Introducción a CSS3 (Parte 1)

1. LA EVOLUCIÓN QUE REPRESENTA CSS3

Si HTML5 es la evolución natural del lenguaje de marcado de la Web para producir documentos más semánticos y ricos; las hojas de estilo en cascada Nivel 3 (**CSS3**) son la evolución de este otro estándar web.

A diferencia de lo que ocurre con las versiones anteriores de las hojas de estilo en cascada, en CSS3 se han dividido sus características en distintos módulos.

De esta forma es más fácil que unas determinadas características evolucionen rápidamente y otras no tanto.

CSS3 incorpora novedades desde distintos puntos de vista, pero sobre todo representa una evolución de para qué se pueden utilizar las hojas de estilo en cascada.

Hasta ahora hemos utilizado las hojas de estilo en cascada como la herramienta para aplicar formato al contenido de la página web o para conseguir la distribución que deseábamos obtener.

Es verdad que podíamos conseguir algún que otro efecto interesante, pero no era lo fuerte de esta tecnología web.

Con CSS3 se amplían las posibilidades, ya que podremos conseguir efectos dinámicos que hasta ahora solo se conseguían con otras tecnologías, como JavaScript o Flash.

CSS3 admite **gradientes**, **transiciones**, **animaciones** e incluso **transformaciones** (desplazar un objeto, cambiar su tamaño, rotarlo, etc.), características que hasta ahora no se podían aplicar con un lenguaje creado para el diseñador y no para el programador de aplicaciones web.

Con **HTML5** y **CSS3** el diseñador tiene a su alcance herramientas mucho más potentes para crear las aplicaciones web que se demandan en la actualidad, como representa todo lo relacionado con la **Web 2.0**.

Pero CSS3 sigue siendo una herramienta para el diseñador y como tal se utiliza en la capa de presentación de una página web.

La diferencia es que ahora no solo nos va a permitir modificar el aspecto o disposición de esa página web, sino que también nos permitirá mejorar la interacción o experiencia que el usuario tiene al visitarla.

A continuación veremos algunas de estas novedades tan interesantes que aporta CSS3.

Como ocurre con el lenguaje HTML5, la especificación de CSS3 sigue evolucionando, por lo que no es una especificación finalizada.

Esto quiere decir que nos vamos a encontrar con que algunas de estas novedades no han sido implementadas por todos los navegadores, ni siquiera por los más modernos; o que han sido implementadas de forma particular por alguno de ellos.

Esto no te debe preocupar o hacerte pensar que todavía no es el momento de utilizar CSS3.

Lo importante es aplicar las novedades CSS3 en el **nivel de presentación** de la página web. De esta forma, si el usuario utiliza un navegador que no es compatible con alguna de estas características, seguirá pudiendo acceder al contenido de la página web, aunque obtendrá una experiencia inferior.

Nos centraremos en esas novedades de **CSS3** que ya podemos utilizar.

¡Empecemos!

2. NUEVOS SELECTORES CSS3

CSS3 proporciona una serie de nuevos **selectores avanzados** que nos permiten seleccionar elementos en función de su lugar en el árbol del documento que hasta ahora no era posible establecer.

Recordemos los selectores que se pueden utilizar en CSS 2.1:

- Los que utilizan los nombres de las etiquetas HTML. Por ejemplo: **h1**

- Los que utilizan identificadores. Por ejemplo: **#btn_enviar**

- Los que utilizan nombres de clases. Por ejemplo: **.email**

- El selector universal (*****)

- Selector de atributos. Por ejemplo: **img[alt]**

- Selector de hijos. Por ejemplo: **body > div**

- Selector de descendientes. Por ejemplo: **body div**

- Selector de adyacentes. Por ejemplo: **h1 + a**

Por ejemplo, es muy habitual disponer el texto de un documento en varios párrafos y aplicar una separación determinada entre ellos.

Una forma lógica de conseguirlo sería simplemente aplicar una clase con la propiedad **padding-bottom** para establecer dicha separación. Esta clase se aplicaría en los párrafos del texto.

Bien, el problema de esta solución es que el último párrafo también tendrá dicha separación o *padding* inferior, lo que seguramente no deseamos.

Hasta ahora, para resolver esta situación se aplicaba otra regla al último párrafo. Sin embargo, con los nuevos selectores CSS3 es posible determinar cuál es ese último párrafo y aplicarle el formato deseado únicamente a él. Para ello, se utilizaría el selector **last-child** (último hijo).

Como hay un buen número de selectores nuevos, vamos a ver primero un ejemplo de su aplicación y después dispondrás del listado completo.

