# 目录

[目录 1](#_Toc464647445)

[一、盒模型 2](#_Toc464647446)

[二、标准流文档 2](#_Toc464647447)

[2.1标准文档流 2](#_Toc464647448)

[2.2文档流标签分类 3](#_Toc464647449)

[2.3文档流标签类型转换 3](#_Toc464647450)

[三、浮动 4](#_Toc464647451)

[3.1 浮动的元素脱标 5](#_Toc464647452)

[3.2 浮动的元素互相贴靠 6](#_Toc464647453)

[3.3 浮动的元素有“字围”效果 7](#_Toc464647454)

[3.3 浮动的元素的收缩效果 7](#_Toc464647455)

[四、浮动的清除 9](#_Toc464647456)

[4.1 清除浮动方法1：给浮动的元素的祖先元素加高度。 9](#_Toc464647457)

[4.2 清除浮动方法2：clear:both; 10](#_Toc464647458)

[4.3 清除浮动方法3：隔墙法 11](#_Toc464647459)

[4.4 清除浮动方法4：overflow:hidden; 12](#_Toc464647460)

[4.5 清除浮动总结与案例 13](#_Toc464647461)

[4.6 浏览器兼容问题 15](#_Toc464647462)

[五、margin 17](#_Toc464647463)

[5.1 margin的塌陷现象 17](#_Toc464647464)

[5.2 盒子居中margin:0 auto; 17](#_Toc464647465)

[5.3 善于使用父亲的padding，而不是儿子的margin 18](#_Toc464647466)

[5.4 关于margin的IE6兼容问题 19](#_Toc464647467)

[六、Fireworks和精确盒子还原 21](#_Toc464647468)

# 一、盒模型

盒模型box model，什么是盒子？所有的标签都是盒子。无论是div、span、a都是盒子。图片、表单元素一律看做文本。

盒模型有哪些组成：width、height、padding、border、margin。

width、height是内容的宽度、高度。

padding，内边距，边框和文字内容之间的距离。padding有颜色。表示方法，能够用padding综合写，4个值“上、右、下、左”，3个值“上、左右、下”，2个值“上下，左右”。还能按方向拆开，padding-left、padding-top、padding-right、padding-bottom。

border，边框，3要素，4条边。3要素：border-width、border-style、border-color；4条边：border-top、border-right、border-bottom、border-left。 比如我们要单独设置某一条边，那么就需要写清楚3要素：

|  |
| --- |
| 1. border-top:3px solid red; |

如果要单独设置要素：

|  |
| --- |
| 1. border-width:3px; 2. border-color:red; 3. border-style:solid; |

还能拆成最小：

|  |
| --- |
| 1. border-bottom-style:solid; |

常用线型：solid、dashed、dotted。

# 二、标准流文档

## 2.1标准文档流

：是网页排版“默认”状态。

标准流的微观现象：

1） 空白折叠现象：

比如，如果我们想让img标签之间没有空隙，必须紧密连接：

|  |
| --- |
| 1. <img src="images/0.jpg" /><img src="images/1.jpg" /><img src="images/2.jpg" /> |

