请在主流浏览器中打开测试页面，在Fiddler里查看http请求。

**1. 隐藏图片**

<img src="1.jpg" style="display: none" />

测试：[test\_1.html](http://lifesinger.org/lab/2009/imghttp/test_1.html)  
结论：只有Opera不产生请求。  
注意：用visibility: hidden隐藏图片时，在Opera下也会产生请求。

**2. 重复图片**

<img src="1.jpg" />

<img src="1.jpg" />

测试：[test\_2.html](http://lifesinger.org/lab/2009/imghttp/test_2.html)  
结论：所有浏览器都只产生一次请求。

**3. 重复背景**

<style type="text/css">

.test1 { background: url(1.jpg) }

.test2 { background: url(1.jpg) }

</style>

<div class="test1">test1</div>

<div class="test2">test2</div>

测试：[test\_3.html](http://lifesinger.org/lab/2009/imghttp/test_3.html)  
结论：所有浏览器都只产生一次请求。

**4. 不存在的元素的背景**

<style type="text/css">

.test1 { background: url(1.jpg) }

.test2 { background: url(2.jpg) } /\* 页面中没有class为test2的元素 \*/

</style>

<div class="test1">test1</div>

测试：[test\_4.html](http://lifesinger.org/lab/2009/imghttp/test_4.html)  
结论：背景仅在应用的元素在页面中存在时，才会产生请求。这对CSS框架来说，很有意义。

**5. 隐藏元素的背景**

<style type="text/css">

.test1 { background: url(1.jpg); display: none; }

.test2 { background: url(2.jpg); visibility: hidden; }

</style>

<div class="test1">test1</div>

测试：[test\_5.html](http://lifesinger.org/lab/2009/imghttp/test_5.html)  
结论：Opera和Firefox对于用display: none隐藏的元素背景，不会产生HTTP请求。仅当这些元素非display: none时，才会请求背景图片。

**6. 多重背景**

<style type="text/css">

.test1 { background: url(1.jpg); }

.test1 { background: url(2.jpg); }

</style>

<div class="test1">test1</div>

测试：[test\_6.html](http://lifesinger.org/lab/2009/imghttp/test_6.html)  
结论：除了基于webkit渲染引擎的Safari和Chrome，其它浏览器只会请求一张背景图。  
注意：webkit引擎浏览器对背景图都请求，是因为支持CSS3中的[多背景图](http://www.w3.org/TR/css3-background/#the-background-image)。

**7. hover的背景加载**

<style type="text/css">

a.test1 { background: url(1.jpg); }

a.test1:hover { background: url(2.jpg); }

</style>

<a href="#" class="test1">test1</a>

测试：[test\_7.html](http://lifesinger.org/lab/2009/imghttp/test_7.html)  
结论：触发hover时，才会请求hover状态下的背景。这会造成闪烁，因此经常放在同一张背景图里通过翻转来实现。  
  
2009-05-13晚补充：上面的解释有误，更详细的解释请参考[续篇](http://lifesinger.org/blog/?p=1681)。翻转技巧指的是Sprite技术，例子：[test\_7b.html](http://lifesinger.org/lab/2009/imghttp/test_7b.html), 在ie6下不会产生闪烁。

**8. JS里innerHTML中的图片**

<script type="text/javascript">

var el = document.createElement('div');

el.innerHTML = '<img src="1.jpg" />';

//document.body.appendChild(el);

</script>

测试：[test\_8.html](http://lifesinger.org/lab/2009/imghttp/test_8.html)  
结论：只有Opera不会马上请求图片。  
注意：当添加到DOM树上时，Opera才会发送请求。

**9. 图片预加载**

最常用的是JS方案：

<script type="text/javascript">

new Image().src = '1.jpg';

new Image().src = '2.jpg';

</script>

在无JS支持的环境下，可以采用隐藏元素来预加载：

<img src="1.jpg" style="visibility: hidden; height: 0; width: 0" />

测试：[test\_9.html](http://lifesinger.org/lab/2009/imghttp/test_9.html)

**终于到总结了**

1. 对于隐藏图片和隐藏元素的背景，Opera不会产生请求。
2. 对于隐藏元素的背景，Firefox也不会产生请求。
3. 对于尚未插入DOM树的img元素，Opera不会产生请求。
4. 基于webkit引擎的Safari和Chrome，支持多背景图。
5. 其它情景，所有主流浏览器保持一致。

