REPÚBLICA BOLIVARIANA DE VENEZUELA
LOGROS
FCURSO DE PROGRAMACION
SECCIÓN: 14859

Exploradores del Código Web Avanzado.

CSS, HTML Y JAVASCRIPT.

PRESENTADO POR:
JOANDER PARRA C.I:31.188.413

MARACAIBO, JUNIO 2025

INTRODUCCIÓN

En la era digital actual, el desarrollo web se ha consolidado como una disciplina fundamental, impulsando la interacción global y facilitando el acceso a una vasta cantidad de información y servicios. En el corazón de esta revolución digital se encuentran tres tecnologías esenciales: HTML, CSS y JavaScript. Estos lenguajes no solo son la base sobre la que se construyen todas las páginas web, sino que también son los responsables de la estructura, el estilo y la interactividad que percibimos al navegar por internet.

HTML (HyperText Markup Language) es el esqueleto de cualquier sitio web, definiendo y organizando el contenido mediante etiquetas semánticas. Sin HTML, una página web sería simplemente un conjunto de texto sin formato ni jerarquía. Por otro lado, CSS (Cascading Style Sheets) es el encargado de dar vida visual a ese esqueleto. Es el artista que aplica colores, fuentes, diseños y adaptabilidad, transformando una estructura básica en una experiencia visualmente atractiva y coherente. Finalmente, JavaScript es el cerebro que dota de dinamismo e interactividad a las páginas web. Es el lenguaje que permite la manipulación del contenido en tiempo real, la validación de formularios, la creación de animaciones complejas y la comunicación con servidores, ofreciendo una experiencia de usuario fluida y personalizada.

El objetivo de este informe es profundizar en cada una de estas tecnologías, explorando sus características fundamentales, sus roles específicos en el proceso de desarrollo web y cómo interactúan entre sí para crear las experiencias digitales que conocemos y utilizamos a diario. Al comprender la sinergia entre HTML, CSS y JavaScript, se puede apreciar la complejidad y la creatividad inherentes a la construcción de la web moderna.

DESARROLLO

- **50 etiquetas HTML más utilizadas en el desarrollo web moderno**

1. `<html>`: La raíz de todo documento HTML.
2. `<head>`: Contiene metadatos sobre el documento (título, enlaces a CSS, scripts).
3. `<body>`: Contiene todo el contenido visible de la página web.
4. `<header>`: Representa el contenido introductorio o un grupo de ayudas a la navegación.
5. `<nav>`: Contiene enlaces de navegación principales.
6. `<footer>`: Representa el pie de sección o pie de página del documento.
7. `<main>`: Representa el contenido dominante de un documento.
8. `<article>`: Contenido independiente y autónomo (ej. un post de blog, un artículo de noticias).
9. `<section>`: Agrupa contenido relacionado, generalmente con un encabezado.
10. `<aside>`: Contenido que no está directamente relacionado con el contenido principal, pero que es relevante (ej. barras laterales).
11. `<div>`: Un contenedor genérico a nivel de bloque para agrupar contenido.
12. `<span>`: Un contenedor genérico a nivel de línea para agrupar contenido.

Etiquetas de Texto y Tipografía

13. `<h1>` a `<h6>`: Encabezados de diferentes niveles de importancia.
14. `<p>`: Párrafo de texto.
15. `<a>`: Enlace de hipertexto (anchor).
16. `<strong>`: Texto con gran importancia (generalmente en negrita).
17. `<em>`: Texto con énfasis (generalmente en cursiva).
18. `<b>`: Texto en negrita (sin implicar importancia semántica).
19. `<i>`: Texto en cursiva (sin implicar énfasis semántico, ej. términos técnicos, frases en otro idioma).
20. `<u>`: Texto subrayado.

