

Sobre HTML Y CSS

Material pre clase - prof. Fiorella Calderón

Ecosistema digital y productos web

Cuando hablamos de ecosistema digital, nos referimos a un sistema interconectado de partes que funcionan mediante internet. Está compuesto por toda la infraestructura, plataformas, aplicaciones, servicios, dispositivos y también por los usuarios que interactúan con ellos para crear, intercambiar, distribuir y consumir información, productos o servicios. Es decir, combina elementos técnicos como software, protocolos de comunicación, redes y datos con actores humanos, generando dinámicas de intercambio de valor e interdependencia.

Este ecosistema digital opera porque estos componentes tecnológicos y humanos se comunican a través de protocolos de red que hacen posible el intercambio de datos, generando flujos de información, transacciones y experiencias digitales.

Para acceder al ecosistema digital existe una parte física que lo hace posible: el hardware. Esto se puede resumir en:

- **Servidores:** grandes centros que almacenan, procesan y gestionan la información para otros dispositivos y programas.
- **Dispositivos:** puntos de contacto que permiten interactuar con la información.
- **Redes:** conjuntos de dispositivos conectados y en constante comunicación mediante protocolos que facilitan la transmisión de datos.

Luego está la parte más digital, que corresponde a las plataformas a través de las cuales se accede a la información. No es una clasificación oficial, pero podríamos dividirlas de esta manera:

- **Software de escritorio:** programas instalados y ejecutados localmente en un dispositivo (por ejemplo, una computadora). Se consideran parte del ecosistema digital solo si están conectados a la red o integrados con servicios en línea (por ejemplo, Microsoft Word sincronizado con OneDrive).
- **Aplicaciones móviles:** software descargable e instalable en dispositivos móviles que interactúa con internet para funcionar (por ejemplo, aplicaciones bancarias o de redes sociales).
- **Productos web:** cualquier sitio accesible a través de un navegador, con contenido estático o dinámico, diseñado para usarse en línea.

Aquí es donde entra el concepto de sitio web. Un sitio web es el conjunto de páginas web interrelacionadas bajo un mismo dominio y alojadas en un servidor (hosting), accesibles mediante una URL base. A veces utilizamos el concepto de “página web” de manera intercambiable con “sitio web”, pero es importante entender que una página

web es una vista o "pantalla" individual (como Inicio, Acerca de o Servicios), que se interconecta con otras a partir de hipervínculos, y en conjunto conforman un sitio web.

Existen muchos tipos de sitios web, definidos principalmente por su propósito y lo que permiten hacer. Algunos de ellos son:

- **Portales de contenido** (informativos, corporativos, comerciales): sitios que presentan uno o varios temas específicos (por ejemplo, la web de una empresa).
- **Aplicaciones web**: herramientas interactivas accesibles desde un navegador que permiten realizar tareas o manipular información (como Google Docs, Trello).
- **E-commerce**: plataformas que permiten el intercambio de bienes, ya sea mediante venta directa (B2B o B2C), como Zara, o a través de marketplaces donde compradores y vendedores interactúan, como Amazon, eBay o Etsy.
- **Plataformas de software (SaaS)**: servicios que permiten descargar o usar software en línea, como Adobe Creative Cloud o Microsoft 365.
- **Plataformas de distribución digital**: que permiten el streaming o la descarga de contenido multimedia. Por ejemplo:
 - Video: Netflix, YouTube.
 - Música: Spotify, Apple Music.
 - Libros electrónicos: Kindle, Audible.
- **CMS (Sistemas de gestión de contenido)**: herramientas que permiten crear, administrar y modificar contenido digital sin conocimientos técnicos avanzados. Ejemplos: WordPress, Medium.
- **Foros y comunidades**: espacios donde los usuarios interactúan, discuten y comparten contenido, como Reddit o Stack Overflow.
- **Blogs**: sitios centrados en la publicación cronológica de artículos o noticias, como Medium o blogs personales.
- **Wikis**: sitios colaborativos editables por los usuarios para construir conocimiento compartido, como Wikipedia.
- **Portafolios**: sitios personales o profesionales donde se exhibe trabajo creativo (diseñadores, fotógrafos, artistas).
- **Landing pages**: páginas únicas enfocadas en la conversión, como captar datos o promocionar productos o servicios en campañas de marketing.

Por último, una parte importante del ecosistema es el **contenido digital descargable o transmisible a través de internet**. Este constituye el valor que se produce y se entrega mediante las distintas plataformas. Incluye música, videos, fotografías, documentos y archivos que deben diseñarse y adaptarse para ser descargados o transmitidos en un entorno digital.

Qué es SEO

SEO significa optimización para motores de búsqueda o “Search Engine Optimization”. Consiste en mejorar o adaptar un sitio web para que buscadores como Google puedan entender mejor su contenido, rastrearlo e indexarlo, es decir, agregarlo al índice de resultados para que, cuando alguien busque un término relacionado, el buscador pueda mostrar el sitio como una de sus respuestas.

La optimización no se limita al uso de palabras clave. Involucra trabajar en la **estructura, el contenido y los aspectos técnicos** de un sitio con el fin de que las páginas sean útiles, accesibles y relevantes. De esta manera, aumentan sus posibilidades de aparecer en posiciones destacadas dentro de los resultados orgánicos.

Por ejemplo, uno de los aspectos fundamentales del SEO es la **accesibilidad para los motores de búsqueda**. Esto significa que el sitio sea rastreable y no contenga barreras que impidan al buscador acceder y comprender el contenido para poder guardarlo en su índice. Esto implica contar con una buena estructura del código, enlaces internos bien organizados, textos alternativos en el contenido multimedia y un sitemap actualizado.

Otro aspecto esencial es la **calidad de la información**, que debe ser útil, confiable y clara. El uso estratégico de palabras clave en títulos, encabezados y textos alternativos facilita que el buscador vincule más rápidamente la búsqueda del usuario con el contenido de la página.

También existen factores más **técnicos**, relacionados con el rendimiento del sitio: la velocidad de carga, la compatibilidad con dispositivos móviles, la seguridad de los protocolos (como HTTPS) y la estabilidad visual.

La importancia de desarrollar sitios considerando la optimización para motores de búsqueda radica en facilitar de manera orgánica que los usuarios encuentren y visiten la página. Además de transmitir confianza y credibilidad, un buen SEO demuestra el cumplimiento de estándares fundamentales como accesibilidad, rapidez y calidad del contenido.

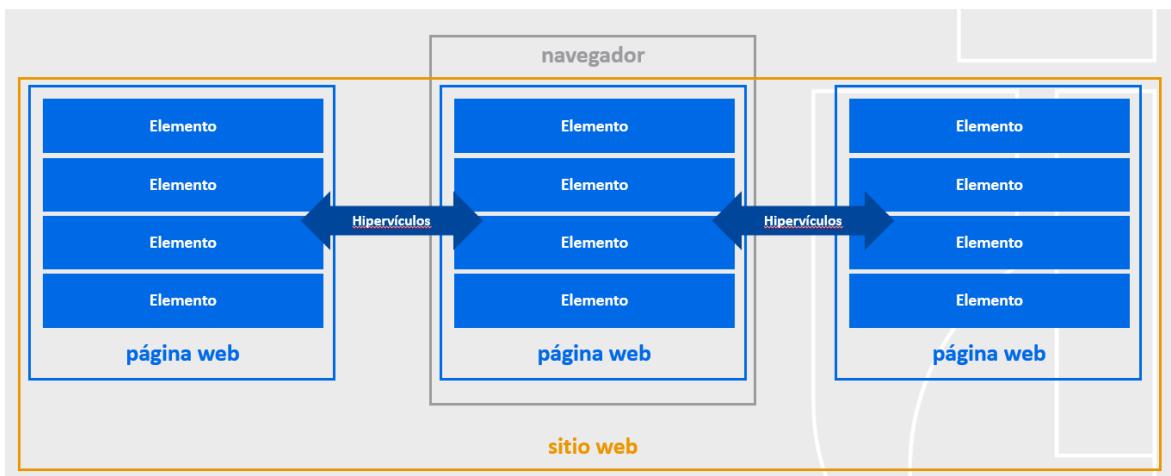
Google ha desarrollado una Guía de optimización en buscadores (SEO) para principiantes, disponible en este enlace:

[Guía de SEO para principiantes de Google](#)

Sitio web

Un sitio web es un espacio en la web, compuesto por un conjunto de **páginas web** interconectadas mediante **hipervínculos** que se alojan bajo un mismo **dominio** en un **servidor (hosting)**, siendo identificadas y accesibles a través de una **dirección URL** única .

- **Página web:** documento digital compuesto principalmente por elementos HTML que presenta información en una sola vista dentro de un sitio web.
- **Hipervínculo:** enlace interactivo que permite al usuario navegar entre diferentes páginas, ya sea dentro del mismo sitio web o hacia otros distintos.
- **Dominio:** nombre único que identifica a un sitio web en Internet y facilita su acceso en lugar de utilizar una dirección numérica (IP).
- Servidor (hosting): sistema informático que almacena y gestiona los archivos de un sitio web y los pone a disposición de los usuarios en Internet.
- **Dirección URL:** secuencia de caracteres que define la ubicación única de un recurso en la web, incluyendo el protocolo de acceso, el dominio y, en algunos casos, la ruta interna hacia un archivo o página específica.



Introducción a HTML



```
① index.html > ② html > ③ body > ④ dl
  5   | ⑤ <title>Document</title>
  6   | ⑥ </head>
  7
  8 <body>
  9   <h1>HTML Definition List</h1>
 10   <dl>
 11     <dt>HTML</dt>
 12       <dd>HyperText Markup Language: The standard
 13         language for creating web pages.</dd>
 14
 15     <dt>CSS</dt>
 16       <dd>Cascading Style Sheets: A stylesheet language
 17         used for describing the look and formatting of a
 18         document written in HTML.</dd>
 19
 20   </dl>
 21 </body>
```

HTML (HyperText Markup Language) es el lenguaje de marcado estándar para crear páginas web. En pocas palabras, **es el responsable de indicar al navegador cuál es el contenido y qué estructura debe tener**, para luego interpretarlo y mostrarlo al usuario como gráficos y elementos visuales. Se compone de una serie de elementos estructurados mediante etiquetas, que funcionan como piezas básicas para construir una página web.

Estas etiquetas, que encierran y delimitan el contenido, le dicen al navegador qué tipo de elemento es, por ejemplo “*esto es un párrafo*” o “*esto es una imagen*”, y cómo debe representarse en pantalla.

