Tabla resumen de propiedades CSS y sus valores

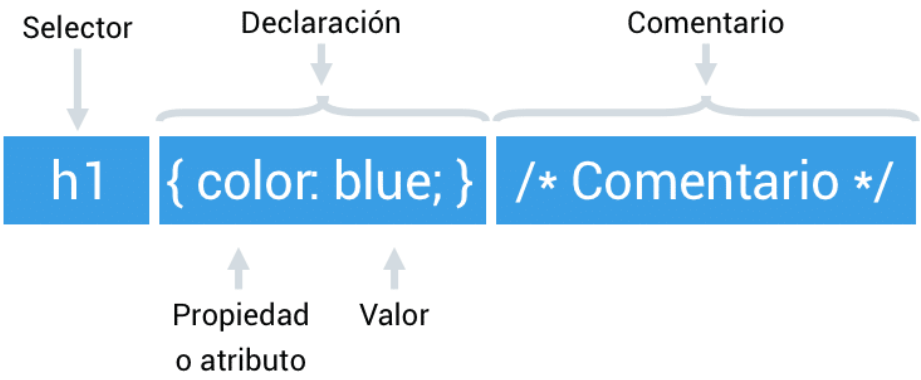
Tabla resumen CSS

A continuación, hemos preparado una práctica **chuleta CSS** en forma de **tabla resumen de propiedades CSS** junto con sus valores correspondientes. Estamos seguros de que esta chuleta en forma de tablas será de gran ayuda en el desarrollo de tus proyectos de diseño web, facilitándote el acceso rápido a la información que necesitas. ¡Esperamos que te resulte útil y que simplifique tu trabajo!

Contenidos del artículo

1. Tabla resumen de propiedades CSS Nivel Inicial

1.1. Sintaxis selector



- **Selector** (*selector*): indica sobre qué elemento se aplican los estilos CSS.
- **Propiedad o atributo** (*property or attribute*): indica qué característica se va a cambiar.
- **Valor** (*value*): indica el valor de la propiedad que se desea modificar.
- **Comentario** (*comment*): los comentarios se escriben entre el carácter de apertura /* y el carácter de cierre */.

1.2. Selectores básicos

Selector	Descripción
*****	Selecciona todos los elementos del DOM
etiqueta	Selecciona todas las etiquetas indicadas
.class	Selección de los elementos con la clase .class
#id	Selección del elemento con id #id
sel1 sel2	Selección de los selectores sel2 que se encuentren dentro de los selectores sel1
.class1.class2	Selección de los elementos con las dos clases: class1 y class2
sel1.class1	Selección de todos los selectores sel1 con clase class1
sel1, sel2	Selección de todos los selectores separados por comas
sel1 > sel2	Selección de los selectores sel2 cuando son hijos de sel1
sel1 + sel2	Selección del selector sel2 cuando es hermano de sel1 (su elemento padre es el mismo)

Tabla 1. Tipos de selectores. Selectores básicos

1.3. Unidades de medida

Longitudes relativas	
px	Píxeles (relativo al dispositivo)
em	Relativo al tamaño de la fuente del elemento (2em significa 2 veces el tamaño de la fuente actual)
%	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
Longitudes absolutas	
in	Pulgadas (1pulgada = 2.54 cm)
cm	Centímetros
mm	Milímetros
pt	Puntos (1pt = 1/72 pulgadas)
pc	Picas (1pica = 12 puntos)

Tabla 2. Unidades de medida

1.4. Colores básicos

Tabla 3: Colores básicos

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

A continuación te muestro el listado de colores HTML y sus valores expresados por nombre, código hexadecimal y en RGB.

Color	Nombre	HEX	RGB
	aliceblue	#f0f8ff	240,248,255
	antiquewhite	#faebd7	250,235,215
	aqua	#00ffff	0,255,255
	aquamarine	#7fffd4	127,255,212
	azure	#f0ffff	240,255,255
	beige	#f5f5dc	245,245,220
	bisque	#ffe4c4	255,228,196
	black	#000000	0,0,0
	blanchedalmond	#ffebcd	255,235,205
	blue	#0000ff	0.0.255

1.5. Color y fondo

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

Tabla 4. Colores y fondo

1.6. Texto

Propiedad	Descripción	Valores
text-indent	Desplazamiento de la primera línea del texto	longitud | porcentaje
text-align	Alineamiento del texto	left | right | center | justify
text-decoration	Efectos de subrayado y tachado	none | underline | overline | line-through
letter-spacing	Espacio entre caracteres	normal | longitud

Propiedad	Descripción	Valores
word-spacing	Espacio entre palabras	normal | longitud
text-transform	Transformación a mayúsculas / minúsculas	capitalize | uppercase | lowercase | none
line-height	Tamaño del espacio entre líneas	longitud | porcentaje
vertical-align	Alineación vertical	top | middle | bottom baseline | sub | super | valor

Tabla 5. Propiedades de los textos

1.7. Fuentes

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

Tabla 6. Propiedades de las fuentes

1.8. Listas

Propiedad	Descripción	Valores
list-style-type	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
list-style-image	Imagen aplicable a las viñetas de las listas	url(«…») | none
list-style-position	Posición de las viñetas dentro de la lista	inside | outside
list-style	Permite establecer varios estilos de la lista en una sola propiedad	list-style-type | list-style-position | list-style-image

Tabla 7. Propiedades de las listas

1.9. Tablas

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

Tabla 8. Propiedades de las tablas

1.10. Pseudo-clases para selección de hijos o hermanos

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

Tabla 9. Pseudo-clases para la selección de hijos o hermanos

1.11. Pseudo-clases para los estados de un elemento

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

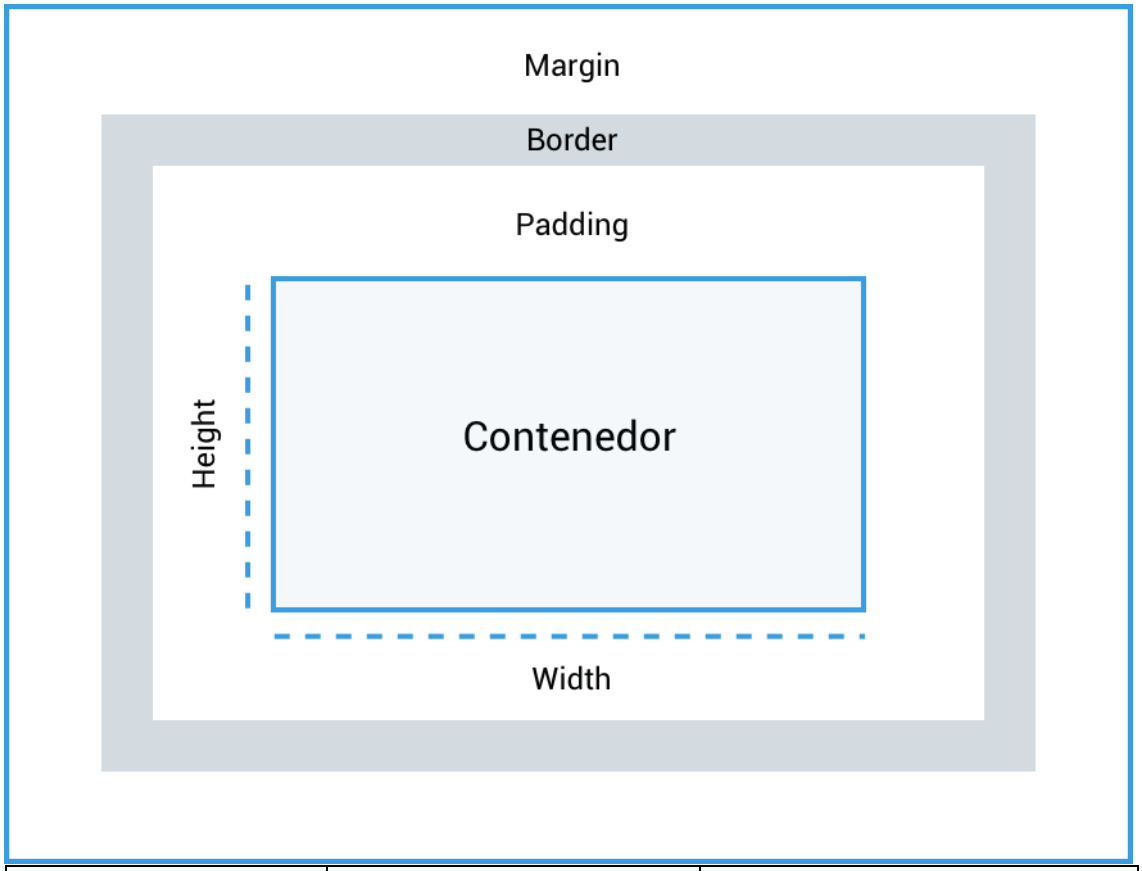
Tabla 10. Pseudo-clases para los estados de un elemento

1.12. Pseudo-elementos

Pseudo-clase	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

Tabla 11. Pseudo-elementos

1.13. Modelo de cajas: márgenes, relleno y bordes



Propiedad	Descripción	Valores
padding-top padding-right padding-bottom padding-left	Tamaño del relleno superior, derecho, inferior e izquierdo	longitud | porcentaje
padding	Tamaño del relleno	longitud | porcentaje {1,4}
margin-top margin-right margin-bottom margin-left	Tamaño del margen superior, derecho, inferior e izquierdo	longitud | porcentaje | auto
margin	Ancho de los margenes	longitud | porcentaje | auto {1,4}
border-top-width border-right-width border-bottom-width border-left-width	Anchura del borde superior, derecho, inferior o izquierdo	thin | medium | thick | longitud

Propiedad	Descripción	Valores
border-width	Anchura del borde	thin | medium | thick | longitud {1,4}
border-top-color border-right-color border-bottom-color border-left-color	Color del borde superior, derecho, inferior e izquierdo	color | transparent
border-color	Color del borde	color | transparent {1,4}
border-top-style border-right-style border-bottom-style border-left-style	Estilo del borde superior, derecho, inferior e izquierdo	none | hidden | dotted | dashed | solid | double | groove | ridge | inset | outset
border-style	Estilo del borde	none | hidden | dotted | dashed | solid | double | groove | ridge | inset | outset {1,4}
border-top border-right border-bottom border-left	Ancho, estilo y color para el borde superior, derecho, inferior e izquierdo	border-top-width | border-top-style | border-top-color
border	Ancho, estilo y color para los bordes	border-width | border-style | border-color
border-radius	Curvatura del borde	longitud | porcentaje {1,4}

Tabla 12. Modelo de cajas: márgenes, relleno y bordes

1.14. Propiedades width, height, max-width, min-width, max-height y min-height

Nombre propiedad	Descripción	Valores
width	Establece el ancho del área de contenido de un elemento	Unidad de longitud (px, em, %, etc.), auto, initial, inherit

Nombre propiedad	Descripción	Valores
height	Establece el alto del área de contenido de un elemento	Unidad de longitud (px, em, %, etc.), auto, initial, inherit
max-width	Establece el ancho máximo que puede tener un elemento	Unidad de longitud (px, em, %, etc.), none, initial, inherit
min-width	Establece el ancho mínimo que debe tener un elemento	Unidad de longitud (px, em, %, etc.), 0, initial, inherit
max-heigt	Establecer el alto máximo que debe tener un elemento	Unidad de longitud (px, em, %, etc.), 0, initial, inherit
min-height	Establecer el alto mínimo que debe tener un elemento	Unidad de longitud (px, em, %, etc.), 0, initial, inherit

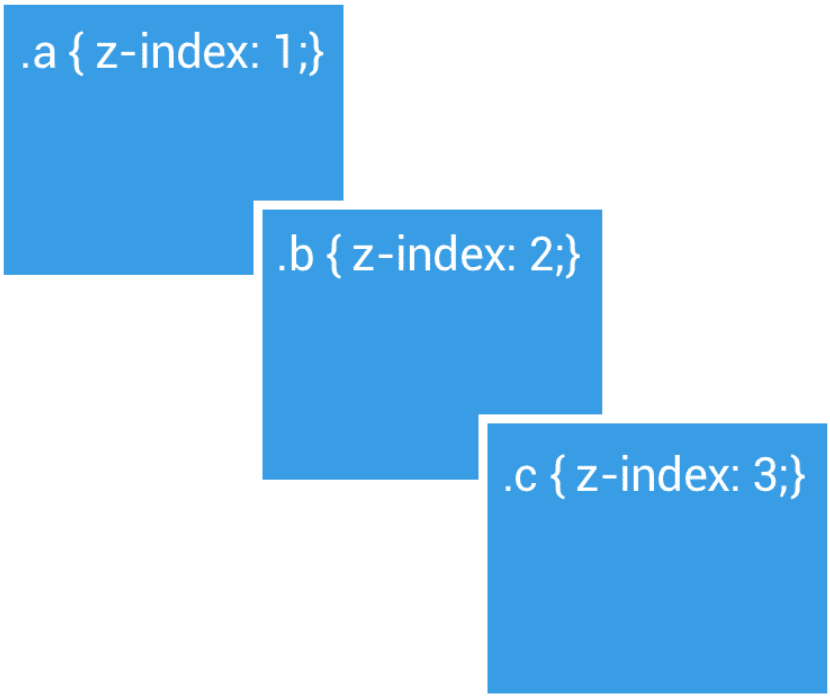
Tabla 13. Tamaño de los elementos en CSS

1.15. Comportamiento de los contenedores

Propiedad	Descripción	Valores
display	Comportamiento del contenedor	inline | block | inline-block | none
position	Esquema de posicionamiento	static | relative | absolute | fixed
top right bottom left	Desplazamiento de la caja respecto al borde superior, derecho, inferior o izquierdo	longitud | porcentaje | auto
float	Posicionamiento flotante	left | right | none
clear	Control de cajas adyacentes a las float	none | left | right | both

Propiedad	Descripción	Valores
z-index	Nivel de la capa	auto | número entero
box-sizing	Control de bordes y relleno en el comportamiento del contenedor	content-box | border-box
visibility	Muestra u oculta un elemento ocupando el espacio	visible | hidden
overflow	Visibilidad de los elementos de tipo bloque	visible|hidden|scroll|auto

Tabla 14. Comportamiento de los contenedores



1.16. Tamaño de los elementos

Nombre propiedad	Descripción	Valores
width	Establece el ancho del área de contenido de un elemento	Unidad de longitud (px, em, %, etc.), auto, initial, inherit
height	Establece el alto del área de contenido de un elemento	Unidad de longitud (px, em, %, etc.), auto, initial, inherit
max-width	Establece el ancho máximo que puede tener un elemento	Unidad de longitud (px, em, %, etc.), none, initial, inherit
min-width	Establece el ancho mínimo que debe tener un elemento	Unidad de longitud (px, em, %, etc.), 0, initial, inherit

Tabla 15. Tamaño de los elementos

1.17. Propiedad overflow, excedente de contenido

Propiedad CSS overflow	Valores
overflow	visible, hidden, scroll, auto
overflow: visible	Por defecto
overflow: hidden	Oculta los contenidos desbordados
overflow: scroll	Muestra barras de desplazamiento
overflow: auto	El navegador decide
overflow-x	visible, hidden, scroll, auto

Propiedad CSS overflow	Valores
overflow-y	visible, hidden, scroll, auto

Tabla 16. Propiedad overflow, excedente de contenido

2. Tabla resumen de propiedades CSS Nivel Intermedio

2.1. Medios en CSS

Medio	Descripción
all	Todos los dispositivos
print	Para documentos paginados y mostrados en vista de impresión
screen	Pantallas de ordenador
speech	Sintetizadores para navegadores de voz utilizados por personas discapacitadas

Tabla 18. Tipos de medios en CSS

Formas de definir los medios:

<link rel="stylesheet" type="text/css" media="screen" href="style.css" />
<link rel="stylesheet" type="text/css" media="print" href="especialStyle.css" />
<link rel="stylesheet" type="text/css" media="screen" href="style.css" /> <link rel="stylesheet" type="text/css" media="print" href="especialStyle.css" />

```
<link rel="stylesheet" type="text/css" media="screen" href="style.css" />

<link rel="stylesheet" type="text/css" media="print" href="especialStyle.css" />
```

@media print { body { font-size: 11pt; } }
@media screen { body { font-size: 12px; } }
@media { body { color: blue; } } /* Se aplica a todo */
@media print { body { font-size: 11pt; } } @media screen { body { font-size: 12px; } } @media { body { color: blue; } } /* Se aplica a todo */

```
@media print  { body { font-size: 11pt; } }

@media screen { body { font-size: 12px; } }

@media { body { color: blue; } } /* Se aplica a todo */
```

```
@import url("estilosPantalla.css") screen;
@import url("estilosImpresora.css") print;
@import url("estilos.css"); /* Se aplica a todo por defecto*/
```

```
@import url("estilosPantalla.css") screen; @import url("estilosImpresora.css") print; @import url("estilos.css"); /* Se aplica a todo por defecto*/
```

```
@import url("estilosPantalla.css") screen;

@import url("estilosImpresora.css") print;

@import url("estilos.css"); /* Se aplica a todo por defecto*/
```

2.2. Media queries

```
@media (min-width: 768px) {
/* Estilos para pantallas con un ancho mínimo de 768px */
}
```

```
@media (min-width: 768px) { /* Estilos para pantallas con un ancho mínimo de 768px */ }
```

```
@media (min-width: 768px) {

 /* Estilos para pantallas con un ancho mínimo de 768px */

}
```

```
@media (max-width: 767px) {
/* Estilos para pantallas con un ancho máximo de 768px */
}
```

```
@media (max-width: 767px) { /* Estilos para pantallas con un ancho máximo de 768px */ }
```

```
@media (max-width: 767px) {

 /* Estilos para pantallas con un ancho máximo de 768px */

}
```

```
@media (min-width: 768px) and (max-width: 1024px) {
/* Estilos para pantallas con un ancho entre 768px y 1024px */
}
```

@media (min-width: 768px) and (max-width: 1024px) { /* Estilos para pantallas con un ancho entre 768px y 1024px */ }

```
@media (min-width: 768px) and (max-width: 1024px) {

  /* Estilos para pantallas con un ancho entre 768px y 1024px */

}
```

2.3. Cursores: propiedad *cursor*

Valor *cursor*	Descripción
crosshair	Cursor tipo cruz para tareas en las que se requiere precisión.
help	Interrogación. Cursor de ayuda.
move	Flechas hacia todos lados. Cursor para mover elementos.
pointer	Mano o apuntador. Cursor para hacer clic.
progress	Barra progreso o similar. Cursor que indica que se está trabajando en segundo plano.
text	Cursor que permite seleccionar texto.
wait	Cursor que indica que se debe esperar.
vertical-text	Flecha vertical, utilizado para texto en orientación vertical.
alias	Flecha diagonal, utilizado para indicar que se puede acceder a un enlace o recurso externo.
no-drop	Círculo con una línea diagonal, utilizado para indicar que no se permite soltar el elemento arrastrado.

