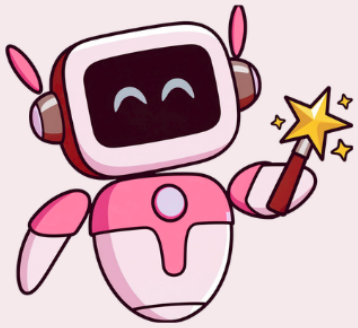
Especialización en
Programador Web

HTML & CSS

Material teórico complementario



Prof. Katherine Molina

Esta guía explica paso a paso las etiquetas de HTML y las propiedades de CSS, con ejemplos de código listos para probar en el simulador. Usala como material de consulta a lo largo de la cursada. No hace falta memorizar todo, usala como material de consulta a lo largo de la cursada.

Índice

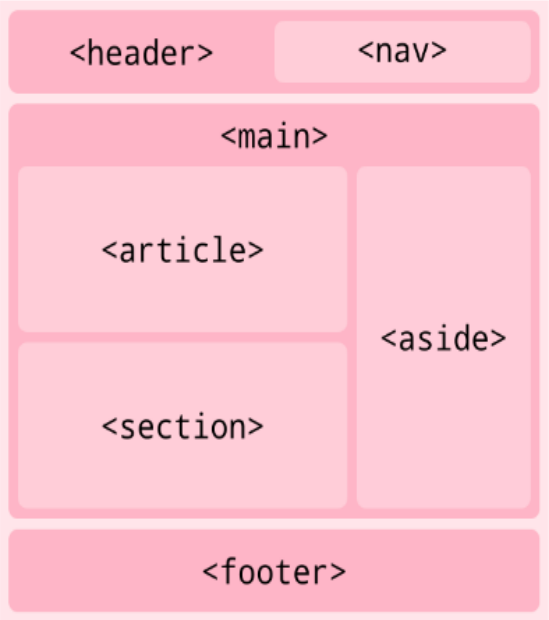
Parte 1 — Conceptos previos

1. ¿Cómo está estructurada una página web?

Antes de programar, conviene conocer las **zonas** que suele tener una página web. Pensá una página como un edificio con sectores bien definidos: arriba el encabezado, en el medio el contenido, abajo el pie.

Qué es cada una:

- **header:** la franja de arriba. Suele tener el logo y el nombre del sitio.
- **nav:** el menú con los enlaces de navegación (Inicio, Productos, Contacto…).
- **main:** el contenido principal y único de la página.
- **section:** una sección temática dentro del contenido (por ejemplo “Nuestros servicios”).
- **article:** un contenido que tiene sentido por sí solo (una nota de blog, una noticia, una tarjeta de producto).
- **aside:** una barra lateral con contenido secundario (publicidad, enlaces relacionados).
- **footer:** el pie de página, con datos de contacto, redes sociales o derechos de autor.



2. Diseño responsive

Responsive (adaptable) significa que una página web se **adapta automáticamente** al tamaño de la pantalla en la que se ve: una computadora de escritorio, una notebook, una tablet o un celular.

Una página responsive **no se ve igual de chica en el celular** que en la compu: los elementos se reacomodan, las imágenes se achican, los menús se transforman, para que siempre se lea bien y sea cómoda de usar.

Hoy es **imprescindible**, porque la mayoría de las personas navegan desde el celular. Más adelante (Parte 5) vamos a ver la herramienta que lo hace posible: las **media queries**.

3. Mobile First

Mobile First (“primero el móvil”) es una forma de trabajar que consiste en **diseñar y programar primero la versión para celular**, y recién después adaptarla a pantallas más grandes.

¿Por qué se hace así?

- La mayoría del tráfico web hoy viene de celulares.
- Es más fácil **agrandar** un diseño simple que **achicar** uno complejo.
- Obliga a priorizar lo más importante (en una pantalla chica entra menos).

En la práctica, empezás escribiendo los estilos para pantalla chica y luego vas **agregando** ajustes para pantallas más grandes con media queries.

