

CSS

属性继承机制



翻译: 窦连军 @八月虎baidu
原稿: Russ Weakley

从文档树开始...

document tree

在了解继承机制之前，我们需要了解
什么是...

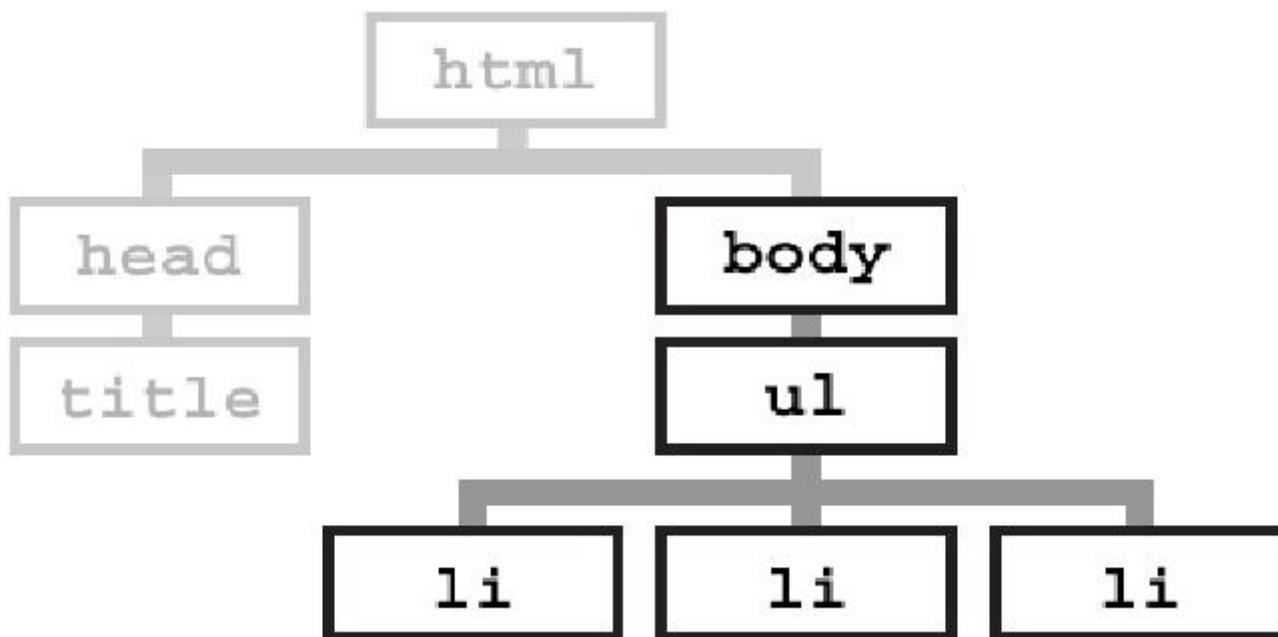
document tree.



