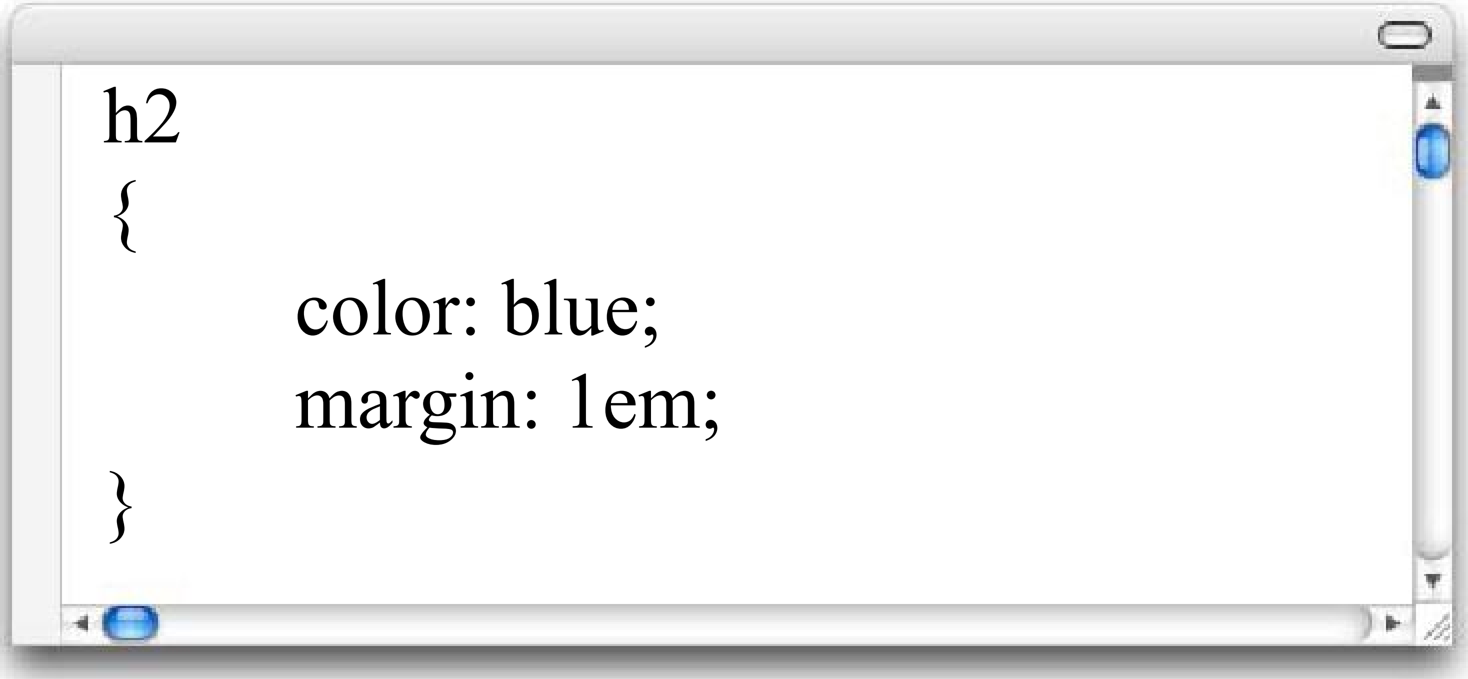
CSS
CASCADE

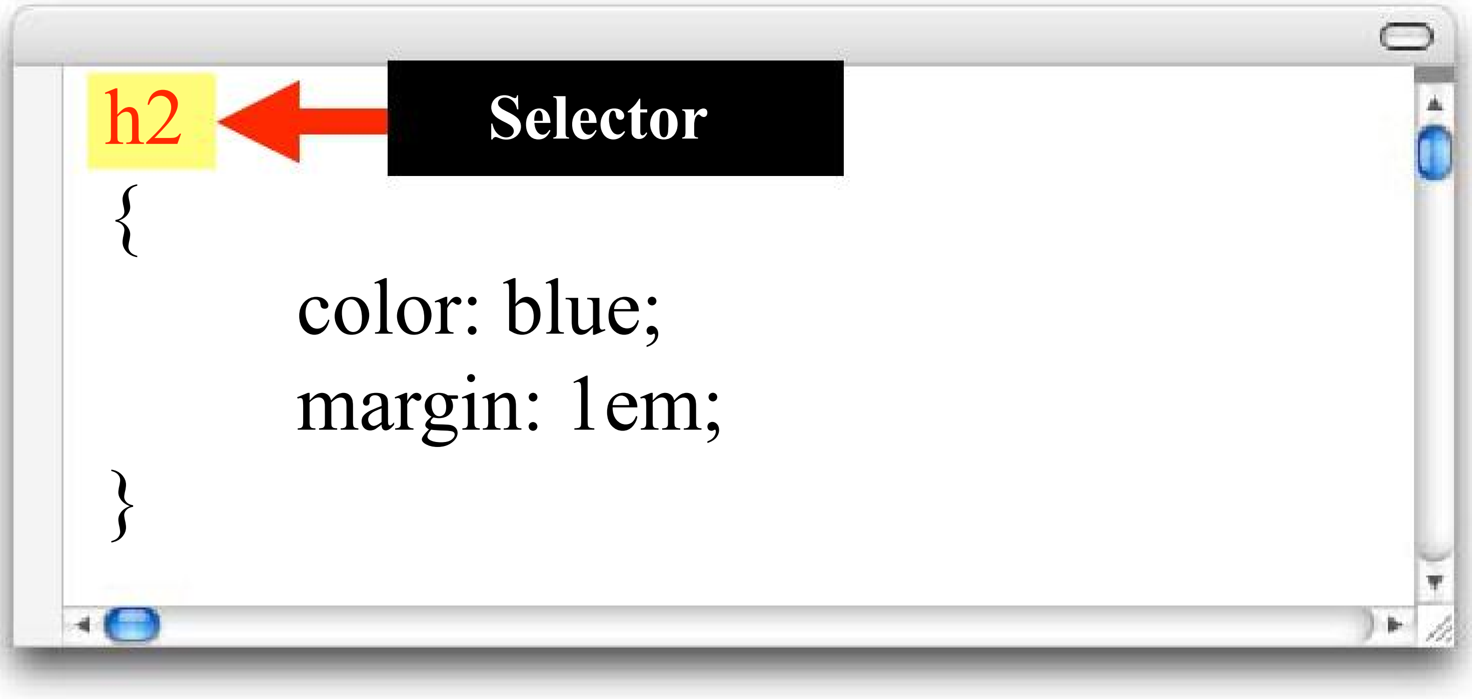
A quick background on CSS rules

CSS rules tell browsers how to render elements in an HTML document.



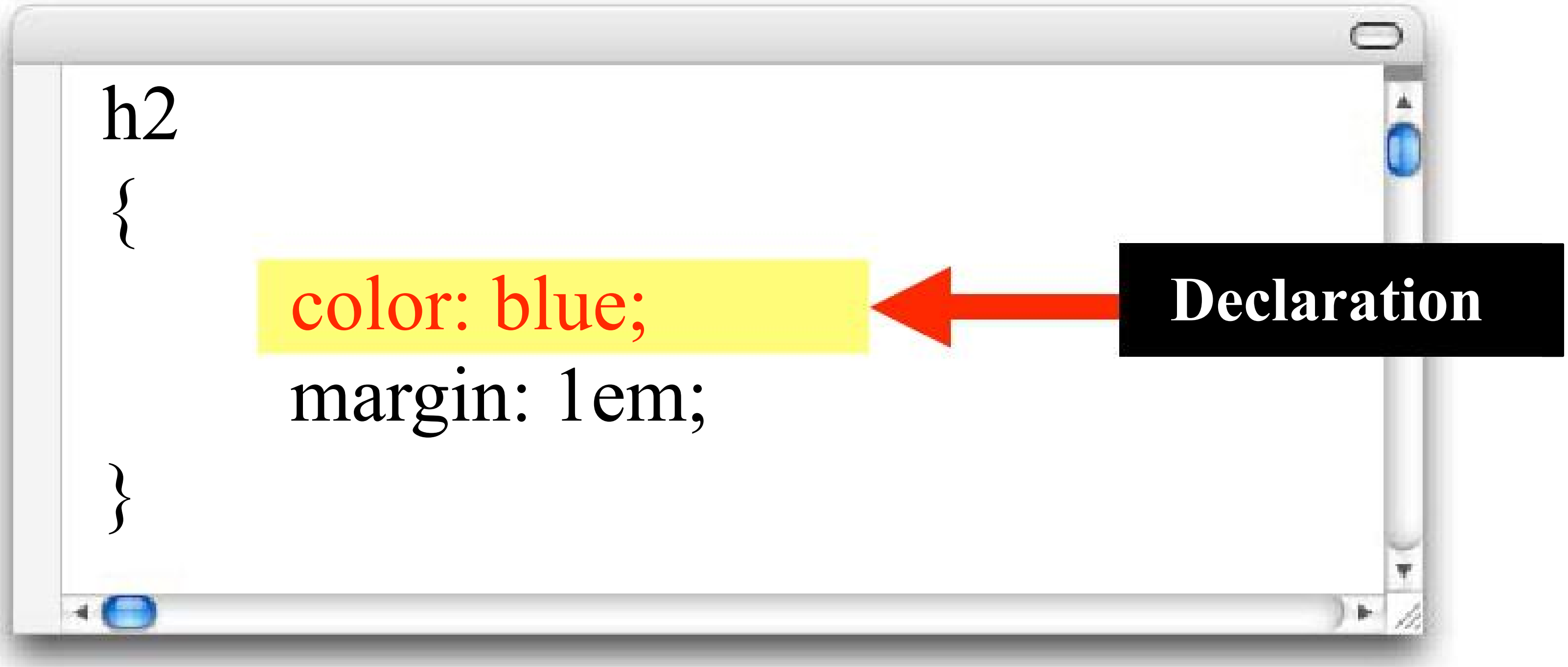
```
h2
{
        color: blue;
        margin: 1em;
}
```

The **selector** "selects" the elements in an HTML document that are to be styled.



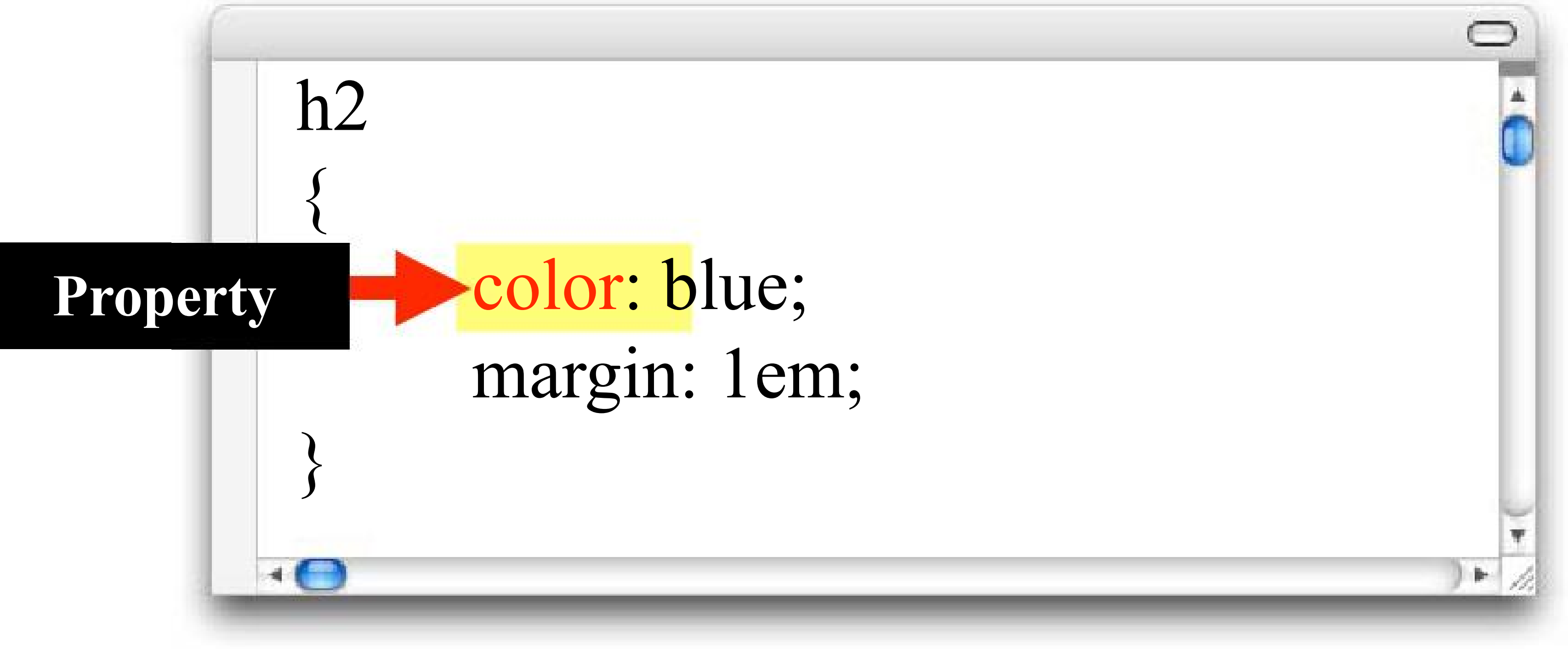
```
h2
{
        color: blue;
        margin: 1em;
}
```

The **declaration** tells a browser how to style the element.



```
h2
{
      color: blue;
      margin: 1em;
}
```

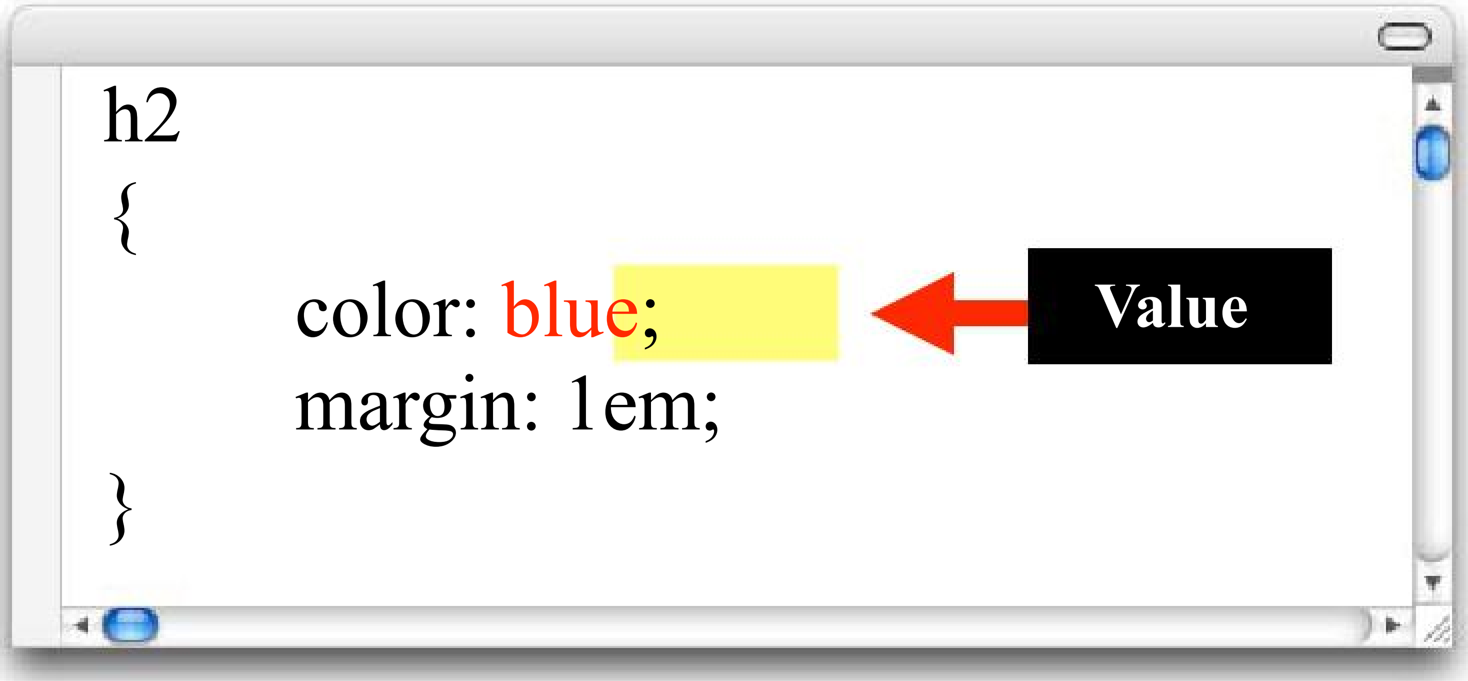
The **property** is the aspect of that element that you are choosing to style.



```
h2
{
        color: blue;
        margin: 1em;
}
```

Property

The **value** is the exact style you wish to set for the property.



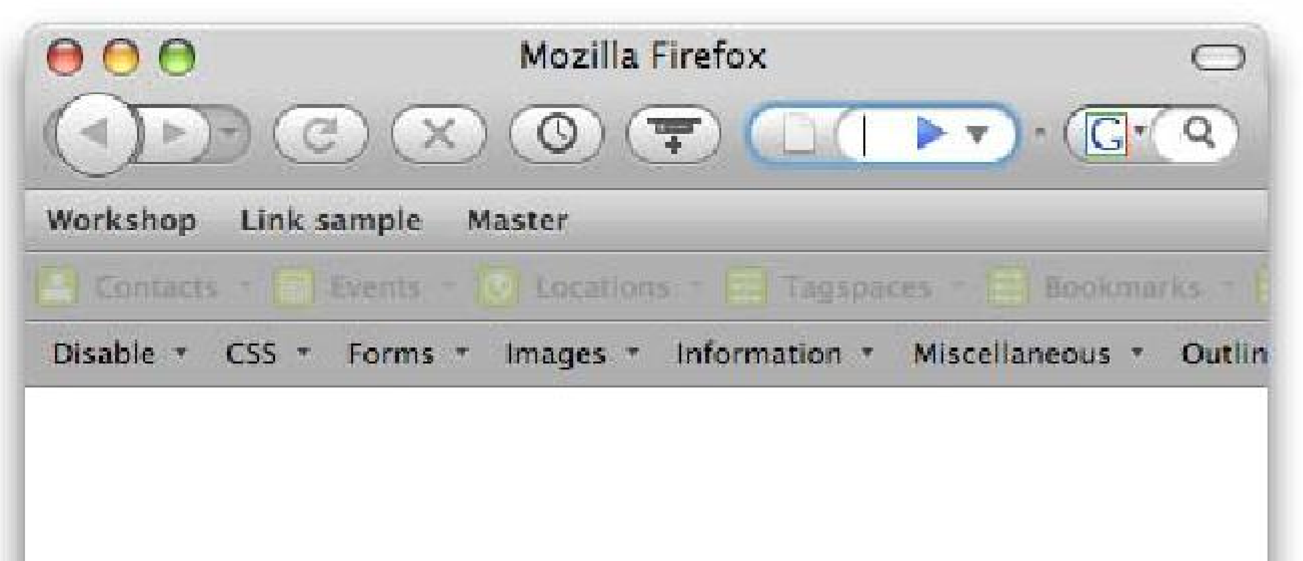
```
h2
{
        color: blue;
        margin: 1em;
}
```

Types of style sheets

HTML documents may have **three types of style sheets** applied to them.

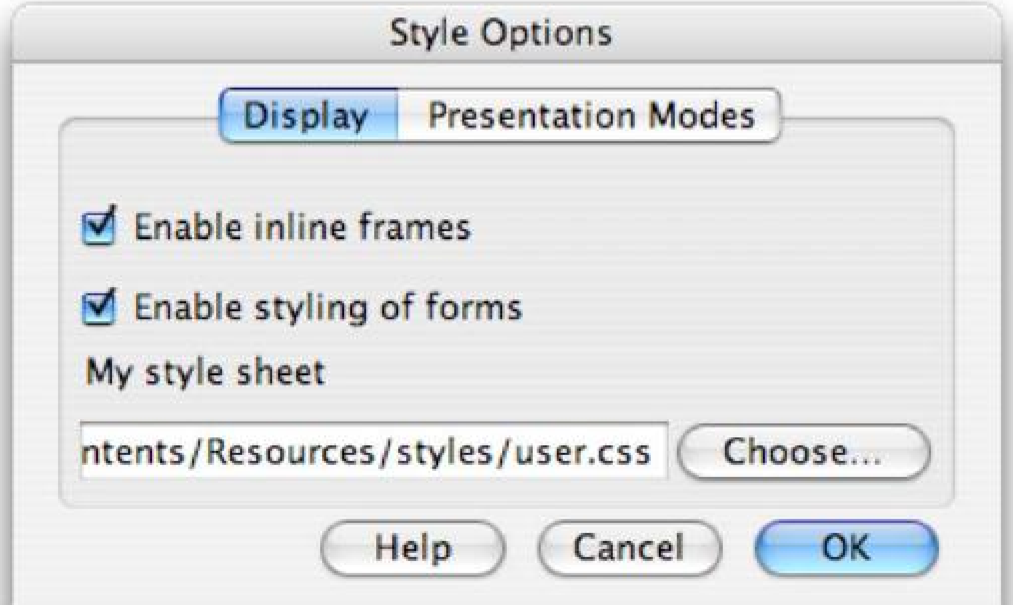
Browser style sheets

Browsers apply style sheets to all web documents. These are referred to as a "default" browser style sheet.



User style sheets

Most modern browsers allow users to apply their own style sheets within the browser.



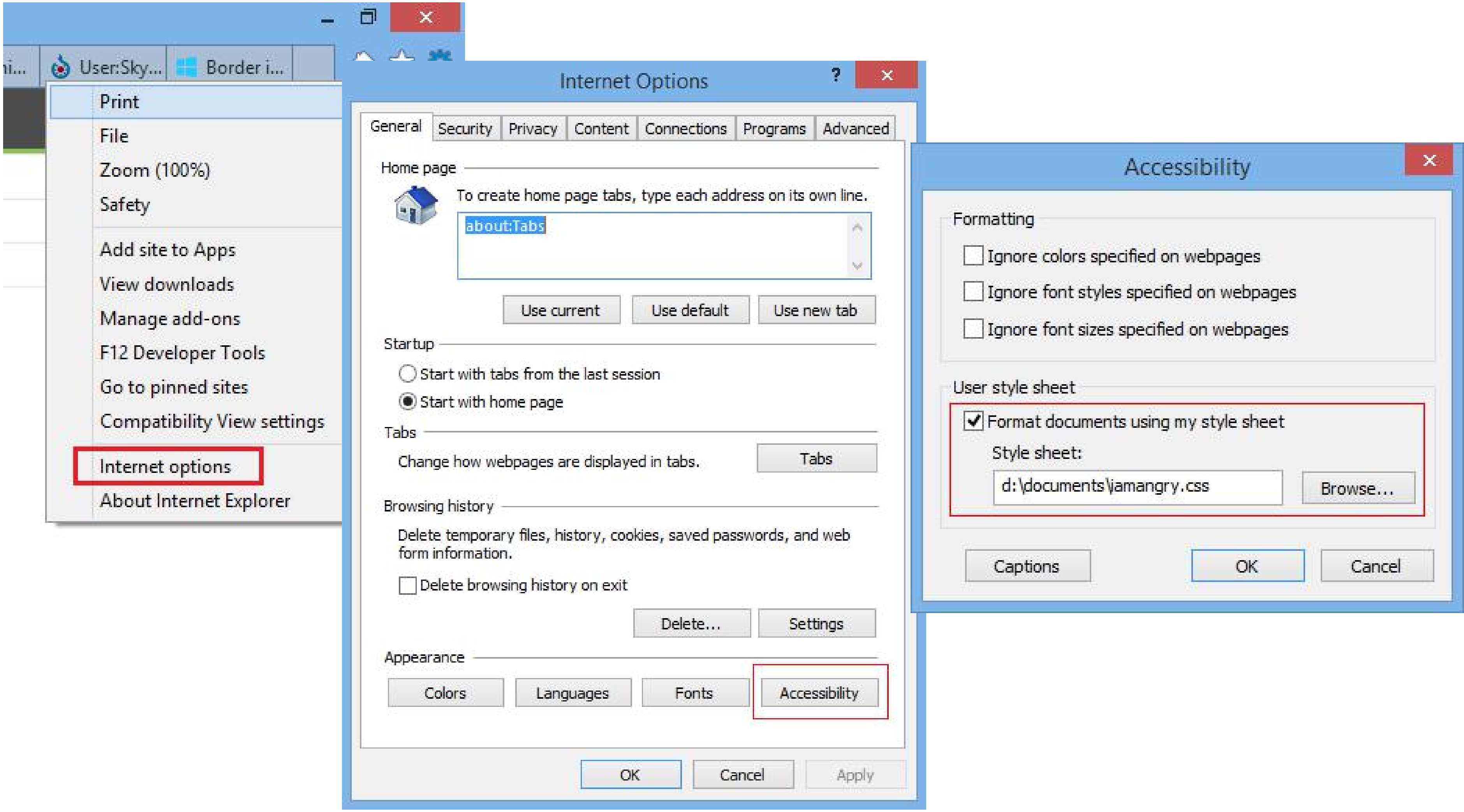
☑ Enable inline frames

☑ Enable styling of forms

My style sheet

ntents/Resources/styles/user.css

User style sheets (IE浏览器)



Home page

To create home page tabs, type each address on its own line.

about:Tabs

Startup

○ Start with tabs from the last session

● Start with home page

Tabs

Change how webpages are displayed in tabs.

Browsing history

Delete temporary files, history, cookies, saved passwords, and web form information.

