比较IE的hasLayout特性和CSS规范中的block formatting context

标签：hasLayout, block formatting context

操作系统版本：
Windows 7 Ultimate build 7600
浏览器版本：
IE6
IE7
IE8
Firefox 3.6.2
Safari 4.0.4
Chrome 5.0.356.2 dev
受影响的浏览器：
所有浏览器

一、hasLayout和block formatting context简介

1. IE专有的Layout及hasLayout属性

"Layout"是IE/Win的专有概念，它决定了元素如何对其内容进行定位和尺寸计算，与其他元素的关系和相互作用，以及对应用还有使用者的影响。

概念说明：

"Layout"可以被某些CSS特性(property)不可逆的触发，而某些HTML元素本身就具有layout。

"Layout"在IE/Win中通过hasLayout属性来判断一个元素是否拥有layout。

hasLayout是IE浏览器渲染引擎的一个内部组成部分。在IE浏览器中，一个元素要么自己对自身的内容进行组织和计算大小，要么依赖于包含块来计算尺寸和组织内容。为了协调这两种方式的矛盾，渲染引擎采用了 hasLayout的属性，属性值可以为true或false。

当一个元素的 hasLayout属性值为true时，我们说这个元素有一个布局（layout），或拥有布局。

触发方式：

默认拥有布局的元素：

```
<html>, <body>
<table>, <tr>, <th>, <td>
<img>
<hr>
<input>, <button>, <select>, <textarea>, <fieldset>, <legend>
<iframe>, <embed>, <object>, <applet>
<marquee>
```

可触发hasLayout的CSS特性：

```
display: inline-block
height: (除auto外任何值)
width: (除auto外任何值)
float: (left 或 right)
position: absolute
writing-mode: tb-rl
zoom: (除 normal 外任意值)
```

Internet Explorer 7 还有一些额外的属性(不完全列表)可以触发hasLayout:

```
min-height: (任意值)
min-width: (任意值)
max-height: (除 none 外任意值)
max-width: (除 none 外任意值)
overflow: (除 visible 外任意值，仅用于块级元素)
overflow-x: (除 visible 外任意值，仅用于块级元素)
overflow-y: (除 visible 外任意值，仅用于块级元素)
position: fixed
```

2. Block formatting context的概念

Block formatting context是CSS规范中的一个概念，它决定了元素如何对其内容进行定位，以及与其他元素的关系和相互作用。

概念说明：

在创建了block formatting context的元素中，其子元素会一个接一个地被放置，他们垂直方向的起点是一个包含块的顶部。两个相邻的元素之间的垂直距离取决于'margin'属性。在block formatting context中相邻的块级元素的垂直边距会折叠(collapse)。

在block formatting context中，每一个元素左外边与包含块的左边相接触（对于从右到左的格式化，右外边接触右边），即使存在浮动也是如此（尽管一个元素的内容区域会由于浮动而压缩），除非这个元素也创建了一个新的block formatting context。

创建方式：

浮动元素，绝对定位元素，inline-blocks，table-cells，table-captions，以及 'overflow'不是'visible'的元素，会创建block formatting context。

关于block formatting context的更多信息，请参考CSS2.1规范中9.4.1的内容。

二、hasLayout和block formatting context的特点

1. 当hasLayout的元素和创建了block formatting context的元素中包含浮动元素时

在hasLayout的元素和创建了block formatting context的元素中，浮动元素参与高度的计算。

情况1：没有创建block formatting context的块级非替换元素，触发了IE的hasLayout。

测试代码如下：

```
<div style="width:300px;">
   <div id="Container" style="background:silver; zoom:1;">
      <span id="SPAN1" style="background:gray;">simple text</span>
      <div id="DIV1" style="width:150px; height:50px; background:dimgray;">in
flow</div>
      <div id="DIV2" style="float:left; background:gold;">float:left</div>
   </div>
</div>
```

