CSS Fundamentals-3

TRANSFORM

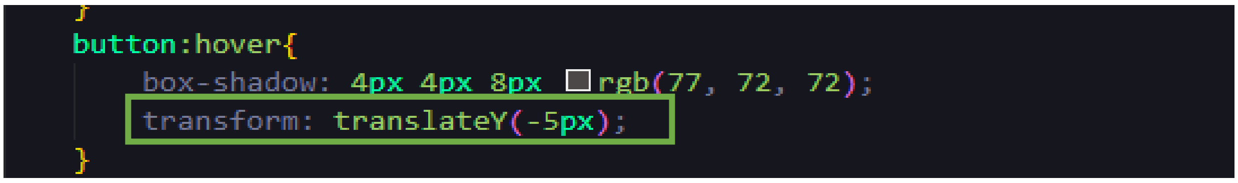
CSS transform Property – In-Depth Guide

The transform property in CSS allows you to **visually manipulate an element** by **translating (moving), rotating, scaling, and skewing** it without affecting the document layout.

◆ **Basic Syntax:**

```
selector {
   transform: transformation-function(value);
}
```

You can apply multiple transformations at once:



```
}
button:hover{
    box-shadow: 4px 4px 8px rgb(77, 72, 72);
    transform: translateY(-5px);
}
```

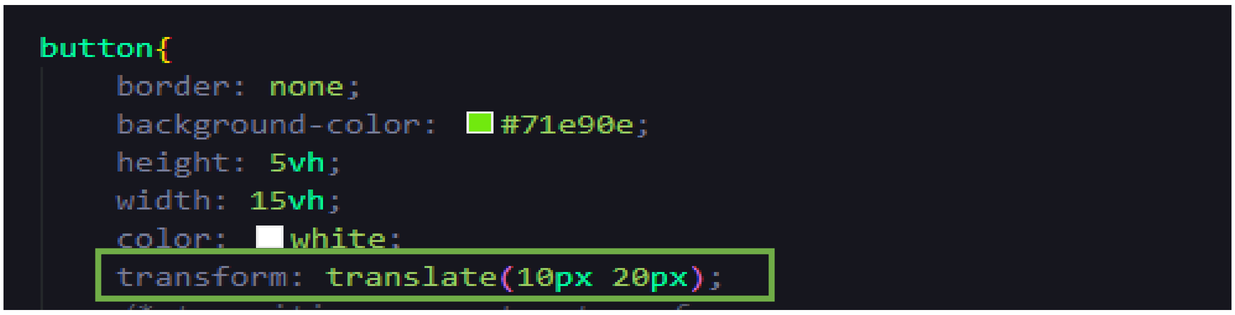
☞ **Transformations do not affect the normal flow of the document.** They only visually modify the element.

🏆 **Types of Transformations**

✅ **1. Translate (Move an Element):** Moves an element from its original position.

📌 **translate(x, y)**

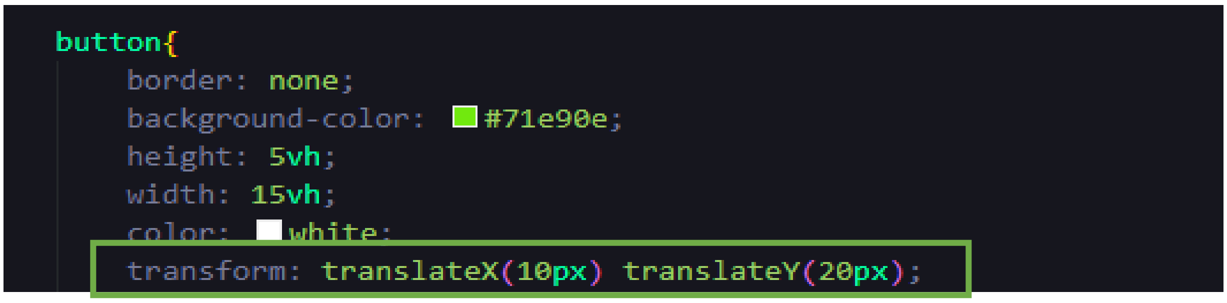
- Moves the element **horizontally (x)** and **vertically (y)**.
- Positive values move **right** (x) and **down** (y), while negative values move **left** (x) and **up** (y).



```
button{
    border: none;
    background-color: #71e90e;
    height: 5vh;
    width: 15vh;
    color: white;
    transform: translate(10px 20px);
```

- translateX(value): Moves only in the X-axis.
- translateY(value): Moves only in the Y-axis.

