Formación en Red

Ministerio de Educación, Cultura y Deporte.

HTLM5 en la educación

Módulo 11: Maquetación con estilos.

Instituto Nacional de Tecnologías Educativas y de Formación del Profesorado 2012

GOBIERNO DE ESPAÑA MINISTERIO DE EDUCACIÓN, CULTURA Y DEPORTE
SECRETARÍA DE ESTADO DE EDUCACIÓN, FORMACIÓN PROFESIONAL Y UNIVERSIDADES
intef INSTITUTO NACIONAL DE TECNOLOGÍAS EDUCATIVAS Y DE FORMACIÓN DEL PROFESORADO

Maquetación con estilos

Cada cosa en su sitio

La separación entre el contenido de una página web y la disposición de cada pequeña parte de ese contenido no sería posible sin la intervención de las hojas de estilo en cascada y sus múltiples propiedades. Con CSS3 contaremos con todas las posibilidades necesarias para definir una maquetación correcta que realce nuestros contenidos y nos proporcione un control absoluto del aspecto que toma cada parte.

A lo largo de este módulo nos dedicaremos a conocer aquellas propiedades que se orientan a la maquetación, a definir sus espacios y la relación entre los diferentes elementos que podemos ubicar en una página web.

Programación

Objetivos específicos

- Aplicar estilos relacionados con tamaños, bordes y márgenes.
- Aplicar estilos relacionados con el posicionamiento de los objetos.

Contenidos

- Estilos de caja.
- Estilos de bordes.
- Estilos de márgenes.
- Posicionamiento y ocultación.

Criterios de evaluación

- Aplicar estilos a diferentes elementos.
- Modificar los estilos aplicados.

Requisitos mínimos

- Conocimientos sobre HTML.
- Conocimientos sobre navegadores web.
- Conocimientos de procedimientos en el ordenador: seleccionar, cortar y pegar.

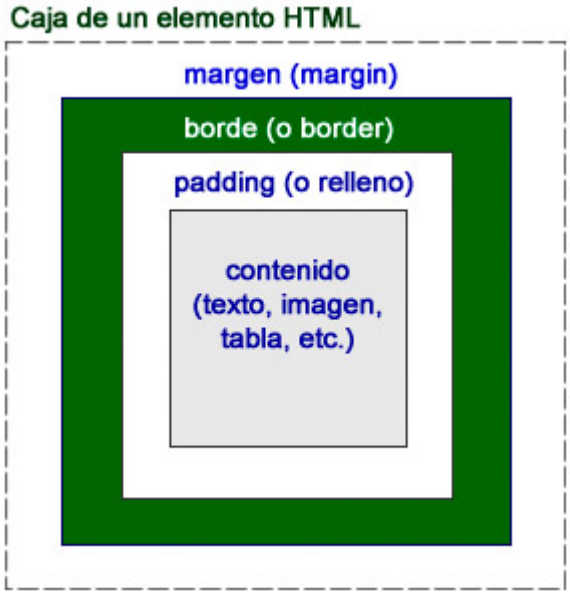
Recurso TIC: Maquetación con estilos

Seguiremos avanzando en nuestro conocimiento de las propiedades más empleadas en las hojas de estilo, centrándonos ahora en aquellas que conllevan posicionamiento de objetos, incluyendo algunas que definen su apariencia, pero más en un ámbito espacial.

El modelo de caja

Cada elemento HTML que podemos encontrar en una página web se encuentra rodeado de una caja con varias propiedades que pueden ser modificadas.

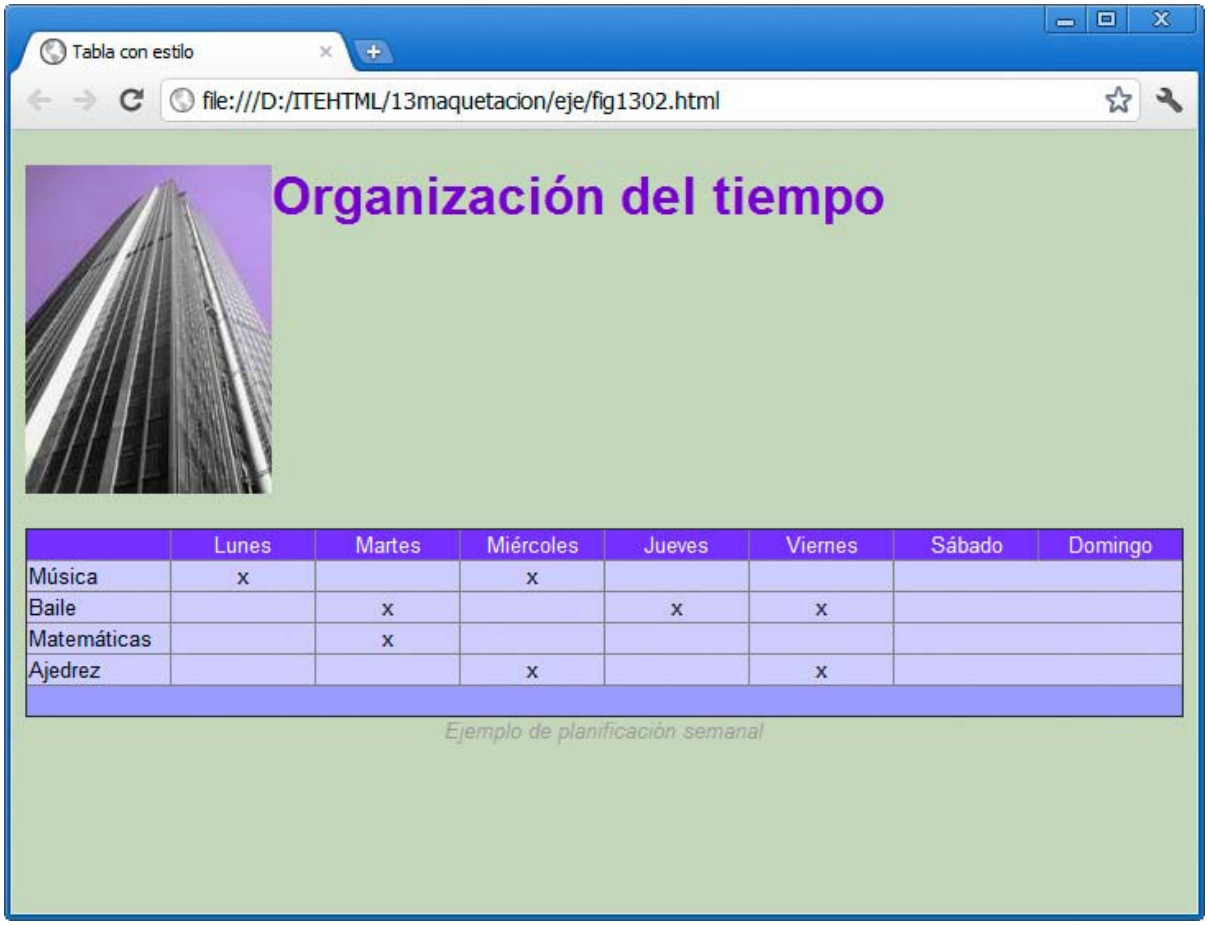
La figura muestra esas propiedades.



Además del contenido, cada elemento puede tener o no un borde y, además, podemos modificar la distancia entre el borde y el límite de la caja (**margin**) y la distancia entre ese borde y el comienzo del contenido (**padding**).

Estas opciones nos proporcionan un gran control sobre cómo debe situarse cada elemento.

Observemos el ejemplo de la figura:



Se trata de una tabla, con un título y una imagen. Cada elemento HTML tiene ya su propio estilo aplicado, todos conocidos. Éstos son los principales que, como se puede observar, hemos aplicado a los elementos genéricos directamente:

```
body {

        text-align: left;
        background-color: rgb(200, 214, 185);
        }

h1 {

        font-family: Arial, Helvetica, sans-serif;
        color: rgb(102, 0, 204);
        vertical-align: top;
        }

caption {

        font-style: italic;
        color: rgb(153, 153, 153);
        }

img {

        text-align: left;
        vertical-align: top;
        }

tbody {

        text-align: center;
        background-color: rgb(204, 204, 255);
        }
```

En la imagen se aprecia que algunos elementos están situados muy pegados entre sí. Se aprecian poco esos espacios de los que estamos hablando. Probemos a modificar alguno de sus valores.

Margen

Comenzaremos con la imagen. Con tan sólo modificar su margen, observaremos cómo se distancia del resto de los elementos. Usaremos la propiedad **margin** seguida de un valor numérico o de un porcentaje:

```
img { margin: 20px; }
```

En la figura se puede apreciar el desplazamiento de la imagen 20 píxeles por cada lado.

	Lunes	Martes	Miércoles	Jueves	Viernes	Sábado	Domingo
Música	x		x				
Baile		x		x	x		
Matemáticas		x					
Ajedrez			x		x		

Si usamos porcentajes en lugar de una medida exacta, obtendremos un resultado relativo al tamaño de la ventana del navegador. Según la ocasión, nos convendrá emplear un tipo u otro.

Relleno

Probaremos ahora a modificar su relleno, es decir, la distancia imaginaria entre un hipotético borde y la imagen propiamente dicha. Para ello emplearemos la propiedad **padding**, exactamente igual que hicimos con la anterior. Probemos con un valor menos exagerado:

```
img { padding: 5px; }
```

Actividad 1

Tomando una página de ejemplo, modificaremos los valores de márgenes y de rellenos de diferentes elementos. Pensando en las etiquetas que tenemos en la página, intentaremos cambiar aquellas que pueda parecer que no tienen esta cualidad.

Borde

Si recargamos la página con esta incorporación, observaremos que, en efecto, la imagen se separa un poco más, esos 5 píxeles por cada lado, pero no es posible distinguir dónde acaba el efecto del margen y comienza el del relleno. Para poder diferenciar los valores, deberíamos tener un borde en la imagen.

```
img {

    border-width: 2px;
    border-style: solid;
    border-color: #007000;
    }
```

Con los conocimientos que tenemos ya de CSS podemos intuir con facilidad qué es lo que hace cada una de esas tres propiedades: en una definimos el **grosor del borde**, en otra el **tipo de línea** y en la última su **color**. El resultado se muestra en la figura:

Ahora sí que se aprecia la distancia entre el borde y el contenido, qué es de 5 píxeles, y entre el borde y el título, por ejemplo, que es de 20 píxeles, más lo que esté configurado de forma predeterminada.

Podemos probar a aplicar esas propiedades a cualquiera de las etiquetas de nuestra página web. Todas ellas se verán afectadas, ya que todas tienen esas propiedades asociadas. En la figura hemos incrementado el relleno de cada celda (la etiqueta **<td>**):

```
td {padding: 5px;}
```

Así conseguimos que la tabla tenga volumen y que el texto no se acerque demasiado a los bordes.

Laterales

Tanto **margin**, como **padding** y **border** se pueden emplear para modificar laterales de una caja, con independencia de los demás. Añadiendo a cada uno de ellos la variación **-left** (izquierda), **-right** (derecha), **-top** (arriba) o **-bottom** (abajo) conseguimos que sólo afecte al valor o valores indicados.

En la figura hemos aplicado estas propiedades para el título de la página:

```
h1 {
    margin-top:40px;
    padding-left: 5px;
    padding-right:5px;
    border-top-width: 2px;
    border-top-style: dotted;
    border-top-color: #007000;
    border-bottom-width: 2px;
    border-bottom-style: double;
    border-bottom-color: #007000;
    }
```

Todo el contenido de la etiqueta **<h1>** (la imagen es parte del encabezado) ha tomado los valores indicados.

Actividad 2

Continuando con el ejemplo anterior, iremos modificando valores parciales, retirando la propiedad general para reemplazarla por otras dos o tres que modifiquen sólo algunos laterales.

Más opciones para los bordes

Para los bordes podemos definir tres propiedades: su anchura, su estilo y su color. La anchura y el color se definen con las medidas habituales y los sistemas que ya hemos analizado. El **estilo**, por su parte, se basa en una serie de valores concretos:

- **dotted**: punteado.
- **dashed**: línea discontinua.
- **solid**: línea continua.
- **double**: línea doble.
- **groove**: tipo de relieve.
- **ridge**: tipo de relieve.
- **inset**: tipo de relieve.
- **outset**: tipo de relieve.
- **none**: empleado para indicar que no habrá borde

Es cuestión de probar algunos modelos. El valor **solid** es la línea sencilla, la más empleada.

Como ya sucedía con otras propiedades, podemos reagrupar los valores referidos a los bordes en una sola propiedad genérica denominada **border**. Para ello estableceremos los valores separados por espacios y en el orden de **tamaño, estilo**

y color, como en este ejemplo que haría la misma función que el recuadro anterior:

```
img { border: 2px solid #007000;}
```

Actividad 3

Reemplazaremos todos los bordes que hayamos aplicado hasta ahora utilizando el modelo condensado.

Esquinas redondeadas

Con los estilos actuales podemos trazar un borde alrededor de una figura y que tenga sus esquinas redondeadas.

