CSS设计笔记：

内联式 > 嵌入式 > 外部式

但是嵌入式>外部式有一个前提：嵌入式css样式的位置一定在外部式的后面。如右代码编辑器就是这样，<link href="style.css" ...>代码在<style type="text/css">...</style>代码的前面（实际开发中也是这么写的）。

其实总结来说，就是--就近原则（离被设置元素越近优先级别越高）。

# 选择器：

除此之外，还可以用html中的标签来作为样式定义中的”容器“使其有两个功能：1为html的标签，2为样式”容器“。

## **类选择器：**

类选择器在css样式编码中是最常用到的，如右侧代码编辑器中的代码:可以实现为“胆小如鼠”、“勇气”字体设置为红色。

语法：

****.****类选器名称{css样式代码;}

注意：

1、****英文圆点开头****

2、其中****类选器名称****可以任意起名（但不要起中文）

如：

.stress{

color:red;

}

</style>

<p>三年级时，我还是一个<span class="stress">胆小如鼠</span>

·若定义了两个类，则可以通过”/空格“把两个类隔开，例如：<span class=”stress setGreen”>

## ID选择器：

在很多方面，ID选择器都类似于类选择符，但也有一些重要的区别：

1、为标签设置id="ID名称"，而不是class="类名称"。

2、ID选择符的前面是井号****（#）****号，而不是英文圆点****（.）****。

右侧代码编辑器中就是一个ID选择符的完整实例

3：在同一个文件中，ID选择器只能出现一个，且只能使用一次。而class可以出现多个，使用多次

## 子选择器:

还有一个比较有用的选择器****子选择器****，即大于符号(>),用于选择指定标签元素的****第一代子元素。****如右侧代码编辑器中的代码：

.food>li{border:1px solid red;}



## 包含(后代)选择器：

****包含选择器****，即加入空格,用于选择指定标签元素下的****后辈元素。****如右侧代码编辑器中的代码：

.first span{color:red;}

这行代码会使第一段文字内容中的“胆小如鼠”字体颜色变为红色。

请注意这个选择器与子选择器的区别，子选择器（child selector）仅是指它的直接后代，或者你可以理解为作用于子元素的第一代后代。而后代选择器是作用于所有子后代元素。后代选择器通过空格来进行选择，而子选择器是通过“>”进行选择。

### 例子：

如：

上文的



若改代码为：**.food li{border:1px solid red;}**

则有：



### 总结：

“>”作用于元素的第一代后代，****空格****作用于元素的所有后代。

## 通用选择器：

匹配html中所有标签元素，如下使用下面代码使用html中任意标签元素字体颜色全部设置为红色：

\* {color:red;}

## 伪类选择器：

### 链接伪类选择器：

·:link 未访问的链接

·:visited 已访问的链接

·:hover 鼠标移动经过时触发

·:actice 鼠标左键不松手

### 结构（位置）伪类选择器：

·:first-child 选取属于其父元素的首个子元素的指定选择器

·:last-child 选取属于其父元素的最后一个子元素的指定选择器

·:nth-child(n) 匹配属于其父的第N个子元素，不论元素的类型

·nth-last-child(n) 匹配属于其父元素的第N个子元素的每个元素，不论元素的类型，。从 最后一个子元素开始计数，N可以是数字、关键字、公式

它允许给html不存在的标签（标签的某种状态）设置样式，

比如说我们给html中一个标签元素的鼠标滑过的状态来设置字体颜色：

a:hover{color:red;}

****关于伪选择符：****

:hover 可以放在任意的标签上，比如说 p:hover，但是它们的兼容性也是很不好的，

所以现在比较常用的还是 a:hover 的组合。

## 分组选择符：

当你想为html中多个标签元素设置同一个样式时，可以使用分组选择符（，），如下代码为右侧代码编辑器中的h1、span标签同时设置字体颜色为红色：

h1,span{color:red;}

它相当于下面两行代码：

h1{color:red;}

span{color:red;}

## 继承：

CSS的****某些样式****是具有继承性的，那么什么是继承呢？继承是一种规则，它允许样式不仅应用于某个特定html标签元素，而且应用于其后代。比如下面代码：如某种颜色应用于p标签，这个颜色设置不仅应用p标签，还应用于p标签中的所有子元素文本，这里子元素为span标签。

p{color:red;}

<p>三年级时，我还是一个<span>胆小如鼠</span>的小女孩。</p>

可见p中的文本与span中的文本都设置为了红色。但注意有一些css样式是不具有继承性的。如border:1px solid red;

p{border:1px solid red;}

<p>三年级时，我还是一个<span>胆小如鼠</span>的小女孩。</p>

在上面例子中它代码的作用只是给p标签设置了边框为1像素、红色、实心边框线，而对于子元素span是没用起到作用的。

## 特殊性：

有的时候我们为同一个元素设置了不同的CSS样式代码，那么元素会启用哪一个CSS样式呢?我们来看一下面的代码：

p{color:red;}

.first{color:green;}

<p class="first">三年级时，我还是一个<span>胆小如鼠</span>的小女孩。</p>

p和.first都匹配到了p这个标签上，那么会显示哪种颜色呢？green是正确的颜色，那么为什么呢？是因为浏览器是根据权值来判断使用哪种css样式的，权值高的就使用哪种css样式。

下面是权值的规则：

****标签的权值为1，类选择符的权值为10，ID选择符的权值最高为100。****例如下面的代码：

p{color:red;} ****/\*权值为1\*/****

p span{color:green;} ****/\*权值为1+1=2\*/****

.warning{color:white;} ****/\*权值为10\*/****

p span.warning{color:purple;} ****/\*权值为1+1+10=12\*/****

#footer .note p{color:yellow;} ****/\*权值为100+10+1=111\*/****

****注意：还有一个权值比较特殊--继承也有权值但很低，有的文献提出它只有0.1，所以可以理解为继承的权值最低。****

## 层叠：

我们来思考一个问题：如果在html文件中对于同一个元素可以有多个css样式存在并且这多个css样式具有****相同权重值****怎么办？

****层叠****就是在html文件中对于同一个元素可以有多个css样式存在，当有相同权重的样式存在时，会根据这些css样式的前后顺序来决定，处于最后面的css样式会被应用。

如下面代码:

p{color:red;}

p{color:green;}

<p class="first">三年级时，我还是一个<span>胆小如鼠</span>的小女孩。</p>

最后 p 中的文本会设置为green，这个层叠很好理解，理解为后面的样式会覆盖前面的样式。

## 重要性：

我们在做网页代码的时，有些特殊的情况需要为某些样式设置具有最高权值，怎么办？这时候我们可以使用****!important****来解决。

如下代码：

p{color:red****!important****;}

p{color:green;}

<p class="first">三年级时，我还是一个<span>胆小如鼠</span>的小女孩。</p>

这时 p 段落中的文本会显示的red红色。

# 文字排版：

## 文字排版--字体：

我们可以使用css样式为网页中的文字设置字体、字号、颜色等样式属性。

定义在<head></head>中

body{font-family:"宋体";} //字体

body{font-family:"微软雅黑";} //字体

body{font-size:12px;color:#666}//大小、颜色

font-weight:bold; //粗体

pfont-style:italic; //斜体

text-decoration:underline; //下划线

text-decoration:line-through; //删除线, 原价现价中，原价被删除的样子

text-indent:2em; //段落前空两格

line-height:1.5em; //上下两行间距提高

letter-spacing:50px;word-spacing:50px//中文字与中文字之间或者单词与单词之间的间距

text-align:center //居中，对齐

# 元素分类

在CSS中，html中的标签元素大体被分为三种不同的类型：****块状元素****、****内联元素(又叫行内元素)****和****内联块状元素****。

****常用的块状元素有：****

<div>、<p>、<h1>...<h6>、<ol>、<ul>、<dl>、<table>、<address>、<blockquote> 、<form>

****常用的内联元素有：****

<a>、<span>、<br>、<i>、<em>、<strong>、<label>、<q>、<var>、<cite>、<code>

****常用的内联块状元素有：****

<img>、<input>

## 元素分类--块级元素（block）：

什么是块级元素？在html中<div>、 <p>、<h1>、<form>、<ul> 和 <li>就是块级元素。设置display:block就是将元素显示为块级元素。如下代码就是将****内联元素a****转换为****块状元素****，从而使a元素具有****块状元素****特点。

a{display:block;}

****块级元素特点：****

1、每个块级元素都从新的一行开始，并且其后的元素也另起一行。（真霸道，一个块级元素独占一行）

2、元素的高度、宽度、行高以及顶和底边距都可设置。

3、元素宽度在不设置的情况下，是它本身父容器的100%（和父元素的宽度一致），除非设定一个宽度。

## 元素分类--内联元素（inline）：

在html中，<span>、<a>、<label>、 <strong> 和<em>就是典型的****内联元素****（****行内元素****）（inline）元素。当然****块状元素****也可以通过代码display:inline将元素设置为****内联元素****。如下代码就是将****块状元素div****转换为****内联元素****，从而使 div 元素具有****内联元素****特点。

div{

display:inline;

}

......

<div>我要变成内联元素</div>

****内联元素特点：****

1、和其他元素都在一行上；

2、元素的高度、宽度及顶部和底部边距****不可****设置；

3、元素的宽度就是它包含的文字或图片的宽度，不可改变。

## 元素分类--内联块状元素（inline-block）：

****内联块状元素（****inline-block****）****就是同时具备内联元素、块状元素的特点，代码display:inline-block就是将元素设置为内联块状元素。(css2.1新增)，<img>、<input>标签就是这种内联块状标签。

inline-block 元素特点：

1、和其他元素都在一行上；

2、元素的高度、宽度、行高以及顶和底边距都可设置。

提示：下一小节是用视频动画来讲解css中的盒模型。

# css布局模型（传统布局）：

清楚了CSS 盒模型的基本概念、 盒模型类型， 我们就可以深入探讨网页布局的基本模型了。布局模型与盒模型一样都是 CSS 最基本、 最核心的概念。 但布局模型是建立在盒模型基础之上，又不同于我们常说的 CSS 布局样式或 CSS 布局模板。如果说布局模型是本，那么 CSS 布局模板就是末了，是外在的表现形式。   
CSS包含3种基本的布局模型，用英文概括为：Flow、Layer 和 Float。  
在网页中，元素有三种布局模型：  
1、流动模型（Flow）  
2、浮动模型 (Float)  
3、层模型（Layer）

## 流动模型（一）：

先来说一说****流动模型****，流动（Flow）是默认的网页布局模式。也就是说网页在默认状态下的 HTML 网页元素都是根据流动模型来分布网页内容的。

流动布局模型具有2个比较典型的特征：

第一点，****块状元素****都会在所处的****包含元素内****自上而下按顺序垂直延伸分布，因为在默认状态下，块状元素的宽度都为****100%****。实际上，块状元素都会以行的形式占据位置。如右侧代码编辑器中三个块状元素标签(div，h1，p)宽度显示为100%。

## 流动模型（二）

第二点，在流动模型下，****内联元素****都会在所处的包含元素内从左到右水平分布显示。（内联元素可不像块状元素这么霸道，独占一行）

右侧代码编辑器中内联元素标签a、span、em、strong都是内联元素。

## 浮动模型：

块状元素这么霸道都是独占一行，如果现在我们想让两个块状元素并排显示，怎么办呢？不要着急，设置元素浮动就可以实现这一愿望。

任何元素在默认情况下是不能浮动的，但可以用 CSS 定义为浮动，如 div、p、table、img 等元素都可以被定义为浮动。如下代码可以实现两个 div 元素一行显示。

div{

width:200px;

height:200px;

border:2px red solid;

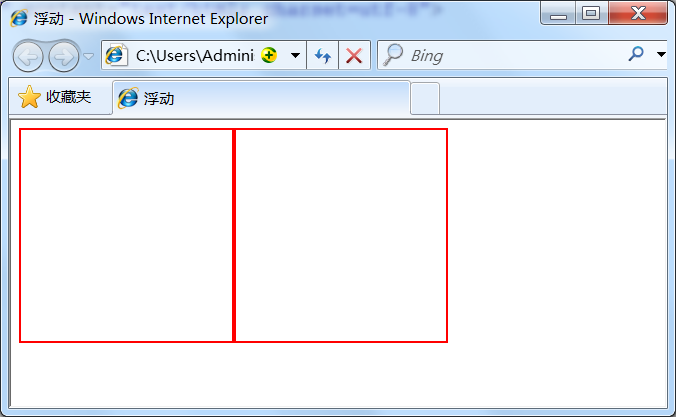
float:left;

}

<div id="div1"></div>

<div id="div2"></div>

效果图

[](http://img.mukewang.com/540e62c60001c56a06760417.jpg)

当然你也可以同时设置两个元素右浮动也可以实现一行显示。

div{

width:200px;

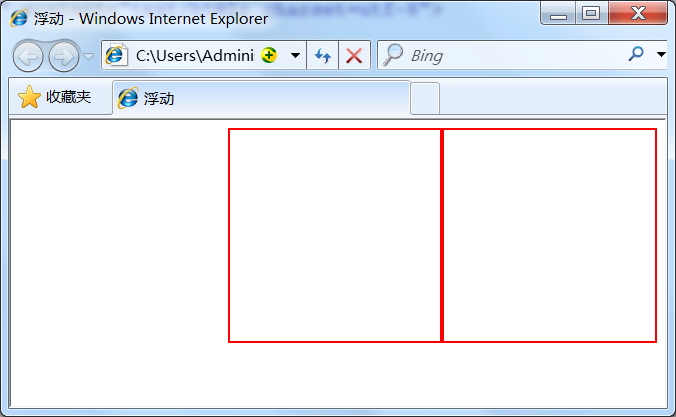
height:200px;

border:2px red solid;

float:right;

}

效果图

[](http://img.mukewang.com/540e632b0001f5f506760417.jpg)

又有小伙伴问了，设置两个元素一左一右可以实现一行显示吗？当然可以：

div{

width:200px;

height:200px;

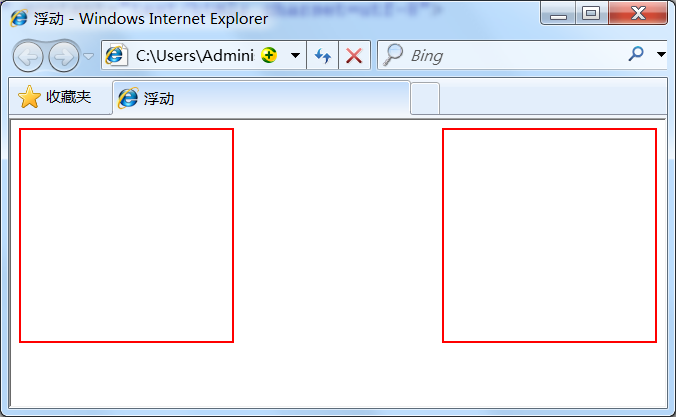
border:2px red solid;

}

#div1{float:left;}

#div2{float:right;}

效果图

[](http://img.mukewang.com/540e63b50001f6a206760417.jpg)

# 层模型：

什么是层布局模型？层布局模型就像是图像软件PhotoShop中非常流行的图层编辑功能一样，每个图层能够精确定位操作，但在网页设计领域，由于网页大小的活动性，层布局没能受到热捧。但是在网页上局部使用层布局还是有其方便之处的。下面我们来学习一下html中的层布局。

如何让html元素在网页中精确定位，就像图像软件PhotoShop中的图层一样可以对每个图层能够精确定位操作。CSS定义了一组定位（positioning）属性来支持层布局模型。

层模型有三种形式：

1、****绝对定位****(position: absolute)

2、****相对定位****(position: relative)

3、****固定定位****(position: fixed)

## 绝对定位(absolute)：

如果想为元素设置层模型中的绝对定位，需要设置****position:absolute****(表示绝对定位)，这条语句的作用将元素从文档流中拖出来，然后使用left、right、top、bottom属性相对于其最接近的一个具有定位属性的父包含块进行绝对定位。如果不存在这样的包含块，则相对于body元素，即相对于****浏览器窗口****。

如下面代码可以实现div元素相对于浏览器窗口向右移动100px，向下移动50px。

div{

width:200px;

height:200px;

border:2px red solid;

position:absolute;

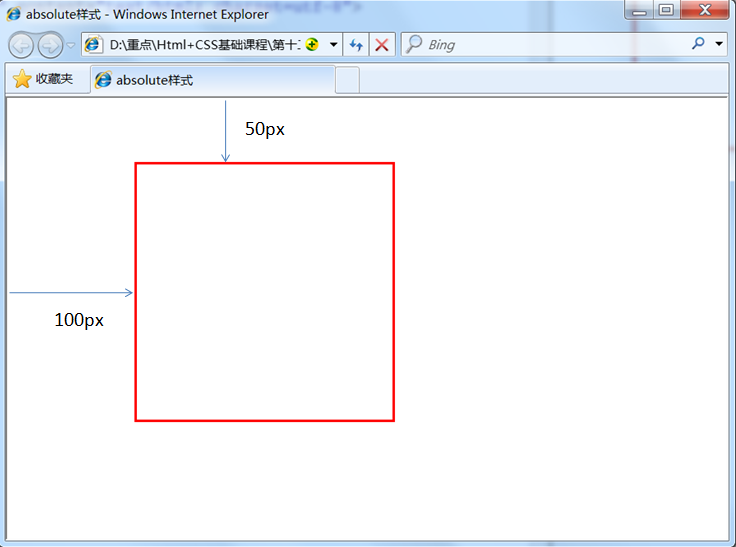
left:100px;

top:50px;

}

<div id="div1"></div>

效果如下：



## 相对定位(relative)：

如果想为元素设置层模型中的相对定位，需要设置position:relative（表示相对定位），它通过left、right、top、bottom属性确定元素在****正常文档流中****的偏移位置。相对定位完成的过程是首先按static(float)方式生成一个元素(并且元素像层一样浮动了起来)，****以前的位置往现在的位置移动，****移动的方向和幅度由left、right、top、bottom属性确定，偏移前的位置保留不动。

如下代码实现相对于以前位置向下移动50px，向右移动100px;

#div1{

width:200px;

height:200px;

border:2px red solid;

position:relative;

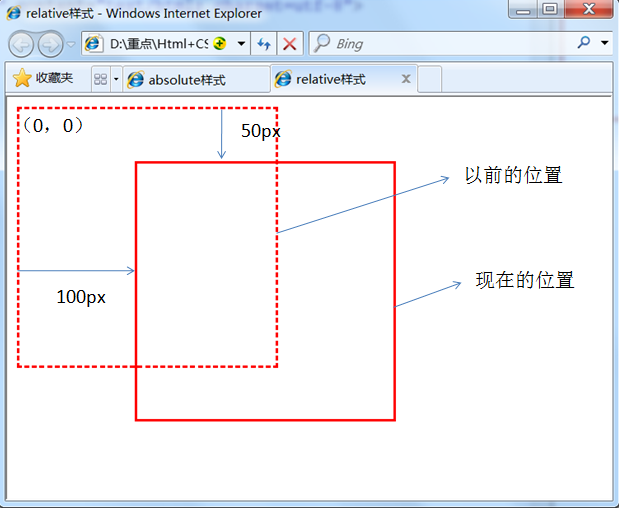
left:100px;

top:50px;

}

<div id="div1"></div>

效果图：



什么叫做“偏移前的位置保留不动”呢？

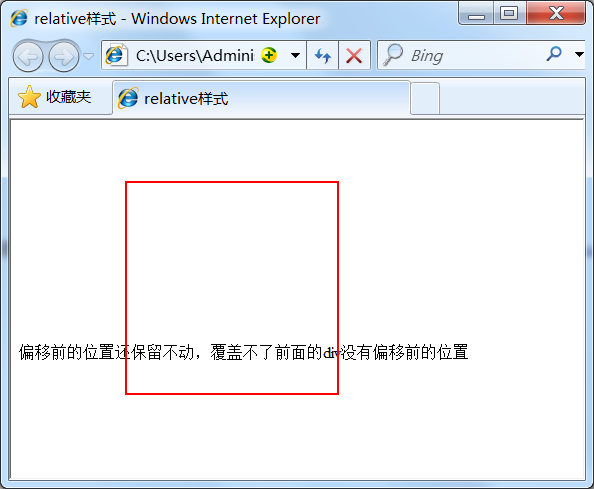
大家可以做一个实验，在右侧代码编辑器的19行div标签的后面加入一个span标签，在标并在span标签中写入一些文字。如下代码：

<body>

<div id="div1"></div><span>偏移前的位置还保留不动，覆盖不了前面的div没有偏移前的位置</span>

</body>

效果图：

[](http://img.mukewang.com/541a4bfc0001abef05940489.jpg)

从效果图中可以明显的看出，虽然div元素相对于以前的位置产生了偏移，但是div元素以前的位置还是保留着，所以后面的span元素是显示在了div元素以前位置的后面。

## 固定定位(fixed)：（一般用于网页流氓广告）

fixed：表示固定定位，与absolute定位类型类似，但它的相对移动的坐标是视图（****屏幕内的网页窗口****）本身。由于视图本身是固定的，它不会随浏览器窗口的滚动条滚动而变化，除非你在屏幕中移动浏览器窗口的屏幕位置，或改变浏览器窗口的显示大小，因此固定定位的元素会始终位于浏览器窗口内视图的某个位置，不会受文档流动影响，这与background-attachment:fixed;属性功能相同。以下代码可以实现相对于****浏览器视图****向右移动100px，向下移动50px。并且拖动滚动条时位置固定不变。

## Relative与Absolute组合使用:

使用position:absolute可以实现被设置元素相对于浏览器（body）设置定位以后，大家有没有想过可不可以相对于其它元素进行定位呢？答案是肯定的，当然可以。使用position:relative来帮忙，但是必须遵守下面规范：

1、参照定位的元素必须是相对定位元素的前辈元素：

<div id="box1"><!--参照定位的元素-->

<div id="box2">相对参照元素进行定位</div><!--相对定位元素-->

</div>

从上面代码可以看出box1是box2的父元素（父元素当然也是前辈元素了）。

2、参照定位的元素必须加入position:relative;

#box1{

width:200px;

height:200px;

position:relative;

}

3、定位元素加入position:absolute，便可以使用top、bottom、left、right来进行偏移定位了。

#box2{

position:absolute;

top:20px;

left:30px;

}

这样box2就可以相对于父元素box1定位了（这里注意参照物就可以不是浏览器了，而可以自由设置了）。

### 注意：

先要有position:relative，后面才能有position:absolute

# 渐变（gradients）：

·线性渐变（Linear Gradients）：上/下/左/右/对角

·径向渐变(Radial Gradients)：由它们的中心定义

## ·线性渐变：

%若有多种颜色渐变，则在框中多加几个颜色

### 默认为：从上到下：

background:linear-gradient(red,blue);

### 向左：

background:linear-gradients(to left,red,blue);

**向上、向右类似**

### 对角：

background:linear-gradient(to bottom,red,blue)

### 带有角度的线性渐变：

background:linear-gradient(120deg,red,blue)

### 带有彩色（手动滑稽）

background:liner-gradient(红橙黄绿青蓝紫)

### 带有透明度：

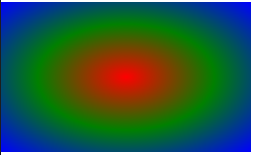
background:liner-gradient(rgba(255.0.0.0),rgba(255.0.0.1))

#rgba(255.0.0.0~1)：前三个表示颜色值，与rgb( , ,)类似，最后则是代表透明度的0~1，越偏0透明度越高

## ·径向渐变：

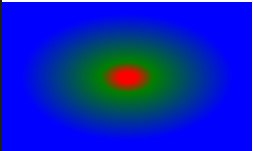
### 颜色结点均匀分布：

background:radial-gradient(red,green,blue)



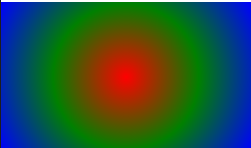
1. 颜色结点不均匀分布：

background:radial-gradient(red5%,green 15%,blue60%);



1. 形状为圆形的径向渐变：

background: radial-gradient(circle, red, green, blue);



# 单、多文本溢出：

## 单纯不显示溢出：

<https://www.cnblogs.com/gopark/p/8875655.html>

多文本溢出时可以设置如下css，其中，“-webkit-line-clamp”的数字表示显示几行

display: -webkit-box;

-webkit-box-orient: vertical;

-webkit-line-clamp: 2;

overflow: hidden;

# 过渡：

它是监听所在的元素的属性的变化，不单单指css上的属性变量的变化

格式：transition: 改变的属性 持续时间 运动曲线 响应时间

·运动曲线有：linear(线性) ease（逐渐变慢） ease-in（逐渐变快） ease-in-out（先快后慢）

·一般是写在原样式中的

·一般是要用hover来改变鼠标经过时的样式

## 2D变形：transform

1. 移动：translate、translateX、translateY

模板：transform: translate(x,y);

·如果用translate(50%)里面的内容是百分比的话，则是以本体宽度的50%来移动。（可以 用来居中）

1. 缩放：scale

模板：transform:scale(比例，比例)，transform:scale(1.2,1.2),即x,y变成原来的一半

或者：transform:scale(比例)，即同时变化

1. 旋转：rotate(deg)，正值为顺时针，负值为逆时针

模板：transform:rotate(90deg)

**3.1旋转中心点：** transform-origin:x位置 y位置写在原样式中

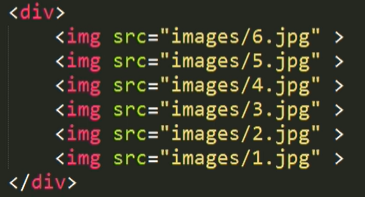
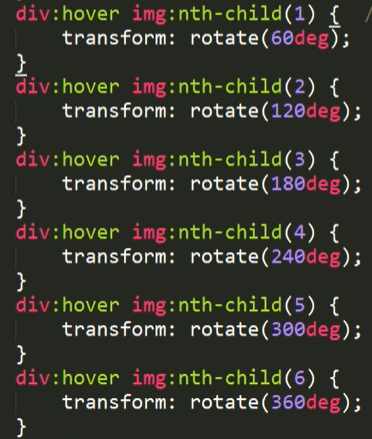
如transform-origin:center center 中心

transform-origin:left top 左上角

transform-origin:left，按左边翻转

或者根据像素：transform-origin:10px 10px

案例：



效果：



1. 倾斜：skew(x,y)单位deg

模板：transform:skew(0,-10deg) 沿着y轴倾斜

## 3D变形：

有X,Y,Z三个坐标系

### 旋转：

rotateX() 沿着X立体旋转

rotateY() 沿着Y立体旋转

rotateZ() 沿着Z立体旋转

但是，现在的图片旋转也只是沿着三个轴的旋转，而不是真正意义上的效果，如，近大远小的效果



（近大远小）

所以需要用到 “透视”

透视：persperctive

·一般是写在父元素上的如<div><a></a></div>中的div上

perspective:1000px 即眼睛（屏幕）到屏幕（里世界）的距离， 越小越明显

相当于，眼睛 叶子 图片， 叶子离越近， 遮着越多

## 移动：

对于translateX 和translateY来说，跟2D是一样的

但是如果是translateZ来说，然后再设置translateZ相当于图片放大.

模型：transform: translate3d(x,y,z) ;x,y可以是px/% 而z只能是px

# 动画animation

由 动画定义 和 动画引用 两个部分组成

## 动画定义：一般写在原元素

**语法格式**

animation: 动画名称 动画时间 运动曲线 何时开始 播放次数 是否反方向

·动画名称：animation-name,是自定义的

·动画时间：animation-duration

·运动时间：animation-timing-function

·何时开始：animation-delay

·播放次数：animation-iteration-count:1/2/3/infinite(无限)

·方向：animation-director:normal/reverse/altnate(先正后反)/altnate-reverse

·规定动画是否停止：animation-play-state:running/paused

**例子：**

animation:go 2s ease 0s 2 reverse

·一般来说只用animation:名字 时间

## 动画引用：

**语法格式：**

@keyframes 动画名称{

from{

}

to{

}

}

### 多组动画：用百分比来写

·百分比可以任意设置，即可以偶0.1% 2%等，但是时间会不同，即动画引用时间的百分比！

@keyframes go{

0%{

}

25%{

}

50%{

}

...

}

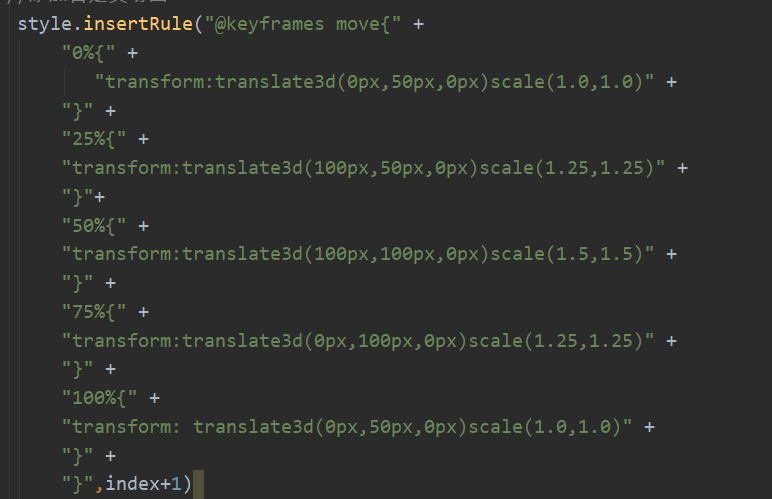
·如果有多组变形，用空格隔开：

transform:translate() rotate()

## JS控制动画：

1、用var style = document.styleSheet[0]获得绑定的css样式表

2、用style.insertRule(“自定义动画”，插入的行数)



·如果用js操作@keyframes，多组变形中间则**不用空格**