2） 高矮不齐，底边对齐：



3） 自动换行，一行写不满，换行写。

## 2.2文档流标签分类

块级元素：占一行的，能设置宽、高，不设置宽度默认就是占满父亲。div、p、h、li

行内元素：和其他行内元素并排，不能设置宽、高，默认宽度就是文字宽度。span、a、b、i、u

## 2.3文档流标签类型转换

|  |
| --- |
| 行内🡪块  display:block;  块🡪行内  display:inline; |

标准流做不出网页：因为能并排的不能改宽高。所以，要脱离标准流。

**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**

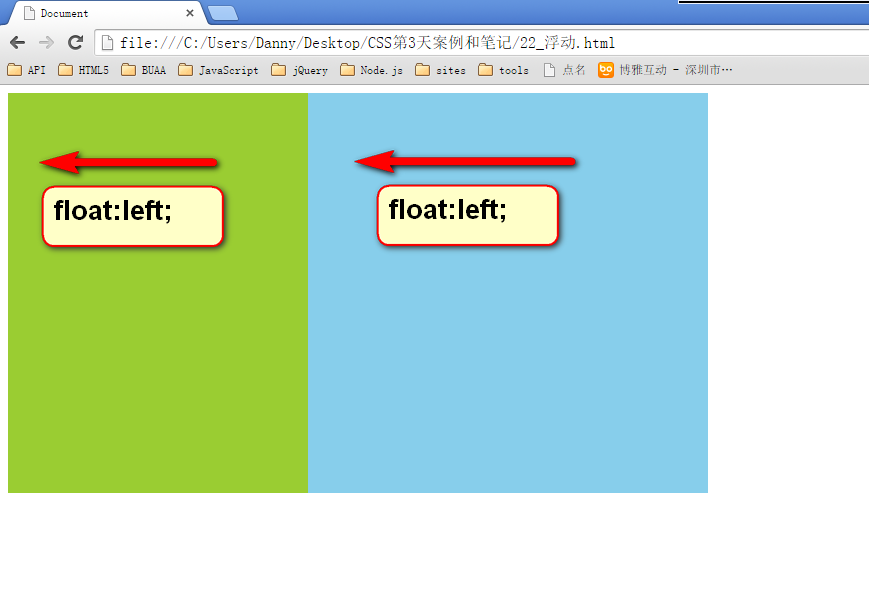
**3） 固定定位**

# 三、浮动

|  |
| --- |
| 1. float:left; 2. 或者 3. float:right; |

**浮动宏观的看，就是做“布局”的。有几个性质：脱标、贴边、字围、收缩。**

一个浮动的a、span ，是不需要设置display:block; 就能够设置宽高了。因为浮动之后，脱离标准流了，所以标准流里面的法律、规则都不适用了。

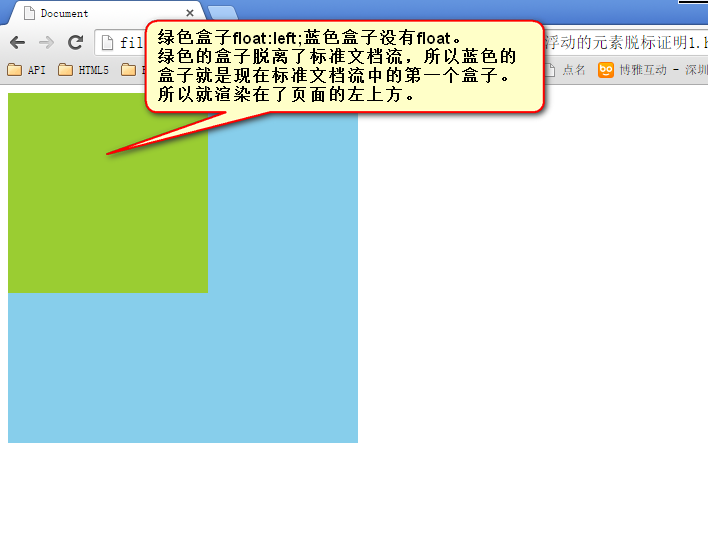


|  |
| --- |
| 1. .box1{ 2. float: left; 3. width: 300px; 4. height: 400px; 5. background-color: yellowgreen; 6. } 7. .box2{ 8. float: left; 9. width: 400px; 10. height: 400px; 11. background-color: skyblue; 12. } |

两个元素并排了，并且两个元素都能够设置宽度、高度了（这在刚才的标准流中，不能实现）。

## 3.1 浮动的元素脱标

证明1：



证明2：

一个span标签不需要转成块级元素，就能够设置宽度、高度了。所以能够证明一件事儿，就是所有标签已经不区分行内、块了。也就是说，**一旦一个元素浮动了，那么，将能够并排了，并且能够设置宽高了。无论它原来是个div还是个span。**

|  |
| --- |
| 1. span{ 2. **float: left;** 3. width: 200px; 4. height: 200px; 5. background-color: orange; 6. } |

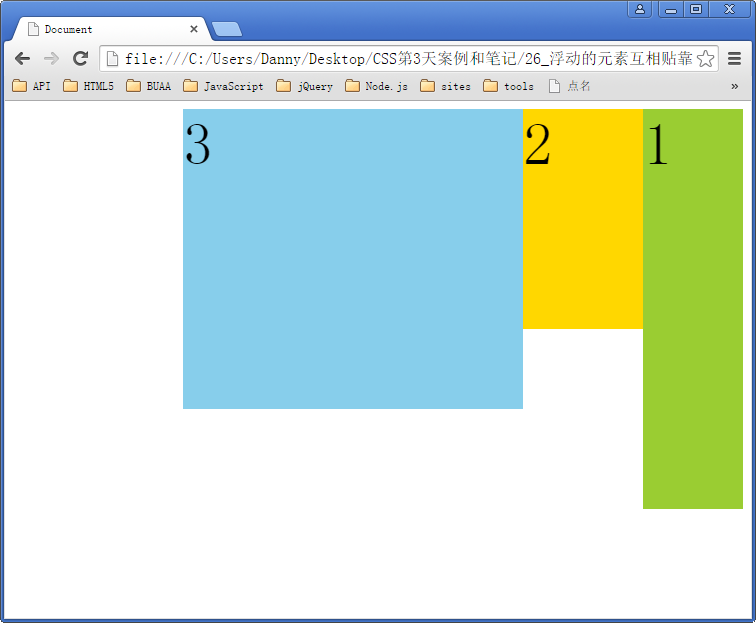
## 3.2 浮动的元素互相贴靠

如果有足够空间，3会靠着2。如果没有足够的空间，那么会靠着1。

如果没有足够的空间靠着1，自己去贴左墙。

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

右浮动： float:right;