La propiedad que lo permite es **border-radius**, acompañada de un valor numérico. El ejemplo anterior, con la incorporación de esta propiedad, daría como resultado el rectángulo de la figura:

```
img {
    border: 2px solid #007000;
    border-radius:25px;
    }
```

Para obtener un efecto apropiado y que el borde no chocase con el contenido, hemos incrementado un poco el relleno. Esta es la regla completa para la imagen:

```
img {
    text-align: left;
    vertical-align: top;
    margin: 20px;
    padding: 20px;
    border: 2px solid #007000;
    border-radius:25px;
    }
```

Pregunta Verdadero-Falso

Las siguientes afirmaciones, ¿son verdaderas o falsas?

La propiedad border-ridge, acompañada de un valor numérico nos proporciona esquinas redondeadas en una

imagen.

Verdadero ○ Falso ○

La opción padding se emplea para modificar laterales de una caja teniendo en cuenta para ello que los parámetros margin y border estén previamente definidos.

Verdadero ○ Falso ○

Sombras

Las modernas hojas de estilo proporcionan a cualquier elemento la capacidad de proyectar una sombra. Ya vimos que esto funcionaba con el texto, pero además contamos con la propiedad **box-shadow** para crear sombras en cualquier caja de nuestra página web, lo que hace que sea posible aplicarselo a cualquier elemento.

```
table {box-shadow: 8px 8px 6px #aaaaaa; }
```

Los valores que conforman la sombra son similares a los que vimos para las sombras de texto, es decir, **desplazamiento horizontal**, **vertical**, **difuminado** y **color de sombra.**

En el caso de la tabla, con una sombra clara, obtenemos el resultado de la figura:

	Lunes	Martes	Miércoles	Jueves	Viernes	Sábado	Domingo
Música	x		x				
Baile		x		x	x		
Matemáticas		x					
Ajedrez			x		x		

Ejemplo de planificación semanal

Este tipo de efectos combinan muy bien con las imágenes, como se puede observar en la figura siguiente:

Nota

Es conveniente emplear siempre los mismos valores de sombra dentro de una determinada página web. De otro modo el efecto que se produce quedaría descompensado, al no quedar definido el origen de la luz que simula la proyección de sombras.

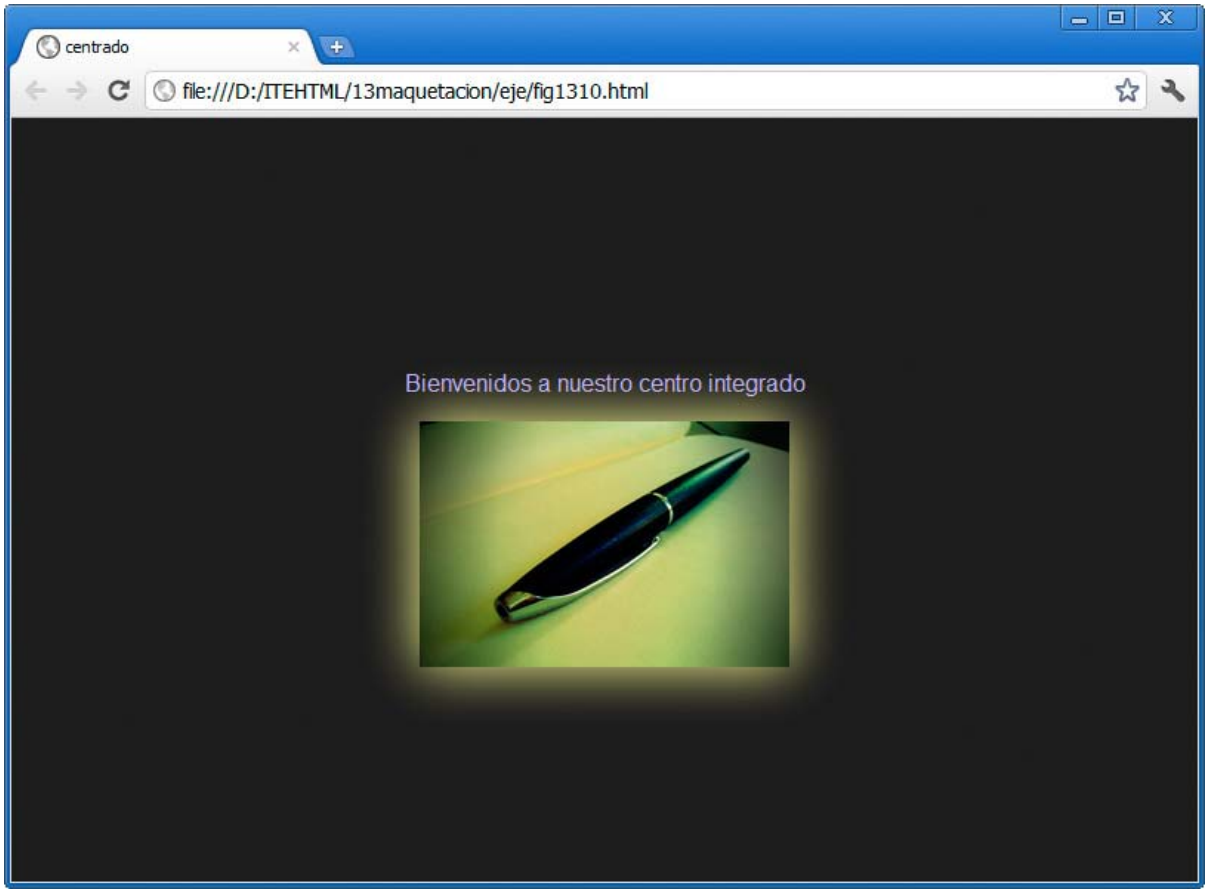
Actividad 4

Probaremos a aplicar bordes redondeados o sombras a diferentes elementos de nuestra página.

Brillo

Desde siempre el efecto de sombra se puede emplear también para producir un efecto de brillo (se suele denominar *glow* en las aplicaciones en inglés). Usando una línea similar a la siguiente podemos conseguir un efecto parecido al de la figura:

```
img{ box-shadow: 0px 0px 60px #FFFF99;}
```



Tenemos una sombra sin desplazamiento, con una difusión muy amplia, de 60 píxeles, y sobre todo con un color blanco o amarillo. Esos efectos hacen que parezca que el objeto está iluminado por detrás.

Actividad 5

Intentaremos conseguir un efecto de brillo, procurando que el resplandor se mueva en espectros de la luz que le den una apariencia fria, con tonos blancos o azules.

Anchura y altura de una caja

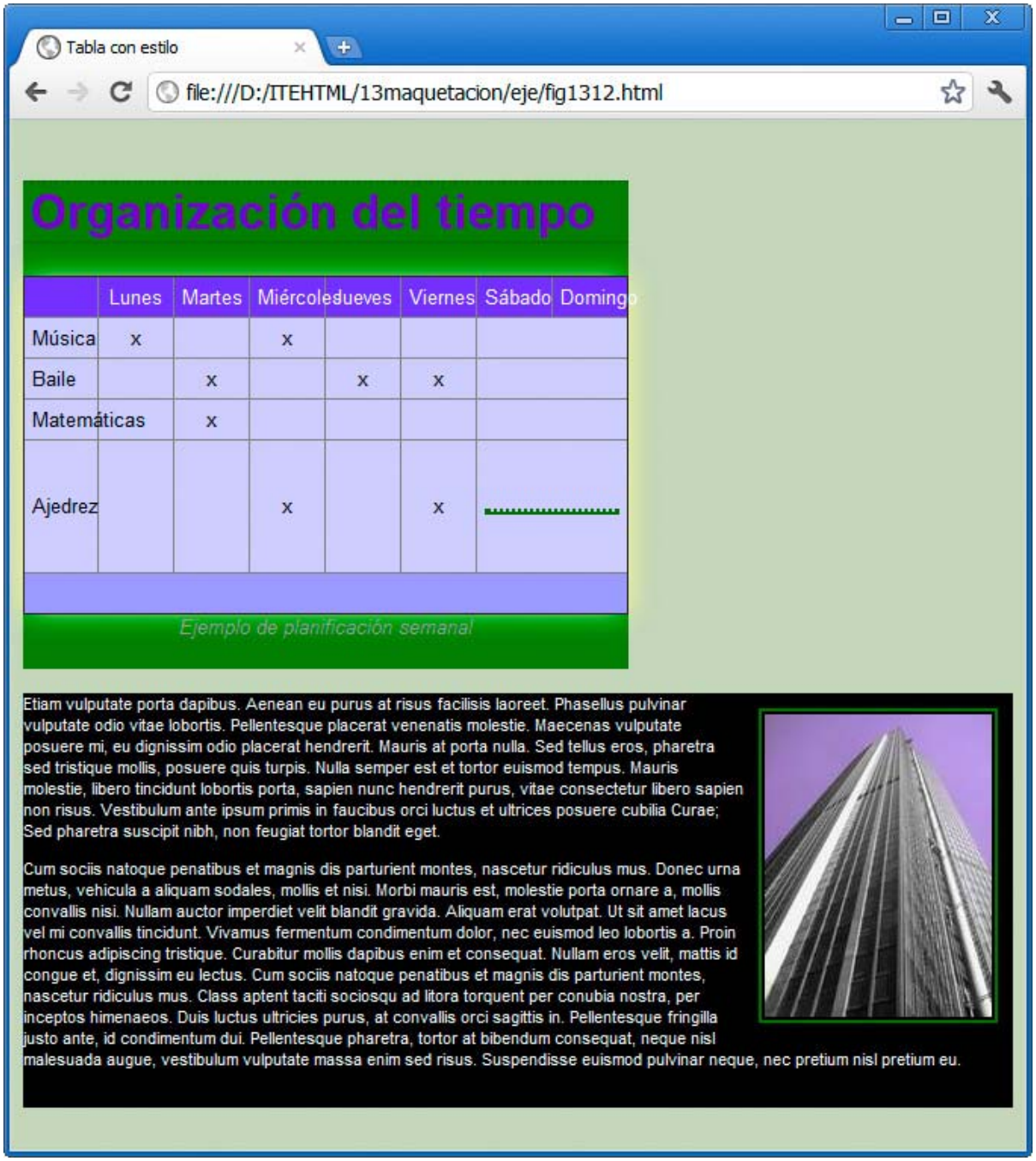
El ejemplo anterior nos abre la puerta para algunas posibilidades más de la ubicación de los elementos de una página web.

Cada elemento HTML de una página web cuenta con una anchura y una altura específica. En muchos casos esas dimensiones se las proporciona el propio contenido, como en un párrafo o una imagen, por ejemplo. Esos valores de anchura (**width**) y de altura (**height**) pueden ser modificados mediante las hojas de estilo, gracias a las propiedades del mismo nombre.

Así podemos hacer párrafos más estrechos, imágenes que se sobredimensionen o simplemente ajustar diferentes bloques, para que se acomoden correctamente en la pantalla.

Los valores **width** y **height** se acompañan de un valor numérico exacto o de un porcentaje, como en otras muchas propiedades.

Observa el ejemplo de la figura:



Hemos dividido los contenidos de la página en dos grandes bloques mediante sendas etiquetas ** **, quedando así la página:

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<title>Tabla con estilo</title>
<link href="general.css" rel="stylesheet" type="text/css">
</head>

<body>

<div id="bloqueprincipal">
<h1>Organizaci&oacute;n del tiempo</h1>
[Tabla de días de la semana]
</div>

<div id="bloquelateral">
<p><img id="imgbuilding" alt="" src="resources/buildingsmall.jpg">
```

```
[Texto de ejemplo en latín.]</p>
</div>

</body>

</html>
```

Y en la hoja de estilos hemos incorporado las propiedades de los dos ** **.

