最近也开始关注hasLayout的东西. 先放在这里, 有空再整理研究

英文原文在此。  
<http://www.satzansatz.de/cssd/onhavinglayout>  
文中所有的 layout 这个单词都未作翻译，一来本身这个单词意思就比较多，翻成啥都觉得别扭，二来它也是专有的属性，所以就意会一下吧。水平有限，很多地方都是模模糊糊地意译，发现错误欢迎留言指出。  
引用一段来自[Dean Edwards](http://dean.edwards.name/weblog/)  
的评价：  
*I recommend that every CSS designer and DOM scripter read this. Understanding “layout” gives a huge insight into lots of other IE bugs and idiosyncrasies.*  
([Dean Edwards](http://dean.edwards.name/weblog/2005/06/layout/))  
  
**4月16日修订的内容**：  
将quirks模式这一部分单独移动到一篇文章中讲述。  
添加：边缘裁切。  
添加：收缩包围(shrink-wrapping)现象。  
  
**5月17日修订的内容**：  
重写了了浮动元素旁边的元素这一部分。  
部分章节小的修正：属性，有关内联级别元素，CSS hacks。  
重新整理了部分章节的语言：定义，数据，问题种种，分析，清除浮动和自动扩展适应高度，绝对定位元素。  
  
  
**On having layout**  
  
本文修订中  
当前版本：Rev. 7 2006–05–17  
<http://www.satzansatz.de/cssd/onhavinglayoutrev07-20060517.html>  
修订历史  
<http://www.satzansatz.de/cssd/layoutchangelog.html>  
各种语言版本  
目录  
  
  
**介绍**  
  
  
Internet Explorer 中有很多奇怪的渲染问题可以通过赋予其“layout”得到解决。John Gallant 和 Holly Bergevin 把这些问题归类为“尺寸bug(dimensional bugs)”，意思是这些 bug 可以通过赋予相应元素某个宽度或高度解决。这便引出关于“layout”的一个问题：为什么它会改变元素的渲染特性，为什么它会影响到元素之间的关系？这个问题问得很好，但却很难回答。在这篇文章中，我们专注于这个复杂问题会有那些方面的表现，某一方面的具体讨论和范例请参考文中给出的相关链接。  
  
  
  
**hasLayout — 定义**  
  
  
  
“Layout”是一个 IE/Win 的私有概念，它决定了一个元素如何显示以及约束其包含的内容、如何与其他元素交互和建立联系、如何响应和传递应用程序事件/用户事件等。  
这种渲染特性可以通过某些 CSS 属性被不可逆转地触发。而有些 HTML 元素则默认就具有“layout”。  
微软的开发者们认为元素都应该可以拥有一个“属性(property)”(这是面向对象编程中的一个概念)，于是他们便使用了 hasLayout，这种渲染特性生效时也就是将 hasLayout 设成了 true 之时。  
**术语**  
当我们说一个元素“得到 layout”，或者说一个元素“拥有 layout” 的时候，我们的意思是指它的微软专有属性 hasLayout [http://msdn.microsoft.com/worksh ... rties/haslayout.asp](http://msdn.microsoft.com/workshop/author/dhtml/reference/properties/haslayout.asp) 为此被设为了 true 。一个“layout元素”可以是一个默认就拥有 layout 的元素或者是一个通过设置某些 CSS 属性得到 layout 的元素。  
而“无layout元素”，是指 hasLayout 未被触发的元素，比如一个未设定宽高尺寸的干净 div 元素就可以做为一个“无layout祖先”。  
给一个默认没有 layout 的元素赋予 layout 的方法包括设置可触发 hasLayout = true 的 CSS 属性。参考默认 layout 元素以及这些属性列表。没有办法设置 hasLayout = false ， 除非把一开始那些触发 hasLayout = true 的 CSS 属性去除。  
  
  
  
**问题种种**  
  
  
hasLayout 的问题不管新手还是老手，不管设计师或者程序员可能都遇到过。Layout 在显示盒子时有着不同寻常而且难以预料的效果，而且有时甚至会牵连到他们的孩子元素。  
一个元素是否具有“layout”可能会引发如下的一些问题：  
IE 很多常见的浮动 bug 。  
元素本身对一些基本属性的异常处理问题。  
容器和其子孙之间的边距重叠(margin collapsing)问题。  
使用列表时遇到的诸多问题。  
背景图像的定位偏差问题。  
使用脚本时遇到的浏览器之间处理不一致的问题。  
  
上面的列表只是列出一个大概，也不完善。下面的文章将尽可能详细彻底的描述有无“layout”所带来的各种问题。  
  
  
  
**Layout 从何而来**  
  
  
不同于标准属性，也不像某些浏览器的私有 CSS 属性，layout 无法通过某一个 CSS 声明直接设定 。也就是说没有“layout属性”这么一个东西，元素要么本身自动拥有 layout，要么借助一些 CSS 声明悄悄地获得 layout。  
**默认layout元素**  
下列元素应该是默认具有 layout 的：

复制内容到剪贴板

代码:

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

**属性**  
下列 CSS 属性和取值将会让一个元素获得 layout：  
position: absolute  
绝对定位元素的包含区块(containing block)就会经常在这一方面出问题。  
float: left|right  
由于 layout 元素的特性，浮动模型会有很多怪异的表现。  
display: inline-block  
当一个内联级别的元素需要 layout 的时候往往就要用到它，这也可能也是这个 CSS 属性的唯一效果——让某个元素拥有 layout。“inline-block行为”在IE中是可以实现的，但是非常与众不同： IE/Win: inline-block and hasLayout <http://www.brunildo.org/test/InlineBlockLayout.html>。  
width: 除 “auto” 外的任意值  
很多人遇到 layout 相关问题发生时，一般都会先尝试用这个来修复。  
height: 除 “auto” 外的任意值  
height: 1% 就在 Holly Hack 中用到。  
zoom: 除 “normal” 外的任意值 (MSDN) [http://msdn.microsoft.com/worksh ... properties/zoom.asp](http://msdn.microsoft.com/workshop/author/dhtml/reference/properties/zoom.asp)  
MS专有属性，无法通过校验。 不过 zoom: 1 可以临时用做调试。  
writing-mode: tb-rl (MSDN) [http://msdn.microsoft.com/worksh ... ies/writingmode.asp](http://msdn.microsoft.com/workshop/author/dhtml/reference/properties/writingmode.asp)  
MS专有属性，无法通过校验。  
  
在 IE7 中，overflow 也变成了一个 layout 触发器：  
overflow: hidden|scroll|auto  
这个属性在之前版本 IE 中没有触发 layout 的功能。  
overflow-x|-y: hidden|scroll|auto  
overflow-x 和 overflow-y 是 CSS3 盒模型中的属性，尚未得到浏览器的广泛支持。他们在之前版本IE中没有触发 layout 的功能。  
  
另外 IE7 的荧幕上又新添了几个 haslayout 的演员，如果只从 hasLayout 这个方面考虑，min/max 和 width/height 的表现类似，position 的 fixed 和 absolute 也是一模一样。  
position: fixed  
./.  
min-width: 任意值  
就算设为0也可以让该元素获得 layout。  
max-width: 除 “none” 之外的任意值  
./.  
min-height: 任意值  
即使设为0也可以让该元素的 haslayout=true  
max-height: 除 “none” 之外的任意值  
./.  
  
以上结论借助 IE Developer Toobar 以及预先测试得出。  
**有关内联级别元素**  
对于内联元素(可以是默认即为内联的比如 span 元素，也可以是 display: inline 的元素)  
width 和 height 只在 IE5.x 下和 IE6 或更新版本的 quirks 模式下触发 hasLayout 。而对于 IE6，如果浏览器运行于标准兼容模式下，内联元素会忽略 width 或 height 属性，所以设置 width 或 height 不能在此种情况下令该元素具有 layout。  
zoom 总是可以触发 hasLayout，但是在 IE5.0 中不支持。  
  
具有“layout” 的元素如果同时也 display: inline ，那么它的行为就和标准中所说的 inline-block 很类似了：在段落中和普通文字一样在水平方向和连续排列，受 vertical-align 影响，并且大小可以根据内容自适应调整。这也可以解释为什么单单在 IE/Win 中内联元素可以包含块级元素而少出问题，因为在别的浏览器中 display: inline 就是内联，不像 IE/Win 一旦内联元素拥有 layout 还会变成 inline-block。  
**脚本属性 hasLayout**  
我们这里称 hasLayout 为“脚本属性”是为了和我们熟知的 CSS 属性相区别。  
注意一旦一个元素拥有了 layout，就没有办法再将其设成 hasLayout = False 了。  
hasLayout-property [http://msdn.microsoft.com/worksh ... rties/haslayout.asp](http://msdn.microsoft.com/workshop/author/dhtml/reference/properties/haslayout.asp)可以用来检测一个元素是否拥有 layout：举个例子，如果它的 id 是“eid”，那么只要在 IE5.5+ 的地址栏里输入 javascript: alert(eid.currentStyle.hasLayout) 即可检测它的状态。  
IE的 Developer Toolbar [http://www.microsoft.com/downloa ... &displaylang=en](http://www.microsoft.com/downloads/details.aspx?FamilyID=e59c3964-672d-4511-bb3e-2d5e1db91038&displaylang=en)可以实时检查一个元素的当前样式；如果 hasLayout 是 true ，那么它的值显示为 “-1”。 我们可以通过实时修改一个元素的属性将“zoom(css)”设置为“1”来触发 hasLayout 以便调试。  
另外一个需要注意的是“layout”会影响脚本编程。如果一个元素没有“layout”，那么clientWidth/clientHeight 总是返回0。这会让一些脚本新手感到困惑，而且这和 Mozilla 浏览器的处理方式也不一样。不过我们可以利用这一点在 IE5.0 中检测“layout”：如果 clientWidth 是零那么这个元素就没有 layout。  
  
  
  
**CSS hacks**  
  
  
  
下面用于触发 haslayout 的 hack 已经经过 IE6 及以下版本测试。今后版本的IE有可能会对此做不同处理。如果新版本浏览器发布我们会重新整理这部分内容。  
John Gallant 和 Holly Bergevin 在2003年发布的 Holly hack [http://www.communitymx.com/conte ... age=2&cid=C37E0](http://www.communitymx.com/content/article.cfm?page=2&cid=C37E0)：