- translateZ(value): Moves along the Z-axis (**3D effect, requires perspective**).

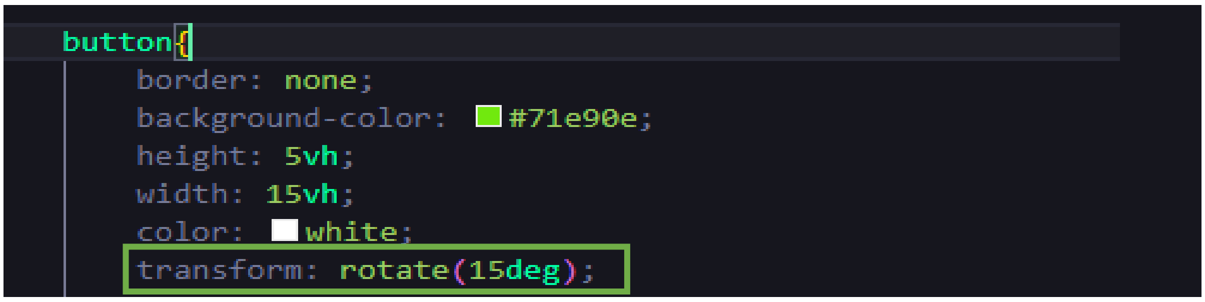


```
button{
    border: none;
    background-color: 🟩#71e90e;
    height: 5vh;
    width: 15vh;
    color: ⬜white;
    transform: translateX(10px) translateY(20px);
```

✅ **2. Rotate (Rotate an Element):** Rotates an element around its **origin (default is center).**

📌 **rotate(angle)**

- **Unit:** deg (degrees), turn (full circles), rad (radians).
- Positive values = **clockwise rotation**.
- Negative values = **counterclockwise rotation**.



```
button{
    border: none;
    background-color: 🟩#71e90e;
    height: 5vh;
    width: 15vh;
    color: ⬜white;
    transform: rotate(15deg);
```

rotateX(angle): Rotates around the **X-axis**.

- rotateY(angle): Rotates around the **Y-axis**.
- rotateZ(angle): Rotates around the **Z-axis**.

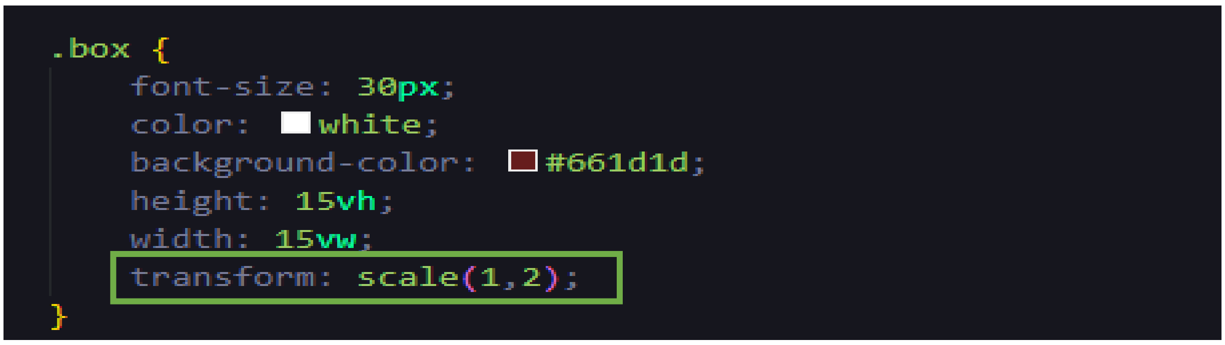
```
.box {
  transform: rotateX(60deg);
  transform: rotateY(120deg);
}
```

✅ **3. Scale (Resize an Element):** Changes the size of an element.

📌 **scale(x, y)**

- **x**: Scaling factor along the X-axis.
- **y**: Scaling factor along the Y-axis.

- Default is (1,1), where 1 means no scaling.