```
#bloqueprincipal {

    width: 400px;
    background-color: rgb(0, 126, 0);
    }

#bloquelateral {

    background-color: rgb(0, 0, 0);
    font-family: Arial, Helvetica, sans-serif;
    font-size: 0.7em;
    color: rgb(255, 255, 255);
    width: 100%
    }
```

Nuestro objetivo al cambiar el color del fondo es que se destaque el efecto que se produce al cambiar el tamaño de cada ** **. Si probamos a redimensionar la pantalla, el primer bloque mantendrá constante su tamaño, mientras que el segundo siempre ocupará el 100% del ancho de la pantalla.

Actividad 6

Hay mucho que experimentar aquí. Probaremos a crear dos bloques similares a los del ejemplo y comprobar lo que sucede al indicar valores relativos como el 50% o valores fijos menores.

Con las alturas sucederá lo mismo. Podemos indicar valores porcentuales o exactos.

Controlando el contenido

Al igual que sucede con la anchura, si el contenido no cabe dentro del bloque, lo rebasará quedando fuera del mismo, como se muestra en la figura:

En este caso el estilo del bloque llamado **#bloquelateral** ha quedado así:

```
#bloquelateral {

    width:30%;
    height: 120px;
    }
```

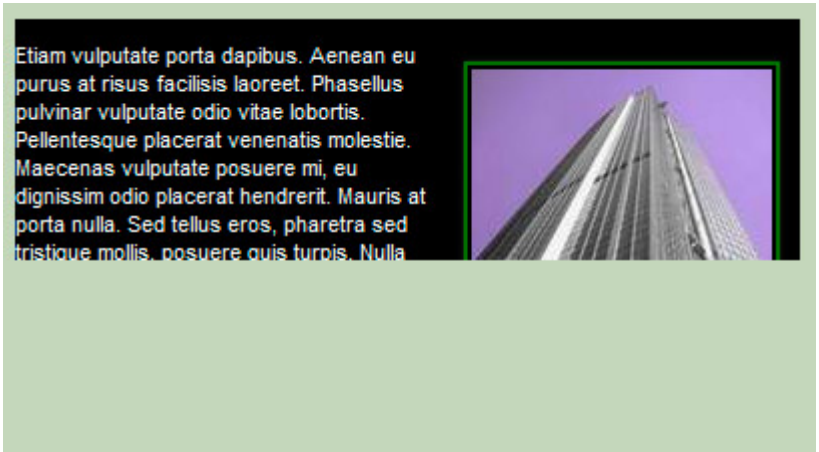
Contamos con una propiedad denominada **overflow** (y otras dos llamadas **overflow-x** y **overflow-y**) que se emplean para evitar que el contenido salga de su caja correspondiente.

Los valores que toma esta propiedad son:

- **visible**: hará que se omita la propiedad de anchura o altura y la caja se ampliará hasta cubrir todo el contenido.
- **hidden**: el contenido que no cabe en la caja se ocultará.
- **scroll**: se añadirá una barra de desplazamiento, para poder desplazarse dentro de la caja.
- **auto**: dejaremos en manos del navegador la decisión, que optará por la más apropiada.

La figura muestra de nuevo ese bloque con una propiedad **overflow** añadida:

```
#bloquelateral {

    width:30%;
    height: 120px;
    overflow:hidden;
    }
```



Y así se vería con la propiedad **overflow-y:scroll;**

Alto y ancho mínimo y máximo

Para completar el control del tamaño de las cajas podemos emplear las propiedades **min-width**, **max-width**, **min-height** y **max-height**. Respectivamente controlan el ancho mínimo, el máximo, el alto mínimo y el máximo de cualquier elemento HTML que incorporemos a una página web.

Con ellas conseguiremos que el elemento en cuestión nunca supere los límites establecidos.

La propiedad **overflow** nos será de gran ayuda para controlar lo que debe hacer el navegador cuando un contenido supera esos valores.

Pregunta de Elección Múltiple

La propiedad "*overflow:scroll*" hará:

- Que se omita la propiedad de anchura o altura y la caja se ampliará hasta cubrir todo el contenido.
- El contenido que no cabe en la caja se ocultará.
- Se añadirá una barra de desplazamiento, para poder desplazarse dentro de la caja.
- Dejará en manos del navegador la decisión, que optará por la más apropiada.

Actividad 7

Realizaremos algunas pruebas con los valores de mínimo y máximo, así como con la propiedad **overflow**.

Visualización de elementos

Hasta el momento todos nuestros elementos se han mostrado de forma similar, con un comportamiento secuencial. Veremos algunos cambios que se pueden aplicar al mostrar nuestros contenidos.

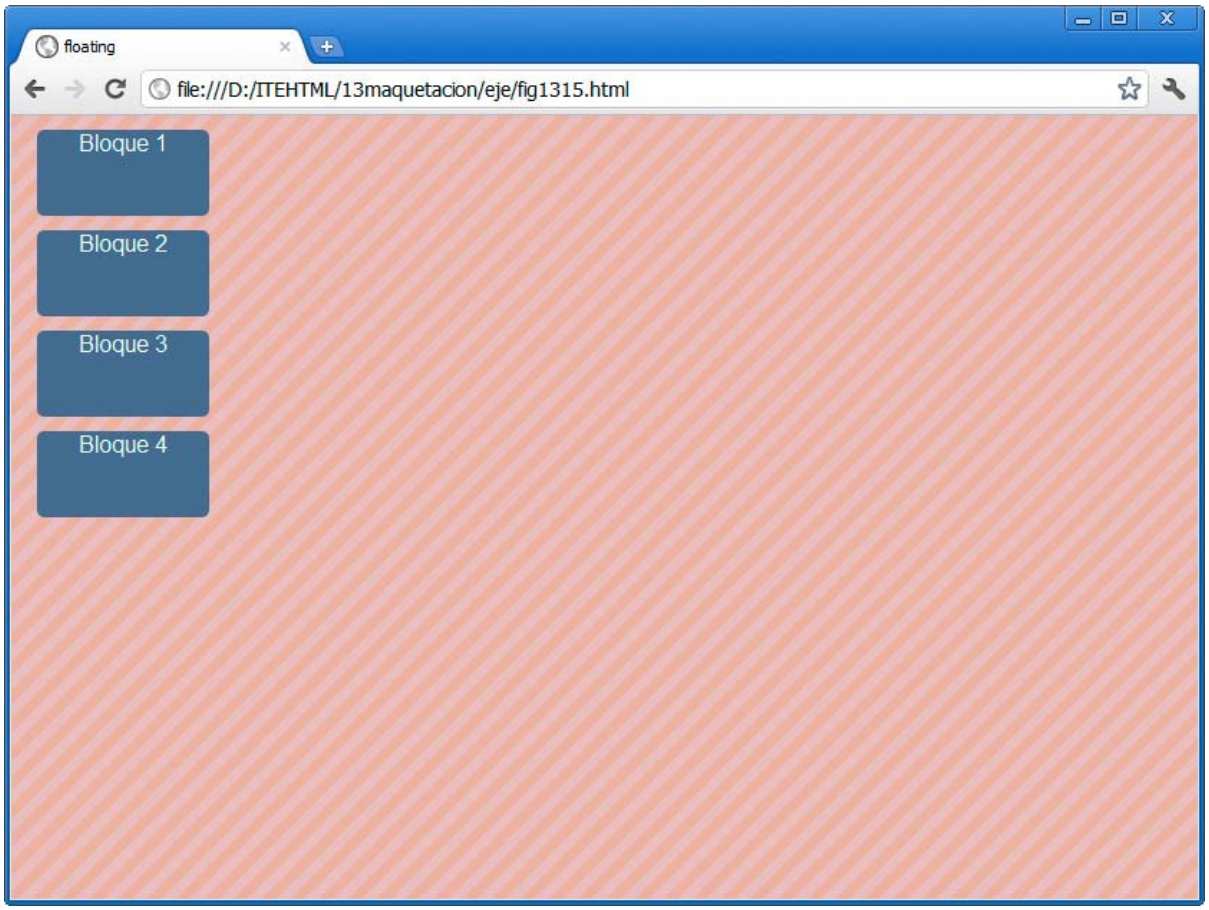
Para hacer nuestra pruebas emplearemos esta sencilla página web:

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<title>floating</title>
```

```
<link href="floating.css" rel="stylesheet" type="text/css">
</head>

<body>
<div id="bloque1">Bloque 1</div>
<div id="bloque2">Bloque 2</div>
<div id="bloque3">Bloque 3</div>
<div id="bloque4">Bloque 4</div>
</body>

</html>
```

Consta de cuatro simples bloques con una pequeña frase. Al aplicarle los estilos siguientes conseguimos el resultado de la figura.

```
body {

    background-image: url("resources/fondo.png");
    font-family: Arial, Helvetica, sans-serif;
    }

div {

    width: 120px;
    height: 80px;
    background-color: rgb(82, 108, 142);
    color: rgb(245, 255, 244);
    margin: 10px;
    border-radius:5px;
    text-align: center;
    }
```



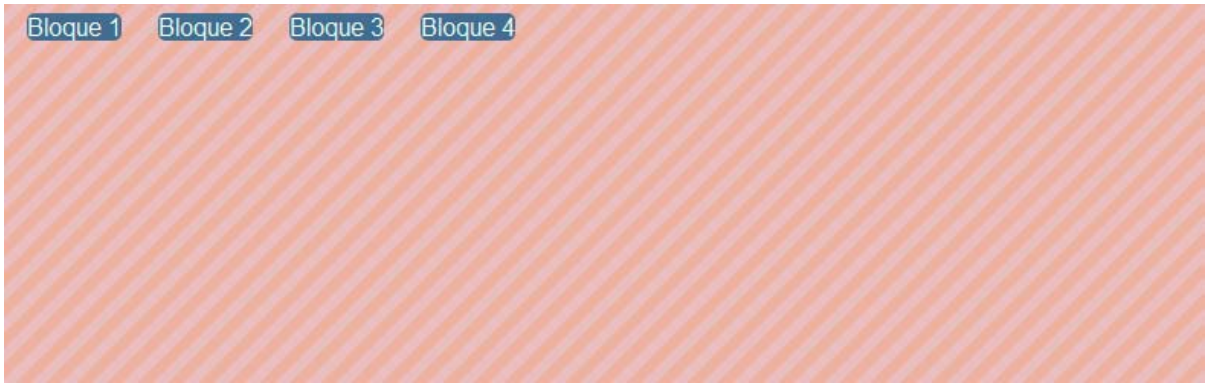
En CSS disponemos de la propiedad **display** para modificar la visualización de los elementos de la página.

Por ejemplo, al añadir a los valores de ** ** la propiedad **display:block,** forzaremos a que antes y después del elemento seleccionado se produzca un salto de línea. Es decir, exactamente igual que se encuentra ahora, ya que los elementos delimitados por ** ** siempre llevan ese salto de línea antes y después.

Para apreciar el efecto de **display,** podemos aplicar la siguiente línea:

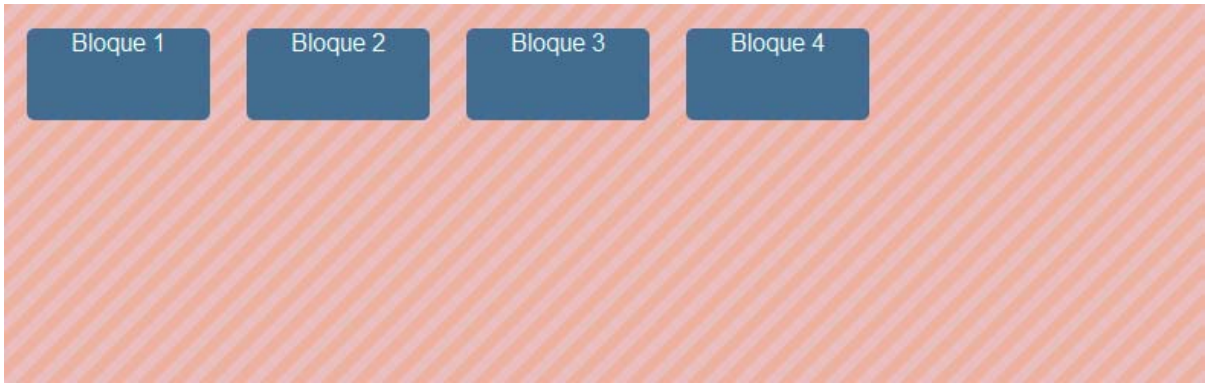
```
div {
    display:inline;
    }
```

El resultado sería el siguiente:



Los elementos se sitúan horizontalmente, sin los saltos de línea. Se pierden sus dimensiones, como se puede observar. En cambio, usando **inline-block** se distribuyen horizontalmente con sus características de bloque intactas.

```
div{
    display:inline-block;
    }
```



Nota

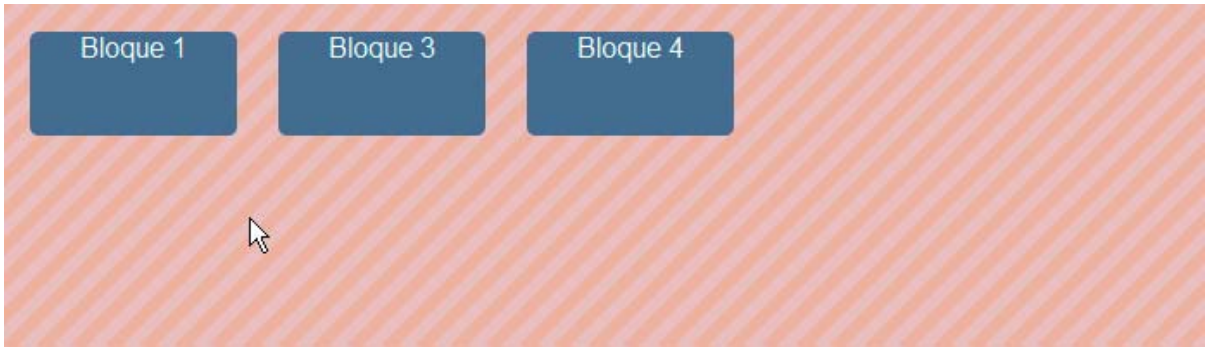
Todas estas opciones son muy apropiadas para diseñar menús dentro de nuestras páginas.

Elementos no mostrados

Otro valor interesante es **none**. Observemos lo que sucede al aplicar este valor al segundo bloque:

```
div#bloque2 { display:none;}
```

Como se observa en la figura, el bloque 2 ha desaparecido:



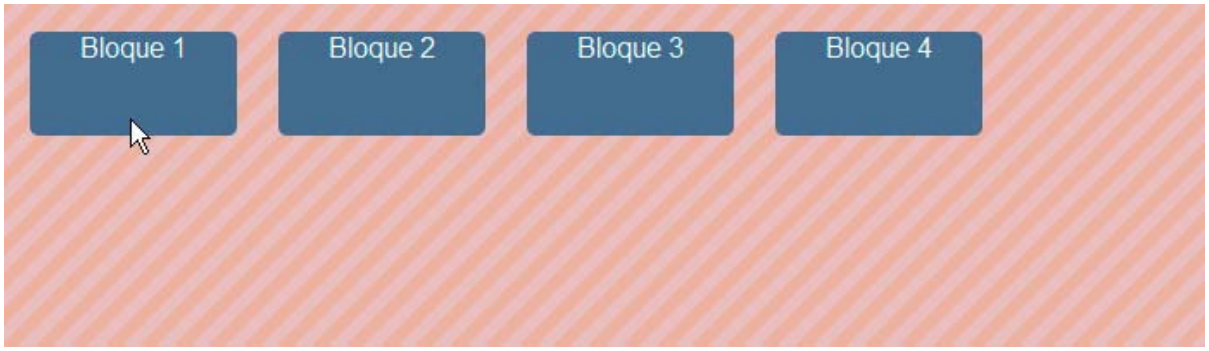
¿Qué utilidad tiene esto? Con *JavaScript*, el lenguaje de *script* que podemos emplear en los navegadores resulta sencillo modificar durante la reproducción de la página una propiedad concreta de una hoja de estilos.