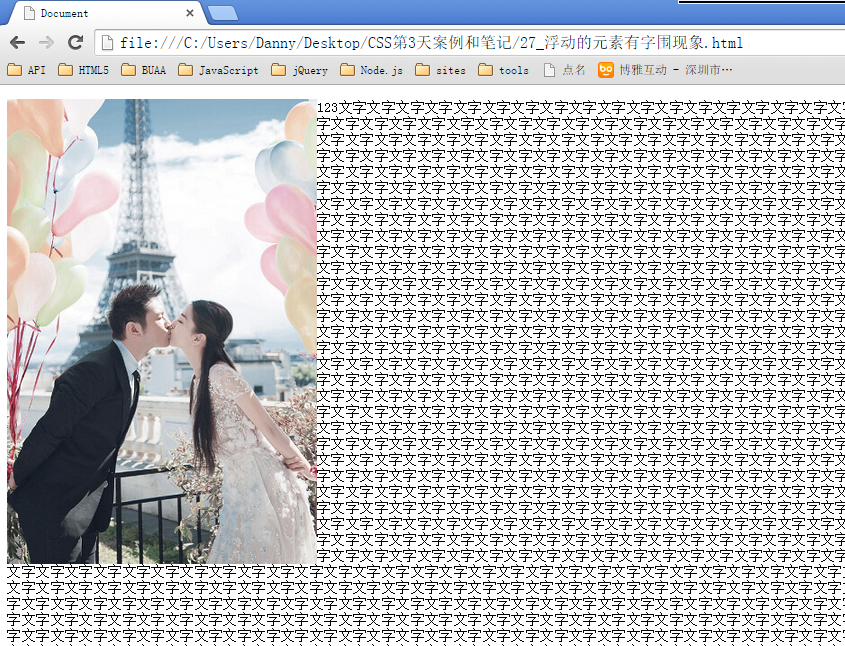


## 3.3 浮动的元素有“字围”效果

HTML：

|  |
| --- |
| 1. <div> 2. <img src="images/1.jpg" alt="" /> 3. </div> 4. <p>123文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字 5. </p> |

让div浮动，p不浮动：



div挡住了p，但是p中的文字不会被挡住，形成“字围”效果。

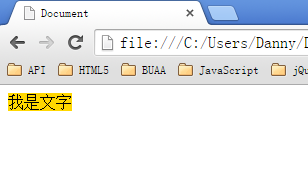
## 3.4 浮动的元素的收缩效果

收缩：一个浮动的元素，如果没有设置width，那么将自动收缩为文字的宽度（这点非常像行内元素）。

比如：

|  |
| --- |
| 1. <style type="text/css"> 2. div{ 3. float: left; 4. background-color: gold; 5. } 6. </style> |

这个div浮动了，且没有设置宽度，那么将自动缩紧为内容的宽度：



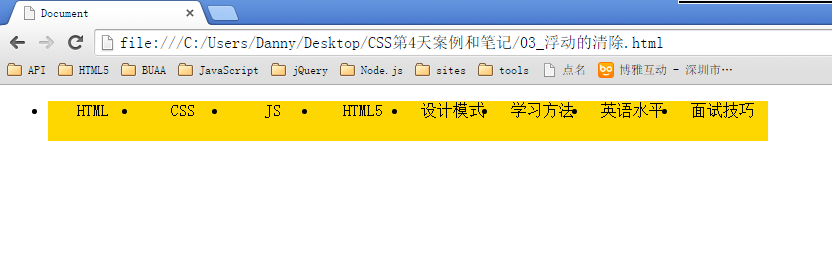
整个网页，就是通过浮动，来实现初步布局的。

# 四、浮动的清除

来看一个实验：现在有两个div，div身上没有任何属性。每个div中都有li，这些li都是浮动的。

|  |
| --- |
| 1. <div> 2. <ul> 3. <li>HTML</li> 4. <li>CSS</li> 5. <li>JS</li> 6. <li>HTML5</li> 7. <li>设计模式</li> 8. </ul> 9. </div> 10. <div> 11. <ul> 12. <li>学习方法</li> 13. <li>英语水平</li> 14. <li>面试技巧</li> 15. </ul> 16. </div> |

