

# TypeScript教程

## 一、TypeScript介绍

---

### 1.1. 什么是TypeScript?

TypeScript 是 JavaScript 的一个超集，支持 ECMAScript 6 标准

TypeScript 由微软开发的自由和开源的编程语言。

TypeScript 设计目标是开发大型应用，它可以编译成纯 JavaScript，编译出来的 JavaScript 可以运行在任何浏览器上。

1. TypeScript 是由微软开发的一款开源的编程语言。
2. TypeScript 是 Javascript 的超集，遵循最新的 ES6、Es5 规范。TypeScript 扩展了 JavaScript 的语法。
3. TypeScript 更像后端 java、C#这样的面向对象语言，可以让 js 开发大型企业项目。
4. 谷歌也在大力支持 Typescript 的推广，谷歌的 angular2.x+就是基于 Typescript 语法。
5. 最新的 Vue 、React 也可以集成 TypeScript。
6. Nodejs 框架 Nestjs、midway 中用的就是 TypeScript 语法。

### 1.2 TypeScript语言特性

TypeScript 是一种给 JavaScript 添加特性的语言扩展。增加的功能包括：

- 类型批注和编译时类型检查
- 类型推断
- 类型擦除
- 接口
- 枚举
- Mixin
- 泛型编程
- 名字空间
- 元组
- Await

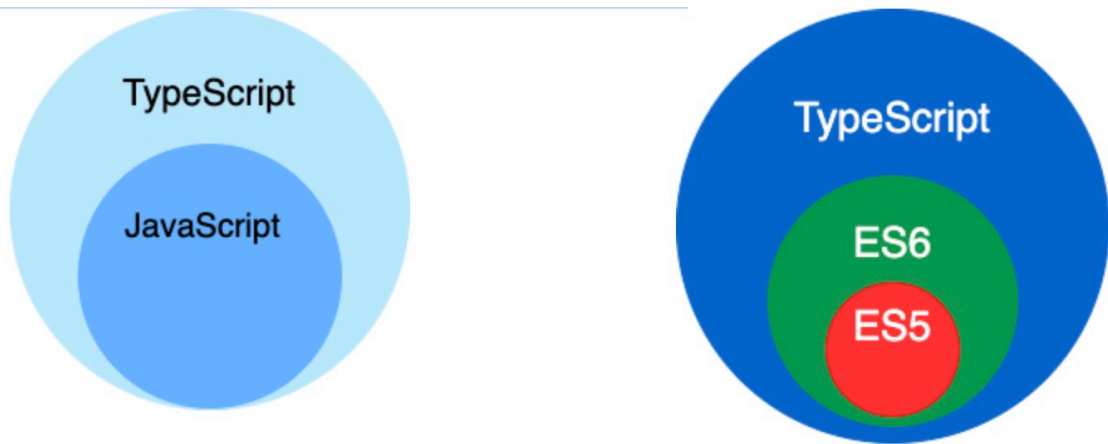
以下功能是从 ECMA 2015 反向移植而来：

- 类
- 模块
- lambda 函数的箭头语法
- 可选参数以及默认参数

## 1.3 JavaScript 与 TypeScript 的区别

TypeScript 是 JavaScript 的超集，扩展了 JavaScript 的语法，因此现有的 JavaScript 代码可与 TypeScript 一起工作无需任何修改，TypeScript 通过类型注解提供编译时的静态类型检查。

TypeScript 可处理已有的 JavaScript 代码，并只对其中的 TypeScript 代码进行编译。



## 二、Typescript 安装 编译

在使用 npm 命令之前电脑必须得安装 nodejs

### 2.1 安装

```
1  npm install -g typescript
2
3  或者
4
5  cnpm install -g typescript
6
7  或者
8
9  yarn global add typescript
```