- **Container**没有创建block formatting context。
- **Container**的zoom:1，是为了触发IE中的**hasLayout**属性；
- **Container**的高度值为auto，并且overflow的值为默认的visible；
- **SPAN1**是一个行内元素，**DIV1**是一个处于普通流中的块元素；
- **DIV2**是一个浮动的块级元素。

根据CSS2.1规范第10.6.3部分的高度计算规则，在进行普通流中的块级非替换元素的高度计算时，浮动子元素不参与计算。
所以，在进行**Container**高度计算时，只受**SPAN1**和**DIV1**的影响，应该是它们两个的高度之和，所以最终银色部分不应该包含金色的部分。

这段代码在不同的浏览器环境中表现如下：

IE6(S)(Q)/IE7(S)(Q)/IE8(Q)

其他浏览器

去掉**Container**的zoom:1后，各浏览器表现一致：

可见，IE浏览器中，触发hasLayout的元素在进行高度计算的时候，其浮动的子元素也会参与运算。

情况2：创建了block formatting context的块级非替换元素，未触发IE的hasLayout。

```
<div style="width:300px;">
   <div id="Container" style="background:silver; overflow:hidden;">
      <span id="SPAN1" style="background:gray;">simple text</span>
      <div id="DIV1" style="width:150px; height:50px; background:dimgray;">in
flow</div>
      <div id="DIV2" style="float:left; background:gold;">float:left</div>
   </div>
</div>
```

- **Container**的overflow:hidden;创建了block formatting context；
- **Container**的overflow:hidden;，在IE6中未触发hasLayout，但在IE7中触发了hasLayout；
- **Container**的高度值为auto；
- **SPAN1**是一个行内元素，**DIV1**是一个处于普通流中的块元素；
- **DIV2**是一个浮动的块级元素。

根据CSS2.1规范第10.6.7部分的高度计算规则，在计算生成了block formatting context的元素的高度时，其浮动子元素应该参与计算。
所以，在进行**Container**高度计算时，**DIV2**也应该参与计算，所以最终银色部分应该包含金色的部分。

这段代码在不同的浏览器环境中表现如下：(注意IE7(S)此时触发了hasLayout)

IE6(S)(Q)/IE7(Q)/IE8(Q)

其他浏览器

可见，只要**Container**创建了block formatting context，其浮动子元素就会参与其高度计算。
（IE7(S)是由于hasLayout导致与其他浏览器的效果相同。）

2. hasLayout的元素和创建了block formatting context的元素与浮动元素的相互作用

与浮动元素相邻的、触发了hasLayout的元素或创建了block formatting context的元素，都不能与浮动元素相互覆盖。如果浮动元素的两侧有足够的空间放置该元素，则元素会紧邻浮动元素放置，必要时，该元素的宽度将会被压缩。否则它们可能会定位到浮动元素的下方。

情况1：没有创建block formatting context的块级非替换元素，触发了IE的hasLayout。

测试代码如下：

```
<div id="Container" style='border:2px solid gold; width:300px; height:150px;
background:url("grid2a.png") repeat;'>
   <div id="DIV1" style="background-color:gold; width:100px; height:100px;
float:left; filter:alpha(opacity=50); opacity: 0.5;">
```

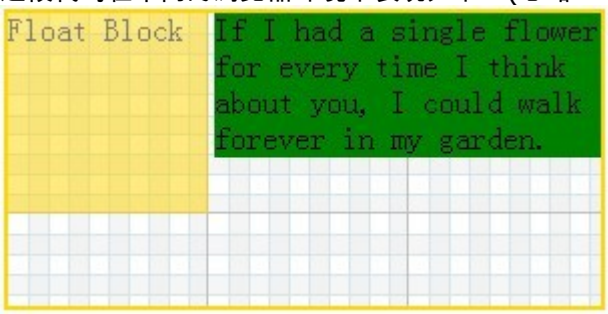
```
        Float Block
    </div>
    <div id="DIV2" style="background-color:green; zoom:1;">
        If I had a single flower for every time I think about you, I could walk forever in my garden.
    </div>
</div>
```

