=

[居中控制 1](#_Toc6493)

[问题来了 1](#_Toc9457)

[同高 垂直居中单行元素 1](#_Toc14553)

[同高垂直居中单行元素 1](#_Toc21603)

[100% height、inline-block元素辅助 垂直居中单行元素 1](#_Toc19059)

[实现原理 1](#_Toc25321)

[:after 优化 2](#_Toc27521)

[其他表现形式 2](#_Toc22015)

[其他 2](#_Toc23331)

[tall-cell居中多行文本 2](#_Toc18826)

[上下padding实现多行在不定高元素中的居中 3](#_Toc25039)

[某table-cell元素 居中于 外部table元素 3](#_Toc6086)

[IE 3](#_Toc15868)

[table-cell inline-block实现水平垂直居中 3](#_Toc26036)

[垂直：外层元素table-cell控制内层元素居中 4](#_Toc1906)

[水平：内层div水平居中显示 4](#_Toc24507)

[垂直+水平(static|relative) 4](#_Toc17862)

[absolute\*relative 4](#_Toc4258)

[table-cell 4](#_Toc7178)

[关于table-cell 4](#_Toc19724)

[与大小不固定元素的垂直居中 5](#_Toc15042)

[与两栏自适应布局 5](#_Toc16412)

[等高布局 5](#_Toc4749)

[列表布局 5](#_Toc18922)

[负margin(transform)绝对居中布局 6](#_Toc21811)

[0000 居中 6](#_Toc9966)

[实现居中 6](#_Toc32746)

[可视居中 7](#_Toc20375)

[偏移|靠边 7](#_Toc16668)

[响应式 7](#_Toc14746)

[带溢出内容 7](#_Toc23789)

[大小可调整 7](#_Toc12481)

[图像 7](#_Toc3916)

[可变高度 7](#_Toc27466)

[其他居中方式 7](#_Toc8035)

[margin auto水平居中 7](#_Toc24078)

[盒布局：box 8](#_Toc14232)

[视口绝对居中vh|vw 8](#_Toc14088)

[50%高、float元素辅助 8](#_Toc28286)

[JS + CSS控制:不漂浮（适用于做登陆页面） 8](#_Toc22776)

[JQ实现 8](#_Toc4200)

[cacl 9](#_Toc18737)

[flex居中 9](#_Toc30331)

[未知元素居中1:table-cell布局 9](#_Toc18651)

[未知元素居中2:inline-block布局 9](#_Toc24437)

[图片列水平垂直居中控制 10](#_Toc23847)

[inline-block居中 10](#_Toc8065)

[background center居中 10](#_Toc26188)

[table-cell居中 10](#_Toc26576)

[图片水平垂直居中4-[最精简的实现图片水平垂直居中显示的方法] 10](#_Toc9031)

[inline block图片辅助居中 11](#_Toc11213)

[inline block空白辅助居中 12](#_Toc12461)

[如何选择居中方式 12](#_Toc6439)

## 居中控制

问题来了

元素A中包含B元素，而且B元素有可能是一行文本、多行文本，图片（未知大小），如何让B元素在A元素中实现水平垂直居中效果.

周刊5# 居中之美:

<http://www.w3cplus.com/collective-5.html>

大小不固定的图片和多行文字的垂直水平居中：

<http://www.zhangxinxu.com/study/200908/img-text-vertical-align.html>

Centering in CSS: A Complete Guide

<https://css-tricks.com/centering-css-complete-guide/>

同高 垂直居中单行元素

|  |  |
| --- | --- |
| 同高垂直居中单行元素 | <**style** type="text/css">  .vertical {  height: 100px;  line-height: 100px;}  .vertical {  height: 3em;  line-height: 3em;  } </**style**> <**div** class="vertical">content</**div**> |
| line-height设置为盒子的height可以实现单行文字的垂直居中：   1. 优点：   适合在所有浏览器，没有足够空间时，内容不会被切掉；  这种方法用来实现单行垂直居中是相当的简单的，你只要保证元素内容是单行，并且其高度是固定不变的，你只要将其“line-height”设置成和“height”值一样就Ok了。   1. 缺点：   不过这种方法局限性太大：  仅适合应用在单行文本和图片上，对小元素上是非常有用的，比如说让一个button、图片或者单行文本字段。  不针对多行元素。 |

100% height、inline-block元素辅助 垂直居中单行元素

|  |  |  |
| --- | --- | --- |
| 实现原理 | <**style** type="text/css">  **/\*\*/  html**,**body**{  height: 100%;  }  **/\*定义高度，让线盒型div#extra有一个参照物，可以是固定值，也可以是百分比\*/** #parent {  width: 500px; text-align: center;**/\*居中\*/** height: 300px;  }  #vertically\_center,#extra {  display: inline-block;  vertical-align: middle;  }  **/\*若不需要水平居中：建议写在后面\*/** #extra {  height: 100%;  } </**style**>  **<!--[if lt IE 8]> <style type="text/css">  /\*IE6-7不支持display:inline-block，所以在ie6-7另外写一个hack，用来支持ie6-7\*/  #vertically\_center,#extra {  display: inline;  zoom: 1;  }  #extra {  width: 1px;  } </style> <![endif]-->** <**div** id="parent">  <**div** id="vertically\_center">  <**p**>I am vertically centered!I am vertically centered!</**p**>  </**div**>  <**div** id="extra">**<!-- ie comment -->**</**div**> </**div**> | |
| 优点：  可以自适应高度，简单易懂  缺点：  需要给元素的父元素设置一个高度，这个高度可以是一个固定值或者百分值高度，另外需要增加一个额外的标签，当作线盒型，如div#extra，并且需要设置其高度为100%。另外就是ie6-7不支持display:inline-block，需要给他们另外写一个样式。  这是一个很有意思的方法，但你要懂得如何使用display。有关于这种方法的操作，大家可以去看一看Jonathan Potter写的《[CSS, Vertical Centering](http://vivifystudios.com/articles-css-vertical-centering/" \t "http://www.w3cplus.com/css/_blank)》。 | |
| :after 优化 | <**style**>  #outDiv{  text-align: center;**/\*水平居中\*/** height: 500px;  width: 500px;  }  **/\*  1+2创建一个辅助行内元素，  3行内元素的高度设置为外盒的高度，  4通过vertical-align控制此行元素 垂直居中 对齐  \*/** #outDiv:before{  content: **''**;  display: inline-block;  height: 100%;  vertical-align: middle;  } </**style**> <**div** id="outDiv">  DIV居中演示 </**div**> | |
|  | |
| 实现[一层结构][单行文本]的上下左右居中显示  实现方式有点类似于line-height ==> height  这种方法是采用的display:inline-block，然后借助另一个元素的高度来实现居中  单行文本可以置换为 | |
| 其他表现形式 | <**style**>  #outDiv {  text-align: center;height: 500px;  width: 500px;  }  #outDiv:before {  content: **''**;  display: inline-block; height: 100%; vertical-align: middle; }  #inDiv {  display: inline;  } </**style**> <**div** id="outDiv">  <**div** id="inDiv">  单行文本的水平居中  单行文本的水平居中  单行文本的水平居中  单行文本的水平居中  单行文本的水平居中  </**div**> </**div**> | <**style**>  #outDiv {  text-align: center; height: 500px;  width: 500px;  }  #outDiv:before {  content: **''**;  display: inline-block; height: 100%; vertical-align: middle; }  #inDiv {  } </**style**> <**div** id="outDiv">  <**span** id="inDiv">  单行文本的水平居中  单行文本的水平居中  单行文本的水平居中  单行文本的水平居中  单行文本的水平居中  </**span**> </**div**> |
|  |  |

