# CSS入门和高级技巧

第7天课堂笔记（本课程共8天）

# 目录

[CSS入门和高级技巧](#_Toc5715)

[目录](#_Toc14912)

[一、浏览器兼容问题](#_Toc23276)

[1.1 现在中国网民用什么浏览器？](#_Toc29822)

[1.2 浏览器的渲染能力](#_Toc4664)

[二、Hack](#_Toc8747)

[2.1 HTML Hack](#_Toc32155)

[2.2 CSS 值Hack](#_Toc13661)

[2.3 CSS 选择器Hack](#_Toc11284)

[三、IE6的问题](#_Toc14699)

[3.1 选择器的兼容问题](#_Toc6819)

[3.2 盒模型的兼容性问题](#_Toc563)

[3.3 浮动的兼容性问题](#_Toc29961)

[3.4 定位的兼容问题](#_Toc32360)

[3.5 文字样式的兼容问题](#_Toc24920)

[四、透明](#_Toc3412)

[4.1 盒子的透明](#_Toc21802)

[4.2 图片的透明](#_Toc31674)

[4.2.1 jpg/jpeg：](#_Toc10623)

[4.2.2 png](#_Toc10430)

[4.2.3 gif](#_Toc2701)

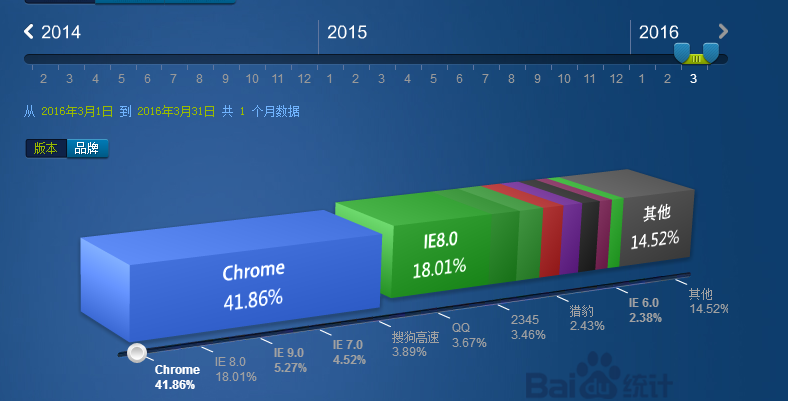
[4.2.4 bmp](#_Toc6895)

# 一、浏览器兼容问题

## 1.1 现在中国网民用什么浏览器？

中国流量最大的网站就是百度，百度在统计着每一个访问者的浏览器、地域、操作系统、分辨率等等信息。

百度流量研究院：http://tongji.baidu.com/data/



Chrome 41.86% 谷歌浏览器，内核名称webkit

IE8 18% win7内置的浏览器

IE9 5% win8内置的浏览器

IE6 2% winxp内置的浏览器

其他 14.52% 火狐、safari、网景、欧朋浏览器等等

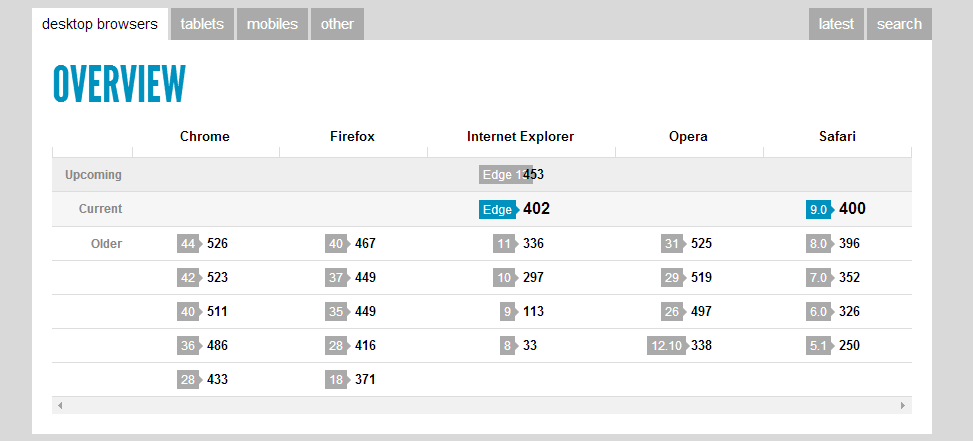
Chrome浏览器的是免费的、开源的；360极速浏览器、百度浏览器、QQ浏览器都是使用的Chrome的内核，这个内核的名字叫做webkit内核。使用Chrome内核的浏览器，超过50%。

