--

[小知识点 1](#_Toc17000)

[多重背景 1](#_Toc6280)

[背景开始的范围 1](#_Toc4415)

[恢复默认样式 1](#_Toc29638)

[vertical-align 1](#_Toc27740)

[属性设置元素的垂直对齐方式。 1](#_Toc20357)

[inline-block去除间隙 1](#_Toc30461)

[margin padding top % 1](#_Toc9902)

[响应式布局media 1](#_Toc6777)

[产生一个随机颜色 1](#_Toc5874)

[Opacity 在IE浏览器中的使用方式 1](#_Toc14511)

[jss获取伪元素的属性 1](#_Toc29165)

[css控制元素不响应用户操作 1](#_Toc32509)

[文字毛玻璃效果 1](#_Toc12511)

[background-origin 1](#_Toc17844)

[background-clip 1](#_Toc9909)

[小tip:中文英文左右padding一致两端对齐实现 2](#_Toc2150)

[说明 2](#_Toc27561)

[实现原理 2](#_Toc9610)

[保证内部的div不会因为padding、margin而撑破外部的div 2](#_Toc5851)

[实现倒影 3](#_Toc18496)

[content："" + before + after 的应用 3](#_Toc15738)

### 小知识点

|  |  |  |
| --- | --- | --- |
|  |  | |
|  |  | |
| 恢复默认样式 | p{color:blue;}  p.nocolor{color:--initital;}  [initial属性值的作用是让各种属性使用默认值]  [在特殊情况下，对元素使用initial属性值后，并不等于将该元素的样式设定直接删除后的结果] | |
| vertical-align | 值 描述  baseline 默认。元素放置在父元素的基线上。  sub 垂直对齐文本的下标。  super 垂直对齐文本的上标  top 把元素的顶端与行中最高元素的顶端对齐  text-top 把元素的顶端与父元素字体的顶端对齐  middle 把此元素放置在父元素的中部。  bottom 把元素的顶端与行中最低的元素的顶端对齐。  text-bottom 把元素的底端与父元素字体的底端对齐。  length  % 使用 "line-height" 属性的百分比值来排列此元素。允许使用负值。  inherit 规定应该从父元素继承 vertical-align 属性的值。 | |
| inline-block去除间隙 | letter-spacing:-4px  font-size:0 | |
| margin padding top % | margin padding 的 % 是按照上层absolut元素的width，直至最顶层body  top 的 % 按照上一个(absolut)\(relative)元素来的，直至最顶层body | |
| 响应式布局media | @media only screen and (max-width: 570px) { } | |
| 产生一个随机颜色 | progressbarValue.css({  "background": '#' + Math.floor( Math.random() \* 16777215 ).toString( 16 )  }); | |
| Opacity 在IE浏览器中的使用方式 | filter:alpha(opacity=50); opacity:0.5; | |
| jss获取伪元素的属性 | <**script**>  **/\*JavaScript获取伪元素属性\*/** // Get the color value of .element:before  **var color** = window.getComputedStyle(  document.querySelector('.element'), ':before'  ).getPropertyValue('color');  // Get the content value of .element:before  **var content** = window.getComputedStyle(  document.querySelector('.element'), ':before'  ).getPropertyValue('content'); </**script**> | |
| css控制元素不响应用户操作 | 通过设置css中pointer-events属性的值为none，可以高效阻止元素的JavaScript事件行为（包含click和addEventListener) 基本用法： .disabled {  pointer-events:none; } | |
