爱创课堂前端培训

# React 16

第1天课堂笔记（本课程共5天）

班级：北京前端训练营16期

讲师：张容铭

日期：2018年11月6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

[React 16 1](#_Toc31245)

[一、React 3](#_Toc28983)

[1.1 react介绍 3](#_Toc17493)

[1.1.1特点 3](#_Toc28443)

[1.1.2 Github 3](#_Toc22682)

[1.2体验react 3](#_Toc7193)

[1.2.1创建虚拟DOM 4](#_Toc22866)

[1.2.3 createElement 4](#_Toc13177)

[1.2.4虚拟DOM 4](#_Toc13509)

[1.2.5渲染库 4](#_Toc29195)

[1.3组件 5](#_Toc25957)

[1.4 jsx语法 6](#_Toc32194)

[下午复习 8](#_Toc16794)

[1.5插值 9](#_Toc25272)

[1.5.1注释 9](#_Toc16156)

[1.6属性 10](#_Toc6565)

[1.6.1自定义数据属性 10](#_Toc28193)

[1.6.2元素属性 10](#_Toc27352)

[1.6.3特殊元素属性 10](#_Toc21215)

[1.6.4非元素属性 11](#_Toc23091)

[1.6.5ref属性 11](#_Toc14386)

[1.6.6 dangerouslySetInnerHTML属性 11](#_Toc27570)

[1.6.7 key属性 11](#_Toc11370)

[1.7组件属性 12](#_Toc19848)

[1.7.1属性分类 13](#_Toc8586)

[1.7.2定义属性 13](#_Toc7135)

[1.7.4默认属性数据 13](#_Toc13441)

[1.7.4属性约束 13](#_Toc21422)

[1.8 DOM事件 15](#_Toc31519)

[1.8.1参数 15](#_Toc30032)

[1.8.2作用域 15](#_Toc30430)

[1.9状态 16](#_Toc7358)

[1.9.1无状态组件 16](#_Toc31409)

[1.9.2有状态组件 16](#_Toc17555)

[1.9.3初始化状态 17](#_Toc1135)

[1.9.4修改状态数据 17](#_Toc21064)

