博学谷
www.boxuegu.com

第一章 React基础加强

一样的在线教育，不一样的教学品质

博学谷
www.boxuegu.com

目录 Contents

- 脚手架应用
- JSX应用与原理分析
- React组件基本应用
- React组件高级应用
- React组件应用案例

1. 脚手架应用

博学谷
www.boxuegu.com

React脚手架意义

- 脚手架是开发现代Web应用的必备
- 充分利用 Webpack，Babel，ESLint等工具辅助项目开发
- 零配置，无需手动配置繁琐的工具即可使用、
- 关注业务，而不是工具配置

1. 脚手架应用

博学谷
www.boxuegu.com

使用React脚手架初始化项目

初始化项目

命令： npx create-react-app my-pro

- npx 目的：提升包内提供的命令行工具的使用体验
- **原来**：先安装脚手架包，再使用这个包中提供的命令
- **现在**：无需安装脚手架包，就可以直接使用这个包提供的命令
- create-react-app 这个是脚手架名称，不能随意更改
- my-pro 自己定义的项目名称

启动项目

在项目根目录执行命令： npm start

博学谷
www.boxuegu.com

目录 Contents

- 脚手架应用
- JSX应用与原理分析
- React组件基本应用
- React组件高级应用
- React组件应用案例

2. JSX应用与原理分析

博学谷
www.boxuegu.com

JSX概述

JSX是JavaScript XML 的简写，表示在JavaScript代码中写HTML格式的代码

优势：声明式语法更加直观，与HTML结构相同，降低了学习成本，提升开发效率

2. JSX应用与原理分析

博学谷
www.boxuegu.com

为什么在脚手架中可以使用JSX语法?

- JSX 不是标准的ECMAScript语法，它是ECMAScript的语法拓展
- 需要使用babel编译处理后，才能在浏览器环境中使用
- create-react-app脚手架中已经默认有该配置，无需手动配置
- 编译JSX语法的包： @bable/preset-reac

2. JSX应用与原理分析

博学谷
www.boxuegu.com

JSX语法

```
import React from 'react'
import ReactDom from 'react-dom'
let msg=<h1> 你好 </h1>
ReactDom.render(msg,document.getElementById('root'));
```

2. JSX应用与原理分析

博学谷
www.boxuegu.com

嵌入JS表达式

```
let userName='张三';
let msg=<h1> 你好<span>{userName}</span> </h1>
```

2. JSX应用与原理分析

博学谷
www.boxuegu.com

条件渲染

根据不同的条件来渲染不同的JSX结构

```
let isResult=true;
function showUserName(user){
    if(isResult){
        return user.firstName+user.lastName;
    }
    return user.firstName;
}
let msg=<h1> 你好<span>{showUserName({firstName:'li',lastName:'si'})}</span> </h1>
```

2. JSX应用与原理分析

博学谷
www.boxuegu.com

列表渲染

如果需要渲染一组数据，应该怎样进行处理呢?

在这里可以使用map( )方法来完成

```
let users=[{id:1,userName:'张三'},{id:2,userName:'王五'},{id:3,userName:'李四'}]
let msg=<h1>你好{
     users.map(function(userInfo){
         return <span key={userInfo.id}>{userInfo.userName}</span>
     })

}</h1>
ReactDom.render(msg,document.getElementById('root'));
```

2. JSX应用与原理分析

博学谷
www.boxuegu.com

属性处理

给img标签添加一个src属性

```
import imgUrl from './www.jpg'
let msg=<div>你好<img src={imgUrl} /></div>
ReactDom.render(msg,document.getElementById('root'));
```

行内样式处理

```
let msg=<div>你好<img src={imgUrl}
style={{width:'200px',height:'200px'}}/></div>
ReactDom.render(msg,document.getElementById('root'));
```

类样式处理

```
import './style.css'
let msg=<div><img src={imgUrl}  className='imgStyle'/></div>
ReactDom.render(msg,document.getElementById('root'));
```

2. JSX应用与原理分析

博学谷
www.boxuegu.com

JSX原理分析

Babel 会把 JSX 转译成一个名为 React.createElement() 函数调用。

例如，有如下代码

```
const element = (
  <h1 className="greeting">
    Hello, world!
  </h1>
);
```

经过转换后得到如下的代码

```
var element = React.createElement("h1", {
  className: "greeting"
}, "Hello, world!");
```

2. JSX应用与原理分析

博学谷
www.boxuegu.com

render方法模拟实现

思路：

- 在render方法中，根据传递过来的对象，获取type属性的值，创建元素。
- 然后在将props中的内容添加到元素中，最后将创建好的元素追加到容器中。

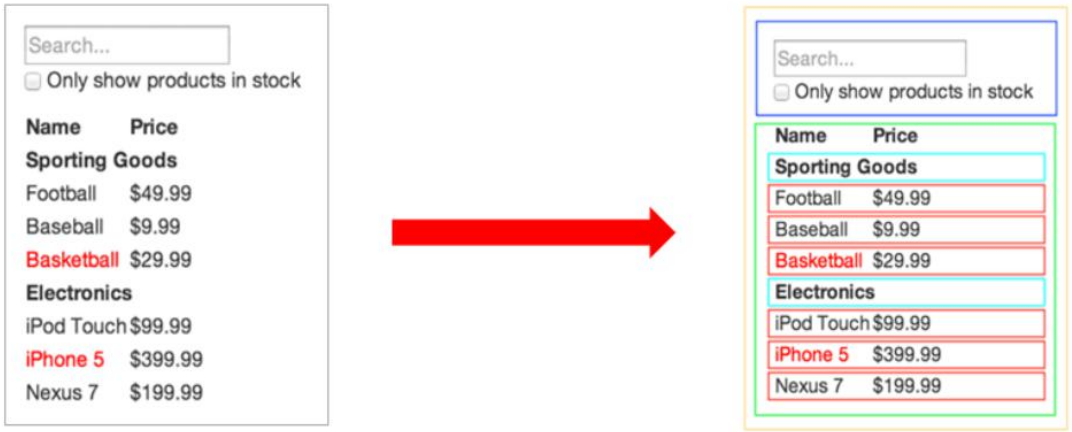
博学谷
www.boxuegu.com

目录 Contents

3. React组件基本应用

组件简介

- 组件是React的一等公民，使用React就是在用组件
- 组件表示页面中的部分功能
- 组合多个组件实现完整的页面功能
- 特点：可复用、独立、可组合



3. React组件基本应用

博学谷
www.boxuegu.com

函数组件创建

关于组件的创建有两种方式，第一种方式是使用“函数来创建组件”，第二种方式使用“类创建组件”。

所谓函数创建组件，其实就是使用JS中的函数或者是箭头函数创建的组件，所以也称之为函数组件。

```
function Message(){
            return (<div><h1>Hello World!</h1></div>)
}
ReactDom.render(<Message/>,document.getElementById('root'))
```

关于函数组件有如下几个约定：

- 约定1：函数名称必须以大写字母开头
- 约定2：函数组件必须有返回值，表示该组件的结构
- 如果返回值为null，表示不渲染任何内容

3. React组件基本应用

博学谷
www.boxuegu.com

类组件创建

所谓的使用类创建组件，其实就是用ES6中的class关键字来创建组件。

```
class Message extends React.Component{
    render() {
        return(
            <div><h1>Hello</h1></div>
        )
    }
}
ReactDom.render(<Message />,document.getElementById('root'))
```