En la página web siguiente tenemos una pequeña tabla. Vamos a aplicar el famoso *"estilo cebra"* en esta tabla, de forma que las filas pares tengan un determinado color de fondo y las impares otro color distinto.

ALGUNAS PROPIEDADES NUEVAS CSS3	IE	Firefox	Chrome	Safari	Opera
border-radius	Soportado	moz-	webkit-	webkit-	Soportado
box-shadow	Soportado	moz-	webkit-	webkit-	Soportado
text-shadow	No soportado	Soportado	Soportado	Soportado	Soportado
opacity	Soportado	Soportado	Soportado	Soportado	Soportado
gradient	No soportado	moz-	webkit-	webkit-	No soportado
Múltiples background-image	Soportado	Soportado	Soportado	Soportado	Soportado
% Compatibilidad	66%	100%	100%	100%	83%

Los selectores que empiezan con : se conocen como pseudoclases.

Podemos hacer esto mediante clases CSS, pero ahora lo conseguiremos mediante algún selector nuevo CSS3, por lo que no será necesario cambiar el código HTML de la tabla.

La idea es aplicar un determinado color de fondo a las filas pares e impares. Por lo tanto, primero tendremos que seleccionar dichas filas.

Podemos hacerlo con el selector **:nth-of-type(n)**.

Este selector o pseudoclase selecciona el elemento indicado pero con la condición de que sea el enésimo elemento hermano de ese tipo. Fíjate en la siguiente regla CSS:

```
tr:nth-of-type(even) {
    background-color: #e4ffca;
}
```

El selector utilizado está seleccionando las filas pares (**even**).

Bueno, en este caso son filas de una tabla, pero este selector sirve para seleccionar los elementos pares de una secuencia de elementos. Por ejemplo, serviría igualmente para seleccionar los elementos pares de una lista ordenada o sin ordenar, o de un conjunto de párrafos.

Si queremos seleccionar los elementos impares, en lugar de even, utilizaremos **odd**:

```
tr:nth-of-type(odd) {
    background-color: #ffffff;
}
```

Vemos, por lo tanto, que ya disponemos de una forma para seleccionar elementos pares o impares sin necesidad de modificar para nada el código HTML. Así pues, no se ha necesitado añadir ninguna clase a cada una de las etiquetas **tr** en dicho código.

En este caso, el selector **:nth-of-type** utiliza una palabra clave, es decir, **even** para los elementos pares y **odd** para los impares.

También se puede utilizar una fórmula en su lugar. Por ejemplo, con **tr:nth-of-type(2)** estaríamos seleccionando la segunda fila de la tabla.

Con las dos reglas anteriores, obtenemos este resultado:

ALGUNAS PROPIEDADES NUEVAS CSS3	IE	Firefox	Chrome	Safari	Opera
border-radius	Soportado	moz-	webkit-	webkit-	Soportado
box-shadow	Soportado	moz-	webkit-	webkit-	Soportado
text-shadow	No soportado	Soportado	Soportado	Soportado	Soportado
opacity	Soportado	Soportado	Soportado	Soportado	Soportado
gradient	No soportado	moz-	webkit-	webkit-	No soportado
Múltiples background-image	Soportado	Soportado	Soportado	Soportado	Soportado
% Compatibilidad	66%	100%	100%	100%	83%

Fíjate que las filas pares muestran el color verde y las impares el color blanco, tal como se lo hemos indicado en la hoja de estilo.

Además, vemos que el texto queda centrado en las distintas celdas, gracias a otra regla CSS que se está aplicando.

Imagínate que no quieres que ocurra esto en las celdas de la primera columna, sino que quieres que queden alineadas a la derecha.

Como la regla para centrar el texto se aplica a la etiqueta **<table>**, tendrías que establecer otra regla para estas celdas de la primera columna...

No hace falta hacerlo, en lugar de eso podemos utilizar otro selector CSS3:

```
th:first-child {
    text-align: right;
}
```

Con la pseudoclase **:first-child** indicamos que queremos seleccionar los elementos **th** pero con la condición de que sean, respectivamente, los primeros de su elemento padre.

ALGUNAS PROPIEDADES NUEVAS CSS3	IE	Firefox	Chrome	Safari	Opera
border-radius	Soportado	moz-	webkit-	webkit-	Soportado
box-shadow	Soportado	moz-	webkit-	webkit-	Soportado
text-shadow	No soportado	Soportado	Soportado	Soportado	Soportado
opacity	Soportado	Soportado	Soportado	Soportado	Soportado
gradient	No soportado	moz-	webkit-	webkit-	No soportado
Múltiples background-image	Soportado	Soportado	Soportado	Soportado	Soportado
% Compatibilidad	66%	100%	100%	100%	83%

Fíjate que ahora las celdas **th** de la primera columna tienen el texto alineado a la derecha.

