



# Diseño de Interfaces Web

## 2º DAW

### Unidad de Trabajo 2

# Creación de interfaces web utilizando estilos



**Junta de Andalucía**  
Consejería de Educación y Deporte

# Secuenciación del módulo

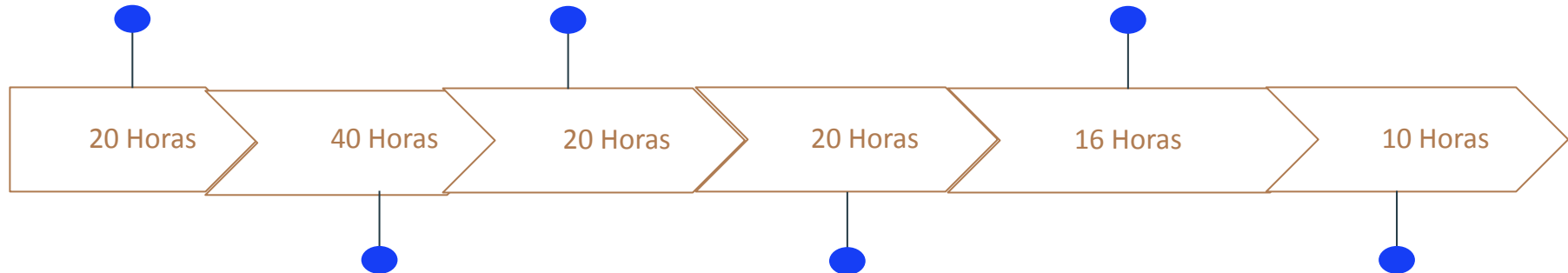


Total: 126 horas

UT1: Planificación de  
Interfaces Gráficas

UT3: Implantación de  
contenido multimedia

UT5: Desarrollo de Webs  
Accesibles



**UT2: Creación de interfaces  
web utilizando estilos**

UT4: Integración de  
contenido interactivo

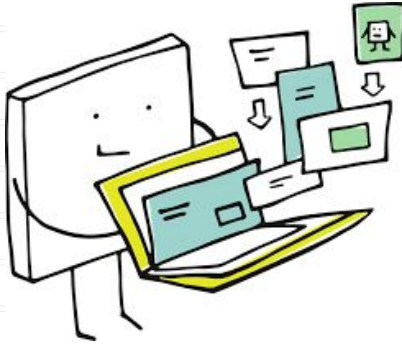
UT6: Desarrollo de  
interfaces Web amigables

# ¿Cómo se va a evaluar?



## ▼ Unidad 2: Creación de interfaces web utilizando estilos

- Evaluación inicial
- Ejercicio 1: Insertar código CSS
- Ejercicio 4, 5 y 6: selectores básicos
- Ejercicio 7 y 8: selectores combinadores
- Ejercicio 9: Unidades de medida
- Ejercicio 10 y 11: Color y fondo
- Ejercicio 12, 13 y 14: Texto y fuentes
- Ejercicio 15: listras



### Portafolio:

- Ejercicios
- Teoría

Defenderlo ante el profesor



### Proyecto:

- Grupal

Exposición



### Examen:

- Test

Autoevaluación

- **Cada tema:**
  - Introducción
  - Desarrollo → Ejercicios de clase
  - Consolidación → Evaluación
- **Cada sesión:**
  - Recordatorio de la clase anterior y objetivos de hoy
  - 5 min de descanso entre horas
- **Cada punto del tema:** Explicaciones - Ejemplos - Ejercicios/Actividades
- **Trabajo de los alumnos:**
  - Autónomo y colaborativo
  - Práctico más que teórico

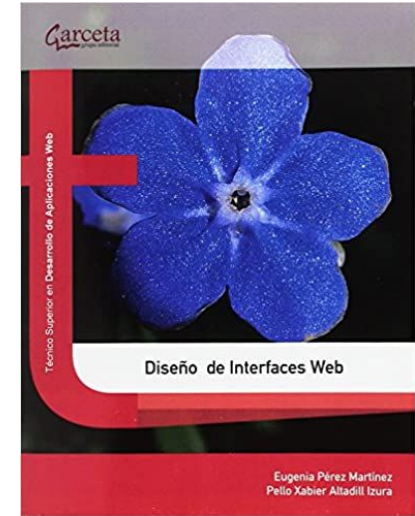
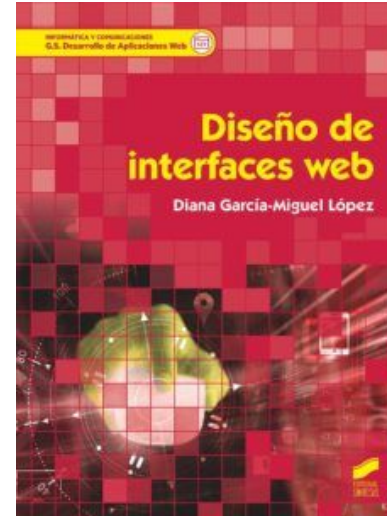


## Bibliografía

- Ed. SINTESIS
- Ed. Garceta
- Ed. Ra-Ma

## Material de clase

- Diapositivas
- Documentación suministrada por el profesor
- Referencias de ampliación y refuerzo
- Actividades y **apuntes**
- Glosario



# ¿Qué vamos a aprender?



2º DAW  
DIWEB

UD 2  
Creación de interfaces web  
utilizando estilos



## Introducción a CSS

- Concepto y evolución de CSS

- Formas de aplicar estilo a un documento HTML

## Lenguaje CSS

- Sintaxis

- Orden en cascada

- Selectores

- Especificidad

- Propiedades

## Modelo de cajas

## Herencia

## Pseudoclases y pseudoelementos

## CSS Layout

- Display

- Max-width

- Position

- Superposición (z-index)

- Overflow

- Flotación

- Alineación

- Box-sizing y resize

## Medios

## Herramientas CSS

## Buenas prácticas

## CSS 3

## Bootstrap

## CSS Avanzado

## YUI

## SASS

# Objetivos



- Definir estilos de forma directa.
- Asociar estilos globales en hojas externas.
- Identificar las distintas propiedades de cada elemento.
- Crear clases de estilos.
- Analizar las diferentes opciones para la presentación de la información en documentos web.
- Valorar la importancia de definir y aplicar una guía de estilo en el desarrollo de una aplicación web.
- Seleccionar colores y otros elementos de diseño para su visualización en pantalla.
- Crear y utilizar plantillas de diseño.
- Reconocer la posibilidad de modificar las etiquetas HTML.



# ¿Qué se va a evaluar?




Resultado de Aprendizaje	% módulo	Criterio de Evaluación	% módulo
RA 2: Crea interfaces Web homogéneos definiendo y aplicando estilos.	47%	a) Se han reconocido las posibilidades de modificar las etiquetas HTML.	3%
		b) Se han definido estilos de forma directa.	3%
		c) Se han definido y asociado estilos globales en hojas externas.	3%
		d) Se han definido hojas de estilos alternativas.	6%
		e) Se han redefinido estilos.	3%
		f) Se han identificado las distintas propiedades de cada elemento.	3%
		g) Se han creado clases de estilos.	3%
		h) Se han utilizado herramientas de validación de hojas de estilos.	3%
		i) Se ha utilizado y actualizado la guía de estilo.	3%

## Your Wooclap question will appear here

1

Install the **Chrome** or **Firefox** extension 

2

Click on “Add a Wooclap vote” and make sure you are logged into your Wooclap account 

3

Ensure you are in **presentation mode** on Google Slides 



# Evaluación Conocimientos Previos

Quiz w3school CSS

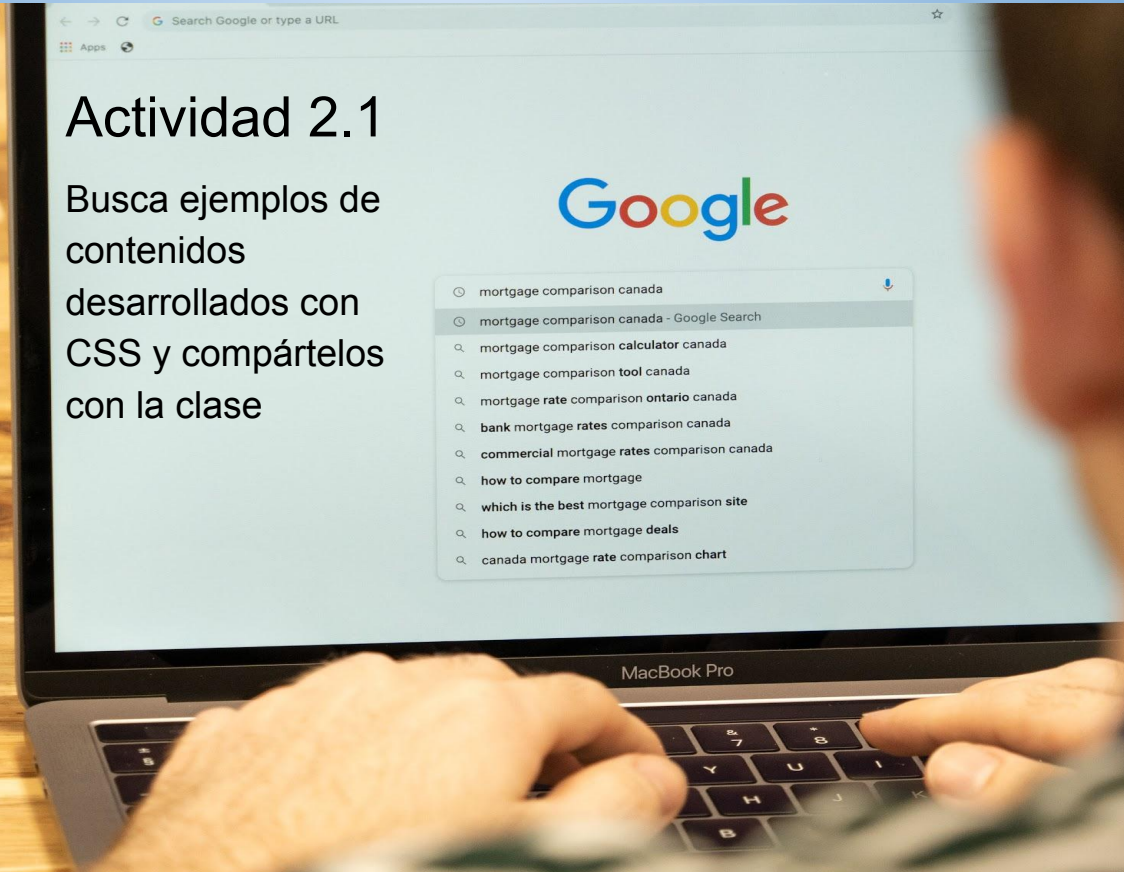
[https://www.w3schools.com/css/css\\_quiz.asp](https://www.w3schools.com/css/css_quiz.asp)

# Explora cómo lo hacen los demás



## Actividad 2.1

Busca ejemplos de  
contenidos  
desarrollados con  
CSS y compártelos  
con la clase




## Your Wooclap question will appear here

1

Install the **Chrome** or **Firefox** extension 

2

Click on “Add a Wooclap vote” and make sure you are logged into your Wooclap account 

3


Ensure you are in **presentation mode** on Google Slides 

## Your Wooclap question will appear here

1

Install the **Chrome** or **Firefox** extension 

2

Click on “Add a Wooclap vote” and make sure you are logged into your Wooclap account 

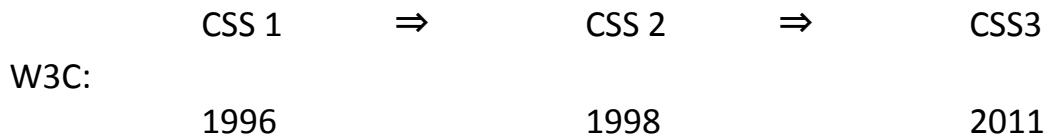
3

Ensure you are in **presentation mode** on Google Slides 

# Introducción: ¿Qué es CSS?



- Cascading Style Sheets (Hojas de estilo en cascada)
- Sirve para dar estilo al documento HTML ⇒ Cómo se va a ver el HTML
- Permite separar los estilos del contenido
  - Reutilización ⇒ Aplicar el mismo estilo a todas las páginas
  - Mantenimiento ⇒ Si tengo que modificar algo solo lo hago en un sitio
  - Simplificación ⇒ Archivos menos cargados de código
- Aplicar diferentes estilos a un mismo documento: [ejemplo](#)
- [Estándar W3C](#)
- También aporta:
  - Acelerar el desarrollo
  - Diseño para varios dispositivos: media queries



¿Qué diferencias encuentras entre cada versión?




## Your Wooclap question will appear here

1

Install the **Chrome** or **Firefox** extension 

2

Click on “Add a Wooclap vote” and make sure you are logged into your Wooclap account 

3

Ensure you are in **presentation mode** on Google Slides 

# Introducción: Historia



Nivel	Año	Descripción
CSS1	1996	Se crean propiedades para tipografías, colores, alineación, etc...
CSS2	1998	Propiedades de posicionamiento, tipos de medios, etc...
CSS2.1	2005	Corrige errores de CSS2 y modifica ciertas propiedades
CSS3	2011	Se comienzan a crear características de CSS como módulos independientes

# Lenguaje CSS: ¿Cómo aplicamos CSS?



- En línea:

```
<p style="text-align:left">This is some text in a paragraph.</p>
```

- Interna:

```
<head>
...
<style>
  body {
    background-color: aquamarine;
  }
</style>
</head>
```

- Hoja de estilo externa (con link o con @import):

- ```
<head><link rel="stylesheet" href="nombre.css" ></head>
```
- ```
<style>@import url ("estilo.css")screen</style>
```


[HTML link tag](#)

## Your Wooclap question will appear here

1

Install the **Chrome** or **Firefox** extension 

2

Click on “Add a Wooclap vote” and make sure you are logged into your Wooclap account 

3

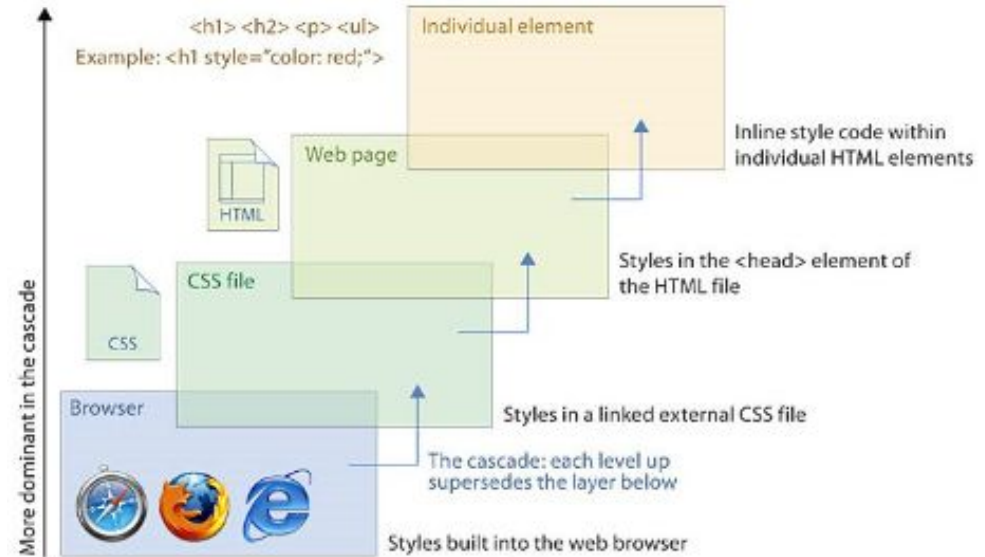
Ensure you are in **presentation mode** on Google Slides 

# Lenguaje CSS: Orden en cascada



Si aplicamos CSS de las 3 formas distintas, un elemento HTML de la página **¿cómo sabe qué estilo le aplica?**

1. Estilo en línea (dentro de un elemento HTML)
2. Hojas de estilo externas e internas (en la sección head)
3. Predeterminado del navegador



## Ejercicio 2.2

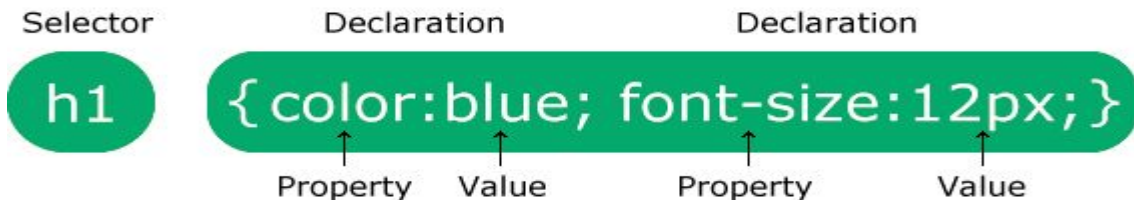
Crea un documento HTML que contenga un sólo párrafo con algún contenido de texto a tu elección.

Aplica diferentes colores de fondo en tres capas distintas:

1. En el head:
  - a. Interna
  - b. Externa
2. Cambia el orden de los anteriores y observa qué ocurre
3. En línea

## ¿Cómo indicamos los estilos de los elementos HTML?

- Selector ⇒ Partes del documento a las que afecta la regla
- Regla ⇒ Cómo se verán los elementos seleccionados por el selector
  - Propiedad : Valor;



### Notas:


- En general No sensible a mayúsculas/minúsculas
- El “;” de la última propiedad se puede omitir
- Comentarios: /\* This is a CSS comment \*/

## Your Wooclap question will appear here

1

Install the **Chrome** or **Firefox** extension 

2

Click on “Add a Wooclap vote” and make sure you are logged into your Wooclap account 

3

Ensure you are in **presentation mode** on Google Slides 



## Ejercicio 2.3

Crea un documento Ejercicio3.html con un encabezado de nivel 1 y un párrafo. Dale un contenido a cada uno y luego aplícales un estilo utilizando CSS “en línea” para representarlos como se indica:

- El encabezado en verde
- El párrafo en rojo.

## Ejercicio 2.4

Haz una copia del fichero del ejercicio anterior con nombre Ejercicio4.html en el que cambiarás la forma de incluir código CSS para hacerlo con las etiquetas `<style>` de manera interna.

¿Qué ventaja encuentras en hacerlo de esta manera? Coméntalo dentro de las etiquetas `<style>` utilizando comentarios CSS.

## Ejercicio 2.5

Haz una copia del fichero del ejercicio anterior con nombre Ejercicio5.html en el que cambiarás la forma de incluir código CSS para hacerlo usando un fichero CSS externo.

¿Qué ventaja encuentras en hacerlo de esta manera con respecto a la del ejercicio 2? Coméntalo dentro del html (justo encima de la importación) y en el código del CSS utilizando comentarios CSS.

Selector de:

- Elemento  $\Rightarrow$  p, h1, ...
- Clase  $\Rightarrow$  .clase
- Identificador  $\Rightarrow$  #id
- Universal  $\Rightarrow$  \*
- Atributo  $\Rightarrow$  [atributo]

```
p{  
  
    propiedad:valor;  
  
}
```

```
#identificador{  
  
    propiedad:valor;  
  
}  
  
<p id="identificador"></p>  
  
<p id="identificador2"></p>
```

```
.clase{  
  
    propiedad:valor;  
  
}  
  
<p class="clase"></p>  
  
<p class="clase2"></p>
```

```
[atributo = "valor"] { propiedad:valor}  
  
<tag atributo="valor"></tag>
```

## Ejercicio 6

1. Crea un archivo **HTML** llamado `index.html`. → <https://codeshare.io/MkZXde>
2. Crea un archivo **CSS** llamado `styles.css`.
3. Conecta el archivo `styles.css` a `index.html` usando la etiqueta `<link>` en el `<head>`.

En el archivo `styles.css`, agrega el código para aplicar estilos usando los diferentes selectores:

1. **Selector de Elemento:** Cambia el color del texto de todos los párrafos a azul.
2. **Selector de Clase:** Agrega un fondo amarillo al párrafo con la clase `importante`.
3. **Selector de ID:** Cambia el color del texto y el tamaño de fuente (*font-size*) del título con `id="titulo"`
4. **Selector Universal:** Elimina los márgenes (*margin*) y el relleno (*padding*) de todos los elementos.
5. **Selector de Atributo:** Aplica un borde rojo al campo de entrada de texto (`input`) que tenga `type="text"`. (*border: 2px solid red;*)

Elemento ⇒ `p, h1, ...`

Clase ⇒ `.clase`

Identificador ⇒ `#id`

Universal ⇒ `*`

Atributo ⇒ `[atributo]`

`[atributo = "valor"]`

### Ejemplo de Selectores Básicos en CSS

Este párrafo está estilizado con el **selector de elemento**, cambiando su color a azul.

Este párrafo usa el **selector de clase** para resaltar el fondo en amarillo.

Nombre:

Email:

## Ejercicio 7

Usando el código proporcionado por el profesor: <https://codeshare.io/ZLrj3o> completa el CSS con los selectores adecuados

No olvides enlazar el css con el html.

---

**Subtítulo de la Página**

Este párrafo tiene que destacarse visualmente.

Elemento de lista 1

Elemento de lista 2

Elemento de lista 3

Usuario:

Contraseña:

## Ejercicio 8

1. Crear un documento HTML que contenga 4 encabezados, todos serán de tipo h1.
2. A cada encabezado habrá que darle un color distinto. Los que ocupen una posición par serán rojos y los impares verdes.
3. Finalmente, haz que el primero tenga color azul y el resto sigan igual que en el punto 2.

**Este encabezado h1 será de color azul**

**Este encabezado h1 será de color rojo**

**Este encabezado h1 será de color verde**

**Este encabezado h1 será de color rojo**

## Según la relación entre elementos

- Multiples  $\Rightarrow$  Separados por comas (Ej: p, h1, h2 {...})
- Descendentes  $\Rightarrow$  Separado por espacios (Ej: div p {...})
- Hijos directos  $\Rightarrow$  selector1 > selector2 {...} /\* No aplica a los nietos, etc.. \*/
- Dependientes  $\Rightarrow$  Seguidos sin espacios (Ej: p.miClase {...})
- Adyacentes  $\Rightarrow$  selector1 + selector2 {...} /\* Aplica a los hermanos \*/
- Todos los hermanos  $\Rightarrow$  div ~ p



## Ejercicio 9.

1. Crea un documento que contenga un body como el siguiente: <https://codeshare.io/Ek0yD8>

### Sección 1

Este es un párrafo de introducción.

Otro párrafo en la sección 1.

### Sección 2

Primer párrafo de la sección 2.

Segundo párrafo en la sección 2.

2. Usa el combinador de **descendiente** para que todos los párrafos dentro de `.container` tengan texto azul.
3. Aplica el combinador de **hijo directo** para que solo el `<h2>` dentro de `.container` sea rojo y no el de dentro de `<section>`.
4. Usa el combinador de **hermano adyacente** para que el primer `<p>` que sigue a un `<h2>` sea verde.
5. Aplica el combinador de **hermano general** para que todos los párrafos después de `.intro` tengan un fondo amarillo.
6. Usa un **selector múltiple** para aplicar el mismo estilo (margen inferior de 10px) a los elementos `<h2>` y `.intro`.
7. Aplica un **selector dependiente** para que cualquier `<p>` dentro de `.container` que también tenga la clase `.intro` tenga un color de fondo gris.

## Ejercicio 10

A partir del código HTML y CSS que se muestra, añadir los selectores CSS que faltan para aplicar los estilos deseados. Cada regla CSS incluye un comentario en el que se explica los elementos a los que debe aplicarse:

Código: <https://codeshare.io/zlnoMD>

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Praesent blandit nibh at felis. Sed nec diam in dolor vestibulum aliquet. Duis ullamcorper, nisi non facilisis molestie, lorem sem aliquam nulla, id lacinia velit mi vestibulum enim.

Phasellus eu velit sed lorem sodales egestas. Ut feugiat. Donec porttitor, **magna eu varius luctus**, metus massa tristique massa, in imperdiet est velit vel magna. Phasellus erat. Duis risus. Maecenas dictum, nibh vitae pellentesque auctor, tellus velit consectetuer tellus, tempor pretium felis tellus at metus.

Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Proin aliquam convallis ante. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Nunc aliquet. Sed eu metus. Duis justo.

Donec facilisis blandit velit. Vestibulum nisi. Proin volutpat, enim id iaculis congue, orci justo ultrices tortor, quis lacinia eros libero in eros. Sed malesuada dui vel quam. Integer at eros.



Avanzados (ya los veremos):

- Pseudoclase  $\Rightarrow$  selector:pseudoclase
- ...

[https://developer.mozilla.org/en-US/docs/Web/CSS/CSS\\_selectors](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_selectors)

Además del orden en cascada hay que tener en cuenta otras normas en caso de conflicto

## Importancia:

- Los estilos marcados con `!important` tienen la máxima prioridad, independientemente de dónde se encuentren en el código.

```
p { color: red !important; }
```

## Especificidad:

- Las reglas más específicas tienen prioridad sobre las más generales. La especificidad se calcula con un sistema de puntos basado en el selector:
  - Estilos en línea (`style="..."`) > 1000 puntos.
  - IDs (`#id`) > 100 puntos.
  - Clases, atributos y pseudoclases (`.class`, `[attr]`, `:hover`) > 10 puntos.
  - Elementos y pseudoelementos (`h1`, `::before`) > 1 punto.

```
<h1 id="#header" class="header-title">Encabezado 1</h1>
```

```
/* Más específico */  
#header { color: blue; } /* 100 puntos */  
.header-title { color: green; } /* 10 puntos */  
h1 { color: red; } /* 1 punto */
```

Además del orden en cascada hay que tener en cuenta otras normas en caso de conflicto

## Orden en el Código:

- Si dos reglas tienen la misma importancia y especificidad, se aplica la que esté más abajo en el archivo CSS.

```
p { color: red; }  
p { color: blue; } /* Este prevalece */
```

## Estilos por Defecto:

- Si no se define un estilo, el navegador utiliza sus estilos por defecto (User Agent Stylesheet).

## En resumen:

!important > Estilo en línea > IDs > Clases > Elementos > Última regla en el archivo > Navegador

## Ejemplos

Varias reglas entran en conflicto dentro de la misma declaración

```
p{  
background: red !important;  
background: crimson ;  
}
```

Especificidad calculada con sistema de puntos

```
<style>  
p{background: crimson;} /* Especificidad de 1 puntos */  
.parrafo{background: pink;} /*Especificidad de 10 puntos*/  
p.parrafo{background: maroon;} /*Especificidad de 11 puntos*/  
#id-parrafo{background: orange;}/*Especificidad de 100 puntos*/  
p#id-parrafo{background: red;}/*Especificidad de 101 puntos*/  
p.parrafo#id-parrafo{background:green;}/*Especificidad de 111 puntos*/  
</style>
```

## Ejercicio 11: estilos en función de la jerarquía

Crea un archivo HTML llamado `index.html` y un archivo CSS llamado `styles.css`.

En `index.html`, estructura el contenido de la siguiente manera:

- Incluye un encabezado (`<h1>`) con el texto "Título Principal" y asígnale la clase `titulo`.
- Agrega un párrafo (`<p>`) con el texto "Este es un párrafo de ejemplo" y asígnale el ID `parrafo`.
- Dentro de un contenedor `<div>`, agrega otro párrafo con la clase `texto` y el texto "Texto dentro de un div".

En `styles.css`, escribe las siguientes reglas en este orden:

- **Regla de elemento:** Aplica un color a todos los encabezados `<h1>`.
- **Regla de clase:** Aplica un color específico a los elementos con la clase `titulo`.
- **Regla de ID:** Define un color para los elementos con el ID `parrafo`.
- **Regla con `!important`:** Aplica un color específico a los elementos con la clase `texto` dentro del div `contenedor`, utilizando `!important`.
- **Última regla en el archivo:** Define un color para todos los elementos `<p>` al final del archivo CSS.

Agrega un estilo en línea al elemento `<h1>` en el archivo `index.html`, especificando un color diferente al resto de reglas CSS.

Carga el archivo en un navegador y responde a las siguientes preguntas:

- ¿Qué color se aplica finalmente al encabezado `<h1>`? Explica cuál fue la regla ganadora y por qué.
- ¿Cuál es el color del párrafo con el ID `parrafo`? ¿Qué regla prevalece?
- ¿Qué color tiene el párrafo con la clase `texto` dentro del div `contenedor`? Explica por qué se aplica ese color en particular.

Hay propiedades cuyos valores se pasan de padres a hijos y otras que no.

No se heredan: **margin, border, padding, width...**

Sí se heredan: **color, font, text, visibility...**

**Inherit** → Fuerza la herencia

**Initial** → Rompe la herencia



## Ejercicio 12:

1. Crea un archivo HTML y CSS con la estructura y estilos que se muestran a continuación.
2. Observa el comportamiento de los elementos según la herencia indicada.
3. Responde a las preguntas siguientes:
  1. ¿Qué estilo toma el texto con `inherit`? ¿Por qué?
  2. ¿Cómo se comporta el texto con `initial`? ¿Qué significa este valor?
  3. ¿Qué sucede con `unset`? ¿Es diferente para propiedades heredables y no heredables?
  4. ¿Cómo actúa `revert` en comparación con los otros valores?

# Los tres conceptos juntos



Concepto	Clave	Ejemplo
Cascada	Orden de prioridad	Última regla aplica.
Especificidad	Peso de los selectores	<code>#id</code> gana sobre <code>.clase</code> .
Herencia	Propiedades transmitidas por defecto	Color de texto en párrafos.

Nos van a dar la clave de cómo se aplica CSS a cada elemento.

## Ejercicio 13:

1. Utiliza el archivo HTML proporcionado como base.
2. Escribe un archivo CSS que cumpla con los requisitos de estilo especificados.
3. Observa cómo interactúan las reglas en el navegador y ajusta según sea necesario.
4. Responde a las preguntas al final del ejercicio.

### Requisitos de Estilo:

1. Usa **herencia** para aplicar un color de texto general a todo el documento.
2. Los párrafos dentro de la sección deben usar una fuente diferente de la predeterminada.
3. El elemento con la clase `highlight` debe tener un fondo amarillo y un texto en negrita.
4. El párrafo con el `id="override"` debe sobrescribir el color de texto general con uno personalizado.
5. Asegúrate de que el título principal (`h1`) herede el color general pero tenga un tamaño mayor definido directamente.
6. Usa al menos una regla con `!important` para ganar prioridad en la cascada.

### Preguntas para Reflexionar:

1. ¿Qué sucede si cambias el orden de las reglas CSS?
2. ¿Cómo afecta la especificidad a los estilos aplicados en el párrafo `#override`?
3. ¿Qué sucede si eliminas el `!important` de alguna regla?
4. ¿Qué reglas son heredadas automáticamente y cuáles necesitas especificar explícitamente?

## Unidades absolutas

px	Píxeles. Útil cuando queremos un tamaño fijo que no se redimensione con la página.
in	Pulgadas (1 pulgada = 2.54 cm)
cm	Centímetros
mm	Milímetros
pt	Puntos (1 pt = 1/72 pulgadas)
pc	Picas (1 pica = 12 puntos)

## Unidades relativas

em	Relativo al tamaño de la fuente del elemento ( 2 em significa 2 veces el tamaño de la fuente actual)
rem	Se basa en el tamaño de fuente del elemento raíz (HTML). Por defecto 16px
%	Porcentaje (relativo al elemento padre)
vh y vw	Medidas relativas de acuerdo al viewport 1vh = 1% de la altura del viewport 100vh = altura del viewport
fr	Flexible Grid Units (fr) Se utiliza en Grid Layout y representa una fracción del espacio disponible en un contenedor

Completa los huecos (\_\_\_) en el CSS con las unidades que se indican en los comentarios:

- `.vh-box`: Completar con `50vw` y `30vh`.
- `.vw-box`: Completar con `80%` de ancho y `200px` de alto.
- `.percent-box`: Completar con `60%`, `200px` y `500px`.

Abre el archivo HTML en el navegador y redimensiona la ventana para ver cómo responde cada bloque a los cambios de tamaño.

## Preguntas para Reflexionar:

- ¿Cómo afecta cada unidad (vh, vw, %) al diseño cuando se cambia el tamaño de la ventana?
- ¿Qué diferencias observas entre el comportamiento del `.vh-box` y el `.vw-box`?
- ¿Cómo limita el tamaño del bloque `.percent-box` la propiedad `max-width`?

<https://codeshare.io/3yNdPB>

En cada regla CSS usamos propiedades y valores



Hay algunas propiedades válidas para todos los elementos HTML:

color: Funciona en todos los elementos que contienen texto.

y otras que sólo funcionan en algunos elementos o circunstancias específicas.

list-style-type: Funciona solo en elementos de lista (<ul>, <ol>).

## Colores básicos

Color	Nombre	HEX	RGB
	black	#000000	0,0,0
	white	#ffffff	255,255,255
	red	#ff0000	255,0,0
	blue	#0000ff	0,0,255
	yellow	#ffff00	255,255,0
	gray	#808080	128,128,128
	green	#008000	0,128,0

HTML soporta alrededor de 140 nombres de colores diferentes



- **Nombre predefinido:** red, yellow, blue, green...
- **Código hexadecimal:** #RRGGBB o #RRGGBBAA
- **RGB:** Red, Green, Blue
- **RGBA:** Red, Green, Blue, Alpha
- **HSL:** Hue, Saturation, Lightness
- **HSLA:** Hue, Saturation, Lightness, Alpha

# Propiedades: colores y fondo



Propiedad	Descripción	Valores
<code>color</code>	Color del texto	RGB   HSL   HEX   nombre del color   RGBA   HSLA
<code>background-color</code>	Color de fondo	RGB   HSL   HEX   nombre del color   RGBA   HSLA   transparent
<code>background-image</code>	Imagen de fondo	url(«...»)   none
<code>background-repeat</code>	Repetición de la imagen de fondo	repeat   repeat-x   repeat-y   no-repeat
<code>background-attachment</code>	Desplazamiento de la imagen de fondo	scroll   fixed
<code>background-position</code>	Posición de la imagen de fondo	percentage   length   left   center   right
<code>background-size</code>	Tamaño de la imagen de fondo	auto   cover   contain   valor
<code>Opacity</code>	Transparencia de un elemento	[ 0 – 1 ] (0 → totalmente transparente)

# Propiedades: colores y fondo: Ejercicio 15



En cada sección del CSS (`.color-box`, `.image-box`, `.gradient-box`, `.multiple-box`, `.text-box`), completa los huecos (\_\_\_) con los valores que se indican en los comentarios:

- En `.color-box`, añade un `background-color` con un color hexadecimal (por ejemplo, `#3498db`) o un nombre de color (como `blue`).
- En `.image-box`, aplica una imagen de fondo y ajusta las propiedades de repetición (`background-repeat`), tamaño (`background-size`), y posición (`background-position`).
- En `.gradient-box`, usa `linear-gradient` para aplicar un gradiente de color de un lado al otro.
- En `.multiple-box`, combina una imagen y un gradiente como fondo múltiple, ajustando cada capa individual.
- En `.text-box`, cambia el color del texto y usa `rgba()` en `background-color` para darle opacidad.
- Redimensiona la ventana para ver cómo se ajustan los fondos y los colores.
- Experimenta cambiando los valores para ver cómo afectan cada bloque de manera visual.

<https://codeshare.io/zlnmPk>

# Propiedades: colores y fondo: Ejercicio 16



Crea un archivo HTML con una estructura básica que incluya al menos tres secciones de contenido (`div` o `section`), de preferencia con texto que permita que la página sea scrollable (es decir, que puedas desplazarte hacia abajo).

Aplica una imagen de fondo a una de las secciones usando `background-image`.

Define el comportamiento de la imagen de fondo en la sección usando diferentes valores de `background-attachment`:

- `Scroll`
- `fixed`
- `Local`

Guarda el archivo y abre la página en un navegador. Haz scroll en la página y observa cómo cambia el comportamiento de la imagen de fondo en cada sección según el valor de `background-attachment`. Escribe un breve párrafo al final de tu página HTML explicando las diferencias que observaste entre los valores scroll, fixed y local.

Puedes usar la documentación, por ejemplo de W3Schools: [https://www.w3schools.com/cssref/pr\\_background-attachment.php](https://www.w3schools.com/cssref/pr_background-attachment.php)

## Ejercicio 16 (continuación)

- Aplica un 50% de transparencia a la imagen del ejercicio
- Utiliza el modo RGBA para crear un texto con fondo blanco sobre la imagen que permita ver tanto la imagen como el texto.

Referencia: [https://www.w3schools.com/css/css\\_image\\_transparency.asp](https://www.w3schools.com/css/css_image_transparency.asp)

Las **propiedades de texto** son las que nos permiten controlar el texto como bloque, es decir, afectan al interlineado, a la separación entre palabras, al tabulado, etc.

Propiedad	Descripción	Valores
<code>text-indent</code>	Desplazamiento de la primera línea del texto	longitud   porcentaje
<code>text-align</code>	Alineamiento del texto	left   right   center   justify
<code>text-decoration</code>	Efectos de subrayado y tachado	none   underline   overline   line-through
<code>letter-spacing</code>	Espacio entre caracteres	normal   longitud
<code>word-spacing</code>	Espacio entre palabras	normal   longitud
<code>text-transform</code>	Transformación a mayúsculas / minúsculas	capitalize   uppercase   lowercase   none
<code>line-height</code>	Tamaño del espacio entre líneas (interlineado)	longitud   porcentaje
<code>vertical-align</code>	Alineación vertical	top   middle   bottom baseline   sub   super   valor

# Ejercicio 17



A partir del código HTML y CSS [provisto](#), aplica los valores adecuados a las propiedades según se indica:

**color:** Usa colores que contrasten y sean fácilmente visibles, como `#333333` para el texto oscuro y `#007acc` para títulos.

**font-size:** Prueba tamaños en píxeles, por ejemplo, `32px` para el título y `24px` para el subtítulo.

**text-align:** Usa `center` o `justify` para alinear el texto.

**text-transform:** Usa `uppercase` para mayúsculas o `capitalize` para capitalizar la primera letra de cada palabra.

**letter-spacing:** Prueba valores como `2px` o `3px` para el subtítulo.

**text-decoration:** Usa `underline` para subrayado en el subtítulo.

**text-indent:** Usa `30px` para indentar la primera línea del párrafo.

**font-style:** Usa `italic` en el bloque de cita.

**background-color:** Usa un color suave como `#f0f8ff` para resaltar el bloque de cita.

**padding y border-left:** Prueba `10px` para padding y `3px solid #007acc` para la línea de color del bloque de cita.

Responde:

1. ¿Cómo cambia la apariencia del texto cuando aplicas diferentes colores y tamaños?
2. ¿Qué combinación de propiedades y colores te parece más profesional o atractiva?

# Propiedades: fuentes



Las propiedades CSS de las **fuentes** son las que permiten controlar el tamaño, el tipo, el grosor o el estilo de las letras, etc.

Propiedad	Descripción	Valores
<code>font-family</code>	Familias de fuentes	nombre-familia   *
<code>font-style</code>	Estilo de la fuente	normal   italic   oblique
<code>font-variant</code>	Convierte a mayúsculas manteniendo todas las letras en un tamaño inferior a la primera	normal   small-caps
<code>font-weight</code>	Anchura de los caracteres. Normal = 400, Negrita = 700	normal   bold   bolder   lighter   100   200   300   400   500   600   700   800   900
<code>font-size</code>	Tamaño de la fuente	xx-small   x-small   small   medium   large   x-large   xx-large   larger   smaller   longitud   porcentaje



# Ejercicio 18



**Completa los valores** de cada propiedad en el archivo CSS:

- **font-family**: Usa familias de fuente estándar como **serif** o **sans-serif** para el título y primer párrafo, y aplica la fuente importada (**Roboto**) en el último párrafo.
- **font-size**: Prueba tamaños en píxeles o em (por ejemplo, **24px** para el título y **16px** para los párrafos).
- **font-weight**: Aplica valores como **bold**, **normal**, o numéricos como **300** o **700**.
- **text-transform**: Usa valores como **uppercase**, **lowercase**, o **capitalize** para cambiar la apariencia del texto.
- **letter-spacing**: Ajusta el espacio entre letras en el segundo párrafo con valores como **1px** o **2px**.
- **line-height**: Define una altura de línea como **1.5** o **1.8** en el último párrafo para mejorar la legibilidad.

Responde:

1. ¿Qué diferencias notas en la legibilidad entre las fuentes serif y sans-serif?
2. ¿Cómo afecta el tamaño de la fuente y el interlineado a la comodidad de lectura?
3. ¿Cuál es tu combinación de propiedades favorita para títulos y párrafos?

Dado el siguiente [código HTML](#), crea un fichero style.css siguiendo estas instrucciones:

## 1. Estiliza el cuerpo (body):

- Usa una fuente sans-serif por defecto, como Arial.
- Define un tamaño de fuente base de 16px.
- Ajusta la altura de línea a 1.6.

## 2. Estilo para el título principal (h1) en el header:

- Usa una fuente serif, como Georgia.
- Define el tamaño de fuente en 36px.
- Aplica un color oscuro, como #333.
- Alinea el texto al centro y usa la propiedad text-transform para poner el título en mayúsculas.

## 3. Estilo para el párrafo introductorio (p.intro) en el header:

- Usa una fuente en cursiva.
- Define un tamaño de fuente un poco más pequeño, como 14px.
- Cambia el color a un tono gris, como #666.

Dado el siguiente [código HTML](#), crea un fichero style.css siguiendo estas instrucciones:

## 4. Estilo para los subtítulos (h2) en el `section`:

- Usa una fuente sans-serif, como `Verdana`.
- Define un tamaño de fuente de `24px`.
- Aplica un color azul, como `#007acc`.
- Añade un subrayado usando `text-decoration`.

## 5. Estilo para el primer párrafo de texto (`p.text1`):

- Usa una fuente sans-serif.
- Define un tamaño de fuente de `16px`.
- Aplica un interlineado de `1.5`.
- Justifica el texto usando `text-align`.

## 6. Estilo para el bloque de cita (`blockquote.highlight`):

- Usa una fuente serif y cambia el color a un tono verde oscuro, como `#2a7a1f`.
- Aplica un fondo de color suave, como `#f0f8f8`.
- Agrega un margen a la izquierda usando `margin-left`.
- Añade un borde a la izquierda, de `3px`, en un tono verde.

Dado el siguiente [código HTML](#), crea un fichero style.css siguiendo estas instrucciones:

## 7. Estilo para el segundo párrafo de texto (p.text2):

- Usa un estilo de texto en negrita.
- Define un tamaño de fuente de **16px** y aplica un espacio entre letras de **1px**.
- Cambia el color a un tono oscuro, como **#444**.

## 8. Estilo para el pie de página (footer):

- Usa un tamaño de fuente más pequeño, como **12px**.
- Centra el texto y cambia el color a un tono gris claro, como **#888**

# Propiedades: listas



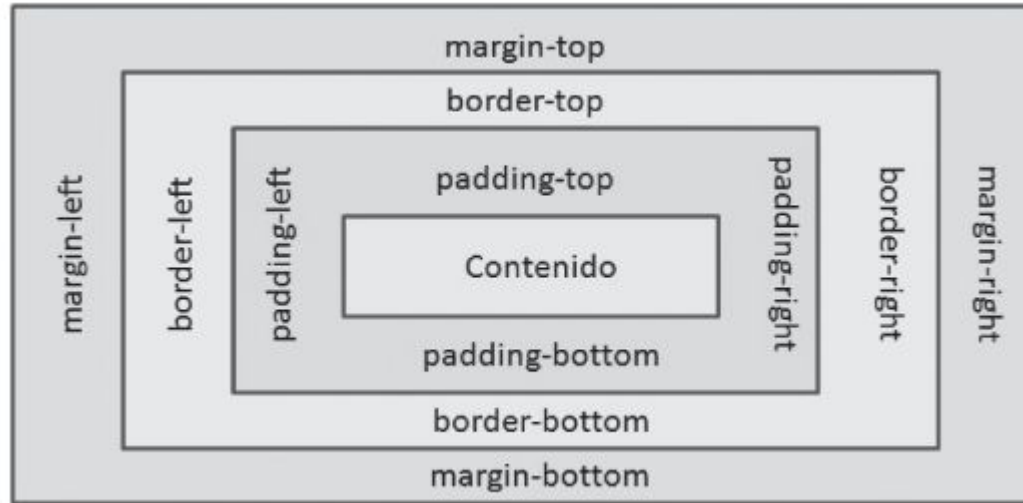
Las propiedades CSS de las **listas** son las que nos permiten controlar los estilos de los marcadores y la posición de los elementos dentro de las listas

Propiedad	Descripción	Valores
<code>list-style-type</code>	Estilo aplicable a los marcadores visuales o viñetas de las listas	disc   circle   square   decimal   decimal-leading-zero   lower-roman   upper-roman   lower-greek   lower-latin   upper-latin   armenian   georgian   lower-alpha   upper-alpha   none
<code>list-style-image</code>	Imagen aplicable a las viñetas de las listas	url()   none
<code>list-style-position</code>	Posición de las viñetas dentro de la lista	inside   outside
<code>list-style</code>	Permite establecer varios estilos de la lista en una sola propiedad	list-style-type   list-style-position   list-style-image

Crea un archivo CSS llamado `styles.css`. Sigue las instrucciones de estilo para aplicar a cada lista en tu archivo `styles.css`:

- **1. Lista ordenada de tareas (`ol.tareas`):**
  - Usa el estilo `list-style-type` con valores como `decimal-leading-zero`.
  - Ajusta la posición de la lista usando `list-style-position` para que las marcas de lista queden dentro del texto.
  - Cambia el color del texto de los elementos de la lista a un tono azul oscuro, como `#003366`.
- **2. Lista no ordenada de materiales (`ul.materiales`):**
  - Cambia el tipo de lista con `list-style-type` usando un estilo `square`.
  - Aplica `list-style-position` para que las marcas queden fuera del texto.
  - Cambia el color del texto a un tono verde oscuro, como `#2a7a1f`.
- **3. Lista personalizada de recursos de estudio (`ul.recursos`):**
  - Usa una imagen personalizada para los elementos de la lista mediante `list-style-image`. Puedes usar una imagen, como un pequeño icono de libro o una estrella.
  - Ajusta la posición de la lista usando `list-style-position` de manera que las marcas de lista queden dentro del texto.
  - Cambia el color del texto a un tono gris oscuro, como `#444444`.

Todo elemento HTML se estructura de esta forma



Todos cuentan para calcular el tamaño real del elemento, salvo el margen que es el espacio entre elementos.

## Ejercicio 21

[https://www.w3schools.com/css/css\\_border.asp](https://www.w3schools.com/css/css_border.asp)

1. **Estructura del archivo HTML:** Diseña una página con tres secciones, cada una con una etiqueta `<div>`, que tengan el siguiente contenido:
  - La primera sección tendrá un título `<h2>` con el texto "Estilo de Bordos".
  - La segunda sección tendrá un párrafo `<p>` con el texto "Bordes con Colores y Grosos".
  - La tercera sección tendrá un botón `<button>` con el texto "Bordes Redondeados".
2. **Estructura del archivo CSS:** aplica las siguientes propiedades de borde a cada sección:
  - **Primera Sección:**
    - Aplica un borde con estilo `dotted` (punteado).
    - El grosor debe ser de 3px y el color debe ser azul.
    - Deja un margen de 20px alrededor y un relleno (padding) de 10px.
  - **Segunda Sección:**
    - Aplica un borde con estilo `solid` (sólido).
    - Define un grosor diferente para cada lado: 5px arriba, 10px derecha, 15px abajo, y 20px izquierda.
    - El color debe ser verde.
  - **Tercera Sección (Botón):**
    - Aplica un borde con estilo `double` (doble línea).
    - Usa un borde rojo de 4px.
    - Opcional: Añade esquinas redondeadas con `border-radius` de 15px.



## Ejercicio 22

1. **Crea un archivo HTML** llamado `margenes.html` y un archivo CSS llamado `margenes.css`.

2. **Estructura del archivo HTML:**

Diseña una página con tres secciones, cada una con una etiqueta `<div>` que contenga el siguiente contenido:

- La primera sección tendrá un título `<h2>` con el texto "Márgenes Simples".
- La segunda sección tendrá un párrafo `<p>` con el texto "Márgenes Asimétricos".
- La tercera sección tendrá un botón `<button>` con el texto "Márgenes Automáticos".

3. **En `margenes.css`**, aplica las siguientes reglas de margen a cada sección:

- **Primera Sección:**
  - Aplica un margen uniforme de 20px a todos los lados.
  - Cambia el fondo para que sea visible la separación entre el contenido y otros elementos.
- **Segunda Sección:**
  - Define márgenes diferentes para cada lado:
    - 10px arriba, 20px derecha, 30px abajo y 40px izquierda.
  - Cambia el color de fondo para resaltar la diferencia.
- **Tercera Sección (Botón):**
  - Utiliza `margin: auto` para centrar el botón horizontalmente dentro de su contenedor.
  - Añade un margen superior de 50px para separarlo del párrafo anterior.

[https://www.w3schools.com/css/css\\_padding.asp](https://www.w3schools.com/css/css_padding.asp)

## Ejercicio 23

1. Crea un archivo HTML llamado `relleno.html` y un archivo CSS llamado `relleno.css`.

2. Estructura del archivo HTML:

Diseña una página con tres secciones, cada una con una etiqueta `<div>` que contenga el siguiente contenido:

- La primera sección tendrá un título `<h2>` con el texto "Relleno Uniforme".
- La segunda sección tendrá un párrafo `<p>` con el texto "Relleno Asimétrico".
- La tercera sección tendrá un botón `<button>` con el texto "Relleno Grande".

3. En `relleno.css`, aplica las siguientes reglas de relleno a cada sección:

○ **Primera Sección:**

- Aplica un relleno uniforme de 20px a todos los lados.
- Cambia el color de fondo para que sea visible el relleno.

○ **Segunda Sección:**

- Define rellenos diferentes para cada lado:
  - 10px arriba, 15px derecha, 20px abajo, y 25px izquierda.
- Cambia el color de fondo para resaltar el espacio interno.

○ **Tercera Sección (Botón):**

- Usa un relleno interno grande de 30px arriba y abajo, y 50px a los lados.
- Cambia el diseño del botón para que sea visualmente atractivo.

## Ejercicio 24

1. Crea un archivo HTML llamado `tamano.html` y un archivo CSS llamado `tamano.css`.

2. Estructura del archivo HTML:

Diseña una página con tres secciones, cada una con una etiqueta `<div>` que contenga el siguiente contenido:

- La primera sección tendrá un cuadro (`<div>`) con el texto "Tamaño Fijo".
- La segunda sección tendrá un cuadro (`<div>`) con el texto "Porcentaje de Ancho".
- La tercera sección tendrá un cuadro (`<div>`) con el texto "Altura Mínima y Máxima".

3. En `tamano.css`, aplica las siguientes reglas a cada sección:

○ **Primera Sección:**

- Establece un tamaño fijo de 200px de ancho y 150px de alto.
- Cambia el color de fondo para que el tamaño sea visible.

○ **Segunda Sección:**

- Establece un ancho del 50% del contenedor padre y una altura de 100px.
- Cambia el color de fondo para resaltar el tamaño relativo.

○ **Tercera Sección:**

- Establece un ancho fijo de 300px.
- Define una altura mínima de 100px y una altura máxima de 200px.
- Cambia el color de fondo para resaltar el comportamiento del tamaño.

## Ejercicio de Autoevaluación

<https://www.w3schools.com/css/exercise.asp>

- Border
- Margin
- Padding
- Height/Width
- Box model

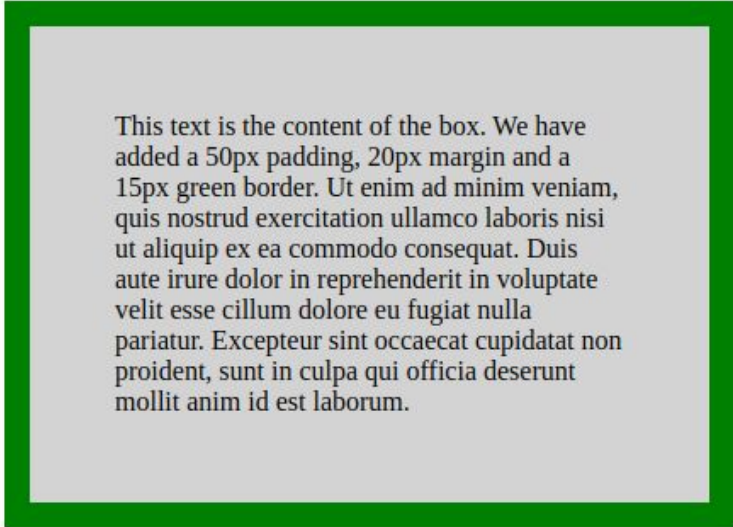
CSS Border
CSS Margin
CSS Padding
CSS Height/Width
CSS Box Model

## Ejercicio 25

Crea un HTML con un elemento div que contenga un texto cualquiera. Luego aplícale los estilos siguientes:

1. Fondo gris
2. Ancho de 300px y alto 300px
3. Borde de 15px sólido y verde
4. Relleno de 50px
5. Margen de 20px

Calcula la anchura y altura total del div



This text is the content of the box. We have added a 50px padding, 20px margin and a 15px green border. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

## content-box vs. border-box

### content-box (El Default)

El `width` y `height` que defines se aplican **SÓLO** al contenido.

El tamaño total del elemento es la suma de todo:

**Tamaño Real = width + padding + border**

Esto es confuso y hace difíciles los cálculos.

### border-box (El Recomendado)

El `width` y `height` que defines **INCLUYEN** el contenido, el padding y el borde.

**El tamaño real del elemento es el `width`** que especificaste.

Es mucho más intuitivo y fácil de maquetar.

## Ejercicio 26

1. Mira el resultado del código en el *preview*. Observa las dos cajas:

- **Caja 1 (Roja, `content-box`):** A pesar de tener un `width: 200px`, verás que es visiblemente más ancha que la Caja 2.
- **Caja 2 (Verde, `border-box`):** El ancho visible es exactamente de `200px`.

2. Modifica el código para lograr lo siguiente:

- **Objetivo 1:** Haz que la **Caja 1 (`content-box`)** también tenga un ancho final (el espacio que ocupa en la pantalla) de `200px`.
- **Objetivo 2:** Mantén el `padding` en `20px` y el `border` en `10px` para ambas cajas.

Para lograr el Objetivo 1, solo puedes cambiar la propiedad `width` de la Caja 1, sin alterar el `box-sizing`.

# Lenguaje CSS: width, max-width y margin: auto



```
div {  
    /* Fija el tamaño máximo. Sin esto ocuparía toda la línea */  
    max-width: 500px;  
    /* El navegador calcula un espacio igual a ambos lados de su contenedor.  
    Sin esto, se situaría al principio de la línea.*/  
    margin: auto;  
}
```

```
div {  
    /* En lugar de ocupar toda la línea, sólo ocupa 500px */  
    /* Pero al estrechar la pantalla no se reduciría y aparecería  
    un scroll horizontal*/  
    width: 500px;  
    /* El navegador calcula un espacio igual a ambos lados de su contenedor.  
    Sin esto, se situaría al principio de la línea.*/  
    margin: auto; |  
}
```



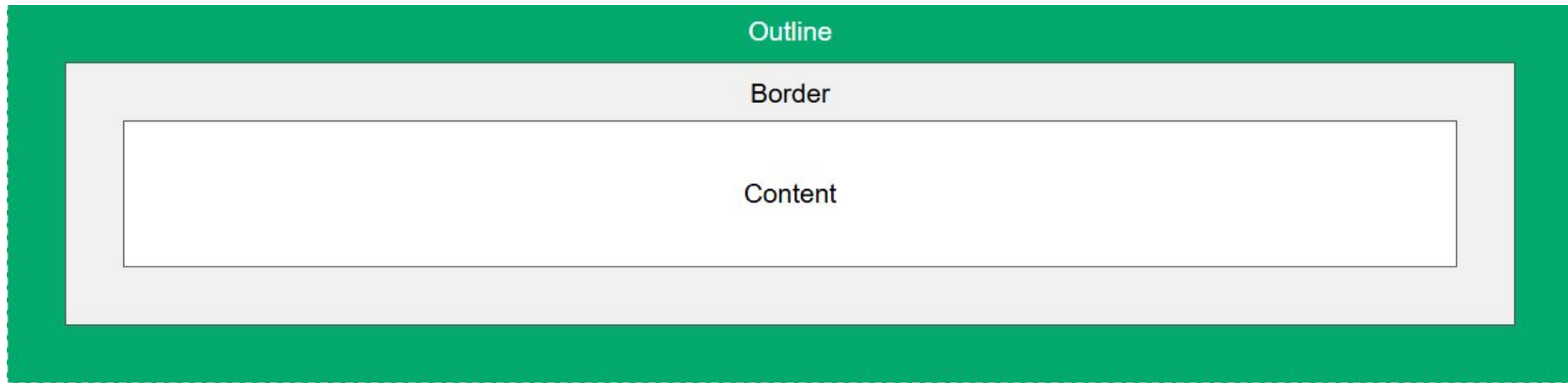
# Ejercicio 27



Imaginad que estáis trabajando en el diseño de un blog. El diseño exige que el contenido principal siempre esté centrado para ser estético, y que nunca se estire tanto en un monitor gigante que sea imposible de leer. Además, en el móvil, debe tener un ancho mínimo para que el texto no se apriete.

La misión es modificar el CSS para que la caja azul (.contenedor) sea 100% responsiva y fácil de leer.

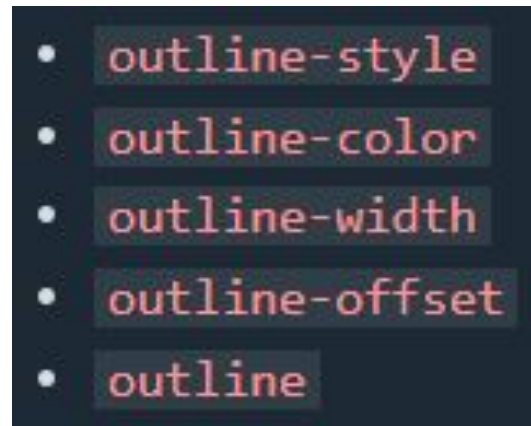
Requisito	Acción Requerida	Propiedad Clave	Valor Esperado
1. Centrado	El <code>.contenedor</code> debe estar perfectamente centrado en la ventana del navegador.	<code>margin</code>	<code>... auto;</code>
2. Ancho por Defecto	Ocupa el 90% del ancho de la pantalla (para dejar espacio a los lados).	<code>width</code>	<code>90%</code>
3. Legibilidad (Gigante)	<b>Nunca</b> debe ser más ancho de 900px, incluso en monitores 4K.	<code>max-width</code>	<code>900px</code>
4. Estructura (Móvil)	<b>Nunca</b> debe ser más estrecho de 300px, para proteger la legibilidad.	<code>min-width</code>	<code>300px</code>
5. Altura Fija	La caja naranja ( <code>.caja-imagen</code> ) debe tener una altura fija de 200 píxeles.	<code>height</code>	<code>200px</code>



Es una especie de borde que se usa para resaltar un elemento durante el cambio de foco o para depurar.

La característica más importante del `outline` es que **no ocupa espacio**.

- No afecta el tamaño total del elemento (como sí lo hacen el `padding` y el `border`).
- No provoca que otros elementos se muevan o cambien de posición.
- Se dibuja por encima de otros contenidos si es necesario.



## Ejercicio 28

1. Crea un archivo HTML llamado `outline.html` y un archivo CSS llamado `outline.css`.

2. Estructura del archivo HTML:

Diseña una página con tres secciones, cada una con un cuadro representado por un `<div>`.

- La primera sección tendrá un cuadro con el texto "Estilo de Contorno".
- La segunda sección tendrá un cuadro con el texto "Ancho de Contorno".
- La tercera sección tendrá un cuadro con el texto "Desplazamiento del Contorno".

3. En `outline.css`, aplica las siguientes reglas:

- **Primera Sección:**

- Usa diferentes estilos de contorno (`solid`, `dotted`, `dashed`) y elige uno.
- Cambia el color del contorno para que sea visible.

- **Segunda Sección:**

- Define un ancho de contorno de 5px y utiliza un color diferente.

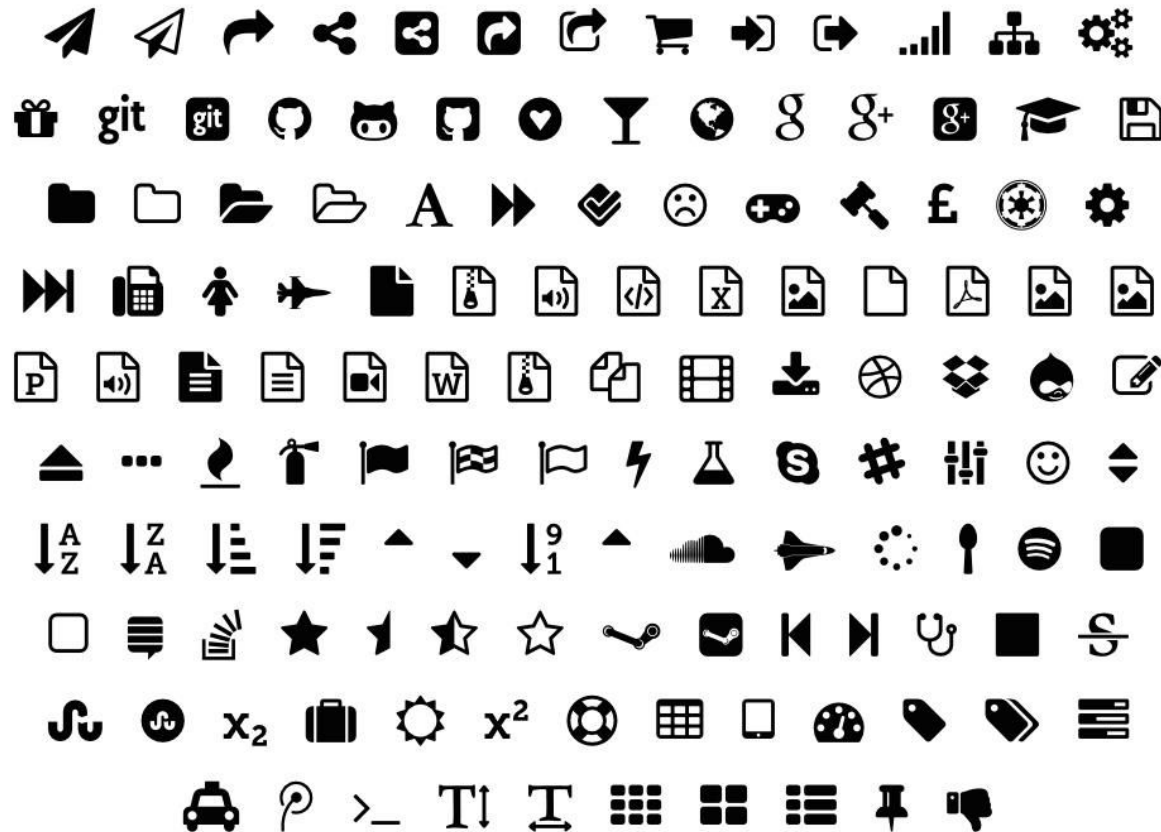
- **Tercera Sección:**

- Aplica un `outline-offset` de 10px para separar el contorno del cuadro.
- Cambia el color y el estilo del contorno para resaltar la diferencia.

4. Visualiza el resultado:

Abre `outline.html` en tu navegador y observa cómo el contorno afecta a los elementos y sus alrededores.

# Lenguaje CSS: añadir glifos (iconos)



Son parte de una fuente y se comportan como texto.

Se le pueden aplicar propiedades de los textos: color, tamaño...

Sólo es necesario importar la librería que deseemos:

- Bootstrap
- Google
- Font Awesome

## 1. Importar librería en nuestro HTML



```
<html>
<head>
  <!-- Place your kit's code here -->
  <script src="https://kit.fontawesome.com/e39c7a24be.js" crossorigin="anonymous"></script>
</head>
```

## 2. Crear los contenedores que contendrán los iconos

`<i></i>` `<span></span>` ... son los más usuales para iconos

## 3. Añadir a estos elementos el valor adecuado en el atributo class.

```
<body>
  <i class="fa-solid fa-thumbs-up fa-5x"></i>
</body>
```

Para ello hay que consultar la documentación de la librería

## Ejercicio 29

Crea un HTML con varios iconos de al menos dos librerías diferentes.

Aplica a uno de los iconos un color y tamaño personalizado.

Opcional:

¿Serías capaz de aplicar alguna animación u otra característica?

- Bordes
- Tamaño
- Alineación del contenido
- Propiedades de las tablas:

Nombre	Apellido
Fran	Gómez
Pepito	Grillo

Propiedad	Descripción	Valores
<code>caption-side</code>	Posición del título respecto la tabla	top   bottom
<code>table-layout</code>	Formato de las celdas, filas y columnas	auto   fixed
<code>border-collapse</code>	Selección del modelo de los bordes	collapse   separate
<code>border-spacing</code>	Espaciado entre los bordes de celdas adyacentes	longitud
<code>empty-cells</code>	Visibilidad de los bordes de celdas sin contenido	show   hide

## Ejercicio 23

1. **Crea un archivo HTML** llamado `tablas.html` y un archivo CSS llamado `tablas.css`.
2. **Estructura en HTML:**
  - Diseña una tabla que represente un horario semanal.
  - La tabla debe tener:
    - Un título usando `<caption>`.
    - Encabezados para los días de la semana.
    - Filas para las horas del día.
    - Algunas celdas vacías para aplicar la propiedad `empty-cells`.
3. **En el archivo CSS**, aplica las siguientes reglas:
  - Cambia la posición del título de la tabla usando `caption-side`.
  - Configura los bordes de la tabla usando `border-collapse` y `border-spacing`.
  - Ajusta el diseño de la tabla con `table-layout`.
  - Personaliza cómo se muestran las celdas vacías usando `empty-cells`.
4. **Añade estilos generales:**
  - Alterna colores en las filas para mejorar la legibilidad.
  - Cambia el color del borde y el fondo de las celdas de encabezado.



Propiedad **display** → Indica cómo se muestra el elemento

Cada elemento tiene un valor **display** por defecto:

- Elementos de bloque (display:block)
- Elementos de línea (display:inline)

Valores: none | inline | block | inline-block

## Ejercicio 24

1. **Crea un archivo HTML** llamado `display.html` y un archivo CSS llamado `display.css`.
2. **Estructura en HTML:**
  - Crea una lista de elementos con diferentes etiquetas (`div`, `span`, `p`, `button`, etc.).
  - Añade un párrafo que explique qué es `display` y cómo funciona cada valor.
3. **En el archivo CSS**, realiza lo siguiente:
  - Aplica `display: block` a un elemento para que ocupe todo el ancho disponible.
  - Usa `display: inline` para colocar varios elementos en línea.
  - Usa `display: inline-block` para combinar características de `block` e `inline`.
  - Oculta un elemento con `display: none`.
4. **Opcional:** Añade un poco de interactividad con JavaScript para que los elementos ocultos puedan aparecer al hacer clic en un botón.

La propiedad `visibility` indica si un elemento es visible o permanece oculto (ocupando el mismo espacio).

Valores: **visible** | **hidden**

¿Cuál es la diferencia entre `visibility: hidden` y `display: none`?

## Ejercicio 31 (continuación)

5. Modifica el ejercicio anterior para incluir otro elemento y juega con los diferentes valores de **visibility**. Si hiciste el botón, crea otro botón para cambiar la propiedad **visibility** a parte de la propiedad **display**.

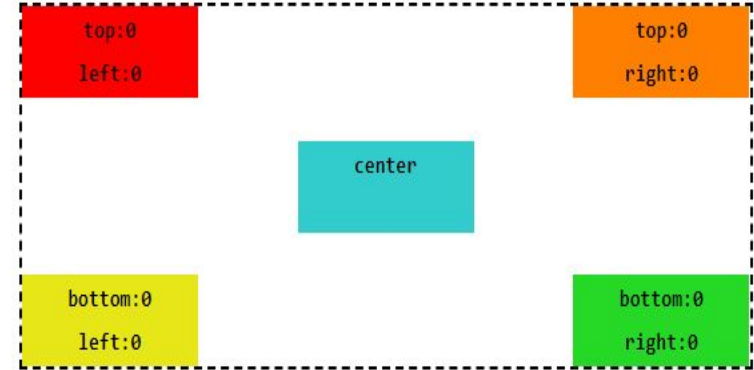
# Posicionamiento (layout): **position**

Para posicionar un elemento usamos:

- top, bottom, left y right

En conjunción con:

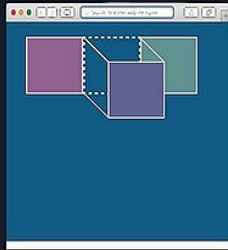
- position: static | relative | absolute | fixed



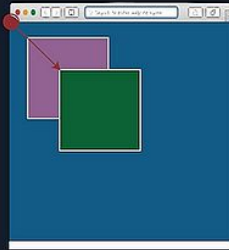
## CSS Position Properties



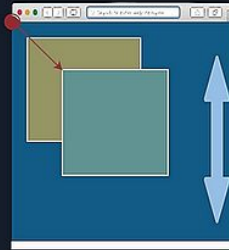
Static



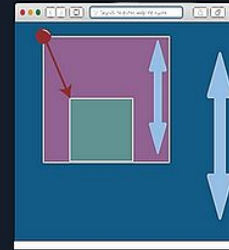
Relative



Absolute



Fixed



Sticky

Valor de position	Descripción del Comportamiento	Relación con las Propiedades top, bottom, left, right	Flujo Normal del Documento
<b>static</b>	Es el valor <b>por defecto</b> . El elemento sigue el flujo normal del documento (de izquierda a derecha, de arriba abajo).	<b>Ignoradas.</b> No tienen efecto.	Permanece en el flujo.
<b>relative</b>	El elemento se posiciona <b>relativo a su posición normal</b> en el flujo.	<b>Aplica:</b> Mueve el elemento de su posición original.	Permanece en el flujo. <b>Su posición original sigue ocupando espacio</b> , incluso si el elemento se ha movido.
<b>absolute</b>	El elemento se posiciona <b>relativo al ancestro posicionado más cercano</b> (cualquier ancestro que no sea <b>static</b> ). Si no hay ancestros posicionados, se posiciona respecto al <b>&lt;body&gt;</b> .	<b>Aplica:</b> Define la distancia desde el borde del ancestro posicionado.	<b>Sale del flujo.</b> No afecta la posición de otros elementos (actúa como si no existiera en el layout).
<b>fixed</b>	El elemento se posiciona <b>relativo a la ventana del navegador (viewport)</b> . No se mueve al hacer <i>scroll</i> .	<b>Aplica:</b> Define la distancia desde el borde de la ventana del navegador.	<b>Sale del flujo.</b> No afecta la posición de otros elementos.
<b>sticky</b>	Es un híbrido. Se comporta como <b>relative</b> hasta que la posición de <i>scroll</i> de la ventana del navegador alcanza un umbral definido por <b>top/bottom/left/right</b> .	<b>Aplica:</b> Define el umbral de activación.	Permanece en el flujo hasta que se "pega". <b>Mantiene su espacio</b> en el flujo normal.

## Usos comunes

### 1. **relative** (Relativo):

- **Tip Útil:** Se usa frecuentemente para establecer un **Contexto de Posicionamiento** para los elementos hijos que serán **absolute**. Un contenedor con **position: relative;** no se mueve, pero permite que sus hijos **absolute** se posicionen respecto a él.

### 2. **absolute** (Absoluto):

- **Uso Común:** Superponer un elemento (como un menú desplegable, una *badge* o un *tooltip*) dentro de un contenedor específico. **Es necesario** que el contenedor padre tenga **position: relative;** para que el elemento **absolute** se ancle a él.

### 3. **fixed** (Fijo):

- **Uso Común:** Barras de navegación persistentes (**top: 0;**), botones de "Volver arriba" o ventanas modales de alerta que deben permanecer visibles sin importar el *scroll*.

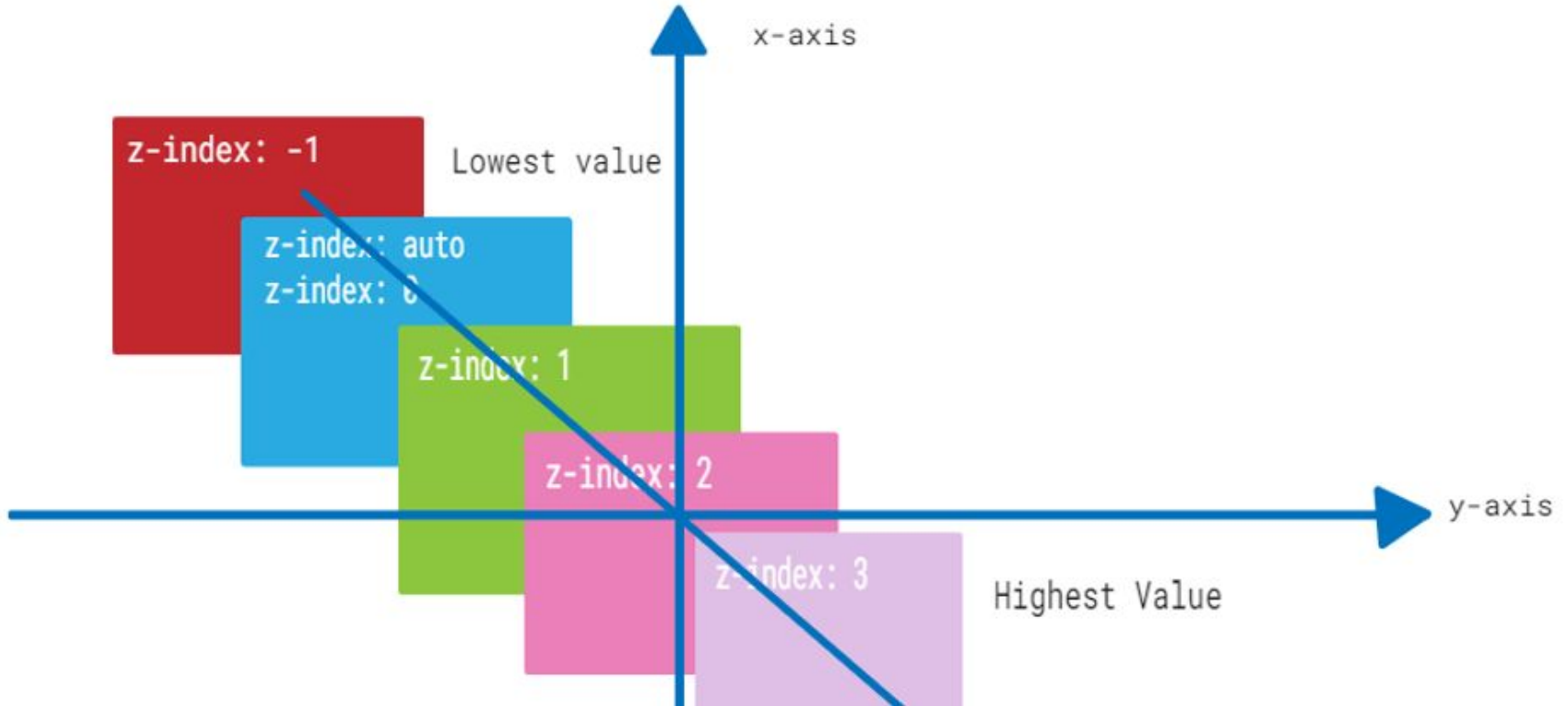
### 4. **sticky** (Pegajoso):

- **Uso Común:** Encabezados de tabla que deben permanecer visibles al desplazarse por una lista larga, o títulos de sección que se "pegan" a la parte superior de la pantalla hasta que la siguiente sección los empuja hacia arriba.

## Ejercicio 32

1. **Crea un archivo HTML** llamado `position.html` y un archivo CSS llamado `position.css`.
2. **Estructura en HTML:**
  - Diseña una página con una caja principal (`div`) que contiene cuatro elementos (`div`) internos, cada uno con una posición diferente.
  - Incluye texto descriptivo para cada tipo de posición.
3. **En el archivo CSS**, realiza lo siguiente:
  - Aplica `position: static` a uno de los elementos y explica su comportamiento por defecto.
  - Usa `position: relative` para desplazar un elemento desde su posición original.
  - Aplica `position: absolute` para posicionar un elemento respecto al contenedor principal.
  - Usa `position: fixed` para crear un elemento que permanezca fijo en la ventana del navegador mientras se desplaza el resto del contenido.
  - Usa `position: sticky` para crear un elemento que permanezca anclado a alguna zona de la ventana aunque se haga scroll
4. **Añade estilos visuales** para diferenciar claramente los elementos con colores, bordes y márgenes.

# Posicionamiento (layout): z-index



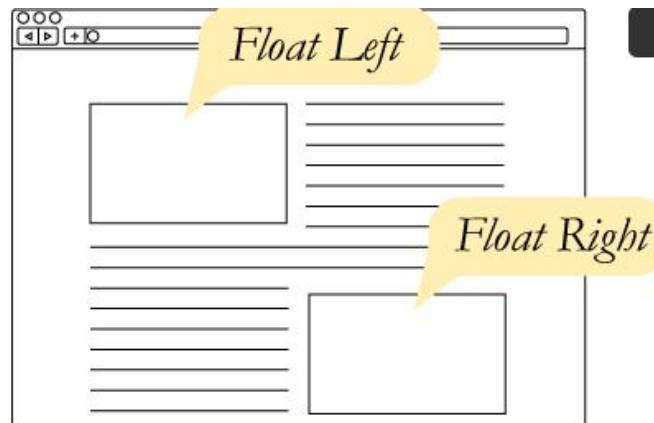


## Ejercicio 33

1. Crea una página web con tres cuadrados
  2. Utiliza la propiedad `z-index` para controlar el orden de apilamiento (qué elemento aparece encima de los demás).
  3. Modifica los valores de `z-index` directamente en el CSS para que los cuadrados cambien su orden de apilamiento.
- No olvides poner el position. Ej: `position: absolute`.

Rompe el flujo natural de los elementos

- left
- right
- none (por defecto)



Si flotamos un elemento el resto se coloca a su izquierda o a su derecha.

Esto se rompe con clear, que hace que el resto de elementos no se floten

- Clear: left → resto de elementos se ponen debajo de los flotados a la izquierda
- Clear: right → debajo de los flotados a la derecha
- Clear: both → ambos

1. Crea una página web que tenga:
  - Un encabezado.
  - Tres columnas flotantes con texto.
  - Un pie de página que quede debajo de las columnas.
2. Usa `float` para alinear las columnas y `clear` para asegurarte de que el pie de página no quede afectado por el flotamiento de las columnas.

## Prueba con Clear:

- Quita la línea `clear: both;` en el pie de página para ver cómo afecta la posición del mismo.
- Vuelve a añadirla y observa cómo se soluciona el problema.

## Añade más columnas:

- Crea una cuarta columna llamada `column extra`.
- Ajusta los tamaños para que todas encajen correctamente.

## Experimenta con el Flujo:

- Cambia `float: left` a `float: right` en una o más columnas para observar cómo afecta la disposición.

## Responde:

- ¿Qué sucede si no se usa `clear`?
- ¿Qué otros elementos además de `clear` pueden solucionar problemas similares?

## Maquetación de Artículo con **float**

El objetivo es lograr que la imagen inicial del artículo flote a la derecha, mientras que el texto del primer párrafo la envuelve suavemente.

1. En la sección CSS, dentro de la clase `.article-image`, añade la propiedad `float: right;`
2. Observa cómo el texto del primer y segundo párrafo envuelve la imagen.
3. En la sección CSS, dentro de la clase `.footer-note`, añade la propiedad `clear: both;`
4. Observa cómo la "Nota de pie" se mueve hacia abajo, quedando debajo de la imagen flotante, resolviendo así el solapamiento.

### El Renacimiento de las Micro-Comunidades Digitales

**\*\*Párrafo 1:\*\*** La historia de Internet es cíclica. Después de décadas de centralización en grandes plataformas sociales, estamos observando un retorno fascinante a las micro-comunidades. Estas son burbujas digitales más pequeñas, privadas y temáticas, donde la calidad de la interacción supera a la cantidad de audiencia. La necesidad de pertenecer a algo más íntimo y curado está impulsando este movimiento.



Imagen Flotante

**\*\*Párrafo 2:\*\*** Este cambio de paradigma presenta desafíos de diseño únicos. La maquetación de contenido debe ser flexible, permitiendo que elementos visuales, como la imagen de la comunidad que vemos a la derecha, se integren perfectamente con el flujo de lectura sin interrumpirlo. Aquí es donde el uso de técnicas como `float` sigue siendo invaluable para el diseño editorial.

- Centrar un elemento de bloque → `margin:auto` y `width`
- Centrar texto en un contenedor → `text-align: center`
- Centrar una imagen → `display: block; margin: auto`
- Alinear a izquierda o derecha
  - → `position:absolute` y `left/right`
  - → `float`
- Centrar verticalmente
  - → `padding`
  - → `line-height = height`



[https://www.w3schools.com/css/css\\_align.asp](https://www.w3schools.com/css/css_align.asp)

- Centrar un texto según el tamaño del viewport (con line-height)
- vh y vw: Centrar cualquier elemento según el ancho del viewport (con propiedad transform)
- Diseño de footer fijo en la parte inferior del viewport solo si hay poco contenido
- Usando inline-block

## Ejercicio Práctico: Centrar una Fila de Chips

A continuación, tenemos una serie de elementos "Chip" que deberían estar centrados horizontalmente, incluso cuando el ancho de la pantalla obliga a que salten de línea.

¡Observa el comportamiento de `inline-block` y ajusta la ventana para ver cómo se centra cada fila!

El contenedor padre usa `text-align: center;`

CSS Básico

HTML5

Modelo de Cajas

Flexbox

Grid CSS

JavaScript ES6

ReactJS

Tailwind CSS

SASS

Accesibilidad

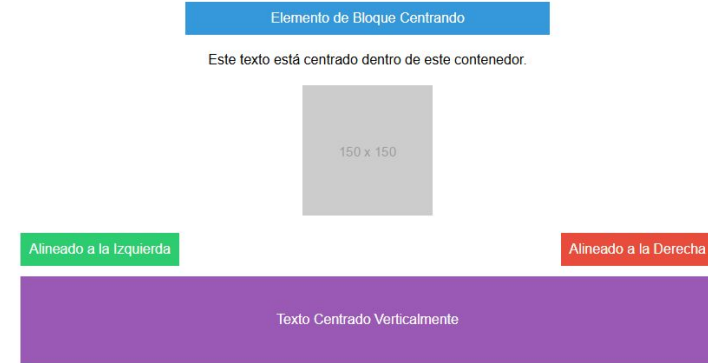
### Tarea:

1. Inspecciona el código CSS del `**.contenedor-chips**` (el padre).
2. Inspecciona el código CSS del `**.chip**` (los hijos).
3. **\*\*Pregunta de Auto-Evaluación:\*\*** Si cambiaras el `display` del `.chip` a `block`, ¿qué ocurriría? ¿Y si lo cambiaras a `inline`?



## Ejercicio 36

1. Crea una página web con un contenedor principal que incluya:
  - Un elemento de bloque centrado horizontalmente.
  - Texto centrado dentro de un contenedor.
  - Una imagen centrada horizontalmente.
  - Elementos alineados a la izquierda y derecha usando diferentes métodos.
  - Un elemento centrado verticalmente utilizando `padding` y `line-height`.



Explora cada método:

- Prueba con diferentes anchos para el elemento de bloque.
- Modifica la alineación del texto para observar cómo afecta el diseño.

Agrega contenido adicional:

- Añade más imágenes y elementos en las secciones para reforzar el concepto.

Experimenta con valores:

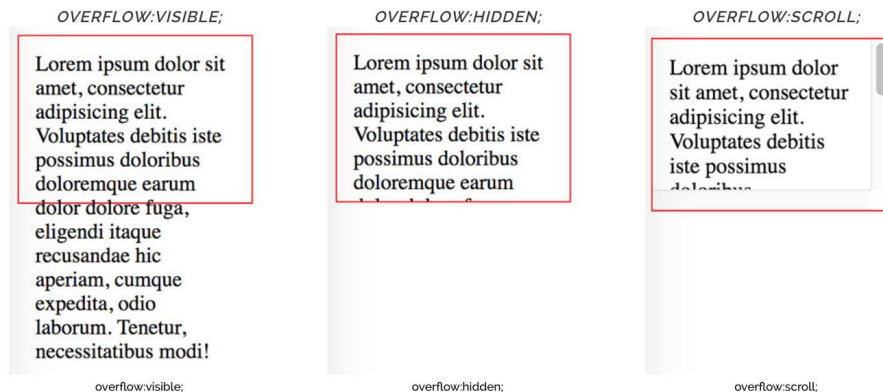
- Cambia los valores de margin, line-height, y padding para observar sus efectos en el diseño.

Preguntas para Reflexión:

- ¿Qué sucede si quitas el margin: auto del elemento de bloque?
- ¿Cómo afecta la altura al centrado vertical con line-height?

## Cuando el contenido sobresale del contenedor

- **Overflow: visible (default):** contenido se sale del contenedor y es totalmente visible.
- **Overflow: hidden:** lo que sobresale se oculta
- **Overflow: scroll:** se muestra una barra de *scroll* (horizontal y vertical)
- **Overflow: auto:** como scroll pero sólo cuando sobresale el contenido



Sólo aplica a  
contenedores de  
bloque

## Ejercicio 37:

1. Crea una página web con un contenedor de tamaño fijo que contenga un texto largo o varios elementos.
2. Aplica diferentes valores a la propiedad `overflow` y observa el comportamiento del contenido.
3. Configura los ejes horizontal (`overflow-x`) y vertical (`overflow-y`) por separado para explorar cómo afectan al scroll.

## Selectores más específicos sin necesidad de clases

```
selector:pseudo-class {  
  property: value;  
}
```

Pseudo-clase	Descripción
:first-child	Primer hijo
:last-child	Último hijo
:first-of-type	Primer hermano de su tipo
:last-of-type	Último hermano de su tipo
:only-child	Hijos únicos
:only-of-type	Únicos hermanos de su tipo
:empty	Elementos que no tienen hijos
:nth-child(n)	Enésimo elemento hijo
:nth-last-child(n)	Enésimo elemento hijo contando desde el último
:nth-of-type(n)	Enésimo hermano de su tipo
:nth-last-of-type(n)	Enésimo hermano de su tipo comenzando desde el último

También sirve para seleccionar por estado del elemento

```
selector:pseudo-class {  
  property: value;  
}
```

Pseudo-clase	Descripción
:link	No visitado por el usuario
:visited	Visitado por el usuario
:hover	Modifica el estilo cuando un elemento apuntador pasa por encima
:active	Se activa cuando el usuario pulsa el elemento
:focus	Se activa cuando tiene el foco sobre el elemento

## Ejercicio 38

1. Crea un documento HTML con un enlace a otra página cualquiera.
2. Según los siguientes estados del elemento enlace anterior, dale un estilo (por ejemplo un color) distinto:
  - a. Estado para cuando el enlace no haya sido aún visitado
  - b. Estado para cuando el enlace ya haya sido visitado
  - c. Estado para cuando el puntero del ratón se coloca encima del enlace
3. Intenta darle una apariencia de botón al enlace

### Referencias:

- [https://www.w3schools.com/cssref/css\\_ref\\_pseudo\\_classes.php](https://www.w3schools.com/cssref/css_ref_pseudo_classes.php)
- <https://www.eniun.com/pseudo-clases-pseudo-elementos-css/>

## Ejercicio 39

Utiliza el HTML de base proporcionado [aquí](#). (copia sólo el HTML y haz tú el CSS)

Añade las siguientes reglas CSS utilizando pseudo-clases:

- Estiliza **el primer hijo** de cada lista (`u1`) con un color de fondo diferente.
- Estiliza **el último hijo** de cada lista con texto en negrita.
- Aplica un estilo único al **tercer elemento** de la lista de tareas.
- Cambia el color de texto de los **hermanos inmediatos** del elemento que contiene "Pagar facturas".
- Agrega un borde al **segundo hijo** de la lista de contactos.

**Prueba tus estilos.** Comprueba cómo las pseudo-clases afectan a los elementos específicos.

Añaden estilos a una parte específica del documento

```
selector::pseudo-element {  
  property: value;  
}
```

Pseudo-elemento	Descripción
::first-line	Primera línea de texto de un elemento
::first-letter	Primera letra de la primera línea de texto de un elemento
::before	Añade contenido al principio del documento
::after	Añade contenido al final del documento
::selection	Coge la porción del texto que se está seleccionando por el usuario



## Ejercicio 40

A partir del [siguiente HTML base](#)

Añade las siguientes reglas CSS utilizando pseudo-elementos:

- **Primera línea:** Cambia el color de la primera línea de cada párrafo a azul y usa un tamaño de fuente más grande.
- **Primera letra:** Aplica un estilo decorativo a la primera letra de cada párrafo (por ejemplo, color rojo, fuente más grande y negrita).
- **Antes del documento (::before):** Agrega un mensaje decorativo antes del contenido principal, por ejemplo, "¡Prepárate para aprender CSS!".
- **Después del documento (::after):** Añade un mensaje motivador después del contenido principal, como "¡Sigue practicando y mejora cada día!".
- **Selección de texto (::selection):** Cambia el color de fondo y el color de texto cuando se selecciona cualquier parte del documento.

Experimenta con diferentes estilos y prueba los efectos en el navegador.

Referencias:

- <https://www.eniun.com/pseudo-clases-pseudo-elementos-css/>
- [https://www.w3schools.com/css/css\\_pseudo\\_elements.asp](https://www.w3schools.com/css/css_pseudo_elements.asp)
- <https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-elements>

## Ejercicio 41

Crea un pequeño menú con tres iconos que representen las siguientes acciones: *Inicio*, *Buscar* y *Configuración*.

Usa una imagen sprite que contenga todos los iconos en un solo archivo y emplea CSS para mostrar solo la parte correspondiente de la imagen en cada botón.

Opcional:

- Intenta hacer algún efecto cuando el ratón pase por los iconos



Referencia: [https://www.w3schools.com/css/css\\_image\\_sprites.asp](https://www.w3schools.com/css/css_image_sprites.asp)

Cómo se muestra un documento en función de dónde se visualice

- Si se cumple la condición, se aplican las reglas
- Sintaxis:

@media	screen	(min-width: 320px)	and	(max-width: 768px)
AT-RULE	MEDIA TYPE	MEDIA FEATURE	OPERATOR	MEDIA FEATURE

► @media

► Media types

Device

► Media features

Viewport

► Operators

and, or, not, only

# Media queries - Ejemplos



```
@media print {  
  body {  
    font-size: 10pt;  
  }  
}  
  
@media screen {  
  body {  
    font-size: 13px;  
  }  
}  
  
@media screen, print {  
  body {  
    line-height: 1.2;  
  }  
}  
  
@media only screen and (min-width: 320px) and (max-width: 480px) and  
(resolution: 150dpi) {  
  body {  
    line-height: 1.4;  
  }  
}
```

## Media types

### **all**

Suitable for all devices.

### **braille**

Intended for braille tactile feedback devices.

### **embossed**

Intended for paged braille printers.

### **handheld**

Intended for handheld devices (typically small screen, limited bandwidth).

### **print**

Intended for paged material and for documents viewed on screen in print preview mode. Please consult the section on [paged media](#) for information about formatting issues that are specific to paged media.

### **projection**

Intended for projected presentations, for example projectors. Please consult the section on [paged media](#) for information about formatting issues that are specific to paged media.

### **screen**

Intended primarily for color computer screens.

### **speech**

Intended for speech synthesizers. Note: CSS2 had a similar media type called 'aural' for this purpose. See the appendix on [aural style sheets](#) for details.

### **tty**

Intended for media using a fixed-pitch character grid (such as teletypes, terminals, or portable devices with limited display capabilities). Authors should not use [pixel units](#) with the "tty" media type.

### **tv**

Intended for television-type devices (low resolution, color, limited-scrollability screens, sound available).

## Ejercicio 42

Escribe un fichero HTML básico que conste de una cabecera y un cuerpo. Dentro del cuerpo, añade tres elementos de tipo párrafo, tres imágenes y dos enlaces. Luego, crea una hoja de estilo y utiliza el elemento @media para aplicar el siguiente estilo:

- Los párrafos deben tener una fuente en *itálica*, **negrita** y un tamaño de 18 px en cualquier dispositivo.
- Las imágenes deben aparecer centradas sólo cuando se visualicen en una pantalla.
- Tanto los párrafos como la imagen deben estar alineados a la izquierda en todos los dispositivos, excepto cuando la imagen se muestre en una pantalla.
- Los enlaces siempre tienen que aparecer centrados.

## @import

Si queremos especificar las media queries mediante otros ficheros CSS como alternativa a hacerlo en HTML con el tag link.

Sintaxis: ***@import url(nombreFichero.css) medio;***

```
@import url("fineprint.css") print;  
@import url("bluish.css") print, screen;  
@import "common.css" screen;  
@import url("landscape.css") screen and (orientation: landscape);
```

Se escriben al principio del fichero CSS

## Ejercicio 43

Crea un fichero HTML básico que incluya una cabecera y un cuerpo. En el cuerpo, añade al menos dos elementos de tipo párrafo y una imagen. Luego, crea una hoja de estilo externa en CSS y utiliza la regla @import para vincularla al archivo HTML. En la hoja de estilo, aplica los siguientes estilos:

- Los párrafos deben tener un tamaño de fuente de 16 px y un color de texto gris.
- La imagen debe tener un borde sólido de 2 px de color negro.
- Ajusta los márgenes y el espaciado para que los elementos estén distribuidos uniformemente en la página.



## Ejercicio 44

1. **Crea el archivo HTML (`index.html`)**
  - Debe contener:
    - Una cabecera con el logotipo del sitio y un menú de navegación.
    - Al menos tres enlaces de menú (`Inicio`, `Servicios`, `Contacto`).
    - Una sección principal de contenido con un título y un párrafo.
2. **Crea el archivo CSS (`styles.css`)**
  - Diseña el menú de navegación con estilos básicos:
    - En pantallas grandes (más de 768px), el menú debe mostrarse en una fila horizontal, con los enlaces alineados a la derecha.
    - En pantallas pequeñas (menos de 768px), el menú debe apilarse en una columna y ocupar el ancho completo de la pantalla.
  - Adapta el diseño para dispositivos en orientación **vertical** (portrait) y **horizontal** (landscape):
    - Cambia el color de fondo del menú dependiendo de la orientación.
  - Usa únicamente **media queries** para lograr la responsividad.

### Retos adicionales

1. Añade un icono (usando Unicode o una imagen) al lado de cada enlace del menú para practicar más con estilos.
2. Haz que el logotipo cambie de tamaño según el ancho de la pantalla.
3. Integra un efecto de transición suave cuando el color de fondo del menú cambie.

**Ejercicio:** Elaboramos una lista de validadores y herramientas útiles para desarrollar CSS y las compartimos en clase. ⇒ Wiki

Ejemplo: <https://jigsaw.w3.org/css-validator/>



¿Qué son las buenas prácticas?

## Ejemplo:

- Los selectores deben de nombrarse en minúsculas (además no pueden empezar por número o carácter especial). Debe usarse el “-” en lugar del “\_”

## Ejercicio 45

1. **Lee las buenas prácticas** mencionadas en los artículos sobre [nombres de clases CSS](#) y [código limpio en CSS](#).
2. **Crea una página HTML** y un archivo CSS aplicando las siguientes recomendaciones, con ejemplos prácticos:

### Parte 1: Nombres de Clases

- **Crea ejemplos que demuestren:**
  - **Nombres semánticos:** Usa clases como `.main-header` en lugar de `.header1`.
  - **Evita redundancias:** En lugar de `.button-red-button`, usa `.button-red`.
  - **Agrupación lógica:** Usa prefijos o estructuras que reflejen bloques o componentes, como `nav-primary` o `card-item`.

## Parte 2: Organización del Código

Divide el archivo CSS en secciones con comentarios, como:

```
/* Global Styles */  
  
body, h1, p { ... }  
  
/* Header Styles */  
  
.main-header { ... }
```

## Parte 3: Buenas Prácticas

- **Reutilización de Estilos:** Diseña una clase genérica como `.btn` que pueda aplicarse a varios botones con modificaciones específicas (`.btn-primary`, `.btn-secondary`).
- **Optimización del Código:** Aplica reglas de CSS abreviadas, como `margin: 10px 15px;` en lugar de definir cada lado individualmente.
- **Separación de Responsabilidades:** Crea un archivo `reset.css` o `base.css` para estilos generales, y otro `main.css` para los específicos de la página.

## Parte 4: Validación

- Incluye un paso para validar el código en [CSS Validator](#) y menciona cómo mejorarías tu archivo basándote en los errores encontrados.