------------------------------------------0712-----------------------------------------------

1、知识点整合：（关于变量问题）

（1）全局变量：不在函数内部定义的变量，任何地方都能使用的地方；

局部变量:函数内部定义的变量；只能函数内部使用。

1. 变量污染：变量覆盖；解决变量污染的方法:加前缀
2. 匿名函数：

（function（job）{

Console.log(111);

Var b = 3；

Window.api = b;

}）（）；

**一切全局变量都是window的属性**

对象封装功能:(function(){

var obj = {

Del:function(){

},

Update:function(){

},

Add:function(){

}

};

Window.admin = obj;//将局部变量变成全局变量。

将变量进行对外开放；通过window进行开放；

})();

**封装：1).便于管理和维护；**

**2).提高代码的重用率；**

**3).提高开发效率；**

**4).简化代码；**

(4)参数注入：参数滥用冲突；依赖注入；

(function($,nick){}

)(juquery,nick)

1. **commonjs规范:一个文件作为模块来用；ECMA script5/ECMA script6/ECMA script7**

2、插件知识点：（插件）

**RequireJs操作方法：**

**模块化1：html文件中**

**（1）.先引入接口文件**

**<scriptsrc='js/libs/require.js'type="text/javascript"charset="utf-8"></script>**

**<script>**

**（2）配置接口；**

**require.config({**

**（3）配置文件根目录；**

**baseUrl:'js',**

**（4）配置每个模块的名称与路径；**

**paths:{**

**jquery:'libs/jquery-1.7.2.min',//加载jquery文件**

**work1:'work/one'**

**}**

**});**

**（5）依赖加载；**

**require(['jquery（可以放在定义模块中）','work1'],function($（若jquery在定义模块中，则不需要传参$）,work1){**

**console.log($);**

**console.log(work1);**

**});**

**</script>**

**模块化2：定义模块define（‘模块名称’，‘依赖和数组（可选）’，回调函数）**

**define('work1',function(){**

**var a = 56;**

**var b = 76;**

**return a+b;**

**});**

**Arguments，获取函数中的参数，是一个集合；**

**Sea.js的操作:**

**AMD:是requirejs在推广过程中的规范化；不过requirejs也支持使用CMD；requirejs需要依赖。**

**CMD：是seajs推广过程中的规范化；**

**无论CMD还是AMD都是模块化开发规范；**

**AMD推荐依赖先加载，CMD什么时候用什么时候加载（require）；**

**------------------------0713**

**（02）-------------------------**

**后台语言：nodejs**

1.安装node

2. 运行cmd 打开小黑窗口

3. 查看node版本 node -v

4. 查看npm版本 npm -v

5. 安装cnpm包管理 npm install -g cnpm --registry=https://registry.npm.taobao.org

6. windows下 按shift加右击 选择打开命令窗口 可以直接在当前目录下运行DOS窗口 cd选择目录 cd ../上一层 cd 目录 dir查看当前目录下的文件结构

7. 运行指定js文件 node xx.js 注意不能直接在目录中打CMD 就写完整路径 比如复制 node F:\xmap\htdocs\0411\week1\02\xx.js

8. **注意 只能开启一个服务器不能同时监听多个！！！**

**Cmd的简单命令：（1）不同盘的切换--默认c盘，切换到D盘--d:**

**查盘内文件的目录信息：dir；**

**进入当前盘内的文件夹：cd 文件夹名**

**Cd ..退出当前目录**

**Windows里不区分大小写 Linux区分大小写；**

Nodejs基础：

编写服务器：（1）.加载http模块

var http = require('http');

（2）.配置服务器端口与服务器地址

var host = '127.0.0.1';

var port = 80;

var server =

http.createServer(function(request,response){

(3) 回调函数第一个参数是 request请求头对象 第二个参数是response响应头对象

(4) 通过响应头设置 http状态码

response.statusCode = 200;

(5)设置请求头；

response.setHeader('Content-Type','text/html;charset=utf-8;');

（6）.输出内容 echo

response.end('这是我们输出的内容')

})

（7） 监听服务器

server.listen(80,'localhost',function(){

console.log('服务器已运行');

})；

加载event事件：（1）创建event事件；

1. 实例化event事件；

new events.EventEmitter()

1. 通过on方法添加事件 on(事件类型,回调函数) 事件类型随便定
2. .emit(事件类型,...给回调函数的参数)

加载util实现继承：

util = require('util');

（5）.通过util工具包实现继承 第一个参数 继承第二个参数的所有属性和方法；

util.inherits(MyEvents,events.EventEmitter);

加载buffer，创建buffer：

读取文件，加载fs模块；

创建文本流：createReadStream('readme.md1')；

-----------------0714（03）-------------------------

**安装cnpm：**

**1. npm是包管理工作 每个功能模块都可以称为一个包 安装 卸载 配置模块**

**2. npm init 命令 初始化一个package.json文件 这个文件保存了包的详细信息、版本、作者、说明、 以及安装的包具体信息**

**3. cnpm install --save-dev express 安装express**

**4. 尽量不要复制迁移node\_modules文件夹 只要package.json文件就可以 有了这个文件 之后 执行cnpm install 会自动安装package.json里面所写的需要用的模块**

**5. express中的问题**

**1. 任何请求都必须使用send向客户端发送数据，如果不发送客户端永远等待**

**2. 同一个URL请求只能发送一次send**

**3. 同一个URL的多个请求时，前面的请求要执行next()才会执行下一个，最后一个可省略**

**4. 一旦向客户端发送数据就不能再设置请求头！！！**

**5. all方法无论何种请求类型都会执行 ，不限请求方式但是匹配URL路径。 \*代表任意路径**

**cnpm install --save express 安装express**

跨域请求设置**：**

response.header("Access-Control-Allow-Origin", "\*");

response.header("Access-Control-Allow-Headers", "X-Requested-With")；

response.header("Access-Control-Allow-Methods", "PUT,POST,GET,DELETE,OPTIONS");

response.header("X-Powered-By", ' 3.2.1');

**加载express插件：**

1. 加载express模块

var express = require('express');

2. 通过express实例化一个http服务或应用

var server = express();

3. 添加一个服务

server.get('/',function(request,response){

console.log(666);

response.send('hello word!');

});

server.get('/news',function(request,response){

response.send('news');

})；

4.添加监听

server.listen(80,'127.0.0.1',function(){

console.log(666）；

})

Node.test 运行开启服务器；

req里面的href path \_raw 里面的属性都是一样的是当前的URL地址；get参数直接通过req.query可以得到；

------------操作数据库0717（04）------------------------

1. 查看库 show databases

2. 使用数据库 use databasename

3. 查看表 show tables

4. 查看表结构 desc tablename

5. 创建数据表 create table tablename(字段名 字段类型 是否为空,字段名 字段类型 )engine = innodb charset=utf8;

