

DISEÑO DE SITIOS WEB DINÁMICOS

FUNCIONES



TEMA 07:

Declaración de funciones tradicionales.
Funciones anónimas. Funciones flecha.

DOCENTE:

MIRIAM HUACCHA HERRERA

2025-II

PROGRAMA DE ESTUDIOS:

II DESARROLLO DE SISTEMAS DE INFORMACIÓN

Funciones declaradas

Se definen con la palabra clave `function`, seguida del nombre de la función, paréntesis para los parámetros y llaves para el cuerpo de la función.

- **Características:** Pueden ser invocadas antes de su declaración en el código.

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

Funciones expresadas (anónimas)

Una función sin nombre asignada a una variable.

Características: No se pueden invocar antes de su declaración. Son útiles para callbacks o funciones que se usarán una sola vez.

```
const multiplicar = function(a, b) {  
  return a * b;  
};
```

Funciones flecha (Arrow Functions)

Una sintaxis más concisa para escribir funciones, que utiliza =>.

Características:

- Si hay un solo parámetro, los paréntesis alrededor de los parámetros son opcionales.
- Si la función tiene una sola expresión y se omiten las llaves {}, el valor se retorna implícitamente.
- A diferencia de las funciones tradicionales, las arrow functions no tienen su propio this.

Con retorno implícito: `const sumarFlecha = (a, b) => a + b;`

Con múltiples líneas: `const saludar = (nombre) => { return
 Hola, ${nombre}; };`

Parámetros y valores de retorno

Son variables listadas dentro de los paréntesis () en la definición de la función. Permiten pasar valores a la función.

Valores de retorno

Se utilizan para devolver un valor desde la función.

En funciones declaradas y expresadas, se usa la palabra clave return

En funciones flecha de una sola línea, el return es implícito. Si se usan llaves, es necesario usar return explícitamente.

```
// `edad` es un parámetro  
function saludar(nombre, edad) {  
    console.log(`Hola, ${nombre}. Tienes ${edad} años.`);  
}  
  
// `nombre` y `edad` son argumentos  
saludar("Ana", 30); // Salida: Hola, Ana. Tienes 30 años.
```

```
// `total` es el valor de retorno
function sumar(a, b) {
  let total = a + b;
  return total;
}

let resultado = sumar(5, 3); // resultado ahora contiene 8 [8]
console.log(resultado); // Salida: 8

function obtenerDatos() {
  let nombre = "Carlos";
  let edad = 25;
  // Para devolver múltiples valores, los agrupamos en un objeto
  return { nombre: nombre, edad: edad };
}

let datosUsuario = obtenerDatos();
console.log(datosUsuario.nombre); // Salida: Carlos
```