白俄罗斯国立大学
应用数学和信息学系
多处理器网络与系统系

Rafeenko E.D.

网络编程

单页面应用程序 (SPA)。Angular 框架

目录

- ▶ Angular 框架结构。
- ▶单页面应用程序（SPA）的概念。
- ▶创建 Angular 项目
- ▶TypeScript

角文学

1. https://angular.io

2. https://www.ngdevelop.tech/

3. 杰里米-威尔肯Angular in Action.- Manning Publications Co, 2018.

4.Freeman A. Angular for professionals.- SPb.：Peter，2018.

Angular 版本

Angular 是一个使用**模型-视图-控制器（MVC）**和**模型-视图-模型（MVVM）**架构开发客户端应用程序的框架。

已发布的版本如下

AngularJS 或 Angular 1.x。

代码由 JavaScript 编写。它由谷歌公司提供支持。旨在促进应用程序开发和测试。

最新版本为 1.7。

Angular 2- 完全重新设计的版本（2016 年）。代码采用 TypeScript 编写。

Angular 4、...、Angular 15（2022 年 11 月）。

JavaEE + Angular

学习目标

在服务器端（后端）使用 Java EE 平台，在客户端（前端）使用 Angular 框架开发网络应用程序。

单页应用程序

单页面应用程序（SPA）是指只有一个入口点，即一个 HTML 页面（可以是 index.html）的应用程序；所有应用程序内容都是从该页面动态添加和删除的。

在单页面网络应用程序中，原始 HTML 文档被发送到浏览器。用户的操作会产生 Ajax 请求，要求将 HTML 小片段或数据插入现有的元素集，并显示给用户。

角度设置

要使用 Angular 框架进行开发，您需要安装

Node.js 是在服务器上运行的 JavaScript 运行环境。在 Node.js 中，当新的 ECMAScript 标准在平台上实现时（使用 TypeScript），您可以快速迁移到这些标准。

npm 是 Node.js 软件包管理器。npm 注册表拥有 50 多万个开源软件包，任何 Node.js 开发人员都可以自由使用。

TypeScript 是 Angular2 开发中使用的语言 (>)

npm install -g typescript

Angular CLI

Angular CLI - 用于创建项目、生成应用程序和库代码、执行测试、链接和部署等持续开发任务。

您可以使用以下命令安装 **Angular CLI：**

npm install -g @angular/cli

Angular CLI 命令：

ng new my-app - 创建一个新的 Angular 项目；

cd my-app

ng serve -open - 启动服务器（Angular CLI 内置了 http 服务器），构建应用程序，并将其部署到服务器上。成功执行命令后，应用程序将默认在 url http://localhost:4200/ 上可用。

Angular 项目结构


tslint.json

Angular 项目结构

启动应用程序：

main.ts - 入口点

加载根模块：app.module.ts

加载根组件：app.component.ts

加载由根组件控制的显示内容：
app.component.html

index.html

Angular 架构

Angular 使用*模块化*来组织代码

一个名为 NgModule 的系统。

Angular 架构

每个 Angular 应用程序都至少有一个使用 @NgModule 装饰器的类。这就是根模块，通常称为 AppModule：

```
@NgModule({
  declarations：[
  AppComponent,
  BikeComponent,
  BikeInfoComponent
```

],
[
浏览器模块、表单模块
、Http 模块
],
[],
引导：[AppComponent].
})
导出类 AppModule { }

Angular 架构

组件是用户界面（UI）的基本构件。

Angular 应用程序是一棵组件树。

每个组件都映射到一个**模板**。

组件包含属性、方法、构造函数、输入事件、输出事件和生命周期方法，如 ngOnInit、ngOnDestroy 等。

```
@Component({
  selector: 'app-root'、
  templateUrl: './app.component.html'
  、
  styleUrls：['./app.component.css'] (
  简体中文)
})
export class AppComponent {
  title = 'DemoAngular'；
  ...
}
```

Angular 架构

模板是定义如何在屏幕上显示组件的 HTML 代码。

除了常规的 HTML 之外，该模板还包含指令、事件、插值、数据绑定和其他组件标记。

app.component.html

```
<h1 class="display-3">欢迎光临 Bike-Shop</h1>
    <p class="lead">在线大卖场</p>
    <hr class="my-2">
<h2>自行车</h2>。
<ul
  <li *ngFor="let bike of bikes" (click)="onSelect(bike)">
    {{自行车型号}}
  </li>
</ul>.
```


```
<app-bike-info [bike]="selectedBike"></app-bike-info>
```