# 一、React

react是由facebook公司开发并推广的框架。

由于React的设计思想极其独特，性能出众，代码逻辑非常简单。被越来越多的人关注和使用，认为它将是未来Web开发的主流工具。

## 1.1 react介绍

### 1.1.1特点

虚拟DOM：

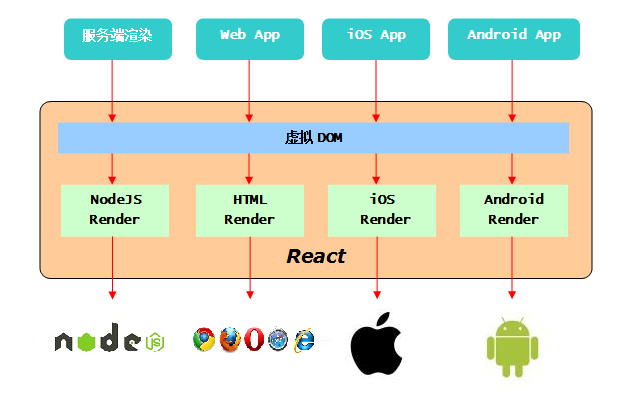
React通过对DOM的模拟，最大限度地减少与DOM的交互

组件化：

React采取组件化开发，极大限度的使组件得到复用，便于开发管理与维护

适用多端：

一处开发，多端适用，将颠覆整个互联网行业



### 1.1.2 Github

https://github.com/facebook/react

最新版本是16版本

我们基于ES6语法，学习该框架

# 1.2体验react

react为了实现多端适配，

通过核心库react.js创建虚拟DOM

在各个端，通过渲染库，渲染这些虚拟DOM，

例如，web端使用react-dom库

### 1.2.1创建虚拟DOM

我们引入react核心库，通过React.createElement方法，创建虚拟DOM

### 1.2.3 createElement

第一个参数表示虚拟DOM名称（组件名称）

第二个参数表示属性对象

从第三个参数开始，表示子虚拟DOM

如果是DOM元素，也要用createElement方法创建

如果是文本节点，可以直接书写

### 1.2.4虚拟DOM

key表示虚拟DOM的id

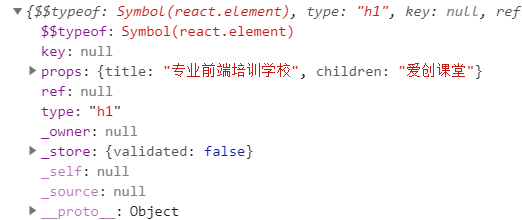
props 表示属性对象

元素的属性

元素的子元素

ref 引用虚拟DOM的属性名称

type 虚拟DOM类型



### 1.2.5渲染库

react-dom提供了一个render方法，可以渲染虚拟DOM

render

第一个参数表示虚拟DOM

第二个参数表示真实DOM

第三个参数表示渲染成功时候的回调函数

|  |
| --- |
| 1. import React, { createElement } from 'react'; 2. // 引入渲染库 3. import ReactDOM, { render } from 'react-dom'; 4. // 创建虚拟DOM，并且渲染到#app元素内 5. // console.log(React) 6. // console.log(ReactDOM, render) 7. // 创建虚拟DOM 8. var h1 = createElement('h1', { title: '专业前端培训学校' }, '爱创课堂') 9. // console.log(h1) 10. // 渲染虚拟DOM 11. render(h1, document.getElementById('app'), () => { 12. console.log('success') 13. }) |

## 1.3组件

虚拟DOM是对真实DOM的抽象，组件就是对虚拟DOM封装，使用组件就是为了复用这组虚拟DOM。

ES5语法中，创建组件用React.createClass方法

ES6语法中，建议面向对象编程，因此定义组件就是定义类，一定要继承组件基类React.Component

16版本中，不建议ES5开发，只能使用ES6语法开发

组件类中，提供另一个render方法，用来渲染虚拟DOM的。返回值就是虚拟DOM

组件是一个类，为了渲染组件，我们要将组件转成虚拟DOM，再渲染。

我们可以通过createElement方法，将组件转成虚拟DOM

再要通过ReactDOm.render方法渲染

注意

1 组件的render方法，渲染虚拟DOM，只能有且只有一个根虚拟DOM

2 组件是一个类，所以首字母要大写

|  |
| --- |
| 1. // 引入组件类 2. import React, { Component, createElement } from 'react'; 3. // 引入渲染库 4. import { render } from 'react-dom'; 5. // console.log(React) 6. // 定义组件类 7. class Demo extends Component { 8. // render方法，用来渲染输出虚拟DOM 9. render() { 10. return createElement( 11. 'ul', 12. null, 13. createElement('li', null, '教你理财'), 14. createElement('li', null, '天猫购物季'), 15. createElement('li', null, '热门房源') 16. ) 17. } 18. } 19. // 使用组件之前，要将组件转换成虚拟DOM 20. var demo = createElement(Demo) 21. // console.log(demo) 22. // 渲染组件 23. // render(demo, document.getElementById('app')) 24. // 每个具有id的元素，我们可以通过id名称，获取这个元素 25. render(demo, app) |