## 1.2 浏览器的渲染能力

最好的判断一个浏览器能力的测试，就是HTML5的支持测试。

[www.html5test.com](http://www.html5test.com)

满分555分，是所有HTML5、CSS3的新特性，支持就得1分，不支持不得分。



一切的罪恶的起源，就是IE6、7、8、9、10分数太低，对HTML5、CSS3的支持太差。

但是没有办法，中国人就是喜欢守旧、并且网银仅对IE支持，所以很多国人不升级浏览器。

哆啦A梦测试：这个机器猫的所有部件都是CSS3写的：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Chrome | 火狐 | IE9 | IE8 | IE6 |

IEtester可以模拟IE6、7



所有搞终端开发的，比如web前端开发、ios、安卓开发都有兼容性问题。比如iOS9有很多性特性，但是就是有很多屌人死活就用iOS6，死活不升级。 搞服务端开发的，都没有兼容问题。

# 二、Hack

我们写的是同一份HTML、CSS代码，要分发给不同的用户。用户使用不同版本的浏览器来进行渲染，而我们写的是同一份代码。

所以，我们就急需一种功能，就是写的确实是一个代码，但是代码中的某个部分，是某个、某些特定的浏览器才能解析，其他浏览器不解析的，这种技术就是Hack（黑客）。**Hack就是针对不同的浏览器去写不同的HTML、CSS样式，从而让各浏览器能达到一致的渲染效果。**

Hack分为两大类：HTML hack、CSS hack。

## 2.1 HTML Hack

写一段html代码，这段html代码只能在IE6、7中运行，IE8不运行。

|  |
| --- |
| 1. **<!--[if lte IE 8]>** 2. 自定HTML内容 3. **<![endif]-->** |

上面的这个壳子很神奇，IE6、7、8会渲染里面的内容；而IE9、IE10、Chrome等等浏览器都会认为这些代码是注释。

所以，我们可以用这个方法，单独的为IE浏览器写特殊的HTML标签。

if就是如果，lte表示less than or equal小于等于，endif表示结束如果。要求背诵！

感谢微软，浏览器比较烂，但是给我们留了Hack的后门，自知之明。

如果版本小于等于IE8，那么识别，否则当做注释了：

|  |
| --- |
| 1. <!--[if **lte** IE 8]> 2. <![endif]--> |

如果版本小于IE8，那么识别，否则当做注释了：

|  |
| --- |
| 1. <!--[if **lt** IE 8]> 2. <![endif]--> |

如果版本大于IE8，那么识别，否则当做注释了：

|  |
| --- |
| 1. <!--[if **gt** IE 8]> 2. <![endif]--> |

如果版本大于等于IE8

|  |
| --- |
| 1. <!--[if **gte** IE 8]> 2. <![endif]--> |

如果版本是IE6：

|  |
| --- |
| 1. <!--[if **IE** 6]> 2. <![endif]--> |

用途，比如让IE6、7、8运行一段JS代码，而其他浏览器不运行：

|  |
| --- |
| 1. **<!--[if lte IE 8]>** 2. <script type="text/javascript" src="1.js"></script> 3. **<![endif]-->** |

特别注意，这个Hack不能写在CSS里：

错误的写法：

|  |
| --- |
| 1. <style type="text/css"> 2. <!--[if lte IE 8]> 3. h1{ 4. color:red; 5. background-color: yellow; 6. } 7. <![endif]--> 8. </style> |

## 2.2 CSS 值Hack

CSS HACK又分为两类：值Hack 和 选择器Hack

先来学习值Hack：

**IE6专用的“hack符”就是短横、下划线。**

|  |
| --- |
| 1. div{ 2. width: 200px; 3. height: 200px; 4. background: red; 5. **\_**background: blue; 6. } |