### 2.2 运行

```
1  tsc helloworld.ts
```

- **注意：**如果电脑上面没有安装过 cnpm，请先安装 cnpm

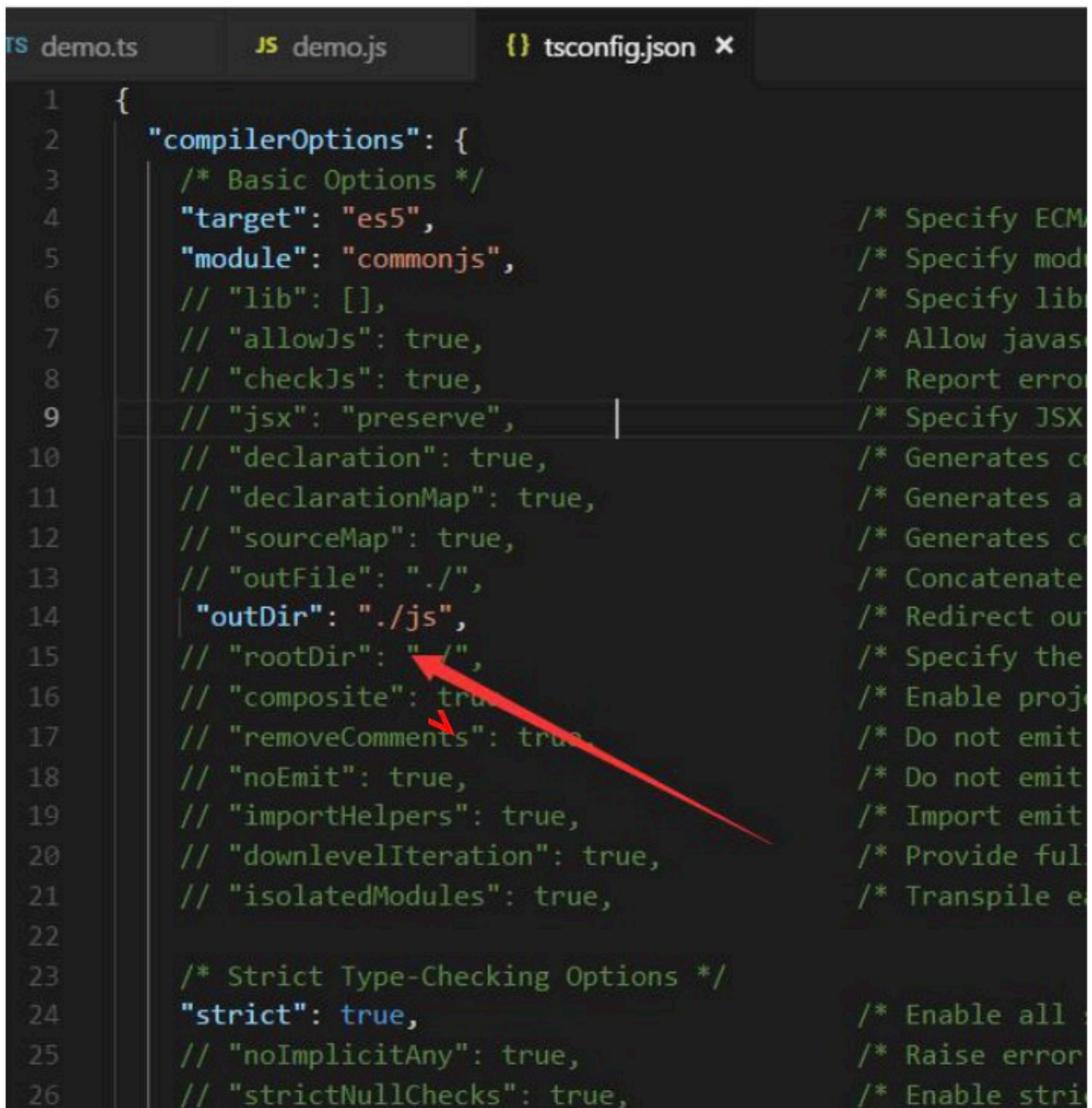
```
1  npm install -g cnpm --registry=https://registry.npm.taobao.org
```

- **注意：**如果电脑上面没有安装过 yarn 请先安装 yarn:

```
1 npm install -g yarn
2
3 或者
4
5 cnpm install -g yarn
```

