

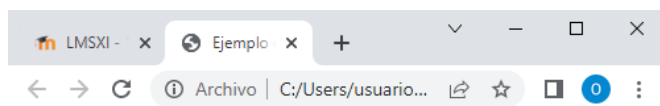
## Concepto

- Cuando se establece el valor de alguna propiedad en un elemento, todos sus descendientes heredan *inicialmente* ese mismo valor.
- Sin embargo, **la herencia de estilos no funciona en todas las propiedades CSS**, por lo que se debe estudiar cada propiedad de forma individual.

Por ejemplo:

```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3 <html xmlns="http://www.w3.org/1999/xhtml">
4 <head>
5   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
6   <title>Ejemplo de herencia de estilos</title>
7   <style type="text/css">
8     body {font-family: Arial; color:gray}
9     h1 {font-family: Courier New; }
10    p {color: red;}
11  </style>
12 </head>
13 <body>
14   <h1>Titular de la página</h1>
15   <p>Un primer <strong>párrafo</strong> de texto no <em>muy largo.</em></p>
16   <p>Un segundo <strong>párrafo</strong> de texto no <em>muy largo.</em></p>
17   <p>Un tercer párrafo de texto no muy largo con un <a href="http://www.xunta.gal">enlace
18     a la Xunta</a>. Vemos que el enlace no hereda el color de su elemento padre</p>
19 </body>
20 </html>
```

Vista en el navegador:



## Titular de la página

Un primer párrafo de texto no muy largo.

Un segundo párrafo de texto no muy largo.

Un tercer párrafo de texto no muy largo con un [enlace a la Xunta](http://www.xunta.gal).  
Vemos que el enlace no hereda el color de su elemento padre

- El elemento **<body>** tiene asignadas dos propiedades: fuente Arial y color gris
- Todos sus descendientes, en principio, deberían heredar esas propiedades, a menos que se indique otro valor para ellas. Es el caso de los elementos siguientes:
  - El elemento **<h1>** mantiene el color heredado (gray) pero cambia a la nueva fuente especificada (Courier New)
  - Los elementos **<p>**, en cambio, heredan la fuente de su elemento padre (Arial) pero adoptan el nuevo color especificado para ellos: color rojo
  - Dicho color rojo lo heredan sus descendientes **<strong>** y **<em>**, pero NO así el elemento **<a>**, que aunque sí hereda la fuente, mantiene el color y la decoración asignados por defecto por el navegador (azul y subrayado)

### Concepto

- A veces ocurre que varias reglas de estilo afectan a un mismo elemento.
- Se produce una colisión de estilos cuando para una misma propiedad de elemento (color, tamaño de letra, fuente...) se aplican diferentes valores que no son compatibles.

Por ejemplo:

```
p{color: red;}
```

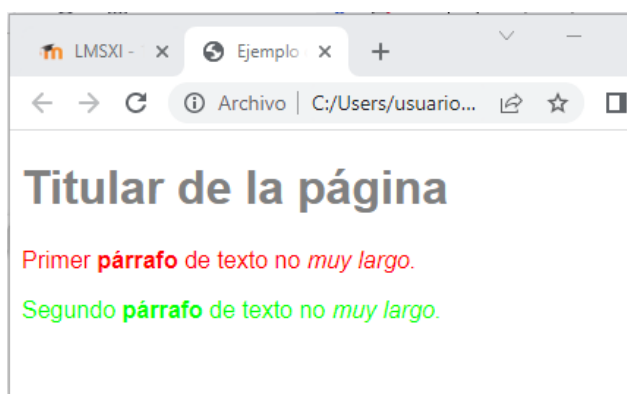
```
p{color: lime}
```

- CSS tiene un mecanismo de resolución de colisiones muy complejo pero, simplificando mucho, la norma a seguir se basa en la especificidad del selector, es decir:
  - Cuanto más específico sea un selector, más importancia tiene su regla asociada
  - A igual especificidad, prevalece la última

