React 是 Facebook 推出的一个用来构建用户界面的 JavaScript 库。具备以下特性：

* 不是一个 MVC 框架
* 不使用模板
* 响应式更新非常简单
* HTML5 仅仅是个开始
* 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。
* 一个简单的组件
* React 组件通过一个render()方法，接受输入的参数并返回展示的对象。
* 以下这个例子使用了 JSX，它类似于XML的语法
* 输入的参数通过render()传入组件后，将存储在this.props
* JSX 是可选的，并不强制要求使用。
* 点击 "Compiled JS" 可以看到 JSX 编译之后的 JavaScript 代码
* Live JSX Editor
* var HelloMessage = React.createClass({
* render: function() {
* return <div>Hello {this.props.name}</div>;
* }
* });
* React.render(<HelloMessage name="John" />, mountNode)
* Hello John
* Compiled JS
* 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);
* 一个有状态的组件
* 除了接受输入数据（通过 this.props ），组件还可以保持内部状态数据（通过 this.state ）。当一个组件的状态数据的变化，展现的标记将被重新调用 render() 更新。
* 一个应用程序
* 通过使用props和state, 我们可以组合构建一个小型的 Todo 程序。
* 下面例子使用state去监测当前列表的项以及用户已经输入的文本。 尽管事件绑定似乎是以内联的方式，但他们将被收集起来并以事件代理的方式实现。
* 一个使用外部插件的组件
* React 是灵活的，并且提供方法允许你跟其他库和框架对接。
* 下面例子展现了一个案例，使用外部库 Markdown 实时转化 textarea 的值。
* Live JSX Editor

# 慕课网总结

**一、 jsx**

　　1. 被称为语法糖：糖衣语法，计算机语言中添加的某种语法，对语言的功能没有影响，更方便程序员使用，增加程序的可读性，降低出错的可能性

　　　　类似的还有（coffeescript，typescript），最终都被解析为js

　　2. 解析jsx的是jsxtransformer.js      指定jsx语法用<text/jsx>

　　3. 通过以下代码渲染dom

1 React.render(<hello name=”world”/>,

2 Document.getElementbyId(“container”));

[复制代码](javascript:void(0);)

var hello = React.createClass({

render: function(){

return <div>hello {this.props.name}</div>;

}

});

React.render(<hello name="world"/>,

document.getElementById("container'));

[复制代码](javascript:void(0);)

　　props是指属性组，this是实例

**二、 css**

　　1. class

　　　不能在标签上直接写class，需要改为className  (由于此处非真正的dom，class是关键字，不能解析）

var Hello = React.createClass({

render: function(){

return <div className="redfont">hello {this.props.name}</div>;

}

});

　　2. 内联式

　     不能字符串来表示，需要用样式对象来表示，样式对象是以驼峰标示写法，值为样式值

var Introduce = React.createClass({

render: function(){

return <div style={{fontSize:'44px'}}>{this.props.info}</div>;

}

});

　　{}中是执行表达式

　　{{}}内联样式写法

　　驼峰表达式：

[复制代码](javascript:void(0);)

render: function(){

//定义样式内容，样式obj

var styleObj = {

color: 'red',

fontSize:'32px'

};

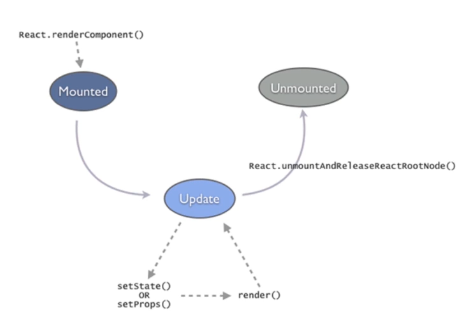
//className代替class

return <div className="redfont" style={styleObj}>hello {this.props.name}</div>;

}

[复制代码](javascript:void(0);)

**三、React components生命周期**



　　1. Mounted是:React Components被render解析生成对应的DOM节点并被插入浏览器的DOM结构的一个过程。

　　2. Update是:一个mounted的React Components被重新render的过程。  
　　　对比当前state变化

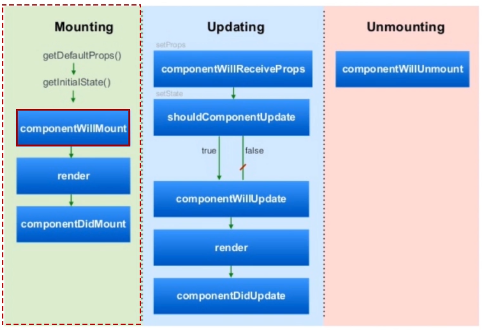
　　　State

　　　每一个状态react都封装了对应的hook函数~钩子

　　　这种函数是Windows消息处理机制的一部分，通过设置“钩子”，应用程序可以在系统级对所有消息、事件进行过滤，访问在正常情况下无法访问的消息。

　　　对事件进行hook后系统会受到相应通知

　　3.Unmounted是:一个mounted的React Components对应的DOM节点被从DOM结构中移除的这样一个过程。



　　　　GetInitialstate

　　　　最终返回一个object其中包含其state

getInitialState:function(){

alert('init');

return {

opacity:1.0

};

}

　　　　This

　　　　是伴随函数生成时的函数内部实例对象

　　　　随着函数运行在不同的环境发生变化

　　　　始终指的是调用函数的那个对象

* 1. 当其出现在settimeout函数参数中时，由于函数参数就是一个纯粹的函数调用，不隶属于其他对象，隶属于全局对象，属于global
  2. 当其出现在setinistialstate这样的函数体内，是作为其所属实例对象的方法来调用，此时this指component实例对象
  3. This出现在构造函数：

function test(){

this.x = 1;