4. Convenciones de escritura de nombres

Cuando programamos, constantemente le ponemos **nombres** a las cosas (clases, archivos, etc.). Como no se pueden usar espacios, existen distintas convenciones para unir varias palabras:

Convención	Cómo se escribe	Ejemplo
kebab-case	minúsculas unidas con guion medio	menu-principal
snake_case	minúsculas unidas con guion bajo	menu_principal
camelCase	unidas, cada palabra (menos la 1ª) con mayúscula	menuPrincipal
PascalCase	igual que camelCase pero la 1ª también con mayúscula	MenuPrincipal

¿Cuál usamos en este curso? Para **HTML y CSS** la convención estándar es **kebab-case** (minúsculas con guion medio). La vamos a usar para nombres de clases, IDs y archivos:

```
<!-- Correcto: kebab-case -->
<section class="caja-producto"></section>

<!-- Evitar en HTML/CSS -->
<section class="cajaProducto"></section>
<section class="caja_producto"></section>
```

> **Nota.** el `camelCase` se usa sobre todo en **JavaScript** y el `snake_case` en lenguajes como **Python** o en **bases de datos**. Pero para la web (HTML/CSS), **kebab-case**.

5. Convenio de archivos

Los archivos de tu sitio pueden tener cualquier nombre, pero por convención deben cumplir estas reglas:

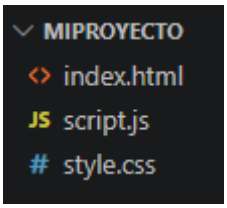
1. **Sin espacios, acentos, eñes ni símbolos.** Si son varias palabras, unilas con guion medio `-` o guion bajo `_`. Ejemplo: `mi-pagina-web`.
2. Deben estar escritos **en minúsculas**.
3. Los documentos web llevan la extensión `.html` **(así el servidor sabe que es una página web).**
4. Las hojas de estilo llevan la extensión `.css`**.**
5. Los archivos de JavaScript llevan la extensión `.js`**.**

> **Atención.** en **Windows**, el sistema oculta las extensiones conocidas, así que podés terminar con una **doble extensión** sin darte cuenta (por ejemplo `pagina.html.txt`, que NO es una página web).

Dos convenciones de nombres muy importantes:

- `index.html` **es el archivo que el navegador y el servidor reconocen como la** página principal o de inicio **de un sitio.**

- Por costumbre, la hoja de estilos principal suele llamarse `style.css` **y el archivo de JavaScript principal** `script.js`**.**



Parte 2 — HTML: la estructura

6. ¿Qué es HTML?

HTML significa *HyperText Markup Language* (Lenguaje de Marcado de Hipertexto). Es el lenguaje que se usa para **crear la estructura y el contenido** de una página web: los títulos, los párrafos, las imágenes, los enlaces, los botones.

HTML **no es un lenguaje de programación**, sino de **marcado**: no hace cálculos ni toma decisiones, simplemente "marca" qué es cada cosa (esto es un título, esto es un párrafo, esto es una imagen).

Es la base de toda página web, el esqueleto sobre el que después se aplica el estilo (CSS) y el comportamiento (JavaScript).

7. Etiquetas (tags)

El HTML se escribe con **etiquetas** (en inglés, *tags*). Una etiqueta es una palabra encerrada entre los signos `<` y `>` que le indica al navegador qué tipo de contenido está mostrando.

La mayoría de las etiquetas vienen **de a pares**: una de **apertura** y una de **cierre** (igual pero con una barra `/`). El contenido va en el medio.

```
<p>Esto es un parrafo de texto.</p>
```

- `<p>` → etiqueta de **apertura**.
- `Esto es un parrafo de texto.` → el **contenido**.
- `</p>` → etiqueta de **cierre**.

Algunas etiquetas **no tienen contenido y se cierran solas** (se llaman *vacías* o *autocontenidas*). No llevan etiqueta de cierre:

```
<br>                    <!-- salto de linea -->
<hr>                    <!-- linea horizontal -->
<img src="foto.jpg">    <!-- imagen -->
```

8. Anidar etiquetas

Se puede meter una etiqueta **dentro de otra** (anidar); es mucho más común de lo que parece. IMPORTANTE: las etiquetas **siempre se cierran en orden inverso al de apertura** (la última que abrís es la primera que cerrás).

```
<!-- Correcto: se cierran en orden inverso -->
<p>Hola <strong>mundo</strong></p>

<!-- Incorrecto: cierre cruzado -->
<p>Hola <strong>mundo</p></strong>
```

Un ejemplo con varios niveles de anidación:

```
<ul>
  <li>Texto con una parte en <strong>negrita</strong></li>
  <li>Otro item de la lista</li>
</ul>
```

