

## css 애니메이션

간단한 움직임 **transition** 속성 예제

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>CSS 애니메이션 예제</title>
  <style>
    /* 기본 div 스타일 */
    div {
      width: 200px;
      height: 100px;
      background: blue;
      color: white;
      text-align: center;
      line-height: 100px; /* 세로 중앙 정렬 */
      font-size: 20px;
      font-weight: bold;
      border-radius: 10px;
      transition: all 2s linear 1s; /* 모든 속성을 2초 동안 선형으로 변화,
1초 후 시작 */
    }

    /* 마우스를 올렸을 때 변경되는 속성 */
    div:hover {
      width: 400px;
      height: 50px;
      color: red;
      background: yellow;
      opacity: 0.5;
      line-height: 50px; /* 높이가 줄어들었으므로 조정 */
    }
  </style>
</head>
<body>

  <div>Hover me!</div>

</body>
```

</html>

일일이 기술 가능

## 설명

이 코드에서는 `div` 요소가 마우스를 올렸을 때(`hover`) 너비(`width`)만 애니메이션 효과가 적용됩니다.

속성	설명
<code>transition-property: width;</code>	너비( <code>width</code> ) 속성에만 애니메이션 적용
<code>transition-duration: 2s;</code>	애니메이션이 2초 동안 진행됨
<code>transition-timing-function: linear;</code>	일정한 속도로 애니메이션 실행됨
<code>transition-delay: 1s;</code>	<code>hover</code> 후 1초 동안 대기한 후 애니메이션 시작

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>CSS 애니메이션 예제</title>
  <style>
    div {
      width: 200px;
      height: 100px;
      background: blue;
      color: white;
      text-align: center;
      line-height: 100px;
      font-size: 20px;
      font-weight: bold;
      transition-property: width; /* width만 변경 */
      /*transition-property: width, height; 여러 속성기술시 사용*/
      transition-duration: 2s;
      transition-timing-function: linear;
      transition-delay: 1s;
    }
  </style>
</head>
<body>
  <div>안녕하세요</div>
</body>
</html>
```

```

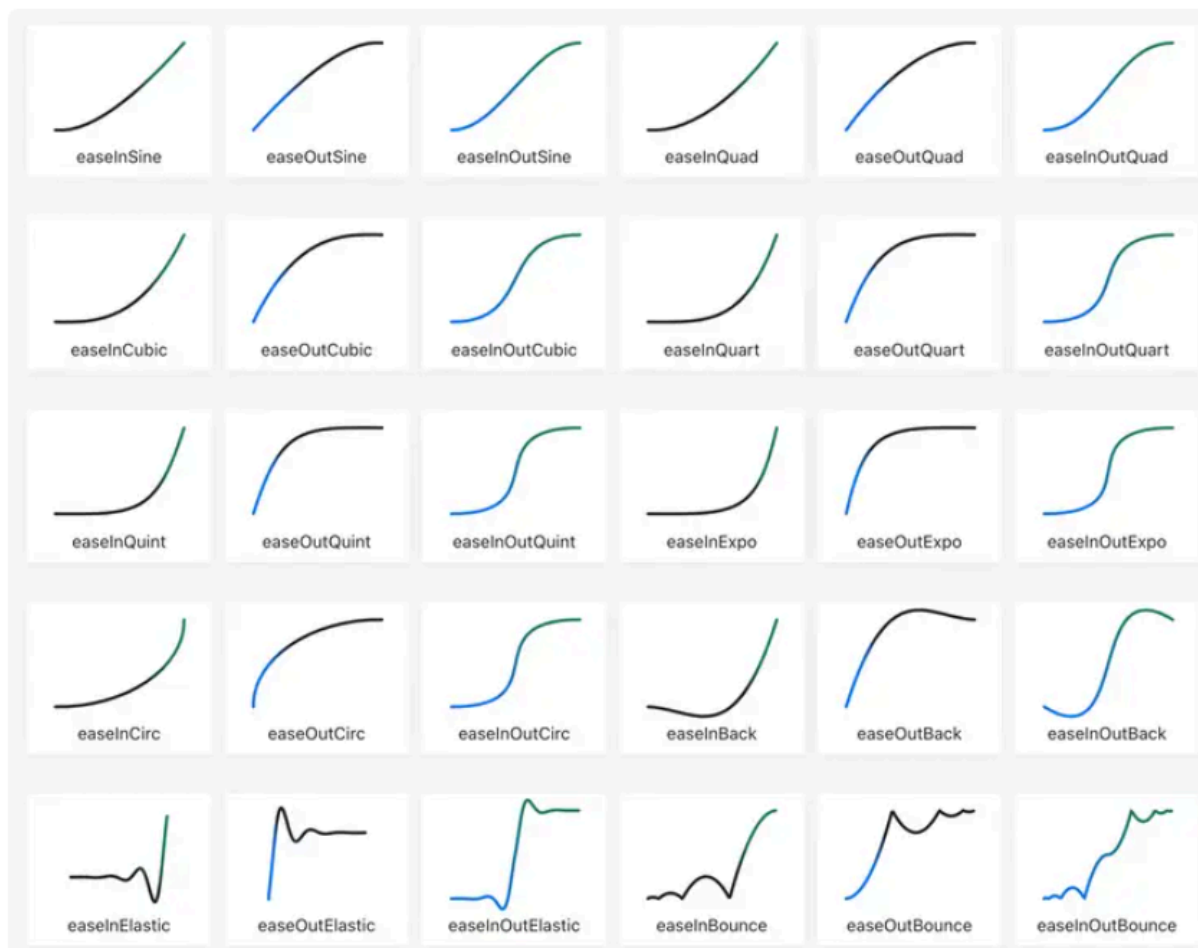
    /* 마우스를 올렸을 때 width 변화 */
    div: hover {
        width: 400px;
    }
</style>
</head>
<body>

    <div>Hover me!</div>

</body>
</html>

