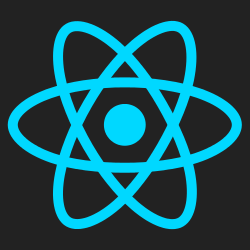
# 0. React介绍

## 0.1 什么是React？

React（有时称为React.js 或ReactJS）是一个为数据提供渲染HTML视图的开源JavaScript库。



React的logo是个原子，中间是原子核，旁边是3个电子的移动轨迹

它由FaceBook、Instagram和一个由个人开发者和企业组成的社群维护，现在国外比较流行的Facebook、Imgur、Airbnb、uber、Instagram，国内的美团、阿里、大搜车、去哪儿等都在使用React技术。

## 0.2 FB为什么推出React？

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](http://www.ruanyifeng.com/blog/2015/02/mvcmvp_mvvm.html" \t "_blank)，都不满意，就决定自己写一套，用来架设 [Instagram](https://instagram.com/" \t "_blank) 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](http://facebook.github.io/react/blog/2013/06/05/why-react.html" \t "_blank)了。

React是解决什么问题的，在官网可以找到这样一句话：

We built React to solve one problem: building large applications with data that changes over time.

## 数据变化会带来两个问题，大量的DOM操作（自动操作DOM），逻辑及其复杂（状态和内容对应起来 ）

## 0.3 React历史

React是由FaceBook的工程师Jordan Walke创建的，是受到php的HTML组件库XHP影响，React在11年时，刚开始是部署在FaceBook的newsfeed；

随后在12年时部署于Instagram，于13年5月在JSConf US宣布开源

14年成为facebook第一个在Github上达到1万star的旗舰开源项目，

15年3月在JSConf，FaceBook发布了React Native，可以使用React来构建nativeApp，并提出自己的理念“learn once，write anywhere”。

16年4月，发布V15正式版本，但是依然不够稳定，这毕竟是最新的技术。

## 0.4 React特点

声明式设计：采用声明范式，可以轻松描述应用

高效：通过对DOM的模拟，最大限度减少与DOM的交互

灵活：可以方便的搭配其它库来使用

JSX：是js语法的扩展

组件：构建组件，方便复用

单向相应的数据流

## 0.5 如何学习React

中文社区：<http://reactjs.cn/>

gitbook：<https://www.gitbook.com/book/hulufei/react-tutorial/details>

菜鸟教程：<http://www.runoob.com/react/react-tutorial.html>

书籍：React引领未来的用户界面开发框架（资料中有pdf电子书）

## 0.6 代价

在你选择之前**需要再考虑一下**：

①一开始 React 会极大地减慢你的开发。理解props、state以及组件通信如何工作并不是很简单，并且文档信息错综复杂。理论上，这将会被克服，你的整个团队都上道之后，开发速度上就会有一个很大的提升。

②React 不支持 IE8 以下的任何浏览器，以后也绝不会。

③如果你的应用/站点不需要频繁的动态页面更新，你可能为了很小的功能而编写大量的代码。

④你会改造很多轮子。React 很年轻，并且因为没有权威的方式来处理事件、组件通信，你必须从零开始创建大量的组件库。你的应用是否有下拉菜单，可调整大小的窗口，或者 lightbox？你同样必须从零开始写这些。

# 1.认识React

## 1.1 概述

### 1.1.1核心思想

React的核心思想是：封装组件。

各个组件维护自己的状态和UI，当状态改变，自动重新绘制整个组件。

通过分割组件的方式，你能够控制页面的复杂性，通过组件的方式去封装复杂的页面或某个功能区块。 并且，组件是可以被复用的。你可以将这个过程想象为：用乐高积木去拼装不同的物体。 因而，我们称这种编程方式称为**组件驱动开发**。

### 1.1.2 核心概念

React中的核心概念：

概念1：组件

概念2：JSX

JavaScriptXml，不是新语言，也没有改变js的语法，只是对js的扩展，

使用React，建议使用JSX语法，原生js也可以，但是由于JSX在定义类似HTML这种树形结构时，十分简单明了，所欲推荐使用JSX语法。、

概念3：Virtual DOM

如果真的这样大面积的操作 DOM，性能会是一个很大的问题，所以 React 实现了一个虚拟 DOM，组件 DOM 结构就是映射到这个虚拟 DOM 上，React 在这个虚拟 DOM 上实现了一个 diff 算法，当要更新组件的时候，会通过 diff 寻找到要变更的 DOM 节点，再把这个修改更新到浏览器实际的 DOM 节点上，所以实际上不是真的渲染整个 DOM 树。这个虚拟 DOM 是一个纯粹的 JS 数据结构，所以性能会比原生 DOM 快很多。

