重点中的重点：（心法篇）

1、@NgModule，<https://www.angular.cn/guide/bootstrapping>

想要在@NgModule中的declarations数组只能是可声明的变量：包括组件、指令、管道，其他的不可以

想要在模块中使用组件、指令、管道，就必须将他们写在@NgModule中的declarations数组中，

组件、指令、管道只能被一个模块声明（也就是写在@NgModule中的declarations数组），他们只要被声明一次即可，因为，有时我们会包含其他模块，其他模块如果声明了组件、指令、管道，我们就不需要再次声明

imports数值只能出现在@NgModule元数据中，因为他表示当前模块依赖的模块，通过imports，当前模块中的各部分就可以使用被imports模块那个的组件、指令、管道了

providers数组：这里列出来模块级别的服务，服务是通过组件类的构造函数来依赖注入的

2、ECMAScript模块和Angular的模块（NgModule）的区别就是：在ECMAScript中，文件就是一个模块，而Angular没有这方面的规定（又或者说压根就不是），在文件开头，需要使用import语句导入依赖的模块，这属于ECMAScript的模块语法，而导入后，我们需要在@NgModule类的元数据对象中的imports数组属性中列出在文件顶部imports进来的模块，所以说，ECMAScript模块和Angular模块是不同但又相互配合的

所以，简而言之，ECMAScript模块就是文件

而Angular模块是为组件、指令、服务（或者就说是应用）提供上下文的编译环境，Angular模块将应用的各部分内聚在一起，共同完成某项任务

3、一个模块中的组件如果想被其他模块使用（也就是出现在组件的模板中），就必须出现在exports数组中

4、入口组件：指Angular命令式加载的任意组件，Angular加载组件的两种方式：1、声明式，也就是声明在组件的模板中，2、命令式，

而命令式又分为两种类型：1、引导用的根组件，2、在路由定义中指定的组件

命令式和声明式加载组件的区别可以这样理解：例如根组件，你可以发现，我们使用根组件，并没有在根模块中的exports数组中指明！而例如特性组件，我们使用它是需要在模板中声明，并写入模块的exports数组中

所有的路由组件都必须是入口组件！

命令式加载的组件都必须在@NgModule元数据的entryComponents数组中，不过平时我们使用入口组件（不管是两种方式的哪一种，Angular都自动帮我们把入口组件添加到这个数组中了，并不需要我们手动去做）

5、超级重点：Angular只会为能从entryComponents数组中直接或者间接访问到的组件生成代码！！，[这与@NgModule中的declarations](mailto:这与@NgModule.declarations)数组无关，所以，如果一个组件既不是入口组件，又没有在模板中使用过，那么就会被tree shaker工具排除

6、限定服务提供商的三种方式（每种级别的提供商都会添加到对于级别的注入器中！提供商可以理解某种级别的作用域，而提供商提供的服务就是对应类的实例）：也就是三中级别的服务

1、@Injectable.provideIn：’root’,应用级别的服务提供商，会添加到根注入器中，

2、@Injectable.providerIn：xxx模块名（惰性加载模块），模块级别的服务供应商，会在跟注入器下创建子注入器，并把跟注入器中的服务供应商添加到子注入器中！！，而该模块会倾向于使用子注入器中的服务实例！！！，而外部模块任然使用根注入器的服务实例。（在模块中指定服务的提供商：@NgModule.providers: [服务名]，这样就不需要@Injectable.providerIn：xxx模块名）

3、@Component.providers：[服务名]，

7、共享模块：一个模块A可以imoprt另外一个模块B，而不放在imoprts数组中，直接放在exports数组中，这样如果其他模块导入模块A，那么他就可以使用模块B，而不需要再次import模块B

模块可以导出组件、指令、管道、甚至另外一个模块，但是，值得注意的是：导入带有服务的模块，你会拥有该服务的一个新的实例！这通常不是你所希望的结果，所以，共享服务常见的方式是通过Angular的依赖注入系统，而不是依靠模块导入