```

transition-timing-function



```

<!DOCTYPE html>
<html lang="ko">
<head>

```

```

<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>CSS 애니메이션 예제</title>
<style>
    /* 모든 div에 공통 스타일 적용 */
    .box {
        width: 100px;
        height: 50px;
        background: red;
        color: yellow;
        border: 1px solid black;
        text-align: center;
        line-height: 50px;
        font-weight: bold;
        position: relative; /* 위치 조정을 위해 relative 설정 */
        transition: width 3s;
    }

    /* transition-timing-function을 개별적으로 적용 */
    #div1 { transition-timing-function: linear; }
    #div2 { transition-timing-function: ease; }
    #div3 { transition-timing-function: ease-in; }
    #div4 { transition-timing-function: ease-out; }
    #div5 { transition-timing-function: ease-in-out; }
    #div6 { transition-timing-function: cubic-bezier(0.1, 0.0, 0.1, 1.0);
}

    /* 마우스를 올리면 모든 박스가 동시에 변화 */
    body:hover .box {
        width: 400px;
    }
</style>
</head>
<body>
    <div id="div1" class="box" style="top: 20px;">linear</div>
    <div id="div2" class="box" style="top: 40px;">ease</div>
    <div id="div3" class="box" style="top: 60px;">ease-in</div>
    <div id="div4" class="box" style="top: 80px;">ease-out</div>
    <div id="div5" class="box" style="top: 100px;">ease-in-out</div>
    <div id="div6" class="box" style="top: 120px;">cubic-bezier</div>
</body>
</html>

```

transform 속성 개요

CSS의 transform 속성은 요소를 변형(이동, 회전, 확대/축소, 기울이기 등)하는 데 사용됩니다.

아래는 transform의 주요 함수들과 예제입니다.

📌 주요 transform 함수 설명 및 예제

함수	설명	예제
translate(x, y)	요소를 x축, y축으로 평행 이동	transform: translate(50px, 100px);
rotate(angle)	요소를 시계 방향으로 회전	transform: rotate(20deg);
scale(x, y)	요소 크기 확대/축소	transform: scale(2, 3);
skewX(angle)	요소를 x축 기준으로 기울이기	transform: skewX(20deg);
skewY(angle)	요소를 y축 기준으로 기울이기	transform: skewY(20deg);
skew(x-angle, y-angle)	x, y축 둘 다 기울이기	transform: skew(20deg, 10deg);
matrix(a, b, c, d, e, f)	2D 변형을 행렬로 표현	transform: matrix(1, -0.3, 0, 1, 0, 0);

