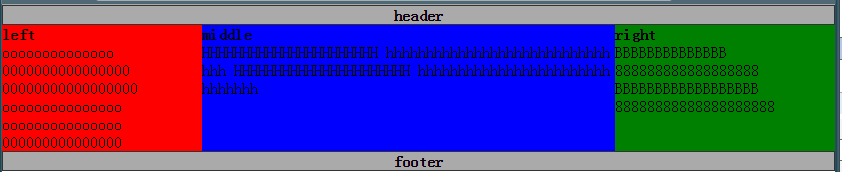
**【圣杯布局】**



在这里，实现了左(200px) 右(220px) 宽度固定，中间自适应，container部分高度保持一致。

[DEMO](http://demoff.sinaapp.com/css_24.html" \t "https://www.cnblogs.com/imwtr/p/_blank)

**稍微说明一下：**

html代码中  middle部分首先要放在container的最前部分。然后是left,right

**1.**将三者都 float:left , 再加上一个position:relative (因为相对定位后面会用到）

**2.**middle部分 width:100%占满

**3.**此时middle占满了，所以要把left拉到最左边，使用margin-left:-100%

**4.**这时left拉回来了，但会覆盖middle内容的左端，要把middle内容拉出来，所以在外围container加上 padding:0 220px 0 200px

**5.**middle内容拉回来了，但left也跟着过来了，所以要还原，就对left使用相对定位 left:-200px  同理，right也要相对定位还原 right:-220px

**6.**到这里大概就自适应好了。如果想container高度保持一致可以给left middle right都加上min-height:130px

不过衰衰地发现ie中有问题.. ie6/7/8/9中 下面的空白高度都不一样..



当然，为了保证窗口不能缩太小无法展示左右，可以给body加上 min-width

**好了，代码敬上：**

[IMG_258](https://www.cnblogs.com/imwtr/p/javascript:void(0);)

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"><html xmlns="http://www.w3.org/1999/xhtml"><head><meta charset="utf-8"><meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1"><title>圣杯布局</title><style type="text/css">

\*{margin: 0;padding: 0;}

body{min-width: 700px;}

.header,

.footer{

border: 1px solid #333;

background: #aaa;

text-align: center;

}

.left,

.middle,

.right{

position: relative;

float: left;

min-height: 130px;

}

.container{

padding:0 220px 0 200px;

overflow: hidden;

}

.left{

margin-left: -100%;

left: -200px;

width: 200px;

background: red;

}

.right{

margin-left: -220px;

right: -220px;

width: 220px;

background: green;

}

.middle{

width: 100%;

background: blue;

word-break: break-all;

}

.footer{

clear: both;

}</style></head><body><div class="header">

<h4>header</h4></div><div class="container">

<div class="middle">

<h4>middle</h4>

<p>HHHHHHHHHHHHHHHHHHHHHH

hhhhhhhhhhhhhhhhhhhhhhhhhhhhhhh

HHHHHHHHHHHHHHHHHHHHHH

hhhhhhhhhhhhhhhhhhhhhhhhhhhhhhh

</p>

</div>

<div class="left">

<h4>left</h4>

<p>oooooooooooooo

0000000000000000

00000000000000000

ooooooooooooooo

ooooooooooooooo

000000000000000</p>

</div>

<div class="right">

<h4>right</h4>

<p>BBBBBBBBBBBBBB

888888888888888888

BBBBBBBBBBBBBBBBBB

88888888888888888888</p>

</div>

</div>

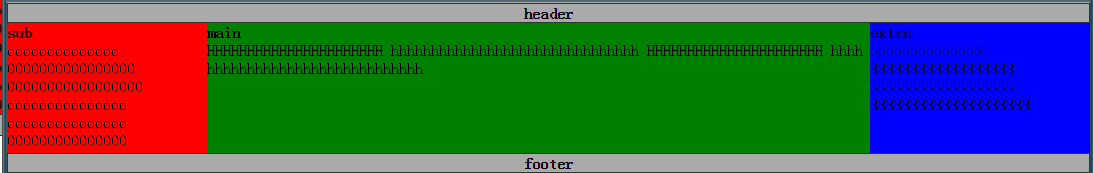
<div class="footer">

<h4>footer</h4>

</div></body></html>

[IMG_259](https://www.cnblogs.com/imwtr/p/javascript:void(0);)

**【双飞翼布局】**

****

[DEMO](http://demoff.sinaapp.com/css_25.html" \t "https://www.cnblogs.com/imwtr/p/_blank)

听说双飞翼布局，始于淘宝UED

如果把三栏布局比作一只大鸟，可以把main看成是鸟的身体，sub和extra则是鸟的翅膀。这个布局的实现思路是，先把最重要的身体部分放好，然后再将翅膀移动到适当的地方.

