### 一．react

Day1 ················································1

Day2 ················································1

Day3 ················································1

Day4 ················································1

Day5 ················································1

Day6 ················································1

Day1

https://github.com/brillout/awesome-react-components

React

学习安排：

①概述

②搭建开发环境

③React的核心思想和

核心概念

④React的组件的

基础知识

一、概述

React是是由facebook推出的,一个是通过组件化，用来实现ui层的开源的js库

why?

①大量的数据操作，带来了大量的DOM操作,浏览器遇到了性能瓶颈问题--》虚拟DOM

②大量的数据操作背后，复杂的业务逻辑，代码的维护和开发成本提高

--》state和组件化

React特点：

①组件化

②高效（通过虚拟DOM，最大限度的减少真实DOM的操作）

③JSX：不是一种新的语言，是对于js的扩展，允许在js中编写html

④灵活 可以非常方便的

搭配其他库来使用

React开发环境的搭建方式：

方式1：

直接引入对应的js文件

方式2：

通过命令行的方式

//安装create-react-app

npm install -g create-react-app

//创建一个my-app的项目

create-react-app my-app

//切换到对应的目录

cd my-app/

//启动开发服务器

npm start

使用React时要考虑的

问题：

①React是一个相对年轻的框架，配套使用的库

并不是特别多

②并不是所有的项目 都适合React来实现

（数据操作非常频繁，页面更新非常多）

③React不支持ie8以下的任何浏览器（官方以后

也不支持）

核心思想：组件（是一个可被反复使用的带有特定功能的视图）

Everything is Component.

二、React开发

1、第一个例子

①引入3个js的文件

react.js 核心文件

react-dom.js 和DOm相关的文件

browser.min.js 浏览器

②指定了一个容器，用来渲染元素（容器是用来盛放根组件的）

③ReactDOM.render(要渲染的元素，准备渲染的容器);

ReactDOM.render(

<h1>Hello ReactJS</h1>,

document.getElementById('example')

)

在js中编写html标签，将这种语法称之为JSX.

④babel

<script type="text/babel">

</script>

babel是一个js的编译器，能够将es6的语法、或者jsx的语法转换为浏览器能够识别的js语法，从而来实现页面的展示

2、JSX （java script xml）

注意：JSX不是一种新的语言，只不过是允许在js中编写html

React是可以完全不使用JSX的，但是用JSX可以更简单的开发，所以还是提倡使用JSX

JSX语法 借鉴了XML标签打开与关闭的优点，提高了组件的可读性；JSX中可以直接使用HTML标签，赋予React强大的表现能力，方便进行代码的开发，最早是受php中一个插件库的影响。

基本用法：

遇到HTML标签(<),就会使用HTML来解析

遇到了代码块({),就是使用js来解析

注意事项：如果我们要渲染多个元素，要将返回的多个元素放到一个顶层标签中。

jsx练习：(11:10 - 11:20)

新建一个文件，在该文件中 指定一个容器，通过jsx语法，渲染h1(逻辑运算)、span(三目运算)

3、组件

在Vue中 创建组件：

Vue.component()

在angular中，创建组件类:

import {Component} from '@angular/core'

@Component({

selector:'',

tempalte:''

})

export class Demo01Component{

}

①在React中 创建组件

//组件类的名称 必须是全驼峰（即使是第一个单词，首字母也要大写）

var ComponentName = React.createClass({

render:function(){

return ‘要渲染的元素’

}

})

②在react中，使用组件

是按照html标签的方式 来调用组件

<ComponentName></ComponentName>

<ComponentName/>

React有一个特点，根据标签的首字母大小写，来决定是按照组件、html标签的方式 来解析内容

组件使用的过程3个注意事项：

①首字母必须大写

②不允许在render方法返回时，顶层标签或者第一个元素，不允许直接换行

③不允许直接返回多个元素，要放在一个顶层标签中

组件的工作原理：

React允许将需要反复使用的视图通过代码块的方式 封装在一个组件中，然后像插入html标签一样，在网页中插入这个组件，将组件类进行实例化，将render方法的返回元素 渲染在指定的容器中

4、React中，是以组件为基本单位

复合组件：并不是一个新的概念，只不过是由其他的组件构成的，本质依然是一个组件

组件树：在实现一个页面的时候，可以将页面中有共性的部分，封装成组件，把这些组件放在一起，构造成一个最终渲染的根组件，那么这个时候，就有一颗组件树。

树的根就是最终渲染的复合组件，树的树叶、树枝就是创建的其它的组件

练习：要求创建一个登录窗口，

离线安装vscode插件的方式：

C:\Users\web-01\.vscode\extensions

将离线好的插件 解压缩到该目录中，然后重启vscode 就可以了

5、props

(React中的核心概念：JSX/组件/props/refs/state)

