

I.E.S. DOMINGO PÉREZ MINIK



CICLO FORMATIVO DE GRADO SUPERIOR:
*"ADMINISTRACIÓN DE SISTEMAS
INFORMÁTICOS EN RED (L.O.E.)"*



MÓDULO:

Lenguajes de marcas y sistemas de
gestión de información

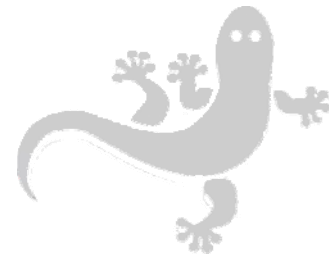
(4 horas semanales)





Índice

TEMA 4.- CSS.	3
1.- HOJAS DE ESTILO EN CASCADA	3
2.- SINTAXIS BÁSICA EN CSS	5
3.- APLICAR CSS A UN ELEMENTO HTML	5
4.- APLICAR CSS A TODA UNA PÁGINA WEB	7
4.1.- Etiqueta style:	7
4.2.- Confrontación de estilos:	8
5.- APLICAR CSS A TODO UN SITIO WEB	8
6.- LOS SELECTORES	9
6.1.- Selector universal:	9
6.2.- Selector de etiqueta:	10
6.3.- Selector descendente:	10
6.4.- Selector de clase:	12
6.5.- Combinando selectores:	14
6.6.- Selector de id:	15
6.7.- Selector de hijos:	16
6.8.- Selector adyacente:	16
6.9.- Selector de pseudo-clase:	16
6.10.- Selector de pseudo-elemento:	18
6.11.- Selector de atributos:	18
7.- EL MODELO DE CAJAS	19
7.1.- Alto y ancho:	21
7.2.- El margen:	21
7.3.- El relleno:	23
7.4.- El borde:	23
7.5.- Nota sobre el tamaño:	25
7.6.- El fondo:	26
8.- LA POTENCIA DE CSS: EL POSICIONAMIENTO	29
8.1.- Posicionamiento normal:	30
8.2.- Posicionamiento relativo:	30
8.3.- Posicionamiento absoluto:	32
8.4.- Posicionamiento fijo:	34
8.5.- Posicionamiento flotante:	34
9.- MOSTRAR Y OCULTAR. DESBORDAMIENTO Y CAPAS	38



TEMA 4.- CSS.

1.- Hojas de estilo en cascada.

Las hojas de estilo en cascada, en adelante **CSS** (*Cascading Style Sheets*), aparecieron como una consecuencia de las limitaciones del HTML para aplicar estilos de diseño a los distintos elementos. En especial surgieron porque cada navegador mostraba las páginas “a su manera” y los programadores pronto demandaron un estándar que permitiera asegurar que una misma página se presentara en pantalla de forma similar independientemente del navegador utilizado.

Se crearon a partir de dos estándares:

- CHSS (Cascading HTML Style Sheets), creado por Håkon Wium Lie.
- SSP (Stream-based Style Sheet Proposal), creado por Bert Bos.

En 1995, el W3C decidió apostar por el desarrollo y estandarización de CSS y lo añadió a su grupo de trabajo de HTML. A finales de 1996, el W3C publicó la primera recomendación oficial, conocida como "**CSS nivel 1**".

A principios de 1997, el W3C decide separar los trabajos del grupo de HTML en tres secciones: el grupo de trabajo de HTML, el grupo de trabajo de DOM y el grupo de trabajo de CSS.

El 12 de Mayo de 1998, el grupo de trabajo de CSS publica su segunda recomendación oficial, conocida como "CSS nivel 2". La versión de CSS que utilizan todos los navegadores de hoy en día es **CSS 2.1**, una revisión de CSS 2 que aún se está elaborando (la última actualización es del 19 de julio de 2007). Al mismo tiempo, la siguiente recomendación de CSS, conocida como "CSS nivel 3", continúa en desarrollo desde 1998 y hasta el momento sólo se han publicado borradores.



La adopción de CSS por parte de los navegadores ha requerido un largo periodo de tiempo. El mismo año que se publicó CSS 1, Microsoft lanzaba su navegador Internet Explorer 3.0, que disponía de un soporte bastante reducido de CSS. El primer navegador con soporte completo de CSS 1 fue la versión para Mac de Internet Explorer 5, que se publicó en el año 2000. Por el momento, ningún navegador tiene soporte completo de CSS 2.1.

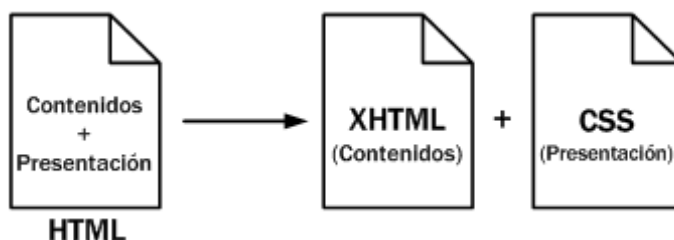
Nos encontramos por tanto con el mismo problema que tiene HTML y DOM, no todos los navegadores reconocen el 100% del estándar. En concreto cada navegador incorpora un programa, que llamamos **motor**, que interpreta el código tanto de HTML como de CSS. En la siguiente tabla se muestra la compatibilidad con CSS para distintos navegadores:

Navegador	Motor	CSS 1	CSS 2.1	CSS 3
Internet Explorer	Trident	Completo desde la versión 6.0	Casi completo desde la versión 7.0	Prácticamente nulo
Firefox	Gecko	Completo	Casi completo	Selectores, pseudo-clases y algunas propiedades
Safari	WebKit	Completo	Casi completo	Todos los selectores, pseudo-clases y muchas propiedades
Opera	Presto	Completo	Casi completo	Todos los selectores, pseudo-clases y muchas propiedades
Google Chrome	WebKit	Completo	Casi completo	Todos los selectores, pseudo-clases y muchas propiedades

La especificación o norma oficial que se utiliza actualmente para diseñar páginas web con CSS es la versión **CSS 2.1**, actualizada por última vez el 19 de julio de 2007 y que se puede consultar libremente en <http://www.w3.org/TR/CSS21/>



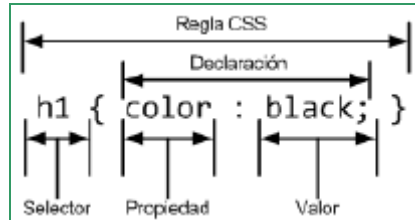
Recuerda que perseguimos el objetivo de separar contenidos de presentaciones:



Así CSS se encargará de cuestiones que afectan a la presentación: márgenes, bordes, alineados, colores, etc.

2.- Sintaxis básica en CSS.

CSS tiene su propia sintaxis, esto es, su propia manera de ser escrito. A continuación vamos a poner la forma básica de escribir código en este lenguaje:



1.- Como ves la primera parte se llama **selector**. El selector indica a qué elementos de HTML se le aplicará el estilo. En el ejemplo anterior se aplicará a todas las etiquetas <h1>. Pero existen muchos tipos de selectores y será necesario conocerlos para comprender la potencia de CSS. Hablaremos de ellos en breve.

2.- Las definiciones de estilo aplicadas a cada selector se encerrarán entre llaves.

3.- A continuación vienen los estilos, que siempre se compondrán de propiedad seguida de dos puntos y valor. En el ejemplo la propiedad color tiene el valor Black, esto es, el color de letra será el negro.

4.- Si quieres aplicar varios estilos los separamos por punto y coma.

RECUERDA:

```
Selector {propiedad1:valor1;
          propiedad2:valor2;
          ....
          propiedadn:valorn;
}
```



Los saltos de línea sólo los ponemos por claridad en el código, no son necesarios.

3.- Aplicar CSS a un elemento HTML.

La manera más simple, y también más ineficiente de aplicar un estilo escrito en CSS a una etiqueta HTML de nuestra página es utilizar el atributo **style**. Todas las etiquetas HTML soportan el atributo style. El código escrito dentro de las comillas en este atributo estará escrito en CSS. Mira el ejemplo:

```
<!-- ... código HTML ... -->
<p style="color:#FF0000;background-color:#FFFFFF">
    Texto en rojo con fondo blanco
</p>
<p style="border:solid red 3px;color:blue">
    Texto en azul con borde rojo de 3 píxeles
</p>
```





PRÁCTICA 01.- Retomamos la práctica xhtml realizada en el tema anterior, que tenía este aspecto:

- 1) Los textos “nombre” y “descripción” deben aparecer en color verde.
- 2) El input para el nombre y el textarea para descripción deben aparecer con fondo gris claro y letra azul.
- 3) El input para el precio debe aparecer con fondo rojo y letra de tipo Arial 16 negrita.
- 4) El texto “descuento 5%” debe aparecer con un borde verde discontinuo de 5 píxeles.
- Para hacer esta práctica tendrás que conocer los nombres de las propiedades CSS, busca la información en Internet.
<http://www.w3c.es/divulgacion/guiasreferencia/css21/>
- 5) Cuando hayas terminado usa el validador CSS publicado en el w3c para ver si el documento es correcto.
http://jigsaw.w3.org/css-validator/#validate_by_upload


4.- Aplicar CSS a toda una página web.

Imagina ahora que tienes una página web con un formulario para recoger datos de un alumno. En esa página hay 100 entradas de texto y quieres que todas tengan la misma apariencia. ¿Qué harás? Podríamos usar el atributo style en todas las entradas de texto y copiar los estilos CSS, pero tendríamos que hacerlo en 100 partes de la página. ¿Y si hemos colocado, por ejemplo, el fondo gris y queremos que ahora las 100 entradas de texto tengan fondo verde claro? Otra vez a revisar toda la página. Esto es mucho trabajo y podemos simplificarlo usando la etiqueta style.

4.1.- Etiqueta style:

La etiqueta style (ojo, es una etiqueta HTML y no nos estamos refiriendo al atributo style que se puede aplicar a todas las etiquetas) nos va a permitir **agrupar** todos los estilos en una única zona.

Se declara dentro de HEAD. Observa el ejemplo:



```
<head>
  <title> Título </title>

  <style type="text/css">
    /* Esto es un comentario en CSS */
    p {color:white;
      background-color:red
    }
    h2 {
      border: 2px solid red
    }
  </style>
</head>
```

Con este ejemplo vemos cómo es el funcionamiento básico de la etiqueta style. También vemos cómo se escriben los comentarios en CSS. Y por último estamos utilizando dos selectores: p y h2. Lo que significa es que en esta página web todos los párrafos <p> aparecerán con letra blanca y fondo rojo, mientras que todas las cabeceras de nivel 2 <h2> aparecerán con un borde continuo y rojo de dos píxeles de ancho.

Volviendo al ejemplo que citábamos al comienzo de este apartado, la página con 100 entradas de texto, nos bastará declarar una única regla CSS que afecte a estos botones y tendremos controlado el estilo de todos ellos.



PRÁCTICA 02.- En el ejercicio anterior agrega una regla CSS que ponga todas las etiquetas <legend> en color violeta.



PRÁCTICA 03.- Al ejercicio anterior agrégale otra regla CSS que coloque todas las etiquetas <label> en color azul y subrayado.



4.2.- Confrontación de estilos:

Por el momento hemos visto dos formas de aplicar estilos. La primera era colocando el **atributo style** a cualquier etiqueta (y el estilo afectaba sólo a esa etiqueta). La segunda es usar la **etiqueta style** dentro de <head>, y ahí crear reglas de CSS que afectaban a un tipo de etiqueta.

¿Qué pasa cuando un elemento HTML tiene dos estilos aplicados, uno con el atributo style y otro con una regla CSS?

Observa el resultado de la práctica 3. Las dos primeras etiquetas continúan estando en verde y no se han cambiado al color azul. Sin embargo TODAS las etiquetas aparecen subrayadas.

RECUERDA:

Ante una confrontación de estilos se aplicará la propiedad que esté más cerca de la etiqueta <HTML>.