项目配置

package.json - 项目配置文件

{

{ "dependencies"： {

运行应用程序所需的软件包（包括 Angular）

},

{ "devDependencies"： {

开发过程中使用的软件包（如编译器和测试框架）

},

{ "脚本": {

用于编译、测试和运行应用程序的命令

}

}

项目配置

package.json - 示例

```
{
"名称":
 "DemoTypeScript"，"脚本
 "：{
  { "start"："tsc -p tsconfig.json && node Ex2_rx.js".
 },
 { "dependencies"：{
  "rxjs"："~7.8.0",
  { "tslib": "^2.3.0"、
},
 { "devDependencies"：{
  "@types/node":"^12.11.1"、
  }, "ts-node": "~7.0.0"、
  }, "tslint"："~5.15.0",
```

```
  { "typescript"：  "~4.9.4"
 }
}
```

npm install -此命令下载 package.json 中指定的软件包，并将它们安装到 node_modules 文件夹中。

npm start - 从脚本部分启动命令

TypeScript

角度　　作为 框架　　已成为　　流行的　　选择

使用 TypeScript 编写的前端开发框架。

语法　　语言　有　　语法语言　相似性　　c　Java、

这简化了 Java 开发人员的工作。

语言特色

- 类型
- 班级
- 装饰者
- 进口
- 语言工具

TypeScript

ECMAScript (ES) 是 JavaScript 实现的一种标准规范。

ES5 是 "常规 "JavaScript；

ES6 试图解决 JavaScript 的不足之处，使其成为一门成熟的语言；

TypeScript 是一种开源解决方案，是 JavaScript 的超集。

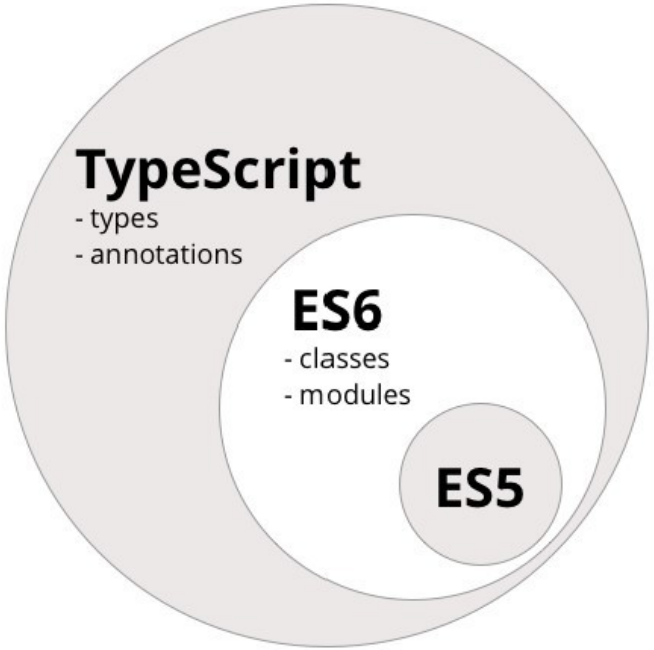
TypeScript 是 JavaScript 的类

型超集，可编译为

为纯 JavaScript

(https://www.typescriptlang.org/)

有一个将 TypeScript 转换为 ES5 的转



ES5, ES6, and TypeScript

换器。

TypeScript 与 Java 脚本

```
class Person {
 name: string;
 age: number;
 构造函数 (name: string, age: number) {
 this.name = name;
 this.age = age;
 }
 show() {
 console.log(`Hello ${this.name} is
${this.age} years old`) ;
 }
}
let p = new Person('bob',35);
p.show();
```

```
var Person = /** @class */ (function () {
    function Person(name, age) {
        this.name = name;
        this.age = age;
    }
    Person.prototype.show = function () {
        console.log("Hello " + this.name + "你好")
"是 " + this.age + " 岁") ;
    };
    返回人;
}());
var p = new Person('bob', 35);
p.show();
```

https://www.typescriptlang.org/#download-links

TSUN - TypeScript 升级版节点

TSUN（TypeScript 升级版节点）是 Node.js 的一个软件包。

支持交互式 stringREPL （读取 评估打印循环）和 TypeScript 解释器。

安装 TSUN： npm install -g tsun

执行 .ts 文件：

tsun <file_name.ts>

tsconfig.json

tsconfig.json 是 TypeScript 编译器的配置文件。它定义了如何将 TypeScript 文件转换（转译）为 Javascript。

```
{
 { "compilerOptions"： {
  "target"： "es5",
  "module"： {
  "commonjs"、
  }, "moduleResolution"： "node",
  "emitDecoratorMetadata": true,
  "experimentalDecorators": true
 },
 { "exclude"： [ "node_modules" ].
```

```
}
```

tsconfig.json

compilerOptions - 编译器设置

"**target**": "es5" - 编译器的 JavaScript 版本。编译器会将 TypeScript 转换为只使用指定版本功能的简单 JavaScript 代码。es5 值对应于大多数浏览器支持的 ES5 标准；