**create table news1 (id int unsigned primary key auto\_increment,t**

**itle varchar(30) not null default "",time int unsigned not null default 0,pv int**

**unsigned not null default 0,del tinyint not null default 0)engine=innodb charse**

**t=utf8;**

**安装mysql模块 cnpm install --save mysql**

**1. 加载mysql模块**

**2.创建数据库连接**

**3. 发送语句**

**4. 创建新闻表**

**5. 添加数据**

**6. 查询数据**

**7. 更新数据**

**8.删除数据**

**9.关闭连接（04mysql）**

第一个参数是error信息 如果没报错就是null 第二参数是数据库返回的信息

配置根据哪个get参数 获取回调函数名称 默认值为callback；

MVC

M：model数据模型；V:view视图要显示的内容；C：controller控制器；

控制>模型获取数据

控制器将获取的数据->交给view 让它来显示

控制—>模型 拿了数据找view视图；

视图->model模型；

异步执行如ajax，查询数据库（query），无法return；

Trim/trimLeft/trimRight:ES5;

ECMA 6

Let 用于声明变量 作用域是{}级别 仅在{}内生效；

Let 完全代替闭包，作用完全代替var的功能；

Let没用变量提升问题；

使用let需要在头部加载一个严格模式：‘use strict’；

使用``反双号字符串可以任意换行，并且通过$(变量名)，在字符串中直接使用变量；

Const d = 1；定义常量；一旦赋值，则无法更改；

{解构赋值：}

声明多个变量；

'use strict';

let [a,b]=[1,2];

console.log(a);//输出1；

console.log(b);//输出2；

解构赋值：

let[f,[[con],mon]]=[1,[[4],5]];//解构赋值；

console.log(f);//1

console.log(con);//4

console.log(mon);//5

console.log(mon+con+f);//10

默认值：

var [g=0,h=666] = [8];

console.log(g);//8

console.log(h);//666

对象解析；var {属性名,属性名1...} = {} 把对象的值 赋值给前面的属性名 变成一个变量.

var obj = {tit:'mysql',id:4,type:5};

var {tit,id,type} = obj;

console.log(tit);//mysql;

console.log(id);//4;

console.log(type);//5;

内部解构json数据:

function json(data){

var {result,error,success}=data;

if(error){

}

for(var i = 0 ; i < result.length;i++){

console.log(result[i]);

}

}

json({result:[1,2,3,4],error:'',success:true});

dom集合 jq对象 等 长的像数组却没有数组的方法 叫伪数组或类数组

//将集合（伪数组）转换成数组

var all = document.querySelectorAll('\*');//伪数组；

1. var arr = Array.from(all);
2. [...all];
3. function from(data){

var arr = [];

if(data.length){

for(var i = 0 ; i < data.length;i++){

arr.push(data[i]);

}

}

else{

for(var j in data){

arr.push(data[i]);

}

}

return arr;

}

Console.log(from(all));

数组的遍历：

1. **find 有返回值；能return；**

第一参数当前数组元素的值 第二参数当前数组索引 第三个是被操作的数组

var arr = [11,22,33,44,55,66];

var result = arr.find(function(value,index,arr){

注意 该函数会遍历指定的数组，在函数内只要返回true 那么 当前被遍历的这个数组元素值就会返回 出去；

return value>=33;

});

console.log(result);

1. **findIndex查找并返回索引值；**

Var arr=[11,22,33,44,55]

var index

=arr.findIndex(function(value,index,arr){

//console.log([...arguments]);

//console.log(values);

Return value>=33;

});

Console.log(index);

**3.forEach;遍历数组；**

var arr = [12,23,34,45];

var num = 0;

arr.forEach(function(value,index,arr){

// console.log([...arguments]);

num += value;

console.log(value);

}); console.log(num);

补：jquery中数组遍历：

var arr = [123,234,567,456];

$.each(arr,function(index,value) {

console.log([...arguments]);

console.log(index);//下标值

console.log(value);//数组中的值

});

**对象：**

对象中写变量，变量名将作为对象属性名，变量值将作为对象的属性值。

var id = 1;

var tit = '没有呀';

var obj = {id,tit};

console.log(obj);//Object { id: 1, tit: "没有呀" }

2在对象中函数可以不写function关键词;

var obj2 = {say(){},hello(){},obj1};

console.log(obj2);//Object { say: say(), name: name(), obj: Object }

**类：**

**-----------0719-------------------------------------**

**框架：**优点----开发快，页面构建快

2--开发规范

3--便于团队协作与维护；

缺点--很难更改框架内部代码

体积庞大（代码冗余）

不易调试（内部报错很难排查）

不利于SEO，毁了SEO 搜索引擎；

适用场景：

小型页面；

不需要SEO的（后台、OA）

开发快，性能低；性能高，开发慢；

**Angularjs：**

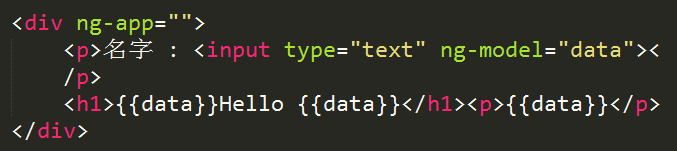
Angular通过给标签添加自定义属性（指令）；

{{}}用于书写angular的表达式 可以写变量、数值、字符串 或+-\*/等

ng-app加在谁身上，作用域就在谁身上。 一般把ng-app加在body或html身上 ，定义angularJS的使用范围；。

Ng-init 用于指令初始化；**ng-init =”变量名=值”;多个变量以;号隔开；**变量{{}}可以在他所在的外围，也能执行，可以执行多个使用；

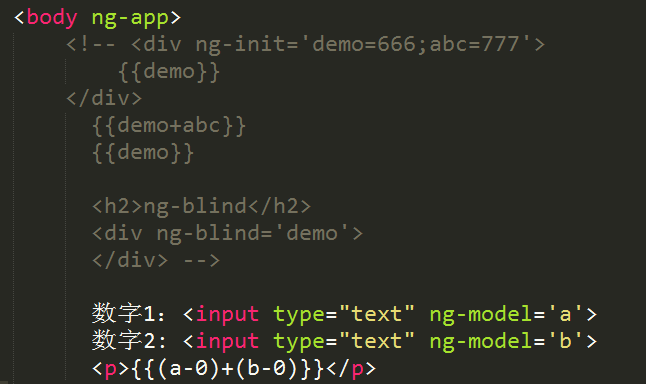
通过 ng-bind与{{}}都能绑定数据到元素，但是ng-bind绑定的表达式页面不显示



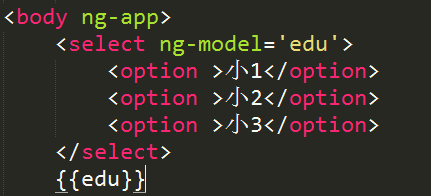
Ng-init 用于初始化创建变量

ng-model也是用于初始化模型（数据、变量） 但是它只对**表单有效**

ng-model 创建一个变量， 将表单的值绑定到这个变量上！！！ {{变量名}} 在其它 地方又能将这个变量的数据绑定到别的元素上



在select中也能生效；



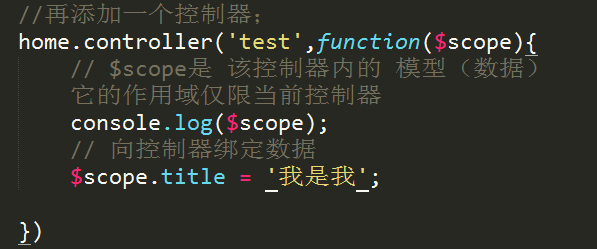
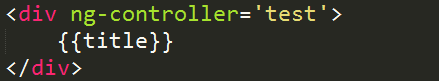
**ng-controller的应用；**