Valor *cursor*	Descripción
grab	Mano cerrada, utilizado para indicar que el elemento se puede agarrar
grabbing	Mano abierta, utilizado para indicar que el elemento está siendo arrastrado o agarrado.
zoom-in	Símbolo de lupa con un signo de más, utilizado para indicar que se puede hacer zoom in.
zoom-out	Símbolo de lupa con un signo de menos, utilizado para indicar que se puede hacer zoom out.
not-allowed	Círculo con una línea diagonal, utilizado para indicar que la interacción no está permitida.
cell	Cursor tipo cruz, utilizado en hojas de cálculo para ajustar celdas.
copy	Puntero con un símbolo de copia, utilizado para indicar que se puede realizar una copia de un elemento.
ew-resize	Flecha bidireccional horizontal, utilizado para indicar que se puede ajustar el tamaño horizontalmente.
ns-resize	Flecha bidireccional vertical, utilizado para indicar que se puede ajustar el tamaño verticalmente.
nwse-resize	Flecha bidireccional diagonal, utilizado para indicar que se puede ajustar el tamaño diagonalmente en dirección noroeste-sureste.
nesw-resize	Flecha bidireccional diagonal, utilizado para indicar que se puede ajustar el tamaño diagonalmente en dirección noreste-suroeste.

Valor *cursor*	Descripción
col-resize	Flecha bidireccional horizontal con una línea vertical, utilizado para indicar que se puede ajustar el tamaño de una columna.
row-resize	Flecha bidireccional vertical con una línea horizontal, utilizado para indicar que se puede ajustar el tamaño de una fila.
context-menu	Puntero con un símbolo de menú, utilizado para indicar que se puede abrir un menú contextual.
none	No se muestra ningún cursor.
Personalized	Imagen o valor personalizado en el cursor.

Tabla 19. Tipos de cursores CSS

2.4. Sombras CSS

Propiedad	Valores	Ejemplo
`text-shadow`	posx posy desenfoque color | none	`text-shadow: 2px 2px 5px red;`
`box-shadow`	posx posy desenfoque posición color inset | none	`box-shadow: 2px 2px 2px 1px rgba(0, 0, 0, 0.2);`

Tabla 20. Sombras

2.5. Selector de atributos CSS

Selector	Descripción
[nombre_atributo]	Selecciona los elementos que tienen establecido el atributo llamado `nombre_atributo` independientemente de su valor

Selector	Descripción
[nombre_atributo=valor]	Selecciona los elementos que tienen establecido un atributo llamado `nombre_atributo` con un valor igual a `valor`
[nombre_atributo~=valor]	Selecciona los elementos que tienen establecido un atributo llamado `nombre_atributo` y al menos uno de los valores del atributo es `valor`
[nombre_atributo|=valor]	Selecciona los elementos que tienen establecido un atributo llamado `nombre_atributo`, cuyo valor es una lista de valores, donde alguno comienza por `valor`
[nombre_atributo$=valor]	Selecciona aquellas etiquetas cuyo atributo acabe en ese valor
[nombre_atributo^=valor]	Selecciona aquellas etiquetas cuyo atributo comience por ese valor

Tabla 21. Selector de atributos

A continuación se muestran algunos ejemplos de estos tipos de selectores:

Atributo	Descripción
`[href]`	El atributo `href` existe en la etiqueta.
`[href="#"]`	El atributo `href` existe y su valor es igual al texto `#`.
`[href^="https://"]`	El atributo `href` existe y su valor comienza por `https://`.
`[href$=".pdf"]`	El atributo `href` existe y su valor termina por `.pdf` (es un enlace a un PDF).
`[class~="eniun"]`	El atributo `class` contiene una lista de valores, que contiene `eniun`.
`[lang|="es"]`	El atributo `lang` contiene una lista de valores, donde alguno empieza por `es`.
not([class])	El atributo class no existe en la etiqueta.

Selecciona aquellas imágenes con extensión png:

```
img[src$=".png"]
```

Se muestran de color azul todos los enlaces que tengan un atributo «class», independientemente de su valor:

```
a[class] { color: blue; }
```

Se muestran de color azul todos los enlaces que tengan un atributo «class» con el valor «externo»:

a[class="externo"] { color: blue; }

Se muestran de color azul todos los enlaces que apunten al sitio «http://www.ejemplo.com».

a[href="http://www.ejemplo.com"] { color: blue; }

Selecciona todos los elementos de la página cuyo atributo «lang» sea igual a «en», es decir, todos los elementos en inglés

*[lang=en] { ... }

Selecciona todos los elementos de la página cuyo atributo «lang» empiece por «es», es decir, «es», «es-ES», «es-AR», etc.

*[lang|="es"] { color : red }

Selecciona todos los enlaces que no tengan el atributo href.

a:not([href]) { color: purple; }

Ejemplo

Utilizando selectores de atributos realiza los siguientes ejercicios:

- Selecciona aquellas imágenes con extensión png y añade un borde de 5 píxeles de color rojo.
- Muestra de color verde todos los enlaces que tengan un atributo “class”, independientemente de su valor.
- Muestra de color rosa todos los enlaces que apunten al sitio “https://www.eniun.com”.
- Muestra de color lila todos los enlaces que tengan un atributo “class” con el valor “externo”.

```
img[src$=".png"]{border: 5px solid red;}
a[class] { color: green; }
a[href="https://www.eniun.com"] { color: pink; }
a[class~="externo"] { color: purple; }
```

2.6. Formatos de las fuentes

Formato	Descripción
EOT	Embedded OpenType, Explorer
TTF	TrueType Font, IOS
WOFF	Web Open Font Format, Chrome
WOFF2	Mejora de WOFF
SVG	Scalar Vector Graphics

Tabla 22. Formatos de las fuentes en CSS

2.7. Flexbox CSS

Propiedad	Descripción	Valores Ejemplo
display	Define el contenedor como un flex container	flex
flex-wrap	Controla si los elementos flex deben trasladarse a la siguiente fila	nowrap (por defecto), wrap, wrap-reverse
justify-content	Alineación horizontal de los elementos de un contenedor flexible	flex-start (por defecto), flex-end, center, space-between, space-around, space-evenly
align-items	Alineación vertical de los elementos de un contenedor flexible	stretch (por defecto), flex-start, flex-end, center, baseline
flex-direction	Dirección de los elementos	row (por defecto), row-reverse, column, column-reverse

Propiedad	Descripción	Valores Ejemplo
flex-grow	Controla cómo los elementos flex se expanden	Número (valor relativo)
flex-shrink	Controla cómo los elementos flex se encogen	Número (valor relativo)
flex-basis	Tamaño inicial de los elementos flex	Valor, `auto` (por defecto)
order	Orden de los elementos flex dentro del contenedor	Número (entero)

Tabla 23. Propiedades Flexbox

3. Tabla resumen de propiedades CSS Nivel Avanzado

3.1. Variables CSS o propiedades personalizadas

```
:root {
  --principal-color: black; /* Valor personalizado */
  --font: "Arial";
}
.clase {
  background-color: var(--principal-color);
  font-family: var(--font);
}
```

3.2. Gradiente linear y radial CSS

Los gradientes nos permiten añadir efectos de colores degradados en nuestros diseños. Los gradientes están disponibles desde CSS3 y se configuran como fondos, por lo que tendremos que usar la propiedad "**background**". **Disponemos de dos tipos de gradientes: lineal y radial.**

El gradiente lineal se define mediante la función linear-gradient() y permite crear fondos degradados en una dirección específica.

El formato es el siguiente:

background: linear-gradient(dirección|ángulo, color, color, color...);

background: linear-gradient(dirección|ángulo, color posición, color posición, color posición...);

background: linear-gradient(color, color, color…);

Significado de las propiedades:

- **Dirección | ángulo:** indica la dirección o ángulo del gradiente. Puede ser especificado usando las palabras clave *to top, to bottom, to left* y *to right*. También puede ser indicado por un ángulo para declarar una dirección específica del gradiente.
- **Color:** permite crear una lista con varias paradas para ir cambiando el color del gradiente. Se pueden definir tantos colores como se necesiten. Se puede utilizar el nombre de color (en inglés), valor hexadecimal, color en RGB o HSL.
- **Posición:** posición a la que comienza a cambiar el color del gradiente. Suele indicarse en porcentajes o píxeles.

Ejemplo de 4 div en html:

```
<div></div>
<div></div>
<div></div>
<div></div>
```

En css

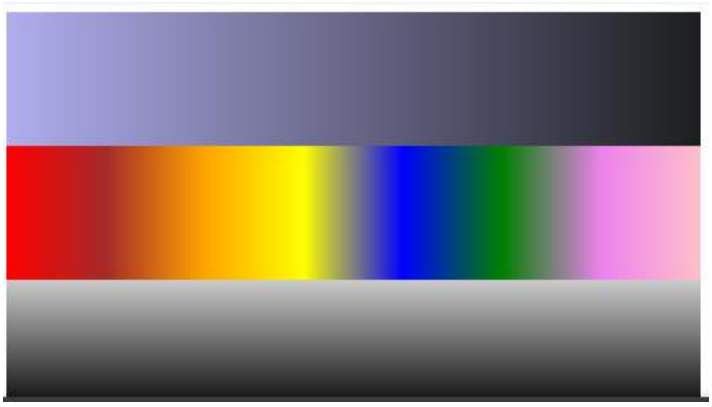
```
background: linear-gradient(90deg, rgba(176,174,238,1) 0%, rgba(29,30,31,1));
background: linear-gradient(to right, red,brown,orange,yellow,blue,green,violet,pink);
background: linear-gradient(to top, #000, #ccc);
background: linear-gradient(to right,red 300px,brown 400px,orange 700px);
```

```
div:nth-child(1){ background: linear-gradient(90deg, rgba(176,174,238,1) 0%, rgba(29,30,31,1));}

div:nth-child(2){ background: linear-gradient(to right,red,brown,orange,yellow,blue,green,violet,pink);}

div:nth-child(3){ background: linear-gradient(to top, #000, #ccc);}
```

div{padding:40px;}

div:nth-child(4){ background: linear-gradient(to right,red 300px,brown 400px,orange 700px);}

Ejecución :



Propiedad	Valores	Ejemplo
background	linear-gradient(dirección{*to top, to bottom, to left* y *to right*}|ángulo, color, color, color…)	background: linear-gradient(to right, red, orange);
background	linear-gradient(dirección{*to top, to bottom, to left* y *to right*}|ángulo, color posición, color posición, color posición…)	background: linear-gradient(to right, red 20%, orange 40%);
background	linear-gradient(color, color, color…)	background: linear-gradient(red, orange);
background	radial-gradient(forma{*circle* y *ellipse*}, color, color, color…)	background: radial-gradient(circle, red, orange);
background	radial-gradient(forma{*circle* y *ellipse*} posición, color, color, color…)	background: radial-gradient(circle closest-side, red, orange);
background	radial-gradient(forma{*circle* y *ellipse*}, color posición, color posición, color posición…)	background: radial-gradient(circle at top left, red 20%, orange 40%);
background	radial-gradient(color, color, color…)	background: radial-gradient(red, orange, blue);

Propiedad	Valores	Ejemplo
background	radial-gradient(*farthest-side*, color, color, …)	background: radial-gradient(farthest-side, *red*, *blue*);

Tabla 24. Gradientes

3.3. Transiciones CSS

```
transition:[propiedad a modificar] [Duración] [Tipo de animación] [Retardo];
```

Todos los parámetros para aplicar las transiciones se pueden establecer en una sola línea y también mediante propiedades por separado. Veamos cómo se implementa en una sola línea mediante la propiedad abreviada **transition**.

Formato:

```
transition:[propiedad a modificar] [Duración] [Tipo de animación] [Retardo];
```

Significado de las propiedades:

[Propiedades a modificar]: algunas de las propiedades que podemos modificar utilizando transiciones son las que se muestran en la siguiente lista:

- background-color
- border
- border-radius
- color
- top
- bottom
- left
- right
- box-shadow
- width
- height
- line-height
- margin
- opacity
- word-spacing
- letter-spacing
- fill
- padding
- stroke
- text-shadow

- vertical-align
- visibility
- z-index
-

[Duración en segundos]: se debe especificar el número de segundos que va a durar la animación. Por ejemplo: 3s (3 segundos).

[Tipo de animación]: **esta propiedad es opcional y sirve para indicar la velocidad de la animación.** Algunas de las posibilidades son las siguientes:

- linear: la velocidad de la animación es uniforme en todo el recorrido.
- ease: la velocidad se acelera al inicio, luego se retarda un poco y se acelera al final de nuevo.
- ease-in: la animación empieza lentamente y va aumentando progresivamente.
- ease-out: la animación empieza muy rápida y va descendiendo progresivamente.
- ease-in-out: la animación empieza y acaba lentamente, y es en la parte central del recorrido donde la velocidad es más rápida.

[Retardo]: tiempo (en segundos) que el navegador esperará antes de poner en marcha la animación. Se especifica el número de segundos a esperar seguido de la «s» (ejemplo: 1s).

Ejemplo:

```
.caja1{
  background-color: #C0392B;
  transition: background-color 1s linear;
}
.caja1:hover{
  background-color: #3F51B5;
}
```

Además de la propiedad reducida transition, **también podemos utilizar las propiedades específicas que permiten ajustar más detalles de las transiciones, como transition-duration, transition-delay, transition-timing-function y transition-property**, que se pueden declarar individualmente para definir cada aspecto de la animación.

3.4. Transformaciones CSS

transform: funcion(<valor>);

Función	Descripción	Ejemplo
Scale	Se establece con uno o dos valores, que representan la cantidad de escala que se aplica en cada dirección: `scale(x)` o `scale(x,y)`. Cuando el valor está fuera del rango [-1, 1], el elemento crece a lo largo de esa dimensión. Cuando está dentro del rango el elemento se encoge.	`transform: scale(0.5)`
Translate	Cambia la posición del elemento hacia la izquierda, derecha, arriba o abajo. La función `translate()` se establece con uno o dos valores: `translate(x)` o `translate(x,y)`. Sus valores pueden estar definidos en píxeles, porcentajes,…	`transform: translate(10px)`
Rotate	Gira o rota los elementos en grados: `rotate(v)`.	`transform: rotate(45deg)`
Skew	Distorsiona los elementos según el ángulo en grados. La función `skew()` se establece con uno o dos valores: `skew(x)` o `skew(x,y)`.	`transform: skew(45deg)`
Matrix	Mueve o transforma los elementos con precisión de píxel. La función `matrix()` se establece con seis valores numéricos: `matrix(a,b,c,d,x,y)`. Los dos últimos valores representan la translación y los primeros la transformación lineal.	`transform: matrix(0.5, 0.1, 0.5, 1, 10, -2)`

Tabla 25. Transformaciones CSS

La propiedad `transform` se usa junto con la propiedad `transition`. Ejemplo:

```
.caja1{
-webkit-transition: 1s linear;
transition: 1s linear;
}
.caja1:hover{
-webkit-transform: scale(.5);
transform: scale(.5);
}
```

.caja1{ -webkit-transition: 1s linear; transition: 1s linear; } .caja1:hover{ -webkit-transform: scale(.5); transform: scale(.5); }

```
.caja1{

  -webkit-transition: 1s linear;

  transition: 1s linear;

}

.caja1:hover{

  -webkit-transform: scale(.5);

  transform: scale(.5);

}
```

Tipos de transformaciones para la propiedad transform

Las transformaciones que podemos aplicar son las siguientes:

- **Scale**: modifica el tamaño de los elementos. La función scale () se establece con uno o dos valores, que representan la **cantidad de escala que se aplica en cada dirección**: **scale (x) o scale(x,y)**. Se define mediante un **valor numérico** de manera que cuando un valor de coordenadas está fuera del rango [-1, 1], el elemento crece a lo largo de esa dimensión. Cuando está dentro del rango el elemento se encoge.

  transform: **scale**(0.5); /* Escala el elemento a la mitad */

- **Translate**: cambia la posición del elemento hacia la izquierda, derecha, arriba o abajo. La función translate() se establece con uno o dos valores: **translate(x) o translate(x,y)**. Los valores x e y son los vectores de translación en las coordenadas x e y. Sus valores pueden estar definidos en píxeles, porcentajes,…

  transform: **translate**(10px); /* Traslada el elemento 10px hacia la derecha */

- **Rotate**: gira o rota los elementos en grados: **rotate(v)**.

  transform: **rotate**(45deg); /* Rota el elemento 45 grados */

- **Skew**: distorsiona los elementos según el **ángulo en grados**. La función skew() se establece con uno o dos valores: **skew(x) o skew(x,y).**

  transform: **skew**(45deg); /* Distorsiona el elemento 45 grados en el eje x */

- **Matrix**: mueve o transforma los elementos con precisión de píxel. La función matrix() se establece con seis valores numéricos: matrix(a,b,c,d,x,y). Los dos últimos valores representan la translación y los primeros la transformación lineal.

  transform: **matrix**(0.5, 0.1, 0.5, 1, 10, -2);

La propiedad transform se usa junto con la propiedad transition vista en la sección anterior para que la transformación pueda tener una transición espaciada en el tiempo:

```
.caja1{

  -webkit-transition: 1s linear;

  transition: 1s linear;

 }

 .caja1:hover{

  -webkit-transform: scale(.5);

  transform: scale(.5);

 }
```

3.5. Animation CSS

Sintaxis propiedad abreviada:

```
selector {
animation: name duration timing-function delay iteration-count direction fill-mode play-state;
}
selector { animation: name duration timing-function delay iteration-count direction fill-mode play-state; }
selector {
  animation: name duration timing-function delay iteration-count direction fill-mode play-state;
}
```

1. `name`: Nombre de la regla `@keyframes` que describe los fotogramas de la animación.
2. `duration`: Cantidad de tiempo que la animación consume en completar su ciclo. Se expresa en segundos (s) o milisegundos (ms).
3. `timing-function`: Ritmo de la animación, es decir, cómo se muestran los fotogramas a lo largo del tiempo. Puede ser lineal, ease, ease-in, ease-out, ease-in-out, cubic-bezier, step-start, step-end, steps, etc.
4. `delay`: Tiempo de espera antes de que la animación comience a ejecutarse. Se expresa en segundos (s) o milisegundos (ms).

5. `iteration-count`: Número de veces que se repetirá la animación. Puede ser un número entero, `infinite` para que se repita infinitamente, o un valor específico como `2`, `3`, etc.
6. `direction`: Define si la animación debe retroceder hasta el fotograma de inicio al finalizar la secuencia. Puede ser `normal`, `reverse`, `alternate`, o `alternate-reverse`.
7. `fill-mode`: Especifica qué valores tendrán las propiedades después de finalizar la animación. Puede ser `none`, `forwards`, `backwards`, o `both`.
8. `play-state`: Permite pausar y reanudar la secuencia de la animación. Puede ser `running` o `paused`.

Sintaxis subpropiedades:

selector{

animation-name: identifier;

subproperty: value;

}

selector{ animation-name: identifier; subproperty: value; }

```
selector{

    animation-name: identifier;

    subproperty: value;

}
```

Las subpropiedades de `animation` son:

- `animation-name`
- `animation-duration`
- `animation-timing-function`
- `animation-delay`
- `animation-iteration-count`
- `animation-direction`
- `animation-fill-mode`
- `animation-play-state`

Diferencias entre las propiedades animation, transition y transform

En una sección anterior vimos cómo hacer **transiciones** y **transformaciones** en CSS3 sobre los elementos de nuestra página web. Veamos las diferencias entre esas dos propiedades y animation.

- **La propiedad transition en CSS** permite realizar cambios suaves entre dos estados de un elemento cuando ocurre un evento, como pasar el cursor sobre él (:hover). Es ideal para efectos básicos como cambios en el color, tamaño o posición, pero solo anima entre un estado inicial y final, sin pasos intermedios.

