# Angular03

## 复习

### 命令行

* 生成项目包: ng n 包名
* 启动: ng s -o
* 生成组件: ng g c 组件名
* 生成管道: ng g p 管道
* 生成指令: ng g d 指令名

### 用法

|  |  |  |
| --- | --- | --- |
| 写法 | 作用 | 示例 |
| {{}} | 显示变量, 可以做数学运算,逻辑,比较 | {{ 变量 }} |
| [属性名]="值" | 属性的绑定 | [src]="值" |
| (事件)="方法名()" | 事件的绑定, 必须带() | (click)="" |
| [innerHTML] | 绑定html内容 |  |
| [ngStyle]="{样式名: 值}" | 动态样式 | [ngStyle]="{color:'red'}" |
| [ngClass]="{样式类: true/false}" | 动态样式类 | [ngStyle]="{success:true}" |
| [(ngModel)]="" | 双向数据绑定. 必须手动加载 Forms 模块 |  |
| \*ngIf | 条件渲染 |  |
| \*ngFor="let item of items; let i = index" | 列表渲染 |  |
| {{值 | 管道}} | 系统管道 和 自定义管道 |  |
| <tag app指令名 /> | 指令用于修改DOM值 |  |
| 生命周期 | ngOnInit: 相当vue的创建<br>数据初始化->视图初始化<br>数据更新 -> UI更新<br>销毁 |  |

### TypeScript

是 微软公司 在 JavaScript 的基础上, 融入了 Java 的特性.

* 静态类型分析
  + 属性名:类型名
  + 优点: vscode 有代码提示 和 报错预警
* 类型:
  + 基础类型: number string boolean any(任意类型) string|number
  + 数组: Array<string> 等价于 string[] 代表数组中都是字符串类型
  + 数组: [string, boolean, string] 规定数组有几个值 且 都是什么类型
  + 自定义对象类型
  + let a : object = {}; object就是{} 空对象  
      
    关键词: interface 接口  
    interface 模板名{  
     属性名: 类型;  
     属性名: 类型;  
    }
* 访问控制词
  + public 公开的 类外 类内 子类
  + protected 保护的 类内 子类
  + private 私有的 类内

## 服务

vue 中的 Vuex. 状态管理器

* 全局组件的状态分享: 登录状态
* 组件间的数据共享: 购物车

Vuex 可以使用 webStorage 替代. 特色是 有变化时 会自动更新相关的组件

创建组件:

* ng g c myc01
* ng g c myc02

服务的生成命令

ng generate service 服务名  
ng g s 服务名

例如: ng g s skill

import { Component, OnInit } from '@angular/core';  
import { SkillService } from '../skill.service';  
  
@Component({  
 selector: 'app-myc01',  
 templateUrl: './myc01.component.html',  
 styleUrls: ['./myc01.component.css'],  
})  
export class Myc01Component implements OnInit {  
 // skills = ['axios', 'vue', 'Vuex', 'jQuery', 'React'];  
  
 // 在 vue 中: this.$store.state.skills  
 // 在 ng 中 , 服务的引入稍微麻烦  
  
 // 变量名:类型名; 就有2个优点: 代码提示 和 报错预警  
 abc: SkillService; //只有成员属性 才能在 html 中使用  
  
 // 依赖声明: 当前组件实例化 必须传递一个 SkillService 类型的参数  
 // 注入: 自动化操作-- 系统会自动实例化当前组件. 实例化时会传递 依赖的 类型的参数  
 constructor(skillS: SkillService) {  
 // 理论上 此处应该写 this.skillS = skillS; 比较合理  
 // 此处写 abc 是为了防止有同学以为: 名字必须一样  
 // 变量名是随意的, 但是应该有含义!  
 // 变量名 应该有自注释效果 -- 看名字 能猜出大概作用!  
  
 this.abc = skillS;  
  
