### 一级目录结构

1.e2e

端对端的测试文件

2.nodes\_modules

依赖库

3.src

开发工作目录，页面，样式，脚本和图片

4.angular.json

Angular配置文件

5.ionic.config.json

ionic配置文件

6.package.json

清单文件

7.tsconfig.json

TypeScript项目编译配置文件

1. tslint.json

TypeScript语法限制（格式化）配置文件

<---------------------运行ionic build --prod 生成以下文件------------------->

1. www

静态文件，可以直接发布到web服务器。

<----运行ionic cordaova platform add ios/android 生成以下文件------->

10.platforms

android或ios平台文件。

11.Plugins

Cordova安装的插件。

1. config.xml

打包成app的配置文件

### 二，二级目录

1.src/app

项目模块组件

2.src/assets

静态资源文件

3.src/environments

配置运行环境

1. src/theme

主题相关

5.src/global.scss

全局的css文件

### 三，三级目录

1.src/app/tabx

Tab切换页面

2.src/app/app-routing.module.ts

路由配置文件

1. 根组件

src/app/app.component.html

Src/app/app.component.spec.ts

Src/app/app.html //入口页的html模板

Scr/app/app.component.ts //入口页的业务逻辑

1. 入口模块配置

Src/app/app.module.ts //入口模块配置，告诉Angular如何编译和启动应用程序

### 四，根模块

Src/app/app.module.ts

//引入ionic angular核心文件

import { NgModule } from '@angular/core';

import { BrowserModule } from '@angular/platform-browser';

import { RouteReuseStrategy } from '@angular/router';

import { IonicModule, IonicRouteStrategy } from '@ionic/angular';

//ionic打包成app后配置启动画面以及导航条

import { SplashScreen } from '@ionic-native/splash-screen/ngx';

import { StatusBar } from '@ionic-native/status-bar/ngx';

//引入路由配置文件

import { AppRoutingModule } from './app-routing.module';

//引入根组件

import { AppComponent } from './app.component';

//@NgModule装饰器表明AppModule是一个NgModule类，它告诉Angular如何编译和启动本应用。

@NgModule({

  //声明属于这个模块的组件，指令，管道，声明后就可以在该模块中使用它们了。

  declarations: [AppComponent],

  entryComponents: [], //配置不会再模块中使用的组件

  //导入依赖的一些其他模块。

  imports: [BrowserModule, IonicModule.forRoot(), AppRoutingModule],

  providers: [ //各种共用服务

    StatusBar,

    SplashScreen,

    { provide: RouteReuseStrategy, useClass: IonicRouteStrategy }

  ],

  //指定这个模块启动时应该启动的组件，这个组件会被自动加入到entryComponents中去，Angular创建它并插入到index.html宿主页面。

  bootstrap: [AppComponent]

})

export class AppModule {}

### 五，路由模块

import { NgModule } from '@angular/core';

import { PreloadAllModules, RouterModule, Routes } from '@angular/router';

//配置路由的地方

const routes: Routes = [

  {

    path: '',

    loadChildren: () => import('./tabs/tabs.module').then(m => m.TabsPageModule)

  }

];

@NgModule({

  imports: [

    RouterModule.forRoot(routes, { preloadingStrategy: PreloadAllModules })

  ],

  exports: [RouterModule]

})

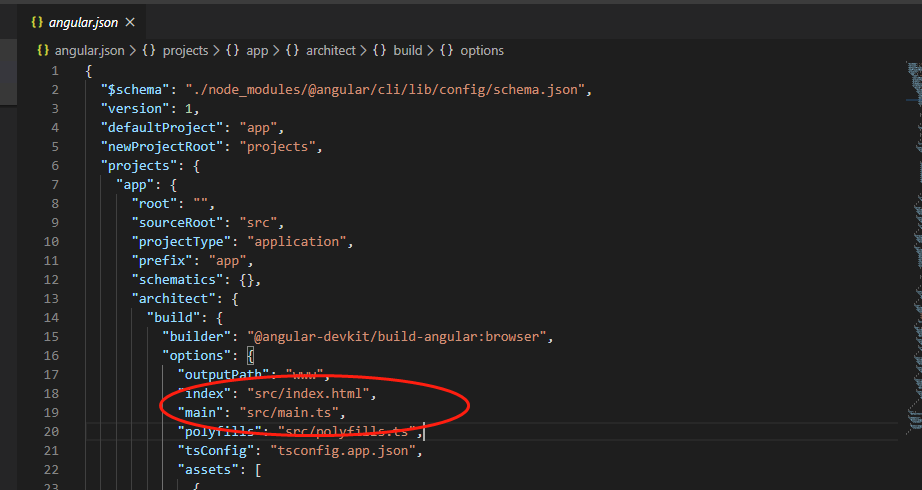
export class AppRoutingModule {}

### 六，项目启动流程

项目启动：Main.ts入口，index.html加载，AppModule加载，AppRoutingModule;