HTML es la base del desarrollo web porque **da forma y estructura a las páginas, es universal y compatible con cualquier navegador**, y junto con CSS y JavaScript permite incorporar diseño e interactividad. Además, un **uso correcto del lenguaje contribuye a mejorar la accesibilidad y el posicionamiento en buscadores (SEO)**.

Elementos HTML

Para entender el concepto de elemento en HTML se puede pensar en ellos como un conjunto de bloques de construcción. Cada bloque tiene un nombre, asociado a una forma y a una función específica. Los elementos son, en esencia, las piezas con las que se arma una página web. Al crear una página, lo que se hace es estructurar distintos elementos HTML que definen la organización y el contenido que, posteriormente, el navegador interpretará y mostrará al usuario.

Etiquetas

Los elementos HTML utilizan un **nombre específico que se representa a través de etiquetas**. La etiqueta es, en otras palabras, el “nombre” del bloque de construcción que **le indica al navegador cuál es su forma y su función**. Una etiqueta es una pieza de código que **encierra el contenido y le comunica al navegador qué tipo de elemento es**. Una vez interpretado, el navegador lo muestra según su configuración.

Las etiquetas normalmente se componen de una **etiqueta de apertura <> y una de cierre </>**, con algunas excepciones llamadas *etiquetas vacías* o *void elements*, como `` o

. Un ejemplo básico es <p></p>, que indica que el contenido encerrado corresponde a un párrafo. La función de este par es marcar **dónde comienza y dónde termina un elemento**. No cerrar una etiqueta es un error común que puede hacer que el navegador interprete incorrectamente todo lo que sigue como parte de ese mismo elemento.

Las etiquetas también pueden anidarse, es decir, colocar un elemento dentro de otro. Para mantener el orden y la legibilidad, una buena práctica es indentar los elementos anidados, agregando un tabulador en cada nivel. De esta forma, el código resulta más claro y es más sencillo llevar control de las etiquetas.

Concepto de renderizado

En este contexto, cuando hablamos de renderizado nos referimos al proceso mediante el cual el navegador interpreta el código fuente de una página y lo transforma en los elementos que componen la interfaz de usuario.

Básicamente, consiste en una serie de pasos:

1. El navegador recibe el documento HTML.
2. Analiza e interpreta (*parsing*) el código fuente.
3. Construye una representación de su contenido llamada **DOM (Document Object Model)** y otra de sus estilos llamada **CSSOM (CSS Object Model)**.
4. Combina el DOM y el CSSOM para crear un **árbol de renderizado**, donde se define la posición y apariencia de cada elemento.
5. Con base en ese árbol, el navegador calcula el **layout**, es decir, la ubicación y tamaño de cada elemento en relación con la pantalla.
6. Finalmente, **pinta** los elementos y los muestra como gráficos visibles en la ventana del navegador.

Estructura de un documento HTML

Para crear un documento HTML, el archivo debe guardarse con la extensión .html. Técnicamente, basta con un editor de texto para escribir el código, pero al guardarlo con esa extensión el navegador lo reconocerá como un documento HTML.

Además, para que el archivo sea interpretado correctamente, debe seguir una estructura básica compuesta por una serie de elementos que siempre se organizan en el mismo orden:

1. <!DOCTYPE html>

Se coloca al inicio del archivo y le indica al navegador que el documento está escrito en **HTML5**, la versión actual del estándar. No es un elemento HTML en sí, sino una declaración que asegura que el navegador renderice la página de acuerdo con las reglas modernas de HTML.

2. <html> ... </html>

Es el **elemento raíz** de la página. Todo el contenido del documento debe estar contenido dentro de estas etiquetas.

3. <head> ... </head>

Aquí se incluyen los **metadatos de la página**, es decir, información que **no se muestra directamente en el cuerpo del sitio**, pero que sirve para configurarlo y describirlo.

- Dentro del <head> se colocan elementos como **<meta>** (configuración de caracteres, autor, descripción) y los **enlaces a hojas de estilo** (**<link>**) y los **scripts externos** (**<script>**).
- Además, se define el título de la página, **<title> ... </title>**, que aparece en la pestaña del navegador y en los resultados de búsqueda.

4. <body> ... </body>

Es el **contenedor del cuerpo del documento**. Aquí se coloca todo el **contenido visible** para el usuario: textos, imágenes, enlaces, tablas, formularios, botones, etc. **Todo lo que el navegador muestra en pantalla se encuentra dentro del <body>**.

En resumen, la estructura mínima de un documento HTML define primero el tipo de documento con <!DOCTYPE html>, luego encierra todo el contenido en <html>, separando entre información para el navegador (<head>) y contenido visible para el usuario (<body>). **Es importante notar que todos los elementos de esta estructura quedan anidados dentro del <html>, siguiendo el orden que se observa en la imagen que se muestra a continuación.**

```

<!DOCTYPE html> Define el documento como html5
<html> Elemento raíz de la página html

<head> Metadatos de la página
    <title> ... </title> Título para mostrar en el navegador
</head> Final de los metadatos de la página

<body> Inicio del contenedor del cuerpo de la página
    ...
    ... Cuerpo de la página
</body> Final del contenedor del cuerpo de la página

</html> Final de la página html

```

```

Apuntes.html: Bloc de notas
Archivo Edición Formato Ver Ayuda
<!DOCTYPE html>
<html>

<head>
<title> Apuntes </title>
</head>
|
<body>
</body>
</html>

Línea 7, columna 1 100% Windows (CRLF) UTF-8

```

Cómo crear un documento HTML:

1. Abrir editor de texto (Bloc de notas/Textedit)
2. Estructurar el documento html
3. Guardar con la extensión .html
4. Abrir en el navegador

Metadatos

En HTML, los metadatos son información que **describe a los navegadores, motores de búsqueda y otros servicios el contenido y las características de una página web**, pero no se muestra directamente en la interfaz que ve el usuario. Se incluyen dentro de la sección `<head>` del documento y sirven principalmente para indicar de qué trata la página y cómo deben procesarla.

Su función principal es tanto describir el contenido como configurar aspectos técnicos de la página, como el idioma, la codificación de caracteres y la adaptación a distintos dispositivos móviles. Todo esto facilita que los motores de búsqueda indexen la página de manera correcta y la asocien a consultas relevantes.

Algunos metadatos principales son:

| | |
|---|--|
| <code><title> ... </title></code> | Título de la página para navegador y resultados de búsqueda. |
| <code><meta charset="UTF-8"></code> | Codificación de caracteres del documento |

| | |
|---|---|
| <code><meta name="viewport" content="width=device-width, initial-scale=1.0"></code> | Escala para distintos dispositivos |
| <code><meta name="description" content="... "></code> | Resumen del contenido de la página (snippets) |
| <code><meta name="author" content="..."></code> | Autor del documento |

Atributos

Son información adicional que se agrega a las etiquetas para definir o modificar las propiedades de un elemento. Se escriben siempre dentro de la etiqueta de apertura y siguen la forma **nombre="valor"**.

`<etiqueta inicial> atributo = "valor" > contenido </etiqueta final>`

Su función principal es aportar detalles extra que permiten a los navegadores entender mejor cómo debe comportarse o representarse un elemento. Por ejemplo, un enlace necesita un atributo que le indique la dirección a la que apunta, o una imagen requiere un atributo que defina el archivo de imagen que debe cargarse.

Los atributos siempre se colocan en la etiqueta de apertura de un elemento y se escriben en pares de nombre y valor, donde el nombre indica la propiedad que se quiere configurar y el valor define cómo debe aplicarse. Existen atributos **globales**, que pueden usarse en cualquier elemento, como *id*, *class* o *title*, y atributos **específicos**, que solo funcionan con determinados elementos, como *src* en ``, *href* en `<a>` o *alt* en ``.

Atributos generales

| | |
|------------------------|--------------------------------------|
| <code>id</code> | Nombre único de un elemento |
| <code>class</code> | Llama una clase que agrupa elementos |
| <code>title</code> | Título emergente |
| <code>lang</code> | Idioma del contenido |
| <code>hidden</code> | Elemento oculto |
| <code>tabindex</code> | Orden de tabulación |
| <code>accesskey</code> | Atajo de teclado |

| | |
|-----------------|---|
| contenteditable | Contenido editable |
| draggable | Contenido “arrastrable” |
| spellcheck | Activa o desactiva el corrector ortográfico |
| translate | Contenido “traducible” |

Elementos de Texto

Jerarquía del texto

En diseño y comunicación digital, la jerarquía del texto se refiere al orden y la importancia que se asigna a los diferentes fragmentos de información dentro de una página. La jerarquía permite que el lector distinga qué partes del texto son más relevantes, qué ideas son principales y cuáles son secundarias. Esto se logra a través de variaciones en el tamaño, peso, color o posición del texto.

Un concepto relacionado es el **escaneo**. Cuando los usuarios visitan una página web, no leen palabra por palabra, sino que escanean el contenido buscando puntos clave que les permitan orientarse y decidir qué leer con más detalle. Por esta razón, una buena jerarquía facilita la comprensión rápida de la información.

En HTML, la jerarquía del texto se representa principalmente mediante los **encabezados**, que van desde `<h1>` hasta `<h6>`.

- `<h1>` corresponde al nivel más alto y suele utilizarse para el título principal de la página.
- `<h2>` se usa para secciones importantes que dependen del `<h1>`.
- `<h3>` a `<h6>` permiten dividir subsecciones en niveles más detallados.
- `<p>` se utiliza para representar párrafos de texto, es decir, bloques de contenido que desarrollan ideas dentro de la estructura definida por los encabezados.

Este sistema crea una **estructura jerárquica** que no solo ayuda a los usuarios a leer y comprender mejor el contenido, sino que también **facilita a los motores de búsqueda interpretar la organización de la información**, lo cual es importante para el SEO.

Etiquetas de texto

| | |
|---------------------------------------|--|
| <code><h1>...</h1></code> | Encabezado, escala de h1 al h6 ajustando su tamaño |
|---------------------------------------|--|

| | |
|--|----------------------|
| <code><p>...</p></code> | Párrafo |
| <code><blockquote></blockquote></code> | Cita |
| <code><pre></pre></code> | Texto preformatado |
| <code></code> | Negrita (semántico) |
| <code></code> | Negrita |
| <code></code> | Cursiva (semántico) |
| <code><i></i></code> | Cursiva |
| <code><mark></mark></code> | Resaltado |
| <code><s></s></code> | Tachado |
| <code><u></u></code> | Subrayado |
| <code><ins></ins></code> | Texto insertado |
| <code></code> | Texto eliminado |
| <code><small></small></code> | Texto pequeño |
| <code><abbr></abbr></code> | Abreviatura |
| <code><cite></cite></code> | Referencia |
| <code><dfn></dfn></code> | Definición |
| <code><dfn></dfn></code> | Superíndice |
| <code><sub></sub></code> | Subíndice |
| <code></code> | Elemento contenedor |
| <code> </code> | Salto de línea |
| <code><hr></code> | Separador horizontal |

El Patito Feo

Introducción

Había una vez una **mamá pata** que estaba incubando sus huevos.

