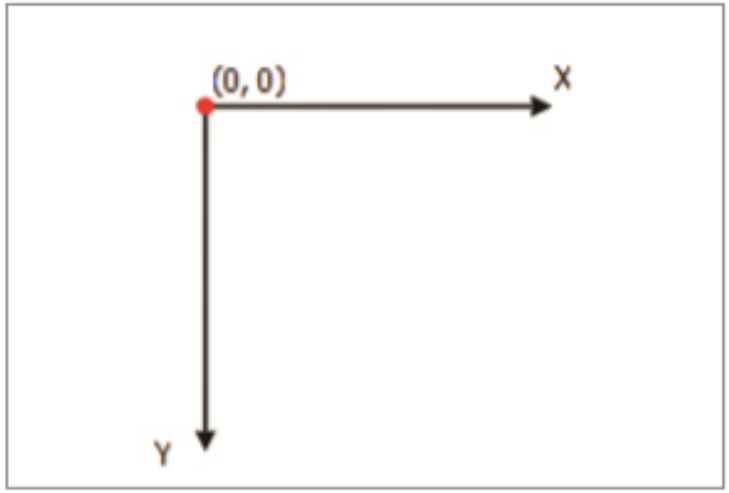
14 css와 애니메이션

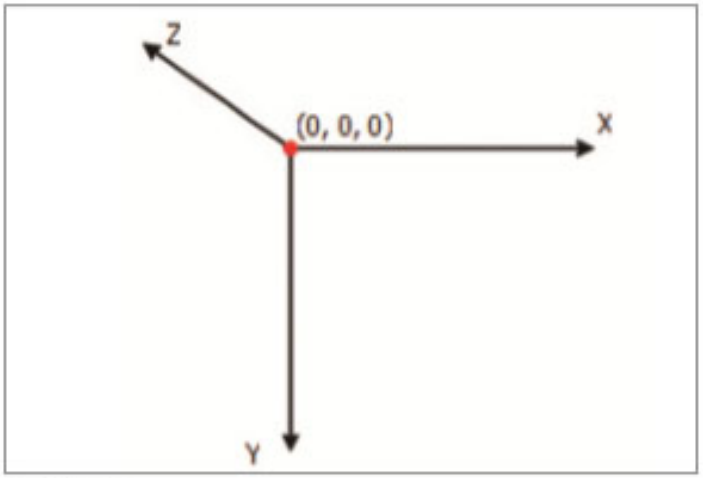
변형(transform)

2차원 좌표계와 회전



2차원 좌표계

3차원 좌표계와 회전



3차원 좌표계

변형(transform)이란

웹 요소의 위치를 옮기거나 크기를 조절하고 회전, 왜곡시킴

transform : <변형 함수>

브라우저 접두사를 붙여야 한다.

- **transform: <변형함수>**
- **-webkit-transform:<변형 함수>**
- **-ms-transform:<변형 함수>**

2차원 변형 함수

- 2차원 변형에서 사용하는 변형함수는 아래와 같다.

변형 함수	설명
translate(tx,ty)	지정한 크기만큼 x축과 y축으로 이동한다
translateX(tx)	지정한 크기만큼 x축으로 이동한다
translateY(ty)	지정한 크기만큼 y축으로 이동한다
scale(sx,sy)	지정한 크기만큼 x축과 y축으로 확대/축소한다
scaleX(sx)	지정한 크기만큼 x축으로 확대/축소한다
scaleY(sy)	지정한 크기만큼 y축으로 확대/축소한다
rotate(각도)	지정한 각도만큼 회전한다
skew(ax,ay)	지정한 각도만큼 x축과 y축으로 왜곡한다
skewX(ax)	지정한 각도만큼 x축으로 왜곡한다
skewY(ay)	지정한 각도만큼 y축으로 왜곡한다

3차원 변형 함수

- 3차원 변형에서 사용하는 변형함수는 아래와 같다.

