

MÓDULOS

MÓDULOS

Fundamentos TS

- ▶ Cada fichero es un módulo
- ▶ Un módulo **exporta** cosas y otros módulos **importan**

- ▶ Exportar con **export**

```
class Shape { ... }  
export class Circle extends Shape { ... }  
export PI: number = Math.PI;  
- or -  
export { Circle, PI as MyPI }
```

} module1.ts

- ▶ Importar con **import**

```
import { Circle } from "./module1";  
- or -  
import { Circle as MyCircle } from "./module1";  
- or -  
import * as module1 from "./module1";
```

} module2.ts

MÓDULOS

➤ JavaScript NO POSEE CARGADOR DE MÓDULOS

- ❑ En JavaScript **NO PODEMOS DESDE UN FICHERO cargar otro fichero** JavaScript
- ❑ Normalmente eso lo hace el navegador, desde un fichero html, por ejemplo, podemos cargar varios ficheros .js
 - ❑ Además, al hacer esto, todos los ficheros cargados, **COMPARTEN EL MISMO ESPACIO DE NOMBRES.**

➤ En sistemas MODULARES, **cada módulo tiene su propio espacio de nombres y sólo puede ver sus objetos**, salvo que importe otros que hayan sido exportados previamente.

➤ TYPESCRIPT sí tiene MÓDULOS

➤ **Todo lo que hay en un módulo TypeScript es privado a ese módulo y sólo visible a él.**

➤ **En ANGULAR lo usaremos constantemente**

MÓDULOS

- TypeScript admite MÓDULOS y JavaScript nativo NO.
 - TypeScript debe traducirse a JavaScript.
 - ¿CÓMO PUEDE HACERSE ESO? ¿Y LOS MÓDULOS?
-
- ❑ En JavaScript **existen librerías de terceros que permiten usar módulos.**
 - Una muy conocida es “**requireJS**”.
 - ❑ En realidad, hay dos estándares (que siguen todas esas librerías de terceros) para simular cargar módulos en JavaScript:
 - a) El estándar commonjs
 - *Es el que sigue node (el servidor)*
 - *Es síncrono: primero se importa y luego se trabaja con el módulo.*
 - b) El estándar amd
 - *Es asíncrono, basado en dependencias e inyección de módulos* (lo veremos en angular).
 - Nosotros decimos que tenemos dependencias de ciertos módulos y por medio de inyección se inyectan cuando hagan falta.

MÓDULOS

- **TYPESCRIPT** confía en estas librerías de terceros y delega en un sistema de carga de módulos que ya exista y que debe basarse en uno de esos estándares:

- **commonjs**


- **amd**

- A Typescript, cuando configuramos el compilador:

fichero: **tsconfig.json**

tenemos que indicarle con qué tipo de sistema de carga de módulos queremos que trabaje:

```
tsconfig.json  {  
                  "compilerOptions": {  
                    "target": "es5",  
                    "module": "commonjs", // "amd"  
                  }  
                }
```



Pero para que funcione, a parte de indicarlo ahí, deberíamos instalar esa librería de terceros **commonjs**, y configurarla bien.

EN ANGULAR NO HAY PROBLEMA, LO CONFIGURA ÉL AUTOMÁTICAMENTE.