

# Manual CSS

## HOJAS DE ESTILO EN CASCADA (CSS)

Las hojas de estilo **CSS** (Cascading Style Sheets) son archivos de actualización automática del formato de la página web. Es un lenguaje usado para **definir la presentación o formato** de un documento escrito en HTML. Surgió a raíz de gestionar y desarrollar los grandes sitios web de forma más optimizada. Las características más importantes del lenguaje CSS son:

- El principal objetivo por el cual se utilizan hojas de estilo es **separar el diseño** de la página (CSS) **de la estructura** de la web (HTML). W3C recomienda tener claramente separado en diferentes archivos la combinación Estructura - Diseño - Interactividad.
- Las hojas de estilo se generan con cualquier editor de texto y **se guardan con la extensión .CSS**. Lo mas recomendable es tener todos los estilos de la web en la misma carpeta.
- Permite **controlar el estilo y el formato de múltiples páginas web** a la vez. Esto es muy positivo a la hora de gestionar grandes sitios web con multitud de páginas. Un pequeño cambio en la hoja de estilos puede afectar a muchas páginas del website.
- Todos los **navegadores son compatibles** con la tecnología CSS y reconocen e interpretan el código de un archivo .CSS. Las hojas de estilo están pensadas para informar al navegador de detalles sobre el aspecto de los objetos. Todos los navegadores interpretan perfectamente bien los estilos salvo algún caso puntual por temas de actualización.
- Con CSS es posible **disponer de diferentes diseños** para un mismo website.
- Un mismo website **puede disponer de varias hojas de estilo** que diseñan los diferentes elementos de la web. Si es un proyecto muy extenso también puede ser interesante subdividir el formato en diferentes hojas de estilo.

- Se pueden definir estilos sobre los elementos de la página que desde HTML no es posible, como por ejemplo, tamaños específicos de letra, sombreados y contornos con estilos específicos, o eliminación de las propiedades sobre la apariencia de los enlaces `<a>`.
- Como normal general las definiciones de los estilos están en uno o varios **archivos externos con extensión .css**. Existen otros métodos para especificar los estilos de la web, pero la hoja externa de estilos con enlace desde nuestra página web es el sistema más utilizado.
- Existen una serie de técnicas o metodologías enfocadas al **mantenimiento y reutilización de código CSS** que facilitan la construcción y optimización de estilos CSS como (SASS, BEM, SMACSS, OOCSS, y otras).

### Comentarios en CSS

Los comentarios en CSS comienzan por `/*` y finalizan por `*/`. Los comentarios pueden abarcar varias líneas dentro del código y se utilizan para comentar código

```
// Esto es una línea de comentario  
  
/* Esto es la forma de comentar  
un bloque de código en css */
```

### Sintaxis y estructura elementos CSS

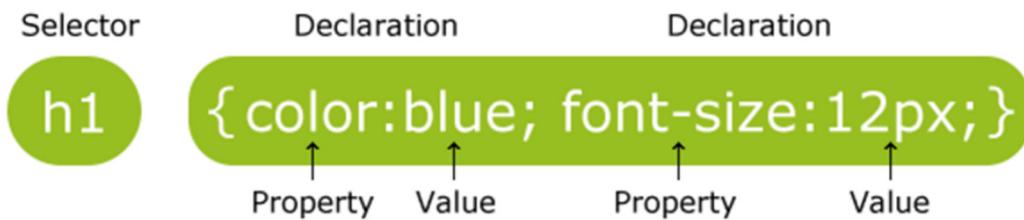
La estructura de una declaración CSS siempre sigue el siguiente esquema:

- **Selector:** Nombre del estilo o etiqueta HTML que se desea configurar.
- **Llave:** Despues del selector hay una llave y al cerrar el estilo hay otra.
- **Propiedad:** Nombre de la propiedad CSS que se desea modificar.
- **Valor:** Especifica la característica de la propiedad.

Todos los selectores pueden contener **tantas propiedades** como queramos. Es importante una buena organización para no repetir estilos que podrían afectar al formato de la web.

Algunos **valores se pueden asignar de varias formas** como por ejemplo el color de un fondo (RGB, HSLA, #Hexadecimal etc...) o el tamaño de una letra (px, em, vh, vw, %, etc...).

Todas las propiedades **se cierran con punto y coma (;**) incluyendo la última.



## Asignación de estilos CSS

Para asignar un estilo a un elemento o etiqueta de la página web se realiza habitualmente utilizando un **enlace externo a una hoja de estilos existente**. Puede contener tantos enlaces externos a hojas de estilos como sean necesarios. Esta sería la forma más habitual de utilizar los estilos, aunque existen dos sistemas más que pueden utilizarse. A continuación se explican las 3 formas:

1. Asignarle una propiedad al elemento utilizando la etiqueta 'style' seguido de los atributos necesarios (forma directa).
2. Introduciendo la declaración del estilo en la etiqueta `<head>` del documento utilizando la etiqueta `<style>` y `</style>`.
3. A través de enlaces con hojas de estilo CSS externas.

- **Método 1 (Asignación directa a la etiqueta):**

Es el sistema menos utilizado porque asigna directamente el estilo a la etiqueta y rompe con la filosofía que recomienda W3C de tener la estructura en un documento (.html) y el estilo en otro documento (.css).

En cualquier caso, este sistema puede ser útil cuando tenemos asignado un estilo genérico a un elemento y puntualmente deseamos modificar alguna propiedad sin necesidad de crear un estilo nuevo.

```
<p style="color:red; margin-left:30px;"> Texto </p>
```

- **Método 2 (Declaración estilo en el <head>):**

Se declaran los estilos dentro de las etiquetas `<head>` y `</head>` del documento. Este sistema tampoco sería el más óptimo, primero porque el `<head>` tiene mucha más información introducida como etiquetas `<meta>`, rutinas Javascript, links a frameworks, etc..., y segundo porque una hoja de estilos es muy extensa y declarar todos los estilos en el `<head>` aumentaría demasiado el volumen de nuestra web para gestionarla con facilidad.

```

<head>

    <style type= "text/css">

        body {background-color: blue}

        p {color: white}

    </style>

</head>

```

La ventaja fundamental por la cual se puede utilizar uno de los dos primeros métodos es que **NO será necesario solicitar al servidor web la hoja de estilos CSS** de la página web ya que estos se encuentran integrados dentro de ella. Esto acelera la carga de la página y evita posibles conflictos con otras hojas que pudieran estar enlazadas pero no es lo mas recomendado.

- **Método 3 (Link hoja externa)**: Es el sistema más utilizado que consiste en enlazar el archivo *.html* con la hoja de estilos *.css*. Un documento *.html* puede tener varias hojas de estilo vinculadas. Para vincular una hoja de estilos a un documento es necesario insertar la etiqueta **<link>** en el documento, entre las etiquetas **<head>** y **</head>**. Esta etiqueta no necesita etiqueta de cierre. Se pueden introducir tantas hojas de estilo como se consideren necesarias siempre en la cabecera.

**<link href="estilos.css" rel="stylesheet" type="text/css" >**



Especifica el nombre de la hoja de estilo que se va a vincular al documento. Puede ser .TXT o .CSS, lo más habitual.



Especifica el enlace a una la hoja de estilo. El valor será siempre 'stylesheet'.



Se tiene que especificar que es un documento de texto. Desde HTML5 no es necesario incluirla.

En caso de conflicto entre dos estilos que afecten al mismo elemento y con la misma propiedad CSS dispone de unas reglas internas para determinar el estilo que prevalece.

Este sistema de reglas internas contabiliza el número de selectores ID (A), de clase (B) y HTML (C) en este orden ABC. A partir de ahí se genera un número que determinará si tiene prioridad.

```
#id1 .clase1 a          (A=1, B=1, C=1 -> peso = 111)  
  
div#id1 a              (A=1, B=0, C=2 -> peso = 102)  
  
.clase1 li.clase2 a    (A=0, B=2, C=2 -> peso = 22)  
  
.clase1                  (A=0, B=1, C=0 -> peso = 10)  
  
div a                  (A=0, B=0, C=2 -> peso = 2)
```

- **Primeros:** El estilo que tenga "mas puntos" dentro de su declaración. CSS establece un sistema de puntuación en caso de estilos combinados que facilitan la prioridad entre ellos. Por ejemplo un estilo que utilice un selector de ID ( `#encabezado{..}` ) tendrá prioridad respecto a una clase que afecte al mismo tipo ( `.encabezado{..}` ).
  - **Segundo:** Si el primer criterio produce empate, la prioridad se determinará por el estilo que esté más cerca de la etiqueta. Por orden seria, mayor preferencia si el estilo está declarado dentro de la etiqueta (incrustado), posteriormente si esta dentro del mismo documento dentro de las etiquetas `<style>`, y por último en una hoja .css externa.
  - **Tercero:** Si se produce empate entre los dos primeros puntos entonces se toma como referencia el que este declarado mas tarde dentro de la hoja de estilos. Si el estilo en conflicto se encuentra en dos hojas externas independientes tendrá prioridad la última hoja declarada en la cabecera (`<head>`). Por tanto el orden de declaración de una hoja de estilo en la cabecera también puede afectar al formato de la página.
  - **Cuarto:** Todas estas reglas dejan de ejecutarse si después de la declaración introducimos el atributo **!important** que dará prioridad al estilo independientemente de las reglas y de su posible ubicación en la página.

## Selectores en CSS

Los selectores son las formas para referenciar a los diferentes elementos HTML desde CSS y asignarles un estilo. Los selectores se pueden combinar entre si para abarcar mas elementos. Una de las principales finalidades de utilizar hojas de estilo es poder indicar un mismo estilo a un grupo de elementos con idénticas propiedades.

Existen diferentes tipos de selectores que se detallan a continuación:

- **Selector Universal:** Este tipo de selector es el asterisco ( \* ) y representa a todas las etiquetas en HTML. Se utiliza para resetear los valores que vienen por defecto con algunas etiquetas como por ejemplo, `<h1>` que le da un tamaño específico a la letra, o la etiqueta `<p>` de párrafo que deja un espacio por delante y por detrás del texto. Este selector se ubica siempre al principio del documento CSS para resetear todas las propiedades de las etiquetas.

```
* {  
    padding: 0; // Sin margen interior  
    margin: 0; // Sin margen exterior  
    color: black;  
}
```

- **Selector de Elemento:** Son los más sencillos de crear y hacen referencia directamente a etiquetas de nuestro HTML. Cuando utilizamos esta etiqueta como en el ejemplo hacemos referencia a todos los elementos de la página que contienen esta etiqueta. Este tipo de selector suele ir combinado con otros selectores para acotar su radio de acción.

```
p {  
    text-align: center; // Alineación horizontal  
    color: red;  
}  
  
<p> Párrafo </p> // El estilo se carga en todas las <p>
```

- **Selector ID:** El atributo ID tiene un valor único en todo el documento. Se hace referencia a él anteponiendo el símbolo # (almohadilla). Se asignar a través del atributo 'id'. Los ID no pueden comenzar por número o por determinados caracteres especiales.

```
#parrafo1 {
    text-align: center;
    color: red;
}

<p id="parrafo1"> Titular </p> // Solo puede tener 1 id.
```

- **Selector Clase:** Es utilizada para asignar un mismo estilo a varios elementos. Se asignar a través del atributo 'class'. La clase siempre estará antepuesta por el símbolo (punto). Un elemento HTML puede tener mas de un selector de clase asignado siempre que estén separados por un espacio en blanco.

```
.centrado { text-align: center;
    color: red; }

<h1 class="centrado"> Encabezado </h1> // Llama a la clase

// Estilos de clase para etiqueta <p>.

p .centrado { text-align:center; }
p .azul { color:blue; }

// Asigna clase azul que está dentro de <p>

<p class="azul"> Texto en color azul. </p>

// Aplica dos estilos de clase a una misma etiqueta <p>

<p class="azul centrado"> Texto en color azul. </p>
```

- **Agrupación de selectores:** Se pueden agrupar selectores que comparten los mismos estilos minimizando el código CSS. Esta técnica es muy útil si realmente los selectores van a tener las mismas propiedades, en caso contrario puede originar duplicidad de código.

```
h1, h2, p { text-align: center;  
color: red; }
```

Tras conocer los selectores básicos se pueden realizar **combinaciones entre selectores** para poder cubrir diferentes elementos HTML con un solo estilo. Un combinador establece una relación entre dos selectores.

Hay cuatro combinadores diferentes en CSS3:

- selector de **descendiente** (espacio).
- selector de **hijo** (>).
- selector de **hermanos adyacentes** (+).
- interruptor general de **hermanos** (~).

- **Selector descendiente:** Afecta a todos los elementos que están dentro del anterior. Se pueden crear tantos niveles de anidamiento como se deseé.

El siguiente código afecta a todas las etiquetas `<p>` que se encuentren dentro de una etiqueta `<div>`, independientemente si se encuentran justo dentro de la etiqueta `<div>` o hay más etiquetas entre medio de la etiqueta `<p>` y `<div>`.

```
div p { background-color: yellow; }
```

El siguiente código afecta a todas las etiquetas `<p>` que se encuentren dentro de la etiqueta `<span>` y que esta a su vez esté dentro de la etiqueta `<div>`

```
div span p { background-color: yellow; }
```

- **Selector hijo (child)**: Este tipo selector afecta a todos los elementos que sean hijos inmediatos del elemento especificado. A diferencia del anterior aquí sí debe estar justo a continuación del elemento seleccionado.

El siguiente código afecta a todas las etiquetas `<p>` que se encuentren justo a continuación de la etiqueta `<div>`. Si entre medio hubiese más etiquetas entonces no se aplicaría el estilo.

```
div > p { background-color: yellow; }
```

- **Selector general de hermano**: Este tipo selecciona todos los elementos hermanos del elemento especificado.

Este código solo afecta a todas las etiquetas `<p>` que sean hermanas de la etiqueta `<div>`.

```
div ~ p { background-color: yellow; }
```

- **Selector hermanos adyacentes**: Este tipo de selector es muy poco utilizado y afecta a los selectores que son del mismo parente, o al elemento que está **inmediatamente a continuación del selector principal**.

El siguiente código afecta a todas las etiquetas `<p>` que son hermanas de la etiqueta `<div>` y se encuentran justo a continuación una de la otra a nivel de código y no hay más etiquetas entre ellas.

```
div + p {background-color: yellow;}
```

```
<div>
```

```
<p> Párrafo 1 </p>
```

```
<p> Párrafo 2 </p>
```

```
</div>
```

```
<p> Párrafo 3 </p>
```

```
<p> Párrafo 4 </p>
```



Solo aplica el estilo al Párrafo3 porque es la primera etiqueta `<p>` que hay después de la etiqueta `<div>` y entre ellas no hay ninguna etiqueta más.

- **Selectores de atributo:** Se utilizan para seleccionar un elemento con un atributo específico. Es una herramienta muy útil cuando queremos asignar estilos a elementos que contienen una serie de características en común pero no todas.

```
// Selecciona todos los elementos con el atributo target
```

```
a[target] {background-color: yellow;}
```

```
// Selecciona los elementos con el atributo target y el valor '_blank'
```

```
a[target="_blank"] {background-color: yellow;}
```

```
// Afecta a los atributos que contienen una palabra específica
```

```
[title~="flower"] {border: 5px solid yellow;}
```

```
// Afecta a todos los atributos que comienzan por la clase 'top'. El valor tiene que ser una palabra completa o separada con guion.
```

```
[class|="top"] {background: yellow;}
```

```
<h1 class="top-header">Bienvenidos</h1>
```

```
// Afecta a todos los atributos que comienzan por la clase 'top'
```

```
[class^="top"] {background: yellow;}
```

```
<h1 class="top-header">Bienvenidos</h1>
```

```
// Afecta a todos los atributos que terminen por la clase 'test'
```

```
[class$="test"] {background: yellow;}
```

```
<h1 class="first-test">Bienvenidos</h1>
```

```
// Selecciona a todos los atributos que contienen el atributo especificado 'test'
```

```
[class*="test"] {background: yellow;}
```

```
<h1 class="first-test">Bienvenidos</h1>
```

```
// Afecta a todos los elementos de formulario tipo 'button'  
  
input[type="button"] {width: 120px; margin-left: 35px;  
display: block;}
```

- **Selectores en formularios:** Los formularios son elementos de las páginas web que se pueden mejorar utilizando las técnicas CSS. Algunos efectos interesantes pueden ser:

- Fondos de color.
- Bordes redondeados.
- Ayuda con texto dentro de los elementos.
- Transiciones con los elementos del formulario.
- Actualizar estilos a través del foco del objeto (pseudo-clases).

Para crear estos estilos personalizados para formularios es necesario utilizar los selectores de atributo para hacer referencia a ellos como, por ejemplo:

- `input[type=text]` - Sólo seleccionará los campos de texto.
- `input[type=password]` - Sólo seleccionará los campos de contraseña.
- `input[type=number]` - Sólo seleccionará campos de números.

#### Ejemplos:

```
input[type=text] {  
  
border: 2px solid red;  
  
border-radius: 4px;  
  
}
```

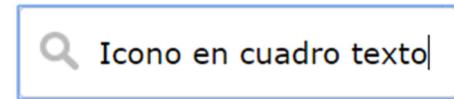


```
input[type=text]:focus {  
  
background-color: lightblue;  
  
}
```



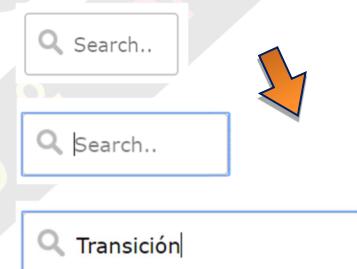
```
// Inserta icono en un cuadro de texto como imagen de fondo.
```

```
input[type=text] {  
  
background-color: white;  
  
background-image: url('searchicon.png');  
  
background-position: 10px 10px;  
  
background-repeat: no-repeat;  
  
padding-left: 40px;  
  
}
```



```
// Produce un efecto de aumento progresivo del cuadro de texto cuando el elemento toma el foco.
```

```
input[type=text] {  
  
transition: width 0.4s ease-in-out;  
  
}  
  
input[type=text]:focus {  
  
width: 100%;  
  
}
```



```
// Afecta a todos los elementos de formulario tipo <select>
```

```
select {  
  
width: 100%;  
  
padding: 16px 20px;  
  
border: none;  
  
background-color: #f1f1f1;  
  
}
```



```
// Agrupación de estilos para elementos de formulario

input[type=button], input[type=submit], input[type=reset] {

    background-color: #4CAF50;

    border: none;
    color: white;
    padding: 16px 32px;
    text-decoration: none;
    margin: 4px 2px;
    cursor: pointer;
}

textarea {

    width: 100%;
    height: 150px;
    padding: 12px 20px;
    box-sizing: border-box;
    border: 2px solid #ccc;
    background-color: #f8f8f8;
    resize: none;
}
```



Texto...

## Pseudo-clases i pseudo-elementos en CSS

Una pseudo-clase se utiliza para definir un estado especial de un elemento. Las pseudoclases se utilizan para dar estilos a elementos respecto al **comportamiento que experimentan** en un determinado momento. Las pseudoclases se definen añadiendo dos puntos antes de introducir el texto. Se pueden combinar con cualquier elemento y se utilizan básicamente para:

- Aplicar un estilo cuando nos posicionamos sobre un elemento.
- Aplicar efectos a enlaces (visitados, no visitados, activos, etc..).
- Aplicar un estilo a un elemento cuando toma el foco (elementos de formulario).

Los cuatro estados son enlaces:

- **:link** - Un enlace sin haber sido visitado.
- **:visited** - Un enlace que el usuario ha visitado.
- **:hover** - Se produce cuando se pasa con el ratón por encima del enlace o elemento.
- **:active** - Un vínculo que se hace clic en el momento.

El siguiente código muestra los 4 estados de enlace con algunas propiedades CSS asignadas.

```
a:link {color: red background-color: cyan;}  
  
a:visited {color: green text-decoration: none;}  
  
a:hover {color: hotpink;}  
  
a:active {color: blue;}
```

Otras pseudoclases (no todas) son:

- **:first-child** – (CSS3) Afecta a la primera etiqueta hija de cualquier elemento.
- **:last-child** – (CSS3) Afecta a la última etiqueta hija de cualquier elemento.

- **:nth-child()** – (CSS3) Afecta al número de etiqueta hija que se especifique empezando por arriba.
- **:nth-last-child()** – (CSS3) Afecta al número de etiqueta hija empezando a contar por el final.

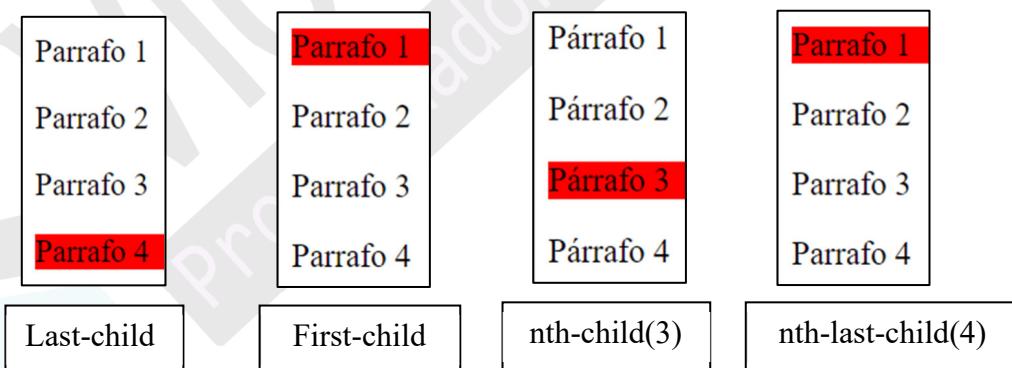
Ejemplos con diferentes combinaciones:

```
// Afecta al último hijo, en este caso a la primera etiqueta <p>
p:last-child {background: #ff0000;}
```

```
// Afecta al primer hijo, en este caso a la ultima etiqueta <p>
p:first-child {background: #ff0000;}
```

```
// Afecta al 3er hijo empezando por el principio.
p:nth-child(3) {background: red;}
```

```
// Afecta al 4º elemento empezando por el final.
p:nth-last-child(4) {background:red;}
```



```
// Afecta los elementos pares.
```

```
tr:nth-child(even) {background: red;}
```

```
// Afecta los elementos impares.
```

```
tr:nth-child(odd) {background: red;}
```

```
// Cambia el color cada dos repeticiones.  
  
tr:nth-child(2n) {background: red;}  
  
// Cambia color cada 3 repeticiones comenzando por 6º elemento.  
  
tr:nth-child(3n+6) {background: red;}  
  
// Cambia color a las 6 primeras líneas.  
  
tr:nth-child(-n+6) {background: red;}  
  
// Cambia color a las 3 últimas líneas.  
  
tr:nth-last-child(-n+3) {background: red;}
```

#### Listado de Pseudoclases de uso común:

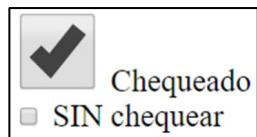
- **:checked** – (CSS3) Afecta a todos los elementos input seleccionados.

// Cambia el estilo a todas las etiquetas <label> que se encuentran a continuación de un <input> tipo checkbox seleccionado

input[type="checkbox"]:checked + label

// Modifica el estilo para todas las etiquetas <input> que se encuentran seleccionadas.

input:checked {height: 50px; width: 50px;}



Cambia de estado (alto y ancho) cuando está seleccionado.

