[基本属性学习 2](#_Toc393973590)

[**一、淡淡的开头语** 2](#_Toc393973591)

[box-flex 2](#_Toc393973592)

[**二、box-flex属性（和谐版）** 2](#_Toc393973593)

[**三、CSS box-flex属性（不和谐版）** 4](#_Toc393973594)

[**四、爸爸其实很厉害，的说~** 6](#_Toc393973595)

[box-orient 6](#_Toc393973596)

[box-direction 6](#_Toc393973597)

[box-align 7](#_Toc393973598)

[box-pack 8](#_Toc393973599)

[box-lines 10](#_Toc393973600)

[box-flex-group和box-ordinal-group 10](#_Toc393973601)

[**五、两个遗漏的属性** 10](#_Toc393973602)

[宽度计算 11](#_Toc393973603)

[垂直居中、水平居中 14](#_Toc393973604)

[已解决的问题 16](#_Toc393973605)

[1. 指定了box-flex分数的盒子，如果其子元素具有width，则该盒子会被撑大，而不再保持分数比例！ 16](#_Toc393973606)

[问题描述 16](#_Toc393973607)

[原因分析 17](#_Toc393973608)

[解决方案 17](#_Toc393973609)

# 基本属性学习

参考：

http://www.zhangxinxu.com/wordpress/2010/12/css-box-flex%E5%B1%9E%E6%80%A7%EF%BC%8C%E7%84%B6%E5%90%8E%E5%BC%B9%E6%80%A7%E7%9B%92%E5%AD%90%E6%A8%A1%E5%9E%8B%E7%AE%80%E4%BB%8B/

**一、淡淡的开头语**

昨天趁着不想工作的时间间隙闲逛24ways，在[My CSS Wish List](http://24ways.org/2010/my-css-wish-list)一文中，见到了个新鲜的CSS属性，就是题目中的box-flex，以前没有见过，顿生疑惑，不知是骡子还是马，于是习惯性谷歌之，真是不谷不知道，一谷吓一跳。倒不是该属性本身，而是此属性作为导火索，让我了解了下CSS3中新的盒子模型——弹性盒子模型(Flexible Box Model)。对于我这样的流体布局控而言，这种盒子模型的出现就好比打麻将杠上开花杠到绝张边七条，让人兴奋不已。在国外，弹性盒子模型早在去年就开始被提及，研究，与应用。然而，自己现在才第一次听到此概念，真是一不留神就out了，学习这东西，果然松懈不得。

本文内容叙述撇开以往顺流而下的方式，直接以box-flex属性为切入口，直入大本营，再铺开叙述。

## box-flex

**二、box-flex属性（和谐版）**

有道桌面词典显示，”flex”一词中文有“收缩”之意。不过，从此属性实际上产生的效果来看，无论怎样用“收缩”一词解释都显得很牵强。所以，这里，直接抛开字面意思，我们可以将”box-flex”理解为”房子-分配”。box为“盒子”的意思，我们可以理解为当下价格巨高的“房子”，”flex”指兄弟几个“分配房子”。

举个更实际点的例子：马林大叔省吃俭用一辈子，终于在上海郊外买了间150平米的商品房。后来，马林大叔想回老家养老，决定把房子分配给他的三个儿子。ok，先暂停下，这里提到的“房子”就是”box-flex”中的”box”，“分配”就是”box-flex”中的”flex”，于是，这个“分配房子”的举动就称为”box-flex”，而box-flex属性的值就是说的如何分配，分配比例是什么。oK，继续我们的例子，马林大叔的三个儿子分别叫做大马，中马和小马，其中大马已经结婚多年，有一堆双胞胎女儿，拖家带口的人多；而中马和小马是优秀的光棍人士。所以，大马要求分配更多的房子，最终，家人一番协商有了下面的分配结果，就是：

#大马 { 房子-分配: 2; }

#中马 { 房子-分配: 1; }

#小马 { 房子-分配: 1; }

我想，上面的分配应该很容易看懂的。房子分成了总共4份，其中有家室的大马分得其中的两份，而为国家省橡胶的中马和小马每人分得其中一份，于是用数值换算就是：  
大马 = 150 \* (2 / (2+1+1)) = 75（平米）;  
中马 = 150 \* (1 / (2+1+1)) = 37.5（平米）;  
小马 = 150 \* (1 / (2+1+1)) = 37.5（平米）;