9. Estructura básica de un documento HTML

Todo archivo HTML tiene siempre la misma estructura mínima. Este es el “molde” con el que empieza cualquier página:

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="UTF-8">
    <title>Mi primera pagina</title>
  </head>
  <body>
    <h1>Hola, mundo!</h1>
    <p>Este es el contenido de mi pagina.</p>
  </body>
</html>
```

Qué hace cada parte:

- `<!DOCTYPE html>` → le avisa al navegador que es un documento HTML moderno (HTML5). Va siempre en la primera línea.
- `<html>` → la etiqueta que envuelve todo el documento. El `lang="es"` indica que está en español.
- `<head>` → la “cabecera”: contiene información que no se ve en la página (el título de la pestaña, los enlaces a los archivos CSS).
- `<meta charset="UTF-8">` → permite que se vean bien los acentos y la ñ.
- `<title>` → el texto que aparece en la pestaña del navegador.
- `<body>` → el “cuerpo”: acá va todo lo que se ve en la página (textos, imágenes, botones, etc.).

> **Nota.** en el simulador del curso, ustedes solo escriben lo que iría **dentro del** `<body>`**, porque el resto ya está puesto por detrás.**

10. Etiquetas semánticas

Las **etiquetas semánticas** son etiquetas cuyo nombre **describe el tipo de contenido** que tienen adentro. Sirven para organizar la página en las zonas que vimos en el punto 1.

```
<body>
  <header>
    <h1>Mi sitio web</h1>
    <nav>
      <a href="#">Inicio</a>
      <a href="#">Productos</a>
      <a href="#">Contacto</a>
    </nav>
  </header>

  <main>
    <section>
      <h2>Bienvenidos</h2>
      <p>Este es el contenido principal.</p>
    </section>

    <article>
      <h2>Ultima noticia</h2>
      <p>Texto de la noticia...</p>
    </article>
  </main>

  <aside>
    <p>Contenido lateral (publicidad, enlaces).</p>
  </aside>

  <footer>
    <p>2026 Mi sitio web</p>
  </footer>
</body>
```

¿Por qué usarlas en vez de `<div>` para todo?

- Hacen el código **más ordenado y fácil de leer**.
- Ayudan a **Google** a entender la página (mejor posicionamiento).
- Mejoran la **accesibilidad** (lectores de pantalla para personas con discapacidad visual).

11. Encabezados y párrafos

Los **encabezados** son los títulos y subtítulos. Hay **seis niveles**, de `<h1>` (el más importante) a `<h6>` (el menos importante). Los **párrafos** se escriben con `<p>`.

```
<h1>Titulo principal</h1>
<h2>Subtitulo</h2>
<h3>Sub-subtitulo</h3>

<p>Este es un parrafo con texto normal de la pagina.</p>
<p>Y este es otro parrafo distinto.</p>
```

> **Nota.** debe haber **un solo** `<h1>` por página (es el título principal). Los demás niveles se usan para organizar el contenido en orden de importancia, como el índice de un libro.

12. Texto de relleno: Lorem Ipsum

Cuando estás armando una página y todavía no tenés el texto definitivo, se usa un texto de relleno (en inglés, *placeholder text*) para no dejar los espacios vacíos. El más famoso es el **Lorem Ipsum**.

¿Qué es? Es un texto en un latín "falso" (sin significado real) que imita cómo se ven las palabras y los párrafos de verdad. Empieza siempre con *"Lorem ipsum dolor sit amet…"* y se usa en diseño desde hace siglos, heredado de la industria de la imprenta.

¿Para qué sirve y cuándo se usa?

- Para **rellenar párrafos** mientras armás la estructura o el diseño, cuando el contenido real todavía no está listo.
- Para ver **cómo se comporta el diseño** con texto: cuánto ocupa, cómo se acomoda, cómo quedan los espacios y los saltos de línea.
- Para **no distraerte** leyendo: como no significa nada, te concentrás en la maqueta y no en el contenido.

> **Nota.** ¿por qué no escribir "aaa" o repetir una palabra? Porque el Lorem Ipsum tiene palabras de largos variados y una distribución parecida a la de un texto real, así el diseño se ve natural.

Cómo conseguirlo (atajos):

- En editores de código con **Emmet** (como VS Code), escribís `lorem` y apretás **Tab**: se genera un párrafo automáticamente. Para una cantidad exacta de palabras, agregás un número: `lorem20` + Tab genera 20 palabras.
- También combinado: `p>lorem` (un párrafo con relleno) o `ul>li*3>lorem5` (una lista de 3 ítems con 5 palabras cada uno).
- Si tu editor no tiene Emmet, hay **generadores online** (por ejemplo, lipsum.com) donde elegís la cantidad y lo copiás.

Ejemplo ya generado:

```
<h2>Sobre nosotros</h2>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.</p>
```

13. Formato de texto

Existen etiquetas para **resaltar** o dar formato a partes del texto:

```
<p>Este texto tiene una palabra en <strong>negrita</strong>.</p>
<p>Este texto tiene una palabra en <em>cursiva</em>.</p>
<p>Primera linea<br>Segunda linea (despues de un salto).</p>
<hr>
<p>El contenido de arriba quedo separado por una linea.</p>
```

- `<strong>` → **texto en** negrita **(e indica importancia).**
- `<em>` → **texto en *cursiva* (e indica énfasis).**
- `<br>` → **un** salto de línea **(se cierra solo).**
- `<hr>` → **una** línea horizontal **divisoria (se cierra sola).**

> **Nota.** también existen `<b>` (negrita) e `<i>` (cursiva), que se ven igual pero no aportan significado. Hoy se recomienda usar `<strong>` y `<em>`.

14. Listas

Hay dos tipos principales de listas. En ambas, cada elemento se escribe con `<li>` (*list item*):

- **Lista desordenada** (`<ul>`): con viñetas (puntitos). Para ítems sin orden.
- **Lista ordenada** (`<ol>`): con números. Para ítems en secuencia.

```
<!-- Lista desordenada -->
<ul>
  <li>Manzanas</li>
  <li>Peras</li>
  <li>Bananas</li>
</ul>

<!-- Lista ordenada -->
<ol>
  <li>Encender la computadora</li>
  <li>Abrir el navegador</li>
  <li>Escribir la direccion</li>
</ol>
```

15. Enlaces (links)

Los **enlaces** permiten ir de una página a otra (o a otro sitio). Se hacen con la etiqueta `<a>` y el atributo `href`, que indica el **destino**.

```
<!-- Enlace a un sitio externo -->
<a href="https://www.google.com">Ir a Google</a>

<!-- Enlace a otra pagina del mismo sitio -->
<a href="contacto.html">Ir a Contacto</a>

<!-- Enlace que abre en una pestaña nueva -->
<a href="https://www.google.com" target="_blank">Abrir en pestaña nueva</a>
```

- `href` → la dirección de destino.
- `target="_blank"` → hace que el enlace se abra en una pestaña nueva.

16. Imágenes

Las imágenes se insertan con la etiqueta `<img>`, que **se cierra sola**. Sus dos atributos principales son `src` y `alt`.

```
<img src="logo.png" alt="Logo de la empresa">

<!-- Imagen desde internet -->
<img src="https://picsum.photos/200" alt="Imagen de ejemplo">
```

- `src` (*source*) → la ruta o dirección de la imagen (un archivo propio o una URL de internet).
- `alt` (*alternative text*) → un texto descriptivo que se muestra si la imagen no carga. Es muy importante para la accesibilidad y para Google.

17. Atributos

Todas las etiquetas aceptan **atributos**. Un atributo es cualquier **característica** que puede ser diferente entre una etiqueta y otra (por ejemplo, en una imagen, el ancho o el alto). Se escriben con la forma `atributo="valor"`:

```
<etiqueta atributo="valor">
  Contenido
</etiqueta>
```

Reglas importantes:

- El **valor** siempre va entre **comillas**.
- Una etiqueta puede tener **más de un atributo**, separados por **espacios** entre sí.
- Los atributos **solo van en la etiqueta de apertura**, nunca en la de cierre.

```
<img src="foto.jpg" alt="Una foto" width="300">
<a href="pagina.html" target="_blank">Un enlace</a>
<p id="intro" class="texto-destacado">Un parrafo</p>
```

Atributos muy comunes:

- `href` → destino de un enlace.
- `src` → fuente de una imagen.
- `alt` → texto alternativo.
- `id` → un identificador único para un elemento (no se puede repetir en la página).
- `class` → una categoría que se puede repetir en varios elementos. Es la que más se usa para aplicar estilos con CSS.

> **Nota.** la diferencia entre `id` y `class` es clave para el CSS: el `id` es único (como el DNI de una persona), la `class` se comparte (como “alumno”, que aplica a muchos).

18. Contenedores genéricos: div y span

Cuando ninguna etiqueta semántica encaja, se usan dos contenedores “neutros” que no tienen significado propio y sirven solo para **agrupar** elementos y aplicarles estilos:

- `<div>` → contenedor de bloque: ocupa todo el ancho disponible. Se usa para agrupar secciones enteras (como el “hero”).
- `<span>` → contenedor en línea: ocupa solo lo que mide su contenido. Sirve para resaltar o estilizar una palabra dentro de un texto.

