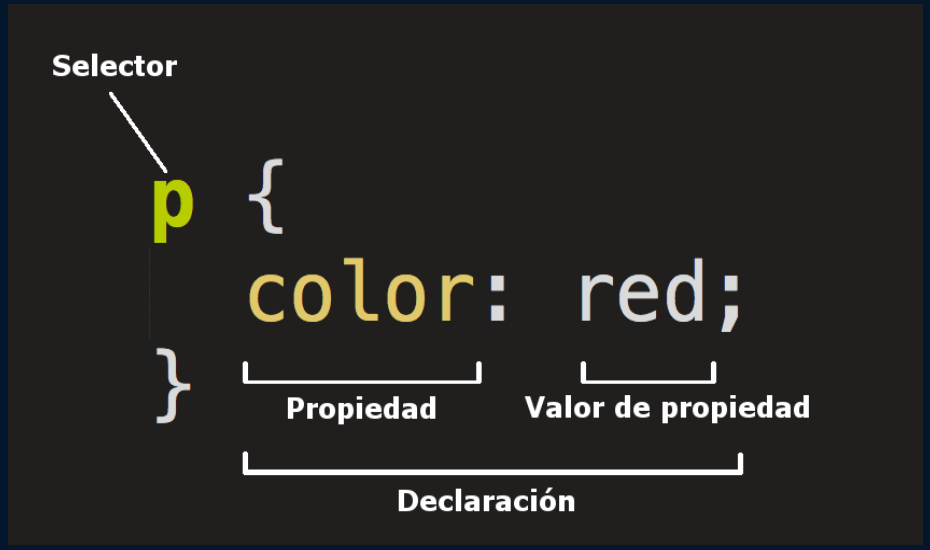
Terminología

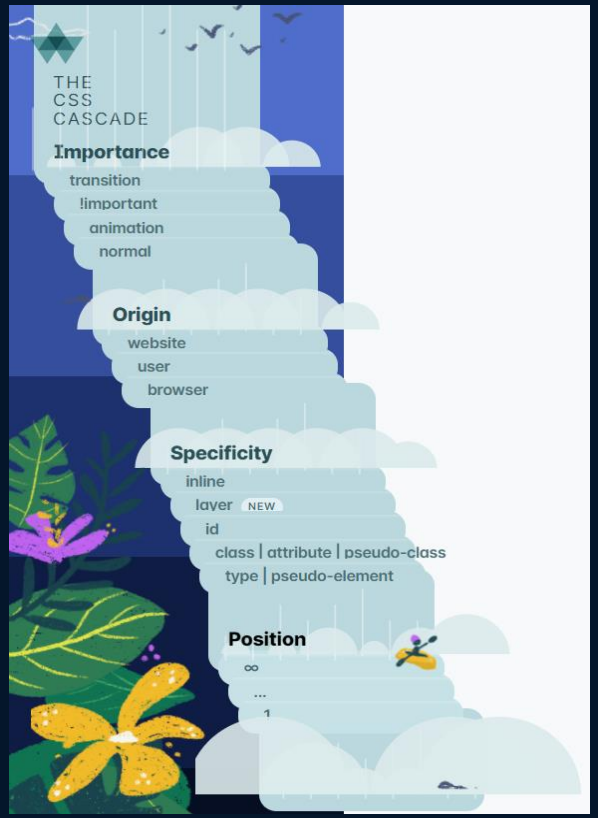


- Una línea en CSS se denomina declaración. La declaración del ejemplo está construida a partir de una propiedad (color) y un valor (red).
- Las propiedades no deben confundirse con los atributos, que son parte de la sintaxis de HTML.
- Como vimos en HTML, por ejemplo en el elemento <a href="/">, href es un atributo de la etiqueta a.

- Un grupo de declaraciones adentro de llaves se llama bloque de declaraciones, y está precedido por un selector (en este caso, un selector de tipo p –párrafo-).
- El par selector-declaración, todo junto, se denominan set de reglas. O también regla.
- Por último están las reglas "at", que son construcciones del lenguaje como las @import.

