U.D. 2. Uso de estilos(CSS). Parte 1

Principios básicos - Introducción a las hojas de estilo CSS.

Recursos

No hace falta decir que existen un gran número de recursos en la Web.

Echa un vistazo al índice de grupos de trabajo CSS de W3C encargado actual de las especificaciones CSS. http://www.w3.org/Style/CSS/current-work
También te puede servir de apoyo las referencias detalladas del navegador y otros artículos de desarrollo web de Peter-Paul Koch en http://quirksmode.org/ .
El proyecto InterAct ofrece un plan de estudios completo y permanente para el aprendizaje y la enseñanza de desarrollo web y diseño web incluyendo CSS. http://interact.webstandards.org/ .
El Centro de desarrolladores de Mozilla ofrece una completa referencia del lenguaje CSS y es ideal para mirar las cosas en un instante. **https://developer.mozilla.org/en/CSS_Reference**.
La red de desarrolladores de Opera ofrece artículos, tutoriales y referencias para todas las áreas del desarrollo web, incluyendo un plan de estudios para seguir uno mismo. http://dev.opera.com/

Escribir CSS

A diferencia de un lenguaje de programación como JavaScript, la sintaxis de CSS no es muy complicada. Las siguientes secciones destacan algunas cosas que debe saber antes de ir a las complejidades de lo que CSS puede hacer mas allá de la simple sintaxis.

Case Sensitive

CSS es no sensible a mayúsculas. Por ejemplo, la `color` propiedad es equivalente a `COLOR`, y la unidad `px` es la misma que una `PX` o `Px`.
Por convención, las propiedades y los valores suelen ser escritos utilizando minúsculas, que es la convención seguida en este libro y en general por todos los desarrolladores.
Las partes del código que no están bajo el control de la CSS como rutas de archivos a los documentos u hojas de estilos, imágenes, nombres de elementos, clases e ID son case sensitive y se definen en su forma original. Por ejemplo, la ruta de archivo en un servidor puede estar en mayúsculas y minúsculas, pero en otro servidor o máquina local puede que no. Para el marcado de los elementos de los documentos HTML se puede usar mayúsculas y minúsculas porque no son case sensitive, sin embargo, los elementos de documentos basados en XML sí que lo son.

Para evitar errores código o confusión, lo mejor es hacer coincidir el código independientemente de si es case sensitve o no.

Comentarios

Sólo hay una manera de escribir un comentario en CSS, empezando con los dos caracteres / * y terminando con los mismos dos caracteres invertidos, * /.
Cualquier texto, código o espacios en blanco entre los dos se ignora.

```
/ * Esto es un comentario * /
```

Los espacios en blanco

En CSS, los espacios en blanco, incluidos espacios en blanco, tabuladores y saltos de línea, no tiene ningún significado fuera de su uso como un selector descendiente (Capítulo 3) o como un separador de valores múltiples en una sola declaración. Fuera de esos dos casos, se considera opcional. Puedes utilizar espacios en blanco (o no) para formatear el CSS y ayudar con la organización y la legibilidad del código.

Comillas y caracteres escapados

La comilla simple (') y dobles (") se pueden utilizar indistintamente para delimitara cadenas en CSS (aunque si una cadena comienza con uno de ellos, debe terminar con el mismo).
La barra invertida (\) Es el carácter de escape en CSS. Se puede utilizar para escapar una comilla que es parte de una cadena (u otra barra invertida que debe aparecer como parte de la cadena). El carácter de barra invertida también se puede utilizar para incluir caracteres a través de sus códigos de caracteres.
Para algunas propiedades cuyo valor es una cadena, tales como con una propiedad url (), se puede usar comillas para delimitar la cadena. Palabras tales como nombres de colores, no son cadenas y no deben ir entre comillas.

Herramientas

La creación de páginas web, debido a los errores producidos por las disputas entre navegadores, necesita algo más que un editor de texto y un navegador. Las siguientes son algunas categorías de herramientas que añaden gran valor para afrontar esta tarea.

Herramientas de validación

El servicio de validación de W3C (Http://jigsaw.w3.org/css-validator/) es un validador de uso común. Como herramienta, los errores de un servicio de validación permiten detectar e identificar los errores que pudieran darse en los navegadores. Por ejemplo, es común que una etiqueta de cierre que falta pueda causar sangrado en una zona inesperada. Pero debes tener cuidado y debes entender los errores de validación antes de revisar tu código ya que la utilización de propiedades que aun no están estandarizadas pueden dar lugar a errores en la pagina que pueden ser subsanados modificando la configuración del validador.

Inspectores Web o Web Inspectors.

Inspectores Web (o los inspectores DOM) son herramientas que le permiten ver la estructura del documento, las propiedades CSS, y otra información acerca de una página web tal como aparece en su navegador, a menudo con un clic en el elemento en sí.
Estas herramientas son de gran valor para escribir y depurar código CSS, proveyendo en tiempo real información sobre las propiedades de estilo y señalando que reglas de estilo han contribuido a la aparición del elemento.

Internet Explorer: A partir de la versión 8, Internet Explorer incluye Herramientas de Desarrollo, un conjunto de herramientas integradas incluyendo la capacidad de inspeccionar Elementos HTML y ver información CSS. Para iniciar las herramientas, pulse F12 o seleccione Opciones> Herramientas de Desarrollo en el menú en IE. Para obtener más versiones de IE, incluyendo 6 y 7, Microsoft ofrece una descarga llamada el Internet Explorer Developer Toolbar.

Firefox: Entre otras características, la extensión para Firefox Firebug (Http://www.getfirebug.org/) permite la visualización y edición del árbol del documento y sus propiedades CSS. Una vez instalado, Firebug puede ser abierto directamente o haciendo clic derecho en un elemento de la página y elegir Inspeccionar elemento. Firebug ya no es necesario y ofrece la herramienta de inspección por defecto.

Safari: Safari en OS X y Windows viene con un conjunto integrado de herramientas de desarrollo incluyendo un inspector web. Estas herramientas están desactivadas por defecto, pero se pueden activar desde el panel Avanzadas dentro de Safari preferencias. Una vez activado, puede abrirlo desde el menú Desarrollo o hacer clic en un elemento de la página y elegir Inspeccionar Elemento.

Chrome: Chrome también se incluye con las herramientas de desarrollo integradas similar a las anteriores. Para acceder a las herramientas, seleccione Ver> Desarrollo> Herramientas de Desarrollo en el menú o haga clic derecho un elemento de la página y seleccione Inspeccionar elemento.

Opera: Opera Dragonfly es otra suite de herramientas para trabajar con documentos web con información visual de estilo. Para activar Dragonfly, seleccione Herramientas> Opciones avanzadas> Opera Dragonfly en el menú de Opera, o haga clic en un elemento de la página y selecciona Inspeccionar Elemento.

Web Developer Toolbar

Chris Pederick ha creado la extensión Web Developer Toolbar (Http://chrispederick.com/work/web-developer/) para Firefox y Chrome que ofrece algunas características ingeniosas que no se encuentran en los inspectores de web estándar tales como la capacidad para agregar una superposición encima del documento que muestra la estructura del documento o atributos de los elementos, cambiar el tamaño de la ventana del navegador a unas dimensiones determinadas para pruebas o llevar el documento directamente a los servicios de validación.

