Angular 2 JavaScript 环境配置 Angular 2 架构

Angular 2 TypeScript 环境配置

本章节使用的是 TypeScript 来创建 Angular 的应用，这也是官方推荐使用的，本教程的实例也将采用 TypeScript 来编写。

TypeScript 是一种由微软开发的自由和开源的编程语言，它是JavaScript的一个超集，扩展了JavaScript的语法。

如果你不了解TypeScript，可以查阅以下资料：

- TypeScript 入门教程
- TypeScript 中文手册

这开始前，你需要确保你已经安装了 npm，如果你还没安装npm或者不了解 npm 可以查看我们的教程：NPM 使用介绍。

由于 npm 官网镜像国内访问太慢，这里我使用了淘宝的npm镜像，安装方法如下：

```
$ npm install -g cnpm --registry=https://registry.npm.taobao.org
```

执行后我们就可以使用 cnpm 命令来安装模块：

```
$ cnpm install
```

第一步：创建与配置项目

创建目录

```
$ mkdir angular-quickstart
$ cd angular-quickstart
```

创建配置文件

Angular 项目需要以下几个配置文件：

- **package.json** 标记本项目所需的 npm 依赖包。
- **tsconfig.json** 定义了 TypeScript 编译器如何从项目源文件生成 JavaScript 代码。
- **typings.json**为那些 TypeScript 编译器无法识别的库提供了额外的定义文件。
- **systemjs.config.js** 为模块加载器提供了该到哪里查找应用模块的信息，并注册了所有必备的依赖包。 它还包括文档中后面的例子需要用到的包。

在 angular-quickstart 中创建以下几个文件，代码如下所示：

package.json 文件：

```
{
"name": "angular-quickstart",
"version": "1.0.0",
```

```
"scripts": {
"start": "tsc && concurrently \"npm run tsc:w\" \"npm run lite\" ",
"lite": "lite-server",
"postinstall": "typings install",
"tsc": "tsc",
"tsc:w": "tsc -w",
"typings": "typings"
},
"license": "ISC",
"dependencies": {
"@angular/common": "2.0.0",
"@angular/compiler": "2.0.0",
"@angular/core": "2.0.0",
"@angular/forms": "2.0.0",
"@angular/http": "2.0.0",
"@angular/platform-browser": "2.0.0",
"@angular/platform-browser-dynamic": "2.0.0",
"@angular/router": "3.0.0",
"@angular/upgrade": "2.0.0",
"core-js": "^2.4.1",
"reflect-metadata": "^0.1.3",
"rxjs": "5.0.0-beta.12",
"systemjs": "0.19.27",
"zone.js": "^0.6.23",
"angular2-in-memory-web-api": "0.0.20",
"bootstrap": "^3.3.6"
},
"devDependencies": {
"concurrently": "^2.2.0",
"lite-server": "^2.2.2",
"typescript": "^2.3.4",
"typings":"^1.3.2"
}
}
```

tsconfig.json 文件：

```
{
"compilerOptions": {
"target": "es5",
"module": "commonjs",
"moduleResolution": "node",
"sourceMap": true,
"emitDecoratorMetadata": true,
"experimentalDecorators": true,
"removeComments": false,
"noImplicitAny": false
}
}
```

typings.json 文件：

```
{
"globalDependencies": {
"core-js": "registry:dt/core-js#0.0.0+20160725163759",
"jasmine": "registry:dt/jasmine#2.2.0+20160621224255",
"node": "registry:dt/node#6.0.0+20160909174046"
}
}
```

systemjs.config.js 文件：

```
/**
* System configuration for Angular samples
* Adjust as necessary for your application needs.
*/
(function (global) {
System.config({
paths: {
// paths serve as alias
'npm:': 'node_modules/'
},
// map tells the System loader where to look for things
map: {
// our app is within the app folder
app: 'app',
// angular bundles
'@angular/core': 'npm:@angular/core/bundles/core.umd.js',
'@angular/common': 'npm:@angular/common/bundles/common.umd.js',
'@angular/compiler': 'npm:@angular/compiler/bundles/compiler.umd.js',
'@angular/platform-browser': 'npm:@angular/platform-browser/bundles/platform-browser.umd.js',
'@angular/platform-browser-dynamic': 'npm:@angular/platform-browser-dynamic/bundles/platform-br
owser-dynamic.umd.js',
'@angular/http': 'npm:@angular/http/bundles/http.umd.js',
'@angular/router': 'npm:@angular/router/bundles/router.umd.js',
'@angular/forms': 'npm:@angular/forms/bundles/forms.umd.js',
// other libraries
'rxjs': 'npm:rxjs',
'angular2-in-memory-web-api': 'npm:angular2-in-memory-web-api',
},
// packages tells the System loader how to load when no filename and/or no extension
packages: {
app: {
main: './main.js',
defaultExtension: 'js'
},
rxjs: {
defaultExtension: 'js'
},
'angular2-in-memory-web-api': {
main: './index.js',
defaultExtension: 'js'
}
}
```

```
});
})(this);
```

