

# Hojas de estilos



## Caso práctico

Como todas las semanas, se reúnen los miembros del equipo de trabajo de la empresa **BK programación**, para comentar las incidencias de la semana anterior.

**Antonio** ha empleado los bocetos realizados por **Juan** y **María** para elaborar la guía de estilo que habrá que tener en cuenta a la hora de desarrollar la interfaz de la página de la empresa "**Migas Amigas**". Ha hecho un gran trabajo, con una presentación muy meticolosa. Sus compañeros lo han felicitado porque era su primera guía y lo ha hecho muy bien.

–¡Caramba, cómo se nota que eres un experto en ofimática! Menuda presentación. ¡Qué elegante! –Dice **Ada** al hojear el ejemplar de la guía que ha traído impreso.

–¿Podemos ponernos ya con el HTML?– pregunta **Carlos**.

**Ada** sabe que **Carlos** está deseando comenzar su primera página pero tiene que frenarlo un poco porque todavía no tiene mucha experiencia y no es consciente de la utilidad de usar las hojas de estilo en cascada para separar la presentación del contenido.

–No corras tanto –le contesta **Ada**–, hay que preparar una o varias hojas de estilo que se adapten a la guía elaborada por **Antonio** y que permita realizar la página según los bocetos de **Juan** y **María**.

–Pero, no entiendo ¿Por qué es necesaria una hoja de estilo? –vuelve a preguntar Carlos.



Elaboración propia (Uso Educativo no comercial).

# 1.- Introducción a CSS



## Caso práctico

**Carlos** sale de la reunión muy intrigado pensando en todo lo que le contó **Ada** con respecto a las hojas de estilo en cascada. Así que, al llegar a casa, decide buscar información en la web sobre los beneficios de su utilización a la hora de mejorar la presentación de las páginas web.

**Carlos** aprovecha para ir confeccionándose un pequeño manual de CSS en el wiki de **BK Programación** que todos podrán ir editando y completando.



Elaboración propia. Uso Educativo no comercial

Las hojas de estilo en cascada como hoy las conocemos, comenzaron cuando Håkon Lie publicó su primer borrador de hojas de estilo HTML en cascada, al que pronto se le unió  [Bert Bos](#), gran impulsor de este estándar.



Mercury999. Evolución de CSS3. (CC BY-SA)



## Para saber más

En el siguiente enlace puedes encontrar el estado de los diferentes módulos relacionados con CSS:

 [Estándares y borradores relacionados con CSS.](#)

En el siguiente enlace podemos conocer un poco más sobre los diferentes estados por los que pasa un documento antes de ser convertido en estándar.

 [Proceso de estandarización W3C.](#)

En el siguiente enlace podemos conocer la compatibilidad de nuestro navegador con las nuevas etiquetas [HTML](#)

 [Grado de compatibilidad de nuestro navegador con HTML5.](#)

Como hemos comentado anteriormente [CSS](#) se ha dividido en módulos, cada uno de los cuales evoluciona de forma individual, convirtiéndose en estándar. Los navegadores van implementando poco a poco las nuevas funcionalidades que se van desarrollando. Es conveniente saber el estado en que se encuentran las reglas [CSS](#) que vamos a utilizar, para saber qué navegadores lo soportan y cuáles no. Para conocer esto podemos consultar la web oficial de cada navegador o de la [W3C](#), aunque también existe una página de tipo colaborativa denominada "  [Can I use](#)" que nos proporciona dicha información.

## Autoevaluación

**Si quieres definir la presentación de una página web utilizas...**

- [CSS](#)
- [HTML](#)
- [Javascript.](#)
- [PHP](#)



**Debes conocer**

Los enlaces que se muestran a continuación deben servirnos de recordatorio sobre cómo funcionan las principales etiquetas estructurales en HTML5 y su significado semántico.

 [Principales etiquetas estructurales en HTML5](#)

 [Estructura de una página en HTML4 y su correspondiente en HTML5](#)

A continuación dejamos el código relacionado con una estructura de una página simple utilizando etiquetas [HTML5](#). Como verás aplicamos algunas reglas [CSS](#) que posteriormente iremos analizando, si ahora mismo no las comprendes, no te preocupes, conforme avance la unidad entenderás su significado.

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <title>Título de la Web&lt;/title>
    <meta charset="UTF-8" />
    <meta name="title" content="Título de la Web" />
    <meta name="description" content="Ejemplo de página con etiquetas HTML5" />
  </head>
  <body>
    <h1>Título de la Web</h1>
    <div id="main">
      <h2>Subtítulo de la Web</h2>
      <p>Este es un párrafo de la Web</p>
      <img alt="Imagen de ejemplo" data-bbox="150 450 350 550" />
    </div>
  </body>
</html>
```



# 1.1.- Añadir estilos a un documento con CSS

---

Sin duda, no existe ninguna desventaja por utilizar CSS en la maquetación de páginas web, son todo ventajas y, entre ellas, podemos destacar las siguientes:

- ✓ **Mayor control** en el diseño de las páginas: Se puede llegar a diseños fuera del alcance de HTML.
- ✓ **Menos trabajo:** se puede cambiar el estilo de todo un sitio con la modificación de un único archivo.
- ✓ **Documentos más pequeños:** las etiquetas <font> y la gran cantidad de tablas empleadas para dar una buena apariencia a los sitios web desaparecen ahora, por lo que se ahorra código en la configuración de la presentación del sitio.
- ✓ **Documentos mucho más estructurados:** los documentos bien estructurados son accesibles a más dispositivos y usuarios.
- ✓ **El HTML como elemento de presentación está en desuso:** todos los elementos y atributos de presentación de las especificaciones HTML han sido declarados obsoletos por el W3C.
- ✓ **Tiene buen soporte:** en este momento, casi todos los navegadores soportan casi toda la especificación CSS2.1 y la mayoría de las especificaciones del nivel 3.



[project4web](#) | Licencia Pixabay

Pero, ¿cómo funciona CSS? El proceso de funcionamiento de las hojas de estilo en cascada podemos resumirlo en tres pasos:

1. Hay que comenzar con un documento HTML. En teoría, el documento tendrá una estructura lógica y un significado semántico a través de los elementos HTML adecuados. Con HTML se crea la **estructura de la página web**.
2. Luego hay que **escribir las reglas de estilo para definir el aspecto ideal de todos los elementos**. Las reglas seleccionan el elemento en cuestión por su nombre y, a continuación, listan las propiedades (fuente, color, etc.) y los valores que se le van a aplicar.
3. Por último, hay que **vincular los estilos al documento**. Las reglas de estilo pueden reunirse en un documento independiente y aplicarse a todo el sitio, o pueden aparecer en la cabecera y aplicarse solo a ese documento.

La captura de pantalla muestra la página principal del W3C España. El menú superior incluye 'ESTÁNDARES', 'PARTICIPAR', 'UNIRSE' y 'SOBRE EL W3C'. La sección 'ESTÁNDARES' muestra artículos como 'Introducción a la Accesibilidad Web – Nuevo curso online W3Cx' y 'La Web cumple 30 años'. Una barra lateral izquierda titulada 'DIVULGACIÓN' incluye secciones para 'W3C DEVO CAMPUS', 'ESTÁNDARES' (seleccionada), 'VALIDADORES Y SOFTWARE' y 'TESTIMONIOS MIEMBROS W3C'. Una barra lateral derecha titulada 'El W3C por Región' incluye información sobre el W3C y logos de socios como 'igalia'.

Elaboración propia. Captura de pantalla de la web W3C. *Ejemplo de página web con estilos.* [CC0](#)

Lo primero que deberías saber es que las hojas de estilo consisten en una o más reglas que describen cómo debería mostrarse en pantalla un elemento.

A la hora de aplicar las reglas de estilo a un documento HTML, debes tener en cuenta que existen tres modos distintos:

- ✓ Estilos en línea.
- ✓ Hojas de estilos incrustados.
- ✓ Hojas de estilos externas: vinculadas o importadas.



## Debes conocer

A continuación tienes una presentación donde se verán las principales diferencias que existen entre los modos de aplicar las reglas de estilo y su sintaxis.

[Resumen textual alternativo](#)

## 1.2.- Hojas de estilo externas

---

En la presentación del apartado anterior ya has podido ver el modo de emplear las hojas de estilo externas: **importándolas o enlazándolas**. También has visto cómo se crea una regla de estilo y sus componentes: **selector, propiedad y valor**. Pero hay algunas cosas que nos quedan por comentar.

Las hojas de estilo son documentos de texto con, por lo menos, una regla. Estos archivos no contienen ninguna etiqueta HTML, ¿para qué?

Al igual que en los documentos HTML, en las hojas de estilo se pueden incluir comentarios, pero en este caso, se escriben del siguiente modo: /\* Este es un comentario \*/

Desde CSS2 se introduce la posibilidad de orientar las hojas de estilo a medios de presentación específicos. Para ello se emplea el atributo media del elemento link del cual ya viste un ejemplo en la presentación del apartado anterior. A esto es lo que se denomina **media type** (tipo de medios).

La siguiente tabla muestra los diferentes medios que podemos definir, en caso de que no se indique ninguno se cogerá por defecto el valor all.



[Nikin](#) | Licencia Pixabay

Medios de presentación	Obsoleto (uso desaconsejado)	Valor atributo media
all	No	Todos los medios definidos.
braille	Sí	Dispositivos táctiles que emplean el sistema Braille.
embossed	Sí	Impresoras que emplean el sistema Braille.
handheld	Sí	Dispositivos de mano: móviles, PDA, etcétera.
print	No	Impresoras y navegadores en el modo "vista previa para imprimir".
projection	Sí	Proyectores y dispositivos para presentaciones.
screen	No	Pantallas de ordenador.
speech	No	Sintetizadores para navegadores de voz empleados por personas discapacitadas.
tty	No	Dispositivos textuales limitados, como teletipos y terminales de texto.
tv	No	Televisores y dispositivos con resolución baja.

tty, tv, projection, handheld, braille,

Así podemos condicionar la carga de una hoja de estilo en función del tipo de medio.

```
link rel="stylesheet" href="estilos_pantalla.css" media="screen"
```

```
link rel="stylesheet" href="estilos_color.css" media="color"
```

Junto a los medios se pueden utilizar los siguientes operadores `and`, `only` y `not`. Además si queremos aplicar un estilo para diferentes medios estos pueden ser separados por comas.

```
link rel="stylesheet" href="estilos_pantalla.css" media="screen"
```

```
link rel="stylesheet" href="estilos_color.css" media="color"
```

## OPERADOR not

El operador `not` se aplica de forma completa a toda la condición, no a una sola de las partes, así en el siguiente ejemplo:

```
link rel="stylesheet" href="estilos_pantalla.css" media="screen"
```

```
link rel="stylesheet" href="estilos_color.css" media="color"
```

```
link rel="stylesheet" href="estilos_no_color.css" media="not color"
```

Haciendo uso de paréntesis como si fuera una fórmula matemática, el resultado sería `not (screen and color)`.

Cuando se aplica el operador `not`, los estilos se aplicarán si la condición no se cumple. En nuestro caso la condición es pantalla y color, como la condición se cumple (por ejemplo en un monitor actual) no se aplica los estilos del fichero `pantalla_monocromo.css`.

Por el contrario si tuviéramos pantalla y monocromo, la condición no se cumpliría y se aplicarían los estilos del fichero. A continuación se deja un ejemplo con dos ficheros para que puedas realizar diferentes pruebas.

 [Ejemplo media con operadores](#) (zip - 910 B).

## OPERADOR only

El operador `only` se aplicará solamente cuando se cumplan las condiciones que se están indicando.

```
link rel="stylesheet" href="estilos_pantalla.css" media="screen"
```

```
link rel="stylesheet" href="estilos_color.css" media="color"
```

```
link rel="stylesheet" href="estilos_no_color.css" media="not color"
```

En este caso se aplicarán los estilos indicados solo cuando nuestra impresora sea de color.

**Nota:** Para probar este ejemplo, habitualmente las impresoras tienen por defecto desactivado la impresión del fondo, ten cuidado con esto.

## OPERADOR and

El operador `and` nos permite concatenar diferentes condiciones. Veremos cómo posteriormente este operador nos permitirá especificar un dispositivo y a la vez que dicho dispositivo cumpla unas determinadas características como por ejemplo una resolución (ancho-alto) u orientación determinada.

Una vez que hemos conocido las **media type** podremos especificar aún más cómo queremos que se apliquen los estilos. Para ello tendremos lo que se denomina **media query** (indirectamente lo hemos visto en el punto anterior).

Las **media queries**, nos permitirán modificar nuestra página web aplicando reglas de estilo en función de valores o parámetros diferentes de nuestros dispositivos como pueden ser:

- ✓ Orientación (vertical u horizontal),
- ✓ Resolución (número de píxeles),
- ✓ Anchura y altura del dispositivo,
- ✓ Color,
- ✓ etc.

Muchas de estas características se pueden utilizar indicando el prefijo min y max y operadores lógicos ( $<$ ,  $>$ ,  $\geq$ , etc). En el siguiente  [enlace](#) podemos ver la descripción de las características principales que podemos utilizar así como ejemplos de las mismas.

Indicar que las media queries son la base del  [diseño responsive](#), ya que dependiendo del medio en que vayamos a visualizar nuestra web podremos mostrarla de una u otra forma, en función de nuestras necesidades..

Continuando nuestro ejemplo del apartado anterior, vamos a cambiar el aspecto de los principales bloques de nuestro diseño en función de la resolución de la pantalla.

## Código CSS (media query).

```
ul {  
    margin: 0;  
    padding: 0;  
}  
  
.cuadro {  
    width: 100%;  
    height: 100%;  
    position: absolute;  
    background-color: #ccc;  
    border: 1px solid black;  
    border-radius: 10px;  
    left: 0;  
    top: 0;  
}  
  
ul li {  
    list-style-type: none;  
    margin: 0;  
    padding: 0;  
}  
  
ul li a {  
    color: inherit;  
    text-decoration: none;  
}  
  
ul li a:hover {  
    background-color: #ccc;  
    border: 1px solid black;  
    border-radius: 10px;  
    padding: 5px;  
}
```



## Para saber más

A continuación se deja un enlace a las últimas especificaciones dadas por la [W3C](#) sobre Media Queries (Nivel 4), aunque dichas especificaciones están aún en modo edición.

 [Especificaciones Media Queries Level 4](#)



## 1.3.- Conceptos clave de CSS

---

Para que te puedas familiarizar con el comportamiento de CSS, es importante comprender una serie de conceptos clave.

Un documento HTML tiene una estructura determinada que es equivalente a un árbol genealógico cuando se hace referencia a la relación entre elementos: Estructura y herencia.

- ✓ Se dice que un elemento es "hijo" de otro si está contenido directamente en él y este último pasa a ser su "padre". Por ejemplo: el elemento p es hijo del elemento body y el elemento body es padre del elemento p.
- ✓ Los elementos que tienen el mismo parente son "hermanos".  
Por ejemplo: un elemento p puede ser hermano de otro elemento p si ambos son hijos directos del elemento body.

Controlar la relación parente-hijo es fundamental para el funcionamiento de CSS. Un hijo puede "heredar" valores de propiedad de su parente. Con una buena planificación, la herencia puede emplearse para hacer más eficiente la especificación de los estilos.

Este principio por el que algunas reglas se ignoran y otras se heredan nos introduce un concepto muy importante: "**la cascada**".

### Reglas de estilo en conflicto: la "cascada".

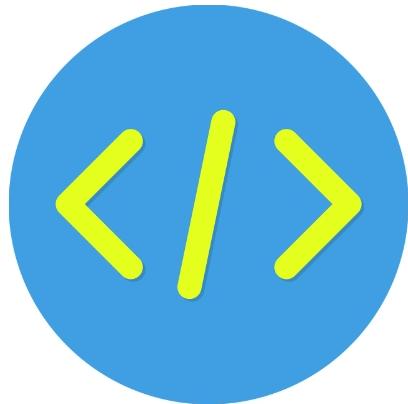
La "cascada", de las hojas de estilo en cascada, se refiere a lo que ocurre si varias fuentes de información de estilo quieren dar formato al mismo elemento de una página. Cuando un navegador encuentra un elemento para el cual hay varias declaraciones de estilo, las ordena de acuerdo al origen de la hoja de estilo, la especificidad de los selectores y el orden de la regla para poder determinar cuál aplicar.

Origen de la hoja de estilo.

Los navegadores otorgan un peso distinto a las hojas de estilo que, ordenadas de menor a mayor peso, son:

- ✓ Hojas de estilo del navegador.
- ✓ Hojas de estilo del lector.
- ✓ Hojas de estilo de la persona que ha diseñado la página web.
- ✓ Declaraciones de estilo !important del lector.

Además de este orden, existe otra jerarquía de pesos que se aplican a las hojas de estilo creadas por la persona que ha diseñado la página web. Es importante entender esta jerarquía y tener en cuenta que las reglas de estilo que están al final de la lista ignorarán a las primeras. La siguiente lista, que como la anterior está ordenada de menor a mayor peso, muestra esta otra jerarquía:



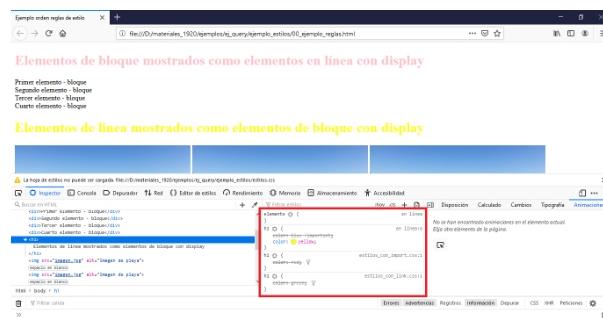
janif93 |Licencia Pixabay|

- ✓ Hojas de estilo externas vinculadas (empleando el elemento link en la cabecera del documento).
- ✓ Hojas de estilo externas importadas (empleando el elemento @import dentro del elemento style en la cabecera del documento).
- ✓ Hojas de estilo incrustadas (empleando el elemento style en la cabecera del documento).
- ✓ Estilos en línea (empleando el atributo style en la etiqueta del elemento).
- ✓ Declaraciones de estilo marcadas como !important.

A continuación se deja un código donde se aplican las diferentes formas de aplicar estilos descritas anteriormente, algunas líneas están en comentarios, para que cuando descargas los ficheros puedas manipularlos y comprobar por ti mismo la prioridad de las reglas.

### [Ejemplo de código aplicando estilos de formas diferentes.](#) (zip - 29.23 KB)

Vamos a analizar los estilos que se aplican al segundo h1 de nuestro código, para eso nos iremos a las opciones del desarrollador de nuestro navegador (normalmente pulsando F12) y se nos mostrará una pantalla como la siguiente:



Elaboración propia. Comprobando estilos en modo desarrollador. [CCO](#)

Dentro del modo desarrollador hemos seleccionado el segundo elemento h1 y vemos los estilos que se están aplicando, así observamos que la regla final que se aplica es la de asignar el color amarillo, también aparecen tachadas, las relacionadas con la hoja de estilos importada, vinculada y la regla de import, aunque esté en el código entre comentarios.

### Especificidad del selector.

Hasta ahora se tuvieron en cuenta las distintas fuentes de la información del estilo, pero aún puede existir algún conflicto a nivel de reglas. Por esa razón, "la cascada" continúa a nivel de reglas. Lo verás mejor con el siguiente ejemplo, que podría estar en una hoja de estilo externa o incrustada. En él se muestran dos reglas que hacen referencia al elemento strong.

Ejemplo:



En el ejemplo anterior, todo el texto del documento HTML marcado con la etiqueta strong aparecerá en color rojo. Sin embargo, si el texto marcado con la etiqueta strong aparece dentro de una cabecera de primer nivel (h1), su color será azul. Esto ocurre porque un elemento en un contexto determinado es más específico que en un contexto general y, por lo tanto, tiene más peso. Debes tener en claro que cuanto más específico sea el selector, se le dará más peso para ignorar las declaraciones en conflicto.

## Orden de las reglas.

Cuando una hoja de estilo contiene varias reglas en conflicto de igual peso, sólo se tendrá en cuenta la que está en último lugar. En el siguiente ejemplo, todas las cabeceras de primer nivel del documento serían rojas porque se impone siempre la última regla:

Ejemplo:



## Autoevaluación

**Ordena de menor a mayor (1-5) la prioridad con la que se aplica una regla de estilo.**

### Ejercicio de relacionar

Forma de aplicar estilos	Orden
Declaraciones de estilo marcadas como !important.	0
Hojas de estilo incrustadas	0
Hojas de estilo externas importadas con la regla @import	0
Hojas de estilo externas vinculadas con link	0
Estilos en línea	0

**Enviar**

## 1.4.- Tipos de elementos CSS

---

Los elementos en CSS, pueden dividirse en dos grandes grupos, elementos de bloque y elementos de línea, aunque la noción de "elemento de bloque" y "en línea" es puramente de presentación. Un elemento de bloque de CSS siempre genera saltos de línea, antes y después de él, mientras que los elementos en línea de CSS no lo hacen, aparecen en el flujo de la línea y sólo pasarán a otra línea si no tienen espacio.

En HTML, los párrafos (p), cabeceras (como h1), listas (ol, ul, dl) y contenedores (div) son los elementos de bloque más comunes, mientras que, el texto enfatizado (em), las anclas (a) y los elementos span son los elementos en línea más comunes.

Con CSS podrás indicarle al navegador cómo quieres que se vea en el documento empleando para ello los atributos block, inline-block e inline de la propiedad display, independientemente de que un elemento sea de bloque o en línea. El valor por defecto de la propiedad display es none.

La diferencia principal entre <code>inline e inline<code>-block, es que esta segunda opción permite definir las dimensiones (alto y ancho) del elemento.

La propiedad display permite más valores como son: list-item, table, flex y grid (estos dos últimos se verán de forma más detenida al final de la unidad). En el siguiente enlace se muestran los posibles valores que puede tener la propiedad display.

### [Valores de la propiedad display](#)

A continuación dejamos un ejemplo de código con la propiedad display donde cambiamos el comportamiento natural de los objetos div, span e img.



[Dark\\_shutterz \(Licencia Pixabay\).](#)



Se deja el código que se muestra anteriormente para que se pueda modificar y hacer diferentes pruebas.

 [Código ejemplo display](#) (zip - 29.03 KB)

## Autoevaluación

Relaciona los elementos **HTML** con el tipo de elemento que son: de bloque o en línea.

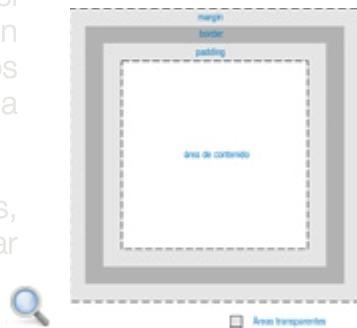
### Ejercicio de relacionar

Elemento HTML	Relación	Tipo de elemento
div	0	1- De bloque
span	0	
strong	0	
em	0	
p	0	

## 1.5.- El modelo de cajas de CSS

El modelo de cajas es un concepto fundamental para comprender el funcionamiento de las hojas de estilo. Aquí podrás ver una introducción básica a este modelo. De acuerdo con el mismo, todos los elementos de una página web generan una caja rectangular alrededor llamada "caja del elemento".

En estas cajas se pueden configurar propiedades como bordes, márgenes y fondos (entre otras). Las cajas también se pueden emplear para posicionar los elementos y diseñar la página.



Manuel Veites Rodríguez. [CC BY-NC-SA](#)

Las cajas de elementos, tal y como muestra la imagen, están hechas de cuatro componentes principales:

- ✓ **Contenido del elemento:** es lo que está en el núcleo de la caja.
- ✓ **Relleno (padding):** es el espacio que rodea al contenido.
- ✓ **Borde (border):** es la parte que perfila el relleno.
- ✓ **Margen (margin):** es el espacio que rodea al borde, la parte más externa del elemento.

Hay algunas características fundamentales del modelo de cajas que vale la pena destacar:

- ✓ El relleno, los bordes y los márgenes son opcionales, por lo que, si ajustas a cero sus valores se eliminarán de la caja.
- ✓ Cualquier color o imagen que apliques de fondo al elemento se extenderá por el relleno.
- ✓ Los bordes se generan con propiedades de estilo que especifican su estilo (por ejemplo: sólido), grosor y color. Cuando el borde tiene huecos, el color o imagen de fondo aparecerá a través de esos huecos.
- ✓ Los márgenes siempre son transparentes (el color del elemento padre se verá a través de ellos).
- ✓ Cuando definas el largo de un elemento estás definiendo el largo del área de contenido (los largos de relleno, de borde y de márgenes se sumarían a esta cantidad).
- ✓ Puedes cambiar el estilo de los lados superior, derecho, inferior e izquierdo de una caja de un elemento por separado.

En la siguiente imagen podemos apreciar desde la opción de desarrollador del navegador las dimensiones reales que tiene una imagen y las dimensiones que esta tiene o le hemos dado en nuestra página web, asignando valores a las propiedades padding, border y margin.

Al seleccionar el atributo src de la etiqueta img, nos aparecen las dimensiones de la imagen real, en nuestro caso 400x195 píxeles. Al seleccionar el elemento en la web, nos muestra el ancho y alto que tiene dicha imagen en nuestra web, como se puede apreciar en la imagen las dimensiones de la imagen que muestra son: 224x121,5 px.

Vamos a centrarnos en el ancho de la imagen en la web, concretamente 224 px. Ahora nosaremos la siguiente pregunta: si nosotros le hemos asignado a nuestra imagen un ancho de 200 px, ¿cómo es que nos muestra 224 px? Vamos a razonar este valor:

El ancho lo hemos definido mediante el atributo width y le hemos asignado: 200 px.

Con respecto al padding tenemos 10 px de padding por la derecha y 10 px de padding por la izquierda, es lo que se ve en la imagen de color morado.

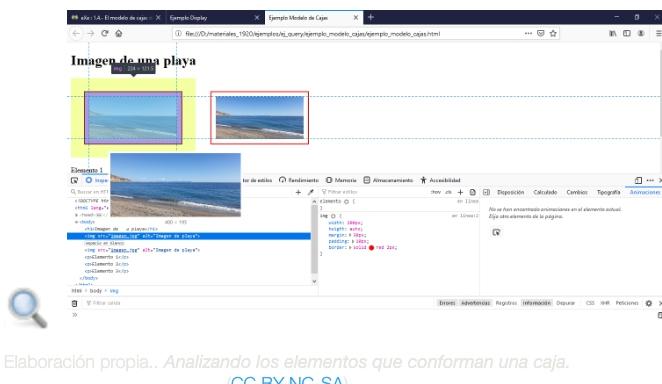
Con respecto al borde tenemos 2 px por el lado derecho y otros dos por el izquierdo, color rojo.

Si sumamos  $200+10+10+2+2$  esto nos da un total de 224 px. Valor que se nos muestra y del que hemos partido.

¿Pero este es el ancho total de la caja que contiene la imagen?

No, faltaría el margen que le hemos asignado a nuestro elemento y que podemos verlo en color amarillo. Hemos asignado 30 px de margen por la derecha y otros 30 px por la izquierda.

Con todo esto podemos decir que nuestra caja ocupará un espacio total de 284 px. (224 px + 60 px).



En los siguientes subapartados podrás ver con detalle las propiedades con las que podremos modificar la apariencia de las cajas.



## Debes conocer

En el siguiente enlace de la web de la W3C encontrarás un resumen de las propiedades CSS más utilizadas en el modelo de cajas.



[Enlace a la web de la W3C que habla sobre las propiedades CSS relacionadas con el modelo de cajas.](#)

## 1.5.1.- Área de contenido y relleno

Si comparamos el modelo de cajas con un  [Huevo Kinder](#), el área de contenido sería la sorpresa (en la imagen, el cochecito deportivo), mientras que el relleno sería la cápsula de plástico de color amarillo en la que viene la sorpresa. El chocolate sería el borde y el envoltorio de aluminio sería el margen.

### Área de contenido.

Recuerda que el área de contenido es la parte más interna de la caja. En el ejemplo siguiente se muestra cómo se pueden modificar las propiedades que afectan al tamaño del área de contenido: su ancho (width) y su altura (height).

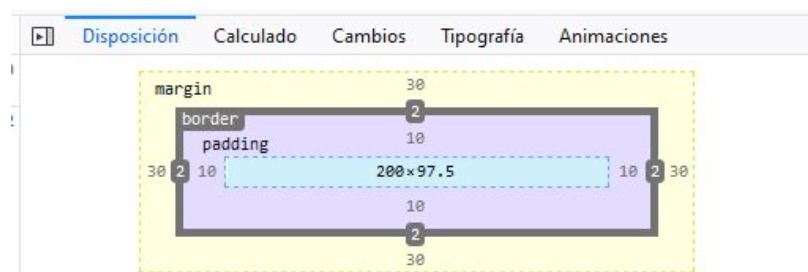
Ejemplo: `div {width:100px; height:200px; }`

Como se ha visto en el ejemplo anterior, estas propiedades son solo una parte de las dimensiones totales de la caja.

En el ejemplo anterior hemos definido para la imagen la propiedad width, pero sin embargo a la propiedad height le hemos asignado el valor auto, y esto lo que hará será asignarle un valor de forma automática para que se mantenga la relación ancho alto de la imagen original. Es lo que se muestra en la siguiente imagen, donde se puede ver el ancho y alto de nuestra caja.



A. Kriesel. [CC BY-SA](#)



The diagram illustrates the box model for a div element. It shows a large outer box representing the margin (30px), a smaller inner box representing the border (2px), and a medium-sized inner box representing the padding (10px). The innermost area, labeled 'content-box', has dimensions of 200x97.5px. The total width of the element is 224px and the total height is 121.5px. The 'content-box' is also labeled as 'static'.

Elaboración propia. [Elementos del modelo de cajas. CC BY-SA](#)

Además de las propiedades width y height, podemos utilizar otras propiedades como son:

- ✓ min-width: asignamos un valor mínimo de ancho a nuestro elemento, es decir, el ancho del elemento puede cambiar pero nunca puede ser inferior a este valor dado.
- ✓ max-width: asignamos un valor máximo al ancho de nuestro elemento, así este podrá crecer pero nunca podrá superar el valor especificado.

Igual pasaría con las propiedades: min-height y max-height.

Estas propiedades podrán modificar los valores especificados en width y height, ¿cómo puede ser esto?

Inicialmente yo asigno unas dimensiones al elemento pero si este se "reescala" por ejemplo al ampliar o reducir el tamaño del navegador, las dimensiones del elemento pueden cambiar para adaptarse a la nuevas dimensiones. Esto veremos que puede ocurrir cuando definamos los elementos con medidas relativas (%) en lugar de medidas absolutas (px).

## Relleno.

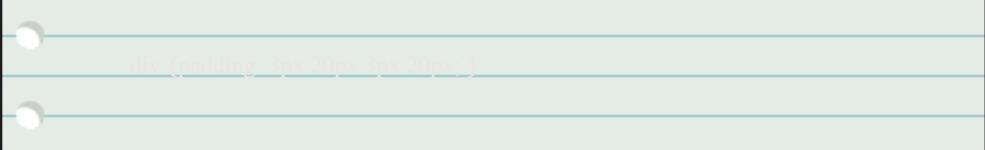
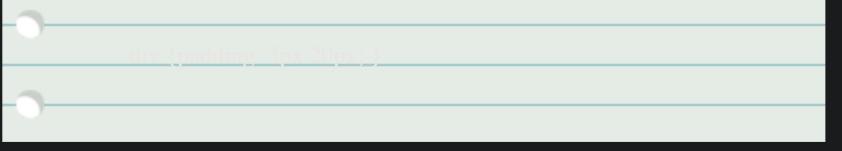
El relleno es una cantidad opcional de espacio existente entre el área de contenido de un elemento y su borde. Es conveniente que establezcas un valor de relleno cuando pones borde a un elemento.

Para establecer el relleno se emplea la propiedad padding. Esta propiedad, como muchas otras en CSS, obliga a configurar los valores en un orden determinado. Estos valores y su orden son:

- ✓ top (arriba).
- ✓ right derecha.
- ✓ bottom (debajo).
- ✓ left (izquierda).

El ejemplo siguiente muestra una tabla que agrupa algunos ejemplos de la asignación de valores y su interpretación por CSS. En todos los ejemplos se ha empleado como selector el elemento div.

## Ejemplos de asignación de valores a la propiedad padding.

EJEMPLO	INTERPRETACIÓN
 <code>div {padding: 3px 20px 3px 20px;}</code>	Establece un relleno para todos los elementos div de 3 píxeles por encima del área de contenido, 20 píxeles a su derecha, 3 píxeles por debajo y 20 píxeles a su izquierda.
 <code>div {padding: 3px 20px 3px 3px;}</code>	Al omitir un valor, asume que el valor del relleno a la izquierda es el mismo que el de la derecha.
 <code>div {padding: 3px 3px 3px 20px;}</code>	Al omitir dos valores, asume que el primer valor corresponde al relleno por encima y por debajo del área de contenido y, el segundo valor corresponde al relleno a la derecha y a la izquierda.
 <code>div {padding: 3px 3px 3px 3px;}</code>	Al omitir tres valores, asume que ese valor es el mismo para todos.

Otras características interesantes del relleno son:

- ✓ El valor del relleno se sumará al de width ya definido en el elemento.
- ✓ Su color es el mismo al del área de contenido.
- ✓ El relleno nunca se "colapsa". Esto lo entenderás cuando veas los márgenes que sí se colapsan.

En el siguiente ejemplo se muestra cómo configuramos el relleno y el color de fondo del área de contenido de los elementos `<h1>` del documento. El color del área de relleno será el mismo que el del elemento.

Ejemplo: `<h1>h1 {padding: 4px 10px; background: #ccc; }</h1>`

# Autoevaluación

¿El padding es el margen exterior del objeto?

- Verdadero
- Falso

## 1.5.2.- Bordes

Un borde es una línea dibujada alrededor del área de contenido de un elemento y de su relleno (padding), aunque ya vimos que éste último era opcional.

**Los bordes funcionan, a la hora de establecer sus valores, de la misma manera que el relleno visto anteriormente, siguiendo un orden: superior, derecho, inferior, izquierdo, siempre en el sentido de las agujas del reloj y comenzando en las 12. Es fácil de recordar.**

**Se pueden establecer valores distintos para cada uno de los bordes y omitir valores, al igual que hacíamos con el relleno.**

Podemos configurar el estilo del borde, su grosor y su color. Las propiedades que nos permiten hacerlo son:

- ✓ **border-style:** con esta propiedad configuramos el estilo del borde. Esta propiedad es, sin duda, la más importante del borde, ya que, si no está presente el borde no existirá. La propiedad border-style puede tener los valores: none, hidden, dotted, dashed, solid, double, groove, ridge, inset, outset e inherit. En el siguiente ejemplo configuramos cada uno de los lados de la caja con un borde distinto.

Ejemplo: `div {border-style: solid dashed dotted double; }`



[steveczajka. CC BY](#)

- ✓ **border-width:** con esta propiedad configuramos el grosor del borde. Los valores que puede tomar esta propiedad son: thin, medium, thick, inherit o un valor concreto en píxeles. Si no se especifica esta propiedad tomará medium como valor por defecto. En el siguiente ejemplo configuramos un grosor distinto en cada uno de los lados del borde.

Ejemplo: `div {border-style: solid; border-width: thin medium thick 12px; }`

- ✓ **border-color:** con esta propiedad configuramos el color del borde. Si no especificamos el color el elemento coge el del "primer plano", es decir, que si por ejemplo, tenemos una caja en cuyo interior hay texto, el color del borde será el color del texto.

Existe el color transparent pero no está soportado por todos los navegadores. En el siguiente ejemplo especificamos un estilo y un ancho igual para todos los bordes y un color distinto para cada borde.

Ejemplo: `div {border-style: solid; border-width: 4px; border-color: #333 #red rgb(0,0,255) #0044AC; }`

Para la asignación de valores a los bordes se sigue la misma estrategia que se ha descrito en el apartado anterior referente al <code>padding. Podemos con un valor asignar los valores de todos los bordes, los bordes horizontales y verticales o dar un valor específico para cada borde. Esto lo veremos más adelante con algunos ejemplos.

En el siguiente código aplicamos algunos de los valores a las propiedades vistas:

```
div {  
    border: 1px solid black;  
    width: 400px;  
    height: 200px;  
    padding: 50px;  
    background-color:  
        border-style: solid dashed dotted double;  
        border-color: red green blue yellow;  
        border-width: thin medium thick 5px;  
    }  
    
```

El resultado de aplicar los estilos anteriores a un div es el siguiente:



Elaboración propia. Caja con bordes diferentes. [CC BY-SA](#)

La propiedad border une todas las propiedades "border" vistas hasta ahora. En esta propiedad, a diferencia de las anteriores, no hay que colocar los valores en ningún orden concreto. La propiedad border se emplea cuando se quieren configurar los cuatro lados por igual.

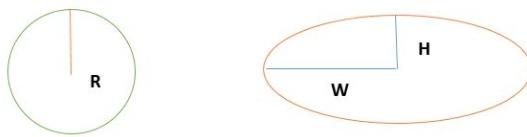
También tenemos las propiedades: border-top, border-right, border-bottom y border-left.

Como hemos visto las anteriores propiedades definen las características de un borde de un lado, pero se puede utilizar también con los estilos, grosor y color y así tendríamos: border-top-style, border-top-color, border-top-width, etc.

Ejemplos:

```
b1: border: 2px solid blue;  
b2: border-left: solid blue 2px;  
b3: border-top: 1px solid red;
```

Con la llegada de [CSS3](#) podemos modificar el comportamiento de los bordes de un objeto. Para ello utilizaremos la propiedad `border-radius`. Esta propiedad nos permitirá dar forma redondeada a los bordes, yendo desde una forma circular a una forma elíptica. En la siguiente imagen podemos ver las dos formas que tenemos para redondear un borde.



Elaboración propia. *Forma de redondear un borde.* [CC BY-SA](#)

Cuando aplicamos esta propiedad podemos jugar con cada una de las esquinas, así podemos definir los elementos `border-top-left-radius`, `border-top-right-radius`, `border-bottom-left-radius` y `border-bottom-right-radius`.

<br /> En el siguiente ejemplo se muestra como se ha aplicado diferentes valores a cada una de las diferentes propiedades. En estos ejemplos estamos aplicando bordes redondeadas, cuando aplicamos el radio con el que queremos deformar la esquina (imágenes de la columna izquierda) y también bordes elípticos (columna derecha de la imagen), que lo explicaremos en el siguiente párrafo.

Modificación de bordes con un radio.	Modificación de bordes con dos radios.
<b>Bordes circulares.</b>	<b>Bordes elípticos.</b>
<code>border-radius: 10px;</code>	<code>border-radius: 40px 100px;</code>
<code>border-top-left-radius: 10px;</code>	<code>40px 0px 0px 100px;</code>
<code>border-top-right-radius: 40px;</code>	<code>border-radius: 0px 20px 0px 0px; 50px;</code>
<code>border-bottom-left-radius: 50px;</code>	<code>border-radius: 0px 0px 50px 0px;</code>
<code>border-bottom-right-radius: 80px;</code>	<code>border-radius: 0px 0px 0px 100px; 50px;</code>



Elaboración propia. *Propiedad Bordes-radius.* [CC BY-SA](#)

Para realizar una deformación elíptica tendremos que aplicar un valor para el radio de la altura (H) y otra para el radio de la anchura (W), así a cada una de las propiedades le podemos aplicar estos valores. En la imagen anterior se muestran diferentes valores para esta propiedad. Destacar como se indican los diferentes valores de las esquinas en los dos radios. Se sigue la misma asignación de valores que hemos visto en el padding.



## Para saber más

Otra posibilidad que nos permite la propiedad border es definir una imagen como borde. Te proponemos un conjunto de enlaces para profundizar un poco más sobre esta forma de aplicar bordes.

 [border-image.](#)

Para aplicar esta propiedad de forma correcta hay que tener también en cuenta las propiedades:

 [border-image-slice.](#)

 [border-image-width.](#)

 [border-image-outset.](#)

 [border-image-repeat.](#)

## 1.5.3.- Márgenes

---

El margen es la cantidad de espacio que se puede añadir alrededor del borde de un elemento.

Esta propiedad se configura con la propiedad margin. Al configurar esta propiedad debemos tener en cuenta que, a la hora de establecer los valores para los márgenes, hay que emplear la misma filosofía que con la propiedad padding.

Los márgenes top y bottom de dos elementos que van seguidos se " colapsan". Es decir, se asume como margen entre ambos elementos el mayor de ellos. El siguiente ejemplo muestra lo que ocurre cuando tenemos dos elementos un h1 y un h2 colocados uno a continuación del otro.

Ejemplo:



En el primer caso el margen superior e inferior es de 10 px. En el segundo caso es de 20 px. El espacio resultante entre los dos elementos será de 20 px.

Por el contrario, si fuesen dos elementos colocados "uno al lado del otro" (como dos elementos span), deberíamos tener en cuenta que los márgenes right y left no se colapsan, sino que se suman.

Si en las propiedades horizontales left y right de un elemento le asignamos el valor auto, éste se centrará de forma automática dentro del objeto en el que esté contenido.

A continuación dejamos un vídeo donde se puede ver como los márgenes superior e inferior de dos elementos de bloque se superponen y los márgenes izquierdo y derecho de dos elementos en línea se suman.



guinn.anya. (CC BY-SA)

# Autoevaluación

¿En qué orden se dan las medidas de los márgenes en CSS?

- top bottom right left
  - left right top bottom
  - top left right bottom
  - top right bottom left



## 1.5.4.- Box-sizing

---

Con lo que conocemos hasta este momento, si queremos calcular el ancho y alto de una caja de nuestra página web, tendremos que saber las dimensiones del área de contenido, sumar el relleno lateral (padding), la anchura de los bordes y el ancho de los márgenes. Esto es un poco engorroso y muchas veces tenemos que realizar demasiados cálculos para saber las dimensiones reales de un objeto. Para facilitar nuestra labor y no tener que realizar tantos cálculos se ha desarrollado una nueva propiedad denominada box-sizing.

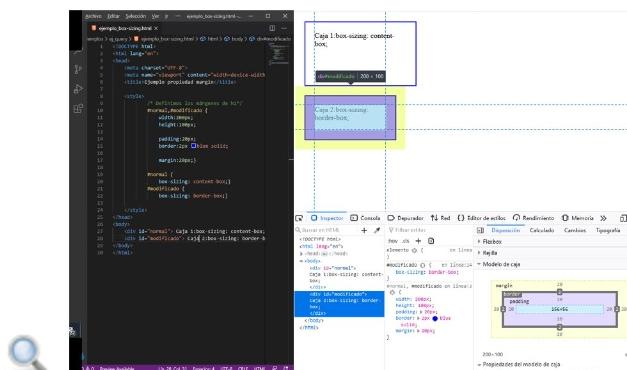
### ¿Qué nos permite box-sizing?

Que cuando nosotros definamos el ancho y el alto de un objeto con las propiedades `width` y `height`, éstas incluyen todo lo referente al relleno lateral y el borde.

A la propiedad box-sizing le podemos asignar los siguientes valores:

- ✓ `content-box`: modelo tradicional como se ha visto en los puntos anteriores donde para saber el tamaño total de la caja tendremos que tener en cuenta el área de contenido, la zona de relleno, el borde y el margen.
- ✓ `border-box`: cuando definimos el ancho y el alto que correspondería a la zona de contenido, se incluye también el espacio del padding y el espacio del border. Para calcular las dimensiones totales de nuestra caja habría que sumarle el tamaño del margin.
- ✓ `inherit`: hereda el valor del parent.

A continuación mostramos un ejemplo de como funciona dicha propiedad.



Elaboración propia. Propiedad box-sizing. (CC BY-SA)

[Código ejemplo.](#) (zip - 0.7 KB)

Si queremos que todos los objetos de nuestra página se comporten de esta forma, tendremos que utilizas el selector universal `*` de la siguiente forma:



Amit Agarwal CC BY

# Autoevaluación

**Cuando definimos la propiedad width de un objeto al que se le ha asignado la propiedad box-sizing con valor border-box, este valor incluye...**

- La zona de contenido.
- La zona de contenido + la zona de relleno.
- La zona de contenido + la zona de relleno + el borde.
- La zona de contenido + la zona de relleno + el borde + el margen.



## Para saber más

En caso de que no utilicemos la propiedad box-sizing es posible que tengamos que realizar cálculos para ajustar bien nuestras cajas, para facilitar esta labor podemos utilizar la función:



Con la llegada de [CSS3](#) también se permite el uso de variables



## 2.- Selectores



### Caso práctico



Elaboración propia (Uso Educativo no comercial).

Después de unos días estudiando CSS, **Carlos** ya tiene claro la estructura de las reglas de estilo que componen los archivos CSS.

**Carlos** también se da cuenta que una parte muy importante de estas reglas son los selectores y, como está empezando, decide pedir ayuda a **Juan**, que ya lleva más de cuatro años desarrollando aplicaciones web.

**Juan** le explicará cómo debe utilizar los selectores para aprovechar todo el potencial que tienen y, entre los dos, completarán el manual de CSS para poder consultarlos en cualquier momento que lo necesiten mientras realizan la aplicación web.

El **selector** es la parte de la regla de estilo que identifica el elemento concreto al que se aplicarán las instrucciones de presentación. [CSS](#) ofrece varios tipos de selectores que permiten mejorar la flexibilidad y la eficiencia en la creación de hojas de estilo.

Como se ha comentado anteriormente las especificaciones sobre los diferentes elementos que conforman las hojas de estilo, están en continua evolución, una parte importante de éstas son los selectores, así está como recomendación el nivel 3 y ya se está trabajando en el documento de selectores nivel 4.



### Para saber más

Se dejan como referencia los enlaces a la documentación proporcionada por la [W3C](#) sobre selectores.

[Selectores Nivel 3 \(estándar\).](#)

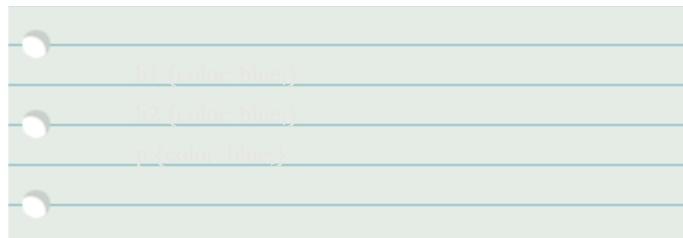
[Selectores Nivel 4 \(documento de trabajo\).](#)

## 2.1.- Selectores de elemento

---

Los selectores de elemento son los más sencillos. Son aquellos que se aplican a un elemento (etiqueta) del lenguaje HTML.

Ejemplos:



[webmove](#) CC BY

Si te fijas en los ejemplos, verás que se está definiendo la misma propiedad (color) en todos los elementos e incluso se está asignando el mismo valor (blue). El ejemplo siguiente muestra cómo se puede escribir una única regla aplicada a varios selectores a la vez.

Ejemplo: h1, h2, p {color: blue;}

Existe un selector de elementos "universal" representado por el asterisco (\*). El ejemplo siguiente muestra una regla que pondrá en gris todos los elementos del documento.

Ejemplo: \* {color: grey;}

Habitualmente el selector universal se suele poner al principio de nuestra hoja de estilos.

Como se ha visto en un apartado anterior, si quisiéramos aplicar a todos los elementos de nuestra página la propiedad box-sizing con el valor border-box, tendríamos que realizar lo siguiente:

```
<code>* {box-sizing: border-box;}
```

**Cuando se realiza una declaración sobre varios selectores a la vez, éstos se separan por comas.**

## 2.2.- Selectores contextuales

Como vimos, los **selectores de elemento** se aplican a todos los casos en los que se encuentre el elemento en el documento HTML. En cambio, los selectores contextuales permiten aplicar estilos a los elementos basándose en su contexto o en su relación con otro elemento.

Hay varios tipos de selectores contextuales: **descendente, hijo y hermano.**

Los **selectores descendentes** hacen referencia a elementos que están contenidos en otro elemento. Un selector descendente se pone a continuación del selector en el que está contenido separado de él por un espacio en blanco. El siguiente ejemplo especifica que los elementos em deben tener color azul, pero sólo si son descendientes de un elemento de lista (li). El resto de los elementos em no se verán afectados.

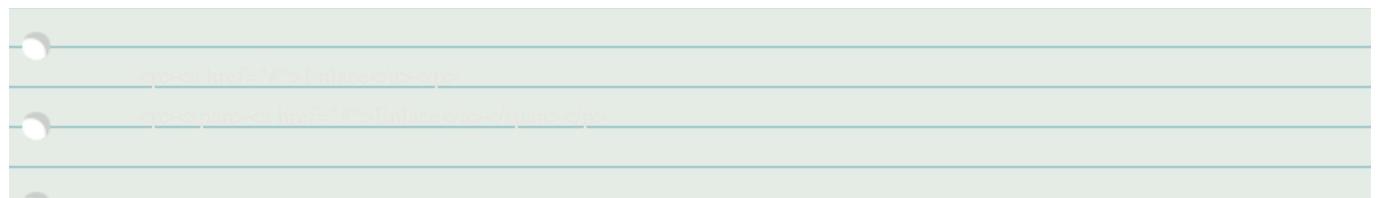
Ejemplo: li em {color: blue;}

Los selectores descendentes también pueden estar anidados en varias capas de profundidad. El siguiente ejemplo pone de color amarillo sólo el texto enfatizado (em) de las anclas (a) que se encuentren en las listas ordenadas (ol).

Ejemplo: ol a em {color: yellow;}

Si se emplea el selector descendente combinado con el selector universal, se puede restringir el alcance de un selector descendente. El siguiente ejemplo muestra dos párrafos que contienen un hipervínculo. En el primer caso el elemento ancla (a) es descendiente directo del elemento de párrafo (p) y, en el segundo caso, es descendiente directo del elemento span que, a su vez, lo es del elemento de párrafo (p).

Ejemplo:



Examinemos las dos reglas de estilo siguientes:



```
/* Descendentes */
li em {color: blue;}
h1 em, h2 em, h3 em {background-color: red;}
ol a em {color: yellow;}
p a { font-size: 0.5em; }
p * a { color: red; }

/* Hijo */
p > em {background-color: gray;}

/* Hermano */
h1 + h2 { color: pink; }
```

Manuel Vieites Rodríguez. Selectores. [CC BY-NC-SA](#)

**Código ejemplo (Haz clic para visualizarlo)**



**Resultado del código ejemplo (Haz clic para visualizarlo)..**

## Contenido Principal: MAIN (etiqueta h2)

Esto es un ejemplo de selectores `+ (soy span) > (soy span) ~ (soy span)` (etiqueta h3)

## Contenido (etiqueta h2)

### Artículo 1: ARTICLE (etiqueta h3)

Lista de elementos (la palabra elementos es una etiqueta span) (etiqueta p)

- Primero
  - Segundo
  - Tercero
  - Cuarto

Fin de la lista de elementos (etiqueta p)

Texto (soy un span) en azul (etiqueta p)

## Contenido (etiqueta `nz`)

#### **Artículo 2: ARTICLE (etiqueta h3)**

#### Lista de elementos (etiqueta p)

- A
  - B
  - C
  - D

Texto en azul (etiqueta p)



Pasamos a explicar de forma más detenida cada una de las reglas CSS que hemos utilizado.

```
section article {  
    border: solid red 2px;  
}
```

Esta regla va a poner borde rojo a todos los elementos `article` que estén contenidos dentro de un elemento `section`, da igual que estén justamente después de `section` o que haya otros elementos entre la etiqueta `section` y `article` (concepto de hijo).

```
.listado_01 {  
    color: green;  
}
```

Esta regla utiliza el concepto de clase (se verá en el punto siguiente) que nos permitirá seleccionar aquellos elementos definidos con dicha clase y ponerlos de color verde, así a todas las etiquetas `li` que estén contenidas dentro de la clase `.lista_01`, se le asignará este color.

```
article p {  
    color: blue;  
}
```

Esta regla pondrá de color azul todas las etiquetas `p` que sean hermanos de una etiqueta `article`, da igual la posición que compartan, es decir, que estén justamente después o algunas etiquetas después, son hermanos y comparten padre. Así, las etiquetas `p` que están dentro de `article` no están de color azul, como puede verse en el ejemplo (concepto de hermano).

```
h2 ~ h3 {  
    color: orange;  
}
```

Esta regla nos va a poner de color naranja todas las etiquetas `h3` que vengan justamente después de una etiqueta `h2`. En nuestro ejemplo, solo se pondrá de color naranja la primera etiqueta `h3`, ya que es la única que viene justamente después de `h2` (concepto de hermano adyacente).

```
h3 ~ span {  
    color: aqua;  
}
```

Esta regla nos pondrá de color agua todas las etiquetas `span` que sean hijos de una etiqueta `h3`, en nuestro ejemplo esto se traduce en que asignará dicho color a los caracteres `±, ≥` y `≈` (contenidos en etiquetas `span`), pero sin embargo el resto de etiquetas `span` no las cambia de color.

Como puedes comprobar el número de posibilidades que se nos presenta es innumerable.

# Autoevaluación

Tenemos el siguiente documento HTML con una hoja de estilo incrustada:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Autoevaluación</title>
    <style>
      h1 {color: red;}
      h2 {color: green;}
      p {color: blue;}
      .sub {color: orange;}
      .sub2 {color: purple;}
    </style>
  </head>
  <body>
    <h1>Título 1</h1>
    <h2>Subtítulo 1</h2>
    <p>Este es el primer párrafo</p>
    <h3>Subtítulo 1.1</h3>
    <h3>Subtítulo 1.2</h3>
    <p>Este es el segundo párrafo</p>
    <h3>Subtítulo 2.1</h3>
    <h3>Subtítulo 2.2</h3>
    <p>Este es el tercer párrafo</p>
  </body>
</html>
```

Al abrir este documento en un navegador ¿qué texto aparecerá en color rojo?

- Título 1.
- Subtítulo 1.
- Párrafo 1
- Subtítulo 2.
- Párrafo 2.
- Subtítulo 1 y Subtítulo 2.
- Párrafo 1 y Párrafo 2.

## 2.3.- Selectores de clase e ID

Para poder hacer uso de selectores más específicos, se hace necesario introducir los conceptos de identificador (id) y clase (class).

### Identificador (id).

Los elementos HTML disponen de un atributo llamado identificador (id), que tiene como finalidad identificar al elemento de manera exclusiva. De este modo, CSS u otro lenguaje podrá hacer referencia a él y distinguirlo del resto de los elementos del documento.



Jim Bahn. Contenedores. (CC BY)

**Un id debe ser único en cada documento HTML.**

Ejemplo:

● `<div id="textocabecera">Este es el texto de la cabecera</div>`

Se recomienda que el valor del id sea un nombre que caracterice o clarifique, de forma breve y esquemática al elemento y que, además, sea fácilmente reconocible por el programador. Se utilizan con frecuencia para identificar las secciones principales de las páginas: contenido, cabecera, pie, etcétera.

Para escribir una regla de estilo que se aplique a un determinado identificador hay que escribir el símbolo de la almohadilla (#) seguido del nombre del identificador. El ejemplo siguiente muestra algunas formas de establecer el tamaño de la fuente en 14 píxeles al elemento p identificado como "textocabecera" del ejemplo anterior:

● `#textocabecera {font-size: 14px;}`

● `.textocabecera {font-size: 14px;}`

Con la primera regla indicamos que se aplique el estilo a un párrafo cuyo identificador sea "textocabecera" pero, como el id es único en cada documento, realmente basta con la segunda forma para decir lo mismo, porque no va a haber otro elemento <p> o diferente de <p> que tenga ese mismo identificador.

Si tenemos varios elementos que necesitan un tratamiento similar, emplearemos el atributo class.

### Clases (class).

Se emplea el atributo class para identificar distintos elementos como parte de un grupo conceptual. Así, los elementos de una clase pueden modificarse con una única regla de estilo.

En el siguiente ejemplo se muestra cómo dos elementos distintos se clasifican de la misma forma mediante la asignación del valor "especial" al atributo class.



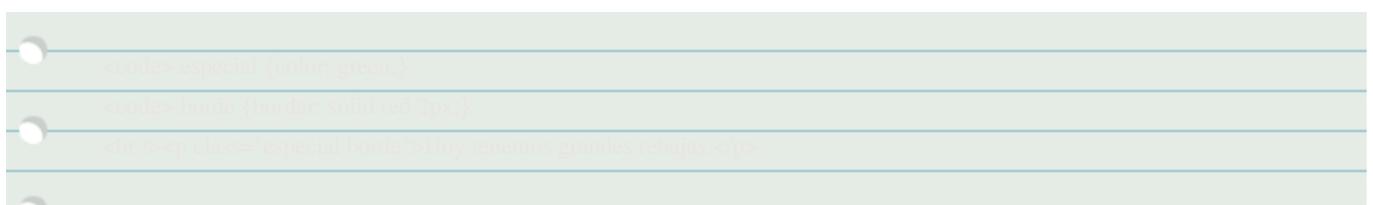
Para escribir una regla de estilo que se aplique a todos los elementos de una determinada clase hay que escribir un punto seguido del nombre de la clase. Por ejemplo: .especial {color: green;}

El siguiente ejemplo muestra la forma de lograr que todos los elementos de la clase "especial" tengan un color verde a excepción de las cabeceras de primer nivel que tienen que ser rojas.

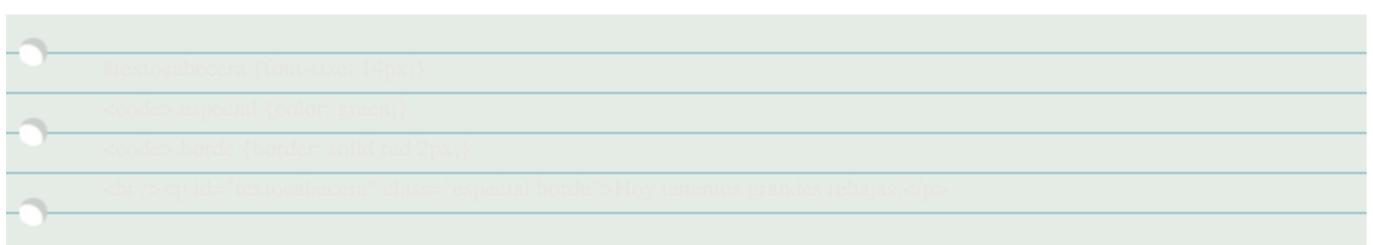


Hemos visto cómo asignar una clase a un elemento, pero ¿es posible que un mismo elemento se le pueda asignar más de una clase?

La respuesta es sí. Para realizar esto, lo único que hay que hacer es poner las clases que queremos asignar a dicho elemento separadas por un espacio. En el siguiente ejemplo el elemento párrafo pertenecerá a dos clases: "textocabecera" y "borde".



Por último tendremos que ver si un elemento puede tener asignado una clase (o más) y a la vez un identificador; para conseguir esto solo habrá que añadir dichos selectores al elemento. A continuación presentamos un ejemplo combinado clases y un identificador.



**Los nombres de clases y de identificadores no pueden contener espacios en blanco.**



## Recomendación

En el siguiente enlace podrás leer unas recomendaciones generales sobre CSS. Son especialmente interesantes aquellas que mencionan las reglas que deben cumplir los nombres de los identificadores y de las clases y cómo se debe estructurar una hoja de estilos.



[Recomendaciones generales sobre CSS.](#)

## 2.4.- Otros selectores

Los selectores vistos hasta el momento son los más conocidos y quizás los más importantes, pero existen otros muchos más, algunos de los cuales vamos a estudiar en este apartado. Como se ha comentado al inicio de este capítulo, las hojas de estilo están en permanente cambio, incorporando nuevos elementos y ampliando la funcionalidad de los que ya existen.

### Selectores de atributo.

Este tipo de selectores nos permitirá seleccionar elementos tanto por el nombre del atributo como por el valor del mismo, por lo tanto los dividiremos en dos grupos:

- ✓ Selectores por nombre de atributo.
- ✓ Selectores por valor de atributo.

### Selectores por nombre de atributo.

Para utilizar este selector bastará con poner el nombre del atributo entre corchetes.

[atributo]

Con este ejemplo seleccionaremos todos los elementos que tengan un identificador asignado (id).



Con este ejemplo seleccionaremos todos los elementos de tipo img que tenga atributo alt.



### Selectores por valor de atributo.

[atributo="valor"]

En este ejemplo seleccionaremos aquellos elementos tengan una clase denominada principal.



En este ejemplo se seleccionarán las imágenes cuya descripción alternativa textual (alt) sea exactamente "Paisaje Playa".

• `img[alt="Paisaje Playa"]`

Dentro de este apartado tenemos diferentes posibilidades más para filtrar el valor del atributo, indicando aspectos como que el valor del atributo: empiece por...; que termine por...; que contenga un valor determinado; etcétera.

- ✓ `[atributo^="valor"]` Seleccionaremos aquellos elementos cuyo atributo comienza con la palabra valor.
- ✓ `[atributo$="valor"]` Seleccionaremos aquellos elementos cuyo atributo termina con la palabra valor.
- ✓ `[atributo*="valor"]` Seleccionaremos aquellos elementos cuyo atributo contenga la palabra valor.
- ✓ `[atributo~="valor"]` Este selector nos permite seleccionar aquellos elementos cuyo atributo contenga la palabra valor aunque esté separada por espacios en blanco. Se seleccionan elementos que tengan atributos con varias palabras. Ejemplo:

• `CSS: <img alt="Playa" data~="playa"> color: red; font-size: 10px; width: 50px; height: 50px;`

## Atributos de datos (atributos personalizados)

Para terminar este apartado y aunque no sea un selector, vamos a indicar cómo HTML5 nos permite ampliar los atributos de un elemento dándonos la posibilidad de definir nuestros propios atributos, es lo que se denomina como atributos `data-*`. Estos atributos nos permitirán almacenar la información que nosotros queramos.

Para definir un atributo de este tipo (de datos) tendremos que escribir el prefijo `data-` y el nombre que queramos (no debe contener punto y coma y no debe contener mayúsculas): `data-info`.

A continuación vamos a definir un atributo de datos en un elemento cualquiera, por ejemplo:

• `<img data-posicion="principal">`

De esta forma hemos definido un atributo personalizado, teniendo en cuenta que tenemos selectores que nos permiten seleccionar elementos por nombre de atributo, se nos abre de forma indirecta una nueva posibilidad de selección de elementos. Ejemplo:

• `data-posicion="principal"`

color: blue;

/

Seleccionaremos aquellos elementos cuyo atributo `data-posicion` (definido por nosotros) tenga el valor igual a `principal`.

## 2.5.- Pseudoselectores: Pseudoclases

---

Si queremos aplicar reglas de estilo a elementos especiales como los vínculos visitados, la primera línea de un párrafo o su primera letra, emplearemos los **pseudoselectores**.

Hay dos tipos de pseudoselectores:

- ✓ **pseudoclases.**
- ✓ **pseudoelementos.**

En los siguientes puntos definiremos de forma más detallada estos pseudoselectores y veremos algunos ejemplos. Por dar una idea general podemos decir que las pseudoclases nos permiten seleccionar elementos en función de su estado y los pseudoelementos nos permiten actuar sobre una parte determinada de un elemento.



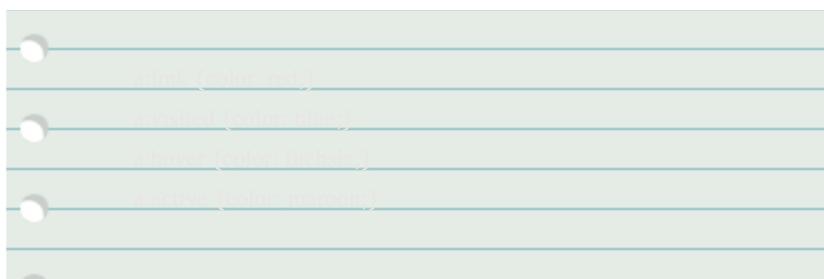
Ciker-Free-Vector-Images | Licencia Pixabay

### Pseudoclases.

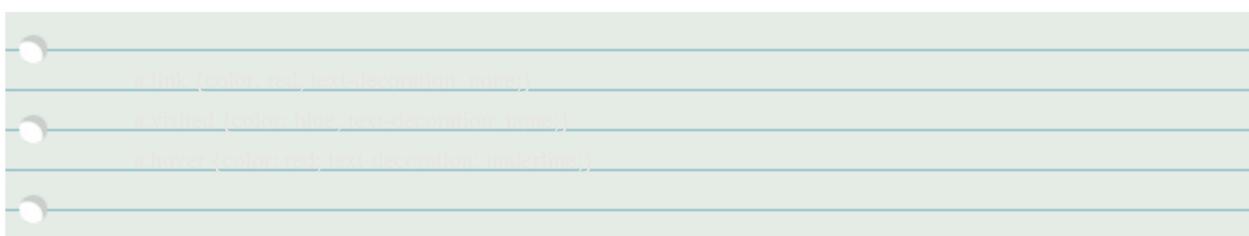
Clasifican a los elementos basándose en características que van más allá de su nombre, atributos o contenido, es decir, nos permitirán poder **seleccionar elementos en función del estado** que estos tengan.

Quizá las pseudoclases más conocidas sean las que afectan a los elementos ancla ([a](#)), pudiendo definir un estilo diferente en función del comportamiento del enlace: si todavía no ha sido visitado (`link`), si ya lo ha sido (`visited`), mientras el ratón pasa por encima (`hover`) o justo cuando se pulsa el ratón sobre él (`active`).

Para emplear una pseudoclase se escribe la misma a continuación del selector separándola de éste por el símbolo de dos puntos (:). El ejemplo siguiente muestra cómo se distinguirían los cuatro estados posibles de un enlace mediante colores diferentes.



Con el empleo de estas pseudoclases podemos quitar el subrayado de los hiperenlaces y hacer que aparezca sólo cuando pasamos el puntero por encima. El ejemplo siguiente muestra el empleo de la propiedad `text-decoration` para conseguir dicho objetivo.



**Debes tener en cuenta que las pseudoclases ancla deben aparecer siempre en un determinado orden. Este orden es:**

**:link, :visited, :hover y :active.**

Por si te ayuda, para recordarlo, se emplean las iniciales: LVHA.

## Código ejemplo 1.

```
    /* Seleccionamos el último elemento de la lista */
    li:last-child {
        border-bottom: 1px solid black;
    }

    /* Todos los div le asignamos un valor */
    div {
        width: 300px;
        height: 100px;
        border: 1px solid blue;
        margin-top: 10px;
    }

    /* Seleccionamos aquellos elementos con clase una en la secuencia 1,2,3 comenzando por el primero */
    .color1, .color2, .color3 {
        width: 300px;
        height: 100px;
        border: 1px solid black;
        margin-top: 10px;
    }

    /* Selección de los elementos que tienen una clase con la secuencia 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96,97,98,99,100 */
    .color1, .color2, .color3, .color4, .color5, .color6, .color7, .color8, .color9, .color10, .color11, .color12, .color13, .color14, .color15, .color16, .color17, .color18, .color19, .color20, .color21, .color22, .color23, .color24, .color25, .color26, .color27, .color28, .color29, .color30, .color31, .color32, .color33, .color34, .color35, .color36, .color37, .color38, .color39, .color40, .color41, .color42, .color43, .color44, .color45, .color46, .color47, .color48, .color49, .color50, .color51, .color52, .color53, .color54, .color55, .color56, .color57, .color58, .color59, .color60, .color61, .color62, .color63, .color64, .color65, .color66, .color67, .color68, .color69, .color70, .color71, .color72, .color73, .color74, .color75, .color76, .color77, .color78, .color79, .color80, .color81, .color82, .color83, .color84, .color85, .color86, .color87, .color88, .color89, .color90, .color91, .color92, .color93, .color94, .color95, .color96, .color97, .color98, .color99, .color100 {
        width: 300px;
        height: 100px;
        border: 1px solid green;
        margin-top: 10px;
    }

    /* Añadimos estilos a la lista */
    li {
        list-style-type: none;
        padding-left: 0px;
        color: red;
    }

    /* Dentro de la lista que seleccionamos los cuállos píxeles de los bordes están en el color rojo */
    li:after {
        content: " ";
        width: 10px;
        height: 10px;
        border: 1px solid red;
        border-radius: 50%;
        display: inline-block;
        vertical-align: middle;
    }

    /* Estilos */
    .style {
        color: blue;
        font-size: 1.2em;
        font-weight: bold;
    }

    .style2 {
        color: green;
        font-size: 1.2em;
        font-weight: bold;
    }

    .style3 {
        color: red;
        font-size: 1.2em;
        font-weight: bold;
    }
```

• Clase de los primeros elementos

• Clase de los segundos elementos

• Clase de los terceros elementos

• Clase de los cuartos elementos

• Clase de los quintos elementos

• Clase de los sextos elementos

• Clase de los séptimos elementos

• Clase de los octavos elementos

• Clase de los novenos elementos

• Clase de los decimos elementos

• Clase de los undécimos elementos

• Clase de los duodécimos elementos

• Clase de los treceños elementos

• Clase de los catorceños elementos

• Clase de los quinceños elementos

• Clase de los dieciséis elementos

• Clase de los diecisiete elementos

• Clase de los dieciochoños elementos

• Clase de los diecinueveños elementos

• Clase de los veinteños elementos

• Clase de los veintiún elementos

• Clase de los veintidós elementos

• Clase de los veintitrés elementos

• Clase de los veinticuatroños elementos

• Clase de los veinticincoños elementos

• Clase de los veintiseis elementos

• Clase de los veintisiete elementos

• Clase de los veintiochoños elementos

• Clase de los veintinueveños elementos

• Clase de los treinta elementos

• Clase de los treinta y uno elementos

• Clase de los treinta y dos elementos

• Clase de los treinta y tres elementos

• Clase de los treinta y cuatro elementos

• Clase de los treinta y cinco elementos

• Clase de los treinta y seis elementos

• Clase de los treinta y siete elementos

• Clase de los treinta y ocho elementos

• Clase de los treinta y nueve elementos

• Clase de los cuarenta elementos

• Clase de los cuarenta y uno elementos

• Clase de los cuarenta y dos elementos

• Clase de los cuarenta y tres elementos

• Clase de los cuarenta y cuatro elementos

• Clase de los cuarenta y cinco elementos

• Clase de los cuarenta y seis elementos

• Clase de los cuarenta y siete elementos

• Clase de los cuarenta y ocho elementos

• Clase de los cuarenta y nueve elementos

• Clase de los cincuenta elementos

## Resultado para el código del ejemplo 1.

### • Enero

- Febrero
- Marzo
- Abril
- Mayo
- Junio

Primer elemento

Segundo elemento

Tercer elemento

Cuarto elemento

Primera etiqueta p

segunda etiqueta p

Quinto elemento

Sexto elemento



Podemos seguir haciendo más específicas las selecciones. Así no solo podemos indicar la posición que ocupa entre un conjunto de hermanos, sino que podemos indicar la selección en función del tipo de elemento de un conjunto de hermanos. Por ejemplo, tenemos:

- ✓ `:nth-of-type()`: similar a `nth-child()`, pero de un tipo de elemento determinado.
- ✓ `:nth-last-of-type()`: selecciona uno o más elementos partiendo desde el final de un tipo determinado de elemento.
- ✓ `:first-of-type`: primer elemento de un tipo determinado de un conjunto de hermanos.
- ✓ `:last-of-type`: selecciona el último elemento de un tipo determinado de un conjunto de hermanos.
- ✓ `:only-of-type`: selecciona aquel elemento que no tiene ningún hermano, teniendo en cuenta el tipo.

