



Introducción al lenguaje de programación

TypeScript

<https://www.typescriptlang.org/>



Juan Vladimir
@juanvladimir13

Agenda

1. Que es deno
2. Comandos de deno
3. Tipos de dato
4. Variables y constantes
5. Funciones
6. Parámetros y valor de retorno
7. Operadores aritméticos, lógicos y comparación
8. Estructuras de control
9. Operadores de asignación avanzado

Que es Deno

Is runtime for **TypeScript** and **JavaScript**. Features built-in dev tools, powerful platform APIs, and native support for TypeScript and JSX.



<https://www.typescriptlang.org/>

<https://developer.mozilla.org/es/docs/Web/JavaScript>

Crear archivo de texto plano TypeScript

Ingresa al **CMD** o **Terminator**

Crear un **archivo de texto plano** con la extensión ***.ts**

Ejecutar un archivo TypeScript

Verificar errores de sintaxis, ejecutar `deno lint nombreDeArchivo.ts`

Formatear el código, ejecutar el comando `deno fmt nombreDeArchivo.ts`

Ejecutar el código, ejecutar el comando `deno run nombreArchivo.ts`

Tipos de datos

Números enteros (positivos, negativos)

temperatura = -13

anio = 2025

Números con decimales (positivos, negativos)

peso = 60.48

latitud = -18.755121545

Texto / Cadenas / Caracteres / String

nombres = 'Juan Vladimir'

carnet = '12345678'

Booleanos

perteneceAlCurso = true

esMayorDeEdad = false

Variables, constantes y asignación

Variable

```
let edad = 36;           // variable de tipo de dato numero  
let nombre = 'juanvladimir13'; // variable de tipo de dato string  
let isDeveloper = false; // variable de tipo de dato boolean
```

Constante

```
const FECHA_NACIMIENTO = '26-12-1987';  
const CANTIDAD_BOMBAS_MAX = 20;
```

Asignación

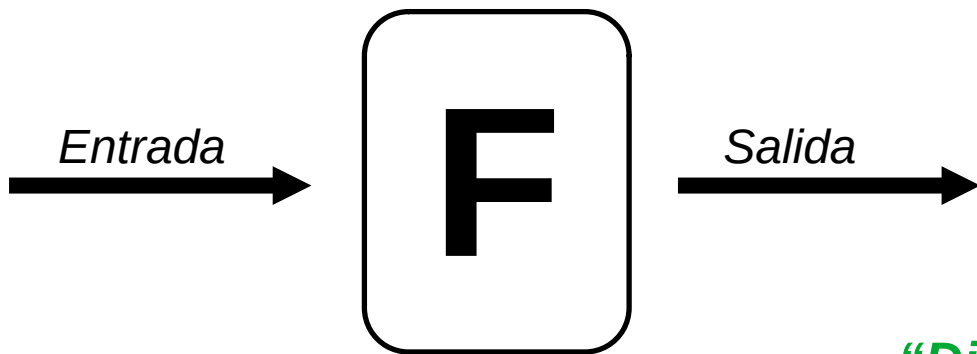
```
edad = 36;  
nombre = 'Ing. Juan Vladimir';  
isDeveloper = true;
```

Función

Es un **bloque de código** reutilizable que realiza una tarea específica.

Las funciones permiten **estructurar el código en partes más pequeñas y manejables**, lo que facilita su **mantenimiento y reutilización**.

Un **algoritmo** se resuelve con *una o muchas funciones*



“Divide y vencerás”

Estructura de una Función

Nombre: Este nombre debe ser descriptivo para indicar qué hace la función.

Parámetros: Son **cero, uno o más valores de entrada**, que la función utiliza para realizar su tarea.

Sentencias/bloque de código: Es el conjunto de instrucciones que se ejecutan cuando la función es llamada.

Valor de retorno: Valor como resultado de la tarea. Este valor puede ser utilizado por el código que llamó a la función.

Llamada a la función: Para usar una función, se "llama" por su **nombre** seguido de paréntesis. Si la función requiere parámetros, estos se colocan dentro de los paréntesis.

