APRENDERAPROGRAMAR.COM

CONCEPTO DE CASCADA Y HERENCIA CSS. ¿QUÉ ES? ESTILOS DE USUARIO. IMPORTANCIA !IMPORTANT. EJEMPLOS. (CU01017D)

Sección: Cursos
Categoría: Tutorial básico del programador web: CSS desde cero

Fecha revisión: 2029

Resumen: Entrega nº17 del Tutorial básico: “CSS desde cero”.

Autor: César Krall

CASCADA EN CSS

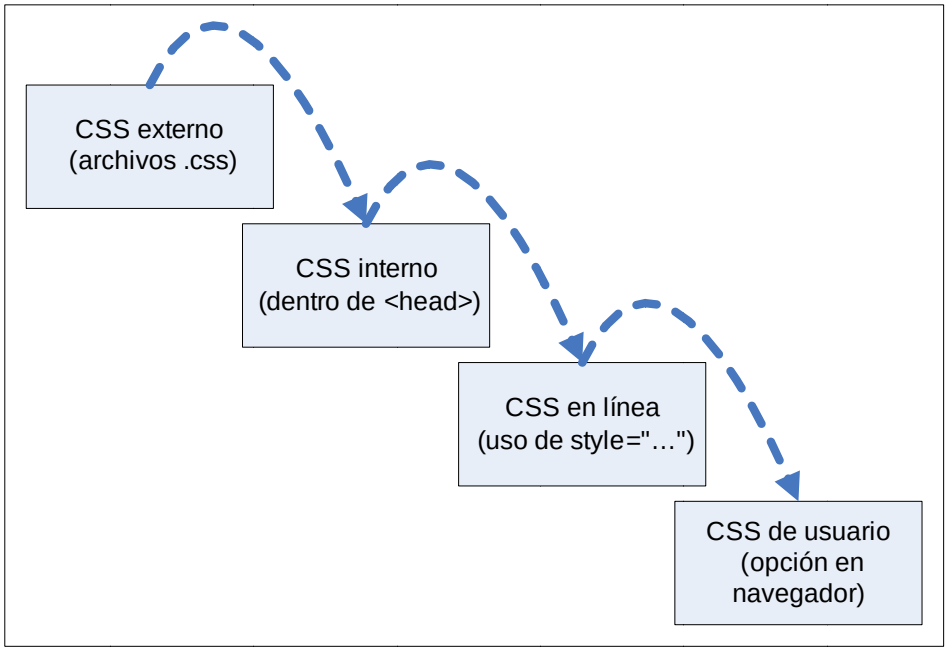
En epígrafes anteriores del curso hemos visto cómo funciona la herencia CSS y cómo los elementos de rango inferior heredan las propiedades CSS de elementos de rango superior. Por ejemplo un párrafo … hereda las propiedades de color definidos para <body> … </body>. Sin embargo, algunas propiedades no se heredan y en algunas propiedades como el color de los link apreciábamos peculiaridades.

Una circunstancia con la que nos tendremos que familiarizar es la existencia de conflictos entre las declaraciones CSS. Por ejemplo si estamos trabajando con un archivo CSS externo podemos incluir en él una declaración p {font-color: blue;}. Pero podría suceder que como CSS interno entre las etiquetas <head> … </head> se encontrara una declaración p {font-color: red;}. Y también podría suceder que en un párrafo concreto dentro del documento HTML apareciera una declaración como <p style = "font-color: yellow;". Estas declaraciones suponen un conflicto o "colisión de estilos" para el navegador, que ha de decidir qué estilo aplicar.

Ya comentamos que de forma general el navegador aplica el criterio de precedencia:

Declaración en línea > Declaración interna > Declaración externa

Esto podemos verlo como una cascada de forma que el navegador va recorriendo la cascada hasta llegar al nivel más bajo posible.



Hemos incluido en el esquema la opción "CSS de usuario" para reflejar que algunos navegadores permiten definir al usuario estilos propios.

