[React Element /组件/JSX](https://www.cnblogs.com/SamWeb/p/6740448.html)

　　初学react，想要了解React 是什么，直接用script标签 引入React就可以了，不过需要引入两个库：React 和ReactDom，React 负责创建React element，ReactDom 则是负责把React创建出来的element， 通过调用DOM API， 创建出真实的DOM 元素，这样浏览器就可能根据DOM渲染出页面了。模板如下

[复制代码](javascript:void(0);)

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8" />

<title>Hello World</title>

<!-- 引入React 和 ReactDOM -->

<script src="https://cdn.bootcss.com/react/16.1.0/umd/react.development.js"></script>

<script src="https://cdn.bootcss.com/react-dom/16.1.0/umd/react-dom.development.js

"></script>

</head>

<body>

<div id="root"></div>

<script>

// react 代码写在这里

</script>

</body>

</html>

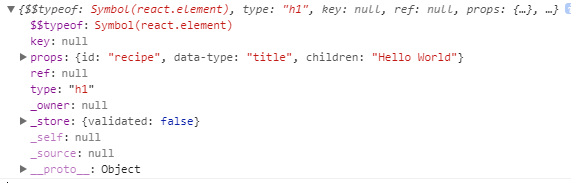
[复制代码](javascript:void(0);)

**React Element**

　　一个普通的JavaScript 对象，用来描述真实的DOM 长什么样子，可以通过React 库提供的createElement() 函数来创建。createElement 函数接受三个参数，第一个是type, 要创建什么类型的element,  第二个是properties, 这个element 有哪些属性，第三个参数是children, 这个element有哪些子元素。

let h1Elem = React.createElement('h1', {id: 'recipe', 'data-type': 'title'}, 'Hello World');

　这时打开浏览器, console.log(h1Elem)  一下, 看一看这个刚刚创建的React element 是不是一个普通的对象。



　　它是一个Js 对象，拥有props，type等等属性， 但是通过它，也确实描述了 一个DOM元素 长什么样子，<h1 id='recipe', data-type='title'>Hello Word</h1>,所以它也称之为virtual DOM, 用于指导构造真正的DOM 元素.

　　当然，createElement 不仅能创建一个element, 还可以创建一个层层嵌套的element树， 这主要在于函数可以接受任意个参数，第三个参数及其以后的参数，都会当成element 的children

[复制代码](javascript:void(0);)

let ulElem = React.createElement(

'ul',

null ,

React.createElement('li', null, 'JS'),

React.createElement('li', null, 'React'),

React.createElement('li', null, 'Redux')

);

[复制代码](javascript:void(0);)

　　　此时也可以console.log(ulElement), 也可以看到它是一个嵌套的js 对象。

[复制代码](javascript:void(0);)

{

type: 'ul',

props: {

children: [

{

type: 'li',

props: {children: 'JS'}

},

{

type: 'li',

props: {children: 'React'}

},

{

type: 'li',

props: {children: 'Redux'}

}

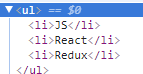
]

}

}

[复制代码](javascript:void(0);)

　　通过ulElem，我们也可以很清楚地知道页面上将要展示什么，就是下面的ul-li，



　　创建完React element, 就需要用到ReactDom 了。因为React element只是js 对象，只是描述了真实的DOM长什么样子，最后还是要创建真实的DOM，否则浏览器无法渲染。ReactDom提供了一个render 方法， 可以把React element(虚拟DOM)转化成真实的DOM

ReactDOM.render(h1Elem, document.getElementById('root'));

　　第一个参数是React element, 第二个参数是渲染到什么地方， ReactDom render 相当于执行下面的操作

let h1 = document.createElement('h1');

h1.setAttribute('id', 'recipe');

h1.setAttribute('data-type', 'title');

h1.textContent = 'Hello World';

document.getElementById('root').appendChild(h1);

　　这时我们刷新浏览器，可以h1. 如果React element 像ul 这样包含children的话，ReactDom render 将执行递归操作，依次创建DOM。