- 约定1：类名称也必须要大写字母开头
- 约定2：类组件应该继承React.Component父类，从而可以使用父类中提供的方法或者属性
- 约定3：类组件必须提供 render 方法
- 约定4：render方法中必须要有return返回值

3. React组件基本应用

博学谷
www.boxuegu.com

组件封装

为了实现组件的可复用，需要将组件单独定义到一个js文件中

这种应用，是以后编程过程中经常使用的方式，也就是将具有独立功能的组件单独的封装到一个js文件中，并把组件进行导出，那么其它文件如果需要用到该组件的时候，直接进行相应的导入就可以了，这样就很好的达到了组件的复用。

3. React组件基本应用

博学谷
www.boxuegu.com

事件绑定

关于组件中的事件处理，这里主要讲解两部分内容，第一部分是关于“事件绑定”，第二部分是关于“事件对象”。

- React事件绑定语法与DOM事件语法相似
- 语法：on+事件名称=事件处理函数，比如 onClick = function(){}
- **注意**：React事件采用驼峰命名法

3. React组件基本应用

博学谷
www.boxuegu.com

事件对象

可以通过事件处理函数的参数获取到事件对象，只不过在React中把事件对象叫做：合成事件。

合成事件：

- 兼容所有浏览器，无需担心跨浏览器兼容问题
- 除兼容所有浏览器外，它还拥有和浏览器原生事件相同的接口，包括 stopPropagation()和preventDefault()

3. React组件基本应用

博学谷
www.boxuegu.com

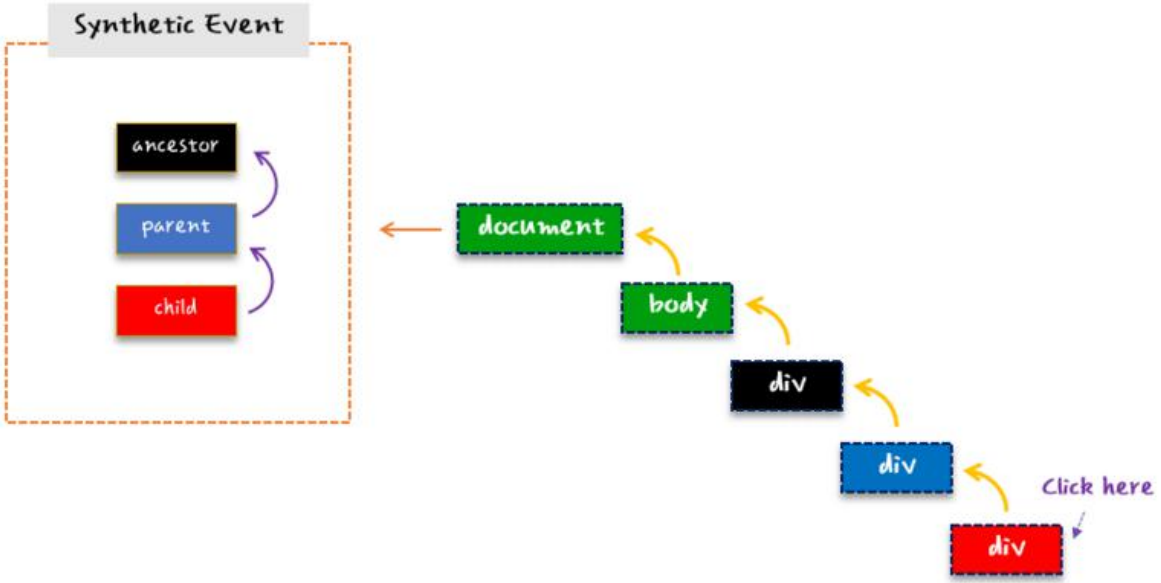
事件流问题

React中的事件，默认的事件传播方式为**冒泡**

3. React组件基本应用

事件委托

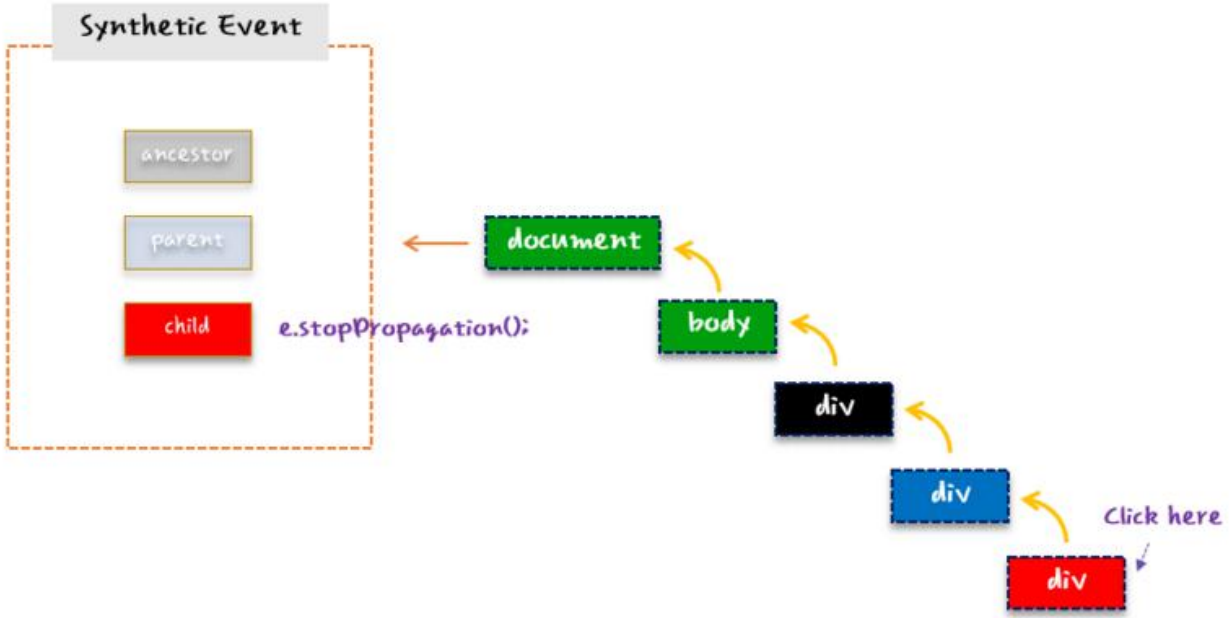
在React事件系统中，所有的事件都是绑定在document元素，虽然我们在某个react元素上绑定了事件，但是，最后事件都委托给document统一触发。如下图所示：



3. React组件基本应用

事件委托

如果想阻止事件流的传播，可以使用`e.stopPropagation( )`方法完成。



3. React组件基本应用

博学谷
www.boxuegu.com

事件对象

关于React中的事件对象，并不是原生的事件对象(event),只是说，我们可以通过它获取到原生event对象上的某些属性。

比如示例中的clientX和clientY .对于这个event对象，在整个合成事件中，只有一个，被全局共享，也就是说，当这次事件调用完成之后，这个event对象会被清空，等待下一次的事件触发，因此，我们无法在异步的操作中获取到event .