我们本以为这些li，会分为两排，但是，第二组中的第1个li，去贴靠第一组中的最后一个li了。

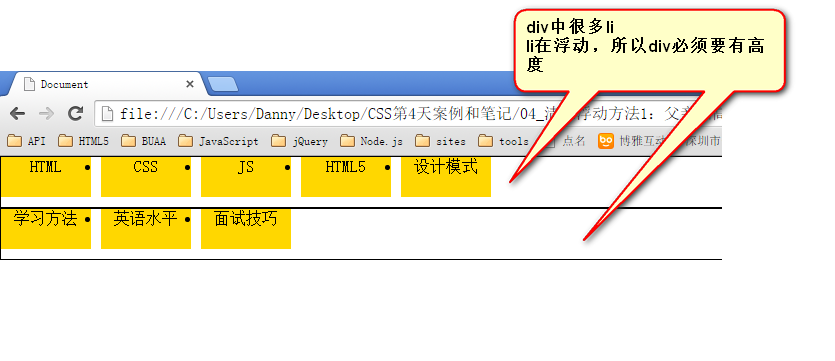


第二个div中的li，去贴第一个div中最后一个li的边了。

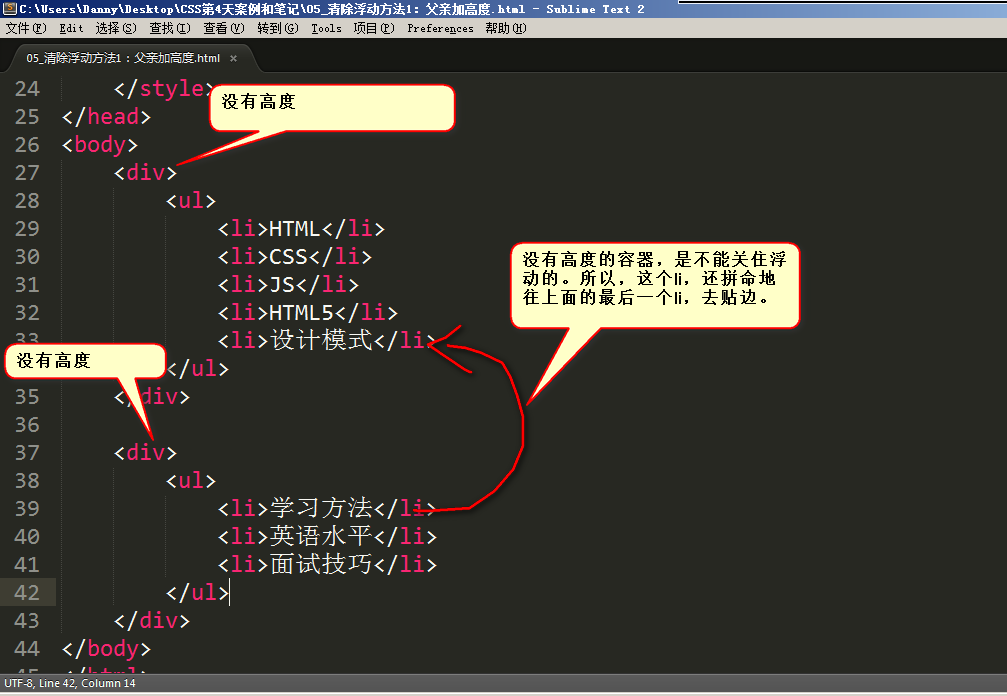
原因就是因为div没有高度，不能给自己浮动的孩子们，一个容器。

## 4.1 清除浮动方法1：给浮动的元素的祖先元素加高度。

如果一个元素要浮动，那么它的祖先元素一定要有高度。**高度的盒子，才能关住浮动。**



只要浮动在一个有高度的盒子中，那么这个浮动就不会影响后面的浮动元素。所以就是清除浮动带来的影响了。



## 4.2 清除浮动方法2：clear:both;

网页制作中，高度height很少出现。因为能被内容撑高！方法1，工作中用的很少。

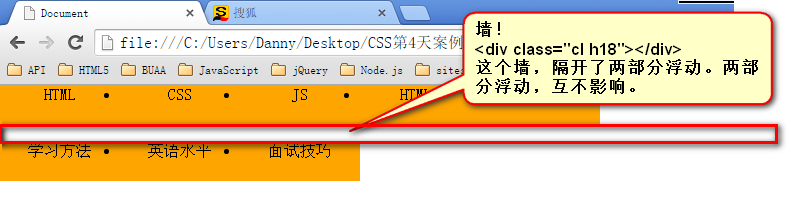
|  |
| --- |
| 1. <div> 2. <ul> 3. <li>HTML</li> 4. <li>CSS</li> 5. <li>JS</li> 6. <li>HTML5</li> 7. <li>设计模式</li> 8. </ul> 9. </div> 10. <div **class="box2"**> → 这个div写一个**clear:both;**属性 11. <ul> 12. <li>学习方法</li> 13. <li>英语水平</li> 14. <li>面试技巧</li> 15. </ul> 16. </div> |