Por ejemplo:

```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3 <html xmlns="http://www.w3.org/1999/xhtml">
4 <head>
5   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
6   <title>Ejemplo de herencia de estilos</title>
7   <style type="text/css">
8     body {font-family: Arial; color:gray}
9     p {color: red;}
10    .distinto {color:lime}
11  </style>
12 </head>
13 <body>
14   <h1>Titular de la página</h1>
15   <p>Primer <strong>párrafo</strong> de texto no <em>muy largo.</em></p>
16   <p class="distinto">Segundo <strong>párrafo</strong> de texto no <em>muy largo.</em></p>
17 </body>
18 </html>
```

Vista en el navegador:



- La regla de la línea 8

```
body {font-family: Arial; color:gray}
```

afecta a los dos párrafos por herencia, por lo que ambos deberían verse de color gris. Pero en ambos casos hay una regla que les afecta de forma directa y que produce una colisión respecto al valor de la propiedad color:

- La regla de la línea 9 `p {color: red;}` afecta también a ambos párrafos, porque son elementos `<p>`, con lo cual deberían verse de color rojo.
- Y la regla de la línea 10 `.distinto {color:lime}` afecta exclusivamente al párrafo segundo, ya que es el único que contiene el atributo `class="distinto"`
- En el caso del segundo párrafo, la regla más específica para él es la que indica el color lima, y por eso se visualiza de ese color, independientemente del orden en el que se hayan codificado las reglas CSS

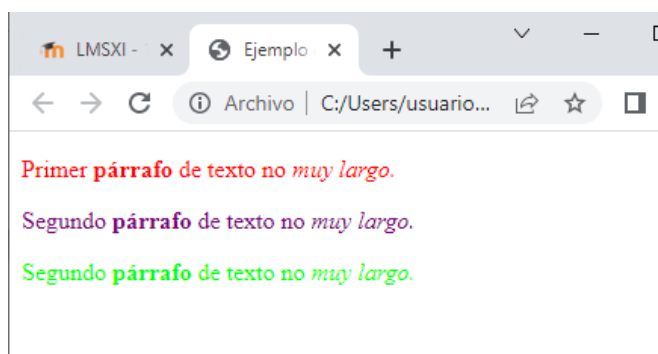
- Otro ejemplo simple

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3 <html xmlns="http://www.w3.org/1999/xhtml">
4 <head>
5   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
6   <title>Ejemplo de herencia de estilos</title>
7   <style type="text/css">
8     body {color:blue}
9     p {color: red;}
10    .distinto {color:lime}
11    #unico {color:purple;}
12  </style>
13 </head>
14 <body>
15   <p>Primer <strong>párrafo</strong> de texto no <em>muy largo.</em></p>
16   <p class="distinto" id="unico">Segundo <strong>párrafo</strong> de texto no <em>muy
17   largo.</em></p>
18   <p class="distinto">Segundo <strong>párrafo</strong> de texto no <em>muy largo.</em></p>
19 </body>
20 </html>

```

Vista en el navegador:



En el segundo párrafo colisionan los posibles colores. Sería azul por herencia, pero existen reglas más específicas que actúan sobre él:

- Sería **rojo** porque es un elemento `<p>`
- Sería **lima** porque es un elemento de clase `class="distinto"`
- Sería **púrpura** porque es un elemento con el identificador `id="unico"`

El identificador es el que le confiere mayor especificidad y, por tanto, se ve púrpura.

- Sin embargo, hay casos más complejos. Por ejemplo:

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3 <html xmlns="http://www.w3.org/1999/xhtml">
4 <head>
5   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
6   <title>Ejemplo de herencia de estilos</title>
7   <style type="text/css">
8     body {color:blue}
9     p strong {color:lime}
10    p .especial {color:yellow;}
11    p strong.especial{color:orange;}
12    strong {color: red;}
13    .especial {color:purple;}
14  </style>
15 </head>
16 <body>
17   <p>Primer <strong>párrafo</strong> de texto no <em>muy largo.</em></p>
18   <p>Segundo <strong>párrafo</strong> de texto no <em>muy largo.</em></p>
19   <p>Segundo <strong class="especial">párrafo</strong> de texto no <em>muy largo.</em></p>
20 </body>
21 </html>

```

Determinar de qué color se vería cada elemento <strong> puede revestir más complejidad sólo razonando el mecanismo de gestión de colisiones simple que hemos comentado antes.

