1. 盒模型
   1. 基础

**1.文档流**

（1）将窗体自上而下分成一行行的，并在每行中按从左至右的顺序排放元素，即为文档流。

（2）绝对定位、固定定位、浮点，可以使元素脱离文档流而存在。

**2.三类元素**

（1）顶级元素：包括html/body/frameset，表现如块级元素，属于高级块级元素。

（2）块级元素：有如下特征：1）独占1行，即总是从新的1行开始，前后会带有换行符。2）宽度缺省时为包含块宽度的100%，与内容无关。即使设置了width，且小于包含块的width，它也会独占1行。3）如果没有内容，高度为0，但仍然独占1行，只是什么都不会显示。4）所有尺寸可设置。

（3）内联元素：有如下特征：1）与其他内联级元素在同1行，直到该行满行。2）设置width/height无效，width根据文字或图片宽度而定。3）设置margin/padding/border，水平方向有效，垂直方向不会撑开行高，但padding/border会显示，只是覆盖其他行（重叠）。4）设置margin：auto不能居中，得在父元素设置text-align：center。

（4）可以使用display转换块级元素和内联元素。display有1个重要的属性值：inline-block，可以使元素既像内联元素一样不会独占1行，又能像块级元素设置高、宽、边距等，且能撑开line-height。

**3.BFC和IFC**

BFC意为块级格式化上下文，IFC意为内联元素格式化上下文。主要内容就是上面的块级元素独占1行、折叠边距；内联元素不能独占1行，不能设置width等内容。可以参考：https://www.w3ctech.com/topic/865。

**4.box分类**

（1）浏览器根据文本生成“内容区域”，内容区域组成“匿名box”。匿名box和内联box组成“线box”。线box和块box构成“文档流”，进而形成整个页面。

（2）块box：块级元素可生成块box。

（3）内联box：内联元素生成内联box。

（4）line（线）box：在包含块内，一行就是一个线box，由这一行的内联box组成，如此，内联box分成了若干线box。线box有如下特点：1）层叠的线box之间没有空间。2）线box总是有足够的高度来适应其包含的全部内联box（线box的高度见后面line-height和vertical-align一节）。

（5）匿名box：如：“<p>这是一行普通的文字，这里有个<em>em</em>标签。</p>”。em将生成内联box。整个<p>将生成块box。而块box中除em外的部分内容将生成匿名box。匿名box将继承其封闭的非匿名块父元素的属性。

（6）内容区域：内容区域是一种围绕文字看不见的box（每个文字都有），其大小与font-size大小相关。

**5.包含块**

（1）根元素html的包含块为初始包括块，为浏览器可见区域（而不是全部区域）。

（2）根元素以外的包含块由其position属性值决定。如果为static/relative，其父元素的内容box边界构成其包含块。如果为absolute，为其定位祖先元素（包括absolute、fixed及relative）的填充边界为包含块。如果为fixed，则为视口或页面区域。

* 1. 专题
     1. html/body特性

**1.width**

（1）html和body作为顶级元素，有block元素性质，默认会占据1整行。因此，html的width就是浏览器“可视区”的width；body的width就是html的width。

（2）显然，width作为顶级元素自身的性质，与内部的元素无关，不会被它们撑开（任何时候width都没有撑开一说）。当内部元素width超过“可视区”的width（就是html和body继承了的width）时，因html和body的overflow默认值为scroll（其他元素默认值为visible），因此会出现滚动条。

（3）由于上面的原因，当出现如1）给内部container设置1个大于“可视区”的width，或2）没有控制width的img等情况，会出现比较难看的局面。解决方法：1）如果给container设置固定的width，就必须要给body也设置同样值的min-width（也有在body下嵌套1个wrap，有同样的width，背景什么的在wrap上）；2）设置img的width或max-width为100%。

**2.height**

（1）由于html和body有block元素性质，因此默认height是0，但不同浏览器不同，有的为8px，有的有padding，但总的来说，不是100%“可视区”的height。

（2）要想设置内部元素的height为“可视区”height的100%，就必须设置html和body的height为100%。

（3）反过来，设置html/body的height为100%的情况只有1种，就是需要设置内部某个元素的height为屏幕height的100%。且还需要设置这个内部元素的overflow为scroll（另外，还需要设置box-sizing=border-box，否则还会因为container的滚动条导致body出现滚动条）。

**3.背景色**

虽然html、body的height不是100%“可视区”的height，但给它们设置背景色会传递给浏览器，导致整个“可视区”都有背景色。

* + 1. 继承

**1.不可继承属性**

（1）显示相关：display/overflow等。

（2）定位相关：position/float等。

（3）盒模型相关：width/height/padding/margin/border等（除line-height外的所有尺寸属性）。

**2.可继承属性**

（1）字体相关：font-size/style/family/weight等。

（2）文本修饰：word-spacing/letter-spacing/text-decoration等

（3）列表相关：list-style(type/position/image)等。

**3.特殊情况**

（1）color可继承，background不能继承。

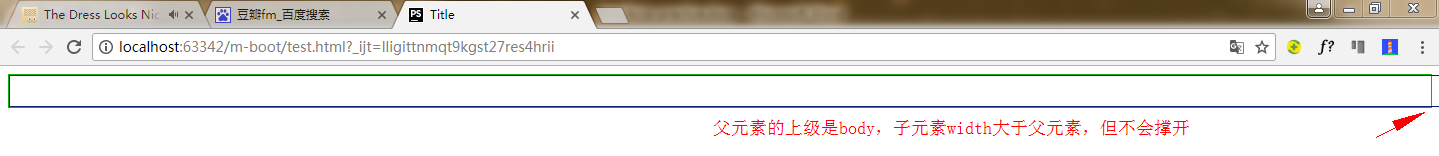
（2）text-align可继承，vertical-align不可继承。

（3）唯一与尺寸相关的可继承属性：line-height。

* + 1. 父子元素关系

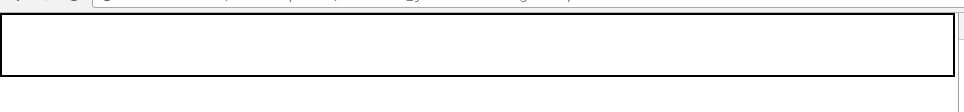
**1.width**

父元素的width，由父元素的父元素决定，没有撑开一说。

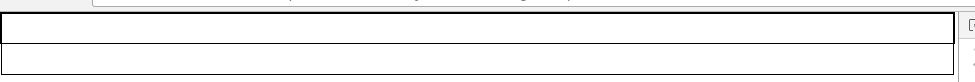


**2.height**

（1）如果父元素没有设置高度，子元素设置，撑开父元素高度。



（2）父元素设置有高度，子元素无法撑开，但默认会显示（overflow的默认值为visibility）出来。



* + 1. 折叠边距（margin折叠）

**1.产生原因**

文本排版需要。

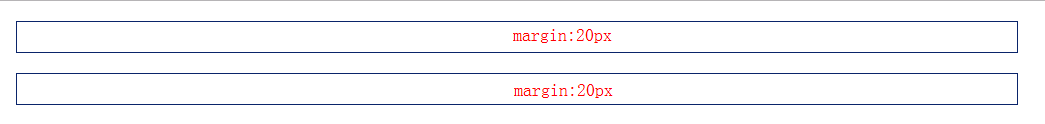
**2.产生情形**

（1）只存在于相邻和父子元素之间，且只发生在块元素上。

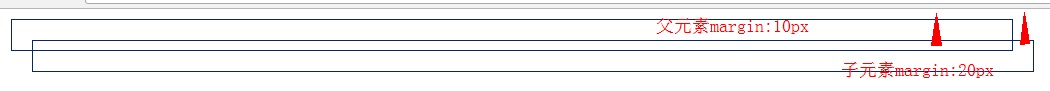
（2）在水平书写模式下，只有垂直方向发生margin折叠，即margin-top/margin-bottom，在垂直书写模式下，margin折叠发生在水平方向上，即margin-right/margin-left。

【注意】块元素才会发生折叠，因此无论如何，水平书写模式下，水平方向不会发生折叠（内联元素不会发生折叠）。

（3）相邻元素间，即使元素有padding和border，始终都会折叠。如下，两个相邻块元素，始终都会margin折叠。



（4）如果父元素有padding和border，则不会发生折叠。如下，因为父元素有border，子元素的margin不会折叠到父元素外。



（5）父子元素之间，如果子元素的margin-bottom没有超过父元素的height，不会发生折叠。如下，因为子元素高度不够，所以margin-bottom没有折叠，但margin-top折叠了，但margin-left也没有折叠（即，水平方向永远都不会发生折叠）。



（6）如果都是负数/正数，取绝对值最大的那个margin作为最终值（负数仍然是负数）。如果1正1负，相加后的值为最终值。

**3.不发生折叠情形**

（1）absolute元素、float元素、清除float元素、inline-block元素（因为它不是块元素）、定义了属性overflow且值不为visible的块元素不发生折叠。





（2）根元素的margin不与其它任何margin发生折叠。

* + 1. text-align

对block/inline-block元素都有效。只是如果inline-block元素不设置宽度，包裹在内容上看不出来，但如果设置的宽度大于内容，就会有效果。

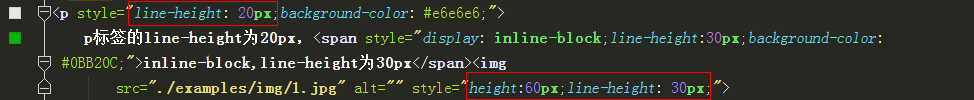
* + 1. line-height和vertical-align

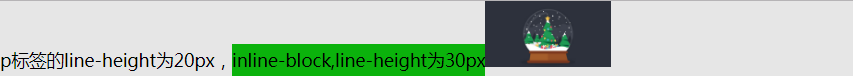
**1.线box的高度**

（1）line-height与线box的高度无关

如前所述，内联元素和内联块元素组成的线box。线box始终有足够高度包含其中的内联元素和内联块元素。对包含块设置line-height并不能决定线box的高度，线box的高度始终由自己计算而得。设置line-height唯一可能的效果就是被内部的子元素继承。

如下，img的height为60px，尽管设置了p的line-height仅为30px，但p的高度仍为69px。





（2）垂直对齐方式影响线box的高度

如上，尽管p中的元素最高为60px，但p的高度却为69px，原因在于垂直对齐方式。vertical-align的值可以为top、middle等关键字，默认值为baseline。关键字代表的含义如下：



真是因为内部元素对齐的是基线baseline而不是底线，导致p的高度增加了一些。如果要让p（线box）的高度等于其中元素最大的那个的height，可以设置vertical-align=bottom/middle/top等。

【注意】如果包含块设置了line-height>0，即使设置其font-size=0也不能达到效果。

**2.vertical-align**

（1）适用场景

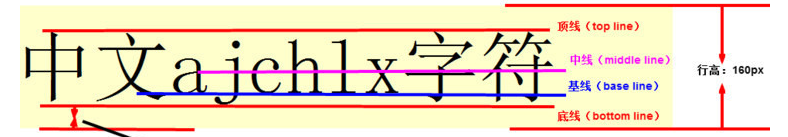
vertical-align只在两种情况下使用：

* 在父元素设置了line-height情况下，可对其内部的inline/inline-block元素可以设置vertical-align，用于设定该元素在line中的对齐方式（不是设置在父元素上，而是直接设置在需要对其的元素上）。
* 在表格的td或table-cell设置，用于控制其内部文本的对齐方式。

最好不要使用百分比值，因为ie对小数点有bug。

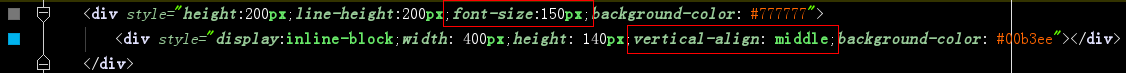
（2）vertical-align=middle如何对齐？

vertical-align=midldle并非严格意义上的居中对齐，而是从基线（baseline）开始，向上半个ex（即字母e、x，基线是ex的下沿，半个ex就是0.5em，半个字体大小）。



如下，如果父元素font-size为150px，子元素无法垂直居中对齐。





处理办法：

（1）设置父元素font-size为0。

（2）将内部元素设为relative，设置top值手动调整。

【说明】这个和上面的线box的高度无关：因为居中对齐前提是包含块的line-height足够高，高到不受内部元素影响，因此和上面的问题无关。

**3.其他说明**

（1）vertical-align参照物是自己的box相对于line box，如下：



（2）line box的高度虽然可以由内部line-height最大的决定，但内部元素之间却不会影响到各自的line-height，可以单独设置，设置多少就是多少。

（3）inline元素是否可以设置line-height？

可以。如果为最大值，也可以决定line box的高度。

（4）line-height和border-box之间的关系？

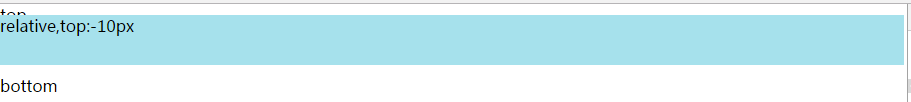
如果没有设置height，那么跟border-box关系不大，因为line-height不会限制到height。如果设置了height，那就从上往下显示。

1. 布局
   1. 基础
      1. 相对定位

（1）元素没有脱离文档流。

（2）block和inline元素均可。

（3）相对定位的偏移位置参照其正常应在文档流中出现的位置。尽管偏移，但仍占据其在原文档流中的位置，其在原文档流中的位置不会被其他元素挤占。



（4）相对于未定位元素，relative元素会显示在其他元素前面（z-index大半级）。

* + 1. 绝对定位

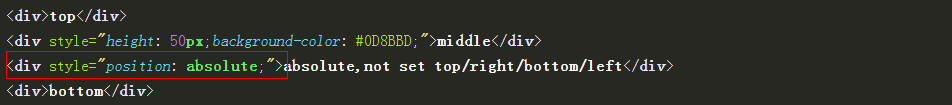
**1.定位属性**

（1）元素脱离文档流，后面的文档流会紧跟着补上来接替它的位置。

（2）要激活元素的定位属性，必须设置left、right、top、bottom属性，否则，它仍漂浮在原来的位置上。

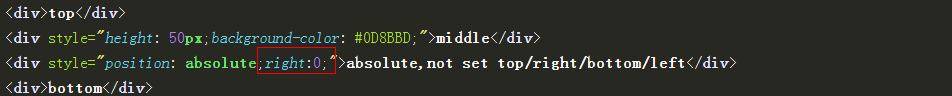
如下，absolute元素仍然“浮在”文档流原来的位置（后面的元素已经补上），且其z-index大半级，显示在bottom上面。





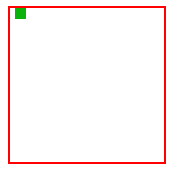
另外，如果absolute元素，只设置（激活）某个定位属性，不会影响其他属性。如下，设置了right，但没有设置top，那么垂直方向仍在文档流原来位置，只是水平方向变化。