Uno de los huevos era diferente; era más grande y tenía un color extraño.

El nacimiento

Finalmente, el **huevo** rompió y de él salió un **patito** que no se parecía en nada a los demás.

"¡Qué patito tan feo!"

Los otros patitos se rieron de él y lo llamaron **feo**.

La búsqueda de aceptación

El **patito feo** decidió abandonar el hogar y buscar un lugar donde lo aceptaran.

Pasó por diferentes lugares, encontrándose con animales que lo despreciaban.

El cambio

Con el paso del tiempo, el patito creció y se convirtió en un hermoso **cisne**.

"Ahora soy **bella**", pensó el **patito**.

Regresó a su hogar y fue bienvenido por todos, quienes se sorprendieron al verlo transformado.

Conclusión

Así, el oso oitap encontró su lugar en el mundo y aprendió que la belleza se encuentra en el interior.

Este cuento nos enseña a valorar la diversidad y a aceptar a los demás tal como son.

```
'''<!DOCTYPE html> == $0
<html>
  > <head> ... </head>
  > <body>
    ><h1>El Patito Feo</h1>
    ><hr>
    ><h2>Introducción</h2>
    ><p>
      "Había una vez una "
      <strong>mamá pata</strong>
      " que estaba incubando sus huevos."
    </p>
    ><blockquote>
      <p>Uno de los huevos era diferente; era más grande y tenía un color extraño.</p>
    </blockquote>
    ><h2>El nacimiento</h2>
    ><p>
      "Finalmente, el "
      <strong>huevito</strong>
      " rompió y de él salió un "
      <strong>patito</strong>
      " que no se parecía en nada a los demás."
    </p>
    ><pre>"¡Qué patito tan feo!"</pre>
    ><p>
      <b>Los otros patitos</b>
      " se rieron de él y lo llamaron "
      <em>feo</em>
      "."
    </p>
    ><hr>
    ><h2>La búsqueda de aceptación</h2>
    ><p>
      <del>El patito feo</del>
      " decidió abandonar el hogar y buscar un lugar donde "
      <abbr>...</abbr>
    </p>
```

Elementos de tabla

Una tabla es una estructura que organiza la información en filas y columnas, lo que permite presentar datos de manera ordenada y fácil de interpretar. Su propósito principal es mostrar información que sigue una relación matricial, como listados, horarios, comparaciones o reportes de datos.

Una tabla se compone de diferentes partes:

- **Cabeza de columna:** es la fila inicial de la tabla, donde se definen los títulos o encabezados de cada columna. En HTML se representa con **<thead>** y sus celdas con **<th>**.
- **Columna:** es el conjunto vertical de celdas que comparte el mismo encabezado. Representa una categoría o tipo de información dentro de la tabla.
- **Fila:** es el conjunto horizontal de celdas que agrupan información relacionada. En HTML se definen con **<tr>**.
- **Celda:** es la unidad básica de una tabla, el espacio donde se coloca un dato específico. Se representa con **<td>** para celdas de contenido y **<th>** para celdas de encabezado.

De manera general, la estructura mínima de una tabla en HTML sigue este orden:

```

<table border="1">
  <thead>
    <tr>
      <th>Categoría A</th>
      <th>Categoría B</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Item 1A</td>
      <td>Item 1B</td>
    </tr>
    <tr>
      <td>Item 2A</td>
      <td>Item 2B</td>
    </tr>
    <tr>
      <td>Item 3A</td>
      <td>Item 3B</td>
    </tr>
  </tbody>
</table>

```

Etiquetas para tablas

| | |
|--|------------------------------|
| <table></table> | Define una tabla |
| <thead></thead> | Encabezado de tabla |
| <tr></tr> | Fila |
| <th></th> | Encabezado de fila |
| <col> | Columna |
| <colgroup></colgroup> | Agrupa las columnas |
| <td></td> | Celda de datos |
| <tbody></tbody> | Agrupa el cuerpo de la tabla |
| <tfoot></tfoot> | Pie de la tabla |
| <caption></caption> | Descripción de la tabla |

Atributos de tablas

| | |
|----------------|--|
| colspan | Centrado de una celda en varias columnas |
| rowspan | Centrado de una celda en varias filas |
| headers | Celda como encabezado |

Elementos de lista

Una **lista** es una estructura que permite organizar y presentar información en forma de conjunto de ítems relacionados. Las listas son útiles para agrupar elementos de manera

ordenada o desordenada, facilitando la lectura y el escaneo del contenido por parte del usuario.

Existen tres tipos principales de listas:

- **Lista desordenada ():** muestra los elementos sin un orden específico, normalmente representados con viñetas. Es ideal para enumerar características, ideas o elementos que no tienen jerarquía entre sí.
- **Lista ordenada ():** muestra los elementos siguiendo un orden numérico o alfabético. Es útil cuando los ítems tienen una secuencia, como pasos de un procedimiento o un ranking.
- **Lista de definición (<dl>):** se utiliza para representar pares de término y definición. Dentro de ella se emplean *<dt>* para el término y *<dd>* para la definición.

Cada ítem de una lista, ya sea ordenada o desordenada, se representa mediante la etiqueta *</i>* (list item).

```
<ul>
  <li>Elemento</li>
  <li>Elemento</li>
  <li>Elemento</li>
</ul>
```

```
<ol>
  <li>Elemento uno</li>
  <li>Elemento dos</li>
  <li>Elemento tres</li>
</ol>
```

```
<dl>
  <dt>Elemento</dt>
  <dd>Descripción<dd>
  <dt>Elemento</dt>
  <dd>Descripción</dd>
</dl>
```

Etiquetas de lista

| | |
|------------------------------|---|
| | Lista desordenada |
| | Lista ordenada |
| | Elementos de una lista ordenada o desordenada |
| <dl></dl> | Lista descriptiva |
| <dt></dt> | Término dentro de una lista descriptiva |
| <dd></dd> | Descripción de un término en una lista descriptiva |

Lista no ordenada **Lista ordenada** **Lista descriptiva**

- Item 1
- Item 2
- Item 3

1. item 1
2. item 2
3. item 3

- Item 1
- descripción
- Item 2
- descripción

Elementos de enlaces

Un **enlace** o **hipervínculo** es un elemento que conecta una página con otra, o que permite al usuario acceder a un recurso externo como un archivo, una imagen, un correo electrónico o una ubicación dentro de la misma página. Son fundamentales para la navegación en la web, ya que permiten desplazarse entre documentos relacionados.

El elemento que define un enlace es **< a >**, conocido como *anchor (ancla)*. Su atributo principal es *href*, que especifica la dirección o recurso al que apunta el enlace.

```
<a href="enlace">contenido</a>
```

En este caso, el texto “contenido” será el contenido clicable, lo que verá el usuario en pantalla y sabrá que debe hacer clic y el navegador llevará al usuario a la dirección definida en *href*.

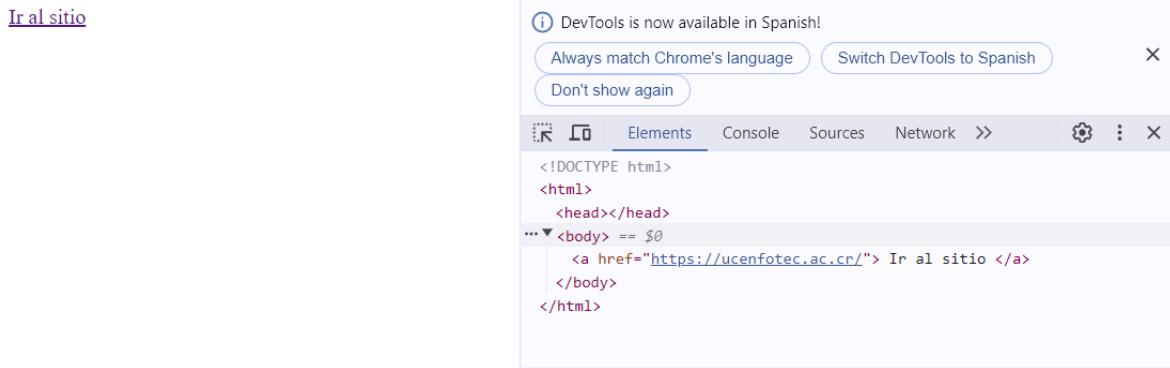
Etiquetas de enlace

| | |
|-------------------------------------|--|
| < a > </ a > | Identifica el elemento como un enlace |
| < link > | Vínculos externos |
| < nav > </ nav > | Para enlaces de navegación interna |

Atributos de enlaces

| | |
|-------------|-----------------------|
| href | URL del enlace |
|-------------|-----------------------|

| | |
|-----------------|--|
| target | Cómo abrir el enlace: self, blank, parent, top |
| rel | Relación con el destino:nofollow, noopener, noreferrer |
| download | Descargar el archivo en el enlace en lugar de abrirlo |



Elementos de Imágenes

Las imágenes son un recurso fundamental en la web, ya que aportan valor visual, ayudan a comunicar ideas y mejoran la experiencia del usuario. En HTML, se insertan mediante la etiqueta ****, cuyo atributo principal es **src**, que define la ruta del archivo, y **alt**, que proporciona un texto alternativo útil para accesibilidad y SEO.

Usos de imagen

- **Fotografía:** imágenes capturadas con cámara, usadas comúnmente para mostrar productos, personas, lugares o situaciones reales.



- **Ilustración:** gráficos creados de manera digital o manual, que permiten representar conceptos de forma creativa o abstracta.

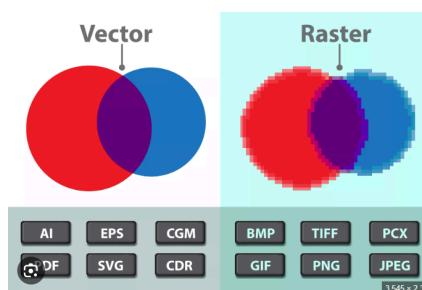


- **Ícono:** gráficos simples y reconocibles, empleados para representar acciones, funciones o categorías en la interfaz (ejemplo: el ícono de búsqueda con una lupa).