前面提到 virtual DOM 和真实的 DOM 有着不用的语义, 但同时也有明显不同的 API。

DOM 树上的节点被称为元素, 而 virtual DOM 是完全不同的抽象, 叫做 components。

component 的使用在 React 里极为重要, 因为 components 的存在让计算 DOM diff 更高效

举例：

比如说，现在你的list是这样，  
<ul>  
<li>0</li>  
<li>1</li>  
<li>2</li>  
<li>3</li>  
</ul>  
你想把它变成这样  
<ul>  
<li>6</li>  
<li>7</li>  
<li>8</li>  
<li>9</li>  
<li>10</li>  
</ul>  
通常的操作是什么？  
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。  
而React会把这两个做一下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，比9次Element操作快多了！

概念4：Data Flow

单向数据绑定。是指数据更新后会自动渲染到页面，避免在业务中频繁的DOM操作。

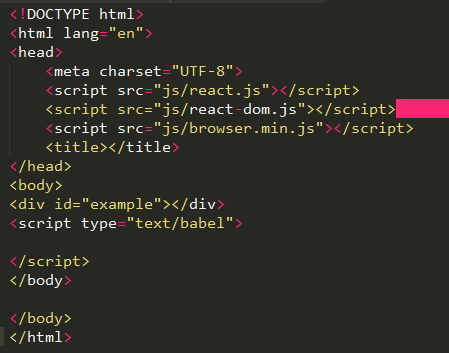
# 2. 搭建React开发环境

## 2.1 依赖

依赖三个库： 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，

## 2.2 创建模板文件

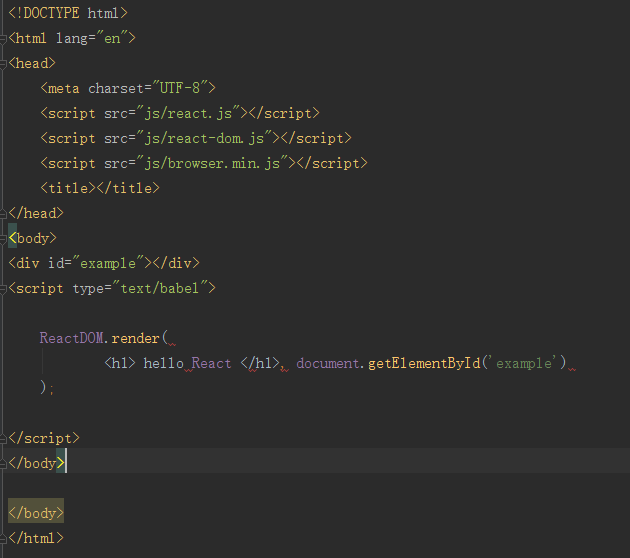


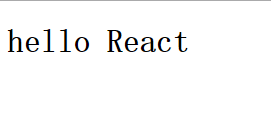
上面代码有两个地方需要注意。首先，最后一个 <script> 标签的 type 属性为 text/babel 。这是因为 React 独有的 JSX 语法，跟 JavaScript 不兼容。凡是使用 JSX 的地方，都要加上 type="text/babel"

# 3. Hello React

## 3.1. 开始

创建1.html





## 3.2. render方法介绍

ReactDOM.render(

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

document.getElementById('example')

);

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

# 4. JSX语法

## 4.1 基本语法规则

遇到HTML标签（以<开头），就用HTML来解析；遇到代码块（以{开头）就用js来解析

JSX允许使用html语法来创建js对象，拿创建a标签为例，如果使用原生js，是这样的：

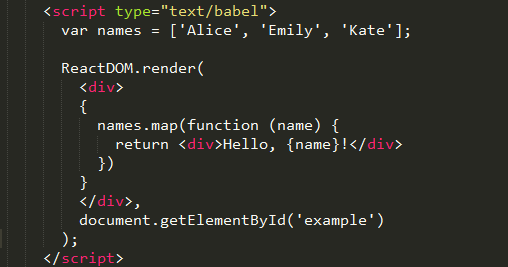
React.createElement('a', {href: 'https://facebook.github.io/react/'}, 'Hello!')