|  |
| --- |
| 1. **clear:both;** |

clear就是清除，both指的是左浮动、右浮动都要清除。意思就是：清除别人对我的影响。

**这种方法有一个非常大的、致命的问题，margin失效了。**

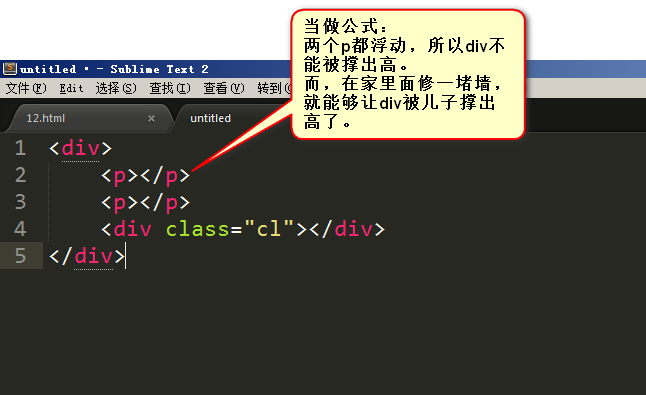
## 4.3 清除浮动方法3：隔墙法



|  |
| --- |
| 1. <div class="box1"> 2. <ul> 3. <li>HTML</li> 4. <li>CSS</li> 5. <li>JS</li> 6. <li>HTML5</li> 7. <li>设计模式</li> 8. </ul> 9. </div> 11. **<div class="cl h16"></div>** 12. <div class="box2"> 13. <ul> 14. <li>学习方法</li> 15. <li>英语水平</li> 16. <li>面试技巧</li> 17. </ul> 18. </div> |

|  |
| --- |
| 1. .cl{ 2. clear: both; 3. } 4. .h16{ 5. height: 16px; 6. } |

近些年，有演化出了“内墙法”：



## 4.4 清除浮动方法4：overflow:hidden;

overflow就是“溢出”的意思， hidden就是“隐藏”的意思。

|  |
| --- |
| 1. overflow:hidden; |

表示“溢出隐藏”。所有溢出边框的内容，都要隐藏掉。

|  |  |
| --- | --- |
| 内容太多，溢出了盒子： | overflow:hidden; 溢出盒子边框的内容，隐藏了。 |

本意就是清除溢出到盒子外面的文字。

一个父亲不能被自己浮动的儿子，撑出高度。但是，只要给父亲加上overflow:hidden; 那么，父亲就能被儿子撑出高了。

|  |
| --- |
| 1. div{ 2. width: 400px; 3. border: 10px solid black; 4. **overflow: hidden;** 5. } |

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



## 4.5 清除浮动总结与案例

总结一下：

1） 加高法：

浮动的元素，只能被有高度的盒子关住。 也就是说，如果盒子内部有浮动，这个盒子有高，那么妥妥的，浮动不会互相影响。但是，工作上，我们绝对不会给所有的盒子加高度，这是因为麻烦，并且不能适应页面的快速变化。

|  |
| --- |
| 1. <div> **→ 设置height** 2. <p></p> 3. <p></p> 4. <p></p> 5. </div> 6. <div> **→ 设置height** 7. <p></p> 8. <p></p> 9. <p></p> 10. </div> |

2) clear:both;法

最简单的清除浮动的方法，就是给盒子增加clear:both；表示自己的内部元素，不受其他盒子的影响。

|  |
| --- |
| 1. <div> 2. <p></p> 3. <p></p> 4. <p></p> 5. </div> 6. <div> **→ clear:both;** 7. <p></p> 8. <p></p> 9. <p></p> 10. </div> |

浮动确实被清除了，不会互相影响了。但是有一个问题，就是margin失效。两个div之间，没有任何的间隙了。

3）隔墙法：

在两部分浮动元素中间，建一个墙。隔开两部分浮动，让后面的浮动元素，不去追前面的浮动元素。

墙用自己的身体当做了间隙。

|  |
| --- |
| 1. <div> 2. <p></p> 3. <p></p> 4. <p></p> 5. </div> 6. **<div class="cl h10"></div>** 7. <div> 8. <p></p> 9. <p></p> 10. <p></p> 11. </div> |

