爱创课堂前端培训

# ReactJS第三期

课程：爱创课堂公开课

讲师：张容铭

日期：2016年12月17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)

[ReactJS第三期 1](#_Toc6393)

[复习 3](#_Toc7867)

[组件声明周期 4](#_Toc16931)

[创建期 4](#_Toc13076)

[存在期 6](#_Toc9012)

[销毁期 9](#_Toc19633)

# 复习

样式

组件中为虚拟DOM添加样式，跟真实的DOM元素是一样的

都是通过style属性为元素添加样式，虚拟DOM的style属性值只能是对象

如果css中的属性名称由多个单词构成，我们要转化成驼峰式命名

如果css中的属性名称包含浏览器前缀，首字母要大写

注释

在jsx语法中使用注释只能在插值符号中，并且是多行注释

事件

为虚拟DOM绑定事件，给真实的DOM元素绑定事件的方式是一样的，

on+事件名称（首字母要大写）= 回调函数

在虚拟DOM中绑定事件回调函数通常是组件实例化上的，通过this绑定

事件回调函数

三个参数

第一个是React封装的事件对象

第二个是reactid

第三个是源生的事件对象

作用域是组件实例化对象，因此可以通过this来修改或者获取组件实例化对象上的数据

状态

React根据组件状态形式分成两种

一种是无状态组件

组件一旦被创建后，就不会再改变，也就是组件是一成不变的

另外一种是有状态组件

组件内部会维护这一个状态，状态的改变会引发组件的更新

状态跟属性很像，都是用来存储数据的，并且发生发生改变时候，都会引发组件的更新，

属性通常是由外部传递的数据（在组件创建时候添加的一些属性数据）

状态通常是由内部维护，

设置初始化状态

getInitialState 必须有返回值，返回值是一个对象，这个对象就是初始化的状态

更新组件的状态

this.setState 更新组件的状态，参数对象的属性就是即将发生改变的状态属性名称

# 组件声明周期

React设计者将组件看成是一个有生命的个体，让组件赋予了生命，组件像一个生物

组件也有三大生命周期

创建期：在组件创建的时候出现的

存在期：在组件发生属性或者状态数据改变的时候的时期

销毁期：组件从页面中移除时候进入的时期

组件的每个周期又分成不同的阶段

## 创建期

创建期分成五大阶段

第一个阶段

getDefaultProps 获取默认属性

作用域是组件的构造函数

我们无法访问任何数据，因此该阶段执行完毕之后，组件才拥有了属性

第二个阶段

getInitialState 初始化状态

作用域是组件实例化对象

所以可以通过组件实例化对象访问到属性，但是这个阶段状态还没有构建出来

这个阶段执行完毕，状态才被构建

因此在这个阶段我们有机会用属性为初始化状态赋值

第三个阶段

componentWillMount 组件即将被构建

作用域是组件实例化对象

我们可以访问到属性，可以访问到状态

不要再这个阶段更改属性或者状态

我们无法访问到dom元素

第四个阶段

Render 组件渲染输出虚拟DOM

作用域是组件实例化对象

获取不到虚拟DOM元素，但是可以访问属性，或者状态

返回值就是虚拟DOM树，因此虚拟DOM构建时刻就是第四个阶段执行完毕之后

第五个阶段

componentDidMount 组件已经构建完成

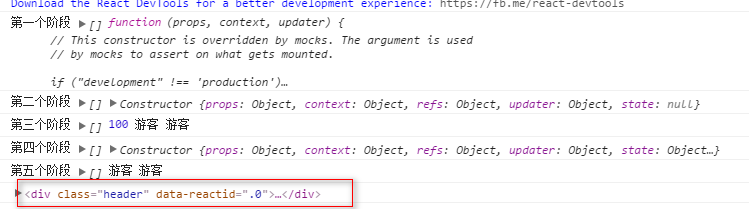
作用域是组件实例化对象

可以使用虚拟DOM，可以使用属性，可以使用状态，

我们通常可以在这个方法中，使用其他的库操作dom，或者绑定事件，发送请求等等

这个阶段一旦执行完毕，创建期就结束

这五个阶段都没有参数，因此想使用数据只能通过组件实例化对象



在react的组件中访问dom用ReactDOM.findDOMNode()