3. React组件基本应用

有状态组件和无状态组件

所谓的有状态组件：指的是用类创建的组件。

所谓的无状态组件：指的是用函数创建的组件。

通过这个名称，可以发现函数组件与类组件的区别就是在状态上，那么状态指的就是数据。

比如计数器案例中，点击按钮让数值加1。0和1就是不同时刻的状态，而由0变为1就表示状态发生了变化。状态变化后，UI也要相应的更新。React中想要实现该功能，就要使用有状态组件来完成。

3. React组件基本应用

博学谷
www.boxuegu.com

组件中的state和setState

state的基本使用与如何使用setState方法完成状态的修改

关于state的基本使用方式

- 状态(state)即数据，是组件内部的私有数据，只能在组件内部使用
- state的值是对象，表示一个组件中可以有多个数据
- 通过this.state来获取状态

3. React组件基本应用

博学谷
www.boxuegu.com

组件中的state和setState

setState( )方法修改状态

- 状态是可变的
- 语法：this.setState({要修改的数据})
- setState() 作用：1.修改 state 2.更新UI

3. React组件基本应用

事件处理中的this问题

如下程序是否会出错?

```
import React,{Component}from 'react'
class CounterCom extends Component{
    state = {
        count:0
    }
   handleCountChange(){
               this.setState({
                    count:this.state.count+1
                })
   }
    render(){
        return(
            <div>
            计算器:{this.state.count}
            <button onClick={this.handleCountChange}>计算</button>
          </div>
        )
    }
}
export default CounterCom
```

3. React组件基本应用

通过箭头函数解决this问题

如下程序所示

```
import React,{Component}from 'react'
class CounterCom extends Component{
    state = {
        count:10
    }
    handleCountChange(){
                this.setState({
                            count:this.state.count+1
                 })
   }
    render(){
        return(
            <div>
            计算器:{this.state.count}
               <button onClick={()=>this.handleCountChange()}>计算</button>
          </div>
        )
    }
}
export default CounterCom
```

3. React组件基本应用

利用bind方法解决this问题

如下程序所示

```
import React,{Component}from 'react'
class CounterCom extends Component{
     constructor(){
         super()
         this.state={
               count:10
               }
         this.handleCountChange=this.handleCountChange.bind(this);
     }
   handleCountChange(){
               this.setState({
                    count:this.state.count+1
                })
   }
    render(){
        return(
            <div>
            计算器:{this.state.count}
             <button onClick={this.handleCountChange}>计算</button>
          </div>
        )
    }
}
export default CounterCom
```

3. React组件基本应用

博学谷
www.boxuegu.com

setState方法的原理说明

setState方法可以用来更新数据，但是要注意的是更新数据的操作是异步的。

3. React组件基本应用

博学谷
www.boxuegu.com

setState方法第二种语法

通过上一个案例，可以发现，第二次执行setState方法时，是无法获取到第一次执行setState的结果的。

如果现在，第二个setState方法想获取第一个setState方法的结果，应该怎样实现呢?

可以采用如下的语法格式。

```
setState((state,props) => {})
```

3. React组件基本应用

博学谷
www.boxuegu.com

setState方法第二个参数

setState方法也可以添加第二个参数，第二个参数是一个回调函数。

语法：`setState(update[,callback])`

在状态更新后，立即执行某个操作那么可以使用第二个参数。

3. React组件基本应用

博学谷
www.boxuegu.com

受控组件

在React中关于表单分为两部分内容，受控组件和非受控组件。我们将其值受到React控制的表单元素称之为受控组件

```
import React,{Component}from 'react'
class App extends Component{
    state={
        num:10
    }
    handleChange=e=>{
        this.setState({
            num:e.target.value
        })
    }
    render(){
        return (
            <div>
                {this.state.num}
                <input type="text" value={this.state.num} onChange={this.handleChange}/>
            </div>
        )
    }
}
export default App
```

3. React组件基本应用

博学谷
www.boxuegu.com

富文本框案例

如下程序所示

```
import React,{Component}from 'react'
class App extends Component{
    state={
        content:'Hello'
    }
    handleChange=e=>{
        this.setState({
            content:e.target.value
        })
    }
    render(){
        return (
            <div>
                {this.state.content}
                <textarea  value={this.state.content}  onChange={this.handleChange}></textarea>
            </div>
        )
    }
}
export default App
```

3. React组件基本应用

下拉框应用

```
import React,{Component}from 'react'
class App extends Component{
    state={
        city:'shanghai'
    }
    handleChange=e=>{
        this.setState({
            city:e.target.value
        })
    }
    render(){
        return (
            <div>
                {this.state.city}
                <select value={this.state.city} onChange={this.handleChange}>
                    <option value='beijing'>北京</option>
                    <option value='shanghai'>上海</option>
                    <option value='guangzhou'>广州</option>
                </select>
            </div>
        )
    }
}
export default App
```

3. React组件基本应用

博学谷
www.boxuegu.com

复选框应用

```
import React,{Component}from 'react'
class App extends Component{
    state={
        isChecked:true
    }
    handleChange=e=>{
        this.setState({
            isChecked:e.target.checked
        })
    }
    render(){
        return (
            <div>
             复选框<input type="checkbox" checked={this.state.isChecked} onChange={this.handleChange}/>
            </div>
        )
    }
}
export default App
```

3. React组件基本应用

博学谷
www.boxuegu.com

受控组件更新state流程

(1) 可以通过在初始state中设置表单的默认值。

(2) 给表单添加onChange，并且当表单中的值发生了变化时，会调用onChange所指定的函数。

(3) 在所对应的处理函数中通过合成事件对象获取到表单改变后的状态，并且更新对应的state.

(4) setState触发页面的重新渲染，完成表单中值的更新。

3. React组件基本应用

博学谷
www.boxuegu.com

表单处理优化

现在，我们思考一个问题，如果一个页面中需要的表单元素比较多，应该怎样进行处理呢?

具体实现步骤如下：

① 给表单元素添加name属性（用来区分是哪一个表单），名称与state相同（用来更新数据的）

② 根据表单内容来获取对应值

③ 在change事件处理程序中通过 [name] 来修改对应的state

3. React组件基本应用

博学谷
www.boxuegu.com

非受控组件

非受控组件，不在受状态state的控制。它需要借助于ref,使用元素DOM方式获取表单元素值。

ref的作用就是用来获取DOM的，所以说这种方式是直接操作DOM的方式。

具体实现步骤如下：

① 调用 React.createRef() 方法创建ref对象

② 将创建好的 ref 对象添加到文本框中

③ 通过ref对象获取到文本框的值

3. React组件基本应用

博学谷
www.boxuegu.com

实现简单的TodoList布局

```
import React,{Component}from 'react'
class TodoList extends Component{
    render(){
        return (
            <div>
                <div><input type="text"/><button type="button">提交</button></div>
                <ul>
                    <li>
                        国内新闻
                    </li>
                    <li>国际新闻</li>
                </ul>
            </div>
        )
    }
}
export default TodoList
```