## 1.4 jsx语法

我们通过es6语法，创建虚拟DOM太麻烦了。所以React团队参考xhtml语法，实现了jsx语法。

jsx语法：像xhtml创建元素一样，创建虚拟DOM元素

例如

xhtml创建div元素： <div></div>

jsx创建div虚拟DOM： <div></div>

xhtml创建input元素： <input type="text" />

jsx语法创建input虚拟DOM： <input type="text" />

这样就简化了我们对虚拟DOM的创建，可是浏览器不支持，我们可以使用工程化的工具编译

编译ES6语法：babel-preset-es2015

jsx语法已经被纳入ES2016规范了，在es6语法之上的规范

所以我们编译jsx语法，也要使用babel编译器。

编译jsx语法，我们使用babel-preset-react插件

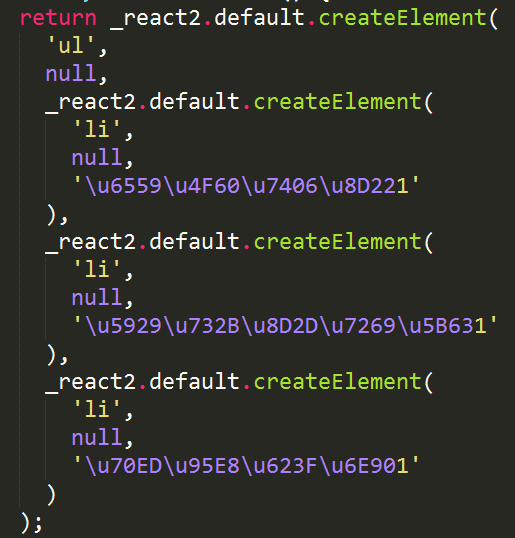
jsx语法，在es6语法之上使用，我们就要使用es2015和react插件了

presets: [‘es2015’, ‘react’]

jsx文件拓展名是.jsx

jsx语法，就是将<>语法，编译成createElement

使用的createElemnt方法，是通过react使用的，因此我们一定要引入React变量



|  |
| --- |
| 1. // 引入库 2. import React, { Component } from 'react'; 3. import { render } from 'react-dom'; 4. // 定义组件类 5. class Demo extends Component { 6. // 渲染 7. render() { 8. return ( 9. <ul> 10. <li>教你理财1</li> 11. <li>天猫购物季1</li> 12. <li>热门房源1</li> 13. </ul> 14. ) 15. } 16. } 17. // 将组件转化成虚拟DOM 18. // createElement(Demo) 19. // xhtml支持但标签，也支持双标签 20. var demo = <Demo></Demo> 21. // <Demo /> 22. render(demo, app) |

## 下午复习

react：虚拟DOM， 组件开发， 多端适配

核心库：react.js

React.createElement(name|component, props, ...children)

虚拟DOM key， type， ref， props

渲染库 react-dom

render(vdom, dom, fn)，用来渲染

jsx语法 <div><div>, <input type=”text” />

presets: [‘es2015’, ‘react’]

## 1.5插值

已经学习过的插值语法

ejs: <%=key%>

es6: ${key}

less: @{key}

scss: #{$key}

MVC: <%key%>

小程序: {{}} 伪js环境

vue: {{}}

这些插值语法，都有一个共同的特征，都是为了提供一个可执行的环境，使用表达式

react提供jsx语法，不是js语法，所以不能使用js表达式，为了在jsx语法中，嵌入js，我们就要使用插值语法

语法 {}

在jsx语法中，我们不仅仅可以在元素内容中使用插值语法，还可以在元素属性上使用插值语法

插值语法，提供了js环境，因此我们可以在插值语法中，使用js表达式

属性中使用插值语法

key=”value”

使用插值语法后，要省略引号

key={value}

### 1.5.1注释

jsx语法中，不能直接书写js注释

js注释要写在js环境中，而插值语法可以创建js环境，所以在jsx语法中，我们将js注释要写在插值语法中。

注意：单行注释不要注释掉插值闭合符号，所以要换行，因此工作中，多行注释更常用

|  |
| --- |
| 1. import React, { Component } from 'react'; 2. import { render } from 'react-dom'; 3. // 定义组件 4. class Header extends Component { 5. render() { 6. // 标题 7. var title = '爱创课堂'; 8. var date = new Date(); 9. // 返回值就是渲染的虚拟DOM 10. // 属性上也可以使用插值语法 11. return ( 12. <div> 13. {// 单行注释 14. } 15. {/\* 多行注释 16. \*/} 17. {/\*多行注释更常用\*/} 18. <div title={title}>{title + ' color'.toUpperCase()}</div> 19. <div>{date.getFullYear()}</div> 20. </div> 21. ) 22. } 23. } 24. // 渲染 25. render(<Header />, app) |

