Manual de HTML y de CSS

1. Fundamentos de HTML

Este manual es tu punto de partida para dominar el lenguaje que da forma a la web. En esta primera sección, entenderás qué es HTML, su historia y cómo se diferencia de otras tecnologías.

1.1 Introducción a HTML

HTML, o HyperText Markup Language , es el lenguaje de marcado predominante para la creación de páginas web. A diferencia de un lenguaje de programación, HTML no ejecuta operaciones, sino que utiliza etiquetas para estructurar y describir el contenido de un documento. En este manual, exploraremos cómo estas etiquetas te permiten crear desde un simple párrafo hasta páginas web interactivas.

1.2 Un Vistazo a la Historia

La primera versión pública de HTML fue un documento llamado "HTML Tags", publicado por Tim Berners-Lee en 1991, que incluía 22 elementos. En 1996, el

W3C (World Wide Web Consortium) se convirtió en el organismo responsable de los estándares de HTML. En 2004, varias empresas fundaron

WHATWG para impulsar el desarrollo de HTML5, lo que llevó al W3C a retomar la estandarización. Este manual se centrará en HTML5, el estándar actual.

1.3 Diferencia con otras tecnologías

Es crucial entender que HTML trabaja junto con otras tecnologías web:

- CSS (Cascading Style Sheets): Se utiliza para describir la apariencia de un documento HTML, como los colores, fuentes y márgenes. HTML se encarga de la estructura, mientras que CSS se encarga del diseño, lo que facilita el mantenimiento del código.
- JavaScript: Es un lenguaje de programación que puede incluirse en una página para controlar el comportamiento del navegador y crear páginas web dinámicas.

---

2. Tu Primer Documento HTML

En este capítulo, crearás tu primer archivo HTML y aprenderás la estructura básica que todo documento debe seguir.

2.1 Herramientas necesarias

Para escribir código HTML, solo necesitas un editor de texto plano. Editores como Bloc de notas en Windows, o Gedit y Geany en Linux, son perfectos para empezar. Es importante evitar programas como Microsoft Word, ya que añaden formato oculto que puede corromper el código.

2.2 Creando la estructura básica

La estructura de todo documento HTML se divide en dos partes: la cabecera (`<head>`) y el cuerpo (`<body>`). Aquí tienes un ejemplo de la estructura mínima que necesitas

```
<!DOCTYPE html>

<html lang="es">

<head>

    <meta charset="UTF-8">

    <title>Título de la Página</title>

</head>

<body>

    </body>

</html>
```

3. ¿Qué etiquetas son imprescindibles y no pueden faltar?

3.1 ¿Qué es una etiqueta?

Las etiquetas HTML son los bloques de construcción esenciales de una página web. Cada etiqueta tiene una función específica y ayuda a organizar el contenido de manera lógica. Están formadas por una etiqueta de apertura y, opcionalmente, una de cierre.

📐 Estructura básica de una etiqueta

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

3. 2 Tipos de etiquetas.

Las etiquetas que tenemos en html las podemos organizar en tres categorías.

1. Etiquetas de Estructura 🏛️

Estas etiquetas son la base, la arquitectura de un documento HTML. No se ven directamente en la página, pero son esenciales para su funcionamiento y para que el navegador la interprete correctamente.

- **HTML:** La etiqueta raíz de todo el documento.
- **Head:** El contenedor para la metainformación de la página (título, enlaces a CSS/JS, etc.).
- **Body:** El contenedor para todo el contenido visible de la página (texto, imágenes, videos, etc.).

2. Etiquetas Semánticas 📝

Estas etiquetas le dan **significado** y **contexto** al contenido. Su propósito es describir la función del texto o del bloque de contenido que contienen, en lugar de simplemente darle formato. Usar estas etiquetas es una práctica moderna y crucial para la accesibilidad y el SEO.

- **Header:** Sección introductoria o de navegación de un documento o sección.
- **Main:** Contenedor del contenido principal y único de la página.
- **Footer:** Contiene información de pie de página (derechos de autor, enlaces, etc.).
- **Nav:** Un contenedor para los enlaces de navegación.
- **Article:** Contenido independiente y autocontenido (como un artículo de blog).
- **Section:** Un agrupador temático de contenido.

- **H1-H6:** Encabezados que establecen una jerarquía de títulos y subtítulos.
- **P:** Para agrupar párrafos de texto.
- **A:** Para crear hipervínculos.
- **Img:** Para incrustar imágenes.
- **Table:** Para crear tablas.
- **Ul/Ol & Li:** Para crear listas.

Con un enfoque didáctico y técnico, podemos clasificar las etiquetas HTML en tres categorías principales, basándonos en su función y propósito en un documento web.

3. Etiquetas de Formato y Énfasis ✍️

Aunque algunas de estas etiquetas han sido reemplazadas por CSS para el estilo, todavía se usan para dar un significado semántico a palabras o frases específicas dentro del texto.

- **Strong:** Indica un texto de gran importancia o urgencia.
- **Em:** Para enfatizar una palabra o frase.
- **I:** Se utiliza para texto que se diferencia del resto, como nombres de especies, frases de otro idioma, etc.
- **B:** Simplemente aplica un estilo de negrita al texto sin un significado semántico adicional.

3.3 ¿Todas las etiquetas hay que cerrarlas?

Sí, todas esas etiquetas **deben cerrarse**. En HTML5, la mayoría de las etiquetas tienen una etiqueta de apertura (por ejemplo, `<html>`) y una etiqueta de cierre (por ejemplo, `</html>`). El contenido y otras etiquetas se anidan entre ellas.

Aunque algunos navegadores pueden ser tolerantes y renderizar una página incluso si falta una etiqueta de cierre, es una **mala práctica**. Cerrar las etiquetas correctamente asegura que el documento sea **válido** y que los navegadores, las herramientas de accesibilidad y los motores de búsqueda lo interpreten de la manera esperada.

Excepciones: etiquetas de cierre opcionales o sin cierre

Hay algunas etiquetas que no requieren una etiqueta de cierre. Se les conoce como etiquetas de "autocierre" o "elementos vacíos", ya que no pueden contener contenido o anidar otras etiquetas. Algunos ejemplos comunes son:

- `<br>`: Salto de línea.
- `<img>`: Inserta una imagen.
- `<input>`: Campo de entrada de datos.
- `<link>`: Enlaza una hoja de estilo.
- `<meta>`: Proporciona metadatos.

3.4 Etiquetas imprescindibles que no deben faltar en tu página

Vamos a desglosar las etiquetas fundamentales que forman la estructura de un documento HTML. Para que una página web funcione, hay un conjunto de etiquetas que son absolutamente esenciales y que no pueden faltar.