Primero se aplicará lo definido en el atributo style de esa etiqueta. Y después las propiedades definidas en las reglas CSS.



5.- Aplicar CSS a todo un sitio web.

Imagina ahora que estamos desarrollando un sitio web que tiene 20 páginas. En todas ellas queremos aplicar los mismos estilos, para que todas tengan nuestra imagen corporativa. ¿Qué podemos hacer?

Tendríamos que copiar las reglas CSS en todas las páginas. Y de nuevo si cambiamos uno de los estilos habría que revisar el conjunto de las páginas. De nuevo trabajo repetitivo que podemos evitar si en este caso **sacamos el código CSS a un fichero externo**. Eso lo logramos escribiendo el código CSS en un fichero de texto plano. Le daremos el nombre que queramos (ejemplo: misestilos.css o misestilos.inc). La extensión .css es opcional, no obligatoria pero será recomendable usarla.

Lo que nos faltaría por hacer es **enlazar** el fichero con los estilos en todas nuestras páginas. De nuevo dentro de <head> usaremos la etiqueta **link**. Antes de ver la sintaxis de esta etiqueta debemos decir que una misma página web puede tener muchos ficheros de estilos enlazados y no uno solo.

```
<link rel="stylesheet" type="text/css" href="url" media="medio">
```



- * **rel**: indica el tipo de relación que tiene el recurso enlazado (en este caso, el archivo CSS) y la página HTML. Para los archivos CSS, siempre se utiliza el valor stylesheet.
- * **type**: indica el tipo de recurso enlazado. Sus valores están estandarizados y para los archivos CSS su valor siempre es text/css.
- * **href**: indica la URL del archivo CSS que contiene los estilos. La URL indicada puede ser relativa o absoluta y puede apuntar a un recurso interno o externo al sitio web.
- * **media**: indica el medio en el que se van a aplicar los estilos del archivo CSS. Existen una serie de medios predefinidos como screen (pantalla), print (impresora), projection (proyector), tv (televisor), braille (dispositivos para discapacitados visuales), etc.

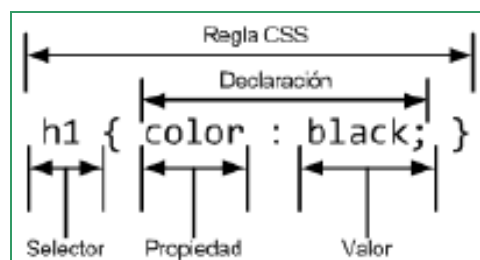
La etiqueta link no es la única manera de enlazar archivos externos. También se pueden utilizar reglas especiales de CSS llamadas reglas de tipo import, pero no las veremos hasta hablar de selectores.



PRÁCTICA 04.- En el ejercicio anterior elimina la etiqueta style pasando las reglas CSS definidas a un archivo llamado miestilo.css y enlazándolo.

6.- Los selectores.

Recordemos primero el esquema de las reglas CSS:



Los selectores nos indican a quién se va a aplicar el estilo. Existen distintos tipos de selectores en CSS y sirven para aplicar estilos a distintos elementos. Veamos algunos de ellos:

6.1.- Selector universal:

Sirve para definir un estilo base, que se aplica a todos los elementos de una página. Se denota por un asterisco. Aquí va un ejemplo:

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




Con este ejemplo todos los elementos de la página tendrán letra roja y aparecerán subrayados.




6.2.- Selector de etiqueta:

Es el que ya hemos utilizado. Afecta a todas las etiquetas de ese mismo tipo en una página web.




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



```
p, h1, h4 {color: red;
  text-decoration: underline
}
```

En este caso serán todos los párrafos de la página los que tendrán letra roja y subrayada. Si queremos aplicar el mismo estilo a varias etiquetas podemos ahorrar espacio separando los selectores por comas. En el segundo ejemplo el mismo estilo se aplica a los párrafos y a las cabeceras de nivel 1 y 4.

Imagina ahora que p, h1 y h4 tienen en común el texto en color rojo y fondo azul, pero cada uno llevará una letra de distinto tamaño. Lo que hacemos es agrupar las propiedades comunes y luego definir las que son propias de cada etiqueta. Observa:



```
p, h1, h4 {color: red;
  background-color: blue
}
```

```
p {font-size: 4em}
h1 {font-size: 6em}
h4 {font-size: 2em}
```

6.3.- Selector descendente:

En HTML un elemento es descendiente de otro si se encuentra dentro de su etiqueta. Por ejemplo:

```
<span> Bienvenido: </span>
```

```
<p> Hola <span> este es un texto </span> en HTML</p>
```

Aquí el segundo elemento es descendiente de <p> porque está dentro del párrafo. El primero no será descendiente. Si queremos aplicar un estilo a los que sean descendientes de un <p> utilizaremos el selector descendente. Para hacerlo basta poner las etiquetas separadas por un **espacio en blanco**:



```
p span {color: orange}
```

```
p a {border: solid blue 3px}
```

En este ejemplo los que estén dentro de <p> tendrán texto en naranja, mientras que los enlaces <a> que estén dentro de <p> tendrán borde azul de 3 píxeles.



PRÁCTICA 05.- Observa la siguiente página web con párrafos del capítulo 4 del Quijote. Para obtener este resultado sigue los siguientes apartados:

Don Quijote de La Mancha

Capítulo cuarto

De lo que le sucedió a nuestro caballero cuando salió de la venta

La del alba sería cuando **Don Quijote** salió de la venta, tan contento, tan gallardo, tan alborozado por verse ya armado caballero, que el gozo le reventaba por las cinchas del caballo.

Mas viniéndole a la memoria los consejos de su huésped acerca de las prevenciones tan necesarias que había de llevar consigo, en especial la de los dineros y camisas, determinó volver a su casa y acomodarse de todo, y de un escudero, haciendo cuenta de recibir a un labrador vecino suyo, que era pobre y con hijos, pero muy a propósito para el oficio escudero de la caballería.

Con este pensamiento guió a **Rocinante** hacia su aldea, el cual casi conociendo la querencia, con tanta gana comenzó a caminar, que parecía que no ponía los pies en el suelo.

No había andado mucho, cuando le pareció que a su diestra mano, de la espesura de un bosque que allí estaba, salían unas **voces** delicadas, como de persona que se quejaba; y apenas las hubo oído, cuando dijo:

gracias doy al cielo por la merced que me hace, pues tan presto me pone ocasiones delante, donde yo pueda cumplir con lo que debo a mi profesión, y donde pueda coger el fruto de mis buenos deseos.

estas **voces** sin duda son de algún menesteroso o menesterosa, que ha menester mi favor y ayuda y volviendo las riendas encaminó a **Rocinante** hacia donde le pareció que las **voces** salían

Obtén el texto de Internet <http://www.elmundo.es/quijote/capitulo.html?cual=4>

En el código xhtml realiza los siguientes cambios:

- 1) El título es una cabecera de nivel 1.
- 2) El capítulo es una cabecera de nivel 3.
- 3) El resumen del capítulo es una cabecera de nivel 5, al igual que el párrafo que aparece con tabulación.
- 4) El resto de párrafos utilizan <p>.
- 5) Encierra las palabras “Don Quijote”, “Rocinante” y “voces” dentro de span.

Aplica los siguientes estilos:

- 1) Todos los elementos de la página tienen color verde (selector universal).
- 2) Los párrafos y las cabeceras de nivel 5 tienen color azul (selector etiqueta).
- 3) Las cabeceras de nivel 5 tienen un margen izquierdo de 100 píxeles.
- 4) La cabecera de nivel 1 tiene un espaciado de palabras de 20 píxeles.
- 5) Aplica un tachado en todos los span que están dentro de p (selector descendente). ¿Qué pasa con el color de la letra en estos span?



PRÁCTICA 06.- Analiza el código y responde ¿Qué hacen las siguientes reglas CSS?

- 1) table p {text-align: justify}
- 2) table p a {word-spacing: 10px}
- 3) table, p, a {background-image: gato.gif}
- 4) p * a {border-top: solid 2px green}



6.4.- Selector de clase:

Es un selector que nos va a permitir crear una “clase de estilos” con el nombre que nosotros deseemos. Imagina el siguiente código xhtml para informar de una noticia:

```
<p>Aviso de tormenta sobre Canarias:</p>
<p>El centro de vigilancia de huracanes ha avisado de la formación de una
tormenta tropical llamada Delta en el centro del Océano Atlántico.</p>
<p>Según las predicciones esta tormenta puede abatirse sobre las Islas Canarias
en unas seis horas, con vientos de hasta 100 kilómetros/hora.
<p>Fuente Minik-News.
```

Queremos que este texto se presente como una noticia. El primer párrafo será el rótulo, los dos siguientes serán el cuerpo y el último será el pie de la noticia.

Con los selectores vistos hasta ahora no podremos aplicar distintos estilos a una misma etiqueta, en este caso `<p>`. Para eso podemos usar el selector de clase. Mira cómo definimos las clases (**ojo que llevan un punto antes del nombre**):

```
.rotulo {color: red;
        font-size: 16px}
```

```
.cuerpo {color: black;
        font-size: 12px}
```

```
.pie {color: gray;
      text-align: right;
      font-size: 9px}
```

Y en el código HTML se aplica la clase que queramos con el atributo **class**:

```
<p class="rotulo"> Aviso de tormenta sobre
Canarias.</p>
<p class="cuerpo"> El centro de ... </p>
<p class="cuerpo"> Según las ... </p>
<p class="pie">Fuente...</p>
```

El selector de clase es muy importante. Nos permite generar una clase de estilos y luego aplicarla a las etiquetas que queramos. Una misma clase puede aplicarse a cualquier etiqueta HTML.



PRÁCTICA 07.- Realiza las siguientes clases de estilos:

- * Tabla-borde: define un borde sólido azul oscuro de 1 píxel de ancho.
- * Cabecera: color de fondo azul oscuro, color de letra blanco, letra de 16 píxeles y negrita.
- * Cuerpo: color de fondo gris claro, color de letra azul oscuro, letra de 12 píxeles justificada por ambos lados.
- * Pie: color de fondo azul oscuro, letra blanca de 10 píxeles alineada a la derecha.

Aplica estas clases a una página web con tablas de forma que quede algo similar a la siguiente:

Lenguaje HTML Lenguaje de marcas genérico para el etiquetado de texto. Incorpora etiquetas para contenido y etiquetas para presentación. El contenido y la presentación se encuentran juntos Tema 2	Lenguaje XHTML Lenguaje de marcas que hereda toda la sintaxis de HTML pero incorpora restricciones en la escritura. Este lenguaje desciende del XML. Se encarga de los contenidos, dejando a CSS las cuestiones que afectan a la presentación. Tema 3
Lenguaje CSS Lenguaje que se encarga de la presentación, esto es, del aspecto de las páginas web. Permite fijar elementos como márgenes, bordes, colores, tamaños y posición Se basa en la definición de reglas. Las reglas comienzan con el selector, que indica qué partes del documento HTML se verán afectadas. Después del selector se abren llaves y se definen los estilos con la sintaxis propiedad:valor. Tema 4	LENGUAJES DE MARCAS Asignatura de primer curso de ADMINISTRACIÓN DE SISTEMAS INFORMÁTICOS EN RED. Código: LND

Observa que las tablas tienen borde y luego varias filas, la primera es una cabecera, la segunda un cuerpo y la tercera un pie.



PRÁCTICA 08.- Cambia tu página para que tenga el siguiente aspecto:

Lenguaje HTML Lenguaje de marcas genérico para el etiquetado de texto. Incorpora etiquetas para contenido y etiquetas para presentación. El contenido y la presentación se encuentran juntos Tema 2	Lenguaje XHTML Lenguaje de marcas que hereda toda la sintaxis de HTML pero incorpora restricciones en la escritura. Este lenguaje desciende del XML. Se encarga de los contenidos, dejando a CSS las cuestiones que afectan a la presentación. Tema 3
Lenguaje CSS Lenguaje que se encarga de la presentación, esto es, del aspecto de las páginas web. Permite fijar elementos como márgenes, bordes, colores, tamaños y posición Se basa en la definición de reglas. Las reglas comienzan con el selector, que indica qué partes del documento HTML se verán afectadas. Después del selector se abren llaves y se definen los estilos con la sintaxis propiedad:valor. Tema 4	LENGUAJES DE MARCAS Asignatura de primer curso de ADMINISTRACIÓN DE SISTEMAS INFORMÁTICOS EN RED. Código: LND

Pasa el validador de CSS a tu página y comprueba que tu código es correcto.



