

UT2. UTILIZACIÓN DE LENGUAJES DE MARCAS EN ENTORNOS WEB II ESTILOS (CSS)

1º CURSO CFGS DAW

LMSGI

Contenido

1. Introducción.....	2
2. Aplicando estilos.....	3
3. Referencias por nombres.	6
3.1. Referencias por nombres.....	6
3.2. Referencias por el atributo id	10
3.3. Referencia por el atributo class.....	11
3.4. Referencias por otros atributos.	13
3.5. Referencias por pseudoclasas.....	15
4. Propiedades.....	17
5. Espacio ocupado por los elementos.....	20
6. Fondo.	21
7. Bordes.	23
8. Sombras.	24
8.1. Sombras de cajas	24
8.2. Sombras en el texto.	25
9. Gradientes.	26
10. Filtros.....	29
11. Transformaciones	31
12. Transiciones	34
13. Animaciones	35
14. Posicionamiento.....	36
14.1. Repaso a los elementos estructurales de HTML5	36
14.2. Modelo de caja tradicional.....	37
14.3. Posicionamiento static y relative	39
14.4. Posicionamiento absolute (absoluto).....	41
14.5. Posicionamiento fixed (fijado).....	44
14.6. Posicionamiento sticky	45
14.7. Contenido flotante	46
14.8. Cajas flotantes	49
14.9. Cajas y posicionamiento (un ejemplo práctico)	52
14.10. Ejemplo de posicionamiento float-absolute-relative	54
14.11. Propiedad z-index.....	57
14.12. Columnas.	59
15. Display.....	62
15.1. Flexbox.	63

1. Introducción

Hasta ahora, hemos creado contenido dentro de HTML, pero no hemos visto cómo se debe mostrar.

De esto se encarga **CSS** (siglas en inglés de Cascading Style Sheets) es decir, **Hojas de Estilo en Cascada**.

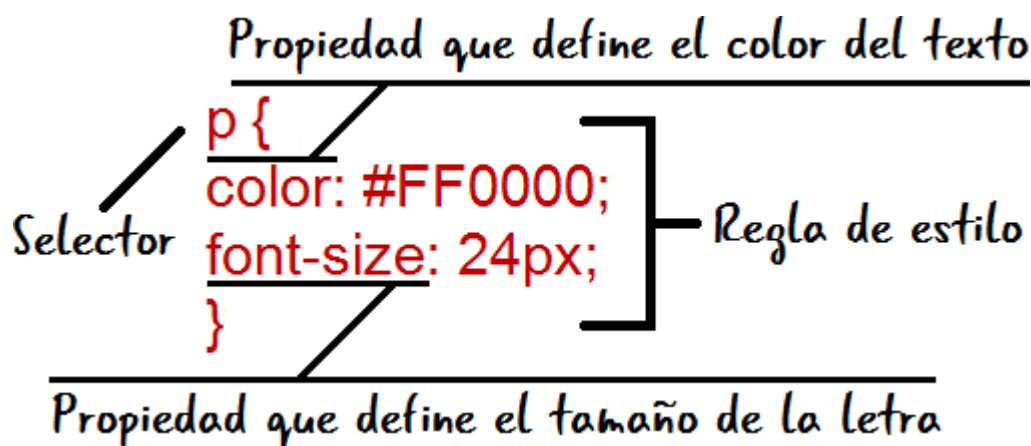
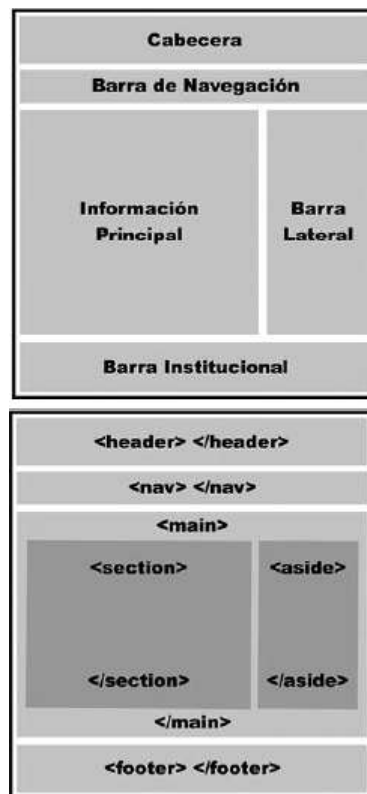
Un estilo se define declarando el nombre de la propiedad y su valor separado por dos puntos (:). Por ejemplo: **font-size:24px;** que cambia el tamaño de la letra a 24 píxeles.

Si queremos cambiar varios estilos, hay que declarar múltiples propiedades en una **regla**.

Una **regla** es una lista de propiedades declaradas **entre llaves** e identificadas por un **selector**.

El **selector** indica qué elementos se verán afectados por las propiedades.

Este primer ejemplo utiliza el **selector p** y, por tanto, afecta a todos los párrafos del documento HTML.



Veamos el siguiente código **sin CSS**.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Ejemplo sin CSS</title>
</head>
<body>
  <header>Esto es la cabecera</header>
  <nav>Principal | Fotos | Videos | Contacto</nav>
  <main>
    <section>
      <article>Mi primer artículo</article>
      <article>Mi segundo artículo</article>
    </section>
    <aside> Cita del artículo uno, dos, tres</aside>
  </main>
  <footer> IES Augustóbriga</footer>
</body>
</html>
```

Esto es la cabecera
Principal | Fotos | Videos | Contacto
Mi primer artículo
Mi segundo artículo
Cita del artículo uno, dos, tres
IES Augustóbriga

Como puedes comprobar, no se produce ningún efecto.

2. Aplicando estilos.

Existen tres técnicas...

Primera.- utilizando el atributo **global** llamado **style** (que hemos venido utilizando en algunos ejemplos hasta ahora).

Veamos el siguiente código:



Segunda.- Incrustando el estilo en la cabecera del documento mediante la etiqueta `<style></style>`. Por ejemplo, si queremos que todos los elementos de `<p>` se vean afectados por `<style>`, debemos definirlo dentro de la cabecera `<head></head>`.



Tercera.- El tercer método son las conocidas como Hojas de Estilos (CSS). Son archivos de texto con la lista de reglas **CSS** que queremos asignar a los elementos del documento HTML.

El elemento `<link>` se usa para incorporar recursos externos al documento, como las hojas de estilo. Para cargar hojas de estilo CSS, sólo necesitamos los atributos **rel** y **href**.

El atributo **rel** significa relación y especifica la relación entre el documento y el archivo que estamos incorporando, por lo que debemos declararlo con el valor

stylesheet para comunicarle al navegador que el recurso es un archivo CSS con los estilos requeridos para representar la página.

Por otro lado, el atributo **href** declara **la URL del archivo** a cargar,

Probemos el siguiente código:

Por un lado creamos el archivo **index.html**

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Lenguaje de marcas</title>

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

  En esta página estamos aplicando<br/>
  <p>
    estilo mediante un archivo CSS
  </p>
</body>
</html>
```

Ahora, creamos el archivo CSS y damos estilo a los elementos. El archivo CSS debe estar alojado según la ruta establecida en el atributo **“href”**.

```
✓ p {
  font-size: 24px;
  color: blue;
}
```

El resultado es:

En esta página estamos aplicando
estilo mediante un archivo CSS

3. Referencias por nombres.

Existen varios métodos para seleccionar los elementos del código HTML para que sean afectados por una regla CSS.

- Referencias por nombres.
- Mediante los atributos globales **id** y **class**.
- Otros métodos específicos.

3.1. Referencias por nombres.

- a. Si utilizamos el nombre del selector, por ejemplo **header**, la regla afectará a todo el contenido en el elemento **header**.

```
header {  
    font-size: 60px;  
    color: #ff0000;  
}
```

- b. Si queremos asignar los **mismos estilos** a elementos con nombres diferentes, declaramos los nombres **separados por coma**.

```
h1, footer {  
    font-size: 48px;  
    color: #0000ff;  
}
```

- c. Podemos referenciar elementos en particular que están incluidos dentro de un elemento, **separando los selectores por un espacio**. Estos tipos de selectores se denominan selectores de descendencia ya que afectan a elementos incluidos en otros.

```
section p {  
    font-size: 24px;  
    color: #ff8000;  
}
```

- d. También es posible hacer una selección más precisa usando el carácter **>** para referenciar un elemento que es hijo de otro.

```
main > p {  
    font-size: 24px;  
    color: #00ff00;  
}
```

- e. Con el carácter **+** se crea otro selector que permite referenciar un elemento que está precedido de otro.

```
h2 + p {  
    font-size: 12px;  
    color: #c00000;  
}
```

- f. Si queremos que la regla afecte a **todos** los elementos que se ubican a continuación de otro, es preciso utilizar el carácter **~** (Alt + 126).

```
h3 ~ p {  
    font-size: 12 px;  
    color: #ff8080;  
}
```


Vamos a ver todo esto con un ejemplo. Creamos el fichero **ejemplocss2.html** con el siguiente código:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Ejemplo2 CSS</title>

  <link rel="stylesheet" href="estilos.css"/>
</head>
<body>
  <header>Esto es la cabecera</header>

  <section>
    <h1>Cabecera de la sección</h1>
    <div>
      <p>Introducimos el primer párrafo</p>
    </div>
    <p>Introducimos un segundo párrafo</p>
  </section>

  <main>
    <h1>Estamos en la cabecera del contenido principal</h1>
    <p>Párrafo 1 del main</p>
    <p>Párrafo 2 del main</p>
    <ul>
      <li>Primera fila de una lista desordenada</li>
      <li>Segunda fila de la lista</li>
      <li>Tercera fila de la lista</li>
      <li>Cuarta fila de la lista</li>
    </ul>
    <span>
      <p>Esto es un párrafo dentro de un span</p>
      <h2>Cabecera 2 dentro del span</h2>
      <p>Otro párrafo dentro del span</p>
      <h3>Cabecera 3 dentro del span</h3>
      <p>y su párrafo correspondiente</p>
      <p>con otro más</p>
    </span>
    <p>Este párrafo sigue bajo la influencia del último
selector...</p>
  </main>

  <footer>IES Augustóbriga en el pie del documento</footer>
```

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

Ahora creamos el fichero de estilo **estilos.css** con la siguiente definición:

```
header {  
    font-size: 60px;  
    color: #ff0000;  
}  
  
h2 + p {  
    font-size: 48px;  
    color: #c00000;  
}  
  
h1, footer {  
    font-size: 48px;  
    color: #0000ff;  
}  
  
section p {  
    font-size: 36px;  
    color: #ff8000;  
}  
  
main p {  
    font-size: 24px;  
    color: #00ff00;  
}  
  
h2 + p {  
    font-size: 48px;  
    color: #c00000;  
}  
  
h3 ~ p {  
    font-size: 12px;  
    color: #ff8080;  
}
```

Hazlo tú y observa los resultados en cada caso según el selector definido.

3.2. Referencias por el atributo **id**

Las reglas anteriores afectan a los elementos del tipo indicado por el selector.

Para seleccionar un elemento HTML sin considerar su tipo, podemos usar el atributo **id**.

Este atributo es un nombre, **un identificador exclusivo** del elemento y, por lo tanto, lo podemos usar para encontrar un elemento en particular dentro del documento.

Para referenciar un elemento usando su **atributo id**, el selector debe incluir el valor del atributo precedido por el carácter numeral o almohadilla (**#**).

Veamos el siguiente ejemplo:

ejemplocss3.html

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Ejemplo 3</title>
  <link rel="stylesheet" href="estilos2.css"/>
</head>
<body>
  <h1>Personajes</h1>
  <p id="principal">
    Don Quijote (caballero andante)
  </p>
  <p>Sancho Panza (escudero)</p>
  <p>Pero Pérez (cura)</p>
  <p>Maese Nicolás (barbero)</p>
</body>
</html>
```

estilos2.css

```
p {  
  font-size: 20px;  
  color: red;  
}  
  
#principal {  
  font-size: 48px;  
  color: green;  
}
```

Resultado.



3.3. Referencia por el atributo class

En lugar de usar el atributo **id** para asignar estilos, en la mayoría de las ocasiones es mejor hacerlo con el atributo **class**. Este atributo es más flexible y se puede asignar a varios elementos dentro del mismo documento.

Para referenciar un elemento usando el atributo **class**, el selector debe incluir el valor del atributo precedido por un punto (.)

Vamos a modificar el ejemplo anterior y probamos esta referencia.

ejemplocss4.html

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Ejemplo 4</title>
  <link rel="stylesheet" href="estilos3.css"/>
</head>
<body>

  <h1 class="principal">Personajes</h1>
  <p class="principal">Don Quijote (caballero andante)</p>
  <p class="principal">Sancho Panza (escudero)</p>
  <p class="secundario">Pero Pérez (cura)</p>
  <p class="secundario">Maese Nicolás (barbero)</p>
  <h1>Otra cabecera</h1>


</body>
</html>
```

estilos3.css

```
.principal {
  font-size: 20px;
  color: red;
}

.secundario {
  font-size: 48px;
  color: green;
}
```

Resultado



Personajes

Don Quijote (caballero andante)

Sancho Panza (escudero)

Pero Pérez (cura)

Maese Nicolás (barbero)



A un mismo elemento se le pueden asignar varias clases. Lo que tenemos que hacer es declarar los nombres de las clases separados por espacio.