En el siguiente ejemplo se ponen en juego estas últimas pseudoclases.

## Código ejemplo 2.

```
<html>
  <head>
    <title>CSS Pseudoclasses</title>
  </head>
  <body>
    <div>
      <span>Elemento 1</span>
      <span>Elemento 2</span>
      <span>Elemento 3</span>
      <span>Elemento 4</span>
      <span>Elemento 5</span>
    </div>
    <div>
      <span>Elemento 6</span>
      <span>Elemento 7</span>
      <span>Elemento 8</span>
      <span>Elemento 9</span>
      <span>Elemento 10</span>
    </div>
  </body>
</html>
```

/ Selección de primer elemento de tipo span

```
div span:first-of-type {
  color: red;
}
```

/ Selección del elemento 4 de tipo span

```
div span:nth-of-type(4) {
  color: green;
}
```

/ Selección de los elementos span 1-3 o 5

```
div span:nth-of-type(1-3) {
  color: blue;
}
div span:nth-of-type(5) {
  color: blue;
}
```

/ Selección de los elementos span 1-3 o 5 empezando desde el final

```
div span:nth-last-of-type(1-3) {
  color: blue;
}
div span:nth-last-of-type(5) {
  color: blue;
}
```

## Resultado para el código de ejemplo 2.

## .Enero

• Febrero  
• Marzo

## .Abril

• Mayo  
• Junio

En la siguiente línea tenemos un conjunto de palabras entre etiquetas span, a las cuales se le aplica estilos el número de span lleva un número

Lorem 1 ipsum es 2 simplemente 3 el texto de 4 relleno de las 5 imprentas y archivOS de 7 texto.

Primer elemento

Segundo elemento

Tercer elemento

Cuarto elemento

Primera etiqueta p  
segunda etiqueta p

Quinto elemento

Sexto elemento



Elaboración propia. *Pseudoclases type-child* (CC BY-SA)

Los siguientes pseudoelementos pueden utilizarse con muchos elementos pero tienen una amplio abanico de posibilidades con los los campos de formularios:

- ✓ :valid: seleccionaremos aquellos campos cuyo valor es válido.
- ✓ :invalid: seleccionamos aquellos campos cuyo valor es inválido.
- ✓ :checked: seleccionamos aquellas opciones de campo tipo select que estén marcadas.
- ✓ :matches (lista de selectores sobre los que se va a aplicar un conjunto de reglas): actualmente no soportado por todos los navegadores.

Ejemplo:

Los siguientes pseudoelementos además de utilizarse en campos de formularios se pueden aplicar a un un conjunto más amplio de elementos:

- ✓ :focus hace referencia a los elementos que tienen el foco, como ocurre, por ejemplo, en los elementos de un formulario.
- ✓ :enabled: seleccionamos aquellos elementos que esten activados.
- ✓ :disabled: seleccionamos aquellos elementos que están desactivados.
- ✓ :lang (idioma en el que está definido un determinado elemento).

Ejemplos:

Se aplica fondo de color amarillo al campo que tenga activo el foco:

input:focus {background-color: yellow;}

Se aplica el estilo a cualquier párrafo que esté escrito en inglés:

p{lang-en~en}{color:red;}

A continuación se muestra un ejemplo más completo con algunas de las últimas pseudoclases estudiadas.

### Código ejemplo 3.

```
input {border: 1px solid black;
      border-radius: 10px;
      padding: 5px;
      width: 150px;
      height: 30px;
      font-size: 14px;
      color: #ccc;
      background-color: white;
      transition: width 0.5s, height 0.5s;
      outline: none;
      margin-bottom: 10px;}
```

/\* Cuando el campo el invita al borde del campo lo ponemos de color rojo \*/

```
input:focus {
    border-color: red;
    width: 200px;
    height: 40px;
}
```

/\* Aquando el campo es válido el borde lo ponemos de color verde \*/

```
input.valid {
    border-color: green;
    width: 200px;
    height: 40px;
}
```

/\* Aquellos campos que están en error se le pone en fondo de color azul \*/

```
input.error {
    background-color: #e0e0ff;
    width: 200px;
    height: 40px;
}
```

/\* Seleccionamos todos los campos desactivados y le aplicamos estilos \*/

```
input:disabled {
    opacity: 0.5;
    background-color: #f0f0f0;
    border-color: #ccc;
    width: 200px;
    height: 40px;
}
```









## Autoevaluación

Teniendo como base el código anterior (código básico con pseudoclase nth-child), ¿qué elementos p no contenidos en la etiqueta main tendrán color rojo?

- 3,7,10,13
- 4,7,10,13
- Todos
- 4,8,12,13

## 2.6.- Pseudoselectores: Pseudoelementos

Si queremos aplicar reglas de estilo a elementos especiales como: la primera línea de un párrafo o su primera letra, emplearemos los pseudoelementos.



Elaboración propia. *Pseudoelementos.*  
[CC BY-SA](#)

### Pseudoelementos.

Suelen ser partes de un elemento ya existente, como puede ser su primera línea (::first-line) o su primera letra (::first-letter), aunque también nos permite hacer referencia a elementos sin concretar en la estructura del documento porque dependen de la estructura del documento (::before y ::after).

Ejemplos:

A screenshot of a web browser displaying three paragraphs of text. The first paragraph has a red border and a white background. The second paragraph has a green border and a white background. The third paragraph has a blue border and a white background. Each paragraph contains a small black dot icon at the top left.

Explicaciones:

- ✓ En el primer ejemplo añadimos espacio extra a la primera linea del texto de cada párrafo,
- ✓ En el segundo modificamos el estilo (tamaño y color) de la primera letra de los párrafos pertenecientes a la clase "definicion".
- ✓ En el tercero, añadimos el texto " continuará ..." al final de cada párrafo perteneciente a la clase "incompleto".

Una diferencia de los **pseudoelementos** de las **pseudoclases** es que éstos van precedidos de dos "dos puntos dobles" (::).

A continuación repasamos algunos pseudoelementos:

- ✓ ::before: permite insertar texto antes de un elemento.
- ✓ ::after: permite insertar texto después de un elemento.
- ✓ ::placeholder: permite aplicar estilos a la sugerencia de texto indicada en la propiedad placeholder de un campo de tipo input.
- ✓ ::marker: permite aplicar estilo a los símbolos de una lista.
- ✓ ::selection: permite aplicar estilo a aquella parte de un documento que ha sido seleccionado o bien con el teclado o el ratón. Con este pseudoelemento solo pueden ser utilizadas las siguientes propiedades: color, background, background-color y text-shadow.

A continuación se deja un código ejemplo y su resultado donde se puede apreciar los estilos aplicados.

### Código ejemplo.

A screenshot of a web browser displaying three paragraphs of text. The first paragraph has a red border and a white background. The second paragraph has a green border and a white background. The third paragraph has a blue border and a white background. Each paragraph contains a small black dot icon at the top left.



Apellido:

Apellido:

Apellido:

Apellido:

Apellido:

Apellido:

Apellido:

Apellido:

## Resultado.

- Mes Enero:
- Mes Febrero:
- Mes Marzo:
- Mes Abril:
- Mes Mayo:
- Mes Junio:

**L**orem Ipsum es simplemente el texto de relleno de las imprentas y archivos de texto. Lorem Ipsum ha sido el texto de relleno estándar de las industrias desde el año 1500, cuando un impresor (N. del T. persona que se dedica a la imprenta) desconocido usó una galería de textos y los mezcló de tal manera que logró hacer un libro de textos especimen. No sólo sobrevivió 500 años, sino que también ingresó como texto de relleno en documentos electrónicos, quedando esencialmente igual al original. Fue popularizado en los 60s con la creación de las hojas "Letraset", las cuales contenían pasajes de Lorem Ipsum, y más recientemente con software de autoedición, como por ejemplo Aldus PageMaker, el cual incluye versiones de Lorem Ipsum.

Apellidos: (Obligatorio)





## Recomendación

En el siguiente enlace podrás hacer pruebas con los distintos tipos de selectores y familiarizarte con ellos. Podrás seleccionar selectores descendentes, hijos y adyacentes, utilizar selectores de clase e ID, los pseudoelementos y las pseudoclases y comprobar en el lado derecho de la página cuáles son los elementos que se verían afectados por una reglas de estilo aplicada a esos selectores.

 [Simulador de selectores](#)

Si aún sigues teniendo dudas sobre los selectores, echale un vistazo al siguiente enlace, donde se explica de forma más detallada los selectores estudiados y algunos más, podrás encontrar ejemplos variados:

 [30 selectores que debes conocer](#)

## 2.7. La palabra clave !important en CSS

!important se utiliza para dar prioridad a ciertas reglas. Cualquier definición de estilo que vaya acompañada de un !important tendrá prioridad sobre cualquier otra.

Como ya sabes, cuando tenemos una propiedad aplicada dos veces, el navegador hará caso a la última. En el ejemplo que se muestra a continuación está claro que se le asignarán 1200 píxeles al ancho del elemento identificado con el id="principal".



Si embargo, en el siguiente ejemplo se muestra como dando prioridad a la primera declaración con la palabra !important, podemos tener ese elemento con un ancho de 800 píxeles.



Recuperamos nuevamente el ejemplo donde poníamos en juego esta palabra clave. Seguramente ahora entenderás mejor dicho ejemplo.

 [Ejemplo de código aplicando estilos de formas diferentes.](#) (zip - 29.26 KB)

**Las declaraciones acompañadas de la palabra ! important tienen prioridad sobre otras declaraciones que afecten al mismo elemento.**

Con los selectores descendentes, hijos y adyacentes, se puede dar el caso de que sin querer, estemos aplicando un estilo que no queremos a un elemento, porque se ve indirectamente afectado por alguna regla de estilo y no sabemos cuál.

Por ejemplo: tenemos un párrafo en color rojo y no nos acordamos dónde lo configuramos porque realmente no tenemos ninguna regla que diga `p {color:red}`. En ese caso, y para no descerebrarnos buscando dónde está el fallo, si sabemos que los párrafos de texto emplean la letra de color verde, creariamos la regla `p { color: green !important; }` y solucionado!

**En la medida que podamos no debemos usar esta palabra clave**, ya que al aplicarla creamos excepciones a la lógica de nuestro código y detectar éstas posteriormente puede ser complicado, haciéndonos perder tiempo o incluso cometer errores en el diseño de nuestra página web.

**Recuerda que el uso de !important debe ser algo excepcional y no lo habitual.**

## Autoevaluación

**¿Cuál de los siguientes es el selector CSS que selecciona a todos los objetos con class="nuevo" que están contenidos en un tabla?**

- `table .nuevo`
- `table.nuevo`
- `table#nuevo`

### 3.- Propiedades de fuente y texto



#### Caso práctico



Elaboración propia (Uso Educativo no comercial).

La web de “**Migas Amigas**” ofrecerá a sus visitantes varias páginas donde se relatará su historia, la descripción del entorno donde se encuentra ubicada y, además, algunas recetas de los productos que en ella se venden.

Todo esto hace necesario que **Carlos** tenga los conocimientos necesarios para configurar la apariencia de todos estos textos.

**Carlos** se pone de nuevo en contacto con **Juan** para que éste le guíe en el aprendizaje de las propiedades y técnicas que necesita.

Las hojas de estilo nos van a permitir un control total sobre el formato del texto en las páginas web. Veamos las propiedades de CSS más utilizadas para formatear el texto.



#### Para saber más

A continuación se te ofrece un enlace a las especificaciones dadas por la **W3C** referente al módulo de fuentes nivel 3.

 [Módulo CSS de fuentes \(nivel 3\).](#)

## 3.1.- Propiedades de fuente

Las propiedades de las fuentes en CSS son usadas para configurar la apariencia deseada para el texto de un documento. Veamos las más empleadas:

### ✓ font-family<br />

Nos permite especificar un nombre de fuente en concreto o bien una familia genérica de fuentes. Se puede especificar una lista de fuentes separadas por comas teniendo en cuenta que si el nombre de la fuente o familia tiene algún espacio en blanco intercalado habrá que encerrarlo entre comillas. Se suelen poner varios tipos de fuentes, para que si falla uno se utilice el siguiente y así hasta llegar a la última.

Tradicionalmente para utilizar una fuente en un equipo cliente, ésta debería estar instalada en él, pero con la llegada de CSS3 esto no es del todo cierto y veremos posteriormente cómo podemos utilizar fuentes sin que estén instaladas.

Continuando con nuestra propiedad, podemos indicar nombres de fuentes concretas o nombres de familias de fuentes, en caso de que pongamos una fuente y no esté disponible el navegador cargará otra fuente de la misma familia. Las familias tradicionales de fuentes suelen ser: serif, sans-serif, monospace, cursive y fantasy.

Para tener control sobre nuestro desarrollo, es conveniente poner al final de las diferentes fuentes, una fuente genérica, de tal forma que si no se carga la fuente especificada, se cargue una fuente de dicha familia.

✓	Se indican diferentes familias de fuentes.
✓	font-family: monospace, serif, sans-serif;
✓	font-family: serif, sans-serif, monospace;
✓	Se indican varios tipos de fuentes y al final una familia de fuentes genérica.
✓	font-family: cursive, sans-serif, serif, monospace;
✓	font-family: cursive, sans-serif, serif, monospace, monospace;





## Recomendación

Accedemos a un recurso en línea donde se dan los pasos a seguir para instalar y desinstalar fuentes en el sistema operativo Windows 10.

[Instalar fuentes en Windows 10.](#)

A continuación se muestran un conjunto de fuentes relacionadas con el fabricante de software **Microsoft**.

[Referencia de fuentes y tipografía de Microsoft.](#)

Y pulsa el enlace: "**Font library**"

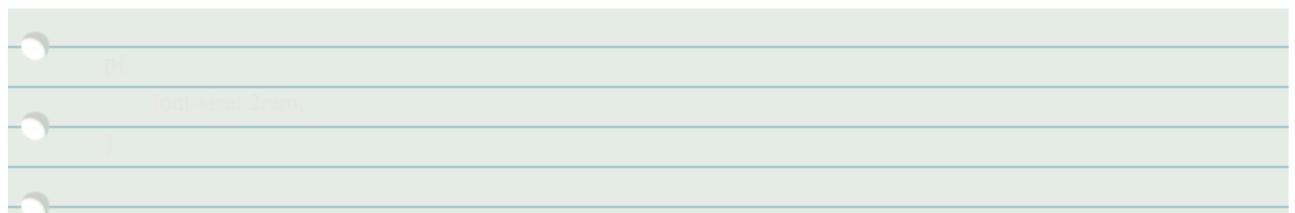
Terminamos con un nuevo enlace que nos permitirá obtener más fuentes para incorporar a nuestro sistema operativo:

[Free Fonts.](#)

### ✓ font-size

Nos permite configurar el tamaño del texto. Mientras que el HTML estándar prevé sólo 7 niveles predefinidos para el tamaño del texto, las hojas de estilo CSS permiten un control mucho más preciso y elástico, sin limitaciones prácticamente.

Podemos establecer tamaños de forma absoluta, de forma relativa, con un valor numérico o en forma de porcentaje.



### ✓ font-weight

Nos permite establecer el espesor o intensidad de las fuentes, como la etiqueta **<b>** del HTML clásico. Es posible asignar hasta 7 valores diferentes: normal, bold, bolder, lighter, 100, 200, 300, 400, 500, 600, 700, 800 ó 900.

### ✓ font-style

Nos permite configurar el "estilo" de la fuente. Hay tres valores posibles:

- ◆ normal: no configura ningún estilo en particular, sino que toma el definido por defecto en el navegador.
- ◆ italic: equivale a la etiqueta del HTML clásico **<i>**, que coloca el texto en cursiva.
- ◆ oblique: funciona aparentemente, como "italic".

#### ✓ font-stretch

Esta propiedad permitirá hacer el texto más ancho o estrecho: ultra-condensed, extra-condensed, condensed, semi-condensed, normal, semi-expanded, expanded, extra-expanded, ultra-expanded. Hay que tener en cuenta que la fuente que estamos utilizando ofrezca estas posibilidad (que no siempre es así).

#### ✓ font-variant

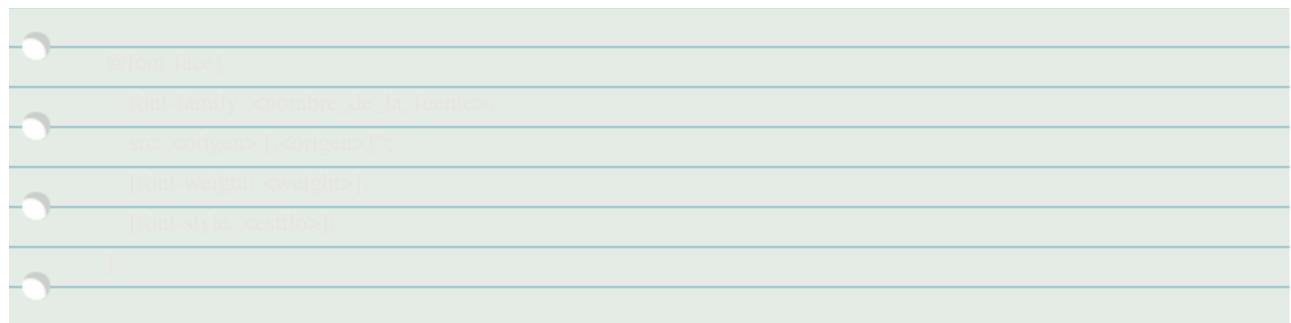
Permite dos posibilidades:

- ◆ normal: el texto no cambia de apariencia.
- ◆ small-caps: el texto pasa a mostrarse en mayúsculas de un tamaño inferior.

#### ✓ @font-face

Con la llegada de CSS3 se incluye esta regla que nos permitirá incorporar a nuestra página cualquier tipo de fuente, independientemente de que la tengamos instalada en nuestro sistema. La fuente podemos incorporarla bien descargándola a nuestro sitio (y haciendo referencia a ella de forma local) o indicando alguna dirección de internet donde esté disponible dicha fuente, como puede ser  [Google Fonts](#). En este sitio disponemos de multitud de fuentes que podemos utilizar, solo habrá que tener en cuenta el tipo de licencia que tiene el recurso que deseemos utilizar.

A continuación se muestra la sintaxis.



En la propiedad font-family indicaremos el nombre con el que vamos a denominar a la fuente que vamos a utilizar en nuestro desarrollo.

En la propiedad src indicaremos el origen de la fuente, puede ser como se ha comentado de forma local (descargada en nuestro sitio) o accediendo a un lugar en Internet donde se encuentre la misma.

Una cuestión a tener en cuenta es que cada navegador soportará un formato de fuente, por lo que si queremos que dicha fuente se visualice en todos los navegadores, debemos asegurarnos de qué formatos soporta cada navegador e incorporarlos. Es recomendable incorporar los formatos [.woff2](#) admitido por Chrome, Firefox y Edge y [.woff](#). Otros posibles formatos pueden ser [.svg](#), [.ttf](#), etcétera. Como se comentó al inicio de nuestra unidad desde [Can I Use](#) podemos ver el  [soporte de algunos de estos formatos](#).

A continuación mostramos cómo utilizar una fuente que hemos descargado de forma local a nuestro sitio web:

• Estilo:

“Nuestro diseño se basa en la tipografía Roboto, donde la fuente estará en el archivo Roboto-Light.”

• Fuente:

“Tipografía: Roboto”

“Fuente: Roboto Light (400).”

• Tamaño:

“Tamaño:

• Tipografía:

Para aplicar nuestra fuente a cualquier elemento, tendremos que hacer lo siguiente:

• CSS:

“Tipografía: Roboto.”

• CSS:

Relacionado con la incorporación de nuevas fuentes a nuestro desarrollo, podemos incorporar estas también mediante la regla @import o añadiéndola mediante el uso de hojas de estilo externas. Concretamente desde **Google Fonts**, se nos proporcionan estas dos posibilidades. Mostramos un ejemplo donde vemos cómo incorporar una fuente con la regla @import o haciendo un enlace a una hoja de estilo externa con link. La forma de aplicar la fuente a nuestro desarrollo, viene normalmente indicado en el lugar donde nos descargamos ésta. Ejemplo:

• Ejemplo para incorporar a un desarrollo la fuente Open Sans diseñada por Steve Matteson, desde Google Fonts. “

• Mediante el enlace de una página web:

“Código:

```
<link href="https://fonts.googleapis.com/css?family=Open+Sans+Light&subset=latin&display=swap" rel="stylesheet">
```

“Código:

• Mediante el uso de la regla @import:

“Código:

```
@import url("https://fonts.googleapis.com/css?family=Open+Sans+Light&subset=latin&display=swap");
```

“Código:

• Para aplicar este tipo de fuente a un elemento tendremos que utilizar la siguiente propiedad:

“font-family: 'Open Sans', sans-serif;”

Para finalizar este apartado indicar que existen otras propiedades menos comunes relacionadas con las fuentes que quedan fuera de nuestra unidad, como pueden ser: font-variant-caps.



## Debes conocer

En el siguiente enlace puedes acceder a una presentación donde se muestran algunos ejemplos de las propiedades anteriores, se habla de las unidades de medida, tanto absolutas como relativas, y de la propiedad font, que permite establecer todos los atributos vistos hasta ahora (y alguno más) en una única declaración.

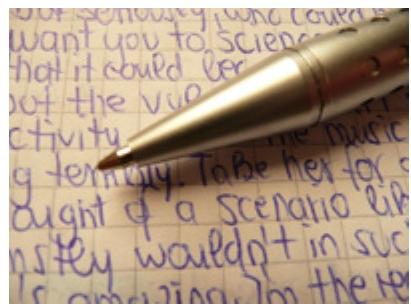
[Resumen textual alternativo](#)

## 3.2.- Propiedades de texto

Las propiedades de texto permiten aplicar estilos a los textos espaciando sus palabras o sus letras, decorándolo, alineándolo, transformándolo, etcétera. Algunas de estas propiedades son:

### ✓ color

Permite definir el color de nuestro texto. En el siguiente apartado veremos las diferentes alternativas que tenemos para asignar un color a una propiedad.



[photosteve101. CC BY](#)

### ✓ text-align

Establecer la alineación de un elemento dentro de un elemento de bloque, los posibles valores que puede tomar son `right`, `left`, `center` y `justify`.

### ✓ vertical-align

Nos permite establecer el alineado vertical de una línea dentro de un bloque o una celda de una tabla. Los posibles valores son `baseline`, `top`, `bottom`, `text-top`, `middle`, `sub`, `super`, `left`, `center` y `justify`, valores porcentuales y absolutos. En el siguiente enlace tenemos un pequeño [generador](#) donde se puede ver cómo afecta cada valor de la propiedad a un elemento.

### ✓ text-decoration

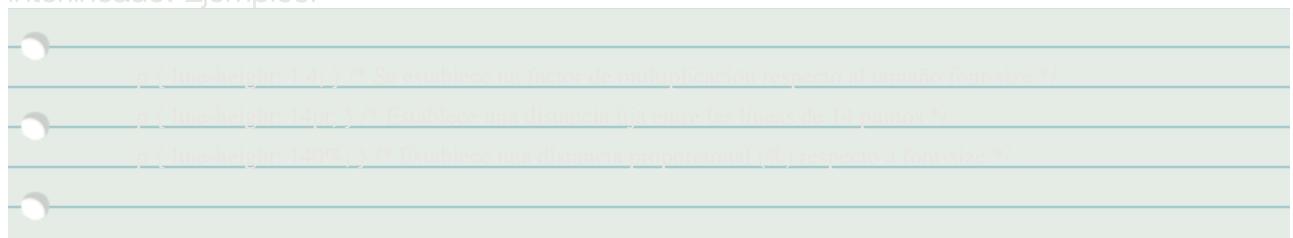
Permite decorar el texto con subrayados y otros efectos. Los valores que puede tener son: `none` (ninguno), `underline` (subrayado), `overline` (con una línea por encima), `line-through` (tachado), `blink` (parpadeante) e `inherit` (heredado).

### ✓ text-transform

Controla la apariencia de las letras en un elemento. Los valores que puede tener son: `none` (texto normal, con mayúsculas y minúsculas), `capitalize` (cada palabra comienza con mayúsculas), `uppercase` (todo el texto aparece en mayúsculas) y `lowercase` (todo el texto aparece en minúsculas).

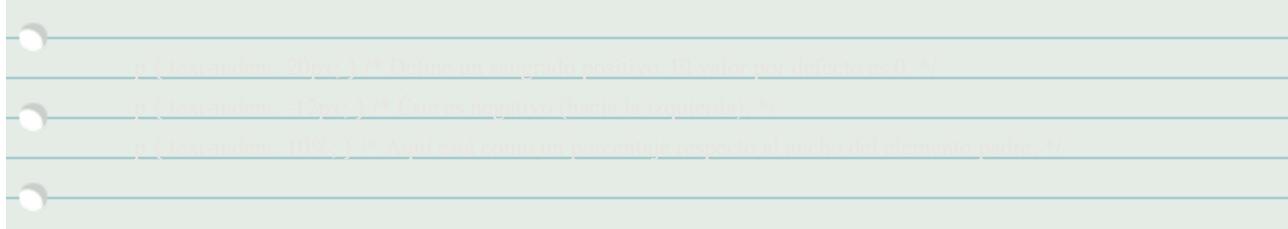
### ✓ line-height

Establece el espacio que hay entre dos líneas consecutivas, lo que se denomina como interlineado. Ejemplos:



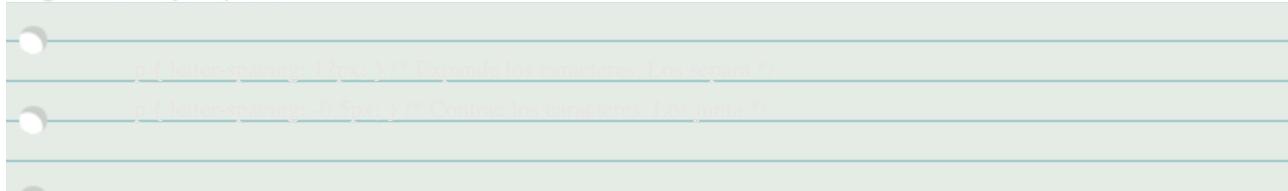
## ✓ text-indent

Sangra la primera línea de texto de un párrafo. Ejemplos:



## ✓ letter-spacing

Configura el espacio que hay entre los caracteres. Este valor puede aumentar o disminuir, ya que al igual que text-indent y otras propiedades, admite valores positivos y negativos. Ejemplos:



## ✓ white-space

Permite establecer cómo se gestionan los espacios en blanco en un elemento. Los valores que puede tener son: normal (los espacios en blanco adicionales son ignorados por el navegador), pre (los espacios en blanco adicionales son utilizados como cuando se emplea la etiqueta pre en [HTML](#)), nowrap (no se produce el ajuste de línea automático por lo que el texto permanecerá en la misma línea hasta que encuentre una etiqueta).

## ✓ word-spacing

Con esta propiedad podemos definir el espacio de separación entre cada palabra de un elemento. Podemos especificar una medida (absoluta o relativa). El valor por defecto es normal y equivale a 0,25em<br /><br />

## ✓ direction

Con esta propiedad podemos definir la orientación de nuestra escritura, pudiendo ser de izquierda a derecha o viceversa, con los valores `ltr` y `rtl`.

## ✓ overflow

Con esta propiedad podemos controlar cómo se comportará un texto cuando no cabe en su contenedor, así podemos indicar que la parte que sobresale se oculte con el valor `hidden`, que se siga viendo aunque sobresalga del contenedor con el valor `visible` (valor por defecto) o que aparezca un barra de scroll (horizontal o verticalmente dependiendo del desbordamiento) con el valor `scroll`.

## ✓ overflow-x y overflow-y

Su funcionamiento es igual que la propiedad anterior, pero especificando valores en el eje x (horizontalmente) o en el eje y (verticalmente).

#### ✓ text-overflow

En el caso de que un texto no se muestre, ya que no cabe en su contenedor (desbordamiento), con esta propiedad podemos indicar cómo queremos que se indique esta situación. Los valores que puede tomar son: clip (valor por defecto) no realiza nada y ellipsis que nos mostrará tres puntos (...). En algunos navegadores podemos incluir el o los caracteres que queremos que aparezcan. Para ello tendremos que incluirlos entre comillas, aunque esta funcionalidad no está aún estandarizada. Destacar que esta propiedad no produce el desbordamiento, solo indica qué señal mostrar en caso de que lo haya.

#### ✓ overflow-wrap (word-wrap)

Con esta propiedad podemos indicar cómo queremos que se corte una palabra para que no sobresalga de un objeto contenedor; los valores que puede tener son normal (valor por defecto) no hará nada, break-word la palabra se cortará en cualquier carácter.

#### ✓ hyphens

Nos permite indicar cómo se divide una palabra que va a producir un desbordamiento por su longitud. Esta propiedad tiene en cuenta el idioma del texto, es decir el valor del atributo lang. Los valores que puede tomar esta propiedad son none no realizará nada, auto será el navegador el que decida como realizar la separación (en función del lenguaje) o manual indicaremos de forma manual cuándo realizar dicha separación en caso de que fuera necesario. Actualmente esta propiedad no está estandarizada.

#### ✓ text-shadow

Esta propiedad permite crear texto con sombras. En un apartado posterior se verá esta propiedad de una forma más detallada.

## Autoevaluación

**Queremos realizar un documento web donde la fuente base de toda la página sea: de color negro, de tipo Arial, con un tamaño 0.9 veces la letra por defecto del navegador y un espaciado entre líneas de 1.4. ¿Cuál sería la regla CSS que consigue esto?**

- body { font: 0.9em/1.4 Arial, Helvetica, sans-serif; color: #000; }
- body { font: 0.9em/1.4 Arial, sans-serif; color: #000; }
- body { color: #000; font: .9em/1.4 Arial; }
- body { font: .9em/1.4 Arial; color: #000; }

[Mostrar retroalimentación](#)



## Para saber más

En el siguiente enlace podemos conocer un poco más sobre algunas propiedades CSS3 relacionadas con las fuentes y el texto:

 [Propiedades sobre texto.](#)

## 3.3.- Columnas

Una de las novedades que nos presenta [CSS3](#) es la posibilidad de insertar texto en múltiples columnas al igual que puede verse un texto en un periódico o revista.

Para mostrar un texto con estas características tendremos que utilizar la propiedad `column-count`, con ella indicaremos el número de columnas que queremos crear. Junto a esta propiedad existen otras tantas que nos permitirán configurar nuestro texto, indicando aspectos como espacio de separación entre columnas, ancho de las columnas, etcétera. Pasamos a describir estas propiedades:

En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgο de los de lanza en astillero, adarga antigua, rocin flaco y galgo corredor.

Una olla de algodón más hacia que camero, salió de la noche en las noches, duelos y quebrantos los sábados, lantatas los viernes, algún palomino de añadura los domingos, consumían las tres partes de su hacienda.

El resto della concluían sayo de velarte, calzas de velludo para las fiestas, con sus pantuflos de lo [mismo](#), y los días de entremesana se honraba con su [vellori](#) de lo más fino.

Tenía en su casa una ama que pasaba de los cuarenta, y una sobrina que no llegaba a los veinte, y un mozo de campo y plaza, que así ensillaba el rocin como tomaba la podadera.

Frisaba la edad de nuestro hidalgο con los cincuenta años; era de complejión recta, seco de carnes, enjuto de rostro, gran madrugador y amigo de la caza,

Quieren decir que tenía el sobrenombre de Quijano o Quesada, pero en esto hay alguna diferencia en los autores que deste caso escriben; aunque por conjetas verosímiles se dejó entender que se llamaba Quijano.

Pero esto importa poco a nuestro cuento: basta que en la narración déjese que salga un punto de la verdad.

Es, pues, de saber que este sobredicho hidalgο, los ratos que estaba ocioso -que eran los más del año-, se daba a leer libros de caballerías, con tanta afición y gusto que olvidó casi de todo punto el ejercicio de la caza, y aun la administración de su hacienda; y llegó a tanto su

curiosidad y deseo en esto, que vendió muchas [haciendas](#), de lejos de sembraduras para comprar libros de caballerías en que leer, y así llevó a su casa todos cuantos pudo haber [de ellos](#).

Y de todos estos libros le parecieron bien como los que compuso el famoso Feliciano de Silva, porque la claridad de su prosa y aquellas [entrecidas](#) razones suyas le parecían de perlas; y más cuando llegaba a leer aquellos requiebros y cartas de desafíos, donde en muchas partes salía escrito el nombre de la sirnión que a mi razón se hace, de tal manera mi razón enfaquece, que con razón me quejo de la vuestra [fermosura](#). Y también cuando leía: Cíos altos cielos que de vuestra divinidad dependiente con las estrelas os fortifican, y que habéis de ser la del mercimiento que merece la vuestra grandeza.

Elaboración propia. [Texto en columnas \(CC BY-SA\)](#)

#### ✓ column-count

Permite definir el número de columnas que tendrá nuestro elemento.

#### ✓ column-fill

Nos permite indicar cómo se rellena el espacio destinado a las columnas, puede tomar los valores auto o balance.

- ◆ auto: ocuparía todo el espacio de una columna y cuando llegara al final de ésta, comenzaría con la siguiente columna. Depende de la longitud del contenedor donde hayas definido este tipo de escritura.
- ◆ balance: rellena con el texto todas las columnas que hayamos definido en column-count.<br /><br />

#### ✓ column-gap<br />

Determina el espacio que habrá de separación entre cada una de las columnas.

#### ✓ column-rule<br />

Configuramos el ancho, color y tipo del separador (regla) que habrá entre las columnas, el único valor que es obligatorio es el estilo del separador. Posteriormente veremos los valores que le podemos asignar.

Esta propiedad al igual que ocurre con <code>border se puede configurar de forma individual con otras tres propiedades que enunciamos a continuación:

- ◆ column-rule-color: especifica el color de la regla o separador.
- ◆ column-rule-style: indica el tipo de separador, puede tomar los valores: dotted, dashed, solid, double, groove, ridge, inset y outset.
- ◆ column-rule-width: especifica el ancho de la regla o separador.

