

TypeScript namespace

1

- [TypeScript 2.0](#)
- [namespace 1](#)
- [namespace 2](#)

namespace `Utils` {

`function isString(value: any): boolean {`
 `return typeof value === 'string';`
`}` TypeScript `Utils` ES

`Utils`

namespace `Utils` {

```
namespace Utils {  
    function isString(value: any) {  
        return typeof value === 'string';  
    }  
  
    // 测试  
    isString('yes');  
}  
  
Utils.isString('no'); // 测试
```

`Utils` `Utils` `isString()` `Utils`

`export` `Utils`

```
namespace Utility {  
    export function log(msg: string) {  
        console.log(msg);  
    }  
}
```

```
Utility.log('Call me');
Utility.error('maybe!');
```

export

```
var Utility;  
  
(function (Utility) {  
    function log(msg) {  
        console.log(msg);  
    }  
  
    Utility.log = log;  
  
    function error(msg) {  
        console.error(msg);  
    }  
  
    Utility.error = error;  
  
})(Utility || (Utility = {}));
```

export

```
import
```

```
namespace App {  
    import isString = Utils.isString;
```

```
isString('yes');  
// 测试  
Utils.isString('yes');  
}
```

测试

import 测试

App 测试

Utils.isString 测试

isString 测试

import 测试

namespace 测试

```
namespace Shapes {  
  export namespace Polygons {  
    export class Triangle {}  
    export class Square {}  
  }  
}  
  
import polygons = Shapes.Polygons;  
  
// 测试 new Shapes.Polygons.Square()  
let sq = new polygons.Square();
```

测试

import 测试

Shapes 测试

Shapes.Polygons 测试

polygons 测试

namespace 测试

```
namespace Utils {  
  export namespace Messaging {  
    export function log(msg:string) {  
      console.log(msg);  
    }  
  }  
}  
  
Utils.Messaging.log('hello') // "hello"
```

测试

Utils 测试

Messaging 测试

Messaging 测试

测试

export 测试

测试

Utils.Messaging.log() 测试

namespace 测试

```
namespace N {
  export interface MyInterface{}
  export class MyClass{}
}
```

namespace

N namespace

namespace

namespace

namespace namespace JavaScript namespace

namespace namespace

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

namespace

namespace export

```
// shapes.ts
export namespace Shapes {
  export class Triangle {
    // ...
  }
  export class Square {
    // ...
  }
}
```

namespace

shapes.ts namespace

export namespace

Shapes namespace

namespace

import namespace

```
// shapes.ts
import { Shapes } from './shapes';
let t = new Shapes.Triangle();

// shapes.ts
import * as shapes from './shapes';
```

```
let t = new shapes.Shapes.Triangle();
```

```
// shapes.ts
export class Triangle {
    /* ... */
}

export class Square {
    /* ... */
}

// shapeConsumer.ts
import * as shapes from "./shapes";

let t = new shapes.Triangle();
```

namespace

--	--	--	--	--

namespace

interface

```
namespace Animals {
  export class Cat {}
}

namespace Animals {
  export interface Legged {
    numberOfLegs: number;
  }
  export class Dog {}
}

// 🐱🐶

namespace Animals {
  export interface Legged {
    numberOfLegs: number;
  }
  export class Cat {}
  export class Dog {}
}
```

```
// 🐱🐶
namespace Animals {
  export interface Legged {
    numberOfLegs: number;
  }
  export class Cat {}
  export class Dog {}
}
```

[illegible]

```
namespace N {
  const a = 0;

  export function foo() {
    console.log(a); // []
  }
}

namespace N {
  export function bar() {
    foo(); // []
    console.log(a); // []
  }
}
```

--	--	--	--	--	--	--	--

N

[illegible]

```
function f() {
  return f.version;
}

namespace f {
  export const version = '1.0';
}

f()    // '1.0'
f.version // '1.0'
```

--	--	--	--	--	--	--	--

[illegible][illegible]

class []

```

class C {
  foo = 1;
}

namespace C {
  export const bar = 2;
}

C.bar // 2

```

□□□□□□□□

C □□

C □□□□□□□□

bar □

□□□□□□□□

Enum □□□

```

enum E {
  A,
  B,
  C,
}

namespace E {
  export function foo() {
    console.log(E.C);
  }
}

E.foo() // 2

```

□□□□□□□□

E □□□

E □□□□□□

foo() □□□

□□□ Enum □□□□□□□□□□□□□□

```

enum E {
  A, // □□
  B,
}

namespace E {
  export function A() {} // □□
}

```

□□□□□□□□

Enum □□□□□□□□□□□□□□

□□ #2

□ □□ □□ 2023-07-09 12:07:22 UTC

□ □□ □□ 2023-08-14 14:01:49 UTC