- **:disabled** - (CSS3) Afecta a los elementos de entrada deshabilitados especificados.

```
// Cambia el estilo de todas las etiquetas <label> que están a
continuación de cualquier <input> (hermano) que esté deshabilitado
input:disabled + label{ background: yellow; }
```

- **:enabled** - (CSS3) Afecta a los elementos de entrada habilitados especificados.

```
// Afecta a todos los elementos <input> tipo texto desactivados.
```

```
input[type="text"]:disabled { background: yellow; }
```

```
// Afecta a todos los elementos <input> tipo texto activados.
```

```
input[type="text"]:enabled { background: gray; }
```

Nombre :	Mickey
Apellido:	Mouse
Ciudad :	Disneyland



Dos primeros campos  
están activados el  
tercero desactivado.

- **:out-of-range** - (CSS3) Afecta a elementos de entrada que exceden del valor establecido. Se utiliza normalmente en elementos que contienen un valor numérico.

```
input:out-of-range { border: 2px solid red; }
```

- **:in-range** - (CSS3) Aplica el estilo cuando el elemento tiene un valor dentro del rango. Se utiliza normalmente en elementos que contienen un valor numérico.

```
// Afecta a todos los elementos de tipo <input> que tienen un
valor dentro de los rangos establecidos en el form del HTML.
```

```
input:in-range { border: 2px solid yellow; }
```

- **:required** - (CSS3) Afecta a los elementos de formulario que son obligatorios.

```
input:required { background-color: yellow; }
```

- **:read-only** - (CSS3) Afecta a los elementos de formulario de **solo lectura**.

```
input:read-only { background-color: yellow; }
```

- **:read-write** - (CSS3) Afecta los elementos de formulario donde **se puede escribir**.

```
input:read-write { background-color: yellow; }
```

- **:focus** - (CSS2) Afecta al elemento de formulario que toma el foco.

```
input:focus { background-color: yellow; }
```

- **:invalid** - (CSS3) Aplica el estilo cuando el valor introducido es incorrecto.

```
input:invalid { border: 2px solid red; }
```

- **:first-of-type** - (CSS3) Afecta al primer elemento en la web de la etiqueta especificada.

```
ul:first-of-type { border: 2px solid red; }
```

- **:nth-of-type** - (CSS3) Afecta al elemento hijo especificado empezando desde el principio.

```
ul:nth-of-type(2) { border: 2px solid red; }
```

- **:last-of-type** - (CSS3) Afecta al último elemento en la web de la etiqueta especificada.

```
ul:last-of-type { border: 2px solid red; }
```

- **:nth-last-of-type** - (CSS3) Afecta al elemento hijo especificado empezando por el final.

```
ul:nth-last-of-type(3) { border: 2px solid red; }
```

- **:valid** – (CSS3) Afecta a cualquier elemento input con valor correcto.

```
input:valid { border: 2px solid red; }
```

- **:optional** – (CSS3) Afecta a todos los elementos input como obligatorios.

```
input:optional { border: 2px solid red; }
```

- **:not** – (CSS3) Aplica el estilo a todos los elementos excepto al que se encuentra ubicado en los paréntesis de la pseudoclase not. Dentro de los paréntesis podemos incluir cualquier tipo de elemento. Además podemos aplicar una pseudoclase :not a uno a varios elementos

```
// Aplica estilo a todas las secciones excepto a las de clase e1
section:not(.e1) { color: blue; }
```

```
// Aplica clase .container excepto a quien contenga class .titulo
.container:not(.titulo) { color: blue; }
```

```
// Aplica clase .container excepto al segundo hijo.
```

```
.container:not(:nth-child(2)) { color: blue; }
```

```
// Afecta a todos los botones no desabilitados de clase .container
```

```
.container button:not([disabled]) { color: blue; }
```

- **:active** – (CSS3) Aplica estilo cuando pulsamos sobre el elemento.

```
a:active { border: 2px solid red; }
```

- **:link** – (CSS3) Aplica estilo cuando el enlace no ha sido visitado todavía.

```
a:link { border: 2px solid red; }
```

- **:visited** – (CSS3) Aplica estilo cuando el enlace ha sido visitado anteriormente.

```
a:visited { border: 2px solid red; }
```

- **:hover** – (CSS3) Aplica estilo cuando nos posicionamos con el mouse sobre el elemento.

```
div>p:active { border: 2px solid red; }
```

- **:first-child** – (CSS3) Aplica estilo al primer hijo de un elemento. En este caso afecta a la primera etiqueta `<p>` que sea la primera hija directa de un elemento `<div>`.

```
div>p:first-child { border: 2px solid red; }
```

- **:last-child** – (CSS3) Aplica estilo al último hijo de un elemento. En este caso afecta a la última etiqueta `<p>` hija directa del elemento `<div>`.

```
div>p:last-child { border: 2px solid red; }
```

- **:nth-child(2)** – (CSS3) Aplica estilo al segundo hijo de un elemento. En este caso afecta a la segunda etiqueta `<p>` que es la hija directa del elemento `<div>`. Esta pseudoclase dispone de un gran número de formas para indicarle cuales de los hijos se verán afectados respecto a su elemento padre.

```
div>p:nth-child(2) { border: 2px solid red; }
```

```
div>p:nth-child(2n) { border: 2px solid red; } // Afecta pares
```

```
div>p:nth-child(2n-1) { border: 2px solid red; } // Afecta impares
```

- **:only-child** – (CSS3) Aplica estilo al elemento que es hijo único.

```
div>p:only-child { border: 2px solid red; }
```

- **:empty** – (CSS3) Aplica estilo al todos los elementos que no contengan nada.

```
div:empty { background-color: blue; }
```

- **:lang** – (CSS3) Aplica estilo al todos los elementos que contienen el atributo 'lang'.

```
p:lang(es-es) { background: blue; }
```

- **:indeterminate** – (CSS3) Aplica estilo a elementos 'checkbox', 'radio' o elemento <progress>. Afecta a todos los elementos que están sin definir (no chequeados).

```
input:indeterminate { background: blue; }
```

- **:fullscreen** – (CSS3) Aplica estilo cuando la web se encuentra en modo de página completa.

```
input:fullscreen { background: blue; }
```

- **:root** – (CSS3) Aplica estilo al elemento raíz de todo el documento.

```
input:root { background: blue; }
```

Existen algunas pseudoclases con las que W3C está trabajando pero que se encuentran en **fase experimental** por lo que no se consideran pseudoclases que formen parte del estándar todavía.

## Pseudo-elementos:

- **::after** - (CSS2) Inserta un texto después del contenido del elemento seleccionado. La propiedad 'content' es imprescindible añadirla.

```
p::after {content: " - después del texto";}
```

- **::before** - (CSS2) Inserta un texto antes del contenido del elemento seleccionado. Se puede incluso concatenar contenido a partir de un atributo de una etiqueta. La propiedad 'content' es obligatorio añadirla.

```
p::before {content: " - antes del texto";}

// Inserta el texto "El buscador" antes de la etiqueta <a>
a::before {content: " El " attr(data-name) " ";}
<a data-name="Buscador" href="http://www.bcn.es">
```

- **::first-letter** - (CSS1) Aplica un estilo a la primera letra del párrafo. Se utiliza para aplicar entre otros efectos la letra capital a un párrafo. Solo funciona en etiquetas que trabajan en bloque.

```
p::first-letter {font-size: 200%; color: #8A2BE2;}
```

- **::first-line** - (CSS1) Aplica un estilo a la primera línea del párrafo.

```
p::first-line {background-color: yellow;}
```

- **::selection** – (CSS3) Establece un estilo para el texto seleccionado por el usuario. Esta pseudoclase se insertó y luego se eliminó. Actualmente los navegadores la ejecutan correctamente y parece que será aceptada.

```
::selection {color: red; background: yellow;}
```

- **::backdrop** – (CSS3) Aplica estilo al fondo exterior de la ventana de diálogo mostrada.

```
::backdrop { border: 2px solid red; }
```

- **::input-placeholder** – (CSS3) Aplica estilo a los textos de sugerencia de los campos de entrada.

```
::input-placeholder { color: blue; }
```

## Visualización de elementos en CSS

A través de CSS se puede controlar la visualización de un elemento. La propiedad '**display**' determina como se visualizará el elemento. El valor por defecto es '**initial**' (lo que determina el estándar en HTML5), pero existen otros valores muy habituales como '**none**' (elimina el elemento de la página), '**inline**' (lo representa como elementos de línea), '**block**' (lo representa como elementos de bloque) y '**inline-block**' (es una mezcla de las propiedades '**inline**' y '**block**').

- Elementos en línea ('inline'): Los elementos en línea NO respetan los anchos, los altos, ni los márgenes verticales de los elementos. No afectan al flujo del resto de elementos en la página y ocupan el espacio mínimo necesario para mostrarse. Suelen afectar a etiquetas de contenido. Etiquetas de este tipo son, por ejemplo, `<span>`, `<a>`, `<b>`, `<img>`, `<button>`, `<label>`, `<textarea>`.
- Elementos de bloque ('block'): Estos elementos ocupan todo el ancho del elemento padre. Este elemento padre puede ser la ventana del navegador o la caja contenedora del elemento que estamos tratando. Etiquetas de este tipo son, por ejemplo, `<div>`, `<h1>` ... `<h6>`, `<p>`, `<form>`, `<header>`, `<footer>`, `<section>`.
- Elementos de bloque ajustado ('inline-block'): Estos elementos son etiquetas que son configuradas con unas medidas específicas. Es una combinación de los elementos 'inline' y los elementos 'block'. Cualquier etiqueta HTML puede ser modificada para convertirse en un elemento 'inline-block'. Este sistema permite maquetar controlando de forma más precisa el tamaño de las diferentes cajas de la web.

Los valores por defecto de los elementos se pueden modificar por otros (de bloque a línea o viceversa).

- '**display**' (CSS1) Esta propiedad determina el comportamiento de un elemento cuando debe ser mostrado en la página. Todos los elementos de forma predefinida tienen su propio 'display' asignado pero es un valor modifiable.

El valor '**none**' oculta o extrae el elemento de la web **dejando un hueco** que puede ser ocupado por otros elementos. Los valores más importantes y utilizados son '**inline**', '**block**' y '**inline-block**' (combinación de los dos primeros) aunque existen otros posibles valores:

```

div {display: block;} // elemento de línea a bloque

div {display: inline;} // Elemento de bloque a línea

div {display: inline-block;} // Combina inline y block

div {display: none;} // Oculta elemento dejando hueco

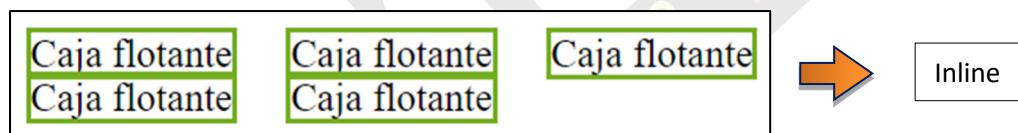
div {display: list-item;} // Se muestra como elemento de lista

div {display: initial;} // Asigna propiedad por defecto

div {display: inherit;} // hereda propiedad elemento padre

```

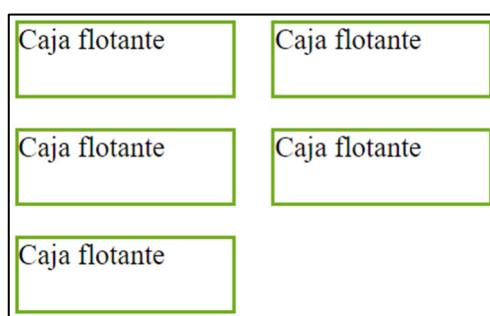
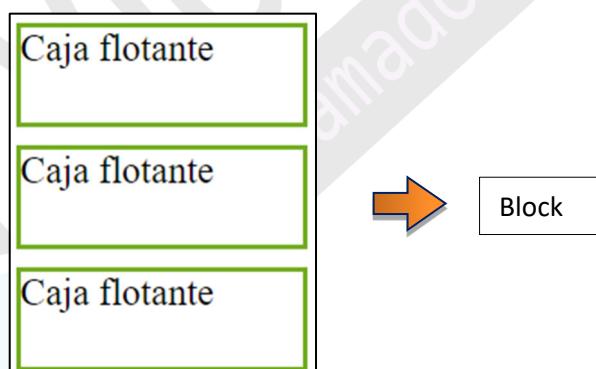
A continuación, se muestra un ejemplo de cada uno de los tres tipos de posicionamiento de los elementos en función del valor introducido en el atributo '*display*'.



```

.floating-box {
  display: inline-block;
  width: 150px;
  height: 50px;
  margin: 10px;
  border: 3px solid #73AD21;
}

```



**Inline-Block.** Los elementos se organizan en línea, pero con tamaños específicos de elementos de bloque.

- '**visibility**' (CSS2) Permite ocultar un elemento, pero a diferencia de la propiedad '*display*' el elemento **sigue ocupando un espacio en la web** y ningún otro elemento podrá ocupar esa posición.

```
h2 { visibility: hidden; } // Oculta el elemento
```

La propiedad dispone de los siguientes valores:

```
visibility: visible|hidden|collapse|initial|inherit;
```

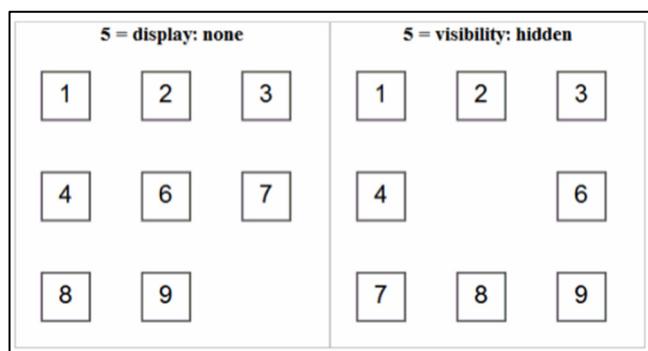
- '*visible*' – El elemento está visible por defecto.
- '*hidden*' – El elemento está invisible, pero ocupa el espacio.
- '*collapse*' – Solo para elementos de tabla y elimina las líneas de las celdas.
- '*initial*' – Establece la propiedad al valor predeterminado.
- '*Inherit*' – Hereda la propiedad del elemento padre.

Los elementos de bloque ocupan todo el ancho de la línea. Mediante '*width*' se puede establecer el ancho que ocuparán. En dispositivos pequeños hay que especificar un ancho máximo '*max-width*' porque si no el contenido sobresaldría y mostraría una barra de desplazamiento.

```
div { width: 500px; border: 3px solid #73AD21; }
```

```
div { max-width: 500px; border: 3px solid #73AD21; }
```

A continuación, se muestra como se distribuirían en la web un conjunto de cajas utilizando las dos propiedades descritas anteriormente.

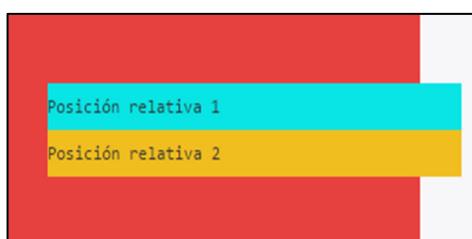


## Posición de elementos en CSS

La propiedad '*position*' especifica el método de posicionamiento de los elementos en la web. Puede disponer de 4 posibles valores (*static*, *relative*, *absolute*, *fixed*), más dos por defecto (*inherit*, *initial*). Los posibles valores de esta propiedad son:

```
position: static|absolute|fixed|relative|initial|inherit;
```

- ***initial*** – Asigna la propiedad por defecto que tiene el elemento. Por ejemplo, el valor por defecto del color del texto en la web es negro. Si en algún momento se modifica el color y se pone a rojo, es posible asignar el valor ('*initial*') que será el color negro por defecto que tendrá el elemento.
- ***inherit*** – Asigna la propiedad al elemento a partir del valor heredado del elemento padre. Por ejemplo, si tenemos una caja madre que tiene un posicionamiento relativo, asignando este valor al elemento el elemento hijo también tendrá un posicionamiento relativo. Si se modifica el color y se pone a rojo, es posible asignar el valor ('*initial*') que será el color negro por defecto que tendrá el elemento.
- ***static*** – Es la opción predeterminada del elemento. Los elementos en bloque se posicionan uno debajo de otro, y los elementos en línea se posicionan siguiendo el flujo normal de la página, es decir, de izquierda a derecha y así sucesivamente para cada línea. Los atributos 'top', 'left', 'right' y 'bottom' no se tendrán en cuenta.
- ***relative*** – En este caso **el elemento se posiciona relativamente al elemento que lo contiene** siguiendo el flujo lógico de posicionamiento de elementos. El principal problema de posicionar elementos de forma relativa es que se pueden producir solapamientos con otros elementos de la página. Los atributos 'top', 'left', 'right' y 'bottom' sí se tendrán en cuenta.

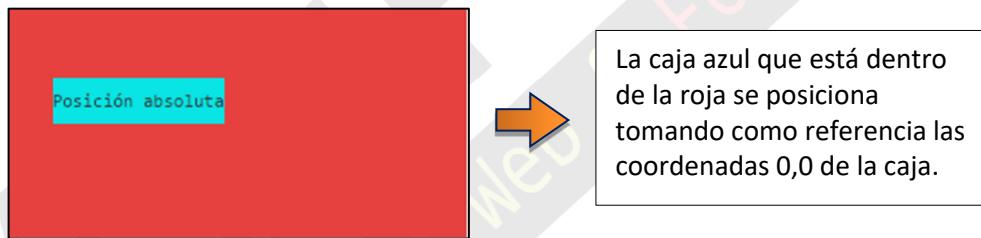


La segunda caja (naranja) se posiciona siguiendo el flujo lógico de posicionamiento que es debajo de la primera caja (azul) dentro las dos de la caja roja.

- **absolute** - Se emplea para **establecer de forma exacta** la posición en la página en la que se muestra la caja de un elemento. El elemento posicionado de forma absoluta toma como referencia su elemento contenedor `<div>` y NO forma parte del flujo natural de los otros elementos. El posicionamiento absoluto el elemento se elimina del diseño por lo que el resto de los elementos también se desplazan para ocupar el nuevo espacio libre.

La posición de las capas con la propiedad "`position:absolute`" se fija mediante unas coordenadas, que vienen dadas por los atributos '`top`' y '`left`'. El punto inicial de estas coordenadas es la esquina superior izquierda de la caja padre. En caso de solapamiento entre cajas con posición absoluta prevalece la que se haya declarado primero.

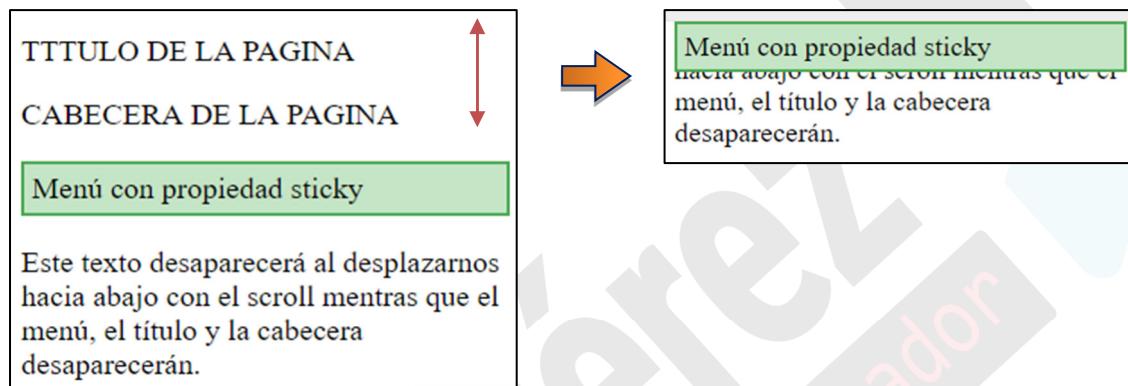
En caso de solapamiento de cajas la propiedad '`z-index`' nos permitirá establecer la prioridad de visualización entre ellas proporcionando mas relevancia aquella que tenga un número mayor.



- **fixed** – Variante del posicionamiento absoluto que **convierte una caja en un elemento inamovible** de forma que su posición en la pantalla siempre es la misma independientemente del resto de elementos. Este valor no tiene en consideración ni el posicionamiento del resto de las cajas ni los posibles desplazamientos de la página con el scroll del mouse.

Por ejemplo, si deseamos tener un banner publicitario en nuestra página en una posición específica, con este valor siempre permanecerá en el mismo sitio. A través de interactividad con Javascript podremos eliminar u ocultar el banner publicitario.

- **sticky** – Es una mezcla entre position 'relative' y 'fixed'. Son elementos denominados autoadhesivos que se fijan y se desplazan en la web. Por ejemplo, disponemos de un menú en la parte superior de nuestra web y al desplazarnos hacia abajo con el scroll el menú desaparecerá como el resto del contenido. Si queremos que el menú se quede fijo en la parte superior cuando llegue al 'top' le asignaremos esta propiedad al elemento.



```
div.sticky { position: sticky;
```

```
    top: 0; }
```

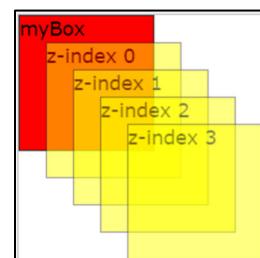
```
<div class="sticky"> Menú con propiedad sticky </div>
```

Al introducir elementos se puede provocar super posicionamiento entre ellos, y el último introducido siempre se mostrará por delante del resto. Con la propiedad 'z-index' se puede establecer un orden de prioridad de los elementos.

- 'z-index' (CSS2) Establece el orden de posicionamiento de los elementos (apilamiento). Solo funciona cuando el posicionamiento sea 'absolute', 'relative' o 'fixed'.

Los valores pueden ser 'auto' (automático) o un número que cuanto mayor sea indica que el elemento estará por delante del resto de elementos.

```
img { position: absolute;
      left: 5px;
      top: 5px;
      z-index: 5; }
```



## Elementos flotantes (float)

La propiedad '**float**' determina si un elemento debe flotar, es decir, si debe estar por encima del resto de elementos o del flujo normal de la página . Es un sistema diferente para maquetar páginas webs que se combina con la propiedad '**clear**' para generar diferentes secciones en la página. Esta propiedad se utiliza para organizar elementos de forma automática dentro de la web. Las páginas que se diseñan utilizando elementos flotantes mayoritariamente se maquetan al 100% con esta técnica.

El atributo '**float**' y '**position**' no se deben utilizar dentro del mismo estilo ya que se consideran atributos contradictorios entre si. Si se produjese esta situación la propiedad que se haya declarado en segundo término prevalecerá respecto a la primera.

- **float** (CSS1) Indica si una caja o elemento debe flotar, y de qué forma se irán posicionando los elementos flotantes que haya. Si la propiedad '*position*' tiene al valor '*absolute*' no se puede utilizar esta propiedad.

