TypeScript 安装 TypeScript 基础类型

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

第一个 TypeScript 程序

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

实例

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

尝试一下 »

以上代码首先通过 **tsc** 命令编译：

```
tsc Test.ts
```

得到如下 js 代码：

```
var message = "Hello World";
console.log(message);
```

最后我们使用 node 命令来执行该 js 代码。

我们可以同时编译多个 ts 文件：

```
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.	**--help** 显示帮助信息
2.	**--module**

	载入扩展模块
3.	**--target** 设置 ECMA 版本
4.	**--declaration** 额外生成一个 .d.ts 扩展名的文件。 `tsc ts-hw.ts --declaration` 以上命令会生成 ts-hw.d.ts、ts-hw.js 两个文件。
5.	**--removeComments** 删除文件的注释
6.	**--out** 编译多个文件并合并到一个输出的文件
7.	**--sourcemap** 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.	**--module noImplicitAny** 在表达式和声明上有隐含的 any 类型时报错
9.	**--watch** 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。

TypeScript 保留关键字

TypeScript 保留关键字如下表所示：

break	as	any	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	catch	

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。

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：

```
console.log("Runoob")
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：

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

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。

注释实例：

```
// 这是一个单行注释

/*
 这是一个多行注释
 这是一个多行注释
 这是一个多行注释
*/
```

TypeScript 与面向对象

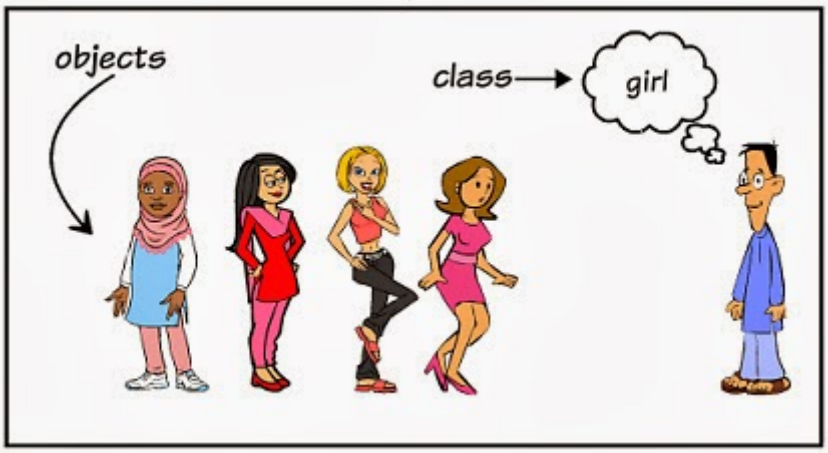
面向对象是一种对现实世界理解和抽象的方法。

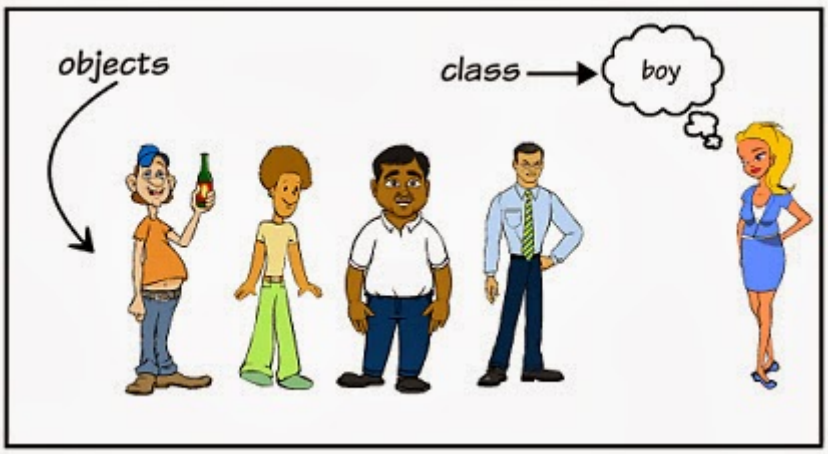
TypeScript 是一种面向对象的编程语言。

面向对象主要有两个概念：对象和类。

- **对象**：对象是类的一个实例（**对象不是找个女朋友**），有状态和行为。例如，一条狗是一个对象，它的状态有：颜色、名字、品种；行为有：摇尾巴、叫、吃等。
- **类**：类是一个模板，它描述一类对象的行为和状态。
- **方法**：方法是类的操作的实现步骤。

下图中男孩女孩为类，而具体的每个人为该类的对象：





TypeScript 面向对象编程实例：

```
class Site {
name():void {
console.log("Runoob")
}
}
var obj = new Site();
obj.name();
```

以上实例定义了一个类 Site，该类有一个方法 name()，该方法在终端上输出字符串 Runoob。

new 关键字创建类的对象，该对象调用方法 name()。

编译后生成的 JavaScript 代码如下：

```
var Site = /** @class */ (function () {
function Site() {
}
Site.prototype.name = function () {
console.log("Runoob");
};
return Site;
}());
var obj = new Site();
obj.name();
```

执行以上 JavaScript 代码，输出结果如下:

```
Runoob
```

← TypeScript 安装 TypeScript 基础类型 →

点我分享笔记