深入React技术栈和企业级项目开发

第2天课堂笔记

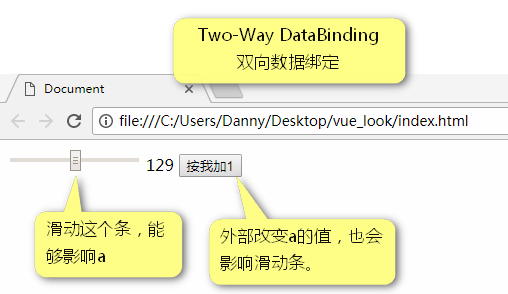
讲师：邵山欢

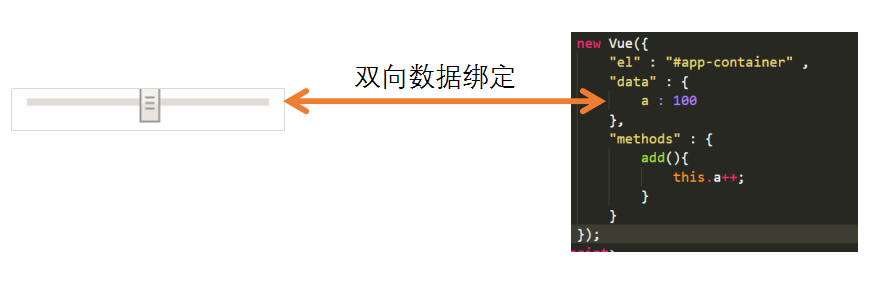
日期：2017年11月19日

# 复习深入昨天的内容

## 1.1 React中没有双向绑定

React没有双向绑定。我们来看看Vue.js框架中，是如何处理一个值和滑动条的关系的。





代码很简单：

<div id="app-container">

<input type="range" min="0" max="255" **v-model="a"** />

{{a}}

<button v-on:click="add()">按我加1</button>

</div>

**通过了一个指令v-model实现了滑动条和a变量的双向数据绑定**。

看一下细节：滑动条上没有value属性，因为它的值就是a；滑动条上没有onchange事件监听，因为它改变了值，就直接影响了a；

也就是说v-model干了两个事儿：① 设置了这个条的值实时等于a； ②设置了这个条的事件监听，让这个条被滑动的时候影响a。

React中没有双向绑定的，所以我们的滑动条上要写两个东西：value、onChange。

value负责让滑动条实时等于绑定的state.a；onChange负责让滑动条的改变影响state.a。

render(){

return <div>

<input

type="range"

min="0"

max="255"

**value={this.state.a}**

**onChange={(event) => {this.rangeChangeHandeler(event.target.value)}}**

/>

{this.state.a}

<button onClick={this.add.bind(this)}>按我加1</button>

</div>

}

## 1.2 说一下符号

只有两个地方能够使用JSX语法，就是render()函数中的return和ReactDOM.render()第一个参数。

**加上{}之后，{}里面的语句就变为JS的了。**

JSX中{}左右自带连字符的：

render(){

return <div>

<h1>我爱你{5000\*2}年</h1>

</div>

}



{}中可以使用变量的：

render(){

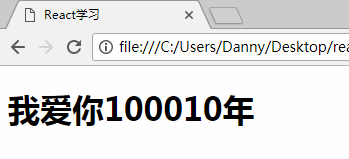
var year = 100010;

return <div>

**<h1>我爱你{year}年</h1>**

</div>

}



style的符号比较多：

写法1：

render(){

var r = 100;

var g = 200;

var b = 123;

return <div>

<h1 style={{"color":"rgb(" + r + "," + g + "," + b + ")"}}>哈哈</h1>

</div>

}

写法2：

render(){

var r = 100;

var g = 200;

var b = 123;

return <div>

<h1 style={{"color":`rgb(${r},${g},${b})`}}>哈哈</h1>

</div>

}

# 受控组件和非受控组件

In a controlled component, form data is handled by a React component.

在受控组件中，表单数据被React component管理、操纵；

The alternative is uncontrolled components, where form data is handled by the DOM itself.

替代品就是非受控组件，费受控组件中的表单数据被DOM元素自己管理。

一个组件只有表单元素，才有资格区分是受控的、还是非受控的。

## 2.1 非受控组件

下面的表单中，有两个输入框一个按钮，输入框都有ref属性，ref属性便于我们的程序取得它们的值。

点击按钮之后，直接通过this.refs.\*\*来取得文本框的值，然后提交。

这里面丝毫没有用到state，没有对组件的state产生任何改变。

import React from "react";

export default class App extends React.Component{

constructor(props){

super(props);

}

submitHandeler(){

var name = **this.refs.nameTxt**.value;

var age = **this.refs.ageTxt**.value;

alert("提交了数据" + name + age);