#### ✓ column-width<br />

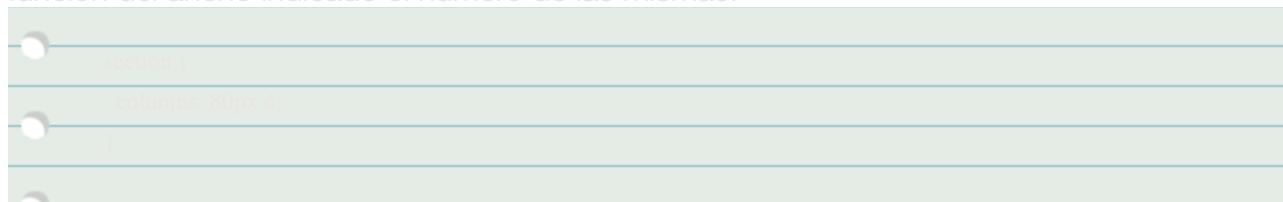
Indicamos el ancho que van a tener nuestras columnas, así se generarán tantas columnas como sean necesarias para que nuestro texto quepa, creando columnas de este ancho. Esto es conveniente para que nuestro diseños se adapten a diferentes dimensiones. Si nosotros definimos un número de columnas con column-count es posible que el ancho de la columna sea mayor que el indicado en esta propiedad, siendo ignorada. También se puede dar el caso de que pongamos un número de columnas superior al necesario para almacenar nuestro texto y en este caso sí se tendrá en cuenta la propiedad <code>column-width creándose menos columnas de las indicadas. Prueba con el código de ejemplo que se deja a continuación. Podemos resumir que se requiere un número mínimo de columnas para que dicha propiedad sea tenida en cuenta.

#### ✓ column-span<br />

Nos permitirá que un elemento ocupe todas las columnas o que se limite solo a una de ellas.

#### ✓ column<br />

Nos permitirá definir el número de columnas y anchura de las mismas. Hay que tener en cuenta que cuando especificamos el número de columnas y el ancho, el número de columnas representa el número máximo que existirán y el ancho, el mínimo ancho que tendrán éstas. Por el contrario, si establecemos solo el ancho, el navegador adaptará en función del ancho indicado el número de las mismas.



A continuación dejamos un ejemplo y cómo se visualiza el mismo.

## Código ejemplo.

```
<div>
    <div>Este es el primer</div>
    <div>segundo</div>
    <div>tercero</div>
    <div>cuarto</div>
    <div>quinto</div>
    <div>sexta</div>
</div>

<div>
    <div>Este es el primer</div>
    <div>segundo</div>
    <div>tercero</div>
    <div>cuarto</div>
    <div>quinto</div>
    <div>sexta</div>
</div>

// Definimos el ancho de nuestras columnas
column-width: 100px;

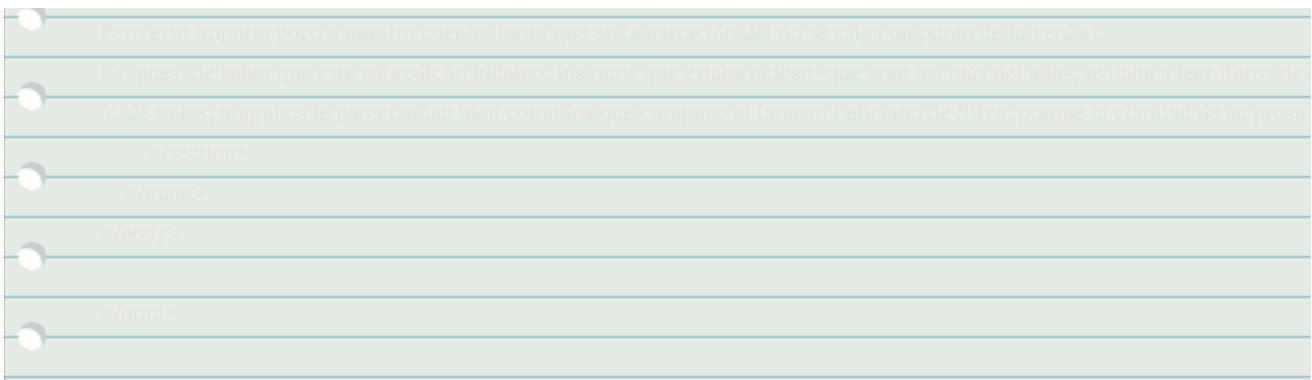
// Forma de colinear el texto de las columnas
column-align: left;
column-fill: balanced;

// Definir el espacio entre las columnas
column-gap: 10px;

// Columna que ocupa la recta horizontal anteriormente definida
// La recta de la que ocupa es: recto, cuadros dobles, gruesos, rígidos, rectos, rectos
// Colores de los cuadros
column-rule: 1px solid black;

// Indicamos si queremos que el elemento ocupe todo el espacio desumido a las columnas (se expande sobre todos) o sea
// no column-span: none; o
// no column-span: all;
column-span: all;

// Cambiar el color de la Margen de los cuadros
// En lugar de la Margen, de cada cuadro en suero acordemos, no le mucha tiempo que dura en la vida de los
// Una pila de algo más vale que cierre, estropea las más rojas, arruina y quebrantes los demás. Siempre los vienes, siempre
// El resto de la conchilla seva de vestir, calza de vestido para los hermos, con sus pañuelos de lo mismo, y los días de verano
// Toma en su casa una cosa que pasión de los cuadros, y una solana que no negaría a los venidos, y un poco de campo y plaza
// Una sombra que no se pierde de la noche ni de la noche
// Que nadie se pierda de la noche con los encantamientos de complejidad social, sea de carnes, enemigo de carnes, grande
// Jardines que no se pierden de la noche, y jardines, y jardines
```



## Resultado.

### Don quijote de la mancha

En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo

que vivía un hidalgo de los de lanza en astillero, adarga antigua, rocin flaco y galgo corredor. Una olla de algo más vaca que carnero, salpicón las más noches, duelos y quebrantos los sábados, lantejas los viernes,

algún palomino de añadidura los domingos, consumían las tres partes de su hacienda. El resto della concluían sayo de velarte, calzas de velludo para las fiestas, con sus pantuflas de lo mismo, y los días de entresemana se

honraba con su vellorí de lo más fino. Tenía en su casa una ama que pasaba de los cuarenta, y una sobrina que no llegaba a los veinte, y un mozo de campo y plaza, que así ensillaba el rocin como tomaba la podadera.

### Continuamos con la obra

Frisaba la edad de nuestro hidalgo con los cincuenta años; era de complexión recia, seco de carnes, enjuto de rostro, gran madrugador y amigo de la caza. Quieren decir que tenía el sobrenombre de Quijada, o Quesada, que en esto hay alguna diferencia en los autores que dese caso escriben; aunque por conjecturas verosímiles se deja entender que se llamaba Quijana. Pero esto importa poco a nuestro

cuento: basta que en la narración dél no se salga un punto de la verdad. Es, pues, de saber que este sobredicho hidalgo, los ratos que estaba ocioso -que eran los más del año-, se daba a leer libros de caballerías, con tanta afición y gusto que olvidó casi de todo punto el ejercicio de la caza, y aun la administración de su hacienda; y llegó a tanto su curiosidad y desatino en esto, que vendió

muchas hanegas de tierra de sembradura para comprar libros de caballerías en que leer, y así, llevó a su casa todos cuantos pudo haber dellos. Y, de todos, ningunos le parecían tan bien como los que compuso el famoso Feliciano de Silva, porque la claridad de su prosa y aquellas entricadas razones suyas le parecían de perlas; y más cuando llegaba a leer aquellos requiebros y cartas de desafíos,

donde en muchas partes hallaba escrito: La razón de la surazón que a mi razon se hace, de tal manera mi razon enflaquece, que con razón me quejo de la vuestra fermosura. Y también cuando leía: los altos cielos que de vuestra divinidad divinamente con las estrellas os fortifican, y os hacen merecedora del merecimiento que merece la vuestra grandeza.

Elaboración propia. *Ejemplo de columnas.* ([CC BY-SA](#))



## 4.- Los colores, fondos y sombras



### Caso práctico



cliff1066™ (CC BY)

Ni que decir tiene que el equipo de **BK** intentará que la web de “**Migas Amigas**” adquiera una apariencia lo más elegante y vistosa posible. Para que esto sea así, un factor muy importante será configurar los colores y los fondos adecuados.

Los colores y las imágenes de fondo ya fueron seleccionados por todo el equipo de diseño. Ahora, **Carlos** debe descubrir cómo aplicar éstos a la web sin que la programación de CSS sea un obstáculo para lograr justo el diseño deseado.

Esta vez será **Ada** la que se ponga en contacto con **Juan** para guiarlo en el aprendizaje de las propiedades y técnicas necesarias.

CSS permite controlar el color y los fondos con unas posibilidades que están a años luz de los efectos que podemos alcanzar empleando sólo HTML.

A lo largo de la unidad hemos visto cómo son muchos los elementos a los cuales podemos configurar sus atributos y entre éstos uno de los más llamativos o vistoso es el color. Así podemos indicar el color de un tipo de letra, el color de un borde, el color de fondo, etcétera. Lo primero que tendremos que conocer es la forma en que podemos definir o indicar el color a los diferentes atributos de un elemento. Las posibilidades que tenemos son varias, pasamos a describirlas:

- ✓ **Nombre de color:** simplemente basta con indicar el nombre del color, algunos de estos posibles valores pueden ser black, green, red, blue. En el siguiente enlace mostramos una  [lista de colores](#) reconocidos.
- ✓ **RGB:** mediante este modelo indicaremos un valor entre 0 y 255 para los colores Rojo (R), Verde (G) y Azul (B). Así mediante la función `rgb()` y los valores de los tres colores anteriores definimos nuestro color. Ejemplo: `rgb(0,0,255)` (correspondería al color azul).
- ✓ **Hexadecimal:** en este caso indicaremos los valores de los colores [RGB](#) mediante notación hexadecimal precedido del carácter #, los valores hexadecimales pueden ir desde 00 a FF, indicando un valor para cada uno de los colores `#RRGGBB`, por ejemplo: `#0000FF` (correspondería al color azul).
- ✓ **RGBA:** partiendo de lo indicado en el apartado anterior [RGB](#), además podremos definir el canal alfa u opacidad de nuestro color. Esto lo indicaremos con el último parámetro que puede tomar los valores comprendidos entre 0 y 1. Para asignar un color con esta nomenclatura utilizaremos la función `rgba()`. Ejemplo: `rgba(0,0,255,0.6)`.
- ✓ **HSL:** es otra forma de indicar un color, los parámetros que ahora tendremos que configurar serán, matiz o color (hue), saturación (saturation) y brillo (lightness). El color puede tomar valores entre 0 y 360 (grados) se corresponde con una rueda cromática donde el valor 0 corresponde al rojo, 120 a verde y el azul al valor 240. La saturación se indicará mediante un valor porcentual entre 0 y 100 (grises) y la luminosidad que se expresará también de forma porcentual y sus valores irán entre 0 (negro) a 100 (blanco). La función que utilizaremos será `hsl()`. Ejemplo: `hsl(240,100%,50%)`
- ✓ **HSLA:** a lo anteriormente indicado añadiremos un nuevo canal alfa, relacionado con la opacidad, al igual que ocurría con la propiedad `rgba()`. El valor de este nuevo parámetro irá entre 0 y 1. La función que utilizaremos será `hsla()`. Ejemplo `hsl(240,100%,50%,0.5)`.

Dejamos en enlace a un generador de colores, donde podremos seleccionar de forma visual un color y nos dará su correspondiente codificación en cada una de los diferentes métodos de explicados.  [Herramienta de selección de color.](#)

Para terminar indicar que hay elementos que pueden heredar el color de un elemento padre con la palabra `inherit`, en algunos casos este valor lo tienen algunos atributos por defecto. En los casos en que no se cumpla esto, podemos definir el color de un elemento hijo igual que el del elemento padre con la palabra clave `currentColor`.



## Para saber más

Se deja enlace con los últimos trabajos realizados por la **W3C** relacionados con el módulo del color:

 [CSS Color Nivel 4 \(En trabajo\)](#)

Además de los modelos de color vistos en nuestro apartado existen otros como por ejemplo el utilizado en el mundo de la impresión.

 [Modelo de color: CMYK](#)



## Recomendación

A medida que vayas estudiando estos puntos también puedes visitar el siguiente enlace de la web de la **W3C** donde encontrarás un resumen de las propiedades CSS más utilizadas en el modelo de cajas.

 [Propiedades CSS para colores y fondo.](#)

## **4.1.- Color del primer plano, color de fondo y degradados.**

---

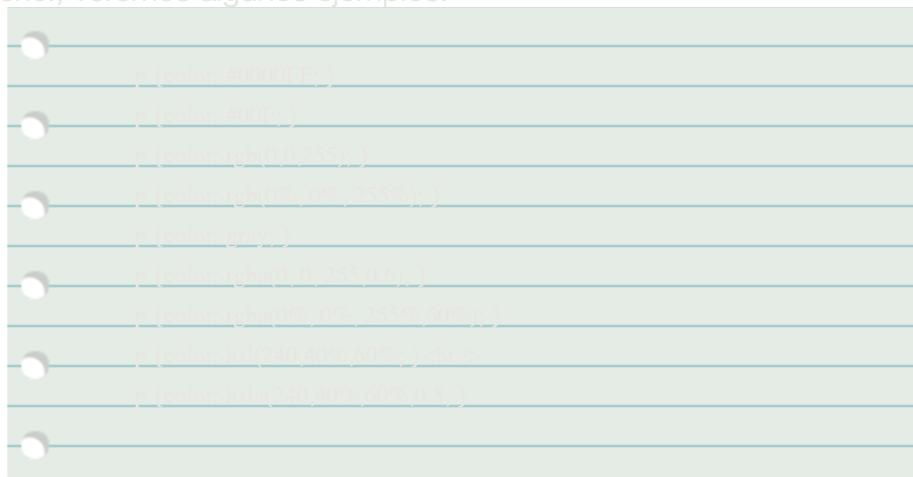
Para establecer los colores de primer plano y de fondo existen dos propiedades distintas. La propiedad **color** es la que debes utilizar para configurar el color del primer plano, es decir, el color del texto y el color por defecto del borde de un elemento. Mientras que para configurar el color de fondo deberás emplear la propiedad **background-color**.

- ✓ **color**  
Esta propiedad ya ha sido visto en el apartado fuentes, pero puede ser utilizada por otros muchos elementos. Con respecto al color del primer plano de un elemento deberás tener en cuenta algunas cuestiones como:
  - ◆ Si añades color al primer plano de una imagen, ésta seguirá viéndose pero el color se aplicará al borde de la imagen.
  - ◆ La propiedad border-color ignora la propiedad color.
  - ◆ Para configurar el color de todo un documento debemos escribir una regla con esta propiedad color para el selector body.
  - ◆ La aplicación de color a los elementos de los formularios no funciona bien en todos los navegadores.



Brad K. CC BY

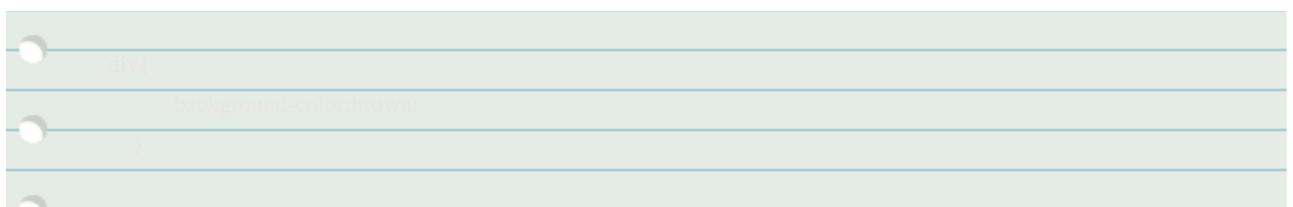
Teniendo en cuenta las posibilidades que tenemos para definir el color de un elemento en el apartado anterior, veremos algunos ejemplos:



- ✓ **background-color**  
Con CSS no sólo se puede proporcionar un color de fondo a toda la página, también se puede configurar el color de fondo de cualquier elemento del documento, tanto si son elementos de bloque como de línea.

Con la aparición de CSS, se recomienda emplear "cajas de color" en sustitución de las tablas. Recuerda que las tablas no deben utilizarse como herramienta de posicionamiento, solo deben utilizarse como elemento de agrupación y muestra de datos.

Ejemplo de como aplicar un color de fondo.



**Las propiedades relativas al fondo no se heredan, pero como el valor predeterminado de esta propiedad es transparent, salvo que se especifique un color concreto, el color de fondo del elemento padre aparecerá a través de sus elementos hijos.**

Ejemplo: p {padding: 5px; background-color: #ccc; }

Hasta ahora hemos visto cómo asignar un color de fondo, pero con la llegada de CSS3, las posibilidades que se nos ofrecen para asignar un fondo son mucho mayores, así podemos realizar gradientes y podemos asignar una o más imágenes de fondo.

**En caso de existir un color de fondo y una imagen, está última se superpondrá al color.**

### Fondos con degradado.

Se contemplan dos tipos de degradados:

- ✓ Lineal.
- ✓ Circular.

Para crear un gradiente tendremos que tener en cuenta:

- ✓ Posición donde queremos que comience el degradado y dirección del mismo.
- ✓ Los colores que utilizaremos para el degradado (puede haber dos o más colores).
- ✓ Con respecto a la dirección podemos utilizar las palabras clave top, bottom, left y right. También se puede utilizar un valor numérico que se expresará en grados.
- ✓ Podremos repetir un gradiente mediante la propiedad: repeating-linear-gradient().
- ✓ Con respecto al gradiente circular tenemos dos alternativas: realizar un gradiente con forma de elipse o círculo, para ello utilizaremos los términos, circle o ellipse.

### Crear un fondo con degradado.

Para realizar un fondo con degradado, tendremos la siguiente sintaxis:

- ✓ Primero, indicaremos el tipo de gradiente: linear (lineal) o radial (radial).
- ✓ Segundo, indicar los colores que utilizaremos en el degradado.
- ✓ Tercero, indicar la dirección en que se producirá dicho degradado. Para ello tendremos que asignar un punto de origen y punto de fin, aquí estaremos indicando la dirección en la que queremos aplicar el degradado. Podemos tener valores enteros, porcentajes o las palabras clave top, bottom, right y left.

Ejemplos:

#### Degradados lineales como fondo.

```
    color: red;
    background-linear-gradient(to right, red, green);
}
div {
    background-linear-gradient(to right bottom, red 10%, green 70%);
}
}
```

## Degrado lineal como fondo y con repetición.

```
color: red;
background-repeating-linear-gradient(to right, white blue black 5px);
}
```

## Degrados radiales como fondo.

```
color: red;
background-radial-gradient(circles, red 10%, green 70%);
}
div {
    background-radial-gradient(ellipse, red 10%, green 70%);
}
}
div {
    background-radial-gradient(rings, 100px 200px, yellow blue 100px);
}
}
```

A continuación se deja código donde se implementan diferentes degradados para su análisis.

### Código ejemplo.

```
<html>
<head>
    <title>Degrados</title>
</head>
<body>
    <h1>Degrados con Degradado Lineal</h1>
    <meta charset="UTF-8">
    <div style="background-color: #000; width: 100px; height: 100px; margin: auto; border-radius: 50%;>
        <div style="background-color: #fff; width: 100px; height: 100px; position: relative; border-radius: 50%;>
            <div style="position: absolute; top: 0; left: 0; width: 100%; height: 100%; background: linear-gradient(to right, red, green);>
                <div style="position: absolute; top: 0; left: 0; width: 100%; height: 100%; background: repeating-linear-gradient(to right, white blue black 5px);>
                    <div style="position: absolute; top: 0; left: 0; width: 100%; height: 100%; background: radial-gradient(circles, red 10%, green 70%);>
                        <div style="position: absolute; top: 0; left: 0; width: 100%; height: 100%; background: radial-gradient(ellipse, red 10%, green 70%);>
                            <div style="position: absolute; top: 0; left: 0; width: 100%; height: 100%; background: radial-gradient(rings, 100px 200px, yellow blue 100px);>
                                <div style="position: absolute; top: 0; left: 0; width: 100%; height: 100%; background: linear-gradient(to right bottom, red 10%, green 70%);>
                                    <div style="position: absolute; top: 0; left: 0; width: 100%; height: 100%; background: linear-gradient(to right, red, green);>
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
</body>
</html>
```

```
diagram-of-type-01
  background-color: brown;
}
diagram-of-type-02
  background-linear-gradient-to-right: red green;
}
diagram-of-type-03
  background-linear-gradient-to-right: right red green;
}
diagram-of-type-04
  background-linear-gradient-to-right: red 30% green 70%;
}
diagram-of-type-05
  background-linear-gradient-to-right bottom: red 30% green 70%;
}
diagram-of-type-06
  background-linear-gradient-to-bottom-left: red pink aqua green;
}
diagram-of-type-07
  background-linear-gradient-diagonal: 45deg pink aqua green;
}
diagram-of-type-08
  background-linear-gradient(130deg, red 30px pink 40px aqua green);
}
diagram-of-type-09
  background-repeating-linear-gradient: linear white pink blue yellow;
}
diagram-of-type-10
  background-color: black 0 0 0 0;
}
diagram-of-type-11
  background-radial-gradient(red green);
}
diagram-of-type-12
  background-radial-gradient(circles, red green);
}
diagram-of-type-13
  background-radial-gradient(farthest-side, 0% 30% 30% 30% green 70%);
}
diagram-of-type-14
  background-radial-gradient(circles closest, red pink aqua green);
}
```

• diamond-type(0)  
  background-radial-gradient(circles-slice, red, pink, aqua, green);

• 1  
  background-color: black;

• diamond-type(70)  
  background-radial-gradient(farthest-corner, red, magenta, blue, cyan, green);

• 7  
  background-image: repeating-linear-gradient(45deg, transparent, transparent 2px, black 2px, black 4px);

• 3  
  background-color: yellow;

• 6  
  background-color: white;

• 2  
  background-color: black;

• 5  
  background-color: black;

• 4  
  background-color: black;

• 8  
  background-color: black;

• 9  
  background-color: black;

• 10  
  background-color: black;

• 11  
  background-color: black;

• 12  
  background-color: black;

• 13  
  background-color: black;

• 14  
  background-color: black;

• 15  
  background-color: black;

• 16  
  background-color: black;

• 17  
  background-color: black;

• 18  
  background-color: black;

• 19  
  background-color: black;

• 20  
  background-color: black;

• 21  
  background-color: black;

• 22  
  background-color: black;

• 23  
  background-color: black;

• 24  
  background-color: black;

• 25  
  background-color: black;

• 26  
  background-color: black;

• 27  
  background-color: black;

• 28  
  background-color: black;

• 29  
  background-color: black;

• 30  
  background-color: black;

• 31  
  background-color: black;

• 32  
  background-color: black;

• 33  
  background-color: black;

• 34  
  background-color: black;

• 35  
  background-color: black;

• 36  
  background-color: black;

• 37  
  background-color: black;

• 38  
  background-color: black;

• 39  
  background-color: black;

• 40  
  background-color: black;

• 41  
  background-color: black;

• 42  
  background-color: black;

• 43  
  background-color: black;

• 44  
  background-color: black;

• 45  
  background-color: black;

• 46  
  background-color: black;

• 47  
  background-color: black;

• 48  
  background-color: black;

• 49  
  background-color: black;

• 50  
  background-color: black;

• 51  
  background-color: black;

• 52  
  background-color: black;

• 53  
  background-color: black;

• 54  
  background-color: black;

• 55  
  background-color: black;

• 56  
  background-color: black;

• 57  
  background-color: black;

• 58  
  background-color: black;

• 59  
  background-color: black;

• 60  
  background-color: black;

• 61  
  background-color: black;

• 62  
  background-color: black;

• 63  
  background-color: black;

• 64  
  background-color: black;

• 65  
  background-color: black;

• 66  
  background-color: black;

• 67  
  background-color: black;

• 68  
  background-color: black;

• 69  
  background-color: black;

• 70  
  background-color: black;

• 71  
  background-color: black;

• 72  
  background-color: black;

• 73  
  background-color: black;

• 74  
  background-color: black;

• 75  
  background-color: black;

• 76  
  background-color: black;

• 77  
  background-color: black;

• 78  
  background-color: black;

• 79  
  background-color: black;

• 80  
  background-color: black;

• 81  
  background-color: black;

• 82  
  background-color: black;

• 83  
  background-color: black;

• 84  
  background-color: black;

• 85  
  background-color: black;

• 86  
  background-color: black;

• 87  
  background-color: black;

• 88  
  background-color: black;

• 89  
  background-color: black;

• 90  
  background-color: black;

• 91  
  background-color: black;

• 92  
  background-color: black;

• 93  
  background-color: black;

• 94  
  background-color: black;

• 95  
  background-color: black;

• 96  
  background-color: black;

• 97  
  background-color: black;

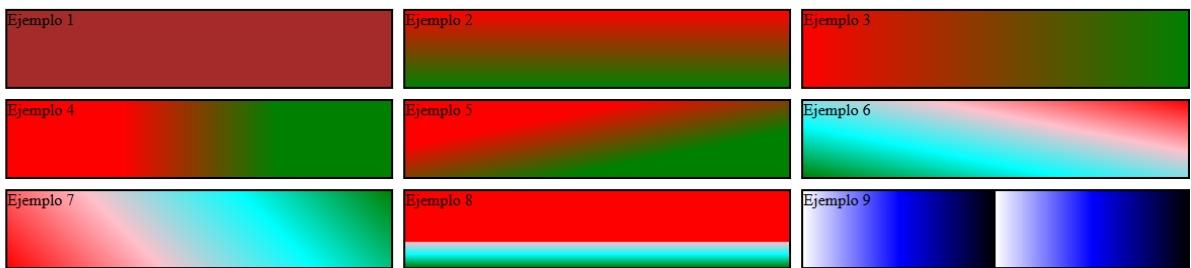
• 98  
  background-color: black;

• 99  
  background-color: black;

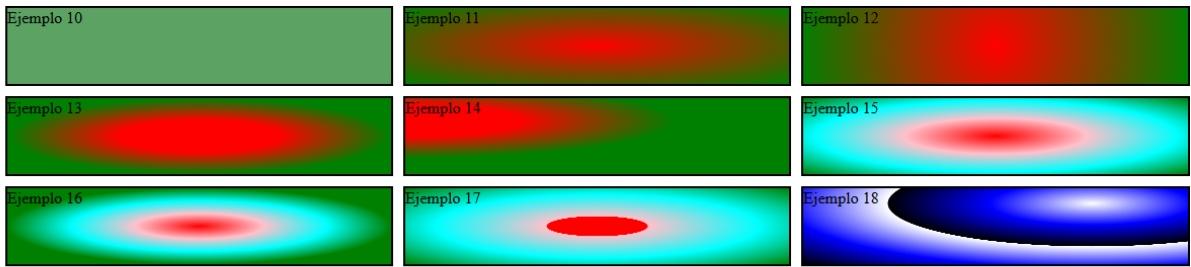
• 100  
  background-color: black;

## Resultado.

## Degrados lineales



## Degrados Radiales



Elaboración propia.. *Fondos y degradados.* ([CC BY-SA](#))



## Para saber más

Como vimos en la primera unidad didáctica parte del trabajo de la creación de una guía de estilo era la definición de la paleta de colores. Estos forman parte de todo nuestro desarrollo. Si dichos colores pudiéramos definirlos con variables, el tratamiento y posterior modificación de los mismos sería mucho más rápido, en el siguiente enlace se muestra como trabajar con variables en [CSS](#).

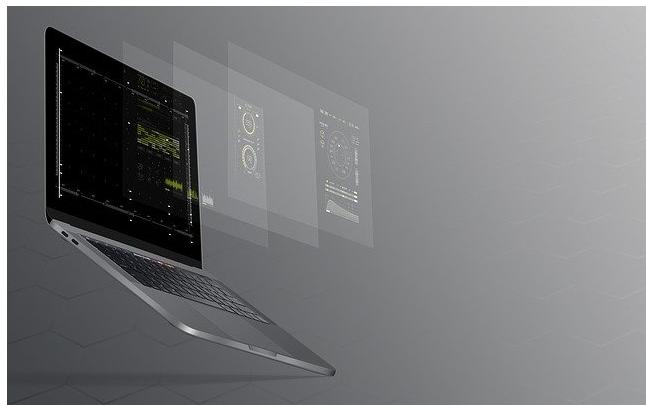
[Declaración y uso de variables en CSS.](#)

## **4.2.- Imágenes de fondo**

---

Las imágenes las podrás configurar de forma muy completa, ya que además de poner una imagen como fondo de una página, podrás ajustar su punto de partida, su patrón de repetición, su posición dentro de un elemento cualquiera y lograr que permanezca fija en esa posición aunque se mueva el resto del documento. Para realizar esta configuración detallada se emplean las propiedades: `background-image`, `background-origin`, `background-size`, `background-clip`, `background-repeat`, `background-position`, `background-attachment` y `background`.

Pasamos a describir estas propiedades.



TayebMEZAHIDIA. Licencia Pixabay

#### ✓ `background-image`

Esta propiedad sirve para configurar la imagen de fondo de cualquier elemento. Actualmente se puede añadir más de una imagen de fondo a nuestra página.

#### ✓ `background-origin`

Indica la posición de origen desde que comenzará la imagen. Se tiene como referencia la esquina superior izquierda. los valores que podrá tener son: `content-box`, `padding-box` y `border-box`, además de los valores `initial` (por defecto) e `inherit` (heredado). En el siguiente enlace se muestra un pequeño simulador donde se puede ver como funciona dicha propiedad.



[Simulador `background-origin`.](#)

#### ✓ `background-size`

Nos permite especificar el tamaño de la imagen, para ello podemos utilizar diferentes valores:

- ◆ `auto` (valor por defecto): muestre el tamaño original de la imagen.
- ◆ `dimensiones`: especificamos el ancho y el alto que ocupará la imagen de fondo.
- ◆ `porcentaje`: similar al anterior parámetro pero lo hacemos de forma porcentual teniendo como referencia el contenedor padre.
- ◆ `contain`: escala la imagen al mayor tamaño posible de la misma sin recortarla ni ampliarla.
- ◆ `cover`: escala la imagen al mayor tamaño posible, sin deformarla, de tal forma que ocupe todo el espacio posible. La imagen recorta aquello que sobresalga sobre el espacio de fondo del elemento.
- ◆ `fixed`: se puede aplicar junto con los dos valores anteriores y hará que la imagen permanezca fija y no se desplace.

Sí

[generador de la propiedad `background-size`.](#)

#### ✓ `background-clip`

Esta propiedad nos permite indicar hasta donde va a llegar el fondo (imagen o color), los valores que puede tomar son parecidos a los vistos en la propiedad `background-origin`: `content-box`, `padding-box` y `border-box`, además de los valores `initial` (por defecto) e `inherit` (heredado).

**La propiedad background-image prevalece sobre la propiedad background-color por lo que si con un elemento realizas declaraciones con estas dos propiedades ignorará la declaración de background-color.**

En el siguiente ejemplo se muestra la configuración de la imagen de fondo, usando un archivo de imagen de nombre fondo.gif, en todo el documento excepto para el párrafo que tiene un identificador “cabecera” que tendrá como imagen de fondo un archivo de imagen de nombre “fondo-cuerpo.gif”:



#### ✓ **background-repeat<br />**

Permite configurar a tu gusto la forma en la que se repetirá la imagen cuando su tamaño sea más pequeño que la ventana del navegador. También permite evitar que la imagen se repita. Ejemplos:



Con el primer ejemplo se consigue que la imagen aparezca sólo una vez, con el segundo ejemplo la imagen se repetirá a lo ancho (horizontalmente) tantas veces como necesite y con el tercer ejemplo la imagen de fondo se repetirá a lo largo del documento (verticalmente).

## ✓ background-position

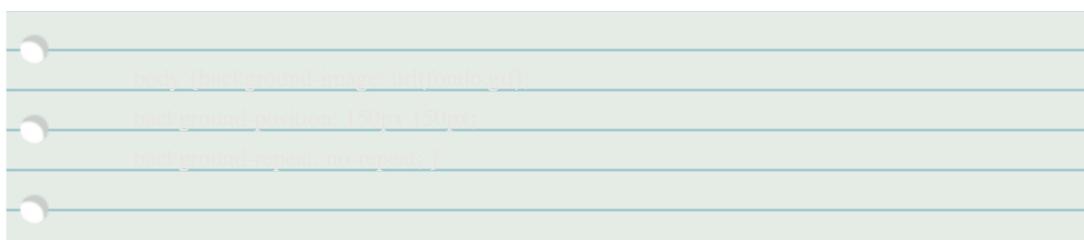
Especifica la posición de la primera imagen que cubrirá el fondo del elemento en el que esté definida esta propiedad. Al posicionamiento podemos asignarles los valores: **top**, **bottom**, **left** y **right** y **center** los cuales se usan, normalmente, de dos en dos sin importar el orden (uno indica su posición horizontal y el otro indica su posición vertical). Si sólo se indica un valor, se supone que el otro será **center**. Se pueden utilizar diferentes unidades de medida, tanto valores absolutos (px) como valores relativos (%). Recordar que para indicar una posición con coordenadas hay que indicar dos valores, uno para el eje X y otro para el eje Y.

En el siguiente ejemplo, se configura el fondo del elemento body con una imagen llamada "fondo.gif" que aparecerá sólo una vez en la parte superior central del cuerpo del documento:

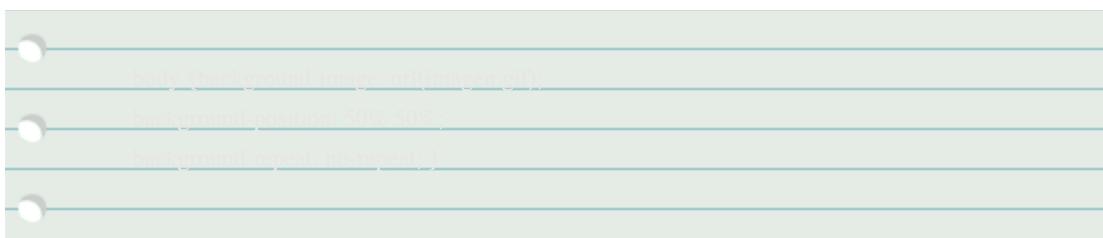


Para el posicionamiento se pueden emplear también las medidas de longitud vistas anteriormente. En este caso, las medidas son relativas al extremo superior izquierdo del elemento.