我们发现，隔墙法好用，但是第一个div，还是没有高度。如果我们现在想让第一个div，自动的根据自己的儿子，撑出高度，我们就要想一些“小伎俩”，“奇淫技巧”。

内墙法：

|  |
| --- |
| 1. <div> 2. <p></p> 3. <p></p> 4. <p></p> 5. **<div class="cl h10"></div>** 6. </div> 7. <div> 8. <p></p> 9. <p></p> 10. <p></p> 11. </div> |

内墙法的优点就是，不仅仅能够让后部分的p不去追前部分的p了，并且能把第一个div撑出高度。这样，这个div的背景、边框就能够根据p的高度来撑开了。

4）overflow:hidden;

这个属性的本意，就是将所有溢出盒子的内容，隐藏掉。但是，我们发现这个东西能够用于浮动的清除。

我们知道，一个父亲，不能被自己浮动的儿子撑出高度，但是，如果这个父亲加上了overflow:hidden；那么这个父亲就能够被浮动的儿子撑出高度了。这个现象，不能解释，就是浏览器的小偏方。

并且,overflow:hidden;能够让margin生效。

清除浮动的案例：



实践中，遇见了清除浮动的问题：



## 4.6 浏览器兼容问题

**上述知识点遇见的浏览器兼容问题**

第一，IE6，不支持小于12px的盒子，任何小于12px的盒子，在IE6中看都大

解决办法很简单，就是将盒子的字号，设置小（小于盒子的高），比如0px。

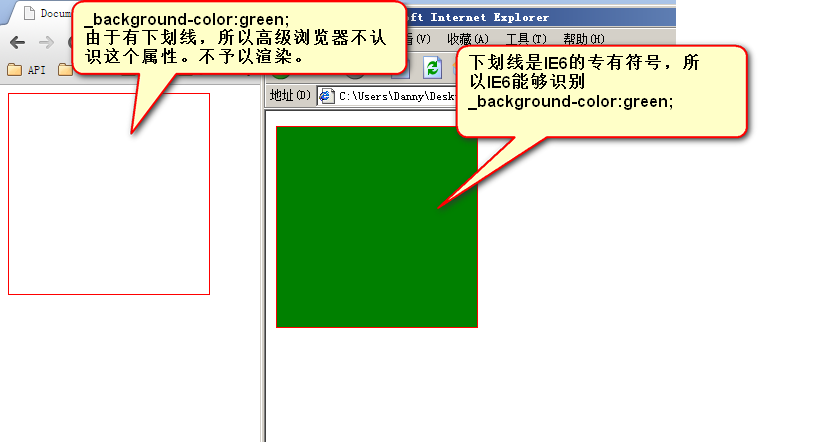
|  |
| --- |
| 1. height: 4px; 2. \_font-size: 0px; |

我们现在介绍一下浏览器hack。hack就是“黑客”，就是使用浏览器提供的后门，针对某一种浏览器做兼容。

**IE6留了一个后门，就是只要给css属性之前，加上下划线，这个属性就是IE6认识的专有属性。**

比如：

|  |
| --- |
| 1. \_background-color: green; |



解决微型盒子，正确写法：

|  |
| --- |
| 1. height: 10px; 2. **\_font-size:0;** |

第二，IE6不支持用overflow:hidden;来清除浮动的

解决办法，以毒攻毒。追加一条

|  |
| --- |
| 1. \_zoom:1; |

完整写法：

|  |
| --- |
| 1. overflow: hidden; 2. **\_zoom:1;** |

实际上，\_zoom:1;能够触发浏览器hasLayout机制。这个机制，不要深究了，因为就IE6有。我们只需要让IE6好用，具体的实现机制，有兴趣的同学，自行百度。

强调一点， overflow:hidden;的本意，就是溢出盒子的border的东西隐藏，这个功能是IE6兼容的。不兼容的是overflow:hidden;清除浮动的时候。