- **La propiedad transform** modifica instantáneamente la apariencia de un elemento, permitiendo rotarlo, escalarlo o moverlo en el espacio. Aunque transform no genera animaciones por sí mismo, puede combinarse con transition o animation para lograr efectos más elaborados.

- **La propiedad animation** permite definir múltiples estados intermedios en una animación y controlar aspectos como la duración, la repetición y la dirección. A diferencia de transition, las animaciones pueden ejecutarse automáticamente al cargar la página o repetirse indefinidamente sin requerir la interacción del usuario.

Animaciones CSS con propiedad animation (propiedad abreviada)

En este apartado veremos **cómo encadenar diferentes animaciones utilizando la propiedad animation y sus subpropiedades**. Para ello, **definiremos la propiedad animation sobre nuestros selectores y después definiremos nuestra secuencia de animación mediante @keyframes.**

Cuando trabajamos con animaciones en CSS, podemos optar por usar la propiedad abreviada animation, que incluye todas las configuraciones en una sola línea, o podemos utilizar las subpropiedades de animation de forma individual para tener un mayor control sobre cada aspecto de la animación.

Sintaxis propiedad animation

La propiedad animation engloba todas las subpropiedades relacionadas con la animación en CSS, por lo que sería la propiedad abreviada. Las subpropiedades permiten controlar aspectos específicos de la animación, como su nombre, duración, dirección, relleno, etc.

selector {

animation: name duration timing-function delay iteration-count direction fill-mode play-state;

}

1. name: Especifica el nombre de la regla @keyframes que describe los fotogramas de la animación.

2. duration: Indica la cantidad de tiempo que la animación consume en completar su ciclo. Se expresa en segundos (s) o milisegundos (ms).

3. timing-function: Define el ritmo de la animación, es decir, cómo se muestran los fotogramas a lo largo del tiempo. Puede ser lineal, ease, ease-in, ease-out, ease-in-out, cubic-bezier, step-start, step-end, steps, etc.

4. delay: Especifica un tiempo de espera antes de que la animación comience a ejecutarse. Se expresa en segundos (s) o milisegundos (ms).

5. iteration-count: Indica el número de veces que se repetirá la animación. Puede ser un número entero, infinite para que se repita infinitamente, o un valor específico como 2, 3, etc.

6. direction: Define si la animación debe retroceder hasta el fotograma de inicio al finalizar la secuencia. Puede ser normal, reverse, alternate, o alternate-reverse.

7. fill-mode: Especifica qué valores tendrán las propiedades después de finalizar la animación. Puede ser none, forwards, backwards, o both.

8. play-state: Permite pausar y reanudar la secuencia de la animación. Puede ser running o paused.

Subpropiedades de animation

Si optamos por usar las subpropiedades, comenzamos añadiendo la propiedad animation-name al selector, a la cual le asignamos un nombre identificativo, denominado identifier. Luego, podemos utilizar las demás subpropiedades para ajustar diferentes aspectos de la animación, como la duración, dirección, entre otros.

Sintaxis:

selector{

animation-name: identifier;

subproperty: value;

}

Las subpropiedades de animation son:

- animation-nameEspecifica el nombre de la regla @keyframes que describe los fotogramas de la animación.

- animation-delayTiempo de retardo entre el momento en que el elemento se carga y el comienzo de la secuencia de la animación.

- animation-directionIndica si la animación debe retroceder hasta el fotograma de inicio al finalizar la secuencia.

- animation-durationIndica la cantidad de tiempo que la animación consume en completar su ciclo (duración).

- animation-iteration-countEl número de veces que se repite. Podemos indicar infinite para repetir la animación indefinidamente.

- animation-play-statePermite pausar y reanudar la secuencia de la animación.

- animation-timing-functionIndica el ritmo de la animación, es decir, como se muestran los fotogramas de la animación, estableciendo curvas de aceleración.

- animation-fill-modeEspecifica qué valores tendrán las propiedades después de finalizar la animación (los de antes de ejecutarla, los del último fotograma de la animación o ambos).

Valores de las subpropiedades de animation

Veamos los posibles valores de las subpropiedades de animation y ejemplos de uso:

- **animation-name**
  Esta subpropiedad vincula el selector con una regla @keyframes. Los identificadores o nombres que utilizamos deben ser únicos.

  .mi-elemento {
  animation-name: mi-animacion;
  }
  @keyframes mi-animacion {
  from { opacity: 0; }
  to { opacity: 1; }
  }

- **animation-delay**
  Determina un tiempo de espera antes de que la animación comience.

  .mi-elemento {
  animation-delay: 500ms; /* Espera medio segundo antes de comenzar */
  }

- **animation-direction**
  Indica si la animación debe alternar la dirección en cada ciclo. Valores: normal, reverse, alternate, alternate-reverse.

  .mi-elemento {
  animation-direction: alternate; /* Va y viene */
  }

- **animation-duration**
  Establece cuánto tiempo tarda en completar un ciclo de la animación, se expresa en segundos (s) o milisegundos (ms).

  .mi-elemento {

  animation-duration: 2s; /* Completa la animación en 2 segundos */
  }

- **animation-iteration-count**
  Define cuántas veces se repetirá la animación. infinite para una animación sin fin.

  .mi-elemento {

  animation-iteration-count: infinite; /* Repite la animación indefinidamente */

  }

- **animation-play-state**
  Permite pausar y reanudar la reproducción de la animación.
  Valores: running, paused.

  .mi-elemento*:hover* {

  animation-play-state: paused; /* Pausa la animación al pasar el mouse */

  }

- **animation-timing-function**
  Controla el ritmo de la animación, definiendo cómo se acelera y desacelera durante su ejecución.
  Valores: linear, ease, ease-in, ease-out, ease-in-out, cubic-bezier(n,n,n,n).

  .mi-elemento {

  animation-timing-function: ease-in-out; /* Comienza lentamente, se acelera en el medio, y termina lentamente */

  }

- **animation-fill-mode**
  Decide los estilos que se aplican a un elemento antes y después de su animación.
  Valores: none, forwards, backwards, both.

  .mi-elemento {

  animation-fill-mode: forwards; /* Mantiene los estilos del último keyframe después de finalizar */

  }

Animaciones CSS: Uso de @keyframes

Una vez añadida la propiedad animation a nuestro selector, haremos uso de los @keyframes para crear animaciones completas.

Los **@keyframes** son un conjunto de fotogramas clave que describen cómo se muestra cada elemento animado durante la secuencia de la animación. La sintaxis es la siguiente:

```
@keyframes identifier {
 from {
    ...
 }
 percentage {
    ...
 }
 to {
    ...
 }

}
```

- identifier Nombre que identifica la lista de keyframe. Debe coincidir con animation-name.
- fromDesde (por ejemplo: desde 0%).
- toHasta (por ejemplo hasta 100%).
- percentage Porcentaje intermedio de las veces que va a ocurrir una animación.

Ejemplos animaciones CSS con propiedad animation

Veamos mejor las propiedades y valores explicados anteriormente mediante varios ejemplos.

Rotación de forma indefinida

Vamos a hacer una animación mediante **@keyframes** de nuestro logo, de forma que se encuentre rotando de forma indefinida. Para ello vamos a utilizar la propiedad **transform** y le daremos varios valores a su rotación: crearemos una animación con tres **keyframes**: uno inicial que tendrá rotación de 0 grados, otro en el 50% que tendrá una rotación sobre el eje y de 180 grados y uno final que volverá a tener una rotación de 0 grados.

Código html:

```
<img class="animacionLogo" src="eniun.png">
```

Código CSS:

```
.animacionLogo {
    animation-duration: 5s;
    animation-name: animLogo;
    animation-iteration-count: infinite;
```

```
  }
@keyframes animLogo{
   from {
      -webkit-transform: rotate(0deg);
      transform: rotateY(0deg);
   }
   50%{
      -webkit-transform: rotate(180deg);
      transform: rotateY(180deg);
   } to {
      -webkit-transform: rotate(360deg);
      transform: rotateY(360deg);
      }
}
```

Ejemplo completo en CodePen:

Deslizar un texto por el navegador

Ahora haremos una animación de un párrafo que se deslizará por el navegador desde el borde derecho de la ventana:

Código CSS:

```
p {
  animation-duration: 4s;
  animation-name: animTexto;
  margin: 20px;
}
@keyframes animTexto {
  from {
    margin-left: 100%;
    width: 100%;
  }
  to {
    margin-left: 0%;
    width: 100%;
  }
}
```

3.6. Funciones matemáticas en CSS

Función	Descripción	Valores Ejemplo
abs()	Devuelve el valor absoluto de un número	`abs(-10)` = 10
sin()	Devuelve el seno de un ángulo	`sin(45deg)` ≈ 0.71
cos()	Devuelve el coseno de un ángulo	`cos(60deg)` = 0.5
tan()	Devuelve la tangente de un ángulo	`tan(30deg)` ≈ 0.58
sqrt()	Devuelve la raíz cuadrada de un número	`sqrt(16)` = 4
pow()	Eleva un número a una potencia	`pow(2, 3)` = 8
min()	Devuelve el valor mínimo entre dos números	`min(10, 5)` = 5
max()	Devuelve el valor máximo entre dos números	`max(10, 5)` = 10
random()	Devuelve un número aleatorio entre 0 y 1	`random()` ≈ 0.35
floor()	Devuelve el valor entero menor o igual	`floor(3.8)` = 3
ceil()	Devuelve el valor entero mayor o igual	`ceil(3.2)` = 4
round()	Devuelve el valor redondeado al entero más cercano	`round(3.7)` = 4
clamp()	Limita un valor dentro de un rango	`clamp(10, 5, 8)` = 8
calc()	Realiza cálculos matemáticos en propiedades CSS	`width: calc(100% - 20px)`
attr()	Obtiene el valor de un atributo HTML y lo utiliza en CSS	`content: attr(data-text)`

Tabla 26. Funciones matemáticas en CSS

Funciones matemáticas

`abs()`	Devuelve el valor absoluto de un número	`abs(-10)` = 10
`sin()`	Devuelve el seno de un ángulo	`sin(45deg)` ≈ 0.71
`cos()`	Devuelve el coseno de un ángulo	`cos(60deg)` = 0.5
`tan()`	Devuelve la tangente de un ángulo	`tan(30deg)` ≈ 0.58
`sqrt()`	Devuelve la raíz cuadrada de un número	`sqrt(16)` = 4
`pow()`	Eleva un número a una potencia	`pow(2, 3)` = 8
`min()`	Devuelve el valor mínimo entre dos números	`min(10, 5)` = 5
`max()`	Devuelve el valor máximo entre dos números	`max(10, 5)` = 10
`random()`	Devuelve un número aleatorio entre 0 y 1	`random()` ≈ 0.35
`floor()`	Devuelve el valor entero menor o igual	`floor(3.8)` = 3
`ceil()`	Devuelve el valor entero mayor o igual	`ceil(3.2)` = 4
`round()`	Devuelve el valor redondeado al entero más cercano	`round(3.7)` = 4
`clamp()`	Limita un valor dentro de un rango	`clamp(10, 5, 8)` = 8
`calc()`	Realiza cálculos matemáticos en propiedades CSS	`width: calc(100% - 20px)`
`attr()`	Obtiene el valor de un atributo HTML y lo utiliza en CSS	`content: attr(data-text)`

Ejemplos:

width: calc(100px + 20px); /* Resultado: 120px */

width: calc(80% - 10px); /* Resultado: El 80% del ancho disponible menos 10px */

width: calc((100px + 10%) / 3); /* Resultado: El resultado de la operación combinada */

width: max(200px, 50%); /* Resultado: El valor máximo entre 200px y el 50% del ancho disponible */

height: max(100px, 150px, 50%); /* Resultado: El valor máximo entre 100px, 150px y el 50% del alto disponible */

font-size: max(16px, 1.2em); /* Resultado: El valor máximo entre 16px y 1.2 veces el tamaño de fuente actual */

padding: round(-2.3em); /* Resultado: -2em */

font-size: round(1.2345, 2); /* Resultado: 1.23 */

width: round(3.7rem); /* Resultado: 4rem */

width: attr(data-width); Obtener el valor del atributo y utilizarlo como ancho.

3.7. CSS Grid

Propiedad	Descripción	Valores Ejemplo
display	Define el contenedor como un grid container	`grid` (por defecto), `inline-grid`
grid-template-columns	Define las columnas del grid	`grid-template-columns: 1fr 2fr 1fr`
grid-template-rows	Define las filas del grid	`grid-template-rows: 100px 200px`
grid-gap	Propiedad abreviada para establecer `grid-column-gap` y `grid-row-gap`	`grid-gap: 10px`
grid-column-gap	Establece el tamaño del espacio entre columnas	`grid-column-gap: 20px`
grid-row-gap	Establece el tamaño del espacio entre filas	`grid-row-gap: 10px`

Tabla 27. CSS Grid

CSS Grid *nos permite maquetar contenido ajustándolo a cuadrículas o rejillas (grids) totalmente configurables mediante estilos CSS.*

Mediante CSS Grid podemos dividir la página en una rejilla a partir de la cual se pueden posicionar los diferentes elementos de manera muy sencilla. Gracias a los sistemas de maquetación de CSS Grid y CSS Flexbox se pueden crear estructuras con menos código y de una forma más fácil que con los métodos tradicionales.

La diferencia básica entre CSS Grid y CSS Flexbox es que Flexbox se creó para diseños de una dimensión, en una fila o una columna. En cambio CSS Grid Layout se pensó para el diseño bidimensional, en varias filas y columnas al mismo tiempo.

Cómo utilizar CSS Grid

Para utilizar **CSS Grid** definiremos un contenedor padre y en su interior los ítems que se necesiten crear. Además, se debe definir una serie de propiedades, tanto para el contenedor padre como para las rejillas que contiene. Veamos en detalle las propiedades más importantes.

Propiedad display: grid | inline-grid

Para crear la cuadrícula **grid** hay que definir sobre el elemento contenedor la propiedad display y especificar el valor **grid** o **inline-grid**.

```
<!DOCTYPE html>
<html>
<head>
<style>
.contenedor{
  display: grid;
}
</style>
</head>
<body>
<div class="contenedor"> <!-- contenedor padre-->
  <div class="rejilla">Item 1</div> <!-- Ítems del grid -->
  <div class="rejilla">Item 2</div>
  <div class="rejilla">Item 3</div>
  <div class="rejilla">Item 4</div>
</div>
</body>
</html>
```

Los valores **inline-grid** y **grid** indican cómo se comporta el contenedor con respecto al contenido exterior. Con el valor inline-grid el contenedor aparece en línea con respecto al contenido exterior. Con el valor grid, el contenedor aparece en bloque con respecto al contenido exterior.

Valor	Descripción
inline-grid	Cuadrícula en línea con respecto al contenido exterior
grid	Cuadrícula en bloque con respecto al contenido exterior

```
<h2>display: inline-grid;</h2>
<b>Contenido exterior</b>
<div class="contenedor"> <!-- contenedor padre-->
  <div class="rejilla">Item 1</div> <!-- Ítems del
grid -->
  <div class="rejilla">Item 2</div>
  <div class="rejilla">Item 3</div>
  <div class="rejilla">Item 4</div>
</div>
<h2>display: grid;</h2>
<b>Contenido exterior</b>
<div class="contenedor2"> <!-- contenedor padre-->
  <div class="rejilla">Item 1</div> <!-- Ítems del
grid -->
```

```
  <div class="rejilla">Item 2</div>
  <div class="rejilla">Item 3</div>
  <div class="rejilla">Item 4</div>
</div>
```

CSS:

```
.contenedor{
  display: inline-grid;
}
.contenedor2{
  display: grid;
}
```

Propiedades grid-template-colums y grid-template-rows
Para definir el tamaño de las columnas y las filas usamos las propiedades grid-template-colums y grid-template-rows.

Propiedad	Valor	Descripción
`grid-template-columns`	[*col1*] [*col2*] ...	Tamaño de cada columna
`grid-template-rows`	[*fila1*] [*fila2*] ...	Tamaño de cada fila

Como vemos a continuación, si definimos los siguientes valores crearemos una cuadrícula de 2 filas por 3 columnas.

```
1.  .contenedor {
2.    display: grid;
3.    grid-template-rows: 200px 200px;
4.    grid-template-columns: 200px 200px 200px;
5.  }
```

Podemos usar las **unidades que ya conocemos como los píxeles o los porcentajes,** o utilizar la **unidad fr (unidad fraccional propia del sistema CSS Grid)** que indica la proporción del espacio que ocupará cada elemento. Por ejemplo, si definimos 3 columnas y le asignamos un valor 1fr a cada una, esto repartirá el espacio a partes iguales entre las 3. En otro caso, si definimos el valor 2fr a una de ellas, esa ocupará el doble de espacio que las otras.

```
1.  .contenedor{
2.    display: grid;
3.    grid-template-columns: auto auto auto auto;
4.  }
5.  .contenedor2{
6.    display: grid;
7.    grid-template-columns: 50px 200px 300px 500px;
8.  }
9.  .contenedor3{
10.   display: grid;
11.   grid-template-columns: 1fr 2fr 1fr 3fr;
12. }
```

html

```
<h2> grid-template-columns: auto auto auto auto;</h2>
<div class="contenedor"> <!-- contenedor padre-->
<div class="rejilla">Item 1</div> <!-- Ítems del grid -->
<div class="rejilla">Item 2</div>
<div class="rejilla">Item 3</div>
<div class="rejilla">Item 4</div>
</div>
<h2> grid-template-columns: 50px 200px 50px 500px;</h2>
<div class="contenedor2"> <!-- contenedor padre-->
<div class="rejilla">Item 1</div> <!-- Ítems del grid -->
<div class="rejilla">Item 2</div>
<div class="rejilla">Item 3</div>
<div class="rejilla">Item 4</div>
</div>
<h2> grid-template-columns: 1fr 2fr 1fr 3fr;</h2>
<div class="contenedor3"> <!-- contenedor padre-->
<div class="rejilla">Item 1</div> <!-- Ítems del grid -->
<div class="rejilla">Item 2</div>
<div class="rejilla">Item 3</div>
<div class="rejilla">Item 4</div>
</div>
```

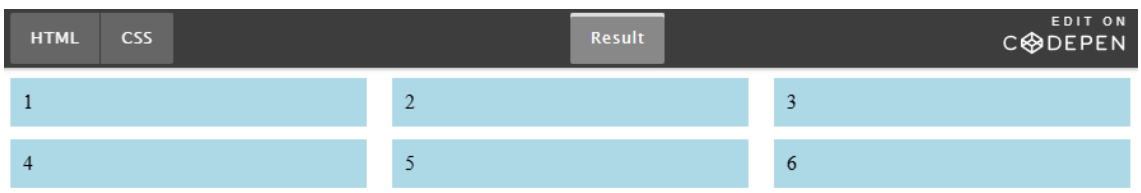
Propiedades row-gap y column-gap

Podemos definir el espaciado entre las rejillas mediante las propiedades row-gap y column-gap. El espaciado se crea entre las columnas/filas, no en los bordes exteriores: laterales, superiores e inferiores. Si quieres añadir espacio en los bordes exteriores de la rejilla, puedes usar propiedades como margin en el contenedor de la rejilla.