复制内容到剪贴板

代码:

/\* \\*/   \* html .gainlayout { height: 1%; }   /\* \*/

可以让 IE5+ 的任意元素获得 layout，除了标准兼容模式 IE6 中的内联元素。  
一般都很有效，除了在某些极少情况下，需要用 height:0 或者 1px 更好一些。  
和 overflow: hidden 不相容，除非在 IE6 的标注兼容模式下(因为这时如果父元素没有定高，那么height: 1% 会被变回 height: auto)。  
  
或者我们可以用 underscore hack:http://wellstyled.com/singlelang.php?lang=en&page=css-underscore-hack.html

复制内容到剪贴板

代码:

.gainlayout { \_height: 0; }

另外，更具有向后兼容性的方法是使用 条件注释(conditional comments):http://msdn.microsoft.com/workshop/author/dhtml/overview/ccomment\_ovw.asp

复制内容到剪贴板

代码:

<!--[if lte IE 6]>  
<style>  
.gainlayout { height: 1px; }  
</style>  
<![endif]-->

在条件注释中链接一个专门对 IE/Win 做修正的外部样式表文件，也不失为一个安全有效的好方法：

复制内容到剪贴板

代码:

<link rel="stylesheet" href="allbrowsers.css" type="text/css" />  
      
<!--[if lte IE 6]>  
<link rel="stylesheet" href="iefix.css" type="text/css" />  
<![endif]-->

我们更倾向于使用 height: 0 和 1px —— 并主张始终使用 height 除非它和别的什么东西冲突 (overflow: hidden)。对于取值，我们则倾向于避免 1% ，因为它可能会(虽然很少)引起一些问题。<http://www.brunildo.org/test/relayout.html>  
height 不能应用于标准模式下的内联元素。在这种情况下我们可以用 display: inline-block 或 zoom: 1。  
我们曾看过一些把 Holly hack 真的当作 holy(神圣的) hack 盲目使用的情况，比如对浮动元素使用或者对已经具有特定宽度的元素也使用这个 hack。要记住这个 hack 的目的不是要给某个元素加一个高度，而只是要触发 hasLayout = True 而已。  
不要给所有元素设置 layout：\* {\_height: 1px;}。所谓过犹不及，获得 layout 不等于获得灵丹妙药，它只是用来改变渲染模式。  
**Hack整理**  
但是浏览器总是会变的，我们需要面对很多问题，比如一些依赖 IE6 的 bug 所做的 hack 会在 IE7 或更高版本的新浏览器中因 bug 修复而失效(甚至有害)的问题；比如新版本浏览器中类似的布局 bug 依然存在但用于 hack 的过滤器比如 \* html 却不能正常工作的问题。这种情况下，MS专有属性 zoom 就可以考虑使用了。

复制内容到剪贴板

代码:

<!--[if lt IE 7]><style>  
/\* IE 6 + IE5.5 + IE5.0 所用样式\*/  
.gainlayout { height: 0; }  
</style><![endif]-->  
      
<!--[if IE 7]><style>  
.gainlayout { zoom: 1;}  
/\* 或者其他任何以后可能需要的东西 \*/  
</style><![endif]-->

zoom: 1; 可以让 IE5.5+ 的任何元素(包括内联元素)获得 layout，但是在 IE5.0 中无效。  
没有其他附带效果(内联元素会变成 inline-block，这个当然)。  
如果需要通过验证，应该用条件注释将 zoom 隐藏起来。  
  
其实当我们考虑到“向后兼容”时是很自相矛盾的，我们强烈建议页面设计者回过头看一下自己页面中用的到的明显的或是不明显的“hacks”，并用条件注释针对不同浏览器重新处理以保万无一失。  
  
  
  
**关于IE Mac 的小问题**  
  
  
IE Mac 和 windows 下的 IE 是完全不同的两个东西，它们各自拥有自己的渲染引擎，IE Mac 就全然不知“hasLayout”(或contenteditable)所谓何物。相比之下 IE Mac 的渲染引擎要更标准兼容一点，比如 height 就是被当作 height 处理，没有别的效果。因此针对“hasLayout”的 hacks 和别的解决方法(特别是通过使用 height 或 width 属性的)往往对 IE Mac 来说是有害的，所以需要对其隐藏。更多的关于 IE Mac 相关的问题可以在 IE Mac, bugs and oddities pages <http://www.l-c-n.com/IE5tests/>找到。  
  