我们刚才学习了两个IE6的兼容问题，这两个IE6的兼容问题，都是通过多写一条hack来解决的。

这个我们称为伴生属性。

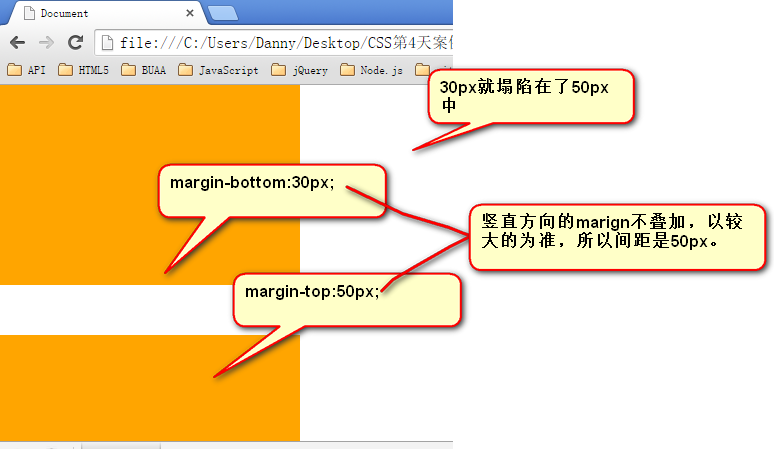
|  |
| --- |
| 1. height:6px; 2. **\_font-size:0;** |

|  |
| --- |
| 1. overflow:hidden; 2. **\_zoom:1;** |

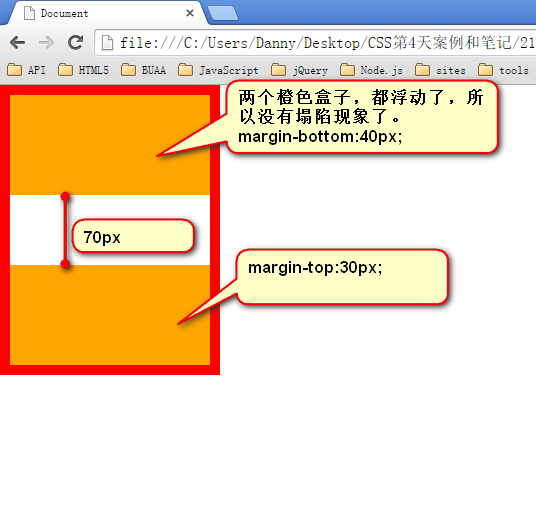
# 五、margin

## 5.1 margin的塌陷现象

**标准文档流中，竖直方向的margin不叠加，以较大的为准。**



如果不在标准流，比如盒子都浮动了，那么两个盒子之间是没有塌陷现象的：



## 5.2 盒子居中margin:0 auto;

margin的值可以为auto，表示自动。当left、right两个方向，都是auto的时候，盒子居中了：

|  |
| --- |
| 1. margin-left: auto; 2. margin-right: auto; |

简写为

|  |
| --- |
| 1. margin:0 auto; |

注意：

1） 使用margin:0 auto; 的盒子，必须有width，有明确的width

2） 只有标准流的盒子，才能使用margin:0 auto; 居中。

也就是说，当一个盒子浮动了、绝对定位了、固定定位了，都不能使用margin:0 auto;

3） margin:0 auto;是在居中盒子，不是居中文本。

文本的居中，要使用

|  |
| --- |
| 1. text-align:center; |

|  |
| --- |
| 1. margin:0 auto; → 让这个div自己在大容器中居中。 2. text-align: center; → 让这个div内部的文本居中。 |

