## **Funciones Anónimas en TypeScript**

**¿Qué son?**

En TypeScript, una función anónima es una función que no tiene un nombre asociado. Se define directamente en el lugar donde se utiliza, a menudo como un valor asignado a una variable o pasada como argumento a otra función. Esto las hace muy útiles para crear funciones de un solo uso o para realizar operaciones cortas de forma concisa.

**Sintaxis básica:**

TypeScript

const miFuncionAnonima = (parametro1: tipo1, parametro2: tipo2): tipoRetorno => { // Cuerpo de la función return resultado; };

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

**Ejemplos prácticos:**

### **Ejemplo 1: Sumar dos números (sumarNumeros.ts)**

TypeScript

// Función anónima que suma dos númerosconst sumar = (a: number, b: number): number => { // Descripción: Esta función toma dos números como entrada y devuelve su suma. // Explicación: Utilizamos la sintaxis de flecha para definir una función anónima. Los parámetros 'a' y 'b' son de tipo número. El cuerpo de la función simplemente devuelve la suma de 'a' y 'b'. return a + b; }; // Uso de la funciónconst resultado = sumar(5, 3); console.log(resultado); // Imprime: 8

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

### **Ejemplo 2: Filtrar un array (filtrarElementos.ts)**

TypeScript

// Función anónima que filtra un array de números para obtener solo los paresconst filtrarPares = (numeros: number[]): number[] => { // Descripción: Esta función toma un array de números como entrada y devuelve un nuevo array con solo los números pares. // Explicación: Utilizamos el método 'filter' de los arrays para crear un nuevo array. La función de flecha dentro de 'filter' verifica si cada elemento es divisible por 2. Si lo es, se incluye en el nuevo array. return numeros.filter(numero => numero % 2 === 0); }; // Uso de la funciónconst numeros = [1, 2, 3, 4, 5]; const pares = filtrarPares(numeros); console.log(pares); // Imprime: [2, 4]

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

### **Ejemplo 3: Transformar un array (transformarArray.ts)**

TypeScript

// Función anónima que transforma un array de números en un nuevo array con cada número elevado al cuadradoconst elevarAlCuadrado = (numeros: number[]): number[] => { // Descripción: Esta función toma un array de números como entrada y devuelve un nuevo array con cada número elevado al cuadrado. // Explicación: Utilizamos el método 'map' de los arrays para crear un nuevo array. La función de flecha dentro de 'map' eleva al cuadrado cada elemento del array original. return numeros.map(numero => numero \* numero); }; // Uso de la funciónconst numeros = [1, 2, 3]; const cuadrados = elevarAlCuadrado(numeros); console.log(cuadrados); // Imprime: [1, 4, 9]

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

### **Ejemplo 4: Crear un objeto (crearObjeto.ts)**

TypeScript

// Función anónima que crea un objeto con propiedades basadas en un valor de entradaconst crearPersona = (nombre: string): { nombre: string, edad: number } => { // Descripción: Esta función toma un nombre como entrada y devuelve un objeto con las propiedades 'nombre' y 'edad'. La edad se inicializa en 30. // Explicación: Creamos un objeto con las propiedades deseadas y lo retornamos. La edad se asigna a un valor fijo en este ejemplo, pero podría calcularse de otra manera. return { nombre, edad: 30 }; }; // Uso de la funciónconst persona = crearPersona("Juan"); console.log(persona); // Imprime: { nombre: 'Juan', edad: 30 }

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

**¿Por qué usar funciones anónimas?**

* **Concisión:** Permiten escribir código más conciso, especialmente cuando se necesitan funciones simples.
* **Flexibilidad:** Se pueden utilizar como valores, pasarse como argumentos y asignarse a variables.
* **Funciones de callback:** Son ideales para funciones de callback en métodos como map, filter, reduce, etc.

**En resumen,** las funciones anónimas son una herramienta poderosa en TypeScript que te permite crear funciones de manera flexible y concisa. Al comprender su sintaxis y uso, podrás escribir código más limpio y eficiente.