placeholder o required:

```
if (!attrSupports('input', 'autofocus')) {
   document.getElementById('search_string').focus();
}
if (!attrSupports('input', 'placeholder')) {
   // Atributo placeholder no soportado
}
if (!attrSupports('input', 'required')) {
   // Atributo required no soportado
}
```

2. CSS3

La especificación de CSS3 viene con interesantes novedades que permitirán hacer webs más elaboradas y más dinámicas, con mayor separación entre estilos y contenidos. Dará soporte a muchas necesidades de las webs actuales, sin tener que recurrir a trucos de diseñadores o lenguajes de programación.

Aunque CSS3 está todavía en fase de desarrollo, la mayoría de navegadores ya dan soporte a casi todos los nuevos estilos, como Firefox, Chrome o Safari. Por el contrario Internet Explorer no ha empezado a incorporar estos nuevos elementos hasta la versión 9.

Las principales propiedades nuevas en CSS3 son:

- Selectores de atributos y propiedades
- Nuevas pseudo-clases
- Formatos de color: colores HSL, colores HSLA, colores RGBA, Opacidad
- Bordes: border-color, border-image, border-radius, box-shadow

- Fondos: background-origin, background-clip, background-size, capas con múltiples imágenes de fondo
- Texto: text-shadow, text-overflow, rotura de palabras largas, Web Fonts, creación de múltiples columnas de texto
- Modelo de caja básico: overflow
- Transiciones y transformaciones

A continuación veremos con más detalle cada una de estas nuevas propiedades.

2.1. Nuevos selectores de atributos

En primer lugar encontramos 3 nuevos selectores de atributos:

- **elemento[atributo^="valor"]**: Selecciona los elementos con ese atributo y que su valor comienza por la cadena de texto indicada en "valor".
- **elemento[atributo$="valor"]**: Selecciona los elementos con ese atributo y que su valor termina por la cadena de texto indicada en "valor".
- **elemento[atributo*="valor"]**: Selecciona los elementos con ese atributo y que su valor contiene la cadena de texto indicada en "valor".

Por ejemplo:

```
// Selecciona todos los enlaces que apunten a una dirección de correo:
a[href^="mailto:"]{...}
// Selecciona todos los enlaces que apuntan a páginas .php
a[href$=".php"]{...}
// Selecciona todos los enlaces que lleven a una página que contenga la
palabra ejemplo:
a[href*="ejemplo"]{...}
```

También incorpora nuevas formas de seleccionar etiquetas adyacentes:

- **h1 + h2{...}**: Etiquetas inmediatamente adyacentes.
- **h1 ~ h2{...}**: Selector general de hermanos. Válido cuando <h2> se encuentre después de <h1>, pero puede haber otras etiquetas de por medio.

Ejemplo:

```
<h1>Título</h1>
<h2>Subtítulo adyacente</h2>

<h1>Título</h1>
<p> párrafo de separación</p>
<h2>Subtítulo con selector general de hermanos</h2>
```

También podemos indicar atributos específicos de una etiqueta, con:

- **etiqueta1[atributo1=“valor1”]**: seleccionaría todas las etiquetas “etiqueta1” que contengan un atributo llamado “atributo1” cuyo valor sea igual a “valor1”. Por ejemplo, si queremos indicar un estilo para todas las etiquetas input que sean de tipo texto:

```
input[type="text"] {
   background: #eee;
}
```

2.2. Nuevas pseudo-clases

Una pseudo-clase es un estado o uso predefinido de un elemento al que se le puede aplicar un estilo independientemente del estilo aplicado al de su estado por defecto. En CSS3 se han añadido muchas nuevas pseudo-clases para facilitar a los programadores el uso de algunos estilos avanzados en el diseño de páginas Web. Las nuevas pseudo-clases son:

- **:nth-child(n)** - Fija el aspecto de una ocurrencia específica del elemento nodo hijo especificado. Por ejemplo, el tercer elemento nodo hijo de una lista sería "li:nth-child(3)". Además se pueden usar pequeñas expresiones como parámetro para por ejemplo seleccionar todos los elementos impares: "nth-child(2n+1)" los pares "nth-child(2n)", etc. Los elementos impares y pares también se pueden seleccionar usando "nth-child(odd)" y "nth-child(even)"
- **:nth-last-child(n)** - igual que ":nth-child(n)" pero empezando a contar desde el final.
- **:nth-of-type(n)** - Fija la apariencia de una ocurrencia específica del elemento con el tipo de selector especificado en un elemento padre. Por ejemplo la segunda lista no ordenada sería ul:nth-of-type(2). También permite los mismos parámetros que ":nth-child(#)".
- **:nth-last-of-type(n)** - igual que ":nth-of-type(n)" pero empezando a contar desde el final.
- **:first-child** - Fija el aspecto del primer elemento de un tipo de selector solo si es el primer nodo hijo de su elemento padre, por ejemplo la primera etiqueta <li> de una lista <ol>.
- **:last-child** - Ultimo elemento de una lista de elementos de un tipo dado.
- **:first-of-type** - Selecciona el primer elemento de un tipo concreto dentro de la lista de hijos.
- **:last-of-type** - Selecciona el último elemento de un tipo.
- **:only-child** - Selecciona el elemento si es el único elemento hijo.
- **:only-of-type** - Selecciona el elemento si es el único elemento hijo de ese tipo.
- **:empty** - Selecciona los elementos que no tienen hijos (incluyendo nodos de texto).
- **:enabled** - Selecciona los elementos de la interfaz que tengan el estado "enable".
- **:disabled** - Selecciona los elementos de la interfaz que tengan un estado "disabled".
- **:not(s)** - Selecciona los elementos que no coincidan con el selector especificado.
- **:lang(language)** - nos permite especificar estilos que dependen del idioma especificado por la propiedad language (en, sp, etc.)

Ejemplos de uso:

```
tr:nth-child(even) {
```

```
   background: silver;
}
tr:nth-child(odd) {
   background: white;
}
p:lang(en) {
   color: gray;
   font-style: italic;
}
```

Formularios

Además también se han añadido nuevas pseudo-clases que podemos usar en los formularios para aplicar un formato según el estado de un campo. Estas propiedades van en concordancia con los nuevos campos introducidos en HTML5 (ver la sección de formularios de HTML5). estas son:

- **:valid** - campo válido (dependerá del tipo de campo).
- **:invalid** - campo inválido (dependerá del tipo de campo).
- **:required** - campo requerido (marcado con el atributo “required”).
- **:optional** - campo opcional (campo no marcado con el atributo “required”).
- **:checked** - elemento marcado (o checked, válido para radio button o checkbox).
- **:in-range** - valor dentro del rango indicado (para campos numéricos o de rango).
- **:out-of-range** - valor fuera de rango (para campos numéricos o de rango).
- **:read-only** - campo de solo lectura.
- **:read-write** - campo de lectura / escritura.

Algunos ejemplos de uso:

```
<head>
   <style>
      #form1 input:valid { background:lightgreen; }
      #form1 input:invalid  { border-color:red; }
      #form1 specialInput input:valid { background:green; }
   </style>
</head>
<body>
   <form id="form1" name="form1" method="post" action="formaction.php">
      <p>Nombre:
       <input type="text" name="nombre" id="nombre" required/>
      <p/>
      <p>Usuario:
         <specialInput>
         <input type="text" name="usuario" id="usuario" required/>
         </specialInput>
      <p/>
   </form>
</body>
```

En este ejemplo cabe destacar la etiqueta “specialInput”, que no es ninguna etiqueta existente, sino una nueva etiqueta que hemos creado para aplicar un formato especial.

Además podemos aplicar estas pseudo-clases en cadena y hacer cosas como:

```
input:focus:required:invalid {
```

```
   background: pink url(ico_validation.png) 379px 3px no-repeat;
}
input:required:valid {
   background-color: #fff; background-position: 379px -61px;
}
```

Dado que Internet Explorer 6-8 no soporta la mayoría de pseudo-clases se han desarrollado algunas librerías de JavaScript que realizan las mismas funciones para estos navegadores, como "select[ivizr]" que podréis descargar de su página oficial "http://selectivizr.com/".