21. <small>: Texto más pequeño (ej. derechos de autor, letra pequeña).
22. <code>: Muestra un fragmento de código de programación.
23. <pre>: Muestra texto preformateado (conserva espacios y saltos de línea).
24. <blockquote>: Citas de bloques de texto.
25. <q>: Citas en línea.
26.
: Salto de línea.
27. <hr>: Línea horizontal temática (separador).

Etiquetas de Listas

28. : Lista desordenada (sin orden específico).
29. : Lista ordenada (con numeración o letras).
30. : Elemento de lista.
31. <dl>: Lista de descripción.
32. <dt>: Término o nombre en una lista de descripción.
33. <dd>: Descripción del término en una lista de descripción.

Etiquetas de Multimedia

34. <img>: Inserta una imagen.
35. <video>: Inserta un video.
36. <audio>: Inserta un audio.
37. <source>: Especifica múltiples recursos multimedia para <video>, <audio> o <picture>.

Etiquetas de Formularios

38. <form>: Define un formulario HTML.
39. <input>: Campo de entrada de datos (texto, número, email, password, checkbox, radio, etc.).
40. <button>: Botón interactivo.
41. <textarea>: Área de texto multilínea.
42. <label>: Etiqueta para un elemento de formulario.
43. <select>: Lista desplegable de opciones.
44. <option>: Opción dentro de un <select>.
45. <fieldset>: Agrupa elementos relacionados en un formulario.
46. <legend>: Título para el contenido de un <fieldset>.

Etiquetas de Tablas

47. `<table>`: Crea una tabla.
48. `<thead>`: Grupo de filas de encabezado de una tabla.
49. `<tbody>`: Grupo de filas del cuerpo de una tabla.
50. `<tr>`: Fila de una tabla.

- **50 etiquetas CSS más utilizadas en el desarrollo web moderno**

1. `<html>`: La raíz de todo documento HTML.
2. `<head>`: Contiene metadatos sobre el documento (título, enlaces a CSS, scripts).
3. `<body>`: Contiene todo el contenido visible de la página web.
4. `<header>`: Representa el contenido introductorio o un grupo de ayudas a la navegación.
5. `<nav>`: Contiene enlaces de navegación principales.
6. `<footer>`: Representa el pie de sección o pie de página del documento.
7. `<main>`: Representa el contenido dominante de un documento.
8. `<article>`: Contenido independiente y autónomo (ej. un post de blog, un artículo de noticias).
9. `<section>`: Agrupa contenido relacionado, generalmente con un encabezado.
10. `<aside>`: Contenido que no está directamente relacionado con el contenido principal, pero que es relevante (ej. barras laterales).
11. `<div>`: Un contenedor genérico a nivel de bloque para agrupar contenido.
12. `<span>`: Un contenedor genérico a nivel de línea para agrupar contenido.

Etiquetas de Texto y Tipografía

13. `<h1>` a `<h6>`: Encabezados de diferentes niveles de importancia.
14. `<p>`: Párrafo de texto.
15. `<a>`: Enlace de hipertexto (anchor).
16. `<strong>`: Texto con gran importancia (generalmente en negrita).
17. `<em>`: Texto con énfasis (generalmente en cursiva).
18. `<b>`: Texto en negrita (sin implicar importancia semántica).
19. `<i>`: Texto en cursiva (sin implicar énfasis semántico, ej. términos técnicos, frases en otro idioma).
20. `<u>`: Texto subrayado.
21. `<small>`: Texto más pequeño (ej. derechos de autor, letra pequeña).

22. `<code>`: Muestra un fragmento de código de programación.
23. `<pre>`: Muestra texto preformateado (conserva espacios y saltos de línea).
24. `<blockquote>`: Citas de bloques de texto.
25. `<q>`: Citas en línea.
26. `<br>`: Salto de línea.
27. `<hr>`: Línea horizontal temática (separador).