PRÁCTICA 09.- En clase te indicaré una página profesional con estilos bien definidos. Tendrás que localizar el/los ficheros de estilos empleados, así como la clase de estilos aplicado a un elemento concreto de la página.



6.5.- Combinando selectores:

Ya hemos visto lo potente que es el selector de clase. Podemos crear, por ejemplo, la clase `titulo` para destacar algunos elementos. Una vez creada la clase podemos aplicarla en una etiqueta `<td>`, en otra etiqueta `<p>`, o en cualquier etiqueta deseada.

En ocasiones nos puede interesar modificar una clase cuando se aplica a un tipo de etiqueta concreta. Esto lo logramos combinando el selector de etiqueta con el selector de clase, como mostramos en el siguiente ejemplo:

```
td.titulo {color: red;
           font-size: 16px}
```



Como ves colocamos primero el selector de etiqueta y luego el selector de clase (**separado por un punto**).

```
a.titulo {color: blue;
          font-size: 12px}
```

En este ejemplo los `td` que tengan `class="titulo"` aplicarán letra roja de tamaño 16.

Sin embargo los enlaces que tengan `class="titulo"` aplicarán letra azul de tamaño 12.

En este punto es muy importante dominar la sintaxis de los selectores estudiados, puesto que las combinaciones pueden inducir a error. Hagamos un recordatorio:

```
* { ... }      → selector universal
p, h3, a { ... } → selector de etiquetas, la “y”.
p h3 a { ... } → selector descendente, el espacio denota “dentro de”
.titulo { ... } → selector de clase, empieza con un “punto”.
```



Estos selectores los podremos combinar para obtener resultados diferentes.



PRÁCTICA 10.- Indica a quién se aplicará estos estilos.

- 1) `a.titulo { ... }`
- 2) `a .titulo { ... }`
- 3) `a, .titulo { ... }`
- 4) `p a.titulo { ... }`

También es posible aplicar varias clases a una misma etiqueta `xhtml`. Para hacerlo dentro del atributo `class` se escribirán los nombres de las clases **separados por un espacio en blanco**. Mira el ejemplo:

```
<a href="yo.html" class="titulo enlace txsubrayado"> Mis datos </a>
```



En este ejemplo este enlace incorpora el código css de tres clases:

```
.titulo {font-size: 20px }
.enlace { color: gray }
.txsubrayado { text-decoration: underline }
```

Y por último observando el ejemplo anterior podrás ver que en ocasiones habrá confrontación de estilos. Si la clase `.titulo` incorporara letra roja y la clase `.enlace` declarara la letra verde, ¿de qué color será al final la letra?

En este sentido es importante que en tus páginas no exista complejidad de combinación de selectores salvo que realmente la necesites. Siguiendo con las combinaciones en el ejemplo anterior podemos hacer un selector de clase múltiple:

```
.titulo.txsubrayado {color: orange }
```

Esta regla CSS indica que se aplicará a los elementos de la página en cuyo atributo `class` estén por lo menos las clases `.titulo` y `.txsubrayado`.

6.6.- Selector de id:

Es el selector que utilizaremos cuando queramos aplicar un estilo a un único elemento de nuestra página. Recuerda que el atributo `id` lo poníamos en las etiquetas HTML para darles un nombre único. En una misma página no deberían existir dos elementos con el mismo identificador.

```
<p> Este es un párrafo normal </p>
<p id="especial"> Este párrafo será diferente al resto </p>
<p> Este párrafo vuelve a ser normal </p>
```

Para aplicar un estilo utilizando el selector de `id` la regla CSS comenzará con el símbolo de almohadilla `#`:

```
#especial { margin:10px; color:red }
```

Lógicamente este selector también podremos combinarlo:

`a#contacto {...}` → esta regla se aplicará a todos los enlaces con atributo `id="contacto"`. ¿Pero no hemos dicho que en una página no debería haber dos elementos con el mismo `id`? Imagina que tenemos un site con 50 páginas y al final de todas tenemos un enlace para contactar. Aquí sí resulta útil esta regla. En todas las páginas colocaremos el `id="contacto"` sólo en esos enlaces.



PRÁCTICA 11.- Indica a quién se aplicará estos estilos.

- 1) `#contacto, .titulo, a {...}`
- 2) `p #contacto {...}`
- 3) `p#contacto {...}`
- 4) `.aviso .avisoerror {...}`
- 5) `div.titulo div.novedad {...}`
- 6) `div.titulo span.urgente {...}`
- 7) `h1 em, h2 em, h3 em {...}`

NOTA: tanto en el selector de clase como en el selector de `id`, los nombres asignados a las clases o a los `id` son **case-sensitive**. No es igual `class="rotulo"` que `class="Rotulo"`.



6.7.- Selector de hijos:

Observa el siguiente código:

```
<p>Texto <strong> en negrita <em> y parte en énfasis </em></strong></p>
<p> Texto <em> solo con parte en énfasis </em></p>
```

Si aplicamos el selector descendiente:
`p em {color: blue}`

Haremos que el texto que está en énfasis dentro de los dos párrafos se muestre en azul. Esto es porque la etiqueta `` es descendiente (está dentro de) la etiqueta `<p>` en los dos casos.

Con el selector hijo lo que indicaremos es que se aplique sólo a **los descendientes directos**. Utiliza el **signo mayor que**:

```
p > em {color: blue}
```



Ahora sólo será azul el segundo ``, puesto que el primero no es un hijo directo de `<p>`.

6.8.- Selector adyacente:

Se trata de un selector avanzado y complejo. Se usa relativamente poco. Utiliza un **signo más** en la sintaxis.

```
h3 + h5 {color: blue}
```



Coloca letra azul a aquellas etiquetas `h3` y `h5` que sean hermanas (tienen el mismo elemento padre) y además aparece un `<h3>` inmediatamente seguido de un `<h5>`.

```
<h1> Título </h1>
<h3> Subtítulo 1 </h3>
<h3> Subtítulo 2 </h3>
<h5> Bienvenidos a la página </h5>
<h5> Programando con CSS </h5>
```




En este ejemplo sólo aparecerán en azul las líneas 3 y 4. El resto no cumple la regla CSS definida.

6.9.- Selector de pseudo-clase:

Las pseudo-clases son un selector que ya existía en CSS 1.0, pero que se ha ampliado. Este selector se aplica no tanto a una etiqueta HTML sino al estado en que esa etiqueta está. El ejemplo típico que existía en CSS se aplicaba sólo a los enlaces, y así teníamos una manera de aplicar un estilo a los enlaces visitados, a los enlaces activos, y enlaces no visitados.

Esto es lo que existía en CSS 1.0:



```
a:link {color: red} /*afecta a enlaces no visitados */
a:visited {color: black} /*afecta a enlaces ya visitados */
a:active {color:blue} /*afecta a enlaces en el momento en
que se va a pulsar sobre ellos */
```

En CSS 2 se amplían las pseudo-clases, y pueden aplicarse a casi todos los elementos. Observa que en su sintaxis utilizan un **signo de dos puntos**. Las pseudo-clases disponibles en CSS 2 son las siguientes (ojo que pueden no estar soportadas por tu navegador):

- 1) **:first-child** → afecta al primer elemento que es hijo de otro. Ejemplo: queremos que el primero de los párrafos de un div tenga sangría, haremos:

```
div > p:first-child {text-indent: 5};
```

- 2) **:link** y **:visited** → permiten configurar el estilo para los enlaces no visitados (:link) y para los enlaces ya visitados (:visited). Ejemplo: pongamos los enlaces visitados en color gris:

```
a:visited {color:gray}
```

- 3) **:hover**, **:active** y **:focus** → permiten configurar el estilo para distintas etiquetas cuando el ratón la está señalando (:hover), cuando vamos a activar el elemento por ejemplo haciendo click del ratón en él (:active) y por último cuando el elemento recibe el cursor (:focus). Ejemplo: hagamos que cuando el cursor pase por encima de un span éste se ponga con fondo verde:

```
span:hover {background-color:green}
```

- 4) **:lang** → en html las etiquetas pueden tener al atributo lang que especifica el idioma del texto. Por ejemplo: `<p lang="en">` indica que ese texto está escrito en inglés. Es=español, Fr=francés, de=alemán, ... Mediante la pseudo-clase :lang podemos aplicar un estilo a los elementos escritos en un determinado idioma. Ejemplo: a los párrafos en francés de nuestra página le aplicaremos un margen de 50 píxeles, una letra de tamaño 10 píxeles y las comillas dobles que se usen serán << y >>.

```
p:lang(fr) { margin-left: 50px;
              font-size: 10px;
              quotes: '« ' ' »' }
```

Nota: algunas versiones de IExplorer no reconocen este selector.



PRÁCTICA 12.- Retoma la práctica 8 y convierte los textos en los pie de tabla en enlaces que apuntes a los dossiers del temario. Cuando se pasa por encima de ellos se pondrán con fondo rojo, y cuando estén visitados se pondrán en color de texto azul oscuro. Además los primeros párrafos de las tablas deben tener indentación de 60 píxeles.



6.10.- Selector de pseudo-elemento:

Los pseudo-elementos también son una novedad en CSS 2. Son cinco y son los siguientes:

1) Selector **:first-letter** → Afecta a la primera letra del contenido de una etiqueta. Típicamente se usa con los párrafos, los span o los div. Ejemplo: hacer que la primera letra de un párrafo tenga tamaño 20 píxeles, negrita y mayúscula.

```
p:first-letter { font-size:20px; font-weight:bold; text-transform: uppercase }
```

2) Selector **:first-line** → Afecta a la primera línea del contenido de una etiqueta. Ejemplo: la primera línea de nuestros párrafos tendrá una indentación de 50 píxeles:

```
p:first-line { text-indent: 50px }
```

Las únicas propiedades que pueden aplicarse a este pseudo-elemento son:

propiedades de la fuente, propiedades del color, propiedades del fondo, 'word-spacing', 'letter-spacing', 'text-decoration', 'vertical-align', 'text-transform', 'line-height', 'text-shadow' y 'clear'.

3) Selectores **:before** y **:after** → Significan antes (before) y después (after). Permiten que insertemos un contenido (típicamente texto o imagen) antes o después de un elemento, a la vez que damos estilo a ese contenido. Usan la propiedad content, observa los ejemplos:

H2:before {content: "Nuevo capítulo: "} → antes de cada <hr> aparecerá el texto “Nuevo Capítulo: “.

table:after {content: url("icono.gif")} → tras cada tabla se mostrará la imagen icono.gif.

body:after {content: "Fin"; color: red;} → al final de la página aparecerá el texto Fin en rojo.

OJO: en algunas versiones de IExplorer no funciona este selector.

6.11.- Selector de atributos:

Son selectores avanzados que permiten seleccionar etiquetas HTML en función de sus atributos y/o valores de estos atributos. Usan como signo **los corchetes**. Existen cuatro formas de especificarlos:

1) [nombre_atributo], selecciona los elementos que tienen establecido el atributo llamado nombre_atributo, independientemente de su valor.

2) [nombre_atributo=valor], selecciona los elementos que tienen establecido un atributo llamado nombre_atributo con un valor igual a valor.

3) [nombre_atributo~=valor], selecciona los elementos que tienen establecido un atributo llamado nombre_atributo y al menos uno de los valores del atributo es valor.

