1position的值为absolute、fixed的元素脱离文档流，static、relative没有脱离文档流

static定位不脱离文档流，设置left/top/right/bottom没有作用，对margin/padding敏感。

父子节点都是没有脱离文档的两种定位（static、relative）的外边距（margin）会合并，显示效果以最大的那个外边距为准。

如果有内边距影响，那么子元素的外边距会基于父元素的内边距来偏移

**行内元素的的左右外边距不会合并**

可见行内元素对定位top/right/bottom/left、宽高width/height、外边距margin不敏感，对边框和内边距敏感。

**外边距会重叠的情况**

相邻的盒模型中，如果其中的一个是浮动的（float），垂直margin不会重叠，并且浮动的盒模型和它的子元素之间也是这样。

设置了overflow属性的元素和它的子元素之间的margin不被重叠（overflow取值为visible除外）。

设置了绝对定位（position:absolute）的盒模型，垂直margin不会被重叠，并且和他们的子元素之间也是一样。

设置了display:inline-block的元素，垂直margin不会重叠，甚至和他们的子元素之间也是一样。

2浮动

**浮动布局**

<style>
.container>div{
height: 200px;
}
.left {
float: left;
width: 300px;
background: red;
}

这种布局主要是控制元素浮动来实现的，要注意的一点就是中间元素要创建BFC（关于BFC的相关内容后面文章会分析）,否则一旦高度变化就会无法正常工作。这种布局的特点就是浮动对旧浏览器兼容性好，缺点就是主体内容需要放到最后加载，当页面元素较多时候可能会影响体验，于是有了下面两种非常经典的改进方案。dom中的center要放到最后 注意浮动会使元素脱离文档流

**圣杯布局**

.container {
padding: 0 300px;

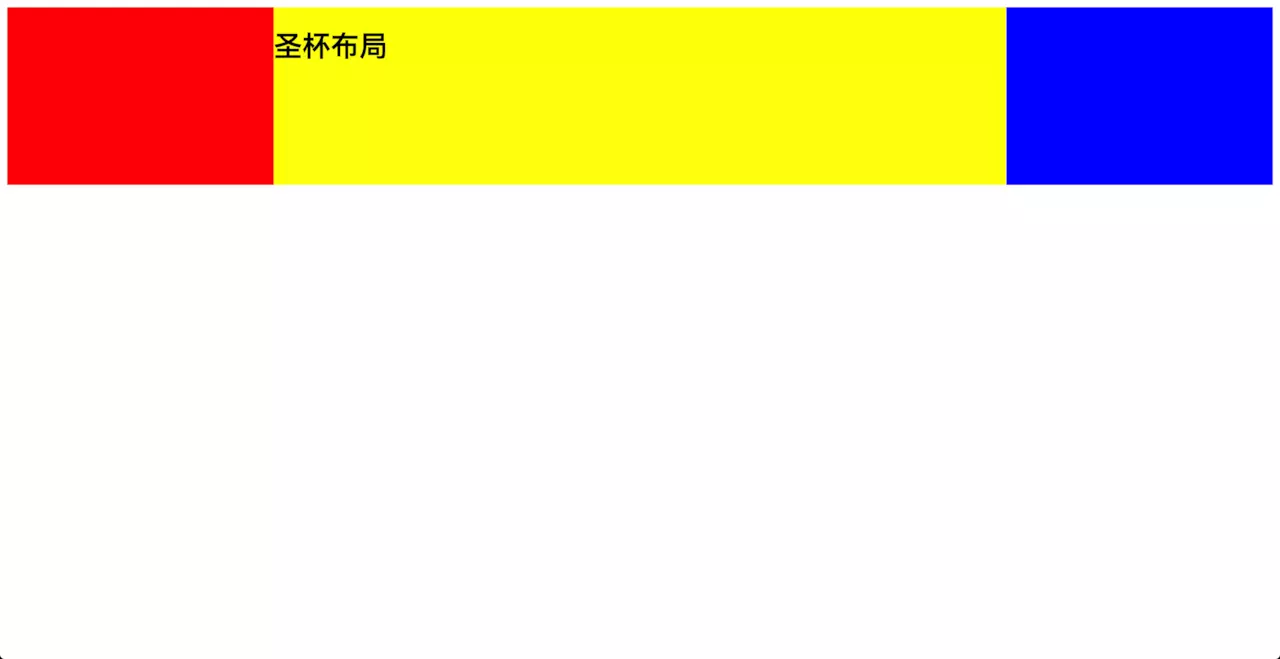


}

.container>div {
height: 200px;
}
.center {
float: left;
width: 100%;
background: yellow;
}
.left {
float: left;
width: 300px;
background: red;

**margin-left:-100%；//负的中间元素的宽度**

**position: relative;//**



left: -300px;
}
.right {
float: left;
width: 300px;
background: blue;

**margin-left:-300%;//负的自己的宽度**

**position: relative;**

right: -300px;
}

这样就实现了圣杯布局，圣杯布局保持了与普通浮动布局同样的兼容性，最大的优点是可以，实现中间部分优先加载，缺点就是处理复杂，而且当中间元素小于两侧元素时候会出现变形，布局混乱，响应效果相对差一点。

**双飞翼布局**

<div class="container">
<div class="center-container">
<div class="center">
<h1>双飞翼布局</h1>
</div

<style>
.container div {
height: 200px;
}
.center-container {
float: left;
width: 100%;
height: 100px;
background: yellow;
}
.center {
margin: 0 300px;//设置文字的位置 }
.left {
background: red;
float: left;
width: 300px;
margin-left: -100%;
}

.right {
background: blue;
float: left;
width: 300px;
margin-left: -300px;
}
</style>

圣杯布局和双飞翼布局都是在浮动布局时代的比较经典的布局方式，对旧的浏览器有很好的兼容性。不过事实上，现代浏览器已经大量普及，需要适配旧浏览器的场景已经开始变少，加上移动端开发越来越盛行，于是有了比较新的布局方式。

flex布局

flex是css3提供的一种新的布局方式，这种布局的产生就是为了实现自适应布局，它是随着移动互联网时代产生而引进的

<style>
.container {
display: flex;
}
.container>div {
height: 200px;
}
.left {
width: 300px;
background: red;
}
.center {
flex: 1;
background: yellow;
}
.right {
width: 300px;
background: blue;
}
</style>
<div class="container">
<div class="left"></div>
<div class="center">

**table布局**

table布局其实我们已经很熟悉了，就是表格布局，那么表格布局是这样实现三栏自适应的效果呢？其实不难理解,就是把三列都看做是表格，控制表格的显示情况即可

.container {
width: 100%;
display: table;
height: 200px;
}
.container>div {
display: table-cell;

布局有缺点，缺点就是不灵活，边框设置、高度设置等等都有很大受限。

**绝对定位布局**

绝对定位布局，很容易，效率也很高，不过实际开发中很少使用，原因也很简单，绝对定位的元素是脱离文档流的，可维护性会受限。