Yahoo! YSlow y Google Page Speed

Yahoo! YSlow (http://developer.yahoo.com/yslow/) es un add-on para Firefox dirigido a analizar y mejorar el rendimiento de los sitios web en áreas tales como el almacenamiento en caché, el tamaño de descarga, y las velocidades, así como la reducción del número de peticiones de entrega de contenido realizadas al servidor y todos sus tipos de activos (imágenes, hojas de estilo, scripts, etc,..).
A través de YSlow y su suite, se puede aprender acerca de las herramientas para comprimir documentos CSS, optimizar las llamadas al servidor cuando llama a archivos CSS, archivos JavaScript e imágenes, y como realizar un montón de trucos de rendimiento diferentes a los cubiertos directamente aquí.

Google Page Speed (http://code.google.com/speed/page-speed/) es otro Firebug add-on en la misma línea como YSlow. Identifica qué declaraciones CSS no están siendo utilizados por un documento HTML y señala cuáles de sus selectores CSS se escriben ineficiente y por qué.

Conceptos Básicos de CSS

Existe una terna de tecnologías basadas en estándares de desarrollo web que cuando se usa en conjunto permite crear emocionantes, vibrantes e interactivos sitios web mas allá de lo que en sí mismo son; sólo un montón de archivos de texto.
HTML proporciona el contenido y la estructura de la página web, JavaScript suministra la interacción y la manipulación del documento, y CSS proporciona la presentación y el estilo.

¿Qué es CSS?

CSS, acrónimo de Cascading Style Sheets, es un lenguaje para describir las propiedades de presentación de los elementos de contenido en documentos estructurados tales como documentos HTML. Nos centraremos en contenido HTML, aunque también puede utilizar CSS para otros documentos estructurados como los creados con XML o SVG.

¿Qué son las hojas de estilo?

Las hojas de estilo son un conjunto de directrices para dotar de estilo a un documento estructurado mediante la definición de reglas de apariencia para diferentes tipos de contenido o diferentes contextos en que los contenidos se pueden encontrar. Las hojas son utilizadas por suites de oficina o programas de correo electrónico. Es común en los programas de presentaciones, tales como PowerPoint o Keynote escoger un tema al empezar, donde cada diapositiva se formatea automáticamente con los mismos tamaños de fuente, colores y diseños, en lugar de comenzar con una diapositiva en blanco y diseñar cada diapositiva individualmente con la esperanza de que sean coherentes.
Cuando un navegador carga el contenido HTML, también carga la información de la hoja de estilos. A partir de esta información de la hoja de estilo, entonces "acumula" el conjunto de reglas de presentación para cada elemento de contenido en función del tipo de elemento, su estado y su locación en el documento. En última instancia, renderiza cada elemento a partir de este conjunto "acumulado" de reglas.

Declaración o instrucción CSS

Las hojas de estilo CSS consisten en una lista de declaraciones. Hay dos tipos de declaraciones: conjuntos de reglas (rule-set, en adelante, reglas) y @-rules (at rules).

Conjuntos de reglas

Los conjuntos de reglas constan de un selector seguido de un bloque de declaración que contiene las declaraciones de propiedades de estilo y sus valores, como se explica en la siguiente lista.

1. Conjunto de reglas: Esta es la definición completa de una regla CSS, incluyendo selector y el bloque de declaración, que contiene declaraciones individuales.
2. Selector: El selector incluye todo hasta la llave de apertura de bloque. El selector describe los elementos de marcado sobre los que hay que aplicar el bloque de declaración. Selectores individuales pueden compartir un bloque de declaraciones, con cada selector separados con una coma (,).
3. Bloque de declaración: El bloque de declaración comienza con la llave de apertura y termina con la llave de cierre. Dentro del bloque hay cero o más declaraciones, cada uno separado por un punto y coma (;).
4. Declaración: Cada declaración tiene dos puntos que separan el par propiedad-valor.
5. Propiedad: La propiedad es la propiedad CSS de la declaración.
6. Valor: Este es el valor que se aplica a la propiedad declarada.

La sintaxis del valor depende de la propiedad, pero pueden ser cosas tales como palabras clave, una <longitud>, Un <porcentaje>, o una mezcla de varios tipos separados por espacios.

Definición de valores para propiedades con cuatro lados

Propiedades tales como `margin, padding` y `border-width` se utilizan para definir los valores de los cuatro lados de un bloque (mientras `margin-right` define solo el margen derecho). Estas propiedades, y otras como estas, pueden tener desde uno hasta cuatro valores separados por espacios que se aplican a los lados de la siguiente manera:
■ Si hay un valor en la lista (por ejemplo, 10px), Ese valor se aplica a los cuatro lados.
■ Si se enumeran dos valores (por ejemplo, 10px 5%), El primer valor se aplica a la parte superior e inferior, mientras que el segundo se aplica a los lados derecho e izquierdo.
■ Si se enumeran tres valores (por ejemplo, 10px 20px 5%), El primer valor se aplica a la parte superior, el segundo a los lados derecho e izquierdo, y el último en la parte inferior.
■ Si se enumeran los cuatro valores (por ejemplo, 10px 20px auto 5%), Los valores se aplican según las agujas del reloj desde la parte superior (arriba, derecha, abajo, izquierda).

At- rules

Las at-rules son declaraciones que comienzan con el carácter de @, seguido por un tipo de regla o identificador, y terminan con un punto y coma. A diferencia de los conjuntos de reglas, no contienen declaraciones directamente, sino ofrecen un contexto adicional o comandos para el procesamiento de la información de hojas de estilo. He aquí un ejemplo:

```
/ * Additional.css archivo a incluir * /
@import "additional.css"
```

Cascada

En CSS, la cascada es el proceso que se sigue para determinar qué declaración para una propiedad dada se debe aplicar a un elemento dado en el documento. Por ejemplo la propiedad color puede ser definida y redefinida varias veces, por lo que el navegador debe determinar cuál de estas definiciones debe aplicar. Los criterios de clasificación a través de la hojas de estilo para determinar cuál es la declaración de propiedad a utilizar son tres: peso, la especificidad y el orden de aparición.

El peso de la declaración está determinado por el origen o fuente del estilo declarado. Las reglas de estilo se pueden encontrar en uno de estos tres lugares en orden descendente de peso:
■ Hojas de estilo de autor: Estas son las hojas de estilo definidas junto con el documento HTML por el autor de la página visitada., bien sea en el head o con el atributo style en una etiqueta HTML.
■ Hojas de estilo de usuario: Hojas de estilo CSS u otro estilo de las preferencias seleccionadas por el usuario del navegador.
■ Hojas de estilo del navegador: Cada navegador de usuario aplica un conjunto predeterminado de reglas que representan comportamientos comunes para cada elemento HTML (Los enlaces se resaltan, las cabeceras son más grandes, y así sucesivamente).

La especificidad de la declaración está determinada por el grado de precisión del selector utilizado para el elemento. Un selector que dice "cualquier párrafo elemento "(<p>) es menos específico que un selector en busca de "cualquier párrafo que se produce en el interior de un bloque de una cita "(<blockquote> <p>).

El orden de aparición, u orden de la fuente de la declaración está determinado por el orden en que las reglas se encuentran en el conjunto de documentos por orden de aparición. De tal forma que la última declaración sustituye a todas las anteriores (ósea, se machaca la propiedad). .
Para el cálculo de la declaración ganadora para una propiedad dada, se considera primero el peso. Si múltiples declaraciones comparten el origen de mismo peso, entonces se considera la especificidad. Por último, si las declaraciones múltiples comparten la misma especificidad, a continuación, se utiliza el orden de aparición.

Herencia

La herencia en CSS es el mecanismo mediante el cual determinadas propiedades de un elemento padre se transmiten a sus hijos. No todas las propiedades CSS son heredadas, porque algunas de ellas no tendría sentido que lo fueran. Por ejemplo, los márgenes no se heredan porque es poco probable que un elemento hijo necesite los mismos márgenes que su padre. Normalmente, el sentido común dicta qué propiedades se heredan y cuáles no, pero para estar del todo seguros, debemos consultar cada propiedad en la tabla de resumen de propiedades de la especificación CSS 2.1.