The diagram illustrates the structure of a function with the following components and labels:

- Nombre de la función:** `nombreAlfabetoIngles`
- Parámetros (0 o mas):**
 - `nick : string` is labeled with **nombre** (name) and **tipo de dato** (data type).
 - `edad : number` is labeled with **tipo de dato** (data type).
- Valor de retorno:** `: number`
- Bloque de código o Sentencias:** The code block inside the curly braces:

```
{  
  // sentencias  
  // sentencias  
  return 0;  
}
```


Ejercicio práctico

1. Abrir un editor de código de texto plano
2. Crear un archivo llamado **main.ts**
3. Escribir el siguiente contenido

```
function calcularEdadDeEstudiante( nombre:string, anioNac:number ):number {  
    // Cuerpo (sentencias)  
    return 0;  
}  
const edad = calcularEdadDeEstudiante( 'juanvladimir', 2005 );  
console.log(edad);
```

4. Guardar el contenido del archivo
5. Abrir el **CMD** en la carpeta donde se encuentra el archivo **main.ts**
6. Escribir el siguiente comando: **deno run main.ts**

Parámetros y valor de retorno

Números

```
function sumar(numX: number, numY: number): number {  
    return 0;  
}
```

Cadenas

```
function enviarSaludo(usuario: string): string {  
    return 'Hola';  
}
```

Booleanos

```
function registrarUsuario(persona: string): boolean {  
    return false;  
}
```

Tipo de dato de retorno void

```
function destruirEnemigo(cantidadBombas: number): void {  
}
```

Operadores Aritméticos

let sueldo : **number** = 3200;

Suma	+	sueldo = sueldo + 5;
Resta	-	sueldo = sueldo - 5;
Multiplicación	*	sueldo = sueldo * 5;
División	/	sueldo = sueldo / 3;
Residuo	%	sueldo = sueldo % 3;
Exponente	**	sueldo = sueldo ** 3;
Incremento	++	sueldo++ ;
Decremento	--	sueldo--;

Operadores Lógicos

Conjunción: AND &&

Disyunción: OR ||

Negación: NOT !

let seComportanBien : **boolean** = true;

let hacenTareas : **boolean** = true;

let hayChurrasco : **boolean** = false;

hayChurrasco = seComportanBien && hacenTareas;

hayChurrasco = seComportanBien || hacenTareas;

hayChurrasco = !hayChurrasco;

Operadores de comparación

```
let sueldoGerente : number = 3200;  
let sueldoEmpleado : number = 5500;
```

Igual	==	sueldoGerente == sueldoEmpleado;
Diferente	!=	sueldoGerente != sueldoEmpleado;
Igual estricto	===	sueldoGerente === sueldoEmpleado;
Diferente estricto	!==	sueldoGerente !== sueldoEmpleado;
Mayor que	>	sueldoGerente > sueldoEmpleado;
Menor que	<	sueldoGerente < sueldoEmpleado;
Mayor igual	>=	sueldoGerente >= sueldoEmpleado;
Menor igual	<=	sueldoGerente <= sueldoEmpleado;

El resultado de una **comparación** siempre es una **proposición** *false* o *true*

Expresión

VAR : variable

FUN : función

OPE : operador/conector

CON : constante

Expresión unaria

OPE { **VAR** | **FUN** | **CON** }

Ejemplos:

a = ~hayChurrasco;

res = calcularSuma(8, 5);

curso = 'Quinto D';

Expresiones

{ **VAR** | **CON** | **FUN** } **OPE** { **VAR** | **CON** | **FUN** }

Ejemplos:

x = 8 + 5;

y = sueldoGerente > sueldoEmpleado;

z = calcularSuma(5,5) * 0.3;

a = haceTareas && esLinda;

b = haceTareas && (sueldo > 3000);

c = (sueldo + 1500) - gastos;

Sentencia

Es una **línea de código** que llevan a cabo una tarea. Las sentencias están formadas por un **conjunto de expresiones** que permiten realizar una acción determinada.

```
let seComportanBien : boolean = true;  
let hacenTareas : boolean = false;  
let hayChurrasco : boolean = false;  
let sueldo : number = 3200;  
let sueldoGerente : number = 5000;
```

```
hayChurrasco = seComportanBien && hacenTareas;  
y = sueldoGerente > sueldoEmpleado;  
z = calcularSuma(5,5) * 0.3;
```

Estructuras de control

Estructura de control condicional (if else)

Ejecuta una **sentencia si una condición específica es evaluada como verdadera**.