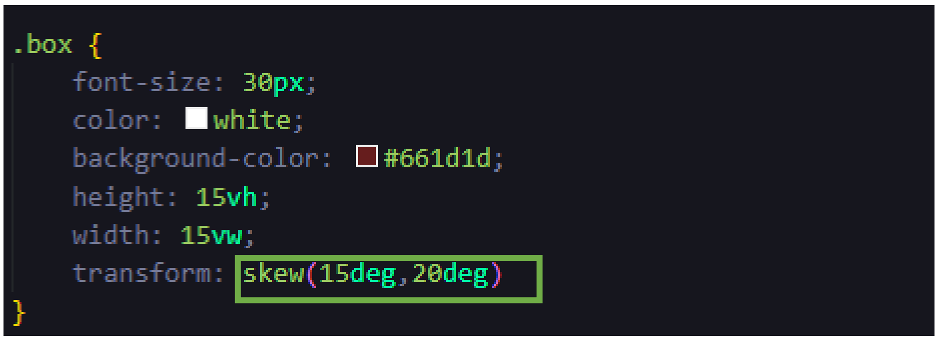


```css
.box {
    font-size: 30px;
    color: white;
    background-color: #661d1d;
    height: 15vh;
    width: 15vw;
    transform: scale(1,2);
}
```

- scaleX(value): Scales only horizontally.
- scaleY(value): Scales only vertically.

```
.box {
  transform: scaleX(2); /* Doubles width */
  transform: scaleY(0.5); /* Shrinks height to 50% */
}
```

☞ **If only one value is given, it applies to both axes.**



```css
.box {
    font-size: 30px;
    color: white;
    background-color: #661d1d;
    height: 15vh;
    width: 15vw;
    transform: skew(15deg,20deg)
}
```

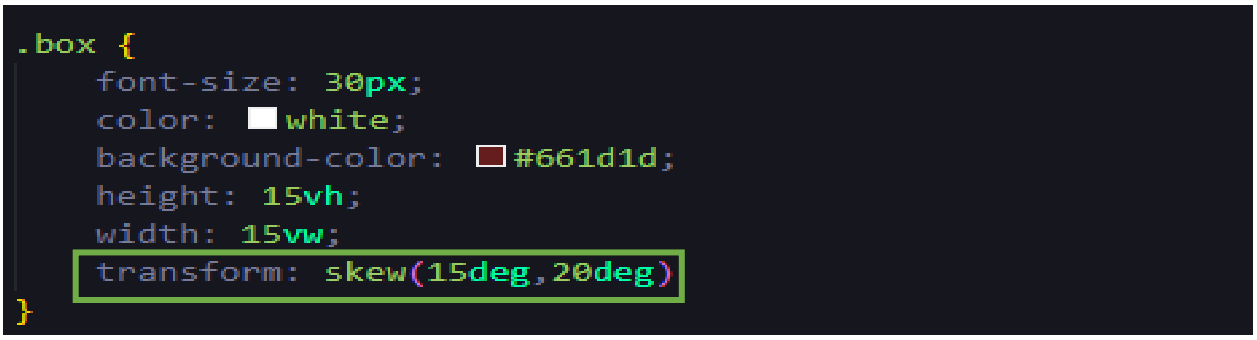
✓ **4. Skew (Slant an Element):** Tilts an element along X or Y axis.

📌 **skew(x-angle, y-angle)**

- **Skews horizontally (x-angle)** and **vertically (y-angle)**.
- Uses **degrees (deg)**.

```
.box {
  transform: skew(30deg, 15deg);
}
```

- skewX(angle): Skews only along the X-axis.
- skewY(angle): Skews only along the Y-axis.


```
.box {
    font-size: 30px;
    color: white;
    background-color: #661d1d;
    height: 15vh;
    width: 15vw;
    transform: skew(15deg,20deg)
}
```

✅ **5. Perspective (Depth Effect in 3D):** Used with 3D transforms to create depth.

📌 **perspective(value)**

- Lower values = **stronger depth effect**.
- **Must be applied to a parent element.**

```
.container {
   perspective: 500px;
}

.box {
   transform: rotateY(45deg);
}
```

🎭 **Combining Multiple Transforms**

You can combine transformations by writing them in a single transform property.

```
.box {
   transform: translate(50px, 20px) rotate(30deg) scale(1.2);
}
```

☞ **Order matters!** translate → rotate → scale may look different than rotate → translate → scale.