## 1.6属性

真实的DOM元素有属性，虚拟DOM是对真实的模拟，所以虚拟DOM也有属性，在react中，将虚拟DOM的属性分成四类：元素属性，特殊元素属性，非元素属性，自定义数据属性。

### 1.6.1自定义数据属性

自定义数据属性，就是用来存储数据，没有其他任何功能和作用。

跟DOM的自定义数据属性一样，通常以 data-开头，定义成其他的也可以

如 data-id, data-type, color, abc

### 1.6.2元素属性

这类属性在真实的DOM中存在，对DOM来说，有一定的功能和作用，

如title， src， href等等，

但是有一个是特殊的：style

style用来定义元素的行内式样式，属性值必须是对象

所以在jsx语法中，通过插值语法{}来提供js环境，

style={{

color: ‘red’,

backgroundColor: ‘green’

}}

外部的{}代表插值语法，内部的{}代表对象符号

样式属性名称默认使用驼峰式命名，通常css3样式前缀首字母大写

不仅仅在jsx语法中，属性是值对象，在js语法中，也是对象

### 1.6.3特殊元素属性

目前在DOM的所有属性中，有两个属性命中了关键字

class定义元素类的属性，命中了关键字class，所以要写成className

for为了让label与input级联的，命中了关键字for，所以要写成htmlFor

|  |
| --- |
| 1. import React, { Component, createElement } from 'react'; 2. import { render } from 'react-dom'; 3. // 定义组件 4. class Header extends Component { 5. render() { 6. // 标题 7. var title = '爱创课堂'; 8. return ( 9. <div> 10. <div style={{ 11. color: 'red', 12. backgroundColor: 'green' 13. }}>{title}</div> 14. <h1 className="demo">{title}</h1> 15. <label htmlFor="username">用户名</label> 16. <input id="username" type="text"/> 17. </div> 18. ) 19. } 20. } 21. // 渲染 22. render(<Header />, app) |

