

LENGUAJES DE MARCAS Y SISTEMAS DE GESTIÓN DE INFORMACIÓN



CSS

Curso 2024-2025
CS DAW
IES Pazo da Mercé, As Neves
Esther Ferreiro Fdez.



Esta obra tiene un licencia
[Creative Commons Atribución-Compartir igual 4.0 Internacional \(CC BY-SA 4.0\)](https://creativecommons.org/licenses/by-sa/4.0/).

CONTENIDO

1	¿QUÉ SON LAS HOJAS DE ESTILO (CSS)?	1
2	DEFINICIÓN DE ESTILOS A NIVEL DE ETIQUETA	2
3	DEFINICIÓN DE ESTILOS A NIVEL DE PÁGINA	2
4	HOJAS DE ESTILO EN UN ARCHIVO EXTERNO	3
5	ELEMENTOS HTML <div> y 	4
6	SINTÁXIS	6
7	PROPIEDADES RELACIONADAS CON LAS FUENTES	6
7.1	font-family	7
7.2	font-size	7
7.3	font-style	8
7.4	font-weight	8
7.5	font-variant	8
7.6	Regla @font-face	8
8	SELECTORES CSS	9
8.1	Selector general	9
8.2	Etiquetas como selector	9
8.3	Clases como selector	9
8.4	ID como selector	11
8.5	Combinación de selectores	12
8.6	Selectores por atributos	14
9	PRIORIDAD EN LA DEFINICIÓN DE ESTILOS	14
9.1	Prioridad según el lugar donde esté definido el estilo	14
9.2	Etiquetas CSS por defecto	15
9.3	!important	15
9.4	Especificidad	15
10	PROPIEDADES RELACIONADAS CON EL TEXTO	16
10.1	color	16
10.2	text-align	17
10.3	text-decoration	17
10.4	letter-spacing, word-spacing, text-indent, text-transform	18
10.5	Interlineado: line-height	19
11	ESTILOS DE CAMPOS DE FORMULARIOS	19
12	HERENCIA DE PROPIEDADES DE ESTILO	20
12.1	Propiedades que no están afectadas por la herencia	21

12.2	Control de la herencia	22
13	PSEUDOELEMENTOS	23
14	PSEUDOCLASES	24
15	CAJAS	26
15.1	Fondos: background-color, background-image	27
15.2	height, width, padding	27
15.3	border	28
16	CAPAS	30
16.1	Visibilidad	31
16.2	Posicionamiento	33
16.3	Capas flotantes	34
17	POSICIONAMIENTO CON CSS ₃ FLEXBOX	37
17.1	Propiedades	38
18	POSICIONAMIENTO CON CSS ₃ GRIDAREAS	43
18.1	Determinar el número de filas y columnas	44
18.2	Tamaños	44
18.3	Plantillas de área	47
19	DISEÑO RESPONSIVE	48
20	VIEWPORT	50

1 ¿QUÉ SON LAS HOJAS DE ESTILO (CSS)?

En las primeras versiones de **HTML** se comenzaron a introducir elementos con la finalidad de mejorar la *presentación* del *contenido* de la página. Así aparecieron elementos para definir las fuentes, tamaños, colores, alineamiento de textos y bloques etc.

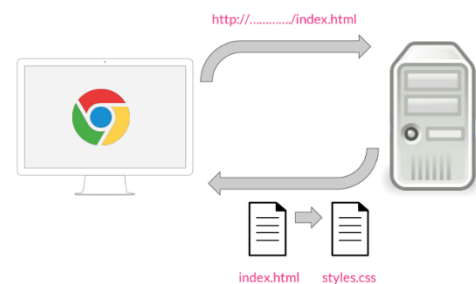
Con la aparición de las **Hojas de Estilo o CSS** (*Cascade Style Sheet*) se consiguió separar el *contenido* de la *presentación*. Con esto quedó perfectamente definido el objetivo de **HTML** (contenido) y **CSS** (presentación de ese contenido).

HTML no pone mucha atención en la apariencia del documento para eso se utilizan las hojas de estilo donde se describen cómo se presentan en el navegador los elementos creados con HTML.

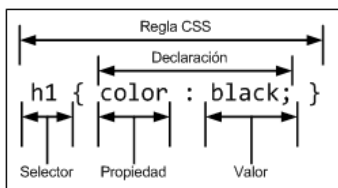
CSS es un lenguaje de diseño gráfico que tiene como objetivo definir y crear la presentación de un documento estructurado escrito en HTML.

Con **CSS** podemos especificar estilos como el tamaño, fuentes, color, espaciado entre textos y recuadros así como el lugar donde disponer texto e imágenes en la página.

Cuando se hace una petición de una URL desde un navegador al servidor Web, éste nos devuelve la página HTML. En el caso de la Web tenga una hoja de estilos asociada, el servidor también la enviará al navegador junto con la URL



El lenguaje de las Hojas de Estilo está definido en la Especificaciones CSS1 y CSS2 y CSS3 del World Wide Web Consortium ([W3C](http://www.w3c.org)) que es un estándar aceptado por toda la industria relacionada con la Web, o por lo menos, gran parte de navegadores (es verdad que los navegadores de *Microsoft* nos pueden dar un dolor de cabeza).



Se denomina **regla de estilos** al grupo de propiedades que se aplica a un selector o grupo de selectores.

Las reglas de estilo se pueden asociar a las etiquetas **HTML** de tres maneras:

- Definición de estilos a nivel de etiqueta.
- Definición de estilos a nivel de página o
- Definición de estilos en un archivo independiente con extensión *.css

En este tema se analizarán las tres metodologías, pero pondremos énfasis en la tercera forma, que es la más adecuada para separar el contenido de la página y la forma como se debe representar la misma por medio de la hoja de estilo.

Para consultar los posibles valores de cada propiedad, consultad en siguiente enlace <https://www.w3schools.com/cssref/index.php>

2 DEFINICIÓN DE ESTILOS A NIVEL DE ETIQUETA

La definición de estilos a nivel de etiqueta es la forma menos recomendada de aplicar un estilo a una etiqueta HTML. Esta práctica consiste en definir en la propiedad **style** los estilos para dicha etiqueta.

Por defecto, todo navegador tiene un estilo definido para cada etiqueta HTML, lo que hacemos con la propiedad *style* es redefinir el estilo por defecto. En este ejemplo definimos que la etiqueta *h1* defina como color de texto el rojo y como color de fondo el amarillo:

```
<h1 style="color:#ff0000;background-color:#ffff00">
  Este mensaje es de color rojo sobre fondo amarillo.
</h1>
```

Cada vez que inicializamos una propiedad debemos separarla de la siguiente por punto y coma.

Ejercicio

Crear una página que muestre el título de una noticia, a continuación un párrafo con el resumen de la noticia y finalmente la noticia completa. Utilizar las propiedades relacionadas con el texto para realzar la noticia en la página.

Noticia

Aquí descripción resumida de la noticia. Aquí descripción resumida de la noticia. Aquí descripción resumida de la noticia.

Aquí descripción detallada. Aquí descripción detallada. / descripción detallada. Aquí descripción detallada. Aquí descripci
detallada. Aquí descripción detallada. Aquí descripción detallada.
Aquí descripción detallada. Aquí descripción detallada. Aquí c
descripción detallada. Aquí descripción detallada. Aquí descripci

3 DEFINICIÓN DE ESTILOS A NIVEL DE PÁGINA.

En el bloque *head* de un documento Web se pueden definir estilos utilizando la etiqueta **<style>**.

Ejemplo 1: Mostrar dos títulos con texto de color rojo sobre fondo amarillo.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Problema</title>
    <style>
      h1 {
        color:#ff0000;
        background-color:#ffff00;
      }
    </style>
  </head>
  <body>
    <h1>Primer título</h1>
    <h1>Segundo título</h1>
  </body>
</html>
```

En este ejemplo indicamos al navegador que en todos los lugares de esta página donde se utilice la etiqueta *h1* debe aplicar como estilo de color de texto el rojo y fondo el amarillo.

En este ejemplo indicamos al navegador que en todos los lugares de esta página donde se utilice la etiqueta *h1* debe aplicar como estilo de color de texto el rojo y fondo el amarillo.

Ejemplo 2: En la etiqueta `<style>` definir distintos estilos para múltiples etiquetas:

```
<!DOCTYPE html>
<html>

<head>

  <title>Problema</title>
  <style>
    h1 {color:#ff0000}
    h2 {color:#00ff00}
    h3 {color:#0000ff}
  </style>
</head>
<body>
  <h1>rojo</h1>
  <h2>verde</h2>
  <h3>azul</h3>
</body>
</html>
```

Ejercicio

Definir un estilo diferente para la etiquetas *h1*, *h2*, *h3*, *h4*, *h5* y *h6* utilizando las propiedades de *color* y *background-color*.

4 HOJAS DE ESTILO EN UN ARCHIVO EXTERNO

La opción más recomendable para gestionar los estilos es el uso de un fichero externo que contendrá todas las reglas de estilo que afectarán a un fichero HTML.

Esta manera de trabajar implica una serie de ventajas:

- Se pueden aplicar las mismas reglas CSS a distintos documentos Web
- Permite separar el diseño del contenido.
- La hoja de estilos CSS se almacena en la caché del ordenador la primera vez que es utilizada, con lo cual, para las sucesivas páginas que requieran el mismo archivo de estilos, ese mismo archivo se rescata de la caché y no requiere que el servidor Web se lo reenvíe (ahorrando tiempo de transferencia).

Ahora veremos la primera página HTML que tiene asociada una hoja de estilo en un archivo externo. El archivo HTML es ([pagina1.html](#)):

```
<html>
  <head>
    <title>Problema</title>
    <link rel="stylesheet" href="stylesheet/estilos.css">
  </head>
  <body>
    <h1>Definición de hojas de estilo en un archivo externo.</h1>
    <p>Esta Web toma los estilos de un fichero css</p>
  </body>
</html>
```

El archivo que tiene las reglas de estilo es (*estilos.css*):

```
body {background-color:#eafadd;}
h1 {color:#0000cc;font-family:times new roman;font-size:25px;
    text-align:center;text-decoration:underline;}
p {color:#555555;font-family:verdana;text-align:justify;}
```

Para indicar el archivo de estilos externo debemos agregar en la cabecera (*head*) del archivo HTML la siguiente etiqueta:

```
<link rel="stylesheet" href="stylesheet/estilos.css">
```

La propiedad **href** hace referencia al archivo externo que afectará la presentación de esta página. En el atributo **rel** indicamos el tipo de información que tiene el fichero enlazado.

También se puede utilizar la siguiente orden

```
@import url("archivo.css")
```

La diferencia con el uso de **link** consisten en que **@import** puede ser usada desde otro fichero CSS o desde una etiqueta **<style>**.

Ejemplo

```
<style>
  @import url ("estilo.css");
  body{background-color:#ffffcc;}
</style>
```

La opción más recomendable y utilizada es la que hace uso de la etiqueta **<style>**.

De ahora en adelante nos acostumbraremos a trabajar con hojas de estilo definidas en un archivo externo, que es la forma más común de desarrollar un sitio web aplicando **CSS**.

En un **fichero de estilos** se pueden **incorporar comentarios** utilizando la siguiente sintaxis **/*comentario*/**

Ejercicio

Crear una página Web que contenga una cabecera de nivel 1 (**h1**), luego una cabecera de nivel 2 (**h2**) y un párrafo.

Define reglas de estilo para las tres etiquetas **h1**, **h2** y **p** dando valor a las propiedades vistas en conceptos anteriores. Crea una página que muestre el contenido de la forma más clara posible.

5 ELEMENTOS HTML <DIV> Y

Los elementos **div** y **span** nos permiten agrupar elementos y aplicar reglas de estilo en HTML. La diferencia fundamental entre estos dos elementos radica en que el elemento **div** genera un salto de línea antes y después del elemento, lo cual lo convierte en una marca de bloque similar a **h1**, **h2**, **p** etc. En cambio, el elemento **span** no produce un salto de línea porque se trata de un elemento en línea como lo son **a**, **strong**, etc.

Los elementos delimitados por **div** se suelen denominar **capas**.

Ejemplo:

- a) La primera línea de cada párrafo aparece indentada (**text-indent**).
- b) Algunas palabras del párrafo se muestran resaltadas en otro color.
- c) Todos los comentarios aparecen en color gris.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>título página</title>
  </head>
  <body>
    <div style="background-color:#eeeeee">
      <h3>Luis Pérez</h3>
      <p style="color:#888888;text-indent:20px">
        Aquí <span style="background-color:#eeee00">comentarios.</span>
        Aquí comentarios.Aquí comentarios.Aquí comentarios.Aquí comentarios
      </p>
      <h3>Ana Rodriguez</h3>
      <p style="color:#888888;text-indent:20px">
        <span style="background-color:#eeee00">Aquí comentarios.</span>
        Aquí comentarios. Aquí comentarios. Aquí...</p>
    </div>
  </body>
</html>
```

Luis Pérez

Aquí **comentarios** Aquí comentarios. Aquí comentarios. Aquí comentarios. Aquí comentarios.

Ana Rodriguez

Aquí comentarios. Aquí comentarios. Aquí comentarios. Aquí comentarios. Aquí...

