

2023/24

U.D.3. USO DE ESTILOS.

3.3 *Propiedades CSS.*

```
343 .widget-area-sidebar {top: 0px; left: 0px; width: 100%; height: 100%; position: absolute; background-color: #f0f0f0; z-index: 1000; }
344 .widget-area-sidebar .wp-block-navigation {background-color: #f0f0f0; border-bottom: 1px solid #e0e0e0; padding: 10px; margin-bottom: 10px; }
345 .widget-area-sidebar .wp-block-navigation ul {list-style-type: none; padding-left: 0; }
346 .widget-area-sidebar .wp-block-navigation li {margin-bottom: 10px; }
347 .widget-area-sidebar .wp-block-navigation li a {color: #333; text-decoration: none; font-weight: bold; }
348 .widget-area-sidebar .wp-block-navigation li a:hover {color: #0070C0; }
349 }
350
351 /* =Menu */
352
353 #access {
354   display: inline-block;
355   height: 69px;
356   float: right;
357   margin: 11px 28px 0px 0px;
358   max-width: 880px;
359 }
360
361 #access ul {
362   font-size: 13px;
363   list-style: none;
364   margin: 0 0 0 -0.8125em;
365   padding-left: 0;
366   color: #0070C0;
367   text-align: right;
368 }
369
370 #access ul li {
371   display: inline-block;
372   margin-right: 10px;
373 }
```





ÍNDICE

1. Unidades de medida	5
1.1 Unidades absolutas	5
1.2 Unidades relativas	5
1.3 Usos de unidades	6
2. Tamaño de los elementos	7
2.1 ¿Qué son las propiedades CSS width y height?	7
2.2 Cómo usar las propiedades CSS width y height.....	8
2.3 ¿Qué son las propiedades CSS max-width y max-height?	8
2.4 ¿Qué son las propiedades CSS min-width y min-height?	9
2.5 Cómo usar las propiedades CSS max-width, min-width, max-height y min-height...9	
3. Media queries para cumplir con las necesidades del diseño responsive.....	11
3.1 Media query indicando el ancho mínimo de pantalla.....	11
3.2 Media query indicando el ancho máximo de pantalla	12
3.3 Media query combinando múltiples condiciones	12
3.4 Más ejemplos para definir el ancho del dispositivo.....	12
3.5 Otras propiedades o condiciones	14
3.6 Operadores lógicos en media queries	15
3.6.1 Operador AND.....	15
3.6.2 Operador NOT.....	15
4. El color	16
4.1 Colores con palabras reservadas o clave.....	16
4.2 Colores con RGB (decimal, hexadecimal y porcentual)	16
4.3 Colores con HSL	17
4.4 Colores con transparencia.....	18
5. El diseño del fondo	20
5.1 Color de fondo ('background-color')	20
5.2 Imagen de fondo ('background-image')	21
5.2.1 Fondo con degradados	22
5.2.1.1 Gradiante lineal ('linear-gradient')	22



5.2.1.2	Degrado radial ('radial-gradient').....	24
5.2.1.3	Degrado cónico ('conic-gradient')	26
5.2.1.4	Generadores de gradientes	27
5.3	Repetición y posición.....	28
5.3.1	background-repeat	28
5.3.2	background-position	29
5.4	background-size	31
5.5	Opacity	33
6.	Propiedades de texto	34
6.1	Interlineado de texto ('line-height').	34
6.2	Alineado de texto ('text-align')	35
6.3	Decoración de texto ('text-decoration')	35
6.4	Alineación vertical ('vertical-align').	38
6.5	Transformación del texto ("text-transform")	40
6.6	Letter spacing	41
6.7	Word spacing.....	42
6.8	Tabulado del texto ('text-indent')	43
6.9	Sombra del texto ('text-shadow')	43
6.10	Sombra a elementos caja ('box-shadow').....	45
6.10.1	Generadores de sombra online	46
7.	Propiedades de Fuente	47
7.1	Fuente de letra ('font-family')	47
7.2	Tamaño de letra ('font-size')	48
7.3	Estilo de letra ('font-style')	50
7.4	Grosor de letra ('font-weight')	51
7.5	Variante de fuente ('font-variant)	53
7.6	La propiedad fundamental 'font'	53
7.7	Fuentes externas.....	55
7.7.1	Fuentes web online	58
7.7.2	Convertidores de fuentes online	58
8.	Display	59
9.	Posicionar o Position	61



9.1	Static	61
9.2	Relative.....	62
9.3	Fixed.....	63
9.4	Absolute.....	64
9.5	Sticky	66
10.	Float y Clear.....	67
11.	Propiedades de visibilidad	70
11.1	'z-index'	70
11.2	'overflow'.....	71
11.3	'visibility'	72
12.	Propiedades de las tablas	74
12.1	'caption-side'.....	74
12.2	'empty-cells'	74
12.3	'border-collapse'	75
12.4	'border-spacing'	75
12.5	'table-layout'	76
13.	Propiedades de las listas en CSS	77
13.1	'list-style-type'	77
13.2	'list-style-image'	78
13.3	'list-style-position'	79
13.4	'list-style'	80
14.	Curores en CSS.....	81
15.	Definición de propiedades personalizadas.....	83
15.1	Uso de propiedades personalizadas	83
16.	Funciones matemáticas	85
16.1	calc().....	86
16.2	min()	87
16.3	max()	88
16.4	round()	89
16.5	attr()	90
17.	Transform	92



17.1 Tipos de transformaciones	93
18. Transition	97
19. Iconografía	100
19.1 Font Awesome.....	100
19.1.1 Registro u online	100
19.1.2 Descarga o local.....	102
19.2 Google Icons	103
19.2.1 Registro u online	103
19.2.2 Descarga o local.....	105
19.3 Icomoon.....	106
19.3.1 Registro u online	106
19.3.2 Descarga o local.....	108
20. Filtros CSS para aplicar efectos en imágenes.....	110
20.1 Filtros en CSS	110
21. El modo oscuro y claro con CSS.....	117
21.1 ¿Qué es prefers-color-scheme?	117
21.2 Cómo usar prefers-color-scheme	117
22. ANEXO I – Lista completa de colores	120



1. Unidades de medida

Como ya hemos visto, el tamaño de los elementos de una web se puede expresar en unidades **absolutas y relativas**.

1.1 Unidades absolutas

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

Unidad	Descripción
in	Pulgadas (1 pulgada = 2.54 cm)
cm	Centímetros
mm	Milímetros
pt	Puntos (1 pt = 1/72 pulgadas)
pc	Picas (1 pica = 12 puntos)

1.2 Unidades relativas

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

Unidad	Descripción
px	Píxeles (relativo al dispositivo). Es la unidad más utilizada y representa un punto en la pantalla. Se usa para tamaños fijos y proporciona control preciso sobre el diseño.
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). Es útil para hacer diseños fluidos y responsivos teniendo en cuenta la relación de los elementos con su contenedor padre .
vh y vw	Representan un porcentaje del ancho y alto de la ventana del navegador , respectivamente (medidas relativas de acuerdo al viewport). 1vh = 1% de la altura del viewport. 100vh = altura del viewport. Son útiles para crear diseños responsive basados en el tamaño de la pantalla .



fr	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.
----	---

1.3 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**.

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**.

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.

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

Hay que tener en cuenta que el tamaño base definido en los navegadores puede ser modificado por los usuarios.



2. Tamaño de los elementos

En este apartado vamos a aprender cómo usar las **propiedades CSS *width*, *height*, *min-width*, *max-height* y *min-height*** para controlar el tamaño de los elementos en una página web.

Veamos una tabla resumen de las propiedades que vamos a tratar en este apartado:

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

2.1 ¿Qué son las propiedades CSS *width* y *height*?

Las **propiedades CSS *width* y *height*** se usan para **establecer el ancho y el alto de un elemento**. Por defecto, estas propiedades **se refieren al área de contenido** del elemento, es decir, el espacio que ocupa el texto, las imágenes u otros elementos dentro del elemento. El área de contenido **no incluye el padding (relleno), el border (borde) ni el margin (margen) del elemento**.

Sin embargo, podemos cambiar este comportamiento con la propiedad CSS ***box-sizing***, que nos permite definir qué partes del elemento se incluyen en el cálculo del ancho y el alto.



2.2 Cómo usar las propiedades CSS width y height

Veamos algunos ejemplos de uso de **width** y **height** con diferentes unidades:

- Para darle un **ancho y un alto fijo a un elemento**, podemos usar **píxeles** como unidad. Por ejemplo, si queremos que un elemento tenga un ancho de 200px y un alto de 100px, podemos usar este código:

```
div {  
    width: 200px;  
    height: 100px;  
}
```

- Para darle un **ancho y un alto relativo al elemento contenedor**, podemos usar **porcentajes** como unidad. Por ejemplo, si queremos que un elemento ocupe el 50% del ancho y el 25% del alto del elemento padre, podemos usar este código:

```
div {  
    width: 50%;  
    height: 25%;  
}
```

2.3 ¿Qué son las propiedades CSS max-width y max-height?

Las propiedades CSS **max-width** y **max-height** se usan para establecer el ancho y el alto máximo de un elemento. Estas propiedades nos permiten limitar el tamaño de un elemento para que no supere cierto valor, incluso si le damos un ancho o un alto mayor con las propiedades **width** o **height**.

Esto puede ser muy útil para crear **diseños responsivos** que se adapten a diferentes tamaños de pantalla. Por ejemplo, podemos usar **max-width** para evitar que una imagen se salga del contenedor o que se distorsione al cambiar la resolución. También podemos usar **max-height** para controlar la altura de un elemento que tenga contenido variable, como un menú desplegable o una caja de texto.



2.4 ¿Qué son las propiedades CSS min-width y min-height?

Las propiedades CSS **min-width** y **min-height** se utilizan para establecer la anchura y la altura mínima de un elemento, respectivamente. Estas propiedades impiden que el elemento se reduzca a un ancho o alto menor que el valor especificado.

Las propiedades **min-width** y **min-height** se pueden utilizar para **garantizar que un elemento tenga un tamaño mínimo**, independientemente del tamaño del contenido que contenga. Por ejemplo, se pueden utilizar para garantizar que un elemento siempre tenga un tamaño mínimo de visualización, incluso en pantallas pequeñas.

2.5 Cómo usar las propiedades CSS max-width, min-width, max-height y min-height

Veamos algunos ejemplos de uso de **max-width**, **min-width**, **max-height** y **min-height** con diferentes unidades:

- 1) Para darle un **ancho y un alto máximo** a un elemento en **píxeles**, podemos usar esta unidad como valor. Por ejemplo, si queremos que una imagen no supere los 300px de ancho ni los 200px de alto, podemos usar este código:

```
img {  
    max-width: 300px;  
    max-height: 200px;  
}
```

- 2) Para darle un **ancho y un alto máximo** a un elemento en **porcentajes**, podemos usar esta unidad como valor. Por ejemplo, si queremos que una imagen no ocupe más del 50% del ancho ni del 25% del alto del elemento contenedor, podemos usar este código:

```
img {  
    max-width: 50%;  
    max-height: 25%;  
}
```



3) El siguiente código establece el **ancho máximo** del elemento **div** a 300 **píxeles**:

```
div {  
    max-width: 300px;  
}
```

4) El siguiente código establece el alto mínimo del elemento **div** a 100 píxeles:

```
div {  
    min-height: 100px;  
}
```

5) **Veamos un ejemplo práctico.** Para crear un contenido a pantalla completa que no se haga más grande al llegar a un ancho máximo, podemos usar la propiedad **max-width** junto con la propiedad **width**. Así, le damos al elemento un ancho relativo al tamaño de la pantalla, pero también le ponemos un límite absoluto.

Por ejemplo, si queremos que el contenido ocupe el 95% del ancho de la pantalla, pero que no supere los 1200px, podemos usar este código:

```
.content {  
    width: 95%; /* El contenido ocupa el 95% del ancho de la pantalla */  
    max-width: 1200px; /* El contenido no ocupará más de 1200px */  
    margin: 0 auto; /* Centrar el contenido */  
}
```

De esta forma, el contenido se adaptará al tamaño de la pantalla, pero no se hará más grande de lo necesario. Esto puede mejorar la legibilidad y el diseño de tu página web.



3. Media queries para cumplir con las necesidades del diseño responsive

Los diseños responsive o diseño web adaptable o adaptativo se lleva a cabo mediante la utilización de `@media` junto con las propiedades `min-width` (anchura mínima) o `max-width` (anchura máxima). El objetivo del diseño responsive es adaptar la apariencia de las páginas web al dispositivo que se esté utilizando para visitarlas.

3.1 Media query indicando el ancho mínimo de pantalla

A continuación, se muestra un **ejemplo de diseño responsive** con `@media` y la propiedad `min-width`, tomando como referencia el tamaño promedio de las pantallas que existen actualmente.

```
body{  
    background: orange;  
}  
  
div{  
    width: 80%;  
    height: 150px;  
    margin: 8%;  
    display: inline-block;  
    background: white;  
}  
  
@media screen and (min-width: 480px){  
    body{  
        background: #34A647;  
    }  
}  
  
@media screen and (min-width: 767px){  
    body{  
        background: #34A2A6;  
    }  
    div{  
        width: 40%;  
        margin: 4%;  
    }  
}  
  
@media screen and (min-width: 950px){  
    body{  
        background: #A76A9E;  
    }  
    div{  
        width: 28%;  
        margin: 2%;  
    }  
}
```

Se han establecido las dimensiones partiendo de un diseño **MobileFirst**, que consiste en diseñar primero para dispositivos móviles debido al aumento considerable de este tipo de dispositivos. Es más fácil llenar que limpiar.

A partir de 480px (aprox.), tenemos un móvil en posición horizontal. 480px es un **punto de ruptura**.

A partir de 767px (aprox.), tenemos una tablet. 767px es otro **punto de ruptura**.

Para el diseño responsive, se recomienda siempre utilizar %

A partir de 950px (aprox.), tenemos una pantalla de escritorio. 950px es otro **punto de ruptura**.



3.2 Media query indicando el ancho máximo de pantalla

Todos los estilos que se incluyan en el interior de la siguiente media query serán utilizados en pantallas que tengan un ancho máximo de 767px (punto de ruptura de 767px).

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

3.3 Media query combinando múltiples condiciones

Es posible combinar múltiples condiciones en una media query para aplicar estilos solo cuando se cumplan todas las condiciones. Todos los estilos que se incluyan en el interior de la siguiente media query serán utilizados en pantallas que tengan un ancho entre 768px y 1024px (puntos de ruptura de 768px y 1024px).

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

3.4 Más ejemplos para definir el ancho del dispositivo

- 1) Supongamos que queremos aplicar un estilo diferente en pantallas de ancho superior a 1024px, inferior a 1024px e inferior a 480px, tendríamos las siguientes *media queries*.

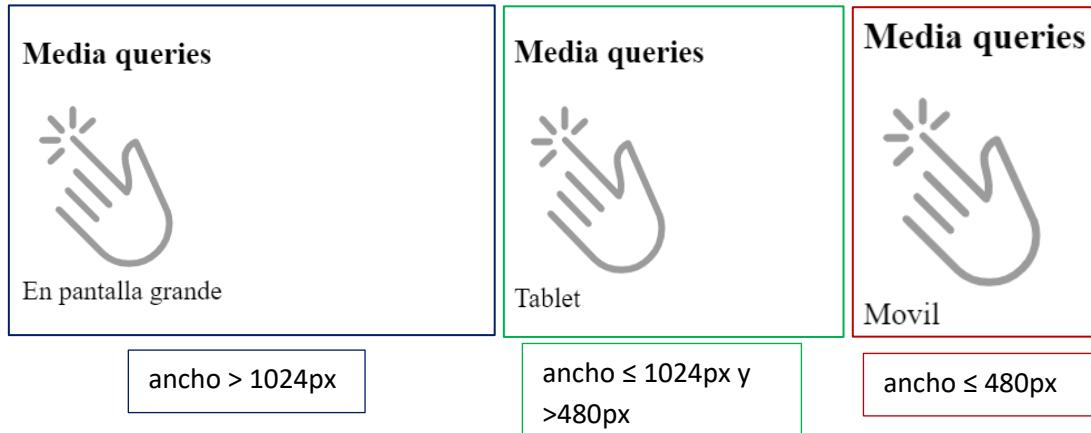
Código CSS:

```
.click:after {  
    content:"En pantalla grande";  
}  
@media (max-width: 1024px) {  
    .click:after {  
        content:"Tablet";  
    }  
}  
@media (max-width: 480px) {  
    .click:after {  
        content:"Movil";  
    }  
}
```



Código HTML:

```
<h3>Media queries</h3>
<div class="click">
  <br>
</div>
```



- 2) La primera *media query* especifica una hoja de estilo para ser utilizada cuando la ventana tiene un ancho entre 450 y 800 píxeles y la segunda *media query* para pantallas menores a 449 píxeles.

Código CSS:

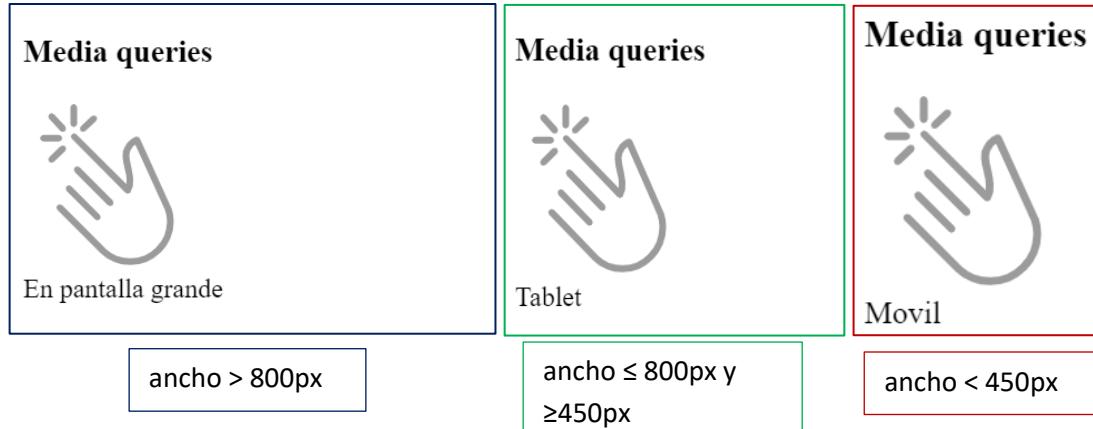
```
.click:after {
  content:"En pantalla grande";
}
@media (min-width: 450px) and (max-width: 800px) {
  .click:after {
    content:"Tablet";
  }
}

@media (max-width: 449px) {
  .click:after {
    content:"Movil";
  }
}
```



Código HTML:

```
<h3>Media queries</h3>
<div class="click">
  <br>
</div>
```



3.5 Otras propiedades o condiciones

Además del ancho hay muchas otras propiedades que se pueden utilizar para aplicar o no estilos. Las siguientes son algunas de ellas:

Nombre	Descripción
<i>width</i>	Anchura del <i>viewport</i>
<i>height</i>	Altura del <i>viewport</i>
<i>aspect-ratio</i>	Relación de aspecto anchura-altura del <i>viewport</i>
<i>orientation</i>	Orientación del <i>viewport</i>
<i>resolution</i>	Densidad de píxeles del dispositivo
<i>scan</i>	Proceso de escaneo del dispositivo
<i>grid</i>	Si el dispositivo es grid o bitmap
<i>update-frequency</i>	Velocidad de actualización del dispositivo para modificar la apariencia del contenido



<i>overflow-block</i>	Cómo maneja el dispositivo el contenido que excede los límites del <i>viewport</i> a lo largo del eje de bloque
<i>overflow-inline</i>	Cómo maneja el dispositivo el contenido que excede los límites del eje <i>inline</i>
<i>color</i>	Componente de número de bits por color del dispositivo, o cero si el dispositivo no es a color
<i>color-index</i>	Número de entradas en la tabla de búsqueda de color del dispositivo, o cero si el dispositivo no usa una tabla
<i>monochrome</i>	Bits por píxel en el buffer de marco monocromático del dispositivo, o 0 si el dispositivo no es monocromático
<i>hover</i>	Si se puede posicionar el puntero sobre los elementos

3.6 Operadores lógicos en media queries

Como hemos visto, las *media queries* nos permiten utilizar operadores lógicos para comprobar si se cumple una condición.

3.6.1 Operador AND

El siguiente código aplicará estilos cuando la pantalla tenga un ancho mínimo de 700px y la orientación de la misma sea horizontal (*landscape*).

```
@media (min-width: 700px) and (orientation: landscape) { ... }
```

3.6.2 Operador NOT

El siguiente código aplicará estilos cuando no se cumpla la condición especificada, es decir a cualquier elemento que no sea una pantalla convencional ni sea monocromo.

```
@media not screen and (monochrome) { ... }
```



4. El color

Uno de los primeros estilos en los que debemos pensar cuando diseñamos una interfaz web es en la paleta de colores que vamos a aplicar.

Los valores de los **colores** se pueden definir mediante su **nombre o palabra clave**, en **código hexadecimal** (#RRGGBBAA) o mediante sus valores en los siguientes **formatos**:

- **RGB** (Red, Green, Blue).
- **HSL** (Hue, Saturation, Lightness).
- **RGBA** (Red, Green, Blue, Alpha).
- **HSLA** (Hue, Saturation, Lightness, Alpha).

Todos los navegadores actuales soportan **140 nombres** de colores. La lista completa la puedes ver en el **Anexo I**.

4.1 Colores con palabras reservadas o clave

Existen colores en CSS que de por sí ya tienen un **nombre propio**, es decir, existe una palabra reservada para cada uno de estos colores. Su sintaxis, utilizando la propiedad **color**, es la siguiente:

```
color: red;  
color: orange;  
color: tan;  
color: rebeccapurple;  
color: transparent;
```

4.2 Colores con RGB (decimal, hexadecimal y porcentual)

Para dar color a nuestra interfaz web, podemos utilizar el **sistema RGB** que ya hemos visto. El sistema **RGB** forma el color mediante la **mezcla de luces** roja, verde y azul. Por tanto, si



no tenemos luz obtendremos el color negro, y si tenemos las tres luces antes mencionadas, obtendremos el color blanco.

A la hora de representar cada uno de los colores, es posible utilizar tanto el sistema de **numeración decimal** (0 a 255) como el **hexadecimal**.

En CSS, un color puede ser especificado en numeración **decimal o en porcentajes**, utilizando la siguiente sintaxis: `rgb(red, green, blue)`; Además, también se puede expresar en **hexadecimal**.

