# 28. Herencia y cascada

Tommy Olsson. 26 de septiembre del 2008. Última modificación: 12 de marzo de 2017 (equipo docente del grado de Multimedia). Publicado en: importancia, fuente, orden, importante, especificidad

La herencia y la cascada son dos conceptos básicos en CSS. Se deben comprender bien para utilizar CSS. Por suerte, no son muy difíciles de entender, aunque algunos detalles pueden ser un tanto complicados de recordar.

Ambos conceptos están relacionados, pero son diferentes. La herencia está relacionada con cómo los elementos del etiquetado de HTML heredan propiedades de sus elementos padres (los que los contienen) y los transmiten a sus hijos, mientras que la cascada tiene que ver con las declaraciones de CSS que se aplican a un documento y cómo las reglas contradictorias se anulan o no entre ellas.

En este apartado, trataremos detalladamente estos dos conceptos. Probablemente, la herencia es un concepto más fácil de captar, de manera que empezaremos con éste y después pasaremos a las particularidades de la cascada.

**Nota**

Descargad el código fuente de los ejemplos de este apartado; el fichero "inheritance_cascade_code.zip" contiene el CSS y HTML acabados, además de la plantilla inicial de HTML para que podáis ir trabajando mientras veis los ejemplos.

**Descargad los ejemplos en:** "inheritance_cascade_code.zip"

Los contenidos de este apartado son los siguientes:



## 28.1. Herencia

> La herencia en CSS es el mecanismo mediante el cual determinadas propiedades de un elemento padre se transmiten a sus hijos. De hecho, se parece mucho a la herencia genética. Si los progenitores tienen los ojos azules, los hijos seguramente también tendrán los ojos azules.

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.

### 28.1.1. 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.

### 28.1.2. 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. De la misma manera que una madre de ojos azules puede tener un hijo de ojos marrones si el padre tiene los ojos marrones, las propiedades heredadas en CSS pueden anularse.

### 28.1.3. 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>
<html lang="es">
   <head>
      <meta 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.

2. 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;
}
```

3. 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">
```

4. 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.)

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

```
html {
   font: 75% Verdana,sans-serif;
   background-color: blue;
   color: white;
}
```

6. 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.

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

```
h1 {
   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.

### 28.1.4. 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.

## 28.2. Cascada

> CSS significa *cascading style sheets* (hojas de estilo en cascada) y, por lo tanto, no debería extrañarnos que la cascada sea un concepto importante. Es el mecanismo que controla el resultado final cuando se aplican varias declaraciones CSS contrapuestas al mismo elemento.

Hay tres conceptos principales que controlan el orden en el que se aplican las declaraciones de CSS:

1. Importancia.
2. Especificidad.
3. Orden en las fuentes.

A continuación, trataremos con detalle estos conceptos uno a uno.

La importancia es uno de los conceptos más... pues... importantes. Si dos declaraciones tienen la misma importancia, la especificidad de las reglas decidirá cuál se debe aplicar. Si las reglas tienen la misma especificidad, el orden de las fuentes controla el resultado.

### 28.2.1. 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 usuario.
3. Declaraciones normales en hojas de estilo de autor.
4. Declaraciones importantes en hojas de estilo de autor.
5. Declaraciones importantes en hojas de estilo de usuario.

Una **hoja de estilos de agente de usuario** es la hoja de estilo integrada del navegador. Cada navegador tiene sus propias reglas sobre cómo mostrar varios elementos de HTML si el usuario o diseñador de la página no especifica ningún estilo. Por ejemplo, los enlaces no visitados suelen ser azules y estar subrayados.

Una **hoja de estilos de usuario** es una hoja de estilo que ha especificado el usuario. No todos los navegadores son compatibles con las hojas de estilo de usuario, pero pueden ser muy prácticas, sobre todo para los usuarios con determinados tipos de minusvalía. Por ejemplo, una persona disléxica puede tener una hoja de estilo de usuario que especifique determinadas fuentes y colores que le faciliten la lectura.

Opera permite especificar hojas de estilo de usuario desde `Tools` (herramientas) (o desde el menú `Opera` en un Mac) > `Preferences...` (preferencias) > pestaña `Advanced` (avanzado) > `Content` (contenido), haciendo clic en el botón `Style Options...` (opciones de estilo...) y después señalando la hoja de estilo de usuario del campo de texto `My style sheet` (mi hoja de estilo) dentro de la pestaña `Display` (mostrar) de este cuadro de diálogo. También se puede especificar si se quiere que la hoja de estilos de usuario anule la hoja de estilos del autor (el diseñador de la web) en la pestaña `Presentation` (presentación) e, incluso, que añada un botón a la interfaz de usuario con el que poder cambiar la hoja de estilo del usuario con la del autor. Para hacerlo, salid totalmente del menú `Preferences...` (preferencias...) haciendo clic en `Aceptar` y después haced clic con el botón derecho o, mientras apretáis `Ctrl`, haced clic en algún punto de la interfaz del navegador Opera, seleccionad la vista `Customize...` (personalizar...) > pestaña `Buttons` (botones) > `Browser` (navegador) y arrastrad el botón `Author Mode` (modo autor) hasta un punto de alguna de las barras de herramientas.

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.

### 28.2.2. 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, como veremos muy pronto. 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.

- El componente "a" es bastante sencillo: es 1 para una declaración en un atributo `style`; si no, es 0.
- El componente "b" es el número de selectores de `id` en el selector (los que empiezan con `#`).
- El componente "c" es el número de selectores de atributo, incluidos los selectores de clase y pseudoclases.
- El componente "d" es el número de tipo de elementos y pseudoelementos 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.

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.

Fijémonos en cómo funciona esto en la práctica.

1. Empezad añadiendo otro párrafo al documento HTML.

