**REPÚBLICA BOLIVARIANA DE VENEZUELA MINISTERIO DEL PODER POPULAR PARA LA EDUCACIÓN SUPERIOR**

**CENTRO EDUCATIVO LOGROS**

**INFORME DE ETIQUETAS HTML Y CSS**

**Autor (a):** Herlis Moya

**Curso:** Programación

**Docente:** Daniel Espitia

Maracaibo, Junio de 2025

**Etiquetas HTML y su Función Principal**

1. <!DOCTYPE html>: Define el tipo de documento y la versión de HTML.
2. <html>: Elemento raíz de todas las páginas HTML.
3. <head>: Contiene metadatos sobre el documento (título, enlaces a CSS, etc.).
4. <title>: Define el título de la página que aparece en la pestaña del navegador.
5. <meta>: Proporciona metadatos sobre el documento HTML (charset, descripción, viewport).
6. <link>: Enlaza recursos externos al documento, como hojas de estilo CSS.
7. <style>: Contiene código CSS directamente incrustado en el documento HTML.
8. <body>: Contiene todo el contenido visible de la página web.
9. <h1> a <h6>: Define encabezados de diferentes niveles de importancia.
10. <p>: Define un párrafo de texto.
11. <a>: Crea un hipervínculo a otra página o recurso.
12. <img>: Inserta una imagen en la página.
13. <ul>: Crea una lista desordenada (sin numerar).
14. <ol>: Crea una lista ordenada (numerada).
15. <li>: Define un elemento de una lista.
16. <div>: Un contenedor genérico a nivel de bloque para agrupar elementos.
17. <span>: Un contenedor genérico en línea para agrupar elementos.
18. <header>: Representa el contenido introductorio o un grupo de navegación.
19. <nav>: Contiene enlaces de navegación principales.
20. <main>: Representa el contenido principal del <body>.
21. <article>: Representa un contenido independiente y autocontenido.
22. <section>: Agrupa contenido temáticamente relacionado.
23. <aside>: Contenido que está indirectamente relacionado con el contenido principal (por ejemplo, barras laterales).
24. <footer>: Contiene información sobre el autor, derechos de autor, enlaces, etc.
25. <form>: Crea un formulario HTML para la entrada de datos del usuario.
26. <input>: Crea un campo de entrada de datos (texto, contraseña, botón, etc.).
27. <button>: Crea un botón clicable.
28. <textarea>: Crea un área de texto multilinea.
29. <select>: Crea una lista desplegable.
30. <option>: Define una opción dentro de una lista desplegable <select>.
31. <label>: Asocia una etiqueta de texto a un control de formulario.
32. <table>: Crea una tabla.
33. <thead>: Agrupa las filas de encabezado de una tabla.
34. <tbody>: Agrupa las filas del cuerpo de una tabla.
35. <tfoot>: Agrupa las filas de pie de una tabla.
36. <tr>: Define una fila en una tabla.
37. <th>: Define una celda de encabezado en una tabla.
38. <td>: Define una celda de datos en una tabla.
39. <video>: Incrusta un reproductor de video.
40. <audio>: Incrusta un reproductor de audio.
41. <source>: Especifica múltiples recursos multimedia para <video> y <audio>.
42. <canvas>: Proporciona un área de dibujo para gráficos programables.
43. <svg>: Incrusta gráficos vectoriales escalables.
44. <iframe>: Incrusta otro documento HTML dentro del actual.
45. <br>: Inserta un salto de línea.
46. <hr>: Inserta una línea horizontal temática.
47. <em>: Enfatiza texto (generalmente se muestra en cursiva).
48. <strong>: Indica texto de gran importancia (generalmente se muestra en negrita).
49. <code>: Muestra código de programación.
50. <pre>: Muestra texto preformateado (conserva espacios y saltos de línea).

