@media 媒体查询

2017年4月23日 11:02

<link href="file" rel="stylesheet" media="logic media and (expression)">
如<link href="style.css" rel="stylesheet" media="screen">
<link href="desktop.css" rel="stylesheet" media="(min-width: 600px)">

logic media可取：
screen 用于电脑屏幕
print 用于打印

具体语法：
@import url('file') logic media and (expression);
@media logic media and (expression) { rules }
@media all and (expression) { rules }
@media (expression) { rules }
@media only media and (expression) { rules }
@media not media and (expression) { rules }

width&height

@media (width: 600px) { span{ color:red;} } 视口宽度为600时span颜色为红色
@media (max-width: 480px) { span{ color:red;}} 视口宽度小于480时...
@media (min-width: 640px) { span{ color:red;}} 视口宽度大于640时

同理@media (height: 600px){....}

像素密度 pixel ratio

ppi 指屏幕上每英寸可以显示的像素点的数量，即屏幕像素密度

DPR：物理像素/CSS像素，device pixel ratio
例如：iphone 5s 物理分辨率为640x1136，CSS分辨率为320*568（默认的viewport尺寸），即一个css像素对应两个物理像素，其DPR为2

单位：
DPCM: dots per centimeter
DPI :dots per inch
DPPX: dots per pixel 一个单位的像素，即一个单位的分辨率，与DPR一比一对应

resolution 分辨率
@media (resolution: 1.5dppx) { rules }
@media (max-resolution: number) { rules }
@media (min-resolution: number) { rules }

例如：在像素密度高于1.5的电脑上用另一个分辨率高的图片

```
E{ background-image: url('image-lores.png'); }
 @media (min-resolution: 1.5dppx) {
        background-image: url('image-hires.png');
        background-size: 100% 100%;
}
```

注：IE10-不支持dppx,支持DPI，1DPR=96DPI

@media (resolution: 1.5dppx), (resolution: 144dpi) { rules }

手机上横向纵向布局

@media (orientation: value) { rules }

landscape 横向

portrait 纵向

扩展：

```
@supports(column-count:2) and (display:flex){  //浏览器支持该属性的情况下，才会进来
        div{
        column-count:2;
        }
}
```

容易混淆

viewport 设置手机浏览器窗口宽度

响应式 设计理念：让页面能在不同的设备下合理适配

多组媒体条件一起使用

@media screen and (orientation: landscape), print and (orientation: portrait) {...}

移动布局

2017年4月19日 15:58

<meta name="viewport" content="width=device-width" >
用来指示手机浏览器以多宽（多少px的初始包含块）的视口渲染页面的

扩展：
<meta name="viewport" content="width=device-width,user-scalable=yes/no,initial-scale=1.0,maximum-scale=5.0" >
user-scalable=yes/no 是否允许用户放大，一般不设置

content值：

width=device-width：不同的手机渲染的尺寸比较接近，越大的手机显示内容越多，需要考虑各种尺寸的显示

width=400：指定宽度值，支持性不太好，新的浏览器才支持

解决办法：
假如视觉稿的宽度是x
希望页面在不同的手机上显示比例相同（就像同一张照片显示在不同的手机）
好处是不用针对不同宽度的屏幕（视口）

视觉稿上的标注能够尽量少去做计算
希望视觉稿上的一个数就对应开发时的一个数值
就需要有一个单位，能够与视觉稿上的数值相差10的n次方倍数
需要1rem=10^n视觉稿数值

100vw=视觉稿宽度
1rem=100vw/视觉稿宽度
可以如下设置：

```
html{
    font-size:calc(100vw/视觉稿宽度)
}
```

页面剩余部分的布局使用rem为长度单位

1、对于支持把viewport写死的浏览器，直接写<meta name="viewport" content= “width=视觉稿宽度">
页面全部使用px为长度单位，视觉稿上标注的所熟知是多少，页面里就用px

2、对于不支持viewport写死的浏览器
写<meta name="viewport" content="width=device-width">
然后重置1rem为视觉上的一个像素
公式：100vw=视觉稿宽度

```
Xrem=100vw
1rem=100vw/X
设置
html{
    font-size:calc(100vw/X);
}
```

当页面使用视觉稿宽度数值太大，calc(100vw/x)将会太小，有些浏览器会重置font-size的值为允许的最小值

于是写成
html{
font-size:calc(100vw/x*100); //量出来元素的宽度需要缩小100倍
}
页面中仍然使用rem为长度开发单位，但视觉稿上的一个数值对应0.01rem
对于不支持calc或vw的浏览器，使用js读取视口宽度，动态设置到html元素上去。

3、如果开发的是一个信息类页面，大量的文字（希望屏幕越大，显示的文字越多）
width=device-width，页面使用px开发

各种情况分析：
1、安卓5.0以上，严格按比例还原视觉稿布局（按比例即如看照片一样，在不同的手机上显示的内容一样多，5.0以上viewport支持写死）
width=视觉稿宽度，使用px开发，缩放时，浏览器自动按比例缩放，同在pc端开发
2、安卓4.3以下，严格按比例还原视觉稿布局（即viewport不支持写死）
width=device-width
html.sytle.fontSize=document.innerWidth/视觉稿宽度+'px'（最优，都支持）
或<!--html{font-size:calc(100vw/x)}-->，需要考虑是否支持calc
使用rem开发
3、安卓5.0以上/安卓5.0以下，页面内容与屏幕大小呈正相关（正相关即不需要等比例缩放，大的手机显示内容多，小得显示内容少）
width=device-width
使用px开发
4、布局（各元素的大小）保持设计稿比例，文字多少与屏幕呈正相关
width=device-width
html.style.fontSize=document.innerWidth/x+ ‘px’
<!--html{font-size:calc(100vw/x)}-->
布局用rem,文字用px书写

可以参看http://m.mi.com/小米手机页面，得出他的设计稿宽度为720px 370(vw)/52 (font-size)

@font-face

2017年4月23日 13:39

@font-face {
font-family: FontName; ——自定义字体名称
src: local('fontname'), url('/path/filename.otf') format('opentype');
——local本地字体名称，url线上字体文件路径，format字体文件格式
}

例：可以把本地名称复杂的字体自定义，这样方便使用

```
@font-face {
    font-family: yh;
    src: local('microsoft yahei'), url('ChunkFive.woff') format('woff');
}
h1.webfont { font-family: yh, sans-serif; }
```

或者这样，让代码看起来更简洁

```
@font-face {
    font-family: X;
    src: local(a), local(b), local(c), serif;
}
div {
    font-family: X;
}
```

同一个自定义字体，可增加一些字体样式，html会自动匹配需要的样式（需要html自带这种样式或在样式表里设置了才匹配的到！！）

```
@font-face {
  font-family: 'Gentium Basic';
  src: url('GenBasR.woff') format('woff');
}
@font-face {
  font-family: 'Gentium Basic';
  font-style: italic;
  src: url('GenBasI.woff') format('woff');
}
h1 { font-family: 'Gentium Basic', sans-serif; }
<h1>I knew him, Horatio</h1>
<h1><em>I knew him, Horatio</em></h1>
```

此时em标签因为本身是斜体，所以会匹配到带斜体的第二种字体

I knew him, Horatio
I knew him, Horatio

注：如果在字体中没有该类字体的斜体，则会自行计算把字体倾斜，即oblique，而非italic

字体子集，只放需要的文字到字体文件中，减小文件大小
中文网站：字蛛

字体图标原理（Icon Font）
把字体里的文字设计成图标的样子，然后通过@font-face声明引入该字体，为了不直接在html里使用unicode,可以通过设定一个类的伪元素的方法，伪元素的content即为unicode值，然后可以直接在标签上加class即可，最终显示为了一个图标。一般把这些图标字体放置在Unicode的空白位置，如 plane \f025
https://netdna.bootstrapcdn.com/font-awesome/4.0.3/css/font-awesome.min.css

Icon Font 与 css sprite
css sprite： 不能变色，不太好变尺寸
Icon Font 字体图标。优点：易用，速度快，矢量图，随意变色；随意放大不失真；文件（流量）尺寸小（远小于图片）。如Font Awesome
缺点：颜色单一（或者渲染成css3渐变色，通过bg-clip到字体）；没有动画；不易制做（设计svg格式的各个图标，使用特定的工具即可生成），大部分为用伪元素生成，无法选中

FOUC：flash of Unstyled Content
页面的css代码过于晚于页面的html代码的解析与加载，浏览器会先展示没有样式的页面，等到css加载完成后，页面由无颜色直接变成有样式，会产生一个闪烁，就叫Fouc
产生：外部css加载比较慢，如css与html代码不在同一个站点上，或者在css中使用了@import指令
解决方法：把css代码直接放入style标签里，这会导致没有缓存了（如果是通用的样式），所以一般只在重要的页面（或内容不经常变换的页面）这么做，如首页
FOUT：flash of Unstyled Text　字体闪烁
字体文件加载缓慢，浏览器先显示fallback的字体，等字体文件加载完成之后，再切换为css里声明的字体，之间也会产生一个闪烁
解决方案：用base64编码把字体文件嵌入到css里面；
减小字体文件的大小（中文字体才需要，英文字体本来就很小），使用font subset,字体子集（只适用于中文）

2017年4月21日　　9:21

text-overflow　只能针对单行文本，需要与overflow:hidden;连用
clip　折断
ellipsis 省略号

如：

```
p{
        overflow:hidden;
        text-overflow:ellipsis;
        white-space:nowrap;
}
```

-wekit-line-clamp 针对多行文本的省略号

```
p{
  overflow : hidden;
  text-overflow: ellipsis;
  display: -webkit-box;
  -webkit-line-clamp: 2;
  -webkit-box-orient: vertical;
}
```

text-align-adjust

-webkit-text-stroke 空心字

```
 p{
        font-size:50px;
        -webkit-text-stroke:2px blue;
         color:transparent;
}
```

用text-shadow实现空心字

```
div{
    color:white;
    font-size:50px;
    text-shadow:
    0 0 2px black,
    0 0 2px black,
    0 0 2px black;
}
```

word-wrap 单词折断，超过块宽度的情况下

break-word

The condition of silicosis is sometimes factitiously referred to as Pneumonoultramicroscopicsilicovolcanoconiosis.

The condition of silicosis is sometimes factitiously referred to as Pneumonoultramicroscopicsilicovolcanoconiosis.

resize 需要与overflow:hidden连用

both

horizontal

vertical

both

none

可以同textarea一样在右下角拖动

multiple columns

2017年4月21日 14:53

因多列而导致内容被折断
解决办法：
让元素display:inline-blcok
使内容一直在方块，不会被折断

column-count 列数

column-width 列宽

column-fill 填充
balance 每列尽量等量填充
auto 前一块填充满了才会往后面一列填充

column-gap 间隙

column-rule 列分隔线，用法同border
column-rule: 57px solid pink;

column-span 让元素不在分列里，横跨多列
all
none 默认
例如：

```
div {
  column-count: 3;
}
h1{
    column-span:all;
    text-align:Center;
}
<div>
<h1>lalala</h1>
  <p>Lorem ipsum dolor sit amet, conse</p>
</div>
```

columns: column-width column-count;
顺序可调，也可省略其一

自动多定宽列布局：
C:\damiao\practice\columns to flex.html

gradient

2017年5月17日 13:17

http://lea.verou.me/css3patterns/#

background-image:

linear-gradient(currentColor,green);渐变背景图片，从当前的颜色渐变到绿色

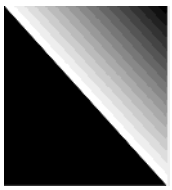
linear-gradient(-50deg, #07beea 20%,red, #444); 从某个角度开始渐变，角度设置：从下到上，从左到右旋转

linear-gradient(to left, #07beea 20%,red 73%,black 95%, #444); 从右边到左边渐变

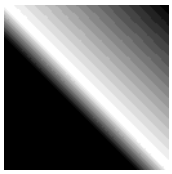
linear-gradient(to top right, #000, #f00 50%, #090); 从左下角开始渐变到右上角

linear-gradient(to left bottom,black,white 50%,black 50%); 线性渐变

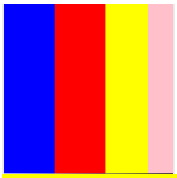
从右上角渐变到左下角，由黑色渐变到白色，在50%的地方渐变结束为白色，开始渐变为下个颜色，黑色从50%的地方开始渐变，由于后面没有其他颜色，所以一直是黑色



如果black改成40%,将没有反应，因为比上个颜色值小，所以被重置为了之前的颜色，改成60%即从60%的地方开始渐变，如下



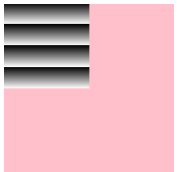
linear-gradient(to right, blue 60px,red 0px,red 120px,yellow 0,yellow 170px,pink 0);



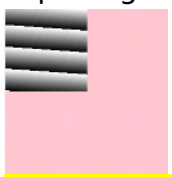
repeating-linear-gradient(black,white 25%); 重复设置

background-repeat:no-repeat;

background-size: 50%,50%;

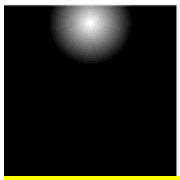


repeating-linear-gradient(4deg,black,white 25%); 角度设置：从下到上，从左到右旋转

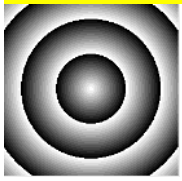


radial-gradient(white,black) 放射渐变，默认值ellipse椭圆，发散到四个角

radial-gradient(circle 50px at 50% 10%,white,black) 50px为圆半径， at后为圆心位置

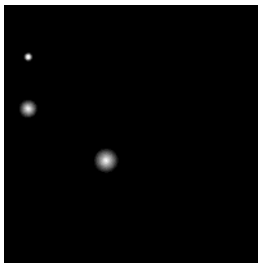


repeating-radial-gradient(circle 40px,white,black)



需要在一个元素上画多个背景图，需要把渐变到最后的颜色改为透明色，否则将看不到

```
background-image:
radial-gradient(circle 5px at 10% 20%,white 1px,black,transparent 6px),
radial-gradient(circle 5px at 10% 40%,white,black,transparent 15px),
radial-gradient(circle 10px at 40% 60%,white,black);
```



2017年5月17日 18:25

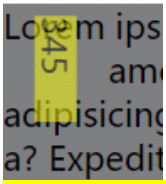
坐标轴：x向右，y向下，z向外

transform 可取以下所有变化值，按空格分开

transform:rotateX(10deg) rotateX(20deg) => 即在10度后的坐标系的标准下，又旋转20度

rotate(10deg) 旋转，顺时针方向，连带坐标系也旋转，此时如果设置translateX或者translateY，都是沿着旋转之后的坐标的方向移动

旋转动画只影响页面的绘制，不影响布局（即设置浮动也不会脱离文档流，一直在正常流里），会覆盖后出现的元素



transform-origin 变化定点，默认水平垂直中心

transform-origin:0 0 ;

left top;//right bottom center

10px 10px; //left top

translate 相对自己位移，连带坐标系产生位移，不产生回流，只是重绘，相对于left,right性能好

取值：

translateX(10px) translateY(20px)

translate(10px,20px)

translate(10px) 水平10px,垂直为0

scale 拉伸，连带坐标系也进行拉伸，此后若有位移则同比例拉伸

scaleX(2) 水平方向拉伸两倍

scaleX(-1) 方向被翻转，即如同照镜子

scale(0.5) 缩小

scaleY(-1)同理

scale(1,1)

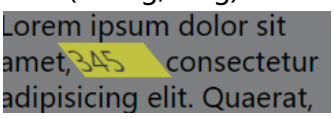
scale(2)

skew 倾斜

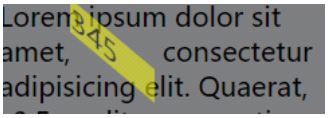
skewX(15deg)

skewY(-15deg)

skew(40deg,0deg)



skew(0deg,40deg)



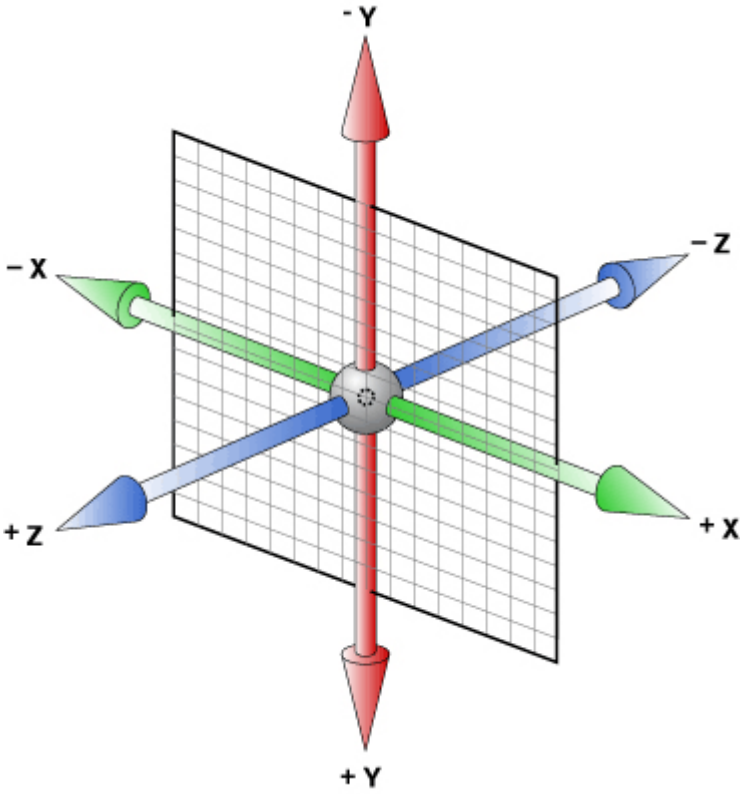
transform: rotate(-15deg);
transform: skewx(15deg) skewy(-15deg); 两者效果相同

Matrices 矩阵变化 上面的所有动画效果最终都会转为矩阵变化
transform:matrix(a,b,c,d,X,Y)
原理：a b x
c d * y 看成两个矩阵相乘，x,y为原来元素的所有点
最终得到新的点 ， 需要加上X,Y为偏移量
(ax+by+X,cx+dy+Y)

如下同以上属性效果
放大：matrix(2,0,0,2,0,0)
平移：matrix(2,0,0,2,15,15)
倾斜: matrix(1,tan(angle),0,1,X,Y) x轴
matrix(1,0,tan(angle),1,X,Y) y轴
transform:matrix(1,0.27,0,1,0,0);
旋转：matrix(cos(angle),sin(angle),-sin(angle),cos(angle),X,Y)
transform:matrix(0.5,0.87,-0.87,0.5,0,0)

2017年5月19日 9:13

坐标轴：x向右，y向下，z向屏幕外



transform 可取以下所有变化值

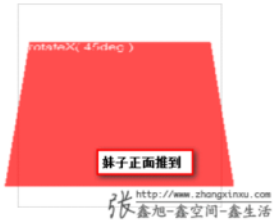
rotate

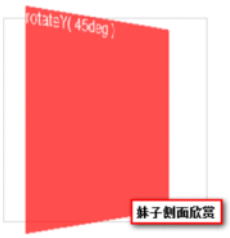
rotateX(angle)

rotateY(angle)

rotateZ(angle)

rotate3d(x,y,z,angle) 从（0,0,0）到（x,y,z）拉一条线，沿着该线转





perspective(depth) 视角深度，作用于自身

必须写在transform的第一位，否则失效

值越大越远，即越小，正常为1000，即在面前

translate

translateZ(length)

translate3d(translateX,translateY,translateZ)　针对border-box产生位移

scale　放大因子
scaleZ(number)
scale3d(scaleX,scaleY,scaleZ)

CSS中zoom:2与transform: scale(2)有何区别
zoom会把元素的css像素放大
scale只是做拉伸，类似图片放大
可以利用zoom值，避免页面被放大，一检测到页面放大，就调整zoom值，即保持 zoom*放大量=1

perspective:depth; 同上perspective(depth) ，默认相当于z=0,只作用于他的子元素
transform:perspective(1000);与perspective:1000 具体区别看
http://www.zhangxinxu.com/study/201209/transform-perspective-same-rotate.html

perspective-origin:x-position y-position; 视角的入口点，默认为center center
x-position 可以取关键字：left/right/center,百分数或长度
y-position可以取关键字：top/bottom/center,百分数或长度

transform-origin:x y z; 不动点，z只能取长度px,z如果为负值，则该点就会到元素的后面
即相当于把坐标轴的原点移动到另一个位置，但其位置还是为（0,0）,因此在该点上用矩阵法始终是0，所以其不会动
默认50% 50% 0，以自己中心为不动点

`transform-origin` CSS属性让你更改一个元素变形的原点。例如，`rotate()`的 transform-origin 是旋转的中心点 (这个属性的应用原理是先用这个属性的负值translate该元素，进行变形，然后再用这个属性的值把元素translate回去)。

即以下两者等价
transform:rotate(30deg)
transform-origin:200px 300px

transform:translate(200px,300px) rotate(30deg) translate(-200px,-300px)
transform-origin:0 0

transform-style 子元素的投影默认在父元素上，用在父元素上
flat 默认，平的，父元素做3d变化时，子元素是平的
preserve-3d 分开, ie10/11不支持

backface-visibility 背面是否显示，要与preserve-3d一起用

hidden
visible

渐变transition

2017年3月29日 16:12

渐变transition，

属性 时间 缓动方式,下一组,...;该元素在【属性】发生变化时，用【时间】的长度按照【缓动】的方式变化到目标值

transition:color 1s linear,width 5s linear; 在1s之内颜色匀速变为目标值，在5s之内宽度匀速变为目标值

transition:all 5s linear; 在5s内所有属性都匀速变为目标值

transition:all 5s cubic-bezier(.17,.67,.83,.67)

缓动函数

linear为匀速

ease为先慢后快， ease-in ,ease-out,ease-in-out

steps(5) 跳跃式,分5次

如 transition:width 20s steps(5,start) 20秒分五步，每步4s,4秒跳一次,从一开始就跳

如 transition:width 20s steps(5,end) 20秒分五步，每步4s,4秒跳一次，4秒之后才开始跳

注：在display为none的情况渐变到display:block动画会失效，源元素不存在无法加动画，数值、颜色才能发生渐变，也不能从值为auto开始。

如：height从0到auto自适应动画无效，但是可以设置max-height:20到max-height:9999，因为最大高度即为自适应，可以实现缓动到自适应的效果

transition-delay:2s; 可用在visibility属性上，延迟2s才开始渐变

transition-property:all;造成的问题

所有的属性都会产生渐变，有可能会引起频繁的回流，造成页面卡顿

同时如果元素的display也发生变化的话，可能会让本来有的动画的属性变成没有

在chrome里，transition-property:all不会使z-index属性发生渐变

可以这样写：

```
transition: all 2s linear 3s;
transition-property: a, b, c, d;
```

当属性数量多于时间数量，则还是从第一个开始取 表示所有属性间隔时间都是2s

```
transitio-property: a, b, c, d;
transition-duration: 2s;
transition-timing-function: linear;
transition-delay: 3s;
```

缓动函数timing function ，元素的各项数据的变化随时间的图像，距离-时间 图像，速度-时间 图像

其原理cubic-bezier函数，四个点确定三阶贝塞尔曲线

如

鼠标移动到div上，渐变成目标值，鼠标放开，又会按原速度返回

例子：若需要鼠标放开时直接回退而非渐变回退，则把transition放到div:hover里即可

```
div{
  width:50%;
  border:10px solid;
  transition:all 5s cubic-bezier(.17,.67,.83,.67)
}
div:hover{
  border:20px solid;
  border-color:red;
  width:100%;
}
```

例子：鼠标移上去不渐变，移开才渐变

```
div{
  border:5px solid;
  width:100px;
  height:50px;
 transition:width 1s linear;
}
div:hover{
  width:400px;
  transition:none;
}
```

具体可参看 http://cubic-bezier.com/

可渐变属性与不可渐变属性的最大区别：

可渐变属性是连续的数值，不可渐变属性是离散的（z-index：都是整数）

2017年5月17日 21:04

IE10支持，IE9不支持，transition、animation、gradient

```
.box {
    background: url(loading_blue.gif) no-repeat center;
    background: url(loading_blue.png) no-repeat center, linear-gradient(to top, transparent, transparent); /* IE10+ */
    animation: spin 1s linear infinite;      /* IE9+ */
}
```

需要注意的是，下面的background只能是background，虽然理论上使用background-image也是可以的，但是在IE7, IE8浏览器下面，background-image如果是个不认识的东西，他们不会认为这行CSS无效，而是认为你这个背景图有问题，于是会导致IE7,IE8浏览器下连gif loading图片都实现不出来

```
.box {
    background: url(test.png);          /* IE8 */
    background: url(test.svg), none;    /* IE9+ */
}
```

```
.box {
    box-shadow: 0 1px 3px rgba(0,0,0,.25);  /* IE9+ */
    border: 1px solid #d0d0d5;                  /* IE7,8 */
    border: 0 rgba(0,0,0,.2);                   /* IE9+ */
}
```

来自 <http://www.zhangxinxu.com/wordpress/2016/10/browser-css-property-down-compatible-hack-technology/>

2017年5月19日 23:02

```
@keyframes name{
        from {
          width:10px
        }
        to{
         width:20px
        }
}
```

animation-name:myani;
animation-duration:6s;
animation-timing-function:ease-in;
animation-delay:1s;
animation-iteration-count:10/infinite; 默认为1次，迭代次数
infinite为无穷次
animation-direction:normal/alternate/reverse/alternate-reverse/nomal reverse/...
alternate:来回播放，来回计为2次，count为2次
animation-fill-mode:none/backwards/forwards/both;（次数为有限时才有效）
forwards:动画在结束后保持最后一帧状态
backwards:动画在delay时间里就变到第一帧的状态
both:由以上两个共同效果
none:开始之前与结束之后都是原始本来的状态
animation-play-state:paused/running;

简写：
animation:name duration timing-function delay iteration-count direction fill-mode play-state;

例子：

```
div{
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name:mydh;
  animation-duration:2s;
  animation-timing-function:ease-in;
  animation-iteration-count:3;
  animation-fill-mode:both;
```

```
  animation-delay:1s;
  animation-direction:alternate;
}
@keyframes mydh{
  0%{
    background-color: pink;
    width:200px;
  }
  50%{
    background-color: yellow;
    width:400px
  }
  100%{
    background-color: green;
    width:500px
  }
}
div:hover{
  animation-play-state:paused;
}
```

2017年5月19日　15:09

设为Flex布局以后，子元素的float、clear和vertical-align属性将失效
所有flex的子元素margin不重叠
flex:flex-grow flex-shrink flex-basis; 简写

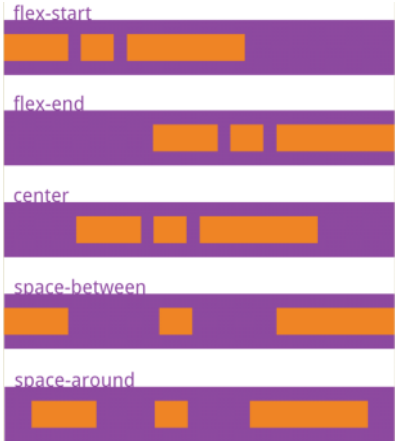
display:flex

flex-direction 项目的排列方向，主轴，项目水平方向都按照主轴排列，换行时才按照纵轴的方向排列
row/row-reverse/column/column-reverse
flex-wrap 如果一条轴线排不下，如何换行，纵轴
nowrap/wrap/wrap-reverse

flex-grow:1;剩余空间分配比例，默认为0，即有剩余也不分配
具体计算方式：
若父元素为1000，三个子元素为width为100，设置grow为1,2,1，则每个分到空白700*1/4,700*2/4,700*1/4
若子元素的权重之和小于1时，如权重之和为0.2,剩余空间的百分之20用来分配，其余不动
flex-shrink:1 默认0,权重需要乘以他自己的宽度或高度
具体计算方式：
若父元素为1000，三个子元素为width为400，设置shrink为1,2,1, 超出了-200，
则每个收缩,400*1/1200*(-200),400*2/1200*(-200),400*1/1200*(-200)
若父元素宽度够，设置该属性没用

flex-basis:40px； 主轴若是水平方向上，该值为宽度，主轴为垂直方向上，为高度；若即设置了宽高，则宽高无效，若为auto，则宽高生效
即当因页面宽度不同，页面内内容宽高度需要变化时，可以用该属性

justify-content 主轴方向上的对齐(没有flex-grow的情况下有效，否则自动填充满)
flex-start/flex-end/center/space-between/space-around;
若主轴从左向右，则效果如下：



align-items 垂直方向上对齐

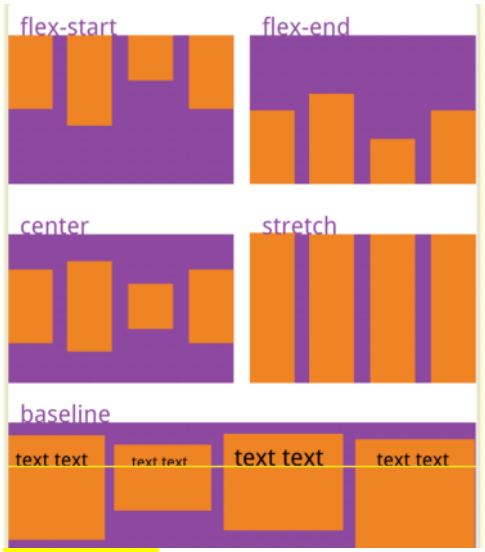
flex-start

flex-end

center

baseline

stretch（默认值）：如果项目未设置高度或设为auto，将占满整个容器的高度(没高度的情况才有效)。



align-content 多根轴的对齐方式，只有一个轴时，无效

flex-start：与交叉轴的起点对齐。

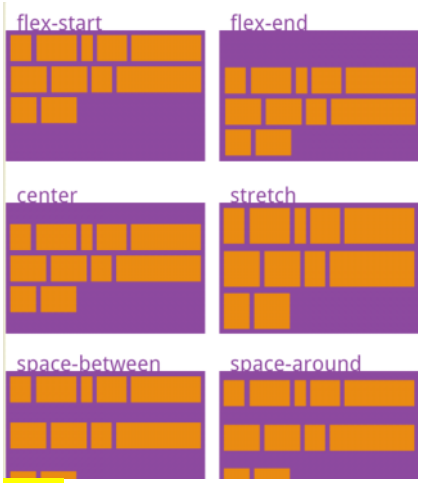
flex-end：与交叉轴的终点对齐。

center：与交叉轴的中点对齐。

space-between：与交叉轴两端对齐，轴线之间的间隔平均分布。

space-around：每根轴线两侧的间隔都相等。所以，轴线之间的间隔比轴线与边框的间隔大一倍。

stretch（默认值）：轴线占满整个交叉轴。



order:-1;改变元素的显示顺序，数值越小，排列越靠前，默认为0，此对计数器没有影响，计数器按html结构

http://www.ruanyifeng.com/blog/2015/07/flex-grammar.html

2017年5月27日 15:44

max-content:元素大到正好所有内容都不换行

```
p{width:max-content;}
```

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Magni, porro!

min-content:元素小到正好没有内容溢出

```
p{width:min-content;}
```

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Magni, porro!

fit-content:元素在小于其父元素的情况下，尽量适应其内容，即被子元素撑开

```
p{width:fit-content;}
```

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Magni, porro!

-webkit-fill-available/fill:宽度或高度跟父元素一样大

```
p{height:-webkit-fill-available;}
```

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Magni, porro!

2017年9月26日 14:13

滤镜效果

filter:blur(20px); 模糊