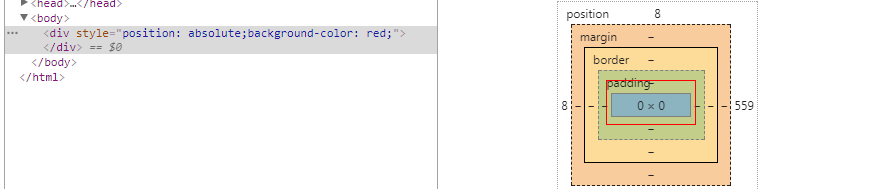
（3）偏移相对的是包含块（最近的relative、absolute元素）。

（4）偏移从哪里开始？参考的是包含块的边缘，这个边缘不包括border，但包括padding。



**2.尺寸**

（1）鉴于脱离文档流，自然没有高度、宽度这个说法，默认都是0。



（2）高度、宽度坍塌：absolute元素的高度/宽度不会撑开父元素。

* + 1. 浮动

**1.基本属性**

（1）如果两个float都为right，则第一个元素更靠右，从右到左排列。

（2）浮动元素只能靠左或靠右，无法居中。

（3）浮动元素脱离文档流，会朝着浮动方向一直移动到包含块的边缘（即使有同行元素也不受影响）。







**2.与同行元素关系**

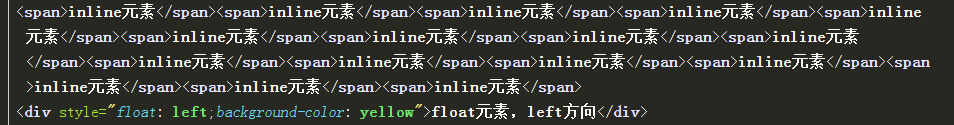
浮动元素脱离文档流，会以如下方式影响同行元素：

（1）如果前面是block元素，block元素不受任何影响，因为block元素独占1行，而float元素只能另起1行。



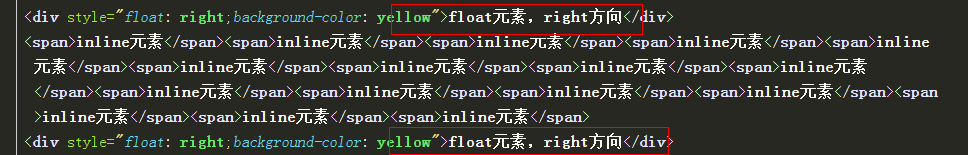
（2）如果前面是内联元素或内联块元素，float元素与之同行，并根据浮动方向影响它们：

* 如果为left，本行的这些元素将会排挤到后面。





* 如果为right，本行的将继续排挤到左边。

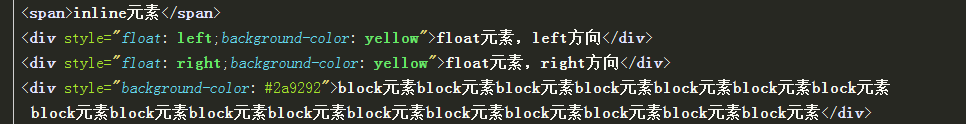


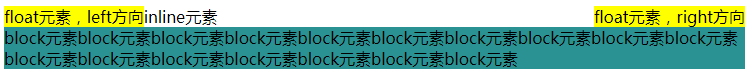


（3）如果后面是1个block元素，后面的元素不会再换行，且浮动元素层级会提升“半级”，即：同行元素文字内容会被挤压，但同行元素本身背景等不会被挤压。



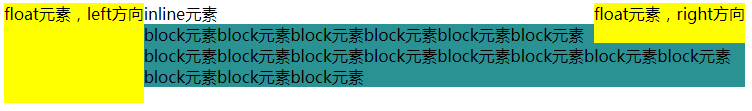
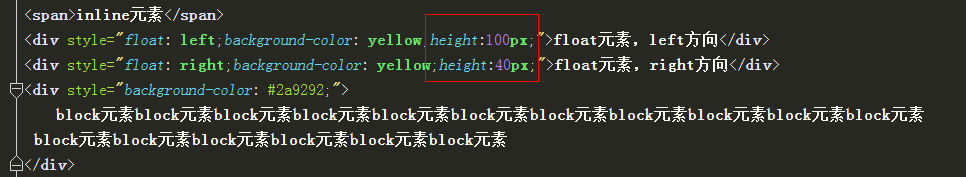
（4）如果前面有inline元素或inline-block元素，后面有block。通常，float元素将只影响前面的inline元素或inline-block元素，而不会影响后面的block元素。



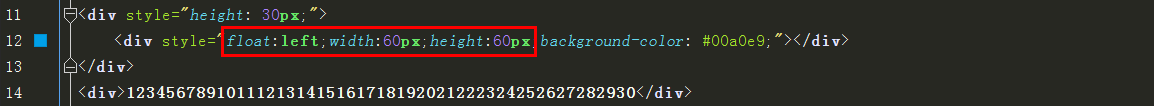


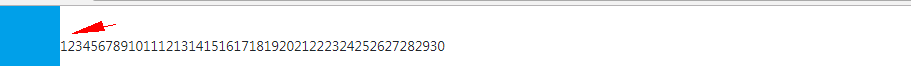
原因是这里的浮动元素只有1行，而block元素独占1行，因为前面有inline-\*元素，它就另起1行，而浮动元素就和这个inline-\*元素1行。

（5）如果浮动元素高度大于1行，就会挤压后面的块元素中的文字。



（6）如果浮动元素包裹在父元素中，还会挤压与其父元素同级的位于其父元素后面的元素。

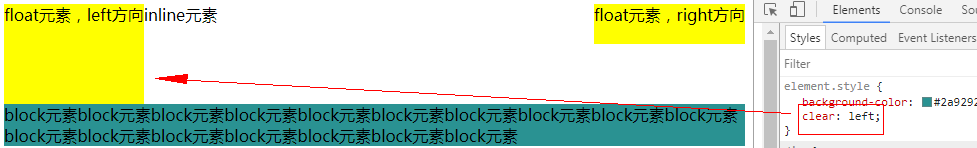




**3.清除浮动**

（1）如上，float元素后面的元素通过设置了clear可以另起1行。





（2）清除浮动的原因，挤压文字、父元素高度坍塌。

（3）浮动坍塌处理办法（父元素不用自身清除浮动）：

* 在父级元素内再加1个空元素，并让其清除浮动。因ie6有最小高度，还得设置font-size为0。
* 父级元素设置overflow，针对ie6还得设置zoom:1。
* 加after/before伪元素。
  + 1. 负margin

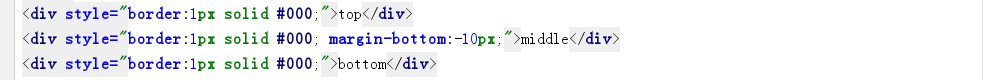
**1.基本性质**

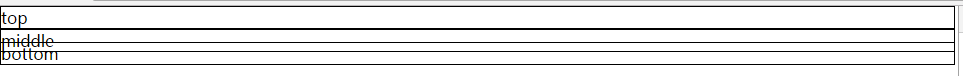
（1）负margin不脱离文档流（没有使用其他脱离文档流样式的话）。

（2）对于static元素，margin-top/left设为负值时，元素自身将被拉进指定方向。



（3）如果是margin-bottom设置为负值，元素自身位置不变，但后面的元素会覆盖自身，即，最终边界是由margin决定的。



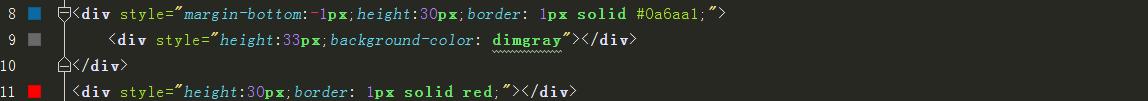


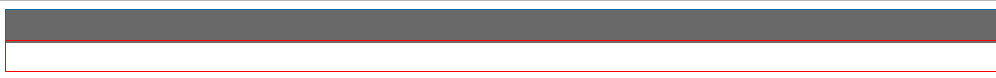
注意，是后面的元素覆盖自身：



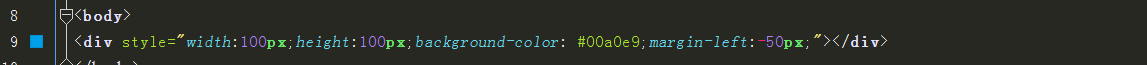


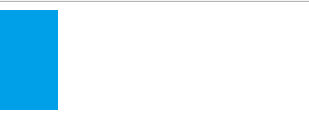
根据7层z-index原理，负margin内部的元素也会被后面的元素遮挡。





（4）负margin并不会撑开父元素宽度（因为父元素宽度默认继承自其父元素），也不会出现crollbar。因此，body的子元素，设置父margin等，也不会撑开body宽度。



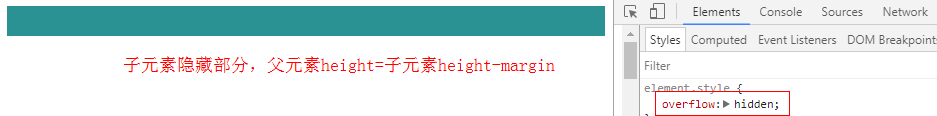


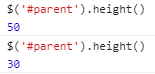
**2.父子元素之间的负margin**

（1）如果父元素没有设置height，子元素设置了height和负的margin。则，如果父元素没有设置overflow=hidden，父元素的高度=子元素height；否则，=子元素height-margin。









无论父元素是否设置overflow=hidden，后面的元素都会覆盖负margin部分。



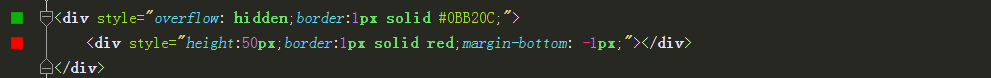
（2）如果父元素没有设置height，但设置有负的margin。没有特别样式，父元素的高度由子元素撑开，=子元素的height+margin。如同普通情况，后面的元素会重叠负margin部分。



【总结】

不能通过给没有height的父元素设置-margin来遮挡子元素，为什么？因为父元素height是子元素决定的，就像没法通过给父元素设置-margin遮挡自己的height、border。

那么怎样才能遮挡子元素？如上（1），只能在子元素自身上设置-margin，同时还要设置父元素overflow=hidden（子元素默认在父元素上啊）。





**3.width和负margin**

（1）如果block元素自身没有设置width，但设置有负值的margin-left/right，则元素自身会向负值margin方向增加宽度，如同padding一样。





如果父元素为inline-block（不可能为inline，包括-margin的这个元素也是，因为inline元素就没有width这个属性设置），后面的元素显然仅跟着父元素后面（跟-margin没有任何关系，就是普通正常文档流性质，因为父元素设置有width），从而发生重叠。



inline-block元素设置负值margin没有这样的效果，和普通负值margin一样。



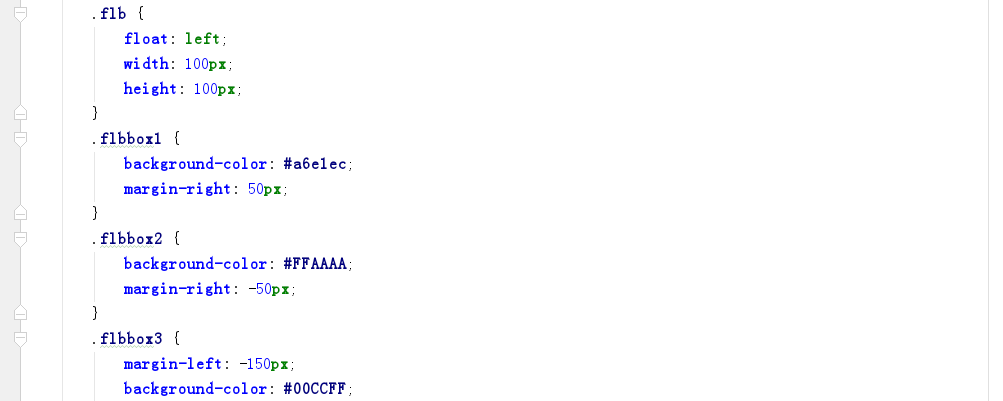
（2）如果给1个浮动元素加上相反方向的负margin，无论是否设置width，都不会使行宽增加，而是向通常情况那样，将后面的元素往前拉，与浮动元素重叠（如果负margin大于浮动元素宽度，将导致其实际宽度为0）。





（3）如果同一行都是浮动元素，就跟普通元素一样，可以设置负值margin来调整它们的位置。





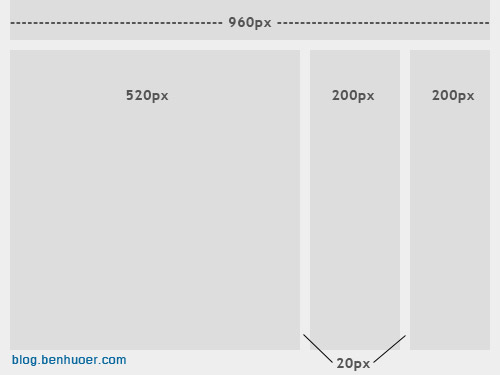


如上，box3距离最左侧的距离为100+50+100-50-150=50。

（4）所有浏览器都兼容，只是在一些非常老旧的浏览器有：链接无法点击、文字难以选中、图片会被截断问题，可以加1个relative即可完美解决。

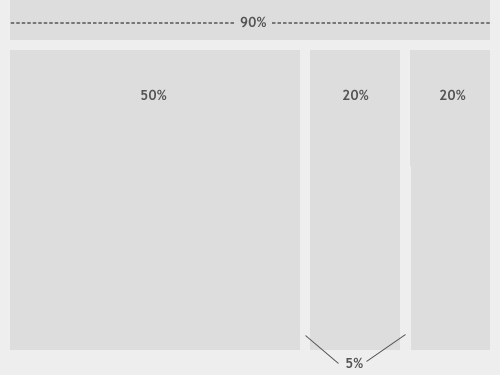
* 1. 布局方案
     1. 固定布局和流动布局比较

固定网页布局指网站内容被一个固定宽度的容器包裹，容器内的区块都有固定的百分比或者像素宽度值。最重要的一点——容器是不能移动的。不管屏幕分辨率如何变化，访客看到的都是固定宽度的内容。