□ Delete browsing history on exit

Formatting

□ Ignore colors specified on webpages

□ Ignore font styles specified on webpages

□ Ignore font sizes specified on webpages

User style sheet

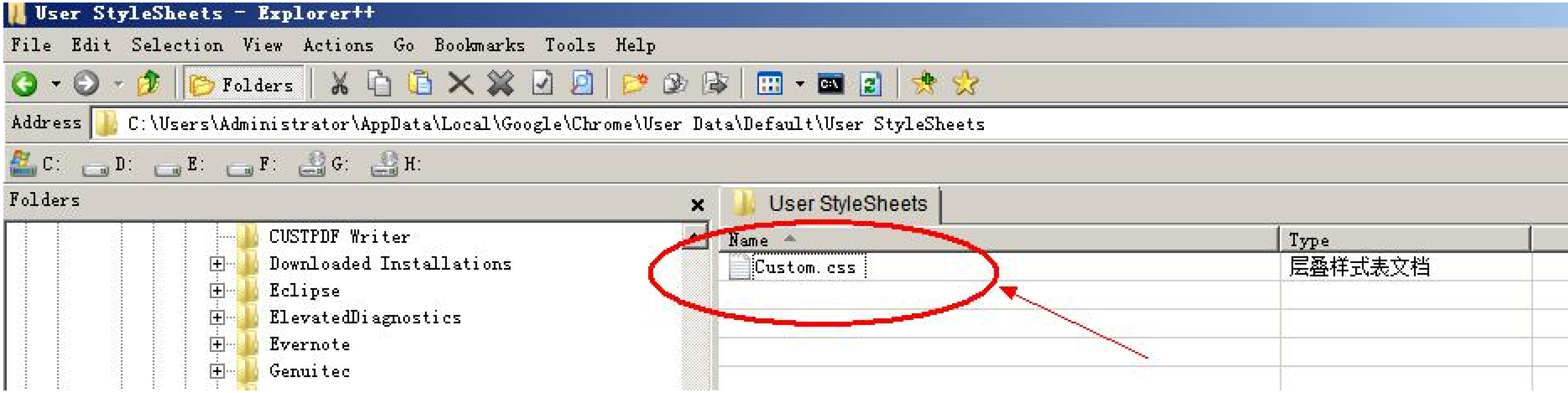
☑ Format documents using my style sheet

Style sheet: d:\documents\iamangry.css

User style sheets (chrome浏览器)

Chrome 的 User CSS 位置:

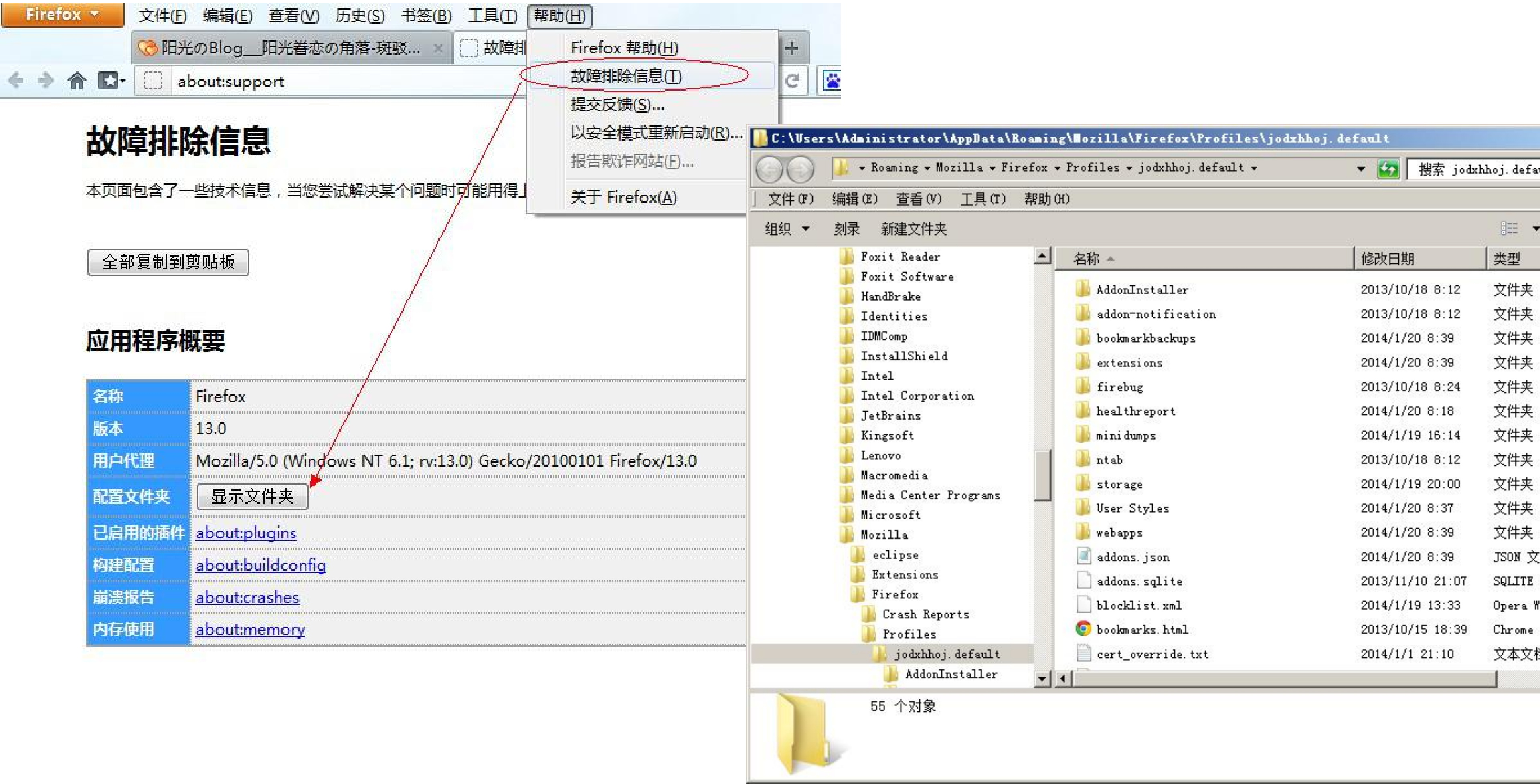
C:\Users\用户名\AppData\Local\Google\Chrome\User Data\Default\User StyleSheets



Address: C:\Users\Administrator\AppData\Local\Google\Chrome\User Data\Default\User StyleSheets

Name	Type
Custom.css	层叠样式表文档

User style sheets (firefox浏览器)



故障排除信息

本页面包含了一些技术信息，当您尝试解决某个问题时可能用得上

全部复制到剪贴板

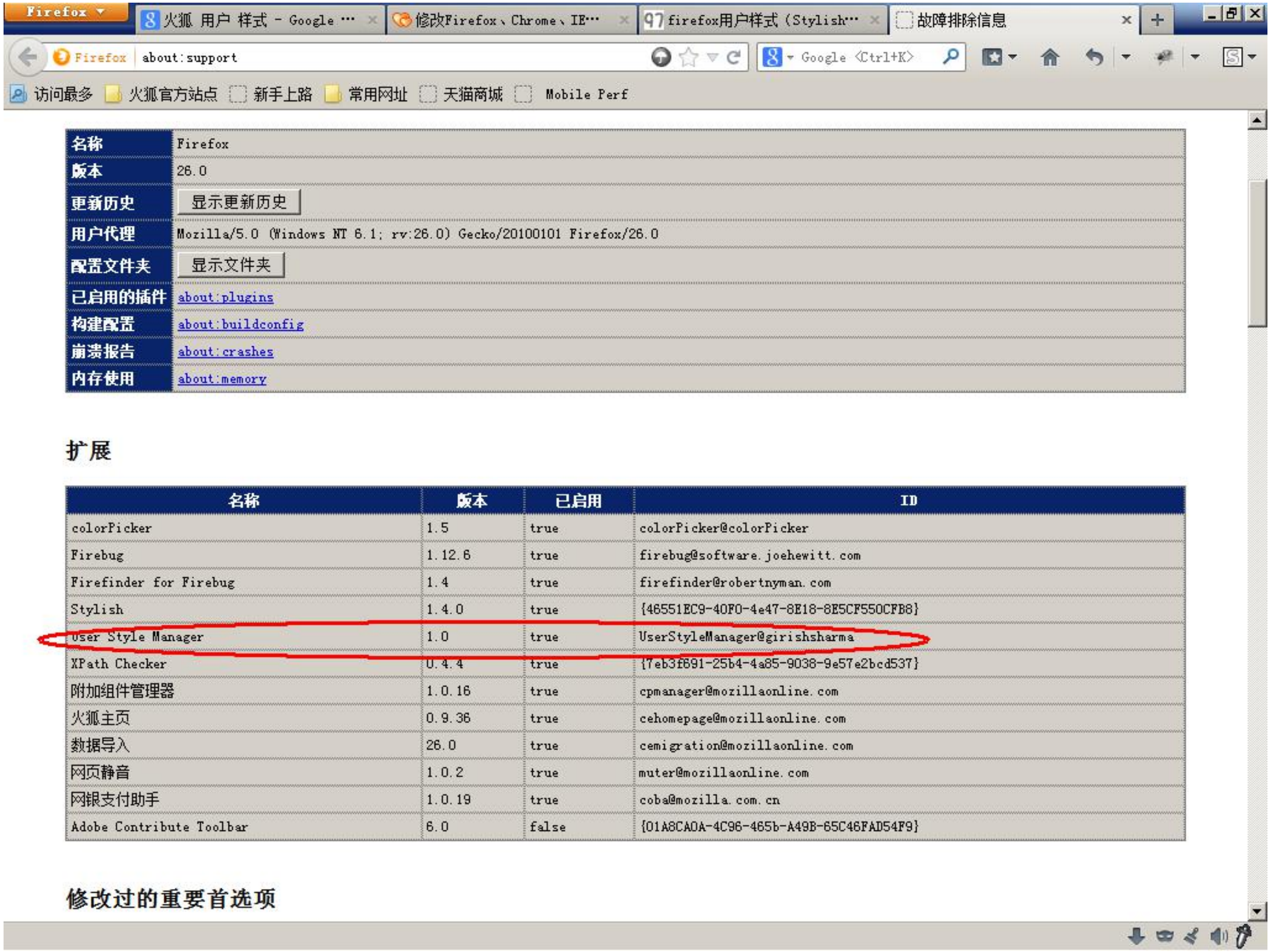
应用程序概要

名称	Firefox
版本	13.0
用户代理	Mozilla/5.0 (Windows NT 6.1; rv:13.0) Gecko/20100101 Firefox/13.0
配置文件夹	显示文件夹
已启用的插件	about:plugins
构建配置	about:buildconfig
崩溃报告	about:crashes
内存使用	about:memory

C:\Users\Administrator\AppData\Roaming\Mozilla\Firefox\Profiles\jodxhhoj.default

名称	修改日期	类型
AddonInstaller	2013/10/18 8:12	文件夹
addon-notification	2013/10/18 8:12	文件夹
bookmarkbackups	2014/1/20 8:39	文件夹
extensions	2014/1/20 8:39	文件夹
firebug	2013/10/18 8:24	文件夹
healthreport	2014/1/20 8:18	文件夹
minidumps	2014/1/19 16:14	文件夹
ntab	2013/10/18 8:12	文件夹
storage	2014/1/19 20:00	文件夹
User Styles	2014/1/20 8:37	文件夹
webapps	2014/1/20 8:39	文件夹
addons.json	2014/1/20 8:39	JSON 文
addons.sqlite	2013/11/10 21:07	SQLITE
blocklist.xml	2014/1/19 13:33	Opera W
bookmarks.html	2013/10/15 18:39	Chrome
cert_override.txt	2014/1/1 21:10	文本文档

User style sheets (firefox浏览器)



名称	Firefox
版本	26.0
更新历史	显示更新历史
用户代理	Mozilla/5.0 (Windows NT 6.1; rv:26.0) Gecko/20100101 Firefox/26.0
配置文件夹	显示文件夹
已启用的插件	about:plugins
构建配置	about:buildconfig
崩溃报告	about:crashes
内存使用	about:memory

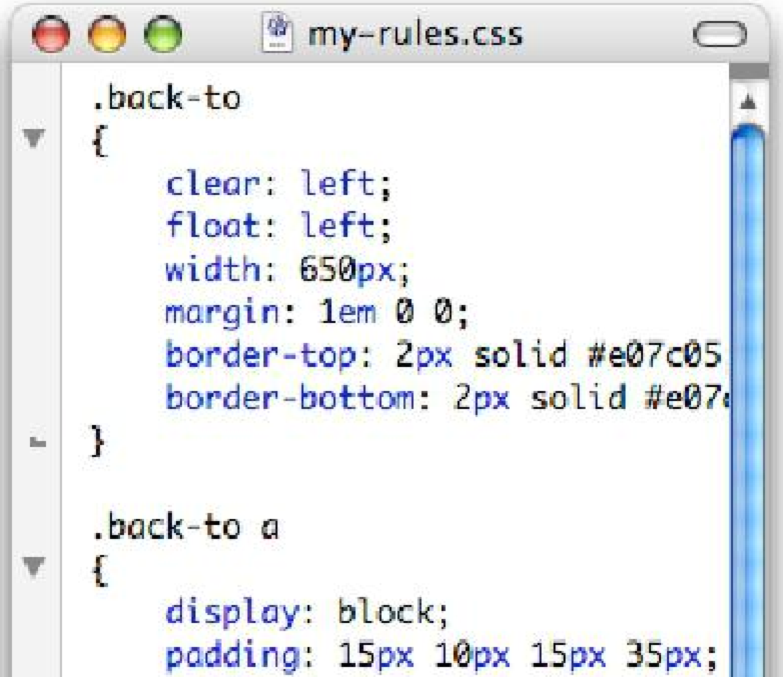
扩展

名称	版本	已启用	ID
colorPicker	1.5	true	colorPicker@colorPicker
Firebug	1.12.6	true	firebug@software.joehewitt.com
Firefinder for Firebug	1.4	true	firefinder@robertnyman.com
Stylish	1.4.0	true	{46551EC9-40F0-4e47-8E18-8E5CF550CFB8}
User Style Manager	1.0	true	UserStyleManager@girishsharma
XPath Checker	0.4.4	true	{7eb3f691-25b4-4a85-9038-9e57e2bcd537}
附加组件管理器	1.0.16	true	cpmanager@mozillaonline.com
火狐主页	0.9.36	true	cehomepage@mozillaonline.com
数据导入	26.0	true	cemigration@mozillaonline.com
网页静音	1.0.2	true	muter@mozillaonline.com
网银支付助手	1.0.19	true	coba@mozilla.com.cn
Adobe Contribute Toolbar	6.0	false	{01A8CA0A-4C96-465b-A49B-65C46FAD54F9}

修改过的重要首选项

Author style sheets

Web authors can apply one or more style sheets to an HTML document.



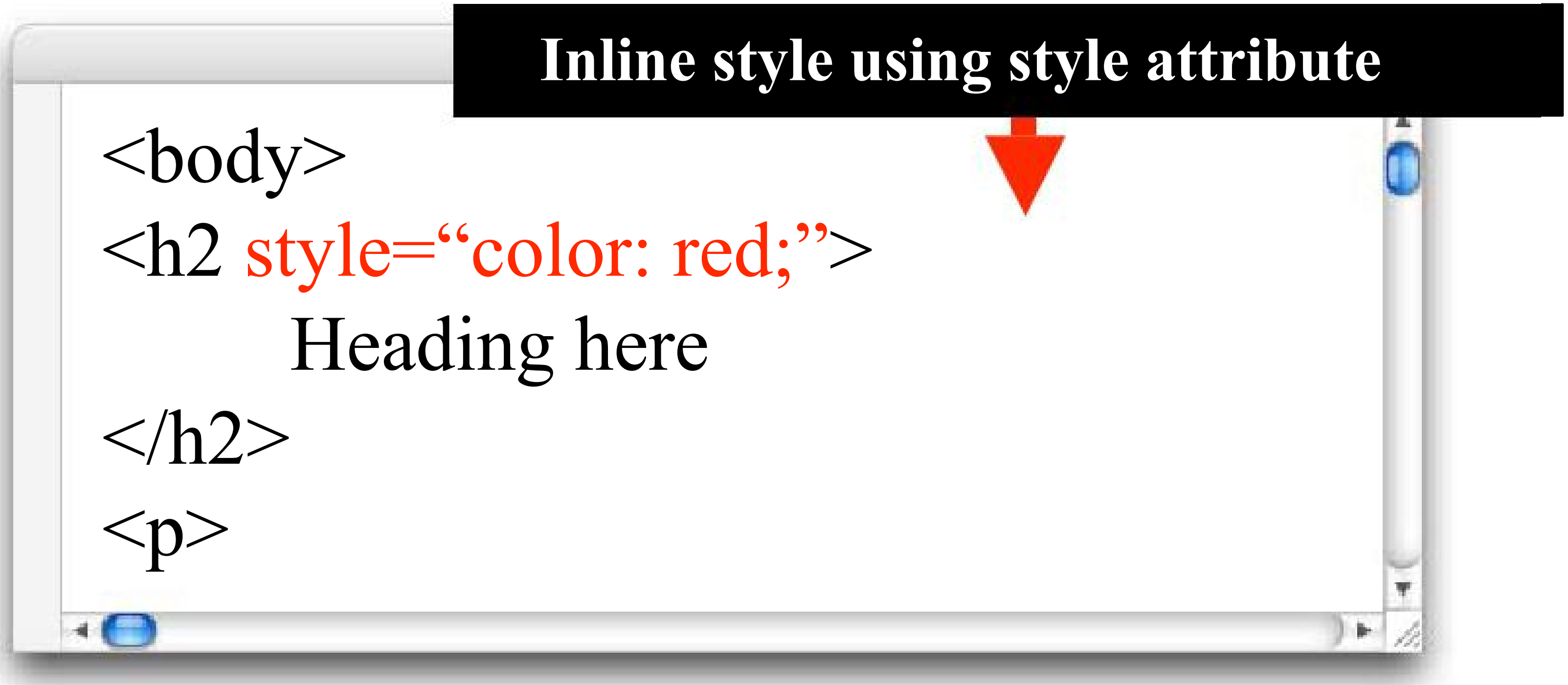
```
.back-to
{
    clear: left;
    float: left;
    width: 650px;
    margin: 1em 0 0;
    border-top: 2px solid #e07c05
    border-bottom: 2px solid #e07
}

.back-to a
{
    display: block;
    padding: 15px 10px 15px 35px;
```

Author styles

There are **three methods** that authors can use to add CSS styles to an HTML document

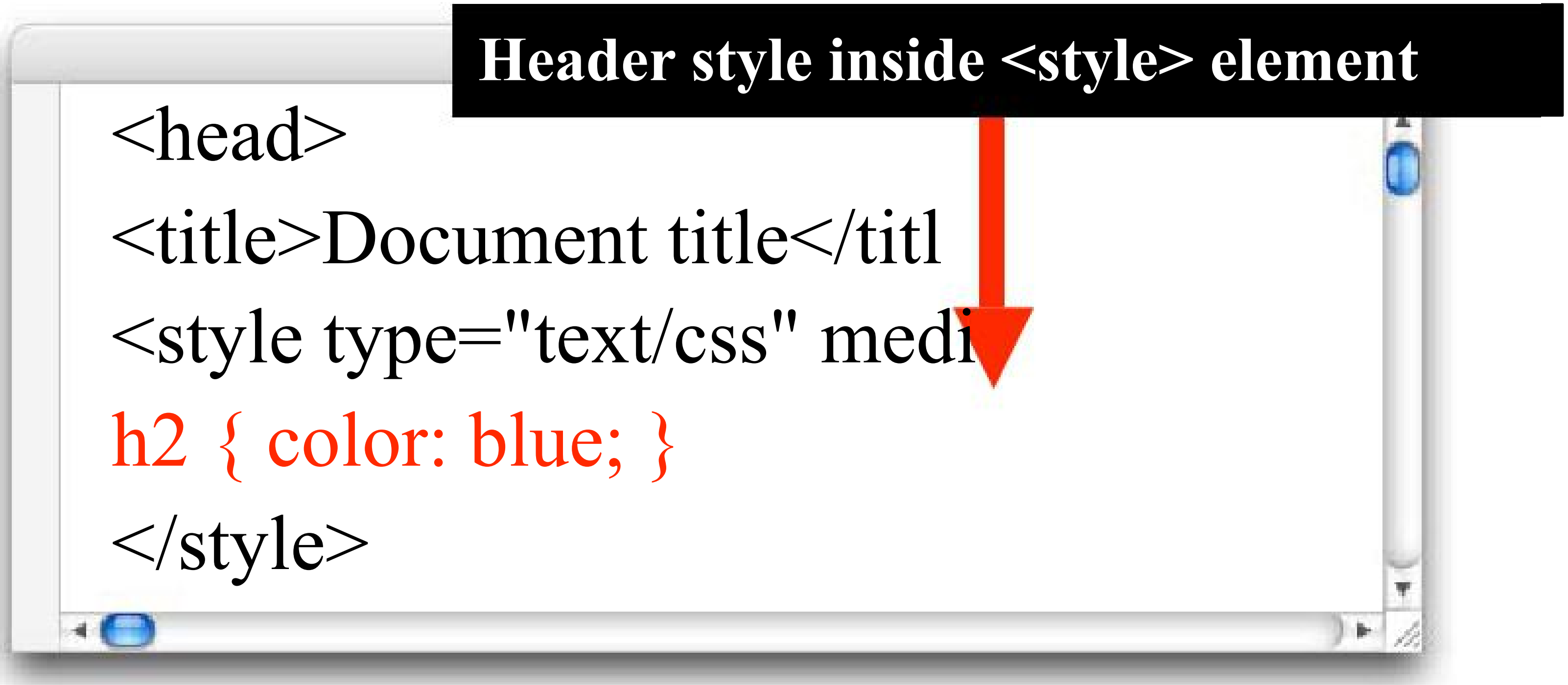
Inline styles are applied to elements in the HTML code using the style attribute.



Inline style using style attribute

```
<body>
<h2 style="color: red;">
        Heading here
</h2>
<p>
```

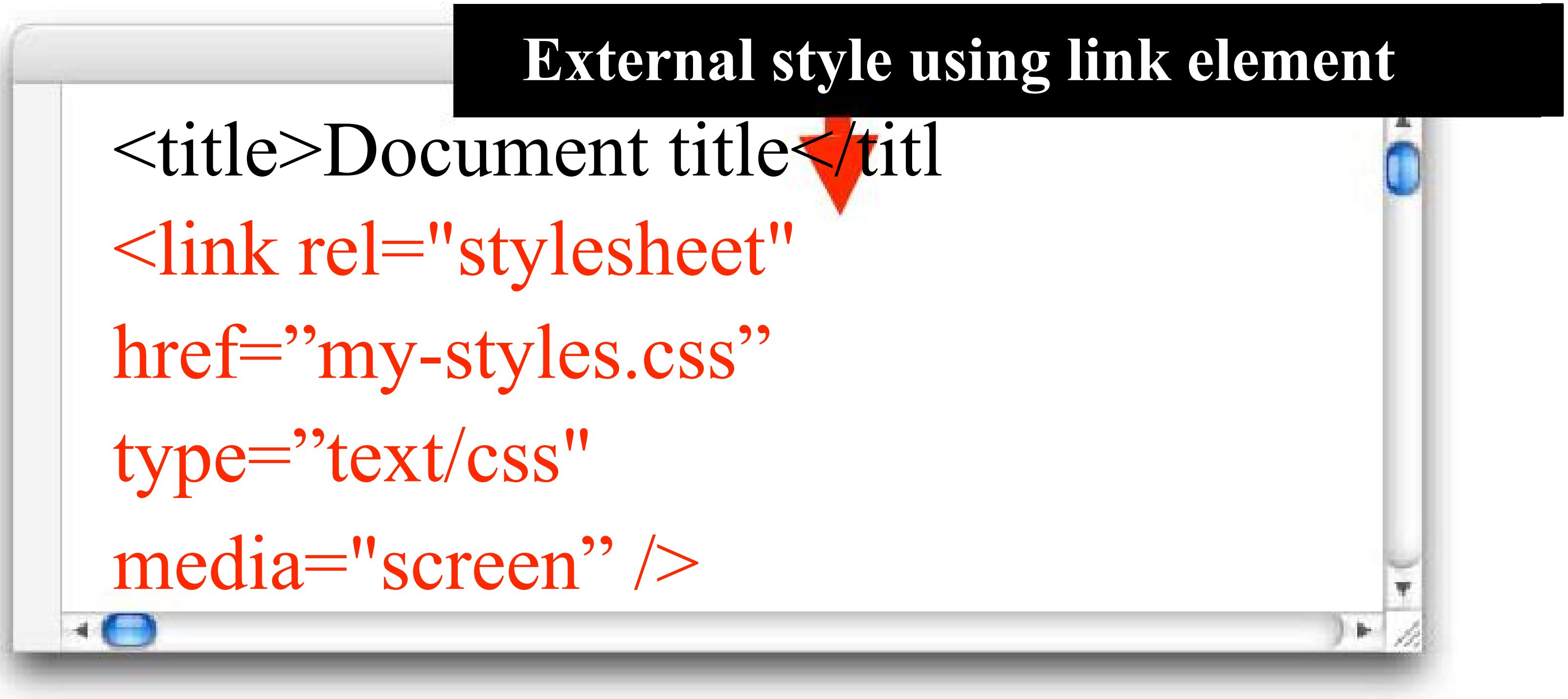
Header styles are placed in the head of the document using the style element



Header style inside <style> element

```
<head>
<title>Document title</titl
<style type="text/css" medi
h2 { color: blue; }
</style>
```

External style sheets are applied using the link or @import.



External style using link element

```
<title>Document title</titl
<link rel="stylesheet"
href=”my-styles.css”
type=”text/css"
media="screen” />
```

CSS rule overload!

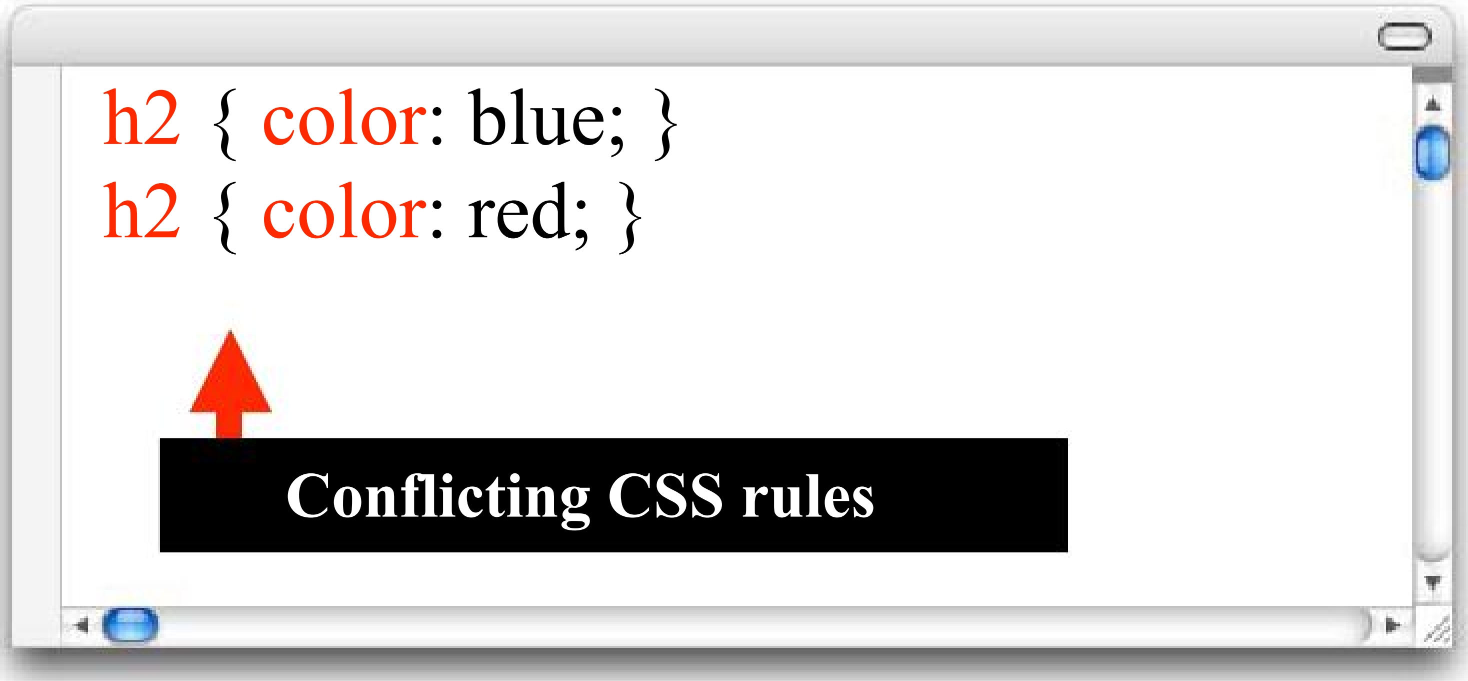
Browsers have to deal with CSS rules coming from the **browser, user and author style sheets**.

Browsers also have to deal with CSS rules coming from different types of **author style sheets** (external, header and inline)

At some point, Browsers have to deal with CSS rules that **conflict**.

What does “conflict” mean?

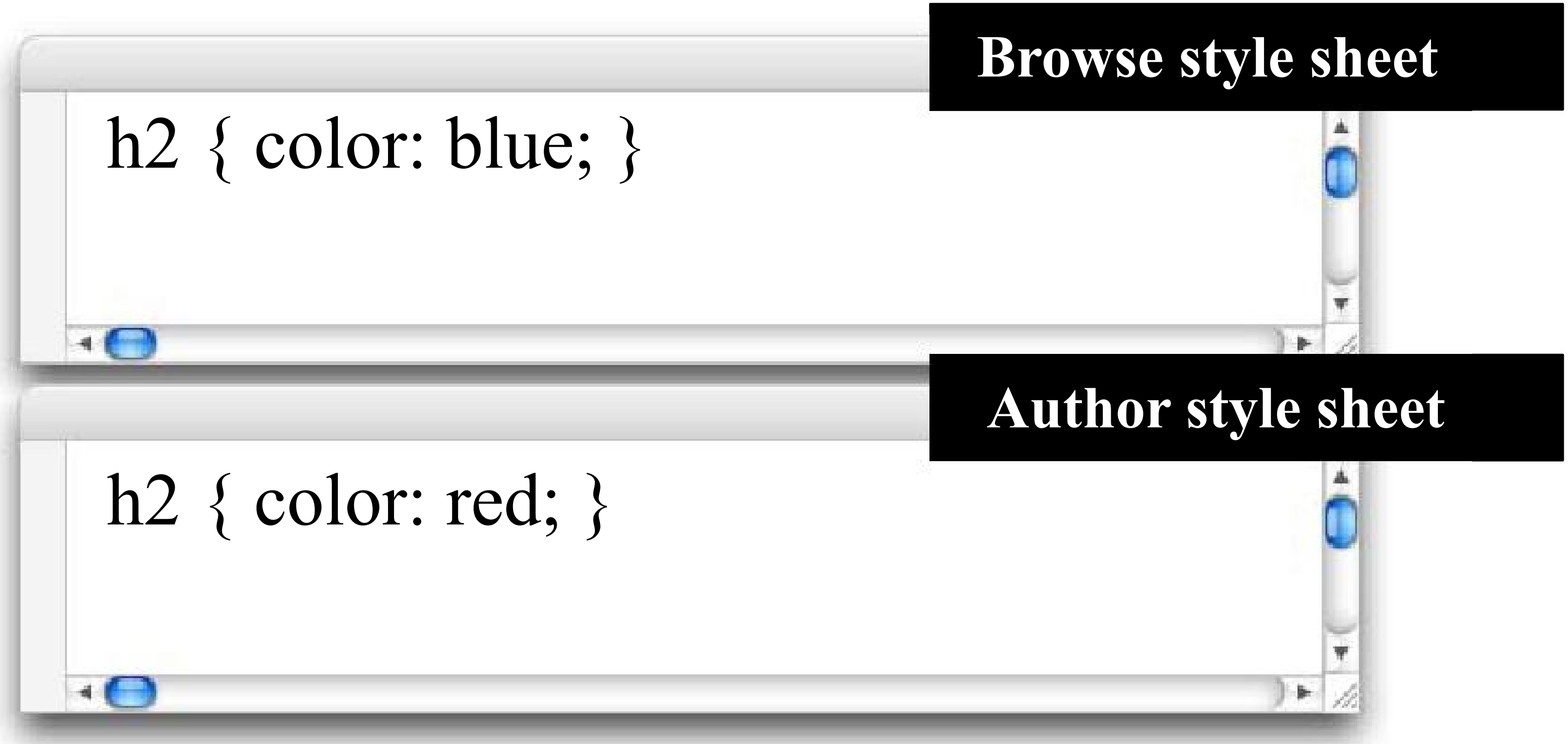
Conflict is where more than one CSS rule refers to the same **element** and **property**.



```
h2 { color: blue; }
h2 { color: red; }
```

Conflicting CSS rules

Conflict can occur between CSS rules in **different types of style sheets**.



