# day01

## 复习：

见思维导图

补充：Ionic项目

## Angular & Ionic 阶段项目

实现步骤：

(1)创建空白的Ionic项目模板

(2)创建项目必需的组件

ng g component main 盛放页签组件的页面

ng g component index 首页(1F/2F/3F)

ng g component product-list 商品列表页

ng g component cart 购物车页

ng g component login 登录页

ng g component not-found 404页

ng g component product-detail 商品详情页

确保每个组件都在app.module.ts中进行了声明：declarations

(3)修改根组件模板app.component.html，使用MainComponent(其中要创建页签组件)

<ion-app>

<ion-router-outlet></ion-router-outlet>

<app-main></app-main>

<ion-app>

(4)编辑main.component.html，声明页签组件

<ion-tabs>

<ion-tab-bar>

<ion-tab-button tab="路由地址">图标+文字</ion-tab-button>

</ion-tab-bar>

</ion-tabs>

(5)编辑路由配置模块文件app-routing.module.ts，声明并注册路由词典

const routes = [

{path: '', component: IndexComponent},

{path: 'product-list', component: ProductListComponent},

{path: 'product-detail/:pid', component: ProductDetailComponent},

{path: 'cart', component: CartComponent},

{path: 'login', component: LoginComponent},

{path: '\*\*', component: NotFoundComponent},

];

此时可以使用客户端进行测试，访问页签中的每个按钮，可以在路由出口显示对应路由组件页面

-------------------------------------------------------------------------------

(6)开始修改主页组件模板：index.component.html，添加页面内容

<ion-header>

img + ion-searchbar

</ion-header>

<ion-content>

slides + ion-item + ion-grid>ion-card + .....

</ion-content>

模型数据：carouselItems、recommendedITem、topSaleItems、newArrialItem、mySlides、http

--------------------------------------------------------------------------------

(7)修改商品列表页模板：product-list.component.html，添加页面内容

<ion-header>

ion-button + ion-searchbar

</ion-header>

<ion-content>

ion-card>ion-item + ion-infinite-scroll + ion-button

</ion-content>

模型数据：productList、pno、hasMore、http

--------------------------------------------------------------------

(7)修改商品详情页模板：product-detail.component.html，添加页面内容

<ion-header>

ion-button + ion-title + ion-button

</ion-header>

<ion-content>

ion-item + ion-slides + ion-card > ion-item

</ion-content>

模型数据：pid、product、http、mySlides

|  |
| --- |
| 面试题：  (1)解释跨域请求？  (2)跨域请求解决方案？ ①JSONP ②CORS ③服务器端代理 |

|  |
| --- |
| HTML元素属性(Attribute) 和 JS DOM对象的属性(Property)  ①元素属性和对象属性同名：  <img src="xxx">  document.createElement('img').src="xxx"  ②元素属性和对象属性不同名：  <input class="box" tabindex="3">  document.createElement('box').className="box"  document.createElement('box').tabIndex="3"  ③元素属性在对象属性中不存在：  <td colspan="3">  document.createElement('td').~~colSpan~~="box"  ④对象属性在元素属性中不存在  document.createElement('div').innerHTML = 'abcd'  <div ~~innerHTML~~=""> |

Vue.js Angular React

## 1.React

由Facebook维护的用于构建用户界面的 JavaScript 库，是三个框架中最简单的；但是React的生态圈很大：

React：核心

React-Router：路由

Redux：状态管理

React Native：移动App

ReactVR/React360：VR开发

使用React有两种方式：

(1)SCRIPT脚本引入式：可以在任意单一HTML页面使用

(2)React脚手架方式：适合于创建大型Web应用/SPA应用

练习：创建一个myp02.html页面，容器#box

使用React在#box中渲染如下内容：

<p class="danger">

AA

<span class="strong">BB</span>

CC

</p>

## 2.React核心概念之一 —— JSX

JavaScript XML：由React定义的融合了XML语法的JS代码——本质还是JS，只是语法外观上像XML！

浏览器不能直接运行JSX语法！需要使用JSX编译器，转换为浏览器可以执行的JS语法： Babel —— 第三方提供的JS编译器！

在页面中运行Babel解释器：

<script src="js/babel.js"></script>

<script type="text/babel">

**//JSX语句**

</script>

JSX语法：

①外观像是XML(标签必须闭合，属性值必须用引号，根元素有且只有一个)