### Algoritmo de gestión de colisiones:

[https://www.aprenderaprogramar.es/index.php?option=com\\_content&view=article&id=722:cascada-de-estilos-calculo-de-especificidad-como-usar-important-en-css-ejercicios-resueltos-cu01018d&catid=75:tutorial-basico-programador-web-css-desde-cero&Itemid=203](https://www.aprenderaprogramar.es/index.php?option=com_content&view=article&id=722:cascada-de-estilos-calculo-de-especificidad-como-usar-important-en-css-ejercicios-resueltos-cu01018d&catid=75:tutorial-basico-programador-web-css-desde-cero&Itemid=203)

### 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 se aplican igual de directamente a un elemento es un valor numérico que utiliza el navegador para ordenar reglas que entran en conflicto.



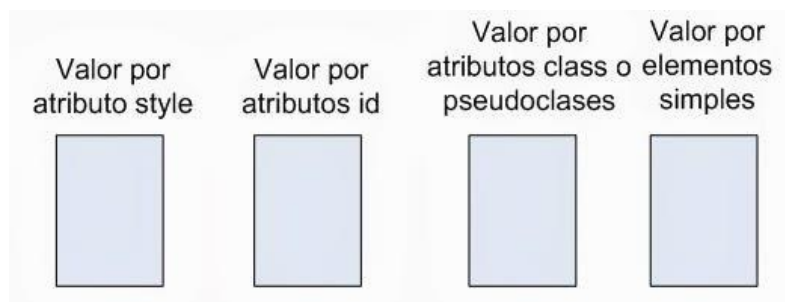
La especificidad se puede calcular como un número que consta de cuatro dígitos ABCD en el cual tenemos:

**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 (atributo 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.

### Ejemplo:

Supongamos que tenemos el siguiente código (X)HTML y las siguientes declaraciones CSS

```
<body>
  <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>
    <!-- aquí podría ir más contenido-->
  </div>
</body>
```

Declaraciones	Puntos especificidad	Color del texto y motivo
body {color: grey;}	Afecta por herencia (0001)	Cyan por especificidad
body div.destacado p {color: cyan;}	0013	
.destacado {color: green;}	Afecta por herencia (0010)	
div.destacado {color: blue;}	Afecta por herencia (0011)	
div.destacado p {color: yellow;}	0012	
body {color:gray;}	Afecta por herencia (única regla)	Gris

body {color: grey;} .destacado {color: green;}	Afecta por herencia (0001) Afecta por herencia (0010)	Green por especificidad
body div.destacado p {color: cyan;} .destacado {color: green;}	0013 Afecta por herencia (0010)	Cyan por especificidad
body div.destacado p {color: cyan;} .destacado {color: green;} p {color: blue;}	0013 Afecta por herencia (0010) 0001	Cyan por especificidad

### USO DEL MODIFICADOR !important

- El modificador !important se escribe al final de una declaración y se convierte en la opción de mayor prioridad aunque no sea la más específica.
- Funciona en las reglas que afectan al elemento de forma directa; es decir, no por herencia  
Nota: puede que no todos los navegadores respondan igual a la palabra clave !important.
- En líneas generales, debe limitarse su uso al máximo e incluso evitarse siempre que sea posible

### Ejemplo anterior con reglas que contienen el modificador !important

```

<body>
  <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>
    <!-- aquí podría ir más contenido-->
  </div>
</body>