复习Vue、angular组件之间通信的方式

5.1 react 父组件与子组件通信的方式

实现具体步骤：

①在调用子组件时候，通过属性传值

<son tips="something"/>

②在子组件内部，通过props属性接收值

this.props.tips

5.2 子组件想与父组件通信的方式 props。

父组件

子组件

有数据要传递给父组件

①在父组件中 定义一个用来 接收数据的方法

rcvMsg:function(data){

}

②在父组件中调用子组件时，可以将父组件中的方法传递给子组件

<son callback={this.rcvMsg}>

③子组件在需要给父组件传值时，调用该方法this.props.callback(this.uname)

5.3 children属性

this.props.children得到在调用组件时，标签内部 所写的元素，比如:

<MyList>

<li>test01</li>

</MyList>

this.props.children 是null 是对象 是数组

React.Children.map(

this.props.children,

function(child){

}

)

6、refs

reference 引用/参考的意思

①直接ref

<input type="text" ref="myInput"/>

②通过ref去找到元素

this.refs.myInput

综合练习：

实现3个组件

MyInput 只有一个输入框

MyButton 只有一个按钮

MyMain是由上述两个组件构成的

功能：在点击MyButton 获取 MyInput用户所输入的值

思路：将MyInput的值保存在MyMain中，在MyButton点击时去MyMain中取值

MyInput OnChange

步骤1：将MyInput的值保存在MyMain中

（子组件 与 父组件通信）

①在父组件中 定义一个用来 接收数据的方法

rcvMsg:function(data){

}

②在父组件中调用子组件时，可以将父组件中的方法传递给子组件

<son callback={this.rcvMsg}>

③子组件在需要给父组件传值时，调用该方法this.props.callback(this.uname)

步骤2：在MyButton点击时去MyMain中取值

Day2

一、state

1、state 概述

组件是一个可被反复使用的视图，呈现给用户，在和用户交互的时候，就避免不了数据的变化，在react中 将数据比作是状态。

React将每一个组件 都比作是一个状态机（由各种各样的状态构成的组件）

2、state的3个基本操作

初始化:getInitialState:function(){

return {myStatus:'吃饭'}

}

读: this.state.myStatus

写: this.setState({myStatus:'睡觉'},function(){

//状态设置异步操作 成功设置完成之后会调用的方法

})

重要：一旦将状态中数据 绑定到了视图，当状态中数据发生变化，视图会自动的更新

（从数据绑定到视图）

组件要显示的内容 只取决于render方法返回的内容；一旦将状态中的数据绑定到 视图，当数据发生变化的时候，react框架观察数据发生了变化，自动的采用最优算法最高效率 来完成DOM的更新(重新去执行render)

注意：不要在render方法中，做任何状态的设置。

练习：（14:17 - 14:27）

MyInput、MyButton、MyMain

按照状态的方式来处理。

二、lifecycle 生命周期

初始化、数据变化、消亡

React的组件 也有自己的生命周期，也有从创建到消亡的过程

分为3个阶段：

mount/update/unmount

对应的生命周期的某个阶段对应的处理函数（钩子函数）

componentWillMount

componentDidMount

componentWillUpdate

componentDidUpdate

componentWillUnmount

注意事项：

①如果要使用ref去寻找元素，必须等到componentDidMount

②在componentWill/DidUpdate的处理函数中是由参数的，分别对应的是准备更新/更新之前的props和state

③在一个组件中，建议：

将生命周期的处理函数 一般都是放在组件的上面，

其次自定的方法，

最下边是render方法

练习：(15:40 - 15:55)

创建一个组件并显示；组件渲染一个h1的标签，内容随便写；

要求：当组件挂载完毕之后，启动一个周期性定时器，在定时器中修改一个数据，然后实现h1的标签透明度的变化

(0~1 0~1)

提示：style={{opacity:this.state.OpacityValue}}

步骤1：完成组件的创建和显示

步骤2：初始化状态

步骤3：指定生命周期处理函数，启动定时器

三、表单受控元素

React中并不是所有的表单都得按照受控元素的方式去处理，这是React建议和解决方案。

什么是表单受控元素？

是一些表单元素，如果指定value/checked/selected,那么

就是表单受控元素（现象：针对表单操作的结果是不对的)