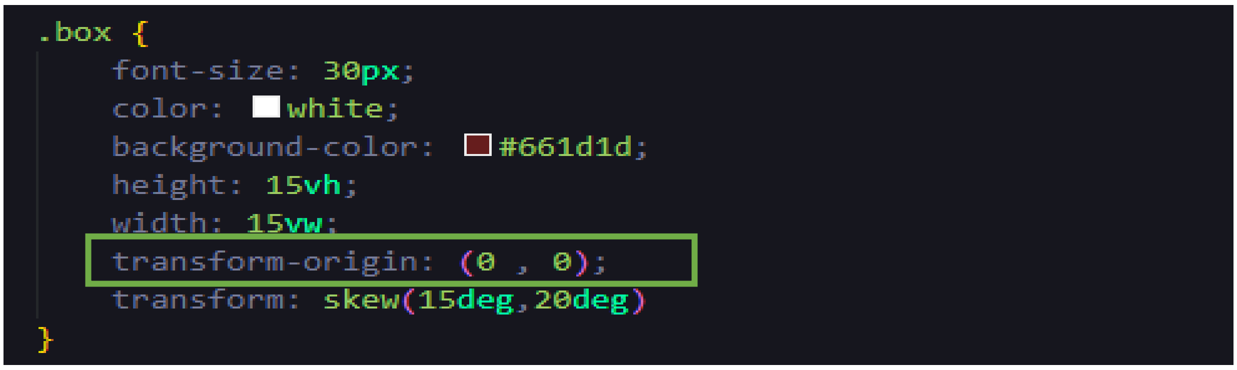
🎯 **Transform-Origin (Changing the Rotation Point)**

By default, transformations happen from the **center** of an element. The transform-origin property changes this.

📌 **Syntax:**

transform-origin: x y;

- **x**: left, right, center, or a percentage (50%).
- **y**: top, bottom, center, or a percentage (50%).



```
.box {
    font-size: 30px;
    color: white;
    background-color: #661d1d;
    height: 15vh;
    width: 15vw;
    transform-origin: (0 , 0);
    transform: skew(15deg,20deg)
}
```

☞ Default is 50% 50% (center).

🎬 **3D Transformations:** 3D transformations require perspective to create depth.

✅ **1. Rotate in 3D**

```
.box {
  transform: rotate3d(1, 1, 0, 45deg);
}
```

(1,1,0) means **rotation along X and Y axes**.

✅ **2. Translate in 3D**

```
.box {
  transform: translate3d(50px, 50px, 100px);
}
```

The **third value** moves the element forward or backward.

TRANSITION

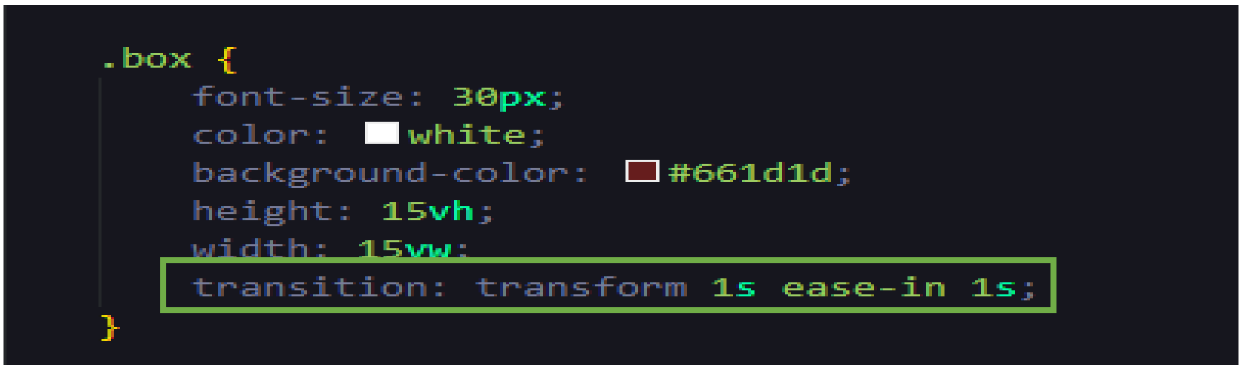
The transition property in CSS allows smooth **animations** between different property values over a specified duration.