`float: none|left|right|initial|inherit;`

- '*none*' – Sin posicionamiento flotante de elementos. Es la opción por defecto y la línea queda reservada como un elemento de bloque.
- '*left*' – Los elementos se posicionarán de izquierda a derecha uno tras otro.
- '*rigth*'- Los elementos se posicionarán de derecha a izquierda uno tras otro.

```
img { float: right; margin: 0 0 10px 10px; }
```

<code>float: none;</code>	<code>float: left;</code>	<code>float: right;</code>
<p>Aquí un párrafo de texto situado antes de la imagen, dentro de un div contenedor</p>  <p>Aquí otro párrafo de texto. CSS es un lenguaje utilizado en la presentación de documentos HTML. Un documento HTML viene siendo coloquialmente "una página web". Así, podemos decir que el lenguaje CSS sirve para dotar de presentación y aspecto, de "estilo", a una página web.</p>	<p>Aquí un párrafo de texto situado antes de la imagen, dentro de un div contenedor</p>  <p>Aquí otro párrafo de texto. CSS es un lenguaje utilizado en la presentación de documentos HTML. Un documento HTML viene siendo coloquialmente "una página web". Así, podemos decir que el lenguaje CSS sirve para dotar de presentación y aspecto, de "estilo", a una página web.</p>	<p>Aquí un párrafo de texto situado antes de la imagen, dentro de un div contenedor</p>  <p>Aquí otro párrafo de texto. CSS es un lenguaje utilizado en la presentación de documentos HTML. Un documento HTML viene siendo coloquialmente "una página web". Así, podemos decir que el lenguaje CSS sirve para dotar de presentación y aspecto, de "estilo", a una página web.</p>

- **clear (CSS1)** Esta propiedad se combina con la propiedad 'float' y especifica en que lado/s de un elemento flotante no está permitido ubicar textos u otros elementos. Realiza un aclarado de elementos a izquierda o derecha de un elemento flotante. **Restaura el flujo normal del documento.** Tiene los mismos atributos que la etiqueta 'float', excepto por el valor 'both'.

`clear: none|left|right|both|initial|inherit;`

'*none*' – Permite elementos flotantes a ambos lados.

'*left*' – No permite elementos flotantes en el lado izquierdo.

'*right*' – No permite elementos flotantes en el lado derecho.

'*both*' – No permite elementos flotantes en ambos lados.

```
div { clear: left; } // Aclarado a la izquierda elemento float
```

## Colores en CSS

A través de CSS se puede especificar el color de los diferentes elementos de nuestra web. Existen diferentes formas de representar el color que se explicarán a continuación, pero cabe destacar que dentro de toda la gama de colores existen los denominados '**colores seguros**' (concretamente 127) que nos aseguran que el color seleccionado será el que realmente visualizaremos por pantalla. Además si realizamos la impresión de la web el color mostrado será exactamente igual en el papel.

Los colores en CSS se pueden especificar de **6 formas**:

- Un **nombre de color** (hay hasta 140 nombres de colores) como *red*, *blue*, *cyan*, etc... también se denominan colores SVG (o X11). NO utilizar este sistema ya que cada navegador puede interpretar un determinado color de forma diferente.

```
<h2 style="background-color:red">
```

- Como **valor RBG** "rgb (255, 0, 0)". Mezcla colores, red , green, blue.

```
<h2 style="background-color: rgb(240, 55, 150)">
```

rgb(255, 0, 0)

rgb(0, 0, 255)

- Como **valor RBGA** "rgba (255, 0, 0, 0.8 )". El último parámetro es la transparencia, denominada valor Alpha (de 0 transparente a 1 opaco).

```
<h2 style="background-color: rgba(240, 55, 150, 0.8)">
```

- Como **valor Hexadecimal** "#FF20AB".

```
<h2 style="background-color: #FF00AB">
```

#3cb371

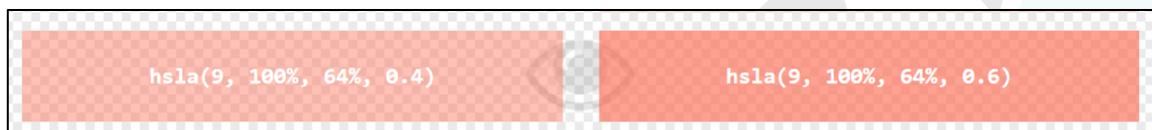
#ee82ee

- Valor para el **modelo HSL** (matiz, saturación y luminosidad) CSS3

```
<h2 style="background-color: hsl(180, 100%, 7%);">
```

- Valor para el **modelo HSLA** (matiz, saturación y luminosidad) CSS3. El último parámetro es la transparencia, denominada valor Alpha (de 0 transparente a 1 opaco).

```
<h2 style="background-color: hsla(180, 100%, 7%, 0.3);>
```



Existen webs que permiten generar colores de forma automática en cualquiera de los formatos. Los editores de texto (VSC y otros) permiten durante la edición cambiar el modelo de representación del color. Además en los editores es posible visualizar el color que se mostrará en la web ya que el mismo color incorpora el color de fondo seleccionado.

```
background-color: #F9B53C;
```

## Unidades de medida

Las unidades de medida permiten especificar los tamaños de los diferentes elementos generados en la página. Este aspecto hay que tenerlo presente cuando diseñamos páginas 'Responsive' porque una incorrecta asignación de medidas puede desquadrar por completo la página. No hay una medida mejor que otra pero sí puede ser más adecuada un tipo en función del contexto.

Hay dos tipos de medidas, las medidas absolutas y las medidas relativas.

- **Unidades absolutas:** Una medida indicada mediante unidades absolutas está completamente definida, ya que su valor no depende de otro valor de referencia. Su principal desventaja es que **son muy poco flexibles y no se adaptan** fácilmente a los diferentes medios.

**px**, píxeles. Un píxel es un punto en la pantalla. Es un valor fijo que siempre estará determinado por el dispositivo de visualización. Los dispositivos con baja resolución 1 píxel puede equivaler a la suma de 4 píxeles en un dispositivo de alta resolución, por tanto, las imágenes se deben adaptar en función de la resolución del dispositivo (media queries). De todas formas, sigue siendo un sistema de medida muy utilizado.

**in**, pulgadas ("inches", en inglés). Una pulgada equivale a 2.54 centímetros independientemente de la resolución del dispositivo. En este caso el problema es que por mucha o poca resolución que tenga un dispositivo 2.54cm siempre serán lo mismo en cualquier pantalla.

**cm**, centímetros. Tienen el mismo problema que las pulgadas.

**mm**, milímetros. Tienen el mismo problema que las pulgadas.

**pt**, puntos. Un punto equivale a 1 pulgada/72, es decir, unos 0.35 milímetros. Tiene los mismos problemas que las unidades anteriores, la adaptación en determinados dispositivos.

**pc**, picas. Una pica equivale a 12 puntos, es decir, unos 4.23 milímetros. Tiene los mismos problemas que las unidades anteriores.

- **Unidades relativas:** Las unidades relativas, a diferencia de las absolutas, no están completamente definidas, ya que su valor **siempre está referenciado respecto a otro valor**. A pesar de su aparente dificultad, son las más utilizadas en el diseño web por la flexibilidad con la que se adaptan a los diferentes medios.

**%:** (porcentaje). Es la forma más habitual de trabajar puesto que los elementos se adaptan al tamaño de los elementos que la contienen.

**em**, (no confundir con la etiqueta `<em>` de HTML). Depende del tamaño de letra del elemento. Por defecto los navegadores asignan un tamaño de letra de **16px=1em**. La 'em' es escalable y **siempre depende del elemento padre**. Es recomendable para definir tamaños de fuente. La unidad **em** hace referencia al tamaño en puntos de la letra que se está utilizando. Si se utiliza una tipografía de 12 puntos, 1em equivale a 12 puntos. El valor de 1ex se puede aproximar por 0.5 em.

**rem**, muy similar a la anterior a diferencia que **NO es escalable**, es decir, no toma la medida del elemento padre sino que accede al tamaño del body. Es recomendable para aplicar a elementos que requieran medidas fijas y para textos que deseamos que tengan un tamaño de fuente que no dependa de su elemento padre.

**ex**, relativa respecto de la altura de la letra x ("*equis minúscula*") del tipo y tamaño de letra del elemento.

**vw**, (*viewport width*). Relativa al **1% del ancho** del Viewport (dispositivo de salida). De esta forma el texto de las páginas (o cualquier propiedad que acepte este formato) es posible ajustarlo de forma automática a medida que se va redimensionando la ventana del navegador. Es una opción muy utilizada actualmente.

```
h1 { font-size: 20vw; }
```

**vh**, (*viewport height*) relativa al **1% de la altura** del Viewport (dispositivo de salida). De esta forma el texto de las páginas (o cualquier propiedad que acepte este formato) es posible ajustarlo de forma automática a medida que se va redimensionando la ventana del navegador.

Por ejemplo, un elemento `<div>` con la propiedad `'height:100vh'` provocará que el elemento ocupe el 100% de la altura del dispositivo.

```
h1 { font-size: 20vh; }
```

**vmin**, representa la medida porcentual de la **dimensión menor** de la ventana gráfica dependiendo si está el dispositivo en modo `'landscape'` o `'portrait'`.

**vmax**, representa la medida porcentual de la **dimensión mayor** de la ventana gráfica dependiendo si está el dispositivo en modo `'landscape'` o `'portrait'`. Estas dos últimas medidas se utilizan sobre todo para trabajar con dispositivos que puedan ir cambiando entre `'landscape'` o `'portrait'`.

## Fondos en CSS

Para introducir un fondo se puede realizar mediante un color sólido o una imagen. CSS dispone de una serie de propiedades que permiten trabajar con el fondo de los objetos.

- '**background-color**' Permite introducir un **color de fondo** en el elemento utilizando cualquiera de los sistemas admitidos.

```
body { background-color: lightblue; }
```

- '**background-image**' Permite introducir una **imagen de fondo** en el elemento. Las imágenes de fondo no deben dificultar la lectura del contenido. Por defecto se repiten tantas veces como sea necesario sobre el eje de las X y de las Y hasta cubrir por completo el tamaño del elemento contenedor.

```
body { background-image: url("textura.gif"); }
```

- '**background-repeat**' permite determinar si se repite en el eje de las X, en el eje de las Y o no se repite. Los valores posibles son:

- '*repeat-x*' se desplazará en el eje de las X (derecha) hasta cubrir toda la horizontalidad del contenedor.
- '*repeat-y*' se desplazará en el eje de las Y (abajo) hasta cubrir toda la verticalidad del contenedor.
- '*no-repeat*' no se repetirá el fondo en ninguno de los ejes.

```
body {  
    background-image: url("degradado.png");  
    background-repeat: repeat-x;  
}
```

- '**background-position**' especifica la posición de la imagen dentro del elemento donde se encuentra ubicado. Se deben combinar dos valores dentro de una misma propiedad, y los posibles valores permitidos son '*right*', '*top*', '*left*', '*bottom*', '*center*'. También es posible asignar la posición del background utilizando porcentajes (%) para los ejes X e Y.

```
body {  
    background-image: url("arbol.png");  
    background-repeat: no-repeat;  
    background-position: right top;  
}
```

- '**background-attachment**' Especifica que la imagen de fondo debe ser fija, es decir, no se debe desplazar con el resto de la página. Los posibles valores para la propiedad son:

- '*scroll*' permite desplazarse a la imagen con el resto de la página.
- '*fixed*' deja la imagen bloqueada como fondo fijo.
- '*local*' solo se desplazará con el elemento que la contiene,
- '*initial*' es el valor por defecto.
- '*inherit*' depende de su objeto madre.

```
background-attachment: scroll|fixed|local|initial|inherit;
```

```
body {  
    background-image: url("arbol.png");  
    background-position: right top;  
    background-attachment: fixed;  
}
```

- '**background-size**' (CSS3) Especifica la forma que se rellenará el fondo de la caja con la imagen. Los valores se pueden pasar como porcentaje, píxeles o heredado, aunque también hay alguna opción más específica como:

- '**contain**' especifica que la imagen de fondo debería ser escalada lo más grande posible mientras se aseguran ambas dimensiones al mismo tiempo siempre que haya espacio en la caja.
- '**cover**' que especifica que la imagen de fondo debe ser escalada lo más pequeño como sea posible asegurando al mismo tiempo tanto en sus dimensiones mayores o iguales que a las dimensiones correspondientes al área de posicionamiento de fondo.

```
background-size: auto|Length|cover|contain|initial|inherit;
```

```
caja {
    background: url(img_flwr.gif);
    background-size: cover;
    background-repeat: no-repeat; }
```



Contain



Cover

- '**background**' Especifica otra forma de introducir estilos sin tener que escribir todas las propiedades. Se realiza mediante el **sistema abreviado**. La propiedad abreviada para el fondo es '**background**'. En este caso todos los valores de las propiedades se escriben en la misma línea uno detrás de otro.

```
body { background: red url("img_tree.png") no-repeat right top; }
```

- 'background-origin' (CSS3) Especifica **desde que zona de la caja se empezará a dibujar el fondo 'background'**.
  - 'border-box': Empieza en el borde de la caja
  - 'padding-box': Empieza justo después del pequeño margen interior de la caja.
  - 'content-box': Empieza con el texto o contenido de la caja.

*background-origin:padding-box/border-box/content-box/inherit;*



- 'background-blend-mode' Esta propiedad permite realizar **diferentes tipos de fusionado** entre dos fondos que se encuentren dentro del mismo elemento. Es necesario que haya **como mínimo dos imágenes de fondo** sino la propiedad no funciona correctamente. Los posibles valores de esta propiedad son los que se encuentran a continuación:

*background-blend-mode: normal/ multiply/ screen/ overLay/ darken/ lighten/ color-dodge/ saturation/ color/ Luminosity;*

```
div {
    width: 400px;
    height: 400px;
    background-repeat: no-repeat, repeat;
    background-image: url("img_tree.gif"), url("paper.gif");
    background-blend-mode: lighten;
}
```

A continuación, se muestran diferentes valores de la propiedad realizando el fundido entre un degradado y una imagen de fondo.

screen

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

overlay

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

darker

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

saturation

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

color

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

luminosity

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

## Bordes en CSS

Las propiedades que afectan a los bordes del elemento **permiten especificar el estilo, el ancho, y el color del borde.**

- 'border-style' especifica qué **tipo de borde** tendrá el elemento. Los diferentes tipos de bordes con un ejemplo de código se muestran a continuación. Ninguna propiedad se hará efectiva si el atributo 'border-style' no está activo. Los posibles valores son:
  - 'dotted' - Define un borde de puntos.
  - 'dashed' - Define un borde punteado.
  - 'solid' - Define un borde continuo.
  - 'double' - Define un borde doble.
  - 'groove' - Define un borde acanalado 3D. El efecto depende de 'border-color'
  - 'ridge' - Define un borde estriado 3D. El efecto depende del valor de 'border-color'.
  - 'inset' - Define una frontera inserción 3D. El efecto depende de 'border-color'.
  - 'outset' - Define una frontera principio 3D. El efecto depende de 'border-color'.
  - 'none' – No Define ningún borde.
  - 'hidden' - Define borde oculto.

```
p.dashed { border-style: dashed; }

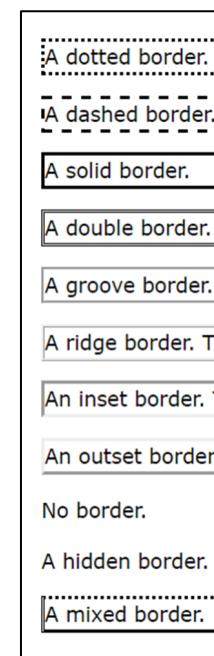
p.groove { border-style: groove; }

p.solid { border-style: solid; }

p.outset { border-style: outset; }

p.ridge { border-style: ridge; }

p.inset { border-style: inset; }
```



Es posible asignar un estilo diferente a cada uno de los bordes:

- '**Border-right-style**' borde derecho.
  - '**Border-left-style**' borde izquierdo.
  - '**Border-top-style**' borde superior.
  - '**Border-bottom-style**' borde inferior.
- 
- '**border-width**' especifica el ancho de los 4 bordes. El ancho se puede establecer mediante el uso de uno de los tres valores predefinidos: '*thin*' (fino), '*medium*' (medio) o '*thick*' (grueso) o el valor en píxeles. El ancho NO se puede especificar en porcentaje (%).

```
border-width: 5px; // Grosor por píxeles
```

```
border-width: thick; // Grosor por nombre
```

```
border-width: 2px 10px 4px 20px; // Grosor personalizado
```

Es posible asignar un borde diferente a cada uno de los bordes:

- '**Border-right-width**' borde derecho.
  - '**Border-left-width**' borde izquierdo.
  - '**Border-top-width**' borde superior.
  - '**Border-bottom-width**' borde inferior.
- 
- '**border-color**' Especifica el color de los cuatro bordes. Se puede especificar el color de las 3 formas existentes: Nombre fijo ("red"), valor hexadecimal ("#AAFF25"), o Valor en formato RGB ("255, 0, 150").

```
border-color: green; // Mismo borde 4 lados
```

```
border-color: red green blue yellow; // Cada borde 1 color
```

Es posible asignar un color de borde diferente a cada uno de los bordes:

- '**Border-right-color**' borde derecho.
- '**Border-left-color**' borde izquierdo.
- '**Border-top-color**' borde superior.
- '**Border-bottom-color**' borde inferior.

- '**border**' Método abreviado para el definir el borde. Son imprescindibles los 3 valores para que se visualice el borde en pantalla.

```
border: 5px solid red; // Por orden, width, style, color
```

Es posible especificar un borde abreviado a cada uno de los bordes:

- '**Border-right**' borde derecho.
- '**Border-left**' borde izquierdo.
- '**Border-top**' borde superior.
- '**Border-bottom**' borde inferior.

- '**border-radius**' (CSS3) Permite aplicar un redondeo a las esquinas. Es necesario que la propiedad '**border**' este asignada.

```
div {  
  
    border: 2px solid #a1a1a1;  
  
    padding: 20px 40px;  
  
    background: #dddddd;  
  
    width: 300px;  
  
    border-radius: 25px;  
}
```

Es posible especificar un borde redondeado a cada una de las esquinas:

- '**Border-top-left-radius**' borde superior izquierdo.
  - '**Border-top-right-radius**' borde superior derecho.
  - '**Border-bottom-left-radius**' borde inferior izquierdo.
  - '**Border-bottom-right-radius**' borde inferior derecho.
- 
- '**border-image-source**' (CSS3) Permite establecer una imagen como contorno del objeto.

```
#borderimg { border-image-source: url(imgborder.png); }
```

- '**border-image-width**' (CSS3) Establece el ancho de la imagen de borde.

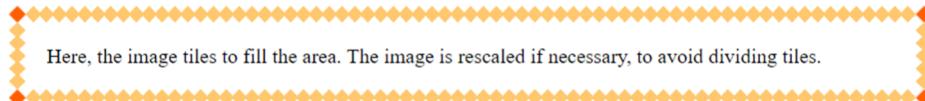
```
#borderimg { border-image-source: url(imgborder.png);  
border-image-width: 5px; }
```



- '**border-image-outset**' (CSS3) Establece cuanto sobresale la imagen que conforma el borde respecto al elemento. El valor se puede especificar en píxeles.

- '**border-image-repeat**' (CSS3) Establece como se va a repetir la imagen de contorno.  
Esta propiedad tiene los valores '*repeat*', '*round*', '*stretch*'.

#### **border-image-repeat: round:**



#### **border-image-repeat: stretch (default):**



Here is the original image:



- '**border-image-slice**' (CSS3) Establece la forma que se distribuirá el contorno de imagen alrededor del contorno.



Here is the image used:



- '**border-image**' (CSS3) Se utiliza como método abreviado para especificar todas las propiedades de 'border-image' explicadas anteriormente.

```
#borderimg { border-image: url(border.png) 30 round; }
```

## Márgenes en CSS

Los márgenes se utilizan para **generar un espacio alrededor del elemento**. Estos pueden configurarse de forma general o de forma específica para cada uno de los lados. Los valores de los márgenes se pueden especificar de diferentes formas:

- Auto: El navegador calcula el margen que debe contener el elemento.
- Longitud: Especifica un margen en px, pt, cm, %, vw, vh, etc..
- % : Especifica un margen en porcentaje de la anchura respecto al elemento que contiene.
- Inherit: Heredado del elemento padre.
  - 'margin' permite aplicar un margen exterior a los bordes. El siguiente código muestra algunas formas de especificar el margen exterior del elemento.

```
div { margin: 10px 15px 10px 8px; } // sup, der, inf, izq  
  
span { margin: 10px 15px; } // sup-inf y der-izq  
  
p { margin: 10px; } // Los 4 márgenes  
  
h1 { margin: auto; } // Centra horizontal en el contenedor  
  
header { margin: inherit; } // heredado del elemento padre
```

Es posible especificar un margen a cada uno de los bordes:

- 'margin-right' borde derecho.
- 'margin-left' borde izquierdo.
- 'margin-top' borde superior.
- 'margin-bottom' borde inferior.

## Relleno en CSS

El relleno en CSS hace referencia al **espacio entre el texto y los bordes del elemento** que lo contiene. La etiqueta utilizada es '**padding**' deshabilita un área determinada alrededor del contenido. Los valores de los márgenes interiores se pueden especificar de las siguientes formas:

- Longitud: Especifica un margen en px, pt, cm, etc..
  - % : Especifica un margen en porcentaje de la anchura del elemento que contiene.
  - Inherit: Heredado del elemento padre.
- 
- 'padding' permite aplicar un margen interior al contenido de una caja. El siguiente código muestra algunas formas de especificar el margen interior con diferentes formatos.

```
div { padding: 10px 15px 10px 8px; } // sup, der, inf, izq  
  
span { padding: 10px 15px; } // sup-inf y der-izq  
  
p { padding: 10px; } // Los 4 márgenes  
  
h1 { padding: auto; } // Centra horizontalmente.  
  
header { padding: inherit; } // Heredado del elemento padre.
```

Es posible especificar un relleno para cada uno de los bordes:

- 'padding-right' relleno derecho.
- 'padding-left' relleno izquierdo.
- 'padding-top' relleno superior.
- 'padding-bottom' relleno inferior.

## Ancho y Alto en CSS

Las propiedades '**width**' y '**height**' se utilizan para ajustar la altura y anchura. Se puede dejar en automático y significa que el navegador calcula la altura y anchura, o especificar en *los valores de longitud*, como píxeles, cm, etc., o en porcentaje (%) del bloque de contención.

- '**width**' y '**height**' Permite establecer el alto y el ancho de un elemento. No incluye el relleno, los bordes y los márgenes

```
div {  
    height: 100px;  
    width: 500px;  
    max-width: 500px;  
    background-color: powderblue;  
}
```

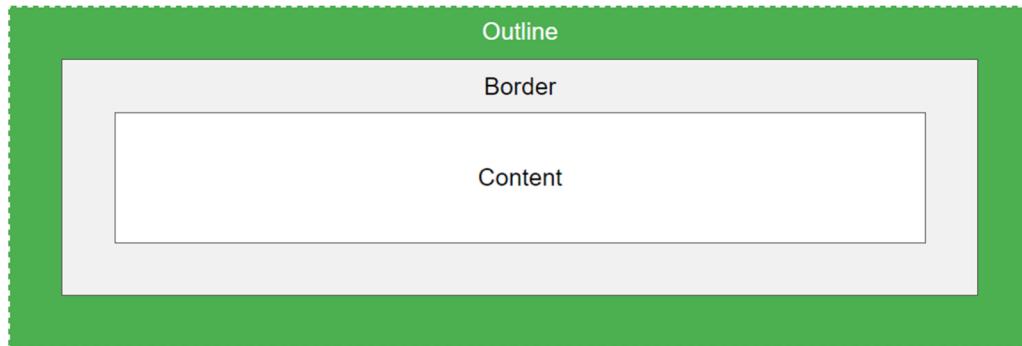
Es posible especificar un máximo y un mínimo para el alto y el ancho:

- '**max-height**' alto máximo.
- '**min-height**' alto mínimo.
- '**max-width**' ancho máximo.
- '**min-width**' ancho mínimo.