```
<p class="principal otraclase">Don Quijote (caballero andante)</p>
```

Las clases también se pueden declarar como exclusivas para un tipo específico de elementos declarando el nombre del elemento antes del punto. Por ejemplo:

```
p.mitexto {
    font-size: 20px;
}
```

3.4. Referencias por otros atributos.

Lo comentado hasta ahora, en ocasiones, es insuficiente para encontrar un elemento exacto.

CSS también permite referenciar un elemento a través de cualquier otro atributo.

La sintaxis para definir esta clase de selectores incluye el **nombre del elemento** seguido del **nombre del atributo entre corchetes**.

Partiremos del siguiente código:

```
<p name="el protagonista">Don Quijote</p>
<p name="elescuero">Sancho Panza</p>
<p name="el cura">Pero Pérez</p>
<p name="elbarbero">Maese Nicolás</p>
<p>Sansón Carrasco</p>
```

Con esta regla	el navegador muestra	
p[name] { font-size: 20px; color:#ff0000; }	Don Quijote Sancho Panza Pero Pérez Maese Nicolás Sansón Carrasco	Los elementos <p> que tienen un atributo llamado "name" .
p[name="elescuero"] { font-size: 20px; color:#ff0000; }	Don Quijote Sancho Panza Pero Pérez Maese Nicolás Sansón Carrasco	Los elemento <p> que tienen un atributo llamado name que posee el valor "elescuero" .
p[name~="el"] { font-size: 20px; color:#ff0000; }	Don Quijote Sancho Panza Pero Pérez Maese Nicolás Sansón Carrasco	~= referencia cualquier elemento <p> cuyo atributo name incluye la palabra "el" .

<pre>p[name^="el"] { font-size: 20px; color:#ff0000; }</pre>	<p>Don Quijote Sancho Panza Pero Pérez Maese Nicolás Sansón Carrasco</p>	<p>^= referencia cualquier elemento <p> cuyo atributo name comienza por “el”.</p>
--	--	---

p[name\$="ero"] { font-size: 20px; color:#ff0000; }	Don Quijote Sancho Panza Pero Pérez Maese Nicolás Sansón Carrasco	\$= referencia cualquier elemento <p> con un atributo name cuyo valor termina en "ero"
p[name*="cu"] { font-size: 20px; color:#ff0000; }	Don Quijote Sancho Panza Pero Pérez Maese Nicolás Sansón Carrasco	*= referencia cualquier elemento <p> con un atributo name cuyo valor contiene la cadena de caracteres "cu"



Prueba lo visto anteriormente.

3.5. Referencias por pseudoclases

Las pseudoclases son herramientas especiales de CSS que nos permiten referenciar elementos HTML por medio de sus características, como sus **posiciones** en el código o sus **condiciones actuales**.

Veamos algunas de las más utilizadas.

:nth-child(valor).- Selecciona un elemento de una lista de elementos hermanos (hijos de otro elemento) que se encuentra en la posición especificada por el valor entre paréntesis.

:first-child.- Selecciona el primer elemento de una lista de elementos hermanos.

:last-child.- Selecciona el último elemento de una lista de elementos hermanos.

:only-child.- Selecciona un elemento cuando es el único hijo de otro elemento.

:not(selector).- Selecciona los elementos que no coinciden con el selector entre paréntesis.

Veamos el siguiente código:

ejemplocss5.html

Código HTML con el que probaremos las pseudoclases anteriores

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <title>hoja 04</title>
    <meta charset="utf-8"/>
    <link rel="stylesheet" href="04_estilos.css"/>
  </head>
  <body>
    <main>
      <h1>Refranes</h1>
      <p class="refran_1">Más vale pájaro en mano que buitres volando.</p>
      <p class="refran_2">Más vale un tomo que dos te daré.</p>
      <p class="refran_3">El consejo de la mujer es poco, y el que no le toma es loco.</p>
      <p class="refran_4">Cuando a Roma fueres, haz como vieres.</p>
    </main>
```



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

...y probamos con las siguiente reglas:

Con este CSS...	El navegador muestra
<pre>p:nth-child(2) { background-color: #f8c471; } p:nth-child(3) { background-color: #fad7a0; } p:nth-child(4) { background-color: #fdebd0; } p:nth-child(5) { background-color: #fef5e7; }</pre>	<p>Primero mira el código. El primer párrafo es el segundo “hijo” del elemento “padre” (main).</p> <h2>Refranes</h2> <p>Más vale pájaro en mano que buitre volando.</p> <p>Más vale un toma que dos te daré.</p> <p>El consejo de la mujer es poco, y el que no le toma es loco.</p> <p>Cuando a Roma fueres, haz como vieres.</p>

Con este CSS...	El navegador muestra
<pre>p:nth-child(odd) { background-color: #f8c471; } p:nth-child(even) { background-color: #fdebd0; }</pre>	<p>La palabra clave odd para la seudoclase :nth-child() afecta a los elementos <p> que son hijos de otro elemento y tienen un índice impar, y la palabra clave even afecta a aquellos que tienen un índice par.</p> <p>Usando estas palabras clave, solo necesitamos dos reglas para configurar la lista completa.</p> <p>Seguro que ya estarás pensando en algo así: tr:nth-child(odd) y tr:nth-child(even) para los elemento de una tabla, por ejemplo.</p> <h2>Refranes</h2> <p>Más vale pájaro en mano que buitre volando.</p> <p>Más vale un toma que dos te daré.</p> <p>El consejo de la mujer es poco, y el que no le toma es loco.</p> <p>Cuando a Roma fueres, haz como vieres.</p>
<pre>p:last-child { background-color: #f8c471; }</pre>	<p>Las seudoclases :first-child, :lastchild y :only-child tienen un efecto sobre el primer hijo de otro elemento, el último o si es único hijo de otro elemento.</p>

	<h2>Refranes</h2> <p>Más vale pájaro en mano que buitre volando.</p> <p>Más vale un toma que dos te daré.</p> <p>El consejo de la mujer es poco, y el que no le toma es loco.</p> <p>Cuando a Roma fueres, haz como vieres.</p>
<pre>p:not(.refran_2) { margin: 20px; background-color: #f8c471; }</pre>	<p>Con la seudoclase :not podemos seleccionar elementos que no coinciden con un selector.</p> <p>Esta regla afecta a todos los párrafos que no son de la clase "refrán_2".</p> <h2>Refranes</h2> <p>Más vale pájaro en mano que buitre volando.</p> <p>Más vale un toma que dos te daré.</p> <p>El consejo de la mujer es poco, y el que no le toma es loco.</p> <p>Cuando a Roma fueres, haz como vieres.</p>

4. Propiedades

Las propiedades son la pieza central de las CSS. Todos los estilos que podemos aplicar a un elemento se definen por medio de propiedades. En los ejemplos que hemos visto hemos trabajado con algunas de ellas pero hay muchas más. Podemos hacer la siguiente clasificación: propiedades de **formato** y propiedades de **diseño**.

- **De formato:** Dan formato a los elementos y su contenido. Tipo de letra, borde, color de fondo, etc.
- **De diseño:** Enfocadas a determinar la posición de los elementos en la pantalla.

En este apartado vamos a ver las propiedades de formato. Partiremos del siguiente código de ejemplo.

ejemplopropiedades1.html

```
<!DOCTYPE html>  
<html lang="es">  
  <head>  
    <title>Lenguaje de marcas</title>  
    <meta charset="utf-8"/>  
    <link rel="stylesheet" href="estilosprop1.css"/>  
  </head>  
  <body>
```

```
<div id="uno">
  CAPÍTULO 1: Que trata de la condición y ejercicio del famoso hidalgo
    D. Quijote de la Mancha
</div>

<div id="dos">
  En un lugar de la Mancha, de cuyo nombre no quiero acordarme,
    no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero
</div>

</body>
</html>
```

estilosprop1.css

```
#uno {
  font-size: 24px;
  color: #ff0000;
}
```

Observa en el código HTML que div (división) es un contenedor general. En este caso a cada una de las dos divisiones se les aplica el atributo global id para distinguirlos.

Texto

font-family.- Se pueden declarar múltiples valores separados por comas. Algunos de los valores estándar son Arial, Helvetica, "Times New Roman". Entre comillas los nombres compuestos.

font-size.- Determina el tamaño de la letra. Normalmente en píxeles (px). El **valor por defecto es 16px.**

font-weight.- Propiedad que determina si el texto se mostrará en negrita o no. Valores: **normal y bold.**

font-style.- Determina el estilo de la letra. Los valores disponibles son **normal, italic, y oblique.**

font.- Permite declarar **múltiples valores** al mismo tiempo separados por un espacio y **orden preciso**, el inverso al descrito. Por ejemplo, **font: bold 24px Arial, sans-serif;**



Elabora para la segunda división del código anterior una nueva regla de estilo. Prueba en ella las propiedades que se acaban de describir.

También existen propiedades para cambiar otros aspectos como **alineamiento, sangría, etc.**

text-align.- Propiedad que alinea el texto. Los valores disponibles son **left, right, center, y justify.**

letter-spacing.- Define el espacio entre letras. Se declara en píxeles (px), porcentaje (%), etc.

word-spacing.- Define el ancho del espacio entre palabras. Igual que la propiedad anterior.

line-height.- Define el espacio entre líneas. Igual que la propiedad anterior.

text-decoration.-Esta propiedad resalta el texto con una línea. Los valores disponibles son underline, overline, line-through, y none. Útil para eliminar la línea de los enlaces, **none**.



Añade al documento anterior una nueva división (**div="tres"**) y crea otra regla para que puedas probar estas propiedades. **Pero ve probando propiedad a propiedad.**



Añade dentro de la división **id="dos"** un enlace a Google, por ejemplo. Añade tres nuevas reglas que referencien el elemento **a** para que el enlace, enlace visitado y para cuando posiciones el cursor en él contengan **text-decoration:none**; y el color que desees. Observa estas nuevas reglas

```
#dos a:link{  
text-decoration:none;  
color:#ff0000;  
}
```

```
#dos a:visited{  
text-decoration:none;  
color:#ff0000;  
}
```

```
#dos a:hover{  
text-decoration:none;  
color:#800000;  
}
```

Color

Las formas más habituales de definir el color es usando una combinación de tres colores básicos: rojo, verde y azul. Las propiedades más utilizadas son las que se definen a continuación:

color.- Esta propiedad es la que hemos venido usando en ejemplos anteriores (#ffff00, amarillo).

rgb(rojo, verde, azul).-Esta función define un color por medio de los valores especificados (de 0 a 255). Por ejemplo, **rgb(153, 102, 51)**.

rgba(rojo, verde, azul, alfa).-Esta función es similar a la función **rgb()**. Incluye un componente para definir la opacidad (alfa), entre 0 (transparente) y 1 (totalmente opaco). Ideal para trabajar con cajas.



Añade en la división (**div="dos"**) la propiedad **background-color** con el valor **#bbffbb**. Después prueba el valor **rgba(187,187,255, 0.5)**. Para esta última propiedad pon una imagen de fondo en la página.

Tamaño

Por defecto, el tamaño de la mayoría de los elementos se determina según el espacio disponible en el contenedor de la información a mostrar. El ancho de un elemento se define como 100 %, lo cual significa que será tan ancho como su contenedor, y tendrá una altura determinada por su contenido.

Esta es la razón por la que el elemento <p>, por ejemplo del HTML con el que hemos estamos practicando se extiende a los lados de la ventana del navegador y no era más alto de lo necesario para contener las líneas del párrafo. Las siguientes propiedades definen un tamaño personalizado:

width.- Esta propiedad declara el ancho de un elemento. El valor se puede especificar en píxeles, porcentaje, o con la palabra auto (por defecto).

height.- Esta propiedad declara la altura de un elemento. El valor se puede especificar en píxeles, porcentaje, o con la palabra auto (por defecto).

Por ejemplo, si a un contenedor se le asigna un espacio menor que el espacio requerido por el contenido, éste desbordará el espacio (**overflow**). Observa la imagen de la derecha. La zona de color (color de fondo) corresponde al contenedor. Lo que está fuera lo ha sobrepasado.

Para tratar este problema puede utilizarse la propiedad **overflow**. A esta propiedad se le pueden asignar tres valores:

- **hidden**.- oculta todo lo que está fuera.
- **scroll**.- muestra barras de desplazamiento.
- **auto**.- muestra las barras de desplazamiento si es necesario.

En esto descubrieron treinta o cuarenta molinos de viento que hay en aquel campo, y así como Don Quijote los vió, dijo a su escudero: la ventura va guiando nuestras cosas mejor de lo que acertáramos a desear; porque ves allí, amigo Sancho Panza, donde se descubren treinta o poco más desaforados gigantes con quien pienso hacer batalla, y quitarles a todos las vidas, con cuyos despojos comenzaremos a enriquecer: que esta es buena guerra, y es gran servicio de Dios quitar tan mala simiente de sobre la faz de la tierra.



Añade una nueva división (**id="cuatro"**) en el documento HTML con una información voluminosa como la de la figura anterior y prueba la siguiente regla de estilo.

```
#cuatro p{
  font-size: 18px;
  color:#000000;
  text-align:justify;
  width:300px;
  height:150px;
  background-color:#d6eaf8;
}
```

Comprueba que la información, si es voluminosa, aparece por debajo del color de fondo asignado al contenedor "p".

Si es así, prueba qué efecto tienen los valores hidden, scroll y auto si utilizas la propiedad **overflow**.

En cierto modo está tocando el concepto de "caja" que trataremos más adelante.

5. Espacio ocupado por los elementos.

Antes de abordar las propiedades asociadas al fondo del documento HTML, no vendrá nada mal hacer una declaración previa.

El **tamaño** ocupado por los elementos descritos en la práctica anterior no sólo queda definido por su **ancho** y **alto**. También queda definido por el tamaño de su **margen, borde y relleno**.



Las propiedades que definen márgenes y rellenos son:

margin.- Esta propiedad declara el margen de un elemento. Puede recibir cuatro valores que representan el margen superior, derecho, inferior e izquierdo, en ese orden y separados por un espacio (por ejemplo, **margin: 10px 30px 10px 30px;**).

Si sólo se declaran uno o dos valores, los otros toman los mismos valores (por ejemplo, **margin:10px 30px** asigna 10 píxeles al margen superior e inferior y 30 píxeles al margen izquierdo y derecho). Los valores se pueden declarar independientemente usando las propiedades asociadas **margin-top**, **margin-right**, **margin-bottom** y **margin-left**. La propiedad también acepta el valor **auto** obligando al navegador a calcular el margen: **margin: 10px auto** centraría el contenido dejando los márgenes superior e inferior en 10 píxeles.

padding.- declara el relleno de un elemento. Este es el espacio entre el contenido del elemento y los límites de su caja. Los valores **se declaran de la misma forma** que lo hacemos para la propiedad margin, aunque también se pueden declarar de forma independiente: **padding-top**, **padding-right**, **padding-bottom** y **padding-left**.

Partimos del siguiente código HTML (**ejemplopropiedades2.html**):

```
<body>

  <div id="capitulo">
    La aventura de los molinos
  </div>

  <div id="texto">
    <p>En este descubrieron treinta o cuarenta molinos de viento
      que hay en aquel campo, y así como Don Quijote los vio, dijo
      a su escudero: la aventura va guiando nuestras cosas mejor de
      lo que acertáramos a desear, porque ves allí, amigo Sancho Panza,
      donde se descubren treinta o poco más desaforados gigantes con
      quien pienso hacer batalla, y quitarles a todos las vidas, con
      cuyos despojos comenzaremos a enriquecer, que esta es buena guerra,
      y es gran servicio de Dios quitar tan mala simiente de sobre la
      faz de la tierra...
    </p>
  </div>

</body>
```

6. Fondo.

Los elementos pueden incluir un fondo que se muestra detrás del contenido del elemento y a través del área ocupada por el contenido y el relleno. Éstas son algunas propiedades:

background-color.- Esta propiedad asigna un fondo de color a un elemento.

background-image.- Asigna la imagen al fondo del elemento. La URL del archivo se declara con la función url() (ejemplo: url("fondo.jpg")).

background-repeat.- Esta propiedad determina cómo se distribuye la imagen de fondo usando cuatro palabras clave: repeat, repeat-x, repeat-y y no-repeat. La palabra clave repeat repite la imagen en el eje vertical y horizontal, mientras que repeat-x y repeat-y lo hacen solo en el eje horizontal o vertical, respectivamente. Finalmente, no-repeat muestra la imagen de fondo una vez.

background.- Esta propiedad permite declarar los atributos del fondo al mismo tiempo.

