COLOMBIA POTENCIA DE LA VIDA

TIC

TALENTO TECH

DESARROLLO *WEB*

FULL STACK

NIVEL BÁSICO

Funciones y Alcance en JavaScript

cymetria | tecnalia colombia

Introducción a las Funciones en JavaScript

Las funciones son bloques de código reutilizables que realizan una tarea específica. Permiten estructurar mejor el código y evitar repeticiones.

¿Por qué usar funciones?

✅ Mejoran la organización del código
✅ Fomentan la reutilización de código
✅ Hacen que el código sea más modular y fácil de mantener

Tipos de Funciones en JavaScript

a) Funciones Declaradas (Function Declaration)

Estas funciones se definen con la palabra clave `function` y pueden llamarse antes de su declaración gracias al "hoisting".

📌 **Ejemplo:**

javascript Copy Edit

```javascript
function saludar() {
    console.log("¡Hola, bienvenido al bootcamp!");
}

saludar(); // Llamada a la función
```

Tipos de Funciones en JavaScript

b) Funciones Expresadas (Function Expression)

Son funciones asignadas a variables. No pueden llamarse antes de su declaración.

📌 **Ejemplo:**

javascript Copy Edit

```javascript
const despedida = function() {
    console.log("¡Hasta luego!");
};

despedida(); // Llamada a la función
```

c) Funciones Flecha (Arrow Functions)

Introducidas en ES6, son una forma más corta de escribir funciones.

📌 **Ejemplo:**

javascript Copy Edit

```javascript
const sumar = (a, b) => a + b;

console.log(sumar(5, 3)); // 8
```

Parámetros y Argumentos

- **Parámetros:** Variables que recibe una función.
- **Argumentos:** Valores que se pasan cuando se llama a la función.

📌 **Ejemplo:**

javascript Copy Edit

```javascript
function multiplicar(x, y) {
    return x * y;
}

console.log(multiplicar(4, 5)); // 20
```

Parámetros y Argumentos

Parámetros por defecto

Se pueden definir valores por defecto para los parámetros.

📌 **Ejemplo:**

javascript Copy Edit

```javascript
function saludar(nombre = "Visitante") {
    console.log(`Hola, ${nombre}`);
}

saludar(); // Hola, Visitante
saludar("Carlos"); // Hola, Carlos
```

Ámbito o Alcance de Variables (Scope)

El alcance define dónde se puede acceder a una variable dentro del código.

a) Alcance Global

Las variables declaradas fuera de cualquier función son accesibles en todo el código.

📌 **Ejemplo:**

javascript Copy Edit

```javascript
let globalVar = "Soy global";

function mostrarGlobal() {
    console.log(globalVar);
}

mostrarGlobal(); // "Soy global"
```

Ámbito o Alcance de Variables (Scope)

b) Alcance Local

Las variables declaradas dentro de una función solo son accesibles dentro de ella.

📌 **Ejemplo:**

javascript Copy Edit

```javascript
function funcionEjemplo() {
    let localVar = "Soy local";
    console.log(localVar);
}

funcionEjemplo(); // "Soy local"
// console.log(localVar); // ❌ ERROR: No está definida fuera de la función
```

Ámbito o Alcance de Variables (Scope)

c) Alcance de Bloque (`let` y `const`)

Las variables declaradas con `let` o `const` dentro de un bloque (`{}`) solo existen dentro de ese bloque.

📌 **Ejemplo:**

javascript Copy Edit

```javascript
if (true) {
    let bloqueVar = "Solo existo aquí";
    console.log(bloqueVar);
}

// console.log(bloqueVar); // ❌ ERROR: No está definida fuera del bloque
```

Diferentes formas de llamar una función en JavaScript y HTML

1 Llamar una función directamente en JavaScript

La forma más sencilla de ejecutar una función es llamarla directamente en el código de JavaScript.

📌 **Ejemplo:**

javascript Copy Edit

```javascript
function saludo() {
    console.log("¡Hola, mundo!");
}
saludo(); // Llamada directa
```

Diferentes formas de llamar una función en JavaScript y HTML

2 Llamar una función desde el HTML con `onclick`

Podemos asociar una función a un evento de un elemento HTML, como un botón, usando el atributo `onclick`.

📌 Ejemplo en HTML:

html Copy Edit

```html
<button onclick="saludar()">Haz clic</button>
```

📌 Código en JavaScript:

javascript Copy Edit

```javascript
function saludar() {
    alert("¡Hola desde el botón!");
}
```

Diferentes formas de llamar una función en JavaScript y HTML

3 Llamar una función usando `addEventListener`

En lugar de definir la función en el HTML, es más recomendable usar `addEventListener()` en JavaScript.

