

React

悟空

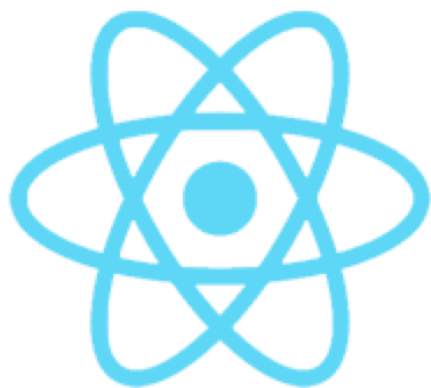
2015年09月07日

尚妆

SHOWJOY.COM

正 品 无 须 代 言

React – form Facebook



React

用于构建用户界面的Javascript库

仅仅是UI

许多人使用React作为MVC架构的V层。尽管React并没有假设过你的其余技术栈，但它仍可以作为一个小特征轻易地在已有项目中使用

虚拟DOM

React为了更高超的性能而使用虚拟DOM作为其不同的实现。它同时也可以由服务端Node.js渲染 – 而不需要过重的浏览器DOM支持

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。

前端组件化

一个Component拯救世界，忘掉烦恼，从此不再操心界面。

JSX -- a faster, safer, easier JavaScript

JSX

通过解析类似于XML的语法来完成一些JavaScript代码完成的内容。

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

属于JSX部分的代码，通过JSX的方式可以使我们在写一些组件代码的时候更加附和平时写html代码时候的习惯。

对比js的写法（相对复杂）：

```
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);
```

JSX -- a faster, safer, easier JavaScript

```
<script src="plugins/JSXTransformer.js"></script>
<script type="text/jsx" src="src/public-react-1.js"></script>
```

★ react-tools public

A set of complementary tools to React, including the JSX transformer.

This package compliments the usage of **React**. It ships with tools that are often used in conjunction.

通过npm安装react-tools可以方便的编译JSX文件而不通过上面引入的方式

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

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

React 组件非常简单。你可以认为它们就是简单的函数，接受 **props** 和 **state** (后面会讨论) 作为参数，然后渲染出 HTML。

React -- Example

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hello React</title>
    <script src="http://fb.me/react-0.13.0.js"></script>
    <script src="http://fb.me/JSXTransformer-0.13.0.js"></script>
  </head>
  <body>
    <div id="example"></div>
    <script type="text/jsx">

      // ** 在这里替换成你的代码 **

    </script>
  </body>
</html>
```

React.render() 实例化根组件，启动框架，注入标记到原始的 DOM 元素中，作为第二个参数提供。

引入解析React文件和JSX的解析文件。

```
var HelloWorld = React.createClass({
  render: function() {
```

React.createClass()
创建一个新的React组件

```
    return (
      <p>
        Hello, <input type="text" placeholder="Your name here" />!
        It is {this.props.date.toTimeString()}
      </p>
    );
```

创建组件内部的内容

Props : 从父级获取数据

```
setInterval(function() {
  React.render(
    <HelloWorld date={new Date()} />,
    document.getElementById('example')
  );
}, 500);
```

混杂了js代码


```
var HelloWorld = React.createClass({
  render: function() {
    return (
      <p>
        Hello, <input type="text" placeholder="Your name here" />!
        It is {this.props.date.toTimeString()}
      </p>
    );
  }
});
```

```
setInterval(function() {
  React.render(
    <HelloWorld date={new Date()} />,
    document.getElementById('example')
  );
}, 500);
```

Hello, ! It is 17:47:43 GMT+0800 (CST)

Hello, ! It is 17:48:14 GMT+0800 (CST)

Hello, ! It is 17:48:27 GMT+0800 (CST)

数据在时刻进行改变的，但是和我们平时进行的操作是不同的。（平时我们需要进行“复杂”的DOM操作）

Virtual DOM

DOM

在Web开发中，我们总需要将变化的数据实时反应到UI上，这时就需要对DOM进行操作。而复杂或频繁的DOM操作通常是性能瓶颈产生的原因。


在浏览器端用Javascript实现了一套DOM API。基于React进行开发时所有的DOM构造都是通过虚拟DOM进行，每当数据变化时，React都会重新构建整个DOM树，然后React将当前整个DOM树和上一次的DOM树进行对比，得到DOM结构的区别，然后仅仅将需要变化的部分进行实际的浏览器DOM更新。

理解 Virtual DOM 为什么快？

并不是说Virtual DOM 操作DOM就比我们直接通过js代码操作DOM的方式要快捷，**React并不解决DOM本身慢的问题。**

React 中的 Diff 操作 更加智能的计算得出优化步骤

