Guía Maestra de Desarrollo Web

HTML5 CSS3 JS

y Manipulación del DOM con JavaScript

Alejandro G Vera

Guía Maestra de Desarrollo Web: HTML5, CSS3 y Manipulación del DOM con JavaScript

Alejandro G Vera

Introducción

Propósito del Informe

Este documento presenta una guía definitiva y exhaustiva de las tecnologías fundamentales que constituyen el núcleo del desarrollo web del lado del cliente: HTML5, CSS3 y JavaScript. Concebido como un recurso de referencia maestro, este informe trasciende la simple sintaxis para explorar en profundidad los principios subyacentes, las mejores prácticas de la industria y las sinergias críticas entre estos tres pilares de la web. El objetivo es proporcionar a los desarrolladores, tanto intermedios como experimentados, un compendio detallado que consolide el conocimiento fundamental y avanzado necesario para construir aplicaciones web modernas, robustas, accesibles y de alto rendimiento.

Estructura del Informe

El informe está meticulosamente estructurado en tres partes interconectadas, cada una dedicada a una de las tecnologías centrales.

- **Parte I: HTML5 – La Estructura Semántica de la Web.** Esta sección establece el Lenguaje de Marcado de Hipertexto no solo como un medio para estructurar contenido, sino como el fundamento esencial para la accesibilidad, la optimización para motores de búsqueda (SEO) y la interoperabilidad en el ecosistema digital contemporáneo.
- **Parte II: CSS3 – El Arte de la Presentación Web.** A continuación, el informe explora las Hojas de Estilo en Cascada, detallando el lenguaje que dota de diseño, layout y estética a los documentos web. Se abordarán desde los fundamentos de la cascada y la especificidad hasta los módulos de layout más avanzados como Flexbox y Grid, y las técnicas para el diseño responsivo.
- **Parte III: JavaScript – Introducción a la Manipulación del DOM.** La sección

final ofrece una introducción concisa pero potente a la manipulación del Modelo de Objetos del Documento (DOM) utilizando JavaScript. Se enfoca en las operaciones esenciales que permiten transformar una página estática en una experiencia de usuario dinámica e interactiva, sentando las bases para la programación web avanzada.

Parte I: HTML5 – La Estructura Semántica de la Web

Esta primera parte del informe se dedica a HTML5, analizando su rol no solo como un lenguaje de marcado para definir la estructura de una página web, sino como el pilar sobre el cual se construyen la accesibilidad, el SEO y la coherencia semántica en la web moderna. Se explorarán desde los componentes más básicos de un documento hasta las etiquetas semánticas avanzadas y las potentes capacidades de los formularios de HTML5.

1. Fundamentos de un Documento HTML5

La base de cualquier página web es su estructura fundamental, definida por un conjunto de elementos y declaraciones que son esenciales para que los navegadores interpreten y rendericen el contenido correctamente.

Anatomía de un Documento HTML5

Un documento HTML5 válido se compone de varios elementos clave que forman su esqueleto.[1]

- **<!DOCTYPE html>:** Esta no es una etiqueta HTML, sino una declaración de tipo de documento (DTD). Su propósito es instruir al navegador para que renderice la página en "modo estándar" (standard mode), lo que garantiza la adhesión a las especificaciones web actuales de HTML y CSS. Su omisión o una declaración incorrecta pueden hacer que el navegador entre en "modo peculiar" (quirks mode), un modo de compatibilidad con versiones anteriores que emula errores de

renderizado de navegadores antiguos, como Internet Explorer 5.[1] La presencia de <!DOCTYPE html> es, por tanto, el interruptor que asegura un comportamiento de renderizado predecible y estandarizado, especialmente en lo que respecta al modelo de caja de CSS, un prerrequisito para cualquier diseño web moderno.

- **Elemento Raíz <html>:** Este elemento actúa como el contenedor principal para todo el contenido de la página. Todos los demás elementos deben ser descendientes de esta etiqueta. Es una práctica recomendada y crucial para la accesibilidad y el SEO incluir el atributo lang para especificar el idioma del documento, como en <html lang="es">.[1]
- **Elemento <head>:** Este elemento es un contenedor para los metadatos de la página, es decir, información sobre el documento que no se muestra directamente en el contenido visible. Esto incluye el título de la página, enlaces a hojas de estilo (CSS), scripts (JavaScript), declaraciones de juego de caracteres y otros metadatos cruciales para los navegadores y los motores de búsqueda. Solo puede existir un elemento <head> por documento.[1]
    - **<meta charset="UTF-8">:** Esta metaetiqueta es fundamental. Declara el juego de caracteres del documento como UTF-8, un estándar universal que incluye caracteres para casi todos los idiomas humanos. Su inclusión previene problemas de codificación y asegura que todo el texto se muestre correctamente.[1]
    - **<meta name="viewport" content="width=device-width, initial-scale=1.0">:** Esta etiqueta es crítica para el diseño web responsivo. Indica al navegador cómo controlar las dimensiones y la escala de la página, asegurando que el layout se adapte correctamente a diferentes tamaños de pantalla, como los de los dispositivos móviles.[3]
    - **<title>:** Define el título del documento que se muestra en la pestaña o la barra de título del navegador y se utiliza en los resultados de búsqueda y marcadores. Es un elemento obligatorio dentro de <head> (a menos que el título sea proporcionado por un protocolo de nivel superior) y tiene un impacto significativo en el SEO y la usabilidad.[1]
    - **<link>:** Se utiliza para establecer relaciones con recursos externos, siendo su uso más común el de enlazar hojas de estilo CSS externas.[4]
    - **<script> y <style>:** Aunque a menudo se colocan en otros lugares por razones de rendimiento, estas etiquetas también pueden residir en el <head> para definir scripts y estilos internos, respectivamente.
- **Elemento <body>:** Este elemento contiene todo el contenido visible de la página web que se presenta al usuario final, incluyendo texto, imágenes, videos, formularios y cualquier otro elemento interactivo. Solo puede haber un elemento <body> por documento, y debe ser el segundo hijo del elemento <html>, después

del <head>.[1]

2. El Poder de la Semántica: Estructurando el Contenido con Significado

El desarrollo de HTML5 marcó un cambio de paradigma desde diseños basados en contenedores genéricos como (un problema a menudo denominado "divitis") hacia un marcado con significado intrínseco. Las etiquetas semánticas no solo estructuran el documento, sino que describen la función y la jerarquía de su contenido, proporcionando un contexto vital para los navegadores, las tecnologías de asistencia y los motores de búsqueda.[6]

El uso de etiquetas semánticas no es una mera convención estilística; funciona como un **contrato de API implícito** para agentes de usuario no humanos.

1. Para una **tecnología de asistencia**, como un lector de pantalla, la estructura semántica es la principal forma de navegar por el contenido. Un lector de pantalla no "ve" un diseño visual, sino que interpreta el árbol del DOM. Una estructura bien definida con <header>, <nav>, <main> y <footer> permite al lector de pantalla ofrecer atajos funcionales al usuario, como "saltar a la navegación principal" o "ir al contenido principal".[6] Al usar
<nav>, el desarrollador le promete al lector de pantalla que ese bloque contiene la navegación principal del sitio. Romper esta promesa con un marcado no semántico degrada la accesibilidad.
2. Para un **motor de búsqueda**, los crawlers utilizan esta misma estructura semántica para comprender la arquitectura de la información de una página. El contenido dentro de un <main> se considera el núcleo del documento, mientras que el de un <article> se identifica como una pieza de contenido autónoma y distribuible. La jerarquía de encabezados (un único <h1> por página, seguido de <h2>, <h3>, etc.) informa al crawler sobre la organización y la importancia relativa de las secciones de contenido.[8]

El uso incorrecto de las etiquetas, como emplear <table> para el layout o <blockquote> simplemente para indentar texto, rompe este contrato semántico, perjudicando tanto la accesibilidad como el SEO y obligando a menudo al uso de parches como los atributos ARIA (Accessible Rich Internet Applications) para reparar una semántica que debería haber sido correcta desde el principio.[8]

Análisis Profundo de los Elementos Estructurales

A continuación se detallan los principales elementos estructurales de HTML5.[6]

- **<header>:** Representa un contenedor para contenido introductorio o un conjunto de enlaces de navegación. Puede funcionar como el encabezado global de la página (cuando es hijo directo de <body>) o como el encabezado específico de una sección o artículo (cuando es hijo de <article> o <section>). No debe confundirse con las etiquetas de encabezado de texto <h1>-<h6>.[6]
- **<nav>:** Define una sección cuyo propósito es proporcionar enlaces de navegación. Aunque puede haber múltiples elementos <nav> en una página (por ejemplo, para la navegación del sitio y para enlaces internos de la página), se utiliza principalmente para la navegación principal. Su ubicación contextualiza su propósito; un <nav> dentro de un <footer> suele contener enlaces secundarios como "Política de Privacidad" o "Contacto".[6]
- **<main>:** Encapsula el contenido principal o dominante del <body> de un documento. El contenido dentro de <main> debe ser único para ese documento y no debe repetirse en otras páginas del sitio, como barras laterales, navegación o pies de página. Solo debe haber un elemento <main> por página, y se recomienda que sea un hijo directo de <body>.[6]
- **<article> vs. <section> - La Nuance Clave:** La distinción entre estas dos etiquetas es fundamental para un marcado semántico correcto.
    - **<article>:** Se utiliza para contenido que es independiente, autocontenido y, crucialmente, distribuible (syndicatable). Piense en ello como una unidad de contenido que tendría sentido por sí sola si se publicara en un feed RSS, como una entrada de blog, un artículo de noticias, un comentario de un foro o un widget interactivo.[6]
    - **<section>:** Se utiliza para agrupar contenido que está temáticamente relacionado. A diferencia de un <article>, una <section> no es necesariamente autocontenida. Es una buena práctica que cada <section> comience con una etiqueta de encabezado (<h2>-<h6>) que identifique su tema. Un <article> puede contener varias <section>s, y una <section> puede contener varios <article>s, dependiendo de la lógica del contenido.[6]
- **<aside>:** Representa una sección de una página que consiste en contenido que solo está tangencialmente relacionado con el contenido principal del documento. A menudo se presenta como barras laterales o call-outs, y puede contener biografías de autor, enlaces relacionados, o publicidad.[6]

- **<footer>:** Define el pie de página de su ancestro de seccionamiento más cercano (como <article>, <section> o <body>). Un pie de página típicamente contiene información sobre su sección, como el autor, enlaces a documentos relacionados, datos de derechos de autor o información de contacto.[6]

Ejemplo de Maquetación Semántica

El siguiente código ilustra cómo estos elementos trabajan juntos para crear una estructura de página lógica y significativa.[7]

