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

JSX: JS XML

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

解析jsx的是jsxtransformer.js

指定jsx语法用<text/jsx>

通过

React.render(<hello name=”world”/>,

Document.getElementbyId(“container”));

渲染到dom中



Props是属性组，this是实例

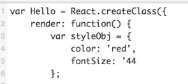
Css

1. 写class  
   不能在标签上直接写class，需要改为classname

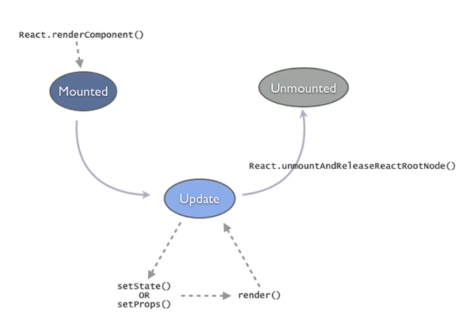
2）内联式  
不能字符串来表示，需要用样式对象来表示，样式对象是以驼峰标示写法，值为样式值  
  
相当于  
  
{}中是执行表达式，因此：

{{ }}内联样式写法

驼峰表达式：



## 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。

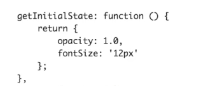
对事件进行hook后系统会受到相应通知

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



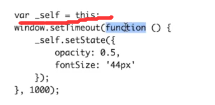
拿到state值：This.state

Props与state

Props：在定义组件时定义，一旦指定一般不改变，尤其对于被调用组件来说，props拥有者是其调用方，结构上是其parent

State：私属于当前组件，可变

改变state：setstate



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

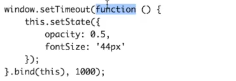
1. 当其出现在settimeout函数参数中时，由于函数参数就是一个纯粹的函数调用，不隶属于其他对象，隶属于全局对象，属于global
2. 当其出现在setinistialstate这样的函数体内，是作为其所属实例对象的方法来调用，此时this指component实例对象
3. This出现在构造函数：



This指新生成的对象

1. 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



Bind到外部的this（component实例）

## React-component-listener

1. Dom更新

传统：直接修改dom的innerhtml或classname

事件绑定：

Eventlistener

React：

stopPropagation()停止事件冒泡

preventDefault()停止事件默认行为

React.findDOMNode(this.fess.<ref属性的值>): 索引子组键的dom元素

新版本需要注意的地方：

1，js文件引用换成最新的

<script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/react/15.3.0/react.min.js"></script>

<script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/react/15.3.0/react-dom.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/babel-core/5.8.23/browser.min.js"></script>

2，

<script type="text/jsx">

换成<script type="text/babel">

3，var tipE = React.findDOMNode(this.refs.tip);

换成var tipE = ReactDOM.findDOMNode(this.refs.tip);

4,React.render

换成ReactDOM.render

用div包裹是为了返回一个结果，而不是两个结果

React.render(

<div>

<TestClickComponent/>

<br/><br/>

<TestInputComponent/>

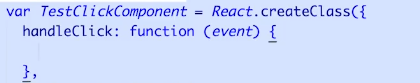
<div>,

document.getElementById('container'));

**给组件添加事件绑定（on驼峰式命名方式）**

添加点击事件，onClick={this.xxxxxxxx}

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



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

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



this.refs.tip

索引到的是react component而不是真实的dom节点

如何拿到dom节点：

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

新版本改为ReactDOM.findDOMNode

2.在dom里获得这个节点，ReactDOM.findDOMNode(this.refs.xxx)

有个坑大家注意，就是组件的首字母必须大写，不然不报错也不显示。