FUNDAMENTOS DE CSS

Cascading Style Sheets

Dando vida y color a la web

1. ¿QUÉ ES CSS?

Cascading **S**tyle **S**heets (Hojas de Estilo en Cascada).

- **HTML** = Estructura (El esqueleto 💀)
- **CSS** = Presentación (La ropa y el estilo 🎨)

"En Cascada": Los estilos se aplican siguiendo un orden de prioridad y herencia.

```
/* Sintaxis Básica */
selector {
  propiedad: valor;
}

h1 {
  color: blue;
  font-size: 20px;
}
```

2. PROPIEDADES PRINCIPALES

CSS controla todo el aspecto visual. Aquí las más comunes:

🎨 Color y Fondo

```css
/* Texto */
color: red;             /* Nombre */
color: #FF0000;         /* Hex */
color: rgb(255,0,0);    /* RGB */

/* Fondo */
background-color: #f5f5f5;
```

🔤 Tipografía

```css
font-family: Arial, sans-serif;
font-size: 16px;        /* Tamaño */
font-weight: bold;      /* Grosor */
text-align: center;     /* Alineación */
text-decoration: none; /* Quitar subrayado */
```

> **Tip Google Fonts:** Añade el `<link>` en tu HTML y úsala en CSS con `font-family: 'Roboto', sans-serif;`.

3. ¿CÓMO VINCULAR CSS?

Existen 3 formas, pero **solo una es la recomendada**.

1. Externo (✅)

```
<head>
 <link rel="stylesheet" href="style.css">
</head>
```

Reutilizable y limpio.

2. Interno

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

Solo para esa página.

3. Inline (❌)

```
<p style="color:red">
    Hola
</p>
```

Difícil de mantener.

4. SELECTORES BÁSICOS

¿A quién le aplicamos el estilo?

Selector	Símbolo	Ejemplo	¿Qué selecciona?
Universal	`*`	`* { margin: 0; }`	**Todo** (Resets).
Etiqueta	(nada)	`p { color: red; }`	Todos los elementos de ese tipo.
Clase	`.`	`.boton { ... }`	Grupos de elementos (**Reutilizable**).
ID	`#`	`#header { ... }`	Elemento **único** (Prioridad alta).

> 💡 **Regla:** Usa **Clases** para casi todo. Usa **ID** solo para cosas únicas.

5. COMBINADORES (RELACIONES)

Para seleccionar elementos según dónde están.

Descendiente (Espacio)	**Hijo Directo (>)**	**Hermano Adyacente (+)**	**Hermano General (~)**
`div p`	`div > p`	`h2 + p`	`h2 ~ p`
Cualquier `<p>` dentro de `<div>` (hijos o nietos).	Solo hijos directos (no nietos).	El `<p>` que va **inmediatamente** después del `<h2>`.	Todos los `<p>` que van después del `<h2>`.

6. PSEUDO-CLASES

Estados especiales de los elementos.

```
/* Cuando pasas el ratón por encima de un enlace */
a:hover {
  color: orange;
  text-decoration: underline;
}

/* El primer elemento de una lista */
li:first-child {
  font-weight: bold;
}

/* Elementos pares (útil para tablas) */
tr:nth-child(even) {
  background-color: #f2f2f2;
}
```

QUIZ CSS 🎨

Pregunta 1

¿Cuál es la forma recomendada de incluir CSS en un proyecto real?

A) Estilos en línea (`style="..."`).
B) Archivo externo (`<link>`).
C) Estilos internos (`<style>`).
D) Importando desde JavaScript.

✅ Respuesta 1: B) Archivo Externo

Es la única forma que permite **separar el contenido (HTML) del diseño (CSS)**, haciendo que el código sea reutilizable en múltiples páginas y fácil de mantener.

Pregunta 2

Si quiero aplicar un estilo a varios elementos diferentes en mi página, ¿qué selector debo usar?

A) ID (`#`)

B) Etiqueta (`h1`)

C) Clase (`.`)

D) Universal (`*`)

✅ Respuesta 2: C) Clase (`.`)

Las **clases** están diseñadas para ser reutilizables.

- Los **ID** deben ser únicos (uno por página).
- Las **etiquetas** afectan a todos los elementos de ese tipo.

Pregunta 3

¿Qué hace el selector `div > p`?

A) Selecciona todos los párrafos dentro de un div.

B) Selecciona el párrafo que está antes del div.