3. React组件基本应用

博学谷
www.boxuegu.com

实现TodoList列表展示

实现TodoList列表展示的功能非常简单，就是将数组中存储的数据，通过map方法进行遍历，然后展示在页面中。

3. React组件基本应用

博学谷
www.boxuegu.com

实现TodoList信息添加

实现TodoList信息的添加，就是将用户在文本框中输入的数据保存到数组中。

3. React组件基本应用

博学谷
www.boxuegu.com

实现TodoList信息删除

在进行信息删除的时候，首先获取要删除的数据在数组中的索引，然后再进行删除操作，这里主要用到了splice方法。

博学谷
www.boxuegu.com

目录 Contents

4. React组件高级应用

博学谷
www.boxuegu.com

props基本使用

组件之间是封闭的，要接收外部数据应该用props来完成，props的作用就是接收传递给组件的数据。

```
import React from 'react'
function ShowMessage(props) {
    return(<div>
        <h1>我叫{props.name},今年{props.age}岁</h1>
    </div>)
}
export default ShowMessage
```

4. React组件高级应用

博学谷
www.boxuegu.com

props特点

1、可以给组件传递任意类型的数据。

2、props对象是只读的，只能读取其中的属性值，不能修改对应的属性值。

3、在使用类组件的时候，如果写了构造函数，应该将props传递给super( ),否则在构造函数中无法获取到props.

4. React组件高级应用

博学谷
www.boxuegu.com

props中的children属性

该属性表示组件标签的子节点，该属性可以接收各种类型的数据。

4. React组件高级应用

博学谷
www.boxuegu.com

props校验

props虽然可以接收组件外部传递过来的各种数据，但是在组件内部使用props的时候，往往是有一定的数据格式上的要求的。

可以通过prop-types来完成对应的校验

安装：npm i prop-types

4. React组件高级应用

博学谷
www.boxuegu.com

校验规则说明

常用的校验规则

(1) 常见类型： array、bool、function、number、object、string

(2) React元素类型：element

(3) 必须填写：isRequired

(4) 特定结构的对象: shape({ })

其它的校验规则

https://react-1251415695.cos-website.ap-chengdu.myqcloud.com/docs/typechecking-with-proptypes.html

4. React组件高级应用

博学谷
www.boxuegu.com

props默认值

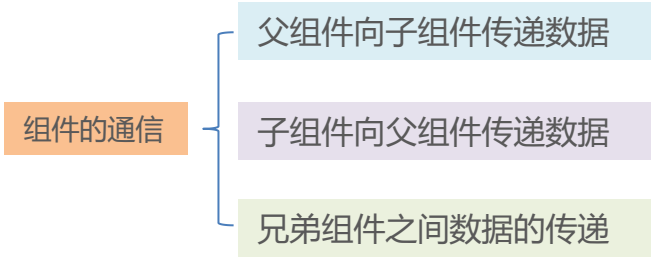
在使用props的时候，可以为其指定相应的默认值，如果在渲染组件的时候，没有指定具体的值，那么就可以使用指定的默认值。

```
import React from 'react'
class App extends React.Component{
    render() {
          return (
            <div>
               {
                this.props.num
               }
             </div>
        )
    }
}
App.defaultProps = {
    num:10
}
export default App
```

