# haslayout

## 关于 haslayout ？

##### 什么是haslayout

“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，这有点类似于一个窗体的概念。微软的开发者们认为盒状元素(box-type elements)应该具有一个“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 layout , 也就是 hasLayout。hasLayout 其实既不是一个属性更不是一个行为，而是 IE 这个渲染引擎代代继承一贯拥有的一个渲染概念，在这个概念下渲染的元素将具有一种特性。实际上这种渲染特性在有些 HTML 元素中与身俱来，而在另外一些元素中也可以通过一些 CSS 属性将其触发为 true ，且一旦触发将不可逆转。

　　当我们说一个元素“拥有layout”或“得到layout”，或者说一个元素“has layout” 的时候，我们的意思是指它的微软专有属性 hasLayout 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。IE Developer Toobar 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。另外可以用javascript获取hasLayout的状态，<script>alert(my.currentStyle.hasLayout);</script>，但是不可以设置。

　　另外一个需要注意的是“layout”会影响javascript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。

##### 什么情况下hasLayout会出现？

实际上一般情况都不会出现的，当然除了使用下面默认具有hasLayout的元素或使用特定样式触发hasLayout以外；它会带来各种诡异表现，当你发现IE6 IE7出现了一些不可思议的问题，首先要检查的就是是否是hasLayout在捣鬼；hasLayout只出现在IE7及更早版本中，IE8之后不存在hasLayout解析模式。

## 默认具有 haslayout 的元素（不完全列表）

　　\* html, body  
　　\* table, tr, th, td  
　　\* img  
　　\* hr  
　　\* input, button, file, select, textarea, fieldset  
        \* legend  
　　\* marquee  
　　\* frameset, frame, iframe  
　　\* objects, applets, embed  
　　对于并非所有的元素都默认有布局，微软给出的主要原因是“性能和简洁”。如果所有的元素都默认有布局，会对性能和内存使用上产生有害的影响。

## 如何激发 haslayout？

　　大部分的 IE 显示错误，都可以通过激发元素的 haslayout 属性来修正。可以通过设置 css 尺寸属性(width/height)等来激发元素的 haslayout，使其“拥有布局”。如下所示，通过设置以下 css 属性即可。  
　　\* display: inline-block  
　　\* height: (任何值除了auto) 通常用 \_height:1%;解决IE6的问题，height:1%不会改变实际高度  
　　\* float: (left 或 right)  
　　\* position: absolute  
　　\* width: (任何值除了auto)  
　　\* writing-mode: tb-rl  
　　\* zoom: (除 normal 外任意值)

## IE7 还有一些额外的属性(不完全列表):

　　\* min-height: (任意值)  
　　\* max-height: (除 none 外任意值)  
　　\* min-width: (任意值)  
　　\* max-width: (除 none 外任意值)  
　　\* overflow: (除 visible 外任意值)  
　　\* overflow-x: (除 visible 外任意值)  
　　\* overflow-y: (除 visible 外任意值)  
　　\* position: fixed  
　　其中 overflow-x 和 overflow-y 是 css3 盒模型中的属性，目前还未被浏览器广泛支持。

　　对于内联元素(默认即为内联的元素，如 span，或 display:inline; 的元素)，width 和 height 只在 IE5.x 下和 IE6 或更新版本的 怪异 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 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 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 一旦内联元素拥有 layout 还会变成 inline-block。

## haslayout 问题的调试与解决

　　当网页在 IE 中有异常表现时，可以尝试激发 haslayout 来看看是不是问题所在。常用的方法是给某元素 css 设定 zoom:1。使用 zoom:1 是因为大多数情况下，它能在不影响现有环境的条件下激发元素的 haslayout。而一旦问题消失，那基本上就可以判断是haslayout 的原因。然后就可以通过设定相应的 css 属性来对这个问题进行修正了。建议首先要考虑的是设定元素的width/height 属性，其次再考虑其他属性。

　　对 IE6 及更早版本来说，常用的方法被称为霍莉破解(Holly hack)，即设定这个元素的高度为 1%(height:1%;)。需要注意的是，当这个元素的 overflow 属性被设置为 visible 时，这个方法就失效了。或者使用 IE的条件注释。

