TypeScript	**JavaScript**	**Consola**

var num: number = 12
console. log(num);
	"use strict";
	var num = 12;
	console.log(num);
		[LOG]: 12

class Greeting {
greet(): void {
console. log("Hello World!!!")
}
}
var obj = new Greeting ();
obj. greet();
	"use strict";
	class Greeting {
	greet() {
	console.log("Hello World!!!");
	}
	}
	var obj = new Greeting();
	obj.greet();
		[LOG]: "Hello World!!!"

var name: string = "John" ;
var score1: number = 50 ;
var score2: number = 42.50
var sum = score1 + score2
console. log("name" + name)
console. log("first score: " + score1)
console. log("second score: " + score2)
console. log("sum of the scores: " + sum)
//ERROR: name es una palabra reservada.
 	
//CORREGIDO 1
var name1: string = "John" ; // Cambiamos el nombre de la variable debiado a que tira error, al parecer es una palabra reservada.
var score1: number = 50 ;
var score2: number = 42.50
var sum = score1 + score2
console. log("name" + name1)
console. log("first score: " + score1)
console. log("second score: " + score2)
console. log("sum of the scores: " + sum)
	//ERROR
	Variable “name” ya fue declarada.
	
	"use strict";
	var name1 = "John";
	var score1 = 50;
	var score2 = 42.50;
	var sum = score1 + score2;
	console.log("name" + name1);
	console.log("first score: " + score1);
	console.log("second score: " + score2);
	console.log("sum of the scores: " + sum);
		[LOG]: "nameJohn"
		--- 	
		[LOG]: "first score: 50"
		--- 	
		[LOG]: "second score: 42.5"
		--- 	
		[LOG]: "sum of the scores: 92.5"

//CORREGIDO 2
export {};
var name: string = "John" ;
var score1: number = 50 ;
var score2: number = 42.50
var sum = score1 + score2
console. log("name" + name)
console. log("first score: " + score1)
console. log("second score: " + score2)
console. log("sum of the scores: " + sum)
 	
/* Para usar esta solución, en typescript se cambió el módulo de código generado de ESNext por CommonJS.*/
	"use strict";
	Object.defineProperty(exports, "__esModule", {
	value: true });
	var name = "John";
	var score1 = 50;
	var score2 = 42.50;
	var sum = score1 + score2;
	console.log("name" + name);
	console.log("first score: " + score1);
	console.log("second score: " + score2);
	console.log("sum of the scores: " + sum);	

var num: number = "hello" // will result in a compilation error //CORREGIDO var num: string = "hello"	Se produce un error de compilación al querer asignar un string a una variable tipo number "use strict"; var num = "hello";	Se produce un error de compilacion al querer asignar un string a una variable tipo number //SALIDA CORREGIDA -

var str = '1'
var str2: number = <number> <any> str //str is now of type number
console. log(str2)
	"use strict";
	var str = '1';
	var str2 = str;
	console.log(str2);
		[LOG]: "1"

<table>
<tr>
<td>

```
var num = 2 ; // data type inferred as
number
console. log( "value of num " + num);
num = "12" ;
console. log( num);


//CORREGIDO:
var num = 2 ; // data type inferred as
number
console. log( "value of num " + num);
num = Number("12") ;
console. log( num);
```

</td>
<td>

```
"use strict";
var num = 2; // data type inferred as number
console.log("value of num " + num);
num = "12";
console.log(num);



"use strict";
var num = 2; // data type inferred as number
console.log("value of num " + num);
num = Number("12");
console.log(num);
```

</td>
<td>

```
No se puede asignar un
tipo number a una
variable tipo string



//SALIDA (CORREGIDA)
[LOG]: "value of num 2"
---
[LOG]: 12
```

</td>
</tr>
<tr>
<td>

```
var global_num = 12 //global variable
class Numbers {
    num_val = 13 ; //class variable
    static sval = 10 ; //static field
    storeNum(): void {
    var local_num = 14 ; //local variable
    }
}
console. log( "Global num: " + global_num)
console. log( Numbers . sval) //static
variable
var obj = new Numbers ();
console. log( "Global num: " + obj. num_val)
```

