Cascada y herencia en CSS

Además de los selectores y la delcaración es necesario entender los conceptos de cascada y herencia en CSS, pues su comprersión te ayudará a dominar este lenguaje en su totalidad.

En un principio, ambos conceptos pueden llegar a ser un poco confusos, pero a medida que avancemos con CSS te darás cuenta de que tan sólo se necesita algo de práctica para poder entenderlos e implementarlos con facilidad.

1. ¿Qué es la Cascada?

El hecho de que la palabra cascada forme parte del nombre del lenguaje CSS (*Cascading Style Sheets / Hoja de estilos de cascada*) indica que se trata de un concepto importante. En términos generales, el concepto de cascada se refiere al mecanismo que determina cuáles estilos se deben aplicar cuando un mismo elemento HTML se ve afectado por múltiples reglas.

Por ejemplo, supongamos que en nuestro documento HTML tenemos el siguiente elemento <p>:

```
<p>Esto es un texto de ejemplo.</p>
```

También, supongamos que en nuestro código CSS hemos escrito dos reglas diferentes para aplicar estilos a este elemento:

```
p {
  font-size: 12px;
  color: green
}

p {
  color: white;
  background-color: teal
}
```

Dado que ambas reglas apuntan al mismo elemento, ¿cuál de las dos aplicará el navegador? La respuesta es que en este caso prevalece la última regla, la cuál sobreescribe y mezcla las dos declaraciones, es decir, que el texto contenido en el elemento <p> se mostrará con un color blanco, un color de fondo “teal” y un tamaño de fuente de 12px . El resultado será lo siguiente:

```
p {
  font-size: 12px;
  color: white;
  background-color: teal
}
```

Ahora bien, en algunos casos puede que por el nivel de complejidad de la hoja de estilos no sea tan sencillo determinar cuál es la declaración que debe prevalecer. Y aquí es cuando entra en juego la cascada CSS para aplicar la regla que tiene la prioridad.

En general, al aplicar estilos CSS a un elemento HTML el navegador se comporta de la siguiente manera:

- Selecciona los estilos y los aplica según su **importancia**. Cuanto más importantes, mayor es el peso.

- Si las declaraciones tienen la misma importancia, debe prevalecer la **especificidad del selector**.

- Si la importancia y especificidad son iguales, el navegador laplica los estilos en función de su **orden de aparición en el código CSS**.

Esto quiere decir que la forma de definir la preferencia se realiza sobre la base de tres factores claves:

1. La **importancia** de la regla.
2. La **especificidad del selector**.
3. El **orden de aparición en el código**.

Vamos a ver en detalle en que consiste cada uno de ellos.

Importancia de la regla

La importancia de una regla se determina dependiendo las hoja de estilos en donde ha sido escrita. CSS define varios tipos de hojas de estilo, cuya prioridad es en orden descendente, tal y cómo se muestra en esta tabla:

Tipo de hojas de estilo	Descripción
Estilos del agente de usuario	Se refiere a los estilos CSS que aplican los navegadores de forma predeterminada.
Estilos del usuario	Son los estilos que aplican los visitantes de una página, gracias a que los navegadores permiten modificar ciertas configuraciones.
Estilos del autor	Se trata de los estilos escritos por nosotros, los diseñadores y desarrolladores web.

En caso de que exista algún conflicto entre las reglas de las diferentes hojas de estilo, las últimas prevalecen sobre las primeras. Es decir, que los estilos de autor (los que escribimos los diseñadores y desarrolladores web) prevalecen sobre los estilos de los usuarios, y estos sobre los del navegador.

Ahora bien, CSS permite cambiar este comportamiento por medio de una cadena de texto denominada **!important** con la cual una determinada regla siempre prevalece sobre todas las demás, independientemente del nivel o la altura a la que estén.

Por ejemplo, en el código anterior el navegador aplicó los estilos de la última regla CSS debido a que cuando un mismo elemento se ve afectado por dos reglas diferentes, siempre prevalecerán los estilos de la úktima regla, por tanto el color del texto encerrado dentro del elemento <p> es blanco en lugar de verde. Sin embargo, podemos cambiar este comportamiento a través de la cadena de texto denominada !important:

```
p {
  font-size: 12px;
  color: green !important;
}

p {
  color:white;
  background-color: teal
}
```

En el código anterior, al agregar la cadena de texto !important a la propiedad *color* de la primera regla, esta prevalece sobre la segunda, por tanto, el resultado final sería:

```
p {
  font-size: 12px;
  color: green;
  background-color: teal
}
```

Debes tener en cuenta que la cadena !important solamente se debe utilizar en los casos que sean estrictamente necesarios, por ejemplo, cuando trabajamos con hojas de estilos muy complejas que han sido escritas por otros desarrolladores, y queremos editar ciertos estilos que no se pueden sobrescribir de ningún otro modo. La razón

se debe a que, en ciertos casos, !important puede causar algunos problemas en el depurado de CSS.

> Si la cadena !important aparece declarada en una misma propiedad tanto en los estilos del usuario como en los del autor, los primeros (los estilos del usuario) prevalecerían sobre los últimos.

Especificidad del selector

El segundo factor clave de la cascada CSS es la especificidad del selector, una medida cuya finalidad es la resolución de conflictos entre las reglas de las diferentes hojas de estilo. Este mecanismo determina qué tan específico es un selector o cuántos elementos puede seleccionar.