C) Selecciona solo los párrafos que son **hijos directos** del div.

D) Selecciona todos los divs dentro de un párrafo.

✅ Respuesta 3: C) Hijo Directo

El símbolo `>` indica una relación directa (Padre > Hijo). Si el párrafo estuviera dentro de otro contenedor dentro del div (un nieto), este selector no lo afectaría.

EL MODELO DE CAJA (BOX MODEL)

La base del diseño web 📦

"Todo en HTML es una caja rectangular"

1. EL CONCEPTO

Imagina que quieres enviar unos **zapatos** por correo:

1. **Content (Contenido):** Los zapatos. Lo que realmente importa.
2. **Padding (Relleno):** El papel de burbujas que protege los zapatos dentro de la caja.
3. **Border (Borde):** La caja de cartón en sí misma.
4. **Margin (Margen):** El espacio entre tu caja y las otras cajas en el camión de reparto.

2. PARTES DEL BOX MODEL

1. Content

Es el elemento real (texto, imagen).

```
width: 100px;
height: 30px;
```

3.Padding

Espacio **interior** (transparente). Separa el contenido del borde.

```
padding: 20px; /* Todos los lados */
padding-top: 10px;
```

2. Border

La línea que rodea la caja.

```
/* Grosor - Estilo - Color */
border: 2px solid blue;
```

3. Margin

Espacio **exterior**. Separa la caja de otras cajas.

```
margin: 20px;
margin: 0 auto; /* Centrar */
```

3. PADDING VS MARGIN (DIFERENCIAS CLAVE)

A menudo se confunden, pero son opuestos.

Padding (Interior)

el padding

también se pinta

Aumenta el tamaño

Margin (Exterior)

transparente

separar

4. EL PROBLEMA MATEMÁTICO (`CONTENT-BOX`)

Por defecto, HTML funciona de una manera extraña:

```
.caja {
  width: 300px;
  padding: 20px;
  border: 5px solid black;
}
```

¿Cuánto mide la caja en total?

$$AnchoTotal = Width + Padding + Border$$

$$300 + (20 \times 2) + (5 \times 2) = \mathbf{350px}$$

¡Es más grande de lo que pediste! Esto rompe los

4.1. La Solución: `border-box`

Para evitar cálculos matemáticos, usamos `box-sizing: border-box`.

❌ `content-box` (Default)

El `width` es solo el contenido. Todo lo demás se suma. *Resultado: Cajas impredecibles.*

✅ `border-box` (Recomendado)

El `width` es el ancho **total** (borde a borde). El navegador ajusta el contenido automáticamente. *Resultado: Cajas exactas.*

4.2. El "Reset" Universal

El truco que usan todos los profesionales. Pon esto **siempre** al principio de tu CSS:

```
/* Selector Universal (*) */
* {
  margin: 0;  /* Elimina márgenes extraños por defecto */
  padding: 0;
  box-sizing: border-box;  /* Activa el cálculo fácil de medidas */
}
```

Así, si dices `width: 300px`, la caja medirá exactamente **300px**. Fin del problema.

QUIZ: BOX MODEL 📦

Pregunta 1

¿Qué propiedad CSS controla el espacio INTERIOR entre el contenido y el borde?

A) Margin

B) Border

C) Padding

D) Spacing

✅ Respuesta 1: C) Padding

El **Padding** es el relleno interno. Recuerda:

- **Padding** = Dentro (como el relleno de un cojín).
- **Margin** = Fuera (como tu espacio personal).

Pregunta 2

Si tengo una caja con `width: 100px`, `padding: 10px` y `border: 1px`, ¿cuál es su ancho TOTAL real por defecto (content-box)?

A) 100px

B) 111px

C) 122px

D) 102px

✅ Respuesta 2: C) 122px

$$100(Ancho) + 20(Padding \times 2) + 2(Borde \times 2) = 122px$$

Para que mida 100px exactos, necesitaríamos usar `box-sizing: border-box`.

Pregunta 3

¿Qué propiedad hace que el borde y el padding estén INCLUIDOS dentro del ancho total (`width`)?

A) `box-sizing: content-box`

B) `box-model: fix`

C) `display: block`

D) `box-sizing: border-box`

✅ Respuesta 3: D) `box-sizing: border-box`

Esta propiedad cambia el modo de cálculo para que el ancho definido sea el ancho final del elemento, facilitando mucho la maquetación.