Propiedad	Descripción
`column-gap`	Espaciado entre columnas
`row-gap`	Espaciado entre filas

Ejemplo

En el siguiente ejemplo, hemos definido una cuadrícula con tres columnas de igual tamaño utilizando la propiedad grid-template-columns. Luego, hemos utilizado la propiedad column-gap para establecer un espacio de 20 píxeles entre las columnas y la propiedad row-gap para establecer un espacio de 10 píxeles entre las filas. También hemos agregado algunos elementos a la cuadrícula y les hemos dado un color de fondo y un relleno para que puedas ver el efecto del espacio entre las columnas y las filas



```css
.grid-container {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  column-gap: 20px;
  row-gap: 10px;
}
.grid-item {
  background-color: lightblue;
  padding: 10px;
}
```

HTML CSS

```html
<div class="grid-container">
  <div class="grid-item">1</div>
  <div class="grid-item">2</div>
  <div class="grid-item">3</div>
  <div class="grid-item">4</div>
  <div class="grid-item">5</div>
  <div class="grid-item">6</div>
</div>
```

Propiedad grid-gap

La propiedad grid-gap de CSS Grid es una propiedad abreviada para establecer tanto grid-column-gap como grid-row-gap en una sola declaración. Esta propiedad acepta uno o dos valores, donde el primer valor especifica el tamaño del espacio entre las filas y el segundo valor especifica el tamaño del espacio entre las columnas. Si solo se proporciona un valor, se aplica a ambas propiedades.

Ejemplo:

.grid-container {
display: grid;
grid-template-columns: 1fr 1fr 1fr;
grid-gap: 10px 20px;
}

En este ejemplo, hemos definido una cuadrícula con tres columnas de igual tamaño utilizando la propiedad grid-template-columns. Luego, hemos utilizado la propiedad grid-gap para establecer un espacio de 10 píxeles entre las filas y un espacio de 20 píxeles entre las columnas.

En este artículo se han visto los primeros pasos para empezar a usar CSS Grid. Además de las propiedades estudiadas, conviene conocer otras muy interesantes para conseguir alinear los elementos, cambiar su tamaño y modificar diferentes aspectos

3.8. CSS Clamp()

propiedad: clamp(valor_mínimo, valor_preferido, valor_máximo);

1. **propiedad:** Propiedad a la que se va a aplicar `clamp()`. Por ejemplo: `font-size` para modificar el tamaño de las fuentes.

2. **valor_mínimo:** Tamaño mínimo que se aplicará a la propiedad, independientemente del tamaño de la pantalla o del contenedor. Si el valor preferido es menor que este, se aplicará el valor mínimo.
3. **valor_preferido:** Valor que se aplicará de manera predeterminada a la propiedad. Este valor puede ser relativo, como en el caso de unidades como vw (viewport width) o em, lo que permite que el tamaño del texto se ajuste dinámicamente en función del tamaño de la pantalla.
4. **valor_máximo:** Tamaño máximo que se aplicará a la propiedad. Si el valor preferido es mayor que este, se aplicará el valor máximo.

Tipografía fluida CSS Clamp: Textos según el tamaño de pantalla

A los textos responsivos que se adaptan al tamaño de pantalla se les conoce como **tipografía fluida** o **tipografía responsive.** En el diseño web, la **tipografía fluida** nos ayuda a comunicarnos de forma efectiva con los usuarios. Garantizar la legibilidad del texto en diferentes dispositivos es esencial para una buena experiencia de usuario. En este artículo, aprenderemos a crear **textos que se adaptan al tamaño de pantalla mediante CSS clamp(),** eliminando la necesidad de múltiples media queries.

1. Tipografía fluida vs media queries

Históricamente, en el diseño web hemos utilizado **media queries** para ajustar el tamaño de la fuente en función del ancho de la pantalla del dispositivo. Sin embargo, esto puede resultar en **transiciones bruscas** entre diferentes tamaños de fuente en los puntos de interrupción definidos. La **tipografía fluida** aborda este problema **ajustando dinámicamente el tamaño de la fuente según el ancho de la pantalla**, lo que crea una experiencia de lectura más uniforme y agradable para los usuarios.

2. Unidades de medida relativas para lograr textos responsive

Para lograr una tipografía responsive es conveniente utilizar **unidades de medida relativas**, como em o **vh y vw,** en lugar de unidades absolutas, como píxeles (px). Las unidades relativas se ajustan dinámicamente según el tamaño de la pantalla del dispositivo, lo que garantiza que el texto se vea bien en cualquier contexto.

Por ejemplo, podemos usar la unidad relativa **vw (viewport width) junto con CSS Clamp.** De este modo, si definimos el tamaño de fuente de un elemento como **font-size: 5vw;**, el tamaño de la fuente será el **5% del ancho total** de la pantalla.

3. Uso de CSS Clamp para conseguir textos que se apapten al tamaño de pantalla

La función CSS clamp() nos permite definir un **rango de valores** para el tamaño de la fuente, incluyendo un t**amaño mínimo, máximo y un valor preferido** que se ajusta dinámicamente en relación con el ancho de la pantalla.

Sintaxis: Su sintaxis básica consiste en tres valores separados por comas.

propiedad: clamp(valor_mínimo, valor_preferido, valor_máximo);

1. **propiedad:** Es la propiedad a la que se va a aplicar clamp(). En el caso de trabajar con tipografías usaremos font-size para modificar el tamaño de las fuentes.
2. **valor_mínimo:** Es el tamaño mínimo que se aplicará a la propiedad, independientemente del tamaño de la pantalla o del contenedor. Si el valor preferido es menor que este, se aplicará el valor mínimo.
3. **valor_preferido:** Es el valor que se aplicará de manera predeterminada a la propiedad. Este valor puede ser relativo, como en el caso de unidades como vw (viewport width) o em, lo que permite que el tamaño del texto se ajuste dinámicamente en función del tamaño de la pantalla.
4. **valor_máximo:** Es el tamaño máximo que se aplicará a la propiedad. Si el valor preferido es mayor que este, se aplicará el valor máximo.

Ejemplo:

En el siguiente ejemplo, el tamaño de la fuente se ajusta entre 1em y 2.5em, con un tamaño preferido que varía en función del ancho de la pantalla. En el tamaño preferido se usa la **unidad de medida relativa vw o viewport-width**, que no es más que el ancho de la pantalla o **viewport**.

El valor 100vw es el 100% del ancho de la pantalla actual. Por lo que el tamaño de nuestra fuente preferida va a ser un **porcentaje del ancho en pixeles de la pantalla**. Por ejemplo, si decimos que nuestro tamaño preferido es de 3vw, cuando el ancho de la pantalla o viewport sea de 1000px, la fuente tendrá un tamaño de 30px. Si el ancho de la pantalla es de 700px, el tamaño de la fuente será de 21px.

CSS:

```
h2 {
  font-size: clamp(1em, 3vw, 2.5em);
}
```

Unidades de medida en clamp

El tamaño de los elementos de una web se puede expresar en unidades **absolutas** y **relativas**.

Unidades absolutas

Las **unidades absolutas** mantienen su aspecto y se visualizan siempre igual independientemente de las características del dispositivo.

px	Píxeles
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

Las **unidades relativas** se ajustan a cada tipo de dispositivo ya que dependen de la resolución de cada pantalla.

em	Relativo al tamaño de la fuente del elemento (2 em significa 2 veces el tamaño de la fuente actual)
%	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

Aunque el píxel se considera una unidad absoluta, su tamaño físico puede variar según la densidad de píxeles (PPI, píxeles por pulgada) del dispositivo de visualización. En dispositivos con una mayor densidad de píxeles, como pantallas Retina, un píxel puede ser más pequeño en términos físicos, lo que resulta en una apariencia más nítida y detallada de los elementos visuales en pantalla.

Usos de unidades

Normalmente es **recomendable usar unidades relativas** en la medida de lo posible, ya que mejora la accesibilidad de la página web y permite que los documentos se adapten fácilmente a cualquier medio. Por tanto, para la creación de una página web, el uso de medidas absolutas queda descartado.

Centímetros (cm), milímetros (mm), pulgadas (in) y puntos (pt)

Unidades de medida físicas para impresión y otros usos específicos.

Unidad em

Es especialmente útil para establecer tamaños proporcionales al tamaño de fuente. Aunque no hay un criterio definido, el organismo W3C, recomienda el uso de la unidad

em para indicar el tamaño del texto. **El tamaño de los ems se establece en base al tamaño que tenga definido el navegador.**

Usualmente **el tamaño de una fuente por defecto en los navegadores es de 16px**. Por tanto, tendríamos que **16px = 1em** y podríamos definir la siguiente conversión entre unidades.

px	em	%
12	0,750	75
14	0,875	87,5
16	1,000	100
18	1,125	112,5
20	1,250	125

Tabla Conversión em y px

Hay que tener en cuenta que el tamaño base definido en los navegadores puede ser modificado por los usuarios. Este tema lo trataremos en detalle en la unidad sobre accesibilidad web.

Existe también la unidad **REM (rem)** que es similar al em, pero se basa en el tamaño de fuente del elemento raíz (generalmente el tamaño de fuente del elemento HTML). No vamos a trabajar con esta unidad de medida.

Píxeles (px)

Es la unidad más utilizada y representa un punto en la pantalla. Se usa para **tamaños fijos** y proporciona control preciso sobre el diseño. Aunque se ha mencionado que normalmente es recomendable utilizar unidades relativas en la medida de lo posible, el píxel sigue siendo una opción muy empleada en el diseño web, especialmente cuando se requiere un diseño más estático o se necesita un control exacto sobre el tamaño de los elementos.

Porcentaje (%)

Representa una proporción del tamaño del elemento padre. Es útil para hacer diseños fluidos y responsivos teniendo en cuenta la **relación de los elementos con su contenedor padre**.

Viewport Width (vw) y Viewport Height (vh)

Representan un porcentaje del ancho y alto de la ventana del navegador, respectivamente. Son útiles para crear **diseños responsive basados en el tamaño de la pantalla**.

Flexible Grid Units (fr)

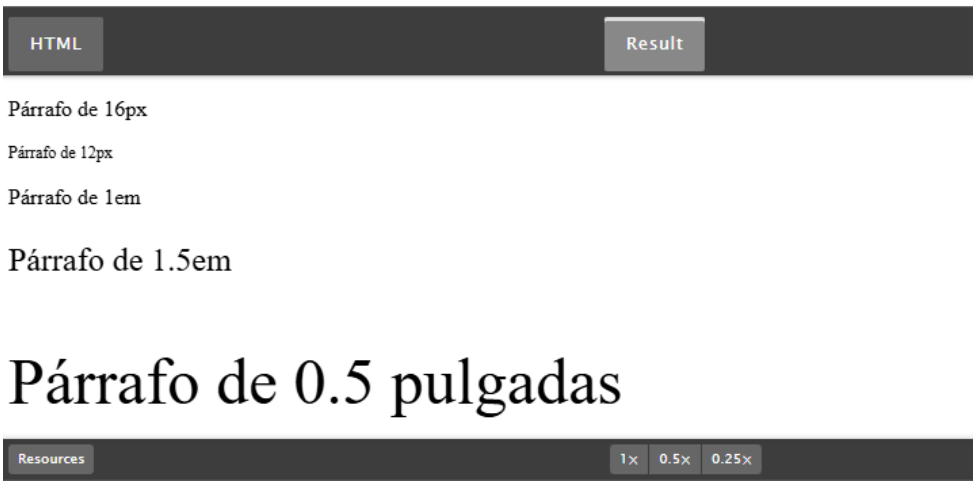
Se utiliza en Grid Layout y representa una fracción del espacio disponible en un contenedor. Es útil para distribuir el espacio disponible entre elementos flexibles.

Ejercicio propuesto

- Si el tamaño de fuente por defecto en un navegador es de 12px. ¿Cuántos em son 16px? Puedes ver la conversión de píxel a ems en el conversor online de w3schools: w3schools.com/tags/ref_pxtoemconversion.asp

- Observa el resultado del siguiente código en un navegador y estudia las diferencias que hay entre los distintos tamaños definidos.

```
<p style="font-size: 16px;">Párrafo de 16px</p>
<p style="font-size: 12px;">Párrafo de 12px</p>
<p style="font-size: 1em;">Párrafo de 1em</p>
<p style="font-size: 1.5em;">Párrafo de 1.5em</p>
<p style="font-size: 0.5in;">Párrafo de 0.5 pulgadas</p>
<p style="font-size: 8mm;">Párrafo de 8 milímetros</p>
<p style="font-size: 12pt;">Párrafo de 12 puntos</p>
```



CSS calc() para mejorar la tipografía fluida

Otra técnica para mejorar la tipografía responsive es el uso de la función calc() en CSS, que permite **realizar cálculos matemáticos** para determinar el tamaño de la fuente. Por ejemplo, podemos combinar calc() con unidades relativas.

En el siguiente ejemplo, el tamaño de la fuente será de al menos 16 píxeles, pero también aumentará en un 0.5% del ancho de la pantalla.

font-size: calc(16px + 0.5vw);

CSS clamp() junto con calc() para cálculos matemáticos y establecer límites

Es común mezclar el uso de calc() y clamp() en CSS para crear estilos más flexibles y dinámicos. Por ejemplo, podrías utilizar calc() para **realizar cálculos matemáticos** en propiedades como el ancho, el margen o el padding de un elemento, y luego usar clamp() para **establecer límites** para esas propiedades.

```
1. /* Usando calc() y clamp() juntos para el tamaño de fuente del texto */
2. .texto {
3.   font-size: clamp(12px, calc(1em + 3vw), 36px);
4.   /* En este ejemplo, el tamaño de la fuente del 'texto' se ajustará dinámicamente entre
   12px y 36px,
5.      pero nunca será menor que el tamaño base de 1em más el 3% del ancho de la ventana. */
6. }
```

En este caso, el tamaño de fuente del texto variará entre 12px y 36px, pero nunca será menor que el tamaño base de 1em más el 3% del ancho de la ventana (vw). Esto significa que a medida que el ancho de la ventana aumenta, el tamaño de la fuente del texto también aumentará, proporcionando así un diseño adaptativo.

Compatibilidad de CSS clamp() en los navegadores

Como es común en CSS, hay que **verificar la compatibilidad de las nuevas propiedades y funciones** con los diferentes navegadores para asegurar una experiencia uniforme para todos los usuarios. Al verificar la compatibilidad de la función clamp() en sitios como **Can I Use**, podemos ver que muchos navegadores modernos la admiten. Sin embargo, aún puede haber versiones anteriores de navegadores que no la soporten completamente.

Una forma de abordar la compatibilidad es mediante el **uso de feature queries**, una técnica que nos permite **verificar si un navegador es compatible con una determinada propiedad o función antes de aplicarla.** Veamos cómo podríamos implementar esto con la función clamp():

```css
h2 {
  font-size: clamp(1em, 3vw, 2.5em);
}

@supports not (font-size: clamp(1em, 3vw, 2.5em)) {
  /* Alternativa para navegadores que no admiten clamp() */
  font-size: min(max(1rem, 3vw), 2.5em);
}
```

En este ejemplo, estamos utilizando una feature query para verificar si el navegador admite la función clamp(). Si no es compatible, aplicamos una alternativa utilizando las funciones min() y max(). Esta alternativa garantiza que el tamaño de la fuente se ajuste dinámicamente dentro de un rango predefinido, incluso en navegadores que no admiten clamp().

7. Herramientas online para el uso de tipografías fluidas

Para facilitar el proceso de creación de tipografías fluidas, existen varias **herramientas online** que permiten a los diseñadores experimentar y ajustar fácilmente el comportamiento de los textos. Veamos un par de herramientas destacadas:

- **Fluidity typography editor by Adrian Bece:** Esta herramienta te permite insertar los valores máximos y mínimos del tamaño de fuente, así como el tamaño preferido (o «fluid size») y un tamaño relativo. Lo destacado de esta herramienta es su capacidad para visualizar los cambios mediante gráficos y tablas para comprender rápidamente cómo se ajustará el tamaño de la fuente en función del ancho de la pantalla. Además, proporciona código CSS listo para copiar y pegar, facilitando la implementación de los ajustes realizados.

- **Min-Max-Value Interpolation by James Gilyead y Trys Mudford:** Esta herramienta nos permite introducir valores máximos y mínimos del tamaño de fuente, así como valores máximos y mínimos del viewport. Al ajustar estos parámetros, podemos ver cómo se comportará la tipografía en función del tamaño de la pantalla. La herramienta proporciona ejemplos visuales en forma de texto y nos permite jugar con el tamaño del navegador para experimentar con diferentes escenarios de diseño responsivo.

Como has podido ver, la tipografía fluida es una parte fundamental del diseño web moderno. Al utilizar unidades relativas de medida como vw y vh, junto con técnicas avanzadas como calc() y clamp(), podemos crear experiencias de lectura agradables y accesibles en cualquier dispositivo sin necesidad de usar media queries.

3.9. CSS line-clamp

La propiedad **CSS line-clamp** nos permite **acortar un texto a un número específico de líneas**, lo que resulta especialmente útil en interfaces de usuario donde el espacio es limitado. Además de truncar el texto, line-clamp se combina con otras propiedades CSS, como display y **overflow**, para ajustar el comportamiento del texto, incluyendo la

adición de puntos suspensivos, también llamados puntos de elipsis, al final del texto truncado.

```
.elemento-truncado {
display: -webkit-box;
-webkit-line-clamp: <numero_de_lineas>;
-webkit-box-orient: horizontal | vertical;
overflow: hidden;
}
.elemento-truncado { display: -webkit-box; -webkit-line-clamp: <numero_de_lineas>; -webkit-box-orient: horizontal | vertical; overflow: hidden; }
```

```
.elemento-truncado {

   display: -webkit-box;

   -webkit-line-clamp: <numero_de_lineas>;

   -webkit-box-orient: horizontal | vertical;

   overflow: hidden;

}
```

- `display: -webkit-box;`: Se utiliza para establecer el tipo de caja de diseño utilizado para el elemento. En este caso, `-webkit-box` es necesario para compatibilidad con navegadores basados en WebKit, como Safari y Chrome.
- `-webkit-line-clamp: <numero_de_lineas>;`: Especifica el número de líneas que se mostrarán antes de truncar el texto. Por ejemplo, si estableces `-webkit-line-clamp: 3;`, solo se mostrarán las primeras tres líneas del texto y el resto será truncado.
- `-webkit-box-orient: vertical;`: Establece la orientación del contenido dentro del contenedor. En el contexto de `line-clamp`, normalmente se usa `vertical` para que los elementos de línea se apilen verticalmente dentro del contenedor flexible, permitiendo así la truncación vertical del texto.
- `overflow: hidden;`: Ocultar el contenido que se desborda del contenedor. Es importante para asegurarse de que el texto truncado no sea visible fuera del área designada.

