一、移动端课程回顾：

①hybridApp

3种mobile开发模式 ：webApp nativeApp hybridApp

搭建Android开发环境

webView （new\setJSEnable\loadUrl\setContentView）

②ionic

ionic = ng + ui库（css样式类以及基于ng扩展的指令和服务）+ uiRouter（配置状态机）+cordova

ionic样式类：button/list/card/checkbox/form/...

ionic基于ng的扩展：ionRefresher/ionInfiniteScroll/$ionicLoading/$ionicPopup/$ionicScrollDelegate...

uiRouter:第三方的框架，可以用在ng1、ng2、React中，最大的特点就是不仅仅支持路由管理，支持路由嵌套

①基本步骤

引入angular-ui-router.js 依赖ui.router

$stateProvider.state() $urlRouterProvider.otherwise()

②跳转

$state.go()

ui-sref

a href='#url'

③传参

发送方、接收方

配置接收方状态所对应的url

.state('kflDetail',{

url:'/kflDetail/:id'

})

接收参数 $stateParams.id

发送参数

③zeptojs

移动端的jQuery，提供了常用的选择器和DOM操作方法，

同时支持在移动端的处理：touch、detect

zepto默认包含的5个模块

①zepto 核心模块

②event 事件模块 on off one trigger

③ajax 通信模块 $.ajax() $.get/post/load/jsonP

④form 表单模块 submit serialize serializeArray

⑤ie ie10+

detect模块： $.os $.browser

touch 模块: tap/longTap/doubleTap swipe/swipeLeft...

fx 模块: $.animate/show/hide/toggle/fade\*

①在线版简历

通过zepto完成服务器通信以及选择器和DOM操作

②将pc端2048游戏重构移动端的app

给项目添加滑动事件、计算出来宽高等属性设置适应不同的屏幕大小、打包成hybridApp、上传到应用商店（蒲公英）、新浪云部署

（①将代码包传上去 ②指定服务器 ③配置数据库）

4、phoneGap/cordova

phoneGap是一个平台，提供丰富的插件（允许使用js调用设备底层的软硬件信息），最终实现接近于原生app体验的应用程序

device/acclerometer/notification/file/capure/camemra/...

二、React概述

what？React是一个开源的js库，将数据渲染成视图

why？

fb遇到：①浏览器性能 ②代码维护太复杂

创建的React，通过VirtualDOM来尽量减少对DOM的直接操作解决浏览器遇到性能瓶颈问题，通过单向数据流来解决代码维护的问题

where?

提供渲染视图，数据操作比较频繁的场景

how？？

①CLI

https://github.com/facebookincubator/create-react-app

npm install -g create-react-app

create-react-app my-app

cd my-app/

npm start

②直接引入对应的js文件

demo01：

<head lang="en">

<meta charset="UTF-8">

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

<script src="js/react-dom.js"></script>

<script src="js/browser.min.js"></script>

<title></title>

</head>

<body>

<div id="example"></div>

<script type="text/babel">

ReactDOM.render(

<h1>Hello React</h1>,

document.getElementById("example")

);

</script>

</body>

核心思想：组件化

核心概念：

①组件

②jsx （并不是强制要求使用jsx，只是建议使用jsx）

并不是一种新的语言，只是js的扩展：允许在js中直接来编写html标签

③VirtualDOM 虚拟DOM

结合着React当中的Diff算法，在于可以极大限度的减少对于真实DOM的操作

④组件相关的属性state、props。。

三、React使用

1、基本用法

react.js 核心文件

react-dom.js 处理dom

browser.js 处理浏览器兼容

babel是一个js编译器，支持es6新特点转换为普通的js，支持jsx转换为js

ReactDOM.render(arg1,arg2) 将arg1中的内容 插入到arg2容器中去渲染视图。

2、jsx语法

如果渲染多个元素，不允许直接渲染多个，但是可以将要渲染的元素放在一个顶层标签是允许的。

jsx语法：

jsx并不是新的语言，只是js的扩展。

如果遇到{就用js解析，

如果遇到<使用html来解析

编写的jsx语法，浏览器默认是不支持的，但是通过babel编译器，转化为了浏览器能够识别的js。

练习：

新建一个文件，引入react对应的js文件，使用jsx的语法来实现：逻辑运算、三目运算。

<!DOCTYPE html>

<html>

<head lang="en">

<meta charset="UTF-8">

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

<script src="js/react-dom.js"></script>

<script src="js/browser.min.js"></script>

<title></title>

</head>

<body>

<div id="example"></div>

<script type="text/babel">

ReactDOM.render(

<div>

<p>三目运算：{ 3>2?'大于':'小于等于' }</p>

<p>关系运算：{3\*5}</p>

<span>逻辑运算：{ true && false ? 'true' : 'false'}</span>

</div>,

document.getElementById('example')