Para qué sirve la herencia

¿Por qué tiene CSS un mecanismo de herencia? Probablemente, la manera más sencilla de responder a esta pregunta sea pensar qué pasaría si no existiera la herencia. Se

deberían especificar cuestiones como la familia de fuentes, el tamaño de la fuente y el color del texto individualmente para todos y cada uno de los tipos de elemento.

Mediante la herencia, por ejemplo, se pueden especificar las propiedades de las fuentes de los elementos `html` o `body` y todo el resto de elementos los heredarán. Se pueden especificar los colores de fondo y de primer plano de un elemento contenedor concreto y todos los elementos hijos de este contenedor heredarán automáticamente el color de primer plano. El color de fondo no se hereda, pero el valor inicial de `background-color` (color de fondo) es `transparent`, lo cual significa que el fondo del padre se verá a través de él. El efecto es el mismo que si se heredara el color de fondo, pero pensad qué sucedería si se heredaran las imágenes de fondo: todos los hijos tendrían la misma imagen de fondo como padre y, por lo tanto, todo parecería un rompecabezas creado por alguien con problemas de drogas, ya que el fondo "volvería a empezar desde cero" para cada elemento.

Cómo funciona la herencia

Todos los elementos de un documento HTML heredan todas las propiedades heredables de su padre excepto el elemento raíz (`html`), que no tiene progenitor. El hecho de que las propiedades heredadas tengan algún efecto o no depende de otros factores, como veremos más adelante cuando hablemos de la cascada.

Un ejemplo de herencia

1. Copiad el siguiente documento HTML en un fichero nuevo del editor de textos que más os guste y guardadlo como `inherit.html`.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html lang="en">
   <head>
      <meta    http-equiv="Content-Type"    content="text/html; charset=utf-8">
      <title>Herencia</title>
   </head>
   <body>
      <h1>Título</h1>
      <p>Un párrafo de texto.</p>
   </body>
</html>
```

   Si abrís el documento en el navegador web, veréis un documento bastante aburrido que se muestra según el estilo por defecto de vuestro navegador.

13. Cread un nuevo fichero vacío con el editor de textos, copiad dentro la regla CSS que se muestra a continuación y guardad el fichero como `style.css` en la misma ubicación que el fichero HTML.

```
html {
   font: 75% Verdana,sans-serif;
}
```

16. Enlazad la hoja de estilos en el documento HTML insertando la línea siguiente antes del tag `</head>`.

```
<link rel="stylesheet" type="text/css" media="screen" href="styles.css">
```

17. Guardad el fichero HTML modificado y recargad el documento en el navegador. La fuente pasará de ser la predeterminada por el navegador (normalmente Times o Times New Roman) a ser Verdana. Si no tenéis Verdana instalada en el ordenador, el texto se mostrará con la fuente Sans Serif especificada por defecto en la configuración del navegador. Además, el texto se verá un 25% más pequeño que en la versión sin estilo.

    La regla CSS que hemos especificado se aplica únicamente al elemento `html`. No hemos especificado ninguna regla para los títulos o los párrafos, pero ahora todo el texto se muestra en Verdana al 75% del tamaño por defecto. ¿Por qué? Por la herencia.

    La propiedad `font` es una propiedad abreviada que establece toda una serie de propiedades relacionadas con las fuentes. Sólo hemos especificado dos, el tamaño de la fuente y la familia de fuentes, pero esta regla equivale a lo siguiente:

    ```
    html {
       font-style: normal;
       font-variant: normal;
       font-weight: normal;
       font-size: 75%;
       line-height: normal;
       font-family: Verdana,sans-serif;
    }
    ```

    Todas estas propiedades se heredan, de manera que el elemento `body` las heredará del elemento `html` y después las transmitirá a sus hijos: el título y el párrafo.

    Pero, ¡un momento! Hay algo que no acaba de quedar claro respecto a la herencia del tamaño de la fuente, ¿verdad? El tamaño de la fuente del elemento `html` se establece en 75%, pero ¿75% de qué? ¿Y el tamaño de la fuente de `body` no debería ser el 75% del tamaño de la fuente de su padre y los tamaños de las fuentes del título y del párrafo deberían ser el 75% del tamaño del elemento `body`?

    El valor que se hereda no es el valor especificado, es decir, el valor que escribimos en la hoja de estilo, sino algo que se llama *el valor computado*. El valor computado es, en el caso del tamaño de la fuente, un valor absoluto medido en píxeles. Para el elemento `html`, que no tiene un elemento padre del cual heredar, un porcentaje del valor de tamaño de fuente se asocia al tamaño de fuente predeterminada del navegador. La mayoría de los navegadores actuales tienen un tamaño de fuente predeterminada de 16 píxeles. El 75% de 16 son 12, de manera que el valor computado del tamaño de la fuente del elemento `html` será probablemente 12 píxeles. Éste es el valor que hereda `body` y que se transmite al título y al párrafo.

(El tamaño de la fuente del título es mayor porque el navegador aplica algunas normas de estilo integradas propias. Podéis ver el tema de la cascada a continuación.)

18. Añadid dos declaraciones más a la regla de la hoja de estilo de CSS:

```
19.  html {
20.      font: 75% Verdana,sans-serif;
21.      background-color: blue;
22.      color: white;
     }
```

23. Guardad el fichero CSS y recargad el documento en el navegador.

    Ahora el fondo es de color azul fuerte y todo el texto es blanco. La regla se aplica al elemento `html`: el documento entero, cuyo fondo será azul. El elemento `body` hereda el color blanco de primer plano y se transmite a todos los hijos de `body`: en este caso, el título y el párrafo. Éstos no heredan el fondo, pero el fondo se establecerá en el valor por defecto de `transparent`, de manera que el resultado visual final será texto blanco sobre fondo azul.

24. Añadid otra regla nueva a la hoja de estilo y guardad y recargad el documento.

```
25.  h1 {
26.      font-size: 300%;
     }
```

Esta regla establece el tamaño de la fuente del título. El porcentaje se aplica al tamaño de fuente heredada (el 75% de la predeterminada por el navegador, que suponemos que es 12 píxeles), de manera que el tamaño del título será el 300% de 12 píxeles, es decir: 36 píxeles.

Forzar la herencia

Mediante la palabra clave `inherit` (heredar) puede forzarse la herencia incluso para propiedades que no se heredan normalmente. Sin embargo, se debe utilizar con mucho cuidado porque Microsoft Internet Explorer (hasta la versión 7 incluida) no es compatible con esta palabra clave.

La regla siguiente hace que todos los párrafos hereden todas las propiedades de fondo de sus padres:

```
p {
   background: inherit;
}
```

Con las propiedades abreviadas se puede utilizar `inherit` en vez de los valores normales. Se debe utilizar la versión abreviada o bien para todo o bien para nada en absoluto. En la versión no abreviada no se pueden especificar algunos valores y utilizar `inherit` para otros porque los valores pueden darse en cualquier orden y no hay manera de especificar qué valores queremos heredar.

Forzar la herencia no es algo que haya que hacer a menudo. Puede ser útil para "deshacer" una declaración de una regla conflictiva o para no tener que introducir los

datos de determinados valores directamente en el código fuente. Un ejemplo de esto sería el típico menú de navegación:

```
<ul id="nav">
   <li><a href="/">Inicio</a></li>
   <li><a href="/news/">Noticias</a></li>
   <li><a href="/products/">Productos</a></li>
   <li><a href="/services/">Servicios</a></li>
   <li><a href="/about/">Sobre nosotros</a></li>
</ul>
```

Para mostrar esta lista de enlaces como menú horizontal, podéis utilizar el CSS siguiente:

```
#nav {
   background: blue;
   color: white;
   margin: 0;
   padding: 0;
}
#nav li {
   display: inline;
   margin: 0;
   padding: 0 0.5em;
   border-right: 1px solid;
}
#nav li a {
   color: inherit;
   text-decoration: none;
}
```

En este caso, el color de fondo de toda la lista se establece en azul en la regla de `#nav`. Así, también se establece el color de primer plano como blanco y todos los elementos de la lista y todos los enlaces heredan el mismo. La regla de los elementos de la lista establece un límite a la derecha, pero no especifica el color del margen, lo que significa que utilizará el color de primer plano heredado (el blanco). Para los enlaces, hemos utilizado `color:inherit` para forzar la herencia y anular el color de los enlaces predeterminado del navegador.

Lógicamente, también podría haber especificado el blanco como color del margen y del texto de los enlaces, pero lo mejor del hecho de dejar que lo haga la herencia es que, si más adelante decidimos cambiar los colores, sólo deberemos hacer un cambio en este punto.

Bloque, in-line y elementos externos

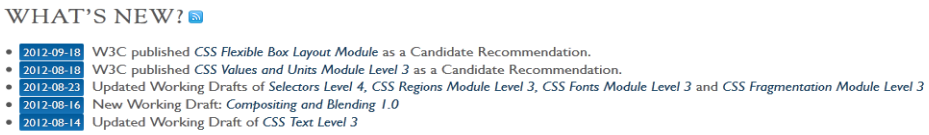
Hay tres tipos principales de elementos en los documentos HTML (generalización) cuyo comportamiento y presentación se puede determinar con CSS. Hay elementos que sirven como contenedores para otros contenidos (,<p>), Hay elementos que diferencian los tipos de contenido de texto (<a>,), Y hay elementos
que se refieren al contenido externo (<img>,<object>).
El primer tipo son elementos de bloque, el segundo tipo elementos en línea, y los últimos son elementos externos.

Estándares Web y Especificaciones

El Consorcio World Wide Web (W3C), junto con otros organismos como WHATWG IETF elaboran normas y especificaciones de documentos, desde CSS hasta el protocolo HTTP. Puedes encontrar las especificaciones para CSS y HTML en el sitio web de W3C http://www.w3.org/.

CSS2, CSS2.1, CSS3, Borradores, recomendaciones, Ack!

El estado de la especificación o "estándar" CSS, aunque activo y emocionante, a menudo puede ser confuso. El Grupo de trabajo W3C CSS está formado por representantes de los proveedores de navegadores y otros expertos que están activamente escribiendo y manteniendo las especificaciones.
A primera vista, hay una gran cantidad de documentos de especificaciones, y todos están en puntos diferentes en su proceso de creación.
Los diferentes estados de cada especificación pueden ser:
■ Borrador de Trabajo(Working Draft) : Un borrador de trabajo **(WD)** es la primera definición de la especificación. Los proveedores de navegadores lo utilizan para crear implementaciones de prueba. Es muy posible que haya cambios en el documento.
■ Último Borrador de Trabajo(Last Call Working Draft): Cuando los problemas, los conflictos y las preguntas del Borrador de Trabajo se resuelven, hay una última llamada **(LC)** o período de comentarios, anunciado para solicitar retroalimentación sobre el borrador.
■ Candidato Recomendación (Candidate Recommendation:): Después de ese período Last Call, un proyecto puede pasar a Candidato Recomendación (**CR**), y el grupo de trabajo solicita implementaciones de prueba de los proveedores para asegurarse de que lo que han propuesto es viable.
■ Propuesta de Recomendación: Para ser una Propuesta a Recomendación (**PR**), la especificación ha de ser estable, y los proveedores de navegadores deben haber creado implementaciones interoperables.
■ Recomendación: El proyecto de ley se ha convertido en una ley o se ha finalizado la Recomendación (**REC**).



CSS Levels

El proceso para llegar a una recomendación puede ser muy largo en el tiempo. Aunque CSS3 está en la mente de todos, CSS 2.1 ahora está listo para convertirse en una recomendación(REC), por lo que es a veces es difícil saber qué parte de la recomendación esta lista para ser usada en un sitio web.
Actualmente la implementación de CSS se establece en niveles:
■ CSS1:(CSS Level 1) se considera obsoleta.
■ CSS2: CSS2 se convirtió en una recomendación en 1998. Como especificación, ha sido reemplazada por CSS2.1, aunque el término "CSS2" puede utilizarse para referirse a cualquiera de los dos.

■ CSS2.1:(CSS Level 2 Revision 1) En la última parte de 2010, la especificación CSS2.1 está preparada para el estado de PR. CSS2.1 ha hecho cambios a CSS2 mediante la actualización de las técnicas de descriptores y la eliminación de las propiedades para reflejar mejor la aplicación (o la falta total de aplicación) de la anterior recomendación.
■ CSS3:(CSS Level 3) Con CSS3, la especificación se ha dividido en módulos para controlar la complejidad, así como proporcionar a los navegadores un medio claro para establecer la compatibilidad. Algunos módulos son CR con mejor soporte o incluso están implementándose, y otros todavía no han sido escritos.

"3. Cascading Style Sheets Definition"

As of 2010, *Cascading Style Sheets (CSS)* is defined by the following specifications.

1. CSS Level 2 Revision 1 (including errata)
2. CSS Style Attributes
3. Media Queries Level 3
4. CSS Namespaces
5. Selectors Level 3
6. CSS Color Level 3

TABLE OF SPECIFICATIONS

Ordered from most to least stable:

Completed	Current	Upcoming	Notes	i
CSS Snapshot 2010	NOTE	–	Latest stable CSS	i
CSS Snapshot 2007	NOTE	–		i
CSS Color Level 3	REC	REC	See Errata	i
CSS Namespaces	REC	REC		i
Selectors Level 3	REC	REC		i
CSS Level 2 Revision 1	REC	REC	See Errata	i
CSS Level 1	REC	–	Unmaintained, see Snapshot	i
Media Queries	REC	REC		i
Stable	**Current**	**Upcoming**	**Notes**	**i**
CSS Style Attributes	CR	PR		i
Testing	**Current**	**Upcoming**	**Notes**	**i**
CSS Backgrounds and Borders Level 3	CR	PR		i
CSS Image Values and Replaced Content Level 3	CR	PR		i
CSS Marquee	CR	PR		i
CSS Multi-column Layout	CR	CR		i
CSS Speech	CR	PR		i
CSS Values and Units Level 3	CR	PR		i
CSS Mobile Profile 2.0	CR	PR	Status unknown	i
CSS TV Profile 1.0	CR	?	Status unknown	i

Para conocer la situación actualizada de CSS 2.1 y los diversos módulos de CSS3, consulte http://www.w3.org/Style/CSS/current-work. Y no sólo se refieren a la situación de la especificación, con todas estas especificaciones, módulos de CSS3; sino también presentan tablas de compatibilidad del navegador para obtener más información sobre si es posible la aplicación de las diversas propiedades que quieras utilizar.

Trabajar con CSS

Ahora que ya sabes todo lo que siempre quisiste saber sobre el definición de la CSS y el proceso de especificación, es posible que necesites saber cómo incluir el código CSS real en sus páginas web.

Colocación de estilos para HTML

Antes de empezar a escribir CSS, tienes que saber dónde colocar el código. Hay algunas maneras para definir reglas, algunas basadas en archivos externos que se pueden compartir entre varios documentos HTML de un sitio y otras que son más específicas a una página o incluso sobre elementos individuales.

Etiqueta <link>

Puedes utilizar la etiqueta <link> en el elemento <head> de un documento para especificar un documento CSS externo. Este documento no contiene marcas o elementos, sólo el código CSS (normas y comentarios).

