# 一、盒模型

## 1.1 盒子中的区域

一个盒子中主要的属性就5个：width、height、padding、border、margin。

width是“宽度”的意思，CSS中width指的是内容的宽度，而不是盒子的宽度。

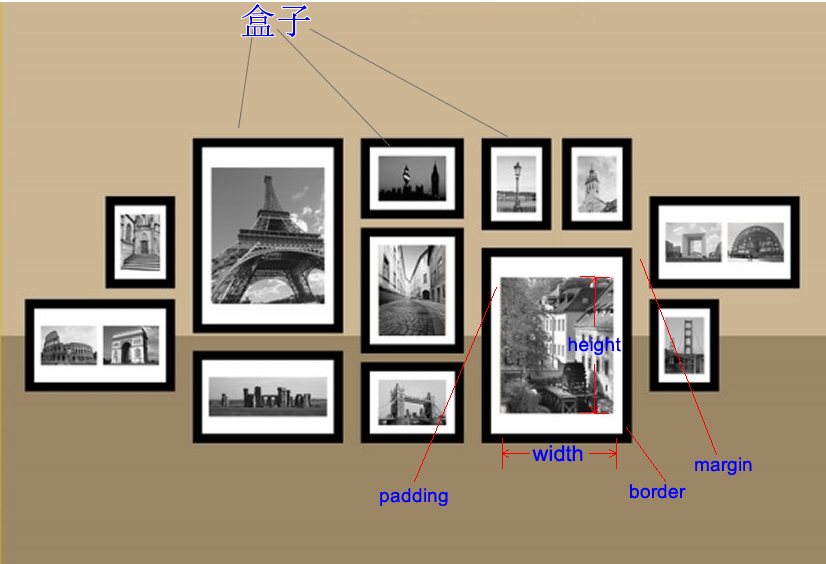
height是“高度”的意思，CSS中height指的是内容的高度，而不是盒子的高度

padding是“内边距”的意思

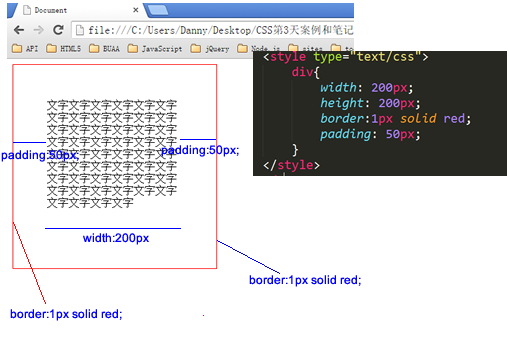
border是“边框”

margin是“外边距”

盒模型的示意图：



代码演示：



**这个盒子width:200px; height:200px; 但是真实占有的宽高是302\*302。 这是因为还要加上padding、border。**

**宽度和真实占有宽度，不是一个概念！！**

## 1.2 认识width、height

下面这两个盒子，真实占有宽高，完全相同，都是302\*302：

|  |
| --- |
| 1. .box1{ 2. width: 100px; 3. height: 100px; 4. padding: 100px; 5. border: 1px solid red; 6. } 7. .box2{ 8. width: 250px; 9. height: 250px; 10. padding: 25px; 11. border:1px solid red; 12. } |

**真实占有宽度= 左border + 左padding + width + 右padding + 右border**

这两个盒子的盒模型图，见下表：

|  |  |
| --- | --- |
|  |  |

小练习，大家自己写三个402\*402的盒子

**答案（答案有无穷多种，我们只写其中三种）：**

|  |
| --- |
| 1. .box1{ 2. width: 400px; 3. height: 400px; 4. border: 1px solid red; 5. } 6. .box2{ 7. width: 200px; 8. height: 200px; 9. border: 6px solid red; 10. padding: 95px; 11. } 12. .box3{ 13. width: 0px; 14. height: 0px; 15. padding: 200px; 16. border: 1px solid red; 17. } |

这三个盒子的盒模型图：

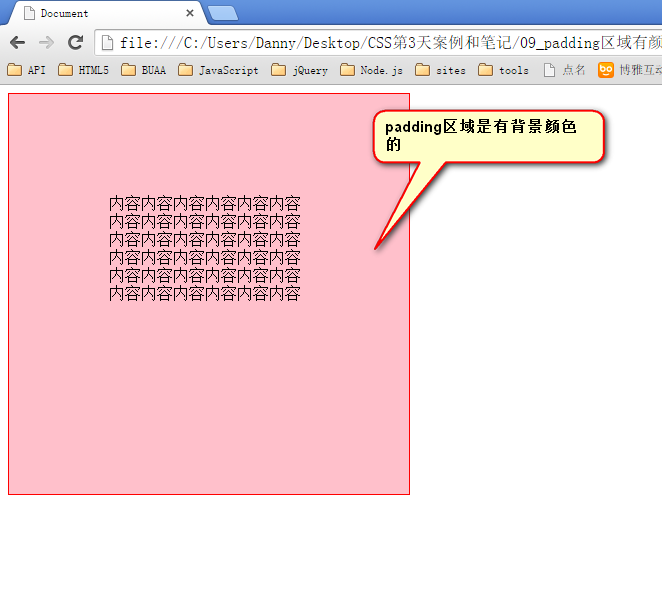
|  |  |  |
| --- | --- | --- |
|  |  |  |