②JSX本质不是标签，而是在创建JS对象；属性都是JS DOM属性，而不是HTML标签属性，如class必须写作className、for必须写作htmlFor

③JSX中使用的HTML标签都必须用纯小写形式！自定义标签都必须采用大驼峰命名法！

④JSX中若遇到 < 则会运行HTML解释器；遇到 { 则会运行JS解释器

⑤JSX中的表达式语法： { express }，其中的表达式可以是：

变量名？ 算术运算？ 比较运算(无错但也无输出)？ 逻辑运算(无错但也无输出)？ 三目运算？ 调用函数？ 调用对象的方法？ 创建新的对象（但可以创建完对象立即调用其成员）？ **选择语句(但可以嵌入在匿名自调函数中——唯一的选择语句写法)？ 循环语句(普通的for不行但可以使用arr.map(fn))？**

|  |
| --- |
| 提示：React JSX中没有类似Angular中的ngIf和ngFor指令（**React中没有指令的概念**），在JSX中可以在 { } 执行函数以返回 if..else..的不同结果，也可以在 { } 执行 arr.map( fn )已返回每个元素遍历后返回的处理结果 |

练习：创建一个myp04.html页面，容器#box

使用React在#box中渲染如下内容：

<form action="do.php">

<label>用户名：</label><input name="uname">

<label>密码名：</label><input name="upwd">

<button type="submit">提交</button>

</form>

练习：创建一个myp05.html页面，容器#box

创建变量 hasMore，值为true就在#box渲染一个：

<div>加载中，请稍候...</div>

值为false渲染一个：

<button>没有更多数据了</button>

练习：创建一个myp06.html页面，容器#box

创建变量 productList，其中保存四个商品的名称，

在页面中渲染一个ul>li\*4呈现数组中的数据

## 3.React核心概念之二 —— 组件

Component：是一段可复用的页面内容

React中声明组件的语法：

**class MyComponent extends React.Component{**

**render(){**

**return (**

**//JSX表达式**

**)**

**}**

**}**

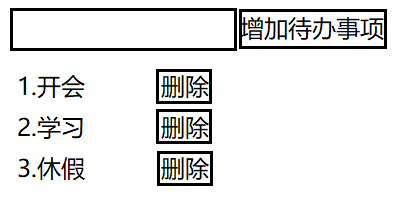
**ReactDOM.render( <MyComponent />, box )**

提示：①React中的自定义组件必须继承自React.Component ②自定义组件中使用render()方法返回组件的模板 ③自定义组件名首字母必须大写(推荐全驼峰法则)；而JSX中的html标签必须使用纯小写形式

## 课后任务：

(1)继续完善“学子商城 Ionic 版”

(2)自定义组件 TodoList，实现“待办事项列表”



提示：事件绑定、组件状态数据绑定需要阅读参考手册完成

# day02

## 复习：

React是由Facebook维护的一个用于构建UI界面库，核心更加简单，生态圈很丰富(React、React-Router、Redux、React Native、React360等 —— Github：awesome react)。

使用React的两种方法：

(1)SCRIPT脚本引入式

(2)React CLI脚手架方式

|  |
| --- |
| 面试题：请问三大框架是如何提高DOM操作效率的？  Vue.js：Virtual DOM，虚拟DOM  Angular：Dirty Check，脏检查  React：Virtual DOM，虚拟DOM |
| React中所有的DOM操作底层：都不是直接修改DOM树！而是React在内存中保存了一份DOM树副本——称为虚拟DOM树，所有的render操作都是针对虚拟DOM树进行；只要虚拟DOM树发生改变，就会执行 DIFF 算法，把虚拟DOM与真实DOM之间的不同点绘制到真实DOM——**用空间(更多的内存)换时间(DOM效率提高)** |

## 1.React核心概念之一 —— JSX

let element = React.createElement('div',{},'hello')

let element = <div className="">hello</div>

项目中可以不使用JSX，但是使用了会更加简单！

JSX：JavaScriptXML，本质是JS！语法采用了XML语法，使用需要注意：

①标签必须结束！

②HTML标签名必须纯小写；自定义标签名必须首字母大写

③标签的属性都是JS DOM元素的Property，不是HTML元素的Attribute，所以class必须className

④一段JSX必须有且只有一个根元素

⑤一段JSX可以用JS中的 ( ) 括起来

⑥JSX中可以使用表达式来进行运算

{ uname }

{ 2+3 }

{ n1>n2 ? '是' : '否' }

{ sum(1, 2) }

{ (function(){})() }

{ new Date().getTime() }

## 2. React核心概念之二 —— Component

组件：就是一段可以反复使用的页面片段；

组件 = 脚本 + 模板 + 样式

class MyC01 extends React.Component{

render(){

//return React.createElement()