2.3. Color

En CSS3 se han incorporado nuevas formas para definir los colores:

- **rgba(red, green, blue, opacity);** - Color RGBA. El valor de opacidad debe de estar entre 0 y 1, siendo 0 totalmente transparente. Por ejemplo, podemos usarlo de la forma:

```
background-color: rgba(255, 115, 135, 0.5);
color: rgba(255, 115, 135, 0.5);
```

- **hsl(hue, saturation, lightness);** - Modelo de color HSL.
- **hsla(hue, saturation, lightness, alpha);** - Modelo de color HSLA.
- **cmyk(cyan, magenta, yellow, black);** - Modelo de color CMYK.
- **opacity: 0.5;** - También podemos indicar el valor de transparencia u opacidad por separado, debiendo de estar este valor entre 0 y 1, siendo 0 totalmente transparente y 1 totalmente opaco. Para dar también soporte a Internet Explorer usaremos: "*filter:alpha(opacity=50);*".

2.4. Bordes

En CSS3 se han incorporado cuatro nuevas propiedades para dar formato a los bordes de una caja. Estas propiedades no están todavía plenamente soportadas en todos los navegadores, por lo que para que funcione en la mayoría de ellos tendremos que usar también las propiedades nativas del navegador (simplemente añadiremos los prefijos -webkit- y -moz-). Las nuevas propiedades son:

- **border-radius**: permite crear cajas con esquinas redondeadas. Hasta ahora esto solo se podía hacer insertando imágenes que simularan esta característica. Ahora lo podemos hacer de una forma mucho más sencilla:

```
-webkit-border-radius: 30px;
-moz-border-radius: 30px;
border-radius: 30px;
```

Además también podemos indicar cada uno de los bordes por separado:

```
-moz-border-radius-topleft: 10px;
-moz-border-radius-topright: 20px;
```

```
-moz-border-radius-bottomright: 30px;
-moz-border-radius-bottomleft: 40px;
-webkit-border-radius: 10px 20px 30px 40px;
border-radius: 10px 20px 30px 40px;
```

- **border-image**: este nuevo estilo nos permite usar una imagen como borde de una caja. Tenemos que indicar tres atributos: la imagen a utilizar, el grosor y la forma de aplicar la imagen (stretch, repeat, round, none). Ejemplo de uso:

```
-webkit-border-image: url(imagen.png) 27 repeat;
-moz-border-image: url(imagen.png) 27 repeat;
border-image: url(imagen.png) 27 repeat;
```

  El resultado dependerá de la imagen que utilicemos para el borde, pero por ejemplo podríamos obtener resultados como el siguiente:

- **border-color**: Permite crear degradados en los bordes de una caja indicando la secuencia de colores del degradado (píxel a píxel y de dentro hacia fuera), de la forma:

```
-webkit-border-bottom-colors:#555 #666 #777 #888 #999 #aaa #bbb #ccc;
-webkit-border-top-colors:#555 #666 #777 #888 #999 #aaa #bbb #ccc;
-webkit-border-left-colors:#555 #666 #777 #888 #999 #aaa #bbb #ccc;
-webkit-border-right-colors:#555 #666 #777 #888 #999 #aaa #bbb #ccc;
-moz-border-bottom-colors:#555 #666 #777 #888 #999 #aaa #bbb #ccc;
-moz-border-top-colors:#555 #666 #777 #888 #999 #aaa #bbb #ccc;
-moz-border-left-colors:#555 #666 #777 #888 #999 #aaa #bbb #ccc;
-moz-border-right-colors:#555 #666 #777 #888 #999 #aaa #bbb #ccc;
border: 8px solid #000;
```

  Con lo que obtendríamos un resultado similar a:

- **box-shadow**: Permite dar sombra a elementos de bloque. Tiene 4 atributos: la distancia horizontal de la sombra, la distancia vertical de la sombra, el desenfoque (blur) y el color de la sombra. Además podemos usar valores negativos para las distancias horizontal y vertical para crear sombras en otros sentidos. Un ejemplo de sombra en color gris:

```
-moz-box-shadow:3px 3px 6px #888888;
-webkit-box-shadow:3px 3px 6px #888888;
box-shadow:3px 3px 6px #888888;
```