```
<head>
<link rel="stylesheet" type="text/css" src="global.css">
</ Head>
```

El atributo type define el idioma que se utiliza en la hoja de estilos. Requerido para HTML4 y XHTML, es opcional en HTML5. El atributo src define la ubicación del documento CSS; si se trata de una ruta relativa, es relativa a la ubicación del documento HTML.

Etiqueta <style>

Puedes utilizar la etiqueta <style> en el elemento <head> de un documento para incrustar el código CSS que se aplica al documento.

```
<head>
<style type="text/css">
[...]
</ Style>
</ Head>
```

El atributo type define el idioma que se utiliza en la hoja de estilos. Requerido para HTML4 y XHTML, es opcional en HTML5.

HTML5 también define un atributo scope (alcance) que permite la etiqueta<style> dentro de un bloque de contenido.

@import Rule

Puedes utilizar el @import en la parte superior de un bloque de código CSS o documento CSS para definir otro documento CSS para ser incluido en el documento actual o bloque de código. La regla @import debe preceder a todos los demás reglas en el documento (con la excepción de la propuesta @charset de CSS3).

```
@ Import "imported.css"
```

Atributo HTML style

Se puede utilizar para asignar una lista de las declaraciones CSS separadas por un punto y coma para un elemento específico.

```
<p style="color: pink"> Un párrafo Pink </ p>
```

Es la forma menos eficiente de utilizar estilos pero la mas prioritaria. Puede ser útil para casos especiales en los que deseemos modificar el estilo a un único elemento.

JavaScript y el DOM

JavaScript puede acceder a las hojas de estilo y las propiedades de estilo a través del Documento Object Model (DOM). JavaScript puede ser usado para fijar estilos a través del estilo del objeto de un elemento `.style` o para leer información de estilo actual (valores utilizados) a través de la `.getComputedStyle` método.
También puede usar JavaScript para leer el contenido de las hojas de estilo a través del objeto `document.styleSheets`.

Propiedades formadas por múltiples palabras en CSS se trasladan a JavaScript poniendo mayúscula en lugar de insertar un guión. Por ejemplo, `margin-left` se convierte en `MarginLeft`.

Estilos de codificación

CSS no tiene en cuenta ni los espacios en blanco ni sangrías para analizar el código pero conjuntamente con la utilización de los comentarios CSS pueden hacer que el código sea mas fácil de entender y que el documento tenga una buena organización.

Ubicaciones Código

CSS tiene mayor valor cuando se comparten hojas entre varios documentos o sitios enteros en lugar de volver a escribir o copiar y pegar el documento. Por lo tanto es habitual separar en grupos de reglas según el tipo de contenido al que se van a aplicar:
■ Hoja de estilo global: Incluye la información de estilo que puede ser aplicado a la sitio. Estos estilos deben ser en uno de los primeros vinculas externos CSS.
■ Hoja de estilo de sección o estilo de tipo de página: Incluye la información de estilo que puede ser aplicado a una subsección o pagina alternativa que aumenta o cambia el contenido de la pagina de estilos global. Se hace con una hoja de estilo externa que debe de incluirse después de la hoja de estilos global y a las paginas que se les debe aplicar se les añade un identificar id a la etiqueta body.
■ Hoja de estilo de página o hoja de estilos específicos de contenido: Incluye la información de estilo que es compartida entre páginas con poca frecuencia o no en entre todas. Si se trata de una pequeña cantidad de código, se puede colocar tanto en el global o en documentos CSS de sección; Sin embargo, si es más extensa, puede ser utilizado un tercer documento vinculado.
■ Estilo para un único documento: Utilizadas para paginas que no se integran específicamente con nuestro sitio. En estos casos, un bloque de estilo y no una hoja externa puede reducir la número de archivos externos que necesitan la gestión de las solicitudes al servidor.

La implementación de este tipo de estructura, si todos los niveles de granularidad son útiles, podría ser algo como lo siguiente:

```
<head>
[...]
type="text/css" <link href="styles/global.css">
type="text/css" <link href="styles/forums.css">
type="text/css" <link href="styles/forum_help.css">
[...]
</ Head>
```

Documentación del Código y comentarios

Aunque las reglas CSS individuales son muy fáciles de leer y entender , incluso la web más básica tienen unos cientos de líneas de código CSS. Dado que CSS no tiene una estructura de árbol como HTML, el uso de comentarios CSS es básico y necesario para distinguir las secciones y la estructura del documento, así como tomar notas sobre los selectores individuales o propiedades es importante.
Comienza cada documento con un poco de información sobre lo que es el documento y debe contener un índice de su contenido, lo que lo hará mas fácil de entender.

```
/ **
* Estilos globales
* Yournewwebsite.com
*
* Contenido:
* 1. Elementos básicos de HTML
* 2. Diseño de cuadrícula
* 3. Estilos de cabecera Content
* 4. Estilos de página Contenido
* 4 bis. Índice de la página de contenido
* 4b. Artículo página de contenido
* 5. Estilos de pie de página de contenido
* /
```

Para distinguir las secciones del documentotales como donde se especifica el marcado de la que cuadrícula de diseño o donde se encuentren los estilos de formularios, usa una o dos líneas de comentarios que llamen la atención a través de el documento.

```
/ ******** Estilos de pie de página de contenido ******** /
```

Para comentar una regla CSS, poner un comentario en la línea antes de la regla.

```
/ * Crear un enlace fuerte y colorido * /
div.newsletterSignup a {
      font-size: 2em;
      color: pink;
}
```

Para hacer comentarios sobre una propiedad individual o el valor, ponga un comentario después la declaración o en la línea antes de ella.

```
article {
      min-width: 500px; / * lo suficientemente amplio para el
contenido de la imagen * /
}
```

Disposición de los selectores

Los comentarios no son la única manera de ayudar a que una gran cantidad de código CSS sea más fácil de leer, analizar y mantener. Las convenciones de codificación sobre la apariencia del formato de la reglas CSS y la organización de código de son otra ayuda clave.
Los espacios en blanco alrededor de las reglas CSS y la sangría (o no) de las reglas y las declaraciones individuales deben ser consistentes. Hay dos formas principales para dar formato a las reglas, y ambas tienen sus ventajas e inconvenientes.
Hasta ahora se han presentado las reglas de la forma más común con una declaración por línea y una sangría desde la izquierda.

```
#footer form.newsletter input[type=text] {
      width: 120px;
      margin-bottom: 5px;
```

```
      color: #666;
      background: #ccc;
}
```
Esto hace que sea fácil de detectar una declaración y hacer comentarios sobre las declaraciones individuales, pero puede hacer que sea más difícil para detectar grupos de selectores. Tener una regla entera en una sola línea puede hacer la exploración y la identificación de grupos de reglas mucho más sencillo, aunque requiere un poco de exploración horizontal para encontrar las propiedades individuales o sus valores.