A continuación, se muestra un ejemplo con la salida correspondiente.

```
<!DOCTYPE html>
<html>
<head>
<style>
.luzRGB-decimal {
    background-color: rgb(0, 255, 0);
}

.luzRGB-porcentajes {
    background-color: rgb(70%, 40%, 75%);
}

.luzRGB-hex {
    background-color: #55BB88;
}

.luzRGB-hex-reducido {
    background-color: #5B8;
}

</style>
</head>
<body>

<h1>Specify colors using RGB values</h1>

<h2 class="luzRGB-decimal">rgb(0, 255, 0)</h2>
<h2 class="luzRGB-porcentajes">rgb(70%, 40%, 75%)</h2>
<h2 class="luzRGB-hex">#55BB88</h2>
<h2 class="luzRGB-hex-reducido">#5B8</h2>

</body>
```

RGB en decimal
RGB en porcentajes
RGB en hexadecimal

Specify colors using RGB values

rgb(0, 255, 0)
rgb(70%, 40%, 75%)
rgb(25, 40%, 128)
#55BB88
#5B8

Cuando se repite un mismo dígito en una de las luces, se puede simplificar y poner un solo dígito.

4.3 Colores con HSL

Además de dar color a nuestra interfaz web mediante el sistema RGB, podemos darle color mediante el sistema **HSL**.



HSL es el acrónimo de **hue** (*matiz o tono*), **saturation** (*saturación*), y **lightness** (*brillo o luminosidad*). Por tanto, el color se formará en base a los valores que le demos a la tonalidad, a la saturación, y a la luz.

Su sintaxis es la siguiente: **hsl(hue, saturation, Lightness);**

- **Hue:** es el grado en el nos situamos dentro del círculo cromático. 0º es rojo, 120º es verde, y 240º es azul.
- **Saturation:** es el porcentaje de saturación. 0% significa una sombra de gris, y 100% es el color completo o full color.
- **Lightness:** es el porcentaje de luz. 0% es negro y 100% es blanco.

A continuación, se presenta un ejemplo, con su correspondiente salida.

```
<!DOCTYPE html>
<html>
<style>
div {
    background-color: hsl(180, 50%, 50%);
    color: hsl(0, 0%, 100%);
    padding: 20px;
}
</style>
<body>

<div>

<p>London is the capital city of England.  
It is the most populous city in the United Kingdom,  
with a metropolitan area of over 13 million  
inhabitants.</p>

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

London is the capital city of England. It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants.

4.4 Colores con transparencia

En CSS, podemos dar color usando los sistemas RGB y HSL vistos anteriormente, y además les podemos dar **opacidad o transparencia**.

⇒ **RGB con un canal alfa u opacidad:** **rgba (red, green, blue, alpha);**

Las luces rojo, verde y azul se definen como ya hemos visto, en porcentaje o en decimal de 0 a 255. Alpha define la opacidad y es un número entre **0.0** (totalmente transparente) y **1.0** (completamente opaco).



```
<!DOCTYPE html>
<html>
<head>
<style>
#p1 {background-color:rgba(255,0,0,0.3);}
#p2 {background-color:rgba(0,255,0,0.3);}
#p3 {background-color:rgba(0,0,255,0.3);}
#p4 {background-color:rgba(192,192,192,0.3);}
#p5 {background-color:rgba(255,255,0,0.3);}
#p6 {background-color:rgba(255,0,255,0.3);}
</style>
</head>
<body>

<h1>The rgba() Function</h1>

<p>RGB colors with opacity:</p>
<p id="p1">Red</p>
<p id="p2">Green</p>
<p id="p3">Blue</p>
<p id="p4">Grey</p>
<p id="p5">Yellow</p>
<p id="p6">Cerise</p>

</body>
</html>
```

⇒ HSL con un canal alfa u opacidad:

`hsla(hue, saturation, lightness, alpha);`

El tono, la saturación y el brillo se definen como ha hemos visto en el apartado anterior. Alpha define la opacidad y es un número entre **0.0** (totalmente transparente) y **1.0** (completamente opaco).

```
<!DOCTYPE html>
<html>
<head>
<style>
#p1 {background-color:hsla(120,100%,50%,0.3);}
#p2 {background-color:hsla(120,100%,75%,0.3);}
#p3 {background-color:hsla(120,100%,25%,0.3);}
#p4 {background-color:hsla(120,60%,70%,0.3);}
#p5 {background-color:hsla(290,100%,50%,0.3);}
#p6 {background-color:hsla(290,60%,70%,0.3);}
</style>
</head>
<body>

<h1>The hsla() Function</h1>

<p>HSL colors with opacity:</p>
<p id="p1">Green</p>
<p id="p2">Light green</p>
<p id="p3">Dark green</p>
<p id="p4">Pastel green</p>
<p id="p5">Violet</p>
<p id="p6">Pastel violet</p>

</body>
</html>
```

The rgba() Function

RGB colors with opacity:

Red

Green

Blue

Grey

Yellow

Cerise

The hsla() Function

HSL colors with opacity:

Green

Light green

Dark green

Pastel green

Violet

Pastel violet



5. El diseño del fondo

En cuanto al diseño del fondo, existen ciertas propiedades fundamentales para establecer el fondo de cada elemento.

Algunas de las propiedades relacionadas con el fondo más utilizadas son las siguientes:

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

5.1 Color de fondo ('background-color')

El color de fondo de un elemento en HTML se define mediante esta propiedad. Como ya se ha visto, el color puede expresarse mediante código hexadecimal, código RGB, sistema HSL o con palabra de color clave ya definidas. La palabra reservada, *transparent*, se utiliza para que el elemento tome el valor del elemento situado debajo de él, es decir, del que hereda.

La **sintaxis** es la siguiente:

```
etiqueta html {  
    background-color: color | transparent | inherit ;  
}
```



Ejemplo:

```
p {  
    background-color: teal;  
    color: white;  
}  
  
h1 {  
    background-color: rgb(153,102,153);  
    color: rgb(255,255,204);  
}  
  
h3 {  
    background-color: #777799;  
    color: #FFFFFF;  
}
```

5.2 Imagen de fondo ('background-image')

La propiedad '**background-image**' permite utilizar una imagen como fondo de cualquier elemento. Si se utiliza esta propiedad, **la imagen quedará por encima del color establecido con la propiedad anterior**. Se recomienda indicar un color de fondo, para que sea este el que aparezca en el navegador si ocurre algún problema con la imagen seleccionada, de esta forma se evitarán apariencias indeseadas.

La **sintaxis** es la siguiente:

```
etiqueta html {  
    background-image: url | none | inherit ;  
}
```

Es recomendable crear una carpeta de imágenes que se encuentre en el mismo directorio que los archivos CSS y que almacene todas las imágenes utilizadas en el diseño de las páginas.

Si la imagen que se quiere mostrar es demasiado grande para el fondo del elemento, solamente se mostrará la parte de imagen comprendida en el tamaño del elemento. Si la imagen es más pequeña que el elemento, CSS la repetirá horizontalmente y verticalmente hasta llenar el fondo del elemento.



5.2.1 Fondo con degradados

Los **degradados CSS** permiten mostrar transiciones suaves entre dos o más colores especificados. Hay que tener especial cuidado con esta propiedad, ya que necesitará, en algunos casos, el **prefijo del navegador** para que sea compatible.

CSS especifica los siguientes **tres tipos** de degradados:

- 1) **Gradientes lineales** (van hacia abajo/arriba/izquierda/derecha/diagonalmente).
- 2) **Degradiados radiales** (definidos por su centro).
- 3) **Degradiados cónicos** (giran alrededor de un punto central)

5.2.1.1 Gradiente lineal ('linear-gradient')

Para crear un degradado lineal se deben definir al menos **dos topes de color** (o "color stop").

Los topes de color son los colores entre los que se quieren hacer transiciones suaves.

También se puede establecer un punto de partida y una dirección (o un ángulo) junto con el efecto de degradado.

Su **sintaxis** es la siguiente:

```
etiqueta html {  
    background-image: linear-gradient(direction|angle, color-  
                                stop1, color-stop2, ...);  
}
```

```
etiqueta html {  
    background-image: linear-gradient(direction|angle, color-stop1  
                                size, color-stop2 size, ...);  
}
```

```
etiqueta html {  
    background-image: linear-gradient(color-stop1, color-stop2, ...);  
}
```

Algunos de los valores de *direction* pueden ser:

- *to bottom*: de arriba a abajo. Es el valor por defecto.



- *to right*: de izquierda a derecha.
- *to bottom right*: en diagonal de arriba a la izquierda a abajo a la derecha.
- etc.

Si queremos un **mayor control** sobre la dirección del gradiente, podemos **definir un ángulo**, en lugar de las direcciones predefinidas (hacia abajo, hacia arriba, hacia la derecha, hacia la izquierda, hacia abajo a la derecha, etc.).

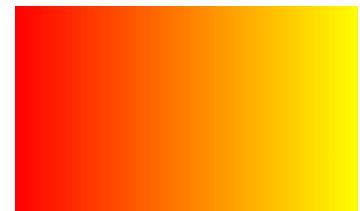
```
etiqueta html {  
    background-image: linear-gradient(angle, color-stop1, color-  
                                stop2, ...);  
}
```

Algunos de los valores de *angle* pueden ser:

- 0deg, que equivale a “hacia arriba”.
- 90deg, que equivale a “hacia la derecha”.
- 180deg, que equivale “hacia abajo”.

A continuación, se muestran algunos **ejemplos**.

```
#grad {  
    background-image: linear-gradient(to right, red, yellow);  
}
```



```
#grad {  
    background-image: linear-gradient(180deg, red, yellow);  
}
```

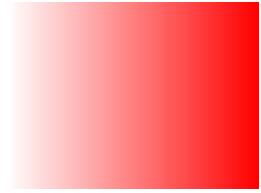


```
#grad {  
    background-image: linear-gradient(red, yellow, green);  
}
```





```
#grad {  
    background-image: linear-gradient(to right, rgba(255,0,0,0), rgba(255,0,0,1));  
}
```



```
#grad {  
    background-image: linear-gradient(to bottom right, red, yellow);  
}
```



```
#grad1 {  
height: 200px;  
background-image: linear-gradient(red 0%, blue 20%, green 50%,  
pink 70%, yellow 90%);  
}
```



5.2.1.2 Degradado radial ('radial-gradient')

Para crear un degradado radial, se deben también definir al menos dos topes de color o 'color stops'.

Su **sintaxis** es la siguiente:

```
etiqueta html {  
    background-image: radial-gradient(shape, start-color, ..., last-color);  
}
```

```
etiqueta html {  
    background-image: radial-gradient(shape size, start-color, ...,  
                                     last-color);  
}
```

```
etiqueta html {  
    background-image: radial-gradient(shape, start-color size, ...,  
                                     last-color size);  
}
```



```
etiqueta html {  
    background-image: radial-gradient(start-color size, ..., last-color);  
}
```

```
etiqueta html {  
    background-image: radial-gradient(shape size at position, start-color, ..., last-color); }
```

Por defecto, el valor de *shape* es *ellipse*, *size* es *farthest-corner*, y *position* es *center*.

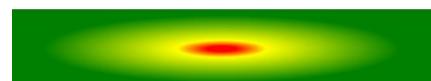
El parámetro de *shape* también puede tomar el valor de *circle*.

El parámetro de *size* define el tamaño del degradado. Puede tomar los siguientes valores:

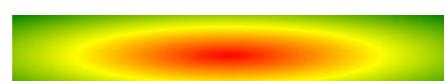
- *closest-side*: la **forma final** del degradado se encuentra con **el lado de la caja más cercano a su centro** (para los círculos) o se encuentra con los lados vertical y horizontal más cercanos al centro (para las elipses).
- *farthest-side*: similar al anterior, excepto que la **forma final** está dimensionada para encontrarse con el **lado de la caja más alejado de su centro** (o los lados vertical y horizontal).
- *closest-corner*: la forma de terminación del degradado tiene un **tamaño** tal que **coincide exactamente con la esquina más cercana** de la caja desde su centro.
- *farthest-corner*: es el valor por defecto. La **forma final** del gradiente tiene un tamaño tal que se encuentra exactamente con la **esquina más lejana de la caja desde su centro**.

A continuación, se muestran algunos **ejemplos**:

```
#grad {  
    background-image: radial-gradient(red 5%, yellow 15%, green 60%);  
}
```

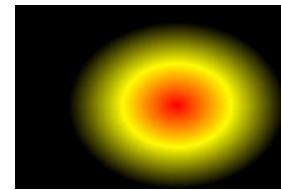


```
#grad {  
    background-image: radial-gradient(red, yellow, green);  
}
```





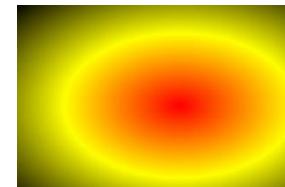
```
#grad1 {  
    height: 150px;  
    width: 220px;  
    background-image: radial-gradient(closest-side at 60% 55%, red,  
yellow, black);  
}
```



```
#grad {  
    background-image: radial-gradient(circle, red, yellow, green);  
}
```



```
#grad4 {  
    height: 150px;  
    width: 220px;  
    background-image: radial-gradient(farthest-corner at 60% 55%, red,  
yellow, black);  
}
```



5.2.1.3 Degradado cónico ('conic-gradient')

Un degradado cónico es un degradado con **transiciones de color rotadas alrededor de un punto central**.

Para crear un degradado cónico se debe definir al menos dos colores.

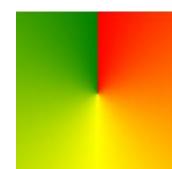
Su **sintaxis** es la siguiente:

```
etiqueta html {  
background-image: conic-gradient([from angle] [at position,] color [degree],  
color [degree], ...); }
```

Por **defecto**, el **ángulo** es *0deg* y la **posición** es *center*. Si no se especifica ningún grado, los colores se repartirán por igual alrededor del punto central.

A continuación, se muestran algunos ejemplos:

```
#grad1 {  
    height: 200px;  
    width: 200px;  
    background-image: conic-gradient(red, yellow, green);  
}
```

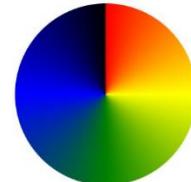




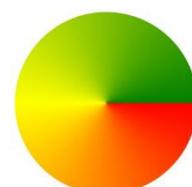
```
#grad1 {  
    height: 200px;  
    width: 200px;  
    background-image: conic-gradient(red 45deg, yellow 90deg, green 210deg);  
}
```



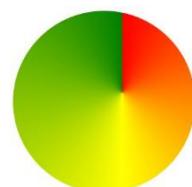
```
#grad1 {  
    height: 200px;  
    width: 200px;  
    background-image: conic-gradient(red, yellow, green, blue, black);  
    border-radius: 50%;  
}
```



```
#grad1 {  
    height: 200px;  
    width: 200px;  
    background-image: conic-gradient(from 90deg, red, yellow, green);  
    border-radius: 50%;  
}
```



```
#grad1 {  
    height: 200px;  
    width: 200px;  
    background-image: conic-gradient(at 60% 45%, red, yellow, green);  
    border-radius: 50%;  
}
```



5.2.1.4 Generadores de gradientes

Para facilitarnos la tarea de la **creación de gradientes** podemos utilizar un **generador de gradientes online** que nos proporcione el código CSS necesario para nuestro diseño. Algunas herramientas online útiles son las siguientes:

- cssgradient.io
- <https://www.colorzilla.com/es/>





5.3 Repetición y posición

En cuanto a las imágenes, CSS incorpora dos propiedades más relativas a la distribución de la imagen, ‘background-repeat’ y ‘background-position’. La primera permite indicar la **repetición** de una imagen en el fondo y la segunda, la **posición exacta** de la imagen.

5.3.1 background-repeat

La propiedad ‘background-repeat’ permite escoger la dirección de repetición de la imagen. En algunos casos es aconsejable que la imagen se repita en todas las direcciones; por ejemplo, si es demasiado pequeña. Para este fin, se utiliza el valor **repeat**. Si solo se desea que se repita en una dirección, bastará con añadir la dirección de repetición.

La **sintaxis** es la siguiente:

```
etiqueta html {  
    background-repeat: repeat | repeat-x | repeat-y | no-repeat | space  
    | round | initial | inherit ;  
}
```

- **repeat**. La imagen de fondo se repite tanto vertical como horizontalmente. La última imagen se recortará si no cabe. Esta es la opción por defecto.
- **repeat-x**. La imagen de fondo se repite sólo horizontalmente.
- **repeat-y**. La imagen de fondo se repite sólo verticalmente.
- **no-repeat**. La imagen de fondo no se repite. La imagen sólo se mostrará una vez.
- **space**. La imagen de fondo se repite tanto como sea posible sin recortarla. La primera y la última imagen se fijan a ambos lados del elemento, y el espacio en blanco se distribuye uniformemente entre las imágenes.
- **round**. La imagen de fondo se repite y se aplasta o estira para llenar el espacio (sin espacios).

A continuación, se muestran algunos ejemplos:

```
#example1 {  
    border: 2px solid black;  
    padding: 25px;  
    background: url("w3css.gif");  
    background-repeat: repeat;  
}
```





```
#example2 {  
    border: 2px solid black;  
    padding: 25px;  
    background: url("w3css.gif");  
    background-repeat: space;  
}
```



5.3.2 background-position

Además, CSS permite controlar la posición de la imagen dentro del fondo del elemento mediante la propiedad '**background-position**'. La sintaxis de esta propiedad incluye la distancia a cada extremo, teniendo en cuenta que **la posición original de toda imagen es la esquina superior izquierda**.

La **sintaxis** es la siguiente:

```
etiqueta html {  
    background-position: value | inherit | initial ;  
}
```

value puede tomar los siguientes valores:

- **left top | left center | left bottom | right top | right center | right bottom | center top | center center | center bottom.** Si solo se especifica una keyword, por ejemplo left, el otro valor será **center por defecto**.
- **x% y%.** El primer valor es la posición horizontal y el segundo la vertical. La esquina superior izquierda es 0% 0%. La esquina inferior derecha es 100% 100%. Si sólo especifica un valor, el otro valor será el 50%. El valor por defecto es 0% 0%
- **xpos ypos.** El primer valor es la posición horizontal y el segundo la vertical. La esquina superior izquierda es 0 0. Las unidades pueden ser píxeles (0px 0px) o cualquier otra unidad CSS. Si sólo especifica un valor, el otro valor será el 50%. Se pueden **mezclar % y posiciones**.

Existe otra propiedad relativa a las imágenes especificadas mediante 'background-imagen' que es '**background-attachment**'. Esta propiedad determina si la imagen estará fija dentro



de la pantalla (**fixed**) se desplazará junto al bloque contenedor (**scroll**) y se mostrará con una barra de scroll. Este es el valor por defecto.

A continuación, se muestran algunos **ejemplos**:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
    background-image: url('w3css.gif');
    background-repeat: no-repeat;
    background-attachment: fixed;
    background-position: 50px 150px;
}
</style>
</head>
<body>

<h1>The background-position Property</h1>
<p>Here, the background image is positioned 50px
from the left, and 150px down from the top.</p>

</body>
</html>
```

The background-position Property

Here, the background image is positioned 50px from the left, and 150px down from the top.



```
<!DOCTYPE html>
<html>
<head>
<style>
body {
    background-image: url('w3css.gif');
    background-repeat: no-repeat;
    background-attachment: fixed;
    background-position: right top;
}
</style>
</head>
<body>

<h1>The background-position
Property</h1>
<p>Here, the background image will be
positioned bottom right.</p>

</body>
</html>
```

The background-position Property

Here, the background image will be positioned bottom right.





5.4 background-size

La propiedad **background-size** especifica el tamaño de las imágenes de fondo. Para utilizar esta propiedad, debemos consultar su compatibilidad con los diferentes navegadores web y sus versiones, y la necesidad de utilizar prefijos (-webkit-, -moz-, etc).

Su sintaxis es la siguiente:

```
background-size: auto | length | cover | contain;
```

- **auto**. Es el valor por defecto. La imagen de fondo se muestra en su tamaño original.
- **length**. Establece la anchura (width) y la altura (height) de la imagen de fondo. El primer valor establece la anchura, el segundo valor establece la altura. **Si sólo se da un valor, el segundo se establece en "auto"**. Se puede usar unidades de medida o porcentajes.
- **cover**. **Redimensiona** la imagen de fondo para que cubra todo el contenedor, aunque tenga que estirar la imagen o cortar un poco uno de los bordes. Se suele utilizar para **un responsive design**.
- **contain**. **Redimensiona** la imagen de fondo para que la imagen sea totalmente visible.

A continuación, se presenta un [ejemplo](#):



```
<!DOCTYPE html>
<html>
<head>
<style>
#example1 {
    border: 2px solid black;
    padding: 25px;
    background: url(mountain.jpg);
    background-repeat: no-repeat;
    background-size: 100% 100%;
}

#example2 {
    border: 2px solid black;
    padding: 25px;
    background: url(mountain.jpg);
    background-repeat: no-repeat;
    background-size: 75% 50%;
}

#example3 {
    border: 2px solid black;
    padding: 25px;
    background: url(mountain.jpg);
    background-repeat: no-repeat;
    background-size: cover;
}

#example4 {
    border: 2px solid black;
    padding: 25px;
    background: url(mountain.jpg);
    background-repeat: no-repeat;
    background-size: contain;
}

</style>
</head>
<body>

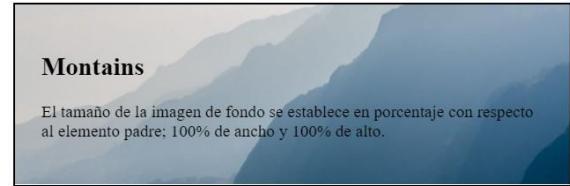
<h2>background-size: 100% 100%:</h2>
<div id="example1">
    <h2>Montains</h2>
    <p>El tamaño de la imagen de fondo se establece en porcentaje con respecto al elemento padre; 100% de ancho y 100% de alto.</p>
</div>

<h2>background-size: 75% 50%:</h2>
<div id="example2">
    <h2>Montains</h2>
    <p>El tamaño de la imagen de fondo se establece en porcentaje con respecto al elemento padre; 75% width and 50% height.</p>
</div>

<h2>background-size: cover:</h2>
<div id="example3">
    <h2>Montains</h2>
    <p>La imagen de fondo se redimensiona para cubrir todo el contenedor, aunque tenga que estirar la imagen o cortar un poco uno de los bordes.</p>
</div>

<h2>background-size: contain:</h2>
<div id="example4">
    <h2>Montains</h2>
    <p>La imagen se redimensiona para que sea completamente visible.</p>
</div>

</body>
</html>
```

background-size: 100% 100%:**background-size: 75% 50%:****background-size: cover:****background-size: contain:**



5.5 Opacity

El estándar de CSS 2.1 no incluye ninguna propiedad para controlar la opacidad de los elementos de la página. Sin embargo, la versión CSS3 incluye una propiedad llamada **opacity**, que permite incluir transparencias en el diseño de la página.