//return JSX

}

}

//使用自定义组件

<MyC01></MyC01>

|  |
| --- |
| 提示：React手册中提供了两种创建自定义组件的方式  (1)function式：简单功能少  function MyButton(){ }  (2)class式：复杂功能多  class MyButton extends React.Component{ } |

## 3.组件中的事件绑定

React组件中的事件绑定语法是React所特有的！

①事件名必须 onXxx

②事件监听必须是一个 function，不能是函数调用，不能加( )

③事件监听函数必须用{ }括起来，不能加双引号

④事件处理函数默认有一个实参：Event对象

<button onClick={ this.goBack }>

|  |
| --- |
| React中事件处理函数中无法获取this的三种解决方法：  方法1：增加匿名函数  onClick={ ()=>{ this.goBack( ) } }  方法2：构造方法中绑定this的指向  constructor(){  super( )  this.goBack = this.goBack.bind(this)  }  方法3：监听函数赋值为对象的属性，属性值是一个“箭头函数”  goBack = ()=>{  //箭头函数中的this指向不会改变  } |

练习：在#box渲染两个按钮，“登录”和“注册”，两个单击事件监听方法只需要输出this即可

练习：自定义组件MyInput, 渲染一个输入框，提示用户输入“搜索关键词”，试着监听其“内容改变”事件，并输出最新的输入内容——**提示：React中没有v-model/ngModel指令 ，监听表单元素值改变只能监听onChange**

## 4.组件中的属性/数据绑定——React核心概念之三 —— state

提示：React中绑定的数据不能简单的是class属性！其采用的策略类似于Vuex——数据保存在状态中(state)

①创建数据状态

this.state = { age:20 } //只能在构造方法中创建

②绑定数据状态

this.state.age

③修改数据状态

//this.state.age = 30; //无效的修改

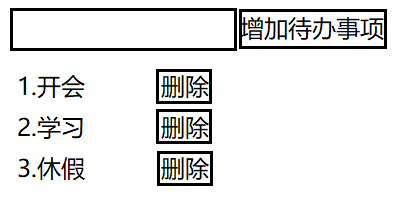
this.setState({age: 30})

提示：setState( obj, fn )方法是“异步”执行的——不会立即执行；如果想在状态数据修改完毕后执行特定的处理，可以使用第二个回调函数参数

练习：自定义组件MyBuyCount, 渲染如下组件内容：



练习：自定义组件MyTodoList, 渲染如下组件内容：



## 5.组件中的选择/循环

## 6.组件间数据传递——React核心概念之四 —— props

每个组件都可以声明自定义属性（Angular中称为“输入属性”）

**<MyInputGroup labelTxt="xx" inputId="yy"/>**

React中组件可以直接使用this.props (Property)读取自己接收到父组件给与的全部输入属性值（自定义属性无需提前声明）。例如：this.props.labelTxt；子组件也可以通过this.props读取父组件传给自己的处理函数，用于把子组件中的数据传递给父组件加以使用

**<MyInputGroup doInputChange={this.doUnameChange}/>**

|  |
| --- |
| React父子组件间数据传递：  (1)父 -> 子： Props Down  (2)子 -> 父： Props Up |

## 7.React核心概念之五 —— ref / refs

Angular组件脚本查找子组件使用的是@ViewChild

Vue.js/React组件脚本中查找子组件都使用“对象引用”： ref

<any ref="myChild01">

this.refs.myChild01.... 此对象及为组件/元素对象

## 8.React核心概念之六 —— 受控组件

由React所渲染的所有 input、select、textarea 表单元素，都做了特殊的处理，称为“受控组件”——只要提供了value但没有提供onChange事件处理方法，输入项都是只读的；特点如下：