**NOTA:** Si el ancho máximo fuese mayor que el tamaño de la pantalla del dispositivo de visualización entonces se mostraría con una barra de desplazamiento. Esto puede ser incómodo para el usuario que navega por la página web.

## Líneas exteriores en CSS

Las líneas exteriores a los elementos NO son los bordes, son líneas que están por fuera de los bordes. La etiqueta para configurar estas líneas exteriores es '**outline**'. Las propiedades de esta etiqueta son las mismas que la etiqueta '**border**' excepto que **NO forma parte de las dimensiones del elemento**. Visualmente es una línea que contornea el borde y está por fuera de este (dibujo línea color verde).



- '**outline-style**' (CSS2) especifica el estilo del contorno exterior. Esta propiedad es obligatoria introducirla si se desea configurar un contorno exterior. Los posibles valores son:
  - '*dotted*' - Define un borde de puntos.
  - '*dashed*' - Define un borde punteado.
  - '*solid*' - Define un borde continuo.
  - '*double*' - Define un borde doble.
  - '*groove*' - Define un borde acanalado 3D. El efecto depende de '*outline-color*'.
  - '*ridge*' - Define un borde estriado 3D. El efecto depende del valor de '*outline-color*'.
  - '*inset*' - Define una frontera inserción 3D. El efecto depende de '*outline-color*'.
  - '*outset*' - Define una frontera principio 3D. El efecto depende de '*outline-color*'.
  - '*none*' – No Define ningún borde.
  - '*hidden*' - Define borde oculto.

```
p {
    border: 1px solid black;      // Propiedades borde
    outline-color: red;          // Color contorno exterior
}

p.dotted {outline-style: dotted;} // Estilo del contorno
```

A dotted outline.

- '**outline-color**' (CSS2) asigna el color del contorno exterior. El color se puede especificar mediante los tres sistemas básicos.

- Nombre - indique un nombre de color, como "rojo".
- RGB - especificar un valor RGB, así como "rgb (255,0,0)".
- Hexadecimal - especificar un valor hexadecimal, como "# FF0000".

```
p {
    border: 1px solid black;
    outline-style: double;
    outline-color: red;
}
```

A colored outline.

- '**outline-width**' (CSS2) asigna el ancho del contorno exterior. El ancho se puede establecer como un tamaño específico (en px, pt, cm, em, etc) o mediante el uso de uno de los tres valores predefinidos: '*thin*' (fino), '*medium*' (medio) o '*thick*' (grueso).

```
p.one {
    outline-style: double;
    outline-color: red;
    outline-width: thick;
}
```

A thinner outline.

- '**outline-offset**' (CSS3) especifica la distanza entre los bordes del elemento y el contorno exterior. No confundir con el '*padding*' o '*margin*' de otros elementos.

```
div.ext1 {  
  
    border: 2px solid black;  
  
    background-color: yellow; Distancia con el borde a 15  
  
    outline: 2px solid red;  
  
    outline-offset: 15px;  
  
}  
  
<div class="ex1"> Distancia con el borde a 15 </div>
```

- '**outline**' (CSS2) asigna los valores utilizando el método abreviado (tamaño, estilo, color), pero no la distancia respecto al contorno.

```
p {  
    border: 1px solid black;  
  
    outline: 5px dotted red;  
}
```

## Formateo de texto en CSS

Las propiedades para modificar el texto son:

- '**color**' (CSS1) Asigna el color del texto. El color se puede especificar mediante los diferentes sistemas de asignación de color. También es posible establecer un valor Alpha (transparencia) al color del elemento. Este valor siempre estará comprendido entre los valores 0 y 1.
  - Nombre - indique un nombre de color, como "rojo".
  - RGB y RGBA- especificar un valor RGB con posibilidad de asignar opacidad.
  - Hexadecimal - especificar un valor hexadecimal, como "# FF0000".
  - HSL y HSLA – mediante los valores de matiz, saturación y luminosidad

```
h1 {  
    color: green;  
    color: #00FF00;  
    color: rgb(0,0,255);  
    color: rgba(0,0,255, 0.6);  
    color: hsl(89, 43%, 51%);  
    color: hsla(89, 43%, 51%, 0.5);  
}
```

- '**text-align**' (CSS1) Establece la alineación horizontal del texto. Los valores posibles para alinear texto horizontal son 'left', 'right', 'center' y 'justify'. El valor 'inherit' hereda el estado del elemento padre.

```
div {  
    text-align: justify; // Texto justificado  
}
```

- '**text-align-last**' (CSS3) Establece la alineación horizontal del texto de la última línea del párrafo. Los valores más habituales para alinear el texto horizontal de la última fila son '*left*', '*right*', '*center*', '*auto*' (alineamiento lógico según párrafo), '*justify*', '*start*' y '*end*'. El valor '*inherit*' depende de la configuración del parent como otros tantos atributos.

```
div {
    text-align: justify;
    text-align: left;
}
```

#### **text-align-last: right:**

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoneus ut.

- '**text-decoration**' (CSS3 revisada) Esta regla elimina cualquier efecto sobre un texto como, por ejemplo, el subrayado que contienen los enlaces. Este método se utiliza como método abreviado para '*text-decoration-color*' y '*text-decoration-line*' nuevas desde CSS3.

Las propiedades de este atributo también permiten asignar pequeños efectos. Los posibles valores del atributo son:

`text-decoration: none|underline|overline|line-through|initial;`

- '*none*' – Opción por defecto muestra el texto sin efectos.
- '*underline*' - Define una línea inferior.
- '*overline*' - Define una línea superior.
- '*line-through*' - Define una línea de tachado.
- '*initial*' – Asigna la propiedad que tenga por defecto, no quita nada.

```
h1 {
    text-decoration: overline;
    text-decoration: line-through;
    text-decoration: underline overline; // Método abreviado
}
```

- '**text-decoration-color**' (CSS3) Esta propiedad permite establecer el color de la línea.

```
p {  
    text-decoration: underline;      text-decoration-color  
    text-decoration-color: red;  
}
```

- '**text-decoration-line**' (CSS3) Esta propiedad permite especificar el posicionamiento de las líneas. Es posible establecer más de una línea sobre un mismo texto.

Los valores permitidos son '*underline*' (subrayado), '*overline*' (sobrerayado), '*line-through*' (tachado) y '*none*' (no dibuja nada). El valor '*inherit*' lo hereda del elemento padre.

```
p {  
    text-decoration-line: underline overline;  
    text-decoration-color: red;          text-decoration-color  
}
```

- '**text-decoration-style**' (CSS3) Esta propiedad permite determinar el estilo de la línea decorada. Existen los tipos '*solid*' (línea sólida), '*double*' (doble subrayado), '*dashed*' (línea discontinua), '*dotted*' (punteada) y '*wavy*' (ondulado). El valor '*inherit*' permite asignarle la misma característica que el elemento padre.

```
p {  
    text-decoration-line: overline;  
    text-decoration-style: wavy;        text-decoration-style  
    text-decoration-color: red;  
}
```

- '**text-transform**' (CSS1) Especifica el texto en mayúscula, minúscula o la primera letra de todas las palabras del texto en mayúscula.

```
text-transform: none|capitalize|uppercase|lowercase|initial;
```

```
p {
    text-transform: uppercase;
    text-transform: lowercase;
    text-transform: capitalize;
}
```

TODO EN MAYÚSCULA.

todo en minúscula.

Primera Letra Cada Palabra Mayúscula.

- '**text-indent**' (CSS1) Especifica el sangrado de la primera línea de un texto. Los valores de la sangría se pueden especificar en % o en unidades (px, cm, pt, em, etc...). Si el valor se expresa en % , el valor 100% toma como referencia la longitud máxima de la línea.

```
text-indent: length|%|initial|inherit;
```

```
p {
    text-indent: 50px;
    text-indent: 50%;
```

In my younger  
advice that I've been  
like criticizing anyone

- '**letter-spacing**' (CSS1) Especifica la distancia entre los caracteres de un texto. No confundir con la distancia entre palabras '*word-spacing*'. Esta propiedad combinada con el tamaño de la fuente puede dar cierta notoriedad a títulos o apartados en un documento.

```
h1 { letter-spacing: 3px; }
```

```
h2 { letter-spacing: -3px; }
```

**Espaciado grande**

**Espaciado pequeño**

- '**line-height**' (CSS 2.1) Especifica el interlineado de un párrafo siempre tomando como referencia la altura de la letra. El valor por defecto es '*normal*' pero se puede especificar el valor en %, número o pixeles. El valor 1.6 es muy utilizado porque deja una distancia adecuada. El valor 100% deja las líneas demasiado juntas y se recomienda en este caso utilizar entre 110% y 120% según el navegador utilizado.

Esta propiedad se utiliza y es útil cuando deseamos alinear verticalmente una sola línea de texto. Si asignamos a '*line-height*' la misma altura que la caja centrará el texto.

```
line-height: normal|number|length|initial|inherit;
```

```
p.small {line-height: 0.8;}
```

Interlineado pequeño
Interlineado pequeño

- '**direction**' (CSS2) Establece la dirección de escritura. Los valores por defecto son:

```
direction: ltr|rtl|initial|inherit;
```

```
div { direction: rtl; } // Texto de izquierda a derecha
```

```
div { direction: ltr; } // Texto de derecha a izquierda
```

- '**word-spacing**' (CSS1) Especifica el espacio en blanco entre las palabras de un texto. No confundir con la distancia entre las letras de un texto '*letter-spacing*'.

```
Word-spacing: normal|length|initial|inherit;
```

```
h1 { word-spacing: 10px; }
```

Con espacio

```
h2 { word-spacing: -5px; }
```

Sin espacio

- '**text-shadow**' (CSS3) especifica la sombra del texto. Esta sombra es configurable con diferentes valores.

```
text-shadow: h-shadow v-shadow blur-radius color|none|initial;
```

- '*h-shadow*' – Desplazamiento horizontal de la sombra.
- '*v-shadow*' - Desplazamiento vertical de la sombra.
- '*blur-radius*' – Desenfoque de la sombra.
- '*color*' - Define el color de la sombra. Se puede asignar por cualquier método.
- '*none*' – Sin sombra.
- '*Initial*' – El valor que tenga por defecto.

```
// Los valores se especifican por orden (h, v, blur, color)
```

```
h1 { text-shadow: 5px 5px 3px red;}
```

## The text-shadow Property

- '**text-overflow**' (CSS3) especifica de qué forma se mostrará el texto en la caja cuando este sobresalga (overflow) de la misma. Los posibles valores de la propiedad son:
  - '*clip*' – Recorta el texto por la finalización de la caja.
  - '*ellipsis*' – Asigna unos puntos suspensivos antes de cerrar la caja.

```
div {text-overflow: clip;}
```

Lorem ipsum dolor sit amet,

```
div {text-overflow: ellipsis;}
```

Lorem ipsum dolor sit ame...

- 'vertical-align' (CSS1) Establece la **alineación vertical de un elemento**. Se utiliza para asignar subíndices, superíndices, en línea con el texto, por encima o por debajo de la línea de escritura. El valor también se puede determinar mediante %, px, o cm.

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

```
img { vertical-align: text-top; }
```

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

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

```
img { vertical-align: 25px; }
```

```
img { vertical-align: -10px; }
```

```
img { vertical-align: 10%; }
```

## Fuentes en CSS

Las propiedades de las fuentes determinan las características de tamaño, tipo de letra, efectos, etc..

Los tipos de fuentes pueden ser de los siguientes tipos:

- **Fuentes Serif:** Son las fuentes que **tienen un acabado más artístico** y elaborado.

Serif	Times New Roman
	Georgia

- **Fuentes Sans Serif:** Son fuentes **SIN efectos especiales o acabados artísticos muy laboriosos.**

Son fáciles de leer y no se producen efectos de solapamiento entre las letras.

Sans-serif	Arial
	Verdana

- **Fuentes Monospace:** Son fuentes que tienen el mismo espacio entre las letras. Son fáciles de leer tanto para las personas como para los pc's. Algunas de estas fuentes pueden disponer de leves acabados artísticos.

Monospace	Courier New
	Lucida Console

Las propiedades para modificar las características en CSS son:

- '**font-family**' (CSS1) Establece la fuente de la letra. Puede contener varios valores por si el navegador no encuentra la primera o no está disponible sigue con la siguiente. Si el nombre de una fuente tiene más de una palabra esta deberá ir entre comillas.

```
p { font-family: "Times New Roman", Times, serif; }
```

- '**font-style**' (CSS1) Establece un efecto de cursiva o inclinado del texto. La cursiva y el inclinado visualmente se parecen mucho, pero son diferentes efectos.

```
p.normal { font-style: normal; }  
  
p.italic { font-style: italic; }  
  
p.oblique { font-style: oblique; }
```

- '**font-size**' (CSS1) Establece el tamaño de la fuente. Los tamaños de las letras pueden ser de dos tipos:

#### Tamaño absoluto:

- Establece el texto a un tamaño especificado
- No permite a un usuario cambiar el tamaño del texto en todos los navegadores (mal por razones de accesibilidad)
- Tamaño absoluto es útil cuando se conoce el tamaño físico de la salida

#### Tamaño relativo:

- Establece el tamaño en relación con los elementos que rodean
- Permite a un usuario cambiar el tamaño del texto en los navegadores

**NOTA:** Si no se especifica un tamaño de fuente, el predeterminado para textos es 16px, equivalente a 1em que es la medida recomendada por W3C.

```
p { font-size: 14px; }           // Tamaño de la letra  
  
p { font-size: 0.875em; }        // 14px/16=0.875em  
  
p { font-size: 2.5em; }          // 40px/16=2.5em  
  
p { font-size: 100%; }           // Expresado en porcentaje
```

- '**font-weight**' (CSS1) aplica un efecto de negrita. El valor puede ser '*bold*', '*bolder*', o determinar un valor numérico que va de 400 a 700. Visualmente en el navegador solo hay un cambio entre negrita-no negrita, es decir, no hay una progresión en el aumento.

```
p { font-weight: bold; }
```

```
p { font-weight: normal; }
```

```
p { font-weight: 600; }
```

- '**font-variant**' (CSS1) aplica un efecto de letra pequeña, pero con todas sus letras en mayúsculas. Estas mayúsculas son de un tamaño menor que una mayúscula original. Los valores permitidos son '*normal*' y '*small-caps*'.

```
p { font-variant: small-caps; } // Letra pequeña en mayúsculas
```

LETRA PEQUEÑA EN MAYÚSCULAS

## Importar fuentes ( @font-face )

Existen dos formas para importar fuentes a nuestra página web.

### 1. A traves de CDN. (recomendado)

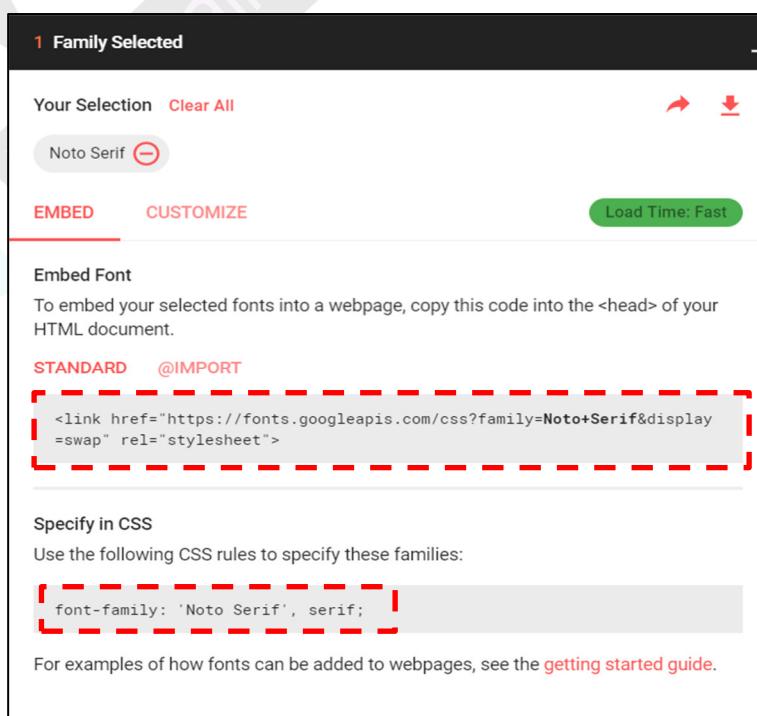
Es el método recomendado en la actualidad para utilizar muchos recursos que se encuentran en la red (fuentes, iconos, imágenes, etc...) ya que se obtienen de lugares reconocidos que nos ofrecen mayor seguridad y rapidez de carga de la información, reducción de costes, y ayuda al posicionamiento SEO.

Para utilizar este sistema tenemos que crear primero un link en nuestro `<head>` con el CDN (Content Delivery Network) que deseemos utilizar que dispone del contenido. A continuación en nuestro código deberemos hacer referencia al elemento que utilizará el recurso.

#### Ejemplo para utilizar un tipo de letra de Google Fonts

Una vez seleccionada la fuente de Google y la hemos personalizado según nuestros criterios, la misma página nos informa del link que hay que introducir en nuestra web y del código para utilizarla en nuestro código CSS. Es posible descargarse varias fuentes a la vez.

Esta forma también nos permite descargar las fuentes en forma de archivo en cada uno de los diferentes formatos necesarios si lo quisiéramos utilizar a través de nuestro hosting.



The screenshot shows the 'Family Selected' section of the Google Fonts website. It displays the font 'Noto Serif' under 'Your Selection'. Below it are two tabs: 'EMBED' (selected) and 'CUSTOMIZE'. A green button labeled 'Load Time: Fast' is visible. The 'EMBED' section contains an 'Embed Font' section with instructions to copy the provided code into the `<head>` of an HTML document. Two options are shown: 'STANDARD' and '@IMPORT'. The '@IMPORT' option is selected, and its corresponding code is highlighted with a red dashed box:

```
<link href="https://fonts.googleapis.com/css?family=Noto+Serif&display=swap" rel="stylesheet">
```

The 'Specify in CSS' section below contains the CSS rule:

```
font-family: 'Noto Serif', serif;
```

This rule is also highlighted with a red dashed box. At the bottom, there's a link to the 'getting started guide'.

## 2. Descarga de fuentes

Este sistema utiliza la descarga de la fuente en **varios archivos de diferentes formatos** que se encontrarán ubicados en el hosting de nuestra página. El problema de utilizar este sistema es el tiempo de carga ya que a mayor numero de fuentes utilizadas en nuestra web el número de archivos crece de forma considerable.

El formato True Type es ampliamente soportado por navegadores modernos. Otros formatos son el OpenType, SVG y WOFF. El formato SVG está destinado para dispositivos móviles y el formato WOFF se creó para unificar todos los diferentes estilos disponibles, es decir, si hay una fuente tipo WOFF esta será legible por todos los navegadores, pretende ser el estándar.

Las fuentes **OpenType** están relacionadas con las fuentes TrueType, pero **incorporan una extensión más amplia del juego de caracteres básico**, como el uso de mayúsculas pequeñas, números en estilo antiguo y formas más detalladas, como glifos y ligaduras. Las fuentes OpenType también se pueden escalar a cualquier tamaño y resultan claras y legibles en todos los tamaños.

Esta diversidad de formatos de fuente hace necesario que indiquemos la mayor cantidad de formatos de fuentes posibles para nuestro sitio proporcionando todos los formatos posibles.

Para importar una fuente que será utilizada por una página web tenemos que utilizar la regla CSS **@font-face**, en la misma indicamos el nombre de la fuente a importar y la dirección web de donde la debe descargar el navegador, la sintaxis es:

```
@font-face {  
    font-family: [nombre de la fuente];  
    src: local(""),  
          url("nombreachivo.woff") format("woff"),  
          url("nombreachivo.otf") format("opentype"),  
          url("nombreachivo.svg#nombre de la fuente") format("svg");  
}
```

Podemos descargar una fuente del sitio [www.fontsquirrel.com](http://www.fontsquirrel.com) o de [Google Fonts](#) pero con **link externo**. Cuantos más formatos se disponen de la misma fuente es menor la probabilidad error.

En la regla **@font-face** hacemos referencia a cuatro fuentes con distintos formatos y definimos su nombre en la propiedad font-family:

```
// Creación del tipo de fuente en la web desde archivos externos (4)

@font-face {

    font-family: "Ubuntu"; // Nombre que tendrá la fuente

    src: url("Ubuntu-Title-webfont.eot") format("eot"),

        url("Ubuntu-Title-webfont.woff") format("woff"),

        url("Ubuntu-Title-webfont.ttf") format("truetype"),

        url("Ubuntu-Title-webfont.svg#UbuntuTitle") format("svg");

}
```

Ahora podemos utilizar la fuente "*Ubuntu*" que acabamos de crear:

```
// Estilo elemento <p> con la fuente

#recuadro1 p {

    font-family: Ubuntu;
    color: #ff0000;
    font-size: 20px;

}
```

#### Recuadro 1

lorem ipsum dolor sit amet, consectetuer adipiscing elit. aenean commodo ligula eget dolor. aenean massa. cum sociis natoque penatibus et magnis dis porturient montes, nascetur ridiculus mus. donec quam felis, ultricies nec.

Font format					
TTF/OTF	4.0	9.0*	3.5	3.1	10.0
WOFF	5.0	9.0	3.6	5.1	11.1
WOFF2	36.0	Not supported	35.0*	Not supported	26.0
SVG	4.0	Not supported	Not supported	3.2	9.0
EOT	Not supported	6.0	Not supported	Not supported	Not supported

## Iconos en CSS

Los iconos son dibujos que se insertan en las páginas. La forma más fácil para insertar un ícono es utilizando una **biblioteca de íconos**.

Las características más significativas de las bibliotecas son:

- Son **totalmente gratuitas**, no hay que pagar por ellas.
- No es necesario descargar **ningún tipo de plug-in adicional**.
- Son **vectores escalables** lo que permite modificar el tamaño del ícono sin distorsionarlo.
- Se pueden **personalizar** los estilos (tamaño, color, sombra, etc...) como cualquier elemento.
- Los íconos se pueden **introducir en botones de formulario**.

Para utilizar una biblioteca de íconos hay que insertar entre las etiquetas `<head>` el código que enlaza con la librería. Mediante las etiquetas correspondientes `<i>` o `<span>` se podrán introducir los íconos en la web incluyendo el estilo si fuese necesario.

Cada biblioteca tiene sus particularidades a nivel de código cuando se desea insertar los íconos, aunque el proceso es muy similar en todas ellas. Las bibliotecas más utilizadas de forma más habitual son las que se muestran a continuación, aunque existen muchas más.