Finalmente, si queremos que en la última fila (ya que parece que es una fila de totales) se aplique un formato distinto, podríamos utilizar la pseudoclase **:last-child**:

```
tr:last-child {
    font-size: 20px;
    font-style: italic;
    color: gray;
}
```

ALGUNAS PROPIEDADES NUEVAS CSS3	IE	Firefox	Chrome	Safari	Opera
border-radius	Soportado	moz-	webkit-	webkit-	Soportado
box-shadow	Soportado	moz-	webkit-	webkit-	Soportado
text-shadow	No soportado	Soportado	Soportado	Soportado	Soportado
opacity	Soportado	Soportado	Soportado	Soportado	Soportado
gradient	No soportado	moz-	webkit-	webkit-	No soportado
Múltiples background-image	Soportado	Soportado	Soportado	Soportado	Soportado
% Compatibilidad	*66%*	*100%*	*100%*	*100%*	*83%*

Puedes ver que se ha aplicado el formato exclusivamente a la última fila de la tabla. Podrías seleccionar únicamente la última celda de esta última columna con este selector: **`tr:last-child td:last-child`**

CSS3 introduce todo un nuevo conjunto de selectores que deberás estudiar detenidamente. Los más interesantes son de este tipo, que dan acceso a la selección de elementos en función de su posición en el árbol del documento.

Listado de los nuevos selectores CSS3:

1 - Nuevos elementos de atributo:

- **elemento[atributo^="valor"]**: selecciona todos los *elementos* que tienen el *atributo* y cuyo *valor* comienza exactamente por la cadena de texto indicada.
- **elemento[atributo$="valor"]**: selecciona todos los **elementos** que tienen el *atributo* y cuyo *valor* termina exactamente por la cadena de texto indicada.
- **elemento[atributo*="valor"]**: selecciona todos los *elementos* que tienen el *atributo* y cuyo *valor* contiene la cadena de texto indicada.

Por ejemplo, para seleccionar todos los enlaces que apunten a una dirección de correo electrónico, podríamos utilizar: `a[href^="mailto:"]`

2 - Selector general de hermanos:

- elemento1 **~ elemento2**: selecciona cualquier *elemento2* que es hermano de *elemento1* en el árbol del documento y que aparece detrás en el código HTML aunque no necesariamente de forma inmediata.

3 - Nuevos pseudo-elementos: cambian la sintaxis y ahora se utiliza :: en lugar de :

- **::first-line** selecciona la primera línea del texto de un elemento.
- **::first-letter** selecciona la primera letra del texto de un elemento.
- **::before** selecciona la parte anterior al contenido de un elemento para insertar nuevo contenido.
- **::after** selecciona la parte posterior al contenido de un elemento para insertar nuevo contenido.
- **::selecion** selecciona el texto que ha seleccionado un usuario con su ratón o teclado.

4 - Nuevas pseudoclases:

- **elemento:nth-child(numero)**: selecciona el *elemento* indicado pero con la condición de que sea el hijo enésimo de su padre. Por ejemplo, para seleccionar un determinado párrafo o elemento de una lista.
- **elemento:nth-last-child(numero)**: igual que el anterior pero el número indicado se empieza a contar desde el último hijo.

- **elemento:empty**: selecciona el *elemento* indicado pero con la condición de que no tenga ningún hijo (ni siquiera nodos de texto).
- **elemento:first-child** y **elemento:last-child**: seleccionan los *elementos* indicados con la condición de que sean los primeros o últimos hijos de su elemento padre, respectivamente.
- **elemento:nth-of-type(numero)**: selecciona el elemento indicado con la condición de que sea el enésimo elemento hermano de ese tipo.
- **elemento:nth-last-of-type(numero)**: igual que el anterior pero el número indicado se empieza a contar desde el último hijo.
- **:not()**: para seleccionar los elementos que no cumplen con una condición. Por ejemplo, para seleccionar todos los elementos de un documento que no sean enlaces, se podría utilizar: **not(a)**

3. ESQUINAS REDONDEADAS Y SOMBRAS

Si preguntas a un diseñador web experimentado qué característica le gustaría disponer en CSS, te aseguro que más de uno contestaría poder crear fácilmente "**esquinas redondeadas**" de los elementos.

Hasta ahora los diseñadores cuando querían que una figura o bloque no tuviera las esquinas rectas tenían que utilizar imágenes de fondo para simularlo y muchas veces eso es difícil de conseguir.

CSS3 incorpora la propiedad **border-radius** que facilita esta tarea tremendamente.