Imágenes ráster vs. Imágenes vectoriales

- **Ráster:** están formadas por píxeles. Su calidad depende de la resolución, por lo que al ampliarlas pueden perder nitidez. Formatos comunes: JPEG, PNG, GIF, WebP, AVIF.
- **Vectoriales:** están formadas por trazos y fórmulas matemáticas. No pierden calidad al escalarse y son ideales para logotipos e íconos. El formato más utilizado en la web es SVG.



Formatos de imagen más comunes

- **JPEG (.jpg/.jpeg):** ideal para fotografías, buena compresión con pérdida.
- **GIF (.gif):** usado para imágenes animadas de baja calidad o gráficos simples.
- **PNG (.png):** soporta transparencia y compresión sin pérdida, útil para gráficos e interfaces.

- **SVG (.svg)**: formato vectorial, escalable sin pérdida de calidad, perfecto para logotipos e íconos.
- **WebP (.webp)**: formato moderno que combina buena calidad con menor peso, soporta transparencia y animaciones.
- **AVIF (.avif)**: aún más eficiente que WebP, con alta compresión y calidad, recomendado en entornos modernos.

Optimización de la imagen

La optimización es clave para mejorar el rendimiento y la experiencia del usuario en la web:

- **Peso**: reducir el tamaño del archivo para que cargue más rápido.
- **Dimensiones**: usar imágenes con el tamaño adecuado, evitando archivos más grandes de lo necesario.
- **Responsive**: adaptar las imágenes a distintos dispositivos y resoluciones mediante atributos como srcset y sizes.
- **Lazy loading**: cargar las imágenes solo cuando estén a punto de mostrarse en pantalla (loading="lazy"), lo que reduce el tiempo inicial de carga de la página.

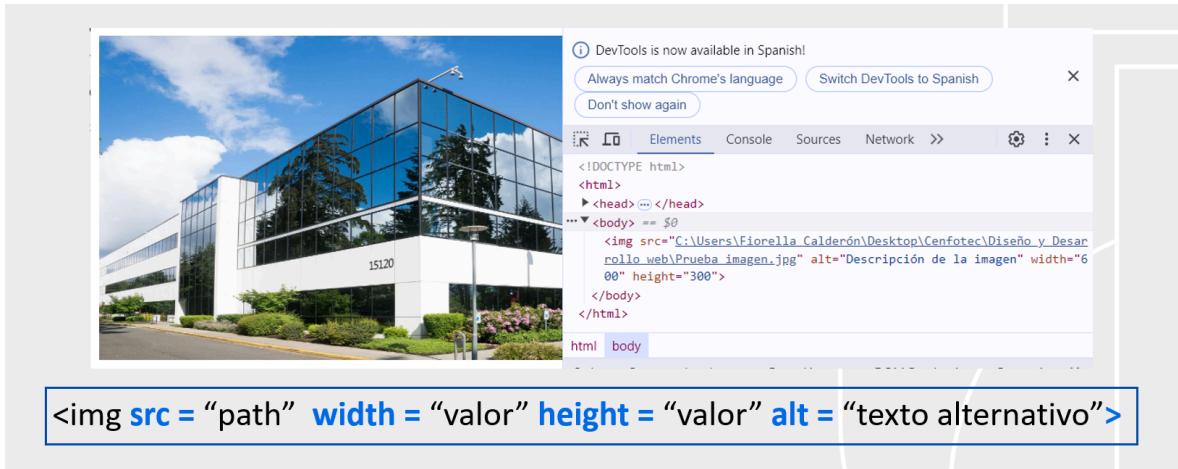
Etiquetas de imagen

| | |
|--|---|
| <code></code> | Inserta una imagen |
| <code><figure></figure></code> | Marca el contenedor de una imagen con su pie de foto |
| <code><figcaption></figcaption></code> | Pie de foto |
| <code><picture></picture></code> | Imagen adaptable |
| <code><source></code> | Alternativas para imagen adaptable |

Atributos de imagen

| | |
|--------------|--|
| src | URL de la imagen |
| alt | Texto alternativo por si la imagen no está disponible |
| width | Ancho de la imagen en px o % |

| | |
|----------------|-----------------------------|
| height | Alto de la imagen en px o % |
| loading | Carga diferida |



Multimedia (audio y video):

Además de texto e imágenes, la web permite integrar elementos multimedia como audio y video, que enriquecen la experiencia del usuario y hacen que los sitios sean más dinámicos e interactivos. En HTML, estos recursos se incorporan mediante etiquetas específicas que facilitan su reproducción directamente en el navegador, sin necesidad de complementos externos.

Audio

El audio se inserta en una página usando la etiqueta `<audio>`. Dentro de ella se pueden definir una o varias fuentes con `<source>` para asegurar compatibilidad con diferentes navegadores.

- Atributos comunes:
 - **controls**: agrega los controles básicos de reproducción (play, pausa, volumen).
 - **autoplay**: reproduce automáticamente el audio al cargar la página (aunque algunos navegadores lo bloquean por accesibilidad).
 - **loop**: reproduce el audio en bucle.



The screenshot shows a browser window displaying a video player interface. At the top, there are standard media controls: a play button, a progress bar indicating 0:07 / 2:21, a volume icon, and a three-dot menu icon. Below these controls, the browser's developer tools are open, specifically the Elements tab. The DOM tree is visible, showing the following code:

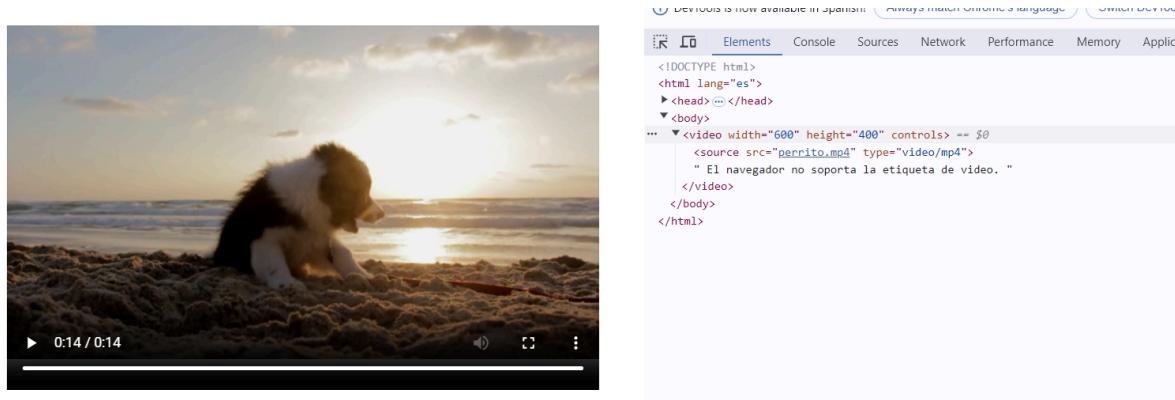
```
<!DOCTYPE html>
<html lang="es">
  <head> ... </head>
  <body>
    <audio controls> == $0
      <source src="audio.mp3" type="audio/mpeg">
        " El navegador no soporta la etiqueta de audio. "
    </audio>
  </body>
</html>
```

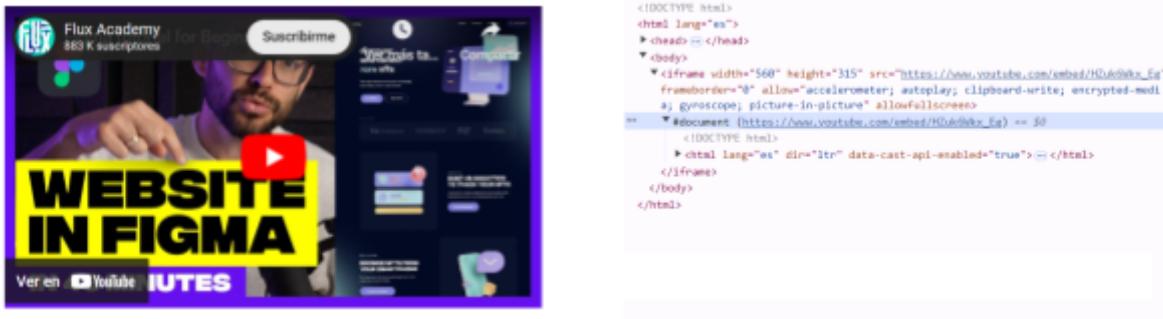
Video

El video se incorpora con la etiqueta **<video>**, que también admite múltiples fuentes.

- Atributos comunes:

- **controls**: agrega los controles básicos (reproducir, pausar, volumen, pantalla completa).
- **autoplay**: inicia la reproducción automáticamente.
- **muted**: silencia el audio por defecto (usado a menudo con autoplay).
- **poster**: define una imagen que se muestra como portada antes de reproducir el video.
- **loop**: repite el video en bucle.





Consideraciones importantes

- **Compatibilidad:** usar varios formatos de archivo (por ejemplo MP3/OGG para audio o MP4/WebM para video) asegura que el contenido funcione en diferentes navegadores.
- **Optimización:** comprimir archivos multimedia reduce el tiempo de carga y mejora la experiencia del usuario.
- **Accesibilidad:** es recomendable incluir alternativas como transcripciones de audio o subtítulos en video para que el contenido sea accesible a más usuarios.

Etiquetas de multimedia

| | |
|---------------------------|--|
| <audio></audio> | Reproductor de audio |
| <video></video> | Reproductor de video |
| <source> | Fuentes alternativas para el archivo multimedia |
| <track></track> | Descripciones o subtítulos |
| <picture></picture> | Contenedor de imágenes adaptativas |
| <figure></figure> | Contenedor con pie de foto |
| <figcaption></figcaption> | Pie de foto |
| <embed> | Contenido incrustado como PDFs |
| <object></object> | Contenido incrustado como animaciones |
| <param> | Define los parámetros de un object |
| <iframe></iframe> | Muestra una página dentro de otra, ej: youtube |

Atributos para multimedia

| | |
|-----------------|--|
| src | URL del archivo multimedia |
| controls | Muestra los controles de producción |
| autoplay | Reproduce automáticamente |
| loop | Reducción en bucle |
| muted | Sin sonido |
| poster | Vista o “thumbnail” del video |

Elementos de formulario

Un formulario en HTML es una estructura que permite a los **usuarios ingresar y enviar información** a un servidor. Se utiliza en procesos comunes como iniciar sesión, registrarse, buscar contenido, comprar en línea o enviar comentarios.