Vamos a desglosar las etiquetas fundamentales que forman la estructura de un documento HTML. Para que una página web funcione, hay un conjunto de etiquetas que son absolutamente esenciales y que no pueden faltar.

Las siguientes etiquetas son la base de cualquier documento HTML. Sin ellas, el navegador no podría interpretar el contenido correctamente.

- `<!DOCTYPE html>`: Esta es la primera línea de todo documento HTML. Aunque no es una etiqueta en sí, es una declaración que le dice al navegador que el documento es de tipo HTML5. Es crucial para que la página se renderice de manera estándar.
- `<html>`: Esta es la etiqueta raíz de la página. Todo el contenido, excepto la declaración <! DOCTYPE html >, debe estar anidado dentro de esta etiqueta .La etiqueta `<html>` es **la última en cerrarse** en un documento HTML.

  La estructura de un documento HTML sigue una jerarquía muy estricta, con etiquetas anidadas como si fueran cajas dentro de otras cajas. El `<html>` es la caja más grande, que contiene todas las demás, por lo que debe cerrarse al final del documento, después de que todo lo demás, incluyendo el `<body>`, ya ha sido cerrado.

HTML

```html
<!DOCTYPE html>
<html>
    <head>
        </head>
    <body>
        </body>
</html>
```

El orden de apertura y cierre de las etiquetas no es aleatorio. Es lo que permite al navegador entender la **estructura anidada** de tu página.

- **Anidamiento correcto**: Cuando cierras una etiqueta, el navegador sabe que esa sección de contenido ha terminado. Si el orden es incorrecto, el navegador podría interpretar el código de forma inesperada, lo que podría provocar problemas de visualización o errores en la página.
- **Validez del código**: Un código que sigue las reglas de anidamiento es un código "válido" o "bien formado". Aunque los navegadores modernos son muy tolerantes y a menudo corrigen errores automáticamente, seguir las reglas asegura que tu código funcione de manera consistente en todos los navegadores y dispositivos, y es una práctica fundamental en la programación.

- `<head>`: Este es el encabezado de la página. Contiene información meta que no se muestra en el cuerpo visible de la página, pero que es fundamental para su funcionamiento. Aquí se incluyen cosas como el título de la página, los metadatos, los enlaces a hojas de estilo (CSS) y los scripts (JavaScript).
- `<title>`: Se encuentra dentro de `<head>` y define el título de la página, que es el texto que aparece en la pestaña del navegador.
- `<body>`: Este es el cuerpo del documento. Contiene todo el contenido visible de la página, como texto, imágenes, enlaces, videos, etc.

Recuerda ¿Cuál es la primera etiqueta que tiene un archivo html? ¿y la siguiente? ¿cuando cerramos la etiqueta <html>?

Etiquetas que no se pueden repetir

Dentro de la estructura básica de un documento HTML, hay varias etiquetas que deben aparecer una sola vez y siempre deben aparecer abiertas y cerradas.

- `<html>`: Como la etiqueta raíz, sólo puede haber una.
- `<head>`: Solo puede haber un encabezado por documento.
- `<body>`: Solo puede haber un cuerpo por documento.
- `<title>`: Solo se puede definir un título para la página.

Aunque otras etiquetas, como `<h1>`, `<p>`, `<img>` o `<a>` pueden repetirse tantas veces como sea necesario para construir el contenido de la página, estas etiquetas estructurales deben seguir la regla de una sola instancia para mantener la coherencia y la validez del documento.

Entender esta jerarquía es el primer paso para construir una página web funcional y bien estructurada. La relación entre `<html>`, `<head>` y `<body>` es la base de todo.

El `<head>` le dice al navegador qué es la página, mientras que el `<body>` le dice al navegador qué mostrar en la página.
Nuestro contenido estará en el body. El `<body>` es el espacio donde se construye el contenido de la web, y hay varias etiquetas que son esenciales para estructurar y dar formato a ese contenido.

3. ¿Qué va dentro del head?

```
<head>
    <meta charset="UTF-8"> <!-- Define la codificación de caracteres como UTF-8 -->
    <meta name="viewport" content="width=device-width, initial-scale=1.0"> <!-- Hace que la página
    <title>Título de tu Página Web</title> <!-- Título que aparece en la pestaña del navegador -->
    <link rel="stylesheet" href="styles.css"> <!-- Enlaza la hoja de estilos CSS -->
    <link rel="icon" href="favicon.ico" type="image/x-icon"> <!-- Icono de la pestaña del navegador
    <script src="script.js"></script> <!-- Enlaza un archivo JavaScript externo -->
</head>
```

Etiquetas del `<head>` y su función

- **`<meta charset="UTF-8">`**: Esta etiqueta es esencial. Le dice al navegador qué conjunto de caracteres debe usar para mostrar el texto,

asegurando que se visualicen correctamente todos los caracteres especiales (como acentos o la letra `ñ`).

- **`<meta name="viewport" content="width=device-width, initial-scale=1.0">`**: Fundamental para el diseño responsivo. Controla cómo el navegador ajusta la página en dispositivos móviles, asegurando que el ancho de la página se adapte al ancho de la pantalla del dispositivo.
- **`<title>Document</title>`**: Define el título que aparece en la pestaña del navegador. Es único por página. Define el título de la página. Este es el texto que aparece en la **pestaña del navegador** y en los resultados de los motores de búsqueda (como Google).
- **`<link rel="stylesheet" href="styles.css">`**: Vincula un archivo de hojas de estilo (CSS) externo a tu documento HTML. Esto separa el diseño del contenido, lo que es una práctica recomendada.
- **`<link rel="icon" href="favicon.ico" type="image/x-icon">`**: Define el ícono de la pestaña del navegador (el "favicon"). Es una etiqueta opcional pero común para mejorar la identidad visual del sitio. Es el logo de tu marca en la pestaña, lo que ayuda a los usuarios a reconocer tu sitio web al instante cuando tienen varias pestañas abiertas.

¿Tiene que tener el favicon siempre el formato .ico? No, el favicon ya no tiene que ser siempre un archivo con extensión `.ico`. Aunque el formato `.ico` fue el estándar durante muchos años y sigue siendo ampliamente compatible, los navegadores modernos admiten otros tipos de archivos de imagen para los favicons.
`.png`: Es el formato más popular y recomendado actualmente. Permite transparencia, lo que es ideal para íconos con formas irregulares, y tiene una excelente calidad de compresión
`.svg`: Este es un formato de gráficos vectoriales. Sus mayores ventajas son que los archivos son muy ligeros y que la imagen se ve perfectamente nítida en cualquier tamaño y resolución, sin pixelarse.