| 文字毛玻璃效果 | <**style**>  .normal{  width:300px;height:100px;  border:1px solid;  box-shadow:1px 1px 2px 2px #ccc;  font-size:26px;   text-shadow: 0 0 10px black;  **transition:all 0.5s ease;  color:rgba(0,0,0,0);**   }  .normal:hover{  color:rgba(0,0,0,1)  } </**style**> <**div** class="normal">文字毛玻璃效果</**div**> | |
| background-image  background-origin  background-clip | 多重背景：  background-image:url(bg\_flower.gif),url(bg\_flower\_2.gif);  背景开始的范围：  background-origin: padding-box|border-box|content-box;  background-clip: border-box|padding-box|content-box; | |
| background-clip 和 background-origin 是 CSS3 中新加的 [background module](http://www.w3.org/TR/css3-background/" \l "dependencies" \t "http://www.blueidea.com/tech/web/2008/_blank) 属性，用来确定背景的定位。  background-origin 用来决定 background-position 计算的参考位置。  background-clip 用来决定视口的大小，将对整个background(color、image)进行裁剪。  --backgroud-origin:[从哪里开始]  是指背景显示的区域，或者说背景是从哪个区域开始绘制(放置)的(边框、补白或内容区域)。  --backgroud-clip:[定位好了之后，背景的哪些部分不显示（显示的部分貌似被剪出来了）]  规定背景图片或颜色能否在边框下显示，默认值是border-box；  值为border-box则背景会在元素的border+padding+content区域中显示  值为padding-box则背景只会在元素的padding+content区域中显示  值为content-box则背景只会在元素的content中进行显示  background-origin主要用于控制背景图片的放置位置（默认padding-box）  background-clip 主要用于控制视口（默认border-box，代表可见） | |
| <**style**>  .basic{  display: inline-block;  width: 100px;  height: 100px;  padding: 20px;  border: 10px dashed lightslategray;  background-color: lightcoral;  background-image: url(**"images/1.jpg"**);  background-repeat: no-repeat;  }  .bg-a{  background-clip: border-box;**/\*默认\*/**  }  .bg-b{  background-clip: padding-box;  }  .bg-c{  background-clip: content-box;  }  .bg-d{   } </**style**> <**div** class="basic bg-a"></**div**> <**div** class="basic bg-b"></**div**> <**div** class="basic bg-c"></**div**> <**br**/> <**div** class="basic bg-d"></**div**> |  |
| <**style**>  .basic{  display: inline-block;  width: 100px;  height: 100px;  padding: 20px;  border: 10px dashed lightslategray;  background-color: lightcoral;  background-image: url(**"images/1.jpg"**);  background-repeat: no-repeat;  }  .bg-a{  -webkit-background-origin: border-box;  }  .bg-b{  -webkit-background-origin: padding-box;**/\*默认\*/**  }  .bg-c{  -webkit-background-origin: content-box;  }  .bg-d{  } </**style**> <**div** class="basic bg-a"></**div**> <**div** class="basic bg-b"></**div**> <**div** class="basic bg-c"></**div**> <**br**/> <**div** class="basic bg-d"></**div**> |  |

小tip:中文英文左右padding一致两端对齐实现

|  |  |
| --- | --- |
| 说明 | IMG_257  就是一个定宽的容器，左右padding值20像素，结果输入一段文字后（有中文也有英文字符），会发现右侧根本就不对齐，有些地方距离右侧的空白大小也不是20像素，感觉不和谐，设计师就希望排列能够好看一点，右边要对齐。  其实呢，要实现两端对齐很简单，使用text-align:justify, 之前有写过“[display:inline-block/text-align:justify下列表的两端对齐布局](http://www.zhangxinxu.com/wordpress/?p=1514)”，也是属于text-align:justify的精彩应用；然后配合其他一些小技巧，就能实现任意中英文的两端对齐排列效果了。  您可以狠狠地点击这里：[中文或英文两端对齐Demo](http://www.zhangxinxu.com/study/201508/chinese-english-text-justify.html" \t "http://www.zhangxinxu.com/wordpress/2015/08/chinese-english-same-padding-text-justify/_blank)  我们可以看到，demo页面上有一段朴实的文字：1  然后我们点击按钮变身一下，会发现，文字内容乖乖两端对齐了：2  这种对齐效果，IE6/IE7浏览器也是支持良好的，例如，下面这个IE7下的截图：3  IMG_258IMG_259IMG_260 |
| 实现原理 | 要理解原理，我们首先要搞清楚文本的两端对齐声明text-align:justify起作用的本质。首先，大家要知道，text-align:justify是专门为英文设计的，谁叫CSS是老外发明的呢，用来实现英文语句的两端对齐。注意这里的，是语句的对齐。大家应该都知道，英文语句是一个单词一个单词组合而成的，每个单词之前使用空格分隔。  text-align:justify之所以可以让英文段落两端对齐，那是因为每个英文单词之前那个透明看不见的空格被拉伸了。注意，是空格被拉伸了，对，只有空格。因此，当我们写下一段洋洋洒洒中文内容的时候，text-align:justify是没有任何作用的，跟没设置没任何区别。为什么呢？很简单，因为中文是一个一个汉字，汉字之间是没有空格的，自然也就不能拉伸，自然也就不能两端对齐。  怎么办呢？难道中文就不行了吗？  当然不是！既然正常的中文文字之间没有空格，我们自己加一点不就好了。例如下面代码：  "宁泽涛小鲜肉".split("").join(" ");  结果就是： IMG_261  会发现，每个中文文字之间都有的空格字符，于是，text-align:justify就能大发神威，实现两端对齐！  OK，现在两端对齐是实现了，但是，空格字符它也是真实存在的字符，是会占据宽度的。如果放任不管，文字就会太稀松，阅读很吃力，会被当bug提出来的，怎么办呢？  很简单，我们使用letter-spacing收缩字符间距就可以了。例如demo页面：  box.style.letterSpacing = '-.15em';  不同字体的letter-spacing负值不一样，使用em单位呢，可以更智能适配各个font-size的文字。demo使用的微软雅黑字体，我大致测试了下，-.15em是个挺合适的值，其他字体我就不详了，你有兴趣，可以自己捣腾下。  至此，我们的任意中文英文的两端对齐效果就实现了。  使用JS再总结下就是（假设box是文字容器元素）：  ox.style.textAlign = "justify";  box.style.letterSpacing = '-.15em';  box.innerHTML = box.innerHTML.split("").join(" ");  寥寥3行代码，我们就大功告成了！  补充于翌日：  上面的方法适用于中文内容，如果段落含有英文片段，会干掉原来的空格，因为三个空格会被当做一个空格距离处理，因此，我们还需要对连续3个空格左下特殊处理，因此代码进化成：  box.innerHTML = box.innerHTML.replace(/^[\s\uFEFF\xA0]+|[\s\uFEFF\xA0]+$/g, '').split("").join(" ").replace(/\s{3}/g, " &nbsp; "); |

保证内部的div不会因为padding、margin而撑破外部的div

|  |  |
| --- | --- |
| box-sizing border-box | .calcwai {  width: 300px;  padding: 4px;  background: yellow;  border: 3px solid red; } .calcnei {  width: 100%;  box-sizing: border-box; /\*控制内部的元素不会超出border\*/  background: lightcoral;  background-clip: content-box;  height: 50px;  border: 1px solid darkgreen;  padding: 10px; }  <div class="calcwai">  <div class="calcnei"></div> </div> |
| calc(---) 计算 | .calcwai{  background: yellow;  width: 300px;  padding: 4px;  border: 3px solid red; } .calcnei{  width: -webkit-calc(100%-(1+10+4)\*2);  background: lightcoral;  background-clip: content-box;  height:50px;  border: 1px solid darkgreen;  padding: 10px; } |
|  | calc是英文单词calculate(计算)的缩写，是[css3](http://caibaojian.com/t/css3" \o "css3)的一个新增的功能，你可以使用calc()给元素的border、margin、pading、font-size和width等属性设置动态值。  以前我们可以使用box-sizing:border-box;来设置盒子的属性为不加上边距。现在我们又多了一个选择了。但要注意，两者只能使用一个哦，否则就会造成冲突了。  怎么使用  calc()可以使用数学运算中的简单加（+）、减（-）、乘（\*）和除（/）来解决问题，而且还可以根据单位如px,em,[rem](http://caibaojian.com/t/rem" \o "rem)和百分比来转化计算。  比如三局平均分布的布局，中间间距为5像素。  IMG_256  \* {  margin: 0;  padding: 0;  }  .col-3 {  width: calc(100%/3 - 5px);  float: left;  margin-right: calc(5px\*3 /2);  background: #eee;  color: #333;  height: 100px;  text-align: center;  margin-bottom: calc(5px\*3 /2);  font-size: 18px;  line-height: 100px;  }  .col-3:nth-child(3) {  margin-right: 0;  } |

实现倒影

|  |  |
| --- | --- |
| --box-reflect | -webkit-box-reflect:below 5px -webkit-gradient(linear, left top, left bottom, from(transparent), color-stop(0.5, transparent), to(white));    [其中transparent类似于rgba(255,255,255,0)] |
| background 1 | <**style**>  .demo-2 {  position: relative; **/\*作用于\*/** }  .demo-2 > **img** {  -webkit-transform: scaleY(-1);  }  .reflect-2 {  position: absolute;  left: 0px;  top: 0px;  width: 88px;  height: 115px;  background-image: -webkit-gradient(linear, left top, left bottom, color-stop(0.9, rgb(255, 255, 255)), color-stop(0.1, rgba(255, 255, 255, 0)));  **/\*类型为linear 从顶部开始到底部 0.1的地方是黑色，0.9的地方是白色，由此进行渐变\*/  /\*background:-webkit-gradient(linear,left top, left bottom, from(transparent),to(black),color-stop(0.5,white));\*/** } </**style**> <**img** src="images/1.jpg" alt="" width="88" height="115"/>  <**div** class="demo-2">  <**img** src="images/1.jpg" alt="" width="88" height="115"/>   <**div** class="reflect-2"></**div**> </**div**> |
| background 2 | <**style** type="text/css">  .demo-1-div {  width: 88px;  height: 115px;  -webkit-transform: rotate(180deg);  background: -webkit-linear-gradient(top, rgba(255, 255, 255, 1), rgba(255, 255, 255, 0)), url(./images/1.jpg);  background-size: 100% 100%;  } </**style**> <**img** src="images/1.jpg" alt="" width="88" height="115" id="demo-1-img"/> <**div** class="demo-1-div"></**div**> |

### content："" + before + after 的应用

|  |  |
| --- | --- |
| content | [normal|string|attr()|uri()|counter()|none]  平常仅仅需要将这两个伪元素用于添加一些自定义字符时，只需使用伪类使用的单冒号写法，以保证浏览器的兼容性：  p:before  {}  不过，在 CSS3 中为了区别伪元素和伪类为伪元素使用了双冒号，因此如果使用了 display 或者 width 等属性时使得显示脱离了原本元素后，建议按照标准双写。过于老的浏览器可能会存在支持问题，不过伪元素大多是配合 CSS3 使用，就无所谓向下兼容了：  img::after {}  这两个伪类下特有的属性 content ，用于在 CSS 渲染中向元素逻辑上的头部或尾部添加内容。注意这些添加不会改变文档内容，不会出现在 DOM 中，不可复制，仅仅是在 CSS 渲染层加入。比较有用的是以下几个值：  [String] – 使用引号包括一段字符串，将会向元素内容中添加字符串。示例：  a:after { content: "↗"; }  attr() – 调用当前元素的属性，可以方便的比如将图片的 Alt 提示文字或者链接的 Href 地址显示出来。示例：  a:after { content:"(" attr(href) ")"; }  url() / uri() – 用于引用媒体文件。示例：  h1::before { content: url(logo.png); }  counter() –  调用计数器，可以不使用列表元素实现序号功能。具体请参见 counter-increment 和 counter-reset 属性的用法。示例：  h2:before { counter-increment: chapter; content: "Chapter " counter(chapter) ". " } |
| 计数器的使用 | .da{  counter-reset:n; } .da div:after{  content: "值" counter(n);  counter-increment: n; }  <div class="da">  <div>1</div>  <div>1</div>  <div>1</div> </div>  [.da：父类中counter-reset:n; .da div:子类中实现增长：increment ] |
| 显示相对应的链接图标 | 指的是针对不同的链接类型，在链接的后面显示对应链接类型的图标，例如，链接的对象是一个图片，则显示图片的小图标，如果链接对象是视频，则显示视频小图标，如果链接就是个URL网页链接，则显示链接的小图标。您可以参见下面的代码：  CSS代码：  p a[href $=".pdf"]:after {  content:url(../image/icon\_pdf.png);  }  p a[rel = "external"]:after {  content:url(../image/icon\_link.png);  }  HTML代码：  <p>您可以查看此PDF文件：<a href="/sample.pdf">web站点的性能优化.pdf</a>，或是在线查看，<a href="http://www.zhangxinxu.com/wordpress/" rel="external">点击这里</a>。</p>  结果如下图所示：  CSS content 显示相对应的链接图标 张鑫旭-鑫空间-鑫生活 |
| 使用属性值attr作为content内容 | a:after {  content: "(" attr(href) ")";  }  abbr:after {  content: "(" attr(title) ")";  }  [CSS3 Values and Units](http://www.w3.org/TR/css3-values/" \l "attribute)草案扩展了attr()表达式的范围，除了返回字符串，还可以返回诸如CSS colors, CSS integer, length, angle, time, frequency以及其他一些单元。  通过使用自定义的data属性，可以实现一些非常强大的效果，诸如简单图表图形的渲染，动画效果的实现。例如我们可以设置根据元素的attribute值设定background-color的颜色值，这在显示在线调色板上会大放异彩；我们还可以根据attribute值指定元素的大小，就像定义图表每个条形的长度一样。总之，attr()的潜力惊人。 |
|  | |
| :before content open-quote | .blockquote{  position: absolute;  width: 100px;  height: 100px;  background: aquamarine;  padding: 30px;  -webkit-background-clip: content-box;  border: 1px solid red; } .blockquote:before {  content: open-quote;  position: absolute;  z-index: 1; **/\*控制显示的次序\*/** color: #DDD;  font-size: 100px;  font-family: serif;  font-weight: bolder;  left: -8px;  top: -8px; } |
| :before实现框框 | <**style**>  **a** {  position: relative;  display: inline-block;  outline: none;  text-decoration: none;  color: #000;  font-size: 32px;  padding: 0 10px;  }  **/\* 大框 \*/  a**:hover::before, **a**:hover::after {  content: **""**;  display: block;  position: absolute;  top: -15%;  left: -14%;  width: 120%;  height: 120%;  border: 4px solid #DDDDDD;  }  **/\* 小框 \*/  a**:hover::after {  top: 0%;  left: 0%;  width: 100%;  height: 100%;  border-width: 2px;  } </**style**> <**a** href="#">点击</**a**> |
| content  +  before  方块 | **a** {  position: relative;  display: inline-block;  outline: none;  text-decoration: none;  color: #000;  font-size: 32px;  padding: 5px 10px; }  **a**:hover::before, **a**:hover::after {  position: absolute; }  **a**:hover::before {  content: **"\5B"**;  left: -20px; } |
| 不同语言不同的引号 | <**style**>  **q**:lang(en) {  quotes: **'"' '"' "'" "'"**;  } </**style**> <**p**><**q**>This is a <**q**>big</**q**> quote</**q**></**p**> |
| none: 规定 "content" 属性的 "open-quote" 和 "close-quote" 的值不会产生任何引号。 s s s s: 定义要使用的引号。 前两个值规定第一级引用嵌套，后两个值规定下一级引号嵌套。 |
| CSS代码：  /\* 为不同语言指定引号的表现 \*/  :lang(en) > q {quotes:'"' '"';}  :lang(no) > q {quotes:"«" "»";}  :lang(ch) > q {quotes:"“" "”";}  /\* 在q标签的前后插入引号 \*/(貌似多余)  q:before {content:open-quote;}  q:after {content:close-quote;}  HTML代码：  <p lang="en"><q>It’s only s you do it.</q></p>  <p lang="no"><q>Hvis du fortel jeg slå deg til jorden.</q></p>  <p lang="ch"><q>欢迎来到上海，欢迎参观世博会！</q></p> |