**React Component**

当我们写大量的React element，你会发现有些代码可以共用，就像上面创建的ulElem，可以用到很多地方，这时我们就想把它们封装起来，这就形成了React component组件。React 提供了两种创建组件的方法，类式和函数式。

　　类式，就是利用es6 class 语法, 所有的组件都继承自React.Component，在render 函数中返回React Element

[复制代码](javascript:void(0);)

class Content extends React.Component {

render() {

return React.createElement('ul', null ,

React.createElement('li', null, 'JS'),

React.createElement('li', null, 'React'),

React.createElement('li', null, 'Redux')

);

}

}

[复制代码](javascript:void(0);)

　　函数式，就是一个函数，返回React element

[复制代码](javascript:void(0);)

const Content = () => {

return React.createElement('ul', null,

React.createElement('li', null, 'JS'),

React.createElement('li', null, 'React'),

React.createElement('li', null, 'Redux')

)

}

[复制代码](javascript:void(0);)

　　我们声明了组件以后，怎么使用这个组件呢？ 这时要注意，我们的组件名称(Content)，只是相当于createElement 函数中的第一个参数type, 它不是一个React element, 它只是返回一个React element, 我们仍然需要调用React.createElement 来创建一个React element

// 我们创建的组件名Content, 只是相当于createElement函数中的第一个参数type, 它是相当于h1的type, 而不是一个React Elemnet. 利用这个type，仍需要创建组件。

let content = React.createElement(Content, null, null);

ReactDOM.render(content, document.getElementById('root'));

　　这时页面中同样显示三个li, 表示成功。创建组件，只是把一段可以重用的React element 进行封装，从而创建一个自定义的type, 然后再利用该type， 随处都可以创建element 元素，进而达到重用的目的。

**React 组件数据**

以上创建React 组件的方式有一个问题，就是数据是固定的，能不能把数据和 UI分离，不同的数据渲染出不同的内容，从而使组件更加通用？ 这是可以的，当我们利用组件创建react element 的时候，第二个参数是null,

let content = React.createElement(Content, null, null);

　　可以利用这个参数向组件内传递数据，因为这个参数就是表示这个组件的属性，它的形式也是一个键值对的形式，比如我们把JS, React ,Redux 数据提出来，形成一个数组，

const item = ['JS', 'React', 'Redux'];

// 向组件中传递一个数组数据 item

let content = React.createElement(Content, {item: item}, null);

　　那么我们组件中怎么获取到这个数据呢？类式组件是通过this.props获取的，而函数式组件则是参数的形式获取

　　先看类式的组件， 在组件中通过this.props 来获取， 我们可以把this.props 找印出来看一下

[复制代码](javascript:void(0);)

class Content extends React.Component {

render() {

// 打印props;

console.log(this.props); // {item: Array(3), children: null}

return React.createElement('ul', null ,

React.createElement('li', null, 'JS'),

React.createElement('li', null, 'React'),

React.createElement('li', null, 'Redux')

);

}

}

[复制代码](javascript:void(0);)

　　可以看到this.props 获取到了我们传递的item 数组，那我们就可以直接使用了数组数据了，这时通过数组的map 渲染li

[复制代码](javascript:void(0);)

class Content extends React.Component {

render() {

return React.createElement('ul', null ,

// this.props.item 获取到传递过来的数据

this.props.item.map((item, index) =>

React.createElement('li', {key:index}, item)

)

);

}

}

[复制代码](javascript:void(0);)

　　函数式的组件，则是它自动会获取props作为参数，组件中直接使用props.items 获取到数据

[复制代码](javascript:void(0);)

// 自动获取props作为参数。

const Content = (props) => {

return React.createElement('ul', null,

props.item.map((item, index) =>

React.createElement('li', {key:index}, item)

)

)

}

[复制代码](javascript:void(0);)

**JSX**

　　在上面的代码中，我们每创建一个React Element 都要调用一次React.createElement 函数，非常繁琐，并且它想表达式的意思只是一个类html的元素，再来看一下我们创建的h1Elem element

let h1Elem = React.createElement('h1', {id: 'recipe', 'data-type': 'title'}, 'Hello World');