那么高级浏览器不识别\_background:blue;这行语句的，视为这是unknown property name;

IE6自己，会渲染为蓝色。

短横和下划线都行，

|  |
| --- |
| 1. \_background:blue; |

等价于：

|  |
| --- |
| 1. -background:blue; |

**如果想同时调教IE6、7：**

**hack符可以是任意一个： ! $ & \* ( ) = % + @ , . / ` [ ] # ~ ? : < > | \*/**

比如：

|  |
| --- |
| 1. <style type="text/css"> 2. div{ 3. width: 200px; 4. height: 200px; 5. background: red; 6. **!**background:green; 7. } 8. </style> |

高级浏览器、IE8、IE9、IE10渲染为红色，而IE6、7渲染为绿色。

如果想调教IE8和9，hack符写在后面，分号前：

|  |
| --- |
| 1. background-color: blue**\0/**; |

那么只有IE8、9会认识这行语句。

如果想调教IE 6、7、8、9、10 ，那么hack符写在后面，分号前：

|  |
| --- |
| 1. background:red**\9**; |

选择符远不止这些，但是常见就是上面这些，更多的可以自己百度。

## 2.3 CSS 选择器Hack

选择器的hack不怎么常用

如果想单独IE6：

|  |
| --- |
| 1. <style type="text/css"> 2. **\* html** div{ 3. width: 200px; 4. height: 200px; 5. background-color: red; 6. } 7. </style> |

等价于：

|  |
| --- |
| 1. div{ 2. **\_**width: 200px; 3. **\_**height: 200px; 4. **\_**background-color: red; 5. } |

IE6和IE7：

|  |
| --- |
| 1. div**,**{ 2. width: 200px; 3. height: 200px; 4. background-color: red; 5. } |

等价于：

|  |
| --- |
| 1. div{ 2. **!**width: 200px; 3. **!**height: 200px; 4. **!**background-color: red; 5. } |

除了IE6：

|  |
| --- |
| 1. **html>body** div{ 2. width: 200px; 3. height: 200px; 4. background-color: red; 5. } |

如果有兴趣，可以自行百度，工作中用不着这么多。

# 三、IE6的问题

## 3.1 选择器的兼容问题

IE6不支持连续类的交集选择器：

|  |
| --- |
| 1. <style type="text/css"> 2. .haha.cur{ 3. color:red; 4. } 5. </style> |