HTML

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Ejemplo de Maquetación Semántica</title>
</head>
<body>
    <header>
        <h1>Mi Sitio Web de Noticias Tecnológicas</h1>
        <nav>
            <ul>
                <li><a href="/inicio">Inicio</a></li>
                <li><a href="/noticias">Noticias</a></li>
                <li><a href="/analisis">Análisis</a></li>
            </ul>
        </nav>
    </header>

    <main>
        <article>
            <header>
                <h2>El Futuro de la Inteligencia Artificial</h2>
                <p>Publicado por: Jane Doe el <time datetime="2025-10-26">26 de Octubre,
```

```
2025</time></p>
        </header>
        <p>El campo de la inteligencia artificial está avanzando a un ritmo sin precedentes, con implicaciones que abarcan todas las industrias...</p>

        <section>
            <h3>Impacto en la Medicina</h3>
            <p>En el sector de la salud, los algoritmos de IA están ayudando a diagnosticar enfermedades con una precisión asombrosa...</p>
        </section>

        <section>
            <h3>Desafíos Éticos</h3>
            <p>A pesar de los avances, persisten importantes desafíos éticos, incluyendo el sesgo en los algoritmos y la privacidad de los datos...</p>
        </section>

        <footer>
            <p>Etiquetas: <a href="/tags/ia">IA</a>, <a href="/tags/tecnologia">Tecnología</a></p>
        </footer>
    </article>

    <aside>
        <h3>Artículos Relacionados</h3>
        <ul>
            <li><a href="/articulo/computacion-cuantica">Introducción a la Computación Cuántica</a></li>
            <li><a href="/articulo/blockchain-mas-alla-cripto">Blockchain más allá de las Criptomonedas</a></li>
        </ul>
    </aside>
</main>

<footer>
    <p>© 2025 Mi Sitio Web de Noticias Tecnológicas. Todos los derechos reservados.</p>
    <nav>
        <a href="/sobre-nosotros">Sobre Nosotros</a> |
        <a href="/contacto">Contacto</a> |
```

```
            <a href="/politica-privacidad">Política de Privacidad</a>
        </nav>
    </footer>
</body>
</html>
```

3. Referencia Completa de Elementos HTML5

HTML5 ofrece un vasto conjunto de elementos para marcar todo tipo de contenido. Esta sección proporciona una referencia exhaustiva, categorizada por función, para facilitar la consulta y el uso correcto de cada etiqueta.[1]

Etiqueta	Descripción	Ejemplo de Código
Raíz y Metadatos		
<html>	Elemento raíz de un documento HTML.	<html lang="es">...</html>
<head>	Contenedor de metadatos del documento.	<head>...</head>
<title>	Título del documento, mostrado en la pestaña del navegador.	<title>Título de la Página</title>
<base>	Especifica la URL base para todas las URLs relativas.	<base href="https://www.ejemplo.com/assets/">
<link>	Enlaza un recurso externo, como una hoja de estilos CSS.	<link rel="stylesheet" href="style.css">
<meta>	Proporciona metadatos como el juego de caracteres o la descripción.	<meta charset="UTF-8">
<style>	Contiene información de estilo (CSS) para un documento.	<style>body { color: blue; }</style>

Seccionamiento y Estructura		
<body>	Contiene el contenido visible de la página.	<body>...</body>
<header>	Encabezado de una página o sección.	<header> Título Principal </header>
<footer>	Pie de página de una página o sección.	<footer> © 2025 </footer>
<nav>	Contiene enlaces de navegación.	<nav> Inicio </nav>
<main>	Contenido principal y único de la página.	<main> Contenido principal. </main>
<article>	Contenido independiente y autocontenido.	<article> Título del Artículo ...</article>
<section>	Agrupa contenido temáticamente relacionado.	<section> Subtema ...</section>
<aside>	Contenido secundario o tangencial.	<aside> Publicidad </aside>
<h1>-<h6>	Encabezados de diferentes niveles de importancia.	Título 1 Título 2
<address>	Información de contacto para el autor/propietario.	<address>Email: ...</address>
Agrupación de Contenido		
	Contenedor genérico sin significado semántico.	...
	Párrafo de texto.	Este es un párrafo.
<hr>	Separador temático (línea horizontal).	Texto. <hr> Otro texto.
<pre>	Texto preformateado, conserva espacios y saltos de	<pre><code>...código...</code></pre>

	línea.	
<blockquote>	Cita larga de otra fuente.	<blockquote> ... </blockquote>
	Lista ordenada (numerada).	Primer ítem
	Lista desordenada (con viñetas).	Ítem
	Elemento de una lista.	Elemento de lista
<dl>	Lista de descripción.	<dl>...</dl>
<dt>	Término en una lista de descripción.	<dt>HTML</dt>
<dd>	Descripción de un término en una lista de descripción.	<dd>Lenguaje de Marcado...</dd>
<figure>	Contenedor para contenido ilustrativo (imágenes, diagramas).	<figure><img src="..."><figcaption>...</figcaption></figure>
<figcaption>	Leyenda o título para un <figure>.	<figcaption>Figura 1: Diagrama.</figcaption>
Semántica de Texto en Línea		
<a>	Hipervínculo.	Enlace
	Contenedor genérico en línea sin significado semántico.	texto
	Énfasis (se muestra típicamente en cursiva).	importante
	Importancia fuerte (se muestra típicamente en negrita).	muy importante
	Texto en negrita sin importancia semántica adicional.	texto
	Texto en cursiva para un tono	texto

	alternativo (término técnico, etc.).	
<u>	Texto subrayado no articulado (ej. para marcar un error ortográfico).	<u>texto</u>
<s>	Texto que ya no es relevante o correcto (tachado).	<s>precio anterior</s>
 	Salto de línea.	Línea 1 Línea 2
<wbr>	Oportunidad de salto de línea opcional.	unapalabramuylarga<wbr>peroseparable
<small>	Texto pequeño (comentarios al margen, letra pequeña).	Copyright...
<sub>	Subíndice.	H₂O
<sup>	Superíndice.	E=mc²
<mark>	Texto resaltado por su relevancia en un contexto.	Resultados: <mark>10 encontrados</mark>.
<time>	Fecha y/o hora legible por máquina.	<time datetime="2025-12-25">Navidad</time>
<data>	Asocia contenido con un valor legible por máquina.	<data value="21053">Producto A</data>
<code>	Fragmento de código de computadora.	<code>const x = 1;</code>
<var>	Variable en una expresión matemática o de programación.	<var>y</var> = <var>m</var><var>x</var> + <var>b</var>
<samp>	Muestra (salida) de un programa de computadora.	<samp>Archivo no encontrado.</samp>
<kbd>	Entrada del usuario (teclado).	Presione <kbd>Ctrl</kbd>+<kbd>S</kbd>

<abbr>	Abreviatura o acrónimo.	<abbr title="Organización Mundial de la Salud">OMS</abbr>
<dfn>	Término de definición.	<dfn>HTML</dfn> es el lenguaje...
<q>	Cita corta en línea.	Él dijo, <q>Hola mundo</q>.
<cite>	Título de una obra creativa (libro, película, etc.).	<cite>El Quijote</cite>
<bdo>	Anula la direccionalidad del texto.	<bdo dir="rtl">Texto</bdo>
<bdi>	Aísla una parte del texto para la direccionalidad.	Usuario <bdi>Pepito</bdi>: 5 posts.
Imagen y Multimedia		
<img>	Imagen.	<img src="imagen.jpg" alt="descripción">
<audio>	Contenido de audio.	<audio controls src="audio.mp3"></audio>
<video>	Contenido de video.	<video controls src="video.mp4"></video>
<source>	Fuente de medios para <audio> o <video>.	<video><source src="v.webm" type="video/webm"></video>
<track>	Pista de texto para <audio> o <video> (subtítulos).	<video><track kind="subtitles" src="sub.vtt"></video>
<svg>	Gráfico vectorial escalable.	<svg width="100" height="100">...</svg>
<canvas>	Lienzo para dibujar gráficos mediante scripting (JS).	<canvas id="miLienzo" width="200" height="100"></canvas>
<map>	Mapa de imagen del lado del cliente.	<map name="mapa"><area...></map>

<area>	Área clickeable dentro de un mapa de imagen.	<area shape="rect" coords="..." href="...">
<picture>	Contenedor para múltiples fuentes de imagen (arte responsivo).	<picture><source...><img...></picture>
Tablas		
<table>	Tabla.	<table>...</table>
<caption>	Título de la tabla.	<caption>Ventas Anuales</caption>
<thead>	Encabezado de la tabla.	<thead><tr><th>Mes</th></tr></thead>
<tbody>	Cuerpo de la tabla.	<tbody><tr><td>Enero</td></tr></tbody>
<tfoot>	Pie de la tabla.	<tfoot><tr><td>Total</td></tr></tfoot>
<tr>	Fila de la tabla.	<tr>...</tr>
<th>	Celda de encabezado.	<th>Nombre</th>
<td>	Celda de datos.	<td>Valor</td>
<colgroup>	Grupo de una o más columnas para formateo.	<colgroup span="2"></colgroup>
<col>	Especifica propiedades para una columna dentro de un <colgroup>.	<col style="background-color: yellow;">
Elementos Interactivos		
<details>	Widget de revelación de información adicional.	<details><summary>Ver más</summary>...</details>
<summary>	Título visible para un elemento <details>.	<summary>Copyright 2025</summary>
<dialog>	Caja de diálogo o ventana modal.	<dialog open>Hola</dialog>

<menu>	Menú de comandos.	<menu> <button>...</button> </menu>
Scripting y Componentes Web		
<script>	Script ejecutable (generalmente JavaScript).	<script src="app.js"></script>
<noscript>	Contenido que se muestra si los scripts están deshabilitados.	<noscript>Por favor, active JavaScript.</noscript>
<template>	Contenedor para contenido HTML que no se renderiza al cargar.	<template>...</template>
<slot>	Placeholder dentro de un componente web (Shadow DOM).	<slot name="nombre-slot"></slot>

4. Formularios Interactivos y Validación en HTML5

HTML5 revolucionó los formularios web al introducir nuevos tipos de entrada y atributos de validación que permiten una validación del lado del cliente nativa, mejorando significativamente la experiencia del usuario (UX), especialmente en dispositivos móviles.[13]

Nuevos Tipos de <input>

Estos nuevos tipos de entrada proporcionan interfaces de usuario especializadas y teclados optimizados en dispositivos móviles, simplificando la entrada de datos para el usuario.[14]

- **email**: Diseñado para direcciones de correo electrónico. Realiza una validación de formato básica y muestra un teclado con la tecla @ en móviles.[14]
- **tel**: Para números de teléfono. No impone un formato específico debido a la variedad internacional, pero a menudo muestra un teclado numérico en móviles.[14]

- **url**: Para direcciones web. Valida que la entrada tenga un formato de URL básico y optimiza el teclado móvil.[14]
- **number**: Para valores numéricos. Muestra controles de incremento/decremento (spinners) y un teclado numérico. Se puede combinar con los atributos min, max y step.[14]
- **range**: Muestra un control deslizante para seleccionar un valor dentro de un rango definido por min y max.[14]
- **search**: Campo de texto estilizado para búsquedas, a menudo con un icono de borrado.[14]
- **Tipos de Fecha y Hora**: date, time, datetime-local, month, week. Muestran selectores de fecha/hora nativos del sistema operativo, lo que estandariza la entrada de datos y mejora la UX.[14]
- **color**: Muestra un selector de color nativo del sistema operativo.[14]

Validación del Lado del Cliente sin JavaScript

HTML5 permite realizar validaciones comunes directamente en el marcado, proporcionando retroalimentación instantánea al usuario sin necesidad de código JavaScript.[13]

- **required**: Este atributo booleano especifica que un campo debe ser completado antes de que el formulario pueda ser enviado. Es la forma más simple de validación.[17]
- **pattern**: Permite especificar una expresión regular (RegExp) contra la cual se valida el valor del campo. Esto ofrece un control de formato extremadamente potente y personalizable.[17]
    - *Ejemplo para un nombre de usuario alfanumérico de 3 a 15 caracteres:* <input type="text" pattern="[a-zA-Z0-9]{3,15}" required>
- **min y max**: Para tipos de entrada numéricos y de fecha, estos atributos definen los valores mínimo y máximo permitidos.[15]
- **step**: Especifica el intervalo legal para los números o fechas. Por ejemplo, step="2" en un campo numérico permitiría solo números pares si min es par.[15]
- **minlength y maxlength**: Definen la longitud mínima y máxima de caracteres para campos de texto.[13]

La Validación HTML5 como una Capa de UX, no de Seguridad

Es crucial entender que la validación del lado del cliente de HTML5 es una característica de **experiencia de usuario (UX)**, no una medida de **seguridad**. Su propósito es guiar a los usuarios legítimos y proporcionarles retroalimentación inmediata, evitando errores comunes y reduciendo peticiones innecesarias al servidor.[14] Sin embargo, un usuario malintencionado puede fácilmente eludir esta validación desactivando JavaScript en su navegador, modificando el HTML en el cliente, o enviando una petición HTTP directamente al servidor.

Por esta razón, **la validación de datos en el lado del servidor es absolutamente obligatoria e ineludible**. El servidor debe siempre verificar y sanitizar todos los datos recibidos antes de procesarlos o almacenarlos. La validación en HTML5 actúa como una primera línea de defensa amigable, pero la seguridad real del formulario y de la aplicación reside en el backend. Esta dualidad —UX en el cliente, seguridad en el servidor— es un principio fundamental del desarrollo web robusto y seguro.

5. Atributos HTML: Configuración Fina de Elementos

Los atributos son valores adicionales que configuran los elementos o ajustan su comportamiento de diversas maneras. Se dividen en atributos globales, que se pueden aplicar a casi cualquier elemento, y atributos específicos de ciertos elementos.[2]

Atributos Globales

Estos atributos son comunes a todos los elementos HTML y proporcionan funcionalidades básicas.[2]

- **id**: Proporciona un identificador único para un elemento en todo el documento. Es fundamental para la selección de elementos con CSS y JavaScript.
- **class**: Asigna una o más clases a un elemento, utilizadas para agrupar elementos con fines de estilo (CSS) o manipulación (JavaScript).
- **style**: Aplica estilos CSS directamente a un elemento (estilo en línea). Su uso se

desaconseja en favor de hojas de estilo externas por razones de mantenibilidad.

- **title**: Proporciona información adicional sobre un elemento, que los navegadores suelen mostrar como un tooltip al pasar el cursor sobre él.
- **lang**: Especifica el idioma del contenido del elemento y sus atributos.
- **contenteditable**: Un atributo booleano que, cuando se establece en true, hace que el contenido del elemento sea editable por el usuario.
- **draggable**: Especifica si un elemento puede ser arrastrado por el usuario.
- **data-***: Permite almacenar datos personalizados privados en la página o aplicación. Estos atributos están diseñados para no interferir con el estilo o el comportamiento del navegador y son accesibles a través de JavaScript para crear experiencias interactivas.

Tabla de Referencia de Atributos Clave

Atributo	Elementos Asociados	Descripción
href	<a>, <area>, <link>	Especifica la URL del recurso al que apunta el enlace.
src	<img>, <audio>, <video>, <script>, <iframe>	Especifica la URL del recurso multimedia o script a incrustar.
alt	<img>, <area>, <input type="image">	Proporciona texto alternativo para una imagen si no se puede mostrar. Crucial para la accesibilidad.
target	<a>, <form>	Especifica dónde abrir el recurso enlazado (ej. _blank para una nueva pestaña).
rel	<a>, <link>	Describe la relación entre el documento actual y el recurso enlazado.
disabled	<button>, <input>, <select>, <textarea>	Deshabilita el control del formulario, impidiendo la interacción del usuario.

readonly	<input>, <textarea>	Hace que un control de formulario sea de solo lectura, pero su valor se envía con el formulario.
checked	<input type="checkbox">, <input type="radio">	Especifica que un control de tipo checkbox o radio debe estar marcado por defecto.
selected	<option>	Especifica que una opción en un menú desplegable (<select>) debe estar seleccionada por defecto.
action	<form>	La URL que procesará los datos del formulario cuando se envíe.
method	<form>	El método HTTP a utilizar al enviar el formulario (generalmente GET o POST).
enctype	<form>	El tipo de codificación de los datos del formulario al enviarlos (ej. multipart/form-data para subir archivos).
async	<script>	Indica que el script se debe ejecutar de forma asíncrona, sin bloquear el renderizado de la página.
defer	<script>	Indica que la ejecución del script se debe aplazar hasta que el análisis del documento haya finalizado.

Parte II: CSS3 – El Arte de la Presentación Web

Esta sección del informe se sumerge en las Hojas de Estilo en Cascada (CSS3), el lenguaje que define la presentación visual de los documentos HTML. Se analizará CSS no como una simple herramienta de decoración, sino como un sistema sofisticado y

potente para crear layouts complejos, diseños responsivos y animaciones dinámicas que definen la experiencia de usuario en la web moderna.

6. Fundamentos de CSS: Aplicación y Sintaxis

Para aplicar estilos a un documento HTML, es fundamental comprender la estructura de una regla CSS y los diferentes métodos para vincularla con el HTML.

Anatomía de una Regla CSS

Toda regla CSS se compone de las siguientes partes [22]:

- **Selector**: Apunta al elemento o elementos HTML a los que se aplicará el estilo. Puede ser simple (como un nombre de etiqueta) o complejo (combinando múltiples selectores).
- **Bloque de Declaración**: Encerrado entre llaves {}, contiene una o más declaraciones de estilo.
- **Declaración**: Un par de propiedad: valor;. La **propiedad** es el atributo de estilo que se desea cambiar (ej. color, font-size), y el **valor** es el ajuste que se le aplica. Cada declaración termina con un punto y coma.

Ejemplo de una regla CSS:

CSS