①只要 value 属性赋值为一个常量，此值就无法由用户修改

②只要 value 属性赋值为一个状态变量，但未提供onChange处理方法，此值就无法由用户修改

③受控组件要求用户必须监听onChange事件，在其中对用户输入进行检查，合格的数据最终要赋值为 state 数据，从而影响输入框的值

## 课后任务：

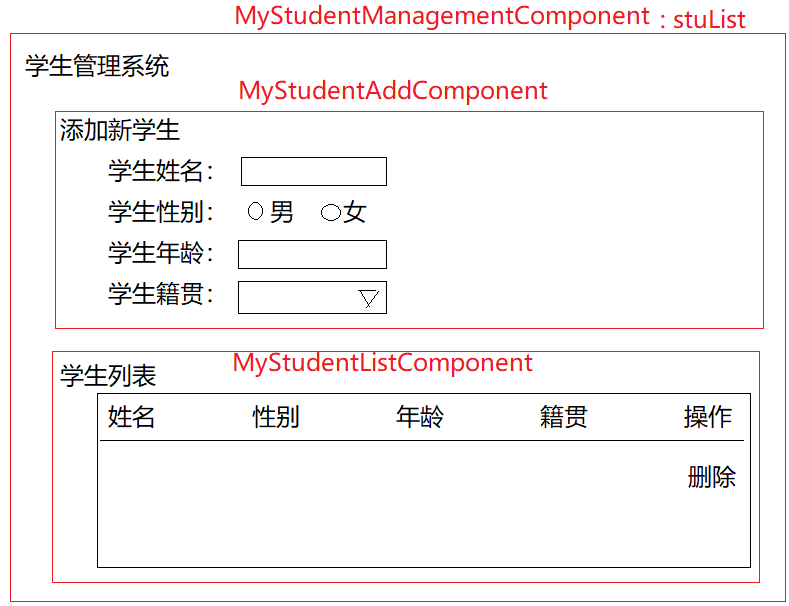
(1)整理React核心概念思维导图：

JSX / Component / state / props / ref / 受控组件

(2)自学：React组件的生命周期（注意废弃的钩子函数）

https://segmentfault.com/a/1190000015025236

(3)使用React完成“学生管理系统”



# day03

## 复习：见思维导图

|  |
| --- |
| NPX工具可以实现的功能：  http://www.ruanyifeng.com/blog/2019/02/npx.html |

## 1.React的两种使用方式

方式1：SCRIPT脚本引入方式

方式2：React CLI脚手架项目方式

①下载React CLI脚手架工具

npm i -g create-react-app

②进入项目仓库目录，运行React脚手架创建一个空白项目

create-react-app myreactapp03

③进入项目根目录，运行此脚手架项目

cd myreactapp03

npm start

|  |
| --- |
| 异步加载数据的时间点：  Vue.js：mounted(){ }  Angular： ngOnInit(){ }  React：componentDidMount( ){ } |

## 2.React中组件的生命周期钩子函数

constructor( )

**阶段一：加载阶段**

componentWillMount( )：组件已创建，即将加载(虚拟DOM与实际不同)

render()

**componentDidMount( )**：组件已经加载完成——适合于加载异步数据

**阶段二：更新阶段**

componentWillReceiveProps( )：(已废弃)组件即将接收到属性数据

shouldComponentUpdate( )：组件应该更新吗？返回true即调用render，否则不调用

componentWillUpdate( )：组件即将更新

render()

componentDidUpdate( )：组件更新完成

**阶段三：卸载阶段**

componentWillUnmount( )：组件即将卸载，用于清除组件数据，如定时器等

|  |
| --- |
| Vue.js： Vue-Resource 、 Axios  Angular： HttpClient  React： 官方未提供 |
| 面试题：常用的发起异步请求的方法有哪些？  ①原生 XHR  ②jQuery.ajax( )：基于XHR，采用回调  ③Axios：基于XHR，采用Promise  ④HttpClient：基于XHR，采用Observable  ⑤Fetch：由W3C提供的XHR的替代方案，采用Promise |

## 3.React中发起异步请求方法

fetch不是对xhr的封装，而是w3c提供的一种新技术，其原生就是基于Promise的，未来有望取代xhr。

fetch获取异步JSON数据的方法：

fetch(url, {}).then( (res)=>{

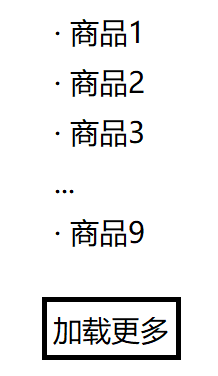
return res.json()

}).then( (data)=>{

//data就是响应消息内容

})