변형 함수	설명
matrix3d(n [, n])	4 * 4 행렬을 이용해 이동과 확대/축소, 회전 등의 변환을 지정한다
translate3d(tx, ty, tz)	지정한 크기만큼 x축과 y축, z축으로 이동한다
translate(tz)	지정한 크기만큼 z축으로 이동한다
scale3d(sx, sy, sz)	지정한 크기만큼 x축과 y축, z축으로 확대/축소한다
scaleZ(sz)	지정한 크기만큼 z축으로 확대/축소한다
rotate3d(rx,ry,rz,각도)	지정한 각도만큼 회전한다
rotateX(각도)	지정한 각도만큼 x축으로 회전한다
rotateY(각도)	지정한 각도만큼 y축으로 회전한다
rotateZ(각도)	지정한 각도만큼 z축으로 회전한다
perspective(길이)	입체적으로 보일 수 있는 깊이 값을 지정한다.

translate – 요소 이동시키기

- x축과 y축, z축으로 이동할 거리를 지정해서 요소 이동.

- transform:translate(tx, ty)
- transform:translate3d(tx, ty, tz)
- transform:translateX(tx)
- transform:translateY(ty)
- transform:translateZ(tz)

translateX(tx)

- x축 방향, 즉 가로 방향으로 tx만큼 이동한다.
- 양수: 오른쪽으로, 음수 : 왼쪽으로

```
<style>
 .translatex{
      transform:translateX(30px);
      -webkit-transform:translateX(30px);
      -ms-transform:translateX(30px);
  }
</style>

<img src="woman.png" class="translatex">
```

translateY(ty)

- y축 방향, 즉 세로 방향으로 ty만큼 이동한다.
- 양수: 아래쪽으로, 음수 : 위쪽으로

```
<style>
 .translatey{
      transform:translateY(20px);
      -webkit-transform:translateY(20px);
      -ms-transform:translateY(20px);
  }
</style>

<img src="woman.png" class="translatey">
```

translateZ(tz)

- z축 방향, 즉 앞뒤 방향으로 tz만큼 이동한다.
- 양수: 앞쪽으로, 음수 : 뒷쪽으로

```
<style>
 .translatez{
      transform:translateY(-40px);
      -webkit-transform:translateY(-40px);
      -ms-transform:translateY(-40px);
  }
</style>

<img src="woman.png" class="translatez">
```

translate3d(tx, ty, tz)

수평과 수직, 그리고 z축 방향 이동을 한꺼번에 지정한다.

```
<style>
.translatexyz {
     transform:translate3d(20px, 30px,40px);
    -webkit-transform:translate3d(20px, 30px, 40px);
    -ms-transform:translate3d(20px, 30px, 40px);
}
</style>

<img src="woman.png" class="translatexyz">
```

scale – 요소 확대/축소하기

- 지정한 크기만큼 확대/축소

- transform:scaleX(sx)
- transform:scaleY(sy)
- transform:scaleZ(sz)
- transform:scale(sx, sy)
- transform:scale3d(sx, sy, sz)

scaleX(sx)

x축 방향으로 sx만큼 확대. scale(sx, 1)과 같다.

- 1 : 원래 크기, 1보다 크면 : 확대, 1보다 작으면 : 축소

```
<style>
.scalex {
      transform:scaleX(2.0);
      -webkit-transform:scaleX(2.0);
      -ms-transform:scaleX(2.0);
}
</style>

<div id="tr">
          <img src="woman.png" class="scalex">
</div>
```

원본 이미지

scaleY(sy)

y축 방향으로 sy만큼 확대. scale(1, sy)과 같다.

- 1 : 원래 크기, 1보다 크면 : 확대, 1보다 작으면 : 축소

```
<style>
 .scaley{
    transform:scaleY(1.5);
    -webkit-transform:scaleY(1.5);
    -ms-transform:scaleY(1.5);
  }
</style>

<div id="tr">
        <img src="woman.png" class="scaley">
</div>
```

원본 이미지

scale(sx, sy)

- x축 방향으로 sx만큼, y축 방향으로 sy만큼 확대
- sy 값이 주어지지 않으면 sx 값과 같다고 간주

  예) scale(2)는 scale(2,2)와 같다.

```
<style>
 .scalexy{
  transform:scale(2);
  -webkit-transform:scale(2);
  -ms-transform:scale(2);
  }
</style>

<div id="tr">
      <img src="woman.png" class="scalexy">
</div>
```

원본 이미지

scaleZ(sz)

- z축 방향으로 sz만큼 확대. scale(1, 1, sz)과 같다.
- 브라우저 상에서 직접 확인 불가 → 다른 효과와 함께 사용

```
<style>
 .scalez{
    transform:scaleZ(0.7);
    -webkit-transform:scaleZ(0.7);
    -ms-transform:scaleZ(0.7);
  }
</style>

<div id="tr">
     <img src="woman.png" class="scalez">
</div>
```