其实跟上边的圣杯布局差不多的，当然也可以改动一下（自己想想有哪些不同吧）

**恩，这里有一只鸟~**

左翅sub有200px,右翅extra..220px.. 身体main自适应未知

**1.**html代码中，main要放最前边，sub  extra

**2.**将main  sub  extra 都float:left

**3.**将main占满 width:100%

**4.**此时main占满了，所以要把sub拉到最左边，使用margin-left:-100%  同理 extra使用margin-left:-220px

（这时可以直接继续上边圣杯布局的步骤，也可以有所改动）

**5.**main内容被覆盖了吧，除了使用外围的padding，还可以考虑使用margin。

给main增加一个内层div-- main-inner, 然后margin:0 220px 0 200px

**6.**main正确展示

代码敬上：

[IMG_261](https://www.cnblogs.com/imwtr/p/javascript:void(0);)

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"><html xmlns="http://www.w3.org/1999/xhtml"><head><meta charset="utf-8"><meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1"><title>双飞翼布局</title><style type="text/css">

\*{margin: 0;padding: 0;}

body{min-width: 700px;}

.header,

.footer{

border: 1px solid #333;

background: #aaa;

text-align: center;

}

.sub,

.main,

.extra{

float: left;

min-height: 130px;

}

.sub{

margin-left: -100%;

width: 200px;

background: red;

}

.extra{

margin-left: -220px;

width: 220px;

background: blue;

}

.main{

width: 100%;

}

.main-inner{

margin-left: 200px;

margin-right: 220px;

min-height: 130px;

background: green;

word-break: break-all;

}

.footer{

clear: both;

}</style></head><body><div class="header">

<h4>header</h4></div>

<div class="main">

<div class="main-inner">

<h4>main</h4>

<p>HHHHHHHHHHHHHHHHHHHHHH

hhhhhhhhhhhhhhhhhhhhhhhhhhhhhhh

HHHHHHHHHHHHHHHHHHHHHH

hhhhhhhhhhhhhhhhhhhhhhhhhhhhhhh

</p>

</div>

</div>

<div class="sub">

<h4>sub</h4>

<p>oooooooooooooo

0000000000000000

00000000000000000

ooooooooooooooo

ooooooooooooooo

000000000000000</p>

</div>

<div class="extra">

<h4>extra</h4>

<p>BBBBBBBBBBBBBB

888888888888888888

BBBBBBBBBBBBBBBBBB

88888888888888888888</p>

</div>

<div class="footer">

<h4>footer</h4>

</div></body></html>

**单列布局水平居中**

水平居中的页面布局中最为常见的一种布局形式，多出现于标题，以及内容区域的组织形式，下面介绍四种实现水平居中的方法（注：下面各个实例中实现的是child元素的对齐操作，child元素的父容器是parent元素）

**使用inline-block 和 text-align实现**

.parent{text-align: center;}.child{display: inline-block;}

优点：兼容性好；  
不足：需要同时设置子元素和父元素

**使用margin:0 auto来实现**

.child{width:200px;margin:0 auto;}

优点：兼容性好  
缺点: 需要指定宽度

**使用table实现**

.child{display:table;margin:0 auto;}

优点:只需要对自身进行设置  
不足:IE6,7需要调整结构

**使用绝对定位实现**

.parent{position:relative;}/\*或者实用margin-left的负值为盒子宽度的一半也可以实现，不过这样就必须知道盒子的宽度，但兼容性好\*/.child{position:absolute;left:50%;transform:translate(-50%);}

不足：兼容性差,IE9及以上可用

**实用flex布局实现**

/\*第一种方法\*/.parent{display:flex;justify-content:center;}/\*第二种方法\*/.parent{display:flex;}.child{margin:0 auto;}

缺点：兼容性差，如果进行大面积的布局可能会影响效率

**垂直居中**

**vertical-align**

我们都知道，每个人都有不同的嗜好，有的人喜欢吃甜食，有的人喜欢吃辣的东西，有的人不喜欢吃芹菜，有的人不喜欢吃羊肉等等。CSS中的有些元素也是这样，他们有的只对牛奶感兴趣，有的只喜欢吃坚果和果冻，而讨厌牛奶。而vertical-align呢，是个比较挑食的家伙，它只喜欢吃果冻，从小吃果冻长大，没有了果冻，它就会闹脾气，对你不理不睬。我称之为“果冻依赖型元素”，又称之为“inline-block依赖型元素”，也就是说,只有一个元素属于inline或是inline-block（table-cell也可以理解为inline-block水平）水平，其身上的vertical-align属性才会起作用。我对css-vertical-align的一些理解与认识