普及一下知识，text-align还有

|  |
| --- |
| 1. text-align:left; 没啥用，因为默认居左 2. text-align:right; 文本居右 |

## 5.3 善于使用父亲的padding，而不是儿子的margin

如果父亲没有border，那么儿子的margin实际上踹的是“流”，踹的是这“行”。所以，父亲整体也掉下来了

这个p有一个margin-top踹父亲，试图将自己下移

|  |
| --- |
| 1. <div> 2. <p></p> 3. </div> |

结果：

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

**margin这个属性，本质上描述的是兄弟和兄弟之间的距离； 最好不要用这个marign表达父子之间的距离。**

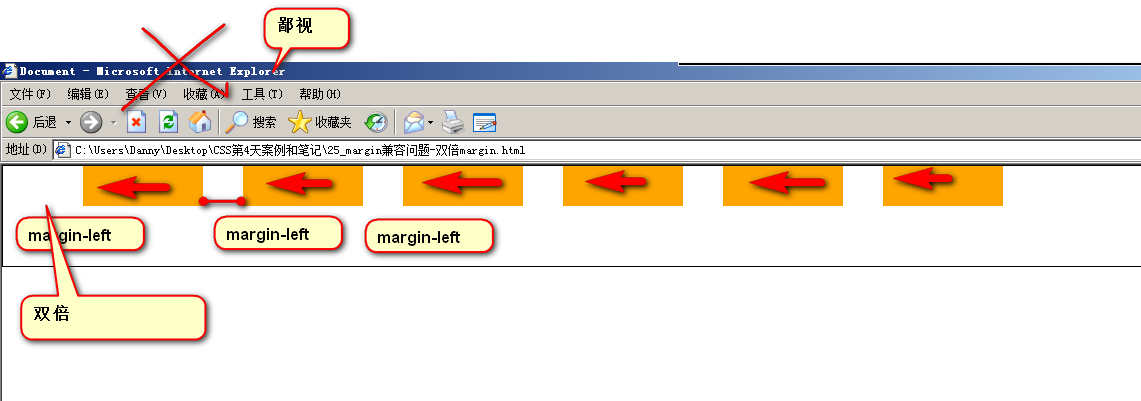
所以，我们一定要善于使用父亲的padding，而不是儿子的margin。

## 5.4 关于margin的IE6兼容问题

IE6双倍margin bug

**当出现连续浮动的元素，携带和浮动方向相同的margin时，队首的元素，会双倍marign。**

|  |
| --- |
| 1. <ul> 2. <li></li> 3. <li></li> 4. <li></li> 5. </ul> |



解决方案：

1）使浮动的方向和margin的方向，相反。

所以，你就会发现，我们特别喜欢，浮动的方向和margin的方向相反。并且，前端开发工程师，把这个当做习惯了。

|  |
| --- |
| 1. float: left; 2. margin-right: 40px; |

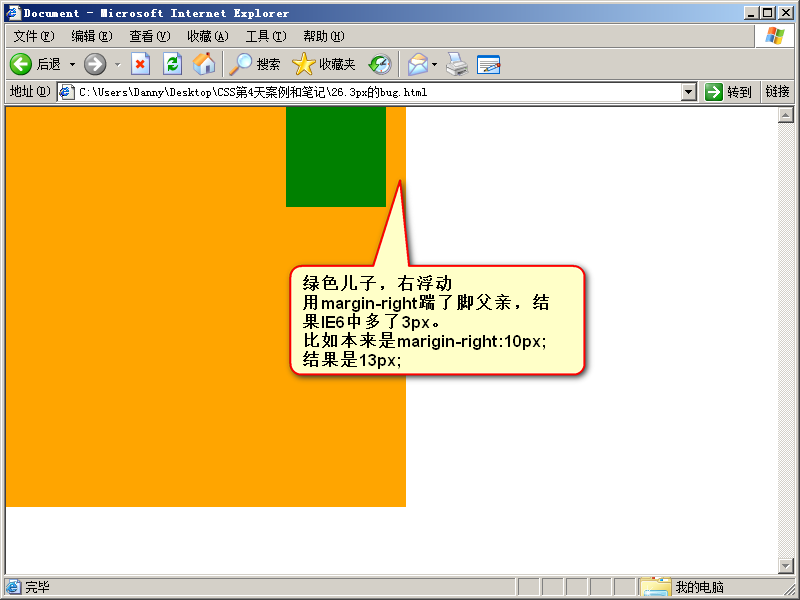
2）使用hack（没必要，别惯着这个IE6）

单独给队首的元素，写一个一半的margin

|  |
| --- |
| 1. <li class="no1"></li> |

|  |
| --- |
| 1. ul li.no1{ 2. \_margin-left:20px; 3. } |

IE6的3px bug



解决办法：

不用管，因为根本就不允许用儿子踹父亲。所以，如果你出现了3px bug，说明你的代码不标准。

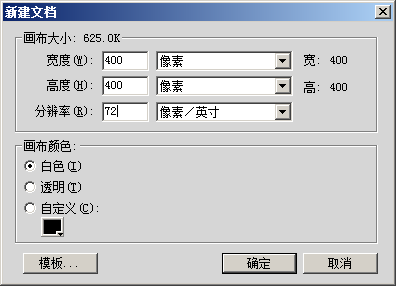
**IE6，千万不要跟他死坑、较劲，它不配。 格调要高，老师给你讲，就是为了增加面试的成功率。不是为了让你成为IE6的专家。**

# 六、Fireworks和精确盒子还原

fireworks是Adobe公司的一个设计软件。功能非常多，我们今天用啥讲啥。

Fireworks的默认文件格式是png。

新建ctrl+N。



分辨率是72像素/英寸 （咱们不用知道，因为设计师把设计图给你）

标尺的快捷键，是ctrl+alt+r。

css中，任何文本都有行高。行高就是

|  |
| --- |
| 1. line-height |

属性。顾名思义，就是行的高度。

首行空两个汉字的格，单位比较奇怪，叫做em，em就是汉字的一个宽度。

|  |
| --- |
| 1. text-indent:2em; |

indent就是“缩进”的意思。