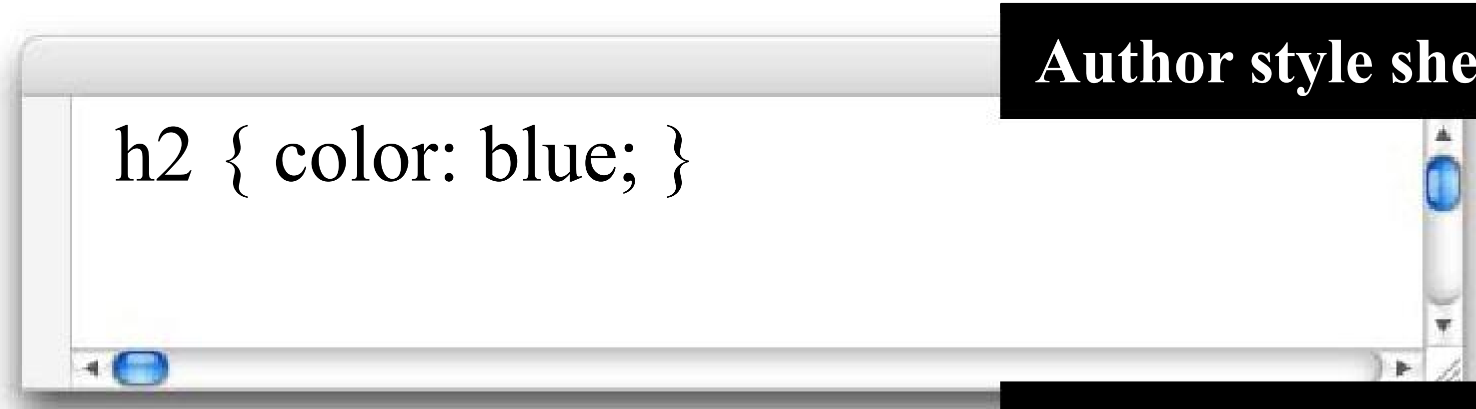
Browse style sheet

```
h2 { color: blue; }
```

Author style sheet

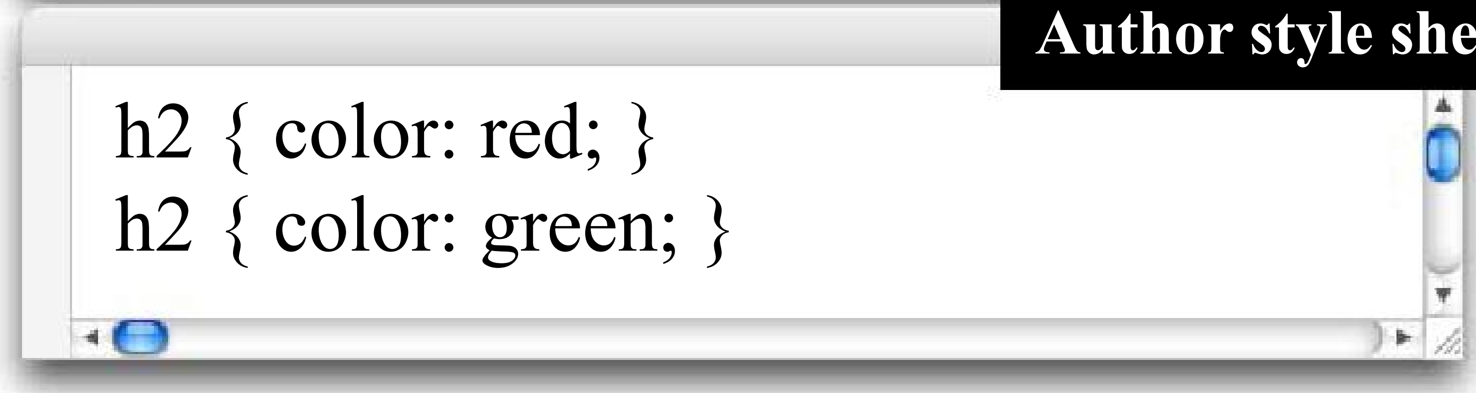
```
h2 { color: red; }
```

Conflict can occur between CSS rules in within the **one or more author style sheets**.



Author style sheet 1

```
h2 { color: blue; }
```



Author style sheet 2

```
h2 { color: red; }
h2 { color: green; }
```

So which CSS rules "win"?

There are **four steps** to determine which CSS rules will “win” (be applied to an HTML document)

Step 1

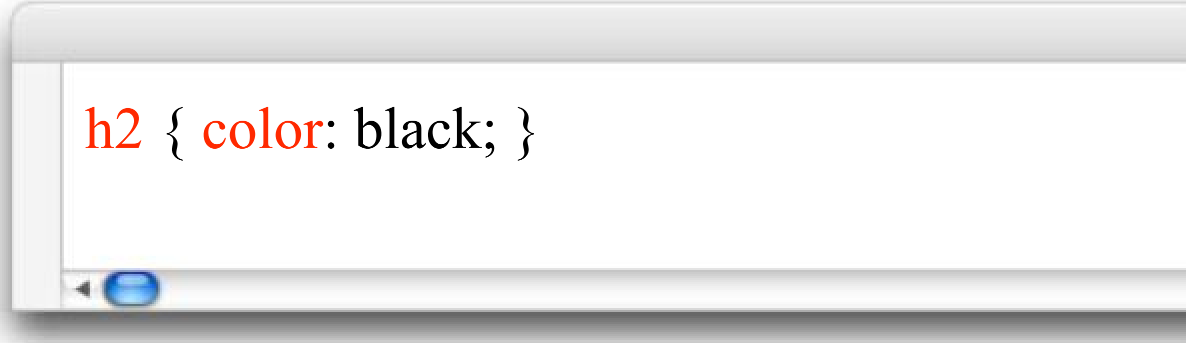
Gather all the **declarations** that apply to an **element and property** from browser, author and user style sheets

For example, find any **declarations** that matches:

element = h2
property = color

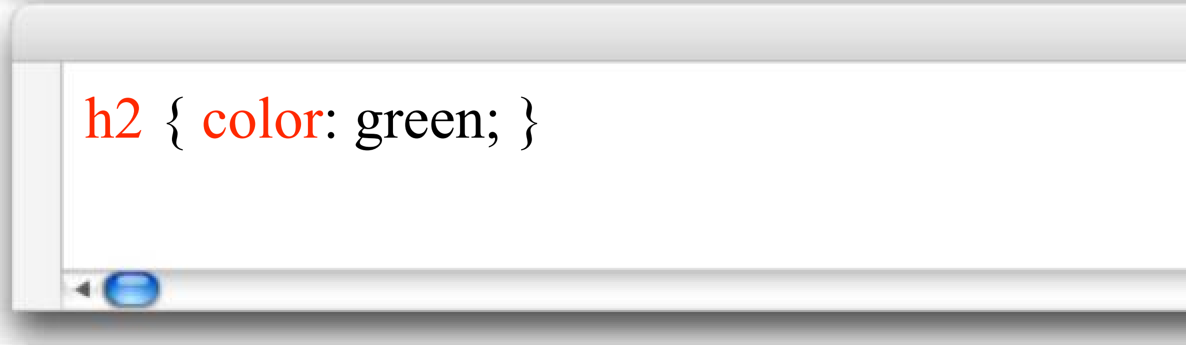
Gathered declarations

Browser style sheet



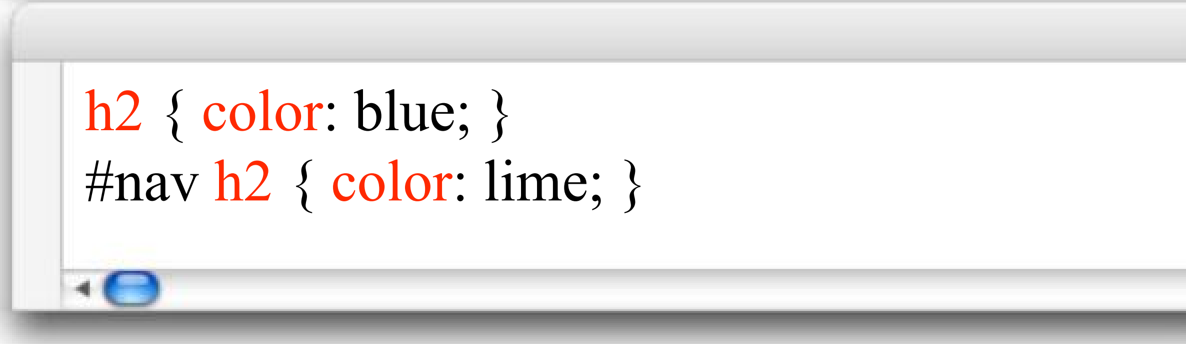
```
h2 { color: black; }
```

User style sheet



```
h2 { color: green; }
```

Author style sheets



```
h2 { color: blue; }
#nav h2 { color: lime; }
```

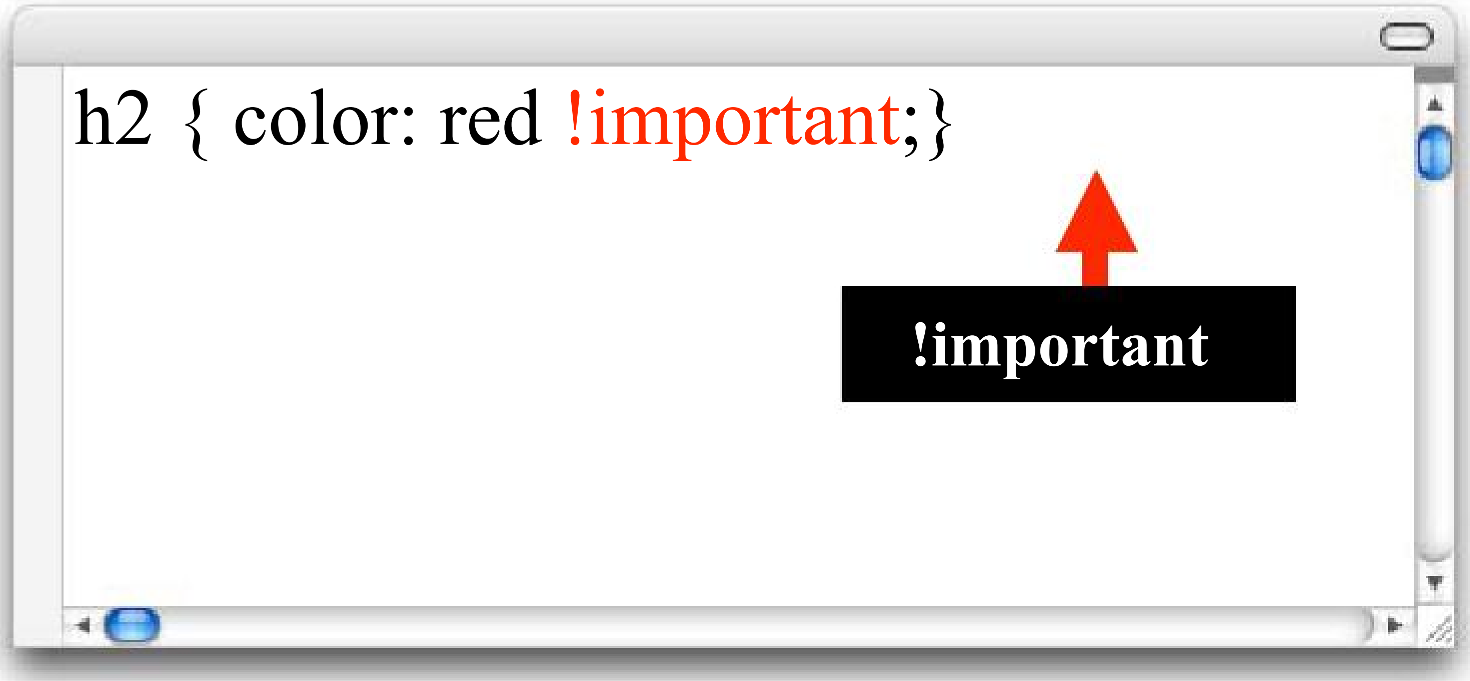
If there are declarations from **more than one of these three sources**, proceed to step 2.

Step 2

Sort the gathered declarations according to **origin** (browser, author, user style sheets) and **importance** (normal or !important).

What is !important?

Authors can assign “**!important**” to any declaration.



```
h2 { color: red !important;}
```

"!important" declarations

override normal declarations

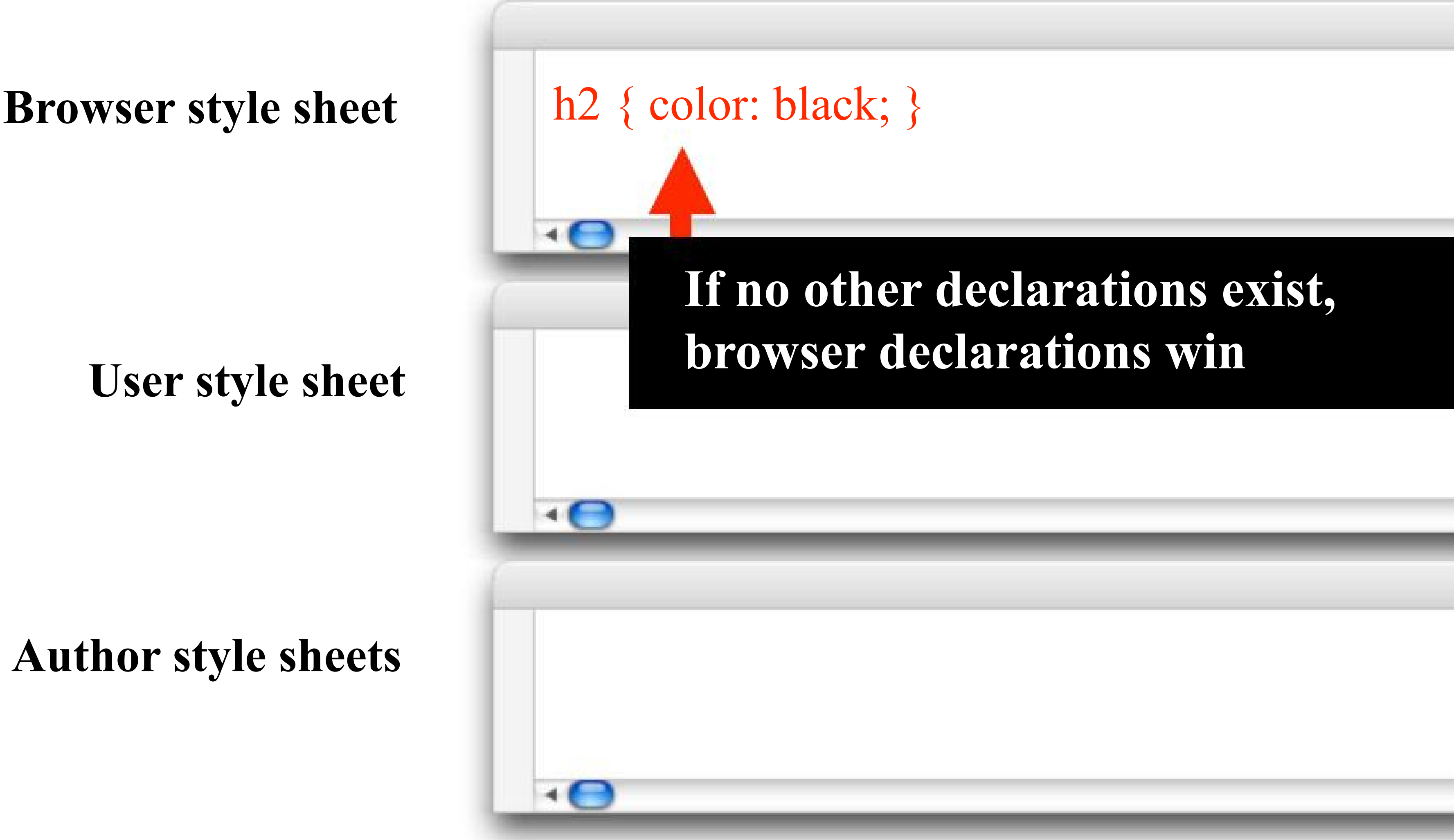
(Normal declarations are declarations that do not contain !important).

So, how are declarations sorted?

From lowest to highest **priority**

1. browser styles
2. normal declarations in user style sheet
3. normal declarations in author style sheet
4. !important declarations in author style sheet
5. !important declarations in user style sheet

1. Browser styles



Browser style sheet

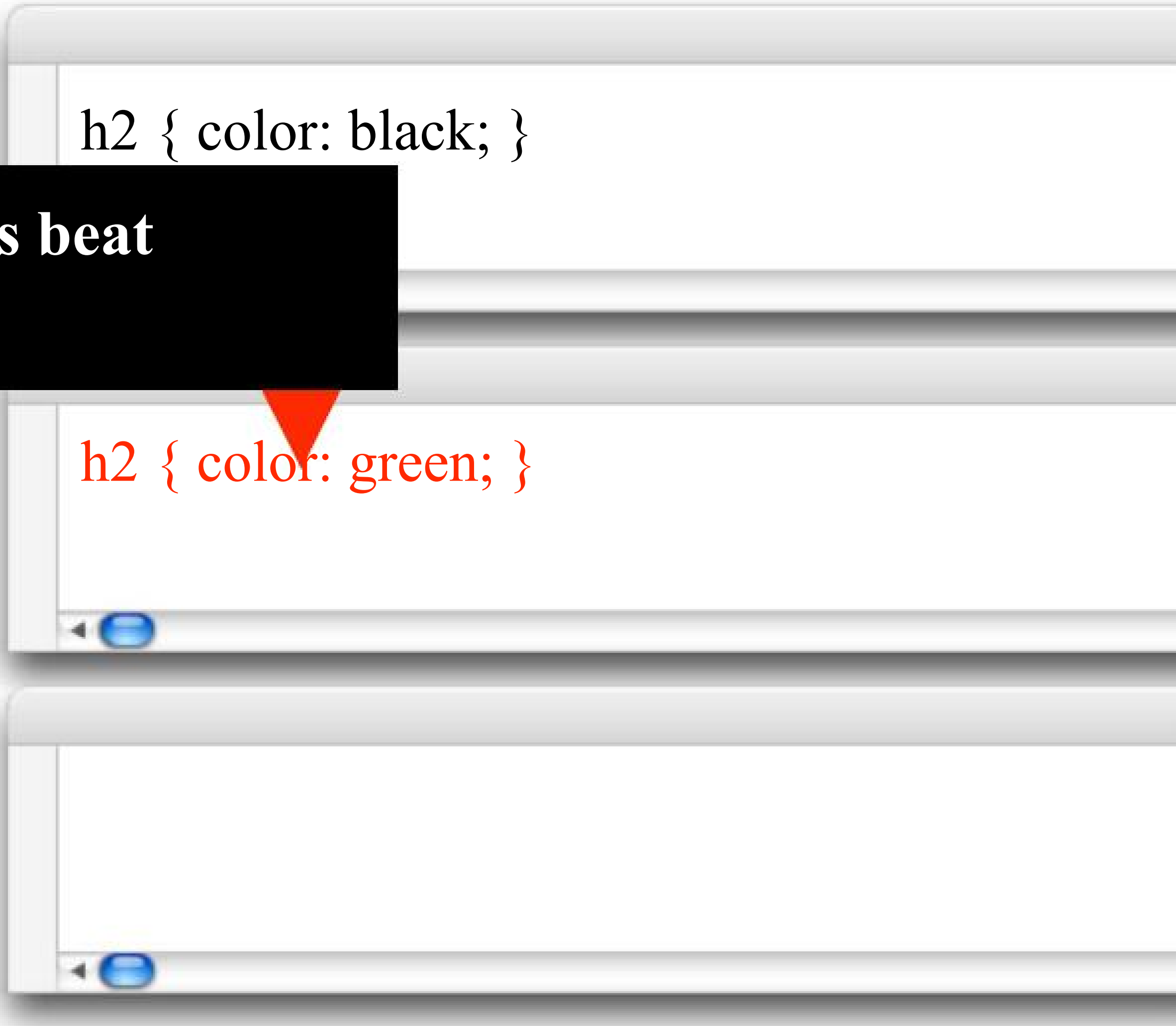
```
h2 { color: black; }
```

If no other declarations exist, browser declarations win

User style sheet

Author style sheets

2. Normal user styles



Browser style sheet

```
h2 { color: black; }
```

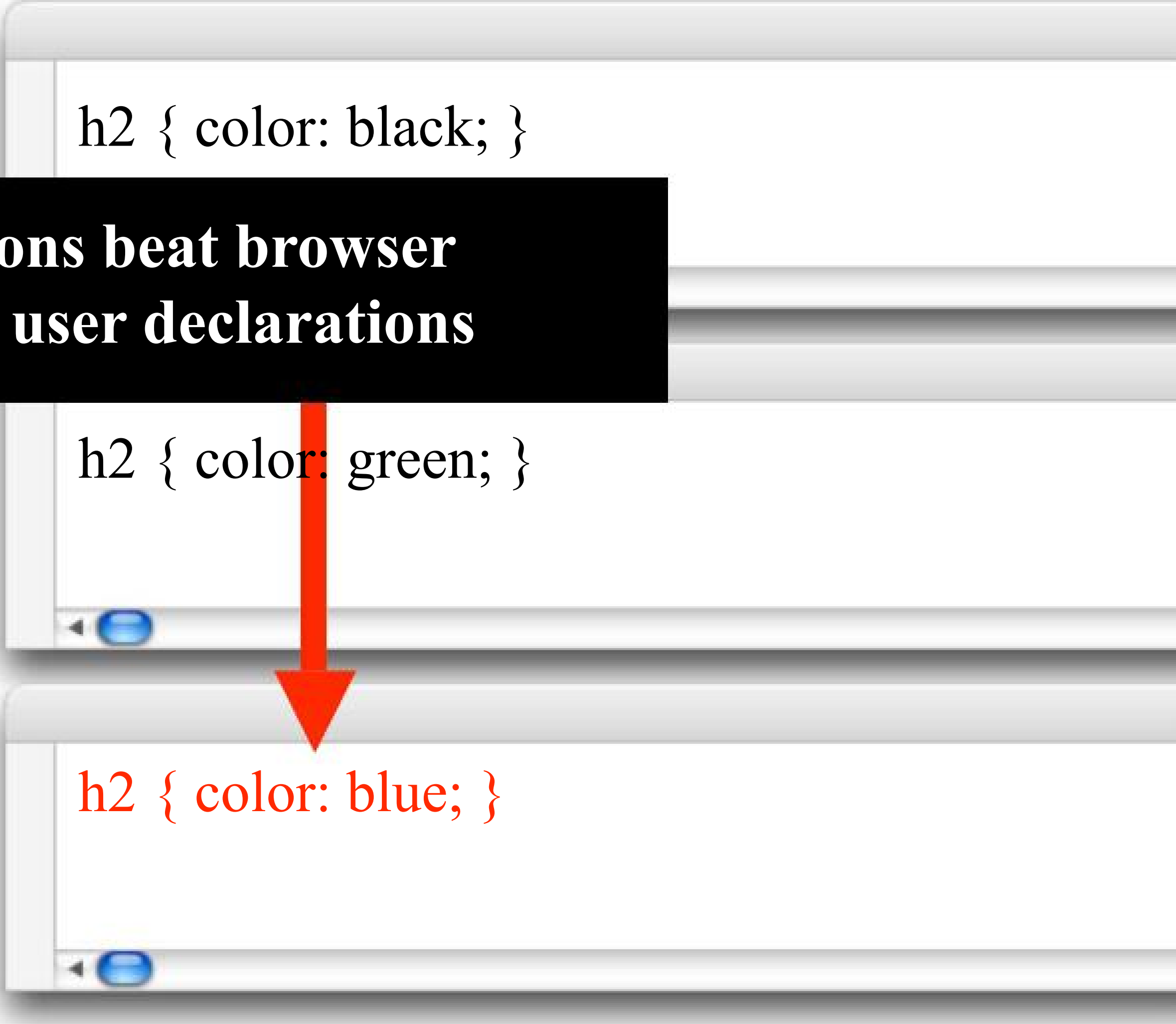
Normal user declarations beat browser declarations

User style sheet

```
h2 { color: green; }
```

Author style sheets

3. Normal author styles



Browser style sheet

```
h2 { color: black; }
```

Normal author declarations beat browser declarations and normal user declarations

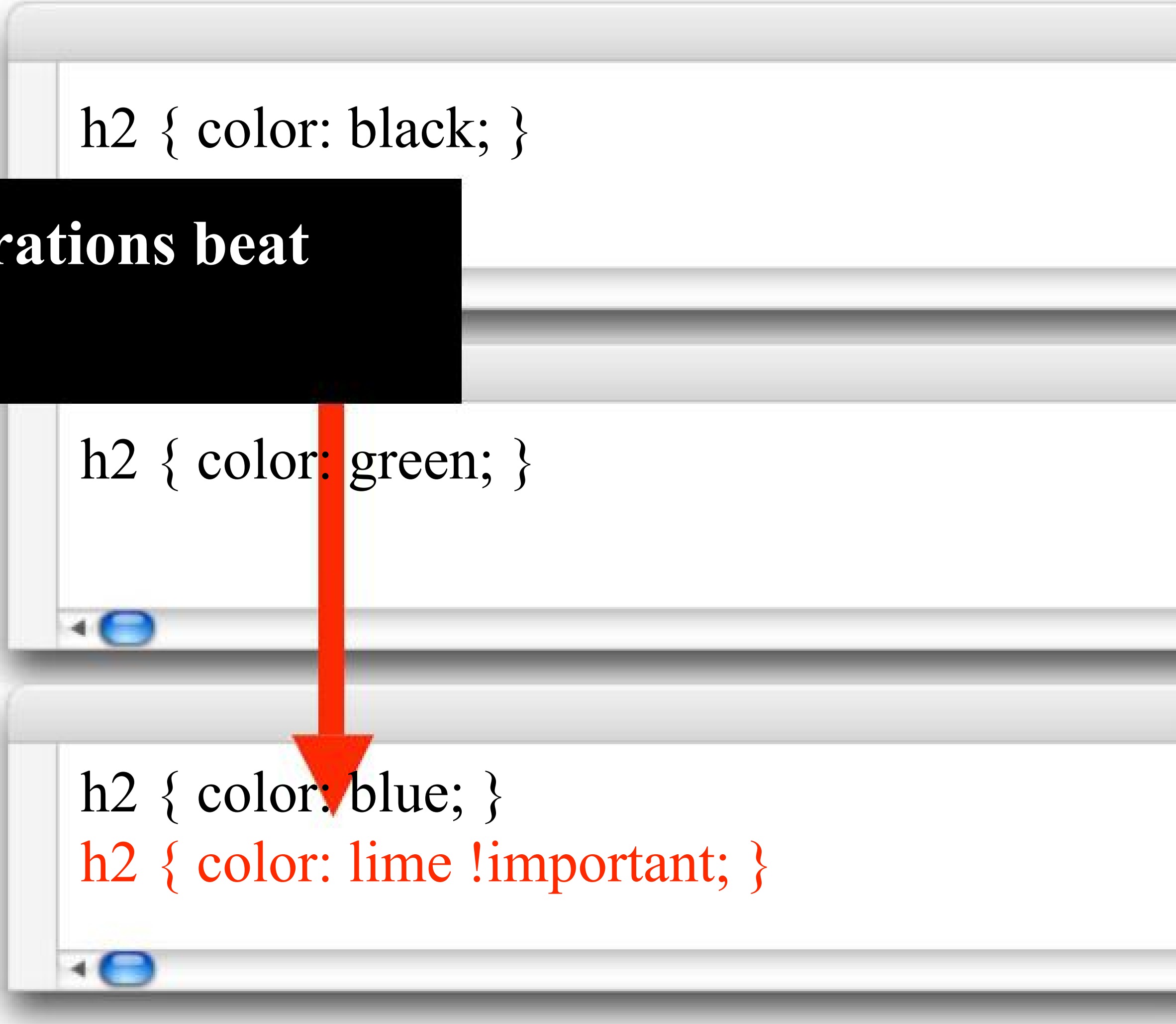
User style sheet

```
h2 { color: green; }
```