¿qué significan esas palabras que van después de rel?

El atributo `rel` (de relationship, relación en inglés) se usa en las etiquetas `<link>` para definir la relación entre el documento HTML y el recurso externo al que se está enlazando. Los valores que se pueden usar en este atributo no son arbitrarios, sino que son palabras clave predefinidas que le dicen al navegador qué tipo de relación existe.

Valores de `rel` más comunes

Valor de `rel`	Significado	Uso común
`stylesheet`	El documento enlazado es una hoja de estilos.	`<link rel="stylesheet" href="style.css">`
`icon`	El documento enlazado es un ícono de página.	`<link rel="icon" href="favicon.ico">`
`preconnect`	El navegador debe precargar la conexión al origen.	`<link rel="preconnect" href="...">`
`alternate`	El documento es una versión alternativa.	`<link rel="alternate" href="...">`
`preload`	El recurso debe ser precargado.	`<link rel="preload" href="...">`

4. ¿Qué va dentro del body?

El diseño de un documento HTML moderno se basa en la **semántica**, es decir, en el significado que cada etiqueta le confiere al contenido que contiene. Abandonamos la práctica de utilizar etiquetas genéricas como `<div>` para todo y, en su lugar, empleamos etiquetas que describen la función del contenido. Esto es vital para la **accesibilidad** y la **optimización para motores de búsqueda (SEO)**.

Los encabezados (`h1` a `h6`) no son simplemente para dar formato a los títulos; establecen una jerarquía de contenido. El `<h1>` debe ser el título principal de la página, y los subsecuentes (`<h2>`, `<h3>`, etc.) deben seguir una estructura lógica y anidada. Saltarse niveles (por ejemplo, pasar de un `<h1>` a un `<h3>`) rompe esta jerarquía y puede confundir tanto a los lectores de pantalla como a los rastreadores de motores de búsqueda.

- **Aplicación**: Un `<h1>` para el título del artículo, un `<h2>` para las secciones principales y un `<h3>` para los subtemas dentro de esas secciones.
- **Importancia Técnica**: Un `h1` bien definido es el elemento más importante para el SEO de una página. Los motores de búsqueda lo utilizan para comprender de qué trata el contenido principal.

`<p>`: Párrafos de Texto

La etiqueta `<p>` se utiliza para agrupar bloques de texto. Su propósito es definir un párrafo, no solo una línea de texto. Es la etiqueta más fundamental para la presentación de texto en un documento web.

- **Aplicación**: Cada bloque de texto coherente debe ir envuelto en su propia etiqueta `<p>`. Evite usar `<p>` para simples saltos de línea (para eso está `<br>`).

`<a>`: Anclas (Hipervínculos)

La etiqueta `<a>` es el pilar de la navegación en la web. Permite crear **hipervínculos** que conectan un documento con otros recursos. El atributo **`href`** (Hypertext Reference) es obligatorio y especifica la URL de destino.

- **Aplicación**: `<a>` puede enlazar a otra página (`href="contacto.html"`), a una sección dentro de la misma página (`href="#seccion-x"`) o a un recurso externo. También se usa para enlaces de correo electrónico (`href="mailto:nombre@email.com"`) o de teléfono (`href="tel:123456789"`).

`<img>`: Imágenes

La etiqueta `<img>` es un elemento vacío, lo que significa que no tiene una etiqueta de cierre. Su función es incrustar una imagen en el documento. Requiere dos atributos esenciales:

- **`src`** (Source): La ruta del archivo de imagen.
- **`alt`** (Alternative Text): Texto descriptivo que se muestra si la imagen no se puede cargar o es leído por tecnologías de asistencia. Este atributo es **crítico para la accesibilidad y el SEO**.
- **Aplicación**: `<img src="ruta/imagen.jpg" alt="Descripción de la imagen para usuarios con discapacidad visual y para SEO">`.

¿Qué etiquetas tenemos para organizar el contenido de nuestro body?

`<main>`: El Contenido Principal

La etiqueta **`<main>`** representa el contenido principal y único de un documento. En una página web, solo debe haber un `<main>`. Su propósito es delimitar el contenido central de la página, diferenciándolo de otros elementos que se repiten en el sitio, como la barra de navegación o el pie de página.

- **¿Qué va dentro?** El contenido que es exclusivo de esa página. Por ejemplo, en un blog, el `<main>` contendría el artículo completo. En una tienda en línea, contendría la lista de productos o los detalles de un producto específico.

- **¿Qué no va dentro del main?** Elementos que se repiten en otras páginas, como el `<header>`, `<nav>`, `<footer>` o barras laterales de anuncios (`<aside>`).

<p>, <section>, <article>, y : las etiquetas de contenido. Parecidas pero diferentes.

Estas etiquetas son fundamentales para estructurar el contenido dentro del `<main>` y otras secciones del `<body>`.

- **`<p>` (Párrafo):** Es la etiqueta más básica para texto. Se utiliza para agrupar un bloque de texto en un párrafo. Su uso es crucial para la legibilidad.
- **`<section>`:** Se usa para agrupar contenido temáticamente relacionado. Piensa en una `<section>` como un capítulo en un libro. Cada `<section>` suele tener un encabezado (h1-h6) que describe su contenido.
- **`<article>`:** Representa un contenido independiente y autónomo. Un `<article>` podría ser un post de un blog, un artículo de noticias o un comentario en un foro. Si pudieras tomar ese contenido y publicarlo de forma independiente, es un buen candidato para un `<article>`.
- **`<div>` (División):** Esta es la etiqueta genérica. No tiene significado semántico y se utiliza para agrupar elementos cuando no hay otra etiqueta más adecuada. A menudo se usa como un contenedor para aplicar estilos CSS o para manipular un grupo de elementos con JavaScript.

`<ul>`, `<ol>` y `<li>`: Listas

Estas etiquetas son fundamentales para estructurar información en formato de lista, mejorando la legibilidad.

- **`<ul>`** (Unordered List): Para listas de elementos sin un orden particular.
- **`<ol>`** (Ordered List): Para listas en las que el orden de los elementos es importante (por ejemplo, una receta, un ranking o una serie de pasos).
- **`<li>`** (List Item): Cada elemento individual de una lista debe ser una etiqueta `<li>`.
- Etiqueta de enlace (`<a>`): Se utiliza para crear hipervínculos que enlazan con otras páginas web, archivos o secciones dentro del mismo documento. El atributo `href` es el más importante, ya que especifica la URL de destino.
- Etiqueta de imagen (`<img>`): Se usa para incrustar imágenes en la página. Requiere el atributo `src` para especificar la ruta de la imagen y el atributo `alt` para proporcionar un texto alternativo que se muestra si la imagen no se carga y es crucial para la accesibilidad.