Estructura de un formulario

Los formularios se definen con la etiqueta **<form>**, que actúa como contenedor de todos los campos y controles. Sus atributos principales son:

- **action**: indica la URL a la que se enviarán los datos del formulario.
- **method**: especifica cómo se enviarán los datos. Los valores más comunes son GET (envía los datos en la URL) y POST (los envía en el cuerpo de la petición).

Agrupación de campos

Para organizar los campos de un formulario, se pueden usar:

- **<fieldset>**: agrupa un conjunto de controles relacionados.
- **<legend>**: etiqueta que da un título al grupo definido por <fieldset>.
- **<label>**: describe cada campo e indica su función. Asociar etiquetas correctamente con los campos mejora la accesibilidad.

Campos de entrada (<input>)

El elemento <input> es el más versátil de los formularios y su comportamiento depende del atributo **type**, que define el tipo de dato que se solicita.

- **Texto y variantes:**
 - **text:** texto libre.
 - **password:** texto oculto con asteriscos.
 - **email:** para direcciones de correo.
 - **tel:** para números de teléfono.
 - **url:** para direcciones web.
 - **search:** campo de búsqueda.
- **Área de texto:**
 - **<textarea>:** permite introducir varias líneas de texto.
- **Números y fechas:**
 - **number:** valores numéricos.
 - **range:** control deslizante con un rango de valores.
 - **date, time, month, week:** seleccionadores de fecha y hora.
- **Selección:**
 - **radio:** permite elegir una sola opción entre varias.
 - **checkbox:** permite seleccionar múltiples opciones.
- **Carga de archivos:**
 - **file:** subir un archivo.

- **image**: botón gráfico que envía el formulario.

Campo selector

Para ofrecer listas desplegables, se usa el elemento `<select>` junto con sus elementos internos:

- **<option>**: define cada opción disponible.
- **<optgroup>**: permite agrupar opciones bajo un título.

Botones

Los botones permiten enviar o reiniciar el formulario, o ejecutar acciones específicas:

- **<button type="submit">**: envía los datos del formulario.
- **<button type="reset">**: reinicia los campos a sus valores por defecto.
- **<button type="button">**: botón genérico que se puede programar con JavaScript.

Formulario de Registro

Información Personal

Nombre:

Correo Electrónico:

Mensaje:

Preferencias

País:

Color favorito:

```

<!DOCTYPE html>
<html lang="es">
  <head>
    <title>Formulario </title>
  </head>
  <body>
    <h1>Formulario de Registro</h1>
    <form action="/submit" method="POST">
      <!-- Agrupación de elementos relacionados -->
      <fieldset>
        <legend>Información Personal</legend>
        <label for="nombre">Nombre:</label>
        <input type="text" id="nombre" name="nombre" required>
        <br>
        <br>
        <label for="email">Correo Electrónico:</label>
        <input type="email" id="email" name="email" required>
        <br>
        <br>
        <label for="mensaje">Mensaje:</label>
        <br>
        <textarea id="mensaje" name="mensaje" rows="4" cols="50"></textarea>
      </fieldset>
      <!-- Selección de opciones -->
      <fieldset> -- $0
        <legend>Preferencias</legend>
        <label for="pais">País:</label>
        <select id="pais" name="pais"> -- /> /> /> /> />
        <br>
        <br>
        <label for="color">Color favorito:</label>
        <input list="colores" id="color" name="color">
        <datalist id="colores"> -- </datalist>
      </fieldset>
      <br>
      <!-- Botón para enviar -->
      <button type="submit" value="Enviar"></button>
      <!-- Resultado -->
      <output name="resultado"></output>
    </form>
  </body>
</html>

```

Etiquetas de formularios

| | |
|--|------------------------------|
| <form></form> | Define el área de formulario |
| <input> | Campo de entrada |
| <textarea></textarea> | Campo de texto multilínea |

| | |
|--|---|
| <code><select></select></code> | Lista desplegable |
| <code><option></option></code> | Opciones de la lista desplegable |
| <code><optgroup></optgroup></code> | Agrupa opciones de la lista desplegable |
| <code><button></button></code> | Botón del formulario, puede tener distintas funciones como enviar o resetear el formulario |
| <code><label></label></code> | Etiqueta descriptiva del campo |
| <code><fieldset></fieldset></code> | Agrupa varios campos del formulario |
| <code><legend></legend></code> | Título de un grupo de campos en el formulario |
| <code><datalist></datalist></code> | Opciones sugeridas para un campo de entrada |
| <code><output></output></code> | Cálculo de un resultado |
| <code><progress></progress></code> | Barra de progreso |
| <code><meter></meter></code> | Valor dentro de un rango |

Atributos de formularios

| | |
|---------------------------|---|
| <code>action</code> | URL para enviar el formulario |
| <code>method</code> | Método de envío del formulario: get, post |
| <code>autocomplete</code> | Activa autocompletado |
| <code>placeholder</code> | Texto de relleno en el campo |
| <code>required</code> | Campo obligatorio |
| <code>readonly</code> | Campo de solo lectura |
| <code>disabled</code> | Campo desactivado |
| <code>value</code> | Valor por defecto del campo |
| <code>name</code> | Nombre del campo para enviar los datos |
| <code>type</code> | Tipo de entrada: texto, contraseña, correo, numérico, checkbox, radio... |

| | |
|------------------|--|
| maxlength | Caracteres permitidos |
| min/max | Valores extremos que puede ocupar el campo |
| Step | Incremento en los valores que puede tomar un campo numérico |
| checked | Marcado por defecto de un campo tipo checkbox o radio |
| multiple | Para selección múltiple |
| pattern | Para validar la entrada |

Comentarios en HTML

Son fragmentos de texto dentro del código que no se muestran en el navegador. Se utilizan para dejar notas, aclaraciones o descripciones que facilitan la comprensión del código.

La sintaxis de un comentario es:

```
<!-- Comentario -->
```

Sirven para:

- Explicar la función de una sección del código.
- Marcar recordatorios o tareas pendientes.
- Desactivar temporalmente fragmentos de código sin eliminarlos.

```
<!-- Inicio formulario -->
<h1>Formulario de inscripción</h1>
<!-- Final del formulario -->

<!--
<h1 style="text-align: center;">Título centrado</h1>
<p style="text-align: right;">Párrafo a la derecha.</p>

-->
```

Elemento <div>

El **<div> (división)** es un contenedor genérico en HTML. Se utiliza para **encerrar y agrupar otros elementos** dentro de una página web. Se le llama contenedor genérico porque, a diferencia de otros elementos, el **<div>** no aporta un significado específico sobre el tipo de contenido que encierra. En otras palabras, le dice al navegador “esto es un contenedor”, pero no describe la naturaleza del contenido dentro de él.

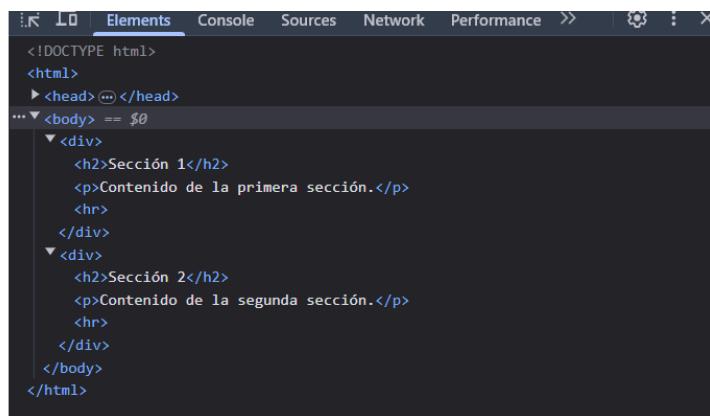
Un **<div>** puede contener cualquier tipo de elemento HTML: texto, imágenes, listas, formularios e incluso otros **<div>**. Cuando se coloca un **<div>** dentro de otro, se habla de **anidación**, lo que permite crear estructuras jerárquicas más complejas para organizar el contenido.

Sección 1

Contenido de la primera sección.

Sección 2

Contenido de la segunda sección.

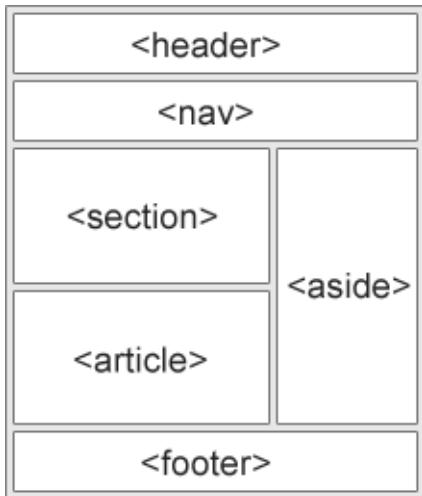


Aunque es un contenedor genérico, en la práctica se recomienda **darle identidad** utilizando atributos como **id** o **class**. Estos atributos permiten manipular el contenedor como un elemento específico, y así trabajar con cada contenedor de manera independiente.

```
<div id="contenedor-principal">
  <div class="sección">
    <h2>Título de la sección</h2>
    <p>Este es un párrafo dentro de un div.</p>
  </div>
</div>
```

Si bien el **<div>** es uno de los elementos más usados en HTML, es importante no abusar de él. Cuando el contenido tiene un propósito específico, es preferible usar **elementos semánticos** como **<section>**, **<article>**, **<aside>** o **<footer>**, ya que enriquecen la estructura de la página y favorecen la accesibilidad y el SEO.

HTML Semántico



Cuando hablamos de **HTML semántico**, nos referimos a escribir código con etiquetas que, además de estructurar el contenido, **tienen un significado propio**. Esto significa que transmiten su propósito claramente, facilitando la comprensión de la página tanto para desarrolladores como para navegadores y motores de búsqueda.

Un elemento es “semántico” cuando indica explícitamente para qué debe ser utilizado. Por ejemplo, un <div> y un <section> son ambos contenedores, pero mientras el <div> es genérico, el <section> le comunica

al navegador y a los buscadores que ese bloque corresponde a una sección concreta del documento.

El uso de HTML semántico aporta varias ventajas, entre ellas una mayor comprensión del contenido tanto para humanos como para máquinas, una mejor accesibilidad gracias a que tecnologías como los lectores de pantalla pueden interpretar la página de forma más clara, y una optimización para SEO, ya que los motores de búsqueda entienden mejor la jerarquía y relevancia de cada bloque. Además, facilita el mantenimiento porque la estructura del código resulta más clara y organizada, y permite un código más intuitivo, lo que mejora la colaboración entre desarrolladores y diseñadores.