如果想保持一个盒子的真是占有宽度不变，那么加width就要减padding。加padding就要减width。

## 1.3 认识padding

padding就是内边距。padding的区域有背景颜色，css2.1前提下，并且背景颜色一定和内容区域的相同。

也就是说，background-color将填充所有boder以内的区域。



padding是4个方向的，所以我们能够分别描述4个方向的padding。

方法有两种，第一种写小属性；第二种写综合属性，用空格隔开。

小属性：

|  |
| --- |
| 1. padding-top: 30px; 2. padding-right: 20px; 3. padding-bottom: 40px; 4. padding-left: 100px; |

top上、right右、bottom下、left左。

这种属性，就是复合属性。比如不写padding-left那么就是没有左内边距。

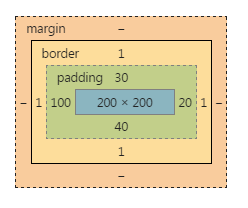
快捷键就是pdt、pdr、pdb、pdl 然后按tab。

综合属性：

如果写了4个值：

|  |
| --- |
| 1. padding:30px 20px 40px 100px; |

上、右、下、左



空格隔开的，四个数字就是上、右、下、左。

也就是说，前端开发工程师眼中的顺序不一样。

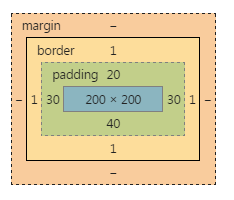
老百姓：上下左右

强调开发工程师：上、右、下、左

如果只写3个值：

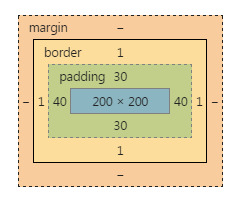
|  |
| --- |
| 1. padding: 20px 30px 40px; |

上、右、下、??和右一样



如果只写2个值：

|  |
| --- |
| 1. padding: 30px 40px; |



也就是说，

|  |
| --- |
| 1. padding: 30px 40px; |

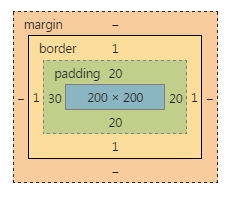
等价于：

|  |
| --- |
| 1. padding-top: 30px; 2. padding-bottom: 30px; 3. padding-left: 40px; 4. padding-right: 40px; |

要懂得，用小属性层叠大属性：

|  |
| --- |
| 1. padding: 20px; 2. padding-left: 30px; |

对应的盒模型图：

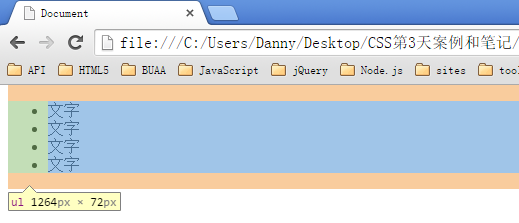


下面的写法错误：

|  |
| --- |
| 1. padding-left: 30px; 2. padding: 20px; |

不能把小属性，写在大属性前面。

一些元素，默认带有padding，比如ul标签。



所以，我们为了做站的时候，便于控制，总是喜欢清除这个默认的padding：

|  |
| --- |
| 1. \*{ 2. margin: 0; 3. padding: 0; 4. } |

\*的效率不高，所以我们使用并集选择器，罗列所有的标签（不用背，有专业的清除默认样式的样式表，今后学习）：

|  |
| --- |
| 1. body,div,dl,dt,dd,ul,ol,li,h1,h2,h3,h4,h5,h6,pre,code,form,fieldset,legend,input,textarea,p,blockquote,th,td{ 2. margin:0; 3. padding:0 4. } |

