React 构建用户界面的JavaScript库，主要用于构建UI界面。Instagram，2013年开源。

## 特点：

1. 声明式的设计
2. 高效，采用虚拟DOM来实现DOM的渲染，最大限度的减少DOM的操作。
3. 灵活，跟其他库灵活搭配使用。
4. JSX，俗称JS里面写HTML，JavaScript语法的扩展。
5. 组件化，模块化。代码容易复用，2016年之前大型项目非常喜欢react
6. 单向数据流。没有实现数据的双向绑定。数据-》视图-》事件-》数据

## 创建项目

1、通过script引入使用，仅用于学习调试使用

|  |
| --- |
| <script crossorigin src="https://unpkg.com/react@16/umd/react.development.js"></script><script crossorigin src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"></script> |

1. 通过react的脚手架，创建项目进行开发，部署。

|  |
| --- |
| 1. 安装脚手架[Create React App](https://react.docschina.org/docs/create-a-new-react-app.html#create-react-app)。   cnpm install -g create-react-app   1. 创建项目   create-react-app 01reactapp(项目名称可以自定义) |

## React元素渲染

|  |
| --- |
| let h1 = <h1>helloworld</h1>;  使用JSX的写法，可以创建JS元素对象  注意：JSX元素对象，或者组件对象，必须只有1个根元素（根节点） |

案例使用：

|  |
| --- |
| //实现页面时刻的显示  function clock(){      let time = new Date().toLocaleTimeString()      let element = (          <div>              <h1>现在的时间是{time} </h1>              <h2>这是副标题</h2>          </div>      )      let root = document.querySelector('#root');      ReactDOM.render(element,root)  }  clock()  setInterval(clock,1000) |

函数式组件渲染

|  |
| --- |
| function Clock(props){      return (                  <div>                      <h1>现在的时间是{props.date.toLocaleTimeString()} </h1>                      <h2>这是函数式组件开发</h2>                  </div>      )  }  function run(){      ReactDOM.render(          <Clock date={new Date()} />,          document.querySelector('#root')      )  }  setInterval(run,1000) |

## React Jsx

优点：

1. JSX执行更快，编译为JavaScript代码时进行优化
2. 类型更安全，编译过程如果出错就不能编译，及时发现错误
3. JSX编写模板更加简单快速。（不要跟VUE比）

注意：

1. JSX必须要有根节点。
2. 正常的普通HTML元素要小写。如果是大写，默认认为是组件。

### JSX表达式

1. 由HTML元素构成
2. 中间如果需要插入变量用{}
3. {}中间可以使用表达式
4. {}中间表达式中可以使用JSX对象
5. 属性和html内容一样都是用{}来插入内容

案例

|  |
| --- |
| import React from 'react';  import ReactDOM from 'react-dom';  import './App.css'  let time = new Date().toLocaleTimeString()  let str = '当前时间是：'  let element = (      <div>          <h1>helloworld</h1>          <h2>{str+time}</h2>      </div>  )  console.log(element)  let man = '发热';  let element2 = (      <div>          <h1>今天是否隔离</h1>          <h2>{man=="发热"?<button>隔离</button>:"躺床上"}</h2>      </div>  )  //let man = '发热';  let element4 = (      <div>          <span>横着躺</span>          <span>竖着躺</span>      </div>  )  man = '正常'  let element3 = (      <div>          <h1>今天是否隔离</h1>          <h2>{man=="发热"?<button>隔离</button>:element4}</h2>      </div>  )  let color = 'bgRed'  let logo = 'https://www.baidu.com/img/pc\_1c6e30772d5e4103103bd460913332f9.png'  //HTML的样式类名要写className,因为class在js当中是关键词  let element5 = (      <div className={color}>          <img src={logo} />          红色的背景颜色      </div>  )  ReactDOM.render(      element5,      document.getElementById('root')  ) |

## JSX\_style 样式

1. Class，style中，不可以存在多个class属性

|  |
| --- |
| <div class=’abc’ class={‘active’}> 错误的表示 |

1. style样式中，如果存在多个单词的属性组合，第二个单词开始，首字母大写。或者用引号引起来，否则会报错。

|  |
| --- |
| let exampleStyle = {      background:"skyblue",      borderBottom:"4px solid red",      'background-image':"url(https://www.baidu.com/img/pc\_1c6e30772d5e4103103bd460913332f9.png)"  } |

1. 多个类共存的操作

|  |
| --- |
| let element2 = (      <div>          <h1 className={"abc "+classStr}>helloworld</h1>      </div>  )  let classStr2 = ['abc2','redBg2'].join(" ")  let element3 = (      <div>          {/\* 这里写注释 \*/}          <h1 className={classStr2} style={exampleStyle}>helloworld</h1>      </div>  ) |

1. 注释

必须在括号的表达式内书写，否则报错：{/\* 这里写注释 \*/}

|  |
| --- |
| let classStr2 = ['abc2','redBg2'].join(" ")  let element3 = (      <div>          {/\* 这里写注释 \*/}          <h1 className={classStr2} style={exampleStyle}>helloworld</h1>      </div>  ) |

## React组件

函数式组件与类组件的区别和使用，函数式比较简单，一般用于静态没有交互事件内容的组件页面。类组件，一般又称为动态组件，那么一般会有交互或者数据修改的操作。

1. 函数式组件

|  |
| --- |
| //函数式组件  function Childcom(props){      console.log(props)      let title = <h2>我是副标题</h2>      let weather = props.weather      //条件判断      let isGo = weather=='下雨' ?"不出门":"出门"      return (          <div>              <h1>函数式组件helloworld</h1>              {title}              <div>                  是否出门？                  <span>{isGo}</span>              </div>          </div>      )  } |

1. 类组件

|  |
| --- |
| //类组件定义  class HelloWorld extends React.Component{      render(){          console.log(this)          return (              <div>                  <h1>类组件定义HELLOWORLD</h1>                  <h1>hello:{this.props.name}</h1>                  <Childcom weather={this.props.weather} />              </div>          )      }    } |

3、复合组件：组件中又有其他的组件，复合组件中既可以有类组件又可以有函数组件。

|  |
| --- |
| import React from 'react';  import ReactDOM from 'react-dom';  import './04style.css';  //函数式组件  function Childcom(props){      console.log(props)      let title = <h2>我是副标题</h2>      let weather = props.weather      //条件判断      let isGo = weather=='下雨' ?"不出门":"出门"      return (          <div>              <h1>函数式组件helloworld</h1>              {title}              <div>                  是否出门？                  <span>{isGo}</span>              </div>          </div>      )  }  //类组件定义  class HelloWorld extends React.Component{      render(){          console.log(this)  //返回的都是JSX对象          return (              <div>                  <h1>类组件定义HELLOWORLD</h1>                  <h1>hello:{this.props.name}</h1>                  <Childcom weather={this.props.weather} />              </div>          )      }    }    // ReactDOM.render(  //     <Childcom weather="出太阳" />,  //     document.querySelector('#root')  // )  ReactDOM.render(      <HelloWorld name="老陈" weather="下雨" />,      document.querySelector('#root')  ) |

## React State

相当于VUE的DATA,但是使用方式跟VUE不一致。