如果ng-app有值 一定要获取这个应用 注意实例化应用一定要写[]注入依赖 没有也得写空数组



//页面输出：我敌人；

再添加一个控制器：

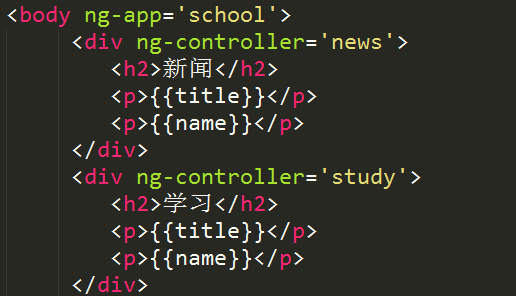


**变量的作用域；**

$scope是局部的 $rootScope是全局的

$scope作用于自己的控制器 ，而$rootScope作用于所有控制器

$scope是数据模型 （挂载） 所有要用到的数据（模型）都放$scope身上

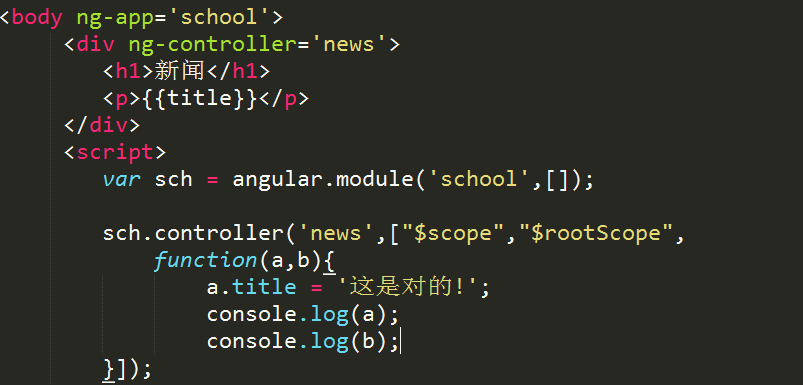


注入的参数如果不是angular定义的那么就报错；

注意angular是依赖注入的，直白的说需要传递参数，通过这些参数来扩展更多的功能

比如$scope 负责局部数据模型

不能注入不存在的模块或功能，否则 报错。（传递的参数如果没有就报错！！！）。

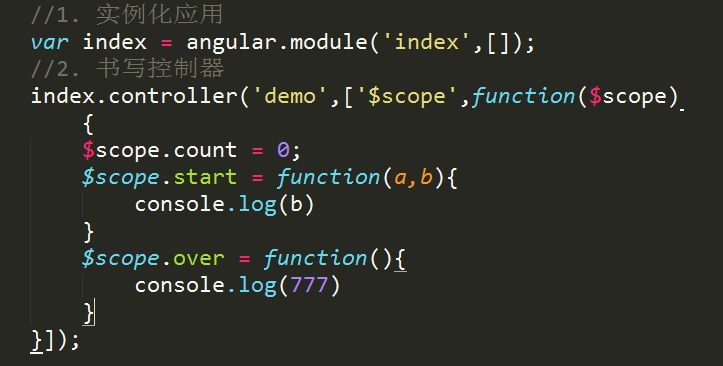


**事件：**

1.添加事件，html代码



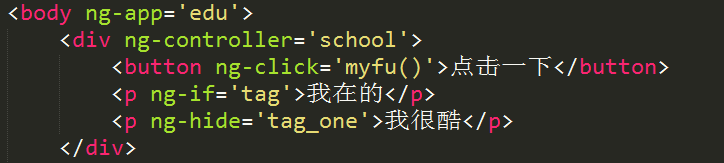
2.js代码



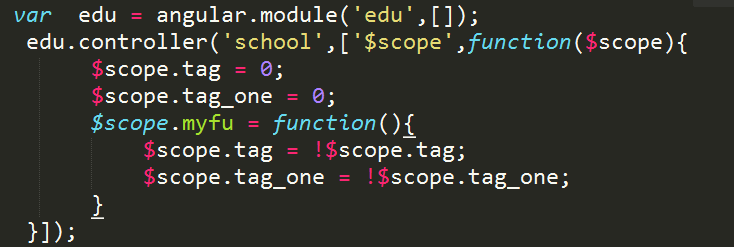
ng-if与ng-hide的区别 如果ng-if为false 那么元素不在DOM中 不渲染

ng-hide若为true 元素隐藏不可见，但依然在DOM中。

（1）.html



1. .js



ng-repeat="data in xxx" ng-repeat写在谁身上 谁被重复

变量 in 数据 这个变量就是遍历数据时当前的数据。

-------------------------0720--------------------------

http:协议发送数据；

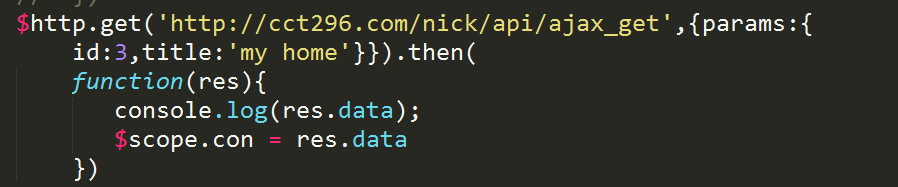
Get请求数据：

$http.get(url).then(function(返回值){})；

第一个参数是对象，data里面是服务器返回的数据；



Get请求传送数据：

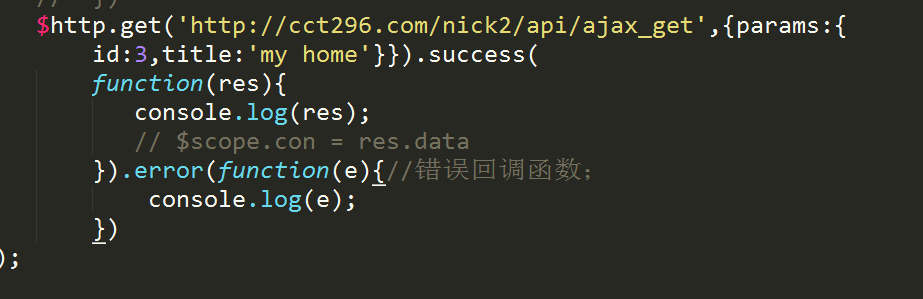
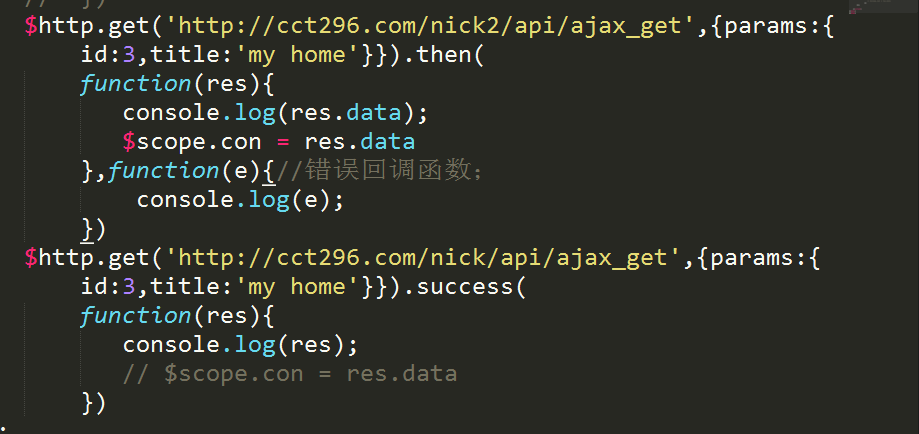