解决思路：

①初始化状态

②将状态的值 绑定属性value/checked/selected

③在对应的onChange处理函数中 完成对应的状态值的修改

注意事项：

如果一个组件包含多个状态，需要初始化：

getInitialState:function(){

return {

num1:'',

num2:''

}

}

需要读取状态：

this.state.num1;

this.state.num2;

设置状态：

需要修改哪个状态，就在setState方法中设置该指定的状态就可以了，其余的不需要修改的无需理会

this.setState({num1:''});

综合练习：

①完成4个组件的创建和使用

ToDoBox

ToDoInput

ToDoList

ToDoItem

②实现add

todoinput要想把数据直接给todolist，属于兄弟组件通信，借助于共同的父组件todobox

当点击add按钮时候，将用户的输入的值 传递给todobox

步骤1：在todobox中定义一个方法（参数）

步骤2：把方法通过属性传递给todoinput

步骤3：按钮单击，调用该方法

③实现delete

点击todoitem中的按钮，调用todobox中的删除数组中指定元素的方法

考察的知识点：

①组件的创建的使用

②props

③state

三个基本操作以及功能（绑定）

④jsx

⑤ref

Day3

三连棋游戏：

1、给Square组件中的按钮 指定 单击处理函数

修改Board组件中 状态的值

（子与父通信）

基本步骤：

①在父组件中 定义一个带有参数的方法

修改数组、修改xIsNext

②将带有参数的方法 传递给子组件

③在子组件中 调用该方法

要求：在状态设置成功的处理函数中 打印当前的数组

2、判断输赢

在组件更新完成之后 去判断-输赢

componentDidUpdate

将下棋结果通过弹窗显示(\*\*\*赢了、平局)

一、ReactNative的概述

移动端开发的3种常见模式：

WebApp/NativeApp/HybridApp

React:

(由React提出的核心概念：Component、JSX、Props、State、Refs)

ReactJS：是一个在web前端中实现层ui的js库

ReactNative：15年 由fb推出的ReactNative是一个用前端技能（React中的核心概念）来编写原生应用程序;在实现基于ReactNative的app时，主要是调用由ReactNative所封装的各种各样的组件（在被渲染时，是会变成真正的原生组件）

强调：ReactNative不是用来实现webApp、HybridApp，是用来实现nativeApp,即使没学过原生的Android\iOS开发，依然可以使用React中的核心概念来完成app

why?

free\open source

成本低（学习成本、时间成本）

nativeApp(性能要好)

learn once,write anywhere

where?

低成本的开发真正的原生应用程序

how？

搭建ReactNative的开发环境

①方式1 cli

//安装

npm install -g create-react-native-app

//创建一个叫做my-app的项目

create-react-native-app my-app

//切换到当前目录下的my-app

cd my-app/

//执行start脚本，启动开发服务器

npm start

②方式2 cli (建议)

//安装 对应的命令行工具 react-native-cli

npm install -g react-native-cli

//安装yarn

npm install -g yarn

//创建项目(会去自动的安装依赖的包)

react-native init my-app

//切换到my-app

cd my-app

//启动开发服务器

react-native start

大致步骤：

①下载ADT

（Android Developer Tools 内置了eclipse/sdk）

②配置环境变量

ANDROID\_HOME(将c盘adt-bundle-window\*\*,中的sdk的路径赋值给ANDROID\_HOME)

(计算机-》右键-》属性-》查看高级设置->环境变量-》创建或者编辑或者删除)

③安装Java运行环境和工具包

一般情况下，安装之后系统的环境已经自动的设置过了

④处理和ReactNative

启动开发服务器：

①进到e盘，找到software，找到RNTPL，点进去，找到myapp

②进入到myapp文件夹，点击右键，选择git bash here

去启动了一个终端（有点类似windows+R->cmd）

③在终端中，react-native start

如果能够成功的启动8081端口，说明环境是ok的，否则环境是有问题的

环境：

①pc端

react-native start 启动开发服务器

在编辑器中 编写项目中的代码

②mobile移动端

安装一个myapp的安卓系统下的应用程序

打开该应用程序，设置要连接的服务器端的ip地址和端口号（一旦连接成功，可以实现当pc端修改reactnative

项目代码，移动端可以直接去预览）

如何将一个apk结尾的文件 安装到模拟器中？

①服务器

将app-debug.apk放到htdocs

②指令

adb(android debug bridge) 实现模拟器或者真机 和pc的连接

adb install app-debug.apk

步骤1：

找到adb所在的文件夹

c:\adt-bundle-windows\*\*\*\*\adt-budnle\*\*\*\sdk\platform-tools

步骤2：

windows+R --> cmd ,将上一步找到的adb.exe直接拖拽到该命令行工具

步骤3：在命令行中 输入 空格 install 空格

步骤4：找到c xampp htdocs目录下的apk文件夹中的app-debug.apk，直接拖拽到install空格的后面，点回车即开始安装

（已经安装过，重复安装是会报错的；没有启动模拟器，安装也会失败）

二、ReactNative的知识点

学习ReactNative所封装的常用组件，包括View/Text/TextInput/Modal/...