Crea el siguiente fichero CSS (**estilosprop2.css**) donde se define la propiedad **background-color** además de **margin** y **padding** comentadas anteriormente.

```
#capitulo{
  font: bold 48px Arial, Helvetica, sans-serif;
  color:rgb(123, 34, 34);
  text-align:center;
  width:750px;
  margin:60px auto;
  padding: 25px;
  background-color: #b0b0ef;
}
```

```
#texto {
  font: 20px Arial, Helvetica, sans-serif;
  font-family: ;
  color:#000080;
  text-align: justify;
  width: 650px;
  margin: 10px auto;
  background-color: #b0b0ef;
}
```



Prueba los siguiente códigos

CSS que definen la imagen de fondo que ocupa todo el contenedor del elemento **body** (primer código) y, después, imagen de fondo que sólo se repite en el eje vertical con color de fondo blanco que aparece allí donde no llega la imagen:

CSS	Navegador
<pre>body{ background-image: url("fondo.png"); }</pre>	<p>La aventura de los molinos</p> <p>En esto descubrieron treinta o cuarenta molinos de viento que hay en aquel campo, y así como Don Quijote los vio, dijo a su escudero: la aventura va guiando nuestras cosas mejor de lo que acertáramos a desear, porque ves allí, amigo Sancho Panza, donde se descubren treinta o poco más desaforados gigantes con quien pienso hacer batalla, y quitarles a todos las vidas, con cuyos despojos comenzaremos a enriquecer, que esta es buena guerra, y es gran servicio de Dios quitar tan mala simiente de sobre la faz de la tierra...</p>


```
#texto {  
  font: 20px Arial, Helvetica, sans-serif;  
  color:#000080;  
  text-align:justify;  
  width:650px;  
  margin:10px auto;  
  padding: 10px 60px;  
  background-color: #ffffff;  
  background-image:  
    url("fondo_lat.png");  
  background-repeat: repeat-y;  
}
```



7. Bordes.

Los elementos pueden incluir un borde en los límites de la caja del elemento. Por defecto, los navegadores no muestran ningún borde, pero podemos usar las siguientes propiedades.

border-width.- Esta propiedad define el ancho del borde. Acepta hasta cuatro valores separados por un espacio para especificar el ancho de cada lado del borde (superior, derecho, inferior, e izquierdo, es ese orden). También podemos declarar el ancho para cada lado de forma independiente con las propiedades **border-top-width**, **border-bottom-width**, **border-left-width**, y **border-right-width**.

border-style.- Esta propiedad define el estilo del borde. Acepta hasta cuatro valores separados por un espacio para especificar los estilos de cada lado del borde (superior, derecho, inferior, e izquierdo, en ese orden). Los valores disponibles son **none**, **hidden**, **dotted**, **dashed**, **solid**, **double**, **groove**, **ridge**, **inset**, y **outset**. El valor por defecto es none, lo que significa que el borde no se mostrará a menos que asignemos un valor diferente a esta propiedad. También podemos declarar los estilos de forma independiente con las propiedades **border-top-style**, **border-bottom-style**, **border-left-style**, y **border-right-style**.

border-color.- Esta propiedad define el color del borde. Acepta hasta cuatro valores separados por un espacio para especificar el color de cada lado del borde (superior, derecho, inferior, e izquierdo, en ese orden). También podemos declarar los colores de forma independiente con las propiedades **border-top-color**, **border-bottom-color**, **border-left-color**, y **border-right-color**.

border.- Esta propiedad nos permite declarar todos los atributos del borde al mismo tiempo. También podemos usar las propiedades **border-top**, **border-bottom**, **border-left**, y **border-right** para definir los valores de cada borde de forma independiente.



Prueba sobre el fichero de estilos anterior los bordes.

Si nuestro diseño requiere esquinas redondeadas, podemos agregar la siguiente propiedad.

border-radius.- Esta propiedad define el radio del círculo virtual que el navegador utilizará para dibujar las esquinas redondeadas. Acepta hasta **cuatro valores** para definir los radios de cada esquina (superior izquierdo, superior derecho, inferior derecho e inferior izquierdo, en ese orden).

CSS	Navegador
<code>border: 8px solid #ff0000; border-radius: 20px;</code>	
Si queremos esquinas diferentes:	
<code>border: 5px solid #ff0000; border-radius: 20px 10px 30px 50px;</code>	

8. Sombras.

Otro efecto interesante que podemos aplicar a un elemento son las sombras. CSS incluye las siguientes propiedades para generar sombras para la caja de un elemento y para formas irregulares como texto.

box-shadow.- Esta propiedad genera una sombra desde la caja del elemento. Acepta hasta seis valores. Podemos declarar el desplazamiento horizontal y vertical de la sombra, el radio de difuminado, el valor de propagación, el color de la sombra y también podemos incluir el valor **inset** para indicar que la sombra deberá proyectarse dentro de la caja.

text-shadow.- Esta propiedad genera una sombra en un texto. Acepta hasta cuatro valores: desplazamiento horizontal y vertical, el radio de difuminado y el color de la sombra.

8.1. Sombras de cajas

La propiedad **box-shadow** necesita al menos **dos valores (opción de un tercero)** para poder determinar el color y el desplazamiento de la sombra. El desplazamiento puede ser positivo o negativo. Los valores indican la distancia horizontal y vertical desde la sombra al elemento: los valores negativos posicionan la sombra a la izquierda y encima del elemento, mientras que los valores positivos crean una sombra a la derecha y debajo del elemento. **El tercer valor del elemento ofrece la posibilidad de generar un efecto de difuminado sobre la sombra.** El siguiente ejemplo agrega una sombra básica a la cabecera de nuestro documento.



Continuamos con el ejemplo anterior, añadimos y probamos la propiedad **box-shadow** a nuestra hoja de estilos.



CSS	Navegador
<pre>#capitulo{ font: bold 48px Arial, Helvetica; color:#000080; text-align:center; width:750px; margin: 60px auto; padding: 25px; background-color: #bbbbbf; border: 5px solid #ff0000; border-radius: 20px 10px 30px 50px; box-shadow: #808080 10px; }</pre>	<p>El código genera una sombra sólida de color gris y diez píxeles de grosos sobre la caja que contiene el título.</p> 
<pre>box-shadow: #808080 10px 5px;</pre>	<p>Para que la sombra parezca más real podemos añadir una distancia de difuminado. Luego, haz pruebas con otros valores.</p> 
<pre>box-shadow: #808080 10px 5px inset;</pre>	<p>Con la palabra clave inset se crea la sombra interna. Provocando sensación de profundidad.</p> 



8.2. Sombras en el texto.

Finalizamos aplicando sombras al texto con la propiedad **text-shadow**. Para poder apreciar mejor los conceptos utilizaremos un texto de gran tamaño para observar los efectos. Añadimos una nueva caja

```
<div id="final">
  FIN
</div>
```

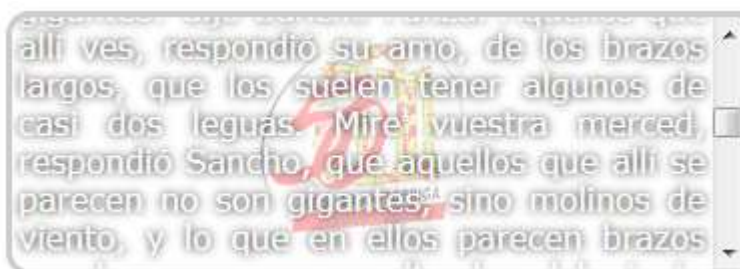
...con el siguiente estilo:

CSS	Navegador
<pre>#final{ width:600px; margin:80px auto; text-align:center; font: bold 120px Arial, Verdana; color:#ff0000; -webkit-text-stroke: 4px black; }</pre>	<p>Si lo que se desea es aplicar una sombra a todo el contorno de la letra (borde), el código más corte que puede utilizarse es el que se muestra a la izquierda.</p> 
<p>Ahora, la propiedad text-shadow establece cinco píxeles de desplazamiento horizontal y vertical. Y de color negro.</p> <pre>text-shadow: 5px #000000;</pre>	

El tercer valor numérico establece el desenfoque . Prueba qué efectos provocan números mayores.	
text-shadow: 5px 4px #000000;	
Si no se provocan desplazamientos, pero sí un desenfoque, aparece un contorno.	
text-shadow: 0px 10px #000000;	



Para terminar, trata de mostrar el texto del capítulo de la manera que muestra la figura. Algunas pistas: letra blanca, sombra sin desplazamiento y desenfoque 4px. En los recursos de la plataforma tienes la imagen del IES Augustóbriga.



9. Gradientes.

Un gradiente se forma mediante una serie de colores que varían continuamente con una transición suave de un color a otro. Se crean como imágenes y se agregan al fondo del elemento con las propiedades **background-image** o **background**. Para crear la imagen con el gradiente, CSS ofrece las siguientes **funciones**:

linear-gradient(posición, ángulo, colores).- Esta función crea un gradiente lineal. El atributo **posición** determina el lado o la esquina desde la cual comienza el gradiente y se declara con los valores **top**, **bottom**, **left** y **right**; el atributo **ángulo** define la dirección del gradiente y se puede declarar en las unidades deg (grados), grad (gradianes), rad (radianes) o turn (espiras), y el atributo **colores** es la lista de colores que participan en el gradiente separados por comas. Los valores para el atributo **colores** pueden incluir un segundo valor en porcentaje separado por un espacio para indicar la posición donde finaliza el color.

radial-gradient(posición, forma, colores).- Esta función crea un gradiente radial. El atributo **posición** indica el origen del gradiente y se puede declarar en **píxeles**, en **porcentaje**, o por medio de la combinación de los valores **center**, **top**, **bottom**, **left** y **right**, el atributo **forma** determina la forma del gradiente y se declara con los valores **circle** y **ellipse** y el atributo **colores** es la lista de los colores que participan en el gradiente separados por comas. Igualmente, pueden incluir un segundo valor en porcentaje por color, separado por un espacio para indicar la posición donde finaliza el color.



Sobre el HTML y CSS de la hoja anterior probaremos y analizaremos estos códigos CSS aplicados al selector “**titulo**”.

CSS	Navegador
<pre>#capitulo{ font: bold 48px Arial, Helvetica; color:#000080; text-align:center; width:750px; margin: 60px auto; padding: 25px; background-color: #bbbbff; border: 5px solid #000080; border-radius: 20px; background: -moz-linear-gradient(top, #ffffff, #aaaaff); }</pre>	



IMPORTANTE: algunas propiedades y funciones CSS se consideran experimentales. Por esta razón, se deben declarar con un prefijo que representa el motor web utilizado. Por ejemplo, si queremos que la función linear-gradient() funcione en Google Chrome, tenemos que declararla como -webkit-linear-gradient(). Los prefijos requeridos para los navegadores más populares son -moz- para Mozilla Firefox, -webkit- para Safari y Google Chrome.

Ahora, deja estas dos líneas conseguirás que el código sea compatible con ambos navegadores:





```
background:-moz-linear-gradient(top, #ffffff, #aaffaa);
background:-webkit-linear-gradient(top, #ffffff, #aaffaa);
```

```
#capitulo {
  font: bold 48px Arial;
  color:rgb(123, 34, 34);
  text-align:center;
  width:750px;
  margin:60px auto;
  padding: 25px;
  background: _webkit-linear-gradient(top,#ffffff,#aaaaff);
  background:-moz-linear-gradient(top,#ffffff,#aaaaff);
  border:rgb(21, 7, 56) double 5px;
  border-radius: 60px;
  box-shadow: black 10px 30px;
}
```



Vamos a seguir probando códigos para generar gradientes.

¡Te darás cuenta que sin utilizar un programa de edición de imágenes puedes crear cosas muy coloridas!



Trabajando con grados	
CSS	background:-webkit-linear-gradient(45deg, #ffffff, #aaaaff);
Navegador	
Es posible agregar más colores para generar un gradiente multicolor.	
CSS	background: -webkit-linear-gradient(top, #00ff00, #FFFFFF, #000000);
Navegador	
Analiza y prueba este código con grados	
CSS	background: -webkit-linear-gradient(0deg, #00ff00, #FFFFFF, #808080);
Navegador	
Si declaras un color transparente, el gradiente es translúcido y se mezcla con el fondo	
CSS	background: -webkit-linear-gradient(left, transparent, #aaaaff);
Navegador	

Además de **gradientes lineales**, también podemos crear **gradientes radiales**.

La única diferencia es que tenemos que usar la función **radial-gradient()** en lugar de la función linear-gradient() e incluir un parámetro para la forma del gradiente con los valores **circle o ellipse**.



Prueba los siguientes códigos.

Trabajando con grados	
CSS	background: -webkit-radial-gradient(center, ellipse, #ffffff, #aaaaff);
Navegador	
Es posible agregar más colores para generar un gradiente multicolor.	
CSS	background: -webkit-radial-gradient(30px 50px, ellipse, #40ff40 50%, #ffffff 70%, #000000 90%);
Navegador	

Excepto por la forma (círculo o elipse), el resto de esta función trabaja igual que linear-gradient().

La posición del gradiente se puede personalizar y podemos usar varios colores con un segundo valor para determinar los límites de cada uno de ellos.

10. Filtros.

Los filtros agregan efectos a un elemento y su contenido. CSS incluye la propiedad **filter** para asignar un filtro a un elemento y las siguientes **funciones para crearlo**.

blur(valor).-Esta función produce un efecto de difuminado. Acepta valores en píxeles desde 1px a 10px.

grayscale(value).-Esta función convierte los colores de la imagen en una escala de grises. Acepta números decimales entre 0.1 y 1.

drop-shadow(x, y, tamaño, color).-Esta función genera una sombra. Los atributos **x** e **y** determinan la distancia entre la sombra y la imagen, el atributo **tamaño** especifica el tamaño de la sombra, y el atributo **color** declara su color.

sepia(valor).-Esta función le otorga un tono sepia (ocre) a los colores de la imagen. Acepta números decimales entre 0.1 y 1. (o en porcentaje

brightness(valor).-Función que cambia el brillo. Acepta números entre 0.1 y 10.

contrast(valor).-Esta función cambia el contraste de la imagen. Acepta números decimales entre 0.1 y 10.

hue-rotate(valor).-Esta función aplica una rotación de tono. Acepta un valor en grados desde 1deg a 360deg.

invert(valor).—Esta función invierte los colores de la imagen y produce un negativo. Acepta números decimales entre 0.1 y 1.

saturate(valor).-Esta función satura los colores. Acepta números entre 0% y 100%.

opacity(valor).-Esta función cambia la opacidad de la imagen. Acepta números decimales entre 0 y 1 (0 es totalmente transparente y 1 totalmente opaco). Acepta también porcentajes entre 0% y 100%

Estos filtros no solo se pueden aplicar a imágenes, sino también a otros elementos en el documento.



Utiliza el HTML de la hoja anterior y prueba los filtros.

Empezaremos esta primera toma de contacto con el difuminado (**blur**).

CSS	Sin filtro
<pre>#capitulo{ font: bold 48px Arial, Helvetica, sans-serif; color:#000080; text-align:center; width:750px; margin: 60px auto; padding: 25px; border: 5px solid #000080; border-radius: 20px; background: -moz-radial-gradient(0px 0px, ellipse, #ff8080 25%, #ffff00 75%, transparent 90%); filter: blur(4px); }</pre>	
	<pre>filter: blur(2);</pre> 
	<pre>filter: blur(4);</pre> 

Ahora, aplicaremos los filtros sobre el logotipo del instituto.

Añadiremos al HTML el código de la derecha.