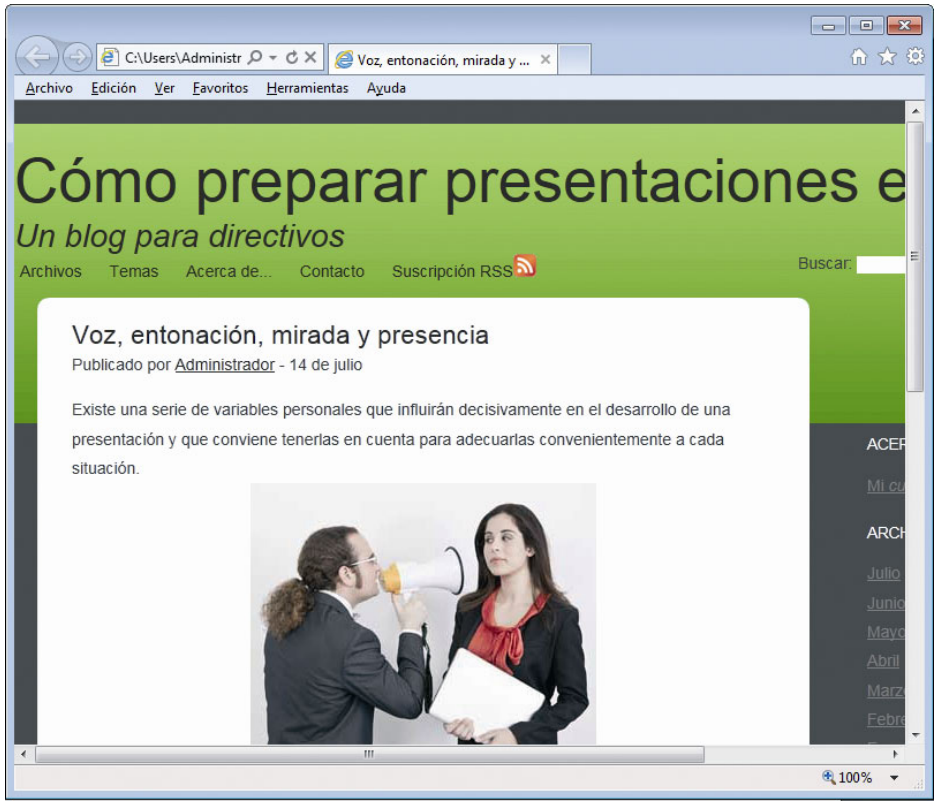
En la figura de la página siguiente puedes ver que el bloque blanco con el texto principal presenta las esquinas redondeadas.

Las esquinas rectas suelen provocar un aspecto más "rudo" que las esquinas redondeadas.

Este efecto se utiliza muchísimo ya que permite "encajar" los elementos de forma suave en el diseño de la página web.

Fíjate en el uso de esta propiedad:

```
#content {
     margin-left: 20px;
     float: left;
     width: 610px;
     padding: 20px 30px 20px 30px;
     background-color: #ffffff;
     border-radius: 10px;
}
```



Vemos que simplemente se indica el valor de la curvatura. En este caso, el valor **10 píxeles** representa el radio de un círculo imaginario que se utilizaría para redondear el borde del elemento. Cuanto mayor es este valor, más redondeada quedará la esquina.

Sin embargo, esta propiedad no está soportada por todos los navegadores. En la tabla siguiente esto queda reflejado junto a otras propiedades nuevas en CSS3.

ALGUNAS PROPIEDADES NUEVAS CSS3	IE	Firefox	Chrome	Safari	Opera
border-radius	Soportado	-moz-	-webkit-	-webkit-	Soportado
box-shadow	Soportado	-moz-	-webkit-	-webkit-	Soportado
text-shadow	No soportado	Soportado	Soportado	Soportado	Soportado
opacity	Soportado	Soportado	Soportado	Soportado	Soportado
gradient	No soportado	-moz-	-webkit-	-webkit-	No soportado
Múltiples background-image	Soportado	Soportado	Soportado	Soportado	Soportado

Como puedes ver, Internet Explorer y Opera soportan esta propiedad, mientras que los otros tres navegadores lo hacen pero de una forma especial.

Algunos navegadores implementan las características nuevas utilizando un prefijo delante del nombre. Esto lo hacen para probar la característica antes de implementarla tal como aparece en la especificación.

En esta tabla puedes ver que, para esta propiedad, **Firefox** utiliza el prefijo **-moz-**, mientras que **Chrome** y **Safari** utilizan el prefijo **-webkit-**.

Esto quiere decir que para que Firefox muestre el efecto de la propiedad border-radius, realmente tenemos que utilizar **-moz-border-radius-**; y lo mismo con **-webkit-border-radius-** para los otros dos navegadores.