- **Biblioteca de Íconos de referencia.** ([Listado íconos, están por categorías](#))

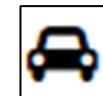
```
<head>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
</head>
```

```
<i class="fa fa-cloud"></i>
```



```
<i class="fa fa-heart"></i>
```

```
<i class="fa fa-car" style="font-size:24px;"></i>
```



Los iconos de referencia permiten realizar más configuraciones que en el resto de librerías. Por ejemplo, dispone de una serie de clases que aplican varios efectos.

- Aumentar iconos: `fa-2x`, `fa-3x`, `fa-4x`, o `fa-5x`

```
<i class="fa fa-car fa-3x"></i>
```

```
<i class="fa fa-car fa-4x"></i>
```

- Listas de iconos: `fa-ul` y `fa-li`

```
<ul class="fa-ul">
```

```
<li><i class="fa-li fa fa-check-square"></i>Icon1</li>
```

```
<li><i class="fa-li fa fa-spinner fa-spin"></i>Icon2</li>
```

```
<li><i class="fa-li fa fa-square"></i>Icon3</li>
```

```
</ul>
```

- Iconos animados: `fa-spin` (rota la figura) y `fa-pulse` (gira en 8 pasos).

```
<i class="fa fa-spin"></i>
```

```
<i class="fa fa-spin" fa-spin="90"></i>
```

```
<i class="fa fa-spin" fa-spin="180"></i>
```

```
<i class="fa fa-spin" fa-spin="270"></i>
```

```
<i class="fa fa-spin" fa-spin="horizontal"></i>
```



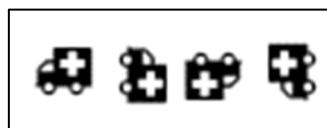
- Rotar y voltear iconos: `fa-rotate-*` (rotar la figura) y `fa-pulse-*` (voltear la figura)

```
<i class="fa fa-shield"></i>

<i class="fa fa-shield fa-rotate-90"></i>

<i class="fa fa-shield fa-rotate-180"></i>

<i class="fa fa-shield fa-rotate-270"></i>
```



- Iconos apilados: `fa-stack` (pila), `fa-stack-1x` (tamaño regular), `fa-stack-2x` (icono más grande), `fa-inverse` (icono con color alternativo).



- Biblioteca de Google Icons. ([Listado de iconos, están por categorías](#))

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

<i class="material-icons">cloud</i>

<i class="material-icons">favorite</i>

<i class="material-icons">attachment</i>   

<i class="material-icons">computer</i>
```

**NOTA:** En la librería de iconos de Google hay que especificar el nombre del ícono fuera del nombre de la etiqueta mientras que en las otras librerías no es necesario.

- **Biblioteca de Bootstrap.** ([Lista de referencia](#))

```
<head>
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
</head>

<i class="glyphicon glyphicon-cloud"></i>
<i class="glyphicon glyphicon-remove"></i>
<i class="glyphicon glyphicon-user" ></i>
```



## Listas en CSS

Hay dos tipos de lista en CSS, listas ordenadas `<ol>` (símbolos que se puedan ordenar) y listas desordenadas `<ul>` (viñetas). Las propiedades para poder modificar los elementos de una lista son:

- '`list-style-type`' (CSS1) especifica el tipo de marcador del elemento lista.

```
ul.a {
    list-style-type: circle; // Circulo
    list-style-type: square; // Cuadrado
    list-style-type: upper-roman; // Números romanos
    list-style-type: lower-alpha; // Alfabética minúscula
}
```

- '`list-style-image`' (CSS1) permite ubicar como marcador una imagen.

```
ul {list-style-image: url('aro.gif');} // Imagen marcador
```

- '`list-style-position`' (CSS1) especifica si los marcadores de la lista de artículos deben aparecer dentro o fuera del flujo del contenido. Los posibles valores son '`inside`' (dentro) y '`outside`' (fuera).

```
ul { list-style-position: inside; }
```

• café
• Té
• Coca Cola

• café
• Té
• Coca Cola

- '`list-style`' (CSS1) es la propiedad para el método abreviado.

```
ul { list-style: square inside url("aro.gif"); }
```

## Tablas en CSS

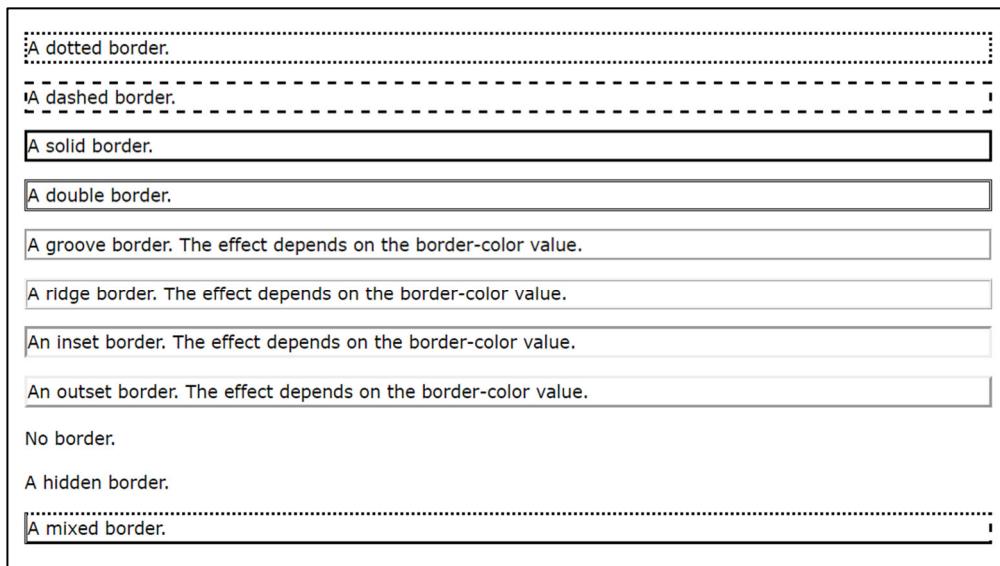
Las tablas generadas con HTML5 se pueden modificar como cualquier elemento con CSS. Las tablas se generan con las etiquetas `<table>`, `<th>` (encabezados), `<tr>` (filas) y `<td>` (columnas). Las propiedades CSS para modificar las tablas son:

- '`border`' (CSS1) especifica las características del borde, es la propiedad abreviada. Las propiedades abreviadas se deben introducir en el siguiente orden y son '`border-width`', '`border-style`', y '`border-color`'.

```
table, th, td {  
  
    border: 1px solid black; // Ancho, estilo, y color  
  
}
```

La propiedad '`border-style`' dispone de los siguientes valores:

- '`dotted`' - Define un borde de puntos.
- '`dashed`' - Define un borde punteado.
- '`solid`' - Define un borde continuo.
- '`double`' - Define un borde doble.
- '`groove`' - Define un borde acanalado 3D. El efecto depende de '`border-color`'.
- '`ridge`' - Define un borde estriado 3D. El efecto depende del valor de '`border-color`'.
- '`inset`' - Define una frontera inserción 3D. El efecto depende de '`border-color`'.
- '`outset`' - Define una frontera principio 3D. El efecto depende de '`border-color`'.
- '`none`' – No Define ningún borde.
- '`hidden`' - Define borde oculto.



- '**border-collapse**' (CSS2) determina si entre las celdas de una tabla habrá una pequeña separación '*separate*' o estarán en contacto '*collapse*'.

```
table {
    border-collapse: collapse;
    border-collapse: separate;
}
```

A table cell	A table cell
A table cell	A table cell

A table cell	A table cell
A table cell	A table cell

- '**width**' y '**height**' (CSS1) determina el ancho y el alto de la tabla, de las filas o columnas que contenga siempre que se haga referencia al elemento en el estilo.

```
table { width: 100%; }
```

- '**caption-side**' (CSS2) determina la **posición del título** de la tabla. Puede contener dos valores '*top*' (superior) y '*bottom*' (inferior). Para que el valor se haga efectivo la tabla debe utilizar la etiqueta <caption> y estará ubicada justo continuación del primer <table>

```
table { caption-side: bottom; }
```

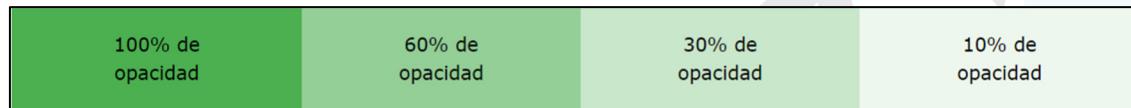
Hay otras propiedades que afectan a las tablas, pero son propiedades genéricas que también pueden afectar a otros elementos y que se han comentado. A continuación, se muestran una serie de atributos (no están todos) que pueden afectar a los elementos de una tabla:

```
td {  
    text-align: left;  
  
    height: 50px;  
  
    vertical-align: bottom;  
  
    padding: 15px;  
  
    border-bottom: 1px solid #ddd;  
  
    background-color: #f5f5f5;  
  
    color: white;  
}
```

## Opacidad en CSS

La opacidad permite dar un cierto grado de transparencia o opacidad a un elemento como cajas o imágenes. La propiedad 'opacity' permite trabajar esta característica.

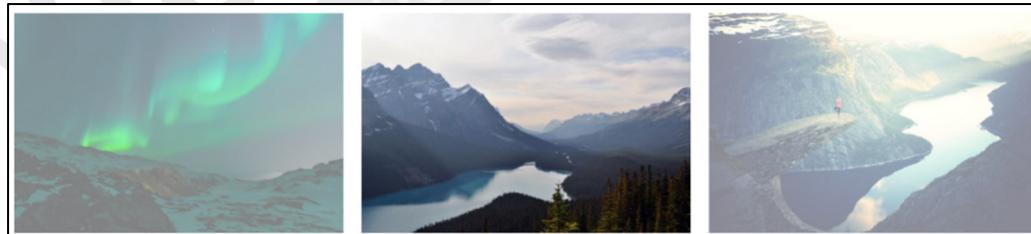
- '**opacity**' (CSS3) Ajusta el nivel de opacidad de un elemento. Los posibles valores que puede tener esta propiedad van entre 1 (opaco) y 0 (transparente). El valor también se puede introducir en porcentaje entre 100% - 0%.



Se pueden realizar efectos muy interesantes jugando con la propiedad de opacidad y la pseudo-clase :hover (al posicionarnos sobre el elemento).

```
img { opacity: 0.5; }

img:hover { opacity: 1.0; } // Posición sobre el elemento
```



La opacidad también se puede establecer mediante la propiedad 'background' utilizando el sistema RGBA (Red, Green, Blue, Alpha).

```
div { background: rgba(76, 175, 80, 0.3) }
```

## Desbordamiento en CSS

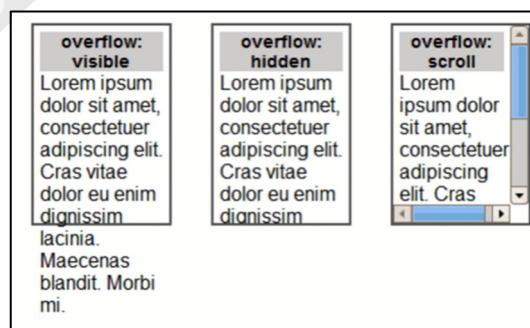
La propiedad '**Overflow**' determina el comportamiento de una caja cuando el contenido excede o desborda las dimensiones de la misma.

- 'overflow' (CSS2) Indica **que sucede si el contenido desborda la caja** de un elemento.

Solo funciona con elementos de bloque de altura específica. Los posibles valores son:

```
overflow: visible|hidden|scroll|auto|initial|inherit;
```

- '*visible*' – Muestra el contenido de la caja, aunque exceda del tamaño de la misma.
- '*hidden*' – Si el contenido sobresale de la caja se recorta mostrando únicamente el texto que cabe dentro de la caja.
- '*scroll*' – El desbordamiento se recorta, pero se añade una barra de desplazamiento para poder visualizar el contenido que ha sobresalido.
- '*auto*' – Solo añade barras de desplazamiento (H y V) cuando sea necesario.
- '*initial*' – Establece la propiedad del valor predeterminado. NO confundir con *inherit*, este es el valor que trae la propiedad por defecto. Ej: color letra es negro.
- '*inherit*' – Heredado. Si el elemento padre tiene asignada la propiedad asignada el elemento hijo adoptará el mismo valor.



- '**overflow-x**' y '**overflow-y**' (CSS3) Especifica que sucede sobre el eje de las X o el eje de las Y si se desborda el área de contenido de un elemento. Funciona igual que 'overflow' y tiene exactamente las mismas propiedades, pero solo actúa sobre el eje especificado.

## Media Queries ( [<link>](#), @mediaqueries )

Las páginas web responsive se caracterizan por adaptar su diseño de forma automática en función del dispositivo de visualización (table, pc, móvil) y su orientación (horizontal, vertical). En la fase de planificación de cualquier proyecto es imprescindible determinar que dispositivo de visualización (viewport) tendrá prioridad (Pc, Tablet, Móvil).

Este tipo de diseño es necesario tenerlo presente a la hora de diseñar una página web ya que sino se debería crear un diseño para cada tipo de dispositivo incluyendo sus posibles resoluciones, y eso es imposible por la cantidad de opciones existentes hoy en dia en el mercado.

Las webs con diseño responsive:

- Mejoran la **usabilidad y facilidad** de lectura para los diferentes dispositivos.
- No es necesario construir versiones diferentes de una misma página. El diseño 'Responsive' se adapta a los diferentes dispositivos y navegadores con una sola versión. El ahorro en la fase de diseño es considerable.
- Se **adapta automáticamente** a nuevos anchos y altos de pantalla, por lo que no importa cómo cambian las tendencias de consumo a nivel de pantallas, tu web se adaptará a ellas.
- Optimiza los anuncios mejorando la experiencia de usuario.

Por otro lado, Google premia a todas las webs que tienen un diseño adaptable premiándolas con un mejor posicionamiento en el ranking de búsquedas. Las páginas

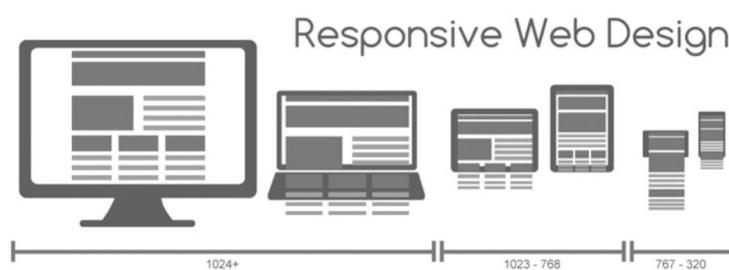
Actualmente el 50% de las webs **NO están adaptadas** a los diferentes tipos de dispositivos, y teniendo en cuenta que más del 95% de los usuarios acceden a Internet a través de móviles es imprescindible generar un contenido adaptado a ellos.

Existen multitud de empresas que dependiendo del producto o servicio que ofrezcan primero realizan la aplicación para móviles y posteriormente es adaptada a pc's. Esta tendencia se denomina '**mobile first**' y es el modelo de negocio lo que determina cual será el dispositivo principal de visualización.

Estas son las resoluciones de pantalla más comunes en todo el mundo para que adaptes tu web a medidas responsive:

- 360×640 (móvil pequeño): 22,64%.
- 1366×768 (ordenador portátil medio): 11,98%.
- 1920×1080 (escritorio grande): 7,35%.
- 375×667 (móvil medio): 5%.
- 1440×900 (escritorio medio): 3,17%.
- 720×1280 (móvil grande): 2,74%.

### Medidas Web Responsive



Las Media Querys son **consultas sobre las características del medio** donde se está visualizando una web. Nos sirven para definir estilos condicionales que solo se aplicarán en caso de que la consulta efectuada sea positiva. Dicho de otra forma, es posible asignar un estilo u otro a partir de unas condiciones (tipo dispositivo, ancho mínimo de pantalla, orientación, etc...).

Si deseamos tener una web "Responsive" para dispositivos móviles hay que tener presente las posibles resoluciones simuladas por los diferentes dispositivos como Smartphones y Tablets. En este caso solucionamos el problema introduciendo la siguiente etiqueta meta en el **<head>**:

```
<meta name="viewport" content="width=device-width, initial-scale= 1",
maximum-scale= 1"/>
```

Las 3 formas más frecuentes de adaptar el diseño web son:

- **<link>**: Se utiliza para importar y enlazar con un archivo CSS externo. Se introduce entre las etiquetas **<head>**. Se debe utilizar conjuntamente con el atributo '**media**' para indicar el medio o medios en los que se aplican los estilos de cada archivo. La sintaxis y varios ejemplos se comentan a continuación:

Sintaxis: `<link rel="stylesheet" media="mediatype and|not|only expressions)" href="print.css">`

// Se carga el estilo para pantallas independientemente del tamaño.

```
<link rel="stylesheet" media="screen" href="estilo_1.css" />
```

// Se carga el estilo si es para imprimir

```
<link rel="stylesheet" media="print" href="estilo_2.css"/>
```

// Se carga el estilo a partir de ancho mínimo o superior a 800px

```
<link rel="stylesheet" media="(min-width:800px)" href="estilo_3.css"/>
```

// Se carga el estilo si la resolución es igual o menor que 480px

```
<link rel="stylesheet" media="screen and (max-device-width: 480px)" href="estilo_3.css"/>
```

El problema de realizar las Media Querries a través de la etiqueta **<link>** radica en que disponemos de varios archivos .css. Inicialmente puede ser bueno para la administración de los estilos porque se encuentran todos ellos separados, pero en la práctica no es óptimo porque se deben realizar varias consultas al servidor para importar todos los archivos .css provocando que la carga de la página se ralentice.

- **@import:** Esta forma de importar estilos se realizará siempre en la zona superior del archivo css, justo a continuación de la etiqueta `@charset`. Esto es necesario que sea así ya que el archivo del estilo se cargará antes de ejecutar el código que contiene.

Los siguientes ejemplos muestran en qué momento o bajo qué criterios serán cargados las hojas de estilos correspondientes. Los tres primeros ejemplos realizan una consulta a medios para determinar si se debe cargar el estilo mientras que el último ejemplo se descargará cuando la página vaya a ser impresa.

```
@import "miestilo.css" screen and (orientation: landscape);

@import "miestilo.css" screen and (max-width: 768px);

@import "miestilo.css" (min-device-width: 320px);

@import "miestilo.css" print; // Carga estilo al imprimir página
```

- **@media:** (CSS3) Es un tipo de etiqueta en CSS que permite definir diferentes estilos para diferentes medios o dispositivos: pantallas, impresoras, móviles, proyectores, etc. Es el sistema más utilizado por desarrolladores.

Desde CSS3 se llama 'consultas a medios' (media queries), es decir, se pregunta por el tipo de dispositivo que está accediendo a la información y a partir de ahí se carga el archivo CSS que corresponda. Hace años había multitud de dispositivos de salida, pero en la actualidad se han reducido a cuatro.

Medio	Descripción
<b>all</b>	Utilizado para todo tipo de dispositivos
<b>print</b>	Utilizado para impresoras.
<b>screen</b>	Utilizado para monitores, tablets y smartphones.
<b>speech</b>	Sintetizadores de voz utilizados por personas discapacitadas.

Las reglas **@media** son un tipo especial de regla CSS que permiten indicar de forma directa el medio o medios en los que se aplicarán los estilos incluidos en la regla. Para especificar el medio en el que se aplican los estilos, se incluye su nombre después de **@media**. Si los estilos se aplican a varios medios, se incluyen los nombres de todos los medios separados por comas. A continuación se muestran algunos ejemplos de declaración mediante esta variable.

La etiqueta **@media** realiza las consultas sobre:

1. Ancho y alto de la ventana gráfica.
2. Ancho y alto del dispositivo.
3. Orientación del dispositivo (horizontal o vertical).
4. Resolución del dispositivo.

Sintaxis: `@media not|only mediatype and (media características) {`

*CSS-Code;  
}*

Los operadores permiten realizar combinaciones de reglas para establecer un determinado criterio con varias condiciones:

- Operador **and**: Las dos o más condiciones deben cumplirse para que se evalúe como verdadero el estilo que se debe aplicar.
- Operador **not**: Es una negación de una condición. Cuando esa condición no se cumpla se aplicarán las 'Media Queries'.
- Operador **only**: Se aplican las reglas solo en el caso que se cumpla cierta circunstancia.
- Operador **or**: No existe como tal, pero puedes poner varias condiciones separadas por comas y cuando se cumpla cualquiera de ellas, se aplicarán los estilos de las 'Media Queries'.

Ejemplo 1:

```
// Este ejemplo especifica el interlineado para la página web
siempre que sean solo dispositivos tipo 'screen' (pc's) con un
ancho mínimo a 320px y un máximo a 480px. En este caso se deben
cumplir las tres condiciones.
```

```
@media (min-device-width: 320px) and (max-device-width: 480px){

    body {

        line-height: 1.4;

        background-color: lightblue;
    }
}
```

Ejemplo 2:

```
// El tamaño de letra de la página por defecto cuando se visualiza
en una pantalla debe ser 13px. Sin embargo, cuando se imprimen los
contenidos de la página, su tamaño de letra debe ser de 10px */


```

```
@media print { body { font-size: 10px } }

@media screen { body { font-size: 13px } }
```

Ejemplo 3:

```
// Se deben cumplir las dos condiciones para poder aplicar el
estilo (ancho>600 y orientación vertical).
```

```
@media (min-width: 600px) and (orientation: portrait) {

    h1 { color: green }

}
```

Ejemplo 4:

```
// Se debe cumplir una de las 2 condiciones, pero la segunda  
condición está compuesta por 2 condiciones  
  
@media (min-width: 600px) handheld and (orientation: portrait) {  
  
    h1 { color: green }  
  
}
```

Ejemplo 5: Código que contiene los estilos que se cargan por defecto en la página web y los otros estilos que se cargarán a través de @mediaqueries. El primero se utilizará cuando la pantalla tenga un ancho superior a 800px y inferior a 1200px y el más pequeño cuando el ancho de pantalla sea superior a 420px y inferior a 800px.

```
<head>  
  
    <style type="text/css"> // Estilo por defecto  
  
        body{ background-color:white; font-size:14px; }  
  
        #bloque1{ font-size:15px; }  
  
        @media only screen and (min-device-width: 800px) and (max-  
device-width: 1200px){  
  
            body{ background-color:red; }  
  
            #bloque1{ font-size:12px; }  
        }  
  
        @media only screen and (min-device-width: 420px) and (max-  
device-width: 800px){  
  
            body{ background-color:red; }  
  
            #bloque1{ font-size:12px; }  
        }  
  
    </style>  
  
</head>
```

## Archivo Reset CSS

Las etiquetas en HTML disponen de unas **propiedades asignadas por defecto**. Por ejemplo, los elementos de bloque ocupan todo el ancho de la página, la etiqueta `<p>` hace un salto de línea de forma automática, los encabezados (`<h1>...<h6>`) cambian el tamaño de la letra, y la etiqueta `<a>` asigna un color dependiendo si el link ha sido pulsado o visitado.

Estas propiedades inherentes y predefinidas son molestas para los diseñadores ya que con CSS se puede aplicar el formato con mayor facilidad. Por este motivo surgen los **archivos de reset** que "inicializan" todos los estilos de tal forma que no tienen propiedades definidas y permite que se vean de la misma forma en todos los navegadores.

Existen varios archivos de reset en Internet que **se pueden descargar de forma gratuita** e incluir en la página web desde el `<head>` que resetean todas las etiquetas de HTML dejándolas con el mismo formato para todos los navegadores como, por ejemplo:

- Normalize CSS (`normalize.css`).
- HTML5 Doctor Reset
- Eric Meyer's reset.
- Yahoo Reset CSS.
- Universal Selector Reset.

Para utilizar un archivo de reset es necesario incluir el enlace de la siguiente forma:

```
<link href="normalize.min.css" rel="stylesheet">
```

A continuación, se muestra un fragmento de lo que podría ser un archivo de Reset.

