这可能是史上最全的CSS自适应布局总结

2016-05-13 前端大全

（点击上方公众号，可快速关注）

> 作者：茄果
> 链接：http://www.cnblogs.com/qieguo/p/5421252.html

标题严格遵守了新广告法，你再不爽，我也没犯法呀！屁话不多说，直入！

所谓布局，其实包含两个含义：尺寸与定位。也就是说，所有与尺寸和定位相关的属性，都可以用来布局。

大体上，布局中会用到的有：尺寸相关的盒子模型，普通流、浮动、绝对定位三种定位机制，CSS3中的transform、弹性盒子模块、试验中的grid模块。逛园子的时候经常可以看到浮动布局，inline-block布局，弹性盒布局这几个名词。现在对布局也算有一点了解，做个总结巩固一下。如果你也看了很多资料，但是实际动手时对布局还是无从下手的话，希望本文可以帮你理清思路。

唠叨一句：看到一个效果图的时候，千万不要急着手贱去敲代码！先思考清楚页面的构造，理清各元素之间的关系，特别需要注意的是在不同的设备下需要有怎样的展现，当你思路清晰找到最好的布局方案时，coding其实真的不需要多少时间。

尺寸相关

为什么要先说尺寸呢？因为尺寸在布局中的作用非常核心，布局方式定位这些只是改变了元素之间的关系，没有尺寸就什么也不是。比如我们通常会用margin来控制跟其他元素的距离，这就是布局。

很多人都会觉得，什么width、margin太简单了，早就掌握了。这种心态我一开始学习CSS的时候也有，觉得很好理解很简单，但是后面才发现自己原来很多东西都没真正掌握。看看张鑫旭大神给我们上的政治课：http://www.zhangxinxu.com/wordpress/2012/07/bottleneck-css-study/

先说说百分比，百分比是相对父对象的，这里特性非常好用，很多时候会用在自适应布局上面。浏览器尺寸的改变，就是根节点html的长宽改变，我们可以用%来将浏览器尺寸和元素尺寸联系起来，做到自适应。

另外一个比较有意思的是auto，auto是很多尺寸值的默认值，也就是由浏览器自动计算。首先是块级元素水平方向的auto，块级元素的margin、border、padding以及content宽度之和等于父

元素width。使用auto属性在父元素宽度变化的时候，该元素的宽度也会随之变化。

但是当该元素被设为浮动时，该元素的width就变成了内容的宽度了，由内容撑开，也就是所谓的有了包裹性。overflow | position:absolute | float:left/right都可以产生包裹性，替换元素也同样具有包裹性。在具有包裹性的元素上想利用width : auto；来让元素宽度自适应浏览器宽是不行的。

高度方向：外边距重叠，外边距auto为0，这两点需要注意。书写方向什么的，接触比较少就不扯了。

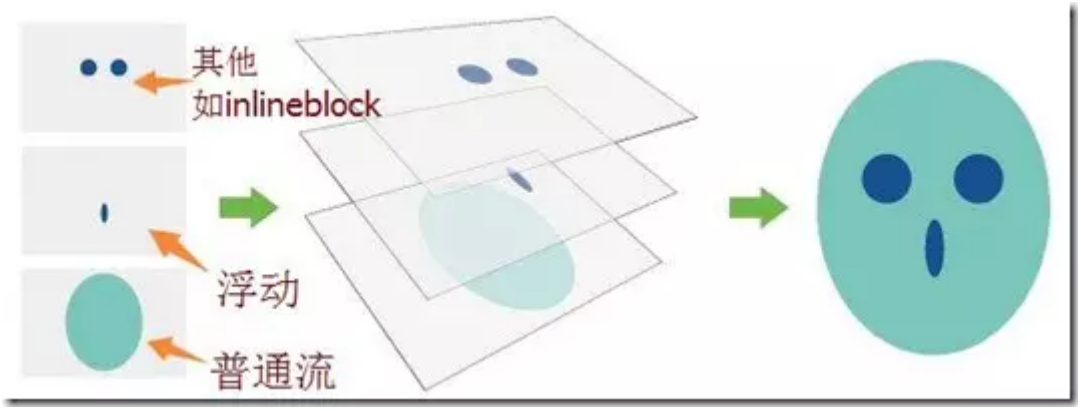
那为什么margin：auto对不能计算垂直方向的值呢？很简单，垂直方向是被设计成可以无限扩展的，内容越多浏览器便产生滚动条来扩展，所以垂直方向都找不到一个计算基准，以此返回一个false，便成了0。