### 1.6.4非元素属性

元素本身不具备的属性，但是由react为元素拓展的，让元素具有某项功能的属性

### 1.6.5ref属性

我们为元素添加ref属性，此时我们就可以在组件中，通过this.refs获取这个元素，

获取的源生DOM元素，因此要用源生的DOM API操作

通常不建议通过ref属性操作元素，

组件中有一个方法，叫componentDidMount，表示组件创建完成，我们可以在方法内部来查看组件实例化对象

### 1.6.6 dangerouslySetInnerHTML属性

以一种危险的方式，设置元素的内容，

例如：

行内式样式不合法，

渲染元素

以上设置方式都是不合法的

属性值是对象，其中\_\_html用来设置内容

工作中，慎用，是通过设置元素的innerHTML属性实现的，因此不会被react优化

### 1.6.7 key属性

为元素设置react-id的，让虚拟DOM与页面中的DOM一一对应

通常在渲染列表的时候使用。

|  |
| --- |
| 1. import React, { Component, createElement } from 'react'; 2. import { render } from 'react-dom'; 3. // 定义一份数据 4. var data = ['爱奇艺高清', '优酷网', '百度视频'] 5. // 定义组件 6. class Header extends Component { 7. // 定义属性方法 8. renderList() { 9. // 将上面的数组映射成下面的数组 10. // return data.map((item, index) => { 11. // return <li>{item}</li> 12. // }) 13. // 省略函数体 14. // 循环中，一定要设置key属性 15. return data.map((item, index) => <li key={index}>{item}</li>) 16. // return [ 17. // <li>教你理财</li>, 18. // <li>天猫购物季</li>, 19. // <li>热门房源</li> 20. // ] 21. } 22. render() { 23. // 标题 24. var title = '爱创课堂'; 25. var info = '<span style="color: red">hello</span>'; 26. return ( 27. <div> 28. {/\*设置ref属性，获取元素\*/} 29. <h1 ref="ickt">{title}</h1> 30. {/\*危险的方式设置内容\*/} 31. <h1 dangerouslySetInnerHTML={{ \_\_html: info }}></h1> 32. {/\*<ul> 33. {[ 34. <li>教你理财</li>, 35. <li>天猫购物季</li>, 36. <li>热门房源</li> 37. ]} 38. </ul>\*/} 39. {/\*将数组拿出去，此时显得很整洁\*/} 40. <ul>{this.renderList()}</ul> 41. </div> 42. ) 43. } 44. // 组件创建完成 45. componentDidMount() { 46. console.log(this, this.refs.ickt) 47. // 设置样式 48. this.refs.ickt.style.color = 'red' 49. } 50. } 51. // 渲染 52. render(<Header />, app) |