Estos son los prefijos que utilizan los navegadores para sus propiedades únicas:

Internet Explorer **-ms-**

Firefox **-moz-**

Chrome **-chrome-** y **-webkit-**

Safari **-webkit-**

Opera **-o-**

Por lo tanto, la regla quedaría así:

```
#content {
     margin-left: 20px;
     float: left;
     width: 610px;
     padding: 20px 30px 20px 30px;
     background-color: #ffffff;
     -moz-border-radius: 10px;
     -webkit-border-top-left-radius: 10px;
     -webkit-border-top-right-radius: 10px;
     -webkit-border-bottom-left-radius: 10px;
     -webkit-border-bottom-right-radius: 10px;
     border-radius: 10px;
}
```

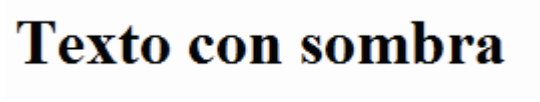
Es importante escribir en la última posición el nombre de la propiedad "estándar", es decir, la que aparece en la especificación sin el prefijo. En este caso es **border-radius**.

Así, cuando los navegadores que utilizan prefijos especiales implementen dicha propiedad tal como aparece en la especificación, esta es la que finalmente se aplicará.

Además, para el caso de **-webkit-border-radius-** tenemos que utilizar las propiedades detallando los cuatro bordes. Esto puede hacerse en cualquier caso si se desea que las esquinas tengan un radio distinto.

Por su parte, las propiedades **text-shadow** y **box-shadow** permiten aplicar sombras en el texto y en la caja delimitadora de un elemento de bloque, respectivamente. Veamos un ejemplo.

Este es el aspecto del texto inicial:



Le aplicamos ahora esta regla:

```
h1 {
   text-shadow: #000000 0px 0px 5px;
}
```

Y conseguimos este resultado:

Los valores que tenemos que establecer para la propiedad **text-shadow** son:

- El **color** de la sombra.
- **Desplazamiento horizontal** de la sombra respecto del texto. Si es un valor positivo, el sentido de la sombra es hacia la derecha; si es negativo, el sentido es hacia la izquierda.
- **Desplazamiento vertical** de la sombra respecto del texto. Si es un valor positivo, el sentido de la sombra es hacia abajo; si es negativo, es hacia arriba.
- Valor de **difuminado**.

Tal como hemos establecido la propiedad tenemos que el color de la sombra es negro, con un desplazamiento nulo respecto del texto y con 5 píxeles de difuminado.

Cambiando algunos de los parámetros:

```
h1 {
   text-shadow: #ac6af0 20px 10px 1px;
}
```

Obtenemos este otro resultado:

Fíjate que ahora sí que se aplica tanto desplazamiento horizontal como vertical, mientras que el difuminado es menor. Además, se utiliza un color distinto al del texto.

Es posible aplicar más de una sombra a la vez. Para ello, simplemente tienes que separar la definición de cada sombra con una coma. Por ejemplo:

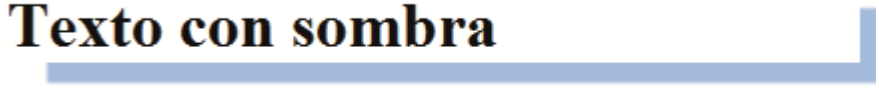
```
text-shadow: #a6caf0 20px 10px 1px, #a6caf0 -20px -10px 1px;
```

La propiedad **box-shadow** funciona de forma similar, pero respecto de la caja delimitadora del elemento de bloque. Por lo tanto, se puede aplicar a cualquier elemento de bloque, no solo al texto.

Los valores que tenemos que establecer para **box-shadow** son los mismos, pero en este caso es necesario utilizar los prefijos particulares de algunos navegadores, ya que, a diferencia de **text-shadow**, esta propiedad no se implementa de la misma forma en todos ellos.

Fíjate cómo quedaría el código y el resultado obtenido:

```
h1 {
   -moz-box-shadow: #a6caf0 20px 10px 1px;
   -webkit-box-shadow: #a6caf0 20px 10px 1px;
   box-shadow: #a6caf0 20px 10px 1px;
}
```



Los efectos de sombra pueden producir resultados sorprendentes. Además, se utiliza mucho **text-shadow** con los hipervínculos.

Así, aplicando sombra al texto del hipervínculo en el estado **hover** podemos conseguir un efecto muy interesante.

4. COLORES

CSS3 introduce algunas novedades desde el punto de vista de la selección o especificación de los **colores**.