```
* { margin: 0;  
    padding: 0;  
    border: 0;  
    font-size: 100%;  
    font: inherit;  
    vertical-align: baseline; }
```

## Barras de navegación en CSS

Una barra de navegación es una lista de enlaces con estilos CSS aplicados que le dan un formato atractivo dentro de la web. Las barras de navegación comienzan siendo enlaces que se modifican y se van desarrollando mediante estilos CSS.

Las barras de navegación pueden crearse de forma horizontal y/o vertical. Para no generar una barra de navegación desde cero, es muy frecuente utilizar barras con todas las opciones creadas y personalizarlas para nuestra página.

### Esquema Barra Vertical:

- Creación de la lista desplegable en formato vertical

Inicio

Noticias

Contacto

Sobre

```
<ul>
    <li><a href="#home">Inicio</a></li>
    <li><a href="#news">Noticias</a></li>
    <li><a href="#contact">Contacto</a></li>
    <li><a href="#about">Sobre</a></li>
</ul>
```

- Configuración de los estilos de la lista



```
ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
    width: 200px;
    background-color: #f1f1f1;
}

li a {
    display: block;
    color: #000;
    padding: 8px 16px;
    text-decoration: none;
}

/* Change the link color on hover */
li a:hover {
    background-color: #555;
    color: white;
}
```

## Menús desplegables en CSS

Los menús desplegables se utilizan para ampliar las opciones de la barra de navegación. A continuación, se muestra el código fuente de un menú desplegable mediante la etiqueta **<button>**.

```

<style>
/* Estilo del botón del menú desplegable */
.dropbtn { background-color: #4CAF50; color: white; padding: 16px;
            font-size: 16px; border: none; cursor: pointer; }

/* Posicion de la caja <div> que contiene el menú */
.dropdown { position: relative; display: inline-block; }

/* Configuración y ocultación de las opciones de los links del menu */
.dropdown-content { display: none; position: absolute;
                        background-color: #f9f9f9; min-width: 160px;
                        box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);
                        z-index: 1; }

/* Configuración links del menu */
.dropdown-content a { color: black; padding: 12px 16px;
                        text-decoration: none; display: block; }

/* Cambio color al pasar sobre los links */
.dropdown-content a:hover {background-color: #f1f1f1}

/* Muestra el submenú al posicionarnos sobre el botón principal*/
.dropdown:hover .dropdown-content { display: block; }

/* Cambia el color del botón al ponernos sobre el botón del menú. */
.dropdown:hover .dropbtn {background-color: #3e8e41; }
</style>

```

```

<div class="dropdown">
    <button class="dropbtn">
        Menú desplegable
    </button>
    <div class="dropdown-content">
        <a href="#">Link 1</a>
        <a href="#">Link 2</a>
        <a href="#">Link 3</a>
    </div>
</div>

```

Menú desplegable

Link 1

Link 2

Link 3

## Herramientas y utilidades en CSS

Los efectos más interesantes a nivel de diseño se realizan combinando las diferentes propiedades que ofrece CSS con las etiquetas HTML. A continuación, se muestran una serie de elementos creados con CSS que pueden ser de gran utilidad.

- Creación de tooltips: Un tooltip es un cuadro de texto de información que se muestra cuando se posiciona el cursor sobre algún elemento de la página.

```
<style>
/* Caja principal del tooltip */
.tooltip {
    position: relative;
    display: inline-block;
    border-bottom: 1px dotted black;}

/* Configuración del texto del tooltip */
.tooltip .tooltiptext {
    visibility: hidden;
    width: 120px;
    background-color: black;
    color: #fff;
    text-align: center;
    padding: 5px 0;
    border-radius: 6px;
    position: absolute;
    z-index: 1;}

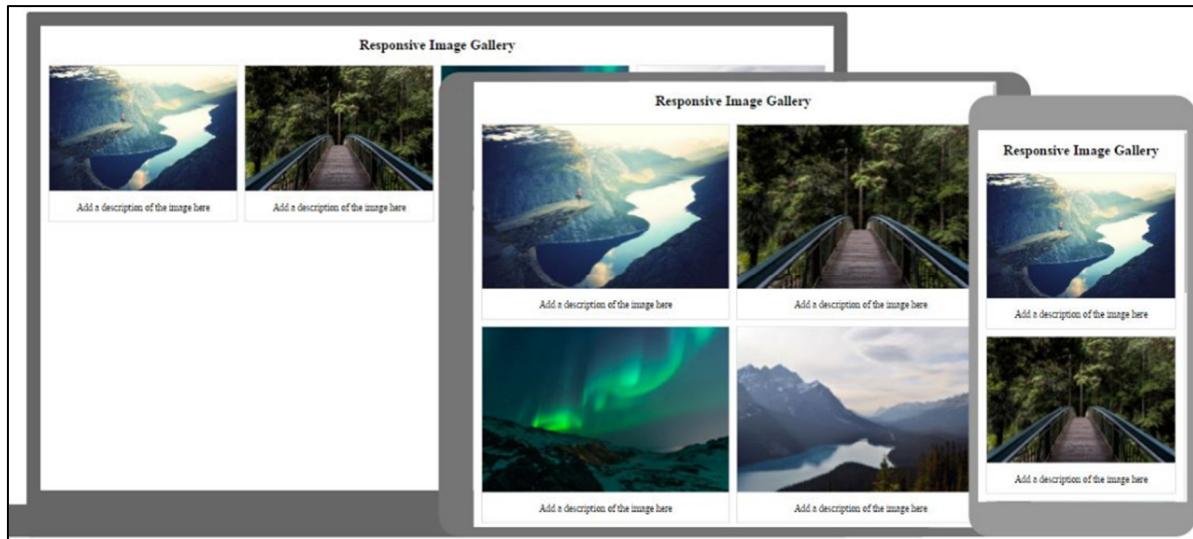
/* Muestra el tooltip al posicionarnos sobre el elemento*/
.tooltip:hover .tooltiptext {
    visibility: visible;}
</style>

<div class="tooltip"> Pasar por encima
    <span class="tooltiptext">Cuadro de texto informativo</span>
</div>
```

Pasar por encima

Cuadro de  
texto  
informativo

- Galerías de imágenes: Un aspecto muy visual es mostrar las imágenes formando una galería. El efecto visual que permite ir apilando imágenes a un lado u otro de la página se realiza utilizando el atributo '*float*'. Utilizando este sistema las imágenes se van posicionando en función del tamaño del dispositivo de visualización.

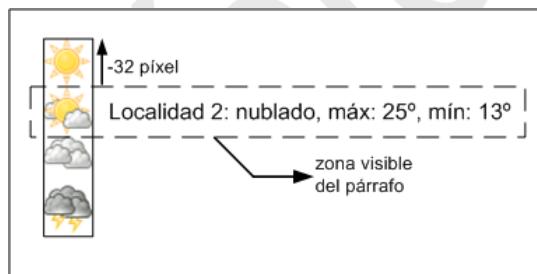


- **Sprites:** Los sprites nos permiten ahorrar tiempo de descarga de imágenes. Cuando se realiza la carga de una web se hacen **tantas peticiones al servidor como imágenes contiene**. Utilizando los sprites solo llamaremos a una imagen que contendrá todas (como un collage de imágenes) y luego mediante el posicionamiento de estas solo seleccionaremos la zona de la imagen que nos convenga en cada caso.

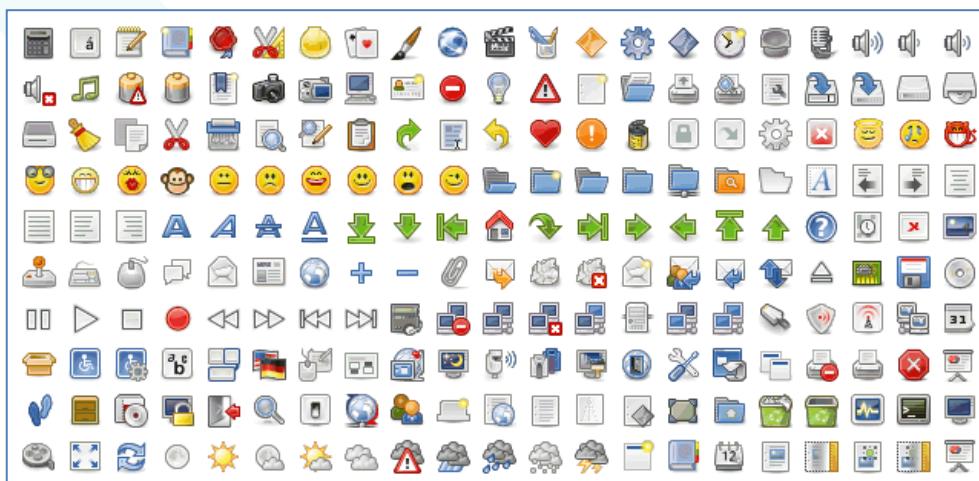
```
#localidad1, #localidad2, #localidad3, #localidad4 {
    padding-left: 38px;
    height: 32px;
    line-height: 32px;
    background-image: url("imagenes/sprite.png");
    background-repeat: no-repeat;
}

#localidad1 { background-position: 0 0; }
#localidad2 { background-position: 0 -32px; }
#localidad3 { background-position: 0 -64px; }
#localidad4 { background-position: 0 -96px; }
```

La siguiente imagen muestra lo que sucede con el segundo párrafo:



A continuación, se muestra una imagen (sprite) formada por muchas pequeñas imágenes y así en cada caso se selecciona la que interese.



## Personalización del cursor

La propiedad '**cursor**' no sólo permite seleccionar un puntero entre los disponibles en el sistema operativo (flecha, mano, reloj de arena, redimensionar, etc.) sino que incluso permite indicar la URL de una imagen que se quiere mostrar como puntero personalizado.

```
:visited { cursor: url(puntero.svg), url(puntero.cur), pointer }
```

Si el navegador soporta las imágenes en formato con extensión **SVG**, el puntero del ratón cambia su aspecto por la imagen '*puntero.svg*'. Si el navegador no soporta el formato SVG, intenta cargar la siguiente URL que define un puntero en formato '**.cur**'. Si no se puede cargar correctamente, se mostraría el valor preestablecido **pointer**.

Puntero	
 <b>cursor: default</b>	 cursor: s-resize
 <b>cursor: crosshair</b>	 cursor: sw-resize
 <b>cursor: hand</b>	 cursor: w-resize
 <b>cursor: pointer</b>	 cursor: nw-resize
 <b>cursor:pointer; cursor: hand</b>	 cursor: progress
 <b>cursor: move</b>	 cursor: not-allowed
 <b>cursor: text</b>	 cursor: no-drop
 <b>cursor: wait</b>	 cursor: vertical-text
 <b>cursor: help</b>	 cursor: all-scroll
 <b>cursor: n-resize</b>	 cursor: col-resize
 <b>cursor: ne-resize</b>	 cursor: row-resize
 <b>cursor: e-resize</b>	 cursor: url(...)
 <b>cursor: se-resize</b>	

Código de ejemplo para diferentes tipos de puntero en diferentes párrafos:

```
.c1 { cursor: pointer; }

.c2 { cursor: url(smiley.gif), url(myBall.cur), auto; }

.c3 { cursor: move; }

.c4 { cursor: no-drop; }

.c5 { cursor: wait; }

.c6 { cursor: help; }

.c7 { cursor: help; }

<p class="c1"> Mano </p>

<p class="c2"> Imagen externa de un smiley</p>

<p class="c3"> Puntero de desplazamiento </p>

<p class="c4"> Prohibido </p>

<p class="c5"> Circunferencia dando vueltas en espera </p>

<p class="c6"> Flecha del mouse con signo de interrogación </p>

<p class="c7"> Una cruz en color negro </p>
```

Property					
cursor	5.0	5.5	4.0	5.0	9.6

Una dirección donde podemos visualizar un ejemplo de cada uno de estos cursores es:

[https://www.w3schools.com/cssref/pr\\_class\\_cursor.asp](https://www.w3schools.com/cssref/pr_class_cursor.asp)

## Flexbox (cajas flexibles)

Es una nueva forma de definir el diseño de cajas en una página web. Los elementos flexibles (cajas básicamente) se adaptan y se redimensionan en función del tamaño de la pantalla. Este modelo funciona ajustando los tamaños y la disposición de los elementos que se encuentran dentro de un contenedor o caja, de tal manera que se adapten siempre al espacio disponible. Permite posicionar dichos elementos internos con gran facilidad, de manera independiente al orden en el que aparezcan en el código.

El sistema Flex puede ser utilizado en combinación con otros sistemas de posicionamiento de cajas. Por ejemplo, es posible disponer de un posicionamiento de los elementos más significativos de la página (main o section) a través de un sistema '**grid**' (rejilla) mientras que en su interior los elementos se distribuyan utilizando en modo flexible.

La declaración de un contenedor con la propiedad '**display: flex**' convierte el contenedor como flexible y de forma automática afecta a todos sus hijos directos. Dicho de otra forma, la configuración de las cajas flexibles se debe realizar sobre el **elemento padre** y no sobre los propios hijos.



### Ejemplo:

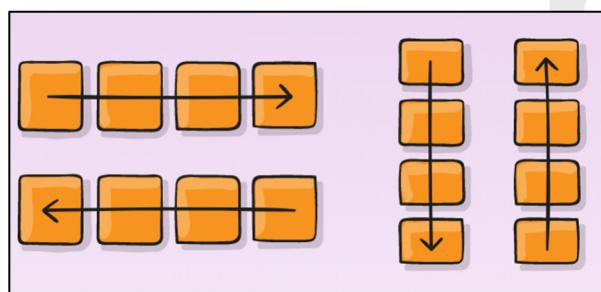
Disponemos de una caja contenedora declarada como caja flexible con el siguiente código. La declaración se realiza sobre la caja madre para que afecte a los hijos directos.

```
#viajes { display: flex; }
```

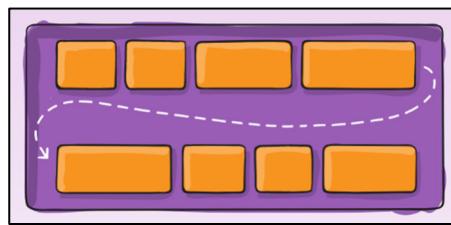
A continuación, hay que especificar de qué forma se distribuirán los elementos hijos **dentro de la caja flexible**. Para ello se utiliza la propiedad abreviada '**flex-flow**' (flujo de las cajas) donde se le especifican los valores '**flex-direction**' y '**flex-wrap**' como en cualquier propiedad CSS.

- '**flex-direction**

- '**row**' se distribuyen en filas.
- '**row-reverse**' en filas de forma inversa (derecha a izquierda).
- '**column**' se distribuyen en columnas.
- '**column-reverse**' en columnas de forma inversa (de abajo hacia arriba).



- '**flex-wrapson forzados en una sola línea** o pueden ser encapsulados en varias líneas. Si se permite el ajuste, esta propiedad también le permite controlar la dirección en la que se apilan las líneas.
  - '**nowrapen una sola línea**, lo que puede hacer que el contenedor flexible rebose o que los ítems interiores se redimensionen para ajustarse a una sola línea. Si se produjera esta situación la propiedad 'width' se omite prevaleciendo el ajuste de las cajas dentro de su contenedor.
  - '**wrappuedan ocupar varias líneas**, es decir, si un conjunto de ítems no cabe en una sola línea se produce un ajuste del contenido ocupando varias líneas.
  - '**wrap-reversevarias líneas en orden inverso**, es decir, empezando por la zona inferior de la caja contendora. Esta propiedad es necesaria cuando alineamos elementos de forma "poco frecuente" (en filas de derecha a izquierda, y en columnas de abajo a arriba) que los elementos queremos que se alineen de forma lógica siguiendo el flujo más adecuado.



El siguiente código muestra dos formas para determinar el flujo de las cajas, la primera mediante dos órdenes, y la segunda utilizando el método abreviado.

```
#viajes {  
    display: flex;  
    flex-direction: row;  
    flex-wrap: wrap;}
```

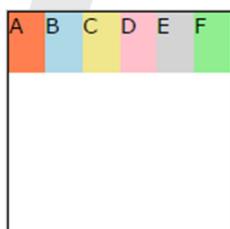
Código especificado en dos líneas.

```
#viajes { display: flex;  
    flex-flow: row wrap; }
```

Método abreviado.

El siguiente cuadro muestra todas las combinaciones posibles para la propiedad 'flex-flow' que es una forma abreviada de combinar las propiedades 'flex-direction' y 'flex-wrap' en una sola propiedad.

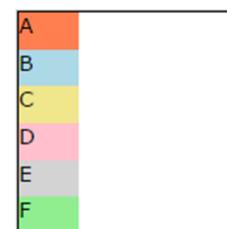
Row nowrap



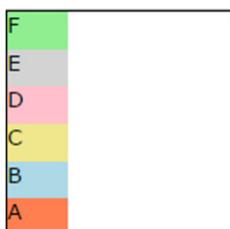
Row-reverse nowrap



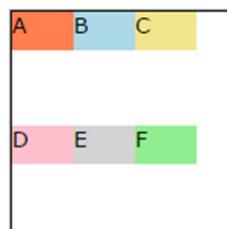
Column nowrap



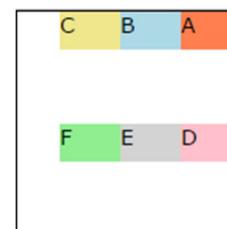
Column-reverse nowrap



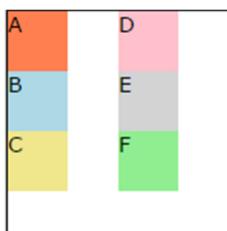
Row wrap



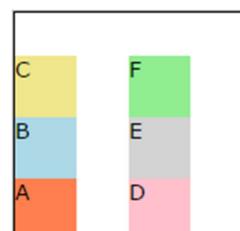
Row-reverse wrap



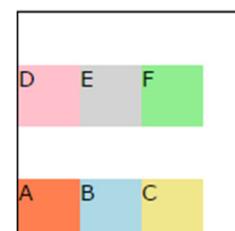
Column wrap



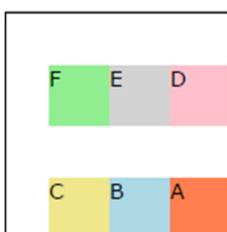
Column-reverse wrap



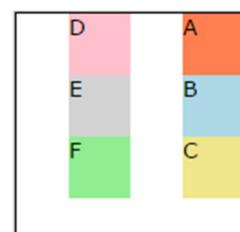
Row wrap-reverse



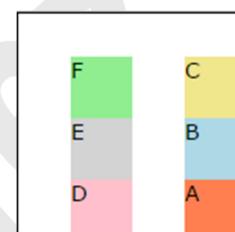
Row-reverse wrap-reverse



column wrap-reverse



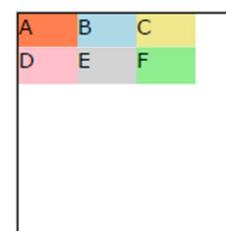
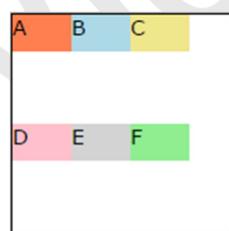
Column-reverse wrap-reverse



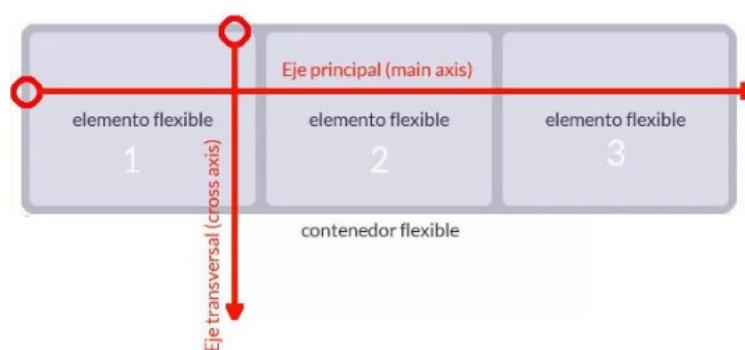
- '**align-content**': Esta propiedad nos permite **agrupar las cajas hijas** dentro de su parente. Los valores aceptados son: '*flex-end*', '*flex-start*', '*center*', '*space-around*', '*space-between*', '*center*'.

Caja 'row wrap' con distribución de los elementos dentro elemento padre.

Caja 'row wrap' con distribución de los elementos con 'align-content:flex-start'.

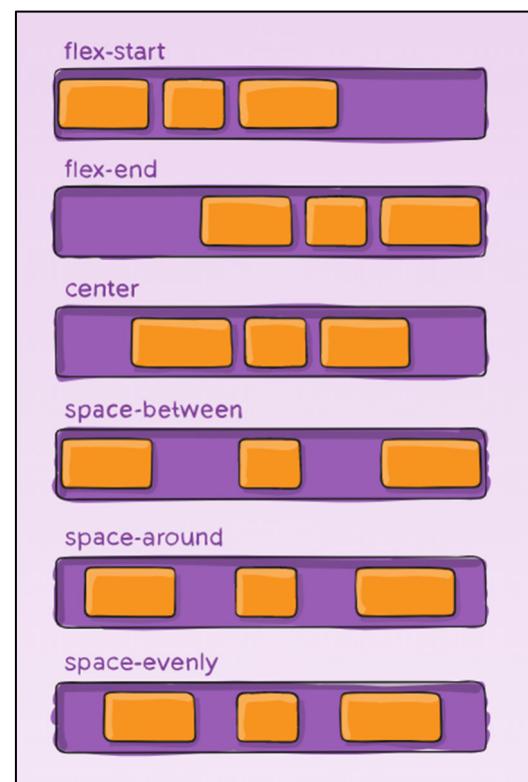
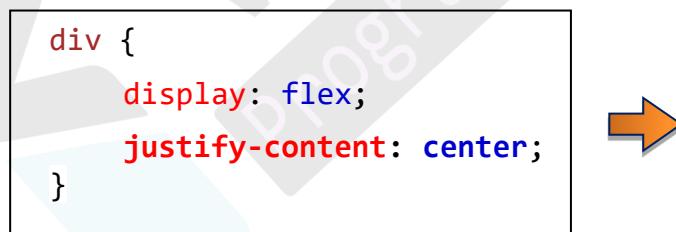


Para alinear los elementos de las cajas se debe tener en cuenta si se desea realizar sobre el eje principal (horizontal) o sobre el eje secundario (vertical), ya que las propiedades de alineación varían. Los ejes y sus diferentes valores se muestran a continuación:



Sobre el Eje principal (HORIZONTAL como normal general) la propiedad '**justify-content**' nos permite distribuir los ítems interiores y los posibles valores son:

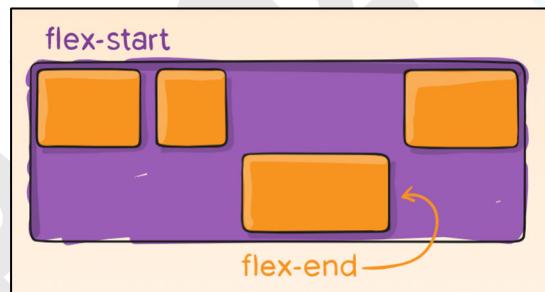
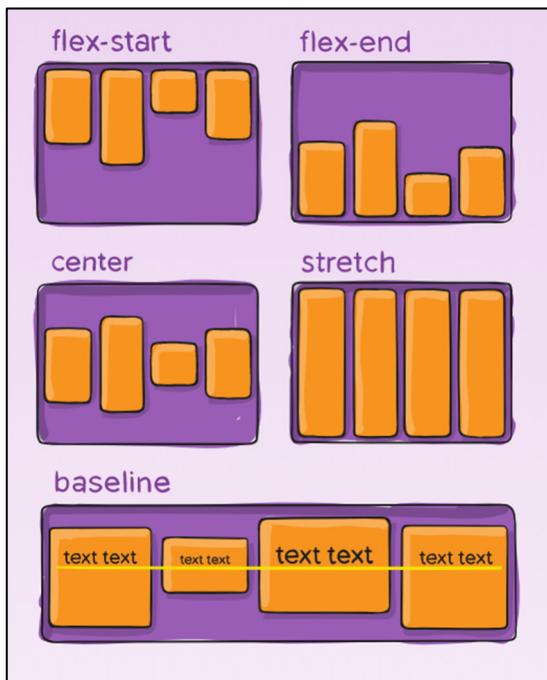
- '**flex-start
- '**flex-end
- '**center
- '**space-betweensin dejar espacio al inicio y al final.**
- '**space-aroundel espacio inicio y fin que son diferentes al resto.**
- '**space-evenly********



En el Eje secundario (**VERTICAL** como normal general) las propiedades para alinear son '**align-items**' (afecta todos los elementos del contenedor), o '**align-self**' (solo al elemento individual seleccionado). La propiedad '**align-self**' se debe especificar en el estilo del propio del hijo ya que solo afecta a un item dentro de todos los existentes dentro del contenedor.

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