```
#footer { [...] }
#footer form.login { [...] }
#footer form.login input[type=text] { [...] }
#footer form.newsletter { [...] }
#footer form.newsletter p { [...] }
#footer form.newsletter input[type=text] { [...] }
#footer form.newsletter input[type=submit] { [...] }
```

Cómo organizar tus reglas es también importante. La cascada depende del orden de aparición de los selectores y la especificidad es un buen punto de partida para ver cómo un documento puede ser organizado, utilizando reglas globales y reglas genéricas en primer lugar, seguidos de normas más específicas para un único tipo de contenido o estructura de marcado. Pero en algún momento tus selectores estarán centrado en tipos más singulares de contenido y que no coincidan con el mismo elemento de contenido, así que el orden no será ya importante. Es bueno mantener agrupado el código pertenecientes a los mismos bloques o tipos de contenido (todo el contenido del header junto y todo el contenido de pie de página juntos, por ejemplo).

El comentario, visto anteriormente en este capítulo, es un buen ejemplo de cómo se puede organizar la estructura general del documento global de un sitio CSS pero no es el único.

```
/ *
* Contenido:
* 1. Elementos básicos de HTML
* 2. Diseño de cuadrícula
* 3. Estilos de cabecera Content
* 4. Estilos de página Contenido
* 4 bis. Índice de la página de contenido
* 4b. Artículo página de contenido
* 5. Estilos de pie de página de contenido
* /
```

Tipos de Selectores

Utilice los selectores para seleccionar los elementos de una página a los que desea aplicar ciertas propiedades. Los elementos del documento puede ser seleccionados en base a la etiqueta HTML utilizada, en base a los atributos de clase o ID, a partir de la relación con otros elementos, o en base al estado actual en el documento. También puede combinar selectores simples para formar una cadena de condiciones que deben cumplirse antes de que la regla de estilo sea aplicada.

E(Selectores de tipo)

El selector de tipo selecciona un elemento de tipo E (E es cualquier etiqueta HTML).

```
h1 {} / * selecciona todos los elementos h1 * /
form {} / * selecciona todos los elementos del formulario * /
```

*(Selector Universal)

El selector universal selecciona cualquier tipo de elemento en el documento. Es implícito si hay una secuencia de otros selectores simples y no se específica tipo de selector.

```
* {} / * Selecciona todos los elementos de un documento * /
*. thumb {} / * selecciona todos los elementos con el pulgar clase
[Ver Class Selector abajo] * /
. thumb {} / * mismo que el anterior, * se presupone * /
```

Id (ID Selector)

El selector de ID equivale a cualquier elemento con el valor especificado en su atributo id.

```
#Header {} / * selecciona el elemento con el ID de cabecera * /
*#Header {} / * mismo que el anterior * /
div #footer {} / * selecciona el elemento div con clase * pie de
página */
```

. Class (Selector de clases)

El selector de clase equivale a cualquier elemento con el nombre de clase especificado. Para elementos cuyo atributo clase contiene múltiples palabras separados por espacios, el selector de clase seleccionará al elemento si cualquiera de esas palabras coincide con el nombre de la clase especificada.

```
.help {} / * equivale a todos los elementos con una clase help * /
img.thumbnail {} / * Elementos de imagen con clase thumbnail * /
```

Selectores de atributos

Además de atributos de identificación y clase, cualquier atributo se puede utilizar para la selección a través de los selectores de atributos.

[Att]

Selecciona los elementos con el atributo att, independientemente del valor del atributo.

```
input [required] {} /* elementos html5 de entrada con el atributo
required */
```

[Att = val]

Selecciona los elementos con el atributo att con el valor igual a val.

```
a [rel = tag] {} / * etiquetas a con el atributo rel igual a tag * /
```

[Att ~ = val]

Selecciona los elementos con el atributo att cuyo valor incluye la palabra val en su espacio delimitado por la lista de palabras. (Piense en varios nombres de clase.)

```
a [rel~=friend] {} / * anclas con derechos de autor como un de muchas palabras en el atributo rel. Por ejemplo, los XFN rel = "met amigo". * /
```

[Att | = val]

Selecciona los elementos con el atributo att cuyo valor es igual a val o comienza con val seguido por el separador -. Esto está destinado a ser utilizado para de acuerdo con el subcódigo de idioma para el atributo hreflang.

```
* [Hreflang|=es] {} / * coincide con todos los elementos con hreflang en, es-es, en au-y en-GB * /
```

[^ Att = val]

Selecciona los elementos con el atributo att cuyo valor comienza con val. Añadido en CSS3.

```
a[href^=http] {}  / * coincide con todos los enlaces que comienzan con el texto "Http" * /
```

[Att $ = val]

Selecciona los elementos con el atributo att cuyo valor termina con val. Añadido en CSS3.

```
a[href$=.pdf] {} /* coincide con todos los enlaces a los archivos PDF */
```

[* Att = val]

Selecciona los elementos con el atributo att cuyo valor contiene el subcadena val en cualquier lugar dentro de ella. Añadido en CSS3.

```
input[id*=phone] {} /* coincide con todos los campos de entrada con la palabra phone como parte de la ID */
```

Aunque los atributos ID y clase se pueden seleccionar a través del selector de atributo, los selectores específicos han sido optimizados por los proveedores de navegadores y se deben utilizar en su lugar.

Selectores de pseudo-clase

Las pseudo-clases se dividen en dos grupos: pseudo-clases dinámicas que representan un estado específico en que el documento o elemento se encuentra (p.e. enlaces visitados) y pseudo-clases estructurales que representan información acerca de la posición de un elemento en la estructura del documento (p.e. el primero lista de elementos).

:link, :visited (pseudo-clases de enlace o link)

Los enlaces en un documento tienen dos estados. Un enlace puede ser seleccionado con la pseudo-clase :link, o se puede seleccionar a través :visited cuando ha sido visitado. Esto se determina por el navegador y su historial.

```
a: link {color: blue;}
a: visited {color: purple;}
```

:hover, :active, :focus (pseudo-clases de acción)

Los tres estados de interacción: ponerse encima (MouseOver), activo que determina que se ha pulsado o usado, y de enfoque, pueden ser seleccionados independientemente mediante las pseudo-clases :hover, :active, y :focus respectivamente.

La pseudo-clase :hover para elementos nonlink (<A> sin atributo href) no se admite en IE6 o anterior, pero es compatible con los enlaces de IE4.

Regla Mnemotecnica "The LoVe/HAte of Hyperlink LVHA"

Los enlaces pueden estar en varios estados al mismo tiempo, y todas las pseudo-clases tienen idéntica especificidad, por lo que el orden en el que las pseudo-clases son definidas es importante. El último que se aplica en un momento dado tendrá prioridad.
En el momento en que se hace clic en un vínculo visitado anteriormente coincidirán los las cinco pseudo-clases. Si a:visited es el último selector que se ha definido, entonces esas propiedades se aplicarán, y diferentes propiedades definidas para :active y :hover nunca se verán.
El recurso mnemotécnico LoVE/ HAte es una manera útil de recordar el orden correcto de los selectores.

: Target (Target Pseudo-clase)

El pseudo-clase target representa el elemento objetivo de una llamada en una URI de un link como top en http://example.com/index.html#top (Ya sea por ID, o un atributo name en el caso de HTML4).

:enabled, :disabled, :checked (UI Pseudo-classes)

Las pseudo-clases :enabled, :disabled, and :checked se utilizan para seleccionar los elementos de formulario en sus posibles estados. Cualquier elemento de formulario puede estar habilitado o deshabilitado, normalmente determinado por la presencia o ausencia de un atributo HTML disabled. Las casillas de verificación y radio botones pueden también estar :checked a través de la interacción del usuario o mediante el atributo HTML checked.
La especificación "CSS3 Basic User Interface" ha definido selectores para los siguientes estados adicionales de interfaz de usuario :default, :valid, :invalid, :required, :optional, :in-range, :out-of-range, :read-only, and :read-write. Estos selectores permiten el diseño de diversos estados disponibles con elementos de formulario HTML5.

:lang ()

La pseudo-clase lang permite la selección basada en el lenguaje de el texto.

