## Props

父传递给子组件数据，单向流动，不能子传递给父。

props的传值，可以是任意的类型。

Props可以设置默认值

HelloMessage.defaultProps = { name:”老陈”，msg：“helloworld” }

注意：props可以传递函数，props可以传递父元素的函数，就可以去修改父元素的state,从而达到传递数据给父元素。

父传子数据传递案例

|  |
| --- |
| //在父元素中使用state去控制子元素props的从而达到父元素数据传递给子元素  class ParentCom extends React.Component{      constructor(props){          super(props)          this.state = {              isActive:true          }          this.changeShow = this.changeShow.bind(this)      }      render(){          return (              <div>                  <button onClick={this.changeShow}>控制子元素显示</button>                  <ChildCom isActive={this.state.isActive} />              </div>          )      }      changeShow(){          this.setState({              isActive:!this.state.isActive          })      }  }  class ChildCom extends React.Component{      constructor(props){          super(props)      }      render(){          let strClass = null;          // if(this.props.isActive){          //     strClass = ' active'          // }else{          //     strClass = ""          // }          strClass = this.props.isActive?" active":"";          return (              <div className={"content"+strClass}>                  <h1>我是子元素</h1>              </div>          )      }  }  ReactDOM.render(      <ParentCom />,      document.querySelector("#root")  ) |

## React数据传递：子传父

调用父元素的函数从而操作父元素的数据，从而实现数据从子元素传递至父元素

|  |
| --- |
| import React from 'react';  import ReactDOM from 'react-dom';  //子传父  class ParentCom extends React.Component{      constructor(props){          super(props)          this.state = {              childData:null          }      }      render(){         return (             <div>                  <h1>子元素传递给父元素的数据：{this.state.childData}</h1>                 <ChildCom setChildData={this.setChildData} />             </div>         )      }      setChildData=(data)=>{          this.setState({              childData:data          })      }  }  class ChildCom extends React.Component{      constructor(props){          super(props)          this.state = {              msg:"helloworld"          }      }      render(){          return (              <div>                  <button onClick={this.sendData}>传递helloworld给父元素</button>                  <button onClick={()=>{this.props.setChildData('直接调用props的函数')}}>传递helloworld给父元素</button>                </div>          )      }      sendData=()=>{          //console.log(this.state.msg)          //将子元素传递给到父元素，实际就是调用父元素传递进来的父元素函数          this.props.setChildData(this.state.msg)      }  }  ReactDOM.render(      <ParentCom />,      document.querySelector('#root')  ) |

# REACT 事件

特点：

1/react事件，绑定事件的命名，驼峰命名法。

2/{},传入1个函数，而不是字符串

|  |
| --- |
| <button onClick={this.sendData}>传递helloworld给父元素</button> |

事件对象：React返回的事件对象是代理的原生的事件对象，如果想要查看事件对象的具体值，必须之间输出事件对象的属性。

注意：

原生，阻止默认行为时，可以直接返回return false；

React中，阻止默认必须e.preventDefault();

React事件传参数

|  |
| --- |
| {/\* 使用ES6箭头函数传递多个参数 \*/}  <button  onClick={(e)=>{this.parentEvent1('msg:helloworld',e)}}>提交</button>  {/\* //不使用ES6箭头函数传递多个参数的方式 \*/}  <button  onClick={function(e){this.parentEvent1('不使用es6,msg:helloworld',e)}.bind(this)}>提交</button> |

## React 条件渲染

React中条件渲染即和JavaScript中，条件运算，如if...else...三元运算符等。

1/直接通过条件运算返回要渲染的JSX对象

2/通过条件运算得出jsx对象，在将JSX对象渲染到模板中。

案例一：

|  |
| --- |
| import React from 'react';  import ReactDOM from 'react-dom';  function UserGreet(props){      return (<h1>欢迎登陆</h1>)  }  function UserLogin(props){      return (<h1>请先登录</h1>)  }  class ParentCom extends React.Component{      constructor(props){          super(props)          this.state = {              isLogin:true          }      }      render(){          if(this.state.isLogin){              return (<UserGreet></UserGreet>)          }else{              return (<UserLogin></UserLogin>)          }      }  }  ReactDOM.render(      <ParentCom></ParentCom>,      document.querySelector('#root')  ) |