怎么解决？很简单，就不要这么写就行了，改成标签开头：

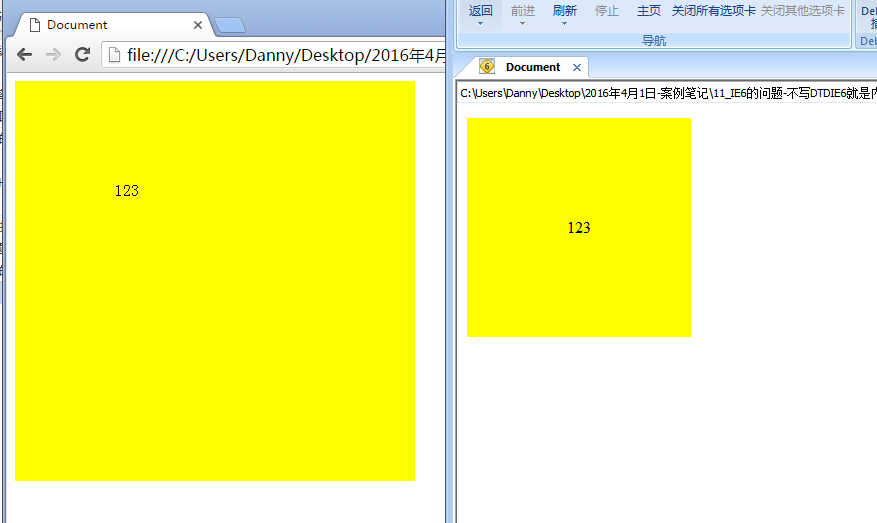
|  |
| --- |
| 1. li.cur{ 2. } |

其他我们学习的7种选择器，IE6支持良好，权重计算良好。

p #id .class div p div.haha div,p \*

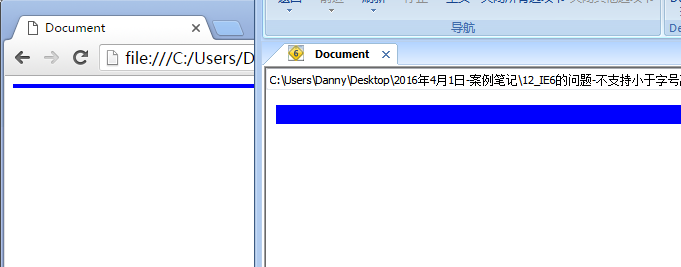
## 3.2 盒模型的兼容性问题

如果不写DTD，那么IE6的盒模型就是内减的，而不是外扩的：



解决办法就是写好DTD，别不写！

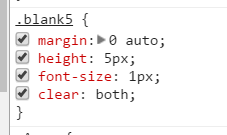
不支持小于一个文字高度的盒子



任何浏览器都有默认字号，IE6的默认字号是14px，所有小于字号的盒子都不能正常渲染，高度不能小于字号。

解决办法很简单，就是用\_单独给IE6设置一个盒子的字号，字号比height小就行了，一般来说是0：

|  |
| --- |
| 1. height: 4px; 2. **\_**font-size: 0; |



## 3.3 浮动的兼容性问题

标准流的盒子不往里面钻

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

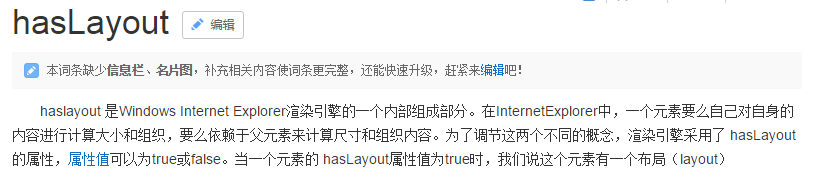
**解决办法：我们根本就不允许大家，用浮动来制作压盖；应该用定位来制作压盖**。

overflow:hidden;小偏方不支持的

父亲里面有脱标的儿子，所以就不能被脱标的儿子撑高，就要写overflow:hidden;这个小偏方。

IE6不领情。解决办法就是触发IE6的haslayout机制。

|  |
| --- |
| 1. div{ 2. border: 10px solid red; 3. overflow: hidden; 4. **\_zoom:1;** 5. } |



翻译成人话：

IE6里面有两个儿子撑父亲的机制，一种是有hasLayout，一种是没有。所谓的layout就是布局的意思。

zoom 总是可以触发 hasLayout。zoom这个属性，是用来控制一个元素缩放倍数的，是IE特有的属性，Chrome现在的版本也支持了。加上zoom属性的元素，都能触发这个元素的hasLayout，所以IE6就用另一种方式渲染盒子了。

也就是说，IE6你要敏感，比如遇见了一个height很小的盒子：

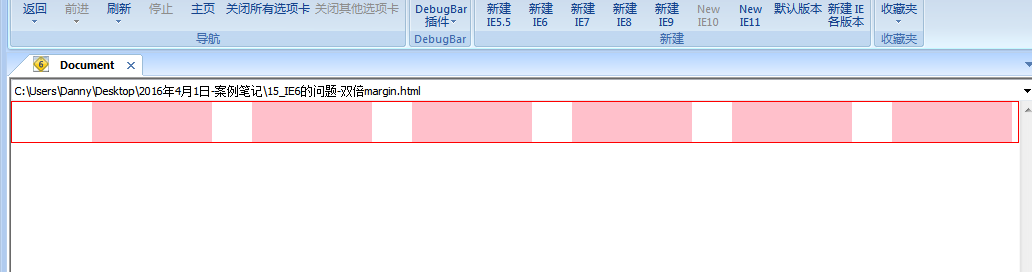
|  |
| --- |
| 1. height:5px; 2. **\_font-size:0;** |