4. React组件高级应用

博学谷
www.boxuegu.com

父组件向子组件传递数据



父组件向子组件传递数据的基本流程：

① 首先在父组件中导入子组件。

② 在父组件中定义好要传递给子组件的数据。

③ 在父组件中引用子组件的时候，给子组件指定属性，并且将需要传递的数据赋值给该属性。

④ 在子组件中通过props接收父组件中传递过来的数据。

4. React组件高级应用

博学谷
www.boxuegu.com

子组件向父组件传递数据

具体步骤：

（1）父组件提供一个回调函数，用于接收子组件传递过来的数据。

（2） 将该函数作为属性的值，传递给子组件。

(3) 子组件通过props调用回调函数。

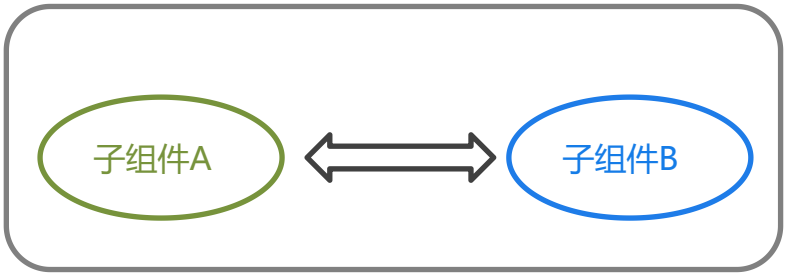
（4） 将子组件的数据作为参数传递给回调函数。

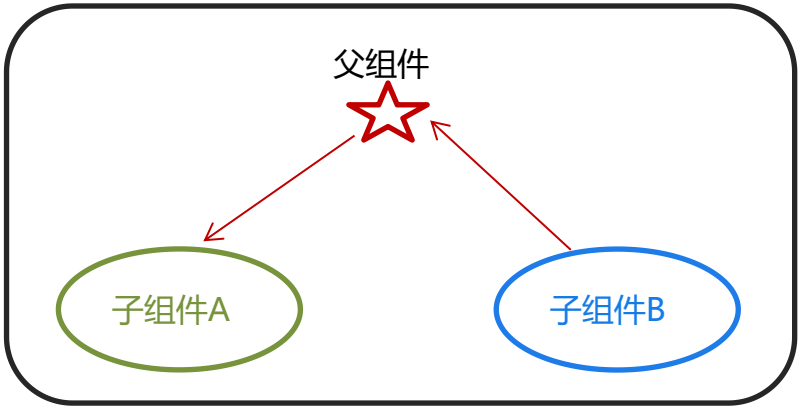
4. React组件高级应用

兄弟组件通信

将共享的数据提升到最近的公共父组件中，有父组件管这个共享的数据。

如下图所示：





4. React组件高级应用

博学谷
www.boxuegu.com

使用自定义事件的方式组件通信

基本思路：

子组件B（child1组件）向子组件A（child2组件）中进行通信。

在子组件A中定义一个事件，这个事件绑定一个方法，也就是当事件发生后，调用该方法。

那么该事件什么时候出发呢？是在单击了子组件B中的按钮后触发，事件触发后会调用所绑定的方法，

在该方法中完成的内容就是状态的更新了。

通过上面的叙述，我们应该很清楚的想到，这就是发布/订阅机制。

需要安装events 包：`npm install events --save`

4. React组件高级应用

博学谷
www.boxuegu.com

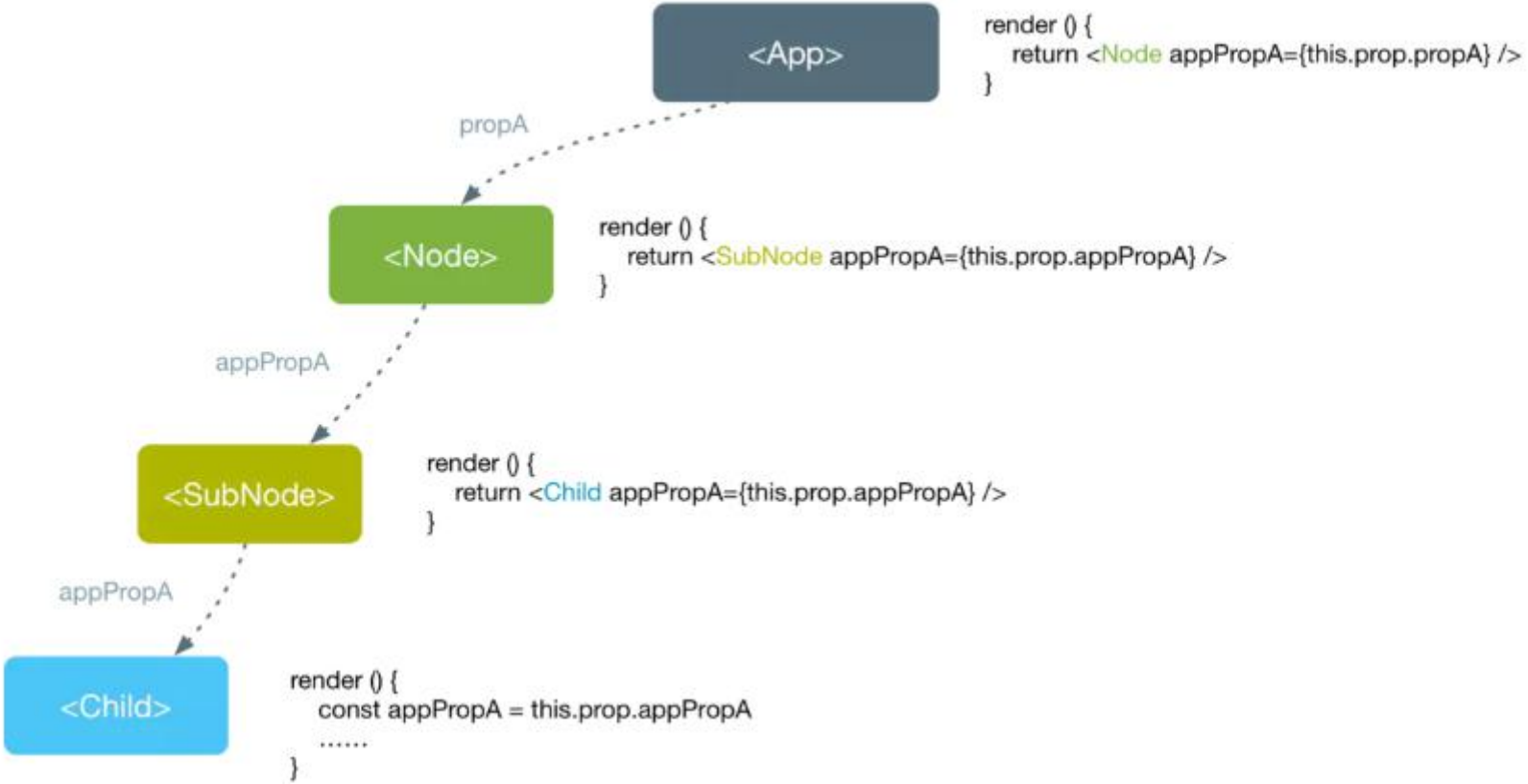
跨级组件通信

如果出现组件层级比较多的情况下（例如：爷爷传递数据给孙子），如果使用props来进行传递会非常的麻烦，这时会使用Context来进行传递。

Context就是用来实现跨组件传递数据的。

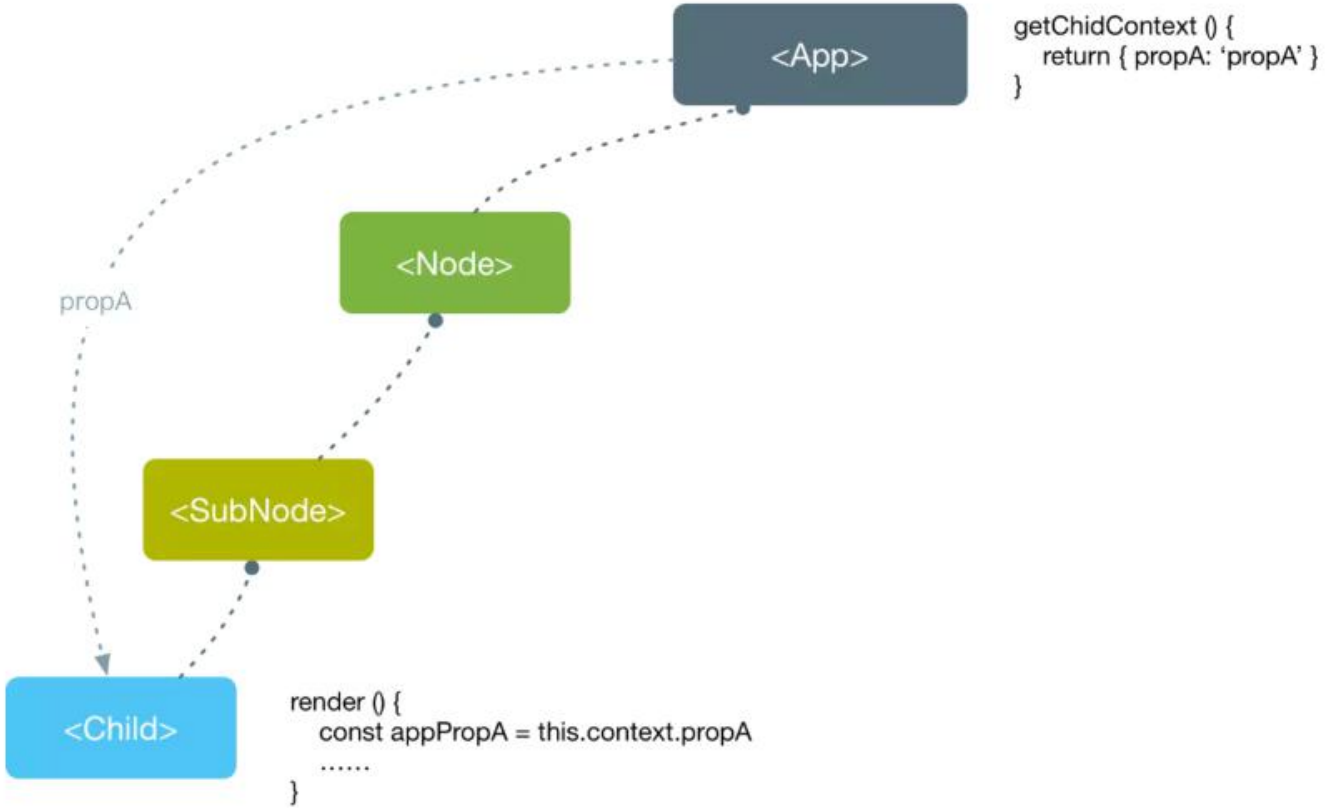
4. React组件高级应用

跨级组件通信



4. React组件高级应用

跨级组件通信



4. React组件高级应用

博学谷
www.boxuegu.com

跨级组件通信

具体的实现步骤：

（1）调用 React.createContext() 创建 Provider(提供数据) 和 Consumer(消费数据) 两个组件

（2）使用Provider 组件作为父节点

（3）设置value属性，表示要传递的数据

(4) 哪一层想要接收数据，就用Consumer进行包裹，在里面回调函数中的参数就是传递过来的值

4. React组件高级应用

博学谷
www.boxuegu.com

