# 第4章 react组件

（1）最好能多一些图，能降低写作难度也能降低阅读难度。比如流程的图、原理的图。哪怕通信过程的图。

（2）好多语句不通顺，写作中要注意写完后读一下。

（3）英文大小写务必注意，比如react还是React？app或APP？无法确定的，就全文统一，我最后替换。

（4）英文简称第一次出现的时候要给全称和中文注解，比如es6。

（5）每个例子列举之前，尽量说出是个什么例子、完成什么功能，别直接说举个栗子然后立刻给代码。代码后也要多给一些解释。

（6）样式要注意，代码样式我们模板已经提供，第一个我改了。代码第1行可以是文件名。

在接下来的一章中，我们会了解 react组件 是什么，它是如何运作以及如何使用它。又是如何在jsx语法种书写react组件。在这一章中将会了解react组件是如何完成webApp的开发，react的生命周期函数的理解。

本章主要涉及到的知识点有：

* 组件类：如何写一个组建类。
* 组件的实例：在react中组件也可以叫做虚拟DOM。
* 组件的挂载：组件是如何渲染到视图，已经组件是如何更新视图数据。

## 4.1 理解组件化开发思想

在学习使用react开发之前，对什么是react组件，其存在的意思及其划分依据等做一个论述是很有必要的。 首先设想现在有一个webApp，这个app可以用来实现徐多功能，依据功能，我们可以把其划分为多个功能模块（以电商为例：可以分为用户模块、商品模块、店铺模块等等）。要实现这么一个又一个功能模块，可能需要把模块拆分成单一功能的独立的逻辑单元，像这样的单元就是组件。举例而言，如果把一个完整的开发拆分成这样的一个有一个的组件开发的前提下，发开一个登陆模块的话，我们先拆分成登陆、注册、登出三个小模块，以登陆为例，登录的依赖于用户的输入那就需要一个输入界面，界面就包含了诸如文本输入框，登陆按钮等，那开发一个文本输入或者一个按钮就可以看作一个组件的开发。像这样从一个webApp拆分成功能模块，再由模块拆分到组件，从组件开发到组装成一个完整的app就是组件开发思想。

## 4.2 组件之间的通信

react中的数据流向的特点是单项的，数据由父组件流向子孙组件，组件之间的通讯尤为关键，理解react的props以及state是操作react的数据的基础知识。

### 4.2.1 state

State的意思就如同的翻译（不通顺），代表一个组件的状态。可以通过setState方法改变一个组件的状态。

采用es6语法和jsx语法，16.2.0的语法，代码实例：

//index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>react-demo</title>

</head>

<body>

<div id="main"></div>

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

</body>

</html>

main.js

import React from 'react'

import ReactDom from 'react-dom'

class StateExample extends React.Component{

constructor(){

super()

this.state={

text: 'i like it'

}

}

render(){

return (<div><h3>{

this.state.text

}</h3>

<p>3秒之后会刷新状态</p>

</div>)

}

}

var t=ReactDom.render(<StateExample />,document.getElementById("main"))

console.log(t)//把挂载的组件打印控制台

// 设置定时器设定三秒之后更新组件状态

setTimeout(() => {

t.setState({

text: 'i dont like it'

})

}, 3000);

.babelrc（babel的配置文件，babel可以把jsx语法转化成可以再浏览器执行的javascript语法）

{

"presets":["react"]

}

全局安装parcel-bundler（一款代码打包工具）再在命令行（windows下打开cmd）输入命令parcel ./index.html 然后浏览打开<http://localhost:1234/>就可以看到代码效果了。

### 4.2.2 props

props是传入组件的参数。可以又父组件的state或者props传入也可以是默认参数。

将4.2.1代码中的main.js修改成如下代码：

import React from 'react'

import ReactDom from 'react-dom'

class PropsExample extends React.Component {

render() {

return (<div><h3>{

this.props.text

}</h3>

<p>3秒之后会更换props</p>

</div>)

}

}

var t=ReactDom.render(<PropsExample text={"第一次props"} />, document.getElementById("main"))

console.log(t)//把挂载的组件打印控制台

// 设置定时器设定三秒之后重新渲染

setTimeout(() => {

ReactDom.render(<PropsExample text={"第二次props"} />, document.getElementById("main"))

}, 3000);

同样的再命令行工具执行parcel ./index.html然后浏览打开<http://localhost:1234/>就可以看到代码效果了。

### 4.2.3 子父组件的通信

父组件向子组件传递消息可以同过props，这里就不举例了，子组件向父组件传递小心的也可以通过props，但是传递的是函数，或者父组件本身，例如如下代码：

把4.2.1的main.js改为如下代码：

import React from 'react'

import ReactDom from 'react-dom'

class Father extends React.Component {

constructor() {

super()

this.state = {

text: 1,

sonWords:''

}

}

render() {

return (<div><h3>父组件的状态：{

this.state.text

}</h3>

<p>子组件传来的消息:<span>{this.state.sonWords}</span></p>

<Son yourFather={this} />

</div>)

}

}

class Son extends React.Component {

updateFather(){

this.props.yourFather.setState({

text: this.props.yourFather.state.text+1

})

}

giveWordsToFather(){

//dom操作写法（不建议）

/\*

var input = document.getElementById('inp')

var words = 'value' in input?input.value:''

this.props.yourFather.setState({

sonWords: words

},function(){

input.value=''

})\*/

var input = this.refs.inp

this.props.yourFather.setState({

sonWords: input.value

}, function () {

input.value = ''

})

}

render() {

return (<div><h3>这里是子组件</h3>

<p onClick={()=>{

this.updateFather()

}}>点一下跟新我父组件的状态</p>

<input id='inp' ref="inp" type='text' /><button onClick={()=>{

this.giveWordsToFather()

}

}>传给父组件</button>

</div>)

}

}

var t = ReactDom.render(<Father />, document.getElementById("main"))

这里传递的父组件本身，也可以通过函数作用域的方式把一个可以访问父组件的函数传递给子组件，例如<Son fun={this.dosomething.bind(this)} />同样的再命令行工具执行parcel ./index.html然后浏览打开<http://localhost:1234/>就可以看到代码效果了。

## 4.3 组件的生命周期

react的核心思想是构建虚拟Dom,再把虚拟Dom渲染到视图上，通过操作虚拟Dom来完成视图的快速跟新，以及快速的响应用户的输入。那既然作为虚拟Dom那就存在着创建、更新、销毁，react给使用框架的程序员暴露了许多组件的生命周期函数。

### 4.3.1 组件的挂载

从组件类创建虚拟Dom再到虚拟Dom渲染到视图称为组件的挂载。

把4.2.1的main.js改为如下代码：

import \* as React from 'react'

import \* as ReactDom from 'react-dom'

class Demo extends React.Component {

constructor(props,content){

super(props)

console.log('component constructor')

this.state={}

}

componentWillMount(){

console.log('component will mount')

}

render(){

console.log("component render")

return (<h1>hello react</h1>)

}

componentDidMount(){

console.log('component did mount')

}

componentWillUnmount(){

console.log('component will unmount')

}

}

// 创建虚拟Dom

var virtualDom = React.createElement(Demo,null)

//挂载组件，也就是渲染视图 获得组件

var demo = ReactDom.render(virtualDom,document.getElementById('main'))

由创建虚拟Dom到组件的挂载，组件经历了constructor、componentWillMount、render、componentDidMount四个生命周期。