杰普软件科技有限公司

www.briup.com

电邮：training@briup.com
主页：http://www.briup.com

昆山地址：昆山市巴城镇学院路828号昆山浦东软件园北楼4、5、8层
邮编：215311
电话：0512-50190298

上海地址：上海市闸北区万荣路1188弄G栋102室-上海服务外包科技园龙软园区
邮编：200436
电话：021-56778147

Briup High-End IT Training

React

Brighten Your Way And Raise You Up.

学习目标

- 掌握 react 工程的创建
- 掌握 JSX 语法，熟悉数据绑定、列表渲染、条件渲染
- 掌握函数组件、类组件的创建方式，生命周期
- 掌握如何使用props机制完成父子组件通信
- 掌握事件绑定机制
- 掌握表单的双向数据绑定
- 熟练使用 react-router 完成路由管理
- 熟悉ant design组件库，精通Table、Modal、Form等常见组件的应用
- 熟练使用redux进行状态管理
- 熟练使用redux-saga/redux-thunk

章节简介

- 第 1 章：react快速入门
- 第 2 章：组件基础
- 第 3 章：双向数据绑定、路由
- 第 4 章：组件库
- 第 5 章：axios
- 第 6 章：redux
- 第 7 章：redux-saga、redux-thunk
- 第 8 章：dva
- 第 9 章：综合应用
- 第10章：react-native

杰普软件科技有限公司

www.briup.com

电邮：training@briup.com
主页：http://www.briup.com

昆山地址：昆山市巴城镇学院路828号昆山浦东软件园北楼4、5、8层
邮编：215311
电话：0512-50190298

上海地址：上海市闸北区万荣路1188弄G栋102室-上海服务外包科技园龙软园区
邮编：200436
电话：021-56778147

Briup High-End IT Training

第 1 章：react快速入门

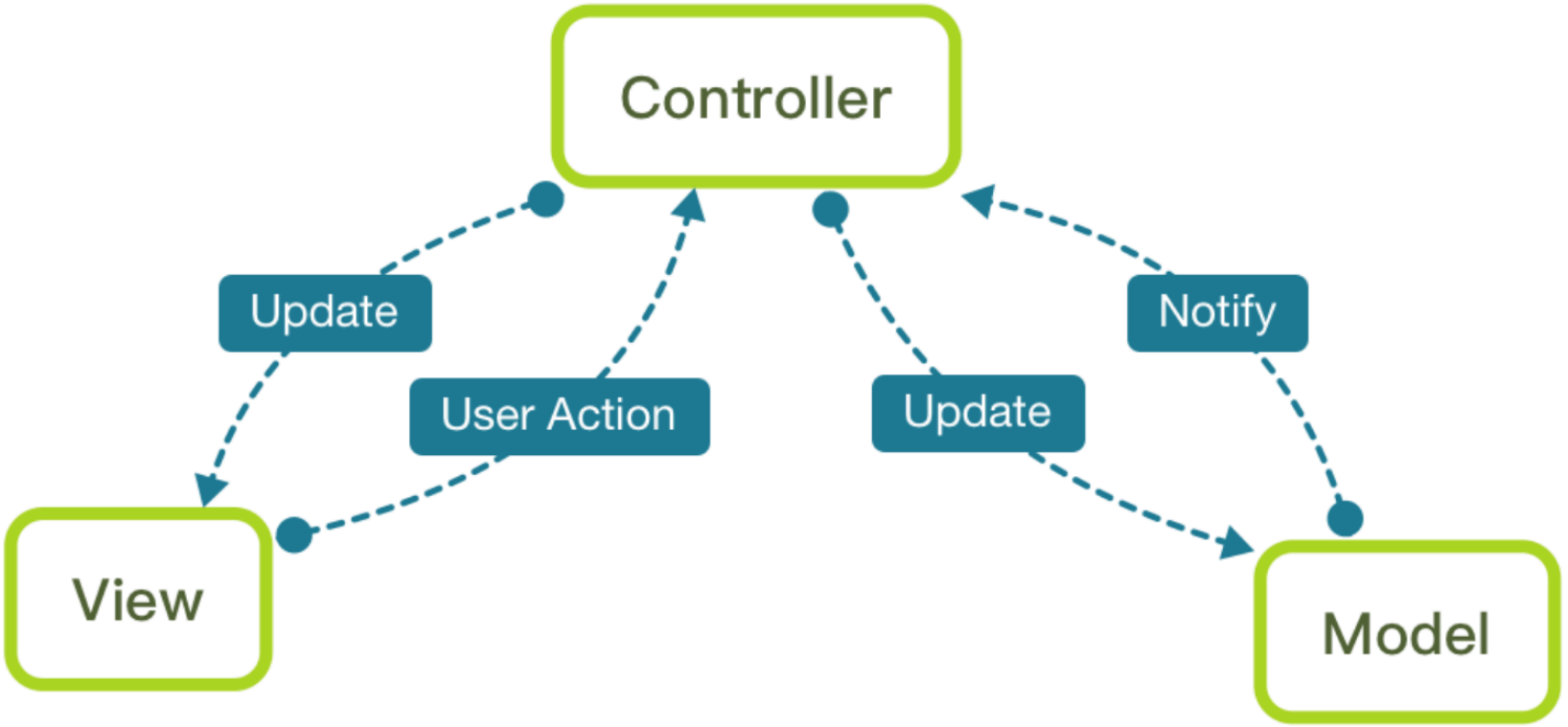
Brighten Your Way And Raise You Up.

