JavaScript

Programación Básica

Variables

- **Declaración**:

```
var numero = 10; // Declaración con 'var' (evitar su uso por alcance global)
let texto = "Hola"; // Declaración con 'let' (alcance de bloque)
const PI = 3.1416; // Declaración con 'const' (valor constante)
```

- **Tipos de Variables**:
  - **Numéricas**:

```
let entero = 5;
let decimal = 3.14;
```

  - **Cadenas de Texto**:

```
let mensaje = "Hola Mundo";
let letra = 'A';
```

    - Uso de comillas simples o dobles.
    - Caracteres especiales:
      - `\\n` Nueva línea
      - `\\t` Tabulador
      - `\\'` Comilla simple
      - `\\"` Comilla doble

        - `\\\\` Barra invertida
    - **Arrays**:

```
let dias = ["Lunes", "Martes", "Miércoles"];
console.log(dias[0]); // "Lunes"
```

    - **Booleanos**:

```
let esMayorDeEdad = true;
let tienePermiso = false;
```

Operadores

- **Asignación**:

```
let x = 10;
x += 5; // x = x + 5
```

- **Incremento y Decremento**:

```
let contador = 0;
contador++; // Incrementa después de usar el valor
++contador; // Incrementa antes de usar el valor
```

- **Lógicos**:
    - Negación: `!`

```
let activo = true;
if (!activo) {
  // Código si activo es false
}
```

    - AND: `&&`

```
if (esMayorDeEdad && tienePermiso) {
  // Código si ambas condiciones son verdaderas
}
```

- OR: `||`

```
if (esEstudiante || esProfesor) {
  // Código si al menos una condición es verdadera
}
```

- **Matemáticos**:
  - Suma: `+`
  - Resta: `-`
  - Multiplicación: `*`
  - División: `/`
  - Módulo: `%` (resto de la división)

```
let resto = 10 % 3; // resto = 1
```

- **Relacionales**:
  - Igualdad: `==` (igualdad débil), `===` (igualdad estricta)
  - Desigualdad: `!=`, `!==`
  - Mayor que: `>`
  - Menor que: `<`
  - Mayor o igual que: `>=`
  - Menor o igual que: `<=`

Estructuras de Control de Flujo

- **if**:

```
if (condicion) {
  // Código si la condición es verdadera
}
```

- **if...else**:

```
if (condicion) {
  // Código si es verdadera
```

```
} else {
  // Código si es falsa
}
```

- **if...else if...else**:

```
if (condicion1) {
  // Código
} else if (condicion2) {
  // Código
} else {
  // Código
}
```

- **for**:

```
for (let i = 0; i < 5; i++) {
  // Código que se repite
}
```

- **for...in** (recorrer propiedades de objetos):

```
for (let indice in dias) {
  console.log(dias[indice]);
}
```

- **while**:

```
while (condicion) {
  // Código que se repite mientras la condición sea verd
adera
}
```

- **do...while**:

```
do {
  // Código que se ejecuta al menos una vez
```

```
} while (condicion);
```

Funciones y Propiedades Básicas

- **Funciones para Cadenas de Texto**:
  - `length`: Obtiene la longitud.

    ```
    let longitud = mensaje.length;
    ```

  - `toUpperCase()`, `toLowerCase()`: Convierte a mayúsculas o minúsculas.
  - `charAt(posicion)`: Obtiene el carácter en la posición indicada.
  - `indexOf(cadena)`: Busca una cadena dentro de otra.
  - `substring(inicio, fin)`: Extrae una porción de la cadena.
  - `split(separador)`: Divide la cadena en un array.
- **Funciones para Arrays**:
  - `length`: Número de elementos.
  - `push(elemento)`: Añade al final.
  - `pop()`: Elimina el último elemento.
  - `shift()`: Elimina el primer elemento.
  - `unshift(elemento)`: Añade al inicio.
  - `reverse()`: Invierte el orden.
- **Funciones para Números**:
  - `toFixed(decimales)`: Formatea el número con el número de decimales especificado.