1. 启动时加载了哪个页面。
2. 启动时执行了那些脚本
3. 这些脚本都做了什么

启动配置文件(Angular.json):

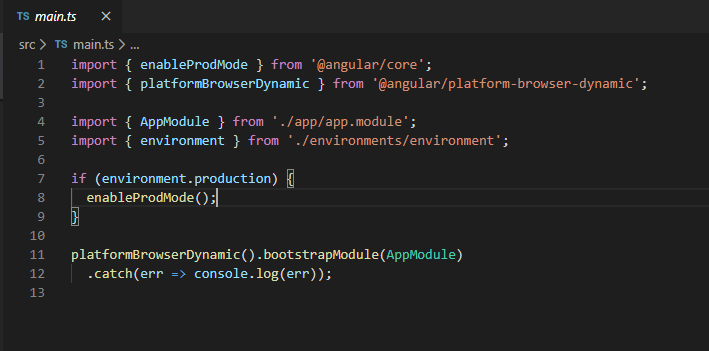


root属性：指定angular启动时从哪个目录找资源文件，默认是src。

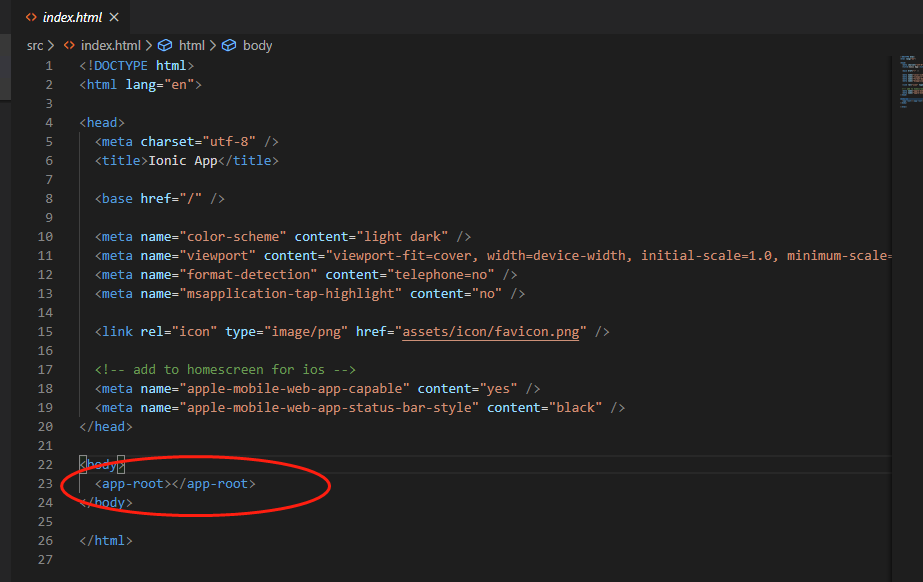
index：指定angular启动时加载的页面，默认是src/index.html

main:指定angular应用启动时加载的脚本，默认src/main.ts

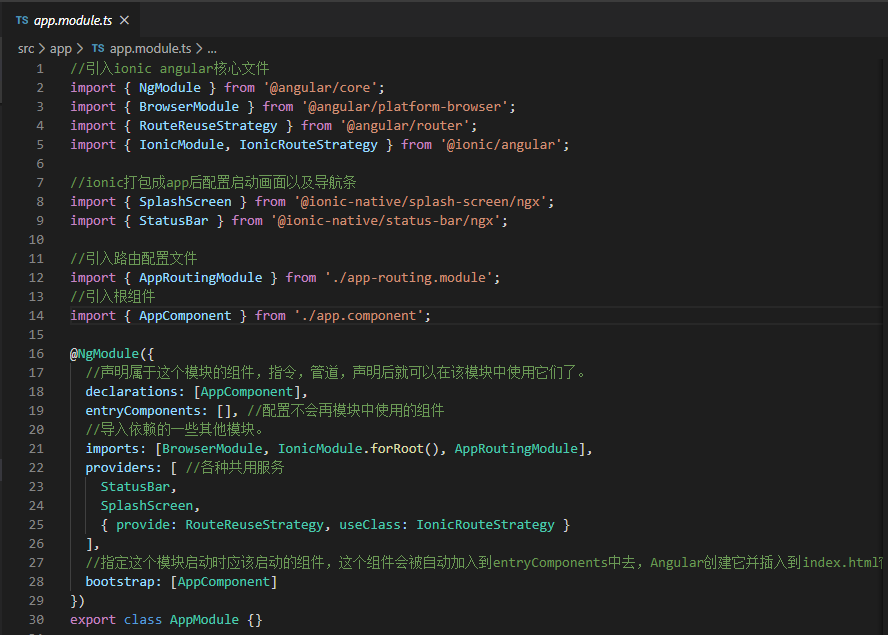
项目启动时会根据这个配置文件加载index.html页面，并执行main.ts脚本。