其他

|  |  |
| --- | --- |
| tall-cell居中多行文本 | <**style**>  **/\*关注此处关于em的应用\*/  /\*外部div标签：\*/  div**.outDiv {  display: table-cell;  vertical-align: middle;  text-align: center;  font-size: 10em;**/\* --> 160px\*/** width: 2.5em;**/\* 160px \* 2.5 = 400px\*/** height: 1em;**/\* 160px\*/** border: 4px solid #beceeb;  }  **/\*内部span标签：\*/  span**.inSpan {  display: inline-block;  width: 300px;  font-size: 0.1em;**/\* 10em \* 0.1 = 1em --> 16px\*/** } </**style**> <**div** class="outDiv">  <**span** class="inSpan">多行文本的垂直居中，多行文本的垂直居中，多行文本的垂直居中，多行文本的垂直居中，多行文本的垂直居中，</**span**> </**div**>    1、包裹div高度固定  要实现高度不固定的文字垂直居中使用padding就好了。  2、包裹div高度不固定  实现的关键是把文字当图片处理。用一个span标签将所有的文字封装起来，设置文字与图片相同的display属性（inline-block属性），然后用处理图片垂直居中的方式处理文字的垂直居中即可。  有几点简要说明： 1.此例子用em做单位，如果您对em单位了解不够，把握不来的话，可以使用px做单位，值要换； 2.外部div不能使用浮动； 3.外部div高度和文字大小比例1.14为宜； 4.内部标签的vertical-align:middle可以省略，但是外部div高度和文字大小比例要修改，我自己试了一下，高度比字体1.5左右的样子； 5.系统原因，我没能够在IE8下测试。  ---文档类型不能是：<!DOCTYPE html> |
|  | 对于高度固定的div，里面文字单行或多行显示，字体大小有大有小的情况；可以借助line-height：  line boxes的高度取决于它的inline元素的最高高度。而这个高度由一个不占据任何空间的空格完成，方法即使设置font-size为0，line-height为所需要的高度。同时，我们为了分隔line boxes，同时要保持在一行上，需要设置display属性为inline-block。如下代码[注意span和i标签间不能有空格!]：  <**style**>  .mulit\_line {  line-height: 150px;  border: 1px dashed #cccccc;  padding-left: 5px;  }   .mulit\_line **span** {  display: -moz-inline-stack;  display: inline-block;  line-height: 16px;  vertical-align: middle;  }   .mulit\_line **i** {  width: 0;  display: -moz-inline-stack;  display: inline-block;  vertical-align: middle;  font-size: 0;  } </**style**> <**p** class="mulit\_line">  <**span** style="font-size:12px;">这里是高度为150像素的标签内的多行文字，文字大小为12像素。<**br**/>这里是第二行，用来测试多行的显示效果。</**span**><**i**>&nbsp;</**i**> </**p**> |
| 上下padding实现多行在不定高元素中的居中 | <**style**>  .context {  padding-top: 30px;  padding-bottom: 30px;  text-align: center;**/\*也可实现居中\*/** }  .outDiv{  text-align: center;**/\*可实现居中\*/** } </**style**> <**div** class="outDiv">  <**div** class="context">test</**div**> </**div**>  给出上下一样的padding值：   1. 优点：   在所有浏览器下能正常工作，支持所有元素，简单易懂，结构清晰  若添加左右padding则可实现水平居中   1. 缺点：   使用这种方法不能给容器固定高度，如果固定高度将无法达到效果。 |

某table-cell元素 居中于 外部table元素

|  |  |
| --- | --- |
| <**style**>  #table{  display: table;**/\*让元素以表格形式渲染\*/** width:500px;  height: 300px;  text-align: center;  }  #table-cell{  display:table-cell;**/\*让元素以表格的单元素格形式渲染\*/** vertical-align: middle;**/\*使用元素的垂直对齐\*/** } </**style**> <**div** id="table">  <**div** id="table-cell">content</**div**> </**div**> |  |
| 优点：  内容层元素没有有高度的限制，此方法可以要据元素内容动态的改变高度，从而也就没有空间的限制，元素的内容不会因为没足够的空间而被切断或者出现难看的滚动条。  缺点：  不足之处呢？这种方法只适合现代浏览器，在IE6-7下无法正常运行；而且较前两者而言，结构也更复杂。  主要缺陷是会产生额外的标签，每一个需要居中的元素需要三个额外的HTML标签。  这种方法在现代浏览器下是非常方便。但是在IE6-7中是不被支持的，因为display:table在IE6-7中不被支持，那么为了解决这种方法在IE6-7的兼容，需要额外增加一个div，并使用hack，下面我们一起来看看其解决办法。 | |
| <**style**>  .table {  height: 300px;**/\*高度值不能少\*/** width: 300px;**/\*宽度值不能少\*/** display: table;  position: relative;  float:left;  }   .tableCell {  display: table-cell;  vertical-align: middle;  text-align: center;  padding: 10px;  \*position: absolute;  \*top: 50%;  \*left: 50%;  }  .content {  \*position:relative;  \*top: -50%;  \*left: -50%;  } </**style**> <**div** class="table">  <**div** class="tableCell">  <**div** class="content">content</**div**>  </**div**> </**div**> | |
| IE | |
| 这个方法我们需要一个线盒型。  用来实现IE下的效果，需要增加一上行内标签比如说“span”（此处最好使用行内标签，千万不要使用块标签，因为使用块标签会没有效果）。 | |
| <**style**>  .table {  display: table;  height: 200px;  width: 200px;  text-align: center;  }  .tableCell {  display: table-cell;  vertical-align: middle;  } </**style**> **<!--[if lte ie 7]> <style type="text/css">  .tableCell {  display: inline-block;  }  b {  display: inline-block;  height: 100%;  vertical-align: middle;  width: 1px;  } </style> <![endif]-->** <**p** class="table">  <**span** class="tableCell">Centering multiple lines <**br**>in a block container.</**span**>  **<!--[if lte IE 7]><b></b><![endif]-->** </**p**> | |
| 1. 优点：   这种方法的优点是不会有高度的限制。   1. 缺点：   不好的地方就是这种方法为了让IE运行正常，需要额外增加一些标签，另外就是线盒型的标签种类有限制。不过用起来还是蛮方便的。 | |

table-cell inline-block实现水平垂直居中

|  |  |  |  |
| --- | --- | --- | --- |
| <style>  /\* 1+2: 用于实现内部元素的垂直居中\*/  /\* 3: 单独使用并不能实现div.center-in的居中，可实现其中内容的居中；结合inline-block可以实现div.center-in的居中\*/  .table-cell-out {  display: table-cell;/\*1\*/  vertical-align: middle;/\*2\*/  text-align: center; /\*3\*/  height: 200px;  width: 200px;  }  /\*内部容器水平居中方式1：有长度div margin auto居中实现\*/  .center-in-1 {  width: 50%;  margin: auto;  }  /\*内部容器水平居中方式2：inline-block元素可以通过外层的text-align来实现居中\*/  .center-in-2{  width: 50%;  display: inline-block;  } </style> <div class="table-cell-out">  <div class="center-in-1">  阿斯蒂芬  </div> </div> | | | .table与.table-cell长宽相互叠影  长度：.table优先  宽度：取大  table-cell的宽度受外部元素宽度的限制 |
| 垂直：外层元素table-cell控制内层元素居中 | <**style**>  **div** {  outline: 1px solid red;  }  #outDiv {  width: 500px;  height: 500px;  display: table-cell;vertical-align: middle; }   #inDiv {  position: relative|static;  width: 200px;  height: 200px;  } </**style**> <**div** id="outDiv">  <**div** id="inDiv">  单行文本的水平居中  单行文本的水平居中  单行文本的水平居中  单行文本的水平居中  单行文本的水平居中  </**div**> </**div**> | outdiv、indiv 非绝对定位  实现内部元素的垂直居中 | |
| <**style**>  #outDiv {  width: 500px;  height: 500px;  display: table-cell; vertical-align: middle; }  #inDiv {  position: absolute;  width: 200px;  height: 200px;  } </**style**> <**div** id="outDiv">  <**div** id="inDiv">  单行文本的水平居中  单行文本的水平居中  单行文本的水平居中  单行文本的水平居中  单行文本的水平居中  </**div**> </**div**> |  | |
| 内部元素绝对定位(absolute)意味着定位点为左上角(left|top)； | | |
| 水平：内层div水平居中显示 | <**style**>  #outDiv{  height: 400px;  width: 500px;  text-align: center;  }  #inDiv {  display: inline-block;  height: 200px;  width: 200px;  } </**style**> <**div** id="outDiv">  <**div** id="inDiv">  DIV居中演示  </**div**> </**div**> |  | |
| 垂直+水平(static|relative) | <**style**>  #outDiv {  width: 500px;  height: 500px;  text-align: center;  display: table-cell; vertical-align: middle; }   #inDiv {  width: 200px;  height: 200px;  display: inline-block;  }</**style**> <**div** id="outDiv">  <**div** id="inDiv">  单行文本的水平居中  单行文本的水平居中  单行文本的水平居中  单行文本的水平居中  单行文本的水平居中  </**div**> </**div**> | 水平居中：  out: text-align:center;  in: display:inline-block;  垂直居中：  out:  display:table-cell;  vertical-align:middle; | |
| absolute\*relative | out absolute  in absolute | out absolute  in relative | |
| out relative  in absolute | out relative  in relative | |
|  | 外层绝对定位将破坏table-cell，是居中无效；  内层绝对定位意味着其不会影响后续元素的位置，其具体位置将通过左上角实现；若未对top、left进行赋值； | | |