练习：创建组件MyProductList, 组件加载完成(componentDidMount)立即获取(fetch)第一页商品数据；点击“加载更多”再加载下一页数据



|  |
| --- |
| 移动端App的三种形式：  **(1)NativeApp (原生App)**：用移动操作系统本身就可以运行的编程语言创建的App，优势：速度快/功能丰富 不足：不跨平台  iOS：ObjectiveC/Swfit； Android：Java/Kotlin  **(2)WebApp(H5)**：用标准Web(HTML/CSS/JS)编写的App，运行于HTML/JS解释器中(WebView对象-浏览器核心)，优势：跨平台 不足：效率低/功能欠佳——MintUI  **(3)HybridApp(混编App)**：用标准Web(HTML/CSS/JS)编写的App，再配合上底层原生代码(OC/Java)做的驱动程序，可以用JS调用手机底层功能 优势：跨平台/功能比WebApp丰富 不足：效率低——Ionic |

## 4.React生态圈 —— React Native (RN)

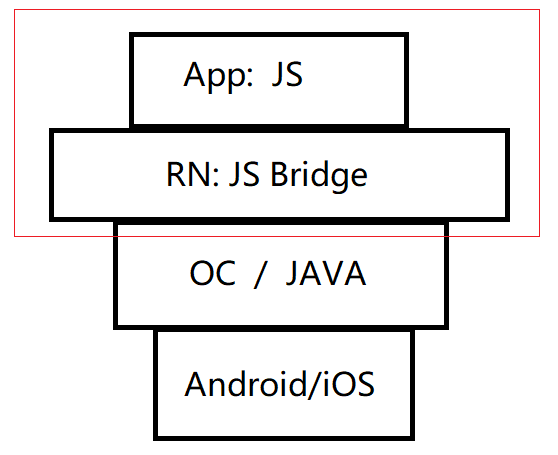
英文官网：https://facebook.github.io/react-native/

中文镜像：https://reactnative.cn/

RN：使用JS和React编写NativeApp（原生应用）

RN定义了一整套的React组件库，这些组件库使用JS语言来调用，运行时会被RN平台编译为OC或者Java代码

注意：RN技术应属于“NativeApp”范畴，不能使用HTML/CSS，所有的JS组件最终都会被编译为原生代码，直接运行在手机操作系统上——无需WebView核心！



RN开发环境的搭建：脱离浏览器，访问测试需要使用iOS/Android设备（要么是物理设备、要么是模拟器设备）：

https://reactnative.cn/docs/getting-started.html

简化版使用步骤：

(1)下载全局的RN脚手架工具

npm i -g react-native-cli

(2)进入项目仓库，运行RN脚手架创建项目

react-native init 项目名

(3)进入项目目录，启动该项目

cd 项目名

npm start

提示：上述三步创建的项目是纯JS中，如果想编译为Android/iOS程序还需要Android/iOS开发软件的支持（Android Studio、 Xcode）；编译后的安装程序：

项目名\android\app\build\outputs\apk\debug\app-debug.apk

## RN项目运行过程：

(1)把项目生成的App安装包安装到手机系统中

直接把app-debug.apk拖到夜神或蓝叠模拟器中就可以完成安装

(2)启动RN服务器端程序

进入RN项目，执行 npm start

启动的是一个Android项目打包和内容发布服务器，默认监听在8081端口上

注意查看服务器所在的IP地址(ipconfig)和端口号(8081)

(3)运行手机中的项目App，会访问RN服务器获取最新的App内容

注意如果提示无法访问，则需要配置服务器的IP地址和端口号，步骤：摇一摇 -> Dev Settings -> Debug Server Host & Port ...

## 今日任务：

(1)整理React知识点思维导图

(2)搭建出可以运行的RN项目运行环境（可以安装最新版本的夜神或蓝叠模拟器）

# day04

## 复习：

Vue.js —— MintUI（WebApp）

Angular —— Ionic（HybridApp）

React —— React Native（不用OC/Java的NativeApp）

提示：MintUI/Ionic最终都是HTML/CSS/JS，都要运行在“浏览器/WebView”中；RN代码语法是JS，但是没有HTML/CSS，JS组件最终会被编译为原生的OC/Java代码！—— RN的运行不依赖于浏览器/WebView！！RN应用中也不存在跨域请求问题！！