Author style sheets

```
h2 { color: blue; }
```

4. !important author styles



Browser style sheet

```
h2 { color: black; }
```

!important author declarations beat all normal declarations

User style sheet

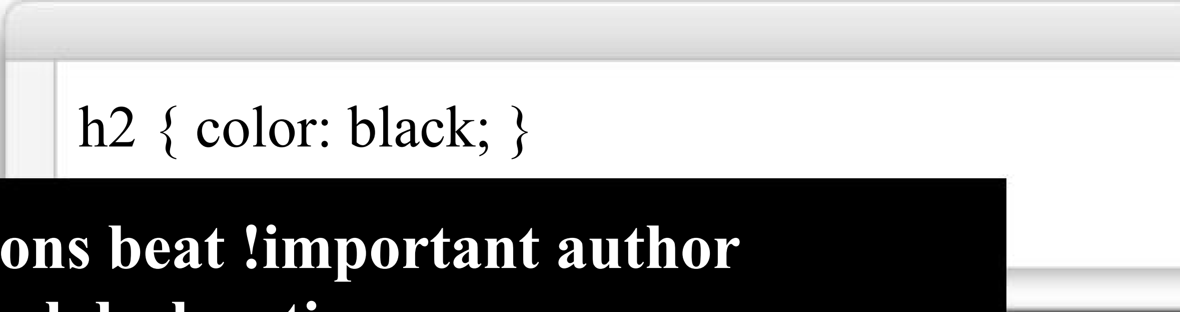
```
h2 { color: green; }
```

Author style sheets

```
h2 { color: blue; }
h2 { color: lime !important; }
```

5. !important user styles

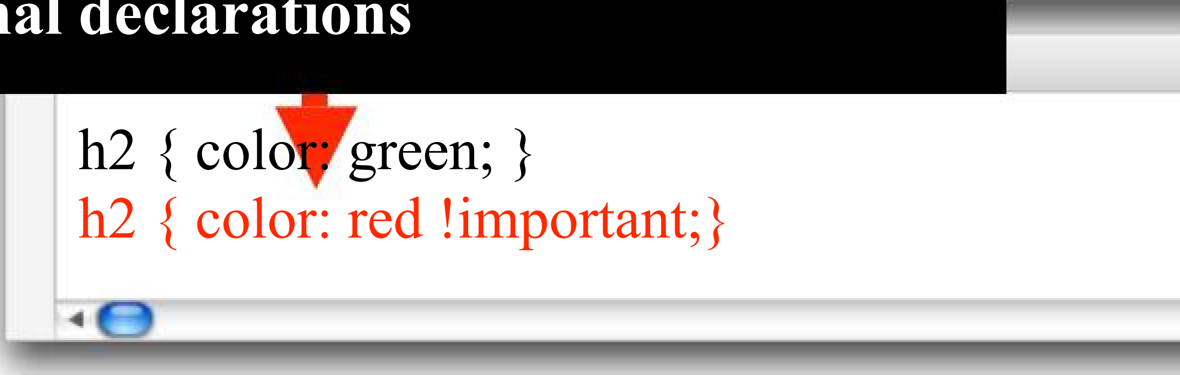
Browser style sheet



```
h2 { color: black; }
```

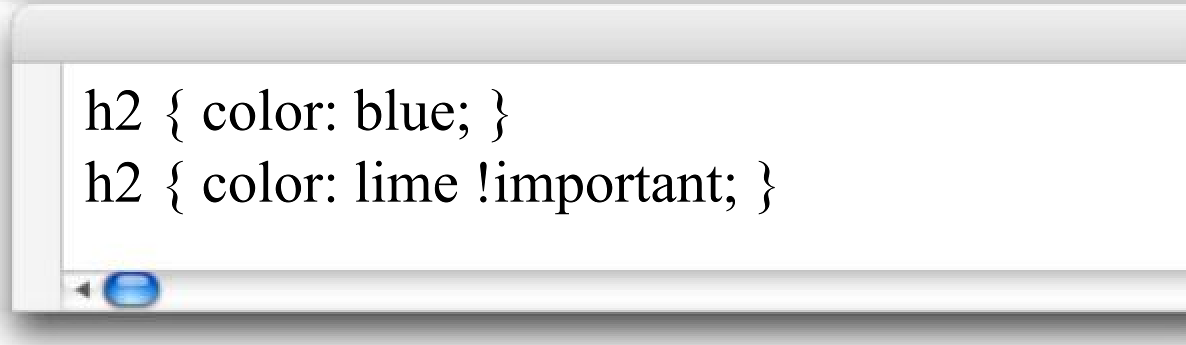
!important user declarations beat !important author declarations and all normal declarations

User style sheet



```
h2 { color: green; }
h2 { color: red !important;}
```

Author style sheets

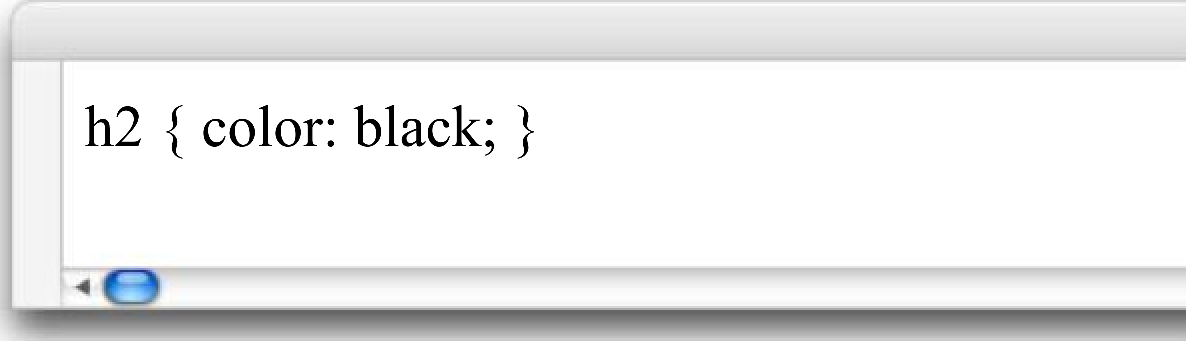


```
h2 { color: blue; }
h2 { color: lime !important; }
```

But what if two declarations have **the same origin or importance**?

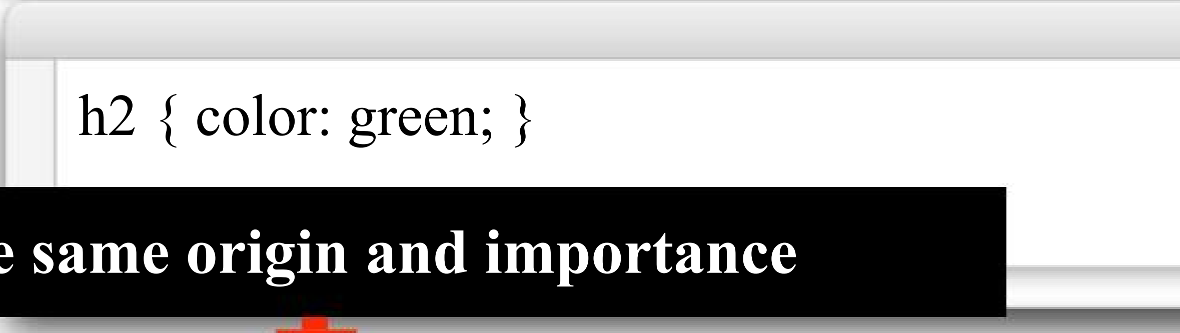
Two matching declarations

Browser style sheet



```
h2 { color: black; }
```

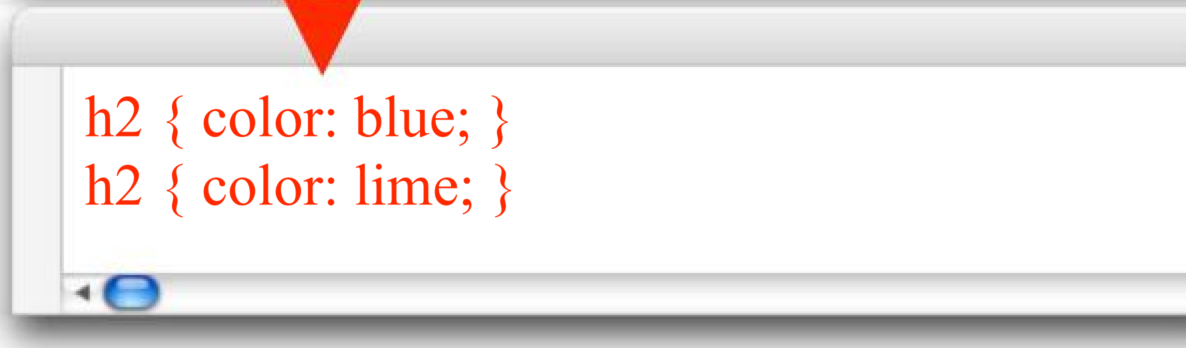
User style sheet



```
h2 { color: green; }
```

Two declarations with the same origin and importance

Author style sheets

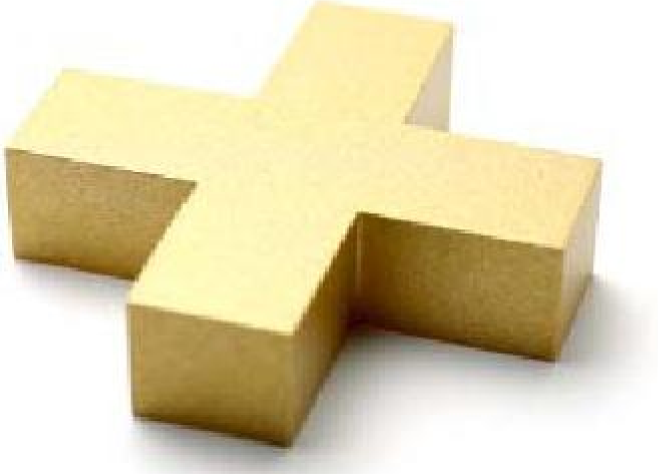


```
h2 { color: blue; }
h2 { color: lime; }
```

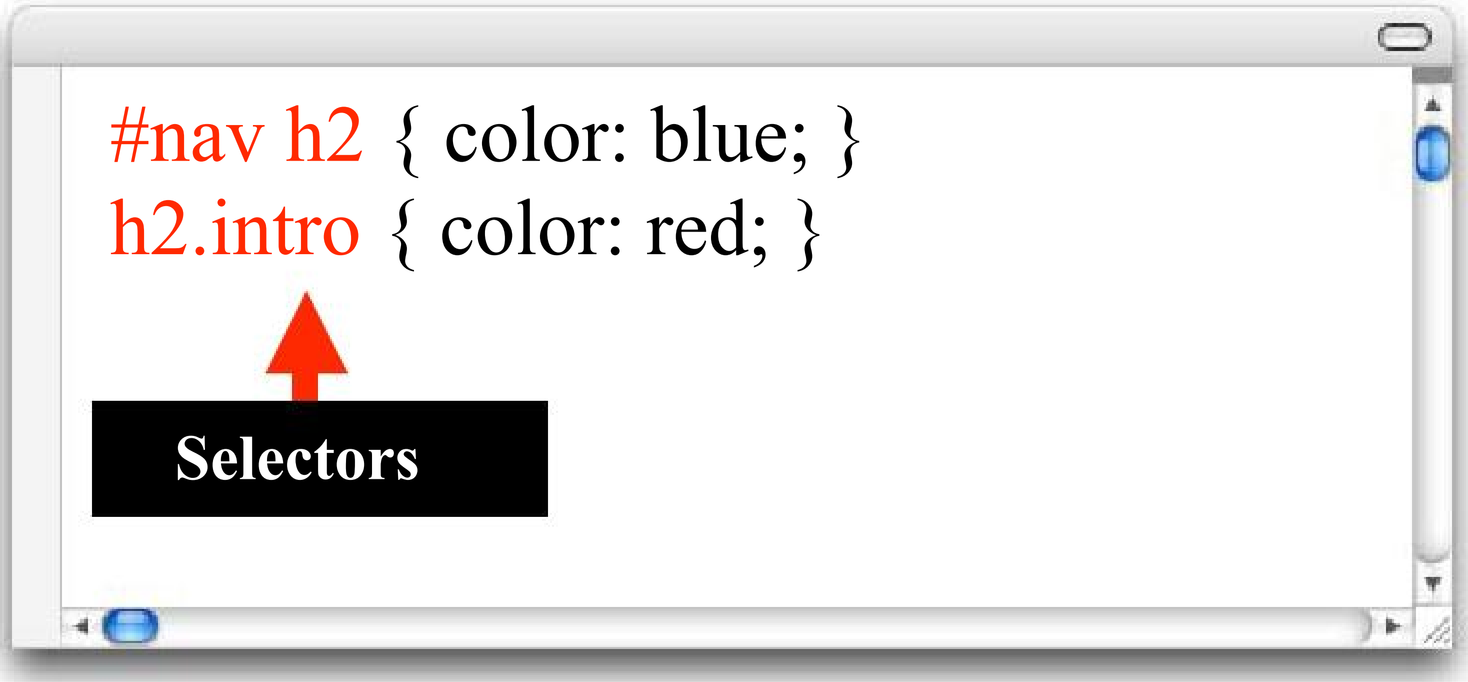
If declarations have **the same origin or importance** then proceed to Step 3.

Step 3

If declarations have the same origin or importance then the **declaration's selectors** need to be scored, to see which declaration will "win".



Selectors



```
#nav h2 { color: blue; }
h2.intro { color: red; }
```

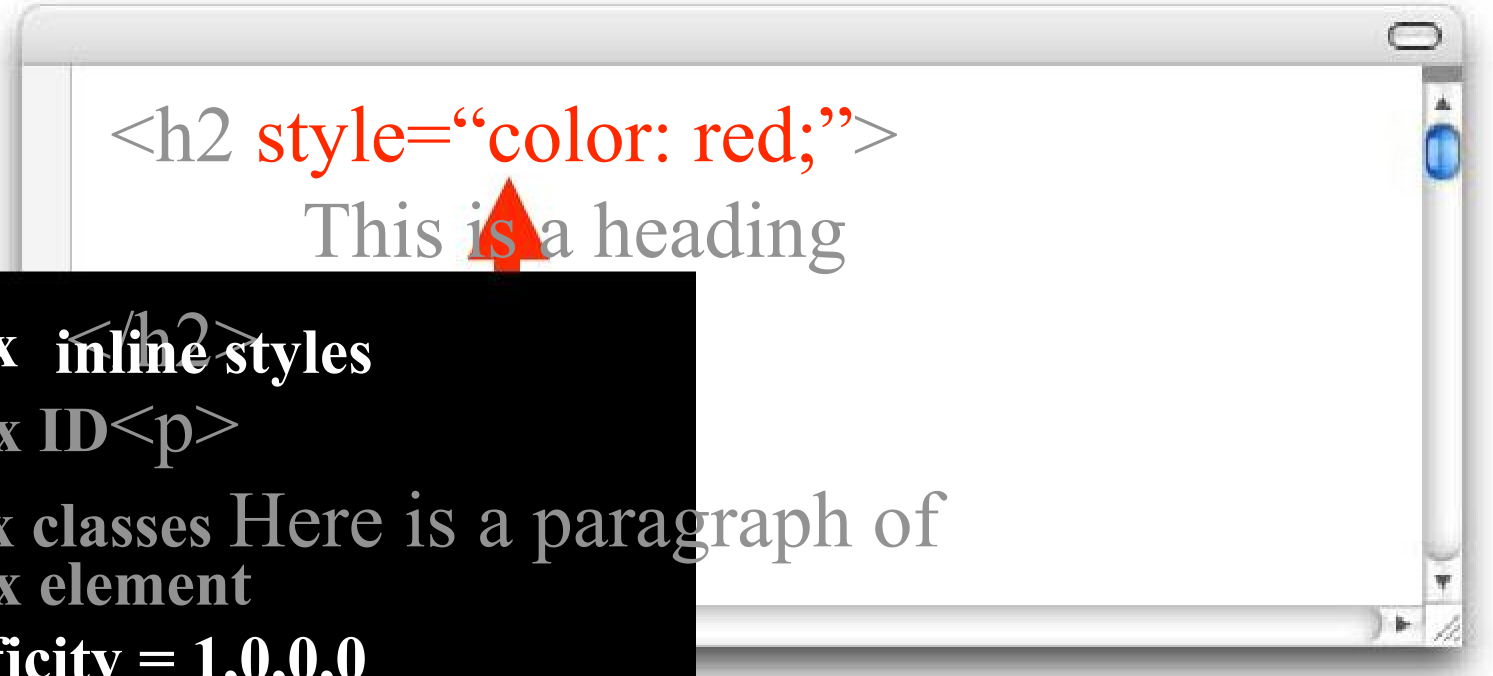
Four scores are **concatenated** (linked together as a chain) to create a final score.

a,b,c,d

This score is referred to as a selector’s **specificity**.

So how is specificity calculated?

A. Is there an inline style?



```
<h2 style=“color: red;”>
        This is a heading
</h2>
<p>
        Here is a paragraph of
```

a = 1 x inline styles

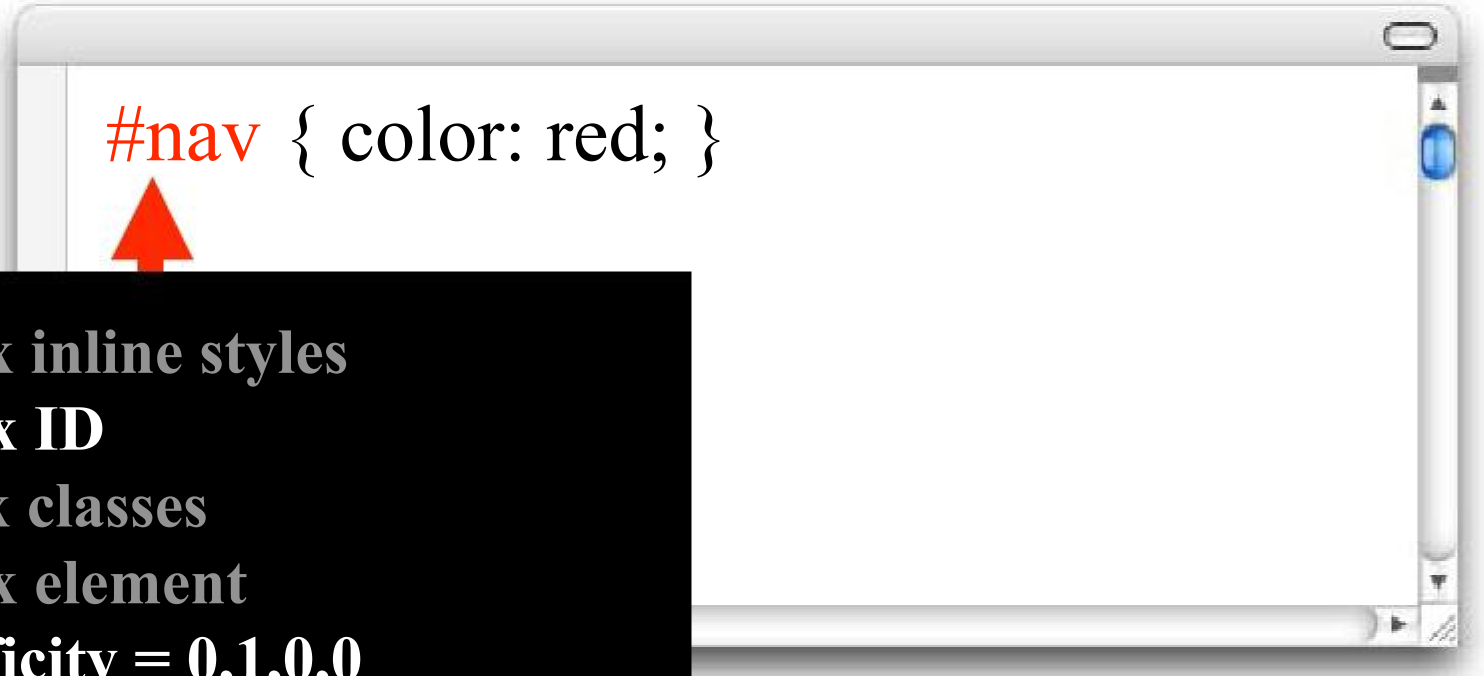
b = 0 x ID

c = 0 x classes

d = 0 x element

Specificity = 1,0,0,0

B. Count the number of IDs in the selectors.



```
#nav { color: red; }
```

a = 0 x inline styles

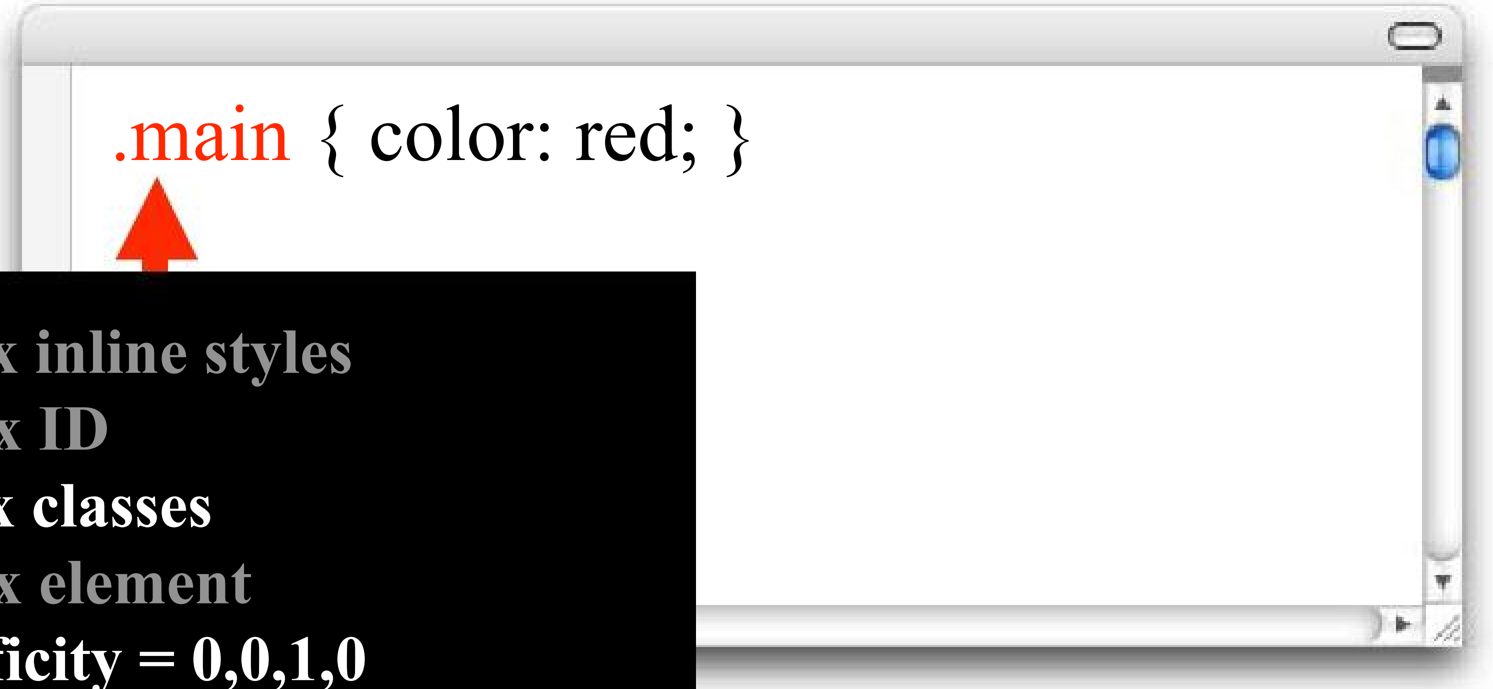
b = 1 x ID

c = 0 x classes

d = 0 x element

Specificity = 0,1,0,0

C. Count the number of classes, attributes and pseudo-classes.



```
.main { color: red; }
```

a = 0 x inline styles

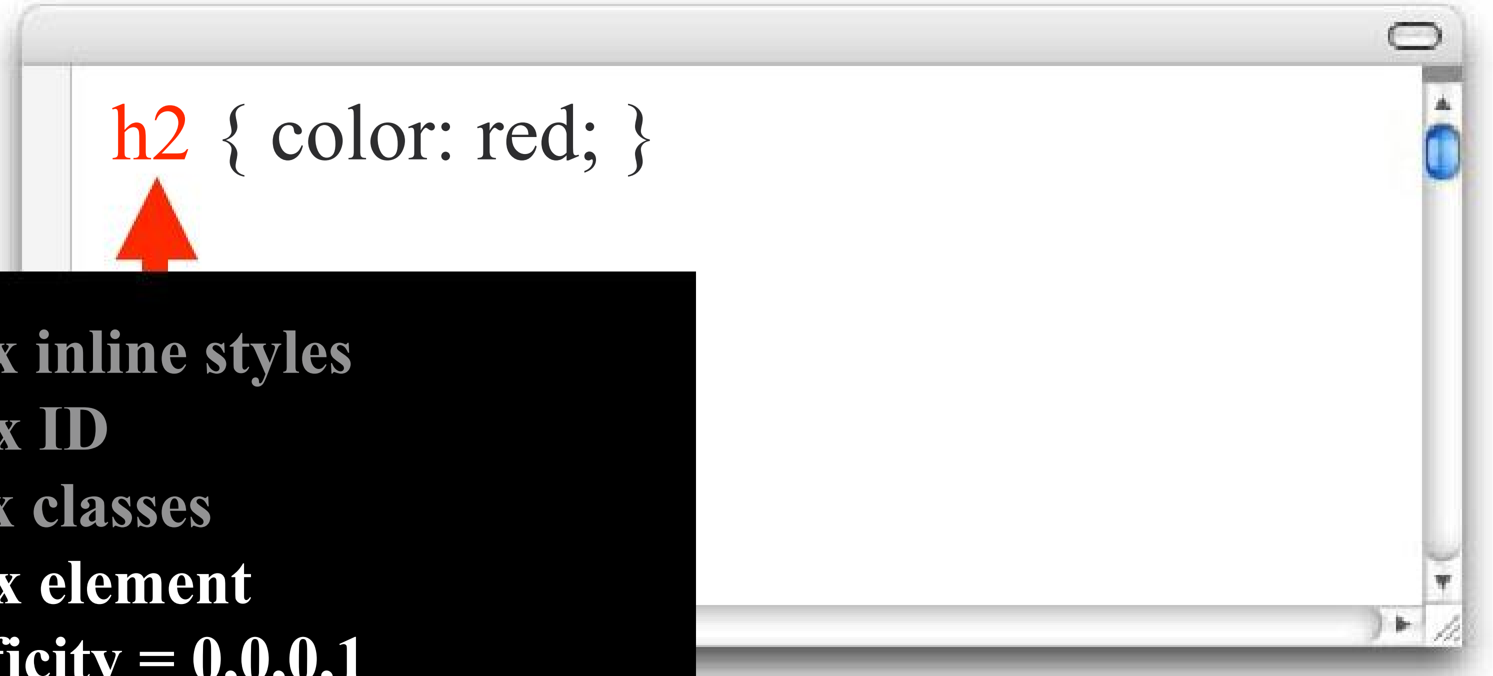
b = 0 x ID

c = 1 x classes

d = 0 x element

Specificity = 0,0,1,0

D. Count the number of element names or pseudo-elements.



```
h2 { color: red; }
```

a = 0 x inline styles

b = 0 x ID

c = 0 x classes

d = 1 x element

Specificity = 0,0,0,1

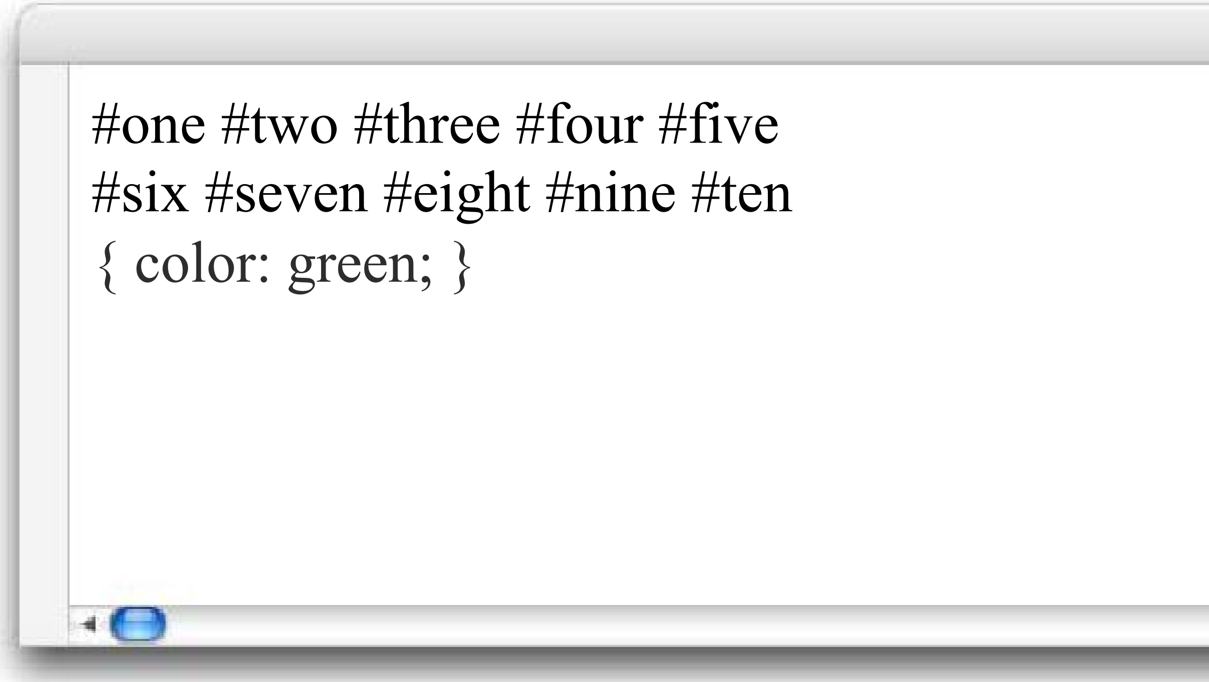
A note on concatenation

“**A**” will always beat “**B**”, which will always beat “**C**”, which will always beat “**D**”.