Etiquetas de Listas

28. `<ul>`: Lista desordenada (sin orden específico).
29. `<ol>`: Lista ordenada (con numeración o letras).
30. `<li>`: Elemento de lista.
31. `<dl>`: Lista de descripción.
32. `<dt>`: Término o nombre en una lista de descripción.
33. `<dd>`: Descripción del término en una lista de descripción.

Etiquetas de Multimedia

34. `<img>`: Inserta una imagen.
35. `<video>`: Inserta un video.
36. `<audio>`: Inserta un audio.
37. `<source>`: Especifica múltiples recursos multimedia para `<video>`, `<audio>` o `<picture>`.

Etiquetas de Formularios

38. `<form>`: Define un formulario HTML.
39. `<input>`: Campo de entrada de datos (texto, número, email, password, checkbox, radio, etc.).
40. `<button>`: Botón interactivo.
41. `<textarea>`: Área de texto multilínea.
42. `<label>`: Etiqueta para un elemento de formulario.
43. `<select>`: Lista desplegable de opciones.
44. `<option>`: Opción dentro de un `<select>`.
45. `<fieldset>`: Agrupa elementos relacionados en un formulario.
46. `<legend>`: Título para el contenido de un `<fieldset>`.

Etiquetas de Tablas

47. `<table>`: Crea una tabla.
48. `<thead>`: Grupo de filas de encabezado de una tabla.
49. `<tbody>`: Grupo de filas del cuerpo de una tabla.
50. `<tr>`: Fila de una tabla.

¿Qué es JavaScript y para qué se utiliza?

JavaScript (JS) es un lenguaje de programación versátil y dinámico que impulsa la interactividad en la web. A diferencia de HTML (estructura) y CSS (estilo), JS dota de comportamiento a las páginas, permitiendo que respondan a acciones del usuario y eventos.

Se utiliza principalmente para:

- **Desarrollo Front-end:** Crea la interactividad que ves en los navegadores, como validación de formularios, animaciones, actualizaciones de contenido en tiempo real y manipulación del DOM.
- **Desarrollo Back-end (con Node.js):** Permite construir el lado del servidor de las aplicaciones, incluyendo APIs, servidores web y aplicaciones en tiempo real, usando el mismo lenguaje.
- **Aplicaciones Móviles y de Escritorio:** Con frameworks como React Native o Electron, se pueden desarrollar apps nativas y de escritorio multiplataforma.

Variables (qué son, cómo se declaran: var, let, const)

Las **variables** en JavaScript son como **contenedores con nombre** que guardan valores de datos. Su contenido puede cambiar durante la ejecución del programa.

Para declararlas, usamos tres palabras clave, cada una con reglas de **alcance (scope)** y **re-asignación** distintas:

1. **`var`**: La forma más antigua. Tiene **alcance de función o global**, lo que significa que puede ser accesible fuera del bloque donde se declara (como un `if` o `for`), y permite **re-declaración** y
2. **re-asignación**. Es menos recomendada hoy en día por sus peculiaridades.
3. **`let`**: La forma moderna y preferida para variables que van a cambiar. Tiene **alcance de bloque**, lo que las limita al `{}` donde son declaradas. Permite **re-asignación**, pero **no re-declaración** en el mismo alcance.
4. **`const`**: Ideal para valores que no deben cambiar después de su asignación inicial. También tiene **alcance de bloque** y **no permite re-declaración ni re-asignación** de la variable en sí. Sin embargo, si el valor es un objeto o un array, sí se pueden modificar sus propiedades o elementos internos.

Tipos de datos (string, number, boolean, array, object)

En JavaScript, los **tipos de datos** definen el tipo de valor que una variable puede almacenar. JavaScript es de **tipado dinámico**, por lo que no necesitas declarar el tipo.