3.10. CSS Scroll Snap

El **Scroll Snap**, desplazamiento con ajuste o con bloqueo en español, es una **propiedad de CSS** que nos permite **controlar el desplazamiento o scroll en contenedores** para que sea suave. Esta característica es particularmente útil en sitios web que necesitan una **navegación fluida y organizada entre secciones de contenido**.

```
.contenedor-scroll {
scroll-snap-type: [x | y | block | inline] [mandatory | proximity];
}
.seccion {
scroll-snap-align: [start | center | end | none];
}
```

```
.contenedor-scroll { scroll-snap-type: [x | y | block | inline] [mandatory | proximity]; } .seccion { scroll-snap-align: [start | center | end | none]; }
```

```
.contenedor-scroll {

  scroll-snap-type: [x | y | block | inline] [mandatory | proximity];

}

.seccion {

  scroll-snap-align: [start | center | end | none];

}
```

- `scroll-snap-type` especifica cómo se debe comportar el desplazamiento dentro del contenedor. Puede definir el tipo de desplazamiento en los ejes horizontal (`x`) o vertical (`y`), así como también en bloques (`block`) o en líneas (`inline`). Además, determina si el desplazamiento debe detenerse obligatoriamente en los puntos de anclaje especificados (`mandatory`) o si puede detenerse en el punto más cercano (`proximity`).
- `scroll-snap-align` se aplica a los elementos hijos del contenedor. Esta propiedad determina cómo se alineará cada elemento con respecto al punto de anclaje. Los valores comunes para `scroll-snap-align` son `start`, `end`, `center`, `none`, etc. Por ejemplo, `scroll-snap-align: start;` alinea el inicio del elemento con el punto de anclaje.

Esta propiedad nos permite **definir puntos de «anclaje»** a los que el desplazamiento se detendrá de forma automática al hacer scroll. Esto es muy interesante en sitios web que presentan una disposición en bloques horizontales o verticales, facilitando que los usuarios naveguen de manera fluida y directa entre sus secciones.

¿Para qué sirve el Scroll Snap en CSS?

El desplazamiento o scroll es la acción de mover el contenido de una página web hacia arriba o hacia abajo para ver áreas no visibles en la pantalla. En un desplazamiento convencional, el usuario puede deslizar el contenido de forma continua, y este se mueve en función de la dirección y la intensidad del deslizamiento.

Sin embargo, con el Scroll Snap de CSS, podemos modificar este comportamiento para que el desplazamiento se alinee con ciertos puntos de anclaje dentro del contenedor.

Por ejemplo, al desplazar hacia abajo en una página web, en lugar de permitir un movimiento fluido, el desplazamiento puede detenerse automáticamente en ciertos puntos predefinidos, como el inicio de cada sección o bloque de contenido.

Implementación de Scroll Snap en CSS

Para implementar Scroll Snap en CSS, aplicaremos la propiedad scroll-snap-type al contenedor correspondiente con la siguiente sintaxis.

```
.contenedor-scroll {
    scroll-snap-type: [x | y | block | inline] [mandatory | proximity];
}
.seccion {
    scroll-snap-align: [start | center | end | none];
}
```

- La propiedad scroll-snap-type especifica cómo se debe comportar el desplazamiento dentro del contenedor. Puede definir el tipo de desplazamiento en los ejes horizontal (x) o vertical (y), así como también en bloque (block) o en línea (inline). Además, determina si el desplazamiento debe detenerse obligatoriamente en los puntos de anclaje especificados (mandatory) o si puede detenerse en el punto más cercano (proximity).

- Un aspecto importante a tener en cuenta es la propiedad scroll-snap-align, que se aplica a los elementos hijos del contenedor. Esta propiedad determina cómo se alineará cada elemento con respecto al punto de anclaje. Los valores comunes para scroll-snap-align son start, end, center, none, etc. Por ejemplo, scroll-snap-align: start; alinea el inicio del elemento con el punto de anclaje.

Ejemplo:

Supongamos que tenemos un contenedor que contiene varias secciones y queremos aplicar Scroll Snap para que el desplazamiento se detenga en cada sección. Para ello, creamos un contenedor con la clase «contenedor-scroll» que contiene varias secciones con la clase «seccion». Aplicamos Scroll Snap al contenedor verticalmente (scroll-snap-type: y mandatory), lo que hace que el desplazamiento se detenga obligatoriamente en cada sección. Además, establecemos scroll-snap-align: start en cada sección para alinear el inicio de cada una con el punto de anclaje.

HTML:

```html
<div class="contenedor-scroll">
    <div class="seccion">Sección 1</div>
    <div class="seccion">Sección 2</div>
    <div class="seccion">Sección 3</div>
    <div class="seccion">Sección 4</div>
</div>
```

CSS:

```css
.contenedor-scroll {
    height: 300px; /* Altura del contenedor */
    overflow-y: scroll; /* Habilitar el desplazamiento vertical */
    scroll-snap-type: y mandatory; /* Desplazamiento vertical obligatorio */
}
.seccion {
    height: 300px; /* Altura de cada sección */
    scroll-snap-align: start; /* Alinear el inicio de cada sección con el punto de anclaje */
}
```

Secciones de contenido que ocupan el 100% del alto

Con Scroll Snap, podemos crear secciones de contenido que ocupen todo el alto de la pantalla y se desplacen de manera suave. Esto es útil para presentaciones de una sola página donde cada sección se alineará perfectamente y los usuarios podrán navegar de manera intuitiva por el contenido.

CSS:

```css
body, html {
  height: 100%;
}
.scroll-container {
  width: 100%;
  height: 100%;
  overflow-y: auto;
  scroll-snap-type: y mandatory; /* Activamos el desplazamiento por defecto */
}
.scroll-item {
  height: 100vh; /* Ocupar toda la altura de la ventana */
  scroll-snap-align: start; /* Alineamos los elementos al inicio de la ventana de visualización */
  display: flex;
  justify-content: center;
  align-items: center;
}
```

HTML:

```html
<div class="scroll-container">
  <div class="scroll-item item1">¡Hola!</div>
  <div class="scroll-item item2">¿Qué tal?</div>
  <div class="scroll-item item3">¿Prácticando con Scroll Snap?</div>
  <div class="scroll-item item4">Me gusta el CSS</div>
  <div class="scroll-item item5">❤</div>
</div>
```

Galerías de productos en contenedores horizontales

En tiendas online, las galerías de productos pueden contar con Scroll Snap para permitir a los usuarios explorar los diferentes productos de forma horizontal. Cada producto se muestra en su propia tarjeta o *card* y el Scroll Snap asegura que cada tarjeta esté alineada perfectamente cuando se hace scroll horizontal, facilitando la comparación entre productos.

CSS:

```
.product-gallery {
   display: flex;
   overflow-x: scroll;
   scroll-snap-type: x mandatory;
}
.product-card {
   flex: 0 0 auto;
   width: 200px;
   margin-right: 10px;
   scroll-snap-align: start;
   text-align: center;
}
.product-card img {
   width: 100px;
   height: auto;
}
.product-card h3, .product-card p {
   margin: 10px 0;
}
.product-card button {
   background-color: #007bff;
   color: #fff;
   border: none;
   padding: 10px 20px;
   cursor: pointer;
   transition: background-color 0.3s;
}
.product-card button:hover {
   background-color: #0056b3;
}
```

HTML:

```
<div class="product-gallery">
   <div class="product-card">
      <img src="https://cdn-icons-png.flaticon.com/128/12411/12411992.png" alt="Producto 1">
      <h3>Producto 1</h3>
      <p>Descripción del Producto 1</p>
      <button>Añadir al carrito</button>
   </div>
   <div class="product-card">
      <img src="https://cdn-icons-png.flaticon.com/128/12412/12412030.png" alt="Producto 2">
      <h3>Producto 2</h3>
      <p>Descripción del Producto 2</p>
      <button>Añadir al carrito</button>
   </div>
```

```
  <div class="product-card">
    <img src="https://cdn-icons-png.flaticon.com/128/12411/12411976.png" alt="Producto 3">
    <h3>Producto 3</h3>
    <p>Descripción del Producto 3</p>
    <button>Añadir al carrito</button>
  </div>
 <div class="product-card">
    <img src="https://cdn-icons-png.flaticon.com/128/12412/12412024.png" alt="Producto 3">
    <h3>Producto 4</h3>
    <p>Descripción del Producto 3</p>
    <button>Añadir al carrito</button>
  </div>
  <div class="product-card">
    <img src="https://cdn-icons-png.flaticon.com/128/12411/12411992.png" alt="Producto 1">
    <h3>Producto 5</h3>
    <p>Descripción del Producto 1</p>
    <button>Añadir al carrito</button>
  </div>
  <div class="product-card">
    <img src="https://cdn-icons-png.flaticon.com/128/12412/12412030.png" alt="Producto 2">
    <h3>Producto 6</h3>
    <p>Descripción del Producto 2</p>
    <button>Añadir al carrito</button>
  </div>
  <div class="product-card">
    <img src="https://cdn-icons-png.flaticon.com/128/12411/12411976.png" alt="Producto 3">
    <h3>Producto 7</h3>
    <p>Descripción del Producto 3</p>
    <button>Añadir al carrito</button>
  </div>
 <div class="product-card">
    <img src="https://cdn-icons-png.flaticon.com/128/12412/12412024.png" alt="Producto 3">
    <h3>Producto 8</h3>
    <p>Descripción del Producto 3</p>
    <button>Añadir al carrito</button>
  </div>
</div>
```

Galería vertical con texto descriptivo

En esta ocasión, vamos a crear una galería de imágenes con texto dispuesto de forma vertical. Esto puede ser útil cuando queremos mostrar imágenes con descripciones o detalles adicionales debajo de cada imagen. Cada imagen está alineada al inicio del contenedor gracias a la propiedad scroll-snap-align. Al hacer scroll, nos detenemos en cada elemento. Esto puede ser útil para presentaciones de productos, portafolios o cualquier otro tipo de contenido visual que queramos destacar.

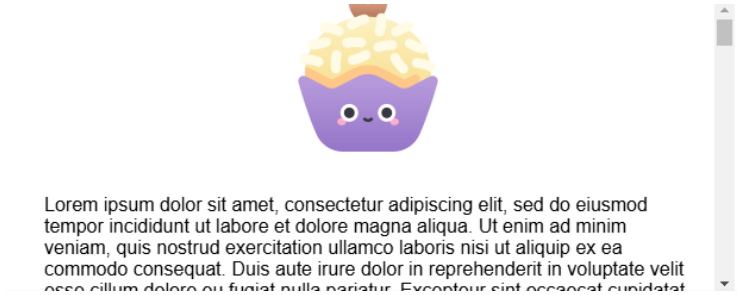
CSS:

```
body, html {
  margin: 0;
  padding: 0;
  font-family: Arial, sans-serif;
}
.gallery {
  width: 100%;
  max-width: 600px;
  margin: 0 auto;
  height: 100vh;
  overflow-y: scroll;
  scroll-snap-type: y mandatory;
  display: flex;
  flex-direction: column;
}
.image {
  display: flex;
  justify-content: center;
  align-items: center;
  font-size: 2em;
  scroll-snap-align: start; /* Hace que cada imagen se alinee al inicio del contenedor */
}
p {
  padding: 20px;
}
```

HTML

```
<div class="gallery">
  <div class="image"><img src="https://cdn-icons-png.flaticon.com/128/12411/12411992.png" alt="Imagen" /></div>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. 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.</p>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. 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.</p>

<div class="image"><img src="https://cdn-icons-png.flaticon.com/128/12412/12412030.png" alt="Imagen" /></div>

  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. 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.</p>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. 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.</p>

  <div class="image"><img src="https://cdn-icons-png.flaticon.com/128/12411/12411992.png" alt="Imagen" /></div>

  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. 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.</p>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. 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.</p>

 <div class="image"><img src="https://cdn-icons-png.flaticon.com/128/12412/12412024.png" alt="Imagen" /></div>

  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. 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.</p>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. 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.</p>

 <div class="image"><img src="https://cdn-icons-png.flaticon.com/128/12411/12411976.png" alt="Imagen" /></div>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. 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.</p>

 <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. 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.</p>
</div>
```



3.11. CSS scroll-behavior

Recientemente estudiamos cómo **crear anclas o marcadores en HTML** para **navegar por una página web** de manera más eficiente. Sin embargo, una vez que clicamos en estos enlaces, **la transición por defecto es seca** y sin ningún tipo de animación, lo que puede resultar en una experiencia de usuario poco atractiva. Afortunadamente, con la **propiedad CSS scroll-behavior,** podemos transformar estas transiciones y hacer que el **desplazamiento entre marcadores sea suave y animado**, sin la necesidad de recurrir a **JavaScript**, **Jquery** o librerías externas.

En esencia, la propiedad **scroll-behavior de CSS** nos permite agregar un efecto de scroll suave (*smooth scroll*) a nuestra página web para **mejorar la experiencia del usuario**, haciendo que la navegación sea más atractiva y fluida. De esta forma, al pulsar enlaces

que llevan a un ancla (*anchor*) o marcador de una zona del documento HTML, en lugar de moverse directa e instantáneamente, que es el comportamiento por defecto del navegador, se moverá con una pequeña animación suave.

Propiedad CSS scroll-behavior: sintaxis y valores

La propiedad scroll-behavior **se aplica típicamente al elemento html** para controlar el comportamiento de desplazamiento en toda la página. Veamos su sintaxis.

Sintaxis:

```
html{
   scroll-behavior: auto | smooth;
}
```

- **auto**: Este es el valor predeterminado. Indica que el desplazamiento será instantáneo, similar al comportamiento estándar del navegador.
- **smooth**: Este valor activa el desplazamiento suave, lo que significa que el movimiento entre los diferentes elementos de la página será gradual y animado.

Compatibilidad de scroll-behavior con navegadores

La propiedad scroll-behavior **es compatible con la mayoría de los navegadores** modernos, pero no es compatible con versiones anteriores de Internet Explorer. Puedes ver la compatibilidad actual según Can I Use.

Diferencias entre scroll-behavior y scroll-snap

La propiedad scroll-behavior y la funcionalidad de scroll-snap son **herramientas de CSS para mejorar la experiencia de desplazamiento en las páginas web**. Aunque ambas están diseñadas para mejorar la navegación, tienen propósitos y funcionalidades diferentes. Por un lado, scroll-behavior se centra en el comportamiento general del desplazamiento entre elementos en la ventana del navegador, mientras que, por otro lado, scroll-snap se utiliza para controlar el desplazamiento dentro de contenedores específicos, permitiendo un desplazamiento centrado en puntos de anclaje definidos.
Ambas **características son complementarias** y pueden mejorar la experiencia de navegación en una página web al proporcionar un desplazamiento más suave, fluido y controlado. Visita el artículo **Scroll snap en CSS: Controla el desplazamiento de tu web** para más información.

Ejemplos prácticos de scroll-behavior

Veamos algunos ejemplos prácticos que puedes necesitar en la creación de tus páginas web para conseguir efectos que mejoren la **usabilidad**.

Menú de navegación con enlaces a secciones con scroll suave

La propiedad scroll-behavior es muy utilizada en páginas de aterrizaje (landing pages) compuestas por un menú principal que enlaza a diversos marcadores o anclas dispuestos en distintos contenedores.

HTML:

```
<nav>
  <ul>
    <li><a href="#section1">Sección 1</a></li>
    <li><a href="#section2">Sección 2</a></li>
    <li><a href="#section3">Sección 3</a></li>
    <li><a href="#section4">Sección 4</a></li>
  </ul>
</nav>
<section id="section1">Sección 1</section>
<section id="section2">Sección 2</section>
<section id="section3">Sección 3</section>
<section id="section4">Sección 4</section>
```

CSS :

```
html{
  scroll-behavior: smooth;
}
body {
  font-family: Arial, sans-serif;
  margin: 0;
  padding: 0;
}
nav {
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  background-color: #333;
  padding: 10px 0;
  z-index: 999;
}
nav ul {
  list-style: none;
  margin: 0;
```

```
  padding: 0;
  text-align: center;
}
nav ul li {
  display: inline-block;
}
nav ul li a {
  color: #fff;
  text-decoration: none;
  font-size: 16px;
  padding: 10px;
  transition: background-color 0.3s;
}
nav ul li a:hover {
  background-color: #555;
}
section {
  height: 300px;
  display: flex;
  justify-content: center;
  align-items: center;
  font-size: 3rem;
  color: black;

}
section:nth-child(odd) {
  background-color: #007bff;
  color: white;
}
```

Botón para volver a la parte superior de la página con scroll suave

Este ejemplo muestra un botón al final de la página que, al hacer clic en él, desplaza la página suavemente hasta la parte superior. Es una forma de proporcionar a los usuarios una manera rápida de volver al inicio de la página. De la misma forma, se podría hacer el típico botón fijo lateral con una flecha hacia arriba que nos permita volver hacia arriba en cualquier momento.

HTML

```
<div class="container" id="scrollBtn">
  <p>Haz scroll hasta el final de la página y presiona el botón "Ir arriba"</p>
</div>
```

```
 <a href="#scrollBtn">↑ Ir arriba</a>
```

CSS:

```
html{
  scroll-behavior: smooth;
}
body {
  font-family: Arial, sans-serif;
}
.container {
  height: 1000px;
}
a {
  display:flex;
  padding: 10px 20px;
  font-size: 1.2rem;
  background-color: #007bff;
  color: #fff;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  text-decoration: none;
  transition: background-color 0.3s ease;
}
a:hover {
  background-color: #0056b3;
}
```

Aside de contenidos con scroll suave

Este ejemplo muestra un diseño con un aside de contenidos a la izquierda y el contenido principal a la derecha. Los enlaces en el aside te llevan a las diferentes secciones del contenido principal, y el scroll es suave gracias a la propiedad scroll-behavior de CSS.

HTML:

```
<div class="container">
   <aside>
      <ul>
        <li><a href="#section1">Sección 1</a></li>
        <li><a href="#section2">Sección 2</a></li>
        <li><a href="#section3">Sección 3</a></li>
        <li><a href="#section4">Sección 4</a></li>
```

```
      </ul>
    </aside>
    <main>
      <section id="section1">
        <h2>Sección 1</h2>
        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed tempor, odio sit amet suscipit fermentum, sem libero placerat nisl, sed pellentesque risus purus ut velit.</p>
        <p>Integer nec elit sit amet metus pulvinar ultrices. Nullam eget nisi ut risus fermentum convallis. Phasellus nec mauris aliquam, fermentum eros vel, sodales lectus.</p>
      </section>
      <section id="section2">
        <h2>Sección 2</h2>
        <p>Integer nec elit sit amet metus pulvinar ultrices. Nullam eget nisi ut risus fermentum convallis. Phasellus nec mauris aliquam, fermentum eros vel, sodales lectus.</p>
        <p>Vivamus auctor eros et massa pretium ultrices. Integer auctor ultricies risus, non eleifend purus. Duis sed neque elit. Duis nec diam augue. Mauris elementum nulla vel magna elementum, at suscipit sapien fermentum.</p>
      </section>
      <section id="section3">
        <h2>Sección 3</h2>
        <p>Nulla facilisi. Pellentesque tempor erat vel tortor luctus hendrerit. In hac habitasse platea dictumst. Cras ac mauris ac odio pharetra vehicula eget eget odio.</p>
        <p>Vivamus auctor eros et massa pretium ultrices. Integer auctor ultricies risus, non eleifend purus. Duis sed neque elit. Duis nec diam augue. Mauris elementum nulla vel magna elementum, at suscipit sapien fermentum.</p>
      </section>
      <section id="section4">
        <h2>Sección 4</h2>
        <p>Vivamus auctor eros et massa pretium ultrices. Integer auctor ultricies risus, non eleifend purus. Duis sed neque elit. Duis nec diam augue. Mauris elementum nulla vel magna elementum, at suscipit sapien fermentum.</p>
        <p>Integer nec elit sit amet metus pulvinar ultrices. Nullam eget nisi ut risus fermentum convallis. Phasellus nec mauris aliquam, fermentum eros vel, sodales lectus.</p>
      </section>
    </main>
  </div>
```

CSS :

```
html{
  scroll-behavior: smooth;
}
body {
  font-family: Arial, sans-serif;
  margin: 0;
  padding: 0;
}
```

```
.container {
  display: flex;
  max-width: 1200px;
  margin: 0 auto;
  padding: 20px;
}
aside {
  flex: 0 0 200px;
  background-color: #f0f0f0;
  padding: 20px;
}
main {
  flex: 1;
  padding: 20px;
}
ul {
  list-style-type: none;
  padding: 0;
}
li {
  margin-bottom: 10px;
}
a {
  text-decoration: none;
  color: #007bff;
}
a:hover {
  text-decoration: underline;
}
section {
  margin-bottom: 40px;
}
section:last-child {
  margin-bottom: 0;
}
```

```
html{
scroll-behavior: auto | smooth;
}
html{ scroll-behavior: auto | smooth; }
```

```
html{
  scroll-behavior: auto | smooth;
}
```

- **auto**: Valor predeterminado. Indica que el desplazamiento será instantáneo, sin transición o animación.
- **smooth**: Activa el desplazamiento suave, lo que significa que el movimiento entre los diferentes elementos de la página será gradual y animado.