对图片请求的处理上，个人觉得Opera走在前列。

**番外**

1. 用Fiddler监控Opera时，如果是本地服务器，需要在Opera的代理服务器设置里，将本地服务器勾选上。

2. 查看HTTP请求数，还有一个万无一失的方法是，直接查看Apache的access.log文件。

3. 我的Firefox对于重复图片和重复背景，会产生重复请求。禁用了所有扩展，问题依旧。有知详情者，还望告知。

**10. IE6下的背景闪烁bug**

这个bug久闻其名，但我自己从来没遇到过。纳闷了许久，后来才发现，这是一个非典型性甚至可以忽略的bug. 因为它的触发条件离普通用户很远：

The cause of flickering is choosing “Every visit to page” in Tools » Internet Options… » Temporary Internet Files » Settings…

Selecting anything other than “Every visit to page” will fix all flickering. Fortunately, the means that the flicker problem plagues developers far more than your common user.

只有IE的缓存选项设置为“Every visit to page”时，才会触发。普通用户的默认设置是“Automatically”, 不会遇到此问题。因此继续往下阅读之前，请确保IE6的缓存设置为“Every visit to page”, 否则你将看不到神奇的现象。

测试：[test\_11.html](http://lifesinger.org/lab/2009/imghttp/test_11.html)

<style type="text/css">

a { background: #f00 url(square.gif) }

</style>

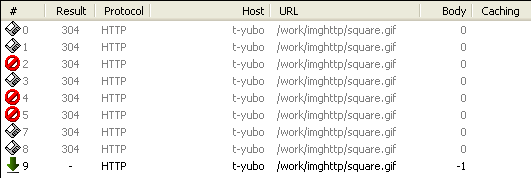
<a href="#">test link</a>

上面的代码平淡无奇，但在IE6下，当鼠标划过test link时，背景会瞬间变红。如果你看不到此现象，请用老虎钳将网线剪掉一半，降低网速后再测试。

除了给链接A设置背景色，还有一些其它触发条件，详细资料请参阅：[Minimize Flickering CSS Background Images in IE6](http://www.fivesevensix.com/studies/ie6flicker/).

注意：只有IE6下，设置“Every visit to page”, 并且给元素A设置了背景色等条件下，才会触发此bug. 上面引用的文章中提到三种解决办法：1. 调整服务器的配置，2. 双缓冲背景，3. js解决办法：

try {document.execCommand("BackgroundImageCache", false, true);}catch(e){};

到此，似乎已经尘埃落定。但是，好奇虽然害死猫，却也能发现猫世界里的更多秘密：  
  
上图是用Fiddler监控鼠标在链接A上移进移出时的HTTP请求。虽然请求的结果是304（Not modified），但从发送请求到服务器返回304，依旧是需要时间的。在这个时间差里，链接A的背景是红色的，而不是图片。**多余的HTTP请求才是背景闪烁的真正原因**。

明白了这一点，可以看出“双缓冲背景”法，是治标不治本，是视觉上的小hack，并没有减少无谓的HTTP请求。采用调整服务器配置或execCommand hack，才能彻底杜绝多余的HTTP请求。

微软很委婉在一个hotfix中提到这个bug：[Available memory decreases when you view a Web page in Internet Explorer 6 Service Pack 1](http://support.microsoft.com/default.aspx?scid=kb%3Ben-us%3B823727). 当script中频繁改变一个设置了背景图的按钮的背景色时，会造成IE内存泄漏。微软的文档太文档了，Resolution那一段，我反复读了好几次，才明白微软要表达什么意思，囧。

鉴于目前很多公司的图片服务器配置里已经自动fix了此bug，以及IE6的逐步淡出，还有触发条件离普通用户较远，我们可以逐步忘掉此bug了，就像我们逐步淡忘掉ie 5.5的很多css hack一样。

休息一下，上面只是引子，精彩内容在下面。

**11. 切换class时，背景图的请求**

链接的hover状态切换, 是class切换一个特例。更一般性的测试如下：

<style type="text/css">

.test1 { background: url(1.jpg) }

.test2 { background: url(2.jpg) }

</style>

<div class="test1" id="J\_Test">test</div>

<a href="#" id="J\_Link">切换className</a>

<script type="text/javascript">

document.getElementById('J\_Link').onclick = function() {

var el = document.getElementById('J\_Test');

