INSTITUTO KHIPU

Semestre IV

Sesión 08

PROGRAMA DE ESTUDIOS

DESARROLLO DE SISTEMAS DE INFORMACIÓN

COMUNITY MANAGER Y MARKETING DIGITAL

Tema

FUNCIONES EN JAVASCRIPT: DECLARACIÓN Y EJECUCIÓN

Funciones en JavaScript: Declaración y Ejecución

1. ¿Qué es una función?

En JavaScript, una función es un bloque de código reutilizable que se ejecuta cuando se le llama. Permiten estructurar mejor el código, evitar la repetición y facilitar la depuración.

Las funciones pueden:

- Recibir parámetros (valores de entrada).
- Realizar cálculos o ejecutar instrucciones.
- Retornar un valor o simplemente ejecutar acciones.

📌 Ejemplo básico:

```
function saludar() {
    console.log("¡Hola, bienvenido a JavaScript!");
}
saludar(); // Llamada a la función
```

2. Sintaxis de una función en JavaScript

JavaScript permite definir funciones de distintas maneras:

2.1 Funciones declaradas

Estas funciones tienen un nombre y pueden ser llamadas en cualquier parte del código, incluso antes de su declaración debido al "hoisting".

📌 Sintaxis:

```
function nombreFuncion() {
```

```
    // Código a ejecutar
}
```

📌 **Ejemplo:**

```
function mostrarMensaje() {
    console.log("Esta es una función declarada.");
}
mostrarMensaje();
```

2.2 Funciones expresadas

Son funciones asignadas a una variable y no pueden ser llamadas antes de su declaración debido a que no se "elevan" con el hoisting.

📌 **Sintaxis:**

```
const miFuncion = function() {
    // Código a ejecutar
};
```

📌 **Ejemplo:**

```
const despedirse = function() {
    console.log("Adiós, hasta luego.");
};
despedirse();
```

3. Funciones con parámetros y valores de retorno

Las funciones pueden recibir valores de entrada llamados **parámetros**, y pueden devolver un resultado usando `return`.

📌 Ejemplo de función con parámetros:

```
function sumar(a, b) {
    return a + b;
}
let resultado = sumar(5, 3);
console.log("La suma es:", resultado); // La suma es: 8
```

📌 Explicación:

- La función `sumar` recibe dos parámetros (`a` y `b`).
- Usa `return` para devolver la suma de ambos números.
- Se almacena el resultado en la variable `resultado` y se muestra en consola.

4. Funciones flecha (`=>`)

Las funciones flecha son una sintaxis más corta para definir funciones en JavaScript.

📌 Sintaxis:

```
const nombreFuncion = (param1, param2) => {
    // Código a ejecutar
    return resultado;
};
```

Si la función tiene una sola línea y devuelve un valor, se pueden omitir las llaves `{}` y la palabra `return`.

📌 Ejemplo de función flecha:

```
const multiplicar = (x, y) => x * y;
console.log(multiplicar(4, 5)); // 20
```

📌 Ejemplo con más de una línea:

```
const esMayorDeEdad = (edad) => {
    if (edad >= 18) {
        return "Eres mayor de edad.";
    } else {
        return "Eres menor de edad.";
    }
};
console.log(esMayorDeEdad(20)); // Eres mayor de edad.
```

5. Creación de funciones simples en la consola del navegador

Para probar funciones en la consola del navegador:

1. Abrir el navegador (Chrome, Firefox, Edge).
2. Presionar `F12` o `Ctrl + Shift + I` para abrir las herramientas de desarrollador.
3. Ir a la pestaña `Consola`.
4. Escribir una función y ejecutarla.

📌 Ejemplo en consola:

```
function saludo() {
    return "¡Hola desde la consola!";
}
saludo();
```

6. Ejemplo de función con parámetros (`calcularAreaRectangulo`)

📌 Ejemplo:

```
function calcularAreaRectangulo(base, altura) {
    return base * altura;
}

let area = calcularAreaRectangulo(10, 5);
console.log("El área del rectángulo es:", area); // 50
```

📌 Explicación:

- La función recibe `base` y `altura` como parámetros.
- Multiplica ambos valores y devuelve el área.
- Se almacena el resultado en `area` y se muestra en consola.

7. Uso de `return` para devolver valores desde una función

El `return` finaliza la ejecución de la función y devuelve un valor.

📌 Ejemplo:

```
function obtenerMensaje() {
    return "Este mensaje se devuelve con return.";
}

let mensaje = obtenerMensaje();
console.log(mensaje); // Este mensaje se devuelve con return.
```

Si una función no tiene `return`, devolverá `undefined`.

📌 Ejemplo sin `return`:

```
function sinRetorno() {
    console.log("Esta función solo ejecuta código.");
}

let resultado = sinRetorno();
console.log(resultado); // undefined
```

Conclusión

- Las funciones en JavaScript permiten reutilizar código y mejorar la organización del programa.
- Hay varias formas de definir funciones: declaradas, expresadas y flecha.
- Se pueden pasar parámetros y devolver valores con `return`.
- Se pueden probar funciones directamente en la consola del navegador.

INSTITUTO

KHIPU