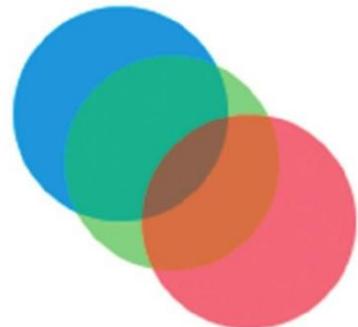
La sintaxis es la siguiente:

```
etiqueta html {  
    opacity: valor;  
}
```

El valor de la propiedad '**opacity**' se define mediante un número decimal, cuyo valor puede estar comprendido **entre 0.0 y 1.0**, siendo 0.0 la máxima transparencia, es decir, el elemento es invisible; y 1.0 la máxima opacidad, esto es, el elemento es completamente visible.

En el siguiente **ejemplo** se establece el valor de transparencia al 50%, para los elementos verde y rojo.

```
#verde, #rojo {  
    opacity: 0.5;  
}  
  
#azul {  
    background-color: blue;  
}  
  
#rojo {  
    background-color: red;  
}  
  
#verde {  
    background-color: green;  
}
```





6. Propiedades de texto

Se denominan propiedades de texto aquellas que **afectan al texto en su conjunto como bloque**. Estas propiedades permiten controlar la alineación del texto, el interlineado, la separación entre palabras, etc.

En la siguiente tabla se muestran **las propiedades de texto** más utilizadas y que vamos a ver en detalle.

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

6.1 Interlineado de texto ('line-height')

Esta propiedad indica el tamaño del espacio entre las líneas. Suele encontrarse en forma de **porcentaje**, siendo 100% el interlineado normal. El valor puede ser expresado en porcentaje o en unidades de medida. El uso de un buen interlineado facilita la lectura del texto del sitio web notablemente.

La **sintaxis** es la siguiente:

```
etiqueta html {  
    line-height: valor;  
}
```



Ejemplo:

```
p {  
    line-height: 120%;  
}  
  
p {  
    line-height: 1.2em;  
}
```

Además de las unidades de medida y los porcentajes, la propiedad ‘`line-height`’ puede indicarse por un número sin unidades, que se interpreta como un múltiplo del tamaño normal de la letra.

6.2 Alineado de texto (‘text-align’)

Esta propiedad establece la alineación del texto. Los valores definidos para alinear el texto son a la izquierda (`left`), a la derecha (`right`), centrado (`center`) y justificado (`justify`). Si no se indica ningún valor, se utiliza **left por defecto**.

La sintaxis es la siguiente:

```
etiqueta html {  
    text-align: left | right | center | justify | initial;  
}
```

6.3 Decoración de texto (‘text-decoration’)

La propiedad ‘`text-decoration`’ añade al texto distintos elementos conocidos: tachado, subrayado, etc. Existe la posibilidad de poner **varios valores al mismo tiempo** en esta propiedad.

El **valor por defecto** es `none`, y, aunque cabe esperar que no se vaya a utilizar nunca, puesto que si no deseamos ninguna decoración basta con indicarlo, existen algunos casos, como **los enlaces** en los que aparecen subrayados por defecto; si se deseara eliminar este efecto, tendría que **utilizarse `none`**.



La **sintaxis** es la siguiente (todas las propiedades en la misma regla):

```
etiqueta html {  
    text-decoration: text-decoration-line text-decoration-color text-  
    decoration-style text-decoration-thickness | initial | inherit;  
}  
  
L → {  
    text-decoration-line: none | underline | overline | line-through | blink | initial;  
    text-decoration-color: color | rgb() | hsl () | initial;  
    text-decoration-style: solid | double | dotted | dashed | wavy | initial;  
    text-decoration-thickness: auto | from-font | length/percentage |initial |inherit;
```

Como se puede ver en la sintaxis, **esta propiedad está formada por otros tipos**:

- 1) En primer lugar, ‘text-decoration-line’ indica el lugar exacto en el que se va a añadir la línea sobre el texto (superior, inferior o media).
- 2) En segundo lugar, ‘text-decoration-color’, el cual define el color.
- 3) La propiedad que define el estilo de dicha línea, ‘text-decoration-style’.
- 4) Por último, ‘text-decoration-thickness’ indica el grosor de la línea.

text-decoration-line		
Underline	Subraya el texto con una línea en su parte inferior.	Ejemplo
Overline	Incorpora una línea en la parte superior del texto.	Ejemplo
Line-Through	Incorpora una línea en la parte media del texto, lo tacha.	Ejemplo
text-decoration-color		
Define el color usando una palabra clave o codificación en hexadecimal	div { text-decoration-color: green; } div { text-decoration-color: #008000; }	
Define el color usando el sistema RGB	div { text-decoration-color: rgb(255,0,0); }	
Define el color usando el sistema HSL	div { text-decoration-color: hsl(120,100%,50%); }	



text-decoration-style

<i>solid</i>	Valor por defecto. Muestra la línea sin ningún estilo adicional.	<hr/>
<i>double</i>	Muestra dos líneas.	<hr/>
<i>dotted</i>	Muestra una línea punteada.	<hr/>
<i>dashed</i>	Muestra una línea discontinua.	<hr/>

Ejemplo 1:

```
<!DOCTYPE html>
<html>
<head>
<style>
    h1 {
        text-decoration: underline;
        text-decoration-thickness: auto;
    }

    h2 {
        text-decoration: underline;
        text-decoration-thickness: 5px;
    }

    h3 {
        text-decoration: underline;
        text-decoration-thickness: 50%;
    }

    /* Use shorthand property */
    h4 {
        text-decoration: underline solid red 50%;
    }
</style>
</head>
<body>

<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<h3>This is heading 3</h3>
<h4>This is heading 4</h4>

</body>
</html>
```

This is heading 1

This is heading 2

This is heading 3

This is heading 4



Ejemplo 2:

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
    text-decoration: underline overline dotted red;
}

h2 {
    text-decoration: underline wavy blue 5px;
}
</style>
</head>
<body>

<h1>This is heading 1</h1>
<h2>This is heading 2</h2>

</body>
</html>
```

This is heading 1

This is heading 2

6.4 Alineación vertical ('vertical-align')

La propiedad **vertical-align** puede utilizarse en **dos contextos**:

- 1) Para alinear verticalmente la caja de un elemento de **nivel inline dentro de su caja de línea contenedora**. Por ejemplo, puede utilizarse para posicionar verticalmente una imagen en una línea de texto.
- 2) Para **alinear verticalmente el contenido de una celda en una tabla**.

Su **sintaxis** es la siguiente:

```
etiqueta html {

    vertical-align: baseline | Length | sub | super | top | text-top
    | middle | bottom | text-bottom | initial | inherit;
}
```

La alineación vertical **sólo se aplica a los elementos inline, inline-block y table-cell**:



Se puede encontrar más información en: https://www.w3schools.com/cssref/pr_pos_vertical-align.php

Ejemplo de alineación vertical de una imagen en una línea de texto:

```
<!DOCTYPE html>
<html>
<head>
<style>
img.a {
    vertical-align: baseline;
}

img.b {
    vertical-align: text-top;
}

img.c {
    vertical-align: text-bottom;
}

img.d {
    vertical-align: sub;
}

img.e {
    vertical-align: super;
}
</style>
</head>
<body>

<h1>The vertical-align Property</h1>

<h2>vertical-align: baseline (default):</h2>
<p>An 
image with a default alignment.</p>

<h2>vertical-align: text-top:</h2>
<p>An 
image with a text-top alignment.</p>

<h2>vertical-align: text-bottom:</h2>
<p>An 
image with a text-bottom alignment.</p>

<h2>vertical-align: sub:</h2>
<p>An 
image with a sub alignment.</p>

<h2>vertical-align: sup:</h2>
<p>An 
image with a super alignment.</p>

</body>
</html>
```

The vertical-align Property

vertical-align: baseline (default):

An image with a default alignment.

vertical-align: text-top:

An image with a text-top alignment.

vertical-align: text-bottom:

An image with a text-bottom alignment.

vertical-align: sub:

An image with a sub alignment.

vertical-align: sup:

An image with a super alignment.



Ejemplo de alineación vertical del contenido de una celda dentro de una tabla:

```
<!DOCTYPE html>
<head>
  <style>
    td{height:100px; border: 1px solid grey;}
    .a { vertical-align: top; }
    .b { vertical-align: middle; }
    .c { vertical-align: bottom; }
  </style>
</head>

<body>
  <h3>Propiedad vertical-align</h3>
  <table>
    <tr>
      <td class="a">vertical-align: top</td>
      <td class="b">vertical-align: middle</td>
      <td class="c">vertical-align: bottom</td>
    </tr>
  </table>
</body>
</html>
```

Propiedad vertical-align

vertical-align: top	vertical-align: middle	vertical-align: bottom
---------------------	------------------------	------------------------

6.5 Transformación del texto (“text-transform”)

La propiedad CSS “text-transform” controla las mayúsculas del texto. Su **sintaxis** es la siguiente:

```
etiqueta html {
  text-transform: none | capitalize | uppercase | lowercase;
}
```

- **capitalize**: convierte la primera letra de cada palabra en mayúscula.
- **uppercase**: convierte todo el texto en mayúscula.
- **lowercase**: convierte todo el texto en minúscula.



Ejemplo:

```
<!DOCTYPE html>
<html>
<head>
<style>
    div.a {
        text-transform: uppercase;
    }
    div.b {
        text-transform: lowercase;
    }
    div.c {
        text-transform: capitalize;
    }
</style>
</head>
<body>
    <h1>The text-transform Property</h1>

    <h2>text-transform: uppercase:</h2>
    <div class="a">Lorem ipsum dolor sit amet, consectetur adipiscing elit.</div>

    <h2>text-transform: lowercase:</h2>
    <div class="b">Lorem ipsum dolor sit amet, consectetur adipiscing elit.</div>

    <h2>text-transform: capitalize:</h2>
    <div class="c">Lorem ipsum dolor sit amet, consectetur adipiscing elit.</div>
</body>
</html>
```

The text-transform Property

text-transform: uppercase:

LOREM IPSUM DOLOR SIT AMET, CONSECTETUR ADIPISCING ELIT.

text-transform: lowercase:

lorem ipsum dolor sit amet, consectetur adipiscing elit.

text-transform: capitalize:

Lorem Ipsum Dolor Sit Amet, Consectetur Adipiscing Elit.

6.6 Letter spacing

Esta propiedad CSS aumenta o disminuye el **espacio existente entre los caracteres** de un texto, es decir, especifica el espacio que hay entre las letras de un texto. Puede tomar valores negativos.

Su sintaxis es la siguiente:

```
etiqueta html {
    letter-spacing: normal | length;
}
```

Ejemplo:

```
<!DOCTYPE html>
<html>
<head>
<style>
    h1 {
        letter-spacing: 3px;
    }

    h2 {
        letter-spacing: -1px;
    }

    h3 {
        letter-spacing: normal;
    }
</style>
</head>
<body>

    <h1>Espaciado entre letras de 3px</h1>
    <h2>Espaciado entre letras de 2px</h2>
    <h3>Espaciado entre letras normal</h3>

</body>
</html>
```

Espaciado entre letras de 3px

Espaciado entre letras de 2px

Espaciado entre letras normal



6.7 Word spacing

Esta propiedad CSS aumenta o disminuye el **espacio en blanco entre las palabras**, es decir, especifica el espacio que hay entre palabras.

Su sintaxis es la siguiente:

```
etiqueta html {  
    word-spacing: normal | length;  
}
```

- **normal**: es el valor por defecto y equivale a un espaciado entre palabras de 0.25 em.
- **length**: define el espacio entre palabras en px, pt, cm, em, etc. No admite valores negativos.

Ejemplo:

```
<!DOCTYPE html>  
<html>  
<head>  
<style>  
    p.a {  
        word-spacing: normal;  
    }  
    p.b {  
        word-spacing: 30px;  
    }  
    p.c {  
        word-spacing: 1cm;  
    }  
</style>  
</head>  
<body>  
    <h1>The word-spacing Property</h1>  
  
    <h2>word-spacing: normal:</h2>  
    <p class="a">This is some text. This is some text.</p>  
  
    <h2>word-spacing: 30px:</h2>  
    <p class="b">This is some text. This is some text.</p>  
  
    <h2>word-spacing: 1cm:</h2>  
    <p class="c">This is some text. This is some text.</p>  
</body>  
</html>
```

The word-spacing Property

word-spacing: normal:

This is some text. This is some text.

word-spacing: 30px:

This is some text. This is some text.

word-spacing: 1cm:

This is some text. This is some text.



6.8 Tabulado del texto ('text-indent')

La propiedad '**text-indent**' permite tabular la primera línea de cada párrafo creando una sangría de primera línea. El valor puede expresarse en porcentaje o en unidades métricas.

Su **sintaxis** es la siguiente:

```
etiqueta html {  
    text-indent: <medida> | <porcentaje> ;  
}
```

Ejemplo:

```
<!DOCTYPE html>  
<html>  
<head>  
<style>  
    div.a {  
        text-indent: 50px;  
    }  
    div.b {  
        text-indent: -2em;  
    }  
    div.c {  
        text-indent: 30%;  
    }  
</style>  
</head>  
<body>  
    <h1>The text-indent Property</h1>  
    <h2>text-indent: 50px;</h2>  
    <div class="a">  
        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.  
Etiam semper diam at erat pulvinar, at pulvinar felis blandit.</p>  
    </div>  
  
    <h2>text-indent: -2em;</h2>  
    <div class="b">  
        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.  
Etiam semper diam at erat pulvinar, at pulvinar felis blandit.</p>  
    </div>  
  
    <h2>text-indent: 30%:</h2>  
    <div class="c">  
        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.  
Etiam semper diam at erat pulvinar, at pulvinar felis blandit.</p>  
    </div>  
</body>  
</html>
```

The text-indent Property

text-indent: 50px:

Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut.

text-indent: -2em:

Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut.

text-indent: 30%:

Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut.

6.9 Sombra del texto ('text-shadow')

La propiedad **text-shadow** añade **sombra** al texto.

Su **sintaxis** es la siguiente:

```
etiqueta html {  
    text-shadow: h-shadow v-shadow blur-radius color|none;  
}
```



- ***h-shadow***. Requerido. Es la posición de la sombra horizontal. Se permiten valores negativos.
- ***v-shadow***. Requerido. Es la posición de la sombra vertical. Se permiten valores negativos.
- ***blur-radius***. Opcional. Es el radio de desenfoque o difuminado. El valor por defecto es 0.
- ***color***. Opcional. Es el color de la sombra.
- ***none***. Es el valor por defecto. Sin sombra.

IMPORTANTE:

- Para aplicar sombras **arriba y a la izquierda** se deben utilizar **valores negativos**.
- Si queremos usar **múltiples sombras** al mismo elemento, basta con **separar cada sombra con una coma (,)** y respetar la estructura.
- Para este tipo de propiedades debemos incluir los **prefijos para navegadores**.



A continuación, se muestra un ejemplo:

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
    text-shadow: 2px 2px red, 4px 4px #FF0000, 8px
8px yellow;
}
</style>
</head>
<body>

<h1>The text-shadow Property</h1>

</body>
</html>
```

The text-shadow Property



6.10 Sombra a elementos caja ('box-shadow')

La propiedad **box-shadow** añade **sombra** a los elementos caja.

Su **sintaxis** es la siguiente:

```
etiqueta html {  
  box-shadow: none | h-offset v-offset blur spread color inset;  
}
```

- **h-offset**. Requerido. Es la posición de la sombra horizontal. Se permiten valores negativos.
- **v-offset**. Requerido. Es la posición de la sombra vertical. Se permiten valores negativos.
- **blur**. Opcional. Es el radio de desenfoque o difuminado.
- **spread**. Opcional. Es el tamaño de la sombra hacia los lados.
- **color**. Opcional. Es el color de la sombra.
- **inset**. Opcional. Posiciona la sombra dentro del marco.

A continuación, se puede ver un **ejemplo**.

```
<!DOCTYPE html>  
<html>  
<head>  
<style>  
#example1 {  
  border: 1px solid;  
  padding: 10px;  
  box-shadow: 5px 5px blue, 10px 10px red, 15px 15px green;  
  margin: 20px;  
}  
  
.example2 {  
  border: 1px solid;  
  padding: 10px;  
  box-shadow: 5px 5px 8px blue, 10px 10px 8px red, 15px 15px 8px green;  
  margin: 20px;  
}  
</style>  
</head>  
<body>  


# Multiple Shadows

  
<div id="example1">  
  <h2>Multiple shadows</h2>  
  <p>box-shadow: 5px 5px blue, 10px 10px red, 15px 15px green:</p>  
</div>  
<br>  
  <h2>Multiple shadows with blur effect</h2>  
  <p>box-shadow: 5px 5px 8px blue, 10px 10px 8px red, 15px 15px 8px green:</p>  
</div>  
</body>  
</html>
```

Multiple Shadows

Multiple shadows

box-shadow: 5px 5px blue, 10px 10px red, 15px 15px green;

Multiple shadows with blur effect

box-shadow: 5px 5px 8px blue, 10px 10px 8px red, 15px 15px 8px green;



6.10.1 Generadores de sombra online

Para facilitarnos la tarea de la creación de sombras podemos utilizar un generador de sombras tanto para los textos como para las cajas o contenedores. Algunas herramientas online útiles son las siguientes:

- **Generador de sombras en textos:**
 - <https://html-css-js.com/css/generator/text-shadow/>
 - <https://cssgenerator.org/text-shadow-css-generator.html>
- **Generador de sombras en cajas o contenedores:**
 - <https://html-css-js.com/css/generator/box-shadow/>
 - <https://cssgenerator.org/box-shadow-css-generator.html>



7. Propiedades de Fuente

La **elección de una fuente adecuada** para una página web es importante, así como las **propiedades** relacionadas con esta, puesto que el conjunto final permite a los usuarios interpretar de una manera u otra el contenido del sitio web, por ejemplo, marcar los elementos más importantes y que mayor grado e información aportan al usuario final.

En la siguiente tabla se muestran **las propiedades de las fuentes** más destacadas.

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

7.1 Fuente de letra ('font-family')

Esta propiedad establece la **fuente de la letra utilizada** en el documento estructurado HTML. Como se vio en la primera unidad, existen algunas tipografías cuyo uso es más recomendable, puesto que se trata de fuentes genéricas disponibles en la mayoría de los navegadores; estas son *Serif*, *Sans-Serif*, *Cursive*, *Monospace* o *Fantasy*. Se podría usar cualquier otra fuente, pero debe tenerse en cuenta que, si el navegador no la reconoce, utilizará otra distinta. Para evitar resultados indeseados se recomienda añadir varios tipos de fuentes.

Su **sintaxis** es la siguiente:

```
etiqueta html {  
    font-family: (nombre_familia | familia_genérica),  
                (nombre_familia | familia_genérica)*;  
}
```



Si el navegador no dispone de la primera tipografía indicada, irá probando con el resto hasta encontrar un tipo válido. Por esta razón, se aconseja añadir siempre como última opción una de las fuentes genéricas nombradas, puesto que estas están disponibles en todos los navegadores.

Debe tenerse en cuenta que, **si el nombre de la familia está formado por más de una palabra, se escribe entre comillas**, por ejemplo, “Times New Roman”. Por el contrario, si utilizamos tipografías tales como Courier, no será necesario entrecomillarlo.

Ejemplo:

```
p {  
    font-family: serif;  
}  
  
p {  
    font-family: Arial, Helvetica, sans-serif;  
}  
  
p {  
    font-family: "Times New Roman", Palatino, times, serif;  
}
```

7.2 Tamaño de letra ('font-size')

La propiedad 'font-size' permite establecer el **tamaño del tipo de letra** del elemento HTML. Se puede expresar el tamaño de varias maneras: tamaño absoluto, tamaño relativo, distancia o porcentaje; generalmente, se aconseja utilizar unidades relativas.

La **sintaxis** es la siguiente:

```
etiqueta html {  
    font-size: valor;  
}
```



- Tamaño **absoluto**:

```
etiqueta html {  
    font-size: xx-small | x-small | small | medium | large |  
              x-large | xx-large;  
}
```

- Tamaño **relativo**:

```
etiqueta html {  
    font-size: smaller | larger;  
}
```

- **Numérico:**

```
etiqueta html {  
    font-size: <longitud> | <porcentaje>;  
}
```

Atributos de tamaño absoluto y relativo

Tamaño absoluto

Dimensiones absolutas. Son las indicadas por el diseñador y no están en función del resto de elementos, como ocurre con las dimensiones relativas. Se aplica *medium* por defecto. (14 px.)

- *xx-small*
- *x-small*
- *small*
- *medium*
- *large*
- *x-large*
- *xx-large*

```
p {  
    font-size:  
          x-large;  
}
```

Tamaño relativo

Estos tamaños se toman con respecto al elemento "padre", y toman el tamaño siguiente o anterior en la lista de tamaños absolutos visto en la fila anterior.

- *larger*
- *smaller*

```
ul {  
    font-size:  
          smaller;  
}
```

Valor numérico con unidades

Se indica una unidad positiva de longitud (pt,px).

```
p {  
    font-size: 16pt;  
}
```

Valor numérico con porcentajes

Se indica un porcentaje del tamaño con respecto al elemento "padre".

```
p {  
    font-size: 80%;  
}
```

A continuación, se muestra un **ejemplo**:



```
<!DOCTYPE html>
<html>
<head>
<style>
    div.a {
        font-size: 15px;
    }
    div.b {
        font-size: large;
    }
    div.c {
        font-size: 150%;
    }
    p.d {
        font-size: larger;
    }
</style>
</head>
<body>
    <h1>The font-size Property</h1>

    <div class="a">Es un texto con tamaño 15 px.</div>
    <div class="b">Es un texto con tamaño large.</div>
    <div class="c">Es un texto con tamaño 150%.
        <p class="d"> Parrafo con letra mayor que su
        parente </p>
    </div>
</body>
</html>
```

The font-size Property

Es un texto con tamaño 15 px.

Es un texto con tamaño large.

Es un texto con tamaño 150%.