◆ **Basic Syntax**

```
selector {
   transition: property duration timing-function delay;
}
```

- **property** → The CSS property to animate (e.g., background-color, transform, opacity).
- **duration** → How long the transition lasts (e.g., 1s, 500ms).
- **timing-function** → Defines the speed curve of the transition.
- **delay** → The delay before the transition starts (e.g., 0.5s).

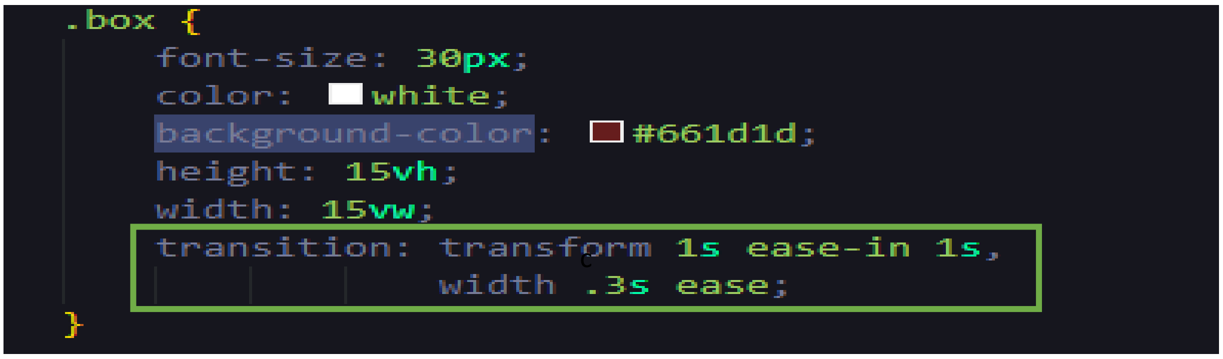
🏆 **1. Transition on a Single Property**



```
.box {
    font-size: 30px;
    color: white;
    background-color: #661d1d;
    height: 15vh;
    width: 15vw;
    transition: transform 1s ease-in 1s;
}
```

☞ **When hovered, the button smoothly changes color over 1 seconds.**

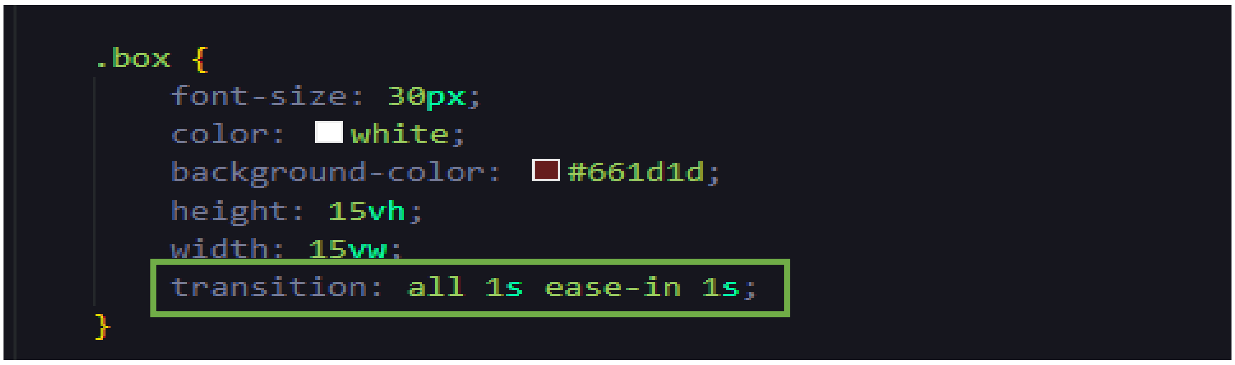
🏆 **2. Transition Multiple Properties:** You can apply transitions to multiple properties at once.



```
.box {
    font-size: 30px;
    color: white;
    background-color: #661d1d;
    height: 15vh;
    width: 15vw;
    transition: transform 1s ease-in 1s,
                width .3s ease;
}
```

☞ **Each property has its own duration and timing function.**

🏆 3. Transition All Properties



```css
.box {
    font-size: 30px;
    color: ⬜white;
    background-color: 🟥#661d1d;
    height: 15vh;
    width: 15vw;
    transition: all 1s ease-in 1s;
}
```

- This applies the transition to **all animatable properties**.
- ⚠ **Be careful!** It can impact performance if used on unnecessary properties.