接下来我们使用 cnpm 命令来安装依赖包：

```
$ cnpm install
```

执行成功后，angular-quickstart 目录下就会生成一个 node_modules 目录，这里包含了我们这个实例需要的模块，我们可以看下项目的目录结构：

第二步：创建应用

我们用 NgModules 把 Angular 应用组织成了一些功能相关的代码块。

Angular 本身是被拆成一些独立的 Angular 模块，这样我们在应用中只需要导入需要的 Angular 部分。

每个 Angular 应用至少需要一个**root module(根模块)**，实例中为 AppModule 。

接下来我们在 angular-quickstart 目录下创建 app 目录：

```
$ mkdir app
$ cd app
```

然后在 app 目录下创建 app.module.ts 文件，代码如下所示：

app.module.ts 文件：

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
@NgModule({
imports: [ BrowserModule ]
})
export class AppModule { }
```

由于 QuickStart 是一个运行在浏览器中的 Web 应用，所以根模块需要从 @angular/platform-browser 中导入 BrowserModule 并添加到 imports 数组中。

创建组件并添加到应用中

每个 Angular 应用都至少有一个**根组件**， 实例中为 AppComponent，app.component.ts 文件代码如下：

app.component.ts 文件：

```
import { Component } from '@angular/core';
@Component({
selector: 'my-app',
template: '<h1>我的第一个 Angular 应用</h1>'
})
export class AppComponent { }
```

代码解析：

- 以上代码从 *angular2/core* 引入了 *Component* 包。
- *@Component* 是 Angular 2 的*装饰器* ，它会把一份元数据关联到 AppComponent 组件类上。
- *my-app* 是一个 CSS 选择器，可用在 HTML 标签中，作为一个组件使用。
- *@view* 包含了一个 *template* ，告诉 Angular 如何渲染该组件的视图。
- *export* 指定了组件可以再文件外使用。

接下来我们重新打开 app.module.ts 文件，导入新的 AppComponent ，并把它添加到 NgModule 装饰器的 declarations 和 bootstrap 字段中：

app.module.ts 文件：

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';
@NgModule({
imports: [ BrowserModule ],
declarations: [ AppComponent ],
bootstrap: [ AppComponent ]
})
export class AppModule { }
```

第四部：启动应用

接下来我们需要告诉 Angular 如何启动应用。

在 angular-quickstart/app 目录下创建 main.ts 文件，代码如下所示：

main.ts 文件：

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { AppModule } from './app.module';
const platform = platformBrowserDynamic();
platform.bootstrapModule(AppModule);
```

以上代码初始化了平台，让你的代码可以运行，然后在该平台上启动你的 AppModule。

定义该应用的宿主页面

在 angular-quickstart 目录下创建 index.html 文件，代码如下所示：

index.html 文件：

```
<html>
<head>
<title>Angular 2 实例 - 菜鸟教程(runoob.com)</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="styles.css">
<!-- 1. 载入库 -->
<!-- IE 需要 polyfill -->
<script src="node_modules/core-js/client/shim.min.js"></script>
<script src="node_modules/zone.js/dist/zone.js"></script>
<script src="node_modules/reflect-metadata/Reflect.js"></script>
<script src="node_modules/systemjs/dist/system.src.js"></script>
<!-- 2. 配置 SystemJS -->
<script src="systemjs.config.js"></script>
<script>
System.import('app').catch(function(err){ console.error(err); });
</script>
</head>
<!-- 3. 显示应用 -->
<body>
<my-app>Loading...</my-app>
</body>
</html>
```

这里值得注意的地方有：

- JavaScript 库： **core-js** 是为老式浏览器提供的填充库， **zone.js** 和 **reflect-metadata** 库是 Angular 需要的，而 SystemJS 库是用来做模块加载的。
- SystemJS 的配置文件和脚本，可以导入并运行了我们刚刚在 main 文件中写的 app 模块。
- <my-app> 标签是应用载入的地方

添加一些样式

我们可以在 angular-quickstart 目录的 styles.css 文件中设置我们需要的样式：

styles.css 文件：