4) [nombre_atributo|=valor], selecciona los elementos que tienen establecido un atributo llamado nombre_atributo y cuyo valor es una serie de palabras separadas con

guiones, pero que comienza con valor. Este tipo de selector sólo es útil para los atributos de tipo lang que indican el idioma del contenido del elemento.

Pero para comprenderlos mejor vemos algunos ejemplos:

`div[class] {background-color:gray}` → pone fondo gris a todos los div que tengan un atributo class, sin importar el valor de ese atributo.

`div[class="titulo"] {color:blue}` → se pondrá la letra azul en todos los div que tengan clase igual a titulo.

`a[href="www.google.es"] {color:green}` → todos los enlaces que apunten a la página de Google aparecerán en verde.

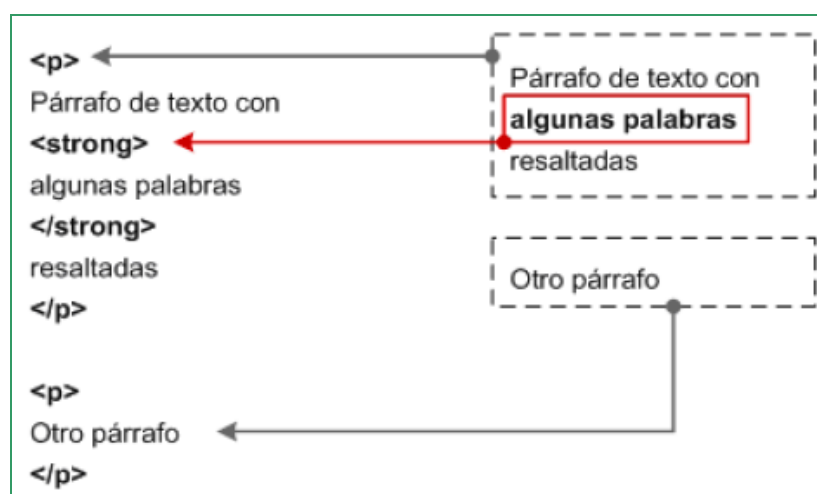
`p[class~="pie"] {text-align:right}` → alinea a la derecha todos los párrafos que en su atributo class tienen al menos definida la clase "pie".

`p[lang="fr"] {font-size:9px}` → fija la fuente a 9 píxeles para aquellos párrafos cuyo atributo lang empiece por "fr".

7.- El modelo de cajas.

El modelo de cajas la característica de CSS que provocó un cambio más brusco en el diseño de páginas web. Antes de trabajar con CSS era necesario utilizar multitud de tablas en HTML para colocar los distintos elementos en una página web. Esto hacía que las páginas tuvieran un mantenimiento problemático, pues al cambiar de sitio alguna parte del código toda nuestra estructura podía verse comprometida.

Cada vez que nosotros insertamos una etiqueta HTML se crea de forma automática una caja imaginaria que la encierra. En el caso de los elementos HTML **en bloque** esta caja ocupa todo el ancho de la pantalla. Si el elemento es una etiqueta **en línea** esta caja sólo ocupa el contenido de la etiqueta. Observa:

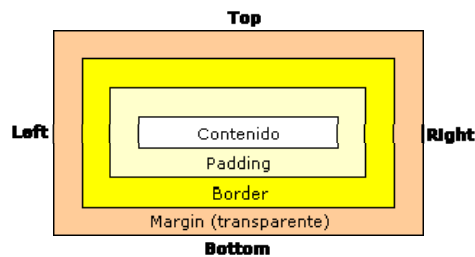




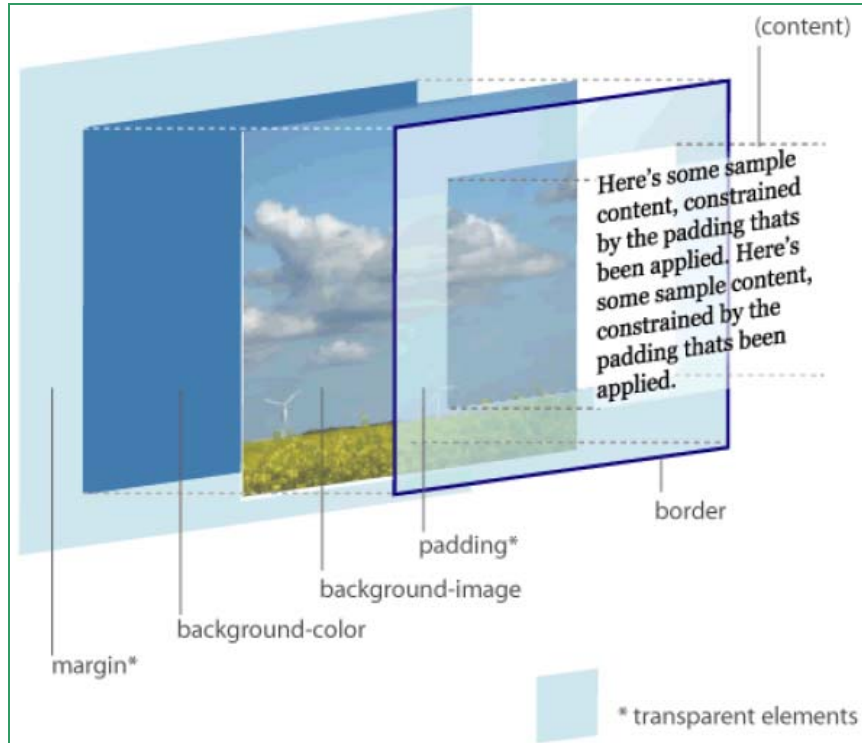
La etiqueta <p> es un elemento en bloque. Su caja ocupa todo el ancho del elemento, por ese motivo detrás de un párrafo siempre parece existir un salto de línea. La etiqueta es un elemento en línea. Su caja sólo rodea el texto en negrita.

La gran novedad es que ahora las páginas xhtml estarán formadas por cajas que nosotros podremos manipular desde CSS, indicando dónde queremos que estén, cuánto ocuparán, que bordes y rellenos tendrán, que fondo o imagen de fondo, cuáles están visibles u ocultas, la posición exacta en la pantalla, etc. Esto ha provocado la desaparición de muchas tablas anidadas y ahora muchos diseñadores web sólo trabajan con etiquetas <div>, asignando a cada una de ellas las propiedades que más le interesan.

Cada caja tiene un área de contenido (por ejemplo, texto, una imagen, etc.) y las áreas circundantes opcionales de padding (relleno), border (borde) y margin (margen).



Visto en tres dimensiones (imagen creada por <http://www.hicksdesign.co.uk/boxmodel/>), sería algo como esto:



Existe un margen (transparente), sobre él se coloca el color de fondo, sobre él una imagen de fondo, luego está el borde, luego el relleno (padding) y por último el contenido, que en la imagen superior es un texto.

Veamos un ejemplo de creación de cajas:

```
<style type="text/css">
.caja1 {border:dotted 1px black;
        margin:5px; position:static;
        width:200px; height:200px}

.caja2 {border:dotted 1px black;
        margin:5px; position:static;
        width:100px; height:100px}
</style>
```

Xhtml:

```
<div class="caja1">
  Esta es una caja de 200 píxeles
  de ancho por 200 píxeles de alto
</div>
```

```
<div class="caja2">
  Esta es una caja de 100 píxeles
  de ancho por 100 píxeles de alto
</div>
```

Resultado en el navegador:

A continuación veremos propiedades para definir las cajas.

Esta es una caja de 200 píxeles
de ancho por 200 píxeles de alto

Esta es una caja
de 100 píxeles
de ancho por
100 píxeles de
alto

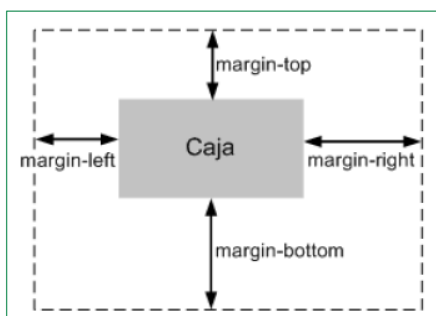
7.1.- Alto y ancho:

Son las propiedades **height** (alto) y **width** (ancho).

- Pueden tener valores absolutos: 3px, 3in (inches = pulgadas), 3cm (centímetros), ...
- Pueden tener valores relativos: 30% (éstos se aplicarán siempre en relación al elemento padre).
- También pueden tener el valor auto (es el valor por defecto) o inherit que significa que se hereda el valor de otro elemento padre.

7.2.- El margen:

El margen es un espacio transparente que separa a la caja de otros elementos que estén alrededor. Usa cuatro propiedades, cada una para un lado, que son:



Puede contener medidas absolutas o relativas. Una cuestión a tener en cuenta es que los márgenes verticales (margin-top y margin-bottom) sólo se pueden aplicar a elementos en bloque y a las imágenes. Los márgenes laterales se pueden aplicar a cualquier elemento. Ejemplo de uso:

```
.miclase { margin-top:10px;
          margin-right: 20px;
          margin-bottom: 10px;
          margin-left: 20px;
          }
```



Propiedad shorthand: así es como se llama a una propiedad CSS que sirve para escribir de forma abreviada un conjunto de propiedades relacionadas. La propiedad shorthand para los márgenes es **margin**. Mira ejemplos de uso:

```
.miclase { margin: 10px, 20px, 10px, 20px }
```

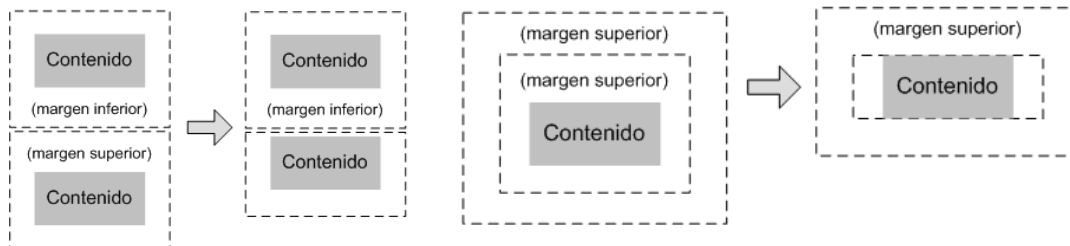


```
.miclase { margin: 10px }
```



- Si solo se indica un valor, todos los márgenes tienen ese valor.
- Si se indican dos valores, el primero se asigna al margen superior e inferior y el segundo se asigna a los márgenes izquierdo y derecho.
- Si se indican tres valores, el primero se asigna al margen superior, el tercero se asigna al margen inferior y el segundo valor se asigna los márgenes izquierdo y derecho.
- Si se indican los cuatro valores, el orden de asignación es: margen superior, margen derecho, margen inferior y margen izquierdo.

Nota: los márgenes verticales dan muchos quebraderos de cabeza a los diseñadores web. Cuando dos elementos con márgenes verticales se encuentran juntos, éstos se fusionan, quedando sólo un margen resultante que es el mayor de los dos. También ocurre con elementos que están dentro de otros. Mira ejemplos de fusión de márgenes:



PRÁCTICA 13.- Utiliza los iconos de los diferentes navegadores para mostrar una página web con ellos. A cada uno asígnale un margen superior e inferior de 150 px. Los márgenes laterales serán de 200 píxeles. Agrega un borde para ver los efectos de los márgenes:



G. Chrome



I. Explorer



Firefox



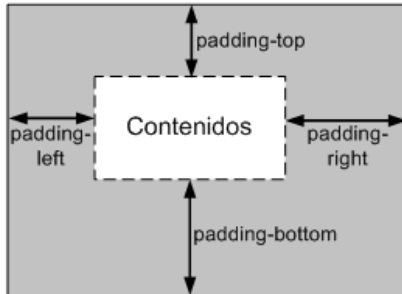
Opera



Safari

7.3.- El relleno:

El relleno es un espacio, también transparente, que separa el borde del contenido de una caja. Recuerda: el margen está fuera del borde. El relleno está dentro del borde. Su funcionamiento es muy similar al de los márgenes:



Puede contener medidas absolutas o relativas. También puede tener el valor inherit (heredado).