**Then和success的区别：**

then里面第二个回调函数是请求失败要执行的回调函数

注意 then与success的区别，then里面的回调函数的参数是http的对象里面有很多请求的数据 比如header url ....data

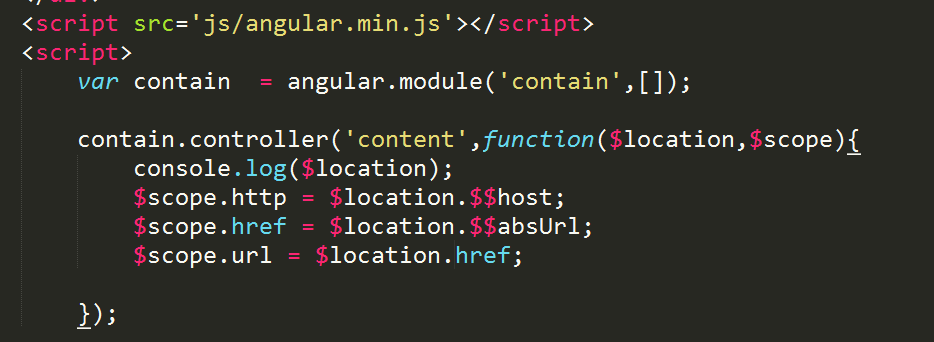
而success只返回服务器的数据 ,success没有失败要执行的回调函数



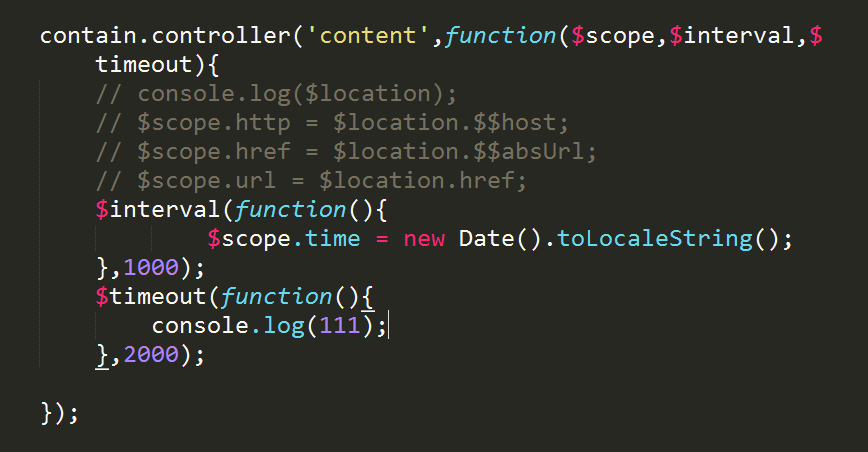
Success后也可以接一写错误回调函数；

Promise解决数据传送；

Loacation的用法：



**angular定时器和延时器的用法：**

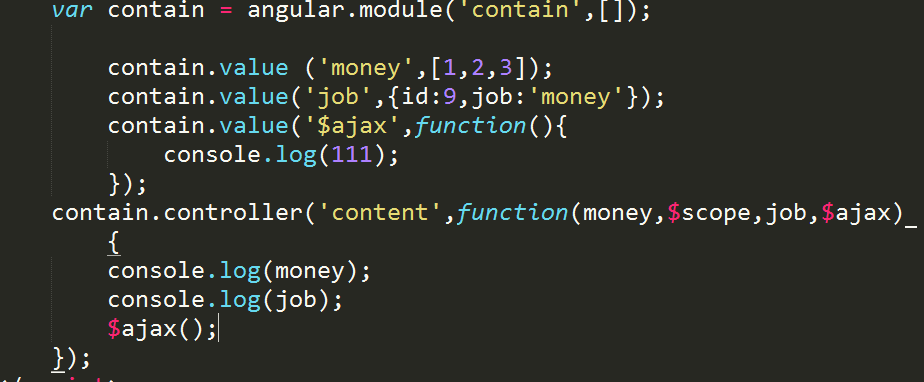


**Value/constant/factory/provider的使用；**

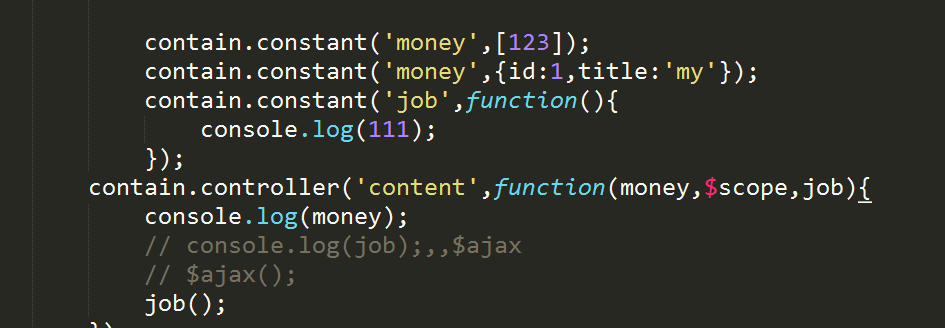
**Value**

**通过应用.value(属性名,值) 注入内容**

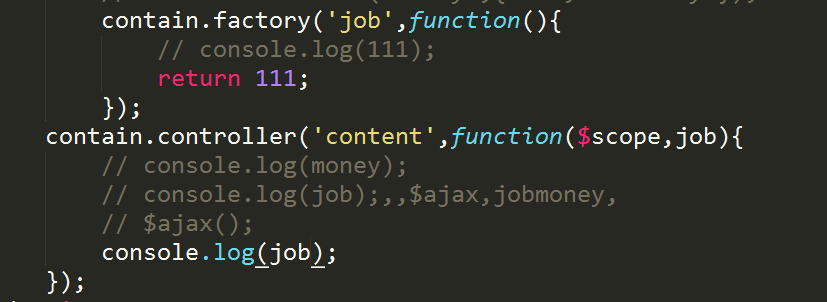
**通过value注入的内容可以在控制器里面使用 属性名是什么 回调函数里面的参数名就是什么**