Sabemos que los monitores y dispositivos que emplean una pantalla utilizan el modelo de color **RGB**, consiguiendo los colores a partir de tres componentes o colores básicos: rojo (Red), verde (Green) y azul (Blue).

Estos colores **RGB** son representados en notación hexadecimal. Por ejemplo, el color **#ff0000** es el rojo puro.

Aquí tenemos un ejemplo en el que veremos las distintas formas que tenemos en CSS para especificar un color.

Archivo **colores.html**:

```
<!DOCTYPE HTML>
<html lang="es">
<head>
<meta http-equiv="Content-Type" content="text/html;
 charset=utf-8">
<link href="css/orange.css" rel="stylesheet"
type="text/css">
<title>Colores</title>
</head>

<body>
<div id="container">
<div id="bloque1" class="bloque"></div>
<div id="bloque2" class="bloque"></div>
<div id="bloque3" class="bloque"></div>
<div id="bloque4" class="bloque"></div>
</div>
</body>
</html>
```

El contenido de la página simplemente es un elemento div contenedor (**container**) en el que aparecen otros elementos div (**bloque1**, **bloque2**, **bloque3** y **bloque4**) para los que hemos establecido un color de fondo y de borde mediante una hoja de estilo en cascada.

Archivo **orange.css**:

```
@charset "utf-8";
/* Reglas generales */

#container {
     width: 400px;
     margin: 0 auto;
     overflow: hidden;
}
```

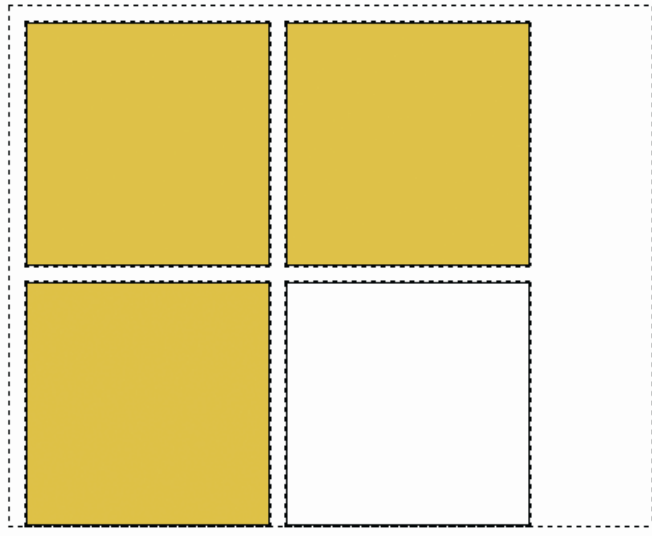
```
.bloque {
     width: 150px;
     height: 150px;
     margin-left: 10px;
     margin-top: 10px;
     float: left;
     border: 1px solid #000;
}

#bloque1 {
     background-color: #f0c425;
}

#bloque2 {
     background-color: rgb(240, 196, 37);
}

#bloque3 {
     background-color: hsl(47, 87%, 54%)
}
```

Y este es el resultado. Por ahora los tres primeros bloques muestran el mismo color anaranjado.



Para el **bloque1** hemos especificado el color con la notación conocida, es decir, el carácter # seguido de los tres valores en hexadecimal de los componentes rojo, verde y azul.

Si el valor hexadecimal de un componente de color es tal que se representa con dos letras o números iguales, entonces es suficiente utilizar un carácter. Por ejemplo, en lugar de **#ff66aa** podríamos utilizar **#f6a**.

Para el **bloque2** el color se consigue utilizando el operador **rgb**. En este caso se utilizan los valores, en decimal, de los tres colores básicos.

La conversión entre valores hexadecimales y decimales se puede realizar con cualquier aplicación de diseño, incluido **Dreamweaver**, o simplemente con una calculadora.

Además, en CSS3 podemos utilizar otro modelo, conocido como **HSL**: matiz (Hue), saturación (Saturation) y luminosidad (Lightness).

Puedes verlo aplicado en la regla para el **bloque3**. Se utiliza el operador **hsl** con estos valores:

Matiz: **47**

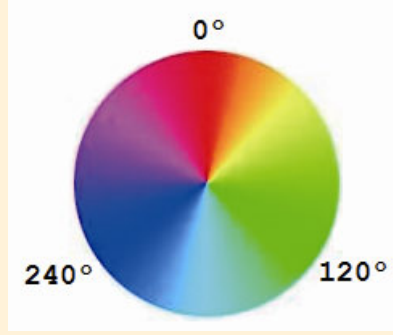
Saturación: **87%**

Luminosidad: **54%**

El color base queda reflejado por el valor del componente matiz, mientras que los otros dos lo modifican. Por eso matiz puede tomar un valor entre **0** y **360** grados, que representa la rueda de colores básicos; mientras que saturación y luminosidad son porcentajes del **0** al **100**%.