```
<body>
   <h1>Título</h1>
   <p>Un párrafo de texto.</p>
   <p>Un segundo párrafo de texto.</p>
</body>
```

2. Añadid una regla a la hoja de estilo para hacer que el texto de los párrafos sea de un color diferente.

```
p {
   color: cyan;
}
```

3. Guardad los dos ficheros y recargad el documento en vuestro navegador. Se deberían ver dos párrafos de color cian.
4. Estableced una `id` en el primer párrafo para que podáis apuntar hacia él fácilmente con un selector CSS.

```
<body>
   <h1>Título</h1>
   <p id="special" >Un párrafo de texto.</p>
   <p>Un segundo párrafo de texto.</p>
</body>
```

5. Añadid una regla para el párrafo especial a la hoja de estilo.

```
#special {
   background-color: red;
   color: yellow;
}
```

6. Guardad los dos ficheros y recargad el documento en el navegador para ver el resultado, que es bastante colorido.

Veamos las declaraciones que se aplican en los dos párrafos.

La primera regla que hemos añadido establece `color:cyan` para todos los párrafos. La segunda regla establece un color de fondo rojo y establece `color:yellow` para el único elemento que tiene `id` de `special`. El primer párrafo encaja con estas dos reglas; es un párrafo y tiene la `id` de `special`.

El fondo rojo no es ningún problema porque sólo se ha especificado para `#special`. No obstante, ambas reglas incluyen una declaración de la propiedad de `color`, lo que significa que hay un conflicto. Ambas reglas tienen la misma importancia, se trata de declaraciones normales en la hoja de estilos de autor, de manera que hay que fijarse en la especificidad de los dos selectores.

El selector de la primera regla es `p`, que tiene una especificidad de 0,0,0,1 según las reglas anteriormente mencionadas, ya que incluye un único tipo de elemento. El selector de la segunda regla es `#special`, que tiene una especificidad de 0,1,0,0 porque está formado por un selector de `id`. 0,1,0,0 y es mucho más específico que 0,0,0,1, de manera que la declaración `color:yellow` gana y se obtiene el texto amarillo sobre fondo rojo.

#### 28.2.3. Orden en las fuentes

Si dos declaraciones afectan al mismo elemento, tienen la misma importancia y la misma especificidad, la señal distintiva final es el orden en las fuentes. La declaración que se ve más adelante en las hojas de estilo "ganará" a las anteriores.

Si tenéis una única hoja de estilo externa, las declaraciones al final del fichero anularán a las que sucedan antes al fichero en caso de conflicto. Las declaraciones contrapuestas también pueden suceder en diferentes hojas de estilo. En este caso, el orden en el que se enlazan, se incluyen o se importan las hojas de estilo determina qué declaración se aplica, de manera que si se tienen dos hojas de estilo enlazadas en el `head` a un documento, la enlazada al último anulará a la enlazada al primero. Veamos un ejemplo práctico de cómo funciona esto.

1. Añadid una regla nueva a la hoja de estilo, justo al final del fichero, como por ejemplo:

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

2. Guardad y recargad la página web. Ahora tenéis dos reglas que dan como resultado todos los párrafos. Tienen la misma importancia y la misma especificidad (ya que el selector es el mismo); por lo tanto, el mecanismo final para distinguirlas será el orden de las fuentes.

La última regla especifica `color:black` y anulará a `color:cyan` de la regla anterior.

Fijaos en cómo esta regla nueva no afecta en absoluto al primer párrafo. Aunque la regla nueva aparece en último lugar, su selector tiene una especificidad más baja que la de `#special`. Esto demuestra claramente cómo la especificidad tiene prioridad sobre el orden de las fuentes.

## Resumen

Herencia y cascada son conceptos básicos que cualquier diseñador web debe comprender.

La herencia permite declarar propiedades en elementos de nivel alto y que estas propiedades se transmitan a todos los elementos descendientes. Sólo algunas propiedades se heredan por defecto, pero la herencia puede forzarse mediante la palabra clave `inherit`.

La cascada soluciona los conflictos cuando varias declaraciones afectan a un elemento determinado. Las declaraciones importantes anulan a las que no lo son tanto. Entre declaraciones de igual importancia, la especificidad de la regla controla cuál se aplica. Y, si todas las demás son iguales, el orden de las fuentes supone la distinción definitiva.

## Preguntas de repaso

Preguntas a las que deberíais poder responder:

1. ¿Se puede heredar la propiedad `width`? Pensad en ello antes de contestar (¿tendría sentido?) y después mirad la respuesta correcta en la especificación CSS.
2. Si añadimos `!important` a la declaración `color:black` de la última regla de la hoja de estilo de ejemplo, ¿afectará al color del texto del primer párrafo "especial"?
3. ¿Qué selector es más específico, "`#special`" o "`html>head+body>h1+p`"?

4. ¿Qué apariencia debería tener una hoja de estilo de usuario para que nuestro documento de prueba aparezca con Comic Sans MS negra sobre fondo blanco, independientemente de la hoja de estilo del autor?