

FUNDAMENTOS DE TYPESCRIPT



¿QUÉ ES?

- Es un lenguaje de programación libre y open source.
- Es un superset de Javascript.
- Utiliza el paradigma POO {Objetos y Clases}

¿QUIENES USAN TS?



y muchos otros!

Consulta: stackshare.com

¿QUÉ SE NECESITA?



Además del manejador de versiones de node: NVM

- Se instala con las instrucciones en github

VISUAL STUDIO ES GENIAL PARA TS

- Nos permite aprovechar al máximo TS

ESTRUCTURA

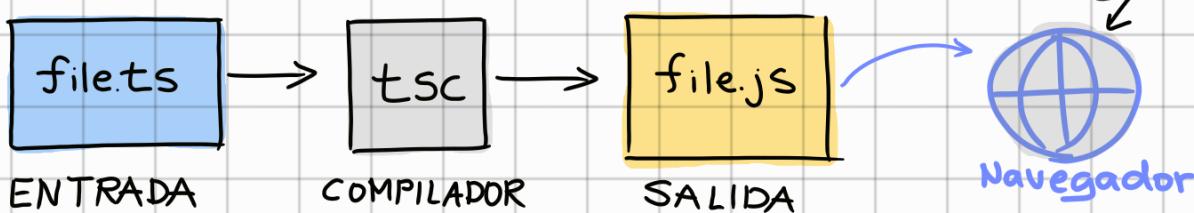


Tools

- Rename
- Quick Fix
- ESLint
- Snippets
- Docs
- Syntax Higlighter

COMPILADOR DE TS

tsc: typescript compiler



USO ⇒ \$ tsc <file>.ts
\$ node <file>.js
¡Ejecutar el resultado!

① Para una compilación automática:

\$ tsc --watch file.ts

CONFIGURACIÓN DE TS

{ } tsconfig.json

- Especifica el root del proyecto.
- Permite configurar opciones para el compilador.
- Se crea con: \$tsc --init

```
{
  "compilerOptions": {
    "module": "commonjs",
    "target": "es5", ← VERSIÓN DE ECS
    "pretty": true,
    "strict": true,
    "removeComments": true
  },
  "include": [
    "src/**/*.ts"
  ],
  "exclude": [
    "node_modules"
  ]
}
```

COMPILEADOR

¿QUÉ ARCHIVOS SE VAN A COMPIALAR?

¿CUALES SE VAN A EXCLUIR?

¿CÓMO SE GENERA UN PROYECTO TS?

1. Generar el tsconfig →
2. Configurar Salida ⇒ "outDir": "./dist"
3. Ejecutar solo \$tsc, esto compila todo.
4. Correr: \$node dist/file.js
5. Por default los archivos se compilan con root = src

TIPOS DE DATOS

→ Implícito

nombre = valor

"TS infiere el tipo"

→ Explícito

nombre : tipo = valor



El valor debe ser del tipo especificado!

TIPOS BÁSICOS

NUMBER

1, 1.0, 10

BOOLEAN

true, false

STRING

" ", ' ', ``

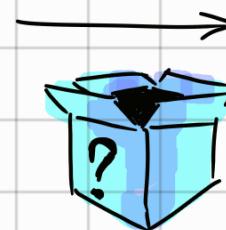
! El tipo de una variable no se puede cambiar.

VALORES Alfanuméricicos

HEX = 0xffff

BIN = 0b101

OCT = 0o756

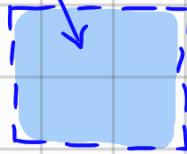


any: Puede ser cualquier cosa.

Útil para valores dinámicos.
Sin embargo, se debe evitar

Void: <<vacío>>

Lo usamos cuando una f(x) no retorna ningún valor.



never: Se usa en f(x) que manejan errores y que <<nunca>> retorne nada.
(throws) (∞ loops)

ERRORS

null: ← Son subtipos de any

Undefined: ↴

Valor: any = null // undefined

No hay nada

Nada definido pero se puede comportar como any.

Object: Tipo no primitivo

user: object; ; ObjectTS ≠ ObjectJS!

user = {

TS

JS

attr1: value,
attr2: value,
attr3: value

- Es un tipo

- Es una clase

3,

Array ⇒ []

→ Conjunto de Datos

→ 1 tipo → Type[]

→ N tipos → Array<type1...>

Generics

.length .push .slice [0]

Tuplas:

- Permiten expresar un arreglo con un número fijo de elementos
- Tipos de datos conocidos
User = [type1, type2];
- Pueden ser n-tuplas
- Lo ideal es que sean valores de tipos diferentes.

enum: Conjuntos que asocian un valor con una llave:

```
enum <Name> {  
    <name1>=value1, //0  
    <name2>=value2, //1  
    :  
}
```

```
var a = Name.name1;
```

TYPE UNION, ALIAS Y TIPOS LITERALES

- a) La unión de tipos: " | " ← pipe

```
Var a = type1 | type2  
↑ Podrá ser de alguno de los dos.
```



ASERCIIONES DE TIPO

@Similar al cast

```
<type>var => type  
var as type => type
```

- b) Alias le da un nombre a los tipos resultantes de la unión de tipos

```
type myType = type1 | type2  
↑ Alias!
```

- c) Es un arreglo de valores literales que solo permite que se le asigne un elemento de este arreglo.

```
Type Letras = "A" | "B" | "C";  
Let myVar: Letras = "A"; ✅ → OK  
myVar = "D"; X → Error <"D" ≠ Letras>
```

FUNCIONES

- Parámetros tipados.
- Pueden tener parámetros opcionales.
- Deben retornar un tipo:
void, primario o personalizado

```
1 function MediaPlayer(config) {  
2     this.media = config.el;  
3     this.plugins = config.plugins || [];  
4  
5     this._initPlugins();  
6 }  
7  
8 MediaPlayer.prototype._initPlugins = function() {  
9     this.plugins.forEach(plugin => {  
10         plugin.run(this);  
11     });  
12 }
```

Fat arrow f(x)

```
Let fun = () => {
```

```
    //body  
};
```

// TODO DEBE ESTAR TIPOADO!!