学习目标

- 了解mvc模式与mvvm模式
- 认识react
- 掌握创建react项目的方式
- 掌握JSX基本语法
- 掌握组件创建方式
- 掌握基本渲染，列表渲染，条件渲染

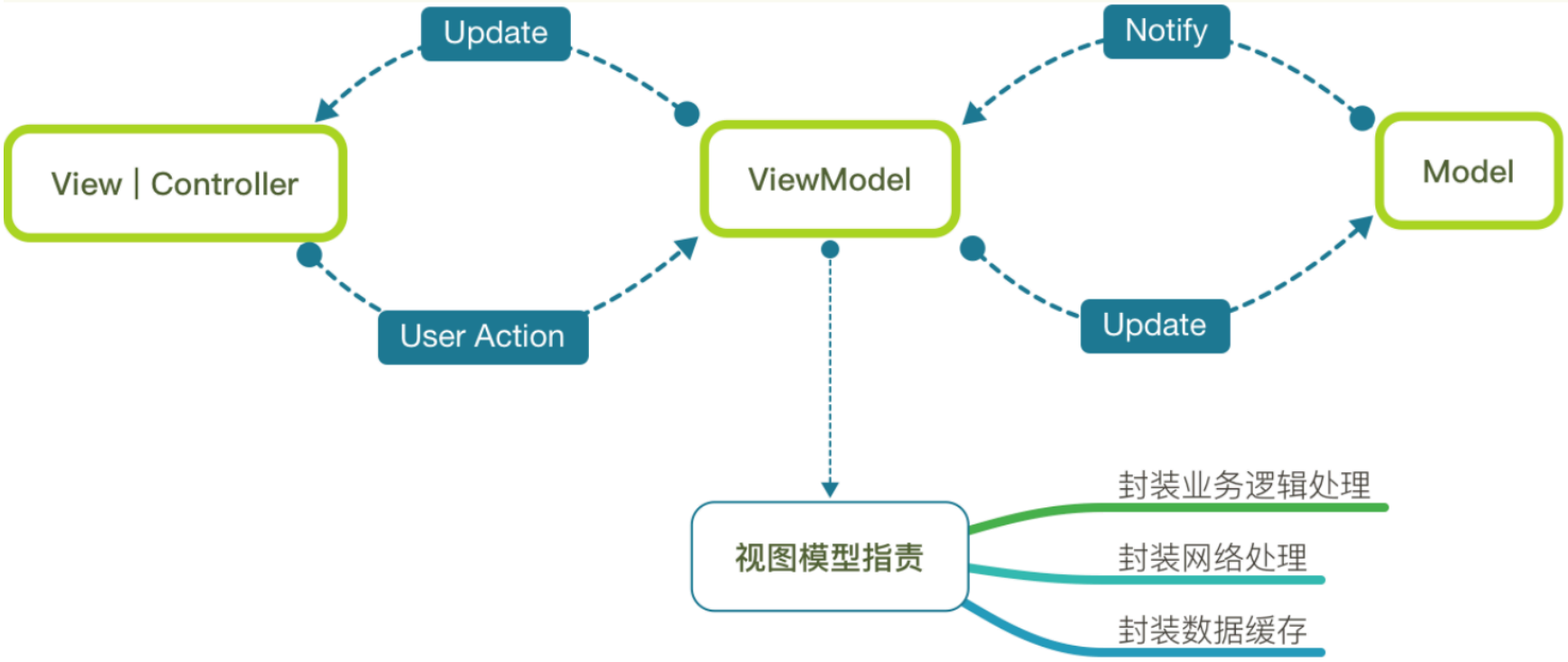
mvc

后台主导开发的思想。这种思想一般应用在后端中的web层，m表示model数据模型，v表示view视图，c表示controller控制器。在servlet/jsp中主要应用的就是mvc思想。



mvvm

前后台分离开发中的前端思想。m表示model数据模型，v表示view视图，vm表示视图模型，负责把Model的数据同步到View显示出来，还负责把View的修改同步回Model。React/Vue、AngularJS中主要应用的就是mvvm思想。



认识 React

- React是一个用于构建用户界面的 JAVASCRIPT 库。
- React主要用于构建UI，很多人认为 React 是 MVC 中的 V（视图）。mvvm
- React起源于 Facebook 的内部项目，用来架设 Instagram 的网站，并于 2013 年 5 月开源。
- React 拥有较高的性能，代码逻辑非常简单，越来越多的人已开始关注和使用它

React特点

- **声明式设计**:React采用声明范式，可以轻松描述应用。
- **高效** :React通过对DOM的模拟，最大限度地减少与DOM的交互。
- **灵活** :React可以与已知的库或框架很好地配合。
- **JSX** :JSX 是JavaScript语法的扩展。React 开发不一定使用 JSX ，但我们建议使用它。
- **组件**:通过 React构建组件，使得代码更加容易得到复用，能够很好的应用在大项目的开发中
- **单向响应的数据流**:React 实现了单向响应的数据流，从而减少了重复代码，这也是它为什么比传统数据绑定更简单

◆ React生态圈

react 的生态体系比较庞大，它在web端，移动端，桌面端、服务器端，VR领域都有涉及。react可以说是目前为止最热门，生态最完善，应用范围最广的前端框架。react结合它的整个生态，它可以横跨web端，移动端，服务器端，乃至VR领域。

可以毫不夸张地说，react已不单纯是一个框架，而是一个行业解决方案

使用CDN

注意：在浏览器中使用 Babel 来编译 JSX 效率是比较低的。如果在html中直接使用react需要导入react、react-dom、babel

```
<!-- 导入react -->
<script src="https://cdn.bootcss.com/react/16.6.0/umd/react.development.js"></script>
<!-- 导入react-dom -->
<script src="https://cdn.bootcss.com/react-dom/16.6.0/umd/react-dom.development.js"></script>
<!-- 导入babel -->
<script src="https://cdn.bootcss.com/babel-standalone/6.26.0/babel.min.js"></script>
```

