npm install webpack style-loader css-loader url-loader babel-loader babel-core babel less-loader less jquery

npm install webpack-dev-server -g

React需要：npm install babel-preset-es2015 babel-preset-react react react-dom

通过localhost:8080只能访问到根文件夹下的html文件，想要访问别的就localhost:8080/xx/xx.html

MVC(一种模式)

Model（模型）:

是应用程序中用于处理应用程序数据逻辑的部分。  
通常模型对象负责在数据库中存取数据。

View（视图）:

是应用程序中处理数据显示的部分。

通常视图是依据模型数据创建的。

Controller（控制器）:

是应用程序中处理用户交互的部分。  
通常控制器负责从视图读取数据，控制用户输入，并向模型发送数据。

MVC数据量越大越难以维护，无法满足facebook扩展需求

React

Web开发工具，高效率，必须写组件，设计思想就是必须写组件，有虚拟DOM，减少重排和重绘

特点：声明式设计：采用声明范式，可以轻松描述应用

高效：通过模拟DOM，尽量减少DOM操作

灵活：可以与已知的库和框架很好地配合

(效率高)

传统web页面操作DOM一般是直接操作，但是代价很大，设计重绘重排非常耗性能，而React 为了尽可能减少对DOM的操作，提供了一种不同的而又强大的方式来更新DOM，代替直接操作DOM。就是Virtual dom, 一个轻量级虚拟的dom，又由于diff算法，更新virtual dom时不保证马上影响真实dom，react会等到事件循环结束，然后利用diff算法，通过当前新的dom表述与之前做比较，计算出最小的步骤更新真实的dom。

(天生组件化)

Components组件：

在dom树上的节点被称为元素，在这里不同称为virtual dom，整体称为component， 众多的virtual dom的节点就是一个完整的抽象的组件。

(清晰)

单项数据流:

从父节点传递到子节点，因为组件是简单而且易于把握的，他们只需从父节点获取props渲染即可。

State 和 Render:

state属性包含定义组件所需要的一些数据，当数据发生变化时，将会调用Render重现渲染，

React 把组件看成是一个状态机（State Machines）。通过与用户的交互，实现不同状态，然后渲染 UI，让用户界面和数据保持一致。

应用场景：

复杂场景下的高性能

重用组件库，组件组合（ant.design-支付宝组件库）

懒人必备

jsx语法，在配置文件{

test: /\.js$/,

loader: 'babel',

exclude: /node\_modules/,

query: {

presets: ['react', 'es2015']

}

}定义了这种写法

var React = require('react');

var ReactDom = require('react-dom');

RectDom.render（

<h1>cst</h1>,

document.getElementById('root')

）

把h1放到id为root的div里

var App = React.createClass({

render: function () {

return <div>hgf</div>;

}

});

App里就是一个组件，看render返回了什么，第一个字母必须大写，把div放进去了，返回值只能有一个根节点

return(<div><span></span></div>)；是可以的

Jsx：<div>hello{this.props.name}</div>

语法糖：计算机语言中添加的语法，对语言的功能没影响， 方便使用，增强可读性。

React官方推荐使用jsx当然你想直接使用js代码，也是可以得只是使用jsx，组件得结构，和组件之间得关系看上去更清晰，使用页更方便

div加class要写className=“xxx”,style要加大括号,style={}或

Var styles = {background: ‘red’,opacity:’0.5’}style={ styles }

大括号里不能有分号

<ul>

{

this.props.arr[0].map(function (item, key) {

return <li key={key + 1000}>{item}</li>;

})

}

{

this.props.arr[1].map(function (item, key) {

return <li key={key + 1200}>{item}</li>;

})

}

</ul>

每个大括号里只能有一个函数

return (<div className="app-base" style={styles}>

//this是组件本身

<span>{this.props.name}</span>

</div>);

}

this指组件本身<App name='cst'></App>里的属性可以用this.props.name访问到

数据单向流动，传到父级里，向子集流动，用this.props.xxx

Mounting(组件生成时触发)

getInitialState:必须返回一个对象，state就是对象的属性，this.

state.xx可以访问到return{}里的属性，不可修改，要用this.setState({})修改

componentWillMount:

render:

componentDidMount:

Updataing(组件更新时触发， state 真实变化时影响dom结构才会，真正的改变dom结构， state，props变化都会触发组件更新)

componentWillUpdate:

render:

componentDidUpdate:

当在父级组件中添加子级组件时想要在子级组件里在添加一些东西，需要在定义子级组件时就加{this.props.children}

<div style={styles}>

{this.props.children}

</div>

注册事件用onCLick这种小驼峰式写法就好

如何用组件里的函数操作DOM

获取DOM：

<input type='text' ref='Inp' onChange={this.onHandleChange}></input>

this.refs.Inp，此时获取到ref为Inp的input标签

ref={function (dom) {this.\_Inp = dom}.bind(this)}

this.\_Inp，相当于把DOM挂载在组件上

<tbody>

{

[<ProductCategory/>,<ProductRow/>]

}

</tbody>也可以用这种方式来添加组件，放在数组里以便于管理