);

</script>

</body>

</html>

3、组件

组件创建和使用

怎么创建？

var Hello = React.createClass({

render:function(){

reutrn 渲染的标签

}

})

怎么使用？

就像使用普通的html标签使用自定义组件

<Hello></Hello>

<Hello/>

注意事项：

①组件名称首字母必须是大写

②返回多个元素 要放在一个顶层标签

③在组件的render方法返回元素时，不允许直接换行

思路切换--》组件化的思维模式：将看到的一切的页面 通过组件的方式来拆分，最终拼接起来构成一个复杂的应用程序

4、复合组件

就是一个普通的组件，组件的内部在渲染的时候，包含其他的组件。

可以类似玩积木一样，创建一个小组件，最终拼在一起，构成复杂的ui。

<!DOCTYPE html>

<html>

<head lang="en">

<meta charset="UTF-8">

<title></title>

<!-- 第一步：引入3个js文件-->

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

<script src="js/react-dom.js"></script>

<script src="js/browser.min.js"></script>

</head>

<body>

<!--第二步：创建视图-->

<div id="example"></div>

<!--第三步：一定要在type=text/babel(编译器)，因为用jsx语法 -->

<script type="text/babel">

var Title=React.createClass({

render:function(){

return <div>

<h1>title</h1>

</div>

}

});

var Content=React.createClass({

render:function(){

return <div>

<h2>小title</h2>

<p>我是一个加油猪</p>

</div>

}

});

//复合组件

var Together=React.createClass({

render:function(){

return <div>

<Title/>

<Content/>

</div>

}

});

ReactDOM.render(

<Together/>,

document.getElementById("example")

)

</script>

</body>

</html>

5、props（properties 属性）

①传值和接收值

在任意一个组件上，指定对应的属性以及对应的值；

如何读取属性中的值？this.props.属性名称

//调用组件并指定属性和对应的值

<PropComponent userName='zhangsan'>

</PropComponent>

//在组件内部通过this.props读取对应的属性的值

var PropComponent = React.createClass({

render:function(){

return <h1>Hello{this.props.userName}</h1>

}

})

练习：使用props来重构登录窗口

<!DOCTYPE html>  
<html>  
<head lang="en">  
 <meta charset="UTF-8">  
 <script src="js/react.js"></script>  
 <script src="js/react-dom.js"></script>  
 <script src="js/browser.min.js"></script>  
 <title></title>  
</head>  
<body>  
  
<div id="example"></div>  
  
<script type="text/babel">  
 var LabelComponent = React.createClass({  
 render:function(){  
 return <label>{this.props.labelName}</label>  
 }  
 })  
   
 var InputComponent = React.createClass({  
 render:function(){  
 return <input type="text"  
 placeholder={this.props.tips}/>  
 }  
 })  
  
 var ButtonComponent = React.createClass({  
 render:function(){  
 return <button>{this.props.btnName}</button>  
 }  
 })  
  
 var LoginWindow = React.createClass({  
 render:function(){  
 return <div>  
 <LabelComponent labelName='用户名'>  
 </LabelComponent>  
 <InputComponent tips="请输入用户名">  
 </InputComponent>  
 <br/>  
 <LabelComponent labelName='密码'>  
 </LabelComponent>  
 <InputComponent tips="请输入密码">  
 </InputComponent>  
 <br/>  
 <ButtonComponent btnName="登录">  
 </ButtonComponent>  
 <ButtonComponent btnName="注册">  
 </ButtonComponent>  
 </div>  
 }  
 })  
   
 ReactDOM.render(  
 <LoginWindow></LoginWindow>,  
 document.getElementById('example')  
 )  
  