No matter how many **IDs** are used in a selector, an **inline style** will always win.

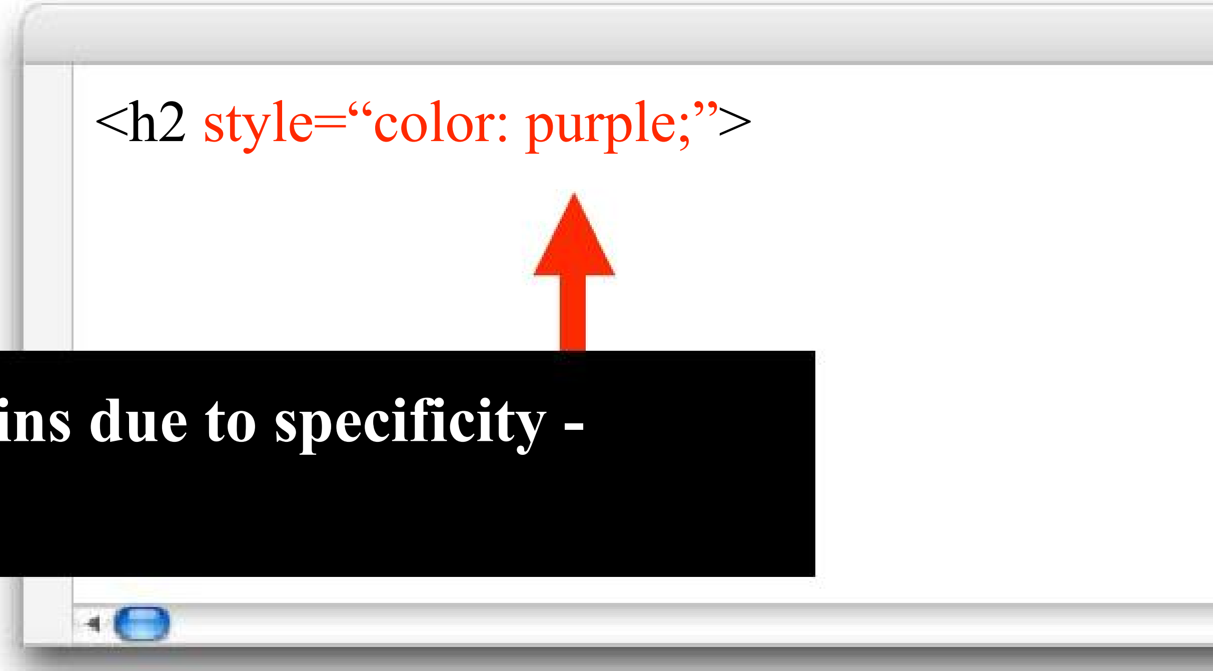
(unless !important is used within the ID’s declaration)

External style sheets and header styles (Author styles)



```
#one #two #three #four #five
#six #seven #eight #nine #ten
{ color: green; }
```

HTML document with inline styles (Author styles)

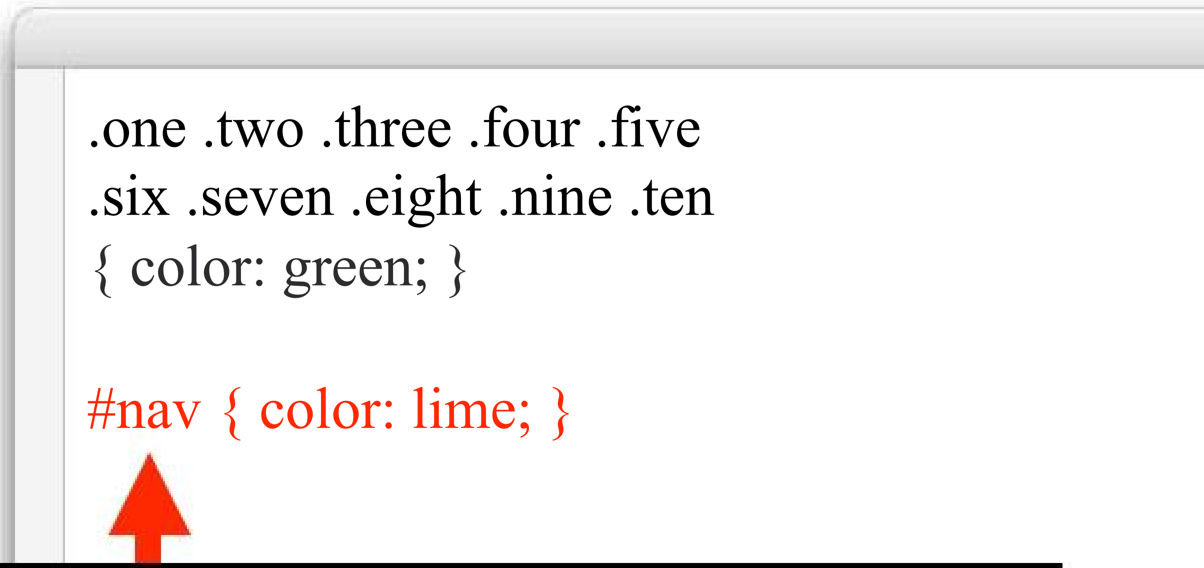


```
<h2 style="color: purple;">
```

The highlighted style wins due to specificity - 1,0,0,0 beats 0,10,0,0

No matter how many **classes** are applied to a selector, an **ID** can easily win

External style sheets and header styles (Author styles)



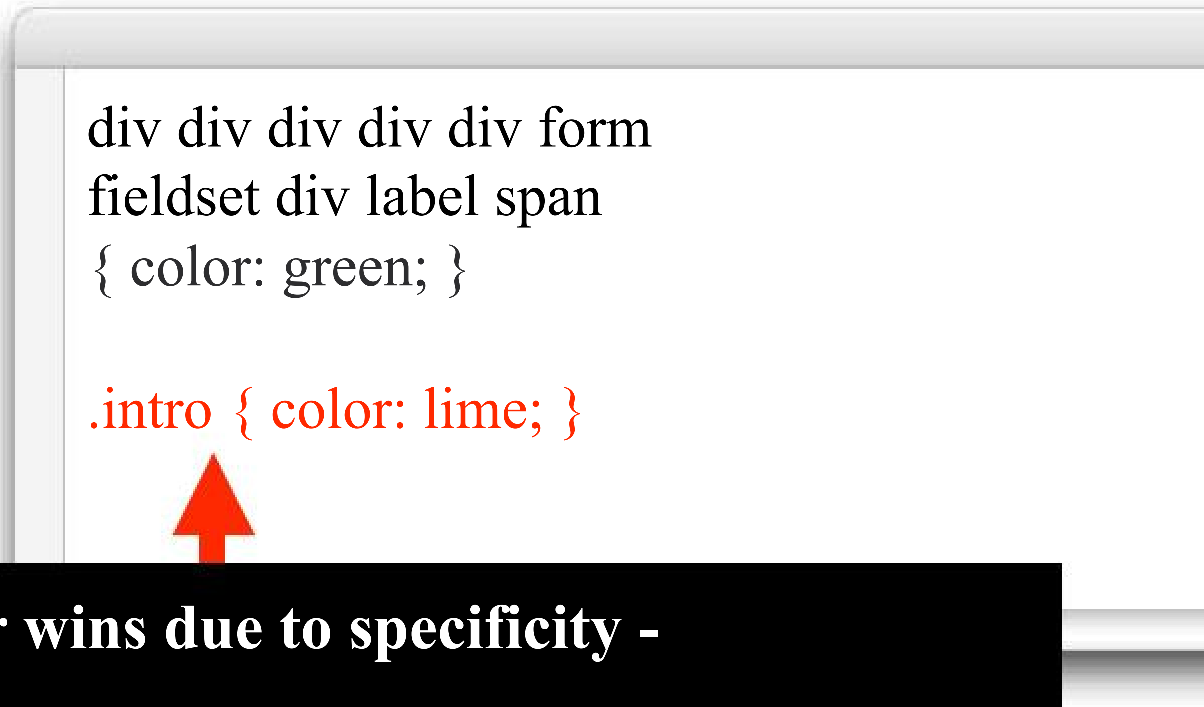
```
.one .two .three .four .five
.six .seven .eight .nine .ten
{ color: green; }

#nav { color: lime; }
```

The highlighted selector wins due to specificity - 0,1,0,0 beats 0,0,10,0

No matter how many **elements** are applied to a selector, a **class** can easily win.

External style sheets and header styles (Author styles)



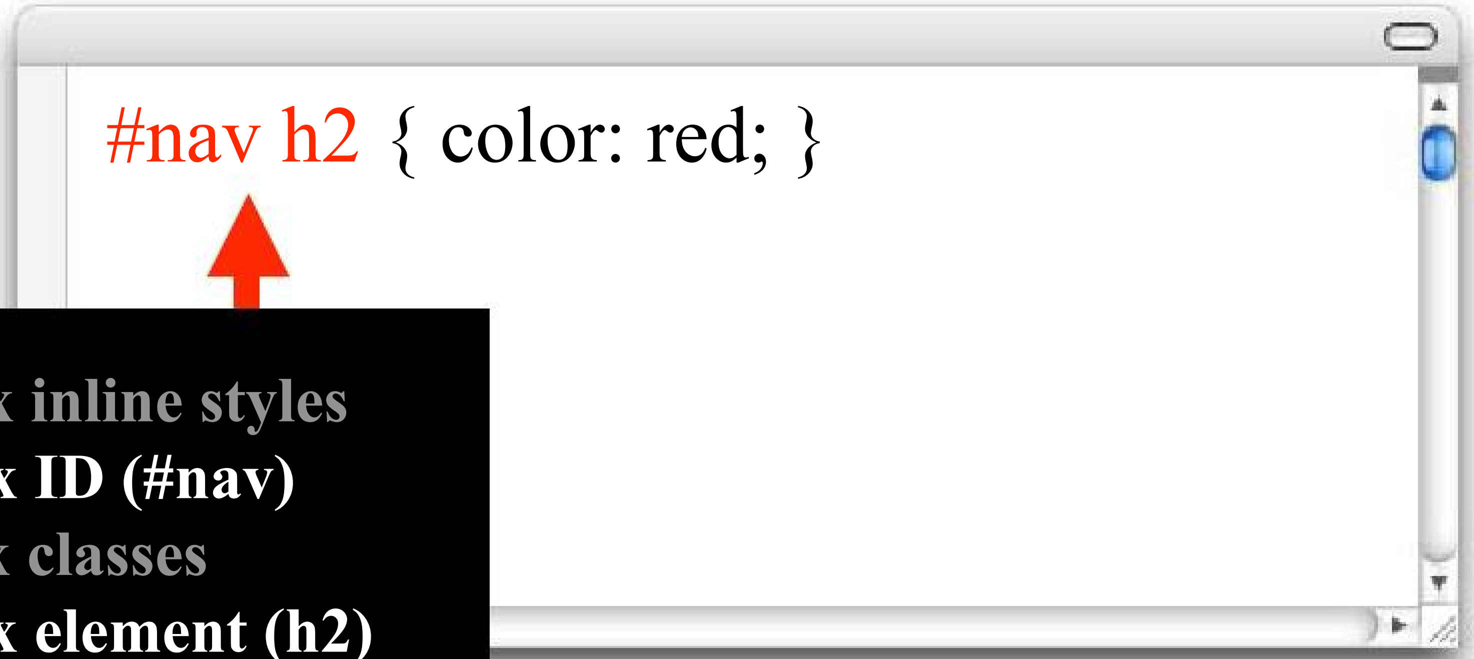
```
div div div div div form
fieldset div label span
{ color: green; }

.intro { color: lime; }
```

The highlighted selector wins due to specificity - 0,0,1,0 beats 0,0,0,10

Complex examples of specificity

ID and element



```
#nav h2 { color: red; }
```

a = 0 x inline styles

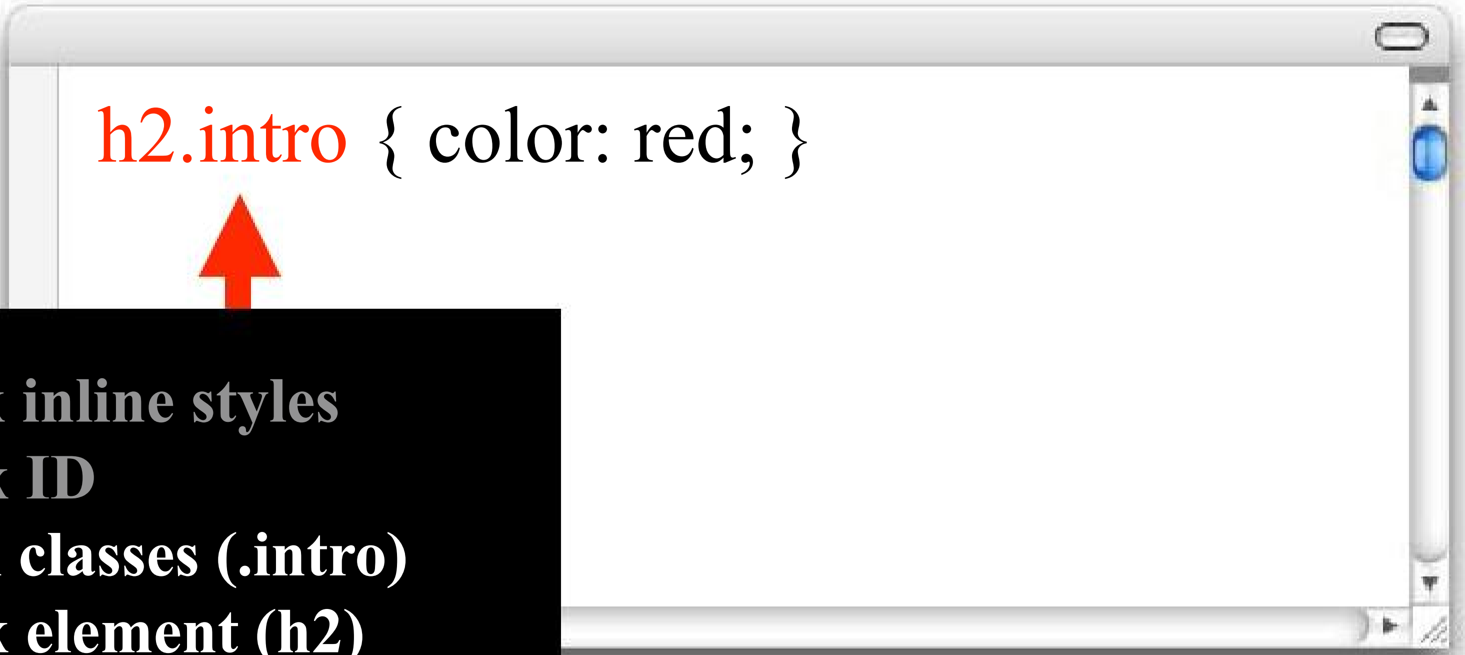
b = 1 x ID (#nav)

c = 0 x classes

d = 1 x element (h2)

Specificity = 0,1,0,1

Element and class



```
h2.intro { color: red; }
```

a = 0 x inline styles

b = 0 x ID

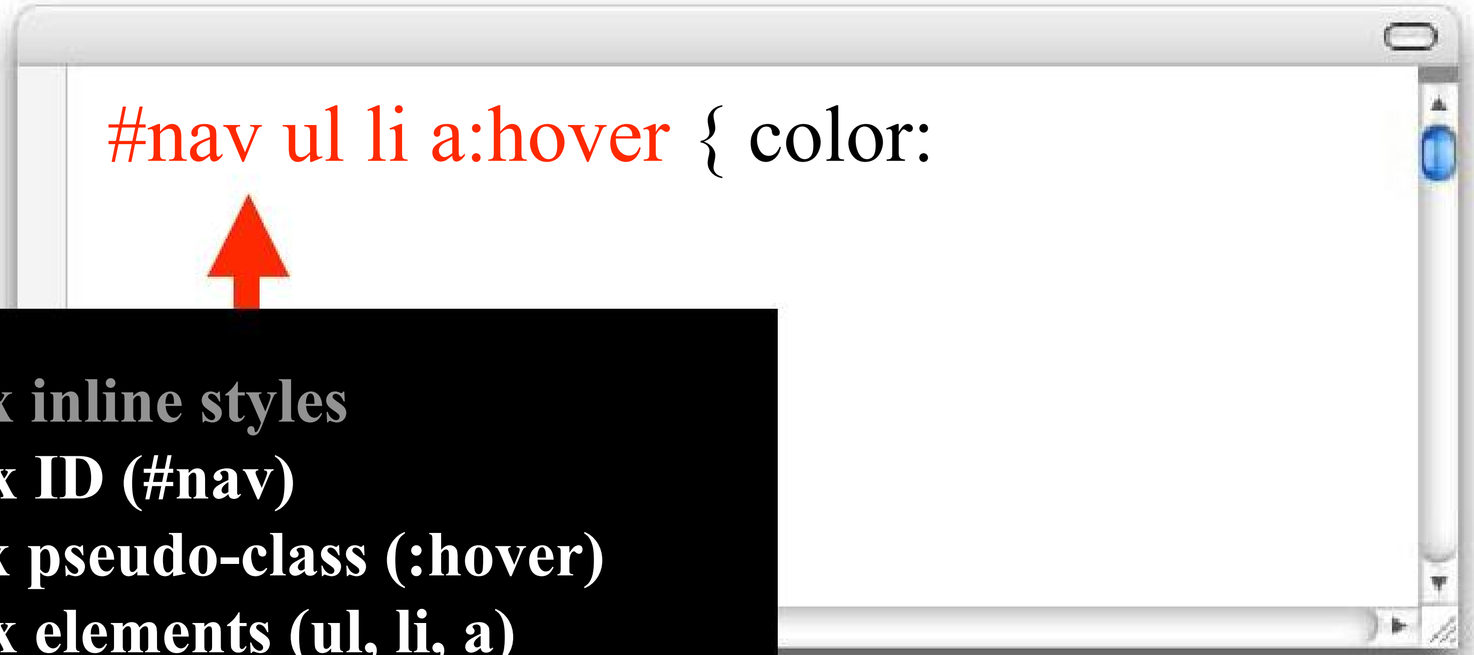
c = 1 x classes (.intro)

d = 1 x element (h2)

Specificity = 0,0,1,1

```
<h2  class="intro"></h2>
<h2  class="abc intro"></h2>
<h2  class="abc intro-a"></h2>
```

ID, elements and pseudo-class



```
#nav ul li a:hover { color:
```

a = 0 x inline styles

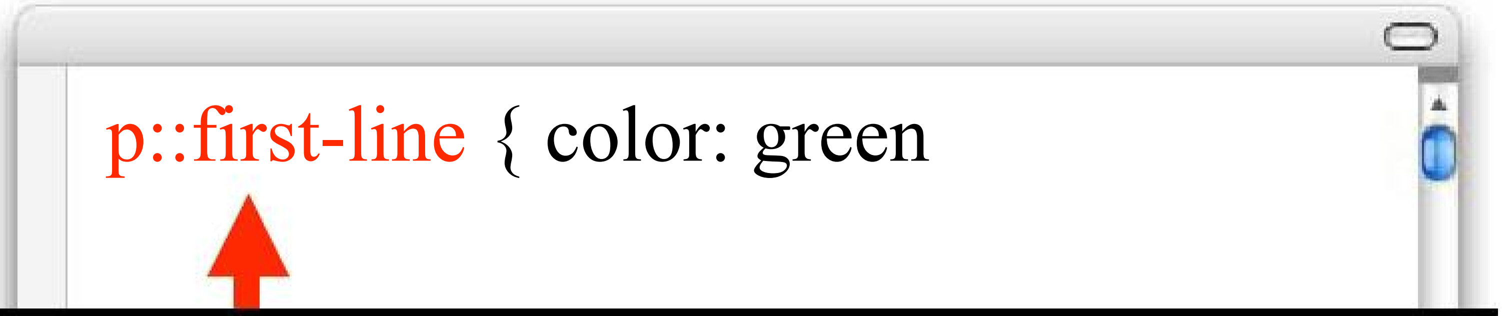
b = 1 x ID (#nav)

c = 1 x pseudo-class (:hover)

d = 3 x elements (ul, li, a)

Specificity = 0,1,1,3

Element and pseudo-element



```
p::first-line { color: green
```

a = 0 x inline styles

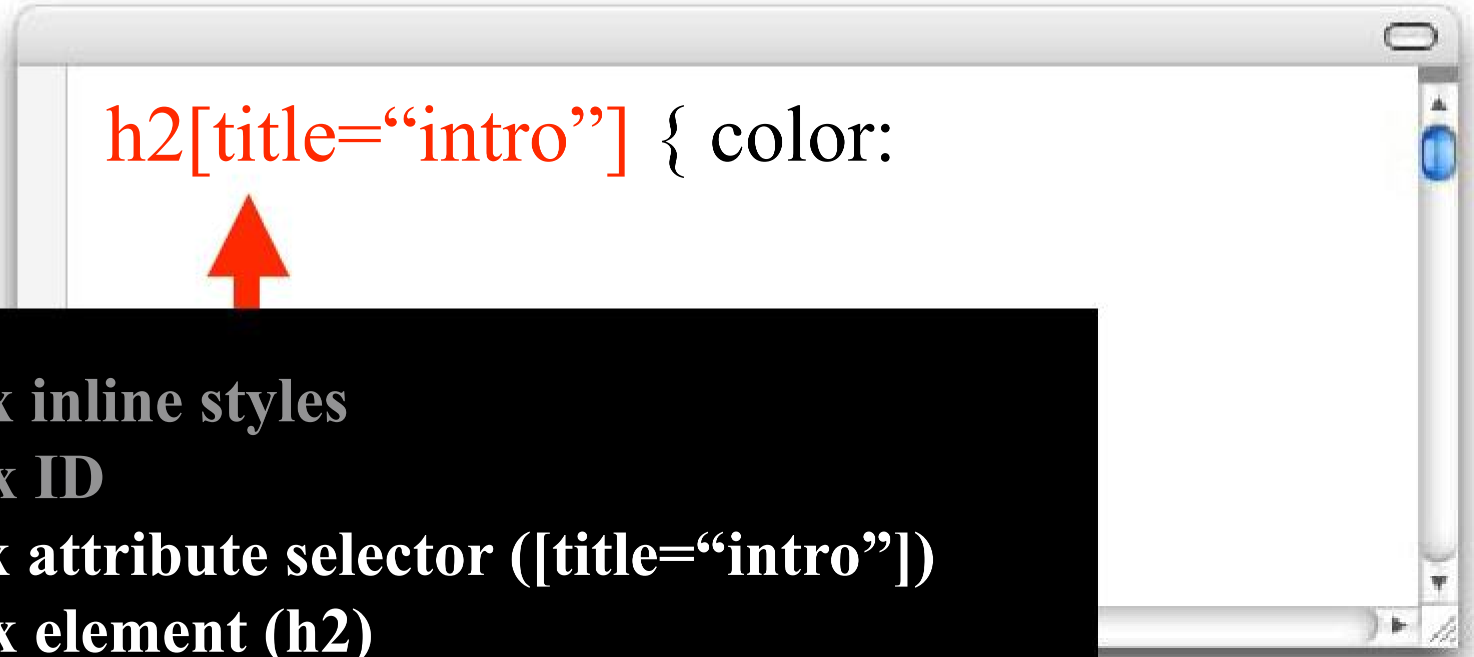
b = 0 x ID

c = 0 x classes

d = 2 x element (p) and pseudo-element (::first-line)

Specificity = 0,0,0,2

Element and attribute selector

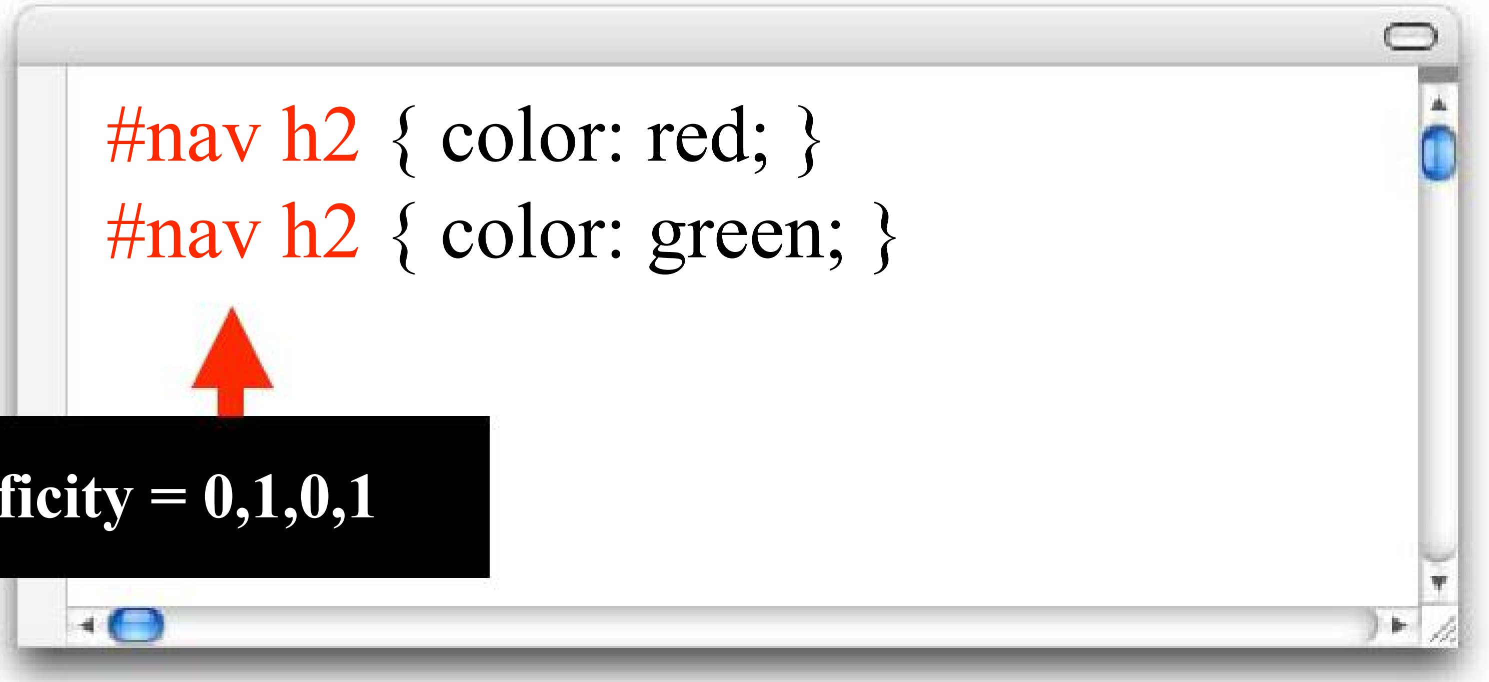


```
h2[title=“intro”] { color:
```

a = 0 x inline styles
b = 0 x ID
c = 1 x attribute selector ([title=“intro”])
d = 1 x element (h2)
Specificity = 0,0,1,1

What if there is still no clear winner?