El **div** agrupa los títulos y los párrafos, y define la propiedad **background-color** que afecta por todos los elementos contenidos por el **div**:

```
<div style="background-color:#eeeeee">
  <h3>Luis Pérez</h3>
  <p style="color:#888888;text-indent:20px">
    Aquí <span style="background-color:#eeee00">comentarios.</span>
    Aquí comentarios.Aquí comentarios. Aquí comentarios.
  </p>
</div>
```

Veamos como definimos los elementos **span**:

```
<span style="background-color:#eeee00">comentarios.</span>
```

Como vemos en el resultado de la página, el texto "**comentarios**" aparece con un color amarillo de fondo y podemos comprobar que el elemento **span** no produce salto de línea. Para ver la diferencia con el elemento **div** prueba a remplazar la palabra **span** por **div** y observa los resultados.

Luis Pérez

Aquí

comentarios.

Aquí comentarios. Aquí comentarios. Aquí comentarios. Aquí comentarios.

Ana Rodriguez

Aquí comentarios.

Aquí comentarios. Aquí comentarios. Aquí comentarios. Aquí...

Ejercicio

Crear una página que contenga dos **div** con una serie de párrafos cada una. Disponer color de fondo distinto para cada capa. En la segunda capa mostrar el mismo texto de la primera sección pero con algunas palabras tachadas (**text-decoration:line-through**)

6 SINTÁXIS

La sintaxis de CSS se basa en las siguientes normas:

- Para definir una propiedad CSS debemos indicar el nombre de la propiedad, como **font-size**, **text-decoration**... seguidos de dos puntos y el valor que le deseemos asignar. Podemos definir varias propiedades en un estilo separándolas por punto y coma.
- Para definir el estilo de una etiqueta se escribe la etiqueta seguida de la lista de propiedades CSS encerradas entre llaves.

```
h1{
  text-align: center;
  color:black
}
```

- A los atributos se les puede asignar distintas unidades de medida
 - Puntos: **pt**
 - Pulgadas: **in**
 - Centímetros: **cm**
 - Milímetros: **mm**
 - Pixels: **px**
 - Tanto por ciento: **%**
 - **Específicos del texto**
 - **em** (relativo a la M de la fuente del navegador)
 - **ex** (relativo a la X de la fuente del navegador)
 - **ch** (relativo al o de la fuente del navegador)
 - **rem** (relativo a la M de la fuente personalizada)

[Conversor de unidades](#)

7 PROPIEDADES RELACIONADAS CON LAS FUENTES

Contamos con una serie de propiedades relacionadas con las fuentes:

- | | | |
|--|--|---|
| • font-family
tipo de letra | • font-style
inclinación (cursiva) | • font-variant:
determina si la letra aparecerá en versalita. |
| • font-size
tamaño de la letra | • font-weight
grosor del trazo (negrita) | |

Ejemplo:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8"> <title>Problema</title>
    <style>
      h1 {
        font-family:times new roman;
        font-size:30px;
        font-style:italic;
        font-weight:bold;
      }
      h2 {
        font-family:verdana;
        font-size:20px;
      }
    </style>
  </head>
  <body>
    <h1>Titulo de nivel 1</h1>
    <h2>Titulo de nivel 2</h2>
  </body>
</html>
```

Titulo de nivel 1

Titulo de nivel 2

En el código se han definido dos reglas de estilos para las etiquetas **h1** y **h2**, eso significa que el navegador utilizará esas reglas para todo el texto marcado con estas etiquetas..

La primera regla definida para la etiqueta **h1** es:

```
h1 {  
  font-family:times new roman;  
  font-size:30px;  
  font-style:italic;  
  font-weight:bold;  
}
```

La segunda regla define sólo dos propiedades relacionadas a la fuente:

```
h2 {  
  font-family:verdana;  
  font-size:20px;  
}
```

Las propiedades que no se inicializan están afectadas por los valores que asigna el navegador.

7.1 font-family

La propiedad **font-family** permite determinar el tipo de fuente. Algunas de las fuentes más comunes son:

- Arial
- Arial Black
- Arial Narrow
- Courier New
- Georgia
- Tahoma
- Times New Roman
- Trebuchet MS
- Verdana

En el caso de que la fuente no esté disponible, el navegador selecciona el estilo por defecto para esa etiqueta HTML.

Podemos definir varias fuentes por si acaso alguna no se encuentra disponible para el navegador (se eligen de izquierda a derecha):

```
font-family: verdana, arial, georgia;
```

7.2 font-size

La propiedad **font-size** permite determinar el tamaño de la fuente. El tamaño se puede expresar en un gran número de medidas, siendo las más recomendadas: **em**, **px**, **%**.

La medida **em** es una medida relativa al tamaño de la letra que se está utilizando, por ejemplo, si utilizamos una tipografía de **12 ptos**, **1em** equivale a **12 ptos**.

También se pueden utilizar valores relativos al tamaño por defecto (que es **médium**)

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

Podemos dar un tamaño de letra relativo al tamaño del elemento padre con las siguientes propiedades.

- larger
- smaller

También podemos determinar un porcentaje positivo del cuerpo de la letra respecto al elemento padre dando un porcentaje.

Ejemplo:

```
/* Ajusta el texto del párrafo a un cuerpo de letra muy grande. */
p { font-size: xx-large }
/* Ajusta la cabecera de h1 a 2,5 veces del tamaño del texto. */
h1 { font-size: 250% }
/* Ajusta el texto incluido en span a 16px */
span { font-size: 16px; }
```

7.3 font-style

La propiedad **font-style**, que puede tener los siguientes valores :

- **normal**
- **italic**
- **oblique**

7.4 font-weight

La propiedad **font-weight**, puede tomar los siguientes valores:

- **normal**
- **bold**
- **bolder**
- **lighter**
- **100, 200..900**

Esta propiedad indica el *peso de la fuente* (mientras tenga un valor mayor los caracteres serán más rellenos). Los valores **bolder** y **lighter** indican más negrita y más clara, respectivamente, que la propiedad *padre*.

7.5 font-variant

La propiedad **font-variant** utiliza *versalitas*. Puede asumir estos dos valores:

- **small-caps**, las letras aparecerán en versalitas. [Enlace sobre fuentes:](#)
- **normal**, muestra las letras sin versalitas. [Enlace1](#) [Enlace2](#)

7.6 Regla @font-face

@font-face es una regla de CSS que permite a los desarrolladores web definir fuentes personalizadas para ser utilizadas en una página web. Esta regla es útil cuando se desea utilizar fuentes que no están disponibles de forma predeterminada en los navegadores del usuario.

El uso de **@font-face** se realiza en tres pasos:

1. **Definir la fuente:** Se indica la ubicación de los archivos de la fuente en el servidor web utilizando la propiedad **src** y se asigna un nombre a la fuente con la propiedad **font-family**.
2. **Especificar estilos:** Se pueden definir estilos para la fuente personalizada, como el tamaño, el estilo (normal, cursiva, negrita, etc.), el peso (normal, negrita, etc.), y otros.
3. **Utilizar la fuente:** Se puede aplicar la fuente personalizada a elementos HTML utilizando la propiedad **font-family** y especificando el nombre de la fuente definido en la regla **@font-face**.

```
@font-face {
  font-family: 'MiFuentePersonalizada';
  src: url('ruta/fuente.ttf'); /* Ubicación del archivo de la fuente */
}
body {
  font-family: 'MiFuentePersonalizada', sans-serif; /* Se aplica la
fuente personalizada a los elementos del cuerpo del documento */
}
```

En este ejemplo, se define una fuente personalizada llamada **MiFuentePersonalizada** y se especifica la ubicación del archivo de la fuente en el servidor web. Luego, se aplica esta fuente personalizada a los elementos del cuerpo del documento utilizando la propiedad **font-family** en la regla de estilo del cuerpo. Si la fuente personalizada no está disponible, se utilizará la fuente **sans-serif** como respaldo.

Ejercicio

Definir reglas para las etiquetas HTML: **h1**, **h2**, **h3**, **h4** y **h5**. Inicializar la propiedad **font-size** con valores decrecientes para cada uno de los títulos (40, 30, 25, 20, 15 y 10 píxeles). Inicializar la propiedad **font-family** para las tres primeras etiquetas con los valores: *Arial*, *Arial Black* y *Arial Narrow*.

8 SELECTORES CSS

Se denomina **selector CSS** al código que identifica a uno o varios elementos HTML con el objetivo de aplicar reglas de estilo. Podemos utilizar como selectores los siguientes:

- Selector general (*)
- Etiquetas HTML (h1, span, div...)
- Clases (.clase1, .clase2...)
- Identificadores (#identificador1, #identificador2...)

También podremos combinar los selectores para incrementar la precisión de los elementos afectados.

8.1 Selector general

El selector global (*) aplica sus estilos a todos los elementos HTML del documento, sin importar el tipo de etiqueta, clase, id o su ascendencia/descendencia.

Ejemplo: La siguiente regla afectaría a todos los elementos HTML.

```
* { color: #0000ff; }
```

8.2 Etiquetas como selector

Se puede usar como selector la etiqueta HTML directamente, la regla definida afectará a todos los elementos con la etiqueta utilizada.

Ejemplo: Definición del color para los elementos **h1**.

```
h1 { color: #0000ff; }
```

8.3 Clases como selector

En el caso de querer aplicar una misma regla de estilo a diferentes elementos, debemos utilizar clases para identificar los elementos HTML y utilizar un selector de clase en el fichero CSS para crear la regla de estilo con las propiedades y valores comunes.

Para conocer la sintaxis para la definición de clases veamos un ejemplo:

```
<!DOCTYPE html>
<html>
<head>
  <title>Problema</title>
  <link rel="StyleSheet" href="estilos.css">
</head>
<body>
  <h1 class="resaltado">Titulo de nivel 1</h1>
  <p> Este parrafo muestra el resultado de aplicar la clase .resaltado
  en la última <span class="resaltado">palabra.</span></p>
</body>
</html>
```

La hoja de estilo externa (*estilos.css*) contiene las siguientes reglas:

```
body {background-color:#eeeeee;}
.resaltado{
  color:#000000;
  background-color:#ffff00;
  font-style:italic;
}
```

A la hora de definir una *clase aplicable a cualquier etiqueta* HTML debemos referenciar al nombre de la clase con un punto delante:

```
.resaltado{
  color:#000000;
  background-color:#ffff00;
  font-style:italic;
}
```

Para aplicar la regla de estilo anterior debemos definir el atributo **class** y asignarle el nombre de la clase (sin el punto).

```
<h1 class="resaltado">Titulo de nivel 1</h1>
```

Una clase puede ser asignada a cualquier elemento HTML, dentro del **body**, tantas veces como sea necesario:

```
<p>Se aplica la clase en la siguiente <span class="resaltado">
palabra. </span> </p>
```

Ejercicio

Problema 1: Definir en la hoja de estilo estas dos clases:

```
.subrayado {
  color:#00aa00;
  text-decoration:underline;
}
.tachado {
  color:#00aa00;
  text-decoration:line-through;
}
```

En la página HTML, definir un título **h1** (subrayar todo el título) y un párrafo que tenga algunas palabras subrayadas y otras tachadas.

Problema 2: Definir estas dos reglas en la hoja de estilo externa. A continuación, crear una página HTML que contenga 3 preguntas y 3 respuestas. A cada pregunta y respuesta disponerla en un párrafo distinto. Asignar los estilos **.pregunta** y **.respuesta**

```
.pregunta {
  font-family:verdana;
  font-size:14px;
  font-style:italic;
  color:#0000aa;
}
.respuesta {
  font-family:verdana;
  font-size:12px;
  font-style:normal;
  text-align:justify;
  color:#555555;
}
```

8.4 ID como selector

Si utilizamos el **id** como selector, la regla solo se aplicará a un único elemento, que se corresponderá con el que contenga ese **id** como identificativo de..

La sintaxis para definir un estilo por medio de **id** es:

```
#cabecera {  
  font-family:Times New Roman;  
  font-size:30px;  
  text-align:center;  
  color:#0000ff;  
  background-color:#bbbbbb;  
}
```

Utilizamos el caracter almohadilla (#) para indicar que se trata de un estilo de tipo **id**.

El atributo **id** se puede introducir dentro de cualquier etiqueta HTML del *body* como cualquier otro atributo

```
<div id="cabecera">
```

Los dos archivos completos del ejemplo entonces quedan (*pagina1.html*):

```
<!DOCTYPE html>  
<html>  
<head>  
  <title>Problema</title>  
  <link rel="StyleSheet" href="estilos.css">  
</head>  
<body>  
  <div id="cabecera">  
    <h1>Título de la cabecera</h1>  
  </div>  
</body>  
</html>
```

Y el archivo de estilos (*estilos.css*) es:

```
#cabecera {  
  font-family:Times New Roman;  
  font-size:30px;  
  text-align:center;  
  color:#0000ff;  
  background-color:#bbbbbb;  
}
```

Ejercicio

Definir tres estilos de tipo **id** (*cabecera*, *cuerpo* y *pie*), luego definir en el archivo HTML tres áreas (**div**) inicializando el atributo **id** con los nombres de estilo creados.

8.5 Combinación de selectores

Un selector es el elemento que identifica a una regla CSS que puede ser, una etiqueta HTML, una clase o un identificador.

Los selectores se pueden combinar de formas diferentes:

- **Separados por comas**, indican que la misma regla se aplica a cada uno de los selectores:

```
h1, .special {  
  color: blue;  
}
```

Afecta a todas las etiquetas h1 y a aquellos elementos con el atributo class="special"

- **Separados por un espacio en blanco**, indica que la regla será aplicada en aquellos elementos que estén afectados por el conjunto de selectores que se referencian, de forma descendiente.