完善TodoList案例---组件拆分与传值

在前面所做的TodoList案例中，我们将所有的内容都写到一个组件中了，但是在最开始的课程中，我们讲解过，组件具有单一功能原则的特点。

也就是说，一个组件原则上只能负责一个功能。

如果它需要负责更多的功能，这时候就应该考虑将它拆分成更小的组件。

4. React组件高级应用

博学谷
www.boxuegu.com

完善TodoList案例---对循环遍历代码的优化

仔细观察前面的代码，可以发现在JSX中写了map的循环，那么这样做会导致JSX的业务太重，JSX是展示数据的。

如果在其中添加业务代码，会让整个JSX的结构变的比较复杂，所以这里可以将这块代码封装到一个函数中进行处理。

4. React组件高级应用

博学谷
www.boxuegu.com

完善TodoList案例---setState方法的修改

回顾一下，setState方法添加一个回调函数作为参数的应用

4. React组件高级应用

博学谷
www.boxuegu.com

生命周期简介

组件的生命周期：组件从被创建到挂载到页面中运行，再到组件不用的时候卸载的过程。

意义：有助于理解组件的运行方式，完成更复杂的组件的功能，分析组件错误原因等。

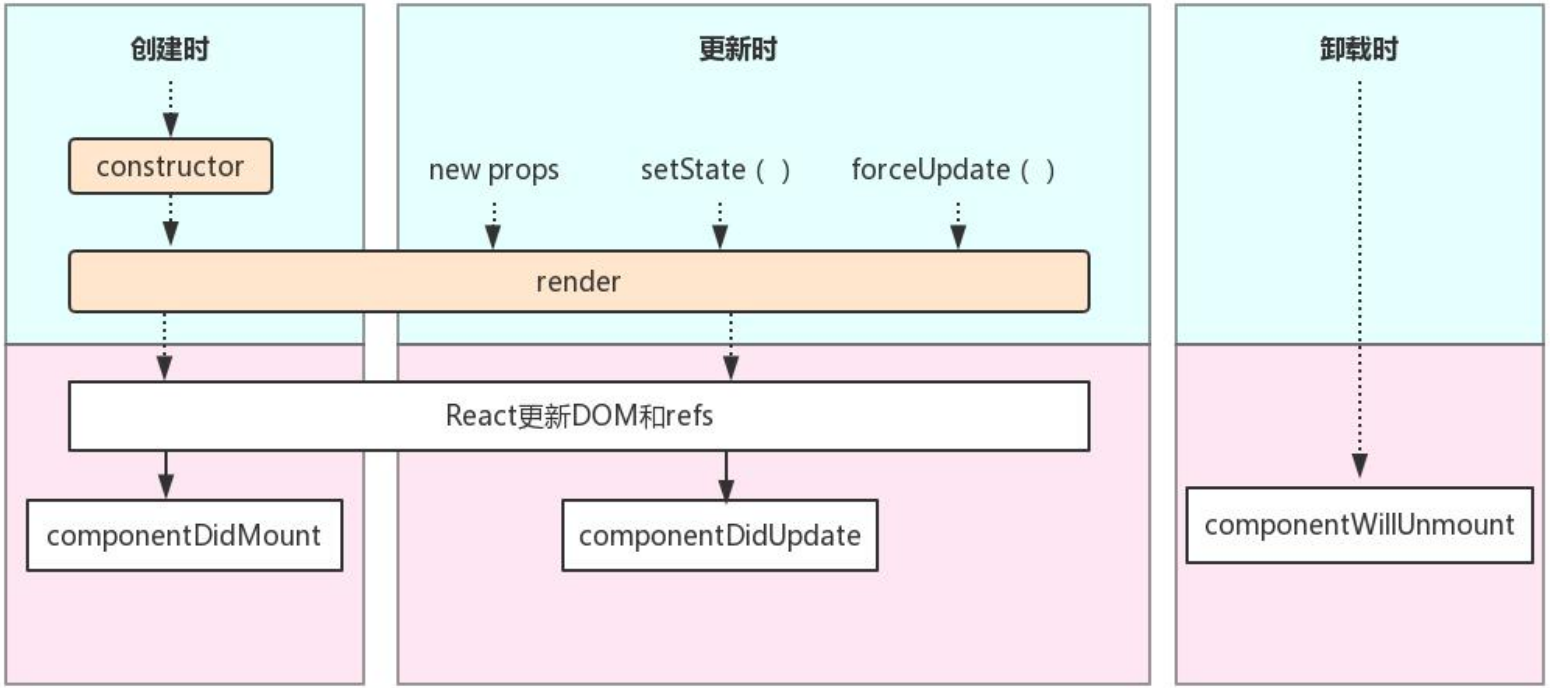
钩子函数：生命周期的每个阶段总是会有一些方法被调用，这些方法就是生命周期中的‘钩子函数’。

注意：只有类组件才有生命周期，函数组件没有。

4. React组件高级应用

生命周期简介

生命周期的阶段



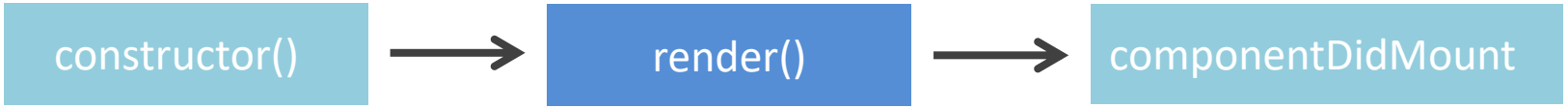
4. React组件高级应用

博学谷
www.boxuegu.com

生命周期---挂载阶段

执行时机：组件创建时（页面加载时）

钩子函数执行顺序：



4. React组件高级应用

博学谷
www.boxuegu.com

生命周期---挂载阶段

以下是这三个钩子函数的触发时机和作用：

钩子函数	触发时机	作用
constructor	创建组件时，最先执行	1. 初始化state 2. 为事件处理程序绑定this
render	每次组件渲染都会触发	渲染UI（注意：不能调用setState()）
componentDidMount	组件挂载（完成DOM渲染）后	1. 发送网络请求 2. DOM操作

4. React组件高级应用

生命周期---更新阶段

执行时机：setState()、 forceUpdate()、 组件接收到新的props

说明：以上三者任意一种变化，组件就会重新渲染。

执行顺序：

钩子函数	触发时机	作用
render	每次组件渲染都会触发	渲染UI（与 挂在阶段 是同一个render）
componentDidUpdate	组件更新（完成DOM渲染）后	1 发送网络请求 2 DOM操作 注意：如果要setState() 必须放在一个if条件中