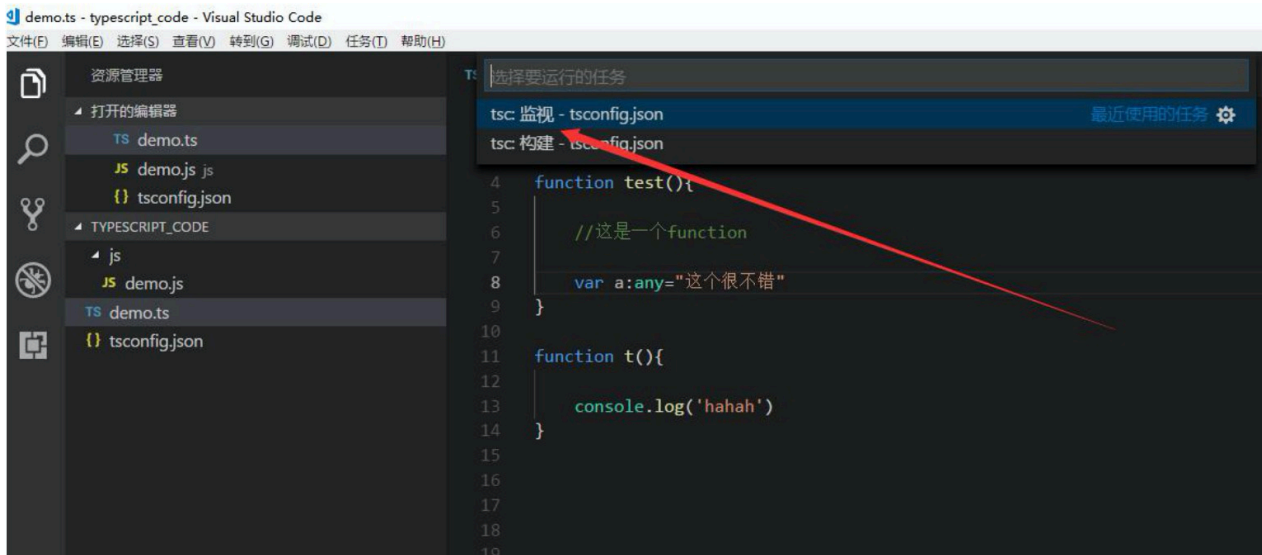
## 三、Typescript 开发工具 Vscode 自动编译.ts 文件

### 3.1 创建 tsconfig.json 文件 tsc --init 生成配置文件



```
1 {
2   "compilerOptions": {
3     /* Basic Options */
4     "target": "es5", /* Specify ECMAScript target version 'es5' | 'es6' | 'es2015' | 'es2016' | 'es2017' | 'esnext' */
5     "module": "commonjs", /* Specify module code generation 'none' | 'commonjs' | 'amd' | 'system' | 'umd' | 'es2015' | 'es2016' | 'esnext' */
6     // "lib": [], /* Specify library files to be included in the compilation */
7     // "allowJs": true, /* Allow javascript files to be compiled */
8     // "checkJs": true, /* Report errors in .js files */
9     // "jsx": "preserve", /* Specify JSX code generation: 'preserve', 'react-native', 'react' */
10    // "declaration": true, /* Generates .d.ts files */
11    // "declarationMap": true, /* Generates .d.ts.map files */
12    // "sourceMap": true, /* Generates source maps */
13    // "outFile": "./", /* Concatenate all compilation output into one file */
14    "outDir": "./js", /* Redirect output structure to the directory */
15    // "rootDir": ".", /* Specify the root directory of source files */
16    // "composite": true, /* Enable project compilation */
17    // "removeComments": true, /* Do not emit comments to output */
18    // "noEmit": true, /* Do not emit output */
19    // "importHelpers": true, /* Import emit helpers from the tslib */
20    // "downlevelIteration": true, /* Provide full support for iterables in 'for-of' loop and Object.entries() etc. */
21    // "isolatedModules": true, /* Transpile each file as a separate module (similar to 'ts.transpileModule') */
22
23    /* Strict Type-Checking Options */
24    "strict": true, /* Enable all strict type-checking options */
25    // "noImplicitAny": true, /* Raise error on expressions and declarations with an implied 'any' type */
26    // "strictNullChecks": true, /* Enable strict null checks */
27  }
28 }
```