比如你现在父亲要得到脱标儿子的宽度：

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

双倍margin

连续多个元素浮动，浮动的方向和margin的方向相同，最后一个、最开头的一个都有可能出现双倍的margin。



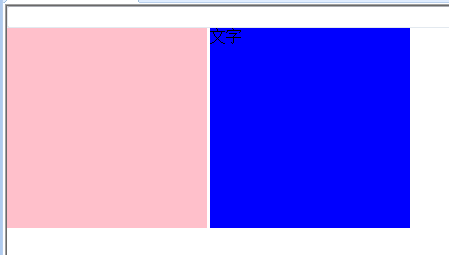
解决办法是：首先，我们严禁用儿子踹父亲，所以这个文字根本不能出现，因为你用儿子踹父亲了。一定要记住，如果父子之间有空隙，一定要用父亲的padding，不要用儿子的margin！

其次，如果你非要踹父亲，就要给第一个元素小一倍的margin：

|  |
| --- |
| 1. <ul> 2. <li class="first"></li> 3. <li></li> 4. <li></li> 5. <li></li> 6. <li></li> 7. <li></li> 8. </ul> |

|  |
| --- |
| 1. ul li{ 2. float: left; 3. width: 120px; 4. height: 40px; 5. background-color: pink; 6. **margin-left: 60px;** 7. } 8. ul li.first{ 9. **\_margin-left:30px;** 10. } |

3像素bug



左边粉色盒子浮动了，蓝色盒子没有浮动，那么之间就有3px的空隙。

解决办法：首先，压根就不能出现一个浮动，另一个不浮动的情况。因为这种情况，在高级浏览器中，就已经蓝色盒子就已经往下钻了，就已经产生了更大层次的不兼容。

如果要解决，就是左边盒子的

|  |
| --- |
| 1. \_margin-right:-3px; |

## 3.4 定位的兼容问题

IE6不支持

|  |
| --- |
| 1. position:fixed; |

怎么解决？用js来模拟固定定位。JS课程再说

## 3.5 文字样式的兼容问题

IE6、7、8中，**所有有超级链接的图片**，都默认有一个蓝色边框：



解决办法：

|  |
| --- |
| 1. a img{ 2. border:none; 3. } |

或者干脆：

|  |
| --- |
| 1. img{ 2. border:none; 3. } |

# 四、透明

[opacity?](https://developer.mozilla.org/zh-CN/docs/Web/CSS/opacity) [Opacity?IE](https://msdn.microsoft.com/en-us/library/ms532910(VS.85).aspx)

## 4.1 盒子的透明

|  |
| --- |
| 1. opacity: 0.40; |

opacity就是透明度的意思，能够背下来这个单词。值是0~1；1就是实心，0就是纯透明。

IE6~8不支持。要写IE自己的属性，自己的滤镜的属性：

|  |
| --- |
| 1. filter:alpha(opacity=40); |

要注意的是，里面的文字也一起有透明了。解决办法就是不要把有opacity属性的盒子里面写文字。把文字单独放出去，用绝对定位给定位到一起：

|  |
| --- |
| 1. <div class="box"> 2. <img src="images/1.jpg" alt="" /> 3. <div class="title"></div> → 半透明黑色 4. <span>文字文字文字文字</span> → 白色文字，给定位到一起。 5. </div> |