- **DIV1**是一个浮动元素，背景是50%的透明
- **DIV2**的zoom:1触发了IE中的hasLayout。

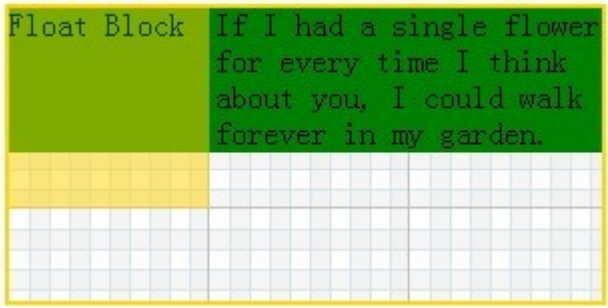
其中，**grid2a.png**背景是100px*100px的图片：

根据W3C CSS2.1 9.5的标准，浮动元素会覆盖普通流中的块容器。
所以，**DIV2**应该有一部分呢被**DIV1**覆盖。

这段代码在不同的浏览器环境中表现如下：(忽略IE中3px bug的影响)



IE6(S)(Q)/IE7(S)(Q)/IE8(Q)



其他浏览器

情况2：创建了block formatting context的块级非替换元素，未触发IE的hasLayout。

测试代码如下：

```
<div id="Container" style='border:2px solid gold; width:300px; height:150px; background:url("grid2a.png") repeat;'>
    <div id="DIV1" style="background-color:gold; width:100px; height:100px; float:left; filter:alpha(opacity=50); opacity: 0.5;">
        Float Block
    </div>
```

```
    <div id="DIV2" style="background-color:green; overflow:hidden;">
        If I had a single flower for every time I think about you, I could walk forever in my garden.
    </div>
</div>
```

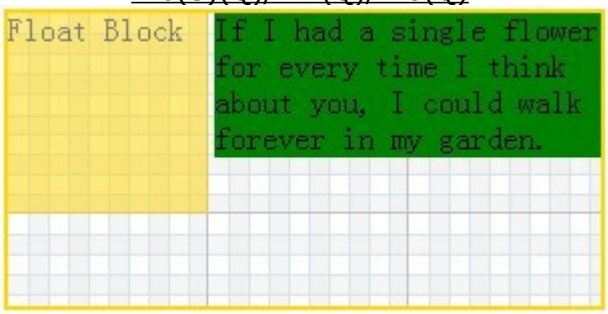
- **DIV1**是一个浮动元素，背景是50%的透明
- **DIV2**的overflow:hidden;在IE6中未触发hasLayout，但在IE7中触发了hasLayout

其中，**grid2a.png**背景是100px*100px的图片：

根据W3C CSS2.1 9.5的标准，浮动元素会覆盖普通流中的块容器。
所以，**DIV2**应该有一部分呢被**DIV1**覆盖。

这段代码在不同的浏览器环境中表现如下：(注意IE7(S)此时触发了hasLayout)

IE6(S)(Q)/IE7(Q)/IE8(Q)



其他浏览器

3. 当hasLayout的元素和创建了block formatting context的元素的外边距折叠(margin collapse)

触发hasLayout的元素和创建了block formatting context的元素不会与它们的子元素发生外边距折叠。

情况1：**没有生成block formatting context的块级非替换元素，触发了IE的hasLayout。**

测试代码如下：

```
<div id="Container" style="width:300px; border:1px solid gold;">
    <div id="DIV1" style="zoom:1; background:darkgray;">
```

```
    <div id="DIV2" style="margin:30px 0; width:60px;">content</div>
  </div>
</div>
```

- **Container**是300px，含有border的块元素，根据标准，它不会与子元素的margin发生空白边折叠。
- **DIV1**的宽度没有设置，所以宽度等于**Container**r的宽度
- **DIV1**的高度也没有设置，所以其高度取决于其内容的高度。
- **DIV1**设置了zoom:1，在IE中触发了hasLayout。

