爱创课堂前端培训

# ReactJS

第1天课堂笔记（本课程共4天）

讲师：张容铭

日期：2018年6月26日

张容铭老师

微博：@张容铭\_YYQH

QQ : 286031482

E-mail : yuye\_qinghe@qq.com

爱创课堂官网 ：[www.icketang.com](http://www.icketang.com)

目录

[名称 1](#_Toc5012)

[一、搭建环境 3](#_Toc15833)

[1.1安装node 3](#_Toc5302)

[1.2安装webpack 3](#_Toc5452)

[1.3安装webapck-dev-server 3](#_Toc12660)

[1.4淘宝镜像 3](#_Toc15087)

[1.5手动安装 3](#_Toc7577)

[二、React 4](#_Toc28024)

[2.1 github 4](#_Toc30388)

[2.2 webpack 5](#_Toc3880)

[2.3创建虚拟DOM 6](#_Toc3395)

[2.4渲染虚拟DOM 7](#_Toc20667)

[三、jsx语法 7](#_Toc20581)

[四、ES6 8](#_Toc10468)

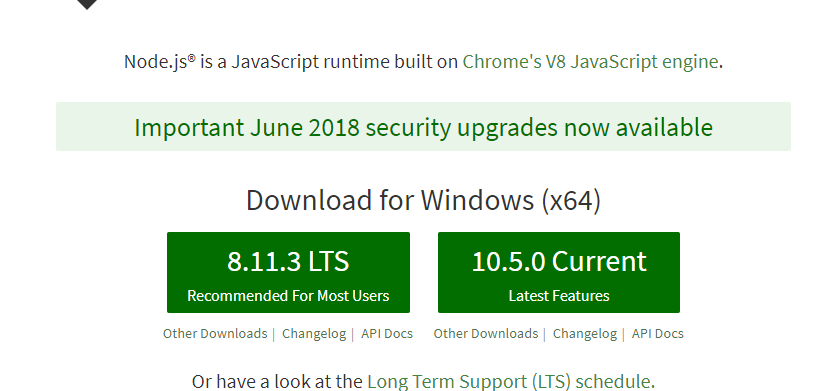
[五、属性 8](#_Toc14310)

[六、虚拟DOM属性 9](#_Toc9363)

# 一、搭建环境

## 1.1安装node

https://nodejs.org/en/



打开cmd

输入 node -v 查看node版本号

输入 npm -v 查看npm版本号

## 1.2安装webpack

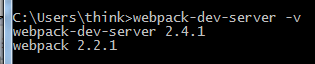
npm install -g webpack@3.5

全局安装，为了提供webpack指令



## 1.3安装webapck-dev-server

npm install -g webpack-dev-server

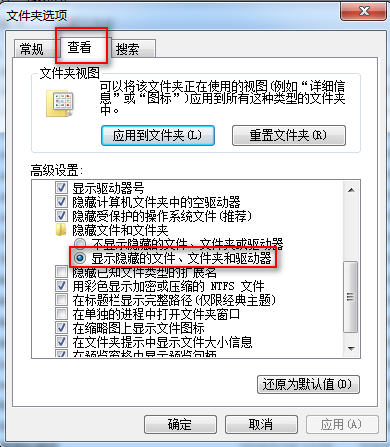
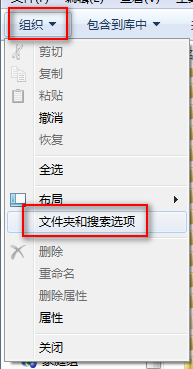


## 1.4淘宝镜像



## 1.5手动安装

显示隐藏文件



全局指令目录

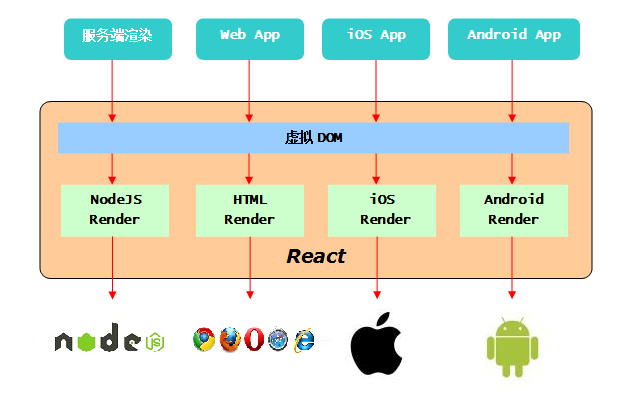
C:\Users\think\AppData\Roaming\npm

全局模块安装目录

C:\Users\think\AppData\Roaming\npm\node\_modules

# 二、React

三个特点：虚拟DOM， 组件开发，多端适配



## 2.1 github

https://github.com/facebook/react

可以通过bower下载react

安装bower

npm install bower -g

获取react

bower install react

可以通过npm安装react包

npm install react

可以通过npm安装react-dom包

npm install react-dom

本地安装有三种方式

npm install react

npm link react

npm install react --save-dev

## 2.2 webpack

入口文件 webpack.config.js 我们可以在里面，基于commonjs规范定义配置

AMD ： requirejs (module transport) deinfie(id, [], fn()) return export, module.export