固定布局的优点设计更简便、潜在的冲突更少。缺点是可能影响用户体验。

流动网页布局，也称为流体网页布局，其实现方法则是大多数组件（包括主容器）都设成百分比宽度，并且根据用户的屏幕分辨率自适应。

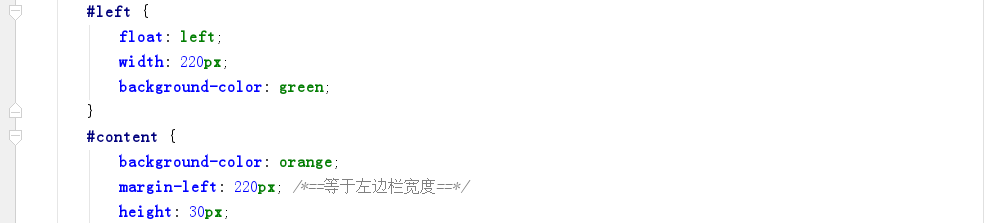


流动布局优点是根据客户端的情况自适应。缺点是可能存在潜在问题，屏幕过大时可能造成过量空白，破坏美感。

* + 1. 两栏布局：左定宽度右自适应宽度
       1. 方法1：浮动布局

根据原理,block元素应该在后面才起作用。



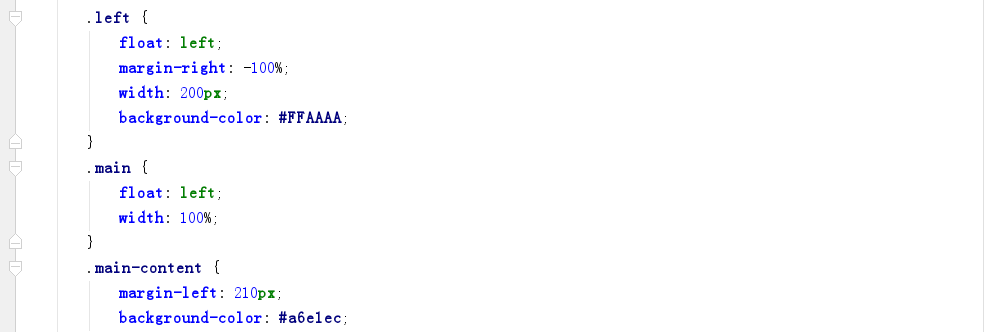


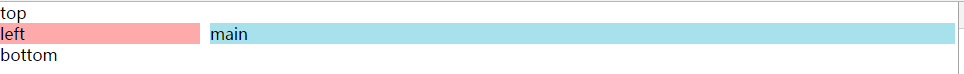


* + - 1. 方法2：浮动+负边距

要用负margin，外层必须都是浮动，内容层设置正margin。





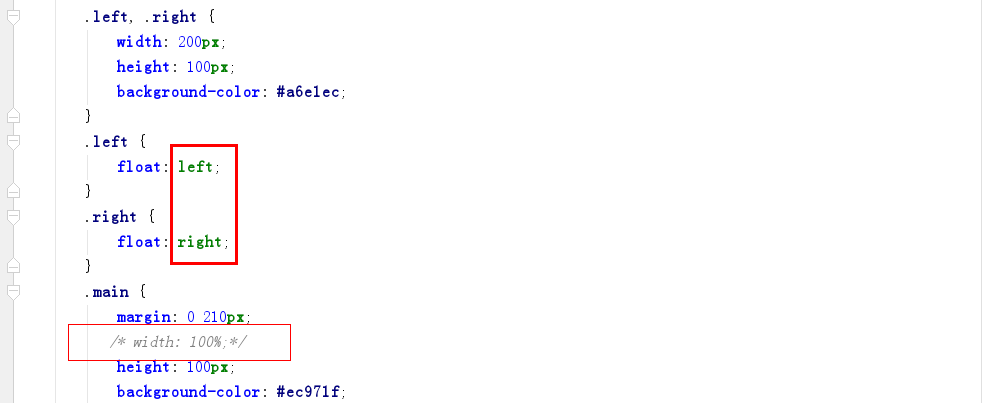


此处，left在前，main在后，与后面的双飞翼布局略有不同，left的负margin为-100%，即自己已经不占宽度了，因此main才不会另起一行（因为main的width为100%，如果前面有任何宽度都会另起一行）。

* + 1. 三栏布局：中间固定两边自适应宽度
       1. 方法1：浮动布局（圣杯布局）

和前面两栏布局方法1其实对应，block元素在最后（必须在最后）。







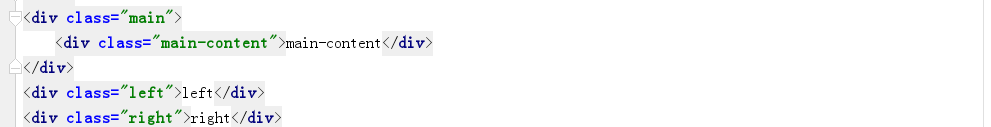
2个浮动的都在前面，1个左浮动，1个右浮动（都在.main前面）。

注意，此时.main的width没有设置。如果设置为100%，将出现下面的情况（原因是已经给main设置了margin）。

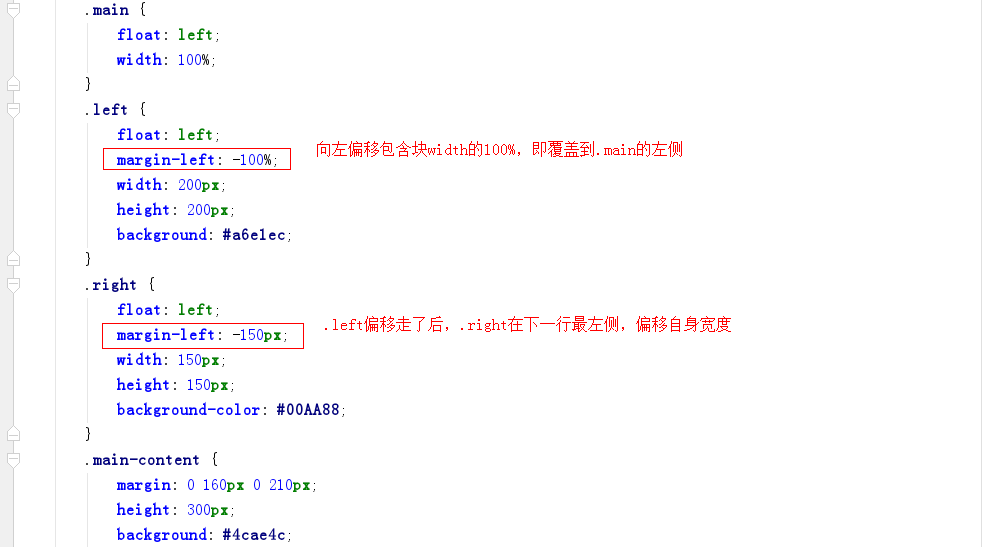


* + - 1. 方法2：负的margin（双飞翼）

和前面两栏布局方法2对应，外层都是浮动。



这样布局还有一个好处就是main先显示。

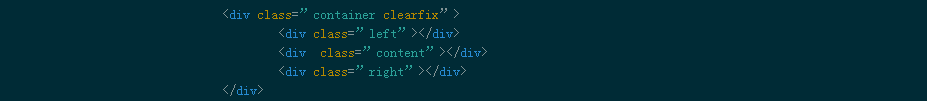




* + 1. 多列等高布局

**1.方法1：假登高列**

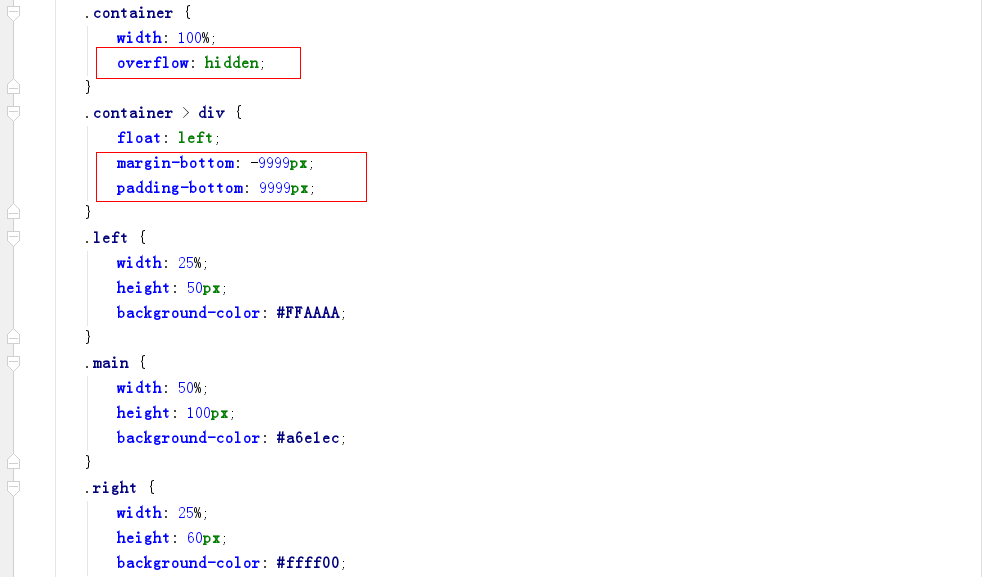
只适用于固定宽度布局，即用1个固定宽度的背景图放在整个容器后面，背景图进行Y轴的铺放，实现一种等高列的假像。

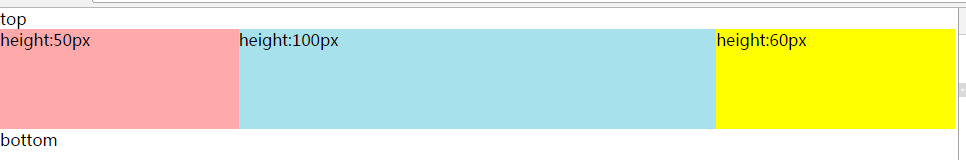




**2.方法2：使用正padding和负margin对冲实现多列布局方法**

见负margin多列登高布局，缺点是无法实现边框。





原理：

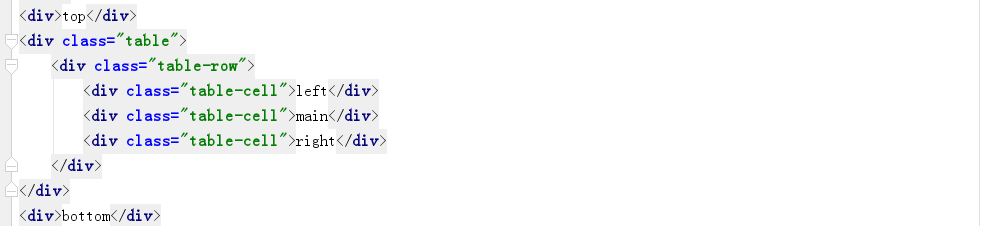
（1）父元素的高度=最大子元素高度。

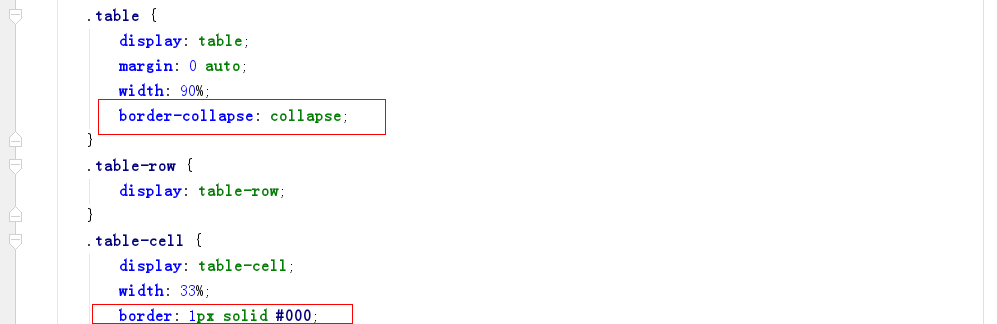
因为每个div的padding+margin=0，并不影响它们的布局高度，从而父元素的高度仍然和正常布局一样。

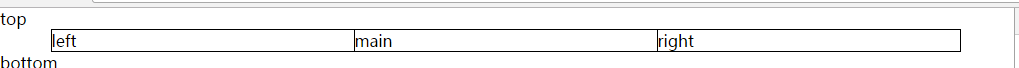
（2）子元素背景色包括padding。

**3.方法3：模仿表格布局实列等高列效果**

只适用于现代浏览器。







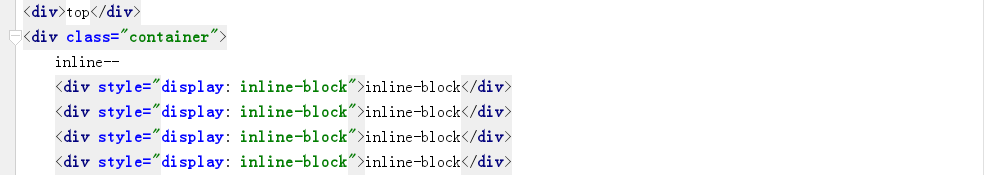
**4.方法4：使用jQuery和javascript**

* + 1. 分散对齐

**1.使用text-align:justify实现分散对齐布局**

（1）text-align:justify适用于inline和inline-block。

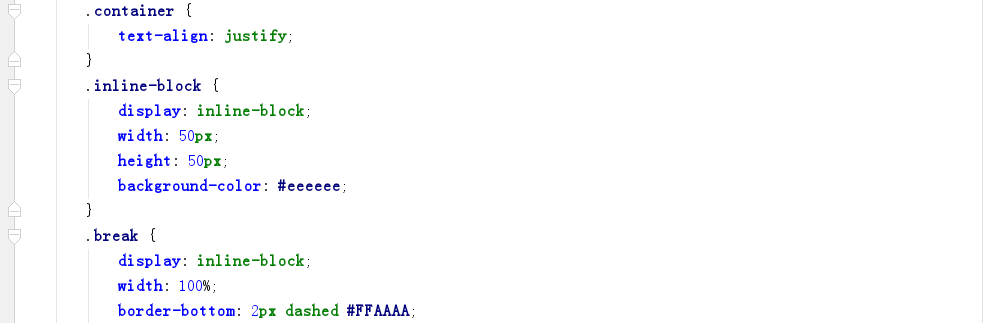
（2）单行情况下，text-align:justify并不能工作。

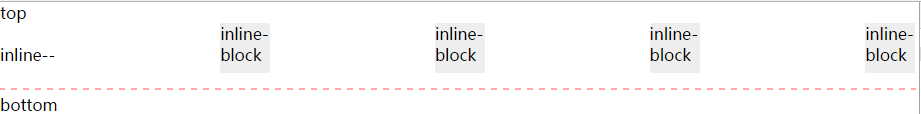




（3）可以加入break元素实现多行，从而实现换行。





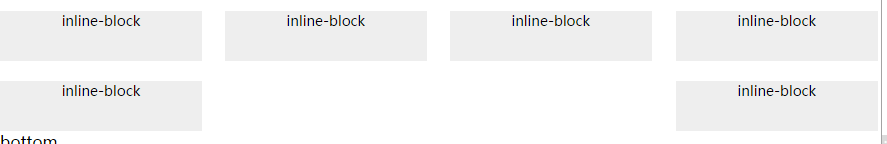


（4）还可以通过伪元素实现多行。



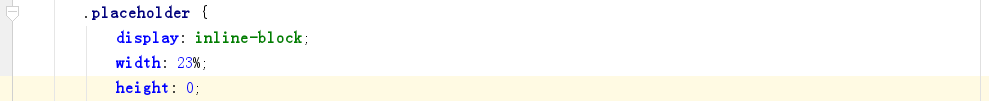


（5）如果出现换行，但下一行不足数量，将会出现类似如下情况：



（6）解决此类办法是添加2个占位元素（任何情况下都只加2个就可以了）。





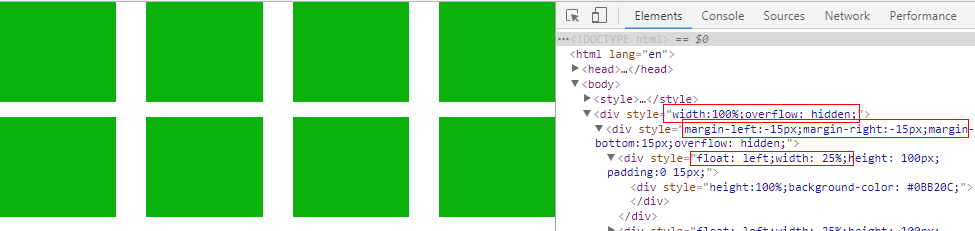


**2.float且padding+负值margin（bootstrap就是这个方法）**

（1）通过相同width的float实现等分，float内有左右padding。

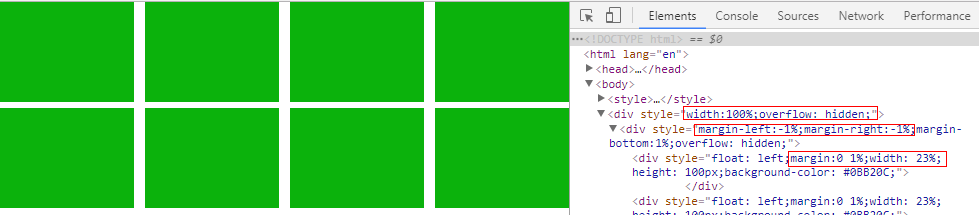
（2）float外面再嵌套有左右负值margin的层，从而实现水平等分。因这1层没有设置明确的width，负值margin会增加width，因此，外面还得再嵌套1层，明确设置width为100%，且overflow=hidden（和里面的overflow左右不同，里面的主要是清除浮动，可以用其他清除浮动方式代替）。