　　它实际上表达的意思就是 <h1 id=’recipe’  data-type=’title’ >Hello World</h1>， 如果我能在代码中直接写h1 就好了。这就是JSX 语法， 可以直接在js 代码中写类html 的语法。React 把createElement 函数作了进一步的封装，提供了JSX语法。

　　在createElement 函数中，它的第一个参数是type，表示创建什么类型，而在html中，表示什么类型直接用html 标签，<h1></h1>  它就表示h1 类型， 第二个参数表示属性，元素有哪些属性，而在html标签中，有什么属性，就直接写在它的标签中，有多少，写多少, 如 <h1 id='recipe' class='title'></h1>. 第三个参数是children, 在html中表示children更简单，直接写在两个标签内部的内容都是children. 这样一一对应以后，就可以理解JSX 写法的用意了，在心理上写起来就比较舒服了，因为明白了。

　　对于组件来说，它也是一样的，因为组件名称，只是一个type, 仍然需要调用createElement 函数来 创建React Element 元素， 只要使用createElement 的地方，我们都可以使用类html 语法，如Content组件，<Content></Content>  就表示创建了一个element了。它的属性或children和上面的h1 用法一致。对于Content 组件，如果没有 chilren 属性，可以直接写单标签<Content />.   现在用JSX的语法来书写Content 组件。

[复制代码](javascript:void(0);)

class Content extends React.Component {

render() {

return (

<ul>

{

this.props.item.map((item, index) =>

<li key={index}>{item}</li>

)

}

</ul>

)

}

}

// 向组件中传递一个数组数据 item

let content = <Content item ={item}></Content>

[复制代码](javascript:void(0);)

　　无论是在组件属性，还是元素属性中，我们都使用了{}，如key={index}. 在jsx 中, {}里面的所有东西都当作js表达式进行解析， React 会把里面的内容进行求值计算。只要写表达式，我们都要用{} 括起来。向组件中传递一个数字1，我们就要写 num = {1}， 传递一个布尔值，就要写 bool={false}。字符串除外，它可以直接写。比如可以直接写如name="sam"。

[复制代码](javascript:void(0);)

// 组件添加了三个p, 有来接受数据

class Content extends React.Component {

render() {

return (

<section>

<ul>

{

this.props.item.map((item, index) =>

<li key={index}>{item}</li>

)

}

</ul>

<p>name 的值是{this.props.name}，类型是 {typeof this.props.name}</p>

<p>bool 的值是{this.props.bool}，类型是 {typeof this.props.bool}</p>

<p>num 的值是{this.props.num}，类型是 {typeof this.props.num}</p>

</section>

)

}

}

// 向组件中另外传递 字符串name ，布尔值bool， 一个数字num。

let content = <Content item ={item} name="sam" bool={false} num={1}></Content>

[复制代码](javascript:void(0);)

　　这时刷新浏览器，可以看到报错了，首先，JSX语法，浏览器是不支持的，我们要把它转换成JS， 所以引入babel库，在head 标签中引入

<script src="https://unpkg.com/babel-standalone@6.15.0/babel.min.js"></script>

　　其次我们要告诉babel， 我们的代码需要转译，所以在写react 代码的script 标签上添加一个type 属性

<script type="text/babel">

　　这时再刷新浏览器，没有问题了，我们也获得了属性，并且它的类型也是对的，num 是Number, 字符串是String。

　　但是这里有一个问题，就是如果我们要传递很多属性，这么 一个一个列出来，非常麻烦，怎么办呢？ 这时可以使用对象，但如果用对象进行传值，又不符合 属性名=属性值的写法，这时要用到es6中的扩展运算符..., React 对es6 中的扩展运算符（…）进行扩展，它能运用到对象上，对对象进行分割。｛…obj｝； var obj = {name:”sam”, num: 1}  , …obj  => name=”sam” , num= 1, 注意，...obj 是一个表达式，仍需要把它用{} 括起来

[复制代码](javascript:void(0);)

<script type="text/babel">

// 组件添加了三个p, 有来接受数据

class Content extends React.Component {

render() {

return (

<section>

<ul>

{

this.props.item.map((item, index) =>

<li key={index}>{item}</li>

)

