APRENDERAPROGRAMAR.COM

CASCADA DE ESTILOS. CÁLCULO DE ESPECIFICIDAD. CÓMO USAR !IMPORTANT EN CSS. EJERCICIOS RESUELTOS (CU01018D)

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

Fecha revisión: 2029

Resumen: Entrega nº18 del Tutorial básico: “CSS desde cero”.

Autor: César Krall

CÁLCULO DE LA ESPECIFICIDAD

El mecanismo de cascada CSS determina que cuando diferentes reglas son de aplicación a un elemento éstas se ordenan en base a unos criterios. Si con el criterio de proximidad o de origen no se ha podido resolver el conflicto entre reglas se valora lo que se denomina especificidad. La especificidad para reglas que aplican igual de directamente a un elemento es un valor numérico que utiliza el navegador para ordenar reglas que entran en conflicto.

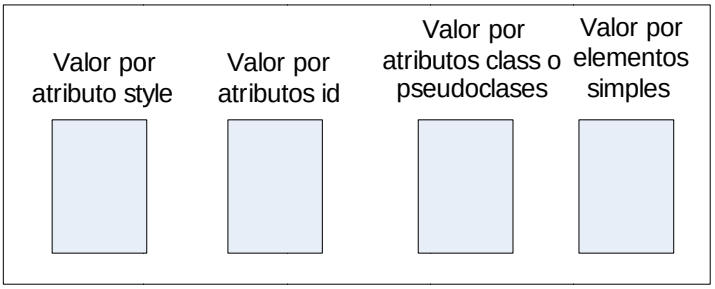
La especificidad se puede calcular como un número que consta de cuatro dígitos ABCD en el cual tenemos:

A o primer dígito: toma valor 1 cuando el estilo se declara en línea o cero en caso contrario.

B o segundo dígito: se calcula sumando 1 por cada identificador de tipo id que afecte a un elemento. Si una declaración es #menu1 #item1 {...} el valor del segundo dígito de especificidad para esta regla será 2, resultado de sumar 1+1, una unidad por cada id que afecte al elemento.

C o tercer dígito: se calcula sumando 1 por cada clase o pseudoclase que afecte a un elemento. Por ejemplo .destacado { ... } aporta un valor 1, mientras que .destacado .especial .suiter { ... } aporta un valor 3 resultado de sumar 1+1+1, una unidad por cada clase o pseudoclase.

D o cuarto dígito: se calcula sumando 1 por cada elemento HTML o pseudoelemento que aparezca en la declaración. Por ejemplo ul li a { ... } aporta un valor 3, resultado de sumar 1+1+1, una unidad por cada elemento HTML referenciado.



Una vez determinado cada dígito, se obtiene un valor numérico (por ejemplo 0112) que podemos ver como el peso de la regla. “Gana” la regla con mayor peso. Es interesante fijarse en que el uso de style siempre ganará a cualquier combinación de reglas, lo cual nos dice que habitualmente los estilos en línea ganarán. Una web bien construida debe prescindir en general del uso de estilos en línea, aunque comprobarás que por un motivo u otro es frecuente encontrarlos cuando se analizan desarrollos web existentes. Después de los estilos en línea, los estilos definidos para un id resultan ganadores respecto al resto. Finalmente, las clases o pseudoclases ganan a la definición de estilos para elementos simples HTML.

Como vimos anteriormente, la palabra clave !important puede introducir excepciones.

EJERCICIO RESUELTO

Supongamos que tenemos el siguiente código HTML:

```
<div> <!—Ejemplo aprenderaprogramar.com-->
<div class="destacado">
<p> Aprender a programar es un objetivo que se plantea mucha gente y que no todos alcanzan.</p>
... </div> ... </div>
```

Si en un archivo css tenemos las siguientes declaraciones, determinar cuál es la puntuación por especificidad cuando proceda, cuál es la regla ganadora, de qué color se visualizará el texto y por qué:

Declaraciones	Puntuación especificidad	Regla ganadora, color del texto y por qué
body {color: grey;} body div.destacado p {color: cyan;} .destacado {color: green;} div.destacado {color: blue;} div.destacado p {color: yellow;} div:first-child {color: magenta;}		
body {color: grey;}		
body {color: grey;} .destacado {color: green;}		
body div.destacado p {color: cyan;} .destacado {color: green !important;}		
body div.destacado p {color: cyan;} .destacado {color: green;} p {color: blue;}		
body div.destacado p {color: cyan;} .destacado {color: green;} p {color: blue !important;}		

SOLUCIÓN AL EJERCICIO PLANTEADO

Nota: no todos los navegadores responden igual a la palabra clave !important, por lo que podrías encontrar algunos navegadores en que la respuesta no fuera exactamente como hemos indicado.

