

# 各浏览器对CSS错误解析规则的差异及CSS hack

标签：\_ , \* , -moz-... , -webkit-... , ! important, hack, 兼容性, CSS, 语法

操作系统版本：

Windows 7 Ultimate build 7600

浏览器版本：

IE5

IE5.5

IE6

IE7

IE8

Firefox 3.6.2

Safari 4.0.4

Chrome 5.0.356.2 dev

Opera 10.51

受影响的浏览器：

所有浏览器

各浏览器对CSS错误解析规则的差异及CSS hack.....	1
一、CSS语法、错误处理及特性值的介绍 .....	2
1. CSS语法 .....	2
2. CSS中的错误解析规则 .....	2
3. 值 .....	2
4. 浏览器解析差异及CSS hack.....	2
二、浏览器对CSS错误解析规则的差异 .....	2
1. 畸形的声明 .....	2
(1). 多余的右大括号.....	2
(2). 未闭合的左大括号 .....	3
(3). 多余的左括号 .....	4
(4). 属性和值由等号连接 .....	5
(5). 空的特性值 .....	5
(6). 选择器与“{”之间出现多余“,” .....	6
(7). 选择器之前出现分号 .....	6
2. 不合法的属性值.....	7
(1). 属性值对之间没有分号 .....	7
(2). 使用!important，且声明之间缺失分号 .....	7
(3). 长度值缺失单位.....	8
3. 其他 .....	8
(1). HTML标签style属性内出现多余大括号.....	8
(2). 注释前出现多余字符 .....	9
三、CSS hack .....	9
1. CSS hack简介 .....	9
2. CSS hack的实现方式 .....	10
3. Windows系统下CSS hack汇总表 .....	10
(1). 利用浏览器对相同代码的支持差异实现的hack .....	10
(2). 以Firefox或Webkit特有的扩展样式实现的hack .....	13
(3). 利用IE对标准的支持缺陷写的hack .....	16
(4). 以IE特有的条件注释为基础的hack .....	16

## 一、CSS语法、错误处理及特性值的介绍

### 1. CSS语法

CSS语法适用于任何版本的CSS，它描述了CSS的核心句法(syntax)、关键字、厂商扩展、可用字符集、规则集合、声明块、选择器、特性以及注释等所有CSS的构成部分。

在此不做累述，详细参见：W3C CSS2.1规范 [4.1](#)。

### 2. CSS中的错误解析规则

CSS语法规定了CSS的写法，但是开发人员还是可能写出不合CSS语法的代码，这时候，浏览器就需要忽略一部分不合法的样式表。

CSS2.1及所有后继版本中，对于任何以破折号、下划线开头的property:value组合和不包含标识符的@-keywords组合，都以忽略的方式处理。

为了保证新的属性和值可以被正确添加，但遇到以下情况时，浏览器必须遵循以下的规则：

1. 未知的属性。浏览器必须忽略带有位置属性的声明。如，`p{yes:'good'}`
2. 不合法的值。浏览器必须忽略带有不合法值的声明。如，`p{height:20}`
3. 畸形的声明。当浏览器解析一个声明时，读取它的代码直到这个声明的结束，同时，检查 `()`, `[]`, `{}`, `"`, 和 `'` 的匹配规则，并且正确的处理编码，这时候，浏览器必须处理它所遇到的意外出现的标记。 如，`p{height}`或 `({}P{width:100px}`
4. 不可用的@关键字。如：`@hello{...}`
5. 样式表的意外结束。浏览器必须自动闭合敞开的结构(如，块，字符串和注解等)如`@media screen { p:before { content: 'Hello`
6. 字符串的意外结束。如，`p { color: green; font-family: 'Courier New Times color: red; color: green; }`

详细说明，请见W3C CSS2.1规范 [4.2](#)

### 3. 值

CSS中的值，主要是指属性的值。

包括，数字、长度、颜色、字符串、百分比、URL和URI等。

详细参见：W3C CSS2.1规范 [4.3](#)