```
div {
    display: flex;
    align-self: flex-end; }
```



La propiedad '**justify-content**' trabaja sobre el eje primario que puede ser '*row*' (filas) o '*column*' (columnas). Si trabaja sobre '*row*' esta propiedad realizará la alineación horizontal de los elementos, mientras que si trabaja sobre '*column*' lo hará de forma vertical. Este es un aspecto muy importante a tener en cuenta porque no siempre '**justify-content**' actúa sobre la horizontal.

Dicho esto, si la propiedad '**justify-content**' en determinados casos actúa sobre la verticalidad de los elementos, la propiedad '**align-items**' hará todo lo contrario al producirse un cambio de eje primario y secundario.

**NOTA 1:** El atributo '**display**' con el valor '**inline-flex**' realiza que una caja madre tenga un comportamiento del tipo '**inline-block**' y sus hijas directas se comporten como elementos flexibles. Es decir, es una mezcla entre '**inline-block**' y '**relative**'. Como no es posible introducir dos atributos '**display**' dentro del mismo estilo con diferente valor, el tipo '**inline-flex**' soluciona este problema.

**NOTA 2:** Para definir elementos flexibles se debe utilizar el valor '**flex**' en el atributo '**display**'. Los valores que se utilizaban antiguamente '**box**' (2009) y '**flexbox**' (2011) son valores que el nuevo estándar NO recoge como válidos. Algunos navegadores y editores aun lo aceptan pero no es adecuado su uso.

Un caso típico es modificar la colocación de las cajas dependiendo del tamaño de la pantalla de visualización. Para ello es necesario realizar una consulta a los '**medios**' (mediaqueries) y a partir de ahí asignar a la propiedad '**flex-direction**' para distribuir las cajas en columnas o filas.

```
// La primera línea pregunta por el ancho del dispositivo y si cumple el criterio se aplica el estilo, en este caso la dirección de las cajas pasa a distribuirse en columnas.
```

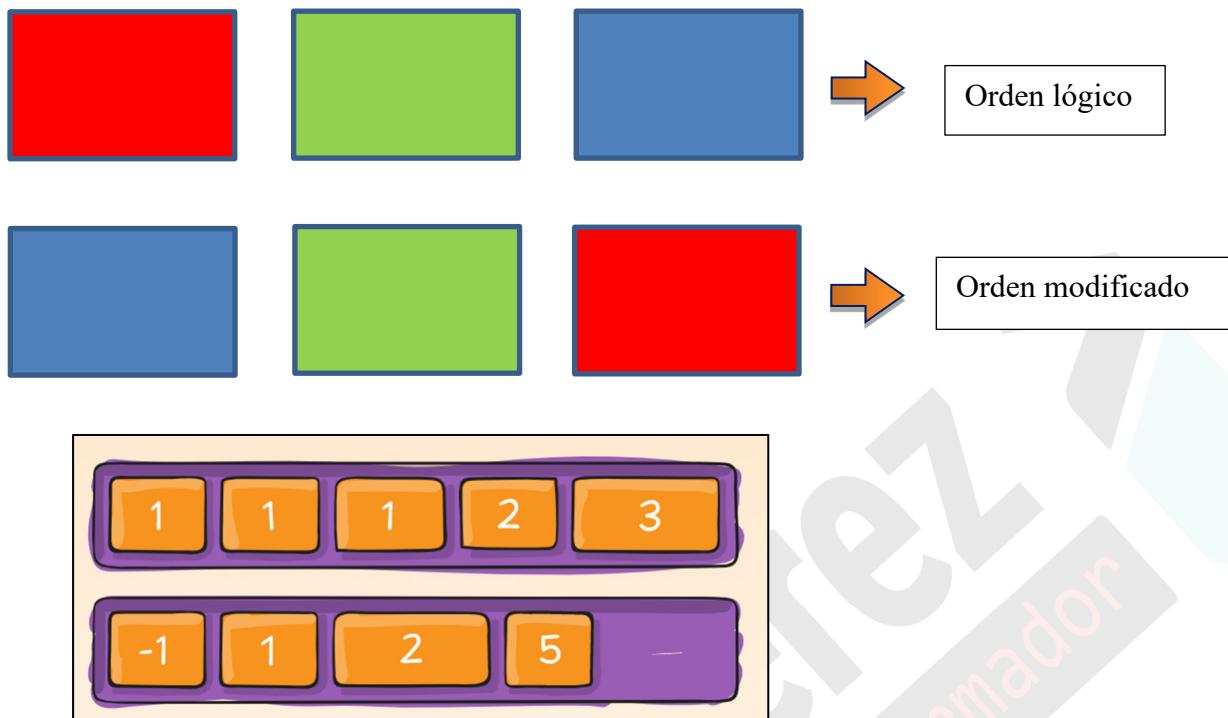
```
@media all and (max-width: 420px) {  
    #viajes { flex-direction: column; }  
}
```

La propiedad '**order**' establece el orden en el que aparecen los componentes de una caja flexible. Por defecto se muestran como aparecen en el código HTML (equivalente '**order:0;**'). Los valores negativos anteponen las cajas a elementos con valores positivos.

La ordenación de cajas flexibles se realiza con la propiedad '**order**' sobre cada una de las cajas flexiles. Es decir esta propiedad NO se debe especificar en la caja madre como la mayoría de las opciones flexbox, sino que se configura en las hijas.

Siguiendo el ejemplo anterior si nuestra caja principal tuviese en su interior 3 cajas el orden lógico de estas se podría modificar de la siguiente forma.

```
.uno { background-color: red; order: 3; }  
.dos { background-color: green; order: 2; }  
.tres { background-color: blue; order: 1; }
```



La propiedad '**flex**' determina como **crece o decrece** un elemento flexible dentro del contenedor en relación con los demás. Por defecto el valor de todos los elementos de un contenedor tiene valor 1, pero este puede ser modificable para que la distribución de espacio sea diferente. En el siguiente código la clase '.dos' tiene un tamaño doble al resto de elementos.

```
.uno { background-color: red; flex: 1; }
.dos { background-color: green; flex: 2; }
.tres { background-color: blue; flex: 1; }
```

La caja 2 será del doble de tamaño que el resto.

La propiedad '**flex**' en formato abreviado queda definida de la siguiente forma:

```
flex: flex-grow, flex-shrink, flex-basis ;
```

Otra forma de determinar los factores de crecimiento y decrecimiento es mediante las propiedades:

- '**flex-growfactor de crecimiento**, es decir, cuanto crecerá el elemento en relación a los demás cuando haya espacio disponible en el contenedor. Por defecto es cero.

- 'flex-shrink': Determina el **factor de reducción**, es decir, cuanto decrecerá el elemento cuando hay espacio negativo en el contenedor. Por defecto es cero.

**NOTA:** Las propiedades 'flex-grow' y 'flex-shrink' siempre realizan la distribución del tamaño de las cajas (crecer o reducir) a partir del **espacio sobrante en la caja madre**, es decir, las cajas por defecto ocupan lo que les corresponda y en función del espacio sobrante en la caja madre se distribuye para dimensionar a las cajas hijas.

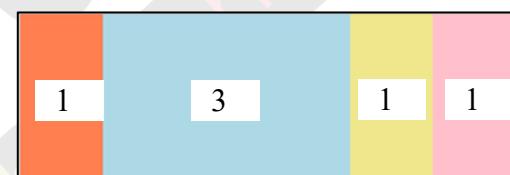
- 'flex-basis': Toma el mismo valor que la propiedad 'width' y establece el tamaño inicial para la caja seleccionada antes de distribuir el espacio libre de acuerdo con las ratios de 'flex-grow' o 'flex-shrink'. Esta opción se utiliza para especificar anchos específicos a determinadas cajas hijas pero si forzar el tamaño y la ubicación del resto de los ítems.

```
#caja1 { flex-grow: 1; }
```

```
#caja2 { flex-grow: 3; }
```

```
#caja3 { flex-grow: 1; }
```

```
#caja4 { flex-grow: 1; }
```

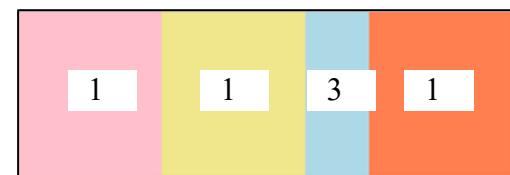


```
#caja1 { flex-shrink: 1; }
```

```
#caja2 { flex-shrink: 1; }
```

```
#caja3 { flex-shrink: 3; }
```

```
#caja4 { flex-shrink: 1; }
```



## Efecto Parallax

Parallax es un diferente sistema para diseñar webs que produce un efecto muy característico cuando nos desplazamos por la página. Al movernos utilizando el scroll el fondo de la página se mueve a una velocidad distinta del contenido. Se utiliza sobre todo para landing pages donde la parte de presentación es mucho más atractiva utilizando este sistema.

Parallax crea un ligero **efecto de profundidad** al mover el fondo de la página a una velocidad distinta al contenido. Este diseño dota de dinamismo a la página y contrarresta la planitud con fondos estáticos que no tienen ningún tipo de movimiento.

El origen de Parallax fue en los estudios de Disney para "animar" los dibujos. Posteriormente saltó al mundo de los videojuegos, y por último ha recalado en el diseño web.

Para realizar el efecto parallax será necesario utilizar HTML5 y CSS3. JavaScript dispone de una librería que facilita este trabajo y incluso detecta la orientación del dispositivo a través del giroscopio.

## CSS Grid

### Introducción a Grid

CSS Grid es un nuevo sistema que intenta **facilitar la maquetación de cajas dentro de una página web**.

Los mecanismos utilizados tradicionalmente como posicionamiento relativo, absoluto, floats, elementos en bloque o en línea son insuficientes (o *muy complejos*) para crear un layout o estructuras para determinadas páginas web actuales.

La idea de CSS Grid es **dividir una web en columnas y filas** (cuadrícula). Una de las ventajas de utilizar Grid es la posibilidad de **cambiar la posición de los elementos modificando simplemente el código CSS**, sin tener que realizar cambio alguno en HTML.

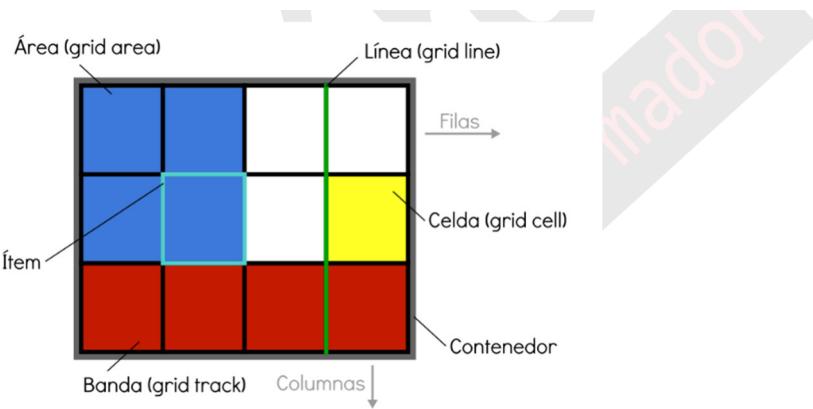
El sistema **flexbox** es una gran mejora, sin embargo, está orientado a estructuras de una sola dimensión, por lo que aún necesitamos algo más potente para estructuras web (grid tiene 2 dimensiones). Con el paso del tiempo, muchos frameworks y librerías utilizan un **sistema grid** donde definen una cuadrícula determinada, y modificando los nombres de las clases de los elementos HTML, podemos darle tamaño, posición o colocación, entre otros aspectos.

CSS Grid está activado por defecto en los navegadores mas importantes (Chrome, Mozilla, Safari, Opera) desde Marzo 2017, pero funciona de forma óptima desde Mayo del mismo año tras varias actualizaciones y errores que surgieron. Posteriormente se han realizado actualizaciones de mejora.

Antes de iniciarse con Grid es necesario dominar FlexBox ya que Grid toma la base y la filosofía de este sistema. Tal y como ocurre en Flexbox, CSS Grid funciona con la idea de un contenedor-padre que alberga unos elementos-hijo. Por lo que nuestro contenedor-padre (**grid-container**), y nuestro elemento-hijo es un (**grid-item**).

- **Container:** Es el elemento padre y es el contenedor que definirá la cuadrícula o rejilla. Es donde se introducirá el atributo (**display:grid**) para especificar que es un elemento que trabaja con grid.
- **Ítem:** Cada uno de los elementos hijos que contiene la cuadrícula (*elemento contenedor*).
- **Línea (grid line):** Separador horizontal o vertical de las celdas de la cuadrícula.

- Celda (grid cell):** Es una celda, el espacio entre dos líneas verticales adyacentes y dos líneas horizontales adyacentes. Es una «unidad» dentro de la rejilla llamada (casilla, celda, etc...) que coincide con la intersección de fila ('grid-row') y columna ('grid-column').
- Área (grid area):** Región o conjunto de celdas de la cuadrícula. El espacio entre dos líneas verticales y dos líneas horizontales, no teniendo que ser forzosamente adyacentes. En la imagen, tendríamos la grid área creada entre las column grid lines 1 y 3, y las row grid lines 1 y 3.
- Banda (grid track):** Es el espacio entre dos 'grid lines' adyacentes. Pueden formar tanto columnas como filas. En la imagen inferior vemos el grid track (color marrón) entre la tercera línea de fila y la cuarta.



Para crear un elemento Grid hay que especificar sobre el elemento contenedor la propiedad '*display*' con el valor '**grid**' o '**inline-grid**'. Este valor influye en como la cuadrícula se comportará con el contenedor exterior.

- inline-grid:** Establece una cuadrícula con ítems en línea, de forma equivalente a 'inline-block'. En este caso la celda se adapta al contenido afectando a toda la columna y tomando el espacio mínimo necesario para mostrarlo y el resto de las columnas se repartirán el restante.
- grid:** Establece una cuadrícula con ítems en bloque, de forma equivalente al 'block', es decir la rejilla (no cada uno de los ítems) toma todo el ancho de la pantalla de navegación.

Soy el 1	2	3
4	5	6
7	8	9

1	2	3
4	5	6
7	8	9

## Grid Columnas y filas

Es posible especificar un tamaño fijo de las cuadrículas con las propiedades '`grid-template-columns`' y '`grid-template-rows`', siempre desde el elemento padre. En la siguiente imagen se especifica el tamaño en píxeles de cada columna y de cada fila.

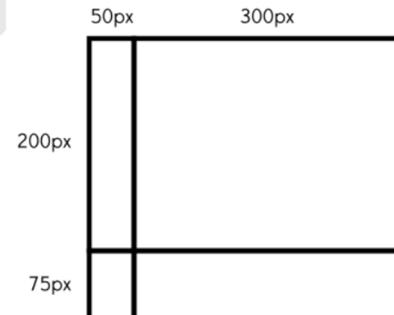
- `grid-template-columns`: Establece el tamaño de cada columna separadas por comas.
- `grid-template-rows`: Establece el tamaño de cada una de las filas separadas por comas.

Ejemplo 1: Creación de un grid de dos formas especificando el número de ítems y sus proporciones, en el primer caso en pixeles y en el segundo caso fracciones (fr).

```
<div class="grid"> <!-- contenedor -->
  <div class="a">Item 1</div> <!-- cada uno de los ítems del grid -->
  <div class="b">Item 2</div>
  <div class="c">Item 3</div>
  <div class="d">Item 4</div>
</div>
```

1

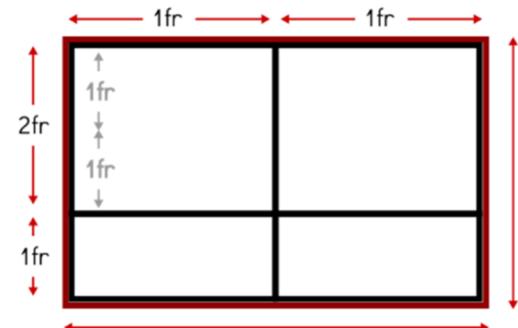
```
.grid {
  display: grid;
  grid-template-columns: 50px 300px;
  grid-template-rows: 200px 75px;
}
```



Añadir ítems

2

```
.grid {
  display: grid;
  grid-template-columns: 1fr 1fr;
  grid-template-rows: 2fr 1fr;
}
```



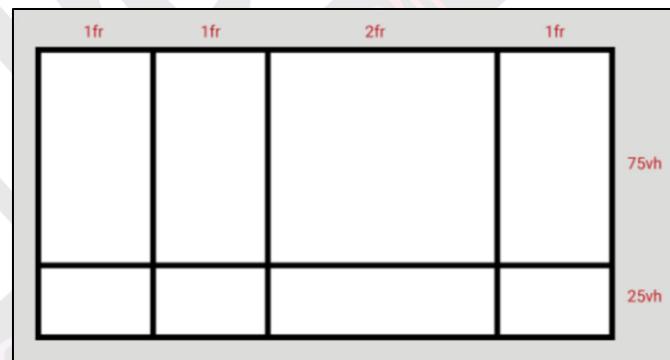
Es posible utilizar otros tipos de unidades además de los píxeles que se ajustan seguramente de forma más adecuada al espacio del contenedor como **porcentajes %** o **fracciones fr**. Incluso se puede realizar la combinación de diferentes unidades dentro de la estructura Grid.

Ejemplo 2: Creación de un grid de dos formas con 8 ítems dentro de un contenedor y especificando con diferentes unidades el tamaño de la misma.

1

```
</div class="container">
</div class="item-1"> </div>
</div class="item-2"> </div>
</div class="item-3"> </div>
</div class="item-4"> </div>
</div class="item-5"> </div>
</div class="item-6"> </div>
</div class="item-7"> </div>
</div class="item-8"> </div>
</div>
```

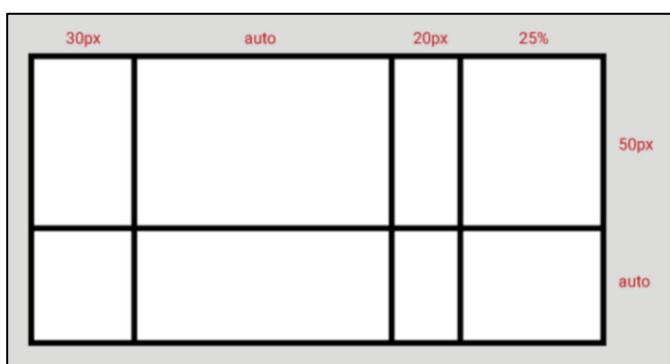
```
.container{
  display: grid;
  grid-template-columns: 1fr 1fr 2fr 1fr;
  grid-template-rows: 75vh 25vh;
}
```



2

```
</div class="container">
</div class="item-1"> </div>
</div class="item-2"> </div>
</div class="item-3"> </div>
</div class="item-4"> </div>
</div class="item-5"> </div>
</div class="item-6"> </div>
</div class="item-7"> </div>
</div class="item-8"> </div>
</div>
```

```
.container{
  display: grid;
  grid-template-columns: 30px auto 20px 25%;
  grid-template-rows: 50px auto;
}
```



Si se añaden más ítems que columnas o filas declaradas con '**grid-template-columns**' o '**grid-template-rows**' estas se configuran como si diesen la vuelta y volvieran a comenzar.

En algunas situaciones las propiedades '**grid-template-columns**' y '**grid-template-rows**' se pueden configurar de tal forma que no se tenga que repetir código innecesariamente una y otra vez. En este caso se utiliza el valor '**repeat**' y entre paréntesis se especifica el número de repeticiones y la configuración de las columnas o filas.

Las dos imágenes siguientes realizan la misma acción pero la sintaxis es diferente, una tabla de 4 X 4. En el siguiente ejemplo la configuración de las columnas será para 4 ítems. El primero de 100px, los dos siguientes por repetición de 50px, y la última de 200px.

```
.grid {  
  display: grid;  
  grid-template-columns: 100px repeat(2, 50px) 200px;  
  grid-template-rows: repeat(2, 50px 100px);  
}
```

Método simplificado

```
.grid {  
  display: grid;  
  grid-template-columns: 100px 50px 50px 200px;  
  grid-template-rows: 50px 100px 50px 100px;;  
}
```

Método normal

Cuando configuramos un grid específico de tantas filas ('**grid-template-rows**') y columnas ('**grid-template-columns**') como queramos podemos tener el inconveniente que si generamos una web dinámica se generen mas casillas que las configuradas.

En este caso para asegurarse que todas las casillas nuevas que pudieran aparecer tendrán el mismo tamaño que el resto del grid se deberá utilizar la propiedad '**grid-auto-rows**' o '**grid-auto-columns**' especificando el tamaño de las nuevas casillas de la rejilla en caso de desbordamiento respecto a su configuración inicial.

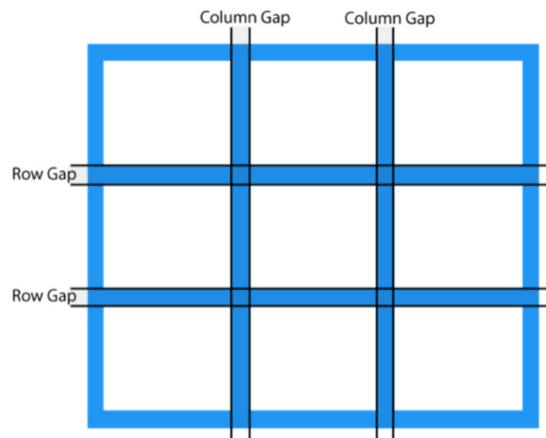
En el siguiente código se puede visualizar la configuración de la rejilla con espacio entre las casillas y la configuración de filas adicionales o sobrantes por combinaciones ('grid-auto-rows').