Declaraciones	Puntuación	Regla ganadora, color del texto y por qué
body {color: grey;} body div.destacado p {color: cyan;} .destacado {color: green;} div.destacado {color: blue;} div.destacado p {color: yellow;} div:first-child {color: magenta;}	**No** **0-0-1-3** **No** **No** **0-0-1-2** **0-0-1-1**	1ª regla: no aplica directamente, si hiciéramos el cálculo body aporta 1 al cuarto dígito. Total 0001 2ª regla: body, div y p aportant 1+1+1 = 3 al cuarto dígito y .destacado aporta 1 al tercer dígito. Total 0013 3ª regla: no aplica directamente. Si lo calculáramos, 0010 4ª regla: no aplica directamente. Si lo calculáramos, 0011 5ª regla: div y p aportan 1+1=2 al cuarto dígito y .destacado aporta 1 al tercer dígito. Total 0012 6ª regla: la pseudoclase first-child aporta 1 al tercer dígito y div aporta 1 al cuarto dígito. Total 0011 Gana: la regla 2, el texto se vería de color cyan.
body {color: grey;}	**No**	Gana: la única regla que, por herencia, afecta. El texto se vería de color gris. Si calculáramos la especificidad sería 0001
body {color: grey;} .destacado {color: green;}	**No** **0-0-1-0**	Gana: la segunda regla por ser la más próxima. El texto se vería de color verde.
body div.destacado p {color: cyan;} .destacado {color: green !important;}	**0-0-1-3** **No**	Gana: la primera regla. Aunque la segunda regla lleva la palabra clave !important es menos directa. La primera regla afecta directamente al elemento p por lo que tiene prevalencia.
body div.destacado p {color: cyan;} .destacado {color: green;} p {color: blue;}	**0-0-1-3** **No** **0-0-0-1**	Gana la primera regla, el texto se verá de color cyan.
body div.destacado p {color: cyan;} .destacado {color: green;} p {color: blue !important;}	**0-0-1-3** **No** **!important**	Gana la tercera regla: hay dos reglas que afectan directamente al elemento p, la primera y la tercera. Al tener la tercera la declaración !important significa que se sobreescriben el resto de reglas, independientemente de su valor de especificidad y origen.

CÓMO USAR !IMPORTANT EN CSS

Considera una declaración de este tipo. Pruébala sobre el código HTML de prueba que estamos usando para el curso:

```
body div.destacado p {color: cyan; text-decoration:underline;}
p {color: red !important;}
```

Aquí apreciamos dos cosas no del todo correctas. En primer lugar, por convenio los programadores y diseñadores web suelen poner las declaraciones más generales en primer lugar y las más específicas a continuación, tanto más abajo en el archivo o definición css cuanto más específicas sean. Esto facilita el análisis y comprensión de hojas de estilo. Por tanto cambiaríamos el orden de la declaración y pondríamos en primer lugar la declaración más general relativa a párrafos en general y en segundo lugar la otra declaración, más específica.

En segundo lugar, si queremos que los párrafos sean rojos: ¿Para qué declarar un color de párrafo cyan que luego anulamos con una declaración con la palabra clave !important? En un archivo CSS con cientos de líneas estas declaraciones generan confusión y dificultan el análisis del código. Cuando vemos cosas de este tipo analizando páginas web en general corresponden a que la persona que generó el código no tenía claros los conceptos de CSS ó a que se han realizado correcciones apresuradas en el código dejando inconsistencias. El problema está en que cuando una hoja de estilos se manipula múltiples veces añadiendo en cada ocasión más inconsistencias, se vuelve incoherente e inmanejable.

El código anterior queda más correcto así. Comprueba que obtienes el mismo resultado:

```
p {color: red;}
body div.destacado p {text-decoration:underline;}
```

Sólo debería usarse !important en casos concretos y en los que resulta estrictamente necesario. Usar la palabra clave !important con frecuencia anulando estilos repetidos es síntoma de un mal código CSS.

En general, para personas que se están iniciando con CSS recomendamos no usar !important en el código, posponiendo su uso para cuando se haya adquirido experiencia y un mayor nivel de destreza.

CONCLUSIONES SOBRE LA CASCADA CSS

El orden final con que se ordenan reglas en conflicto se basa en un proceso denominado “cascada”. El proceso es complejo, las reglas van cambiando en el tiempo y pueden existir diferencias entre navegadores. Por ello no resulta de interés aprender todas las reglas del proceso de cascada ni estar realizando continuamente cálculos para determinar precedencias. Sin embargo, sí resulta de interés comprender el concepto y las reglas básicas porque nos puede ayudar a resolver problemas que aparezcan en la visualización de páginas web. Con esto, más la aplicación de sentido común y la experiencia que iremos ganando a medida que trabajemos con CSS, será suficiente para crear páginas web con un código CSS de calidad.

Próxima entrega: CU01019D

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