所有 HTML 文档都是“树 (**tree**)”。



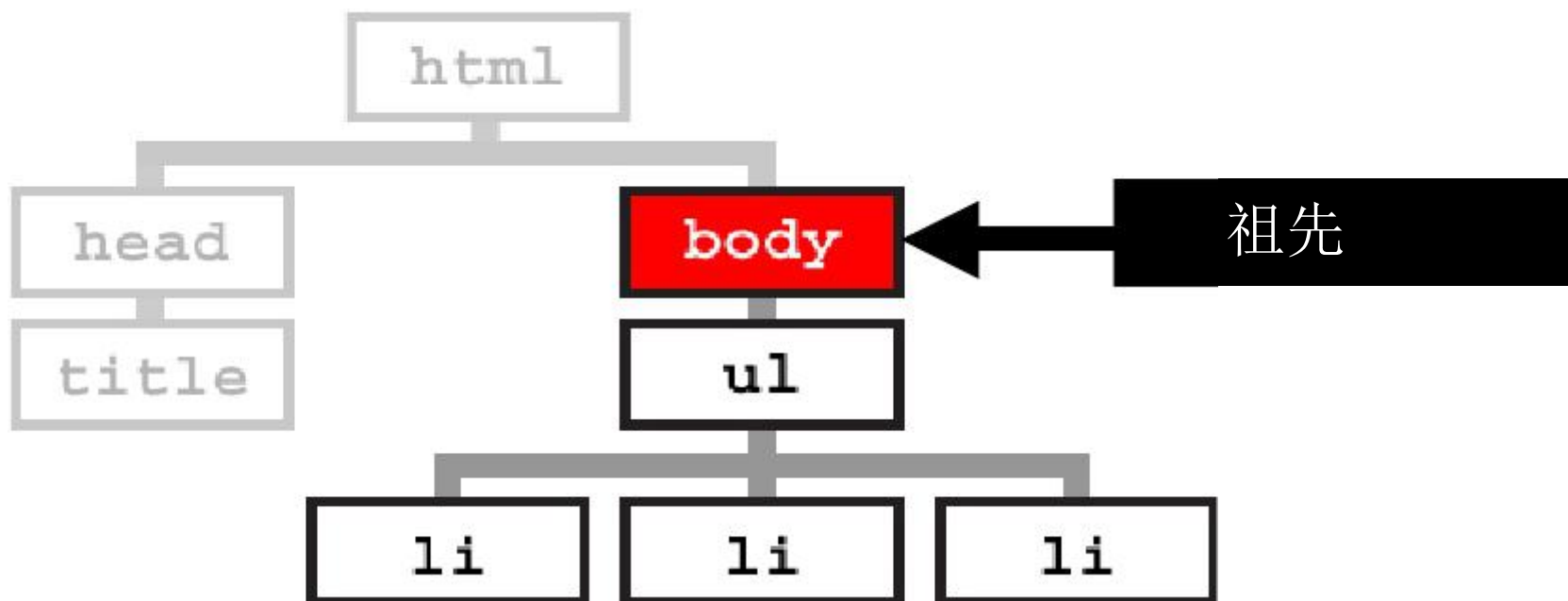
文档树由 **HTML** 元素构成



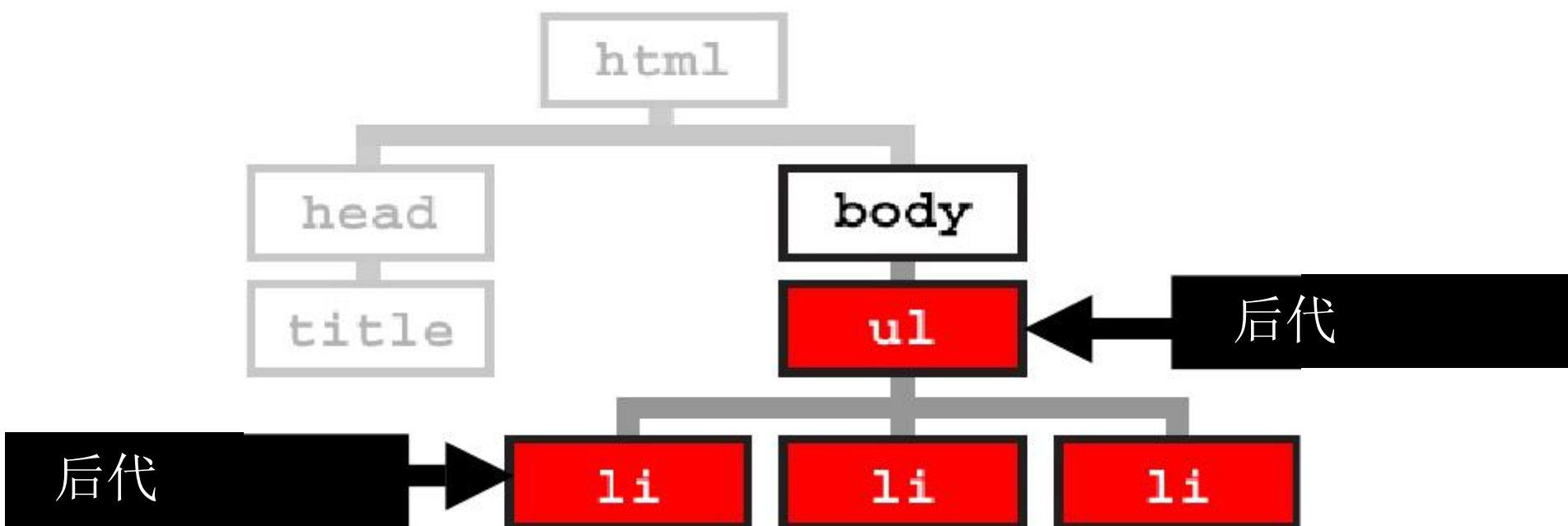
文档树好比你的家谱



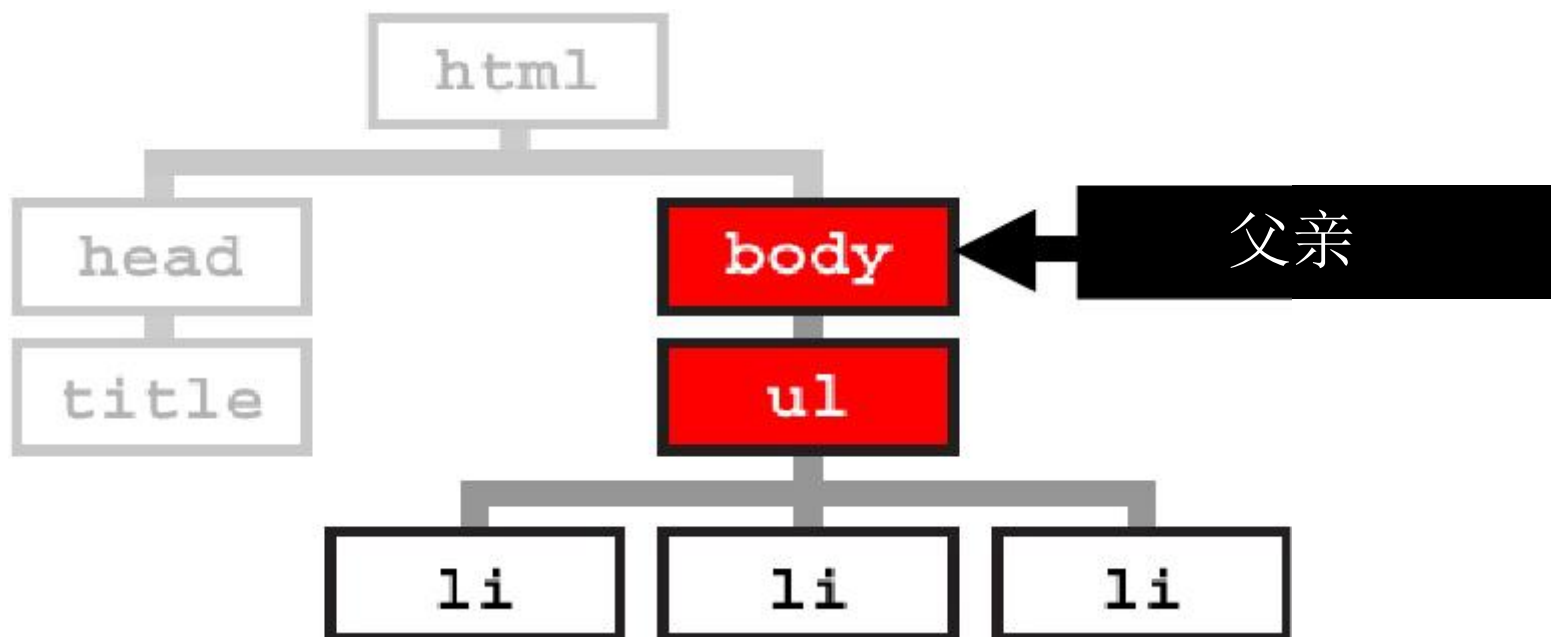
一个 **祖先** 指所有能够连接到的
处于文档树顶端的元素。



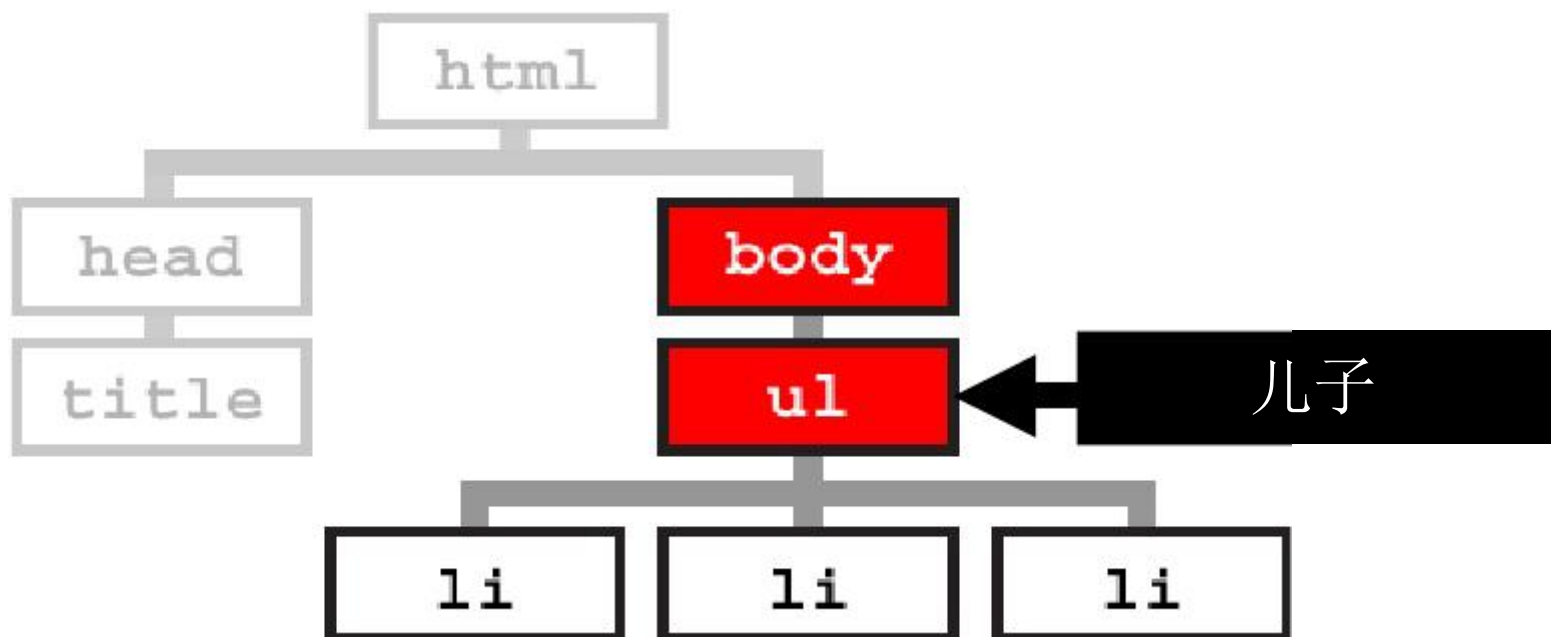
后代 指所有连接到的处于文档树底部的元素。



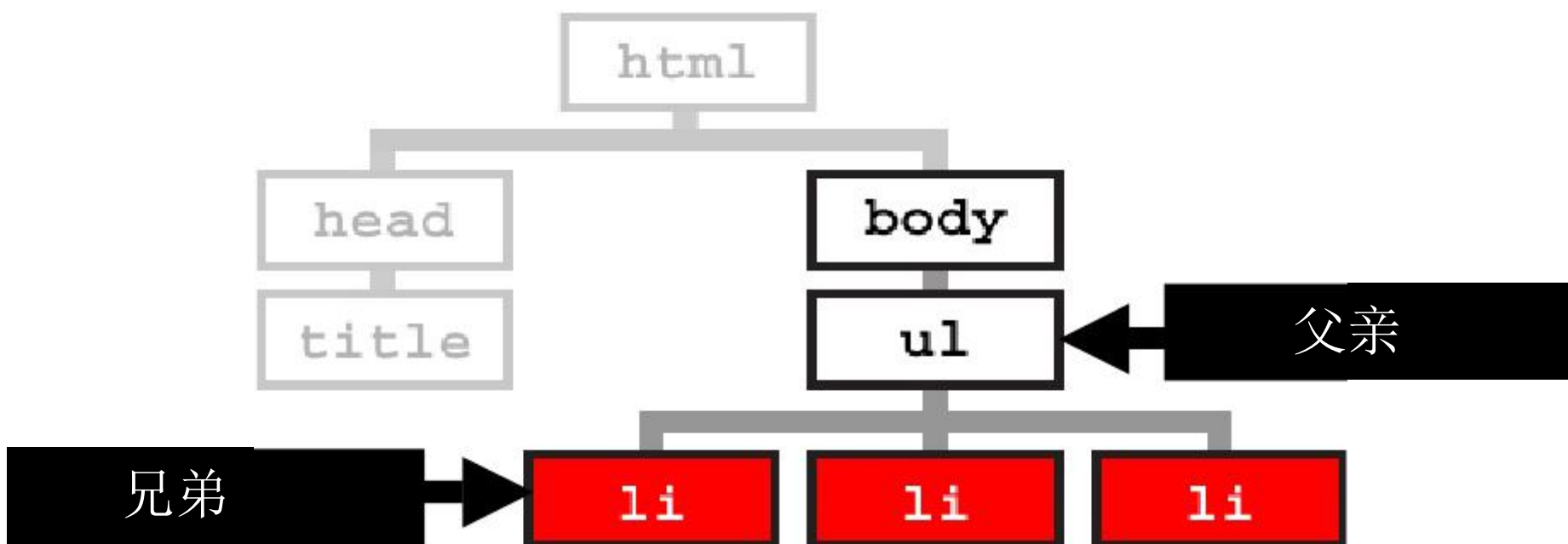
父亲 指连接到的在文档树中的直接上级。



儿子 是指连接到的在文档树上的直接下级。



兄弟 指与自己共同拥有同一父亲的其他元素。



接下来,讲一些
CSS知识

在了解CSS属性继承之前，
我们也需要懂一些基本的**CSS规则**



CSS规则告诉浏览器
如何去渲染 指定的HTML元素到
页面屏幕上。



CSS 规则由**5大部分**组成



选择器（selector） “选择” HTML 页面上那些满足指定条件的元素。

A diagram showing a code editor window. Inside the editor, the text 'p { color: red; }' is displayed. The 'p' is highlighted with a yellow background. A red arrow points from a black box containing the Chinese characters '选择器' (Selector) to the 'p'.

p { color: red; }

选择器

样式声明块 (declaration block)
是指两个大括号中间的部分内容。



The diagram shows a CSS declaration block within a simulated browser window. The text 'p { color: red; }' is displayed. The selector 'p' is on the left. The opening curly brace '{' is on the left of the declaration. The declaration 'color: red;' is highlighted in yellow. The closing curly brace '}' is on the right of the declaration. A red arrow points from the label '样式声明块' to the yellow highlighted area.

