HTML5 & CSS3

GUÍA DE DESARROLLO WEB FRONT-END

IES Fco. Grande Covián
José Antonio Sallán
Informática II - Versión 3.1.2

1 El estándar Web y sus tecnologías

El desarrollo web moderno se basa en un estándar que combina tres tecnologías fundamentales. Cada una tiene una responsabilidad única y diferenciada, permitiendo la creación de sitios web complejos y profesionales:

- **HTML (HyperText Markup Language):** Es el lenguaje de etiquetas que permite definir la **estructura y el esqueleto** de la página. Indica al navegador qué elementos hay (un título, un párrafo, una imagen).
- **CSS (Cascading Style Sheets):** Es el lenguaje de diseño que permite dar **formato y estilo** visual (colores, tipografías, disposiciones). Separa el contenido de la estética.
- **JavaScript (JS):** Es el lenguaje de programación que añade **interactividad, lógica y dinamismo** (menús desplegables, galerías de fotos, validación de datos).

Objetivo de la unidad

Dada la amplitud de estas tecnologías, en esta materia realizaremos una aproximación práctica al **HTML5** y **CSS3**. El objetivo es dominar las bases de esas tecnología para elaborar sitios web funcionales y bien estructurados.

2 Primeros conceptos HTML

HTML es un lenguaje de etiquetas que permite definir la estructura de un sitio web.

> **Etiqueta:** Bloque de código limitado por los símbolos menor que (<) y mayor que (>).

La inmensa mayoría de las etiquetas funcionan por pares:

1. **Etiqueta de apertura**: Define el inicio de un elemento (ej: <html>).
2. **Etiqueta de cierre**: Define el final del elemento. Se distingue por llevar una barra inclinada (/) antes del nombre de la etiqueta (ej: </html>).

> ⚠️ **Recomendación de Oro**: Todo el código de la página web debe estar anidado correctamente, formando una estructura de árbol, donde unas etiquetas contienen a otras sin solaparse.

Estructura Básica y Obligatoria

Según el estándar HTML5 que estudiamos en este curso, todo documento debe seguir este orden estricto:

1. **El Doctype:**

Lo primero que debe aparecer, incluso antes que la etiqueta html, es:

```
<!DOCTYPE html>
```

Esta línea no es una etiqueta HTML propiamente dicha, sino una instrucción para el navegador. Le indica: "Atención, este documento sigue el estándar moderno de HTML5". Es crucial para que la web se visualice correctamente.

2. **El contenedor raíz: <html>**

Es la etiqueta principal que envuelve todo el contenido de la web (excepto el Doctype). Normalmente, le añadiremos el atributo de idioma:

La etiqueta principal en cualquier página web que siga el estándar html5 es:

```
<html lang='es'>
  … contenido...
```

```
</html>
```

Todo el código de la página web estará incluido entre las dos etiquetas html.

Normalmente contendrá dos secciones, el head y el body.

3. **La Cabecera: <head>**

El código que se incluye aquí no es visible directamente en la página web. Se utiliza para definir metadatos e instrucciones para el navegador y los buscadores.

¿Qué solemos incluir en el <head>?

- **Título de la página**: (Etiqueta <title>) Es el texto que aparece en la pestaña del navegador.
- **Hojas de estilo CSS**: Enlaces a los archivos que darán color y forma a la web.
- **JavasCript**: Scripts para interactividad (que veréis en cursos superiores).
- **Iconos**: El favicon que aparece junto al título en la pestaña.
- **Etiquetas <meta>**: Son vitales para:
  - **Codificación**: Indica cómo gestionar tildes y eñes (ej: <meta charset="UTF-8">).
  - **Responsive**: Indispensable para que la web se adapte a móviles (ej: <meta name="viewport" content="...">).
  - **SEO Básico**: Descripción de la web para buscadores.

4. **El Cuerpo: <body>**

Es la sección del contenido. Contiene toda la información visible de la página web (textos, imágenes, vídeos, formularios, etc.). Es donde aplicaremos la estructura semántica que veremos en el próximo punto.

De momento hemos generado la siguiente estructura:

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

```
  </body>
</html>
```

Vamos a comenzar a programar.

> Aunque es posible utilizar un editor de texto plano (Editor de texto en Linux Mint o el Bloc de notas en Windows) existen programas especialmente diseñados para editar código que nos ofrecerán múltiples ventajas. En nuestro caso utilizaremos el editor ***Visual Studio Code*** de la empresa Microsoft.

3 Visual Studio Code – Estructura básica html

3.1 Instalación de Visual Studio Code

La forma más sencilla de instalar la aplicación en sistemas basados en Debian como Linux Mint es a través de la terminal.

Sin embargo, VSC no está disponible en los repositorios de Linux Mint (no es software 100% libre), por lo que es necesario iniciar el proceso agregando el repositorio oficial de Microsoft. Para ello seguiremos los siguientes pasos.

1. En primer lugar, actualizaremos los índices de paquetes existentes e instalaremos herramientas necesarias para gestionar descargas seguras (wget) y claves de cifrado (gpg).

```
sudo apt update
sudo apt install -y wget gpg
```

2. Añadiremos la clave GPG de Microsoft (control de seguridad descarga):

```
wget -qO- https://packages.microsoft.com/keys/microsoft.asc | gpg
--dearmor | sudo tee /usr/share/keyrings/microsoft.gpg > /dev/null
```

3. Añadiremos el repositorio de Visual Studio Code:

```
echo "deb [arch=amd64 signed-by=/usr/share/keyrings/microsoft.gpg]
https://packages.microsoft.com/repos/code stable main" | sudo
tee /etc/apt/sources.list.d/vscode.list
```

4. Actualizaremos la lista de paquetes nuevamente:

```
sudo apt update
```

5. Instalamos la aplicación Visual Studio Code:

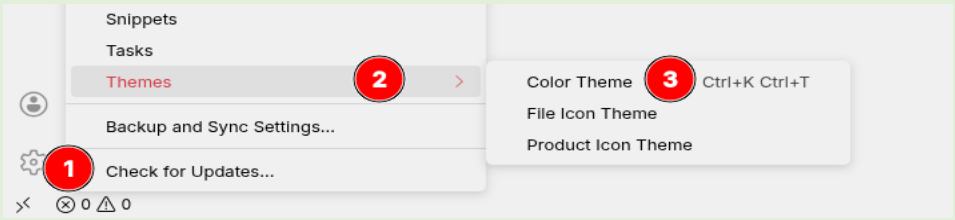
```
sudo apt install -y code
```

Podemos verificar la instalación ejecutando la apliación desde el terminal:

```
code
```

Consejo:

Puedes modificar la **apariencia** de la apliacación siguiendo la ruta:



Y seleccionado alguno delos temas predeterminados, or su claridad recomiendo el tema ***Atom One X GitHub – Light gray***

3.2 Configurando VSC para html 5

El primer paso antes de escribir una sola línea de código es preparar nuestro espacio de trabajo (Workspace). En desarrollo web, nunca trabajamos con archivos sueltos, sino con Sitios Web.

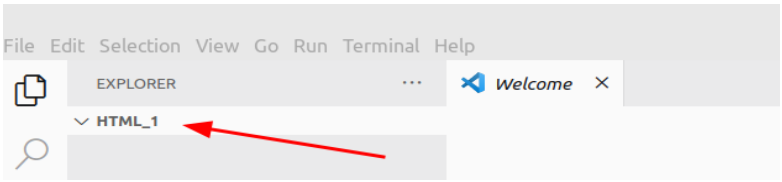
> **Sitio web:** Carpeta que contiene el conjunto de todos los archivos que vamos a utilizar en nuestro proyecto web.

Creación del sitio web

Iniciamos el proceso haciendo clic en el botón ***Open Folder...*** y creamos una carpeta en la que iremos guardando todos los archivos de nuestro sitio web. Le asignamos un nombre (en este ejemplo html_1).

Si la aplicación nos pregunta si confiamos en los autores de la carpeta, seleccionaremos la opción Yes.

Se mostrará la carpeta en la columna de la izquierda:



Creación del archivo principal (index.html)

En la columna de la izquierda (explorador), hacemos clic en el botón derecho y seleccionamos la opción ***New File...*** Creamos un archivo al que llamaremos index.html.

Automatización con Emmet

> **Emmet**: Herramienta incorporada en el editor de código Visual Studio Code que proporciona una forma abreviada y fácil de generar código HTML y CSS.

Si escribimos en la ventana de la derecha la expresión ***html*** se mostrará:

Por defecto funciona el ***asistente Emmet*** que ofrece tres opciones. Seleccionamos la segunda: ***html:5*** que creará una estructura básica de un archivo html5:

Gestión de cambios y guardado de la página

A la derecha del nombre de la página (index.html) se mostrará un punto. Eso quiere decir que todavía no hemos guardado los cambios. Pulsamos Ctrl+S para guardar. El punto se transformará en una x.

3.3 Estudiando la estructura básica generada

Al utilizar el atajo html:5 de Emmet, VS Code genera un código base. Vamos a analizar qué significa cada línea y cómo debemos adaptarla a nuestro proyecto en España.

El atributo de idioma

La ***etiqueta html*** admite un atributo:

```
<html lang="en">
```

Indica que el lenguaje utilizado para el sitio de web es inglés. Si queremos utilizar el español como lenguaje para nuestra web, modificaremos la etiqueta:

```
<html lang="es">
```

Esto ayuda a los lectores de pantalla (accesibilidad) y a que los buscadores no ofrezcan la traducción automática de la página.

La sección <head>: Metadatos técnicos

Dentro de la sección ***head*** aparecen tres etiquetas. Dos etiquetas meta (*no tienen etiqueta equivalente de cierre*), y una etiqueta title. Veamos su significado:

Codificación de caracteres

```
<meta charset="UTF-8" />
```

Indica al navegador que use el estándar **UTF-8**, el cual incluye soporte para eñes, tildes y caracteres especiales de casi todos los idiomas del mundo. Sin ella, estos caracteres podrían verse como símbolos extraños.

Diseño reponsive (Viewport)

```
<meta name="viewport" content="width=device-width,
initial-scale=1.0" />
```

Es la instrucción que permite que la web sea **Responsive**. Le dice al navegador: "ajusta el ancho del contenido al ancho de la pantalla del

dispositivo (width=device-width) y empieza con un zoom de 1:1 (initial-scale=1.0)". Sin esta línea, tu web se vería diminuta en un móvil.