LA CASCADA EN CSS

- CSS se trata de declarar reglas: bajo distintas condiciones, queremos que pasen ciertas cosas.
- Por ejemplo: si le agrego una clase a un elemento, quiero que aplique ciertas reglas. Si el elemento X es hijo de un elemento Y, que aplique otras. El navegador toma estas reglas, interpreta cuáles aplican en dónde, y muestra la página.
- Cada vez que declaramos propiedades o reglas en CSS, esta entra en la “cascada”, que determina si esta regla termina o no como estilo final en la página (el “juez” que determina esto será el navegador, que interpreta el archivo CSS).
- Cuando dos o más reglas tienen como objetivo el mismo elemento de tu página, las reglas pueden que provean declaraciones conflictivas.



ACADEMY by NUMEN

LA CASCADA EN CSS

- En los ejemplos de abajo, tenemos tres juegos de reglas que aplican sobre el mismo elemento (la etiqueta <h1>).
- En el archive css tenemos: un selector de tipo (h1{}), un selector de id (#titulo-pagina) y un selector de clase (.titulo).
- Las tres reglas intentan ponerle una familia de fuente diferente a nuestro título:

```html
<body>
    <header class="encabezado-pagina">
        <h1 id="titulo-pagina" class="titulo">Tostadoras de Café Wombat</h1>
        <nav>
            <ul id="nav-principal" class="nav">
                <li><a href="/">Home</a></li>
                <li><a href="/cafes">Cafes</a></li>
                <li><a href="/cafeteras">Cafeteras</a></li>
                <li><a href="/especiales" class="featured">Especiales</a></li>
            </ul>
        </nav>
    </header>
</body>
```

estilos.css > .titulo

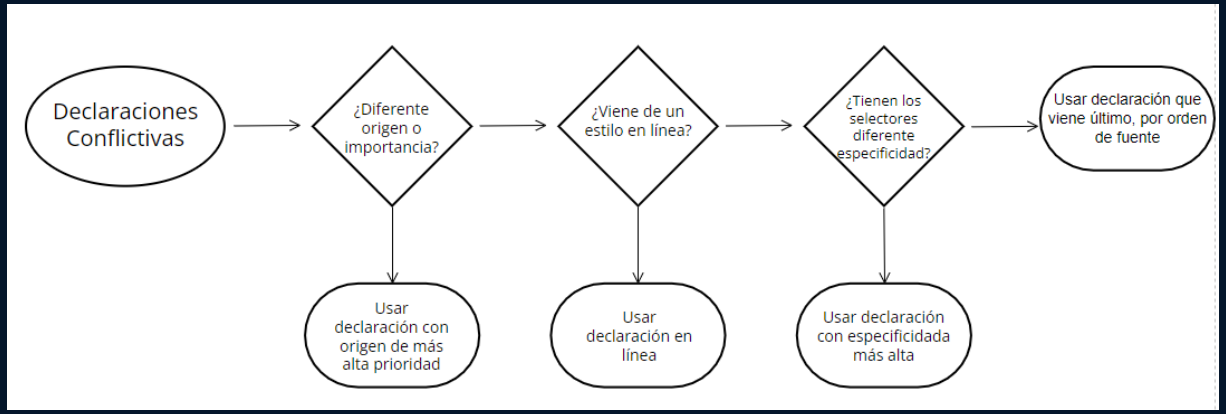
```css
h1 {
    font-family: serif;
}

#titulo-pagina {
    font-family: sans-serif;
}

.titulo {
    font-family: monospace;
}
```

GANA ID!