    ```
    let num = 3.1416;
    console.log(num.toFixed(2)); // "3.14"
    ```

Programación Avanzada

Funciones

- **Definición**:

```
function nombreFuncion(parametros) {
  // Código a ejecutar
  return resultado;
}
```

- **Argumentos y Valores de Retorno**:
  - Los argumentos se pasan al llamar la función.

```
function sumar(a, b) {
  return a + b;
}
let resultado = sumar(5, 3); // resultado = 8
```

- **Ámbito de Variables (Scope)**:
  - **Local**: Declaradas dentro de una función con `let` o `const`.
  - **Global**: Declaradas fuera de cualquier función o sin `let`, `var` o `const`.

```
let global = "Soy global";
function ejemplo() {
  let local = "Soy local";
}
```

Sentencias `break` y `continue`

- **break**: Termina el bucle actual.

```
for (let i = 0; i < 10; i++) {
  if (i === 5) break;
  console.log(i); // Imprime números del 0 al 4
}
```

- **continue**: Salta a la siguiente iteración.

```
for (let i = 0; i < 10; i++) {
  if (i % 2 === 0) continue;
```

```
  console.log(i); // Imprime números impares
}
```

Otras Estructuras de Control

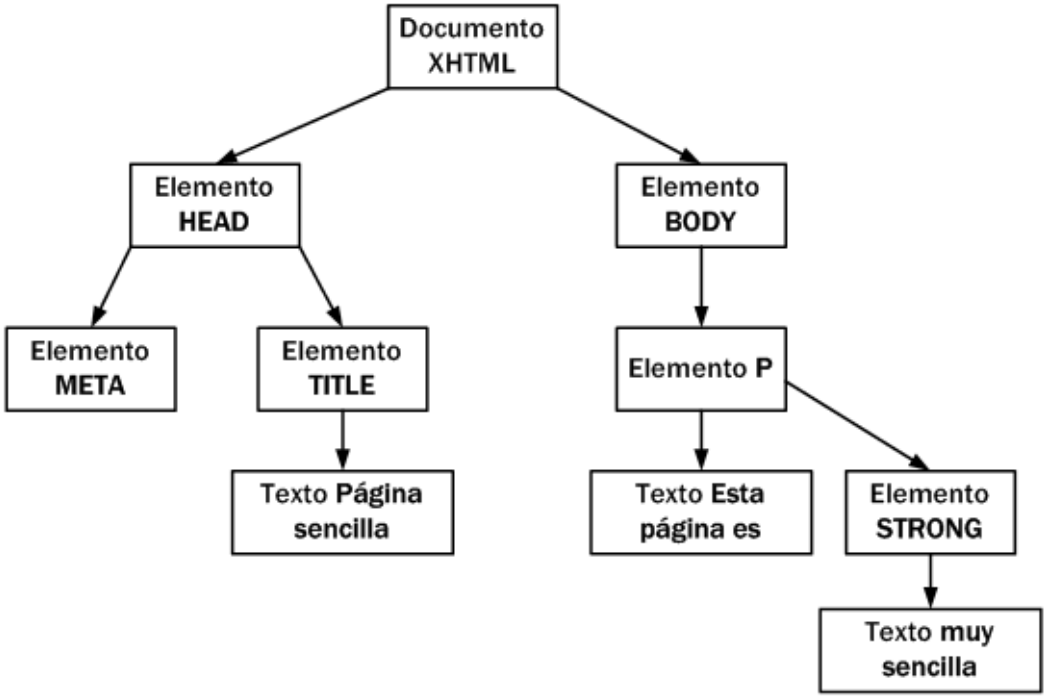
- **switch**:

```
switch (expresion) {
  case valor1:
    // Código
    break;
  case valor2:
    // Código
    break;
  default:
    // Código por defecto
}
```

DOM (Document Object Model)

Introducción al DOM

- **Árbol de Nodos**: Estructura jerárquica que representa la página web.
  - **Nodos Principales**:
    - **Document**: Nodo raíz.
    - **Element**: Representa elementos HTML.
    - **Text**: Contenido textual.
    - **Attr**: Atributos de elementos.
    - **Comment**: Comentarios en el HTML.



Acceso Directo a Nodos

- **getElementById(id)**: Selecciona un elemento por su `id`.

```
let elemento = document.getElementById("miId");
```

- **getElementsByTagName(etiqueta)**: Selecciona elementos por su etiqueta.

```
let parrafos = document.getElementsByTagName("p");
```

- **getElementsByName(name)**: Selecciona elementos por su atributo `name`.

```
let campos = document.getElementsByName("usuario");
```

- **querySelector(selector)**: Selecciona el primer elemento que coincide con el selector CSS.

```
let elemento = document.querySelector(".clase");
```

- **querySelectorAll(selector)**: Selecciona todos los elementos que coinciden con el selector CSS.

```
let elementos = document.querySelectorAll("div.clase");
```

Creación y Eliminación de Nodos

- **Crear Elementos**:

```
let nuevoParrafo = document.createElement("p");
```

- **Crear Nodos de Texto**:

```
let texto = document.createTextNode("Este es un nuevo pá
rrafo.");
```