```
#principal .tipo2 p span {  
  margin: 10px;  
}
```

Afecta al contenido dentro de una etiqueta que esté afectada por una etiqueta <p>, que a su vez esté dentro de un elemento con la clase "tipo2" que también forme parte de un elemento con el id="principal"

- **Separador por un signo >**, selecciona el selector hijo de uno dado.

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

Afecta a todos los p, hijos directos de div

- **Separados un signo +**, indica que la regla será aplicada en aquellos elementos que sean hermanos adyacentes en el árbol DOM.

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

Afecta a aquellas etiquetas <p> que estén después de un <div>

Ejemplo 1: Probar la agrupación de selectores separador por comas, >, y + con el siguiente código:

```
<html>  
<head>  
  <title>Ejemplo</title>  
  <style> div+p { background-color: yellow; }</style>  
</head>  
<body>  
  <h1>Bienvenido a mi Blog</h1>  
  <div>  
    <h2>Mi nombre es Ana</h2><p>Vivo en Vigo</p>  
  </div>  
  <div>  
    <p>Estoy aprendiendo CSS</p>  
  </div>  
  <p>Y es fantástico!!!</p>  
</body>  
</html>
```

Ejemplo 2: Agrupación de varias etiquetas html con una misma regla de estilo. Supongamos que queremos la misma fuente y color para las etiquetas **h1**, **h2** y **h3**

```
<!DOCTYPE html>
<html>
<head>
  <title>Problema</title>
  <style>
    h1, h2, h3 {color: #0000ff;}
  </style>
</head>
<body>
  <h1>Título de nivel 1</h1>
  <h2>Título de nivel 2</h2>
  <h3>Título de nivel 3</h3>
</body>
</html>
```

Título de nivel 1

Título de nivel 2

Título de nivel 3

Ejemplo 3: En algunas circunstancias, es necesario desglosar la inicialización de propiedades en distintas reglas. Supongamos que queremos todas las cabeceras con la misma fuente pero tamaños de fuente distinta, entonces podemos implementarlo de la siguiente manera:

```
<html>
<head>
  <title>Problema</title>
  <style>
    h1,h2,h3,h4,h5,h6{
      font-family: Verdana; }
    h1 {font-size: 40px;}
    h2 {font-size: 30px;}
    h3 {font-size: 25px;}
    h4 {font-size: 20px;}
    h5 {font-size: 15px;}
    h6 {font-size: 10px;}
    p {font-size: 12px;}
  </style>
</head>
<body>
  <h1>Título de nivel 1</h1><h2>Título de nivel 2</h2>
  <h3>Título de nivel 3</h3><h4>Título de nivel 4</h4>
  <h5>Título de nivel 5</h5><h6>Título de nivel 6</h6>
  <p>Este es el párrafo utilizado para este ejercicio</p>
</body>
</html>
```

Título de nivel 1

Título de nivel 2

Título de nivel 3

Título de nivel 4

Título de nivel 5

Título de nivel 6

Es posible inicializar un conjunto de etiquetas HTML con una serie de propiedades de estilo comunes. Luego, agregamos en forma individual las propiedades no comunes a dichas etiquetas. Así, a la etiqueta **h1** realmente le hemos definido 2 propiedades: **font-family** y **font-size**. Lo mismo para las otras etiquetas HTML.

Ejercicio 1

Definir para la etiqueta **h5** y **p** el mismo tamaño de fuente (12px) pero poner color azul para el título y gris claro (**color:#aaaaaa**) para el párrafo. Agrupar las dos etiquetas para la definición de la fuente y posteriormente agregar a cada etiqueta el valor respectivo para el color del texto.

Ejercicio 2

Definir la misma fuente, color y tamaño para las etiquetas **p** (párrafo) y **h5** (tener en cuenta que cuando se muestre la página, el texto que se encuentra en la etiqueta **h5** difiere del texto que

se encuentra dentro del párrafo por la propiedad **font-weight** (ya que la etiqueta **h5** es de tipo *bold* y la etiqueta *p* tiene por defecto normal)

8.6 Selectores por atributos

Existen selectores que permiten filtrar los elementos HTML seleccionados por los atributos que poseen:

- **[nombre]**, selecciona los elementos que tienen establecido el atributo llamado *nombre*, independientemente de su valor.
- **[nombre=valor]**, selecciona los elementos que tienen establecido un atributo llamado *nombre* con el valor indicado.
- **[nombre~=valor]**, selecciona los elementos que tienen establecido un atributo llamado *nombre* y cuyo valor es una lista de palabras separadas por espacios en blanco en la que al menos una de ellas es exactamente igual al valor indicado.

Ejemplo

- Se muestran de color azul todos los enlaces que tengan un atributo **class**, independientemente de su valor

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

- Se muestran de color azul todos los enlaces que tengan un atributo **class** con el valor "externo"

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

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

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

- Se muestran de color azul todos los enlaces que tengan un atributo **class** en el que al menos uno de sus valores sea "externo"

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

9 PRIORIDAD EN LA DEFINICIÓN DE ESTILOS

9.1 Prioridad según el lugar donde esté definido el estilo

- **1º el estilo inline**, que prevalece siempre.

```
<h1 style="color:#ff0000;background-color:#ffff00">
  Este mensaje es de color rojo sobre fondo amarillo.
</h1>
```

- **2º la hoja de estilos interna:**

```
<head>
  <style> h1 {color: orange;} </style>
</head>
```

- **3º Y por último, el archivo externo.**

```
<head>
  <title>El título de la web</title>
  <link rel="stylesheet" href="stylesheet/estilos.css">
</head>
```

En el caso de que la llamada a una hoja de estilos externa estuviese después de la definición de hoja de estilos interna, la hoja de estilos externa tendría prioridad.

9.2 Etiquetas CSS por defecto

Los navegadores poseen ciertas reglas CSS por defecto (márgenes, tamaños de letra, estilo de los enlaces, etc.) que se aplican automáticamente a los elementos HTML en un sitio web cuando no se especifican estilos personalizados mediante CSS. Estas reglas determinan cómo se ven los elementos en términos de tamaño, color, fuente, espaciado, etc.

Se pueden sobrescribir las reglas por defecto, por ejemplo, es habitual sobrescribir los *márgenes* y la *padding* con el objetivo de ajustar el espaciado y la disposición de los elementos según el diseño previsto.

Para borrar todos los valores por defecto de los navegadores en CSS, se usa el llamado *reset* CSS. La regla más común para realizar esta acción es la siguiente:

```
*{
  margin:0;
  padding:0;
  box-sizing: border-box}
```

9.3 !important

En CSS, **!important** es una declaración que se puede añadir a una regla de estilo para indicar que debe prevalecer sobre otras reglas, incluso si tienen mayor prioridad.

```
/* Ejemplo de uso de !important */
.mi-elemento {
  color: red !important; /*Este estilo prevalecerá sobre otras reglas */}
```

Consejos de uso:

1. Úsalo con moderación.
2. Prioriza la especificidad en tus reglas de estilo.
3. Revisa el código existente antes de usarlo.
4. Documenta claramente su uso en tu código.
5. Prueba en varios navegadores.

Recuerda que **!important** debe utilizarse con precaución y solo en casos necesarios para evitar complicaciones en el mantenimiento y la escalabilidad del código CSS.

9.4 Especificidad

La especificidad es el factor más importante para determinar la prioridad de los estilos. Cuanto más específica sea una regla de estilo, mayor prioridad tendrá sobre otras reglas menos específicas.

Existe una ponderación de especificidad para las reglas CSS que se basa en un sistema de conteo que asigna puntos a diferentes tipos de selectores. Estos puntos se suman para determinar qué regla se aplicará a un elemento en caso de que haya múltiples reglas que puedan aplicarse. Aquí está la descomposición de cómo se asignan los puntos:

1. **Selectores de ID:** Cada selector de ID cuenta como 100 puntos.
Ejemplo: #miID = 100 puntos.
2. **Selectores de clase, atributos y pseudo-clases:** Cada uno, 10 puntos.
Ejemplo: .miClase, [type="text"], :hover = 10 puntos cada uno.
3. **Selectores de etiquetas (elementos) y pseudo-elementos:** Cada uno, 1 punto.
Ejemplo: div, p, span, ::before, ::after = 1 punto cada uno..
4. **Selectores universales y combinadores:** No cuentan para la especificidad.
Ejemplo: * (universal), >, +, ~ (combinadores) = 0 puntos.

Ejemplo:

Dado el siguiente código HTML:

```
<div class="container">
  <p class="text">Este es un párrafo</p>
</div>
```

Y el siguiente código CSS:

```
.container p.text {color: red}
p {color: blue}
```

En este ejemplo, la regla de estilo **.container p.text** tiene una mayor especificidad que la regla de estilo **p**, ya que utiliza clases y un selector de etiqueta, mientras que la regla **p** solo utiliza un selector de etiqueta. Por lo tanto, el párrafo dentro del elemento con la clase **container** se mostrará en rojo, ya que prevalece sobre la regla **p** que lo define en azul.

10 PROPIEDADES RELACIONADAS CON EL TEXTO

10.1 color

La propiedad CSS **color** nos permite definir el color del texto. Los valores de la propiedad pueden ser asignados por medio de tres valores hexadecimales que indican la mezcla de rojo, verde y azul. Por ejemplo si queremos rojo puro debemos indicar:

```
color:#ff0000
```

Si queremos verde puro:

```
color:#00ff00
```

Si queremos azul puro:

```
color:#0000ff
```

Luego si queremos amarillo debemos mezclar el rojo y el verde:

```
color:#ffff00
```

Hay muchos programas que nos permiten seleccionar un color y nos descomponen dicho valor en las proporciones de rojo, verde y azul.

Otra forma de indicar el color, si tenemos los valores en decimal, es por medio de la siguiente **Sintaxis**:

```
color:rgb(255,0,0);
```

Es decir, por medio de la función **rgb** (*red, green, blue*), indicamos la cantidad de rojo, verde y azul en formato decimal.

También se puede utilizar la función **rgba** (*red, green, blue, alpha*) donde le indicamos el canal Alpha. El canal Alpha indica el grado de transparencia de todos los colores.

Para convertir un color en formato RGB: <https://www.peko-step.com/es/tool/tfcolor.html>

Para consultar los códigos de colores ir a: http://es.wikipedia.org/wiki/Colores_HTML



10.2 text-align

La propiedad **text-align**, que puede tomar alguno de estos cuatro valores:

Si especificamos:

```
text-align:center;
```

El texto aparecerá centrado. Si queremos justificar a derecha, emplearemos el valor **right** y si queremos a la izquierda, el valor será **left**.

10.3 text-decoration

La propiedad **text-decoration** que nos permite entre otras cosas que aparezca subrayado el texto, tachado o una línea en la parte superior, los valores posibles de esta propiedad son:

- **none**
- **underline**
- **overline**
- **line-through**

Como ejemplo, definiremos estilos de para las etiquetas de cabecera **h1**, **h2** y **h3**:

```
<html>
<head>
<title>Problema</title>
<style>
  h1 {
    color:#ff0000;text-align:left;
    text-decoration:underline;}
  h2 {
    color:#dd0000; text-align:center;
    text-decoration:line-through ;}
  h3 {
    color:#aa0000;text-align:right;
    text-decoration:overline;}
</style>
</head>
<body>
  <h1>Título de nivel 1.</h1><h2>Título de nivel 2.</h2>
  <h3>Título de nivel 3.</h3>
</body>
</html>
```

Título de nivel 1

Título de nivel 2

Título de nivel 3

Es decir, para los títulos de *nivel 1* tenemos la regla:

```
h1 {
  color:#ff0000;
  text-align:left;
  text-decoration:underline;}
```

Es decir, color de texto rojo claro, el texto debe aparecer ajustado a la izquierda y subrayado. Luego para la etiqueta **h2** tenemos:

```
h2 {
  color:#dd0000;
  text-align:center;
  text-decoration:line-through ;}
```

El color sigue siendo rojo pero un poco más oscuro, el texto debe aparecer centrado y tachado. Y por último:

```
h3 {
  color:#aa0000;
  text-align:right;
  text-decoration:overline;}
```

Para los títulos de nivel tres, el texto es rojo más oscuro, alineado a derecha y subrayado por encima del texto.

10.4 letter-spacing, word-spacing, text-indent, text-transform

Las propiedades **letter-spacing** y **word-spacing** permiten indicar el espacio que debe haber entre los caracteres y entre las palabras.

La propiedad **text-indent**, sangra la primera línea de un texto. A partir de la segunda línea, el texto aparece sin sangrado. Podemos indicar un valor negativo con lo que la indentación es hacia la izquierda.

Por último, la propiedad **text-transform** puede inicializarse con alguno de los siguientes valores:

- **none**
- **capitalize**
- **lowercase**
- **uppercase**

Cada uno de estos valores transforma el texto como sigue:

- **capitalize**: Dispone en mayúsculas el primer carácter de cada palabra.
- **lowercase**: Convierte a minúsculas todas las letras del texto.
- **uppercase**: Convierte a mayúsculas todas las letras del texto.
- **none**: No provoca cambios en el texto.

El siguiente ejemplo define reglas para las etiquetas **h1** y **p**:

```
<html>
<head>
  <title>Problema</title>
  <style>
    h1 {
      letter-spacing:10px;
      word-spacing:30px;
      text-transform:capitalize;
    }
    p { text-indent:20px; }
  </style>
</head>
<body>
  <h1>Este es un título de nivel 1</h1>
  <p>Todo el texto siguiente se encuentra encerrada en la etiqueta de párrafo y con el objetivo de ver el efecto de la propiedad text-indent. La propiedad text-indent podemos inicializarla con un valor positivo, como es el caso actual o podemos inicializarla con un valor negativo lo que provocará que el texto de la primera línea del párrafo comience en una columna inferior al de todo el bloque. </p>
</body>
</html>
```

E s t e E s U n T í t u l o D e N i v e l 1

Todo el texto siguiente se encuentra encerrada en la marca de párrafo y con el objetivo de ver el efecto de la propiedad text-indent. La propiedad text-indent podemos inicializarla con un valor positivo, como es el caso actual o podemos inicializarla con un valor negativo lo que provocará que el texto de la primera línea del párrafo comience en una columna inferior al de todo el bloque.

La cabecera de *nivel uno*, tiene la siguiente regla:

```
h1 {
  letter-spacing:10px;
  word-spacing:30px;
  text-transform:capitalize;
}
```

Es decir que las letras deben tener una separación de 10 píxeles, las palabras deben estar separadas por 30 píxeles y por último deben disponerse en mayúsculas la primera letra de cada palabra.

Para la etiqueta *p* tenemos la siguiente regla:

```
p { text-indent:20px; }
```

Es decir, la primera línea del párrafo deberá imprimirse 20 píxeles a la derecha donde normalmente debería aparecer.

Ejercicio

Definir para la etiqueta de párrafo (*p*) el **color verde puro**, texto **centrado**. Imprimir varios párrafos con la **primera línea sangrada** para ver los resultados según la regla de estilo definida. Que valor debemos definir para que el texto aparezca con **color rojo oscuro**.

10.5 Interlineado: line-height

La propiedad **line-height** se utiliza para especificar el interlineado de un bloque de texto. Lo más normal es usar porcentajes o **em** para expresar el valor, aunque se puede indicar en cualquier unidad. Expresado en porcentajes el valor 100% corresponde al interlineado normal.

Ejemplos:

Párrafo con interlineado normal:

```
p { line-height: 100%; }
```

Párrafo con interlineado doble:

```
p { line-height: 200%; }
```

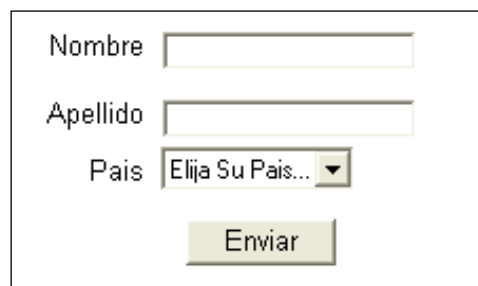
Ejercicio

Modifica el ejercicio anterior para ver el funcionamiento de distintos interlineados.

Si quieres consultar más información sobre el interlineado visita el siguiente [enlace](#).

11 ESTILOS DE CAMPOS DE FORMULARIOS

Un formulario por lo general se ve así...



El diseño que el navegador le da a los controles es el mismo que el del sistema operativo, en el caso del gráfico de arriba es de *Windows*, el problema surge cuando tenemos una web con un diseño muy cuidado y nos toca agregar formularios los cuales no concuerdan para nada con el diseño del sitio, para ello lo mejor que podemos hacer es utilizar estilos.

```
<style>
input {
  font-family: Tahoma, Verdana, Arial;
  font-size: 11px;
  color: #FFFFFF;
  background-color: #6699CC;
  border: #000099;
```

```

border-style: solid;
border-top-width: 1px;
border-right-width: 1px;
border-bottom-width: 1px;
border-left-width: 1px
}
select {
font-family: Tahoma, Verdana, Arial;
font-size: 11px;
color: #FFFFFF;
background-color: #6699CC;
border: #000099;
border-style: solid;
border-top-width: 1px;
border-right-width: 1px;
border-bottom-width: 1px;
border-left-width: 1px}
</style>

```

Después de aplicar estilos el formulario se vería de la siguiente manera:

12 HERENCIA DE PROPIEDADES DE ESTILO

El conjunto de las etiquetas HTML forman un árbol DOM (*Document Object Model*) en memoria y esta estructura condiciona el funcionamiento de CSS de tal forma que las propiedades de los elementos superiores del árbol son heredadas por los nodos enlazados en la misma rama.

En los siguientes ejemplos implementaremos muchos estilos que se heredan, es decir, si definimos la propiedad color para la etiqueta *h1*, luego si dicha etiqueta incorpora un texto con la etiqueta *b* (*bold*) en su interior, la propiedad color de la etiqueta *b* tendrá el mismo valor que la propiedad *h1* (es decir la etiqueta *b* hereda las propiedades de la etiqueta *h1*)

Ejemplo:

```

<!DOCTYPE html>
<html>
<head>
<title>Problema</title>
<style>
  body { color:#0000ff; font-family:verdana; }
</style>
</head>
<body>
  <h1>Este es un título de nivel 1 y con la etiqueta 'em' la palabra:
  <em>Hola</em></h1>
  <p>Todo este párrafo debe ser de color azul ya que lo hereda de la
  etiqueta body.</p>
  <a href="pagina2.html">Siguiendo ejemplo</a>
</body>
</html>

```

En este ejemplo hemos definido la siguiente regla para la etiqueta *body*:

```
body {color:#0000ff;font-family:verdana;}
```

Inicializamos la propiedad **color** con el valor de azul y la fuente de tipo *verdana*. Con esto, todas las etiquetas contenidas en el *body* que no redefinan estas dos propiedades heredarán las mismas. En este ejemplo la cabecera de primer nivel, el párrafo y el *hipervínculo* tienen como color el azul y la fuente es de tipo *verdana*.

Ahora bien, en muchas situaciones podemos redefinir propiedades para determinadas etiquetas. Veamos como podemos hacer que el texto contenido en las etiquetas *span* y *p* aparezcan de color distinto:

```
<html>
<head>
<title>Problema</title>
<style>
body {color:#0000ff;font-family:verdana;}
span {color:#008800;}
p {color:#999999;}
}
</style>
</head>
<body>
</body>
<h1>Este es un título de nivel 1 y con la etiqueta 'span' la palabra:
<spam>Hola</spam></h1> <p>Todo este párrafo debe ser de color gris ya
que lo redefine la etiqueta p y no lo hereda de la etiqueta body.</p>
</html>
```

Ahora hemos definido tres reglas, la primera define la propiedad **color** en azul y la **fuente** de tipo *verdana* para la etiqueta *body*:

```
body {color:#0000ff;font-family:verdana;}
```

La segunda regla define la propiedad **color** en verde para la etiqueta *span*, con esto no heredará el color azul de la etiqueta *body* (que es la etiqueta padre):

```
span {color:#008800;}
```

Algo similar hacemos con la etiqueta *p* para indicar que sea de **color** gris:

```
p {color:#999999;}
```

Pero podemos ver que todas las etiquetas heredan la **fuente** *verdana* ya que ninguna etiqueta la sobrescribe.

Ejercicio

Confeccione una página que define una regla para la etiqueta *body* e inicialice las propiedades *color*, *font-family*. Luego defina reglas de estilo para las etiquetas *h1*, *h2* y *h3* que redefinan la fuente con los valores: *Times New Roman*, *Courier* y *Arial*. Muestre tres *títulos*, cada uno con las etiquetas *h1*, *h2* y *h3*. Por último, muestre un párrafo.

12.1 Propiedades que no están afectadas por la herencia

Existen ciertas propiedades que no se heredan como son:

1. **Color y fondo:**
 - background
 - background-color
 - border-color

2. **Tamaño y medidas:**
 - width
 - height
 - margin
 - padding
3. **Box Model (Modelo de Caja):**
 - border
 - overflow
 - display (excepto en elementos que son inline)
4. **Posicionamiento y flotación:**
 - position
 - top, right, bottom, left
 - float
 - clear
5. **Estilos de listas:**
 - list-style
 - list-style-type
 - list-style-position
6. **Otras propiedades:**
 - opacity
 - visibility
 - z-index
 - text-align
 - text-indent
 - white-space

En el siguiente ejemplo el color de fondo del `<div>` padre será **lightblue**, pero no se heredará al `<p>` hijo. El color del texto será rojo, ya esta propiedad sí se hereda:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Ejemplo propiedades que no se heredan</title>
  <style>
    .padre {
      background-color: lightblue; /* No se heredará */
      width: 300px; /* No se heredará */
      padding: 20px; /* No se heredará */
      color: red; /* Se heredará */
    }
  </style>
</head>
<body>
  <div class="padre">
    <p class="hijo">Este texto será rojo y el fondo no se
heredará.</p>
  </div>
</body>
</html>
```

12.2 Control de la herencia

Todas las propiedades de CSS contienen los siguientes valores que pueden afectar a la herencia:

- **inherit**: valor por defecto, activa la herencia.
- **initial**: desactiva la herencia, establece como valor de la propiedad aquel que posea la hoja de estilos por defecto del navegador.

Para saber [más](#) sobre el control de la herencia

13 PSEUDOELEMENTOS

Los **pseudoelementos** son especificaciones de los elementos según su lugar o características dentro del elemento principal.

La sintaxis varía con respecto al concepto de clase visto anteriormente:

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

Existen selectores que permiten seleccionar partes del texto afectado para aplicar reglas concretas al mismo. Para ello podemos usar las siguientes pseudoclases:

- **:first-line** → afecta a la primera línea de texto de un elemento.
- **:first-letter** → afecta a la primera letra de texto de un elemento.
- **:first-child** → afecta al primer elemento de un tipo de contenido dentro de otro.

Ejemplo 1:

```
<!DOCTYPE html>  
<html>  
<head>  
  <meta charset="utf-8">  
  <title>Página ejemplo de CSS</title>  
  <style>  
    div:first-line{color: red; }  
    p:first-line{text-transform: uppercase;  
  </style>  
</head>  
<body>  
  <h1>Titulo de la página</h1>  
  <p>Parrafo de la página <br/> con regla de estilo propia</p>  
  <div>  
    Esto es una línea <br/>  
    y esto otra  
  </div>  
</body>  
</html>
```

Titulo de la página

PARRAFO DE LA PÁGINA
con regla de estilo propia

Esto es una línea
y esto otra

Ejemplo 2:

```
div {  
  border: black 5px solid;  
  margin: 10px;  
  padding: 10px;  
}  
  
div p:first-child {  
  color: red;  
}
```

Este es el primer párrafo <p> en una división que contiene tres párrafos.
Este es el segundo párrafo <p> en una división que contiene tres párrafos.
Este es el tercer párrafo <p> en una división que contiene tres párrafos.

14 PSEUDOCASES

Las pseudoclases son unas clases especiales que se refieren a distintos estados del elemento HTML, las que se utilizan fundamentalmente son las que se aplican a la etiqueta `<a>` (*ancla*).

La sintaxis varía con respecto al concepto de clase visto anteriormente:

```
a:pseudoclase {  
  propiedad: valor;  
}
```

Para la etiqueta HTML `<a>` tenemos 4 **pseudoclases** fundamentales:

- **link** - Enlace sin utilizar
- **visited** - Enlace ya utilizado
- **hover** - Enlace que tiene la flecha del mouse encima
- **active** - Es la que tiene foco

El orden en que definimos las pseudoclases es fundamental para su funcionamiento (debe respetarse el orden: **link-visited-hover-active**)

Este ejemplo permite ver los distintos estados que puede tener un enlace:

```
<html>  
<head>  
  <title>Problema</title>  
  <link rel="stylesheet" href="estilos.css">  
</head>  
<body>  
  <a href="http://www.google.com">Google</a>  
  <a href="http://www.yahoo.com">Yahoo</a>  
  <a href="http://www.msn.com">Msn</a>  
</body>  
</html>
```

La hoja de estilo es:

```
a:link{  
  background-color:#00ff00;  
  color:#ff0000; }  
a:visited{  
  background-color:#000000;  
  color:#ffffff; }  
a:hover{  
  background-color:#ff00ff;  
  color:#ffffff; }  
a:active{  
  background-color:#ff0000;  
  color:#ffff00; }
```

Al ejecutar la página los tres enlaces aparecen de color rojo con fondo verde:

```
a:link{  
  background-color:#00ff00;  
  color:#ff0000; }
```

Google Yahoo Msn

Si presionamos la tecla *tab* podremos ver que el enlace que tiene foco aparece de color amarillo con fondo rojo. Cabe destacar que esta pseudoclase no funciona como debería en la mayoría de los navegadores.

