# Animationen mit CSS

## 1. Animationen mit transition

Mit `transition` kann man einfache Animationen erstellen, die eine CSS-Eigenschaft über eine bestimmte Zeit hinweg ändern. Transitions werden häufig für Effekte wie Hover-Zustände genutzt.

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

- **`property`**: Welche Eigenschaft animiert werden soll (z. B. `opacity`, `background-color`).
- **`duration`**: Wie lange die Animation dauert (z. B. `0.5s`).
- **`timing-function`**: Art der Beschleunigung, z. B. `ease`, `linear`, `ease-in`, `ease-out`.
- **`delay`**: Verzögerung, bevor die Animation beginnt.

Beispiel: Farbwechsel bei Hover

```
.box {
    width: 100px;
    height: 100px;
    background-color: blue;
    transition: background-color 0.5s ease;
}

.box:hover {
    background-color: red;
}
```

Hier ändert sich die Hintergrundfarbe der Box von Blau zu Rot, sobald man mit der Maus darüber fährt, und das in einer halben Sekunde.

## 2. Transformationen mit transform

Die `transform`-Eigenschaft wird verwendet, um Elemente im Raum zu verschieben, zu skalieren, zu drehen oder zu verzerren. Sie kann auch in Verbindung mit `transition` und `@keyframes` genutzt werden.

### Häufig verwendete Transformationsarten:

- **`translate(x, y)`**: Verschiebt ein Element.
- **`scale(x, y)`**: Vergrößert oder verkleinert ein Element.
- **`rotate(angle)`**: Dreht ein Element.
- **`skew(x-angle, y-angle)`**: Verzerrt ein Element.

```
.scale-box {
    width: 100px;
    height: 100px;
    background-color: green;
    transition: transform 0.3s ease;
}

.scale-box:hover {
```

```
    transform: scale(1.2);
}
```

➔ Wenn man mit der Maus über die grüne Box fährt, wird sie um 20% größer.

**Beispiel: Verschieben bei Hover**

```
.move-box {
    width: 100px;
    height: 100px;
    background-color: orange;
    transition: transform 0.5s ease;
}

.move-box:hover {
    transform: translateX(100px);
}
```

➔ Hier verschiebt sich die orangefarbene Box um 100 Pixel nach rechts, wenn man mit der Maus darüber fährt.

## 3. Animationen mit @keyframes

`@keyframes` erlaubt es, komplexere Animationen zu definieren, die über mehrere Schritte hinweg ablaufen. Man kann damit einen Ablauf von Zustandsänderungen festlegen, die durch bestimmte Prozentangaben oder Schlüsselwörter (`from`, `to`) gesteuert werden.

```
@keyframes animation-name {
    from {
        /* Initialer Zustand */
    }
    to {
        /* Endzustand */
    }
}
```

*Oder mit Zwischenstufen:*

```
@keyframes animation-name {
    0% {
        /* Zustand am Anfang */
    }
    50% {
        /* Zustand in der Mitte */
    }
    100% {
        /* Zustand am Ende */
    }
}
```

**Beispiel: Eine Box pulsieren lassen**

```
.pulse-box {
    width: 100px;
    height: 100px;
    background-color: purple;
    animation: pulse 2s infinite;
}

@keyframes pulse {
    0% {
        transform: scale(1);
    }
    50% {
        transform: scale(1.2);
    }
    100% {
        transform: scale(1);
    }
}
```

➔ Die Box wird alle 2 Sekunden größer und kleiner, da sie von scale(1) auf scale(1.2) und zurück skaliert wird.

**Beispiel: Bewegung entlang einer Diagonale**

```
.diagonal-box {
    width: 50px;
    height: 50px;
    background-color: crimson;
    animation: move-diagonal 3s ease-in-out infinite;
}

@keyframes move-diagonal {
    0% {
        transform: translate(0, 0);
    }
    50% {
        transform: translate(100px, 100px);
    }
    100% {
        transform: translate(0, 0);
    }
}
```

➔ Diese Animation bewegt die Box entlang einer diagonalen Linie hin und zurück.