  Con lo que obtendríamos un resultado similar a:

2.5. Fondos

CSS3 también ha introducido nuevas propiedades para definir el estilo de las imágenes de fondo:

- **background-origin: border-box | padding-box | content-box** - permite definir el origen de coordenadas sobre el que se va a colocar la imagen de fondo. Acepta tres posible valores: “border-box” para que el fondo empiece desde el mismo borde del elemento, “padding-box” para que la imagen de fondo se coloque a partir del espaciado de padding, y por último “content-box” para que la imagen de fondo se coloque donde empieza el contenido del elemento, sin tener en cuenta el borde ni el padding.
- **background-clip: border-box | padding-box | content-box** - define el área sobre la que se extiende la imagen de fondo, puede tomar tres valores: “border-box” se extiende por toda el área dentro de la zona definida a partir del borde, “padding-box” se extiende a partir del espaciado de padding y “content-box” el fondo se extiende solo dentro del área de contenido.
- **background-size**: Permite indicar el tamaño de la imagen de fondo. Acepta diferentes atributos:
  - background-size: 200px; // especifica ancho y alto a la vez
  - background-size: 200px 100px; // 200px de ancho y 100px de alto
  - background-size: auto 200px; // ajustar la anchura automáticamente
  - background-size: 50% 25%; // También podemos indicar el tamaño con porcentajes
  - background-size: contain; // Escalar la imagen al tamaño máximo posible (conservando las proporciones originales) para que quepa dentro del área asignada.
  - background-size: cover; // Escalar la imagen para que cubra completamente el área asignada (conservando las proporciones originales).
- Capas con múltiples imágenes de fondo: Con la propiedad **background** ahora podemos indicar varias imágenes de fondo, simplemente separándolas con comas. Para cada propiedad background debemos definir cuatro valores: imagen de fondo, posición vertical, posición horizontal, modo de repetición (repeat, repeat-x, repeat-y, no-repeat). Ejemplo:

```
background: url(imagen1.png) 10px center no-repeat,
            url(imagen2.png) 0 center repeat-x;
```

Dado que estas propiedades no son soportadas todavía en todos los navegadores, deberemos de definirlas también añadiendo los prefijos “-webkit-” y “-moz-” para dar un mayor soporte.

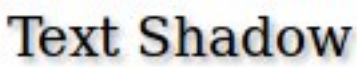
2.6. Texto

Las nuevas propiedades de CSS3 para dar formato a textos son:

- **text-shadow**: Permite dar sombra a un texto. Sus propiedades son distancia horizontal, distancia vertical, desenfoque (*blur*) y color de la sombra. Por ejemplo:

```
text-shadow: 2px 2px 2px #9e9e9e;
filter: dropshadow(color=#9e9e9e, offx=2, offy=2);
```

  Con lo que obtendríamos un resultado similar a:



- **word-wrap: break-word;** - Permite separar palabras muy largas dentro de un elemento de bloque. Por defecto toma el valor “normal”, por lo que las palabras largas se saldrían del borde del elemento. Con el valor “break-word” indicamos que las palabras pueden ser partidas para que quepan en el ancho de la caja, de la forma:

- **text-overflow: clip | ellipsis;** - Indica la forma de partir texto cuando excede el tamaño de su contenedor. Con “clip” el texto sobrante será cortado directamente aunque se quede una palabra por la mitad, mientras que “ellipsis” quitará la última palabra que no quepa y pondrá en su lugar unos puntos suspensivos. Esta propiedad de momento no funciona en Firefox.
- **font-face**: Permite utilizar tipografías diferentes a las estándar, que serán importadas desde un fichero indicado. De momento soporta los formatos: .eot, .ttf y .otf. Para importar una fuente hay que seguir la siguiente sintaxis:

```
@font-face{
   font-family:<nombre_fuente>;
   src: <source>;
   [font-weight:<weigth>];
   [font-style:<style>];
}
```

  Con “font-family” indicamos el nombre que le damos a la fuente, y “src” nos permite seleccionar el fichero a cargar. Los otros dos parámetros son opcionales y tendrán valor “normal” por defecto. Por ejemplo:

```
@font-face {
   font-family: ’LeagueGothic’;
   src: url(LeagueGothic.otf);
}
```

```
// Ahora ya podemos usar esta fuente:
p {
   font-family: 'LeagueGothic';
}
```