Estas etiquetas, entre otras, son los bloques de construcción que permiten organizar y presentar la información de manera legible y accesible para los usuarios.

Ancla vs Enlace.

Enlace (link)

Es cualquier **elemento <a> que apunta a otro recurso** (una página web, un archivo, una sección de la misma página, un correo…).

Ir a Google

Ancla (anchor)

Un **ancla** es un **punto de referencia dentro de la misma página** al que podemos "saltar" con un enlace.Se define con un **id** en un elemento, y el enlace apunta a ese id con #.

<!-- Enlace -->

Ir a la sección 2

¿Para qué mas nos valdrá ese id?

<!-- Contenido de destino -->

<h2 id="seccion2">Sección 2</h2>

Este es el texto de la sección 2.

La relación de Padre e Hijo

En HTML, las etiquetas se anidan unas dentro de otras, creando una jerarquía. La etiqueta que contiene a otra se llama **padre**, y la que está contenida se llama **hijo**.

```
</header>

<main> <section> <h2>Nuestra Empresa</h2> <p>Este es un párrafo sobre nuestra compañía.</p> </section>
/main>
/body>
/html>
```

En este ejemplo, `<main>` es el padre de `<section>`, y a su vez, `<section>` es el padre de `<h2>` y `<p>`. Esta relación de padre e hijo es la base de toda la estructura de un documento HTML y es fundamental para aplicar estilos y manipular elementos con JavaScript.

Que una etiqueta sea hija de otra es fundamental para la **estructura semántica** y la **funcionalidad** de un documento HTML. Esta relación de "padre e hijo" crea una jerarquía lógica que es vital para la organización del contenido, el diseño y la interactividad de la página.

La relación de padre e hijo es la base de cómo CSS y JavaScript manipulan el diseño y el comportamiento de la página:

CSS: Puedes aplicar estilos a un grupo de elementos de manera eficiente. Por ejemplo, al aplicar un estilo al padre (`<section>`), todos sus hijos (`<h2>`, `<p>`) pueden heredar ese estilo, o puedes seleccionar un hijo específico (`<section> p`) sin afectar a otros párrafos de la página. Esto hace que tus hojas de estilo sean más limpias y fáciles de mantener.

JavaScript: La manipulación del DOM se basa en esta jerarquía. Puedes seleccionar un elemento padre y luego buscar o modificar a sus hijos. Esto te permite, por ejemplo, cambiar el color de todos los elementos de una lista (`<ul>`) con una sola línea de código, o hacer que un botón (`<button>`) oculte o muestre un párrafo (`<p>`) que es su hermano en el código.

En la estructura de un documento HTML, un **hermano** es una etiqueta que comparte el mismo padre que otra etiqueta. En otras palabras, son elementos que están anidados en el mismo nivel y dentro del mismo contenedor directo.

Esta relación de hermandad es muy importante para la organización del contenido y para poder manipular los elementos con CSS y JavaScript.

La relación de hermandad es clave en programación web por varias razones:

- **Diseño con CSS:** Puedes aplicar estilos a elementos basándote en su relación con sus hermanos. Por ejemplo, en CSS, el selector de hermanos adyacentes (`+`) te permite seleccionar un elemento que viene inmediatamente después de otro.
- **Manipulación con JavaScript:** Puedes seleccionar un elemento y luego moverte a través del Árbol DOM para acceder a sus hermanos. Esto te permite modificar un elemento en la página sin tener que saber su posición exacta en el código.

Veamos esta relación en nuestro ejemplo

```
<body>
    <header>
        <img src="../logo.png" alt="Logo de mi sitio" />
        <h1>Una bonita web</h1>
        <nav>
            <ul>
                <li><a href="../index.html">Inicio</a></li>
                <li><a href="etiquetas.html">Etiquetas</a></li>
                <li><a href="enlaces.html">Enlaces</a></li>
            </ul>
        </nav>
    </header>
    <main>
```

. Padres e Hijos

- La etiqueta `<header>` es el **padre** de `<img>`, `<h1>` y `<nav>`.
- La etiqueta `<nav>` es el **padre** de `<ul>`.
- La etiqueta `<ul>` es el **padre** de los tres `<li>` (elementos de la lista).
- Cada `<li>` es el **padre** de su respectiva etiqueta `<a>`.

2. Hermanos

- `<img>`, `<h1>` y `<nav>` son **hermanos** porque todos son hijos directos del mismo padre, `<header>`.
- Los tres elementos `<li>` son **hermanos** entre sí, ya que todos comparten el mismo padre, `<ul>`.
- Las etiquetas `<a>` también son hermanas entre sí, porque tienen el mismo padre, el `<li>` en cada caso.

Este ejemplo ilustra cómo una estructura simple como un encabezado de página está organizada en una jerarquía clara y lógica. Esta anidación es lo que le permite al navegador entender la relación entre los elementos y aplicar correctamente los estilos y las funcionalidades.

LA ACTIVIDAD

Hoy vamos a hacer una pequeña página web. Debes replicar esta web que yo te doy. Haz dos archivos: uno para el contenido y otro para el estilo. Crea una carpeta

ponle el nombre que quieras y crea dentro dos archivos ¿te acuerdas que extensiones deben llevar estos archivos?

Actividad Práctica: "Parques de Sevilla"

Objetivo: Crear una página web de una sola sección puedes elegir el tema que quieras pero la apariencia debe ser parecida a esta. Deberán utilizar las etiquetas de estructura, semánticas y de contenido vistas.

Instrucciones para la Actividad

Paso 0. piensa que vas a crear. ¿Cuántos contenedores hay? ¿qué otros contenedores hay dentro de cada contenedor padre?

1. Configuración de Archivos

Crea una carpeta de proyecto. Dentro de ella, crea dos archivos:

- index El archivo principal para la estructura de la página.
- styles o estilo El archivo para aplicar los estilos de diseño.

1. Estructura Básica del Documento

- <!DOCTYPE html> y <html lang="es">: Estas son las primeras líneas y las más importantes. <!DOCTYPE html> declara el tipo de documento, asegurando que el navegador lo interprete como HTML5. La etiqueta <html> es el elemento raíz, el contenedor de todo el contenido de la página. El atributo lang="es" es crucial para la accesibilidad y el SEO, indicando que el idioma principal del documento es el español.
- <head>: Este es el encabezado de la página. Su contenido no es visible para el usuario, pero es vital para el funcionamiento del sitio.
    - <meta charset="UTF-8">: Declara el conjunto de caracteres, asegurando que todos los símbolos y letras (como la ñ y los acentos) se muestren correctamente.
    - <meta name="viewport" ...>: Esta metaetiqueta es fundamental para el diseño responsivo. Le dice al navegador que el ancho de la página debe adaptarse al ancho del dispositivo, optimizando la visualización en teléfonos y tablets.