CMD : seajs

Commonjs

ES Module

|  |
| --- |
| 1. // 基于comonjs规范，定义配置 2. module.exports = { 3. // 通过entry定义入口文件 4. // entry: './es/02.es', 5. // entry: './js/main.js', 6. entry: { 7. // key表示发布的文件名称 8. '02': './es/02.es', 9. '04': './es/04.es', 10. '05': './esx/05.esx', 11. '06': './es/06.es', 12. '07': './jsx/07.jsx', 13. '08': './jsx/08.jsx', 14. '09': './jsx/09.jsx' 15. }, 16. // mode: 'production', 17. // 发布的文件 18. output: { 19. // 文件名称，可以包含路径，也可以通过path属性定义路径 20. // filename: './pack/02.js' 21. // 可以通过[name]表示发布的文件名称 22. filename: './pack/[name].js' 23. }, 24. // 编译ES6的语法 25. module: { 26. // 编译文件要定义加载机（器） 27. rules: [ 28. // 每一个成员代表一个加载机 29. { 30. // 定义文件规则 31. test: /\.es$/, 32. // 定义加载机 33. loader: 'babel-loader?presets[]=es2015' 34. // loader: 'babel-loader?presets[]=es2015' 35. // loader: 'babel-loader', 36. // query: { 37. // presets: ['es2015'] 38. // } 39. }, 40. // jsx语法加载机 41. { 42. // 定义文件规则 43. test: /\.esx$/, 44. // 加载机 45. // loader: 'babel-loader', 46. loader: 'babel-loader', 47. query: { 48. presets: ['react', 'es2015'] 49. } 50. }, 51. // jsx语法加载机 52. { 53. // 定义文件规则 54. test: /\.jsx$/, 55. // 加载机 56. // loader: 'babel-loader', 57. loader: 'babel-loader', 58. query: { 59. presets: ['react', 'es2015'] 60. } 61. } 62. ] 63. } 64. } |

React

从16版本开始，已经完全将与DOM相关的属性方法剥离了，只保留与组件，虚拟DOM相关的属性方法

## 2.3创建虚拟DOM

通过React.createElement(虚拟DOM名称|组件类, 属性对象, ... 表示子虚拟DOM元素)

文本节点可以直接写

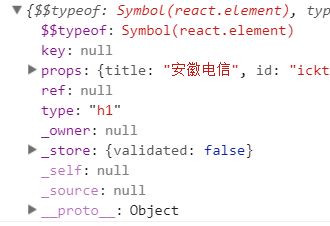
虚拟DOM

key表示虚拟DOM id

props 所有的属性（子虚拟DOM也会添加在里面）

ref 获取对真实DOM的引用

type 表示虚拟DOM类型



## 2.4渲染虚拟DOM

渲染库react-dom

提供render方法，可以将虚拟DOM渲染到页面中

render(虚拟DOM, 真实的DOM， fn)

|  |
| --- |
| 1. // 基于ES Module规范使用react 2. // import React from "react"; 3. // 可以解构createElement方法 4. // let { createElement } = React; 5. // 引入的之后即解构 6. import { createElement } from 'react'; 7. // 引入渲染库 8. import { render } from 'react-dom'; 9. let h1 = createElement('h1', { 10. // 添加属性 11. title: '安徽电信', 12. id: 'ickt' 13. }, '安徽电信培训') 14. // 渲染虚拟DOM 15. render(h1, document.getElementById('app')) |

# 三、jsx语法

我们开发中，创建虚拟DOM太麻烦了。不如XHTML创建元素方便

如果我们能够像创建xhtml元素一样创建虚拟DOM，开发就更加的方便了，所以React团队提供jsx语法：

像xhtml语法创建DOM元素一样，创建虚拟DOM

在XHTML中创建div， <div></div>

在jsx语法中创建div， <div></div>

在XHTML中创建input, <input type=”text” />

在jsx语法中创建input, <input type=”text” />

我们通过babel-preset-react插件编译jsx语法

为了区别jsx语法，react会在脚本文件后面添加x, 如js文件改成jsx, es改成esx， ts改成tsx

|  |
| --- |
| 1. // 引入的之后即解构 2. import React, { Component, createElement } from 'react'; 3. // 引入渲染库 4. import { render } from 'react-dom'; 5. // 定义组件 6. class Nav extends Component { 7. // 通过render方法渲染组件的虚拟DOM 8. render() { 9. // 返回值就是渲染的虚拟DOM 10. return ( 11. <ul> 12. <li>教你理财</li> 13. <li>天猫购物季1</li> 14. <li>热门房源</li> 15. </ul> 16. ) 17. } 18. } 19. // 将组件转换成虚拟DOM 20. let nav = <Nav></Nav>; 21. // 渲染虚拟DOM 22. render(nav, app) |

# 四、ES6

类：ES6中的类是一个安全类

js中常见继承：构造函数式，类式继承，组合式继承，寄生式继承，寄生组合式继承，原子继承，多继承，静态继承，特性继承，特性寄生式继承，构造函数式继承变形，特性寄生组合式继承...