table-cell

|  |  |
| --- | --- |
|  | <http://www.zhangxinxu.com/wordpress/2010/10/%E6%88%91%E6%89%80%E7%9F%A5%E9%81%93%E7%9A%84%E5%87%A0%E7%A7%8Ddisplaytable-cell%E7%9A%84%E5%BA%94%E7%94%A8/> |
| 关于table-cell | display:table-cell属性指让标签元素以表格单元格的形式呈现，类似于td标签。目前IE8+以及其他现代浏览器都是支持此属性的，但是IE6/7只能对你说sorry了，这一事实也是大大制约了display:table-cell属性在实际项目中的应用（写于2016年8月24日：时代在变，当下，是时候登上舞台了）。  我们都知道，单元格有一些比较特别的属性，例如元素的垂直居中对齐，关联伸缩等，所以display:table-cell还是有不少潜在的使用价值的，虽说IE6/7不支持此属性，但是幸运的是，IE6/7一些乱糟糟的属性与渲染，我们可以其他方法实现同样或是类似的效果。  与其他一些display属性类似，table-cell同样会被其他一些CSS属性破坏，例如float, position:absolute，所以，在使用display:table-cell与float:left或是position:absolute属性尽量不用同用。  设置了display:table-cell的元素对宽度高度敏感，对margin值无反应，响应padding属性，基本上就是活脱脱的一个td标签元素了。 |
| 与大小不固定元素的垂直居中 | <**style**>  **div** {  display: table-cell;  width: 1em;  height: 1em;  border: 1px solid #beceeb;  font-size: 144px;  text-align: center;  vertical-align: middle;  }   **div img** {  vertical-align: middle;  } </**style**> <**div**><**img** src="images/1.jpg" alt=""/></**div**> <**div**><**img** src="images/2.jpg" alt=""/></**div**> <**div**><**img** src="images/3.jpg" alt=""/></**div**> <**div**><**img** src="images/4.jpg" alt=""/></**div**> <**div**><**img** src="images/4.jpg" alt=""/></**div**> <**div**><**img** src="images/4.jpg" alt=""/></**div**> <**div**><**img** src="images/4.jpg" alt=""/></**div**> |
| ---同级别的table-cell元素不会换行；  ---内部元素（img）垂直居中 |
| 与两栏自适应布局 | <**style**>  **@import "http://www.zhangxinxu.com/study/css/common.css"**;  .box {  width: 60%;  margin: 60px auto 0;  padding: 20px;  background: #f5f5f5;  }  .content {  display: table-cell;  \*display: inline-block;  } </**style**> <**div** class="box f9 fix">  <**a** href="#prettyGirl" class="l mr10"><**img** border="0" src="http://image.zhangxinxu.com/image/study/s/s128/mm2.jpg" /></**a**>  <**div** class="content">  <**p**><**a** href="#prettyGirl">大美女一枚</**a**> 来自上海</**p**>  <**p** class="mt5">签名：想找个保鲜盒把你给我的那些感动都装起来。当你让我伤心的时候就拿出来回味一下。</**p**>  <**p** class="mt5">微博：坐在办公室，只听轰隆隆几声巨响，晴天也能打雷吗？原来街对面的芭莎咖啡厅被炸成了两截。这定点爆破也太失败了，也不清下场，把路过的汽车震得灰头土脸，愣在路中央不知如何是好。其次，房子只炸了一半，另一半屹立不倒，是乍药太水还是房子质量太好？</**p**>  </**div**> </**div**> |
|  | 本例中，左侧为头像，右侧内容自适应。其中头像部分使用了float属性，左浮动，IE8+以及Firefox、Chrome、Opera等现代浏览器右侧使用了display:table-cell属性，结果就自适应了，很简单的代码，很神奇的效果。  对于不认识display:table-cell属性的IE6/7呢？哦呵呵，很简单，使用display:inline-block属性代替display:table-cell就完全ok的啦！  原因在于：IE6/7下block属性的元素对inline-block属性是有反应，但是却不是纯洁的反应，而是怪蜀黍看到粉嫩小萝莉的一点邪念，就是让元素有个怪异的haslayout属性。//zxx:大家似乎都喜欢用haslayout解析一些老IE下的一些怪异现象，但我自己打心底里是不认同这个概念。  如果IE6/7是很标准的纯洁的解释inline-block属性的话，是无法实现自适应的，右侧的文字描述内容会跑到头像的下面，哦呵呵~~有点负负得正，以毒攻毒的意味。代码如下：  display:table-cell; \*display:inline-block;  就万事大吉，收工回家了。  在本例demo中，右侧内容足够多，所以宽度完整的撑开了，如果内容有限，则宽度就是内容的宽度，此时想要让某个元素（例如关闭按钮）右侧定位就会有问题，解决方法就是定义一个非常宽的宽度，就像上面facebook截图中的CSS属性一样，所以，考虑到各种情况，更健壮耐用的CSS代码应如下：  display:table-cell; \*display:inline-block; width:2000px; \*width:auto;  或者使用：  display:table-cell; width:2000px; \*width:auto; \*zoom:1;  这种两栏的自适应布局，不仅不要分别丈量与计算两列的宽度，连“[页面重构鑫三无准则 之无宽度准则](http://www.zhangxinxu.com/wordpress/?p=1152)”中absolute自适应布局的头像宽度都不需要亮了，可以说是更加懒惰，更加直接的好方法。 |
| 等高布局 | table表格中的单元格最大的特点之一就是同一行列表元素都等高。所以，很多时候，我们需要等高布局的时候，就可以借助display:table-cell属性。说到table-cell的布局，不得不说一下“匿名表格元素创建规则”：  CSS2.1表格模型中的元素，可能不会全部包含在除HTML之外的文档语言中。这时，那些“丢失”的元素会被模拟出来，从而使得表格模型能够正常工作。所有的表格元素将会自动在自身周围生成所需的匿名table对象，使其符合table/inline-table、table-row、table- cell的三层嵌套关系。  举个例子吧，如果我们为元素使用“display:table-cell;”属性，而不将其父容器设置为“display:table-row;”属性，浏览器会默认创建出一个表格行，就好像文档中真的存在一个被声明的表格行一样。如果您还不是很理解，可见参见支付宝UED的“[基于display:table的CSS布局](http://ued.alipay.com/wd/2008/10/29/%E5%9F%BA%E4%BA%8Edisplaytable%E7%9A%84css%E5%B8%83%E5%B1%80/" \t "http://www.zhangxinxu.com/wordpress/2010/10/%E6%88%91%E6%89%80%E7%9F%A5%E9%81%93%E7%9A%84%E5%87%A0%E7%A7%8Ddisplaytable-cell%E7%9A%84%E5%BA%94%E7%94%A8/_blank)”一文。  实现等高布局，毫无疑问，display:table-cell是首选，这就好比鼹鼠，生下来就是为了打洞用的。考虑到匿名创建表格元素的问题，所有table-cell元素外一定要留有一个用来包裹的标签。于是，我们有类似下面的CSS代码： |
|  | <**style**>  .list\_row {  display: table-row;**/\*+++\*/** }  .list\_cell {  display: table-cell;**/\*+++\*/** width: 30%;  padding: 1.6%;  background-color: #f5f5f5;  outline: 1px solid red;  }  .list\_center {  width: 50%;**/\*中间加长\*/** } </**style**>  <**div** class="list\_row">  <**div** class="list\_cell">你一定也有过这种感觉的。...罗兰《寂寞的感觉》你一定也有过这种感觉的。...罗兰《寂寞的感觉》</**div**>  <**div** class="list\_cell list\_center">作为一个被基阿异捅过...水，四积阴功五读书。</**div**>  <**div** class="list\_cell">奔波了一...永远幸福快乐！</**div**> </**div**> |
