一、React简介



来自Facebook，2013诞生。

和Angular不同，React并不是一个完整的MVC/MVVM框架，它只专注于提供清晰、直接的View视图层的解决方案。它的功能全部以构建组件视图为核心，并提供类似控制器的函数接口和生命周期函数，在React中没有控制器、没有服务、没有指令、没有过滤器等等。

React非常轻，目标是Minimal API interface，只提供组件化相关的非常少量的API。简化了一切可以简化的东西。比如说Angular中做一个组件，实际上叫做E级别的指令，要写scope、link函数，但是React最大限度的简化了一个组件的开发，事实上React就是为了组件开发而诞生的。

React可以搭配Flux等架构（Redux实现）使用。有Flux架构护航，数据的单向流动，自称是比MVC、MVVM更高级的一种模式，可以开发大型应用的。

React使用Virtual DOM技术，使得它性能卓越，开销很小。

Virtual DOM是React中一个很重要的概念，在日常开发中，前端工程师们需要将后台的数据呈现到界面中，同时要能对用户的操作提供反馈，作用到UI上……这些都离不开DOM操作。但是我们知道，频繁的DOM操作会造成极大的资源浪费，也通常是性能瓶颈的原因。于是React引入了Virtual DOM。Virtual DOM的核心就是计算改变前后的DOM区别，然后用最少的DOM操作语句对DOM进行操作。

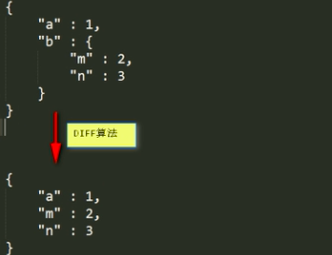
比如说要进行这样的DOM转换：

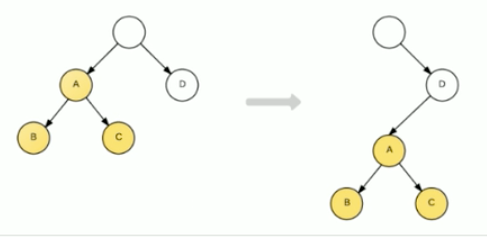


不用Virtual DOM技术：删除4个li，插入5个li。

使用Virtual DOM技术：通过计算，我们只需要改变4个li的innerHTML，并且插入一个li即可。

Virtual DOM技术使用了DIFF算法，DIFF算法是一个比较计算层次结构区别的算法。主要用来计算DOM之间的差异。

用DIFF算法可以计算两个JSON结构的区别。现在DIFF算法为Virtual DOM技术服务，用来比较DOM结构的差异。

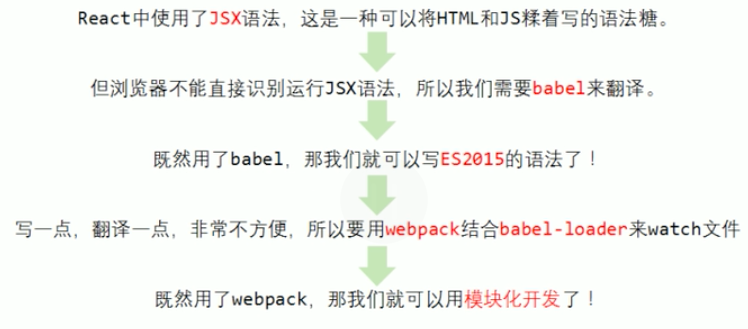


如果不适用Virtual DOM技术，此时要删除3个节点，插入3个节点。使用Virtual DOM技术仅仅需要移动节点即可。

React中使用JSX语法糖，JSX = Javascript + XML。js文件中写HTML模板，代码语境不需要来回切换的。也就是说，在Angular中模板卸载.html文件中。React中使用.js文件来当做HTML模板，语境是统一的，非常清晰简明。JSX语法比’’运算符还要简单，是最优雅的将HTML和JS结合的方案。

👂二、React配置和Hello World

2.1 React配置



我们创建一个项目文件夹01\_HelloWorld，在这个文件夹，配置webpack + babel环境。让webpack可以指导babel翻译ES6语法。

创建package.json项目身份证：

$ cnpm init

安装webpack，并且设置为项目依赖：

$ cnpm install --save-dev webpack

当然你必须之前已经在-g下安装过webpack。

我们创建一个webpack.config.js文件，这个文件是webpack工作的参考。

我们对着<http://webpack.js.org/configuration/> 这个文档来创建webpack.config.js文件。

webpack.config.js文件：

var path = require('path');

module.exports = {

entry: "./app/main.js"

output: {

path: path.resolve(\_\_dirname, "dist"),

filename: 'all.js'