Selectors with same specificity



```
#nav h2 { color: red; }
#nav h2 { color: green; }
```

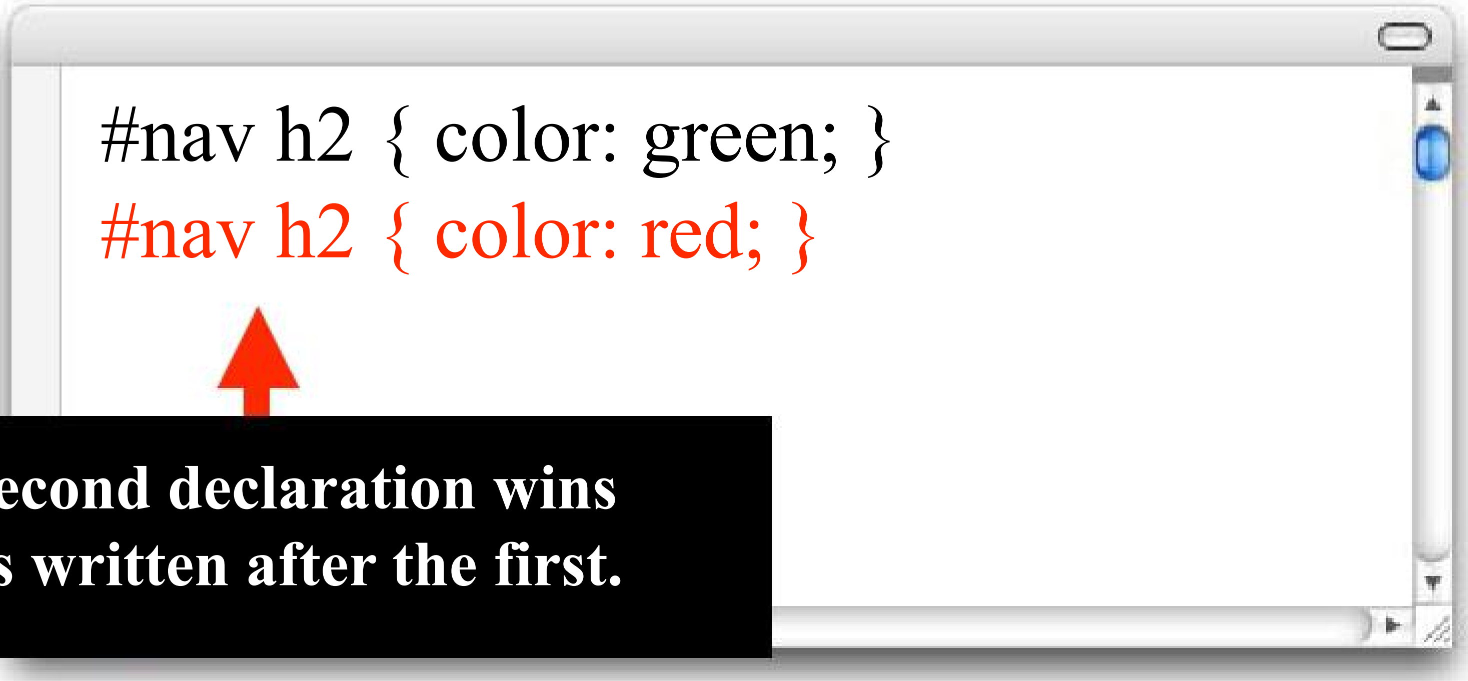
Specificity = 0,1,0,1

If there is still **no clear winner** then proceed to Step 4.

Step 4

If two declarations have the same importance, origin and specificity, the **latter specified** declaration wins

Equal-weight declarations



```
#nav h2 { color: green; }
#nav h2 { color: red; }
```

The second declaration wins as it is written after the first.

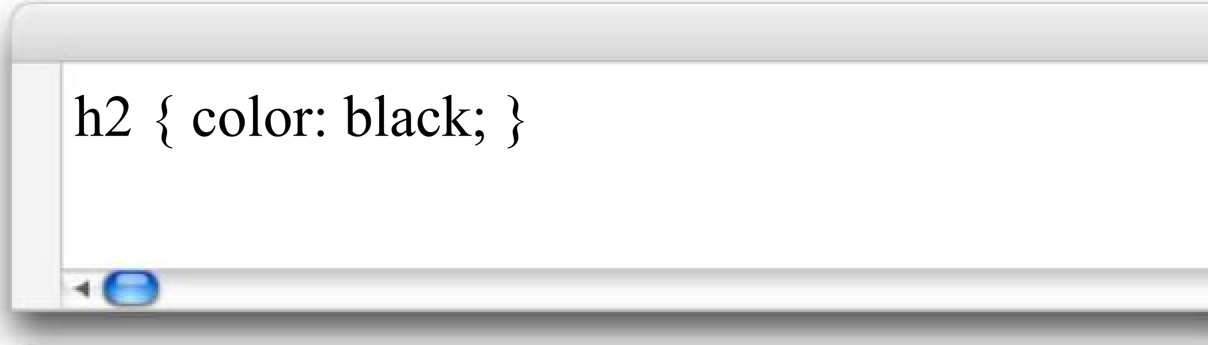
And now… a guessing game

Exercise 1

browser, user, author

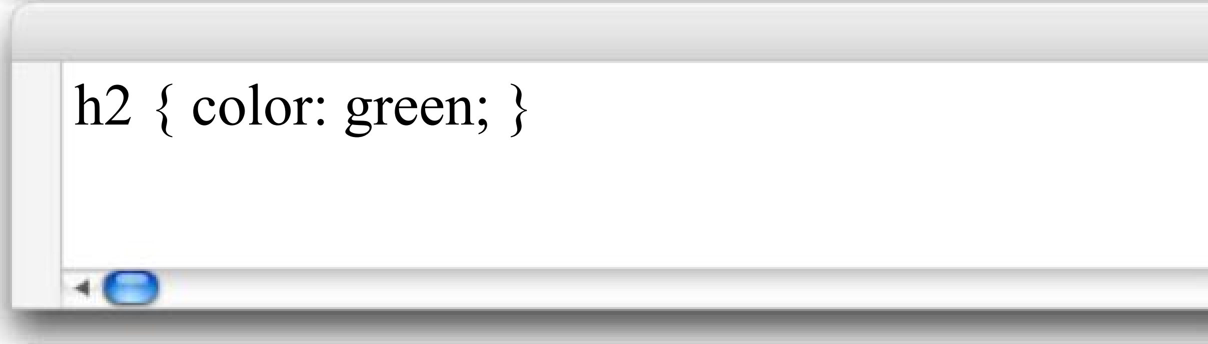
Part 1: Which one **wins?**

Browser style sheet



```
h2 { color: black; }
```

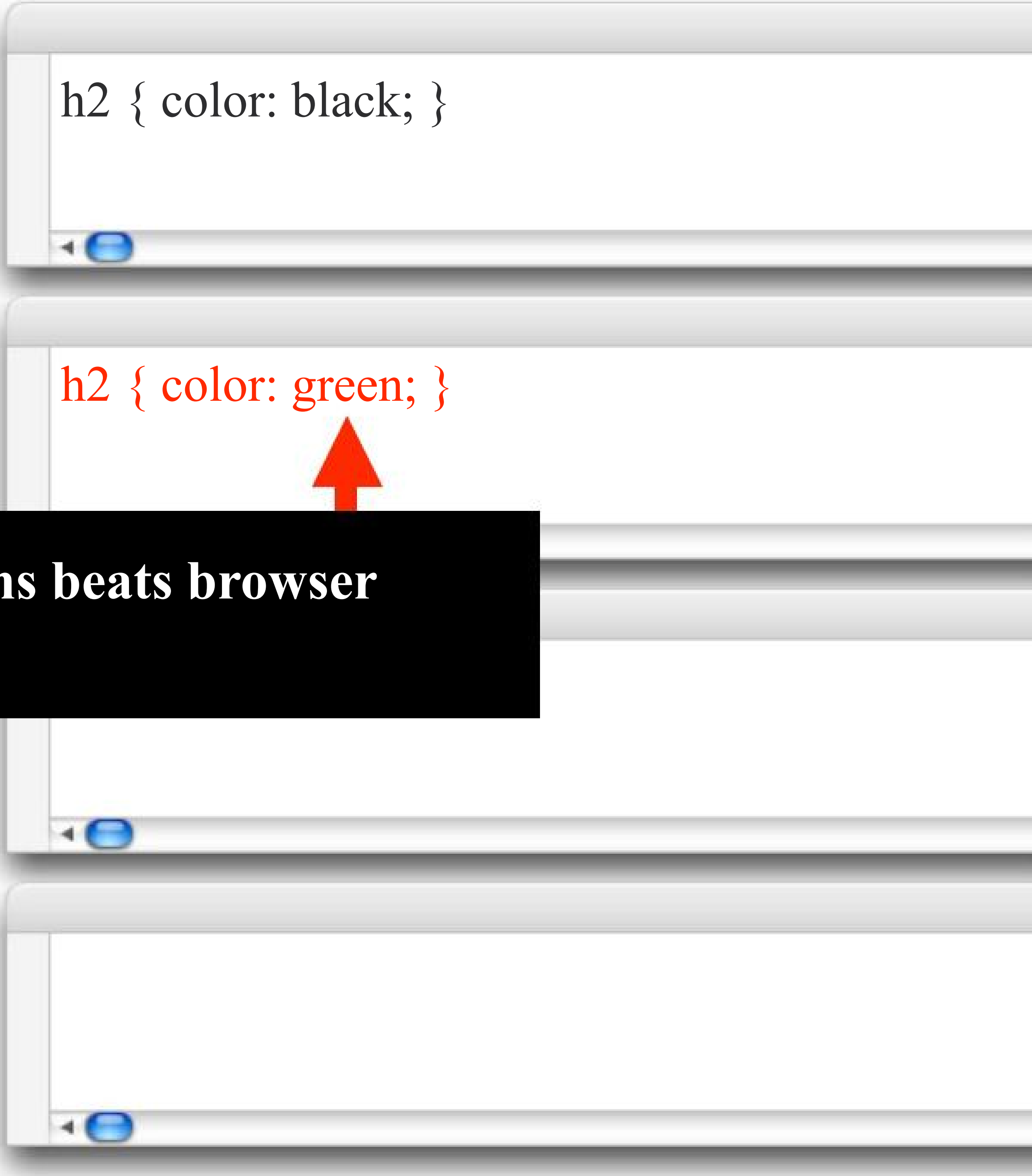
User style sheet



```
h2 { color: green; }
```

External style sheets and header styles (Author styles)

HTML document with inline styles (Author styles)



Browser style sheet

```
h2 { color: black; }
```

User style sheet

```
h2 { color: green; }
```

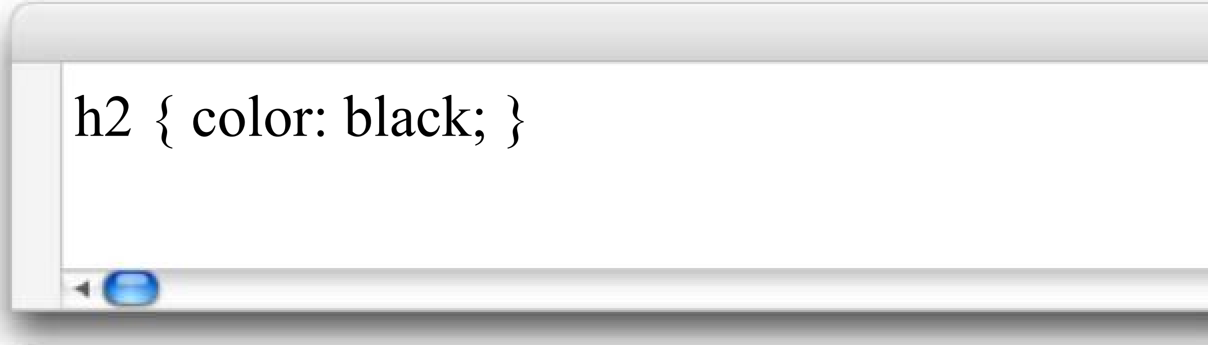
Normal user declarations beats browser

and header styles
(Author styles)

HTML document with
inline styles
(Author styles)

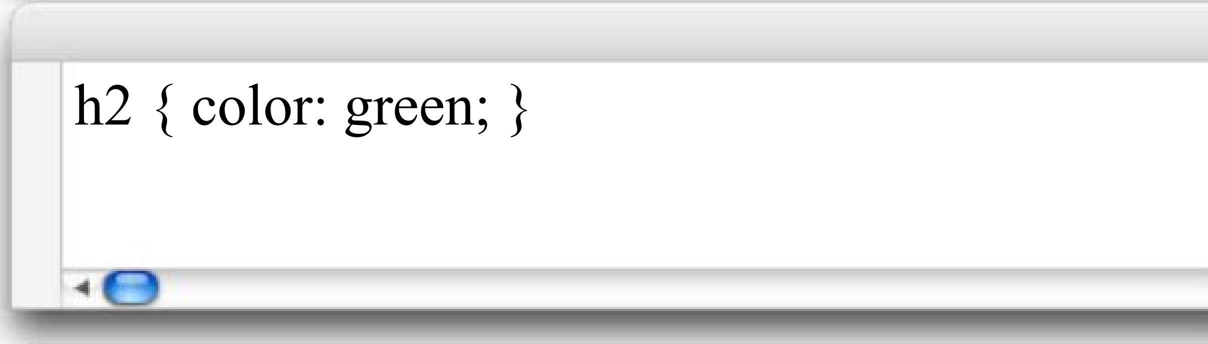
Part 2: Which one **wins?**

Browser style sheet



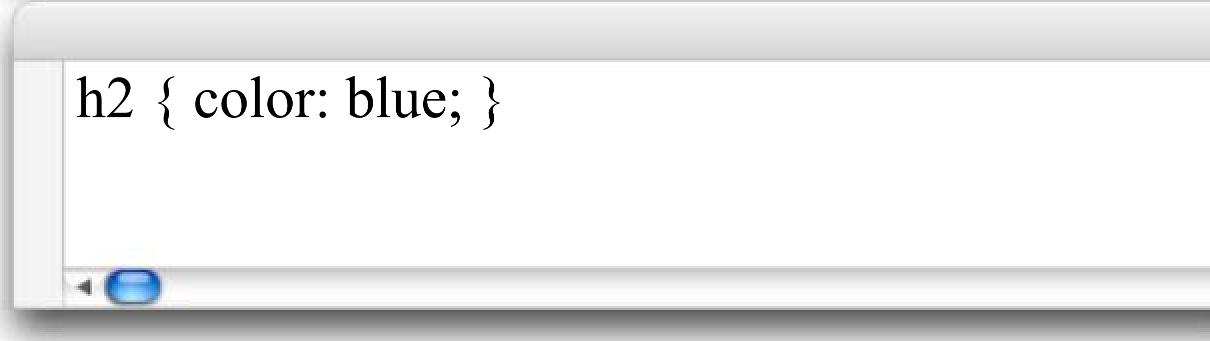
```
h2 { color: black; }
```

User style sheet



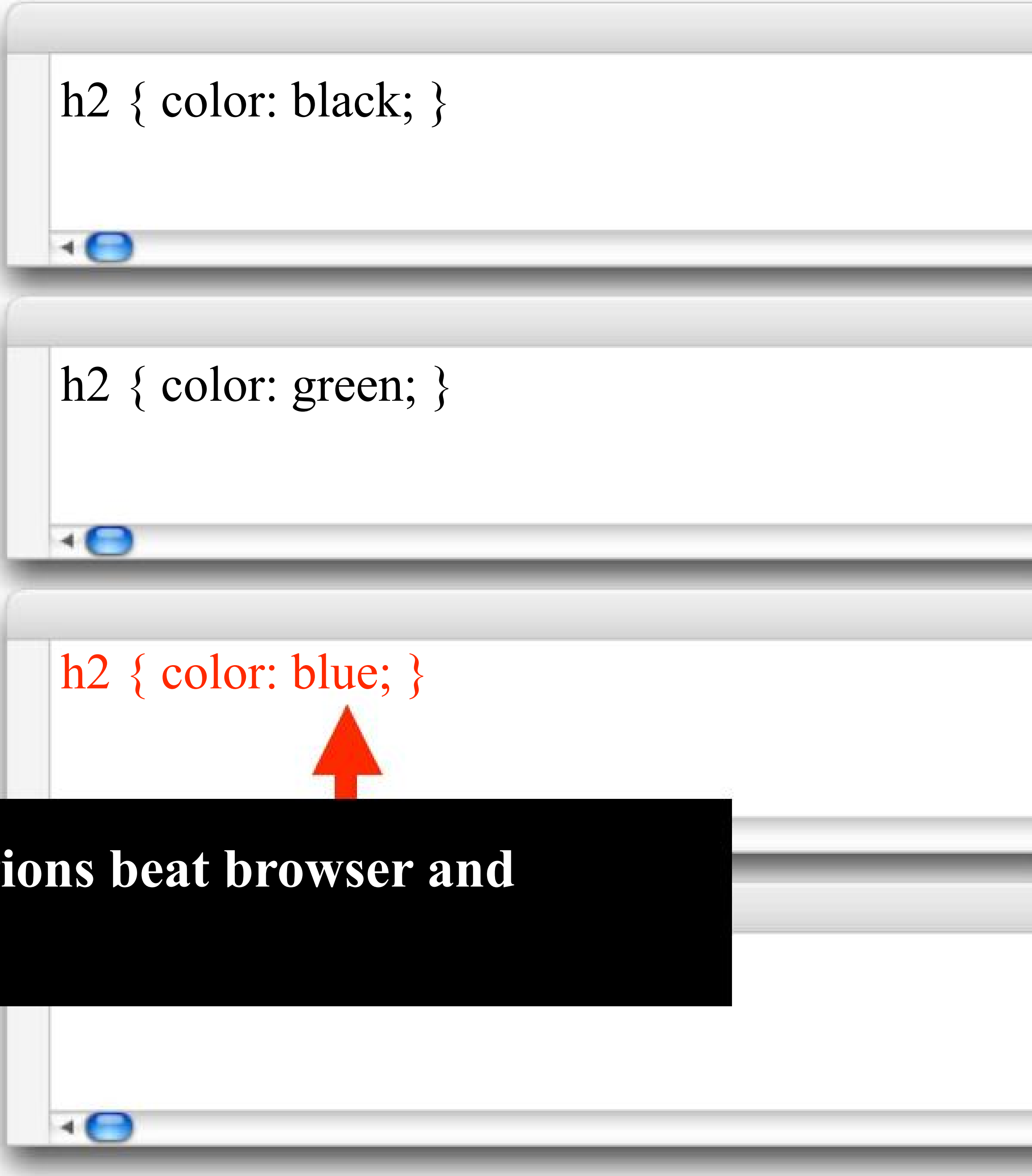
```
h2 { color: green; }
```

External style sheets and header styles (Author styles)



```
h2 { color: blue; }
```

HTML document with inline styles (Author styles)



Browser style sheet

```
h2 { color: black; }
```

User style sheet

```
h2 { color: green; }
```

External style sheets and header styles (Author styles)

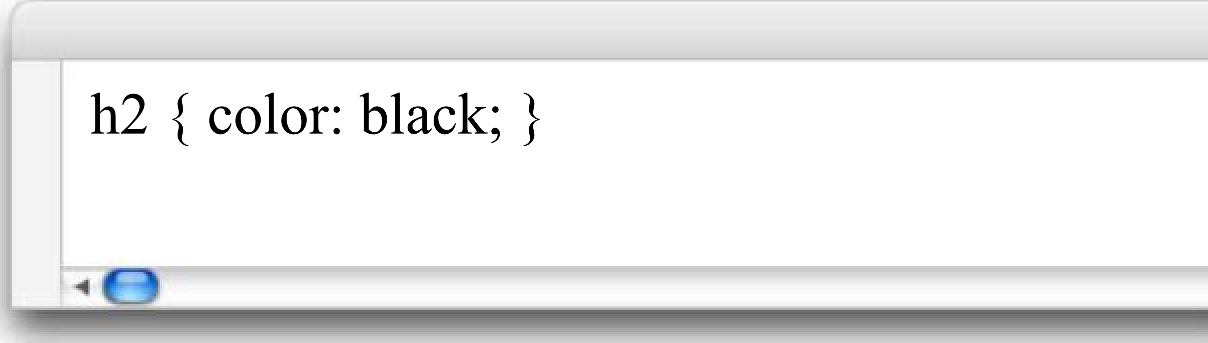
```
h2 { color: blue; }
```

Normal author declarations beat browser and

Inline styles (Author styles)

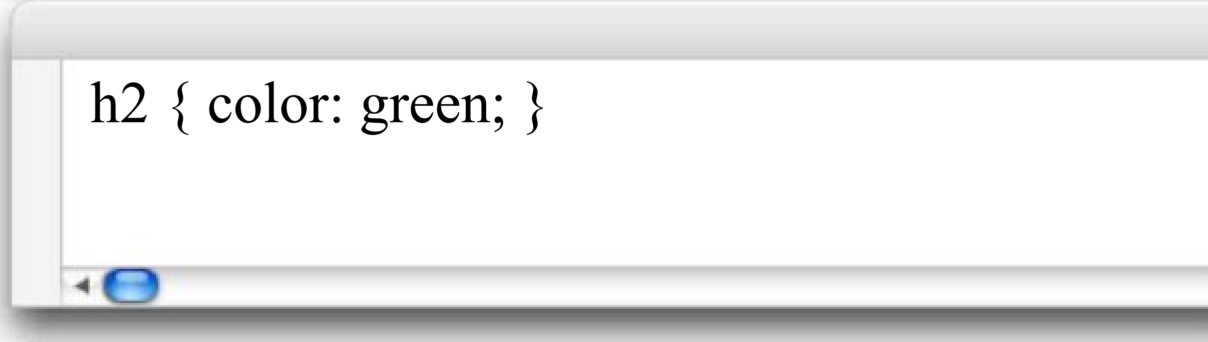
Part 3: Which one **wins?**

Browser style sheet



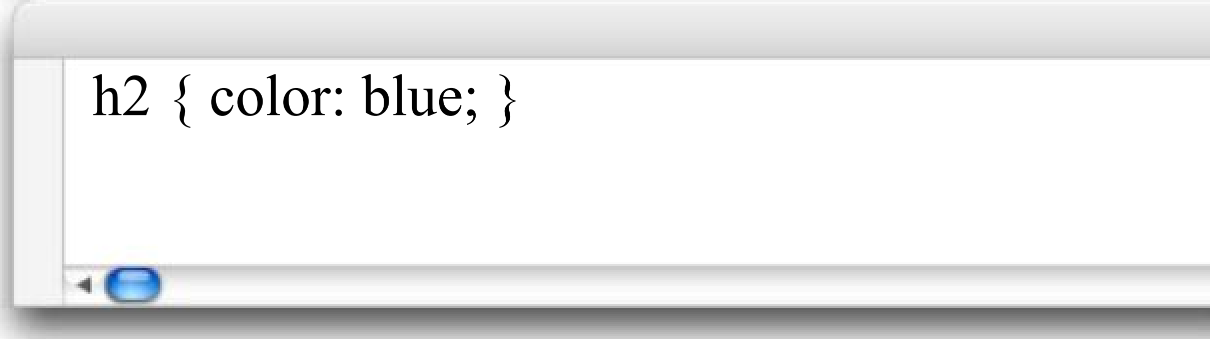
```
h2 { color: black; }
```

User style sheet



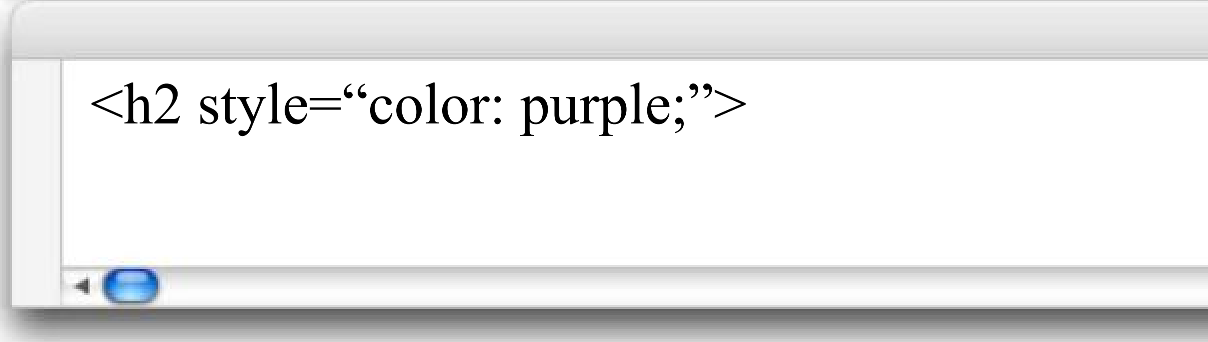
```
h2 { color: green; }
```

External style sheets and header styles (Author styles)