En el siguiente ejemplo, se configura el mismo fondo de antes que aparecerá a 150 píxeles de la esquina superior izquierda del cuerpo del elemento (horizontal y verticalmente):



También puedes utilizar los valores porcentuales, así, si indicas sólo un valor se asume que el otro es un 50%. Debes tener en cuenta que el valor porcentual se aplica "al contenedor y a la imagen en sí". En el siguiente ejemplo se muestra la regla de estilo que logra que la imagen quede colocada en el centro del elemento body. Ejemplo:



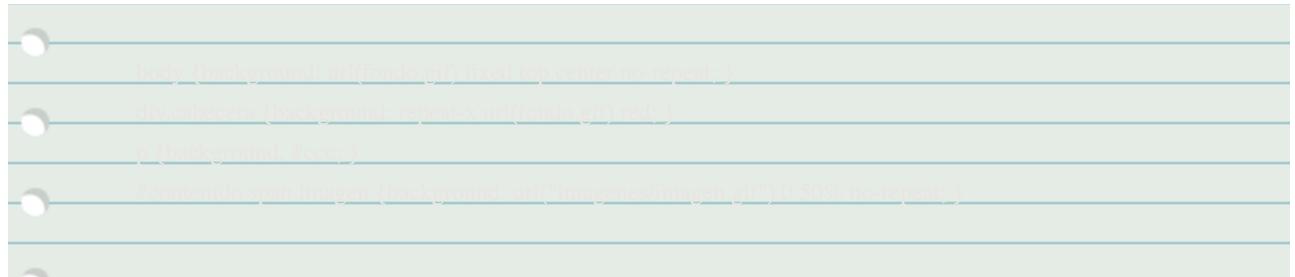
## ✓ background-attachment

Con esta propiedad puedes fijar la imagen en una posición concreta. Se le pueden asignar los valores: **scroll**, **fixed** e **inherit**, siendo scroll el valor por defecto. Se suele emplear el valor **fixed** para conseguir que la imagen no se desplace con el documento.

## ✓ **background.**

Esta propiedad permite configurar todas las propiedades de fondo vistas anteriormente usando una única declaración, de forma similar a lo que ocurría con la propiedad font, pero a diferencia de ésta, no tiene ninguna propiedad obligatoria y sus valores pueden aparecer en cualquier orden.

Sólo debes tener una restricción: la posición se indica con dos valores que deben aparecer juntos, primero el horizontal, seguido inmediatamente después por el vertical, ya que si sólo aplicamos un valor el otro se configura por defecto a center. Ejemplos:



A continuación se deja un código aplicando las diferentes propiedades vistas en este apartado. La imagen a la que hace referencia el ejemplo es la misma que la que se ha utilizado en el apartado 1.4.

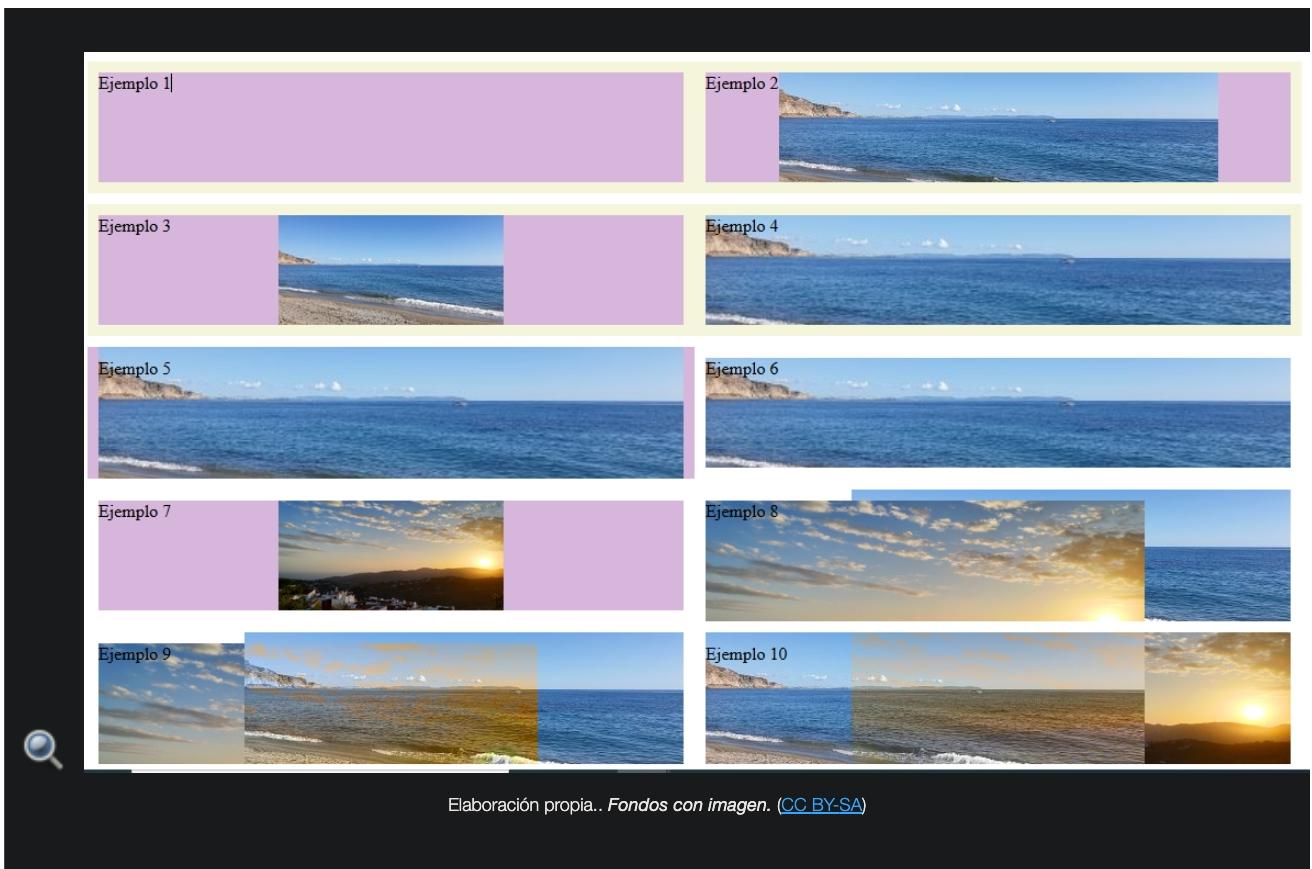
### **Código ejemplo.**

```
background-color: #ccc;
background-size: 100px 100px;
background-repeat: repeat;
background-attachment: fixed;
background-image: url('https://www.w3schools.com/html/pic_mountain.jpg');

body {
    background-color: #ccc;
    background-size: 100px 100px;
    background-repeat: repeat;
    background-attachment: fixed;
    background-image: url('https://www.w3schools.com/html/pic_mountain.jpg');
}
```







## Ejercicio resuelto

Queremos realizar un documento web donde el contenedor con id=pie tiene las siguientes características:

- ✓ Relleno superior e inferior de 0.5em.
- ✓ Relleno derecho e izquierdo de 0em.
- ✓ Margen superior de 1em.
- ✓ Bordes superior e inferior de tipo sólido, de 1 píxel de grosor y de color #C5C5C5.
- ✓ Fondo de color #F8F8F8.

¿Cuál sería la regla CSS que consigue esto?

[Mostrar retroalimentación](#)



## Para saber más

Otra propiedad que podemos utilizar en relación con la configuración de nuestro fondo del elemento es **background-blend-mode**. Con esta propiedad podemos indicar cómo se fusionan diferentes imágenes de fondo, así podemos combinar dos imágenes o una imagen y un color de fondo. Esta propiedad admite múltiples valores. Dejamos un enlace para profundizar un poco más sobre esta propiedad y los diferentes valores que puede tener.



[Valores de la propiedad background-blend-mode.](#)

## 4.3.- Opacidad y filtros



### Reflexiona

Si te fijas en las ventanas, persianas y cortinas de una casa, verás que algunas ventanas como las de los cuartos de baño son de cristal pero no son transparentes, dejan ver el exterior pero no de forma nítida. Cuando se bajan las persianas, no se ve absolutamente nada del exterior, y las cortinas, dependiendo de su tejido, pueden dejar pasar la claridad en mayor o menor medida.

Al comienzo de este apartado hemos visto cómo al indicar el color a utilizar con un elemento, teníamos posibilidades de indicar el color y a la vez la opacidad del mismo, para ello podemos definir un color mediante las funciones `rgba()` y `hsla()`. Pero antes quizás, tendríamos que haber definido en qué consiste la opacidad. Así podemos decir que la opacidad es una característica de los elementos que nos permite mostrar, o no, otros elementos que tengan por debajo. Para conseguir efectos de transparencia en algunos elementos tienes las siguientes propiedades: `opacity` y `filter`.

- ✓ `opacity`   
 Esta propiedad, que es compatible con todos los navegadores que soporten CSS3, permite asignar valores comprendidos entre 0 (invisible o totalmente transparente) y 1 (totalmente opaco).
- ✓ `filter`   
 Permite definir diferentes efectos como son: degradaciones, desenfocados, sombras, etcétera. Para lograr la transparencia hay que aplicar el filtro `alpha`, con valores entre 0 y 100. Este tipo de filtros se suelen utilizar preferentemente con las imágenes. Algunos de los filtros que podemos utilizar son:
  - ◆ `blur`: desenfoque.
  - ◆ `grayscale`: nivel de grises, valores en porcentaje 0% original y 100% elemento en color blanco y negro.
  - ◆ `sepia`: color sepia valores en porcentaje 0% original y 100% elemento en color sepia.
  - ◆ `opacity`: nivel de opacidad 0% tendremos un elemento invisible y 100% visible.
  - ◆ `sature`: grado de saturación, tendrá el valor 0% para el blanco y negro y 100% para el valor original. Se puede aplicar valores superiores a 100%.
  - ◆ `hue-rotate`: rotación en el color matiz, se vio en el punto 4 de la unidad. Su valor se expresa en grados.
  - ◆ `invert`: invierte los colores del elemento, con un valor 0% se deja la imagen como está y con el valor 100% se invierten los colores completamente.
  - ◆ `brightness`: podemos definir el brillo, admite valores iguales que `sature`.
  - ◆ `drop-shadow`: está relacionado con la sombra de un elemento, en el siguiente apartado veremos como se aplican sombras a diferentes elementos y nos puede servir un poco mejor como aplicar este efecto.



Lenore Edman CC BY-SA

En el ejemplo siguiente tienes el código donde se configura la opacidad de algunos elementos.

```
const styles = {
    container: {
        display: "flex",
        flexWrap: "wrap"
    },
    card: {
        margin: "10px",
        width: "300px",
        height: "300px",
        position: "relative"
    },
    cardImage: {
        width: "100%",
        height: "100%",
        filter: "blur(10px)"
    },
    cardContent: {
        position: "absolute",
        top: "0",
        left: "0",
        width: "100%",
        height: "100%",
        background: "white",
        display: "flex",
        align-items: "center",
        justify-content: "center",
        color: "black",
        opacity: "0.9"
    }
};
```

Para comprender un poco mejor cómo se aplican los filtros a una imagen dejamos un [simulador interactivo](#).



# Autoevaluación

A la vista del ejemplo anterior, ¿cuáles de las siguientes afirmaciones son correctas?

- El color del fondo de toda la página es blanco.
- El color de fondo del saludo es blanco.
- El color de fondo de la frase es blanco
- El color de fondo de la despedida es blanco.
- El saludo no se ve.
- La frase no se ve.
- La despedida no se ve.
- El fondo del saludo es el mismo que el fondo de la página.
- El fondo del texto es el mismo que el de la página.
- El fondo de la despedida es el mismo que el de la página.

[Mostrar retroalimentación](#)

Algunos filtros más que podemos aplicar con la propiedad filter son: gris, contraste, brillo, etcétera. Además dichos filtros pueden combinarse, aplicando varios a la vez. En el siguiente enlace vemos las posibilidades que nos ofrece dicha etiqueta.  [Ejemplos de la propiedad filter.](#)

A continuación, y partiendo del ejemplo anterior, mostramos otro código para probar:

Este código muestra un efecto de filtro en la frase "Hello World".  
El efecto es una mezcla entre el efecto "grayscale" y el efecto "sepia".  
El resultado es una imagen en tonos marrones y grises.  
Este efecto se aplica tanto al saludo como a la despedida.  
El efecto se aplica tanto a la frase como a la despedida.  
El efecto se aplica tanto a la frase como a la despedida.

```
    <Image> { background-color: white;
      width: 150px; height: 150px;
      filter: brightness(100%); }

    <Image> { background-color: black;
      width: 150px; height: 150px;
      filter: brightness(0%); }

    <Image> { background-color: black;
      width: 150px; height: 150px;
      filter: brightness(50%); }

    <Image> { background-color: white;
      width: 150px; height: 150px;
      filter: brightness(150%); }

    <Image> { background-color: blue;
      width: 150px; height: 150px;
      filter: opacity(30%) brightness(200%); }

    <Image> { background-color: red;
      width: 150px; height: 150px;
      filter: drop-shadow(0px 0px 10px); }

    <Image> { background-color: green;
      width: 150px; height: 150px;
      filter: blur(10px); }

    <Image> { background-color: yellow;
      width: 150px; height: 150px;
      filter: grayscale(100%); }

    <Image> { background-color: cyan;
      width: 150px; height: 150px;
      filter: invert(1); }

    <Image> { background-color: magenta;
      width: 150px; height: 150px;
      filter: contrast(1000%); }

    <Image> { background-color: orange;
      width: 150px; height: 150px;
      filter: sepia(100%); }

    <Image> { background-color: purple;
      width: 150px; height: 150px;
      filter: hue-rotate(180deg); }

    <Image> { background-color: pink;
      width: 150px; height: 150px;
      filter: invert(1) sepia(100%); }

    <Image> { background-color: limegreen;
      width: 150px; height: 150px;
      filter: invert(1) sepia(100%) contrast(1000%); }

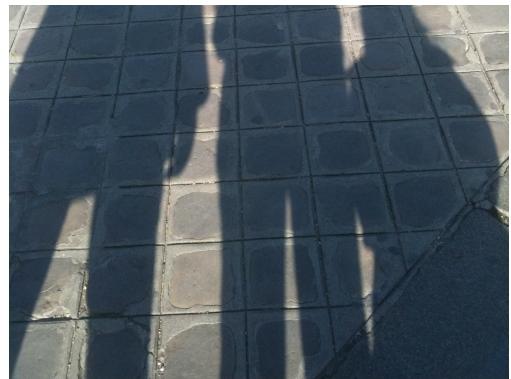
    <Image> { background-color: lightblue;
      width: 150px; height: 150px;
      filter: invert(1) sepia(100%) contrast(1000%) hue-rotate(180deg); }
```



## 4.4.- Sombras

---

Para terminar esta sección vamos a indicar cómo podemos aplicar sombras a un texto y a una caja, para ello utilizaremos las propiedades box-shadow y text-shadow.



[piasels.com](#) Sombras (CC0)

## ✓ box-shadow

Esta propiedad permite poner una sombra alrededor de un objeto, para definir la sombra tendremos que indicar las coordenadas X e Y donde vamos a posicionar ésta. Si dichas coordenadas son positivas la sombra se propagará hacia abajo y hacia la derecha, con valores negativos la sombra irá hacia arriba y hacia la izquierda. Indicamos la posición donde se colocará la sombra con respecto al elemento.

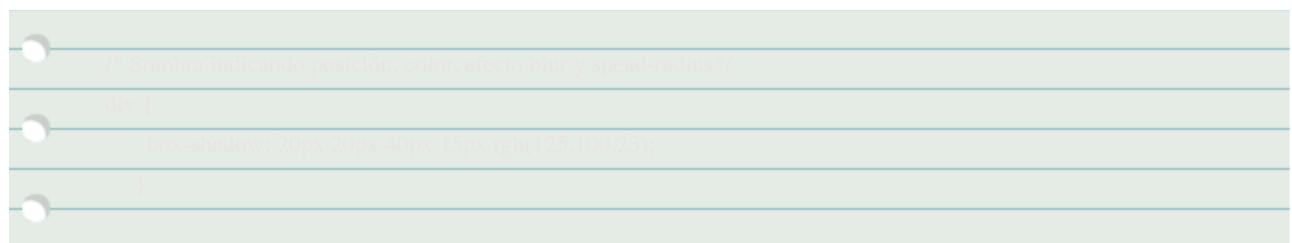
Posteriormente tendremos que indicar el grado de desenfoque o valor del parámetro blur, cuanto mayor sea éste, mayor será el difuminado (solo admite valores positivos).

Otro parámetro que podemos definir se denomina spread-radius y nos permite indicar si lo que queremos es agrandar o achicar la sombra con respecto al objeto que la está proyectando. El valor por defecto es 0, lo que significa que la sombra es igual que el objeto.

Por defecto la sombra se aplica hacia afuera. Para que la sombra se comporte de forma diferente y se expanda hacia adentro de la caja tendremos que utilizar la propiedad inset. En caso de utilizar esta propiedad (por defecto no se aplica), la sombra se aplicará por encima del fondo y por debajo del contenido.

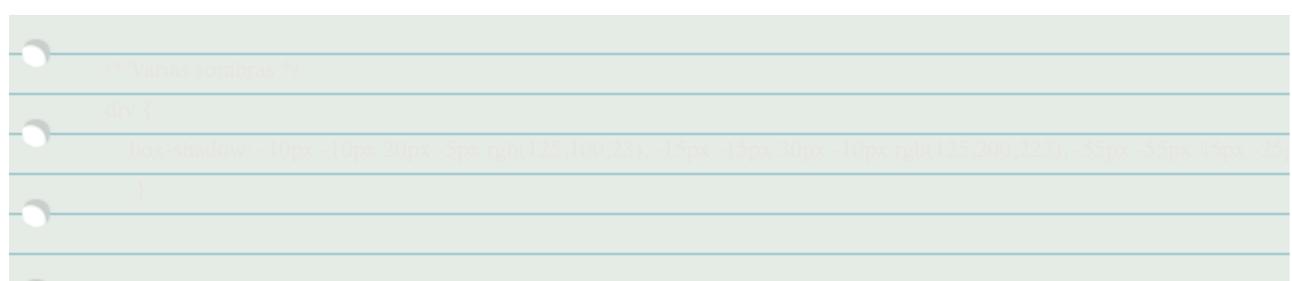
Por último nos faltaría indicar el color a aplicar a la sombra.

Hemos visto que podemos utilizar diferentes parámetros, así solo serán obligatorios los relacionados con la posición. En caso de incluir un tercer parámetro éste se entenderá como el parámetro blur y si se indica un cuarto parámetro se tomará como el parámetro spread-radius.



La mejor forma de ver cómo funciona esta propiedad es viendo los diferentes valores y efectos que puede tomar. Para ello dejamos otro generador de sombras de cajas como ejemplo:  [generador propiedad box-shadow](#).

Para finalizar este apartado, indicar que se puede aplicar más de una sombra a un elemento. Para eso lo único que tendremos que hacer es poner cada una de las sombras separadas por comas: box-shadow: sombra1, sombra2, sombra3,...; Ejemplo:



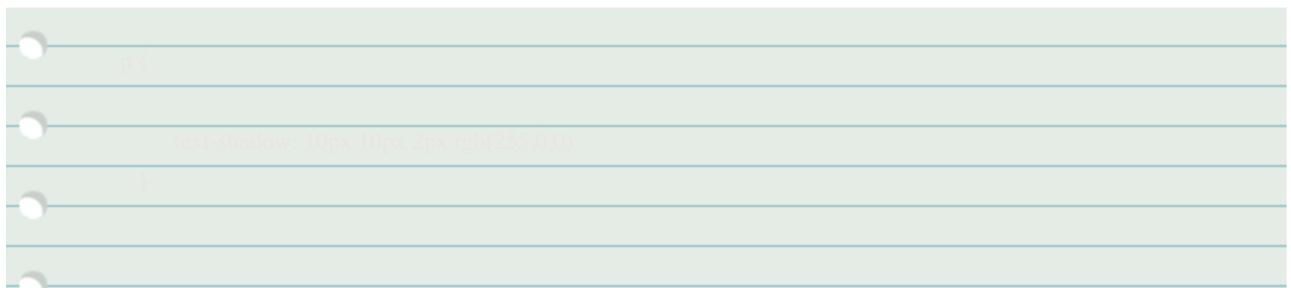
# Autoevaluación

**La sombra de una caja solo puede propagarse hacia abajo y a la derecha.**

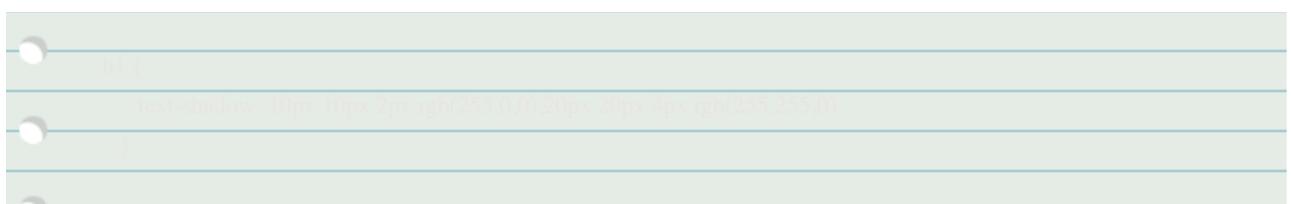
- Verdadero
- Falso



Al igual que hemos añadido sombras a una caja, podremos añadir sombras a un texto. Esta propiedad funciona de forma similar a box-shadow, pero es un poco más simple. Así tendremos que indicar la posición de la sombra, el difuminado y el color. Solo será obligatorio indicar la posición de la sombra. Recordemos que valores positivos sitúan la sombra abajo y a la derecha y valores negativos arriba y a la izquierda. Ejemplo:



Se puede aplicar más de una sombra a la propiedad text-shadow, para ello tendremos que separarlo por comas.



Para poder practicar esta propiedad dejamos el siguiente  [generador de text-shadow](#).

A continuación se deja un código ejemplo y su resultado con las propiedades vistas sobre las sombras.

## Código ejemplo.



background-color:

background-color:

background-color: transparent;

background-color: transparent;

background-color:

}

/> Sombra sin indicando posición, color y distancia

background-color: #333;

background-color: #333; background-color: #333;

background-color: #333;

background-color: #333; background-color: #333;

}

/> Sombra solo indicando posición, color y distancia blanca

background-color: #333;

background-color: #333; background-color: #333;

background-color: #333;

background-color: #333; background-color: #333;

}

/> Sombra indicando posición, color, distancia y spread-radius

background-color: #333;

background-color: #333; background-color: #333;

background-color: #333;

background-color: #333; background-color: #333;

}

/> Sombra indicando posición, color, distancia blanca y spread-radius negativo

background-color: #333;

background-color: #333; background-color: #333;

background-color: #333;

background-color: #333; background-color: #333;

}

/> Sombra indicando posición, distancia

background-color: #333;

background-color: #333; background-color: #333;

background-color: #333;

background-color: #333; background-color: #333;

}

/> Sombras

background-color: #333;

background-color: #333; background-color: #333;

background-color: #333;

background-color: #333; background-color: #333;

}

```
    <!-- Se aplican los estilos de fondo -->
    <body>
        <div style="text-align: center;">
            <h1>¡Buenas tardes!</h1>
            <p>Este es un sitio web que muestra una página web con un diseño simple y responsive. La página incluye un encabezado con el título "¡Buenas tardes!", un pie de página con el copyright "© 2023" y un pie de página con el copyright "© 2023". La página también incluye un pie de página con el copyright "© 2023".</p>
        </div>
    </body>
```

## Resultado.

## Sombras

Ejemplo 1  
*Ejemplo 1*

Ejemplo 2  
*Ejemplo 2*

Ejemplo 3  
*Ejemplo 3*

Ejemplo 4  
*Ejemplo 4*

Ejemplo 5  
*Ejemplo 5*

Ejemplo 6  
*Ejemplo 6*

Elaboración propia. ([CC BY-SA](#))

## Autoevaluación

¿Es posible aplicar a una etiqueta span tres sombras a la vez?

- Verdadero
- Falso

## 5.- Maquetación (Layout)



### Caso práctico



Elaboración propia (Uso Educativo no comercial).

Para la web de "**Migas Amigas**" el equipo de BK programación decidió al final seleccionar un diseño de dos columnas clásico, para toda la página a excepción de la página de portada.

Para realizar este diseño será necesario que **Carlos** comprenda cómo flota y se posiciona cada uno de los elementos que aparecen en las webs.

**Juan** le explica las propiedades y técnicas para flotar y posicionar los elementos, **Carlos** va tomando conciencia de lo que esto implica y piensa que estas herramientas son suficientes para posicionar una web, pero la cosa no queda aquí. Posteriormente **Juan**, continua explicando y le indica a **Carlos** que existen otras técnicas que permiten crear web que puedan adaptarse a diferentes dispositivos y que pueden facilitar el diseño responsive, como son flexbox y grid.

**Carlos** piensa: "Sin duda, me voy a tener que dedicar a fondo".

Como ya se ha comentado en alguna ocasión en la unidad, CSS es algo que está en continua evolución, adaptándose a los cambios tecnológicos que existen en nuestro día a día. El ejemplo más claro de esta evolución es cómo nuestras páginas web hace unos años solo se visualizaban en ordenadores personales o portátiles, pero sin embargo hoy en día, las páginas web son más visitadas desde dispositivos móviles que desde ordenadores, con las consecuencias que este cambio de dispositivos conllevan en el diseño de una web.

Habitualmente una página web se compone de los siguientes elementos principales:

- ✓ Encabezado.
- ✓ Zona de navegación.
- ✓ Cuerpo principal.
- ✓ Pie.

Esto es lo que conformará nuestra página o maqueta y es lo que tendremos que aprender a posicionar.

Sin extendernos mucho indicaremos cómo han ido evolucionando las diferentes **técnicas de maquetación**. Así, en un inicio ésta se realizaba con **marcos y tablas**, estas dos técnicas están hoy en día totalmente desaconsejadas por múltiples motivos. Posteriormente, estas técnicas fueron sustituidas por el **posicionamiento** (que nos permitía sacar un elemento de su flujo normal) y posteriormente la **flotación**, hasta llegar a las técnicas más actuales como son el modelo de **cajas flexible (flexbox)** y más reciente todavía el **grid**, que nos permitirá definir diferentes regiones en nuestra página web.

En los dos puntos siguientes nos centraremos en las técnicas un poco "más viejas" como son el **flotado y el posicionamiento**. Éstas nos permiten tener el máximo control sobre el lugar que ocupa cada elemento en una página web, sus condiciones de visibilidad y "flotabilidad", así como controlar el manejo de capas. El principal inconveniente de estas técnicas hoy en día, viene dado por lo que se ha comentado anteriormente y es la visualización de nuestra web, que debe estar preparada para adaptarse a diferentes tipos de dispositivos, lo que se denomina como **diseño responsivo o adaptativo**. Hacer una página web responsive con estas técnicas es más complicado.

Como veremos al final de la unidad las técnicas de **flexbox y grid** sí permiten realizar **diseños responsivos** de una forma mucho más cómoda y rápida.

Para terminar, resaltar que aunque haya técnicas más modernas, no significa que no haya que conocer y saber dominar técnicas más antiguas como el flotado y el posicionamiento, ya que en una web pueden convivir diferentes tecnologías, bien porque nos interese o porque haya que mantener desarrollos ya creados.



FirmBee | Licencia Pixabay

**Un término que aparecerá a menudo es “flujo normal”.**

**Cuando hablamos de que los objetos de una página siguen el flujo normal del documento, queremos indicar que la forma en la que se disponen en la ventana del navegador coincide con el lugar que ocupan en el documento escrito (en el código HTML), donde el orden de lectura es de arriba a abajo y de izquierda a derecha.**

**Flotando y posicionando con CSS conseguimos que los elementos abandonen su flujo normal. De esta forma un elemento que esté en el documento escrito más abajo que otro en el documento puede verse en el navegador por encima de él.**

## 5.1.- Posicionamiento

---

Comenzaremos estudiando el modelo de posicionamiento y del formato visual para poder tener una visión completa de cómo se organizan los elementos en una página. Para ello vamos a ver algunas de las propiedades que nos permiten organizar los elementos:

- ✓ **display<br /><br />**

Esta propiedad entre otras muchas cosas, permite al documento interpretar de otra forma los elementos de tipo bloque y los elementos de tipo línea. Ya se vio en un apartado anterior cómo podemos cambiar el comportamiento de un elemento bloque para que se comporte como un elemento en línea (`display:inline-block`) y viceversa (`display:block`). Si aplicamos a esta propiedad el valor `none`, no genera caja para el elemento, no muestra su contenido y no ocupa espacio en la página.

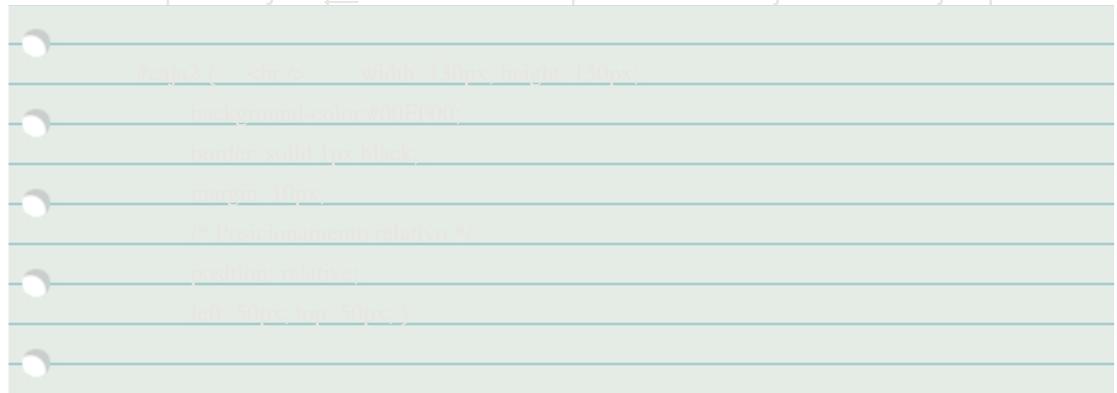


[billac CC BY](#)

#### ✓ position<br /><br />

Permite posicionar los elementos en un documento. Esta propiedad admite cinco valores:

- ◆ static: que permite colocar los elementos según el flujo normal. Es el valor que asumirá por defecto en todos los elementos HTML.
- ◆ relative: permite dejar el elemento exactamente donde está. Un elemento posicionado de esta forma se puede cambiar desde su punto de partida estableciendo una distancia vertical y/o horizontal. En el siguiente ejemplo se desplaza la "caja2" 50px del extremo izquierdo y 50px del extremo superior de su flujo normal. Ejemplo:



**Un elemento posicionado relativamente, que siga en el flujo normal del HTML inmediatamente después a otro elemento posicionado también relativamente, calculará su origen de la forma siguiente:**

- ◆ Si el elemento es hijo del anterior, su origen estará en el final del anterior (su padre).
- ◆ Si el elemento no es hijo del anterior, tendrá su origen donde el anterior tenga su final si no se fijaron valores distintos de cero en sus propiedades top y left.

- ◆ absolute: permite abandonar el flujo normal haciendo que el elemento no ocupe ningún espacio de forma que el resto de elementos del flujo normal actuarán como si el elemento no estuviese allí. El modo de determinar el origen de coordenadas de nuestro elemento será el siguiente:
  - Si el elemento posicionado de modo absoluto no está contenido dentro de ningún otro, su origen de coordenadas se mide respecto a la esquina superior izquierda del body (contenedor principal).
  - Si el elemento posicionado de modo absoluto está contenido dentro de otro elemento, el origen de coordenadas del elemento se calcula con respecto a la posición de la esquina superior izquierda del elemento que lo contiene.
- ◆ fixed: funciona de forma parecida al posicionamiento absoluto pero se posiciona con respecto a la ventana del navegador apareciendo en la misma posición aunque el usuario se desplace por la página con las barras de desplazamiento.
- ◆ sticky: este nuevo valor podemos decir que es una mezcla de los valores relative y fixed. Así, el elemento inicialmente se posiciona en su flujo normal. Su comportamiento lo cambia cuando se llega a una determinada altura o posición vertical (especificada por la propiedad top en nuestro caso) y a partir de ahí permanece en una posición fija. Así con la propiedad top especificamos la separación que habrá hasta la ventana principal (o elemento padre), cuando se llegue a esa distancia, el elemento permanecerá con esa distancia fija aunque hagamos scroll. Podemos utilizar las propiedades bottom, left o right.

✓ visibility<br /><br />

Esta propiedad controla si el elemento será visualizado según le asigne el valor `visible` o `hidden`. Debes tener en cuenta que, aunque un elemento no sea visible, éste continúa ocupando su espacio en el flujo normal del documento, al contrario de lo que ocurría con la propiedad `display` cuando se le asignaba el valor `none`.

✓ z-index<br /><br />

Permite controlar el orden en el que se presentan los elementos que quedan solapados por efecto de otras propiedades. Si cuando definimos algún elemento con posición absoluta, éste tiene que visualizarse en el mismo lugar ocupado por otro elemento, se producirá una superposición de elementos visualizándose, en la parte coincidente, sólo el que está ocupando la "posición superior". La propiedad `z-index` permite especificar el orden en el eje Z (profundidad) de los elementos, esto es, el orden de apilamiento en capas del documento.

Por defecto, los elementos se apilan en el orden en que aparecen: el elemento situado más abajo en el flujo normal del documento quedará encima. Si quieres que esta posición no sea tenida en cuenta, debes saber que los elementos con un valor mayor de la propiedad `z-index` son colocados encima de los que tienen un valor menor `z-index`, quedando estos últimos tapados por los primeros.

También debes saber que esta propiedad sólo se aplica a elementos que tengan la propiedad `position` con valor `absolute`, `sticky` o `relative`.



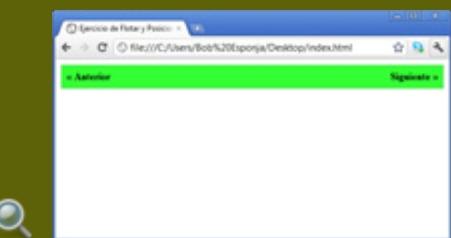


## Ejercicio resuelto

A partir del siguiente código HTML:

```
<ul style="list-style-type: none; padding-left: 0; margin: 0; border: 1px solid black; border-radius: 10px; background-color: #f0f0f0; width: fit-content; height: fit-content; position: absolute; left: 50%; top: 50%; transform: translate(-50%, -50%);>  
    <li><a href="#">Home</a></li>  
    <li><a href="#">Nosotros</a></li>  
    <li><a href="#">Contacto</a></li>  
    <li><a href="#">Ayuda</a></li>  
    <li><a href="#">Ayuda al cliente</a></li>  
    <li><a href="#">Ayuda a los desarrolladores</a></li>  
    <li><a href="#">Ayuda a los administradores</a></li>  
    <li><a href="#">Ayuda a los usuarios</a></li>  
    <li><a href="#">Ayuda a los administradores de la web</a></li>  
    <li><a href="#">Ayuda a los desarrolladores de la web</a></li>  
    <li><a href="#">Ayuda a los usuarios de la web</a></li>  
    <li><a href="#">Ayuda a los administradores de la web de la web</a></li>  
    <li><a href="#">Ayuda a los desarrolladores de la web de la web</a></li>  
    <li><a href="#">Ayuda a los usuarios de la web de la web</a></li>  
</ul>
```

Debes determinar las reglas CSS que pondrías en el elemento `style`, con las clases, identificadores y elementos que creas necesarios y modificar el código para que el resultado sea lo más parecido posible a la siguiente imagen:



Elaboración propia. ([CC0](#))

[Mostrar retroalimentación](#)

Para finalizar este apartado se deja un nuevo ejemplo, donde se posicionan diferentes objetos. Para comprobar su funcionamiento copia el código y ejecútalo. Se han incluido bloques verticalmente para hacer que exista un scroll en la pantalla y se pueda ver cómo funciona la propiedad `position` con el valor `sticky`.