```
/* El selector 'p.intro' apunta a los elementos <p> que tienen la clase "intro" */
p.intro {
  color: navy; /* Declaración 1 */
  font-size: 1.2em; /* Declaración 2 */
}
```

Métodos de Implementación

Existen tres formas de aplicar CSS a un documento HTML, cada una con sus propios casos de uso y nivel de especificidad.[4]

1. **Hojas de Estilo Externas (External Stylesheets)**: Este es el método más común y recomendado. Los estilos se definen en un archivo .css separado y se enlazan al documento HTML mediante la etiqueta <link> dentro del <head>.
   - **Ventajas**:
     - **Separación de preocupaciones**: Mantiene la estructura (HTML) y la presentación (CSS) en archivos distintos, lo que resulta en un código más limpio y mantenible.
     - **Reutilización y escalabilidad**: Un único archivo CSS puede dar estilo a múltiples páginas de un sitio web, facilitando cambios globales.
     - **Rendimiento**: El archivo CSS puede ser cacheado por el navegador, acelerando la carga de páginas posteriores.
   - **Ejemplo**:

```
HTML
<head>
    <link rel="stylesheet" href="styles.css">
</head>

CSS
/* En el archivo styles.css */
body {
    font-family: Arial, sans-serif;
}
```

2. **Hojas de Estilo Internas (Internal Stylesheets)**: Los estilos se colocan directamente dentro de una etiqueta <style> en el <head> del documento HTML.
   - **Ventajas**: Útil para aplicar estilos que son únicos para una sola página o para prototipar rápidamente sin crear archivos adicionales.
   - **Desventajas**: Los estilos no son reutilizables en otras páginas y aumentan el tamaño del archivo HTML.
   - **Ejemplo**:

```
HTML
<head>
    <style>
        h1 {
            color: teal;
```

```
    }
  </style>
</head>
```

3. **Estilos en Línea (Inline Styles)**: Los estilos se aplican directamente a un elemento HTML individual utilizando el atributo style.
    - **Ventajas**: Útil para aplicar un estilo muy específico a un solo elemento, a menudo manipulado dinámicamente por JavaScript.
    - **Desventajas**: Este método tiene la máxima especificidad, lo que puede dificultar la anulación de estilos. Mezcla contenido y presentación, lo que se considera una mala práctica para el mantenimiento y la escalabilidad.
    - **Ejemplo**:
      HTML
      ```
      <p style="color: red; font-weight: bold;">Este texto es importante.</p>
      ```

La jerarquía de aplicación de estos métodos es clara: los estilos en línea anulan a los estilos internos y externos. Entre los estilos internos y externos, el resultado final depende de la cascada, la especificidad y el orden de la fuente, conceptos que se explorarán a continuación.[23]

7. La Cascada, la Especificidad y la Herencia

CSS (Cascading Style Sheets) debe su nombre a la "cascada", un algoritmo fundamental que los navegadores utilizan para resolver conflictos cuando múltiples reglas de estilo apuntan al mismo elemento. Comprender este sistema es esencial para predecir y depurar el comportamiento de CSS.[24]

La Cascada

El algoritmo de la cascada determina qué declaración de estilo se aplica a un elemento basándose en tres factores principales, evaluados en el siguiente orden de precedencia [24]:

1. **Origen e Importancia**: Las declaraciones se procesan en el siguiente orden de

prioridad:
    - Transiciones y animaciones CSS.
    - Declaraciones marcadas con !important en las hojas de estilo del autor (desarrollador).
    - Declaraciones normales en las hojas de estilo del autor.
    - Declaraciones en las hojas de estilo del agente de usuario (estilos por defecto del navegador).
      El uso de !important debe ser un último recurso, ya que rompe el flujo natural de la cascada y hace que el CSS sea extremadamente difícil de depurar y mantener.24
2. **Especificidad**: Si dos declaraciones tienen la misma importancia, se aplica la que proviene de una regla con un selector más "específico" o "pesado". La especificidad es un valor numérico que se calcula para cada selector.[24]
3. **Orden de Fuente**: Si dos declaraciones tienen la misma importancia y especificidad, la que aparece más tarde en el código fuente (o en la última hoja de estilo importada) es la que "gana" y se aplica.[24]

Cálculo de la Especificidad

La especificidad no es solo un mecanismo de resolución; es una medida de la complejidad y fragilidad de una hoja de estilos. Una regla con una especificidad muy alta, como #sidebar.user-profile a:hover, es difícil de anular. Para hacerlo, se requiere una regla aún más específica, lo que puede llevar a una "guerra de especificidad" y a selectores cada vez más largos, frágiles y difíciles de mantener. El uso excesivo de !important es a menudo un síntoma de que la especificidad se ha vuelto inmanejable.

Por esta razón, mantener una **baja especificidad** es un objetivo arquitectónico clave en CSS moderno. Metodologías como BEM (Block, Element, Modifier) se basan en el uso de selectores de clase planos y únicos para mantener la especificidad baja y predecible. Gestionar la especificidad no es solo una tarea de depuración, sino una estrategia proactiva para crear CSS escalable y evitar la acumulación de deuda técnica.

La especificidad se calcula asignando un peso a cada tipo de selector. Un modelo común para visualizar esto es una puntuación de cuatro partes (A, B, C, D) [24]:

- **A (Estilos en línea)**: 1 si la declaración está en un atributo style, 0 en caso

contrario.

- **B (IDs)**: El número de selectores de ID en la regla (ej. #mi-id).
- **C (Clases, Atributos y Pseudo-clases)**: El número de selectores de clase (.mi-clase), de atributo ([type="text"]) y de pseudo-clase (:hover).
- **D (Elementos y Pseudo-elementos)**: El número de selectores de tipo de elemento (p, div) y de pseudo-elemento (::before).

El selector universal (*) y el combinador (>) no tienen valor de especificidad.

Tipo de Selector	Valor de Especificidad (A,B,C,D)	Ejemplo
Estilo en línea	1,0,0,0	style="..."
ID	0,1,0,0	#header
Clase	0,0,1,0	.btn-primary
Atributo	0,0,1,0	[type="submit"]
Pseudo-clase	0,0,1,0	:hover
Elemento	0,0,0,1	h1
Pseudo-elemento	0,0,0,1	::after

Ejemplo de cálculo: div#main.nav a:hover tiene una especificidad de (0,1,2,2): 1 ID (#main), 2 clases/pseudo-clases (.nav, :hover), y 2 elementos (div, a).

Herencia

La herencia es el mecanismo por el cual ciertas propiedades CSS aplicadas a un elemento padre son heredadas por sus elementos hijos. Propiedades como color, font-family, font-size, y text-align son heredables por defecto. Otras, como background-color, border, y padding, no lo son.[24]

Se puede controlar la herencia explícitamente con las siguientes palabras clave:

- **inherit**: Fuerza a un elemento a heredar el valor de su padre para una propiedad específica.

- **initial**: Restablece una propiedad a su valor inicial definido por el navegador.
- **unset**: Si la propiedad es heredable, se comporta como inherit. Si no lo es, se comporta como initial.

8. El Modelo de Caja (Box Model)

Todo elemento HTML en una página puede ser considerado como una caja rectangular. El modelo de caja de CSS describe cómo se compone esta caja y cómo interactúa con otras en el layout. Es un concepto fundamental para el diseño y la alineación.[27]

Las Cuatro Capas

Cada caja se compone de cuatro áreas concéntricas:

1. **content (Contenido)**: El área central donde se muestra el contenido real del elemento, como texto, imágenes o video. Sus dimensiones se controlan con las propiedades width y height.
2. **padding (Relleno)**: Un espacio transparente que rodea el área de contenido. El padding se encuentra *dentro* del borde y el color de fondo del elemento se extiende a través de él. Se controla con padding-top, padding-right, etc., y el shorthand padding.
3. **border (Borde)**: Una línea que rodea el padding y el content. Se define con propiedades como border-width, border-style, y border-color, o el shorthand border.
4. **margin (Margen)**: Un espacio transparente que rodea el borde. El margin empuja a otros elementos, creando espacio *fuera* de la caja. Se controla con margin-top, margin-right, etc., y el shorthand margin.

box-sizing: border-box

Por defecto, el modelo de caja de CSS es content-box. Esto significa que las

propiedades width y height que se definen para un elemento se aplican solo al área de content. El padding y el border se añaden *adicionalmente* a ese ancho y alto, lo que a menudo complica los cálculos de layout.

Para simplificar esto, se puede utilizar el modelo de caja alternativo border-box.[28]

CSS

```
* {
  box-sizing: border-box;
}
```

Cuando se establece box-sizing: border-box;, las propiedades width y height incluyen el content, el padding y el border. Por ejemplo, si se define width: 200px;, la caja total (contenido + relleno + borde) medirá 200px de ancho, y el área de contenido se encogerá para acomodar el padding y el border. Este modelo es mucho más intuitivo y es una práctica estándar en el desarrollo web moderno.

Colapso de Márgenes

El colapso de márgenes es un comportamiento específico de los márgenes verticales de los elementos de bloque. Cuando dos márgenes verticales adyacentes se encuentran, se "colapsan" en un único margen, cuyo tamaño es el del mayor de los dos márgenes (o el del margen si son iguales). Esto no ocurre con los márgenes horizontales.[27]

El colapso ocurre en tres escenarios principales:

- **Hermanos adyacentes**: El margen inferior de un elemento colapsa con el margen superior de su siguiente hermano.
- **Padre e hijo**: Si no hay border o padding que separe el margen superior de un elemento padre de su primer hijo, sus márgenes superiores colapsarán. Lo mismo ocurre con los márgenes inferiores.
- **Bloques vacíos**: Si un bloque no tiene contenido, padding o border, sus márgenes superior e inferior pueden colapsar entre sí.

9. Selectores CSS: Precisión Quirúrgica para Estilizar

Los selectores son el corazón de CSS, ya que permiten apuntar con precisión a los elementos HTML que se desean estilizar. Dominar los selectores es crucial para escribir CSS eficiente y mantenible.[22]

- **Selectores Simples**:
  - **Selector Universal (*)**: Selecciona todos los elementos.
  - **Selector de Tipo (h1, p)**: Selecciona todos los elementos de un tipo de etiqueta específico.
  - **Selector de Clase (.nombre-clase)**: Selecciona todos los elementos que tienen el atributo class especificado.
  - **Selector de ID (#nombre-id)**: Selecciona el único elemento que tiene el atributo id especificado.
- **Selectores de Atributo**: Seleccionan elementos basados en la presencia o el valor de sus atributos.
  - [target]: Selecciona elementos que tienen el atributo target.
  - [type="submit"]: Selecciona elementos cuyo atributo type es exactamente "submit".
  - [href^="https"]: Selecciona elementos cuyo href comienza con "https".
  - [href$=".pdf"]: Selecciona elementos cuyo href termina con ".pdf".
  - [class*="btn-"]: Selecciona elementos cuya clase contiene la subcadena "btn-".
- **Combinadores**: Crean relaciones entre selectores simples.
  - **Descendiente ( )**: article p selecciona todos los <p> que son descendientes de un <article>.
  - **Hijo Directo (>)**: ul > li selecciona solo los <li> que son hijos directos de un <ul>.
  - **Hermano Adyacente (+)**: h2 + p selecciona el primer <p> que sigue inmediatamente a un <h2> (y comparte el mismo padre).
  - **Hermano General (~)**: h2 ~ p selecciona todos los <p> que siguen a un <h2> (y comparten el mismo padre).
- **Pseudo-clases**: Seleccionan elementos basados en su estado o posición en el DOM.
  - **De Estado de UI**: :hover, :focus (cuando un elemento recibe foco), :active (mientras se hace clic), :checked (para checkboxes/radios), :disabled,

:required, :valid, :invalid (para validación de formularios).
    - **Estructurales**: :first-child, :last-child, :nth-child(n) (ej. :nth-child(2n) para elementos pares), :nth-of-type(n), :only-child, :not(.clase-excluida).
- **Pseudo-elementos**: Permiten estilizar partes específicas de un elemento que no están representadas por una etiqueta HTML.
    - ::before y ::after: Crean pseudo-elementos antes y después del contenido de un elemento, respectivamente. Se usan con la propiedad content para añadir elementos decorativos o texto.
    - ::first-letter: Estiliza la primera letra de un bloque de texto.
    - ::first-line: Estiliza la primera línea de un bloque de texto.
    - ::marker: Estiliza la viñeta o el número de un elemento de lista <li>.
    - ::selection: Estiliza la porción de texto que el usuario ha seleccionado.

10. Layout Moderno: Flexbox y Grid

CSS3 introdujo dos potentes módulos de layout, Flexbox y Grid, que han revolucionado la forma en que se construyen los diseños web, eliminando la necesidad de hacks y soluciones complejas.

Flexbox (Layout Unidimensional)

El Módulo de Caja Flexible (Flexible Box Layout), o Flexbox, está diseñado para el layout en **una sola dimensión**, ya sea una fila o una columna. Es ideal para alinear y distribuir el espacio entre ítems dentro de un contenedor, como menús de navegación, barras de herramientas o conjuntos de tarjetas.[35]

- **Conceptos Clave**: El layout se organiza a lo largo de un **eje principal** (main axis) y un **eje transversal** (cross axis). La dirección de estos ejes se define con flex-direction.
- **Propiedades del Contenedor (Padre)**:
    - display: flex;: Activa el contexto de formato flex para los hijos directos.
    - flex-direction: Define la dirección del eje principal (row, row-reverse, column, column-reverse).
    - flex-wrap: Permite que los ítems se envuelvan en múltiples líneas (nowrap,

wrap, wrap-reverse).
    - justify-content: Alinea los ítems a lo largo del eje principal (flex-start, center, flex-end, space-between, space-around).
    - align-items: Alinea los ítems a lo largo del eje transversal (stretch, flex-start, center, flex-end, baseline).
    - align-content: Alinea las líneas de ítems (cuando hay flex-wrap: wrap) dentro del contenedor.
    - gap: Define el espacio entre los ítems.
- **Propiedades de los Ítems (Hijos)**:
    - flex-grow: Define la capacidad de un ítem para crecer y ocupar el espacio disponible.
    - flex-shrink: Define la capacidad de un ítem para encogerse si no hay suficiente espacio.
    - flex-basis: Define el tamaño inicial de un ítem antes de que se distribuya el espacio.
    - flex: Es el shorthand para flex-grow, flex-shrink y flex-basis.
    - order: Cambia el orden visual de los ítems.
    - align-self: Permite anular el align-items para un ítem individual.

Grid (Layout Bidimensional)

El Módulo de Layout de Cuadrícula (Grid Layout) está diseñado para el layout en **dos dimensiones**: filas y columnas simultáneamente. Es la herramienta más potente para crear layouts de página complejos y estructurados, como la maquetación de una página web completa con encabezado, pie de página, contenido principal y barras laterales.[37]

- **Conceptos Clave**: El layout se basa en una cuadrícula de **pistas** (filas y columnas), **líneas** de cuadrícula, **celdas** y **áreas** nombradas.
- **Propiedades del Contenedor**:
    - display: grid;: Activa el contexto de formato de cuadrícula.
    - grid-template-columns y grid-template-rows: Definen el número y el tamaño de las columnas y filas. Se puede usar la unidad fr (fracción) para una distribución flexible del espacio.
    - grid-template-areas: Permite definir un layout visualmente nombrando las áreas de la cuadrícula.
    - gap (o grid-gap): Define el espacio entre las pistas de la cuadrícula.

    - justify-items y align-items: Alinean los ítems dentro de sus celdas de cuadrícula en los ejes horizontal y vertical, respectivamente.
- **Propiedades de los Ítems**:
    - grid-column-start, grid-column-end, grid-row-start, grid-row-end: Definen en qué líneas de la cuadrícula comienza y termina un ítem. Se pueden abreviar con grid-column y grid-row.
    - grid-area: Asigna un ítem a un área nombrada definida en grid-template-areas, o funciona como un shorthand para las cuatro propiedades anteriores.
    - justify-self y align-self: Permiten anular la alineación para un ítem individual dentro de su celda.

Tabla Comparativa: Flexbox vs. Grid

Característica / Caso de Uso	Flexbox (Unidimensional)	Grid (Bidimensional)
Dimensionalidad	Optimizado para layout en una sola dirección (fila o columna).	Diseñado para layout en dos direcciones (filas y columnas a la vez).
Enfoque Principal	**Alineación de contenido**: Distribuye el espacio dentro de una línea.	**Layout de estructura**: Define una cuadrícula rígida en la que se colocan los elementos.
Flexibilidad de Ítems	Los ítems tienen flexibilidad inherente para crecer y encogerse a lo largo del eje principal.	Los ítems se colocan en pistas de cuadrícula definidas, pero pueden abarcar múltiples pistas.
Casos de Uso Ideales	Componentes de UI, menús de navegación, alineación de elementos en un formulario, galerías de tarjetas.	Layouts de página completos, estructuras de aplicaciones complejas, cualquier diseño que requiera alineación estricta en filas y columnas.
Complejidad	Más simple de aprender para layouts unidimensionales.	Más potente y complejo, con más propiedades para controlar el layout bidimensional.

En resumen, la regla general es: usar Flexbox para alinear contenido y Grid para definir el layout.[38] A menudo, se utilizan juntos: Grid para la estructura principal de la página y Flexbox para alinear los componentes dentro de las áreas de la cuadrícula.

11. Diseño Responsivo y Adaptativo

El diseño responsivo (Responsive Web Design, RWD) es una aproximación al diseño web que busca que las páginas se vean bien en una variedad de dispositivos y tamaños de pantalla. CSS3 proporciona las herramientas clave para lograrlo: Media Queries y Variables CSS.

Media Queries

Las Media Queries permiten aplicar bloques de estilos CSS solo cuando se cumplen ciertas condiciones sobre el dispositivo o el viewport, como su ancho, altura u orientación.[39]

- **Sintaxis**: Una media query se define con la regla @media, seguida de un tipo de medio opcional (como screen o print) y una o más expresiones de características de medios entre paréntesis.