8、关于NgModule的一些知识：

1、exports数组：默认情况下，NgModule中的可声明对象都是私有的，也就是说，如果模块A导入了commonModule，模块B不会因为其导入了模块A而可以使用CommonModule中的东西，例如\*ngIf指令，除非，模块A重新导出了CommonModule！（好像AppModule例外？？）

2、bootstrap数组：要自动启动的组件列表，可以有多个，通常只有一个，也就是根组件！他们会被自动添加到entryComponents数组中！！

3、entryComponents数组：那些可以动态加载进视图的组件列表

入口组件包括：1、根组件，2、路由组件，因为你都需要动态加载他们，所以他们都会被自动添加到NgModule元数据对象的entrtyComponents数组中，所以，如果你需要动态加载组件，你就必须将他们添加在entryComponents数组中！，

9、forRoot()和forChild()静态方法：

1、forRoot()掉渣天，配置全应用级的Router服务，也就是单例服务，但是还是没有@Injectable中指定providedIn:’root’吊（默认为单例服务），

2、RouterModule .forRoot()用于配置全应用级的Router服务，而RouterModule .forChild()用于配置惰性加载模块的路由

具体知识点篇：

NgModule为组件集声明了编译的上下文环境，它可以将其组件和一组相关的代码（如服务）关联起来，形成功能单元

组件和模块共同定义了Angular的视图

Angular在每个javascript事件循环中处理所有的数据绑定，从组件树的根部开始，递归处理全部子组件

其实，组件从技术角度来说就是一个指令，但是他的特殊性，angular专门为他定义了一个@Component装饰器，他使用了面向模板的特性扩展了@Directive装饰器

指令包括：结构型指令（添加或移除dom元素）、属性型指令（改变元素外观，看起来像html元素的属性）

当Angular创建组件类的新实例时，它会通过查看该组件类的构造函数，来决定该组件依赖哪些服务或者其他的依赖项

三种级别的服务：

1、根级别（在@Injectable中使用providedIn:’root’属性）：单一共享实例，任何类都可以使用，（很好的一种方式，这样Angular可以通过移除那些从来没有被使用的服务来优化应用）

2、模块级别（在@NgModule中使用providers:[]属性）：同一模块下的所有组件共享同一实例，

3、组件级别（在@Component中使用providers:[]属性）：该组件的每一个实例都会新创建一个服务实例

差值表达式：包括模板表达式、模板语句

模板表达式：双括号之间的东西叫模板表达式（并不包括双括号，在属性绑定中，他是用双引号包裹起来的，出现在等号右侧，注意，在事件绑定中，出现在右侧用双引号包裹起来的叫：模板语句），Angular会对其求值，并转化为字符串，与相邻的字符串字面量连接起来

注意，Angular会将模板表达式转化为属性绑定，并不是简单的赋值操作

被禁用的javascript表达式：<https://www.angular.cn/guide/template-syntax>，赋值、new运算符、自增自减运算符，不支持位运算|和&，拥有新的运算符：（|）、（？.）、（!）

表达式的上下文：三种来源：

1、模板变量（模板输入变量、模板引用变量）

2、指令的上下文变量

3、组件对象实例

优先级依次降低

注意，模板表达式不能引用全局命名空间中的任何东西，比如window、document，也不能调用console.log、Math.max等，他们只能引用表达式上下文中的成员

最佳实践：最好使用幂等的模板表达式，因为他完全没有副作用，幂等表达式总是返回完全相同的东西

常见的模板表达式：属性名或者方法的调用

模板语句：用来相应有绑定目标（如 HTML 元素、组件或指令）触发的事件，有副作用，但是，这是事件处理的关键

注意：模板语句和模板表达式不同，他支持：赋值（=）、表达式链（;和,），但是，new、自增自减、操作并赋值（+=、-=）、位操作符、模板表达式操作符等不支持

语句的上下文：（注意与表达式的上下文的联系与区别）两种来源：

1、组件实例

2、模板自身的上下文（例如：模板的$event对象、模板变量）

优先级依次升高

同表达式上下文一样，他不能引用全局中的任何东西

函数调用、属性赋值是常见的模板语句

注意：数据绑定绑定的是DOM、组件、指令中的Property或事件，而不是html的attribute，这两者有很大的区别

在各种绑定中，出现在等号左边的是目标名（DOM中的某些东西，property或者事件），就是Property，而不是attribute，见<https://www.angular.cn/guide/template-syntax>