**Etiquetas CSS y su Función Principal**

1. color: Establece el color del texto.
2. background-color: Establece el color de fondo de un elemento.
3. font-family: Define la fuente del texto.
4. font-size: Establece el tamaño de la fuente.
5. font-weight: Establece el grosor de la fuente (negrita, normal, etc.).
6. text-align: Alinea el texto (left, right, center, justify).
7. line-height: Establece la altura de una línea de texto.
8. margin: Establece el margen alrededor de un elemento.
9. padding: Establece el espacio interno de un elemento.
10. border: Establece el borde de un elemento (grosor, estilo, color).
11. width: Establece el ancho de un elemento.
12. height: Establece la altura de un elemento.
13. display: Define cómo se muestra un elemento (block, inline, flex, grid, none).
14. position: Define el tipo de posicionamiento de un elemento (static, relative, absolute, fixed, sticky).
15. top, right, bottom, left: Establece las posiciones de un elemento posicionado.
16. z-index: Establece el orden de apilamiento de los elementos posicionados.
17. overflow: Controla cómo se maneja el contenido que desborda el cuadro de un elemento.
18. float: Posiciona un elemento a la izquierda o derecha, permitiendo que el texto fluya alrededor.
19. clear: Especifica qué lados de un elemento no pueden tener elementos flotantes a su lado.
20. opacity: Establece la opacidad (transparencia) de un elemento.
21. box-shadow: Agrega una sombra a un elemento.
22. text-shadow: Agrega una sombra al texto.
23. border-radius: Crea esquinas redondeadas en un elemento.
24. background-image: Establece una imagen de fondo.
25. background-repeat: Controla cómo se repite una imagen de fondo.
26. background-position: Controla la posición inicial de una imagen de fondo.
27. background-size: Controla el tamaño de la imagen de fondo.
28. list-style-type: Define el estilo del marcador de la lista (disc, circle, square, none).
29. text-decoration: Agrega o quita decoraciones de texto (underline, overline, line-through, none).
30. cursor: Define el tipo de cursor que se muestra cuando se pasa el ratón sobre un elemento.
31. transition: Permite animar cambios de propiedades CSS de forma suave.
32. transform: Aplica transformaciones 2D o 3D a un elemento (translate, rotate, scale, skew).
33. flex-direction: Define la dirección en la que los elementos flexibles se colocan en un contenedor flex.
34. justify-content: Alinea los elementos flexibles a lo largo del eje principal.
35. align-items: Alinea los elementos flexibles a lo largo del eje transversal.
36. gap (o grid-gap): Establece el espacio entre las filas y columnas en diseños de cuadrícula o flexbox.
37. grid-template-columns: Define el número y el tamaño de las columnas en una cuadrícula.
38. grid-template-rows: Define el número y el tamaño de las filas en una cuadrícula.
39. grid-column: Especifica en qué columnas se extiende un elemento de cuadrícula.
40. grid-row: Especifica en qué filas se extiende un elemento de cuadrícula.
41. media (dentro de @media): Define condiciones para aplicar estilos (ej. para pantallas de cierto tamaño).
42. font-style: Define el estilo de la fuente (normal, italic, oblique).
43. text-transform: Convierte el texto a mayúsculas, minúsculas o capitaliza.
44. letter-spacing: Establece el espacio entre caracteres de texto.
45. word-spacing: Establece el espacio entre palabras.
46. white-space: Controla cómo se maneja el espacio en blanco dentro de un elemento.
47. box-sizing: Define cómo se calcula el ancho y alto de un elemento (content-box, border-box).
48. outline: Dibuja una línea alrededor de los elementos, fuera del borde.
49. visibility: Controla si un elemento es visible o no (visible, hidden).
50. pointer-events: Controla cómo un elemento reacciona a los eventos del puntero del ratón (clicks, etc.).

## Conceptos Básicos de JavaScript y su Aplicación

JavaScript es un lenguaje de programación de alto nivel, interpretado y orientado a objetos que se utiliza principalmente para hacer que las páginas web sean interactivas. Se ejecuta en el lado del cliente (en el navegador del usuario), pero también puede ejecutarse en el servidor (con Node.js).

### Conceptos Clave:

1. **Variables:** Contenedores para almacenar datos. Se declaran con var, let o const.
   * var: Alcance de función o global.
   * let: Alcance de bloque. Permite reasignación.
   * const: Alcance de bloque. No permite reasignación una vez inicializada.

**Ejemplo:**

JavaScript

let nombre = "Juan"; // Variable de tipo string

const edad = 30; // Variable de tipo number, no se puede reasignar

var esEstudiante = true; // Variable de tipo boolean

1. **Tipos de Datos:**
   * **Primitivos:**
     + String: Texto ("Hola").
     + Number: Números (enteros y flotantes) (10, 3.14).
     + Boolean: Verdadero o falso (true, false).
     + Undefined: Una variable declarada pero sin valor asignado.
     + Null: Representa la ausencia intencional de cualquier valor.
     + Symbol: Valores únicos e inmutables (ES6).
     + BigInt: Números enteros muy grandes (ES11).
   * **No Primitivos (Objetos):**
     + Object: Colecciones de pares clave-valor ({ clave: valor }).
     + Array: Listas ordenadas de valores ([valor1, valor2]).
     + Function: Bloques de código reutilizables.

**Ejemplo:**

JavaScript

let persona = {

nombre: "Ana",

apellido: "García",

edad: 25

};

let numeros = [1, 2, 3, 4, 5];

1. **Operadores:** Símbolos que realizan operaciones sobre uno o más operandos.
   * **Aritméticos:** +, -, \*, /, % (módulo), \*\* (exponenciación).
   * **Asignación:** =, +=, -=, \*=, /=.
   * **Comparación:** == (igualdad de valor), === (igualdad de valor y tipo), !=, !==, <, >, <=, >=.
   * **Lógicos:** && (AND), || (OR), ! (NOT).