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

样式声明块

样式声明 (declaration) 告诉浏览器如何渲染那些页面上被选中的元素。

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

样式声明

属性（**property**）是指将被渲染的元素样式的某个方面。

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



属性名

属性值（value） 指为样式属性所设置的具体样式。

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



属性值

现在， ...
什么是
属性继承？

所谓“属性继承”是指特定的CSS属性被**向下传递**到后代元素身上。



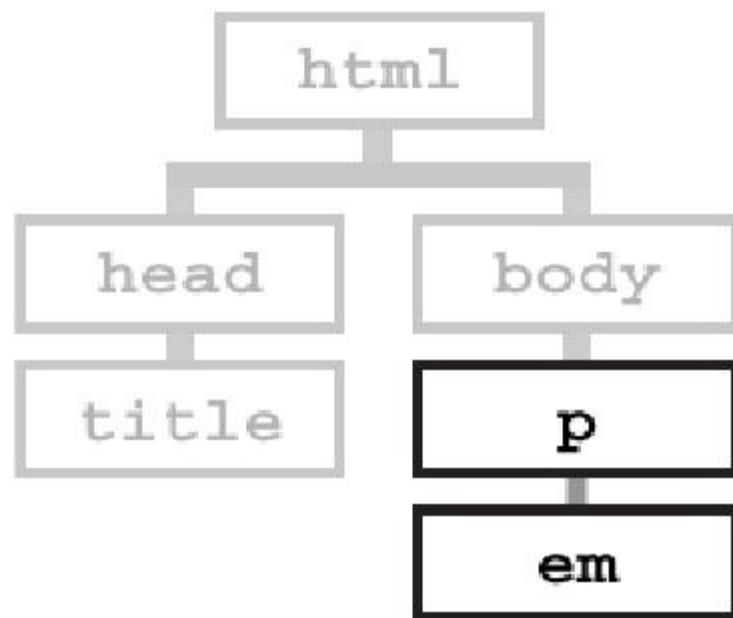
我们通过使用下面的 **HTML 代码** 来具体说明“属性继承”现象。

`<p>`

Lorem ``ipsum`` dolor
sit amet consectetur.

`</p>`

注意到 `` 元素 被包含在
`<p>` 元素之内。



我们接着继续使用该 CSS 代码。
我们注意到 `` 元素目前是
没被选中！

A screenshot of a code editor window with a light gray border and a white background. The window has a standard macOS-style title bar with a red close button, a yellow maximize button, and a blue window control button. The code inside the editor is a CSS rule: `p { color: red; }`. The text is in a monospaced font, and the word 'color' is highlighted in red.

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

但在浏览器中, `<p>` 和 ``
元素都变成 红色啦



为啥子 `` 元素变成 红色 的呢？
这个元素可是没有被CSS渲染过的呀！



因为这个 `` 元素从那个 `<p>` 元素
中 **继承** 了颜色属性。



为啥
“属性继承”
很有用？

“属性继承”机制的建造目的就是想给网页作者提供 **编写便利**。



否则，我们就不得不来为
所有的后代元素指定属性了

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

CSS文件将变得 **尺寸非常大**，
不仅下载缓慢，而且很难来编写和维护。



难到所有的属性
都会遵守继承
机制吗？

不是！并不是所有CSS属性都可以
被**继承**！



如果每个 CSS 属性都能自动继承，
那么这将给网页作者带来
巨大的麻烦！



作者将不得不 **关闭** 并不想要被继承的那些CSS属性。



例如，你这么想想
默认情况下，如果
border（边框）属性被继承了



我们设想在<p>元素上应用了
border 属性

A screenshot of a code editor window with a light gray border and a white background. The code is written in a monospaced font. The text 'p { border: 1px solid red; }' is displayed, with 'border: 1px solid red;' highlighted in red. The window has a standard macOS-style title bar with a red close button, a yellow maximize button, and a blue window control button on the right side.

```
p { border: 1px solid red; }
```

位于<p>之内的 元素也自动有了
一个 红色边框。。。



万幸的是, border (边框) 是会被继承的。元素不会有一个红边框。



一般来说，只有那些能给我们的
工作**带来便利的CSS 属性**才会有
拥有继承机制。



因而,究竟哪些
CSS属性可以被
继承呢?

下面的CSS 属性是可以**被继承**的...



azimuth, border-collapse, border-spacing,
caption-side, color, cursor, direction, elevation,
empty-cells, font-family, font-size, font-style,
font-variant, font-weight, font, letter-spacing,
line-height, list-style-image, list-style-position,
list-style-type, list-style, orphans, pitch-range,
pitch, quotes, richness, speak-header, speak-
numeral, speak-punctuation, speak, speech-
rate, stress, text-align, text-indent, text-
transform, visibility, voice-family, volume, white-
space, widows, word-spacing

啊？！怎么有 **这么多属性？**



为了简化理解，我们在这些属性集合中找到若干 **关键组** 出来。



文本相关的属性 是可以继承的：



azimuth, border-collapse, border-spacing,
caption-side, color, cursor, direction, elevation,
empty-cells, font-family, font-size, font-style,
font-variant, font-weight, font, letter-spacing,
line-height, list-style-image, list-style-position,
list-style-type, list-style, orphans, pitch-range,
pitch, quotes, richness, speak-header, speak-
numeral, speak-punctuation, speak, speech-
rate, stress, **text-align, text-indent, text-**
transform, visibility, voice-family, volume, white-
space, widows, word-spacing

List（列表）相关的属性也是可以继承的：



azimuth, border-collapse, border-spacing,
caption-side, color, cursor, direction, elevation,
empty-cells, font-family, font-size, font-style,
font-variant, font-weight, font, letter-spacing,
line-height, list-style-image, list-style-position,
list-style-type, list-style, orphans, pitch-range,
pitch, quotes, richness, speak-header, speak-
numeral, speak-punctuation, speak, speech-
rate, stress, text-align, text-indent, text-
transform, visibility, voice-family, volume, white-
space, widows, word-spacing