En dicho ejemplo se podrá ver cómo funcionan además de todos los valores de `position` las propiedades `display`, `visibility` y `z-index`. En los comentarios del código puedes conocer la explicación de cada propiedad utilizada.

## Código ejemplo.

```
 1<html>
 2<head>
 3  <meta charset="UTF-8">
 4  <meta name="viewport" content="width=device-width, initial-scale=1.0">
 5  <title>Primeras propiedades CSS</title>
 6</head>
 7<body>
 8  <div id="block1">
 9    <P>Párrafo dentro de nuestro primer elemento y le cambiaremos el fondo!</P>
10    width: 70px;
11    height: 10px;
12    border: black 1px solid;
13  </div>
14  <div id="block2" style="display: inline-block; vertical-align: top;">
15    <P>Establecemos un conjunto de bloques mediante la clase horizontal como bloques en linea!</P>
16    <div id="block2_1" style="display: inline-block; vertical-align: top; width: 100px; height: 200px; border: black 1px solid; position: relative; top: 20px; left: 20px; background-color: #9999ff; margin-right: 20px;">
17      <P>Bloque 2_1 es hijo de su padre horizontal!</P>
18      <div id="block2_1_1" style="position: absolute; top: 10px; left: 10px; background-color: #99ff99; width: 100px; height: 100px;">
19        <P>Este bloque está en la misma posición que el Bloque 2_1!</P>
20        <div id="block2_1_1_1" style="position: absolute; top: 10px; left: 10px; background-color: #9999ff; width: 50px; height: 50px;">
21          <P>Este bloque estará en la misma posición que el Bloque 2_1_1!</P>
22          <div id="block2_1_1_1_1" style="position: absolute; top: 10px; left: 10px; background-color: #99ff99; width: 50px; height: 50px;">
23            <P>Este bloque es hermano del anterior!</P>
24            <div id="block2_1_1_1_1_1" style="position: absolute; top: 10px; left: 10px; background-color: #9999ff; width: 50px; height: 50px;">
25              <P>Este bloque es hermano del anterior!</P>
26            </div>
27          </div>
28        </div>
29      </div>
30    </div>
31  </div>
32</body>
```

```
left: 100px;
background-color: yellow;
```

•

```
position: absolute;
```

•

```
top: 100px;
left: 100px;
```

•

```
background-color: pink;
```

•

```
position: relative;
```

•

```
position: absolute;
top: 100px;
left: 100px;
background-color: pink;
```

•

```
position: absolute;
```

•

```
position: absolute;
top: 100px;
left: 100px;
background-color: limegreen;
```

•

```
position: absolute;
```

•

```
position: absolute;
top: 100px;
left: 500px;
background-color: blue;
```

•

```
position: absolute;
top: 100px;
left: 500px;
background-color: orange;
```

•

```
position: absolute;
top: 100px;
left: 500px;
background-color: red;
```

```
    <div class="row">
      <div class="col-12 col-md-6" style="background-color: #f2f2f2; padding: 10px; margin-bottom: 10px;">
        <div style="border: 1px solid #ccc; padding: 5px; border-radius: 5px; display: flex; align-items: center; justify-content: space-between; gap: 10px;">
          <div>
            <img alt="Logo de la capa 1" style="width: 20px; height: 20px; border-radius: 50%; border: 1px solid #ccc; object-fit: cover;"/>
            <div>
              <p>Capa 1</p>
              <p>Subcapa 1</p>
            </div>
          </div>
          <div>
            <img alt="Logo de la capa 2" style="width: 20px; height: 20px; border-radius: 50%; border: 1px solid #ccc; object-fit: cover;"/>
            <div>
              <p>Capa 2</p>
              <p>Subcapa 2</p>
            </div>
          </div>
        </div>
      </div>
      <div class="col-12 col-md-6" style="background-color: #f2f2f2; padding: 10px; margin-bottom: 10px;">
        <div style="border: 1px solid #ccc; padding: 5px; border-radius: 5px; display: flex; align-items: center; justify-content: space-between; gap: 10px;">
          <div>
            <img alt="Logo de la capa 3" style="width: 20px; height: 20px; border-radius: 50%; border: 1px solid #ccc; object-fit: cover;"/>
            <div>
              <p>Capa 3</p>
              <p>Subcapa 3</p>
            </div>
          </div>
          <div>
            <img alt="Logo de la capa 4" style="width: 20px; height: 20px; border-radius: 50%; border: 1px solid #ccc; object-fit: cover;"/>
            <div>
              <p>Capa 4</p>
              <p>Subcapa 4</p>
            </div>
          </div>
        </div>
      </div>
    </div>
```

## Resultado.

### Elementos de bloque mostrados como elementos en línea con display inline-block



### Elementos de bloque



Elaboración propia. ([CC BY-SA](#))

## 5.2.- Flotar

Flotar sirve para mover una caja a la izquierda o a la derecha hasta que su borde exterior toque el borde de la caja que lo contiene o toque otra caja flotante.

Las cajas flotantes no se encuentran en el "flujo normal" del documento, por lo que las cajas que sí siguen el flujo normal se comportan como si las flotantes no estuviesen ahí.

### ✓ float<br />

Flotar es algo más que mover una imagen. Sirve para crear diseños multicolumna, barras de navegación de listas no numeradas, poner contenido en forma tabular pero sin emplear tablas y mucho más.



LeRe Pics (CC BY-SA)

La propiedad float puede tener los siguientes valores:

- ➡ none: hará que el objeto no sea flotante.
- ➡ left: hace que el elemento flote izquierda.
- ➡ right: hace que el elemento flote a la derecha.
- ➡ inherit: hará que el elemento tome el valor de esta propiedad de su elemento padre.

### ✓ clear<br />

Sirve para mantener limpia el área que está al lado del elemento flotante y que el siguiente elemento comience en su posición normal dentro del bloque que lo contiene.

La propiedad clear puede tener los siguientes valores:

- ➡ left: indica que el elemento comienza por debajo de cualquier otro elemento del bloque al que pertenece que estuviese flotando a la izquierda.
- ➡ right: funciona como left pero en este caso el elemento deberá estar flotando a la derecha.
- ➡ both: mueve hacia abajo el elemento hasta que esté limpio de elementos flotantes a ambos lados.
- ➡ none: permite elementos flotantes a ambos lados. Es el valor por defecto.
- ➡ inherit: indica, al igual que en float, que heredará el valor de la propiedad clear de su elemento padre.

**Para que un elemento pueda flotar debe tener definido implícita o explícitamente su tamaño.**



## Debes conocer

Es importante saber aplicar las técnicas de flotado en contenedores y cajas en general. En la siguiente presentación veremos con detalle el funcionamiento de las propiedades **float** y **clear**.

[Resumen textual alternativo](#)

A continuación se adjunta código de la presentación.

**Código HTML.**

```
<div>
    <div>Caja 1</div>
    <div>Caja 2</div>
    <div>Caja 3</div>
    <div>Caja 4</div>
    <div>Caja 5</div>
    <div>Caja 6</div>
    <div>Caja 7</div>
    <div>Caja 8</div>
    <div>Caja 9</div>
    <div>Caja 10</div>
    <div>Caja 11</div>
    <div>Caja 12</div>
    <div>Caja 13</div>
    <div>Caja 14</div>
    <div>Caja 15</div>
    <div>Caja 16</div>
    <div>Caja 17</div>
    <div>Caja 18</div>
    <div>Caja 19</div>
    <div>Caja 20</div>
    <div>Caja 21</div>
    <div>Caja 22</div>
    <div>Caja 23</div>
    <div>Caja 24</div>
    <div>Caja 25</div>
    <div>Caja 26</div>
    <div>Caja 27</div>
    <div>Caja 28</div>
    <div>Caja 29</div>
    <div>Caja 30</div>
    <div>Caja 31</div>
    <div>Caja 32</div>
    <div>Caja 33</div>
    <div>Caja 34</div>
    <div>Caja 35</div>
    <div>Caja 36</div>
    <div>Caja 37</div>
    <div>Caja 38</div>
    <div>Caja 39</div>
    <div>Caja 40</div>
    <div>Caja 41</div>
    <div>Caja 42</div>
    <div>Caja 43</div>
    <div>Caja 44</div>
    <div>Caja 45</div>
    <div>Caja 46</div>
    <div>Caja 47</div>
    <div>Caja 48</div>
    <div>Caja 49</div>
    <div>Caja 50</div>
    <div>Caja 51</div>
    <div>Caja 52</div>
    <div>Caja 53</div>
    <div>Caja 54</div>
    <div>Caja 55</div>
    <div>Caja 56</div>
    <div>Caja 57</div>
    <div>Caja 58</div>
    <div>Caja 59</div>
    <div>Caja 60</div>
    <div>Caja 61</div>
    <div>Caja 62</div>
    <div>Caja 63</div>
    <div>Caja 64</div>
    <div>Caja 65</div>
    <div>Caja 66</div>
    <div>Caja 67</div>
    <div>Caja 68</div>
    <div>Caja 69</div>
    <div>Caja 70</div>
    <div>Caja 71</div>
    <div>Caja 72</div>
    <div>Caja 73</div>
    <div>Caja 74</div>
    <div>Caja 75</div>
    <div>Caja 76</div>
    <div>Caja 77</div>
    <div>Caja 78</div>
    <div>Caja 79</div>
    <div>Caja 80</div>
    <div>Caja 81</div>
    <div>Caja 82</div>
    <div>Caja 83</div>
    <div>Caja 84</div>
    <div>Caja 85</div>
    <div>Caja 86</div>
    <div>Caja 87</div>
    <div>Caja 88</div>
    <div>Caja 89</div>
    <div>Caja 90</div>
    <div>Caja 91</div>
    <div>Caja 92</div>
    <div>Caja 93</div>
    <div>Caja 94</div>
    <div>Caja 95</div>
    <div>Caja 96</div>
    <div>Caja 97</div>
    <div>Caja 98</div>
    <div>Caja 99</div>
    <div>Caja 100</div>
```

**Código CSS.**

```
body { font-weight: bold; color: white; }
```

```
  <div>
```

```
    <a href="#" style="color: inherit; text-decoration: none; font-size: 1em; font-weight: bold; border-bottom: 1px solid black; padding-bottom: 2px; margin-bottom: 10px;">
```

```
      Home
```

```
      <br><br><br><br><br><br>
```

```
  <div>
```

```
    <img alt="Logo" style="width: 100px; height: 100px; border-radius: 50%; border: 2px solid black; margin-bottom: 10px;">
```

```
    <div> <img alt="Logo" style="width: 100px; height: 100px; border-radius: 50%; border: 2px solid black; margin-bottom: 10px;">
```

```
      <br> <br> <br> <br> <br> <br>
```

```
    <div>
```

```
  <div>
```

```
    <img alt="Logo" style="width: 100px; height: 100px; border-radius: 50%; border: 2px solid black; margin-bottom: 10px;">
```

```
    <div> <img alt="Logo" style="width: 100px; height: 100px; border-radius: 50%; border: 2px solid black; margin-bottom: 10px;">
```

```
      <br> <br> <br> <br> <br> <br>
```

```
    <div>
```

```
  <div>
```

```
    <img alt="Logo" style="width: 100px; height: 100px; border-radius: 50%; border: 2px solid black; margin-bottom: 10px;">
```

```
    <div> <img alt="Logo" style="width: 100px; height: 100px; border-radius: 50%; border: 2px solid black; margin-bottom: 10px;">
```

```
      <br> <br> <br> <br> <br> <br>
```

```
    <div>
```

```
  <div>
```

```
    <img alt="Logo" style="width: 100px; height: 100px; border-radius: 50%; border: 2px solid black; margin-bottom: 10px;">
```

```
    <div> <img alt="Logo" style="width: 100px; height: 100px; border-radius: 50%; border: 2px solid black; margin-bottom: 10px;">
```

```
      <br> <br> <br> <br> <br> <br>
```

```
    <div>
```

```
  <div>
```

```
    <a href="#" style="color: inherit; text-decoration: none; font-size: 1em; font-weight: bold; border-bottom: 1px solid black; padding-bottom: 2px; margin-bottom: 10px;">
```

```
      Home
```

```
      <br><br><br><br><br><br>
```

```
    <div> <img alt="Logo" style="width: 100px; height: 100px; border-radius: 50%; border: 2px solid black; margin-bottom: 10px;">
```

```
      <br> <br> <br> <br> <br> <br>
```

```
    <div>
```



## Para saber más

A continuación se deja un enlace con otro ejemplo sobre posicionamiento con float. En dicho enlace vemos la explicación de la propiedad float con un ejemplo, cuyo código puede obtenerse.

 [Posicionamiento flotante](#)

## 5.3.- Diseño responsivo. Viewport

---

Como se comentó al inicio de este punto, actualmente cuando diseñamos una web, no solo tenemos que tener en cuenta el diseño para un dispositivo, es decir, la web que diseñemos tiene que tener la capacidad de adaptarse y visualizarse correctamente en el mayor número de dispositivos.

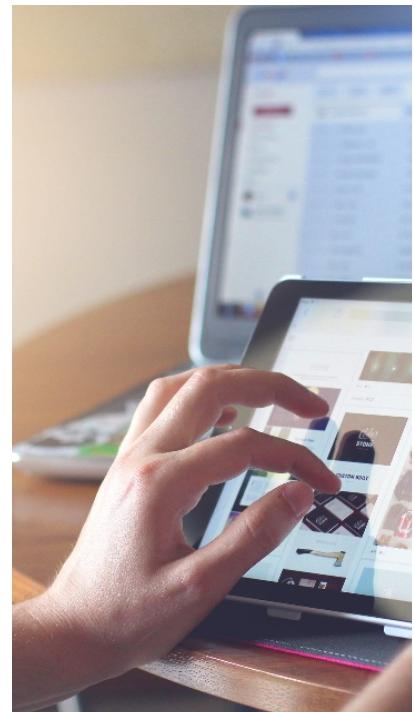
Llegados a este punto y teniendo en cuenta nuestra realidad actual, **cuando diseñemos una web tendremos que realizar un diseño responsive**, es decir, que dependiendo del dispositivo donde visualicemos nuestra web, ésta tendrá que adaptarse a dicho dispositivo y verse correctamente. Para conseguir este propósito los elementos que utilicemos deben ser flexibles de tal forma se adapten a diferentes dimensiones de forma automática. Otra cuestión que necesitaremos es la **posibilidad de modificar o cambiar los elementos de nuestra web dependiendo del dispositivo** en el que se esté visualizando.

Planteadas estas dos necesidades, definir elementos flexibles y cambiar el aspecto de los elementos de nuestra web, vamos a ver cómo satisfacerlas. La segunda necesidad podemos conseguirla como se vio al principio de la unidad de trabajo con las **media queries**. Así podemos detectar las características de un dispositivo y mostrar, ocultar o modificar el comportamiento de nuestros elementos en función del mismo.

Para intentar que los elementos sean flexibles y se adapten a las dimensiones de un dispositivo, se creó hace unos años una etiqueta denominada **viewport**. Hoy en día esta etiqueta es soportada por todos los navegadores más importantes. Dicha etiqueta se define dentro de la etiqueta head de nuestro código HTML y aunque no te hayas dado cuenta la hemos estado utilizando en bastantes ejemplos de la unidad. Ejemplo:



A continuación vamos a analizar los diferentes parámetros que puede contener esta etiqueta.



pigels.com CC0

#### ✓ width=device-width

Con este parámetro indicamos que el ancho de nuestra página va a ser igual que el ancho de la interfaz gráfica del dispositivo. También podría realizarse con la propiedad height (`height=device-height`) pero lo más habitual es realizarlo con la propiedad width. Con este parámetro lo que vamos a conseguir es "encajar" nuestra página web en la resolución que tenga el dispositivo donde se vaya a visualizar.

#### ✓ initial-scale=1.0

Definimos el nivel de zoom con el que se va a mostrar al inicio (al cargarse), nuestra página. En el caso que se muestra no existe zoom.

#### ✓ user-scalable=no

Indicamos que al usuario no le está permitido realizar zoom.

#### ✓ maximun-scalable=1

En caso de poder realizar zoom, indicamos el máximo zoom permitido.

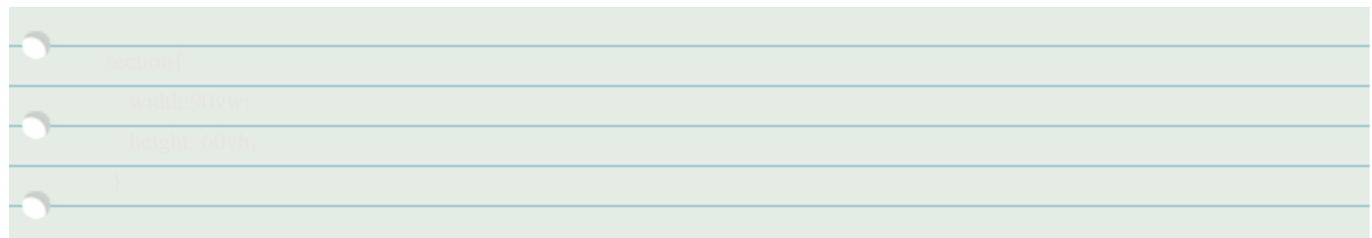
No es necesario definir o indicar todos estos parámetros. Por ejemplo el siguiente código sería válido:



Al usar la etiqueta viewport disponemos de dos unidades de medida que nos van a favorecer nuestro diseño responsive como son:

- ✓ vw: equivaldría al 1% del ancho de la ventana gráfica. El 100% sería todo el ancho.
- ✓ vh: equivaldría al 1% del alto de la ventana gráfica.
- ✓ vmin: equivale al valor mínimo entre la anchura y la altura, si tenemos un dispositivo móvil en posición vertical el valor de vmin sería 1 vw de la ventana gráfica del dispositivo.
- ✓ vmax: equivale al valor máximo entre la anchura y la altura, si tenemos un dispositivo móvil en posición vertical el valor de vmax sería 1 vh de la ventana gráfica del dispositivo.

Así podemos definir las dimensiones de un elemento utilizando estas medidas. Ejemplo:



En este ejemplo estamos definiendo un elemento section con unas dimensiones del 90% de ancho de la interfaz del dispositivo y un alto del 60%.

Anteriormente hablamos de las **media-queries** para realizar un diseño responsive. La pregunta que nos hacemos es, ¿cuándo tendremos que utilizarlas?.

La respuesta viene dada por las dimensiones de los diferentes dispositivos, es decir, dependiendo de la dimensión de nuestro dispositivo tendremos que modificar el comportamiento de nuestra web. Así, cuando aplicamos una media query para cambiar nuestra maqueta lo que estamos haciendo es crear un **punto de ruptura o breakpoint**, por tanto tendríamos que saber cuáles son las principales medidas de nuestros dispositivos. Hoy en día, saber esto es complicado, debido a la gran cantidad de dispositivos existentes en el mercado. Una posible solución es fijarnos en los puntos de ruptura que utilizan  framework que incorporan diseño responsivo, como puede ser el caso de Bootstrap, si accedemos a su documentación podemos comprobar que aplica los siguientes puntos de ruptura: `576px`, `768px`, `992px` y `1200px`.

El siguiente enlace nos lleva a la  [página oficial de Bootstrap](#) donde podemos conocer un poco más sobre la forma en la que este software aplica los breakpoints.

## 5.4.- FlexBox

**Flexbox o modelo de caja flexible**, es una nueva tecnología que facilita mejoras frente a las técnicas de posicionamiento y flotación vistos en los apartados anteriores. Esta técnica como veremos nos permitirá configurar nuestra maqueta de una forma mucho más fácil y precisa, además facilitará el diseño responsivo.

Flexbox se caracterizará por ser unidimensional, es decir podremos configurar los elementos contenidos dentro de un contenedor en una dirección, bien horizontal (por defecto) o verticalmente.

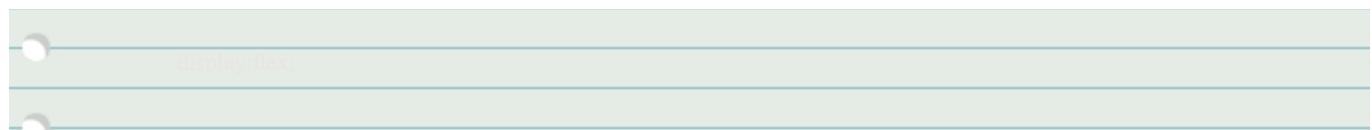
Al utilizar flexbox, los elementos configurados se adaptarán de forma muy sencilla al espacio disponible, intentando ocupar todo el espacio en el que están contenidos. Esto facilita que nuestro desarrollo se adapte de forma sencilla a diferentes dispositivos.

Debemos diferenciar dos tipos de elementos, el elemento padre (contenedor) y los elementos hijos (elementos), la mayoría de las operaciones que se pueden realizar, se harán sobre el objeto contenedor o padre, aunque también existen otras que podremos realizar sobre los hijos. Antes de comenzar a explicar cómo diseñar una maqueta utilizando flexbox, habría que tener claros algunos conceptos, que pasamos a describir.

- ✓ **Eje principal:** eje sobre el que se alinearán los elementos hijos. Así podemos tener un eje horizontal que irá desde la derecha a la izquierda o viceversa.
- ✓ **Eje secundario:** eje perpendicular al eje principal. Si nuestro eje principal era horizontal, éste será el vertical y podrá ir de arriba abajo o viceversa.
- ✓ **Padre:** elemento que contendrá aquellos elementos que vamos a posicionar.
- ✓ **Hijo:** cada uno de los elementos contenidos dentro del padre.

Para poder utilizar flexbox, tendremos que utilizar la propiedad display (definiéndola en el elemento padre) y asignarle uno de los siguientes valores:

- ✓ **flex:** los elementos hijos se comportan como elementos de bloque (definido por block).
- ✓ **inline-flex:** los elementos hijos se comportan como elementos a los que hubiéramos definido como inline-block, visto en apartados anteriores.



Lo primero que tendremos que indicar es cómo se van a posicionar nuestros elementos hijos sobre el eje principal, de forma horizontal o vertical. Para eso utilizaremos la propiedad flex-direction, la cual puede tener los siguientes valores:

- ✓ **row:** posiciona los elementos de forma horizontal.
- ✓ **column:** posiciona los elementos de forma vertical.
- ✓ **row-reverse:** posiciona los elementos en forma horizontal pero en orden inverso.
- ✓ **column-reverse:** posiciona los elementos en forma vertical pero en orden inverso.

```
#contenedor {  
    display: flex;  
    flex-direction: row;  
}
```

Lo siguiente que tendremos que definir es si queremos que todos nuestros elementos estén contenidos en una única línea o que puedan distribuirse en varias líneas (según nuestras necesidades). Para definir esto utilizaremos la propiedad **flex-wrap**, que puede tomar los valores:

- ✓ **nowrap**: los elementos no se desbordan y se adaptan a una única línea (por defecto).
- ✓ **wrap**: los elementos se agrupan en varias líneas, utilizando aquellas que necesite.
- ✓ **wrap-reverse**: igual que la anterior propiedad pero en sentido inverso.

Si tenemos un conjunto de elementos cuyo ancho es superior al contenedor padre y la propiedad **flex-wrap** tiene asignado el valor **nowrap**, el ancho de los elementos se adaptará para que éstos quepan en el objeto padre. Por el contrario, si la propiedad tiene el valor **wrap**, éstos tendrán el ancho indicado y utilizarán varias líneas. Como ya se puede comprobar esta característica puede servirnos para adaptar nuestra web a diferentes dispositivos.

La propieda **flex-flow**, nos permite indicar las dos propiedades anteriores **flex-direction** y **flex-wrap** de forma simultánea.

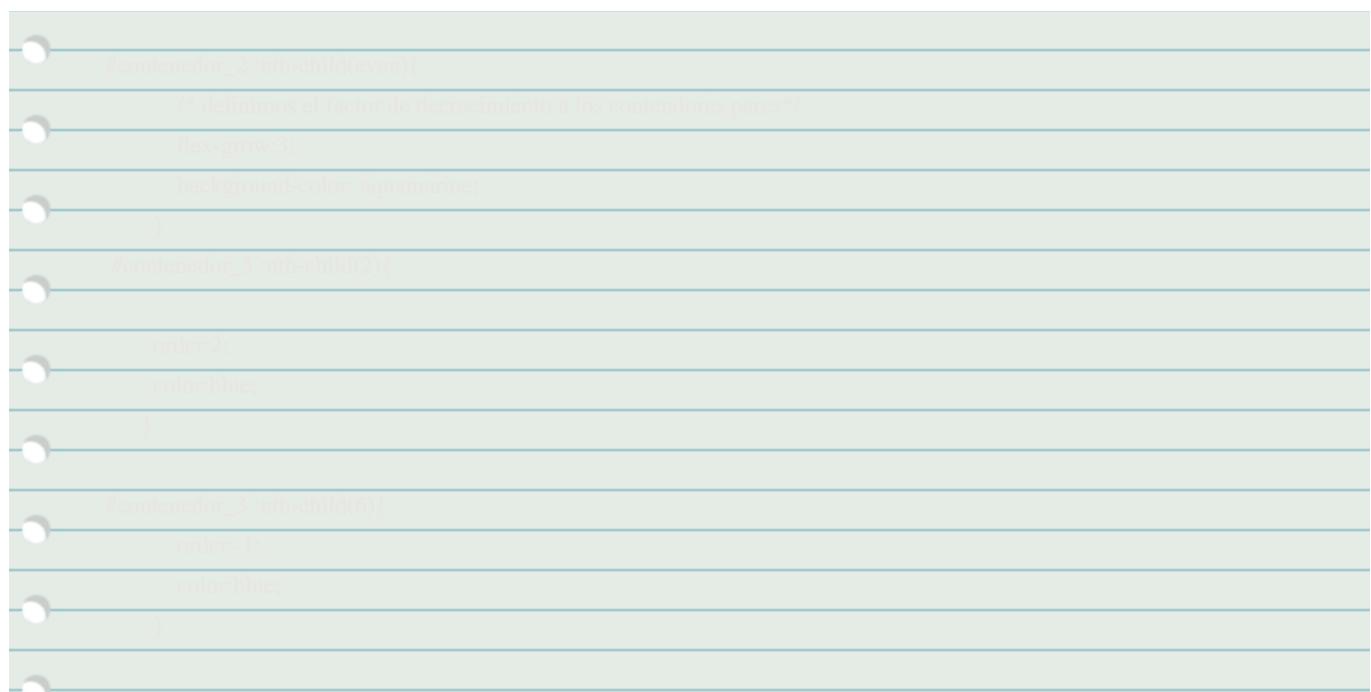
```
#contenedor {  
    border: 1px solid black;  
    display: flex;  
    flex-direction: row;  
    flex-wrap: nowrap;  
}
```

Hasta ahora todas las propiedades que hemos visto actúan sobre el objeto contenedor o padre, pero podemos actuar también sobre los elementos hijos como veremos con las siguientes propiedades. Pero antes de eso tendremos que ver cómo se distribuyen los elementos hijos dentro del padre.

Si tenemos un conjunto de elementos y éstos tienen un tamaño definido, ocuparán dicho espacio, suponiendo que el espacio que ocupan es menor que el espacio del contenedor (tenemos que tener en cuenta el eje principal), tendremos un espacio libre o sobrante, esto es lo que se denomina como **espacio disponible**. Nosotros podemos indicar cómo nuestros elementos queremos que se modifiquen para que ocupen dicho espacio disponible, con las propiedades:

- ✓ flex-basis: nos permitirá definir la longitud inicial de un elemento (dimensión base), así podemos asignarle una unidad de medida o el valor content, con lo que cogerá el tamaño necesario para mostrar su contenido. Partiendo de este tamaño base el objeto podrá crecer o decrecer según le indiquemos. El valor por defecto para esta propiedad es auto.
- ✓ flex-grow: con esta propiedad indicaremos cómo queremos que crezca cada elemento hijo, utilizando ese espacio sobrante. Tendrá un valor numérico.
- ✓ flex-shrink: esta propiedad funcionará al contrario que la anterior, indicando cómo queremos que nuestro elemento hijo decrezca, para ajustarse al espacio del contenedor. Valor numérico.
- ✓ order: nos permitirá configurar el orden en que aparecen los elementos hijos y podrá tomar valores positivos y negativos. Los elementos aparecerán por defecto en el orden indicado en el documento [HTML](#). Al asociar esta propiedad a un elemento, lo que hará será colocar éste al final del conjunto de elementos en el orden indicado, si tiene valor positivo y al principio si tiene valor negativo.

Como se puede ver en el ejemplo final se han modificado los valores positivos de los hijos 4 y 6. De esta forma estos elementos se irán al final y pondremos primero el que tiene el orden más bajo (en nuestro caso el 4) y posteriormente el que tiene el orden 2 (en nuestro caso el 2). Con el valor negativo ocurre igual pero empezando por el principio (en nuestro caso por la izquierda).



La propiedad `flex` permite definir las propiedades `flex-basis`, `flex-grow` y `flex-shrink` de forma simultánea. Habrá que incluir los valores para cada propiedad en este orden.

Para finalizar el apartado veremos cómo los elementos hijos se posicionan dentro del contenedor, es decir, cómo se alinean en los diferentes ejes.

Para alinear los elementos en el eje principal tenemos las propiedades:

- ✓ `justify-content`: en el caso en que solo haya una línea.
- ✓ `align-content`: en el caso que tengamos una estructura multilínea, podremos alinear las filas de forma vertical (tenemos un conjunto de líneas).

Para alinearlos en el eje secundario tendremos:

- ✓ `align-items`: en el caso de que solo haya una línea.

## Código ejemplo Flexbox.

<!DOCTYPE html>

```
<html lang="es">
```

1100-1105, 1978.

www.wright.com

ANSWER

- **contenidos**
  - se crean los datos de acuerdo al modelo jerárquico y se componen de contenidos.
- **bloques**
  - **border-block-type solid**
  - **margin-top**
  - **padding**
- **accesorios**
  - **border-blue-type solid;**
  - **display flex**
  - **flex-direction row**
  - **flex-grow margin**
- **ítems**
- **contenedor 1 (subcontenidos)**
  - se le damos el valor de desplazamiento a los hijos iguales del contenedor 1
  - **flex-grow**
  - **background-color aquamarine**
- **ítems**
- **contenedor 2 (subcontenidos)**
  - se le damos el valor de desplazamiento a los hijos iguales del contenedor 2
  - **flex-grow**
  - **background-color aquamarine**
- **ítems**
- **contenedor 3 (subcontenidos)**
  - se le damos el valor de desplazamiento a los hijos iguales del contenedor 3
  - **flex-grow**
  - **background-color aquamarine**
- **ítems**
- **contenidos 1**
  - **border-green-type solid**
  - **display flex**
  - **flex-direction row**
- **ítems**
- **el contenido 1 que se sitúa al final en la posición 7 tiene todos los elementos que no tienen asignado color.**
- **contenedor 4 (subcontenidos)**
  - **display**
  - **colorblue**
- **ítems**

• Propiedades de los elementos de la lista:

○ Propiedad `list-style-type`: define el tipo de los elementos de la lista. Los valores definidos en la pág. 27.

○ Propiedad `list-style-image`:

○ `list-style-type`:

○ `list-style-image`:

○ `list-style-type`:

• Propiedades para la alineación de los párrafos en la página:

○ Propiedad `text-align`:

○ `text-align`:

○ `text-align`:

○ `text-align`:

○ Propiedad `text-align`:

○ `text-align: center;`

○ `text-align: right;`

○ `text-align: left;`

○ `text-align: justify;`

○ `text-align: center;`

○ Diferentes tipos de justificación para la propiedad `justify-content`:

○ `justify-content: space-around;`

○ `justify-content: space-between;`

○ `justify-content: space-around;`

○ `justify-content: space-between;`

○ `justify-content: flex-end;`

○ `justify-content: flex-end;`

○ Diferentes tipos de alineación para la propiedad `align-items`:

○ `align-items: flex-start;`

○ `align-items: flex-end;`

○ `align-items: center;`

○ `align-items: space-around;`

○ `align-items: space-between;`

○ `align-items: flex-end;`

○ Propiedad `align-items`:

○ `align-items: flex-start;`

○ `align-items: flex-end;`

○ `align-items: center;`

○ `align-items: flex-end;`

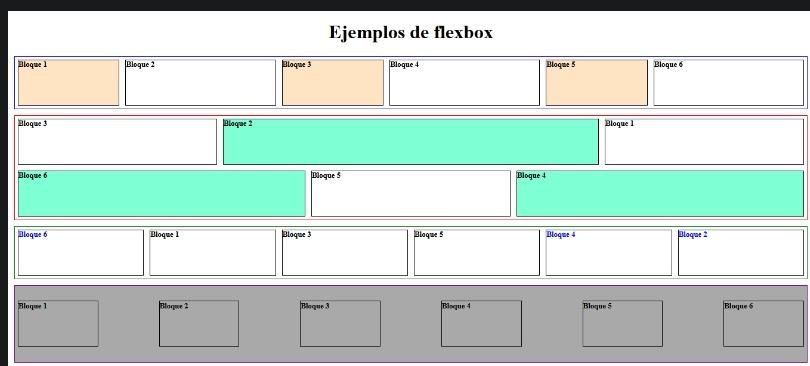
○ `align-items: flex-end;`

○ Ejemplos de flexbox:

○ `<div><div></div></div>`

• `display: flex;`  
• `display: flex; justify-content: space-around;`  
• `display: flex; align-items: center;`  
• `display: flex; align-items: flex-end;`  
• `display: flex; align-items: flex-start;`  
• `display: flex; align-items: space-around;`  
• `display: flex; align-items: space-between;`  
• `display: flex; align-items: center; justify-content: space-around;`  
• `display: flex; align-items: center; justify-content: space-between;`  
• `display: flex; align-items: flex-end; justify-content: space-around;`  
• `display: flex; align-items: flex-end; justify-content: space-between;`  
• `display: flex; align-items: flex-start; justify-content: space-around;`  
• `display: flex; align-items: flex-start; justify-content: space-between;`  
• `display: flex; align-items: center; justify-content: space-around; align-self: flex-end;`  
• `display: flex; align-items: center; justify-content: space-between; align-self: flex-end;`  
• `display: flex; align-items: flex-end; justify-content: space-around; align-self: flex-end;`  
• `display: flex; align-items: flex-end; justify-content: space-between; align-self: flex-end;`  
• `display: flex; align-items: flex-start; justify-content: space-around; align-self: flex-end;`  
• `display: flex; align-items: flex-start; justify-content: space-between; align-self: flex-end;`

## Resultado del código.



Elaboración propia. ([CC BY-SA](#))



## Para saber más

Como las propiedades y valores que hemos descrito son muy numerosas dejamos un enlace donde se puede repasar lo estudiado con más ejemplos:

[Ejemplos flexbox](#)

Para completar la explicación se deja un enlace donde se puede ver de forma visual un resumen de todas las propiedades y valores vistos:

[Resumen visual flexbox.](#)

Y para terminar y practicar de forma interactiva se deja un enlace a un generador de código en línea:

[Generador flexbox](#)

## Autoevaluación

**Dado el siguiente código:**

### Código ejercicio

```
<div>
    <div>Este es el primer elemento</div>
    <div>Este es el segundo elemento</div>
    <div>Este es el tercero elemento</div>
    <div>Este es el cuarto elemento</div>
</div>
```

¿Cuál es el resultado? ¿Por qué?