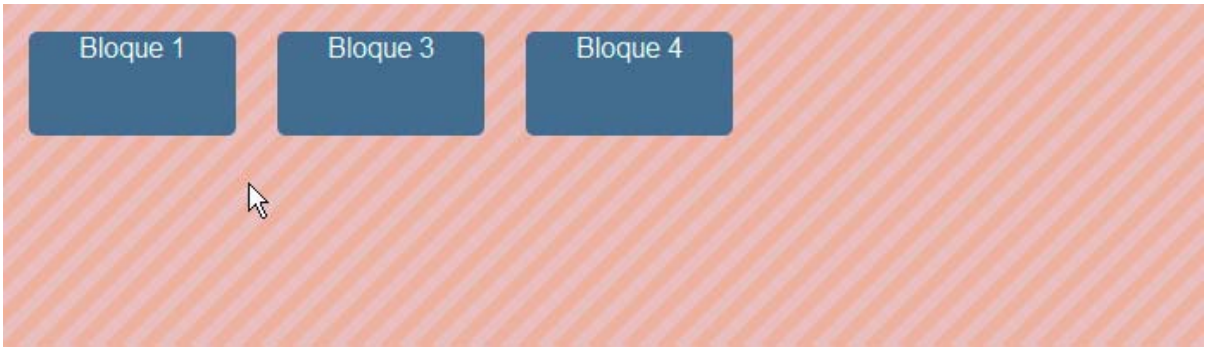
Podemos probar a añadir el código de un pequeño *script* a la cabecera de la página:

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<title>floating</title>
<link href="floating.css" rel="stylesheet" type="text/css">
<script language="JavaScript">

        function mostrar() {
        document.getElementById('bloque2').style.display="inline-block";
        }

</script>

</head>

<body>
<div id="bloque1" onmouseover="mostrar()">Bloque 1</div>
<div id="bloque2">Bloque 2</div>
<div id="bloque3">Bloque 3</div>
<div id="bloque4">Bloque 4</div>
</body>

</html>
```

Además hemos modificado el bloque 1 añadiendo un parámetro denominado **onmouseover**. Lo que sucede en adelante es que el ** ** *bloque1* será sensible al momento en el que el ratón pase sobre él. El resultado es que, cuando eso suceda, se ejecutará una pequeña acción denominada mostrar. Esa acción lo único que hace es localizar el bloque 2 (**document.getElementById('bloque2')**) y modificar su propiedad **display** (**style.display="inline-block";**).

Probaremos la página para ver si funciona. La figura siguiente incluye un ejemplo un poco más sofisticado que muestra el bloque y a su vez lo oculta, cuando el ratón deja de estar sobre el bloque 1 (se hace mediante el parámetro **onmouseout**).





Si es la primera vez que utilizamos *JavaScript,* nos parecerá mágico por un lado, pero complejo por otro. Es normal. Debemos prestar muchísima atención a cada carácter que introducimos. Cualquier despiste hace que el ejemplo deje de funcionar completamente.

Actividad 8

El primer ejemplo es una buena práctica para familiarizarnos con estas técnicas y, aún más, probar a hacer el que muestra y oculta el bloque. Como ayuda, éste sería el código *JavaScript* completo de este último caso:

```
<script language="JavaScript">

    function mostrar() {
    document.getElementById('bloque2').style.display="inline-block";
    }
    function ocultar() {
    document.getElementById('bloque2').style.display="none";
    }

</script>
```

Para terminar con la visualización, además de **block**, **inline**, **inline-block** y **none**, contamos con otras opciones específicas relacionadas con tablas (**table**, **table-cell**, etc.), para mostrarlos como elementos de lista (**list-item**) y alguna opción más, aunque las primeras que hemos visto son las más empleadas.

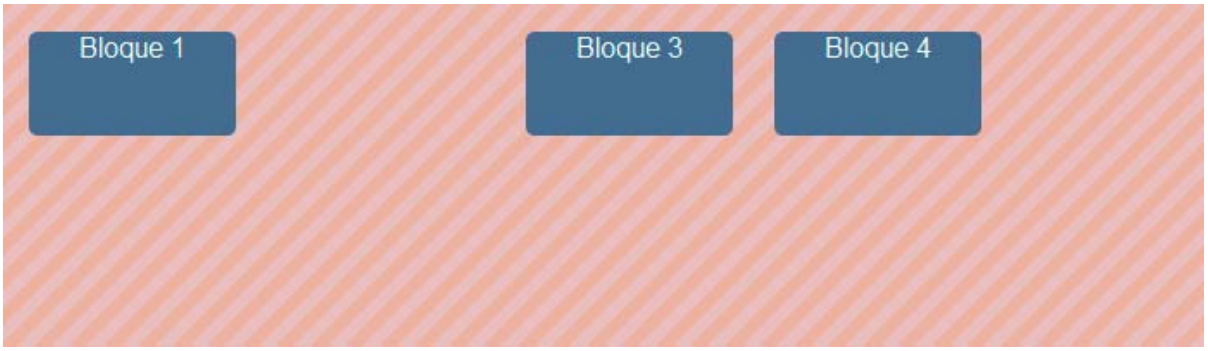
Visibilidad

El ejemplo que retiraba de la vista un elemento queda más completo al conocer la propiedad **visibility**. Esta propiedad se emplea para ocultar o mostrar un elemento HTML, con la particularidad de que el espacio que ocuparía el objeto se mantiene en la pantalla, aunque éste esté oculto. Esa es la principal diferencia con **display:none**, que hace que el hueco del objeto se cubra con el resto de los elementos de la página.

Así, el siguiente código:

```
div#bloque2 { visibility:hidden;}
```

se vería así, dejando el hueco.



Los valores que puede tomar la propiedad **visibility** son **hidden** (oculto), **visible** (se muestra el elemento) o **collapse** (se aplica a las tablas; en este caso no se dejará el hueco).

Pregunta de Elección Múltiple

Si dentro de nuestra web queremos saltar un contenido, pero queremos que quede su hueco, ¿qué código debemos teclear?

- *div{display:inline-block}.*
- *div#contenido { display:none;}.*
- *div#contenido { visibility:hidden;}.*

Actividad 9

En una página web escribiremos esta secuencia:

```
<p> No es lo mismo pero es <span id="palabra">bastante</span> parecido</p>
```

Utilizando la propiedad **display** y **visiblity** haremos desaparecer la palabra situada en la etiqueta ****. Debemos apreciar la diferencia entre ambos métodos.

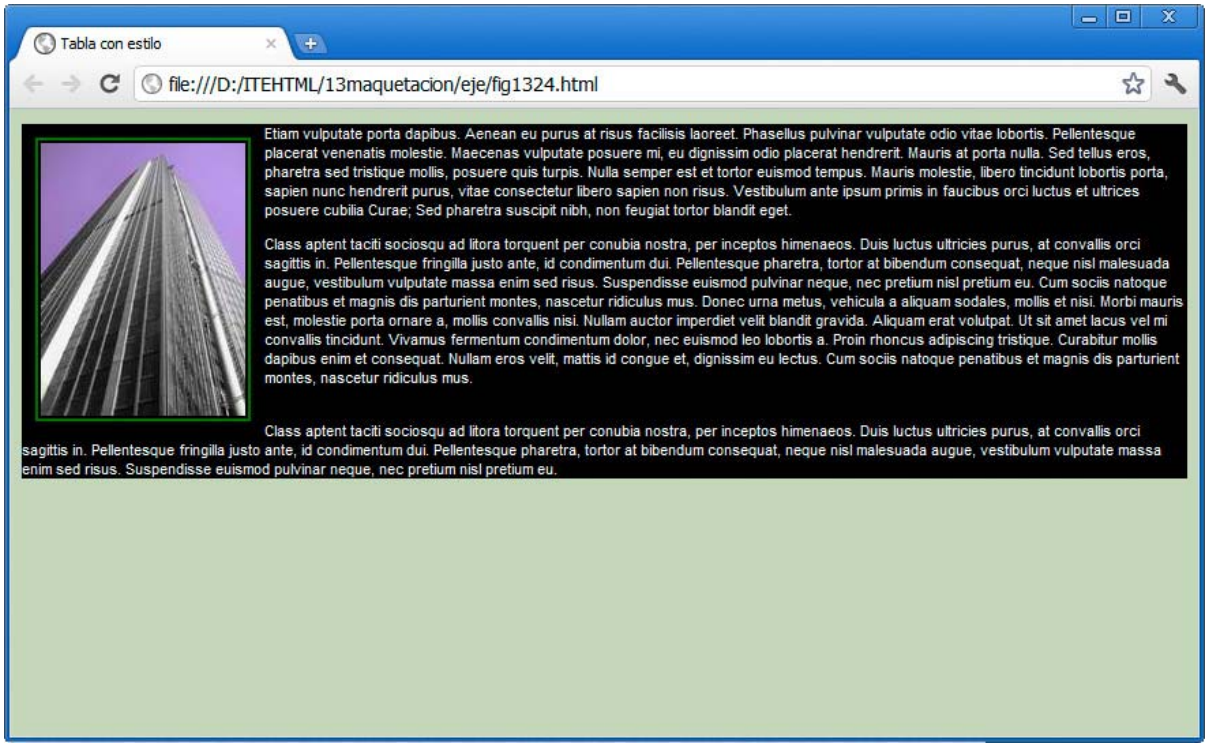
Elementos flotantes

Los elementos de una página web pueden reubicarse a la izquierda o a la derecha con tan sólo emplear la propiedad **float**, haciendo que el resto del contenido se sitúe alrededor de ese elemento.

En el siguiente ejemplo la regla:

```
img { float:left;}
```

provoca que el texto se sitúe alrededor de la imagen.



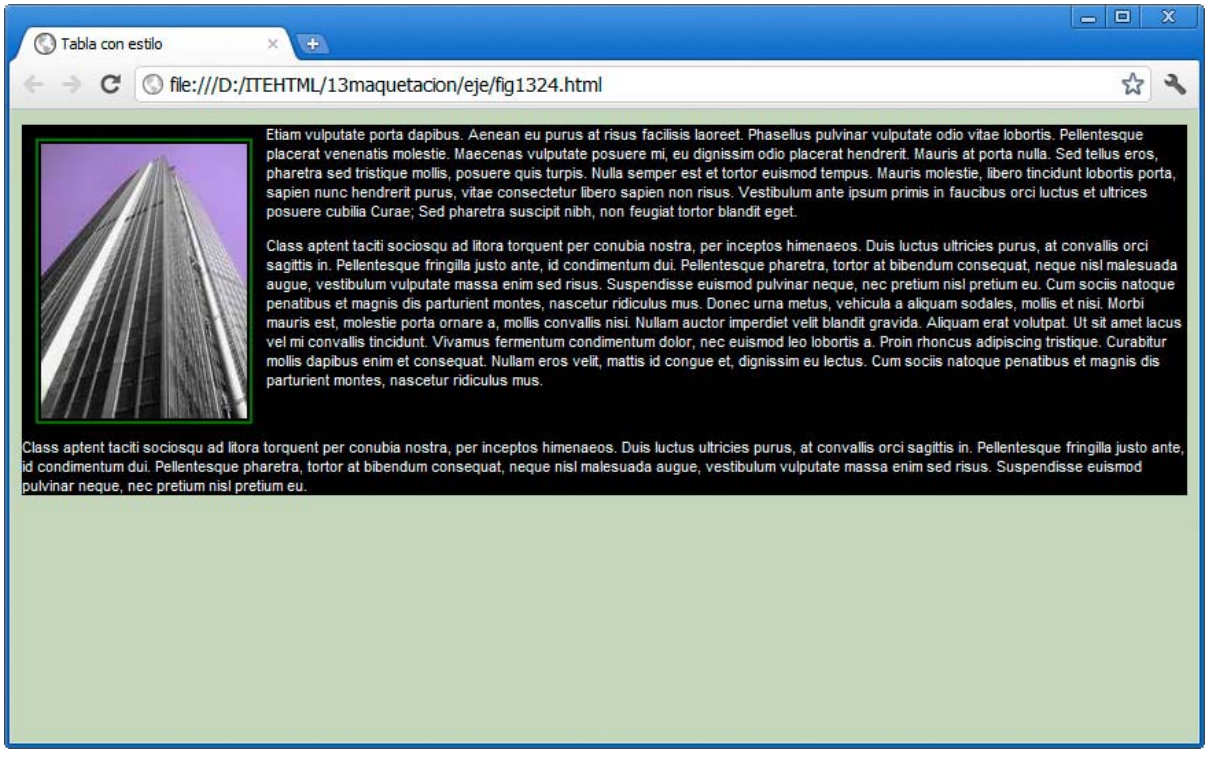
Sus valores pueden ser **right** (derecha), **left** (izquierda) o **none** (para eliminar cualquier flotación).

Eliminar flotaciones

La propiedad **clear**, seguida del valor **right** (derecha), **left** (izquierda), **both** (tanto izquierda como derecha o **none** (desactivar valores previos) nos permite deshacer flotaciones de objetos anteriores y restaurar el orden normal.

Observa cómo varía el ejemplo anterior con esta línea:

```
div#bloque2 { clear:both;}
```



El último párrafo, que se encuentra en el *bloque2*, ya no se ajusta a la imagen. Esta técnica se emplea con frecuencia al comenzar nuevas secciones en un documento, como las cabeceras, el pie de página, etc.

Pregunta Verdadero-Falso

La siguiente afirmación, ¿es verdadera o falsa?

La técnica *div#bloque2 { clear:both;}* se emplea al comenzar nuevas secciones en un documento, como las cabeceras, el pie de página, etc.ya que deshace flotaciones de objetos anteriores y restaura el orden normal.

Verdadero ○ Falso ○

Posicionamiento

Avanzaremos ahora un escalón más en el camino de libertad organizativa que nos proporcionan las hojas de estilo en cascada mediante las propiedades de posicionamiento.

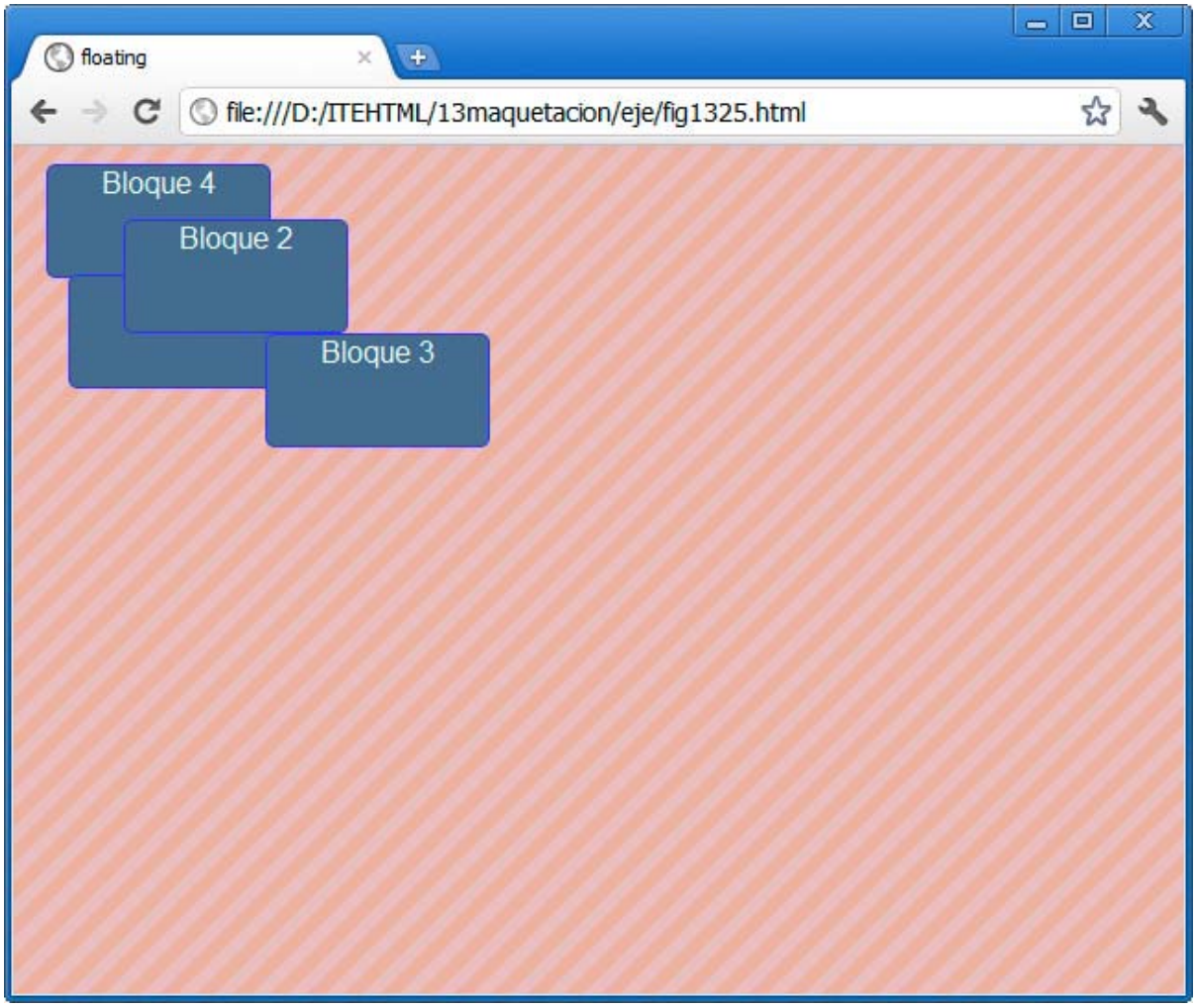
La propiedad ***position*** se emplea para definir el tipo de posicionamiento de un objeto en la pantalla según diferentes valores. Además se acompaña de las propiedades **left** (izquierda) y **top** (arriba) para especificar la distancia que tendrá el objeto respecto al borde izquierdo y al superior.

Hagamos una prueba con este ejemplo:

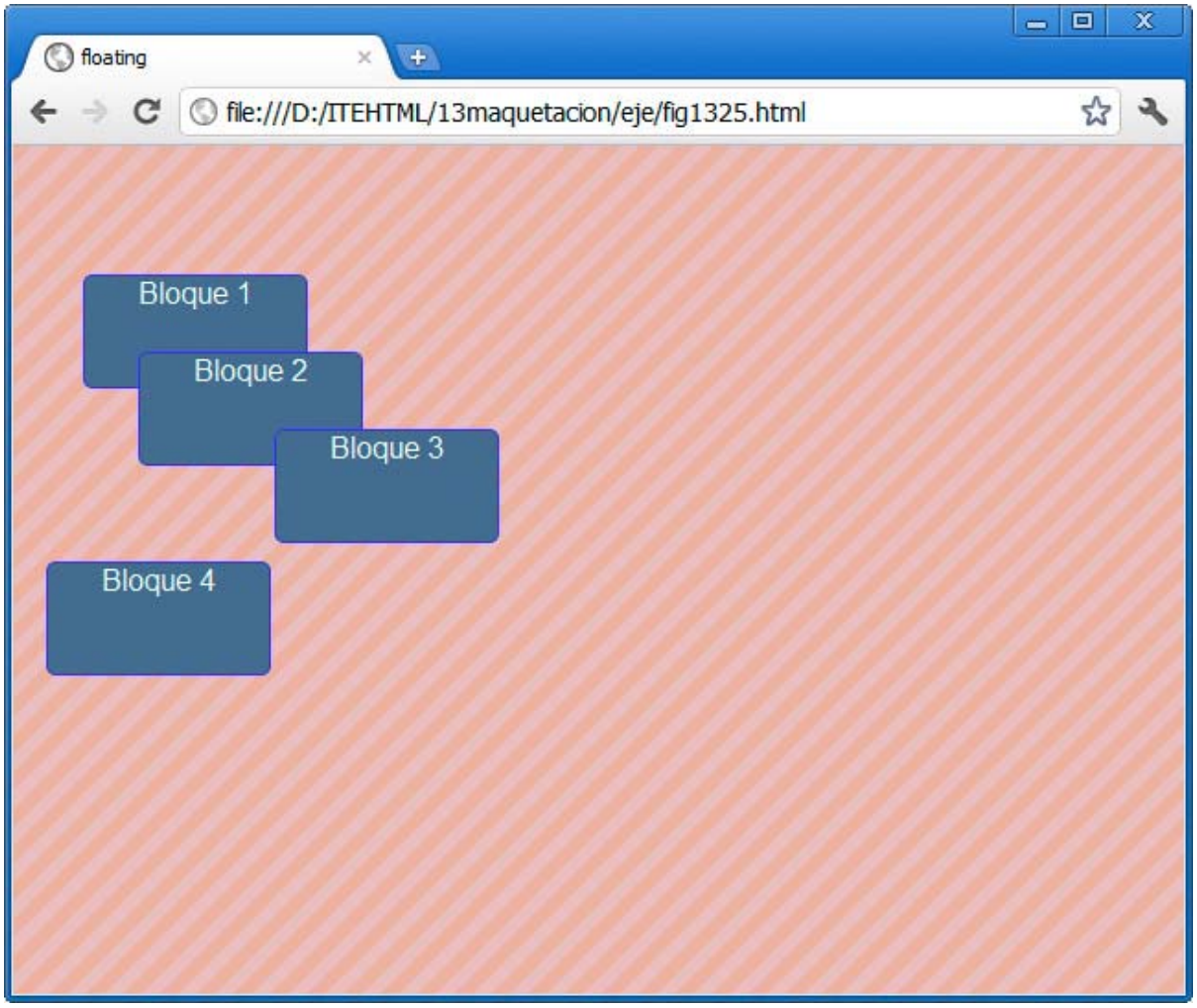
```
div#bloque1 { position:fixed;top:60px;left:20px;}
div#bloque2 { position:fixed;top:30px;left:50px;}
div#bloque3 { position:fixed;top:20%;left:20%;}
```

Cada valor diferente de *position* mostrará un resultado diferente:

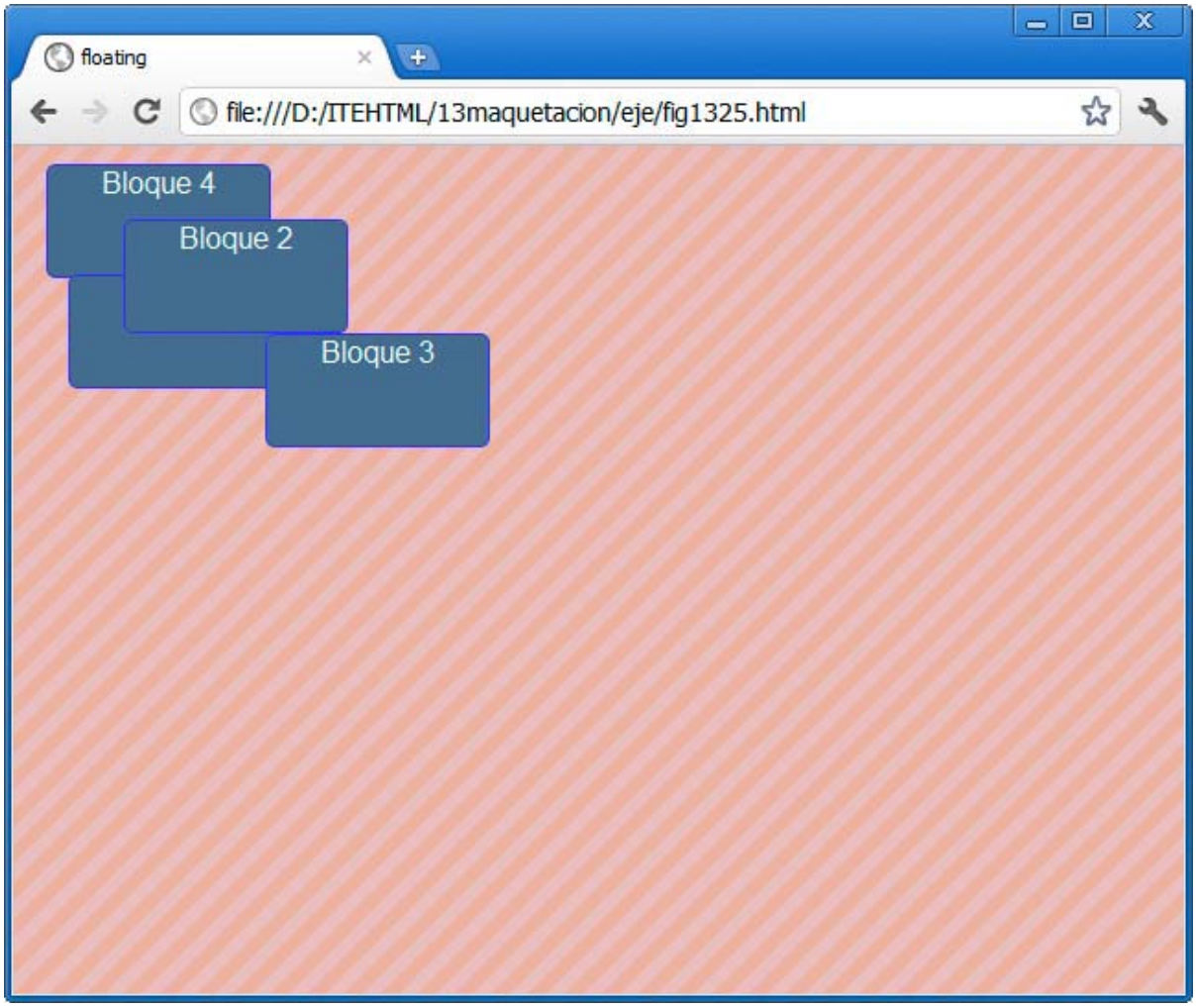
- **fixed**: indicamos que se van a especificar unos valores determinados; los elementos dejan de estar ocupando sus lugares en la pantalla. La figura muestra un ejemplo:



- **relative**: se indicarán también los lugares, pero en este caso sus huecos sí permanecerán. En la figura se ven los valores relativos. Podemos ver que el bloque 4 no se ha desplazado a la parte superior, como en el caso anterior, ya que las posiciones de los otros tres bloques se reservan.



- **absolute**: la posición de cada elemento se define respecto al anterior que la tenga modificada o del inicio de la página, si es el primer elemento absoluto.



Ya sea mediante porcentajes o con valores exactos, el uso de este tipo de posicionamiento nos permitirá distribuir los bloques de una página web en la posición deseada.

En el modelo que hemos trabajado a lo largo de este módulo resultaría muy simple reubicarlos horizontalmente, con el siguiente código:

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<title>Tabla con estilo</title>
<link href="position.css" rel="stylesheet" type="text/css">
</head>

<body>

	<div id="cabecera">
	...
	</div>

	<div id="bloqueprincipal">
	….
	</div>

	<div id="bloquelateral">
	….
	</div>

</body>
```

```
</html>
```