"**module**"：“commonjs"--JavaScript 模块创建格式；该值必须与项目中使用的加载器相匹配；

}, "**moduleResolution**"：“node" - 编译器处理导入命令的模式。如果值为 node，则会在 NPM 放置软件包的 node_modules 文件夹中搜索软件包；

"**emitDecoratorMetadata**": true - 如果为 true，编译器会包含装饰器信息，可以使用 reflect-matadata 包访问这些信息；

"**experimentalDecorators**": true

},

{ "**exclude**"： ["node_modules"- 参数告诉编译器要忽略哪些目录
}

TypeScript

TypeScript 是基于 于 原则 严格 数据类型

JavaScript

```
function sum(a, b) {
    return a + b;
}

sum( a: 1,  b: '2'); // результат "12"
```

TypeScript

```
698
699  function sum(a: number, b: number): number {
700      return a + b;
701  }
702
703  sum( a: 1,  b: '2');
```

5: Argument of type '"2"' is not assignable to parameter of type 'number'.

TypeScript

与 JavaScript 相比，TypeScript 的优势在于：提供更好的数据类型、代码结构和可读性，并向后兼容 JS。

缺点 - 类型只能在编译阶段进行控制，因此所有代码都会变成 JavaScript。

这意味着，如果从服务器传入的参数类型不正确，代码将不会报告。

TypeScript - 项目设置 全局安装

TypeScript 编译器：

```
npm install -g typescript
```

将 index.ts 文件中的 TypeScript 代码编译成 JavaScript：

```
TSC 指数
```

TypeScript 会向文本编辑器报告错误，但无论是否有错误，它都会编译代码。

创建 TS 配置文件：

```
tsc -init
```

命令执行后，tsconfig.json 文件将出现在项目根目录。

TypeScript - 数据类型

原始类型

字符串、数字、布尔、大整数、未定义、空、符号。

明确的注释，如

```
let firstname : string = 'Danny'
```

过分，因为　　类型。
变量（输出类型）：

```
let firstname = 'Danny'
```

TypeScript - 数据类型

动态类型：

any - 表示变量/参数可以是任何内容。

```
let age: any =100;
age = true;
```

工会类型

一个变量可以分配多个类型：

```
告诉 年龄: 数字       | 字符串;
年龄 = 22;
年龄 = '22';
```

TypeScript - 数据类型

字面意思:

```
let season: 'winter'| 'summer';

season = 'winter';
```

物品 - 公告:

```
let employee: object;
or

let employee: {
    firstName: string;
    lastName: string;

    age: number;
};
```

对象 - 创建:

```
雇员 = {
```

```
        名："约翰"，姓："无名
        氏"，年龄：25。
    };
```

TypeScript - 数据类型

数组

```
let skills: string[];
skills.push('Software Design');
```

元组 - 像一个数组、但元素数量固定 元素的数量是固定的，元素的类型是已知的：

```
let skill: [string, number];
```

Never 类型不包含值、

表示总是产生错误的函数或包含无限循环的函数的返回类型。

DOM 和类型转换

TypeScript 无法访问 DOM。这意味着在访问 DOM 元素时，TypeScript 无法确定它们是否存在。

使用非零验证操作符！可以告诉编译器表达式不等于 null 或未定义：

```
const link = document.querySelector('a')!
```

如果我们需要通过 DOM 元素的类或 ID 查找该元素，我们需要告诉 TypeScript 该元素存在，并且是 HTMLFormElement 类型。这可以通过类型转换（as 关键字）来实现：

```
const form =
document.getElementById('signup-form') 作为 HTMLFormElement;
```

TypeScript 中的函数

对于函数，必须指定函数参数的类型和函数返回值的类型。

```
function circle(diam: number): string {
        return 'Circle length: ' + Math.PI * diam;
}
```

同理，但使用的是 ES6 箭头功能：

```
const circle = (diam: number): string => {
        return 'Circle length: ' + Math.PI * diam;
}
```

您可以添加 ?使其成为可选项。