|  | ---在30% 50% 30%的比例划分下表现为：300px+500px+200px;  ---对于不支持display:table-cell属性的IE6/7浏览器，我们可以使用“补差等高法”，就是一个一个很大的margin-bottom负值配上一个同样大小的padding-bottom值，本例中为了实例需要，margin-bottom值就百来像素。显然，由于两者原理不同，难免需要用到hack，所以demo完整CSS代码如下;认识display:table-cell属性的元素对margin属性不敏感，所以上面margin-bottom属性前没有加\*号。 |
|  | <**style**>  .list\_row {  display: table-row;  overflow: hidden;  }   .list\_cell {  display: table-cell;  width: 30%;  margin-bottom: -100px;  padding: 1.6%;  \*padding-bottom: 110px;  background-color: #f5f5f5;  \*float: left;  } </**style**> |
| 列表布局 | 一般这类布局都是使用浮动的。但是浮动布局的不足在于：一是需要清除浮动造成影响；二是不支持不定高列表的浮动。替代浮动布局的方法是有的，如果深究细节以及一些思想，方法还不少。其中有一个方法就是使用display:table-cell。  当然，说句心里话，将display:table-cell属性用在列表元素布局上，有点类似于张飞绣衣服，大马拉小车，优势并没有多大发挥，但是，毕竟也算浮动布局的一个备用替换方案。  我的下一篇文章将会详细讲解浮动布局的替换方案，其中table-cell方法可以说是里面最不好的一个方法，本着过渡和热身的目的，这里只简单讲过。  首先是效果：[display:table-cell下的列表布局](http://www.zhangxinxu.com/study/201010/table-cell-list-flow-layout.html" \t "http://www.zhangxinxu.com/wordpress/2010/10/%E6%88%91%E6%89%80%E7%9F%A5%E9%81%93%E7%9A%84%E5%87%A0%E7%A7%8Ddisplaytable-cell%E7%9A%84%E5%BA%94%E7%94%A8/_blank)  可以看到即使模块高度不一致，也不会产生浮动布局可能产生的错位。  由于table-cell需要每行包裹一个独立的标签，所以，在后台repeat输出的时候，需要两次循环，而是列数是限死的（与浮动布局一样）。对于简单的列表，使用display:table-cell是很难看到什么优势的，但是，如果列表复杂，数据多，内容细，display:table-cell可能会像大S订婚的传闻那样让人震惊。  display:table-cell下的列表布局最适用的场景是：列表个数不固定，但是，无论列表几个，都平分容器空间。什么意思呢？就是如果4个列表，希望每个宽度25%，3个就33.3333%，2个列表希望每个宽度50%。此时，没有比display:table-cell更合适的技术了。(单行列表布局)  父级设置display:table同时宽度为容器宽度，或者直接width:100%，此时，display:table-cell子元素就会自动等分。 |
|  |  |
| 负margin(transform)绝对居中布局 | |
| <**style**>  **/\*absolute DOM 相对于首个非static父元素进行定位\*/** #outDiv {  position: absolute;**/\*absolute relative均可\*//\*默认值为static\*/** height: 600px;  width: 500px;  }  **/\*1:建议写为absolute (若outDiv内部无其他元素，可写为relative，(relative元素位置承上启下))；且需保证外层元素非static\*/** #inDiv {  position: absolute;**/\*1\*/** height: 200px;  width: 200px;  top: 50%;  left: 50%;  margin: -100px 0 0 -100px;  } </**style**> <**div** id="outDiv">  asdfasdf asdfasdf asdfasdf asdfasdf asdfasdf asdfasdf asdfasdf asdfasdf asdfasdf asdfasdf  <**div** id="inDiv">  DIV居中演示  DIV居中演示  </**div**> </**div**> | |
|  | |
| 1. 总结：   如果知道了各个元素的大小，设置等于宽高一半大小的负margin值（如果没有使用box-sizing: border-box样式，还需要加上padding值），再配合top: 50%; left: 50%;样式就会使块元素居中。  给容器设置绝对定位（position:absolute）,并且定位高度（top:50%）和margin-top为高度的一半（margin-top:-height/2）  这就意味着使用这种方法来实现垂直居中的效果，那么元素必须要有一个固定的高度。这样一来，你给元素设置了固定高度，如果你又给他设置了”overflow:auto“，那么当元素内容超过容器后，这样元素的就会出现滚动，而不会自适应内容的高度。   1. 缺点：   由于固定死元素的高度，致使没有足哆的空间，当内容超过容器的大小时，要么会消失，要么会出现滚动条（如果元素在body内，当用户缩小浏览器窗口时，body的滚动条将不会出现）。  这种方法主要是针对多行元素来进行元素的垂直居中，而并非是此元素的内容垂直居中，这一点先要理解并分开清楚。另外此方法是通过绝对定位来实现的，那么用这种方法实现元素的垂直居中需要注意以下几点：  1、元素定位关系到一个相对定位参考，所以要保证元素是相对于哪个为参考坐标；  2、需要设置给元素明确定一个高度值并且给元素设置一个负值的margin-top，而且值为元素高度的一半。  这里我建议大家给元素定一个宽度值，因为元素进行绝对定位后脱离文档流，其宽度是根据元素内容所占宽度来计算的，为这能让其视觉效果更好，最好给元素定义一个宽度值。  这种方法可以实现元素的水平垂直居中，常用于页面在最中间的布局，使用这种方法一定要把握住：水平垂直居中的元素要有明确的大小（换句话说就是要有确切的宽和高度值）；给元素进行绝对定位，并设置left,top值为“50%”（此处最好使用 对定位，如果只是单单水平居中，此处可以换成相对定位）；最后设置margin-top和margin-left的负值，而且其值分别为元素高度和宽度的一半。  1.这是个非响应式的方法，不能使用百分比的大小，也不能设置min-/max-的最大值最小值。  2.内容可能会超出容器  3.需要为padding预留空间，或者需要使用box-sizing: border-box样式。  4.对此元素设置一个负的“margin-left”并且值等于宽度的一半，另外如果元素不是相对于浏览器的话，还需要在其父元素上设置一个相对定位“position: relative|absolute”。   1. 优点   这或许是最常用的方法。  需要注意的是，这是按照预想情况也能在工作在IE6-7下的唯一方法。  需要的编码量很少 | |
| **<!--外层无DIV则会居中于body-->**  **<!--外层无非 relative DIV则会居中于body-->** <**style**>  .inDiv{  top:50%;**/\*配合margin-top的负值实现垂直居中\*/**  width: 200px;**/\*元素的宽度\*/** height:200px;**/\*元素的高度\*/** position: absolute;**/\* 层漂浮，让top left起作用 \*/** left: 50%;**/\*配合margin-left的负值实现水平居中\*/** margin-top: -100px;**/\*值的大小等于元素高度的一半\*/** margin-left: -100px;**/\*值的大小等于元素宽度的一半\*/** } </**style**> <**div** class="inDiv">content</**div**> | |
| <**style**>  #outDiv {  position: relative;  height: 600px;  width: 500px;  }   #inDiv {  position: absolute;  width: 250px;  height: 50px;  top: 50%;  left: 50%;   **-webkit-transform: translate(-50%, -50%);  -ms-transform: translate(-50%, -50%);  transform: translate(-50%, -50%);**  } </**style**>      <**div** id="outDiv">  <**div** id="inDiv">  DIV居中演示  DIV居中演示  </**div**> </**div**>  好处：  内容高度可变、代码量小  注意：  不支持IE8  需要写厂商前缀  会和其他transform样式有冲突  某些情况下的边缘和字体渲染会有问题 | |