（3）垂直方向，不能直接通过float那1层实现，而是在嵌套层实现。



**3.float层且margin+负值margin**

和上面的基本相同，只是float层不设padding，而是设置margin，从而内部不用再嵌套1层（但没法实现间距为固定值，只能是百分比）。



* 1. 专题
     1. z-index

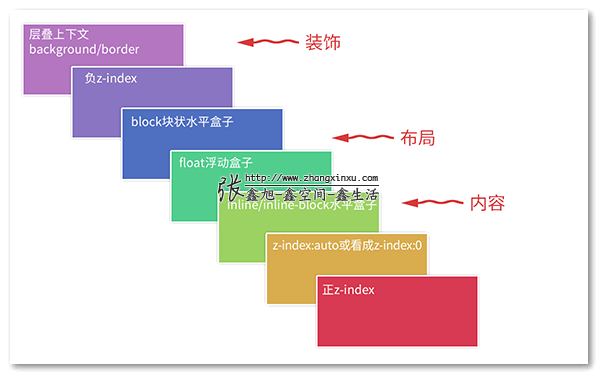
**1.基本概念**

（1）仅能用于定位元素（relative/absolute/fixed）。

（2）定位优先：只要设置了非static的position值，该定位元素自动被赋予一个默认的z-index值，居于普通元素之上。

（3）后来优先：同样的定位元素，位于后面的默认z-index值更大，显示在前。

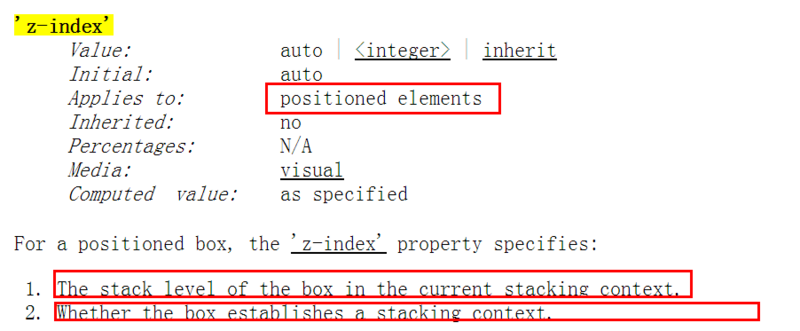
**2.七阶层叠水平**



（1）谁大谁上：层叠水平值大的那一个覆盖小的那一个。

（2）后来居上：当元素的层叠水平一致、层叠顺序相同的时候，在DOM流中处于后面的元素会覆盖前面的元素。

**3.如何让子元素层叠在父元素下面**



（1）由上面可知，z-index的作用有：

* 设置在当前堆叠上下文中的层级。
* 创建一个新的堆叠上下文。

因此，z-index并不是设置的值越高，就会越靠近用户，还和堆叠上下文有关系。

（2）在同一个堆叠上下文中的元素，z-index越高越靠近用户。

（3）在不同堆叠上下文中的元素，如果堆叠上下文一距离用户更近，那么它的所有子元素都在另一个堆叠上下文子元素的前面，也就是离用户更近，不同堆叠上下文中的子元素不可能发生交叉。因此，z-index其实不是一个绝对值，而是一个相对值。



综上，如果是上面这样，子元素将会隐藏。因为子元素和父元素在同一个堆叠上下文。如果给父元素添加1个z-index，将无法实现隐藏，因为父元素创建了1个新的堆叠上下文，子元素就在这个新创建的上下文中，它始终高于父元素所在的那个上下文。

* + 1. 单位

**1.%**

**（1）尺寸属性**

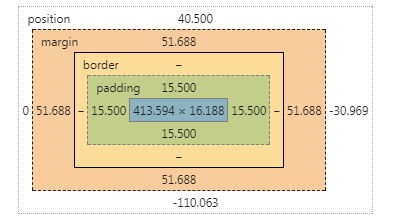
left/right和margin/padding/width：包含块的width；top/bottom和height：包含块的height。

1）对于绝对定位元素，无论父元素是否明确定义了width/height，都会准确计算出。

2）对于普通元素，父元素的width没有明确定义不会影响%计算；父元素的height没有明确定义，不影响top/bottom定位属性的计算，但子元素height属性值为0。



父元素由height为60px的子元素撑开，绝对定位元素的top、margin、padding、width、height都能够计算。注意：margin/padding都是是按照父元素的width计算，导致特别大。如下：



而对于普通block/inline-block元素，其width/left能够计算，top也能够正确计算，但height为0。



3）浮动元素同普通元素完全一样。



4）没有任何兼容性问题。

（2）其他属性

1）font-size：直接父元素的font-size。

2）line-height：元素自身的font-size。

3）border-radius/transform/background-size：元素自身的尺寸，如border-box情况下，width为60px，height为50px，则border-radius:50%为30px/20px。

**2.em**

（1）任何元素均是相对于自身字体大小（与%不一样，经测，任何属性的尺寸都是相对于font-size）。如果当前元素没有设置字体大小，则根据父级字体大小（其实还是自身，因为自身继承了父级字体大小）。

（2）浏览器body默认的font-size为16px，ie还需要另行设置html{font-size:100%;}。

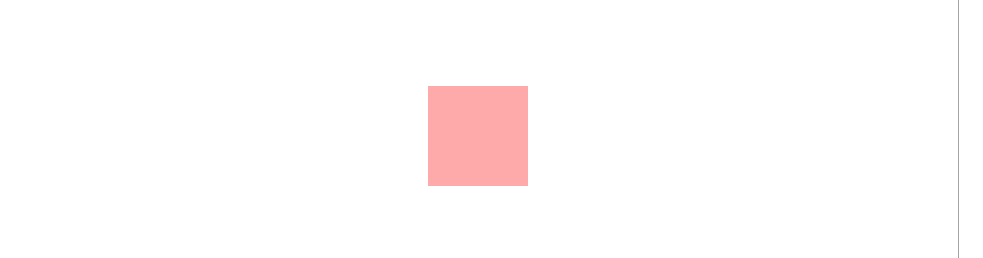
（3）既然是响应式，那么font-size也应该设置为em。因此，如果要设置元素的font-size，要根据父元素的font-size计算。但元素自身的其他属性，仍然要用元素自己计算后的em在计算（根据计算后的px再计算）。使用时注意别层层加码。

（4）没有任何兼容性问题。

* + 1. 负margin布局相关应用

**1.负边距+定位：水平垂直居中**



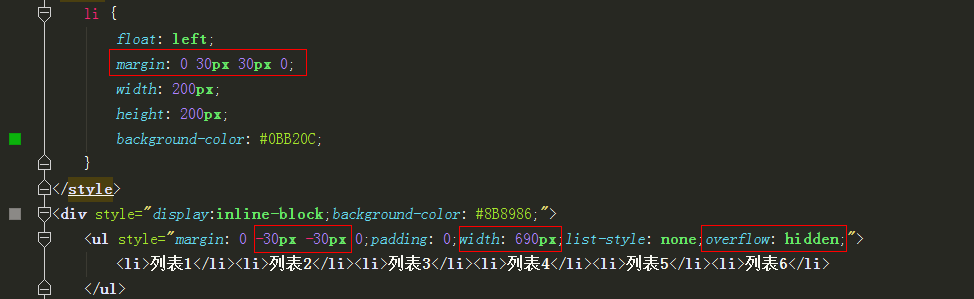


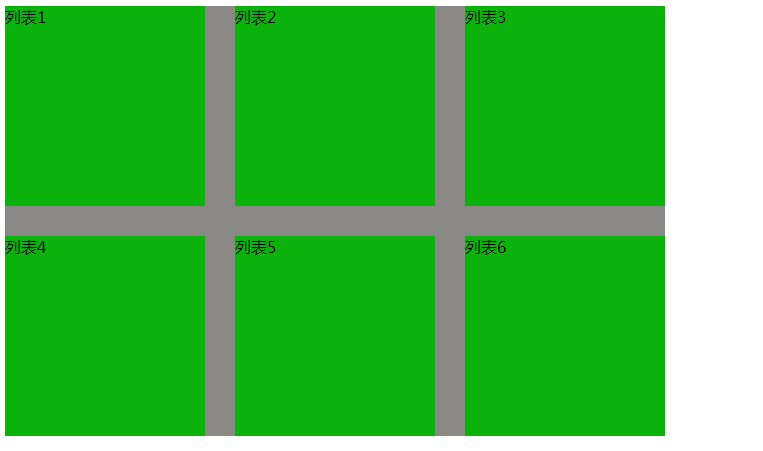
**2.左右固定，中间自适应（双飞翼）**

见前面三栏布局方法2。

**3.去除列表靠边元素的margin**

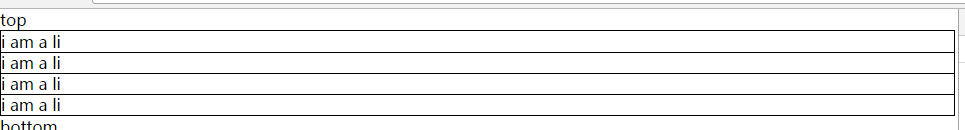
如下：列表li浮动，有右、下margin。父元素ul必须设置width，要不怎么float（但不能设置height，要不不能自动排布）？ul设置的右、下-margin并不会改变其自身尺寸（见前面基础），但会影响后面的元素和包裹元素，因此再给它加上1个父元素，从而完美实现。





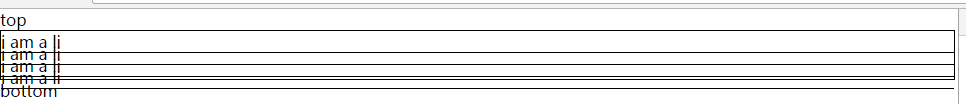
**4.去除列表最后一个li元素的border-bottom**



注意，父元素不设置高度，则其高度由子元素决定，最下面的border是父元素的，父元素遮挡了子元素的border。

由于每个li的高度因为负margin减少了1px，如果减少10px，效果为：



**5.多列等高布局**

见多列登高布局。

1. 移动端重构
   1. viewport

**1.移动端的像素**

（1）物理像素

可以看作成显示器上一个个的点。

（2）设备独立像素

可以认为是计算机坐标系统中的一个点。这个点代表一个可以由程序使用的虚拟像素**(比如说CSS像素)**，然后由相关系统转换为物理像素。

（3）CSS像素

主要使用在浏览器上，用来精确度量Web页面上的内容。通常**CSS像素就是设备独立像素**（device-independent pixel），简称DIPs。

**2.高清显示屏原理**

（1）屏幕密度为设备表面上存在的像素数量，通常以每英寸有多少像素来计算（PPI）。

（2）高清屏就是通过提高单位面积内的像素密度实现。但设备独立像素并没有改变。如从iphone4开始，苹果公司便推出Retina屏，分辨率提高了1倍，变成640×960，但屏幕尺寸却没变化。这就意味着同样大小的屏幕上，像素却多了1倍。

（3）设备像素比

简称dpr，为物理像素和设备独立像素的对应关系。



在CSS中，对不同dpr的设备，可以通过-webkit-device-pixel-ratio、-webkit-min-device-pixel-ratio、-webkit-max-device-pixel-ratio进行媒体查询。在js中，可以使用window.devicePixelRatio获取到当前设备的dpr。

例如，iPhone6的设备宽度和高度为375pt×667pt，可以理解为设备的独立像素；而其dpr为2，可知其物理像素为750pt×1334pt。

（4）其他因素

还有1个因素也会引起css中px的变化，那就是用户缩放。例如，当用户把页面放大1倍，那么css中1px所代表的物理像素也会增加1倍；反之把页面缩小1倍，css中1px所代表的物理像素也会减少1倍。

（5）高清屏导致的问题

同样像素大小的图片，在高清屏下会模糊。因为同样大小，高清屏需要更多的像素。另外，如果是位图的话，如果没有显示设置尺寸，将会按devicePixelRatio放大。

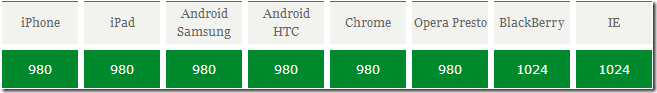
**2.什么是viewport**

ppk认为，移动设备上有3个viewport。

（1）layout viewport

首先，移动设备上的浏览器认为自己必须能让所有的网站都正常显示，即使是那些不是为移动设备设计的网站。但如果以浏览器的可视区域作为viewport的话，因为移动设备的屏幕都不是很宽，所以那些为桌面浏览器设计的网站放到移动设备上显示时，必然会因为移动设备的viewport太窄，而挤作一团，甚至布局什么的都会乱掉（css中的1px并不是代表屏幕上的1px，不然就会因为太小而看不清）。为此，浏览器通常默认情况下把viewport设为一个较宽的值，比如980px，这样的话即使是那些为桌面设计的网站也能在移动浏览器上正常显示了。ppk把这个浏览器默认的viewport叫做**layout viewport**。这个layout viewport的宽度可以通过 document.documentElement.clientWidth（即，默认通过滚动条来实现）来获取。

下图列出了一些设备上浏览器的默认viewport的宽度。



（2）visual viewport

再次，因此浏览器默认的layout viewport大于浏览器可视区域的宽度。所以我们还需要一个viewport来代表 浏览器可视区域的大小，ppk把这个viewport叫做**visual viewport**。visual viewport的宽度可以通过window.innerWidth来获取，但某些浏览器无法正确获取。

（3）ideal viewport

再次，因为越来越多的网站都会为移动设备进行单独的设计，所以必须还要有一个能完美适配移动设备的viewport。所谓完美适配就是指：首先不需要用户缩放和横向滚动条就能正常的查看网站的所有内容；第二，显示的文字的大小是合适，比如一段14px大小的文字，不会因为在一个高密度像素的屏幕里显示得太小而无法看清。即，无论是何种密度屏幕、何种分辨率下，显示出来的大小都是差不多的（不只是文字，其他元素如图片等也同样如此）。ppk把这个viewport叫做ideal viewport。ideal viewport不同的设备是不同的。所有iphone的ideal viewport宽度都是320px，即无论屏幕宽度是320px还是640px。即在iphone中，css中的320px就代表iphone屏幕的宽度。但是安卓设备就比较复杂了，有320px的，有360px的，有384px的等，可以通过http://viewportsizes.com查看。

**3.利用meta标签对viewport进行控制**

移动设备默认的viewport是layout viewport。但在进行移动设备网站的开发时，通常需要的是ideal viewport。这个可以通过meta标签设置。常见的viewpoint设置如下：



该meta标签的作用是让当前viewport的宽度等于设备的宽度，同时不允许用户手动缩放。也不允许用户缩放。meta viewport有6个属性：

|  |  |
| --- | --- |
| width | 设置layout viewport的宽度，为1个正整数，或字符串width-device。 |
| initial-scale | 设置页面的初始缩放值，为1个数字，可以带小数。 |
| minimum-scale | 允许用户的最小缩放值，为1个数字，可以带小数。 |
| maximum-scale | 允许用户的最大缩放值，为1个数字，可以带小数。 |
| height | 设置layout viewport的高度，很少使用。 |
| user-scalable | 是否允许用户进行缩放，值为"no"或"yes"，no代表不允许，yes代表允许。 |

又如，小米官网设置如下，和内部body的宽度1样，都是固定值。



**4.如何将viewport宽度设置为ideal viewport的宽度**

通常设置initial-scale=1就能将当前的的viewport变为ideal viewport。因为它就是缩放是相对于ideal viewport来进行缩放的。即，不对当前的页面进行缩放，也就是页面本该是多大就是多大。另外，当width和initial-scale=1冲突时，不是按书写顺序解决冲突，而是取两个中较大的那个值。因此，使用"width=device-width,initial-scale=1"合并处理。

需要说明的是，单独使用width=device-width在iphone和ipad中，无论竖屏还是横屏，宽度都是竖屏时ideal viewport的宽度。因此还需要设置initial-scale=1。

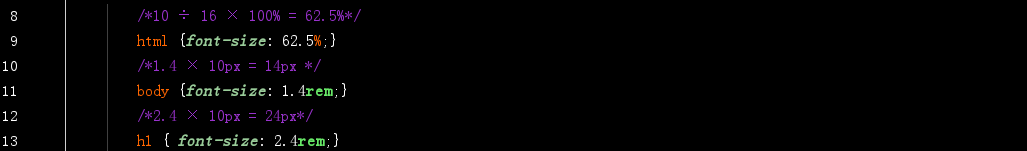
* 1. 相关单位

**1.%**

**2.em**

**3.rem**

* css3中的属性，兼容ie9。
* rem是相对于根元素html。浏览器默认字体大小为16px，重置为10px后，即可使用rem设置任何层级元素的字体大小。



* + 1. vw、vh、vmin(vm)

css3中的属性，兼容ie9。

vw：相对视区的宽度（1个视区宽度为100vw）；vh：相对视区的高度；vm：取视区高度和宽度较小的那个。视区为浏览器内部的可视区域大小，即window.innerWidth/window.innerHeight大小，不包含任务栏标题栏以及底部工具栏的浏览器区域大小。

* 1. 移动端特殊问题
     1. Retina屏1px边框

因为物理像素倍数于独立像素的原因（还是其他原因？），手机端设置1px边框，实际显示2px（还是就是1px，只是原生的用了0.5px？），如下：

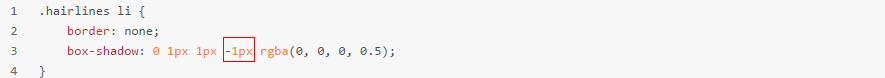
 

实现0.5px边框方法，一共可能有7种，比较合适的有：

（1）使用图片，但没法改变颜色。

（2）多背景渐变，但没法实现圆角。

（3）使用box-shadow模拟



（4）使用viewport+rem

在devicePixelRatio=2时，输出如下viewport：



在devicePixelRatio=3时，输出如下viewport：

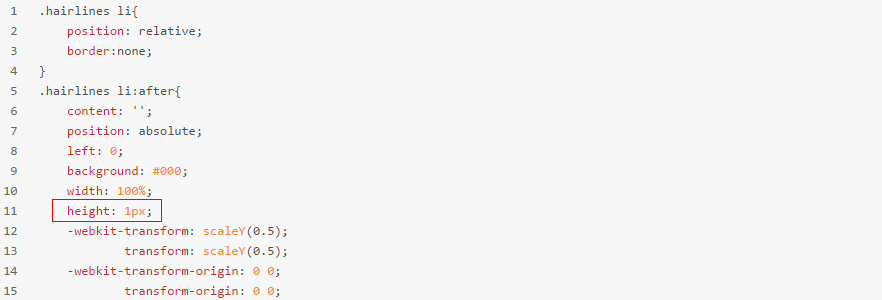


同时，设置对应viewport的rem基准值。

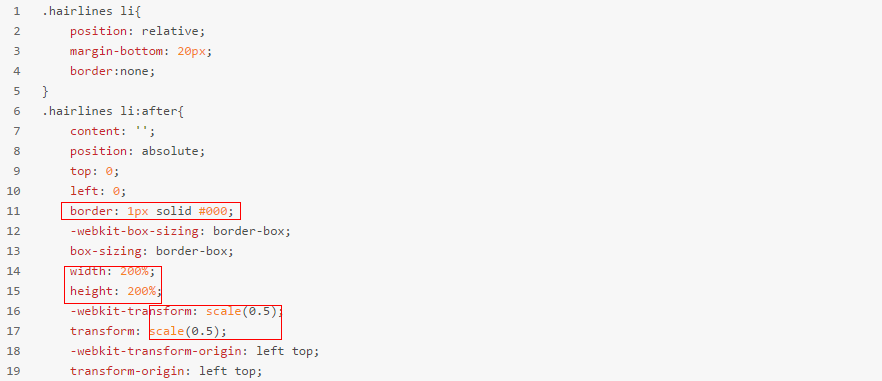
（5）伪类+transform

将原先元素的border去掉，然后利用:before或:after重做border，并使用transform的scale缩小一半。原先的元素相对定位，新做的border绝对定位。

单条border：



四条border：



然后结合js代码，判断是否Retina屏。



* + 1. fixed

2个方法：（1）css3的transform带来的fixed相对定位问题；（2）虚拟键盘弹出导致的各种bug。有的虚拟键盘会改变窗口大小。有些非默认的虚拟键盘则是以弹层的形式覆盖在上面的，并没有改变窗口大小，也就没有办法监听window的onresize事件。

http://efe.baidu.com/blog/mobile-fixed-layout/

如下图<header/>和<footer/>都是fixed区域，<main/>是可以滚动的区域。



如果软键盘唤起后，页面的fixed元素将失效（即无法浮动，也可以理解为变成了absolute定位）。所以，当页面超过一屏且滚动时，失效的fixed元素就会跟随滚动了。

处理方法：

思路：如果页面不会过长出现滚动，那么即便fixed元素失效，也无法跟随页面滚动，也就不会出现上面的问题了。即，将原body滚动的区域域移到main内部，而header和footer的样式不变。不过此时存在另外1个问题，就是main元素内的滚动非常不流畅，滑动的手指松开后，滚动立刻停止，失去了原本的流畅滚动特性。可以设置属性-webkit-overflow-scrolling=touch恢复弹性滚动。

其他细节处理：

（1）有时候focus输入框后，会出现软键盘遮挡输入框的情况。可以尝试input元素的scrollIntoView修复。

（2）在iOS下使用第三方输入法时，输入法在唤起经常会盖住输入框，只有在输入了一条文字后，输入框才会浮出。目前也不知道有什么好的办法能让唤起输入框时正确显示。这暂时算是iOS下的一个坑吧。

（3）有些第三方浏览器底部的工具栏是浮在页面之上的，因此底部fixed定位会被工具栏遮挡。解决办法也比较简单粗暴——适配不同的浏览器，调整fixed元素距离底部的距离。

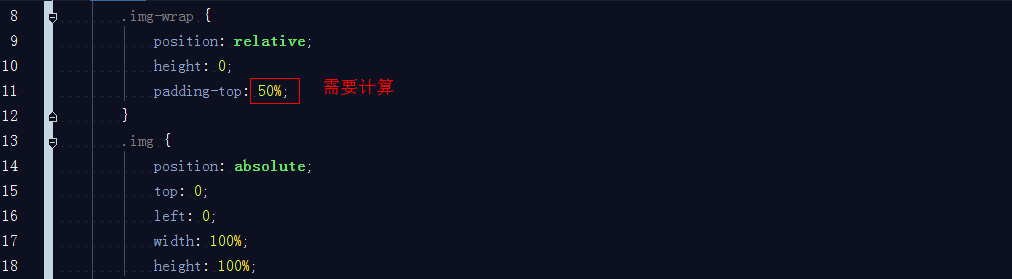
（4）最好将header和footer元素的touchmove事件禁止，以防止滚动在上面触发了部分浏览器全屏模式切换，而导致顶部地址栏和底部工具栏遮挡住header和footer元素。

（5）在页面滚动到上下边缘的时候，如果继续拖拽会将整个View一起拖拽走，导致页面的“露底”。 为了防止页面露底，可以在页面拖拽到边缘的时候，通过判断拖拽方向以及是否为边缘来阻止touchmove事件，防止页面继续拖拽。

* + 1. 图片高度占位

与pc端不同，移动端的图片很多都不是固定宽高（icon图标、头像等一些小图还是固定大小的）。由此导致1个问题：由于不能设置1个具体的高度，导致加载过程中后面的内容随着图片的加载慢慢向下移动。

解决方法：给图片提供一个容器，设置高度为0，根据宽度按照图片的比例计算出1个高度值，设置padding-top为这个值。内部的图片设置为绝对定位，宽高为100%即可。



* + 1. 居中

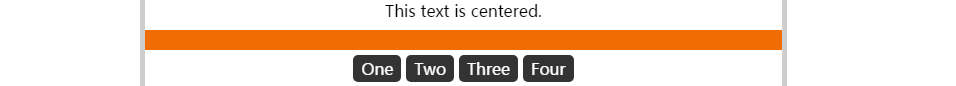
除了CSS2时代的常规方法，更多使用CSS3的transform及flex方法。而对于img或video的最新object-position还得等待兼容的时代。

**1.水平居中**

（1）inline-\*元素

在块级父容器中让行内元素居中，使用text-align=center即可。



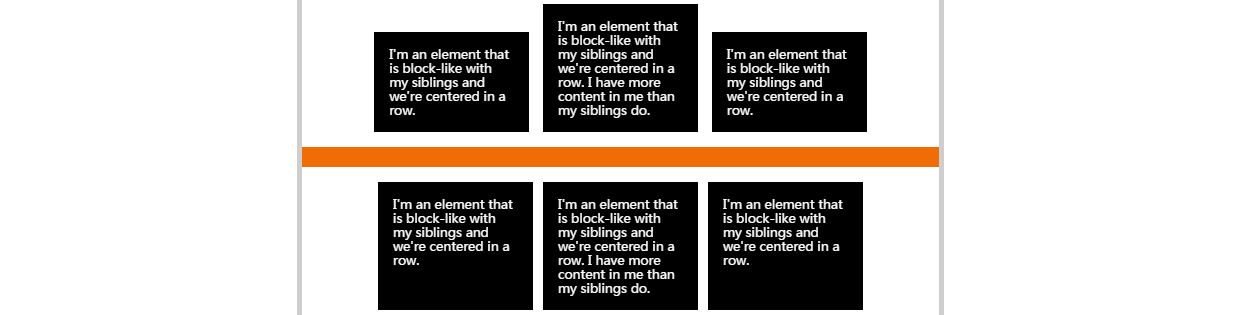


（2）单个块级元素

设置margin-left和margin-right为auto（前提是已经为元素设置了适当的width宽度，否则块级元素的宽度会被拉伸为父级容器的宽度），可以使单个块级元素居中。

（3）多个块级元素

设置display为inline-block或flexbox。



**2.垂直居中**

（1）inline-\*元素

如果想让单行内联元素垂直居中，可以设置相同的上下padding。



如果不用padding，也可以使用line-height，但要确保不换行。



如果是多行，也可以使用上下相同的padding。或使用table-cell，并设置vertical-align=middle。



或者使用flexbox。

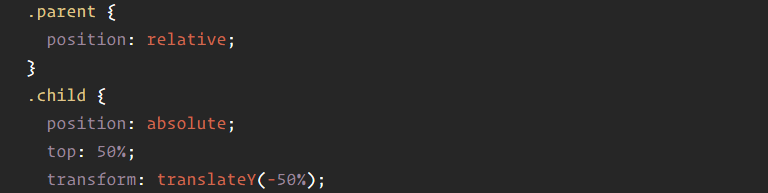


使用上面的方法前提是父元素有确定高度。

（2）块级元素

如果已知块级元素高度，使用绝对定位+top+margin实现。

如果未知元素高度，可以使用如下方式：

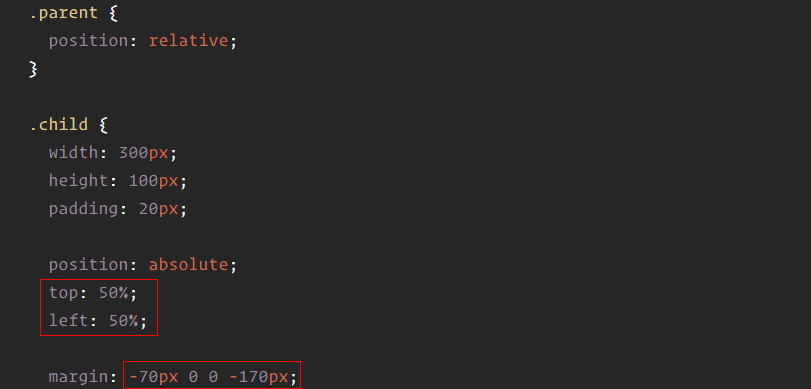


使用flexbox最为简单：

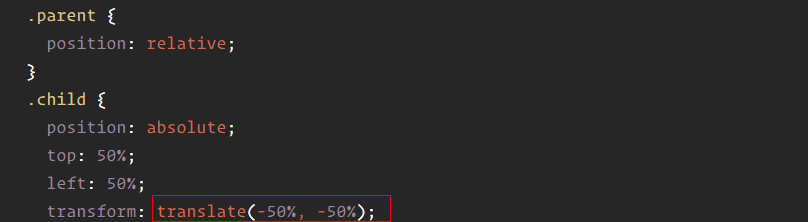


**3.水平且垂直居中**

（1）宽高固定元素



（2）宽高不固定元素



最简单的还是使用flexbox。



* + 1. 等分
    2. css3动画

**1.动画库**

https://github.com/daneden/animate.css/

https://ianlunn.github.io/Hover/

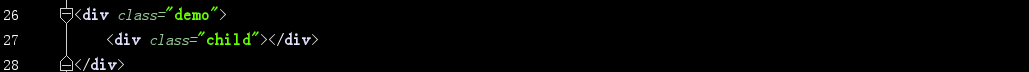
https://leaverou.github.io/animatable/#transform/10

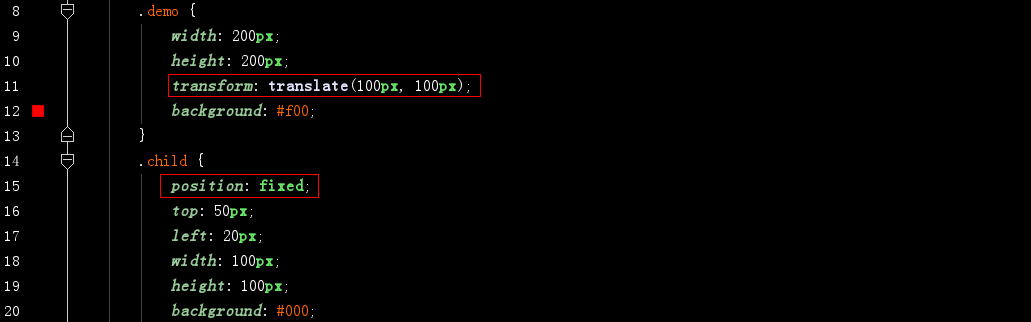
https://h5bp.github.io/Effeckt.css/

**2.重要属性**

（1）transform

任何非none值的transform会导致创建一个堆栈上下文和包含块。所以如果父级元素设置了transform属性，子元素position=relative/absolute/fixed会基于transform后的父元素定位。







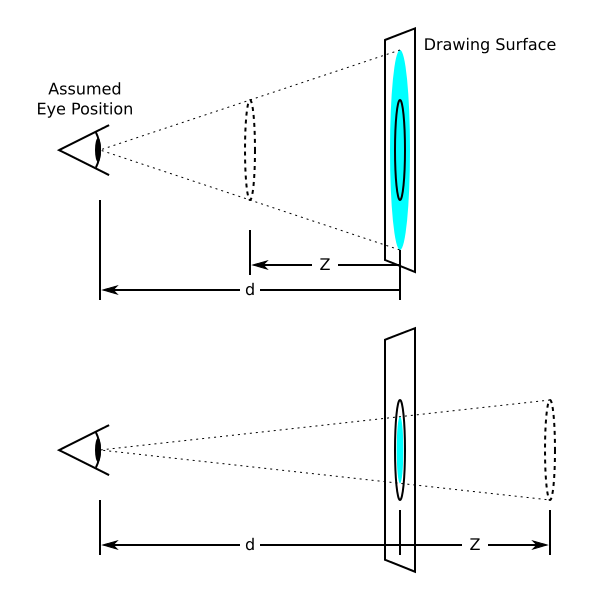
transform属性值覆盖问题：

transform可以有四个不同的变换，分别为scale、translate、skew、rotate。当有两个transform设置不同变换时，权重大的覆盖权重小的。但在动画中存在严重的问题，因为后面的值必须拷贝之前设置的值。

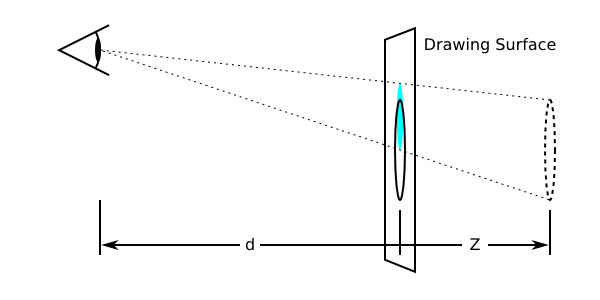
另外，transform四个值需要按顺序写，否则会混乱坐标轴和中心点。

（2）perspective

perspective属性指定了观察者与z=0平面的距离，使具有三维位置变换的元素产生透视效果。z大于0的时候三维元素比正常大，而z小于0时则比正常小，大小程度由该属性的值决定。该属性不可继承，只对一级子元素管用。



察者站在哪里也会影响效果，如下图改变perspective-origin：

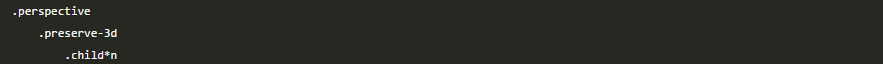


（3）transform-style

transform-style属性指定了，该元素的子元素是（看起来）位于三维空间内，还是在该元素所在的平面内被扁平化。如值为preserve-3d则创建一个3D渲染上下文，其直接子元素有一个共同的三维坐标系。同样该属性不可继承，只应用于直接子元素。

（4）perspective 与 preserve-3d 差别

perspective为其直接的具有三维变换的子元素产生一个透视效果；而preserve-3d则为其直接的子元素提供一个3d渲染空间。最佳实践吧。外面一层提供透视，然后再包裹一层提供3d渲染空间。



https://output.jsbin.com/quxuro

**3.动画**

（1）transition

* 不可自动触发，可以通过改变class，改变状态(:hover,:active,:checked等)触发
* display的none与其他值的切换不行，通过delay设置也不行，除非通过回调函数或setTimeout先切换display，再设置动画样式改变。
* 可设置delay为负值，表示动画已经运行到了该时间，前面的动画效果忽略
* 可对自己或子元素进行动画动画
* 或对同级下面的元素及其子元素进行动画控制
* 可在状态内添加transition，覆盖默认的transition

（2）animation

* 可自动触发，也可以通过状态或增加class触发
* 安卓低端机不支持伪元素（::before和::after）动画
* animation-fill-mode可设置动画结束及开始的状态。如为backwards，则元素默认应用第一关键帧的样式，忽略delay，可通过一开始就暂停观察（animation-play-state:paused;）；如为forwards，则在动画结束后，元素将应用动画结束后的属性值；如果animation-fill-mode的值为both，则动画会遵循backwards和forwards的规则。也就是说，它会设置开始和结束的状态。
* animation-timing-function默认应用在每个关键帧之间的变化，而不是开始到结束整个流程。所以@keyframes中的每个关键帧可以重新定义animation-timing-function
* 可以用于none到block的动画切换

（3）动画js事件

transition动画只有一个transitionend事件，而webkit现在既支持webkitTransitionEnd，也支持标准的transitionend事件，所以只能绑定一个，不然会触发两次事件

如有多个属性参与动画，就会出现多个transitionend事件（这个事件标准还是有不少bug的），所以请使用jquery的one事件，或者绑定事件调用函数中随即取消绑定事件

（4）不支持的动画属性切换

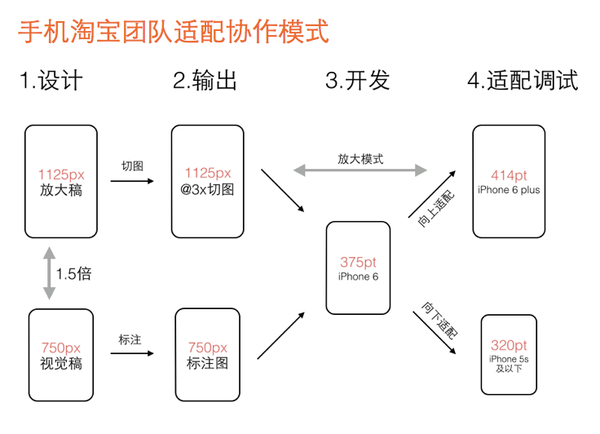
* background-image
* float
* height/width/top/right/bottom/left等auto值向具体值的变换
* display在none和其他值之间切换
* position在static和absolute之间切换
  1. 手淘移动端适配

**1.基本思路**

（1）选择一种尺寸作为设计和开发基准。

（2）定义一套适配规则，自动适配剩下的两种尺寸。

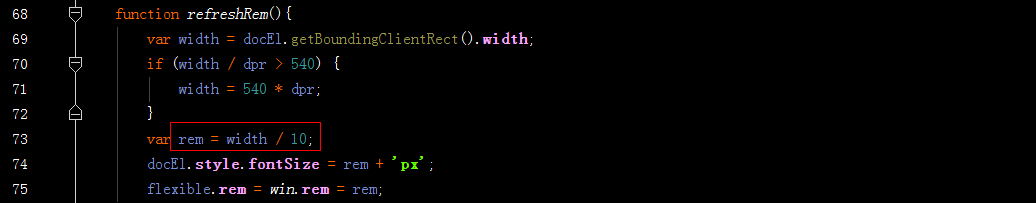
（3）特殊适配效果给出设计效果。



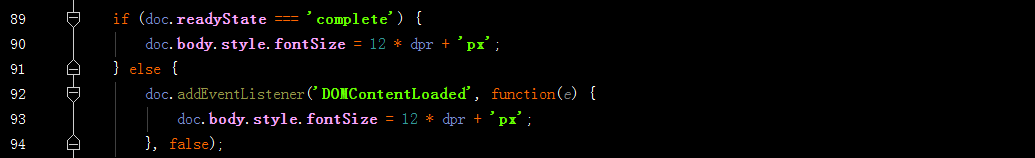
手淘设计师常选择iPhone6作为基准设计尺寸，交付给前端的设计尺寸是按750×1334px为准(高度会随着内容多少而改变)。

**2.实现**

（1）给html设置font-size，按屏幕宽度10等分计算。



（2）body的font-size不同于html。



**3.使用**

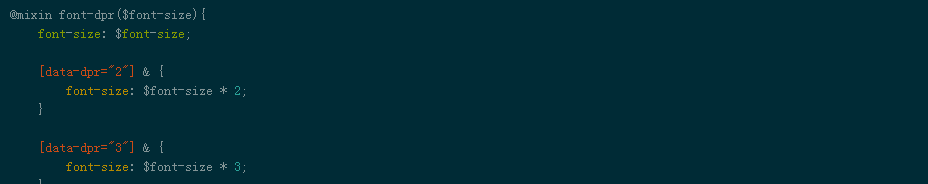
（1）将设计图中的像素照写，然后通过postCss自动改为rem。

Flexible会将视觉稿分成100份（主要为了以后能更好的兼容vh和vw），而每一份被称为一个单位a。同时1rem单位被认定为10a（1个除以100，1个除以10）。例如，对于iphone6，rem为75px（设计稿为2倍放大）。

（2）字体不使用rem。

主要目的如：不希望文本在Retina屏幕下变小，又希望在大屏手机上看到更多文本。另外，大多数的字体文件都自带点阵尺寸，通常是16px和24px，所以不希望出现13px和15px这样的奇葩尺寸。

如果使用rem，小屏会特别小，大屏特别大，且字体会出现上面的13、15px情况。这就用到给html添加的[data-dpr]属性。这样，通过postCss转换后，所有屏幕字体一样大。



1. 总结
   1. 样式总是作为特定属性组合出现

如color和bakground-color，通常以组合方式出现。除只做小调整，不然需要一起调整他们。

* 1. 隐藏图片式按钮中的文字

在bootstrap中定义了.hide-text混合。



font: ~"0/0" a表示：字号大小0，行高0，字体为a。a代表font-family，实际并无a字体，换成b也行。

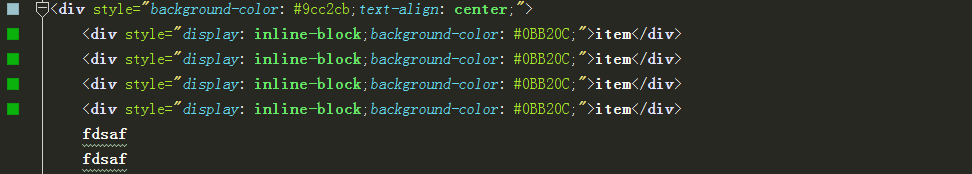
其他样式应该是去掉继承或默认值用。

* 1. transition多个属性连写



或使用transition-property。

* 1. 标签书写换行导致的空白间距



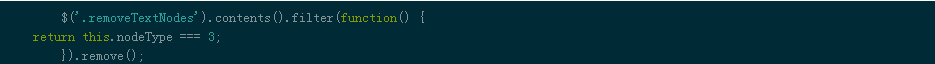


即使没有设置间距，因为这些标签在书写时换行，导致显示时元素之间也会有4px间距。除了把它们写在一行外，还可以通过如下方法：

（1））父元素font-size设为0，但不兼容safari，要兼容safari，还得设置letter-spacing和word-spacing。



（2）使用jquery，过滤掉inline-block元素之间的换行符文本（看不见的）。



* 1. 滚动条

**1.什么时候出现滚动条**

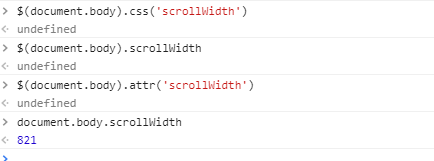
（1）父元素设置overflow=auto或scroll时。

（2）子元素的width/height大于父元素时。所以负margin，绝对定位、相对定位设置left等不会出现滚动条。

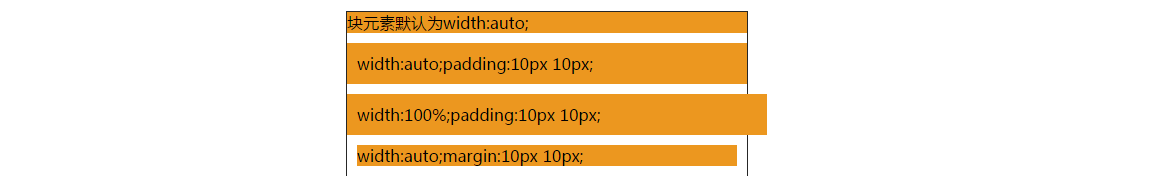
**2.计算**

（1）scrollTop：为元素最顶端据可见顶部的距离。因此，$(window).scrollTop()为上面看不见的部分。

（2）获取scrollWidth

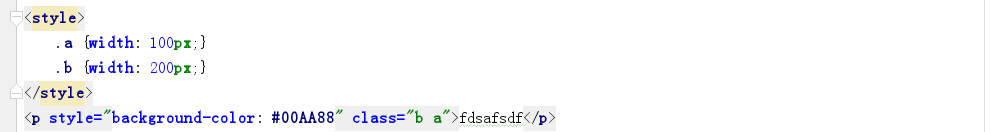


* 1. width:auto和width:100%
* width:auto：元素整个宽度将撑开至父元素width，并减去元素自己的margin/padding/border。
* width:100%：会强制将元素width等于父元素，并有自己额外的margin/padding/border，可能会导致很多问题。



* 1. 样式层叠覆盖问题

如果样式发生冲突，是根据样式定义的先后顺序，而不是使用顺序来进行覆盖。



尽管使用上，a类在b类之后，但因为定义在b类之前，最终样式还是被b覆盖。

* 1. a嵌套img

a标签默认为inline，嵌套img应将其设置为block或inline-block。img默认在垂直方向是基线对齐，此时应该设置为bottom。这样，嵌套时不会再出现a中底部留白问题。

* 1. a标签伪类样式

（1）经测，只要对a标签设置样式后，它的4个伪类都一并被设置了。

（2）四个伪类为：

1）:link未访问（这也是1个伪类）。

2）:visited当前页面没有关闭的情况下点击过，包括后退回来。

3）:hover。

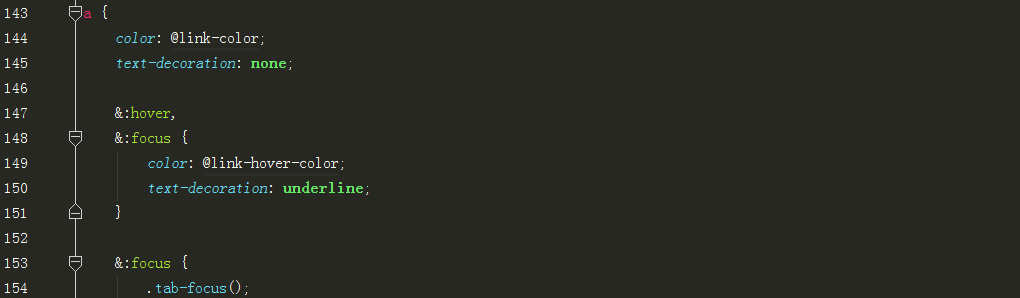
4）:active点击后鼠标还没有放开时。

4个伪类默认都有下划线text-decoration:underline，默认样式为：



（3）如果要单独设置4个伪类的样式，需要按固定顺序就是LVHA（爱恨原则LoVe/HAte）设置。

（4）bootstrap基础样式中设置了a的样式（相当于设置了所有伪类的样式），以及hover的样式，从而导致hover时有下划线。



* 1. li设置display:block

ul不做任何设置，对于li设置display:block（没有其他设置），现代浏览器不会显示左边圆点，但ie6/7会显示。

不要给外部容器设置height，如果有height，那也是在内部元素上设置，再由其自然撑开。

1. 一些不重要的问题
   1. 默认值

**1.使用inherit属性值继承其父元素样式来覆盖UA自带样式**

像button、input这样的表单控件，不同的浏览器都会有自己的样式风格（UA样式）。可以使用inherit继承其父元素样式，从而覆盖浏览器的UA样式。例如bootstrap中：



**2.页边距**

IE默认为10px，通过body的margin属性设置，FF默认为8px，通过body的padding属性设置。

**3.图片链接样式**

IE默认为紫色2px的边框线，FF默认为蓝色2px的边框线，因此需要都清除。

**4.元素居中**

IE默认为text-align:center;，FF默认为margin-left:auto;margin-right:auto;。

**5.段间距**

IE默认为19px，由p的margin-top属性设置。FF默认为1.12em，由p的margin-bottom属性设置。因此需要清除段间距。

**6.列表**

（1）ul、ol、dl都有上下margin。

（2）ul和ol有左padding（用于显示标点或数字），dl没有。

（3）ul和ol的list-style-type为disc/decimal，可以通过ul、ol统一设置，也可以对li单个设置。

（4）dl和dt无缩进，但内部元素dd有缩进。

**7.table**

（1）table的默认样式为table，不是block，更像是inline-block，独占1行，但并非100%，而是包裹其内容宽度（td也是内容决定其宽度）。

（2）单元格并非平分table宽度，即使设置table的width为100%，而是根据每个td中的内容宽度比例分配。

（3）td垂直居中，水平居左。th垂直水平均居中。td/th均有1px的padding。

（4）单元格默认分开，即border-collapse为separate（border-collapse与border-spacing和empty-cell相关，如果为collapse，将忽略两者的值，separate依赖两者的值）。

（5）对所有td/th设置border，设置border-collapse为collapse时，仍然只显示1px的border。

（6）一旦对table设置了width，里面的单元格绝对不会撑开table，而是按内容比例分配。如果为设置，则有内容决定（撑开）。

（7）不要对table/th/td以外的元素设置样式。

* 1. 元素嵌套规则

（1）CSS中，对于元素如何嵌套没有任何限制。但在HTML中，块级元素不能继承自内联元素。即，块级元素可以包含内联元素或某些块级元素，但内联元素不能包含块级元素，它只能包含其它内联元素。

（2）块级元素与块级元素并列、内联元素与内联元素并列（错误的：<div><h2></h2><span></span></div>）。

（3）有几个特殊的块级元素只能包含内联元素，不能包含块级元素。如h1-6/p/dt等。

（4）li内可以包含div。

* 1. 字体

（1）浏览器默认字体为“宋体/simsun字体/16像素”。

（2）因sans-serif/serif分别表示“无衬线字体”和“衬线字体”，一旦设置，其将终结font-family属性，即跟在sans-serif/serif后面的字体都无效。

（3）只要在css中对font/font-family属性中出现了对中文字体的设置，那么浏览器的自定义的默认字体就会失效。

（4）主要网站字体设置

|  |  |
| --- | --- |
| 淘宝 | 12px/1.5 tahoma,arial,'Hiragino Sans GB','\5b8b\4f53',sans-serif |
| 百度 | 没有设置，用默认值arial |
| 腾讯 | 12px 宋体, "Arial Narrow", HELVETICA; |
| 人人 | Tahoma,Verdana,STHeiTi,simsun,sans-serif |
| 搜狐 | "\5B8B\4F53","Arial Narrow",HELVETICA; |
| 滴滴 | "PingHei",Microsoft Yahei,SimSun,Arial,Helvetica,sans-serif; |
| 京东 | 12px/1.5 Microsoft YaHei,tahoma,arial,Hiragino Sans GB,\\5b8b\4f53,sans-serif; |
| 拉勾 | 14px/22px "Hiragino Sans GB", "Microsoft Yahei", "SimSun", Arial, "Helvetica Neue", Helvetica; |
| Uber | Hiragino Sans GB,ff-clan-web-pro,Helvetica Neue,Microsoft YaHei; |

（5）“\5b8b\4f53”意思是宋体，因为中文ie9不支持英文字体，如果没有设置中文字体，将导致line-height设置垂直居中文字不能垂直居中。

* 1. 书写规范

（1）使用html5的文档声明。前后不能有空格、空行。



（2）必须申明文档的编码charset，且与文件本身编码保持一致，推荐使用UTF-8编码。



（3）从ie8起，ie加入兼容模式。即，在meta标签中可以指定使用那种ie版本进行页面渲染。使用下面的写法，表示强制使用IE最新内核，并且如果安装了chrome插件，强制使用chrome渲染。



说明：只有银行等才会在兼容模式中使用ie6/7/8渲染。

（4）兼容国产浏览器，使用极速模式渲染。



（5）任何情况下，都能在移动端显示。



（6）必须写title，控制在25个字，50个字节以内。以“二级栏目-一级栏目-网站名称”书写。

（7）keywords使用5个单词左右，单个8汉字以内。

（8）description网站描述，字数尽量空制在80个汉字，160个字符以内。

（9）常用实体名称

|  |  |  |
| --- | --- | --- |
| 字符 | 名称 | 实体名 |
|  | 半角空格 | &nbsp; |
| < | 左尖括号（小于号） | &lt; |
| > | 右尖括号（大于号） | &gt; |
| & | &符 | &amp; |
| © | 版权 | &copy; |
| ® | 注册商标 | &reg; |
| ™ | 商标TM | &trade; |
| ¥ | 人民币符号：元 | &yen; |

（10）浮动的地方不要加注释，可能导致布局错位或文字的BUG。

（11）img添加alt属性，增加可访问性。

（12）自定义标签或者属性，用data-开头。

（13）需要程序单独赋值或者控制的地方尽量用独立的标签包括起来，方便添加id和其他自定义属性。

（14）避免元素的滚动条和body的滚动条同时出现，影响体验。

（15）需要异步加载的地方给出加载中（loading）状态效果。

（16）光标类型：default一个箭头、pointer一只手、crosshair十字线、text指示文本、help帮助（通常是一个问号或一个气球）。

* 1. CSS书写注意事项

（1）指定页面编码。



（2）中文字体转码：微软雅黑：\5FAE\8F6F\96C5\9ED1或MicrosoftYaHei；黑体：\9ED1\4F53；新宋体：\65b0\5b8b\4f53；宋体：\5b8b\4f53。

（3）不要使用@import引入css文件，应使用<link>标签。

（4）属性的书写顺序：

* 位置：position、left、right、float等。
* 盒模型属性：display、margin、padding、width、height等。
* 边框与背景：border、background。
* 段落与文本：line-height、text-indent、font、color、text-decoration等。
* 其他属性：overflow、cursor、visibility等。

（5）使用具有语义化的classname或id。

（6）勿使用冗余低效的CSS写法，例如：ul li a span{...}。

（7）上下模块之间的间距统一使用下一个模块的margin-top来实现。好处是即使没有下一个模块也不会多出一段空隙。

* 1. css命名相关

（1）结构相关：header(hd)、content(cnt)、footer(ft)、section、sidebar、column、row、cell。

（2）内容相关：title(tit)、summary、item；menu、submenu、dropdown；flink友情链接、copyright版权；hot热点、news新闻、banner/AD广告、download下载；share分享、digg顶、like喜欢；msg提示信息、status状态、vote投票、tips小技巧、guild指南、note注释；time。

（3）导航相关：nav导航、mainbav主导航、subnav子导航、topnav顶部导航、breadcrumb面包屑导航。

（4）用户相关：member会员、ucenter用户中心、loginBar登陆条、login登录、Btn-login登录按钮、btn-regsiter注册按钮、name用户名、password密码、nickname昵称。

（5）搜索相关：searchBar搜索条、search搜索条、searchTxt搜索框、searchBtn搜索按钮、search\_key搜索词。

（6）动作相关：transition动画、shadow阴影、fade淡入淡出、flip翻页、slide滑动、slideup上滑动、slidedown下滑动、turn翻页、horizontal水平、vertical垂直、collapsible折叠、corners拐角、flow流、reverse反向、pop弹窗。

* 1. HTML5
     1. 新写法和新元素

**1.新写法**

（1）doctype



（2）根元素也有所简化。可以只写一个<html>，也可以指明语言。



（3）HTML4和HTML5字符编码写法分别为：



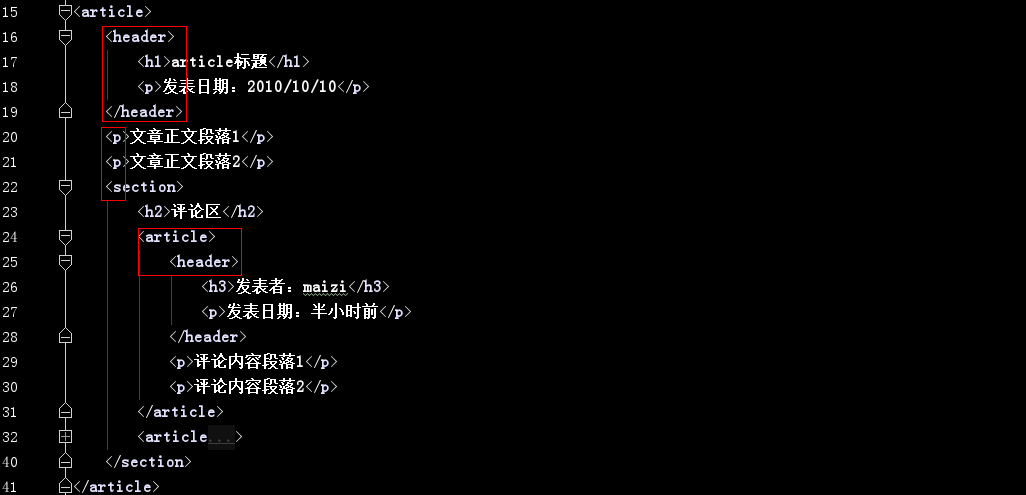


**2.新标签**

（1）<main>：页面的主体部分，1个页面最多只能有1个<main>。

（2）<section>：对页面上的内容分块，或对文章分段。1个<section>应由内容及其标题组成，不推荐为那些没有标题的内容使用<section>。如果符合<article>、<nav>、<aside>条件，那么就不要用<section>。

（3）<article>：独立的、完整的、可以独自被外部引用的内容。更多的是1篇博客或报刊中的文章、1篇论坛帖子、1段用户评论等。article可以嵌套（内部可以有section，也可以有article）。

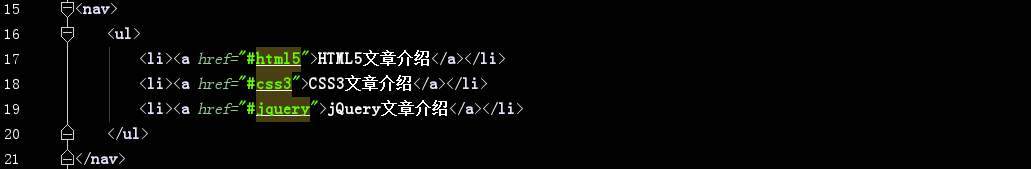


（4）<header>：可以是页眉，包含网站标志、主导航、全站链接以及搜索框等。也可以是文档、文档分块的导航性标记。

（5）<footer>：通常为作者、出版权信息等。通常1个页面1个footer。如果有必要，section也可以有footer。

（6）<aside>：侧边栏，与内容相关，与位置无关。应该是与主内容相关，但又不是主内容的部分，如作者介绍、相关产品列表、广告等。

（7）<nav>：用于导航的链接列表，内部是ul标签。



（8）h[n]：1个页面最多只能出现1次h1。如果标题与搜索词匹配，这些标题就会被赋予很高的权重，尤其是h1。

（9）Html5中，<em>是表示强调的唯一元素；<strong>表示重要程度；<b>出于实用目的提醒读者的一块文字，不传达任何额外的重要性；<i>表示不同于其他文字的文字或图标。

（10）<div>块级通用容器；<span>短语内容无语义容器。

* + 1. data-\*属性

（1）data-\*属性用于在DOM中存储自定义信息。

（2）js中，可以通过getAttribute和dataset访问。



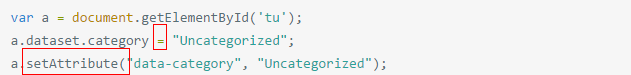




（2）css中，可以通过如下方式获取：



（3）可以通过如下方法修改data-\*属性。



* + 1. 利用HistoryAPI无刷新更改地址栏

（1）pushState方法



浏览器历史记录可以视为一个栈。执行pushState函数之后，会往浏览器的历史记录中添加一条新记录，同时改变地址栏的地址内容。pushState接收3个参数：

* 对象或者字符串，会被一并添加到历史记录中以供以后使用。
* 字符串，为新页面的标题。但几乎所有浏览器都会忽略这个参数。
* 字符串，新页面的相对地址。

（2）popstate事件

当用户点击浏览器的前进、后退按钮时，就会触发popstate事件。



e.state就是当初pushState时传入的第1个参数。

（3）replaceState方法

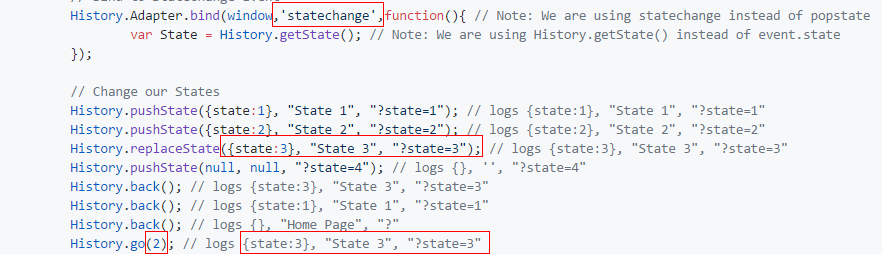
如果希望不添加一个新记录，而是替换当前的记录，则可以使用replaceState方法。参数和pushState完全一样。

（4）兼容性

IE10以上支持。可使用History.js库兼容老版本浏览器。

* + 1. History.js

简单使用如下：



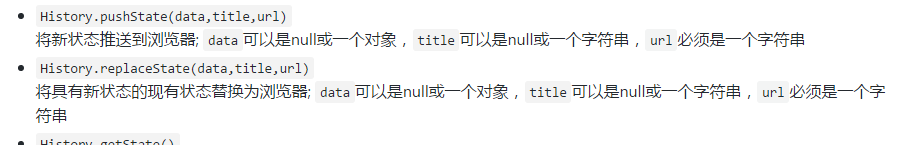
在支持HTML5浏览器中地址栏如下：



在支持HTML4浏览器中地址栏如下：



在HTML4浏览器中添加SUID是内部需要。然后，就可以使用History.js提供的API了。



* + 1. 新的表单类型

（1）添加增添placeholder、autofocus等新功能。

（2）新增search、color、email、url、number、range、date、time、datetime、month、week等13个新类型。

（3）设置某些type会自动验证。还可以使用pattern、min/max、required等属性约定验证。可以使用novalidate取消验证。

（4）新增form属性，可以将form标签外的input项目归属到某个form中。

* + 1. 地理位置API

（1）请求用户许可

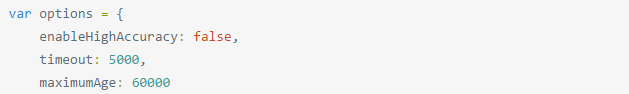


（2）获得地理位置

在上面on\_success回调中，会自动传入pos参数，可以获取。on\_error为错误处理回调。

（3）可选项

可选项可以设定如下参数：



* + 1. 本地存储

HTM5涉及存储的特性一共有3个，分别是WebStorage，WebSQLDatabase和IndexedDB。其中支持最广泛的是LocalStorage，其他存在兼容性。WebStorage主要存储键/值对，使用如下：



HTML5还为我们带来了和Storage相关的事件。事件发生有两个条件：（1）localStorage的内容被改变；（2）改变发生在另一个网页中（很多人都忽略了这个条件）。

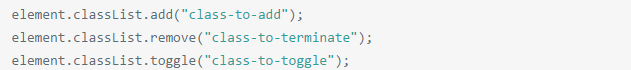
每个域名下的localStorage可以存储5M的数据。但这个大小并不是标准强制规定的，只是一个建议。

localStorage是按照scheme+hostname+uniqueport来区分的。

* + 1. 在线检测、contenteditable、classList等特性

HTML5添加了一个hidden属性，所有元素都可以有，效果就是把当前元素隐藏起来。

新增的classListAPI使得给元素添加、删除一个类更简单。



ContentEditable属性。添加之后，用户就可以更改其中的内容了。

navigator.onLine来检测现在用户是否联网。

* 1. IE浏览器兼容
     1. IE条件注释

通常放在<head>标签中，一些关键字说明：gt：以上版本（不包含）。gte：以上版本（包含）。lt：以下版本（不包含）。lte：以下版本（包含）。！：以外的版本。



* + 1. IE的Hack写法

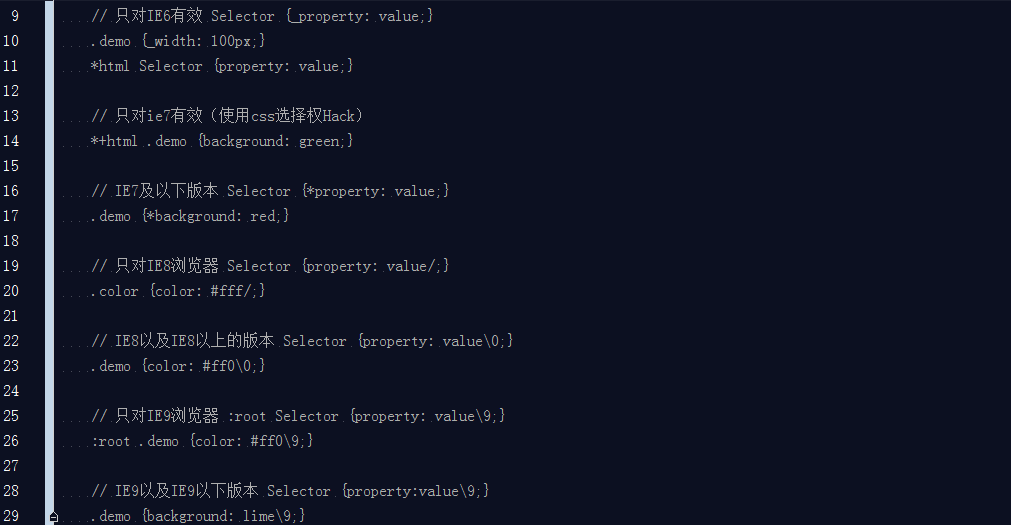
Hack是针对不同的浏览器去写不同的CSS样式，从而让各浏览器能达到一致的渲染效果。需要说明的是，未到情非得以的情况下才使用HACK。

**1.css选择器的Hack写法**



其他比较难记，用得比较少。

**2.css属性的Hack写法（结合css选择器Hack）**



* + 1. IE常见Bug

**1.浮动元素的双倍Margin的Bug（IE6及以下）**



如上，float为left，当有margin-left时，显示的margin-left加倍。



修复方法：添加“display:inline”属性。

**2.百分比尺寸bug（ie6）**

ie6对百分比尺寸不友好。

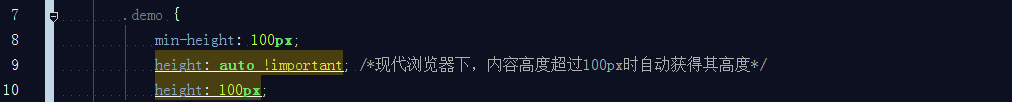
修复方法：使用!important（相当于没有设置百分比）。



**3.元素的min-width和min-height的bug（ie6）**

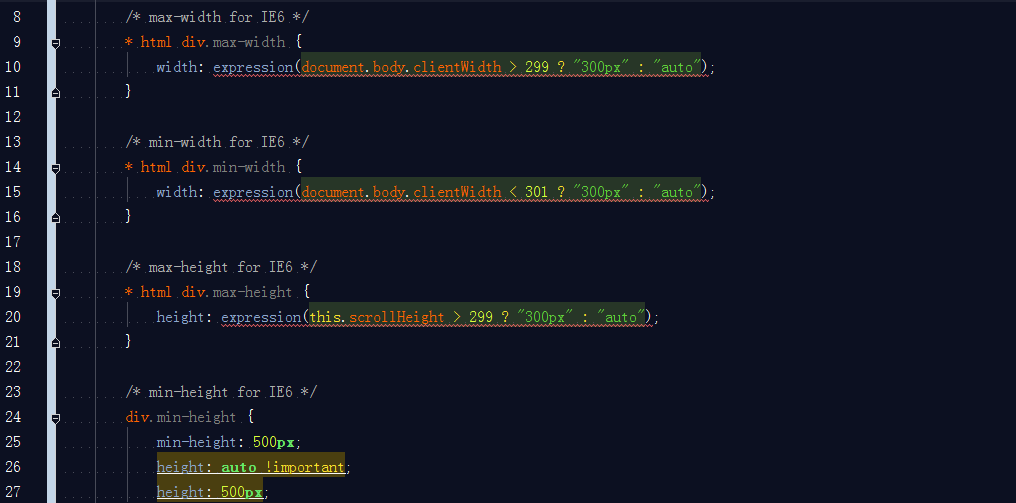
直接不识别。

修复方法：使用“!important”来解决。



**4.min-width/max-width和max-heigt/min-height的Bug（ie6）**

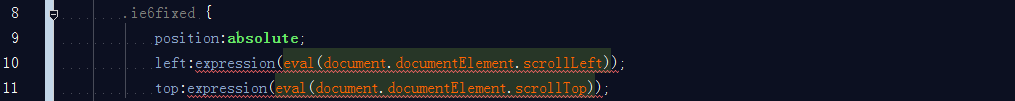
修复方法：使用ie6的css专用表达式。



**5.Ie6下position:fixed的Bug（ie6）**

直接不能识别。

修复方法：使用ie表达式。

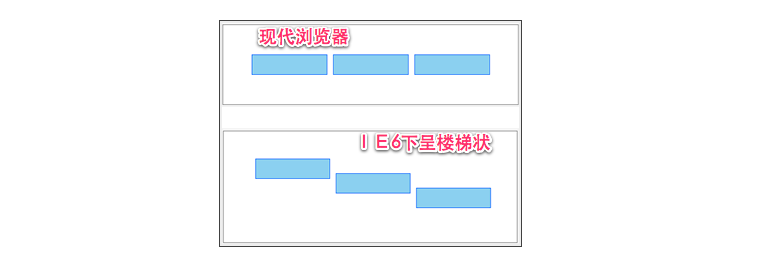


**6.Select中Option的宽度（ie6/7）**

当在<select>上设置一个静态的宽度时，当其<option>选项内容超过设置的宽度时，其内容也将直接被裁剪。

修复方法：只能使用js，当鼠标移动到select上时，宽度变成自动。

**7.列表li的楼梯Bug（ie6）**



修复方法：1）给每个li添加浮动。2）给每个li添加“display:inline”。



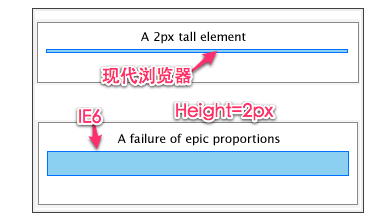
**8.li垂直方向空白间距（ie6）**



修复方法：1）给每个li>a添加width。2）给每个li>a添加float，并清除float。



**9.无法设置较小高度（ie6）**



如上，无法设置较小的height。原因是拒绝高度小于字号的设置。

修复方法：设置font-size=0，且line-height=0。也可以使用“overflow:hidden”实现。

**10.overflow:auto与position:relative冲突bug（ie6/7）**

两个块元素，其中父元素设置“overflow:auto”，子元素设置“position:relative”且其高度大于父元素，子元素不会被隐藏而溢出父元素块显示。

修复方法：给父元素也设置position:relative属性。

**11.浮动层错位（ie6）**

如果浮动元素的内容超过元素设置的width，普通浏览器仅仅是内容显示在元素外，而ie6则撑开元素宽度（即使是显式设置了width）。

修复方法：给元素添加“overflow:hidden”。

**12.块元素水平居中（IE6怪癖模式）**

IE6怪癖模式下不识别margin:auto。

修复方法：1）添加DOCTYPE声明。2）居中元素的父元素中加上“text-align:center”，然后在居中元素中加上“text-align:left”重新让元素（内部）文本左对齐。

**13.盒模型Bug（IE6以下，不包含ie6）**

属性width和height包含了padding和border的宽度。

修复方法：在标签内再嵌套1层标签，如下：



**14.显示注释（ie6）**

有的情况下，ie6可能会把注释显示出来。

修复方法：去掉注释。

**15.图片缩放bug（ie6/7）**

会自动减小图片分辨率。

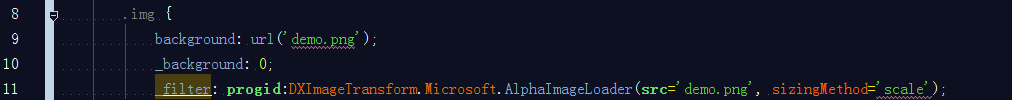
修复方法：添加如下样式。



**16.PNG图片透明Bug（ie6）**

IE6下的png透明部分会有一层淡蓝色的背景。

修复方法：使用滤镜。



**17.iframe背景bug（ie6/7/8）**

其他浏览器背景透明，但ie6/7/8不是。

修复方法：设置iframe属性allowTransparency为true，且设置其中的body背景色为透明。





**18.hover的bug（ie6）**

ie6只支持a:hover，其他元素hover均不支持。

修复方法：使用jQuery的hover。

**19.position:relative的Bug（ie6）**

ie6对元素设置的position:relative不友好。

修复方法：添加“zoom:1”。

**20.修复margin的负值（ie6）**

ie6对元素设置的负值margin不友好。

修复方法：添加“position:relative”和“zoom:1”。

**21.修复overflow的问题（ie6）**

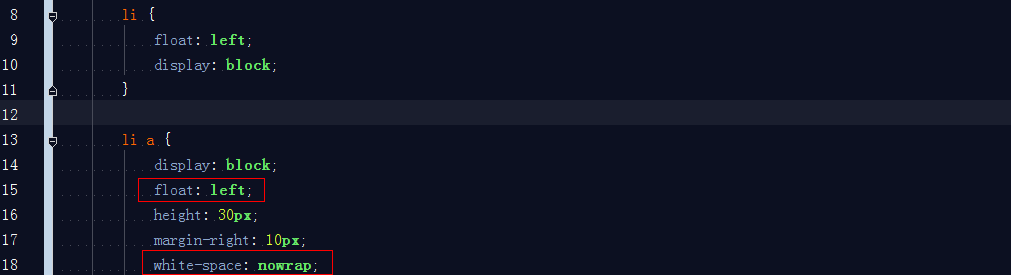
ie6对元素设置的overflow不友好。

修复方法：给元素添加“zoom:1”，且给该元素的父元素添加“overflow-x:hidden”。

**22.浮动bug（ie6）**

如果1个浮动元素的子元素触发hasLayout且没有设置宽度，会导致浮动元素不形成包裹而是100%占据整个宽度。

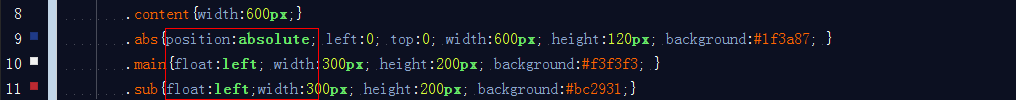
修复方法：给子元素也设置width，且设置“withe-space:nowrap”，以修复另外1个宽度不足时的bug。



**23.绝对定位、浮动元素混用时的BUG（ie6）**

包含块内，有2个浮动box，1个绝对定位box。如果设置如下样式，会发生IE6绝对定位元素消失的BUG。







修复方法：给content添加“display:inline”。

**24.100%高度的Bug（ie6）**

如果要给元素定义100%高度，必须要明确定义它的父级元素的高度，如果你需要给元素定义满屏的高度，就得先给html和body定义height:100%。

**25.3像素间距bug（ie6）**

当文本（或非浮动元素）跟在1个浮动的元素之后，文本和这个浮动元素之间会多出3像素的间隔。

修复方法：给浮动层添加“display:inline”和-3px负值margin。给中间的内容层定义margin-right。

**26.IE6调整窗口大小的Bug**

当把body居中放置，改变IE浏览器大小的时候，任何在body里面的相对定位元素都会固定不动了。

修复方法：给body定义“position:relative”。