Título de la pestaña

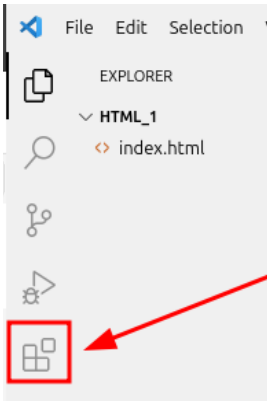
```
<title>Document</title>
```

Contiene la expresión a mostrar como título en la barra del navegador. También es una expresión que los buscadores estudiaran para indexar nuestra web. En este caso podemos escribir:

```
<title>Mi primer sitio</title>
```

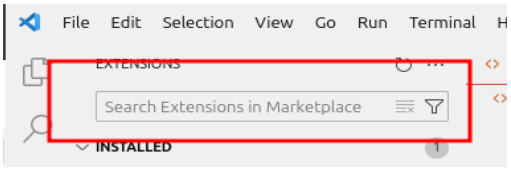
3.4 Instalando extensiones

Para transformar VS Code en una herramienta de alto rendimiento, instalaremos "extensiones": pequeños complementos que añaden funcionalidades extra.



En la barra de herramientas de la columna de la izquierda seleccionamos la opción Extensions (Ctrl+Shift+X):

En la caja de selección iremos escribiendo el nombre de las extensiones a instalar.



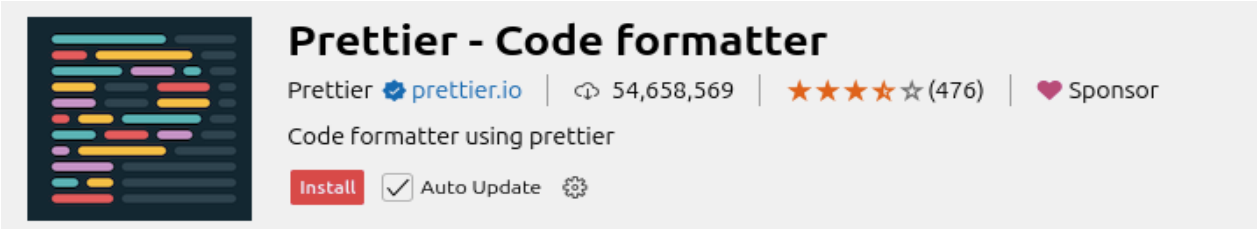
En este apartado instalaremos la extensión Prettier y un servidor local.

Prettier – Formateador de código:

Es la extensión encargada de que nuestro código sea legible, elegante y cumpla con las normas de indentación (espaciado).

Instalación y Configuración Única:

Buscamos "Prettier" en la caja de selección y la instalamos.



Para activarla por primera vez, haremos clic derecho sobre nuestro código en index.html y elegiremos la opción Format Document.

Si VS Code nos pregunta, seleccionaremos "Configure Default Formatter..." y elige Prettier.

Automatización (Format on Save):

Para no tener que realizar el proceso de formateo a mano, activaremos el "auto-formateo" al guardar:

1. Vamos al icono del engranaje (inferior izquierda) → Settings.
2. Buscamos "Format on Save" y marcamos la casilla.

A partir de ahora, cada vez que pulses Ctrl + S, tu código se ordenará mágicamente.

Comprobamos el funcionamiento de prettier

Escribe un código sencillo pero con un formato incorrecto en index.html:

```
<body>
  <h1>Hola Mundo
                    </h1>
</body>
```

Guarda el documento y el el resultado será:

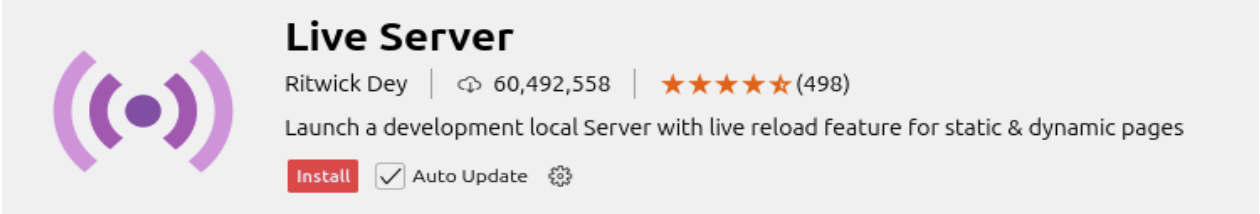
```
  <body>
    <h1>Hola Mundo</h1>
  </body>
</html>
```

Live Server

Live Server es un ***servidor local:***

> **Servidor local:** Aplicación que permite ver nuestro sitio web en nuestro propio ordenador a través de un navegador.

Al realizar cambios en nuestro sitio se mostrarán en tiempo real en el navegador.



Cómo utilizarlo:

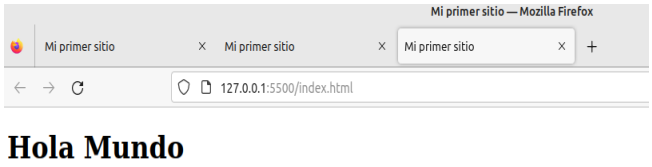
Haz clic derecho sobre tu archivo index.html.

Selecciona Open with Live Server.

Se abrirá tu navegador predeterminado apuntando a una dirección similar a http://127.0.0.1:5500.

> Concepto Técnico: El número 5500 es el puerto. Es como la "puerta de entrada" específica que usa este servidor para enviarte la web a tu pantalla.

De momento nuestra página web se mostrará:



Se mostrará el título en la barra de título del navegador. En la parte de contenido se mostrará la expresión “Hola Mundo” en una fuente negrita y de tamaño grande.

4 Conectar a un servidor remoto vía FTP

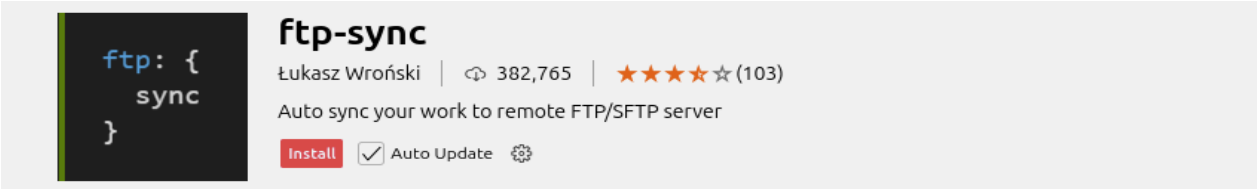
Basado en: https://www.youtube.com/watch?v=omRjSqlRRUw

Una vez que nuestra web funciona en nuestro ordenador (Local), el siguiente paso es subirla a un servidor en internet (Remoto). Para ello utilizaremos el protocolo FTP (File Transfer Protocol).

Instalación de la extensión

Para no depender de programas externos como FileZilla, integraremos el FTP directamente en VS Code.

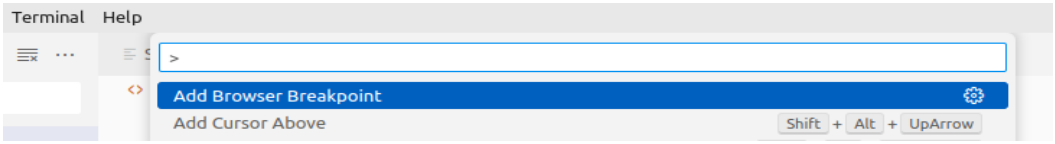
Buscamos e instalamos la extensión ***ftp-sync***:



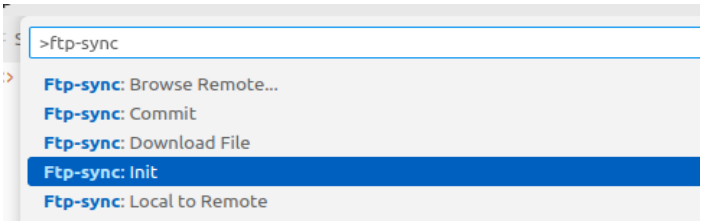
Configuración del archivo ftp-sync.json

Para que la extensión sepa a dónde enviar los archivos, debemos crear un archivo de configuración, para ello:

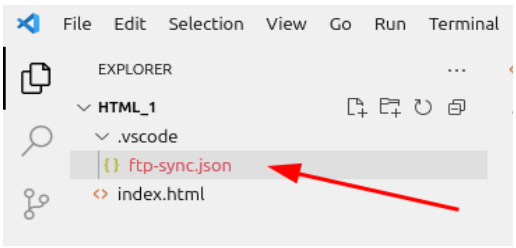
Entramos en el ***Command Palette*** pulsando F1:



En la ventana escribimos ftp-sync. Se abrirán las opciones de la aplicación y seleccionamos init:



Se creará y abrirá un fichero ftp-sync.json, en la raíz de nuestro proyecto.



Su código contiene una configuración genérica de conexión:

```
{
  "remotePath": "./",
  "host": "host",
  "username": "username",
  "password": "password",
  "port": 21,
  "secure": false,
  "protocol": "ftp",
  "uploadOnSave": false,
  "passive": false,
  "debug": false,
  "privateKeyPath": null,
  "passphrase": null,
  "agent": null,
  "allow": [],
  "ignore": [
    "\\.vscode",
    "\\.git",
    "\\.DS_Store"
  ],
  "generatedFiles": {
    "extensionsToInclude": [
      ""
    ],
    "path": ""
  }
}
```

A modo de ejemplo modificaremos el código anterior creando una conexión a un servidor byethost, modificando las siguientes opciones: tal y como se muestra en la siguiente imagen:

```
{
  "remotePath": "./htdocs",
  "host": "**********************",
  "username": "******************",
  "password": "******************",
  "port": 21,
```

```
  "secure": false,
  "protocol": "ftp",
  "uploadOnSave": true,
  "passive": false,
  "debug": false,
  "privateKeyPath": null,
  "passphrase": null,
  "agent": null,
  "allow": [],
  "ignore": [
    "\\.vscode",
    "\\.git",
  "\\.DS_Store"
  ],
  "generatedFiles": {
    "uploadOnSave": true,
    "extensionsToInclude": [
      ".html",
      ".php",
      ".css",
      ".js"
    ],
    "path": ""
  }
}
```

Los datos **remotePath**, **host**, **username**, **password** serán los obtenidos a través de tu servidor de servicios en Intenet.