使用脚手架

```
# 安装node，并且node版本应该满足 Node >= 6 and npm >= 5.2
$ node -v
v10.14.1
# 全局安装脚手架
$ cnpm install -g create-react-app
# 使用脚手架创建工程
$ create-react-app my-app
# 启动工程
$ cd my-app/
$ npm start
```

Hello world

这里使用在html的头部通过cdn导入react、react-dom、babel来应用react。但是要注意的是在企业级开发普遍使用create-react-app脚手架来创建项目

```
<div id="app"></div>
<script type='text/babel'>
        ReactDOM.render( <h1>Hello, world!</h1>, document.getElementById('app') );
</script>
```

◆ 简介

React 使用 JSX 来替代常规的 JavaScript。JSX 是一个看起来很像 XML 的 JavaScript 语法扩展。JSX是javascript的语法糖

我们不需要一定使用 JSX，但它有以下优点：

1. JSX 执行更快，因为它在编译为 JavaScript 代码后进行了优化。
2. 它是类型安全的，在编译过程中就能发现错误。
3. 使用 JSX 编写模板更加简单快速。

```
const element = <h1>Hello, world!</h1>;
=》
<MyButton color="blue" shadowSize={2}>
 Click Me
</MyButton>
```

组件

◆ 组件定义

定义组件最简单的方式就是编写 JavaScript 函数，该函数是一个有效的 React 组件，因为它接收唯一带有数据的 “props”（代表属性）对象与并返回一个 React 元素。这类组件被称为“函数组件”，因为它本质上就是 JavaScript 函数。

```
function Welcome(props) {
        return <h1>Hello, {props.name}</h1>;
}
```

◆ 组件调用

通过标签来调用已经定义好的组件

```
<Welcome name=‘张三’/>
```

◆ 基本渲染

将变量的值直接显示到页面中。在jsx中可以在大括号直接编写Js代码，如果是变量，则直接输出。

```
function Welcome(props) {
  let msg = 'hello world'
  return <h1>{msg}</h1>;
}
```

◆ 列表渲染

可以通过使用 {} 在 JSX 内构建一个元素集合，key 帮助 React 识别哪些元素改变了，比如被添加或删除。因此你应当给数组中的每一个元素赋予一个确定的标识

```
<ul className="list">
  {
      arr.map((item,index) => return <li key={index}>{item}</li>)
  }
</ul>
```

◆ 条件渲染

React中的条件渲染和JavaScript中的一样，使用JavaScript运算符if或者条件运算符去创建元素来表现当前的状态，然后让React根据它们来更新UI

```
function UserInfo(props){
  let {user} = props;
  if(user) {
    return (
        <div>欢迎您 {user.name} 头像</div>
    )
  }
  return (
    <div><a href="#">亲，请登录</a></div>
  )
}
ReactDOM.render(<UserInfo user={{name:'zhangsan'}}/>,document.getElementById('app'));
```

组件传值

props

调用组件的时候可以传递任何参数作为prop,如果传递字符串需要将字符串直接作为属性值进行传递，如果是要传递其他数据类型，需要将值放入到{}中进行传递

- 传递字符串

```
<MyComponent foo='this is foo' />
```

- 传递数字

```
<MyComponent foo={1,2,3} />   3
```

- 传递布尔类型

```
<MyComponent foo={true} />
```

- 传递数组类型

```
<MyComponent foo={[1,2,3,4]} />  1234
```

- 传递对象

```
<MyComponent foo={{name: 'terry' }} />
```

Fragment

Fragments 允许将子列表分组，而无需向 DOM 添加额外节点。

```
class Table extends React.Component {
  render() {
       return (<table><tr><Columns /></tr></table>);
  }
}
```

如果Columns为如下代码则无效，因为在tr标签中不能插入div标签

```
class Columns extends React.Component {
  render() {
       return ( <div> <td>Hello</td> <td>World</td> </div> );
  }
}
```

杰普软件科技有限公司
www.briup.com

电邮：training@briup.com
主页：http://www.briup.com

昆山地址：昆山市巴城镇学院路828号昆山浦东软件园北楼4、5、8层
邮编：215311
电话：0512-50190298

上海地址：上海市闸北区万荣路1188弄G栋102室-上海服务外包科技园龙软园区
邮编：200436
电话：021-56778147

Briup High-End IT Training

第 2 章：组件基础

Brighten Your Way And Raise You Up.

学习目标

- 掌握组件的类创建方式
- 掌握react的内部状态
- 掌握react的生命周期
- 掌握react事件机制

组件创建方式

◆ 类组件

可以通过**ES6**中的类来创建组件，该类继承**React.Component**，并且拥有一个render()函数，该函数的作用于函数组件的那个函数一样，用于返回一个JSX。

```
class Clock extends React.Component {
  // render函数表示渲染，每次重新渲染都要调用该函数
  render(){
    return (
      <div>
        <h2>Clock, {this.props.name}</h2>
      </div>
    );
  }
}
```

组件局部状态

state

是组件内部维护的一组用于反映组件UI变化的状态集合。state需要在构造函数中进行初始化，如果要在组件类中重写构造函数，那么需要在构造函数的第一行显式调用super(props)

```
class Clock extends React.Component {
  constructor(props){
    super(props);
    this.state = {
      now:new Date().toLocaleString()
    }
  }
}
```

