**初始包含块：**视窗大小的矩形框

**定位：**定位都是根据包含块定位的

看css的属性：首先看默认值，在看继承性，position是比较重要，写在前面

width没给值就是auto

left top bottom right width height 没给都是auto

margin padding 没给值都是0

**百分比：**

margin padding 都是相对于包含块的width

top相对于包含块的height

left相对于包含块的width、

浮动会提升半级

最小宽度：防止变形，min-width：2\*left+right

**圣杯布局**

要求：1.两边固定 当中自适应（中间设置100%）

2.当中列要完整显示（中间部分都是不变形）

3.当中列要优先加载（middle要先显示，放在left和right前面）

**浮动:** 搭建完整的布局框架（三个都浮动，这时候要给content清除浮动）

margin 为赋值:调整旁边两列的位置(使三列布局到一行上)（浮动是一条的 要left到最左头 就-100%，right在右就-200px）

使用相对定位:调整旁边两列的位置（使两列位置调整到两头）（给content加一个padding左右就是两边的width，然后就relation来）

**双飞翼：**

前面步骤都和圣杯一样，在后面处理两边的时候，不要动，就给middle套一个m-inner然后设置m-inner的padding像内200px机可以了

<!--两组实现的对比:

1.俩种布局方式都是把主列放在文档流最前面，使主列优先加载。

2.两种布局方式在实现上也有相同之处，都是让三列浮动，然后通过负外边距形成三列布局。

3.两种布局方式的**不同之处**在于如何处理中间主列的位置：

圣杯布局是利用父容器的左、右内边距+两个从列相对定位；

双飞翼布局是把主列嵌套在一个新的父级块中利用主列的左、右外边距进行布局调整

height:100%要给HTML和body一起加才行，才会逐级击沉下去

清除浏览器系统默认滚动条

html{

margin: 30px;

border: 1px solid;

height: 100%;

overflow: scroll; }

body{

margin: 30px;

border: 1px solid pink;

height: 100%;

overflow: scroll; }

**使用绝对定位来模拟固定定位**

html{

overflow: hidden;

height: 100%;

}

body{

overflow: auto;

height: 100%;

}

#test{

position: absolute;

left: 50px;

top: 50px;

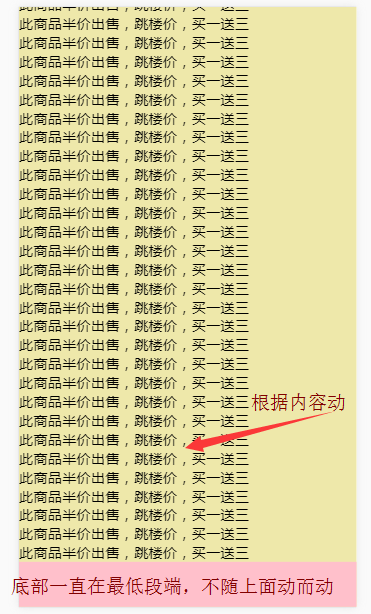
width: 100px;

height: 100px;

background: pink;

}

**粘贴布局：**

1.先建立一个div，再嵌div拿来写内容，好分内容和底部

2.body，html设置高度为100%便于下面最小高度继承100%

3.首先上面的大div要设置一个padding-botton：底部的高度

4.底部的设置一个margin-top往上，就会固定在这里面

body,html{

height:100%;

}

.header{

background-color: palegoldenrod;

min-height: 100%;

width: 100%;

padding-bottom: 50px;

}

#footer{

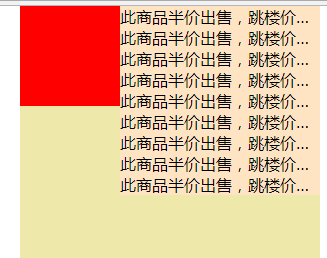
background-color: pink;

height: 50px;

margin-top: -50px; }

这个要加overflow或者清除浮动，为了之后布局里面有浮动会高度塌陷

**两列布局：**

1. 这是类似商品列表，两列布局
2. **左边浮动，右部开启BFC(over)**
3. 右边的冒号，是为了防止字数超过,如下，缺一不可。

overflow: hidden;

white-space: nowrap;

text-overflow: ellipsis;

**BFC：**

**BFC是什么**

BFC(Block formatting context)直译为"块级格式化上下文"。它是一个独立的渲染区域，只有Block-level box参与，

它规定了内部的Block-level Box如何布局，并且与这个区域外部毫不相干

**BFC布局规则：**

1.内部的Box会在垂直方向，一个接一个地放置。（独占一行）

2.BFC的区域不会与float box重叠。（如两列布局）

3.内部的Box垂直方向的距离由margin决定。属于同一个BFC的两个相邻Box的margin会发生重叠（解决重叠）,

给下一个盒子加一个外层的div开启overflow是两个BFC L

去除相邻：magrin：给父元素加一个border就分开了，

4.计算BFC的高度时，浮动元素也参与计算。(清除浮动 haslayout)

5.BFC就是页面上的一个隔离的独立容器，容器里面的子元素不会影响到外面的元素。反之也如此。

**BFC什么时候出现(哪些元素会生成BFC?**

根元素

float属性不为none

position为absolute或fixed

overflow不为visible

display为inline-block, table-cell, table-caption, flex, inline-flex

清除浮动

1、给父级加高度 (扩展性不好)

2、给父级加浮动 (页面中所有元素都加浮动，margin左右自动失效（floats bad ！）)

3、空标签清除浮动 (IE6 最小高度 19px；（解决后IE6下还有2px偏差）)

4、.br清浮动 (不符合工作中：结构、样式、行为，三者分离的要求)

5、overflow

6、伪元素after

**检测低版本的方法:**

先获得这个元素，然后再给他写版本，在写内容

console.log(isIE(8));

//js中的作用域都是函数作用域

function isIE(version){

var b = document.createElement("b");

b.innerHTML="<!--[if IE "+version+"]><i></i><![endif]-->";

return b.getElementsByTagName("i").length == 1 ;

}

**左右查询：**

**右查询：**对等号非左边变量的查询

在整个作用域链中，如果没有找到变量的声明，直接抛ReferenceError错误。

**左查询：**对等号左边变量的查询

在整条作用域链中，如果没有找到变量的声明，js引擎会自动在全局声明一个同名变量

殊的右查询(typeof 的安全机制)

例子：

name();

function name(){

var b = 7

a = b;//如果没有写 var 那么a就会是全部的变量，如果有var 才是函数内部自己的变量

console.log(a) }

console.log(a)

**元素垂直水平居中**

* **已知长宽的垂直水平居中（用绝对定位以及他的特性来自动设置padding居中的值）**

**绝对定位盒子的特性（**高宽有内容撑开**）margin设置为auto,其余的上下左右设置为0**

**水平方向上：** left + right + width + padding + margin = 包含块padding区域的尺寸

0 0 100 0 auto 400（包含块的宽）

**垂直方向上：** top + bottom + height + padding + margin = 包含块padding区域的尺寸

0 0 100 0 0 600（包含块的高）

#inner{

position: absolute;

**left: 0;**

**right: 0;**

**top: 0;**

**bottom: 0;**

**margin: auto;**

width: 100px;

height: 100px;

background: deeppink;

}

* **未知高度的元素水平垂直居中方案**

#inner{

position: absolute;

**left: 50%;**

**top: 50%;//先定位，在根据内容往回走来调试位置**

**transform: translate3d(-50%,-50%,0);//但是这个的浏览器兼容性不好**

background: deeppink;

}

* 绝对定位50%+margin的负值移动（根据元素本身尺寸来确定偏移量）

Vertical-align用来指定元素的垂直对齐方式**默认值：**baseline 不可继承

**用途：**经常用img（替换元素、占位元素）上使用Vertical-align属性，只对inline-block有用，img高宽自适应

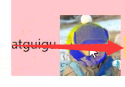
大部分取值相对于父元素来说

**基线选取：**元素基线与父元素的基线对齐，没有给对齐方式。上中下都是以基线为主，没有规定基线可替换元素以外边框的底线为基线

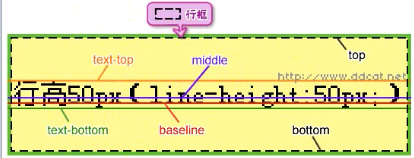
**行高最高：**一行上垂直对称时一行高值最大的行内框为基线，其他行内采用自己的对齐方式，最终计算行框的高度

**text-top：和服元素的文字顶部对齐**

text-bottom 和父元素的文字的底部对齐

middle元素的中线与父元素的字母的中线对齐

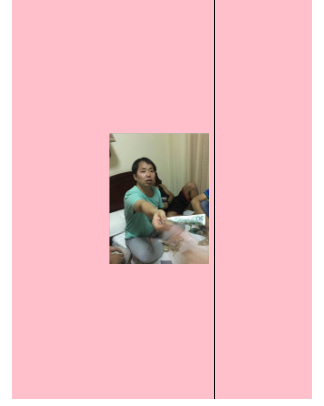
图解分析各属性



应用场景：设置图片的水平垂直居中

让一行上所有的元素在垂直方向上对齐（一般对齐是按照里面最高的元素对齐）

**分析：**加伪类里面的线是为了让vertical-align有最高的对齐线，先加空内容，然后变为行内块元素，到后面把背景色去除就可以了，就当做没有了



.box:after{

content: "";

display: inline-block;

height: 100%;

width: 1px;

background-color: black;

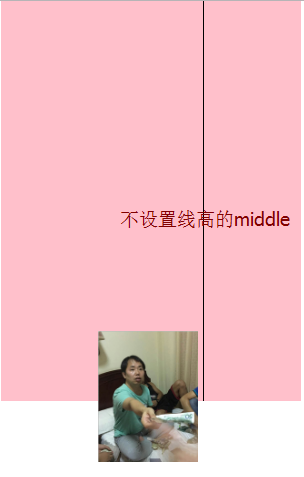
vertical-align: middle;//不设置,下边的img就会按照图片的中间位置对齐如图（1）

}

img{

width: 100px;

vertical-align: middle;//不设置图片的底部就会对齐上面中间位置的先,因为不是文字,所以就不会有文字的四条线，如图（2）}

图（1） 图（2）