Parrafo con letra mayor que su parente

7.3 Estilo de letra ('font-style')

Esta tercera propiedad relativa a la fuente de la letra se encarga de **modificar la inclinación** de las palabras. En concreto, pueden encontrarse dos tipos de letra inclinada: por un lado, la **italica** (más conocida como cursiva), y por otro, las **oblicuas**.

La **sintaxis** es la siguiente:

```
etiqueta html {
    font-style: normal | italic | oblique;
}
```



Resumen de estilos de texto

Normal	Normal	p { font-family: sans-serif; font-size: 200%; font-style: normal; }
Cursiva	Italic	p { font-family: sans-serif; font-size: 50%; font-style: italic; }
Oblicua	Oblique	p { font-family: sans-serif; font-size: 10%; font-style: oblique; }

Ejemplo:

```
<!DOCTYPE html>
<html>
<head>
<style>
    p.a {
        font-style: normal;
    }

    p.b {
        font-style: italic;
    }

    p.c {
        font-style: oblique;
    }
</style>
</head>
<body>
    <h1>The font-style Property</h1>

    <p class="a">This is a paragraph, normal.</p>
    <p class="b">This is a paragraph, italic.</p>
    <p class="c">This is a paragraph, oblique.</p>
</body>
</html>
```

The font-style Property

This is a paragraph, normal.

This is a paragraph, italic.

This is a paragraph, oblique.

7.4 Grosor de letra ('font-weight')

Esta propiedad modifica el **grosor** de las letras. La aplicación de esta propiedad se basa en la utilización de un conjunto de valores numéricos ya determinados, desde la opción más estrecha hasta la más gruesa (100, 200, 300, 400, 500, 600, 700, 800, y 900). Ahora bien,



también es posible utilizar las palabras *normal* y *bold*, correspondientes al valor **400**, el grosor usual utilizado en los trazos, y **700**, respectivamente.

La sintaxis es la siguiente:

```
etiqueta html {  
    font-weight: valor;  
}
```

Al igual que ocurre con la propiedad ‘font-size’, esta propiedad incorpora dos valores que permiten reducir o aumentar el grosor a un valor inferior o superior, respectivamente, de la lista de valores posibles. Estos son: *lighter* y *bolder*.

En función del navegador y del tipo de fuente escogido, la diferencia entre grosores es más o menos perceptible.

Resumen grosor de letra ('font-weight')

Normal	Equivale a un grosor de 400.
Bold	Equivale a un grosor de 700.
Lighter	Disminuye al valor inmediatamente inferior al del elemento “padre”.
Bolder	Aumenta al valor inmediatamente superior al del elemento “padre”.

Ejemplo:

```
<!DOCTYPE html>  
<html>  
<head>  
<style>  
    p.normal {  
        font-weight: normal;  
    }  
    p.thick {  
        font-weight: bold;  
    }  
    p.lighter {  
        font-weight: lighter;  
    }  
    p.thick2 {  
        font-weight: 900;  
    }  
</style>  
</head>  
<body>  
    <h1>The font-weight Property</h1>  
    <p class="normal">Parrafo con grosor normal.</p>  
    <p class="thick2">Parrafo con grosor de 900.</p>  
    <p class="thick">Parrafo con grosor bold.  
        <p class="lighter">Parrafo con un grosor mas fino que su padre p</p>  
    </p>  
</body>  
</html>
```

The font-weight Property

Parrafo con grosor normal.

Parrafo con grosor de 900.

Parrafo con grosor bold.

Parrafo con un grosor mas fino que su padre p



7.5 Variante de fuente ('font-variant')

La propiedad **font-variant** especifica si un texto debe mostrarse o no en minúsculas, de manera que **convierte a mayúsculas** manteniendo todas las letras en un tamaño inferior a la primera.

Su **sintaxis** es la siguiente:

```
etiqueta html {  
    font-variant: normal|small-caps|initial|inherit;  
}
```

A continuación, se muestra un **ejemplo**:

```
<!DOCTYPE html>  
<html>  
<head>  
<style>  
p.normal {  
    font-variant: normal;  
}  
  
p.small {  
    font-variant: small-caps;  
}  
</style>  
</head>  
<body>  
<h1>The font-variant Property</h1>  
  
<p class="normal">My name is Hege Refsnes.</p>  
<p class="small">My name is Hege Refsnes.</p>  
  
</body>  
</html>
```

The font-variant Property

My name is Hege Refsnes.

MY NAME IS HEGE REFSNES.

7.6 La propiedad fundamental 'font'

La propiedad principal para definir el tipo de fuente, tamaño y posición, entre otras es la propiedad **'font'**, que se considera una propiedad combinada, puesto que permite definir las siguientes propiedades individuales: **'font-style'**, **'font-variant'**, **'font-weight'**, **'font-size'**, **'line-height'** y **'font-family'**.

La **sintaxis** es la siguiente:

```
etiqueta html {  
    font : font-style, font-variant, font-weight, font-size/line-  
          height, font-family;  
}
```



Es muy importante que **las propiedades aparezcan en el orden indicado**. Si no se hace así, los navegadores no las interpretarán y no darán el formato indicado.

- Siempre deben aparecer las propiedades ‘font-size’ y ‘font-family’ al utilizar la propiedad ‘font’.
- Si se escribe la propiedad ‘line-height’ debe aparecer separada de ‘font-size’ por una barra inclinada (/).

Ejemplo:

```
<!DOCTYPE html>
<html>
<head>
<style>
  p.a {
    font: 15px Arial, sans-serif;
  }

  p.b {
    font: italic small-caps bold 12px/30px Georgia, serif;
  }
</style>
</head>
<body>

  <h1>The font Property</h1>

  <p class="a">This is a paragraph. The font size is set to 15 pixels, and the font family is Arial.</p>

  <p class="b">This is a paragraph. The font is set to italic and bold, with small-caps (all lowercase letters are converted to uppercase). The font size is set to 12 pixels, the line height is set to 30 pixels, and the font family is Georgia.</p>

</body>
</html>
```

The font Property

This is a paragraph. The font size is set to 15 pixels, and the font family is Arial.

THIS IS A PARAGRAPH. THE FONT IS SET TO ITALIC AND BOLD, WITH SMALL-CAPS (ALL LOWERCASE LETTERS ARE CONVERTED TO UPPERCASE). THE FONT SIZE IS SET TO 12 PIXELS, THE LINE HEIGHT IS SET TO 30 PIXELS, AND THE FONT FAMILY IS GEORGIA.

Las propiedades ‘font’ admite los valores `caption`, `icon`, `menu`, `message-box`, `small-caption` y `status-bar`, que hacen referencia a los tipos de letra que utiliza el sistema operativo para esos elementos.

Palabra clave	Elementos del Sistema Operativo a los que hace referencia
<code>caption</code>	Controles y campos de formulario (botones, listas desplegables, etc)
<code>icon</code>	Texto mostrado debajo de cada ícono
<code>menu</code>	Menús desplegables y otros tipos de menús
<code>message-box</code>	Cuadros de diálogo (ventanas con mensajes de error, información, advertencia, etc.)
<code>small-caption</code>	Controles y campos de formulario más pequeños
<code>status-bar</code>	Barra de estado de las ventanas



Ejemplo:

```
<!DOCTYPE html>
<html>
<body>

<h1>The font Property</h1>

<p style="font:caption">The font used in captioned controls.</p>
<p style="font:icon">The font used in icon labels.</p>
<p style="font:menu">The font used in dropdown menus.</p>
<p style="font:message-box">The font used in dialog boxes.</p>
<p style="font:small-caption">A smaller version of the caption font.</p>
<p style="font:status-bar">The font used in the status bar.</p>
<p><b>Note:</b> The result of the font keywords is browser dependant.
</p>

</body>
</html>
```

The font Property

The font used in captioned controls.

The font used in icon labels.

The font used in dropdown menus.

The font used in dialog boxes.

A smaller version of the caption font.

The font used in the status bar.

Note: The result of the font keywords is browser dependant.

7.7 Fuentes externas

Utilizar las **fuentes del sistema** tiene la **desventaja** de que dichas fuentes dependen del navegador web y del sistema operativo del usuario. Por ejemplo, las fuentes que están instaladas en sistemas operativos Windows, como pueden ser la *Times new Roman*, *Tahoma*, *Arial*, etc, se ven correctamente en navegadores con dicho sistema operativo, pero no ocurre lo mismo en dispositivos con Linux o Mac.

Es por ello que si queremos asegurarnos de que todos los usuarios verán la misma tipografía en nuestra página web independientemente del dispositivo que utilicen, o queremos usar una tipografía especial, debemos utilizar fuentes externas.

Un catálogo de fuentes muy utilizado es el de [Google Fonts](#). Es bastante cómodo, ya que **Google** nos proporciona un código por cada fuente que tan solo tenemos que copiar y pegar en nuestro proyecto para disponer de la fuente que deseemos.

Tenemos **dos formas** de agregar las fuentes externas a nuestro proyecto:

- 2) A **nivel interno**, descargando la fuente, y mediante el uso de **@font-face**.

Una buena forma de trabajar con las fuentes en local es crear un directorio llamado **fons**, en el mismo directorio donde tengamos nuestra carpeta **css**, en donde se alojarán los archivos de las fuentes que se vayan a utilizar en el proyecto.

A continuación, se muestra cómo **descargar** las fuentes desde Google Fonts y cómo obtener el **código** por cada fuente:



The screenshot shows the Google Fonts interface. A user has selected the 'Robo' font family. Three specific styles are highlighted with red boxes and arrows pointing to numbered callouts:

- 1**: Buscamos y seleccionamos la fuente que queremos.
- 2**: Descargamos la fuente y vemos el código.

The 'Selected families' sidebar on the right lists 'Open Sans' and 'Robo'. Below it, under 'Use on the web', there is a code snippet for embedding the font:

```
<link rel="preconnect" href="https://fonts.googleapis.com">
```

At the bottom right of the sidebar, there are 'API docs' and 'Download all' buttons.

Para utilizar `@font-face` correctamente debemos definir un nombre para la fuente (por ejemplo, Roboto), y a continuación indicar dónde se encuentra ésta.

Ejemplo:

- 1) En primer lugar, incluimos la fuente Roboto-Regular que se encuentra en la carpeta /fonts.

```
@font-face {  
    font-family: 'Roboto';  
    src: url('../fonts/Roboto/Roboto-Regular.ttf');  
}
```

- 2) Luego, podemos declarar una clase en nuestra hoja de estilos como la que sigue para utilizarla:

```
.roboto {  
    font-family: 'Roboto', sans-serif;  
}
```

- 3) A nivel externo, cuando no queremos tener una fuente en local, por ejemplo, por cuestiones de copyright. Esta opción es mucho más sencilla y se puede utilizar si no tenemos reparos para usar recursos externos en nuestra página web. Se puede hacer de dos maneras:



- Incluir fuente con `<link>` a través del HTML. El código nos lo proporciona Google Fonts.

Ejemplo: Si queremos utilizar la fuente Roboto de Google Fonts, debemos seguir los siguientes pasos:

- 1) Agregamos el siguiente código que nos proporciona Google Fonts, en la cabecera o `<head>` de nuestra hoja HTML.

```
<link rel="preload" href="https://fonts.googleapis.com">
<link rel="preload" href="https://fonts.gstatic.com" crossorigin>
<link href="https://fonts.googleapis.com/css2?family=Roboto:ital,wght@1,300&display=swap" rel="stylesheet">
```

- 2) Luego, podemos declarar una clase en nuestra hoja de estilos como la que sigue para utilizarla (según nos indique Google):

```
.roboto {
    font-family: 'Roboto', sans-serif;
}
```

- Incluir fuente con `@import` o a través del CSS. El código nos lo proporciona Google Fonts. El proceso es similar al anterior.

Ejemplo: Si queremos utilizar la fuente Roboto de Google Fonts, debemos seguir los siguientes pasos:

- 1) Agregamos en nuestra hoja de estilos el código que nos indique Google, al principio del documento:

```
@import url('https://fonts.googleapis.com/css2?family=Roboto:ital,wght@1,300&display=swap');
```

- 2) Luego, podemos declarar una clase en nuestra hoja de estilos como la que sigue para utilizarla (según nos indique Google):

```
.roboto {
    font-family: 'Roboto', sans-serif;
}
```



7.7.1 Fuentes web online

Nombre	URL
Font Squirrel	fontsquirrel.com
Da Font	dafont.com
Google Fonts	google.com/fonts

7.7.2 Convertidores de fuentes online

Nombre	URL
Font Squirrel converter	fontsquirrel.com/tools/webfont-generator
Online Font Converter	onlinefontconverter.com
Files conversion	files-conversion.com/font-converter.php
Font converter	fontconverter.org



8. Display

La propiedad **display** permite especificar si un elemento se muestra o no y en caso de hacerlo, cómo se mostrará.

Todo elemento HTML tiene un valor de **display por defecto** que, dependiendo del elemento de qué elemento se trate, este valor podrá ser **block** o **inline**.

- Los elementos de **bloque** son los que al renderizarse lo hacen en una **nueva línea** y utilizan todo el ancho disponible como, por ejemplo:
 - `<div>`
 - `<h1> .. <h6>`
 - `<p>`
 - `<form>`
 - `<header>`
 - `<section>`
- Los **elementos de línea** no comienzan una nueva línea y solo utilizan el espacio necesario cuando se crean como, por ejemplo:
 - ``
 - `<a>`
 - ``

Para **saltarse el display por defecto** solo hay que especificar en la propiedad **display** cómo queremos que se muestre el elemento en cuestión. Algunos de los valores que puede tomar dicha propiedad son:

- **inline**: por defecto para ``. Se comporta como un solo bloque de línea. No permite modificar ancho y alto. **No se le puede aplicar las propiedades width, height ni margin**, ya que aunque tengamos mucho texto, se considera como una sola línea.
- **block**: permite modificar ancho y alto, pero añade salto de línea (líneas independientes). **Se aceptan las propiedades width, height y márgenes verticales**.



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

Es precisamente una de las características del **responsive design**, ya que en este caso lo que queremos es que los elementos se pongan unos al lado o debajo de los otros según el tamaño de la pantalla.

- **none**: permite **ocultar** un elemento y **no muestra el espacio reservado**. Este valor también se aplica al **responsive design**, ya que puede que nos interese ocultar algunos elementos si el tamaño de la pantalla es pequeño.

No confundir con *visibility:hidden*, el cual también oculta al elemento, pero sigue ocupando espacio en la página.



A continuación, se presentan algunos **ejemplos**:

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
    border: 2px solid black;
    background: orange;
    display: block;
}
</style>
</head>

<body>
<h2> Display: block </h2>
<p> Mensaje de texto </p>
</body>
</html>
```

Display: block

Mensaje de texto
Mensaje de texto

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
    border: 2px solid black;
    background: orange;
    display: inline;
}
</style>
</head>

<body>
<h2> Display: inline </h2>
<p> Mensaje de texto </p>
</body>
</html>
```

Display: inline

Mensaje de texto Mensaje de texto Mensaje de texto
Mensaje de texto Mensaje de texto Mensaje de texto

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
    border: 2px solid black;
    background: orange;
    display: inline-block;
}
</style>
</head>

<body>
<h2> Display: inline-block </h2>
<p> Mensaje de texto </p>
</body>
</html>
```

Display: inline-block

Mensaje de texto Mensaje de texto
Mensaje de texto Mensaje de texto
Mensaje de texto



9. Posicionar o Position

La propiedad ***position*** de CSS establece en qué punto de la página comenzará a posicionarse o a mostrarse el elemento HTML que hayamos establecido en el código.

Es importante saber en qué punto se va a comenzar a “pintar” porque eso es lo que dirá al 100%, qué espacio ocupará cada elemento.

Existen **5 posibles** y diferentes **valores** que puede adoptar ***position***:

- *static*
- *relative*
- *fixed*
- *absolute*
- *sticky*

Los elementos se posicionan entonces utilizando las **coordenadas *top*, *bottom*, *left*, y *right***. Sin embargo, **estas coordenadas no funcionarán a menos que se establezca primero la propiedad *position***. También funcionan de manera diferente dependiendo del valor de la posición.

9.1 Static

Los elementos HTML se posicionan de **forma estática por defecto**. Es la posición natural de los elementos.

Los elementos posicionados estáticamente **no se ven afectados por las coordenadas *top*, *bottom*, *left* y *right***.

Un elemento con ***position: static;*** no se posiciona de ninguna manera especial; siempre **se posiciona según el flujo normal de la página**.

A continuación, se muestra un [ejemplo](#).



```
<!DOCTYPE html>
<html>
<head>
<style>
.padre {
  width: 250px;
  height: 250px;
  margin: 25px;
  background: orange;
}

.hijo {
  width: 50px;
  height: 50px;
  background: red;
  position: static;
}

</style>
</head>
<body>

<h2>position: static;</h2>

<div class="padre">
  <div class="hijo"></div>
</div>

</body>
</html>
```

position: static;



9.2 Relative

Un elemento con **position: relative;** se posiciona en relación a su posición normal.

Por tanto, podemos “relativizar” un elemento hijo con respecto a su padre, e indicarle en qué punto se tiene que “pintar” con respecto al origen. Esto se hace con las coordenadas **top, bottom, left y right**.

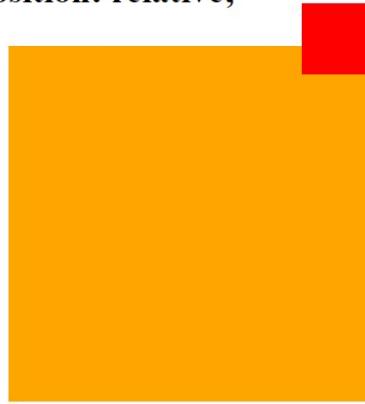
El resto del contenido no se ajustará para encajar en el hueco dejado por el elemento.

A continuación, se muestran algunos [ejemplos](#).



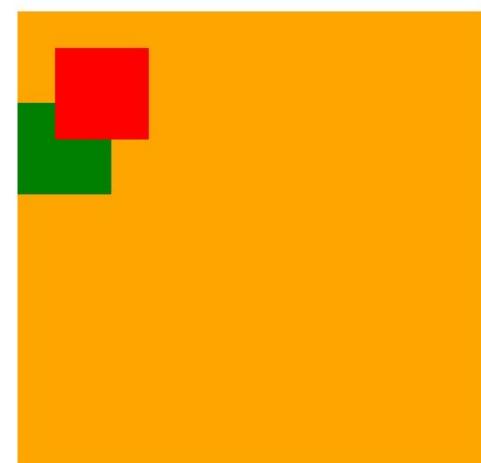
```
.padre {  
    width: 250px;  
    height: 250px;  
    margin: 25px;  
    background: orange;  
}  
  
.hijo {  
    width: 50px;  
    height: 50px;  
    background: red;  
    position: relative;  
    top: -30px;  
    left: 200px;  
}
```

position: relative;



```
<!DOCTYPE html>  
<html>  
<head>  
<style>  
.padre {  
    width: 250px;  
    height: 250px;  
    margin: 25px;  
    background: orange;  
}  
  
.hijo1 {  
    width: 50px;  
    height: 50px;  
    background: red;  
    position: relative;  
    top: 20px;  
    left: 20px;  
}  
  
.hijo2 {  
    width: 50px;  
    height: 50px;  
    background: green;  
}  
  
</style>  
</head>  
<body>  
  
<h2>position: fixed;</h2>  
  
<div class="padre">  
    <div class="hijo1"></div>  
    <div class="hijo2"></div>  
</div>  
  
</body>  
</html>
```

position: relative;



9.3 Fixed

Un elemento con **position: fixed;** se posiciona en relación con la ventana gráfica (**o body**), lo que significa que siempre se mantiene en el **mismo lugar** aunque se desplace la página. Las coordenadas **top**, **right**, **bottom** y **left** se utilizan para posicionar el elemento.

A continuación, se muestra un [ejemplo](#):



```
<!DOCTYPE html>
<html>
<head>
<style>
.padre {
    width: 250px;
    height: 250px;
    margin: 25px;
    background: orange;
}

.hijo1 {
    width: 50px;
    height: 50px;
    background: red;
    position: fixed;
    top: 0px;
    left: 0px;
}

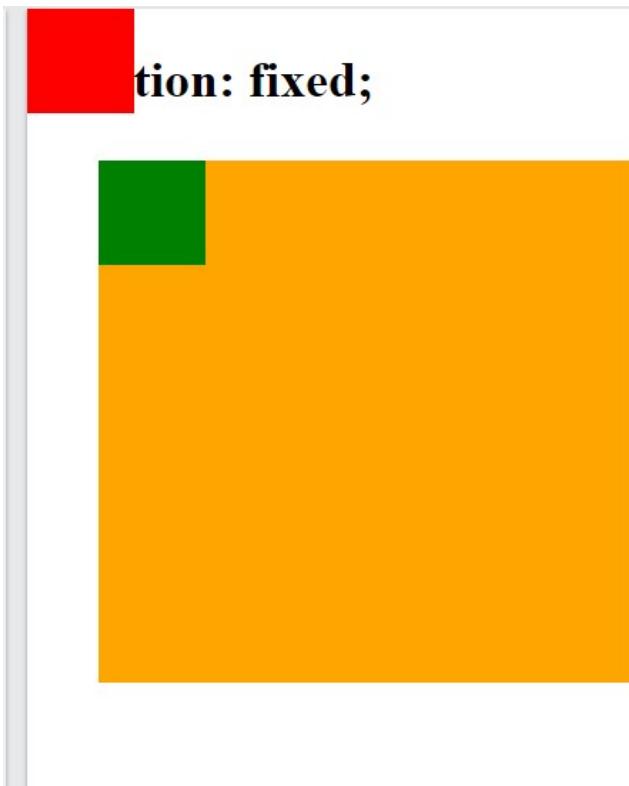
.hijo2 {
    width: 50px;
    height: 50px;
    background: green;
}

</style>
</head>
<body>

<h2>position: fixed;</h2>

<div class="padre">
    <div class="hijo1"></div>
    <div class="hijo2"></div>
</div>

</body>
</html>
```



Un elemento con ***position: fixed;*** se queda fijado y **no se mueve nunca**. Esto es muy útil, por **ejemplo**, en algunos menús en donde queremos que se queden en la parte superior mostrando siempre las opciones de dicho menú, o para anunciar algún producto, o para poner un formulario de suscripción, etc. Lo único que hay que tener en cuenta es que **no moleste al usuario**, ya que si es de un tamaño muy grande puede ocasionar un **efecto negativo** en la UX.

9.4 Absolute

Un elemento con ***position: absolute;*** se **posiciona en relación con el ancestro posicionado más cercano que tenga una posición distinta a *static*** (en lugar de posicionarse en relación con la ventana gráfica, como el fijo).

Los elementos posicionados absolutos son eliminados del flujo normal, y pueden superponerse a los elementos.