 // 类似vue中, 网络请求完毕时的 this.data = res.data.data; 操作  
 }  
  
 ngOnInit(): void {  
 // this.abc  
 }  
}  
  
/\*\*  
 \* 程序中的 依赖注入 机制  
 \*  
 \* 生活中的例子:  
 \* 1. 超市门口的摇摇车: 标识 1元/次. 松松小朋友要玩, 则需要投币1元 才可以  
 \* 声明依赖: 1元/次 使用注入: 要玩就要投币1元  
 \*  
 \* 2. 新闻: 丈母娘说: 我家女儿20万彩礼...  
 \* 声明依赖: 20万 使用: 给20万  
 \*/  
  
// 声明依赖: 要一个 string 类型的 参数  
function show(name: string) {  
 console.log(name);  
}  
  
// 调用时： 就必须传递 string 类型的参数  
show('mike');  
  
class Demo {  
 // 构造方法: 实例化时触发  
 constructor(name: string) {}  
 // 依赖注入机制: 声明要 string 类型的参数  
}  
  
// 使用时 就必须传递 string 类型的参数  
new Demo('mike');

<p>myc01 works!</p>  
  
<ul>  
 <!-- abs: 绝对值 -->  
 <!-- 生成命令: ng g p abs -->  
 <li>{{ 9 | abs }}</li>  
 <!-- 生成指令 ng g d danger -->  
 <li appDanger>{{ -9 | abs }}</li>  
</ul>  
  
<ul>  
 <li \*ngFor="let item of abc.skills; let i = index">  
 <span>{{ item }}</span>  
 <button (click)="abc.skills.splice(i, 1)">删除</button>  
 </li>  
</ul>

