Lenguaje de Marcas: Javascript

Iván Nieto Ruiz

il.nietoruiz@edu.gva.es

Evaluación:

- Para la evaluación del módulo, se aplicará el siguiente criterio al finalizar cada una de las evaluaciones:
  - El 30% de la calificación corresponderá a la entrega de las prácticas.
  - El 70% de la calificación lo determinará dos examenes teórico/prácticos.
- Es importante destacar que se debe obtener al menos un **4,5 en cada parte** para poder realizar el promedio final entre ambas.

Introducción a JavaScript

¿Qué es JavaScript?

- Es un lenguaje de **programación interpretado**.
- Comúnmente usado para el desarrollo web.
- Ejecutado por un motor integrado en navegadores (como V8 de Chrome).

Características principales:

- Lenguaje de debilmente tipado y dinámico.
- Multi-paradigma: Soporta programación orientada a objetos, funcional e imperativa.
- Interactúa con HTML y CSS para crear experiencias dinámicas.

ECMAScript y JavaScript

¿Qué es ECMAScript?

- Es el estándar que define las características de JavaScript.
- Versiones importantes: ES5, ES6 (ES2015), y posteriores (hasta ES2023).

¿Por qué es importante?

Define mejoras en el lenguaje, como:

- `let`, `const`, y `var` para declarar variables.
- Funciones flecha (`=>`)
- Clases y módulos.

Herramientas para JavaScript

Editores recomendados:

- **Visual Studio Code** : Ligero, extensible y con soporte para JavaScript y Node.js.
    - Descarga: https://code.visualstudio.com/
- Otros: WebStorm, Atom, Sublime Text.

Editores online:

JSFiddle: https://jsfiddle.net/
CodePen: https://codepen.io/

Herramientas del navegador:

Chrome DevTools (F12): Depura y prueba código JavaScript en el navegador.
O las DevTools del resto de navegadores

Integrar JavaScript en HTML

Inline (no recomendado):

```
<body>
 <script>
  console.log("Hola Mundo!");
 </script>
</body>
```

Archivo externo (recomendado):

```
<body>
  <script src="js/app.js"></script>
</body>
```

Archivo app.js

```
console.log("Hola Mundo!");
```

Funciones de E/S

Funciones de salida:

console.log(mensaje) y **console.error(texto)**

Muestra información en la consola del navegador.

```
console.log("Esto es un mensaje en la consola");
```

alert(mensaje)

Muestra un cuadro de diálogo con un mensaje al usuario.

```
alert("Bienvenido a nuestra página web");
```

document.write(mensaje)

Muestra un cuadro de diálogo con un mensaje al usuario.

```
document.write("<h1>Hola Mundo</h1>");
```

Funciones de E/S

Funciones de entrada:

prompt(mensaje):

- Muestra un cuadro de diálogo que permite al usuario introducir datos.
- Devuelve siempre un valor de tipo `string`.

```
let nombre = prompt("¿Cuál es tu nombre?");
console.log("Hola, " + nombre);
```

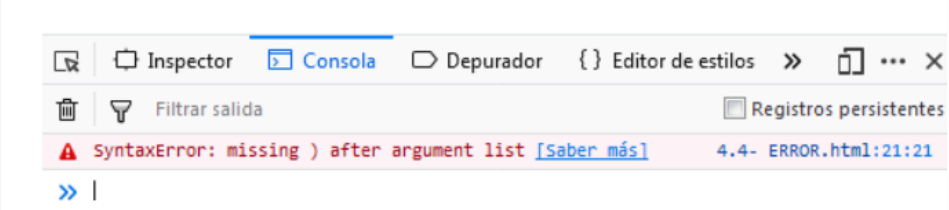
confirm()

La función confirm() en JavaScript se utiliza para mostrar un cuadro de diálogo con un mensaje, botones "Aceptar" y "Cancelar".

```
const respuesta = confirm("¿Estás seguro de que quieres continuar?");
if (respuesta) {
  console.log("El usuario hizo clic en 'Aceptar'.");
} else {
  console.log("El usuario hizo clic en 'Cancelar'.");
}
```

Errores de sintaxis

- En la solapa consola de las herramientas de desarrollador nos aparecerán los errores sintácticos que cometamos.
- El error aparecerá cuando el interprete trate de ejecutar la instrucción que contenga el error
- En el error nos indicará el fichero y la línea donde se produzca el error



Variables y tipos de datos en JS

Tipos de declaración:

- `var` (obsoleto): Alcance de función.
- `let`: Alcance de bloque, recomendado.
- `const`: Constante, no se puede reasignar.

```
let nombre = "Juan";
const edad = 25;
var saludo = "Hola";
```

Tipos de datos

Primitivos:

- `string` (cadena de texto)
- `number` (números)
- `boolean` (true/false)
- `undefined` (declarada pero sin valor asignado)
- `null` (valor intencionadamente vacío)

Objeto:

Arrays, funciones, objetos, etc.

```
let cadena = "Hola"; // string
let numero = 42; // number
let booleano = true; // boolean
let sinDefinir; // undefined
let vacio = null; // null
```

Operadores básicos

Aritméticos:

+, -, *, /, % (módulo)

Relacionales:

>, <, >=, <=, ==, !=, ===, !==

```
let a = 10, b = 20;
console.log(a + b); // 30
console.log(a > b); // false
console.log(a == "10"); // true (igualdad débil)
console.log(a === "10"); // false (igualdad estricta)
```

Variables en JS

Reglas básicas:
El nombre de una variable puede contener:

- Letras (a-z, A-Z)
- Números (0-9)
- El carácter _ (guion bajo)
- El carácter $

Importante: El primer carácter debe ser una **letra**, _, o $, **pero nunca un número.**

JavaScript es case sensitive (Sensibilidad a Mayúsculas y Minúsculas**)**

Se distingue entre mayúsculas y minúsculas en los nombres de las variables.

```
let nombre = "Juan";
let Nombre = "Pedro";

console.log(nombre); // Muestra: "Juan"
console.log(Nombre); // Muestra: "Pedro"
```

Variables en JS

Tipado Débil en JavaScript

El tipo de las variables depende del **valor asignado.**

No es necesario declarar el tipo explícitamente como en otros lenguajes.

```
let variable = 42; // Número
variable = "Hola"; // Cambia a cadena
variable = true; // Ahora es booleano
```

Nota: El tipo puede cambiar dinámicamente según el valor.

Cambio de Tipo Dinámico

JavaScript permite cambiar el tipo de una variable.

Esto puede provocar comportamientos inesperados si no se maneja correctamente.

Ejemplo:

```
let dato = "123"; // Tipo: string
dato = dato * 2; // Automáticamente se convierte a número
console.log(dato); // Resultado: 246
```

Cuidado con las operaciones dinámicas:

```
let x = "5";
let y = 3;

console.log(x + y); // Resultado: "53" (concatenación)
console.log(x * y); // Resultado: 15 (conversión automática a número)
```

Variables en JS

Buenas Prácticas

Usa nombres descriptivos para las variables:

```
let edad = 25; // Claro
let e = 25; // Poco claro
```

Utiliza camelCase para variables compuestas:

```
let nombreCompleto = "Juan Pérez";
let precioProducto = 19.99;
```

Declara variables con `let` o `const` en lugar de `var`:

```
const PI = 3.14159; // Valor constante
let total = 100; // Variable mutable
```

Evita usar nombres reservados de JavaScript:

```
let class = 10; // Inválido (class es una palabra reservada)
```

Hoisting en JS

El **hoisting** es un comportamiento de JavaScript en el que las declaraciones de variables, funciones o clases se "mueven" al principio de su contexto (función o script) durante la fase de compilación.

```
console.log(miVariable); // Resultado: undefined
var miVariable = 10;
console.log(miVariable); // Resultado: 10
```

Hoisting con **let y const:**

- Las variables declaradas con let y const también se "elevan", pero **no se inicializan automáticamente.**
- Intentar acceder a ellas antes de la declaración genera un **Error de referencia.**

```
console.log(miLet); // ReferenceError
let miLet = 20;
```

Hoisting también se da en Funciones, clases, expresiones

Tipos de variables

typeof: Indica el tipo de dato que en ese momento tiene la variable.

```
let v1 = "Hola Mundo!";
console.log(typeof v1); // Imprime -> string

v1 = 123;
console.log(typeof v1); // Imprime -> number
```

Hasta que le asignemos un valor, las variables tendrán un tipo especial conocido como **undefined**.

Este valor es diferente de **null** (que si se considera como un valor).

```
let v1;
console.log(typeof v1); // Imprime -> undefined
if (v1 === undefned) { // (!v1) or (typeof v1 === "undefned") también funciona
console.log("Has olvidado darle valor a v1");
}
```

¿Preguntas?