## RN项目的部署结构：

**服务器端： 172.163.100.239:8081**

npm i -g react-native-cli

react-native init 项目名

cd 项目名

npm start

**App客户端：**

安装调试用App

启动并配置App，指定服务器端地址和端口

重启App，即可自动连接服务器获取最新内容

后续想刷新记得“摇一摇>Reload”

|  |
| --- |
| ES6箭头函数：有{ }必须自己声明return才有返回值；没有{}相当于有默认的return语句，下面两句才是相等的：  ()=>{  return 1+2  }  ()=> 1+2 |

## 1.RN中常用的组件 —— 重点

注意：RN应用中不允许使用任何HTML标签——底层没有浏览器

## (1)Text——显示一段文字

用法：<Text style={} numberOfLines={3} ellipisizeMode='tail'>内容</Text>

注意：Text的父组件如果是Text则会继承样式，如果是其他组件就不会样式继承；推荐把View当前Text的父组件

## (2)View——一块视图区域，类似于HTML中的DIV

用法：<View>...</View>

注意：View默认最大只能铺满手机屏幕，当内容太多超出了屏幕则无法显示

## (3)Button——按钮

用法：<Button title="按钮上的文字" onPress={this.doPress} color="red" disabled={true}/>

## (4)Image——图片

用法1：加载手机本地图片

<Image source={require('./assets/x.jpg')}/>

无需指定宽和高

用法2：加载远程服务器图片

<Image source={{uri: 'http://xxx.com/x.jpg'}} style={{width:200, height: 100}} resizeMode="stretch" loadingIndicatorSource={require('./loading.gif')}/>

必需事先指定需要的宽和高！不指定的话0x0

## (5)ImageBackground——背景图片

使用：<ImageBackground source={require('...')}>

//将要显示在背景上方的内容

</ImageBackground>

说明：背景图是双标记标签；一般推荐背景图都使用本地图片，但背景图即使是本地图片，也必须声明width和height！

## (6)ScrollView——能滚动的视口

用法：<ScrollView>...</ScrollView>

注意：当ScrollView显示的内容超过了屏幕的尺寸，就开始进行水平/竖直的滚动

## (7)TextInput——文本输入框

用法：<TextInput value={this.state.uname} onChangeText={this.doChange}/>

提示：TextInput属于典型的“受控组件”，想获得用户的输入，必须经过三步：①在构造方法中声明状态数据 this.state = {uname:''} ②输入框的value绑定到状态数据 ③监听输入改变事件，处理方法中修改状态数据

## (8)Switch——开关

用法：<Switch value={this.state.isOnline} onValueChange={this.doChange}/>

提示：Switch属于典型的“受控组件”，想获得用户的输入，必须经过三步：①在构造方法中声明状态数据 this.state = {isOnline:false} ②输入框的value绑定到状态数据 ③监听输入改变事件，处理方法中修改状态数据

|  |
| --- |
| “JS远程调试”：console.log默认在手机的操作系统控制台输出，用户无法查看到——可以把手机控制台输出使用WebWorker导出到一个浏览器控制台： 摇一摇 > Debug JS Remotely |
| uri和url的关系： |

## 2.为RN组件添加样式

(1)行内样式

<Any style={ {color:'red'} }></Any>

(2)内部样式

let styles = StyleSheet.create({

danger: {

color: 'red'

}  
 })

<Any style={styles.danger}></Any>

## 课后任务：

(1)整理目前学过的所有RN组件

(2)自学RN手册中的组件： ActivityIndicator、 FlatList —— 模拟实现一个类似Ionic中的无限滚动组件：MyInfiniteScroll

(3)自学RN手册中的工具对象：Alert

# day05

## 复习：

见思维导图

## 1.RN常用组件

**(1)ActivityIndicator**——活动提示器，“加载中”提示符号

用法：<ActivityIndicator size="large" color="gray" animating={true}/>

提示：指示器只要“动画停止”，默认就会隐藏

练习：点击按钮“显示/隐藏”加载中指示器

**(2)TouchableOpacity**——为可触摸组件提供透明效果

用法： <TouchableOpacity onPress={...}>

<Any />

</TouchableOpacity>

**(3)FlatList**——列表组件，高性能的列表(相较于已废弃的ListView）

用法：<FlatList **data={this.state.plist}** **renderItem={this.\_renderItem}** onEndReached={this.fn} onEndReachedThreshold={0.1} ListHeaderComponent={} ListFooterComponent={} keyExtractor={(item,i)=>i+''}>