案例2

|  |
| --- |
| import React from 'react';  import ReactDOM from 'react-dom';  function UserGreet(props){      return (<h1>欢迎登陆</h1>)  }  function UserLogin(props){      return (<h1>请先登录</h1>)  }  class ParentCom extends React.Component{      constructor(props){          super(props)          this.state = {              isLogin:false          }      }      render(){          let element = null;          if(this.state.isLogin){              element = <UserGreet></UserGreet>;          }else{              element = (<UserLogin></UserLogin>);          }            return (              <div>                  <h1>这是头部</h1>                  {element}                  <h1>这是三元运算符的操作</h1>                  {this.state.isLogin?<UserGreet></UserGreet>:<UserLogin></UserLogin>}                  <h1>这是尾部</h1>              </div>          )      }  }  ReactDOM.render(      <ParentCom></ParentCom>,      document.querySelector('#root')  ) |

## 列表渲染

将列表内容拼装成数组放置到模板中。将数据拼装成数组的JSX对象。

使用数组的map方法，对每一项数据按照JSX的形式进行加工，最终得到1个每一项都是JSX对象的数组，在将数组渲染到模板中。

Key值需要放置到每一项中。

简单案例

|  |
| --- |
| import React from 'react';  import ReactDOM from 'react-dom';  class Welcome extends React.Component{      constructor(props){          super(props)          this.state = {              list:[                  {                      title:"第一节 React事件",                      content:"事件内容"                  },                  {                      title:"第二节 React数据传递",                      content:"数据传递内容",                  },                  {                      title:"第三节 条件渲染",                      content:"条件渲染内容",                  }              ]          }      }      render(){          let listArr = [];          for(let i=0;i<this.state.list.length;i++){              let item = (                  <li>                      <h3>{this.state.list[i].title}</h3>                      <p>{this.state.list[i].content}</p>                  </li>              )              listArr.push(item)          }          return (              <div>                  <h1>                      今天课程内容                  </h1>                  <ul>                      {listArr}                      <li>                          <h3>这是标题</h3>                          <p>内容</p>                      </li>                  </ul>              </div>          )      }  }  ReactDOM.render(      <Welcome></Welcome>,      document.querySelector('#root')  ) |

复杂案例

|  |
| --- |
| import React from 'react';  import ReactDOM from 'react-dom';  function ListItem(props){      return (          <li>              <h3>{props.index+1}:listItem:{props.data.title}</h3>              <p>{props.data.content}</p>          </li>      )  }  class ListItem2 extends React.Component{      constructor(props){          super(props)      }      render(){          return (              <li onClick={(event)=>{this.clickEvent(                  this.props.index,                  this.props.data.title,                  event                  )}}>                  <h3>{this.props.index+1}:listItem:{this.props.data.title}</h3>                  <p>{this.props.data.content}</p>              </li>          )      }      clickEvent=(index,title,event)=>{          alert((index+1)+"-"+title)      }  }    class Welcome extends React.Component{      constructor(props){          super(props)          this.state = {              list:[                  {                      title:"第一节 React事件",                      content:"事件内容"                  },                  {                      title:"第二节 React数据传递",                      content:"数据传递内容",                  },                  {                      title:"第三节 条件渲染",                      content:"条件渲染内容",                  }              ]          }      }      render(){          let listArr = this.state.list.map((item,index)=>{              return (                  <ListItem2 key={index} data={item} index={index}></ListItem2>                )          })            return (              <div>                  <h1>                      今天课程内容                  </h1>                  <ul>                      {listArr}                      <li>                          <h3>这是标题</h3>                          <p>内容</p>                      </li>                  </ul>                  <h1>复杂没有用组件完成列表</h1>                  <ul>                  {                      this.state.list.map((item,index)=>{                          return (                              <li key={index} onClick={(event)=>{this.clickFn(index,item.title,event)}}>                                  <h3>{index+1}-复杂-{item.title}</h3>                                  <p>{item.content}</p>                              </li>                          )                      })                  }                  </ul>              </div>          )      }      clickFn=(index,title,event)=>{          alert((index+1)+"-clickFn-"+title)      }  }  ReactDOM.render(      <Welcome></Welcome>,      document.querySelector('#root')  ) |

## React制作疫情地图

疫情数据获取

|  |
| --- |
| <https://c.m.163.com/ug/api/wuhan/app/index/feiyan-data-list?t=1580892315332> |