3.12. CSS mix-blend-mode

Los **Blend Modes**, o **modos de mezcla** en español, son una característica de **CSS** que permite combinar elementos HTML con otros elementos de fondo como imágenes o degradados. Estos modos nos permiten **mezclar los colores del fondo, que puede ser una imagen o un color, con el elemento que se superpone**.

```
.elemento {
mix-blend-mode: <modo-de-mezcla>;
}
.elemento { mix-blend-mode: <modo-de-mezcla>; }
```

```
.elemento {

   mix-blend-mode: <modo-de-mezcla>;

}
```

Blend Mode	Descripción	Características
multiply	Multiplica los colores de la capa superior con los de la capa inferior.	– Oscurece la imagen resultante.
screen	Invierte los colores de la capa superior y los multiplica con los de la capa inferior.	– Aclara las áreas de la imagen.
overlay	Combina Multiply y Screen, oscureciendo o aclarando la capa inferior dependiendo de sus colores.	– Produce un efecto de contraste y saturación.

Blend Mode	Descripción	Características
darken	Conserva los colores más oscuros de las dos capas.	– Selecciona el color más oscuro de cada píxel de ambas capas.
lighten	Conserva los colores más claros de las dos capas.	– Selecciona el color más claro de cada píxel de ambas capas.
color-dodge	Aclara los colores de la capa inferior dependiendo de los colores de la capa superior.	– Aclara las áreas de la imagen de la capa inferior.
color-burn	Oscurece los colores de la capa inferior dependiendo de los colores de la capa superior.	– Oscurece las áreas de la imagen de la capa inferior.
difference	Resta los colores de la capa superior de los de la capa inferior.	– Produce un efecto de inversión de colores.
exclusion	Similar a Difference, pero con una apariencia menos extrema.	– Combina los colores de ambas capas con un efecto de exclusión.
hard-light	Combina Multiply y Screen, pero de una manera más intensa.	– Produce un efecto más fuerte de contraste y saturación.
soft-light	Similar a Overlay, pero con una apariencia más suave.	– Produce un efecto más suave de contraste y saturación.
hue	Preserva el brillo y la saturación de la capa inferior, pero adopta el tono de la capa superior.	– Aplica el tono de la capa superior a la capa inferior.

Blend Mode	Descripción	Características
saturation	Preserva el brillo y el tono de la capa inferior, pero adopta la saturación de la capa superior.	– Aplica la saturación de la capa superior a la capa inferior.
color	Preserva el brillo de la capa inferior, pero adopta el tono y la saturación de la capa superior.	– Aplica el tono y la saturación de la capa superior a la capa inferior.
luminosity	Preserva el tono y la saturación de la capa inferior, pero adopta el brillo de la capa superior.	– Aplica el brillo de la capa superior a la capa inferior.

Tabla 28: Valores de mix-blend-mode

¿Qué son los Blend Modes en CSS?

Los **Blend Modes en CSS** son una propiedad, de reciente incorporación, llamada mix-blend-mode, que permite **diferentes valores**, como por ejemplo para **oscurecer, aclarar o contrastar**, entre otros modos. Estos valores permiten mezclar el color de fondo de un elemento con el color del elemento superpuesto, que puede ser un texto o un contenedor.

Con las Blend Modes, **podemos jugar con la transparencia y la opacidad de los elementos**, logrando efectos de fusión, aclarado, superposición, saturación o contraste, entre otros.

3.13. CSS Hyphens

¿Te ha pasado que al diseñar una página web te encuentras con el **problema de que las palabras no caben correctamente en las líneas de texto** y necesitas usar el guion para dividirlas de forma automática? Este inconveniente puede hacer que tus diseños se vean desordenados y poco profesionales. Pero ¡no te preocupes! Existe una **solución fácil de implementar con CSS: la propiedad hyphens CSS**.

```
.hyphenation {
hyphens: auto | manual | none;
}
.hyphenation { hyphens: auto | manual | none; }
```

```
.hyphenation {

   hyphens: auto | manual | none;

}
```

- auto: El navegador decide dónde dividir las palabras para evitar el desbordamiento del texto. Utiliza las reglas de separación de sílabas del idioma predeterminado del documento.
- manual: Las palabras no se dividirán automáticamente. Debes utilizar el carácter de guion (-) para indicar dónde se pueden dividir las palabras manualmente.
- none: Desactiva la división automática y manual de palabras. El texto puede desbordarse si no cabe en el contenedor.

3.14. Filtros CSS

Los **filtros CSS** permiten aplicar **efectos visuales a cualquier elemento HTML y a las imágenes directamente en CSS** sin utilizar previamente un programa de diseño gráfico.

Para aplicar un filtro CSS a una imagen o contenedor se utiliza la propiedad ***filter*** y se pueden aplicar efectos como desenfoque, contraste, saturación, ajuste del brillo o conversión a escala de grises, entre otros.

Sintaxis:

```
selector {
filter: <función-de-filtro>(<valor>) ;
}
selector { filter: <función-de-filtro>(<valor>) ; }
```

```
selector {
  filter: <función-de-filtro>(<valor>) ;
}
```

Diferencias entre blend modes y filtros: los blend modes se utilizan para mezclar colores entre elementos superpuestos, mientras que los filtros se utilizan para aplicar efectos visuales directamente al contenido de un elemento.

Filtros CSS: Sintaxis de la propiedad filter de CSS

Como hemos dicho, la propiedad filter de CSS permite aplicar efectos visuales a elementos de la página. Estos efectos se aplican mediante funciones de filtro, que pueden tener uno o más valores según el efecto deseado. Veamos la sintaxis de la propiedad filter en CSS y los valores que acepta.

Valores más comunes que acepta la propiedad `filter`:

Función	Descripción	Valores
`blur(<valor>)`	Aplica un desenfoque al elemento	`<valor>` cantidad de desenfoque a aplicar
`brightness(<valor>)`	Ajusta el brillo del elemento	`<valor>` 0: completamente oscuro, 1: original
`contrast(<valor>)`	Ajusta el contraste del elemento	`<valor>` 0: completamente gris, 1: original
`grayscale(<valor>)`	Convierte el elemento a escala de grises	`<valor>` 0: original, 1: escala de grises
`hue-rotate(<valor>)`	Gira el espectro de colores del elemento	`<valor>` es la cantidad de grados en sentido horario 0 a 360deg
`invert(<valor>)`	Invierte los colores del elemento	`<valor>` 0: original, 1: invertido
`opacity(<valor>)`	Ajusta la opacidad del elemento	`<valor>` 0: transparente, 1: original
`saturate(<valor>)`	Ajusta la saturación del elemento	`<valor>` 0: grisáceo , 100%: original
`sepia(<valor>)`	Aplica un tono sepia al elemento	`<valor>` 0: original, 1: sepia completo

Tabla 29. Valores destacados de la propiedad filter de CSS

3.15. Prefijos para los navegadores

Prefijo	Navegador
-moz-	Firefox
-webkit-	Safari y Chrome

-o-	Opera
-khtml-	Konqueror
-ms-	Internet Explorer
-chrome-	Google Chrome

Tabla 29. Prefijos para los navegadores

El uso de prefijos CSS específicos para navegadores ha disminuido considerablemente en los últimos años. Anteriormente, los prefijos **como -webkit-, -moz-, -ms-, y -o-** eran necesarios para habilitar funciones experimentales o nuevas de CSS en ciertos navegadores.

Sin embargo, con la evolución de los estándares web y la mejor compatibilidad entre navegadores, hoy en día la mayoría de las propiedades CSS no requieren estos prefijos.

Aun así, en casos específicos donde una característica aún no está completamente estandarizada o no es compatible uniformemente entre navegadores, algunos prefijos podrían ser necesarios. Sin embargo, es recomendable consultarlos solo cuando sea estrictamente necesario, y usar herramientas como Can I use para verificar la compatibilidad de las propiedades CSS antes de añadir prefijos manualmente.

Además, existen herramientas como autoprefixer, que se encargan de añadir los prefijos automáticamente según las necesidades de compatibilidad de cada proyecto, lo que facilita mucho este proceso.

Centrar horizontal y verticalmente en CSS un elemento

Existen diversas formas de centrar horizontal y verticalmente un elemento en un contenedor mediante CSS. Cada método es adecuado para una situación concreta y, por ello, es necesario saber cómo funcionan todas las propiedades para reconocer fácilmente qué técnica utilizar en cada caso.

A continuación, se describe cómo se puede realizar el centrado de algunos elementos mediante cajas flexibles, con la propiedad «**transform**«, con las propiedades «**line-height**» y «**text-align**» y con la propiedad **vertical-align**.

Centrar vertical y horizontalmente elementos con flexbox

Este es el método más útil porque funciona para todos los elementos que se posicionen dentro de un contenedor. **Por tanto, es válido para imágenes, textos, vídeos, etc.** Sin embargo, para comprender cómo funciona es necesario tener conocimientos sobre el modelo de **caja flexible o flexbox**.

Para utilizar las propiedades de flexbox es necesario crear un contenedor y declararle la propiedad «**display**: **flex**«. Para centrar horizontal y verticalmente el contenido utilizaremos las propiedades «**justify-content**: center» y «**align-items:center**«.

```
div{
    display:flex;
    justify-content: center;
    align-items: center;
}
```

Ejemplo:

HTML:

```
<div>
 <!-- Cualquier elemento incluido dentro del div se centrará vertical y horizontalmente con flexbox -->
 <h1>Centrado con flexbox</h1>
</div><br>
<div>
 <img src="https://www.eniun.com/wp-content/uploads/ENIUN@2x.png">
</div>
```

CSS:

```
div{
 height: 200px;
 background-color: #EEE;
 display:flex;
 justify-content: center;
 align-items: center;
}
```

Centrar vertical y horizontalmente un texto con line-height y text-align

Mediante las propiedades **line-height** y **text-align** podemos **alinear únicamente textos**. Así que este método sólo va a ser **válido para el centrado de párrafos, encabezados**, etc.

```
div {
    text-align: center;
    height: 200px;
}
div h1 {
    line-height: 200px;
}
```

Ejemplo:

HTML:

```
<div>
 <h1>Centrar texto</h1>
</div>
```

CSS:

```
div {
 height: 200px;
 background-color: #EEE;
 text-align: center;
}
div h1 {
   line-height: 200px;
}
```

Centrar vertical y horizontalmente una imagen o contenedor con propiedad transform y posición absoluta

Mediante este método se **puede alinear tanto vertical como horizontalmente imágenes y contenedores**. **No es válido para la alineación de textos ya que no tiene en cuenta el tamaño de la letra. El centrado de elementos se realiza mediante posición absoluta y la propiedad transform**.

```
div {
    position: relative;
}
div img{
    position: absolute;
    left: 50%;
    top: 50%;
    transform: translate(-50%, -50%);
}
```

Ejemplo:

HTML:

```
<div>
 <img src="https://www.eniun.com/wp-content/uploads/ENIUN@2x.png">
</div>
```

CSS:

```
div {
  height: 300px;
  background-color: #EEE;
  position: relative;
}
div img{
    position: absolute;
    left: 50%;
    top: 50%;
    transform: translate(-50%, -50%);
    -webkit-transform: translate(-50%, -50%);
}
```

Centrar un elemento vertical y horizontalmente utilizando posición absoluta y margen negativo

Centrado absoluto con posición absoluta y margen negativo:

```
.container {
  position: relative;
}

.centered-element {
  position: absolute;
  top: 50%;
  left: 50%;
  margin-top: -50px; /* Ajusta el valor según la mitad de la altura del elemento */
  margin-left: -50px; /* Ajusta el valor según la mitad del ancho del elemento */
}
```

Ejemplo:

HTML:

```
<div class="container">
  <div class="centered-element">Contenido centrado</div>
</div>
```

CSS:

```
.container {
  position: relative;
  width: 400px;
  height: 300px;
  background-color: lightgray;
```

```
}

.centered-element {
  position: absolute;
  top: 50%;
  left: 50%;
  margin-top: -32px; /* Ajusta el valor según la mitad de la altura del elemento */
  margin-left: -95px; /* Ajusta el valor según la mitad del ancho del elemento */
  background-color: white;
  padding: 20px;
}
```

Centrar un elemento con grid

Para centrar un elemento con grid se debe utilizar display: grid y place-items: center en el contenedor. De esta forma, el elemento se centrará tanto vertical como horizontalmente dentro del contenedor.

```
.container {
  display: grid;
  place-items: center;
}
```

EJEMPLO:

HTML:

```
<div class="container">
  <div class="centered-element">Elemento centrado</div>
</div>
```

CSS:

```
.container {
  display: grid;
  place-items: center;
  height: 300px;
  border: 1px solid #ccc;
}

.centered-element {
  background-color: #f1f1f1;
  padding: 20px;
}
```

Alinear un texto verticalmente con una imagen con vertical-align

Con esta técnica podemos **alinear un texto a una imagen de forma vertical**. Esta propiedad admite los valores ***top***, ***middle*** y ***bottom***, entre otros valores.

```
img{
    vertical-align: middle;
}
```

EJEMPLO:

HTML:

```
<img src="https://cdn.pixabay.com/photo/2019/11/02/21/45/maple-leaf-4597501_960_720.jpg" width="500px"> Alinear texto
```

CSS:

```
img{
  vertical-align: middle;
}
```

Alinear verticalmente un texto en un div con vertical-align y el modelo de tabla (*table mode*)

Vamos a alinear un texto en un div con la propiedad vertical-align. Hay que tener en cuenta que **esta propiedad no se comporta de la misma manera con todos los elementos.** En este caso, haremos uso de las reglas de estilo «**display: inline-table**» y «**display: table: cell**».

```
div{
   height: 200px; width: 200px;
   background-color: #EEE;
   display: inline-table;
 }
 p{
   display: table-cell;
   vertical-align: middle;
 }
```

EJEMPLO:

HTML:

```
<div class="a">
  <p>Texto dentro de un div</p>
</div>
```

CSS:

```
div{
  height: 200px; width: 200px;
  background-color: #EEE;
  display: inline-table;
  width: 100%;
}
p{
  text-align: center;
  display: table-cell;
  vertical-align: middle;
}
```

Alinear horizontalmente una imagen con text-align

Las imágenes se pueden alinear a la izquierda, derecha o centro insertándolas dentro de un div e indicando la propiedad text-align correspondiente.

```
.a{text-align: left;}
.b{text-align: center;}
.c{text-align: right;}
```

EJEMPLO:

HTML:

```
<div class="a"><img width="60px"
src="https://images.vexels.com/media/users/3/143166/isolated/lists/f76b5ed0f0de39
d7e4c3bf59b8763c35-logo-de-la-flor-del-hipster.png" /></div>
<div class="b"><img width="60px"
src="https://images.vexels.com/media/users/3/143166/isolated/lists/f76b5ed0f0de39
d7e4c3bf59b8763c35-logo-de-la-flor-del-hipster.png" /></div>
<div class="c"><img width="60px"
src="https://images.vexels.com/media/users/3/143166/isolated/lists/f76b5ed0f0de39
d7e4c3bf59b8763c35-logo-de-la-flor-del-hipster.png" /></div>
```

CSS :

```
.a{text-align: left;}
.b{text-align: center;}
.c{text-align: right;}
```

Alinear verticalmente un checkbox con un label

Cuando el tamaño del texto no coincide con el tamaño del checkbox, estos dos elementos no quedan alineados verticalmente. **Para conseguir esta alineación utilizaremos** «**vertical-align:middle**» y definiremos el tamaño del texto a las dos etiquetas, tal y como se muestra a continuación.

```
label, input{
    font-size: 28px;
    vertical-align: middle;
}
```

EJEMPLO:

HTML:

```
<label for="texto">TEXTO</label><input type="checkbox" id="texto"/>
```

CSS:

```
label, input{
  font-size: 28px;
  vertical-align: middle;
}
```

Alinear un elemento según el ancho y alto de la pantalla (viewport)

Para centrar elementos vertical y horizontalmente según el tamaño de la pantalla (viewport) en CSS, puedes utilizar las unidades de medida relativas vh y vw junto con las propiedades vistas anteriormente, por ejemplo, con **Flexbox**. Esto garantiza que los elementos permanezcan centrados independientemente del tamaño de la pantalla del dispositivo.

```
div{
   height: 100vh;

   display:flex;
   justify-content: center;
   align-items: center;
}
```

EJEMPLO:

HTML:

```
<div>
  <h1>Centrado según tamaño viewport</h1>
</div>
```

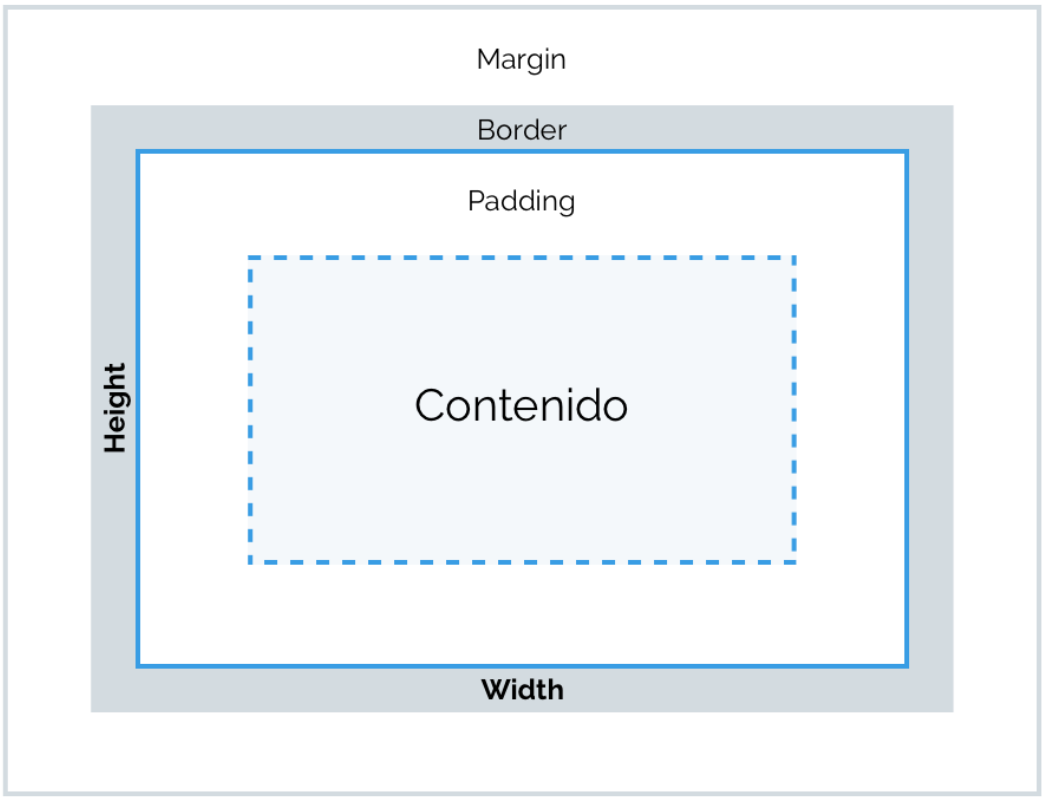
CSS:

```
*{margin:0;padding:0;}
div{
  height: 100vh;
  background-color: #EEE;
  display:flex;
  justify-content: center;
  align-items: center;
}
```

Modelo de cajas: márgenes, relleno y bordes

Cualquier elemento incluido en un documento HTML dispone de una estructura tipo caja que se puede modificar usando las propiedades CSS. Las propiedades más importantes de las cajas o contenedores son las siguientes: **margin** (margen externo)**, border** (borde) y **padding** (margen interno).

