

# Prog. Orientada a Objetos

## Carrera Programador full-stack

*TypeScript*

# Agenda

- Características del Lenguaje
- Instalación
- Traducción: TypeScript → JavaScript
- Tipado en las variables
- Restricciones impuestas en el tipado
- Demostración
  - Comparación código JS/TS
  - Restricciones de tipos
- Razones para usar TypeScript
- Diferencias TypeScript/JavaScript
- Recomendaciones

# Características de TypeScript

- Variables con tipos
  - Característica más importante (de ahí el nombre)
  - Código más legible/entendible
  - Más chequeos al momento de desarrollar → mayor seguridad
- Se “traduce” a código Javascript
  - Comando tsc
  - Es compatible con todas las librerías de JS
- Soporte de clases
  - Programación orientada a objetos (en la próxima clase)
- La forma en que se definen los if/for/while se mantiene

# Instalación

- NPM: `npm install -g typescript`
- Junto con el lenguaje, se instala el comando “tsc”
  - Se encarga de hacer la traducción del lenguaje TypeScript, al lenguaje JavaScript

```
PS C:\Users\Francisco\Documents\CFP\3. P00\Ejercicios> npm install -g typescript
C:\Users\Francisco\AppData\Roaming\npm\tsserver -> C:\Users\Francisco\AppData\Roaming\npm\node_modules\typescript\bin\tsserver
C:\Users\Francisco\AppData\Roaming\npm\tsc -> C:\Users\Francisco\AppData\Roaming\npm\node_modules\typescript\bin\tsc
+ typescript@3.5.2
added 1 package from 1 contributor in 0.663s
```

# Pasaje TypeScript → JavaScript

Código TypeScript  
*ejemplo.ts*



**tsc ejemplo.ts**



Código JavaScript  
*ejemplo.js*

- Tiene como entrada un archivo con extensión .ts
- Tiene como salida un archivo con extensión .js
- Ejemplo: `tsc ejemplo.ts` → en la misma carpeta va a generarse el archivo “ejemplo.js”
- Una vez generado el JS, se lo ejecuta como siempre → `node ejemplo.js`

# Pasaje TypeScript → JavaScript

```
function ejemplo(nombre) {  
    return "Hola " + nombre;  
}  
  
let nombre = 'Juan';  
  
console.log(ejemplo(nombre));
```

```
function ejemplo(nombre) {  
    return "Hola " + nombre;  
}  
  
let nombre = 'Juan';  
  
console.log(ejemplo(nombre));
```

**Código TypeScript**  
*ejemplo.ts*



**tsc ejemplo.ts**



**Código JavaScript**  
*ejemplo.js*

**¡El código es igual!**

# Variables con Tipos

```
function ejemplo(nombre: string): string {  
    return "Hola " + nombre;  
}
```

```
let nombre: string;
```

```
nombre = 'Juan';
```

```
console.log(ejemplo(nombre));
```

```
function ejemplo(nombre) {  
    return "Hola " + nombre;  
}
```

```
let nombre;
```

```
nombre = 'Juan';
```

```
console.log(ejemplo(nombre));
```

**Código TypeScript**  
*ejemplo.ts*



**tsc ejemplo.ts**



**Código JavaScript**  
*ejemplo.js*

**Los tipos se *eliminaron* → JS no tiene soporte de tipos**

# Restricción de Tipos en Variables (1)

```
function ejemplo(nombre: string): string {  
    return "Hola " + nombre;  
}
```

```
let nombre: string;  
nombre = 8;
```

```
console.log(ejemplo(nombre));
```

***Al querer compilar a JS nos va a dar error porque los tipos no son compatibles***



# Restricción de Tipos en Variables (2)

```
function ejemplo(nombre: string): string {  
    return "Hola " + nombre;  
}
```

```
let nombre: string;  
nombre = 8;
```

```
console.log(ejemplo(nombre));
```

```
PS C:\Users\Francisco\Documents\CFP\3. POO\Ejercicios> tsc ejemplo.ts  
ejemplo.ts:6:1 - error TS2322: Type '8' is not assignable to type 'string'.
```

```
6 nombre = 8;  
   ~~~~~
```

Found 1 error.

***A una variable de tipo “string” se le asignó un número***

# Restricción de Tipos en Funciones

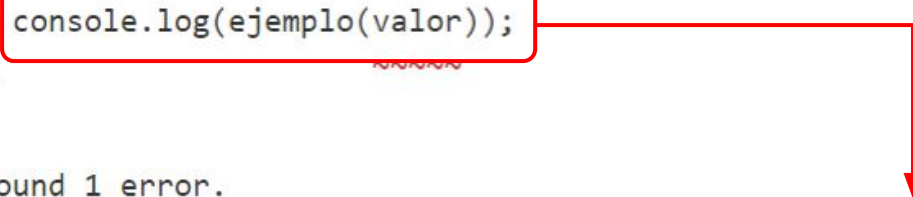
```
function ejemplo(nombre: string): string {  
    return "Hola " + nombre;  
}
```