```CSS
@media screen and (min-width: 768px) {
 /* Estilos que se aplican solo si el ancho del viewport es 768px o más */
.container {
   display: flex;
 }
}
```

- **Características de Medios Comunes**:
  - width, min-width, max-width: Basadas en el ancho del viewport. Son las más utilizadas para RWD.
  - height, min-height, max-height: Basadas en la altura del viewport.
  - orientation: portrait (vertical) o landscape (horizontal).
  - prefers-color-scheme: Detecta si el usuario prefiere un tema claro (light) u oscuro (dark) en su sistema operativo.

- **Operadores Lógicos**:
  - and: Combina múltiples características.
  - not: Niega una media query.
  - , (coma): Actúa como un operador OR, permitiendo aplicar los mismos estilos a múltiples queries.

Variables CSS (Propiedades Personalizadas)

Las variables CSS, también conocidas como propiedades personalizadas, permiten definir valores reutilizables en una hoja de estilos. Esto centraliza los valores de diseño (como colores, tamaños de fuente o espaciado) y facilita enormemente el mantenimiento y la creación de temas.[42]

- **Declaración**: Las variables se declaran con un prefijo de dos guiones (--) y generalmente se definen en el pseudo-selector :root para que tengan un alcance global.

```CSS
:root {
  --primary-color: #3498db;
  --base-font-size: 16px;
}
```

- **Uso**: Se accede al valor de una variable utilizando la función var(). Se puede proporcionar un valor de respaldo opcional.

```CSS
body {
  color: var(--primary-color);
  font-size: var(--base-font-size);
}
a {
  color: var(--primary-color, blue); /* 'blue' es el valor de respaldo */
}
```

Estrategia Avanzada: Variables CSS + Media Queries

La verdadera potencia se desata al combinar estas dos características. Se pueden redefinir los valores de las variables CSS dentro de las media queries. Esto permite crear diseños que no solo cambian de layout, sino que ajustan su escala, espaciado y temas de manera fluida y mantenible.[43]

- **Ejemplo de Tipografía Responsiva**:

```CSS
:root {
  --font-size-base: 1rem; /* 16px por defecto */
  --h1-size: 2.5rem;
}

/* En pantallas más grandes, simplemente aumentamos el tamaño base */
@media (min-width: 900px) {
  :root {
    --font-size-base: 1.125rem; /* 18px */
  }
}

body {
  font-size: var(--font-size-base);
}

h1 {
  font-size: var(--h1-size);
}
```

- **Ejemplo de Theming (Modo Claro/Oscuro)**:

```CSS
:root {
  --background-color: #ffffff;
  --text-color: #333333;
  --link-color: #007bff;
}

@media (prefers-color-scheme: dark) {
  :root {
    --background-color: #121212;
```

```
    --text-color: #eeeeee;
    --link-color: #64b5f6;
  }
}

body {
  background-color: var(--background-color);
  color: var(--text-color);
  transition: background-color 0.3s ease, color 0.3s ease;
}

a {
  color: var(--link-color);
}
```

Las Variables CSS como Puente entre CSS y JavaScript

Las variables CSS no son solo para la mantenibilidad dentro de CSS. Son dinámicamente accesibles y modificables desde JavaScript a través de element.style.setProperty('--mi-variable', 'nuevo-valor').[42] Esto establece un poderoso puente de comunicación bidireccional. CSS puede definir los estilos por defecto y las reglas responsivas, mientras que JavaScript puede anular estos valores en tiempo real en respuesta a la interacción del usuario (por ejemplo, un interruptor manual de tema claro/oscuro que anule la preferencia del sistema). Esta sinergia permite la creación de sistemas de diseño complejos y reactivos que serían mucho más difíciles de gestionar solo con la manipulación de clases CSS o estilos en línea desde JavaScript.

12. Efectos Visuales y Animaciones

CSS3 proporciona un conjunto nativo de herramientas para crear efectos visuales y animaciones sofisticadas que mejoran la experiencia del usuario sin depender de JavaScript.

Transformaciones 2D y 3D

La propiedad transform permite modificar el sistema de coordenadas de un elemento, lo que permite moverlo, rotarlo, escalarlo o inclinarlo sin afectar el flujo del documento (es decir, sin desplazar a otros elementos).[48]

- **translate(x, y)**: Mueve un elemento en los ejes X e Y. translateX() y translateY() son variantes.
- **scale(x, y)**: Aumenta o disminuye el tamaño de un elemento. scale(1.5) lo haría un 50% más grande.
- **rotate(angle)**: Gira un elemento un ángulo especificado (ej. 45deg).
- **skew(x-angle, y-angle)**: Inclina un elemento a lo largo de los ejes X e Y.
- Las transformaciones 3D, como rotate3d() o translate3d(), añaden profundidad y son a menudo aceleradas por hardware (GPU), lo que resulta en animaciones más fluidas.

Transiciones

Las transiciones permiten animar suavemente el cambio de valor de una propiedad CSS a lo largo de un período de tiempo determinado. Son ideales para crear efectos de retroalimentación en interacciones del usuario, como :hover.[48]

- La propiedad transition es un shorthand para:
  - transition-property: La propiedad que se animará (ej. background-color).
  - transition-duration: La duración de la animación (ej. 0.3s).
  - transition-timing-function: La curva de aceleración de la animación (ease, linear, ease-in-out, etc.).
  - transition-delay: Un retraso antes de que comience la animación.
- **Ejemplo**:

```
.button {
background-color: blue;
transition: background-color 0.3s ease-in-out;
}
.button:hover {
background-color: darkblue;
```

}
```

### Animaciones con @keyframes

Para animaciones más complejas que involucran múltiples pasos o propiedades, se utiliza la regla @keyframes. Esta regla permite definir una secuencia de "fotogramas clave" por los que pasará la animación.[48]