  
**MSDN 文档**  
  
  
MSDN 中涉及到 hasLayout 这个 MS 属性的地方寥寥无几，而具体解释 layout 和 IE 渲染模型之间关系的则少之又少。  
在IE4的时候，除了未经绝对定位也未指定宽高的内联元素，几乎所有元素都有某种 layout(MSDN)。[http://msdn.microsoft.com/worksh ... mentandlocation.asp](http://msdn.microsoft.com/workshop/author/css/overview/measurementandlocation.asp)在这种早期的layout概念中，像 border, margin, padding 这些属性被称作“layout属性”，它们是不能应用到一个简单的内联元素上的。换句话说，“拥有layout”就可以粗略理解成：“可以拥有这几个属性”。  
MSDN 上仍然使用 layout 属性这种说法， 只是含义变了，它们和拥有 layout 的元素已经没有什么关系了。在 IE5.5 中方才引入了 MS 的这个专有属性 hasLayout，[http://msdn.microsoft.com/worksh ... rties/haslayout.asp](http://msdn.microsoft.com/workshop/author/dhtml/reference/properties/haslayout.asp)也只是某种内部的标志位而已。  
在 IE5.5 中，MSHTML Editing Platform(即可以通过设置来允许用户实时编辑、拖动 layout 元素以及调整其尺寸等)的文档中描述了三个和 layout 相关的重要特性：  
如果一个 layout 元素中有内容，内容的排版布局将由它的边界矩形框决定。  
拥有 layout 的意思基本上就是表示某元素是一个矩形。  
从内部来说，拥有 layout 意思就是一个元素将自己负责绘制其内部内容。  
  
(Editing Platform) [http://msdn.microsoft.com/librar ... mshtmleditplatf.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnmshtml/html/mshtmleditplatf.asp)  
  
和 layout 自身相关的内部工作机制直到2005年8月才有相应文档描述，当时由于 The Web Standards Project <http://www.webstandards.org/>和微软的特别工作小组的原因，Markus Mielke [MSFT] 打开了深入讨论的大门：  
一般来说，在 Internet Explorer 的 DHTML 引擎中，元素是不对自己的位置安排负责的。虽然一个 div 或者一个 p 元素都在源码中有一个位置，在文档流有一个位置，但是它们的内容却是由它们最近的一个 layout 祖先(经常是 body)控制安排的。这些元素依赖它们祖先的 layout 来为他们处理诸如决定大小尺寸和测量信息等诸多繁重的工作。  
  
(HasLayout概述)http://msdn.microsoft.com/library/default.asp?url=/library/en-us/IETechCol/cols/dnexpie/expie20050831.asp  
  
  
**分析**  
  
  
  
我们的分析试图解释在已知案例下发生了什么事情，这种分析也应该可以作为未知案例下的指导。但我们这种试图利用种种测试案例投石探路的黑箱测试方法，是注定无法消除黑箱的神秘感的——我们无法回答“为什么”的问题。我们只能去尝试了解整个“hasLayout”模式的工作框架，以及它会怎样影响网页文档的渲染。因此，最终我们只能提供一些指导方针(而且只能是指导方针，而不是绝对的解决方案)。  
我们认为他们所指的是一个小窗体。一个 layout 元素的内部内容是完全独立的，而且也无法影响其边界外的任何内容。  
而 MS 属性 layout 只是某种标志位：一旦它被设定，这个元素就会拥有 layout“特性”，这包括体现在其自身以及其非 layout 孩子元素身上的特殊性能——比如浮动和层叠等。  
这种独立性也许正可以解释为什么 layout 元素通常比较稳定，而且它们可以让某些 bug 消失。这种情况的代价有二，一是偏离了标准，二是它没有考虑到今后可能因此出现的 bug 和问题。  
MS 的“页面”模式，从符号学角度考虑，可以看做是由很多互不相关的小的区块构成，而 HTML 和 W3C 的模式则认为“页面”模式应该是叙述完备的，故事性的相关信息区块构成的。  
  
  
  
**各种情况的详细说明**  
  
  
  
**清除浮动和自动扩展适应高度**  
浮动元素会被 layou 元素自动包含。这是很多新手经常遇到的问题：在 IE 下完成的页面到了标准兼容浏览器下所有未清除的浮动元素都伸出了其包含容器之外。  
Containing Floats <http://www.complexspiral.com/publications/containing-floats>  
how to clear floats without structural markup <http://positioniseverything.net/easyclearing.html>  
  
相反的情况：如果确实需要一个浮动元素伸出其包含容器，也就是自动包含不是想要的效果时，该怎么办？你很可能也会遇到这种头疼的问题，下面的深入讨论就是一个例子：  
acidic float tests <http://www.satzansatz.de/cssd/acidicfloat.html>  
  