以前版本中，this.getDOMNode访问

返回值就是获取的虚拟DOM对应的真实DOM元素

通常我们用ReactDOM.findDOMNode(this)，获取组件容器元素

|  |
| --- |
| 1. // 创建头部组件 2. var Header = React.createClass({ 3. // 第一个阶段 4. getDefaultProps: function() { 5. console.log('第一个阶段', arguments, this) 6. return { 7. userName: '游客' 8. } 9. }, 10. // 第二个阶段 11. getInitialState: function () { 12. console.log('第二个阶段', arguments, this) 13. return { 14. num: 100, 15. // 用属性为状态赋值 16. userName: this.props.userName 17. } 18. }, 19. // 第三个阶段 20. componentWillMount: function() { 21. console.log('第三个阶段', arguments, this.state.num, this.props.userName, this.state.userName) 22. // 查看是否可以访问dom元素 23. // console.log(ReactDOM.findDOMNode(this)) 24. }, 25. // 第四个阶段 26. render: function() { 27. console.log('第四个阶段', arguments, this) 28. // 查看是否可以访问dom元素 29. // console.log(ReactDOM.findDOMNode(this)) 30. // 虚拟DOM必须通过return输出 31. return ( 32. <div className="header"> 33. <div className="container"> 34. <span>{this.state.userName}</span> 35. </div> 36. </div> 37. ) 38. }, 39. // 第五个阶段 40. componentDidMount: function () { 41. // 缓存组件实例化对象 42. var me = this; 43. console.log('第五个阶段', arguments, this.props.userName, this.state.userName) 44. // 查看是否可以访问dom元素 45. console.log(ReactDOM.findDOMNode(this)) 46. // 获取异步请求的数据 47. // $.get('data/getUser.json', function (res) { 48. // // 更新状态 49. // me.setState(res) 50. // }) 51. $.get('data/getUser.json', function (res) { 52. // 更新状态 53. // console.log(this) 54. this.setState(res) 55. }.bind(this)) 56. } 57. }) 58. // 将组件渲染到页面中 59. ReactDOM.render(<Header />, document.getElementById('app')) |

## 存在期

一旦状态或者属性发生改变，组件就进入了存在期

存在期分成五个阶段

第一个阶段

componentWillReceiveProps 组件即将接收新的属性

状态发生改变的时候，不会触发这个阶段，这个阶段的触发只能由属性的改变实现

第一个参数表示新的属性

我们可以在这个阶段将属性数据，转化成状态数据

第二个阶段

shouldComponentUpdate 组件是否应该更新

第一个参数表示新的属性

第二个参数表示新的状态

必须有返回值

True可以更新

False表示不可以更新

第三个阶段

componentWillUpdate 组件即将更新

第一个参数表示新的属性

第二个参数表示新的状态

这个阶段更新了属性和状态

第四个阶段

render 渲染输出虚拟DOm的

这个阶段更新虚拟DOM，因此前面的方法访问的都是旧的虚拟DOM，只有之后一个阶段访问的才是新的虚拟DOM

第五个阶段

componentDidUpdate 组件已经更新完毕

第一个参数表示上一个属性

第二个参数表示上一个状态

因此我们可以在这个方法中修改虚拟dom

前三个方法的参数都是新的属性，新的状态的原因是组件实例化对象中的属性和状态还没有被更新，因此可以通过参数访问新的属性，新的状态。可以通过作用域（组件实例化对象）访问旧的属性，旧的状态

最后一个方法参数是旧的属性旧的状态，原因是组件实例化对象的属性和状态已经更新了，因此可以通过作用域访问。旧的属性，旧的状态为了方便访问，作为参数传递

属性数据是由外部提供的，因此更新属性数据要在父组件中实现

父组件与子组件

虚拟DOM跟真实的DOM元素的行为表现是一致的，例如，真实的dom可以添加属性，添加样式，绑定事件，虚拟DOM同样可以添加属性，添加样式，绑定事件，组件是对一组虚拟DOM的封装，因此组件与虚拟DOM的行为表现也是一致的，可以添加属性等（绑定的事件，添加样式等都是自定义属性数据）