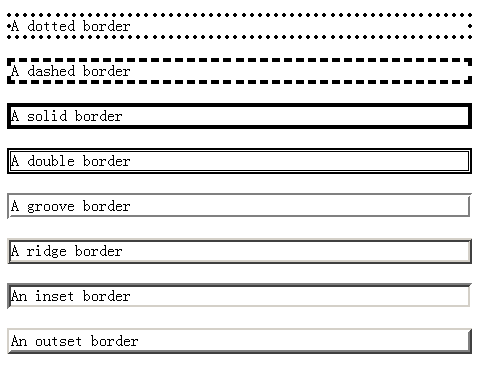
## 1.4 border

就是边框。边框有三个要素：粗细、线型、颜色。

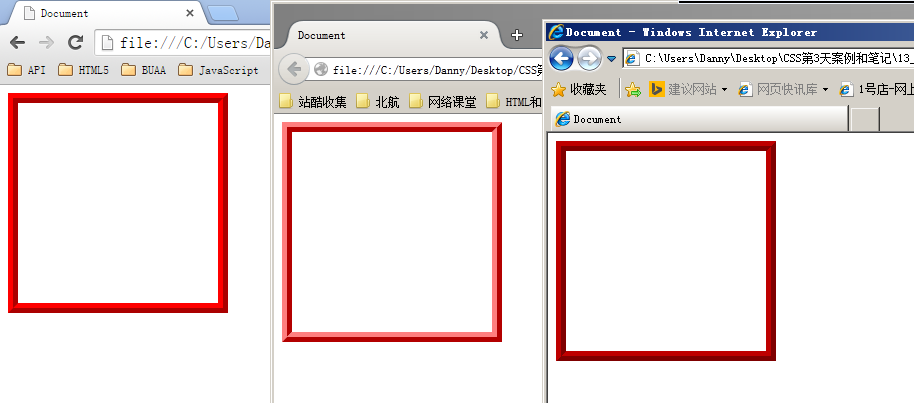
颜色如果不写，默认是黑色。另外两个属性不写，要命了，显示不出来边框。

|  |
| --- |
| 1. border: 1px dashed red; |

所有的线型：



比如，border:10px ridge red; 在chrome和firefox、IE中有细微差别：



如果公司里面的设计师，处女座的，追求极高的页面还原度，那么不能使用css来制作边框。

就要用到图片，就要切图了。所以，比较稳定的就几个：solid、dashed、dotted

border是一个大综合属性，

|  |
| --- |
| 1. border:1px solid red; |

就是把4个边框，都设置为1px宽度、线型实线、red颜色。

border属性能够被拆开，有两大种拆开的方式：

1） 按3要素:border-width、border-style、border-color

2） 按方向：border-top、border-right、border-bottom、border-left

按3要素拆开：

|  |
| --- |
| 1. **border-width**:10px; → 边框宽度 2. **border-style**:solid; → 线型 3. **border-color**:red; → 颜色。 |

等价于：

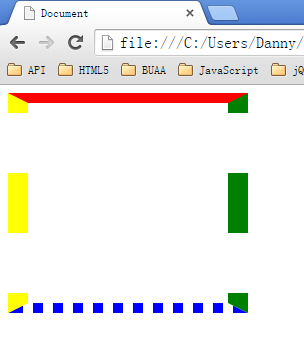
|  |
| --- |
| 1. border:10px solid red; |

现在心里要明白，**原来一个border是由三个小属性综合而成：**

**border-width border-style border-color。**

如果某一个小要素后面是空格隔开的多个值，那么就是**上右下左**的顺序：

|  |
| --- |
| 1. border-width:10px 20px; 2. border-style:**solid dashed dotted**; 3. border-color:**red green blue yellow**; |



按方向来拆

|  |
| --- |
| 1. border-top:10px solid red; 2. border-right:10px solid red; 3. border-bottom:10px solid red; 4. border-left:10px solid red; |

等价于

|  |
| --- |
| 1. border:10px solid red; |

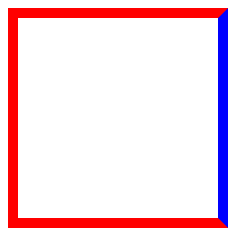
按方向还能再拆一层，就是把每个方向的，每个要素拆开，一共12条语句：

|  |
| --- |
| 1. border-top-width:10px; 2. border-top-style:solid; 3. border-top-color:red; 4. border-right-width:10px; 5. border-right-style:solid; 6. border-right-color:red; 7. border-bottom-width:10px; 8. border-bottom-style:solid; 9. border-bottom-color:red; 10. border-left-width:10px; 11. border-left-style:solid; 12. border-left-color:red; |

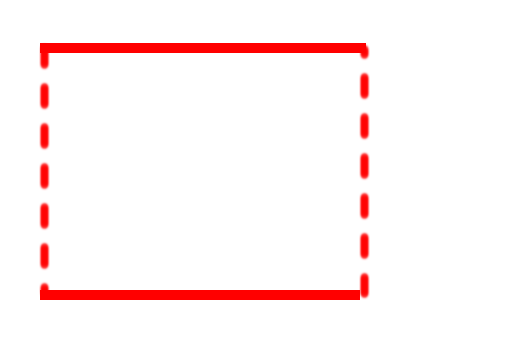
等价于

|  |
| --- |
| 1. border:10px solid red; |

工作中到底用什么？很简答：什么简单用什么？

 写法：

|  |
| --- |
| 1. border:10px solid red; 2. border-right-color:blue; |

写法：

|  |
| --- |
| 1. border:10px solid red; 2. border-style:solid dashed; |

border可以没有，

|  |
| --- |
| 1. border:none; |

某一条边没有：

|  |
| --- |
| 1. border-left: none; |

也可以调整左边边框的宽度为0：

|  |
| --- |
| 1. border-left-width: 0; |

# 二、标准文档流

宏观的讲，我们的web页面和photoshop等设计软件有本质的区别：web页面的制作，是个“流”，必须从上而下，像“织毛衣”。而设计软件，想往哪里画个东西，都能画。