    - <title>Parques de Sevilla</title>: Define el título de la página, que aparece en la pestaña del navegador. Es una de las pocas etiquetas que sólo puede aparecer una vez dentro del <head>.
    - <link rel="stylesheet" href="../styles.css">: Esta etiqueta enlaza el documento HTML con la hoja de estilos CSS. rel="stylesheet" es un valor reservado que le indica al navegador que el archivo es una hoja de estilos, y href especifica la ruta del archivo.

2. Contenido Visible (<body>)

El <body> contiene todo el contenido visible de la página, organizado de forma semántica.

- <header>: Contiene el encabezado de la página. Es un buen ejemplo de una etiqueta semántica que agrupa elementos introductorios.
    - <h1> y <p>: El <h1> establece el título principal de la página, crucial para el SEO. El <p> le da un breve resumen. Ambos son hermanos dentro del <header>.
    - <nav>: Otra etiqueta semántica para la navegación. Contiene una lista <ul> de enlaces (<a>), que son los hijos directos de <li>.
    - , <li> y <a>: Un excelente ejemplo de jerarquía y anidamiento. El <ul> es el contenedor de la lista, los <li> son sus hijos (y hermanos entre sí) y los <a> (enlaces) son los hijos de los <li>.
- <main>: Es el contenedor principal de la página, donde se encuentra el contenido único y central.
- <section id="sobre-los-parques">: Esta etiqueta agrupa el contenido relacionado temáticamente. La id="sobre-los-parques" la hace única en la página, permitiendo que se le apliquen estilos específicos en CSS. Dentro, encontramos una jerarquía lógica de títulos (<h2>), párrafos (<p>) e imágenes (<img>).
- <img>: Esta etiqueta es de autocierre y no requiere una etiqueta final. El atributo src especifica la ruta de la imagen, mientras que el atributo alt es crucial para la accesibilidad, ya que describe la imagen para los lectores de pantalla y se muestra si la imagen no se carga.
- con class="lista-parques": Se utiliza para una lista de itinerarios. El uso de una clase es perfecto para aplicar el mismo estilo a todos los elementos <li> y hacerlos lucir como "tarjetas".
- <footer>: Contiene la información de cierre de la página. En este caso, un párrafo con un enlace <a> a un correo electrónico. Esto demuestra que los enlaces pueden usarse para más que solo navegar entre páginas.

¿Recuerdas los atajos de Emmet?

Estructura básica de un documento HTML: Escribe un signo de exclamación (`!`) y presiona `Tab`.

Crear etiquetas anidadas (hijo): Usa el signo `>`. **Escribe:** `div>ul>li`

```
HTML
```

```
<div>
  <ul>
    <li></li>
  </ul>
</div>
```

Crear etiquetas hermanas (mismo nivel): Usa el signo `+`. **Escribe:** `h1+p+h2`

```
HTML
```

```
<h1></h1>
<p></p>
<h2></h2>
```

Crear etiquetas con clases: Usa el signo ..

- Escribe: p.texto-intro
- Obtienes:

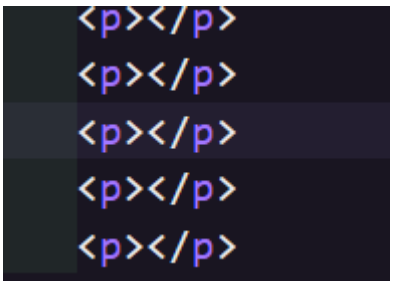
Crear etiquetas con IDs: Usa el signo #.

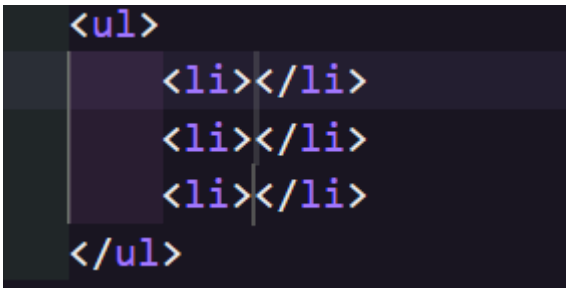
- Escribe: div#contenedor-principal
- Obtienes:

Crear múltiples elementos: Usa el asterisco *.

Escribe: li*5 por ejemplo obtienes cinco elementos dentro de una lista

Escribe p*5 obtienes 5 párrafos.





¿Un poco de css?

El código está organizado por secciones para que sea más fácil de entender. Cada sección se enfoca en una parte de tu HTML.

- `body`: Son los estilos **generales** de toda la página. El color de fondo, el tipo de letra y el espaciado se aplican a todo el contenido.
- `header`: Estilos para la cabecera. Le dan un color de fondo azul y centran el texto.
- `nav ul` y `nav ul li a`: Estos estilos transforman tu lista de navegación en una barra horizontal de enlaces, quitando las viñetas y añadiendo colores.
- `#sobre-los-parques`: Este es un estilo **específico** para la sección del `id`. Le da un fondo blanco, un borde y la centra en la página.
- `.lista-parques li`: Este estilo de **clase** es el que convierte cada elemento de la lista (`<li>`) en una tarjeta individual, con fondo y borde. No te preocupes vamos a verlo.

Lo primero empezar en un folio blanco.

Lo primero y fundamental es anular el estilo que traen los navegadores. Anotar `* { margin: 0; padding: 0; box-sizing: border-box; }` al comienzo de tu archivo CSS es una práctica estándar y fundamental en el desarrollo web. Es un paso crucial para asegurar que tus diseños se vean de la misma manera en todos los navegadores y para simplificar la forma en que trabajas con los elementos.

Las propiedades margin: 0;, padding: 0; y box-sizing: border-box; resuelven los problemas más comunes y frustrantes para los desarrolladores.

¿Qué hace esta regla?