}

render(){

return <div>

<p>

姓名：

<input type="text" **ref="nameTxt"**/>

</p>

<p>

年龄：

<input type="text" **ref="ageTxt"**/>

</p>

<button onClick={this.submitHandeler.bind(this)}>提交</button>

</div>

}

}

## 2.2 受控组件

受控组件中的所有表单元素，都使用onChange和value属性和state进行模拟“双向绑定”。

import React from "react";

export default class App extends React.Component{

constructor(props){

super(props);

this.state = {

name : "",

age : ""

}

}

submitHandeler(){

alert("提交了！" + JSON.stringify(this.state));

}

changeName(name){

this.setState({

name

});

}

changeAge(age){

this.setState({

age

});

}

render(){

return <div>

<p>

姓名：

<input

type="text"

**value={this.state.name}**

**onChange={(e)=>{ this.changeName(e.target.value) }}**

/>

</p>

<p>

年龄：

<input

type="text"

**value={this.state.age}**

**onChange={(e) => { this.changeAge(e.target.value) }}**

/>

</p>

<button onClick={this.submitHandeler.bind(this)}>提交</button>

</div>

}

}

## 2.3 ref和e.target.value

ref是非受控组件中用的，受控组件绝对出现不了ref这个东西。

ref表示组件内部的选择，表单控件身上加ref

<input type="text" **ref="nameTxt"**/>

得到它的值：

this.**refs**.nameTxt

受控组件中，组件自己要和state进行模拟“双向数据绑定”。此时onChange事件中，要把自己的值传出去，此时就要写一些箭头函数，**将小e对象的e.target.value属性传出去**。在JS进阶中老师告诉大家，e.target就是e.srcElement，表示获得事件的最小元素。

<input type="text" onChange={(e)=>{this.changeHandler(e.target.value)}} />

# 组件之间数据传递

React中，组件之间几乎都是通过HTML参数进行传递，传递是单向的，父亲到儿子；参数对于儿子是只读的，儿子要用state承接父亲的参数，如果要改变父亲的参数，要调用父亲的函数。

## 3.1 父组件通过HTML标签的属性将值传给子组件

父组件用HTML标签的属性来传给儿子自己的值，儿子用this.props.a来接收。

父组件：

import React from "react";

import Erzi from "./Erzi.js";

export default class Baba extends React.Component{

constructor(props){

super(props);

this.state = {

a : 100

}

}

render(){

return <div>

<h1>我是爸爸组件，a值是{this.state.a}</h1>

<hr />

<Erzi **a={this.state.a}**></Erzi>

</div>

}

}

子组件：

import React from "react";

export default class Erzi extends React.Component{

constructor(props){

super(props);

}

render(){

return <div>

<h1>我是儿子组件，我收到了父亲的传给我的a是**{this.props.a}**</h1>

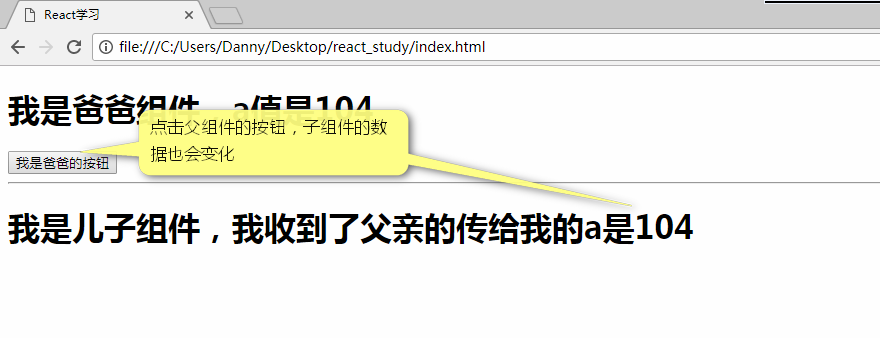
</div>

}

}

## 3.2 如果父组件的值改变，子组件也会刷新视图

因为父组件的render重新执行了，所以相当于重新给子组件传值了。



## 3.3 对于子组件来说，父亲传入的数据是只读的

下面的语句是错误的：

this.props.a = this.props.a + 1;

因为props是只读的！

**有的时候，子组件不想改变父组件的值，只想让父亲传入一个初始值，此时内部改变这个值，不影响父亲。**

**此时可以用state来承接父亲的props。**

import React from "react";

export default class Erzi extends React.Component{

constructor(props){

super(props);

**this.state = {**

**a : props.a**

**}**

}

render(){

return <div>

<h1>我是儿子组件，我收到了父亲的传给我的a是**{this.state.a}**</h1>

<button **onClick={() => {this.setState({a : this.state.a + 1})}}**>我是儿子中的按钮</button>

</div>

}

}

## 3.4 子组件如果要改变父组件传入的值，父组件要将函数一并传给子组件

父组件：

import React from "react";

import Erzi from "./Erzi.js";

export default class Baba extends React.Component{

constructor(props){

super(props);

this.state = {

a : 100

}

}

**addA(){**

**this.setState({**

**a : this.state.a + 1**

**});**

**}**

render(){

return <div>

<h1>我是爸爸组件，a值是{this.state.a}</h1>

<button onClick={()=>{this.setState({"a" : this.state.a + 1})}}>我是爸爸的按钮</button>

<hr />

<Erzi a={this.state.a} **addA={this.addA.bind(this)}**></Erzi>

</div>

}

}

子组件中调用父亲传入的函数：

import React from "react";

export default class Erzi extends React.Component{

constructor(props){

super(props);