```
<ul>
<li>0</li>
<li>1</li>
<li>2</li>
<li>3</li>
</ul>
```



```
<ul>
<li>6</li>
<li>7</li>
<li>8</li>
<li>9</li>
<li>10</li>
</ul>
```

正常的 DOM 操作：

先把0，1，2，3这些Element删掉，然后加几个新的Element 6，7，8，9，10进去，这里面就有4次Element删除，5次Element添加。

React 的操作：

把这两个做一下Diff，然后发现其实不用删除0，1，2，3，而是可以直接改innerHTML，然后只需要添加一个Element（10）就行了，这样就是4次innerHTML操作加1个Element添加

“你给我一个数据，我根据这个数据生成一个全新的Virtual DOM，然后跟我上一次生成的Virtual DOM去 diff，得到一个Patch，然后把这个Patch打到浏览器的DOM上去，完事。”

渠道效果监控

商品受访数据监控

受访页面分析

搜索记录监控

会员信息监控

渠道效果监控

今天

昨天

最近7天

最近30天

```
<!DOCTYPE html>
<html>
<head lang="en">
  <meta http-equiv='Content-type' content='text/html; charset=utf-8'>
  <title></title>
  <!--CSS Link-->
  <link rel="stylesheet" type="text/css" href="plugins/bootstrap-3.3.4-dist/css/bootstrap.min.css"/>
  <link rel="stylesheet" type="text/css" href="css/public-min.css"/>
  <!--Js Link-->
  <script src="plugins/jquery-1.11.3.min.js"></script>
  <script src="plugins/bootstrap-3.3.4-dist/js/bootstrap.min.js" type="text/javascript"></script>
  <script src="plugins/bootstrap-datetimepicker.js" type="text/javascript"></script>
  <script src="plugins/bootstrap-datetimepicker.zh-CN.js" type="text/javascript"></script>
  <script src="plugins/react.js"></script>
  <script src="plugins/JSXTransformer.js"></script>
  <script type="text/jsx" src="src/public-react-1.js"></script>
</head>
<body>
<div id="header">
</div>
<div id="wrapper">
</div>
<div id="footer">
</div>
</body>
</html>
```

完全用React
构建的页面。

都采用组件化的方式
聚合在js中

利弊？

React 组件化

Data Analysis ShowJoy

 user name

退出

渠道效果监控

商品受访数据监控

受访页面分析

搜索记录监控

会员信息监控

渠道效果监控

今天

昨天

最近7天

最近30天

划分的粒度 & 可复用的程度

```
var HeadNav = React.createClass({
  render: function() {
    return (
      <div className="header-nav">
        <HeadNavLogoFont firLogoName="Data Analysis" secLogoName="ShowJoy"/>
        <HeadUser imgPath="img/public-head-img.png" userName="user name"/>
        <ClearDiv/>
      </div>
    );
  }
});
```

```
var HeadNavLogoFont = React.createClass({
  render: function() {
    return (
      <div className='header-logo'>
        <span className='header-logo-fir'>{this.props.firLogoName}</span>
        <span className='header-logo-sec'>{this.props.secLogoName}</span>
      </div>
    );
  }
});
```

```
var HeadUser = React.createClass({
  render: function() {
    return (
      <div className='header-user'>
        <div className='header-user-ori'>
          <img src={this.props.imgPath}/>
          <span>{this.props.userName}</span>
        </div>
        <div className='header-user-hover j_header-user-hover'>退出</div>
      </div>
    );
  }
});
```



user nameuser na...

退出

{value} : 表示变量

className == class

.....

React – Props & State

数据的呈现 —— 交互式界面

```
var FancyCheckbox = React.createClass({
  render: function() {
    var fancyClass = this.props.checked ? 'FancyChecked' : 'FancyUnchecked';
    return (
      <div className={fancyClass} onClick={this.props.onClick}>
        {this.props.children}
      </div>
    );
  }
});

React.render(
  <FancyCheckbox checked={true} onClick={console.log.bind(console)}>
    Hello world!
  </FancyCheckbox>,
  document.body
);
```

! 不可以在子级对props进行修改 !

可以把props理解成一种从父级到子级静态的数据展示。

我的理解：

当修改了父级的数据时对整个组件进行再次渲染？如何直接改变父级props？

React – Props & State