</td>
<td>

```
"use strict";
var global_num = 12; //global variable
class Numbers {
    constructor() {
        this.num_val = 13; //class variable
    }
    storeNum() {
        var local_num = 14; //local variable
    }
}
Numbers.sval = 10; //static field
console.log("Global num: " + global_num);
console.log(Numbers.sval); //static variable
var obj = new Numbers();
console.log("Global num: " + obj.num_val);
```

</td>
<td>

```
[LOG]: "Global num: 12"
---
[LOG]: 10
---
[LOG]: "Global num: 13"
```

</td>
</tr>
<tr>
<td>

```
var num1: number = 10
var num2: number = 2
var res: number = 0
res = num1 + num2
```

</td>
<td>

```
var num1 = 10;
var num2 = 2;
var res = 0;
res = num1 + num2;
```

</td>
<td>

```
[LOG]: "Sum: 12"
---
[LOG]: "Difference: 8"
---
```

</td>
</tr>
</table>

console. log("Sum: " + res);
res = num1 - num2;
console. log("Difference: " + res)
res = num1* num2
console. log("Product: " + res)
res = num1/ num2
console. log("Quotient: " + res)
res = num1% num2
console. log("Remainder: " + res)
num1++
console. log("Value of num1 after increment " + num1)
num2--
console. log("Value of num2 after decrement " + num2)
	console.log("Sum: " + res);
	res = num1 - num2;
	console.log("Difference: " + res);
	res = num1 * num2;
	console.log("Product: " + res);
	res = num1 / num2;
	console.log("Quotient: " + res);
	res = num1 % num2;
	console.log("Remainder: " + res);
	num1++;
	console.log("Value of num1 after increment " + num1);
	num2--;
	console.log("Value of num2 after decrement " + num2);
		[LOG]: "Product: 20"
		--- 	
		[LOG]: "Quotient: 5"
		--- 	
		[LOG]: "Remainder: 0"
		--- 	
		[LOG]: "Value of num1 after increment 11"
		--- 	
		[LOG]: "Value of num2 after decrement 1"

var num1: number = 5 ;
var num2: number = 9 ;
console. log("Value of num1: " + num1);
console. log("Value of num2 :" + num2);
var res = num1> num2
console. log("num1 greater than num2: " + res)
res = num1< num2
console. log("num1 lesser than num2: " + res)
res = num1>= num2
console. log("num1 greater than or equal to num2: " + res)
	"use strict";
	var num1 = 5;
	var num2 = 9;
	console.log("Value of num1: " + num1);
	console.log("Value of num2 :" + num2);
	var res = num1 > num2;
	console.log("num1 greater than num2: " + res);
	res = num1 < num2;
	console.log("num1 lesser than num2: " + res);
	res = num1 >= num2;
	console.log("num1 greater than or equal to num2: " + res);
		[LOG]: "Value of num1: 5"
		--- 	
		[LOG]: "Value of num2 :9"
		--- 	
		[LOG]: "num1 greater than num2: false"
		--- 	
		[LOG]: "num1 lesser than num2: true"
		--- 	
		[LOG]: "num1 greater than or equal to num2: false"

var avg: number = 20 ;
var percentage: number = 90 ;
console. log("Value of avg: " + avg+ " ,value of percentage: " + percentage);
var res: boolean = ((avg> 50)&&(percentage> 80));
console. log("(avg>50)&&(percentage>80): " , res);
var res: boolean = ((avg> 50)||(percentage> 80));
console. log("(avg>50)||(percentage>80): " , res);
	"use strict";
	var avg = 20;
	var percentage = 90;
	console.log("Value of avg: " + avg + " ,value of percentage: " + percentage);
	var res = ((avg > 50) && (percentage > 80));
	console.log("(avg>50)&&(percentage>80): ", res);
	var res = ((avg > 50) || (percentage > 80));
	console.log("(avg>50)||(percentage>80): ", res);
	var res = !((avg > 50) && (percentage > 80));
	console.log("!((avg>50)&&(percentage>80)): ", res);
		[LOG]: "Value of avg: 20 ,value of percentage: 90"
		--- 	
		[LOG]: "(avg>50)&&(percentage>80): ", false
		--- 	
		[LOG]: "(avg>50)||(percentage>80): ", true
		--- 	
		[LOG]: "!((avg>50)&&(percentag

