第 8 天【css3 新特性】

主要内容

1、圆角
2、阴影
3、背景渐变
4、转换
5、过渡
6、动画
7、绘制特殊图形

学习目标

节数	知识点	要求
第一节	Border-radius	掌握
	Box-shadow	掌握
	Text-shadow	掌握
	线性渐变和射线渐变	了解
第二节	Transform	掌握
第三节	Transition	掌握
第四节	Animation	掌握
第五节	绘制特殊图形	了解

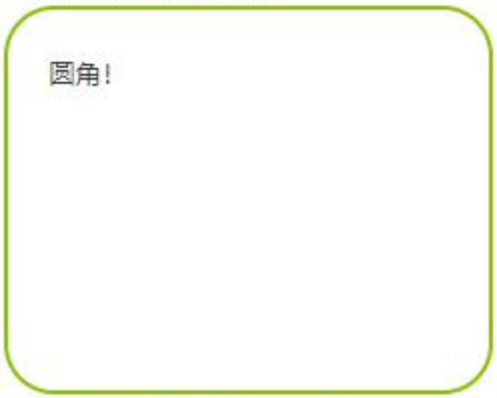
一、圆角

1、定义

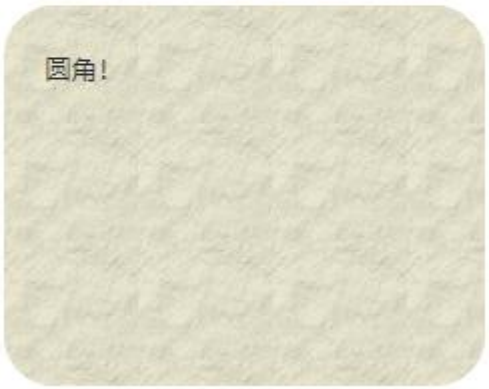
使用 CSS3 border-radius 属性，你可以给任何元素制作 "圆角"。

1. 指定背景颜色的元素圆角：

2. 指定边框的元素圆角：



3. 指定背景图片的元素圆角：



2、语法

CSS3 border-radius - 指定每个圆角

如果你在 border-radius 属性中只指定一个值，那么将生成 4 个 圆角。

但是，如果你要在四个角上一一指定，可以使用以下规则：

（1）四个值: 第一个值为左上角，第二个值为右上角，第三个值为右下角，第四个值为左下角。

（2）三个值: 第一个值为左上角，第二个值为右上角和左下角，第三个值为右下角

（3）两个值: 第一个值为左上角与右下角，第二个值为右上角与左下角

（4）一个值： 四个圆角值相同

3、实例

```
<div class="rcorners1"></div>

<div class="rcorners2"></div>

<div class="rcorners3"></div>
```

```
.rcorners1 {
   border-radius: 15px 50px 30px 5px;
   background: #8AC007;
   padding: 20px;
   width: 200px;
   height: 150px;
}
.rcorners2 {
   border-radius: 15px 50px 30px;
   background: #8AC007;
```

```
    padding: 20px;
    width: 200px;
    height: 150px;
}
.rcorners3 {
    border-radius: 15px 50px;
    background: #8AC007;
    padding: 20px;
    width: 200px;
    height: 150px;
}
```

效果如下：

1. 四个值 - border-radius: 15px 50px 30px 5

2. 三个值 - border-radius: 15px 50px 30px:

3. 两个值 - border-radius: 15px 50px:

二、阴影

1、盒阴影

1.1 定义

box-shadow 向框添加一个或多个阴影。

1.2 语法

```
box-shadow: h-shadow v-shadow blur spread color inset;
```

值	描述
h-shadow	必需。水平阴影的位置。允许负值。
v-shadow	必需。垂直阴影的位置。允许负值。
blur	可选。模糊距离。
spread	可选。阴影的尺寸。
color	可选。阴影的颜色。请参阅 CSS 颜色值。
inset	可选。将外部阴影 (outset) 改为内部阴影。

1.3 实例

1.3.1 实例 1

给阴影添加颜色

```
.box{
        width: 200px;
        height: 200px;
        background-color: #8ac007;
        margin: 50px;
        box-shadow: 10px 10px green;
    }
```

效果如下：

1.3.2 实例 2

给阴影添加一个模糊效果

```
.box{
        width: 200px;
        height: 200px;
        background-color: #8ac007;
        margin: 50px;
        box-shadow: 10px 10px 5px green;
    }
```

效果如下：

1.3.3 实例 3

三个方向的阴影效果

```
.box{
        width: 200px;
        height: 200px;
        background-color: #8ac007;
        margin: 50px;
        box-shadow: 0 10px 30px rgba(0,0,0,.5);
    }
```

