初识React

1.1 React简介

4个特点：

声明式的视图层。视图是声明式的，根据视图状态声明视图形式，采用JavaScript（JSX）语法，可以在视图层中随意使用各种状态数据。

简单的更新流程。声明式的视图定义方式简化视图更新流程，只需要定义UI状态，当状态变化，React会渲染出最新UI。

灵活的渲染实现。React并不是把视图直接渲染成最终的终端界面，而是把它渲染诚虚拟DOM。虚拟DOM只是普通的JavaScript对象，可以结合其他依赖渲染诚不同终端上的UI，如react-dom在浏览器渲染，Node在服务器端渲染，React Native在手机上渲染。

高效的DOM操作。React直接操作的时虚拟DOM，一个JavaScript对象，操作一个对象当然比操作真是DOM要快，而且，基于React优异的差异比较算法，React可以尽量减少虚拟DOM到真实DOM的渲染次数，以及每次渲染需要改变的真实DOM节点数。

1.2 ES6语法简介

1. let、const

let和const是ES6中新增的两个关键字，用来声明变量，都是块级作用域，let声明的变量只在let命令所在的代码块内有效，const声明一个只读的常量。

#### 2.React基础

2.1.1 JSX语法

长期以来，UI和数据分离一直时前端领域一个关注点。一般使用模板，将UI的定义放在模板内，将数据的逻辑维护在JS中，然后通过模板引擎，根据数据和模板文件渲染出最终的HTML文件或代码片段。

React通过组件的概念将页面进行拆分并实现组件复用，一个组件应该是具备UI描述和UI数据的完整体，不应该将他们拆开，于是有了JSX，JSX是一种用于描述UI的JavaScript扩展语法。

2.1.2 JSX语法

1.与xml语法相同，都是使用成对的标签构成一个树状结构的数据，例如：

const element=(

<div>

<h1>hello,world!</h1>

</div>

)

2.标签类型

有两种，一种时普通的DOM标签（div、span等），标签首字母小写，一种时React组件类型的标签，标签首字母大写，二者可以相互套用。例如

//DOM类型标签

const element=<h1>hello,world!</h1>;

//React组件类型标签

const element=<HelloWorld/>;

3．JavaScript表达式

在JSX中使用JavaScript表达式需要将表达式用“{}”括起来，表达式的使用场景有两个，一是通过表达式给标签属性赋值，一是通过表达式定义子组件。例如：

//通过表达式给标签属性赋值

const element=<MyComponent foo={1+2}/>

//通过表达式定义组件（map虽然是函数，但它的返回值是JavaScript表达式）

const todos=[‘item1’,’item2’,’item3’];

const element=(

<ul>

{todos.map(message=><Item key={message} message = {message}/>)}

</ul>

)

JSX中只能使用JavaScript表达式，不能使用多行JavaScript语句。

JSX中可以使用三目运算符或逻辑与运算代替if语句的作用。

4.标签属性

当JSX的标签是DOM的标签时，class变成className，onclick变成onClick，当JSX标签时React组件标签时，可以任意自定义标签属性名。

5.注释

JSX中的注释需要用“{}”将/\*\*/包裹，例如

const element=(

<div>

{/\*这里是一个注释\*/}

<span>React</span>

</div>

)

2.1.3 JSX不是必须的

JSX只是React.createElement(component,props,…children)的语法糖，所有的JSX语法都会转换成这个方法的调用。

//JSX语法

const element=<div className=’foo’>Hello, React</div>

//准换后

const element=React.createElement(‘div’,{className:’foo’},’Hello,React’)

#### 2.2 组件

2.2.1 组件的定义

React正式一个一个的组件搭建而成的，组件的定义有两种方式，一是使用ES6 class（类组件），一是使用函数（函数组件）。先介绍class组件。

使用class组件需要满足两个条件：

1. class继承子React.Component.
2. class内部必须定义render方法，render方法返回代表该组件UI的React元素。

例如定义PostList组件

//PostList.js

Import React, {component } from ‘React’;

Class PostList extends Component {

render(){

return(

<div>

帖子列表：

<ul>

<li>大家一起讨论React吧</li>

<li>前端框架，你最喜欢哪一个</li>

</ul>

</div>

);

}

}

export default PostList;

在定义组件之后，使用ES6的export将PostList作为默认导出。将PostList挂在到页面的DOM节点上，使用ReactDOM.render()

//index.js

import React from “react”;

import ReactDOM from “react-dom”;

import PostList from “./PostList”;

ReactDOM.render(<PostList/>,document.getElementById(“root”));

使用ReactDOM.render()需要导入react-dom库，这个库完成组件所代表的DOM到浏览器的DOM的转换。

2.2.2 组件的props

组件的props用于把父组件的数据或方法传递给子组件。

<User name=’React’ age = ‘4’ address = ‘America’ >

此时User组件的props结构如下：

props={

name:’React’,

age:’4’,

address:’America’

}

利用props定义PostItem组件：

//PostItem.js

Import React,{ Component } from “React”;

Class PostItem extends Component {

render(){

const{title,author,date}=this.props;

return(

<li>

<div>

{title}

</div>

<div>

创建人:<span>{author}</span>

</div>

<div>

创建时间：<span>{date}</span>

</div>

</li>

);

}

}

export default PostItem;

然后在PostList中使用PostItem：

//PostList.js

import React, {Component} from “React”;

import PostItem from “./PostItem”;

const data=[

{title:”aaa”,author:”bbb”,date:”ccc”}

{title:”ddd”,author:”eee”,date:”fff”}

]

Class PostList extends Component {

render(){

return(

<div>

帖子列表：

<ul>

{data.map(item=><PostItem title={item.title} author={item.author} date={item.date}/>)}

</ul>

</div>

);

}

}

export default PostList;