La rueda de colores **HSL** se dispone de forma que:

- Rojo: 0º
- Amarillo: 60º
- Verde: 120º
- Cyan: 180º
- Azul: 240º
- Magenta: 300º



Para obtener un color puro en el modelo HSL:

- Elige un valor del **0** a **360** para el matiz.
- Elige el valor **100%** para la saturación.
- Elige el valor **50%** para la luminosidad.

¿Por qué aparece el modelo **HSL**? ¿No es suficiente con **RGB**?

El problema con el modelo **RGB** es que no es fácil adivinar los valores de cada componente de color.

Así pues, para especificar un determinado color en **RGB** no tenemos más remedio que acudir a alguna aplicación para seleccionarlo de algún conjunto de muestras o ir probando con los distintos valores hasta conseguirlo.

Sin embargo, en el modelo **HSL**, esto es más intuitivo, ya que el primer componente (matiz) es el que determina la mayor parte de ese color. Los otros dos componentes lo modifican.

Por ejemplo, si deseas obtener el color anaranjado que hemos utilizado, es lógico pensar que partirás de un color **HSL** cuyo primer componente debe estar entre el **rojo** y el **amarillo**. Es decir, entre un valor de **0** y **60** grados. Es difícil decir lo mismo si utilizas el modelo **RGB**.

Además, para modificar un color **HSL**, por ejemplo para hacerlo más claro, podemos jugar con la saturación y la luminosidad, sin tocar el valor del matiz.

Esto es muy habitual hacerlo. Por ejemplo, si tenemos un color predominante en una página web, podríamos utilizar una modificación de dicho color para los enlaces o títulos del texto.

Con **HSL** esto es más sencillo de conseguir modificando únicamente la saturación y/o luminosidad:

- A medida que se reduce la saturación, se consigue un color más gris.
- A medida que se aumenta la luminosidad, el color se acerca al blanco; y si la disminuimos, se acerca al negro.

Por eso los diseñadores web suelen preferir trabajar con **HSL** en lugar de **RGB**, ya que pueden modificar los colores de forma más intuitiva.

Como puedes ver, estamos consiguiendo colores distintos a partir de un mismo color o matiz. Esto es muy útil a la hora de crear combinaciones de colores en una página web.

Finalmente, tanto el operador **rgb** como el operador **hsl** aceptan un componente de **transparencia**. Para ello, se utiliza **rgba** y **hsla**.

En la figura de la página siguiente puedes ver en la parte izquierda que el color blanco del artículo o post es opaco respecto del fondo de la página; en la parte derecha se muestra el mismo color con cierta transparencia.

Para ello, se ha utilizado el operador **rgba**:

```
#content {
     margin-left: 20px;
     float: left;
     width: 610px;
     padding: 20px 30px 20px 30px;
     background-color: rgba(255, 255, 255, 0.7);
     -moz-border-radius: 10px;
     -webkit-border-top-left-radius: 10px;
     -webkit-border-top-right-radius: 10px;
     -webkit-border-bottom-left-radius: 10px;
     -webkit-border-bottom-right-radius: 10px;
     border-radius: 10px;
}
```

Aquí el cuarto componente es el nivel de transparencia. Un valor de **0.0** representa el color completamente transparente; mientras que **1.0**, completamente opaco.

También está disponible en CSS3 la propiedad **opacity**, que define cómo de opaco es un elemento. El valor **1** significa completamente opaco; mientras que **0**, completamente transparente.

Por ejemplo:

```
.prueba {
   opacity: 0.5;
}
```

Exactamente igual podríamos haber utilizado el operador **hsla** si especificamos colores **HSL**. El último valor también puede variar entre **0.0** y **1.0**.

5. GRADIENTES DE COLOR

CSS3 permite utilizar **gradientes** de color como una forma distinta de establecer una imagen. Esto es importante, ya que un gradiente no se entiende como un color, sino como una **imagen**.

Por lo tanto, podremos utilizar los gradientes allí donde es posible especificar una imagen. Por ejemplo, con las propiedades **background-image**, **list-style-image** y **border-image** (esta última es nueva en CSS3).

Las malas noticias a la hora de utilizar gradientes CSS3 es que todavía no existe una sintaxis estandarizada, aunque actualmente se está trabajando en ello.

En su lugar, tanto **Webkit** como **Mozilla** implementan los gradientes a su manera, por lo que tendremos que conocer la sintaxis particular de cada caso. Internet Explorer es compatible con los gradientes de color desde la versión 10 con la propiedad linear-gradient.