Elementos principales:

- **<header>**: define un encabezado para un documento o una sección.
- **<nav>**: representa un conjunto de enlaces de navegación.
- **<section>**: define una sección dentro de un documento.
- **<article>**: indica contenido independiente y autónomo, como una noticia o una entrada de blog.
- **<aside>**: representa contenido relacionado pero no central, como barras laterales o notas adicionales.
- **<footer>**: define el pie de página de un documento o sección.
- **<details>**: permite mostrar detalles adicionales que el usuario puede desplegar bajo demanda.

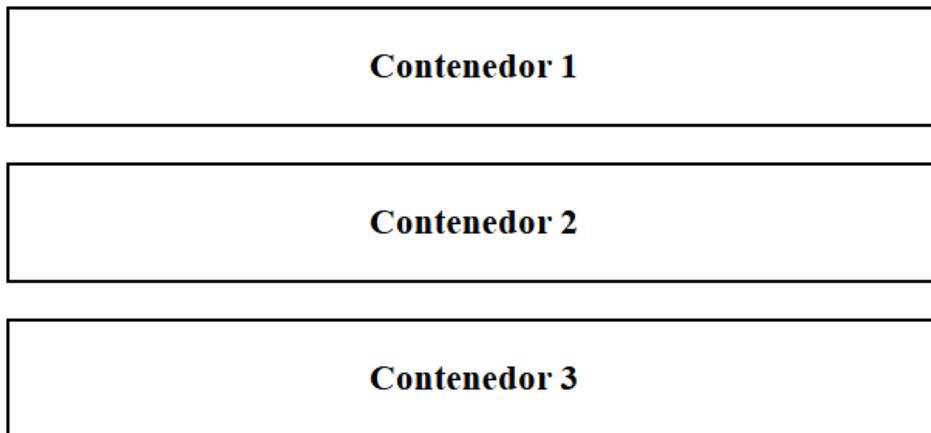
- **<summary>**: establece un encabezado para el elemento **<details>**.



Elementos de bloque

En HTML, un elemento de bloque es aquel **que ocupa todo el ancho disponible de su contenedor y fuerza un salto de línea antes y después de sí mismo**. Esto significa que el contenido que venga después se colocará automáticamente debajo.

Los contenedores, como `<div>` o los elementos semánticos (`<section>`, `<header>`, `<footer>`, `<article>`, `<nav>`), son ejemplos de elementos de bloque. Que sean de bloque implica que pueden encerrar otros bloques y elementos en línea, funcionando como **piezas estructurales que dividen la página en secciones grandes y bien delimitadas**.



Introducción a CSS

Como dijimos antes, los elementos HTML son elementos de bloque, lo que significa que, por defecto, se apilan unos debajo de otros ocupando todo el ancho disponible. Para poder crear composiciones más complejas y dar forma a la interfaz, es necesario “**alterar la apariencia y la disposición de esos elementos**”. Para esto se utiliza **CSS (Cascading Style Sheets)**.

CSS es el **lenguaje utilizado para describir la presentación de documentos HTML**. A través de **reglas de estilo**, le indica al navegador cómo debe mostrarse cada elemento de la página. Esto incluye desde aspectos visuales como colores, tipografías y márgenes, hasta cuestiones de layout como tamaños, alineaciones, distribuciones o incluso animaciones.

En resumen, mientras HTML se encarga de **estructurar el contenido**, CSS se ocupa de **estilizarlo y organizarlo en pantalla**.

Concepto de estilo

En el contexto de la web, un **estilo** es el **conjunto de reglas que determinan la apariencia visual de los elementos HTML en una página**. Estas reglas son escritas en CSS y permiten controlar cómo se muestran los distintos componentes de la interfaz en el navegador.

Cada estilo está compuesto por un **selector** (que indica a qué elemento HTML se aplicará) y una o varias **propiedades** con sus respectivos valores. Estas propiedades pueden definir características como el **color**, la **tipografía**, los **márgenes**, el **tamaño**, la **alineación**, la **disposición en pantalla** o incluso el **comportamiento visual dinámico** de un elemento.

El diagrama muestra dos historias paralelas: 'El Patito Feo' y 'La Historia del Patito Feo'. Una flecha azul apunta de 'El Patito Feo' hacia 'La Historia del Patito Feo', indicando la transformación del personaje.

El Patito Feo:

- Introducción:** Había una vez una **mamá pato** que estaba incubando **sus huevos**. Uno de los huevos era diferente; era más grande y tenía un color extraño.
- El nacimiento:** Finalmente, el huevo rompió y de él salió un **patito** que no se parecía en nada a los demás.
"¡Qué patito tan feo!"
- Los otros patitos se rieron de él y lo llamaron feo.**
- La búsqueda de aceptación:** El patito feo decidió abandonar el hogar y buscar un lugar donde lo aceptaran. Pasó por diferentes lugares, encontrándose con animales que lo despreciaban.
- El cambio:** Con el paso del tiempo, el patito creció y se convirtió en un hermoso **cisne**. "Ahora soy bello", pensó el cisne.
- Conclusión:** Regresó a su hogar y fue bendecido por todos, quienes se sorprendieron al verlo transformado.

La Historia del Patito Feo:

- Un Patito Diferente:** Era una vez una mamá pata que estaba muy emocionada porque iba a tener patitos. Cuando finalmente nacieron, todos eran adorables, excepto uno que era **diferente**. Este patito era más grande y más **feo** que los demás.
- El Rechazo:** Los otros patitos y los animales del estanque se burlaban de él. Se sentía triste y solo, así que decidió alejarse de los demás y buscar su **lugar** en el mundo.
- El Viaje:** Durante su viaje, el patito feo encontró muchos animales, pero nunca se sintió aceptado. Se sentía fuera de lugar y anhelaba encontrar a alguien que lo quisiera.
- La Transformación:** Con el tiempo, el patito feo creció y se convirtió en un hermoso cisne. Cuando se vio en el agua, no podía creer lo que veía. Finalmente, encontró su lugar y fue **aceptado** por todos.
- La Moraleja:** La historia del patito feo nos enseña que la belleza verdadera viene de adentro y que, a veces, aquellos que son **diferentes** son los que terminan siendo los más especiales.

Atributo style

The screenshot shows a storybook titled "La Historia del Patito Feo". The story is about a ugly duckling who is rejected by other ducklings but finds acceptance and happiness. The code for the story includes several examples of the `style` attribute being used to apply CSS directly to HTML elements.

```
<!DOCTYPE html>
<html lang="es">
  <head></head>
  <body>
    <h1 style="background-color: #FFFF00; font-family: Arial, sans-serif; color: #000; text-align: center;">La Historia del Patito Feo</h1>
    <h2 style="text-align: left; font-size: 1.5em; color: #00008B; font-family: 'Times New Roman', serif;"><h2>Un Patito Diferente</h2></h2></h1></h1>
    <p style="text-align: left; margin: 20px; border: 1px solid black; padding: 10px; background-color: #FFFF00; color: #000; font-family: 'Times New Roman', serif;">Erase una vez una mamá pato que estaba muy emocionada porque iba a tener patitos. Cuando finalmente nacieron, todos eran adorables, excepto uno que era diferente. Ese patito era más grande y más feo que los demás.</p>
    <h2 style="text-align: left; font-size: 1.5em; color: #00008B; font-family: 'Times New Roman', serif;"><h2>El Rechazo</h2></h2></h1></h1>
    <p>Los otros patitos y los animales del estanque se burlaban de él. Se sentía triste y solo, así que decidió alejarse de los demás y buscar su lugar en el mundo.</p>
    <h2 style="text-align: left; font-size: 1.5em; color: #00008B; font-family: 'Times New Roman', serif;"><h2>El Viaje</h2></h2></h1></h1>
    <p>Durante su viaje, el patito feo encontró muchos animales, pero nunca se sintió aceptado. Se sentía fuera de su grupo y se sentía solo. Finalmente, encontró su lugar y fue a vivir allí, donde se sentía bien y se sentía aceptado.</p>
    <h2 style="text-align: left; font-size: 1.5em; color: #00008B; font-family: 'Times New Roman', serif;"><h2>La Transformación</h2></h2></h1></h1>
    <p>Con el tiempo, el patito feo creció y se convirtió en un hermoso cisne. Durante su viaje, el patito feo encontró muchos animales, pero nunca se sintió aceptado. Se sentía fuera de su grupo y se sentía solo. Finalmente, encontró su lugar y fue a vivir allí, donde se sentía bien y se sentía aceptado.</p>
    <h2 style="text-align: left; font-size: 1.5em; color: #00008B; font-family: 'Times New Roman', serif;"><h2>La Moraleja</h2></h2></h1></h1>
    <p style="background-color: #FFFF00; color: #000; font-family: 'Times New Roman', serif; border: 1px solid black; padding: 5px; text-align: center; margin: 10px 0; font-size: 0.8em; font-weight: bold;">La historia del patito feo nos enseña que la belleza verdadera viene de dentro y que, a veces, aquellos que son diferentes son los más especiales.</p>
  </body>
</html>
```

En HTML, el atributo **style** permite aplicar reglas de CSS directamente sobre un elemento específico, escribiéndolas dentro de la etiqueta de apertura. La sintaxis consiste en definir una o varias propiedades seguidas de sus valores, separadas por punto y coma, así:

`<etiqueta style="propiedad:valor;">`

Por ejemplo, podemos usar **style** para **aplicar un color de fondo** con la propiedad **background-color**. Si lo usamos en el `<body>`, podemos darle un color a toda la página; en un `<h1>`, podemos destacar un encabezado con un fondo llamativo; y en un `<p>`, diferenciar un párrafo del resto con un color distinto.

The screenshot shows a code editor with a red header bar containing the text "Este es un encabezado (h1)". Below it, there is a blue box containing the text "Este es un párrafo.". The main code area shows the following HTML and CSS:

```
<!DOCTYPE html>
<html lang="es">
  <head></head>
  <body>
    <h1 style="background-color: #008080; color: white; text-align: center; padding: 10px; margin: 0;">Este es un encabezado (h1)</h1>
    <p style="background-color: #ADD8E6; color: black; padding: 10px; margin: 0;">Este es un párrafo.</p>
  </body>
</html>
```

Otra opción es **definir el color de los bordes** con la propiedad *border-color*. Esto permite resaltar un encabezado o un bloque de texto encerrándolo en un marco visible, que además se puede personalizar en grosor y estilo de línea.