用处：通过width、height控制大小，各个方向的margin值控制与边界或者其他元素的距离来定位。

浮动

目前PC网站大多使用float布局，从成本上考虑大改的概率很小，所以不要说浮动无用，总是会有机会让你维护的！代表网站：淘宝、腾讯、百度，好吧BAT都到齐了。

浮动听得多了，博客园上关于用浮动布局的介绍也非常的多。浮动原本用于文本环绕，但却在布局被发扬光大，这就是命！我的理解：浮动布局的核心就是让元素脱离普通流，然后使用width/height，margin/padding将元素定位。脱离普通流的元素，就像脱离地心引力一样，与普通流不在一个高度上。这个跟图层的概念类似。高度不同所以可以叠在其他元素上面产生重叠或者使用负边距跑到父元素外，理解了这一点浮动布局就很好理解了。



下面用个圣杯布局的例子说明一下，理解了这个之后其他布局更加简单：

left，宽度固定,高度可固定也可由内容撑开
right，宽度固定，高度可固定也可由内容撑开
center，可以自适应浏览器宽度，高度可固定也可由内容撑开。
HTML & CSS：

```html
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>宽度自适应布局</title>
        <style>
            .wrap {
                background-color: #D66464;
            }
            .clearfix:after {
                content: "";
                clear: both;
                display: block;
            }
            .left {
                float: left;
                width: 100px;
                background: #00f;
                height: 180px;
            }
            .right {
                float: right;
                width: 150px;
                background: #0f0;
                height: 200px;
            }
            .center {
                background: #FFFFFF;
                margin-left: 110px;
                margin-right: 160px;
                height: 150px;
            }
        </style>
    </head>
    <body>
        <div class="wrap clearfix">
            <div class="left">left，宽度固定，高度可固定也可以由内容撑开。</div>
            <div class="right">right，宽度固定，高度可固定也可以由内容撑开。</div>
            <div class="center">center，可以自适应浏览器宽度，高度可固定也可以由内容撑开。</div>
        </div>
    </body>
</html>
```

原理非常简单，左右侧边栏定宽并浮动，中部内容区放最后不浮动、默认width：auto并设置相应外边距，让左右侧边栏浮动到上面。注意：子元素设置为浮动之后，父对象的高度就坍塌了，需要设置父对象后的元素清除浮动，这样父对象的高度才能被浮动子元素撑起来了。

当然，我们也要问一下，为啥父对象高度会坍塌呢？上面也说过了，浮动元素已经脱离了普通流，父对象所在的普通流比喻成地表，那浮动元素就已经上天了。但是父对象还在地表啊，从外太空看浮动元素在父对象里面，但是其实并不在，又怎么能撑开父对象呢？宽度如果我们不设置的话，其实也是为0的，因为父对象里面空空如也，所以宽高为0。

要撑开的办法就两个，1是让父对象也上天（。。。你咋不上天呢），2是把浮动元素的边框边界拉下来。

父对象也上天（即浮动）的话，那就不能实现宽度自适应了。因为float元素的width：auto是包裹内容的，参考前面说的！

办法2就是在后面的元素里加一个clear语句。说到这个问题就要扯到clear与BFC了，我就不献丑了。传送门：https://developer.mozilla.org/zh-CN/docs/Web/CSS/clear

这个三列布局还有个双飞（是双飞翼！想啥呢）的变种，就是在HTML中center部分也就是内容区提到最前面，也就是内容先行渲染。在网络不好的时候，左右双翼能不能出来不要紧，先让主体内容出来！这种想法，明显的优秀工程师思维，但，尼玛的双翼都是广告啊。广告不出来，哪能赚钱养你们这群工程师？所以提出双飞的玉伯才离开了淘宝？？？（纯属意淫，如真属实，当我扯淡，哈哈哈！）