```
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>
    );
  }
});

React.render(
  <LikeButton />,
  document.getElementById('example')
);
```

**getInitialState: function(){}
设置组件的初始化状态**

**setState()
设置组件的状态**

绑定事件 通过事件触发组件的状态改变

组件其实是状态机

State Machines

把用户界面想像成拥有不同状态然后渲染这些状态

React 里，只需更新组件的 state，然后根据新的 state 重新渲染用户界面（不要操作 DOM）。React 来决定如何最高效地更新 DOM。

React – Props ? State ?

什么时候应该使用State ?

大部分组件的工作应该是从 props 里取数据并渲染出来。但是，有时需要对用户输入、服务器请求或者时间变化等**作出响应**，这时才需要使用 State。

尝试把尽可能多的组件无状态化。

常用的模式是**创建多个只负责渲染数据的无状态 (stateless) 组件**，在它们的上层**创建一个有状态 (stateful) 组件并把它的状态通过 props 传给子级**。这个有状态的组件封装了所有用户的交互逻辑，而这些无状态组件则负责声明式地渲染数据。

在实践中学习 ~

来源	UV	访问深度	跳失	停留时间	注册数	加购物车次数	订单数	成交额	转化率
http://www....	1212121212	12121212...	12121212...	12432s	21212121	12121212	12121212	1212121212	12121212%
www.baidu....	78787878	7878787878	78787877	5656556s	09090	232323	565656	67667	212121%
http://www....	989898989	67676776...	34343	98989s	1232313	324234	456546456	324234234	4565463%
http://1231...	22	1232	1223	24413s	1723	1253	1283	21031	2113%
http://1231...	21232	132424232	1567223	56724413s	3531723	1234253	23451283	45252436...	5422113%
http://1231...	252452	2521232	3453261223	524524413s	24521723	4351253	65361283	657821031	53425211...
http://1231...	25422	143768232	12859523	63546244...	87651723	365361253	5361283	263541031	2113%
http://1231...	2632	126256232	1265323	2441633s	177323	14365463...	3653371283	3563621031	56363211...
http://1231...	265362	1235555552	125555523	24455555...	15555723	125553	1555283	2155031	2555113%

如何通过React来把这个表格动态化的显示出来？

React

采用原来的方式，当表格需要进行改变的时候需要进行繁琐的DOM操作：

```
/******重置表格******/
var tableResetOperation = function(ison){
  $('tbody').remove();
  $('table').append(tableBody(jison));
  sortOperation(jison);
  setFormat();
};