2.2.3 组件的状态

组件的state是组件内部的状态，state的变化最终反应在组件UI的变化上。在组件的构造方法constructor中通过this.state定义组件的初始状态，通过调用this.setState方法改变组件的状态，也是唯一改变组件内部状态的方法。

React组件可以看做是一个函数，入参是props和state，函数的输出时组件的UI。React组件正式由props和state两种类型的数据驱动渲染组件UI。props时组件对外的接口，组件通过props接收外部传入的数据（包括方法）；state是组件对内的接口，组件内部状态的变化通过state来反应。props是只读的，不能在组件内部修改props，组件状态的修改通过修改state来实现。

2.2.4 有状态和无状态的组件

组件不一定有状态，开发React应用时，尽可能多的设计成无状态的组件，这种组件不关心状态的变化，聚焦渲染，容易被复用。React组件设计的一般思路是，通过定义少数的有状态的组件管理整个应用的状态变化，将状态通过props传递给其余的无状态的组件，有无状态组件完成绝大部分的渲染工作。有状态的组件聚焦状态变化的业务逻辑，无状态的组件聚焦组件UI的渲染。

使用无状态组件时，尽量使用函数方法定义组件：

function Welcome(props){

return <h1>Hello, {porps.name}</h1>;

}

2.2.5 属性校验和默认默认属性

React提供PropTypes这个对象用于组件属性类型的校验：

import PropTypes from ‘prop=types’;

2.2.6 组件样式

1.外部CSS样式

这种方式和平时开发CSS文件相同，CSS样式中用HTML标签类型、ID、class等选择器定义元素的样式，唯一区别时React元素要使用className代替class作为选择器。

样式表的引入方式有两种：通过标签引入，常用于该样式表文件作用与整个应用的所有组件（一般时基础样式表）

<link rel=”stylesheet” type=”text/css” href=”style.css”>

另一种是使用模块的形式，像导入其他组件一样导入样式表，常用于该样式作用于某个组件。:

import ‘./style.css’;

2.内联样式

将css写到JS中：

function Welcome(props){

return(

//style使用两个大括号，第一个表示style的值时一个JavaScript表达式，第二个括号表示这个JavaScript表达式是一个对象。

<h1 style={{

width:”100%”,

hight:”50px”,

backgroudColor:”blue”,

fontSize:”20px”

}}

)

}

2.5 事件处理

在React中绑定事件有两点注意：

1. 在Reac中，事件的命名采用驼峰形式，而不是DOM元素的小写字母命名方式。
2. 处理事件的相应函数要以对象的形式赋值给事件属性，而不是DOM中的字符串形式。

React中的事件时合成事件，并不是原生的DOM事件，在DOM中，可以通过处理函数返回false来阻止事件的默认行为，在React中必须显示的调用事件对象的preventDefault方法来阻止事件的默认行为。如果必须使用DOM的原生事件，可以通过React事件对象的nativeEvent属性获取。

在React中，处理事件最容易出错的地方时事件处理函数中的this指向的问题，ES6 class并不会为方法自动绑定this到当前对象。有三种形式，不同的写法解决this指向问题的方式也不同。

#### 3.react16新特性

1. render的新返回类型，可以返回字符串和由react组件或元素构成的数组；
2. 用错误边界（componentDidCatch(error,info)方法的组件）友好的处理错误；
3. 增加Protals，一个ReactDOM.createProtal(child（要挂载的元素或组件）,container（child挂载到的节点）)方法，返回一个弹框返回给任何想挂载的react树结构的container节点。
4. React把自定义dom的属性，不识别的也会渲染，而老的DOM不会渲染。

#### 4.深入理解组件

4.1 组件的state

4.1.1 设计合适的state

1.state设计为能表示组件所有的变化的最小集合，包括UI渲染需要的数据和判断是否需要渲染的条件两种数据；

2.普通属性，即props和state之外的属性，如果需要一个变量，与组件的渲染无关，不表示组件的状态，只用来改变组件的state的属性，设计成普通属性，挂载到this下；

4.1.2 正确修改state

1.用this.setState()，而不是this.state.xxx=”yyy”，因为后者不会引起render;

2.state的更新时异步的，多个改变state的操作时，哪一个先发生不确定，所以如果是后一个的改变需要前一个已经改变作为前提时，用this.setState(preState,props)=>({counter:preStatequantity+1});

3.更新state时，setState（需要改变的变量）。

4.2.3 state与不可变对象

React把state当做不可变变量，即当setState（状态）时，应该是重新创建对象对象，而不是改变当前的对象。

1. 状态是不可变类型（数字、字符串、布尔值、null、undefined）时，直接重新赋值；
2. 状态是数组时
3. 用preState、concat重新创建数组

this.setState(preState=>({books:preState.books.concat{[‘react guide’]}})

1. ES6 spread syntax重新创建数组

this.setState(preState=>({books:[...preState.books,’react guide’];})

取出部分时，用slice方法：

this.setState(preState=>({books:preState.books.slice（1,3）})

过滤部分时，用filter方法：

this.setState(preState=>({books:preState.books.filter(item=>{return item!==’react’})})

不能用push、pop、shift、unshift、splice等，因为这些方法只改变当前数组的，并不是返回一个新的数组。

1. 状态是对象类型
2. 使用ES6的Object.assgin方法：

this.setState(preState=>({

owner:Object.assgin({},preState.owner,{name:’faith’});

}))

1. 使用对象扩展语法

this.setState(preState=>({

owner:{...preState.owner,name:’faith’}

}))

总结，改变组件的状态，避免使用改变原对象的方法，而应该使用会返回新对象的方法，因为组件的生命周期的shouldComponentUpdate方法就可以通过判断两次对象的引用是否已经改变，从而避免不必要的render，提升性能。