在IE中，一个浮动元素总是“隶属于”它的 layout 包含容器。而后面的元素会受这个 layout 包含容器影响而不是这个浮动元素影响。  
这个特性和IE6的那个自动扩展以适应内部内容宽度的特性，都可以看成是受这个规则影响的：“由它的边界矩形框决定”。  
更糟的问题：clear 无法影响其 layout 包含容器之外的 float 元素。如果依赖这个 bug 在 IE 中布局的页面要转到标准兼容浏览器中，只有全部重做。  
IE 的自动包含浮动元素也是经常需要的效果，它在其他浏览器中也可以达到：参考我们的 “和 CSS 规范类似的地方” 这一部分来了解一下包含浮动元素的相关内容。  
**浮动元素旁边的元素**  
当一个块级元素紧跟在一个左浮动元素之后时，它应该——作为一个块级元素——忽略这个浮动元素，而它的内容则应该因这个浮动元素而移位：一个紧跟在左浮动元素后的块级元素内的文字内容，应该沿着浮动元素的右边顺序排列并会（如果它的长度超过浮动元素）继续排列到浮动元素下方。但是如果这个块级元素有 layout，比如由于某种原因被设置了宽度，那么这整个元素则会因浮动元素而移位，就好像它自己也是一个浮动元素一样，因此其中的文字就不再环绕这个左浮动元素了（而会形成一个矩形区域，保持在它的右边。）  
在 IE5 中一个块级元素的百分比宽度是基于浮动元素旁边的剩余空间计算的，而在 IE6 中则是依照整个父块级元素的可用空间计算的。所以在 IE6 中设置 width: 100% 会导致某种浮动元素旁边的溢出现象，于是各种布局问题也会因此而来。  
一些关于浮动块旁边的 hasLayout 块的测试案例：  
by using width <http://dev.l-c-n.com/IEW2-bugs/float-layout.php>  
by using min-width (IE 7) and zoom (IE 6) <http://dev.l-c-n.com/IEW2-bugs/float-adjecant.php>  
  
与此类似，和浮动元素相邻的相对定位元素，它的位置偏移量应该参照的是父元素的补白(padding)边缘(例如，left: 0; 应该将一个相对定位元素叠放于它前面的浮动元素之上)。在 IE6 中，偏移量 left: value; 是从浮动元素的右边距(margin)边缘开始算起的，这会因浮动元素所占的宽度变化导致水平方向的错位(一个解决方法是用 margin-left 代替，但是也要注意如使用百分值时会有一些怪异问题)。  
layout blocks with relative positioning adjacent to floated blocks <http://dev.l-c-n.com/IEW2-bugs/float-layout2-rp.php>  
  
根据规范所述，浮动元素应该与其后的盒子交织在一起。而对于没有交叉的二维空间中的矩形而言这是无法实现的。  
如果谁真的需要向 IE 的这种不当行为屈服，那么如何让标准兼容浏览器中的盒子也有类似行为——即类似于 layout 盒子会自动“收缩”而给其前置的浮动元素让出空间的行为——就是一个问题了。我们给出的方法是跟着一个浮动元素创建一个新的块级格式化范围（block formatting context），这在我们的“和 CSS 规范类似的地方” 有讨论。  
可以(再次)访问下面这个页面：  
three pixel text-jog <http://positioniseverything.net/explorer/threepxtest.html>  
  
我们可以看到跟在一个浮动元素后的 layout 元素不会显示这个3px间隙的 bug，因为浮动元素外围的3px硬边无法影响一个 layout 元素的内部内容，所以这个硬边将整个 layout 元素右推了3px。好比一个防护罩，layout 可以保护其内部内容不受影响，但是浮动元素的力量却将整个防护罩推了开来。  
**列表**  
无论是列表本身(ol, ul) 还是单个的列表元素(li)，拥有 layout 后都会影响列表的表现。不同版本 IE 的表现又有不同。最明显的效果就体现在列表符号上(如果你的列表自定义了列表符号则不会受这个问题影响)。这些符号很可能是通过某种内部机制附到列表元素上的(通常是附着在它们外面)。不幸的是，由于是通过“内部机制”添加的，我们无法访问它们也无法修正它们的错误表现。  
最明显的效果有：  
列表获得 layout 后，列表符号会消失或者被放置在不同的或者错误的位置。  
  
有时它们又可以通过改变列表元素的边距而重新出现。这看起来似乎是以下事实导致的结果：layout 元素会试图裁掉超出其边界的内部内容。  
列表元素获得 layout 之后，会有和上面一样的问题出现，更多参考 (extra vertical space between list items)http://www.brunildo.org/test/IEWlispace.php  
  
进一步又有一个问题就是(在有序列表中)任何具有 layout 的列表元素似乎都有自己独立的计数器。比如我们有一个含五个列表元素的有序列表，只有第三个列表元素有 layout。我们会看到这样：  
1… 2… 1… 4… 5…  
此外，如果一个有 layout 的列表元素跨行显示时，列表符号会底部对齐(而不是按照预料的顶部对齐)。  
以上某些问题还是无法解决的，所以如果需要列表符号的时候最好避免让列表和列表元素获得 layout。如果需要限定尺寸，最好给别的元素设定尺寸，比如给整个列表外面套一个元素并设定它的宽度，又或者比如给每个列表元素中的内容设定高度等等。  
另一个IE中列表的常见问题出现在当某个 li 中的内容是一个 display: block 的锚点(anchor)时。在这种情况下，列表元素之间的空格将不会被忽略而且通常会显示成额外的一行夹在每个 li 之间。一种避免这种竖直方向多余空白的解决方法是赋予这些锚点 layout。这样还有一个好处就是可以让整个锚点的矩形区域都可以响应鼠标点击。  
**表格**  
table 总是有 layout 的，它总表现为一个已定义宽度的对象。在IE6中，table-layout: fixed[http://msdn.microsoft.com/worksh ... ies/tablelayout.asp](http://msdn.microsoft.com/workshop/author/dhtml/reference/properties/tablelayout.asp)通常和一个宽度设为100%的表格相同，同时这也会带来很多问题(一些计算方面的错误)。另外在IE5.5和IE6的quirks模式下还有一些别的需要注意的情况。<http://dev.l-c-n.com/tables_2/>  
**相对定位元素(r.p.)**  
注意，由于 position: relative 并不触发 hasLayout，所以很多诸如内容消失或错位的渲染错误就会因此而起。这些现象可能会在刷新页面、调整窗口大小、滚动页面、选中内容等情况下出现。原因是 IE 在据这个属性对元素做偏移处理时，却似乎忘了发出信号让其 layout 孩子元素进行“重绘”(而如果是一个layout元素，那么在其重绘事件的信号链中，这个传给其孩子的信号是会正常发出的)。  
r.p. parent and disappearing floated child <http://www.satzansatz.de/cssd/rpfloat.html>  
disappearing list-background bug <http://positioniseverything.net/explorer/ie-listbug.html>  
  