### 4. 浏览器解析差异及CSS hack

因为各大厂商对标准的实现不尽相同，所以，不同浏览器对CSS代码的解析标准不同。

因此，对相同的CSS代码，各浏览器的解析可能会有差异。

而在此差异的基础上，写出的只有个别浏览器或某些浏览器识别的CSS代码，就形成了CSS hack。

## 二、浏览器对CSS错误解析规则的差异

### 1. 畸形的声明

#### (1). 多余的右大括号

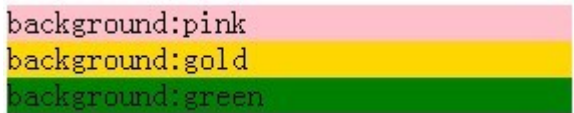

测试用例：

```

<style>
.pink {background:pink;} }
.gold {background:gold;}
.green {background:green;}
</style>
<div class="pink">background:pink</div>
<div class="gold">background:gold</div>
<div class="green">background:green</div>

```

各浏览器截图：

IE5/IE5.5/ IE6(Q)(S)/IE7(Q)(S)/IE8(Q)	IE8(S)/Firefox/Chrome/Safari/Opera
<pre> .pink {background:pink;} .gold {background:gold;} .green {background:green;} </pre>	<pre> .pink {background:pink;} }.gold {background:gold;} .green {background:green;} </pre>
	

- 在IE5/IE5.5/IE6(Q)(S)/IE7(Q)(S)/IE8(Q)下，浏览器直接忽略了多余的右大括号"}"，错误的声明没有影响到`.gold`的解释，DIV[class="gold"]的金色背景色样式被正确渲染；
- 在IE8(S)/Firefox/Chrome/Safari/Opera(S)(Q)下，浏览器将多余的右大括号"}"与其下一条声明`.gold`合并，`.gold`变为`}.gold`，导致选择器无法匹配到DIV[class="gold"]而没有被渲染上背景色。

## (2). 未闭合的左大括号

测试用例：

```

<style>
.pink {background:pink; /*大括号没有闭合*/
.gold {background:gold;}
.gray {background:gray;}
.cyan {background:cyan;}
</style>
<div class="pink">background:pink</div>
<div class="gold">background:gold</div>
<div class="gray">background:gray</div>
<div class="cyan">background:cyan</div>

```

各浏览器截图：

IE5/IE5.5/IE6(Q)/IE7(Q)/IE8(Q)	IE6(S)/IE7(S)/IE8(S)/Firefox/ Chrome/Safari/Opera
<pre> .pink{background:pink; .gold {background:gold;} .gray {background:gray;} .cyan {background:cyan;} ↓ .pink{background:pink;} </pre>	<pre> .pink {background:pink; .gold {background:gold;} .gray {background:gray;} .cyan {background:cyan;} ↓ .pink {background:pink;} </pre>

<pre>.gray {background:gray;} .cyan {background:cyan;}</pre>	
<pre>background:pink background:gold background:gray background:cyan</pre>	<pre>background:pink background:gold background:gray background:cyan</pre>

- 在IE5/IE5.5/IE6(Q)/IE7(Q)/IE8(Q)下，未闭合的左大括号把下一个声明的右大括号当作了闭合括号，而**.gold** 被当作错误的属性忽略。
- 在IE6(S)/IE7(S)/IE8(S)/Firefox/Chrome/Safari/Opera下，未闭合的左大括号把最后一个声明的右大括号当作了闭合括号，而把**.gold**、**.gray**、**.cyan** 当作了错误的属性抛弃。符合W3C标准中对此类问题的处理标准，“读取CSS代码直到这个声明的结束”。

### (3). 多余的左括号

情况1：左括号位于选择器之后

测试用例：

```
<style>
.pink { {background:pink;} /*多余的大括号*/
.gold {background:gold;}
</style>
<div class="pink">background:pink</div>
<div class="gold">background:gold</div>
```

各浏览器截图：

IE5/IE5.5/IE6/IE7/IE8(Q)	IE8(S)/Firefox/Chrome/Safari/Opera
<pre>.pink {background:pink;} .gold {background:gold;}</pre>	<pre>.pink{}</pre>
<pre>background:pink background:gold</pre>	<pre>background:pink background:gold</pre>

- 在IE5/IE5.5/IE6/IE7/IE8(Q)下，浏览器直接忽略了多余的左大括号“{”，错误的声明没有影响到后面CSS的解释，DIV[class="gold"]的金色背景色样式被正确渲染；
- 在IE8(S)/Firefox/Chrome/Safari/Opera下，未闭合的左大括号把最后一个声明的右大括号当作了闭合括号，而把**{background:pink;}**和**.gold**当作了错误的声明抛弃。符合W3C标准中对此类问题的处理标准，“读取CSS代码直到这个声明的结束”。