4. React组件高级应用

生命周期---卸载阶段

执行时机：组件从页面中消失

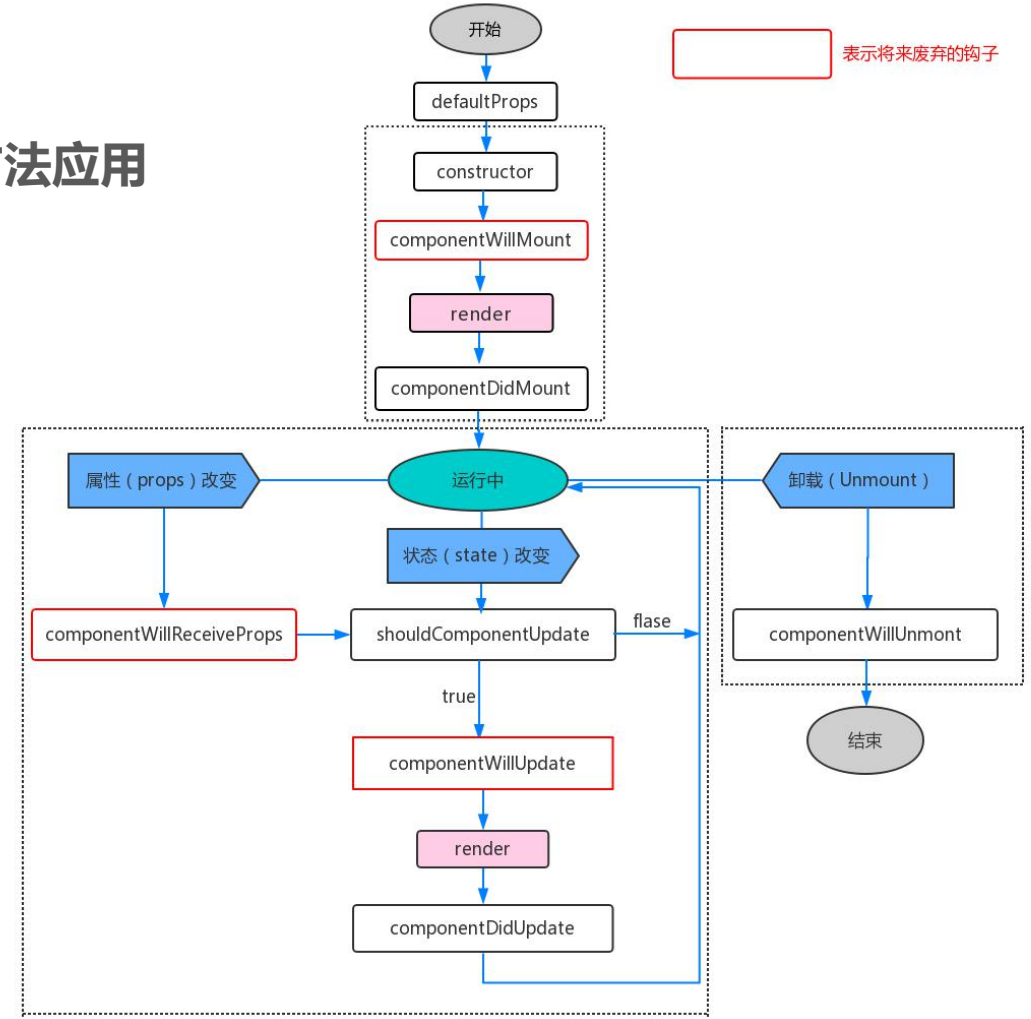
作用：用来做清理操作

钩子函数	触发时机	作用
componentWillUnmount	组件卸载（从页面中消失）	执行清理工作（比如：清理定时器等）

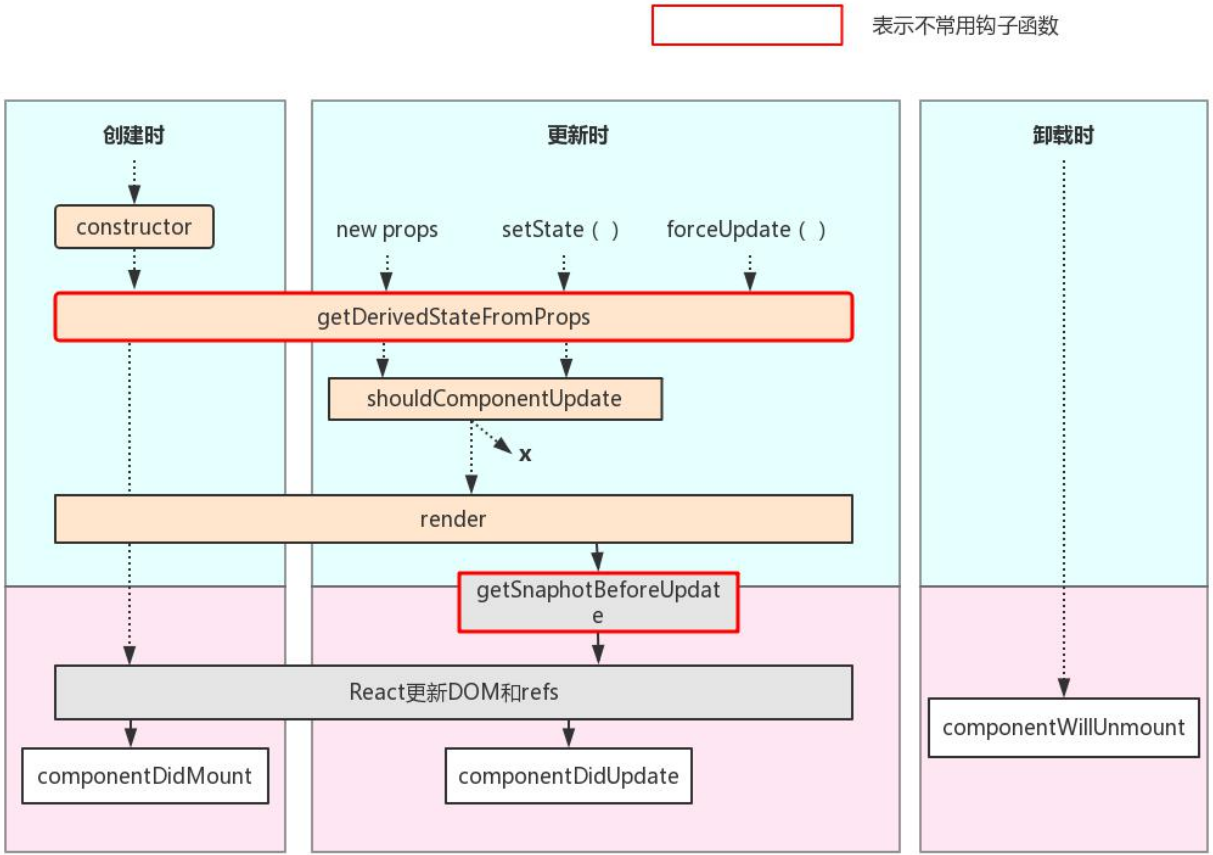
4. React组件高级应用

shouldComponentUpdate方法应用

旧版的生命周期钩子函数



4. React组件高级应用



4. React组件高级应用

博学谷
www.boxuegu.com

shouldComponentUpdate方法应用

shouldComponentUpdate方法的作用：

- 当组件要更新的时候，询问一下是否要真正的更新。
- 如果该方法返回的是false,表示不更新，也就是不执行render函数。
- 该方法默认返回的是true.