如果装换成CSS表示就是：

#first\_boy { box-flex: 2; }

#second\_boy { box-flex: 1; }

#three\_boy { box-flex: 1; }

哇咔咔，box-flex的含义与作用理解瞬间柳暗花明：用来按比例分配父标签的宽度(或高度)空间。

box-flex的值为至少为1的整数时起作用。但是，仅仅一个box-flex属性是不足以实现子元素间的空间分配，因为还要看其老爸的意思。所谓，我爸是李刚，撞人很嚣张；恨爸不是刚，撞人心慌慌。只有老爸开口说：“这个房子现在你们随意分配。”其子女才能分配。

所以，父元素也是需要添加必要的声明的。此声明就是：

#father { display: box; }

似乎也可以是：

#father { display: inline-box; }

此声明好像是在说：孩子们，现在我把这个房子变成了可随意分配状态，非固定财产，你们可以自己协商分配了。  
display: box;的声明其实就是弹性盒子模型的声明，此声明下的子元素的行为与表现与CSS2中的传统盒子模型的表现是有显著的差异的。

毕竟属于CSS3的东西，目前而言，仅Firefox/Chrome/Safari浏览器支持弹性盒子模型（IE9不详，Opera尚未），且使用的时候，需要附带私有前缀。就是诸如-moz-, -webkit-之类。

**CSS实例**  
现在把上面的马林分房子的例子CSS实例化，看看在web页面上是个如何的表现：//zxx:有把小说拍成电影的感觉，抠鼻

主要CSS代码如下：

.test\_box {

display: -moz-box;

display: -webkit-box;

display: box;

...

}

.list {

...

}

.list\_one {

-moz-box-flex: 1;

-webkit-box-flex: 1;

box-flex: 1;

}

.list\_two{

-moz-box-flex: 2;

-webkit-box-flex: 2;

box-flex: 2;

}

HTML代码如下：

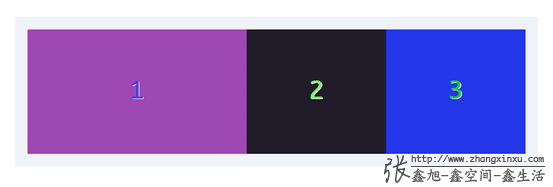
<div class="test\_box">

<div class="list list\_two">1</div>

<div class="list list\_one">2</div>

<div class="list list\_one">3</div>

</div>

结果如下缩略图：  


从上图可以看去，老大大马确实分配到了2份的房子空间，而中马和小马均分到了一份房子空间。

您可以狠狠地点击这里：[box-flex弹性布局测试demo](http://www.zhangxinxu.com/study/201012/css-box-flex-demo-1.html)

**三、CSS box-flex属性（不和谐版）**

继续上面马林大叔分房的例子。原本兄弟三人和和睦睦是一点问题都没有的，房子怎么分也基本都定下来了。然而，突然，事情起了波澜。老三小马突然有了个彪悍的女朋友，叫阿凤。小马本人对分配房子的大小是觉得无所谓的，即使两个人住，近40平米的屋子也足够了，何必为了这点事情伤了兄弟们间的和气。然而，小马的女友阿凤却是个吃不了亏的人，说什么也要争口气，于是，找来大马中马，强烈要求要加大他们房子的分配面积。

在大马，中马看来，阿凤还属于外来人，凭什么对他们兄弟的房子指手划脚，于是，没得妥协，于是，争执不断，于是，愈演愈烈，于是，不可开交。于是，有天，阿凤实在憋不住了，在厨房做菜的时候突然拿着菜刀跑出来，大声咆哮：“不管怎样，反正我家小马至少要50平米的房子，其余的怎么分是你们的事情，我不管，这是我的底线了，再低就没得商量！！”大马等被这架势吓住了，最终还是妥协了：小马就50平米（即使以后房子扩建还是50平米），剩下的面积大马，中马2:1比例再分配。于是就有：

#大马 { 房子-分配: 2; }

#中马 { 房子-分配: 1; }

#小马 { 房子-分配: 50m2; }

改编成CSS剧本就是：

#first\_boy { box-flex: 2; }

#second\_boy { box-flex: 1; }

#three\_boy { width: 50px; }

还是不难理解，当子元素中有宽度值的时候，此元素就定宽处理，剩下的空间再按比例分配。  
于是，此时，大马的房子大小是：(150 – 50) \* (2 / (1 + 2)) = 66.7平米，中马分配房子大小是：(150 – 50) \* (1 / (1 + 2)) = 33.3平米。