练习: 生成服务, 保存人的名字

ng g s name

## 网络请求服务

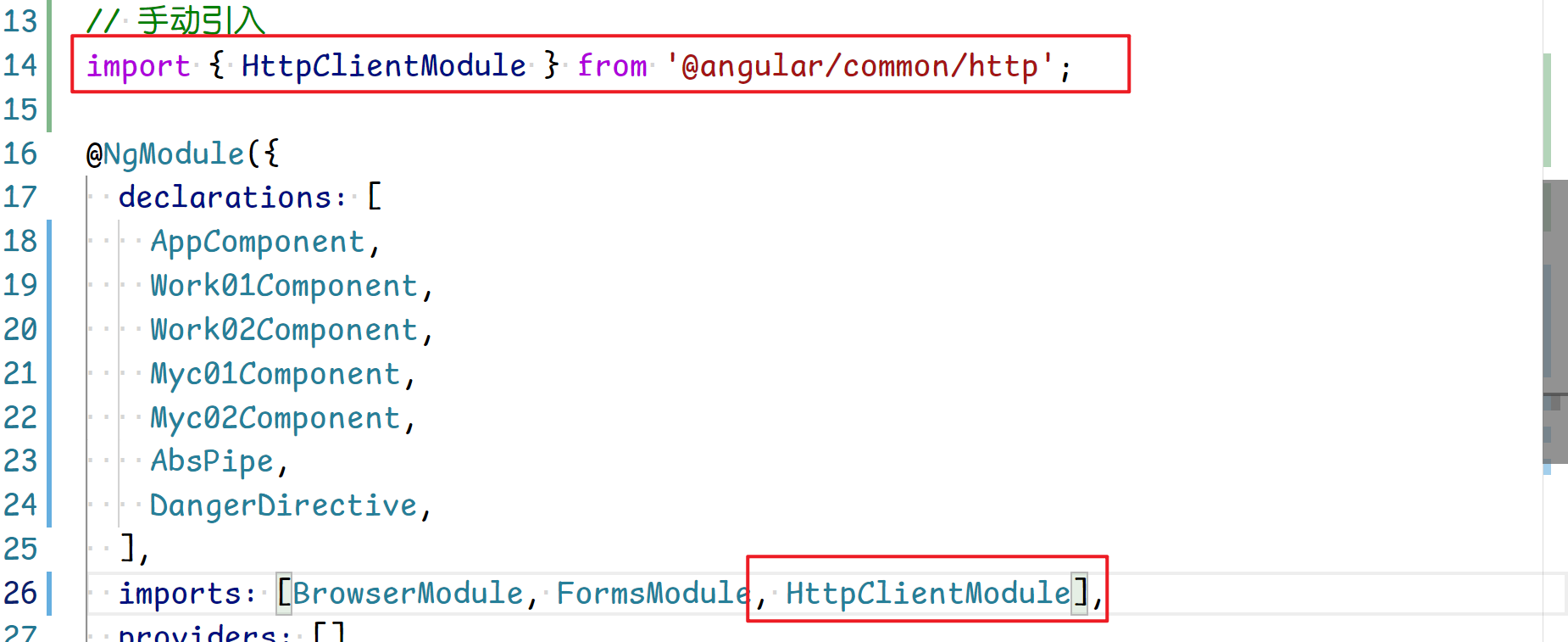
系统提供了很多完整的服务, 可以直接使用, 例如 **网络服务**

vue 使用 axios 进行网络请求.

angular 本身自带 网络服务, 不需要第三方

类似于 双向数据绑定 需要手动加载 Forms 模块;

网络服务 默认不加载, 必须手动进行模块的添加



生成组件: ng g c myc03

import { HttpClient } from '@angular/common/http';  
import { Component, OnInit } from '@angular/core';  
  
@Component({  
 selector: 'app-myc03',  
 templateUrl: './myc03.component.html',  
 styleUrls: ['./myc03.component.css'],  
})  
export class Myc03Component implements OnInit {  
 // 声明依赖  
 constructor(public http: HttpClient) {}  
  
 ngOnInit(): void {  
 // 请求的方式: 常见4种  
 // GET POST PUT DELETE -- RestFul 服务器风格  
 // GET查询数据. POST更新数据 PUT增 DELTE 删除  
 // 最常见的是 GET POST  
 let url = 'https://api.apiopen.top/getImages?page=7';  
 // axios: this.axios.get(url).then(res=>{})  
  
 // subscribe: 订阅 结果.. 单词与axios不同, 原理一样  
 this.http.get(url).subscribe((res: MeiTu) => {  
 console.log(res);  
 //res默认是 Object , 不同类型赋值 vscode 会预警  
 //要告诉vscode res是 MeiTu 类型, vscode 就不会报错  
 this.res = res;  
 });  
 }  
  
 // 属性才能在 html 中使用  
 res: MeiTu;  
}  
  
////////////////////////////////////////////  
////////////// 自定义返回值的数据类型 ////////  
///////////////////////////////////////////  
interface MeiTu {  
 code: number;  
 message: string;  
 result: Result[]; // 数组类型 中的值 是 任意类型  
}  
  
interface Result {  
 id: number;  
 img: string;  
 time: string;  
}

<p>myc03 works!</p>  
  
<!-- Cannot read property 'result' of undefined -->  
<!-- 不能够对 undefined 读取 result 属性 -->  
<!-- 请求的异步性: 发请求 不影响页面的渲染. 请求完毕前 res 没有值, 页面渲染要用. 此时就会报错 -->  
<!-- 判断: res 有值之后 再渲染相关的页面 -->  
<div \*ngIf="res">  
 <img  
 \*ngFor="let item of res.result"  
 [src]="item.img"  
 alt=""  
 width="80px"  
 height="80px"  
 />  
</div>