情况2：左括号位于选择器之前

测试用例：

```
<style>
{.pink {background:pink;}
.gold {background:gold;}
</style>
<div class="pink">background:pink</div>
<div class="gold">background:gold</div>
```

各浏览器截图：

IE5/IE5.5/IE6/IE7(Q)/IE8(Q)	IE7(S)/IE8(S)/Firefox/Chrome/Safari/Opera
<pre>{.pink {background:pink;} .gold {background:gold;}}</pre>	
<pre>background:pink background:gold</pre>	<pre>background:pink background:gold</pre>

- 在IE5/IE5.5/IE6/IE7(Q)/IE8(Q)下，浏览器将多余的左大括号“{”与其后的选择器合并，导致.pink失效，而并没有影响到其后面的其他CSS代码；
- 在IE7(S)/IE8(S)/Firefox/Chrome/Safari/Opera下，未闭合的左大括号把最后一个声明的右大括号当作了闭合括号，是一个没有选择器名称的声明块。符合W3C标准中对此类问题的处理标准，“读取CSS代码直到这个声明的结束”。

#### (4). 属性和值由等号连接

测试用例：

```
<style>
.pink {background=pink; color=blue; font-size=28px;}
</style>
<div class="pink">background:pink</div>
```

各浏览器截图：

IE5/IE5.5/IE6(Q)/IE7(Q)/IE8(Q)	IE6(S)/IE7(S)/IE8(S)/Firefox/Chrome/Safari/Opera
<pre>.pink {background:pink; color:blue; font-size:28px;}</pre>	<pre>.pink {}</pre>
<pre>background:pink</pre>	<pre>background:pink</pre>

- 在IE5/IE5.5/IE6(Q)/IE7(Q)/IE8(Q)下，浏览器直接将错误的等号替换为了正确的冒号，DIV[class="pink"]的CSS声明被正确的解释及渲染；
- 在IE6(S)/IE7(S)/IE8(S)/Firefox/Chrome/Safari/Opera下，浏览器将错误的等号认为是错误的CSS声明而将其忽略，导致“.pink”内的声明为空。

#### (5). 空的特性值

测试用例：

```
<style>
.nullvalue{
background:pink;
background:;
}
</style>
<div class="nullvalue">background:pink</div>
```

各浏览器截图：

IE6/IE7/IE8(Q)	IE8(S)/Firefox/Chrome/Safari/Opera
----------------	------------------------------------

<code>style=""</code>	<code>style="background:pink"</code>
background:pink	background:pink

- 在IE6/IE7/IE8(Q)下，浏览器将没有值的“background:”替换了之前的“background:pink;”，而由于值为空，所以使用默认样式值渲染；
- 在其他浏览器下，浏览器将此空的特性值忽略。

#### (6). 选择器与“{”之间出现多余“,”

测试用例：

```
<style>
  .pink, {background:pink;}
  .gold {background:gold;}
</style>
<div class="pink">background:pink</div>
<div class="gold">background:gold</div>
```

各浏览器截图：

IE6/IE7/IE8(Q)	IE8(S)/Firefox/Chrome/Safari/Opera
.pink { background:pink; } .gold {background:gold; }	.gold {background:gold; }
background:pink background:gold	background:pink background:gold

- 在IE6/IE7/IE8(Q)下，浏览器将选择器后多余的逗号忽略；
- 在其他浏览器下，多余的逗号导致第一个选择器失效。

#### (7). 选择器之前出现分号

测试用例：

```
<style>
  .pink {background:pink;} ;
  .gold {background:gold;}
  .gray {background:gray;}
</style>
<div class="pink">background:pink</div>
<div class="gold">background:gold</div>
<div class="gray">background:gray</div>
```

各浏览器截图：

IE6/IE7/IE8(Q)	IE8(S)/Firefox/Chrome/Safari/Opera
.pink { background:pink; } .gold {background:gold; } .gray { background:gray; }	.pink { background:pink; } ;.gold {background:gold; } .gray { background:gray; }

background:pink background:gold	background:pink background:gold
------------------------------------	------------------------------------

- 在IE6/IE7/IE8(Q)下，多余的分号被忽略；
- 在其他浏览器下，多余的逗号导致其下一个选择器失效。

## 2. 不合法的属性值

### (1). 属性值对之间没有分号

测试用例：

```
<style>
.pink {background:pink color:blue font-size:28px}
</style>
<div class="pink">background:pink</div>
```