원본 이미지

rotate – 요소 회전하기

- 지정한 각도만큼 웹 요소 회전

- transform:rotateX(각도), transform:rotateY(각도)
- transform:rotateZ(각도), transform:rotate(rx, ry, 각도)
- transform:rotate3d(rx, ry, rz, 각도)

- rotate 변형에서 사용하는 각도는 도(degree) 또는 라디안
  1라디안은 1/180°

예) transform:rotateY(45deg)

rotateX(각도)

x축 방향으로 지정한 각도만큼 회전.

```
<style>
 .rotatex{
    transform:rotateX(45deg);
    -webkit-transform:rotateX(45deg);
    -ms-transform:rotateX(45deg);
  }
</style>

<div id="tr">
        <img src="woman.png" class=“rotatex“>
</div>
```

rotateY(각도)

y축 방향으로 지정한 각도만큼 회전.

```
<style>
 .rotatey{
    transform:rotateY(45deg);
    -webkit-transform:rotateY(45deg);
    -ms-transform:rotateY(45deg);
  }
</style>

<div id="tr">
        <img src="woman.png" class=“rotatey“>
</div>
```

rotateZ(각도)

z축 방향으로 지정한 각도만큼 회전.

```
<style>
 .rotatez{
    transform:rotateZ(45deg);
    -webkit-transform:rotateZ(45deg);
    -ms-transform:rotateZ(45deg);
  }
</style>

<div id="tr">
        <img src="woman.png" class=“rotatez“>
</div>
```

rotate3d(rx, ry, rz, 각도)

방향 벡터(rx, ry, rz)와 방향이 같은 직선을 기준으로 각도만큼 회전 .

```
<style>
 .rotatexyz{
    transform:rotate3d(2.5, 1.2, -1.5, 45deg);
    -webkit- transform:rotate3d(2.5, 1.2, -1.5, 45deg);
    -ms- transform:rotate3d(2.5, 1.2, -1.5, 45deg);
  }
</style>

<div id="tr">
        <img src="woman.png" class=“rotatexyz“>
</div>
```

skew – 요소를 비틀어 왜곡하기

- skew 변형은 지정한 각도만큼 요소를 비틀어 왜곡
- 양쪽 방향으로 또는 한쪽 방향으로만 비틀 수 있다.

- transform:skewX(각도)
- transform:skewY(각도)
- transform:skew(각도, 각도)

skewX(각도)

- x축으로 지정한 각도만큼 비튼다

```
<style>
 .skewx{
    transform:skewX(30deg);
    -webkit-transform:skewX(30deg);
    -ms-transform:skewX(30deg);
  }
</style>

<div id="tr">
        <img src="woman.png" class="skewx">
</div>
```

skew(각도, 각도)

- x축과 y축으로 동시에 왜곡시킴

```
<style>
 .skew{
     transform:skew(-25deg, -15deg);
     -webkit-transform:skew(-25deg, -15deg);
     -ms-transform:skew(-25deg, -15deg);
}
</style>

<div id="tr">
          <img src="woman.png" class=“skew“>
</div>
```

transform-origin

지정한 요소의 변형 원점을 설정하는 것

같은 변형 함수라도 변형의 원점에 따라 결과가 달라진다

transform-orgin: x축 y축 z축 | initial | inherit;

- x축 : 원점에서의 x 좌표값. 길이 값이나 백분율, left, center, right 중에서 사용
- y축 : 원점에서의 y 좌표값. 길이 값이나 백분율, top, center, bottom 중에서 선택.
- z축 : 원점에서의 z 좌표값. 길이 값만 사용

transform-origin

```
<style>
.ltop{
        transform-origin:left top;
        -webkit-transform-origin:left top;
        -moz-transform-origin:left top;
}
</style>

<div>
   <img src="sunset.jpg" class="ltop">
</div>
```

```
<style>
.rtop{
        transform-origin:right top;
        -webkit-transform-origin:right top;
        -moz-transform-origin:right top;
}
</style>

<div>
   <img src="sunset.jpg" class="rtop">
</div>
```

transform-origin

```
<style>
.lbt{
        transform-origin:left bottom;
        -webkit-transform-origin:left bottom;
        -moz-transform-origin:left bottom;
}
</style>

<div>
   <img src="sunset.jpg" class="lbt">
</div>
```

```
<style>
.rbt{
        transform-origin:right bottom;
        -webkit-transform-origin:right bottom;
        -moz-transform-origin:right bottom;
}
</style>

<div>
   <img src="sunset.jpg" class="rbt">
</div>
```

perspective

3차원 변형에서 사용되는 속성

원래 위치에서 사용자가 있는 방향이나 반대 방향으로 잡아당기거나 밀어내 원근감을 갖게 한다.