```
<!DOCTYPE html>
<html>
<head>
<style>
div.padre {
    width: 400px;
    height: 200px;
    background: orange;
}

div.hijo {
    position: absolute;
    top: 50px;
    right: 0;
    width: 200px;
    height: 100px;
    background: LightCoral;
}
</style>
</head>
<body>

<h2>position: absolute; </h2>

<div class="padre">Este div NO
TIENE POSICION.
    <div class="hijo">Este div
TIENE POSICION position:
absolute;</div>
</div>

</body>
</html>
```

position: absolute;

Este div NO TIENE POSICION.

Este div TIENE POSICION
position: absolute;

Sin embargo, **si un elemento con posición absoluta no tiene ancestros posicionados, utiliza el cuerpo del documento** y se mueve con el desplazamiento de la página.

```
<!DOCTYPE html>
<html>
<head>
<style>
.padre {
    position: relative;
    width: 250px;
    height: 250px;
    margin: 25px;
    background: orange;
}

.hijo1 {
    width: 50px;
    height: 50px;
    background: red;
    position: absolute;
    top: 20px;
    left: 20px;
}

.hijo2 {
    width: 50px;
    height: 50px;
    background: green;
}

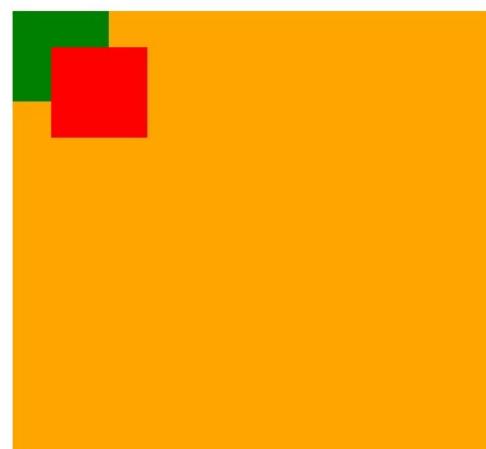
</style>
</head>
<body>

<h2>position: absolute;</h2>

<div class="padre">
    <div class="hijo1"></div>
    <div class="hijo2"></div>
</div>

</body>
</html>
```

position: absolute;





9.5 Sticky

Un elemento con **position: sticky;** se posiciona en función de la posición del scroll del usuario. Por tanto, los elementos son posicionados **de forma relativa hasta que su bloque contenedor alcanza un límite establecido.**

Un elemento **sticky** alterna entre **relative** y **fixed**, dependiendo de la posición del scroll. Se posiciona de forma relativa hasta que se alcanza una determinada posición de desplazamiento en la ventana gráfica - entonces se "pega" en su lugar (como position:fixed).

```
<!DOCTYPE html>
<html>
<head>
<style>
div.sticky {
    position: -webkit-sticky;
    position: sticky;
    top: 0;
    padding: 5px;
    background-color: #cae8ca;
    border: 2px solid #4CAF50;
}
</style>
</head>
<body>

<p>Try to <b>scroll</b> inside this frame to understand how sticky positioning works.</p>

<div class="sticky">I am sticky!</div>

<div style="padding-bottom:2000px">
    <p>In this example, the sticky element sticks to the top of the page (top: 0), when you reach its scroll position.</p>
    <p>Scroll back up to remove the stickyness.</p>
    <p>Some text to enable scrolling.. Lorem ipsum dolor sit amet, illum definitiones no quo, maluisset concludaturque et eum, altera fabulas ut quo. Atqui causae gloriatur ius te, id agam omnis evertitur eum. Affert laboramus repudiandae nec et. Inciderint efficiantur his ad. Eum no molestiae voluptatibus.</p>
    <p>Some text to enable scrolling.. Lorem ipsum dolor sit amet, illum definitiones no quo, maluisset concludaturque et eum, altera fabulas ut quo. Atqui causae gloriatur ius te, id agam omnis evertitur eum. Affert laboramus repudiandae nec et. Inciderint efficiantur his ad. Eum no molestiae voluptatibus.</p>
</div>

</body>
</html>
```

Try to **scroll** inside this frame to understand how sticky positioning works.

I am sticky!

In this example, the sticky element sticks to the top of the page (top: 0), when you reach its scroll position.

Scroll back up to remove the stickyness.

Some text to enable scrolling.. Lorem ipsum dolor sit amet, illum definitiones no quo, maluisset concludaturque et eum, altera fabulas ut quo. Atqui causae gloriatur ius te, id agam omnis evertitur eum. Affert laboramus repudiandae nec et. Inciderint efficiantur his ad. Eum no molestiae voluptatibus.

Some text to enable scrolling.. Lorem ipsum dolor sit amet, illum definitiones no quo, maluisset concludaturque et eum, altera fabulas ut quo. Atqui causae gloriatur ius te, id agam omnis evertitur eum. Affert laboramus repudiandae nec et. Inciderint efficiantur his ad. Eum no molestiae voluptatibus.

I am sticky!

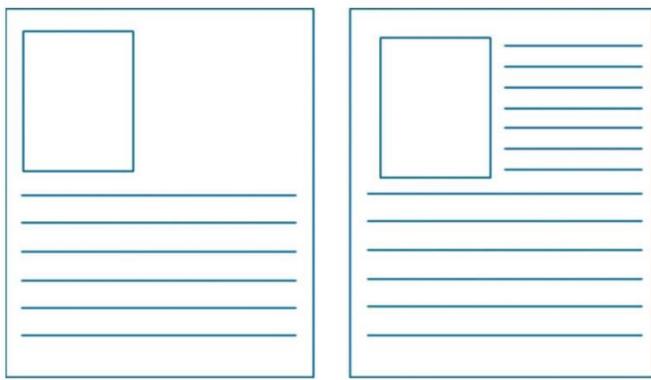
definitiones no quo, maluisset concludaturque et eum, altera fabulas ut quo. Atqui causae gloriatur ius te, id agam omnis evertitur eum. Affert laboramus repudiandae nec et. Inciderint efficiantur his ad. Eum no molestiae voluptatibus.

I am sticky!



10. Float y Clear

Cuando se trabaja a través de cajas para el modelado del código de la página web, **debe tenerse en cuenta la posición** de cada una de ellas. Normalmente, estas se sitúan la una a continuación de la otra, bien en horizontal o en vertical, pero existen ocasiones en las que es deseable que diversas cajas contenedoras de información, ya sea texto, imagen o cualquier otro elemento, **se adapten entre sí**.



Uso de la propiedad 'float'.
Posicionamiento normal frente a
posicionamiento 'float' de la imagen.

En la figura de la izquierda, se observa que en su parte izquierda aparece una fotografía y, a continuación, el texto asociado. Ahora bien, se puede observar cómo en la figura de la derecha, texto e imagen se funden ofreciendo una mejor estética a la web. En este segundo caso, se está utilizando la propiedad de flotación, es decir, '**float**'.

La existencia de cajas flotantes condiciona la situación del resto de cajas. Esto es debido a que aquellos elementos no flotantes, es decir, los que están en línea, adaptan su posición para evitar que se solapen las cajas en función a la colocación de las flotantes.

El **funcionamiento** concreto de esta propiedad se basa en el desplazamiento total hacia la derecha o hacia la izquierda de una caja contenedora con respecto de su posición inicial.

La **sintaxis** es la siguiente:

```
etiqueta html {  
    float: left | right | none;  
}
```

Los tres posibles valores que puede tomar esta propiedad son hacia la **izquierda** (**left**), hacia la **derecha** (**right**) o mantenerse en la **posición original** (**none**). Este último es el valor por defecto y es usado en aquellos casos en los que se desea eliminar la propiedad



'float' definida para una caja. En los casos **left** y **right**, el resto de los elementos se adaptan para mostrarse respectivamente a la derecha o a la izquierda de la caja flotante; si no se desea que el resto de los elementos tomen este papel, se utilizará la propiedad '**clear**'.

La propiedad '**clear**' establece si un elemento debe estar al lado de los elementos flotantes que lo preceden o si debe situarse bajo 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.

Ejemplos:

```
<!DOCTYPE html>
<html>
<head>
<style>
img {
  float: left;
  margin-right: 2em;
  margin-bottom: .3em;
}
</style>
</head>
<body>

<h2>Float Left</h2>

<p>
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac. In at libero sed nunc venenatis imperdiet sed ornare turpis. Donec vitae dui eget tellus gravida venenatis. Integer fringilla congue eros non fermentum. Sed dapibus pulvinar nibh tempor porta. Cras ac leo purus. Mauris quis diam velit.</p>

</body>
</html>
```

Float Left



Facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac. In at libero sed nunc venenatis imperdiet sed ornare turpis. Donec vitae dui eget tellus gravida venenatis. Integer fringilla congue eros non fermentum. Sed dapibus pulvinar nibh tempor porta. Cras ac leo purus. Mauris quis diam velit.

```
<!DOCTYPE html>
<html>
<head>
<style>
img {
  float: none;
}
</style>
</head>
<body>

<h2>Float none</h2>

<p>
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac. In at libero sed nunc venenatis imperdiet sed ornare turpis. Donec vitae dui eget tellus gravida venenatis. Integer fringilla congue eros non fermentum. Sed dapibus pulvinar nibh tempor porta. Cras ac leo purus. Mauris quis diam velit.</p>

</body>
</html>
```

Float none



Facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac. In at libero sed nunc venenatis imperdiet sed ornare turpis. Donec vitae dui eget tellus gravida venenatis. Integer fringilla congue eros non fermentum. Sed dapibus pulvinar nibh tempor porta. Cras ac leo purus. Mauris quis diam velit.



```
<!DOCTYPE html>
<html>
<head>
<style>
.div1 {
  float: left;
  padding: 10px;
  border: 3px solid #73AD21;
}

.div2 {
  padding: 10px;
  border: 3px solid red;
}

.div3 {
  float: left;
  padding: 10px;
  border: 3px solid #73AD21;
}

.div4 {
  padding: 10px;
  border: 3px solid red;
  clear: left;
}
</style>
</head>
<body>

<h2>Without clear</h2>
<div class="div1">div1</div>
<div class="div2">div2 - Notice that div2 is after div1 in the HTML code.
  However, since div1 floats to the left, the text in div2 flows
  around div1.</div>
<br><br>

<h2>With clear</h2>
<div class="div3">div3</div>
<div class="div4">div4 - Here, clear: left; moves div4 down below the
  floating div3. The value "left" clears elements floated to the left.
  You can also clear "right" and "both".</div>

</body>
</html>
```

Without clear

div1 div2 - Notice that div2 is after div1 in the HTML code.
However, since div1 floats to the left, the text in div2 flows around div1.

With clear

div3

div4 - Here, clear: left; moves div4 down below the floating div3. The value "left" clears elements floated to the left. You can also clear "right" and "both".



11. Propiedades de visibilidad

En este apartado veremos algunas propiedades CSS relacionadas con la visibilidad de los elementos HTML en nuestra interfaz web. Estos son: ***z-index***, ***overflow***, y ***visibility***.

11.1 'z-index'

La propiedad ***z-index*** indica la posición de un elemento en el eje z, es decir, la **profundidad** que tiene un elemento respecto a otro de cara al usuario. Vendría a ser el equivalente a las capas de Photoshop.

z-index sólo funciona **en elementos posicionados** (*position: absolute*, *position: relative*, *position:fixed* o *position:sticky*) y **elementos flex** (elementos que son hijos directos de elementos *display:flex*).

Si dos elementos posicionados se **superponen** sin un ***z-index*** especificado, el elemento posicionado en último lugar en el código HTML es el que se mostrará en la parte superior.

Su **sintaxis** es la siguiente:

```
etiqueta html {  
    z-index: auto | number;  
}
```

- ***auto***: establece el z-index igual al de sus padres. Es el valor por defecto.
 - ***number***: establece la posición del elemento en el eje z en número. Se permiten números negativos. Si no se especifica el valor, tendrá un valor de 0. **Cuanto mayor es su valor, más al frente tendremos al elemento.**
- Recomendación:** enumerar las capas de z-index de 10 en 10, ya que así evitamos modificar los valores si introducimos una capa nueva en medio.

Ejemplo:



```
<!DOCTYPE html>
<html>
<head>
<style>
img {
    position: absolute;
    left: 0px;
    top: 0px;
    z-index: -1;
    opacity: .3
}
</style>
</head>
<body>

<h1>The z-index Property</h1>



<p>Because the image has a z-index of -1, it
will be placed behind the heading.</p>

</body>
</html>
```

The z-index Property

Because the image has a z-index of -1, it will be placed behind the heading.



11.2 'overflow'

La propiedad CSS **overflow** controla lo que ocurre con el contenido que es demasiado grande para caber en un área. Por tanto, **especifica si se recorta el contenido o se añaden barras de desplazamiento cuando el contenido de un elemento es demasiado grande para caber en el área especificada.**

overflow sólo funciona **para elementos de bloque con una altura especificada**.

La propiedad **overflow** tiene los siguientes valores:

- **visible** - Por defecto. El desbordamiento no se recorta. El contenido se muestra fuera de la caja del elemento.
- **hidden** - El desbordamiento se recorta, y el resto del contenido será invisible. Se utiliza mucho en animaciones.
- **scroll** - El desbordamiento se recorta y se añade una barra de desplazamiento para ver el resto del contenido.
- **auto** - Similar a scroll, pero añade barras de desplazamiento sólo cuando es necesario.

A continuación, se presentan algunos **ejemplos**:



```
<!DOCTYPE html>
<html>
<head>
<style>
div {
    background-color: orange;
    width: 200px;
    height: 65px;
    border: 1px solid;
    overflow: visible;
}
</style>
</head>
<body>
```

Overflow: visible

You can use the overflow property when you want to have better control of the layout. The overflow property specifies what happens if content overflows an element's box.

```
<div>You can use the overflow property when you want to have better control of the layout. The overflow property specifies what happens if content overflows an element's box.</div>

</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
    background-color: orange;
    width: 200px;
    height: 65px;
    border: 1px solid;
    overflow: hidden;
}
</style>
</head>
<body>
```

Overflow: hidden

You can use the overflow property when you want to have better control of the layout. The overflow property specifies what happens if content overflows an element's box.

```
<div>You can use the overflow property when you want to have better control of the layout. The overflow property specifies what happens if content overflows an element's box.</div>

</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
    background-color: orange;
    width: 200px;
    height: 80px;
    border: 1px solid;
    overflow: scroll;
}
</style>
</head>
<body>
```

Overflow: scroll

You can use the overflow property when you want to have better control of the layout. The overflow property specifies what happens if content overflows an element's box.

```
<div>You can use the overflow property when you want to have better control of the layout. The overflow property specifies what happens if content overflows an element's box.</div>

</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
    background-color: orange;
    width: 200px;
    height: 80px;
    border: 1px solid;
    overflow: auto;
}
</style>
</head>
<body>
```

Overflow: auto

You can use the overflow property when you want to have better control of the layout. The overflow property specifies what happens if content overflows an element's box.

```
<div>You can use the overflow property when you want to have better control of the layout. The overflow property specifies what happens if content overflows an element's box.</div>

</body>
</html>
```

11.3 'visibility'

La propiedad **visibility** especifica si un elemento es **visible** o **no**. Si ocultamos elementos con esta propiedad, tenemos que tener en cuenta que **el flujo de posicionamiento de elementos no cambia**, simplemente no se muestra el elemento. Por tanto, **los elementos ocultos ocupan espacio** en la página.

Si queremos ocultar y eliminar un elemento en el diseño del documento, debemos **utilizar la propiedad display**.



Su sintaxis es la siguiente:

```
etiqueta html {  
    visibility: visible | hidden | collapse;  
}
```

- *visible*. Es el valor por defecto. El elemento es visible.
- *hidden*. El elemento está oculto (pero sigue ocupando espacio).
- *collapse*. Sólo para filas de tabla (*<tr>*), grupos de filas (*<tbody>*), columnas (*<col>*), grupos de columnas (*<colgroup>*). Este valor elimina una fila o una columna, pero no afecta al diseño de la tabla. El espacio ocupado por la fila o columna estará disponible para otros contenidos.
- Si se utiliza el *collapse* en otros elementos, se representa como "oculto".

Ejemplo:

```
<!DOCTYPE html>  
<html>  
<head>  
<style>  
h2.a {  
    visibility: visible;  
}  
  
h2.b {  
    visibility: hidden;  
    background: yellow;  
}  
</style>  
</head>  
<body>  
  
<h1>La propiedad visibility</h1>  
  
<h2 class="a">Este heading es visible</h2>  
  
<h2 class="b">Este heading esta oculto</h2>  
  
<p>Fijate que el heading que esta oculto sigue ocupando espacio en la pagina.</p>  
  
</body>  
</html>
```

La propiedad visibility

Este heading es visible

Fijate que el heading que esta oculto sigue ocupando espacio en la pagina.



12. Propiedades de las tablas

En las páginas web existen muchos elementos específicos a los que debemos dar forma para así conseguir la apariencia que más nos interese. Uno de estos elementos son las tablas.

Para controlar la presentación de las tablas tenemos las propiedades: `caption-side`, `table-layout`, `border-collapse`, `border-spacing`, y `empty-cells`.

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

12.1 ‘caption-side’

Esta propiedad sirve para indicar **dónde se pone el título** de la tabla. Puede tener los valores: `top`, `bottom`, `left` y `right`.

La **sintaxis** es la siguiente:

```
etiqueta html {  
    caption-side: top | bottom | left | right;  
}
```

12.2 ‘empty-cells’

Esta propiedad soluciona la carencia del XHTML que, al no dibujar las celdas que estaban vacías, obligaba a poner un espacio en blanco usando el carácter `&nbsp`. Los valores que admite son:

- `show` que permite mostrar los bordes y fondos como en las celdas con contenido.
- `hide` que permite ocultar los bordes y fondos de las celdas vacías.
- `inherit` que permite heredar el valor de empty-cells que tenga su elemento padre.



La **sintaxis** es la siguiente:

```
etiqueta html {  
    empty-cells: show | hide | inherit;  
}
```

12.3 ‘border-collapse’

Permite establecer el modo en el que se dibujan los bordes de las tablas: **separate** (separados), **collapse** (juntos) e **inherit**. En el modo **separate**, cada celda está rodeada por su borde haciendo el efecto de un borde con una línea doble, mientras que, en el modo **collapse** las celdas contiguas comparten sus bordes.

La **sintaxis** es la siguiente:

```
etiqueta html {  
    border-collapse: separate | collapse | inherit;  
}
```

12.4 ‘border-spacing’

Permite establecer la **separación entre celdas contiguas**. Para hacerlo se indica el valor del espaciado horizontal seguido del valor del espaciado vertical. Si se escribe un único valor, la separación horizontal y vertical serán iguales.

Esta propiedad sólo funciona cuando border-collapse es separate.

Los valores que admite son:

- **length length** que especifica la distancia entre los bordes de celdas adyacentes en px, cm, etc. No admite números negativos.
- **initial** que pone la propiedad a su valor por defecto.
- **inherit** que permite heredar el valor de border-spacing que tenga su elemento padre.

La **sintaxis** es la siguiente:

```
etiqueta html {  
    border-spacing: length | initial | inherit;  
}
```



12.5 'table-layout'

Permite definir el modo en el que el navegador dibujará la tabla ya que puede hacerse de dos formas.

Los valores que admite son:

- **fixed**: dibuja la tabla basándose en las medidas establecidas en el código fuente.
Con este valor se consigue que el sistema trabaje más rápido.
- **auto**: dibuja la tabla basándose en el contenido de sus celdas. Es el valor **por defecto**.

A continuación, se muestra un **ejemplo** de todas las propiedades CSS de tablas vistas en este apartado:

```
<!DOCTYPE html>
<html>
<head>
<style>
table, td, th {
    border: 1px solid black;
}

#table1 {
    border-collapse: separate;
    caption-side: top;
    empty-cells: hide;
    border-spacing: 15px 20px;
    border: 1px solid green;
    table-layout: auto;
}

#table2 {
    border-collapse: collapse;
    caption-side: bottom;
    empty-cells: show;
    table-layout: fixed;
    border: 10px dashed green;
}
</style>
</head>
<body>

<table id="table1">
<caption>Table 1 </caption>
<tr>
    <th>Firstname</th>
    <th>Lastname</th>
</tr>
<tr>
    <td>Peter</td>
    <td>Griffin</td>
</tr>
<tr>
    <td>Lois</td>
    </td>
</tr>
</table>

<br><br><br>
<table id="table2">
<caption>Table 2 </caption>
<tr>
    <th>Firstname</th>
    <th>Lastname</th>
</tr>
<tr>
    <td>Peter</td>
    <td>Griffin</td>
</tr>
<tr>
    <td>Lois</td>
    </td>
</tr>
</table>

</body>
</html>
```

The diagram shows two tables, Table 1 and Table 2, side-by-side. Table 1 has a caption 'Table 1' above it. It contains three rows: the first row has two columns labeled 'Firstname' and 'Lastname'; the second row contains two cells with the values 'Peter' and 'Griffin'; the third row contains two cells with the value 'Lois'. Table 2 has a caption 'Table 2' below it. It also contains three rows: the first row has two columns labeled 'Firstname' and 'Lastname'; the second row contains two cells with the values 'Peter' and 'Griffin'; the third row contains two cells with the value 'Lois'. The main difference is that in Table 1, the 'Lastname' column is wider than the 'Firstname' column, while in Table 2, they are of equal width. This visual representation demonstrates how 'table-layout: auto' creates a table where cells expand to fit their content, while 'table-layout: fixed' creates a table where cells have a fixed width based on their definition in the CSS.

Firstname	Lastname
Peter	Griffin
Lois	

Firstname	Lastname
Peter	Griffin
Lois	



13. Propiedades de las listas en CSS

Las propiedades CSS de las listas son las que nos permiten controlar los estilos de los marcadores y la posición de los elementos dentro de las listas. En la siguiente tabla se muestran **las propiedades de las listas** más utilizadas.

Propiedad	Descripción	Valores
<i>list-style-type</i>	Estilo aplicable a los marcadores visuales 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
<i>list-style-image</i>	Imagen aplicable a los elementos de las listas	url(«...») none
<i>list-style-position</i>	Posición del marcador dentro de la lista	inside outside
<i>list-style</i>	Permite establecer varios estilos de la lista en una sola propiedad	list-style-type list-style-position list-style-image

13.1 'list-style-type'

Esta propiedad CSS especifica el **tipo de marcador** de elemento en una lista.

Su **sintaxis** es la siguiente:

```
etiqueta html {  
    list-style-type: disc | circle | square | decimal | decimal-leading-zero | lower-roman | upper-roman | lower-greek | lower-latin | upper-latin | armenian | georgian | lower-alpha | upper-alpha | none;  
}
```

Se pueden ver los diferentes estilos en:

https://www.w3schools.com/cssref/playdemo.php?filename=playcss_list-style-type

A continuación, se presenta un **ejemplo** sencillo:



```
<!DOCTYPE html>
<head>
    <style>
        .a {list-style-type: circle;}
        .b {list-style-type: square;}
        .c {list-style-type: upper-roman;}
        .d {list-style-type: lower-alpha;}
    </style>
</head>
<body>
    <h3>Propiedad list-style-type</h3>
    <ul class="a">
        <li>Tipo círculo</li>
        <li>Tipo círculo</li>
        <li>Tipo círculo</li>
    </ul>
    <ul class="b">
        <li>Tipo cuadrado</li>
        <li>Tipo cuadrado</li>
        <li>Tipo cuadrado</li>
    </ul>
    <p>Listas ordenadas:</p>
    <ol class="c">
        <li>Uno</li>
        <li>Dos</li>
        <li>Tres</li>
    </ol>
    <ol class="d">
        <li>Uno</li>
        <li>Dos</li>
        <li>Tres</li>
    </ol>
</body>
</html>
```

Propiedad list-style-type

- Tipo círculo
- Tipo cuadrado

Listas ordenadas:

- I. Uno
- II. Dos
- III. Tres
- a. Uno
- b. Dos
- c. Tres

13.2 'list-style-image'

Esta propiedad CSS **sustituye el marcador** de elemento de lista **por una imagen**.

Su **sintaxis** es la siguiente:

```
etiqueta html {
    list-style-image: none | url | initial | inherit;
}
```

A continuación, se presenta un ejemplo sencillo:

```
<!DOCTYPE html>
<html>
<head>
<style>
ul {
    list-style-image: url('sqpurple.gif');
}
</style>
</head>
<body>

<h1>The list-style-image Property</h1>

<p>The list-style-image property replaces the list-item marker with an image:</p>

<ul>
    <li>Coffee</li>
    <li>Tea</li>
    <li>Coca Cola</li>
</ul>

</body>
</html>
```

The list-style-image Property

The list-style-image property replaces the list-item marker with an image:

- Coffee
- Tea
- Coca Cola



13.3 'list-style-position'

Esta propiedad especifica la posición de las viñetas o marcadores de los elementos de la lista. Su **sintaxis** es la siguiente:

```
etiqueta html {  
    list-style-position: inside | outside | initial | inherit;  
}
```

- **list-style-position: outside;** significa que las viñetas estarán fuera del elemento de la lista. El inicio de cada línea de un elemento de lista se alinearán verticalmente.
- **list-style-position: inside;** significa que las viñetas estarán dentro del elemento de la lista. Al formar parte del elemento de la lista, formarán parte del texto y lo empujarán al principio

- | |
|---|
| • Coffee - A brewed drink prepared from roasted coffee beans... |
| • Tea |
| • Coca-cola |

- | |
|---|
| • Coffee - A brewed drink prepared from roasted coffee beans... |
| • Tea |
| • Coca-cola |

Esta propiedad se puede ver en el siguiente **ejemplo**:

```
<!DOCTYPE html>  
<html>  
<head>  
<style>  
ul.a {  
    list-style-position: outside;  
}  
  
ul.b {  
    list-style-position: inside;  
}  
</style>  
</head>  
<body>  
  
<h1>The list-style-position Property</h1>  
  
<h2>list-style-position: outside (default):</h2>  
<ul class="a">  
    <li>Coffee - A brewed drink prepared from roasted coffee beans, which are the seeds of berries from the Coffea plant</li>  
    <li>Tea - An aromatic beverage commonly prepared by pouring hot or boiling water over cured leaves of the Camellia sinensis, an evergreen shrub (bush) native to Asia</li>  
    <li>Coca Cola - A carbonated soft drink produced by The Coca-Cola Company. The drink's name refers to two of its original ingredients, which were kola nuts (a source of caffeine) and coca leaves</li>  
</ul>  
  
<h2>list-style-position: inside:</h2>  
<ul class="b">  
    <li>Coffee - A brewed drink prepared from roasted coffee beans, which are the seeds of berries from the Coffea plant</li>  
    <li>Tea - An aromatic beverage commonly prepared by pouring hot or boiling water over cured leaves of the Camellia sinensis, an evergreen shrub (bush) native to Asia</li>  
    <li>Coca Cola - A carbonated soft drink produced by The Coca-Cola Company. The drink's name refers to two of its original ingredients, which were kola nuts (a source of caffeine) and coca leaves</li>  
</ul>  
  
</body>  
</html>
```

The list-style-position Property

list-style-position: outside (default):

- Coffee - A brewed drink prepared from roasted coffee beans, which are the seeds of berries from the Coffea plant
- Tea - An aromatic beverage commonly prepared by pouring hot or boiling water over cured leaves of the Camellia sinensis, an evergreen shrub (bush) native to Asia
- Coca Cola - A carbonated soft drink produced by The Coca-Cola Company. The drink's name refers to two of its original ingredients, which were kola nuts (a source of caffeine) and coca leaves

list-style-position: inside:

- Coffee - A brewed drink prepared from roasted coffee beans, which are the seeds of berries from the Coffea plant
- Tea - An aromatic beverage commonly prepared by pouring hot or boiling water over cured leaves of the Camellia sinensis, an evergreen shrub (bush) native to Asia
- Coca Cola - A carbonated soft drink produced by The Coca-Cola Company. The drink's name refers to two of its original ingredients, which were kola nuts (a source of caffeine) and coca leaves



13.4 'list-style'

Esta propiedad CSS es una abreviatura de las propiedades anteriores, esto es, con esta propiedad podemos establecer **varios estilos de una lista en una sola línea** de código.

Su **sintaxis** es la siguiente:

```
etiqueta html {  
    list-style: list-style-type list-style-position list-style-image  
              | initial | inherit;  
}
```

Ejemplo:

```
<!DOCTYPE html>  
<html>  
<head>  
<style>  
ul {  
    list-style: square inside url("sqpurple.gif");  
}  
</style>  
</head>  
<body>  
  
<h1>The list-style Property</h1>  
  
<p>The list-style is a shorthand property for  
all the list properties.</p>  
  
<ul>  
    <li>Coffee</li>  
    <li>Tea</li>  
    <li>Coca Cola</li>  
</ul>  
  
</body>  
</html>
```

The list-style Property

The list-style is a shorthand property for all the list properties.

- Coffee
- Tea
- Coca Cola



14. Cursos en CSS.

En CSS, los cursos se utilizan para **cambiar la apariencia del cursor** del ratón cuando se encuentra **sobre un elemento interactivo**. Esto permite personalizar la forma del cursor y proporcionar retroalimentación visual al usuario.

CSS proporciona una variedad de valores para la propiedad **cursor**, que se utiliza para especificar el estilo del cursor. Por **ejemplo**:

```
.div {  
    cursor: crosshair;  
}
```

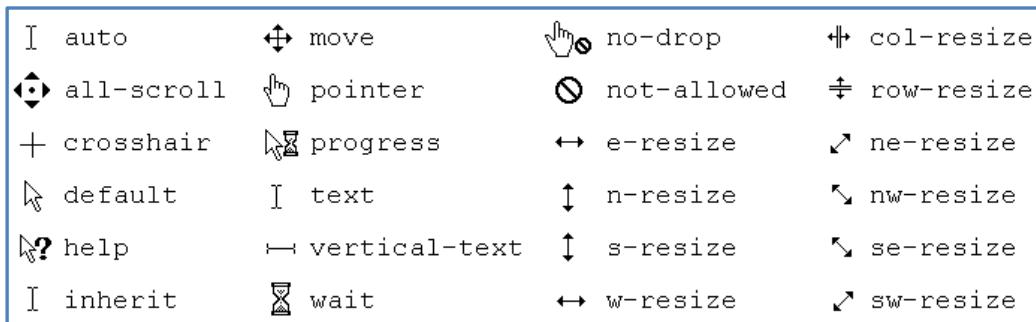
En la siguiente tabla se pueden ver algunos **ejemplos de diferentes cursos** en CSS:

Valor	Descripción
crosshair	Cursor tipo cruz para tareas en las que requieres 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.
grabbing	Mano cerrada, utilizado para indicar que el elemento se puede agarrar.
grab	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.
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.

A continuación, se muestra la apariencia de algunos de ellos:



Estas explicaciones proporcionan una descripción breve de cada cursor, pero es posible encontrar más detalles y variaciones de estos cursores en la especificación CSS y en la documentación de los navegadores. Es importante tener en cuenta que **algunos valores de cursor pueden no ser compatibles en todos los navegadores** o en todos los elementos.

Veamos cómo se visualizan los cursores en un **ejemplo práctico**:

https://www.w3schools.com/cssref/tryit.php?filename=trycss_cursor



15. Definición de propiedades personalizadas

Las **CSS Custom Properties** o **propiedades personalizadas** (también llamadas variables CSS) nos permiten **definir nombres para valores determinados de las propiedades**. De esta forma, se puede **utilizar ese nombre como si fuese una variable**. La gran ventaja es que al modificar el valor de esas «variables» se cambia automáticamente en todos los lugares del documento CSS en los que se haya usado ese nombre.

Esta característica es **utilizada recientemente** ya que no estaba presente en las primeras versiones de CSS ni soportada en todos los navegadores en las siguientes versiones.

Para definir una **propiedad personalizada** pondremos **dos guiones** delante del nombre que queramos utilizar.

Para definir la propiedad personalizada para todo el documento HTML la incluiremos en el **pseudoelemento :root**. **Ejemplo:**

```
:root {  
    --principal-color: black; /* Valor personalizado */  
    --font: "Arial";  
}
```

15.1 Uso de propiedades personalizadas

Para utilizar una **propiedad personalizada** hay que insertar su nombre dentro de la expresión **var()**:

```
.clase {  
    background-color: var(--principal-color);  
    font-family: var(--font);  
}
```



Ejemplo:

```
<!DOCTYPE html>
<head>
  <style>
    :root {
      --principal-color: brown;
      --secondary-color: #e1f4fc;
      --font: "Arial";
    }
    p { color: var(--principal-color);}
    div { background-color: var(--secondary-color);
          font-family: var(--font);}
  </style>
</head>

<body>
  <h1>CSS Custom Properties</h1>
  <p>Este texto se ve de color marrón</p>
  <div>Este fondo se ve verde azul y el texto en
        Arial</div>
  <span>Este texto no tiene ningún estilo definido</
        span>
</body>
</html>
```

CSS Custom Properties

Este texto se ve de color marrón

Este fondo se ve verde azul y el texto en Arial
Este texto no tiene ningún estilo definido



16. Funciones matemáticas

CSS proporciona varias funciones matemáticas que se pueden utilizar para **realizar cálculos y manipular valores numéricos** en propiedades CSS. Algunas de las funciones matemáticas más comunes en CSS son las siguientes:

Propiedad	Descripción	Valores Ejemplo
<code>abs()</code>	Devuelve el valor absoluto de un número.	<code>abs(-10) = 10</code>
<code>sin()</code>	Devuelve el seno de un ángulo.	<code>sin(45deg) ≈ 0.71</code>
<code>cos()</code>	Devuelve el coseno de un ángulo.	<code>cos(60deg) = 0.5</code>
<code>tan()</code>	Devuelve la tangente de un ángulo.	<code>tan(30deg) ≈ 0.58</code>
<code>sqrt()</code>	Devuelve la raíz cuadrada de un número.	<code>sqrt(16) = 4</code>
<code>pow()</code>	Eleva un número a una potencia.	<code>pow(2, 3) = 8</code>
<code>min()</code>	Devuelve el valor mínimo entre dos números.	<code>min(10, 5) = 5</code>
<code>max()</code>	Devuelve el valor máximo entre dos números.	<code>max(10, 5) = 10</code>
<code>random()</code>	Devuelve un número aleatorio entre 0 y 1.	<code>random() ≈ 0.35</code>
<code>floor()</code>	Devuelve el valor entero menor o igual.	<code>floor(3.8) = 3</code>
<code>ceil()</code>	Devuelve el valor entero mayor o igual.	<code>ceil(3.2) = 4</code>
<code>round()</code>	Devuelve el valor redondeado al entero más cercano.	<code>round(3.7) = 4</code>
<code>clamp()</code>	Limita un valor dentro de un rango. Es equivalente a realizar la siguiente operación: <code>max(MIN, min(VAL, MAX))</code> , donde el primer valor sería el valor mínimo, el segundo sería el valor ideal, y el tercero es el valor máximo.	<code>clamp(4, 5, 8) = 5</code> <code>clamp(320px, 90%, 48rem)</code>
<code>calc()</code>	Realiza cálculos matemáticos en propiedades CSS.	<code>width: calc(100% - 20px)</code>
<code>attr()</code>	Obtiene el valor de un atributo HTML y lo utiliza en CSS.	<code>content: attr(data-text)</code>

Se pueden ver más funciones y ejemplos en: <https://lenguajecss.com/css/unidades-css/funciones-css/>.



16.1 calc()

La función matemática `calc()` en CSS se utiliza para realizar cálculos aritméticos y combinar diferentes unidades de medida. Aquí tienes algunos ejemplos de cómo se puede utilizar:

- **Ejemplo 1: Suma de valores.** En este ejemplo, se realiza la suma de 100px y 20px, lo que da como resultado un ancho de 120px.

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

- **Ejemplo 2: Resta de valores.** En este caso, se resta 10px al 80% del ancho disponible, lo que dará como resultado un ancho relativo.

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

- **Ejemplo 3: Multiplicación de valores.** Aquí se realiza la multiplicación de 2 por el 50% del ancho disponible, obteniendo un ancho relativo igual al doble del 50%.

```
width: calc(2 * 50%); /* Resultado: El doble del 50% del ancho  
disponible */
```

- **Ejemplo 4: Combinación de operaciones.** En este ejemplo, se realiza una combinación de operaciones donde se suma 100px con el 10% del ancho disponible y luego se divide el resultado por 3.

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

Es posible anidar llamadas a `calc()` dentro de otras llamadas `calc()`.



Ahora veamos un **ejemplo práctico**: Realización de **cálculos con calc()** de CSS para conseguir un contenedor fijo y otro variable.



```
<!DOCTYPE html>
<head>
<style>
#div1 {
    float:left;
    width: calc(100% - 230px);
    background-color: yellow;
    padding: 15px;
    margin-bottom: 15px;
    clear: both;
    box-sizing: border-box;
}
#div2{
    float:left;
    width: 230px;
    padding: 15px;
    box-sizing: border-box;
}
</style>
</head>

<body>
<div id="div1">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.</div>
<div id="div2">Other text</div>

<div id="div1">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.</div>
<div id="div2"><button>Button</button></div>

<div id="div1">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.</div>
<div id="div2">
    </div>
</body>
</html>
```

Other text

Button

UNO

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.

16.2 min()

La función matemática **min()** en CSS se utiliza para obtener el **valor mínimo entre varios valores** o dimensiones. A continuación, se pueden ver algunos **ejemplos** de cómo se puede utilizar:

- **Ejemplo 1: Obtener el valor mínimo entre dos anchos.** En este ejemplo, se obtiene el valor mínimo entre 200px y el 50% del ancho disponible, lo que dará como resultado el valor más pequeño entre ambos.

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



- **Ejemplo 2: Obtener el valor mínimo entre tres alturas.** Aquí se obtiene el valor mínimo entre 100px, 150px y el 50% del alto disponible, resultando en la altura más pequeña entre los tres valores.

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

- **Ejemplo 3: Combinar valores absolutos y relativos.** En este caso, se obtiene el valor mínimo entre 16px y 1.2 veces el tamaño de fuente actual, lo que dará como resultado el tamaño de fuente más pequeño entre ambos.

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

16.3 max()

La función matemática **max()** en CSS se utiliza para obtener el valor máximo entre varios valores o dimensiones. A continuación, se muestran algunos **ejemplos** de cómo se puede utilizar:

- **Ejemplo 1: Obtener el valor máximo entre dos anchos.** En este ejemplo, se obtiene el valor máximo entre 200px y el 50% del ancho disponible, lo que dará como resultado el valor más grande entre ambos.

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

- **Ejemplo 2: Obtener el valor máximo entre tres alturas.** Aquí se obtiene el valor máximo entre 100px, 150px y el 50% del alto disponible, resultando en la altura más grande entre los tres valores.

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



- **Ejemplo 3: Combinar valores absolutos y relativos.** En este caso, se obtiene el valor máximo entre 16px y 1.2 veces el tamaño de fuente actual, lo que dará como resultado el tamaño de fuente más grande entre ambos.

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

`min()`, `max()`, `clamp()` pueden utilizarse junto con `calc()` y además pueden anidarse, es decir, utilizarse combinadas:

`calc(min()), max(calc()), calc(max(min()))`, etc.



16.4 round()

La función matemática `round()` en CSS se utiliza para redondear un número al entero más cercano. Aquí tienes algunos ejemplos de cómo se puede utilizar:

- **Ejemplo 1: Redondear un número decimal.** En este ejemplo el número decimal 3.7rem se redondea al entero más cercano, que es 4rem.

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

- **Ejemplo 2: Redondear un número negativo.** Aquí, el número negativo -2.3em se redondea al entero más cercano, que es -2em.

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

- **Ejemplo 3: Redondear un número con decimales a un número específico de lugares decimales.** En este caso, el número 1.2345 se redondea a 2 lugares decimales, resultando en 1.23.

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



16.5 attr()

La función matemática **attr()** en CSS se utiliza para obtener el valor de un atributo HTML y utilizarlo en propiedades CSS. Aquí tienes algunos ejemplos de cómo se puede utilizar:

- **Ejemplo 1: Obtener el valor de un atributo y utilizarlo como ancho.** En este ejemplo, se obtiene el valor del atributo **data-width** y se utiliza como valor para la propiedad **width**. Esto puede ser útil cuando se necesita establecer dinámicamente el ancho de un elemento basado en un atributo específico.

```
width: attr(data-width);
```

- **Ejemplo 2: Obtener el valor de un atributo y utilizarlo en una operación matemática.** Aquí, se obtiene el valor del atributo **data-height** y se utiliza en una operación matemática dentro de la función **calc()**. Esto permite realizar cálculos con el valor del atributo y aplicar el resultado como valor para la propiedad **height**.

```
height: calc(attr(data-height) * 2);
```

- **Ejemplo 3: Obtener el valor de un atributo y utilizarlo como contenido de un pseudo-elemento.** En este caso, se obtiene el valor del atributo **data-content** y se utiliza como contenido para un pseudo-elemento. Esto permite mostrar dinámicamente el contenido de un atributo HTML en el estilo de un elemento.

```
content: attr(data-content);
```

Veamos dos **ejemplos** prácticos: en el primero utilizamos el atributo **data-author** para mostrarlo como contenido en el documento HTML, y en el segundo insertaremos el valor del atributo **href** después de cada link utilizando para ello el pseudoelemento **::after**.



1

```
<!DOCTYPE html>
<head>
    <style>
        .element::after {
            content: attr(data-author);
            font-weight: bold;
            color: green;
        }
    </style>
</head>

<body>
    <p class="element" data-author="Andrea">El autor es </p>
</body>
</html>
```

El autor es **Andrea**

2

```
<!DOCTYPE html>
<html>
<head>
<style>
    a::after {
        content: " (" attr(href) ")";
        color: green;
    }
    a { text-decoration: none; }
</style>
</head>
<body>

<h1>La función attr() </h1>
<p>Inserta el valor del atributo href entre paréntesis después de
cada link:</p>
<p><a href="https://www.w3schools.com">Visita W3Schools</a></p>
<p><a href="https://www.w3schools.com/html/">Visita W3Schools'
HTML tutorial</a></p>
</body>
</html>
```

La función attr()

Inserta el valor del atributo href entre paréntesis después de cada link:

Visita W3Schools (<https://www.w3schools.com>)

Visita W3Schools' HTML tutorial (<https://www.w3schools.com/html/>)



17. Transform

Las transformaciones consisten en hacerle **cambios** al diseño a un mismo elemento. Esto no significa que vaya a cambiar su código HTML, sino simplemente la forma en la que se visualiza. Esto es, aplica una **transformación 2D o 3D** a un elemento.

La propiedad **transform** nos permite rotar, escalar, mover, sesgar, etc, elementos.

Antes de usar esta propiedad, habrá que consultar si es **compatible** con el navegador web del usuario y si necesita ir acompañada por un **prefijo**.



Las dos **propiedades** que nos sirven para definir las transformaciones son **transform** y **transform-origin**.

- **transform-origin**. La posición de origen que se utilizará para la transformación es **por defecto el lado superior izquierdo** del elemento.
- **transform**. La **posición de origen** para realizar la transformación es el **eje central** del elemento.

Su **sintaxis** es la siguiente:

```
etiqueta html {  
    transform: none | transform-functions;  
  
    transform-origin: x-axis y-axis z-axis; }
```

- **x-axis**: define dónde se coloca la vista en el eje x. Posibles valores: left, center, right, una medida de longitud, o un porcentaje (%).
- **y-axis**: define dónde se coloca la vista en el eje y. Posibles valores: top, center, bottom, una medida de longitud, o un porcentaje (%).
- **z-axis**: define dónde se coloca la vista en el eje z. Posibles valores: una medida de longitud.



17.1 Tipos de transformaciones.

Las principales transformaciones (*transform-functions*) que tenemos a nuestro alcance son:

- **rotate**. Nos permite rotar los elementos en el sentido de las agujas del reloj. Para ello utilizaremos ángulos (*deg*).

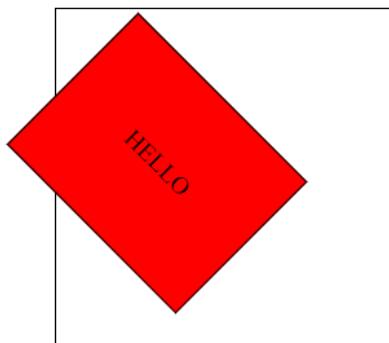
```
transform: rotate(45deg); /* Rota el elemento 45 grados */
```

Con esta transformación se suele utilizar la propiedad '*transform-origin*' que indica el punto de referencia sobre el que va a rotar el elemento, sobre el ejeX, ejeY y ejeZ. Su valor **por defecto es 50% 50% 0**.

A continuación, se presenta un **ejemplo**:

```
<!DOCTYPE html>
<html>
<head>
<style>
#div1 {
    position: relative;
    height: 200px;
    width: 200px;
    margin: 100px;
    padding: 10px;
    border: 1px solid black; }
#div2 {
    padding: 50px;
    position: absolute;
    border: 1px solid black;
    background-color: red;
    transform: rotate(45deg);
    transform-origin: 20% 40%; }
</style>
</head>
<body>
<h1>The transform-origin Property</h1>
<div id="div1">
    <div id="div2">HELLO</div>
</div>
</body>
</html>
```

The transform-origin Property



- **skew**. Nos permite 'deformar' los elementos. Para ello utilizaremos ángulos (*deg*). Esta función se establece con uno o dos valores: *skew(x)*, *skewX()*, *skewY()* o *skew(x,y)*.

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



- **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)`, `translateX()`, `translateY()` 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, etc.

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

- **scale**. Nos permite **cambiar la escala** de los elementos, es decir, nos permite **modificar su tamaño**. La función `scale()` se establece con uno o dos valores, que representan la cantidad de escala que se aplica en cada dirección: `scale(x)`, `scaleX()`, `scaleY()` 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 */
```

- **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)`.

```
matrix(scaleX(), skewY(), skewX(), scaleY(), translateX(), translateY())
```

Los dos últimos valores representan la translación y los primeros la transformación lineal.