📌 예제 코드

```
html
복사편집
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
```

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>CSS Transform 예제</title>
<style>
    .container {
        display: flex;
        flex-direction: column;
        gap: 20px;
    }



    .box {
        width: 100px;
        height: 50px;
        background: tomato;
        color: white;
        text-align: center;
        line-height: 50px;
        font-weight: bold;
        border: 1px solid black;
        transition: transform 0.5s ease-in-out;
    }

    .box:hover { transform: translate(50px, 100px); }
    #rotate:hover { transform: rotate(20deg); }
    #scale:hover { transform: scale(2, 3); }
    #skewX:hover { transform: skewX(20deg); }
    #skewY:hover { transform: skewY(20deg); }
    #skew:hover { transform: skew(20deg, 10deg); }
    #matrix:hover { transform: matrix(1, -0.3, 0, 1, 0, 0); }
</style>
</head>
<body>
    <div class="container">
        <div class="box" id="translate">translate</div>
        <div class="box" id="rotate">rotate</div>
        <div class="box" id="scale">scale</div>
        <div class="box" id="skewX">skewX</div>
        <div class="box" id="skewY">skewY</div>
        <div class="box" id="skew">skew</div>
        <div class="box" id="matrix">matrix</div>
    </div>
</body>
</html>
```

---

### 실행 방법

1. 각 요소 위에 마우스를 올리면 **transform** 효과가 적용됩니다.
2. **translate**, **rotate**, **scale**, **skew** 등이 각각 다르게 동작하는 것을 확인할 수 있습니다.

 이제 **transform**을 직접 테스트해보세요! 

CSS 애니메이션은 요소의 스타일을 일정 시간 동안 변경하면서 부드러운 전환 효과를 줄 수 있는 기능입니다. JavaScript 없이도 애니메이션을 구현할 수 있어 성능이 뛰어나고 간단하게 적용할 수 있습니다.

---

## 1. CSS 애니메이션의 주요 속성

CSS 애니메이션은 `@keyframes` 규칙과 함께 `animation` 속성을 사용하여 구현됩니다.

### 1) @keyframes

애니메이션의 단계별 스타일을 정의합니다.

```
@keyframes move {
  0% {
    transform: translateX(0);
  }
  100% {
    transform: translateX(100px);
  }
}
```

위 코드는 `0%`에서는 원래 위치에 있고, `100%`에서는 `x`축으로 `100px` 이동하도록 설정합니다.

---

### 2) animation 속성

애니메이션을 적용할 요소에 설정하는 속성입니다.

```
animation: move 2s ease-in-out infinite alternate;
```

위 코드는 `move`라는 애니메이션을 2초 동안 실행하며, `ease-in-out` 효과를 적용하고, 무한 반복(`infinite`)되며, 한 방향으로 실행 후 반대로 실행(`alternate`)됩니다.



```
<!DOCTYPE html>

<html lang="ko">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>CSS Animation Example</title>

    <style>

        /* 애니메이션 정의 */

        @keyframes move {

            0% {

                transform: translateX(0);

            }

            100% {

                transform: translateX(100px);

            }

        }

        /* 애니메이션 적용 */

        .box {

            width: 100px;

            height: 100px;

            background-color: tomato;

            position: relative;
```

```
        animation: move 2s linear infinite alternate;
    }
</style>

</head>

<body>

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

</body>

</html>
```

---

## 2. animation 속성 상세 설명

속성	설명	예제
animation-name	사용할 애니메이션 이름	animation-name: move;
animation-duration	애니메이션 지속 시간	animation-duration: 2s;
animation-timing-function	속도 곡선 설정	animation-timing-function: ease-in-out;
animation-delay	애니메이션 시작 전 지연 시간	animation-delay: 1s;
animation-iteration-count	애니메이션 반복 횟수	animation-iteration-count: infinite;
animation-direction	애니메이션 진행 방향 설정	animation-direction: alternate;
animation-fill-mode	애니메이션 전후 상태	animation-fill-mode: forwards;

---

## animation-fill-mode 란?

애니메이션이 **시작 전이나 종료 후**에 스타일을 어떻게 유지할지를 결정하는 속성입니다.



## animation-fill-mode 속성 값 종류

속성 값	설명
none	기본값. 애니메이션이 끝난 후 원래 상태로 돌아감.
forwards	애니메이션이 끝난 후, <b>마지막 키프레임 상태 유지</b> .
backwards	애니메이션이 시작되기 전에, <b>첫 번째 키프레임 상태 유지</b> .
both	<b>forwards</b> + <b>backwards</b> 효과를 모두 적용.



## 3. CSS 애니메이션 예제

예제 1: 버튼에 호버하면 크기 커지기

```
@keyframes grow {  
  from {  
    transform: scale(1);  
  }  
  to {  
    transform: scale(1.2);  
  }  
}
```

```
button:hover {
```

```
    animation: grow 0.3s ease-in-out;
}
```

➡ 마우스를 올리면 버튼이 커집니다.

```
<!DOCTYPE html>

<html lang="ko">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Button Animation</title>

    <style>

        /* 애니메이션 정의 */

        @keyframes grow {

            from {

                transform: scale(1);

            }

            to {

                transform: scale(1.2);

            }

        }

        /* 버튼 스타일 */

        button {

            padding: 15px 30px;
```

```
    font-size: 16px;

    background-color: tomato;

    color: white;

    border: none;

    border-radius: 5px;

    cursor: pointer;

    transition: background-color 0.3s;
}

/* 버튼에 마우스를 올렸을 때 애니메이션 적용 */
button:hover {

    animation: grow 0.3s ease-in-out forwards;

    background-color: darkred; /* 즉시 변경 */
}

</style>

</head>

<body>

    <button>Hover Me!</button>

</body>

</html>
```

```

@keyframes spin {
  0% {
    transform: rotate(0deg);
  }
  100% {
    transform: rotate(360deg);
  }
}

.loader {
  width: 50px;
  height: 50px;
  border: 5px solid #ccc;
  border-top: 5px solid blue;
  border-radius: 50%;
  animation: spin 1s linear infinite;
}

```

➡ `.loader` 요소가 무한히 회전하는 로딩 아이콘이 됩니다.

```

<!DOCTYPE html>

<html lang="ko">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Loading Spinner</title>

  <style>

    /* 회전 애니메이션 정의 */

    @keyframes spin {

```

```
0% {  
  
    transform: rotate(0deg);  
  
}  
  
50% {  
  
    transform: rotate(180deg);  
  
}  
  
100% {  
  
    transform: rotate(360deg);  
  
}  
}  
  
/* 로딩 스피너 스타일 */  
  
.loader {  
  
    width: 50px;  
  
    height: 50px;  
  
    border: 5px solid #ccc; /* 회색 테두리 */  
  
    border-top: 5px solid blue; /* 상단만 파란색 */  
  
    border-radius: 50%;  
  
    animation: spin 1s linear infinite; /* 무한 회전 */  
  
    position: absolute;  
  
    top: 50%;  
  
    left: 50%;  
  
    transform: translate(-50%, -50%);  
  
}
```

```
        </style>

</head>

<body>

    <div class="loader"></div>

</body>

</html>
```

---

#### 4. transition vs animation 차이

속성	설명
<b>transition</b>	시작과 끝 상태만 지정할 때 사용 (호버 효과 등)
<b>animation</b>	여러 단계의 애니메이션이 필요할 때 사용 (@keyframes 활용)

예제: **transition** 사용

```
button {
    transition: transform 0.3s ease-in-out;
}

button:hover {
    transform: scale(1.2);
}
```



➡ **hover** 시 크기 변경이 되지만, 자동으로 실행되지는 않음.

예제: **animation** 사용

```
@keyframes bounce {  
  0%, 100% { transform: translateY(0); }  
  50% { transform: translateY(-20px); }  
}  
  
button {  
  animation: bounce 1s infinite;  
}
```

➡ 버튼이 계속 위아래로 움직이는 애니메이션.

---

## 5. 결론

CSS 애니메이션은 간단한 움직임을 부드럽게 표현할 수 있어 웹페이지에 생동감을 더할 수 있습니다.

복잡한 애니메이션이 필요하면 **JavaScript + CSS** 조합으로 활용하는 것도 좋은 방법입니다. 🚀