Existen varias empresas que permiten alojar tus datos en sus servidores de forma gratuita. Este curso utilizaremos los servidores de la empresa Byethost.

En la ultima sección de estos apuntes puedes encontrar un tutorial que te guiará a realizar la configuración de este servicio.

Observa que se han modificado varios datos en el archivo de configuración:

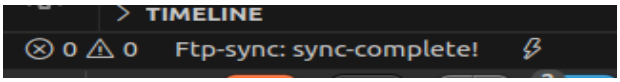
- Como "***remotePath***" hemos escrito "./htdocs" ya que es la carpeta en la que tendremos que guardar nuestro sitio web en el servidor de byethost".
- Cambiamos la opción **u*ploadOnSave to "true"***
- Esa línea se copia también en la sección ***"generatedFiles"*** tal y como se muestra en la imagen
- En la sección "***extensionsToInclude***", añadiremos las extensiones que queremos que suban al sitio automáticamente. De momento añadimos html, php, css, js.

Guardamos el archivo ftp-sync.json y lo podemos cerrar.

Sincronización inicial

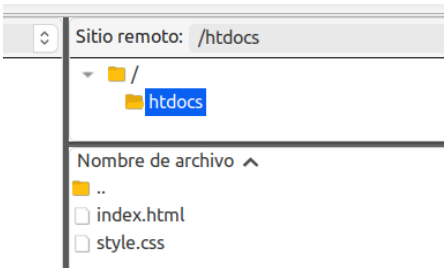
La primera vez debemos subir todo el proyecto de golpe haremos:

3. Clic en F1 para abrir la ***Command Palette...,*** teclea ***ftp-sync: Local to Remote***
4. Seleccionamos la opción ***.Choose this folder***, ahora ***full sync*** y por último ***Run***
5. Aparece un mensaje en la esquina inferior izquierda:

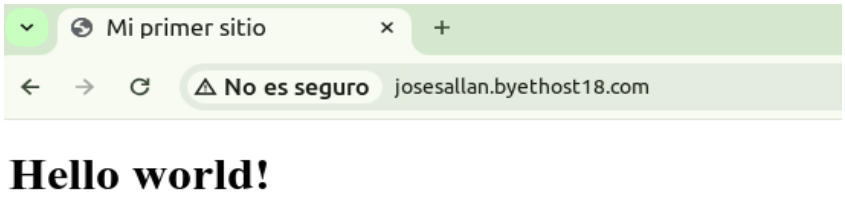


Todo ha ido bien.

Si vamos a un cliente ftp (filezilla) veremos que se ha subido la carpeta con el proyecto y su contenido al interior de la carpeta htdocs:



Si visitamos ahora la dirección web de nuestro sitio, obtendremos el resultado:



5 Hipervínculos – Ideas básicas

En este apartado vamos a añadir una segunda página en nuestro sitio web y aprenderemos a enlazarla con la página principal utilizando hipervínculos.

> **Hipervínculo:** Elemento de un documento electrónico que permite acceder a otro recurso, ya sea dentro del mismo documento o en otro lugar de la web, haciendo clic en él.

Sintaxis básica:

Aunque tiene muchos más parámetros la etiqueta que crea un hipervínculo tiene la forma:

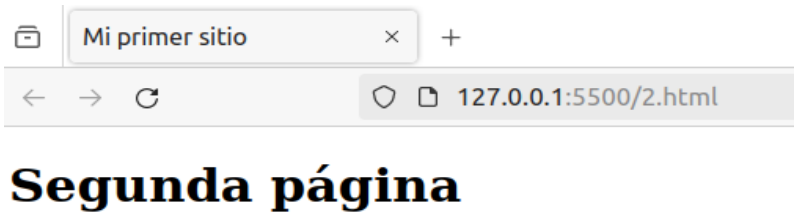
```
<a href=""></a>
```

- El atributo **href** especifica la URL del recurso al que queremos enlazar.
- **Dentro de las comillas** escribiremos el destino al que nos queremos redirigir al hacer clic en el vínculo.
- Entre las etiquetas escribiremos el ***texto*** del elemento que ha de ***mostrar*** el enlace.

Veamos en nuestro ejemplo:

En primer lugar crearemos una segunda página en nuestro sitio al que llamaremos 2.html.

Siguiendo los pasos descritos en los puntos anteriores, intenta obtener el siguiente resultado:



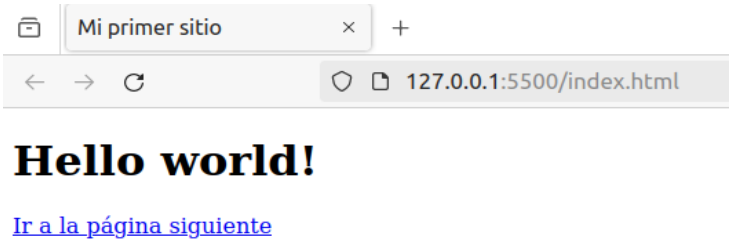
Tenemos de momento dos páginas. La primera, index.html, con el título "Hello world!" y la segunda, 2.html, con el título "Segunda página".

Para crear el hipervínculo que nos permita ir de la página principal a la página 2.html, podemos escribir tras las etiquetas <h1>:

```
<body>
  <h1>Hello world!</h1>
  <p><a href="2.html">Ir a la página
  siguiente</a></p>
</body>
```

Hemos incluido el hipervínculo dentro del texto de una etiqueta <p> pero podría estar asociado a cualquier otro elemento de nuestra página.

El destino del hipervínculo será la página 2.html. El resultado se mostrará en el navegador en la forma:



Observa como el texto "Ir a la página siguiente" aparece subrayado y con un color azul. Este es el formato por defecto para los hipervínculos en el navegador Firefox.

Si haces clic en el enlace, serás redirigido a la página 2.html.

Modifica ahora el código de la página 2.html para obtener un resultado similar al de la imagen que te permita regresar a la página index.html:

Intenta conseguir, como el la imagen, que en este caso el hipervínculo sea solamente la expresión "página de inicio".

En ejemplo hemos creado un enlace a otra página web de nuestro sitio. Si quisiéramos enlazar con una página externa (de otro sitio web) sería suficiente con escribir dicha dirección como valor del atributo href.

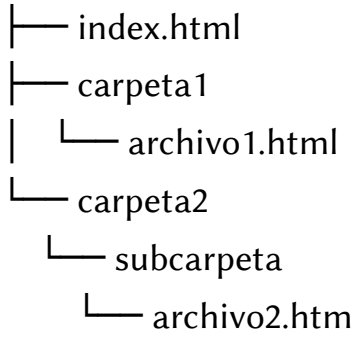
Más adelante aprenderemos a configurar parámetros en este tipo de etiquetas.

5.1 Subdirectorios o directorios padres

En el ejemplo anterior, las páginas de origen y de destino están dentro del mismo directorio. Cuando una etiqueta tiene que buscar el elemento indicado a través del parámetro href, lo hace dentro del directorio en la que está el documento que contiene el enlace.

Si nuestro sitio web está organizado en carpetas, esto originaría múltiples errores.

Supongamos que tienes la siguiente estructura de directorios:



Trabajando como lo hemos hecho hasta ahora, sería imposible enlazar por ejemplo desde la página index.html a la página archivo1.html.

Veamos como resolver esta situación:

Enlaces a archivos en subcarpetas

Para enlazar a archivos ubicados en subcarpetas, escribe las rutas relativas desde el archivo actual. Por ejemplo, si estás en index.html y deseas enlazar a archivo2.html, que está en carpeta2/subcarpeta, puedes hacerlo así:

```
<a href="carpeta2/subcarpeta/archivo2.html">Ir a archivo</a>
```

Enlaces a un nivel superior

Para enlazar a archivos ubicados en carpetas de nivel superior, es necesario especificar la ruta relativa desde la raíz del sitio. Para ir a la raíz del sitio utiliza el código "../".

Por ejemplo, si estás en carpeta1/archivo1.html y deseas enlazar a index.html, puedes hacerlo así:

```
<a href="../index.html">Ir a la página principal</a>
```

Intenta pensar como podríamos hacer un enlace entre la página archivo1.html y la página archivo2.html.

5.2 Atributo target en la etiqueta <a>

El atributo **target** es utilizado en la etiqueta de hipervínculo <a> define cómo se abre el enlace al hacer clic en él.

Hay varias opciones;

- **target="_blank"**: Este valor hace que el enlace se abra en una nueva ventana o pestaña del navegador, proporcionando al usuario una experiencia de navegación no disruptiva.
- **target="_self"**: Con este valor, el enlace se abre en la misma ventana o pestaña en la que se encuentra la página actual. Es útil cuando no queremos que la navegación interrumpa la experiencia de lectura actual.
- **target="_parent"**: Utilizado en páginas que están incrustadas en un marco (frame), este valor abre el enlace en el marco padre del marco actual.
- **target="_top"**: Abre el enlace en la ventana principal del navegador, reemplazando cualquier marco existente. Es útil cuando queremos que el enlace tenga un efecto en toda la ventana del navegador.

El atributo target permite mejorar la experiencia de usuario y controlar la forma en que los enlaces interactúan con el entorno de navegación del usuario.

6 Modelo de Cajas – Introduciendo CSS

Hasta ahora hemos creado la estructura de nuestra página usando HTML. El resultado visual es el que el navegador aplica por defecto (texto negro, fondo blanco).

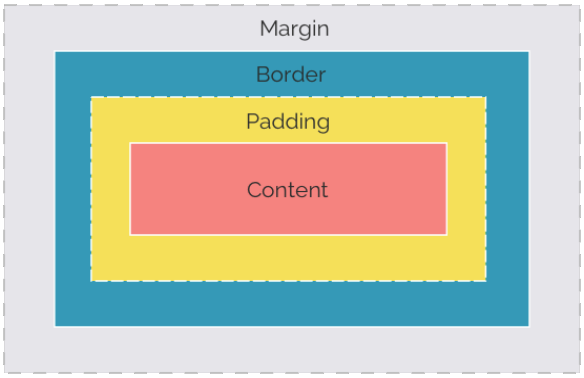
Antes de comenzar a dar aspecto de "verdadera página web" a nuestro trabajo, debemos entender que **todos los elementos en HTML son cajas rectangulares.**

Un párrafo (<p>), un título (<h1>), una imagen (<img>) o un enlace (<a>)... todos son cajas invisibles.

Anatomía de una Caja HTML

El "Box Model" (Modelo de Cajas) establece que cada elemento está formado por cuatro capas concéntricas, desde dentro hacia afuera:

1. **Contenido (Content):** Es el corazón de la caja. Puede ser el texto, una imagen o un vídeo.
2. **Relleno (Padding):** Es el espacio transparente o "aire" que hay *entre el contenido y el borde* de la caja. Evita que el texto se pegue a los bordes.
3. **Borde (Border):** La línea que rodea la caja. Puede ser visible (continua, de puntos) o invisible.
4. **Margen (Margin):** Es el espacio exterior o "aire" que hay *por fuera del borde*. Sirve para separar nuestra caja de las cajas vecinas.



Nuestra primera prueba con CSS

Para cambiar colores, tamaños y posiciones, necesitamos usar un nuevo lenguaje: CSS (Hojas de Estilo en Cascada).

Para ver estas cajas en acción, vamos a crear un componente sencillo (una tarjeta de perfil) y le daremos color.

¿Cómo conectamos HTML y CSS?

Para mantener el orden, normalmente separaremos el contenido del diseño.

Escribiremos nuestro HTML en un archivo (index.html) y nuestros estilos en otro archivo nuevo (por ejemplo, estilos.css).

Para que el archivo HTML "sepa" dónde están sus estilos, debemos enlazarlos. Esto se hace dentro de la sección <head> (ya que es información técnica) usando la etiqueta <link>:

```
<head>
  <meta charset="UTF-8" />
  <title>Probando Cajas</title>
  <link rel="stylesheet" href="estilos.css" />
</head>
```

- rel="stylesheet": Indica que el archivo enlazado es una hoja de estilos.
- href="estilos.css": Es la ruta donde se encuentra nuestro archivo CSS.

Agrupando elementos: La etiqueta y las Clases

Antes de dar diseño, necesitamos aprender a empaquetar nuestro contenido.

Imagina que quieres hacer una "tarjeta de perfil" que contenga una foto y un texto. Para que ambos elementos se muevan y decoren juntos, debemos meterlos en una caja contenedora.

La etiqueta reina para crear contenedores genéricos es (de division).

```
<div>
  <h1>Mi Primer Componente</h1>
  <p>Este texto está dentro de una caja a la que le
  vamos a dar estilo.</p>
</div>
```

Problema:

En una web real habrá decenas de . ¿Cómo le decimos a CSS que queremos pintar solo el del perfil y no los demás?

Solución:

El atributo **class** (clase), permite ponerle una "etiqueta o nombre" a cualquier elemento HTML para luego llamarlo desde CSS.

```
<div class="mi-tarjeta">
  <h1>Mi Primer Componente</h1>
  <p>Este texto está dentro de una caja a la que le
  vamos a dar estilo.</p>
</div>
```

Vamos a poner todo lo anterior en práctica.

Nuestro html tendrá la forma:

```
<!doctype html>
<html lang="es">
  <head>
    <meta charset="UTF-8" />
    <title>Probando Cajas</title>
    <link rel="stylesheet" href="estilos.css" />
  </head>
  <body>
    <div class="mi-tarjeta">
      <h1>Mi Primer Componente</h1>
      <p>Este texto está dentro de una caja a la que
      le vamos a dar estilo.</p>
    </div>
  </body>
</html>
```

El archivo CSS (estilos.css)

Creamos un archivo nuevo llamado estilos.css en la misma carpeta. Usaremos CSS para "llamar" a la clase .mi-tarjeta y pintarla. Escribe lo siguiente:

```
/* Seleccionamos la caja por su nombre de clase (el
punto es obligatorio) */
.mi-tarjeta {
  background-color: lightblue; /* Color de fondo */
```

```
  border: 3px solid darkblue; /* Borde visible */
  padding: 20px; /* Aire dentro: el texto respira */
  margin: 30px; /* Aire fuera: separa la caja de los
  márgenes de la pantalla */
  text-align: center; /* Centra el texto */
  border-radius: 10px; /* ¡Bordes redondeados! */
}
```

En nuestro archivo CSS, para llamar a un elemento por su clase, siempre debemos poner un punto (`.`) delante del nombre:

Ver el resultado

Abre tu index.html con *Live Server*. ¡Acabas de transformar un texto soso en un componente visual atractivo!

💡 **Ejercicio para clase:** Prueba a cambiar los valores numéricos del padding y el margin en tu archivo CSS (ponles 50px o 0px). Guarda y observa en el navegador cómo la caja "respira" por dentro o se separa por fuera. Esta es la base de la maquetación web.

6.1 Sintaxis básica de CSS

Antes de seguir, debemos entender las reglas de escritura de este nuevo lenguaje. En CSS, a cada bloque de código se le llama Regla CSS, y siempre tiene la misma estructura anatómica:

```
selector {
  propiedad: valor;
  propiedad: valor;
}
```

Vamos a diseccionar qué significa cada parte usando nuestro ejemplo:

1. El Selector:

Es la primera palabra de la regla (por ejemplo, .mi-tarjeta o h1). Le indica al navegador a qué elemento HTML le vamos a aplicar los cambios.

2. Las Llaves { } (El contenedor):

Todo lo que escribamos dentro de las llaves se aplicará únicamente al selector que hemos puesto delante.

La llave de apertura { significa: "Aquí empiezan los estilos".

La llave de cierre } significa: "Aquí terminan los estilos para este elemento".

3. Declaraciones (Propiedad : Valor):

Dentro de las llaves damos las instrucciones. Siempre funcionan por parejas:

- **Propiedad**: Es la característica que queremos cambiar (el color, el tamaño, el margen...). Se escribe a la izquierda.
- **Dos puntos**: Separan la propiedad de su valor.
- **Valor**: Es el resultado a aplicar (rojo, 20 píxeles, centrado...).

4. El punto y coma ; :

Cada instrucción en CSS debe terminar con un punto y coma. Es la forma de decirle al navegador "he terminado con esta orden, pasa a la siguiente". Si nos olvidamos de poner un punto y coma, la siguiente línea de código dará error y no funcionará.

5. El formato visual (Una línea por propiedad):

Al ordenador le da igual que escribamos todo en una sola línea larguísima. Sin embargo, los programadores escriben una propiedad en cada línea y dejan un espacio (tabulación) por la izquierda. Esto hace que el código sea mucho más fácil de leer y de corregir si hay errores.

7 Selectores, Colores y Tipografías

En el punto anterior conseguimos dar forma a nuestra caja (.mi-tarjeta) usando colores básicos en inglés (lightblue, darkblue).

Sin embargo, en el diseño web profesional no se trabaja así. Las páginas modernas requieren precisión en los colores, fuentes de texto elegantes y estilos específicos para cada elemento.

Vamos a tomar la tarjeta que ya tenemos y le daremos un acabado más profesional.

7.1 Colores: Más allá de los nombres básicos

Los profesionales no usan palabras como *red* o *blue*, ya que limitan mucho la paleta. En las pantallas, cualquier color que ves es en realidad una mezcla de tres luces: **Rojo (Red), Verde (Green) y Azul (Blue)**. A esto se le llama el modelo **RGB**.

Para crear un color, le decimos al monitor cuánta intensidad queremos de cada luz, usando una escala que va del **0 (apagado)** al **255 (intensidad máxima)**.

Podemos escribir esto en CSS de dos formas:

A) El método RGB:

Escribimos directamente los tres valores (del 0 al 255) entre paréntesis.

- color: rgb(255, 0, 0); Rojo al máximo, verde apagado, azul apagado = **Rojo puro**.
- color: rgb(0, 0, 0); Todas las luces apagadas = **Negro**.
- color: rgb(255, 255, 255); Todas las luces al máximo = **Blanco**.

B) El método Hexadecimal (El estándar profesional):

Es exactamente lo mismo que el RGB, pero escrito en un idioma más corto que usan los ordenadores (base 16 en lugar de base 10). En hexadecimal, en lugar de contar del 0 al 255, contamos del **00** al **FF**.

Un color hexadecimal siempre empieza por una almohadilla # seguida de 6 caracteres (dos para el Rojo, dos para el Verde y dos para el Azul):

- #FF0000 (FF de rojo, 00 de verde, 00 de azul) = **Rojo puro**.
- #000 000 (00 en todos) = **Negro**.
- #FFFFFF (FF en todos) = **Blanco**.
- #2c3e50 Es una mezcla específica que da como resultado un **azul oscuro elegante**.

7.2 Tipografías: Google Fonts

El navegador usa por defecto fuentes que hacen que la web parezca antigua.

Hoy en día, el estándar es utilizar el catálogo gratuito de Google Fonts.

1. **Selección en la web:** Vamos a fonts.google.com y buscamos una fuente moderna (ej: Roboto, Montserrat, Open Sans).
2. **Elegir estilos:** Seleccionamos los grosores que vamos a usar (ej: Regular 400 y Bold 700). *Consejo: no selecciones demasiados o la web tardará más en cargar.*
3. **Vincular en HTML:** Google generará un código <link> que debemos copiar y pegar dentro del <head> de nuestra página HTML, siempre antes de nuestro propio archivo .css.
4. **Aplicar en CSS:** Por último, debemos decirle a CSS dónde usar la fuente de Google mediante la propiedad font-family.

Ejemplo: Para aplicar la fuente Montserrat a todo el documento, en primer lugar añadiríamos a la sección head:

```
<link rel="preconnect"
href="https://fonts.googleapis.com">
<link rel="preconnect"
href="https://fonts.gstatic.com" crossorigin>
<link href="https://fonts.googleapis.com/css2?
family=Montserrat:ital,wght@0,100..900;1,100..900&dis
play=swap" rel="stylesheet">
```

A continuación, añadiríamos esto a nuestro archivo CSS:

```
body{
  font-family: 'Montserrat', sans-serif;
}
```

7.3 Selectores Descendientes: Apuntando con precisión

Hasta ahora hemos pintado toda la caja entera llamando a .mi-tarjeta. Pero, ¿qué pasa si queremos que el título (<h1>) tenga un color oscuro y el párrafo (<p>) sea gris clarito?

Para ello usamos los Selectores Descendientes. Consiste en escribir la clase de la caja padre, dejar un espacio, y escribir la etiqueta del hijo al que queremos apuntar:

```
.mi-tarjeta h1 { ... } (Afecta SOLO a los h1 que
estén dentro de la tarjeta).

.mi-tarjeta p { ... } (Afecta SOLO a los p que estén
dentro de la tarjeta).
```

7.4 Práctica: Evolucionando nuestra tarjeta

Vamos a aplicar estos tres conceptos para modernizar nuestra tarjeta. Además, cambiaremos ese feo borde sólido por un efecto muy utilizado hoy en día: el box-shadow (sombra de caja).

Paso 1: Actualizamos el archivo HTML (index.html)

Solo tenemos que añadir el enlace a la fuente de Google en la cabecera:

```
<head>
  <meta charset="UTF-8" />
  <title>Probando Cajas</title>
  <link href="https://fonts.googleapis.com/css2?
  family=Roboto:wght@400;700&display=swap"
  rel="stylesheet">
  <link rel="stylesheet" href="estilos.css" />
</head>
```

Paso 2: Actualizamos el archivo CSS (estilos.css)

Borramos el CSS definido en el punto anterior y escribimos:

```
/* 1. ESTILOS GENERALES DE LA PÁGINA */
body {
```

```
  font-family: 'Roboto', sans-serif; /* Aplicamos la
  nueva fuente a todo */
  background-color: #f4f4f9; /* Fondo gris clarito
  para la pantalla */
}

/* 2. ESTILOS DE LA CAJA CONTENEDORA */
.mi-tarjeta {
  background-color: #ffffff; /* Fondo blanco limpio
  para la tarjeta */
  padding: 30px;
  margin: 40px auto; /* 'auto' centra la caja en la
  pantalla */
  max-width: 400px; /* Evita que la tarjeta ocupe
  todo el monitor */
  text-align: center;
  border-radius: 15px;
  /* Quitamos el border solid y ponemos una sombra
  elegante */
  box-shadow: 0px 4px 12px rgba(0, 0, 0, 0.1);
}

/* 3. SELECTORES DESCENDIENTES (Estilos para el
contenido) */
.mi-tarjeta h1 {
  color: #2c3e50; /* Título oscuro y elegante */
  margin-top: 0; /* Quitamos el espacio extra que
  traen los h1 por defecto */
}

.mi-tarjeta p {
  color: #666666; /* Un texto gris se lee mejor que
  un negro puro */
  line-height: 1.5; /* Separamos un poco las líneas
  de texto para que respiren */
}
```

¡Guarda y visualiza en Live Server! Has pasado de tener un documento básico a un componente real de diseño web.

7.5 Explicando el CSS

Si observas el código CSS anterior, hemos introducido varias propiedades nuevas que son fundamentales en el diseño web moderno. Vamos a analizar qué hace cada una:

A) Estilos a nivel de página (La etiqueta body)

```
body {
  font-family: 'Roboto', sans-serif;
  background-color: #f4f4f9;
}
```

En CSS, las propiedades se heredan. Si le decimos a la etiqueta <body> (que engloba toda la página visible) que use la fuente Roboto, todos los elementos que estén dentro (títulos, párrafos, listas) heredarán esa fuente automáticamente sin tener que ponérsela uno por uno.

Además, le damos un color de fondo gris muy claro a toda la pantalla para que nuestra tarjeta blanca destaque sobre él.

B) El truco para centrar cajas (margin: auto)

```
margin: 40px auto;
max-width: 400px;
```

Centrar una caja en la pantalla es algo que harás constantemente. Para lograrlo, primero debemos limitar el ancho de la caja (con max-width). Si la caja ocupa el 100% de la pantalla, es imposible "centrarla".

Una vez limitada, usamos el valor auto en el margen. La instrucción margin: 40px auto; le dice al navegador: "Pon 40 píxeles de margen arriba y abajo. Para la izquierda y la derecha (auto), calcula el espacio libre que queda en la pantalla y repártelo a partes iguales".

C) Estética moderna (border-radius y box-shadow)

```
border-radius: 15px;
box-shadow: 0px 4px 12px rgba(0, 0, 0, 0.1);
```

Las webs antiguas eran cuadradas y planas. Hoy en día buscamos un efecto más suave y tridimensional:

- **border-radius**: Suaviza las esquinas de la caja. Cuanto mayor sea el número de píxeles, más redonda será la esquina.
- **box-shadow**: Crea una sombra detrás de la caja para que parezca que está "flotando". Sus cuatro valores significan:

- 0px: Desplazamiento horizontal de la sombra.
- 4px: Desplazamiento vertical (la sombra cae hacia abajo).
- 12px: Desenfoque (para que la sombra sea suave y no un bloque negro).
- rgba(...): El color de la sombra. Usamos negro con un 10% de opacidad (0.1) para que sea muy sutil.

8 Flexbox: Organizando cajas

Hasta este momento, nuestra tarjeta se ve genial, pero si creamos tres tarjetas en nuestro HTML, veremos que se colocan una exactamente debajo de la otra. Esto se debe a que, por defecto, los son elementos de bloque y ocupan todo el ancho disponible.

Para diseñar páginas web reales (con barras laterales, cuadrículas de productos o galerías), necesitamos colocar elementos uno al lado del otro.

La herramienta necesaria para hacer esto se llama CSS Flexbox (Cajas Flexibles).

La regla fundamental: El Padre y los Hijos

Con Flexbox no le decimos a una caja cómo debe comportarse, se lo decimos a la caja que la contiene. Para usar Flexbox siempre necesitamos dos niveles:

- Un **Contenedor Padre** (al que le daremos la orden de ser flexible).
- **Los Elementos Hijos** (las cajas que queremos alinear, que están dentro del padre).

Las tres propiedades clave de Flexbox

Todas estas instrucciones se aplican siempre a la caja Padre:

- **display: flex;**

  Al poner esto en el padre, las cajas hijo se colocan en fila horizontal, uno al lado del otro.

- **gap: 20px;**

  El contenedor padre decide que haya exactamente 20 píxeles de "aire" entre las cajas hijo.

- **Justify-content:**

> Tiene sentido solamente si las cajas hijo tienen tamaño fijo

  Decide cómo se distribuyen los hijos horizontalmente si sobra espacio.

  - **center**: Todas las cajas hijas juntas desde el centro.
  - **space-between**: El primer hijo pegado a la izquierda, el último a la derecha, y el resto de hijos repartidos uniformemente.
  - **space-around**: Espacio igual alrededor de todas las cajas.

Práctica: Nuestra primera galería de tarjetas

Vamos a transformar nuestro proyecto de una sola tarjeta a una galería de tres tarjetas alineadas perfectamente.

Paso 1: El HTML (Creando al Padre)

Vamos a envolver nuestras tarjetas en un nuevo que actuará como contenedor flex. Le pondremos la clase contenedor-flex.

```
<body>
  <div class="contenedor-flex">
    <div class="mi-tarjeta">
      <h1>Tarjeta 1</h1>
      <p>Este es el primer componente de nuestra
      galería.</p>
    </div>
    <div class="mi-tarjeta">
      <h1>Tarjeta 2</h1>
      <p>Flexbox hace que alinear esto sea súper
      sencillo.</p>
    </div>
    <div class="mi-tarjeta">
      <h1>Tarjeta 3</h1>
      <p>Ya no tenemos que usar propiedades antiguas
      como float.</p>
    </div>
  </div>
</body>
```

Paso 2: El CSS

Añade esta nueva regla a tu archivo estilos.css. Fíjate que se la aplicamos al .contenedor-flex, no a las tarjetas individuales.

```
/* 2. Configuramos la caja padre para que organice a
sus hijos */
.contenedor-flex {
  display: flex; /* Enciende Flexbox: Los hijos se
  ponen en fila */
  gap: 30px; /* Separa las tarjetas 30 píxeles entre
  sí */
  justify-content: center; /* Centra todo el bloque
  en  la pantalla */
  padding: 40px; /* Un poco de aire por los bordes de
  la pantalla */
```

```
}
```

> Nota: Asegúrate de que tu clase .mi-tarjeta en el CSS ya no tenga el margin: 40px auto; del punto anterior, ya que ahora es el padre quien se encarga de posicionarlas.

Al guardar y abrir en el navegador, verás cómo tus tres tarjetas se han alineado horizontalmente.

9 Estructura Semántica HTML5

Hasta ahora hemos usado la etiqueta para crear todas nuestras cajas. Si quisiéramos hacer una página web completa (con su cabecera, su menú, su contenido y su pie de página), podríamos hacerla usando solo y aplicándoles CSS (Flexbox, colores, etc.). El resultado visual para el usuario sería perfecto.

En HTML se utilizan las etiquetas semánticas. No son más que cajas (funcionan exactamente igual que un), pero tienen un nombre específico que le dice al navegador qué tipo de contenido hay en su interior.

Usar estas etiquetas mejora el SEO (Search Engine Optimization, para salir primeros en Google) y la Accesibilidad.

Las etiquetas estructurales principales

Estas son las cajas "con nombre" que debes usar para estructurar el esqueleto de cualquier página web:

- **<header>** (Cabecera): Va en la parte superior. Suele contener el logo de la página y el título principal.
- **<nav>** (Navegación): Contiene el menú principal con los enlaces a otras partes del sitio web. A menudo va dentro o justo debajo del <header>.
- **<main>** (Contenido Principal): Es la caja más importante. Engloba el contenido único y central de esa página. Solo debe haber un <main> por documento.
- **<section>** (Sección): Sirve para agrupar contenido relacionado dentro de la página (por ejemplo, una sección de "Nuestros Servicios" y otra de "Contacto").
- **<article>** (Artículo): Una caja diseñada para contenido que tiene sentido por sí mismo y podría ser publicado de forma independiente (como una noticia de un periódico, una entrada de un blog o la ficha de un producto).

- **<aside>** (Barra lateral): Contenido secundario que acompaña al principal (como banners de publicidad, enlaces relacionados o una biografía corta del autor).
- **<footer>** (Pie de página): Va al final del todo. Suele contener los enlaces legales (Aviso Legal, Cookies), redes sociales y el copyright.

Uso con CSS

Estas etiquetas se comportan como cajas normales, puedes seleccionarlas directamente en tu CSS y aplicarles Flexbox, márgenes, colores o sombras.

Ejemplo:

```
/* 5. Le damos estilo al pie de página sin usar
clases, llamando a la etiqueta directamente */
footer {
  background-color: #2c3e50;
  color: white;
  text-align: center;
  padding: 20px;
}

/* 6. Usamos Flexbox en la navegación para poner los
enlaces en fila */
nav {
  display: flex;
  justify-content: center;
  gap: 15px;
  background-color: #e0e0e0;
}
```

10 Práctica Guiada: Maquetando la estructura de un Blog

La estructura más clásica de Internet es la de un Blog o periódico digital: una cabecera arriba, el contenido principal a un lado, una barra lateral (con enlaces o publicidad) al otro, y el pie de página abajo.

Vamos a aplicar la teoría sobre Flexbox y estructuas semánticas para construir este diseño.

> **El truco**: Como queremos poner la etiqueta <main> y la etiqueta <aside> una al lado de la otra, tendremos que envolverlas en un padre y decirle que sea display: flex;.

Paso 1: El HTML (index.html)

Creamos un sitio web nuevo archivo nuevo y escribimos la estructura:

```
<!doctype html>
<html lang="es">
  <head>
    <meta charset="UTF-8" />
    <title>Mi Primer Blog</title>
    <link
      href="https://fonts.googleapis.com/css2?
family=Roboto:wght@400;700&display=swap"
rel="stylesheet"
    />
    <link rel="stylesheet" href="estilos.css" />
  </head>
  <body>
    <header>
      <h1>Blog de Informática II</h1>
    </header>

    <div class="contenedor-columnas">
      <main>
        <h2>Última noticia: ¡Aprender CSS es
divertido!</h2>
        <p>
          Hoy en clase hemos aprendido a usar Flexbox
para hacer columnas. Ha
sido mucho más fácil de lo que parecía.
        </p>
```

```
        <p>
          Ahora sabemos que las cajas no tienen por
qué estar siempre apiladas
una debajo de otra.
        </p>
      </main>

      <aside>
        <h3>Sobre Mí</h3>
        <p>
          Soy estudiante de 2º de Bachillerato y
estoy creando mi primera web
completa.
        </p>
      </aside>
    </div>
    <footer>
      <p>&copy; 2026 - Creado en IES Grande
Covián</p>
    </footer>
  </body>
</html>
```

Paso 2: El CSS (estilos.css)

Presta especial atención a la propiedad flex que le hemos dado al main y al aside para repartir el espacio disponible.

```
/* 1. ESTILOS GENERALES */
body {
  font-family: 'Roboto', sans-serif;
  background-color: #e9ecef; /* Gris de fondo de la
web */
  margin: 0; /* Quitamos el margen blanco que traen
los navegadores por defecto */
}

/* 2. CABECERA Y PIE (Ocupan todo el ancho) */
header, footer {
  background-color: #2c3e50;
  color: white;
  text-align: center;
  padding: 20px;
}

/* 3. EL PADRE FLEXBOX (Para las columnas centrales)
*/
.contenedor-columnas {
```

```
  display: flex; /* ¡Enciende la magia de las
columnas! */
  gap: 20px; /* Separación entre main y aside */
  max-width: 1000px; /* Ancho máximo para que no se
estire infinito */
  margin: 30px auto; /* Centramos el bloque entero en
la pantalla */
  padding: 0 20px; /* Aire a los lados por si la
pantalla es pequeña */
}

/* 4. LOS HIJOS (Repartiendo el espacio) */
main {
  background-color: white;
  padding: 30px;
  border-radius: 10px;
  box-shadow: 0px 4px 10px rgba(0,0,0,0.1);
  /* TRUCO: Le decimos que ocupe 3 partes del espacio
disponible */
  flex: 3;
}

aside {
  background-color: #f8f9fa;
  padding: 20px;
  border-radius: 10px;
  border: 2px solid #dee2e6; /* Un bordecito gris
para que destaque */
  /* TRUCO: Le decimos que ocupe 1 parte del espacio
disponible */
  flex: 1;
}
```

¿Qué es eso de flex: 3 y flex: 1?

Es una forma facilísima de crear proporciones. Le estamos diciendo al navegador: "Divide el espacio horizontal en 4 partes (3+1). Dale 3 partes (el 75%) al <main> y 1 parte (el 25%) al <aside>". ¡Así de sencillo es hacer una barra lateral profesional!

¡Guarda y visualiza en Live Server!

> Reset de estilos: Hemos introducido el margin: 0; en el body, que es otro truco vital para quitar el molesto borde blanco que dejan los navegadores.

11 Contenido Multimedia: Imágenes

Avmos a aprender a insertar imágenes para ilustrar nuestro blog.

La etiqueta <img> y sus atributos obligatorios

Para insertar una imagen utilizaremos la etiquteta <img>. Esta etiqueta es una etiqueta "vacía" (se cierra a sí misma) que necesita dos atributos obligatorios para funcionar correctamente:

```
<img src="ruta/mi-foto.jpg" alt="Texto descriptor">
```

- **src (Source / Origen):** Indica al navegador dónde está el archivo de la imagen. Puede ser el nombre de un archivo que esté en tu misma web (ej: foto.jpg) o un enlace de Internet (ej: https://www.ejemplo.com/foto.jpg).
- **alt (Alternative Text / Texto Alternativo)**: Descripción de la foto. Nunca dejarla en blanco. Si la imagen no carga por mala conexión, aparecerá este texto. Además, las personas invidentes usan programas que leen este texto en voz alta, y a Google le encanta para saber de qué trata tu página (mejorando el SEO).

CSS: Imágenes Responsivas

Las imágenes en HTML tienen un "peligro": si insertamos una fotografñua de altísima resolución, la imagen intentará mostrarse a su tamaño real, rompiendo todas las cajas y columnas que has creado con Flexbox.

Para solucionar esto, incluiremos siempre la siguiente regla en nuestro archivo CSS:

```
/* Hacemos que todas las imágenes sean fluidas y no
rompan las cajas */
img {
    max-width: 100%;
    height: auto;
}
```

¿Qué hace exactamente esto?

- **max-width: 100%;** Con ello consegimos que la imagen unca seas más ancha que la caja en la que está contenida.
- **height: auto;** Hace que cuando por acción de la regla anterior se reduzca la anchura de la imagen, la altura se calculará tu automáticamente para no deformarse.

Diseño avanzado: Recortes perfectos con object-fit

A veces queremos que todas las fotos de nuestro blog tengan el mismo tamaño exacto (por ejemplo, cuadrados perfectos de 200px por 200px), pero las fotos originales tienen proporciones diferentes. Si forzamos el ancho y el alto en CSS, la foto se verá estirada y deforme.

Para evitar que la foto se deforme, usamos la propiedad object-fit: cover;. Esta regla recorta lo que sobra de la imagen para que encaje perfectamente en la caja sin aplastarse.

/* Ejemplo de foto de perfil perfectamente redonda y sin deformar */

```
.foto-perfil {
  width: 150px;
  height: 150px;
  border-radius: 50%; /* 50% hace un círculo perfecto
  */
  object-fit: cover; /* Recorta la imagen para que
  encaje en el círculo sin deformarse */
}
```

11.1 Práctica: Añadiendo imágenes a nuestro Blog

Vamos a actualizar el Blog que hicimos en el punto anterior. Busca y guarda dos imagenes para tu web

> **Buenas prácticas**: Es buena idea ser organizado. Agrupa todas tus imágenes en una carpeta específica (imagenes).

1. En tu archivo HTML:

Vamos a añadir una imagen principal dentro de nuestra etiqueta <main> (justo debajo del <h2>) y una foto de perfil en nuestro <aside>.

```
<main>
  <h2>Última noticia: ¡Aprender CSS es
divertido!</h2>
  <img class="foto-articulo"
src="PEGAR_AQUI_EL_ENLACE_DE_LA_FOTO" alt="Alumnos
programando en clase">
  <p>Hoy en clase hemos aprendido a usar
Flexbox...</p>
</main>
<aside>
  <h3>Sobre Mí</h3>
  <img class="foto-perfil"
src="PEGAR_AQUI_OTRO_ENLACE" alt="Mi foto de perfil">
  <p>Soy estudiante de 2° de Bachillerato...</p>
</aside>
```

2. En tu archivo CSS:

Añadimos estas reglas al final de tu archivo estilos.css para darle un aspecto profesional a las imágenes.

```
/* Regla para TODAS las imágenes de la web */
img {
  max-width: 100%;
  height: auto;
}

/* Estilo específico para la foto del artículo */
.foto-articulo {
  border-radius: 8px; /* Bordes redondeados */
  margin-bottom: 15px; /* Separamos la foto del texto
  que va debajo */
}

/* Estilo para la foto de perfil en la barra lateral
*/
.foto-perfil {
  width: 100px;
  height: 100px;
  border-radius: 50%; /* Círculo perfecto */
  object-fit: cover; /* Evita que se aplaste si la
  foto no era cuadrada */
  display: block; /* Nos permite centrarla con margin
  auto */
```

```
  margin: 0 auto 15px auto; /* Centrada y con margen
  por debajo */
  border: 3px solid #2c3e50; /* Un borde decorativo
  */
}
```

Guardamos y visualizamos en el navegador.

Nuestro blog parece un sitio web profesional de verdad, con fotografías perfectamente integradas y adaptadas a sus cajas contenedoras.

12 Interactividad: Creando un Menú de Navegación Profesional

En el Punto 5 aprendimos a crear hipervínculos con la etiqueta <a> para viajar entre las distintas páginas de nuestro sitio web. Sin embargo, recordarás que visualmente eran muy básicos: texto de color azul y subrayado.

En una web profesional, los enlaces de la barra de navegación no tienen ese aspecto; parecen verdaderos botones que cambian de color al pasar el ratón. En este apartado vamos a coger esos enlaces que ya sabes hacer, los agruparemos y les daremos vida con CSS.

Agrupando elementos: Las Listas (<ul> y <li>)

Cuando tenemos un grupo de enlaces que forman un menú, la regla semántica dice que no debemos poner las etiquetas <a> sueltas. Debemos agruparlas en una lista.

Usaremos las listas desordenadas <ul> (Unordered List), que por defecto ponen un "puntito" (viñeta) delante de cada elemento de la lista <li> (List Item).

```
<ul>
  <li><a href="index.html">Inicio</a></li>
  <li><a href="sobre-mi.html">Sobre Mí</a></li>
  <li><a href="contacto.html">Contacto</a></li>
</ul>
```

Limpiando los estilos por defecto con CSS

Si ponemos el código anterior en nuestra web, se verá como una lista típica de un documento de texto. Para convertirlo en un menú, primero debemos "limpiar" los estilos que el navegador pone por defecto:

Para la lista (<ul>): Usaremos list-style: none; para borrar los puntos (viñetas) y le quitaremos el margen interior que trae de serie.

Para los enlaces (<a>): Usaremos text-decoration: none; para eliminar ese molesto subrayado azul, y le daremos el color de texto que nosotros queramos.

La Magia de la Pseudoclase :hover

En CSS, podemos cambiar el estilo de un elemento solo cuando ocurre una acción. A esto se le llama pseudoclase. La más utilizada es :hover, que se activa justo cuando el usuario pone el cursor del ratón encima del elemento.

```
/* El enlace es blanco normalmente... */
a {
  color: white;
}

/* ...pero se vuelve amarillo al pasar el ratón por
encima */
a:hover {
  color: yellow;
}
```

12.1 Práctica: Construyendo nuestro Menú Superior

Vamos a añadir un menú dentro de la cabecera (<header>) de nuestro Blog.

Recuerda que la caja semántica correcta para envolver un menú es <nav>.

Paso 1: El HTML (index.html)

Busca tu etiqueta <header> y añade el <nav> con la lista de enlaces justo debajo del título <h1>:

```
<header>
  <h1>Blog de Informática II</h1>
  <nav>
    <ul class="menu-principal">
      <li><a href="index.html">Inicio</a></li>
      <li><a href="#">Artículos</a></li>
      <li><a href="#">Sobre Mí</a></li>
      <li><a href="#">Contacto</a></li>
    </ul>
  </nav>
</header>
```

Paso 2: El CSS (estilos.css)

Añade este bloque a tu archivo CSS. Fíjate en cómo usamos Flexbox en la lista para poner los elementos en horizontal, ¡Flexbox sirve para todo!

```
/* 1. Limpiamos la lista y la ponemos en horizontal
con Flexbox */
.menu-principal {
  list-style: none; /* ¡Adiós a los puntitos! */
  padding: 0; /* Quitamos la sangría por defecto */
  margin: 20px 0 0 0; /* Espacio para separarlo del
  título h1 */
  display: flex; /* Pone los li en fila horizontal */
  justify-content: center; /* Centra los botones en
  la pantalla */
  gap: 15px; /* Separación entre los botones */
}

/* 2. Damos estilo de "botón" a los enlaces */
.menu-principal a {
  text-decoration: none; /* ¡Adiós al subrayado! */
  color: white; /* Letra blanca */
  background-color: #34495e; /* Un azul un poco más
  claro que la cabecera */
  padding: 10px 20px; /* Hacemos el botón grande y
  clicable */
  border-radius: 5px; /* Bordes redondeados */
  font-weight: bold; /* Letra en negrita */
  transition: 0.3s; /* Magia: hace que el cambio de
  color sea suave y animado */
}

/* 3. La interacción al pasar el ratón (HOVER) */
.menu-principal a:hover {
background-color: #f1c40f; /* El fondo se vuelve
  amarillo al pasar el ratón */
  color: #2c3e50; /* La letra oscura para que
  contraste bien */
}
```

¡Guarda y visualiza en tu navegador!

Pasa el ratón por los botones del menú. Verás cómo cambian de color de forma suave (gracias a transition: 0.3s;).

13 Organizando datos: Tablas en HTML

En los inicios de Internet, las tablas se usaban para todo, ¡incluso para hacer columnas y menús! Hoy en día, gracias a herramientas como Flexbox, sabemos que eso es un error. Las tablas solo deben usarse para mostrar datos tabulares (horarios, clasificaciones, listas de precios, etc.).

Por defecto, el navegador dibuja las tablas de una forma muy anticuada, pero con HTML5 y un poco de CSS, podemos crear tablas modernas y fáciles de leer.

Anatomía de una Tabla HTML

Las tablas tienen una estructura jerárquica muy estricta. Una tabla es una cuadrícula donde primero creamos las filas horizontales y, dentro de cada fila, vamos metiendo las celdas.

Sus componentes semánticos básicos son:

- **<table>**: Es la caja principal que envuelve toda la tabla.
- **<tr>** (Table Row): Define una fila horizontal.
- **<th>** (Table Header): Define una celda de encabezado (suele ir en la primera fila y el texto sale en negrita).
- **<td>** (Table Data): Define una celda de datos normal.

13.1 Práctica: Nuestro Horario de Clases

Vamos a crear una tabla sencilla para mostrar los horarios de nuestras asignaturas favoritas y luego le daremos un diseño profesional con CSS.

Paso 1: El HTML

Crea una nueva página web (por ejemplo, horario.html) y añade este código en la zona principal de tu proyecto:

```
<table class="tabla-moderna">
  <tr>
```

```
    <th>Día</th>
    <th>Asignatura</th>
    <th>Aula</th>
  </tr>
  <tr>
    <td>Lunes</td>
    <td>Informática II</td>
    <td>Aula de Informática 1</td>
  </tr>
  <tr>
    <td>Miércoles</td>
    <td>Matemáticas</td>
    <td>Aula 204</td>
  </tr>
  <tr>
    <td>Viernes</td>
    <td>Informática II</td>
    <td>Aula de Informática 1</td>
  </tr>
</table>
```

Si guardamos y miramos esto en el navegador... veremos que los datos están ahí, pero se ven apelotonados y sin líneas. ¡Es hora de usar CSS!

Paso 2: El CSS

Añade estas reglas a tu archivo estilos.css. Vamos a introducir una propiedad nueva llamada border-collapse que es fundamental para que las tablas no tengan bordes dobles feos.

```
/* 1. Estilos generales de la tabla */
.tabla-moderna {
  width: 100%; /* La tabla ocupará todo el ancho
  disponible */
  max-width: 600px; /* Pero nunca será más ancha de
  600px */
  margin: 30px auto; /* Centramos en la pantalla */
  border-collapse: collapse; /* ¡CLAVE! Fusiona los
  bordes dobles en uno solo */
  background-color: white;
  box-shadow: 0px 4px 10px rgba(0,0,0,0.1); /* Sombra
  elegante */
}

/* 2. Estilos comunes para celdas y encabezados */
.tabla-moderna th, .tabla-moderna td {
```

```
  padding: 15px; /* Aire por dentro para que respiren
  los datos */
  text-align: left; /* Alineamos el texto a la
  izquierda */
  border-bottom: 1px solid #dddddd; /* Una línea
  sutil debajo de cada fila */
}

/* 3. Estilos específicos para el encabezado (th) */
.tabla-moderna th {
  background-color: #2c3e50; /* Azul oscuro */
  color: white;
}
```

Paso 3: CSS: Tablas "Cebra" y Efecto Hover

Para que una tabla con muchos datos sea fácil de leer, los profesionales usan dos trucos visuales: pintar las filas de colores alternos (efecto cebra) y hacer que la fila se ilumine al pasar el ratón.

Añade esto al final de tu CSS:

```
/* EFECTO CEBRA: Pinta de gris muy claro solo las
filas PARES (even) */
.tabla-moderna tr:nth-child(even) {
background-color: #f8f9fa;
}

/* EFECTO HOVER: Ilumina la fila por la que pasa el
ratón */
.tabla-moderna tr:hover {
background-color: #e2e6ea;
}
```

Guardamos y visualizamos en el navegador.

Pasa el ratón por encima de las filas de tu horario. Fíjate en cómo la pseudoclase **:nth-child(even)** hace todo el trabajo por nosotros, pintando filas alternas sin que tú tengas que ir una a una poniéndoles clases. ¡Puro diseño profesional!

14 El Selector Universal y el CSS Reset

A medida que nuestro sitio web crece, observaremos que los navegadores son un poco "entrometidos". Por defecto, añaden márgenes invisibles a la página, espacios a las listas o alteran el tamaño de las cajas.

Si diseñamos una caja que mida 300px de ancho, pero luego le ponemos un padding (margen interior) de 20px, el navegador suma los valores y tu caja de repente mide 340px. ¡Esto destroza cualquier diseño con Flexbox!

Para evitar que nuestras columnas se rompan, pondremos un código específico una en la primera línea de nuestro archivo CSS.

La Vacuna CSS: El box-sizing

Para aplicar esta "vacuna" a todas las etiquetas de nuestra web a la vez, usamos el **Selector Universal**, que se representa con un asterisco (*). Lo que pongamos dentro del asterisco, se aplicará absolutamente a todo.

Abre tu archivo estilos.css, ve a la línea 1 y pega el código:

```
/* EL RESETEO PROFESIONAL (La Vacuna) */
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}
```

¿Qué hace exactamente este código?

- **margin:** 0; y padding: 0;: Borra todos los espacios invisibles que los navegadores ponen por defecto. A partir de ahora, si hay un espacio en nuestra web, es porque hemos decidido ponerlo.
- **box-sizing:** border-box;: Le dice al navegador: "Si te digo que una caja mide 300px, mide 300px. Si le pongo padding o bordes, mételos hacia adentro, pero nunca, bajo ningún concepto, agrandes el tamaño total de la caja".

¡Atención! El efecto secundario del Reset

Al aplicar el reseteo universal con el asterisco (*), hemos hecho que el navegador elimine todos los márgenes y rellenos por defecto.

Esto es genial para que nuestra estructura y nuestras columnas con Flexbox no se rompan, pero tiene un efecto secundario: también hemos borrado los márgenes que sí eran útiles.

Si miras tu página ahora mismo, notarás que los títulos (<h1>, <h2>) y los párrafos de texto (<p>) están totalmente pegados unos a otros. El texto ya no "respira".

¿Cuál es la solución?

Ahora que nosotros tenemos el control total de la web, somos nosotros quienes debemos decidir ese espaciado.

Cuando apliques un Reset CSS, debemos dar un repaso a la hoja de estilos y añadir márgenes inferiores (margin-bottom) a los textos para devolverles su espacio natural.

Por ejemplo:

```
/* Devolvemos el "aire" a los textos tras aplicar el
Reset */
h1, h2, h3 {
  margin-bottom: 15px;
}

p {
  margin-bottom: 10px;
}
```

Con este pequeño repaso, la web volverá a ser fácil de leer, pero esta vez con las medidas exactas que hayamos decidido.

15 Diseño responsive (la web en el móvil)

Hasta ahora hemos construido una página web espectacular: tenemos nuestro menú alineado, un blog con su artículo principal a la izquierda y su barra lateral a la derecha. En la pantalla del ordenador se ve perfecta.

Pero, ¿qué pasa si abrimos esa misma página web en un teléfono móvil? Como la pantalla del móvil es muy estrecha, el navegador intentará encoger las dos columnas (el <main> y el <aside>) para que quepan una al lado de la otra. El resultado: el texto se verá minúsculo, las fotos se aplastarán y la web será imposible de leer.

La solución a esto es el Diseño Responsive (Diseño Adaptativo). Consiste en darle unas instrucciones especiales al archivo CSS para conseguir que en función de cual sea la resolución de nuestro dispositivo, el resultado se presente con formatos diferentes.

El condicional de CSS: Las Media Queries (@media)

Para hacer esto, usamos una regla especial llamada Media Query. Funciona como un interruptor automático. Le decimos al navegador a partir de qué tamaño de pantalla queremos que se activen unos estilos nuevos.

La sintaxis es muy sencilla y siempre se coloca al final del todo en nuestro archivo estilos.css:

```
@media (max-width: 768px) {
  /* Todo el CSS que escribamos aquí dentro SOLO
  funcionará si la pantalla mide 768 píxeles de ancho
  o menos (tamaño tablet/móvil) */
}
```

El requisito invisible: La etiqueta <meta viewport>

Antes de escribir el CSS para adaptar nuestra web, necesitamos preparar nuestro archivo HTML.

Si abrimos una página web en un móvil y no damos ninguna instrucción especial, el navegador del teléfono pensará: "Esta web seguro que está hecha

para un monitor gigante, así que voy a alejar el zoom al máximo para que quepa todo en mi pantallita".

¿El resultado? La web se ve entera, pero los textos y las cajas son microscópicos. Además, como el móvil está "haciendo trampa" con el zoom, nuestras Media Queries de CSS no funcionarán.

Para solucionar esto y darle permiso al móvil para usar nuestro diseño adaptativo, es obligatorio incluir esta línea de código dentro de la etiqueta <head> de nuestro documento HTML:

```
<meta name="viewport" content="width=device-width,
initial-scale=1.0">
```

¿Qué significa este conjuro?

- **viewport:** Es la "ventana" a través de la cual el usuario ve la página web (la pantalla).
- **width=device-width:** Le da una orden estricta al móvil: "No inventes tamaños raros. El ancho de esta web debe ser exactamente el ancho físico de tu pantalla".
- **initial-scale=1.0:** Le dice que el nivel de zoom inicial debe ser el normal (100%), sin alejar ni acercar la cámara.

15.1 Práctica: Nuestro Blog Responsive

Vamos a arreglar nuestro Blog. El problema principal en móviles son las cosas que están puestas "en horizontal" (como nuestro menú superior y nuestras columnas del blog). Lo que queremos es que, en el móvil, todo se apile en vertical, una cosa debajo de la otra.

Como usamos Flexbox para hacer las columnas, ¡cambiarlas a vertical es cuestión de una sola línea de código!

Ve al final de tu archivo estilos.css y añade este bloque:

```
/* ========================================
DISEÑO PARA MÓVILES (Pantallas pequeñas)
```

```
========================================== */

@media (max-width: 768px) {
  /* 1. Arreglamos las columnas del Blog */
  .contenedor-columnas {
    flex-direction: column; /* ¡Magia! Cambia la
    dirección de Flexbox de fila a columna */
    padding: 10px; /* Reducimos un poco el aire
    lateral para aprovechar la pantalla */
}

  /* 2. Arreglamos el menú de navegación */
  .menu-principal {
    flex-direction: column; /* Ponemos los botones
    del menú uno debajo del otro */
    align-items: center; /* Los centramos */
    gap: 5px; /* Reducimos el espacio entre ellos */
  }
}
```

Guarda tu archivo CSS, abre tu web en el ordenador y... ¡haz la ventana del navegador más estrecha arrastrando desde un borde! Verás que, al llegar a cierto punto, las columnas "saltan" mágicamente y se colocan una debajo de la otra. ¡Acabas de programar tu primera web profesional adaptativa!

16 Más posibilidades CSS

Con todo lo que hemos visto hasta este punto (cajas, Flexbox, colores, tipografías, imágenes, listas, tablas y diseño responsive) ya tenemos los conocimientos necesarios para crear sitios web avanzados.

Sin embargo, el mundo del diseño web es inmenso y siempre hay nuevas herramientas por descubrir. Si eres de los que disfrutan con el diseño y quieres llevar tu proyecto un paso más allá, hay dos conceptos avanzados que los profesionales utilizan muchísimo:

A. CSS Grid: El hermano mayor de Flexbox

Mientras que Flexbox es perfecto para alinear cosas en una sola dirección (una fila o una columna), CSS Grid está diseñado para trabajar en dos dimensiones a la vez.

Te permite crear cuadrículas complejas (como el diseño de una revista digital o el panel de control de una aplicación) definiendo filas y columnas exactas y diciéndole a cada caja qué celdas debe ocupar.

B. Menús Desplegables (Dropdowns)

En muchos menús al pasar el ratón por encima de uno de sus botones (por ejemplo, "Servicios"), se despliega hacia abajo un submenú con más opciones.

Para lograr esto con CSS, se utilizan propiedades avanzadas de posicionamiento (position: relative y position: absolute) que permiten "sacar" una caja de su flujo normal y hacer que flote por encima de los demás elementos de la página.

¿Aceptas el reto?

Estos dos temas son un poco más complejos, por lo que forman parte del temario obligatorio de este bloque. Pero si quieres aplicarlos en tu web, tienes a tu disposición dos Anexos independientes en la página web de la asignatura con la teoría paso a paso y ejemplos prácticos para que puedas copiarlos, entenderlos y adaptarlos a tu proyecto.

17 Anexo: Unidades medida CSS

En CSS existen distintas formas de definir tamaños, espacios y proporciones. Estas unidades pueden ser **absolutas** o **relativas**, y cada una tiene su utilidad según el contexto.

Unidades absolutas

Estas unidades siempre representan el mismo tamaño, sin importar el entorno (pantalla, zoom, etc.).

Unidad	Significado	Ejemplo
px	píxeles (unidad más común)	padding: 20px;
cm	centímetros	width: 5cm;
mm	milímetros	width: 50mm;
in	pulgadas (1in = 2.54 cm)	width: 2in;
pt	puntos (1pt = 1/72 de una pulgada)	font-size: 12pt;
pc	picas (1pc = 12pt)	font-size: 1pc;

Unidades relativas

Dependen del contexto, como el tamaño de fuente del elemento padre o del html.

Unidad	Basada en...	Ejemplo	Uso típico
%	Porcentaje del tamaño del contenedor	width: 50%;	Layouts fluidos
em	Tamaño de fuente del elemento padre	margin: 2em;	Espaciado relacionado con el texto
rem	Tamaño de fuente del elemento raíz (html)	padding: 1rem;	Más predecible que em
vw	1% del ancho de la ventana	width: 50vw;	Diseños adaptables horizontalmente
vh	1% del alto de la ventana	height: 100vh;	Altura de secciones en pantalla
vmin	1% del lado más pequeño del	font-size: 5vmin;	Tipografía responsiva

	viewport		
vmax	1% del lado más grande del viewport	font-size: 5vmax;	Tipografía responsiva

Recomendaciones prácticas

- Usa rem para márgenes, paddings y fuentes: garantiza coherencia global.
- Usa px cuando necesites precisión absoluta (por ejemplo, en bordes o íconos).
- Usa %, vh y vw para diseños fluidos o responsivos.
- Evita pt, pc, in, cm y mm en pantalla: se usan más en documentos impresos.

18 Anexo II - Mi servidor en byet

000webhost es una empresa proveedora de alojamiento y dominios web que ofrece la posibilidad de alojar un sitio web sencillo en sus servidores.

Si no dispones de otra posibilidad de alojamiento, esta opción te permitirá de forma sencilla comprobar el funcionamiento de tu sitio web en la red.

Para crear una cuenta en 000webhost debes visitar su sitio web a través del enlace:

https://byet.host/

Haz clic en el botón "Get Free Hosting Now". Introduce tus datos en la siguiente página.

Sub Domain Name hace referencia a la palabra que estará asociada a la dirección de tu sitio web en los servidores de byet

Una vez que envíes tus datos a byet, recibirás un correo de confirmación en la cuenta que hayas utilizado para registrarte. ***Es posible que el correo se encuentre en la bandeja de spam***. Haz clic en el vínculo para activar tu cuenta.

Serás redirigido a una página en la que aparecerán tus datos de acceso:

Haz una captura de pantalla de estos datos. Los necesitarás más adelante:

Para poder ***acceder a tu cpanel*** debes utilizar la dirección que encontrarás en tu hoja de datos como Control panel URL.

Ahora debes introducir tus datos de usuario (en la misma hoja de control) y la contraseña del control panel (la misma que hayas utilizado para crear tu cuenta de usuario en byethost).

Por último tenemos que encontrar nuestros datos de acceso vía FTP (para acceder al espacio reservado para nuestro sitio en los servidores de byethost).

Esta información estará disponible en la columna de la derecha, dentro de la sección Account Details. La información que necesitarás es:

Main Domain:	*******
FTP hostname:	******
FTP username:	*******
Protocolo conexión	ftp

Puerto	21

Please take a note of the details below these are needed to access your account.
Your account is now active.

Control panel username:	[redacted]
Control panel password:	**********
Control panel URL:	[redacted]
MySQL username	
MySQL password:	**********
MySQL hostname:	[redacted]
FTP username:	
FTP password:	**********
FTP host name:	[redacted]
Your Website URL:	http://info2.byethost22.com

Thank you for choosing us to host your websites!

byethost22.com hosting services 2026

Como contraseña para la conexión vía FTP utilizarás la misma contraseña que utilizas para acceder a tu cuenta de usuario en byethost.

Debes guardar tu sitio dentro de la carpeta htdocs.

Sumario