组件局部状态

◆ state特点

- 不能直接修改state

需要调用this.setState()方法进行修改

修改的时候要注意：

1. 状态的类型是不可变类型：直接赋新值
2. 状态的类型是数组：通过concat方法或者ES6数组拓展语法，不能使用在原数组的基础上修改的方法，例如push之类
3. 状态的类型是普通对象：通过Object.assgin或者对象拓展语法

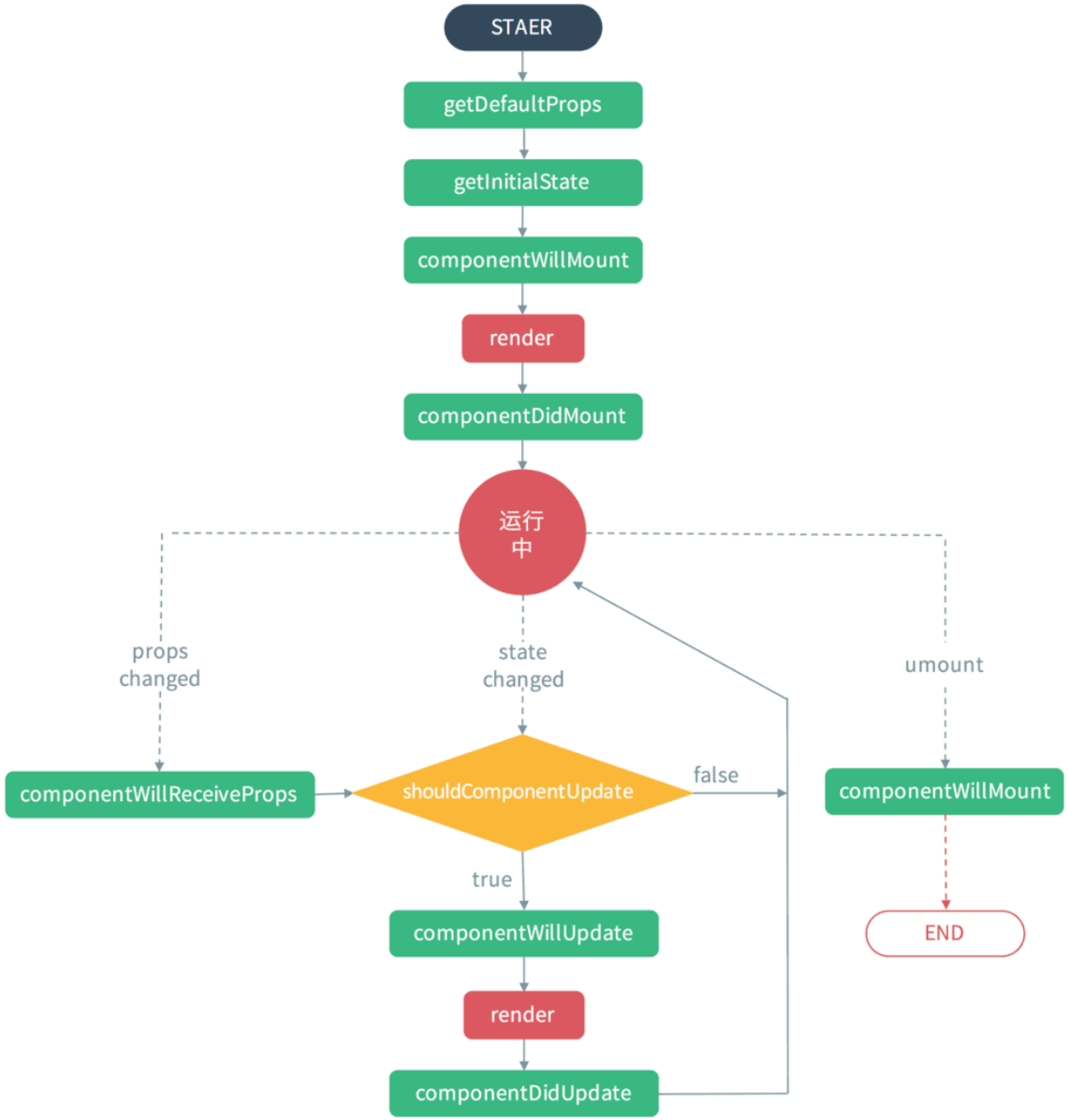
- state的更新是异步的

调用setState，组件的state并不会立即改变，

- state的更新是一个浅合并的过程。

组件的生命周期

◆ 生命周期



组件的生命周期

◆ 生命周期

- componentWillMount

在组件将要被挂载前调用

- componentDidMount

在组件被挂载之后立即调用，可以进行初始化网络请求，如果调用setState，然后可以再次渲染，但是这次渲染会发生在浏览器更新屏幕之前，用户不会发现中间状态。

- shouldComponentUpdate

组件是否被更新，如果返回true则更新，否则不更新

- componentWillUpdate【UNSAFE_componentWillUpdate】

组件将要被更新

- componentDidUpdate

组件被更新

- componentWillUnmount

在组件将要被卸载的时候调用

◆ 事件绑定

与vue绑定方式类似，需要在组件上通过’onXxx’来绑定事件，事件处理函数必须在大括号内通过this来指定。事件处理函数应该定义在类中，与render()函数在同一级别

```
class MyComponent extends React.Component{
  handleClick(){
    alert('handleClick...');
  }
  render(){
    return (
      <div onClick={this.handleClick}>hello MyComponent</div>
    )
  }
}
```