```
main{  
    height: 100vh; /* Altura de la caja 100% viewport*/  
  
    display:grid; /* Especifica que sus elementos hijos son grid */  
    grid-template-columns: 1fr 1fr 1fr 1fr; /* Tamaño columnas */  
    grid-template-rows: 1fr 2fr repeat(2, 1fr); /*Tamaño filas */  
  
    grid-gap: 10px; /*Shorthand espacio filas y columnas */  
    grid-auto-rows: 1fr; /*Cualquier fila adicional sera como el resto*/  
}
```

## Espaciado entre items

De inicio, el espacio entre los '**grid items**' es nulo, no existe un espacio vacío entre ellos. A veces es necesario ese espaciado entre nuestros ítems de la rejilla. Este problema lo solucionaremos con dos propiedades y una que engloba a las dos:

- '**row-gap**', Hace referencia al espaciado existente entre dos filas contiguas. (anteriormente denominado '*grid-row-gap*').
- '**column-gap**': Hace referencia al espaciado existente entre dos filas contiguas. (anteriormente denominado '*grid-column-gap*').
- '**grid-gap**': Este es el método abreviado '*shorthand*' conjunto de filas y columnas. Cuando lo usemos, hay que tener en cuenta una cosa muy importante. El primer grid gap se corresponde con la segunda grid line, que es la que realmente está entre dos elementos de nuestra rejilla.

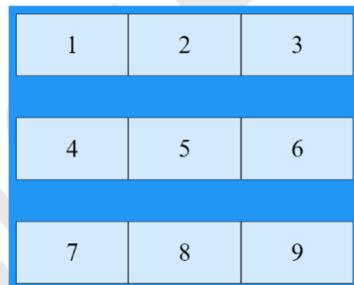


```
.grid-container {
  display: grid;
  grid-column-gap: 50px;
}

.grid-container {
  display: grid;
  grid-row-gap: 50px;
}

.grid-container {
  display: grid;
  grid-gap: 50px 100px;
}
```

1	2	3
4	5	6
7	8	9



1	2	3
4	5	6
7	8	9

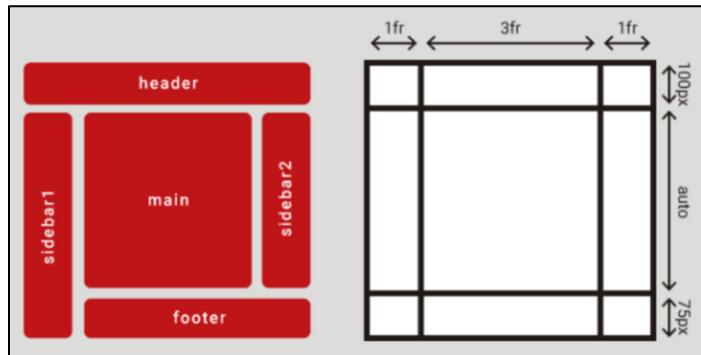
## Posicionando elementos: Grid Items

Tras conocer los elementos esenciales de creación de grid es necesario **aprender como posicionar y mover los diferentes elementos que componen nuestra web** dentro de la rejilla, es decir, es necesario entrar de lleno en la manipulación de los Grid Items.

Para conocer el funcionamiento utilizaremos un ejemplo. A continuación se muestra el código HTML y CSS para generar una estructura grid.

```
<div class="container">
    <div class="header"> </div>
    <div class="sidebar1"> </div>
    <div class="main"> </div>
    <div class="sidebar2"> </div>
    <div class="footer"> </div>
</div>
```

```
.container{
    display: grid;
    grid-template-rows: 100px auto 75px;
    grid-template-columns: 1fr 3fr 1fr;
}
```



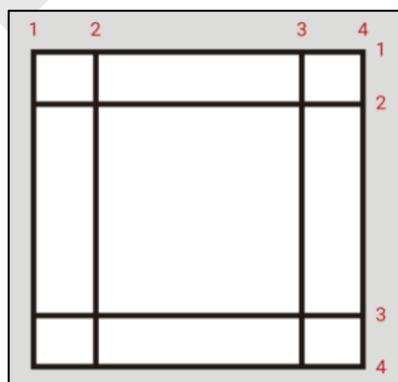
Después de la creación de la rejilla (grid) nos planteamos la siguiente pregunta:

**¿Cómo posicionamos cada uno de los *divs* existentes dentro de nuestro *container* en la rejilla que hemos creado ocupando un número de celdas específicas?**

Existen diferentes métodos para realizar esta acción:

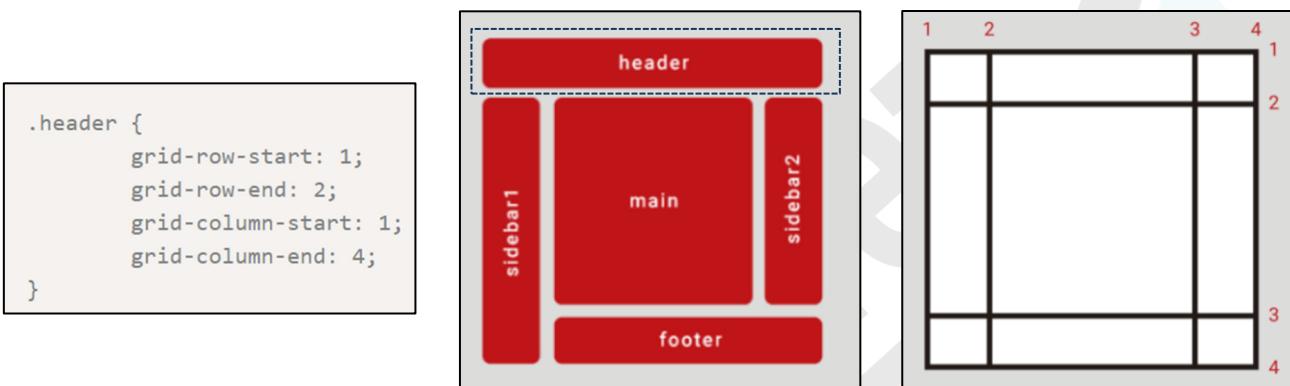
1. Posicionado mediante grid lines (*por número*)

Por defecto CSS grid ordena las rejillas de forma numérica, de izquierda a derecha y de arriba abajo. Esta es la distribución estándar de un grid.



Para empezar es necesario especificar donde empiezan y donde acaban el conjunto de filas y columnas que vamos a utilizar. Las propiedades que se deben utilizar son '**grid-row-start**', '**grid-row-end**', '**grid-column-start**', y '**grid-column-end**'.

A continuación, se configura la caja con el estilo '*header*' para que ocupe toda la parte superior de la rejilla a modo de cabecera. Se determina la **fila de inicio y de fin**, y la **columna de inicio y de fin**, de tal forma que todo lo que queda dentro de los límites formará parte de la caja.

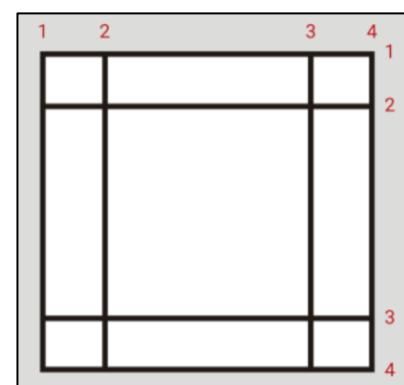
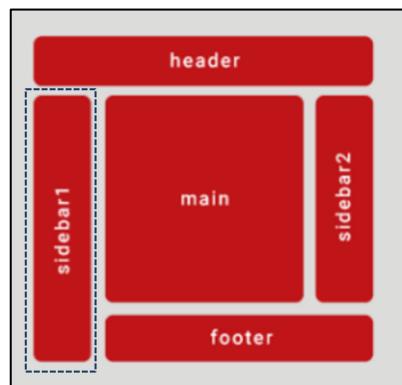


El mismo código CSS se puede realizar utilizando el **método abreviado** para las propiedades '**grid-column**' y '**grid-row**'. La diferencia respecto al sistema anterior es la eficiencia de código al escribir menos líneas. En este caso si solo se especifica un número toma como referencia la fila o la columna. Cuando se combinan varias filas o columnas es necesario la utilización de la barra ( / ).

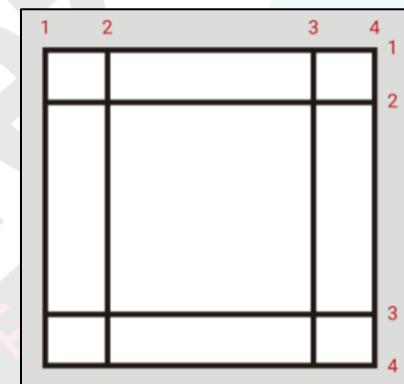
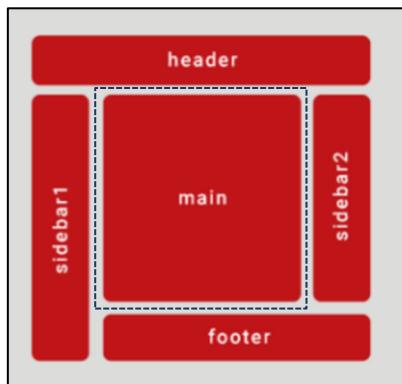
```
.header {  
    grid-row: 1;  
    grid-column: 1 / 4;  
}
```

Después es necesario configurar el resto de las cajas y sus correspondientes estilos para darle el formato que deseamos. Las siguientes imágenes muestran la configuración de cada caja utilizando el sistema abreviado.

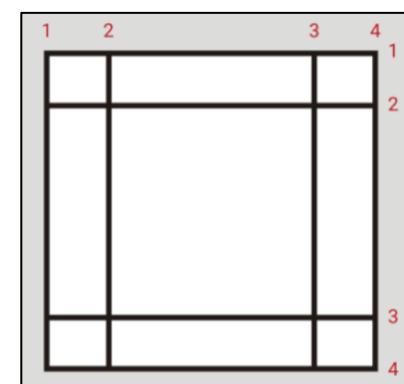
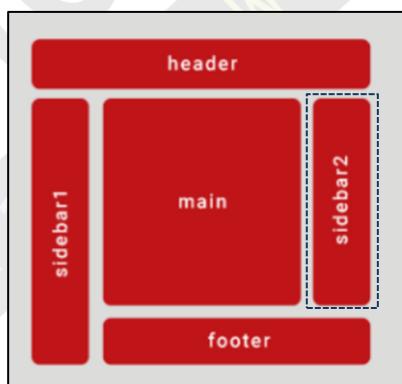
```
.sidebar1 {
    grid-row: 2 / 4;
    grid-column: 1;
}
```



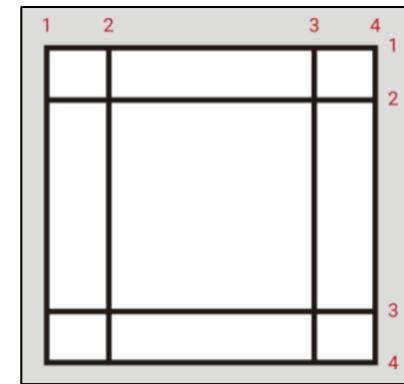
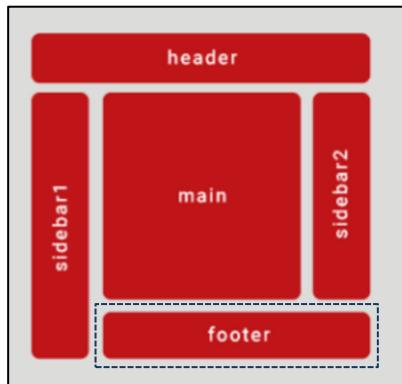
```
.main {
    grid-row: 2;
    grid-column: 2;
}
```



```
.sidebar2 {
    grid-row: 2;
    grid-column: 3;
}
```



```
.footer {
    grid-row: 3;
    grid-column: 2 / 4;
}
```



## 2. Posicionado mediante grid areas (*por nombre*)

Mediante la propiedad '**grid-area**' aplicada a nuestros Grid Items, asignaremos los límites tanto verticales como horizontales de nuestros elementos. Es como si cogiésemos a la vez las propiedades '**grid-row**' y '**grid-column**', y las comprimíesemos en una sola.

Este sistema de creación se puede realizar de dos formas aunque el resultado final es el mismo. C

En el caso de las '**grid-area**', el orden de configuración por orden que seguiremos será el que se muestra a continuación. Curiosamente siguen un sentido antihorario, justamente todo el contrario al resto de propiedades CSS cuando se deben construir.

**Grid-area:** límit superior / límit izquierdo / límit inferior / límit derecho

### Primera forma de creación: (por números) RECOMENDADO

Esta forma de configurar las áreas utiliza los números para definir cada uno de los límites de las áreas que hacen referencia a las filas de la rejilla.

```
.container{
    display: grid;
    grid-template-rows: 100px auto 75px;
    grid-template-columns: 1fr 3fr 1fr;
}
```

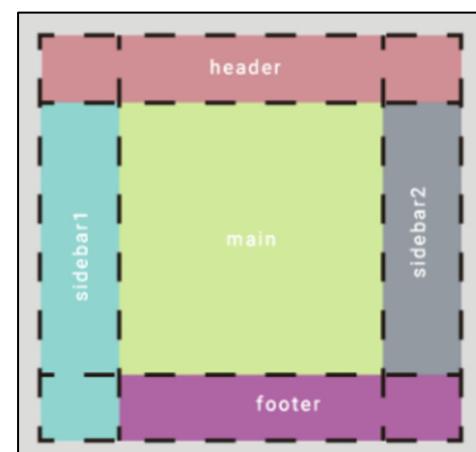
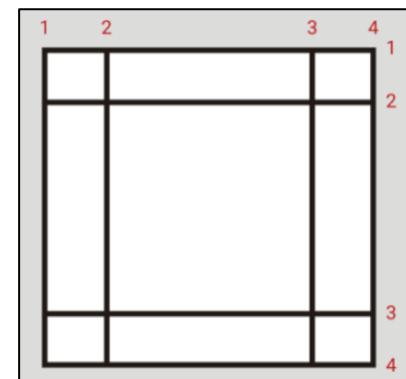
```
.header {
    grid-area: 1 / 1 / 2 / 4;
}

.sidebar1 {
    grid-area: 2 / 1 / 4 / 2;
}

.main {
    grid-area: 2 / 2 / 3 / 3;
}

.sidebar2 {
    grid-area: 2 / 3 / 3 / 4;
}

.footer {
    grid-area: 3 / 2 / 4 / 4;
}
```



## Segunda forma de creación de áreas: (por palabras)

Esta forma utiliza palabras para definir cada una de las 'casillas' que conforman nuestro grid. Aquí es necesario utilizar la propiedad '**grid-template-areas**' y configurarla en la caja madre. Las cajas hijas solo harán referencia al nombre en la configuración que se haya realizado en la caja madre.

Se define cada fila de la rejilla en la caja madre escribiendo el nombre de las cajas que lo formarán. Si algun área ocupa más de una fila se escribirá la configuración en dos filas independientes (ejemplo 'side1').

- Entre las diferentes filas no hay que escribir ningún carácter de separación como la coma (error habitual).
- Todas las filas tendrán el mismo número de palabras para que la cuadrícula esté perfectamente equilibrada en el número de filas y columnas.

### Ejemplo 1

```
.header {
    grid-area: head;
}

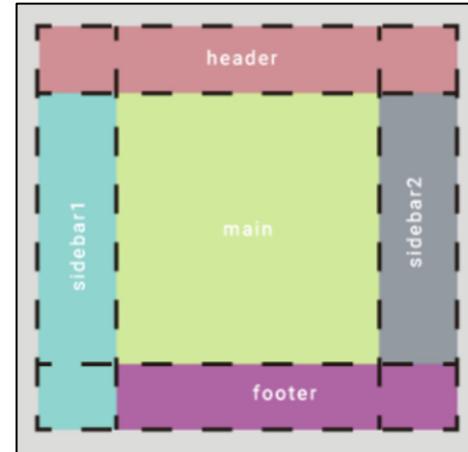
.sidebar1 {
    grid-area: side1;
}

.main {
    grid-area: main;
}

.sidebar2 {
    grid-area: side2;
}

.footer {
    grid-area: foot;
}
```

```
.container{
    display: grid;
    grid-template-rows: 100px auto 75px;
    grid-template-columns: 1fr 3fr 1fr;
    grid-template-areas: "head head head"
                        "side1 main side2"
                        "side1 foot foot"
}
```



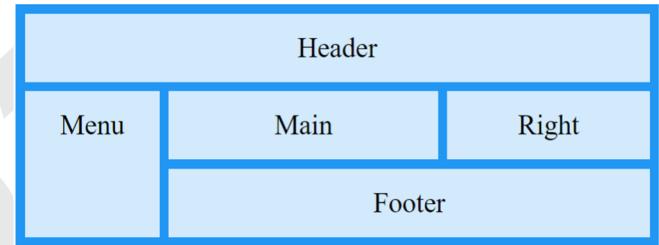
## Ejemplo 2

```
.grid-container {
    background-color: #2196F3;
    padding: 10px;

    display: grid;
    grid-gap: 10px;
    grid-template-areas:
        'header header header header header header'
        'menu main main main right right'
        'menu footer footer footer footer footer';
}
```

```
.item1 { grid-area: header; }
.item2 { grid-area: menu; }
.item3 { grid-area: main; }
.item4 { grid-area: right; }
.item5 { grid-area: footer; }
```

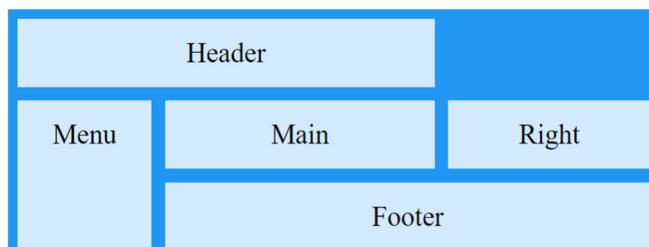
```
<div class="grid-container">
<div class="item1">Header</div>
<div class="item2">Menu</div>
<div class="item3">Main</div>
<div class="item4">Right</div>
<div class="item5">Footer</div>
</div>
```



Si la configuración de nuestra rejilla tuviese espacios en blanco entonces se substituye el nombre por un punto. Esta es la forma de decirle a grid que hay uno, o mas, espacios en blanco.

En el siguiente ejemplo se puede ver como las dos últimas posiciones que contiene el 'header', que en el apartado anterior estaban ocupadas, ahora se han dejado vacías debido a los dos puntos que se han sustituido es la definición de 'grid-template-areas'.

```
.grid-container {
    display: grid;
    grid-template-areas:
        'header header header header . .'
        'menu main main main right right'
        'menu footer footer footer footer footer';
    grid-gap: 15px;
    background-color: #2196F3;
    padding: 10px;
}
```

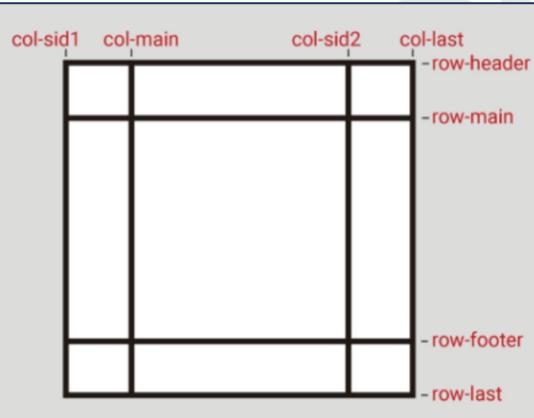


### 3. Posicionado mediante grid lines (*por nombre columna*) [POCO UTILIZADO]

El resultado final es exactamente igual que en el apartado anterior pero la sintaxis es diferente. En lugar de utilizar números para definir filas y columnas se utilizan nombres que previamente han sido definidos en la creación del grid.

La declaración de los nombres se asigna entre corchetes y siempre van precediendo a la unidad que especifica el tamaño de la fila o columna. La siguiente imagen muestra la declaración del grid asignando nombres a las filas y columnas.

```
.container{
    display: grid;
    grid-template-rows: [row-header] 100px [row-main] auto [row-footer] 75px [row-last];
    grid-template-columns: [col-sid1] 1fr [col-main] 3fr [col-sid2]
    1fr [col-last];
}
```



```
.header {
    grid-row: row-header;
    grid-column: col-sid1 / col-last;
}

.sidebar1 {
    grid-row: row-main / row-last;
    grid-column: col-sid1;
}

.main {
    grid-row: row-main;
    grid-column: col-main;
}

.sidebar2 {
    grid-row: row-main;
    grid-column: col-sid2;
}

.footer {
    grid-row: row-footer;
    grid-column: col-main / col-last;
}
```

## Métodos y propiedades CSS para Grid

Grid dispone de una serie de métodos que permite establecer unas configuraciones muy interesantes a la cuadrícula para que se adapten de la forma más correcta a los diferentes tamaños del viewport.

- **auto-fill:** Es una propiedad que permite distribuir el numero de columnas en función del ancho que se haya especificado a la caja madre.
- **minmax():** Este método permite establecer el mínimo y máximo de las columnas o las filas para evitar que las celdas del grid adopten un tamaño inapropiado para el dispositivo.

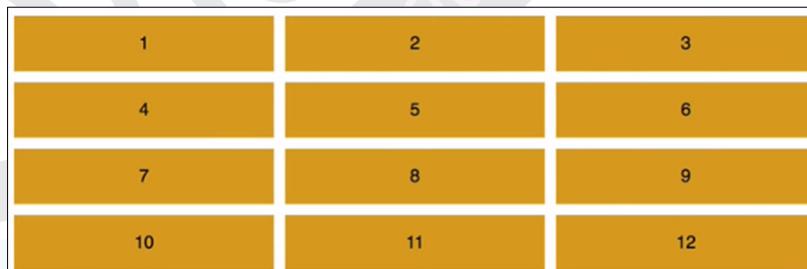
```
/*Crea 8 columnas de minimo 100px y maximo 1 fraccion*/
grid-template-columns: repeat(8, minmax(100px, 300px));

/*Crea 8 columnas de minimo 100px y maximo 1 fraccion*/
grid-template-columns: repeat(8, minmax(100px, 1fr));

/*Crea 8 columnas de ancho minimo al contenido de la celda y maximo 1 fraccion*/
grid-template-columns: repeat(8, minmax(min-content, 1fr));

/*Crea 8 columnas de ancho min y max respecto al contenido de la celda*/
grid-template-columns: repeat(8, minmax(min-content, max-content));
```

- **grid-auto-flow:** Es una propiedad de grid que determina cual será el flujo de creación de la cuadrícula. De la misma forma que sucedía con flexbox disponemos de un flujo que por defecto tiene el valor de '**row**' pero también puede ser '**column**'.



- **justify-items:** Propiedad que al igual que flexbox permite alinear el contenido de los elementos de cada celda del grid en el eje principal que por normal general es el horizontal (flex-start, center, flex-end, stretch que es el valor por defecto).



Por defecto el valor del atributo es **strech** y aprovecha el máximo espacio de la caja.