```
<div class="hero">
  <h1>Bienvenidos</h1>
</div>

<p>El precio es <span class="destacado">$1500</span> por unidad.</p>
```

19. Tablas

Las **tablas** sirven para mostrar datos organizados en filas y columnas. Se arman combinando varias etiquetas:

- `<table>` → **la tabla completa.**

- `<tr>` **(*table row*) → una** fila**.**
- `<th>` **(*table header*) → una celda de** encabezado **(se ve en negrita y centrada).**
- `<td>` **(*table data*) → una celda de** datos **normal.**

```
<table>
  <tr>
    <th>Nombre</th>
    <th>Edad</th>
    <th>Ciudad</th>
  </tr>
  <tr>
    <td>Ana</td>
    <td>25</td>
    <td>Buenos Aires</td>
  </tr>
  <tr>
    <td>Luis</td>
    <td>30</td>
    <td>Cordoba</td>
  </tr>
</table>
```

20. Formularios

Los **formularios** permiten que el usuario **ingrese y envíe información** (un login, un registro, una búsqueda, un contacto). Se arman con la etiqueta `<form>` y distintos campos adentro.

```
<form>
  <label for="nombre">Nombre:</label>
  <input type="text" id="nombre" name="nombre" placeholder="Escribi tu nombre" required>

  <label for="email">Email:</label>
  <input type="email" id="email" name="email" placeholder="tucorreo@ejemplo.com">

  <label for="clave">Contraseña:</label>
  <input type="password" id="clave" name="clave">

  <label for="mensaje">Mensaje:</label>
  <textarea id="mensaje" name="mensaje"></textarea>

  <label for="pais">Pais:</label>
  <select id="pais" name="pais">
    <option>Argentina</option>
    <option>Uruguay</option>
    <option>Chile</option>
  </select>

  <label>
    <input type="checkbox" name="acepto"> Acepto los terminos
  </label>

  <button type="submit">Enviar</button>
</form>
```

Las etiquetas más usadas en formularios:

- `<form>` → el contenedor de todo el formulario.

- `<label>` → la etiqueta de texto que describe un campo. El atributo `for` lo conecta con el `id` de su campo.
- `<input>` → un campo de entrada (se cierra solo). Es la etiqueta más versátil: cambia según su `type`.
- `<textarea>` → un campo de texto grande (para mensajes largos).
- `<select>` / `<option>` → una lista desplegable de opciones.
- `<button>` → un botón; con `type="submit"` envía el formulario.

Los tipos de `input` (atributo `type`) más comunes:

- `text` (texto libre) · `email` (correo) · `password` (oculta lo escrito) · `number` (números).
- `checkbox` (casilla de tildar) · `radio` (opción única entre varias) · `date` (fecha) · `file` (subir archivo).

Atributos típicos de los campos:

- `name` → el nombre del campo; es el que se envía al servidor con el dato.
- `placeholder` → un texto de ayuda que aparece dentro del campo antes de escribir.
- `required` → hace que el campo sea obligatorio (no deja enviar el formulario si está vacío).
- `value` → un valor por defecto que ya viene cargado en el campo.

21. Comentarios en HTML

Los **comentarios** son notas que escribís en el código **pero que no se ven** en la página. Sirven para aclarar, organizar o "apagar" temporalmente una parte del código. Se escriben entre `<!--` y `-->`.