Las propiedades operan en el siguiente orden: **superior, derecha, inferior e izquierda**.



Padding o relleno

El **padding** es el **margen interno** de un elemento, también se le llama relleno y es la cantidad de espacio entre el borde y el contenido del elemento. Veamos una tabla con las propiedades para dar un margen interno a un elemento.

Propiedad	Descripción	Valores
`padding-top` `padding-right` `padding-bottom` `padding-left`	Tamaño del relleno superior, derecho, inferior e izquierdo	longitud | porcentaje
`padding`	Tamaño del relleno	longitud | porcentaje {1,4}

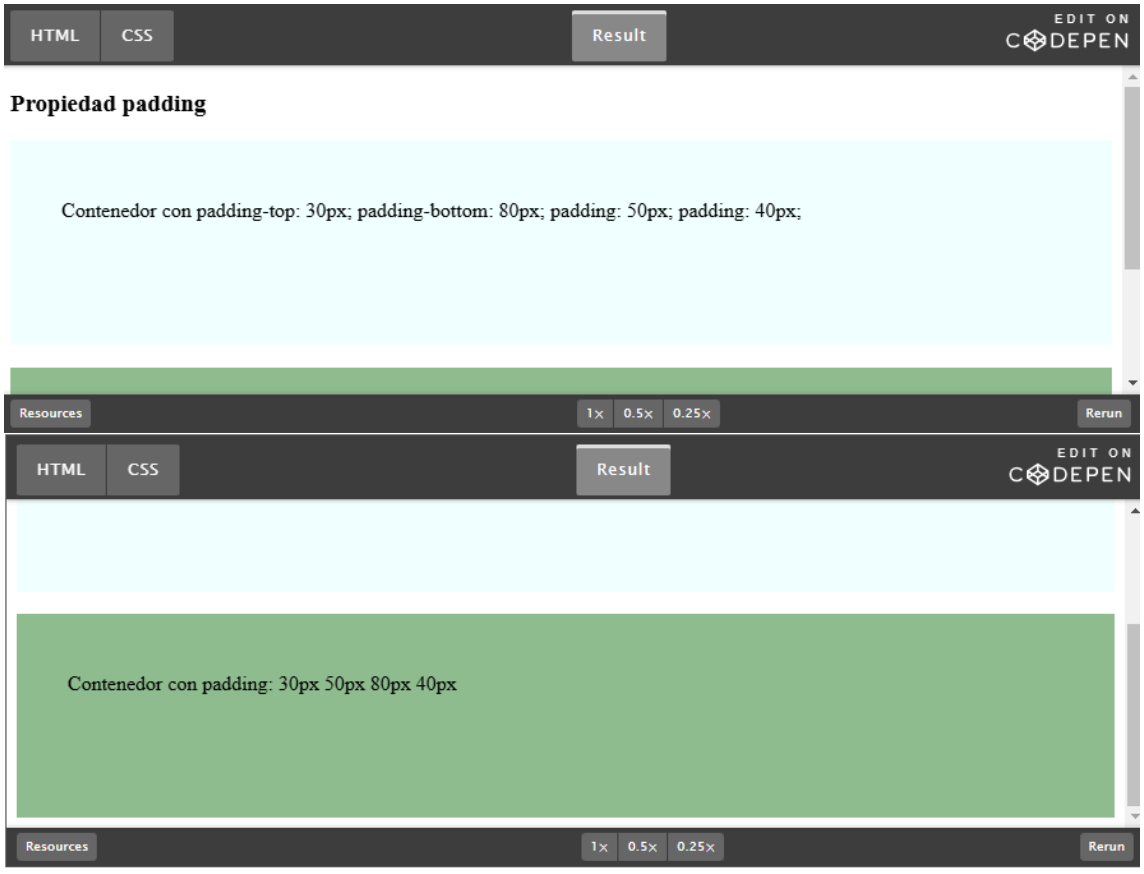
Tabla 1: Padding

Valores del padding (también aplicables a la propiedad margin):

- **Un valor:** se aplica el estilo a los **4 lados**.
- **Dos valores:** el primer valor se aplica a **arriba y abajo**, el segundo valor se aplica a **derecha e izquierda**.
- **Tres valores:** el primer valor se aplica a **arriba**, el segundo valor a **derecha e izquierda** y el tercer valor se aplica a **abajo** del elemento.
- **Cuatro valores:** el primer valor se aplica a **arriba**, el segundo valor se aplica a la **derecha**, el tercer valor se aplica a **abajo** y el cuarto valor se aplica a la **izquierda**.

Ejemplo Padding

```
div.a {
   padding-top: 30px;
   padding-bottom: 80px;
   padding-right: 50px;
   padding-left: 40px;
   border: 1px solid black;
   background-color: azure;
 }
 div.b { /* Propiedad corta */
   padding: 30px 50px 80px 40px; /*superior, derecha, inferior e izquierda*/
   border: 1px solid black;
   background-color: darkseagreen;
 }
```



Margin

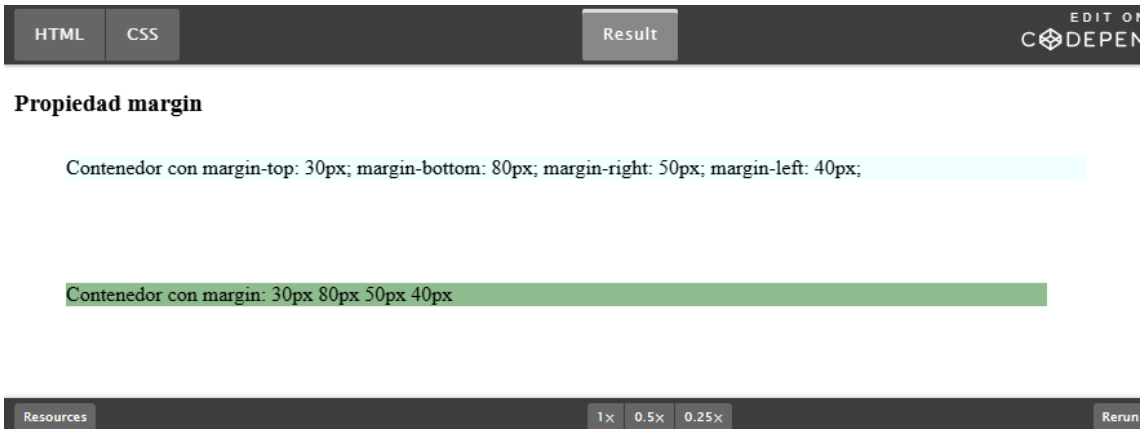
El **margin** es el **margen externo** de un elemento, fuera de cualquier borde definido. Veamos las distintas propiedades para dar estilo a los márgenes de un elemento.

Propiedad	Descripción	Valores
`margin-top` `margin-right` `margin-bottom` `margin-left`	Tamaño del margen superior, derecho, inferior e izquierdo	longitud | porcentaje | auto
`margin`	Ancho de los márgenes	longitud | porcentaje | auto {1,4}

Tabla 2: Margin

```
div.a {
   margin-top: 30px;
   margin-bottom: 80px;
   margin-right: 50px;
   margin-left: 40px;
   border: 1px solid black;
   background-color: azure;
 }
 div.b { /* Propiedad corta */
   margin: 30px 80px 50px 40px;
   border: 1px solid black;
   background-color: darkseagreen;
 }
```

EJERCICIO margin:



HTML:

```
<html>
<head>
  <meta charset="utf-8">
  <title>Propiedad margin</title>
  <link rel="stylesheet" href="style.css">
```

```
</head>
<body>
  <h3>Propiedad margin</h3>

  <div class="a" border="1">
   <p>Contenedor con margin-top: 30px; margin-bottom: 80px; margin-right: 50px; margin-left:
40px;</p>
  </div>
  <div class="b" border="1">
   <p>Contenedor con margin: 30px 80px 50px 40px</p>
  </div>
</body>
</html>
```

CSS:

```
div.a {
  margin-top: 30px;
  margin-bottom: 80px;
  margin-right: 50px;
  margin-left: 40px;
  background-color: azure;
}
div.b { /*Propiedad corta*/
  margin: 30px 80px 50px 40px;
  background-color: darkseagreen;
}
```

Bordes

La propiedad ***border*** en CSS permite especificar el estilo, el ancho y el color de los bordes de un elemento. Puedes usar diferentes valores para crear distintos tipos de bordes, como líneas lisas, de puntos, redondeados, etc. Veamos las propiedades para dar estilo a los bordes de un elemento.

Propiedad	Descripción	Valores
`border-top-width` `border-right-width` `border-bottom-width` `border-left-width`	Anchura del borde superior, derecho, inferior o izquierdo	thin | medium | thick | longitud
`border-width`	Anchura del borde (reducida)	thin | medium | thick | longitud {1,4}
`border-top-color` `border-right-color` `border-bottom-color` `border-left-color`	Color del borde superior, derecho, inferior e izquierdo	color | transparent
`border-color`	Color del borde (reducida)	color | transparent {1,4}
`border-top-style` `border-right-style` `border-bottom-style` `border-left-style`	Estilo del borde superior, derecho, inferior e izquierdo	none | hidden | dotted | dashed | solid | double | groove | ridge | inset | outset
`border-style`	Estilo del borde (reducida)	none | hidden | dotted | dashed | solid | double | groove | ridge | inset | outset {1,4}

`border-top` `border-right` `border-bottom` `border-left`	border-top-width | border-top-style | border-top-color
`border`	border-width | border-style | border-color
`border-radius`	longitud | porcentaje {1,4}

Tabla 3: Bordes

***Las propiedades del borde no tendrán efecto hasta que se defina la propiedad **border-style**.*

Ejemplo border

```
.a { border: 4px solid blue; }
.b { border: 6px dotted green; }
.c { border: 3px dashed purple; }
.d {
     border-top: 3px double orange;
     border-right: 3px double brown;
     border-bottom: 3px double brown;
     border-left: 3px double brown;
 }
```

HTML:

```
<h3>Propiedad border</h3>
<p class="a">Párrafo con un borde sólido de color azul</p>
<p class="b">Párrafo con borde verde punteado</p>
<p class="c">Párrafo con borde violeta a rayas</p>
<p class="d">Párrafo con borde mixto</p>
```

CSS:

```
.a { border: 4px solid blue; }
.b { border: 6px dotted green; }
.c { border: 3px dashed purple; }
.d {
    border-top: 3px double orange;
    border-right: 3px double brown;
    border-bottom: 3px double brown;
    border-left: 3px double brown;
 }
```

Ejemplo border-radius

```
.a { background-color: blue; border-radius: 20px 20px 0 0; height:100px; width: 100px;}
.b { background-color: green; border-radius: 0 0 40px 40px; height:100px; width:
100px;}
.c { background-color: purple; border-radius: 50px; height:100px; width: 100px;}
```

HTML:

```
<div class="a"></div><br>
<div class="b"></div><br>
<div class="c"></div><br>
```

CSS :

```
.a { background-color: blue; border-radius: 20px 20px 0 0; height:100px; width: 100px; }
.b { background-color: green; border-radius: 0 0 40px 40px; height:100px; width: 100px; }
.c { background-color: purple; border-radius: 50px; height:100px; width: 100px; }
```

Posición y comportamiento de contenedores en CSS

En el proceso de creación de una web es imprescindible organizar elementos como imágenes, textos o tablas. Para ello, necesitaremos conocer los elementos de ordenación y las propiedades que nos ayudan a organizar todos los componentes. Las propiedades más importantes se definen en la siguiente tabla y se explican a continuación.

Propiedad	Descripción	Valores
`display`	Comportamiento del contenedor	inline | block | inline-block | none
`position`	Esquema de posicionamiento	static | relative | absolute | fixed | sticky
`top` `right` `bottom` `left`	Desplazamiento de la caja respecto al borde superior, derecho, inferior o izquierdo	longitud | porcentaje | auto
`float`	Posicionamiento flotante	left | right | none
`clear`	Control de cajas adyacentes a las float	none | left | right | both
`z-index`	Nivel de la capa	auto | número entero
`box-sizing`	Control de bordes y relleno en el comportamiento del contenedor	content-box | border-box
`visibility`	Muestra u oculta un elemento ocupando el espacio	visible | hidden

Tabla 1: Posición y comportamiento de las cajas o contenedores

Las propiedades `top`, `bottom`, `left` y `right` se utilizan para posicionar elementos en una página web cuando se está usando posicionamiento mediante las propiedades de CSS como `position: relative`, `position: absolute`, `position: fixed` o `position: sticky`; estas propiedades definen distancias desde los bordes del contenedor de referencia (que puede ser el contenedor padre o la ventana del navegador) y sus valores pueden expresarse en unidades como píxeles (`px`), porcentaje (`%`) en relación al contenedor padre, unidades relativas al tamaño de fuente como `em` o `rem`, entre otras.

Display

Valores: **none | inline | block | inline-block**

- **none:** los elementos se ocultan y no se muestra el espacio reservado.
- **inline:** los elementos se muestran en la misma línea (respetando el flujo) y no se aceptan las propiedades width, height ni márgenes verticales.

- **block:** los elementos se muestran en líneas independientes y se aceptan las propiedades width, height y márgenes verticales.
- **inline-block:** su comportamiento es una mezcla entre los dos anteriores, los elementos se muestran en la misma línea (respetando el flujo) y se aceptan las propiedades width, height y márgenes verticales.

Ejemplo display

Aplica los siguientes estilos sobre diferentes elementos y comprueba el resultado.

```
.a { display: none; }
.b { display: inline; width: 100px; height: 50px;}
.c { display: block; }
.d { display: inline-block; width: 100px; height: 50px;}
```

EJERCICIO:

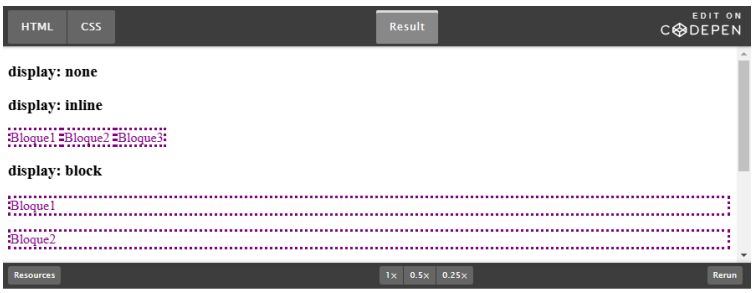
HTML:

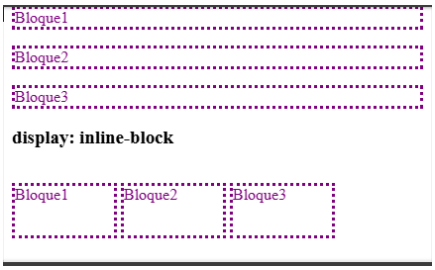
```
<h3>display: none</h3>
   <p class="a">Bloque1 </p>
   <p class="a">Bloque2 </p>
   <p class="a">Bloque3 </p>
 <h3>display: inline</h3>
   <p class="b">Bloque1 </p>
   <p class="b">Bloque2 </p>
   <p class="b">Bloque3</p>
<h3>display: block</h3>
   <p class="c">Bloque1 </p>
   <p class="c">Bloque2 </p>
   <p class="c">Bloque3 </p>
 <h3>display: inline-block</h3>
   <p class="d">Bloque1 </p>
   <p class="d">Bloque2 </p>
   <p class="d">Bloque3 </p>
```

CSS :

```
.a { display: none; }
.b { display: inline; width: 100px; height: 50px;}
.c { display: block; }
.d { display: inline-block; width: 100px; height: 50px;}
p  { color: purple; border: dotted;}
```

RESULTADO:





Position

Valores: **static | relative | absolute | fixed**

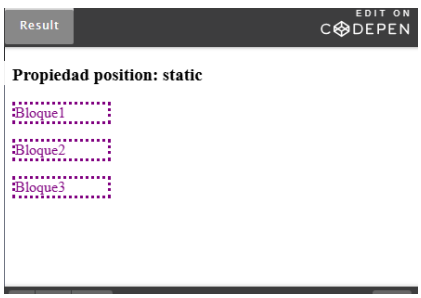
- **static:** los elementos se posicionan de acuerdo al flujo normal de la página. Es la **posición natural de los elementos**. No son afectados por las propiedades top, bottom, left y right.
- **relative:** los elementos se posicionan de forma relativa a su posición normal.
- **fixed:** los elementos se posicionan de forma relativa a la ventana del navegador. Su posición permanece fija aunque se desplace la ventana.
- **absolute:** los elementos se posicionan de forma relativa al primer elemento padre que tenga una posición distinta a static. Si un elemento con position: absolute no tiene un contenedor padre posicionado (con position: relative, absolute, fixed o sticky), entonces se posicionará en relación a la ventana del navegador.
- **sticky**: los elementos son posicionados de forma relativa hasta que su bloque contenedor alcanza un límite establecido.