📌 **Ejemplo:**

html

Copy Edit

```html
<button id="btnSaludo">Saludar</button>
```

javascript

Copy Edit

```javascript
document.getElementById("btnSaludo").addEventListener("click", function() {
    alert("¡Hola desde JavaScript!");
});
```

Diferentes formas de llamar una función en JavaScript y HTML

4 Llamar una función anónima (sin nombre)

Podemos definir una función dentro de `addEventListener` sin darle un nombre.

📌 **Ejemplo:**

javascript Copy Edit

```javascript
document.getElementById("btnSaludo").addEventListener("click", function() {
    console.log("Hola desde una función anónima");
});
```

5 Llamar una función con parámetros

Si queremos pasar valores a la función al hacer clic en un botón, podemos hacerlo de la siguiente manera:

📌 **Ejemplo en HTML:**

html Copy Edit

```html
<button onclick="mostrarMensaje('Bienvenido al Bootcamp!')">Mostrar Mensaje</button>
```

📌 **Código en JavaScript:**

javascript Copy Edit

```javascript
function mostrarMensaje(mensaje) {
    alert(mensaje);
}
```

Diferentes formas de llamar una función en JavaScript y HTML

6 Llamar una función con `setTimeout()`

Podemos ejecutar una función después de un tiempo determinado.

📌 **Ejemplo:**

javascript Copy Edit

```javascript
function mostrarAlerta() {
    alert("Esta alerta aparece después de 3 segundos");
}
setTimeout(mostrarAlerta, 3000); // Se ejecuta después de 3 segundos
```

7 Llamar una función con `setInterval()`

Ejecuta una función repetidamente cada cierto intervalo de tiempo.

📌 **Ejemplo:**

javascript Copy Edit

```javascript
function mostrarHora() {
    console.log(new Date().toLocaleTimeString());
}
setInterval(mostrarHora, 2000); // Se ejecuta cada 2 segundos
```

EJEMPLOS

EJEMPLO NUMERO 1

Vamos a crear un código en el que tendremos un input numérico, donde el usuario podrá ingresar un valor, y un menú desplegable (select) con diferentes opciones predefinidas.
Al hacer clic en el botón "Consultar", el sistema tomará el número ingresado y lo sumará con el valor correspondiente a la opción seleccionada.
El resultado de la operación se mostrará en pantalla de manera dinámica. Para ello, usaremos JavaScript, capturando los valores ingresados y realizando la suma con una función que se encargará de devolver el valor correcto según la selección del usuario. Además, aplicaremos estilos con CSS para que la interfaz sea más atractiva y ordenada.

EJEMPLO NUMERO 1

Vamos a crear un código en el que tendremos:

Funcionalidad:
1.El usuario ingresa un número en el **input**.
2.Al presionar el botón **"Iniciar Cuenta Regresiva"**, comienza la cuenta regresiva.
3.El número disminuirá cada segundo hasta llegar a **0**.
4.El resultado se actualizará dinámicamente en pantalla.

Diferentes formas de llamar una función en JavaScript y HTML

🔷 Actividad 1: Calculadora de Propinas 💰

Objetivo:

Los estudiantes deben crear una calculadora de propinas donde el usuario ingrese el monto de la cuenta y seleccione el porcentaje de propina. Se utilizarán funciones **declaradas y expresadas**, con y sin parámetros, llamadas desde el DOM y JavaScript.

🔷 Requisitos:

- Usar una **función declarada** para calcular la propina.
- Usar una **función expresada** para mostrar el resultado en la interfaz.
- Llamar las funciones desde un **botón HTML (onclick)** y desde **addEventListener()** en JS.
- Practicar el **alcance de variables** (global y local).

📝 Instrucciones:

1. Crea un archivo **HTML** con un input para el monto, un select para el porcentaje de propina y un botón para calcular.
2. En JavaScript, define una **función declarada** para calcular la propina.
3. Usa una **función expresada** para mostrar el resultado en pantalla.
4. Llama a las funciones desde el **DOM usando eventos.**

Actividad 2 Simulador de Pedidos de Comida

💡 **Descripción:** Los estudiantes simularán un sistema de pedidos de comida rápida. Los usuarios eligen un menú y el sistema muestra un **mensaje de preparación con** `setTimeout` y una **cuenta regresiva con** `setInterval` hasta que la comida esté lista. Además, se aplicarán funciones con parámetros y retornos para modular el código.

📌 ¿Qué Aprenden los Estudiantes?

- ✅ **Funciones con parámetros y retornos** para modular el código.
- ✅ `setTimeout` y `setInterval` para simular la preparación del pedido.
- ✅ **Interactividad realista**, ya que cada comida tiene un tiempo de espera diferente.
- ✅ **Dinamismo**, los estudiantes pueden jugar con diferentes tiempos de preparación.