各浏览器截图：

IE5/IE5.5/IE6(Q)/IE7(Q)/IE8(Q)	IE6(S)/IE7(S)/IE8(S)/Firefox/Chrome/Safari/Opera
.pink {background:pink color:blue font-size:28px} ↓ .pink {background:pink;}	.pink {background:pink color:blue font-size:28px} ↓ .pink {}
background:pink	background:pink

- 在IE5/IE5.5/IE6(Q)/IE7(Q)/IE8(Q)下，浏览器会正确解释缺失的第一个分号之前的声明 background:pink被正确的解释及渲染；
- 在IE6(S)/IE7(S)/IE8(S)/Firefox/Chrome/Safari/Opera下，浏览器将background属性的值是到一个分号出现为止，所以pink和后面的属性值对被当作了background的值，但是这个值是不合法的，所以被抛弃。

### (2).使用!important，且声明之间缺失分号

测试用例：

```
<style>
.pink {background:pink !important color:blue !important font-size:28px}
</style>
<div class="pink">background:pink</div>
```

这段CSS在各浏览器环境中的解释如下：

IE5	IE5.5 IE6/7 IE8(Q)	IE8(S)	Firefox Chrome Safari Opera
.pink { background:pink	.pink { background:pink !important;	.pink { font-size:28px; }	.pink { }

<code>!important;</code> <code>}</code>	<code>color:blue !important;</code> <code>font-size:28px;</code> <code>}</code>		
--	---	--	--

- 在IE5下，浏览器会正确解释第一个**!important**之前的声明；
- 在IE5.5/IE6(Q)/IE7(Q)/IE8(Q)下，浏览器会正确解释所有声明；
- 在IE8(S)下，浏览器会正确解释最后一个声明；
- 在Firefox(SQ)/Chrome(SQ)/Safari(SQ)/Opera(SQ)下，浏览器忽略所有错误的声明。

### (3). 长度值缺失单位

测试用例：

```
<style>
.pink {background:pink; font-size:28; height:50;}
</style>
<div class="pink">background:pink</div>
```

这段CSS在各浏览器环境中的解释如下：

IE5/IE5.5/ IE6(Q)/IE7(Q)/IE8(Q), Firefox(Q)/Chrome(Q)/Safari(Q)/Opera(Q)	IE6(S)/IE7(S)/IE8(S), Firefox(S)/Chrome(S)/Safari(S)/Opera(S)
.pink {background:pink; font-size:28px; height:50px;}	.pink {background:pink;}
background:pink	background:pink

- 在IE5/IE5.5/其他各浏览器的混杂模式(Q)下，浏览器会为缺失px的属性值添加“px”以正确解释其含义；
- 在所有浏览器的标准模式(S)下，浏览器会忽略缺失px的属性值。

## 3. 其他

### (1). HTML标签style属性内出现多余大括号

测试用例：

```
<div style="{background:pink;}">background:pink</div>
```

这段CSS在各浏览器环境中的解释如下：

IE6/IE7/IE8(Q)/Firefox(Q)/Opera(Q)	IE8(S), Firefox(S)/Chrome/Safari/Opera(S)
style="background:pink"	.pink {}
background:pink	background:pink

- 在IE5/IE5.5/IE6/IE7/IE8(Q)/Firefox(Q)/Opera(Q)下，浏览器会忽略style属性中最外层多余的一对大括号；
- 在其他浏览器下，浏览器将此属性值当做错误值处理。



## (2). 注释前出现多余字符

测试用例：

```
<style>
  ABC123
  <!--
  .pink {background:pink;}
  .gold {background:gold;}
  -->
</style>

<div class="pink">background:pink</div>
<div class="gold">background:gold</div>
```

各浏览器截图：

IE6/IE7/IE8(Q)	IE8(S)/Firefox/Chrome/Safari/Opera
<code>.pink { background:pink; } .gold {background:gold; }</code>	<code>.gold {background:gold; }</code>
<code>background:pink background:gold</code>	<code>background:pink background:gold</code>