🎭 Timing Functions (Speed Curves)

Defines how the transition progresses over time.

Value	Description
linear	Constant speed
ease	Slow → Fast → Slow (default)
ease-in	Slow start, then fast
ease-out	Fast start, then slow
ease-in-out	Slow start & end, fast in the middle
cubic-bezier(n,n,n,n)	Custom speed curve

Example:

```
.box {
   transition: transform 1s cubic-bezier(0.17, 0.67, 0.83, 0.67);
}
```

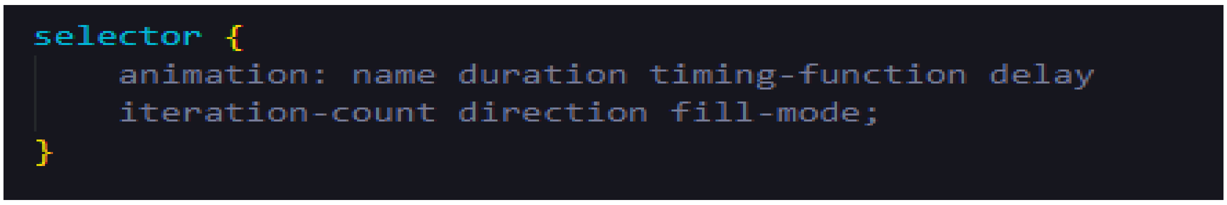
☞ **Custom cubic-bezier() lets you fine-tune animations.**

ANIMATION

CSS animations allow elements to change styles smoothly over time without using JavaScript.

◆ 1. Basic Syntax

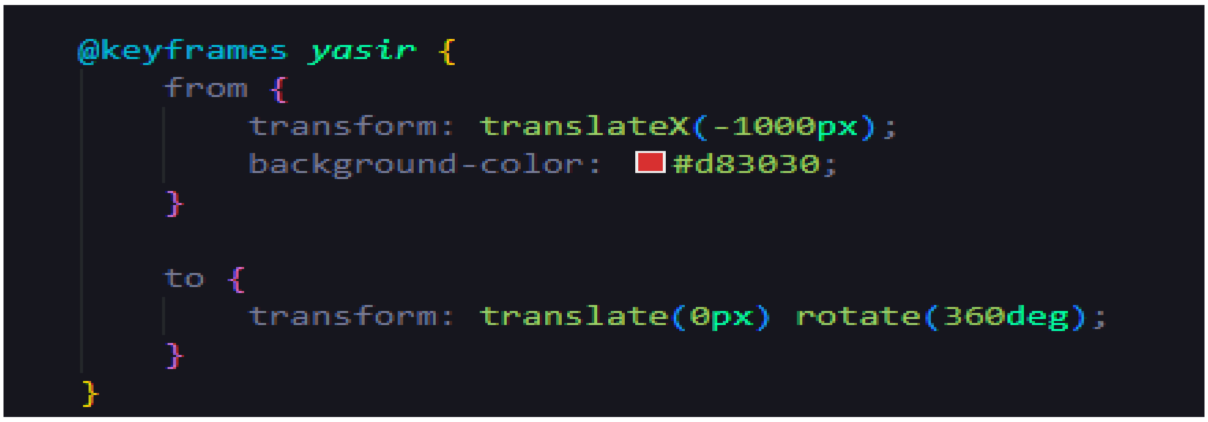
```
@keyframes animation-name {
   from { property: value; }
   to { property: value; }
}
```



```
selector {
    animation: name duration timing-function delay
    iteration-count direction fill-mode;
}
```

- **animation-name** → The name of the animation (defined using @keyframes).
- **duration** → How long the animation lasts (2s, 500ms).
- **timing-function** → Speed curve (ease, linear, ease-in-out).
- **delay** *(optional)* → Time before the animation starts (0.5s).
- **iteration-count** → Number of times animation runs (infinite, 3).
- **direction** *(optional)* → Controls the animation flow (normal, reverse, alternate).
- **fill-mode** *(optional)* → Defines the state before/after animation (forwards, backwards).

🏆 **2. Creating a Simple Animation:** Let's animate a box **moving from left to right**.