ESTILOS DE USUARIO CSS

A través de una opción de menú de algunos navegadores, se puede definir por ejemplo que el texto se debe mostrar de un tamaño determinado o de un color determinado. Cuando el usuario elige estas opciones normalmente anula aquello que ha sido definido por el autor de la página web, de modo que sus opciones prevalecen respecto a cualquier otro tipo de declaración. Esta situación no es habitual, de hecho quizás la mayoría de los usuarios la desconocen o no la usan. Su empleo quizás se limita a personas con problemas de vista para poner tamaños de fuente grandes o colores específicos que les faciliten la visión, o a usuarios avanzados que tienen un gran dominio de las opciones de configuración. A pesar de su relativo poco uso, como programadores web debemos tener constancia de ello porque es posible que tengamos que resolver problemas donde esta configuración afecte a la solución del problema.

Aprovechando que hablamos de opciones de configuración de los navegadores, citaremos también que algunos navegadores permiten configurar opciones relativas a los estilos por defecto. Ya hemos dicho que cuando un documento HTML carece de estilos aplicados, en realidad sí tiene lo que podríamos denominar estilos básicos o estilos por defecto que aplica el navegador. Por ejemplo, una navegador recién instalado puede tener un tamaño de fuente por defecto de 12 pixeles, pero en aquellos navegadores que lo permiten, este parámetro podría cambiarse y establecerse en 16 pixeles a través de las opciones de configuración del navegador. También algunos navegadores permiten configurar otros parámetros adicionales. Este es uno de los motivos por los que es recomendable no confiar en los estilos por defecto cuando se crean páginas web, sino especificar todos los parámetros desde nuestra hoja de estilos para asegurarnos de que se aplican exactamente las reglas CSS que nosotros deseamos.

ORIGEN, IMPORTANCIA Y ESPECIFICIDAD

Nos encontramos con que existen distintos tipos de CSS como el predefinido por el navegador, el definido por el autor de la página web (que puede ser externo, interno o en línea) ó el definido por el usuario en la configuración de su navegador. A su vez, las reglas pueden entrar en conflicto (que una regla diga una cosa y otra regla una cosa distinta) y no sólo a nivel navegador – autor – usuario, sino en lo que para nosotros es la definición CSS principal, la que escribimos cuando creamos una web. Por ejemplo en un archivo css externo podemos tener una regla body {color: red;} y otra regla p { color: blue;} y otra regla #destacado {color: yellow;}. Si en el documento HTML nos encontramos algo así como … ¿Qué estilo aplica el navegador entre los tres posibles?

Todas estas situaciones se denominan situaciones de conflicto o colisión y CSS define una forma de resolución de conflictos o colisiones. A este mecanismo de resolución de conflictos se le denomina “cascada”. La cascada se basa en determinar todas las reglas que son de aplicación a un elemento y que entran en conflicto, en ordenar esas reglas en base a unos criterios y mostrar como resultado en el navegador la regla que haya quedado en primer lugar, a la que denominamos “regla ganadora”.

El mecanismo es similar a la asignación de “pesos” a cada regla CSS para lograr la ordenación. Supongamos que la regla body {color: red;} tiene peso 45 sobre 100 para ser aplicada a los párrafos que la regla p { color: blue;} tiene peso de 55 para ser aplicada a los párrafos y que la regla #destacado {color: yellow;} tiene peso 70 para ser aplicada a un párrafo con id = “destacado”. Cuando el navegador

se encuentra un párrafo dentro del documento HTML ve las reglas que le son de aplicación y el peso de cada una, aplicando como regla ganadora la que mayor peso tenga.

¿Cómo se determinan los pesos y el orden que se asigna a las reglas, es decir, cómo se determina qué regla "gana" cuando hay varias reglas que entran en conflicto?

Las reglas de resolución de conflictos son complejas y no todos los navegadores responden de la misma manera. Por ello en cursos de aprendizaje de CSS no recomendamos un estudio en profundidad, sino conocer las reglas básicas y la filosofía del procedimiento de cascada.

El navegador sigue un proceso similar al siguiente para la resolución de colisiones de estilos:

1) Crea una lista con todas las reglas que son de aplicación directa o indirecta por herencia a un elemento concreto. Cuanto más directa o próxima esté la regla al elemento, antes se coloca. La proximidad viene dada por cuán próxima está la regla a una definición directa del elemento afectado. Por ejemplo si un elemento div con id="menu1" tiene un párrafo con class="items" entre #menu1 {...} y .items {...} la regla ganadora a la hora de aplicar estilo al párrafo es .items { ... } porque es la que más directamente define el elemento. En este ejemplo menu1 define al elemento indirectamente a través del div, mientras que items define directamente la clase del párrafo y por ello resulta ganadora.
   Otro ejemplo: para un párrafo dentro de un div, una regla que hereda de div está más próxima que una regla que hereda de body. Si existiera una regla directa relativa a los párrafos, esta iría en primer lugar antes que las demás debidas a herencia.
   Si existe una sola regla directa relativa al elemento que se está evaluando, se aplica y termina el proceso de cascada.