Los tipos fundamentales son:

- **`String`**: Texto, encerrado entre comillas simples, dobles o backticks (para interpolación y multilínea). Ej: `"Hola Mundo"`.
- **`Number`**: Números, tanto enteros como decimales. Incluye valores especiales como `NaN` (no es un número) e `Infinity`. Ej: `42`, `3.14`.
- **`Boolean`**: Valores lógicos, solo `true` (verdadero) o `false` (falso). Esencial para la lógica condicional. Ej: `true`.
- **`Array`**: Colecciones **ordenadas** de elementos, accesibles por índice numérico (empezando en 0). Se declaran con corchetes `[]`. Ej: `["manzana", "banana"]`.

- **`Object`**: Colecciones de **pares clave-valor** (propiedades). Representan entidades con características y métodos. Se declaran con llaves `{}`. Ej: `{ nombre: "Ana", edad: 30 }`.

Operadores (aritméticos, de comparación, lógicos)

En JavaScript, los **operadores** son símbolos que realizan acciones sobre valores para producir un resultado. Los más comunes son:

- **Aritméticos**: Para cálculos matemáticos. Incluyen `+` (suma/concatenación), `-` (resta), `*` (multiplicación), `/` (división), `%` (módulo), `**` (exponenciación), `++` (incremento) y `--` (decremento).
- **De Comparación**: Comparan valores y devuelven `true` o `false`. Destacan `==` (igualdad suelta, con conversión de tipo) y `===` (igualdad estricta, sin conversión de tipo, **más recomendada**), así como `!=` (desigualdad suelta), `!==` (desigualdad estricta), `>`, `<`, `>=`, y `<=`.
- **Lógicos**: Combinan o invierten valores booleanos. Son `&&` (AND, ambos deben ser `true`), `||` (OR, al menos uno debe ser `true`) y `!` (NOT, invierte el valor).

Condicionales (if, else if, else)

Las **condicionales** en JavaScript (`if`, `else if`, `else`) son herramientas para ejecutar diferentes bloques de código basándose en si una **condición es verdadera** (true) o falsa (false).

- **`if`**: Ejecuta un bloque de código **solo si su condición es `true`**.
- **`else`**: Ejecuta un bloque de código **si la condición del `if` asociado es `false`**.
- **`else if`**: Permite probar **múltiples condiciones en secuencia** si las anteriores fueron `false`. Se ejecuta el primer `else if` cuya condición sea `true`. Un `else` final puede actuar como un caso por defecto.

if (condicion1) { // Si condicion1 es verdadera }

else if (condicion2) { // Si condicion1 es falsa Y condicion2 es verdadera } else { // Si ninguna es verdadera }

Bucles (for, while)

El bucle `for` se utiliza cuando se sabe (o se puede determinar) **cuántas veces** se debe repetir un bloque de código. Es ideal para iterar sobre colecciones de elementos como arrays o para repetir un número fijo de veces.

for (inicializacion; condicion; expresion_final) {

// Código a ejecutar en cada iteración}

El bucle `while` se utiliza cuando se quiere repetir un bloque de código **mientras una condición específica sea verdadera**. Es ideal cuando el número exacto de repeticiones no se conoce de antemano y depende de la evolución de la condición.

while (condicion) {

// Código a ejecutar mientras la condición sea verdadera

// Es crucial que algo dentro del bucle cambie la condición para que eventualmente sea falsa,

// de lo contrario, se creará un bucle infinito.}

Funciones (qué son, cómo se declaran y se llaman)

Las **funciones** en JavaScript son **bloques de código reutilizable** que realizan una tarea específica. Son esenciales para organizar tu código, evitar repeticiones (principio DRY) y facilitar el mantenimiento. Piensa en ellas como "mini-programas" que puedes ejecutar cuando los necesites.

