# CSS Transitions and Animations

CSS provides powerful tools for creating smooth, engaging user experiences through **transitions** and **animations**. These effects can enhance the visual appeal and usability of your website without needing JavaScript.

---

## 1. CSS Transitions

A **transition** is used to change CSS properties **smoothly** over a given duration.

### Basic Syntax

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

- `property` : The CSS property to animate (e.g., `background-color` , `transform` , etc.)
- `duration` : How long the transition lasts (e.g., `0.3s` , `1s` )
- `timing-function` : The pace of the transition ( `ease` , `linear` , `ease-in` , `ease-out` , etc.)
- `delay` : Optional delay before starting

### Example

```
.button {
  background-color: blue;
  color: white;
  transition: background-color 0.3s ease;
}
```

```
.button:hover {
  background-color: green;
}
```

This smoothly changes the button’s background color on hover.

---

### Shorthand vs Longhand

Shorthand:

```
transition: all 0.5s ease;
```

Longhand:

```
transition-property: background-color;
transition-duration: 0.5s;
transition-timing-function: ease;
transition-delay: 0s;
```

---

## 2. CSS Animations

CSS animations allow more **complex, keyframe-based** changes over time.

### Basic Syntax

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

### Keyframes

Define how the animation should behave at different points:

```
@keyframes slideIn {
  from {
    transform: translateX(-100%);
    opacity: 0;
  }
  to {
    transform: translateX(0);
    opacity: 1;
  }
}
```

### Example

```
.box {
  width: 100px;
  height: 100px;
  background-color: red;
  animation: slideIn 1s ease-in-out;
}
```

---

## Animation Properties

| Property | Description |
|---|---|
| `animation-name` | Name of the `@keyframes` to use |
| `animation-duration` | How long the animation takes |
| `animation-delay` | Delay before starting |
| `animation-iteration-count` | Number of times to run (or `infinite`) |
| `animation-direction` | `normal`, `reverse`, `alternate` |
| `animation-fill-mode` | Defines final state: `forwards`, `backwards`, `both` |

| Property | Description |
|---|---|
| `animation-play-state` | `running` or `paused` |

---

### Looping Animations

```
.pulse {
  animation: pulse 2s infinite;
}

@keyframes pulse {
  0%, 100% {
    transform: scale(1);
  }
  50% {
    transform: scale(1.1);
  }
}
```

---

## Combining Transitions and Animations

Transitions are great for hover and interactive effects. Animations are better for more **dynamic**, self-running effects.

Example using both:

```
.card {
  transition: transform 0.3s;
}

.card:hover {
  transform: scale(1.05);
}

@keyframes fadeIn {
```

```css
  from { opacity: 0; }
  to { opacity: 1; }
}

.card {
  animation: fadeIn 1s ease;
}
```

---

## Summary

- Use **transitions** for smooth changes on hover, focus, etc.
- Use **animations** for keyframe-driven effects like entrance, bounce, etc.
- Keep animations subtle and purposeful — avoid overwhelming the user.