Ejemplo de uso:

```
.miclase { padding-top:10px;
padding-right: 20px;
padding-bottom: 10px;
padding-left: 20px;
}
```

Propiedad shorthand: para el relleno es la propiedad **padding**. Y funciona igual que margin, podemos indicar 4 valores, 3, 2 o 1. Mira ejemplos de uso:

```
.miclase { padding: 10px, 20px, 10px, 20px}
```

```
.miclase {padding: 10px}
```



PRÁCTICA 14.- Utilizando de nuevo los iconos de navegadores aplica a las imágenes un relleno de 200 píxeles en todos los lados. Aplica borde para que puedas observar cómo se alejan del borde.



PRÁCTICA 15.- En la práctica anterior deja la propiedad de relleno que escribiste y agrega otra nueva para que el relleno izquierdo sea cero píxeles. ¿Cómo afecta a la visualización?

7.4.- El borde:

El borde separa el contenido (y su relleno) del margen. Si una caja no tiene margen el borde será su límite exterior. El borde no es transparente, lo que quiere decir que podremos aplicarle colores y formas (sólido, discontinuo, etc). Luego el conjunto de propiedades CSS para el manejo del borde se amplía puesto que para cada uno de los cuatro bordes podremos definir: anchura, color y estilo.

Propiedades para la anchura:

border-top-width
border-right-width
border-bottom-width
border-left-width

Valores admitidos:

Una medida o thin (fino) o medium (medio) o thick (grueso) o inherit.

Propiedad shorthand:

border-width
(puede llevar 4, 3, 2 o 1 elementos)

Propiedades para el color:

border-top-color
border-right-color
border-bottom-color
border-left-color

Valores admitidos:

Un color o transparent (transparente) o inherit (heredado).

Propiedad shorthand:

border-color
(puede llevar 4, 3, 2 o 1 elementos)

**Propiedades para el estilo:**

border-top-style
border-right-style
border-bottom-style
border-left-style

Valores admitidos:

none | hidden | dotted | dashed | solid |
double | groove | ridge | inset | outset |
inherit (la | significa ó)

Propiedad shorthand:

border-style
(puede llevar 4, 3, 2 o 1 elementos)

**Propiedades shorthand globales:**

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

Valores admitidos:

Primero el ancho después el color y
después el estilo, separados por
espacios en blanco.

Ejemplos:

border-top {10px red solid}
border-left {green}

Ejemplos de visualización de los estilos:

None

Hidden

Dotted

Dashed

Solid

Double

Groove

Ridge

Inset

Outset

**Propiedad shorthand absoluta:**

border

Valores admitidos:

Hasta cuatro configuraciones de bordes

Ejemplos:

border {10px red solid} → todos
border {green, red} → verdes los
verticales y rojos los horizontales.

**PRÁCTICA 16.-** Partiremos del siguiente texto del Quijote:

<p>La del alba sería cuando Don Quijote salió de la venta, tan contento, tan gallardo, tan alborozado por verse ya armado caballero, que el gozo le reventaba por las cinchas del caballo. </p>

<p> Mas viniéndole a la memoria los consejos de su huésped acerca de las prevenciones tan necesarias que había de llevar consigo, en especial la de los dineros y camisas, determinó volver a su casa y acomodarse de todo, y de un escudero, haciendo cuenta de recibir a un labrador vecino suyo, que era pobre y con hijos, pero muy a propósito para el oficio escuderial de la caballería. </p>

<p> Con este pensamiento guió a Rocinante hacia su aldea, el cual casi conociendo la querencia, con tanta gana comenzó a caminar, que parecía que no ponía los pies en el suelo. </p>

<p>No había andado mucho, cuando le pareció que a su "diestra mano", de la espesura de un bosque que allí estaba, salían unas voces delicadas, como de "persona" que se quejaba; y apenas las hubo oído, cuando dijo:</p>

<p> gracias doy al cielo por la merced que me hace, pues tan presto me pone ocasiones delante, donde yo pueda cumplir con lo que debo a mi profesión, y donde pueda coger el fruto de mis buenos deseos.</p>

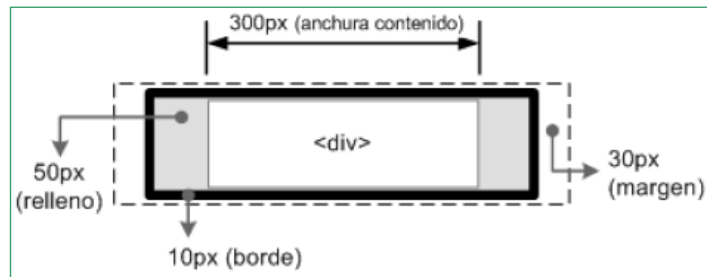
<p> estas voces sin duda son de algún menesteroso o menesterosa, que ha menester mi favor y ayuda y volviendo las riendas encaminó a Rocinante hacia donde le pareció que las voces salían</p>

Crea clases de estilos para hacer lo siguiente:

- 1) el primer párrafo tiene márgenes laterales de 100 píxeles.
- 2) El segundo párrafo tiene bordes de 3 píxeles de color verde y no tiene relleno.
- 3) El tercer párrafo tiene bordes de 2 píxeles discontinuos de color azul los verticales y amarillo los horizontales.
- 4) El cuarto sólo tiene borde inferior, es negro de 5 píxeles y de estilo groove.
- 5) El quinto párrafo tiene sólo 200 píxeles de ancho con un relleno de 50 píxeles y un borde fucsia sólido de 2 píxeles.

7.5.- Nota sobre el tamaño:

Una cuestión importante que debemos saber es que width y height se refieren al contenido. Si además tenemos márgenes, rellenos y bordes el tamaño de estos se sumarán al de width y height.



Esto hay que tenerlo en cuenta al diseñar páginas con el modelo de cajas, dado que si no lo hacemos los elementos se nos solaparán.

En su momento los fabricantes de navegadores no tuvieron en cuenta este aspecto e introdujeron mejoras en sus navegadores para adaptar los contenidos. Esto provocó que las páginas no se visualizaran siempre de forma correcta. ¿Qué hicieron los fabricantes? Pues idearon dos “modos de funcionamiento” del navegador:

- El modo compatible con las páginas antiguas, llamado "**modo quirks**".
- El modo compatible con los nuevos estándares, llamado "**modo estándar**".

Estos modos se activan de forma automática al leer la línea <!DOCTYPE> de la página. Si no tenemos etiqueta doctype o la tenemos pero es anterior a HTML 4.0 inmediatamente se activa el modo quirks.



¿Cuál es el problema con el modo quirks? Pues que la visualización del modelo de cajas cambia drásticamente y no se ajusta al estándar que estamos viendo en este tema. En concreto existe otro problema en Internet Explorer: se puede activar el modo quirks en xhtml 1.0 si se incluye la declaración de xml para el sistema utf-8:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

Recuerda: esto afecta a Internet Explorer provocando que entre en modo quirks (donde el ancho incluye los márgenes y el relleno).

7.6.- El fondo:

El fondo está dentro del borde. Podemos asignarle un color o una imagen. Afecta al contenido y al relleno (recuerda que el relleno es transparente, luego si colocamos fondo éste ocupará el área del relleno). Si quieres que el fondo afecte a toda la página no tienes más que aplicarlo al selector body. Veamos las propiedades:



Propiedad:	Valores posibles:
background-color	Un color ó transparent ó inherit (heredado)
background-image	Una url ó none ó inherit (heredado)
background-repeat	repeat ó repeat-x ó repeat-y ó no-repeat o inherit
Background-position	Dos medidas ó una medida ó dos palabras reservadas como top, middle, bottom, left, center o right
Background-attachment	Scroll ó fixed ó inherit

Propiedad shorthand:

background

el contenido indicará el color, la imagen, la repetición, la posición y si lo hay el attachment. Ver ejemplo al final de la explicación sobre fondos.

Para definir un color de fondo usamos **background-color**. Si hay imagen ésta se coloca sobre el color de fondo y usaremos para ello la propiedad **background-image**. Podemos definir las dos propiedades juntas y el color de fondo se verá en los lugares en que la imagen es transparente. Si la imagen es muy pequeña para el contenido de la caja ésta se repetirá horizontal y verticalmente. Si la imagen es muy grande sólo se mostrará la porción que quepa en el tamaño de la caja.

Ejemplo:

```
.cabecera {background-color: gray;
          background-image: url(/mifondo.jpg)
          }
```

La visualización o repetición de la imagen se puede controlar con **background-repeat**. Si el valor de la propiedad es repeat se aplica el comportamiento por defecto (se

repite en todas direcciones). Si es repeat-x sólo se repetirá horizontalmente. Repeat-y es para repetir verticalmente. No-repeat coloca la imagen y no la repite.

No siempre una imagen de fondo ocupará todo el área de una caja. En ocasiones será un pequeño icono que agregamos a la caja, y en ese caso podemos desear colocar la imagen en una zona concreta. Para esto se utiliza la propiedad **background-position**, y mediante ella indicaremos la distancia que existe desde el origen de coordenadas, que es la esquina superior izquierda de la caja.

Si background-position contiene dos medidas (ya sean medidas absolutas o porcentajes) el primero será cuánto desplazamos la imagen en horizontal y el segundo en vertical.

Si contiene sólo una medida será el desplazamiento horizontal y al vertical se le asignará un 50%.

Pero background-position puede contener también algunas palabras reservadas. Para horizontal será left (0%), center (50%) o right (100%). Para distancias verticales usamos top (0%), middle (50%) y bottom (100%).

Ejemplos:

```
.cabecera {background-image: url(/miicono.gif);
background-repeat: no-repeat;
background-position: left top; → coloca el icono en la esquina superior izquierda
background-position: 20 px bottom; → lo coloca abajo y 20 píxeles a la derecha
background-position: center middle; → la imagen está en el centro de la caja.
}
```

La última de las propiedades del fondo es **background-attachment**. Sus valores pueden ser Scroll (que es el valor por defecto si no se usa la propiedad) o fixed (fijo) o inherit (heredado del elemento padre).

Esta propiedad sirve para hacer que el fondo permanezca en la misma posición aunque se usen las barras de desplazamiento lateral del navegador, con lo que una caja puede quedarse fija. Para hacer esto debemos usar el valor **fixed**. Ten en cuenta que esto resulta muy molesto para los usuarios de la web y además no lo soportan todos los navegadores.