```

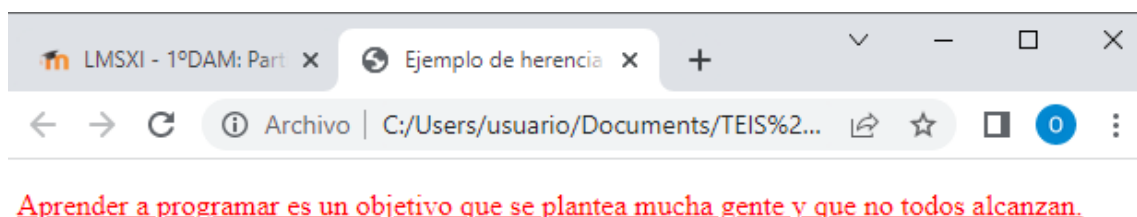
Declaraciones	Puntos especificidad	Color del texto y motivo
body div.destacado p {color: cyan;} .destacado {color: green;} p {color: blue !important;}	0013 Afecta por herencia (0010) 0001 pero lleva !important	Blue: hay dos reglas que afectan directamente al elemento p, la primera y la tercera. Al tener la tercera la declaración !important significa que se sobrescriben el resto de reglas, independientemente de su valor de especificidad y origen.
body div.destacado p {color: cyan;} .destacado {color: green !important;}	0013 Afecta por herencia (0010)	Cyan: aunque la segunda regla lleva la declaración !important no afecta al elemento de forma directa (como la primera regla) sino por herencia

## COMENTARIO IMPORTANTE

Vamos a considerar las siguientes declaraciones.

```
body div.destacado p {color: cyan; text-decoration:underline;}
p {color: red !important;}
</style>
</head>
<body>
  <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>
    <!-- aquí podría ir más contenido-->
  </div>
</body>
</html>
```

Vista en el navegador:



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.



## Prioridad CSS

Cuando dos declaraciones afectan a un mismo elemento. ¿cual de ellas se interpreta en el navegador como más importante?

```
1 <!doctype html>
2 <html>
3 <head>
4 <meta charset="utf-8">
5 <title>Herencia</title>
6 <style type="text/css">
7 #caja header h1 { color: blue; }
8 #caja .cabecera h1 { color: red; }
9 header h1 { color: lime; }
10 h1 { color: purple; }
11 h1 { color: grey; }
12 </style>
13 </head>
14 <body>
15 <div id="caja">
16 <header class="cabecera">
17 <h1>Cabecera: header</h1>
18 </header>
19 </div>
20 </body>
21 </html>
22
```

A = 0 estilos en línea  
B = 0 ID  
C = 0 clases  
D = 1 elemento  
Puntuación = 0,0,0,1

Hay que calcular la tupla (A, B, C, D) ganadora de todas las reglas CSS que compiten. A tiene máximo peso y D mínimo. Si hay empate en A, se mira B y así sucesivamente.

A = estilo en línea  
B = número de IDs  
C = número de clases  
D = número de marcas HTML

## Prioridad CSS

Cuando dos declaraciones afectan a un mismo elemento. ¿cual de ellas se interpreta en el navegador como más importante?

```
1 <!doctype html>
2 <html>
3 <head>
4 <meta charset="utf-8">
5 <title>Herencia</title>
6 <style type="text/css">
7 #caja header h1 { color: blue; }
8 #caja .cabecera h1 { color: red; }
9 header h1 { color: lime; }
10 h1 { color: purple; }
11 h1 { color: grey; }
12 </style>
13 </head>
14 <body>
15 <div id="caja">
16 <header class="cabecera">
17 <h1>Cabecera: header</h1>
18 </header>
19 </div>
20 </body>
21 </html>
22
```

A = 0 estilos en línea  
B = 0 ID  
C = 0 clases  
D = 2 marcas  
Puntuación = 0,0,0,2