LA CASCADA EN CSS

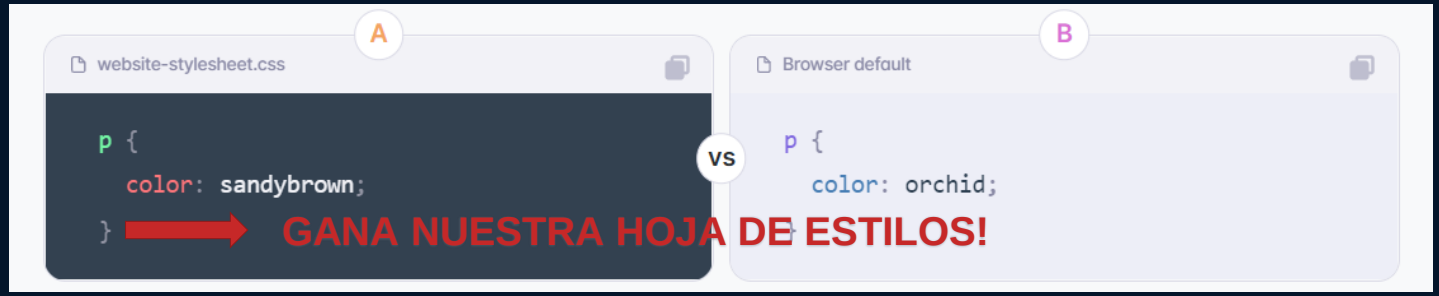
Cuando hay conflicto de declaraciones, la cascada considera tres cosas principales para resolverlo (con excepciones, que ya veremos):

1. Origen de la hoja de estilos: de dónde vienen los estilos. Tus estilos son aplicados en conjunto con los del navegador.
2. Especificidad del selector: qué selectores tienen precedencia sobre cuáles otros.
3. Orden de la Fuente o Posición: orden en el cual los estilos son declarados en la hoja de estilos.



Origen de la Hoja de Estilos

- Las hojas de estilo que le agregamos a nuestras páginas no son las únicas que el navegador aplica.
- Hay diferentes orígenes: las que hacemos nosotros, son los estilos de autor, pero también están los estilos del agente de usuario, que son los estilos por defecto de nuestro navegador.
- Los estilos por defecto son los que se aplican cuando hacemos el archive HTML sin estilos.
- Los estilos del agente de usuario tienen menor prioridad que los de autor, así que los que hagamos nosotros sobreescriben a los del navegador.



¿Quién gana?

Declaraciones !important

- Hay una **excepción**, como dijimos antes, de las reglas de estilo de origen: las declaraciones que son marcadas agregando !important al final, antes del punto y coma.
- Las declaraciones !important son tratadas como origen de más alta prioridad, así que el orden de preferencia general es: 1- Autor !important; 2- Autor; 3- Agente de usuario (navegador).
- La cascada resuelve de forma independiente los conflictos para cada propiedad de cada elemento en la página. Por ejemplo, si ponemos una fuente en negrita en un párrafo, el margen de arriba y de abajo (top y bottom) de la hoja de estilos del agente todavía aplica (excepto que sobreescribamos específicamente esas reglas).

¿Quién gana?

Especificidad

- Si las declaraciones conflictivas no pueden ser resueltas en base a su origen, lo próximo es que el navegador trata de resolverlas mirando su especificidad.
- El navegador evalúa la especificidad en dos partes: primero, los estilos aplicados en línea (inline) en el documento HTML y, segundo, los estilos aplicados utilizando selectores.

1. Estilos en línea

- Si utilizamos el atributo de HTML style para aplicar estilos, las declaraciones son aplicadas solo a ese elemento. Estas son declaraciones con ámbito (scoped), que sobreescriben cualquier declaración aplicada desde tu hoja de estilos o desde la etiqueta <style> en el head.
- Los estilos en línea no tienen selector porque son aplicados directamente al elemento.

```
<li>
    <a href="/speciales" class="featured"
    style="background-color: orange;">
    Especiales
    </a>
</li>
```