◆ 事件对象

与DOM中的事件类似，通过事件对象获取目标元素，默认情况下，可以通过在事件处理函数上声明形式参数来获取event。你可以在调用event.preventDefault()来阻止默认行为，调用event.target获取目标元素，调用event.cancelable阻止事件冒泡

```
class MyComponent extends React.Component{
  handleClick(event){
    alert('handleClick...');
  }
  render(){
    return (
      <div onClick={this.handleClick}>hello MyComponent</div>
    )
  }
}
```

this指针

如果通过es6的函数声明方式来定义事件处理函数，那么在事件处理函数中的this为undefined。

我们可以事件箭头函数来定义事件处理函数，这时候箭头函数中的this指向组件对象。

```
class MyComponent extends React.Component{
  handleClick = (event)=>{
    alert('handleClick...');
    console.log(this);
  }
  render(){
    return (
      <div onClick={this.handleClick}>hello MyComponent</div>
    )
  }
}
```

◆ 参数传递方式1

如果想要传递参数给事件处理函数，需要在事件绑定的时候调用bind方法，然后将this作为第一个实参，其他的参数可以自定义。但是要注意，在事件处理函数中，第一个参数为你绑定的第二个形参，... ，最后一个参数为event对象。

```
class MyComponent extends React.Component{
  handleClick = (param, event)=>{
    alert('handleClick...');
    console.log(this);
  }
  render(){
    return (
      <div onClick={this.handleClick.bind(this,1)}>hello MyComponent</div>
    )
  }
}
```

◆ 参数传递方式2

也可以使用箭头函数传参。

```
class MyComponent extends React.Component{
  handleClick = (p1,p2,event)=>{
    alert('handleClick...');
    console.log(event);
  }
  render(){
    return (
      <div onClick={(e)=>this.handleClick(1,2,e)}>hello MyComponent</div>
    )
  }
}
```

◆ 介绍

React提供的这个ref属性，表示为对组件真正实例的引用，其实就是ReactDOM.render()返回的组件实例。ref属性可以挂载到组件上也可以是dom元素上

ref 属性接受一个回调函数，它在组件被加载或卸载时会立即执行,回调函数的参数为该组件的真正实例的引用

ref属性接受一个字符串,例如foo，通过this.refs.foo来访问该组件真正实例

```
<CourseForm ref={this.courseFormRefs}/>
<CourseForm ref='courseForm'/>
<div ref="hello">hello</div>
<div ref="world">world</div>

componentDidMount(){ console.log("refs",this.refs); }
courseFormRefs(p1){console.log("ref参数",p1);  }
```

样式

- 在jsx中添加样式

```
<div style={{color:'red'}} onClick={(e)=>this.handleToLogOut(e,1)}>已登录</div>;
```

杰普软件科技有限公司
www.briup.com

电邮：training@briup.com
主页：http://www.briup.com

昆山地址：昆山市巴城镇学院路828号昆山浦东软件园北楼4、5、8层
邮编：215311
电话：0512-50190298

上海地址：上海市闸北区万荣路1188弄G栋102室-上海服务外包科技园龙软园区
邮编：200436
电话：021-56778147

Briup High-End IT Training

第 3 章：

双向数据绑定、路由

Brighten Your Way And Raise You Up.

学习目标

- 掌握表单元素的双向数据绑定
- 掌握react路由机制

◆ 表单

表单是接收用户输入最重要的方式之一，之前的做法是通过dom操作获取表单元素，然后获取表单元素的值，进而封装成参数对象与后端进行交互。在 HTML 中，表单元素（如<input>、<textarea> 和 <select>）之类的表单元素通常自己维护state，并根据用户输入进行更新。而在 React 中，可变状态（mutable state）通常保存在组件的 state 属性中，并且只能通过使用 setState()来更新。我们可以把两者结合起来，使 React 的 state 成为“唯一数据源”。渲染表单的 React 组件还控制着用户输入过程中表单发生的操作。被 React 以这种方式控制取值的表单输入元素就叫做“受控组件”。

- input的双向数据绑定

单行输入框、密码框、多行文本框

```
this.state = {username:'', password:'', description:''};
handleChange = (event)=>{
  this.setState({username:event.target.value})
}
用户名 <input type="text" name="username" value={this.state.username}
onChange={this.handleChange}/> <br/>
密码 <input type="password" name="password" value={this.state.password}
onChange={this.handleChange}/> <br/>
介绍 <textarea name="description" value={this.state.description}
onChange={this.handleChange}></textarea> <br/>
```

- input的双向数据绑定

  单选按钮、复选按钮

```
this.state = {gender:'男'};
handleChange = (event)=>{
  this.setState({gender:event.target.value})
}
性别 <input type="radio" name="gender" value="男" onChange={this.handleChange}
checked={this.state.gender==="男"?true:false}/>男
<input type="radio" name="gender" value="女" onChange={this.handleChange}
checked={this.state.gender==="女"?true:false}/>女 <br/>
```

◆ select下拉菜单

```
this.state = {address :'山西'};
handleChange = (event)=>{
  this.setState({address:event.target.value})
}
地址:
<select name="address"
  value={this.state.address}
  onChange={this.handleChange}>
  <option value="1">江苏</option>
  <option value="2">山西</option>
  <option value="3">河南</option>
  <option value="4">陕西</option>
</select>
```

路由机制

◆ React路由机制

react-router：实现了路由的核心机制，Switch、Router、Route

react-router-dom：基于react-router，加入了在浏览器运行环境下的一些功能，例如：Link组件，会渲染一个a标签，Link组件源码a标签行；BrowserRouter和HashRouter组件，前者使用pushState和popState事件构建路由，后者使用window.location.hash和hashchange事件构建路由。

react-router-native：基于react-router，类似react-router-dom，加入了react-native运行环境下的一些功能。

路由机制

◆ react-router-dom

基于浏览器环境的开发，只需要安装react-router-dom

```
import {BrowserRouter, Route,Link,Switch} from 'react-router-dom’
function App() {
  return (
   <div>
     <BrowserRouter>
       <Link to="/form">form</Link> <br/>
       <Link to="/clock">clock</Link>
          <Switch>
         <Route path="/form" component={Form}></Route>
         <Route path="/clock" component={Clock}></Route>
       </Switch>
     </BrowserRouter>
   </div>
);}
```