效果如下：

1.3.4 实例 4

内阴影效果

```
.box{
        width: 200px;
        height: 200px;
        background-color: #8ac007;
```

```
        margin: 50px;
        box-shadow: 0 0 30px rgba(0,0,0,.5) inset;
    }
```

效果如下：

2、字阴影

2.1 定义

text-shadow 属性向文本设置阴影。

2.2 语法

```
text-shadow: h-shadow v-shadow blur color;
```

值	描述
h-shadow	必需。水平阴影的位置。允许负值。
v-shadow	必需。垂直阴影的位置。允许负值。
blur	可选。模糊的距离。
color	可选。阴影的颜色。

2.3 实例

```
h1{
font-size: 30px;
text-shadow: 5px 5px 5px #FF0000;
}
```

三、背景渐变

3.1 定义

CSS3 渐变（gradients）可以让你在两个或多个指定的颜色之间显示平稳的过渡。

以前，你必须使用图像来实现这些效果。但是，通过使用 CSS3 渐变（gradients），你可以减少下载的时间和宽带的使用。此外，渐变效果的元素在放大时看起来效果更好，因为渐变（gradient）是由浏览器生成的。

CSS3 定义了两种类型的渐变（gradients）：

（1）线性渐变（Linear Gradients）- 向下/向上/向左/向右/对角方向

（2）径向渐变（Radial Gradients）- 由它们的中心定义

线性渐变相关属性：**background-image。**

3.2 线性渐变

3.2.1 定义

为了创建一个线性渐变，你必须至少定义两种颜色结点。颜色结点即你想要呈现平稳过渡的颜色。同时，你也可以设置一个起点和一个方向（或一个角度）。

3.2.2 语法

```
Background-image:
```

```
linear-gradient(direction, color-stop1, color-stop2, ...);
```

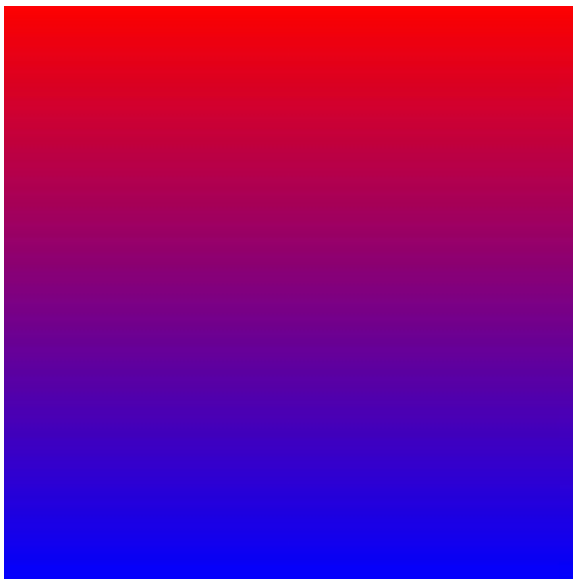
3.2.3 实例

3.2.3.1 实例 1

线性渐变 - 从上到下（默认情况下）

下面的实例演示了从顶部开始的线性渐变。起点是红色，慢慢过渡到蓝色：

```
#grad {
width: 300px;
height: 300px;
background: linear-gradient(red, blue);
}
```



3.2.3.2 实例 2

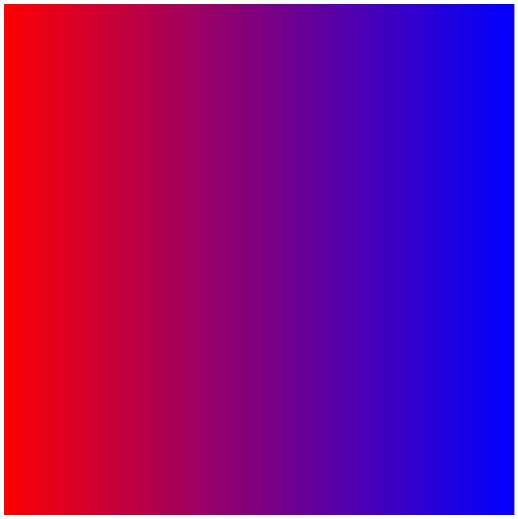
线性渐变 - 从左到右

下面的实例演示了从左边开始的线性渐变。起点是红色，慢慢过渡到蓝色：

```
#grad {
width: 300px;
height: 300px;
```

```
background:linear-gradient(to right, red , blue);

}
```



