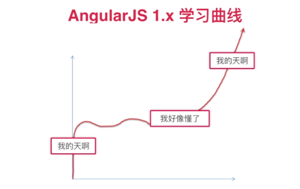
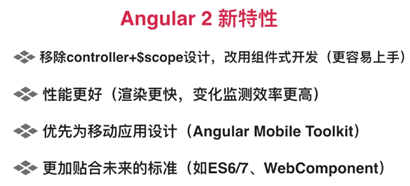
# Angular历史

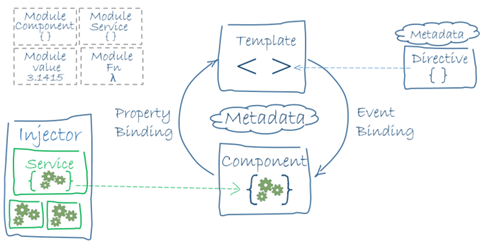
Angular是Google推出的Web前端开发框架，从12年发布起就受到了强烈的关注，他首次提出了双向绑定的概念，让人耳目一新。



**Angular 2特性**

就在2016年9月中旬，时隔4年，Google正式发布了Angular的第二代开发框架Angular 2，2017年3月推出了Angular4。Angular 5 出来了，不过你也不用太过担心，因为不论是Angular 4 还是Angular 5，其内核依然是Angular 2。从 2.0 版本开始出的新版本会保持向后兼容，不会出现 1 到 2 这么大的变化.

****



这个架构图展现了 Angular 应用中的 8 个主要构造块：

* 模块 (module)
* 组件 (component)
* 模板 (template)
* 元数据 (metadata)
* 数据绑定 (data binding)
* 指令 (directive)
* 服务 (service)
* 依赖注入 (dependency injection)

1. **1.       模块 (module)**

Angular 应用是模块化的，并且 Angular 有自己的模块系统，它被称为 *Angular 模块*或 *NgModules*。

Angular 或者 Ionic 新建的项目，都会存在一个根模块，默认名是 AppModule。如果你使用了模块化来创建应用，包括 AppModule，每个都会存在一个 @NgModule 装饰器的类(很像 java 中的注解，但官方命名叫装饰器)。新建的页面，如果不使用懒加载，都要在 @NgModule 中先声明后使用。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | //app.module.ts    import { NgModule }      from '@angular/core';  import { BrowserModule } from '@angular/platform-browser';  @NgModule({    imports:      [ BrowserModule ],    providers:    [ Logger ],    declarations: [ AppComponent ],    exports:      [ AppComponent ],    bootstrap:    [ AppComponent ]  })  export class AppModule { } |

* imports 本模块声明的组件模板需要的类所在的其它模块。
* providers 服务的创建者，并加入到全局服务列表中，可用于应用任何部分。
* declarations 声明本模块中拥有的视图类。Angular 有三种视图类：**组件、指令和管道**。
* exports declarations 的子集，可用于其它模块的组件模板。
* bootstrap 指定应用的主视图（称为根组件），它是所有其它视图的宿主。只有根模块才能设置 bootstrap 属性。

1. **2.       组件 (component)、模板 (template)、元数据 (metadata)**

[复制代码](javascript:void(0);)

//hero-list.component.ts

@Component({

selector: 'hero-list',

templateUrl: './hero-list.component.html',

providers: [ HeroService ]

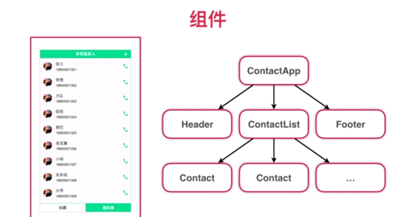
})

export class HeroListComponent implements OnInit {

/\* . . . \*/

}

[复制代码](javascript:void(0);)



组件就是一个类，是Angular的核心。**@Component** 是组件元数据装饰器，用来修饰一个组件，他能接受一个配置对象， Angular 会基于这些信息创建和展示组件及其视图。

* selector：CSS 选择器，它告诉 Angular 在父级 HTML 中查找 <hero-list> 标签，创建并插入该组件。
* templateUrl：组件 HTML 模板的模块相对地址，如果使用 template 来写的话是用“`”这个符号来直接编写 HTML 代码。
* providers：组件所需服务的依赖注入。