}

}

至此，我们的webpack就已经可以进行标准的CMD模块开发了。

我们引入babel翻译ES6:

我们参考 <http://github.com/babel/babel-loader> 改变webpack.config.js文件。

var path = require('path');

module.exports = {

entry: "./app/main.js"

output: {

path: path.resolve(\_\_dirname, "dist"),

filename: 'all.js'

}

module: {

rules: [

{

test: /\.js$/,

exclude: /(node\_modules|bower\_components)/,

use: {

loader: 'babel-loader',

options: {

presets: ['es2015']

}

}

}

]

}

}

友情提醒：webpack1和2在配置上差别最大的就是loader的书写方式。Module->rules->test/use属性，这是标准webpack的写法。

安装：

$ cnpm install --save-dev babel-loader

$ cnpm install --save-dev babel-core

$ cnpm install --save-dev babel-preset-es2015

此时运行

$ webpack

命令，将自动的使用CMD构建app文件夹中main.js文件，并且使用babel翻译。

下面继续配置react，非常简单！

$ cnpm install --save-dev react

$ cnpm install --save-dev react-dom

$ cnpm install --save-dev babel-preset-react

你可能会问

11 react为什么不是bower来安装，而是npm来安装。

22 为什么是save-dev而不是save，也就是说为什么是开发依赖，而不是运行依赖。

稍后配置完毕解答。

改变webpack.config.js文件，增加一个babel翻译的配置项presets

var path = require('path');

module.exports = {

entry: "./app/main.js",

output: {

path: path.resolve(\_\_dirname, "dist"),

filename: 'all.js'

},

module: {

rules: [

{

test: /\.js$/,

exclude: /(node\_modules|bower\_components)/,

use: {

loader: 'babel-loader',

options: {

resets: ['es2015','react']

}

}

}

]

},

watch: true

}

至此我们已经配置完毕react起步的所有的依赖，我们目前的package.json文件:

{

"name": "react\_day1",

"version": "1.0.0",

"description": "",

"main": "index.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

"author": "",

"license": "ISC",

"devDependencies": {

"babel-core": "^6.26.0",

"babel-loader": "^7.1.2",

"babel-preset-es2015": "^6.24.1",

"babel-preset-react": "^6.24.1",

"react": "^16.2.0",

"react-dom": "^16.2.0",

"webpack": "^3.11.0"

}

}

2.2 Hello World

创建app/App.js文件，这是一个组件：

import React from "react";

//我们定义一个叫做App的组件，在HTML中可以用<App></App>来使用它

//React要求自定义组件必须以大写字母开头！

//React要求自定义组件的类必须继承于React.Component类

class App extends React.Component {

//组件中最重要的方法就是render方法，render是渲染的意思

render() {

//返回一个jsx语法，非常牛逼语法

return <h1>Hello World </h1>;

}

}

//向外暴露

export default App;

下面就要使用这个App组件，所以我们来到/app/main.js文件：

import React from "react";

import {render} from "react-dom"

//引入组件

import App from "./App.js"

//使用、挂在组件，两个参数

//第一个参数是jsx语法

//第二个参数表示组件挂在哪里

render(

<App></App>,

document.getElementById("app-container")

)

三、简单介绍jsx语法糖

jsx不能直接运行，是被babe-loader中react的这个preset翻译的：

|  |  |
| --- | --- |
| 翻译前： | 翻译后： |

需要注意：

11被一个单独的大元素包裹，比如div或者section

错误的

return <h1>Hello World</h1><h1>Hello World</h1>;

正确的：应该是一个div包裹起来,小括号也挺好用的：

return (

<div>

<h1>Hello World</h1>

<h1>Hello World</h1>

</div>

);

22标签必须封闭

错误的：

return (

<div>

<img src="" >

</div>

);

正确的：

return (

<div>

<img src="" />

</div>

);

33class要写成className，for要写成htmlFor

错误的：

return (

<div>

<div class="box" ></div>

</div>

);

正确的：

return (

<div>

<div className="box" ></div>

</div>

);

44HTML注释不能，只能使用JS注释。

55原生标签比如p、li、div如果要使用自定义属性，必须用data-前缀。

错误：

return (

<div>

<p a="10"></p>

</div>

);

正确：

return (

<div>

<p data-a="10"></p>

</div>

);