3.2.3.3 实例 3

线性渐变 - 对角

你可以通过指定水平和垂直的起始位置来制作一个对角渐变。

下面的实例演示了从左上角开始（到右下角）的线性渐变。起点是红色，慢慢过渡到蓝色：

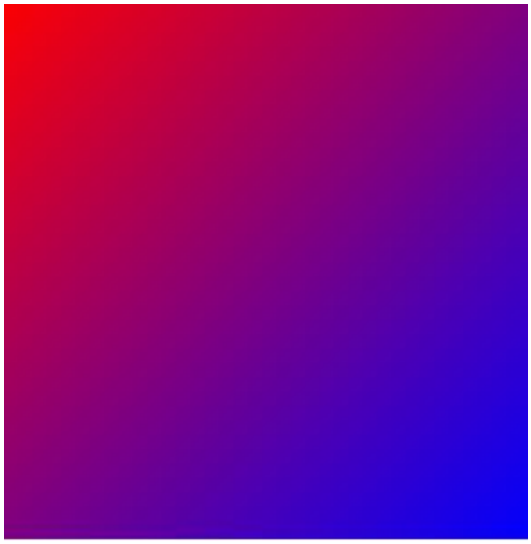
```
#grad {

  width: 300px;

  height: 300px;

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

}
```



3.2.4 使用角度

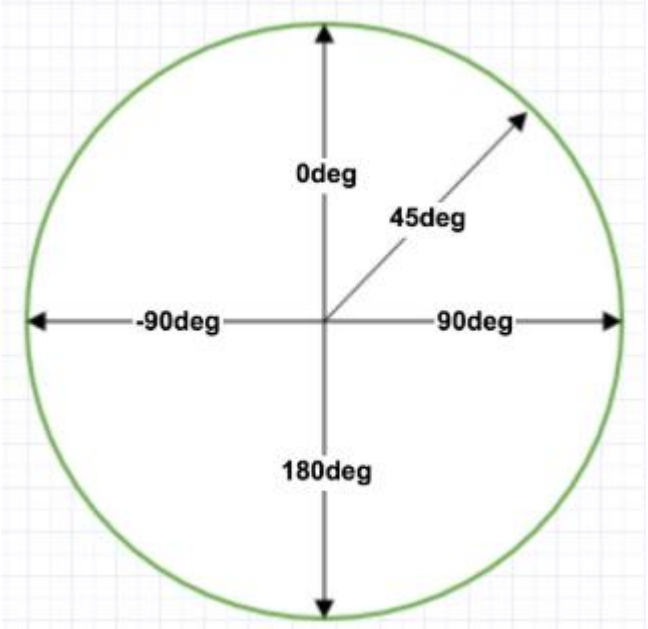
3.2.4.1 定义

如果你想要在渐变的方向上做更多的控制 ,你可以定义一个角度 ,而不用预定义方向(to bottom、to top、to right、to left、to bottom right，等等）。

3.2.4.2 语法

```
Background-image: linear-gradient(angle, color-stop1, color-stop2);
```

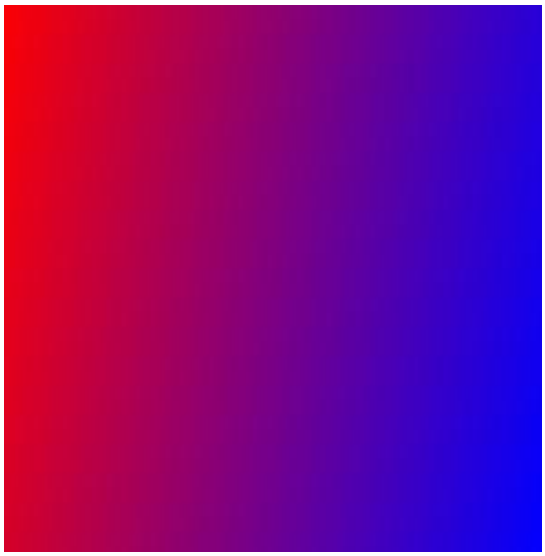
角度是指水平线和渐变线之间的角度，逆时针方向计算。换句话说，0deg 将创建一个从下到上的渐变，90deg 将创建一个从左到右的渐变。



3.2.4.3 实例

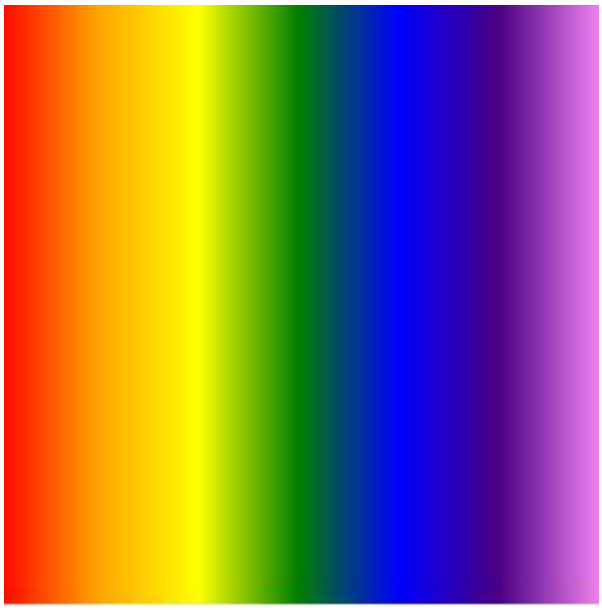
【实例 1】

```
#grad {
   width: 300px;
   height: 300px;
   background: linear-gradient(100deg, red, blue);
}
```



