## **Funciones Básicas en TypeScript**

En TypeScript, las funciones son bloques de código reutilizables que realizan una tarea específica. Nos permiten modularizar nuestro código, hacerlo más legible y mantenerlo organizado.

**Estructura básica de una función en TypeScript:**

TypeScript

function nombreDeLaFuncion(parametro1: tipo1, parametro2: tipo2, ...): tipoDeRetorno { // Cuerpo de la función // Aquí se ejecuta el código return valorDeRetorno; }

Use code [with caution.](file:///C:/faq#coding)

* **nombreDeLaFuncion:** El nombre que le damos a la función para identificarla.
* **parametro1, parametro2:** Los valores que recibe la función para realizar su tarea. Cada parámetro tiene un tipo asociado.
* **tipoDeRetorno:** El tipo de dato que devuelve la función. Si no devuelve nada, se omite.
* **Cuerpo de la función:** El conjunto de instrucciones que se ejecutan cuando se llama a la función.

## **Ejemplos con Explicaciones Detalladas**

### **Ejemplo 1: Sumar dos números**

TypeScript

// SumaDosNumeros.tsfunction sumar(a: number, b: number): number { // Esta función suma dos números y devuelve el resultado. // Los parámetros 'a' y 'b' son de tipo número. // El tipo de retorno también es número. return a + b; }

Use code [with caution.](file:///C:/faq#coding)

### **Ejemplo 2: Saludar a una persona**

TypeScript

// SaludarPersona.tsfunction saludar(nombre: string): string { // Esta función crea un saludo personalizado. // Recibe un nombre como parámetro de tipo cadena. // Devuelve un saludo con el nombre de la persona. return `¡Hola, ${nombre}!`; }

Use code [with caution.](file:///C:/faq#coding)

### **Ejemplo 3: Verificar si un número es par**

TypeScript

// EsPar.tsfunction esPar(numero: number): boolean { // Esta función determina si un número es par. // Recibe un número como parámetro. // Devuelve un valor booleano (true si es par, false si no). return numero % 2 === 0; }

Use code [with caution.](file:///C:/faq#coding)

### **Ejemplo 4: Calcular el factorial de un número**

TypeScript

// Factorial.tsfunction factorial(n: number): number { // Esta función calcula el factorial de un número. // Recibe un número entero positivo. // Devuelve el factorial de ese número. if (n === 0) { return 1; } else { return n \* factorial(n - 1); } }

Use code [with caution.](file:///C:/faq#coding)

### **Ejemplo 5: Función flecha (arrow function)**

TypeScript

// MultiplicarPorDos.tsconst multiplicarPorDos = (numero: number): number => numero \* 2; // Esta es una forma más concisa de escribir una función.// Se utiliza mucho en TypeScript.

Use code [with caution.](file:///C:/faq#coding)

## **¿Qué más puedes hacer con las funciones en TypeScript?**

* **Funciones con parámetros opcionales:** Puedes definir parámetros que sean opcionales.
* **Funciones con parámetros por defecto:** Puedes asignar valores por defecto a los parámetros.
* **Funciones con sobrecarga:** Puedes crear múltiples definiciones de una función con diferentes tipos de parámetros.
* **Funciones anónimas:** Puedes crear funciones sin nombre y asignarlas a variables.
* **Funciones de flecha:** Son una sintaxis más concisa para escribir funciones.

**En resumen,** las funciones son una herramienta fundamental en TypeScript. Te permiten organizar tu código, reutilizarlo y crear aplicaciones más robustas y mantenibles.