## 1.7组件属性

虚拟DOM跟真实DOM一样，定义属性的目的是为了让元素展示出多样式。

创建组件的目的是为了复用虚拟DOM，因此组件与虚拟DOM的行为是一致的，

为虚拟DOM添加属性，是为了让虚拟DOM展示出多样式

为组件天添加属性，是为了让组件展示出多样式

### 1.7.1属性分类

虚拟DOM有四类属性数据：元素属性，特殊元素属性，非元素属性，自定义数据属性

组件的属性数据只有一类，就是自定义数据属性。

### 1.7.2定义属性

为组件添加属性的语法跟为虚拟DOM添加属性的语法是一样的

虚拟DOM： <div id=”demo”></div>

组件： <Ickt id=”demo”></Ickt>

为组件添加的属性，我们可以在组件内部，通过this.props获取

这样我们就可以在组件内部使用属性数据

注意：在React中，我们不能在组件内部修改数据，对于组件来说，属性属于外部传递的数据，应该在外部维护（只能在外部修改）。

### 1.7.4默认属性数据

我们通过组件类的静态属性deaultProps，定义组件的默认属性数据

ES6中，有两种定义静态属性的方法

1 在类的内部，通过static关键字，定义get特性方法，来定义静态属性

2 在组件外部，直接为类通过点语法，添加静态属性

react建议我们使用第二种方式定义默认属性的数据

作用：没有传递数据的时候，使用默认数据，传递了属性数据，就使用传递的数据

### 1.7.4属性约束

react16版本之后，将属性约束模块拆分出来了，作为一个单独的模块存在，使用就要安装

npm install prop-types

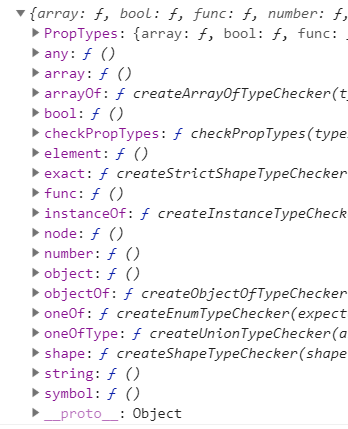
该模块提供了大量的数据类型，可以约束我们的属性数据，

跟默认属性数据一样，我们为组件定义propTypes静态属性，即可约束组件的属性

prop-types提供的类型可以约束属性的类型，并且每种类型下，都有一个isRequired方法，设置该方法，表示该属性是必填的

注意：

一旦定义了默认属性数据，此时属性的约束将失效。



|  |
| --- |
| 1. import React, { Component } from 'react'; 2. import { render } from 'react-dom'; 3. // 引入属性约束模块 4. import PropTypes from 'prop-types'; 5. // 定义组件 6. class Nav extends Component { 7. // static get color() { 8. // return 'red' 9. // } 10. // 创建列表的方法 11. createList() { 12. // 根据属性数据创建 13. return this.props.data.map((item, index) => <li key={index}>{item}</li>) 14. } 15. render() { 16. console.log(this) 17. return ( 18. <div> 19. {/\*创建列表\*/} 20. <ul>{this.createList()}</ul> 21. </div> 22. ) 23. } 24. } 25. // 在类的外部，定义静态属性 26. // Nav.num = 100; 27. // 定义静态属性数据 28. Nav.defaultProps = { 29. data: ['默认数据'] 30. } 31. // 约束性数据 32. Nav.propTypes = { 33. num: PropTypes.number.isRequired, 34. // num: PropTypes.bool 35. // data是数据，必填 36. data: PropTypes.array.isRequired 37. } 38. // 静态属性只有组件类能够使用 39. // console.log(Nav.color, Nav.num) 40. // 渲染 41. render(<Nav num={100} data={['教你理财', '天猫购物季', '热门房源']} />, app) 42. // 渲染第二组导航 43. render(<Nav num={0} data={['天猫', '当当网', '1号店']}></Nav>, app2) 44. // 渲染第三组导航 45. render(<Nav num={20}></Nav>, app3) |

## 1.8 DOM事件

在jsx语法中，绑定事件的语法跟xhtml中绑定事件的语法是相似的

xhtml中，绑定DOM事件：

<div onclick=”fn()”></div>

jsx语法中，绑定DOM事件

<div onClick={this.fn}></div>

注意：

1 事件名称首字母要大写

2 通过插值语法，绑定组件中的方法

3 方法不能执行（不能添加参数集合）

绑定的是一个定义的方法，而不是一个执行的方法

事件回调函数定义在组件类中

### 1.8.1参数

默认一个参数就是react封装后的事件对象（16版本中）

可以准确获取触发事件的目标对象，以及绑定事件的对象

react是通过事件委托模式实现的（减少事件数量，预言未来元素，避免内存外泄）

vue绑定的是源生的事件，事件对象是源生的是事件对象

jQuery也实现了事件委托，因此事件对象是jQuery封装的事件对象

### 1.8.2作用域

在ES5语法中，事件回调函数作用域中this就是组件实例化对象，时候不能改变的

在ES6语法中，事件回调函数作用域中this默认是undefined，我们可以是改变的

改变作用域中this指向有两种方式

第一种，使用ES5语法提供的bind方法，改变this指向组件实例化对象。

第二种，使用es6语法提供的箭头函数，定义时作用域，this指向组件实例化对象

改变this指向的同时，我们可以传递自定义参数

bind方法：自定义参数添加在事件对象的前面

箭头函数，事件对象可以写在任何位置

我们可以将this指向其他对象，但是react不建议

|  |
| --- |
| 1. import React, { Component } from 'react'; 2. import { render } from 'react-dom'; 3. // 定义对象 4. let obj = { 5. color: 'red' 6. } 7. // 定义组件 8. class Demo extends Component { 9. // 点击按钮事件回调函数 10. clickBtn(e) { 11. console.log(arguments, e.target, e.currentTarget, this) 12. } 13. render() { 14. return ( 15. <div> 16. {/\*绑定DOM事件\*/} 17. <button onClick={this.clickBtn}>按钮</button> 18. <button onClick={this.clickBtn.bind(this, 100, 'red', true)}>按钮1</button> 19. {/\*省略箭头函数的函数体\*/} 20. <button onClick={e => this.clickBtn(200, e, 'green', false)}>按钮2</button> 21. {/\*react不建议，将this指向组件实例化对象之外的对象\*/} 22. <button onClick={this.clickBtn.bind(obj)}>按钮3</button> 23. </div> 24. ) 25. } 26. } 27. // 渲染 28. render(<Demo></Demo>, app) |