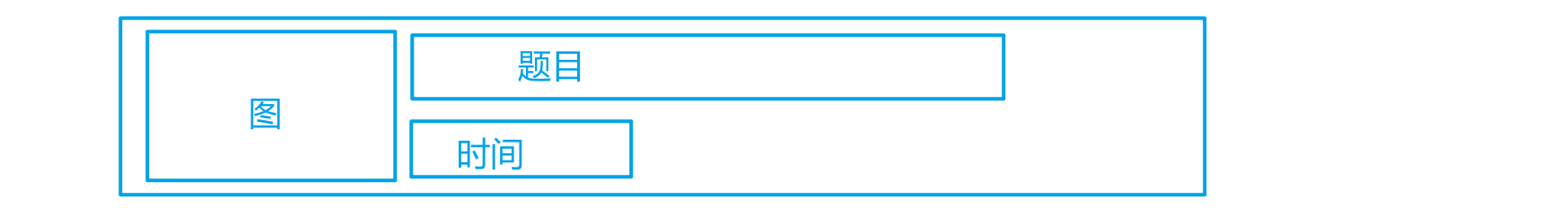
## 网易新闻练习

接口: https://api.apiopen.top/getWangYiNews

组件名: myc04

tips: 引入 网络服务 , 发送请求, 进行展示. 注意 返回值的类型声明

每一条数据的样式如下:



## 跨域

跨域问题常见于 **前后端分离** 项目

前端有自己的服务器, 后端也有自己的服务器. 让服务器压力减轻!

跨域: 浏览器的同源策略

网页访问接口时, 必须 协议 域名 端口号 都一致, 否则被认为不安全, 会阻止访问

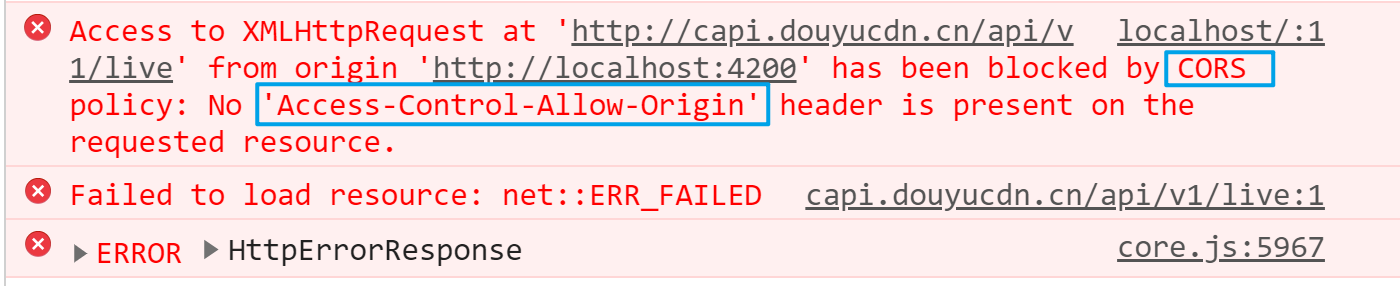
网址结构: 协议://域名:端口号 例如 http://localhost:8080

跨域解决方案: 百度上有9种, 而常用的是3种

* cors: 纯服务器解决, 添加跨域模块即可! **最常用,不需要前端做操作**
* jsonp: 服务器要反馈固定类型的数据结构, 前端也要发送固定结构的请求
  + 原理: 前端是通过script 的 src 发送请求. 同源策略 管不到 src, 只管html
* proxy
  + vue, angular, nginx 都带 proxy 方式解决

生成组件 myc05

跨域报错:



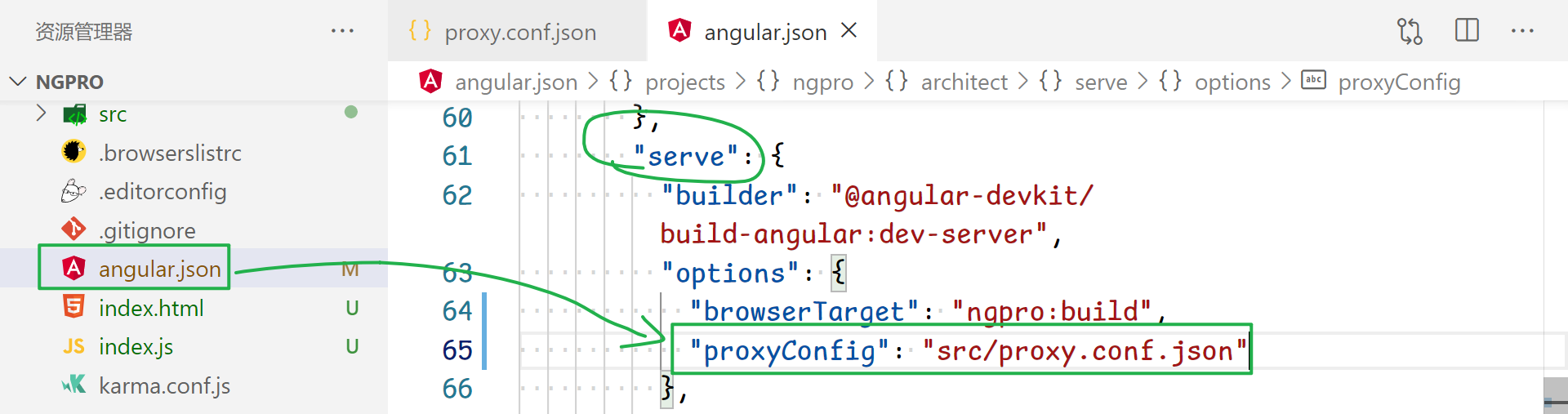
解决方案: 同 vue

参考地址: https://angular.cn/guide/build#proxying-to-a-backend-server

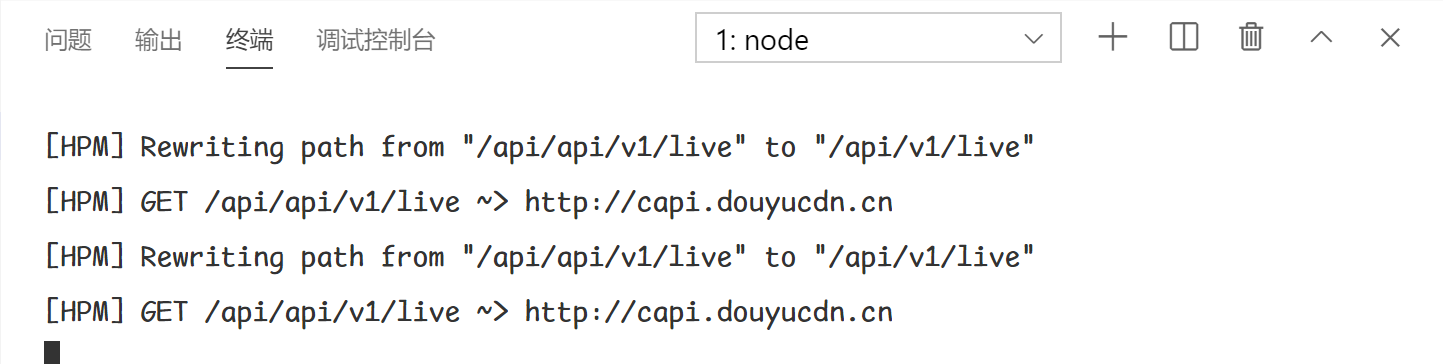
1. 在项目的 src/ 目录下创建一个 proxy.conf.json 文件



1. angular.json 中为 serve 目标添加 proxyConfig 选项：



1. 重启服务器即可

* 
* image-20201229175150318

## 作业

接口地址:

http://101.96.128.94:9999/mfresh/data/news\_select.php?pageNum=1  
参数 pageNum 是页数  
返回值中 pageCount是总页数 pageNum 为当前页

效果参考原网站: http://101.96.128.94:9999/mfresh/news.html

难点: 分页. **参考上午作业 的轮播图**



一些免费的接口, 在 FTP 的 **/18\_Angular/xxx.pdf** 中