Como ya se ha comentado, en las transformaciones `skew`, `scale` y `translate` tenemos **variaciones**, es decir, podemos llevar a cabo una variación para el eje X, otra para el eje Y, o en ambos ejes a la vez:

```
transform: transform-function(cantidad);  
transform: transform-function X(cantidad);  
transform: transform-function Y(cantidad);  
transform: transform-function (cantidadX, cantidadY);  
transform: transform-function () transform-function () transform-  
function ()...;
```



La **diferencia** entre '`translate`' y '`position`' con coordenadas, es que el '`translate`' **no afecta al flujo** de colocación de elementos del resto de cosas que haya en la página web.

Cuando se hace un '`transform`', normalmente es porque está bajo estados de **pseudoclases** (por ejemplo, `:hover`). En este caso, la transición se hace muy brusca, y es por ello que la propiedad `transform` se suele utilizar junto con la propiedad `transition`, con el objetivo de tener una **transición espaciada en el tiempo**.

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

A continuación, se muestran algunos **ejemplos sencillos**:



```
<!DOCTYPE html>
<html>
<head>
<style>
div.a {
    width: 150px;
    height: 80px;
    background-color: orange;
    transform: rotate(20deg);
    transform-origin: bottom center; /*ejeX, ejeY*/
}

div.b {
    width: 150px;
    height: 80px;
    background-color: orange;
    transform: skew(30deg, 5deg);
    transform-origin: 50% -80%;
}

div.c {
    width: 150px;
    height: 80px;
    background-color: orange;
    transform: scaleY(1.5);
}

div.d {
    width: 150px;
    height: 80px;
    background-color: orange;
    transform: translateX(80px);
}
</style>
</head>
<body>

<h1>The transform Property</h1>

<h2>transform: rotate(20deg);</h2>
<div class="a">Hello World!</div>
<br>

<h2>transform: skew(30deg, 5deg);</h2>
<div class="b">Hello World!</div>
<br>

<h2>transform: scaleY(1.5);</h2>
<div class="c">Hello World!</div>

<h2>transform: translateX(80px);</h2>
<div class="d">Hello World!</div>

</body>
</html>
```

The transform Property

transform: rotate(20deg);



transform: skew(30deg, 5deg);



transform: scaleY(1.5);



transform: translateX(80px);





18. Transition

Las transiciones CSS permiten **cambiar** los valores de las propiedades **suavemente**, **durante una duración determinada**. Nos permiten **rotar**, **torcer**, **escalar** o **desplazar** los elementos de nuestra página web.

Este tipo de propiedades son muy interesantes para convertir el lenguaje de hojas de estilo en un sistema capaz de realizar todo tipo de efectos visuales.

Para crear un efecto de transición, se deben especificar **dos** cosas:

- 1) La **propiedad CSS** a la que se quiere añadir un efecto. **El efecto de la transición comenzará cuando la propiedad CSS especificada cambie de valor.**
- 2) La **duración** del efecto. **Si no se especifica la duración, la transición no tendrá efecto**, porque el valor por defecto es 0.

Para realizar las transiciones tenemos a nuestra disposición las siguientes **propiedades**:

- **transition-property**. Indica a qué propiedades queremos que se le aplique la transición. Por ejemplo, **all**, **width**, **height**, etc. Su **sintaxis** es la siguiente:

```
transition-property: none | all | property | initial | inherit;
```

- **transition-duration**. Indica la **duración de la transición**, es decir, cuánto tiempo queremos que tarde el navegador en reproducir el cambio entre las características iniciales a las características que le hemos indicado (no solo a características de **transform**, sino también a características como pueden ser el color).

```
transition-duration: time | initial | inherit;
```

- **transition-timing-function**. Especifica la **curva de velocidad** del efecto de transición. Permite que un efecto de transición **cambie de velocidad** a lo largo de su duración, por ejemplo, primero más rápido y luego más lento.

```
transition-timing-function: linear | ease | ease-in | ease-out |  
ease-in-out | step-start | step-end | steps(int,start|end) | cubic-  
bezier(n,n,n,n) | initial | inherit;
```



- **linear**: misma velocidad de principio a fin.
- **ease**: Valor por defecto. Especifica un inicio lento, luego rápido y un final lento.
- **ease-in**: comienzo lento.
- **ease-out**: final lento.
- **ease-in-out**: comienzo y final lentos.
- **steps(int,start/end)**: especifica una función de pasos. El primer parámetro especifica el número de **intervalos** de la función. Debe ser un número entero positivo (mayor que 0). El segundo parámetro, que es opcional, tiene el valor "start" o "end", y especifica el punto en el que se produce el cambio de valores dentro del intervalo. Si se omite el segundo parámetro, se le da el valor "end".
- **step-start**: equivalente a steps(1, start).
- **step-end**: equivalente a steps(1, end).
- **cubic-bezier(n,n,n,n)**: es una función de tiempo personalizada y por tanto podemos definir nuestros propios valores en la función cubic-bezier. Los valores posibles son valores numéricos de 0 a 1.

https://www.w3schools.com/cssref/tryit.php?filename=trycss_func_cubic-bezier

- **transition-delay**. Especifica el **retardo** (en segundos) para el efecto de transición.

```
transition-delay: time | initial | inherit;
```

- **transition**. Establece las **cuatro propiedades de transición anteriores en una sola propiedad**.

```
transition: transition-property | transition-duration | transition-timing-function | transition-delay | initial | inherit;
```

A continuación, se muestran un **ejemplo**:



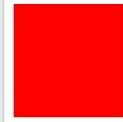
```
<!DOCTYPE html>
<html>
<head>
<style>
div {
    width: 100px;
    height: 100px;
    background: red;
    transition-property: all;
    transition-duration: 5s;
    transition-timing-function: ease-in-out;
    transition-delay: 2s;
}
div:hover {
    transform: rotate(240deg) skew(20deg);
    background: green;
    width: 300px;
}
</style>
</head>
<body>
```

Al poner el ratón sobre el elemento, este rotará 240º y se deformará 20º. Al mismo tiempo se pondrá verde y aumentará su ancho a 300px:

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

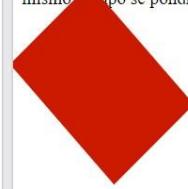
La propiedad transition

Al poner el ratón sobre el elemento, este rotará 240º y se deformará 20º. Al mismo tiempo se pondrá verde y aumentará su ancho a 300px:



La propiedad transition

Al poner el ratón sobre el elemento, este rotará 240º y se deformará 20º. Al mismo tiempo se pondrá verde y aumentará su ancho a 300px:



La propiedad transition

Al poner el ratón sobre el elemento, este rotará 240º y se deformará 20º. Al mismo tiempo se pondrá verde y aumentará su ancho a 300px:



La propiedad transition

Al poner el ratón sobre el elemento, este rotará 240º y se deformará 20º. Al mismo tiempo se pondrá verde y aumentará su ancho a 300px:





19. Iconografía

Los iconos pueden añadirse fácilmente a nuestro código HTML, utilizando una biblioteca de iconos como **Fontawesome**, **Google Icons**, **Bootstrap Icons**, o **icomoon**. Estas librerías son **vectores escalables** que se pueden personalizar con CSS (tamaño, color, sombra, etc.).

19.1 Font Awesome

Para utilizar los iconos gratuitos de Font Awesome, podemos optar por los siguientes dos métodos:



- 1) **Registrarnos** para obtener una cuenta en Font Awesome, y obtener un código (llamado KIT CODE) para utilizarlo cuando añadamos Font Awesome a nuestra página web.
- 2) **Descargar** la biblioteca de Font Awesome en nuestro almacenamiento local.

A continuación, se describe cómo añadir iconos y su aplicación de estilos por cada uno de los métodos mencionados.

19.1.1 Registro u online

Para integrar los iconos deseados en nuestro código HTML mediante el registro en la página u online, debemos llevar a cabo los siguientes pasos:

- 1) Nos registramos en la página de Font Awesome. Vamos a nuestro perfil, y allí encontraremos nuestro KIT Code, el cual insertaremos dentro del `<head>` de nuestro código HTML.

The screenshot shows the 'd905c456b2' kit page. At the top, there are tabs for 'How to Use', 'Settings', and 'Uploaded Icons'. Below the tabs, a step-by-step guide says '1 Add Your Kit's Code to a Project'. It includes a code snippet: `<script src="https://kit.fontawesome.com/d905c456b2.js" crossorigin="anonymous"></script>`. There are buttons for 'Copy Kit Code!' and 'Download Example File'. A note at the bottom says: 'Copy your kit's code into the <head> of each template or page in your project where you want to use Font Awesome via this kit. If you've got other versions or flavors of Font Awesome already in use, make sure to remove them, mmkay?'.



```
<!doctype html>
<html>
  <head>
    <!-- Place your kit's code here -->
    <script src="https://kit.fontawesome.com/d905c456b2.js" crossorigin="anonymous"></script>
  </head>
```

- 2) Navegamos en la página hasta encontrar los iconos que nos interesen, y **copiamos su código** en nuestro código HTML como cualquier otra clase, y le damos estilo.

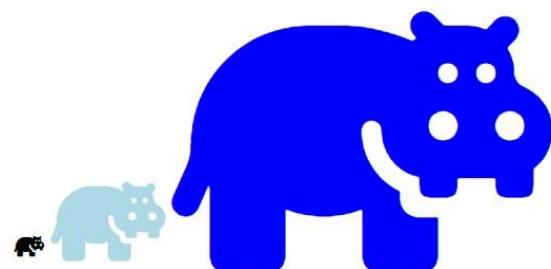
The screenshot shows the Font Awesome website interface. At the top, there are tabs for 'Classic' and 'Sharp'. Below the tabs, there are several icons: a blue elephant, four small squares, and a large blue hippo. The word 'hippo' is typed into a search bar. On the right, there are icons for 'f6ed' and a blue elephant. Below the search bar, there is a code editor window with three tabs: 'HTML', 'React', and 'Vue'. The 'HTML' tab contains the code: `class="fa-sharp fa-solid fa-hippo"></i>`. A red arrow points to this line of code. At the bottom, there is a yellow button with the text 'Start Using This Pro Icon'.

Font Awesome está diseñado para usar con elementos inline. Los elementos `<i>` y `` son ampliamente utilizados para iconos.

Ejemplo:

```
<!doctype html>
<html>
  <head>
    <!-- Place your kit's code here -->
    <script src="https://kit.fontawesome.com/d905c456b2.js" crossorigin="anonymous"></script>
  </head>

  <body>
    <i class="fa-sharp fa-solid fa-hippo"></i>
    <i class="fa-sharp fa-solid fa-hippo" style="font-size:60px;color:lightblue;"></i>
    <i class="fa-sharp fa-solid fa-hippo" style="font-size:200px;color:blue;"></i>
  </body>
</html>
```





19.1.2 Descarga o local

Para integrar los iconos deseados en nuestro código HTML mediante su descarga en nuestro almacenamiento local, debemos llevar a cabo los siguientes pasos:

- 1) Descargamos los iconos de la página web (<https://fontawesome.com/download>) y los guardamos en una carpeta debidamente nombrada, por ejemplo, con el nombre de *icons*.
- 2) Insertaremos dentro del `<head>` de nuestro código HTML, el siguiente `<script>` con la ruta en donde hayamos guardado los iconos.

```
<!DOCTYPE html>
<html>
<head>
<title>Font Awesome Icons</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
<script src="fonts/js/all.js"></script>
</head>
<body>
```

```
<h1>Font Awesome icon library</h1>
```

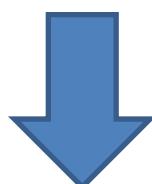
```
<p>Some Font Awesome icons:</p>
<span class="fas fa-cloud"></span>
<i class="fas fa-heart"></i>
<i class="fas fa-car"></i>
<i class="fas fa-file"></i>
<i class="fas fa-bars"></i>
<i class="fa-solid fa-envelope"></i>
<i class="fas fa-spinner fa-pulse"></i>
```

Este ícono es animado.

La clase `fa-spin` consigue que cualquier ícono gire, y la clase `fa-pulse` consigue que cualquier ícono gire con 8 pasos.

```
<p>Styled Font Awesome icons (size and color):</p>
<i class="fa-solid fa-envelope" style="font-size:24px;color:red;"></i>
<i class="fas fa-file" style="font-size:40px;"></i>
<span class="fas fa-cloud" style="font-size:36px;color:aqua;"></span>
<i class="fas fa-bars" style="font-size:48px;color:orange;"></i>
<i class="fas fa-heart" style="font-size:60px;color:lightblue;"></i>
<i class="fas fa-spinner fa-pulse" style="font-size:50px;color:red;"></i>
```

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





Font Awesome icon library

Some Font Awesome icons:



Styled Font Awesome icons (size and color):



Toda la información relativa a los iconos de Fontawesome se pueden consultar en
https://www.w3schools.com/icons/fontawesome5_intro.asp

19.2 Google Icons

Para utilizar los iconos de Google Icons, podemos optar por los siguientes dos métodos:

- 1) **Utilizarlos online, o a nivel externo**, cuando no queremos tener iconos en local. Esta opción es mucho más sencilla y se puede utilizar si no tenemos reparos para usar recursos externos en nuestra página web.
- 2) A **nivel interno**, descargando la fuente, y mediante el uso de **@font-face**.

A continuación, se describe cómo añadir iconos y su aplicación de estilos por cada uno de los métodos mencionados.

19.2.1 Registro u online

El proceso para utilizar los iconos de Google Icons directamente en nuestro código, es el siguiente:

- 1) En primer lugar, debemos insertar dentro del **<head>** de nuestro código HTML, la siguiente línea:

```
<link rel="stylesheet"  
      href="https://fonts.googleapis.com/icon?family=Material+Icons">
```



- 2) Se declarará una **clase** de **CSS** adicional llamada **.material-icons**. Cualquier elemento que use esta clase tendrá el CSS correcto para renderizar estos íconos desde la fuente web.

Un ejemplo sería el siguiente:

```
<!DOCTYPE html>
<html>
<head>
<title>Google Icons</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">
</head>
<body>

<h1>Google icon library</h1>

<p>Some Google icons:</p>
<i class="material-icons">cloud</i>
<i class="material-icons">favorite</i>
<i class="material-icons">attachment</i>
<i class="material-icons">computer</i>
<i class="material-icons">traffic</i>
<span class="material-icons">logout</span>
<br><br>

<p>Styled Google icons (size and color):</p>
<i class="material-icons" style="font-size:24px;">cloud</i>
<i class="material-icons" style="font-size:36px;">favorite</i>
<i class="material-icons" style="font-size:48px;color:red;">attachment</i>
<i class="material-icons" style="font-size:60px;color:lightblue;">traffic</i>

</body>
</html>
```



Google icon library

Some Google icons:



Styled Google icons (size and color):





19.2.2 Descarga o local

Para aquellos desarrolladores web que quieran alojar la fuente web por su cuenta, se necesita un poco de configuración adicional.

- 1) Alojamos la fuente del ícono que deseamos utilizar, en una ubicación (por ejemplo, <https://example.com/material-icons.woff>) y agregamos la siguiente regla de CSS:

```
@font-face {  
    font-family: 'Material Icons';  
    font-style: normal;  
    font-weight: 400;  
    src: url(https://example.com/MaterialIcons-Regular.eot); /* For IE6-8 */  
    src: local('Material Icons'),  
        local('MaterialIcons-Regular'),  
        url(https://example.com/MaterialIcons-Regular.woff2) format('woff2'),  
        url(https://example.com/MaterialIcons-Regular.woff) format('woff'),  
        url(https://example.com/MaterialIcons-Regular.ttf) format('truetype');  
}
```

- 2) Declaramos las reglas de CSS para renderizar el ícono a fin de renderizar la fuente de forma adecuada. A continuación, se muestra un ejemplo:

```
.material-icons {  
    font-family: 'Material Icons';  
    font-weight: normal;  
    font-style: normal;  
    font-size: 24px; /* Preferred icon size */  
    display: inline-block;  
    line-height: 1;  
    text-transform: none;  
    letter-spacing: normal;  
    word-wrap: normal;  
    white-space: nowrap;  
    direction: ltr;  
}
```

- 3) Añadimos los iconos. En el siguiente ejemplo se añade el ícono de un rostro:

```
<span class="material-icons">face</span>
```



Podemos encontrar más información en las siguientes url:

https://developers.google.com/fonts/docs/material_icons#icon_font_for_the_web

https://developers.google.com/fonts/docs/material_icons



19.3 Icomoon

Para utilizar los iconos de Icomoon, podemos optar por los siguientes dos métodos:

- 1) **Utilizarlos online, o a nivel externo**, cuando no queremos tener iconos en local. Si no tenemos la cuenta premium, nos darán un acceso temporal que durará 24 horas.
- 2) A **nivel interno**, descargando la fuente en el almacenamiento local o hosting.

19.3.1 Registro u online

El **proceso** es el siguiente:

- 1) Vamos a la página web <https://icomoon.io/> y seleccionamos “IcoMoon App”. Al hacerlo, podremos **seleccionar** los iconos que nos interesen, y al finalizar clicamos sobre la opción “Generate Font”.

The screenshot shows the IcoMoon website interface. At the top, there's a navigation bar with links: Home, Icon Packs, Plans, FAQ, Docs, News, and About. A red box highlights the "IcoMoon App" button in the top right corner. Below the navigation, there's a large grid of various icons. In the middle of the page, there are three sections: "IcoMoon App", "Icon Packs", and "Go Premium". A blue arrow points downwards from the "IcoMoon App" section towards the main icon grid. The "IcoMoon App" section lists features: Browse 5500+ free vector icons, Custom & crisp icon font generator, Import your own SVGs to make. The "Icon Packs" section lists: 4000+ premium icons, Handcrafted on a grid, In various formats: SVG, font, etc. The "Go Premium" section lists: Cloud syncing, Quick usage links, Hosting icon fonts or SVG icons. At the bottom, there's a toolbar with icons for Import Icons, a search bar, and project management buttons. Below the toolbar is a large grid of icons, with several specific ones highlighted by orange boxes: a GitHub icon, a Facebook icon, an Instagram icon, a WhatsApp icon, a YouTube icon, and a Twitter icon. At the very bottom, there are buttons for "Generate SVG & More", "Selection (6)", and "Generate Font F".



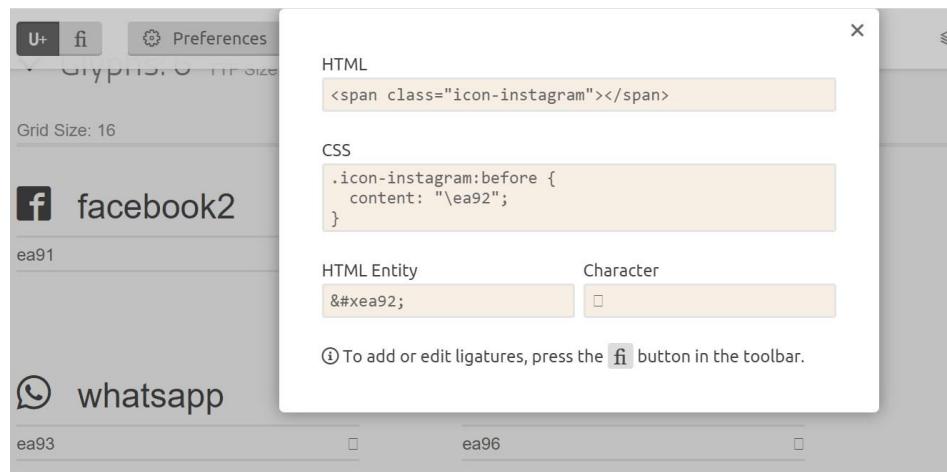
- 2) Seleccionamos la opción “Quick Usage”, y nos aparece un código <link> que introduciremos en el <head> de nuestro código HTML. Hay que tener especial cuidado ya que este link deja de estar disponible pasadas **24 horas**.

The screenshot shows the "Quick Usage and Sharing" section of the IcoMoon interface. It displays a generated CSS code block:

```
<link rel="stylesheet" href="https://i.icomoon.io/public/temp/e4477d05e5/UntitledProject/style.css">
```

Below the code, there's a note: "These links expire in 24 hours. To get permanent unchanging links, Go Premium." There are also "Disable Quick Usage" and "View on CodePen" buttons.

- 3) Añadimos cada uno de los iconos seleccionados, mediante el código HTML o CSS que se nos indica en la opción “Get Code” en cada uno de los respectivos iconos.



A continuación, se muestra un ejemplo.

```
<!DOCTYPE html>
<html>
<head>
<title>IcoMoon Icons</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://i.icomoon.io/public/temp/e4477d05e5/UntitledProject/style.css">
</head>
<body>

<p>Styled Icomoon icons (size and color):</p>
<span class="icon-youtube2" style="font-size:50px; color:red;"></span>
<span class="icon-facebook2" style="font-size:50px; color:blue;"></span>
<span class="icon-whatsapp" style="font-size:50px; color:green;"></span>
<span class="icon-instagram" style="font-size:60px; color:pink;"></span>
<span class="icon-twitter" style="font-size:60px; color:aqua;"></span>
</body>
</html>
```





Styled Icomoon icons (size and color):



19.3.2 Descarga o local

Debido a que el link que nos da icomoon sólo está disponible 24 horas, nos conviene descargar y almacenar los iconos en nuestro propio hosting o ficheros locales de nuestra página web. En este caso, para utilizar dichos iconos, debemos seguir los pasos que se indican a continuación.

