

Ejercicios de HTML y CSS resueltos

Los 2 ejercicios de HTML y CSS de la web en una sola hoja: cada uno con su enunciado, los datos que necesitas (código de partida, ejemplos de entrada y salida o la base de datos) y, al final, la solución explicada.

Índice

Ejercicios largos

1. Landing de una cafetería: Flexbox, Grid y diseño adaptable

2. Página de un evento accesible: HTML semántico y ARIA

Ejercicios largos

1. Landing de una cafetería: Flexbox, Grid y diseño adaptable

Difícil · 90 minutos · apuntesdam.com/ejercicios/css/landing-de-una-cafeteria

Café Molino abre su web y el diseñador ya ha entregado el HTML de la página de inicio: una cabecera con el logo y el menú, una portada con el titular y el botón de reservas, y la carta con los platos. Falta todo el CSS, y el formulario de reservas lo hizo un becario sin pensar en la accesibilidad.

La mayoría de las visitas llegarán desde el móvil, así que el diseño se piensa primero para pantallas estrechas y se amplía con media queries (mobile first). Las pruebas de cada apartado comprueban la página a varios anchos, como lo haría un cliente al girar el móvil o abrirla en el ordenador.

Los apartados construyen la misma hoja de estilos poco a poco: cada uno empieza con lo hecho en el anterior y añade una parte nueva.

Requisitos

- Cada apartado se corrige por separado con pruebas que se ejecutan dentro de la página (algunas a varios anchos de pantalla). Debajo del editor tienes el resultado en directo.
- Los apartados de CSS empiezan con la solución del anterior ya escrita: así la página va creciendo. Puedes hacer los apartados en cualquier orden, pero conviene seguirlo.
- Respeta los nombres de las clases del HTML y los valores que pide cada apartado (separaciones, colores, tamaños): las pruebas miden la página.

Página de partida (html)

```
<header class="cabecera">
  <a class="logo" href="#">Café Molino</a>
  <nav class="menu" aria-label="Principal">
    <a href="#carta">Carta</a>
    <a href="#local">El local</a>
    <a href="#reservas">Reservas</a>
  </nav>
</header>
<main>
  <section class="hero">
    <h1>Café de especialidad en el centro</h1>
    <p>Tostamos nuestro propio grano cada semana.</p>
    <a class="boton" href="#reservas">Reserva tu mesa</a>
  </section>
  <section id="carta" class="carta">
    <article class="plato"><h2>Espresso</h2><p class="precio">1,80 €</p></article>
    <article class="plato"><h2>Flat white</h2><p class="precio">2,90 €</p></article>
    <article class="plato agotado"><h2>Cold brew</h2><p class="precio">3,20 €</p></article>
    <article class="plato"><h2>Tarta de queso</h2><p class="precio">4,50 €</p></article>
    <article class="plato"><h2>Croissant</h2><p class="precio">2,10 €</p></article>
    <article class="plato agotado"><h2>Bizcocho de limón</h2><p class="precio">3,00 €</p></article>
  </section>
</main>
```

Apartados

- La cabecera: con Flexbox, el logo a la izquierda y el menú a la derecha en la misma línea, centrados en vertical, con 16px de relleno arriba y abajo y 24px a los lados; los enlaces del menú en fila con 24px de separación. En pantallas de menos de 600px, el menú pasa debajo del logo.
- La carta (`.carta`): una rejilla con CSS Grid, 24px de relleno y 20px de separación entre platos; una columna en el móvil, dos columnas iguales a partir de 600px y tres a partir de 900px. Cada `.plato` con un borde de 1px, las esquinas redondeadas 8px y 16px de relleno.
- Colores y tipografía: define en `:root` las variables `--cafe: #6b3e26` y `--crema: #fff8f0`, usa `--crema` como fondo de la página y haz que el `h1` crezca con la pantalla con `clamp(1.75rem, 5vw, 3rem)`. El `.boton` es una píldora: `inline-block`, relleno de 12px arriba y abajo y 20px a los lados, esquinas de 999px, fondo `--cafe`, texto

blanco y sin subrayado.