Después aplicaremos los filtros a cada división.

```
<div id="filtro1">filter: grayscale(0.9)<br/></div>
<div id="filtro2">filter: sepia(0.7);<br/></div>
<div id="filtro3">filter: brightness(0.7);<br/></div>
<div id="filtro4">filter: hue-rotate(90deg);<br/></div>
<div id="filtro5">filter: invert(0.8);<br/></div>
<div id="filtro6">filter: saturate(0.3);<br/></div>
<div id="filtro7">filter: opacity(0.3);<br/></div>
```

Imagen sin filtro



CSS

```
#filtro1{
width:175px;
margin: 15px auto;
filter: grayscale(0.9);
}
```

filter: grayscale(0.9);



```
#filtro2{
width:175px;
margin: 15px auto;
filter: sepia(0.7);
}
```

filter: sepia(0.7);



```
#filtro3{
width:175px;
margin: 15px auto;
filter: brightness(0.7);
}
```

filter: brightness(0.7);



```
#filtro4{
width:175px;
margin: 15px auto;
filter: hue-rotate(90deg);
}
```

filter: hue-rotate(90deg);



<pre>#filtro5{ width:175px; margin: 15px auto; filter: invert(0.8); }</pre>	
<pre>#filtro6{ width:175px; margin: 15px auto; filter: saturate(30%); }#</pre>	
<pre>filtro7{ width:175px; margin: 15px auto; filter: opacity(30%); }</pre>	

11. Transformaciones

Una vez que se crean los elementos HTML, estos permanecen inmóviles, pero podemos modificar su posición, tamaño y apariencia por medio de la propiedad **transform**. Esta propiedad realiza cuatro transformaciones básicas a un elemento: **escalado**, **rotación**, **inclinación** y **traslación**. Las siguientes son las **funciones** definidas para este propósito.

scale(x, y).-Esta función modifica la escala del elemento. Existen otras dos funciones relacionadas llamadas scaleX() y scaleY() para especificar los valores horizontales y verticales independientemente. Esta función recibe dos parámetros, el valor **x** para la escala horizontal y el valor **y** para la escala vertical. **Se puede utilizar un único valor** para ambos parámetros.

rotate(ángulo).-Esta función rota el elemento. El atributo representa los grados de rotación y se puede declarar en deg (grados), grad (gradianes), rad (radianes) o turn (espiras).





skew(ángulo).-Esta función inclina el elemento. El atributo representa los grados de desplazamiento. La función puede incluir dos valores para representar el ángulo horizontal y vertical. Los valores se pueden declarar en deg (grados), grad (gradianes), rad (radianes) o turn (espiras).

translate(x, y)—Esta función desplaza al elemento a la posición determinada por los atributos **x** e **y**.




Utiliza la división con identificador “**capítulo**” para probar esta propiedad (**transform**) con los códigos que se muestran a partir de aquí.

Función “scale”

CSS	Elemento <div id="capitulo"> sin transformar
<pre>#capitulo{ font: bold 48px Arial, Helvetica; color:#000080; text-align:center; width:750px; margin: 200px auto; padding: 25px; border: 5px solid #000080; border-radius: 20px; background-color:#ccccff; transform:scale(1, 3); /*Es equivalente a scaley(3)*/ }</pre>	
	Transformación sobre el eje y 
Los valores entre 0 y 1 reducen el tamaño. Y los valores negativos invierten la imagen. Observa:	
transform:scale(0.5, 0.5);	
transform:scale(1, -1);	

Función “rotate”

Esta función permite girar el elemento. Los números positivos giran el elemento como las agujas del reloj. Y los números negativos en sentido inverso.

CSS	Navegador
transform:rotate(-10deg);	

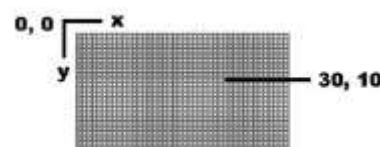
Función “skew”

Cambia la simetría en una o dos direcciones..


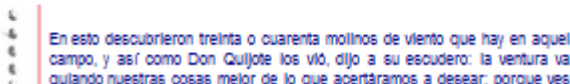
CSS	Navegador
<code>transform:skew(-20deg);</code>	
<code>transform:skew(20deg, 10deg);</code>	

Función “translate”


La pantalla de un dispositivo se divide en filas y columnas de píxeles (la mínima unidad visual de la pantalla). Con el propósito de identificar la posición de cada píxel, los ordenadores usan un sistema de coordenadas, donde las filas y columnas de píxeles se cuentan desde la esquina superior izquierda a la esquina inferior derecha de la cuadrícula, comenzando por el valor 0 (los valores se incrementan de izquierda a derecha y de arriba abajo). Por ejemplo:



Usando la propiedad **transform** podemos cambiar la posición de un elemento en la cuadrícula. La **función** que tenemos que asignar a la propiedad es **translate()**.

CSS	Navegador
<code>transform: translate(100px);</code> Este código desplaza el elemento 100 píxeles hacia la derecha con respecto a su posición inicial.	 

En ocasiones puede ser útil aplicar varias transformaciones, para ello hay que declarar las funciones separadas por un espacio.

CSS	Navegador
<code>transform: translatey(230px) rotate(20deg) scalex(0.6);</code> Este código desplaza el elemento 230 píxeles hacia abajo con respecto a su posición inicial.	

12. Transiciones

Con la **seudoclase :hover** podemos realizar transformaciones dinámicas. Para este propósito, CSS ofrece las siguientes propiedades.


transition-property.- Esta propiedad especifica las propiedades que participan en la transición. Además de los nombres de las propiedades, podemos asignar el valor **all** para indicar que todas las propiedades participarán de la transición.

transition-duration.- Esta propiedad especifica la duración de la transición en segundos.

transition-timing-function.- Esta propiedad determina la función que se usa para calcular los valores para la transición. Los valores disponibles son ease, ease-in, ease-out, ease-in-out, linear, step-start, y step-end.

transition-delay.- Esta propiedad especifica el tiempo que el navegador esperará antes de iniciar la animación.

transition.- Esta propiedad nos permite declarar todos los valores de la transición al mismo tiempo. Implementando estas propiedades le indicamos al navegador que tiene que crear todos los pasos de la animación y generar una transición entre el estado actual del elemento y el especificado por las propiedades. Las siguientes reglas implementan la propiedad transition para animar la transformación introducida en el ejemplo anterior.

CSS	Navegador
<pre>#capitulo{ font: bold 48px Arial, Helvetica; color:#000080; text-align:center; width:750px; margin: 30px auto; padding: 25px; border: 5px solid #000080; border-radius: 20px; background-color:#ccccff; transition: transform 1s ease-in-out 0s; } #capitulo:hover { transform: rotate(5deg); }</pre>	

La propiedad **transition** puede recibir hasta cuatro parámetros **separados por un espacio**. El primer valor es la propiedad que se considerará para crear la transición. En el ejemplo, usamos la propiedad **transform**, el segundo parámetro determina la **duración de la animación** (1 segundo), el tercer parámetro es un valor que determina **cómo se llevará a cabo la transición** y el último parámetro determina cuántos **segundos tarda la animación en comenzar**.



Prueba el código anterior sobre el HTML y CSS con el que hemos venido trabajando. Después, trata de aplicarlo sobre el elemento que contiene el inicio de capítulo. **¿Sabrías reutilizar la línea #capitulo:hover para que también sirviera para el elemento que contiene el inicio del capítulo?**

13. Animaciones

La propiedad `transition` crea una animación básica, pero solo se involucran dos estados en el proceso: el estado inicial determinado por los valores actuales de las propiedades y el estado final, determinado por los nuevos valores. Para crear una animación real, necesitamos declarar más de dos estados, como los fotogramas de una película. CSS incluye las siguientes propiedades para componer animaciones más complejas.

animation-name.- Esta propiedad especifica el nombre usado para identificar los pasos de la animación. Se puede usar para configurar varias animaciones al mismo tiempo declarando los nombres separados por coma.

animation-duration.- Esta propiedad determina la duración de cada ciclo de la animación. El valor se debe especificar en segundos (por ejemplo, 1s).

animation-timing-function.- Esta propiedad determina cómo se llevará a cabo el proceso de animación por medio de una curva Bézier declarada con los valores `ease`, `linear`, `ease-in`, `ease-out` y `ease-in-out`.

animation-delay.- Especifica el tiempo que hay que esperar para iniciar la animación.

animation-iteration-count.- Esta propiedad declara la cantidad de veces que se ejecutará la animación. Acepta un número entero o el valor `infinite`, el cual hace que la animación se ejecute por tiempo indefinido. El valor por defecto es 1.

animation-direction.- Esta propiedad declara la dirección de la animación. Acepta cuatro valores: `normal` (por defecto), `reverse`, `alternate`, y `alternate-reverse`. El valor `reverse` invierte la dirección de la animación, mostrando los pasos en la dirección opuesta en la que se han declarado. El valor `alternate` mezcla los ciclos de la animación, reproduciendo los que tienen un índice impar en dirección normal y el resto en dirección invertida. El valor `alternate-reverse` hace lo mismo que `alternate`, pero en sentido inverso.

animation-fill-mode.- Esta propiedad define cómo afecta la animación a los estilos del elemento. Acepta los valores `none` (por defecto), `forwards`, `backwards`, y `both`. El valor `forwards` mantiene al elemento con los estilos definidos por las propiedades aplicadas en el último paso de la animación, mientras que `backwards` aplica los estilos del primer paso tan pronto como se define la animación (antes de ser ejecutada). Finalmente, el valor `both` produce ambos efectos.

Animation.- Esta propiedad permite definir todos los valores de la al mismo tiempo.

Estas propiedades configuran la animación, pero los pasos se declaran por medio de la **regla @keyframes**. Esta regla se debe identificar con el nombre usado para configurar la animación, y debe incluir la lista de propiedades que queremos modificar en cada paso.

La posición de cada paso de la animación se determina con un valor en porcentaje, donde 0 % corresponde al primer fotograma o al comienzo de la animación, y 100 % corresponde al final.



En nuestro código CSS, añadiremos a la regla con selector `#capitulo` lo que se muestra a continuación. Más la regla `@keyframes`. Prueba los dos CSS y haz todas las pruebas que desees.

CSS(1)	CSS(2)
<pre>#capitulo{ ... animation: mianimacion 2s ease-in-out 0s infinite normal none; }</pre> <pre>@keyframes mianimacion { 0% { background: #FFFFFF; } 50% { background: #FF0000; } }</pre>	<pre>#capitulo{ ... animation: mianimacion 2s ease-in-out 0s infinite normal none; }</pre> <pre>@keyframes mianimacion { 0% { background: #FFFFFF; } 25% { transform: scale(0.5); background: #FFFF00; } 50% { transform: scale(1.5); background: #FF0000; } 75% { transform: scale(0.5); background: #FFFF00; } 100% { background: #FFFFFF; } }</pre>

14. Posicionamiento

14.1. Repaso a los elementos estructurales de HTML5

Un poco de historia. En HTML 4 se utilizaban dos elementos genéricos para agrupar la información que se quiere visualizar en el navegador: **<div>** y ****.

div.- Agrupa secciones de contenido en bloque. Aquellas que ocupan todo el ancho del navegador.

span.- Agrupa secciones de contenido en línea. En línea con los elementos del navegador.

En CSS existe una propiedad llamada **display** que determina el tipo de caja utilizada por el elemento.

Si es **block** el elemento será una caja en bloque que se ubicará debajo del elemento anterior y ocupará todo el ancho del navegador.

Si el valor de display es **inline**, se acomodará en línea con los elementos de su alrededor.

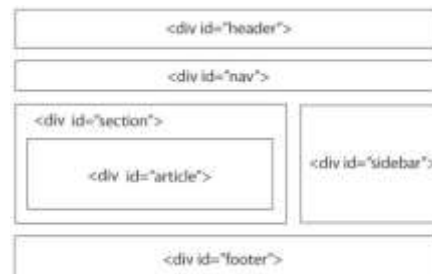
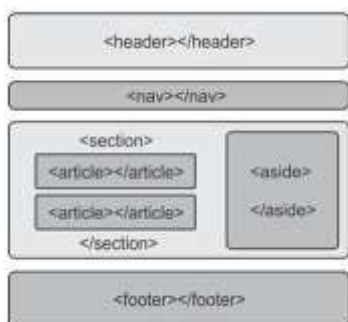
NAVEGADOR (interpretación de los códigos de la derecha)	HTML
<div>div1</div> <div>div2</div> <div>div3</div> <div>span1</div> <div>span2</div> <div>span3</div>	<pre><div class="borde">div1</div> <div class="borde">div2</div> <div class="borde">div3</div> span1 span2 span3</pre>
	CSS
	<pre>.borde{ border:1px solid; }</pre>



Prueba los códigos anteriores para comprobar su funcionalidad.

Algunos elementos que tienen **display block** son: `<header>`, `<article>`, `<section>`, `<footer>`, etc. Los que poseen un comportamiento tipo **inline** son `<a>`, ``, `<td>`, etc.

Tanto, **div** como **span** carecen de semántica. Sólo al aplicarles un estilo, mediante el atributo `id`, podremos indicar su comportamiento. Esto es lo que se hacía en **HTML 4**. Observa la figura de la derecha.



← En **HTML5** existen unos elementos estructurales que evitan utilizar el atributo `id` para cada uno de estos elementos. Estas secciones permiten un código más ordenado y limpio.

Efectivamente, estas etiquetas (o elementos) tienen un comportamiento tipo **block** (generan saltos de línea). Es decir, el contenido aparece en el navegador en bloques, unos debajo de otros. Y de eso tratarán las páginas siguientes, de mejorar la **presentación y distribución de estos elementos**.

14.2. Modelo de caja tradicional

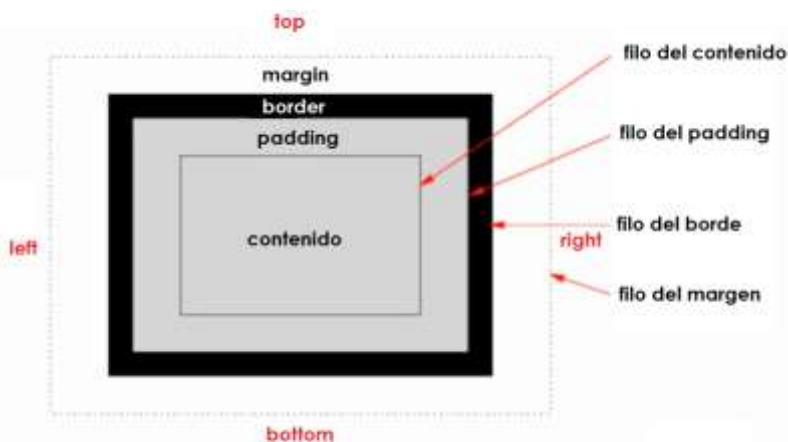
Cada elemento de una página es una caja rectangular. Y puede tener un **ancho**, **alto**, **relleno**, **borde** y un **margen**. En CSS estos elementos se definen con las propiedades **width**, **height**, **padding**, **border** y **margin**.

El **centro** de la caja es lo que se denomina **área del contenido** y está definido por las propiedades **width** y **height**.

El **rectángulo** que rodea **cada área** (contenido, relleno, borde o margen) recibe el nombre de **filo**.

Además la parte superior, derecha, inferior e izquierda reciben el nombre de **top**, **right**, **bottom** y **left** respectivamente.

El **margin** no afecta al tamaño de la caja, pero sí a la interacción con otras cajas. Por tanto, es parte importante en el modelo de caja. Tiene un color transparente y sólo a **border** se le puede asignar un color.