Si la condición es evaluada como **falsa**, otra sentencia puede ser ejecutada.

```
let sueldo : number = 3200;
```

```
let hacenTareas : boolean = true;
```

Sintaxis

```
if ( sueldo >= 3200 ) {  
    // sentencias o bloque de código  
}
```

```
if ( hacenTareas ) {  
    // sentencias o bloque de código  
}
```

```
if ( sueldo > 3200 && hacenTareas ) {  
    // bloque de código por true  
} else {  
    // bloque de código por false  
}
```

```
if ( ( sueldo + 1000 ) > 3200 ) {  
    // sentencias o bloque de código  
}
```


Estructura de control condicional

Ejecuta un **bloque de código o sentencias** si una **condición** específica **es evaluada como verdadera**

```
let condicional = true;
```

Variable de tipo boolean

```
const estudioEnCasa = false;  
condicional = !estudioEnCasa;
```

*Variable de tipo boolean
Aplicando la **negación** en su resultado*

```
const notaTrimestral = 51;  
condicional = notaTrimestral > 50;
```

Operador de comparación

```
const edad = 16;  
condicional = !(edad > 18)
```

*Operador de comparación
Aplicando la **negación** en su resultado*

Variable de tipo boolean

```
if (condicional) {  
  console.log("Hay premios");  
  console.log("Hay juegos");  
}
```

*Bloque de código
o
Sentencias*

```
let condicional = true;
```

```
const realizoTareas = true; const obedezcoEnTodo = true; const edad = 17;
```

Variables de tipo boolean

```
condicional = realizoTareas && obedezcoEnTodo;
```

Variable boolean

Operador de comparación

```
condicional = realizoTareas && edad > 15;
```

```
condicional = ( realizoTareas && obedezcoEnTodo ) && edad > 15;
```

Asociación de variables booleanas

```
condicional = ( edad > 15 || obedezcoEnTodo ) && realizoTareas
```

Asociación de un operador de comparación y una variable booleana

```
condicional = ( realizoTareas && obedezcoEnTodo ) || ( edad > 16 && edad < 19 );
```

Asociación de variables booleanas

Asociación de operadores de comparación

Estructuras de control

Estructura de control repetitiva (while)

Ejecuta una **sentencia especificada mientras cierta condición se evalúe como verdadera**. Dicha condición es evaluada antes de ejecutar la sentencia.

Sintaxis

```
let n: number = 0;
while ( n > 13 ) {
  // sentencias
  $n++;
}
```

```
const estaLloviendo: boolean = true;
while ( estaLloviendo ) {
  // sentencias
}
```

Estructura de control repetitiva

Ejecuta un **bloque de código** mientras cierta **condición** se evalúe como verdadera

```
let edad = 15;  
    Variable de control  
while (edad < 18) {  
    // sentencias  
    edad++;  
    // sentencias  
}
```

Sentencia que modifica la variable de control

```
let porcentajeDeAvance = 1;  
let proyectoFinalizado = true;  
    Variable de control  
while (proyectoFinalizado) {  
    // sentencias  
    if (porcentajeDeAvance > 100) {  
        proyectoFinalizado = false;  
    }  
    porcentajeDeAvance++;  
    // sentencias  
}
```

Estructura de control que modifica la variable de control

Estructuras de control

Estructura de control repetitiva (for)

Expresión inicial: Declaración de variable y asignación

Condición: Expresión para ser evaluada antes de cada iteración del bucle. Si esta expresión se evalúa como verdadera, se ejecuta sentencia, si es falsa, la ejecución salta a la siguiente expresión.

Expresión final: Expresión evaluada al final de cada iteración del bucle. Esto ocurre antes de la siguiente evaluación de la condición. Generalmente se usa para actualizar o incrementar la variable contador.

Sintaxis

```
for (let index=0; index < 13; index++) {  
    // sentencias  
}
```

Operadores de Asignación Avanzados

```
let sueldo : number = 3200;  
let hayChurrasco : boolean = false;
```

Suma	+=	sueldo += 5;
Resta	-=	sueldo -= 5;
Multiplicación	*=	sueldo *= 5;
División	/=	sueldo /= 3;
Residuo	%=	sueldo %= 3;
Exponente	**=	sueldo **= 3;
AND	&&=	hayChurrasco &&= sueldo > 3000;
OR	=	hayChurrasco = sueldo > 3000;