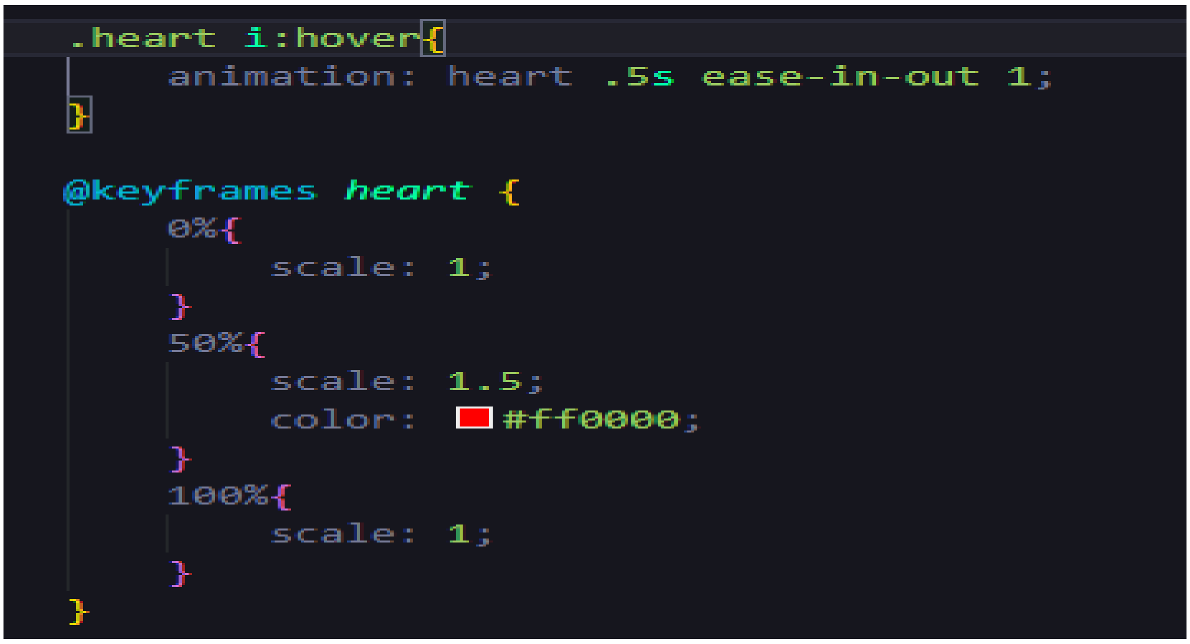


```
@keyframes yasir {
    from {
        transform: translateX(-1000px);
        background-color: 🟥#d83030;
    }

    to {
        transform: translate(0px) rotate(360deg);
    }
}
```

☞ **The box moves smoothly from its original position to 200px right over 2 seconds.**

🏆 **3. Using 0% and 100% for More Control:** Instead of from and to, you can define steps using percentages.



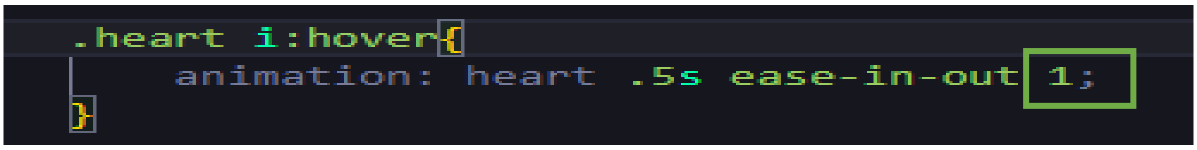
```css
.heart i:hover{
        animation: heart .5s ease-in-out 1;
}

@keyframes heart {
        0%{
                scale: 1;
        }
        50%{
                scale: 1.5;
                color: #ff0000;
        }
        100%{
                scale: 1;
        }
}
```

☞ **The element scales up and every time it hovered.**

🏆 **4. Controlling Animation Repetitions (iteration-count)**

- infinite → Runs forever.
- 1 (default) → Runs once.
- 3 → Runs three times.