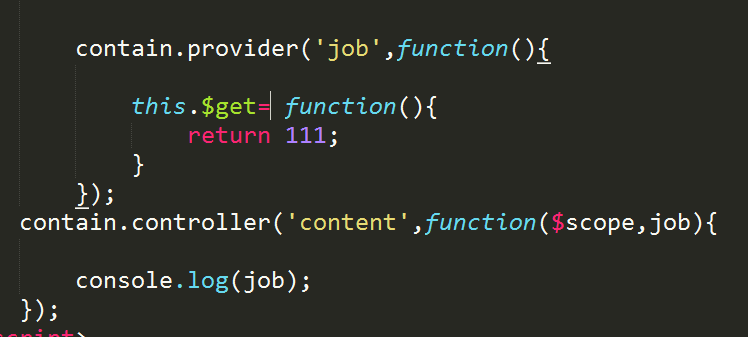
**Constant：与value类似；**



**Factory：必须是函数，并且又返回值，不然一律报错；必须通过回调函数 返回要注入的内容！！！！ 必须return 。不能return undefined； 也会覆盖！**



**Provider：必须通过this.$get函数，return返回值**



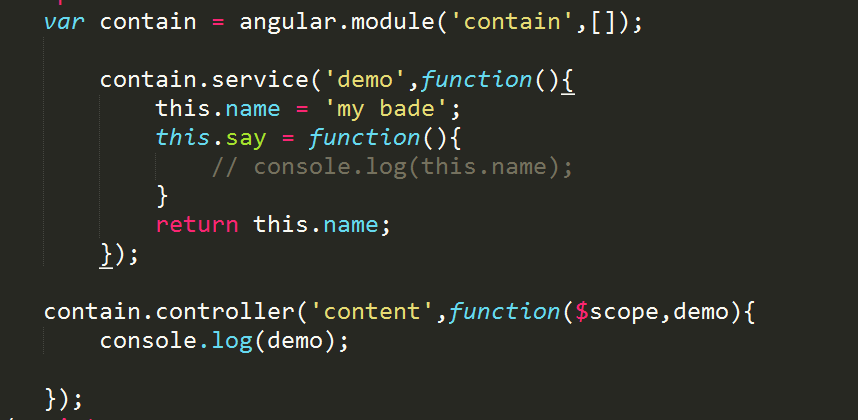
**1. value可以覆盖**

**2. constant 不能覆盖**

**3. factory 通过函数的返回值作为注入的数据 能覆盖**

**4. provider 必须this.$get=function(){} 函数return什么就注入什么**

**Service:**



注意 如果return的是对象 注入的就是对象

如果不return 对象 注入的是这个函数的实例。把这个回调函数new之后 注入

\* 1.定义不需要再更改的用constant

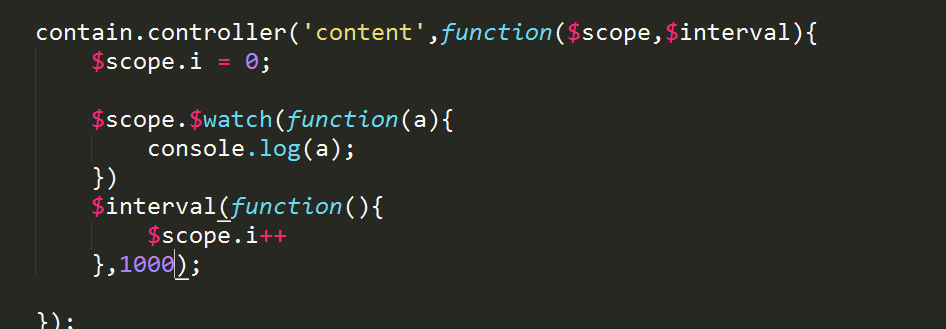
\* 2. 直接定义数据的可以使用constant 或value

\* 3. 返回的数据如果需要写代码处理或依赖其它功能处理就得用factory provider service

\* 可以在回调函数中先处理 再return

**Watch:监听；**

**任何数据的改变都会执行$watch 通过这个函数可以监听 数据的变化**



**$applay():**

**$scope.$aaply()方法用于强制刷新页面（渲染页面）**

**$interval的定时器返回值是对象 里面有$$intervalId 通过 clearInterval($$intervalId)可终止;**



**Directive指令基础：**

template的内容就是调用指令时生成的内容

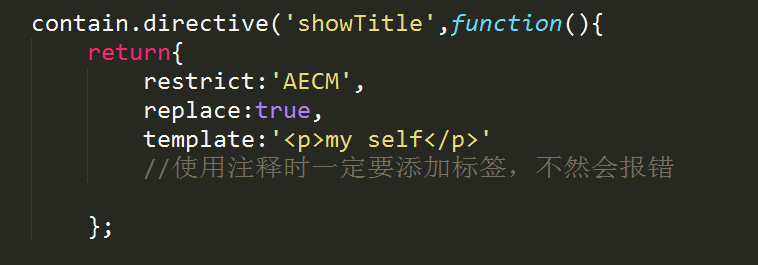
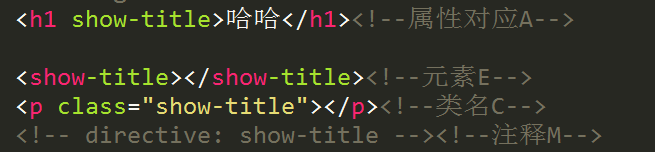
restrict 限制指令的使用形式 A 属性 E 元素（标签名） C 类名class M 注释

默认使用限制是AE 标签名与属性

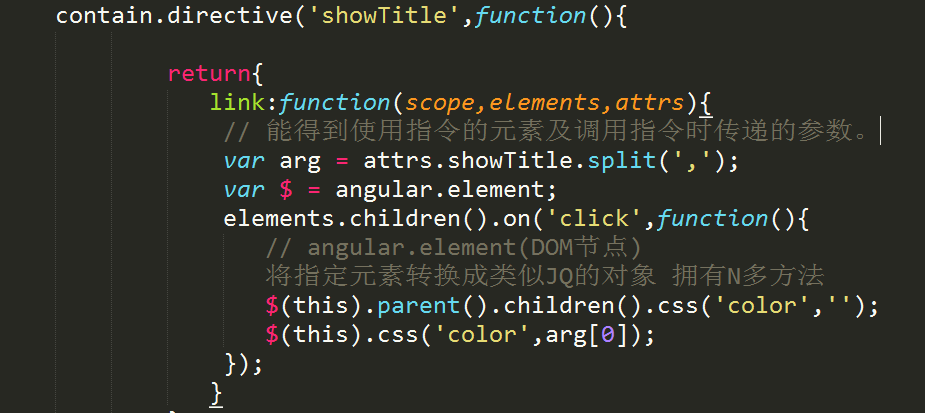
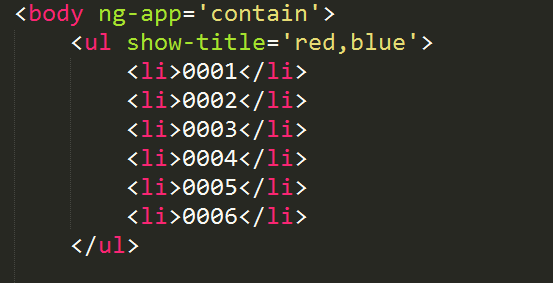
replace默认值为false 如果为真 使用模板中的标签替换调用指令的元素（标签、注释）