perspective: <크기> | none;

- <크기> : 원래 위치에서 사용자가 있는 방향으로 얼마나 이동하는지를 픽셀로 지정
- none : perspective를 지정하지 않는다.(기본값)

perspective-origin

3차원 변형에서 사용되는 속성

좀 더 높은 곳에서 원근을 조절하는 듯한 느낌을 만들때 사용.

perspective-origin: <x축 값> | <y축 값>;

- <x축 값> : 웹 요소가 x축에서 어디에 위치하는지를 지정한다. 사용할 수 있는 값은 길이 값이나 백분율, left, right, center이다. (기본값은 50%)
- <y축 값> : 웹 요소가 y축에서 어디에 위치하는지를 지정한다. 사용할 수 있는 값은 길이 값이나 백분율, top, center, bottom이다. (기본값은 50%)

perspective

```
#pers {
    -moz-perspective: 300px;
    -ms-perspective: 300px;
    -webkit-perspective: 300px;
    perspective: 300px;
}
```

transform-style

여러가지 변형을 동시에 적용할 때 사용한다.

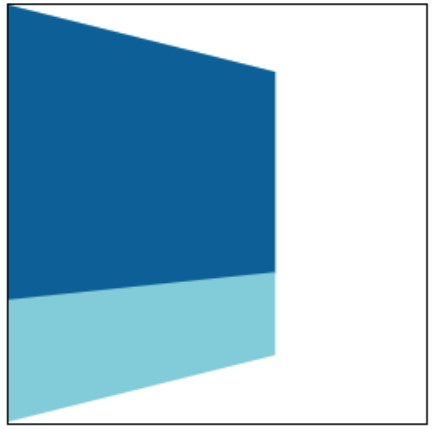
transform-style: flat | preserve-3d;

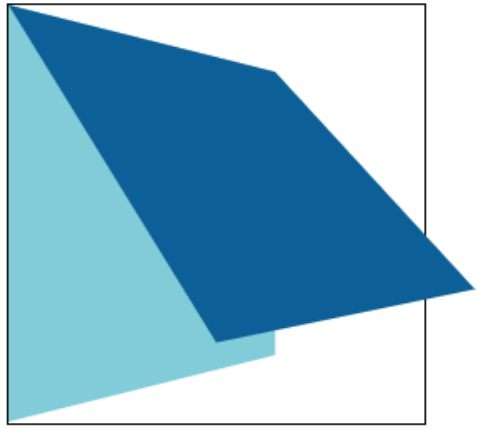
- flat : 하위 요소를 평면으로 처리한다.
- preserve-3d : 하위 요소들에 3d 효과를 적용한다.

transform-style

```
background: #0d6097;
-webkit-transform-origin: left top;
-moz-transform-origin: left top;
-ms-transform-origin: left top;
-o-transform-origin: left top;
transform-origin: left top;
-webkit-transform: rotateY(45deg);
-moz-transform: rotateX(45deg);
-ms-transform: rotateX(45deg);
-o-transform: rotateX(45deg);
transform: rotateX(45deg);
```

```
#tr-style1 {
    -webkit-transform-style: flat;
    transform-style: flat;
}
#tr-style2 {
    -webkit-transform-style: preserve-3d;
    transform-style: preserve-3d;
}
```





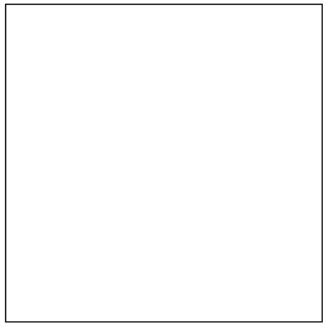
backface-visibility

뒷면을 보일지 안보일지 여부를 결정하는 속성.

backface-visibility : visible | hidden;