HTML+：用封装了 HTML 的组件创建新元素，并把它们当作原生 HTML 元素在模板中使用。也就是指令、组件扩展的html

属性绑定还是差值表达式：当目标需要的是字符串，推荐使用差值表达式，当目标需要的是对象，就只能使用属性绑定

重点：属性绑定如果未加[],会报什么错？

attribute绑定（attr前缀）：因为有些元素并没有相应的property，例如：colspan，这样就需要attribute绑定了

css类绑定（class前缀）：三种形式：

1、覆盖：直接使用[class]=”xx”的形式，

2、添加或者移除某类：[class.xx] = “返回true/false的表达式”，设置单一css类

3、使用NgClass指令，设置多个css类

style样式绑定（style前缀）：两种形式：

1、使用[style.xxx]=””，设置单个样式

2、使用NgStyle指令，设置多个内联样式

事件绑定：有等号左边的目标事件和等号右边的模板语句组成

Angular解析属性绑定和事件绑定的顺序：都是先指令再元素！，为什么会这样，因为有些指令它就像元素的属性一样写在元素中，比如说routerLink指令

注意：赋值给\*ngFor的语句不是模板表达式，他angular的微语法——有angular自己解释的小型语言

模板引用变量：使用#来声明一个模板引用变量：用来引用模板中的某个DOM元素，它还可以引用Angular的组件或者指令甚至web component，你可以在模板的任何地方使用模板变量，

其作用域为整个模板，可以使用ref-代替#

重点：本组件的属性若需要绑定到其他组件，就必须带@input（属性绑定）、或者@output（事件绑定）属性，而若是绑定本组件的公共属性，则无需带上他们，因为他们是组件的组成部分（组件类+模板），

重点：而绑定到其他组件的属性时，其他组件的属性位于等号左侧

终于理解了：

为什么@input的表示流入，而@output表示流出！，流入表示：再属性绑定中，等号右边的属性（本组件的属性）流入到等号左侧的属性（其他组件的属性），同理

想要让其他组件访问本组件的某些属性，（Typescript的public是不能达到这种效果的），就必须在属性前面加上@input或者@output装饰器

1、管道操作符：在绑定之前对数据进行处理

2、安全导航操作符：(?.)，会在遇到属性路径上面遇到第一个空值就跳出！并显示为空,这个使用&&是一个道理，但是使用?.更好，所以下面这两条模板表达式是等价的：

{{a?.b?.c}}, {{a&&a.b&&a.b.c}}

3、非空断言操作符（!）：如果类型检查器在运行时刻无法确认一个变量是否为null或者undefined，它就会抛异常（在开启了--strictNullChecks 标志强制开启严格空值检查），为了阻止这一行为，可以使用非空断言操作符!，

与安全导航操作符不同，它不会阻止出现null或者undefined，它只是告诉类型检查器对于特定类型不要做“严格空值检查”

4、类型转换函数（$any()）

以上四个是模板表达式特有的操作符

标准的DOM事件都有一个target对象，用来引用触发该事件的元素

按键事件的过滤：(keyup.enter)这表示只监听enter按键

失去焦点事件：(blur)

路由与导航

路由器需要先配置才会有路由信息的，每个路由都会把一个URL映射到一个组件，path不能以（/）斜杠开头，path为空表示默认路径，path为\*\*表示（全匹配），当路径不匹配前面任何路径时，就会选择这个路由