使用M 注释形式时 必须replace为true 不替换注释看不见！

推荐使用AE C类 怕有样式类名相同 导致添加上样式



**Directive指令的另一种用法：**



**点击事件的用法；**

**广播：**

$broadcast的作用：将事件从父级作用域传播至子级作用域,包括自己。格式如下：$broadcast(eventName,args)；

$emit的作用：将事件从子级作用域传播至父级作用域,包括自己，直至根作用域。格式如下：$emit(eventName,args)；

$on的作用：用于在作用域中监控从子级或父级作用域中传播的事件以及相应的数据。格式如下：$on(event,data)；

上述说明中，**eventName是需要监控的事件的名称，$on 方法中的参数event是事件的相关对象，data是事件传播的数据。**

**0724------------------**

**Sass less**

**Gulp文件**

<http://www.ydcss.com/archives/34】>

1. 安装全局gulp cnpm install -g gulp

2. 安装项目依赖 cnpm install --save-dev gulp

3. 创建gulpfile.js文件 里面写具体的代码

4. 安装gulp-less cnpm install --save-dev gulp-less

5. 执行gulp 命令默认启用default任务 gulp 任务名称

**SVN**

1. svn客户端与服务器端 服务器负责文件的存储，客户端负责连接服务器端进行上传下载。。。。

2. 一般公司只有一个SVN服务器端

3. repositores 是仓库资源库 右击可以创建仓库

4. users 用户 右击 添加用户（一定要添加用户）

5. groups 用户组 将若干用户分为某个组 右击 添加组 向组添加用户

6. https://nick-PC/svn/myproject 通过地址可以访问SVN服务器

7. 右击文件夹-> checkout检出 配置SVN仓库地址 -> 输入用户名和密码 账号与SVN仓库关联

8. 常用操作

8.1 右击文件选择add将文件添加到版本控制

8.2 右击commit提交 操作 建议写注释（操作说明）

8.3 右击update 更新 将仓库最新的数据下载到本地

9. 软件问题电脑问题解决思路

9.1 先百度搜索报错的内容 照网上人家说的试

9.2 试不成功？再找几个人家的方法试试

9.3 以管理员身份运行

9.4 关闭杀毒软件

9.5 别装到C盘

9.6 重启

9.7 再安装

9.8 卸载要清除残留 建议使用第三方管理进行删除 360

9.9 到用户目录找appdata看到这个软件相关命名的直接删除

React:

Props是组件的属性；

**Webpack安装：需要在node.js环境下进行安装；**

Cmd安装命令：**cnpm install webpack -g**

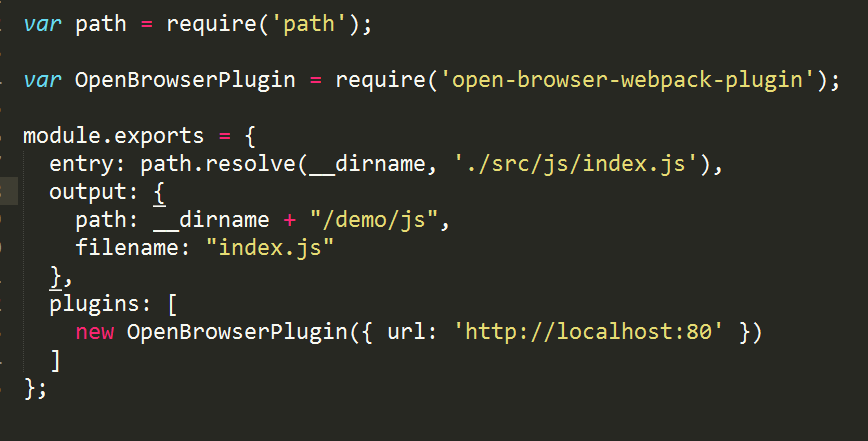
创建app文件夹：**mkdir app**

**打包绑定：**webpack 文件夹名/文件名下的js文件 需要存放的js文件夹名/js文件

列如： **webpack app/index.js app/demo.js**

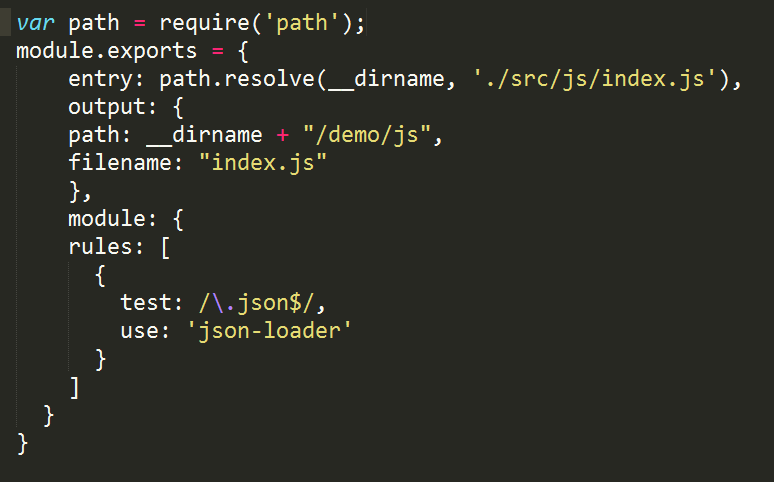
安装css和style：**cnpm install css-loader style-loader（不要装全局，一定是局部）**

cnpm install --save-dev bower-webpack-plugin 浏览器加载安装；



Webpack执行命令；

cnpm install --save-dev json-loader 【json-loader加载安装】



通过webpack --module-bind ‘json=json-loader’;

**热更新**

1. npm init 初始化包配置 里面写上版本 作者 说明 。。。

2. cnpm install --save-dev webpack 安装webpack

3. 创建webpack.config.js 进行配置

4. css依赖 cnpm install --save-dev style-loader css-loader

5. bable依赖 cnpm install --save-dev babel-loader babel-core babel-preset-es2015

6. less依赖 cnpm install --save-dev less-loader less

7. json依赖 cnpm install --save-dev json-loader

8. url依赖 cnpm install --save-dev file-loader url-loader

9. html模块依赖 cnpm install --save-dev html-webpack-plugin

10. webpack-dev-server依赖 cnpm install --save-dev webpack-dev-server 开启服务器

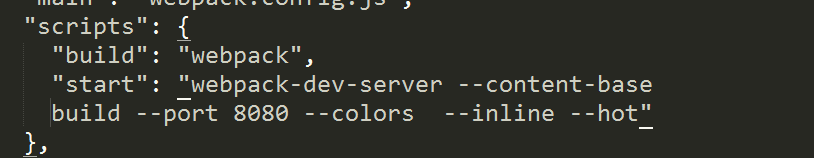
11. open-browser-webpack-plugin 依赖

cnpm install --save-dev open-browser-webpack-plugin

12. react依赖 cnpm install --save-dev react react-dom babel-preset-react

13. html文件修改后不会自动更新 css js less...等都可以 html修改要手动刷新

Packjson文件中的script中修改为：start：‘"webpack-dev-server --content-base build --port 8080 --colors --inline --hot"’；



编译：webpack或者 npm run build;

打开浏览器：npm run start;

加载 .babelrc 文件：{

"presets":["react","es2015"]