在使用vertical-align的时候，由于对齐的基线是用行高的基线作为标记，故需要设置line-height或设置display:table-cell;

/\*第一种方法\*/.parent{display:table-cell;vertical-align:middle;height:20px;}/\*第二种方法\*/.parent{display:inline-block;vertical-align:middle;line-height:20px;}

**实用绝对定位**

.parent{position:relative;}.child{positon:absolute;top:50%;transform:translate(0,-50%);}

**实用flex实现**

.parent{display:flex;align-items:center;}

**水平垂直全部居中**

利用vertical-align,text-align,inline-block实现

.parent{display:table-cell;vertical-align:middle;text-align:center;}.child{display:inline-block;}

**利用绝对定位实现**

.parent{position:relative;}.child{position:absolute;top:50%;left:50%;transform:translate(-50%,-50%);}

**利用flex实现**

.parent{display:flex;justify-content:center;align-items:center;}

**多列布局左列定宽，右列自适应**

该布局方式非常常见，适用于定宽的一侧常为导航，自适应的一侧为内容的布局  


**利用float+margin实现**

.left{float:left;width:100px;}.right{margin-left:100px;}

注：IE6会有3px的bug

**利用float+margin(fix)实现**



<div class="parent">

<div class="left"></div>

<div class="right-fix">

<div class="right"></div>

</div></div>

.left{width:100px;float:left;}.right-fix{width:100%;margin-left:-100px;float:right;}.right{margin-left:100px;}

**使用float+overflow实现**

.left{width:100px;float:left;}.right{overflow:hidden;}

overflow:hidden，触发bfc模式，浮动无法影响，隔离其他元素，IE6不支持，左侧left设置margin-left当作left与right之间的边距，右侧利用overflow:hidden 进行形成bfc模式  
如果我们需要将两列设置为等高，可以用下述方法将“背景”设置为等高，其实并不是内容的等高

.left{width:100px;float:left;}.right{overflow:hidden;}.parent{overflow:hidden;}.left,.right{padding-bottom:9999px;margin-bottom:-9999px;}

**使用table实现**

.parent{display:table;table-layout:fixed;width:100%;}.left{width:100px;}.right,.left{display:table-cell;}

**实用flex实现**

.parent{display:flex;}.left{width:100px;}.right{flex:1;}

利用右侧容器的flex:1，均分了剩余的宽度，也实现了同样的效果。而align-items 默认值为stretch，故二者高度相等

**右列定宽，左列自适应**

**实用float+margin实现**

.parent{background:red;height:100px;margin:0 auto;}.left{background:green;margin-right:-100px;width:100%;float:left;}.right{float:right;width:100px;background:blue;}

**使用table实现**

.parent{display:table;table-layout:fixed;width:100%;}.left{display:table-cell;}.right{width:100px;display:table-cell;}

**实用flex实现**

.parent{display:flex;}.left{flex:1;}.right{width:100px;}

**两列定宽，一列自适应**



基本html结构为父容器为parent,自容器为left,center,right.其中，left,center定宽，right自适应

**利用float+margin实现**

.left,.center{float:left:width:200px;}.right{margin-left:400px;}

**利用float+overflow实现**

.left,.center{float:left:width:200px;}.right{overflow:hidden;}

**利用table实现**

.parent{display:table;table-layout:fixed;width:100%;}.left,.center,.right{display:table-cell;}.left,.center{width:200px;}

**利用flex实现**

.parent{display:flex;}.left,.center{width:100px;}.right{flex:1}

**两侧定宽，中栏自适应**



**利用float+margin实现**

.left{width：100px;float:left;}.center{float:left;width:100%;margin-right:-200px;}.right{width:100px;float:right;}

**利用table实现**

.parent{width:100%;display:table;table-layout:fixed}.left,.center,.right{display:table-cell;}.left{width:100px;}.right{width:100px;}

**利用flex实现**

.parent{display:flex;}.left{width:100px;}.center{flex:1;}.right{width:100px;}

**一列不定宽，一列自适应**



**利用float+overflow实现**

.left{float:left;}.right{overflow:hidden;}

**利用table实现**

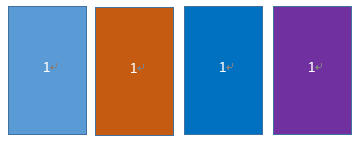
.parent{display:table;table-layout:fixed;width:100%;}.left{width:0.1%;}.left,.right{display:table-cell;}