- Se define una animación con @keyframes y se le da un nombre. Dentro, se especifican los estilos en diferentes puntos porcentuales de la animación (de 0% o from a 100% o to).
- Luego, se aplica la animación a un elemento usando la propiedad animation y sus sub-propiedades (animation-name, animation-duration, animation-iteration-count, etc.).
- **Ejemplo de un efecto de pulso**:

```CSS
@keyframes pulse {
  0% {
    transform: scale(1);
    opacity: 1;
  }
  50% {
    transform: scale(1.1);
    opacity: 0.7;
  }
  100% {
    transform: scale(1);
    opacity: 1;
  }
}
```

```
.live-indicator {
animation: pulse 2s infinite;
}
```
```

Parte III: JavaScript – Introducción a la Manipulación del DOM

Esta sección final se centra en las operaciones más comunes y esenciales de JavaScript para interactuar con el Modelo de Objetos del Documento (DOM). La manipulación del DOM es lo que permite que una página web pase de ser un documento estático a una aplicación web dinámica e interactiva.

13. Selección de Elementos del DOM

Antes de poder manipular cualquier parte de una página, primero se debe seleccionar el elemento o los elementos deseados. JavaScript proporciona varios métodos para este fin, cada uno con sus propias características y casos de uso.[50]

Métodos de Selección

A continuación se describen los métodos más importantes para seleccionar nodos del DOM.[53]

- **document.getElementById('id')**: Selecciona un único elemento por su atributo id. Dado que los IDs deben ser únicos en un documento, este método es el más rápido y directo para apuntar a un elemento específico. Devuelve el objeto del elemento o null si no se encuentra.
- **document.getElementsByTagName('tag')**: Selecciona todos los elementos que coinciden con un nombre de etiqueta dado (ej. 'p', 'div'). Devuelve una HTMLCollection de los elementos encontrados.
- **document.getElementsByClassName('class')**: Selecciona todos los elementos que contienen una clase específica. También devuelve una HTMLCollection.
- **document.querySelector('selector')**: Este es un método moderno y muy versátil. Selecciona el **primer** elemento del documento que coincide con un selector CSS especificado. Puede usar cualquier selector CSS válido (ID, clase, atributo, combinadores, etc.). Devuelve el primer elemento encontrado o null.

- **document.querySelectorAll('selector')**: Similar a querySelector, pero selecciona **todos** los elementos que coinciden con el selector CSS y los devuelve como una NodeList estática.

La Diferencia Crítica entre HTMLCollection (Viva) y NodeList (Estática)

Una de las distinciones más importantes y una fuente común de errores es la diferencia entre las colecciones devueltas por los métodos getElementsBy... y querySelectorAll.[53]

- **HTMLCollection (Viva)**: Las colecciones devueltas por getElementsByTagName y getElementsByClassName son "vivas". Esto significa que se actualizan automáticamente en tiempo real si el DOM cambia. Por ejemplo, si se obtiene una colección de elementos con una clase y luego se añade un nuevo elemento con esa misma clase al DOM, la HTMLCollection reflejará inmediatamente ese nuevo elemento.
- **NodeList (Estática)**: La NodeList devuelta por querySelectorAll es "estática". Es una instantánea de los elementos que coincidían con el selector en el momento en que se llamó al método. No se actualizará si se añaden o eliminan elementos del DOM posteriormente.

Este comportamiento tiene implicaciones prácticas. Si se itera sobre una HTMLCollection viva mientras se eliminan elementos de esa misma colección (por ejemplo, eliminando todos los elementos con una clase específica), es probable que se salten elementos, ya que la longitud de la colección y los índices de los elementos cambian con cada eliminación. Para operaciones de modificación del DOM dentro de un bucle, es más seguro y predecible usar una NodeList estática de querySelectorAll o convertir la HTMLCollection en un array estático antes de iterar, utilizando Array.from().

Tabla Comparativa de Selectores JS

Método	Argumento	Valor de	Tipo de	Viva/Estática	Caso de Uso

		Retorno	Colección		Principal
getElementById()	string (ID)	Elemento único o null	N/A	N/A	Seleccionar un elemento único y específico.
getElementsByTagName()	string (tag)	HTMLCollection	HTMLCollection	Viva	Seleccionar todos los elementos de un tipo (ej. todos los).
getElementsByClassName()	string (clase)	HTMLCollection	HTMLCollection	Viva	Seleccionar todos los elementos que comparten una clase.
querySelector()	string (selector CSS)	Elemento único o null	N/A	N/A	Seleccionar el primer elemento que coincida con un selector CSS complejo.
querySelectorAll()	string (selector CSS)	NodeList	NodeList	Estática	Seleccionar todos los elementos que coincidan con un selector CSS complejo.

14. Modificación Dinámica de la Página

Una vez seleccionado un elemento, se puede modificar su contenido, atributos, clases y estilos para crear una experiencia dinámica.

Modificar Contenido

Existen dos propiedades principales para cambiar el contenido de un elemento [61]:

- **element.textContent**: Obtiene o establece el contenido de texto de un elemento y todos sus descendientes. Cualquier marcado HTML se interpreta como texto literal. Es la opción más segura y, a menudo, la de mejor rendimiento para manipular solo texto.
  JavaScript
  ```
  const titulo = document.getElementById('titulo-principal');
  titulo.textContent = 'Bienvenido al Nuevo Sitio!';
  ```

- **element.innerHTML**: Obtiene o establece el contenido HTML de un elemento. El navegador interpreta la cadena asignada como HTML, creando los nodos del DOM correspondientes. Es muy potente, pero debe usarse con precaución, ya que puede exponer el sitio a vulnerabilidades de seguridad como Cross-Site Scripting (XSS) si se inserta contenido no sanitizado proveniente del usuario.
  JavaScript
  ```
  const contenedor = document.getElementById('contenido');
  contenedor.innerHTML = '<h2>Nuevo Título</h2><p>Este es un párrafo generado dinámicamente.</p>';
  ```

Modificar Atributos

Se pueden leer y cambiar los atributos de un elemento de varias maneras [62]:

- **getAttribute() y setAttribute()**:
  - element.getAttribute('nombre-del-atributo'): Devuelve el valor del atributo especificado.
  - element.setAttribute('nombre-del-atributo', 'nuevo-valor'): Establece o cambia el valor de un atributo.

  JavaScript
  ```
  const enlace = document.querySelector('a');
  enlace.setAttribute('href', 'https://www.nueva-url.com');
  ```

```
enlace.setAttribute('target', '_blank');
```

- **Acceso directo a propiedades**: Muchos atributos comunes (como id, src, href, value) son accesibles directamente como propiedades del objeto del elemento.
  JavaScript
  ```
  const imagen = document.getElementById('logo');
  imagen.src = 'nuevo-logo.png';
  imagen.alt = 'Nuevo logo de la empresa';
  ```

Modificar Clases

La forma moderna y recomendada de manipular las clases de un elemento es a través de la propiedad classList, que proporciona un conjunto de métodos convenientes.[65]

- **element.classList.add('clase1', 'clase2')**: Añade una o más clases al elemento.
- **element.classList.remove('clase')**: Elimina una clase del elemento.
- **element.classList.toggle('clase')**: Añade la clase si no está presente, y la elimina si sí lo está. Es ideal para interruptores de estado.
- **element.classList.replace('clase-antigua', 'clase-nueva')**: Reemplaza una clase existente por una nueva.
- **element.classList.contains('clase')**: Devuelve true o false dependiendo de si el elemento tiene la clase especificada.

Ejemplo de uso de toggle:

JavaScript

```
const menu = document.getElementById('menu-movil');
const boton = document.getElementById('boton-menu');

boton.addEventListener('click', () => {
  menu.classList.toggle('abierto');
});
```

Modificar Estilos

Los estilos en línea de un elemento se pueden modificar a través de la propiedad style. Esta es una forma directa de cambiar la apariencia de un elemento, aunque para cambios de estado complejos, es preferible manipular clases.[47]

- **Importante**: Las propiedades CSS que usan guiones (kebab-case), como background-color o font-size, deben convertirse a camelCase en JavaScript: backgroundColor, fontSize.

```JavaScript
const alerta = document.getElementById('mensaje-alerta');
alerta.style.backgroundColor = 'lightcoral';
alerta.style.color = 'white';
alerta.style.padding = '1em';
alerta.style.borderRadius = '5px';
```

15. Creación y Eliminación de Nodos

JavaScript permite crear nuevos elementos HTML desde cero y añadirlos o eliminarlos del DOM, lo que es la base de las aplicaciones web dinámicas.

Creación e Inserción de Elementos

El proceso generalmente implica dos pasos: crear el elemento en memoria y luego insertarlo en el documento.[50]

1. **document.createElement('tag')**: Crea un nuevo nodo de elemento del tipo de etiqueta especificado. Este elemento existe en memoria pero aún no es parte del DOM visible.
2. **padre.appendChild(hijo)**: Añade el elemento hijo como el último hijo del elemento padre, haciéndolo visible en la página.

3. **padre.insertBefore(nuevoNodo, nodoDeReferencia)**: Inserta nuevoNodo en la lista de hijos de padre, justo antes de nodoDeReferencia.

Ejemplo de creación de un nuevo elemento de lista:

JavaScript

```
// Seleccionar el contenedor <ul>
const listaTareas = document.getElementById('lista-de-tareas');

// Crear un nuevo elemento <li>
const nuevaTarea = document.createElement('li');

// Añadir contenido al nuevo elemento
nuevaTarea.textContent = 'Aprender manipulación del DOM';

// Añadir el nuevo <li> a la <ul>
listaTareas.appendChild(nuevaTarea);
```

Optimización del Rendimiento con DocumentFragment

Cada vez que se modifica el DOM con métodos como appendChild, el navegador puede realizar operaciones costosas de "reflow" (recalcular layouts) y "repaint" (redibujar la pantalla). Si se necesita añadir muchos elementos en un bucle, hacerlo uno por uno puede ser muy ineficiente y causar parpadeos visuales.

Para optimizar este proceso, se puede utilizar un DocumentFragment.[71] Un

DocumentFragment es un nodo DOM ligero y sin padre que actúa como un contenedor temporal. Se pueden añadir múltiples elementos al fragmento en memoria, lo que no desencadena reflows. Luego, se añade el fragmento completo al DOM de una sola vez con una única operación appendChild. Esto resulta en un solo reflow/repaint, mejorando significativamente el rendimiento.

Ejemplo con DocumentFragment:

JavaScript

```
const lista = document.getElementById('mi-lista');
const fragmento = document.createDocumentFragment();
const items =;

items.forEach(itemText => {
  const li = document.createElement('li');
  li.textContent = itemText;
  fragmento.appendChild(li); // Añadir al fragmento (en memoria)
});

// Añadir todo el fragmento al DOM de una sola vez
lista.appendChild(fragmento);
```

Eliminación de Elementos

Para eliminar nodos del DOM, existen dos métodos principales [50]:

- **padre.removeChild(hijo)**: El método tradicional, que requiere una referencia tanto al elemento padre como al hijo que se va a eliminar.
- **elemento.remove()**: El método moderno y más simple. Se llama directamente sobre el elemento que se desea eliminar, y este se elimina del DOM sin necesidad de referenciar a su padre.

Ejemplo con remove():

JavaScript

```
const tareaAeliminar = document.getElementById('tarea-1');
if (tareaAeliminar) {
  tareaAeliminar.remove();
```