路由器匹配的规则：先匹配者优先，所以，越具体的路由应该越靠前

RouterOutlet指令：一个占位符，当路径匹配时，对应的组件将作为<router-outlet>的兄弟元素放在它之后

路由文件的导入顺序很重要，因为，路由配置是按照导入顺序追加的

只用在根模块中使用RouterModule.forRoot来注册顶级路由，其他模块使用RouterModule.forChild来注册附属路由即可

路由参数（必选参数）指的是在路径中的参数，查询参数（可选参数）指的是?之后的参数，并不是路径的一部分

这些参数都可以通过paramMap这个可观察对象得到get（）方法

Observable可观察对象是用来复用组件实例的

矩阵URL标记法（Matrix URL）：查询参数之间使用;分割，而不是?和&，这种写法虽然为曾写入html标准，但是在浏览器的路由系统中，它作为从父路由和子路由中单独隔离出参数的方式而广受欢迎，Angular路由器就是这样一个路由系统

Angular应用推荐的模式：

1、把每个特性放在自己的目录

2、每个特性都有自己的Agular的特性模块

3、每个特性区都有自己的根组件

4、每个特性区的根组件都有自己的路由出口以及其子路由

5、特性区的路由很少（完全不）与其他特性区的路由交叉

在链接参数数组使用相对路径导航：1、如果使用routerLink指令，则不需要使用relativeTo属性（因为该指令以及带上了该属性），而使用router.navigate方法这需要带上该属性

惰性加载和重新配置只会在首次请求时候发生！后续的请求，该模块和路由都是立即可用的！

canActivate守卫：阻止用户访问未授权的页面，但是路由器仍然会加载该模块

canLoad守卫：和canActivate守卫的区别就是：此时路由器并不会加载该模块，而且，它会堵塞预加载

resolve守卫：预先获取组件需要的数据，而不是先激活路由，这样有可能先导航到一个空组件（因为数据还没有准备好），用户体验不是很好

立即加载、惰性加载（按需加载）、预加载（异步加载）模块，三者的区别和应用场景！！

1、首页（也就是用户打开应用第一眼看到的界面）应该立即加载！

2、用户不常访问的页面，应该采用惰性加载！

3、用户经常访问的，但是又不是首页，应该采用预加载

在这里可能会疑惑，预加载所有惰性加载的模块不就好了吗？但是有些场景是不正确的：例如在移动设备和低宽带的场景下！或者没有流量的场景下

预加载的原理：在每次成功导航后，路由器会在自己的配置中查找尚未加载并且可以预加载的模块

路由器内置的三种预加载策略：

1、完全不预加载，这是默认值。惰性加载的特性区间仍然会按按需加载！

2、预加载所有的惰性加载（带loadChildren属性的路由）的特性区

3、自定义预加载

PreloadAllModules策略不会预加载被CanLoad守卫保护的特性区，所有，如果你想预加载，有想防止未授权访问，那就只能依赖CanActivate守卫了，而不能添加CanLoad守卫

路由器在每一层的路由配置只会处理一次重定向，这样可以防止无限循环重定向

查看路由器最终配置信息的方法：

export class AppModule {

// Diagnostic only: inspect router configuration

constructor(router: Router) {

// Use a custom replacer to display function names in the route configs

const replacer = (key, value) => (typeof value === 'function') ? value.name : value;

console.log('Routes: ', JSON.stringify(router.config, replacer, 2));

}

}

链接参数数组：使用对象来表示可选参数

index.html中的<base href=”/”>标签是用来使浏览器的history.pushState API导航功能正常使用的！

浏览器要使用<base href>的值为引用CSS、脚本和图片文件时使用的相对UR了来添加前缀！如果不这样做，当通过“深链接”（直接将应用的链接站台进浏览器的地址栏中，或者点击邮件中的链接！）来进入应用时，这些资源的加载就会失败

依赖注入（DI）

定义：类从外部源的请求中获取依赖，而不是自己创建他们