Algunas preguntas que te podrían hacer tus compañeros:

- ¿Qué significa el valor "flex-grow: 1;"?
- ¿Qué significa el valor "flex-shrink: 1;"?
- ¿Qué significa el valor "flex-basis: 100px;"?
- ¿Qué significa el valor "flex: 1 1 100px;"?
- ¿Qué significa el valor "flex: 1 1 auto;"?
- ¿Qué significa el valor "flex: 1 1 100px; align-items: center;"?
- ¿Qué significa el valor "flex: 1 1 100px; justify-content: space-around;"?
- ¿Qué significa el valor "flex: 1 1 100px; justify-content: space-between;"?
- ¿Qué significa el valor "flex: 1 1 100px; align-items: flex-end;"?
- ¿Qué significa el valor "flex: 1 1 100px; align-items: flex-start;"?
- ¿Qué significa el valor "flex: 1 1 100px; align-items: center; justify-content: space-around;"?
- ¿Qué significa el valor "flex: 1 1 100px; align-items: center; justify-content: space-between;"?
- ¿Qué significa el valor "flex: 1 1 100px; align-items: flex-end; justify-content: space-around;"?
- ¿Qué significa el valor "flex: 1 1 100px; align-items: flex-end; justify-content: space-between;"?
- ¿Qué significa el valor "flex: 1 1 100px; align-items: flex-start; justify-content: space-around;"?
- ¿Qué significa el valor "flex: 1 1 100px; align-items: flex-start; justify-content: space-between;"?

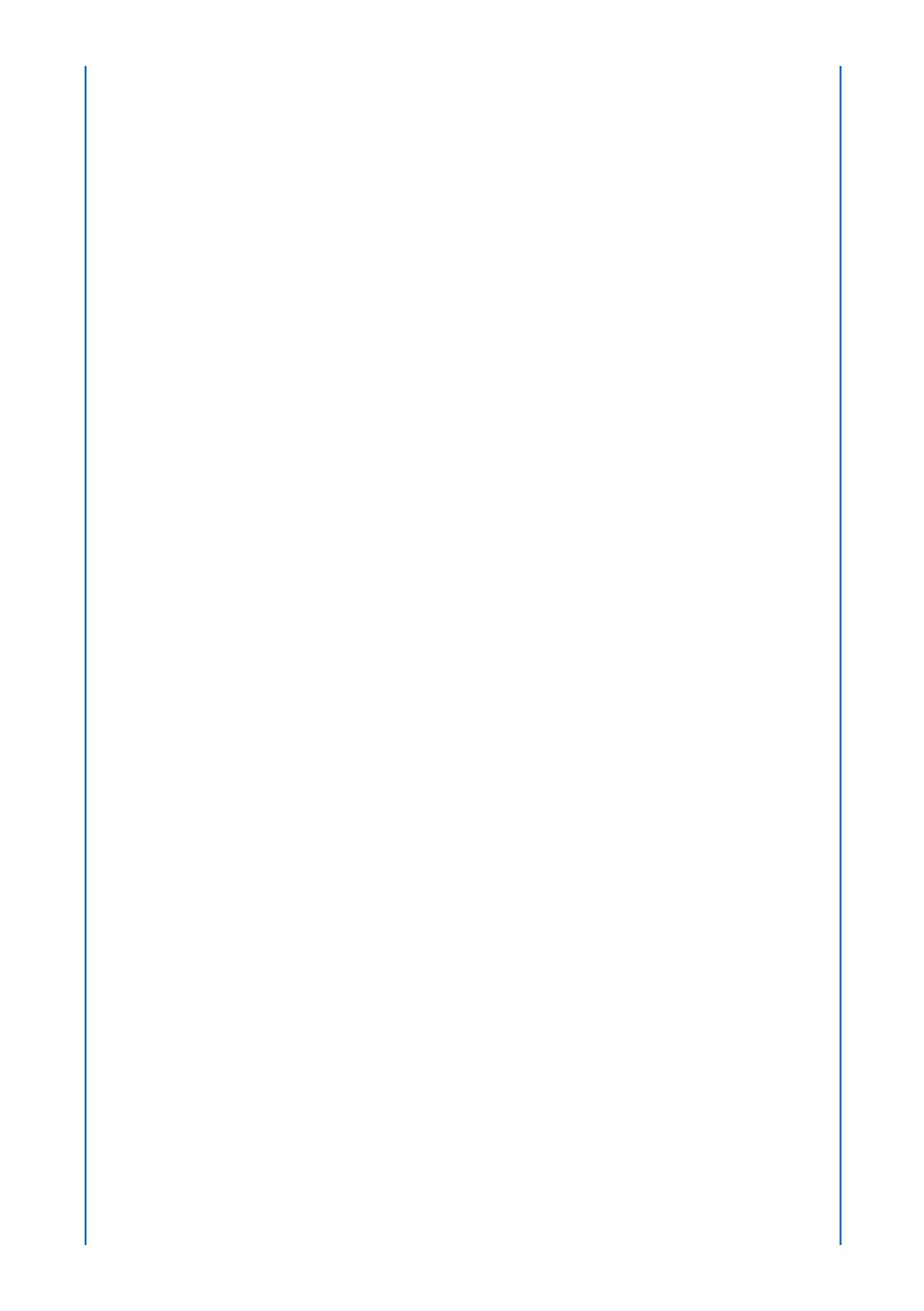
Digitized by srujanika@gmail.com

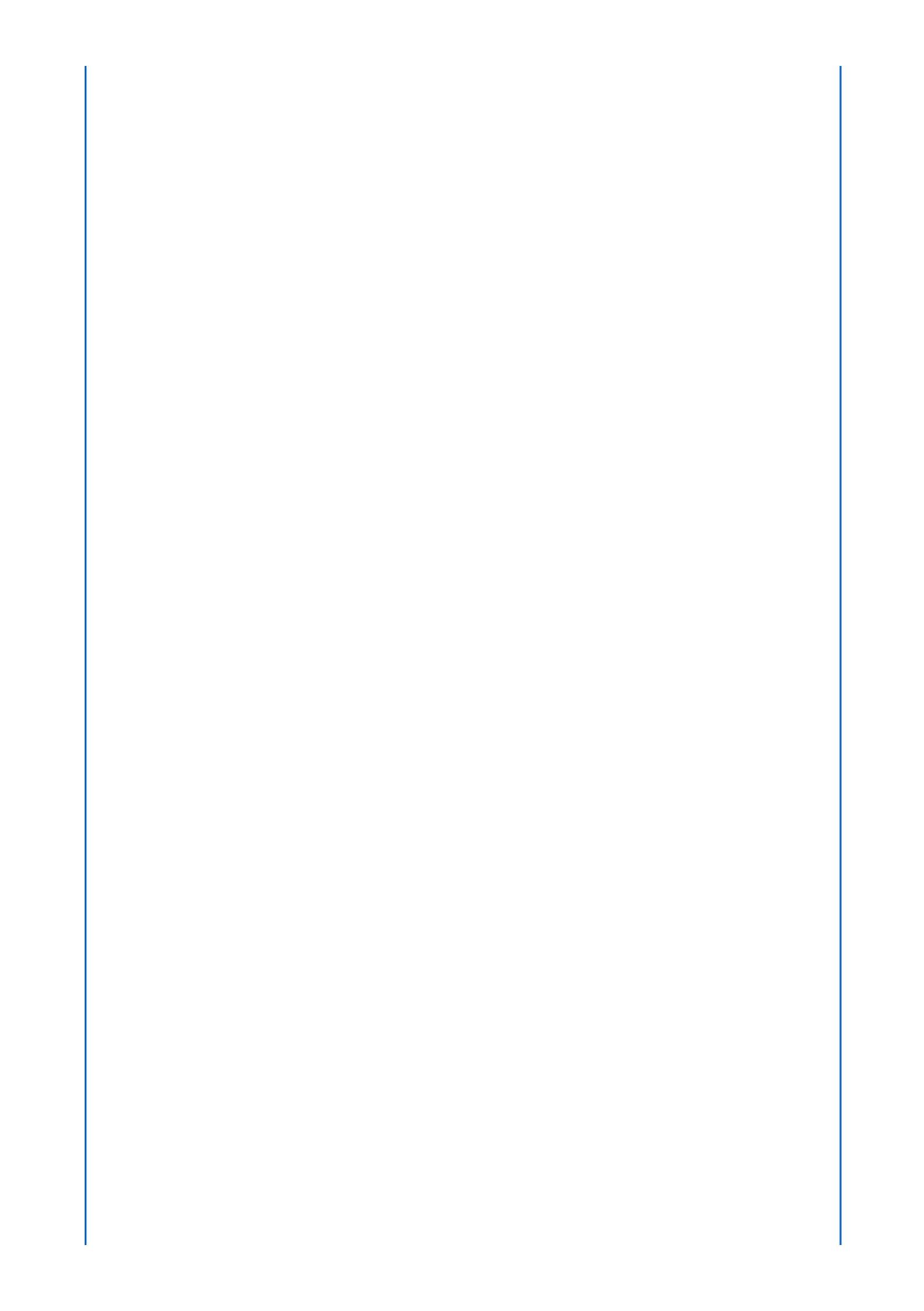
Digitized by srujanika@gmail.com

7

Digitized by srujanika@gmail.com

Digitized by srujanika@gmail.com





**¿Qué ocurrirá con la barra lateral (etiqueta aside) cuando la resolución de la interfaz de visualización sea de 800px?**

- Dicho bloque desaparecerá.
- Cambiará de color.
- Cambiará de color y se posicionará debajo de la etiqueta main.
- Seguirá igual.

## 5.5.- Grid

El nuevo sistema que tenemos para realizar una maquetación de nuestra web se denomina **grid**. Esta tecnología continúa el camino comenzado por **flexbox**, permitiendo realizar diseños responsivos de forma fácil. El sistema grid está estandarizado a día de hoy y por tanto soportado por casi la mayoría de los navegadores. A grandes rasgos, lo que nos va a permitir esta nueva tecnología es el poder definir una matriz, dentro de la que tendremos un conjunto de elementos o celdas, que podrán agruparse en zonas según nuestras necesidades.

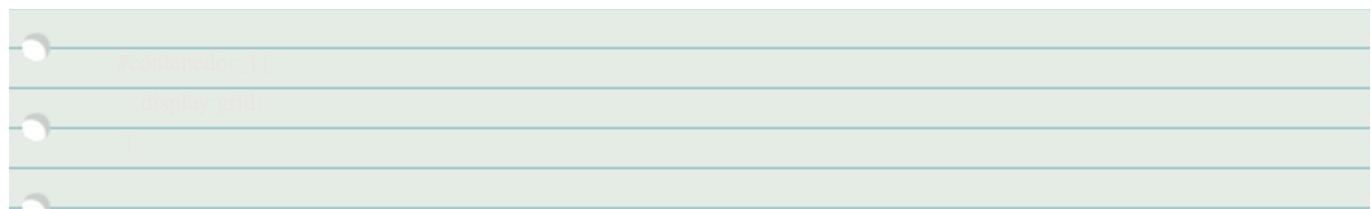
Grid es flexible, al igual que flexbox, entonces podríamos hacernos la siguiente pregunta: **¿qué diferencia hay entre grid y flexbox?**

La principal diferencia es que **flexbox es un sistema unidimensional**, mientras que **grid es bidimensional**.

Comenzaremos a explicar esta nueva tecnología, para ello primero tendremos que definir un conjunto de términos básicos, al igual que ocurría con técnicas anteriores.

- ✓ **Contenedor o padre:** será el elemento que contendrá los elementos que queremos posicionar.
- ✓ **Elementos, hijos o ítems:** cada uno de los elementos contenidos en el contenedor.
- ✓ **Celda:** unidad mínima que tendrá una rejilla.
- ✓ **Área:** será un conjunto de celdas, separadas por líneas.
- ✓ **Separación (gap):** nos referiremos al espacio que separara cada una de las celdas. Esta separación es horizontal y vertical.
- ✓ **Pista (track):** conjunto de celdas en sentido horizontal o vertical, puede asociarse con el concepto de fila o columna. Conjunto de celdas consecutivas en una orientación o sentido.
- ✓ **Punto de inicio:** será el punto en el que se inicie una celda (horizontal o verticalmente). Corresponde con una línea.
- ✓ **Punto de fin:** será el punto en el que finalice una celda (horizontal o verticalmente). Corresponde con una línea.

Para indicar que vamos trabajar con este sistema lo haremos como siempre con la propiedad `display`, así permitirá los valores `grid` e `inline-grid`.



Posteriormente tendremos que definir el número de filas y columnas que vamos a tener en nuestra estructura, y para ello utilizaremos las propiedades:

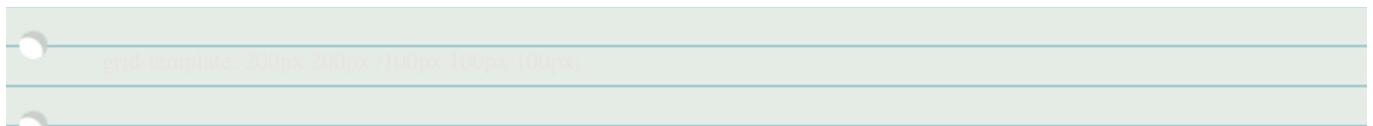
- ✓ grid-template-columns: indicaremos el número de columnas.
- ✓ grid-template-rows: indicaremos el número de filas.
- ✓ grid-template: aglutina las dos propiedades anteriores. Primero se indicarán las filas que estarán separadas de las columnas por el carácter / y cada una de las filas (o columnas) están separadas entre ellas por el espacio en blanco.

Vamos a ver algunos ejemplos con estas propiedades:



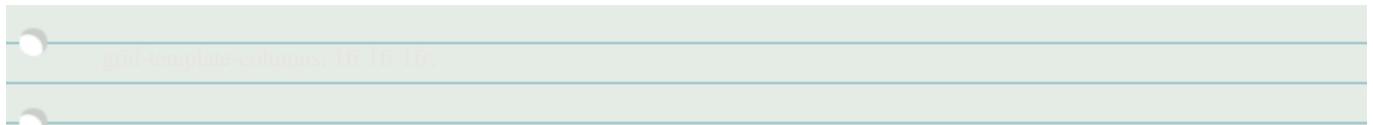
Se definirán 3 columnas con dimensiones de 300 píxeles, 200 píxeles y 100 píxeles.

Se puede utilizar cualquier unidad de medida, px, porcentajes, etcétera.

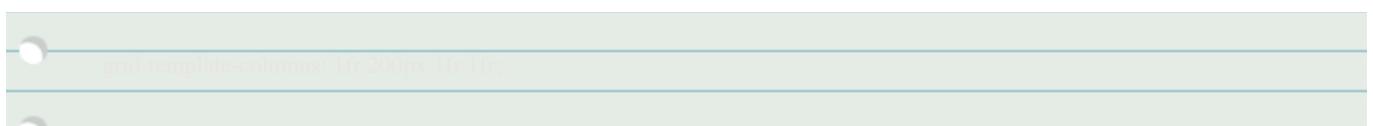


Se definen las filas y las columnas conjuntamente, hemos definido 2 filas de 200px cada una y 3 columnas de 100px cada una.

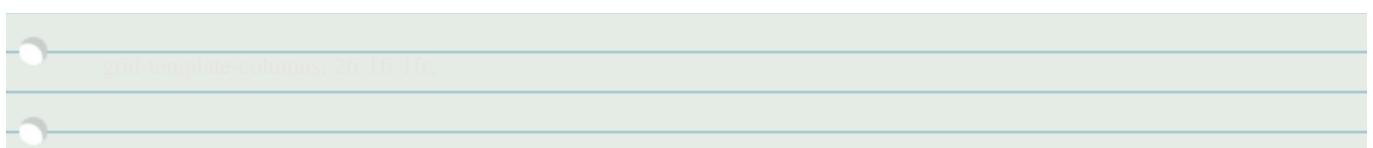
Como ha podido comprobarse, para cada fila y columna le hemos indicado una medida. Si queremos que todas las columnas o filas sean iguales tendremos que poner el valor auto. También podemos utilizar la medida fr, que está relacionado con el espacio sobrante, es decir, la unidad de medida fr, lo que va a realizar es repartir el espacio sobrante de nuestro contenedor. Así, 1 fr será una parte del espacio disponible en el contenedor de la cuadrícula. Ejemplos:



Tendremos tres columnas con igual tamaño. Al no definir tamaño de las columnas, se coge todo el espacio disponible y se divide en tres partes. Esta medida se puede combinar con otras unidades de medida.

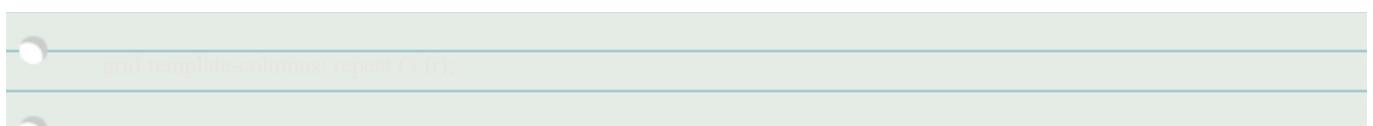


Definimos 4 columnas, una de 200px y el resto del espacio se repartirá en tres columnas iguales.



Definimos 3 columnas siendo la primera el doble de ancha que la segunda y tercera.

En caso de que queramos definir varias filas o columnas con el mismo tamaño podremos utilizar la expresión o función repeat() donde indicaremos el número de veces a repetir y el valor o tamaño de la columna. Ejemplo:



Hemos definido tres columnas con un ancho cada una de 1 fr.

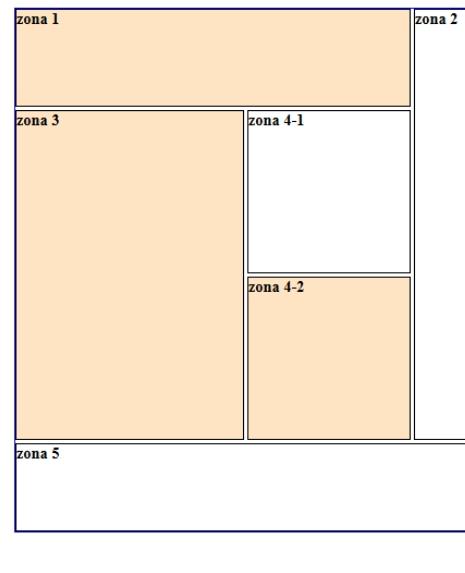
Los ejemplos anteriores están referidos con la propiedad `grid-template-columns`, pero puede aplicarse también a `grid-template-rows`.

Lo siguiente que trataremos es como definir la separación entre cada una de las filas y columnas, para realizar esto utilizaremos las propiedades:

- ✓ `grid-column-gap`: separación a nivel de columna.
- ✓ `grid-row-gap`: separación a nivel de fila.
- ✓ `grid-gap`: separación de fila y columna.

Una vez que tenemos definida nuestra cuadrícula tendremos que posicionar los elementos que deseamos, para ello dispondremos de las propiedades: `grid-row-start`, `grid-row-end`, `grid-column-start` y `grid-column-end` donde definimos los puntos de inicio y fin de la fila (de igual forma actuaríamos con las columnas).

A continuación mostramos la maqueta sobre la que vamos a trabajar y que mostraremos como ejemplo al final del apartado.

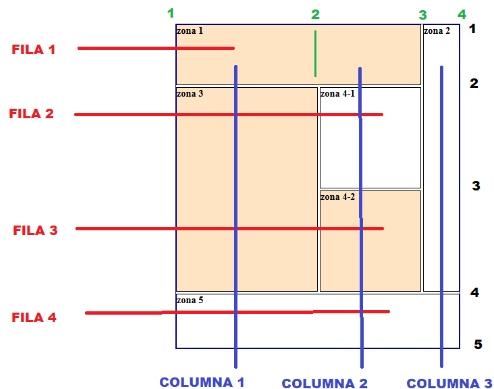


Elaboración propia. [CC0](#)

Para explicar estos atributos nos serviremos del siguiente dibujo, teniendo como referencia la maqueta que vamos a implementar al final.

En el dibujo se marcan las filas con una línea de color rojo, tendremos 4 filas. Las columnas con una línea de color azul, tendremos 3 columnas y los puntos inicio y fin de cada uno de los diferentes componentes. En negro, tendremos los puntos de inicio y fin de cada uno de los elementos a nivel de fila (tenemos 5) y en verde tendremos los puntos de inicio y fin de cada uno de los componentes a nivel de columna (tenemos 4).

Así para definir la zona 1, tendremos que definir el punto de inicio y fin a nivel de fila y columna, en nuestro caso:



Elaboración propia. [CCO](#)

- ✓ Para la fila nuestro elemento comenzaría en el punto 1 y terminaría en el punto 2 (color negro).
- ✓ Para la columna nuestro elemento comenzaría en el punto 1 y terminaría en el punto 3 (color verde).

Hemos tenido como referencia las líneas de nuestra estructura.

Para expresar esto tenemos las propiedades indicadas anteriormente. A continuación vemos cómo quedaría nuestra definición:

```

grid-area: 1/2;
grid-column: start 1;
grid-column: end 2;
grid-row: start 1;
grid-row: end 2;

```

También podríamos utilizar las propiedades `<code>grid-row</code>` y `<code><code><code>grid-column sin indicar los puntos de inicio y fin; esto no siempre es posible, pero en el caso en que nos encontramos, estamos solo centrándonos en la fila 1, por lo que podríamos definir nuestro elemento también de la siguiente manera.`

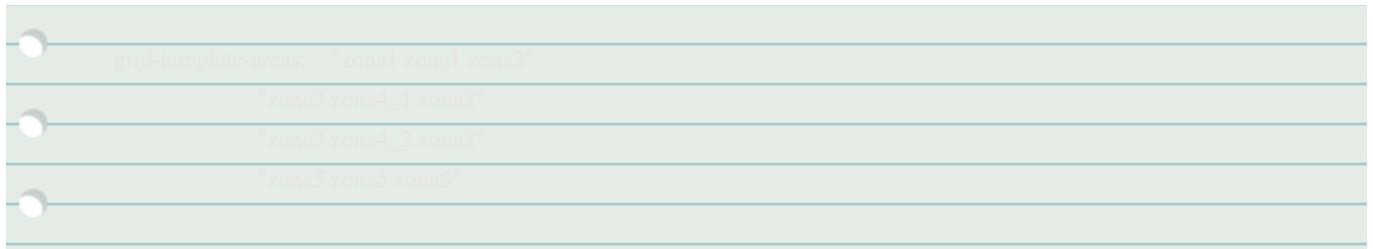
```

grid-area: 1/3;
grid-column: start 1;
grid-column: end 3;
grid-row: start 1;
grid-row: end 1;

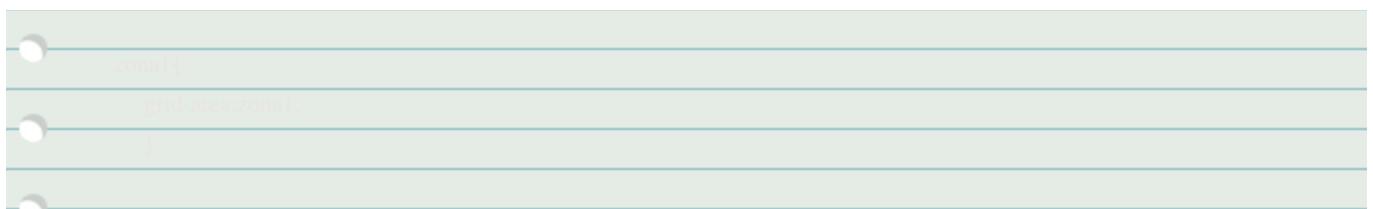
```

Si analizamos este código nos centramos en la fila 1 y cogemos solo la parte de la columna que está delimitada entre los puntos 1 y 3 (puntos de color verde).

Para terminar aún podemos definir nuestra maqueta definiendo áreas (conjunto de celdas), así podemos dar nombre a cada una de las zonas de nuestra maqueta y asignarle un conjunto de celdas formando un área. Para ello tendremos que utilizar la propiedad `<code><code>grid-areas`, donde vamos a definir las celdas en las que se divide nuestro contenedor.



Realizado esto tendremos que asignar a cada elemento un área; esto lo realizaremos con la propiedad `<code><code>grid-areas`. Para el caso que hemos definido anteriormente tendríamos:



Cuando estamos definiendo las celdas en el contenedor, podemos utilizar el símbolo punto (.) que nos permitirá no asignar esa celda, es decir se interpreta como una zona no definida.

Al igual que hemos visto con flexbox podemos alinear los elementos contenidos dentro de la cuadrícula, para ello tenemos la propiedad `justify-content`.

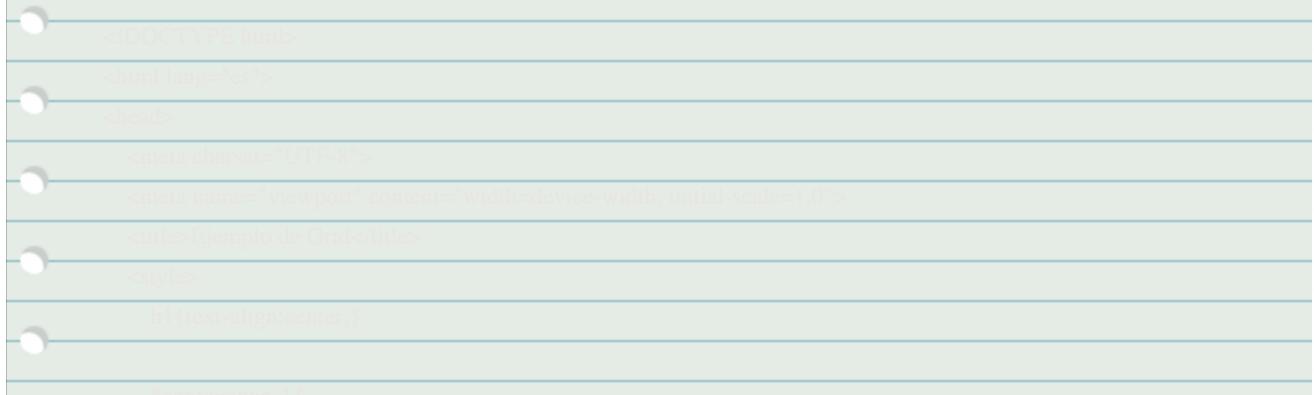
Para que esta propiedad tenga efecto la suma de las dimensiones horizontales de los elementos debe ser menor que el ancho del contenedor. ¿Por qué esto es así?

Porque si no, los elementos se adaptan al tamaño del contenedor, ocupándolo completamente y no pudiendo realizar ninguna operación de centrado, desplazamiento a uno u otro lado, etc. Los valores que puede tener son: `center`, `start`, `end`, `space-around`, `space-between` y `space-evenly`.

Con la propiedad `align-content` podemos alinear de forma vertical. Tendremos los mismos valores y el mismo funcionamiento que lo indicado para la propiedad `justify-content`.

Para finalizar dejamos el código del ejemplo realizado, mostramos dos códigos: uno trabajando con áreas y otro sin ellas con la propiedades `grid-row` y `grid-column`.

## Maqueta con grid (filas y columnas)



margin-top: auto;

width: 100px;

height: 100px;

background-color:

background-size:

100%

background-position:

border: 1px solid;

display: flex;

justify-content: space-around;

flex-wrap: wrap;

grid-template-columns: 1fr 1fr 1fr 1fr;

grid-template-rows: 1fr 1fr 1fr 1fr;

grid-template-areas: "c1 c2 c3 c4";

grid-column-gap: 10px;

grid-row-gap: 10px;

grid-template-columns: 1fr 1fr 1fr 1fr;

grid-template-rows: 1fr 1fr 1fr 1fr;

grid-template-areas: "c1 c2 c3 c4";

grid-column-gap: 10px;

grid-row-gap: 10px;

grid-template-columns: 1fr 1fr 1fr 1fr;

grid-template-rows: 1fr 1fr 1fr 1fr;

grid-template-areas: "c1 c2 c3 c4";

grid-column-gap: 10px;

grid-area: 1;

grid-area: 2;

grid-area: 3;

grid-area: 4;

grid-area: 1;

grid-area: 2;

grid-area: 3;

grid-area: 4;

grid-area: 1;

grid-area: 2;

grid-area: 3;

grid-area: 4;

grid-area: 1;

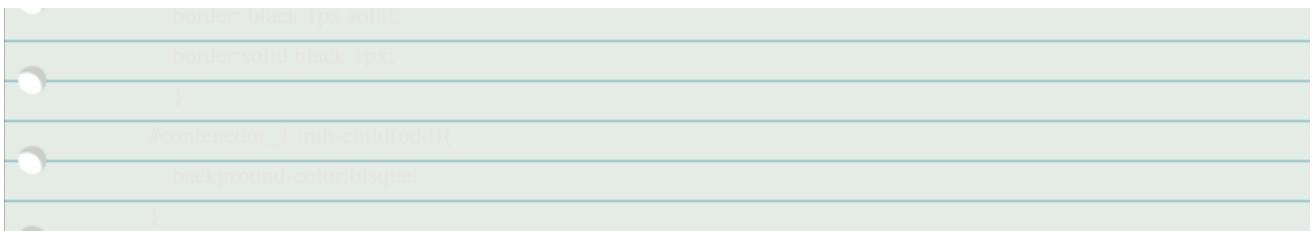
grid-area: 2;

• `background-color`  
• `color`  
• `font-family`  
• `font-size`  
• `font-weight`  
• `text-align`  
• `background-color` / `color`  
• `border-radius` (px solid)  
• `border-radius` (px solid)  
• `border`  
• `background-color` / `color`  
• `background-color` / `color` / `border-radius`  
• `background-color` / `color` / `border-radius`

## Maqueta con grid (áreas)

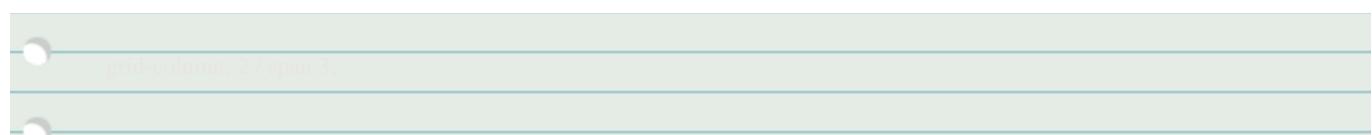
• `display: flex`  
• `flex-wrap: wrap`  
• `grid`  
• `grid-template-columns`  
• `grid-template-rows`  
• `grid-template-areas`  
• `grid-area: "header"`  
• `grid-area: "content"`  
• `grid-area: "main-content"`  
• `grid-area: "side-content"`  
• `grid-area: "left"`  
• `grid-area: "right"`

• *Antennaealata*  
• *Argyresthia* spp.  
• *Autostichus*  
• *Baccharis* spp.  
  
• *Carex* spp.  
• *Carex* spp.  
•  
  
• *Zosterops* spp.  
• *Coracina* spp. (including)  
• *Distincta*  
• *Pholidornis* spp. (including)  
• *Grallirostris* spp. (including)  
• *Myiotheretes* spp. (including)  
• *Oreolais* spp. (including)  
• *Prodotiscus*  
• *Sclateriana*  
• *Sericornis* spp. (including)  
• *Sericornis* spp. (including)  
• *Trochocercus*  
• *Zosterops* spp.  
• *Zosterops* spp.  
  
• *Zosterops*  
• *Zosterops* spp. (including)  
  
•  
  
• *Zosterops*  
• *Zosterops* spp.  
•  
• *Zosterops*  
• *Zosterops* spp. (including)  
• *Zosterops*  
• *Zosterops* spp. (including)  
•  
• *Zosterops*  
• *Zosterops* spp.  
•  
• *Zosterops*  
• *Zosterops* spp.



La propiedad `grid-auto-flow` nos permitirá indicar cómo se posicionan los elementos de forma automática. Puede tomar los valores `column` y `row`. En caso de que nuestra estructura tenga huecos, éstos se pueden llenar de forma automática utilizando la palabra reservada `dense`.

Si queremos que un elemento se expanda por más de una fila o columna podemos utilizar la palabra reservada `span`. Así podríamos hacer que un elemento ocupe más espacio del inicialmente asignado.



Son muchas más las propiedades que podemos encontrar relacionadas con ésta tecnología, pero quedan fuera de nuestra unidad.

Para finalizar indicar que dentro de una estructura definida con grid, pueden crearse otras estructuras utilizando algunas de las tecnologías vistas anteriormente, como por ejemplo flexbox.



## Para saber más

Como las propiedades y valores que hemos descrito son muy numerosas dejamos un enlace donde se puede repasar los estudiado con más ejemplos:



Ejemplos grid

Para completar la explicación se deja un enlace donde se puede ver de forma visual un resumen de todas las propiedades y valores vistos:



Resumen visual grid. (pdf - 499 KB)

Y para terminar y practicar de forma interactiva se deja un enlace a un generador de código en línea:



Generador grid.

## Autoevaluación

**Si quisiéramos que el elemento zona 3 de nuestro ejemplo ocupara también la zona 4\_1 y la zona 4\_2, ¿qué reglas de las siguientes tendríamos que aplicar a nuestra clase .zona3 manteniendo las mismas filas y columnas definidas en el ejemplo?**

- .zona3{ grid-row-start: 2; grid-row-end: 4; grid-column: 1/3; }
- .zona3{ grid-row-start: 2; grid-row-end: 4; grid-column-start: 1; grid-column-end: 3; }
- .zona3{ grid-row-start: 2; grid-row-end: 4; grid-column-start: 2; grid-column-end: 3; }
- .zona3{ grid-row: 2; grid-column: 2;

[Mostrar retroalimentación](#)

## 6.- Cambiar la apariencia de otros elementos web



### Caso práctico

Como casi todas las web, la web de "**Migas Amigas**" tendrá imágenes y texto y otros elementos web como listas, tablas y formularios.

Para este tipo de elementos HTML, el estándar CSS ofrece propiedades y técnicas específicas que permiten darles la apariencia más adecuada en cada momento.

Todo esto obliga a que **Carlos** posea los conocimientos necesarios para configurar la apariencia de todos estos elementos y en especial las listas, pues una parte importante de la web de "**Migas Amigas**" son los menús de navegación, un elemento al que han dedicado muchas hora durante la planificación del diseño.

**Carlos** se pone de nuevo en contacto con **Ada** para que ésta le guíe en el aprendizaje de las propiedades y técnicas necesarias.



Elaboración propia (Uso Educativo no comercial).