Como siempre podemos abreviar usando la propiedad shorthand **background**. Hay que indicar las propiedades que queremos usar y aunque el orden es indiferente es mejor que nos ajustemos a color, imagen, repetición y después posición. Los valores los separamos por un espacio en blanco. Mira un ejemplo:

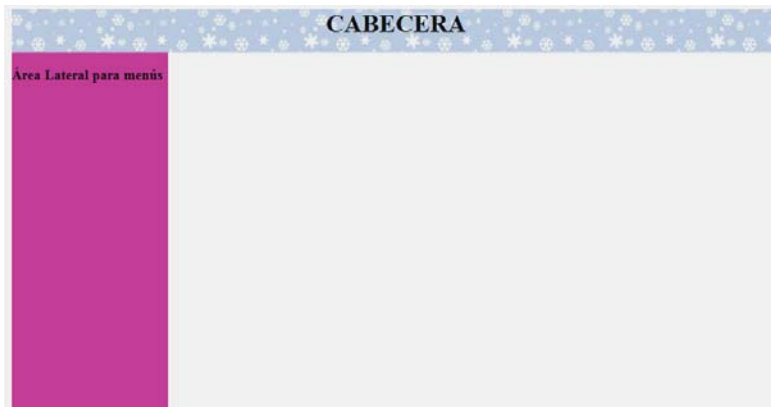
```
.rotulo {background: red url(/fondo.jpg) repeat-x left top}
Equivale a:
.rotulo {background-color: red;
background-image: url(/fondo.jpg);
background-repeat: repeat-x;
background-position: left top
}
```



PRÁCTICA 17.- Busca en Internet un pequeño icono con una textura y crea una página web con ese icono de fondo repetido en todas direcciones. Ejemplo de textura:



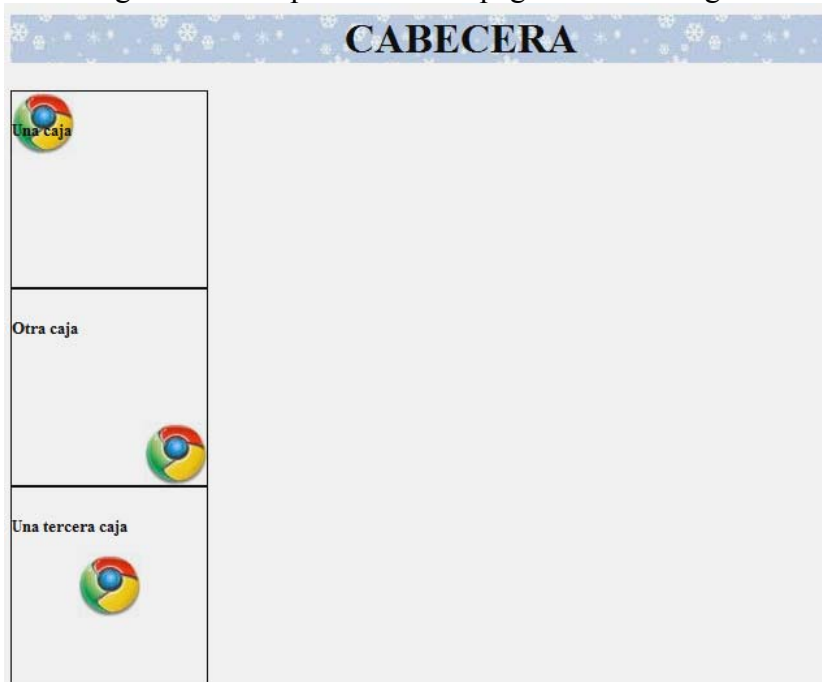
PRÁCTICA 18.- Usando dos div crea una página como la siguiente:



El primer div repite la imagen de textura horizontalmente y ocupa un 100% de la pantalla. El segundo div tiene un ancho de 200px y un alto de 500px. Usa un tono fucsia como color de fondo.



PRÁCTICA 19.- Seguimos con el modelo de cajas, utiliza el icono de chrome como imagen de fondo para crear una página como la siguiente:



Las tres cajas nuevas tienen 150*150 píxeles. La imagen de fondo se ajusta con background-position.

8.- La potencia de CSS: el posicionamiento.

Ya hemos comprendido el modelo de cajas. Ahora podemos usar muchos <div> y ajustarles las propiedades de márgenes, bordes, relleno y fondo. Sólo nos falta una cosa: colocarlas donde queramos. Una vez que sepamos posicionar los div en las pantallas ya no necesitaremos construir nuestras web con tablas. Para hacer esto es necesario recurrir al sistema de **posicionamiento de CSS**.

El posicionamiento en CSS depende de muchos factores: tamaño de la ventana del navegador, propiedades width y height, árbol jerárquico de los elementos (un elemento dentro de otro es hijo), etc. Pero un factor determinante es si el elemento es un elemento **en bloque o en línea**. Si no uso div sino otras etiquetas tendré que tener en cuenta esta característica pues las cajas se pueden crear automáticamente:

Los párrafos son elementos de bloque.
Los enlaces son elementos en línea
Dentro de un párrafo, los enlaces siguen siendo elementos en línea.

Recuerda que en los elementos en bloque su caja llega hasta el final de la ventana. El siguiente elemento empezará en una nueva línea.

Elementos en bloque:

- No pueden insertarse dentro de elementos en línea.
- Pueden insertarse dentro de otros elementos en bloque.

Elementos en línea:

- Pueden insertarse dentro de elementos en línea y dentro de elementos en bloque.
- Generalmente no se insertan dentro de otro elemento en línea del mismo tipo. Ejemplo: no colocamos un enlace <a> dentro de otro enlace <a>.

Para indicar la posición de una caja usaremos la propiedad CSS **position** que puede contener un valor de entre cinco. Es decir, en CSS, tenemos cinco formas de posicionar una caja. En combinación con position usaremos las propiedades top, bottom, left y right que fijarán el **desplazamiento**.

Propiedad para la posición: position

Valores admitidos:

static → es posicionamiento normal
Relative → posicionamiento relativo
Absolute → posicionamiento absoluto
fixed → posicionamiento fijo
inherit → posicionamiento heredado

Propiedades para el desplazamiento: top, bottom, left, right

Valores admitidos:

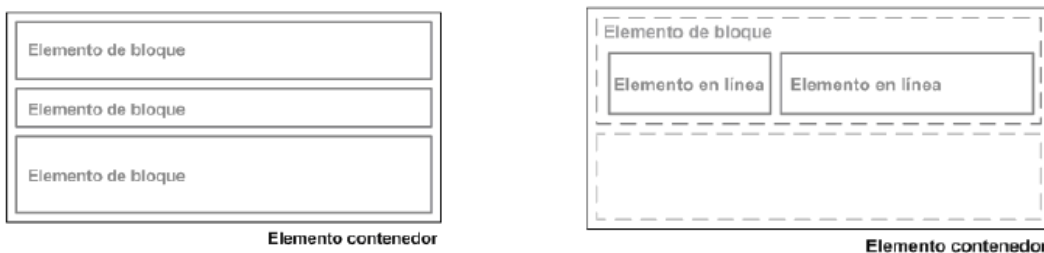
Una medida ó un porcentaje ó auto ó inherit



8.1.- Posicionamiento normal:

Utiliza **position:static**. Es el posicionamiento de las cajas por defecto. Lo único que mira el navegador es dónde le corresponde colocar una caja según el flujo de las etiquetas anteriores o de la etiqueta padre (en caso de haberla). Se ignoran los valores de top, right, bottom y left puesto que no hay desplazamiento. Es el que hemos usado en las prácticas anteriores sin saberlo.

Las cajas se colocan una detrás de otra empezando desde la izquierda. Lo único importante es si el elemento es de bloque (empezará en una nueva línea) o el elemento es de línea (se pondrá al lado del elemento anterior). También influye si hay un elemento padre (**llamado contenedor**). Observa ejemplos:

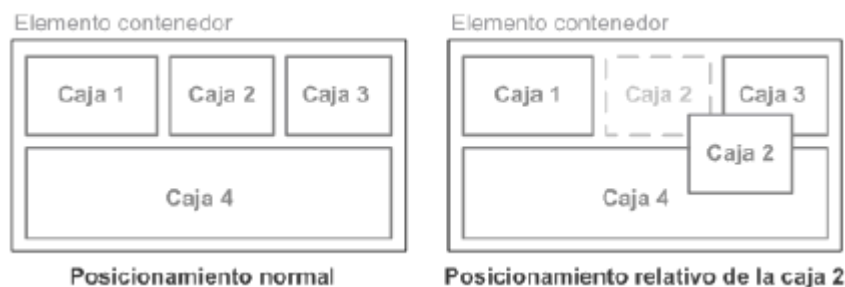


El contenedor o padre determinará el tamaño y la posición de los elementos. Recuerda que si una etiqueta no está dentro de otra su contenedor es el elemento `<body>`.

Aquí podríamos usar propiedades como text-align para establecer alineaciones.

8.2.- Posicionamiento relativo:

Utiliza **position:relative**. Primero se calcula la posición de la caja usando el mismo método que en el posicionamiento normal. Pero en lugar de mostrar la caja en ese sitio calculado se aplica un desplazamiento, que vendrá definido por las cuatro propiedades de desplazamiento (top, left, bottom, right). Observa la diferencia:



En la segunda imagen se muestra como la caja 2 se ha desplazado.

Nosotros desplazamos la caja desde su esquina superior izquierda. Left, top, bottom y right nos permiten desplazar desde ese punto inicial hacia la izquierda, arriba, abajo y derecha. Estas cuatro propiedades pueden tener valores positivos y negativos.

Ejemplo:

```
.caja2 {position: relative;
        right: 50px;
        bottom: 15px;
        }
```

En este ejemplo movemos caja2 desde su posición normal un total de 50 píxeles a la derecha y 15 píxeles hacia abajo. Las propiedades left y top que no hemos usado podríamos haberlas escrito como left:auto; top:auto.

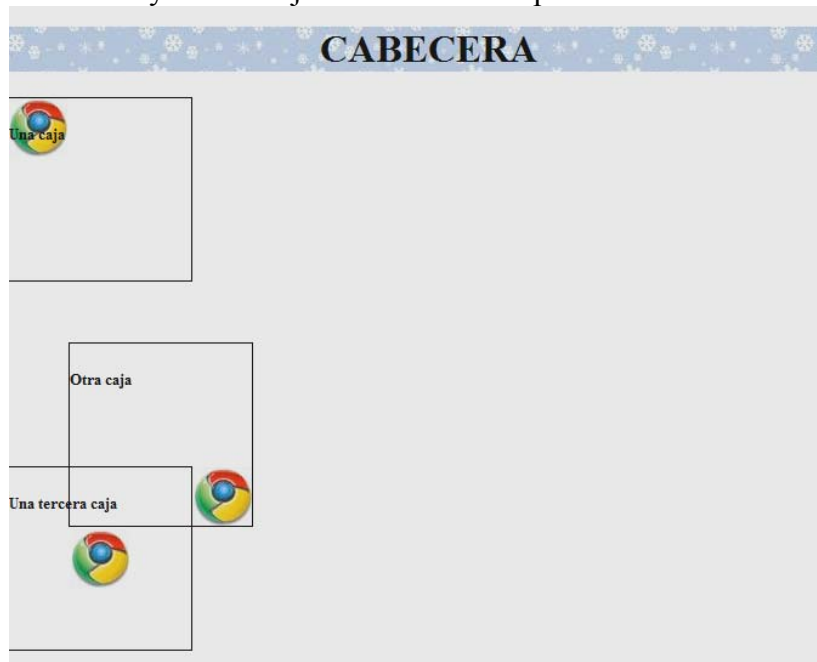
Lógicamente si usamos left porque movemos una caja a la izquierda entonces no usaremos right (porque esto es para moverla a la derecha). También puedo decir left:-50px y lo que estamos diciendo es que movemos a la derecha 50 píxeles.

Existe una propiedad CSS relacionada llamada **direction** que marca la dirección de escritura de un texto. Sus valores pueden ser: ltr (left to right) para textos escritos de izquierda a derecha (como en castellano) o rtl (right to left) para textos que se escriben de derecha a izquierda (como en árabe).



PRÁCTICA 20.- Comprueba que utilizaste <div> para crear las tres cajas con el logo de chrome en la práctica 19.

- 1) ¿Por qué aparecen una debajo de la otra?
- 2) ¿Qué tipo de posicionamiento utilizan?
- 3) Modifica el código CSS de la segunda caja para desplazarla 50 px a la derecha y hacia abajo en relación a su posición normal. Mira el resultado:





8.3.- Posicionamiento absoluto:

Utiliza **position: absolute**. También utiliza desplazamiento con las cuatro propiedades vistas (top, left, right y bottom), pero en este caso se aplican en relación al elemento padre.

Ocurre que si el elemento padre es <body> podremos indicar la posición exacta donde queramos el elemento. Ejemplo:

<pre>.miclase { Position: absolute; left: 237px; top: 20%; }</pre>	Definimos .miclase y decimos el elemento se mueva 237 píxeles desde las coordenadas 0,0 del elemento padre. Y además se mueva desde arriba (top) un 20% hacia abajo.
<pre>... código xhtml ... <body> <div class="miclase"> Hola </div> </body></pre>	El texto Hola aparece 237 píxeles a la derecha y un 20% hacia abajo.