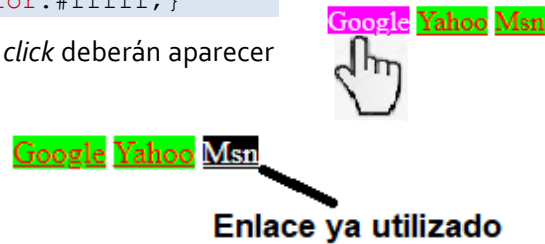
```
a:active{ background-color:#ff0000; color:#ffff00; }
```

Si pasamos la flecha del ratón sobre algún enlace veremos que aparece de color blanco con fondo lila:

```
a:hover{background-color:#ff00ff; color:#ffffff;}
```

Por último todos los enlaces que hayamos hecho *click* deberán aparecer de color blanco con fondo negro:

```
a:visited{
background-color:#000000;
color:#ffffff;}
```



Ejercicio

Definir la propiedad *font-size* con un valor distinto cuando la flecha del ratón está sobre el enlace.

CREACIÓN DE UN MENÚ CONFIGURANDO LAS PSEUDOCLASSES

A continuación, vamos a mostrar como hacer un menú para una web utilizando sólo **CSS** y **HTML** (en otros casos se requiere además de *JavaScript*).

El menú a implementar requiere agrupar en un **div** una serie de párrafos que contienen un hipervínculo cada uno. Cuando la flecha del mouse se encuentra sobre el hipervínculo cambiamos el color del fondo y la letra del hipervínculo.

El problema resuelto es:

```
<html>
<head>
  <title>Problema</title>
  <link rel="stylesheet" href="hojasestilo/AuxHojaEstilo.css">
</head>
<body>
  <div id="menu">
    <p><a href="https://www.w3schools.com/css/">Tutorial css</a></p>
    <p><a href="https://htmlcolorcodes.com/es/">Códigos de
colores</a></p>
    <p><a href="https://www.mclibre.org/consultar/htmlcss/">Apuntes
HTML/CSS</a></p>
    <p><a href="https://www.xataca.com">Xataca</a></p>
  </div>
</body>
</html>
```

La hoja de estilo asociada a esta página es:

```
#menu {font-family: Arial;}
#menu p {margin:0px; padding:0px;}
#menu a {display: block;padding: 3px;width: 160px;background-color:
#f7f8e8;border-bottom: 1px solid #eeeeee;text-align:center;}
#menu a:link, #menu a:visited {color:#1d7c14;text-decoration:none;}
#menu a:hover {background-color: #336699;color: #ffffff;}
```

Podemos decir que definimos un estilo por medio de un **id** llamado **menu**. Definimos una regla para este **id**:

```
#menu {font-family: Arial;}
```

La segunda regla:

```
#menu p {margin:0px; padding:0px;}
```

Estamos indicando que todos los párrafos contenidos en el estilo **#menu** deben tener cero en **margin** y **padding**.

Luego las **anclas** definidas dentro del estilo **#menu** definen las siguientes características:

```
#menu a {display: block; padding: 3px; width: 160px;
background-color: #f7f8e8; border-bottom: 1px solid #eeeeee;
text-align:center;}
```

El valor **block** para la propiedad **display** permite que el **ancla** ocupe todo el espacio del párrafo, indistintamente de la longitud del hipervínculo.

La propiedad **width** fija el tamaño máximo que puede tener el hipervínculo antes de provocar un salto de línea.

Por último inicializamos las pseudoclasas:

```
#menu a:link, #menu a:visited {color:#ff0000;text-decoration:none;}
#menu a:hover {background-color:#336699;color:#ffffff;}
```

Estamos definiendo el mismo color de texto para los vínculos seleccionados como aquellos que no han sido seleccionados:

```
#menu a:link, #menu a:visited {color:#ff0000;}
```

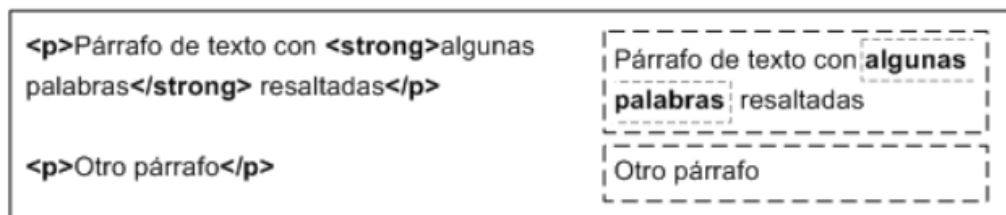
Por último cambiamos el color de fondo de la opción que tiene la flecha del mouse encima:

```
#menu a:hover {background-color:#336699;}
```

15 CAJAS

CSS trabaja con un modelo de cajas en el que considera a todo elemento HTML limitado por etiquetas es una caja.

Las cajas de una página se crean automáticamente. Cada vez que se inserta una etiqueta **HTML**, se crea una nueva caja rectangular que encierra los contenidos de ese elemento. La siguiente imagen muestra las tres cajas rectangulares que crean las tres etiquetas **HTML** que incluye la página:



15.1 Fondos: background-color, background-image

Las cajas pueden tener color de fondo (**background-color**) e imagen de fondo (**background-image**). En caso de tener las dos propiedades, predomina la *imagen de fondo*.

Ejemplo

```
<!DOCTYPE html>
<html>

<head>
  <meta charset="utf-8">
  <title>Página ejemplo de CSS</title>
  <style>
    div#primera {
      height: 100px;
      background-image:url("images/foto.jpg");
    }
  </style>
</head>

<body>
  <h1>Titulo de la página</h1>
  <div id="primera">
    La montaña mágica
  </div>
</body>
</html>
```

Titulo de la página



15.2 height, width, padding

Al trabajar con un modelo de caja, debemos tener en cuenta que existen dos propiedades principales para definirla, **height** y **width**, que definirán el ancho y alto del **área de contenido** de nuestra caja.

También existen las propiedades **max-height** y **max-width** para determinar el máximo valor de ancho y de largo en aquellas circunstancias en que se modifica el tamaño debido a la modificación de las dimensiones del navegador/pantalla.



En las cajas se pueden definir características de margen interno con la propiedad **padding** y del margen externo con la propiedad **margin**. También podemos modificar los valores independientes que se indican a continuación:

margin-bottom	padding-bottom	border-bottom
margin-left	padding-left	border-left
margin-right	padding-right	border-right
margin-top	padding-top	border-top

Ejemplo

```
<html>
<head>
  <meta charset="utf-8">
```

CSS 3

Podemos hacer que la caja de un texto se ajuste a su contenido con la siguiente regla:

```
h1{
  width: fit-content;
  height: fit-content;}
```

Titulo de la página

En la primera caja

Esto es una línea
y esto otra

```

<title>Página ejemplo de CSS</title>

<style>
  div#primera {
    width: 50%; height: 100%;
    padding: 10px; background-color: black;
    color: cornsilk; }
  div#segunda {
    margin: 20px;
    max-width: 500px;
    padding: 10px;
    background-color: rgb(243, 237, 148);
    color: darkblue; }
</style>

</head>
<body>
  <h1>Titulo de la página</h1>
  <div id="primera">
    <p>En la primera caja </p>
    <div id="segunda">
      Esto es una línea <br />y esto otra
    </div>
  </div>.....

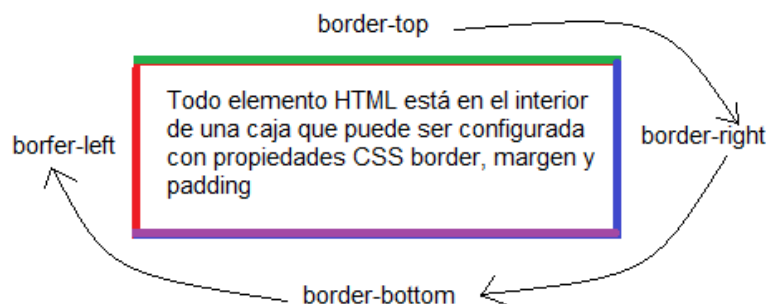
```

Probar a modificar los distintos valores para comprender el funcionamiento de las propiedades analizadas.

[Ver más información del modelo de caja.](#)

15.3 border

La propiedad compuesta **border** permite establecer características de los bordes de un texto, párrafo, celda o cualquier otra agrupación HTML.



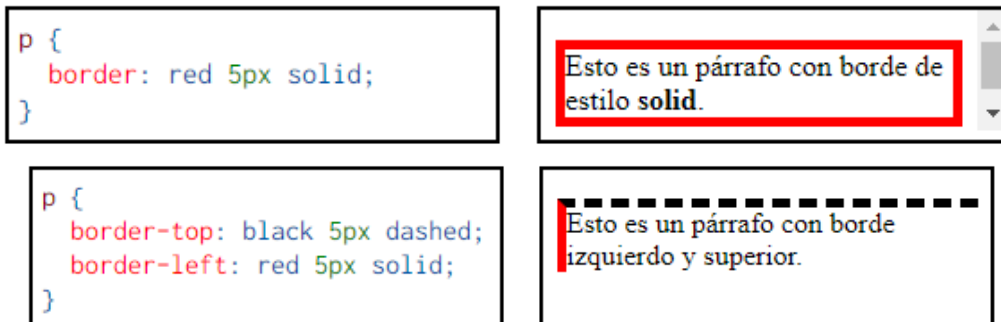
Es posible establecer propiedades de los bordes de forma individual: **border-left**, **border-top**, **border-right** y **border-bottom** o fijar las propiedades de todos ellos en una misma propiedad identificada por **border**.

Cada uno de los lados de los bordes o su conjunto permiten definir las siguientes propiedades, que se pueden introducir en cualquier orden:

- **color** → nombre del color o código RGB
- **grosor** → puede ser **thin** (fino), **medium** (medio) o **thick** (grueso))

- **estilo** --> → *none* (ninguno), *hidden* (oculto), *dotted* (puntos), *dashed* (rayas), *solid* (continuo), *double* (doble), *groove* (efecto hundido), *ridge* (efecto hacia afuera), *inset* (efecto 3D hacia adentro) y *outset* (efecto 3D hacia afuera).

Ejemplo:

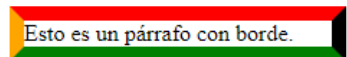


Ver más ejemplos de la propiedad **border**

En los bordes también podemos cambiar forma individual las propiedades de los bordes utilizando las siguientes propiedades:

- **border-color** → permite cambiar el color del borde.
- **border-width** → permite cambiar el grosor del borde.
- **border-style** → permite cambiar el estilo del borde .

```
p {
  border-color: red black green orange;
  border-width: 10px;
  border-style: solid;
}
```



Ver más ejemplos de las propiedades **border-color**, **border-width**, **border-style**

16 CAPAS

Una capa es un bloque HTML que puede ser posicionado dinámicamente en una página Web. Esto quiere decir que podemos indicar su posición por medio de reglas CSS en un lugar diferente del que corresponde a la capa por suposición en el código HTML.

En HTML las capas se creaban con la etiqueta **div**, en HTML5 también puede determinar una capa todas las etiquetas semánticas de estructura de una Web (**section**, **nav**, **article**, etc.)

Ejemplo: Definimos una serie de capas con **div**.

```
<!DOCTYPE html>
<html>

<head>
  <meta charset="utf-8">
  <title>Página ejemplo de CSS</title>

  <style>
    #capa1 {
      width: 400px;
      height: 100px;
      border: solid 5px rgb(17, 68, 209);
    }
    #capa2 {
      width: 400px;
      height: 100px;
      border: solid 10px rgb(24, 133, 30);
    }
    #capa3 {
      width: 200px;
      height: 50px;
      border: solid 5px rgb(203, 216, 31);
    }
  </style>
</head>

<body>

  <div id="capa1">En la capa1</div>
  <p>Este es un texto no está en ninguna capa</p>

  <div id="capa2">
    En la capa 2

    <div id="capa3">
      En la capa 2 que está dentro de la 2
    </div>

  </div>

</body>
</html>
```

En la capa1

Este es un texto no está en ninguna capa

En la capa 2

En la capa 2 que está dentro de la 2

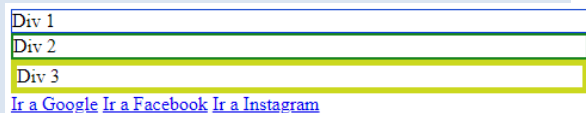
16.1 Visibilidad

Existen distintas propiedades que determinan la visibilidad de una capa:

- **display** → Determina distintos modos de visualización/comportamiento del elemento:
 - **inline**: Visualiza un elemento en forma de elemento en línea, independientemente del tipo de elemento que se trate. No acepta las propiedades *width*, *height* ni márgenes verticales.
 - **block**: Muestra un elemento como si fuera un elemento de bloque, independientemente del tipo de elemento que se trate y acepta las propiedades *width*, *height* y márgenes verticales.
 - **inline-block**: Combina la visualización de *inline* y *block*, se muestran en la misma línea (respetando el flujo) todos los elementos y además, acepta las propiedades *width*, *height* y márgenes verticales.
 - **none**: No se muestra el elemento ni ocupa ningún lugar en la página Web, hace que el resto de los elementos se desplacen como si no existiera.