真实DOM元素可以拥有子DOM元素，因此虚拟DOm元素也可以拥有子虚拟DOM元素，所以组件也可以创建在一个组件的内部，作为这个组件的子组件，

例如

Slider组件

<div>

<GoBack />

</div>

此时

GoBack组件就是Slider组件的子组件

Slider组件就是GoBack的组件的父组件

在父组件中，可以为子组件添加属性，这些属性数据就是传递给子组件的数据（简单类型（变量，字符串），状态，属性，方法）

如果字符串的属性数据值发生改变，子组件将进入存在期的五个阶段

在react中使用侵入插件，一定要注意react组件的生命周期，所以我们尽量少用侵入式插件

|  |
| --- |
| 1. // 创建一个新闻列表 2. var News = React.createClass({ 3. // 初始化状态 4. getInitialState: function() { 5. return { 6. // 子组件有属性数据，因此我们可以通过属性数据更新状态数据 7. list: this.props.news 8. } 9. }, 10. // 创建列表 11. createList: function () { 12. // 根据状态数据更新列表 13. return this.state.list.map(function (value, index) { 14. return (<li key={index}>{value}</li>) 15. }) 16. }, 17. // 更新列表数据 18. updateNews: function () { 19. // 请求数据更新状态 20. $.get('data/news1.json', function (res) { 21. // 更新状态 22. this.setState(res) 23. }.bind(this)) 24. }, 25. // 存在期第一个阶段 26. componentWillReceiveProps: function(nextProps) { 27. console.log('第一个阶段', nextProps, this) 28. // 我们可以在这里更新状态，新的属性数据更新状态 29. this.setState({ 30. // 状态保存的是list数据，属性保存的是news数据 31. list: nextProps.news 32. }) 33. }, 34. // 存在期第二个阶段 35. shouldComponentUpdate: function(nextProps, nextState) { 36. console.log('第二个阶段', nextProps, nextState, this) 37. console.log(ReactDOM.findDOMNode(this).getElementsByTagName('li')[0].innerHTML) 38. // 通过返回值判断组件是否应该更新 39. return true 40. // return false 41. }, 42. // 存在期第三个阶段 43. componentWillUpdate: function(nextProps, nextState) { 44. console.log('第三个阶段', nextProps, nextState, this) 45. }, 46. // 存在期第四个阶段 47. render: function () { 48. console.log('第四个阶段', this) 49. return ( 50. <div className="news"> 51. {/\*<span onClick={this.updateNews}>换一换</span>\*/} 52. <ul>{this.createList()}</ul> 53. </div> 54. ) 55. }, 56. // 存在期第五个阶段 57. componentDidUpdate: function(prevProps, prevState) { 58. console.log('第五个阶段', prevProps, prevState, this) 59. console.log(ReactDOM.findDOMNode(this).getElementsByTagName('li')[0].innerHTML) 60. // 解决侵入式插件更新问题，只能在存在以及创建期分别执行 61. $('li').css('background', 'red') 62. }, 63. // 我们想用jquery将页面中的li元素背景设置成红色 64. componentDidMount: function() { 65. // 用jquery修改li背景 66. $('li').css('background', 'red') 67. } 68. }) 69. // 定义父组件 70. var Parent = React.createClass({ 71. // 初始状态数据 72. getInitialState: function() { 73. return { 74. news: ['朝鲜柳京饭店竣工', '玄彬姜素拉热恋', '黄轩被曝恋情'] 75. } 76. }, 77. // 更新新闻数据 78. updateNewsDate: function () { 79. // 发送异步请求，请求数据 80. $.get('data/news2.json', function (res) { 81. // 请求回来的数据更新状态 82. this.setState(res) 83. }.bind(this)) 84. }, 85. render: function () { 86. return ( 87. <div> 88. {/\*点击换一换，发送异步请求，获取数据，更新子组件\*/} 89. <span onClick={this.updateNewsDate}>换一换</span> 90. <News news={this.state.news}></News> 91. </div> 92. ) 93. } 94. }) 95. // 将组件渲染到页面中 96. ReactDOM.render(<Parent />, app) |

## 销毁期

当组件从页面中删除的时候，组件会进入销毁期

只有一个阶段

componentWillUnmount

没有参数

作用域是组件实例化对象

这个方法是我们最后一次可以访问组件的方法

|  |
| --- |
| 1. // 创建一个Demo组件 2. var Demo = React.createClass({ 3. render: function () { 4. return ( 5. <div> 6. <h1>爱创课堂 http://www.icketang.com</h1> 7. </div> 8. ) 9. }, 10. // 销毁期只有一个阶段 11. componentWillUnmount: function() { 12. console.log(this, arguments, 111) 13. } 14. }) 15. // 将组件渲染出来 16. ReactDOM.render(<Demo />, app) 17. // 重新渲染虚拟DOM 18. setTimeout(function () { 19. // 重新渲染页面 20. ReactDOM.render(<h1>爱创课堂公开课</h1>, app) 21. }, 3000) |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |