React

主讲：崔译

一、简介

[首页]
[文档]
[中文文档]

> A JavaScript library for building user interfaces
> 用于构建用户界面的Javascript 库

- 是 JS 库
- 用于 构建 HTML ==> 在 React 项目中， 不存在 HTML， 纯 JS 代码

二、HelloWorld

1、安装react 以及 babel

```
<!--使用CDN安装-->
<!--development 开发  开发版-->
<!--React的核心库，提供关于React的核心API-->
<script crossorigin src="https://unpkg.com/react@16/umd/react.development.js"></script>
<!--React 的动态渲染库，提供的是对DOM的操作-->
<!-- 依赖于核心库 -->
<script crossorigin src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"></script>
<!--production 生产  相当于之前接触的 min 版 -->
<script crossorigin src="https://unpkg.com/react@16/umd/react.production.min.js"></script>
<script crossorigin src="https://unpkg.com/react-dom@16/umd/react-dom.production.min.js"></script>
```

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

NPM 安装

```
sudo npm install react
sudo npm install react-dom
```

2、编写代码

```
<!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>Document</title>
    <!--导入顺序-->
    <script src="./js/react.development.js"></script>
    <script src="./js/react-dom.development.js"></script>
    <script src="./js/babel.min.js"></script>

</head>
<body>
    <div id="root"></div>
    <!--标签位置、type值-->02-
    <script type="text/babel">
        ReactDOM.render(
            <h1>Hello React</h1>,
            document.getElementById("root")
        );
     </script>
</body>
</html>
```

三、JSX

> https://github.com/facebook/jsx
> JSX is an XML-like syntax extension to ECMAScript without any defined semantics.
> 类似于XML语法扩展，在ES标准中并没有任何的语义定义

```
<!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>Document</title>
    <script src="./js/react.development.js"></script>
    <script src="./js/react-dom.development.js"></script>
    <script src="./js/babel.min.js"></script>
</head>
<body>
    <abc>asdasdas<</abc>
    <div id="root"></div>
    <script  type="text/babel">

        let formatUser = (user)=> `hello,${user.firstName} . ${user.lastName} `;

        let user = {
            firstName:'隔壁',
            lastName:'老王'
        }

        const element  = (
            <div>
                <abc>{formatUser(user)} </abc>
                <h1>{formatUser(user)}</h1>
                <h1>{1+2}</h1>
            </div>
        );

        ReactDOM.render(
            element,
            document.getElementById("root")
        );
    </script>

</body>
</html>
```

3-1 XML

- 可扩展标记语言
- 语法类似于HTML
- 特点

- 有且只能有一个**根标签**
- 标签名任意，属性名任意
- 标签属性 **区分大小写**
- 标签一定要闭合

```
html 对 <input type='text'>
xml 对 <input type='text' />  xml 错 <input type='text' >
```

3-2 JSX 解析规则

在 JSX 中，可以嵌入 javascript 的基本语法 和 函数的调用， 但是 **不能使用结构化的代码，分支，循环等等**, 如 `if`, `for`, `while`.....

解析规则

1. 标签如果是 `<小写字母开头>`, 使用html规则进行解析
2. 标签如果是 `<大写字母开头>`, 使用React的 Component 规则解析
3. 对于JSX中的代码块 `{xxxx}`, 使用javascript 规则解析
4. 在JSX 的代码块中，一定要小心 **`false`**, **`null`**, **`undefined`**, **`true`**, 以上值 都是 React 的 有效 子代，不会直接被渲染（即：直接使用，在页面中无法显示），可以使用 类型转换 将其转换成 `string` 类型
5. 在JSX 的代码块中，不能直接写对象类型，使用`toString` 方法
6. 不能使用结构化的代码，分支，循环等等

> 练习： 随机产生两个[0,10]的整数,
> 在页面中显示 第一个数为：4， 第二个数为：7， 第一个数大于第二个数： false,

```
<!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>Document</title>
    <script src="./js/react.development.js"></script>
    <script src="./js/react-dom.development.js"></script>
    <script src="./js/babel.min.js"></script>
</head>
<body>
    <pre>
            练习： 随机产生两个[0,10]的整数,
            在页面中显示 第一个数为：4， 第二个数为：7， 第一个数大于第二个数: false,
    </pre>
    <div id="root"></div>
```

```
18
19
20      <script  type="text/babel">
21          const nums = {
22              m : Number.parseInt(Math.random()*11),
23              n : Number.parseInt(Math.random()*11)
24          }
25          // console.log(nums.m)
26
27          const element  = (
28              <div>
29                  <h1>第1个数m为:{ nums.m }</h1>
30                  <h1>第2个数n为:{ nums.n }</h1>
31                  <h1>m > n:{ nums.m > nums.n ? "true" : "false"}</h1>
32                  <h1>m > n:{ "" + (nums.m > nums.n) }</h1>
33                  <h1>m > n:{ String(nums.m > nums.n) }</h1>
34                  <h1>m > n:{ (nums.m > nums.n).toString() }</h1>
35              </div>
36          );
37
38          ReactDOM.render(
39              element,
40              document.getElementById("root")
41          );
42      </script>
43
44  </body>
45  </html>
```

3-3 数组的展开渲染（循环）

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <meta name="viewport" content="width=device-width, initial-
    scale=1.0">
6       <meta http-equiv="X-UA-Compatible" content="ie=edge">
7       <title>Document</title>
8       <script src="./js/react.development.js"></script>
9       <script src="./js/react-dom.development.js"></script>
10      <script src="./js/babel.min.js"></script>
11  </head>
12  <body>
```

```
    <pre>
            练习：  实现将用户数组变成ul->li 展示在页面上，
            li中的内容是 姓名-年龄
    </pre>
    <div id="root">

    </div>


    <script  type="text/babel">

        const userList = [
            {name:'张三',age:22},
            {name:'张思',age:33},
            {name:'彰武',age:44},
            {name:'张柳',age:55},
        ];

        let fun = () => {
            // let arr = [];
            // for(let i = 0 ; i < userList.length ; i++)
            // {
            //     arr.push(<li>{userList[i].name}-{userList[i].age}</li>);
            // }

            // map 方法是js数组中的一个方法，
            // 参数是一个函数callback，
            //      作用：  会遍历原数组中的所有元素，对每一个元素调用callback方法，
            //              callback方法 第一个参数是每次遍历的数组元素
            //                           第二个参数是数组元素的  索引（下标index）
            //              callback方法 具有返回值，map 方法会返回 每次调用callback方法返回的值 组成的数组
            /*
                        例如：
                            let arr = [1,2,3]
                            let newArr = [];
                            for(let item of arr)
                            {
                                // 次方运算 item的2次方
                                newArr.push(item ** 2);
                            }
                            console.log(newArr);
                        等效于
                            let arr = [1,2,3];
                            let newArr = arr.map((item) =>  item ** 2);

```

```
            */
            let arr = userList.map((user,i) => <li key={i}>{user.name}-{user.age}-{i}</li>);
            return arr;
        }

        const element  = (
            <ul>
               {fun()}
            </ul>
        );

        ReactDOM.render(
            element,
            document.getElementById("root")
        );
    </script>

</body>
</html>
```

3-4 条件判断

```
<!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>Document</title>
    <script src="./js/react.development.js"></script>
    <script src="./js/react-dom.development.js"></script>
    <script src="./js/babel.min.js"></script>
</head>
<body>

    <pre>
        用户输入一个学生成绩，prompt，
        [0,60)分 页面显示不及格
        [60-80)显示及格
        [80-90)显示良好
        [90-100]显示优秀
    </pre>

    <div id="r"></div>

    <script type="text/babel">
```

```
25
26          let calcScore = () => {
27              // let score = prompt('') * 1;
28              let score = Number.parseInt(prompt("请输入学生成绩",0) / 10);
29              let result = "不及格";
30              switch(score){
31                  case 6:
32                  case 7:
33                      result = "及格"
34                      break;
35                  case 8:
36                      result = "良好"
37                      break;
38                  case 9:
39                  case 10:
40                      result = "优秀"
41                      break;
42              }
43              return result;
44          }
45
46          let element = (
47             <p> {calcScore()} </p>
48          )
49
50          ReactDOM.render(
51              element,
52              document.querySelector("#r")
53          );
54      </script>
55  </body>
56  </html>
```

3-5 属性的绑定

1. HTML 属性的绑定 直接使用字符串渲染

```
1  let hl = 'aaa';
2  <span itany='abc' helloworld={hl} > {result} </span>;
```

2. class 属性 使用DOM属性进行绑定 （class —> className）

```
1  let cls = 'danger';
2  <span className={cls} > {result} </span>;
```

3. style 属性 使用 js 对象进行绑定 （**js对象中的css样式名使用aaaBbbCcc的方式**）

```
 let style = {
      color:'red',
      fontSize:'60px'
 }
<span style={style}> {result} </span>
{
    //我是注释
    /*我是注释*/
    /**我是注释*/
}
<span style={{color:'red'}}>aaaa</span>
```

四、元素渲染

> React elements are immutable. Once you create an element, you can't change its children or attributes. An element is like a single frame in a movie: it represents the UI at a certain point in time.
> React 元素都是immutable 不可变的。当元素被创建之后，你是无法改变其内容或属性的。一个元素就好像是动画里的一帧，它代表应用界面在某一时间点的样子。

```
<!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>Document</title>
    <script src="./js/react.development.js"></script>
    <script src="./js/react-dom.development.js"></script>
    <script src="./js/babel.min.js"></script>
    <style>
        .danger{
            color: cadetblue;
        }
    </style>
</head>
<body>

    <div id="r"></div>

    <script type="text/babel">



        // setInterval(()=> {
```

```
        //     curTime = new Date().toLocaleString()
        //     console.log(curTime)
        // },1000);

        let load = ()=>{
            let curTime = new Date().toLocaleString();

            let element = (
                <div>
                    <p>{curTime}</p>
                    <img src={"https://www.baidu.com/img/bd_logo1.png?where=super&d="+new Date()} />
                    <p>
                        aksdjkladjkladjklasdjkladjkalsdjkl
                        aksdjkladjkladjklasdjkladjkalsdjkl
                        aksdjkladjkladjklasdjkladjkalsdjkl
                        aksdjkladjkladjklasdjkladjkalsdjkl
                        aksdjkladjkladjklasdjkladjkalsdjkl
                        aksdjkladjkladjklasdjkladjkalsdjkl
                        aksdjkladjkladjklasdjkladjkalsdjkl
                    </p>
                </div>
            )

            ReactDOM.render(
                element,
                document.getElementById("r")
            );
        }
        setInterval(load,1000);
    </script>
</body>
</html>
```

> React DOM compares the element and its children to the previous one, and only applies the DOM updates necessary to bring the DOM to the desired state.
> React DOM 首先会比较元素内容先后的不同，而在渲染过程中只会更新改变了的部分。

但是

> in practice, most React apps only call `ReactDOM.render()` once.
> 在实际生产开发中，大多数React应用只会调用一次 `ReactDOM.render()`

五、组件

- React 中的组件 实际上是一个符合React规范的 javascript 对象

- React 组件的名称 **首字母必须大写**
- React 组件 **有且只有一个根标签**

5-1 使用函数定义React组件

```
<!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>Document</title>
    <script src="./js/react.development.js"></script>
    <script src="./js/react-dom.development.js"></script>
    <script src="./js/babel.min.js"></script>
</head>
<body>

    <div id="d"></div>



    <script type="text/babel">
        // 定义组件
        // 组件名首字母大写
        // 组件函数接收一个参数，参数名一般情况下为props，是所有组件参数构成的对象
        function ComA(props)
        {
            let user = props.user;
            console.log(props['hello-world']);
            return (
                <div>
                    <table>
                        <tbody>
                            <tr>
                                <td>姓名</td>
                                <td>年龄</td>
                            </tr>
                            <tr>
                                <td>{user.name}</td>
                                <td>{user.age}</td>
                            </tr>
                        </tbody>
                    </table>
                </div>
            )
        }

```

```
44
45          const element = (
46              <div>
47                  <ComA user={{name:'z3',age:22}} abc='123' hello-
    world="haha"></ComA>
48                  <ComA user={{name:'z4',age:223}} abc='234'></ComA>
49                  <ComA user={{name:'z5',age:224}} abc='2345'></ComA>
50              </div>
51          )
52
53          ReactDOM.render(
54              element,
55              document.getElementById('d')
56          );
57
58      </script>
59  </body>
60  </html>
```

5-2 使用React.createClass() 创建组件

该方法在16以上版本中 被废弃

5-3 使用ES6 中的class 创建组件

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <meta name="viewport" content="width=device-width, initial-
    scale=1.0">
6       <meta http-equiv="X-UA-Compatible" content="ie=edge">
7       <title>Document</title>
8       <script src="./js/react.development.js"></script>
9       <script src="./js/react-dom.development.js"></script>
10      <script src="./js/babel.min.js"></script>
11  </head>
12  <body>
13
14      <div id="d"></div>
15
16
17
18      <script type="text/babel">
19
20          // 使用ES6 的class 定义组件
21          class ComA extends React.Component {
22              render(){
```

```
                // 返回的是 JSX
                // 组件参数的获取方式 this.props.参数名
                return <p>{this.props.user.name}</p>
            }
        }

        // import {ComB} from "./component"

        const element = (
            <div>
                <ComA user={{name:'z3',age:22}} abc='123' hello-world="haha"></ComA>
                <ComA user={{name:'z4',age:223}} abc='234'></ComA>
                <ComA user={{name:'z5',age:224}} abc='2345'></ComA>
                {
                    //<ComB data="hehe"/>
                }
            </div>
        )

        ReactDOM.render(
            element,
            document.getElementById('d')
        );

    </script>
</body>
</html>
```

六、React 中事件的绑定

React 中事件的绑定方式类似于DOM，但是事件名需要使用 驼峰命名法

```
<!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>Document</title>
    <script src="./js/react.development.js"></script>
    <script src="./js/react-dom.development.js"></script>
    <script src="./js/babel.min.js"></script>
</head>
<body>
```

```
13
14      <div id="d"></div>
15
16
17
18      <script type="text/babel">
19
20          // 使用ES6 的class 定义组件
21          class ComA extends React.Component {
22              msg='hehe'
23              // constructor(){
24              //      // 是一个函数，调用父类的构造函数
25              //      // 详见10-extends.html
26              //      super();
27              //      // console.log('cons.....')
28              //      // console.log(this,this.msg);
29              //      this.func();
30              // }
31              handler(){
32                  console.log(this)
33              }
34              handlerWithThis(){
35                  console.log(this.msg)
36              }
37              render(){
38                  // 事件的绑定
39                  // 事件名 使用驼峰命名法：  首字母小写，从第二个单词开始每个单词首
字母大写
40                  // 值是 class中的一个方法
41                  return (
42                      <p>
43                          {/*使用该方式绑定的事件处理函数中，this 是
undefined*/}
44                          <button onClick={this.handler}>我是一个按钮
</button>
45                          {/*为JSX事件绑定this对象,this.事件名
称.bind(this)*/}
46                          <button onClick=
{this.handlerWithThis.bind(this)}>我是一个有this的按钮</button>
47                      </p>
48                  )
49              }
50          }
51
52
53          const element = (
54              <div>
55                  <ComA user={{name:'z3',age:22}} abc='123' hello-
world="haha"></ComA>
```

```
            </div>
        )

        ReactDOM.render(
            element,
            document.getElementById('d')
        );

    </script>
</body>
</html>
```

```
class SomeClass{
    constructor(name){
        console.log('someClass const....')
        this.name = name;
    }
}

class OtherClass extends SomeClass{
    // 创建子类对象，如果子类中没有显式的声明构造方法，会默认调用父类的构造方法

    // 如果显式的声明了 构造方法，需要手动的在访问（使用）this之前，使用super() 调用父类构造方法
    constructor(){
        console.log(1111);
        // 特指父类的构造方法
        // super 方法创建并返回this对象
        console.log(super('zhangsan') === this);
        this.age = 22;

    }
}

let oth = new OtherClass();
console.log(oth);
```

作业

```
页面提供输入框，
当用户向输入框中输入内容时，在控制台实时的打印用户输入的内容
abcdef
a
ab
abc
abcd
abcdef
```