根据标准8.3.1 第一条，两个相邻的普通流中的块框在垂直位置的空白边会发生折叠现象。
DIV1和**DIV2** 应该发生空白边折叠，深灰色的**DIV1**应该刚好包含"content"文本。

这段代码在不同的浏览器环境中表现如下：

IE6(S)(Q)/IE7(S)(Q)/IE8(Q)

其他浏览器

可见，在IE中，触发hasLayout的元素，阻止了它自身与子元素间的空白边折叠。

情况2：生成block formatting context的块级非替换元素，未触发IE的hasLayout。

测试代码如下：

```
<div id="Container" style="width:300px; border:1px solid gold;">
  <div id="DIV1" style="overflow:hidden; background:darkgray;">
    <div id="DIV2" style="margin:30px 0; width:60px;">content</div>
  </div>
</div>
```

- **Container**是300px，含有border的块元素，根据标准，它不会与子元素的margin发生空白边折叠。
- **DIV1**的宽度没有设置，所以宽度等于**Container**r的宽度。
- **DIV1**的高度也没有设置，所以其高度取决于其内容的高度。
- **DIV1**设置了overflow:hidden，在IE6中未触发hasLayout，但在IE7中触发了hasLayout。

根据标准8.3.1 第一条，两个相邻的普通流中的块框在垂直位置的空白边会发生折叠现象。
DIV1和**DIV2** 不应该发生空白边折叠，深灰色的**DIV1**应该撑满**Container**。

这段代码在不同的浏览器环境中表现如下：(注意IE7(S)此时触发了hasLayout)

IE6(S)(Q)/IE7(S)(Q)/IE8(Q)

其他浏览器

可见，在IE中，创建了block formatting context，未触发hasLayout的元素，它自身与子元素间的空白边折叠还是会发生。

三、hasLayout和block formatting context的异同及可能产生的问题

区别：

- 在IE8(S)之前的版本中，没有规范中提及的block formatting context和Inline formatting context
  概念，而是用hasLayout来达到相同的目的。
- 在IE中可通过设置width、height、min-width、min-height、max-width、max-height、zoom、writing-mode、display值触发hasLayout，而这些特性值的设置不能够使元素创建block formatting context。
- 在IE中很多元素默认就是拥有布局的，如<input>, <button>, <select>, <textarea>等，但是这些元素在标准中会形成Inline formatting context。

共同点：

- 两者都是决定了对内容如何定位及大小计算的规则。
- 两者都决定了与其他元素的相互作用的规则。
- table-cell，table-caption即是hasLayout的元素，又可以创建block formatting context的元素。
- 浮动元素，绝对定位元素，inline-block元素以及除visible外任意值的overflow(IE7)在IE中可以触发hasLayout，同时在标准中，又可以创建block formatting context。

可能产生的兼容性问题：

由于hasLayout和block formatting context是对一类事物的不同理解，并且他们的启用条件不尽相同，因此如果一个元素设计时，在IE早期版本中触发了hasLayout，但在其他浏览器中又没有创建block formatting context，或者相反，一个元素在IE早期版本中没有触发hasLayout，在其他浏览器中却创建了block formatting context（如设置了overflow:hidden），将导致页面布局的重大差异。

四、如何避免受此问题影响

仅当一个元素即在IE早期版本中触发了hasLayout，又在其他浏览器中创建了block formatting context时，才能避免上述问题的发生。即同时启用上述两者以保证各浏览器的兼容，或者相反，两者皆不启用：

1. 使元素即生成了block formatting context，又触发了hasLayout
   - 对于触发hasLayout的元素，通过CSS设置，使它产生block formatting context；
   - 生成block formatting context但是没有触发hasLayout的元素，通过设置"zoom:1"，使其触发hasLayout。
2. 使元素即没有触发hasLayout，又没有创建block formatting context。