路由机制

◆ BrowserRouter

使用HTML5提供的history API(pushState, replaceState 和 popstate 事件)来保持UI和URL的同步。

路由地址格式：

```
http://localhost:3000/student
```

路由机制

◆ HashRouter

使用URL的hash部分（即 window.location.hash）来保持UI和URL的同步。

路由地址格式：

http://localhost:3000/#/student

- Link

为应用提供声明式的、可访问的导航链接。

路由机制

◆ Route

是React Router中最重要的组件，它可以帮助你理解和学习如何更好的使用React Router。它最基本的职责是在其 path 属性与某个 location 匹配时呈现一些 UI。

路由机制

Switch

用于渲染与路径匹配的第一个子〈Route〉或〈Redirect〉

路由机制

◆ API跳转

在组件中通过this.props.history控制路由的改变

```
this.props.history.push('/content')
```
将新的路径压入到history中

```
this.props.history.push({
   pathname:'/studentDetails',
   payload:record
})
```

这种方式跳转可以通过 this.props.location.payload来获取传递的参数record

```
this.props.history.go(n)
```
n为正数或者负数，表示前进或者后退

```
this.props.history.goBack()
```
后退

```
this.props.history.goForward()
```
前进

杰普软件科技有限公司
www.briup.com

电邮：training@briup.com
主页：http://www.briup.com

昆山地址：昆山市巴城镇学院路828号昆山浦东软件园北楼4、5、8层
邮编：215311
电话：0512-50190298

上海地址：上海市闸北区万荣路1188弄G栋102室-上海服务外包科技园龙软园区
邮编：200436
电话：021-56778147

Briup High-End IT Training

第 4 章:组件库

Brighten Your Way And Raise You Up.

学习目标

- ant Design介绍
- ant Design安装

◆ 介绍

antd 是基于 Ant Design 设计体系的 React UI 组件库，主要用于研发企业级中后台产品。

特点：

1. 提炼自企业级中后台产品的交互语言和视觉风格。
2. 开箱即用的高质量 React 组件。
3. 使用 TypeScript 构建，提供完整的类型定义文件。
4. 全链路开发和设计工具体系

支持环境：

1. 现代浏览器和 IE9 及以上（需要 polyfills）。
2. 支持服务端渲染。
3. Electron（桌面版）

安装

可以使用npm或者yarn来安装。

```
$ npm install antd --save
```

或者

```
$ yarn add antd
```

◆ 配置

我们需要对 create-react-app 的默认配置进行自定义，这里我们使用 react-app-rewired （一个对 create-react-app 进行自定义配置的社区解决方案）。引入 react-app-rewired 并修改 package.json 里的启动配置。由于新的 react-app-rewired@2.x 版本的关系，你还需要安装 customize-cra。

1.安装react-app-rewired、customize-cra

```
$ yarn add react-app-rewired customize-cra
```

2.配置package.json

```
/* package.json */
"scripts": {
      -   "start": "react-scripts start",
      +   "start": "react-app-rewired start",
      -   "build": "react-scripts build",
      +   "build": "react-app-rewired build",
      -   "test": "react-scripts test",
      +   "test": "react-app-rewired test",
}
```

◆ 配置

3.安装babel-plugin-import

babel-plugin-import 是一个用于按需加载组件代码和样式的 babel 插件

```
$ yarn add babel-plugin-import
```

4.配置config-overrides.js

```
/* 在项目根目录中创建文件config-overrides.js */
const { override, fixBabelImports } = require('customize-cra');
module.exports = override(
        fixBabelImports('import', {
                libraryName: 'antd',
                libraryDirectory: 'es',
                style: 'css',
        }),
)
```

◆ 配置

5. 自定义主题

安装less与less-loader

```
$ yarn add less less-loader
```

修改config-overrides.js

```
/* config-overrides.js */
const { override, fixBabelImports, addLessLoader } = require('customize-cra');
module.exports = override(
        fixBabelImports('import', {
                libraryName: 'antd',
                libraryDirectory: 'es',
                style: true,
        }),
        addLessLoader({
                javascriptEnabled: true,
                modifyVars: { '@primary-color': 'teal' },
        }),
)
```