- visible : 뒷면을 표시한다. 기본 값.
- hidden : 뒷면을 표시하지 않는다.

```
#back1 {
    -webkit-backface-visibility: hidden;
    -moz-backface-visibility: hidden;
    backface-visibility: hidden;
}
#back2 {
    -webkit-backface-visibility: visible;
    -moz-backface-visibility: visible;
    backface-visibility: visible;
}
```



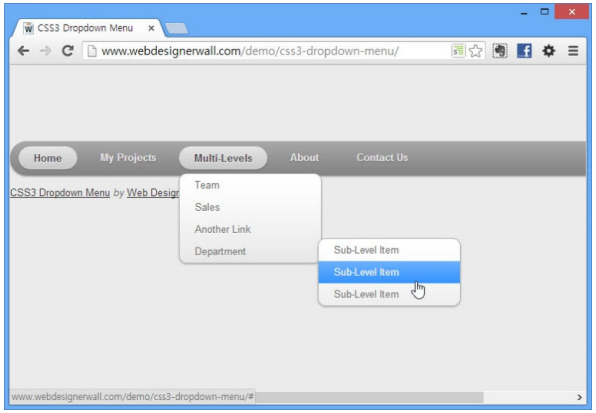
시간에 따른 변화를 보여주는 transition 속성

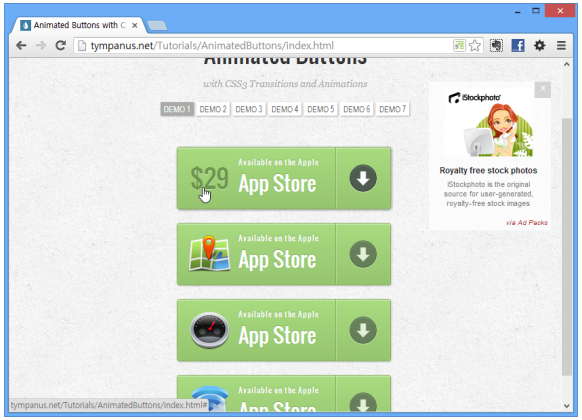
트랜지션이란

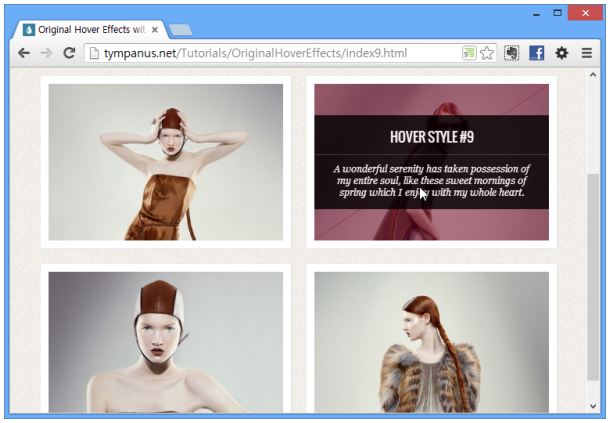
- 시간이 흐르면서 한 스타일에서 다른 스타일로 바뀌는 것
- 스타일이 바뀌는 시간을 조절해서 애니메이션 효과
- 플래시나 자바스크립트를 사용하지 않고 CSS 소스만으로 애니메이션 효과를 낼 수 있다.
- 트랜지션 속성은 브라우저 접두사를 붙여 사용해야 한다.

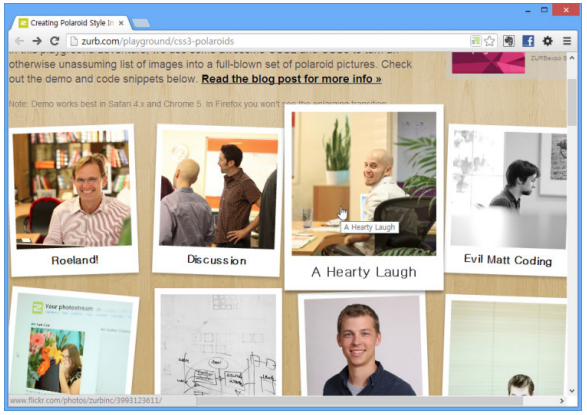
트랜지션이란

클릭하면 예제 사이트로 연결됩니다









transition-property

- 트랜지션 효과를 만드는 첫 번째 단계 :
- CSS 속성 중 어떤 속성에 트랜지션을 적용할 것인지 선택
- transition-property: 속성값
  - **none** : 트랜지션 동안 아무 속성도 바뀌지 않음.
  - **all** : 요소의 모든 속성이 트랜지션 대상.
  - **속성 이름** : 트랜지션 효과를 적용할 속성 이름 지정. 속성이 여럿일 경우 쉼표(,)로 구분하여 나열.