当使用jsx语法时，就变成了这样：

<a href="https://facebook.github.io/react/">Hello!</a>

JSX的基本语法规则：

接下来，创建2.html，验证上述语法。



细心的同学会发现，上边的代码运行时，会有个警告，是因为：

React为了方便DOM渲染，对于每个东西产生时都希望有一个单独的key，我们上边没有设置，所以会有警告，这里先不管它。

## 4.2 JavaScript表达式

我们可以在JSX中使用JavaScript表达式，表达式写在{}中，比如说：创建3.html

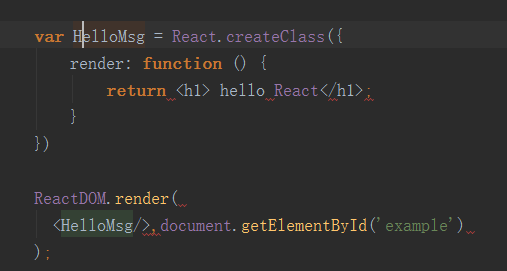


# 5 组件

## 5.1 创建并使用组件

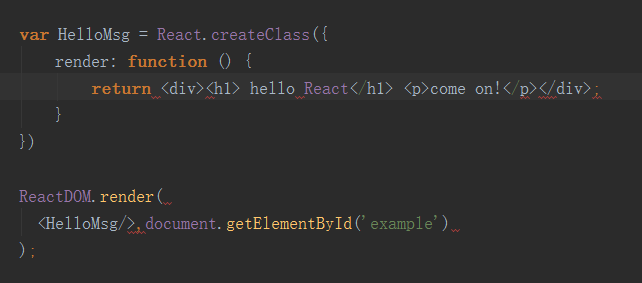
React 允许将代码封装成组件（component），然后像插入普通 HTML 标签一样，在网页中插入这个组件。React.createClass 方法就用于生成一个组件类

创建4.html

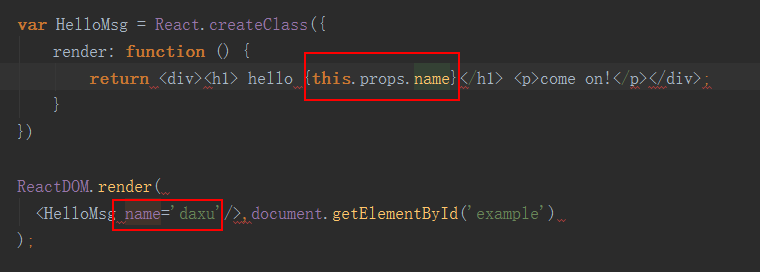


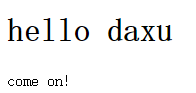
上面代码中，变量 HelloMsg 就是一个组件类。模板插入 < HelloMsg  /> 时，会自动生成 HelloMsg 的一个实例（下文的"组件"都指组件类的实例）。所有组件类都必须有自己的 render 方法，用于输出组件。

注意，组件类的第一个字母必须大写，否则显示会有问题，比如HelloMessage不能写成helloMessage。另外，组件类只能包含一个顶层标签，否则也会有问题。



在上述代码上稍作修改：



组件的用法与原生的 HTML 标签完全一致，可以任意加入属性，比如 <HelloMsg name="daxu"> ，就是 HelloMsg 组件加入一个 name 属性，值为 daxu。组件的属性可以在组件类的 this.props 对象上获取，比如 name 属性就可以通过 this.props.name 读取。上面代码的运行结果如下。

## 5.2 认识props

## 5.3 常用方法

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](https://facebook.github.io/react/docs/top-level-api.html" \l "react.children" \t "_blank) 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

创建5.html

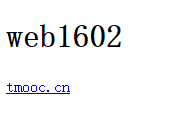


## 5.3 复合组件

我们可以通过创建多个组件来合成一个组件，即把组件的不同功能点进行分离。

以下实例我们实现了输出网站名字和网址的组件：

创建6.html







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