◆24栅格系统

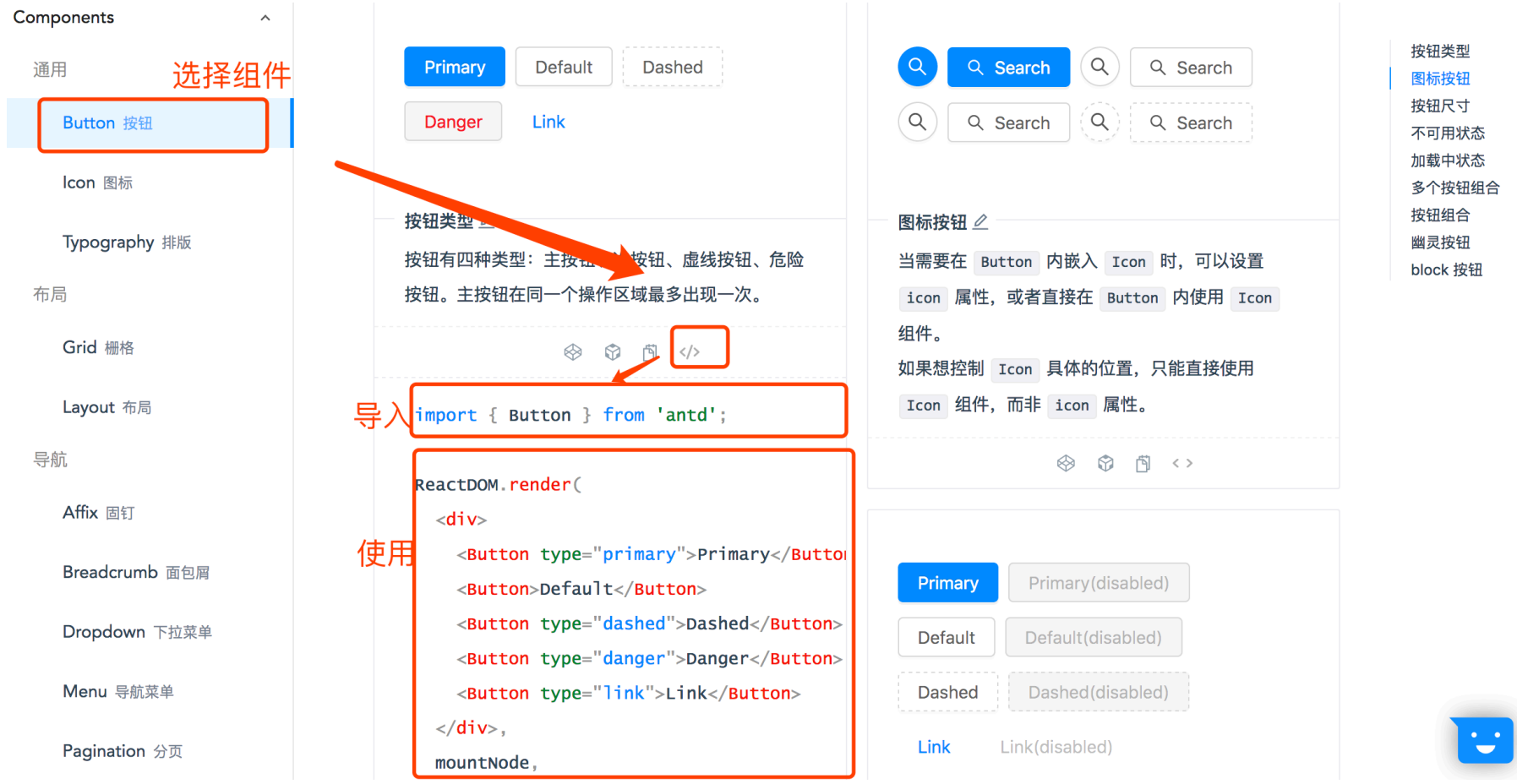
布局的栅格化系统，我们是基于行（row）和列（col）来定义信息区块的外部框架，以保证页面的每个区域能够稳健地排布起来。下面简单介绍一下它的工作原理：

- ✓ 通过row在水平方向建立一组column（简写 col）
- ✓ 内容应当放置于col内，并且，只有col可以作为row的直接元素
- ✓ 栅格系统中的列是指 1 到 24 的值来表示其跨越的范围。例如，三个等宽的列可以使用.col-8来创建
- ✓ 如果一个row中的col总和超过 24，那么多余的col会作为一个整体另起一行排列

注意：具体的属性可以查看官方文档

◆学习方法

使用antd最简单的方式就是访问其官方文档



表单

表单一般单独封装在一个组件中

```
const { getFieldDecorator } = this.props.form;
<Form onSubmit={this.handleSubmit} className="login-form">
  <Form.Item>
      ({getFieldDecorator('username', {rules: [{ required: true, message: 'Please input
your username!' }],
      })(
        <Input prefix={<Icon type="user" style={{ color: 'rgba(0,0,0,.25)' }} />}
placeholder="Username"/>,
      )}
  </Form.Item>
  <Form.Item>
      <Button type="primary"
        htmlType=“submit” className=“login-form-button”>Log in </Button>
  </Form.Item>
</Form>
export default Form.create()(StudentForm);

handleSubmit = (event)=>{
  event.preventDefault();
  this.props.form.validateFields((err, values) => {
      if (!err) { console.log('Received values of form: ', values);}
  });
}
```

◆ 表单 Form.create(参数)()

参数：

mapPropsToFields　　把父组件的属性映射到表单项上（如：把 Redux store 中的值读出），需要对返回值中的表单域数据用 Form.createFormField 标记

onValuesChange　　任一表单域的值发生改变时的回调

…其余参数见官方文档

经过 Form.create 包装的组件将会自带 this.props.form 属性，this.props.form 提供的 API

getFieldDecorator　　用于和表单进行双向绑定

validateFields　　校验并获取一组输入域的值与 Error，若 fieldNames 参数为空，则校验全部组件

setFieldsValue

…其余见官方文档

杰普软件科技有限公司
www.briup.com

电邮：training@briup.com
主页：http://www.briup.com

昆山地址：昆山市巴城镇学院路828号昆山浦东软件园北楼4、5、8层
邮编：215311
电话：0512-50190298

上海地址：上海市闸北区万荣路1188弄G栋102室-上海服务外包科技园龙软园区
邮编：200436
电话：021-56778147

Briup High-End IT Training

第 5 章:axios

Brighten Your Way And Raise You Up.