Cada uno de los tres bloques tiene algunos estilos aplicados:

```
#cabecera {

    font-family: Arial, Helvetica, sans-serif;
    color: rgb(255, 255, 255);
    }

#bloqueprincipal {

    width: 68%;
    background-color: rgb(0, 126, 0);
    position:absolute; top: 100px;
    }

#bloquelateral {

    background-color: rgb(0, 0, 0);
    font-family: Arial, Helvetica, sans-serif;
    font-size: 0.7em;
    color: rgb(255, 255, 255);
    position: absolute;left:70%;100px;
    }
```

Y éste es el resultado:

La combinación del 68% de anchura del bloque principal con el comienzo del bloque lateral, que lo hace en el 70%, nos permite tener un diseño que se ajusta a la resolución de la pantalla.

También podríamos haber optado por un modelo fijo, basado en píxeles en todos los tamaños.

Nota

Ésta es una de la partes del uso de las hojas de estilo que más cuesta controlar; la combinación entre tamaños, porcentajes y posiciones es a veces compleja y cuesta conseguir los resultados esperados, pero con práctica se consiguen.

Actividad 10

Diseñaremos una página web que tenga cuatro bloques diferenciados (los podemos hacer como ** **). El primero será la cabecera, el segundo y el tercero serán bloques centrales de contenidos y el cuarto lo emplearemos para hacer las veces de pie de página. Tanto el primero como el último deben ocupar todo el ancho de la página, mientras que los centrales deben repartirse el resto del espacio. Colorearemos cada ** ** con un tono diferente, para destacar más las diferencias.

Los planos

Al utilizar la propiedad *position* podemos colocar las capas una encima de otras. El plano en que se sitúa cada capa se puede modificar mediante la propiedad **z-index** seguida de un número.

Los valores más altos indican que el bloque se situará encima de los demás, en un plano más alto, mientras que los valores menores se situarán detrás. Este sistema de poner unos bloques sobre otros se utiliza para mostrar ventanas emergentes sobre la página, galerías de imágenes, etc.

Pregunta de Elección Múltiple

¿Qué significa el parámetro *fixed*?

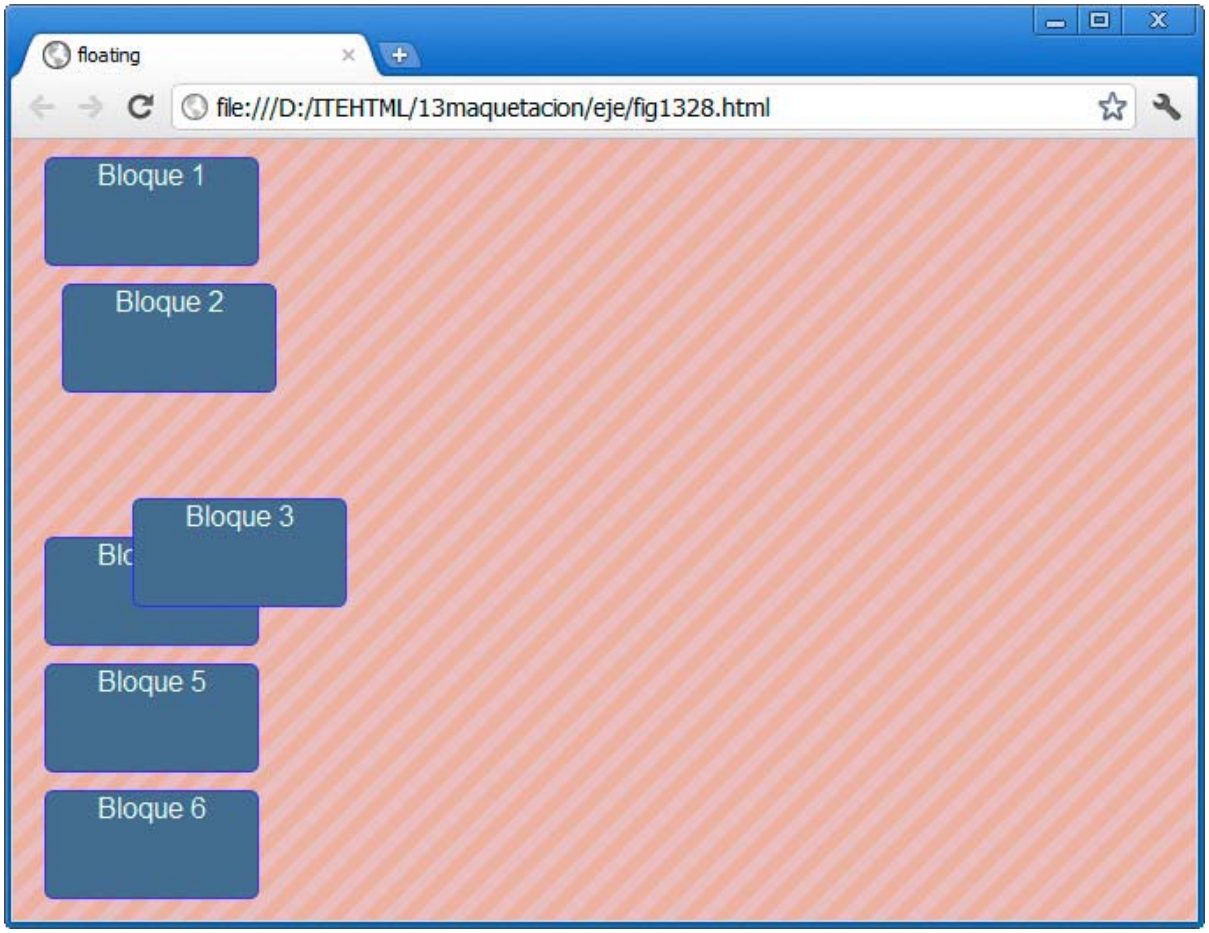
- Indica la posición de cada elemento definido respecto al anterior que la tenga modificada o del inicio de la página, si es el primer elemento absoluto.
- Indica que se van a especificar unos valores determinados respecto a la posición que van a ocupar.
- Indicarán el lugar del contenido pero en este caso sus huecos permanecerán.

Transformaciones y prefijos

Entre las opciones que encontramos en las hojas de estilo modernas se encuentra la propiedad **transform** que se emplea para aplicar transformaciones a un elemento. Éstas son las opciones disponibles:

- **translate**, **translate3d**, **translateX**, **translateY** y **translateZ**: definen diferentes tipos de traslaciones sobre el objeto indicado.
- **scale**, **scale3d**, **scaleY**, **scaleY** y **scaleZ**: redimensionan el objeto, en su conjunto, en ejes parciales.
- **rotate**, **rotate3d**, **rotateX**, **rotateY**, **rotateZ**: aplican una rotación al objeto, como en los casos anteriores; se rota en su conjunto o sólo en un plano específico. Van seguidas de un ángulo.
- **skew**, **skewX** y **skewY**: realizan un abatimiento del elemento.
- **perspective**: modifica su perspectiva.
- **matrix** y **matrix3d**: definen una transformación utilizando 6 valores o 16 en el segundo caso.

Para no alargar demasiado este apartado mostraremos sólo un par de ejemplos. La primera figura muestra el resultado obtenido mediante el uso de *traslate*:



Éste es el código que tendríamos que incorporar para conseguirlo:

```
div#bloque1 { -webkit-transform:none;}
div#bloque2 { -webkit-transform:translate(10px);}
div#bloque3 { -webkit-transform:translate(50px,50px);}
div#bloque4 { -webkit-transform:translate3d(50px,50px,40px);}
div#bloque5 { -webkit-transform:none;}
div#bloque6 { -webkit-transform:none;}
```

Los prefijos

Las transformaciones nos sirven para explicar un pequeño inconveniente del uso de propiedades modernas. En algunos casos, las propiedades más actuales no funcionan en todos los navegadores y en otros, cada navegador ha realizado su propia implementación de la propiedad, a la espera de que el grupo que define las hojas de estilo CSS3 complete del todo la definición de propiedades.

¿Cómo se diferencian, pues, las implementaciones que hace cada navegador? Añadiendo un prefijo específico diferente en cada caso. Así la propiedad **transform** es un caso prototípico. En la actualidad el panorama es el siguiente:

Tenemos la propiedad real, llamada **transform**, que no funciona aún en ningún navegador.

- *Internet Explorer* puede realizar todas las transformaciones 2D, pero para ello usaremos su propia versión llamada **-ms-transform**.
- *Firefox* también hace lo mismo, pero con su versión llamada **-moz-transform**.
- *Opera* funciona como los dos anteriores, pero con una versión llamada **-o-transform**.
- *Chrome* y *Safari*, que emplean el mismo motor de presentación de páginas, además de las 2D hacen las transformaciones 3D ya, pero con su propiedad **-webkit-transform**.

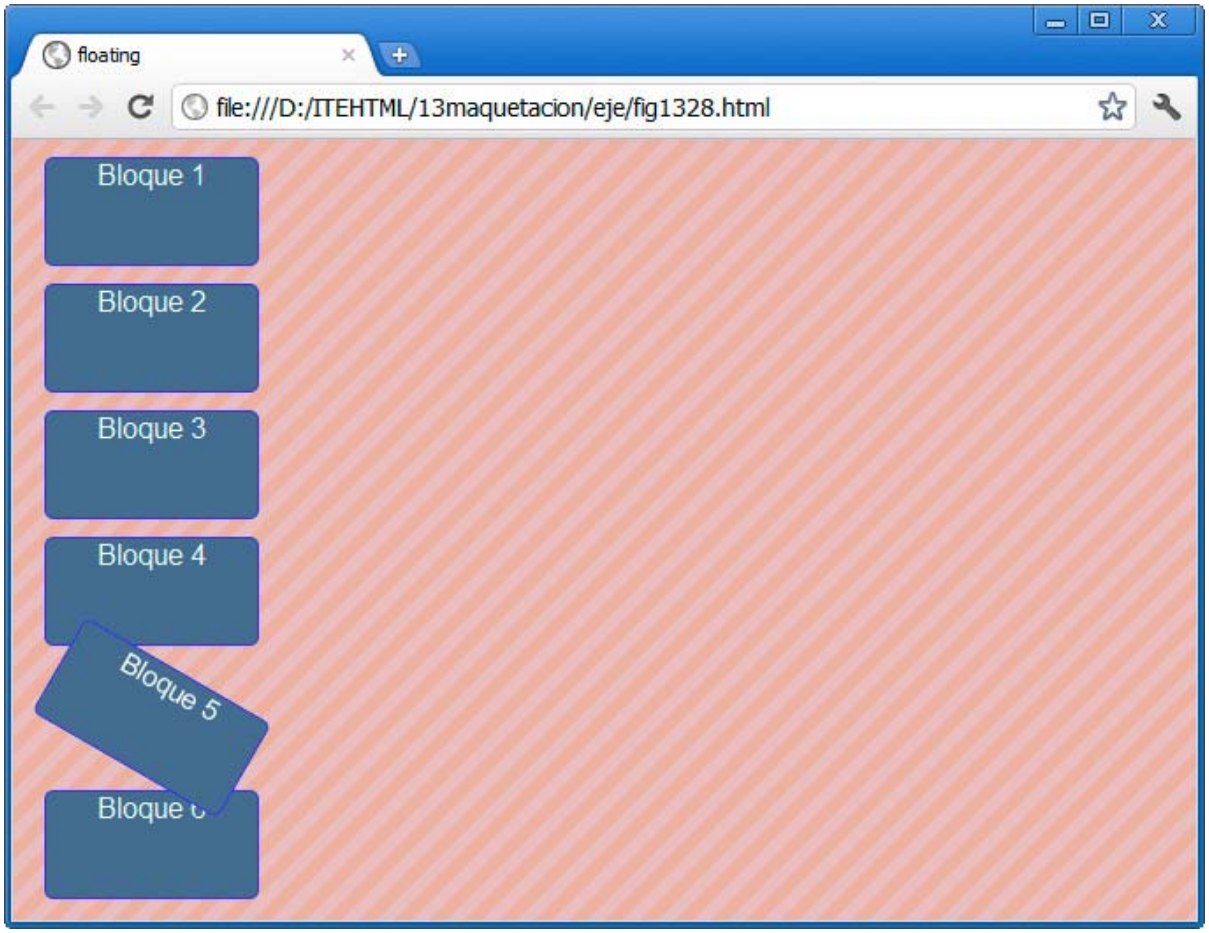
En este punto sólo podemos estar agradecidos porque sólo haya cinco navegadores web populares: imaginemos un mundo

con 50 de ellos...