- 1) Vamos a la página web <https://icomoon.io/> y seleccionamos “IcoMoon App”. Al hacerlo, podremos **seleccionar** los iconos que nos interesen, y al finalizar clicamos sobre la opción “Generate Font”.

The screenshot shows the IcoMoon website interface. At the top, there's a navigation bar with links: Home, Icon Packs, Plans, FAQ, Docs, News, About, and a purple button labeled "IcoMoon App". A red box highlights the "IcoMoon App" button. Below the navigation is a large logo with the text "Pixel Perfect Icon Solutions". The main area features three sections: "IcoMoon App", "Icon Packs", and "Go Premium", each with a list of features. A large blue arrow points downwards from the "IcoMoon App" section towards the icon grid. The icon grid itself contains a variety of social media and utility icons, with several (including YouTube, Instagram, Twitter, and others) highlighted with orange boxes. At the bottom of the grid, there are buttons for "Generate SVG & More", "Selection (6)", and "Generate Font".



- 2) Seleccionamos en la opción “Download”, y nos descargará un fichero .zip que descomprimiremos. El archivo descargado contendrá el siguiente contenido:

- 📁 demo-files
- 📁 fonts
- 🌐 demo.html
- 📄 Read Me.txt
- 📄 selection.json
- ⚡ style.css

- 3) En la carpeta /fonts tendremos nuestros iconos, que copiaremos a nuestra carpeta /fonts de nuestro proyecto. Además, icoMoon ya nos da el código CSS hecho, que solo tendremos que vincular a nuestro código HTML.

A continuación, se muestra un ejemplo:

```
<!DOCTYPE html>
<html>
<head>
<title>IcoMoon Icons</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="style_icomoon.css">
</head>
<body>

<p>Styled Icomoon icons (size and color):</p>
<span class="icon-youtube2" style="font-size:50px; color:red;"></span>
<span class="icon-facebook2" style="font-size:50px; color:blue"></span>
<span class="icon-whatsapp" style="font-size:50px; color:green"></span>
<span class="icon-instagram" style="font-size:60px; color:pink"></span>
<span class="icon-twitter" style="font-size:60px; color:aqua"></span>
<span class="icon-github" style="font-size:60px;"></span>
</body>
</html>
```



Styled Icomoon icons (size and color):





20. Filtros CSS para aplicar efectos en imágenes

Los filtros CSS permiten aplicar efectos visuales a las imágenes. Para aplicar un filtro CSS a una imagen se utiliza la propiedad **filter**.

Aquí se muestra un **ejemplo** de cómo aplicar un **filtro de desenfoque** a una imagen:

HTML:

```

```

CSS:

```
.filtro-desenfoque {  
    filter: blur(5px);  
}
```

En este ejemplo, la clase filtro-desenfoque se aplica a la etiqueta **** para seleccionar la imagen a la que se le aplicará el filtro. La propiedad **filter** se utiliza con el valor **blur(5px)** para aplicar un desenfoque de 5 píxeles a la imagen.

Se puede ajustar el filtro y su valor según el efecto que se deseé lograr. También se pueden combinar múltiples filtros separándolos por espacios en la propiedad **filter**, por ejemplo: **filter: blur(5px) grayscale(100%);** aplicaría tanto un desenfoque como una escala de grises a la imagen.

Es importante tener en cuenta que **la compatibilidad de los filtros CSS puede variar entre los navegadores**.

20.1 Filtros en CSS

A continuación, se presentan algunos **ejemplos** de filtros CSS con una breve explicación.

Los valores utilizados en los ejemplos son solo ilustrativos y se pueden ajustar según tus necesidades.



A. EJEMPLO 1. `filter: grayscale(100%);` Convierte la imagen a **escala de grises**.

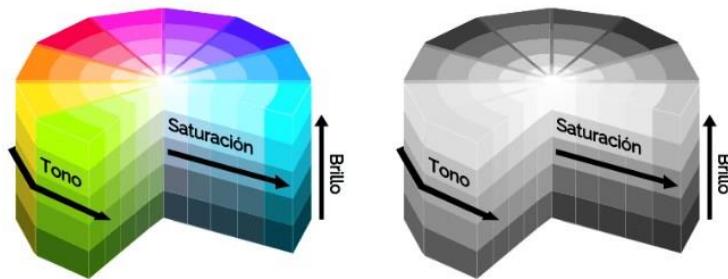
HTML:

```
  
  

```

CSS:

```
.filtro{  
    filter: grayscale(100%);  
}
```



B. EJEMPLO 2. `filter: sepia(100%);` Aplica un **tono sepia** a la imagen.

HTML:

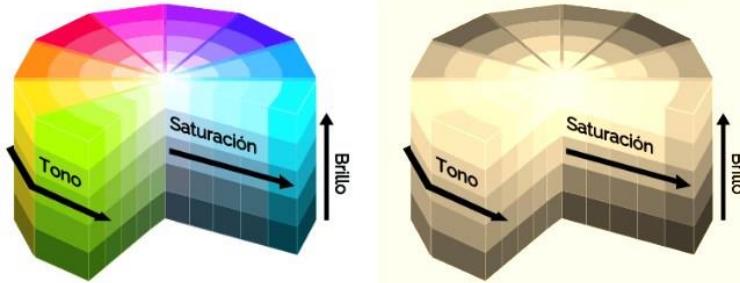
```
  
  

```



CSS:

```
.filtro{  
    filter: sepia(100%);  
}
```



C. EJEMPLO 3. `filter: blur(5px);` Aplica un efecto de **deseñfoque** a la imagen.

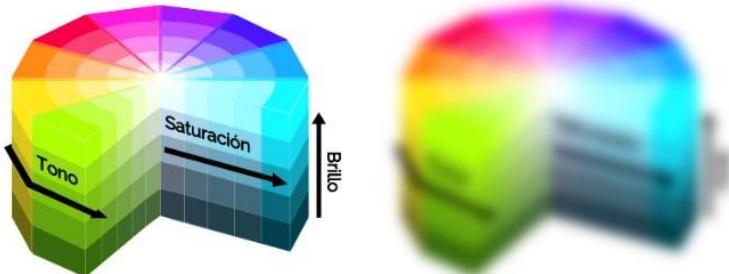
HTML:

```
  
  

```

CSS:

```
.filtro{  
    filter: blur(5px);  
}
```





D. EJEMPLO 4. *filter: brightness(190%);* Ajusta el **brillo** de la imagen.

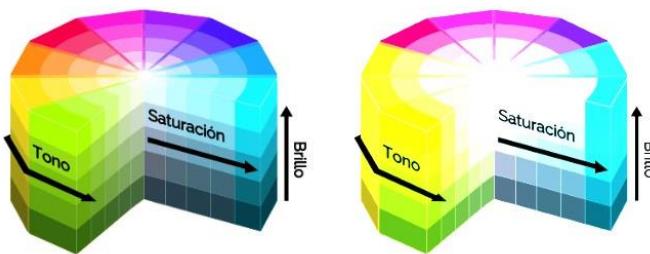
HTML:

```
  
  

```

CSS:

```
.filtro{  
    filter: brightness(190%);  
}
```



E. EJEMPLO 5. *filter: contrast(200%);* Ajusta el **contraste** de la imagen.

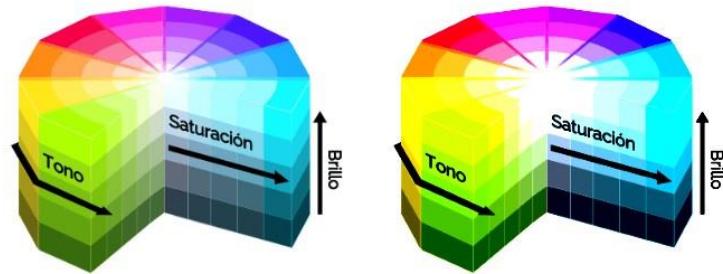
HTML:

```
  
  

```

CSS:

```
.filtro{  
    filter: contrast(200%);  
}
```



F. EJEMPLO 6. `filter: saturate(200%);` Aumenta o disminuye la **saturación** de la imagen.

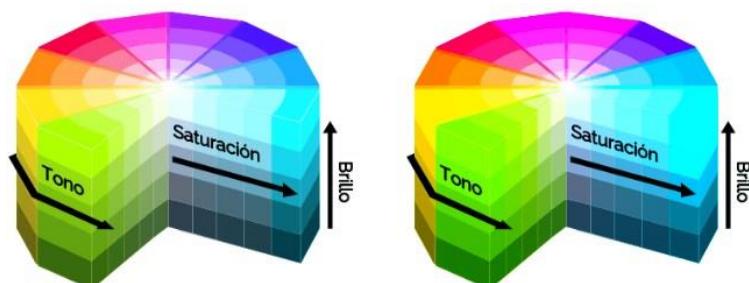
HTML:

```
  
  

```

CSS:

```
.filtro{  
    filter: saturate(200%);  
}
```





G. EJEMPLO 7. *filter: hue-rotate(90deg);* Rota el matiz (tono) de la imagen.

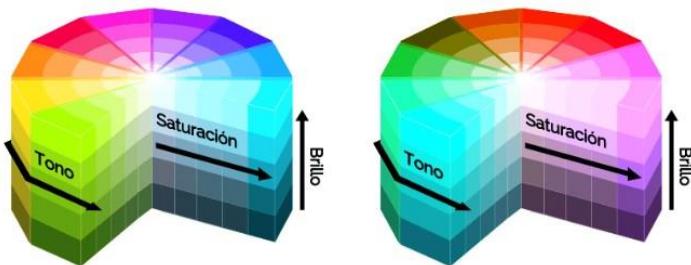
HTML:

```
  
  

```

CSS:

```
.filtro{  
    filter: hue-rotate(90deg);  
}
```



H. EJEMPLO 8. *filter: invert(100%);* Invierte los colores de la imagen.

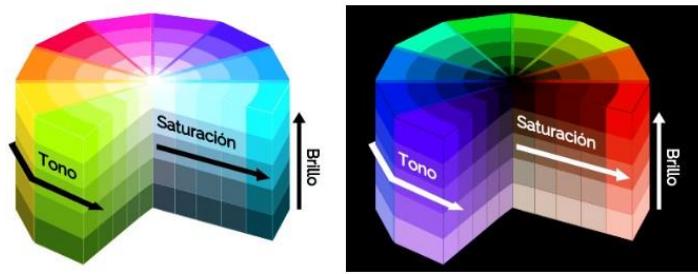
HTML:

```
  
  

```

CSS:

```
.filtro{  
    filter: invert(100%);  
}
```



I. EJEMPLO 9. `filter: opacity(50%);` Ajusta la opacidad de la imagen.

HTML:

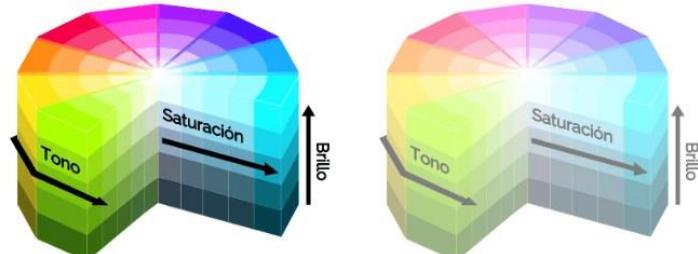
```



```

CSS:

```
.filtro{
  filter: opacity(50%);
}
```





21. El modo oscuro y claro con CSS

A medida que los dispositivos electrónicos forman parte integral de nuestra vida cotidiana, la experiencia del usuario se ha vuelto esencial. Una de las características más solicitadas en la interfaz de usuario es la capacidad de cambiar entre el modo oscuro y el modo claro. Veamos cómo se puede hacer con CSS utilizando la propiedad ***prefers-color-scheme***.

21.1 ¿Qué es prefers-color-scheme?

prefers-color-scheme es una característica de ***Media Query*** introducida en CSS para **detectar** si el **usuario** tiene una **preferencia de esquema de color establecida en su sistema operativo**. Esta característica permite a los desarrolladores adaptar la estética de sus aplicaciones y sitios web al esquema de color preferido del usuario, ya sea oscuro o claro.

21.2 Cómo usar prefers-color-scheme

Para empezar, podemos usar la propiedad dentro de una consulta de medios para aplicar estilos específicos según la preferencia del usuario:

```
/* Estilos por defecto (modo claro) */
body {
    background-color: white;
    color: black;
}

/* Estilos para el modo oscuro, si así lo tiene configurado el usuario
en el sistema operativo */
@media (prefers-color-scheme: dark) {
    body {
        background-color: black;
        color: white;
    }
}
```

Con el código anterior, si un usuario ha establecido su preferencia a modo oscuro en su sistema operativo, nuestro sitio web se adaptará automáticamente para reflejar esta preferencia.



Veamos un ejemplo sencillo de una página web que puede alternar entre modos claro y oscuro utilizando ***prefers-color-scheme*** y un **botón** para permitir a los usuarios cambiar manualmente entre los dos modos usando JS.

- **HTML:**

```
<header>
  <h1>Modo Oscuro / Claro</h1>
  <button id="toggleMode">Cambiar Modo</button>
</header>
<main>
  <p>¡Bienvenido a mi sitio web! Aquí podrás cambiar entre modos
    claro y oscuro.</p>
</main>
```

- **CSS:**

```
body {
  font-family: Arial, sans-serif;
  transition: background-color 0.3s, color 0.3s; }

/* Estilos predeterminados (modo claro) */
body {
  background-color: white;
  color: black; }

header {
  padding: 20px;
  display: flex;
  justify-content: space-between;
  align-items: center; }

button {
  padding: 10px 20px;
  cursor: pointer;
  border: none;
  border-radius: 5px;
  background-color: gray;
  color: white; }

/* Estilos para modo oscuro (detectado automáticamente) */
@media (prefers-color-scheme: dark) {
  body {
    background-color: #333;
    color: white; }

}

/* Estilos para modo oscuro (cambiado manualmente) */
[data-theme='dark'] {
  background-color: #333;
  color: white; }
```



- JS:

```
const toggleButton = document.getElementById('toggleMode');

toggleButton.addEventListener('click', () => {
    if (document.body.getAttribute('data-theme') === 'dark') {
        document.body.removeAttribute('data-theme');
    } else {
        document.body.setAttribute('data-theme', 'dark');
    }
});
```



22. ANEXO I – Lista completa de colores

A continuación, se muestra el listado de colores soportados por los navegadores web 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
	blueviolet	#8a2be2	138,43,226
	brown	#a52a2a	165,42,42
	burlywood	#deb887	222,184,135
	cadetblue	#5f9ea0	95,158,160
	chartreuse	#7fff00	127,255,0
	chocolate	#d2691e	210,105,30
	coral	#ff7f50	255,127,80
	cornflowerblue	#6495ed	100,149,237
	cornsilk	#fff8dc	255,248,220
	crimson	#dc143c	220,20,60
	cyan	#00ffff	0,255,255



	<i>darkblue</i>	#00008b	0,0,139
	<i>darkcyan</i>	#008b8b	0,139,139
	<i>darkgoldenrod</i>	#b8860b	184,134,11
	<i>darkgray</i>	#a9a9a9	169,169,169
	<i>darkgreen</i>	#006400	0,100,0
	<i>darkgrey</i>	#a9a9a9	169,169,169
	<i>darkkhaki</i>	#bdb76b	189,183,107
	<i>darkmagenta</i>	#8b008b	139,0,139
	<i>darkolivegreen</i>	#556b2f	85,107,47
	<i>darkorange</i>	#ff8c00	255,140,0
	<i>darkorchid</i>	#9932cc	153,50,204
	<i>darkred</i>	#8b0000	139,0,0
	<i>darksalmon</i>	#e9967a	233,150,122
	<i>darkseagreen</i>	#8fbcb8	143,188,143
	<i>darkslateblue</i>	#483d8b	72,61,139
	<i>darkslategrey</i>	#2f4f4f	47,79,79
	<i>darkturquoise</i>	#00ced1	0,206,209
	<i>darkviolet</i>	#9400d3	148,0,211
	<i>deeppink</i>	#ff1493	255,20,147
	<i>deepskyblue</i>	#00bfff	0,191,255
	<i>dimgrey</i>	#696969	105,105,105
	<i>dodgerblue</i>	#1e90ff	30,144,255
	<i>firebrick</i>	#b22222	178,34,34
	<i>floralwhite</i>	#ffffaf	255,250,240
	<i>forestgreen</i>	#228b22	34,139,34



	<i>fuchsia</i>	#ff00ff	255,0,255
	<i>gainsboro</i>	#dcdcdc	220,220,220
	<i>ghostwhite</i>	#f8f8ff	248,248,255
	<i>gold</i>	#ffd700	255,215,0
	<i>goldenrod</i>	#daa520	218,165,32
	<i>gray</i>	#808080	128,128,128
	<i>green</i>	#008000	0,128,0
	<i>greenyellow</i>	#adff2f	173,255,47
	<i>grey</i>	#808080	128,128,128
	<i>honeydew</i>	#f0ffff	240,255,240
	<i>hotpink</i>	#ff69b4	255,105,180
	<i>indianred</i>	#cd5c5c	205,92,92
	<i>indigo</i>	#4b0082	75,0,130
	<i>ivory</i>	#ffffff	255,255,240
	<i>khaki</i>	#f0e68c	240,230,140
	<i>lavender</i>	#e6e6fa	230,230,250
	<i>lavenderblush</i>	#fff0f5	255,240,245
	<i>lawngreen</i>	#7fcfc00	124,252,0
	<i>lemonchiffon</i>	#ffffacd	255,250,205
	<i>lightblue</i>	#add8e6	173,216,230
	<i>lightcoral</i>	#f08080	240,128,128
	<i>lightcyan</i>	#e0ffff	224,255,255
	<i>lightgoldenrodyellow</i>	#fafad2	250,250,210
	<i>lightgray</i>	#d3d3d3	211,211,211
	<i>lightgreen</i>	#90ee90	144,238,144



	<i>lightgrey</i>	#d3d3d3	211,211,211
	<i>lightpink</i>	#ffb6c1	255,182,193
	<i>lightsalmon</i>	#ffa07a	255,160,122
	<i>lightseagreen</i>	#20b2aa	32,178,170
	<i>lightskyblue</i>	#87cefa	135,206,250
	<i>lightslategrey</i>	#778899	119,136,153
	<i>lightsteelblue</i>	#b0c4de	176,196,222
	<i>lightyellow</i>	#ffffe0	255,255,224
	<i>lime</i>	#00ff00	0,255,0
	<i>limegreen</i>	#32cd32	50,205,50
	<i>linen</i>	#faf0e6	250,240,230
	<i>magenta</i>	#ff00ff	255,0,255
	<i>maroon</i>	#800000	128,0,0
	<i>mediumaquamarine</i>	#66cdAA	102,205,170
	<i>mediumblue</i>	#0000cd	0,0,205
	<i>mediumorchid</i>	#ba55d3	186,85,211
	<i>mediumpurple</i>	#9370db	147,112,219
	<i>mediumseagreen</i>	#3cb371	60,179,113
	<i>mediumslateblue</i>	#7b68ee	123,104,238
	<i>mediumspringgreen</i>	#00fa9a	0,250,154
	<i>mediumturquoise</i>	#48d1cc	72,209,204
	<i>mediumvioletred</i>	#c71585	199,21,133
	<i>midnightblue</i>	#191970	25,25,112
	<i>mintcream</i>	#f5ffff	245,255,250
	<i>mistyrose</i>	#ffe4e1	255,228,225



	<i>moccasin</i>	#ffe4b5	255,228,181
	<i>navajowhite</i>	#ffdead	255,222,173
	<i>navy</i>	#000080	0,0,128
	<i>oldlace</i>	#fdf5e6	253,245,230
	<i>olive</i>	#808000	128,128,0
	<i>olivedrab</i>	#6b8e23	107,142,35
	<i>orange</i>	#ffa500	255,165,0
	<i>orangered</i>	#ff4500	255,69,0
	<i>orchid</i>	#da70d6	218,112,214
	<i>palegoldenrod</i>	#eeee8aa	238,232,170
	<i>palegreen</i>	#98fb98	152,251,152
	<i>paleturquoise</i>	#afeeee	175,238,238
	<i>palevioletred</i>	#db7093	219,112,147
	<i>papayawhip</i>	#ffefd5	255,239,213
	<i>peachpuff</i>	#ffdab9	255,218,185
	<i>peru</i>	#cd853f	205,133,63
	<i>pink</i>	#ffc0cb	255,192,203
	<i>plum</i>	#dda0dd	221,160,221
	<i>powderblue</i>	#b0e0e6	176,224,230
	<i>purple</i>	#800080	128,0,128
	<i>red</i>	#ff0000	255,0,0
	<i>rosybrown</i>	#bc8f8f	188,143,143
	<i>royalblue</i>	#4169e1	65,105,225
	<i>saddlebrown</i>	#8b4513	139,69,19
	<i>salmon</i>	#fa8072	250,128,114



	<i>sandybrown</i>	#f4a460	244,164,96
	<i>seagreen</i>	#2e8b57	46,139,87
	<i>seashell</i>	#fff5ee	255,245,238
	<i>sienna</i>	#a0522d	160,82,45
	<i>silver</i>	#c0c0c0	192,192,192
	<i>skyblue</i>	#87ceeb	135,206,235
	<i>slateblue</i>	#6a5acd	106,90,205
	<i>slategrey</i>	#708090	112,128,144
	<i>snow</i>	#fffffa	255,250,250
	<i>springgreen</i>	#00ff7f	0,255,127
	<i>steelblue</i>	#4682b4	70,130,180
	<i>tan</i>	#d2b48c	210,180,140
	<i>teal</i>	#008080	0,128,128
	<i>thistle</i>	#d8bfd8	216,191,216
	<i>tomato</i>	#ff6347	255,99,71
	<i>turquoise</i>	#40e0d0	64,224,208
	<i>violet</i>	#ee82ee	238,130,238
	<i>wheat</i>	#f5deb3	245,222,179
	<i>white</i>	#ffffff	255,255,255
	<i>whitesmoke</i>	#f5f5f5	245,245,245
	<i>yellow</i>	#ffff00	255,255,0
	<i>yellowgreen</i>	#9acd32	154,205,50