CSS LAYOUT

Estructura y Posicionamiento

Dejando atrás el flujo aburrido de arriba a abajo

1. ¿QUÉ ES CSS LAYOUT?

Ya dominas los colores y las cajas. Ahora toca **organizarlo todo**.

Definición: Conjunto de técnicas para organizar y posicionar elementos. Define la **estructura visual** de la web.

> "Probablemente lo más complejo de aprender al principio, pero lo más potente."

¿Por qué es vital?

- ✅ Diseños profesionales.

2. LAS 5 HERRAMIENTAS DEL LAYOUT

No hay una sola forma de hacerlo. Depende de qué necesites:

1. Display

El comportamiento básico. ¿Es un bloque? ¿Es texto en línea? ¿Está oculto? *(block, inline, none)*

2. Position

Ubicación precisa. Mover cosas fuera del flujo normal. *(absolute, fixed, sticky)*

3. Flexbox

Layout 1D. Ideal para alinear elementos en una **fila** O una **columna**.

4. Grid

5. Media Queries

CSS DISPLAY

Controlando el flujo del documento

¿Línea, Bloque o Híbrido?

1. LOS 3 COMPORTAMIENTOS CLAVE

La propiedad `display` decide cómo se relaciona una caja con las demás.

1. `block` **(El Egoísta):** Quiere toda la línea para él. Empuja a los demás abajo. *(Ej:* `<div>` , `<p>` , `<h1>` *)*
2. `inline` **(El Sociable):** Fluye con el texto. Ocupa lo mínimo posible. *(Ej:* `<span>` , `<a>` , `<strong>` *)*
3. `inline-block` **(El Híbrido):** Se pone en fila (como inline) pero acepta tamaño (como block). *(Ej:* `<img>` , `<button>` *)*

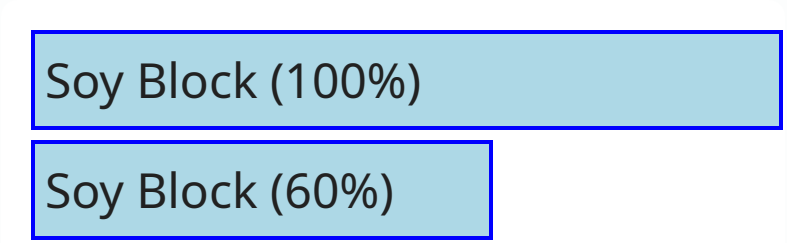
2. LA "CHULETA" DEFINITIVA

Propiedad	BLOCK 🟥	INLINE 🟦	INLINE-BLOCK 🟪
Nueva línea	✅ Sí	❌ No	❌ No
Width / Height	✅ Sí	❌ Ignorado	✅ Sí
Márgenes	✅ Todos	⚠️ Solo lados	✅ Todos

3. DISPLAY: BLOCK

Ocupa el 100% del ancho. Salto de línea obligado.

```
.caja {
  display: block;
  width: 60%; /* Ocupa lo que digas */
  height: 50px;
}
```



- ✅ Respeta medidas.
- ⬇️ Empuja elementos.

4. DISPLAY: INLINE

Solo ocupa su contenido. No rompe la línea.

```
.texto {
  display: inline;
  /* width: 200px; ❌ NO FUNCIONA */
  /* margin-top: 50px; ❌ NO FUNCIONA */
}
```

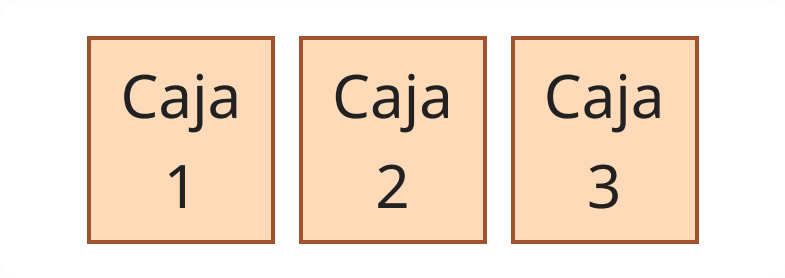
- ❌ Ignora medidas y márgenes verticales.
- ✅ Fluye con el texto.

Texto normal y inline. Aunque le ponga width/height, lo ignora y sigue fluyendo.

5. DISPLAY: INLINE-BLOCK

Se alinean, pero tienen cuerpo.

```
.boton {
  display: inline-block;
  width: 100px; /* ✅ SÍ FUNCIONA */
  height: 40px;
}
```



- Ideal para: **Menús, Botones y Tarjetas**.