【实例 2】

```
#grad {
width: 300px;
height: 300px;
background:linear-gradient(toright,red,orange,yellow,green,blue,indigo,
violet);
}
```



3.3 射线渐变

3.3.1 定义

径向渐变由它的中心定义。

为了创建一个径向渐变，你也必须至少定义两种颜色结点。颜色结点即你想要呈现平稳过渡的颜色。同时，你也可以指定渐变的中心、形状（圆形或椭圆形）、大小。默认情况下，渐变的中心是 center（表示在中心点），渐变的形状是 ellipse（表示椭圆形），渐变的大小是 farthest-corner（表示到最远的角落）。

3.3.2 语法

```
Background-image: radial-gradient(center, shape size, start-color, ..., last-color);
```

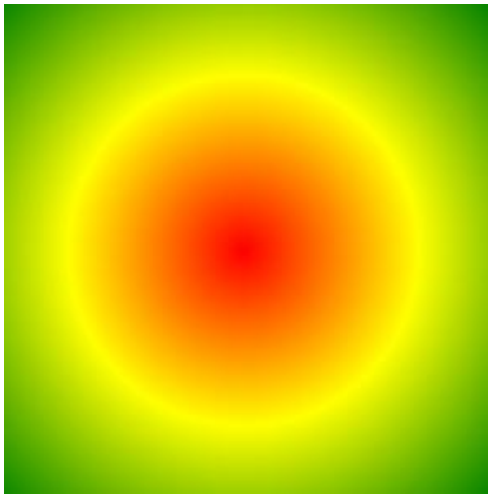
3.3.3 实例

3.3.3.1 实例 1

径向渐变 - 颜色结点均匀分布（默认情况下）

```
#grad {
  width: 300px;
  height: 300px;
```

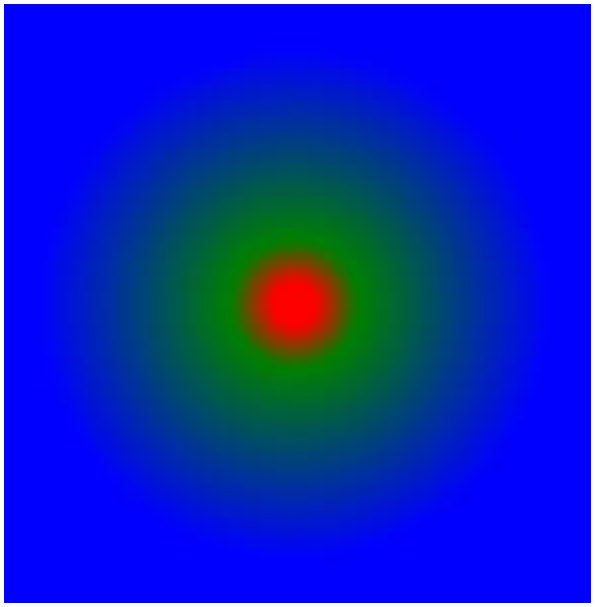
```
    background: radial-gradient(red, yellow, green);
}
```