先上码：

```html
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>宽度自适应布局</title>
        <style>
            .wrap {
                background-color: #FBD570;
                margin-left: 100px;
                margin-right: 150px;
            }
            .clearfix:after {
                content: "";
                clear: both;
                display: block;
            }
            .left {
                float: left;
                width: 100px;
                background: #00f;
                height: 180px;
                margin-left: calc(-100% - 100px);
            }
            .right {
                float: right;
                width: 150px;
                background: #0f0;
                height: 200px;
                margin-right: -150px;
            }
            .center {
                background: #B373DA;
                height: 150px;
                float: left;
                width: 100%;
            }
        </style>
    </head>
    <body>
        <div class="wrap clearfix">
            <div class="center">center，可以自适应浏览器宽度，高度可固定也可以由内容撑开。</div>
            <div class="left">left，宽度固定，高度可固定也可以由内容撑开</div>
            <div class="right">right，宽度固定，高度可固定也可以由内容撑开</div>
        </div>
    </body>
</html>
```

思路：

1)既然HTML里面要让center放前面，为了让left跑到center前面，那center也必须浮动了，否则因为都是块元素他们会分两行。

2)浮动之后还要让center宽度自适应，那明显width只能100%，然后在父元素中设width:auto，还有两侧margin，其实也就是父对象宽度自适应，center只是继承content的宽度。

3)对left使用负的margin让他们浮动到上方去。

代码里面我用到了一个calc()，这个CSS3带来的计算函数简直酷毙了！本例里如果不使用calc函数，那么就需要wrap左边距为0，left左边距-100%，然后center多加一层子块DIV设置margin-left：100px，可以达到同样的效果！calc函数与百分比配合就足以实现自适应的要求！目前所有的自适应布局都在利用浏览器来为我们计算尺寸，但是有了calc之后我们就可以自己制定规则！单是想想都高潮了吧？

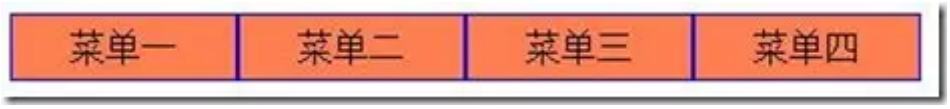
总结：使用浮动来进行布局，一个比较大的问题是清除浮动。这个可以使用一个after伪类来清除。更大的问题是浮动性像水一样向上流动，难以把握。在元素较多而且元素高度尺寸不一的情况下，单纯使用浮动只能实现上端对齐，这对于适应多种设备的布局就显得力不从心了。目前的做法是牺牲一部分内容，将元素做成等高排列，从美观上看也当然也是极好的，比参差不齐的排列要美观。

普通流布局

普通流布局：display : inline-block！这是一个传说中取代float布局的存在。看了一些网站，PC端浮动为主，移动端的也用的不多啊，已经有些使用flex的了，说好的inline-block一统江湖呢？

使用inline-block之前先处理点小障碍：inline-block元素会有4px左右的空隙，这个是因为我们写代码时候的换行符所致。

解决办法很简单：在inline-block的父元素中设置样式font-size：0；letter-spacing: -4px; 然后设置inline-block的所有兄弟元素 font-size：值；letter-spacing: 值px；恢复正常的显示。