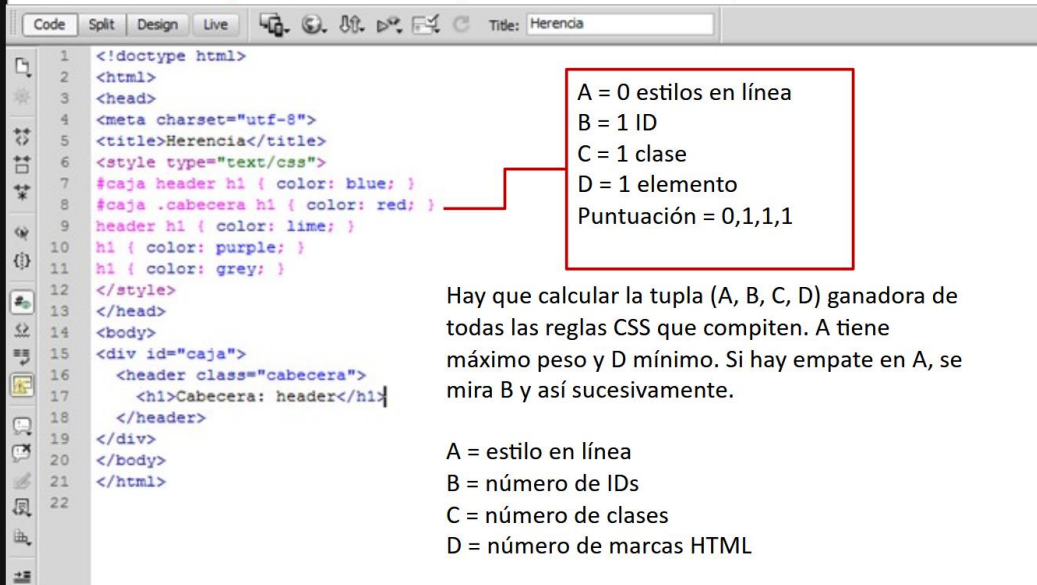
Hay que calcular la tupla (A, B, C, D) ganadora de todas las reglas CSS que compiten. A tiene máximo peso y D mínimo. Si hay empate en A, se mira B y así sucesivamente.

A = estilo en línea  
B = número de IDs  
C = número de clases  
D = número de marcas HTML



# Prioridad CSS

Cuando dos declaraciones afectan a un mismo elemento. ¿cual de ellas se interpreta en el navegador como más importante?



The screenshot shows a code editor with the following CSS rules:

```
1 <!doctype html>
2 <html>
3 <head>
4 <meta charset="utf-8">
5 <title>Herencia</title>
6 <style type="text/css">
7 #caja header h1 { color: blue; }
8 #caja .cabecera h1 { color: red; }
9 header h1 { color: lime; }
10 h1 { color: purple; }
11 h1 { color: grey; }
12 </style>
13 </head>
14 <body>
15 <div id="caja">
16 <header class="cabecera">
17 <h1>Cabecera: header</h1>
18 </header>
19 </div>
20 </body>
21 </html>
22
```

A red box highlights the first two rules, with a line pointing to the calculation:

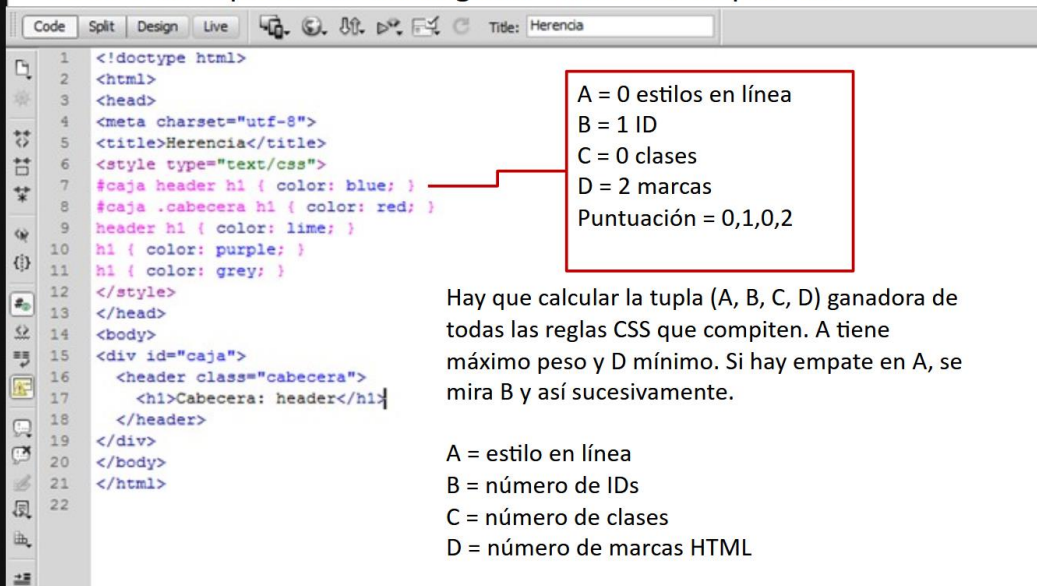
A = 0 estilos en línea  
B = 1 ID  
C = 1 clase  
D = 1 elemento  
Puntuación = 0,1,1,1