还是类似上面的demo，看看含有定宽元素的子元素是如何表现的。

新增CSS样式如下：

.list\_w300 { width: 300px; }

HTML代码如下：

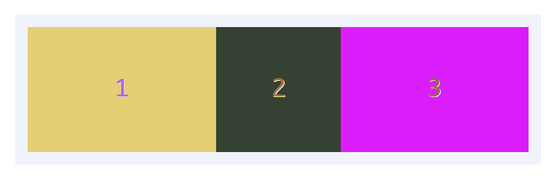
<div class="test\_box">

<div class="list list\_two">1</div>

<div class="list list\_one">2</div>

<div class="list list\_w300">3</div>

</div>

结果如下缩略图：  


老三分得300像素的宽度，剩下的500像素宽度老大和老二2:1比例分配。

您可以狠狠地点击这里：[含定宽元素弹性布局demo](http://www.zhangxinxu.com/study/201012/css-box-flex-demo-2.html)

然而，事情还没有结束。兄弟几个相处了一段时间后发现，偌大的屋子如果全部都是私有的话，会有诸多生活上的不便。所以，需要腾出些公共空间，给屋子透个气。咋办呢，老三小马的女友阿凤死活不妥协，没有办法，老马和中马只能牺牲自己的住所面积作为公共空间了。

反应到CSS上，大致就是增加了margin间距，如下HTML：

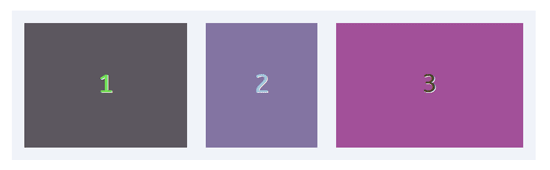
<div class="test\_box">

<div class="list list\_two">1</div>

<div class="list list\_one" style="margin:0 30px;">2</div>

<div class="list list\_w300">3</div>

</div>

结果如下缩略图：  


老大，老二的空间同时被压榨了，老大还好，原本比例高。只是可怜了二当家的，地方越来越小。不过，老二的隐忍换来了和睦，所做的牺牲没有白费。

**四、爸爸其实很厉害，的说~**

语言小知识：“厉害”用日语说的话，动漫里面经常用的比较文雅的就是“すごい”，现在年轻人常用的就是“スゲ”，还有一种说法“よくできるね”是一种称赞的说法，语气比较柔和。

弹性盒子模型下的爸爸（父标签）其实是很有货的，男人嘛，就应该这样，够沉稳够内涵。  
爸爸肚子中的货有：box-orient, box-direction, box-align, box-pack, box-lines. 现在依次讲讲这里box打头的属性都是干嘛用的。

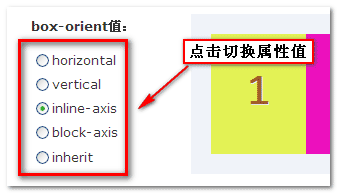
## box-orient

box-orient用来确定子元素的方向。是横着排还是竖着走。可选的值有：

horizontal | vertical | inline-axis | block-axis | inherit

其中，inline-axis是默认值。且horizontal与inline-axis的表现似乎一致的，让子元素横排；而vertical与block-axis的表现也是一致的，让元素纵列。

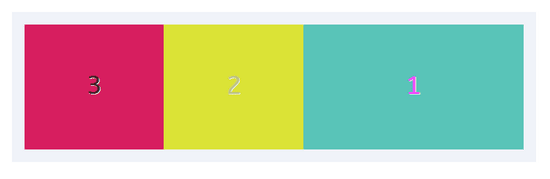
我专门做了个demo页面，方便您查看各个值的行为与表现。您可以狠狠地点击这里：[box-orient值作用测试页面](http://www.zhangxinxu.com/study/201012/css-box-orient-demo.html)

切换demo页面左边的单选选项卡（如果您的浏览器为Firefox/Chrome/Safari），就可以看到不同的box-orient属性值的行为表现了。//zxx:对比可以发现，Firefox下的display:box会收缩外框（有点display:inline-block的感觉），而Chrome则没有收缩。  


box-direction  
box-direction是用来确定子元素的排列顺序，可选值有：

normal | reverse | inherit

其中normal是默认值，表示按照正常顺序排列。所谓正常顺序，就是我们看书写文字的顺序，从左往右，由上至下，先出现的元素，就上面或是左边。而reverse表示反转，原本从左往右应该是1-2-3的，结果显示确实3-2-1。