QUIZ RÁPIDO 

Pregunta 1

Quiero resaltar una palabra en un texto con color de fondo. ¿Qué uso?

A) `block`
B) `inline`
C) `none`

✅ Respuesta: B) `inline`

Como la etiqueta `<span>`. Permite dar estilo sin romper la frase.

Pregunta 2

Quiero poner 3 cajas de 200px una al lado de la otra. ¿Qué uso?

A) `block` (Las pone debajo)
B) `inline` (Ignora los 200px)
C) `inline-block` (Las pone al lado y respeta el tamaño)

✅ Respuesta: C) `inline-block`

Es el híbrido perfecto para maquetar elementos horizontales con dimensiones controladas.

CSS POSITION

Controlando la ubicación exacta

Moviendo cajas fuera del flujo normal

1. ¿QUÉ ES `POSITION`?

Permite sacar un elemento de su lugar natural. Para moverlo, necesitas **dos pasos**:

1. **Tipo:** `position: absolute` (u otro).
2. **Coordenadas:** `top`, `right`, `bottom`, `left`.

```
.caja {
  position: absolute; /* Paso 1 */
  top: 50px;          /* Paso 2 */
  left: 20px;
}
```

> ⚠️ **Nota:** Las coordenadas **NO funcionan** en `static`.

2. LOS 5 TIPOS (RESUMEN)

1. Static
Por defecto. Flujo normal. No se mueve.

2. Relative
Se mueve respecto a sí mismo. **Reserva su hueco original**.

3. Absolute
Flota libremente. Busca al **padre posicionado**.

4. Fixed
Fijo a la **pantalla**. No le afecta el scroll.

5. Sticky
Híbrido. Normal hasta que haces scroll y se "pega".

3. STATIC VS RELATIVE

📦 Static (Default)

El flujo aburrido. Uno debajo del otro. *(Las coordenadas top/left no hacen nada).*

👻 Relative

Se mueve, pero deja un **hueco vacío**.

```
.caja {
  position: relative;
  left: 30px; top: 20px;
}
```

- **Uso:** Pequeños ajustes o para controlar hijos absolutos.

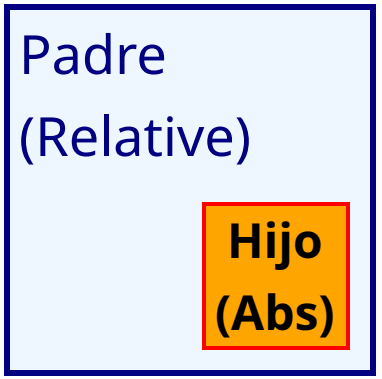
4. POSITION: ABSOLUTE

El elemento **flota** y pierde su espacio.

La Regla de Oro:

> "El hijo `absolute` se coloca respecto al padre `relative`".

Si el padre no tiene posición, se irá a la ventana del navegador.



5. EFECTOS DE SCROLL (FIXED VS STICKY)

⚓ Fixed (Ancla)

Se pega a la **Ventana**.

- **Ej:** Botón "Volver arriba" o Chat.
- **Espacio:** No ocupa espacio (flota).

🍬 Sticky (Pegajoso)

Se pega al **Contenedor**.

- **Ej:** Encabezados de tabla.
- **Espacio:** Sí ocupa espacio inicial.

Resumen Final

Valor	¿Ocupa Espacio?	Referencia de movimiento
Static	✅ Sí	Flujo normal (No se mueve)
Relative	✅ Sí (Hueco vacío)	Su posición original
Absolute	❌ No (Flota)	Padre posicionado más cercano
Fixed	❌ No (Flota)	Ventana (Viewport)
Sticky	✅ Sí	Contenedor padre (al hacer scroll)

QUIZ RÁPIDO 📍

Pregunta 1

Por defecto, todos los elementos son...

A) `relative`
B) `absolute`
C) `static`

✅ Respuesta: C) `static`

Es el comportamiento natural.

Pregunta 2

Para poner un icono en la esquina de una tarjeta, la combinación ganadora es:

A) Padre `static` + Hijo `absolute`
B) Padre `relative` + Hijo `absolute`
C) Ambos `absolute`

✅ Respuesta: B) Padre `relative` + Hijo `absolute`

El padre define el límite y el hijo se mueve libremente dentro.

Pregunta 3

Quiero un menú que se quede fijo arriba al bajar la página.

A) `sticky` o `fixed`
B) `absolute`
C) `relative`

✅ Respuesta: A) `sticky` o `fixed`

Depende de si quieres que esté fijo desde el principio (`fixed`) o solo al llegar arriba (`sticky`).

CSS FLEXBOX

Flexible Box Layout

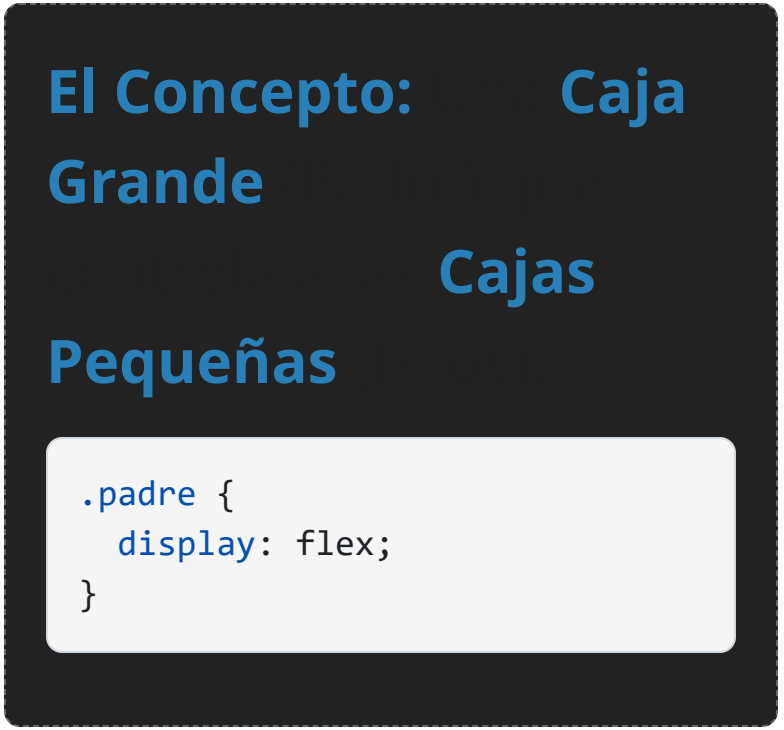
El fin de las pesadillas de alineación

1. ¿QUÉ ES FLEXBOX?

Es un sistema de diseño unidimensional (**Fila** O **Columna**).

Adiós a los `floats`.
Flexbox facilita:

- ✅ Alinear elementos (vertical/horizontal).
- ✅ Distribuir el espacio sobrante.
- ✅ Ordenar elementos sin cambiar el HTML.



2. LOS DOS PROTAGONISTAS

Todo en Flexbox se basa en la relación Padre-Hijo.

🏗️ El Contenedor (Padre)

El elemento con `display: flex`. Define las **reglas del juego**.

- ¿En fila o columna?
- ¿Alineados al centro o a los lados?

📦 Los Ítems (Hijos)

Los elementos directos dentro. Siguen las reglas, pero **pueden adaptarse**.

- ¿Crecer para ocupar espacio? (`grow`)

3. PROPIEDADES DEL CONTENEDOR

El "jefe" del layout.

Propiedad	Descripción	Valores típicos
`flex-direction`	¿Hacia dónde van?	`row` (fila), `column` (columna)
`justify-content`	Alineación **Eje Principal**	`center`, `space-between`, `flex-start`
`align-items`	Alineación **Eje Secundario**	`center`, `stretch`, `flex-end`
`flex-wrap`	¿Saltan de línea?	`nowrap` (apretados), `wrap` (saltan)
`gap`	Espacio entre ítems	`20px`, `1rem`

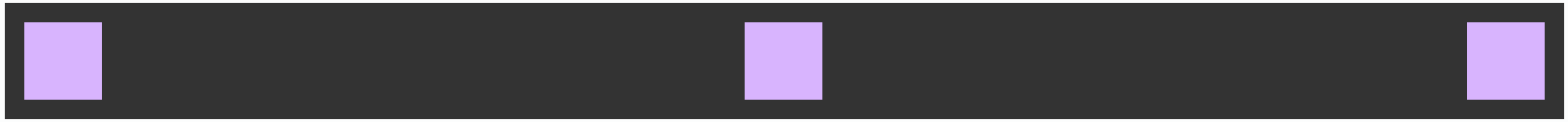
4. VISUALIZANDO ALINEACIONES (EJE X)

`justify-content` : Controla la distribución horizontal (en modo fila).

`flex-start` **(Default)**

`center`

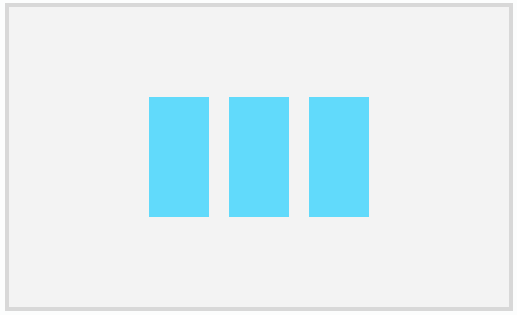
`space-between` **(Ideal menús)**



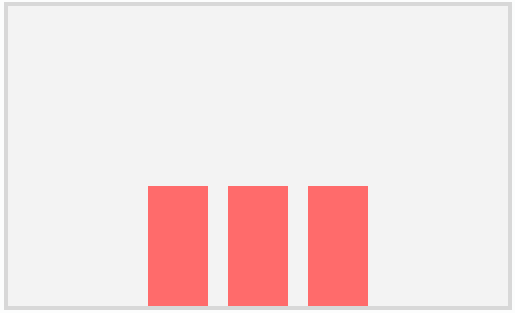
5. VISUALIZANDO ALINEACIONES (EJE Y)

`align-items` : Controla la distribución vertical.

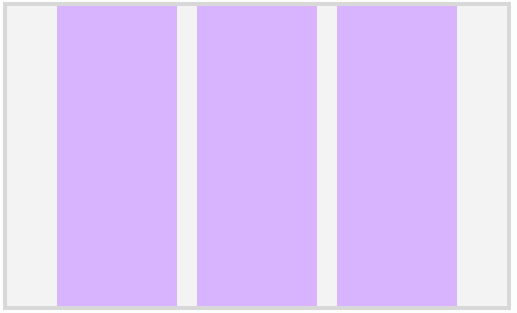
center



flex-end



stretch



6. PROPIEDADES DE LOS HIJOS (ÍTEMS)

No solo el padre manda. Los hijos tienen personalidad.

- `flex-grow`: "Yo crezco". Si sobra espacio, este ítem se estira para ocuparlo.
  - `flex-grow: 1` (todos crecen igual).
- `flex-shrink`: "Yo encojo". Si falta espacio, este ítem se hace pequeño para no romper la caja.
- `order`: "Yo cambio de sitio". Permite cambiar el orden visual (1, 2, 3) sin tocar el HTML.

QUIZ: FLEXBOX 💪

Pregunta 1

Para activar Flexbox, ¿qué propiedad aplico y a quién?

A) `display: flex` a los hijos.
B) `position: flex` al padre.
C) `display: flex` al contenedor (padre).
D) `flex-align: center` al contenedor.

✅ Respuesta 1: C) `display: flex` al padre

Al aplicar esto al contenedor, todos sus hijos directos se convierten mágicamente en "flex items" y empiezan a alinearse en fila por defecto.

Pregunta 2

Quiero alinear mis ítems horizontalmente en el CENTRO del contenedor. ¿Qué propiedad uso?

A) `align-items: center`

B) `justify-content: center`

C) `text-align: center`

D) `margin: center`

✅ Respuesta 2: B) `justify-content: center`

Por defecto (en dirección fila), `justify-content` controla el eje principal (horizontal).

- `align-items` controlaría el vertical.

Pregunta 3

Si quiero que los elementos se distribuyan dejando el máximo espacio posible ENTRE ellos, pegándose a los bordes:

A) `justify-content: center`

B) `justify-content: space-around`

C) `justify-content: space-between`

✅ Respuesta 3: C) `space-between`

- **Space-between:** Pegados a los bordes, espacio en medio.
- **Space-around:** Espacio alrededor de cada uno (no pegan a los bordes).
- **Space-evenly:** Espacio exactamente igual por todos lados.

CSS GRID

Layout Bidimensional

El sistema de diseño más potente de la web

1. ¿QUÉ ES CSS GRID?

Es un sistema para organizar elementos en **filas Y columnas** a la vez.

La diferencia clave:

- **Flexbox:** 1 Dimensión (Fila O Columna).
- **Grid:** 2 Dimensiones (Filas Y Columnas).

Analogía: Piensa en Grid como una **Tabla de Excel** supervitaminada donde

Conceptos Básicos:

1. **Contenedor:** El Padre (`display: grid`).
2. **Ítems:** Los Hijos (Celdas).
3. **Líneas:** Las guías numeradas.

2. PROPIEDADES DEL PADRE (CONTENEDOR)

Aquí definimos la estructura de la cuadrícula.

Propiedad	Descripción	Ejemplo
`display`	Activa el grid.	`grid`
`grid-template-columns`	Define el ancho de las **columnas**.	`100px 200px`
`grid-template-rows`	Define el alto de las **filas**.	`50px auto`
`gap`	Espacio entre huecos (calles).	`20px`

La unidad mágica: `fr` (fracción)

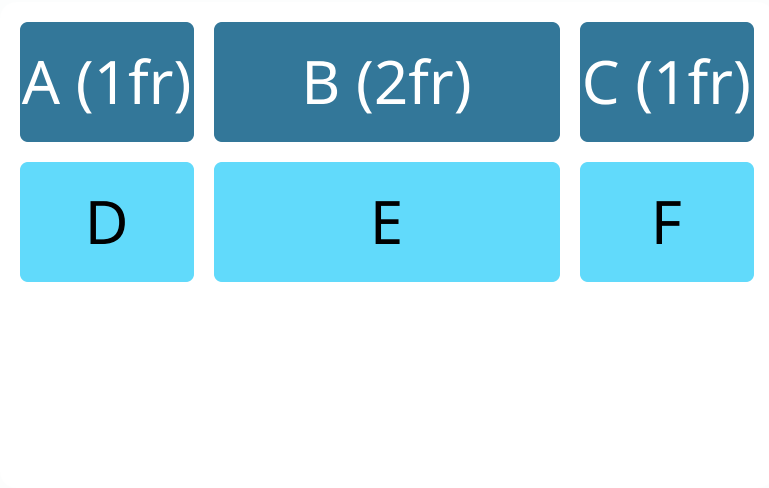
En lugar de `%` o `px`, usamos `fr` para repartir el espacio disponible.

- `1fr 1fr` = Dos columnas iguales (50% cada una).
- `1fr 2fr` = La segunda es el doble de grande que la primera.

3. EJEMPLO VISUAL: COLUMNAS Y `FR`

Vamos a crear 3 columnas: `1fr`, `2fr` y `1fr`.

```
.contenedor {
  display: grid;
  /* 3 columnas */
  grid-template-columns: 1fr 2fr 1fr;
  /* 2 filas de 60px */
  grid-template-rows: 60px 60px;
  gap: 10px;
}
```



4. PROPIEDADES DE LOS HIJOS (ÍTEMS)

Por defecto, los ítems ocupan 1 celda. Pero podemos hacer que se **expandan**.

Las líneas del Grid están numeradas (empiezan en 1).

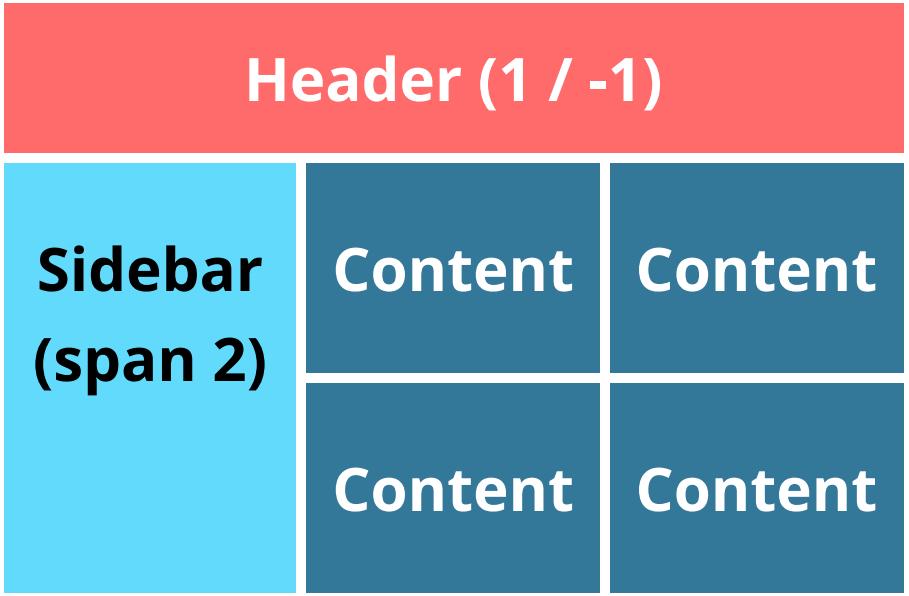
- `grid-column: start / end` : De qué línea a qué línea va.
- `grid-row: start / end` : De qué fila a qué fila va.

```
.item-grande {
  /* Empieza en la línea 1 y acaba en la 3 (ocupa 2 columnas) */
  grid-column: 1 / 3;

  /* Ocupa 2 filas */
  grid-row: span 2;
}
```

5. EJEMPLO: LAYOUT COMPLEJO

```
.grid {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  grid-template-rows: 50px 100px;
  gap: 5px;
}
.full {
  grid-column: 1 / -1; /* -1 es el final */
}
.alto {
  grid-row: span 2;
}
```



RESUMEN: ¿GRID O FLEXBOX?

No son enemigos, son **complementarios**.

Flexbox (1D)

Úsalo para componentes lineales o pequeños.

- Menú de navegación.
- Lista de etiquetas.
- Alinear iconos y texto.
- **Contenido** dicta el layout.

Grid (2D)

Úsalo para la estructura grande de la página.

- Layout principal (Header + Sidebar + Main).
- Galerías de fotos complejas.
- **Layout** dicta el

QUIZ: CSS GRID 🏁

Pregunta 1

¿Qué propiedad define cuántas columnas tiene mi Grid y de qué tamaño son?

A) `grid-columns-count` B) `grid-template-columns` C) `display: columns` D) `grid-size`

✅ Respuesta 1: B) `grid-template-columns`

Ejemplo: `grid-template-columns: 100px 1fr;` crea dos columnas, una fija y otra flexible.

Pregunta 2

¿Qué significa la unidad `1fr`?

A) 1 frame (un fotograma).

B) 1 fracción del espacio disponible restante.

C) 1 frecuencia de repetición.

D) 100 píxeles fijos.

✅ Respuesta 2: B) 1 Fracción

Es una unidad flexible. Si tienes `1fr 1fr 1fr`, el espacio se divide en 3 partes iguales. Si tienes `1fr 2fr`, el espacio se divide en 3, y la segunda columna se lleva 2 partes.

Pregunta 3

Quiero que un ítem ocupe TODO el ancho del grid (de principio a fin), sin importar cuántas columnas haya.

A) `width: 100%`

B) `grid-column: all`

C) `grid-column: 1 / -1`

D) `grid-row: full`

✅ Respuesta 3: C) `grid-column: 1 / -1`

En CSS Grid, el **1** es la primera línea y el **-1** representa siempre la **última línea**, independientemente de cuántas columnas hayas definido.

TRANSICIONES CSS (TRANSITION)

Hacen que los cambios entre dos estados (ej. :hover) sean graduales.

Ejemplo básico: Botón

```
.boton {
  background: #379;
  transition: background 0.3s, transform 0.2s; /* Propiedad de transición */
}
.boton:hover {
  background: #fa7;
  transform: scale(1.1); /* Efecto al pasar el mouse */
}
```

> **Resultado:** El color de fondo y el tamaño cambian de forma suave al pasar el mouse

Propiedades clave	Descripción
`transition-property`	Qué propiedad se anima.
`transition-duration`	Cuánto tiempo dura la animación.
`transition-timing-function`	Velocidad/Interpolación (ej. `ease`).

ANIMACIONES CSS (ANIMATION)

Permiten crear movimientos y efectos más complejos, controlando varios pasos y repeticiones.

Ejemplo básico: Movimiento

```css
.cuadro {
  animation: mover 2s infinite alternate; /* Aplica la animación */
}

@keyframes mover {
  from { transform: translateX(0); }
  to { transform: translateX(200px); }
}
```

> **Resultado:** El cuadro se mueve suavemente de izquierda a derecha y vuelve a repetir el ciclo

Propiedades clave	Descripción
`animation-name`	Nombre de la animación (`@keyframes`).
`animation-duration`	Duración de un ciclo completo.
`animation-iteration-count`	Número de repeticiones (ej. `infinite`).

¿POR QUÉ USAR ANIMACIONES Y TRANSICIONES?

- Mejoran la **experiencia del usuario**.
- Destacan **acciones importantes** (ej. feedback al hacer click).
- Hacen la interfaz más **atractiva** y moderna.
- Proporcionan **claridad visual** sobre los cambios de estado.