}

render(){

return <div>

<h1>我是儿子组件，我收到了父亲的传给我的a是{this.props.a}</h1>

<button **onClick={() => { this.props.addA()}}**>我是儿子中的按钮</button>

</div>

}

}

# 组件封装小案例

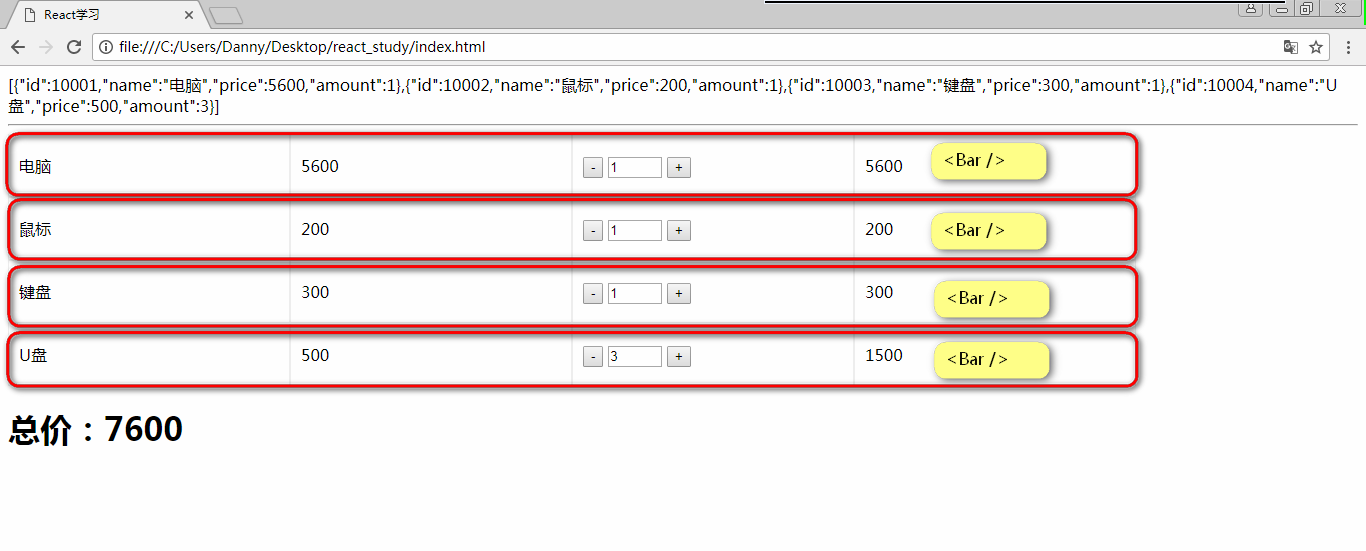
## 4.1 组件化的调色板

我们现在的一个初步认知，就是页面上有HTML、CSS、JS行为都挺类似的部分，可以封装为组件，比如三个Bar：



代码见案例。

## 4.2 组件化的购物车结算页面



## 4.3 日历组件

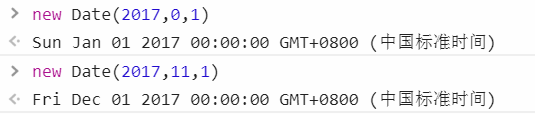
## 日历原理

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 日 | 一 | 二 | 三 | 四 | 五 | 六 |
| 29 | 30 | 31 | 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| 19 | 20 | 21 | 22 | 23 | 24 | 25 |
| 26 | 27 | 28 | 29 | 30 | 1 | 2 |

决定一个月的样子，由三个要素决定：

1. 本月1号星期几
2. 这个月一共几天
3. 上一个月有几天

JS内置了一个Date对象，Date对象浏览器全线



需要注意的是，月份从0开始算，所以new Date(2017,0,1)表示2017年1月1日。

如果要得到某一天星期几，此时先new出这个日期对象（比如2017年11月19日），然后调用getDay()方法即可。

var d = new Date(2017,10,19);

d.getDay()

0表示星期天

如果要得到某月第一天星期几，那么new的时候，最后一位写1即可，比如想知道2017年11月第1天星期几， 此时：



我们要得到2017年11月共几天？只需要求2017年11月的下一月的1号的前1毫秒是哪天就行。

(new Date(new Date(2017 , 10 + 1 , 1) - 1)).getDate() //30

自己看手册，-1得到时间戳，再次用时间戳new，又能得到-1毫秒的时间。

我们要得到2017年11月的上一个月有几天，只需要求2011年11月1日的前1毫秒是哪天就行。

(new Date(new Date(2017,10,1) - 1)).getDate() //31