需要将模拟器中的myapp软件 连接当前pc端的通过reactnative start所启动的开发服务器

①cmd->ipconfig

记录当前pc端的ip地址

172.163.0.155

②打开模拟器中的myapp，点击menu按键，选中Dev Settings,设置Debug Server Host&Port

如果模拟器连接debug server失败：

①debug server 是否在运行

②确认下在模拟器中通过浏览器 访问debug server比如(172.163.0.155:8081)

③在myapp这个软件中，点击menu按键，查看debug js remotely选项，如果当前处于stop remote js debugging,点即该选项把远程调试 关掉，再去重新reload

④重启模拟器

1、常用reactNative组件类

View 指定一个容器

Text 指定一段文本内容

StyleSheet 指定样式：

①import {StyleSheet} from 'react-native'

②根据需要去创建要用到的样式类

const myStyles = StyleSheet.create({

myText:{

color:'red'

},

myView:{

backgroundColor:'#eeeeee'

}

})

③使用样式类

<Text style={myStyles.myText}></Text>

注意事项：

①在ReactNative中指定宽高等数据时候，无需加上单位，reactNative会帮助我们自动处理

②指定样式时候，要采用驼峰式的写法

backgroundColor

fontSize

任务：复习css core day1 flexbox

Day4

准备环境

①启动开发服务器

找到e\software\RNTPL\myapp ，点击右键，git bash here,

在命令行窗口中：react-native start

②打开eclipse，找到avd(android virtual device manager安卓虚拟设备管理器),启动模拟器

一、Button

使用步骤：

①引入

import {Button} from 'react-native';

②使用

<Button title="clickMe" onPress={this.handlePress}></Button>

二、State

①在构造函数中去初始化状态

constructor(){

super();

//初始化状态

this.state={count:0}

}

②读

<Text>{this.state.count}</Text>

③设置

this.setState({count:1})

三、Image

①引入对应的组件类

import {Image} from 'react-native';

②根据要加载图片的路径

本地图片:

<Image source={require('../img/1.jpg')}/>

网路图片

<Image source={{uri:'\*\*\*\*.com/\*\*\*.png'}}/>

四、TextInput

使用方式：

①引入

import {TextInput} from 'react-native'

②使用

handleChange=(data)=>{

console.log(data);

}

<TextInput

onChangeText={this.handleChange}

placeholder="请输入用户名"/>

练习：

新建一个文件夹demo07，在该文件夹中创建一个文件demo07\_lianxi.js

在该文件中，创建一个基于ReactNative的一个组件，渲染一个TextInput,要求默认显示'zhangsan'

按照表单受控元素的方式来处理

常见错误：

unable to connect the remote debugger

①查看debugger service是否在运行

②检查模拟器 通过浏览器 测试 是否能够连接开发服务器，如果不行,重启模拟器

五、FlatList

①引入

import {FlatList} from 'react-native'

②调用

this.list = [];

showListItem=()=>{

//必须要返回组件

return <Text/>

}

<FlatList

data={this.list}

renderItem={this.showListItem}

/>

六、Fetch接口的使用

fetch('url')

.then((result)=>{return result.json()})

.then((result)=>{

//result就是服务器端返回的json格式的数据

})

七、TouchableOpacity

给容器中的元素 添加 触控或者点击时 透明度变化的效果

①引入

import {TouchableOpacity} from 'react-native'

②直接使用（作为容器去使用）

<TouchableOpacity>

<Text></Text>

<Image source={{uri:''}}/>

</TouchableOpacity>

练习：

创建一个demo11的文件夹，在该文件夹中创建一个demo11\_lianxi.js的文件

FlatList --> 列表项 Text (在点击Text时有透明度的变化)

当组件挂载完毕之后，向服务器发起请求，将请求到的数据渲染在列表中，当点击列表时候，弹窗显示当前的的列表项的序号

数据存在状态中