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

**Con la aparición de CSS, que permite la separación de la información de presentación del contenido, y con el modelo de cajas, que permite colocar cada elemento en el lugar deseado, te podrías preguntar: ¿entonces por qué seguimos con las tablas?**

**Realmente con CSS, las tablas ya no son necesarias, al menos en lo que a la maquetación de los elementos principales se refiere. No son necesarias para indicar dónde queremos el encabezado o el pie del documento y tampoco son necesarias para hacer un diseño de dos o tres columnas. Pero, cuando se trata de colocar información de contenido en forma tabular, siguen siendo muy útiles.**

***La W3C no recomienda es uso de tablas como elemento de maquetación.***

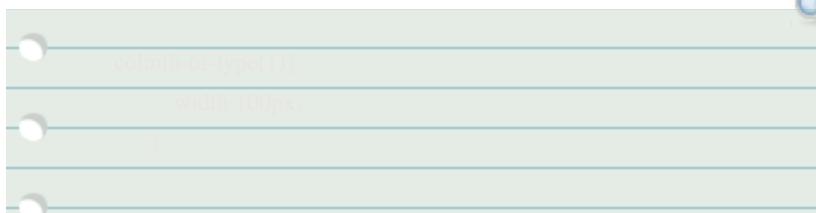


## 6.1- Las tablas con CSS

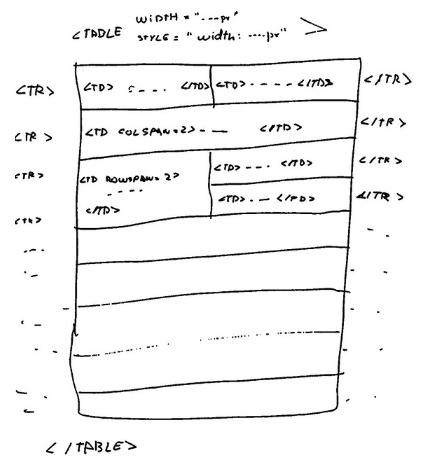
Para controlar la presentación de las tablas tenemos las propiedades: `caption-side`, `table-layout`, `border-collapse`, `border-spacing`, `empty-cells` y `display`, además de otras propiedades más que hemos visto a lo largo de la unidad. Algunas de estas propiedades tendrán la misma finalidad pero enfocada al elemento con el que estamos trabajando tablas, filas o celdas.

### ✓ `width`

Permite definir el ancho de la tabla y de una celda. El valor podemos darlo en cualquiera de las unidades de medida vistas en la unidad, por ejemplo, en términos porcentuales o con medidas absolutas (píxeles). Ejemplo:



[teleniek0 CC BY](#)

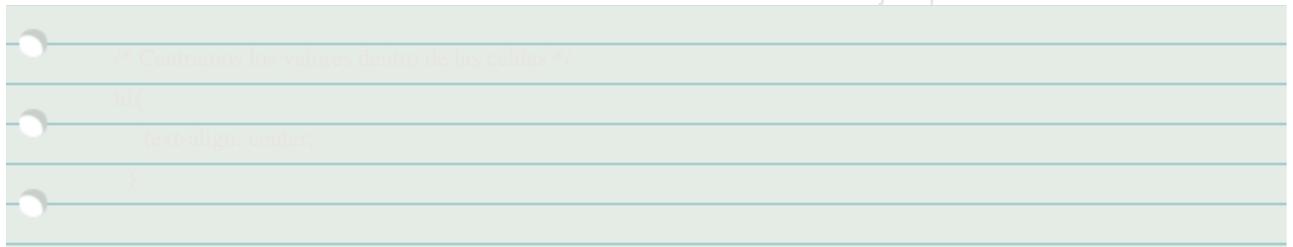


### ✓ `height`

Funciona igual que la propiedad anterior pero con el alto del elemento.

### ✓ `text-align`

Permite alinear un texto horizontalmente dentro de una celda. Ejemplo:



### ✓ `vertical-align`

Permite alinear un texto verticalmente dentro de una celda. El funcionamiento de esta propiedad y la anterior queda descrito en el apartado relacionado con las propiedades de texto.

## ✓ border

Permite definir el borde de nuestra tabla o celda, el funcionamiento queda descrito en el apartado relacionado con bordes.

border-bottom-color: black;
border-bottom-style: solid;
border-bottom-width: 3px;
border-left-color: black;
border-left-style: solid;

## ✓ 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 recomendación recoge la posibilidad de desplazar el contenido de la etiqueta `caption` a la izquierda o a la derecha con `text-align`, pero siempre manteniéndose por encima o por debajo de la tabla.

## ✓ empty-cells

Esta propiedad soluciona la carencia del HTML 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.

## ✓ 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.

## ✓ border-spacing

Permite establecer la separación entre celdas contiguas. Para hacerlos 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. Si el valor de la propiedad `border-collapse` tiene el valor `separate`.

## ✓ 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 tablas basándose en el contenido de sus celdas. Es el valor por defecto.

Ejemplos:

width: 50%;	Determina el ancho y el alto de la tabla dentro del objeto contenedor.
height: 300px;	
border-collapse: collapse;	Indica que no habrá separación entre las celdas.
border-spacing: 0px;	Simplifica el trazo en la parte superior.
border: 1px solid black;	Define los bordes y los cuadros de las celdas vacías.
margin: 10px auto;	Centra los cuadros horizontales de la tabla dentro del contenedor.
border-spacing: 2em 3em;	Separación de los bordes en mielvo caso no tiene efecto ya que la propiedad border-collapse: collapse;.

También se puede utilizar otras propiedades vistas en apartados anteriores como: padding, background y overflow.



## Ejercicio resuelto

En el siguiente enlace podrás descargar el archivo que contiene el código fuente del que tendrás que partir para realizar el ejercicio propuesto. Para que no se abra y se muestre en el navegador, ponte sobre el enlace, pulsa el botón derecho y elige en el menú contextual "**Guardar enlace como...**".

[Código fuente \(0.01 MB\)](#)

Una vez descargado el código HTML puedes abrirlo con tu navegador. Verás que el resultado es el que se muestra en la imagen siguiente:

El navegador muestra una tabla titulada "Distancias planetarias" con los siguientes datos:

Planeta	Distancia desde la Tierra	Web
Venus	108 Millones de Kilómetros	<a href="#">Web Venus</a>
Saturno	1.281 Millones de Kilómetros	<a href="#">Web Saturno</a>
Urano	2.720 Millones de Kilómetros	<a href="#">Web Urano</a>
Neptuno	4.504 Millones de Kilómetros	<a href="#">Web Neptuno</a>

Astronomía Gobierno de España



Elaboración propia. ([CCO](#))

Tendrás que editar el código y determinar las reglas CSS necesarias para que el resultado sea lo más parecido posible al de la siguiente imagen (Nota: Aunque no sale en la imagen, el puntero está sobre la primera fila, por eso se ve de color amarillo):

El navegador muestra una tabla titulada "DISTANCIAS PLANETARIAS" con los siguientes datos:

Planeta	Distancia desde la Tierra	Web
Venus	108 Millones de Kilómetros	<a href="#">Web Venus</a>
Saturno	1.281 Millones de Kilómetros	<a href="#">Web Saturno</a>
Urano	2.720 Millones de Kilómetros	<a href="#">Web Urano</a>
Neptuno	4.504 Millones de Kilómetros	<a href="#">Web Neptuno</a>

Astronomía Gobierno de España



Elaboración propia. ([CCO](#))

[Mostrar retroalimentación](#)

Partiendo del ejemplo de tabla anterior, pasamos a detenernos de forma más detallada en aquellas etiquetas que nos van a permitir agrupar la información de nuestra tabla, ya que nos permitirán aplicar estilos de una forma más fácil y eficiente:

- ✓ `thead`: nos permitirá agrupar los datos de la cabecera.
- ✓ `tbody`: nos permite agrupar los datos del cuerpo de la tabla.
- ✓ `tfoot`: nos permite agrupar los datos del pie de la tabla.
- ✓ `colgroup`: nos permitirá agrupar un conjunto de columnas.

Dejamos un nuevo ejemplo donde se pueden ver diferentes formas de aplicar estilo a los elementos que conforman una tabla, con algunos de los selectores vistos en la unidad de trabajo.

## Código ejemplo.

```
<table border="1">
    <thead>
        <tr>
            <th>Nombre del producto</th>
            <th>Precio unitario</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td>Televisor de 50 pulgadas</td>
            <td>1000</td>
        </tr>
        <tr>
            <td>Smartphone</td>
            <td>500</td>
        </tr>
        <tr>
            <td>Tablet</td>
            <td>300</td>
        </tr>
    </tbody>
</table>
```

```
.table {
    width: 100%; /* Establecemos el ancho y el alto de la tabla dentro del objeto contenedor */
    border-collapse: collapse; /* Indicando que no habrá separación entre las celdas */
    border: 1px solid black; /* Estableciendo el resto de la parte superior */
    border-spacing: 0; /* Separación horizontal entre las celdas y los bordes de las celdas vecinas */
    margin: 0 auto; /* Centramos horizontalmente la tabla dentro del container */
    border-spacing: 0 0 0 2px; /* Separación de los bordes en nuestro caso no tiene efecto ya que la propiedad border-collapse */
}

.table tr {
    border-bottom: 1px solid black; /* Aplicando estilos al tr de la tabla */
    background-color: #f2f2f2; /* Cambio de fondo para cada fila */
}

.table th {
    padding: 5px; /* Añadiendo padding a los encabezados */
    font-weight: bold; /* Haciendo que los encabezados sean negritas */
    color: #0070C0; /* Cambio de color a los encabezados */
    margin-bottom: 10px; /* Añadiendo margen a los encabezados */
}

.table td {
    padding: 5px; /* Añadiendo padding a los datos */
    background-color: #e6f2ff; /* Cambio de fondo para los datos */
}
```

```
    /* Definimos el ancho de la primera columna */
    column-width: 100px;
    width: 100px;
}

/* Contenemos los valores dentro de las celdas */
td {
    text-align: center;
}

/* Aplicamos estilos a la primera celda de cada fila */
tr td:first-child {
    text-align: left;
    font-weight: bold;
    color: blue;
    background-color: cyan;
}

/* Aplicamos estilos al pie de la tabla */
tfoot {
    background-color: #ccc;
}

/* Aplicamos estilos a la clase activos (podemos hacerlo también por la posición de la columna) */
.activo {
    background-color: #00f;
}

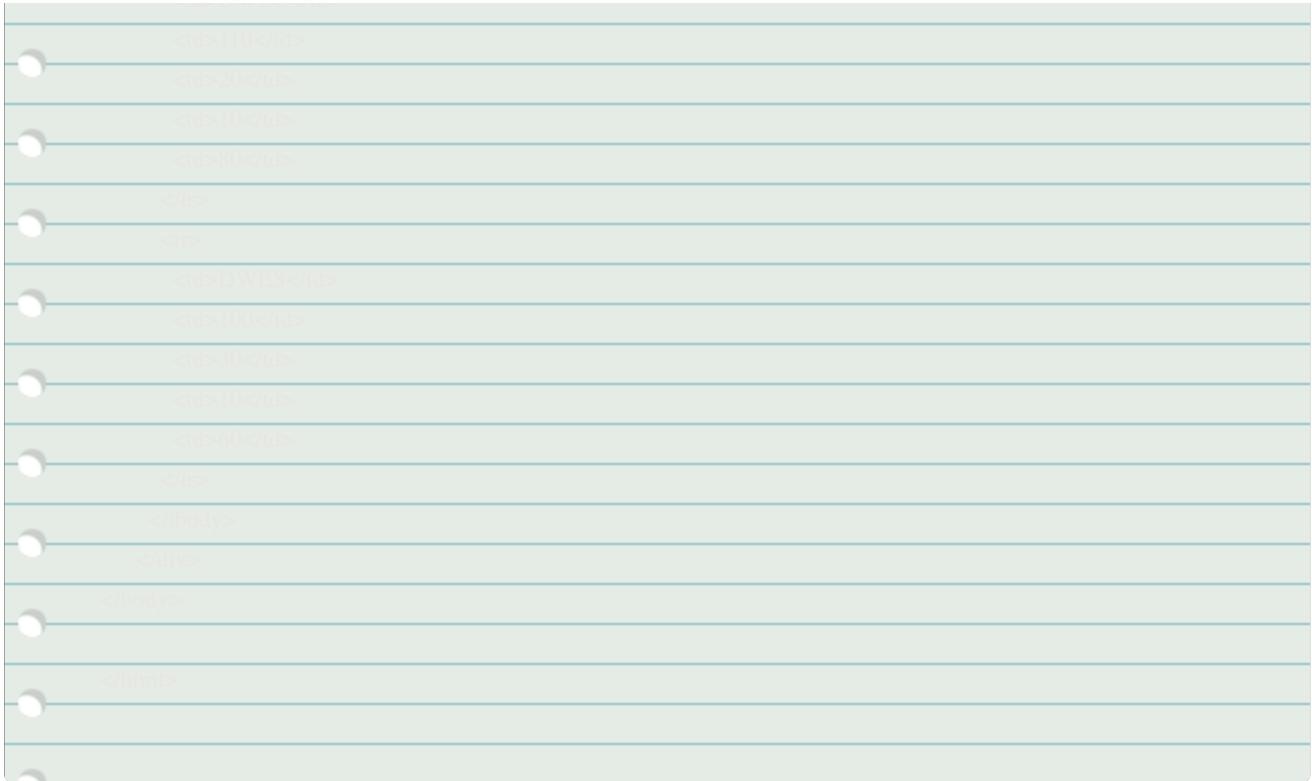
/* Aplicamos estilos a la clase inactivos */
.inactivo {
    background-color: #ccc;
}

/* Definimos un border-top que pase todo (1px) */
table tr:last-child td {
    border-bottom: 1px solid black;
}

/* Definimos fondo blanco para la última zona del pie que no es la celda */
tfoot tr:nth-last-child(1) {
    background-color: white;
}

/* Ajustamos altura para el contenido de la celda que contiene el pie (el pie tiene una altura fija) */
tfoot td {
    height: 20px;
    vertical-align: middle;
}
```





## Para saber más

Una forma común de trabajar es usando plantillas que adaptamos a nuestro gusto. En el siguiente enlace podremos ver diferentes formas de obtener código CSS referente a tablas.

 [Estilos varios para tablas](#)

## 6.2.- Las listas con CSS

---

Las listas son un elemento muy utilizado en las páginas web. Hoy en día, su empleo no está limitado a la simple enumeración de elementos en el contenido, también se utilizan para crear barras de navegación verticales y horizontales.

Para dar formato a las listas tenemos, entre otras, las propiedades: `list-style-type`, `list-style-image`, `list-style-position` y `list-style` además del pseudoelemento `marker` (visto anteriormente). Pasamos a describir estas propiedades:

### ✓ `list-style-type`

Permite elegir el marcador visual de la lista asignando a la propiedad uno de los siguientes valores:

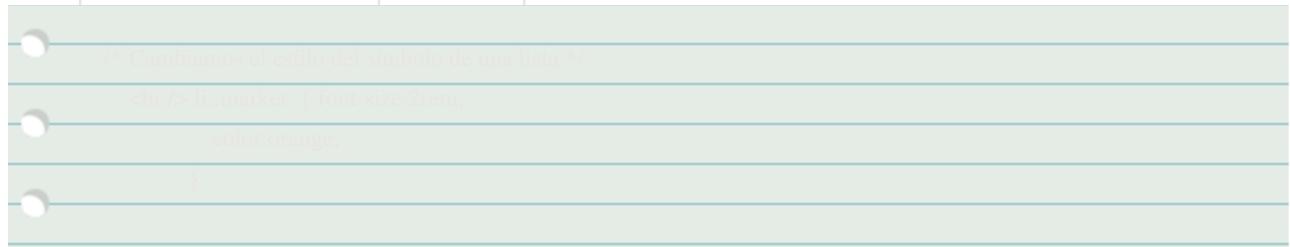
- ➡ `none`: eliminar el marcador.
- ➡ `square`: cuadrado.
- ➡ `disc`: círculo.
- ➡ `circle`: circunferencia.
- ➡ `lower-roman`: números romanos en minúscula.
- ➡ `lower-alpha`: letras en minúscula.
- ➡ `upper-alpha`: letras en mayúscula.
- ➡ `decimal`: números.
- ➡ `decimal-leading-zero`: números comenzando con cero.
- ➡  [más valores](#)



Jurgen Appelo CC BY

### ✓ `::marker`

Este pseudoelemento nos permitirá aplicarle estilos al marcador.

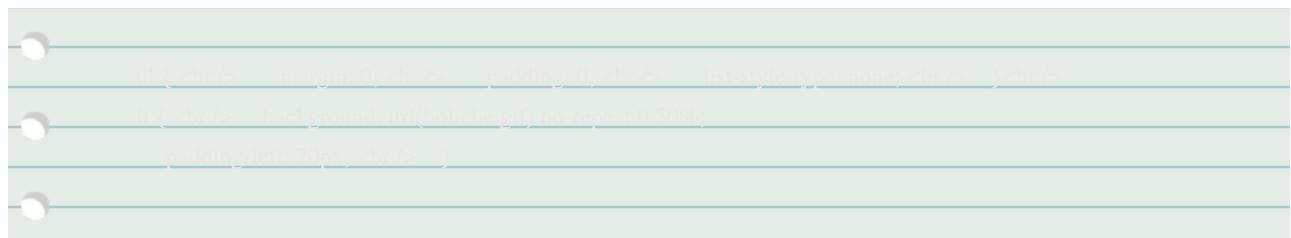


- ✓ list-style-image <br /><br />Permite especificar una imagen como marcador. Para ello deberemos indicar la dirección o URL donde se encuentra la imagen. Cuando se usa esta propiedad conviene declarar también la propiedad list-style-type en prevención de un fallo en la localización de la imagen.

Esto lo podríamos realizar, también empleando la propiedad background del elemento li. En este caso, mostrado en el ejemplo siguiente, debemos seguir los siguientes pasos:

- ◆ Eliminar previamente el marcador visual estableciendo none como valor de la propiedad list-style-type.
- ◆ Añadimos relleno a la izquierda de cada uno de los elementos de la lista.
- ◆ Colocamos de nuevo el marcador visual, declarando la propiedad background a la que asignaremos la URL de una imagen.
- ◆ Si cada elemento de la lista ocupa una sola línea, el marcador deberá centrarse verticalmente estableciendo su posición vertical al 50%.

Ejemplo:

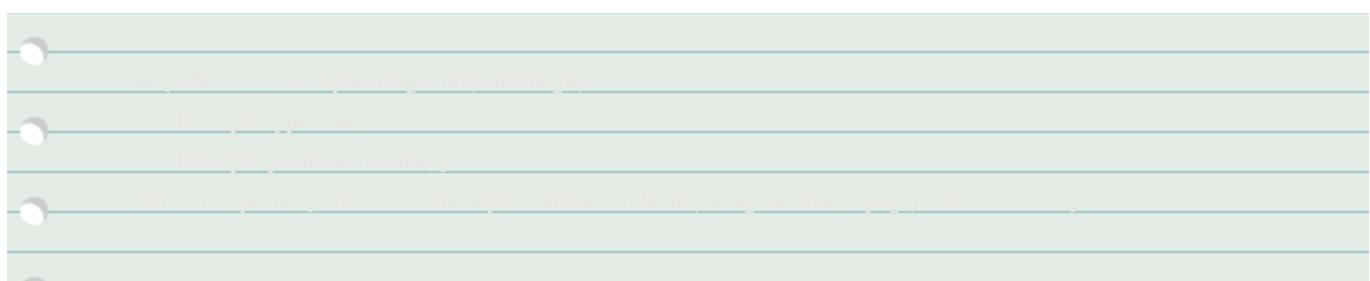


- ✓ list-style-position <br /><br />Establece la posición del marcador de los elementos de la lista. Se puede colocar el marcador dentro del área de contenido con lo que todas las líneas de este elemento estarán alineadas por la izquierda (incluida la que lleva el marcador), o se puede colocar fuera del área de contenido (como en una sangría francesa). Los valores que permiten posicionar el marcador son: inside (dentro) y outside (fuera).

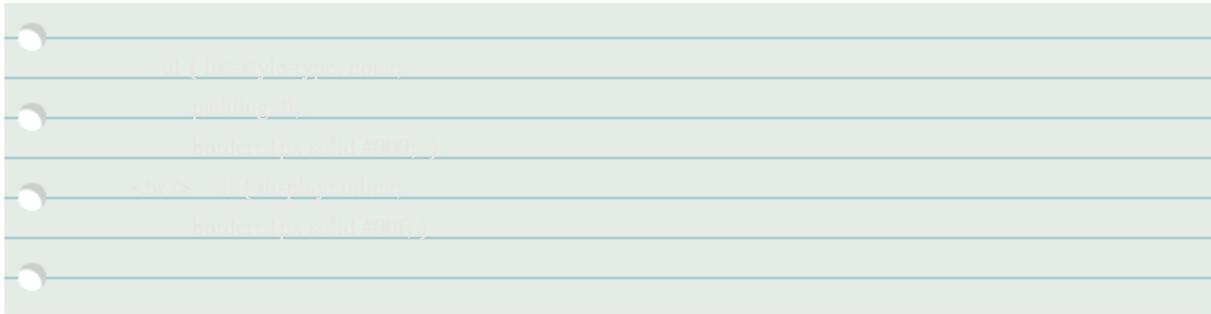
- ✓ list-style<br />

Al igual que ocurría con otras propiedades que se vieron anteriormente, esta propiedad permite configurar las listas estableciendo, de forma abreviada y en cualquier orden, el valor de una o más de las propiedades individuales vistas en este apartado.

En el siguiente ejemplo se muestra el uso de las propiedades vistas hasta ahora.



Un efecto muy utilizado para convertir una lista en una barra de menú de navegación es colocar sus elementos dispuestos horizontalmente en la misma línea. Para ello deberemos utilizar la propiedad display, mostrando un elemento de bloque en línea. El ejemplo siguiente muestra cómo hacerlo.



Si nos interesa eliminar los bordes simplemente debemos asignar el valor 0 a la propiedad border.

Es importante saber crear menús de navegación horizontales y verticales. En la siguiente presentación puedes ver con detalle cómo convertir una simple lista en un menú de navegación. En el siguiente vídeo aprenderás a realizar esta transformación.

[Resumen textual alternativo](#)

A continuación se deja el código del vídeo para poder analizarlo de forma más detenida.

## Código HTML ejemplo.

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet" type="text/css" href="estilos.css">
    <script src="https://kit.fontawesome.com/your-code-here.js" type="text/javascript">
  </head>
  <body>
    <div class="contenedor">
      <a href="https://es.wikipedia.org/wiki/HTML">HTML</a>
      <a href="https://es.wikipedia.org/wiki/CSS">CSS</a>
      <a href="https://es.wikipedia.org/wiki/Javascript">JS</a>
    </div>
  </body>
</html>
```

## Código CSS ejemplo.

```
.contenedor {
  padding: 10px;
  border-style: none;
  border-left: 1px solid black;
  width: 200px;
  background-color: orange;
}

.a {
  display: block;
  padding: 10px;
  text-align: center;
  color: white;
  font-weight: bold;
  border: 1px solid black;
}

.a:hover {
  border-left: none;
  border-bottom: 1px solid black;
  background-color: yellow;
```



## Para saber más

Con CSS podemos crear un contador y con esto, la posibilidad de numerar un conjunto de elementos de forma automática. Esto no es una lista, pero puede relacionarse con ellas, ya que el efecto puede ser algo parecido. En el siguiente enlace podemos aprender cómo con pseudoelementos se puede crear un contador utilizando las propiedades `content`, `counter-increment` y `counter-reset` entre otras.

 [Creación de un contador.](#)

## 6.3.- Formularios con CSS

Los formularios son una parte esencial de toda página web en la que se consigue la interacción con los usuarios. Son muy parecidos a los formularios de papel tradicional, solo que en el caso de la web habrá que emplear el ratón o el teclado para rellenarlo en lugar de un simple bolígrafo.

Un formulario (elemento `form` de HTML) puede contener diferentes elementos:

- ✓ Elementos en los que el usuario tendrá que escribir: cajas de texto, áreas de texto.
- ✓ Elementos que el usuario podrá o no seleccionar: botones de opción, casillas de verificación, cuadros de listas.
- ✓ Elementos decorativos o descriptivos: etiquetas, textos.
- ✓ Elementos de agrupación de otros elementos.
- ✓ Elementos que permiten limpiar el formulario o enviar los datos para su procesamiento: botones de comando.

A la hora de diseñar un formulario hay que tener claras algunas cuestiones que permitan al usuario rellenarlo fácilmente, así los campos deben tener un orden lógico, en la medida que sea posible deben ponerse primero los campos obligatorios, los mensajes utilizados deben ser claros y cada campo debe estar etiquetado correctamente.

Para poder llevar a cabo estas indicaciones nos podemos ayudar de los elementos de agrupación como son: `fieldset` y `legend` y a la vez podemos utilizar todas las propiedades y selectores vistos en la unidad. Recordemos que con respecto a los formularios son especialmente importantes las pseudoclases: `:valid`, `:invalid`, `:checked`, `:focus`, `:enabled` y `:disabled` y los pseudoelementos `::placeholder`, `::before` y `::after`.

Con los elementos de posicionamiento CSS vistos, podemos colocar los elementos del formulario en la ubicación más adecuada y podemos configurar la apariencia de los mismos según nuestros intereses con propiedades relacionadas con el color, el texto, la fuente, los fondos, etcétera.

A continuación dejamos un pequeño formulario donde se ponen en práctica muchas de las propiedades y selectores que se han citado:

### Código ejemplo

```
<form>
    <div>
        <label>Nombre: </label>
        <input type="text" value="Juan" />
    </div>
    <div>
        <label>Apellido: </label>
        <input type="text" value="Pérez" />
    </div>
    <div>
        <label>Edad: </label>
        <input type="text" value="25" />
    </div>
    <div>
        <label>Sexo: </label>
        <input checked="" type="radio" value="Masculino" /> Masculino
        <input type="radio" value="Femenino" /> Femenino
    </div>
    <div>
        <label>Número de teléfono: </label>
        <input type="text" value="987654321" />
    </div>
    <div>
        <label>Correo electrónico: </label>
        <input type="text" value="juan.perez@correo.com" />
    </div>
    <div>
        <label>Contraseña: </label>
        <input type="password" value="contraseña" />
    </div>
    <div>
        <label>Imagen: </label>
        <input type="file" value="Imagen" />
    </div>
    <div>
        <label>Botón: </label>
        <input type="button" value="Enviar" />
    </div>
</form>
```



Elaboración propia. ([CC BY-NC-SA](http://creativecommons.org/licenses/by-nc-sa/3.0/))

1

✓ Indicamos con asterisco los campos que son obligatorios (\*)

checkbox

content ("Obligatorio")

1

✓ Cuando el campo es valido el fondo lo ponemos de color verde

input valid

background-color: green;

1

✓ Aquellos campos que tienen el focus se le pone en fondo de color azul

input focus

background-color: blue;

1

✓ Se desactivan todos los campos desactivados (input disabled)

input disabled

pointer-events: none;

background-color: #cccccc;

border-color: #cccccc;

outline: none;

1

✓ Puedes la etiqueta `input` se pone con letra de estrecha

input emerald

color: blue;

1

body {

width: 100%;

background-color: white;

1

legend {

color: blue;

1

width: 100px;

height: 20px;

1

background:

1

body {

color: black;

1

background:

1

background-color: #cccccc;

border-color: #cccccc;

outline: none;

1

background-color: white;

border-color: #cccccc;

outline: none;

```
<div>
    <label>Nombre: </label>
    <input type="text" id="name" name="name" placeholder="Introduce tu nombre" />
    <span class="message"></span>
</div>
<br>
<div>
    <label>Apellido: </label>
    <input type="text" id="surname" name="surname" placeholder="Introduce tu apellido" />
    <span class="message"></span>
</div>
<br>
<div>
    <label>Correo Electrónico: </label>
    <input type="email" id="correo" name="correo" placeholder="example@correo.com" title="Introduce tu correo electrónico" />
    <span class="message"></span>
</div>
<br>
<div>
    <label>Contraseña: </label>
    <input type="password" id="password" name="password" />
    <span class="message"></span>
</div>
<br>
<div>
    <input type="checkbox" id="checkbox1" name="checkbox1" /> Acepto los términos y condiciones
</div>
<br>
<div>
    <input type="checkbox" id="checkbox2" name="checkbox2" /> Recibiré información sobre ofertas
</div>
<br>
<div>
    <input type="button" value="Enviar" />
</div>
```

Al igual que con las tablas, existen gran cantidad de plantillas para formularios que se pueden utilizar como base para nuestras webs. En el siguiente enlace puedes encontrar algunos ejemplos interesantes.

 [Ejemplos de formularios](#)

# Autoevaluación

**Marca las reglas de estilo que eliminan el marcador de los elementos de una lista, de entre las que se muestran a continuación.**

- ul {list-style-type: none; }
- ul li {list-style-type: none; }
- ul {list-style: none; }
- ul li {list-style: none; }

[Mostrar retroalimentación](#)



## Recomendación

Con la llegada de HTML5 los formularios son más fáciles de confeccionar, así se introducen nuevos campos de tipo input y nuevos atributos que permiten mejorar la entrada de datos y las restricciones asociadas a éstos. En el siguiente enlace se muestran algunas novedades que introduce HTML5 con relación a los formularios.

 [Formularios con HTML5](#)

## 7.- Herramientas y test de verificación



### Caso práctico

Poco a poco la web "**Migas de Pan**" ha ido creciendo, se ha generado bastante código. **Juan** le pregunta a **Carlos** que si el último código que ha desarrollado es correcto y **Carlos** responde con tono dubitativo que sí, a lo que **Juan** vuelve a preguntar.

¿**Carlos** le has pasado algún test a tu código?

A lo que **Carlos** responde que no.

**Juan** le comenta a **Carlos** que existen test y herramientas que pueden verificar si nuestro código es correcto. **Carlos** de inmediato comienza a pasar a su código algunos de los test que le ha comentado **Juan**.



Elaboración propia (Uso Educativo no comercial).

Una vez que hemos visto cómo dotar de estilos a nuestra página web mediante CSS, nos queda comprobar que el código que hemos generado sea correcto. Para ello podemos pasar test de verificación. Es conveniente pasar el test tanto al código HTML como al código CSS generado.

La organización W3C nos ofrece la posibilidad de testear nuestro código en las siguientes direcciones:

- ✓  [Test W3C HTML.](#)
- ✓  [Test W3C CSS](#)

Ambas páginas nos permiten dar una dirección URL, subir un fichero o pegar un trozo de código directamente para ser verificado. También nos presentan diferentes opciones de configuración (**More Options**).

Con respecto al test CSS en la configuración podemos indicar qué nivel o versión de CSS queremos validar (1,2,3,...).

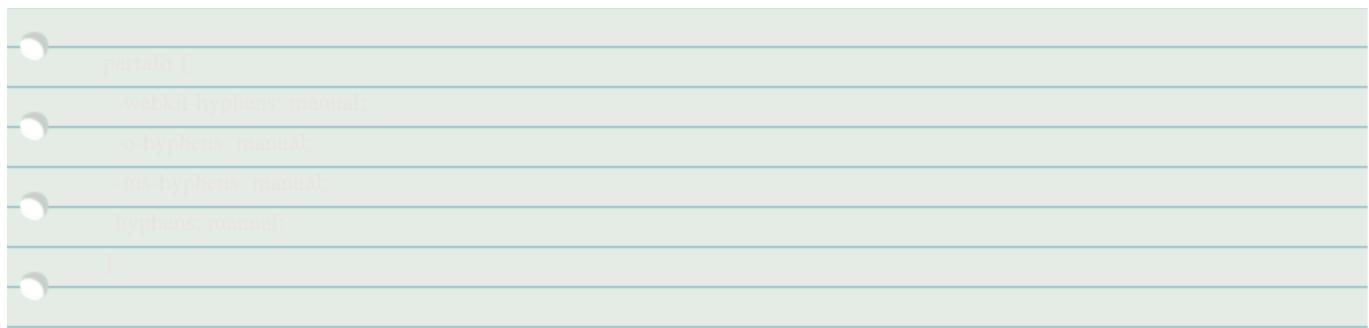
Otra herramienta importante para comprobar que nuestro desarrollo es correcto es el depurador del navegador, habitualmente accedemos a él pulsando F12. Aquí podremos analizar todos los elementos de nuestra web y cómo cambian los valores de las propiedades conforme interactuamos, este recurso muchas veces no se utiliza y es muy potente.

Dentro del depurador, también existe una opción muy interesante para comprobar cómo nuestra página web se puede ver en diferentes tipos de dispositivos, así podremos seleccionar el dispositivo que queramos y comprobar cómo se visualizaría nuestra web. Esta posibilidad nos la ofrecen otras muchas aplicaciones como  [Google Resizer](#), aunque tiene la pega que tenemos que introducir una URL.

Como se indicó al principio de la unidad, CSS está en continua evolución y hemos comprobado cómo algunas propiedades no están totalmente estandarizadas. Para estas propiedades habría que ponerle el prefijo de cada navegador para poder incorporarlas a nuestro desarrollo. Esto nos garantiza una mejor compatibilidad de nuestro código con los diferentes navegadores. A continuación indicamos los prefijos de los navegadores más importantes.

- ✓ Chrome y Safari: -webkit-
- ✓ Firefox: -moz-
- ✓ Opera: -o-
- ✓ Intentet Explorer y Microsoft Edge: -ms-

Ejemplo de cómo utilizar dichos prefijos con una propiedad experimental:



Para finalizar y a modo de resumen os dejo el enlace de una página donde se pueden probar de forma interactiva muchas de las propiedades vistas a lo largo de la unidad:

-  [Resumen de propiedades CSS](#)

# Condiciones y términos de uso de los materiales

Materiales desarrollados inicialmente por el Ministerio de Educación, Cultura y Deporte y actualizados por el profesorado de la Junta de Andalucía bajo licencia Creative Commons BY-NC-SA.



MINISTERIO  
DE EDUCACIÓN  
Y FORMACIÓN PROFESIONAL



Antes de cualquier uso leer detenidamente el siguiente [Aviso legal](#)

# Historial de actualizaciones

<b>Versión:</b> 02.00.01	<b>Fecha de actualización:</b> 16/10/20	
Actualización de materiales y correcciones menores.		
<b>Versión:</b> 02.00.00	<b>Fecha de actualización:</b> 16/06/20	<b>Autoría:</b> Jesús Moreno Ortiz
<p><b>Ubicación:</b> Toda la unidad.</p> <p><b>Mejora (tipo 3):</b> La mayor parte de la unidad está desarrollada con CSS 2.1, habría que actualizar toda la unidad a CSS3 (fuentes, tablas, modelo de cajas, colores, bordes, fondos, etc.).</p> <p>Además, introducir el concepto de diseño responsive, media query, unidades relativas, etc.</p> <p>Incluir nuevos temas CSS,</p> <p>Incluir y desarrollar elementos nuevos de maquetación como son flexbox y grid. Incluir el concepto preprocesadores de CSS.</p> <p><b>Ubicación:</b> Todo</p> <p><b>Mejora (Mapa conceptual):</b> Se actualiza con respecto a los contenidos.</p> <p><b>Ubicación:</b> Todo el documento</p> <p><b>Mejora (Orientaciones del alumnado):</b> Se adapta al nuevo formato.</p>		
<b>Versión:</b> 01.03.00	<b>Fecha de actualización:</b> 13/12/14	<b>Autoría:</b> Diego Rodríguez Gracia
Añadido tablas CSS. Glosario incrustado y erratas.		
<b>Versión:</b> 01.02.00	<b>Fecha de actualización:</b> 12/12/14	<b>Autoría:</b> Diego Rodríguez Gracia
Tablas con CSS		
<b>Versión:</b> 01.01.00	<b>Fecha de actualización:</b> 19/05/14	<b>Autoría:</b> Eliana Yemina Manzano Fernández
Ampliación a HTML5. Nuevos contenidos de CSS3.		
<b>Versión:</b> 01.00.00	<b>Fecha de actualización:</b> 19/05/14	
Versión inicial de los materiales.		