Especificidad

2. Especificidad de Selector

- La segunda parte de la especificidad es determinada por los selectores.
- Por ejemplo, un selector con dos nombres de clase tiene mayor especificidad que un selector con un solo nombre de clase. O si una declaración pone el color de fondo como naranja pero otra con más especificidad lo pone como verde, el navegador aplica el verde.
- En el ejemplo de la derecha, intento sobreescribir el selector de ID. Si aplico la clase destacado al elemento a del nav, no funciona. ¿Por qué? Porque el primer selector (de ID+etiqueta) es más específico que el segundo (sólo clase).
- Diferentes tipos de selectores tienen diferentes especificidades. Un selector de ID tiene más especificidad que un selector con cualquier número de clases. Del mismo modo, un selector de clase tiene más especificidad que un selector de etiqueta o tipo.

```css
#nav-principal a {
    color: white;
    background-color: #13a4a4;
    padding: 5px;
    border-radius: 2px;
    text-decoration: none;
}

.destacado {
    background-color: orange;
}
```

Especificidad

2. Especificidad de Selector (cont.)

- Las reglas exactas de especificidad son:
  - Si un selector tiene más IDs, gana (es más específico).
  - Si el resultado es un empate, gana el selector con más clases.
  - Si eso resulta a su vez en un empate, gana el selector con más nombres de etiqueta.
- Bueno, vimos las reglas. Ahora, ¿quién gana en la pelea de la imagen?

- El selector más específico de la imagen es el de un ID, así que se aplica el color rojo al título.
- El segundo más específico es el dos clases. Si el de ID estuviera ausente, ganaría este porque tiene más clases que el de arriba que tiene sólo una.
- El tercero más específico es el que tiene una clase.
- En el último lugar queda el de varias etiquetas.

```
html body header h1 {
    color: blue;
}

body header.encabezado-pagina h1 {
    color: orange;
}

.encabezado-pagina .title {
    color: green;
}

#titulo-pagina {
    color: red;
}
```

GANA ID!

Especificidad

Notación para especificidad

- Podemos comparar con números para saber qué selector es más específico partiéndolo en números: IDs, clases y etiquetas
- Una especificidad de 1,0,0 (cantidad de IDs, cantidad de clases, cantidad de etiquetas) toma preferencia ante una especificidad de 0,2,2 o aún ante una de 0,10,10.
- También podríamos agregar un número para un estilo en línea: 1,0,0,0 (siempre gana!).

```css
html body header h1 {
    color: blue;
}

body header.encabezado-pagina h1 {
    color: orange;
}

.encabezado-pagina .title {
    color: green;
}

#titulo-pagina {
    color: red;
}
```

Selector	Ids	Clases	Etiquetas	Notación
html body header h1	0	0	4	0,0,4
body header .encabezado-pagina h1	0	1	3	0,1,3
.encabezado-pagina .titulo	0	2	0	0,2,0
#titulo-pagina	1	0	0	1,0,0

Especificidad

Arreglemos los selectores anteriores... ¿Cómo hacer que el selector de abajo tenga predominancia sobre el de arriba?

```css
#nav-principal a {
    color: white;
    background-color: #13a4a4;
    padding: 5px;
    border-radius: 2px;
    text-decoration: none;
}

.destacado {
    background-color: orange;
}
```

```css
#nav-principal a {
    color: white;
    background-color: #13a4a4;
    padding: 5px;
    border-radius: 2px;
    text-decoration: none;
}

#nav-principal .destacado {
    background-color: orange;
}
```

Especificidad

Intentemos ver otra opción de arreglo...

```css
#nav-principal a {
    color: white;
    background-color: #13a4a4;
    padding: 5px;
    border-radius: 2px;
    text-decoration: none;
}

.destacado {
    background-color: orange;
}
```

```css
.nav {
    margin-top: 10px;
    list-style: none;
    padding-left: 0;
}

.nav li {
    display: inline-block;
}

.nav a {
    color: white;
    background-color: #13a4a4;
    padding: 5px;
    border-radius: 2px;
    text-decoration: none;
}
```