0000 居中

|  |  |
| --- | --- |
| 实现居中 | <**style**>  **/\*1:可以为relative-absolute-fixed；因为absolute元素的top-left作用于外层最近的非static元素\*/** .out {  position: relative;**/\*1\*/** height: 300px;  width: 200px;  }  .in {  position: absolute;**/\*必需\*/** width: 50%;  height: 50%;  **/\*overflow: auto;\*/** margin: auto;**/\*必需\*/** top: 0; left: 0; bottom: 0; right: 0;**/\*必需\*/** } </**style**> <**div** class="out">  <**div** class="in">asdf</**div**> </**div**> |
| 好处：  外层元素为body，则只要设置内容元素的高度即可  跨浏览器，兼容性好（无需hack，可兼顾IE8~IE10）；浏览器支持：Chrome、Firefox、Safari、Mobile Safari、IE8-10。 “完全居中”经测试可以完美地应用在最新版本的Chrome、Firefox、Safari、Mobile Safari中，甚至也可以运行在IE8~IE10上  无特殊标记，样式更精简  自适应布局，可以使用百分比和最大最小高宽等样式  居中时不考虑元素的padding值（也不需要使用box-sizing样式）  布局块可以自由调节大小  img的图像也可以使用  注意：  必须声明元素高度  推荐设置overflow:auto;样式避免元素溢出，显示不正常的问题  这种方法在Windows Phone上不起作用 |
|  |
|  |
| 可视居中 | <**style**>  .out {  position: relative;  height: 300px;  width: 200px;  }  .in {  width: 50%;**/\* closest(非static) \*//\*第一个非static父元素\*/** height: 50%;  overflow: auto;  margin: auto;  position: absolute;**/\*必需\*/** top: 0; left: 0; bottom: 0; right: 0;  }  **/\*在可视区域内居中\*/** .in.to-window {  position: fixed;  z-index: 999;  } </**style**> <**div** class="out">  <**div** class="in window ">asdf</**div**> </**div**> |
| 想要使内容区在可视区域内居中：设置position: fixed样式，并设置一个较高的z-index值 移动版Safari的说明：如果外面没有一层设置position: relative的容器，内容区会以整个文档的高度的中心点为基准居中，而不是以可视区域的高度中心点为基准居中。 |
| 偏移|靠边 | top: 0; left: auto; bottom: 0; right: 10px; |
| 如果需要添加固定的标题，或者其他带偏移样式的元素，可以直接把类似top: 70px; 的样式写进内容区域的样式中。一旦声明了margin: auto; 的样式，内容块的top left bottom right的属性值也会同时计算进去。  如果想让内容块在贴近侧边的过程中保持水平居中，可以使用right: 0; left: auto; 让内容贴在右侧，或者使用left: 0; right: auto; 使内容贴在左侧。 |
| 响应式 | .in.to-responsive {    width: 60%;    height: 60%;    min-width: 200px;    max-width: 400px;    padding: 40px;  } |
| 使用absolute的最大好处就是可以完美地使用带百分比的宽高样式！就算是min-width/max-width或者min-height/max-height也能够有如预期般的表现。  再进一步加上padding样式的话，absolute式的完全居中也丝毫不会破坏！ |
| 带溢出内容 | .in.to-verflow {    overflow: auto;  } |
| 内容区高度大于可视区域或者一个position: relative的容器，其内容可能会溢出容器，或被容器截断。只要内容区域没有超出容器（没有给内容容器预留padding的话，可以设置max-height: 100%;的样式），那么容器内就会产生滚动条。 |
| 大小可调整 | .in.to-resizable {    min-width: 20%;    max-width: 80%;    min-height: 20%;    max-height: 80%;    resize: both;    overflow: auto;  } |
| 使用其他样式，或者使用JavaScript调整内容区的大小，也是不用手动重新计算的！如果设置了resize的样式，甚至可以让用户自行调节内容区域的大小。 “完全居中”法，无论内容区怎么改变大小，都会保持居中。  设置了min-/max- 开头的属性可以限制区块的大小而不用担心撑开容器。  如果不设置resize: both的样式，可以设置transition样式平滑地在大小间切换。一定要记得设置overflow: auto样式，因为改变大小后的容器高宽很有可能会小于内容的高宽。 “完全居中”法是唯一一种能支持使用resize: both样式的方法。  使用注意：  1.需要设置max-width/max-height给内容区域留足够的空间，不然就有可能使容器溢出。  2.resize属性不支持移动版浏览器和IE8-10，如果用户体验很重要的话，请确保用户可以有其他替代方法来改变大小。  3.同时使用resize样式和transition会使用户在开始改变大小时产生等于transition效果时间等长的延时。 |
| 图像 | <img src="http://placekitten.com/g/500/200" alt="" />  .Absolute-Center.is-Image {    height: auto;  }  .Absolute-Center.is-Image img {    width: 100%;    height: auto;  } |
| 图像也同样有效！提供相应的class，并指定样式 height: auto; ，就得到了一张随着容器改变大小的响应式图片。  请注意，height: auto; 样式虽然对图片有效，如果没有用到了后面介绍的‘可变高技巧’，则会导致普通内容区域伸长以适应容器长度。  浏览器很有可能是根据渲染结果填充了图像高度值，所以在测试过的浏览器中，margin: auto; 样式就像是声明了固定的高度值一般正常工作。 |
| 可变高度 | /\* Modernizr Test for Variable Height Content \*/  Modernizr.testStyles('#modernizr { display: table; height: 50px; width: 50px; margin: auto; position: absolute; top: 0; left: 0; bottom: 0; right: 0; }', function(elem, rule) {  Modernizr.addTest('absolutecentercontent', Math.round(window.innerHeight / 2 - 25) === elem.offsetTop);  });  .absolutecentercontent .Absolute-Center.is-Variable {    display: table;    height: auto;  } |
| “完全居中”法的确需要声明容器高度，但是高度受max-height样式的影响，也可以是百分比。这非常适合响应式的方案，只需要设置好带溢出内容就行。  另一种替代方案是设置display: table样式居中，，不管内容的长度。这种方法会在一些浏览器中产生问题（主要是IE和Firefox）。我在ELL Creative的朋友Kalley写了一个基于Modernizr 的测试，可以用来检查浏览器是否支持这种居中方案。现在这种方法可以做到渐进增强。  注意要点： 这种方法会破坏浏览器兼容性，如果Modernizr测试不能满足你的需求，你可能需要考虑其他的实现方案。  1.与大小可调整技术是不兼容的  2.Firefox/IE8中使用display: table，内容区在垂直方向靠上，水平方向仍然居中。  3.IE9/10中使用display: table，内容区会跑到左上角。  4.移动版Safari中内容区是水平对齐的，但是如果使用了百分比的宽度，水平方向上会稍稍偏离中心。 |
| 其他居中方式 | |
| margin auto水平居中 | <**style**>  .horizontal {  width: 200px;  margin: 0 auto;  } </**style**> <**div** class="horizontal">content</**div**> |
| 这种方法主要使用margin: auto配合元素的width来实现水平居中的效果：  1.元素需要有一个固定宽度值；  2.元素的margin-left和margin-right都必须设置为auto  3.定位方式：[position: absolute|fixed;无效][position: static|relative;有效]  条件1、2少了任何一个都无法让元素达到水平居中的效果。 |
| 此方法使用水平居中，支持所有浏览器运行，因此他也常用来实现Web页面水平居中的布局效果，如果用在布局，需要注意IE下的效果。  换句话说，如果你的Web页面没有明确声明"!DOCTYPE"，那么在IE也会以怪异模式解析你的Web页面，此时上面这种办法将无法让你的页面实现水平居中，但你可以使用下面这种方法解决这个bug：  <**style**>  **body** {  text-align: center;**/\*实现IE在怪异模式下页面水平居中\*/** }  .container {  text-align: left;**/\*让文本恢复到左对齐\*/** width: 960px;**/\*设置页面总宽度\*/** margin: 0 auto;**/\*使页面水平居中对齐\*/** } </**style**> <**div** class="container">页面内容。... </**div**> |
| 盒布局：box | .out{  display:--box;  --box-align:center;  --box-pack:center; } |
| 视口绝对居中vh|vw | <style>  .blackSquare {  background: black;  position: fixed;  height: 50vh;  width: 50vw;  left: 25vw;  top: 25vh;  } </style> <div class="blackSquare"></div> |
|  |
| 50%高、float元素辅助 | <**style**>  **html**,**body** {height: 100%;}  #floater{  float:left;  height:50%;**/\*相对于父元素高度的50%\*/** margin-bottom: -120px;**/\*值大小为居中元素高度的一半（240px/2）\*/** }  #content {  clear:both;**/\*清除浮动\*/** height: 240px;  position: relative;  } </**style**> <**body**>  <**div** id="floater"></**div**>  <**div** id="content">Content section</**div**> </**body**> |
| #parent {height: 250px;}  #floater {  float: left;  height: 50%;  width: 100%;  margin-bottom: -50px;  }  #child {  clear: both;  height: 100px;  }  <div id="parent">  <div id="floater"></div>  <div id="child">Content here</div>  </div> |
|  |
| 这种方法有点新意，用这种方法你需要在居中元素前面放一个空的<div>（块元素就可以），然后设置这个<div>的高度为50%，margin-bottom为元素高度的一半，而且居中元素需要清除浮动。需要注意的是，使用这种方法，如果你的居中元素是放在body中的话，你需要给html,body设置一个“height:100%”的属性。  优点：  这种方法能兼容所有浏览器，在没有足哆空间下，内容不会被切掉  缺点：  元素高度被固定死，无法达到内容自适应高度，如果居中元素加上overflow属性，要么元素出现滚动条，要么元素被切掉；另外就是一个不算缺点的缺点，那就是加了一个空标签。 |
| JS + CSS控制:不漂浮（适用于做登陆页面） | <**script** type="text/javascript">    **function** cen(){  //居中高度等于页面内容高度减去DIV的高度 除以2  **document**.getElementById(**"divNme"**).**style**.**marginTop** = (**document**.**body**.**clientHeight** - **document**.**all**(**"testdiv"**).**clientHeight**)/2; } //页面大小发生变化时触发     **window**.onresize = cen;  </**script**> <**body** style="height:100%; width:100%; text-align:center;"  onload=cen()> <**div** id = "testdiv" name="testdiv"   style="margin:0 auto; border:1px solid red; height:400px; width:400px;">  DIV居中演示 </**div**> </**body**> |
