Unidad 7. JavaScript

1. Lenguajes de Script del Lado Cliente

- **Características generales →** Se ejecutan en el navegador web del usuario, permitiendo interactividad y dinamismo en función de eventos del usuario. Los scripts pueden estar integrados en HTML o en archivos independientes.
- **Diferencias con lenguajes del lado servidor →** Los lenguajes del lado servidor realizan tareas como acceso a bases de datos antes de enviar contenido al navegador, mientras que los del lado cliente complementan la experiencia con interactividad.
- **AJAX →** Tecnología que combina HTML, CSS, JavaScript y XML para transferencias asíncronas entre cliente y servidor, permitiendo actualizaciones en tiempo real sin recargar la página.
- **Limitaciones**:
  - No todos los navegadores soportan JavaScript.
  - Puede ser desactivado por el usuario.
  - Problemas de accesibilidad en navegadores de voz y algunos dispositivos móviles.

2. Características de JavaScript

- **Generalidades →** Es un lenguaje interpretado, orientado a objetos y con tipado débil. Fue creado por Brendan Eich en 1995 y sigue el estándar ECMA-262 desde 1997.
- **Usos principales**
  - Añadir interactividad (menús desplegables, efectos dinámicos).
  - Validar datos antes de enviarlos al servidor.
  - Modificar estilos y contenidos dinámicamente.
  - Realizar solicitudes al servidor.
- **Limitaciones**:
  - Incompatibilidades entre navegadores.
  - Dependencia de que el usuario no desactive JavaScript.

3. Estructura y Sintaxis Básica

- **Componentes principales**:
  - Variables → Declaradas con `const`, `let` o `var`.
  - Funciones → Bloques de código con una finalidad específica que pueden devolver resultados.
  - Instrucciones → Órdenes a ejecutar.
  - Comentarios → Explican el código (línea única `//` o varias líneas `/* */`).
- **Métodos para integrar JavaScript**:
- **Integración en línea →** Código dentro de HTML (no recomendado por falta de reutilización).
- **Código embebido →** Uso de etiquetas `<script>` dentro de `<head>` o `<body>`.
- **Código externo →** Archivos `.js` enlazados desde HTML. Es la opción recomendada por modularidad y reutilización.

4. Elementos Fundamentales del Lenguaje

- **Tipos de datos**
  - **Numéricos** → enteros, decimales, notación científica, hexadecimal y octal.
  - **Cadenas** → datos alfanuméricos entre comillas.
  - **Booleanos** → `true`/`false`.
  - **Punteros** → direcciones de memoria para funciones.
- **Variables**
  - Declaración con `const`, `let`, o `var`.
    - **Const** → Valor constante e inmutable.
    - **Let** / **var** → Valor modificable durante la ejecución.

5. Utilización de Objetos y DOM

5.1 Objetos en JavaScript

- Los objetos son estructuras que agrupan propiedades y métodos relacionados.

```
let persona = {
    nombre: "Juan",
    edad: 30,
    saludar: function() {
        console.log("Hola, soy " + this.nombre);
    }
};
persona.saludar(); // Output: Hola, soy Pepito
```

5.2 DOM (Document Object Model)

- Representa la estructura jerárquica de un documento HTML o XML como un árbol de nodos.
- Permite manipular elementos del documento mediante JavaScript.

```
document.getElementById("miElemento").style.color = "red";
```

- **Window()** → controla todo lo relativo a la ventana.

```
document.write("El nombre de la ventana es: [" + window.name + "]
"); window.name = "ventana";
```

Método	Descripción
alert()	Muestra una ventana emergente de alerta con un botón de aceptar
blur()	Elimina el foco de la ventana actual
close()	Cierra la ventana actual
confirm()	Muestra una ventana emergente con un mensaje, un botón de aceptar y otro de cancelar
focus()	Coloca el foco en la ventana actual
open()	Abre una nueva ventana de navegación
prompt()	Muestra una ventana de diálogo para introducir datos

- **navigator()** → contiene información sobre el navegador.
- **screen()** → permite obtener información sobre la pantalla del usuario.
- **document()** → enlaza directamente con window() y contiene la jerarquía de objetos del HTML.
- **history()** → permite acceder al historial de navegación.
- **location()** → proporciona información sobre la URL actual.

- Los nodos más utilizados en el DOM son
  - **document** → es la raíz de la estructura en árbol.
  - **element** → nodo hijo de raíz, identifica <html> y a su vez es padre de otros nodos element que contiene HTML.
  - **attr** → identifica los atributos de las etiquetas → "atributo=valor".
  - **text** → identifica el texto encerrado por una etiqueta.
  - **comment** → identifica los comentarios en la página HTML.

Acceder a nodos	Añadir nodos al DOM	Eliminar nodos
getElementById() getElementsByTagName() getElementByClassName() getElementByName() parentNode childNodes firstChild	**Crear nodos** createElement() createTextNode() **Añadir nodos** appendChild() insertBefore()	removeChild() replaceChild()
Modificar contenido de nodos	**Modificar atributo style**	**Modificar atributo class**
Contenido elementos: innerHTML, textContent - Atributo elementos: setAttribute(), getAttribute(), removeAttribute(), hasAttribute()	- Acceder al objeto style - Dar propiedades style - Crear un objeto style	- className - classList: add(), remove(), toggle(), contains(), items()

6. Eventos en JavaScript

- Los eventos son acciones que ocurren en el navegador (clics, movimientos del ratón, teclado, etc.).
- Métodos comunes para manejar eventos

```
document.getElementById("boton").addEventListener("click", function() {
    alert("¡Botón presionado!");
});
```

- **Evento de ratón** → usuario usa el ratón.
- **Evento de teclado** → usuario usa el teclado.
- **Evento de página** → al cambiar el estado de la página.
- **Eventos de formulario** → al cambiar el estado de algún elemento de un formulario.

7. Formularios y Validación

- Validación en el lado cliente mediante JavaScript antes de enviar datos al servidor.

```
function validarFormulario() {
    let nombre = document.getElementById("nombre").value;
    if (nombre === "") {
        alert("El campo nombre no puede estar vacío.");
        return false;
    }
    return true;
}
```