- Los platos con la clase `agotado` se ven al 60 % de opacidad y llevan una etiqueta «Agotado» en la esquina superior derecha, hecha solo con CSS mediante `::after`: posición absoluta a 8px del borde superior y del derecho (el plato tiene que ser su referencia), fondo `#b00020` y texto blanco.
- El formulario de reservas no es accesible: no tiene etiquetas, los tipos de campo no ayudan en el móvil y el botón es un `div`. Rehaz su HTML: cada campo con su `<label for>`; nombre y correo obligatorios con `autocomplete` (`name` y `email`) y el correo de tipo `email`; personas de tipo `number` entre 1 y 8; fecha de tipo `date` y obligatoria; el turno con dos radios (`comida` y `cena`, cada uno con su `id` y su `label`) dentro de un `<fieldset>` con la `<legend>` «Turno»; y un `<button type="submit">` «Reservar». Conserva los `id` `nombre`, `email`, `personas` y `fecha`.

2. Página de un evento accesible: HTML semántico y ARIA

Medio · 60 minutos · apuntesdam.com/ejercicios/css/pagina-de-un-evento-accesible

El instituto organiza unas jornadas de software libre y alguien hizo la página a toda prisa con `div` y `span`. Se ve bien, pero una persona que navega con teclado o con un lector de pantalla no puede usarla: el menú no se puede enfocar, no hay encabezados para saltar de sección en sección, las imágenes no tienen texto alternativo y la tabla del programa es una cuadrícula sin sentido.

Las pautas WCAG 2.2 (nivel AA, el que exige la ley a las webs públicas) piden que todo funcione con teclado, que la estructura se pueda leer sin verla y que los controles anuncien su nombre y su estado. Casi todo se consigue con el elemento HTML adecuado; ARIA solo hace falta para lo que el HTML no tiene, como el estado abierto o cerrado de un menú.

Requisitos

- Cada apartado se corrige por separado con pruebas que se ejecutan dentro de la página (algunas a varios anchos de pantalla). Debajo del editor tienes el resultado en directo.
- Conserva las clases y los textos visibles de la página: las pruebas buscan los elementos por ellos.

Página de partida (html)

```
<div class="cabecera">
  
  <div class="menu">
    <span onclick="location='#programa'">Programa</span>
    <span onclick="location='#entradas'">Entradas</span>
  </div>
</div>
<div class="contenido">
  <div class="titulo">Jornadas de Software Libre 2026</div>
  
  
  <div class="subtitulo">Programa</div>
  <table>
    <tr><td>Hora</td><td>Charla</td><td>Sala</td></tr>
    <tr><td>10:00</td><td>Linux en el aula</td><td>A</td></tr>
    <tr><td>11:30</td><td>Accesibilidad web</td><td>B</td></tr>
  </table>
  <a href="programa.pdf">Pincha aquí</a>
</div>
```

Apartados

- Rehaz el HTML de la página: un enlace «Saltar al contenido» (clase `saltar`) como primer elemento, que lleve al `main` (con `id` `principal`); la cabecera en un `header`, el menú en un `nav` con `aria-label="Principal"` y sus opciones como enlaces (`#programa` y `#entradas`); el contenido en `main`; el título como `h1` y «Programa» como `h2` con `id` `programa`; el logo con el `alt` «Jornadas de Software Libre», el cartel con un `alt` que describa su información (fechas y lugar) y el separador, que es decorativo, con `alt=""`; la tabla con un `caption` y su primera fila con `th scope="col"`; y el enlace al PDF con un texto que diga qué es y que es un PDF (sin «aquí»).

- En el móvil, el menú se despliega con un botón. El HTML ya es correcto (un `button` con `aria-expanded` y `aria-controls`, y el `nav` con `hidden`); escribe el JavaScript: al pulsar el botón, el menú se muestra u oculta (atributo `hidden`) y `aria-expanded` pasa a `"true"` o `"false"`; con el menú abierto, la tecla Escape lo cierra y devuelve el foco al botón.