**Ejemplo:**

JavaScript

let a = 10;

let b = 5;

let suma = a + b; // 15

let esMayor = (a > b) && (b < 10); // true

1. **Estructuras de Control (Condicionales y Bucles):**
   * **if...else if...else:** Ejecuta un bloque de código si una condición es verdadera. **Ejemplo:**

JavaScript

let hora = 14;

if (hora < 12) {

console.log("Buenos días");

} else if (hora < 18) {

console.log("Buenas tardes");

} else {

console.log("Buenas noches");

}

* + **switch:** Una alternativa a if...else if cuando hay muchas condiciones posibles. **Ejemplo:**

JavaScript

let dia = "Lunes";

switch (dia) {

case "Lunes":

console.log("Es el primer día de la semana.");

break;

case "Viernes":

console.log("Ya casi es fin de semana.");

break;

default:

console.log("Es un día normal.");

}

* + **for loop:** Repite un bloque de código un número específico de veces. **Ejemplo:**

JavaScript

for (let i = 0; i < 5; i++) {

console.log(i); // Imprime 0, 1, 2, 3, 4

}

* + **while loop:** Repite un bloque de código mientras una condición sea verdadera. **Ejemplo:**

JavaScript

let contador = 0;

while (contador < 3) {

console.log("Contador: " + contador);

contador++;

}

* + **forEach (para arrays):** Itera sobre cada elemento de un array. **Ejemplo:**

JavaScript

let frutas = ["manzana", "banana", "cereza"];

frutas.forEach(function(fruta) {

console.log(fruta);

});

1. **Funciones:** Bloques de código reutilizables que realizan una tarea específica. Pueden tomar parámetros y devolver un valor.

**Ejemplo:**

JavaScript

// Función declarada

function saludar(nombre) {

return "Hola, " + nombre + "!";

}

console.log(saludar("Pedro")); // Salida: Hola, Pedro!

// Función de expresión

const sumar = function(a, b) {

return a + b;

};

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

// Función de flecha (ES6)

const multiplicar = (a, b) => a \* b;

console.log(multiplicar(4, 2)); // Salida: 8

1. **Manipulación del DOM (Document Object Model):** JavaScript interactúa con el HTML y CSS de una página web a través del DOM, que representa la estructura de la página como un árbol de objetos.

**Ejemplo:**

HTML

<!DOCTYPE html>

<html lang="es">

<head>

<title>Manipulación del DOM</title>

</head>

<body>

<h1 id="titulo">Hola Mundo!</h1>

<button id="miBoton">Cambiar Texto</button>

<script>

// Acceder a un elemento por su ID

const tituloElemento = document.getElementById("titulo");

console.log(tituloElemento.textContent); // Salida: Hola Mundo!

// Cambiar el texto de un elemento

tituloElemento.textContent = "¡Hola JavaScript!";

// Acceder a un botón y añadir un evento click

const boton = document.getElementById("miBoton");

boton.addEventListener("click", function() {

alert("Botón clicado!");

// También puedes cambiar el estilo

tituloElemento.style.color = "blue";

});

// Crear un nuevo elemento y añadirlo al body

const nuevoParrafo = document.createElement("p");

nuevoParrafo.textContent = "Este es un párrafo añadido con JavaScript.";

document.body.appendChild(nuevoParrafo);

</script>

</body>

</html>

1. **Eventos:** Acciones que ocurren en el navegador (clics, pulsaciones de teclas, carga de página, etc.). JavaScript permite responder a estos eventos.

**Ejemplo (incluido en el ejemplo de DOM):**

JavaScript

boton.addEventListener("click", function() {

alert("Botón clicado!");

});

Otros eventos comunes son mouseover, keydown, submit, load.

1. **Programación Asíncrona (Conceptos básicos):**
   * **Callbacks:** Funciones que se pasan como argumentos a otras funciones y se ejecutan después de que la función principal haya terminado.
   * **Promises:** Objetos que representan la eventual finalización (o falla) de una operación asíncrona y su valor resultante.
   * **async/await:** Una sintaxis más moderna y legible para trabajar con Promises, haciendo que el código asíncrono parezca síncrono.

**Ejemplo básico con setTimeout (simulando una operación asíncrona):**

JavaScript

console.log("Inicio de la operación.");

setTimeout(function() {

console.log("¡Operación asíncrona completada después de 2 segundos!");

}, 2000); // Espera 2000 milisegundos (2 segundos)

console.log("Esto se ejecuta antes que la operación asíncrona.");

**Salida:**

Inicio de la operación.

Esto se ejecuta antes que la operación asíncrona.

¡Operación asíncrona completada después de 2 segundos!