3.2 老版本 vscode 点击: 任务->运行任务-> tsc:监视-tsconfig.json 然后就可以自动生成代码了



3.3、最新版本 vscode 点击: 终端->运行任务->typescript->tsc:监视-tsconfig.json 然后就可以自动生成代码了

## 四、TypeScript 基础语法

TypeScript 程序由以下几个部分组成:

- 模块
- 函数
- 变量
- 语句和表达式
- 注释

### 4.1 第一个 TypeScript 程序

我们可以使用以下 TypeScript 程序来输出 "Hello World" :

```
1  const hello : string = "Hello World!"
2  console.log(hello)
```

以上代码首先通过 **tsc** 命令编译:

```
1  tsc Runoob.ts
```

得到如下 js 代码:

```
1  var hello = "Hello World!";
2  console.log(hello);
```

最后我们使用 **node** 命令来执行该 js 代码。

```
1  $ node Runoob.js
2
3  Hello World
```

整个流程如下图所示：



我们可以同时编译多个 ts 文件：

```
1 tsc file1.ts file2.ts file3.ts
```

tsc 常用编译参数如下表所示：

序号	编译参数说明
1.	<b>--help</b> 显示帮助信息
2.	<b>--module</b> 载入扩展模块
3.	<b>--target</b> 设置 ECMA 版本
4.	<b>--declaration</b> 额外生成一个 .d.ts 扩展名的文件。 <div><pre>tsc ts-hw.ts --declaration</pre></div> 以上命令会生成 ts-hw.d.ts、ts-hw.js 两个文件。
5.	<b>--removeComments</b> 删除文件的注释
6.	<b>--out</b> 编译多个文件并合并到一个输出的文件
7.	<b>--sourcemap</b> 生成一个 sourcemap (.map) 文件。 sourcemap 是一个存储源代码与编译代码对应位置映射的信息文件。
8.	<b>--module noImplicitAny</b> 在表达式和声明上有隐含的 any 类型时报错
9.	<b>--watch</b> 在监视模式下运行编译器。会监视输出文件，在它们改变时重新编译。

## 4.2 TypeScript 保留关键字

TypeScript 保留关键字如下表所示：

break	as	catch	switch
case	if	throw	else
var	number	string	get
module	type	instanceof	typeof
public	private	enum	export
finally	for	while	void
null	super	this	new
in	return	true	false
any	extends	static	let
package	implements	interface	function
new	try	yield	const
continue	do		

## 4.3 空白和换行

TypeScript 会忽略程序中出现的空格、制表符和换行符。

空格、制表符通常用来缩进代码，使代码易于阅读和理解。

## 4.4 TypeScript 区分大小写

TypeScript 区分大写和小写字符

分号是可选的

每行指令都是一段语句，你可以使用分号或不使用，分号在 TypeScript 中是可选的，建议使用。

以下代码都是合法的：

```
1 console.log("Runoob")
2 console.log("Google");
```

如果语句写在同一行则一定需要使用分号来分隔，否则会报错，如：

```
1 console.log("Runoob");console.log("Google");
```

## 4.5 TypeScript 注释

注释是一个良好的习惯，虽然很多程序员讨厌注释，但还是建议你在每段代码写上文字说明。

注释可以提高程序的可读性。

注释可以包含有关程序一些信息，如代码的作者，有关函数的说明等。

编译器会忽略注释。

TypeScript 支持两种类型的注释

- **单行注释** (//) – 在 // 后面的文字都是注释内容。
- **多行注释** (/\* \*/) – 这种注释可以跨越多行。

```
1 // 这是一个单行注释
2
3 /*
4  这是一个多行注释
5  这是一个多行注释
6  这是一个多行注释
7  */
```