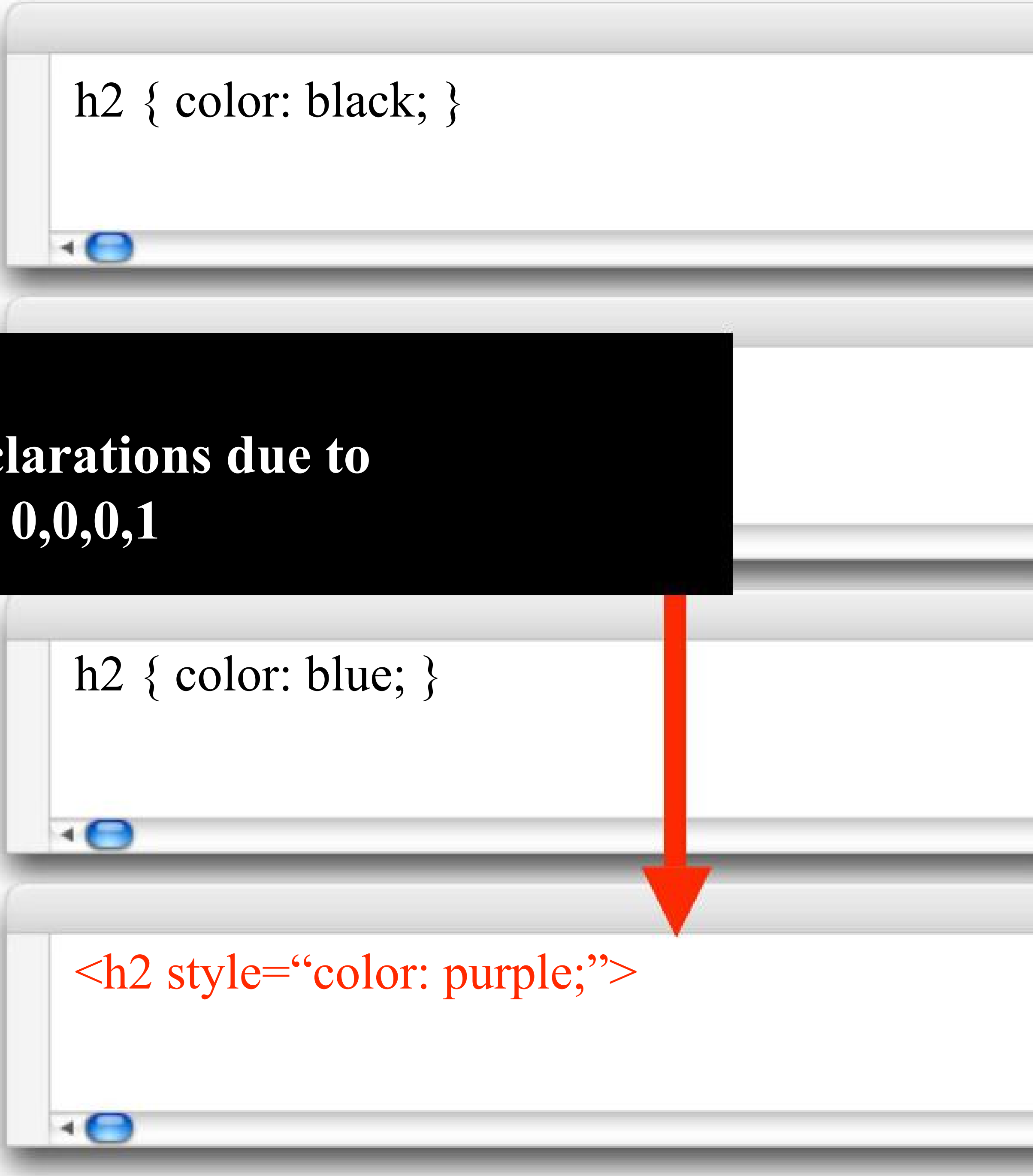


```
h2 { color: blue; }
```

HTML document with inline styles (Author styles)



```
<h2 style=“color: purple;”>
```



Browser style sheet

```
h2 { color: black; }
```

external and header declarations due to specificity: 1,0,0,0 beats 0,0,0,1

External style sheets and header styles (Author styles)

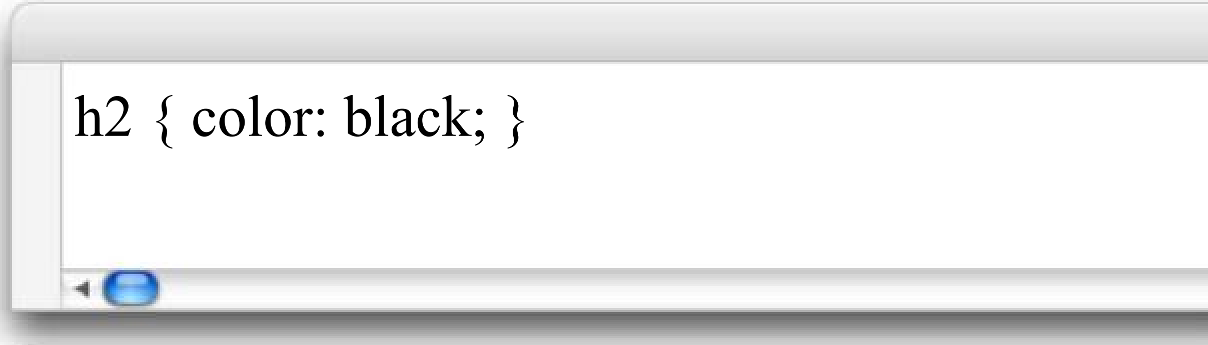
```
h2 { color: blue; }
```

HTML document with inline styles (Author styles)

```
<h2 style="color: purple;">
```

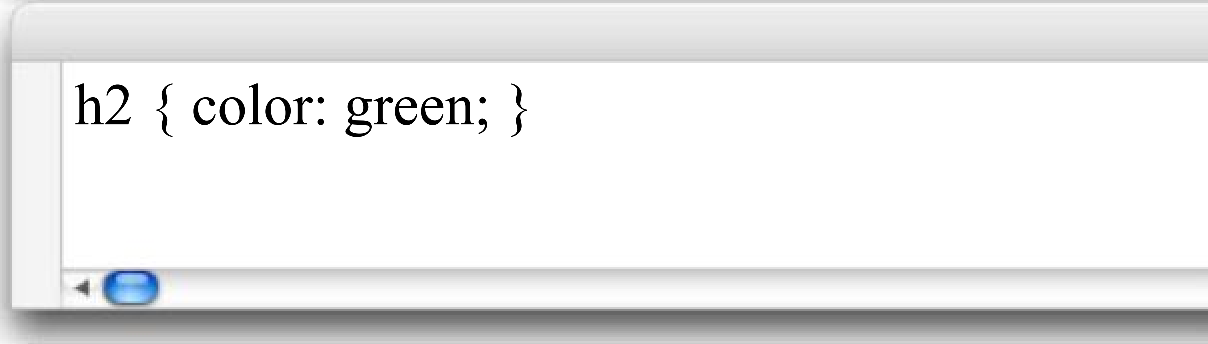
Part 4: Which one **wins?**

Browser style sheet



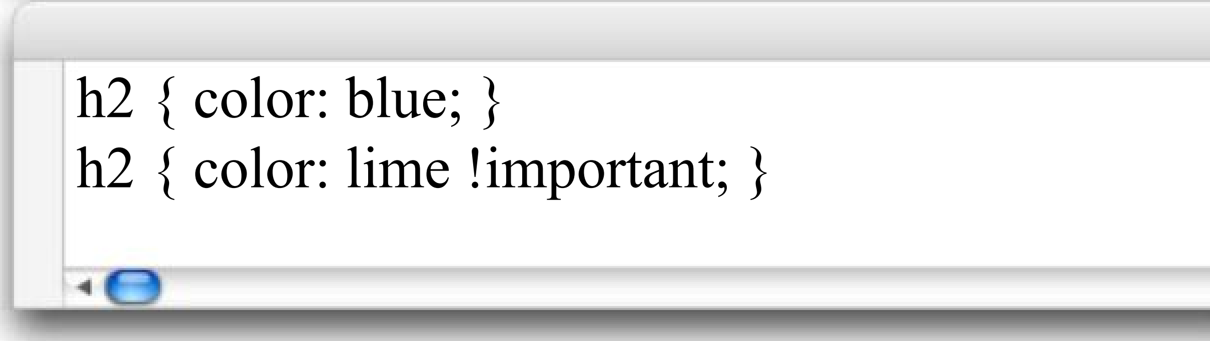
```
h2 { color: black; }
```

User style sheet



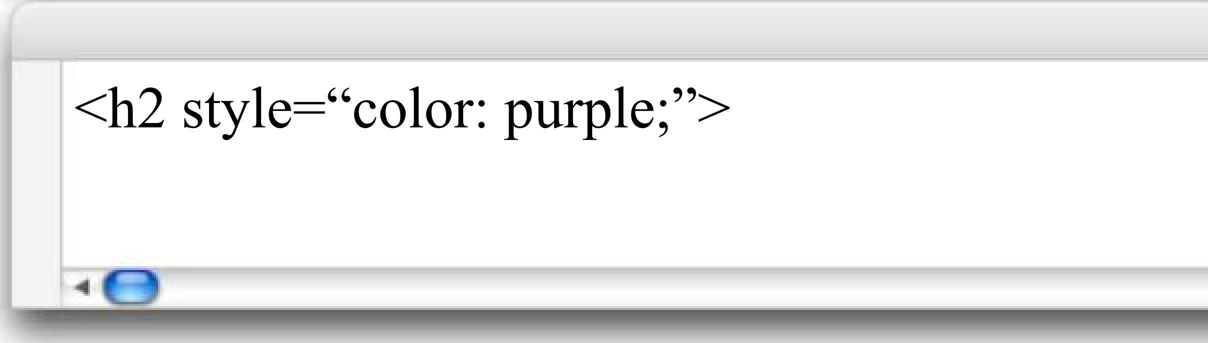
```
h2 { color: green; }
```

External style sheets and header styles (Author styles)

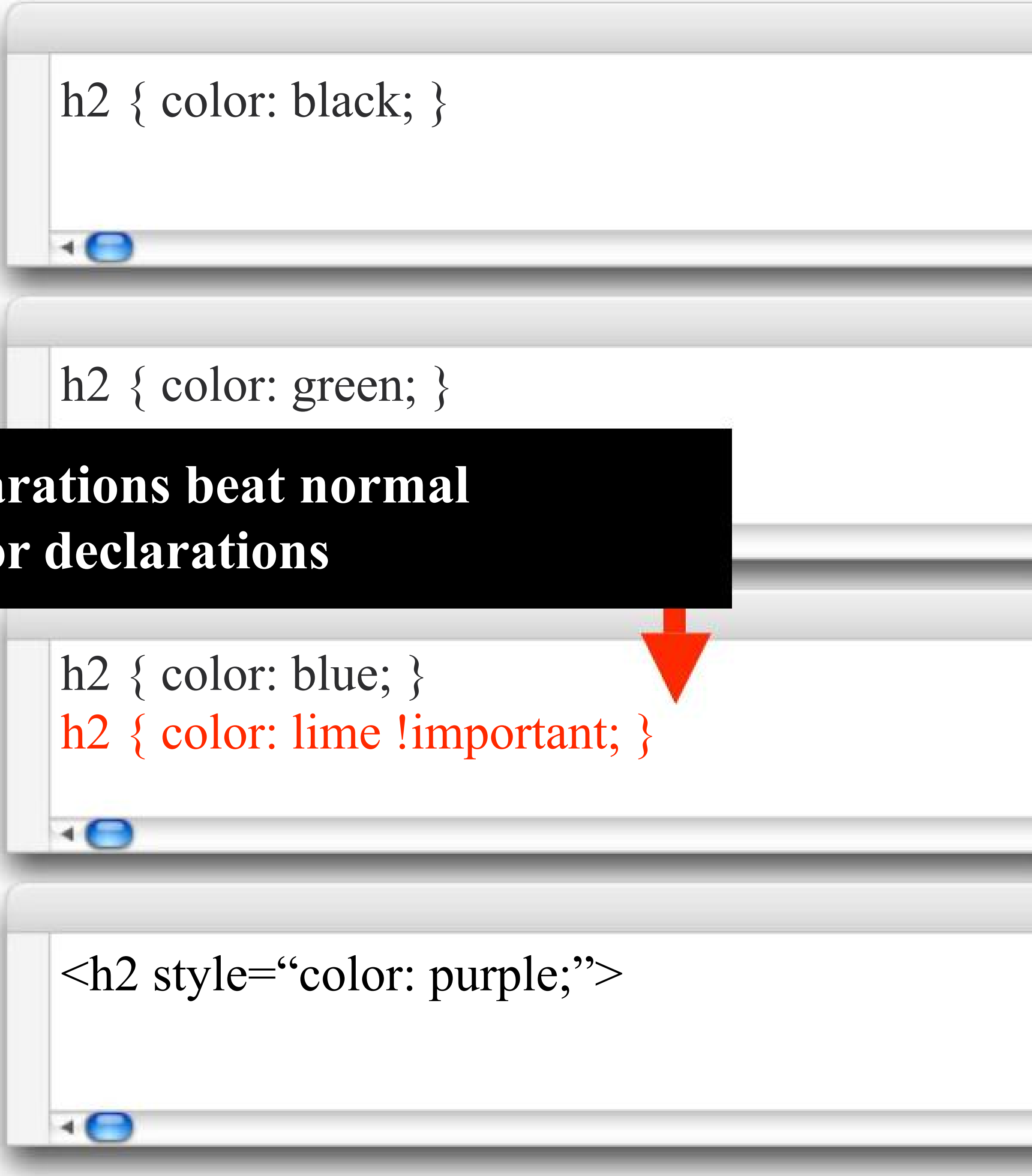


```
h2 { color: blue; }
h2 { color: lime !important; }
```

HTML document with inline styles (Author styles)



```
<h2 style="color: purple;">
```



Browser style sheet

```
h2 { color: black; }
```

User style sheet

```
h2 { color: green; }
```

!important author declarations beat normal browser, user and author declarations

External style sheets and header styles (Author styles)

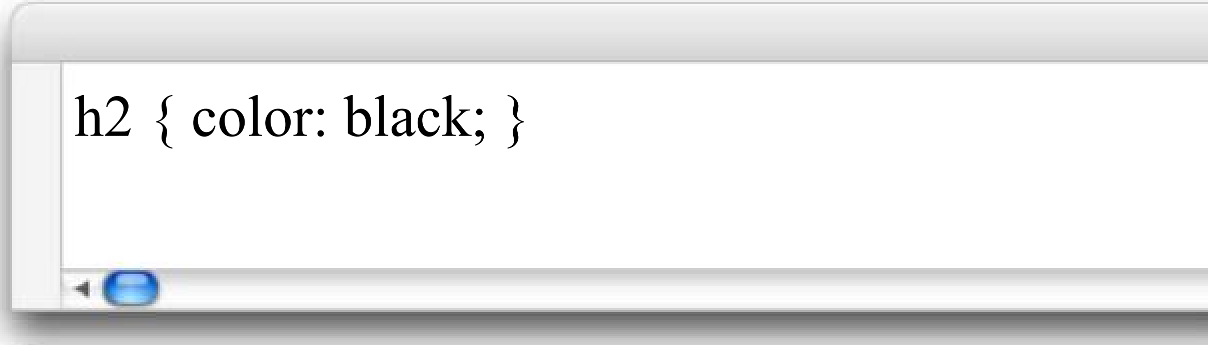
```
h2 { color: blue; }
h2 { color: lime !important; }
```

HTML document with inline styles (Author styles)

```
<h2 style="color: purple;">
```

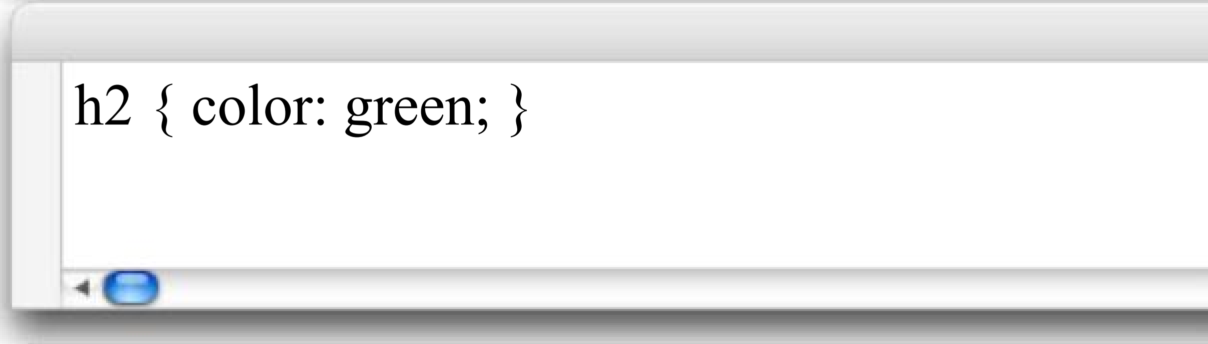
Part 5: Which one **wins?**

Browser style sheet



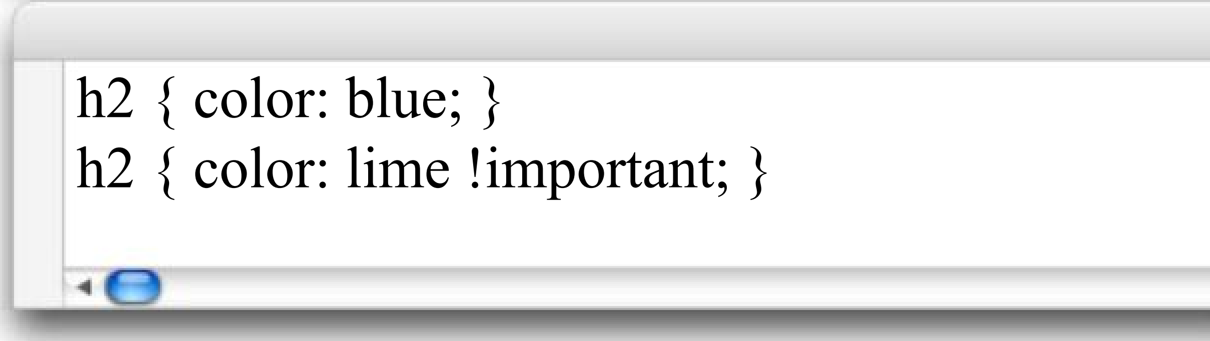
```
h2 { color: black; }
```

User style sheet



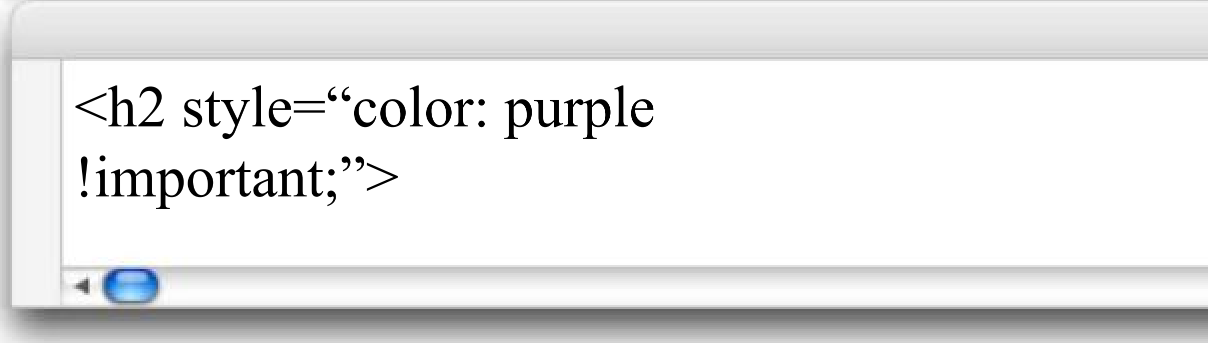
```
h2 { color: green; }
```

External style sheets and header styles (Author styles)



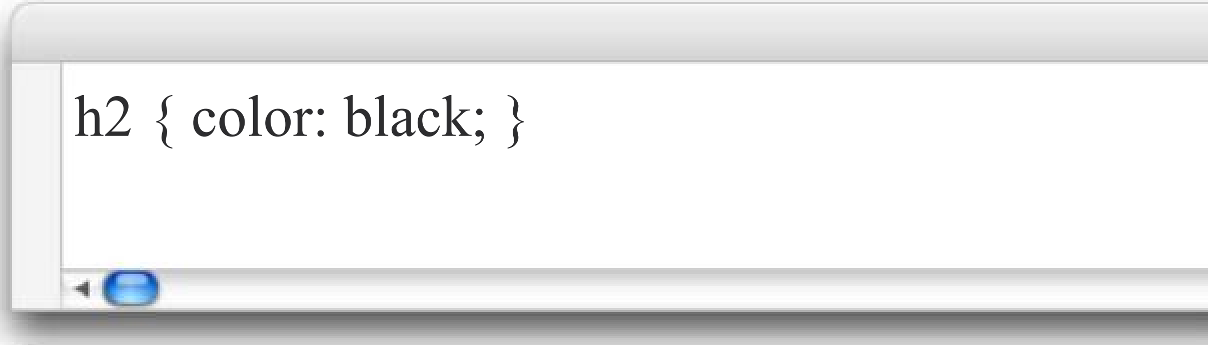
```
h2 { color: blue; }
h2 { color: lime !important; }
```

HTML document with inline styles (Author styles)



```
<h2 style="color: purple
!important;">
```

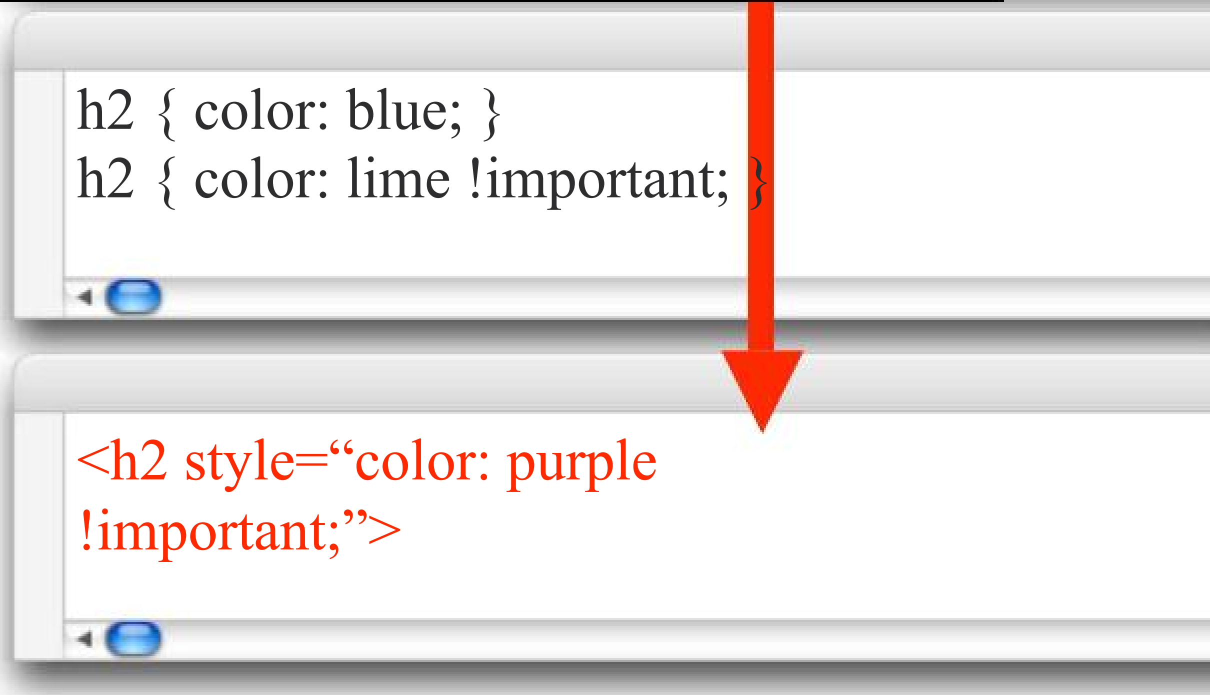
Browser style sheet



```
h2 { color: black; }
```

!important external author and header declarations due to specificity: 1,0,0,0 beats 0,0,0,1

External style sheets and header styles (Author styles)



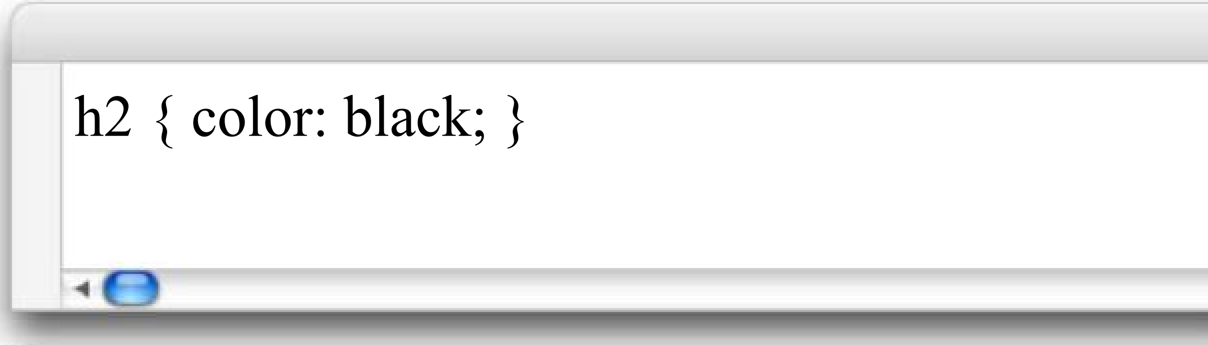
```
h2 { color: blue; }
h2 { color: lime !important; }
```

HTML document with inline styles (Author styles)

```
<h2 style="color: purple
!important;">
```

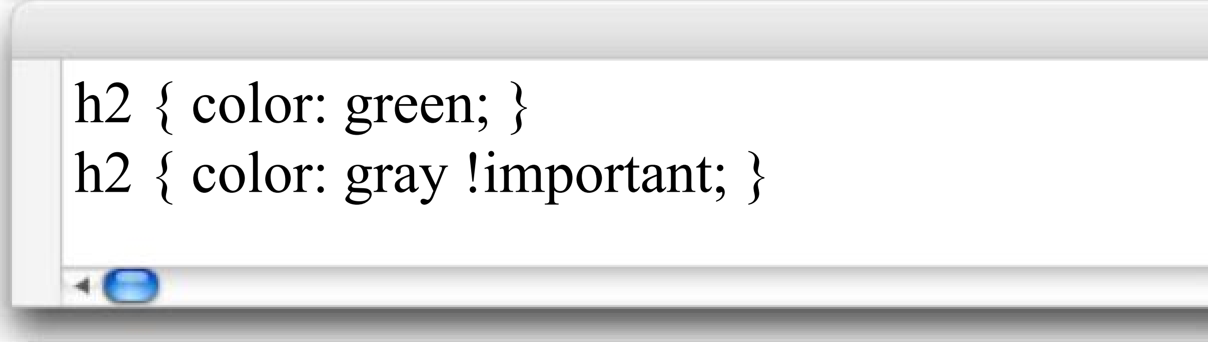
Part 6: Which one **wins?**

Browser style sheet



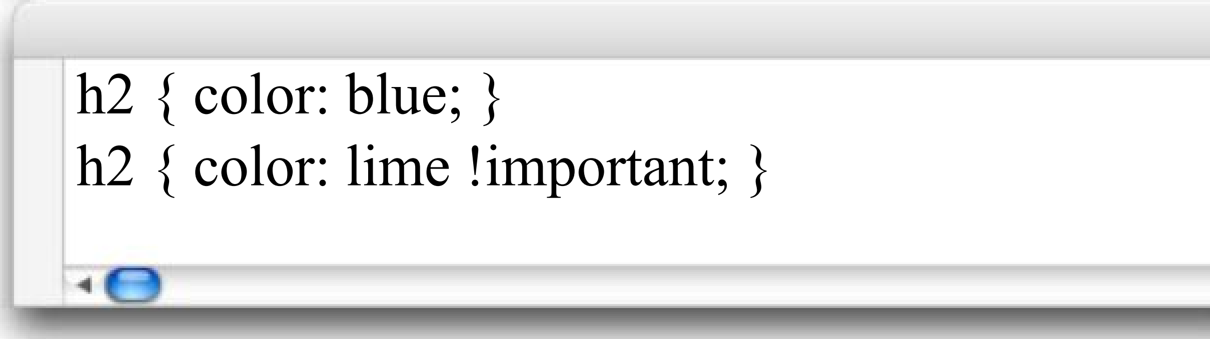
```
h2 { color: black; }
```

User style sheet



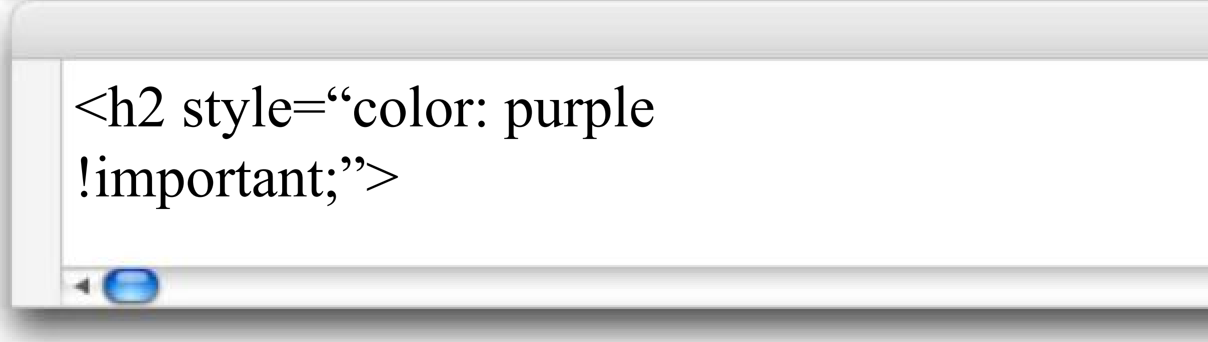
```
h2 { color: green; }
h2 { color: gray !important; }
```