INTERFACES

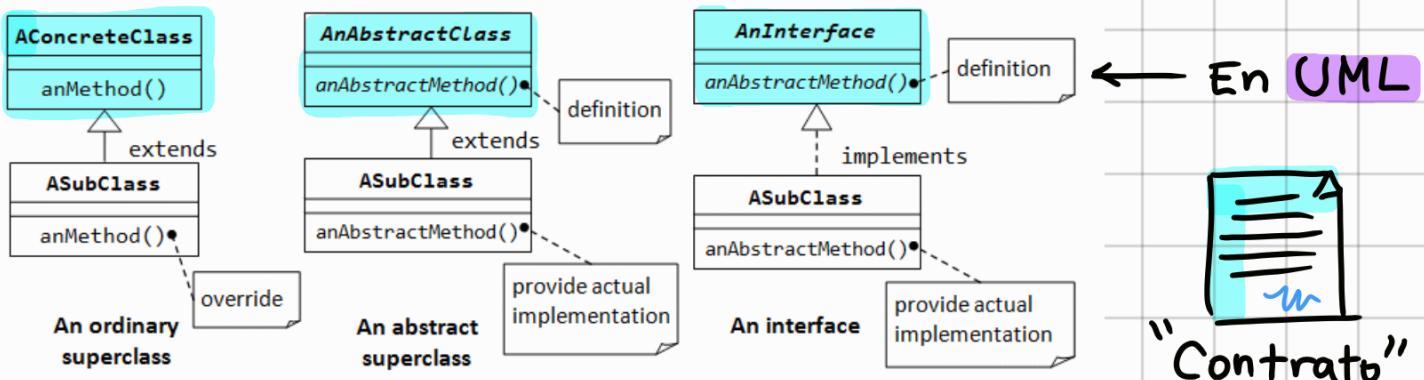
Moldes para nuestras clases. Indican qué atributos y métodos deben tener.

Cualquier clase que implemente una interfaz DEBE tener las propiedades y métodos.

```
interface Picture {  
    title: string,  
    date: string,  
    orientation: string,  
    imgUrl: string  
}  
applyFilter: (filter: number) => void
```

Propiedades
Método

Cada clase implementa el método como lo necesite sólo debe cumplir con tenerlo.



* Propiedades opcionales

Se agrega un "?" en el nombre de la propiedad

```
You, seconds ago | 1 author (You)  
interface PictureConfig {  
    title: string,  
    date?: string,  
    orientation: PhotoOrientation  
}
```

Propiedad Opcional!

Con el Keyword readonly no se podrá modificar el valor de una variable.

EXTENCIÓN DE INTERFACES

Herencia de Interfaces

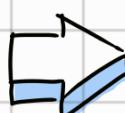
```
interface Person {  
    name: string;  
    lastname: string;  
}  
interface Student extends Person {  
    person: Person  
}
```

«Hereda»

⇒ Las propiedades de la Interfaz padre se heredan en la Interfaz hija.

CLASES

- Abstracción de un objeto.
- Tiene atributos y métodos.
- Tiene scope público ó privado.
- Heredan de otras class. «super()»
- Puede implementar interfaces.
- Ad.. más puede tener propiedades estáticas.



```
You, seconds ago | 1 author (You)  
class Picture {  
    private id: number;  
    private title: string;  
    orientation: PhotoOrientation;  
    constructor(id: number, title: string, orientation: PhotoOrientation) {  
        this.id = id;  
        this.title = title;  
        this.orientation = orientation;  
    }  
    getTitle(): string {  
        return this.title;  
    }  
}
```

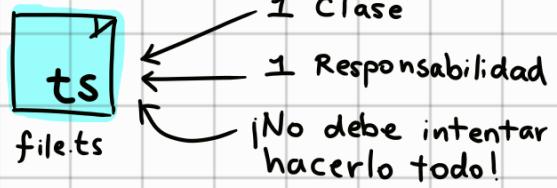
«Crea instancias de este Objeto!»

Método público!

MÓDULOS

- Permiten organizar el código.
- Los módulos se exportan e importan.
- Módulo = Herramienta.

SINGLE RESPONSABILITY



RESOLVER MÓDULOS

```
...  
Mod CommonJS, Más config  
$ tsc --moduleResolution node  
$ tsc --moduleResolution classic  
↑  
Módulos AMD, ES2015, poco config
```

Import Relativo

"./path/Module"

Import No-Relativo

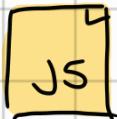
"Module"

WEBPACK

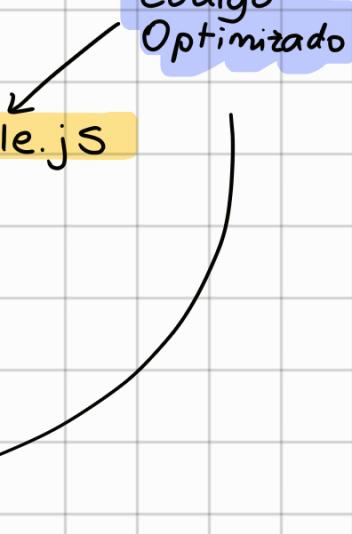
<< Empaquetador de Módulos >>

\$ npm init -y

\$ npm install webpack webpack-cli --save-dev



webpack.config.js \Rightarrow ./dist/bundle.js



¡Al Server!