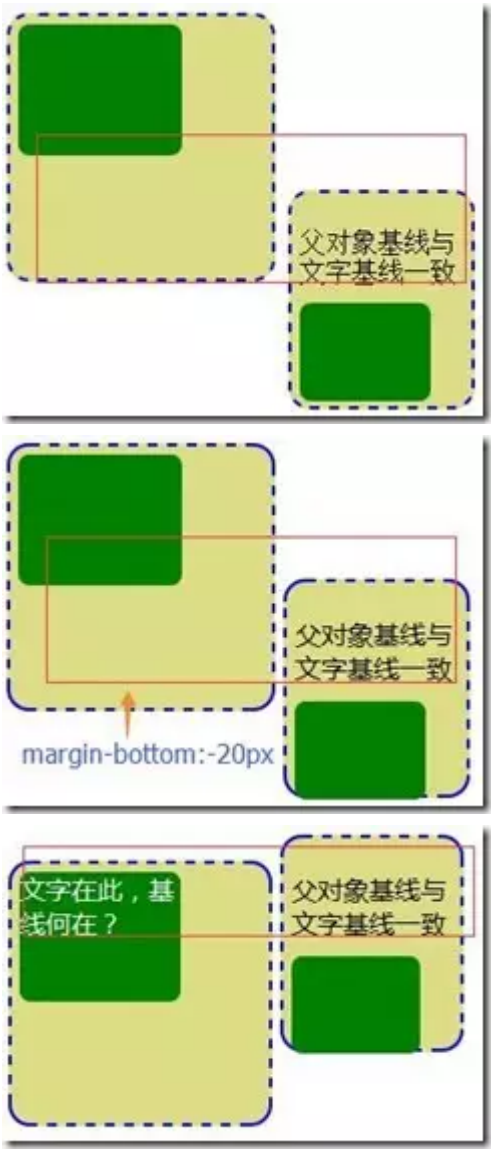


另外还有一点需要注意的是inline-block默认是基线对齐的，而inline-block的基线又跟文本基线一致，所以在内容不同的时候并不能水平对齐。只需要用vertical-align显式声明一下top/bottom/middle对齐即可。这里补充一下基线的内容，没你想的那么简单哦。分有文字和无文字两种情况：

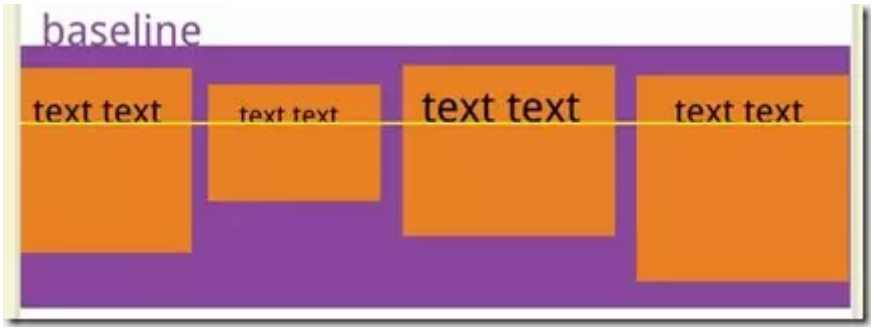
1）无文字：容器的margin-bottom下边缘。与容器内部的元素没一毛钱关系。

2）有文字：最后一行文字的下边缘，跟文字块（p,h等）的margin、padding没关系！注意是最后一行，无论文字在什么子对象容器内在什么位置都没关系，浏览器会找到最后一行文字对齐底部。