IMPORTANTE: Hay que tener en cuenta que si el elemento al que se aplica está dentro de otro elemento el desplazamiento es en relación a la esquina superior izquierda del **elemento contenedor más cercano que no esté posicionado con static**. Con frecuencia esto hace que los elementos que usan posicionamiento absoluto aparezcan como fuera del flujo del documento.



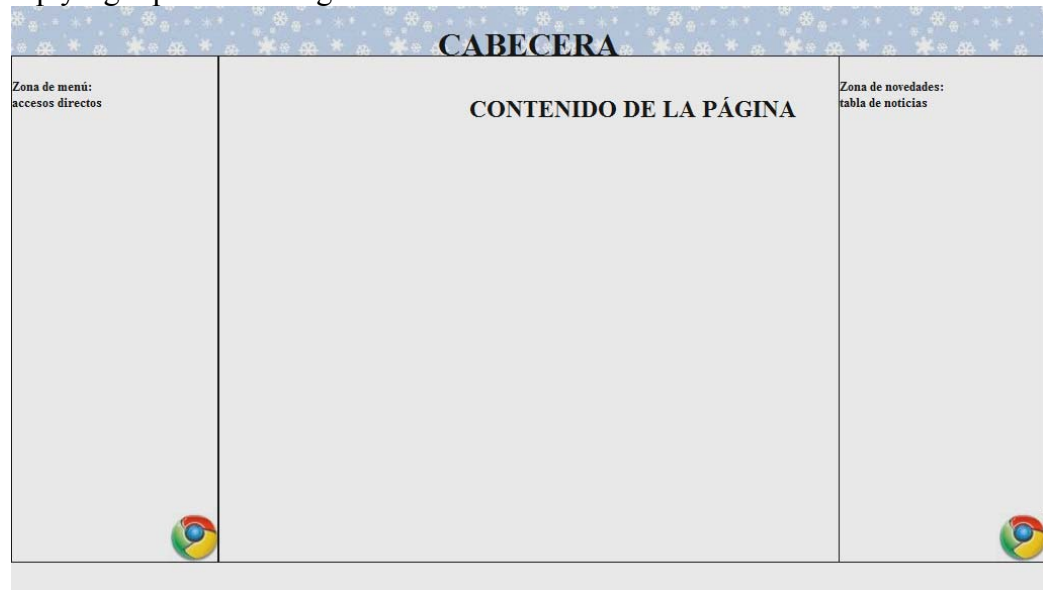
PRÁCTICA 21.- modifica la página anterior para que sea como esta:



Pista, la separación entre cajas es de 50 píxeles y cada caja está 50px más abajo que la anterior. La primera está separada del borde de la página 100px.



PRÁCTICA 22.- Esta práctica te puede dar algún quebradero de cabeza. Tendrás que combinar el posicionamiento con width y height, además de left, top y right para hacer algo como esto:



El siguiente párrafo de estos apuntes te puede ayudar. La haremos en clase.

Determinar el origen de coordenadas a partir del cual se desplaza una caja posicionada de forma absoluta es un proceso complejo que se compone de los siguientes pasos:

- Se buscan todos los elementos contenedores de la caja hasta llegar al elemento <body> de la página.
- Se recorren todos los elementos contenedores empezando por el más cercano a la caja y llegando hasta el <body>.
- De todos ellos, el navegador se queda con el primer elemento contenedor que esté posicionado de cualquier forma diferente a position: static.
- La esquina superior izquierda de ese elemento contenedor posicionado es el origen de coordenadas.

Una vez obtenido el origen de coordenadas, se interpretan los valores de las propiedades top, right, bottom y left respecto a ese origen y se desplaza la caja hasta su nueva posición.



Nota importante:

- Estás usando un div y dentro colocas elementos (imágenes, párrafos, etc).
- Después a uno de estos elementos decides aplicarle un posicionamiento absoluto.
- **¡ PUEDE OCURRIR QUE SE NOS SALGA DEL DIV !**
- **¿cómo evitarlo?**
- Cambia el posicionamiento del div para que no sea static. Ponlo relative por ejemplo.



8.4.- Posicionamiento fijo:

Utiliza **position:fixed**. Es un tipo especial de posicionamiento absoluto. Se usa muy poco (no está soportado en versiones de IExplorer anteriores y a la 6) y además puede resultar muy molesto para los usuarios. Pero tiene alguna ventaja.

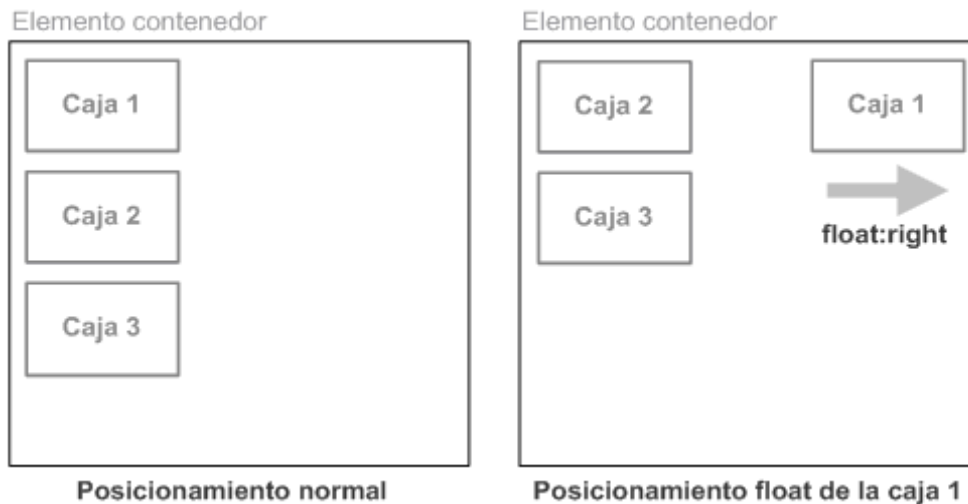
Una caja posicionada como fija se coloca exactamente con las mismas reglas que en el posicionamiento absoluto. Sin embargo si movemos la ventana del navegador usando las barras de scroll laterales la caja **no se moverá**. Se queda fija en su sitio.

En elementos paginados como las impresoras estas cajas se repetirán en todas las páginas. A veces esto se usa para crear cabeceras y pies en estos medios. Pero eso es para hojas de estilos creadas con el atributo `media="print"`. Sólo hemos citado los medios al comienzo de este tema, no los hemos explicado.

8.5.- Posicionamiento flotante:

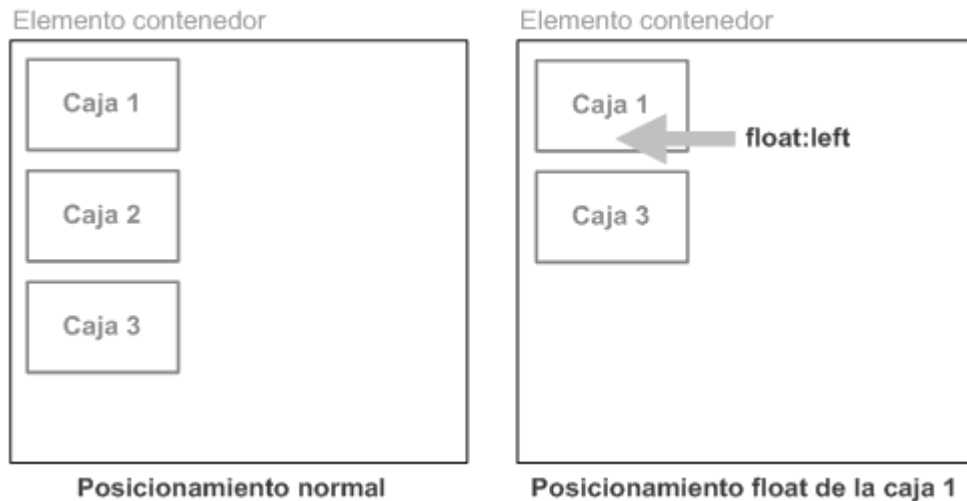
No utiliza position. Es el tipo de posicionamiento más avanzado que existe en CSS. Hoy en día se utiliza mucho para crear páginas profesionales. Es un poco difícil de comprender.

Cuando una caja se posiciona con el modelo de posicionamiento flotante, automáticamente se convierte en una caja flotante, lo que significa que se desplaza hasta la zona más a la izquierda o más a la derecha de la posición en la que originalmente se encontraba. Pero eso no es todo, esa caja se sale del flujo normal de la página, queda **flotando**. Observa un ejemplo:



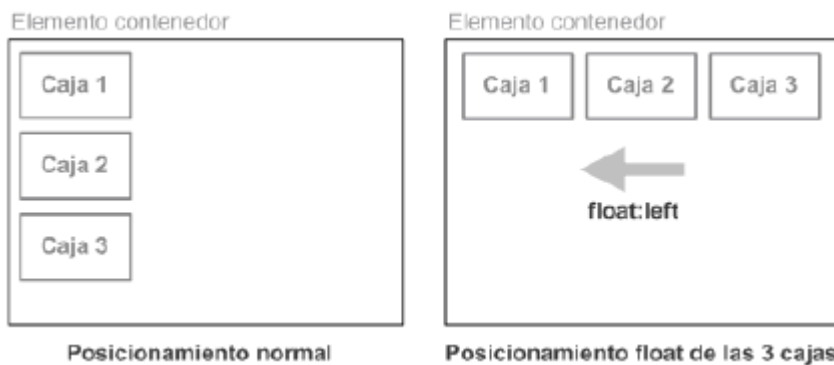
Como ves en el ejemplo las cajas 2 y 3 se desplazan hacia arriba porque la 1 ha salido del flujo de la página, luego las otras ocupan su lugar.

Si en el mismo ejemplo la caja 1 la posicionamos como flotante pero a la izquierda ocurrirá esto:

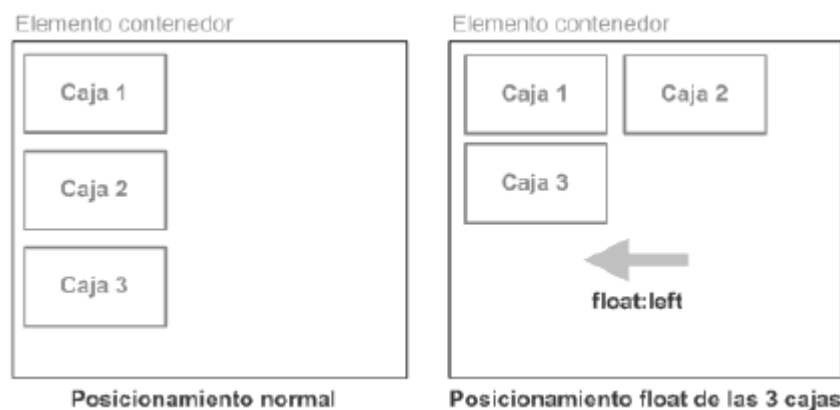


Se desplaza la caja 1 a la izquierda y tapa la caja 2, que queda debajo. Los objetos flotantes están flotando sobre la página.

Ahora imagina que aplicamos flotante a la izquierda a las tres cajas, el resultado es que la caja 2 detecta otro objeto flotante y se coloca “a continuación”. Mira cómo queda:



Esto nos permite colocar objetos que pueden ser de tipo bloque en la misma línea. Si no existiera sitio para caja 3 en la misma línea se colocaría debajo.



En este ejemplo no había sitio suficiente para la caja 3 en la misma línea.

Pero hay más: las cajas flotantes influyen en la disposición de todas las demás cajas. Los elementos en línea hacen sitio a las cajas flotantes adaptando su anchura al espacio libre dejado por la caja desplazada. Los elementos de bloque no les hacen sitio, pero sí que adaptan sus contenidos para que no se solapen con las cajas flotantes.



¿Cómo declarar una caja con posicionamiento flotante?