```
<!-- Esto es un comentario y no se mostrara en la pagina -->

<h1>Titulo visible</h1>

<!--
  Tambien se pueden escribir
  comentarios de varias lineas
-->

<!-- <p>Este parrafo esta "apagado": no se muestra.</p> -->
```

Parte 3 — CSS: el estilo

22. ¿Qué es CSS?

CSS significa *Cascading Style Sheets* (Hojas de Estilo en Cascada). Es el lenguaje que se usa para darle **apariencia** a una página web: los colores, los tamaños, las fuentes, los espacios, las posiciones.

Si el **HTML es la estructura** (el esqueleto), el **CSS es el estilo** (la ropa). El HTML dice *qué* es cada cosa; el CSS dice *cómo se ve*.

La idea central es **separar** la estructura del diseño: el contenido va en el HTML y la apariencia va en el CSS. Así, podés cambiar todo el look de un sitio sin tocar el contenido.

23. Las 3 formas de aplicar CSS

1. CSS en línea (*inline*) — dentro de la propia etiqueta, con el atributo `style`. Sirve para algo muy puntual, pero no se recomienda porque desordena el HTML.

```
<p style="color: red;">Texto rojo</p>
```

2. CSS interno (*internal*) — dentro de una etiqueta `<style>` en el `<head>`. Útil para una sola página.

```
<head>
  <style>
    p {
      color: red;
    }
  </style>
</head>
```

3. CSS externo (*external*) — en un archivo aparte `.css` enlazado desde el HTML. Es la forma recomendada: mantiene todo ordenado y permite reutilizar el estilo en muchas páginas.

```
<!-- En el HTML, dentro del <head> -->
<link rel="stylesheet" href="style.css">

/* En el archivo style.css */
p {
  color: red;
}
```

> **Nota.** en el simulador, el panel de CSS funciona como si fuera CSS interno/externo: lo que escribís ahí se aplica automáticamente a tu HTML.

24. Anatomía de una regla CSS

Una **regla CSS** le dice al navegador: “a este elemento, aplicale este estilo”. Tiene tres partes: **selector**, **propiedad** y **valor**.

```
selector {
  propiedad: valor;
}

/* Ejemplo real */
h1 {
  color: blue;
  font-size: 32px;
}
```

- **Selector** (`h1`) → **a quién** se le aplica el estilo.
- **Propiedad** (`color`, `font-size`) → **qué** se quiere cambiar.
- **Valor** (`blue`, `32px`) → **cómo** se quiere cambiar.

Detalles de sintaxis: la propiedad y el valor se separan con **dos puntos** `:`; cada línea termina con **punto y coma** `;`; y todo va entre **llaves** `{ }`.

25. Selectores

El **selector** indica a qué elementos se aplica la regla. Los tres básicos son:

1. Selector de etiqueta — apunta a todas las etiquetas de ese tipo.

```
p {
  color: gray;
}
```

2. Selector de clase — apunta a los elementos con esa `class`. Se escribe con un **punto** `.` adelante.

```
<p class="destacado">Texto importante</p>

.destacado {
  color: red;
  font-weight: bold;
}
```

3. Selector de ID — apunta al elemento con ese `id`. Se escribe con un **numeral** `#` adelante.

```
<h1 id="titulo-principal">Bienvenidos</h1>

#titulo-principal {
  color: green;
}
```

> **Nota.** en la práctica, lo que más se usa es el **selector de clase**, porque es flexible y reutilizable.

26. Colores

En CSS los colores se pueden escribir de varias formas:

```
/* 1. Por nombre */
color: red;
color: hotpink;

/* 2. Hexadecimal (HEX) - la mas usada */
color: #ff4354;   /* coral */
color: #ffdde6;   /* rosa claro */
color: #000000;   /* negro */
color: #ffffff;   /* blanco */

/* 3. RGB (Rojo, Verde, Azul de 0 a 255) */
color: rgb(255, 67, 84);   /* el mismo coral */
```

Dos propiedades donde más se usan los colores: `color` **(color del texto) y** `background-color` **(color de fondo).**

```
.boton {
  color: #ffffff;            /* texto blanco */
  background-color: #ff4354; /* fondo coral */
}
```

> **Nota.** el código HEX es el que vas a ver en herramientas como Canva o Figma. Por eso conviene acostumbrarse a usarlo.

27. Texto y fuentes

CSS ofrece muchas propiedades para dar formato al texto:

```
p {
  font-family: Arial, sans-serif;  /* tipo de letra */
  font-size: 18px;                 /* tamaño */
  font-weight: bold;               /* grosor (normal / bold) */
  font-style: italic;              /* estilo (normal / italic) */
  color: #333333;                  /* color del texto */
  text-align: center;              /* left, center, right, justify */
  line-height: 1.5;                /* espacio entre lineas */
  text-decoration: underline;      /* subrayado, o "none" */
}
```

- `font-family` → el tipo de letra (se pone una opción y un respaldo: `Arial, sans-serif`).
- `font-size` → el tamaño (normalmente en `px`).
- `font-weight` → el grosor (`normal` o `bold`).
- `text-align` → la alineación horizontal (`left`, `center`, `right`, `justify`).

28. Fondos

Las propiedades de **fondo** permiten poner un color o una imagen detrás de un elemento.

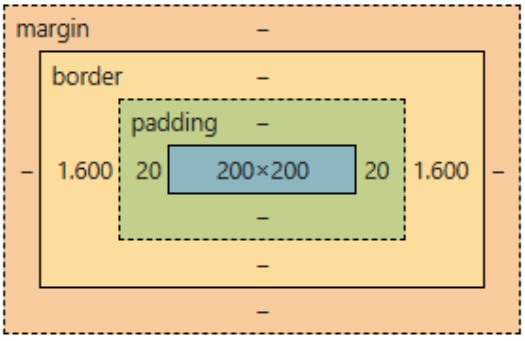
```
body {
  background-color: #ffdde6;   /* color de fondo */
}

.banner {
  background-image: url("fondo.jpg");  /* imagen de fondo */
  background-size: cover;              /* que cubra todo el espacio */
  background-position: center;         /* centrada */
  background-repeat: no-repeat;        /* que no se repita */
}
```

- `background-color` → color de fondo (nombres, HEX o RGB).
- `background-image` → imagen de fondo, con `url("...")`.
- `background-size: cover` → la imagen cubre todo el elemento sin deformarse.
- `background-repeat: no-repeat` → evita que la imagen se repita en mosaico.

29. El modelo de caja (box model)

Este es uno de los conceptos **más importantes** de CSS. En CSS, **todo elemento es una caja rectangular**, y esa caja tiene cuatro capas, de adentro hacia afuera:



- **Contenido → el texto o la imagen en sí.**
- `padding` **(relleno) → el espacio interno, entre el contenido y el borde.**
- `border` **(borde) → la línea que rodea la caja.**
- `margin` **(margen) → el espacio externo, que separa la caja de otros elementos.**

```
.tarjeta {
  width: 250px;              /* ancho del contenido */
  height: 150px;             /* alto del contenido */
  padding: 20px;             /* espacio interno */
  border: 2px solid #ff4354; /* grosor, estilo y color del borde */
  margin: 30px;              /* espacio externo */
  background-color: #ffdde6;
}
```

> **Nota.** entender el box model es lo que te permite controlar los **espacios** de una página. Es la base de todo el maquetado.

30. Comentarios en CSS

Igual que en HTML, podés escribir **comentarios** que no afectan el estilo. En CSS se escriben entre `/*` y `*/`.

```
/* Este es un comentario en CSS */

h1 {
  color: #ff4354;  /* color coral para el titulo */
}

/*
  Los comentarios tambien
  pueden ocupar varias lineas
*/
```

Parte 4 — Maquetado (layout)

31. Flexbox

Flexbox (caja flexible) es una herramienta de CSS para **acomodar elementos en una fila o en una columna** y distribuir el espacio entre ellos de forma fácil. Es ideal para alinear menús, tarjetas y botones.

Para usarlo, se le pone `display: flex;` al **contenedor** (el elemento que envuelve a los que querés acomodar).

```
<div class="contenedor">
  <div class="caja">1</div>
  <div class="caja">2</div>
  <div class="caja">3</div>
</div>
```

```css
.contenedor {
  display: flex;                  /* activa flexbox */
  justify-content: space-between; /* distribucion horizontal */
  align-items: center;            /* alineacion vertical */
  gap: 10px;                      /* espacio entre los elementos */
}

.caja {
  background-color: #ff4354;
  color: white;
  padding: 20px;
}
```

Las propiedades clave del contenedor:

- `display: flex` → activa el modo flexible.
- `flex-direction` → la dirección: `row` (fila, por defecto) o `column` (columna).
- `justify-content` → distribución en el eje principal: `flex-start`, `center`, `flex-end`, `space-between`, `space-around`.
- `align-items` → alineación en el eje perpendicular: `flex-start`, `center`, `flex-end`, `stretch`.
- `gap` → el espacio entre los elementos.

> **Nota.** Flexbox es ideal para acomodar elementos **en una sola dirección** (una fila o una columna).

32. CSS Grid

CSS Grid (cuadrícula) es una herramienta más potente para crear diseños en **dos dimensiones a la vez**: filas **y** columnas. Es ideal para maquetar la estructura completa de una página o galerías de elementos. Se activa con `display: grid;` en el contenedor.