继承：特性寄生组合式继承

# 五、属性

React为了让组件展示的样式不同，为组件定义了属性，所以组件的属性使组件展示出多样性

组件定义属性，跟虚拟DOM定义属性一样，虚拟DOM定义属性跟真实DOM定义属性是一样的

所以组件定义属性的语法跟真实的DOM定义属性的语法是一样的，直接在jsx语法中，添加属性

我们在组件中，可以通过this.props获取属性数据

插值语法

jsx语法不是js语法，所以想在jsx语法中，使用js变量，我们要使用插值语法 {}

注意：jsx语法不是js语法，所以不能直接在jsx语法中，写js注释

想写js注释，要提供js环境，我们可以通过{}实现

注意：单行注释不要注释掉插值符号的闭合符号 }

所以工作中，建议使用多行注释

react为虚拟DOM提供了一个key的非元素属性，用来设置元素的id，

key的属性值是唯一的，在循环中索引值是唯一的，因此我们可以设置索引值

默认属性

组件类的静态属性中，定义默认属性数据（在组件外部定义）

|  |
| --- |
| 1. // 引入核心库和渲染库 2. import React, { Component } from 'react'; 3. import { render } from 'react-dom'; 4. // 定义组件 5. class Nav extends Component { 6. // 渲染li的方法 7. renderList() { 8. // return [<li>{this.props.data[0]}</li>, 9. // <li>{this.props.data[1]}</li>, 10. // <li>{this.props.data[2]}</li>]; 11. // 根据map方法，将data数据映射成虚拟DOM数组 12. return this.props.data.map(function(item, index) { 13. // 返回虚拟DOM 14. return <li key={index}>{item + '111'}</li> 15. }) 16. } 17. // 定义渲染的方法 18. render() { 19. // {[<li>{this.props.data[0]}</li>, 20. // <li>{this.props.data[1]}</li>, 21. // <li>{this.props.data[2]}</li>]} 22. console.log(this) 23. // 返回值是渲染的结果 24. return ( 25. <ul> 26. {// 单行注释 27. } 28. {/\* 29. \* 多行注释 30. \*/} 32. {/\*如果有一个方法，能更具数组，映射一个虚拟DOM数组，我们就不需要一边一边的写li了\*/} 33. {this.renderList()} 34. </ul> 35. ) 36. } 37. } 38. // 定义默认属性数据 39. Nav.defaultProps = { 40. data: ['默认数据'] 41. } 42. // 定义两组数据 43. let arr1 = ['新浪新闻', '腾讯新闻', '百度新闻']; 44. let arr2 = ['淘宝网', '苏宁易购', '今日特价']; 45. // 渲染组件 46. render(<Nav data={arr1}></Nav>, app1) 47. render(<Nav data={arr2} />, app2) 48. render(<Nav />, app2) |

# 六、虚拟DOM属性

组件只有一类属性数据

所有为组件添加的属性数据，都是自定义数据属性

虚拟DOM有四类属性数据

元素属性：title，id， type

非元素属性：key， dangerouslySetInnerHTML, ref

key 设置虚拟DOM的ID

dangerouslySetInnerHTML 以危险的方式设置元素内容

1 我们不能为虚拟DOM行内式的样式字符串，只能设置样式对象

2 设置的内容包含元素，不会渲染（会做html转义并输出文本）

...

我们想写行内式样式字符串，想渲染元素，我们可以使用dangerouslySetInnerHTML

属性值是对象，通过\_\_html设置内容

工作中，尽量不要使用该属性

ref 对真实的DOM反向引用

我们为元素设置该属性

通过实例化对象的refs属性即可获取该属性值对应的元素

特殊元素属性：className, htmlFor

自定义数据属性 data-id，abc

|  |
| --- |
| 1. // 引入核心库和渲染库 2. import React, { Component } from 'react'; 3. import { render } from 'react-dom'; 4. // 定义组件 5. class Demo extends Component { 6. // 定义渲染的方法 7. render() { 8. console.log(this) 9. // 样式对象 10. let divStyle = { 11. background: 'green' 12. } 13. // 设置内容 14. let html = '<a href="">安徽电信</a>' 15. let html2 = '<a style="color: pink" href="">安徽电信</a>' 16. // 返回值是渲染的结果 17. return ( 18. <div style={divStyle} className="red"> 19. <label htmlFor="username" style={{ 20. color: 'red' 21. }}>用户名</label> 22. {/\*获取元素\*/} 23. <input id="username" ref="inp" type="text"/> 24. <h1>{html}</h1> 25. <h2 dangerouslySetInnerHTML={{ 26. \_\_html: html2 27. }}></h2> 28. </div> 29. ) 30. } 31. // 组件创建完成 32. componentDidMount() { 33. console.log(this.refs.inp) 34. } 35. } 36. // 定义默认属性 37. Demo.defaultProps = { 38. color: 'red' 39. } 40. // 渲染组件 41. render(<Demo></Demo>, app) |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |