CSS

Formas de conectar con HTML

1. Inline: directamente en el elemento HTML atributo estilo
Hola mundo

2. Interno, dentro de HTML: estilos en la etiqueta `<style>` dentro del `<head>`.

```
<head>
  <style>
    p {
      propiedad: valor;
      color: blue;
      font-size: 20px;
    }
  </style>
</head>
```

3. Archivo externo: `.css` y enlazar desde HTML dentro del <head>
HTML <link rel="stylesheet" href="estilos.css">
CSS p {
color: green;…

Comments /* Esto es un comentario en CSS */

a
atributo: más correctamente llamado **propiedad**, define qué aspecto se va a modificar del elemento. (color, background..)
La manera en que un elemento se comporta, agrega información.

animación:
Transiciones / transition: se puede aplicar a cualquier propiedad de css y también a todas simultáneamente. Para que funcionen, debe haber una diferencia entre el estilo inicial y el estilo final (un "antes" y un "después"). Su función es que el cambio sea gradual en la cantidad de tiempo especifico.

Propiedades:

- **transition-property**: indica qué propiedad se va a animar (ej. background-color, transform, width, etc.). Puede usarse all para aplicar a todas las propiedades animables.

- **transition-duration**: define cuánto dura la transición. Se expresa en segundos (s) o milisegundos (ms).

- **transition-timing-function**: controla la curva de aceleración del cambio:
  - linear: velocidad constante (uniforme)
  - ease: cambio suave (valor por defecto).
  - ease-in: empieza lento, termina rápido.
  - ease-out: empieza rápido, termina lento.
  - ease-in-out: lento al inicio y final.
  - steps(n): salto en intervalos definidos (no continuo).

- **transition-delay**: retrasa el inicio de la transición, puede establecer en segundos o milisegundos

Ejemplo:

```
t.elemento {
  transition: background-color 0.3s ease-out 1s;
}
```

Transformación / transform: modifica las propiedades geométricas de un elemento sin afectar el flujo del documento. Se pueden combinar múltiples transformaciones en una sola línea.

- translate(x, y): mueve en el eje X y/o Y.
- rotate(deg): rota el elemento.
- scale(x, y): escala el tamaño.
- skew(x, y): inclina el elemento.
- matrix(...): combinación compleja de transformaciones.

Ejemplo:

```
.elemento {
  transform: scale(1.2) rotate(10deg);
}
```

Animaciones / animation: crear movimientos más complejos y personalizados mediante reglas @keyframes. A diferencia de transition, no requieren interacción del usuario para iniciar.

Estructura:

- **@keyframes:** define cambios para aplicarlos en momentos clave, usando palabras clave from y to o porcentajes (0%, 50%, 100%, etc.).

- **animation-name:** nombre de la animación definida en @keyframes.
- **animation-duration:** tiempo en completarse
- **animation-delay:** tiempo antes de iniciar.
- **animation-iteration-count:** numero de repeticiones o bien usar infinite
- **animation-direction:** permite ser reversible, mas fluida de forma sencilla
  - normal (por defecto),
  - reverse,
  - alternate (ida y vuelta),
  - alternate-reverse.
- **animation-timing-function:** igual que en transition
- **animation-fill-mode:** como se mantienen los estilos después de completar la animación
  - none: sin efecto posterior.
  - forwards: mantiene el estilo final.
  - backwards: aplica el estilo inicial durante el delay.
  - both: aplica ambos.
- **Eventos de animación** (opcional): ejecución en JavaScript como animationstart, animationend, y animationiteration.

Ejemplo:

```
@keyframes mover {
  from { transform: translateX(0); }
  to { transform: translateX(100px); }
}
.elemento {
  animation: mover 2s infinite alternate;
}
```

b

background-color: establece color de fondo.

background-image: establece una imagen (almacenar recursos de forma local - assets).

background-image: url("imagenes/fondo.jpg");

background-position: establece la posición del elemento fondo.

Valores: left, right, center, top, bottom. Combinaciones: top left, bottom right..

Acepta % o unidades (px, em, ..)

background-position: center center;

Orden: primero eje X (horizontal), luego el eje Y (vertical)
background-position: HORIZONTAL VERTICAL;

background-repeat: establece si la imagen debe repetirse para cubrir el fondo.
`no-repeat` (no se repite)
`repeat` (repite en ambas direcciones)
`repeat-x` (solo horizontal)
`repeat-y` (solo vertical)
`space` (espacio entre repeticiones sin recorte)
`round` (ajusta el tamaño para que encaje justo)
background-repeat: no-repeat;

background-size: establece tamaño de la imagen de fondo
`auto` (valor por defecto)
`cover` (la imagen cubre todo el contenedor sin deformarse, puede recortar sobrantes)
`contain` (la imagen se ajusta para caber completa sin recorte, puede dejar espacios vacíos)
Medidas específicas: `100px 200px`, `50% 50%`, etc.
background-size: cover;

Shorthand: se puede escribir todas las propiedades en una sola línea con `background:`

```
background: url("img.jpg") no-repeat center center / cover;
```

border radius: redondea las esquinas.

```
button { border-radius: 8px; }
border-radius: 10px 5px 15px 0;
 /* orden: top-left, top-right, bottom-right, bottom-left */
```

box shadow: agrega sombra alrededor del elemento.
box-shadow: offset-x offset-y blur-radius color;
div { box-shadow: 2px 2px 10px rgba(0,0,0,0.2); }
`offset-x` (2px): desplazamiento horizontal de la sombra respecto al elemento.
Valor positivo → la sombra se mueve hacia la derecha.
Valor negativo → la sombra se mueve hacia la izquierda.

`offset-y` (2px): desplazamiento vertical de la sombra respecto al elemento.
Valor positivo → la sombra se mueve hacia abajo.
Valor negativo → la sombra se mueve hacia arriba.

`blur-radius` (10px):
Controla qué tan difusa o borrosa es la sombra.
Valor 0 → sombra nítida, borde definido.
Valores mayores → sombra más suave y difusa.
ejemplo, `10px` hace que la sombra se difumine y se vea suave.

`color` (rgba(0,0,0,0.2))
Define el color y la transparencia de la sombra.
Aquí es negro (`rgb(0,0,0)`) con 20% de opacidad (`0.2`).
da una sombra negra muy sutil y translúcida.

box sizing: controla cómo se calcula el tamaño de un elemento.
`content-box` (por defecto): ancho solo considera contenido.
`border-box`: ancho incluye padding y border. **Limita o controla** el tamaño total del elemento para que no crezca inesperadamente cuando se agregas padding o bordes.
* { box-sizing: border-box; }

Para aplicar a todo
```
*,
*::before,
*::after {
  box-sizing: border-box;
}
```

c
cameCase: estilo de escritura donde las palabras se juntan y cada nueva palabra inicia en mayúscula (menos la primera).
JS: backgroundColor (no background-color como en CSS)

color: 17 colores básicos definidos desde HTML inicial
147 colores extendidos con nombres reconocidos por navegadores modernos

valores RGB /* named value */ color: red;
valores hexadecimales /* RGB hex value */color: #ff0000;
valores de 0 a 255
/* RGB */ color: rgb(255, 0, 0);
/* RGB porcentual */ color: rgb(100%, 0%, 0%);
/* RGBA (con transparencia) */ color: rgba(255, 0, 0, 0.5);
255, 0, 0 0,255,0 0,0,255 0,0,0 255,255,255
valores HSL y HSLA
color: hsl(0, 100%, 50%);
color: hsla(0, 100%, 50%, 0.5);

Color Safe selección de colores de contraste para asegurar que el sitio sea accesible para todos los usuarios.
https://colorhunt.co/ paletas de colores.

d
display: define **el tipo de caja** o **modelo de visualización** es decir **cómo se comporta visualmente:** de bloque, en línea, flexible, en cuadrícula, etc.

Valor	Función
block	El elemento ocupa todo el ancho disponible. Empieza en línea nueva. Ej: , <p>, <section>
inline	El elemento se quede en la misma línea. Ocupa el mínimo ancho posible. Ej: , <a>
inline-block	Como inline, pero permite definir ancho/alto, márgenes y paddings.
none	Oculta el elemento completamente (ni se ve ni ocupa espacio).
flex	Activa flexbox, sistema para distribuir elementos en línea o columnas con control total.
grid	Activa CSS Grid, para hacer diseños por cuadrículas.
inline-flex	Como flex, pero el contenedor se comporta como inline

inline-grid	Como grid, pero el contenedor se comporta como inline.

Tipo de elemento	Centrado horizontal	Ejemplo
inline / inline-block	text-align: center en el contenedor	 Texto o elemento inline

block con ancho fijo `width` definido, porque si no ocupa 100% del ancho y no hay nada que centrar.	margin: 0 auto	div {
		width: 200px;
		margin: 0 auto;
		}

Funciona para todo, sin importar si es block o inline. (**Flexbox**)	display: flex + justify-content: center	.container {
		display: flex;
		justify-content: center; /* eje horizontal */
		align-items: center; /* eje vertical (si quieres) */
		}

Texto o enlace dentro de un contenedor	text-align: center en el padre	
Una caja completa (bloque)	margin: 0 auto + display: block	

e

elementos: estructura HTML completa, representa una parte del contenido en una página web.
Incluye etiqueta de apertura, el contenido (si lo tiene) y la etiqueta de cierre.

especifidad: Cada parte suma especificidad. Cuanto más "específico", más prioridad tendrá ese estilo.
Si dos reglas aplican al mismo elemento, **gana la que tenga mayor especificidad**.
Si tienen igual especificidad, **gana la que esté escrita después** (por cascada).
La prioridad se puede alterar también con `!important`, pero es mejor evitarlo a menos que sea necesario.

```
body #main .section p {
  color: black;
}
```

!important	
style="…"	1-0-0-0
#id	0-1-0-0
.class	0-0-1-0
Div	0-0-0-1

etiqueta: palabra clave que indica el tipo de contenido HTML

f
fallback: establece una lista de fuentes para un mismo selector, para asegurar el estilo deseado. Si la primera fuente no es compatible con el navegador, usará la siguiente de la lista como alternativa, y así sucesivamente.

```
selector {
  font-family: "Primera fuente", "Segunda fuente", fuente genérica;
  font-family: "Helvetica Neue", Arial, sans-serif;
}
```

Uso de comillas "" cuando la fuente tiene espacios o caracteres especiales.
Fuentes genéricas `serif`, `sans-serif`, `monospace` nunca van entre comillas.

firts-of-class: seudoclase estructural, selecciona elementos **según su tipo y su posición** dentro del padre.

font-family: define la familia tipográfica a usar.

font-size: indica el tamaño de fuente, ajuste de tamaño:
absoluto se establece en píxeles y siempre usa el tamaño especificado. No escala.
relativo (recomendado: es flexible y escalable) Se adapta según el tamaño base del navegador o del elemento padre, mediante:
rem (root em): HTML { font-size: XXpx; } tamaño de la fuente del elemento raíz (`html`). Escala todo el sitio proporcionalmente desde un solo punto (`html`)

em: body/div.. { font-size: XXpx; } tamaño de la fuente del elemento padre, (padre → hijo). Escala dentro de un componente o bloque específico, según su contexto.

tamaño fuente predeterminado **16 píxeles**.

```
html {
  font-size: 62.5%; /* 62.5% de 16px = 10px */
}
```

1rem = 10px,
1.6rem = 16px
2.4rem = 24px

font-style: controla la inclinación del texto.
- normal: texto sin inclinación
- *italic*: cursiva verdadera.
- *oblique*: cursiva artificial.

font-weight: controla el grosor del texto.
Numerico 100 - 900.
Extraligero: 100
Normal: 400
Bold 700 (la mas usada)
Extrapesado: 900

flexbox: (flexible layout) sistema de layout para alinear y distribuir elementos. 1D linea o columna. Requiere `display: flex;` en el contenedor padre.

Propiedad	Eje	Valores	Notas
justify-content	alinea elementos horizontal	**flex-start**: Alinea elementos al lado izquierdo del contenedor.	cuando es una columna (flex-direction: column), **justify-content** cambia a vertical y **align-items** a horizontal
		flex-end: Alinea elementos al lado derecho del contenedor.	
		center: Alinea elementos en el centro del contenedor.	
		space-between: Muestra elementos con la misma distancia entre ellos.	
		space-around: Muestra elementos con la misma separación alrededor de ellos, lo que genera **espacios más grandes en los extremos** comparado con space-between.	
align-items	alinea elementos vertical	**flex-start**: Alinea elementos a la parte superior del contenedor.	
		flex-end: Alinea elementos a la parte inferior del contenedor.	
		center: Alinea elementos en el centro (verticalmente hablando) del contenedor.	
		baseline: Alinea los elementos según la línea base del texto, útil cuando los elementos tienen diferentes tamaños de fuente.	
		stretch: Elementos se estiran para ajustarse al contenedor.	

align-content	establece múltiples líneas están separadas una de otra	**flex-start**: Las líneas se posicionan en la parte superior del contenedor.	**align-content** determina el espacio entre las líneas, mientras que **align-items** determina como los elementos en su conjunto están alineados dentro del contenedor. Cuando hay solo una línea, **align-content** no tiene efecto. **align-self** aplica a elementos individuales
		flex-end: Las líneas se posicionan en la parte inferior del contenedor.	
		center: Las líneas se posicionan en el centro (verticalmente hablando) del contenedor.	
		space-between: Las líneas se muestran con la misma distancia entre ellas.	
		space-around: Las líneas se muestran con la misma separación alrededor de ellas.	
		stretch: Las líneas se estiran para ajustarse al contenedor.	
flex-direction	define la dirección de los elementos en el contenedor	**row**: Elementos son colocados en la misma dirección del texto.	**flex-flow:** combina **flex-direction** y **flex-wrap.** **ej.** flex-flow: column wrap
		row-reverse: Elementos son colocados en la dirección opuesta al texto.	
		column: Elementos se colocan de arriba hacia abajo.	
		column-reverse: Elementos se colocan de abajo hacia arriba.	
flex-wrap	define si los elementos se deben ajustar (hacer wrap) cuando ya no caben	**nowrap**: (por defecto) Cada elemento se ajusta en una sola línea.	
		wrap: los elementos se envuelven alrededor de líneas adicionales.	
		wrap-reverse: Los elementos se envuelven alrededor de líneas adicionales en reversa.	

order		Por defecto, los elementos tienen un valor 0, al usar esta propiedad modifica visual (sin impactar en HTML) del orden de aparición de los hijos directos en un contenedor display: flex o display: grid, uso de valores positivos y negativos, como order: -1;, para que aparezcan antes que los demás. No hace falta que los números sean consecutivos (100, 200, etc.).	

Como actua las propiedades

Propiedad	Alinea en eje	Eje por defecto
justify-content	Principal	Horizontal
align-items	Cruzado	Vertical
align-content	Cruzado, entre lineas	Vertical
align-self	Individual, determina eje principal	Cruzado

Como actua flex-direction

flex-direction	Eje principal	Eje cruzado
row (por defecto)	Horizontal	Vertical
row-reverse	Horizontal	Vertical
column	Vertical	Horizontal
column-reverse	Vertical	Horizontal

En display: flex hay dos ejes:
Eje principal (main axis): se colocan los elementos hijos, por defecto es horizontal pelo lo puede modificar flex-direction.
Eje cruzado (cross axis): eje perpendicular al principal

Hijos .item

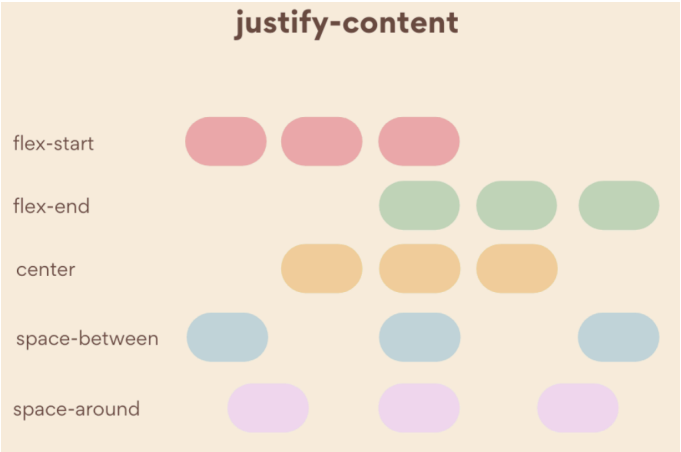
`flex-grow`	Cuanto crece	Se pueden definir en una única propiedad flex: 1 1 auto flex-grow:1 flex-shrink:1 flex-basis: auto
`flex-shrink`	Cuanto encoge	
`flex-basis`	Tamaño base, tiene prioridad sobre widht	

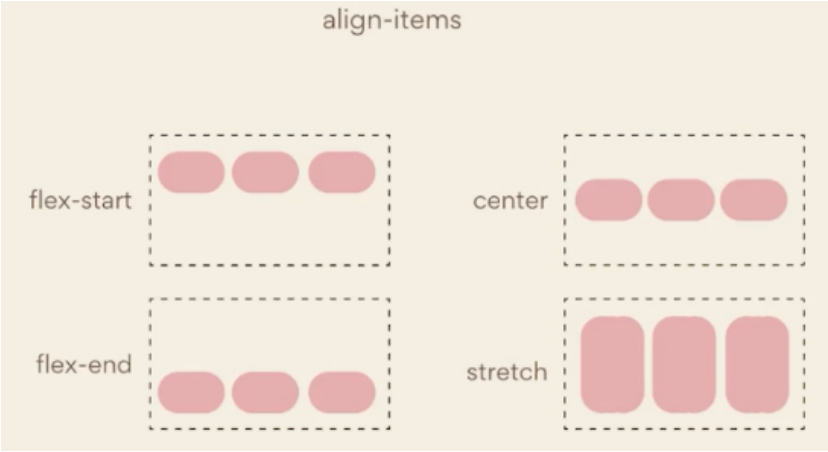
gap: define el espacio entre elementos hijos dentro de un contenedor con `display: flex` (flex-wrap:wrap) o `display: grid`.
Más limpio que usar `margin-right, margin-bottom`, etc.
Evita errores de espacio extra al final (como cuando el último hijo tiene un `margin`).
Posible usar cualquier unidad: `px`, `rem`, `%`, etc.
Para solo separar horizontal o verticalmente, usar `row-gap` o `column-gap`:

*flex-container: primer div
*flex-items: elementos hijos

Contenedor:

```
.container {
  display: flex;
  justify-content: center; /* eje horizontal */
  align-items: center;     /* eje vertical */
  flex-direction: row;     /* fila horizontal (por defecto) */
  gap: 10px;               /* espacio entre hijos */
}
```





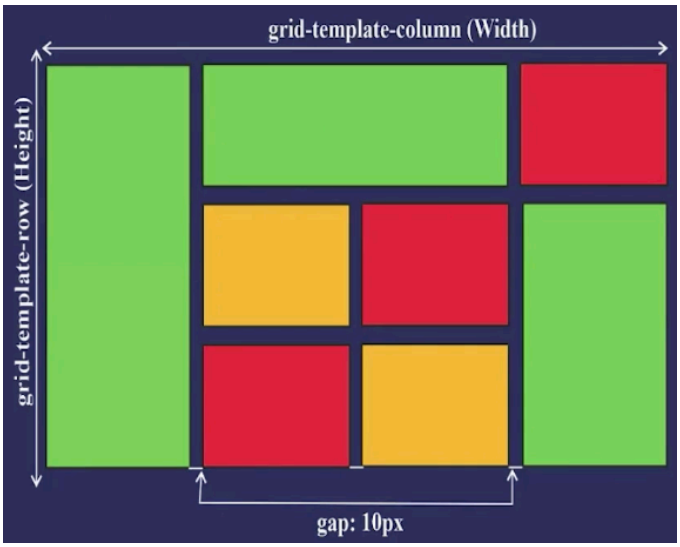
g

gradiente: fondo que cambia de un color a otro.

Gradiente	Función	Ejemplo
linear-gradient	Lineal: to right, to left, to bottom right, etc.	`background: linear-gradient(to right, red, yellow);` (Angulos) `linear-gradient(45deg, red, blue)`
radial-gradient	Circular	`background: radial-gradient(circle, red, yellow, green);`
conic-gradient	Reloj	`background: conic-gradient(red, yellow, green, red);`
Otros		`background: linear-gradient( 135deg, #ffb6c1 0%, #ff69b4 50%, #c71585 100% );`
Imagen		`background: linear-gradient(rgba(0,0,0,0.5), rgba(0,0,0,0.5)), url('fondo.jpg');`

grid: sistema de diseño bidimensional 2D (filas y columnas).

Declarar grid




```
.container {
  display: grid;
 grid-template-columns: 1fr 1fr 1fr;
/* 3 columnas iguales */
  gap: 20px;
}
```

1fr: fracción de espacio disponible, flexible y responsivo

```
.grid {
  display: grid;
 grid-template-columns: repeat (3 1fr);
/* 3 columnas iguales col1 33% col2 33% col3 33%*/
}
grid-template-columns: 2fr 1 fr 1fr;
/* 3 columnas iguales col1 50% col2 25% col3 25%*/
```

grid-template-columns: auto auto auto;
/* cada auto es una columna*/
grid-auto-rows: 200px 300px 50px;
/* 3 medidas para el largo de columnas*/
O bien
grid-auto-rows: 200px
/* misma medida largo para todas las columnas */

grid-template-columns: repeat(auto-fit, minmax(200px, 1 fr);
/* tantas columnas de mínimo 200px, si sobra espacio crece hasta ocupar 1fr*/

grid-area (nombrar cada area ej a-head z-footer)
grid-template-areas (especificación de orden del layout a cada nombre)

Grid	Función	Ejemplo

`grid-template-columns`	Define cuántas columnas y su tamaño	`grid-template-columns: 200px 1fr 2fr;`
		`px %`: tamaños fijos o relativos
		`fr:` fracción del espacio disponible
		`repeat(n, val):` atajo
		`grid-template-columns: repeat(3, 1fr);`
		3 columnas iguales
		`grid-template-columns: auto;`
		Cada auto es una columna (12)

`grid-template-rows`	Igual que columnas, pero en vertical	
`gap`	Espacio entre columnas y filas	`gap: 10px;`

h
HSL: tono, saturación, luminosidad
H = Hue (0–360°)
S = Saturación (%)
L = Luminosidad (%)

l
librería/marco de trabajo (framework): conjunto de herramientas que extienden las **funcionalidades** de JavaScript para facilitar el desarrollo de

interfaces y aplicaciones web. React.js (librería), Angular y Vue (frameworks).
Librería: da piezas para armar la aplicación, controla el flujo (React).
Framework: define una estructura más rígida, controla más el código (bootstrap.
Una librería es como pedir ayuda. Un framework es como seguir órdenes.

m
media queries: bloque CSS que se ejecuta si se cumple una condición, / (viewport) permiten aplicar estilos **condicionales** dependiendo del tamaño de pantalla, resolución, orientación, tipo de dispositivo, etc.

```
@media (condición) {
  /* reglas CSS que aplican sólo si se cumple la condición */
}
@media(max-width: 500px){
    div.grid-container{
        grid-template-columns: auto;
    }
}
```

Condiciones mas comunes

Condición	Función	Ejemplo
max-width	máximo ancho Si la pantalla es igual o más chica que X, aplica esto	`(max-width: 768px)` para tablets y móviles
min-width	mínimo ancho Si la pantalla es igual o más grande que X, aplica esto	`min-width: 1024px)` para desktop grandes
orientation	orientación de la pantalla	`portrait o landscape`
prefers-color-scheme	modo claro u oscuro	`light, dark`

Anatomía avanzada de un breakpoint

@media [tipo de media] [operador and] [condición]

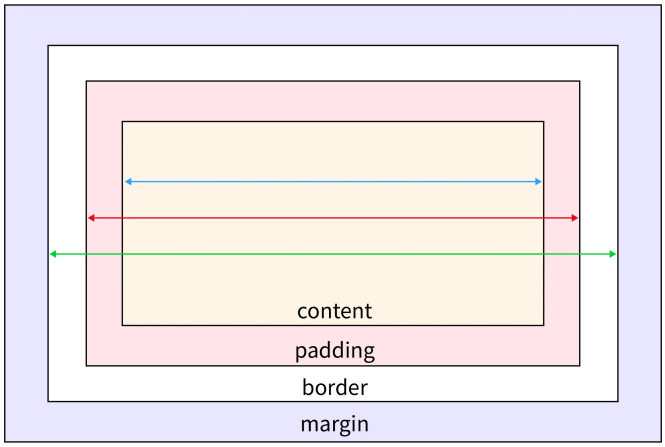
```
@media screen and (min-width: 768px) and (orientation: landscape) {
  .card {
    grid-template-columns: repeat(3, 1fr);
```

```
  }
}
```

	Función	Condición
`@media`	Inicio de una medida query	
`screen`	Tipo de medio (pantalla)	all: aplica a todos los medios screen: se enfoca en la pantalla de un dispositivo speech: sintetizadores de voz print: diseño dedicado a la impresión
`and`	Combinador lógico (poner varias condiciones)	@media screen and (min-device-width: 320px) and (max-device-width: 480px)
`(min-widt: )`	Aplicar solo si el ancho de pantalla es mayor o igual	min-width / max-width
`(orientation: )`	Solo si la pantalla es horizontal o vertical	portrait (vertical) o landscape (horizontal)
`.card {…}`	Estilos que se activan si cumplen ambas condiciones	

modelo de cajas:

Margin (externo)
  Border
    Padding
      Content (height y width)



width
- content-box: content
- padding-box: content + padding
- border-box: content + padding + border

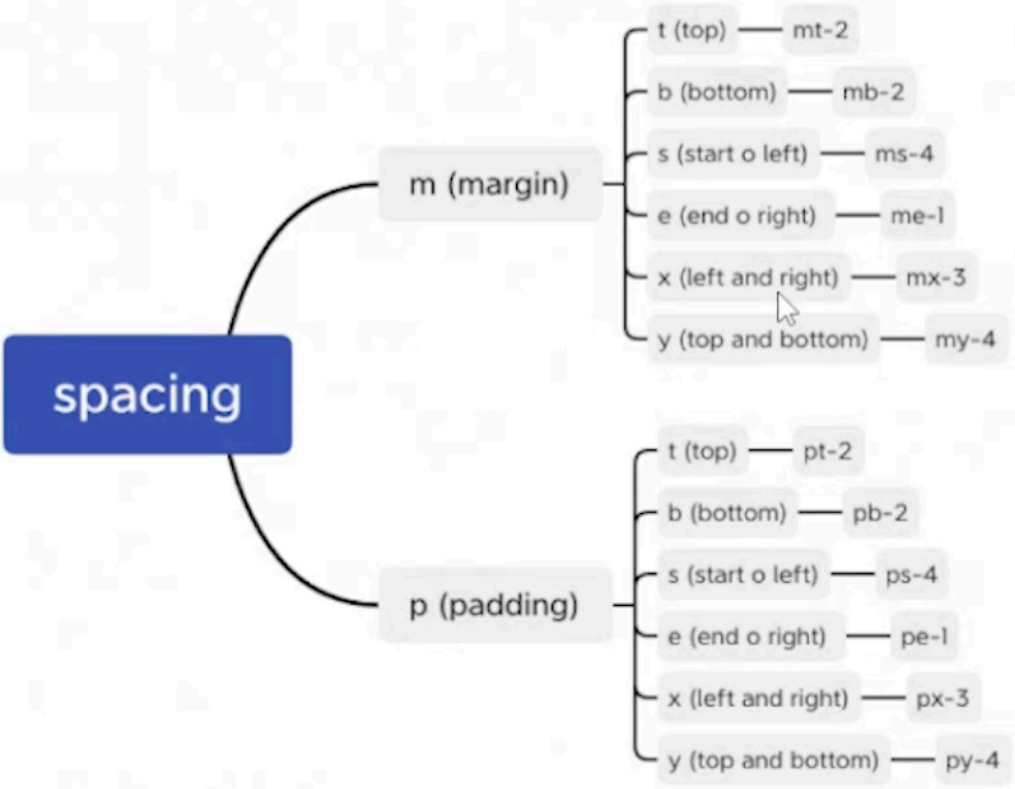
Content: texto, imágenes, o cualquier contenido.
Widht: tamaño de ancho
Height: tamaño de altura

Padding (relleno): espacio interno entre contenido y borde, separa el contenido del borde.

```
padding-top, padding-right, padding-bottom, padding-left
padding: 10px; /* para todos los lados */
```

Border: Rodea el relleno y el contenido. Se puede definir ancho, estilo y color.

```
border: 2px solid black;
```



Margin: espacio exterior a la caja, separando esta caja de otras cajas o elementos.

```
margin: 20px; /* para todos lados */
margin-left: 5px;
margin: 60px, 10px, 30px, 50px; ⏱️
Arriba (60), derecha(10), abajo(30), izquierda(50)
```

Aquí interviene box-sizing.

`box-sizing: content-box;` → comportamiento por defecto, el tamaño solo es del contenido, padding y borde se suman.

`box-sizing: border-box;` → el padding y borde **se incluyen dentro** del ancho y alto definidos. Esto hace que el tamaño total de la caja sea más fácil de controlar.

```
width: 200px;              /* ancho del contenido */
height: 100px;             /* alto del contenido */
padding: 20px;             /* espacio interno alrededor del contenido */
border: 5px solid #333;    /* borde alrededor del padding */
margin: 30px;              /* espacio fuera del borde */
box-sizing: content-box;   /* valor por defecto */
```

p
posicionamiento: se combina con
top: X;
left: X;
right: X;
bottom: X;
z-index: X;

Posicionamiento	Función	Ejemplo
static (default)	El elemento sigue el flujo normal del documento. No se puede mover con top, left, etc.	`position: static;`
relative (Mover sin romper layout)	Se mueve relativo a su posición original, puede moverse ligeramente con `top`, `left`, etc.	`div { position: relative; top: 10px; left: 20px; }`
absolute (Tooltips, íconos flotantes dentro de contenedor)	Se posiciona respecto al **contenedor con `position: relative`** más cercano (o al <body>). Se sale del flujo normal (como si flotara).	`.padre { position: relative; } .hijo { position: absolute; top: 0; right: 0; }`
fixed (Botones, menús flotantes)	Siempre está en el mismo lugar (fijo en el viewport) **aunque se haga scroll**. Ejemplo: barra de navegación.	`nav { position: fixed; top: 0; width: 100%; }`
sticky (Encabezados fijos al hacer scroll)	Funciona como relative **hasta que haces scroll**, luego se pega en una posición.	`h2 { position: sticky; top: 0; }`

pseudoclases: cambia estado visual, palabras clave precedidas por dos puntos **:** añade estilos a un **estado específico de un elemento**, sin necesidad de agregarle una clase manualmente. Cómo interactúa el usuario con ellos. (:hover, :active, :visited, :link, :fist-child, :last-child, :nth-child())

Seudoclases a

se aplican a enlaces (<a href="...">) html	Función	Ejemplo
Link	Enlace normal, no visitado	a:link { color: blue;}
Visited	Enlace que el usuario ya visitó	a:visited { color: purple;}
Hover	Cuando el mouse pasa por encima	a:hover { color: red;}
Active	Mientras se hace clic (activo)	a:active { color: orange; }
	respetar el orden: **LVHA** → link, visited, hover, active	

Seudoclases child

Childs	Función	Ejemplos
First-child	Seleccionar el primer elemento que aparece dentro de una etiqueta.	div p:first-child { ... }
Last-child	Seleccionar el ultimo elemento que aparece dentro de una etiqueta.	div p:last-child { ... }
Nth-child(n)	Seleccionar n elemento un orden específico dentro de una etiqueta.	ul li:nth-child(2) { ... }
Nth-child(odd)	Seleccionar **impares** elemento que aparece dentro de una etiqueta.	ul li:nth-child() { ... }
Nth-child(even)	Seleccionar **pares** elemento que aparece dentro de una etiqueta.	ul li:nth-child() { ... }
(3n+1)...	...	...

Diferencia entre seudoclase childs y of-class

HTML

```
<div>
  <h2>Título</h2>
  <p>Primer párrafo</p>
  <p>Segundo párrafo</p>
</div>
```

CSS

Selector	¿Funciona con <p>?	¿Por qué?
p:first-child	❌ No	Porque el primer hijo real es un <h2>, no un <p>
p:first-of-type	✅ Sí	Porque es el **primer <p>** dentro del contenedor
h2:first-child	✅ Sí	Porque es el **primer hijo total** del
h2:first-of-type	✅ Sí	Porque es el **primer <h2>** dentro del

Pseudoclases avanzadas

Pseudoclase	Función	Ejemplo

:focus	Aplica solo cuando el input está activo, es decir, cuando el usuario hace clic o navega con el teclado.	`html <input type="text" placeholder="Escribe algo”>`
		
		`Css input:focus { outline: 2px solid blue; background-color: lightyellow; }`

:empty	Cuando un elemento no tiene contenido visible Se oculta automáticamente si no tiene nada adentro. Útil en contenido dinámico, tarjetas automáticas o placeholders.	`Html <div class="cuadro"></div> <!-- está vacío —>`
		
		`Css .cuadro:empty { display: none; }`

:not()	Selecciona elementos **que NO cumplan** cierta condición, sin necesidad de agregar clases.	`li:not(.especial) { color: gray; }`
		`O`
		`p:not(:first-child) { margin-top: 20px; }`

pseudoelementos: inserta contenido extra, permite aplicar estilos a una parte específica de un elemento, como crear un sub-elemento que no existe en el HTML.
Se escribe con **doble dos puntos `::`** para diferenciarlo de las pseudoclases (`:`).

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

Pseudoelemento	Función	Ejemplo
`::before`	Inserta contenido antes del elemento	`h2::before { content: "→ "; color: green; }`
`::after`	Inserta contenido después del elemento	`h2::after { content: " ✔"; color: red; }`
`::first-letter`	Aplica estilo a la primera letra	`p::first-letter { font-size: 200%; color: crimson; }`
`::first-line`	Aplica estilo a la primera línea del texto	`p::first-line { font-weight: bold; }`
`::selection`	Estiliza el texto seleccionado (cuando lo marcas con el mouse)	`::selection { background: hotpink; color: white; }`

pseudo-selectores: selectores que apuntan al estado de un elemento

Tipo de selector	Descripción	Ejemplos
pseudo-clase	Selecciona y aplica estilos a elementos en función de su estado/ relación con otros elementos	`:hover` `:active` `:focus`
pseudo-elemento	Selecciona y aplica estilos a partes especificas de un elemento	`::before` `::after` `::first-line`

r

regla horizontal: `<hr>`, línea divisoria horizontal que separa visualmente un párrafo/sección de otro, en HTML. Mejora la estructura visual y ayuda en accesibilidad.

Estilo en CSS: margin vertical generoso (20px o 30px) para darle aire.

Estilos	Código
Eliminar el borde y usar background-color, controla mejor el grosor y color de la línea	`hr { border: none; height: 2px; background: black; }`

Gradiente suave	`hr { border: none; height: 2px; background: linear-gradient(to right, #f0f, #0ff);`

Línea punteada o rayada	`hr { border: none; border-top: 2px dotted #999; }`
	
	`hr { border: none; border-top: 2px dashed #aaa; }`

Doble línea sutil	`hr { border: none; border-top: 3px double #444; }`
Sombra con profundidad	`hr { border: none; height: 1px; background: #ccc; box-shadow: 0 2px 5px rgba(0,0,0,0.2); }`
Línea centrada con degradado a los lados	`hr { border: none; height: 2px; background: linear-gradient(to right, transparent, #333, transparent); margin: 30px 0; }`

responsive: diseño adaptable a distintos tamaños de pantalla, implica que el sitio sea usable y legible sin hacer zoom o scroll horizontal.

Como ser responsive

Técnica	Como ayuda
Unidades relativas	Usar %, em, rem, vw, vh en lugar de px rígidos.
Media Queries	Reglas CSS que aplican solo si se cumple cierta condición de pantalla. (@media)
Flexbox y Grid	Distribuyen espacio de forma flexible entre los elementos.
Imágenes fluidas	Usar width: 100% para que las imágenes no se salgan de su contenedor.

selector: indica **a qué elemento HTML** se le va a aplicar un estilo. (body, div, .clase, #id, …)
Selector universal *
3 tipos:
- Elementos: selección directa por etiqueta
- Clases: elementos pertenecientes a la clase .class
- ID: unico #ID

Selector especial :root
Representa el elemento `<html>`, con más peso para definir **variables globales** de CSS. Se usa principalmente para declarar variables personalizadas (`--nombre`) que luego se puede reutilizar en todo el archivo CSS.

```
:root {
  --color-principal: #ff66cc;
  --fuente-grande: 20px;
}

h1 {
  color: var(--color-principal);
  font-size: var(--fuente-grande);
}
```

selector descendiente: selecciona elementos dentro de otros es decir se aplica desde el "padre" hacia cualquier "hijo", "nieto", etc.
div p { color: blue; }
Afecta todos los `<p>` dentro de un `<div>`.

skewer-casing: (kebab-case) formato de escritura para nombrar palabras usando letras minúsculas y guiones medios (–) entre palabras

safe-fonts: Fuentes ampliamente compatibles instaladas en la mayoría de sistemas.
Se definen en `font-family` listando varias opciones: fuentes preferidas, luego fuentes comunes, y al final una genérica como `sans-serif`.

CSS

Sans-serif: Arial, Verdana, Helvetica, Tahoma, TrebuchetMS.
Serif: TimesNewRoman, Georgia, Garamond.
Monospace (ancho fijo): CourierNew, Lucida Console.
Cursive: BrushScriptMT.
Sistema (UI): Segoe UI, Roboto, San Francisco.

SEO: proceso de optimizar un sitio web para mejorar su **posición en los resultados de búsqueda** de Google u otros motores, **de forma orgánica** (sin pagar publicidad). Engloba HTML (uso correcto de etiquetas), enlaces, velocidad (imágenes comprimidas), accesibilidad y responsive.

sistema (UI): Sistema (User Interface): da una apariencia nativa según el sistema del usuario.
Segoe UI (windows), Roboto (android), San Francisco (iOS).

sans-serif: garantiza que el texto sea una fuente legible. Sin remates, como Arial
serif: con remates, como Times New Roman

: elemento **en línea** usado para aplicar estilos a partes pequeñas especificas (una palabra) del texto sin romper el flujo.

t
text-align: left, right, center, justify.

text-decoration: aplica estilo alrededor, pero no modifica la propia fuente real underline, overline o line-through (sobre el texto)

text-transform: uppercase, lowercase, capitalize, none.

text-shadow: none, <lenght> <lenght> <lenght>
text-shadow: pink 5px 1px 4px
- Color
- Desplazamiento horizontal de la sombra
- Desplazamiento vertical de la sombra
- Blur

u

UI User Interface: Interfaz de Usuario, lo visual

UX User Experience: Experiencia de Usuario, cómo se siente y funciona la interacción

v

valor: se le da a una propiedad (atributo) para definir **cómo se verá** el estilo. Va después de la propiedad, separado por dos puntos `:` y termina con punto y coma `;`.

Estructura básica

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

Selector → el elemento que se quiere modificar (`p`, `h1`, `.clase`, `#id`)
Propiedad → aspecto visual que se quiere cambiar (`color`, `font-size`, `text-align`)
Valor → lo que determina **cómo** se verá (`red`, `16px`, `center`)

Ej.

```
h1 {
  color: blue;
  font-size: 24px;
  text-align: center;
}
```

variables: es un nombre personalizado que guarda un valor CSS (color, tamaño, fuente, etc.) y que puedes **reutilizar** en todo el archivo.
Las variables **empiezan con —**
Van dentro de una regla CSS (como `:root` o cualquier otro selector)
Su valor se accede con `var(--nombre-variable)`

Términos básicos

Termino	Definición	Ejemplo
Elemento	Unidad completa de HTML: etiqueta de apertura, contenido, etiqueta de cierre	`<p>Hola mundo</p>`
Etiqueta	Palabra clave que indica el tipo de contenido HTML	`<h1>, <div>, <img>, <a>`
Atributo	Información adicional dentro de una etiqueta	`src="gatito.jpg"` → `src` es el atributo, `"gatito.jpg"` es su valor

Valor	Lo que se le asigna a un atributo o propiedad	CSS
		`color: red;`
		→ `red` es el valor de la propiedad `color`
		HTML
		`<p class="importante">`
		→ `"importante"` es valor del atributo `class`

Selector	En CSS, indica a qué elemento HTML se aplicarán los estilos	`p, .clase, #id, div, h1`
Propiedad	Característica que se quiere modificar con CSS	`color, font-size, margin, background-color`

Jerarquía / Prioridad en CSS

Prioridad	
Entre inline, style y sheet	
1	Estilo inline
2	Estilo interno
3	Hoja externa
Especifidad	
1	!Important
2	style="…" 1-0-0-0
3	#id 0-1-0-0
4	.class 0-0-1-0
5	div 0-0-0-1

Unidades comunes en CSS

Unidad	Definición	Significado
px	Unidad absoluta. Medida fija, útil para precisión (no escalable).	Pixeles
%	Unidad relativa al contenedor padre. Ideal para diseño fluido y responsive.	Porcentaje
em	Relativa al tamaño de fuente del elemento padre. Afecta padding, margin, etc.	Element
rem	Relativa al tamaño de fuente del elemento raíz (<html>). Más predecible que em.	Root element
vh	1vh = 1% del alto del viewport (pantalla visible). Útil para secciones a pantalla completa.	Viewport Height
vw	1vw = 1% del ancho del viewport. Útil para diseños adaptables al tamaño de pantalla.	Viewport Width

Convenciones de escritura

Convención	Definición	Ejemplo
kebab-case	Palabras en minúsculas separadas por guiones. Usada en nombres de clases y archivos.	`.mi-clase-ejemplo {}`
camelCase	Primera palabra minúscula, siguientes con mayúscula inicial. Común en JavaScript.	`miVariableImportante`
PascalCase	Cada palabra con mayúscula. Se usa para nombres de componentes en frameworks.	`MiComponentePrincipal`
snake_case	Palabras en minúsculas separadas por guiones bajos. Menos común en CSS	`nombre_variable_secundaria`
CSS shorthand	Agrupar propiedades en una sola línea para simplificar código.	`margin: 10px 20px;`
Comentarios CSS	Ayudan a organizar el código, no se interpretan.	`/* Sección de botones */`