PROPIEDAD:

float

VALORES ADMITIDOS:

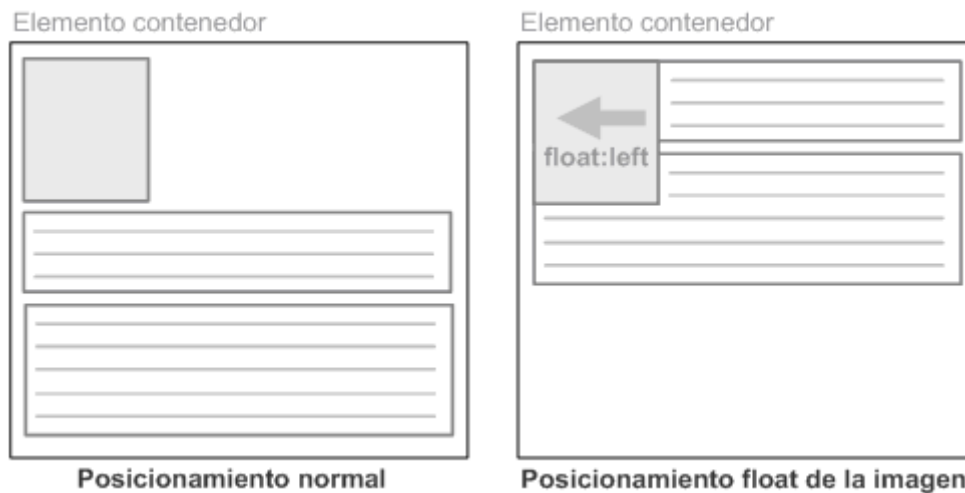
left ó right ó none ó inherit



Left desplaza la caja a la zona más a la izquierda de la línea actual. Right lo hace a la derecha. None quita el posicionamiento flotante. Inherit significa que este elemento hereda el posicionamiento flotante que se haya declarado

en el padre.

Con esta técnica los elementos que están cerca de la caja flotante adaptan su posición para fluir alrededor de ella. Esto se creó porque a menudo en los diseños de páginas web interesa colocar imágenes y que el texto fluya alrededor, cosa que con HTML no puede hacerse. Mira el ejemplo:



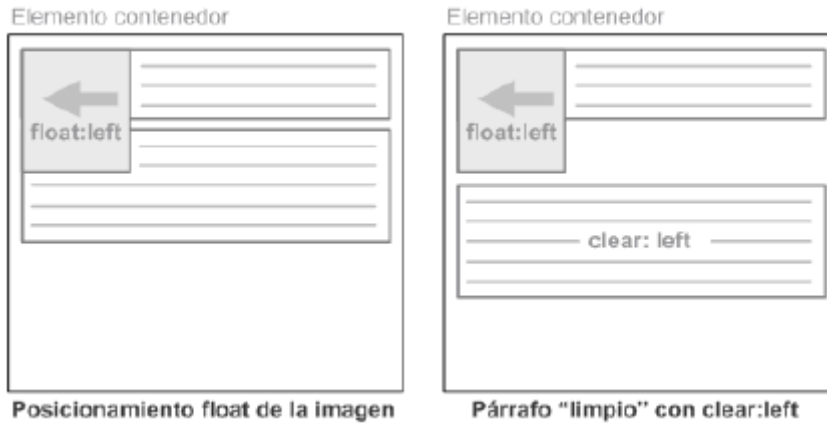
Los párrafos se adaptan para rodear a la imagen, quedando un aspecto más profesional. Muy utilizado para presentar noticias, productos, etc.

Sólo tendríamos que hacer:
`img { float: left }`

Pero puede interesarnos que sólo el primer párrafo se ajuste a la imagen, mientras que el segundo no lo haga. En ese caso al segundo párrafo le aplicaremos la propiedad CSS **clear**, indicando que queremos borrar el posicionamiento flotante a la izquierda:

`.parrafo2 { clear: left }`

El resultado sería el siguiente:



PROPIEDAD:

clear

VALORES ADMITIDOS:

left ó right ó both o none ó inherit



Both significa “ambos lados”

La propiedad clear hay que usarla más a menudo de lo crees. Veamos una práctica.



PRÁCTICA 23.-

PASO1: crea una página que tenga un div con fondo de la textura que usamos en las prácticas anteriores y ancho el 100%. Dentro del div inserta dos iconos o flechas del tipo “anterior” y “siguiente”. Como esto:



PASO2: Ahora aplica a la imagen “anterior” un float:left y a la izquierda un float:right. Obtendrás algo como esto:



Pero el div con el fondo ha desaparecido. ¿Por qué? Pues al poner las imágenes como flotantes éstas se salen del div. Y como ahora el div no tiene ningún contenido pues no se muestra. Para evitarlo dentro del div y después de las imágenes tendremos que crear otro div donde borramos con clear el posicionamiento flotante anterior. Este segundo div se colocará debajo de las imágenes y dentro del div principal. En él tendremos que borrar el posicionamiento flotante por ambos lados.

Aplica a este div interno un clear:both. El resultado será algo como esto:



Resultado: Nuestra barra con las dos imágenes colocadas y el fondo visible.



9.- Mostrar y ocultar. Desbordamiento y capas.

Este será el último apartado que veamos en este tema. Se trata de un conjunto de propiedades que muchos diseñadores utilizan con javascript y que permiten dar dinamismo a las páginas, mostrando y ocultando cajas, superponiendo unas cajas sobre otras, etcétera. La verdadera potencia de estas propiedades se observa cuando programamos eventos (uso de javascript).

* **Para mostrar y ocultar** cajas hay dos propiedades CSS:

PROPIEDAD:

display

VALORES ADMITIDOS:

None, block, inherit

PROPIEDAD:

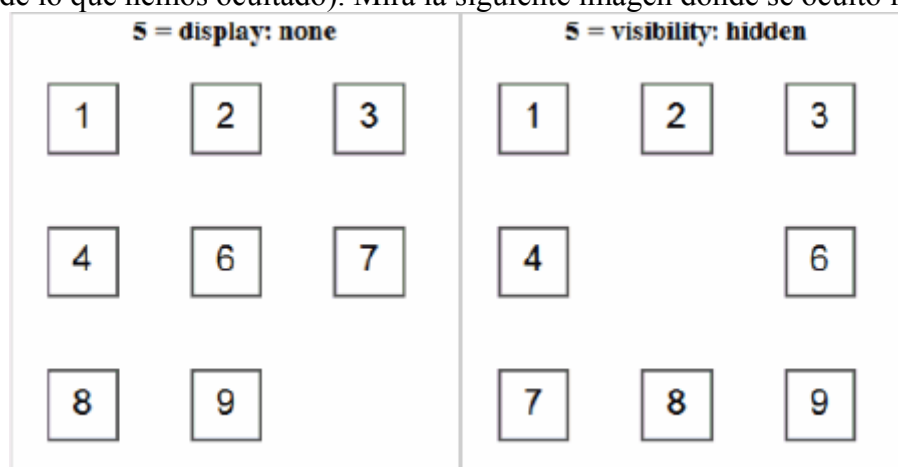
visibility

VALORES ADMITIDOS:

Visible, hidden, inherit

Si display:none la caja se oculta. Si display:block la caja vuelve a mostrarse. Lo mismo ocurre con visibility: si es visible se muestra y si es hidden se oculta.

No obstante el funcionamiento de estas dos propiedades es un poco diferente: cuando ocultamos con display el resto de cajas se trasladan para ocupar el espacio de la caja oculta. Cuando ocultamos con visibility el resto de cajas no se trasladan (queda el hueco de lo que hemos ocultado). Mira la siguiente imagen donde se ocultó la caja5.



Con display las cajas del número 7 al 9 se trasladaron.

Ahora prueba y analiza el siguiente código:

... parte de CSS ...

```
.micaja {border: solid 1px Black}
```

... parte de xhtml ...

```
<div id="caja1" style="display:block">
```

Este es el contenido de la caja 1

```
</div>
```



```

<div id="caja2" style="display:none">
  Y aquí tenemos la caja 2
</div>

<br>

<input type="button" value="mostrar la caja 2"
  onclick="caja1.style.display='none'; caja2.style.display='block'; ">
<input type="button" value="ocultar la caja 2"
  onclick="caja1.style.display='block'; caja2.style.display='none'; ">

```

IMPORTANTE SABER QUE:

Desde JavaScript podemos acceder a todas las propiedades de CSS y transformarlas dinámicamente.

* Hablemos ahora del **desbordamiento**: imagina que creas un div y utilizas las propiedades width y height para darle ancho y alto. Luego dentro del div colocas un texto o más elementos. ¿Qué pasa si el contenido necesita un espacio mayor que el tamaño del contenedor? Pues que cuando visualicemos la página veremos como el texto se sale por fuera del div. Esto se conoce como desbordamiento, en inglés, **overflow**. Podemos controlarlo desde CSS con la propiedad del mismo nombre:

PROPIEDAD:

overflow

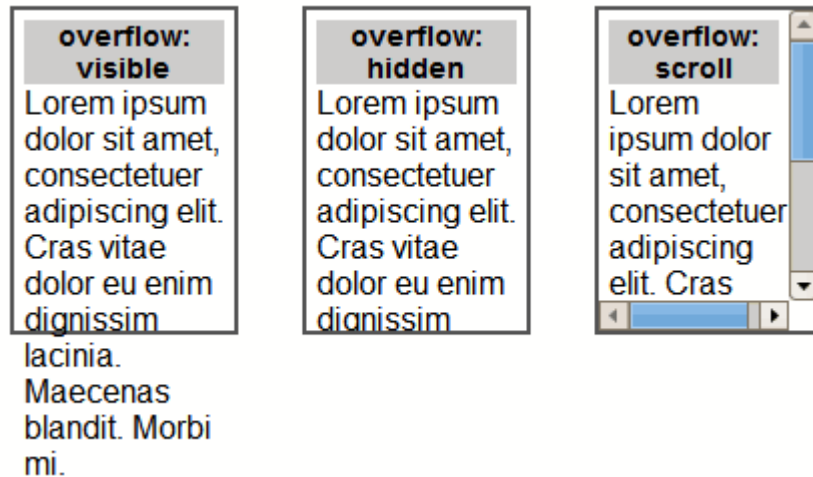
VALORES ADMITIDOS:

visible, hidden, scroll, auto, inherit



- visible: el contenido no se corta y se muestra sobresaliendo la zona reservada para visualizar el elemento. Este es el comportamiento por defecto.
- hidden: el contenido sobrante se oculta y sólo se visualiza la parte del contenido que cabe dentro de la zona reservada para el elemento.
- scroll: solamente se visualiza el contenido que cabe dentro de la zona reservada para el elemento, pero también se muestran barras de scroll que permiten visualizar el resto del contenido.
- auto: el comportamiento depende del navegador, aunque normalmente es el mismo que la propiedad scroll.

Observa las diferencias:



Y para terminar hablemos de **las capas**: hemos puesto varias cajas en nuestra página. Puede que algunas de ellas se solapen (ocupen el mismo espacio). Esto ocurre cuando usamos posicionamiento. Si dos cajas se solapan ¿cuál está por encima y cual por debajo? CSS permite usar la propiedad **z-index** para establecer un modelo de capas. A cada caja le podemos dar un valor de z-index numérico entero, el que queramos, pero eso sí, la caja que tenga un valor más alto estará siempre por encima de la caja que tenga el valor más bajo.

Modificando z-index con javascript podremos hacer páginas cuyos contenidos cambien colocándose cajas encima de otras.

Nota final: apuntes creados con contenidos y prácticas propias, recursos web gratuitos y extractos de la obra “Introducción a CSS” de Javier Eguíluz. No apto para usos comerciales, sólo fines educativos.

I.E.S. DOMINGO PÉREZ MINIK



CICLO FORMATIVO DE GRADO SUPERIOR:
*"ADMINISTRACIÓN DE SISTEMAS
INFORMÁTICOS EN RED (L.O.E.)"*



MÓDULO:

Lenguajes de marcas y sistemas de
gestión de información

(4 horas semanales)