Esta anotación usa el **selector universal** `*`, que aplica la regla a **todos los elementos** de tu página HTML. En esencia, está haciendo dos cosas muy importantes:

1. **Elimina los estilos predeterminados del navegador:** Por defecto, los navegadores (como Chrome, Firefox o Safari) aplican un poco de `margin` y `padding` a ciertos elementos (`<h1>`, `<p>`, `<ul>`, etc.). Esto puede causar inconsistencias y dificultad para controlar el diseño. Con `margin: 0;` y `padding: 0;`, te aseguras de que todos los elementos comiencen desde cero, dándote un lienzo limpio. Esto es lo que se conoce como un **"reset" de CSS**.
2. **Cambia el modelo de caja:** La propiedad `box-sizing: border-box;` es una de las más útiles en CSS. Por defecto, cuando le das un ancho (`width`) a un elemento, el `padding` y el `border` se añaden a ese ancho, haciendo que el elemento sea más grande de lo que esperabas. Con `border-box`, el `padding` y el `border` se incluyen **dentro** del ancho total que has definido. Esto hace que sea mucho más fácil y predecible trabajar con el diseño, ya que lo que ves es lo que obtienes.

¿Por qué es importante?

- **Consistencia:** Asegura que tu diseño se vea igual en cualquier navegador, sin sorpresas causadas por estilos predeterminados.
- **Facilidad de diseño:** Simplifica el cálculo de anchos y espacios. Si defines un `div` con `width: 200px`, sabes que su ancho total será exactamente 200px, incluso si le añades `padding` y `border`.
- **Base sólida:** Es la base para cualquier proyecto de desarrollo web. Poner esta regla al principio te ahorrará muchos dolores de cabeza y tiempo de depuración.

En resumen, esta simple línea de código es la primera regla que debes escribir en cualquier proyecto CSS. Es un ritual que te prepara para un desarrollo más limpio, predecible y consistente. Y te hace el trabajo mucho más fácil.

¿Qué selectores tenemos en nuestro html?

La forma en que CSS aplica estilos es a través de **reglas**. Cada regla consta de dos partes:

1. **El Selector:** Esto es lo más importante. Es la parte de la regla que le dice al navegador **a qué elemento(s) de HTML** debe aplicarse el estilo.
2. **La Declaración:** Es la instrucción de estilo en sí misma, compuesta por una **propiedad** (por ejemplo, `color`) y un **valor** (por ejemplo, `blue`).

La sintaxis de una regla es: `selector { propiedad: valor; }`.

Dominar los selectores es la clave para dominar CSS. Son las "direcciones" que usamos para comunicarnos con los elementos HTML. Existen diferentes tipos de selectores, cada uno con un nivel de especificidad o "prioridad"

¿Qué tipo de selector es body? body es un selector de etiqueta; es una etiqueta que ya está establecida

1. Selectores de Tipo (o de Etiqueta) Son los más sencillos y menos específicos. Seleccionan todos los elementos de un tipo de etiqueta.

- Sintaxis: El nombre de la etiqueta (`p`, `h2`, `a`).
- Ejemplo: `p { font-size: 16px; }`
    - Significado: Aplica un tamaño de fuente de 16px a todos los párrafos de la página.

2. Selectores de Clase (Class Selectors)

Estos son los más versátiles y comunes. Seleccionan todos los elementos que tienen un atributo `class` con un valor específico. A diferencia de los ID, las clases se pueden reutilizar en múltiples elementos.

- Sintaxis: Un punto (`.`) seguido del nombre de la clase.
- Ejemplo: `.resaltado { color: red; }`
    - Significado: Aplica un color de texto rojo a cualquier elemento que tenga la clase `resaltado`.
    - En HTML: `<p class="resaltado">Este texto es importante.</p>`

3. Selectores de ID (ID Selectors)

Son los más específicos. Un `id` debe ser **único** en todo el documento HTML. Se usan para seleccionar un solo elemento.

- **Sintaxis:** Un signo de almohadilla (`#`) seguido del nombre del ID.
- **Ejemplo:** `#menu-principal { background-color: black; }`
    - **Significado:** Aplica un color de fondo negro a **un único elemento** que tenga el ID `menu-principal`.
    - **En HTML:** `<nav id="menu-principal">...</nav>`

¿Cómo sabe el navegador que css es el que le afecta por ejemplo a un párrafo que hemos diseñado un estilo para la etiqueta párrafo está párrafo esta dentro de una sección que tiene una clase y el párrafo como tal tiene un id?

La Especificidad y la Cascada: El Desempate en CSS

Cuando un elemento tiene más de una regla de estilo que compite por él (por ejemplo, una regla para el `p` y otra para una clase en ese mismo párrafo), el navegador no se confunde. Usa un sistema de puntuación para decidir qué regla gana. La jerarquía de selectores que vimos es su "tabla de puntuación".

Puntuación: Los selectores de ID (`#`) tienen la puntuación más alta. Las clases (`.`) tienen una puntuación intermedia. Los selectores de etiqueta (`p`) tienen la puntuación más baja.

La Cascada: Si dos reglas tienen la misma puntuación de especificidad, el navegador se rige por el **orden de aparición** en el código: la última regla declarada es la que gana y se aplica.

Hemos dicho que hay tres formas de declarar los estilos de nuestro documento ¿ y si mezclamos una hoja de estilo externa con estilo en línea y estilo dentro del documento? La regla es lo mas especifico gana.

Nivel 1: Estilos en Línea (Inline) 🥇

Estos estilos se aplican directamente en la etiqueta HTML usando el atributo `style`. Son los más específicos y siempre sobrescriben cualquier otra regla, sin importar dónde se defina.

- **Sintaxis:** `<p style="color: red;">Texto.</p>`
- **Puntuación:** Muy alta (imposible de igualar con otros selectores).
- **Uso:** Generalmente se evita, ya que rompe la separación de HTML y CSS.

Nivel 2: ID (#) 🥈

El selector de `id` apunta a un elemento único en la página. Tiene una especificidad muy alta.

- **Sintaxis:** `#mi-elemento { color: blue; }`
- **Puntuación:** Media.
- **Uso:** Para elementos únicos y específicos, como un contenedor principal o un formulario.

Nivel 3: Clases (.) 🥉

Estos selectores tienen la misma especificidad y son muy comunes.

- **Clases:** Apuntan a elementos que tienen un nombre de clase específico.
- **Sintaxis:** `.mi-clase { font-weight: bold; }`
- **Puntuación:** Baja.
- **Uso:** Para componentes reutilizables y estados dinámicos.

Nivel 4: Etiquetas 🏅

Estos selectores son los menos específicos y tienen la prioridad más baja.

- **Etiquetas:** Seleccionan todos los elementos de un tipo (ej. `p`, `h1`).
- **Sintaxis:** `p { font-size: 16px; }`
- **Puntuación:** Muy baja.
- **Uso:** Para estilos genéricos en toda la página.

¿Y si aun así hay dos reglas con la misma puntuación? Entonces la última gana

Tres conceptos imprescindibles para empezar la maquetación.