- **Añadir Nodos**:

```
nuevoParrafo.appendChild(texto); // Añade el texto al pá
rrafo
document.body.appendChild(nuevoParrafo); // Añade el pár
rafo al body
```

- **Eliminar Nodos**:

```
let elemento = document.getElementById("miId");
elemento.parentNode.removeChild(elemento);
```

- **Reemplazar Nodos**:

```
let nuevoElemento = document.createElement("div");
elemento.parentNode.replaceChild(nuevoElemento, element
o);
```

Acceso Directo a Atributos y Propiedades

- **Atributos HTML**:

```
let enlace = document.getElementById("enlace");
let href = enlace.getAttribute("href");
```

```
enlace.setAttribute("href", "<https://nuevo-enlace.com
>");
```

- **Propiedades DOM**:

```
enlace.href = "<https://nuevo-enlace.com>";
```

- **Modificar Clases**:

```
elemento.className = "nueva-clase";
```

- **Estilos CSS**:

```
elemento.style.color = "red";
elemento.style.fontSize = "20px";
```

  - Propiedades CSS compuestas se escriben en camelCase:
    - `background-color` ⇒ `backgroundColor`
    - `margin-top` ⇒ `marginTop`

Eventos

Introducción a Eventos

- **Programación Basada en Eventos**: El código se ejecuta en respuesta a acciones del usuario o del navegador.
- **Tipos de Eventos**:
  - **Eventos de Ratón**: `onclick`, `onmouseover`, `onmouseout`, `onmousedown`, `onmouseup`, `onmousemove`
  - **Eventos de Teclado**: `onkeydown`, `onkeypress`, `onkeyup`
  - **Eventos de Formulario**: `onsubmit`, `onreset`, `onfocus`, `onblur`, `onchange`
  - **Eventos de Ventana**: `onload`, `onresize`, `onunload`

Manejadores de Eventos

- **Asignación como Atributo HTML** (no recomendado):

```
<button onclick="saludar()">Click aquí</button>
```

- **Asignación mediante Propiedades DOM**:

```
let boton = document.getElementById("miBoton");
boton.onclick = saludar;
```

- **Uso de** `addEventListener` (recomendado):

```
boton.addEventListener("click", saludar);
```

- **El Objeto** `this`:
  - Dentro de un manejador de eventos, `this` hace referencia al elemento que generó el evento.

```
boton.onclick = function() {
  this.style.backgroundColor = "blue";
};
```

Información sobre Eventos

- **El Objeto** `event`:
  - Contiene información detallada sobre el evento.
  - Acceso al objeto `event`:

```
boton.addEventListener("click", function(event) {
  // Código que utiliza 'event'
});
```

- **Propiedades Comunes**:
  - `event.type`: Tipo de evento (ej. "click").
  - `event.target`: Elemento que generó el evento.
- **Eventos de Teclado**:
  - `event.key`: Tecla pulsada (ej. "Enter", "a", "1").

- `event.keyCode`: Código numérico de la tecla (obsoleto).
- Detectar tecla pulsada:

```
document.addEventListener("keydown", function(event)
{
  if (event.key === "Enter") {
    // Código al presionar Enter
  }
});
```

- **Eventos de Ratón**:
  - `event.clientX`, `event.clientY`: Posición del cursor respecto a la ventana.
  - `event.pageX`, `event.pageY`: Posición del cursor respecto al documento.
  - `event.button`: Indica qué botón del ratón se presionó.

Previniendo Comportamientos por Defecto

- `event.preventDefault()`:
  - Evita la acción por defecto del navegador.

```
enlace.addEventListener("click", function(event) {
  event.preventDefault();
  // Código adicional
});
```

Propagación de Eventos

- **Fases**:
  - **Captura**: El evento se propaga desde el elemento más externo hasta el objetivo.
  - **Burbuja**: El evento se propaga desde el objetivo hacia los elementos padres.
- **Detener la Propagación**:
  - `event.stopPropagation()`:

```
elemento.addEventListener("click", function(event) {
  event.stopPropagation();
  // Código que no se propaga a elementos padres
});
```

Notas Adicionales

- **window.onload**:
  - Ejecuta código JavaScript una vez que toda la página ha cargado.

```
window.onload = function() {
  // Código a ejecutar después de cargar la página
};
```

- **DOMContentLoaded**:
  - Evento que se dispara cuando el DOM ha sido completamente cargado, sin esperar a que las hojas de estilo, imágenes y subframes terminen de cargar.

```
document.addEventListener("DOMContentLoaded", functio
n() {
  // Código a ejecutar cuando el DOM está listo
});
```