我们要看看标准流有哪些微观现象：

1） 空白折叠现象：

2） 高矮不齐，底边对齐：

3） 自动换行，一行写不满，换行写。

## 2.1 块级元素和行内元素

学习的初期，你就要知道，标准文档流等级森严。标签分为两种等级：

1） 块级元素

● 霸占一行，不能与其他任何元素并列

● 能接受宽、高

● 如果不设置宽度，那么宽度将默认变为父亲的100%。

2） 行内元素

● 与其他行内元素并排

● 不能设置宽、高。默认的宽度，就是文字的宽度。

在HTML中，我们已经将标签分过类，当时分为了：文本级、容器级。

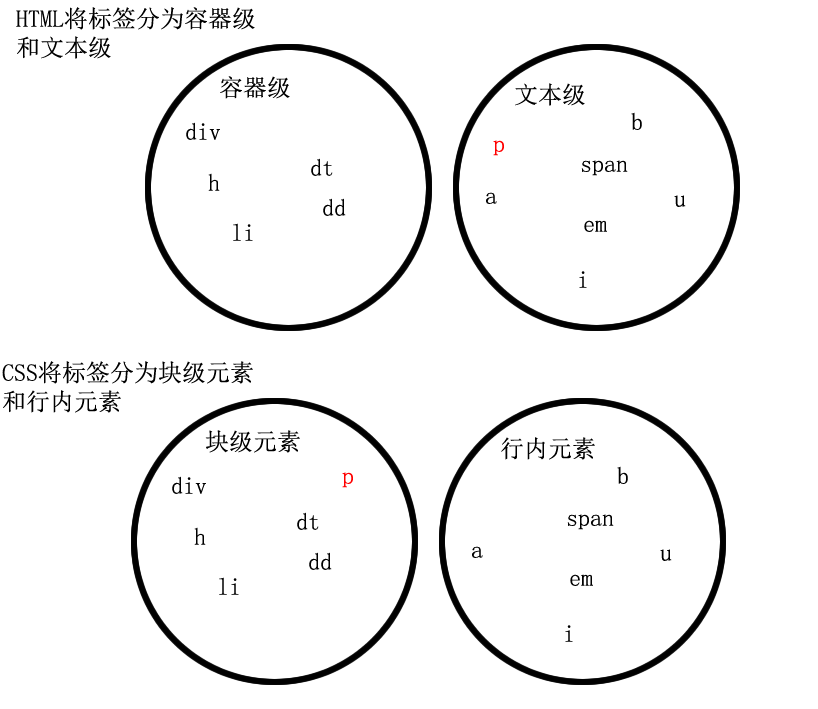
文本级：p、span、a、b、i、u、em

容器级：div、h系列、li、dt、dd

CSS的分类和上面的很像，就p不一样：

所有的文本级标签，都是行内元素，除了p，p是个文本级，但是是个块级元素。

所有的容器级标签都是块级元素。



## 2.2 块级元素和行内元素的相互转换

块级元素可以设置为行内元素

行内元素可以设置为块级元素

|  |
| --- |
| 1. div{ 2. **display: inline;** 3. background-color: pink; 4. width: 500px; 5. height: 500px; 6. } |

display是“显示模式”的意思，用来改变元素的行内、块级性质

inline就是“行内”。

一旦，给一个标签设置

|  |
| --- |
| 1. **display: inline;** |

那么，这个标签将立即变为行内元素。此时它和一个span无异：

● 此时这个div不能设置宽度、高度；

● 此时这个div可以和别人并排了

同样的道理，

|  |
| --- |
| 1. span{ 2. **display: block;** 3. width: 200px; 4. height: 200px; 5. background-color: pink; 6. } |

“block”是“块”的意思

让标签变为块级元素。此时这个标签，和一个div无异：

● 此时这个span能够设置宽度、高度

● 此时这个span必须霸占一行了，别人无法和他并排

● 如果不设置宽度，将撑满父亲

标准流里面限制非常多，标签的性质恶心。比如，我们现在就要并排、并且就要设置宽高。

所以，移民！脱离标准流！

css中一共有三种手段，使一个元素脱离标准文档流：

1） 浮动

2） 绝对定位

3） 固定定位