`padding`: El Espacio Interno

La propiedad `padding` crea un espacio de relleno **dentro** de un elemento, entre su contenido y su borde. Piensa en el `padding` como el relleno de una caja de regalo: el contenido (`h1`, `p`, `img`, etc.) está en el centro, y el `padding` es el espacio entre ese contenido y las paredes de la caja.

- `padding` puede ser aplicado a los cuatro lados (arriba, derecha, abajo, izquierda) con un solo valor (`padding: 20px;`) o con valores específicos para cada lado (`padding: 10px 20px 30px 40px;`).

Comprender la diferencia entre `margin` (espacio externo que empuja a otros elementos) y `padding` (espacio interno que empuja el contenido del elemento) es crucial. Juntos, `display`, `padding` y `margin` son el trío fundamental para la maquetación en CSS.

`margin`: El Espacio de Separación

El `margin` crea un área de separación entre tu caja y las cajas vecinas.

- **Sintaxis de `margin`**: Puedes aplicar `margin` a los cuatro lados de un elemento de la misma forma que con `padding`.
    - **`margin: 20px;`**: Aplica un margen de 20 píxeles en los cuatro lados.
    - **`margin: 10px 20px;`**: Aplica 10px de margen arriba y abajo, y 20px a la izquierda y derecha.
    - **`margin-top`** / **`margin-right`** / **`margin-bottom`** / **`margin-left`**: Para controlar un solo lado.

Un uso muy común de `margin` es centrar un bloque horizontalmente con `margin: 0 auto;`. Si el elemento tiene un ancho definido, `auto` le dice al navegador que calcule automáticamente los márgenes laterales para que se posicione en el centro.

`display`: El Comportamiento de los Elementos

La propiedad `display` es una de las más importantes en CSS porque define cómo se muestra un elemento en la página. Es como decirle a un elemento si debe comportarse como un bloque (`block`), como una línea de texto (`inline`) o como un contenedor flexible (`flex`).

- **`display: block;`**: Un elemento de tipo bloque ocupa todo el ancho disponible y crea una nueva línea. Ejemplos son `<h1>`, `<p>`, `<div>` y `<section>`. Imagina que son como cajas apiladas una encima de la otra.
- **`display: inline;`**: Un elemento en línea sólo ocupa el espacio que necesita y no crea una nueva línea. Ejemplos son `<a>` y `<span>`. Son como palabras en una frase, se colocan una al lado de la otra.
- **`display: flex;`**: Esta propiedad convierte un elemento en un contenedor flexible, permitiendo que sus hijos se distribuyan y se alineen de forma eficiente en una fila o columna. Es la herramienta moderna para crear diseños complejos como barras de navegación o galerías de imágenes.

Y ya nos tenemos que meter con las dos grandes aportaciones de CSS3. **Flexbox** y **CSS Grid** son, sin duda, dos de las mayores y más importantes aportaciones de CSS3 al desarrollo web. Ambas revolucionaron la forma en que los desarrolladores maquetan páginas, haciendo que el diseño sea mucho más eficiente, predecible y potente que con métodos anteriores.

Antes de empezar: ¿qué problema resuelven?

Imagina que tienes cajas (divs) y quieres colocarlas en una página:

- Una al lado de otra.
- Centrar algo vertical y horizontalmente.
- Crear una galería tipo tabla pero más flexible.

Antes de **Flexbox** y **Grid**, esto era un dolor de cabeza: había que usar tablas, `float`, o trucos poco prácticos.
👉 Hoy tenemos **dos sistemas modernos de maquetación**: **Flexbox** y **Grid**.

Flexbox: El Diseño Unidimensional

Antes de Flexbox, alinear elementos en una sola dirección (ya sea en una fila o en una columna) era bastante complicado y no quedaba bien mucho menos si el tamaño de la pantalla cambiaba. Había que recurrir a trucos complejos con `float` o `inline-block`, lo que a menudo llevaba a problemas de alineación y espaciado.

Flexbox introdujo un modelo de diseño que te permite distribuir el espacio y alinear los ítems de un contenedor. Es perfecto para componentes pequeños o medianos

como barras de navegación, formularios o tarjetas de producto. Flexbox te da un control total sobre cómo los elementos se comportan cuando el espacio del contenedor cambia.

Piensa en una fila de personas cogidas de la mano.

- Flexbox está diseñado para **organizar elementos en una sola dirección**:
    - Horizontal (fila)
    - Vertical (columna)

Es como una fila de fichas de dominó: puedes alinearlas a la izquierda, derecha, centro, o distribuirlas con espacios.

Piensa en la web que estamos haciendo ¿qué elemento aparecen en una fila como si estuvieran cogidos de la mano? ¿qué display crees que tienen?

```css
nav ul {
  /* Quita los estilos de lista y espacios */
  list-style: none;
  padding: 0;
  margin: 0;
  /* Pone los elementos en una fila */
  display: flex;
}
```

y preguntamos porque al poner el selector usamos nav ul y no sólo nav? ¿Has probado a poner solo nav? ¿y por que no sólo ul si realmente el navegador es una lista?
Usamos nav ul en CSS porque es un **selector descendiente**. Esto significa que solo estamos seleccionando la etiqueta `<ul>` que es hija de una etiqueta `<nav>`.

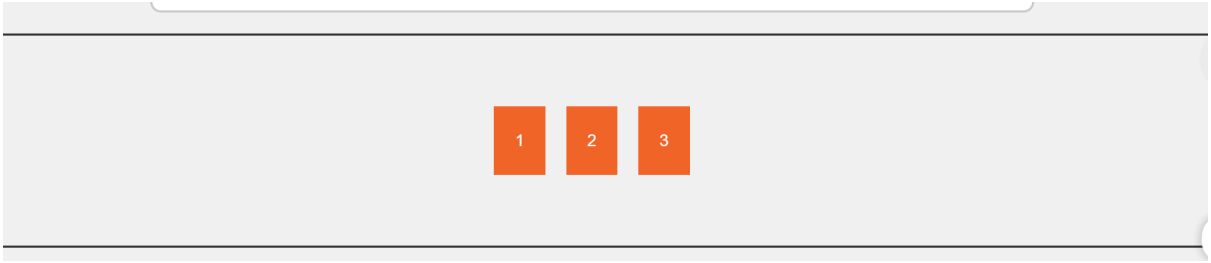
¿Por qué no solo nav?

Si solo usáramos el selector nav, los estilos se aplicarían a la etiqueta <nav> misma, pero nuestro objetivo es estilizar la lista (<ul>) que está dentro de ella. Las propiedades como list-style: none; y display: flex; funcionan sobre la lista, no sobre el contenedor nav.