var resetTableBody = function(array, jsonColumns){
  $('tbody').remove();
  var tbodyData = '<tbody>';
  for(var i=0; i<array.length; i++){
    tbodyData += '<tr>';
    $.each(array[i], function(index, data){
      tbodyData += '<td class="j_' + index + setCategory(jsonColumns, index) + '>' + data + '</td>';
    });
    tbodyData += '</tr>';
  }
  tbodyData += '</tbody>';
  return tbodyData;
};
```

而当我们采用React来处理这个表格的时候，我们需要转换我们的思路：

我们不需要担心DOM操作是怎样进行的，我们需要关注数据的传输和数据之间的关系。

来源	UV	访问深度	跳失	停留时间	注册数	加购物袋次数	订单数	成交额	转化率
http://www....	1212121212	12121212...	12121212...	12432s	21212121	12121212	12121212	1212121212	12121212%
www.baidu....	78787878	7878787878	78787877	5656556s	09090	232323	565656	67667	212121%
http://www....	989898989	67676776...	34343	98989s	1232313	324234	456546456	324234234	4565463%
http://1231...	22	1232	1223	24413s	1723	1253	1283	21031	2113%
http://1231...	21232	132424232	1567223	56724413s	3531723	1234253	23451283	45252436...	5422113%
http://1231...	252452	2521232	3453261223	524524413s	24521723	4351253	65361283	657821031	53425211...
http://1231...	25422	143768232	12859523	63546244...	87651723	365361253	5361283	263541031	2113%
http://1231...	2632	126256232	1265323	2441633s	177323	14365463...	3653371283	3563621031	56363211...
http://1231...	265362	1235555552	125555523	24455555...	15555723	125553	1555283	2155031	2555113%

数据传进来之后表格内容自己会进行改变，我们的关注点只关注在：

我们如何构件表格，表格内数据之间是否有什么关联的关系。

高效视图

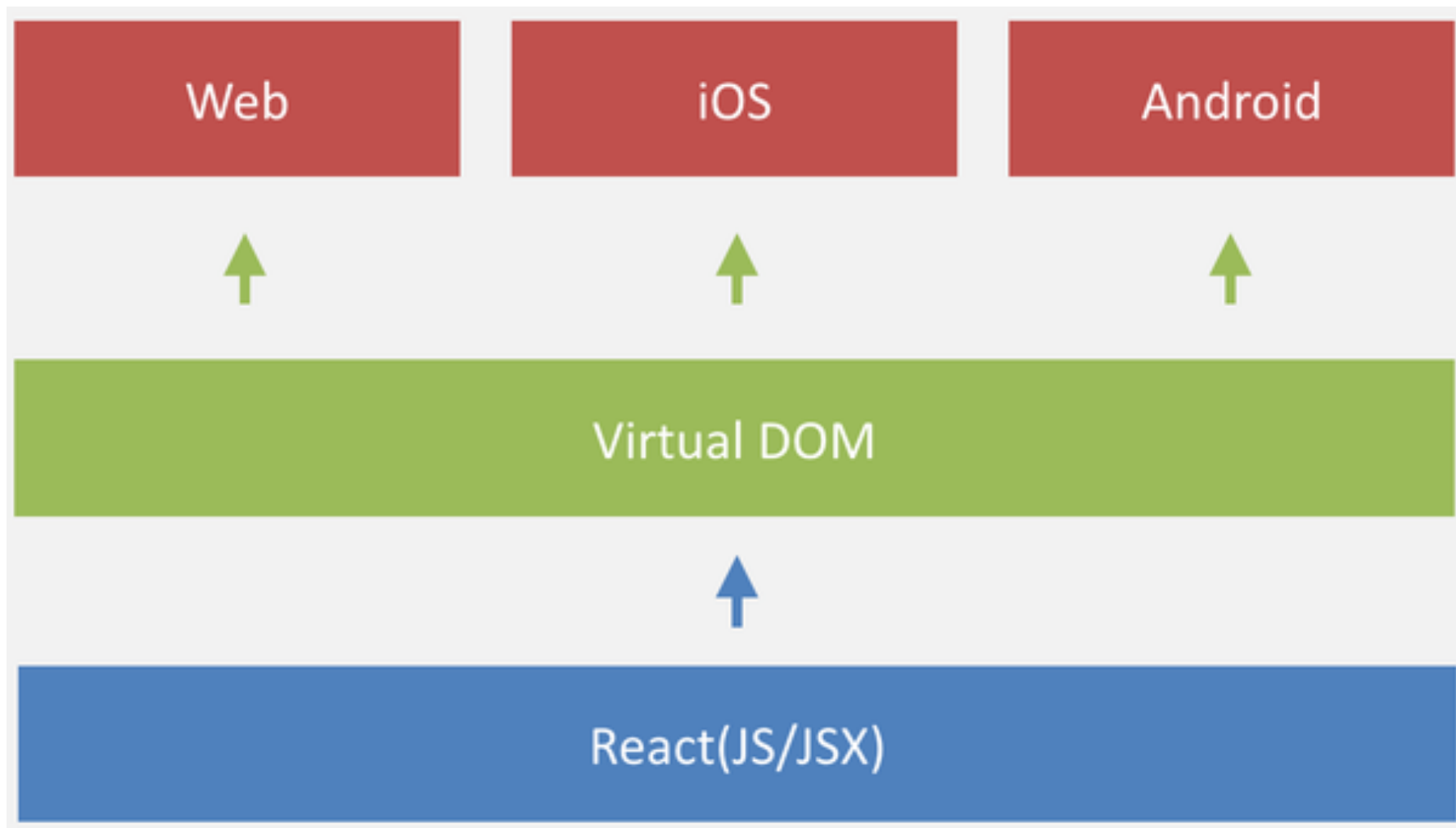
React.js并不是一个MVC框架，而是一个高效的视图。以组件开发为基础，分割你的页面使之变成一个个分离的、可复用的组件。**组件驱动开发**

关注数据，Virtual DOM

React是管理DOM的操作的，通过独有的diffing算法帮助计算出DOM中需要改变的地方，减少因为DOM操作而带来的消耗。

.....

下期预告：React Native





THANKS FOR YOUR ATTENTION

尚妆
SHOWJOY.COM
正 品 无 须 代 言