布局

2017年5月2日 9:52

等高布局：

圣杯

双飞翼

1 块水平垂直居中

块水平垂直居中

- Span 高度与父盒子高度一致，inline-block，中线对齐vertical-align:middle;
- left：50%
  Top:50%
  Margin-top:-fu;
  Margin-left:-fu;

  Right:50%;
  Bottom:50%;
  Margin-right:-fu;
  Margin-bottom:-fu;
  ;
- left:0;
  top:0;
  right:0;
  bottom:0;
  margin:auto

等高布局：

等高布局是指子元素在父元素中高度相等的布局方式。等高布局的实现包括伪等高和真等高，伪等高只是看上去等高而已，真等高是实实在在的等高。

来自 <http://blog.csdn.net/libin_1/article/details/51318708>

伪等高

边框模拟

因为元素边框和元素高度始终是相同高度，用元素的边框颜色来伪装左右两个兄弟元素的背景色。然后将左右两个透明背景的元素使用absolute覆盖在中间元素的左右边框上，实现视觉上的等高效果

[注意]左右两侧元素的内容高度不能大于中间元素内容高度，否则无法撑开容器高度

```
<style>
body,p{margin: 0;}
```

```
.parent{
    position: relative;
}
.center{
    box-sizing:border-box;      规定两个并排的带边框的框
    padding: 0 20px;
    background-clip: content-box;
    border-left: 210px solid lightblue;
    border-right: 310px solid lightgreen;
}
.left{
    position: absolute;
    top: 0;
    left: 0;
    width: 200px;
}
.right{
    position: absolute;
    top: 0;
    right: 0;
    width: 300px;
}
</style>
```

```
<div class="parent" style="background-color: lightgrey;">
    <div class="left">
        <p>left</p>
    </div>
    <div class="center" style="background-color: pink;">
        <p>center</p>
        <p>center</p>
    </div>
    <div class="right">
        <p>right</p>
    </div>
</div>
```

负margin

因为背景是在padding区域显示的，设置一个大数值的padding-bottom，再设置相同数值的负的margin-bottom，使背景色铺满元素区域，又符合元素的盒模型的计算公式，实现视觉上的等高效果

[注意]如果页面中使用<a>锚点跳转时，将会隐藏部分文字信息

[注意]如果页面中的背景图片定位到底部，将会看不到背景图片

```
<style>
body,p{margin: 0;}
.parent{
    overflow: hidden;
}
.left,.centerWrap,.right{
    float: left;
```

```
    width: 50%;
    padding-bottom: 9999px;
    margin-bottom: -9999px;
}
.center{
    margin: 0 20px;
}
.left,.right{
    width: 25%;
}
</style>
```

```
<div class="parent" style="background-color: lightgrey;">
    <div class="left" style="background-color: lightblue;">
        <p>left</p>
    </div>
    <div class="centerWrap">
        <div class="center" style="background-color: pink;">
            <p>center</p>
            <p>center</p>
        </div>
    </div>

    <div class="right" style="background-color: lightgreen;">
        <p>right</p>
    </div>
</div>
```

真等高

table

table元素中的table-cell元素默认就是等高的

```
<style>
body,p{margin: 0;}
.parent{
    display: table;
    width: 100%;
    table-layout: fixed;
}
.left,.centerWrap,.right{
    display: table-cell;
}
.center{
    margin: 0 20px;
}
</style>
```

```
<div class="parent" style="background-color: lightgrey;">
    <div class="left" style="background-color: lightblue;">
```

```
            <p>left</p>
        </div>
        <div class="centerWrap">
            <div class="center" style="background-color: pink;">
                <p>center</p>
                <p>center</p>
            </div>
        </div>
        <div class="right" style="background-color: lightgreen;">
            <p>right</p>
        </div>
</div>
```