Suponemos la siguiente combinación de HTML/CSS en la que aparecen elementos que por defecto son de **bloque (div)** e **inline (a)**.

```
<!DOCTYPE html>
<html>
<head>
  <title>Página ejemplo de CSS</title>
  <style>
    #capa1 {border: solid 1px rgb(17, 68, 209);}
    #capa2 {border: solid 2px rgb(24, 133, 30);}
    #capa3 {border: solid 5px rgb(203, 216, 31); }
  </style>
</head>
<body>
  <div id="capa1">Div 1</div>
  <div id="capa2">Div 2</div>
  <div id="capa3">Div 3</div>
  <a href="http://www.google.es">Ir a Google</a>
  <a href="http://www.twitter.es">Ir a Facebook</a>
  <a href="http://www.instagram.com">Ir a Instagram</a>
</body>
</html>
```



Por defecto, los elementos de *block* provocan un salto de línea y los elementos *inline*, no.

Podemos cambiar su comportamiento modificando las reglas de estilo:

```
<style>
  #capa1 {display: inline;
    border: solid 1px rgb(17, 68, 209);}
  #capa2 {display: inline;
    border: solid 2px rgb(24, 133, 30);}
  #capa3 {display: inline;
    border: solid 5px rgb(203, 216, 31);}
  a{ display: block;}
</style>
```

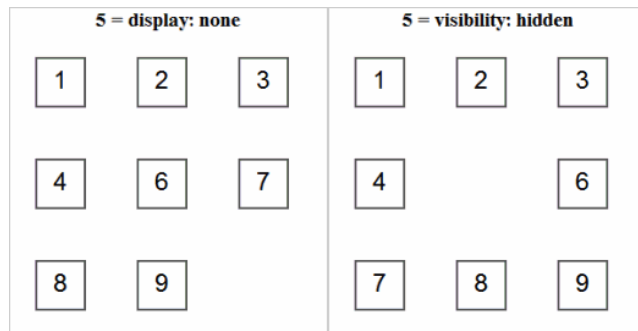


Para saber más de **display** Para ejecutar online distintas propiedades *inline-block*

- **visibility** → Permite hacer invisible un elemento, lo que significa que el navegador crea la caja del elemento pero no la muestra. En este caso, el resto de elementos de la página no modifican su posición, ya que aunque la caja no se ve, sigue ocupando sitio.

- **visibility**: visible.
- **hidden**: invisible.

En la siguiente imagen se ve la diferencia entre hacer la **capa 5 display: none** y **visibility: hidden**.



- **overflow** → Esta propiedad determina el comportamiento de los elementos cuando desborda el espacio reservado para los mismos. Podemos determinar recortar contenido, dibujar barras de desplazamiento o mostrar el contenido excedente en un elemento a nivel de bloque.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="author" content="Francesc Ricart">
  <title>Document</title>

  <style>
    div {width: 150px; height: 100px; border: 1px solid red;
        padding: 1%; float: left}
    #capa1 { overflow: visible; }
    #capa2 { overflow: hidden; }
    #capa3 { overflow: scroll; }
  </style>

</head>
<body>
  <div id="capa1">
    Visible Lorem ipsum dolor
    sit amet, insectetur
    adipisicing elit.
    Voluptates debitis iste
    possimus doloribus
    doloreque earum dolor dolore fuga, eligendi itaque recusandae
    hic aperiarn, cumque expedita, odio laborum. Tenetur,
    necessitatibus modi!
  </div>
  <div id="capa2">
    Hidden Lorem ipsum dolor sit amet, consectetur adipisicing
    elit. Voluptates debitis iste possimus doloribus doloreque
    earum dolor dolore fuga, eligendi itaque recusandae hic
    aperiarn, cumque expedita, odio laborum. Tenetur,
    necessitatibus modi!
  </div>
  <div id="capa3">
    Scroll Lorem ipsum dolor sit amet, consectetur adipisicing
    elit. Voluptates debitis iste possimus doloribus doloreque
    earum dolor dolore fuga, eligendi itaque recusandae hic
    aperiarn, cumque expedita, odio laborum. Tenetur,
    necessitatibus modi!
  </div>
</body>
</html>
```

```
earum dolor dolore fuga, eligendi itaque recusandae hic aperiam,
cumque expedita, odio laborum. Tenetur, necessitatibus modi!
</div>
<div id="capa3">
  Scroll Lorem ipsum dolor sit amet, consectetur adipisicing
elit. ...., necessitatibus modi!
</div>
```

Las capas se alinean por la parte inferior, si queremos ver alineadas por la parte superior deberíamos añadir a cada **#capa*i*** la propiedad **float:left**; . La propiedad **float** se utilizó durante muchos años para maquetar las capas en una página Web. CSS3 aportó nuevos mecanismos para este función que veremos a continuación.

- **z-index** → Su valor es un entero mayor o igual que 1, indica el orden de apilamiento y por tanto que capa se visualizará por encima de otra, en el caso de que se solapen.

16.2 Posicionamiento

Existen numerosos atributos para posicionar con CSS cualquier elemento de la página. Además, a medida que van siendo presentadas nuevas versiones de CSS, estos atributos y sus posibles valores van aumentando.

- **position** → Determina distintos modos de visualización/comportamiento del elemento:
 - **absolute**: El punto de referencia es la esquina superior izquierda del navegador o del elemento contenedor en el que se encuentra. No afecta al resto de elementos que se desplazan como si el elemento afectado por esta propiedad no existiera.
 - **fixed**: El punto de referencia es la esquina superior izquierda del navegador o del elemento contenedor en el que se encuentra. El elemento se mantendrá en su lugar aunque nos desplazemos con el *scroll*.
 - **relative**: Permite que un elemento se desplace respecto a lo que hubiera sido su posición normal; el resto de elementos continúan en su posición ignorando al que se desplaza.
 - **static**: Es el comportamiento por defecto. Este valor indica que la capa no tiene posicionamiento específico y que el navegador utilizará un sistema automático para ubicarlo según la posición del código HTML.
- **top, right, bottom y left** → Estas propiedades permiten definir el desplazamiento de la posición de un elemento respecto a un origen de coordenadas. Si el atributo *position* es **absolute**, el valor dado a una de estas propiedades, tendrá como referencia la distancia del borde superior de la capa con respecto al borde superior de la página web. Si el atributo *position* era **relative**, el valor indica la distancia desde donde se estaba escribiendo en ese momento en la página hasta el borde superior de la capa en la que está contenido.

Casos particulares

- Se admiten valores de desplazamientos negativos.
- En el caso de usar porcentajes, éste se calcula:
 - a) Respecto al valor de altura (**height**) del elemento si se aplican con **top** ó **bottom**.
 - b) Respecto al valor de anchura (**width**) del elemento si se aplican con **right** ó **left**.
- No tiene sentido utilizar **top** y **bottom** al mismo tiempo, porque sería decir que el elemento sube y baja. **Hay que utilizar sólo una de estas dos propiedades**. Lo mismo podemos decir para **right** y **left**.

16.3 Capas flotantes

Un elemento flotante es una caja que es desplazada a la izquierda o a la derecha en la línea actual. La característica más interesante de una caja flotante es que el contenido puede fluir por sus lados. El contenido fluye por el lado derecho de una caja flotante a la izquierda y por el lado izquierdo de una caja flotante a la derecha.

Una caja flotante se desplaza a la derecha o izquierda de su contenedor o al borde externo de otro elemento flotante. La parte superior de la caja flotante se alinea con la parte superior de su contenedor o la parte inferior del elemento anterior (si no existe un contenedor). Si no hay suficiente espacio horizontal en la línea actual para la caja flotante, ésta es desplazada hacia abajo, línea a línea, hasta que encuentra espacio suficiente.

Puesto que una caja flotante no está en el flujo normal, las cajas de bloque creadas antes y después fluyen verticalmente como si la caja flotante no existiera. Sin embargo, las líneas de texto al lado de la caja flotante son acortadas para darle espacio.

La propiedad **float** indica donde se va a posicionar la caja flotante y como va a fluir el texto a su alrededor. Los posibles valores de esta propiedad son:

- **left**: el elemento genera una caja de bloque que flota a la izquierda. El contenido fluye por el lado derecho de la caja, comenzando en la parte superior
- **right**: igual que *left*, pero la caja se posiciona a la derecha y el contenido flota a la izquierda.
- **none**: la caja no es flotante.

Si una capa flotante tiene a su altura imagenes u otros elementos alineados a la derecha o la izquierda, con la propiedad **clear** hacemos que se coloque en un lugar donde ya no tenga esos elementos en el lado que indiquemos. Los posibles valores de esta propiedad son:

- **left**: no deja flotar nada a su izquierda.
- **right**: no deja flotar nada a su derecha
- **both**: no deja flotar ni a su izquierda ni a su derecha
- **none**: no pone ninguna restricción.

Disposición de 2 columnas, cabecera y pie

A continuación, se aportan los códigos necesarios para crear la estructura de una cabecera de página, seguida de dos columnas y un pie de página. Implementala en tu equipo y analiza las el funcionamiento de las capas flotantes.

```
<html>
<head>
<title>Ejercicio</title>
```

```

<link rel="stylesheet" href="hojasestilo/AuxHojaEstilo.css">
</head>
<body>
  <div id="contenedor">image.png
    <div id="cabecera">
      <h1>Título de la página</h1>
    </div>
    <div id="columna1">
      <h2>Título de la columna 1</h2>
      <p>Contenido de la columna 1</p>
    </div>
    <div id="columna2">
      <h2>Título de la columna 2</h2>
      <p>Contenido de la columna 2. </p>
    </div>
    <div id="pie">
      Pie de página.
    </div>
  </div>
</body>
</html>

```

Título de la página	
Título de la columna 1 Contenido de la columna 1	Título de la columna 2 Contenido de la columna 2.
Pie de página.	

----- AuxHojaEstilo.css -----

```

* {
  margin:0px;
  padding:0px;
  text-align: center;
}
#contenedor{
  width:100%;
  margin:0px;
  border:1px solid #000000;
  line-height:130%;
  background-color:#f2f2f2;
}
#cabecera{
  padding:10px;
  color:#fff;
  background-color:#becdfe;
  clear:left;
}
#columna1{
  float:left;
  width:50%;
  margin:0;
  padding:1em;
}
#columna2{
  margin-left:50%;
  border-left:1px solid #aaaaaa;
  padding:1em;
}
#pie {
  padding:10px;
  color:#fff;
  background-color:#becdfe;
  clear:left;
}

```

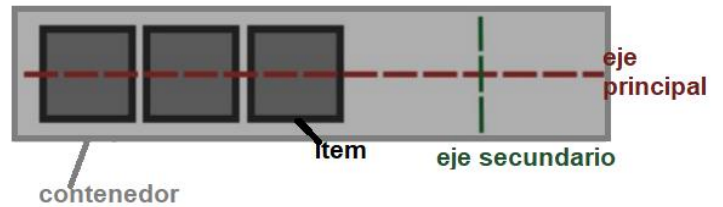
Para practicar y obtener más información de la maquetación con float ir a [Enlace1](#) [Enlace2](#)

17 POSICIONAMIENTO CON CSS3 FLEXBOX

CSS3 ha definido un nuevo modelo de posicionamiento que se denomina **flexbox** que permite distribuir elementos en filas o columnas.

Los elementos de un **flexbox** fluyen de tal forma que se adaptan al tamaño de la pantalla en las direcciones establecidas. Una caja **flexbox** se compone de los siguientes elementos:

- **Eje principal:** El eje principal es el eje a lo largo del cual se disponen los distintos elementos del **flexbox** y se define con la propiedad *flex-direction*.
- **Eje secundario:** El eje perpendicular al eje primario se llama eje secundario o transversal y su dirección siempre depende del eje principal.
- **Elemento contenedor:** La posición del contenedor puede variar según el



tamaño del navegador si utilizamos la propiedad *flexible: flex*

- **Items:** Los hijos del contenedor flexible son los elementos que contiene, que pueden ser cualquier tipo de bloque.

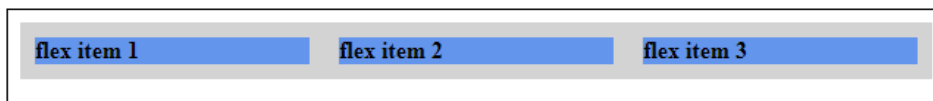
Ejemplo: realizaremos las pruebas de las propiedades de *flexbox* con el siguiente código.

```
<!DOCTYPE html>
<html>
<head>
<title>Ejemplo de posicionamiento con flexbox</title>
<style>
  .contenedor {display: flex; background-color: lightgrey;}
  .item {background-color: cornflowerblue; width: 33%; margin: 10px;}
</style>
</head>
<body>
<div class="contenedor">
  <div class="item"><strong>flex item 1 </strong> </div>
  <div class="item"><strong>flex item 2 </strong> </div>
  <div class="item"><strong>flex item 3 </strong> </div>
</div>
</body>
</html>
```

Para utilizar las distintas propiedades CSS relacionadas con el posicionamiento en un contenedor, en primer lugar, se debe indicar el tipo de contenedor con las propiedades:

- a. **display: flex** → El contenedor se comporta como un bloque y sus elementos ocupan todo el espacio horizontal disponible.

```
.contenedor{display: flex;}
```



- b. **display: inline-flex** → Los ítems solo ocupan el espacio horizontal necesario.