并且, 更重要的是
Color（颜色）属性 也是能继承的:



azimuth, border-collapse, border-spacing,
caption-side, **color**, cursor, direction, elevation,
empty-cells, font-family, font-size, font-style,
font-variant, font-weight, font, letter-spacing,
line-height, list-style-image, list-style-position,
list-style-type, list-style, orphans, pitch-range,
pitch, quotes, richness, speak-header, speak-
numeral, speak-punctuation, speak, speech-
rate, stress, text-align, text-indent, text-
transform, visibility, voice-family, volume, white-
space, widows, word-spacing

font-size可以
继承吗？

简单的答案：“yes”。
然而，font-size 属性是采用了一种
与众不同 的继承方式。



不同于直接继承元素的原始属性值，
真正被继承的值是一种**被计算过的值**。



在解释font-size属性如何被继承之前，
我们看一看：

为啥**font-size**属性不被直接继承



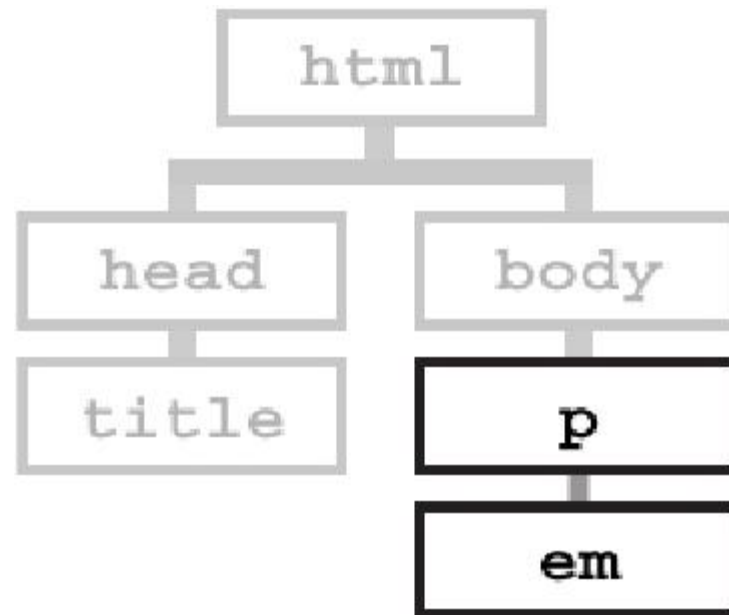
我们还是以先前用到的
同一**HTML 代码例子**开始

<p>

Lorem ipsum dolor
sit amet consectetur.

</p>

跟原来一样， ``元素
包含在 `<p>`之中。



现在，font-size 样式仅应用在
<p> 元素。 元素并没有被选中。

A screenshot of a code editor window with a light gray border and a white background. The code 'p { font-size: 80%; }' is displayed in a monospaced font. The text 'font-size: 80%;' is highlighted in red. The window has a standard macOS-style title bar with a red close button, a yellow maximize button, and a blue minimize button on the right side.

```
p { font-size: 80%; }
```

如果font-size 的数值有 80%被继承，
那么， 的大小将会是<p>元素的**80%**



那么，实际渲染出的页面应该如下图所示：



然而, **这不是事实!**
 元素的大小与<p>一模一样!



那么， **font-size**究竟是如何继承法儿呢？



我们来看看 三个例子...



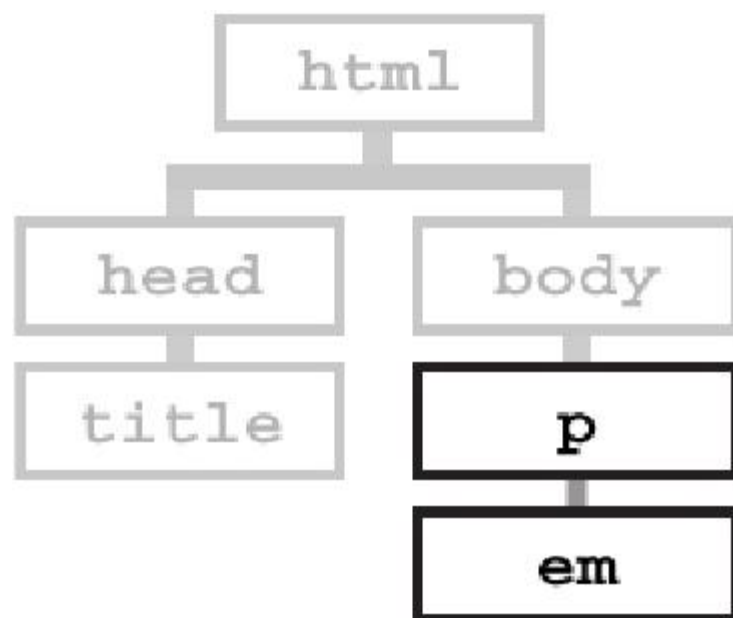
当然，用的例子还是
先前的**HTML**代码

<p>

Lorem ipsum dolor
sit amet consectetur.