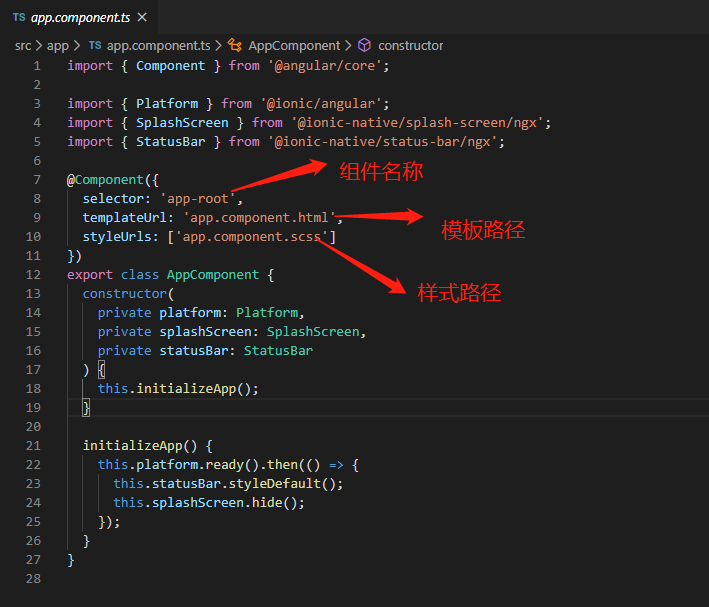
在main.ts里面加载了AppModule。



<app-root></app-root>这个标签是组件展示的地方。



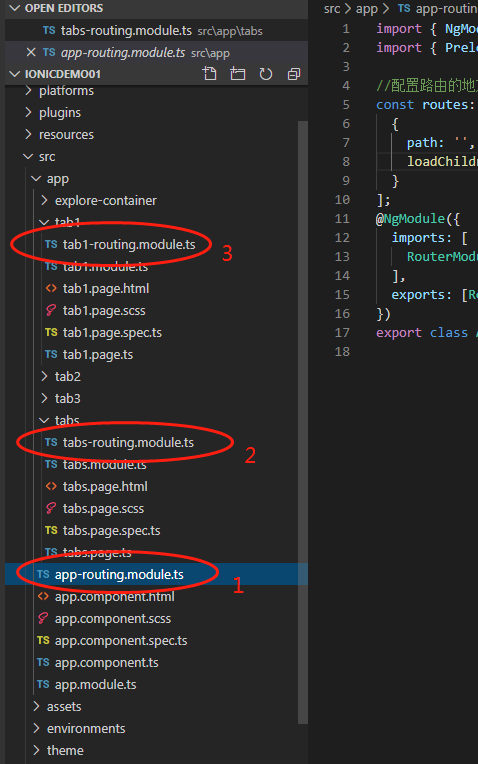
在AppModule中，加载并提供AppRoutingModuel，用于后面的Url加载。同时在AppModule中加载启动了AppComponent。



最后，根据component.ts的数据信息，将组件渲染到index.html。

### 七，url执行流程

执行路程是路由匹配的过程：



### 八，模块和组件

//模块

1.一个模块可以有多个组件，一个组件可以有多个样式表，但只有一个HTML模板。

2.模块是将一系列组件还有指令，服务整合封装到一起，提供一个完整的功能，模块可以被外部引用。

3.每个应用都有一个根模块，即AppModule，应用启动时，就会加载该模块。

//组件

1. 使用import语句导入需要的模块。
2. 使用@Component()来描述组件的元数据：

* Selector:组件名称，像HTML标签的名字一样。
* templateUrl:HTML模板路径，./表示在当前目录下。
* styleUrls:HTML模板使用的样式表路径，可以有多个。