```
article: lang (en) {} /*selecciona un elemento artículo en Inglés*/
article: lang (es) {} /*selecciona un elemento artículo en español*/
```

:root

La pseudo-clase root es un atajo para seleccionar el nodo raíz de un documento. Para documentos HTML, éste es siempre el elemento html.

:nth-child(),:nth-last-child()

Estos selectores de pseudo-clase localizan elementos que aparecen en una determinada posición en una lista de elementos que contienen ese elemento de lista y todos sus hermanos. La posición se define a través del patrón an+b donde a y b son números enteros.
Los valores de odd y even son atajos para seleccionar cada elemento impar (2n +1) o par (2n).

```
tr {background-color: #FFF; color: #000;}
tr:nth-child (odd) {background-color: #AAA;}
```

1	2
3	4
5	6
7	8
9	10

:first-child,:last-child

Los pseudo-selectores :first-child y :last-child representan el primer y último hijo de algún elemento padre. Son equivalentes a :nth-child (1) y :nth-last-child (1), respectivamente.
Estos selectores pueden ser extremadamente útiles para generar diseños en listas o tablas con las propiedades de borde.

```
li {border-top: 1px solid red;} / * Colocar el borde superior de los
elementos li* /
li:first-child {border-top: none;} / * quitar el borde
de la parte superior del primer elemento * /
li {border-bottom: 1px solid red;} / * Colocar borde superior todos
los elementos li * /
li:last-child {border-bottom: none;} / * quitar el borde de la parte
inferior del último elemento * /
```

- Item 1
- Item 2
- Item 3
- Item 4
- Item 5
- Item 6

:nth-of-type(), :nth-last-of-type()

Mientras que nth-child () coincide con cualquier tipo de elemento, nth-of-type(), y nth-last-of-type() seleccionan cada an+ b elementos donde el conjunto de elementos se basa en una colección de elementos del mismo tipo independientemente de qué tipo de contenido está entre ellos.

Una aplicación del selector n-ésimo de tipo es para establecer estilos alternados en imágenes sucesivas que se encuentran en un elemento primario, independientemente de qué tipo de contenido está entre ellos.

:first-of-type, :last-of-type

Igual que :first-child y :last-child permiten acceder de forma rápida al primer y último hijo de un determinado tipo,y representan a :nth-of-type(1)y :nth-last-of-type(1), respectivamente.

:only-child

Esta pseudo-clase selecciona los elementos que son el único hijo de su padre. En este caso, cualquier elemento seleccionado por :only-child coincide con los selectores :first-child o :last-child.

:only-of-type

Esta pseudo-clase coincide con un elemento si es el único elemento de su tipo entre sus elementos del mismo nivel. Es decir, que coincide tanto los selectores la :first-of-type y :last-of-type.

:empty

Esta pseudo-clase selecciona elementos sólo cuando no contienen nodos secundarios. Esto incluye los nodos de texto y caracteres en blanco.

```
td:empty {background-color: rgb(80%, 80%, 80%);} / * fondo
celdas vacías de tabla * /
```

:not()

La pseudo-clase negación le permite seleccionar aquellos elementos que no coinciden con el selector.

```
p:not(.note) {} /* paragraph elements without the class note */
input:not(:required) {} /* optional input fields */
td:not(:nth-child(odd)) {} /* not odd, or the equivalent of
:nth-child(even) */
```

Selectores pseudo-elemento

Los pseudo-elementos son elementos ficticios que no aparecen en el documento HTML sino que representan una parte del documento que el navegador superpone en la estructura del documento para representar propiedades del diseño. Por ejemplo, debido a diferencias en los tamaños de fuente, longitudes de línea, y los dispositivos, no se puede ajustar el texto que aparecerá en la primera línea de un párrafo de texto en una etiqueta con mucha precisión. Sin embargo, los navegadores colocarán un pseudo-elemento alrededor de la primera línea de texto como si tu lo hicieras.

CSS1 y CSS2 define los pseudo-elementos first-letter, first-line, before, and after con dos puntos (:). Para distinguir mejor pseudo-elementos de pseudo-clases, la especificación CSS3 ha cambiado esto a cuatro puntos (::).
Para los cuatro pseudo-elementos originales, los navegadores deberían soportar ambas sintaxis. El otro pseudo-elemento (::selection) sólo se puede escribir con los dos puntos dobles (::).

:: First-letter

Este pseudo-elemento selecciona la primera letra de un bloque o bloques como (table-cell, table-caption, inline-block, list-item). Si los elementos de bloque están anidados, la primera letra del texto será igualado por ambos elementos.

```
div :: first-letter {font-weight: bold;}
p :: first-letter {color: blue;}
[...]
párrafo <div> <p> Primero. </ p> <p> Párrafo segundo. </ p> </ div>
```

Dado este ejemplo, la P en la palabra Primero será seleccionado por ambas normas y será negrita y azul, mientras que la P en segundo lugar sólo será azul.

Un efecto de letra capital puede lograrse fácilmente mediante la aplicación de un left float y un aumento de tamaño de la fuente a la primera letra de un bloque.

::first-line

El pseudo-elemento ::first-line selecciona la primera línea de un bloque o bloques como elemento después de que el bloque ha sido formateado según su contexto.

```
p::first-line { text-transform: uppercase; }
```

::before, ::after

Estos pseudo-elementos se utilizan para seleccionar el contenido generado antes o después de un elemento. Este contenido hereda las propiedades del elemento al que está unido. El siguiente ejemplo colocaría el texto NOTE: antes de cualquier instancia de p.note.

```
p.note::before {
content: “NOTE: “;
font-weight: bold;
}
```

::selection

Este pseudo-elemento representa un elemento virtual que envuelve cualquier texto seleccionado por un visitante. Con este pseudo-elemento, se puede establecer un número limitado de propiedades que incluyen el color y el color de fondo del texto.

```
::selection { color: black; background-color: yellow; }
```

El pseudo-elemento selección se ha eliminado del módulo "CSS3 Selectors Recomendación", pero se implementa en algunos navegadores como ::selection en Firefox y otros navegadores Gecko ::moz-selection.

Combinadores de selectores

Los selectores descritos hasta este punto seleccionan elementos sobre la base de un tipo de elemento o las propiedades de un elemento. Las combinaciones se utilizan para combinar estos selectores simples en la forma en la que se encuentran dentro de la estructura del documento.

E F (Combinacion/relación descendiente)

Representado por espacios en blanco, el selector descendiente describe una elemento (F) que está contenido dentro de otro elemento (E). El número de elementos entre los antepasados y descendientes no importa. Como ejemplo, el siguiente selector coincidirá con todos los elementos dentro del cuerpo:

```
body * {}
```

E> F (Combinacion/relación hijo)

El selector hijo describe un elemento (F), que es el hijo (o descendiente directo) de otro elemento (E).

```
ul> li {font-family: san-serif;}
ol> li {font-family: monospace;}
```

En este caso, los elementos de lista que son hijos directos de una lista desordenada será del tipo de letra sans-serif, y los que son hijos directos de una lista ordenada será de tipo monospace.

E + F (hermanos adyacentes)

Los elementos combinatorios hermanos adyacentes seleccionan (F) que vienen directamente después de otros elementos (E) y comparten el mismo elemento padre.

```
h1 + p {font-size: 1.2em}
```

En este ejemplo, se seleccionarán el párrafo que vienen directamente después de cualquier <h1> y se les aplica un tamaño de fuente mayor.

E ~ F (Combinacion/relación hermanos en general)

Con este selector, el elemento (F) se selecciona si aparece en algún momento después de su elemento hermano (E).