3.3.3.2 实例 2

径向渐变 - 颜色结点不均匀分布

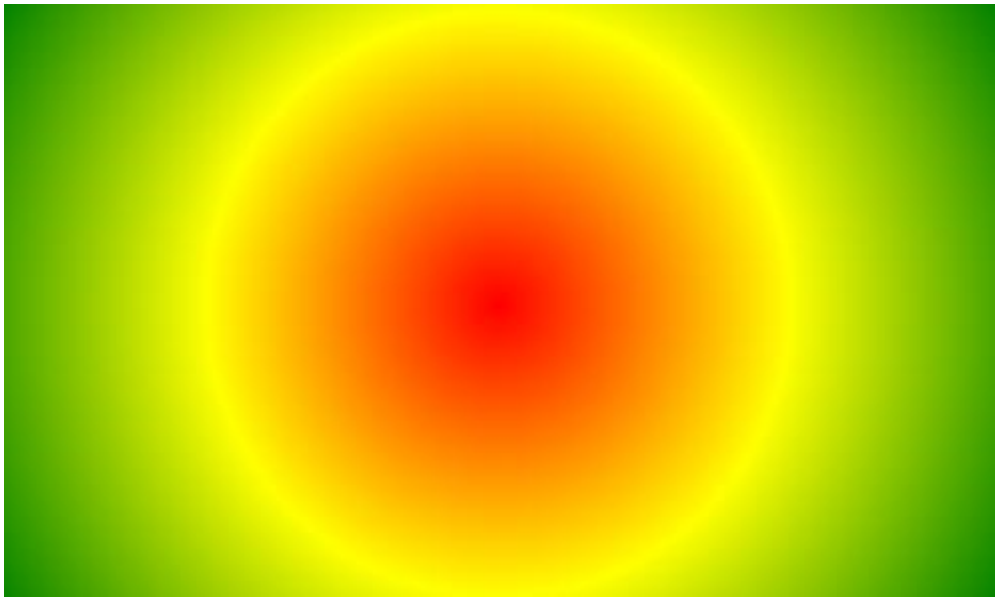
```
#grad {
    width: 300px;
    height: 300px;
    background: radial-gradient(red 5%, green 15%, blue 60%);
}
```



3.3.3.3 实例 3

shape 参数定义了形状。它可以是值 circle 或 ellipse。其中 ,circle 表示圆形 ,ellipse 表示椭圆形。默认值是 ellipse。

```
#grad {
   width: 500px;
   height: 300px;
   background: radial-gradient(circle, red, yellow, green);
}
```



四、转换

4.1 定义

转换的效果是让某个元素改变形状，大小和位置。

transform 属性向元素应用 2D 或 3D 转换。该属性允许我们对元素进行旋转、缩放、移动或倾斜。

4.2 语法

```
transform: none|transform-functions;
```

值	描述
none	定义不进行转换。
translate(*x*,*y*)	定义 2D 转换。
translate3d(*x*,*y*,*z*)	定义 3D 转换。
translateX(*x*)	定义转换，只是用 X 轴的值。
translateY(*y*)	定义转换，只是用 Y 轴的值。
translateZ(*z*)	定义 3D 转换，只是用 Z 轴的值。
scale(*x*,*y*)	定义 2D 缩放转换。
scale3d(*x*,*y*,*z*)	定义 3D 缩放转换。
scaleX(*x*)	通过设置 X 轴的值来定义缩放转换。
scaleY(*y*)	通过设置 Y 轴的值来定义缩放转换。
scaleZ(*z*)	通过设置 Z 轴的值来定义 3D 缩放转换。
rotate(*angle*)	定义 2D 旋转，在参数中规定角度。
rotate3d(*x*,*y*,*z*,*angle*)	定义 3D 旋转。
rotateX(*angle*)	定义沿着 X 轴的 3D 旋转。
rotateY(*angle*)	定义沿着 Y 轴的 3D 旋转。
rotateZ(*angle*)	定义沿着 Z 轴的 3D 旋转。
skew(*x-angle*,*y-angle*)	定义沿着 X 和 Y 轴的 2D 倾斜转换。
skewX(*angle*)	定义沿着 X 轴的 2D 倾斜转换。
skewY(*angle*)	定义沿着 Y 轴的 2D 倾斜转换。

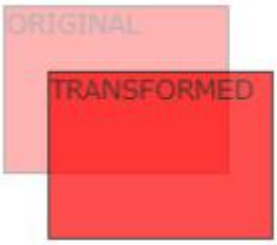
4.3translate 方法

4.3.1 定义

translate()方法，根据左(X 轴)和顶部(Y 轴)位置给定的参数，从当前元素位置移动。

4.3.2 实例

```
.box {
   width: 100px;
   height: 80px;
   background-color: rgba(255,0,0,.8);
   transform: translate(50px,100px);
 }
```



translate 值（50px，100px）是从左边元素移动 50 个像素，并从顶部移动 100 像素。

4.4rotate 方法

4.4.1 定义

rotate()方法，在一个给定度数顺时针旋转的元素。负值是允许的，这样是元素逆时针旋转。

4.4.2 实例

```
.box {
   width: 100px;
   height: 80px;
   background-color: rgba(255,0,0,.8);
   transform: rotate(30deg);}
```

rotate 值（30deg）元素顺时针旋转 30 度。

4.5scale 方法

4.5.1 定义

scale()方法，该元素增加或减少的大小，取决于宽度（X 轴）和高度（Y 轴）的参数：

4.5.2 实例

```
.box {
   width: 100px;
   height: 80px;
   background-color: rgba(255,0,0,.8);
   transform: scale(2,3);
}
```

scale（2,3）转变宽度为原来的大小的 2 倍，和其原始大小 3 倍的高度。

4.6skew 方法

4.6.1 定义

```
transform:skew(<angle> [,<angle>]);
```

包含两个参数值，分别表示 X 轴和 Y 轴倾斜的角度，如果第二个参数为空，则默认为 0，参数为负表示向相反方向倾斜。

（1）skewX(<angle>);表示只在 X 轴(水平方向)倾斜。

（2）skewY(<angle>);表示只在 Y 轴(垂直方向)倾斜。

4.6.2 实例

```
.box {
    width: 100px;
    height: 80px;
    background-color: rgba(255,0,0,.8);
    transform: skew(30deg,20deg);
}
```

