TypeScript 对象 TypeScript 模块

TypeScript 命名空间

命名空间一个最明确的目的就是解决重名问题。

假设这样一种情况，当一个班上有两个名叫小明的学生时，为了明确区分它们，我们在使用名字之外，不得不使用一些额外的信息，比如他们的姓（王小明，李小明），或者他们父母的名字等等。

命名空间定义了标识符的可见范围，一个标识符可在多个名字空间中定义，它在不同名字空间中的含义是互不相干的。这样，在一个新的名字空间中可定义任何标识符，它们不会与任何已有的标识符发生冲突，因为已有的定义都处于其他名字空间中。

TypeScript 中命名空间使用 `namespace` 来定义，语法格式如下：

```
namespace SomeNameSpaceName {
export interface ISomeInterfaceName { }
export class SomeClassName { }
}
```

以上定义了一个命名空间 SomeNameSpaceName，如果我们需要在外部可以调用 SomeNameSpaceName 中的类类和接口，则需要在类和接口添加 `export` 关键字。

要在另外一个命名空间调用语法格式为：

```
SomeNameSpaceName.SomeClassName;
```

如果一个命名空间在一个单独的 TypeScript 文件中，则应使用三斜杠 /// 引用它，语法格式如下：

```
/// <reference path = "SomeFileName.ts" />
```

以下实例演示了命名空间的使用，定义在不同文件中：

IShape.ts 文件代码：

```
namespace Drawing {
export interface IShape {
draw();
}
}
```

Circle.ts 文件代码：

```
/// <reference path = "IShape.ts" />
namespace Drawing {
export class Circle implements IShape {
public draw() {
console.log("Circle is drawn");
}
}
}
```

Triangle.ts 文件代码：

```
/// <reference path = "IShape.ts" />
namespace Drawing {
export class Triangle implements IShape {
public draw() {
console.log("Triangle is drawn");
}
}
}
```

TestShape.ts 文件代码：

```
/// <reference path = "IShape.ts" />
/// <reference path = "Circle.ts" />
/// <reference path = "Triangle.ts" />
function drawAllShapes(shape:Drawing.IShape) {
shape.draw();
}
drawAllShapes(new Drawing.Circle());
drawAllShapes(new Drawing.Triangle());
```

使用 tsc 命令编译以上代码：

```
tsc --out app.js TestShape.ts
```

得到以下 JavaScript 代码：

JavaScript

```
/// <reference path = "IShape.ts" />
var Drawing;
(function (Drawing) {
var Circle = /** @class */ (function () {
function Circle() {
}
Circle.prototype.draw = function () {
console.log("Circle is drawn");
};
return Circle;
}());
Drawing.Circle = Circle;
})(Drawing || (Drawing = {}));
/// <reference path = "IShape.ts" />
var Drawing;
(function (Drawing) {
var Triangle = /** @class */ (function () {
function Triangle() {
}
Triangle.prototype.draw = function () {
console.log("Triangle is drawn");
};
return Triangle;
```

```
}());
Drawing.Triangle = Triangle;
})(Drawing || (Drawing = {}));
/// <reference path = "IShape.ts" />
/// <reference path = "Circle.ts" />
/// <reference path = "Triangle.ts" />
function drawAllShapes(shape) {
shape.draw();
}
drawAllShapes(new Drawing.Circle());
drawAllShapes(new Drawing.Triangle());
```

使用 node 命令查看输出结果为：

```
$ node app.js
Circle is drawn
Triangle is drawn
```

嵌套命名空间

命名空间支持嵌套，即你可以将命名空间定义在另外一个命名空间里头。

```
namespace namespace_name1 {
export namespace namespace_name2 {
export class class_name { }
}
}
```

成员的访问使用点号 `.` 来实现，如下实例：

Invoice.ts 文件代码：

```
namespace Runoob {
export namespace invoiceApp {
export class Invoice {
public calculateDiscount(price: number) {
return price * .40;
}
}
}
}
```

InvoiceTest.ts 文件代码：

```
/// <reference path = "Invoice.ts" />
var invoice = new Runoob.invoiceApp.Invoice();
console.log(invoice.calculateDiscount(500));
```

使用 tsc 命令编译以上代码：

```
tsc --out app.js InvoiceTest.ts
```

得到以下 JavaScript 代码：

JavaScript

```
var Runoob;
(function (Runoob) {
var invoiceApp;
(function (invoiceApp) {
var Invoice = /** @class */ (function () {
function Invoice() {
}
Invoice.prototype.calculateDiscount = function (price) {
return price * .40;
};
return Invoice;
}());
invoiceApp.Invoice = Invoice;
})(invoiceApp = Runoob.invoiceApp || (Runoob.invoiceApp = {}));
})(Runoob || (Runoob = {}));
/// <reference path = "Invoice.ts" />
var invoice = new Runoob.invoiceApp.Invoice();
console.log(invoice.calculateDiscount(500));
```

使用 node 命令查看输出结果为：

```
$ node app.js
200
```

← TypeScript 对象 TypeScript 模块 →