```
#t2 h5~p {color:yellow; background: blue;};
```

Selecciona cualquier elemento p que sigue a un elemento h5 que es descendiente de cualquier elemento con un atributo id que equivale a t2.

La combinación y encadenamiento de selectores

Los siguientes ejemplos muestran la potencia de estos encadenamientos:

```
nav ul li:first-child {}
.vevent h2+* {}
ul ul ul ul>li {}
#page .hentry a.entry-title:link {}
div#main form#registration fieldset input[type=”text”]:invalid
{}
```

Herramienta para traducir selectores

Si alguna vez no está seguro de lo que un selector hace, la herramienta en línea SelectOracle (http://gallery.theopalgroup.com/selectoracle/) traduce los selectores CSS2 y CSS3 en Inglés (o español).

Especificidad

La especificidad es algo que todos los autores de CSS deben comprender y tener en cuenta. Puede considerarse una medida de cuán específico es el selector de una regla. Un selector de especificidad baja puede dar como resultado muchos elementos (como *, que da como resultado todos los elementos del documento), mientras que un selector con una especificidad elevada puede que sólo dé como resultado un único elemento de una página (como #nav, que sólo da como resultado el elemento con una id de nav).
La especificidad de un selector puede calcularse fácilmente. Si dos o más declaraciones entran en conflicto por un elemento determinado y todas las declaraciones tienen la misma importancia, la de la regla con el selector más específico será la que "gane".
La especificidad tiene cuatro componentes; por ejemplo a, b, c y d. El componente "a" es el más distintivo y el "d", el que menos.

1. El componente "a" es bastante sencillo: es 1 para una declaración en un atributo style; si no, es 0.

2. El componente "b" es el número de selectores de id en el selector (los que empiezan con #).

3. El componente "c" es el número de selectores de atributo, incluidos los selectores de clase y pseudoclases.

4. El componente "d" es el número de tipo de elementos y pseudo-elementos del selector.

Así, después de contar un poco, podemos unir estos cuatro componentes para conseguir la especificidad de cualquier regla. Las declaraciones de CSS en un atributo style no tienen selector, de manera que su especificidad siempre es 1,0,0,0. y es por ello que tambien se calcula considerando solo los tres últimos valores como a,b,c.

Veamos unos cuantos ejemplos que nos ayudarán a aclarar cómo funciona este proceso.

Selector	a	b	c	d	Especificidad
h1	0	0	0	1	0,0,0,1
.foo	0	0	1	0	0,0,1,0
#bar	0	1	0	0	0,1,0,0
html >head+body ul#nav *.home a:link	0	1	2	5	0,1,2,5

Pasemos ahora a comentar el último ejemplo con más detalle. Se obtiene a=0 porque es un selector, no una declaración de un atributo style. Hay un selector ID (#nav), de manera que b=1. Hay un selector de atributos (.home) y una pseudoclase (:link), por lo tanto, c=2. Hay cinco tipos de elemento (html, head, body, ul y a), de manera que d=5. Por lo tanto, la especificidad final es 0,1,2,5.

Hay que mencionar que los combinadores (como >, + y el espacio en blanco) no afectan a la especificidad de un selector. El selector universal (*) tampoco cuenta para calcular la especificidad.También hay que tener en cuenta que existe una gran diferencia de especificidad entre un selector id y un selector de atributos que por casualidad haga referencia a un atributo id. Aunque den como resultado el mismo elemento, tienen especificidades muy diferentes. La especificidad de #nav es 0,1,0,0 y la especificidad de [id="nav"] es sólo 0,0,1,0.
Selectores de negación dentro de la pseudo-clase se cuentan como cualquier selector.

Un ejemplo un poco mas complejo sería este (no consideramos style):

```
#user { color: rgb(1,1,1); } /* a=1 b=0 c=0 */
#user:first-child { color: rgb(2,2,2); } /* a=1 b=1 c=0 */
div div+div div div+div div div div>div div div:first-child {
color: rgb(3,3,3);
} /* a=0 b=0 c=13 */
div.foo { color: rgb(4,4,4); } /* a=0 b=1 c=1 */
.foo .foo+.foo .foo+.foo .foo .foo .foo .foo>.foo .foo
➥ .foo:first-child {
color: rgb(5,5,5);
} /* a=0 b=12 c=1 */
.foo.bar { color: rgb(6,6,6); } /* a=0 b=2 c=0 */
[id=”user”] { color: rgb(7,7,7); } /* a=0 b=1 c=0 */
```

Si todos estos selectores se aplican al mismo elemento de texto en un documento HTML , el texto en ese elemento sería el color rgb (2,2,2).

Importancia

La importancia de una declaración de CSS depende de dónde se ha especificado. Las declaraciones contrapuestas se aplicarán en el orden siguiente: las nuevas anularán a las más antiguas.

1. Hoja de estilos de agente de usuario.
2. Declaraciones normales en hojas de estilo de autor.
3. Declaraciones normales en hojas de estilo de usuario.
4. Declaraciones importantes en hojas de estilo de autor.
5. Declaraciones importantes en hojas de estilo de usuario.

Cuando hablamos de "hojas de estilo", normalmente hacemos referencia a una hoja de estilo de autor. Es la hoja de estilos que ha creado o enlazado el autor del documento (o, más probablemente, el diseñador de la web).

Las declaraciones normales son exactamente lo que su nombre indica: declaraciones normales. Lo contrario son las **declaraciones importantes**, que son las declaraciones que van seguidas de una directiva `!important`.

Como se puede observar, las declaraciones importantes de una hoja de estilo de usuario tienen prioridad sobre todas las demás, lo cual es lógico. Siguiendo el ejemplo de la persona disléxica, podría ser que quisiera ver todo el texto con Comic Sans MS en caso de que le facilitara la lectura. Entonces podría tener una hoja de estilo de usuario con la regla siguiente:

```
* {
   font-family: "Comic Sans MS" !important;
}
```

En este caso, independientemente de lo que haya especificado el diseñador, e independientemente de aquello que se haya establecido como familia de fuentes predeterminada del navegador, todo se mostrará con Comic Sans MS. El método de presentación por defecto del navegador sólo se aplicará si las declaraciones no quedan

anuladas por alguna regla de una hoja de estilo de usuario o una hoja de estilo de autor, ya que la hoja de estilo de agente de usuario tiene precedencia menor.

En realidad, la mayoría de los diseñadores no se deben preocupar demasiado de la importancia porque no se puede hacer nada al respecto. No hay ninguna manera de saber si un usuario tiene una hoja de estilos de usuario definida que anule nuestro CSS. Y si la tiene, seguramente sea por alguna buena razón. Aun así, es útil saber qué es la importancia y cómo puede afectar a la presentación de nuestros documentos.

Compatibilidad con navegadores para selectores

Selectores como tipo, identificación, clase y enlaces son compatibles en todos los navegadores. Peter-Paul Koch mantiene algunas tablas de soporte útiles para los selectores para los navegadores de escritorio (Http://www.quirksmode.org/css/contents.html) y para los navegadores móviles (Http://www.quirksmode.org/m/css.html).

Agrupación de selectores

Selectores diferentes pueden estar separados por una coma lo que produce que se asignen las mismas propiedades a todos. Esto le permitirá reducir el uso repetido de la mismas propiedades, mantener los diseños más uniformes, y hacer los cambios y pequeños retoques más fácilmente.

```
p, blockquote, ul, li, dl {
padding: 0;
margin: 0 1em 1.5em;
}
```

Esta declaración selecciona muchos elementos de bloque que pueden haber en la áreas de contenido del documento para que tengan el mismo espacio en blanco.