```
.contenedor{display: inline-flex;}
```

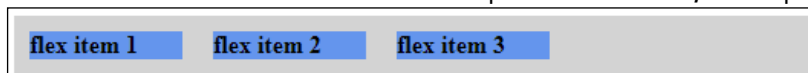


17.1 Propiedades

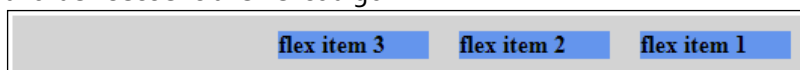
- **flex-direction** → establece el orden de los ítems en el eje principal.

Modificar la propiedad de la regla **.item** para dar un ancho fijo → **width: 100px;**
La prueba de estas propiedades se hace sobre la clase **.contenedor** que tiene la propiedad **display: flex**.

- **row** → los elementos se visualizan de izquierda a derecha, es la opción por defecto.



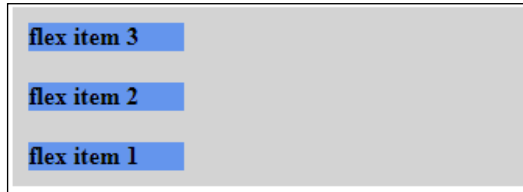
- **row-reverse** → los elementos se visualizan de derecha a izquierda, en orden inverso al orden secuencial en el código HTML.



- **column** → Los elementos se visualizan de arriba hacia abajo.



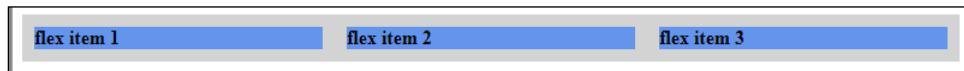
- **column-reverse** → Los elementos se visualizan de abajo hacia arriba.



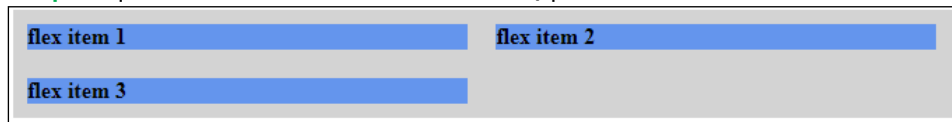
- **flex-wrap** → define la forma en que se distribuyen los elementos en la fila cuando la suma de los anchos correspondientes a los ítems supera la anchura del contenedor.

Modificar la propiedad de la regla `.item` para dar un ancho relativo al total, con el objetivo de que no quepan todos los ítems a lo ancho → `width: 35%;`

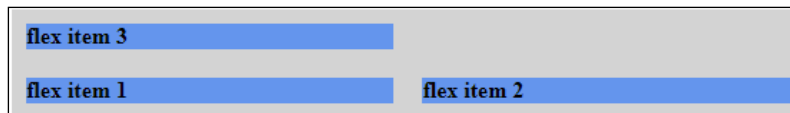
- **nowrap** → (valor por defecto), la anchura de los ítems se adaptan al tamaño total del contenedor.



- **wrap** → que si los ítems no caben a lo ancho, pueden cambiar de línea.



- **wrap-reverse**: igual que **wrap** pero las líneas de elementos flex aparecen ordenadas en sentido contrario.

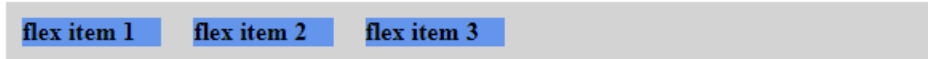


Más ejemplos de flex-wrap

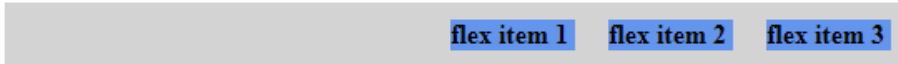
- **justify-content** → define la forma de posicionarse los ítems en el eje principal. Esta propiedad se asigna al contenedor.

Modificar la propiedad de la regla `.item` para dar un ancho fijo → `width: 75px;` y las propiedades `.contenedor {display: flex; background-color: lightgrey;}`

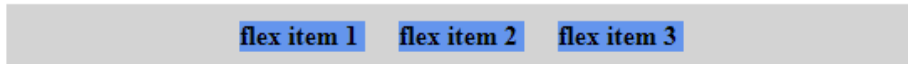
- **flex-start** → (valor por defecto) Alineación a la izquierda de los ítems.



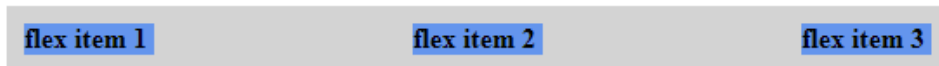
- **flex-end** → Alineación a la derecha de los ítems.



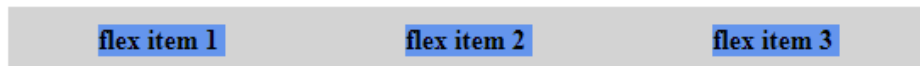
- **center** → Alineación central de los ítems.



- **space-between** → Alineación justificada de los ítems.



- **space-around** → Distribuye de forma uniforme los ítems en el eje principal.

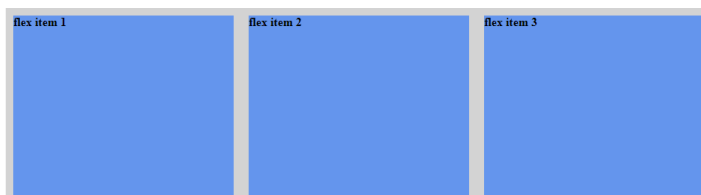


- **align-items** → define la forma de posicionarse los ítems en el eje secundario. **Actualizar la regla de estilo `.contenedor` con los valores siguientes:**

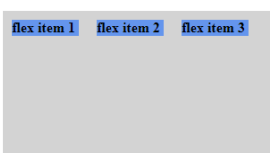
```
.contenedor {
  display: flex; height: 240px;
  background-color: lightgrey;}
```

```
.item {
  background-color: cornflowerblue;
  width: 33%; margin: 10px;}
```

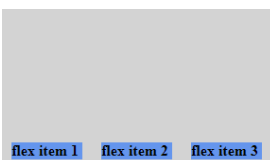
- **stretch** → (valor por defecto), la altura de los elementos se ajusta al tamaño del contenedor (o fila), los ítems adaptan su altura a la del contenedor; funciona siempre que los ítems no tengan especificada una altura con la propiedad `height`.



- **flex-start** → alinea los elementos a la parte superior del contenedor, respetando la altura definida de los ítems.



- **flex-end** → alinea los elementos a la parte inferior del contenedor, respetando la altura definida de los ítems.

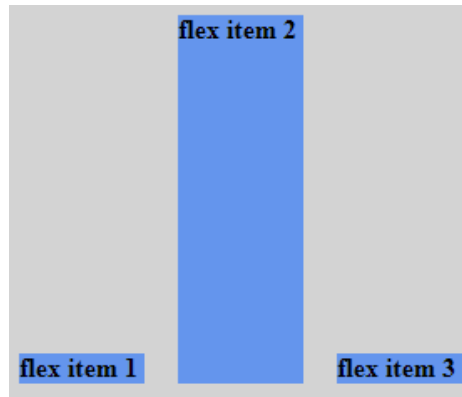


- **align-self** → define una particularidad del posicionamiento en el eje secundario, que varía lo definido en *align-items*, es decir, permite alinear elementos de forma independiente.

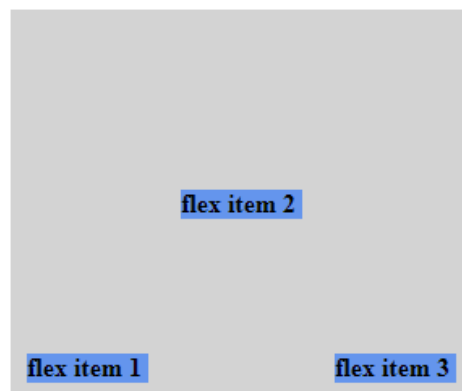
En el ejemplo siguiente suponemos que la propiedad del contenedor **align-items** es **flex-end**, también que el segundo ítem tiene el siguiente identificador, *id="diferente"* y le aplicamos el siguiente estilo cuyo valor de la propiedad **align-self** iremos modificando para comprobar su funcionamiento.

```
#diferente {
  align-self: stretch | center | flex-end | flex-start;
}
```

align-self: stretch



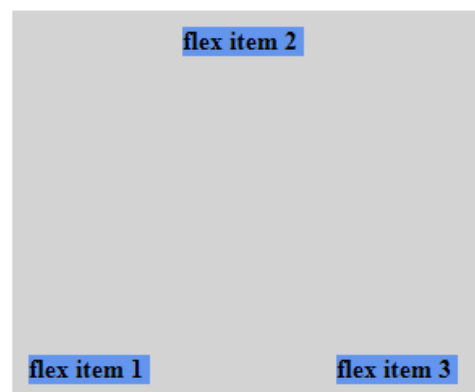
align-self: center



align-self: flex-end

Quedarían todos alineados a la parte inferior.

align-self: flex-start



- **align-content** → solo funciona cuando tenemos más de una fila de elementos, su objetivo es alinear las filas de ítems de un contenedor.

Elimina el estilo #diferente y actualiza la regla de estilo .contenedor con los siguientes valores:

```
.contenedor {  
  display: flex;  
  background-color: lightgrey;  
  flex-wrap: wrap;  
  height: 240px;}
```

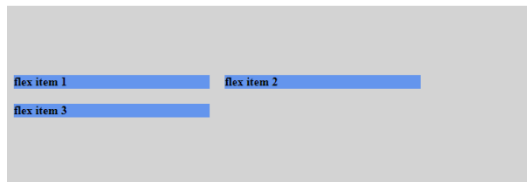
align-content: stretch

(valor por defecto)

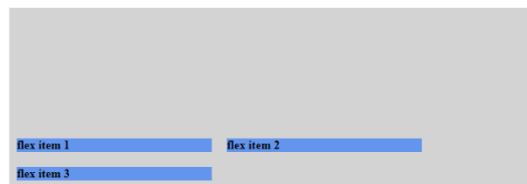
Ocupa todo la altura y distribuye



align-content: center



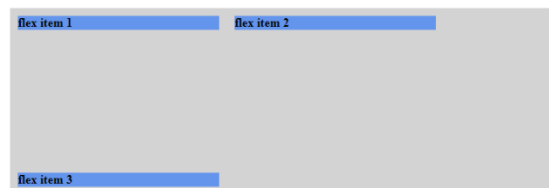
align-content: flex-end



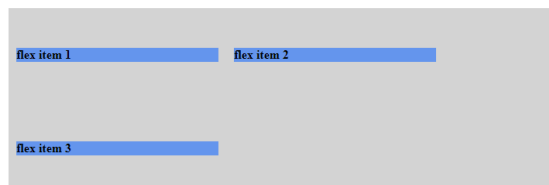
align-content: flex-start



align-content: space-between



align-content: space-around

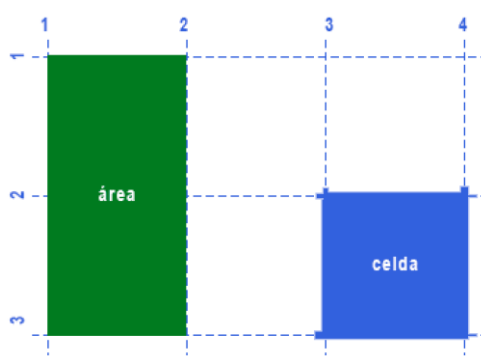


En los siguientes enlaces podrás probar código con flex:

- <https://yoksel.github.io/flex-cheatsheet/>
- <https://flexboxfroggy.com/#es>

18 POSICIONAMIENTO CON CSS3 GRIDAREAS

Una **GridArea** permite estructurar, organizar y diseñar los elementos de una página Web en una serie de celdas similares a una tabla.



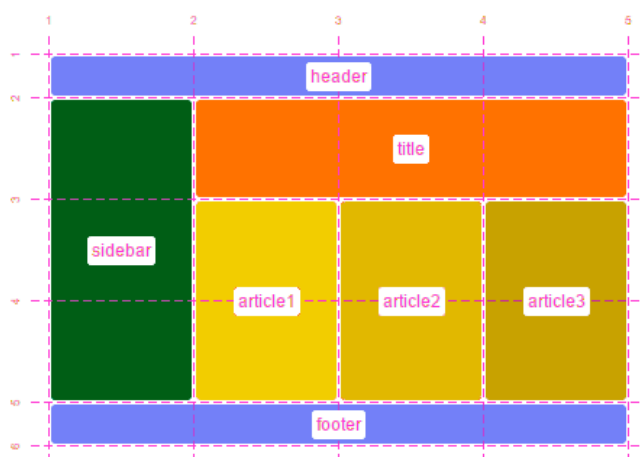
Los elementos de una **GridArea** son:

- **Grid Container:** Es el elemento contenedor donde se indica que se va a crear un grid (**display:grid**).
- **Grid Item:** Es un elemento que depende del *Grid Container* y se corresponderán con las filas y las columnas que contiene.
- **Grid Line:** En el caso de la imagen que hemos tomado como ejemplo hay 4 líneas verticales y 3 líneas horizontales. Por defecto, las líneas reciben números que comienzan desde 1. Las líneas verticales se muestran de izquierda a derecha, pero esto

depende de la dirección de escritura. Para facilitar la referencia a las líneas se les pueden asignar nombres que facilitan su referencia en CSS.

- **Grid Track:** Es el espacio entre dos líneas paralelas. En el ejemplo tenemos 3 tracks verticales y 2 tracks horizontales que forman tres columnas y dos filas.
- **Grid Area:** Es una o más celdas contiguas en horizontal o en vertical. En el ejemplo es un área que a unido dos celdas de distintas filas.
- **Grid Cell:** Una celda es donde se encuentran una pista horizontal y una vertical. En la figura anterior, solo se ha resaltado una celda, pero hay 6 celdas en la cuadrícula.

Con estos elementos podemos determinar los componentes de una página como se puede ver en la imagen.



18.1 Determinar el número de filas y columnas

En las propiedades del contenedor se utiliza:

- **grid-template-columns** para indicar el número de columnas.
- **grid-template-rows** para indicar el número de filas.

El número de filas/columnas se corresponderá con el número de elementos definidos en cada caso. Así, si indicamos **grid-template-columns: tamaño1, tamaño2;** definiendo 2 columnas y de forma equivalente se definiría con las filas.

18.2 Tamaños

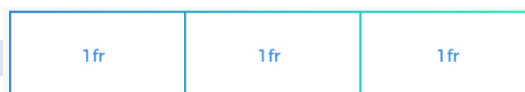
A la hora de crear una cuadrícula podemos utilizar porcentajes y valores fijos, utilizando pixeles y otras de las unidades de medida vista anteriormente.

Existe una unidad de medida específica **fr** (*fracción*) que representa una fracción del espacio disponible en el contenedor del *grid*.

Por ejemplo, para un **display: grid;** vemos distintos casos a continuación:

Caso1

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



Caso2



CSS 3