Todo esto nos lleva a que, si quisiéramos aplicar una rotación de 30 grados a un elemento y quisiéramos que funcionase en cualquier navegador, deberíamos añadir una propiedad similar a ésta:

```
div#bloque5 {

    -ms-transform:rotate(30deg);
    -moz-transform:rotate(30deg);
    -o-transform:rotate(30deg);
    -webkit-transform:rotate(30deg);
    }
```

Y éste sería el resultado, ahora sí, en cualquier navegador:



Nota

Este es un problema a extinguir. Con cada nueva versión de los navegadores, desaparecen este tipo de prefijos y se tiende más a acoger los estándares definidos por CSS3.

En gran medida aplicaremos estilos mediante un editor web, por lo que esta complejidad añadida no será una tarea nuestra, sino del editor.

Transiciones

Las transiciones permiten definir la velocidad y el momento en que se aplica una variación en una propiedad.

Al igual que con las transformaciones, las transiciones no están implementadas en ningún navegador sin utilizar los prefijos

anteriores. Si lo hacemos, contaremos con la propiedad **transition-property**, que va seguida de alguna transición y de la propiedad **transition-duration**, seguida de un valor en segundos. Opcionalmente podemos añadir **transition-delay** para hacer una pausa, antes de que comience la transición.

El resultado es una animación que se desarrolla a lo largo del tiempo. Para ver cómo funciona probaremos a añadir lo siguiente a algún elemento de nuestra página web:

```
div#bloque1:hover {

    width:500px ;
    -webkit-transition-property: width;
    -webkit-transition-duration: 10s;
    -webkit-transition-delay: 1s;
    }
```

Al aplicarlo en *Chrome*, veremos cómo el elemento en cuestión va incrementando su anchura durante 10 segundos, pero sólo cuando haya pasado 1 segundo desde que se cargó la página. Para que esto tenga lugar, necesitaremos cambiar la propiedad de algún modo. Por eso definimos el estilo como: **hover** (cuando el ratón pase sobre el elemento), aunque también podríamos realizar la operación al pulsar un botón externo, usando eventos, etc.

La clave de su funcionamiento está en que va a transformar la anchura (**transition-property: width**) del elemento bloque1 para ampliar hasta los 500 píxeles pero, en lugar de hacerlo brúscamente, lo hará progresivamente durante 10 segundos (**transition-duration**).

Podemos indicar tantas propiedades como necesitemos y cambiaremos sólo aquellas que queramos mediante **transition-property** seguido de las propiedades, o todas juntas si indicamos **transition-property:all**.

Observa el vídeo para ver cómo funcionan las transiciones.

Vídeo

Cómo realizar una transición

Con esta técnica podemos añadir interactividad a cualquier elemento de una página web, cambiando colores progresivamente, mostrando sombras poco a poco, etc.

Pregunta Verdadero-Falso

La siguiente afirmación, ¿es verdadera o falsa?

Imagina que queremos realizar el abatimiento de un elemento y para que se pueda visualizar en diferentes navegadores escribimos el siguiente código:

```
div#elemento {

-ms-transform:skew(30deg);
-moz-transform:skew(30deg);
-o-transform:skew(30deg);
-webkit-transform:skew(30deg);
}
```

Verdadero ○ Falso ○

Actividad 11

Empleando transiciones diseñaremos una sombra casi imperceptible que se destaca fuertemente, al pasar el ratón sobre el elemento al que se lo apliquemos, por ejemplo un ** ** o un **<h1>**. Todo lo que necesitamos hacer es crear una sombra casi transparente y definir una propiedad **:hover** para ese elemento en el que la sombra es casi opaca, añadiendo, además las propiedades **transition** necesarias para que se produzca paulatinamente.

Es decir, los dos estilos que definiríamos podrían quedar así:

```
div#bloque1 {

    box-shadow: 8px 8px 6px #333333;
    }

div#bloque1:hover {

    box-shadow: 8px 8px 6px #333333;
    -webkit-transition-property: box-shadow;
    -webkit-transition-duration: 5s;
    -webkit-transition-delay: 1s;
    }
```

Probaremos también a variar el número de segundos hasta encontrar un equilibrio entre estética y usabilidad. El efecto no debe ser pesado ni entorpecer el trabajo del usuario, pero al mismo tiempo debe percibirse.

Animaciones

Las transiciones se pueden llevar un paso más allá empleando animaciones. El concepto es similar, pero utilizando fotogramas clave que definimos a través de una especie de función dentro de la hoja de estilos.

Estas líneas definirían esa función, en la que podemos definir tantos estados como necesitemos. En el ejemplos estamos indicando que, cuando la animación esté al 50%, la anchura del cuadro debe haber crecido hasta los 500 píxeles, manteniendo su altura. En el 100%, además de esos 500 píxeles, la altura variará hasta los 300 píxeles y cambiará su color.

```
@-webkit-keyframes mianimacion{

    0% {    }
    50% {width:500px; height:60px; }
    100% {  width:500px; height:300px; background-color:#222266; }
    }
```

Ahora, para que esa función se ejecute, la invocaremos desde la propiedad habitual, utilizando varias propiedades del tipo **animation-**.

```
div#bloque1:hover {

    -webkit-animation-name: "mianimacion";
    -webkit-animation-duration: 5s;
    -webkit-animation-interaction-count: infinite;
    -webkit-animation-timing-function: ease-in-out;
    }
```

La primera línea llama a la función, la segunda define su duración, la tercera indica cuántas veces se repetirá (podemos usar **infinite** para repeticiones infinitas) y la última línea indica cómo se realizará la transición (rápido al principio y lento al final, uniforme, etc).

Podemos emplear algunas características más, pero las fundamentales son **name**, **duration** e **interaction-count**.

Observe en el vídeo siguiente la diferencia de funcionamiento de esta animación. Es más completa y permite muchas más posibilidades.

Vídeo

Animación mediante CSS3

Nota

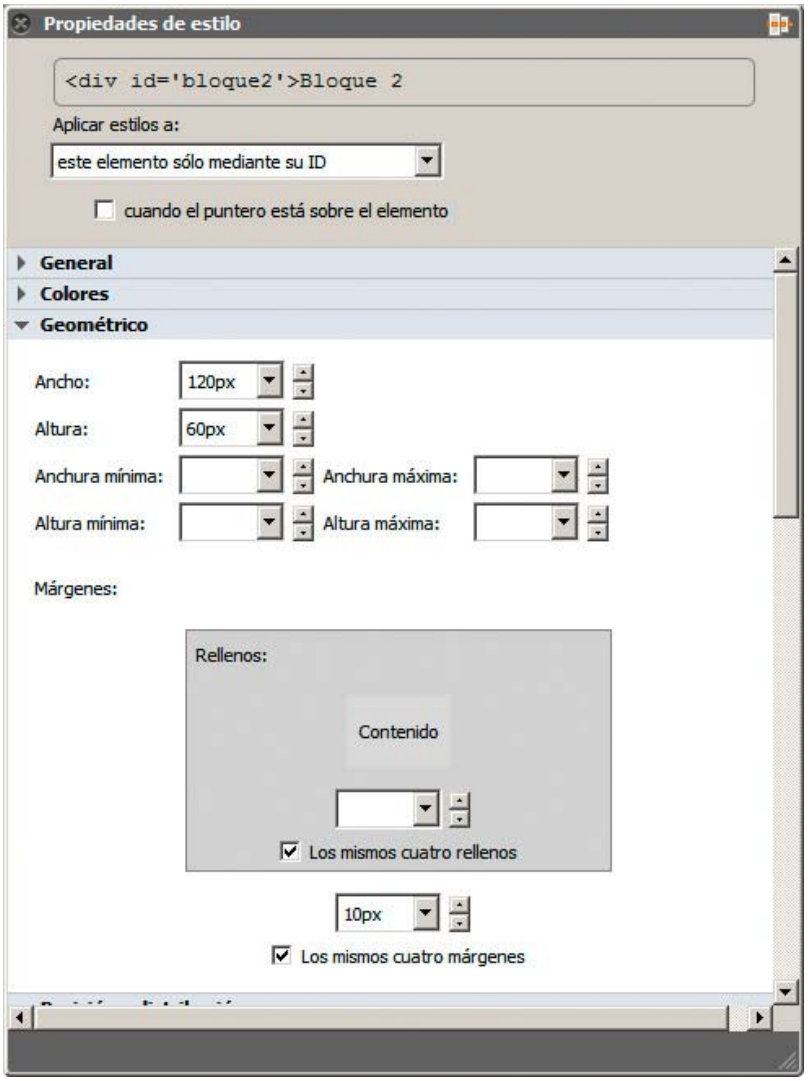
En todos los ejemplos de esta página hemos definido transiciones y animaciones exclusivamente empleando el prefijo **-webkit**, por lo que sólo funcionarán en navegadores tipo *Chrome, Chromium*, etc. Para que funcione en *Internet Explorer* o *Firefox* debemos recrear el mismo código con los prefijos necesarios.

Aplicar estilos con un editor web

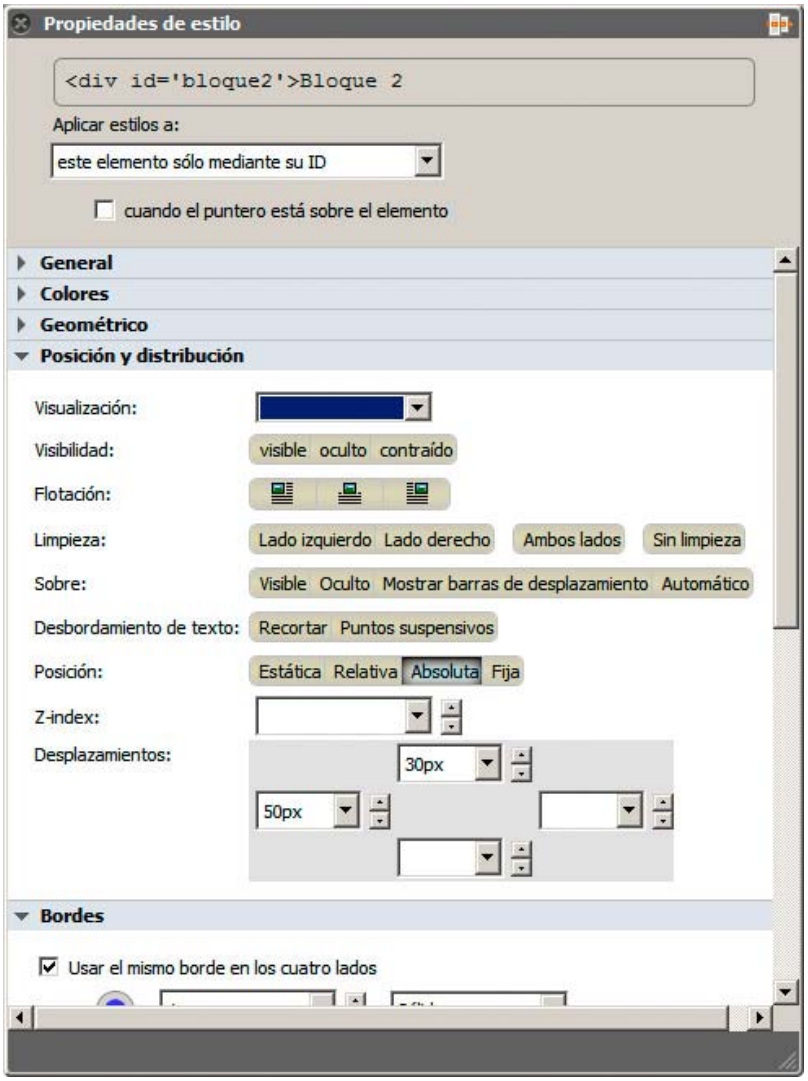
Como hemos podido ver, el número de propiedades crece y crece sin medida. Recordarlas todas, su sintaxis, sus valores y sus pequeños trucos no es sencillo. Por suerte contaremos con diferentes editores para ayudarnos en la tarea.