2.7. Columnas

Se han añadido nuevas propiedades que nos permiten crear columnas directamente a partir de un texto, estas son:

- **column-count:** Define el número de columnas en el que se va a dividir el texto. El texto será dividido de la mejor forma posible para que ocupe todo el espacio.
- **column-width:** Define el ancho de la columna (en unidades CSS).
- **column-gap:** Define el espacio entre las columnas (en unidades CSS).
- **column-rule:** Mediante esta propiedad podemos añadir una línea separadora entre las columnas, si no especificamos esta propiedad no se añadirá ninguna línea. Debemos de indicarle tres valores: ancho de la línea (en unidades CSS), estilo de la línea (solid, dotted, double, etc.) y color de la línea.

Para dar un mayor soporte antepondremos los prefijos -webkit- y -moz-, de la forma:

```
-webkit-column-count: 3;
-webkit-column-rule: 1px solid silver;
-webkit-column-gap: 10px;
-moz-column-count: 3;
-moz-column-rule: 1px solid silver;
-moz-column-gap: 10px;
column-count: 3;
column-rule: 1px solid silver;
column-gap: 10px;
```

Con lo que obtendríamos un resultado similar a:

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

2.8. Modelo de caja básico

Se han añadido nuevas propiedades para la disposición de elementos dentro de una caja:

- **overflow: visible | hidden | scroll | auto;** - permite indicar que ocurrirá si el contenido excede el área de un elemento, acepta cuatro posibles valores:
  - *visible*: No se recorta el contenido, la parte que quede fuera será visible. Es el valor por defecto.

  - *hidden*: El contenido que sobresalga será ocultado y tampoco se mostrará la barra de scroll.
  - *scroll*: El contenido se recorta y el navegador muestra la barra de scroll para ver el resto del contenido.
  - *auto*: Si el contenido se recorta el navegador mostrará una barra para ver el resto del contenido.
- **overflow-x**: igual que overflow pero indicaremos solo la propiedad en horizontal.
- **overflow-y**: igual que overflow pero solo para vertical.
- **resize: none | horizontal | vertical | both;** - habilita la posibilidad de redimensionar "manualmente" una caja. Puede tomar los valores: none, horizontal (permitir redimensionar solo en horizontal), vertical (solo en vertical), o both (redimensionar ambas dimensiones). Se recomienda además añadir la propiedad "overflow: hidden" para ocultar los elementos al redimensionar. Por ejemplo:

```
resize:both;
overflow:auto;
```

2.9. Transiciones

Una de las propiedades más novedosas que incorpora CSS3 es la posibilidad de crear animaciones mediante transiciones y transformaciones. Se pueden aplicar transiciones a la mayoría de propiedades (posiciones, fondo, color, tamaño, etc.). Desafortunadamente, no todos los navegadores usan los nombres estándares, por lo que tendremos que añadir los prefijos "-webkit-", "-moz-" y "-o-" para dar un mayor soporte. La buena noticia es que la sintaxis para los valores en todos ellos es consistente:

- **transition-property: propertyName;** - Indica la propiedad sobre la que se aplicará la transición. Se puede aplicar sobre casi todas las propiedades: background, color, height, width, border, etc. Además también podemos usar el valor "all" para que se aplique sobre todas las propiedades disponibles, por ejemplo:

```
-webkit-transition-property: all;
-moz-transition-property: all;
-o-transition-property: all;
transition-property: all;
```

- **transition-duration: duration;** - Indica el tiempo que debe durar la transición en segundos (0.5s) o en milisegundos (500ms):

```
-webkit-transition-duration: 1s;
-moz-transition-duration: 1s;
-o-transition-duration: 1s;
transition-duration: 1s;
```

- **transition-timing-function: timingFunction;** - Es la función de tiempo que seguirá la transición, indica los cambios de velocidad a lo largo de la animación. Puede tomar cinco valores diferentes: ease (valor por defecto), linear, ease-in, ease-out, ease-in-out y cubic-bezier(cp1x, cp1y, cp2x, cp2y). Por ejemplo:

```
-webkit-transition-timing-function: linear;
-moz-transition-timing-function: linear;
-o-transition-timing-function: linear;
transition-timing-function: linear;
```