Orden de la Fuente

- El tercer paso (final) para resolver la cascada es el orden de la fuente.
- Si el origen y la especificidad son los mismos, la declaración que aparece última en la hoja de estilos toma precedencia.
- Esto significa que podemos manipular el orden de la fuente para estilizar nuestro link.
- Veamos quién gana en la siguiente pelea:

```
.nav a {
    color: white;
    background-color: #13a4a4;
    padding: 5px;
    border-radius: 2px;
    text-decoration: none;
}

a.destacado {
    background-color: orange;
}
```

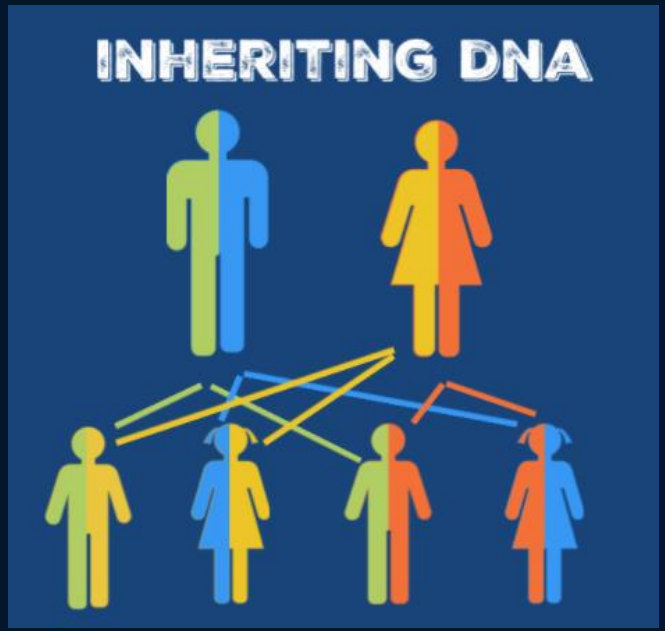
GANA LA ÚLTIMA!

- En esta solución, aunque los selectores están ordenados de diferente forma, las especificidades son iguales: clase y etiqueta (0,1,1), y etiqueta y clase (0,1,1).
- El orden de la fuente determinará entonces cuál declaración se aplica a nuestro link, resultando en un botón naranja.
- **La cascada** es la razón por la cual este orden importa: dada la misma especificidad, predomina el último estilo.

Reglas de Juego

- Hay reglas de juego para trabajar con la cascada que pueden ayudar:
  1. No uses IDs en tus selectores. Aún un solo ID en tu selector aumenta un montón la especificidad. Cuando necesites sobreescribirlo, por lo general no vas a encontrar otro ID significativo para utilizar, así que vas a terminar copiando el selector original y añadiendo otra clase para distinguirlo del que estás intentando sobreescribir.
  2. No uses !important. Esto es aún más difícil de sobreescribir que un ID. Una vez que lo usaste, cada vez que necesites sobreescribir la declaración original vas a tener que agregar !important.
- Estas reglas son consejos, no hace falta seguirlas siempre. Hay excepciones en donde utilizar IDs o !important puede estar bien, pero hay que tratar de no abusar.

HERENCIA



- Existe una última forma en la cual un elemento puede recibir estilos: la herencia.
- El concepto de cascada frecuentemente viene en combinación con el de herencia pero, aunque estos temas están relacionados, debemos entenderlos por separado.
- Si un elemento no tiene valor en cascada para una propiedad determinada, puede heredar uno de un elemento ancestro. Por ejemplo, es común aplicar una font-family al elemento <body>.
- Todos los elementos derivados de este ancestro van a heredar la misma fuente, por lo que no hace falta aplicarla a cada elemento de nuestra página.
- También, los elementos derivados o hijos pueden aceptar esa herencia, o no hacerlo y sobreescribirla.

