中都是没问题的。

14）、一切都是component：代码更加模块化，重用代码更容易，可维护性高。

15）、单向数据流：Flux是一个用于在JavaScript应用中创建单向数据层的架构，它随着React视图库的开发而被Facebook概念化。

16）、同构、纯粹的javascript：因为搜索引擎的爬虫程序依赖的是服务端响应而不是JavaScript的执行，预渲染你的应用有助于搜索引擎优化。

17）、兼容性好：比如使用RequireJS来加载和打包，而Browserify和Webpack适用于构建大型应用。它们使得那些艰难的任务不再让人望而生畏。

**14 缺点**

1） 并不是一个完整的框架，基本都需要加上ReactRouter和Flux才能写大型应用

2）React本身只是一个V而已，并不是一个完整的框架，所以如果是大型项目想要一套完整的框架的话，基本都需要加上ReactRouter和Flux才能写大型应用。

3）大概就是现在还太新了很难说将来有没有大的API变化，目前在大的稳定的项目上采用React的，我也就只知道有Yahoo的Email。

所以现在很少有批评React的声音，也许不是他真的就没有坑，而是那些坑还没有被踩出来而已。