以上是一些相关问题的描述。作为经验之谈，相对定位一个元素时最好给予其 layout。再有，我们也需要检查拥有这种结构的父元素是否也需要 layout 或者position: relative亦或二者都需要，如果涉及到浮动元素这点就十分重要。  
**绝对定位元素(a.p.)：**  
**包含区块，什么是包含区块？**  
理解 CSS 的包含区块概念很重要，它回答了绝对定位元素是相对哪里定位的问题：包含区块决定了偏移起点，包含区块定义了百分比长度的计算参考。  
对于绝对定位元素，包含区块是由其最近的定位祖先决定的。如果其祖先都没有被定位，那么就使用初始包含区块 html。  
通常情况下我们会用 position: relative 来设定任意包含区块。这就是说，我们可以让一个绝对定位元素所参考的原点和长度等不依赖于元素的排列顺序，这可以满足诸如“内容优先”这种可访问性概念的需要，也可以给复杂的浮动布局带来方便。  
但是由于 layout 概念的存在，这种设计理念的效果在IE中就要打个问号了：因为在IE中绝对定位只有当其包含元素拥有 layout 时才会计算正确，而且绝对定位元素的百分比宽度参考也搞错了对象。这里 IE5 和 IE6 的行为不同但都有问题。IE7b2 的行为就要好很多，虽然有些小地方还是有错误。总之尽可能的让绝对元素的包含区块拥有 layout，而且尽量让其就是绝对定位元素的父级元素（也就是说这个包换元素和绝对定位元素之间没有绝对定位元素的别的祖先了）。  
假设一个无 layout 的父元素被相对定位了——我们就得给它赋予 layout 才能使偏移量起作用：  
Absolutely Buggy II <http://www.positioniseverything.net/abs_relbugs.html>  
  
假设一个未定位的父元素需要特定尺寸，而且页面设计是基于百分比宽度的——我们就可以放弃这个想法了，因为浏览器支持不佳：  
absolutely positioned element and percentage width<http://www.satzansatz.de/cssd/tmp/apboxpercentagewidth.html>  
**滤镜**  
MS专有的滤镜属性 filter <http://msdn.microsoft.com/workshop/author/filter/filters.asp>是只适用于 layout 元素的。有些滤镜扩展了对象的边界。它们会显示出自身特有的缺陷。<http://www.satzansatz.de/cssd/tmp/alphatransparency.html>  
**对已渲染元素的重排(re-flow)**  
当所有元素都已渲染完成时，如果有一个因鼠标经过而引起的变化产生(比如某个链接的 background 有变化)，IE会对其 layout 包含区块进行重排。有时一些元素就会因此被排到了新的位置，因为当这个鼠标经过发生时，IE已经知道了所有相关元素的宽度、偏移量等数据了。这在文档首次载入时则不会发生，那时由于自动扩张的特性，宽度还无法确定。这种情况会导致在鼠标经过时页面出现跳变。  
Jump on :hover <http://www.satzansatz.de/cssd/pseudocss.html#hoverjump>  
quirky percentages: the reflow <http://www.positioniseverything.net/explorer/percentages.html>  
  
这些和重排问题相关的 bug 会给百分比边距和补白使用较多的流动布局带来不少麻烦。  
**背景原点**  
MS专有的这个 hasLayout 还会影响背景的定位和扩展。比如，根据 CSS 规范，<http://www.w3.org/TR/CSS21/colors.html#q2background-position>: 0 0 应该指元素的“补白边缘(padding edge)”。而在 IE/Win 下，如果 hasLayout = false 则指的是“边框边缘(border edge)”，当 hasLayout=true 时指的才是补白边缘：  
Background, Border, hasLayout <http://www.brunildo.org/test/BackgroundBorderLayout.html>  
  
**边距重叠**  
hasLayout 会影响一个盒子和其子孙的边距重叠。根据规范，一个盒子如果没有上补白和上边框，那么它的上边距应该和其文档流中的第一个孩子元素的上边距重叠：  
Collapsing Margins <http://www.w3.org/TR/CSS21/box.html#collapsing-margins>  
Uncollapsing Margins <http://complexspiral.com/publications/uncollapsing-margins>  
  