例如我们将此属性应用在我们一开始的分配房子的demo上的话，最后的显示就会如下缩略图——顺序反过来的：  


相关CSS代码如下：

.test\_box {

display: -moz-box;

display: -webkit-box;

display: box;

-moz-box-direction:reverse;

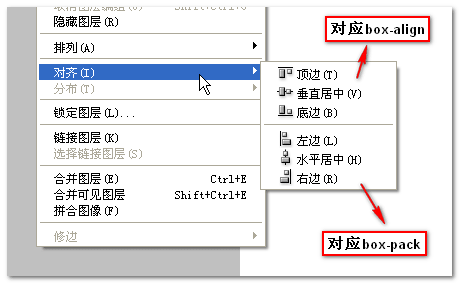
-webkit-box-direction:reverse;

box-direction:reverse;

...

}

您可以狠狠地点击这里：[列表顺序反转显示demo](http://www.zhangxinxu.com/study/201012/css-box-direction-reverse-demo.html)

box-align  
box-align与box-pack都是决定盒子内部剩余空间怎么使用的。在行为效果上就是表现为“对齐”，这跟Adobe的软件中的一些“对齐”是一致的，例如化妆大师photoshop中的图层-对齐：  


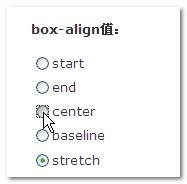
其中box-align决定了垂直方向上的空间利用，也就是垂直方向上的对齐表现。为了便于记忆，我们可以拿来和CSS2中的vertical-align隐射记忆，两者都有”align”，都是都是垂直方向的对齐；而剩下的box-pack就是水平方向的了。

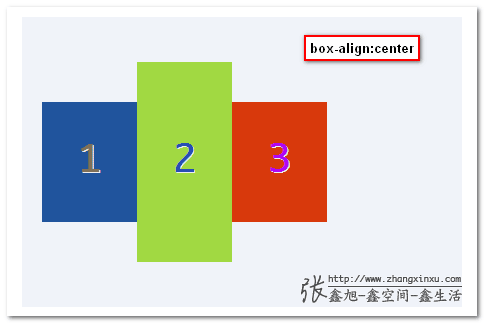
box的可选参数有：

start | end | center | baseline | stretch

其中stretch为默认值，为拉伸，也就是父标签高度过高，其孩子元素的高度就多高，//zxx:以后等高布局不用愁了。start表示顶边对齐，end为底部对齐，center为居中对齐，baseline表示基线（英文字母o,m,n等的底边位置线）对齐。

为了直观的知道各个值的效果，我做了个可实时查看效果的demo，您可以狠狠地点击这里：[css box-align各值效果demo](http://www.zhangxinxu.com/study/201012/css-box-align-demo.html)

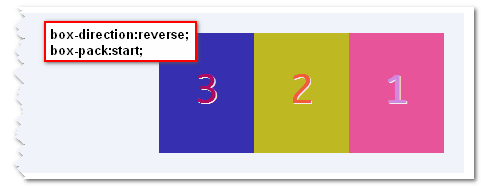
点击demo左边的单选按钮组，即可查看各个属性值的效果。  


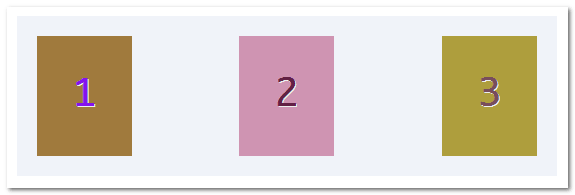
例如，选中”center”这个单选按钮，结果右侧的显示如下面的截图：  


其他各个属性值的效果您可以自己点击查看（非IE浏览器），这里就不一一展示效果截图了。

box-pack  
box-pack决定了父标签水平遗留空间的使用，其可选值有：

start | end | center | justify

就大部分的行为表现来说分别对应text-align属性的值：left | right | center | justify；但是，之所以box-pack不使用”left”, 而是”start”，是因为box-direction属性，这玩意可以反转原本的排列，原本的“左对齐”反转后结果是“右对齐”了，此时”left”显然就词不达意了，所以使用”start”更具有概括性，就是与父标签的起始位置对齐，从而不会产生语义与行为上的困扰。  


其中”start”是box-pack属性的默认值，justify表示两端对齐。  


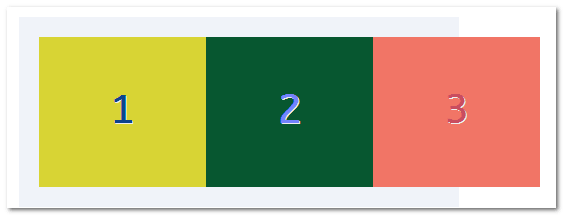
为了方便直观的查看各个属性值的效果，我制作了与上面类似风格的demo，您可以狠狠地点击这里：[box-pack属性值效果demo](http://www.zhangxinxu.com/study/201012/css-box-pack-demo.html)