}

var o = new test();

 this 指新生成的对象

4.   This出现在apply call bind等方法

　作用函数的调用对象，指第一个参数

**四、 React-component-listener**

1. Dom更新

　　传统：直接修改dom的innerhtml或classname

　　事件绑定：Eventlistener

　　React：

**给组件添加事件绑定（on驼峰式命名方式）**

[复制代码](javascript:void(0);)

render: function (){  
　　return(  
　　　　<div>  
　　　　　　<button onClick={this.handleClick}>显示|隐藏</button>  
　　　　　　<span ref="tip"> 点击测试</span>  
　　　　</div>  
　　);  
}

[复制代码](javascript:void(0);)

　　添加点击事件：onClick={this.xxxxx}

　　与dom绑定不一样，这里不是真实的dom节点，大小写敏感，通过对象属性定义在对象实例上

var Btnclick = React.createClass({

handleClick: function(event){

},

　　Event对象是在js原生基础上封装的，因此同时具有原生event方法

　　2. 获取组件

　　1）使用‘ref’ property标记组件

　　用ref属性给子组件添加名字，通过this.refs可以索引到子组件

[复制代码](javascript:void(0);)

render: function (){

return(

<div>

<button onClick={this.handleClick}>显示|隐藏</button>

<span ref="tip"> 点击测试</span>

</div>

);

}

[复制代码](javascript:void(0);)

　　this.refs.tip

　　索引到的是react component而不是真实的dom节点

　　2）在dom里获得这个节点：

　　使用react提供的方法：ReactDOM.findDOMNode(react component)

**五、 注意事项**

　　1. 注意react更新后的变化

　　2. 返回虚拟dom时包装为一个div，保证返回一个结果

　　3. 组件的首字母必须大写，不然不报错也不显示

# React入门—阮一峰教程

## jsx

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](http://facebook.github.io/react/docs/displaying-data.html#jsx-syntax)，它允许 HTML 与 JavaScript 的混写

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

ReactDOM.render(

<div>

{

names.map(function (name) {

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

})

}

</div>,

document.getElementById('example')

);

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。

# React实战—画廊

一、yeuman

1. 安装yo：npm install -g yo

2. 查看yo版本：yo --version

3. 查看generator相关npm已经安装包：npm ls -g --depth=1 2>/dev/null grep generator-

npm ls -g --depth=1: x显示包目录最多向下伸展一层

2>/dev/null: 错误消息指向null，即过滤掉错误消息

| : 通道，将上一个命令执行的结果输出到后面命令

grep generator--: 在前面的输出结果中检索generator关键字

克隆远程项目：git clone 'gitadress'

创建脚手架：yo react-webpack projectName

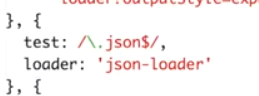
启动服务：npm start or npm run serve

* 1. 通过命令：npm install autoprefixer-loader –save-dev添加自动补全css插件

步骤：

1. 创建data文件夹，添加image.json存储数据
2. 找加载json的loader  
   npm install json-loader –save-dev

在配置文件中添加



1. 在主js中引入image json文件

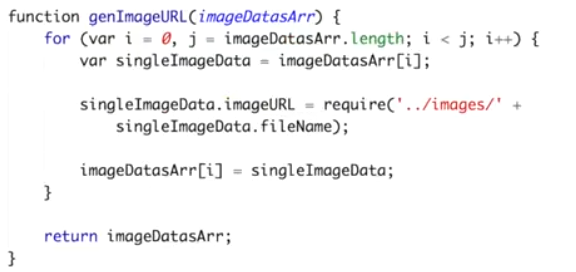


处理imagedatas，将该文件名处理为图片地址

使用的是url-loader

直接引用图片的路径即可，

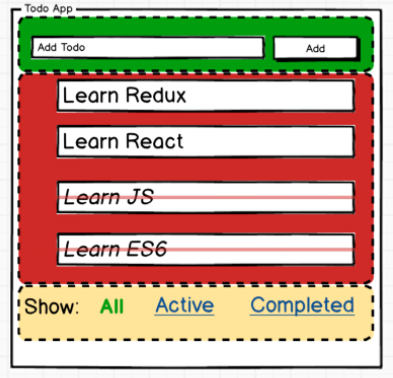


scrollWidth：对象的实际内容的宽度，不包边线宽度，会随对象中内容超过可视区后而变大。   
clientWidth：对象内容的可视区的宽度，不包滚动条等边线，会随对象显示大小的变化而改变。   
offsetWidth：对象整体的实际宽度，包滚动条等边线，会随对象显示大小的变化而改变。

# React-redux

步骤：

1. 写一个详细的界面
2. 划分组件  
     
   每个组件需要：  
   1） 基于数据渲染dom，这种数据成为state  
   2） 监听用户和其他事件，发送这些事件给js函数，这些成为action
3. 列出每个组件的state和action  
   由于我们分为三个组件：addtodo，todolist，filter，分别列出每个组件的action和state  
   1）addtodo组件的状态和行为  
   这个组件中，没有状态，因为组件的外观不会根据任何数据而改变，但是用户创建新的todo时，需要让其他组件知道，把此行为成为“ADD\_TODO”  
   2）todolist状态和行为  
   需要一个todo条目的数组渲染，因此需要一个状态，称之为todos  
   还需要知道哪个filter打开的，可以正确显示todo条目，需要另一个状态，成为TOGGLE\_TODO  
   3）filter组件的状态和行为  
   该组件根据它激活与否，将其本身渲染为一个链接或者一个简单的文本，称这种状态为Currentfilter  
   当用户点击filter时，还需要让其他组件知道，称该行为为SET\_VISIBILITY\_FILTER

# React-Router