absolute

设置子元素的top:0;bottom:0;使得所有子元素的高度都和父元素的高度相同，实现等高效果

```
<style>
body,p{margin: 0;}
.parent{
    position: relative;
    height: 40px;
}
.left,.center,.right{
    position: absolute;
    top: 0;
    bottom: 0;
}
.left{
    left: 0;
    width: 100px;
}
.center{
    left: 120px;
    right: 120px;
}
.right{
    width: 100px;
    right: 0;
}
</style>
```

```
<div class="parent" style="background-color: lightgrey;">
    <div class="left" style="background-color: lightblue;">
        <p>left</p>
    </div>
    <div class="center" style="background-color: pink;">
        <p>center</p>
        <p>center</p>
    </div>
    <div class="right" style="background-color: lightgreen;">
        <p>right</p>
    </div>
</div>
```

flex

flex中的伸缩项目默认都拉伸为父元素的高度，也实现了等高效果

```
<style>
body,p{margin: 0;}
.parent{
    display: flex;
}
.left,.center,.right{
    flex: 1;
}
.center{
    margin: 0 20px;
}
</style>
```

```
<div class="parent" style="background-color: lightgrey;">
    <div class="left" style="background-color: lightblue;">
        <p>left</p>
    </div>
    <div class="center" style="background-color: pink;">
        <p>center</p>
        <p>center</p>
    </div>
    <div class="right" style="background-color: lightgreen;">
        <p>right</p>
    </div>
</div>
```

js

当子元素高度不同时，进行js判断，增加较低子元素的padding-bottom，使得各个子元素实现等高效果

```
<style>
body,p{margin: 0;}
.parent{overflow: hidden;}
.left,.center,.right{
    float: left;
    width: 25%;
}
.center{
    width: 50%;
    padding: 0 20px;
    background-clip: content-box;
    box-sizing: border-box;
}
</style>
```

```
<div class="parent" id="parent" style="background-color: lightgrey;">
    <div class="left" style="background-color: lightblue;">
        <p>left</p>
    </div>
    <div class="center" style="background-color: pink;">
        <p>center</p>
        <p>center</p>
    </div>
    <div class="right" style="background-color: lightgreen;">
        <p>right</p>
    </div>
</div>
```

```
<script>
function getCSS(obj,style){
    if(window.getComputedStyle){
        return getComputedStyle(obj)[style];
    }
    return obj.currentStyle[style];
}
var oParent = document.getElementById('parent');
var oLeft = oParent.getElementsByTagName('div')[0];
var oCenter = oParent.getElementsByTagName('div')[1];
var oRight = oParent.getElementsByTagName('div')[2];
function eqHeight(obj1,obj2){
    var oDis = obj1.clientHeight - obj2.clientHeight;
    if(oDis > 0){
        obj2.style.paddingBottom = parseFloat(getCSS(obj2,'padding-
bottom')) + oDis + 'px';
    }else{
        obj1.style.paddingBottom = parseFloat(getCSS(obj1,'padding-
bottom')) +  Math.abs(oDis) + 'px';
    }
}
eqHeight(oLeft,oCenter);
eqHeight(oLeft,oRight);
</script>
```

来自 <http://blog.csdn.net/libin_1/article/details/51318708>

来自 <http://blog.csdn.net/libin_1/article/details/51318708>

双飞翼：

仔细分析各种布局的技术实现，可以发现下面三种技术被经常使用：

浮动 float

负边距 negative margin

相对定位 relative position

这是实现布局的三个最基本的原子技术。只要巧妙组合，并加以灵活运用，就能“拼”出各种布局的实现方案。

来自 <http://www.imooc.com/wenda/detail/254035>

双飞翼布局主要解决俩问题：1、三列布局，中间宽度自适应，两边定宽； 2、中间栏要在浏览器中优先展示渲染。

三列布局的实现方式就是让三列外面套一个 ，如下代码：

```
<div id="container">
    <div id="center_div" class="column">主要内容</div>
    <div id="left_div" class="column"></div>
    <div id="right_div" class="column"></div>
</div>
```

浏览器是按照你写html的顺序渲染的，所以把中间div要写到前面，这就满足了问题里的第二个。但是正常是，写到前面也会显示到前面，咋办，别急，接着看：

然后让这三列div都浮动起来, float:left，

```
.column{
    float: left;
}
```

接着，让中间列div宽度占满整个宽度100%（宽度设成百分数是相对于父元素的宽度，所以container的100%是浏览器宽度，center_div宽度就是container全宽度，也等于浏览器宽度）：

```
#container{
    width: 100%;
}
#center_div{
    width: 100%;
}
```

浮动的特点就是这一行占满了就既不进来别的元素了，只要宽度能挤下，就会挤进来。中间列div不是占了全部了吗，挤不下怎么办，办法是让左右两个div覆盖在上面，这样就挤下了，这就是负外边距的作用。

所以，

再接着，把左边div设置负边距margin-left:-100%和固定width（假设是300px），负边距的作用就是让左边div盖在中间div上面，设成100%就会盖在中间div最左边。（这个100%是指的外面div，也就是那个id="container"的宽度）

```
#left_div{
    width: 300px;
    margin-left: 100%;
}
```

右边div设置固定width，假设是200px，负边距margin-left:200px和这它的固定宽度一致，这样会盖在最中间div右边。

```
#right_div{
    width: 200px;
    margin-left: 200px;
}
```

----------分割线----------

到这一步为止，如果你给每个div加上颜色，会看到它们已经形成了三列，但是问题在于，中间div的内容被挡住了，

所以最后一步，就是在中间div里再创建一个子div，让这个子div的左右外边距分别等于左边div和右边div的固定宽度，

如下：

```
<div id="container">
    <div id="center_div" class="column">
    <div id="mainWrap">主要内容</div>
  <div id="left_div" class="column"></div>
  <div id="right_div" class="column"></div>
</div>
```

假设这个子div的id=mainWrap，那么CSS按照上面的取值就是这么写：

```
#center_div  #mainWrap{
```

```
    margin-left: 300px;
    margin-right: 200px;
}
```

OK，这样就完工了。你在各个div里放入一些文字，缩放一下浏览器窗口看看，左右宽度是不变的，中间的内容会随着内容的多少让网页高度变化。

来自 <https://zhidao.baidu.com/question/2267561297621475628.html>

CSS布局中圣杯布局与双飞翼布局的实现思路差异在哪里？

来自 <https://www.zhihu.com/question/21504052>

作者：知乎用户
链接：https://www.zhihu.com/question/21504052/answer/50053054
来源：知乎

圣杯布局的来历是2006年发在a list part上的这篇文章：
In Search of the Holy Grail · An A List Apart Article
圣杯是西方表达“渴求之物"的意思，不是一种对页面的形象表达。
双飞翼据考源自淘宝UED，应该是一种页面的形象的表达。
圣杯布局和双飞翼布局解决的问题是一样的，就是两边顶宽，中间自适应的三栏布局，中间栏要在放在文档流前面以优先渲染。
圣杯布局和双飞翼布局解决问题的方案在前一半是相同的，也就是三栏全部float浮动，但左右两栏加上负margin让其跟中间栏div并排，以形成三栏布局。
不同在于解决”中间栏div内容不被遮挡“问题的思路不一样：
圣杯布局，为了中间div内容不被遮挡，将中间div设置了左右padding-left和padding-right后，将左右两个div用相对布局position: relative并分别配合right和left属性，以便左右两栏div移动后不遮挡中间div。
双飞翼布局，为了中间div内容不被遮挡，直接在中间div内部创建子div用于放置内容，在该子div里用margin-left和margin-right为左右两栏div留出位置。
多了1个div，少用大致4个css属性（圣杯布局中间divpadding-left和padding-right这2个属性，加上左右两个div用相对布局position: relative及对应的right和left共4个属性，一共6个；而双飞翼布局子div里用margin-left和margin-right共2个属性，6-2=4），个人感觉

比圣杯布局思路更直接和简洁一点。
简单说起来就是”双飞翼布局比圣杯布局多创建了一个div，但不用相对布局了“，而不是你题目中说的”去掉relative"就是双飞翼布局“。