🌐 CSS Transitions Complete Guide

This document explains everything you need to know about the CSS `transition` property — including syntax, supported properties, timing functions, usage examples, and best practices.

📌 What is `transition` in CSS?

The `transition` property allows you to **smoothly animate changes** in CSS properties over a specified duration, without using JavaScript.

🔧 Syntax

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

🎯 Transition Properties

Individual Properties

```
.element {
  transition-property: background-color;
  transition-duration: 2s;
  transition-timing-function: ease-in-out;
  transition-delay: 0.5s;
}
```

Commonly Transitioned Properties

- ✨ `background-color`, `color`
- 📏 `width`, `height`
- 🔲 `margin`, `padding`
- 👁️ `opacity`
- 🔄 `transform`
- ⚡ `border`
- 📝 `font-size`

⏱️ Timing Functions

- `linear` → Constant speed
- `ease` → Slow start, fast middle, slow end

- `ease-in` → Slow start
- `ease-out` → Slow end
- `ease-in-out` → Slow start and end
- `cubic-bezier()` → Custom timing

💡 Examples

Basic Button Transition

```
.button {
  background-color: #3498db;
  padding: 10px 20px;
  transition: all 0.3s ease-in-out;
}

.button:hover {
  background-color: #2980b9;
  transform: scale(1.1);
}
```

Multiple Properties Transition

```
.card {
  transition: background-color 2s ease-in-out, transform 0.5s linear,
    opacity 1s ease;
}
```

🎯 Best Practices

1. **Performance Optimization**
    - Prefer `transform` and `opacity` for smooth animations
    - Avoid transitioning layout properties (width, height, padding)
2. **Duration Guidelines**
    - Quick transitions: 200-300ms
    - Medium transitions: 300-500ms
    - Long transitions: 500-1000ms (use sparingly)
3. **Browser Support**

```
.element {
  -webkit-transition: all 0.3s ease;
  -moz-transition: all 0.3s ease;
```

```
  -o-transition: all 0.3s ease;
  transition: all 0.3s ease;
}
```

🚀 Common Use Cases

1. **Navigation Menus**

```
.nav-link {
  color: #333;
  transition: color 0.3s ease;
}

.nav-link:hover {
  color: #007bff;
}
```

2. **Card Hover Effects**

```
.card {
  box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
  transition: all 0.3s ease;
}

.card:hover {
  transform: translateY(-5px);
  box-shadow: 0 5px 15px rgba(0, 0, 0, 0.2);
}
```

⚠ Common Pitfalls to Avoid

1. Over-using transitions
2. Making transitions too slow
3. Transitioning too many properties
4. Not considering performance impact
5. Forgetting fallback for older browsers

🔍 Additional Resources

- MDN Web Docs - CSS Transitions
- CSS Tricks - Transition Guide

🎨 CSS Transforms (2D & 3D)

2D Transforms

1. translate()

```
.element {
  transform: translate(50px, 100px); /* X and Y axes */
  transform: translateX(50px); /* X axis only */
  transform: translateY(100px); /* Y axis only */
}
```

2. scale()

```
.element {
  transform: scale(2, 1.5); /* Width and height */
  transform: scaleX(2); /* Width only */
  transform: scaleY(1.5); /* Height only */
}
```

3. rotate()

```
.element {
  transform: rotate(45deg); /* Rotates clockwise */
  transform: rotate(-45deg); /* Rotates counter-clockwise */
}
```

4. skew()

```
.element {
  transform: skew(20deg, 10deg); /* X and Y axes */
  transform: skewX(20deg); /* X axis only */
  transform: skewY(10deg); /* Y axis only */
}
```

3D Transforms

1. perspective

```
.container {
  perspective: 1000px; /* Viewing depth */
}
```

2. translate3d()

```
.element {
  transform: translate3d(x, y, z);
  transform: translateZ(100px); /* Z axis only */
}
```

3. rotate3d()

```
.element {
  transform: rotate3d(x, y, z, angle);
  transform: rotateX(45deg); /* Around X axis */
  transform: rotateY(45deg); /* Around Y axis */
  transform: rotateZ(45deg); /* Around Z axis */
}
```

4. scale3d()

```
.element {
  transform: scale3d(x, y, z);
  transform: scaleZ(2); /* Z axis only */
}
```

🎯 3D Transform Properties

1. **transform-style**

```
.container {
  transform-style: preserve-3d; /* Maintains 3D space for children */
}
```

2. **backface-visibility**

```
.element {
  backface-visibility: hidden; /* Hides the back side */
}
```

📚 Detailed Transform Properties Explanation

2D Transform Properties

1. translate()