Ejemplo position: static

Aplica los siguientes estilos sobre diferentes elementos y comprueba que los elementos mantienen su posición natural. Como puedes comprobar, las propiedades top, bottom, left y right no están permitidas con «position: static».

```
.a { position: static; }
.b { position: static; }
.c { position: static; }
```

EJEMPLO:



HTML:

```
<html>
<head>
   <meta charset="utf-8">
   <title>Propiedad position static</title>
   <link rel="stylesheet" href="style.css">
</head>
<body>
   <h3>Propiedad position: static</h3>

   <p class="a">Bloque1 </p>
   <p class="b">Bloque2 </p>
   <p class="c">Bloque3 </p>
</body>
</html>
```

CSS:

```
.a { position: static; top: 800px; left: 400px;} /*Como puedes ver, en position static no funcionan las
        propiedades top, bottom, left y right*/
.b { position: static; }
.c { position: static; }
p  { width: 100px; color: purple; border: dotted;}
```

Ejemplo position: relative

Aplica el siguiente estilo sobre un elemento y comprueba que con las propiedades left: 20px y top: 10px el elemento se desplaza 20px hacia la derecha y 10 px hacia abajo desde su posición por defecto (sin eliminar el hueco de su posición por defecto).


```
.bloque2 { position: relative; left: 20px; top: 10px;  }
```

HTML:

```
<html>
<head>
   <meta charset="utf-8">
   <title>Propiedad relative</title>
   <link rel="stylesheet" href="style.css">
</head>
<body>
   <h3>Propiedad position: relative</h3>
   <p>Bloque1 </p>
   <p class="bloque2">Bloque2 </p>
   <p>Bloque3 </p>
</body>
</html>
```

CSS :

```
.bloque2 { position: relative; left: 20px; top: 10px;  }
p  { width: 100px; color: purple; border: dotted;}
```

Ejemplo position: fixed

Tal y como puedes comprobar en el siguiente ejemplo, los elementos con position: **fixed** toman como referencia la ventana del navegador y permanecen fijos.

```
.bloque2 { position: fixed; top: 130px; left: 100px; }
```

HTML:

```
<h3>Propiedad position: fixed</h3>
<p>Bloque1 </p>
<p class="bloque2">Bloque2 </p>
<p>Bloque3 </p>
```

CSS:

```
.bloque2 { position: fixed; top: 130px; left: 100px; }
 p { width: 100px; color: purple; border: dotted;}
```

Ejemplo position: absolute

Establece el siguiente estilo sobre dos contenedores y comprueba cómo el elemento «.b» deja de seguir la posición del flujo normal de la página, sin crearse espacio alguno para el elemento, y se posiciona de forma relativa al primer elemento padre que tiene una posición distinta a static, en este caso el elemento «.a».

Si no hubiese ningún elemento padre con propiedad distinta a static, se ubicaría de forma relativa al contenedor inicial (puede ser <body>).

Haz la prueba cambiando el valor de position a «static» en el elemento a. Como puedes ver, su posición final está definida por los valores de top, right, bottom, y left.

```
1.  .a {
2.    position: relative; /*cambia este valor a static para ver la diferencia*/
3.    width: 300px;
4.    height: 300px;
5.    border: 2px dotted purple;
6.  }
7.  .b {
8.    position: absolute; /*Se posiciona de forma relativa al primer elemento padre c
    una posición distinta a static*/
9.    background-color: #EEEEEE;
10.   top: 60px;
11.   right: 0px;
12.   width: 200px;
13.   height: 80px;
14.   border: 3px solid green;
15. }
```

HTML:

```
<div class="a">
  <p>Contenedor con position: relative</p>
   <p>Contenedor con position: relative</p>
   <p>Contenedor con position: relative</p>
  <p>Contenedor con position: relative</p>
   <p>Contenedor con position: relative</p>
  <div class="b"><p>Contenedor con position: absolute</p></div>
</div>
```

CSS:

```
.a {
  position: relative; /*cambia este valor a static para ver la diferencia*/
  width: 300px;
  height: 300px;
  border: 2px dotted purple;
}
.b {
  position: absolute;/*Se posiciona de forma relativa al primer elemento padre que tiene una posición
        distinta a static*/
  background-color: #e1f4fc;
  top: 60px;
  right: 0px;
  width: 200px;
  height: 80px;
  border: 3px solid blue;
}
```

Ejemplo position: sticky

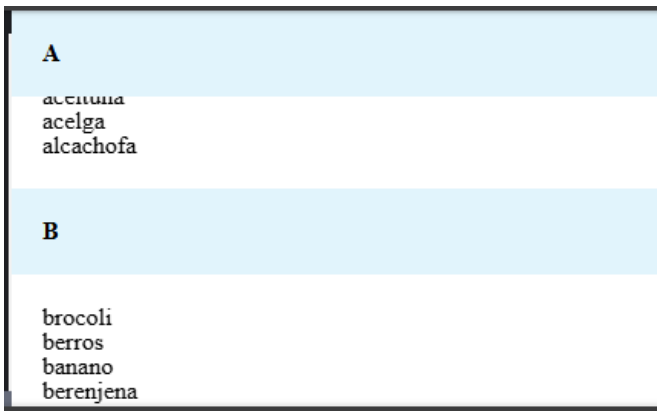
La posición **sticky** se usa cuando queremos que un elemento tenga una posición relativa hasta un punto y que luego cambie a una posición fija, usando solo CSS sin necesidad de código JavaScript.

Por ejemplo, si tenemos un banner de publicidad flotante en el **sidebar** y nos interesa que una vez aparezca al hacer **scroll** se mantenga fijo y visible.

La propiedad position: **sticky** se utiliza junto con valores de desplazamiento (**top, right, bottom, left**) para definir cuándo y dónde debe volverse fijo el menú o contenedor específico que queramos.

Otro ejemplo puede ser el que se muestra a continuación y que nos sirve para posicionar los encabezados en una lista alfabética en la parte alta del documento **HTML**.

```css
p {
  position: -webkit-sticky;
  position: sticky;
  top: 0px;
  background-color: #e1f4fc;
  padding: 20px;
  font-weight: bold;
}
ul{
  margin: 20px 0;
}
li{
  padding-left: 20px;
}
```



HTML:

```
<p>
 A
</p>

<ul>
 <li>
  aguacate
 </li>

 <li>
  ahuyama
 </li>

 <li>
  avena
 </li>

 <li>
  azucar
 </li>

 <li>
  ajo
 </li>

 <li>
```

```
  arroz
 </li>

 <li>
  arepa
 </li>

 <li>
  ajiaco
 </li>

 <p>
 B
</p>

<ul>
 <li>
  brocoli
 </li>

 <li>
  berros
 </li>

 <li>
  banano
 </li>

</ul> .....
```

CSS:

```
p {
 position: -webkit-sticky;
 position: sticky;
 top: 0px;
 background-color: #e1f4fc;
 padding: 20px;
 font-weight: bold;
}
ul{
 margin: 20px 0;
}
li{
 padding-left: 20px;
}
```

Float

Valores**: left | right | none**

Cuando a un elemento HTML se le aplica un estilo con la propiedad float, el elemento sale del flujo normal y aparece posicionado a la izquierda o a la derecha de su contenedor, donde el resto de elementos de la página se posicionarán alrededor.

Ejemplo float

Aplica los siguientes estilos sobre diferentes contenedores y comprueba el resultado.

```
.a { float: left; }
.b { float: right; }
.c { float: right; }
.d { clear: both; }
```

HTML:

```
<div class="a">
   <div></div>
</div>
<div class="a">
   <div></div>
</div>

<div class="b">
   <div></div>
</div>
<div class="b">
   <div></div>
</div>
<div class="c">
   <div></div>
</div>
<div class="d">
   <div></div>
</div>
```

CSS:

```
.a {
  float: left;
  padding: 10px;
}
.a div{
  height: 100px;
  width: 100px;
```

```
  background-color: #2980B9;
}
.b {
  float: right;
  padding: 10px;
}
.b div{
  height: 100px;
  width: 100px;
  background-color: #17A589;
}
.c {
  float: right;
  padding: 10px;
}
.c div{
  height: 100px;
  width: 100px;
  background-color: red;
}
.d {
  clear: both; /* Cambia este valor a float:left, float:right para ver resultados */
  padding: 10px;
}
.d div{
  height: 100px;
  width: 100px;
  background-color: purple;
}
```

Clear

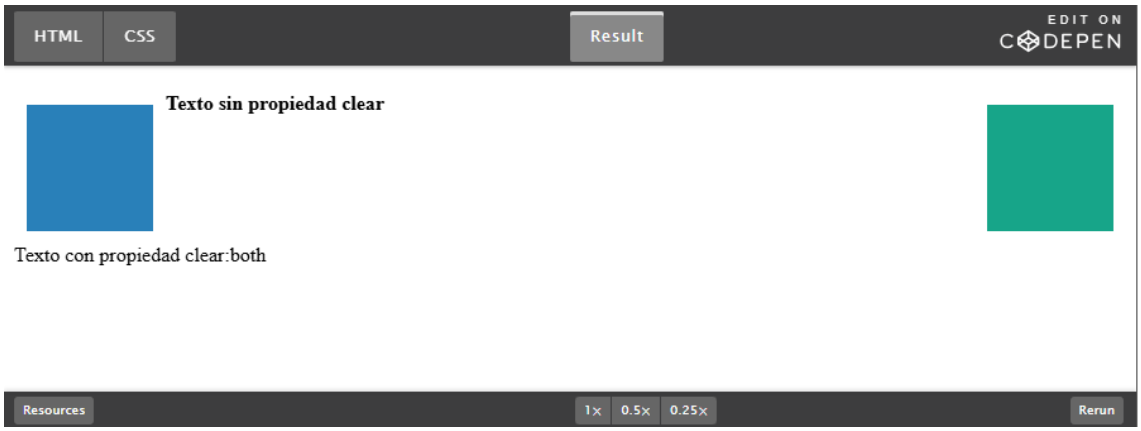
Valores: **none | left | right | both**

La propiedad clear establece si un elemento debe estar al lado de los elementos flotantes que lo preceden o si debe situarse bajo de ellos. Se suele utilizar para restaurar el flujo normal del documento y así los elementos dejan de flotar hacia la izquierda, la derecha o ambos lados.

Ejemplo clear

Crea un contenedor y aplícale la propiedad float con el valor left. Sitúa un texto bajo del contenedor creado utilizando la propiedad clear.

```
p{ clear: both; }
```



HTML:

```
<div class="a">
   <div></div>
</div>
<div class="b">
   <div></div>
</div>
<h4>Texto sin propiedad clear</h4>
<p>Texto con propiedad clear:both</p>
```

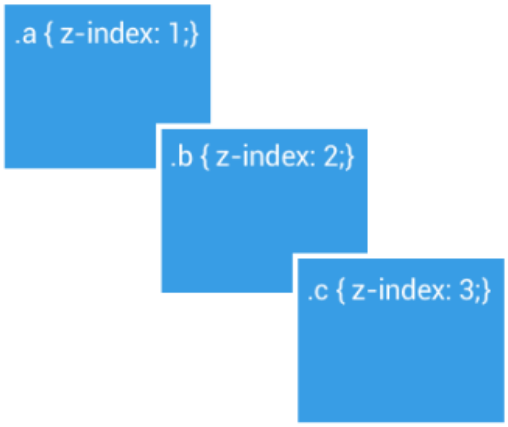
CSS:

```
.a {
  float: left;
  padding: 10px;
}
.a div{
  height: 100px;
  width: 100px;
  background-color: #2980B9; /*Azul*/
}
.b {
  float: right;
  padding: 10px;
}
.b div{
  height: 100px;
  width: 100px;
  background-color: #17A589;
}
p{
  clear: both;
}
```

Z-index

Valores**: auto | número entero**

Mediante el atributo **z-index** podemos organizar cada uno de los elementos del contenido de una página web. Como se puede apreciar en la imagen, cuando varios elementos se superponen, **los elementos con mayor valor z-index se sitúan por encima de los que tienen menor valor**.



Ejemplo z-index

Aplica los siguientes estilos sobre diferentes contenedores que se encuentren superpuestos y comprueba el resultado.

```
.a { z-index: 3;}
.b { z-index: 1;}
.c { z-index: 2;}
```

Box-sizing

Por defecto en el modelo de cajas de CSS, el ancho y alto asignado a un elemento es aplicado solo al contenido de la caja del elemento. Si el elemento tiene algún borde (border) o relleno (padding), este es entonces añadido al ancho y alto del tamaño de la caja o contenedor. Esto significa que cuando se define el ancho y alto, se tiene que ajustar el valor para permitir cualquier borde o relleno que se pueda añadir.

Valores: **content-box | border-box**

- **content-box** es el comportamiento CSS por defecto para el tamaño de la caja (box-sizing). Si se define el ancho de un elemento en 100 pixeles, la caja del contenido del elemento tendrá 100 pixeles de ancho, y el ancho de cualquier borde o relleno será añadido al ancho final desplegado.
- **border-box** toma en cuenta cualquier valor que se especifique de borde o de relleno para el ancho o alto de un elemento. Es decir, si se define un elemento con un ancho de 100 pixeles. Esos 100 pixeles incluirán cualquier borde o relleno que se añada, y la caja de contenido se encogerá para absorber ese ancho extra. Esta propiedad es especialmente útil para redimensionar cualquier elemento.

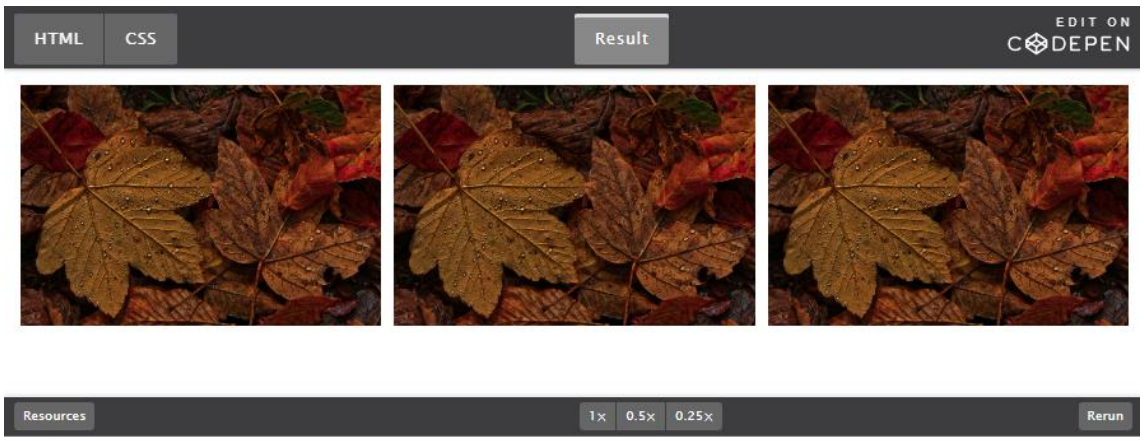
Ejemplo box-sizing

Crea un contenedor que ocupe el 100% del ancho de la pantalla. Posiciona tres imágenes en línea y define que cada imagen ocupe el 33,333%. Observa que el conjunto ocupa el 100% de la pantalla.

Si a continuación dotamos a las imágenes de un padding o relleno, el conjunto ocupará más del 100%. En este punto podríamos establecer un “box-sizing: border-box“ para incluir en el conjunto el relleno definido. Acuérdate de añadir los prefijos para los navegadores:

```
-webkit-box-sizing: border-box;
box-sizing: border-box;
```

Como puedes ver en el código, se han añadido los prefijos para navegadores necesarios para esta nueva propiedad.



HTML:

```
<div class="img-container">
  <img src="https://cdn.pixabay.com/photo/2019/11/02/21/45/maple-leaf-4597501_960_720.jpg">
  </div>
  <div class="img-container">
  <img src="https://cdn.pixabay.com/photo/2019/11/02/21/45/maple-leaf-4597501_960_720.jpg">
  </div>
  <div class="img-container">
  <img src="https://cdn.pixabay.com/photo/2019/11/02/21/45/maple-leaf-4597501_960_720.jpg">
  </div>
```

CSS:

```
.img-container {
  box-sizing: border-box;
  -webkit-box-sizing: border-box;
  float: left;
  width: 33.33%;
  padding: 5px;
}
.img-container img{
  width: 100%;
}
```

Visibility

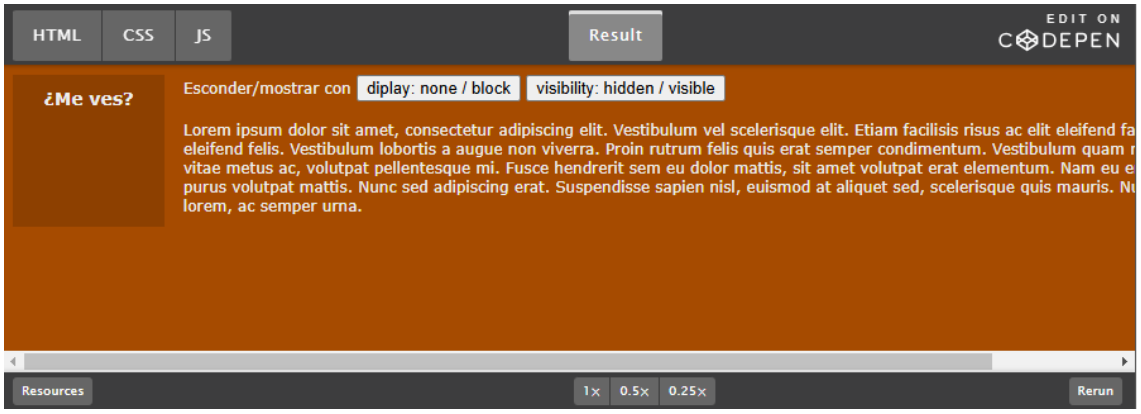
La propiedad visibility indica si un elemento es visible o permanece oculto (ocupando el mismo espacio).

Valores: **visible | hidden**

Ejemplo visibility

Aplica las siguientes propiedades sobre un elemento y observa las diferencias. Como puedes ver, la diferencia principal es que **display: none no reserva el espacio del elemento mientras que visibility: hidden sí**.

```
.a {display:none;}
.b {visibility: hidden;}
```



HTML:

```
<div>
 <div id="hide-me">¿Me ves?
 </div>
 <p>Esconder/mostrar con <button id="displayNone">diplay: none / block</button> <button id="visibilityHidden">visibility: hidden / visible</button>
 </p>
 <p>
  Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum vel scelerisque elit. Etiam facilisis risus ac elit eleifend facilisis. Mauris ut eleifend felis. Vestibulum lobortis a augue non viverra. Proin rutrum felis quis erat semper condimentum. Vestibulum quam nisl, commodo vitae metus ac, volutpat pellentesque mi. Fusce hendrerit sem eu dolor mattis, sit amet volutpat erat elementum. Nam eu enim sit amet purus volutpat mattis. Nunc sed adipiscing erat. Suspendisse sapien nisl, euismod at aliquet sed, scelerisque quis mauris. Nunc ac mattis lorem, ac semper urna.
 </p>
<div>
```

CSS:

```
body {
 background: #a64b00;
 color: #eee;
 font-family: verdana;
}

p {
 padding: 0;
 margin: 0 0 15px 0;
 font-size: 12px;
```

```
  width: 1000px;
}
#hide-me {
  width: 100px;
  height: 100px;
  background: #bf7130;
  display: block;
  padding: 10px;
  margin: 0 15px 0 0;
  font-size: 14px;
  font-weight: bold;
  float: left;
  text-align: center;
  background: rgba(0, 0, 0, .15);
}
```

JS:

```
$('#displayNone').click(function(e) {

  // Resetear, por si acaso has estado jugando con la otra propiedad
  $('#hide-me').css('visibility', 'visible');

  if( $('#hide-me').is(":visible") ) {
    $('#hide-me').css('display', 'none');
  } else {
    $('#hide-me').css('display', 'block');
  }
});

$('#visibilityHidden').click(function(e) {

  // Resetear, por si acaso has estado jugando con la otra propiedad
  $('#hide-me').css('display', 'block');

  if( $('#hide-me').css('visibility') != 'hidden' ) {
    $('#hide-me').css('visibility', 'hidden');
  } else {
    $('#hide-me').css('visibility', 'visible');
  }
});
```