**metadata**   
**元数据**装饰器用类似的方式来指导 Angular 的行为。 例如 @Input、@Output、@Injectable 等是一些最常用的装饰器。

1. **1.       数据绑定 (data binding)**

这里一共展示四种数据绑定，看一下示例代码：

1)          //插值表达式    显示组件的hero.name属性的值

<li>{{hero.name}}</li>

2)          //属性绑定    把父组件selectedHero的值传到子组件的hero属性中，也可以把组件中的属性绑定到模板中元素的dom属性上。

<hero-detail [hero]="selectedHero"></hero-detail>

3)          //事件绑定    用户点击英雄的名字时调用组件的selectHero方法

<li (click)="selectHero(hero)"></li>

4)          //双向绑定    数据属性值通过属性绑定从组件流到输入框。用户的修改通过事件绑定流回组件，把属性值设置为最新的值

<input [(ngModel)]="hero.name">

1. **4.       指令 (directive)**

严格来说组件就是一个指令，但是组件非常独特，并在 Angular 中位于中心地位，所以在架构概览中，我们把组件从指令中独立了出来。  
我们这里提到的指令有两种类型：   
结构型 structural 指令和属性 attribute 型指令。

1)     **结构型指令**是 ngFor、ngIf 这种的，通过在 DOM 中添加、移除和替换元素来修改布局。

2)     **属性型指令**是 ngModel 这种的，用来修改一个现有元素的外观或行为。

Angular 还有少量指令，它们或者修改结构布局（例如 ngSwitch ）， 或者修改 DOM 元素和组件的各个方面（例如 ngStyle 和 ngClass）。

1. **5.       服务 (service)**

服务是**用来封装可重用的业务逻辑**

1. **6.       依赖注入 (dependency injection)**

依赖注入是提供类的新实例的一种方式，还负责处理类所需的全部依赖。**大多数依赖都是服务**。 Angular 使用依赖注入来提供新组件以及组件所需的服务。

比如我们要给某组件导入HeroService这个服务，看这段代码：

constructor(private service: HeroService) {

...

}

这个constructor就是构造函数，依赖注入在 constructor 中进行。前面写上 private、public（在构造函数内写上public会自动创建类的成员变量），这是 TypeScript 语法。

当 Angular 创建组件时，会首先为组件所需的服务请求一个注入器 injector。我们必须先用注入器 injector 为 HeroService 注册一个提供商 provider（服务）。

看一下下面的代码，意思就是我们必须在 providers 写上才能用（如果不在组件中注册，则需要在模块中注册）

@Component({

selector: 'hero-list',

templateUrl: './hero-list.component.html',

providers: [ HeroService ]

})

依赖注入主要用来导入服务。

# Angular-cli

简言之：就是NG团队自行维护的一个｀脚手架｀[内置单元测试及webpack2打包工具等]　－－　前身是ember-cli；

官网：<https://cli.angular.io/>   
Github: <https://github.com/angular/angular-cli>   
npm: <https://www.npmjs.com/package/angular-cli>

| **范围** | **命令** | **作用** |
| --- | --- | --- |
| new | ng new new\_project | 初始化新项目 |
| Component | ng g component my-new-component | 新建一个组件 |
| Directive | ng g directive my-new-directive | 新建一个指令 |
| Pipe | ng g pipe my-new-pipe | 新建一个管道 |
| Service | ng g service my-new-service | 新建一个服务 |
| Class | ng g class my-new-class | 新建一个类 |
| Interface | ng g interface my-new-interface | 新建一个接口 |
| Enum | ng g enum my-new-enum | 新建一个枚举 |
| Module | ng g module my-module | 新建一个模块 |

npm install -g @angular/cli

ng new my-app

cd my-app

ng serve –open

ng server –port 4321 --disable-host-check

生成组件的命令：

ng g component \*\*

创建 UserComponent 组件：ng g component user

angular-cli工具自动为我们生成的相关基本文件和代码

### 调试

Ng server的compiling

F12-source-webpack

易错：Typescript的类型

# 项目

## bootstrap 安装 ngx-bootstrap

**npm install ngx-bootstrap bootstrap –save**

bootstrap 安装需要些配置，可以网上找下或者在index.html中引入

可能缺少jq

**$ npm install jquery**

参考：

<https://www.cnblogs.com/zhuzy/p/8031718.html>

ngx-bootstrap网址 <http://ngx-bootstrap.com/#/buttons>

## 如何在Angular4中引入jquery

**npm install --save jquery**

### npm install @types/jquery --save

然后在组件中引入jquery即可使用：

### import \* as $ from 'jquery';

# 路由

# Http

# form

# 跨域

1.开机配置nginx（192.168.1.5），运行nginx

2.运行 ng server --disable-host-check

3.

前端请求：http://edu.xk.com/

html： http://edu.xk.com/xkweb/

接口请求：http://edu.xk.com/xkapi

### refrence

Angular中文文档  <https://www.angular.cn/docs>

英文版 <https://angular.io/guide/quickstart>

其中有live example：

<https://stackblitz.com/angular/nlvrooxxdvy?file=src%2Fapp%2Fheroes%2Fheroes.component.ts>

[TypeScript中文文档](https://www.tslang.cn/docs/handbook/basic-types.html) (https://www.tslang.cn/docs/handbook/basic-types.html)