HERENCIA

PROPIEDADES HEREDADAS Y NO HEREDADAS

HERENCIA

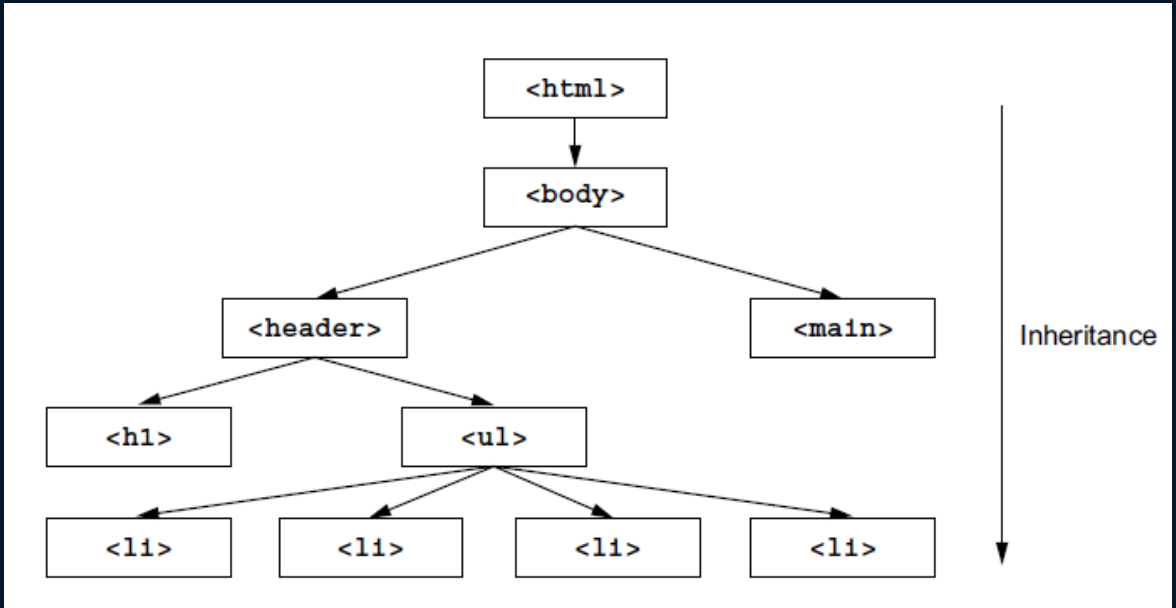
Propiedades heredadas (Inherited properties)

- Las propiedades heredadas son las que pertenecen principalmente al texto: color, font, font-family, font-size, font-weight, font-variant, font-style, line-height, letter-spacing, text-align, text-indent, text-transform, white-space, y word-spacing.
- También se heredan algunas otras, como las propiedades de la lista: list-style, list-style-type, list-style-position, y list-style-image.
- Las propiedades del borde de la tabla, border-collapse y border-spacing, también se heredan.

Propiedades no heredadas (Non-inherited properties)

- La herencia no siempre se aplica, y eso NO es necesariamente algo malo. Para algunas propiedades, la herencia tendría un efecto negativo en la apariencia de una página.
- Los márgenes, el padding y los bordes (entre otras propiedades) no son heredados por las etiquetas y no es deseable que lo hagan.
- Por ejemplo, si tenemos un borde en un elemento párrafo <p>, ¿sería beneficioso que un elemento hijo tal como (énfasis) lo heredara? Se vería raro. Por tal motivo, se ha decidido que no es práctico que la herencia comprenda ciertas cosas.

HERENCIA



Las propiedades heredadas van pasando hacia abajo en el árbol del DOM, desde los nodos “padres” a sus descendientes. Cada elemento tiene solo un padre directo.

FUENTES Y PÁGINAS DE INTERÉS

- CSS in Depth – Keith J. Grant. Manning (2018).
- The CSS Cascade: https://wattenberger.com/blog/css-cascade
- Cascada y Herencia: https://developer.mozilla.org/es/docs/Learn/CSS/Building_blocks/Cascade_and_inheritance

ACADEMY
by NUMEN