4.7 3D 转换

4.7.1 定义

CSS3 允许您使用 3D 转换来对元素进行格式化。

在本章中，您将学到其中的一些 3D 转换方法：

（1）rotateX()

（2）rotateY()

4.7.2rotateX 方法

rotateX()方法，围绕其在一个给定度数 X 轴旋转的元素。

```
.box {
    width: 100px;
    height: 80px;
    background-color: rgba(255,0,0,.8);
    transform: rotateX(120deg);
}
```

4.7.3rotateY 方法

rotateY()方法，围绕其在一个给定度数 Y 轴旋转的元素。

```
.box {
    width: 100px;
    height: 80px;
    background-color: rgba(255,0,0,.8);
    transform: rotateY(130deg);
}
```

五、过渡

5.1 定义

CSS3 过渡是元素从一种样式逐渐改变为另一种的效果。

通过过渡 transition，可以让 web 前端开发人员不需要 javascript 就可以实现简单的动画交互效果。

5.2 语法

过渡 transition 是一个复合属性，包括 transition-property、transition-duration、transition-timing-function、transition-delay 这四个子属性。通过这四个子属性的配合来完成一个完整的过渡效果。

属性	描述
transition	简写属性，用于在一个属性中设置四个过渡属性。
transition-property	规定应用过渡的 CSS 属性的名称。
transition-duration	定义过渡效果花费的时间。默认是 0。
transition-timing-function	规定过渡效果的时间曲线。默认是 "ease"。
transition-delay	规定过渡效果何时开始。默认是 0。

5.2.1 过渡属性

1、transition-property

值: none | all | <transition-property>[,<transition-property>]*

初始值: all

应用于: 所有元素

继承性: 无

```
none: 没有指定任何样式
all: 默认值，表示指定元素所有支持transition-property属性的样式
<transition-property>: 可过渡的样式，可用逗号分开写多个样式
```

2、可过渡的样式

不是所有的 CSS 样式值都可以过渡，只有具有中间值的属性才具备过渡效果

1）取值为颜色

2）取值为数值

3）阴影（box-shadow,text-shadow）

4）转换（transform）

5）背景渐变（gradient）

5.2.2 过渡持续时间

transition-duration

该属性的单位是秒 s 或毫秒 ms

初始值: 0s

应用于: 所有元素

继承性: 无

[注意]该属性不能为负值

[注意]若该属性为 0s 则为默认值，若为 0 则为无效值。所以必须带单位

[注意]该值为单值时，即所有过渡属性都对应同样时间；该值为多值时，过渡属性按照顺序对应持续时间

5.2.3 过渡时间函数

transition-timing-function

过渡时间函数用于定义元素过渡属性随时间变化的过渡速度变化效果

值: <timing-function>[,<timing-function>]*

初始值: ease

应用于: 所有元素

继承性: 无

5.2.4 过渡延迟时间

transition-delay

该属性定义元素属性延迟多少时间后开始过渡效果，该属性的单位是秒 s 或毫秒 ms

值: <time>[,<time>]*

初始值: 0s

应用于: 所有元素

继承性: 无

[注意]该属性若为负值，无延迟效果，但过渡元素的起始值将从 0 变成设定值(设定值=延迟时间+持续时间)。若该设定值小于等于 0，则无过渡效果；若该设定值大于 0，则过渡元素从该设定值开始完成剩余的过渡效果

[注意]若该属性为 0s 则为默认值，若为 0 则为无效值。所以必须带单位

[注意]该值为单值时，即所有过渡属性都对应同样时间；该值为多值时，过渡属性按照顺序对应持续时间

取值：ease|ease-in|ease-out|ease-in-out

5.3 复合属性

过渡 transition 的这四个子属性只有<transition-duration>是必需值且不能为 0。其中，<transition-duration>和<transition-delay>都是时间。当两个时间同时出现时，第一个是<transition-duration>，第二个是<transition-delay>；当只有一个时间时，它是<transition-duration>，而<transition-delay>为默认值 0

```
transition: <transition-property> || <transition-duration> || <transition-timing-function> ||
<transition-delay>
```