你们感受一下：



警示：inline-block的基线是最后一行文字的底部，flex里面的基线是第一行文字的底部（请看下文阮老师的文章）



满满的都是泪啊。。。既然都叫baseline，何必呢？

使用inline-block进行圣杯布局：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>宽度自适应布局</title>
        <style>
            .wrap {
                background-color: #FBD570;
                font-size: 0;
                letter-spacing: -4px;  /*用于兼容safari，根据不同字体字号或许需要做一定的调整*/
                margin-left: 100px;
                margin-right: 150px;
            }
            .wrap * {
                font-size: 1rem;
                letter-spacing: normal;
            }
            .left {
                display: inline-block;
                vertical-align: top;
                width: 100px;
                background: #00f;
                height: 180px;
                margin-left: -100px;
            }
            .right {
                display: inline-block;
                vertical-align: top;
                   width: 150px;
                background: #0f0;
                height: 200px;
                margin-right: -150px;
            }
            .center {
                display: inline-block;
                vertical-align: top;
                background: #B373DA;
                height: 150px;
                min-width: 150px;
                width: 100%;
            }
        </style>
    </head>
    <body>
        <div class="wrap">
            <div class="left">left，宽度高度固定</div>
            <div class="center">center，可以自适应浏览器宽度，高度固定。</div>
            <div class="right">right，宽度高度固定</div>
        </div>
    </body>
</html>
```

这里也没什么好说的，用到的也是width：auto和width：100%这两点，简单知识点的简单用法。

双飞的话，代码跟圣杯的基本相同，注意在html的顺序变为center>right>left，只改左栏移动的margin-left: calc(-100% – 100px)到预定位置即可。不能用calc的话可以在center里面再加一层，跟浮动一样的处理方式。更简单的方法是使用CSS3带给我们的box-sizing属性。请看代码：

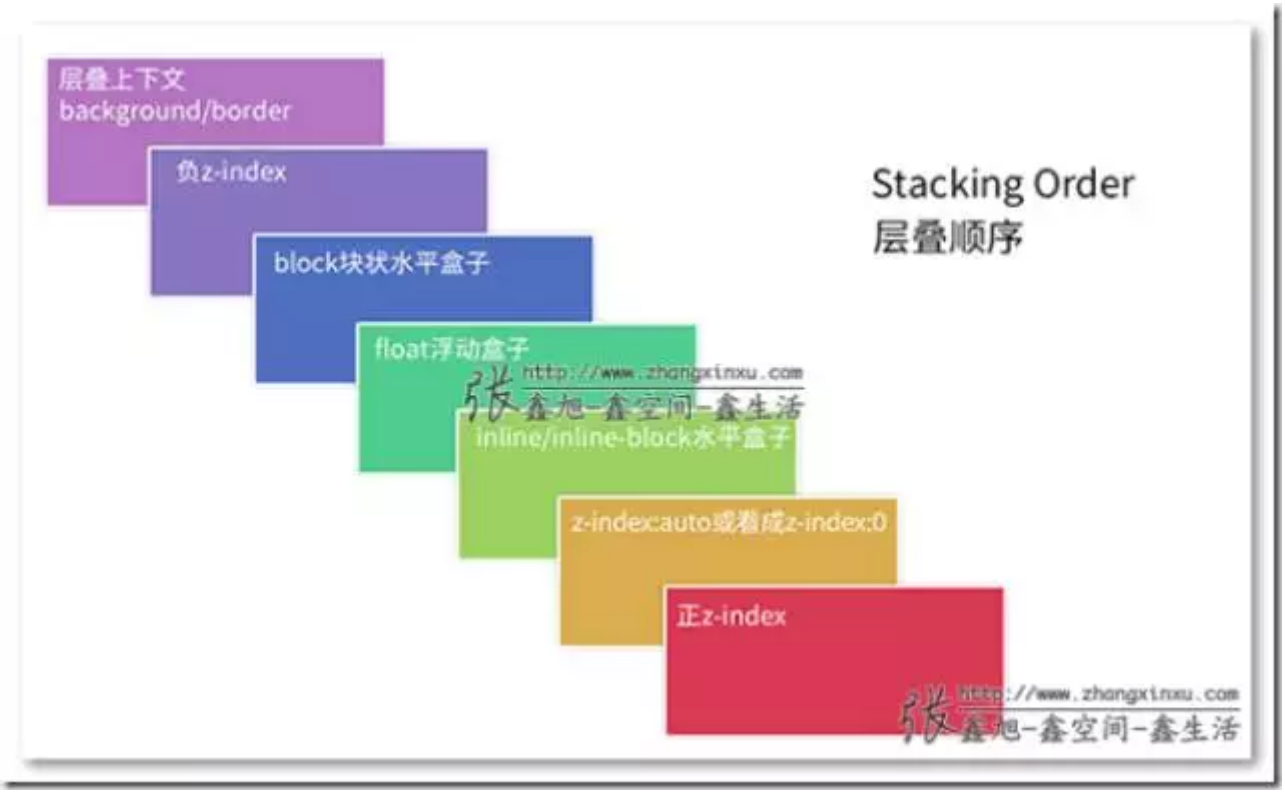
```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>宽度自适应布局</title>
        <style>
            .wrap {
                background-color: #FBD570;
                font-size: 0;
                letter-spacing: -4px;  /*用于兼容safari，根据不同字体字号或许需要做一定的调整*/
                margin-right: 150px;
            }
            .wrap * {
                font-size: 1rem;
                letter-spacing: normal;
            }
            .left {
                display: inline-block;
                vertical-align: top;
                width: 100px;
                background: #00f;
                height: 180px;
                margin-left: -100%;
            }
            .right {
                display: inline-block;
                vertical-align: top;
                   width: 150px;
                background: #0f0;
                height: 200px;
                margin-right: -150px;
            }
            .center {
                display: inline-block;
                vertical-align: top;
                background: #B373DA;
                height: 150px;
                min-width: 150px;
                width: 100%;
                box-sizing: border-box;
                padding-left: 100px;
                background-origin: content-box;
                background-clip: content-box;
            }
        </style>
    </head>
    <body>
        <div class="wrap">
            <div class="center">
                center，可以自适应浏览器宽度，高度固定。
            </div>
            <div class="right">right，宽度高度固定</div>
            <div class="left">left，宽度高度固定</div>
        </div>
    </body>
</html>
```

总结：相比浮动inline-block更加容易理解，也更符合我们的认知，结合盒子模型的几个控制属性就可以进行布局了。对于元素高度不同的情况，目前浮动布局的做法都是将元素做成等高元素进行展现，这从美学上看也符合整齐的要求，不过牺牲了一部分内容。但inline-block有vertical-align属性，可以很好地解决元素高度不同而带来的布局问题。用过之后，你也会喜欢上inline-block的。。。至少我会！

绝对定位

前面的浮动和普通流中其实定位都是靠盒子模型控制的，与我们常说的定位还是有差别的。而绝对定位就是我们平常所说的定位，给定参考坐标系+坐标确定位置。关于绝对定位的资料太多，我就不说了。提一点就是absolute定位的基准是最近的非static定位父对象，而fixed是相对html根节点的定位。两种定位都会脱离普通流，跟之前说的浮动一样，上天了。

当然，他们跟浮动在空间中的位置还是有差别的，项目中有遇到这个问题的请参考张大婶的文章： http://www.zhangxinxu.com/wordpress/2016/01/understand-css-stacking-context-order-z-index/ 还是要结合项目来看，否则看过也只是看过而已，并不会存到你的脑子里，毕竟还是相当抽象相当理论性的东西。借用张大神的一个总结图：



使用绝对定位（特指absolute）做自适应布局跟前面两种方式没太大差别，宽度自适应还是在auto和100%上做文章，而位置则由top/bottom/left/right等控制。还是以圣杯布局来举例：

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>宽度自适应布局</title>
    <style>
      .wrap {
        position: relative;
        background-color: #FBD570;
        margin-left: 100px;
        margin-right: 150px;
        height: 250px;
      }
      .left {
        position: absolute;
        top: 0;
        left: -100px;
        width: 100px;
        background: #00f;
        height: 180px;
      }
      .right {
        position: absolute;
        top: 0;
        right: 0;
          width: 150px;
        background: #0f0;
        height: 200px;
        margin-right: -150px;
      }
      .center {
        position: absolute;
        top: 0;
        left: 0;
        background: #B373DA;
        height: 150px;
        min-width: 150px;
```

```
            width: 100%;
        }
    </style>
</head>
<body>
    <div class="wrap">
        <div class="center">
            center，可以自适应浏览器宽度，高度固定。
        </div>
        <div class="left">left，宽度高度固定</div>
        <div class="right">right，宽度高度固定</div>
    </div>
</body>
</html>
```

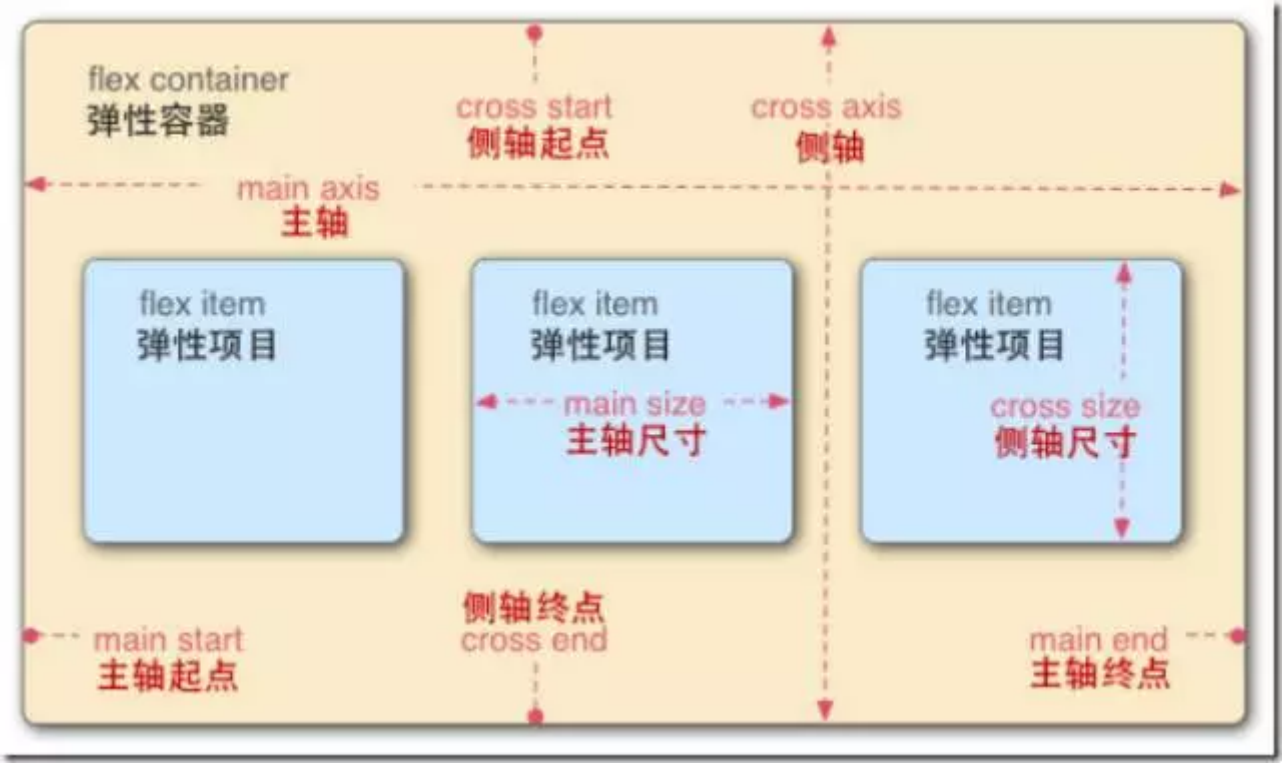
父元素为relative，子元素为absolute，这样的话，又会出现跟浮动一样的问题：父对象高度坍塌，子元素不能撑起父对象。原因也跟浮动一样，解决办法的话目前我知道的只有给父对象指定一个确定height值，大家如果有更好的办法，请联系我！

总结：单纯使用绝对定位进行自适应布局的情况很少，一般绝对定位都用在尺寸固定的元素定位上。而且fixed定位的渲染效率很低，因为它会频繁触发浏览器的重排。另外提一点：CSS3的transform会对绝对定位产生影响哦~比如说让fixed定位不再固定在浏览器视窗的黑魔法：http://www.zhangxinxu.com/wordpress/2015/05/css3-transform-affect/

弹性盒子

CSS3中对布局影响最大的莫过于弹性盒子模块了，这是一套区别于以往盒子模型布局的全新方案。上面几种方法你可以看到，为了实现自适应我们用的都是width：auto和100%的嵌套以及各种边距的移动定位，这套规则并不符合我们的认知。为什么不能开拓出一块区域，横竖排列都可以，内部所有元素的尺寸可以按照一个规则和这个区域的大小联系起来？终于CSS3做出了改变，引入了flex弹性布局方案，弹性盒布局有如下优势：
1.独立的高度控制与对齐。
2.独立的元素顺序。
3.指定元素之间的关系。
4.灵活的尺寸与对齐方式。

在MDN上有非常简单易懂的基础教程：https://developer.mozilla.org/zh-CN/docs/Web/CSS/CSS_Flexible_Box_Layout/Using_CSS_flexible_boxes



上面也已经给出了圣杯布局的自适应布局方案，所以代码就不贴了不过这个例子实现的是3栏成比例缩放，左右栏如果需要固定值的话可以写成 flex: 0 0 150px; 的样式。

但是上面的教程没有给出各个属性的详细解释，建议看看阮一峰的博文，详细易懂而且配图超漂亮的有木有：http://www.ruanyifeng.com/blog/2015/07/flex-grammar.html

总结：弹性盒子在移动端的应用会越来越普遍，这套模型值得去好好研究。语法规则都是非常贴近人性，非常灵活，浏览器兼容性也非常好，当然国内百花齐放的移动浏览器会有哪些大坑呢？我们拭目以待~

其他

其他包括position：relative和CSS3中的transform都可以实现定位，但是由于他们在原来的普通流中还占着一个坑，所以很少用来布局啥的。transform是个很酷炫的东西，可以用平面的素材做出很多3D的效果，而且不需要js就可以做，非常好玩。此文已经很长，就不多说了，以后会写一篇文章来专门说说她的故事。

【今日微信公号推荐↓】