**利用flex实现**

.parent{display:flex;}.right{flex:1;}

**多列等分布局**

多列等分布局常出现在内容中，多数为功能的，同阶级内容的并排显示等。



html结构如下所示

<div class="parent">

<div class="column">1</div>

<div class="column">1</div>

<div class="column">1</div>

<div class="column">1</div></div>

**实用float实现**

.parent{margin-left:-20px}/\*假设列之间的间距为20px\*/.column{float:left;width:25%;padding-left:20px;box-sizing:border-box;}

**利用table实现**

.parent-fix{margin-left:-20px;}.parent{display:table;table-layout:fixed;width:100%;}.column{display:table-cell;padding-left:20px;}

**利用flex实现**

.parent{display:flex;}.column{flex:1;}.column+.column{margin-left:20px;}

**九宫格布局**

**使用table实现**

<div class="parent">

<div class="row"><div class="item"></div><div class="item"></div><div class="item"></div></div>

<div class="row"><div class="item"></div><div class="item"></div><div class="item"></div></div>

<div class="row"><div class="item"></div><div class="item"></div><div class="item"></div></div>

</div>

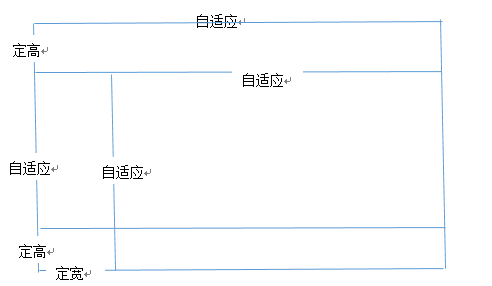
.parent{display:table;table-layout:fixed;width:100%;}.row{display:table-row;}.item{display:table-cell;width:33.3%;height:200px;}

**实用flex实现**

<div class="parent"><div class="row"><div class="item"></div><div class="item"></div><div class="item"></div></div><div class="row"><div class="item"></div><div class="item"></div><div class="item"></div></div><div class="row"><div class="item"></div><div class="item"></div><div class="item"></div></div></div>

.parent{display:flex;flex-direction:column;}.row{height:100px;display:flex;}.item{width:100px;background:red;}

**全屏布局**



**利用绝对定位实现**

<div class="parent"><div class="top">top</div><div class="left">left</div><div class="right">right</div><div class="bottom">bottom</div></div>

html,body,parent{height:100%;overflow:hidden;}.top{position:absolute:top:0;left:0;right:0;height:100px;}.left{position:absolute;top:100px;left:0;bottom:50px;width:200px;}.right{position:absolute;overflow:auto;left:200px;right:0;top:100px;bottom:50px;}.bottom{position:absolute;left:0;right:0;bottom:0;height:50px;}

**利用flex实现**

<div class="parent"><div class="top">top</div><div class="middle"><div class="left">left</div><div class="right">right</div></div><div class="bottom">bottom</div></div>

.parent{display:flex;flex-direction:column;}.top{height:100px;}.bottom{height:50px;}.middle{flex:1;display:flex;}.left{width:200px;}.right{flex:1;overflow:auto;}

**响应式布局**

**meta标签的实用**

设置布局宽度等于设备宽度，布局viewport等于度量viewport

<meta name="viewport" content="width=device-width,initial-scale=1">

**媒体查询**

HTML 4和CSS 2目前支持为不同的媒体类型设定专有的样式表, 比如, 一个页面在屏幕上显示时使用无衬线字体,  
而在打印时则使用衬线字体, screen 和 print 是两种已定义的媒体类型, 媒体查询让样式表有更强的针对性,  
扩展了媒体类型的功能;媒体查询由媒体类型和一个或多个检测媒体特性的条件表达式组成,  
媒体查询中可用于检测的媒体特性有width、height和color（等）, 使用媒体查询, 可以在不改变页面内容的情况下,  
为特定的一些输出设备定制显示效果。

**语法**

@media screen and (max-width:960px){....}<link rel="stylesheet" media="screen and (max-width:960px)" href='xxx.css' />

<https://leaverou.github.io/css3patterns/>

渐变网站

<http://css3generator.com/>

<https://enjoycss.com/>

<http://www.patternify.com/>

<https://www.colorzilla.com/gradient-editor/>

<https://www.cssmatic.com/>

<http://csstypeset.com/>

<https://www.css3buttongenerator.com/>

<https://www.bestcssbuttongenerator.com/>

10种css生成工具