学习目标

- axios的安装与配置
- axios的核心api
- axios的速写形式
- axios的拦截器

介绍

axios是使用promise机制封装的ajax，可以在浏览器和nodejs使用。相对比jquery来说axios是更加纯粹的ajax的框架

安装

```
$ yarn add axios
```

示例

```
axios.get('/user?ID=12345')
.then(function (response) {
        console.log(response);
})
.catch(function (error) {
        console.log(error);
});
```

默认配置

通过默认配置可以使得所有axios都具有某些特性。

```
axios.defaults.baseURL = 'https://api.example.com';
axios.defaults.headers.common['Authorization'] = AUTH_TOKEN;
axios.defaults.headers.post['Content-Type'] = 'application/x-www-form-urlencoded';
```

◆ 参数

通过参数的设定可以改变axios的请求行为，常见的参数如下

```
url: '/user',
method: 'get', // default
baseURL: 'https://some-domain.com/api/',
responseType: 'json', // default
headers: {'X-Requested-With': 'XMLHttpRequest'},
transformRequest: [function (data, headers) {return data;}],
transformResponse: [function (data) {return data; }],
params: {  ID: 12345 },
paramsSerializer: function(params) {
       return Qs.stringify(params, {arrayFormat: 'brackets'})  },
data: {firstName: 'Fred'},
withCredentials: false,
… 其余查看API
```

拦截器

可以在请求发送前或者是响应回来后添加拦截器

```
// Add a request interceptor
axios.interceptors.request.use(function (config) {
       // Do something before request is sent
       return config;
}, function (error) {
       // Do something with request error
       return Promise.reject(error);
});
// Add a response interceptor
axios.interceptors.response.use(function (response) {
    // Do something with response data
    return response;
}, function (error) {
    // Do something with response error
    return Promise.reject(error);
});
```

response格式

可以在请求发送前或者是响应回来后添加拦截器

```
{
        data: {},
        status: 200,
        statusText: 'OK',
        headers: {},        //服务器端返回的头部信息
        config: {},
        request: {}
}
```

快捷api

可以调用如下api完成ajax请求

```
axios#request(config)
axios#get(url[, config])
axios#delete(url[, config])
axios#head(url[, config])
axios#options(url[, config])
axios#post(url[, data[, config]])
axios#put(url[, data[, config]])
axios#patch(url[, data[, config]])
```

杰普软件科技有限公司
www.briup.com

电邮：training@briup.com
主页：http://www.briup.com

昆山地址：昆山市巴城镇学院路828号昆山浦东软件园北楼4、5、8层
邮编：215311
电话：0512-50190298

上海地址：上海市闸北区万荣路1188弄G栋102室-上海服务外包科技园龙软园区
邮编：200436
电话：021-56778147

Briup High-End IT Training

第 6 章:Redux

Brighten Your Way And Raise You Up.

学习目标

- ant Design介绍
- ant Design安装

介绍

使用Javascript开发单页面程序的时候，我们需要管理很多很多状态（state），例如：服务器的响应，本地生成的尚未持久化到服务器的数据（表单数据），激活的路由，选中的标签，是否显示加载动效或分页器等等。管理不断变化的状态非常困难，状态的改变可能会引起页面的变化，而页面的改变也会引起状态的改变，这种改变异常复杂，以至于我们很难捋清业务实现功能。为了规范的管理各种状态，我们可以使用状态管理机制。

安装

安装依赖redux、react-redux

```
$ yarn add redux react-redux
```

- model

模型，用于保存状态；注意：不能直接修改model中的值。

```
{
        todos: [{
            text: 'Eat food',
            completed: true
        }, {
            text: 'Exercise',
            completed: false
        }],
        visibilityFilter: 'SHOW_COMPLETED'
}
```

action

如果想要修改model中的值，必须发起一个action，这里需要注意的是Action实际上是一个普通的Javascript对象。使用action来修改state的好处在于可以很清楚的描述发生了什么事情，例如type取值为'ADD_TODO'，那我们就明白要添加TODO，text值为TODO的内容。

```
{ type: 'ADD_TODO', text: 'Go to swimming pool' }
{ type: 'TOGGLE_TODO', index: 1 }
{ type: 'SET_VISIBILITY_FILTER', filter: 'SHOW_ALL' }
```

action是数据从应用传递到store中的有效载荷。它是store数据的唯一来源！一般来说你会通过 store.dispatch() 将 action 传到 store。Action本质上是一个对象，但是我们约定，action 内必须使用一个字符串类型的 type 字段来表示将要执行的动作。除了 type 字段外，action 对象的结构完全由你自己决定。

action 创建函数

action创建函数，注意，action创建函数是用来创建action的函数，而action是一个普通JavaScript对象。在 Redux 中的 action 创建函数只是简单的返回一个 action。

```
const ADD_TODO = 'ADD_TODO';
function addTodo(text) {
        return { type: ADD_TODO, text }
}
```

如果我们想要发起一个action就可以

```
store.dispatch({type: 'ADD_TODO', text: 'Go to swimming pool'})
//或者
store.dispatch(addTodo('Go to swimming pool'))
```

注意：默认情况下action不能是异步的！必须是同步的。action生成函数也必须是纯函数，只能返回一个对象，并且是同步返回。

Reducer

那么，如何把action与model关联起来呢？我们可以开发一些函数，这些函数用来接收state与action，并且返回state

```
function todos(state = [], action) {
  switch (action.type) {
      case 'ADD_TODO':
              return state.concat([{ text: action.text, completed: false }]);
      case 'TOGGLE_TODO':
              return state.map((todo, index) =>
                action.index === index ?
                        { text: todo.text, completed: !todo.completed } : todo )
      default: return state;
  }
}
```

使用方式

1. 定义reducer
2. 利用createStore生成store,如果是多个reducer,需要使用combineReducers先将reducer进行合并
3. 在index.js中为<App/>标签添加父标签<Provider>,并为其指定store属性，这样store就可以注入到整个项目中了