```
}
```

16. Manejo de Eventos para la Interactividad

Los eventos son acciones que ocurren en la página web, como un clic del ratón, la pulsación de una tecla o la finalización de la carga de una página. JavaScript puede "escuchar" estos eventos y ejecutar código en respuesta, lo que hace que las páginas sean interactivas.[74]

addEventListener()

El método addEventListener() es la forma moderna y recomendada para registrar manejadores de eventos.[74]

- **Sintaxis**: elemento.addEventListener('tipoDeEvento', funcionManejadora, [opciones])
- **Ventajas**:
  - Permite añadir múltiples manejadores para el mismo evento en un solo elemento.
  - Ofrece un control detallado sobre la fase de propagación del evento (captura vs. burbujeo).
  - Funciona en cualquier EventTarget, no solo en elementos HTML.

Ejemplo:

JavaScript

```
const boton = document.getElementById('mi-boton');

function mostrarAlerta() {
  alert('¡Botón presionado!');
}
```

```
boton.addEventListener('click', mostrarAlerta);
```

El Objeto event

Cuando se dispara un evento, el navegador pasa automáticamente un objeto event a la función manejadora. Este objeto contiene información valiosa sobre el evento.[74]

- **event.target**: Es una referencia al elemento que originó el evento (por ejemplo, el botón exacto en el que se hizo clic).
- **event.currentTarget**: Es una referencia al elemento al que se adjuntó el manejador de eventos (puede ser diferente de event.target en el caso de la delegación de eventos).
- **event.preventDefault()**: Previene el comportamiento por defecto del navegador para ese evento. Por ejemplo, se puede usar en un evento submit de un formulario para detener el envío de la página y realizar una validación con JavaScript.

El Contexto this

Dentro de un manejador de eventos añadido con addEventListener y una función tradicional (function() {}), el valor de this se refiere al elemento al que se adjuntó el manejador (event.currentTarget).[75] Si se utilizan funciones de flecha (

() => {}), this conserva el valor del contexto en el que se definió la función, lo que es útil dentro de clases u objetos. Para forzar un contexto específico en una función tradicional, se puede usar el método .bind().

Delegación de Eventos

La delegación de eventos es un patrón de programación potente y eficiente. En lugar de añadir un manejador de eventos a cada uno de muchos elementos hijos (por

ejemplo, a cada <li> de una lista larga), se añade un único manejador a su contenedor padre (el <ul>).[20]

Dentro del manejador del padre, se utiliza event.target para identificar qué elemento hijo específico desencadenó el evento.

- **Ventajas**:
  - **Rendimiento**: Reduce la cantidad de memoria utilizada, ya que solo hay un manejador de eventos en lugar de muchos.
  - **Simplicidad**: El código es más limpio y fácil de mantener.
  - **Contenido Dinámico**: Funciona automáticamente para los elementos que se añaden al contenedor después de que se haya registrado el manejador, eliminando la necesidad de añadir manejadores a los nuevos elementos.

Ejemplo de Delegación de Eventos:

HTML

```
<ul id="lista-de-compras">
  <li>Manzanas</li>
  <li>Leche</li>
  <li>Pan</li>
</ul>
```

JavaScript

```
const lista = document.getElementById('lista-de-compras');

lista.addEventListener('click', function(event) {
  // Comprobar si el elemento clickeado (event.target) es un <li>
  if (event.target && event.target.tagName === 'LI') {
    // Aplicar un estilo al elemento específico que se clickeó
    event.target.style.textDecoration = 'line-through';
    console.log('Se hizo clic en el ítem:', event.target.textContent);
  }
```