var res: boolean =!((avg> 50)&&(percentage> 80)); console. log("!((avg>50)&&(percentage>80)): " , res);		e>80)): ", true

var a: number = 2 ; // Bit presentation 10
var b: number = 3 ; // Bit presentation 11
var result;
result = (a & b);
console. log("(a & b) => " , result)
result = (a | b);
console. log("(a | b) => " , result)
result = (a ^ b);
console. log("(a ^ b) => " , result);
result = (~ b);
console. log("(~b) => " , result);
result = (a << b);
console. log("(a << b) => " , result);
result = (a >> b);
console. log("(a >> b) => " , result);
	"use strict";
	var a = 2; // Bit presentation 10
	var b = 3; // Bit presentation 11
	var result;
	result = (a & b);
	console.log("(a & b) => ", result);
	result = (a | b);
	console.log("(a | b) => ", result);
	result = (a ^ b);
	console.log("(a ^ b) => ", result);
	result = (~b);
	console.log("(~b) => ", result);
	result = (a << b);
	console.log("(a << b) => ", result);
	result = (a >> b);
	console.log("(a >> b) => ", result);
		[LOG]: "(a & b) => ", 2
		--- 	
		[LOG]: "(a | b) => ", 3
		--- 	
		[LOG]: "(a ^ b) => ", 1
		--- 	
		[LOG]: "(~b) => ", -4
		--- 	
		[LOG]: "(a << b) => ", 16
		--- 	
		[LOG]: "(a >> b) => ", 0

var num: number = - 2
var result = num > 0 ? "positive" :
"non-positive"
console. log(result)
	"use strict";
	var num = -2;
	var result = num > 0 ? "positive" :
	"non-positive";
	console.log(result);
		[LOG]: "non-positive"

DIFERENCIAS:

1) En TypeScript se especifica el tipo de dato (fuertemente tipado) -> te obliga a tipificar.

Ej TS:
var x: string = “Hola”

Ej js:
var x = “Hola”

2) En TypeScript se especifica el tipo de retorno de las funciones, por ejemplo, void (sin retorno), string, etc…, caso contrario en js donde lo mismo no es necesario.

class Greeting {
greet(): void {
console. log("Hello World!!!")
}
	"use strict";
	class Greeting {
	greet() {
	console.log("Hello World!!!");
	}

}
var obj = new Greeting ();
obj. greet();
	}
	var obj = new Greeting();
	obj.greet();

3) Casteos

En TS, los datos pueden ser tratados usando la siguiente sintaxis:

```
var markerSymbolInfo = <MarkerSymbolInfo> symbolInfo;
var markerSymbolInfo = symbolInfo as MarkerSymbolInfo;
```

En JS, se realiza la conversión del tipo de dato:
Ej:
var s = 1;

s= String(s);
console.log(s + " : " + typeof s); //outputs 1 : string

4) En JS, se definen los atributos de la clase por medio de un constructor de forma explícita.

Ej:

```
class Numbers {
    constructor() {
        this.num_val = 13; //class variable
    }
}
```

Ej: TS:

```
class Numbers {
    num_val = 13 ; //class variable
}
```

Además, en TS, los campos estáticos se pueden declarar dentro de la clase:

```
class Numbers {
    static sval = 10 ; //static field
}
console. log( Numbers . sval) //static variable
```

En cambio, en js:

```
"use strict";
class Numbers {
}
Numbers.sval = 10; //static field
console.log(Numbers.sval); //static variable
```

5) El código generado por ts contiene la etiqueta `"use strict"` que indica que el código debe ejecutarse en "modo estricto".

Con el modo estricto, no se pueden utilizar variables no declaradas.

6) TypeScript permite generar resultados débilmente tipados y entrega código validado (js).