下为选中end的界面截图缩略图：  


貌似发现在Firefox浏览器下，justify是没有反应的，可能还未支持。Chrome浏览器表现良好。

box-lines  
box-lines是用来决定子元素是可以换行显示呢？还是就算挤出油还是单行显示。两个可选值：

single | multiple

其中single是默认值，表示死活不换行，如下图所示：  


设置box-lines:multiple后，就多行显示了。不过我自己测试了下，貌似现在无论是Firefox浏览器还是Chrome都不认识box-lines:multiple属性，是暂不支持呢，还是什么什么？

您可以狠狠地点击这里：[看不到换行效果的demo](http://www.zhangxinxu.com/study/201012/css-box-lines-demo.html)

## box-flex-group和box-ordinal-group

**五、两个遗漏的属性**

子元素除了box-flex属性，还有两个属性，box-flex-group和box-ordinal-group，其中box-flex-group的作用不详，貌似目前浏览器也不支持；box-ordinal-group的作用是拉帮结派。还是上面马林大叔分房子的例子。小马女友阿凤又不消停，眼瞅着大马的房子面积比自己大好多，心里不平衡，于是，就去拉拢中马，一起打大马房子的主意。这个“拉拢”就是这里的box-ordinal-group，拉拢的组织团伙是有一个数字级别的，决定了你这个组织的位置。

数值越小，位置就越靠前，这不难理解，第一组在最前嘛，随后第二组，第三组… 例如：box-ordinal-group:1的组就会在box-ordinal-group:2的组前面显示。于是，我们可以利用这个属性改变子元素的顺序。例如下面这个例子：  
HTML代码如下：

<div class="test\_box">

<div class="list list\_two">1</div>

<div class="list list\_one">2</div>

<div class="list list\_one">3</div>

</div>

相关CSS如下：

.list\_one {

...

-moz-box-ordinal-group: 1;

-webkit-box-ordinal-group: 1;

box-ordinal-group: 1;

}

.list\_two{

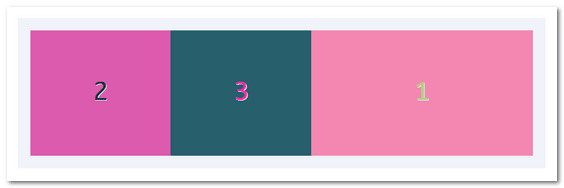
...

-moz-box-ordinal-group: 2;

-webkit-box-ordinal-group: 2;

box-ordinal-group: 2;

}

结果后面两个class为”list\_one”的元素跑到前面去了。如下图所示：  


您可以狠狠地点击这里：[box-ordinal-group分组改序demo](http://www.zhangxinxu.com/study/201012/css-box-ordinal-group-demo.html)

# 宽度计算

弹性盒模型计算

弹性盒模型计算

DIV1

DIV2

DIV3

DIV4

弹性盒模型宽度为0的前提下，如果弹性盒模型内设置了具体宽度，计算盒模型的宽度时，应该用父级的宽度减去具体设置的值，然后剩下的宽度按照flex等分计算。  
也可理解为：父级减去内部所有固定宽度后，按照flex总份数等分，每个盒子再乘以所占份数最后加上自己的固定宽度（margin、padding、border）。

**示例：**

此例设置DIV1、DIV2、DIV3为弹性盒模型，DIV1占2份，DIV2、DIV3各占1份。DIV4为固定宽度200px，左边距60px。所有DIV设置内边距50px，边框20px。父级总宽1000px。  
  
第一步：用父级总宽减去所有固定宽度，4个div共有8个50px的横向padding、8个20px的横向border，1个200px的固定div，以及一个横向60px的margin  
1000 - 50 \* 8 - 20 \* 8 - 200 - 60 = 180  
  
第二步：将剩下的宽度按照flex的份数等分，div1为2、div2为1、div3为1，一共为4份  
180 / 4 = 45  
  
结果：  
DIV1的宽度为 45 \* 2 + 50 \* 2 = 190  
DIV2的宽度为 45 \* 1 + 50 \* 2 = 145  
DIV3的宽度为 45 \* 1 + 50 \* 2 = 145  
DIV4的宽度为 200 + 50 \* 2 = 300