```
let valor: number;  
valor = 8;
```

```
console.log(ejemplo(valor));
```

```
PS C:\Users\Francisco\Documents\CFP\3. POO\Ejercicios> tsc ejemplo.ts  
ejemplo.ts:8:21 - error TS2345: Argument of type 'number' is not assignable to parameter of type 'string'.  
1
```

```
8 console.log(ejemplo(valor));
```



```
Found 1 error.
```

***A una función con parámetro de tipo “string”  
se le pasó una variable numérica***

# Tipos básicos en TypeScript

Tipo	Significado
number	Cualquier tipo de número
boolean	Verdadero/falso
string	Texto
null	Cuando un elemento no tiene valores
undefined	Cuando una variable no está inicializada
any	Cualquier tipo
void	<i>Cuando las funciones no retornan nada</i>

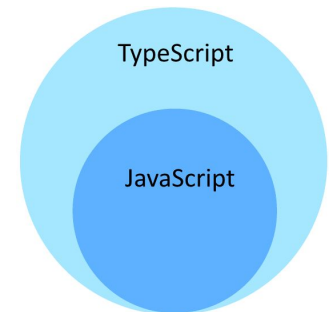
# Demostración

## Ejemplo Básico de Tipos

- Implementar una función que sume los elementos de un arreglo
- Forzar algún tipo de error a modo de ejemplificación

# Razones para usar TypeScript

- Muchos de los errores en JS surgen al momento de ejecutar el código
  - Por ejemplo llamar a una función con un parámetro de otro tipo
  - En proyectos grandes, este tipo de cuestiones hacen perder mucho tiempo
- El chequeo de tipos hace que gran parte de los errores que surjan, se puedan detectar al momento de escribir el código
  - Es decir antes de ejecutarlo
- Se trata de JavaScript pero con más funcionalidades



# Diferencias entre TypeScript y JavaScript

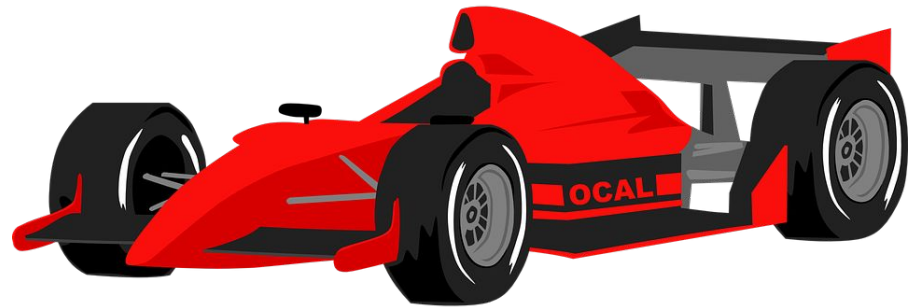
- TS tiene orientación a objetos, JS es de scripting
- Tipado estático vs. dinámico
- TS tiene soporte de interfaces
- TS tiene parámetros opcionales
- Cambia la forma en que se importan las librerías en el código
  - JS → `let readlineSync = require("readline-sync")`
  - TS → `import * as ReadlineSync from 'readline-sync';`

# Recomendaciones Generales

- Usar “camelCase” para definir funciones y variables
- Usar nombres descriptivos para las variables
- La estructura “for” permite declarar variables en el mismo lugar
- Tener en cuenta que el código que hagamos lo van a leer otras personas, o nosotros mismos dentro de varios meses o incluso años
  - Lo mejor es que el código sea fácil de leer
  - En caso de que igualmente sea complicado, usar comentarios para facilitar la lectura

# Muestra de código

- En una prueba, un piloto tiene que completar 4 vueltas
- Se necesita un programa que permita ingresar por teclado el tiempo de cada vuelta
- El programa debe retornar el tiempo total y el promedio de vuelta





TypeScript

**CFP**  
**Programador**  
**full-stack**

*Ejercicios*

# Ejercicios - En Clase

Definir funciones (con todos los tipos necesarios) para hacer lo siguiente:

- Cargar un listado de palabras (por esta vez, usar el arreglo como variable global)
- Insertar/eliminar/buscar/actualizar una palabra del listado

# Manejo de Archivos de Texto

- Instalar paquete → `npm install @types/node`
- Crear archivo 'abc.txt'
  - Escribir adentro 'hola como andas todo bien'


```
import * as fs from 'fs';
```

```
let texto: string = fs.readFileSync('abc.txt', 'utf8');
```

```
let palabras: string[] = texto.split(' ');
```

```
console.log(palabras);
```

**Nombre del archivo que  
vamos a leer**



```
PS C:\Users\Francisco\Documents\CFP\3. PO0\Ejercicios> tsc ejemplo-txt.ts
PS C:\Users\Francisco\Documents\CFP\3. PO0\Ejercicios> node ejemplo-txt.js
[ 'hola', 'como', 'andas', 'todo', 'bien' ]
```