TypeScript 基础类型 TypeScript 运算符

TypeScript 变量声明

变量是一种使用方便的占位符，用于引用计算机内存地址。

我们可以把变量看做存储数据的容器。

TypeScript 变量的命名规则：

- 变量名称可以包含数字和字母。
- 除了下划线 `_` 和美元 `$` 符号外，不能包含其他特殊字符，包括空格。
- 变量名不能以数字开头。

变量使用前必须先声明，我们可以使用 var 来声明变量。

我们可以使用以下四种方式来声明变量：

声明变量的类型及初始值：

```
var [变量名] : [类型] = 值;
```

例如：

```
var uname:string = "Runoob";
```

声明变量的类型及但没有初始值，变量值会设置为 undefined：

```
var [变量名] : [类型];
```

例如：

```
var uname:string;
```

声明变量并初始值，但不设置类型类型，该变量可以是任意类型：

```
var [变量名] = 值;
```

例如：

```
var uname = "Runoob";
```

声明变量没有设置类型和初始值，类型可以是任意类型，默认初始值为 undefined：

```
var [变量名];
```

例如：

```
var uname;
```

实例

```
var uname:string = "Runoob";
var score1:number = 50;
var score2:number = 42.50
var sum = score1 + score2
console.log("名字: "+uname)
console.log("第一个科目成绩: "+score1)
console.log("第二个科目成绩: "+score2)
console.log("总成绩: "+sum)
```

注意：变量不要使用 name 否则会与 DOM 中的全局 window 对象下的 name 属性出现了重名。

使用 tsc 命令编译以上代码，得到如下 JavaScript 代码：

```
var uname = "Runoob";
var score1 = 50;
var score2 = 42.50;
var sum = score1 + score2;
console.log("名字: " + uname);
console.log("第一个科目成绩: " + score1);
console.log("第二个科目成绩: " + score2);
console.log("总成绩: " + sum);
```

执行该 JavaScript 代码输出结果为：

```
名字: Runoob
第一个科目成绩: 50
第二个科目成绩: 42.5
总成绩: 92.5
```

TypeScript 遵循强类型，如果将不同的类型赋值给变量会编译错误，如下实例：

```
var num:number = "hello"     // 这个代码会编译错误
```

类型断言（Type Assertion）

类型断言可以用来手动指定一个值的类型，即允许变量从一种类型更改为另一种类型。

语法格式：

```
<类型>值
```

或:

```
值 as 类型
```

实例

```
var str = '1'
var str2:number = <number> <any> str //str 现在是 number 类型
console.log(str2)
```

TypeScript 是怎么确定单个断言是否足够

当 S 类型是 T 类型的子集，或者 T 类型是 S 类型的子集时，S 能被成功断言成 S。这是为了在进行类型断言时提供额外的安全性，完全毫无根据的断言时危险的，如果你想这么做，你可以使用 any。

它之所以不被称为**类型转换**，是因为转换通常意味着某种运行时的支持。但是，类型断言纯粹是一个编译时语法，同时，它也是一种为编译器提供关于如何分析代码的方法。

编译后，以上代码会生成如下 JavaScript 代码：

```
var str = '1';
var str2 = str; //str 现在是 number 类型
console.log(str2);
```

执行输出结果为：

```
1
```

类型推断

当类型没有给出时，TypeScript 编译器利用类型推断来推断类型。

如果由于缺乏声明而不能推断出类型，那么它的类型被视作默认的动态 any 类型。

```
var num = 2; // 类型推断为 number
console.log("num 变量的值为 "+num);
num = "12"; // 编译错误
console.log(num);
```

- 第一行代码声明了变量 num 并=设置初始值为 2。 注意变量声明没有指定类型。因此，程序使用类型推断来确定变量的数据类型，第一次赋值为 2，**num** 设置为 number 类型。
- 第三行代码，当我们再次为变量设置字符串类型的值时，这时编译会错误。因为变量已经设置为了 number 类型。

```
error TS2322: Type '"12"' is not assignable to type 'number'.
```

变量作用域

变量作用域指定了变量定义的位置。

程序中变量的可用性由变量作用域决定。

TypeScript 有以下几种作用域：

- **全局作用域** – 全局变量定义在程序结构的外部，它可以在你代码的任何位置使用。
- **类作用域** – 这个变量也可以称为 **字段**。类变量声明在一个类里头，但在类的方法外面。 该变量可以通过类的对象来访问。类变量也可以是静态的，静态的变量可以通过类名直接访问。
- **局部作用域** – 局部变量，局部变量只能在声明它的一个代码块（如：方法）中使用。

以下实例说明了三种作用域的使用：

```
var global_num = 12 // 全局变量
class Numbers {
num_val = 13; // 类变量
static sval = 10; // 静态变量
storeNum():void {
var local_num = 14; // 局部变量
}
}
console.log("全局变量为: "+global_num)
console.log(Numbers.sval) // 静态变量
var obj = new Numbers();
console.log("类变量: "+obj.num_val)
```

以上代码使用 tsc 命令编译为 JavaScript 代码为：

```
var global_num = 12; // 全局变量
var Numbers = /** @class */ (function () {
function Numbers() {
this.num_val = 13; // 类变量
}
Numbers.prototype.storeNum = function () {
var local_num = 14; // 局部变量
};
Numbers.sval = 10; // 静态变量
return Numbers;
}());
console.log("全局变量为: " + global_num);
console.log(Numbers.sval); // 静态变量
var obj = new Numbers();
console.log("类变量: " + obj.num_val);
```

执行以上 JavaScript 代码，输出结果为：

```
全局变量为: 12
10
```

```
类变量: 13
```

如果我们在方法外部调用局部变量 local_num，会报错：

```
error TS2322: Could not find symbol 'local_num'.
```

← TypeScript 基础类型 TypeScript 运算符 →