- 在IE6/IE7/IE8(Q)下，浏览器直接忽略“<!--”之前多余的字符；
- 在其他浏览器下，多余的字符将会影响到第一个选择器。

## 三、CSS hack

### 1. CSS hack简介

由于不同的浏览器，比如IE6、IE7、IE8、Firefox等，对CSS解析的标准不同，因此对于相同的CSS代码，可能会生成不同的页面效果，从而无法在所有浏览器中得到我们想要的效果。这时，我们就需要针对不同的浏览器去写不同的CSS代码，让它能够在所有浏览器中获得相同的效果。这个过程，就是CSS hack。也就是上面说到的：“写出只有个别浏览器或某些浏览器识别的CSS代码”。

CSS hack是因为现有浏览器对标准的解析不同，为了兼容各浏览器，所采用的一种补救方法。

CSS hack是一种类似作弊的手段，以欺骗浏览器的方式达到兼容的目的，是用浏览器的兼容性差异来解决浏览器的兼容性问题。

因此，在设计之初，写CSS hack需要遵循以下三条原则：

- 有效：能够通过 Web 标准的验证
- 只针对太古老的/不再开发的/已被抛弃的浏览器，而不是目前的主流浏览器
- 代码要丑陋。让人记住这是一个不得已而为之的 Hack, 时刻记住要想办法去掉它。

现在很多hacks已经抛弃了最初的原则。

而滥用hack会导致浏览器更新之后产生更多的兼容性问题。

因此，并不推荐使用CSS hack来解决兼容性问题。

## 2. CSS hack的实现方式

1. 利用浏览器对相同代码的解析和支持的不同实现的hack
2. 以Firefox或Webkit特有的扩展样式实现的hack
3. 利用IE对标准的支持缺陷写的hack
4. 以IE特有的条件注释为基础的hack

## 3. Windows系统下CSS hack汇总表

说明：

1. 此汇总表中测试浏览器的版本为
  - IE6
  - IE7
  - IE8
  - Firefox 3.6.2
  - Safari 4.0.4
  - Chrome 5.0.356.2 dev
  - Opera 10.51
2. 其中，多数CSS hack是在selector{property:value;}基础上更改的。selector代表CSS选择器，property代表CSS特性，value代表特性的值。
3. FF代表Firefox，Ch代表Chorme，Sa代表Safari，Op代表Opera
4. Q代表Quirks Mode，S代表Standards Mode。

	IE6		IE7		IE8		FF		Ch		Sa		Op	
	Q	S	Q	S	Q	S	Q	S	Q	S	Q	S	Q	S
*+html selector	N	N	N	Y	N	N	N	N	N	N	N	N	N	N
*html selector	Y	Y	Y	N	Y	N	N	N	N	N	N	N	N	N
_property:value	Y	Y	Y	N	Y	N	N	N	N	N	N	N	N	N
-property:value	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N
html* selector	Y	Y	Y	Y	Y	N	N	N	N	N	N	N	N	N
*property:value	Y	Y	Y	Y	Y	N	N	N	N	N	N	N	N	N
property:value\9	Y	Y	Y	Y	Y	Y	N	N	N	N	N	N	N	N
selector, x:-moz-any-link{...}	N	N	Y	Y	Y	N	Y	Y	N	N	N	N	N	N
selector, x:-moz-any-link, x:default{...}	N	N	Y	Y	Y	N	Y	Y	N	N	N	N	N	N
@-moz-document url-prefix(){...}	N	N	N	N	N	N	Y	Y	N	N	N	N	N	N
html>/**/body selector, x:-moz-any-link	N	N	N	N	N	N	Y	Y	N	N	N	N	N	N
html>/**/body selector, x:-moz-any-link, x:default{...}	N	N	N	N	N	N	Y	Y	N	N	N	N	N	N
以-moz开头的Firefox特有扩展样式	N	N	N	N	N	N	Y	Y	N	N	N	N	N	N
@media screen and (-webkit-min-device-pixel-ratio:0) {...}	N	N	N	N	N	N	N	N	Y	Y	Y	Y	N	N
以-webkit开头的Webkit浏览器特有扩展样式	N	N	N	N	N	N	N	N	Y	Y	Y	Y	N	N
@media all and (-webkit-min-device-pixel-ratio:10000), not all and (-webkit-min-device-pixel-ratio:0){...}	N	N	N	N	N	N	N	N	N	N	N	N	Y	Y