| JQ实现 | 正如前面所述，如果元素固定高度后要实现垂直居中，用CSS实现很方便，但对于自适应高度的就比较棘手了。那么第七种方法，要给大家介绍的是使用jQuery的方法  Html Markup  <div class="container">  <p>Centered In The Middle Of The Page With jQuery</p>  </div>    CSS Code  .container{  background-color:#338BC7;  width:270px;  height:150px;  }    jQuery Code  $(document).ready(function(){  $(window).resize(function(){  $('.container').css({  position:'absolute',  left: ($(window).width() - $('.container').outerWidth())/2,  top: ($(window).height() - $('.container').outerHeight())/2  });  });  // 最初运行函数  $(window).resize();  });    优点：  这种方法，结构简单，易懂，不需要固定元素的大小。兼容各浏览器。  缺点：  需要调用jQuery，如果不支持js或者用户禁掉了js，那么将无法正常运行，那就是杯具了。  上面这种方法对实现页面居中布局很方便，下面我根据上面的思路和方法二写了一个实现元素水平垂直居中的小插件  jQuery Plugin  (function($){  $.fn.vhAlign = function(){  return this.each(function(i){  //获取元素的内容高度  var h = Math.ceil($(this).height());  //outerHeight=padding+border+height  var oh = Math.ceil($(this).outerHeight());  //取得margin-top值  var mt = Math.ceil(oh / 2);  //取得元素宽度  var w = Math.ceil($(this).width());  //outerWidth=padding+border+width  var ow = Math.ceil($(this).outerWidth());  //取得margin-left  var ml = Math.ceil(ow / 2);  //实现元素居中效果  $(this).css({  "margin-top": "-" + mt + "px",  "top": "50%",  "margin-left": "-" + ml + "px",  "left": "50%",  "width":w,  "height":h,  "position": "absolute"  });  });  };  })(jQuery);    Html Markup  <div class="wrap">  <p>test jquery</p>  </div>    接下来需要调用刚才写好的jQuery插件  <script type="text/javascript" src="vhAlign.js"></script>    最后需要在div.wrap调用这个function  <script type="text/javascript">  $(document).ready(function(){  $(".wrap").vhAlign();  });  </script>    这里有一点需要特别注意，如果元素不是相对于body居中，那么需要在其父元素中进行相对定位。  上面我们一起见证了八种不同方法实现元素的垂直居中效果，下面我们在简单的看一下如何实现元素的水平居中（其实上面有一些效果也实现了水平居中，你可以慢慢回想一下）。 |
|  |  |
| cacl | <**style**>  .center {  background: hsl(300, 100%, 97%);  min-height: 600px;  position: relative;  }   .center **img** {  width: 40%;  height: auto;  position: absolute;  top: calc(50% - 20%);  left: calc(50% - 20%);  } </**style**>  Very simply, calc allows you to do calculations based on the current layout of the page. In the simple calculations above, 50% is the halfway point in the container element, but using that alone would set the top left corner of the image to the center of the <div>. We need to bring that position back by half the width and half the height of the image. Expressing that in a longer form:  top: calc(50% - (40% / 2));  left: calc(50% - (40% / 2));  In current browsers you’ll find that this technique tends to work best on content with fixed, known dimensions:  <**style**>  .center **img** {  width: 500px;  height: 500px;  position: absolute;  top: calc(50% - (300px / 2));  left: calc(50% - (300px–2));  } </**style**>  I [discuss calc in depth in another article](http://thenewcode.com/953/Layout-Math-with-CSS-Understanding-calc).  The technique has many of the same potential drawbacks as flexbox: while it has good support in modern browsers, it will require vendor prefixes for earlier versions, and has no support in IE 8. <**style**>  .center **img** {  width: 40%;  height: auto;  position: absolute;  top: calc(50% - 20%);  left: calc(50% - 20%);  } </**style**>  While there are still more options such as [using pseudo-elements for vertical alignment](http://jsfiddle.net/g8mnf), understanding these six techniques will give any web developer a solid repertoire to draw on when centering elements. |
|  | about calc():  <http://www.w3cplus.com/css3/how-to-use-css3-calc-function.html> |
| flex居中 | <**style**>  .center {  background: hsl(240, 100%, 97%);  display: flex;  justify-content: center;  align-items: center;  }   .center **img** {  width: 30%;  height: auto;  } </**style**>  Will probably become the centering solution of choice once the property variants and vendor prefixes have faded away.  In many respects [flexbox](http://thenewcode.com/css/flexbox) is the simplest solution, but one that is held back by the variety of older syntaxes and lack of support in earlier versions of IE (although display: table-cellmakes for an acceptable fallback).  Now that the spec has been finalized and browser support is settling down, I have written extensively on [flexbox layout and its uses.](http://thenewcode.com/780/A-Designers-Guide-To-Flexbox) |
|  | flex：  对父级元素写css:  单行居中：  display : flex;  justify-content : center;  align-items : center;  多行垂直居中：  display : flex;  justify-content : center;  flex-wrap : wrap/wrap-reverse;  align-items : center; |
| 未知元素居中1:table-cell布局 | <https://css-tricks.com/centering-in-the-unknown/>  在表格中这种效果是显而易见的： <**table** style="width: 100%;">  <**tr**>  <**td** style="text-align: center; vertical-align: middle;"> Unknown stuff to be centered.</**td**>  </**tr**> </**table**> 借助表格的具体实现： <**style**>  .something-semantic {  display: table;  width: 100%;  }  .something-else-semantic {  display: table-cell;  text-align: center;  vertical-align: middle;  } </**style**> <**div** class="something-semantic">  <**div** class="something-else-semantic"> Unknown stuff to be centered.</**div**> </**div**> |
| 未知元素居中2:inline-block布局 | <**style**>  **/\* This parent can be any width and height \*/** .block {  text-align: center; **/\* May want to do this if there is risk the container may be narrower than the element inside \*/  /\*white-space: nowrap;\*/** }   **/\* The ghost, nudged to maintain perfect centering \*/** .block:before {  content: **'\200B'**;  **/\* content: '';  margin-left: -0.25em; \*//\* Adjusts for spacing \*/** display: inline-block;  height: 100%;  vertical-align: middle;  }   **/\* The element to be centered, can also be of any width and height \*/** .centered {  display: inline-block;  vertical-align: middle;  width: 300px;  } </**style**>  <**div** class="block" style="height: 300px;">   <**div** class="centered">  <**h1**>Some text</**h1**>  <**p**>But he stole up to us again, and suddenly clapping his hand on my shoulder, said&mdash;"Did ye see anything looking like men going towards that ship a while ago?"</**p**>  </**div**>  </**div**>  <**div** class="block" style="height: 200px;">   <**div** class="centered">  <**h1**>Some text</**h1**>  <**p**>But he stole up to us again, and suddenly clapping his hand on my shoulder, said&mdash;"Did ye see anything looking like men going towards that ship a while ago?"</**p**>  </**div**>  </**div**>  <**div** class="block" style="height: 600px;">   <**div** class="centered">  <**h1**>Some text</**h1**>  <**p**>But he stole up to us again, and suddenly clapping his hand on my shoulder, said&mdash;"Did ye see anything looking like men going towards that ship a while ago?"</**p**>  </**div**>  </**div**> |
| I'd like to tell you the ghost element technique is way better and should be the go-to centering technique for the ages. But in reality, it's just about the same as the table trick. The browser support for this is essentially everything and IE 8+. IE 7 doesn't support pseudo elements. But it doesn't support CSS tables either, so it's a horse apiece. If IE <= 7 support is needed, it's <table> time (or use an equally un-semantic<span> or something for the ghost element).  This stuff isn't brand new territory. Gary Turner [wrote about it](http://gtwebdev.com/workshop/vcenter/vcenter-inline-css.php) like 5 years ago. But I credit Michał for doing it with a pseudo element and making it the most semantic approach yet.  Note: The 0.25em nudge-back is a little janky. To do it perfectly, you could set font-size: 0; on the parent and then notch the font size back up inside the content container. |
|  |  |

