TypeScript 函数 TypeScript String

TypeScript Number

TypeScript 与 JavaScript 类似，支持 Number 对象。

Number 对象是原始数值的包装对象。

语法

```
var num = new Number(value);
```

注意： 如果一个参数值不能转换为一个数字将返回 NaN (非数字值)。

Number 对象属性

下表列出了 Number 对象支持的属性：

序号	属性 & 描述
1.	**MAX_VALUE** 可表示的最大的数，MAX_VALUE 属性值接近于 1.79E+308。大于 MAX_VALUE 的值代表 "Infinity"。
2.	**MIN_VALUE** 可表示的最小的数，即最接近 0 的正数 (实际上不会变成 0)。最大的负数是 -MIN_VALUE，MIN_VALUE 的值约为 5e-324。小于 MIN_VALUE ("underflow values") 的值将会转换为 0。
3.	**NaN** 非数字值（Not-A-Number）。
4.	**NEGATIVE_INFINITY** 负无穷大，溢出时返回该值。该值小于 MIN_VALUE。
5.	**POSITIVE_INFINITY** 正无穷大，溢出时返回该值。该值大于 MAX_VALUE。
6.	**prototype** Number 对象的静态属性。使您有能力向对象添加属性和方法。
7.	**constructor** 返回对创建此对象的 Number 函数的引用。

TypeScript

```
console.log("TypeScript Number 属性: ");
console.log("最大值为: " + Number.MAX_VALUE);
console.log("最小值为: " + Number.MIN_VALUE);
```

```
console.log("负无穷大: " + Number.NEGATIVE_INFINITY);
console.log("正无穷大:" + Number.POSITIVE_INFINITY);
```

编译以上代码，得到以下 JavaScript 代码：

JavaScript

```
console.log("TypeScript Number 属性: ");
console.log("最大值为: " + Number.MAX_VALUE);
console.log("最小值为: " + Number.MIN_VALUE);
console.log("负无穷大: " + Number.NEGATIVE_INFINITY);
console.log("正无穷大:" + Number.POSITIVE_INFINITY);
```

输出结果为：

```
TypeScript Number 属性:
最大值为: 1.7976931348623157e+308
最小值为: 5e-324
负无穷大: -Infinity
正无穷大:Infinity
```

NaN 实例

TypeScript

```
var month = 0
if( month<=0 || month >12) {
month = Number.NaN
console.log("月份是: "+ month)
} else {
console.log("输入月份数值正确。")
}
```

编译以上代码，得到以下 JavaScript 代码：

JavaScript

```
var month = 0;
if (month <= 0 || month > 12) {
month = Number.NaN;
console.log("月份是: " + month);
}
else {
console.log("输入月份数值正确。");
}
```

输出结果为：

```
月份是: NaN
```

prototype 实例

TypeScript

```
function employee(id:number,name:string) {
this.id = id
this.name = name
}
var emp = new employee(123,"admin")
employee.prototype.email = "admin@runoob.com"
console.log("员工号: "+emp.id)
console.log("员工姓名: "+emp.name)
console.log("员工邮箱: "+emp.email)
```

编译以上代码，得到以下 JavaScript 代码：

JavaScript

```
function employee(id, name) {
this.id = id;
this.name = name;
}
var emp = new employee(123, "admin");
employee.prototype.email = "admin@runoob.com";
console.log("员工号: " + emp.id);
console.log("员工姓名: " + emp.name);
console.log("员工邮箱: " + emp.email);
```

输出结果为：

```
员工号: 123
员工姓名: admin
员工邮箱: admin@runoob.com
```

Number 对象方法

Number对象 支持以下方法：

序号	方法 & 描述	实例

1.	toExponential()
	把对象的值转换为指数计数法。
		`//toExponential()`
		`var num1 = 1225.30`
		`var val = num1.toExponential();`
		`console.log(val) // 输出: 1.2253e+3`

2.	toFixed()
	把数字转换为字符串，并对小数点指定位数。
		`var num3 = 177.234`
		`console.log("num3.toFixed() 为 "+num3.t`
		`oFixed())  // 输出: 177`
		`console.log("num3.toFixed(2) 为 "+num3.`
		`toFixed(2))  // 输出: 177.23`

		console.log("num3.toFixed(6) 为 "+num3.toFixed(6)) // 输出：177.234000
3.	toLocaleString() 把数字转换为字符串，使用本地数字格式顺序。	var num = new Number(177.1234); console.log(num.toLocaleString()); // 输出：177.1234

4.	toPrecision()
	把数字格式化为指定的长度。
		var num = new Number(7.123456);
		console.log(num.toPrecision()); // 输出：7.123456
		console.log(num.toPrecision(1)); // 输出：7
		console.log(num.toPrecision(2)); // 输出：7.1

5.	toString()
	把数字转换为字符串，使用指定的基数。数字的基数是 2 ~ 36 之间的整数。若省略该参数，则使用基数 10。
		var num = new Number(10);
		console.log(num.toString()); // 输出10进制：10
		console.log(num.toString(2)); // 输出2进制：1010
		console.log(num.toString(8)); // 输出8进制：12

6.	valueOf() 返回一个 Number 对象的原始数字值。	var num = new Number(10); console.log(num.valueOf()); // 输出：10

← TypeScript 函数 TypeScript String →