### (1). 利用浏览器对相同代码的支持差异实现的hack

不同浏览器对相同的CSS代码的支持情况可能不同。尤其是对错误的写法。

例如，

```
#test{
  _width:80px;
}
```

在IE7及以上版本的浏览器中会被当作错误特性而舍弃，但是在IE6中可以被正常的解析。这时候，可以把\_width当作hack，专门针对IE6来设置元素的宽度。

更多关于浏览器对错误写法的处理，请参照：[http://docs.google.com/Doc?docid=0AXBWac2e3BJzZGZwNWg0cGdfMTA3OHBjbnBwcm4z&hl=zh\\_CN](http://docs.google.com/Doc?docid=0AXBWac2e3BJzZGZwNWg0cGdfMTA3OHBjbnBwcm4z&hl=zh_CN)

### \*+html selector

测试用例：

```
<style type="text/css">
  *+html #test {
    color: red;
  }
</style>
<h1 id="test">TEXT</h1>
```

各浏览器截图：

IE7(S)	IE6(Q)(S)/IE7(Q)/IE8(Q)(S)/ Firefox/Chrome/Safari/Opera
TEXT	TEXT

### \*html selector

测试用例：

```
<style type="text/css">
  *html #test {
    color: red;
  }
</style>
<h1 id="test">TEXT</h1>
```

各浏览器截图：

IE6(Q)(S)/IE7(Q)/IE8(Q)	Firefox/Chrome/Safari/Opera
TEXT	TEXT

### html\* selector

测试用例：

```
<style type="text/css">
  html* #test {
    color: red;
  }
</style>
<h1 id="test">TEXT</h1>
```

各浏览器截图：

<u>IE6(Q)(S)/IE7(Q)(S)/IE8(Q)</u>	<u>IE8(S)/Firefox/Chrome/Safari/Opera</u>
TEXT	TEXT

### **\_property:value**

测试用例：

```
<style type="text/css">
  #test {
    _color: red;
  }
</style>
<h1 id="test">TEXT</h1>
```

各浏览器截图：

<u>IE6(Q)(S)/IE7(Q)/IE8(Q)</u>	<u>IE7(S)/IE8(S)/Firefox/Chrome/Safari/Opera</u>
TEXT	TEXT

### **-property:value**

测试用例：

```
<style type="text/css">
  #test {
    -color: red;
  }
</style>
<h1 id="test">TEXT</h1>
```

各浏览器截图：

<u>IE6(Q)(S)</u>	<u>IE7(Q)(S)/IE8(Q)(S)/Firefox/Chrome/Safari/Opera</u>
TEXT	TEXT

### **\*property:value**

测试用例：

```
<style type="text/css">
  #test {
    *color: red;
  }
</style>
```

```
<h1 id="test">TEXT</h1>
```

各浏览器截图：

IE6(Q)(S)/IE7(Q)(S)/IE8(Q)	IE8(S)/Firefox/Chrome/Safari/Opera
TEXT	TEXT

### property:value\9

测试用例：

```
<style type="text/css">
  #test {
    color: red\9;
  }
</style>
<h1 id="test">TEXT</h1>
```

各浏览器截图：

IE6(Q)(S)/IE7(Q)(S)/IE8(Q)(S)	Firefox/Chrome/Safari/Opera
TEXT	TEXT

## (2). 以Firefox或Webkit特有的扩展样式实现的hack

以-moz或-webkit开头的扩展样式，是浏览器对CSS标准的扩展。这些特性只在相应的浏览器里才可以被正常的解析。因此可以被当作CSS hack来使用。

### 以-moz开头的Firefox特有扩展样式

Mozilla浏览器支持的一些扩展是以-moz开头的。这些扩展包括了一些功能，例如圆形边界等。这种CSS只适用于Mozilla浏览器。

比如，-moz-opacity是在Firefox2.0版本出现的特性，用来实现元素的透明，但Firefox3.0以后的版本中不再支持，而是直接支持标准中的opacity特性。而其他浏览器不支持-moz-opacity。所以，可以使用-moz-opacity来针对Firefox2.0设置元素的透明度。

关于Firefox扩展样式的详细信息，见[Mozilla CSS Extensions](#)。

### 以-webkit开头的Webkit浏览器特有扩展样式

与以-moz开头的Firefox特有扩展样式相同，以-webkit开头的样式是Webkit浏览器特有的，只有Webkit浏览器可以解析。

比如，在Webkit浏览器中可以用 -webkit-border-radius实现圆角。

### @media screen and (-webkit-min-device-pixel-ratio:0) {... }

测试用例：

```
<style type="text/css">
@media screen and (-webkit-min-device-pixel-ratio:0) {
  #test {
```

```

        color: red;
    }
}
</style>
<h1 id="test">TEXT</h1>