## 1.9状态

状态跟属性一样，都是为组件存储数据的

属性维护的是组件外部的数据，数据在组件外部传递

状态维护的是组件内部的数据，数据在组件内部维护

根据组件是否具有状态数据，我们将组件分成两类：

一类是无状态组件

一类是有状态组件

### 1.9.1无状态组件

如果组件从创建开始，就是一成不变的，也就是组件不会因与用户交互产生数据，不会发送异步请求等等，这类组件我们称之为是无状态组件

目前为止我们学习的组件都是无状态组件

甚至无状态组件可以简写成一个函数

### 1.9.2有状态组件

如果组件创建之后，会与用户产生交互，产生数据，会发送异步请求，获取数据等等，此时我们就要在组件内部存储这些数据，他们就是状态数据，这类组件就是有状态组件

跟属性一样，我们可以通过this.state来访问这些数据。

### 1.9.3初始化状态

我们可以在构造函数中，初始化状态，

在组件类中，通过constructor定义构造函数，由于组件类继承Component组件基类，我们重写了构造函数，为了实现构造函数式继承，我们要使用super关键字继承，

super(props)

注意：不传递props，this.props还是undefined

此时构造函数有参数，就是属性数据对象，并且组件实例化对象中props属性（this.props）也表示属性数据对象

由于实现了构造函数式继承，因此他们没有区别了

我们可以为this.state赋值，实现对状态数据的初始化

注意：

我们可以在构造函数中访问属性数据，因此可以用属性数据为状态数据赋值，实现了外部的数据存储在组件内部维护（属性数据转换成状态数据）

### 1.9.4修改状态数据

我们可以通过this.setState方法，在组件中，修改状态数据

参数是对象

key表示状态数据名称

value表示状态数据

不论是属性数据改变了，还是状态数据改变了，组件都会重新执行render方法。渲染新的虚拟DOM。

|  |
| --- |
| 1. import React, { Component } from 'react'; 2. import { render } from 'react-dom'; 3. // 无状态组件，可以简写成函数 4. // const Demo = () => (<button>按钮</button>) 5. // 定义有状态组件 6. class Demo extends Component { 7. // 构造函数 8. constructor(props) { 9. // 构造函数继承 10. super(props); 11. // console.log(this.props, props) 12. // 定义状态参数 13. this.state = { 14. // 默认显示第一个 15. num: 0 16. } 17. } 18. // 渲染列表 19. createList() { 20. // 获取数据列表长度 21. let len = this.props.data.length; 22. // 根据属性数据，将视图创建出来 23. return this.props.data.map((item, index) => ( 24. <li key={index} style={{ 25. // 显示第num个 26. display: this.state.num % len === index ? 'block' : 'none' 27. }}>{this.createChildrenList(item)}</li> 28. )) 29. } 30. // 渲染字列表 31. createChildrenList(item) { 32. // 遍历参数数据，并返回视图结果 33. return item.map((childItem, index) => <span key={index}>{childItem}</span>) 34. } 35. // 改变显示的成员 36. change() { 37. // var num = this.state.num; 38. // // 修改状态数据 39. // this.setState({ 40. // num: num + 1 41. // }) 42. // 合并成一句话 43. this.setState({ 44. num: ++this.state.num 45. }) 46. } 47. // 渲染 48. render() { 49. return ( 50. <div> 51. <span onClick={e => this.change()}>换一换</span> 52. <ul>{this.createList()}</ul> 53. </div> 54. ) 55. } 56. } 57. // 渲染 58. render(<Demo data={[ 59. ['人民币中间价上调', '双11消费提示'], 60. ['货车与火车相撞', '石智勇破世界纪录'], 61. ['日本大型无人潜艇', '陈妍希拍婚纱照'] 62. ]}></Demo>, app) 63. render(<Demo data={[ 64. ['海上聚众黄赌毒'], 65. ['最严遛狗规定'], 66. ['苹果新机砍单'] 67. ]}></Demo>, app2) |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |