--

[理论方面 1](#_Toc6010)

[www 1](#_Toc26147)

[CSS3初级 2](#_Toc1208)

[--border-radius 2](#_Toc15418)

[--box-shadow 2](#_Toc805)

[www 2](#_Toc12419)

[box-shadow与border的区别 2](#_Toc695)

[写法1-重叠 3](#_Toc6944)

[写法2-不同边 3](#_Toc2825)

[写法3-inset 3](#_Toc7506)

[使用样例 3](#_Toc22560)

[--text-shadow 3](#_Toc19368)

[demo 3](#_Toc23141)

[--transfrom 4](#_Toc12959)

[基础 4](#_Toc6993)

[矩阵matrix 4](#_Toc20746)

[位移 4](#_Toc21666)

[缩放 4](#_Toc16131)

[旋转 4](#_Toc13842)

[拉伸(skew) 4](#_Toc11582)

[transform-origin 4](#_Toc30180)

[例子 4](#_Toc2019)

[具体测试matrix 5](#_Toc27335)

[--transition 5](#_Toc8548)

[--animation 6](#_Toc8671)

[animation 与transition的简单对比 6](#_Toc11503)

[animation 与 transition 的启动方式 6](#_Toc14827)

[transition-滚动 6](#_Toc19810)

[animation-运动 6](#_Toc21740)

[animation-颜色渐变效果 7](#_Toc25838)

[3d prespective transform-style 7](#_Toc15289)

[CSS3 perspective 属性 7](#_Toc29391)

[--grandient 7](#_Toc27483)

[说明 7](#_Toc30654)

[写法 7](#_Toc16940)

[线性渐变 8](#_Toc24336)

[径向渐变 8](#_Toc29451)

[关于文本样式 8](#_Toc6550)

[CSS中的单位[颜色] 9](#_Toc22314)

[媒体查询@media 10](#_Toc16576)

[简介 10](#_Toc15161)

[demo 10](#_Toc2959)

[使用CSS制作各种形状的图形 10](#_Toc27388)

[对 border的应用 10](#_Toc27734)

[对border-radius的应用 11](#_Toc5782)

[n边形 11](#_Toc9571)

[img的srcset、sizes属性, w描述符 13](#_Toc18178)

[关于img的srcset 13](#_Toc21505)

[关于图片的响应实现 13](#_Toc924)

[用字体画icon小图标 14](#_Toc11380)

[方法1 14](#_Toc6309)

[方法2 14](#_Toc14039)

[font-face 14](#_Toc12065)

理论方面

### www

|  |  |  |
| --- | --- | --- |
|  | CSS3属性教程与案例分享  <http://www.w3cplus.com/resources/css3-tutorial-and-case>  [21条CSS高级技巧](http://caibaojian.com/css-protips.html)  <http://caibaojian.com/css-protips.html>  [Vue.js起手式+Vue小作品实战](http://caibaojian.com/learning-vue-js.html)  <http://caibaojian.com/learning-vue-js.html> | |
| 藤藤每日一练  <http://www.w3cplus.com/demo/tags/242.html> | |
| 特效  <http://www.5iweb.com.cn/> |
| 框架搭建  <http://my.csdn.net/zoutongyuan> |
| 高质量前端资源汇聚  Div.IO |
| 关于inline-block最好的一篇文章  http://forsigner.com/2013/03/28/inline-block/ |
| 各种效果演示  http://tympanus.net/Development/MorphingDevices/ |
| requestAnimationFrame还有毛线用  http://www.zhangxinxu.com/wordpress/2013/09/css3-animation-requestanimationframe-tween-%E5%8A%A8%E7%94%BB%E7%AE%97%E6%B3%95/ |
| CSS3 Transitions, Transforms和Animation使用简介与应用展示  http://www.zhangxinxu.com/wordpress/2010/11/css3-transitions-transforms-animation-introduction/ |
| [JavaScript设计模式与开发实践](http://www.baidu.com/link?url=XMfT815Aveu709G7qXjvN4l59lgWTLa2_V7vrO05Dx0Pr15HE05rgS5MZT8nXKts" \t "https://www.baidu.com/_blank)  http://www.jb51.net/books/419084.html |

|  |  |
| --- | --- |
|  | CSS 函数的 8 个妙用  <http://web.jobbole.com/87821/>  [探讨判断横竖屏的最佳实现](https://jdc.jd.com/archives/3862" \o "探讨判断横竖屏的最佳实现)  <https://jdc.jd.com/archives/3862>  京东前端开发团队  <https://jdc.jd.com/archives/category/5-frontend>  <http://blog.bingo929.com/transform-translate3d-translatez-transition-gpu-hardware-acceleration.html>  使用CSS3开启GPU硬件加速提升网站动画渲染性能 |

CSS3初级

|  |  |  |
| --- | --- | --- |
| --border-radius | | |
| demo | <style>  #test{  width: 100px;  height: 100px;  background: aquamarine;  /\*以下两者效果相同\*/  border-radius: 50px;  border-radius: 50%;  } </style> <div id="test"></div> |  |
| border-radius: 100px/70px; |  |
| border-radius: 100px 25px 80px 5px /  45px 25px 30px 15px; |  |

--box-shadow

|  |  |  |  |
| --- | --- | --- | --- |
| www | <http://www.w3cplus.com/content/css3-box-shadow> | | |
|  | 语法：  E {box-shadow: <length> <length> <length>?<length>?||<color>}  也就是：  E {box-shadow:inset x-offset y-offset blur-radius spread-radius color}  换句说：  对象选择器 {box-shadow:投影方式 X轴偏移量 Y轴偏移量 阴影模糊半径 阴影扩展半径 阴影颜色}  取值：  box-shadow属性至多有6个参数设置，他们分别取值：  阴影类型：此参数是一个可选值，如果不设值，其默认的投影方式是外阴影；如果取其唯一值“inset”,就是将外阴影变成内阴影，也就是说设置阴影类型为“inset”时，其投影就是内阴影；  X-offset:是指阴影水平偏移量其值可以是正负值可以取正负值，如果值为正值，则阴影在对象的右边，反之其值为负值时，阴影在对象的左边；  Y-offset:是指阴影的垂直偏移量，其值也可以是正负值，如果为正值，阴影在对象的底部，反之其值为负值时，阴影在对象的顶部；  阴影模糊半径：此参数是可选，，但其值只能是为正值，如果其值为0时，表示阴影不具有模糊效果，其值越大阴影的边缘就越模糊；  阴影扩展半径：此参数可选，其值可以是正负值，如果值为正，则整个阴影都延展扩大，反之值为负值是，则缩小  阴影颜色：此参数可选，如果不设定任何颜色时，浏览器会取默认色，但各浏览器默认色不一样，特别是在webkit内核下的safari和chrome浏览器将无色，也就是透明，建议不要省略此参数。  浏览器的兼容：  IMG_256  层次  边框>内阴影>背景图片>背景颜色>外阴影    这张图可以告诉我们很多信息，比如说borer-radius圆角，阴影扩展、阴影模糊以及padding是如何影响对象阴影的：非零值的border-radius将会以相同的作用影响阴影的外形，但border-image不会影响对象阴影的任何外形；对象阴影同box模型的层次一样，外阴影会在对象背景之下，内阴影会在边框之下背景之上。所以整个层级就是：边框>内阴影>背景图片>背景颜色>外阴影。因为大家都知道，我们的背景图片是在背景颜色之上的。 | | |
|  | IE滤镜方法：  在前面我们讲过，IE9以下是不支持CSS3的box-shadow的，但为了处理这个兼容问题，我们可以在IE下使用IE的shadow阴影滤镜来实现  filter: progid:DXImageTransform.Microsoft.Shadow(color=’颜色值’, Direction=阴影角度（数值）, Strength=阴影半径（数值）);    注意：该[滤镜](http://msdn.microsoft.com/en-us/library/ms532979(VS.85).aspx" \t "http://www.w3cplus.com/content/_blank)必须配合background属性一起使用，否则该滤镜失效。除了使用[滤镜](http://msdn.microsoft.com/en-us/library/ms532979(VS.85).aspx" \t "http://www.w3cplus.com/content/_blank)的方法外，我们还有一种方法可以实现IE下的效果。那就是使用jQuery的插件[jquery.boxshadow.js](http://www.hintzmann.dk/testcenter/js/jquery/boxshadow/jquery.boxshadow.js" \t "http://www.w3cplus.com/content/_blank)。那么具体如何使用呢？其实很简单，你先下载这个[jquery.boxshadow.js](http://www.opensource.org/licenses/mit-license.php" \t "http://www.w3cplus.com/content/_blank)插件到你的项目中，接着把jquery版本库和jquery.boxshadow.js加载到页面上，如  <script type="text/javascript" src="../js/jquery.min.js"></script>  <script type="text/javascript" src="../js/jquery.boxshadow.js"></script>    然后你可以创建一个单独的js文件来处理，或者直接在页面的<head></head>里欠入一个<script> </script>，我们这里就只例出一个直接在head插入的解决办法：  $(document).ready(function(){  if($.browser.msie) {  $('.demo1').boxShadow(0,0,5,"#888"); //demo1元素使用了box-shadow  $('.demo2').boxShadow(-10,-10,5,"#f36"); //demo2元素使用了box-shadow  }  }); | | |
| box-shadow与border的区别 | <**style**>  #test\_border{  width: 100px;  height: 100px;  background: aquamarine;  border: 30px solid red;  }  #test\_box\_shadow{  width: 100px;  height: 100px;  background: aquamarine;  box-shadow: 0 0 0 30px red;  } </**style**> <**div** id="test\_border"></**div**> <**div** id="test\_box\_shadow"></**div**> | | box-shaow不会影响布局 |
|  | <style>  .fa {  width: 20px;  height: 20px;  border: 20px solid #000;  box-shadow: 20px 20px 0px 0px lightcoral;  }  </style>  <div class="fa"></div> | | 包含border |
|  | <**style**>  .box-shadow-1 {  width: 100px;  height: 30px;  background: lemonchiffon;  }  .box-shadow-1:hover {  box-shadow: -10px 0 0 green;  } </**style**> <**div** class="box-shadow-1"></**div**> <**div** class="box-shadow-1"></**div**> <**div** class="box-shadow-1"></**div**> <**div** class="box-shadow-1"></**div**> | | 用于获焦边框可以除去边框宽度的考虑 |
| 写法1-重叠 | box-shadow:0 0 30px 20px red,0 0 50px 40px blue;  box-shadow:0 0 50px 40px blue,0 0 30px 20px red; | | 重叠的效果 |
| 写法2-不同边 | <**style**>  .normal {  width: 100px;  height: 100px;  background: lemonchiffon;  }  .box-shadow-1 {  box-shadow: -10px 0 0 green;  }  .box-shadow-2 {  box-shadow: -10px 0 0 green,  0 -10px 0 blue,  0 10px 0 red,  10px 0 0 yellow;  } </**style**>  <**div** class="normal box-shadow-1"></**div**> <**br**/> <**br**/> <**div** class="normal box-shadow-2"></**div**> | | 边不同效果 |
| 写法3-inset | <**style**>  .normal {  width: 100px;  height: 100px;  border: 1px solid black;  background: lemonchiffon;  }  .box-shadow-1 {  box-shadow: 0 0 30px 0 red, inset 0 0 10px 0 red;  }  .box-shadow-2 {  box-shadow: 0 0 0 30px red, inset 0 0 0 10px red;  }  .box-shadow-3 {  box-shadow: 0 0 30px 30px red, inset 0 0 10px 10px red;  } </**style**>  <**div** class="normal box-shadow-1"></**div**> <**br**/> <**br**/> <**br**/> <**div** class="normal box-shadow-2"></**div**> <**br**/> <**br**/> <**br**/> <**div** class="normal box-shadow-3"></**div**> | | 内外三种写法的对比 |
| 使用样例 | <**style**>  .drop-shadow {  position: relative;  margin: auto;  width: 300px;  height: 150px;  background: #ccc;  }   .drop-shadow:before,  .drop-shadow:after {  content: **""**;  position: absolute;  z-index: -1; **/\*置后\*/** width: 50%;  max-width: 150px;  height: 20%;  **/\*add box-shadow\*/** -webkit-box-shadow: 0 15px 10px rgba(125,125,125,0.8);  -moz-box-shadow: 0 15px 10px rgba(125,125,125,0.8);  box-shadow: 0 15px 10px rgba(125,125,125,0.8);  }   .drop-shadow:before{  bottom: 15px;  left: 10px;  -webkit-transform: rotate(-3deg);  -moz-transform: rotate(-3deg);  -o-transform: rotate(-3deg);  transform: rotate(-3deg);  }  .drop-shadow:after {  bottom: 15px;  right:10px;  -webkit-transform:rotate(3deg);  -moz-transform:rotate(3deg);  -o-transform:rotate(3deg);  transform:rotate(3deg);  } </**style**> <**div** class="drop-shadow">drop shadow effect</**div**> | |  |
| --text-shadow | | | |
|  | <http://www.w3cplus.com/blog/52.html> | | |
|  | [blur:模糊的半径]  [color:颜色，可以出现在前面或者后面][color x y b][x y b color]  [IE不支持text-shadow,使用滤镜filter:shadow来进行处理]  [IE滤镜 E {filter:shadow(Color=sth,Direction=num,Strength=num)}]  [其中E是元素选择器，Color用于设定对象的阴影色；Direction用于设定投影的主向，取值为0即零度（表示向上方向），45为右上，90为右，135为右下，180为下方，225为左下方，270为左方，315为左上方；Strength就是强度，类似于text-shadow中的blur值。] | | |
| demo |  |  | |
| color:white;  text-shadow:2px 2px 4px #000000; |  | |
| color:black;  text-shadow:2px 2px 8px #FF0000; |  | |
| color:black;  text-shadow:0 0 3px #FF0000; |  | |
| color:black;  text-shadow: 15px 15px 0 red, 27px 27px 0 blue; |  | |
| text-shadow: 5px 5px 0 #666, 7px 7px 0 #eee; |  | |
|  | <style>  .text-shadow {  text-shadow: 0px 1px 0px rgb(204, 204, 204), 0px 2px 0px rgb(201, 201, 201), 0px 3px 0px rgb(187, 187, 187), 0px 4px 0px rgb(185, 185, 185), 0px 5px 0px rgb(170, 170, 170), 0px 6px 1px rgba(0, 0, 0, 0.1), 0px 0px 5px rgba(0, 0, 0, 0.1), 0px 1px 3px rgba(0, 0, 0, 0.3), 0px 3px 5px rgba(0, 0, 0, 0.2), 0px 5px 10px rgba(0, 0, 0, 0.25), 0px 20px 20px rgba(0, 0, 0, 0.15);  color: #FFFFFF;  background: black;  line-height: 1.5;  font-size: 30px;  /\*font-family: 'League Gothic', Impact, sans-serif;\*/  font-family: '微软雅黑'  }  </style>  <div class="text-shadow">  test测试  </div> | | |

--transfrom

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 理解CSS3 transform中的Matrix(矩阵)  <http://www.zhangxinxu.com/wordpress/2012/06/css3-transform-matrix-%E7%9F%A9%E9%98%B5/>  <http://www.w3cplus.com/content/css3-transform> | | | | |
| 基础 | 偏移 | translate(50px,100px); translateX(x); translateY(Y); | | | |
| 缩放 | scale(2,1.5); scaleX(2); scaleY(2); | | | |
| 旋转 | rotate(30deg); rotateX(30deg); rotateY(30deg); | | | |
| 扭曲 - 水平旋转 | skew(30deg,20deg); | | | |
| 矩阵 | matrix() | | | |
| 矩阵matrix | CSS3 transform的matrix()方法写法：  transform: matrix(a,b,c,d,e,f);  具体可转换为：  a c e  b d f  0 0 1  坐标转换公式：    其中，x, y表示转换元素的所有坐标（变量）；ax+cy+e为变换后的水平坐标，bx+dy+f表示变换后的垂直位置。 | | | | |
| 位移 | transform: matrix(1, 0, 0, 1, 30, 30);就等同于transform: translate(30px, 30px);  注意：translate, rotate等方法都是需要单位的，而matrix方法e, f参数的单位可以省略。  根据这个矩阵偏移元素的中心点，假设是(0, 0)，即x=0, y=0。  于是，变换后的x坐标就是ax+cy+e = 1\*0+0\*0+30 =30, y坐标就是bx+dy+f = 0\*0+1\*0+30 =30.  于是，中心点坐标从(0, 0)变成了→(30, 30)：往右下方同时偏移了30像素。 | | | | |
| 缩放 | matrix(1, 0, 0, 1, 30, 30);的元素比例与原来一样，1:1, 而这几个参数中，有两个1就是缩放相关的参数。  其中，第一个缩放x轴，第二个缩放y轴。  用公式就很明白了，假设比例是s，则有matrix(s, 0, 0, s, 0, 0);，于是，套用公式，就有： x' = ax+cy+e = s\*x+0\*y+0 = s\*x; y' = bx+dy+f = 0\*x+s\*y+0 = s\*y;  也就是matrix(sx, 0, 0, sy, 0, 0);，等同于scale(sx, sy); | | | | |
| 旋转 | 方法以及参数使用如下（假设角度为θ）：  matrix(cosθ,sinθ,-sinθ,cosθ,0,0)  结合矩阵公式，就有：  x' = x\*cosθ-y\*sinθ+0 = x\*cosθ-y\*sinθ  y' = x\*sinθ+y\*cosθ+0 = x\*sinθ+y\*cosθ  这个与[IEMatrix滤镜中的旋转](http://www.zhangxinxu.com/wordpress/?p=1505)是有些类似的(M11表示矩阵第1行第1个（参数a），M21表示矩阵第2行第一个（参数b）……)：  filter:progid:DXImageTransform.Microsoft.Matrix(M11=cosθ,M21=sinθ,M12=-sinθ,M22=cosθ'); | | | | |
| 拉伸(skew) | 拉伸也用到了三角函数，不过是tanθ，而且，其至于b, c两个参数相关，书写如下（注意y轴倾斜角度在前）：  matrix(1,tan(θy),tan(θx),1,0,0)  套用矩阵公式计算结果为：  x' = x+y\*tan(θx)+0 = x+y\*tan(θx)  y' = x\*tan(θy)+y+0 = x\*tan(θy)+y  对应于skew(θx + "deg"，θy+ "deg")这种写法。  其中，θx表示x轴倾斜的角度，θy表示y轴，两者并无关联。 | | | | |
|  | 既然有简单的skew, rotate..，那matrix有何用？  我想有人会奇怪，既然CSS3 transform中提供了像skew, rotate, …效果，那还需要掌握和熟悉让人头大的矩阵方法干嘛呢？    好问题，确实，对于一般地交互应用，transform属性默认提供的些方法是足够了，但是，一些其他的效果，如果transform属性没有提供接口方法，那你又该怎么办呢？比方说，“镜像对称效果”！  没辙了吧，这是，就只能靠matrix矩阵了。要知道，matrix矩阵是transform变换的基础，可以应付很多高端的效果，算是一种高级应用技巧吧。掌握了基础，才能兵来将挡水来土掩啊。  OK，这里就演示下，如何使用CSS3 transform matrix矩阵实现镜像效果。  这个有点难度，因此，我们先看demo，您可以狠狠地点击这里：[matrix与镜像对称效果demo](http://www.zhangxinxu.com/study/201206/css3-transform-matrix-mirror.html" \t "http://www.zhangxinxu.com/wordpress/2012/06/css3-transform-matrix-%E7%9F%A9%E9%98%B5/_blank)  框框中输入旋转的角度值（用来确定镜像的对称轴），然后失去焦点，就会呈现出对应的镜像对称效果了：  镜像对称效果操作示意 镜像对称效果截图 镜像渐变题目  您可以在FireFox或是Chrome等浏览器上体验下matrix实现的镜像渐变效果。  demo页面中的一个轴是为了便于理解我加上的效果，实际上，在镜像对称的时候轴是看不见的。  轴围绕的那个点就是CSS3中transform变换的中心点，自然，镜像对称也不例外。因为该轴永远经过原点，因此，任意对称轴都可以用y = k \* x表示。则matrix表示就是：  matrix((1-k\*k) / (1+k\*k), 2k / (1 + k\*k), 2k / (1 + k\*k), (k\*k - 1) / (1+k\*k), 0, 0)  这个如何得到的呢？ 啊，高中数学来了，就当再高考一次吧，如下图，已经y=kx，并且知道点(x, y)坐标，求其对称点(x’, y’)的坐标？  很简单，一是垂直，二是中心点在轴线上，因此有：  (y-y') / (x - x') = -1/ k → ky-ky' = -x+x'  (x + x') / 2 \* k = (y + y')/2 → kx+kx' = y+y'  很简单的，把x'和y'提出来，就有：  x' = (1-k\*k)/(k\*k+1) \*x + 2k/(k\*k+1) \*y;  y' = 2k/(k\*k+1) \*x + (k\*k-1)/(k\*k+1) \*y;  再结合矩阵公式：  x' = ax+cy+e;  y' = bx+dy+f;  我们就可以得到：  a = (1-k\*k)/(k\*k+1);  b = 2k/(k\*k+1);  c = 2k/(k\*k+1);  d = (k\*k-1)/(k\*k+1);  也就是上面matrix方法中的参数值啦！ | | | | |
| transform-origin | transform进行的rotate,translate,scale,skew,matrix等操作都是以元素自己中心位置进行变化的。但有时候我们需要在不同的位置对元素进行这些操作，那么我们就可以使用transform-origin来对元素进行基点位置改变，使元素基点不在是中心位置，以达到你需要的基点位置。  transform-origin:(X,Y);  [x:left center right][y:top center bottom] | | | | |
| transform-origin:(left,top) | | transform-origin:right | transform-origin(25%,75%) | |
| IMG_256 | | IMG_257 | IMG_258 | |
| 例子 | <**style**>  #edgeChange{  width: 50px;  height: 50px;  **/\*transform:translate(50px,100px);\*/** box-shadow:0 0 50px 50px #999;  border-radius:8px;  } </**style**> <**div** id="edgeChange"></**div**> <**script**>  **var num** = 0;  setInterval(**function** () {  $("#edgeChange").css({transform: "rotate(" + **num** + "deg)", "z-index": -1});  **num**++;  }, 100); </**script**> | | | | js控制旋转： |
| <**style**>  .inDiv {  position: relative;  height: 100px;  width: 100px;  border: 1px solid grey;  background-color: lightgrey;  transform: rotateX(180deg);  -webkit-transform: rotateX(180deg); **/\*Chrome Safari\*/** -moz-transform: rotateX(180deg); **/\* Firefox \*/** }   #div1 {  -webkit-backface-visibility: hidden;  -moz-backface-visibility: hidden;  -ms-backface-visibility: hidden;  }   #div2 {  -webkit-backface-visibility: visible;  -moz-backface-visibility: visible;  -ms-backface-visibility: visible;  } </**style**> <**div** style="outline:1px solid red;">  <**div** id="div1" class="inDiv">旋转180度，背面不可见</**div**>  <**div** id="div2" class="inDiv">旋转180度，背面仍可见</**div**> </**div**> | | | | 旋转的元素背面不可见 |
| 具体测试matrix | <**style**>  .matrix\_box {  width: 150px;  height: 150px;  background-color: #a0b3d6;  -moz-transform-origin: top left;  -webkit-transform-origin: top left;  -o-transform-origin: top left;  transform-origin: top left;} </**style**>  <**p** id="matrixDetail">目前属性值为：matrix(1,0,0,1,0,0)</**p**> <**p**>x轴比例(1-3)：<**input** type="range" id="matrixRangeX" value="1" min="1" max="3" step="0.1" autocomplete="off"/></**p**> <**p**>y轴比例(1-3)：<**input** type="range" id="matrixRangeY" value="1" min="1" max="3" step="0.1" autocomplete="off"/></**p**> <**div** id="matrixBox" class="matrix\_box"></**div**>  <**script**>  (**function** () {  **var $** = **function** (selector) {  **return** document.querySelector(selector);  };  **var $css3Transform** = **function** (element, value) {  **var** arrPriex = ["O", "Ms", "Moz", "Webkit", ""], length = arrPriex.length;  **for** (**var** i = 0; i < length; i += 1) {  element.style[arrPriex[i] + "Transform"] = value;  }  };  **var** eleRangeX = **$**("#matrixRangeX"), eleRangeY = **$**("#matrixRangeY"),  eleDetail = **$**("#matrixDetail"), eleBox = **$**("#matrixBox");   **if** (eleDetail && eleRangeX && eleRangeY && eleBox) {  **var** arrRange = [eleRangeX, eleRangeY];  **for** (**var** start = 0; start < 2; start += 1) {  arrRange[start].addEventListener("change", **function** () {  **var** maxVal = **this**.getAttribute("max"), minVal = **this**.getAttribute("min"), value = **this**.value;  **if** (value < minVal) {  value = minVal;  **this**.value = minVal;  }  **if** (value > maxVal) {  value = maxVal;  **this**.value = maxVal;  }  **var** valTransform = 'matrix(' + eleRangeX.value + ',0,0,' + eleRangeY.value + ',0,0)'  eleDetail.innerHTML = '目前属性值为：' + valTransform;  **$css3Transform**(eleBox, valTransform);  });  }  }  })(); </**script**> | | | | |

--transition

|  |  |
| --- | --- |
|  | transition: property duration timing-function delay;    [transition:针对的属性 过渡的持续时间 过渡方式 延迟]  transition-property :\* //指定过渡的性质，比如transition-property:backgrond 就是指backgound参与这个过渡  transition-duration:\*//指定这个过渡的持续时间  transition-delay:\* //延迟过渡时间  transition-timing-function:\*//指定过渡类型，有ease | linear | ease-in | ease-out | ease-in-out | cubic-bezier |
| 被控制属性 | transition-property是用来指定当元素其中一个属性改变时执行transition效果，其主要有以下几个值：  none(没有属性改变)；  all（所有属性改变）这个也是其默认值；  indent（元素属性名）。  当其值为none时，transition马上停止执行；  当指定为all时，则元素产生任何属性值变化时都将执行transition效果；  ident是可以指定元素的某一个属性值。其对应的类型如下：  1、color: 通过红、绿、蓝和透明度组件变换（每个数值处理）如：background-color,border-color,color,outline-color等css属性；  2、length: 真实的数字 如：word-spacing,width,vertical-align,top,right,bottom,left,padding,outline-width,margin,min-width,min-height,max-width,max-height,line-height,height,border-width,border-spacing,background-position等属性；  3、percentage:真实的数字 如：word-spacing,width,vertical-align,top,right,bottom,left,min-width,min-height,max-width,max-height,line-height,height,background-position等属性；  4、integer离散步骤（整个数字），在真实的数字空间，以及使用floor()转换为整数时发生 如：outline-offset,z-index等属性；  5、number真实的（浮点型）数值，如：zoom,opacity,font-weight,等属性；  6、transform list:详情请参阅：《[CSS3 Transform](http://www.w3cplus.com/content/css3-transform" \t "http://www.w3cplus.com/content/_blank)》  7、rectangle:通过x, y, width 和 height（转为数值）变换，如：crop  8、visibility: 离散步骤，在0到1数字范围之内，0表示“隐藏”，1表示完全“显示”,如：visibility  9、shadow: 作用于color, x, y 和 blur（模糊）属性,如：text-shadow  10、gradient: 通过每次停止时的位置和颜色进行变化。它们必须有相同的类型（放射状的或是线性的）和相同的停止数值以便执行动画,如：background-image  11、paint server (SVG): 只支持下面的情况：从gradient到gradient以及color到color，然后工作与上面类似  12、space-separated list of above:如果列表有相同的项目数值，则列表每一项按照上面的规则进行变化，否则无变化  13、a shorthand property: 如果缩写的所有部分都可以实现动画，则会像所有单个属性变化一样变化  具体什么css属性可以实现transition效果，在W3C官网中列出了所有可以实现transition效果的CSS属性值以及值的类型，大家可以点[这里](http://www.w3.org/TR/css3-transitions/" \l "properties-from-css-" \t "http://www.w3cplus.com/content/_blank)了解详情。这里需要提醒一点是，并不是什么属性改变都为触发transition动作效果，比如页面的自适应宽度，当浏览器改变宽度时，并不会触发transition的效果。但上述表格所示的属性类型改变都会触发一个transition动作效果。 |
| 时间 | transition-duration是用来指定元素 转换过程的持续时间，取值：<time>为数值，单位为s（秒）或者ms(毫秒),可以作用于所有元素，包括:before和:after伪元素。其默认值是0，也就是变换时是即时的。 |
| 方式 | transition-timing-function的值允许你根据时间的推进去改变属性值的变换速率，transition-timing-function有6个可能值：  1、ease：（逐渐变慢）默认值，ease函数等同于贝塞尔曲线(0.25, 0.1, 0.25, 1.0).  2、linear：（匀速），linear 函数等同于贝塞尔曲线(0.0, 0.0, 1.0, 1.0).  3、ease-in：(加速)，ease-in 函数等同于贝塞尔曲线(0.42, 0, 1.0, 1.0).  4、ease-out：（减速），ease-out 函数等同于贝塞尔曲线(0, 0, 0.58, 1.0).  5、ease-in-out：（加速然后减速），ease-in-out 函数等同于贝塞尔曲线(0.42, 0, 0.58, 1.0)  6、cubic-bezier：（该值允许你去自定义一个时间曲线）， 特定的[cubic-bezier曲线](http://en.wikipedia.org/wiki/B%C3%A9zier_curve" \t "http://www.w3cplus.com/content/_blank)。 (x1, y1, x2, y2)四个值特定于曲线上点P1和点P2。所有值需在[0, 1]区域内，否则无效。其是cubic-bezier为通过贝赛尔曲线来计算“转换”过程中的属性值，如下曲线所示，通过改变P1(x1, y1)和P2(x2, y2)的坐标可以改变整个过程的Output Percentage。初始默认值为default. |
| 延迟 | transition-delay是用来指定一个动画开始执行的时间，也就是说当改变元素属性值后多长时间开始执行transition效果，其取值：<time>为数值，单位为s（秒）或者ms(毫秒)，其使用和transition-duration极其相似，也可以作用于所有元素，包括:before和:after伪元素。  默认大小是"0"，也就是变换立即执行，没有延迟。 |
| 例子 | -webkit-transition: all 0.5s ease-in;  -webkit-transition: background 0.5s ease-in,color 0.3s ease-out;  -webkit-transition: all .5s ease-in-out 1s; |

### --animation

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | IMG_256  --transition:pro time timeFuntion delayTime;  --animation:name time timeFuncion delayTime num fill-mode direction;   |  |  |  |  | | --- | --- | --- | --- | | name | duration-time | time-function | delay-time | |  |  | linear  ease  ease-in  ease-out  ease-in-out  cubic-bezier(0-1,0-1,0-1,0-1) |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | | number | direction | fill-mode |  | | n  infinite | normal  reverse  alternate | none  backwards  forwards  both |  | |  | | | | | num: n | infinite  infinite:规定动画应该无限次播放 | | | | | direction: normal | reverse | alternate  normal  reverse:方向相反  alternate:动画会在奇数次数（1、3、5 等等）正常播放，而在偶数次数（2、4、6 等等）向后播放。 | | | | | fill-mode：none | backwards | forwards | both  规定对象动画时间之外的状态。  "none"：动画结束后自动回到原点  "backwards":开始执行是立即跳到0%处  "forwards":执行结束后，跳到100%处  "both":是forwards和backwards的结合。动画可以在开始前就已经是第一个关键帧的样式，然后，在动画完成后，保持最后一个关键帧的样式。  <https://segmentfault.com/q/1010000003867335> | | | | |
|  |

animation 与transition的简单对比

|  |  |
| --- | --- |
| animation 与 transition 的启动方式 | animation自动运行 与 transition的获焦启动（都具有运行过程） |
| <**style**>  .testTransition{  height: 100px;  width: 100px;**/\*0px\*/** background: limegreen;  **-webkit-transition: width 1s;**  }  .testTransition:hover{  width: 200px;**/\*0px\*/** } </**style**> <**div** class="testTransition">  2 </**div**> |
| <**style**>  .testTransition{  height: 100px;  width: 100px;**/\*0px\*/** background: limegreen;  -webkit-animation: donut-show-one 1s linear 1s both;  }   **@-webkit-keyframes** donut-show-one {  to {  width: 200px;  }  } </**style**> <**div** class="testTransition">  2 </**div**> |
| transition-滚动 | <**style**>  .inner {  height: 100px;  width: 100px; **/\*0px\*/** background: limegreen;  -webkit-transition: all 1s ease-in-out;   text-align: center;  line-height: 100px  }  .outer:hover .inner {  transform: rotate(180deg);  margin: 0 0 0 314px;  border-radius: 50%  } </**style**> <**div** class="outer">  <**div** class="inner">  </**div**> </**div**> |
| animation-运动 | <**style**>  .inner {  height: 100px;  width: 100px; **/\*0px\*/** background: limegreen;   text-align: center;  line-height: 100px;  border: 10px solid lightgreen;  }  .outer:hover .inner {  animation: test-animation 3s 4 alternate linear;  }  **@keyframes** test-animation {  0% {  transform: rotate(0deg);  border-radius: 0 0 0 0;  margin-left: 0;  }  25% {  transform: rotate(90deg);  border-radius: 0 0 50% 0;  margin-left: 100px;  }  50% {  transform: rotate(180deg);  border-radius: 0 50% 50% 0;  margin-left: 200px;  }  75% {  transform: rotate(270deg);  border-radius: 50% 50% 50% 0;  margin-left: 300px;  }  100% {  transform: rotate(360deg);  border-radius: 50% 50% 50% 50%;  margin-left: 400px;  }  } </**style**> <**div** class="outer">  <**div** class="inner">  </**div**> </**div**> |
| animation-颜色渐变效果 | <**style**>  #test>**div**{  position: absolute;  width: 100px;  height: 100px;   line-height: 100px;  text-align: center;  }  .animation{  animation: test-animation 3s linear infinite;  }  **@keyframes** test-animation {  0%{  opacity: 0;  }  50%{  opacity: 1;    }  100%{  opacity: 0;  }    } </**style**> <**div** id="test">  <**div** class="" style="background: blue;">1</**div**>  <**div** class="animation" style="background: red;">2</**div**> </**div**> |

3d prespective transform-style

|  |  |
| --- | --- |
| CSS3 perspective 属性 | <**style**>  .outer {  width: 200px;  height: 200px;  border: 2px solid #000;  margin: 100px auto;  **/\*perspective: 800px;\*/** }  .inner {  width: 100%;  height: 100%;  background: lime;  font-size: 20px;  text-align: center;  -webkit-transition: transform 1s;  }  .outer:hover .inner {  -webkit-transform: perspective(800px) rotateY(90deg) translateZ(300px);  **/\*-webkit-transform: translateZ(300px) rotateY(55deg);\*/  /\*-webkit-transform: rotateY(55deg);\*/** } </**style**> <**div** class="outer">  <**div** class="inner son-hover">rotateY</**div**> </**div**> |
|  |  |
|  | CSS3 3D transform  <http://www.zhangxinxu.com/wordpress/2012/09/css3-3d-transform-perspective-animate-transition/> |

--grandient

|  |  |  |
| --- | --- | --- |
|  | <http://www.w3cplus.com/content/css3-gradient> | |
| 说明 | [CSS3 Gradient](http://dev.w3.org/csswg/css3-images/" \t "http://www.w3cplus.com/content/_blank)分为[linear-gradient](http://lists.w3.org/Archives/Public/www-style/2011Apr/0083.html" \t "http://www.w3cplus.com/content/_blank)(线性渐变)和[radial-gradient](http://lists.w3.org/Archives/Public/www-style/2011Feb/0537.html" \t "http://www.w3cplus.com/content/_blank)(径向渐变)。  目前的几种现代浏览器的内核，主流内容主要有[Mozilla](http://www.mozilla.org/" \t "http://www.w3cplus.com/content/_blank)（Gecko）（熟悉的有Firefox，Flock等浏览器）、[WebKit](http://www.webkit.org/" \t "http://www.w3cplus.com/content/_blank)（熟悉的有Safari、Chrome等浏览器）、[Opera](http://www.opera.com/" \t "http://www.w3cplus.com/content/_blank)(presto)（Opera浏览器）、Trident（讨厌的IE浏览器）。 | |
| 写法 |  | |
| 线性渐变 | ---渐变写法1：渐变颜色的方向默认为top，可以改写为left、--deg  background: -webkit-linear-gradient(#ace, #f96);  background: -webkit-linear-gradient(left,red,blue);  background: -webkit-linear-gradient(30deg, #ace, #f96);  background: -webkit-linear-gradient(left, #ace, #f96, #ace, #f96, #ace);  background: -webkit-linear-gradient(left, #ace, #f96 5%, #ace, #f96 95%, #ace);  background: -webkit-linear-gradient(left top, #ace, #f96 5%, #ace, #f96 95%, #ace);  ---渐变写法2：  background: -webkit-gradient(linear,center top,center bottom,from(blue),to(red));  background: -webkit-gradient(linear,left top,right bottom,from(blue),to(red),color-stop(50%,yellow));  background: -webkit-gradient(linear,left center,right center,from(blue),to(red),color-stop(50%,yellow));  background: -webkit-gradient(linear,0% 30%,right center,from(blue),to(red),color-stop(50%,yellow),color-stop(70%,green));  background: -webkit-gradient(linear,0 50%,100% 50%,from(#ace),to(#f96)); | |
| 径向渐变 | ---渐变写法1：  background: -webkit-radial-gradient(#ace, #f96, #1E90FF);  background: -webkit-radial-gradient(#ace 5%, #f96 25%, #1E90FF 50%);  background: -webkit-radial-gradient(bottom left, circle, #ace, #f96, #1E90FF);  background: -webkit-radial-gradient(bottom left, ellipse, #ace, #f96, #1E90FF);  background: -webkit-radial-gradient(ellipse closest-side, #ace, #f96 10%, #1E90FF 50%, #f96);  background: -webkit-radial-gradient(ellipse farthest-corner, #ace, #f96 10%, #1E90FF 50%, #f96);  background: -webkit-radial-gradient(circle closest-side, #ace, #f96 10%, #1E90FF 50%, #f96);  ---渐变写法2：从 一个圆(中心点坐标+半径)( x y , r) 到 另一个圆(中心点坐标+半径)( x y , r) 的颜色变化方式  background: -webkit-gradient(radial,100 50,10,100 50,100,from(red),to(blue),color-stop(0.5,yellow)); | |
|  | ---IE兼容写法  filter: progid:DXImageTransform.Microsoft.gradient(GradientType=0, startColorstr=#1471da, endColorstr=#1C85FB);/\*IE<9>\*/  -ms-filter: "progid:DXImageTransform.Microsoft.gradient (GradientType=0, startColorstr=#1471da, endColorstr=#1C85FB)";/\*IE8+\*/ | |
|  | /图片模糊化/  background: -webkit-linear-gradient(right,rgba(255,255,255,0),rgba(255,255,255,1)),url(flowers-pattern.jpg); | |
|  | /画小格子/  background: #ace; -webkit-background-size: 30px 30px; background-image: -webkit-linear-gradient(rgba(255, 255, 255, .5) 25%, transparent 50%, rgba(255, 255, 255, .9) 75%);    <style type="text/css">  .background {  background: #ace;  background-size: 30px 30px;  background-image: -webkit-linear-gradient(rgba(255, 255, 255, .5) 50%, rgba(255, 255, 255, 0.1) 50%);  width: 100px;  height: 1000px;  }  </style>  <div class="background">ok</div> | |
| background: -moz-repeating-radial-gradient(#ace, #ace 5px, #f96 5px, #f96 10px);  background: -webkit-repeating-radial-gradient(#ace, #ace 5px, #f96 5px, #f96 10px); |  |
| background: -moz-repeating-linear-gradient(top left -45deg, #ace, #ace 5px, #f96 5px, #f96 10px);  background: -webkit-repeating-linear-gradient(top left -45deg, #ace, #ace 5px, #f96 5px, #f96 10px); |  |
|  | <style type="text/css">  .samples {  font-family: 'League Gothic', Impact, sans-serif;  letter-spacing: 0.02em;  text-transform: uppercase;  text-align: center;  font-size: 2em;  width: 100%;  height: 300px;  line-height: 285px;  float: left;  margin: 1px;  }  .background-1 {  background: #282537;  background-image: -webkit-radial-gradient(top, circle cover, #3c3b52 0%, #252233 80%);  background-image: -moz-radial-gradient(top, circle cover, #3c3b52 0%, #252233 80%);  background-image: -o-radial-gradient(top, circle cover, #3c3b52 0%, #252233 80%);  background-image: radial-gradient(top, circle cover, #3c3b52 0%, #252233 80%);  }  </style>  <div class="background-1 samples">ok</div> | |
|  | <style type="text/css">  .background {  background: #ace;  background-size: 30px 30px;  background: -webkit-gradient(radial,20 20,89,70 70,1,from(gray),to(red),color-stop(50%,yellow),color-stop(90%,green));  width: 100px;  height: 100px;  border-radius: 50%;  }  </style>  <div class="background"></div> | |
|  | <style type="text/css">  .background {  background: #ace;  background-size: 30px 30px;  background: -webkit-gradient(radial,20 20,89,70 70,1,from(gray),to(yellow),color-stop(50%,yellow));  width: 100px;  height: 100px;  border-radius: 50%;  line-height: 90px;  text-align: center;  text-shadow: 2px 2px 0 #666, 3px 3px 0 #eee;  font-size: 40px;  font-weight: bolder;  color: #eee;  font-family: '微软雅黑'  }  </style>  <div class="background">帅</div> | |

### 关于文本样式

|  |  |
| --- | --- |
|  | <http://sojuker.blog.163.com/blog/static/1387908792012760243916/>  <http://www.divcss5.com/wenji/w738.shtml>  <http://www.jb51.net/css/20608.html>  <http://zhidao.baidu.com/link?url=WMco4j5Bgl2Jzkjjut3Mb_a5CsoMr3FSGxtKgQuwcQFMP8tnHUTCV2WKHrs9N8ZRvL8WDKyvy8ct-qGGWtekNdLsJcL8gs7qWL8c4zbiYVO>  <http://www.zhangxinxu.com/wordpress/2010/06/%E5%8F%AF%E7%94%A8%E6%80%A7%E4%B9%8B%E6%B5%8F%E8%A7%88%E5%99%A8%E9%BB%98%E8%AE%A4%E5%AD%97%E4%BD%93%E4%B8%8Ecss%E5%AD%97%E4%BD%93/> |
| 水平对齐方式 | text-align:left; |
| 首行缩进 | text-indent:50px;[文本第一行缩进] |
| 字体的样式 | text-transform: uppercase; /\*大写\*/ letter-spacing: 0.2em; /\*字间距\*/ font-weight: 900; /\*字体加粗\*/ color: #ce4233; /\*字体的颜色\*/ |
| 字间距 | letter-sapcing:50px;  ---单个字母与单个字母、单个汉字与单个汉字之间的间距 |
| 词间距 | word-spacing:50px;  ---单词与单词间的间距 |
| 定义鼠标选中后的样式 | 可以用下面几行代码改变被选中文本的颜色：  \*::selection {  color: #fff;  background: #000;  }  \*::-moz-selection {  /\*Only Firefox still needs a prefix\*/  color: #fff;  background: #000;  }  除了定义 color，还能定义 shadow 和 backgrounds。 |
| 换行的方式 | <**style** type="text/css">  .box {  width: 100px;  height: 100px;  border: 2px solid #f00;  color: #333;  font: 12px arial;  margin: 10px;  overflow: hidden;**/\*---\*/** display: inline-block;  }   **/\*空格、中文标点符 换行\*/** .keepall {  word-break: keep-all;  }   **/\*自动换行\*/** .breakall {  word-break: break-all;  }   **/\*单个长英语单词换行方式较特别，强行把连续的长文本在达到父容器边界的位置掰断\*/** .breakword {  word-wrap: break-word;  } </**style**> **<!--不常用-->** <**div** class="box keepall">  CSS3的一个非常酷的特性是允许创建各，AAAAAAAAAA BBBBBBBBBBBBB,CCCCCCCCC，的右端自动换行。 </**div**> **<!--此例中注意keep-all在不同浏览器中的兼容，特殊符号处的换行！-->** <**div** class="box breakall">  CSS3的一个非常酷的特性是允许创建各，AAAAAAAAAA BBBBBBBBBBBBB,CCCCCCCCC，的右端自动换行。 </**div**> **<!--此例中注意逗号的位置!-->** <**div** class="box breakword">  CSS3的一个非常酷的特性是允许创建各，AAAAAAAAAA BBBBBBBBBBBBB,CCCCCCCCC，的右端自动换行。 </**div**> |
| word-wrap:break-word;  内容将在边界内换行，仅用于块对象，内联对象要用的话，必须要设定height、width或display:block或position:absolute。  word-break:break-all;  word-break:keep-all;  用于处理单词折断。(注意与第一个属性的对比) |
| 超出文本... | overflow:hidden; [超出部分隐藏]  text-overflow:esllipsis; [超出部分显示省略号...] [clip直接进行简单的截取]  white-space:nowrap; [保证不换行，只在一行内显示] |
| 多行超出 | <http://www.zhangxinxu.com/wordpress/2009/09/%E5%85%B3%E4%BA%8E%E6%96%87%E5%AD%97%E5%86%85%E5%AE%B9%E6%BA%A2%E5%87%BA%E7%94%A8%E7%82%B9%E7%82%B9%E7%82%B9-%E7%9C%81%E7%95%A5%E5%8F%B7%E8%A1%A8%E7%A4%BA/> |
| 对于现代浏览器，例如webkit内核的浏览器，或者移动端，是可以实现多行文本内容超出点点点…最后一行显示的，典型的CSS组合如下：  .box {  display: -webkit-box;  -webkit-line-clamp: 3;  -webkit-box-orient: vertical;  overflow: hidden;  }  其中-webkit-line-clamp就是控制行数的，是3就是显示3行，3行结束点点点，如果是2则最多2行。 |
| <**style**>  .zxx\_text\_overflow {  width: 24em;  height: 1.13em; **/\* 18px/16px \*/** overflow: hidden;  zoom: 1;  }   .zxx\_text\_overflow .text\_con {  float: left;  height: 1.13em;  margin-right: 3em;  overflow: hidden;  }   .zxx\_text\_overflow .text\_dotted {  width: 3em;  height: 1.13em;  float: right;  margin-top: -1.3em;  } </**style**> <**div** class="zxx\_text\_overflow" >  <**div** class="text\_con" >这是一段比较长的文字，用来测试是否文字溢出时会用省略号显示。</**div**>  <**div** class="text\_dotted" >…</**div**> </**div**> |
| --font-smoothing | 这个属性可以使页面上的字体抗锯齿,使用后字体看起来会更清晰舒服,特别是图标文字流行的今天。  它有三个属性：  none                                     ------ 对低像素的文本比较好  subpixel-antialiased                ------默认值  antialiased                             ------抗锯齿很好  例子：  <**style**>  **body** {  -webkit-font-smoothing: antialiased;  -moz-osx-font-smoothing: grayscale; **/\* [inherit | grayscale]\*/** } </**style**> |
| 不换行显示 | 当一行文字超过DIV或者Table的宽度的时候，浏览器中默认是让它换行显示的；用CSS让文字在一行内显示不换行：  一般的文字截断(适用于内联与块)：  <style>  .text-overflow{  outline: 1px solid red;  display:block; /\*内联对象需加\*/  white-space:nowrap; /\* 不换行 \*/  overflow:hidden; /\* 内容超出宽度时隐藏超出部分的内容 \*/  } </style> |
| 对于表格，定义有点不一样： <**style**>  **table** {  width: 30em;  table-layout: fixed; **/\* 只有定义了表格的布局算法为fixed，下面td的定义才能起作用。 \*/** }   **td** {  width: 100%;  word-break: keep-all; **/\* 不换行 \*/** white-space: nowrap; **/\* 不换行 \*/** overflow: hidden; **/\* 内容超出宽度时隐藏超出部分的内容 \*/** text-overflow: ellipsis; **/\* 当对象内文本溢出时显示省略标记(...) ；需与overflow:hidden;一起使用。\*/** } </**style**> 注：这个只对单行的文字的效，如果你想把它用在多行上，也只有第一行有作用的。 这个写法只有IE会有“...”，其它的浏览器文本超出指定宽度时会隐藏。 |
|  |  |

### CSS中的单位[颜色]

|  |  |  |
| --- | --- | --- |
| rgba(-,-,-,-) | background: rgb(255,255,0);  opacity: 0.6;  filter:alpha(opacity=60); | background:rgba(255,255,0,0.6); |
|  | |
| /\* IE5 - 7 \*/  filter: alpha(opacity=80); /\* IE 8 \*/  -ms-filter:"progid:DXImageTransform.Microsoft.Alpha(Opacity=80)"; /\* Everyone else \*/  opacity: 0.8; | |
| -ms-filter: "progid:DXImageTransform.Microsoft.gradient(GradientType=1,startColorstr=#80000000,endColorstr=#80000000)";  /\*Filter for IE8 \*/  filter: progid:DXImageTransform.Microsoft.gradient(GradientType=1,startColorstr=#80000000, endColorstr=#80000000);  /\*Filter for older IEs \*/  这里需要注意的是startColorStr和endColorStr的值#80000000,其中前两位是十六进制的透明度80,也就是透明值为0.5而后面六位是十六进制的颜色#000000(black黑色)。如果你和我一样不知道怎么转换这个值，你可以采用下面这个工具：  [CSS背景颜色属性值转换](http://labs.pufen.net/views/hex_change.html" \o "CSS背景颜色属性值转换" \t "http://www.w3cplus.com/content/_blank) 有关于这个工具的使用大家可以点击**[志哥](http://blog.linxz.de/" \t "http://www.w3cplus.com/content/_blank)**的《[使用IE过渡滤镜和CSS3中的RGBA属性完成背景色透明效果](http://blog.linxz.de/ie_filter_css3_rgba/" \o "使用IE过渡滤镜和CSS3中的RGBA属性完成背景色透明效果" \t "http://www.w3cplus.com/content/_blank)》  RGBA颜色与兼容性的半透明背景色：  http://www.zhangxinxu.com/wordpress/2010/05/rgba%E9%A2%9C%E8%89%B2%E4%B8%8E%E5%85%BC%E5%AE%B9%E6%80%A7%E7%9A%84%E5%8D%8A%E9%80%8F%E6%98%8E%E8%83%8C%E6%99%AF%E8%89%B2/ | |
| 应用 | color: rgba(255, 0, 0,0.5);  border:5px solid rgba(255,0,0,0.5);  text-shadow : 0 2px 1px rgba(255,0,0,0.3);  -webkit-box-shadow: 0 2px 2px rgba(255,0,0,0.6);  background: -webkit-linear-gradient(top,rgba(255,255,255,1),rgba(255,255,255,0)),url(tupian.jpg); | |

媒体查询@media

|  |  |
| --- | --- |
| 简介 | 使用@media查询，可以正对不同的媒体类型定义不同的样式  @media mediatype and|not|only (media feature){ } |
| @media only screen and (max-width: 500px) { /\*当屏幕尺寸小于500px时，应用下面的CSS样式\*/     .gridmenu {         width:100%;     } } |
| demo | <**style**>  .basic{  height: 100px;  outline: 1px solid red;  display: inline-block;  }  **@media** only screen and (min-width: 1024px) {  .content{  width: 1024px;  margin: auto;  }  .leftBox {  min-width: 12%;  }  .middleBox {  width: 74%;  }  .rightBox {  min-width: 12%;  }  }  **@media** only screen and (min-width: 600px) and (max-width: 1024px) {  .leftBox {  width: 30%  }  .middleBox {  width: 65%  }  .rightBox {  width: 0;  }  }  **@media** only screen and (max-width: 600px) {  .leftBox, .middleBox, .rightBox {  width: 98%;  height: 200px;  }  } </**style**>  <**div** class="content">  <**div** class="basic leftBox" style="background: limegreen;"></**div**>  <**div** class="basic middleBox" style="background: lightblue;"></**div**>  <**div** class="basic rightBox" style="background: lightpink;"></**div**> </**div**> |

使用CSS制作各种形状的图形

|  |  |  |
| --- | --- | --- |
|  | 使用CSS3制作各种形状的图形  <http://www.htmleaf.com/ziliaoku/qianduanjiaocheng/201504051631.html> | |
| 对 border的应用 | 关键是要找到一个基点，中心点，以中心点位中心向三个方向  中心点是在等腰三角形的上顶点； | |
| <**style**>  .btn {  width: 0;  height: 0;  border-top: 15px solid blue;  border-right: 15px solid red;  border-bottom: 15px solid orange;  border-left: 15px solid yellow;  **/\*border-radius: 100%;\*/  /\*outline: red 1px solid;\*/** } </**style**> <**div** class="btn"></**div**> |  |
| <**style** type="text/css">  .syl{  width: 0;  height: 0;  border-left: 100px solid transparent;  border-right: 100px solid transparent;  border-bottom: 100px solid red;  **/\*border-radius: 100%;\*/  /\*outline: 1px solid red;\*/** } </**style**> <**div** class="syl"></**div**> |  |
| <**style** type="text/css">  .syl{  width: 0;  height: 0;  border-width: 50px ;  border-style: solid;  border-color: red transparent transparent transparent;  border-radius: 100%;  outline: 1px solid red;  } </**style**> <**div** class="syl"></**div**> |  |
| <**style**>  .border {  width: 0;  height: 0;   border: 100px solid gray;  border-top-color: transparent;   border-radius: 100%;  outline: 1px solid red;  } </**style**> <**div** class="border"> | 在一个圆形里面制作一个透明的三角形 |
| 对border-radius的应用 | <**style**>  .border {  height:100px;  width:200px;  background: red;  border-radius:100px/50px;  } </**style**> <**div** class="border"> |  |
| <**style**>  #egg{  width: 136px;  height: 190px;  background: #ffc000;  display: block;  -webkit-border-radius: 63px 63px 63px 63px / 108px 108px 72px 72px;  border-radius: 50% 50% 50% 50% / 60% 60% 40% 40%;  }    </**style**> <**div** id="egg"></**div**> | 蛋形时椭圆形的一个变体，它的高度要比宽度稍大，并且设置正确的border-radius属性即可以制作出一个蛋形。 |
| <**style**>  #infinity {  width: 220px;  height: 100px;  position: relative;  }  #infinity:before,#infinity:after {  content: **""**;  width: 60px;  height: 60px;  position: absolute;  top: 0;  left: 0;  border: 20px solid #06c999;  -moz-border-radius: 50px 50px 0;  border-radius: 50px 50px 0 50px;  -webkit-transform: rotate(-45deg);  -moz-transform: rotate(-45deg);  -ms-transform: rotate(-45deg);  -o-transform: rotate(-45deg);  transform: rotate(-45deg);  }  #infinity:after {  left: auto;  right: 0;  -moz-border-radius: 50px 50px 50px 0;  border-radius: 50px 50px 50px 0;  -webkit-transform: rotate(45deg);  -moz-transform: rotate(45deg);  -ms-transform: rotate(45deg);  -o-transform: rotate(45deg);  transform: rotate(45deg);  }   </**style**> <**div** id="infinity"></**div**> | 通过border属性和设置伪元素的角度来实现 |
| n边形 | <**style**>  #diamond {  width: 120px;  height: 120px;  background: #1eff00;  **/\* Rotate \*/** -webkit-transform: rotate(-45deg);  -moz-transform: rotate(-45deg);  -ms-transform: rotate(-45deg);  -o-transform: rotate(-45deg);  transform: rotate(-45deg);  **/\* Rotate Origin \*/** -webkit-transform-origin: 0 100%;  -moz-transform-origin: 0 100%;  -ms-transform-origin: 0 100%;  -o-transform-origin: 0 100%;  transform-origin: 0 100%;  **/\*Can see\*/** margin: 100px;  } </**style**> <**div** id="diamond"></**div**> |  |
|  | <**style**>  #parallelogram {  width: 160px;  height: 100px;  background: #8734f7;  -webkit-transform: skew(30deg);  -moz-transform: skew(30deg);  -o-transform: skew(30deg);  transform: skew(30deg);  } </**style**> <**div** id="parallelogram"></**div**> |  |
|  | <**style**>  #star{   width: 0;  height: 0;  margin: 50px 0;  color: #fc2e5a;  position: relative;  display: block;  border-right: 100px solid transparent;  border-bottom: 70px solid #fc2e5a;  border-left: 100px solid transparent;  -moz-transform: rotate(35deg);  -webkit-transform: rotate(35deg);  -ms-transform: rotate(35deg);  -o-transform: rotate(35deg);  }  #star:before {  height: 0;  width: 0;  position: absolute;  display: block;  top: -45px;  left: -65px;  border-bottom: 80px solid #fc2e5a;  border-left: 30px solid transparent;  border-right: 30px solid transparent;  content: **''**;  -webkit-transform: rotate(-35deg);  -moz-transform: rotate(-35deg);  -ms-transform: rotate(-35deg);  -o-transform: rotate(-35deg);  }  #star:after{  content: **''**;  width: 0;  height: 0;  position: absolute;  display: block;  top: 3px;  left: -105px;  color: #fc2e5a;  border-right: 100px solid transparent;  border-bottom: 70px solid #fc2e5a;  border-left: 100px solid transparent;  -webkit-transform: rotate(-70deg);  -moz-transform: rotate(-70deg);  -ms-transform: rotate(-70deg);  -o-transform: rotate(-70deg);  } </**style**> <**div** id="star"></**div**> | 三个区域组成如下所示： |
|  | <style>  #star\_six\_points {  width: 0;  height: 0;  display: block;  position: absolute;  border-left: 50px solid transparent;  border-right: 50px solid transparent;  border-bottom: 100px solid #de34f7;  margin: 10px auto;  }  #star\_six\_points:after {  content: "";  width: 0;  height: 0;  position: absolute;  border-left: 50px solid transparent;  border-right: 50px solid transparent;  border-top: 100px solid #de34f7;  margin: 30px 0 0 -50px;  } </style> <div id="star\_six\_points"></div> | 相倒的三角形组成 |
|  | <style>  #pentagon {  width: 54px;  position: relative;  border-width: 50px 18px 0;  border-style: solid;  border-color: #277bab transparent;  }  #pentagon:before {  content: "";  height: 0;  width: 0;  position: absolute;  top: -85px;  left: -18px;  border-width: 0 45px 35px;  border-style: solid;  border-color: transparent transparent #277bab;  }  </style> <div id="pentagon"></div> | 创建CSS五边形需要结合两个图形：一个梯形，然后在它的上面放一个三角形，共同组成一个五边形。 |
|  | <style>  #hexagon {  width: 100px;  height: 55px;  background: #fc5e5e;  position: relative;  margin: 10px auto;  }  #hexagon:before {  content: "";  width: 0;  height: 0;  position: absolute;  top: -25px;  left: 0;  border-left: 50px solid transparent;  border-right: 50px solid transparent;  border-bottom: 25px solid #fc5e5e;  }  #hexagon:after {  content: "";  width: 0;  height: 0;  position: absolute;  bottom: -25px;  left: 0;  border-left: 50px solid transparent;  border-right: 50px solid transparent;  border-top: 25px solid #fc5e5e;  }   </style> <div id="hexagon"></div> | 先制作一个长方形，然后在它的上面和下面各放置一个三角形。 |
|  | <style>  #octagon {  width: 100px;  height: 100px;  background: #ac60ec;  position: relative;  }  #octagon:before {  content: "";  width: 42px;  height: 0;  position: absolute;  top: 0;  left: 0;  border-bottom: 29px solid #ac60ec;  border-left: 29px solid #f4f4f4;  border-right: 29px solid #f4f4f4;  }  #octagon:after {  content: "";  width: 42px;  height: 0;  position: absolute;  bottom: 0;  left: 0;  border-top: 29px solid #ac60ec;  border-left: 29px solid #f4f4f4;  border-right: 29px solid #f4f4f4;  }   </style> <div id="octagon"></div> | 先制作两个相同的梯形，然后在两边分别放置一个三角形。 |
|  | <style>  #heart {  position: relative;  }  #heart:before,#heart:after {  content: "";  width: 70px;  height: 115px;  position: absolute;  background: red;  left: 70px;  top: 0;  -webkit-border-radius: 50px 50px 0 0;  -moz-border-radius: 50px 50px 0 0;  border-radius: 50px 50px 0 0;  -webkit-transform: rotate(-45deg);  -moz-transform: rotate(-45deg);  -ms-transform: rotate(-45deg);  -o-transform: rotate(-45deg);  transform: rotate(-45deg);  -webkit-transform-origin: 0 100%;  -moz-transform-origin: 0 100%;  -ms-transform-origin: 0 100%;  -o-transform-origin: 0 100%;  transform-origin: 0 100%;  }  #heart:after {  left: 0;  -webkit-transform: rotate(45deg);  -moz-transform: rotate(45deg);  -ms-transform: rotate(45deg);  -o-transform: rotate(45deg);  transform: rotate(45deg);  -webkit-transform-origin: 100% 100%;  -moz-transform-origin: 100% 100%;  -ms-transform-origin: 100% 100%;  -o-transform-origin: 100% 100%;  transform-origin: 100% 100%;  }  </style> <div id="heart"></div> | 心形的制作是非常复杂的，可以使用伪元素来制作，分别将伪元素旋转不同的角度，并修改transform-origin属性来元素的旋转中心点。 |
|  | <**style**>  .syl{  position: relative;**/\*将after与before元素限制在自身区域\*/** top: 100px;  left: 100px;  height: 100px;  width: 100px;  background: red;  }  .syl:before{  position: absolute;**/\* 必须 \*/** display: block;  content: **""**;  top: -10px;  left: -10px;  right: 10px;  bottom: 10px;  border: 5px solid blue;   }  .syl:after{  position: absolute; **/\* 必须 \*/** display: block;  content: **""**;  top: 10px;  left: 10px;  right: -10px;  bottom: -10px;  border: 5px solid yellowgreen;  } </**style**> <**div** class="syl"></**div**> | (left top的应用)  (0000居中) |
|  | <**style**>  #demo{  width: 100px;  height: 100px;  border:2px solid #ff0000;  }  #demo:before{  content: **""**;  display: block;  width: 0px;  height: 0px;  position: relative;  **/\*position: absolute; \*/** left: 100px;  top:30px;  border-left: 15px solid #ff0000;  border-top: 15px solid transparent;  border-bottom: 15px solid transparent;  }  #demo:after{  content: **""**;  display: block;  width: 0px;  height: 0px;  position: relative;  **/\*position: absolute; \*/** left: 100px;  top:2px;  border-left: 13px solid #ffffff;  border-top: 13px solid transparent;  border-bottom: 13px solid transparent;  } </**style**> <**div** id="demo"></**div**> |  |
|  | <style>  #comment\_bubble {  width: 140px;  height: 100px;  background: #088cb7;  position: relative;  -moz-border-radius: 12px;  -webkit-border-radius: 12px;  border-radius: 12px;  }  #comment\_bubble:before {  content: "";  width: 0;  height: 0;  right: 100%;  top: 38px;  position: absolute;  border-top: 13px solid transparent;  border-right: 26px solid #088cb7;  border-bottom: 13px solid transparent;  }  </style> <div id="comment\_bubble"></div> | 先制作一个圆角矩形，然后在需要的地方放置一个三角形 |

img的srcset、sizes属性, w描述符

|  |  |
| --- | --- |
| 关于img的srcset | 如下HTML示例：  <img class="image" src="mm-width-128px.jpg"  srcset="mm-width-128px.jpg 128w, mm-width-256px.jpg 256w, mm-width-512px.jpg 512w"  sizes="(max-width: 360px) 340px, 128px">  其中1： srcset用来指向提供的图片资源，注意，仅仅是资源指向，没有以前的1x, 2x什么的，这个都交给浏览器了，我们不需要关心！例如这里，指向了3个尺寸图片，分别实际尺寸128像素，256像素和512像素。  其中2： sizes用来表示尺寸临界点，主要跟响应式布局打交道。语法如下：  sizes="[media query] [length], [media query] [length] ... etc"  例如上述代码中，size = "(max-width: 360px) 340px, 128px"表示当视区宽度不大于360像素时候，图片的宽度限制为340像素，其他情况下，使用128像素（对应下面demo页面第1张图）。  如果sizes="128px", 则尺寸就一直是128像素，图片只会根据设备像素比发生变化。  注意，这里所有的值都是指宽度值，且单位任意，em, px, cm, vw, ...都是可以的，甚至可以CSS3的calc计算（对应下面demo页面第2张图），例如：  sizes="(max-width: 360px) calc(100vw - 20px), 128px"  表示当视区宽度不大于360像素时候，图片宽度为整个视区宽度减去20像素的大小。 |
| 关于图片的响应实现 | img {  max-width: 100%;  } |
| @media only screen and (-webkit-min-device-pixel-ratio: 2), only screen and (min–moz-device-pixel-ratio: 2), only screen and (-moz-min-device-pixel-ratio: 2), only screen and (-o-min-device-pixel-ratio: 2/1), only screen and (min-device-pixel-ratio: 2), only screen and (min-resolution: 192dpi), only screen and (min-resolution: 2dppx) { / styles for Retina-type displays / } |
| 此处指的确定宽度图片，是指样式属性宽度设置为一个确定px的图片。  确定宽度的照片所占物理的像素只与设备像素比有关，所以只需要 srcset 属性的 x 标识符。  <img src="200px.png" srcset="400px.png 2x, 600px.png 3x, 800px.png 4x">  这样加上srcset属性，浏览器就会根据自己的设备像素比来加载不同的图片，不支持srcset属性的浏览器就默认加载200px.png。这个方法是向下兼容的。 |
| 此处指的不确定宽度的图片，是指样式属性宽度设置为确定百分比的图片。  如果是不确定宽度的照片的话（或者是手机上不确定图片宽度，电脑上确定宽度等），也可以通过 srcset 解决。由于像素是不能确定的，所以很难达到照片本身和照片所占物理像素是一样的，因为用户的窗口大小是不确定的。  不确定宽度的照片所占物理的像素与设备像素比和所占宽度有关，使用了 srcset 属性的 w 标识符后，只需要指定宽度，浏览器会自动根据设备像素比来选择最优图片。注意，每张图片的宽度都要用w描述符来声明，w的值是图片以px为单位的真实宽度。但是w标识符只是一个描述符，不是单位。具体解释可以参考：[响应式图片srcset全新释义sizes属性w描述符](http://www.zhangxinxu.com/wordpress/2014/10/responsive-images-srcset-size-w-descriptor/)。  如果这个图片占整个窗口的 100% 大小（即 100vw 宽），那么只需要这样设置 srcset 就能达到效果。  <img src="800px.png" srcset="400px.png 400w, 800px.png 800w, 1600px.png 1600w, 2400px.png 2400w, 3200px.png 3200w">  不过此时还有一个小问题，浏览器比较笨，它会认为图片宽度始终是父容器 100% 宽，如果图片不是占父容器的 100%，两边留有边框怎么办呢？此时就需要使用 size 属性。  sizes 属性里制定图片的宽度，不能使用百分比单位，但可以使用 vw、px 等单位（100vw 就是整个父容器的宽度)，也可以使用 calc 运算，同时支持媒体查询。要保证使用 sizes 里计算出来的宽度始终是图片所占父容器的宽度。  以下几种典型案例示范：  1. 两边边框为百分比  如果图片两侧边框（指图片边缘到窗口边缘长度）始终是占屏幕的一定的百分比时，那么图片总会有一个相对于整个窗口的百分比。比如图片两侧边框始终是整个窗口的 10%，那么 size 属性就可以如下设置：  sizes="80vw"  2. 两边边框为确定的像素  比如图片两侧边框始终是 10px，那么 size 属性就可以如下设置：  sizes="calc( 100vw - 10px )"  4. 图片有最大尺寸  sizes还可以做一些媒体查询。比如图片两侧边框始终是 10px，但是图片最大只能是 1000px，那么 size 属性就可以如下设置：  sizes="(min-width: 1020px) 1000px, calc( 100vw - 10px )"  5. 综合实例如：  <img src="800px.png" srcset="400px.png 400w, 800px.png 800w, 1600px.png 1600w, 2400px.png 2400w, 3200px.png 3200w" sizes="(min-width: 1020px) 1000px, calc( 100vw - 10px )" alt="hello world!"> |
| 火狐有效，360无效？？？ |
|  |

用字体画icon小图标

|  |  |  |
| --- | --- | --- |
| 方法1 | CSS sprite  使用背景图片定位的方式  background-image:  background-position: |  |
| 方法2  font-face | font+HTML  字体图标（HTML实体）    优点：  1、灵活性：轻松地改变图标的颜色或者其他的CSS效果  2、可扩展性：图标的大小很容易改变[font-size]  3、矢量性：图标是矢量的，缩放图标不会影响图标的性·清晰度，PC平板手机上都能高清地显示  4、字体图标兼容所有现代浏览器 包括IE6  5、本地使用：通过添加定制字体到您的本地系统中，你可以在各种不同的设计和编辑应用程序中使用它们  工具：IconMoon[https://icomoon.io](https://iconmoon.io)；可以生成图标与SVG   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | **字体简介** | | | | | | | | | **EOT** | Embedded OpenType Fonts | | | 微软开发的用于嵌入网页中的字体，ie专用字体 | | | | | **WOFF** | The Web Open Font Format | | | web字体中最佳格式，是一个开发的TrueType/OpenType的压缩版本，2009年被开发，如今被W3C组织推荐标准 | | | | | **TTF** | TrueType Fonts | | | 1980s，由Microsoft和Apple联合开发的一套字体标准，是Mac OS和WIN操作系统中最常见的一种字体 | | | | | **SVG** | SVG Fonts | | | 用于SVG字体渲染的一种格式，他是有W3C指定的开发标准的图形格式 | | | | | **浏览器对字体的支持** | | | | | | | | | **font format** | | ie | 谷歌 | | 火狐 |  | opeate | | **EOT fonts** | | 6.0 | X | | X | X | X | | **WOFF fonts** | | 9.0 | 5.0 | | 3.6 | 5.1 | 11.1 | | **TTF fonts** | | 9.0\* | 4.0 | | 3.5 | 3.1 | 10.0 | | **SVG fonts** | | X | 4.0 | | X | 3.2 | 9.0 | | |
| @font-face  自定义 | 用于将自己定义的WEB字体应用到网页之中。  @font-face {  font-family: <YourWebFontName>;  src: <source> [<format>][,<source> [<format>]]\*;  [font-weight: <weight>];  [font-style: <style>];  }  使用方式：  @font-face {  font-family: 'YourWebFontName';  src: url('YourWebFontName.eot?') format('eot');/\*IE\*/  src: url('YourWebFontName.woff') format('woff'),  url('YourWebFontName.ttf') format('truetype');/\*non-IE\*/  }  @font-face {  font-family: 'name';  src: url('name.eot'); /\* IE9 Compat Modes \*/  src: **url('name.eot?#iefix') format('embedded-opentype'), /\* IE6-IE8 \*/**  url('name.woff') format('woff'), /\* Modern Browsers \*/  url('name.ttf') format('truetype'), /\* Safari, Android, iOS \*/  url('name.svg#name') format('svg'); /\* Legacy iOS \*/  } | |
| 应用1  font+HTML实体 | @font-face{  font-family: "imooc-icon";  src: url("../fonts/icomoon.eot"); /\* IE9 兼容模式 \*/  src: url("../fonts/icomoon.eot?#iefix") format("embedded-opentype")  ,url("../fonts/icomoon.woff") format("woff")  ,url("../fonts/icomoon.ttf") format("truetype")  ,url("../fonts/icomoon.svg") format("svg");  font-weight: normal;  font-style: normal; } .imooc-icon{  font-family: "imooc-icon";  font-style: normal;  font-weight: normal;  font-size: 64px;  -webkit-font-smoothing: antialiased; /\* 增强清晰度\*/  -moz-osx-font-smoothing: grayscale; }  **<i style="color: red;" class="imooc-icon">&#xf048;</i>**  **<i style="font-size:60px;" class="imooc-icon">&#xe607;</i>** | |
| 应用2  font+css |  | |

css选择器高效使用

|  |  |
| --- | --- |
|  | CSS选择器效率从高到低的排序如下：  ID选择器 比如#header  类选择器 比如.promo  元素选择器 比如 div  兄弟选择器 比如 h2 + p  子选择器 比如 li > ul  后代选择器 比如 ul a 7. 通用选择器 比如 \*  属性选择器 比如 type = "text"  伪类/伪元素选择器 比如 a:hover  以上引用自[Steve Souders](http://stevesouders.com/)的Even Faster网站: |
|  | 组合选择器  你可以有一个标准的选择器比如 #nav，来选择任何带有ID为"nav"的元素，或在你可以有一个组合选择器比如#nav a,来选择任何在ID为’nav’的元素里面的链接元素  此刻，我们读这些是从左到右的方式。我们是先找到#nav，然后从它的里面找其他元素。但是浏览器解析这些不是这样的：浏览器解析选择器是从右到左的方式。  在我们看来，#nav里面带了一个a，浏览器却是看到的a在#nav里面。这些细微的差异对选择器的效率有很大的影响，同时学这些差异也是很有价值的。  如果想要知道更多浏览器这样解析的原因，请看[Stack Overflow](http://stackoverflow.com/questions/5797014/why-do-browsers-match-CSS-selectors-from-right-to-left)上的讨论  浏览器从最右边的元素开始（它想要渲染的元素），然后用它的方式回溯DOM树比从DOM树的最高层开始选择向下寻找，甚至可能达不到最右边的选择器—关键的选择器要高效。  这些对CSS选择器的效率有很大的影响。  关键选择器  关键选择器，正如前面讨论的一样，是一个复杂的CSS选择器中最右边部分。它是浏览器最先寻找的。  现在我们回到讨论开始的地方，哪类选择器是最高效的？哪个是会影响选择器效率的关键选择器；写CSS代码的时候，关键选择器是能否高效的决定因素。 一个关键CSS选择器像这样：  CSS   |  |  | | --- | --- | | 1 | #content .intro {..} |   是不是高效选择器比如类选择器天生就高效？浏览器会寻找.intro的实例（可能会很多），然后沿着DOM树向上查找，确定刚才找到的实例是否在一个带有ID为"content"的容器里面。  但是，下面的选择器就表现的不是那么好了：  CSS   |  |  | | --- | --- | | 1 | #content \* {..} |   这个选择器所做的是选择所有在页面上的单个元素（是每个单个的元素），然后去看看它们是否有一个 #content 的父元素。这是一个非常不高效选择器因为它的关键选择器执行开销太大了。  运用这些知识我们就可以在分类和选择元素的时候做出更好的选择。  假设你有一个复杂的页面，它相当巨大并且在你的一个很大很大的站点上。在那个页面上有成百上千甚至上万的 a 标签。它还有一个小的社交链接区域放在一个ID为#social的Ul里面。我们假设它们是Twitter，Facebook，Dribbble还有 Google+的链接吧。在这个页面上我们有四个社交链接和成百上千的其他链接。 下面的这个选择器就自然的不是那么高效和合理了：  CSS   |  |  | | --- | --- | | 1 | #social a {…} |   这里发生的情况是浏览器会在定位到#social区域下的四个链接之前得到页面上所有成千上万的链接。我们的关键选择器匹配了太多我们不感兴趣的其他元素。  为了补救我们可以给每个在社交链接区域的 a 增加一个更特殊、明确的选择器 .social-link , 但是这好像有点违背我们的认知：当我们能用组合选择器的时候就不要放不必要的类标示在元素上。  这就是为什么我对选择器的性能如此感兴趣的原因了：必须在web 标准最佳实践和速度之间的保持平衡。  通常我们有：  CSS   |  |  | | --- | --- | | 1  2  3  4  5  6 | <ul id="social">      <li><a href="#" class="twitter">Twitter</a></li>      <li><a href="#" class="facebook">Facebook</a></li>      <li><a href="#" class="dribble">Dribbble</a></li>      <li><a href="#" class="gplus">Google+</a></li>  </ul> |   CSS:  CSS   |  |  | | --- | --- | | 1 | #social a {} |   我们现在最好有：  XHTML   |  |  | | --- | --- | | 1  2  3  4  5  6 | <ul id="social">      <li><a href="#" class="social-link twitter">Twitter</a></li>      <li><a href="#" class="social-link facebook">Facebook</a></li>      <li><a href="#" class="social-link dribble">Dribbble</a></li>      <li><a href="#" class="social-link gplus">Google+</a></li>  </ul> |   加上CSS：  CSS   |  |  | | --- | --- | | 1 | #social .social-link {} |   这个新的关键选择器将会匹配更少的元素，这意味着浏览器能够很快的找到它们并渲染特定的样式，然后专注于下一件事。  另外，事实上我们可以用.social-link{}更清晰的选择，而不是过分限制它。阅读下一部分你会原因…  简单的重述一次，你的关键选择器会决定浏览器的工作量，因此，我们应该重视一下关键选择器  过度限制选择器  现在我们知道了什么是关键选择器，还有它是大部分工作的来源，但是我们可以更乐观一点。拥有一个明确的关键选择器最大的好处就是你可以避免使用过度限制选择器。一个过度限制选择器可能像：  CSS   |  |  | | --- | --- | | 1 | html body .wrapper #content a {} |   这里的写的太多了，至少3个选择器是完全不需要的。它可以最多像这个样子：  CSS   |  |  | | --- | --- | | 1 | #content a {} |   这会发生什么呢？ 首先第一个意味着浏览器不得不寻找所有的 a 元素，然后检查他们是否在一个ID为"content"的元素中，然后如此循环直到HTML标签。这样造成了太多的我们不太想要的花费。了解了这个，我们得到一些更现实的例子：  CSS   |  |  | | --- | --- | | 1 | #nav li a{} |   变成这个：  CSS   |  |  | | --- | --- | | 1 | #nav a {} |   我们知道如果a在li里面，它也必定在#nav里面，所有我们可以马上把li从选择器组中拿掉。然后，既然我们知道在页面中只有一个ID为nav的元素，那么它依附的元素就是完全没有关系得了，我们也可以拿掉ul  过度限制选择器使浏览器工作比它实际需要的更繁重，花费的时间更多。我们可以删掉不必需的限制，来使我们的选择器更简单和高效。  这些真的需要吗？  最短的答案是:或许不是。  最长的答案是：它取决于你正在搭建的站点。如果你正在为你的晋升而努力，那么就好好写出简单、高效的CSS代码吧，因为你可能不会感觉到它给你带来的改变。 如果你正在搭建下一个每个页面都以毫秒计算的Amazon网站，这样有时速度会很快，但有时可能不是。  浏览器将会在解析CSS的速度上变得更好，甚至在手机端。在一个网站上，你不太可能会觉察到一个低效的CSS选择器，但是….  但是  它确实发生了，浏览器还是不得不去做我们讨论的所有工作，无论它们变得多快。即使你不需要或者甚至不想实践任何一个，但是它都是我们值得学习的知识。请记住选择器可能会让你付出很大代价，你应该避免盯着一个看。这意味着如果你发现你自己在写像这样的：  CSS   |  |  | | --- | --- | | 1 | div:nth-of-type(3) ul:last-child li:nth-of-type(odd) \*{ font-weight:bold } |   这时，你可能就做错了。  现在，在高效选择器的世界我还是一个新人。所以如果我忘记了什么，或者你有需要补充的，请在评论里面留言。  更多高效选择器  我还不能完全介绍Steve Souders的网站和书籍（《[更快速网站](http://www.amazon.cn/gp/product/B003B00A7E/ref=as_li_qf_sp_asin_il_tl?ie=UTF8&camp=536&creative=3200&creativeASIN=B003B00A7E&linkCode=as2&tag=vastwork-23" \t "http://blog.jobbole.com/35339/_blank)》、《[高性能网站](http://www.amazon.cn/gp/product/B001A4E092/ref=as_li_qf_sp_asin_il_tl?ie=UTF8&camp=536&creative=3200&creativeASIN=B001A4E092&linkCode=as2&tag=vastwork-23" \t "http://blog.jobbole.com/35339/_blank)》），它们是如此之好，以至于值得你花更多时间来阅读和推荐。这个家伙只有他自己才了解自己！  [High Performance Web Sites](http://www.amazon.cn/gp/product/B003B00A7E/ref=as_li_qf_sp_asin_il_tl?ie=UTF8&camp=536&creative=3200&creativeASIN=B003B00A7E&linkCode=as2&tag=vastwork-23)[Even Faster Web Sites](http://www.amazon.cn/gp/product/B001A4E092/ref=as_li_qf_sp_asin_il_tl?ie=UTF8&camp=536&creative=3200&creativeASIN=B001A4E092&linkCode=as2&tag=vastwork-23) |