¿Por qué no solo ul?

Si solo usáramos el selector ul, el estilo se aplicaría a **todas las listas no ordenadas** en toda tu página. Esto podría causar problemas si tienes otras listas en el <body> (por ejemplo, en un artículo o una sección) a las que no quieres dar el mismo estilo.

Al usar nav ul, te aseguras de que el estilo solo afecte a la lista que está específicamente dentro de la navegación. Esto hace que tu código sea más preciso, modular y fácil de mantener. Es una de las mejores prácticas para evitar conflictos de estilos inesperados.




```
.contenedor {
  display: flex;              /* Activamos Flexbox */
  justify-content: center;    /* Alineación horizontal */
  align-items: center;        /* Alineación vertical */
  height: 200px;
  border: 2px solid #333;
}

.caja {
  background: #f16529;        /* Naranja HTML */
  color: white;
  padding: 20px;
  margin: 10px;
}
```

CSS Grid: El Diseño Bidimensional

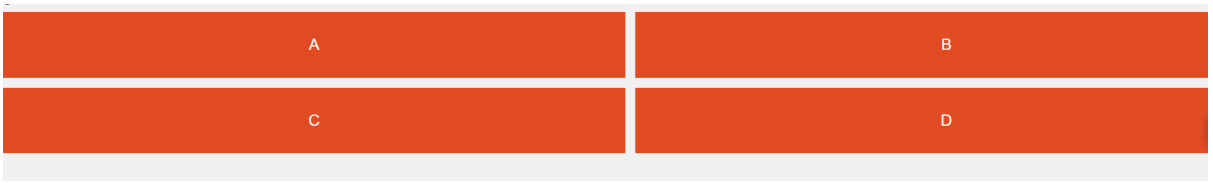
Si Flexbox resolvió el problema de la maquetación en una dimensión, **CSS Grid** lo hizo en dos. Por primera vez, los desarrolladores tenían una herramienta nativa para crear diseños complejos de filas y columnas, como los de una revista o un periódico. Antes, esto se lograba con tablas o frameworks de diseño muy complicados.

CSS Grid te permite definir una cuadrícula completa en un contenedor y luego colocar elementos en celdas específicas de esa cuadrícula. Es la herramienta ideal para la maquetación general de una página web, donde necesitas definir la posición de elementos como el encabezado, la barra lateral y el contenido principal.

En resumen, Flexbox y Grid no son solo nuevas características; son un cambio de paradigma en la maquetación web. Hacen que el diseño responsivo sea mucho más sencillo y que las páginas sean más robustas y flexibles en diferentes tamaños de pantalla.

Piensa en una cuadrícula como un tablero de ajedrez.

- Grid sirve para **diseños en dos dimensiones** (filas y columnas).
- Es perfecto para **páginas completas, galerías, dashboards…**
- Con Grid, tú defines las “líneas de la cuadrícula” y colocas los elementos dentro.




```
<div class="grid-contenedor">

  <div class="item">A</div>

  <div class="item">B</div>

  <div class="item">C</div>

  <div class="item">D</div>
```

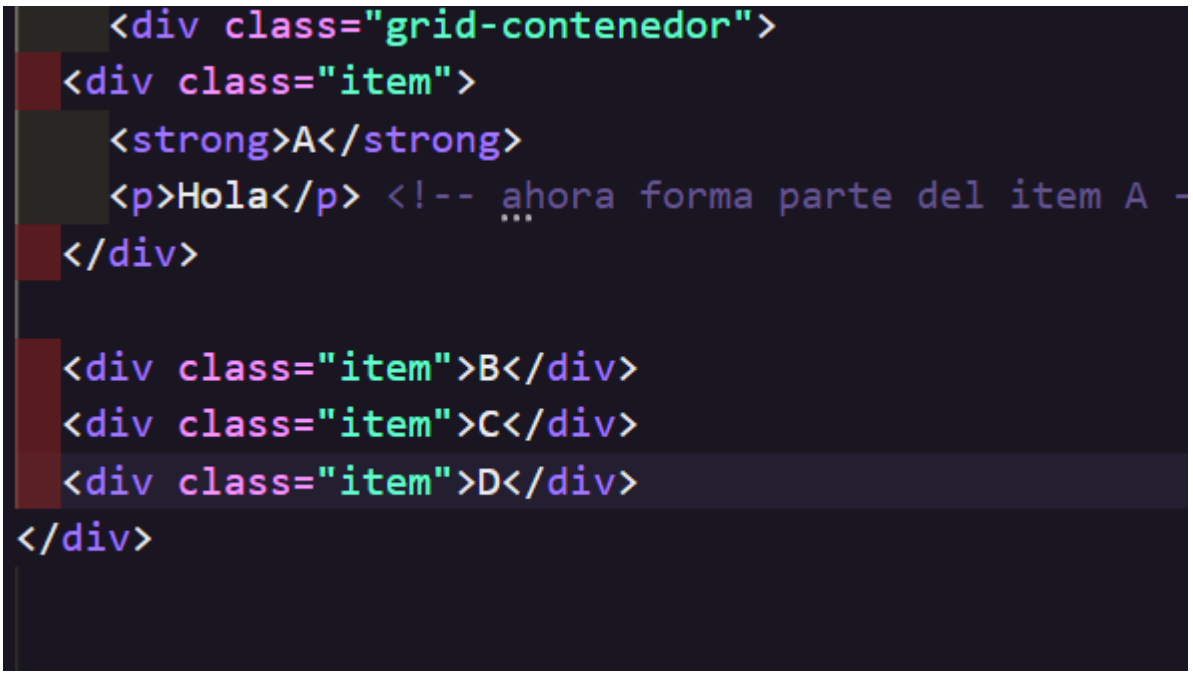
```
</div>
```

```
.grid-contenedor {
  display: grid;
  grid-template-columns: 1fr 1fr; /* 2 columnas iguales */
  grid-template-rows: auto auto;  /* 2 filas automáticas */
  gap: 10px;                      /* espacio entre celdas */
}

.item {
  background: #e44d26;
  color: white;
  padding: 20px;
  text-align: center;
}
```

Vamos a suponer que nosotros en esas rejillas queremos introducir algo recordad que no es una caja que solo es un tablero

entonces no podemos en el tablero que sería el grid.contenedor si no dentro de una de las cajas a las que hemos llamado item. Item no es una palabra reservada ni un selector de etiqueta pero es muy usado para los hijos de un flex o un grid.



ACTIVIDAD DE CAJAS.