Como regla general, los *selectores de etiquetas* son poco específicos. Los *selectores class* son más específicos y prevalecen sobre los selectores de etiquetas. Los *selectores id* son más específicos, y por lo tanto tienen mayor peso que los selectores class. Sin embargo, la cadena !important prevalece sobre los selectores id.

La especificidad se calcula sobre la base de los valores de 4 componentes, en el orden siguiente:

- **Componente A:** define el número de estilos en línea aplicados al elemento a través del atributo *style*.
- **Componente B:** considera el número de selectores que son un *id*.
- **Componente C:** define el número de selectores de tipo *class*, o de atributos, así como también aquellos que son una *pseudoclase*.
- **Componente D:** el número de selectores de etiquetas o de *pseudoelementos*.

Si deseas saber si un selector es más específico que otro, tan sólo debes calcular sus componentes. Debes tener en cuenta que, por defecto, cada componente tiene un valor igual a 0.

La siguiente tabla muestra algunos ejemplos de selectores, junto con la especificidad atribuida a cada uno:

Selector	A	B	C	D	Especificidad
div {…}	0	0	0	1	0001
#destacado {…}	0	1	0	0	0100
#destacado div:active {…}	0	1	1	1	0111
div#destacado p {…}	0	1	0	2	0102
body div#destacado + p.especial:active { … }	0	1	2	3	0123

En el primer selector, A = 0 debido a que no es una declaración de estilo en línea, B = 0 dado que no contiene un atributo *id*, C = 0 porque no se trata de un selector *class*, un selector de atributos o una *pseudoclase*, y D = 1 debido a que es un selector de etiquetas. El resultado es una especifidad de 0001.

El último ejemplo es mucho más específico debido a que tiene un id, un class, una pseudoclase y tres selectores de etiquetas.

> El selector universal (*), los combinadores de selectores (+, >, ~, ' ') y la pseudo-clase de negación (:not) no afectan a la especificidad.

Vamos a ver otro ejemplo en el cual una regla prevalece sobre otra. Consideremos el siguiente código HTML:

```
<p>Esto es un ejemplo de <em>especificidad</em> en CSS.</p>
```

Tenemos un elemento <p> que a su vez contiene un elemento . Vamos a crear dos reglas para cambiar el color del texto encerrado dentro del elemento , y además aplicar un fondo y modificar el tamaño de fuente:

```
/* Especificidad: 0002 debido a que contiene dos selectores
de etiquetas (p y em) */

p em {
  background-color: green;
  color: white;
}

/* Especificidad: 0001 debido a que contiene un selector
de etiquetas (em) */

em {
  font-size: 15px;
  color: red;
}
```

El el código anterior, la primera regla es mucho más específica y por tanto la propiedad *color* prevalece sobre la segunda regla. Sin embargo, tanto la propiedad *background-color* de la primera regla como *font-size* de la segunda se aplican sobre texto encerrado dentro del elemento .

Orden de aparición en el código

El tercer factor que determina cuál regla debe prevalecer es el orden de aparición en el código CSS, es decir, en qué lugar del documento esta ha sido escrita.

Esto significa que si dos o más selectores tienen la misma importancia y especificidad, las últimas reglas prevalecen sobre las primeras. Por ejemplo:

```
/* Especificidad: 0101 debido a que contiene un selector id y un selector
de etiquetas */

 h2#destacado {
   font-size: 22px !important;
 }

 /* Especificidad: 0101 debido a que contiene un selector id y un selector
 de etiquetas */

 h2#destacado {
   font-size: 18px !important;
 }
```

Dado que en en el código de arriba existe una igualdad entre la importancia y la especificidad, el navegador aplicará los estilos de la segunda regla debido a que tiene mayor peso.

Lo mismo, si tenemos varios archivos CSS, independientemente del método de inclusión, en caso de conflictos se aplicarán los estilos del último.

2. Herencia

Otro concepto que debes conocer para poder dominar el lenguaje CSS es la herencia, un mecanismo mediante el cual ciertas propiedades pasan de un elemento padre a sus hijos, con lo cual se modifica el valor que tienen por defecto. Obviamente, podemos sobreescribir los valores de los elementos hijos si es necesario.

Supongamos que escribimos una regla para aplicar un tamaño de fuente de 15px al elemento <body> de un documento HTML:

```
 body {
   font-size: 15px;
}
```

Por defecto, el navegador aplicará el tamaño definido en el código anterior a todo el texto contenido en el documento. Si luego agregamos un elemento <section> al código HTML, todo el texto contenido en su interior también se mostrará con el mismo tamaño de fuente de su elemento padre, que en este caso es el <body>. Y esto es bastante útil, pues evita que tengamos que establecer la fuente base para cada elemento por separado.

Sin embargo, debes tener en cuenta que no todas las propiedades CSS se heredan, por ejemplo propiedades como *border*, *margin*, *padding*, etc., no se pasan de los elementos padres a sus hijos.

Valores especiales

Las herencias en CSS se puede manejar mediante tres valores especiales que son comunes a la mayoría de propiedades:

- inherit: permite que un elemento herede de su padre el valor de una propiedad.
- initial: establece el valor que la propiedad tenía inicialmente, es decir, el valor por defecto que aplican los estilos del navegador. Si este no existe, la propiedad adopta el valor de *inherit*.
- unset: Este valor es una combinación de los anteriores, pues hereda el valor de la propiedad del elemento padre, y en caso de no existir, de su valor inicial.