</script>  
</body>  
</html>

props 不仅可以传递普通的值，传递一些方法，就像是传递值传方法

怎么传？？

parentComponent:

handleLogin:func,

name:'zhangsan'

<ButtonCom btnName={this.name}/>

<ButtonCom funcClick={this.handleLogin}/>

怎么接受？

this.props.btnName

this.props.funClick()

②this.props.children

this.props对象中的属性和创建组件时指定的属性 是一一对应的，但是有一个比较特殊

this.props.children:当组件在被调用时，所有的子元素。当有一个子元素时，只返回一个对象；有多个子元素时，返回一个数组；没有任何子元素，返回undefined。

React.Children.map可以帮助来处理this.props.children

<!DOCTYPE html>  
<html>  
<head lang="en">  
 <meta charset="UTF-8">  
 <title></title>  
 *<!-- 第一步：引入3个js文件-->* <script src="js/react.js"></script>  
 <script src="js/react-dom.js"></script>  
 <script src="js/browser.min.js"></script>  
</head>  
<body><div id="example"></div><script type="text/babel">  
 //创建组件  
 //将MyList中指定的3个P，显示在li，插入到ul中  
 var MyList=React.createClass({  
 displayList:function(){  
 return React.Children.map(  
 this.props.children,  
 function(child){  
 return <li>{child}</li>  
 }  
 )  
 },  
 render:function(){  
 return <ul>  
 {  
 //调用这个方法  
 this.displayList()  
 }  
 </ul>  
 }  
 })  
 ReactDOM.render(  
 <MyList>  
 <p>test01</p>  
 <p>test02</p>  
 <p>test03</p>  
 </MyList>,  
 document.getElementById("example")  
 )  
</script>  
</body>  
</html>

6、ref

VirtualDOM(VDOM)：是由各个组件构成一种数据结构。

ul

li 10

li 11

li 12

li 13

/ul

ul

li 09

li 10

li 11

li 12

li 14

/ul

VDOM有一个diff算法，当需要更新视图时，找到VDOM与真实DOM的区别，最后最大限度的减少对真实的DOM的修改。

在组件中，如果要拿到真实的DOM节点，必须是在插入到DOM之后，可以通过ref属性找到真实的DOM节点进行操作。

怎么指定引用？

<customComponent ref='refName'>

怎么通过引用找到元素呢？

this.refs.refName

练习：实现一个组件，组件中有两个输入框，一个求和按钮；点击按钮，将输入的值求和并弹窗显示。

要求：通过ref

<head lang="en">  
 <meta charset="UTF-8">  
 <title></title><script src="js/react.js"></script>  
 <script src="js/react-dom.js"></script>  
 <script src="js/browser.min.js"></script>  
</head>  
<body><div id="example"></div><script type="text/babel">  
 //创建组件  
 var AddComponent=React.createClass({  
 handleClick:function(){  
 //拿到用户输入的数据并求和 弹窗显示  
 var num1 = this.refs.numberTop.value;  
 var num2 = this.refs.numberBottom.value;  
 var sum = parseInt(num1)+parseInt(num2);  
 alert(sum);  
 },  
 render:function(){  
 return <div>  
 <input type="text" ref='numberTop'/>  
 <br/>  
 <input type="text" ref='numberBottom'/>  
 <br/>  
 <button onClick={this.handleClick}>求和</button>  
 </div>  
 }  
 });  
 ReactDOM.render(  
 <AddComponent></AddComponent>,  
 document.getElementById("example")  
 )  
</script>  
</body>

综合练习：

A --> input

B --> button

Main:A+B

实现的功能：(5:03 - 5:30)

点击B组件的按钮，能够调用A组件中输入值

A组件和B组件 是兄弟组件要想通信，借助于Main组件

1、完成A组件中用户的输入能够存在main组件

①只要是用户输入，就可以拿到输入的值并打印出来

②如何传递个Main组件

Main组件是可以给A组件传值、传方法

Main

save:function(arg){}

A funcSave={this.save}

A onChange this.props.funcSave()

2、完成B组件按钮点击 读取Main组件中的值

在main中定义一个方法，传递给B组件，按钮点击弹窗显示结果

每日一练：完成综合练习！