```
});
```

En este ejemplo, un solo event listener en el <ul> gestiona los clics en todos los <li> presentes y futuros.

Conclusión

Este informe ha proporcionado una guía exhaustiva y detallada sobre HTML5, CSS3 y la manipulación del DOM con JavaScript, las tres tecnologías fundamentales que impulsan la web moderna.

En la **Parte I**, se estableció que **HTML5** es mucho más que un simple lenguaje de marcado. Su estructura semántica es la base de una web accesible y optimizada para los motores de búsqueda. El uso correcto de etiquetas como <main>, <article> y <nav> no es una opción estilística, sino un contrato funcional con tecnologías de asistencia y crawlers. Además, las capacidades de los formularios de HTML5, con sus nuevos tipos de entrada y atributos de validación, han demostrado ser una capa crucial para mejorar la experiencia del usuario, aunque siempre deben ser complementadas con una validación robusta en el lado del servidor.

La **Parte II** exploró **CSS3** como un sistema sofisticado para la presentación y el diseño. Se desmitificaron conceptos centrales como la cascada y la especificidad, enmarcando esta última no solo como un mecanismo de resolución, sino como una métrica de la salud y mantenibilidad del código. Los módulos de layout modernos, **Flexbox** y **Grid**, se presentaron como las soluciones definitivas para el diseño unidimensional y bidimensional, respectivamente. Finalmente, la combinación de **Variables CSS** y **Media Queries** se reveló como la estrategia preeminente para crear diseños responsivos, adaptativos y temáticos de una manera limpia y escalable, con las variables actuando como un puente dinámico hacia la manipulación con JavaScript.

Finalmente, la **Parte III** ofreció una introducción práctica a la **manipulación del DOM con JavaScript**, el motor de la interactividad web. Se analizaron los métodos modernos y eficientes para seleccionar, modificar, crear y eliminar elementos. Se destacaron patrones de optimización de rendimiento como el uso de DocumentFragment y técnicas de manejo de eventos avanzadas como la **delegación de eventos**, un patrón indispensable para aplicaciones dinámicas y de alto

rendimiento.

En conjunto, el dominio de estas tres tecnologías y, lo que es más importante, la comprensión de sus profundas interconexiones, es lo que distingue al desarrollo web competente del desarrollo web experto. La web es un ecosistema en constante evolución, pero los principios de estructura semántica, presentación escalable e interactividad performante aquí detallados seguirán siendo los pilares sobre los que se construirán las experiencias digitales del futuro.

Obras citadas

1. Conceptos básicos de HTML - Aprende desarrollo web | MDN, fecha de acceso: junio 25, 2025, https://developer.mozilla.org/es/docs/Learn_web_development/Getting_started/Your_first_website/Creating_the_content
2. HTML: Lenguaje de etiquetas de hipertexto - MDN Web Docs, fecha de acceso: junio 25, 2025, https://developer.mozilla.org/es/docs/Web/HTML
3. : The Document Metadata (Header) element - HTML - MDN Web Docs, fecha de acceso: junio 25, 2025, https://developer.mozilla.org/en-US/docs/Web/HTML/Reference/Elements/head
4. Empezar con CSS - Aprende desarrollo web - MDN Web Docs, fecha de acceso: junio 25, 2025, https://developer.mozilla.org/es/docs/Learn_web_development/Core/Styling_basics/Getting_started
5. The Document Body element - HTML - MDN Web Docs - Mozilla, fecha de acceso: junio 25, 2025, https://developer.mozilla.org/en-US/docs/Web/HTML/Reference/Elements/body
6. Structuring documents - Learn web development | MDN, fecha de acceso: junio 25, 2025, https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Structuring_content/Structuring_documents
7. Estructura HTML5: descubre la semántica del código | Blog de Arsys, fecha de acceso: junio 25, 2025, https://www.arsys.es/blog/estructura-de-html5-y-para-que-se-utiliza-la-semantica-del-codigo
8. HTML Semántico: Qué Es y Cómo Usarlo Correctamente - Semrush, fecha de acceso: junio 25, 2025, https://es.semrush.com/blog/html-semantico/
9. The Generic Section element - HTML - MDN Web Docs - Mozilla, fecha de acceso: junio 25, 2025, https://developer.mozilla.org/en-US/docs/Web/HTML/Reference/Elements/section
10. Nuevas etiquetas en HTML5 - Diseño Web Fácil, fecha de acceso: junio 25, 2025, https://disenowebfacil.wordpress.com/2014/03/26/nuevas-etiquetas-en-html5/
11. Lista completa de etiquetas HTML: qué son y cómo usarlas - Blog ..., fecha de acceso: junio 25, 2025,

https://www.swhosting.com/es/blog/lista-completa-de-etiquetas-html-que-son-y-como-usarlas
12. Referencia de Elementos HTML - HTML: Lenguaje de etiquetas de ..., fecha de acceso: junio 25, 2025, https://developer.mozilla.org/es/docs/Web/HTML/Reference/Elements
13. HTML5 Form elements and attributes | input type date, email, number, tel, range, datalist, fecha de acceso: junio 25, 2025, https://tutorial.techaltum.com/html5-form.html
14. Tipos de input de HTML5 - Aprende desarrollo web | MDN, fecha de acceso: junio 25, 2025, https://developer.mozilla.org/es/docs/Learn_web_development/Extensions/Forms/HTML5_input_types
15. The HTML5 input types - Learn web development | MDN, fecha de acceso: junio 25, 2025, https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Forms/HTML5_input_types
16. HTML5 Forms: Min, Max, Step Type Attribute | Wufoo, fecha de acceso: junio 25, 2025, https://www.wufoo.com/html5/minmaxstep-attributes/
17. Client-side form validation - Learn web development | MDN, fecha de acceso: junio 25, 2025, https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Forms/Form_validation
18. HTML Forms & Validation: Step-by-Step Guide | devChallenges, fecha de acceso: junio 25, 2025, https://devchallenges.io/learn/2-responsive-web/form-and-form-validation
19. HTML5 Form Validation Examples < HTML | The Art of Web, fecha de acceso: junio 25, 2025, https://www.the-art-of-web.com/html/html5-form-validation/
20. Event Delegation in JavaScript - GeeksforGeeks, fecha de acceso: junio 25, 2025, https://www.geeksforgeeks.org/javascript/event-delegation-in-javascript/
21. Referencia de Atributos HTML - HTML: Lenguaje de etiquetas de hipertexto | MDN, fecha de acceso: junio 25, 2025, https://developer.mozilla.org/es/docs/Web/HTML/Reference/Attributes
22. Selectores | web.dev, fecha de acceso: junio 25, 2025, https://web.dev/learn/css/selectors?hl=es
23. 3 tipos de estilos CSS: Interno, Externo e Inline - Guía Completa, fecha de acceso: junio 25, 2025, https://www.hostinger.es/tutoriales/tipos-de-estilos-css
24. Cascada CSS: ¿Por qué es importante saber como funciona?, fecha de acceso: junio 25, 2025, https://www.apinem.com/cascada-css/
25. La Cascada en CSS: Qué es y cómo funciona - Blog de Código Facilito, fecha de acceso: junio 25, 2025, https://codigofacilito.com/articulos/978
26. Especificidad - CSS - MDN Web Docs, fecha de acceso: junio 25, 2025, https://developer.mozilla.org/es/docs/Web/CSS/CSS_cascade/Specificity
27. Box model - W3C, fecha de acceso: junio 25, 2025, https://www.w3.org/TR/CSS2/box.html
28. CSS Box Model Module Level 3 - W3C, fecha de acceso: junio 25, 2025,

https://www.w3.org/TR/css-box-3/
29. CSS / Selectores CSS - W3C Wiki, fecha de acceso: junio 25, 2025, https://www.w3.org/wiki/CSS_/_Selectores_CSS
30. CSS selector structure - MDN Web Docs - Mozilla, fecha de acceso: junio 25, 2025, https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_selectors/Selector_structure
31. Complex Selectors - Learn to Code Advanced HTML & CSS, fecha de acceso: junio 25, 2025, https://learn.shayhowe.com/advanced-html-css/complex-selectors/
32. Advanced CSS Selectors: A Comprehensive Guide | devChallenges, fecha de acceso: junio 25, 2025, https://devchallenges.io/learn/2-responsive-web/css-advanced-selectors
33. Advanced Selectors in CSS - GeeksforGeeks, fecha de acceso: junio 25, 2025, https://www.geeksforgeeks.org/css/advanced-selectors-in-css/
34. CSS Selectors Cheat Sheet (Basic & Advanced) | BrowserStack, fecha de acceso: junio 25, 2025, https://www.browserstack.com/guide/css-selectors-cheat-sheet
35. CSS Flexbox Layout Guide | CSS-Tricks, fecha de acceso: junio 25, 2025, https://css-tricks.com/snippets/css/a-guide-to-flexbox/
36. Conceitos básicos de flexbox - CSS - MDN Web Docs, fecha de acceso: junio 25, 2025, https://developer.mozilla.org/pt-BR/docs/Web/CSS/CSS_flexible_box_layout/Basic_concepts_of_flexbox
37. Getting Started with CSS Grid, fecha de acceso: junio 25, 2025, https://css-tricks.com/getting-started-css-grid/
38. A Beginner's Guide to CSS Grid Layout — SitePoint, fecha de acceso: junio 25, 2025, https://www.sitepoint.com/introduction-css-grid-layout-module/
39. Using media queries - CSS - MDN Web Docs, fecha de acceso: junio 25, 2025, https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_media_queries/Using_media_queries
40. Responsive web design - Learn web development - MDN Web Docs - Mozilla, fecha de acceso: junio 25, 2025, https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/CSS_layout/Responsive_Design
41. Uso de media queries - CSS | MDN - MDN Web Docs - Mozilla, fecha de acceso: junio 25, 2025, https://developer.mozilla.org/es/docs/Web/CSS/CSS_media_queries/Using_media_queries
42. Harnessing the Power of CSS Custom Properties (CSS Variables) - DEV Community, fecha de acceso: junio 25, 2025, https://dev.to/waelhabbal/harnessing-the-power-of-css-custom-properties-css-variables-396e
43. Working with Variables and Media Queries in CSS - Linkysoft, fecha de acceso: junio 25, 2025, https://www.linkysoft.com/knowledgebase/446/Working-with-Variables-and-Med

ia-Queries-in-CSS.html?language=english
44. Using CSS custom properties (variables) - CSS | MDN, fecha de acceso: junio 25, 2025, https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_cascading_variables/Using_CSS_custom_properties
45. Custom properties (--*): CSS variables - MDN Web Docs, fecha de acceso: junio 25, 2025, https://developer.mozilla.org/en-US/docs/Web/CSS/--*
46. Using CSS Variables for Responsive Design | Responsive Design in ..., fecha de acceso: junio 25, 2025, https://www.chucksacademy.com/en/topic/responsive-design/custom
47. Modificando el CSS en Elementos HTML con el DOM - Paulo Galarza, fecha de acceso: junio 25, 2025, https://paulogalarza.com/modificando-el-css-en-elementos-html-con-el-dom/
48. Transformaciones, transiciones y animaciones en CSS | Blog de ..., fecha de acceso: junio 25, 2025, https://pedroprieto.github.io/post/transiciones_css/
49. Blog: Animaciones CSS: Mejora la Experiencia del Usuario con Transiciones y Keyframes Efectivos - Kranio, fecha de acceso: junio 25, 2025, https://www.kranio.io/blog/introduccion-a-css-parte-3-animaciones-css-para-mejorar-la-experiencia-del-usuario
50. Manipulación del DOM | JSvis - GitHub Pages, fecha de acceso: junio 25, 2025, https://centrogeo.github.io/JSvis/16-Manipulacion-DOM.html
51. DOM Comprension del modelo de objetos de documento DOM Marcado - FasterCapital, fecha de acceso: junio 25, 2025, https://fastercapital.com/es/contenido/DOM--Comprension-del-modelo-de-objetos-de-documento--DOM--Marcado.html
52. Cómo Manipular el DOM con JavaScript: Una Guía Práctica, fecha de acceso: junio 25, 2025, https://www.mgpanel.org/post/como-manipular-el-dom-con-javascript-una-guia-practica
53. Document: getElementsByClassName() method - Web APIs | MDN, fecha de acceso: junio 25, 2025, https://developer.mozilla.org/en-US/docs/Web/API/Document/getElementsByClassName
54. Explaining JavaScript Selectors: getElementById, getElementsByClassName, getElementsByTagName, querySelector, querySelectorAll - SheCodes, fecha de acceso: junio 25, 2025, https://www.shecodes.io/athena/116655-explaining-javascript-selectors-getelementbyid-getelementsbyclassname-getelementsbytagname-queryselector-queryselectorall
55. Formas de Seleccionar Elementos del DOM en JavaScript, fecha de acceso: junio 25, 2025, https://paulogalarza.com/formas-de-seleccionar-elementos-del-dom-en-javascript/
56. Diferencia entre querySelector, querySelectorAll, getElementbyId, getElementbyTag - Stack Overflow en español, fecha de acceso: junio 25, 2025,

https://es.stackoverflow.com/questions/22538/diferencia-entre-queryselector-queryselectorall-getelementbyid-getelementbyta
57. Buscar: getElement*, querySelector* - El Tutorial de JavaScript Moderno, fecha de acceso: junio 25, 2025, https://es.javascript.info/searching-elements-dom
58. Un repaso a getElementById, getElementsByClassName, getElementsByTagName y querySelector() - Desarrollo Web by esther solà, fecha de acceso: junio 25, 2025, https://www.esthersola.com/getelementbyelementid-queryselector/
59. Document.getElementsByClassName() - API web | MDN, fecha de acceso: junio 25, 2025, https://developer.mozilla.org/es/docs/Web/API/Document/getElementsByClassName
60. fecha de acceso: diciembre 31, 1969, https.shecodes.io/athena/116655-explaining-javascript-selectors-getelementbyid-getelementsbyclassname-getelementsbytagname-queryselector-queryselectorall
61. Modificació del DOM · Javascript - Sergi Coll, fecha de acceso: junio 25, 2025, https://seicoll.gitbooks.io/javascript/content/modificacio-del-dom.html
62. Modificar elementos del DOM con JavaScript - Luis Llamas, fecha de acceso: junio 25, 2025, https://www.luisllamas.es/modificar-elementos-del-dom-javascript/
63. Element: setAttribute() method - Web APIs - MDN Web Docs, fecha de acceso: junio 25, 2025, https://developer.mozilla.org/en-US/docs/Web/API/Element/setAttribute
64. How to use setAttribute and getAttribute in JavaScript DOM | by ARYAN - Medium, fecha de acceso: junio 25, 2025, https://medium.com/@aryanpandit1718/how-to-use-setattribute-and-getattribute-in-javascript-dom-9ec990342232
65. Cómo modificar atributos, clases y estilos en el DOM | DigitalOcean, fecha de acceso: junio 25, 2025, https://www.digitalocean.com/community/tutorials/como-modificar-atributos-clases-y-estilos-en-el-dom-es
66. Element: classList property - Web APIs | MDN, fecha de acceso: junio 25, 2025, https://developer.mozilla.org/en-US/docs/Web/API/Element/classList
67. DOMTokenList: toggle() method - Web APIs | MDN - MDN Web Docs, fecha de acceso: junio 25, 2025, https://developer.mozilla.org/en-US/docs/Web/API/DOMTokenList/toggle
68. DOMTokenList: replace() method - Web APIs | MDN - MDN Web Docs, fecha de acceso: junio 25, 2025, https://developer.mozilla.org/en-US/docs/Web/API/DOMTokenList/replace
69. HTMLStyleElement - API web - MDN Web Docs, fecha de acceso: junio 25, 2025, https://developer.mozilla.org/es/docs/Web/API/HTMLStyleElement
70. 9.4 Altering DOM Element Styles · A Practical Introduction to JavaScript, fecha de acceso: junio 25, 2025, https://shawnr.gitbooks.io/practical-introduction-to-javascript/content/the-docu

ment-object-model/94-altering-dom-element-styles.html
71. Insert and remove elements in the DOM with JavaScript - Luis Llamas, fecha de acceso: junio 25, 2025, https://www.luisllamas.es/en/insert-and-remove-elements-from-the-dom-javascript/
72. Creación y Eliminación de Elementos | Manipulación de DOM en JavaScript, fecha de acceso: junio 25, 2025, https://www.chucksacademy.com/es/topic/javascript-dom/creating-and-deleting-elements
73. Node: removeChild() method - Web APIs - MDN Web Docs, fecha de acceso: junio 25, 2025, https://developer.mozilla.org/en-US/docs/Web/API/Node/removeChild
74. Introduction to events - Learn web development | MDN, fecha de acceso: junio 25, 2025, https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Scripting/Events
75. EventTarget: addEventListener() method - Web APIs | MDN, fecha de acceso: junio 25, 2025, https://developer.mozilla.org/en-US/docs/Web/API/EventTarget/addEventListener
76. Manejo de Eventos Click en JavaScript: Guía Completa, fecha de acceso: junio 25, 2025, https://kaboomeventos.com.ar/javascript-add-click-event/
77. Best way to add event listener? - javascript - Reddit, fecha de acceso: junio 25, 2025, https://www.reddit.com/r/javascript/comments/60ix26/best_way_to_add_event_listener/
78. javascript - The value of "this" within the handler using ..., fecha de acceso: junio 25, 2025, https://stackoverflow.com/questions/1338599/the-value-of-this-within-the-handler-using-addeventlistener
79. Event bubbling - Learn web development | MDN, fecha de acceso: junio 25, 2025, https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Scripting/Event_bubbling
80. Understanding Event Delegation in JavaScript: From Bubbling to ..., fecha de acceso: junio 25, 2025, https://dev.to/ayako_yk/understanding-event-delegation-in-javascript-from-bubbling-to-data-attributes-4hk3
81. Event delegation - JavaScript.info, fecha de acceso: junio 25, 2025, https://javascript.info/event-delegation