```html
<div class="grilla">
  <div class="item">1</div>
  <div class="item">2</div>
  <div class="item">3</div>
  <div class="item">4</div>
  <div class="item">5</div>
  <div class="item">6</div>
</div>
```

```css
.grilla {
  display: grid;                        /* activa grid */
  grid-template-columns: 1fr 1fr 1fr;  /* 3 columnas iguales */
  gap: 15px;                            /* espacio entre celdas */
}

.item {
  background-color: #ffdde6;
  padding: 30px;
  text-align: center;
}
```

- `display: grid` → activa la cuadrícula.
- `grid-template-columns` → define las columnas. `1fr 1fr 1fr` crea 3 columnas iguales (la unidad `fr` reparte el espacio en fracciones).

- `grid-template-rows` → define las filas (opcional).
- `gap` → el espacio entre las celdas.

Ejemplos de columnas:

```
grid-template-columns: 1fr 1fr;     /* 2 columnas iguales */
grid-template-columns: 200px 1fr;   /* una fija de 200px y otra flexible */
grid-template-columns: 1fr 2fr;     /* la segunda el doble de ancha */
```

33. ¿Cuándo conviene Flexbox y cuándo Grid?

Las dos sirven para acomodar elementos, pero cada una brilla en distintas situaciones:

	Flexbox	Grid
Dimensiones	Una sola (fila o columna)	Dos a la vez (filas y columnas)
Ideal para	Menús, botones, alinear en línea	Estructura de página, galerías
Forma de pensar	“Acomodá esto en una línea”	“Dividí el espacio en una grilla”

Regla práctica: si necesitás alinear cosas **en una dirección** → **Flexbox**; si necesitás un diseño **en filas y columnas** → **Grid**.

> **Nota.** no son rivales: en proyectos reales se **combinan**. Por ejemplo, Grid para la estructura general y Flexbox para los botones dentro de cada sección.

Parte 5 — Diseño adaptable (responsive)

34. Media queries

Una **media query** es una regla de CSS que permite **aplicar estilos distintos según el tamaño de la pantalla**. Es la herramienta que hace que una página sea **responsive**. La idea: “si la pantalla mide menos de tal ancho, aplicá estos estilos”.

```
/* Estilos normales (para pantallas grandes) */
.contenedor {
  display: flex;
}

/* Cuando la pantalla mida 768px o menos, aplicar esto: */
@media (max-width: 768px) {
  .contenedor {
    flex-direction: column;  /* las cajas se apilan en vertical */
  }
}
```

- `@media` → indica que empieza una media query.
- `(max-width: 768px)` → la condición: se aplica cuando el ancho es 768px o menos (tablets y celulares).

Los puntos de corte (*breakpoints*) más usados:

```
@media (max-width: 1024px) { }  /* tablets */
@media (max-width: 768px)  { }  /* tablets chicas / celulares grandes */
@media (max-width: 480px)  { }  /* celulares */
```

35. Mobile First aplicado en CSS

Como vimos en la Parte 1, **Mobile First** significa diseñar primero para el celular. Llevado al CSS, se traduce en una forma concreta de escribir las reglas:

6. Primero escribís los estilos **base** pensados para **pantalla chica** (celular).
7. Después, con media queries de `min-width`**, vas** agregando **ajustes para pantallas más grandes.**

```
/* 1. Estilos base: pensados para celular */
.contenedor {
  display: flex;
  flex-direction: column;  /* en el celu, las cajas van apiladas */
}

/* 2. A partir de 768px (tablets en adelante): */
@media (min-width: 768px) {
  .contenedor {
    flex-direction: row;   /* en pantallas grandes, van en fila */
  }
}
```

La diferencia entre los dos tipos de media query:

- `max-width` → “de este ancho para abajo” (se piensa primero en grande y se adapta hacia chico).
- `min-width` → “de este ancho para arriba” (se piensa primero en chico y se adapta hacia grande). Esta es la que usa Mobile First.

> Nota. ambas funcionan; lo importante es ser consistente. Si adoptás Mobile First, usá `min-width`.