Soluciones

1. Landing de una cafetería: Flexbox, Grid y diseño adaptable

Difícil · 90 minutos · apuntesdam.com/ejercicios/css/landing-de-una-cafeteria

Apartado 1 (css)

```
* { box-sizing: border-box; }
body { margin: 0; font-family: system-ui, sans-serif; }

/* 1. Cabecera */
.cabecera {
  display: flex;
  flex-wrap: wrap;
  justify-content: space-between;
  align-items: center;
  gap: 8px;
  padding: 16px 24px;
}
.menu { display: flex; gap: 24px; }
@media (max-width: 599px) {
  .cabecera { flex-direction: column; align-items: flex-start; }
}
```

Apartado 2 (css)

```
* { box-sizing: border-box; }
body { margin: 0; font-family: system-ui, sans-serif; }

/* 1. Cabecera */
.cabecera {
  display: flex;
  flex-wrap: wrap;
  justify-content: space-between;
  align-items: center;
  gap: 8px;
  padding: 16px 24px;
}
.menu { display: flex; gap: 24px; }
@media (max-width: 599px) {
  .cabecera { flex-direction: column; align-items: flex-start; }
}

/* 2. Carta */
.carta { display: grid; gap: 20px; padding: 24px; }
@media (min-width: 600px) { .carta { grid-template-columns: repeat(2, 1fr); } }
@media (min-width: 900px) { .carta { grid-template-columns: repeat(3, 1fr); } }
.plato { border: 1px solid #d9cfc7; border-radius: 8px; padding: 16px; }
```

Apartado 3 (css)

```
* { box-sizing: border-box; }
body { margin: 0; font-family: system-ui, sans-serif; }

/* 1. Cabecera */
.cabecera {
  display: flex;
  flex-wrap: wrap;
  justify-content: space-between;
  align-items: center;
  gap: 8px;
  padding: 16px 24px;
}
.menu { display: flex; gap: 24px; }
@media (max-width: 599px) {
  .cabecera { flex-direction: column; align-items: flex-start; }
}

/* 2. Carta */
.carta { display: grid; gap: 20px; padding: 24px; }
@media (min-width: 600px) { .carta { grid-template-columns: repeat(2, 1fr); } }
@media (min-width: 900px) { .carta { grid-template-columns: repeat(3, 1fr); } }
.plato { border: 1px solid #d9cfc7; border-radius: 8px; padding: 16px; }

/* 3. Colores y tipografía */
:root {
  --cafe: #6b3e26;
  --crema: #fff8f0;
}
body { background: var(--crema); }
h1 { font-size: clamp(1.75rem, 5vw, 3rem); }
.boton {
  display: inline-block;
  padding: 12px 20px;
  border-radius: 999px;
  background: var(--cafe);
  color: #fff;
  text-decoration: none;
}
```

Apartado 4 (css)

```
* { box-sizing: border-box; }
body { margin: 0; font-family: system-ui, sans-serif; }

/* 1. Cabecera */
.cabecera {
  display: flex;
  flex-wrap: wrap;
  justify-content: space-between;
  align-items: center;
  gap: 8px;
  padding: 16px 24px;
}
.menu { display: flex; gap: 24px; }
@media (max-width: 599px) {
  .cabecera { flex-direction: column; align-items: flex-start; }
}

/* 2. Carta */
.carta { display: grid; gap: 20px; padding: 24px; }
@media (min-width: 600px) { .carta { grid-template-columns: repeat(2, 1fr); } }
@media (min-width: 900px) { .carta { grid-template-columns: repeat(3, 1fr); } }
.plato { border: 1px solid #d9cfc7; border-radius: 8px; padding: 16px; }

/* 3. Colores y tipografía */
:root {
  --cafe: #6b3e26;
  --crema: #fff8f0;
}
body { background: var(--crema); }
h1 { font-size: clamp(1.75rem, 5vw, 3rem); }
.boton {
  display: inline-block;
  padding: 12px 20px;
  border-radius: 999px;
  background: var(--cafe);
  color: #fff;
  text-decoration: none;
}

/* 4. Platos agotados */
.plato { position: relative; }
.plato.agotado { opacity: 0.6; }
.plato.agotado::after {
  content: "Agotado";
  position: absolute;
  top: 8px;
  right: 8px;
  padding: 2px 8px;
  border-radius: 4px;
  background: #b00020;
  color: #fff;
  font-size: 0.75rem;
}
}
```