Este es un encabezado

```
<!DOCTYPE html>
<html lang="es">
  <head> ... </head>
  ... <body> == $0
    <hr>
    <hr>
    <hr>
    <hr>
    <hr>
    <hr>
    <hr>
    <hr>
    <hr style="border:2px solid Tomato;">Este es un encabezado</h1>
  </body>
</html>
```

Con la propiedad *color*, se puede **modificar el color del texto** de cualquier elemento HTML. Por ejemplo, un título puede mostrarse en azul y un párrafo en rojo, lo que refuerza jerarquía o diferenciación de contenidos.

Este es un encabezado

Este es un párrafo.

```
<!DOCTYPE html>
<html lang="es">
  <head> ... </head>
  ... <body> == $0
    <hr>
    <hr style="color:blue;">Este es un encabezado</h1>
    <p style="color:red;">Este es un párrafo.</p>
  </body>
</html>
```

En cuanto a la tipografía, con *font-family* es posible **cambiar la familia de letras** usada en encabezados y párrafos. Por ejemplo, un título puede mostrarse en Arial mientras un párrafo utiliza Courier, generando contrastes tipográficos.

Este es un encabezado con Verdana

Este es un párrafo con Courier.

```
<!DOCTYPE html>
<html lang="es">
  <head> ... </head>
  ... <body> == $0
    <h1 style="font-family: Verdana;">Este es un encabezado con Verdana</h1>
    <p style="font-family: Courier;">Este es un párrafo con Courier.</p>
  </body>
</html>
```

También se puede **modificar el tamaño del texto** con la propiedad *font-size*. Un encabezado puede crecer al 300% de su tamaño normal para resaltar, mientras que un párrafo puede ajustarse al 160% para mejorar la legibilidad.

Este es un encabezado grande

Este es un párrafo con un tamaño de fuente mayor.

```
<!DOCTYPE html>
<html lang="es">
  <head> ... </head>
  ... <body> == $0
    <h1 style="font-size: 300%;">Este es un encabezado grande</h1>
    <p style="font-size: 160%;">Este es un párrafo con un tamaño de fuente mayor.</p>
  </body>
</html>
```

Por último, la propiedad *text-align* permite **alinear el texto** en diferentes posiciones dentro de la página. Un título puede centrarse para destacarse en la parte superior, mientras que un párrafo puede alinearse a la derecha, dependiendo de la composición buscada.

Título centrado

Párrafo a la derecha.

```
<!DOCTYPE html>
<html lang="es">
  <head> ... </head>
  ...<body> == $0
    <h1 style="text-align: center;">Título centrado</h1>
    <p style="text-align: right;">Párrafo a la derecha.</p>
  </body>
</html>
```

Métodos de aplicación de estilo

Ahora, para usar CSS podemos aplicar los estilos de tres formas distintas. La primera es el **método en línea**, y además existen otros dos: el **método interno** y el **método externo**. Cada uno tiene sus ventajas y se utiliza según la necesidad del proyecto.

1. Estilo en línea

Se aplica directamente dentro de una etiqueta HTML utilizando el atributo style. Esto significa que el estilo afecta únicamente a ese elemento en específico. Este método es útil para pruebas rápidas o para aplicar un estilo muy puntual. Sin embargo, no es recomendable para proyectos grandes porque puede hacer que el código sea difícil de mantener. Es importante recordar que el estilo en línea **tiene la prioridad más alta**: si un mismo elemento tiene varios estilos declarados en diferentes lugares, el que se escribió en línea será el que el navegador muestre.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Estilo en línea</title>
  </head>
  <body>
    <p style="color: red; font-weight: bold;">Este párrafo tiene estilo en línea.</p>
  </body>
</html>
```

2. Estilo interno

Cuando todos los elementos de un mismo documento HTML comparten estilos, podemos definirlos en el encabezado del archivo utilizando la etiqueta <style>. De esta manera, los estilos aplican a todos los elementos del documento que coincidan con las reglas definidas. Es un método más organizado que el estilo en línea, pero sigue estando limitado a un solo archivo HTML.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Estilo interno</title>
    <style>
      p {
        color: green;
        font-style: italic;
      }
    </style>
  </head>
  <body>
    <p>Este párrafo tiene estilo definido en la cabecera del documento HTML.</p>
  </body>
</html>
```

3. Estilo externo

El método más recomendado en proyectos profesionales es trabajar con una **hoja de estilo externa**. En este caso, se escribe el CSS en un archivo aparte con extensión .css y se enlaza al documento HTML mediante el elemento `<link>` en el `<head>`. Este método permite separar completamente el contenido (HTML) del diseño (CSS). Así, si queremos cambiar la apariencia de toda la página o incluso de un sitio completo, basta con modificar el archivo CSS sin tener que tocar el HTML.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Estilo externo</title>
    <link rel="stylesheet" href="estilos.css">
  </head>
  <body>
    <p>Este párrafo usa estilos definidos en un archivo CSS externo.</p>
  </body>
</html>
```

```
css

p {
    color: blue;
    text-decoration: underline;
}
```

Sintaxis de CSS

La sintaxis de CSS sigue la siguiente forma:

Selector { propiedad : valor }

- **Selector:** indica a qué elementos HTML se les aplicará el estilo. Puede ser un elemento por nombre de etiqueta (por ejemplo, p para párrafos), por clase, por ID, o incluso estados especiales mediante pseudo-clases.
- **Propiedad:** define qué aspecto visual del elemento se va a modificar (como color, tamaño de fuente, márgenes, alineación, etc.).
- **Valor:** especifica cómo se va a modificar esa propiedad (por ejemplo, red, 16px, center).

Cada regla CSS se cierra con un punto y coma (;) para separar las propiedades y valores dentro de un mismo bloque.

Selectores

En CSS, los **selectores** son la forma de indicar a qué elementos HTML se les van a aplicar las reglas de estilo. Según el propósito, existen distintos tipos de selectores:

Selector de tipo: Se utiliza para aplicar estilos directamente a todos los elementos de un mismo nombre de etiqueta HTML.

Por ejemplo, si se define un estilo para p, este se aplicará a todos los párrafos de la página.

```
css

p {
    color: blue;
    text-decoration: underline;
}
```

Selector por ID: Permite aplicar estilos a un único elemento específico dentro del documento, ya que un ID debe ser único en toda la página. Se identifica con el símbolo numeral (#) seguido del nombre del ID.

Selector de clase: Permite aplicar estilos a un conjunto de elementos que comparten una misma clase. Se identifica con un punto (.) seguido del nombre de la clase.

Selector de pseudoclase: Se aplica a un elemento **cuando está en cierto estado**, normalmente relacionado con la interacción del usuario o el estado del formulario. Se escriben con un solo dos puntos (:).

Selector de pseudoelemento: Se aplica a **una parte específica de un elemento**. Se escriben con doble dos puntos (::).

```
.titulo-principal {  
    font-size: 24px;  
    font-weight: bold;  
    color: #2c3e50;  
    text-align: center;  
}
```

Atributo class (._{}) / class =“_”)

El atributo **class** en HTML es un identificador que permite agrupar elementos bajo un mismo nombre para aplicarles estilos o comportamientos comunes desde CSS (y también desde JavaScript).

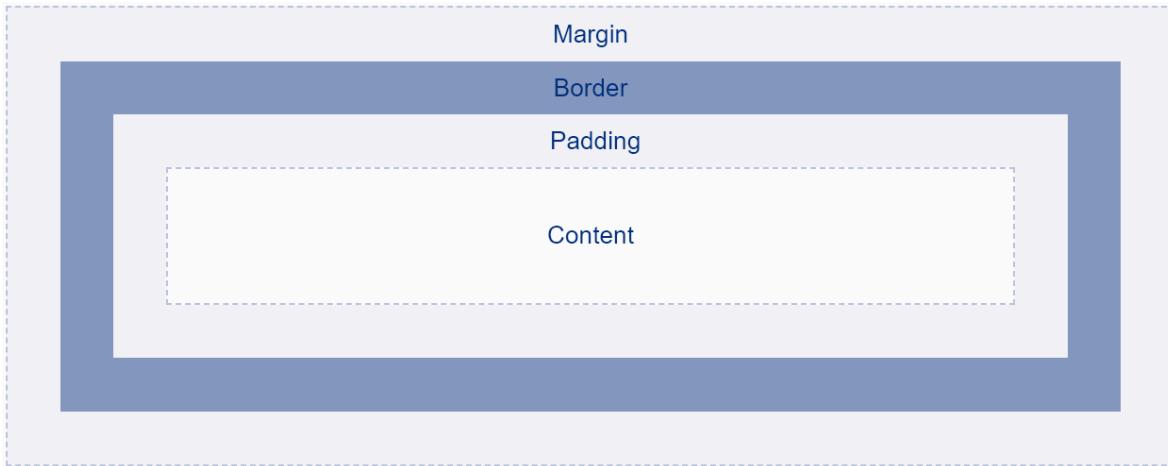
Cuando un elemento HTML tiene una clase asignada, se puede dirigir desde CSS utilizando un punto (.) seguido del nombre de la clase, lo que facilita aplicar un mismo estilo a múltiples elementos de forma consistente.

Ventajas de usar class:

- Permite aplicar estilos de forma uniforme a múltiples elementos.
- Mantiene la consistencia en el diseño reutilizando clases en distintas partes de la página.
- Facilita la organización y mantenimiento del código, evitando repeticiones innecesarias.
- Se integra con otros lenguajes como JavaScript para añadir comportamientos dinámicos.

Box model

En CSS, todos los elementos de una página web se representan como **cajas rectangulares**. Este concepto se conoce como el modelo de caja o “box model”. Este define que cada elemento está compuesto por cuatro áreas principales que se suman entre sí:



- **Content (contenido):**

Es la parte central de la caja, donde se ubica el texto, las imágenes u otros elementos. Su tamaño puede definirse con propiedades como width y height.

- **Padding (relleno):**

Es el espacio entre el contenido y el borde de la caja. Su función es generar separación interna, asegurando que el contenido no quede pegado directamente al borde. El padding aumenta el tamaño total de la caja, aunque el color de fondo del elemento también se aplica sobre él.

- **Border (borde):**

Es el marco que rodea tanto al contenido como al padding. Puede tener grosor, estilo (sólido, punteado, doble, etc.) y color, definidos con propiedades como border-width, border-style y border-color.

- **Margin (margen):**

Es el espacio externo al borde de la caja. Su función es separar un elemento de otros elementos vecinos en la página. A diferencia del padding, el margin no hereda el color de fondo del elemento, siempre se mantiene transparente.

I'm a box

I'm also a box with some text in it.. Lorem ipsum, dolor sit amet consectetur adipiscing elit. Quod molestiae cumque dolores provident! Quo repellat odio rerum, ipsa maiores adipisci veritatis beatae, non ipsam minus dignissimos amet cupiditate nesciunt quas?

- this list is a box
- and all the list items in it are boxes

even [buttons](#) and [links](#) are boxes.

Why bank with us?
Find out how we make your day to day banking easier. Take a look at some of the reasons to bank with ALB.

Mortgages
Check out our range of mortgage options.

Savings & Deposits
We're here to help you start saving today.

Insurance
Get protected for all of life's challenges.

Loans
Check out what loans we have to offer.

Diagramación/layout con CSS

Volviendo al concepto de diagramación, entendiendo que cada elemento en HTML es un bloque y está conformado por una caja (según el modelo de caja), es necesario modificar y definir cómo se organizan y disponen esos elementos en la página, esto lo hacemos con **CSS**, que nos da las propiedades necesarias para controlar la posición y la disposición de los elementos.



Una de las propiedades más importantes para esto es ***display***, que define cómo se debe mostrar un elemento en el flujo del documento. Con *display* podemos decidir si un elemento ocupa toda la línea, si se alinea junto a otros o si combina comportamientos.

- **block**: el elemento ocupa toda la línea disponible, empujando a los demás hacia abajo. Ejemplo: <div>.
- **inline**: el elemento se muestra en la misma línea que otros, sin forzar saltos de línea. Ejemplo: .
- **inline-block**: combina ambos comportamientos; el elemento se alinea en línea, pero conserva la posibilidad de tener dimensiones propias como un bloque.