**External style sheets
and header styles
(Author styles)**



```
h2 { color: blue; }
h2 { color: lime !important; }
```

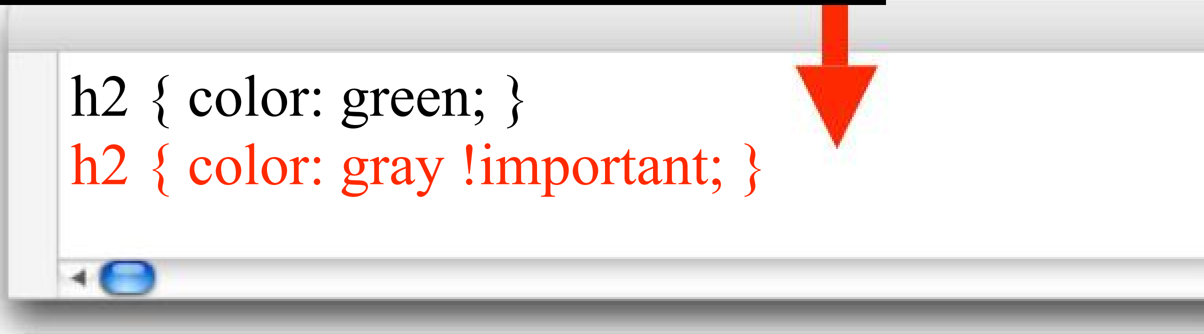
**HTML document with
inline styles
(Author styles)**



```
<h2 style=“color: purple
!important;”>
```

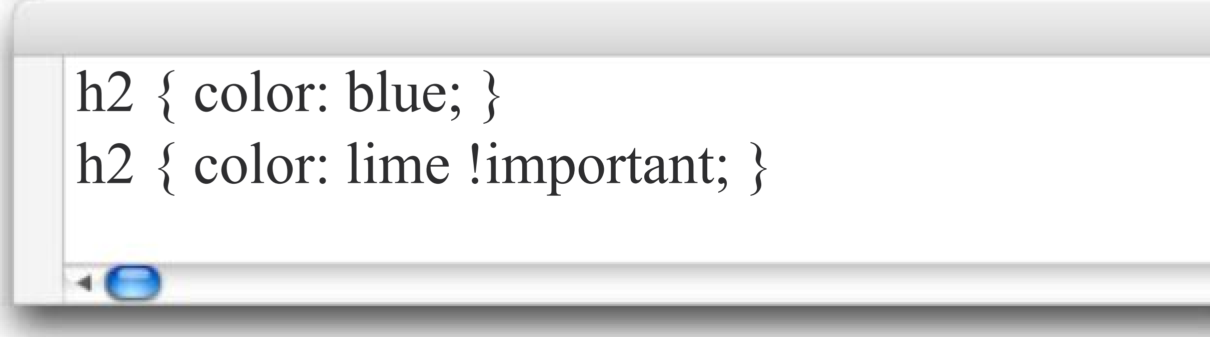
author declarations (regardless of whether they are external, header or inline)

User style sheet



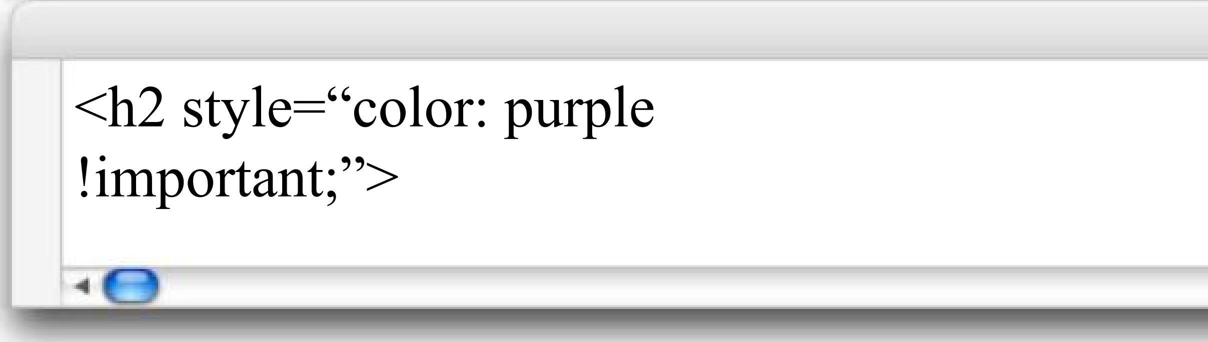
```
h2 { color: green; }
h2 { color: gray !important; }
```

External style sheets and header styles (Author styles)



```
h2 { color: blue; }
h2 { color: lime !important; }
```

HTML document with inline styles (Author styles)



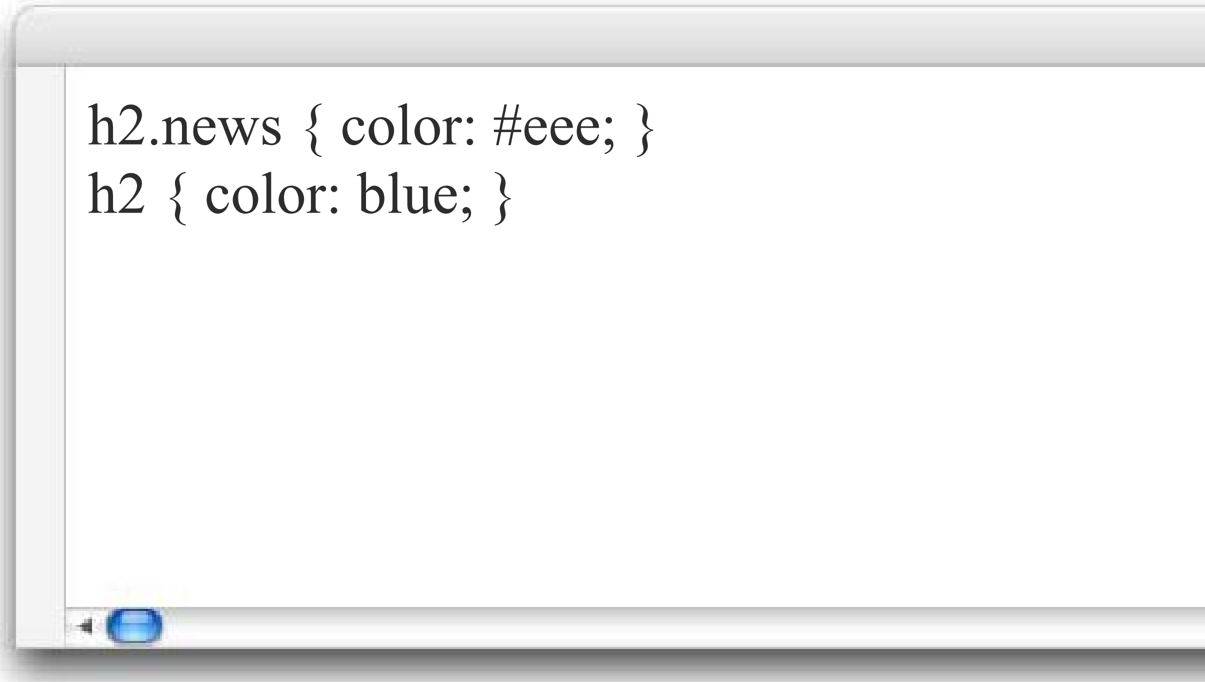
```
<h2 style=“color: purple
!important;”>
```

Exercise 2

author external, header and inline CSS

Part 1: Which one **wins?**

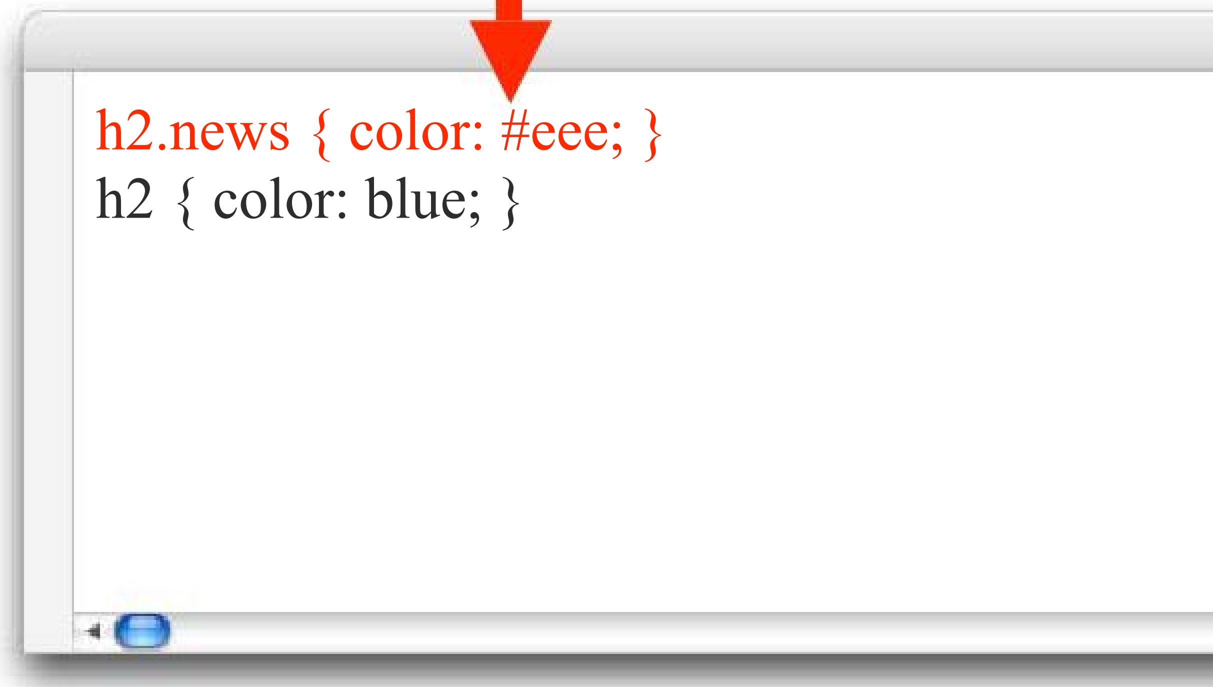
External style sheets
and header styles
(Author styles)



```
h2.news { color: #eee; }
h2 { color: blue; }
```

The highlighted declaration wins due to specificity - 0,0,1,1 beats 0,0,0,1

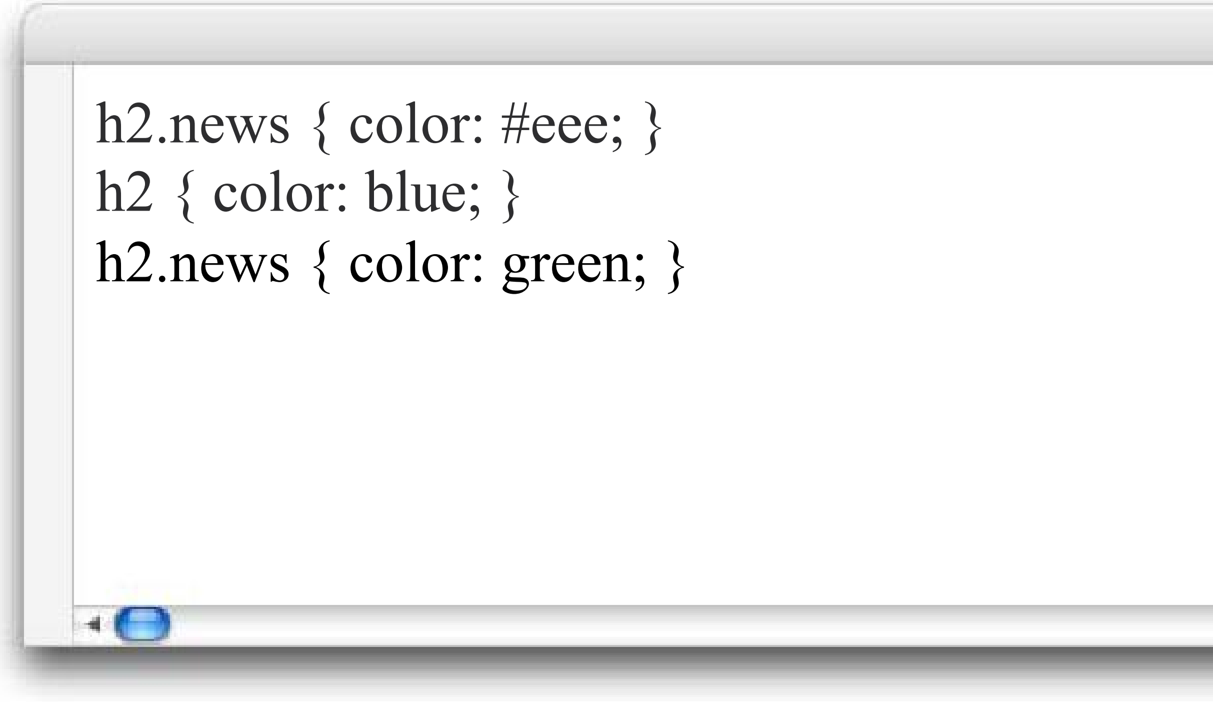
External style sheets
and header styles
(Author styles)



```
h2.news { color: #eee; }
h2 { color: blue; }
```

Part 2: Which one **wins?**

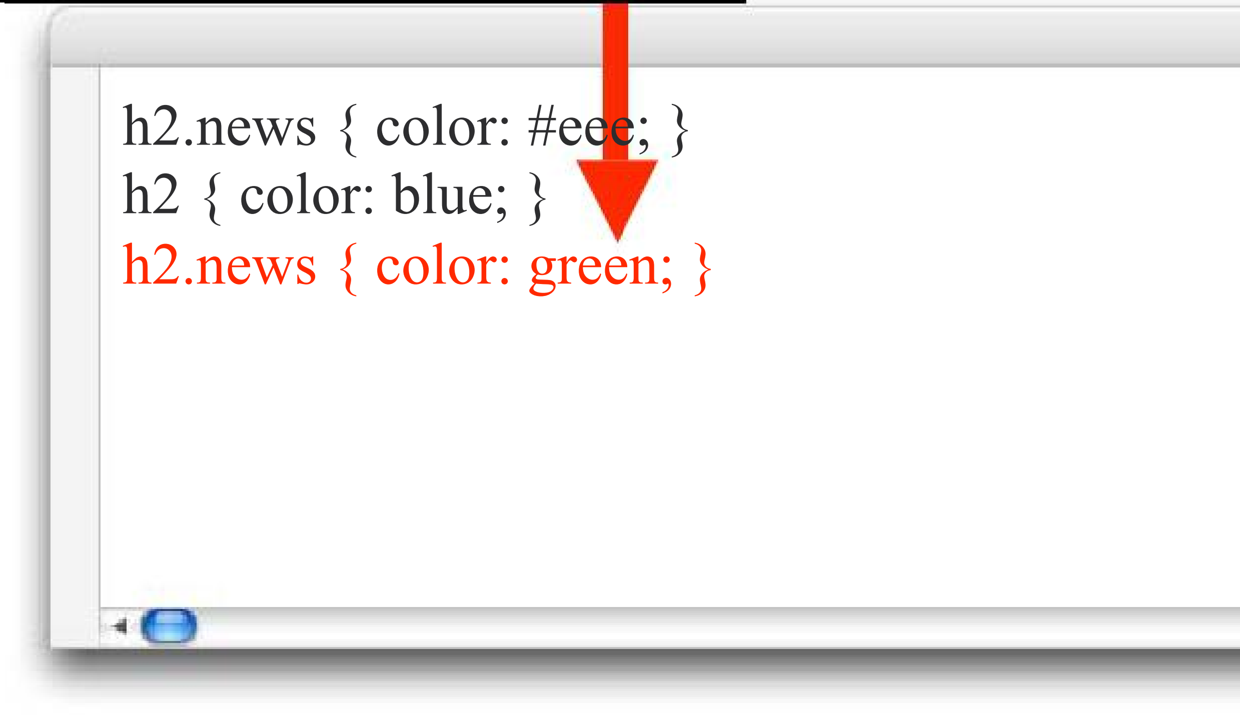
External style sheets and header styles (Author styles)



```
h2.news { color: #eee; }
h2 { color: blue; }
h2.news { color: green; }
```

The highlighted declaration has the same specificity as the first declaration (0,0,1,1). However, as it is written later, it wins!

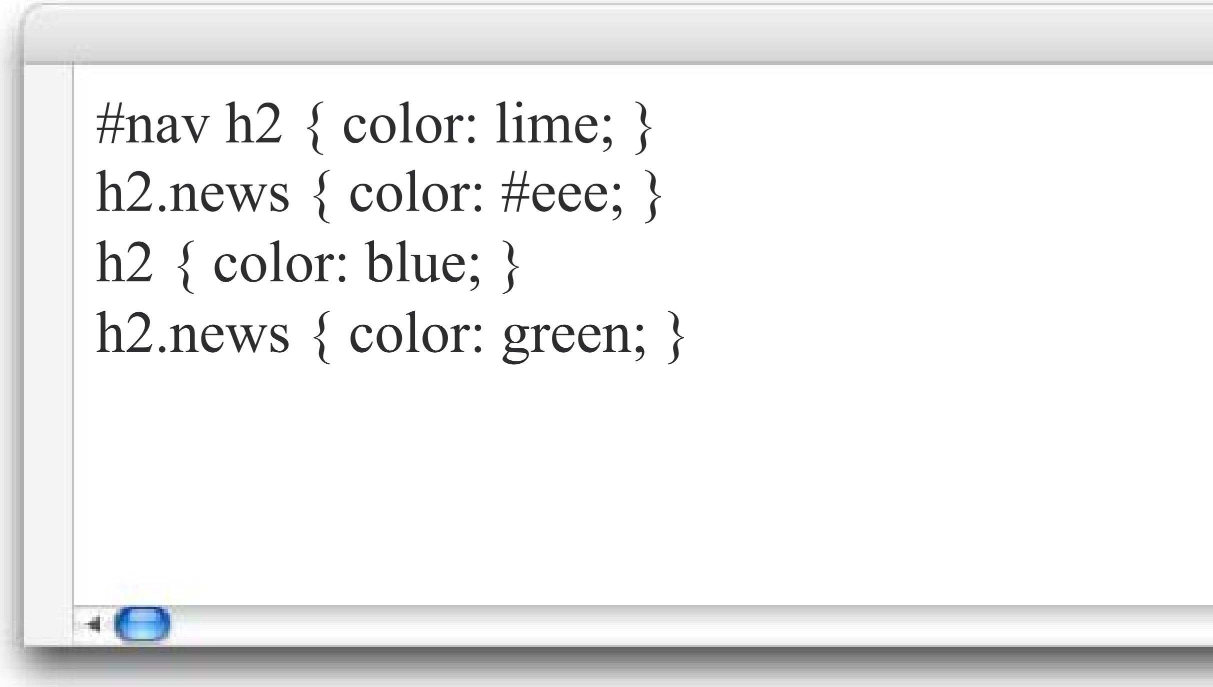
External style sheets and header styles (Author styles)



```
h2.news { color: #eee; }
h2 { color: blue; }
h2.news { color: green; }
```

Part 3: Which one **wins?**

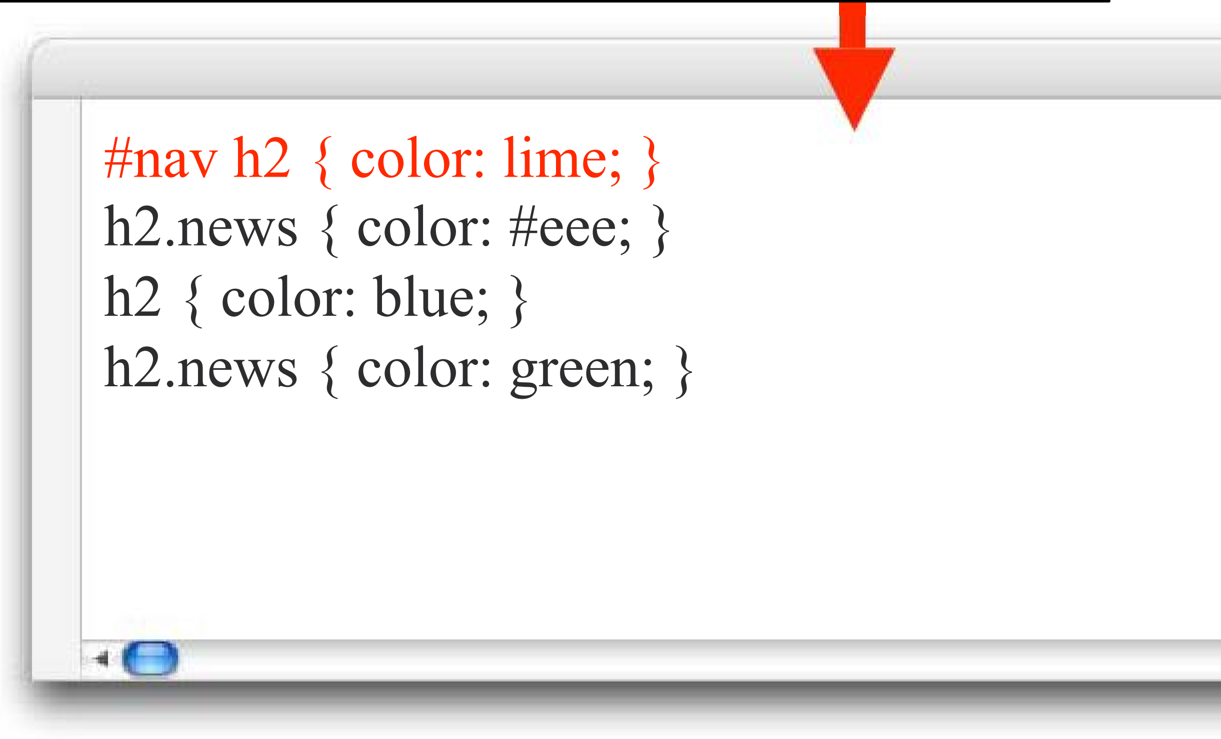
External style sheets and header styles (Author styles)



```
#nav h2 { color: lime; }
h2.news { color: #eee; }
h2 { color: blue; }
h2.news { color: green; }
```

The highlighted selector wins due to specificity - 0,1,0,1 beats 0,0,1,1 and 0,0,0,1

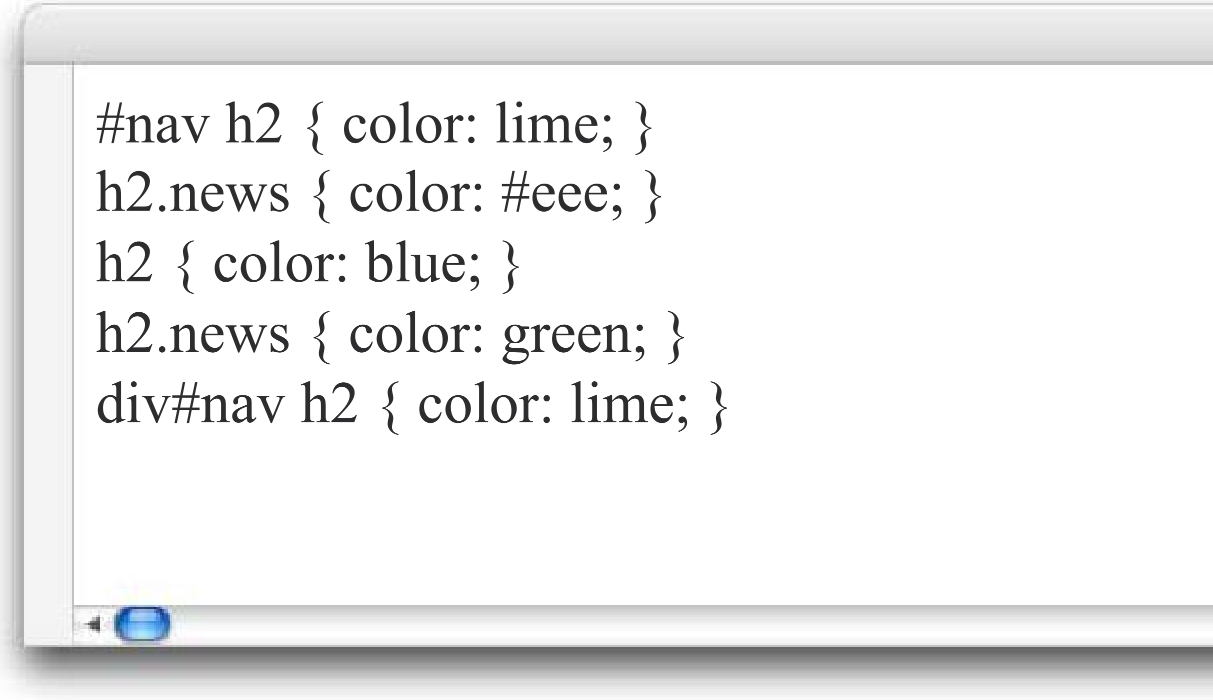
External style sheets and header styles (Author styles)



```
#nav h2 { color: lime; }
h2.news { color: #eee; }
h2 { color: blue; }
h2.news { color: green; }
```

Part 4: Which one **wins?**

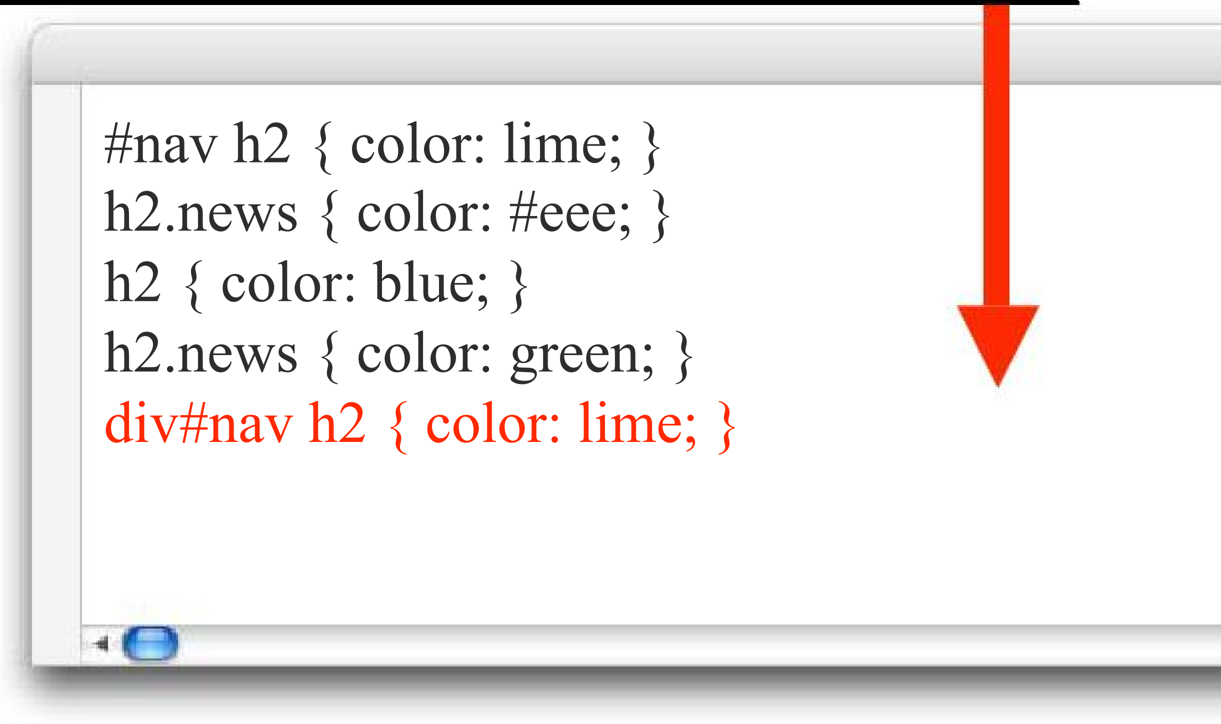
External style sheets and header styles (Author styles)



```
#nav h2 { color: lime; }
h2.news { color: #eee; }
h2 { color: blue; }
h2.news { color: green; }
div#nav h2 { color: lime; }
```

The highlighted selector wins due to specificity - 0,1,0,2 beats 0,1,0,1 and 0,0,1,1 and 0,0,0,1

External style sheets
and header styles
(Author styles)



```
#nav h2 { color: lime; }
h2.news { color: #eee; }
h2 { color: blue; }
h2.news { color: green; }
div#nav h2 { color: lime; }
```

We’re done!