- **transition-delay: delay;** - Permite establecer un retraso inicial antes de ejecutar la transición. El tiempo de retraso se debe de indicar en segundos (0.5s) o en milisegundos (500ms):

```
-webkit-transition-delay: 0.2s;
-moz-transition-delay: 0.2s;
-o-transition-delay: 0.2s;
transition-delay: 0.2s;
```

- **transition: propertyName duration timingFunction delay;** - También podemos indicar las cuatro propiedades explicadas en una sola línea:

```
-webkit-transition: all 1s linear 0.2s;
-moz-transition: all 1s linear 0.2s;
-o-transition: all 1s linear 0.2s;
transition: all 1s linear 0.2s;
```

En general, lo mejor es declarar la transición en la propiedad base, sin pseudo-clases. De esta forma conseguiremos que se ejecute en ambas direcciones, por ejemplo:

```
.btn1 {
   background: #9c3;
   -webkit-transition: background 0.3s ease;
   -moz-transition: background 0.3s ease;
   -o-transition: background 0.3s ease;
   transition: background 0.3s ease;
}
.btn1:hover {
   background: #690;
}
```

2.10. Transformaciones

La propiedad "**transform**" nos permite aplicar transformaciones 2D o 3D a un elemento. Por ejemplo nos permite rotar, escalar, mover, etc. el elemento indicado. Esta propiedad todavía no es soportada por todos los navegadores, por lo que tendremos que añadir los prefijos "-ms-", "webkit-", "-moz-" y "-o-" para dar un mayor soporte. Algunas de las funciones de transformación que podemos utilizar son:

- **none:** Indica que no se tiene que aplicar ninguna transformación.
- **translate(x,y):** Define una traslación 2D.
- **translateX(x):** Traslación en la coordenada X.
- **translateY(y):** Traslación en la coordenada Y.
- **scale(x,y):** Define una transformación de escalado 2D, deberemos de indicar valores entre 0.1 y 2.
- **scaleX(x):** Escalado en la coordenada X, deberemos de indicar valores entre 0.1 y 2.

- **scaleY(y):** Escalado en la coordenada Y, deberemos de indicar valores entre 0.1 y 2.
- **rotate(angle):** Aplica una rotación, el ángulo debe ser indicado en grados (ejem: "30deg").
- **skew(x-angle,y-angle):** Define una transformación 2D de sesgo (o torsión), indicada en grados (deg).
- **skewX(angle):** Define una transformación de sesgo sobre la coordenada X (indicada en grados).
- **skewY(angle):** Define una transformación de sesgo sobre la coordenada Y (indicada en grados).

Además también podemos indicar varias transformaciones en una misma línea, de la forma:

```
#myDIV {
   -moz-transform: scale(1.2) rotate(9deg) translate(5px, 2px) skew(5deg,
5deg);
   -webkit-transform: scale(1.2) rotate(9deg) translate(5px, 2px)
skew(5deg, 5deg);
   -o-transform: scale(1.2) rotate(9deg) translate(5px, 2px) skew(5deg,
5deg);
   -ms-transform: scale(1.2) rotate(9deg) translate(5px, 2px) skew(5deg,
5deg);
   transform: scale(1.2) rotate(9deg) translate(5px, 2px) skew(5deg,
5deg);
}
```

Hay muchos más tipos de transformaciones, aunque algunas de ellos no son funcionales todavía (sobre todo las funciones 3D), para más información consulta: "http://www.w3schools.com/cssref/css3_pr_transform.asp".

2.11. Más información

Existen algunas páginas Web que proporcionan "generadores de estilos CSS", facilitando en ocasiones este tipo de tareas:

- http://css3generator.com/
- http://www.colorzilla.com/gradient-editor/

Además, para obtener mucha más información sobre CSS3 podemos consultar:

- http://www.w3schools.com/css3/default.asp
- http://www.w3.org/TR/2001/WD-css3-roadmap-20010523/