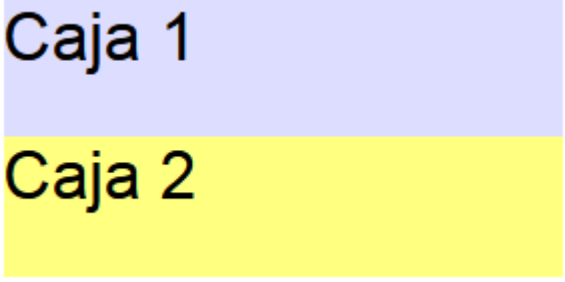
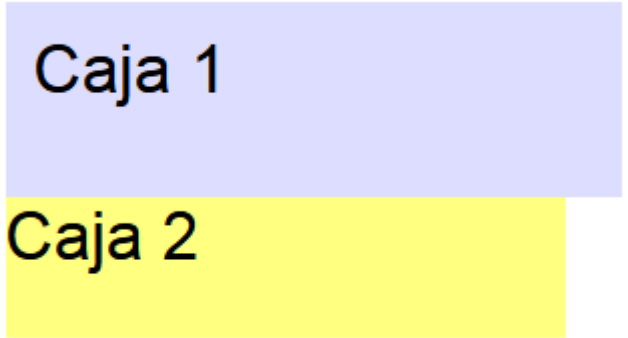
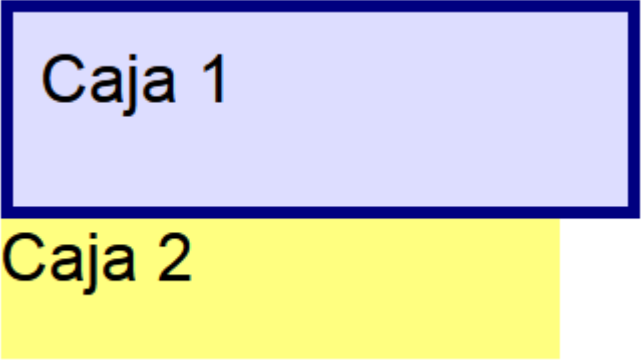


Para probar estos conceptos, partiremos del siguiente código HTML y CSS asociado.

```
<body>
  <div class="caja1">Caja 1</div>
  <div class="caja2">Caja 2</div>
</body>
```

CSS

Navegador

<pre>.caja1{ font: 48px Arial; background: #ddddff; width: 400px; height:100px; } .caja2{ font: 48px Arial; background: #ffff80; width: 400px; height: 100px; }</pre>	
<p>Si en el selector .caja1 aplicamos un padding de 20 píxeles, observa qué es lo que ocurre. El tamaño de la caja aumenta.</p> <pre>.caja1{ font: 48px Arial; background: #ddddff; width: 400px; height:100px; padding:20px; }</pre>	
<p>Ahora, si de nuevo al selector .caja1 aplicamos un borde de 10 píxeles, sólido y de color azul oscuro, la caja vuelve a incrementar el tamaño.</p> <pre>.caja1{ font: 48px Arial; background: #ddddff; width: 400px; height: 100px; padding: 20px; border: 10px solid #000080; }</pre>	

Prueba la propiedad **margin-top:20px**; en el selector **.caja2** y observa. ¿Aumenta el tamaño de la caja?

ejemploposicionamiento2.html

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

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Posicionamiento2</title>
  <link rel="stylesheet" href="posicionamiento2.css" </head>

<body>

  <div class="caja1">Caja 1</div>
  <div class="caja2">Caja 2</div>

</body>

</html>
```


posicionamiento2.css

```
.caja1{
  font: 48px Arial;
  background: #ddddff;
  width: 400px;
  height:100px;
}

.caja2{
  font: 48px Arial;
  background: #ffff80;
  width: 400px;
  height: 100px;
}
```

14.3. Posicionamiento static y relative

Una de las mejores cosas de CSS es que nos da la capacidad de colocar dentro de una página web contenidos y elementos en casi cualquier ubicación imaginable. Brindando una mejor estructura a nuestro diseño ayudando a ordenar mejor nuestro contenido.

La propiedad **position** controla el tipo de posicionamiento utilizado por un elemento dentro de sus elementos padres. Puede aceptar varios valores y cada uno tiene su propia aplicación.

static

La propiedad, **position: static;** es el valor predeterminado de los elementos. Es decir, los elementos se colocarán según el flujo normal del documento HTML como cualquier otro contenido tipo **block**. Por decirlo de otra manera, **static no provoca ningún posicionamiento especial** de los elementos y por tanto, los atributos **top**, **left**, **right** y **bottom** no se tendrán en cuenta.



Prueba el siguiente código. Observarás que aunque lo asocies a un estilo (con position:static;) los valores aplicados a las propiedades top, left, right y bottom no producen ningún efecto.

HTML	CSS:
<pre><body> <div> Este elemento no posee ningún posicionamiento especial si se declara como position: static; ya que es el posicionamiento por defecto. </div> </body></pre>	<pre>div{ position:static; top:50px; right:50px; bottom:50px; left:50px; background:#ddddff; }</pre>



Vuelve a copiar el elemento **div** del HTML dos o tres veces más en el propio documento, en el CSS añade la propiedad **width:200;** y comprueba que el elemento div aparece varias veces **pero siguiendo el flujo normal** de la información mostrada por el navegador.

relative



Para entender la propiedad **position: relative**; partiremos del siguiente código HTML asociado al CSS que se muestra a continuación. **Prueba estos códigos y los siguientes.**

HTML

```
<header>
  Petición de Sancho al doctor de la ínsula que gobernaba.
</header>
<main>
  -Mirad, señor doctor: de aquí adelante no os curéis de...
</main>
<footer>
  Suceso acontecido en el capítulo 49 de la segunda parte
</footer>
```

CSS	Navegador
<pre>header{ background:#ffaana; } main{ background:#aaffaa; } footer{ background:#aaaaff; }</pre>	

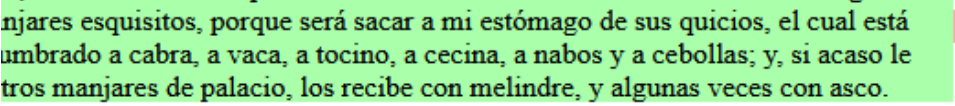
Ahora, al selector **main** le asignaremos la propiedad **position: relative**; y usaremos las propiedades **top**, **right**, **bottom** y **left** con valor cero. Así, el elemento principal (con fondo verde) seguirá en la misma posición. Es decir, seguirá en la posición asignada por el flujo normal de los elementos en el navegador.

CSS	Navegador
<pre>main{ position: relative; top: 0px; right: 0px; bottom: 0px; left: 0px; background: #aaffaa; }</pre>	

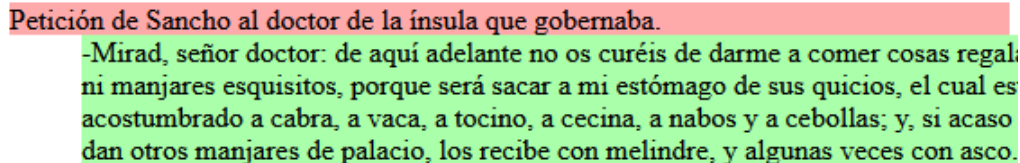
Por ejemplo, si a la propiedad **left** y **top** le aplicamos un desplazamiento de **40 píxeles**, observaremos que el elemento **main**, se moverá hacia la derecha 40 píxeles y hacia abajo otros 40 píxeles. Por tanto, debemos interpretar que **left** y **top** indican el lugar **desde donde se empuja al elemento** que se quiere desplazar, según la posición relativa que ocupa el elemento en la ventana del navegador. Incluso, en este caso, ocupará el espacio que se le había asignado al elemento footer (elemento de color morado).

CSS	Navegador
<pre>main{ position: relative; top: 40px; left: 40px; background: #aaffaa; }</pre>	

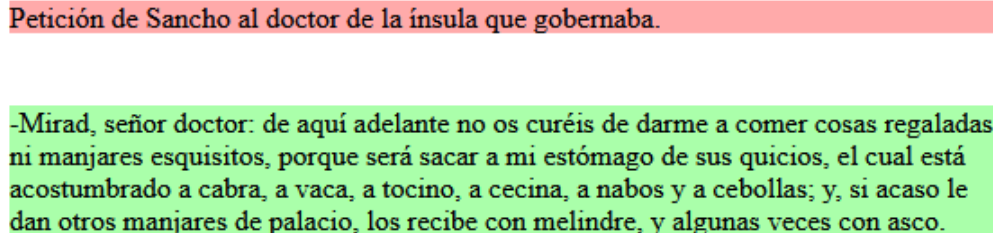
Por supuesto, si utilizamos las propiedades **right** y **bottom** podemos conseguir que el elemento se desplace hacia la izquierda y hacia arriba (tapando el elemento header). Observa y prueba.

CSS	Navegador
<pre>main{ position:relative; right:40px; bottom:40px; background:#aaffaa; }</pre>	 <p>Suceso acontecido en el capítulo 49 de la segunda parte.</p>

Llegados a este punto sólo nos queda probar una cosa que puede ser extraña y contradictoria, a la vez. ¿Qué pasaría si a la propiedad **left** y **right** le aplicamos un valor a la vez? La respuesta es que el valor asignado a left prevalece sobre el de right.

CSS	Navegador
<pre>main{ position:relative; right:40px; left:40px; background:#aaffaa; }</pre>	 <p>Suceso acontecido en el capítulo 49 de la segunda parte.</p>

Lo mismo sucederá si aplicamos un valor a **top** y **bottom**. El valor asignado a **top** también prevalece frente al asignado a **bottom**.

CSS	Navegador
<pre>main{ position:relative; bottom:40px; top:40px; background:#aaffaa; }</pre>	

14.4. Posicionamiento absolute (absoluto)

El comportamiento de los elementos con el posicionamiento **absolute** es muy similar al **relative**. También, utiliza las propiedades **top**, **right**, **bottom** y **left** para desplazar o posicionar el elemento.

Sin embargo, antes de analizar las diferencias leamos los siguientes dos puntos:

1. Cuando el elemento se ubica con posicionamiento **relative** el elemento se desplaza **pero el diseño no se modifica**.
2. Con posicionamiento **absolute**, el elemento se **elimina del diseño**, por lo que **el resto de los elementos se desplazarán para ocupar el nuevo espacio libre**.

Recuerda el código HTML, el CSS con posicionamiento **relative** y lo que mostraba el navegador.

HTML

<header>

Petición de Sancho al doctor de la ínsula que gobernaba.

</header>

<main>

-Mirad, señor doctor: de aquí adelante no os curéis de...

</main>

<footer>

Suceso acontecido en el capítulo 49 de la segunda parte

</footer>

CSS

```
header{
background:#ffaana;
}

main{
position:relative;
top:10px;
left:100px;
background:#aaffaa;
}

footer{
background:#aaaaff;
}
```

Navegador

Petición de Sancho al doctor de la ínsula que gobernaba.

-Mirad, señor doctor: de aquí adelante no os curéis de darme a comer cosas regaladas ni manjares esquisitos, porque será sacar a mi estómago de sus quicios, el cual está acostumbrado a cabra, a vaca, a tocino, a cecina, a nabos y a cebollas; y, si acaso le dan otros manjares de palacio, los recibe con melindre, y algunas veces con asco.

Suceso acontecido en el capítulo 49 de la segunda parte.

Ahora vamos a ver qué ocurre con posicionamiento **absolute**.

CSS

```
header{
background:#ffaana;
}

main{
position:absolute;
top:10px;
left:100px;
background:#aaffaa;
}

footer{
background:#aaaaff;
}
```

Navegador

Petición de Sa-Mirad, señor doctor: de aquí adelante no os curéis de darme a comer cosas regaladas ni manjares esquisitos, porque será sacar a mi estómago de sus quicios, el cual está acostumbrado a cabra, a vaca, a tocino, a cecina, a nabos y a cebollas; y, si acaso le dan otros manjares de palacio, los recibe con melindre, y algunas veces con asco.



En la imagen anterior vemos que el elemento **main**, abandona el flujo de la información mostrada por el navegador. Por esta razón, el elemento **footer** ocupa el espacio no ocupado por **main**. ¡Prueba los códigos anteriores y los de la página siguiente!

Entonces, los valores de las propiedades **top**, **right**, **bottom** y **left**, ¿con respecto a qué elemento se tienen en cuenta?

Pues, con respecto al elemento padre que lo contiene. En nuestro caso el elemento body (ventana del navegador). Y eso sí, **el elemento padre no debe tener posicionamiento static**.

Es decir, si el elemento padre posee un posicionamiento **relative**, ahora el posicionamiento **absolute**, se calcula teniendo como referencia los bordes del elemento contenedor “padre” con posicionamiento **relative**.

Observa el siguiente código y el nuevo comportamiento del elemento **main**. Eso sí, observa que hora está **dentro de otra división que llamaremos “contenedor”** (con posicionamiento **static**).

HTML	
<pre> <div id="contenedor"> <main> -Mirad, señor doctor: de aquí adelante no os curéis de darme cosas regaladas ni manjares exquisitos, porque será sacar a mi estómago de sus quicios, el cual está acostumbrado a cabra, a vaca, a tocino, a cecina, a nabos y a cebolla; y, si acaso le dan otros manjares de palacio, los recibe con melindre y algunas veces con asco. </main> </div> </pre>	
CSS	Navegador
<pre> #contenedor{ width:400px; height:200px; background: #ffffaa; position:relative; } main{ width:300px; background:#aaffaa; position:absolute; right:0px; bottom:0px; } </pre>	

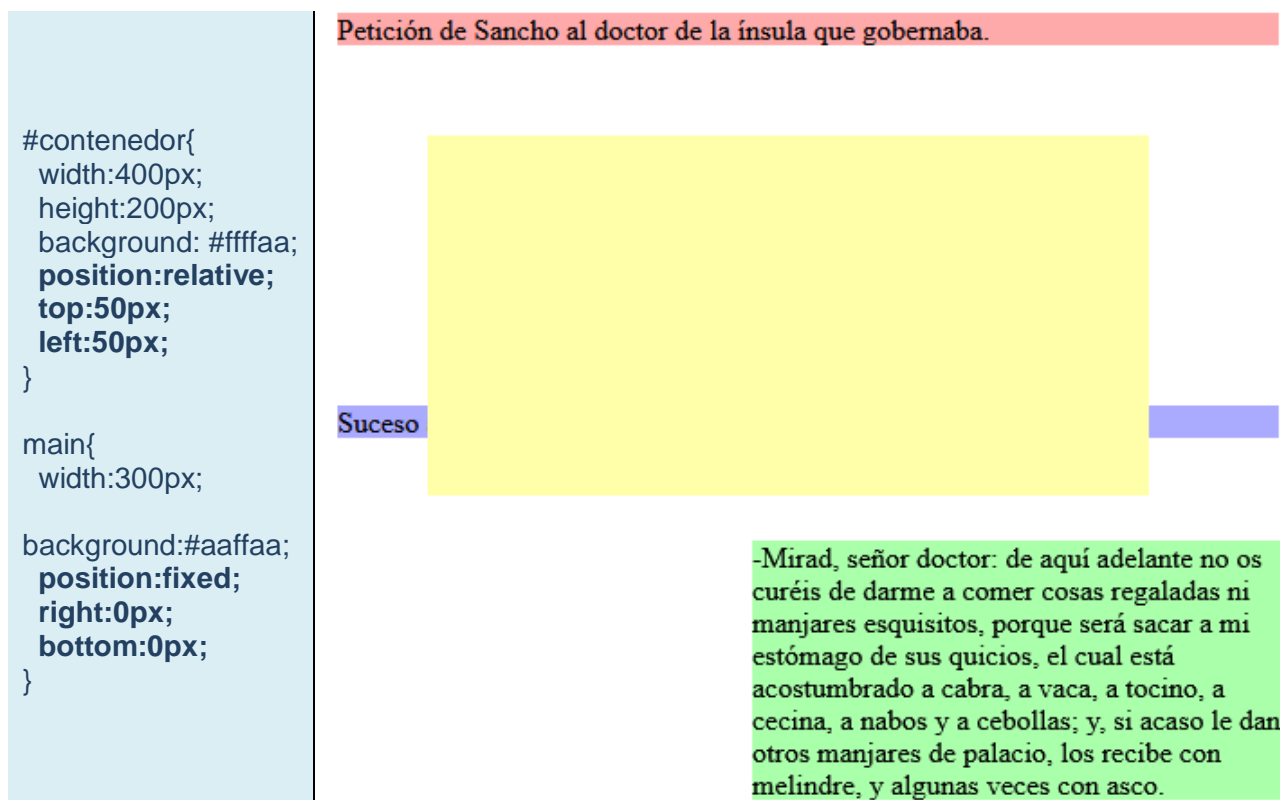
Observa en el siguiente CSS los valores asignados al posicionamiento **relative** de contenedor. Y cómo **footer** no ocupa ningún espacio (**no aparece pegado a header**). Esto es debido a que el elemento que se “mueve” de su lugar posee posicionamiento **relative**. Y éste, a diferencia de **absolute**, no desaparece del diseño seguido por el navegador. Deberá aparecerte algo así:

CSS	Navegador
<pre> #contenedor{ width:400px; height:200px; background: #ffffaa; position:relative; top:50px; left:50px; } main{ width:300px; background:#aaffaa; position:absolute; right:0px; bottom:0px; } </pre>	

}

14.5. Posicionamiento fixed (fijado)

Para el estudio de este posicionamiento partiremos del último código.		<div>HTML</div> <div><body> <header>...</header> <div id="contenedor"> <main> -Mirad, señor doctor: de aquí adelante no os curéis... </main> </div> <footer>...</footer> </body></div>
<div>CSS</div> <div>#contenedor{ width:400px; height:200px; background: #ffffaa; position:relative; } main{ width:300px; background:#aaffaa; position:absolute; right:0px; bottom:0px; }</div>	<div>Navegador</div> <div><div>Petición de Sancho al doctor de la ínsula que gobernaba.</div><div>-Mirad, señor doctor: de aquí adelante no os curéis de darme a comer cosas regaladas ni manjares esquisitos, porque será sacar a mi estómago de sus quicios, el cual está acostumbrado a cabra, a vaca, a tocino, a cecina, a nabos y a cebollas; y, si acaso le dan otros manjares de palacio, los recibe con melindre, y algunas veces con asco.</div><div>Suceso acontecido en el capítulo 49 de la segunda parte.</div></div>	
Con el posicionamiento fixed , el elemento se coloca tomando como referencia la ventana del navegador . Independientemente del lugar o contenedor en el que esté declarado dicho elemento.		
<div>CSS</div>	<div>Navegador</div>	



14.6. Posicionamiento sticky

Partimos del código que hemos estado utilizando con alguna modificación. Observa el código html:

```
<body>
  <header>
    Petición de Sancho al doctor de la insula que gobernaba.
  </header>
  <div class="contenedor">
    <div class="barra">
      Barra
    </div>
    <div class="contenedor2">
      -Mirad, señor doctor: de aquí adelante no os curéis de darme a comer
      cosas regaladas ni manjares esquisitos, porque será sacar a mi estómago
      de sus quicios, el cual está acostumbrado a cabra, a vaca, a tocino,
      a cecina, a nabos y a cebollas;
    </div>
  </div>
```

Con el posicionamiento **sticky** el elemento se comporta como si tuviese un posicionamiento **relative** hasta que alcanza un umbral específico (por ejemplo, el borde del contenedor padre) y desde ese momento actúa como si tuviese posicionamiento **fixed** mientras puedes desplazarte dentro del contenedor. Aplica al CSS de nuestro ejemplo el

siguiente código para el elemento nuevo **barra** y cambia el elemento div por la clase **contenedor2**: (así debe quedarte el fichero .css)

```
header{
    background: rgb(236, 141, 141);
    margin-bottom: 10px;
}
.barra{
    border:2px solid black;
    box-shadow: 2px 5px rgb(42, 42, 116) inset;
    text-align: center;
    background-color: rgb(119, 227, 233);
    margin-bottom: 10px;
    position:sticky;
    top:10px;
}
.contenedor{
    height: 1000px;
    background-color: rgb(243, 243, 68);
    position:relative;
}

.contenedor2{
    width: 300px;
    background-color: rgb(123, 240, 108);
    margin-left: 10px;
}
footer{
    background: rgb(243, 87, 193);
}
```

En resumen:

1. **static.-** El elemento sigue el flujo normal del documento; top, right, bottom y left **no tienen efecto**.
2. **relative.-** La posición se calcula según la posición relativa que el elemento debe ocupar.
3. **absolute.-** El nuevo posicionamiento, **dentro del contenedor padre**, no deja hueco en el diseño.
4. **fixed.-** El posicionamiento tiene como referencia la ventana del navegador.
5. **sticky.-** permite que el elemento se mantenga visible cuando el usuario se desplaza por debajo de su posición inicial.


14.7. Contenido flotante

Las propiedades **float** y **clear** se usaron originalmente para hacer que el contenido flote alrededor de un elemento. Por ejemplo, si queremos que un texto se muestre al lado de una imagen, **podemos hacer flotar la imagen** hacia la izquierda o la derecha, y el texto compartirá con la imagen el espacio disponible en la misma línea.

Partiremos del código de la derecha:

HTML

```
<body>
  <header>
    Capítulo 18: Donde se cuentan las razones que pasó
    Sancho Panza con su señor Don Quijote con otras
    aventuras dignas de ser contadas.
  </header>
  <main>
    
    ... se entró por medio del escuadrón de las ovejas, y
    comenzó de alanceallas con tanto coraje y denuedo, como
    si de veras alanceara a sus mortales enemigos. Los pastores y
    ganaderos que con la manada venían, dábanle voces que no
    hiciese aquello; pero viendo que no aprovechaban, descifñéronse
    las ondas, y comenzaron a saludarle los oídos con piedras como
    el puño...
  </main>
  <footer>
    Parte del párrafo en donde se narra cómo D. Quijote se
    enfrenta a un rebaño de ovejas.
  </footer>
</body>
```

CSS	Navegador
<pre>main{ width:800px; margin:20px auto; padding: 10px; background:#eeeeff; color: #000080; font: normal 26px Arial; text-align: justify; } header, footer{ width:900px; margin: 20px auto; padding: 10px; text-align: center; background: #ccccff; color: #000080; font: bold 26px Arial, Verdana; border: 5px solid #000080; border-radius: 20px; }</pre>	

Si queremos que el texto se muestre alrededor de la imagen, podemos hacer que ésta **“flote”** a la izquierda o la derecha. Observa cómo puedes utilizar la **propiedad float** y el efecto que produce:

CSS	Navegador
-----	-----------

<pre>main img{ margin:5px; float:left; }</pre>	 <p>... se entró por medio del escuadrón de las ovejas, y comenzó de alanceallas con tanto coraje y denuedo, como si de veras alanceara a sus mortales enemigos. Los pastores y ganaderos que con la manada venían, dábanle voces que no hiciese aquello; pero viendo que no aprovechaban, desciñéronse las ondas, y comenzaron a saludarle los oídos con piedras como el puño...</p>
CSS	Navegador
<pre>main img{ margin:5px; float:right; }</pre>	<p>... se entró por medio del escuadrón de las ovejas, y comenzó de alanceallas con tanto coraje y denuedo, como si de veras alanceara a sus mortales enemigos. Los pastores y ganaderos que con la manada venían, dábanle voces que no hiciese aquello; pero viendo que no aprovechaban, desciñéronse las ondas, y comenzaron a saludarle los oídos con piedras como el puño...</p> 



Prueba los códigos anteriores.

Los navegadores no pueden calcular el tamaño de un contenedor a partir del tamaño de elementos flotantes, por lo que si el elemento afectado por la propiedad float es más alto que el resto de los elementos, desbordará al contenedor.

Por ejemplo, la imagen ilustra lo que ocurre si utilizamos una imagen o elemento flotante que es más alto que el espacio ocupado por el texto.



Prueba esto último acortando el texto para apreciar lo comentado. Y, luego, los siguientes códigos.

Capítulo 18: Donde se cuentan las razones que pasó Sancho Panza con su señor Don Quijote con otras aventuras dignas de ser contadas.



... se entró por medio del escuadrón de las ovejas, y comenzó de alanceallas con tanto coraje y denuedo...

Parte del párrafo en donde se narra cómo D. Quijote se enfrenta a un rebaño de ovejas.

Una forma de asegurarnos que el contenedor es lo suficientemente alto, como para contener los elementos flotantes, **es asignándole (al contenedor) la propiedad overflow el valor auto**. Esto fuerza al navegador a calcular el tamaño del contenedor considerando todos los elementos en su interior.

```
main{
  width:800px;
  margin:20px auto;
  padding: 10px;
  background:#eeeeff;
  color: #000080;
  font: normal 26px Arial, Verdana;
  text-align: justify;
  overflow: auto;
}

main img{
  margin: 5px;
  float: left;
}
```

Capítulo 18: Donde se cuentan las razones que pasó Sancho Panza con su señor Don Quijote con otras aventuras dignas de ser contadas.



... se entró por medio del escuadrón de las ovejas, y comenzó de alanceallas con tanto coraje y denuedo...

Parte del párrafo en donde se narra cómo D. Quijote se enfrenta a un rebaño de ovejas.

Otra alternativa para **normalizar el flujo del documento** es utilizando la propiedad **clear** allí donde haga falta. Esta técnica **requiere que agreguemos un elemento adicional al documento HTML asociado a su correspondiente regla CSS**.

Por ejemplo, podemos incluir un elemento <div> debajo del texto que describe parte del capítulo para impedir que los siguientes elementos continúen flotando hacia los lados. Empecemos a probar lo dicho.

HTML

```
<main>
  
  ... se entró por medio del escuadrón de las ovejas, y...
  <div class="salto_bajo_ele_flotantes"></div>
  Cervantes nació en Alcalá de Henares (1547). La primera parte del
  Quijote se publicó en 1605 y la segunda en 1615. Muere en 1616.
</main>
```

CSS

```
.salto_bajo_ele_flotantes {
  clear: both;
}
```

Navegador

Capítulo 18: Donde se cuentan las razones que pasó Sancho Panza con su señor Don Quijote con otras aventuras dignas de ser contadas.



... se entró por medio del escuadrón de las ovejas, y comenzó de alanceallas con tanto coraje y denuedo...

Cervantes nació en Alcalá de Henares (1547). La primera parte del Quijote se publicó en 1605 y la segunda en 1615. Muere en 1616.

Parte del párrafo en donde se narra cómo D. Quijote se enfrenta a un rebaño de ovejas.

Los valores de **clear** son: **left**, **right** y **both**.

14.8. Cajas flotantes

En este nuevo apartado, veremos cómo podemos utilizar cajas flotantes para crear columnas de información.

El siguiente documento incluye el elemento <main> con cuatro elementos <div> que debemos hacer flotar a un lado (izquierda o derecha) para convertirlos en columnas dentro de la página.

Y un elemento <div> al final que usaremos para recuperar el flujo normal del documento.

Al abrir el HTML, sin aplicar estilos personalizados, el contenido de los elementos se muestra de arriba hacia abajo, siguiendo el flujo del documento.

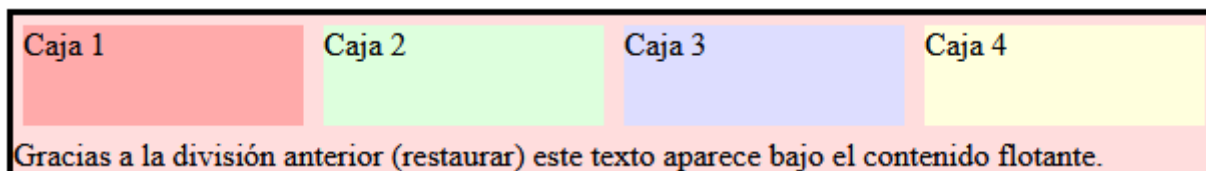
EjemploCajas.html

HTML	Navegador
<pre> <body> <main id="cajapadre"> <div id="caja-1">Caja 1</div> <div id="caja-2">Caja 2</div> <div id="caja-3">Caja 3</div> <div id="caja-4">Caja 4</div> <div class="restaurar"></div> Gracias a la división anterior (restaurar) este texto aparece bajo el contenido flotante. </main> </body> </pre>	<p>Caja 1 Caja 2 Caja 3 Caja 4</p> <p>Gracias a la división anterior (restaurar) este texto aparece bajo el contenido flotante.</p>

Debido a que queremos que estas cajas se ubiquen una al lado de la otra representando columnas de nuestra página web, tenemos que asignarles un tamaño fijo y hacerlas flotar a un lado o al otro. El tamaño se determina mediante las propiedades **width** y **height**, y el modo en el que flotan lo determina la propiedad **float**, pero el valor asignado a esta última propiedad depende de lo que queremos lograr.

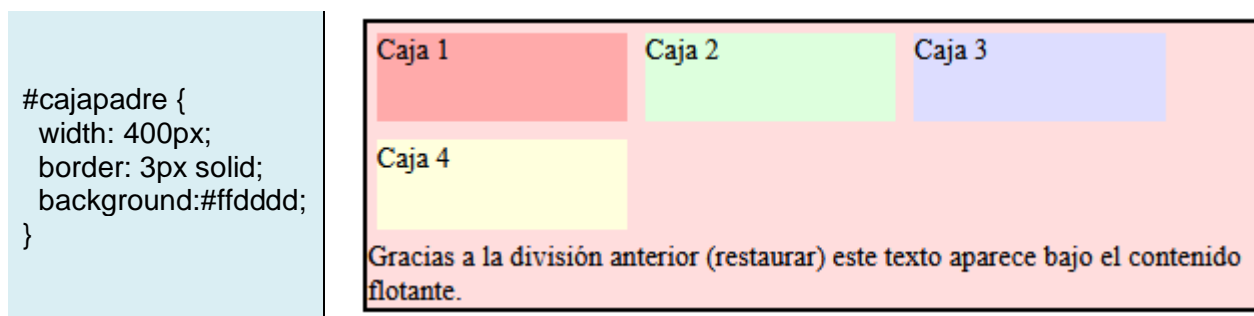
Si flotamos las cajas hacia la izquierda, se alinearán de izquierda a derecha, y si las hacemos flotar hacia la derecha, harán lo mismo pero de derecha a izquierda. Además, irán apareciendo en el orden declarado.

CSS		
<pre> #cajapadre { width: 600px; margin: 50px, auto; border: 3px solid; background:#ffdddd; } #caja-1 { float: left; width: 140px; height: 50px; margin: 5px; background-color: #ffaana; } </pre>	<pre> #caja-2 { float: left; width: 140px; height: 50px; margin: 5px; background-color: #ddffdd; } #caja-3 { float: left; width: 140px; height: 50px; margin: 5px; background-color: #dddfff; } </pre>	<pre> #caja-4 { float: left; width: 140px; height: 50px; margin: 5px; background-color: #ffffdd; } .restaurar { clear: both; } </pre>
Navegador		



En el siguiente ejemplo, como el contenedor no tiene espacio suficiente, los elementos que no se pueden ubicar en la misma línea se moverán a una nueva. La siguiente regla reduce el tamaño del elemento **#cajapadre** a 400 píxeles. Observa qué ocurre.

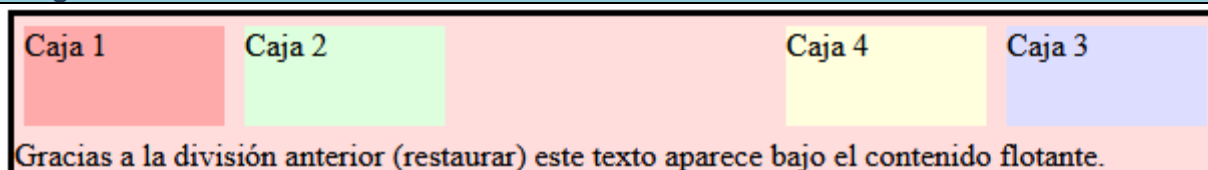
CSS	Navegador
-----	-----------



Por otro lado, si tenemos más espacio disponible en el contenedor del que necesitan las cajas, el espacio restante se ubica a los lados (izquierdo o derecho, dependiendo del valor de la propiedad float).

Si queremos ubicar el espacio restante en medio de las cajas, podemos hacer flotar algunas cajas hacia la izquierda y otras hacia la derecha. Por ejemplo, las siguientes reglas asignan un tamaño de 100 píxeles a cada caja, dejando un espacio vacío de 160 píxeles, pero como las dos últimas cajas flotan a la derecha en lugar de a la izquierda, el espacio libre se ubica en el medio del contenedor, no a los lados.

CSS		
<pre>#cajapadre { width: 600px; border: 3px solid; background:#ffdddd; } #caja-1 { float: left; width: 100px; height: 50px; margin: 5px; background-color: #ffaana; }</pre>	<pre>#caja-2 { float: left; width: 100px; height: 50px; margin: 5px; background-color: #ddffdd; } #caja-3 { float: right; width: 100px; height: 50px; margin: 5px; background-color: #ddffdd; }</pre>	<pre>#caja-4 { float: right; width: 100px; height: 50px; margin: 5px; background-color: #ffffdd; } .restaurar { clear: both; }</pre>
Navegador		



Debido al orden en el que se han declarado los elementos en el código, el elemento caja- 4 se ubica en el lado izquierdo del elemento caja-3. El navegador procesa los elementos en el orden en el que se han declarado en el documento; por lo tanto, cuando se procesa el elemento caja-3,

se mueve a la derecha hasta que alcanza el límite derecho del contenedor, pero cuando se procesa el elemento caja-4, no se puede mover hacia el lado derecho del contenedor porque el elemento caja-3 ya ocupa ese lugar y, por lo tanto, se ubica en el lado izquierdo del elemento caja-3.



Acaba el estudio de esta hoja probando los códigos anteriores.

14.9. Cajas y posicionamiento (un ejemplo práctico)

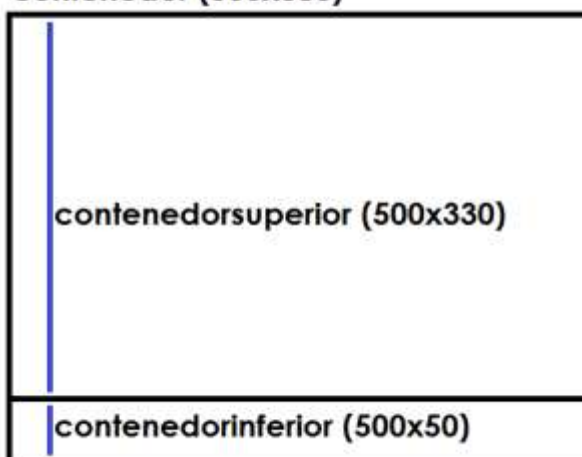
En algunas páginas en las que abundan las fotografías, como la de los periódicos, al pasar el puntero del ratón sobre ellas aparece un texto explicativo en la parte inferior.

Lo explicaré haciendo referencia a la siguiente imagen.

cajapadre (500x330)



contenedor (500x380)



En primer lugar utilizaremos un elemento que llamaremos **cajapadre** con las siguientes características, tendrá un posicionamiento **relative** y en él situaremos la **imagen** y otro elemento que llamaremos **contenedor**. En el CSS vendrá definido por la siguiente regla:

```
#cajapadre {  
  width:500px;  
  height:330px;  
  margin:20px auto;  
  position:relative;  
  overflow:hidden;  
}
```

En segundo lugar, vamos a definir el elemento **contenedor** con posicionamiento **absolute**. Tendrá el mismo ancho que **cajapadre** pero el alto será 50 píxeles mayor. Es decir, **contenedor** estará dividido en dos partes:

1. **contenedorsuperior** con idénticas dimensiones que **cajapadre**. Al no contener nada permitirá visualizar lo que hay debajo, la imagen.
2. **contenedorinferior** que contendrá el texto. Sólo será visible cuando el elemento padre (**contenedor**) suba 50 píxeles. Lo haremos cuando **el puntero del ratón se sitúe por encima del elemento contenedor**.

Las reglas que definen las características de **contenedor**, **contenedorsuperior** y **contenedorinferior** son las que se describen a continuación.

ejemplocontenedores.html
contenedores.css

<pre>#contenedor { position: absolute; top:0px; width: 500px; height: 380px; } .rotulo{ width:900px; margin: 20px auto; padding: 10px; text-align: center; background: #dddfff; font: bold 26px Arial,Verdana; color: #000080; border: 5px solid #000080; border-radius: 20px; }</pre>	<pre>#contenedorsuperior { width: 500px; height: 330px; }</pre>	<pre>#contenedorinferior { width: 500px; height: 40px; padding-top: 10px; background-color: rgba(200, 200, 200, 0.8); text-align: center; font: bold 26px Arial; }</pre>	<pre>#contenedor:hover { top: -50px; }</pre>
---	---	--	--

Con este HTML

```
<body>
  <div class="rotulo">
    Empezamos casi de broma con personajes y textos del libro "Don Quijote de la Mancha"...
  </div>

  <section id="cajapadre">
    
    <div id="contenedor">
      <div id="contenedorsuperior"></div>
      <div id="contenedorinferior">
        <span><strong>Sello sobre el inicio del libro</strong></span>
      </div>
    </div>
  </section>

  <div class="rotulo">
    Pero después fuimos utilizándolo. Los personajes, refranes y textos de algunos capítulos...
  </div>
</body>
```

Y este CSS		
<pre>.rotulo{ width:900px; margin: 20px auto; padding: 10px; text-align: center; background: #ddddff; font: bold 26px Arial,Verdana; color: #000080; border: 5px solid #000080; border-radius: 20px; }</pre>	<pre>#cajapadre { width:500px; height:330px; margin:20px auto; position:relative; overflow:hidden; } #contenedor { position: absolute; top:0px; width: 500px; height: 380px; }</pre>	<pre>#contenedorsuperior { width: 500px; height: 330px; } #contenedorinferior { width: 500px; height: 40px; padding-top: 10px; background-color: rgba(200, 200, 200, 0.8); text-align: center; font: bold 26px Arial, Verdana; } #contenedor:hover { top: -50px; }</pre>

Empezamos casi de broma con personajes y textos del libro "Don Quijote de la Mancha" para aprender HTML5.



Pero después fuimos utilizándolo. Los personajes, refranes y textos de algunos capítulos nos sirvieron para aprender los entresijos de los archivos CSS.

Al situar el puntero del ratón sobre la imagen aparece el texto en la parte inferior.



Seguro que estarás deseando probar estos códigos. ¡Adelante!

14.10. Ejemplo de posicionamiento float-absolute-relative

Como en la entrega anterior, trataremos de explicar sin adornos ni florituras cómo podemos utilizar elementos flotantes, junto al posicionamiento relativo y absoluto, para ordenar la información que el navegador mostrará en pantalla.

Empezaremos visualizando el objetivo de lo que interpretará el navegador. Ahora habrá **dos imágenes** dentro del contenedor. Una de ellas **flotará hacia la izquierda** y la otra **hacia la derecha**. Y de la misma forma que antes, al situar el cursor del ratón sobre ellas aparecerá un **texto explicativo en la parte inferior**.

Empezamos casi de broma con personajes y textos del libro "Don Quijote de la Mancha" para aprender HTML5.



Pero después fuimos utilizándolo. Los personajes, refranes y textos de algunos capítulos nos sirvieron para aprender los entresijos de los archivos CSS.

Diferencias con respecto al código mostrado en la entrega anterior:

Se declarará un elemento (**main**) que contendrá las imágenes. Con posicionamiento **relativo** y propiedad **overflow:auto**; para que calcule el espacio de los elementos flotantes (imágenes) que situaremos en su interior:

```
main{
  width:1100px;
  margin:20px auto;
  background:#eeeeff;
  position:relative;
  overflow:auto;
}
```

El elemento **cajapadre**, de la hoja anterior, desaparece. Pero se crean otros dos muy similares (**cajapadre_izq** y **cajapadre_der**) que tendrán comportamiento **flotante** (uno a izquierda y el otro a derecha).

```
.cajapadre_izq {
  width:500px;
  height:330px;
  margin:20px;
  position:relative;
  overflow:hidden;
  float:left;
}
```

```
.cajapadre_der {
  width:500px;
  height:330px;
  margin:20px;
  position:relative;
  overflow:hidden;
  float:right;
}
```




Aquí tienes el HTML y su CSS asociado para que los pruebes.

ejemploimagenesflotantes.html imagenesflotantes.css

HTML

```
<body>
  <div class="rotulo">
    Empezamos casi de broma con personajes y textos del libro "Don Quijote de la Mancha"...
  </div>

  <main>

    <section class="cajapadre_izq">
      
      <div class="contenedor">
        <div class="contenedorsuperior"> </div>
        <div class="contenedorinferior">
          <span><strong>Sello sobre el inicio del libro</strong></span>
        </div>
      </div>
    </section>

    <section class="cajapadre_der">
      
      <div class="contenedor">
        <div class="contenedorsuperior"> </div>
        <div class="contenedorinferior">
          <span><strong>Batalla con las ovejas</strong></span>
        </div>
      </div>
    </section>
  </main>

  <div class="rotulo">
    Pero después fuimos utilizándolo. Los personajes, refranes y textos de algunos capítulos...
  </div>
</body>
```

CSS

```
.rotulo{
width:900px;
margin: 20px auto;
padding: 10px;
text-align: center;
background: #ddddff;
font: bold 26px Arial, Verdana;
border: 5px solid #000080;
border-radius: 20px;
}

/*Novedad con respecto a la hoja anterior*/
main{
width:1100px;
margin:20px auto;
background:#eeeeff;
position:relative;
overflow:auto;
}

.cajapadre_izq {
width:500px;
height:330px;
margin:20px;

.cajapadre_der {
width:500px;
height:330px;
margin:20px;
position:relative;
overflow:hidden;
float:right;
}
/* ----- */
.contenedor {
position: absolute;
top:0px;
width: 500px;
height: 380px;
}
.contenedorsuperior {
width: 500px;
height: 330px;
}
.contenedorinferior {
width: 500px;
height: 40px;
padding-top: 10px;
background-color: rgba(200, 200, 200, 0.8);
}
```

```
position:relative;
overflow:hidden;
float:left;
}
```

```
text-align: center;
font: bold 26px Arial, Verdana, sans-serif;
}
.contenedor:hover {
top: -50px;
}
```



Actividad2_4

14.11. Propiedad z-index

Hasta ahora hemos trabajado con el **posicionamiento** static, relative, absolute y fixed. También, hemos visto el **comportamiento de caja flotante** (propiedad **float**). Gracias a esto podemos mejorar la estética y presentación de los distintos elementos.

Pero existe otra propiedad con la que, en caso de que las cajas se superpongan, **podemos indicar qué caja ha de poder presentarse en primer plano**. Dando así una sensación de profundidad en los elementos presentados.

Observemos este primer código HTML, su CSS y la interpretación que hace el navegador.

ejemplozindex.html
zindex.css

```
<body>
  <div class="uno">1</div>
  <div class="dos">2</div>
  <div class="tres">3</div>
</body>
```

CSS	Navegador
<pre>div{ border:2px solid; border-radius:10px; width:200px; height:200px; position:relative; font-size:72px; } /*Posicionamiento*/ .uno{ background:#ffa aaa; } .dos{ background:#aaffaa; top:-100px; left:100px; } .tres{ background:#aaaaff; top:-200px; left:200px;</pre>	

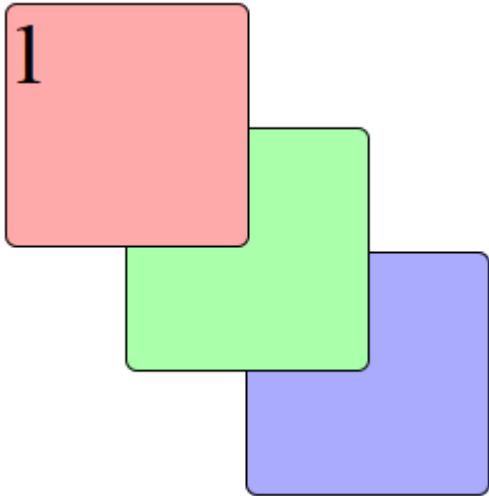
}

Vemos que los tres elementos (de clase uno, dos y tres) aparecen en el orden descritos en el documento HTML.

¿Pero cómo podemos hacer para que el elemento que aparezca en primer plano sea el uno, luego el dos y por último el tres? Vamos a verlo el la otra cara de la hoja.

Existe una propiedad **z-index** que dependiendo del valor numérico, el elemento ocupará un lugar u otro. Los números mayores harán que el elemento ocupe un plano superior frente a otro que contenga un número inferior.

Veamos esto con un ejemplo. Ahora la propiedad **z-index** se la aplicaremos a las tres clases de elementos. Utilizaremos rangos numéricos altos (múltiplos de 10) para hacer, después, las pruebas que consideremos. Observa el resultado aplicando esta propiedad:

CSS	Navegador
<pre>.uno{ background:#ffaana; z-index: 30; } .dos{ background:#aaffaa; top:-100px; left:100px; z-index: 20; } .tres{ background:#aaaaff; top:-200px; left:200px; z-index: 10; }</pre>	



Analiza la imagen inferior y trata de codificar el HTML y su CSS asociado para que muestre lo mismo. Las imágenes aportadas tienen un tamaño de 500x330 píxeles. Por lo que el **width** y **height** que utilizarás será de **500x500** píxeles. Suficiente para alojar el número y la propia imagen.

Otra pista para que sólo te preocupes del CSS:

```
<div class="uno">1<br/></div>
```

```
<div class="dos">2<br/></div>
```

```
<div class="tres">3<br/></div>
```



14.12. Columnas.

Además de las propiedades que hemos estudiado para organizar las cajas en pantalla, CSS también incluye un grupo de propiedades para facilitar la **creación de columnas**.

column-count.- Esta propiedad especifica el número de columnas que el navegador tiene que generar para distribuir el contenido del elemento.

column-width.- Esta propiedad declara el ancho de las columnas. El valor se puede declarar como auto (por defecto) o en cualquiera de las unidades de CSS, como píxeles o porcentaje.

column-span.- Esta propiedad se aplica a elementos dentro de la caja para determinar si el elemento se ubicará en una columna o repartido entre varias columnas. Los valores disponibles son all (todas las columnas) y none (por defecto).

column-fill.- Esta propiedad determina cómo se repartirá el contenido entre las columnas. Los valores disponibles son auto (las columnas son completadas de forma secuencial) y balance (el contenido se divide en partes iguales entre todas las columnas).

column-gap.- Esta propiedad define el tamaño del espacio entre las columnas. Acepta un valor en cualquiera de las unidades disponibles en CSS, como píxeles y porcentaje.

columns.- Esta propiedad nos permite declarar los valores de las propiedades column-count

y column-width al mismo tiempo.

El elemento, cuyo contenido queremos dividir en columnas, es un elemento común y su contenido se declara del mismo modo que lo hemos hecho antes. El navegador se encarga de crear las columnas y dividir el contenido por nosotros. El siguiente ejemplo incluye un elemento **<main>** con un texto extenso que vamos a presentar en dos columnas.



Nada mejor para entenderlo que probar el siguiente código. Y luego, el último de la hoja.

ejemplocolumnas.html **columnas.css**

HTML