</FlatList>

**(4)SectionList**——列表组件，列表项可以进行分组

|  |
| --- |
| 面向对象扩展知识点： 实例属性和静态属性  class Student{  sname='未命名' //实例属性：每个该类的实例都有一份  static guoJi='中国' //静态属性：整个类的所有实例共用一份  }  let s1 = new Student( ); //s1有一个专有的sname  s1.sname = 'ding'  let s2 = new Student( ); //s2有一个专有的sname  s2.sname = 'yaya'  Student.guoJi = '中华人民共和国'  console.log( Student.guoJi ); |

## 2.RN中的布局

RN中的组件布局采用的类似FlexBox布局方案，但要比CSS3的flexbox要简单，相关属性就四个：

① flex：弹性占比：number

<Any style={{flex:1}}/>

<Any style={{flex:2}}/>

<Any style={{flex:1}}/>

② flexDirection：排列方向: **'column'**， 'row'

③ alignItems：子组件的对齐方式： 'start', 'center', 'end'

④ justifyContent：内容调整方式： 'start', 'end', 'center', 'space-evenly'，'space-around', 'space-between'

练习：完成如下的页面布局



View：竖直排列两个孩子，高度1:1

> View：水平排列两个孩子，宽度1:1

> View：竖直放置Image+Text，水平居中，竖直居中，紧挨一起

练习：完成如下组件布局



View：水平排列两个孩子，均匀分配空白空间

|  |
| --- |
| 小知识：增量排错  1  2  3  4  ~~5~~  ~~6~~  ~~7~~  ~~8~~ |

## 课后任务：

1.整理React和ReactNative所有知识点思维导图

2.根据设计图，完成RN项目三个页面的布局和网页请求功能：

①LoginScreen

②MainScreen

③ProductListScreen

# day06

## 复习：

见思维导图

## 1.ReactNative阶段项目

实现步骤：

(1)创建空白的RN项目模板，运行服务器端，并用App端来访问

(2)创建项目必需的组件

LoginScreen 盛放登录组件的屏幕

MainScreen 盛放主菜单组件的屏幕

ProductListScreen 商品列表屏幕

ProductDetailScreen 商品列表屏幕

(3)声明并注册路由词典，修改App.js

let routes = createStackNavigator({

login: LoginScreen, //第一个路由即是“首屏内容”

main: MainScreen,

productList: ProductListScreen,

productDetail: ProductDetailScreen,

})

//创建主组件(自带页头)，在其中注册路由词典

export default createAppContainer( routes )

此时可以使用客户端进行测试，访问页签中的每个按钮

---------------------------------------------------------------------

(6)开始修改登录组件：LoginScreen.js，添加页面内容

View

> TextInput

> TextInput.secureTextEntry

> Button

> View > Image + Text

> Text

模型数据：this.state = { adminName: '', adminPwd:'' }

----------------------------------------------------------------------

(7)修改主菜单模板：MainScreen，添加页面内容

模型数据：

--------------------------------------------------------------------

(8)修改商品列表屏幕模板：ProductListScreen.js，添加页面内容

模型数据：

--------------------------------------------------------------------

(9)修改商品详情屏幕模板：ProductDetailScreen.js，添加页面内容

模型数据：

## 2.RN中的路由和导航

React和React Native官方都没有提供路由模块，可以使用第三方模块：React-Navigation( https://reactnavigation.org )——这套方案彻底摆脱了浏览器和地址栏/URL的限制。

使用步骤：

①创建页面组件

②定义路由词典

let routes = createStackNavigator({

//path:'...', component: '...'

login: LoginScreen, //<LoginScreen **navigation**={}>

main: MainScreen

})

③在根组件中注册路由词典

export default createAppContainer( routes )

④路由跳转

//<LoginScreen navigation={}>

//所有的路由组件，在创建时都被注入一个属性navigation

//组件内部可以使用this.props.navigation访问该属性

this.props.navigation.navigate('main')

this.props.navigation.goBack()

this.props.navigation.popToTop()

⑤跳转传参

页面1：this.props.navigation.navigate('页面2', { pid: 999, uname:'' } )

页面2：let pid = this.props.navigation.getParam('pid', 默认值)

⑥为每个页面提供头部标题栏内容：

class XxxScreen extends React.Component{

static navigationOptions = {

title: '页面标题字',

//headerTitle: <Image.../>

//headerRight: <Button.../>

}

}