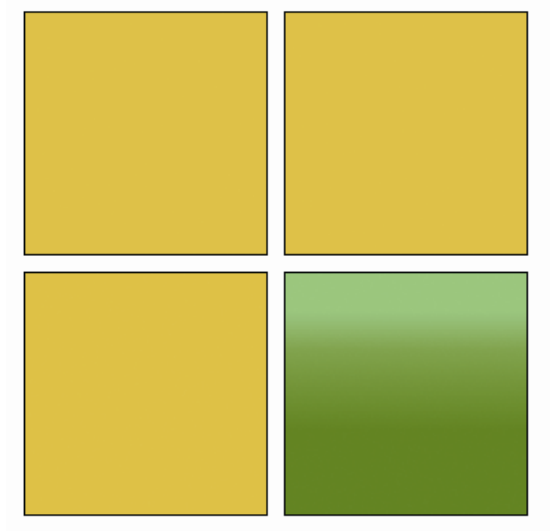
Podemos utilizar tanto gradientes lineales como radiales, de forma similar a lo que se puede hacer con el elemento **canvas**.

Fíjate en el aspecto que tiene la definición de un gradiente en CSS3:

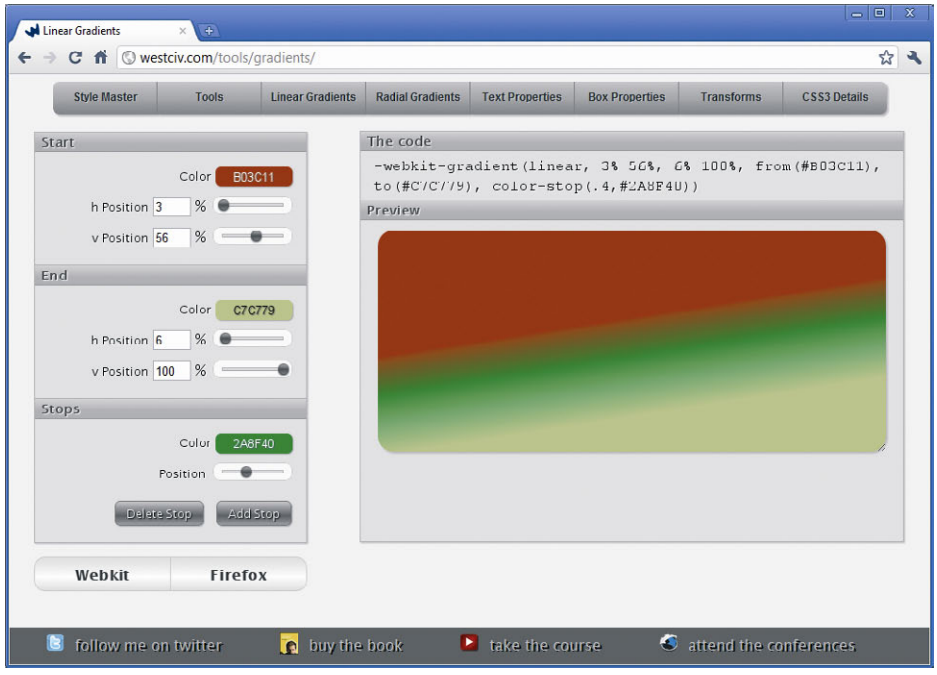
```
#bloque4 {
    background-image: -webkit-gradient(
    linear,
    left bottom,
    left top,
    color-stop(0.35, rgb(103,139,41)),
    color-stop(0.68, rgb(134,167,70)),
    color-stop(0.84, rgb(161,201,98))
    );
    background-image: -moz-linear-gradient(
    center bottom,
    rgb(103,139,41) 35%,
    rgb(134,167,70) 68%,
    rgb(161,201,98) 84%
    );
}
```

Utilizamos el gradiente para establecer la propiedad **background-image**. Hay dos reglas para ello: una utilizando la sintaxis para los navegadores Webkit y otra para Firefox o Mozilla.

En la figura siguiente puedes ver que el cuarto cuadrado tiene como imagen de fondo un determinado gradiente de color.



Como no se hace fácil recordar la sintaxis necesaria para utilizar estos gradientes, sobre todo hasta que dicha sintaxis no esté estandarizada, lo mejor es acudir a un generador de gradientes. Puedes encontrar uno aquí: *http://westciv.com/tools/gradients*



En este caso indicaríamos las características del gradiente y obtendríamos el código para Webkit o para Mozilla.

Recuerda que los navegadores Internet Explorer y Opera no disponen de una sintaxis personalizada, ya que se supone que admitirán la estandarizada en la especificación CSS3.

Como se suele decir, eso es *"trabajo en curso"*.