[复制代码](javascript:void(0);)

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<title>弹性盒子模型计算</title>

<style>

body { font-size:1em; line-height:1.5em; padding:50px; color:#333; font-family:"微软雅黑"; }

.box { width:1000px; background:#EBEBEB; border:10px solid #CCC; display:-webkit-box; -webkit-box-orient:horizontal; }

.box div { padding:50px; border:20px solid #00e5ed; color:#FFF; text-shadow:2px 2px 2px rgba(0, 0, 0, .5); width:0; }

.box .div1 { -webkit-box-flex:2; background:#12d698; }

.box .div2 { -webkit-box-flex:1; background:#128ed6; }

.box .div3 { -webkit-box-flex:1; background:#efb017; }

.box .div4 { width:200px; background:gray; margin-left:60px; }

</style>

</head>

<body>

<h1>弹性和模型计算</h1>

<div class="box">

<div class="div1">DIV1</div>

<div class="div2">DIV2</div>

<div class="div3">DIV3</div>

<div class="div4">DIV4</div>

</div>

<p>弹性盒模型宽度为0的前提下，如果弹性盒模型内设置了具体宽度，计算盒模型的宽度时，应该用父级的宽度减去具体设置的值，然后剩下的宽度按照flex等分计算。<br />

也可理解为：父级减去内部所有固定宽度后，按照flex总份数等分，每个盒子再乘以所占份数最后加上自己的固定宽度（margin、padding、border）。

</p>

<h2>示例：</h2>

<p>

此例设置DIV1、DIV2、DIV3为弹性盒模型，DIV1占2份，DIV2、DIV3各占1份。DIV4为固定宽度200px，左边距60px。所有DIV设置内边距50px，边框20px。父级总宽1000px。<br /><br />

第一步：用父级总宽减去所有固定宽度，4个div共有8个50px的横向padding、8个20px的横向border，1个200px的固定div，以及一个横向60px的margin<br />

1000 - 50 \* 8 - 20 \* 8 - 200 - 60 = 180<br /><br />

第二步：将剩下的宽度按照flex的份数等分，div1为2、div2为1、div3为1，一共为4份<br />

180 / 4 = 45<br /><br />

结果：<br />

DIV1的宽度为 45 \* 2 + 50 \* 2 = 190<br />

DIV2的宽度为 45 \* 1 + 50 \* 2 = 145<br />

DIV3的宽度为 45 \* 1 + 50 \* 2 = 145<br />

DIV4的宽度为 200 + 50 \* 2 = 300

</p>

</body>

</html>

[复制代码](javascript:void(0);)

# 垂直居中、水平居中

使用弹性盒子布局（如果需要设置盒子的高度（如为100%），则不能采用这种方式垂直居中！）：

1.整体的外表块元素显示方式为弹性方框

2.框的子代水平排列

3.box-pack和box-align是设置框子元素在框内的排列方式



实现方式非常简单：

1.整体的外表块元素显示方式为弹性方框

2.框的子代水平排列

3.box-pack和box-align是设置框子元素在框内的排列方式

由于div的子元素是水平排列的，所以box-pack设置的是水平方向上子元素的排列方式，这里为center也就是水平居中。而box-align设置的就是垂直方式。

这样就构造了一个块，其内部水平和垂直都居中。

效果图：



这里每个盒子的高度并不是100%！

如果设置了盒子高度，则达不到盒子里的内容垂直居中的效果！

因为box-pack和box-align设置的是盒子的排列方式，而不是盒子里的内容的排列方式！

所以，对于如果需要设置盒子的高度（如为100%），则不能采用这种方式垂直居中！

# 已解决的问题

## **1、指定了box-flex分数的盒子，如果其子元素具有width，则该盒子会被撑大，而不再保持分数比例！**

### 问题描述

.box { width:800px; background:#EBEBEB; border:10px solid #CCC; display:-webkit-box;height:100px; }

.box .div1 { -webkit-box-flex:1; background:#12d698; }

.box .div2 { -webkit-box-flex:1; background:#128ed6; }

<div class="box">

<div class="div1">zzzzzzzzzzzzzzzzzzzzzzzz</div>

<div class="div2"></div>

</div>

运行结果如下所示：



如图所示，div1与div2不是1:1的比例了

### 原因分析

因为div1子元素有宽度了，因此div1宽度=子元素的宽度，从而宽度计算后的值不再等于div2的宽度了。

### 解决方案

将设置了box-flex的div的width设为0