在 IE/Win 中如果这个盒子有 layout 那么这种现象就不会发生了：似乎拥有 layout 会阻止其孩子的边距伸出包含容器之外。此外当 hasLayout = true 时，不论包含容器还是孩子元素，都会有边距计算错误的问题出现。  
Margin collapsing and hasLayout <http://www.brunildo.org/test/IEMarginCollapseLayout.html>  
  
**块级别的链接**  
hasLayout 会影响一个块级别链接的鼠标响应区域(可点击区域)。通常 hasLayout = false 时只有文字覆盖区域才能响应。而 hasLayout = true 则整个块状区域都可响应。添加了 onclick/onmouseover 等事件的任意块级元素也有同样的现象。  
Block anchors and hasLayout <http://www.brunildo.org/test/IEABlock1.html>  
  
**在页面内使用键盘浏览：探索中**  
当使用 tab 在页面中浏览时，如果进入了一个页内链接(in-page link)，那么接下来再按的 tab 键就不会正常继续了：  
hasLayout Property Characterizes IE6 Bug <http://jimthatcher.com/news.htm#haslayout>  
Keyboard Navigation and Internet Explorer <http://juicystudio.com/article/ie-keyboard-navigation.php>  
tab 键会把用户带到(这通常是错误的)其最近的 layout 祖先中的第一个目标(如果这个祖先是由 table， div， span 或某些别的标签构成)。  
  
**收缩包围(shrink-wrapping)现象**  
给已经有 width: auto 的元素添加某些属性会导致它们在计算自身宽度时使用一种收缩包围的算法。比如这些属性 float: left|right, position: absolute|fixed, display: table|table-cell|inline-block|inline-table.  
这些属性造成的现象在IE/Win中也存在，当然这是只对那些它支持的属性而言。但是当一个应该收缩包围的元素中包含一个拥有“layout”的块级元素时，在绝大多数情况下，这个孩子元素的宽度会尽可能地扩展而与其中包含的内容无关，同时也阻止了父元素的收缩包围现象。  
例子： <http://dev.l-c-n.com/IEW2-bugs/shrinkwrap.php>  
一个浮动的纵向导航无序列表并没有收缩包围，因为其中的链接为了消除列表的多余空白bug并扩展可点击区域而拥有了 layout：a {display: block; zoom: 1;}。  
  
这时收缩包围现象只有在以下情况仍然有效：拥有 layout 的孩子元素同时也被赋予了一个特定宽度，或者这个孩子元素本身也是一个具有收缩包围特性的元素，比如浮动元素。  
**边缘裁切**  
通常而言，当一个盒子包含了诸如伸出其边缘的内容这种更复杂的结构时，这个容器就经常需要“hasLayout”来避免一些渲染错误。但使用这种常用方法又会在边界处理时左右为难，因为一个获得“layout”的元素会变成某种自封闭的盒子。  
内部的内容盒子会被裁切，比如使用负边距向外移动时。  
Clipping of negative margined blocks in a hasLayout container <http://dev.l-c-n.com/IEW2-bugs/min-width-clip.php>  
  
被裁掉的部分当内容盒子触发了“layout”时可以再次出现，但在 IE6 中需要同时拥有 position: relative 才行。IE7 在这方面要略有改观，它不再需要额外的 position: relative 了。  
  
  
  
**堆叠，分层和 layout**  
  
  
IE/Win 中似乎有两种分层和堆叠顺序：  
一种是(伪)试图采用CSS的模式：Effect of z-index value to RP and AP blocks<http://www.aplus.co.yu/css/z-pos/>  
还有一种是由“hasLayout”及其孪生兄弟“contenteditable”的行为产生的堆叠顺序。正如在上面相对定位的例子中展现的那样，在 layout 影响下的堆叠现象就好像 Harry Houdini (译者注：魔术师，以纸牌魔术成名)的拿手戏法儿一样。  
  
两种堆叠模式虽互不相容，但却共存于IE的渲染引擎中。经验之谈：调试的时候，两种情况都要考虑到。我们可能会有规律地在下拉菜单或者类似的复杂菜单中看到相关问题，因为它们往往牵涉到堆叠，定位和浮动等诸多令人头疼的问题。给那些 z-index 定位的元素 layout 是一种可能的修正方法，不过也不限于此，这里只是提醒一下。  
混乱的 contenteditable  
如果给一个 HTML 标签设定 contenteditable=true 属性，比如，将会允许对该元素以及其 layout 子元素进行实时的编辑、拖动改变尺寸等操作。你可以把这属性用在浮动元素或者一个有序列表中的 layout 列表元素上看看效果。  
为了对元素进行操作(编辑它们)，“contenteditable”和“hasLayout”为那些 hasLayout 返回 true 的元素引入了一套单独的堆叠顺序。  
Editing Platform [http://msdn.microsoft.com/librar ... mshtmleditplatf.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnmshtml/html/mshtmleditplatf.asp)继承了 layout 概念，对 layout 的误解多是因 contenteditable 而起即可作为证明(那些某种程度上集成了IE编辑引擎的应用软件多暗含着对layout概念的某种强制向后兼容性)。  
More on contenteditable <http://annevankesteren.nl/2005/07/more-contenteditable>  
  