}

</ul>

<p>name 的值是{this.props.name}，类型是 {typeof this.props.name}</p>

<p>bool 的值是{this.props.bool}，类型是 {typeof this.props.bool}</p>

<p>num 的值是{this.props.num}，类型是 {typeof this.props.num}</p>

</section>

)

}

}

// 要把传递的属性写到一个对象中，

const obj = {

item : ['JS', 'React', 'Redux'],

name: 'sam',

bool: false,

num: 1

}

// 把对象进行分割

let content = <Content {...obj}></Content>

ReactDOM.render(content, document.getElementById('root'));

</script>

[复制代码](javascript:void(0);)

　　在Content 组件中，你可能发现外层包了一个section, 这是因为所有的组件都返回一个单一的根节点，主要还是createElemet 函数第一个属性type 是一个值， 不能接受多个值。

　　你可能还发现所有的组件名是大写，还是因为createElement 的第一个参数type, type 有两种类型，一种是html原有的类型如h1, 一种是自定义的类型，就是component, 当我们传入时，React  无法区分这两种类型，所以它用大小写进行区分。如果是小写，它就以为是html原有的类型，如果是大写，就是自定义类型。如果我们组件使用了小写，React 按html原有的类型进行渲染，但是它又找不到这个类型，所以什么都不会渲染，组件名必须大写。

 　　在JSX语法中， 我们还要注意以下两点：

　　html 属性关键字： 如 我们可以给html元素添加属性<h1 class="book">, 但class 在js 中是关键字， 所以class 要变成 className. 由于JSX 最终会转换成原生js 函数，所以js中的一些关键字在JSX中是不能用的，如class, for. 但在JSX 的类html 模版中，html 元素属性中又有class 和for, 这就冲突了。React 对html 元素中有冲突的属性进行了重新命名，for 变成了htmlFor, class 变成了className.  所有变量的命名都要用 驼峰命名法。如label 元素 <label htmlFor=”input” className=”text”></label>

　　样式：在JSX 中，给一个html 元素添加样式有两种方法，一种是上面提到的className,  它的取值是一个样式名字符串，一种是内联样式style, 它的取值必须是一个对象。 <div className=”col-md-3” style ={style}></div> ，style 是组件内部定义的一个对象变量。 因为render 是一个函数，里面可以声明变量

[复制代码](javascript:void(0);)

class Content extends React.Component {

render() {

// 定义样式变量

　　var inlineStyle = {

color: 'green' ,

　　　// css3 一些属性有些需要带浏览器厂商前缀，这时厂商前缀首字母必须大写, 所有的样式都是字符串

　　　WebkitFilter: blur('5px')

　　}

return (

<section>

<p style={inlineStyle}>name 的值是{this.props.name}，类型是 {typeof this.props.name}</p>

</section>

)

}

}

react总结

1. react 渲染

react中元素就是一个普普通通的对象，与浏览器DOM不同，利用reactDOM 可以确保创建虚拟dom来渲染。

在开发过程中定义一个根节点

React的渲染方式：ReactDOM.render() 两个参数：内容；渲染目标js对象

创建方式

1. script CDN 引入
2. react脚手架，创建项目
3. cnpm i -g create-react-app
4. creact-react-app my-project
5. react JSX

有点：1jsx执行更快，编译js代码时进行了优化;类型更安全，及时发现错误；JSX编写模板更简单快速

注：JSX必须要有根节点，，正常的html元素要小写。

2.JSX表达式

·有HTML元素组成；中间有需要插入变量用{}；{}中间可以使用表达式；{中间表达式中可以使用JXS对象 }；属性可html内容一样都是{}来插入。

3.JSX——style样式

(1)样式驼峰命名；

(2)Class 的style中不可以出现多个class属性。

(3)注释{ /\* \*/}

4.react 组件

（1）函数式组件 无状态

Function Child(){}

<Child/>

1. 类组件 render的方法渲染返回，一般用于有状态

Class HelloWord extends React.Component{

render(){

Return()

}

}

1. react State

相当于vue的data，但是使用方式不一致