```
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ejemplo de Div con Estilos Inline y CSS</title>
  <style> .div {
    width: 300px;
    display: inline-block;
    margin-right: 1.25px;
    text-align: center;
    line-height: 200px;
    height: 200px;
    color: white;
  }</style>
</head>
<body>
  <!-- Bloque 1 -->
  <div style="background-color: #3498db;"> Bloque 1 </div>
  <!-- Bloque 2 -->
  <div style="background-color: #e67e22;"> Bloque 2 </div>
  <!-- Bloque 3 -->
  <div style="background-color: #2ecc71;"> Bloque 3 </div>
</body>
</html>
```

Otra propiedad clave para el control del layout es ***position***, que define cómo se coloca un elemento dentro del flujo de la página en relación con su contenedor o con la ventana del navegador:

- **static**: es el valor por defecto. El elemento se posiciona siguiendo el flujo normal de la página, sin desplazamientos especiales.
- **relative**: el elemento se coloca en relación con su posición original en el flujo. A partir de ahí, se puede mover usando propiedades como top, left, right o bottom.
- **absolute**: el elemento se posiciona de manera absoluta respecto a su contenedor más cercano que tenga una posición distinta de static. Sale del flujo normal y no afecta al resto de los elementos.
- **fixed**: el elemento queda fijado en relación con la ventana del navegador, sin importar el scroll. Es común usarlo en menús o barras fijas.
- **sticky**: combina características de relative y fixed. El elemento se comporta como relativo hasta que se alcanza un punto específico en el scroll, a partir del cual queda fijo en pantalla.

Además de controlar la posición de un elemento en el eje horizontal y vertical, también se puede controlar su orden en el eje “Z”, es decir, la superposición de los elementos. Para esto se utiliza la propiedad ***z-index***.

El z-index define qué elemento aparece por encima o por debajo de otros cuando se solapan en la pantalla. Funciona como una especie de “nivel” en capas:

- Un valor mayor significa que el elemento estará por encima.
- Un valor menor significa que quedará por debajo.
- Si no se especifica, los elementos se apilan en el orden natural en que aparecen en el código HTML.

Es importante notar que z-index **solo tiene efecto en elementos que tienen un valor de position distinto a static** (por ejemplo, relative, absolute, fixed o sticky).

Flexbox

Es un sistema de diagramación diseñado para organizar elementos dentro de un contenedor de manera flexible. Se le llama **unidimensional** porque trabaja en un solo eje a la vez, ya sea horizontal o vertical, según lo definamos. Esto permite controlar con precisión cómo se alinean, distribuyen y adaptan los elementos al espacio disponible.

Para activar Flexbox se utiliza la propiedad display:

- **display: flex** convierte al contenedor en un sistema flexible.

- **display: inline-flex** activa el mismo sistema, pero mantiene el comportamiento de un elemento en línea.

Una vez activado, se pueden definir varias propiedades clave:

- **flex-direction**: indica la dirección principal del sistema. Puede ser row (fila), row-reverse (fila invertida), column (columna) o column-reverse (columna invertida).
- **justify-content**: distribuye los elementos a lo largo del eje principal. Las opciones más comunes son flex-start, flex-end, center, space-between y space-around.
- **align-items**: alinea los elementos en el eje perpendicular (por ejemplo, si el eje principal es horizontal, aquí se controla la alineación vertical). Los valores incluyen stretch, center, baseline, flex-start y flex-end.
- **gap**: define el espacio entre los elementos de manera uniforme, pudiendo expresarse en pixeles o en porcentaje.

Grid Layout

Grid, es más bien, un sistema de diseño bidimensional que **permite organizar los elementos en filas y columnas dentro de un contenedor**. A diferencia de Flexbox, que trabaja en un solo eje a la vez, Grid ofrece control sobre ambos ejes (horizontal y vertical) de manera simultánea.

Para activar este sistema se utiliza la propiedad display:

- **display: grid** convierte al contenedor en una grilla.
- **display: inline-grid** activa la grilla pero mantiene el comportamiento de un elemento en línea.

Una vez definido el contenedor como grilla, se pueden usar distintas propiedades para controlar la estructura:

- **grid-template-columns**: define el número y tamaño de las columnas. Puede expresarse en unidades fijas (px), relativas (fr), automáticas (auto) o mediante funciones como repeat(cantidad, tamaño).
- **grid-template-rows**: define el número y tamaño de las filas, utilizando las mismas unidades o funciones que en columnas.
- **grid-column** y **grid-row**: determinan en qué línea comienza y en cuál termina un elemento dentro de la grilla, lo que permite posicionarlo en un área específica.

Además, Grid incluye propiedades de alineación:

- **justify-content**: ajusta la distribución de las columnas a lo largo del eje horizontal.
- **align-items**: alinea los elementos verticalmente dentro de cada celda.
- **place-items**: combina alineación horizontal y vertical en una sola propiedad.
- **Gap**: permite definir el espacio entre filas y columnas de forma uniforme, expresado en px o en %.

sin importar si accede desde un monitor grande, una tablet o un smartphone.

Unidades y medidas

Muchas propiedades necesitan valores que se expresan en **unidades de medida**. Existen dos grandes tipos de unidades en CSS:

Unidades absolutas

Son fijas y no cambian en función del contexto o del dispositivo. Siempre representan el mismo valor, sin importar el tamaño de pantalla, la resolución o la configuración del usuario.

Estas son:

- **px (píxeles)**
- **cm, mm, in (centímetros, milímetros, pulgadas)**
- **pt (puntos) y pc (picas)**

Estas, aunque pueden ser más precisas, no se adaptan bien a diferentes pantallas o configuraciones de accesibilidad, lo cuál puede ser un problema para el diseño responsive.

Unidades relativas

Se ajustan en función del contexto del elemento o del dispositivo. Su valor depende de otra medida de referencia (como el tamaño de fuente del parente, el ancho de la ventana o el elemento contenedor).

Estas pueden ser:

- **em**: relativa al tamaño de fuente del elemento parente.
- **rem (root em)**: relativa al tamaño de fuente definido en el elemento raíz (html).
- **% (porcentaje)**: relativo al tamaño del contenedor del elemento.

- **vw (viewport width)**: relativo al ancho de la ventana del navegador.
- **vh (viewport height)**: relativo al alto de la ventana del navegador.
- **vmin / vmax**: relativo al lado más pequeño o más grande de la ventana.

Estas son la base para el diseño responsivo , ya que permiten que la página se vea correctamente en distintos dispositivos y resoluciones.

Responsive design

Es una estrategia que busca que las interfaces se adapten de manera automática a distintos tamaños de pantalla y dispositivos. En lugar de crear versiones separadas de un sitio para escritorio, tablet o móvil, se construye una sola interfaz capaz de reorganizarse, redimensionarse y redistribuir sus elementos según el espacio disponible.

Esto se logra principalmente a través de **unidades relativas** (como %, em, rem, vh, vw) y del uso de **media queries** en CSS, que permiten definir estilos específicos dependiendo del ancho o características del dispositivo.

Enfoque mobile first

Plantea diseñar y desarrollar **primero para dispositivos móviles** (pantallas pequeñas) y luego ir **escalando hacia pantallas más grandes**.

La lógica detrás de esta estrategia es que:

- Los dispositivos móviles suelen ser el principal punto de acceso a internet.
- Obliga a priorizar lo esencial: contenido claro, navegación simple y carga rápida.
- Al construir desde lo más restringido (móvil) hacia lo más amplio (desktop), se asegura una experiencia sólida en todos los contextos.

Concepto de *breakpoints*

Se refiere a los puntos de quiebre que definen en qué momento el diseño debe ajustarse a un nuevo conjunto de reglas de estilo en función de los dispositivos o tamaños de pantalla más comunes.

Algunos ejemplos frecuentes de breakpoints son:

- **320px – 480px**: teléfonos móviles.
- **768px**: tablets en orientación vertical.
- **1024px**: tablets en orientación horizontal o pantallas pequeñas de laptops.

- **1200px o más:** pantallas grandes de escritorio.

Media queries (@media)

Las media queries son las reglas de CSS que permiten aplicar estilos condicionales según las características del dispositivo o la ventana del navegador. Básicamente, son la manera en la que le indigamos al navegador cuándo tiene que aplicar un estilo u otros. Algo como decirle al navegador: *"si la pantalla tiene estas características, aplique estos estilos"*.

La más común es el ancho de pantalla, aunque también se pueden usar otras propiedades como la orientación. **Por ejemplo: se puede indicar que un menú horizontal se convierta en un menú desplegable cuando la pantalla sea menor a cierto tamaño.**

```
/* Breakpoint para pantallas grandes (desktops, >=1024px) */
@media (min-width: 1024px) {
    body {
        background-color: lightgreen;
    }

    h1 {
        font-size: 2.5rem;
        text-align: left;
    }
}
```