Declaración de Función (`function`): La forma clásica. Son "elevadas" (hoisted), lo que significa que puedes llamarlas antes de su definición en el código.

function nombreFuncion(parametro1, parametro2) {

** // código**

** return resultado; // opcional}**

Expresión de Función: Se asigna una función a una variable. No son elevadas, por lo que debes definirlas antes de usarlas.

const nombreFuncion = function(parametro1, parametro2) {

** // código};**

Funciones de Flecha (`=>`): Una sintaxis más concisa (ES6) para expresiones de función. Son útiles para funciones anónimas y tienen un manejo diferente de `this`.

const nombreFuncion = (parametro1, parametro2) => {

** // código**

** return resultado;**

};

// O más conciso para una sola expresión:

const sumar = (a, b) => a + b;

Para llamar una funcion:

nombreFuncion(argumento1, argumento2); // Ejemplo de llamada

Manipulación básica del DOM (qué es el DOM, getElementById, innerHTML, querySelector, addEventListener)

La **Manipulación del DOM** en JavaScript te permite hacer que tus páginas web sean dinámicas. El **DOM (Document Object Model)** es una representación en forma de árbol de tu documento HTML, que JavaScript puede acceder y modificar.

Para interactuar con el DOM, necesitas primero **seleccionar** los elementos HTML:

- **`document.getElementById('id')`**: Busca un elemento por su ID único. Devuelve un solo elemento.
- **`document.querySelector('selectorCSS')`**: Devuelve el **primer** elemento que coincida con un selector CSS (ej. `#id`, `.clase`, `p`).
- **`document.querySelectorAll('selectorCSS')`**: Devuelve una lista (NodeList) de **todos** los elementos que coincidan con un selector CSS.

Una vez que tienes un elemento, puedes **modificar su contenido** con:

- **`elemento.innerHTML`**: Permite obtener o establecer el contenido HTML (incluidas etiquetas) de un elemento. Úsalo con precaución por seguridad.

Finalmente, para que tu página reaccione a las acciones del usuario, usas el **manejo de eventos**:

- **`elemento.addEventListener('evento', funcion)`**: Es el método preferido para "escuchar" un evento (como un `'click'`) en un elemento y ejecutar una función específica cuando ocurre.

En resumen, el DOM es el puente entre tu JavaScript y lo que el usuario ve y con lo que interactúa en la página.

CONCLUSIÓN

El recorrido a través de HTML, CSS y JavaScript nos revela la arquitectura esencial y la sinergia que impulsa la World Wide Web moderna. Cada uno de estos lenguajes cumple un rol irremplazable, y su dominio conjunto es lo que transforma una idea en una experiencia digital interactiva y visualmente atractiva.

HTML es la **estructura y el contenido**, el esqueleto semántico que da forma a la información. Sin él, no habría dónde colocar texto, imágenes o videos. Es la base sobre la que todo lo demás se construye, asegurando que el contenido sea accesible y esté bien organizado.

CSS es el **estilo y el diseño**, la vestimenta que embellece y organiza el esqueleto HTML. Es el responsable de la estética, la tipografía, los colores y la disposición visual, garantizando que la página no solo funcione, sino que también sea atractiva y usable en diferentes dispositivos.

Finalmente, **JavaScript** es la **interactividad y el comportamiento**, el cerebro dinámico que da vida a la página. Permite que el usuario interactúe con el sitio, que el contenido cambie sin recargar y que la aplicación reaccione a eventos. Desde la validación de formularios hasta animaciones complejas y la comunicación con servidores, JavaScript convierte una página estática en una aplicación rica y responsiva.

En conjunto, esta tríada constituye el núcleo del desarrollo front-end. HTML crea la información, CSS la hace hermosa, y JavaScript la hace funcional e interactiva. Comprender y dominar estas tecnologías es el primer y más crucial paso para cualquier persona que desee construir y moldear el paisaje digital de hoy.

BIBLIOGRAFIA

- HTML: HyperText Markup Language | MDN
- CSS: Cascading Style Sheets | MDN
- JavaScript | MDN
- Document Object Model (DOM) | MDN
- W3C HTML Standard
- W3C CSS Current Work
- Libros y cursos de programación web