

--	--

- | | |
|--|--|
| | |
|--|--|

```
export {};
```

```
import 
```

```
// a.ts
export interface A {
  foo: string;
}

export let a = 123;

// b.ts
import { A, a } from './a';
```

Diagram illustrating the layout of a JavaScript module file, showing various sections and their relative positions:

- `a.ts` (Starts at 0)
- `export` (Starts at 10)
- `A` (Starts at 20)
- `a` (Starts at 30)
- `b.ts` (Starts at 40)
- `TypeScript` (Starts at 50)
- `import` (Starts at 60)
- `type` (Starts at 70)

```
import { type A, a } from './a';
```

```

import A
import type

```

```
// ❶
import type { A } from './a';
let b:A = 'hello';

// ❷
import type { a } from './a';
let b = a;
```



Diagram illustrating the layout of a module object in memory. The module object is represented as a sequence of boxes, each containing a label and a size in bytes:

- import type (100 bytes)
- A (100 bytes)
- A (100 bytes)
- a (100 bytes)
- a (100 bytes)

```
import type DefaultType from 'moduleA';
```

```
import type [ ]
```

```
import type * as TypeNS from 'moduleA';
```

□□□□ export □□□□□□□□□□□□□□□□

```
type A = 'a';  
type B = 'b';  
  
// 型別  
export {type A, type B};  
  
// 型別  
export type {A, B};
```

The diagram illustrates the export of a type from a module. It consists of three parts:

- Module:** A box labeled "module" containing a "type" box and a "value" box.
- Export:** An arrow labeled "export type" points from the "type" box in the module to a "type" box in the "export type" box.
- Export Type:** A box labeled "export type" containing a "type" box and a "value" box.

```
class Point {
  x: number;
  y: number;
}

export type { Point };
```



 export type 
 Point 
 Point

```
import type { Point } from './module';

const p:Point = { x: 0, y: 0 };
```

[illegible]

importsNotUsedAsValues ☐ ☐ ☐ ☐

TypeScript █████ type█ import █████ JavaScript █████

[illegible]

1 remove import

```
2 preserve import
```

3 error import preserve import type

import

```
import { TypeA } from './a';
```

TypeA

remove

preserve

```
import './a';
```

preserve

import

a.js

a.js

a.js

error

preserve

import

import type

```
import type { TypeA } from './a';
```

CommonJS

CommonJS Node.js ES

import =

TypeScript import = CommonJS

```
import fs = require('fs');
const code = fs.readFileSync('hello.ts', 'utf8');
```

import =

require()

CommonJS

Node.js

import = TypeScript

import * as [] from " " CommonJS

```
import * as fs from 'fs';  
// 测试  
import fs = require('fs');
```

export =

TypeScript 中 `export =` 的用法

CommonJS 的 `require`

CommonJS 的 `module.exports`

```
let obj = { foo: 123 };  
  
export = obj;
```

`export =` 的用法

`import =` 的用法

```
import obj = require('./a');  
  
console.log(obj.foo); // 123
```



模块的 `module resolution` 规则

`import` 和 `export`

```
// 测试  
import { TypeA } from './a';  
  
// 测试  
import * as $ from "jquery";
```

在 TypeScript 中，`./a` 和 `jquery` 的 `module resolution` 规则

“

模块的 `moduleResolution` 规则在 Classic 和 Node 模式下

在 CommonJS 模式下，`moduleResolution` 规则与 Node 模式下的 `Node.js` 一致，支持 `es2015`、`esnext`、`amd`、`system`、`umd` 等模块格式。而在 Classic 模式下，则只支持 CommonJS 模块格式。

2. `node_modules` `package.json` `types`
`index.d.ts`
3. `node_modules` `@types` `b.d.ts`
4. `node_modules` `b` `index.ts` `index.tsx`
`index.d.ts`
5. `4`



TypeScript `tsconfig.json`

1 `baseUrl`

`baseUrl`

```
{
  "compilerOptions": {
    "baseUrl": "."
  }
}
```

`baseUrl` `tsconfig.json`

2 `paths`

`paths`

```
{
  "compilerOptions": {
    "baseUrl": ".",
    "paths": {
      "jquery": ["node_modules/jquery/dist/jquery"]
    }
  }
}
```

`jquery` `node_modules/jquery/dist/jquery`
`baseUrl`

`jquery`
`node_modules/jquery/dist/jquery`

3 `rootDirs`

rootDirs

```
{
  "compilerOptions": {
    "rootDirs": ["src/zh", "src/de", "src/#{locale}"]
  }
}
```

rootDirs

tsc --traceResolution

tsc --traceResolution

```
$ tsc --traceResolution
```

traceResolution

tsc --noResolve

tsc --noResolve

app.ts

```
import * as A from "moduleA";
import * as B from "moduleB";
```

```
$ tsc app.ts moduleA.ts --noResolve
```

--noResolve moduleA.ts moduleB

- [tsconfig](#) [importsNotUsedAsValues](#)

□□ #2

□ □□ □□ 2023-07-09 12:07:22 UTC

□ □□ □□ 2024-01-09 10:45:30 UTC