　　对 IE7 来说，最好的方法时设置元素的最小高度为 0 (min-height:0;)。

## haslayout 问题引起的常见 bug

　　E6 的躲躲猫(peek-a-boo) bug

　　bug 修复: \_height:1%;

## hasLayout的触发条件：

* position: absolute(IE5+)
* float: left|right(IE5+)
* display: inline-block(IE5+)
* width: "auto"以外的任何值(IE5+; 对inline元素无效)
* height: "auto"以外的任何值(IE5+; 对inline元素无效)
* zoom: "normal"以外的任何值(IE5.5+; IE私有属性)
* writing-mode: tb-rl(IE5+; IE私有属性)
* overflow: hidden|scroll|auto(IE7; 此属性在IE6及更早版本中不能应用在未触发hasLayout的元素上)
* overflow-x|-y: hidden|scroll|auto(IE7; 此属性在IE6及更早版本中不触发hasLayout; 此属性在CSS3中才获支持)
* position: fixed(IE7)
* min-width: 任何值(IE7; 即使是0)
* max-width: "none"以外的任何值(IE7)
* min-height: 任何值(IE7)
* max-height: "none"以外的任何值 (IE7)
* position: fixed(IE7)

## 清除或重置hasLayout：

* position: static(IE5+)
* float: none(IE5+)
* display: "inline-block"以外的任何值(IE5+)
* width|height: "auto"(IE5+; 对inline元素无效)
* zoom: "normal"(IE5.5+; IE私有属性)
* writing-mode: 从'tb-rl'到'lr-tb'(IE5+; IE私有属性)
* max-width|max-height: "none"(IE7)
* overflow: visible(IE7)

## 实际操作中

zoom:1; 是很不错的触发方法，不会改变原来的任何式样，而且仅仅是IE可以识别，但是唯一的坏处就是他不能通过W3C。

设置 display:inline-block 然后再设置回原始的 display 属性,这样不会移除 layout，我们就可以达到设置 layout 而不使用IE的条件注释的目的。下面是例子：

1. div { **display**: inline-block; }
2. div { **display**: block; } /\* 分别在两段 css 块中设置 \*/

## hasLayout都会引发什么问题？

##### 浮动元素会被layout元素自动包含。

**正常情况下，浮动元素会按照left和top的设置偏离原来文档流中的位置，父元素是不会调整高宽去包含该浮动元素的（也就解释了为什么浮动元素不能撑开父容器），但在IE中，layout元素会自动调整高和宽以包含浮动元素（给父容器\_height:1%;即可解决）**[**http://blog.csdn.net/hedong37518585/article/details/6639731**](http://blog.csdn.net/hedong37518585/article/details/6639731)

##### 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，其中的文字内容应该沿着浮动元素的右边顺序排列并会滑到浮动元素下方。但是如果这个块级元素有 layout，那么这个元素就会表现为一个矩形，其中文字不会滑向浮动元素下方。**

1. .box{
2. width:220px;
3. overflow:auto;
4. font-size:12px;
5. }
6. .leftbox{
7. background:#CCC;
8. width:100px;
9. height:100px;
10. float:left;
11. \*margin-right:-3px; /\*针对IE6 浮动元素水平右外边距移动-3px 即可解决\*/
12. }
13. .textbox{
14. background:#FFCCCC;
15. height:1%;/\*去掉后 IE6下 文字不会再围绕浮动\*/
16. }
18. **<div** class="box"**>**
19. **<div** class="leftbox"**>**浮动元素**</div>**
20. **<p** class="textbox"**>**文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本**</p>**
21. **</div>**

##### IE专有的滤镜属性filter是只适用于 layout 元素的

**也就是说如果你给一个DIV设置透明用的是filter:alpha(opacity=80);如果你没有让DIV触发hasLayout，那么这个透明将无效**

1. .textbox{
2. background:#FFCCCC;
3. opacity:0.8;
4. filter:alpha(opacity=80);
5. zoom:1;
6. }
8. **<div** class="textbox"**>**文本文本文本文本文本文本文本文本文本本文本文本文本文本文本文本文本文本文本文文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本**</div>**

##### 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。**

1. a{
2. display:block;
3. height:1%;/\*去掉后就不具有Layout布局 则只有文字覆盖区域以响应\*/
4. }