```
/* Master Styles */
h1 {
color: #369;
font-family: Arial, Helvetica, sans-serif;
font-size: 250%;
}
h2, h3 {
color: #444;
font-family: Arial, Helvetica, sans-serif;
font-weight: lighter;
}
body {
margin: 2em;
}
```

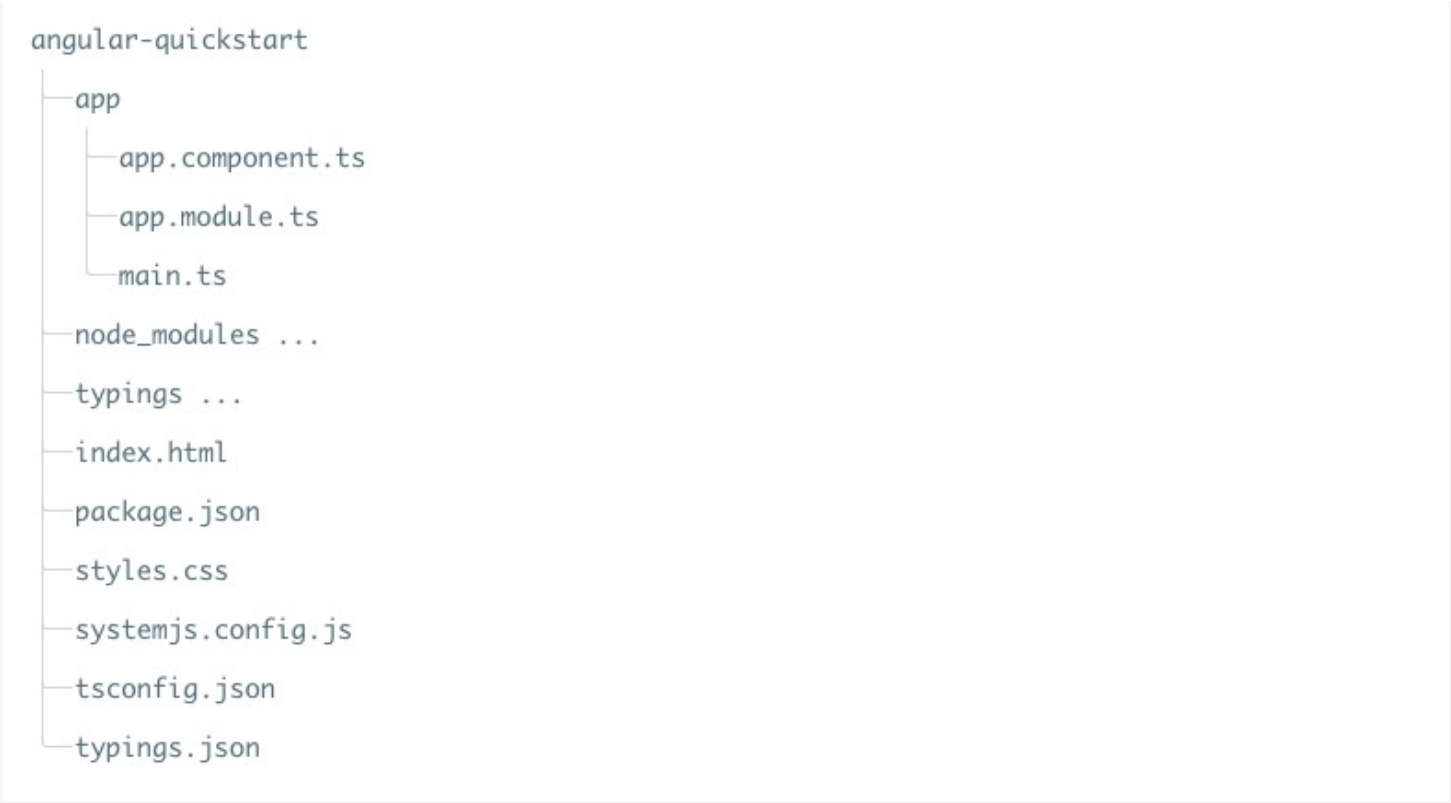
第六步：编译并运行应用程序

打开终端窗口，输入以下命令：

```
npm start
```

访问 http://localhost:3000/，浏览器显示结果为：

这样我们的第一个 Angular2 的应用就算创建完成了，最终的目录结构为：



本文所使用的源码可以通过以下方式下载，不包含 node_modules 和 typings 目录。

源代码下载

Angular 2 JavaScript 环境配置 Angular 2 架构

点我分享笔记