```
grid-template-columns: 2fr 1fr 1fr;
```

Caso3

Archivo CSS

```
.contenedor{
  display: grid;
  grid-template-columns: 200px 2fr 1fr;
  height: 500px;}
.item{
  background-color: #b4b8b9;
  border: 1px solid #ffffff;
  /*rem→ tamaño de letra del
  elemento raíz*/
  padding: 1rem;
  font-family: arial;
  color: white;
  text-align: center;}
```



Body do arquivo HTML

```
<div class="contenedor">
  <div class="item">item 1</div>
  <div class="item">item 2</div>
  <div class="item">item 3</div>
  <div class="item">item 4</div>
  <div class="item">item 5</div>
  <div class="item">item 6</div>
</div>
```

Caso4

En el caso de querer **repetir una medida** un número de veces en las filas o las columnas utilizamos **repeat (num-veces, medida)**:

Para definir una grilla de 4 **columnas** que ocupen cada una **1fr**.

```
grid-template-columns: repeat(4, 1fr);
```



Ejemplo: Suponemos las siguientes capas en un documento Web:

```
<div class="contenedor">
  <div class="item" id="item1">item 1</div>
  <div class="item" id="item2">item 2</div>
  <div class="item" id="item3">item 3</div>
  <div class="item" id="item4">item 4</div>
  <div class="item" id="item5">item 5</div>
</div>
```

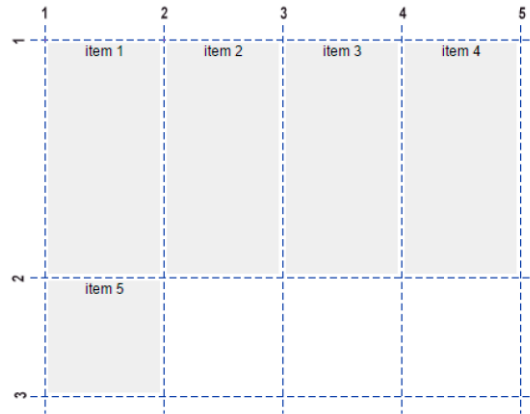
Y definimos, inicialmente, la siguiente regla de estilo para el **contenedor**, dejando la regla para **.item** como no anterior ejemplo.

```
.contenedor {
  display: grid;
  grid-template-rows: 200px 100px;
  /* Definiremos una cuadrícula que tiene 2 filas y 4 columnas.
  repeat () permite definir 4 veces 100px 100px 100px 100px.*/
```



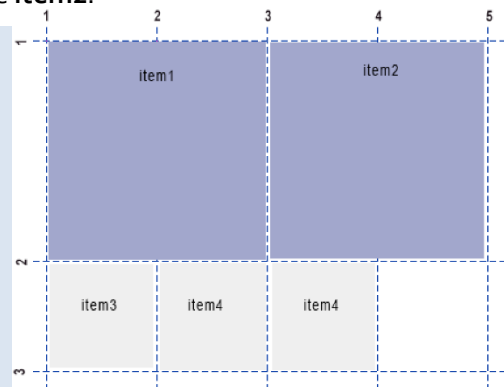
```
grid-template-columns: repeat(4, 100px);
```

Con esta regla definimos una grilla como la siguiente:



A continuación determinamos la posición de **item1** e **item2**.

```
#item1 {  
  /*área que tomará item1*/  
  grid-row-start: 1;  
  grid-row-end: 2;  
  grid-column-start: 1;  
  grid-column-end: 3; }  
#item2 {  
  /*área que tomará item2*/  
  grid-row-start: 1;  
  grid-row-end: 2;  
  grid-column-start: 3;  
  grid-column-end: 5; }
```



18.3 Plantillas de área

Ahora, vamos a ver como organizar la grilla utilizando plantillas de áreas. Para ello, creamos una nueva página Web con el siguiente contenido:

```
<div class="contenedor">
  <div class="header">header</div>
  <div class="sidebar">sidebar</div>
  <div class="footer">footer</div>
  <div class="contenido">
    <p>Lorem ipsum dolor sit amet...</p>
  </div>
</div>
```

Creamos una regla de estilo de **.contenedor** con las siguientes características:

- Definimos 3 filas, la superior e inferior tienen 100 px de alto, mientras que la del medio se adaptará a su contenido.
- Definimos 2 columnas, la izquierda de 100px de ancho y la derecha se adaptará a su contenido.
- Determinamos la denominación de las áreas en función de las cuadrículas establecidas, así el *header* y el *footer* ocupará dos celdas (se repite la denominación del área dos veces)

```
.contenedor {
  display: grid;
  grid-template-rows: 100px auto 100px;
  grid-template-columns: 100px auto;
  grid-template-areas:
    "header header"
    "sidebar contenido"
    "footer footer";
}
```

header	header
sidebar	contenido
footer	footer

También se pueden definir la posición de las áreas utilizando puntos, uno por cada celda, así, para determinar que un área (*area1*) se sitúe a la mitad de una grilla de 3x3 pondríamos la siguiente regla:

```
.contenedor {
  display: grid;
  grid-template-rows: repeat(3, 100px);
  grid-template-columns: repeat(3, 100px);
  grid-template-areas:
    ". . ."
    ". area1 ."
    ". . .";
}
```

Para que funcione la plantilla tenemos que enlazar cada área con su definición en el contenedor:

```
.header { grid-area: header; }
.sidebar { grid-area: sidebar; }
.footer { grid-area: footer; }
.contenido { grid-area: contenido; }
```

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

Si añadimos un texto en el body al contenido, tendremos una vista en pantalla similar a la siguiente (*sin las líneas*)

Un diseño alternativo podría ser:

```
.contenedor {  
  display: grid;  
  grid-template-rows: auto;  
  grid-template-columns: 100%;  
  grid-template-areas:  
    "header"  
    "sidebar"  
    "contenido"  
    "footer";  
}
```

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

Ejercicio

Practicar con las posibilidades que da el siguiente [enlace](#).

19 DISEÑO RESPONSIVE

Las técnicas de diseño Web que se adaptan a los distintos tamaños de pantalla y dispositivos se denomina **diseño responsive**.

Para conseguir un diseño *responsive* podemos utilizar las **Media Queries**, que son los elementos CSS que se utilizan para detectar el tipo de dispositivo en el que se está mostrando una página Web.

Existen dos formas de implementar las **Media Queries**

1. **Utilizando la etiqueta link** de HTML e indicando la condición en la que se ejecuta la hoja de estilos referenciada.

A continuación, veremos distintos ejemplos:

- a. Si queremos que la hoja de estilos *estilos.css* se active cuando se muestre en una pantalla que tenga un **tamaño máximo de 768px**:

```
<link rel="stylesheet" media="(max-width: 768px)"  
href="estilos.css">
```

- b. Si queremos que la hoja de estilos *estilos.css* cuando el navegador tenga un **mínimo de 640 px** de ancho:

```
<link rel="stylesheet" media="(min-width: 640px)"  
href="max-640px.css">
```

- c. Hoja de estilos para dispositivos según su orientación:

```
<link rel="stylesheet" media="(orientation:portrait)"  
href="portrait.css">  
<link rel="stylesheet" media="(orientation:landscape)"  
href="landscape.css">
```

Este método tiene el inconveniente que cambiar de tamaño se deberá cargar otra hoja de estilos, esto conlleva invertir un tiempo en otra solicitud HTTP al servidor.

2. **Implementando las propiedades Media Queries** en la hoja de estilo utilizando la anotación **@media**.

Ejemplo1: Esta **Media Query** se ejecutará sólo cuando la anchura de la ventana del navegador sea menor de 320 píxeles.

```
@media (max-width: 320px) { /*Aquí van todos los estilos CSS */ }
```

Ejemplo2: Esta Media Query se ejecutará sólo cuando la anchura de la ventana del navegador sea mayor de 768 píxeles.

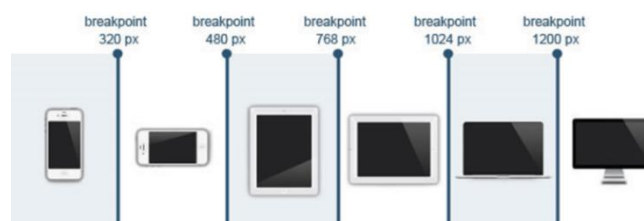
```
@media (min-width:768px) {  
/*Aquí van todos los estilos CSS */  
}
```

Ejemplo3: Esta Media Query se ejecutará para la **orientación horizontal**.

```
@media (orientación: landscape) {  
/*portrait es la orientación vertical */  
}
```

Para saber más.

- https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_col-s
- <https://aukera.es/blog/diseño-responsive/>



Práctica las distintas posibilidades del diseño responsive en el siguiente enlace.

20 VIEWPORT

Viewport es el área visible del usuario de un sitio web. La etiqueta **meta viewport** controla el ancho de la página dependiendo del dispositivo en el que se encuentre el usuario.

Si no se utiliza **Viewport** en una Web, cuando ésta se muestra en los dispositivos móviles, la Web se escalará para ajustarlo al tamaño de pantalla y el texto y los gráficos se verán miniaturizados.

Al utilizar **Viewport**, la visualización de la Web en los dispositivos móviles se ajusta, mostrando menos información en cada pantalla pero manteniendo el tamaño de los elementos ajustada de forma más adecuada.

En la siguiente imagen se puede ver como en la primera pantalla de móvil la página se ve con todos sus elementos excesivamente pequeños y en la segundo, que tiene **Viewport**, el contenido está más adaptado.



Para adaptar el contenido a los dispositivos móviles todas las páginas Web deben contener la siguiente etiqueta en su **head**.

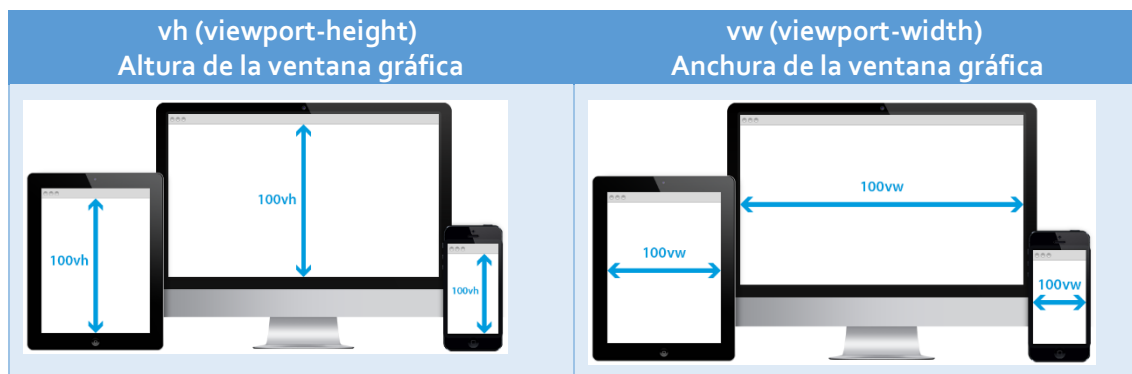
```
<meta name="viewport" content="width=device-width,initial-scale=1.0" />
```

Se puede probar el funcionamiento de una Web en los dispositivos móviles utilizando el siguiente [enlace](#).

También se pueden utilizar medidas relativas al tamaño de la pantalla en la que se muestre el dispositivo, estas unidades de medidas son **vh**, **vw**, **vmin** y **vmax**.

vh hace referencia a la centésima parte de la altura del *viewport* y **vw** a la centésima parte del ancho del *viewport*.

- **1vh** = 1% de la altura del viewport
- **100vh** = altura del viewport
- **1vw** = 1% del ancho del viewport
- **100vw** = ancho del viewport



Puedes ver distintos ejemplos en el siguiente [enlace](#).