4. React组件高级应用

博学谷
www.boxuegu.com

发送AJAX请求

发送Ajax请求，可以在componentDidMount函数中完成,在这里要发送Ajax请求可以借助于axios来完成。

axios安装

```
npm i axios
```

4. React组件高级应用

博学谷
www.boxuegu.com

高阶组件介绍

高阶组件(HOC、Higher-Order Component) 是一个函数，接收要包装的组件，返回增强后的组件。

高阶组件实现的具体思路：

高阶组件内部创建了一个类组件，在这个类组件中提供复用的状态逻辑代码，通过prop将复用的状态传递给被包装组件。

如下的伪代码所示：

```
function withMouse(WrappedComponent) {

    class Mouse extends React.Component {
        //这里的代码是复用状态的逻辑代码

         render() {
            return <WrappedComponent {...this.state} />
        }
    }
     return Mouse
}
```

4. React组件高级应用

博学谷
www.boxuegu.com

高阶组件介绍

高阶组件具体的实现步骤：

① 创建一个函数，名称约定以with开头

② 指定函数参数，参数应该以大写字母开头

③ 在函数内部创建一个类组件，提供复用的状态逻辑代码，并返回

④ 在该组件中，渲染参数组件，同时将状态通过prop传递给参数组件

⑤ 调用该高阶组件，传入要增强的组件，通过返回值拿到增强后的组件，并将其渲染到页面

4. React组件高级应用

博学谷
www.boxuegu.com

高阶组件案例1

下面通过一个获取鼠标坐标的这么一个案例，来讲解一下关于高阶组件的具体使用方式。

怎样获取鼠标的坐标，是由高阶函数来完成的。

4. React组件高级应用

博学谷
www.boxuegu.com

高阶组件案2

现在，又有一个需求了，需要让一张图片跟随鼠标移动，那么你可以想一下，这个案例中是不是也要获取鼠标的位置，然后来决定图片的位置。

获取鼠标的位置的方式和前面案例中的高阶函数中实现的是一样的，所以可以很好的复用这部分的代码。

这就是高阶组件的好处。

4. React组件高级应用

博学谷
www.boxuegu.com

高阶组件---传递props

在高阶组件中不仅将state向包装的组件进行传递，也要将props向包装的组件进行传递。

4. React组件高级应用

博学谷
www.boxuegu.com

高阶组件案例---权限校验1

案例要求：

系统中有多个页面，要求这些页面必须有Admin的的角色才能访问。

分析一下：

获取角色信息以及关于角色的判断其实都是相同的，所以这里可以将获取角色的信息以及角色的判断都写到高阶组件中。

4. React组件高级应用

博学谷
www.boxuegu.com

高阶组件案例---权限校验2

现在关于权限校验的需求又发生了变化，要求PageA页面只能是VIP的角色才能够访问。应该怎样进行处理呢?

创建一个高阶组件叫做withVIPAuth来处理VIP的角色呢?

更简单的解决办法:

在高阶组件之上再抽象一层。因为高阶组件中的角色校验基本上是一样的，所以我们要做的就是实现一个返回高阶组件的函数，把变化的部分（Admin,VIP）抽离出来。

4. React组件高级应用

博学谷
www.boxuegu.com

高阶组件---装饰者模式

装饰者模式:

在不改变对象自身的前提下在程序运行期间动态的给对象添加一些额外的属性或行为。

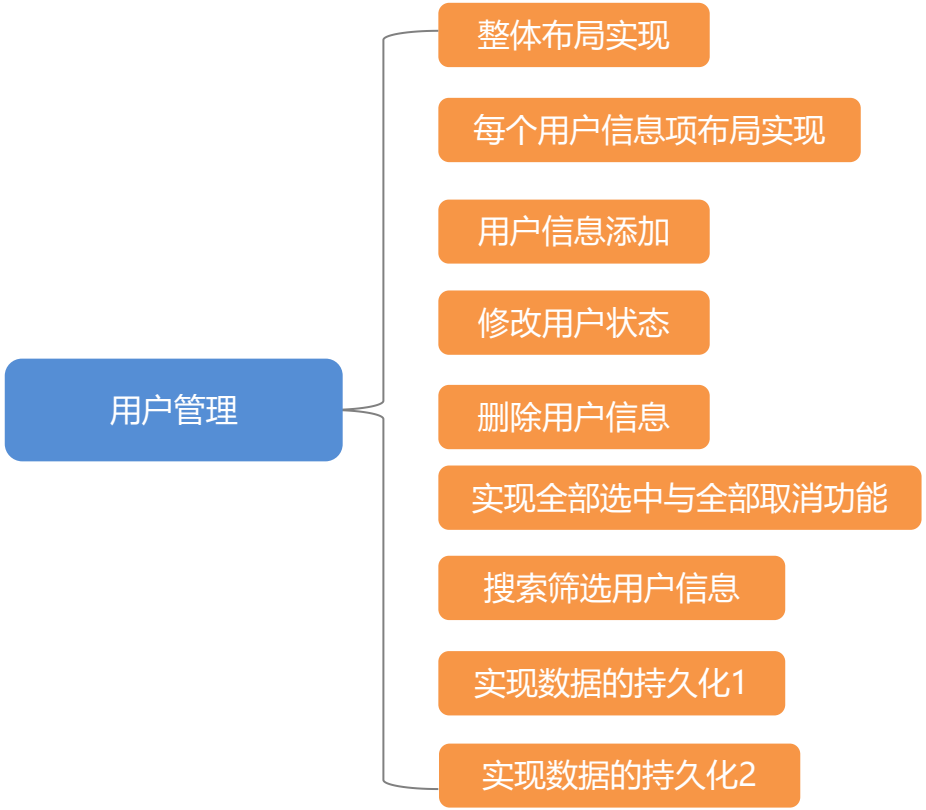
高阶组件总结:

- 高阶组件不是组件，是一个把某个组件转换成另一个组件的函数
- 高阶组件的主要作用是代码复用
- 高阶组件是装饰器模式在React中的实现

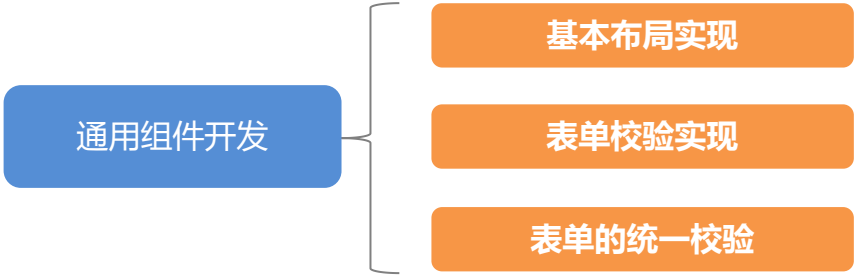
博学谷
www.boxuegu.com

目录 Contents

5. React组件应用案例



5. React组件应用案例



一样的在线教育，不一样的教学品质