Hay que calcular la tupla (A, B, C, D) ganadora de todas las reglas CSS que compiten. A tiene máximo peso y D mínimo. Si hay empate en A, se mira B y así sucesivamente.

A = estilo en línea  
B = número de IDs  
C = número de clases  
D = número de marcas HTML

# Prioridad CSS

Cuando dos declaraciones afectan a un mismo elemento. ¿cual de ellas se interpreta en el navegador como más importante?



The screenshot shows a code editor with the following CSS rules:

```
1 <!doctype html>
2 <html>
3 <head>
4 <meta charset="utf-8">
5 <title>Herencia</title>
6 <style type="text/css">
7 #caja header h1 { color: blue; }
8 #caja .cabecera h1 { color: red; }
9 header h1 { color: lime; }
10 h1 { color: purple; }
11 h1 { color: grey; }
12 </style>
13 </head>
14 <body>
15 <div id="caja">
16 <header class="cabecera">
17 <h1>Cabecera: header</h1>
18 </header>
19 </div>
20 </body>
21 </html>
22
```

A red box highlights the first two rules, with a line pointing to the calculation:

A = 0 estilos en línea  
B = 1 ID  
C = 0 clases  
D = 2 marcas  
Puntuación = 0,1,0,2

Hay que calcular la tupla (A, B, C, D) ganadora de todas las reglas CSS que compiten. A tiene máximo peso y D mínimo. Si hay empate en A, se mira B y así sucesivamente.

A = estilo en línea  
B = número de IDs  
C = número de clases  
D = número de marcas HTML

# Prioridad CSS

La ganadora es, #caja .cabecera h1 = 0,1,1,1

The screenshot shows a code editor on the left and a Firefox browser window on the right. The code editor contains the following HTML and CSS:

```
1 <!doctype html>
2 <html>
3 <head>
4 <meta charset="utf-8">
5 <title>Herencia</title>
6 <style type="text/css">
7 #caja header h1 { color: blue; }
8 #caja .cabecera h1 { color: red; }
9 header h1 { color: lime; }
10 h1 { color: purple; }
11 h1 { color: grey; }
12 </style>
13 </head>
14 <body>
15 <div id="caja">
16 <header class="cabecera">
17 <h1>Cabecera: header</h1>
18 </header>
19 </div>
20 </body>
21 </html>
22
```

The browser window shows the rendered page with the text "Cabecera: header" in red. A red box highlights the text "Ganadora:" and the following specificity calculation:

Ganadora:  
A = 0 estilos en línea  
B = 1 ID  
C = 1 clase  
D = 1 elemento  
Puntuación = 0,1,1,1

# Prioridad CSS

Cuando dos declaraciones tienen el mismo valor:  
Será la última especificada

The screenshot shows a code editor on the left and a Firefox browser window on the right. The code editor contains the following HTML and CSS:

```
1 <!doctype html>
2 <html>
3 <head>
4 <meta charset="utf-8">
5 <title>Herencia</title>
6 <style type="text/css">
7 h1 { color: purple; }
8 h1 { color: grey; }
9 </style>
10 </head>
11 <body>
12 <div id="caja">
13 <header class="cabecera">
14 <h1>Cabecera: header</h1>
15 </header>
16 </div>
17 </body>
18 </html>
19
```

The browser window shows the rendered page with the text "Cabecera: header" in grey.