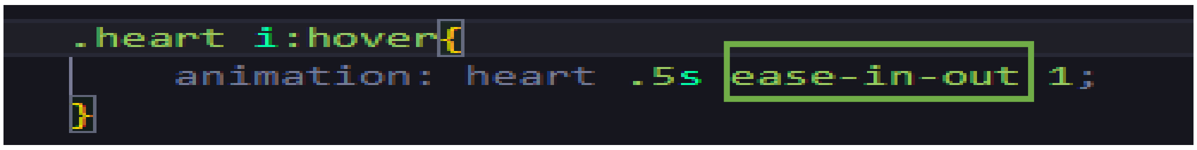
```css
.heart i:hover{
        animation: heart .5s ease-in-out 1;
}
```

☞ **The animation repeats 3 times and then stops.**

🎭 5. Animation Timing Functions

Defines the speed curve of the animation.

Value	Description
linear	Constant speed
ease	Slow → Fast → Slow (default)
ease-in	Slow start, then fast
ease-out	Fast start, then slow
ease-in-out	Slow start & end, fast in the middle
cubic-bezier(n,n,n,n)	Custom speed curve

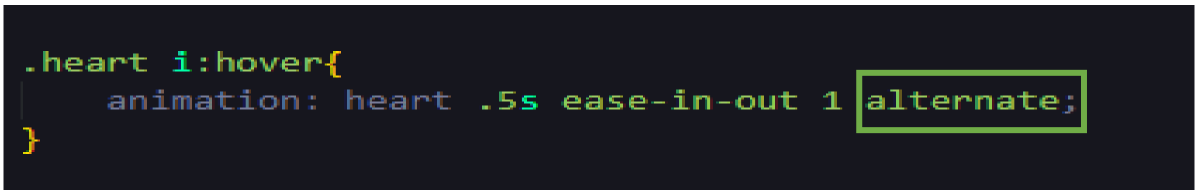


```
.heart i:hover{
    animation: heart .5s ease-in-out 1;
}
```

☞ **Cubic Bezier allows fine-tuned animations.**

🎬 6. Animation Direction (direction)

- normal → Runs forward (default).
- reverse → Runs backward.
- alternate → Runs forward, then backward.
- alternate-reverse → Runs backward, then forward.



```
.heart i:hover{
    animation: heart .5s ease-in-out 1 alternate;
}
```

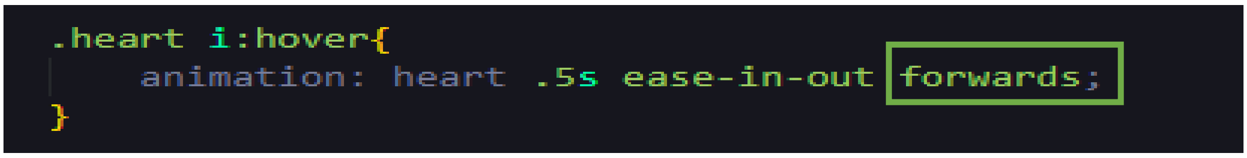
☞ **The box moves right, then left, then right, etc.**

🏆 7. Animation Fill Modes (fill-mode)

Controls the element's style before and after the animation.

- none (default) → No effect after animation ends.

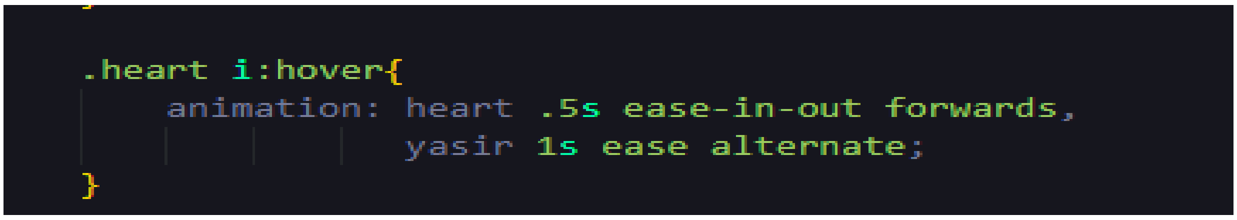
- forwards → Keeps final state (100% or to).
- backwards → Applies starting state (0% or from) before animation starts.
- both → Combines forwards and backwards.



```css
.heart i:hover{
    animation: heart .5s ease-in-out forwards;
}
```

☞ **The element stays in the final forward state.**

🎯 8. Combining Multiple Animations



```css
.heart i:hover{
    animation: heart .5s ease-in-out forwards,
               yasir 1s ease alternate;
}
```

☞ **Both heart and yasir run together.**