  
  
**和 CSS 规范类似的地方**  
  
  
  
你的 MSIE 页面在别的浏览器中一团糟？我们可没必要让这种事情发生。如果使用恰当，任何好的浏览器都能摆平 MSIE 的页面——只要你使用一些正确的 CSS。  
利用 hasLayout 和“新的块级格式化范围”http://www.w3.org/TR/CSS21/visuren.html#q15之间的细微相似之处，我们可以有几种方法在标准兼容浏览器中重新实现 hasLayout 的“包含浮动元素”http://www.w3.org/TR/CSS21/visudet.html#root-height效果，和一些“浮动元素旁边的元素”http://www.w3.org/TR/CSS21/visuren.html#floats所特有的效果。  
Reverse engineering series <http://www.gunlaug.no/contents/wd_example_01.html>  
Simulations <http://dev.l-c-n.com/IEW/simulations.php>  
  
  
**Quirks 模式**  
  
  
关于这种渲染模式的的信息，请参考我们的 quirks 模式<http://www.satzansatz.de/cssd/quirksmode.html>章节。  
Layout ——结论  
整个 layout 概念和一些基本 CSS 概念是不兼容的，即包含，排列，浮动，定位和层叠等。  
由于页面中元素或有或没有 layout，会导致 IE/Win 的行为和 CSS 规范相违背。  
  
  
  
**拥有 layout ——另外一个引擎？**  
  
  
  
IE 的对象模型看起来是文档模型和他们传统的应用程序模型的糅合。我之所以提到这点是因为它对于理解IE如何渲染页面很重要。而从文档模型切换到应用程序模型的开关就是给一个元素“layout”。  
  
[(Dean Edwards)](http://dean.edwards.name/IE7/notes/#layout)  
  
有时候要解释清楚某种行为是不可能的：就比如 hasLayout，会根据它的状态选择两种不同渲染引擎的一种使用，而且每一种都有其自己的 bug 和怪异之处。  
  
  
  
**不可消除的 bug**  
  
  
  
软件 bug 是由于在制作过程中对完整性和逻辑问题考虑不周等人为错误而导致的。这是人类的固有缺陷，目前还没有什么好的解决方法。  
同样由于这种缺陷，任何试图不重写软件而修复 bug 的做法，都将会不可避免的导致软件中出现更复杂的bug。  
所有依赖别的软件的软件——当然包括依赖操作系统，也会同样依赖他们的 bug。于是我们会从所有关联的软件中得到一连串的 bug，这也更说明找到一个无 bug 软件是几乎不可能的。  
  
[(Molly, the cat‛)](http://www.gunlaug.no/contents/molly_1_15.html)  
  
本文创建于2005年6月30日，最后一次修改于2006年4月2日。  
编者：  
Holly Bergevin <http://positioniseverything.net/>  
Ingo Chao <http://www.satzansatz.de/css.html>  
Bruno Fassino <http://www.brunildo.org/>  
John Gallant <http://positioniseverything.net/>  
Georg Sørtun <http://www.gunlaug.no/>  
Philippe Wittenbergh <http://emps.l-c-n.com/>  
  
特别致谢给予此项目支持的：  
Dean Edwards, and Molly ‚the cat‛  
  
各种语言版本：  
[Original(English)](http://www.satzansatz.de/cssd/onhavinglayout.html)  
[Brazilian Portuguese](http://www.maujor.com/tutorial/haslayout.php) by          [Mauricio Samy Silva](http://www.maujor.com/)  
[中文版本](http://old9.blogsome.com/2006/04/11/onhavinglayout/) by [old9](http://old9.blogsome.com/)  
[Italian](http://gabrieleromanato.altervista.org/css/onhavinglayout.html) by [Gabriele Romanato](http://gabrieleromanato.altervista.org/)  
  
相关讨论：  
[dean.edwards.name/weblog/](http://dean.edwards.name/weblog/2005/06/layout/)  
  
[url=mailto:spam.layout@satzansatz.de]联系作者：[/url]  
  
版权说明：  
[本文基于创作共用协议发布。](http://creativecommons.org/licenses/by-nc-sa/2.0/)  
  
  
  
**目录**  
  
  
  
介绍  
hasLayout —— 定义  
术语  
问题种种  
Layout 从何而来  
默认 layout 元素  
属性  
有关内联级别元素  
脚本属性 hasLayout  
CSS hacks  
Hack整理  
关于 IE Mac 的小问题  
MSDN文档  
分析  
各种情况的详细说明  
清除浮动和自动扩展适应高度  
浮动元素旁边的元素  
列表  
表格  
相对定位元素(r.p.)  
绝对定位元素(a.p.)：包含区块，什么是包含区块？  
滤镜  
对已渲染元素的重排(re-flow)  
背景原点  
边距重叠  
块级别的链接  
在页面内使用键盘浏览：探索中  
收缩包围(shrink-wrapping)现象  
边缘裁切  
堆叠，分层和 layout  
混乱的 contenteditable  
和 CSS 规范类似的地方  
Quirks 模式  
Layout —— 结论  
拥有 layout —— 另外一个引擎？  
不可消除的 bug