图片列水平垂直居中控制

|  |  |  |
| --- | --- | --- |
| www | 大小不固定的图片和多行文字的垂直水平居中：  <http://www.zhangxinxu.com/study/200908/img-text-vertical-align.html#zhangxinxu> | |
| inline-block居中 | <**style**>  .image\_ul {  overflow: hidden;  zoom: 1;  }  .image\_ul **li** {  float: left;  width: 150px;  height: 150px;  text-align: center;  line-height: 150px;  \*font-size: 125px;  list-style: none;  }  .image\_ul **li**:after {  content: **' '**;  vertical-align: middle;  }  .image\_ul **li img** {  vertical-align: middle;  } </**style**>  <**ul** class="image\_ul">  <**li**><**img** src="http://image.zhangxinxu.com/image/study/s/s128/mm1.jpg" /></**li**>  <**li**><**img** src="http://image.zhangxinxu.com/image/study/s/s128/mm2.jpg" /></**li**>  <**li**><**img** src="http://image.zhangxinxu.com/image/study/s/s128/mm3.jpg" /></**li**>  <**li**><**img** src="http://image.zhangxinxu.com/image/study/s/s128/mm4.jpg" /></**li**>  <**li**><**img** src="http://image.zhangxinxu.com/image/study/s/s128/mm5.jpg" /></**li**>  <**li**><**img** src="http://image.zhangxinxu.com/image/study/s/s128/mm6.jpg" /></**li**> </**ul**> | |
| background center居中 | 使用一个透明图片做覆盖层，高宽拉伸至所需要的大小，然后给这个透明图片一个background-position:center; 的属性。  而background-image建议写在页面上，以适应动态的图片url地址。  <**style**>  **ul**.box {  list-style: none;  }  **ul**.box **li** {  width: 1em;  height: 1em;  padding: 0.1em;  margin: 0 0.1em 0 0;  font-size: 128px;  float: left;  border: 1px solid #beceeb;  }   **ul**.box **li img** {  display: block;  width: 100%;  height: 100%;  background-repeat: no-repeat;  background-position: center;  } </**style**> <**ul** class="box">  <**li**>  <**img** src="images/000.png" style="background-image:url(images/1.jpg);" />  </**li**>  <**li**>  <**img** src="images/000.png" style="background-image:url(images/ttt.jpg);" />  </**li**>  <**li**>  <**img** src="images/000.png" style="background-image:url(images/2.jpg);" />  </**li**> </**ul**> | |
| table-cell居中 | 1. display:table-cell和文字大小控制居中   这个方法可以说就是本文综述部分提到的淘宝团队想出的图片垂直居中的方法。  但是本文展示的这个方法的css代码又不是淘宝工程师们的原版代码，我对其做了一定的修改，把里面没有用可以剔除的hack都去掉了。  说明：   1. 原版的代码中有两个hack，一个是针对文字大小的，另外一个是针对block属性的；后经过我测试推敲后证实这两个hack都是多余的，而\*display:block更是多余，因为IE6,IE7根本就不认识display:table-cell是谁！ 2. 这个通过文字大小控制IE下图片垂直居中是个很不错的方法，要比使用position:relative这类高消耗的css方法要好多了。但是这个方法不足之处在于：不支持img外标签的浮动，所以当多图片显示时需要再在外面嵌套一层标签——资源消耗多了！ 3. 高度：文字大小=1.14，这个比例我一直记着，但是后来我将这个比例的概念淡化了，原因在于多次遇到其他比例实现效果的情况。例如上面，就是1:1实现的。 | |
| <**style**>  .ul\_box{  list-style: none;  }  .ul\_box **li** {  float: left;  margin-right: 10px;  }  .ul\_box **li div** {  display: table-cell;  width: 300px;  height: 300px;  border: 1px solid red;  text-align: center;  vertical-align: middle;  }  .ul\_box **li div img**{  vertical-align: middle; **/\*???是否为必须\*/** } </**style**>  <**ul** class="ul\_box fix">  <**li**>  <**div**><**img** src="images/1.jpg" /></**div**>  </**li**>  <**li**>  <**div**><**img** src="images/2.jpg" /></**div**>  </**li**>  <**li**>  <**div**><**img** src="images/3.jpg" /></**div**>  </**li**>  <**li**>  <**div**><**img** src="images/4.jpg" /></**div**>  </**li**> </**ul**> | |
| 图片水平垂直居中4-[最精简的实现图片水平垂直居中显示的方法] | 1. display:inline-block和文字大小控制居中   此方法的灵感来自于Google picasa相册的图片显示，代码相当简洁，是个成本很低，效果惊人的方法。适用于多图显示的情况。基本上用裸标签就实现了想要达到的效果。一般而言，图片阵列排列显示的时候，外面都有一个a标签的，起到链接的作用。而本处的方法就只要这一个a标签就足以实现图片垂直且居中的显示效果。其关键是将a标签默认的inline属性设置为inline-block属性，这样a标签既支持宽度，又支持vertical-align:middle，配合img的vertical-align:middle就可以实现图片的水平垂直居中显示了。 img外的标签需可以使a标签或span这类行标签；  也可以是div、li这样的块标签，但是css代码多些: display:inline-block;\*display:inline; zoom:1; IE下要先变成inline属性，再转变成类似于inline-block的属性，原因见上 | |
| <**style**>  .out\_div{  border:1px solid red;  font-size: 0px; **/\*消除空隙\*/** }  .out\_div **a** {  display: inline-block;  width: 300px;  height: auto; **/\*不能具体定义height，影响内部图片的垂直居中。\*/** font-size: 128px; **/\*???意义何在\*/** padding: 6px;  border:1px solid red;  text-align: center;  vertical-align: middle;  }   .out\_div **a img** {  **/\*在此处对图片的边框作处理效果更佳\*/** vertical-align: middle;  **/\*此处middle无效果，实现垂直居中的是：具备middle属性的inline-block元素在父元素中垂直居中。\*/** } </**style**>  <**div** class="out\_div">  <**a** href=""><**img** src="images/1.jpg" /></**a**>  <**a** href=""><**img** src="images/2.jpg" /></**a**>  <**a** href=""><**img** src="images/3.jpg" /></**a**>  <**a** href=""><**img** src="images/4.jpg" /></**a**>  <**a** href=""><**img** src="images/5.jpg" /></**a**>  <**a** href=""><**img** src="images/6.jpg" /></**a**> </**div**>    ？？？inline-block元素中inline-block元素（img）的垂直居中问题 | |
| inline block图片辅助居中 | 透明图片拉伸对齐实现垂直居中显示  这个方法是[我](http://www.zhangxinxu.com/wordpress/2009/08/%e5%a4%a7%e5%b0%8f%e4%b8%8d%e5%9b%ba%e5%ae%9a%e7%9a%84%e5%9b%be%e7%89%87%e3%80%81%e5%a4%9a%e8%a1%8c%e6%96%87%e5%ad%97%e7%9a%84%e6%b0%b4%e5%b9%b3%e5%9e%82%e7%9b%b4%e5%b1%85%e4%b8%ad/" \l "zhangxinxu" \o "张鑫旭)自认为相当不错的一个方法，说实话，想到这个方法是自然而然的，经过试验发现真的很管用。对比淘宝团队的方法，发现优势很多，基本上淘宝的table-cell+font-size方法会带来的些问题都避免了。  原理简述：  一句话，将要显示的图片与一张透明的高度100%，宽度0像素的透明图片vertical-align:middle对齐。  其核心原理其实与利用font-size大小实现IE下图片垂直居中是一致的。将font-size设置得很大，目的是撑开IE下默认文字空间的高度，其性质类似于空格，然后通过vertical-align:middle属性让图片与这个高高的空白空格空间垂直居中对齐；  而这里将这个看不见的文字空间实例成一张透明的gif图片，高度可以轻松设置为外部标签的高度，宽度压缩至最小，然后通过vertical-align:middle对齐，就实现效果了.  由于影响布局的差不多就vertical-align:middle这一个属性，所以出现兼容性问题的可能性比table-cell的方法要低得多。  需要注意的：   1. img外容器宽度要大于要显示的图片的最大宽度+1像素； 2. img外容器的字体大小设为0px，也可以不设，但是两个img标签要连着写，避免空格；   要显示的图片后面跟了一个高度撑满容器，宽度为0即可，透明的图片。这两个图片分别vertical-align:middle，就实现了要显示的图片与这个拉伸的透明图片的居中对齐了，由于透明图片是透明的，不可见的，宽度也为0，所以看上去就是要显示的图片相对于容器垂直居中了。  <**style**>  .ul\_box {  list-style: none;  }   .ul\_box {  overflow: hidden;  zoom: 1;  }   **li** {  height: 300px;  width: 300px;  padding: 13px 0;  float: left;  margin-right: 10px;  border: 1px solid #beceeb;  text-align: center;  font-size: 0; **/\*防止两个inline-block元素中间存在空格\*/**  }   **li** .assist\_img {  height: 100%;  width: 0px; **/\*1px\*/** vertical-align: middle; **/\*必须\*/** }   **li** .show\_img {  vertical-align: middle; **/\*必须\*/** } </**style**> <**ul** class="ul\_box">  <**li**><**img** src="images/1.jpg" class="show\_img "/><**img** src="images/000.png" alt="" class="assist\_img"/></**li**>  <**li**><**img** src="images/2.jpg" class="show\_img "/><**img** src="images/000.png" alt="" class="assist\_img"/></**li**>  <**li**><**img** src="images/3.jpg" class="show\_img "/><**img** src="images/000.png" alt="" class="assist\_img"/></**li**>  <**li**><**img** src="images/4.jpg" class="show\_img "/><**img** src="images/000.png" alt="" class="assist\_img"/></**li**>  <**li**><**img** src="images/5.jpg" class="show\_img "/><**img** src="images/000.png" alt="" class="assist\_img"/></**li**>  <**li**><**img** src="images/6.jpg" class="show\_img "/><**img** src="images/000.png" alt="" class="assist\_img"/></**li**> </**ul**>    另外，其实这里不一定是透明的图片，使用任意的图片都可以，所以您其实说不定可以减少一次http请求，使用任意的图片将其拉伸至高度为容器高，宽度0就可以了。  甚至，使用span标签，div标签也可以实现同样的效果。只要将span标签或div标签转换成inline-block属性或类似于inline-block属性就可以了。 推荐使用行内元素，例如span，i标签等，只要应用类似下面的CSS：  span {  display: inline-block;  height: 100%;  width: 0;  vertical-align: middle; }  相比图片而言，多了个display:inline-block; 但是会少一次链接请求。效果图和上面的一样的，完美的水平垂直居中。原理也与图片一样，相信很容易理解的。 | |
| inline block空白辅助居中 | 没有hack，兼容性上佳（支持IE6、IE7、Firefox、chrome、Safari浏览器）（在IE8浏览器和Opera浏览器下是不垂直居中的，此方法还是有待商榷的）。 此方法原理是:  1、在IE下使用font-size使图片垂直居中显示，  2、Firefox，chrome等现代浏览器使用line-height配合img本身的vertical-align属性使垂直居中显示，由于两者不冲突，所以没有hack就实现的效果。  css代码简洁明了，关键是HTML代码非常清晰，一层外标签，里面就是img标签，我想很难再找出比这个方法更出色的让图片水平垂直居中的方法了。好吧，其实这个方法是有一点不完美的，就是opera浏览器下图片无法垂直居中显示，幸好国内而言，使用opera浏览器的比例很低，所以综合来讲这个方法是个非常优秀的图片水平垂直居中的方法。 | |
| <**style**>  .ul\_box {  list-style: none;  }   .ul\_box {  overflow: hidden;  zoom: 1;  }   .ul\_box **li** {  float: left;  width: 300px;  height: 300px;  text-align: center;  line-height: 300px;  \*font-size: 200px;  **/\*???IE使用font-size使图片居中 height:150px; -> \*font-size:125px;\*/** }   .ul\_box **li**:after {  content: **' '**;  vertical-align: middle;  }   .ul\_box **li img** {  vertical-align: middle;  } </**style**> <**ul** class="ul\_box">  <**li**><**img** src="images/1.jpg" /></**li**>  <**li**><**img** src="images/2.jpg" /></**li**>  <**li**><**img** src="images/3.jpg" /></**li**>  <**li**><**img** src="images/4.jpg" /></**li**>  <**li**><**img** src="images/5.jpg" /></**li**>  <**li**><**img** src="images/6.jpg" /></**li**> </**ul**> | |
|  |  | |
| 如何选择居中方式 | | |
| “完全居中”并不是本篇文章中唯一的选项。要做到垂直居中，还存在着其他方法，各有各的长处。采取什么样的方法，取决于你所支持的浏览器，以及现有标签的结构。下面这张对照表能够帮你选出最符合你需要的方法。   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **所用样式** | **支持的浏览器** | **是否 响应式** | **内容溢出后的样式** | **resize:both** | **高度可变** | **主要缺陷** | | Absolute | 现代浏览器&IE8+ | 是 | 会导致容器溢出 | 是 | 是\* | ‘可变高度’的特性不能跨浏览器 | | 负margin值 | 所有 | 否 | 带滚动条 | 大小改变后不再居中 | 否 | 不具有响应式特性，margin值必须经过手工计算 | | Transform | 现代浏览器&IE9+ | 是 | 会导致容器溢出 | 是 | 是 | 妨碍渲染 | | Table-Cell | 现代浏览器&IE8+ | 是 | 撑开容器 | 否 | 是 | 会加上多余的标记 | | Inline-Block | 现代浏览器&IE8+&IE7\* | 是 | 撑开容器 | 否 | 是 | 需要使用容器包裹和hack式的样式 | | Flexbox | 现代浏览器&IE10+ | 是 | 会导致容器溢出 | 是 | 是 | 需要使用容器包裹和厂商前缀（vendor prefix） | | |
| 在研究了规范和文档后，我总结出了“完全居中”的工作原理：  1. 在普通文档流里，margin: auto; 的意思是设置元素的margin-top和margin-bottom为0。  [W3.org](http://www.w3.org/TR/CSS2/visudet.html" \l "normal-block):?If ‘margin-top’, or ‘margin-bottom’ are ‘auto’, their used value is 0.  2. 设置了position: absolute; 的元素会变成块元素，并脱离普通文档流。而文档的其余部分照常渲染，元素像是不在原来的位置一样。 [Developer.mozilla.org](https://developer.mozilla.org/en-US/docs/Web/CSS/position" \l "Absolute_positioning):?…an element that is positioned absolutely is taken out of the flow and thus takes up no space  3. 设置了top: 0; left: 0; bottom: 0; right: 0; 样式的块元素会让浏览器为它包裹一层新的盒子，因此这个元素会填满它相对父元素的内部空间，这个相对父元素可以是是body标签，或者是一个设置了position: relative; 样式的容器。 [Developer.mozilla.org](https://developer.mozilla.org/en-US/docs/Web/CSS/position" \l "Notes):?For absolutely positioned elements, the top, right, bottom, and left properties specify offsets from the edge of the element’s containing block (what the element is positioned relative to).  4. 给元素设置了宽高以后，浏览器会阻止元素填满所有的空间，根据margin: auto; 的要求，重新计算，并包裹一层新的盒子。 [Developer.mozilla.org](https://developer.mozilla.org/en-US/docs/Web/CSS/position" \l "Notes):?The margin of the [absolutely positioned] element is then positioned inside these offsets.  5. 既然块元素是绝对定位的，又脱离了普通文档流，因此浏览器在包裹盒子之前会给margin-top和margin-bottom设置一个相等的值。 [W3.org](http://www.w3.org/TR/CSS2/visudet.html" \l "abs-non-replaced-height):?If none of the three [top, bottom, height] are ‘auto’: If both ‘margin-top’ and ‘margin-bottom’ are ‘auto’, solve the equation under the extra constraint that the two margins get equal values.?AKA: center the block vertically  使用“完全居中”，有意遵照了标准margin: auto; 样式渲染的规定，所以应当在与标准兼容的各种浏览器中起作用。 | |
| Additional Resources  In the sections above I linked to articles specific to each method. Below are some resources that cover more than a single method for centering.  *[Vertical Centering With CSS](http://blog.themeforest.net/tutorials/vertical-centering-with-css/)* — covers all the methods above except for the padding method  *[Two Simple Ways to Vertically Align with CSS](http://www.sohtanaka.com/web-design/vertical-alignment-css/)* — Covers absolute positioning and negative margins and the line height methods  *[Vertical centering using CSS](http://www.student.oulu.fi/~laurirai/www/css/middle/)* — Covers all methods except the floater div method and adds a few more.  *[CSS tests and experiments](http://www.brunildo.org/test/)* — Contains a variety of css experiments. Search the page for the word vertical and you’ll find several vertical centering experiments, including the 2 links below.  *[Centering (horizontally and vertically) an image in a box](http://www.brunildo.org/test/img_center.html)*  *[Shrink-wrap and Center](http://www.brunildo.org/test/shrink_center_5.html)* | |