[注意]transition 的这四个子属性之间不能用逗号隔开，只能用空格隔开。

5.4 实例

5.4.1 实例 1

//鼠标移动到元素上，会出现宽度变化效果

```
.box {
    width: 100px;
    height: 100px;
    background-color: rgba(255,0,0,.8);
    transition-property: all;
    transition-duration: 3s;
}
.box:hover{
    width: 500px;
}
```

5.4.2 实例 2

```
.box {
    width: 100px;
    height: 100px;
    border-radius: 5px;
    background-color: #D3E399;
    transition: all 2s;
}
.box:hover{
    background-color: #1ec7e6;
    transform: rotate(45deg) scale(1.5);
}
```

六、动画

6.1 概念

动画是使元素从一种样式逐渐变化为另一种样式的效果。

您可以改变任意多的样式任意多的次数。

请用百分比来规定变化发生的时间，或用关键词 "from" 和 "to"，等同于 0% 和 100%。

0% 是动画的开始，100% 是动画的完成。

为了得到最佳的浏览器支持，您应该始终定义 0% 和 100% 选择器。

animation 比较类似于 flash 中的逐帧动画，逐帧动画就像电影的播放一样，表现非常细腻并且有非常大的灵活性。然而 transition 只是指定了开始和结束态，整个动画的过程也是由特定的函数控制。学习过 flash 的同学知道，这种逐帧动画是由关键帧组成，很多个关键帧连续的播放就组成了动画，在 CSS3 中是由属性 keyframes 来完成逐帧动画的。

6.2@keyframes

6.2.1 定义

使用@keyframes 规则，你可以创建动画。

创建动画是通过逐步改变从一个 CSS 样式设定到另一个。

在动画过程中，您可以更改 CSS 样式的设定多次。

指定的变化时发生时使用%，或关键字"from"和"to"，这是和 0%到 100%相同。

0%是开头动画，100%是当动画完成。

为了获得最佳的浏览器支持，您应该始终定义为 0%和 100%的选择器。

6.2.2 语法

```
@keyframes  name  {
from|0%{
css 样式
}
percent{
css 样式
}
to|100%{
css 样式
}
}
```

name：动画名称，开发人员自己命名；

percent：为百分比值，可以添加多个百分比值；

6.3animation-name

它是用来设置动画的名称，可以同时赋值多个动画名称，用,隔开：

```
.animation{

   animation-name: name1,name2,...;

}
```

6.4animation-duration

它是用来设置动画的持续时间，单位为 s 或者 ms，默认值为 0：

```
.animation{

   animation-duration: time1,time2,...;

}
```

6.5animation-timing-function

它是来设置动画效果的速率

ease：逐渐变慢（默认）

linear：匀速

ease-in：加速

ease-out：减速

ease-in-out：先加速后减速，

```
.animation{

animation-timing-function: ease | linear | ease-in | ease-out | ease-in-out;

}
```

6.6animation-delay

它是来设置动画的开始时间，单位是 s 或者 ms，默认值为 0：

```
.animation{

   animation-delay: time1,time2,...;

}
```

6.7animation-iteration-count

它是来设置动画循环的次数，默认为 1，infinite 为无限次数的循环

```
.animation{

   animation-iteration-count:infinite | number;

}
```

6.8animation-direction

它是来设置动画播放的方向，默认值为 normal 表示向前播放，alternate 代表动画播放在第偶数次向前播放，第奇数次向反方向播放：

```
.animation{

   animation-direction: normal | alternate;
```

```
}
```

6.8animation-play-state

它主要是来控制动画的播放状态：running 代表播放，而 paused 代表停止播放，running 为默认值：

```
.animation{
    animation-play-state:running | paused ;
}
```

6.9animation

是 animation-name、animation-duration、animation-timing-function、animation-delay、animation-iteration-count、animation-direction 的简写：

```
animation: name duration timing-function delay iteration-count direction ;
```

6.10 实例

6.10.1 切换背景颜色

```
<div class="animation"></div>
.animation{
    width: 300px;
    height: 300px;
    background-color: red;
    animation: anima 3s linear infinite;
}
.animation:hover{
```

```
    animation-play-state: paused;
}
@keyframes anima {
0%{
    background-color: red;
}
50%{
    background-color: green;
}
100%{
    background-color: blueviolet;
}
}
```

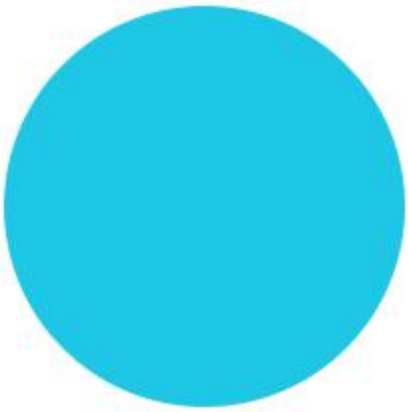
七、绘制特殊图形

7.1 绘制圆形

利用圆角绘制圆

```
.box{
    width: 200px;
    height: 200px;
    background-color: #1EC7E6;
    border-radius: 50%;
```

```
}
```



7.2 绘制三角形

1. 首先需要有个元素作为三角的容器

```
<div class="box"></div>
```

2. 制作三角型使用的是 border 属性，内容区宽高值为 0

```
.box{
    width:0;
    height: 0;
    border-top:50px solid red;
    border-left:50px solid blue;
    border-right:50px solid orange;
    border-bottom:50px solid green;
}
```

7.3 绘制梯形

绘制三角型时宽和高都是 0 像素，给它加 100 的宽度看看效果。这样是不是我们只取下面的红色的区域就构成了一个梯形？

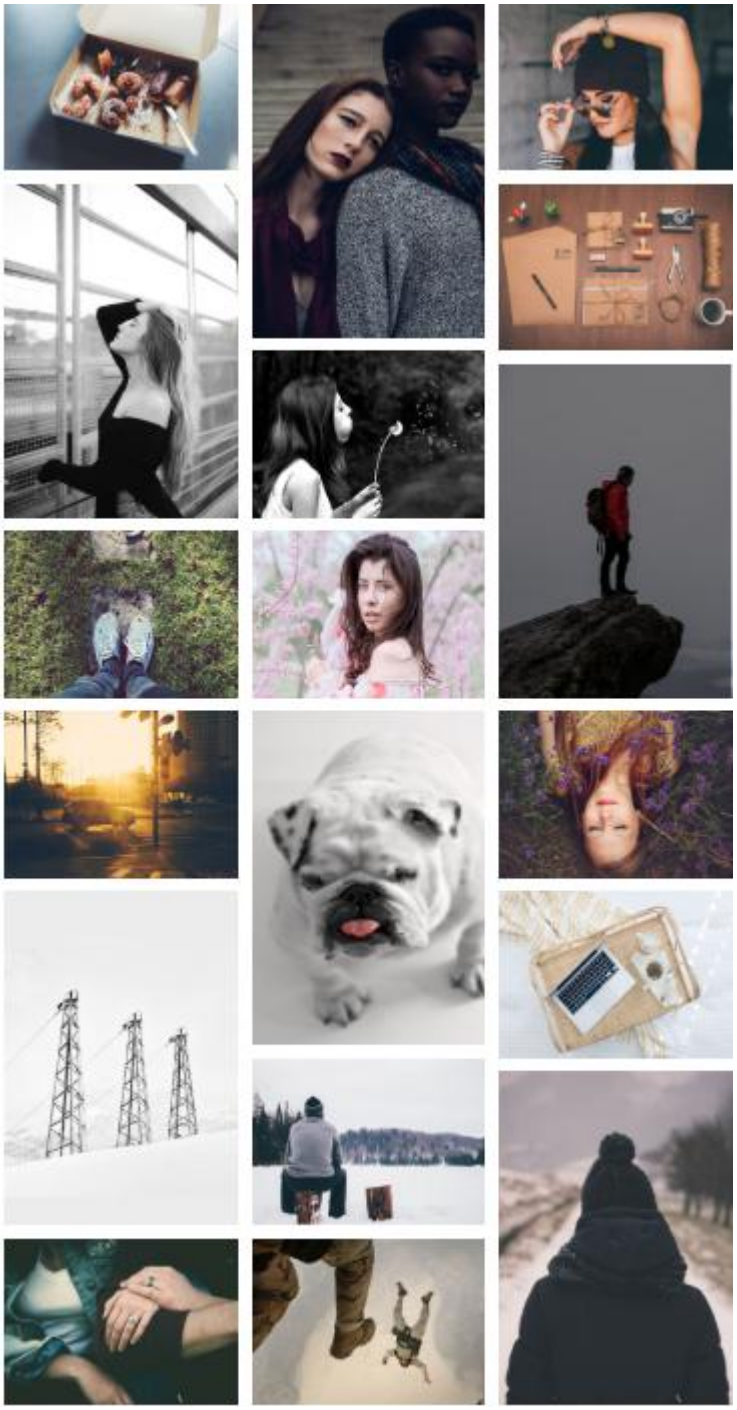
这个红色区域实际上是底部的边 border-bottom，所以梯形的高度是底部边的宽度，梯形也是借助 border 属性完成的。

```
.box{
   width: 100px;
   height: 0;
   border-bottom: 80px solid red;
   border-left: 50px solid transparent;
   border-right: 50px solid transparent;}
```

八、作业

8.1 照片墙

8.2 过渡练习



8.3 图片缩放