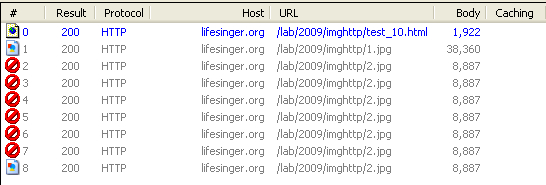
el.className = (el.className == 'test1') ? 'test2' : 'test1';

return true;

};

</script>

测试：[test\_10.html](http://lifesinger.org/lab/2009/imghttp/test_10.html)

结论：当快速点击“切换className”时，除了IE，其它浏览器都只会产生两次请求。但在IE（包括IE8）中，用Fiddler能捕捉到如下请求：  
  
带红色停止标志的，是Aborted Session. 也就是说，在IE下，当一张图片未下载完成时，如果class就已经切换，正在下载的图片会被中止掉。被中止掉的图片，下次用到时，会再次发送请求。直到图片成功下载后，切换class时才不会再次请求图片。

注意：如果你的观察结果和上面的结论有异，请检查 Internet Options… » Temporary Internet Files » Settings… 没有选中”Every visit to page”, 以及Fiddler中没有禁用Cache. 否则图片下载完成后，依旧会重新发送请求。如果观察不到请求，请清空IE缓存后再测试。另外，一定要记得用老虎钳剪掉一半网线。

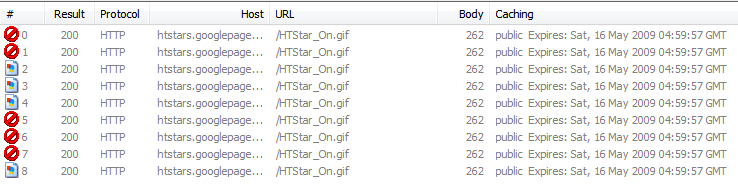
可以看出，**即便在IE8下，当网速较慢时，快速切换某个元素的class，依旧会造成闪烁**。要一直到背景图片下载完成后，才不会闪烁。

知道了问题所在，解决办法就很容易想到了：采用Sprite技术，切换class时只改变background-position.  
测试：[test\_10b.html](http://lifesinger.org/lab/2009/imghttp/test_10.html)  
注意：用Sprite合成一张图片后，你点得再快，IE也始终只发送一个图片请求。

很想结束本文了，无奈IE下还有一个更诡异的闪烁现象… Come on~

**12. 神秘的闪烁**

直奔测试页面：[test\_12.html](http://lifesinger.org/lab/2009/imghttp/test_12.html)

第一行星星，当鼠标每次悬浮上去，都会产生一个图片请求：  


原来老怀疑JS写法有问题，琢磨了许久，才注意到上图中Caching栏的值：public Expires: Sat, May 2009 04:59:57 GMT. 超级汗googlepages的设置。

再来看第二行星星，悬浮时，没有第一行星星的问题。从Fiddler中可以看出，第二行星星的背景文件也是过期的，为什么却不重新下载呢？

相信你已经留意到测试页面左下角那颗孤零零的星星，这就是第二行星星正常的救星。

根据上面的现象，我们可以猜测IE请求图片的虚拟过程为：

1. 需求：改变classA, 需要背景图imgA.
2. 在页面中寻找是否已经有元素使用imgA, 有的话，直接共用。
3. 在缓存中找到了imgA，判断Expire头，如果过期了，则重新发送请求，全新下载。
4. 如果Expire头没过期，直接复用。
5. 在缓存中没找到，发送请求，从服务上下载。

其中第3点，IE的处理方式比其它浏览器严格，其它浏览器在刚才的测试页面中，不会重新下载。

另外，前面已经总结过，当切换到classA，开始下载imgA后，当classA又没用了时，IE会中断imgA的下载，而其它浏览器则会一直下载完。从逻辑上讲，IE的处理方式很严谨。但实际体验上，IE的处理方法很恼火。好与坏，有时真难评说。

**后记**

上面的结论，都仅仅是“定律”，建立在黑盒测试的基础上。可能会有不少错误之处，如果发现了，还望各位能及时反馈给我。