结构和层叠

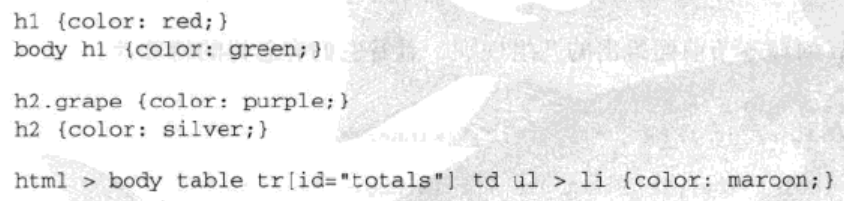
1. 继承(Inheritance)

从一个元素向其后代元素传递属性值。

确定应当向一个元素应用哪些值时，用户代理不仅要考虑继承，还有考虑声明的特殊性，另外需要考虑声明本身的来源。这个过程就叫做层叠(cascade)

**特殊性-继承-层叠**

1. 特殊性



对于每个选择器的特殊性(specificity)。对于每个规则，用户代理会计算选择器的特殊性，并将这个特殊性附加到规则中的各个声明。如果一个元素有两个或多个冲突的属性声明，那么有最高特殊性的声明就会胜出。

* 对于选择器中给定的各个ID属性值，加0,1,0,0.
* 对于选择器中给定的各个类属性值、属性选择或伪类，加0，0，1，0。
* 对于选择器中给定的各个元素和伪元素，加0，0，0，1。
* **结合符【空格 > +】**和通配选择器对特殊性没有任何贡献（即没有特殊性）。也就是其特殊性为0，0，0，0

例子:



特殊性值：

0，0，0，1

0，0，0，2

0，0，1，0

0，0，1，0

0，0，2，2，

0，1，0，0

0，1，1，1

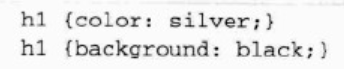
1. 声明和特殊性

一旦确定了一个选择器的特殊性，这个值将授予其所有相关声明。

**由于特殊性的缘故，用户代理必须相应地处理这个规则，将其‘解组’为单独的规则。**



会被解组为：



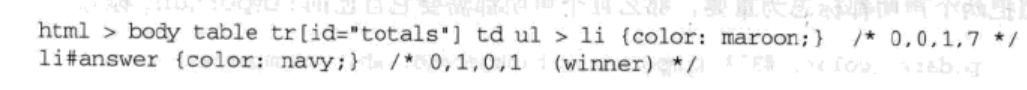
1. 通配选择器特殊性

**通配选择器的特殊性值为：0，0，0，0**



**结合符根本根本没有特殊性，甚至连0特殊性都没有。**

1. ID 和属性选择的特殊性



1. 内联样式特殊性

第一个0是为内联样式声明保留的，它比所有其他声明的特殊性都高。

**每一个内敛的声明的特殊性都是1，0，0，0.**

1. 重要性

**允许在这些声明的结束分号之前插入!important来标志。**

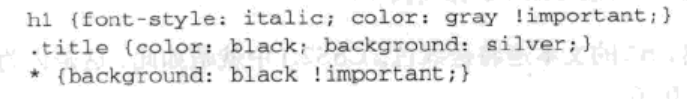


**必须将!important标志放在声明的最后。**

备注：

标志为!important的声明并没有特殊的特殊性值，不过要与非重要声明分开考虑。

实际上，所有!important声明会分组在一起，重要声明的特殊性冲突会在重要声明内部解决，而不会与非重要声明相混。类似的，我们认为非重要的声明也会归为一组，使用特殊来在内部解决。如果一个重要声明和一个非重要声明冲突，胜出的总是重要声明。



<h1 class="title" style="background: blue;">NightWing</h1>

!important 非重要

h1{font-style:italic}

.title{color:black;background:silver;}

h1{color:gray }  
\*{background:back}

引用的样式：

h1{color:gray; font-style:italic; background:back;}

即使使用内联元素，!important也会其作用。