Apartado 5 (html)

```
<section id="reservas">
  <h2>Reserva tu mesa</h2>
  <form class="reserva">
    <label for="nombre">Nombre</label>
    <input id="nombre" name="nombre" autocomplete="name" required>

    <label for="email">Correo</label>
    <input id="email" name="email" type="email" autocomplete="email" required>

    <label for="personas">Personas</label>
    <input id="personas" name="personas" type="number" min="1" max="8" value="2" required>

    <label for="fecha">Fecha</label>
    <input id="fecha" name="fecha" type="date" required>

    <fieldset>
      <legend>Turno</legend>
      <input type="radio" id="comida" name="turno" value="comida" required>
      <label for="comida">Comida</label>
      <input type="radio" id="cena" name="turno" value="cena">
      <label for="cena">Cena</label>
    </fieldset>

    <button type="submit">Reservar</button>
  </form>
</section>
```

La cabecera usa Flexbox con `justify-content: space-between` para separar logo y menú, y una media query que cambia la dirección a columna en el móvil. La carta usa Grid con `repeat(n, 1fr)`: las columnas se reparten el ancho por igual y `gap` pone la separación sin márgenes que haya que compensar.

El enfoque mobile first deja el CSS base para el móvil y añade columnas con `min-width`. Las media queries se escriben de menor a mayor para que la más ancha gane cuando se cumplen varias.

Las variables CSS centralizan los colores de la marca y `clamp()` hace que el título crezca de forma continua entre un mínimo y un máximo. El pseudoelemento `::after` añade la etiqueta sin tocar el HTML, y `position: relative` en el plato la ancla a su esquina.

El formulario accesible muestra que la accesibilidad empieza en el HTML: con `label for`, tipos de campo adecuados, `fieldset` y un `button` real, el formulario funciona con teclado, con lector de pantalla y con el teclado numérico o el calendario del móvil, sin una línea de JavaScript.

2. Página de un evento accesible: HTML semántico y ARIA

Medio · 60 minutos · apuntesdam.com/ejercicios/css/pagina-de-un-evento-accesible

Apartado 1 (html)

```
<a class="saltar" href="#principal">Saltar al contenido</a>
<header class="cabecera">
  
  <nav class="menu" aria-label="Principal">
    <a href="#programa">Programa</a>
    <a href="#entradas">Entradas</a>
  </nav>
</header>
<main id="principal" class="contenido">
  <h1 class="titulo">Jornadas de Software Libre 2026</h1>
  
  
  <h2 class="subtitulo" id="programa">Programa</h2>
  <table>
    <caption>Charlas del sábado</caption>
    <tr><th scope="col">Hora</th><th scope="col">Charla</th><th scope="col">Sala</th></tr>
    <tr><td>10:00</td><td>Linux en el aula</td><td>A</td></tr>
    <tr><td>11:30</td><td>Accesibilidad web</td><td>B</td></tr>
  </table>
  <a href="programa.pdf">Descarga el programa completo (PDF, 120 KB)</a>
</main>
```

Apartado 2 (javascript)

```
const boton = document.querySelector(".abrir-menu");
const menu = document.getElementById(boton.getAttribute("aria-controls"));

function abrir(abierto) {
  boton.setAttribute("aria-expanded", String(abierto));
  menu.hidden = !abierto;
}

boton.addEventListener("click", () => abrir(boton.getAttribute("aria-expanded") !== "true"));

// Escape cierra el menú y devuelve el foco al botón
document.addEventListener("keydown", (e) => {
  if (e.key === "Escape" && boton.getAttribute("aria-expanded") === "true") {
    abrir(false);
    boton.focus();
  }
});
```

Casi todas las barreras se arreglan cambiando `div` y `span` por el elemento que describe su función: `header`, `nav`, `main`, `h1`, `a`, `button`, `th`. Los navegadores y los lectores de pantalla ya saben qué hacer con ellos.

Los textos alternativos y los textos de los enlaces son contenido: deben tener sentido fuera de contexto, porque los lectores de pantalla permiten listar todos los enlaces de una página. «Pincha aquí» repetido diez veces no dice nada.

El menú desplegable usa ARIA solo para el estado (`aria-expanded`) y la relación (`aria-controls`); el resto lo da el `button`. Cerrar con Escape y devolver el foco al botón evita que el usuario de teclado se quede perdido.

Las pruebas de este ejercicio comprueban lo mismo que herramientas automáticas como axe o Lighthouse (estructura, alternativas, nombres de controles), pero ninguna herramienta sustituye a probar la página con teclado y con un lector de pantalla.