Con *BlueGriffon*, nuestro editor de cabecera, podemos aplicar todos los estilos que hemos visto en este módulo. Para hacerlo desplegaremos el panel **Propiedades de estilo**, seleccionaremos la opción más apropiada en el cuadro **Aplicar Estilos a** y nos desplazaremos por las diferentes secciones. Éstas son las que hemos visto en este módulo:

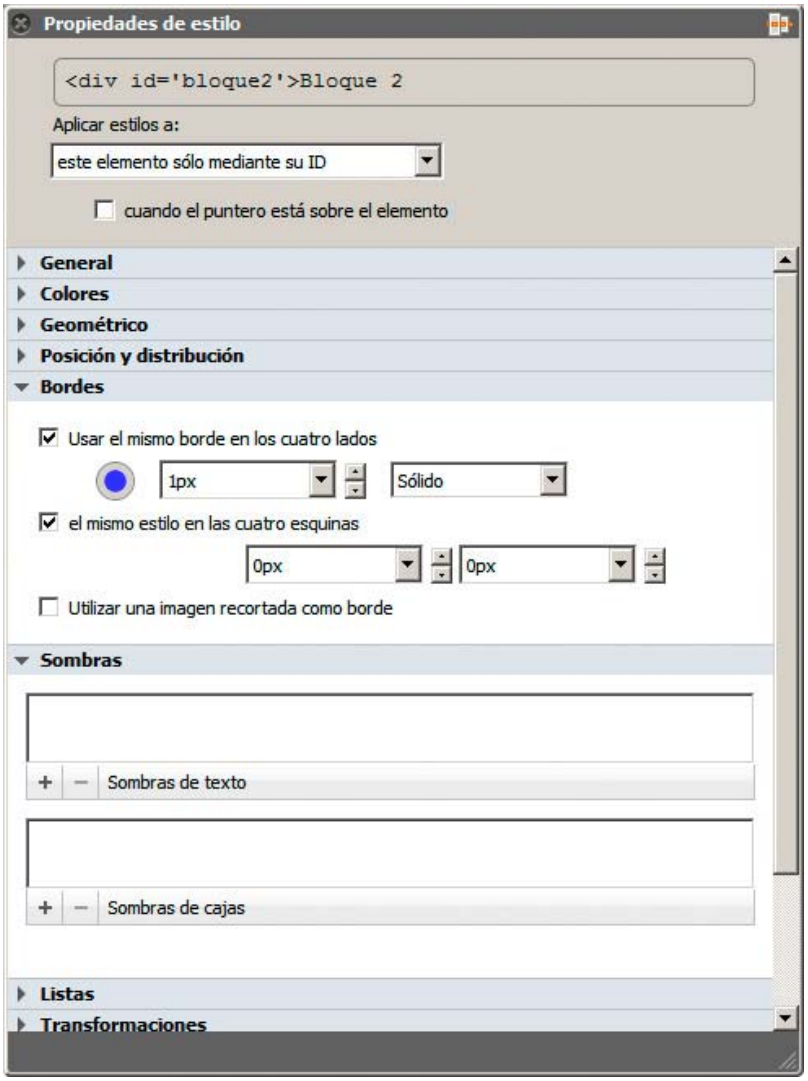
- **Geométrico**: contiene las opciones asociadas a los elementos básicos de una caja, como son sus márgenes, relleno, dimensiones, etc. De hecho, muestra incluso una pequeña representación del concepto de caja, como se recoge en la figura:



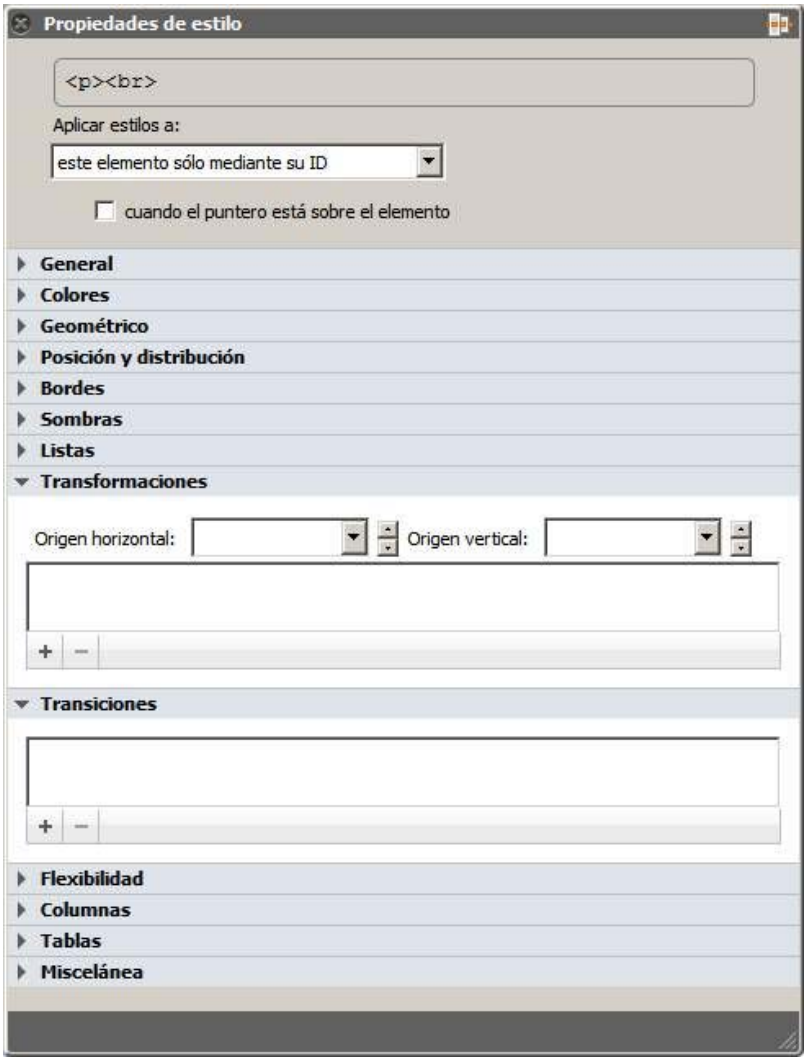
- **Posición y distribución**: incluye las propiedades de **display**, **visibility**, **float**, **clean**, **overflow**, **position**, **left** y **top**. Todas ellas van juntas en el espacio de la figura:



- **Bordes** y **Sombras**: nos servirán para aplicar estos elementos estéticos a nuestras cajas. En el caso de la sombra encontramos también la propiedad **text-shadow**, que vimos en el módulo dedicado al texto. La figura muestra ambas categorías de forma visual.



- **Transformaciones y Transiciones**: las secciones de la figura se emplean para añadir estos dos tipos de efectos. Estos dos grupos de propiedades son perfectas para aplicarlas con el editor, en lugar de hacerlo a mano.



La aplicación de transiciones puede ser un poco más compleja, ya que normalmente requiere la definición de un estado normal para una propiedad y la de un estado **:hover** (cuando el ratón está sobre el elemento) para esta misma propiedad. Para ello contamos con la casilla **cuando el puntero está sobre el elemento**, situada en la parte superior del panel **Propiedades de estilo**. El siguiente vídeo explica cómo se aplica.

Vídeo

Aplicar una transición con BlueGriffon

Pregunta Verdadero-Falso

La siguiente afirmación, ¿es verdadera o falsa?

Con *BlueGriffon* podemos aplicar los estilos CSS desde el panel **Propiedades** de estilo,pudiendo elegir entre estilo **Geométrico**; **Posición y Distribución**; **Bordes y Sombras**; y **Transformaciones y Transiciones.**

Verdadero ○ Falso ○

El resto de las categorías contienen propiedades menores que hemos ido viendo entremezcladas con las demás o que no

nos aportarán grandes mejoras en estas primeras fases de aprendizaje de las hojas de estilo.

Resumen

Con las propiedades analizadas en este capítulo obtenemos una gran libertad creativa para distribuir los espacios de nuestras páginas web, al tiempo que hemos profundizado en procedimientos muy útiles para manipular cualquier elemento insertado.

La forma de mostrar los objetos, su posicionamiento y cómo decorarlos adecuadamente son técnicas que deberíamos controlar con fluidez.

Actividades y ejemplos

Actividad 1. El modelo de caja

Tomando una página de ejemplo, modificaremos los valores de márgenes y de rellenos de diferentes elementos. Pensando en las etiquetas que tenemos en la página, intentaremos cambiar aquellas que pueda parecer que no tienen esta cualidad.

Actividad 2. El modelo de caja

Continuando con el ejemplo anterior, iremos modificando valores parciales retirando la propiedad general para reemplazarla por otras dos o tres que modifiquen sólo algunos laterales.

Actividad 3. El modelo de caja

Reemplazaremos todos los bordes que hayamos aplicado hasta ahora utilizando el modelo condensado.

Actividad 4. Sombras

Probaremos a aplicar bordes redondeados o sombras a diferentes elementos de nuestra página.

Actividad 5. Sombras

Intentaremos conseguir un efecto de brillo, procurando que el resplandor se mueva en espectros de la luz que le den una apariencia fria, con tonos blancos o azules.

Actividad 6. Anchura y altura de una caja

Hay mucho que experimentar aquí. Probaremos a crear dos bloques similares a los del ejemplo y comprobar lo que sucede, al indicar valores relativos como el 50% o valores fijos menores.

Actividad 7. Anchura y altura de una caja

Realizaremos algunas pruebas con los valores de mínimo y máximo, así como con la propiedad **overflow**.

Actividad 8. Visualización de elementos

El primer ejemplo es una buena práctica para familiarizarnos con estas técnicas y, aún más, probar a hacer el que muestra y oculta el bloque. Como ayuda, éste sería el código *JavaScript* completo de este último caso.

```
<script language="JavaScript">

    function mostrar() {
    document.getElementById('bloque2').style.display="inline-block";
    }
    function ocultar() {
    document.getElementById('bloque2').style.display="none";
    }

</script>
```

Actividad 9. Visualización de elementos

En una página web escribiremos esta secuencia

```
<p> No es lo mismo pero es <span id="palabra">bastante</span> parecido</p>
```

Utilizando la propiedad **display** y **visiblity** haremos desaparecer la palabra situada en la etiqueta ****. Debemos apreciar la diferencia entre ambos métodos.

Actividad 10. Posicionamiento

Diseñaremos una página web que tenga cuatro bloques diferenciados (los podemos hacer como ** **). El primero será la cabecera, el segundo y el tercero serán bloques centrales de contenidos y el cuarto lo emplearemos para hacer las veces de pie de página. Tanto el primero como el último deben ocupar todo el ancho de la página, mientras que los centrales deben repartirse el resto del espacio. Colorearemos cada ** ** con un tono diferente para destacar más las diferencias.

Actividad 11. Transformaciones y prefijos

Empleando transiciones diseñaremos una sombra casi imperceptible que se destaca fuertemente al pasar el ratón sobre el elemento al que se lo apliquemos, por ejemplo un ** ** o un **<h1>**. Todo lo que necesitamos hacer es crear una sombra casi transparente y definir una propiedad **:hover** para ese elemento en el que la sombra es casi opaca, añadiendo además las propiedades transition necesarias para que se produzca paulatinamente.

Es decir, los dos estilos que definiríamos podrían quedar así:

```
div#bloque1 {
    box-shadow: 8px 8px 6px #333333;
    }

div#bloque1:hover {
    box-shadow: 8px 8px 6px #333333;
    -webkit-transition-property: box-shadow;
    -webkit-transition-duration: 5s;
    -webkit-transition-delay: 1s;
    }
```

Probaremos también a variar el número de segundos hasta encontrar un equilibrio entre estética y usabilidad. El efecto no debe ser pesado ni entorpecer el trabajo del usuario, pero al mismo tiempo debe percibirse.

Ejemplos

Las diferentes prácticas, recursos y ejemplos realizadas en este módulo están disponibles para realizar pruebas.

Ejemplos del módulo

Aplicación al aula

Maquetación con estilos

El alumnado definirá una página web al estilo de un periódico o una publicación, preparando diferentes espacios mediante el uso de estilos.

Programación dirigida al alumnado

Objetivos

- Aplicar conceptos de posicionamiento con hojas de estilo en cascada.
- Modificar una hoja de estilos externa.
- Transformar un documento mediante los estilos.

Contenidos

- Hojas de estilo.
- Estilos de posicionamiento y distribución.
- Creación de estilos con un editor web.

Materiales y recursos

- Ordenador con acceso a Internet.

Temporalización

- Dos sesiones.

Planificación

El alumnado realiza una práctica de maquetación probando e investigando las posibilidades de las hojas de estilo en este aspecto.

Organización del aula

Trabajaremos en un aula con ordenadores con un agrupamiento individual o por parejas.

Desarrollo de la actividad

- Se repasan los conceptos de maquetación con hojas de estilos en cascada.
- Se diseña un modelo de página con diferentes bloques.
- Se genera una hoja de estilos externa que modificará la ubicación de los bloques.
- Se comparten los resultados.

Presentación y evaluación de los resultados

La evaluación se realizaría mediante la revisión del resultado y la observación del proceso. Se pueden evaluar varios aspectos a lo largo de todo el proceso:

- Capacidad para aplicar los estilos.
- Inserción apropiada de los estilos en los lugares indicados.
- Apariencia de los cambios aplicados en el documento.

Sugerencias metodológicas

La metodología empleada es la de **proyecto**.

Para su aplicación proponemos:

Primera sesión

Explicamos el objetivo de la actividad, describimos los conceptos necesarios.

Describimos los bloques que se deben crear y se presentan algunos ejemplos de resultado final.

Generaremos una hoja de estilos externa.

Segunda sesión

Retomando el ejemplo anterior, aplicamos los estilos necesarios.

Revisamos los primeros resultados y proponemos alternativas.

Compartimos los resultados entre los alumnos. Ponemos en común cuáles son las maquetaciones más apropiadas.

Atención a la diversidad

Actividad de refuerzo

Para aquellos alumnos/as que puedan tener más dificultad, se les entrega el documento con los bloques prediseñados.

Actividad de ampliación

La profundización en esta actividad se basaría en que el alumnado probase algunas propiedades más complejas, transformaciones, etc.