



# TypeScript (parte 1)

LABORATORIO DE COMPUTACIÓN III

UTNFRA – TÉCNICO SUPERIOR EN PROGRAMACIÓN --

# Características faltantes de Javascript

- TIPOS DE VARIABLES
- ERRORES EN TIEMPO DE ESCRITURA
- AUTO COMPLETADO DEPENDIENDO DE LA VARIABLE
- MÉTODO ESTÁTICO DE PROGRAMACIÓN
- CLASES Y MÓDULOS (ANTES DE ES6)
- ENTRE OTRAS COSAS...

# Errores en Javascript

- Errores porque una variable no estaba definida
- Errores porque el objeto no tiene la propiedad esperada
- Errores porque no se tiene idea de como trabajan las funciones de otros
- Errores porque se sobre escriben variables, clases, funciones o constantes

# Tipos de Datos

## Primitivos

- ▶ String
- ▶ Números
- ▶ Booleanos
- ▶ Null/Undefined

## Compuestos

- ▶ Objetos Literales
- ▶ Clases
- ▶ Funciones

# String

“Juan Pérez”

‘UTN FRA’

`<h1> Hola Mundo </h1>`

# Números

PI = 3.14159265359

sueldo = 15000.00

entero = 1

# Booleanos

Verdadero (TRUE)

y

Falso (FALSE)

# Null y Undefined

numero = null

persona = undefined



# Objetos Literales

```
var persona = {  
    nombre:"Juan",  
    edad:3  
}
```

# Clases

```
class Persona{  
    nombre;  
    edad;  
}
```

# Funciones

```
function saludar(){  
    return "Hola";  
}
```



# También es posible:

Crear tipos nuevos

Interfaces

Tipos genéricos...

# Variables var y let

Se diferencian en el alcance de las variables:

- let delimita su alcance al bloque donde se ha declarado.

- var tiene un alcance global.

# Booleanos

```
let esSuperman:boolean = true;  
let esBatman:boolean;  
let esAquaman = true;
```



# Numbers

soporta tanto enteros como flotantes

```
let avengers:number = 5;
```

```
let villanos:number;
```

```
let otros = 2;
```

# Strings

```
let batman:string = "Batman;
```

```
let flash:string= 'Flash';
```

```
let linternaVerde = `Linterna Verde`;
```

(Backtick `)



# Any


```
let vengador:any = "Dr. Strange";  
vengador = 150; ✓
```

# Arrays


Son iguales que en Javascript pero le podemos definir que tipo de datos tienen.

```
let vector = [1, 2, 3, 4, 5, 6];
```


```
vector.push("123");
```




```
let vector:number = [1, 2, 3, 4, 5, 6];
```



```
let vector:number[] = [1, 2, 3, 4, 5, 6];
```



```
let villanos:String[]=["Lex Luthor", "Loki", "Duende Verde"];
```



# Tuplas

```
let hero: [string, number] = ["Dr.Strange", 100];
```

```
hero[0] = 123; ❌
```

```
hero[1] = "Ironman"; ❌
```

# Enumeraciones

```
enum Especialidades{  
    pediatra,  
    cardiologo,  
    clinico,  
}
```

# Funciones Básicas

```
let hero:string = "Flash";  
function imprime_heroe():string{  
    return hero;  
}  
let activar_batisenal 0 function():string{  
    return "Batiseñal activada";  
}  
console.log( imprime_heroe() );  
console.log( activar_batisenal() );
```

# Parámetros Obligatorios

```
function nombreCompleto(nombre:string, apellido:string):string{  
  
    return nombre + ' ' + apellido;  
}
```

# Parámetros Opcionales

```
function nombreCompleto(nombre:string, apellido?:string):string{  
  if(apellido)  
    return nombre + ' ' + apellido;  
  else  
    return nombre;  
}
```

# Parámetros por defecto

```
function nombreCompleto(nombre:string, apellido:string, capitalizado:boolean = false):string{
  var cadena:string;
  if(capitalizado)
    cadena = capitalizar(nombre) + " " + capitalizar(apellido);
  else
    cadena = nombre + ' ' + apellido;
  return cadena;
};

function capitalizar(cadena:string):string{
  return cadena.charAt(0).toUpperCase() + cadena.slice(1).toLowerCase();
};

console.log(nombreCompleto("tony", "stark", true));
```



# Parámetros REST

```
function nombreCompleto( nombre:string, ...losDemasParametros:string[] ):string{  
    return nombre + " " + losDemasParametros.join(" ");  
}  
  
let superman:string = nombreCompleto("Clark", "Joseph", "Kent");  
let ironman:string = nombreCompleto("Anthony", "Edward", "Tony", "Stark");  
  
console.log( superman );  
console.log( ironman );
```

# Tipo Función

```
function sumar(a:number, b:number):number{  
  return a+b;  
}
```

```
let miFuncion: (x:number, y:number)=>number;
```