晚自习自己练习17号案例。

也就是说，盒子透明有兼容性问题，但是能够轻松解决，所以等于没有兼容性问题。

## 4.2 图片的透明

网页中的图片格式：

### 4.2.1 jpg/jpeg：

**压缩格式，是颜色失真的**，为了保存尺寸小，所以有压缩算法，所以是颜色失真的。网页中的照片、新闻图片、banner、焦点图都要用jpg的格式，因为这样保存的尺寸就小。没有图层。**不支持透明和半透明**。

在导出之前，用导出预览调整一下“质量”。

### 4.2.2 png

**不可压缩，颜色不失真**，是fireworks这个软件的默认保存格式，**可以有图层**。

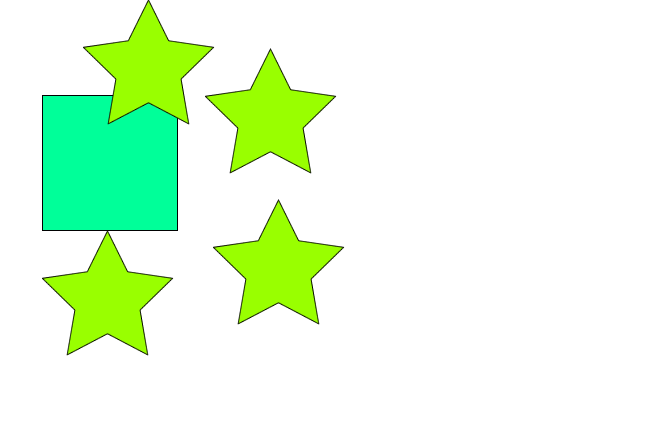
在传到服务器上面的时候，所有的png图片，**一定要记住去掉所有图层**！去掉的方法很简单，就是“文件导出”。

我们说jpg是压缩的，png是不压缩的。但是，同一个图，jpg不一定比png尺寸小，

我们经过测试，发现这样的图片jpg的尺寸更小：



但是这样的图片png的反而更小：



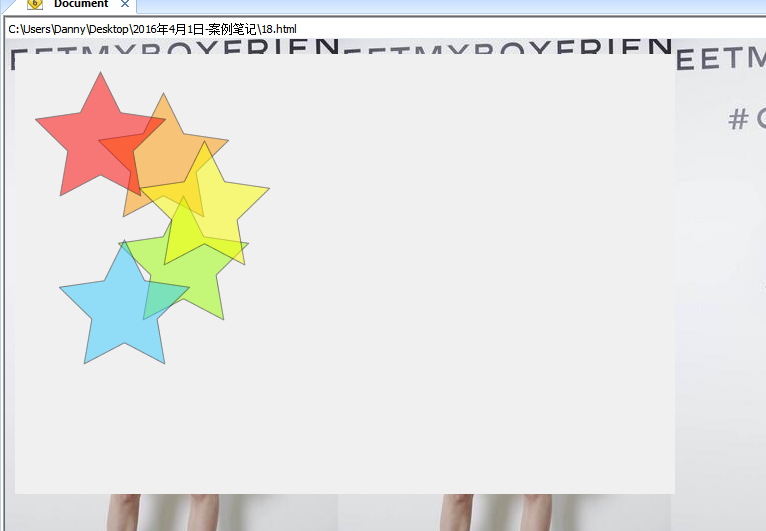
**所以，网页上的杂碎图标，都要存为png，尺寸更小！**

**png支持透明和半透明。**



IE6不支持png格式的透明和半透明。解决办法明天说。

IE6中：

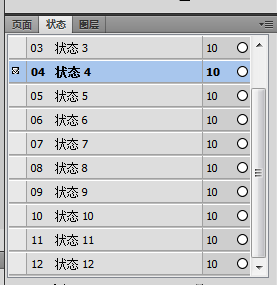


### 4.2.3 gif

gif不是压缩不压缩的问题，它就支持固定数量的颜色，可以是256种，可以是128种，可以是64种……2种。

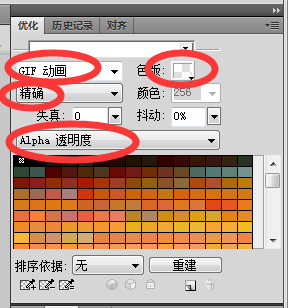
所以是**严重颜色失真**！根本表示不全自然界的所有颜色！

**支持动画！**



**尺寸比较小**，因为颜色数量小，如果不动的图，那么尺寸更小！但是为什么不用gif？因为会严重失真！

**gif也支持透明**，fireworks设置比较复杂:



|  |  |
| --- | --- |
| png：  png支持透明，也支持半透明 | **gif：**  **支持透明，不支持半透明** |

**gif在IE6中的显示状态是没有任何兼容问题的，都是支持透明，不支持半透明。**

所以，工作中，如果要做一个透明的元素，可以用gif，而不用png。因为pngIE6不兼容。

当然了，如果你们公司不考虑IE6，就全用png就行了。

### 4.2.4 bmp

是windows画图的保存格式，不压缩的，不失真的，不能动，不能有透明、不能有半透明、没有图层。