```
<body>
  <div class="rotulo">
    CAPÍTULO 60: De lo que sucedió a don Quijote yendo a Barcelona (segunda parte)
  </div>

  <main>
    Levantóse Sancho, y desvióse de aquel lugar un buen espacio; y, yendo a arrimarse a otro
    árbol, sintió que le tocaban en la cabeza, y, alzando las manos, topó con dos pies de persona,
    con zapatos y calzas. Tembló de miedo; acudió a otro árbol, y sucedióle lo mismo. Dio voces
    llamando a don Quijote que le favoreciese. Hízolo así don Quijote, y, preguntándole qué le había
    sucedido y de qué tenía miedo, le respondió Sancho que todos aquellos árboles estaban llenos
    de pies y de piernas humanas. Tentólos don Quijote, y cayó luego en la cuenta de lo que podía
    ser, y díjole a Sancho:
    <p>-No tienes de qué tener miedo, porque estos pies y piernas que tientes y no ves, sin
    duda son de algunos forajidos y bandoleros que en estos árboles están ahorcados; que por aquí
    los suele ahorcar la justicia cuando los coge, de veinte en veinte y de treinta en treinta; por donde
    me doy a entender que debo de estar cerca de Barcelona. </p>
  </main>
</body>
```

CSS

```
.rotulo{
  width:900px;
  margin: 20px auto;
  padding: 10px;
  text-align: center;
  background: #ddddff;
  font: bold 26px Arial,Verdana;
  color: #000080;
  border: 5px solid #000080;
  border-radius: 20px;
}
```

```
main {
  width: 900px;
  margin: 20px auto;
  padding: 10px;
  border: 5px solid #000080;
  border-radius: 20px;
  background: #eeeeff;
  font: normal 18px Arial, Verdana;
  color: #000080;
  text-align:justify;
  column-count: 3;
  column-gap: 20px;
}
```

Resultado mostrado por el navegador

CAPÍTULO 60: De lo que sucedió a don Quijote yendo a Barcelona (segunda parte)

Levantóse Sancho, y desvióse de aquel lugar un buen espacio; y, yendo a arrimarse a otro árbol, sintió que le tocaban en la cabeza, y, alzando las manos, topó con dos pies de persona, con zapatos y calzas. Tembló de miedo; acudió a otro árbol, y sucedióle lo mismo. Dio voces llamando a don Quijote que le

favoreciese. Hízolo así don Quijote, y, preguntándole qué le había sucedido y de qué tenía miedo, le respondió Sancho que todos aquellos árboles estaban llenos de pies y de piernas humanas. Tentólos don Quijote, y cayó luego en la cuenta de lo que podía ser, y díjole a Sancho:

-No tienes de qué tener miedo, porque estos pies y piernas que tientes y no vees, sin duda son de algunos forajidos y bandoleros que en estos árboles están ahorcados; que por aquí los suele ahorcar la justicia cuando los coge, de veinte en veinte y de treinta en treinta; por donde me doy a entender que debo de estar cerca de Barcelona.

La propiedad `column-gap` define el tamaño del espacio entre las columnas. Esta separación es solo espacio vacío, pero CSS ofrece las siguientes propiedades para generar una línea que ayude al usuario a visualizar la división.

`column-rule-style`.- Define el estilo de la línea para la división. Los valores disponibles son `hidden` (por defecto), `dotted`, `dashed`, `solid`, `double`, `groove`, `ridge`, `inset`, y `outset`.

`column-rule-color`.- Especifica el color de la línea usada para representar la división.

`column-rule-width`.- Especifica el ancho de la línea usada para presentar la división.

`column-rule`.- Propiedad que permite definir todos los valores de la línea al mismo tiempo.

El ejemplo define la propiedad **column-rule** que crea una línea de 5 píxeles, sólida y de color azul oscuro.

```
main {  
  width: 900px;  
  margin: 20px auto;  
  padding: 10px;  
  border: 5px solid #000080;  
  border-radius: 20px;  
  background: #eeeeff;  
  font: normal 18px Arial, Verdana;  
  color: #000080;  
  text-align: justify;  
  column-count: 3;  
  column-gap: 20px;  
  column-rule: 5px solid #000080;  
}
```

Interpretación del navegador

CAPÍTULO 60: De lo que sucedió a don Quijote yendo a Barcelona (segunda parte)

Levantóse Sancho, y desvióse de aquel lugar un buen espacio; y, yendo a arrimarse a otro árbol, sintió que le tocaban en la cabeza, y, alzando las manos, topó con dos pies de persona, con zapatos y calzas. Tembló de miedo; acudió a otro árbol, y sucedióle lo mismo. Dio voces llamando a don Quijote que le

favoreciese. Hízolo así don Quijote, y, preguntándole qué le había sucedido y de qué tenía miedo, le respondió Sancho que todos aquellos árboles estaban llenos de pies y de piernas humanas. Tentólos don Quijote, y cayó luego en la cuenta de lo que podía ser, y díjole a Sancho:

-No tienes de qué tener miedo, porque estos pies y piernas que tientes y no ves, sin duda son de algunos forajidos y bandoleros que en estos árboles están ahorcados; que por aquí los suele ahorcar la justicia cuando los coge, de veinte en veinte y de treinta en treinta; por donde me doy a entender que debo de estar cerca de Barcelona.

15. Display.

La propiedad display define cómo se comporta un elemento en el flujo del documento, es decir, cómo se muestra y distribuye en la página.

Valores de display:

block: el elemento ocupa toda la línea y genera un salto de línea. (div, p, h1, section)

inline: ocupa sólo el ancho del contenido y no genera salto de línea (span, a, strong). No permite modificar el tamaño.

inline-block: se comporta como inline pero permite modificar el tamaño con width y height (button, img)

flex: activa el modelo de cajas flexibles (Flexbox)

inline-flex: es como flex pero con el comportamiento de inline

grid: Activa el modelo de rejilla

inline-grid: es como grid pero con el comportamiento de inline

none: oculta el elemento

table: permite que un elemento se comporte como una tabla

Ejemplos:

Realiza los siguientes ficheros ejemplosflex.html y ejemplosflex.css y prueba los distintos valores de display.

ejemplosflex.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Ejemplos Flex</title>
  <link rel="stylesheet" href="css/estiloflex.css">
</head>
<body>
  <div class="caja">Bloque 1</div>
  <div class="caja">Bloque 2</div>
  <span class="texto">Texto 1</span>

</body>
</html>
```

ejemplosflex.css

```
.caja{
    display: block;
    background: lightblue;
    padding: 10px;
    margin: 5px;
    width: 200px;
}
.texto {
    display: block;
    background-color: blanchedalmond;
}
```

15.1. Flexbox.

Vamos a ver más en detalle los modelos de caja flexible. Como se ha visto, el contexto de flexbox viene determinado por la definición de display como **flex** e **inline-flex**, que establecen un contenedor como flexible.

flex: convierte el contenedor en flexible que distribuye su contenido de manera flexible a lo largo del eje principal (por defecto el horizontal de izda a drcha). Los elementos secundarios dentro del contenedor se comportarán como bloques y se ajustarán automáticamente para llenar el contenedor en la dirección principal.

inline-flex: similar a flex pero el contenedor se comporta como un elemento en línea en lugar de un bloque, de manera que ocupará sólo el espacio necesario y permitirá otros elementos en línea. Veamos cómo aplicar cada uno de ellos.

En el mismo ejemplo anterior añade el siguiente código en el fichero html:

```
<h2>display: flex</h2>
  <div class="contenedor flex-container">
    <div class="item">Item 1</div>
```



```
<div class="item">Item 2</div>
<div class="item">Item 3</div>
</div>

<h2>display: inline-flex</h2>
<div class="contenedor inline-flex-container">
  <div class="item">Item 1</div>
  <div class="item">Item 2</div>
  <div class="item">Item 3</div>
</div>
```

...y el siguiente código en el fichero css:

```
.contenedor {
  border: 1px solid #ccc;
  background-color: #EEE;
  padding: 10px;
}

.flex-container {
  display: flex;
}

.inline-flex-container {
  display: inline-flex;
}

.item {
  border: 1px solid #ccc;
  padding: 10px;
  margin: 5px;
}
```

flex-wrap: es una propiedad que se aplica cuando hemos definido un contenedor como flex y que nos permite indicar qué elementos se deben trasladar cuando no hay suficiente espacio en el contenedor. Los valores son **nowrap**: los elementos pasan a la siguiente fila y se reduce su anchura para mostrarlos (defecto); **wrap**: los elementos pasan a la siguiente fila y conservan su anchura; **wrap-reverse**: los elementos pasan a la siguiente fila pero en sentido inverso al de su declaración.

Veamos un ejemplo donde aplicar esta propiedad:

flexwrap.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Ejemplo de flex-wrap</title>
```

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

```
<body>
  <h2>flex-wrap: no-wrap;</h2>
  <div id="flex-container">
    <div>1</div>
    <div>2</div>
    <div>3</div>
    <div>4</div>
    <div>5</div>
  </div>

  <h2>flex-wrap: wrap;</h2>
  <div id="flex-container1">
    <div>1</div>
    <div>2</div>
    <div>3</div>
    <div>4</div>
    <div>5</div>
  </div>
  <h2>flex-wrap: wrap-reverse;</h2>
  <div id="flex-container2">
    <div>1</div>
    <div>2</div>
    <div>3</div>
    <div>4</div>
    <div>5</div>
  </div>
</body>

</html>
```

flexwrap.css

```
#flex-container {
  display: flex;
  flex-wrap: no-wrap;
}

#flex-container div {
  width: 200px;
  height: 150px;
  color: #FFF;
  text-align: center;
  line-height: 150px;
  margin: 10px;
  font-size: 20px;
  font-family: Arial;
  background-color: #E67E22;
```

```
}

#flex-container1 {
  display: flex;
  flex-wrap: wrap;
}

#flex-container1 div {
  width: 200px;
  height: 150px;
  color: #FFF;
  text-align: center;
  line-height: 150px;
  margin: 10px;
  font-size: 20px;
  font-family: Arial;
  background-color: #C0392B;
}

#flex-container2 {
  display: flex;
  flex-wrap: wrap-reverse;
}

#flex-container2 div {
  width: 200px;
  height: 150px;
  color: #FFF;
  text-align: center;
  line-height: 150px;
  margin: 10px;
  font-size: 20px;
  font-family: Arial;
  background-color: #3498DB;
}
```

Resultado:

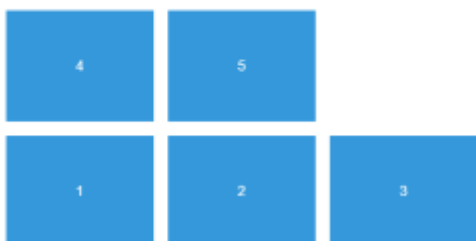
`flex-wrap: no-wrap;`



`flex-wrap: wrap;`



`flex-wrap: wrap-reverse;`



justify-content: define la justificación horizontal de los elementos hijos de un contenedor flexible.

Acepta los valores:

- **flex-start:** posiciona los elementos a la izquierda del contenedor (defecto)
- **flex-end:** posiciona los elementos a la derecha del contenedor.
- **center:** centra los elementos en el contenedor
- **space-between:** añade un espacio idéntico entre los elementos alineando el primero a la izquierda y el último a la derecha
- **space-around:** da espacio de forma regular a los elementos alineando el primero y el último a la mitad de espacio que el resto
- **space-evenly:** da espacio de forma regular a todos los elementos

Prueba estos valores en el css del ejemplo anterior y observa los resultados.

align-items: define la justificación vertical. Es necesario que esté definida la propiedad **flex-direction:row**

Acepta los valores:

- **stretch:** los elementos se amplían verticalmente para ocupar toda la altura del contenedor (defecto)
- **flex-start:** los elementos se colocan en la parte superior del contenedor.
- **flex-end:** los elementos se colocan en la parte inferior del contenedor.
- **center:** los elementos se colocan en el centro del contenedor
- **baseline:** los elementos se alinean sobre la línea de base del texto. Este es muy útil cuando los elementos tienen distintos elementos y quieres que estén alineados según su texto

```
<div><h2>1</h2></div>
<div>2</div>
<div><h3>3</h3></div>
```

Prueba con estos códigos en el último ejemplo realizado

```
#flex-container {
    height: 300px;
    display: flex;
    flex-wrap: no-wrap;
    flex-direction: row;
    align-items: flex-end;
}

#flex-container div {
    width: 300px;
    color: #FFF;
    text-align: center;
    line-height: 150px;
    margin: 10px;
    font-size: 20px;
    font-family: Arial;
    background-color: #E67E22;
}
```

flex-direction: determina la dirección en la que los elementos se colocan dentro de un contenedor flex. (fila o columna)

Acepta los siguientes valores:

- **row:** coloca los elementos en una fila (defecto).
- **row-reverse:** coloca los elementos en una fila pero en orden inverso.
- **column:** coloca los elementos en una columna.
- **column-reverse:** coloca los elementos en una columna pero en orden inverso.

Existen otras propiedades que se aplican sobre los elementos contenidos dentro del contenedor flex, como son **flex-grow:** define cómo un elemento debe crecer en relación con el resto de elementos del contenedor.

flex-grow:1; /*significa que todos los elementos crecen igual...*/

Vamos a probar con estos códigos:

Html:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ejemplo de flex-wrap</title>
    <link rel="stylesheet" href="css/flexwrap.css">
</head>

<body>
    <div class="flex-container">
        <div class="flex-item">Item 1</div>
        <div class="flex-item">Item 2</div>
```

```
        <div class="flex-item">Item 3</div>
    </div>
</body>

</html>
```

CSS

```
.flex-container {
    display: flex;
}
.flex-item {
    flex-grow: 1; /* Todos los elementos crecen de igual manera */
    border: 2px solid #161515;
    color: #FFF;
    text-align: center;
    line-height: 150px;
    margin: 10px;
    font-size: 20px;
    font-family: Arial;
    background-color: #E67E22
}
.flex-item:nth-child(3) {
    flex-grow: 3; /* Este elemento crece la mitad que los demás */
    background-color: #2ECC71;
}
```

flex-shrink: define cómo los elementos se encogen en relación con los demás cuando no hay espacio suficiente en el contenedor

flex-basis: establece el tamaño inicial de un elemento antes de que se distribuya el espacio restante

Prueba con este código en CSS sobre el ejemplo anterior.

```
.flex-container {
    display: flex;
    background-color: rgb(5, 116, 116);
    width: 500px; /* Ancho total menor que la suma de los flex-basis */
}
.flex-item {
    flex-basis: 200px;
    border: 2px solid #161515;
    color: #FFF;
    text-align: center;
    line-height: 150px;
    margin: 10px;
    font-size: 20px;
    font-family: Arial;
    background-color: #E67E22
}
```

```
}
/* Primer elemento */
.flex-item:first-child {
    flex-shrink: 0; /* Este elemento no se encogerá */
}
/* Segundo elemento */
.flex-item:nth-child(2) {
    flex-shrink: 2; /* Este elemento se encogerá el doble que el normal */
}
/* Tercer elemento */
.flex-item:nth-child(3) {
    flex-shrink: 1; /* Este elemento se encogerá normalmente */
}
```

flex: es una propiedad abreviada sobre los elementos flex que combina flex-grow, flex-shrink y flex-basis

flex: 1 0 100px; /*flex-grow: 1, flex-shrink: 0, flex-basis: 100px */