}

的内容；

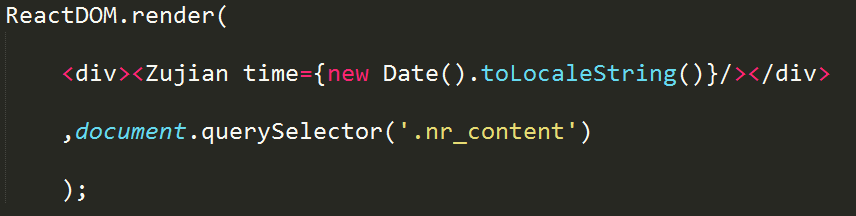
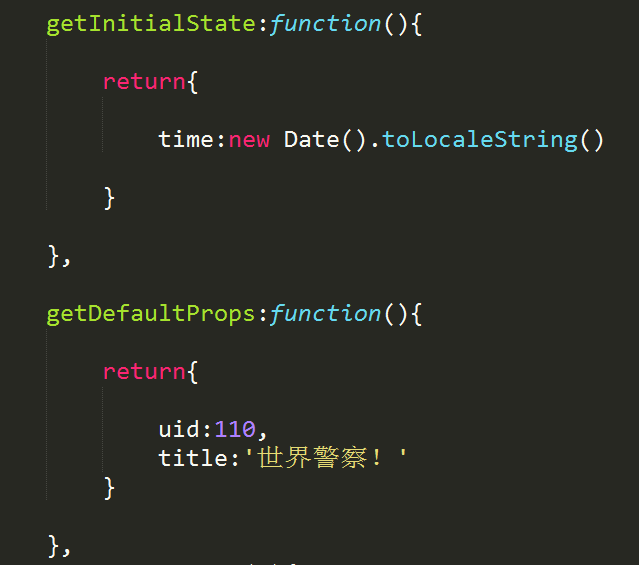
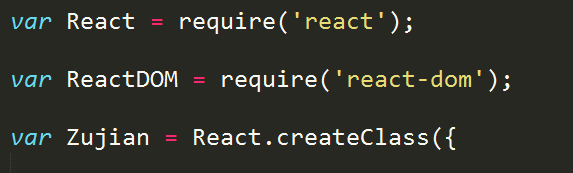
入口文件：es5则用require加载依赖；react 和 react-dom依赖

Es6用import加载依赖；

加载react依赖时，必须大写；

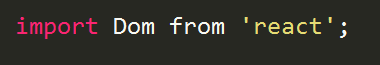
添加点击事件，props属性/states属性/refs绑定元素；

可以通过className添加类名：设置样式。



通过es6语法加载事件：import加载依赖；

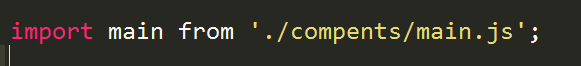
Import加载文件依赖：import 名称 from 地址；



如果这个模块里面有若干属性可以单独引入某个属性 import {模块的属性} from 模块；



加载react依赖时变量首字母必须大写；无论是模块属性还是正常的方法；还可以加载文件路径；



Es6：创建类的方法：创建类时，必须加载模块属性，不然就是报错；

import {模块的属性} from 模块；



**Webpack--Jquery:**

**安装jquery：cnpm install --save jquery**

**加载jquery依赖：**

**Import $ from ‘jquery’**

**Fetch方法：fetch属于内置函数；**

**(注释)--Js document:/\***

**\***

**\*/**

**/\***

**\* 为什么封装？**

**\* 1. 用的方便，提高开发效率**

**\* 2. 提高代码重用率，提高开发效率**

**\* 3. 提高代码可维护性，提高开发效率**

**\* 4. 提升性能**

**\* 何时需要封装？**

**\* 1. 使用频率高**

**\* 2. 业务逻辑复杂（功能）**

**\* 3. 很多代码段相同，部分不同！ 不同的地方一般通过传递参数（注入）来解决 \*/**

**路由规则：**

**安装react-router模块： cnpm install --save react-router；**

**路由规则：**

**1.<Router>组件用于定义路由规则**

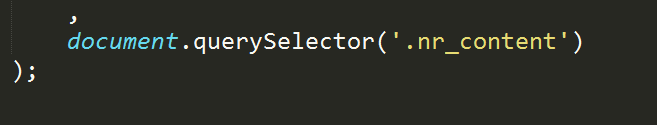
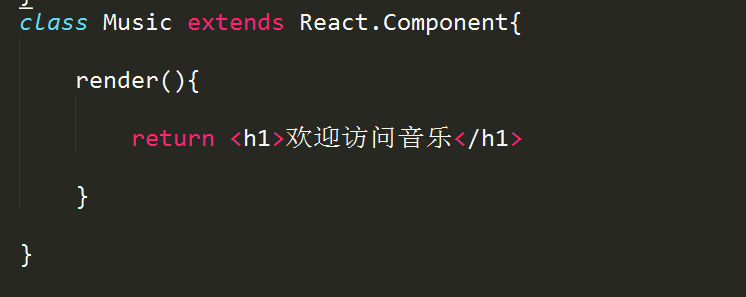
**2. Router上必须有history={hashHistory}属性 主要是用于创建历史记录**

**3. <Route></Route>组件则是Router下的规则 ，每个route都有path定义要匹配的URL地址，通过component={组件名} 显示指定的组件**

**4. 匹配到url之后 显示指定的组件 组件里面有什么就显示什么**

**5. Link可以创建a标签 href设置地址 有的版本可能 是用to设置 标签里面的文字就是链接里面的文字**

**6. 建议不要用link直接用a 何须多此一举！**

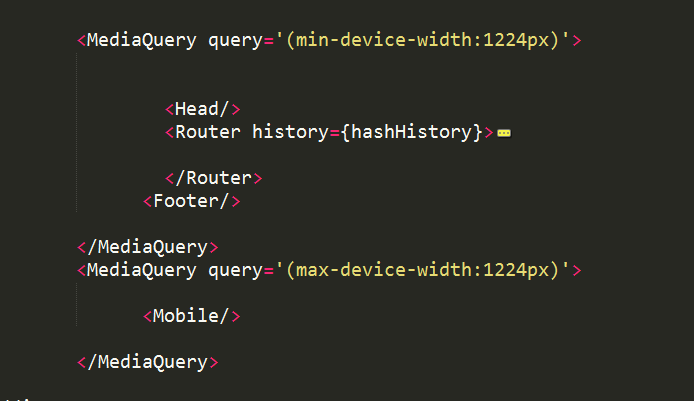
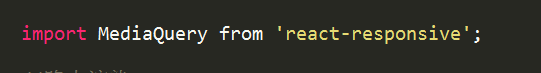


Cnpm install

Cnpm install --save antd 组件

React-responsive 响应式布局；

cnpm install react-responsive --save；安装组件依赖；



**Mockjs生成随机数；**

安装mockData组件：cnpm install --save mockjs

Redux:安装--cnpm install --save redux；

createStore用于创建一个存储；

State对象必须return出去；

**Vuejs教程：安装cnpm install**

*Cnpm install –g –save vue-cli ：脚手架 构建工具*

*Vue 查看vue信息*

vue init webpack my-project自动创建配置，大型框架；

(安装vue路由：cnpm install --save vue-router

)