如果是自定义标签，此时可以随便传属性：

<App a=”12”></App>

66JS表达式用{}单大括号包裹，在JSX中不能使用if else语句，但可以使用conditional（三元运算）表达式来替代。以下实例中如果变量i等于1，浏览器将输出true，如果修改i的值，则会输出false。

<h1>Hello World{1+2}</h1>

{3 > 8 ? “A” : “B”}

77可以运行函数：

import React from "react";

//我们定义一个叫做App的组件，在HTML中可以用<App></App>来使用它

//React要求自定义组件必须以大写字母开头！

//React要求自定义组件的类必须继承于React.Component类

class App extends React.Component {

test(param){

if(param == "football") {

return (

<ul>

<li>big</li>

<li>middle</li>

<li>small</li>

</ul>

);

} else if (param == "house") {

return (

<ul>

<li>wooden</li>

<li>steel</li>

<li>glass</li>

</ul>

);

}

}

//组件中最重要的方法就是render方法，render是渲染的意思

render() {

//返回一个jsx语法，非常牛逼语法

return (

<div>

{this.test("house")}

</div>

);

}

}

//向外暴露

export default App;

88样式使用双大括号

<h3>JSX可以设置样式哦！</h3>

<div style={{“width”:”100px”,”height”:”100px”,”backgroundColor”:”red”}}></div>

99可以使用数组，数组中如果是JSX语法，数组会被自动展开，所以不要使用ng-repeat这样的指令展开数组。

import React from "react";

//我们定义一个叫做App的组件，在HTML中可以用<App></App>来使用它

//React要求自定义组件必须以大写字母开头！

//React要求自定义组件的类必须继承于React.Component类

class App extends React.Component {

//组件中最重要的方法就是render方法，render是渲染的意思

render() {

//定义一个数组，定义的JSX项目上要求有key属性，只要是重复的数组项目，都要key属性，都要有不能重复的key，这是要求。

//返回一个jsx语法，非常牛逼语法

let arr = ["wooden","steel","glass"].map((item,index)=>{

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

});

return (

<div>

{arr}

</div>

);

}

}

//向外暴露

export default App;

生成

<div>

<li>wooden</li>

<li>steel</li>

<li>glass</li>

</div>

四、React的数据传递

React中跟数据有关系的就三个东西：state、props、context，我们可以叫做数据传递三兄弟。

4.1组件自己身上的属性值变化，不会引起视图改变

我们可以再组件的类蒂尼里面写constructor构造函数，里面定义this.a=100,表示给组件的实例绑定一个a属性，值是100，在jsx使用的时候，直接this.a即可

11绑定监听使用onClick、onMousedown、onMouseenter、onBlur，把on后面的字母大写，React会自动识别React事件.

22绑定监听函数的时候，this上下文是有问题的，所以需要用bind()方法设置上下文。

33绑定监听函数的时候，注意用{}而不是””。

<input type=”button” value=”按我” onClick={this.add.bind(this)} />

44React中，组件自己的属性的变化不会引发视图的变化！！

4.2闭包中的值变化，不会引起视图改变

import React from "react";

//我们定义一个叫做App的组件，在HTML中可以用<App></App>来使用它

//React要求自定义组件必须以大写字母开头！

//React要求自定义组件的类必须继承于React.Component类

class App extends React.Component {

//组件中最重要的方法就是render方法，render是渲染的意思

render() {

//定义一个数组，定义的JSX项目上要求有key属性，只要是重复的数组项目，都要key属性，这是要求。

//返回一个jsx语法，非常牛逼语法

let b = 99;

return (

<div>

<p>{b}</p>

<p>

<input type="button" value="按我" onClick={()={b++;console.log(b);}}

</div>

);

}

}

//向外暴露

export default App;

闭包中的值变化，也不会引起视图的改变！

4.3三兄弟之一state

**Flux Standard Action**

11 介绍

一个对Flux动作对象用户友好的标准。

22 属性

一个action必须1.一个plain js对象（含有零个或多个key/value对）2.有一个type属性

一个action可能有1.一个error属性2.一个payload属性3.一个meta属性

一个action不能有除了type，error、payload、meta以外的其他属性。

type:是一个动作标识，用string常量表示，如果两个types相同则他们必须严格相同。

payload(optional):可为任意类型，任何不是表示type或者status的信息都属于payload字段。

error(optional):设置为true表示该动作代表失败，此时payload代表一个error对象。如果他的值为true以外的值如null、undefined，则不代表失败。

meta(optional):不属于payload的额外信息。

33通用函数