```

各浏览器截图：

IE6(Q)(S)/IE7(Q)(S)/IE8(Q)(S)/ Firefox/Opera	Chrome/Safari
TEXT	TEXT

### @-moz-document url-prefix(){...}

测试用例：

```

<style type="text/css">
@-moz-document url-prefix(){
    #test {
        color: red;
    }
}
</style>
<h1 id="test">TEXT</h1>

```

各浏览器截图：

IE6(Q)(S)/IE7(Q)(S)/IE8(Q)(S)/ Chrome/Safari/Opera	Firefox
TEXT	TEXT

### selector, x:-moz-any-link{...}

测试用例：

```

<style type="text/css">
    #test, x:-moz-any-link {
        color: red;
    }
</style>
<h1 id="test">TEXT</h1>

```

各浏览器截图：

IE6(Q)(S)/IE8(S)/Chrome/Safari/ Opera	Firefox/IE7(Q)(S)/IE8(Q)
TEXT	TEXT

### selector, x:-moz-any-link, x:default{...}

测试用例：

```
<style type="text/css">
  #test, x:-moz-any-link, x:default {
    color: red;
  }
</style>
<h1 id="test">TEXT</h1>
```

各浏览器截图：

<u>IE6(Q)(S)/IE8(S)/Chrome/Safari/Opera</u>	<u>Firefox/IE7(Q)(S)/IE8(Q)</u>
TEXT	TEXT

### html>/\*\*/body selector, x:-moz-any-link

测试用例：

```
<style type="text/css">
  html>/**/body #test, x:-moz-any-link {
    color: red;
  }
</style>
<h1 id="test">TEXT</h1>
```

各浏览器截图：

<u>IE6(Q)(S)/IE7(Q)(S)/IE8(Q)(S)/Chrome/Safari/Opera</u>	<u>Firefox</u>
TEXT	TEXT

### html>/\*\*/body selector, x:-moz-any-link, x:default{...}

测试用例：

```
<style type="text/css">
  html>/**/body #test, x:-moz-any-link, x:default {
    color: red;
  }
</style>
<h1 id="test">TEXT</h1>
```

各浏览器截图：

<u>IE6(Q)(S)/IE7(Q)(S)/IE8(Q)(S)/Chrome/Safari/Opera</u>	<u>Firefox</u>
--	----------------

TEXT	TEXT
------	------

**@media all and (-webkit-min-device-pixel-ratio:10000), not all and (-webkit-min-device-pixel-ratio:0){...}**

测试用例：

```
<style type="text/css">
@media all and (-webkit-min-device-pixel-ratio:10000), not all and (-webkit-min-device-pixel-ratio:0){
  #test {
    color: red;
  }
}
</style>
<h1 id="test">TEXT</h1>
```

各浏览器截图：

IE6(Q)(S)/IE7(Q)(S)/IE8(Q)(S)/ Chrome/Safari/Firefox	Opera
TEXT	TEXT

### (3). 利用IE对标准的支持缺陷写的hack

这个类别以中的hack以IE对标准的支持缺陷为基础，可以让CSS代码针对IE6或IE7以外的浏览器生效。

例如，!important只有IE7及以上版本的IE及其他浏览器支持，所以，可以用!important来针对IE6以外的浏览器写CSS代码；再如，**head:first-child+body selector**，**:first-child**不被IE6支持，所以可以用来针对IE6以外的浏览器编写CSS代码。

此处不在一一列举。IE对标准的支持缺陷，请参照：[IE对一些CSS2.1标准的支持缺陷](#)。

当然，有的观点认为应用CSS2.1标准，不属于CSS hack。见[Tantek's Thoughts](#)，**Using A CSS2 Feature Is NOT a Hack**。

### (4). 以IE特有的条件注释为基础的hack

IE浏览器中特有的条件注释也经常被用作hack，可以针对特定版本的IE写CSS代码。

例如：

测试用例：

```
<!--[if IE 8]>
<style type="text/css">
  #test {
    color: red;
  }
}
```



```
</style>  
<![endif]-->  
<h1 id="test">TEXT</h1>
```

以上代码中的**TEXT**，只在IE8中才会是红色。

更详细的说明，请参照：[IE浏览器中特有的条件注释](#)。