2) Si existe más de una regla con igual nivel de proximidad, se dividen las reglas por origen en base a si se trata de estilos de usuario, predefinidos del navegador, estilos de autor, y en caso de estilos de autor si son css externo, interno o en línea. Se ordenan en base al origen. Básicamente se colocan primero las declaraciones en línea, luego las internas y luego las externas.

3) Se dividen las declaraciones de reglas en declaraciones importantes y declaraciones normales. Las declaraciones importantes son las que están escritas como:

nombreElemento {propiedad: valor **!important**;}

Si dos declaraciones son iguales y una lleva !important y la otra no, prevalece la que lleve !important incluso aunque la otra aparezca con posterioridad. Por ejemplo si en un archivo css escribimos:

```
div {color:blue !important;}

div {color: pink; }
```

Podría pensarse que los elementos div tendrán color rosa por aparecer esta regla en último lugar y sobreescribir a la anterior. Sin embargo, no ocurre así porque la palabra clave !important hace que esa declaración prevalezca sobre otra igual que no lleve la palabra clave !important. El significado de !important en este caso sería similar a "esta regla sobreescribe a cualquier otra que defina la misma propiedad para el mismo elemento y no es sobreescribible".

Si en un archivo css externo tenemos la declaración p {color:blue !important;} y en el documento HTML tenemos en línea … la palabra clave !important anulará la precedencia del origen, de modo que el texto se verá de color azul. El significado de !important quedaría ampliado a "esta regla no es sobreescribible y anula la precedencia debida al origen".

Algunos navegadores antiguos no reconocen la palabra clave !important. Entre los navegadores modernos, no todos aplican exactamente el mismo significado a la presencia de esta palabra clave.

Teniendo en cuenta el origen y si existe o no la palabra clave !important, las reglas se ordenan situándose en primer lugar las declaraciones de estilos en línea, excepto cuando una regla de aplicación directa al elemento lleva la palabra clave !important. Si existe una regla ganadora, se aplica.

4) Si no existe una regla ganadora, para las reglas en conflicto se valora un parámetro denominado especificidad. Estaríamos en el caso de que existan reglas que definan directamente un elemento y que entren en conflicto. Por ejemplo estas dos reglas:

   body div.destacado p {color: cyan;}
   div.destacado p {color: yellow;}

Definen directamente el estilo para párrafos que se encuentren dentro de elementos div cuyo atributo class sea "destacado". No hay una regla entre las dos que defina más directamente el valor de la propiedad para ese tipo de elemento, por lo que hay que proceder a una resolución de conflicto. Esta resolución se hace con el procedimiento de cálculo de especificidad que explicaremos más adelante. La regla con mayor especificidad será la que se mostrará al visualizar la web.

Otro ejemplo de conflicto sería en un párrafo con id="atelier" y esta declaración en un archivo css:

```
p {color: red; text-decoration:none;}
#atelier {color: yellow;}
```

Ambas reglas son de aplicación directa al párrafo, por lo que la regla ganadora se determina mediante el cálculo de especificidad.

5) Si se llega a que dos reglas tienen igual especificidad, igual importancia e igual origen y se contradicen, prevalecerá la que esté escrita en último lugar (criterio de orden o de sobreescritura).

EJERCICIO

En el archivo CSS externo a un documento HTML encontramos esta declaración:

```
.magicTitle {
margin:0px;
width:20px;
height:30px;
color:green !important;
border-style: hidden !important;
}
```

Por otro lado dentro del código HTML tenemos el siguiente código:

```
<div class="magicTitle" style="border:1px solid blue; color: blue;">
Aprendiendo a programar
</div>
```

Responde a las siguientes preguntas:

a) ¿Se mostrará un borde para el div? ¿Por qué?

b) ¿Con qué color se mostrará el texto, con color verde o con color azul? ¿Por qué?

c) ¿Se respetarán los valores de width y height definidos en el archivo CSS externo o quedarán anulados? ¿Por qué?

Para comprobar si tus respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

Próxima entrega: CU01018D

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203