</p>

该代码对应的文档树也是一样滴



例1:

Pixels （像素）

<p> 元素被赋予**14px** 的字体大小。

注： 不推荐以像素作为字体大小单位，因为一些老的浏览器比如IE5,IE6等其可访问性存在问题。



```
p { font-size: 14px; }
```

像素值 (14px) 覆盖了浏览器的缺省
font-size字体值 (大约16px).
新的像素值 (14px) 被后代
元素继承下去！



这样, 元素继承的数值为:

14px

元素	属性值	计算值
缺省字体大小	大约16px	
<body>	未指定	大约16px
<p>	14px	14px
	未指定	继承值 = 14px

例 2: 百分比

CSS给出的 <p> 元素的属性值
font-size 为 **85%**



```
p { font-size: 85%; }
```


浏览器缺省font-size(16px) 和
百分比值（85%）一起进行运算：
(16px x 85% = 13.6px)

后代元素继承的是该运算结果（13.6px）。



the 元素继承了这个
13.6px 计算值。

元素	属性值	计算值
缺省字体大小	大约16px	
<body>	未指定	大约16px
<p>	85%	16px x 85% = 13.6px
	未指定	继承的值 = 13.6px

例3:

EM

<p> 元素得到的font-size值为
.85em.

Note: Avoid using EMs for font-size values under 1em as IE5 renders these values in pixels instead of EMs (.8em is rendered as 8px).

A screenshot of a web browser window. The address bar is empty. The main content area displays a CSS rule: `p { font-size: .85em; }`. The text is rendered in a serif font, and the value `.85em` is highlighted in red. The browser's scrollbar is visible on the right side.

```
p { font-size: .85em; }
```

浏览器缺省值font-size(16px)
根据EM 值 (.85em)进行计算，可以
得出一个计算结果：

$$16\text{px} \times .85\text{em} = 13.6\text{px}$$

该计算结果由后代元素所继承！



这样, 元素所继承的是
13.6px 计算值

元素	属性值	计算值
缺省字体大小	大约16px	
<body>	未指定	大约16px
<p>	.85em	16px x .85em = 13.6px
	未指定	继承的值 = 13.6px

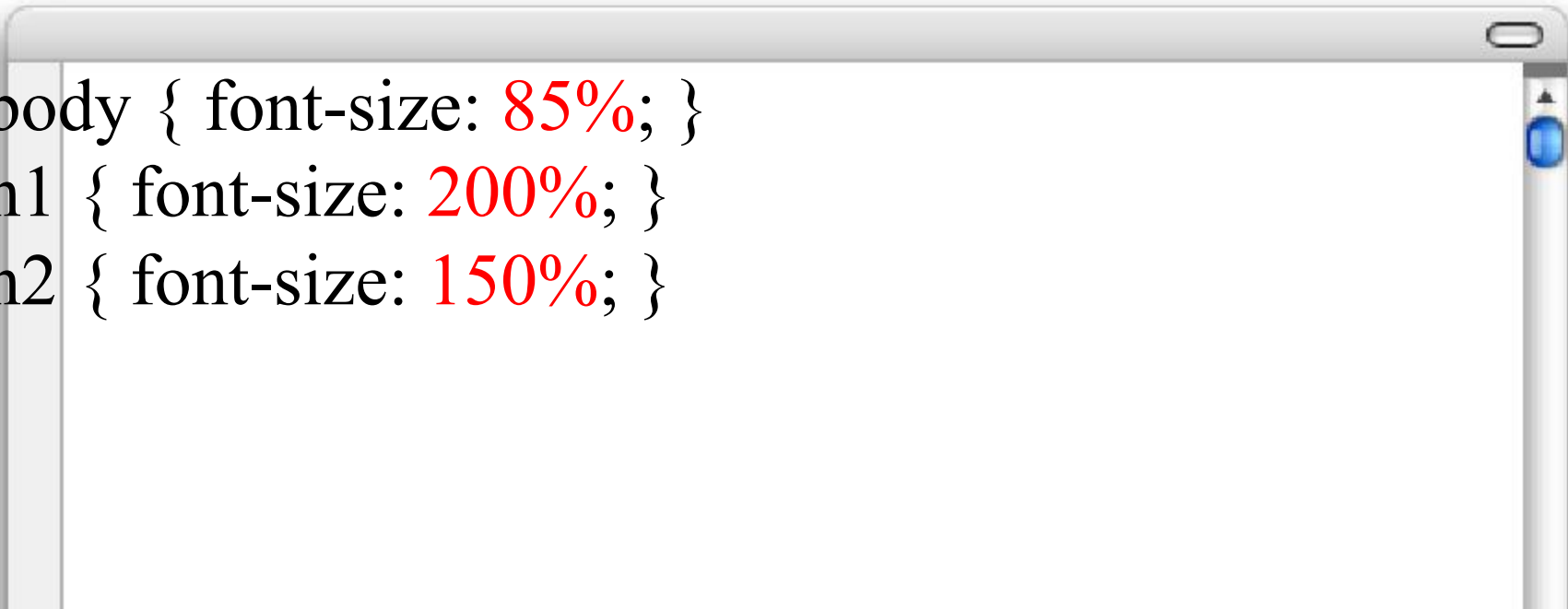
刚才的案例都非常的简单。

要是再复杂一些的例子，
使用**不同的元素**会怎样呢？



例4:

所有元素都使用**百分比** 大小



```
body { font-size: 85%; }  
h1 { font-size: 200%; }  
h2 { font-size: 150%; }
```

浏览器font-size缺省值 (16px)
body元素的百分比值 (85%) 进行计算,
得出计算结果 ($16\text{px} \times 85\% = 13.6\text{px}$).
该结果被后代元素所继承

除非有新的数值被指定。

font-size 继承 的步骤

元素	属性值	font-size计算值
缺省字体大小	大约16px	
<body>	85%	16px x 85% = 13.6px
<h1>	200%	继承值13.6px x 200% = 27.2px
<h2>	150%	继承值13.6px x 150% = 20.4px
<p>	未指定	继承值= 13.6px
	未指定	继承值= 13.6px

更有效地使用
属性继承

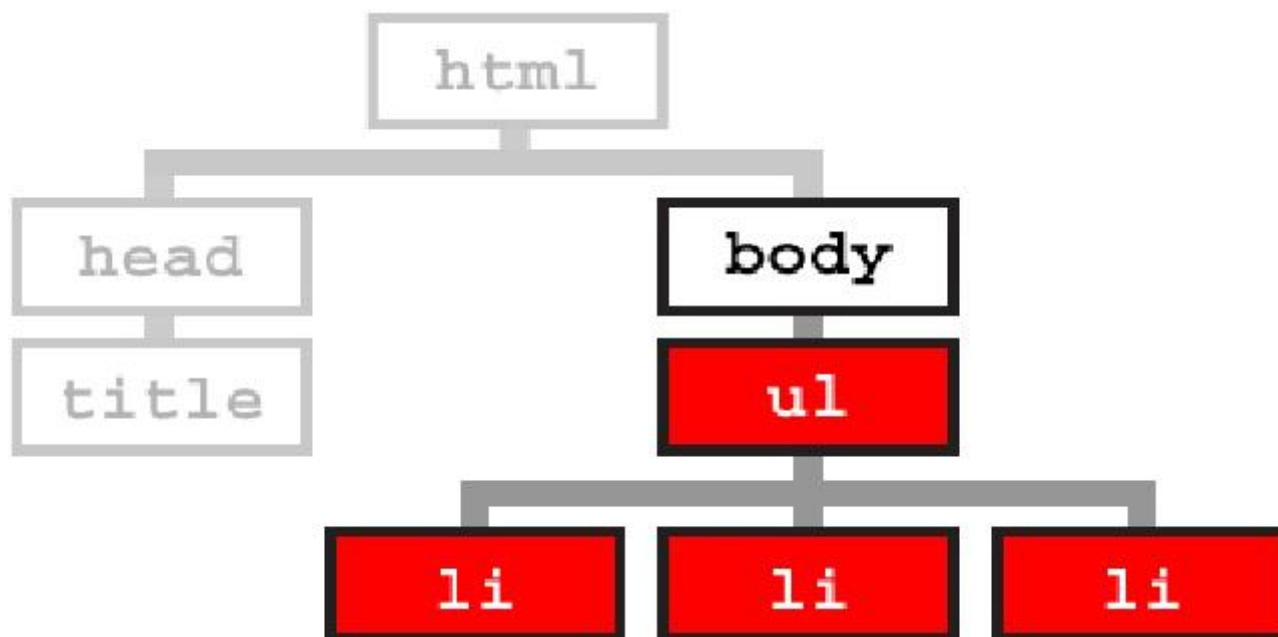
网页作者可以使用继承机制
编写 **高效的 CSS**.



例如，你可以设置
color, font-size 和font-family 在
body 元素上

```
body {  
    color: #222;  
    font-family: arial,  
    helvetica, sans-serif;  
    font-size: 90%;  
}
```

这些属性将被所有后代元素所继承



后面，你可以 **覆盖** 这些属性，
视需要，可以设置新的颜色值...


```
body {  
    color: #222;  
    font-family: arial,  
    helvetica, sans-serif;  
    font-size: 90%;  
}
```

```
h1, h2, h3 { color: green; }  
h4, h5, h6 { color: black; }
```

新的 **font-family values...**

```
body {  
    color: #222;  
    font-family: arial,  
    helvetica, sans-serif;  
    font-size: 90%;  
}
```

```
h1, h2, h3 { color: green; }  
h4, h5, h6 { color: black; }
```

```
h1, h2, h3, h4, h5, h6 {  
    font-family: georgia,  
    times, serif;  
}
```

和新的 **font-size** 值

}

```
h1, h2, h3 { color: green; }  
h4, h5, h6 { color: black; }
```

```
h1, h2, h3, h4, h5, h6 {  
    font-family: georgia,  
    times, serif;  
}
```

```
h1 { font-size: 200%; }  
h2 { font-size: 150%; }  
h3 { font-size: 125%; }  
#footer { font-size: 90%; }
```

OK，立刻开始吧！好好“使用继承”！



总结：属性继承

- 文本显示相关的属性 是被后代元素所 继承 的.
- 盒模型相关的样式属性，如div、p等其他块标签元素创建的盒子，如 borders, padding, margins, 内容区高宽 是 不能继承的.

没有了！ 结束。