transition-property

예제_ Samples\10장\transition1.html

```
<style>
#ex div{
  width:100px;  /* 너비 */
  height:100px;  /* 높이 */
  background-color:red;  /* 배경색 빨강 */
  border-radius:0px;  /* 모서리 부분 둥글게 처리 안 함 */
  transition-property:background-color, border-radius ;
                                  /* 트랜지션 대상: background-color, border-radius */
  -webkit-transition-property:background-color, border-radius;
  -moz-transtion-property:background-color, border-radius;
}
#ex:hover div{
  background-color:blue;  /* 배경색 파랑 */
  border-radius:50px;  /* 모서리 부분 50px */
}
</style>
```

배경은 파란색으로, 모서리는 50px로 스타일 지정

빨간색 사각형의 배경과 모서리에 트랜지션 속성을 적용하는 스타일 지정

도형 위로 마우스 포인터를 올려보세요

도형 위로 마우스 포인터를 올려보세요

transition-duration

- 진행 시간을 지정해 줘야 그 시간 동안 속성이 자연스럽게 바뀌면서 애니메이션 효과를 만들 수 있다.
- 트랜지션 대상이 되는 속성이 여럿이라면 트랜지션 진행 시간도 쉼표(,)로 구분한다. 시간은 초(s) 또는 밀리초(ms) 단위.

transition-duration: 시간;

transition-duration

예제_ Samples\10장\transition2.html

```
<style>
#ex div{
  width:100px; /* 너비 */
  height:100px; /* 높이 */
  background-color:red; /* 배경색 빨강 */
  border-radius:0px; /* 모서리 부분 둥글게 처리 안 함 */
  transition-property:background-color, border-radius;
                                /* 트랜지션 대상: background-color, border-radius */
  -webkit-transition-property:background-color,
  border-radius;
  -moz-transtion-property:background-color, border-radius;
  transition-duration:1s; /* 트랜지션 진행 시간 1초 */
  -webkit-transition-duration:1s;
  -moz-transition-duration:1s;
}
#ex:hover div{
  background-color:blue; /* 배경색 파랑 */
  border-radius:50px; /* 모서리 부분 50px */
```

배경은 파란색, 모서리는 50px로 스타일 지정

transition-timing-function

트랜지션 효과의 속도를 지정할 수 있으며, 베지에 곡선을 이용해 표현
기본 값은 ease

- linear 시작에서 끝까지 똑같은 속도로 트랜지션 진행.
- ease 처음에는 천천히 시작하고 점점 빨라지다가 마지막엔 천천히 끝남.
- ease-in 트랜지션 시작을 느리게.
- ease-out 트랜지션을 느리게 끝냄.
- ease-in-out 느리게 시작하고 느리게 끝냄.
- cubic-bezier(n,n,n,n) 직접 베지에 함수를 정의해서 사용. n에 들어가는 숫자는 0~1사이 값

transition-delay

- 트랜지션 효과를 언제부터 시작할 것인지 지연시간 설정
- 이 속성에서 지정하는 시간만큼 기다렸다가 트랜지션이 시작됨.

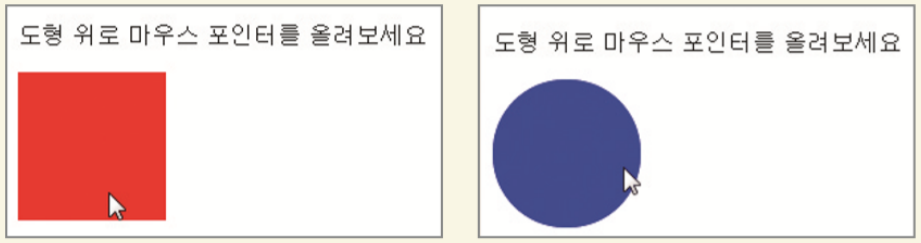
transition-delay: 시간

- 사용할 수 있는 값은 초(secons)나 밀리초(milliseconds)
- 기본 값은 0s

예제_ Samples\10장\transition3.html

```
<style>
#ex div{
    width:100px; /* 너비 */
    height:100px; /* 높이 */
    background-color:red; /* 배경색 빨강 */
    border-radius:0px; /* 모서리 부분 둥글게 처리 안함 */
    transition-property:background-color, border-radius;
                                /* 트랜지션 대상: background-color, border-radius */
    -webkit-transition-property:background-color,
    border-radius;
    -moz-transtion-property:background-color, border-radius;
    transition-duration:1s; /* 트랜지션 진행 시간 1초 */
    -webkit-transition-duration:1s;
    -moz-transition-duration:1s;
    transition-delay:2s; /* 트랜지션 지연 시간 2초 */
    -webkit-transition-delay:2s;
    -moz-transition-delay:2s;
}
</style>
```

● 구현된 결과



2초 후에 빨간색 사각형이 파란색 원으로 변함.

transition

트랜지션 속성을 모두 표기해야 할 경우 소스가 길어진다.
transition 속성을 한줄로 표시할 수 있다.
transition-duration 속성은 반드시 표기한다.

- property : 트랜지션이 적용될 css 속성의 이름
- duration : 트랜지션 실행 시간. 초나 밀리초.
- timing-function : 트랜지션 효과의 곡선 형태.
- delay : 트랜지션 지연 시간. 초나 밀리초.

예) 요소의 모든 부분에 1초 동안 linear 트랜지션을 적용한다면

transition: all 1s linear;

예제_ Samples\10장\transition4.html

```
<style>
#ex div{
    width:100px;  /* 너비 */
    height:100px;  /* 높이 */
    border-radius:0px;  /* 모서리 부분 둥글게 처리 안 함 */
    border:2px solid black;  /* 2px 검은색 테두리 */
    background:url(f1.png) no-repeat center center
    padding-box;  /* f1.png를 요소 중앙에 표시 */
    transition:all 1s ease-in 0.3s;
                                    /* 모든 요소를 대상으로 0.3초 동안 트랜지션 */
    -moz-transition:all 1s ease-in 0.3s;
    -webkit-transition:all 1s ease-in 0.3s;
}
</style>
```

한 줄에 표기한 트랜지션 속성

● **구현된 결과**

0.3초 후에 배경 이미지와 도형 테두리가 바뀜.

쉽게 애니메이션을 만드는 animation 속성

transition 속성과 animation 속성

공통점

- 시작 스타일과 끝나 는 스타일을 지정하면 전체적으로 부드럽게 변화하는 애니메이션 효과를 만든다.
- 애니메이션에 소요되는 시간이나 지연 시간 등을 지정한다.

다른점

- 애니메이션의 시작에서부터 끝날 때까지 어느 지점이든 @keyframes 속성을 사용해 애니메이션을 정의할 수 있다

@keyframes

- 사용자가 애니메이션을 정의한다.
- 애니메이션이 시작할 상태와 애니메이션이 끝날 때의 상태, 최소한 두 가지 상태를 설정해야 한다.
- 시작할 때의 상태 는 0%(또는 from) 스타일을 정의하고 끝나는 상태는 100%(또는 to)스타일을 정의한다.
- 중간에 원하는 부분에 애니메이션을 추가할 수 있다.

```
@keyframes myani{
    0% {
        background-color:red;
    }
    100%{
        background-color:blue;
    }
}
```

animation-name, animation-duration

animation-name

사용자가 @keyframe을 사용해 만든 애니메이션 이름
예) animation-name: myani;

animation-duration

애니메이션을 얼마 동안 재생할 것인지 설정
사용할 수 있는 값은 초(s)나 밀리초(ms). 기본 값은 0
속성 값을 정하지 않으면 애니메이션은 일어나지 않는다.

예) 애니메이션 진행 시간을 1초로 지정하려면
animation-duration:1s;

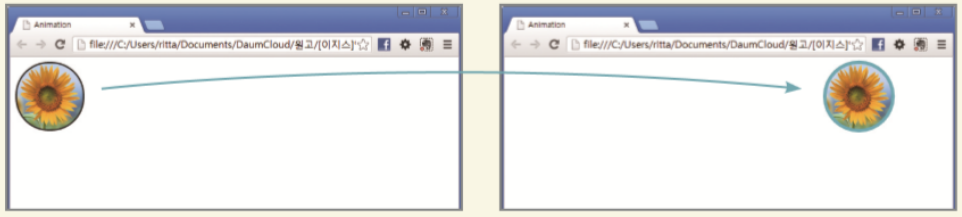
animation-name, animation-duration

```
<style>

#box{
        ……
        animation-name:myani;
        -webkit-animation-name:myani;
        animation-duration:3s;
        -webkit-animation-duration:3s;
}

@keyframes myani{
        0% {
                background-color:red;
                border-radius:0;
        }
        100%{
                background-color:blue;
                border-radius:50px;
         }
}
@-webkit-keyframes myani{
        ……..
}

</style>
```

예제_ Samples\10장\ani1.html

```
<style>
@keyframes myani {
     0%{ left:10px; } /* 시작할 때 x 좌표 위치 10px */
     100% { left:500px; } /* 끝날 때 x 좌표 위치 500px */
}
@-webkit-keyframes myani { /* webkit 브라우저 */
      0% { left:10px; } /* 시작할 때 x 좌표 위치 10px --> */
      100% { left:500px; } /* 끝날 때 x 좌표 위치 500px */
}
@-moz-keyframes myani { /* moz 브라우저 */
      0% { left:10px;  } /* 시작할 때 x 좌표 위치 10px */
      100% { left:500px;  } /* 끝날 때 x 좌표 위치 500px */
}
</style>
```

● 구현된 결과



이미지가 x 좌표 10px 위치에서 500px 위치까지 이동하는 애니메이션이 만들어짐.

animation-direction

- 애니메이션 방향 조절
- 기본은 애니메이션 실행 후 원래 위치로 되돌아가기
- 사용할 수 있는 값
  - ✓normal : 애니메이션 실행하고 원래 위치로 돌아가기. 기본값.
  - ✓reverse : 반대 쪽부터 애니메이션 실행
  - ✓alternate : 실행이 끝나면 반대 방향으로 실행하기
  - ✓alternate-reverse : 반대 쪽부터 시작해서 실행이 끝나면 반대 방향으로.

animation-iteration-count

- 반복 횟수 지정하기
- 애니메이션은 기본적으로 한 번만 실행하고 끝난다.
- 반복해야 할 경우에는 animation-iteration-count 속성을 사용해 반복 횟수를 지정한다.
- 사용할 수 있는 값은 반복 횟수, infinite(무한 반복)

예) 애니메이션을 무한 반복하려면

animation-iteration-count:infinite;

animation-iteration-count

```
-webkit-animation-name: moving;
-moz-animation-name: moving;
-o-animation-name: moving;
animation-name: moving;
-webkit-animation-duration: 3s;
-moz-animation-duration: 3s;
-o-animation-duration: 3s;
animation-duration: 3s;
-webkit-animation-direction: alternate;
-moz-animation-direction: alternate;
-o-animation-direction: alternate;
animation-direction: alternate;
-webkit-animation-iteration-count: infinite;
-moz-animation-iteration-count: infinite;
-o-animation-iteration-count: infinite;
animation-iteration-count: infinite;
```

CSS3 Animation

animation-time-function

- 트랜잭션과 마찬가지로 애니메이션에서도 애니메이션의 시작, 중간, 끝에서의 속도를 선택해 전체적인 속도 곡선을 지정할 수 있다.

animation-timing-function : linear | ease | ease-in | ease-out | ease-in-out | cubic-bezier(n,n,n,n)

- 각 값의 역할은 transform에서의 역할과 같다.

animation

- 애니메이션 관련 속성을 한꺼번에 표기
- animation- duration 속성은 반드시 표기해야 한다.

animation: name값 duration값 timing-function값 delay값
iteration-count값 direction값;

```
<style>
@keyframes myani{
	0% {
		left:10px;
	}
	30%{
		transform:scale(2);
		transform-rotate(360deg);
		-webkit-transform:scale(2);
		-webkit-transform:rotate(360deg);
	}
	100% {
		left:500px;
	}
}
@-webkit-keyframes myani{
	……
}
```

```
#myball {
	……
	animation:myani 3s alternate infinite;
	-webkit-animation:myani 3s alternate infinite;
    }

#yourball {
	……
	animation:myani 4s alternate-reverse infinite;
	-webkit-animation:myani 4s alternate-reverse
infinite;
    }

</style>
```