**ESlint安装 时N属于严格模式；（不上线，不安装）；**

**Karma和Mocha不需要安装；N；**

**E2e 测试工具；**

**黑盒测试：就是手动测试；**

**白盒测试：代码测试；**

**Cnpm install 安装node\_modules;**

**Npm run dev 运行;**

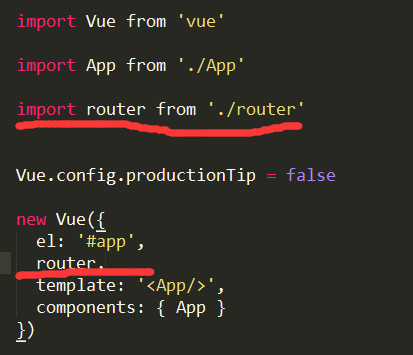
**cnpm install （vue-router ）vue-resource --save 安装网络请求资源;**

**Vue中：**

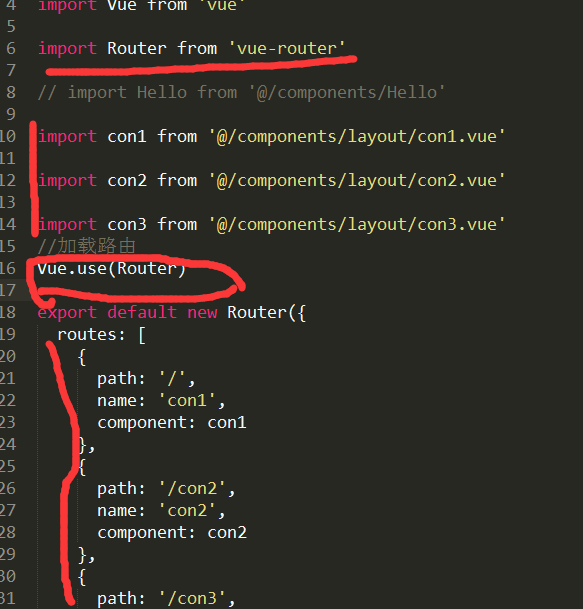
**methods方法调用时需要加（）调用；**

**Computed不需要：**

**加载路由时：主文件--main.js**

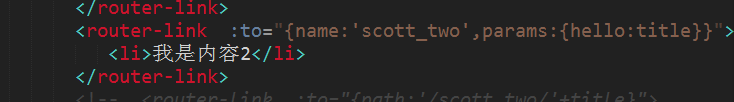


**路由文件：index.js**

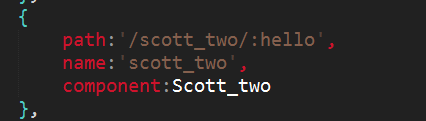


**路由传递数据（参数）（params接收参数）：**

**存在两种方式：先写一种官网推荐传递参数的方式；**



**在index.js文件中：**

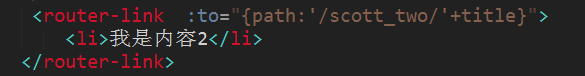


**在data（）{title：‘my，job’}函数保存参数值；**

**传递的参数在scott\_two里面调用：{{$route.params.hello}}渲染;**

**而router-link里面的name值必须与路由文件中的name保持一致，不然页面不会渲染，报一个警告，该页面中文件内容不会显示；**

**第二种是简便的写法，比较实用；**



（一般不推荐使用）

**数据传递（数据共享）：**

**npm install vuex --save 安装vuex单独安装；**

1.Vue Components -> (Dispatch) Actions -> (Commit) Mutations -> (Mutate) state -> Vue Components

2.Vue Components -> (Commit) Mutations -> (Mutate) state -> Vue Components

3.Vue Components -> (Dispatch) Actions -> (Mutate) state -> Vue Components

**在main.js中加载vuex加载依赖：**

**import Vuex from 'vuex'**

**import store from './store'（完成数据共享的文件夹）**

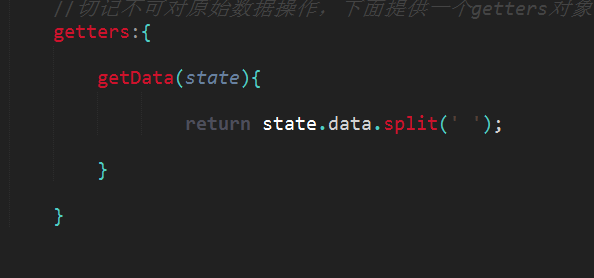
**在store中有一个index.js(文件名随意)**

**在store/index.js中的加载vuex的依赖；**



而对象中的data完成全局共享；vuex里面的state对象的数据达到共享所有的vue文件都能进行访问调用，为了避免后面的文件频繁对state对象中的原始数据的操作，所以采用**getters对象**访问state数据，避免了对原始数据的破坏；

而要操作data数据，需要通过getter对象操作数据；



生命周期函数：

**beforecreate : 可以在这加个loading事件   
created ：在这结束loading，还做一些初始化，实现函数自执行   
mounted ： 在这发起后端请求，拿回数据，配合路由钩子做一些事情  
beforeDestory： 你确认删除XX吗？ destoryed ：当前组件已被删除，清空相关内容**

**如何完成数据共享：**

Actions:异步需求

Mutations:同步需求

**网络请求：**

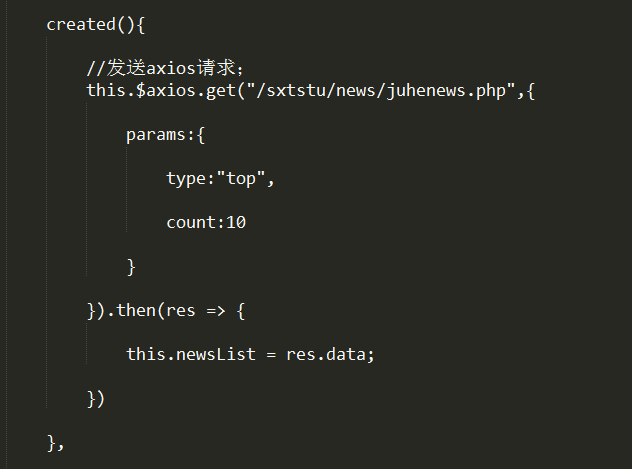
**1.resourse**

**2.axios：cnpm install axios -S**

网络配置：axios网络公共配置；main.js



Index.vue请求axios数据



动态创建类名：



动态创建类名-----  **:class =” [‘index-board-’+borad.id,{‘line-last’ : index%2 !==0}]”**

后面的大括号表示创建偶数个类名，三目运算符，等价于’line-last’? ’line-last’ : index%2 !==0;

**加载element-ui组件；**

cnpm i element-ui -S安装ui组件依赖；

cnpm install babel-plugin-component -D

安装plugin组件依赖；

更改.babelrcwen文件

见链接：http://element.eleme.io/#/zh-CN/component/quickstart

**文件打包编译：npm run build；**

Props：接收传递数据；[{}]

**码云git操作命令：**

Git add

Git commit -m ‘必须的’

Git push

Git pull

Git status

Git clone

Git操作命令：

echo "# 123" >> README.md

git init

git add README.md

git commit -m "first commit"

git remote add origin https://github.com/scottdao/123.git

git push -u origin master