- **Purpose**: Moves an element from its current position
- **Syntax**: `translate(x,y)`
- **Values**:
  - Length units (px, em, rem)
  - Percentages (relative to element size)

```
.element {
  transform: translate(50px, 30px); /* moves 50px right, 30px down */
}
```

2. scale()

- **Purpose**: Changes the size of an element
- **Syntax**: `scale(x,y)`
- **Values**:
  - Numbers (1 = original size)
  - Less than 1 = smaller
  - Greater than 1 = larger

```
.element {
  transform: scale(2); /* doubles size in both dimensions */
  transform: scale(2, 1); /* doubles width, height unchanged */
}
```

3. rotate()

- **Purpose**: Rotates an element clockwise
- **Syntax**: `rotate(angle)`
- **Values**:
  - deg (degrees)
  - turn (1turn = 360deg)
  - rad (radians)

```
.element {
  transform: rotate(45deg); /* rotates 45 degrees clockwise */
  transform: rotate(-1turn); /* rotates full circle counter-clockwise */
}
```

4. skew()

- **Purpose**: Tilts an element along X and/or Y axis

- **Syntax**: `skew(x-angle,y-angle)`
- **Values**: Angles in degrees

```
.element {
  transform: skew(20deg); /* skews horizontally */
  transform: skew(20deg, 10deg); /* skews both horizontally and vertically
*/
}
```

3D Transform Properties

1. perspective

- **Purpose**: Creates depth for 3D transforms
- **Syntax**: `perspective(n)`
- **Values**: Length units (px)

```
.container {
  perspective: 1000px; /* smaller = more intense effect */
}
```

2. translate3d()

- **Purpose**: Moves element in 3D space
- **Syntax**: `translate3d(x,y,z)`
- **Values**: Length units or percentages

```
.element {
  transform: translate3d(50px, 30px, 20px);
  /* moves: right 50px, down 30px, forward 20px */
}
```

3. rotate3d()

- **Purpose**: Rotates element in 3D space
- **Syntax**: `rotate3d(x,y,z,angle)`
- **Values**:
  - x,y,z: 0 or 1 (axis vectors)
  - angle: degrees

```
.element {
  transform: rotate3d(1, 0, 0, 45deg); /* rotates 45deg around X axis */
```

```
  transform: rotateX(45deg); /* same as above */
}
```

Transform Origin

- **Purpose**: Sets the origin point for transformations
- **Default**: 50% 50% (element's center)

```
.element {
  transform-origin: left top; /* origin at top-left corner */
  transform-origin: 100% 0; /* origin at top-right corner */
  transform-origin: 50px 50px; /* origin at specific coordinates */
}
```

Multiple Transforms

- **Purpose**: Combine multiple transform effects
- **Note**: Order matters! Transforms are applied from right to left

```
.element {
  transform: rotate(45deg) scale(1.5) translateX(100px);
  /* 1. translates right 100px
       2. scales up 1.5x
       3. rotates 45 degrees */
}
```

Real-World Examples

1. Hover Card Effect

```
.card {
  transform-origin: center bottom;
  transition: transform 0.3s ease-out;
}

.card:hover {
  transform: scale(1.05) translateY(-10px);
}
```

2. 3D Flip Effect

```
.flip-container {
  perspective: 1000px;
```

```
}

.flipper {
  transform-style: preserve-3d;
  transition: transform 0.6s;
}

.flipper:hover {
  transform: rotateY(180deg);
}
```

3. Image Tilt Effect

```
.image {
  transform-origin: center bottom;
  transition: transform 0.2s;
}

.image:hover {
  transform: rotate(-5deg) scale(1.1);
}
```

💡 Practical Examples

1. Flip Card Effect

```
.card-container {
  perspective: 1000px;
}

.card {
  transform-style: preserve-3d;
  transition: transform 0.6s;
}

.card:hover {
  transform: rotateY(180deg);
}
```

2. 3D Cube

```
.cube {
  transform-style: preserve-3d;
  transform: rotateX(-30deg) rotateY(45deg);
}
```

```
.face {
  position: absolute;
  backface-visibility: hidden;
}

.front {
  transform: translateZ(100px);
}
.back {
  transform: rotateY(180deg) translateZ(100px);
}
.right {
  transform: rotateY(90deg) translateZ(100px);
}
.left {
  transform: rotateY(-90deg) translateZ(100px);
}
.top {
  transform: rotateX(90deg) translateZ(100px);
}
.bottom {
  transform: rotateX(-90deg) translateZ(100px);
}
```

⚠ Best Practices

1. **Performance**
    - Use `transform` instead of changing position properties
    - Enable GPU acceleration with `transform: translateZ(0)`
    - Use `will-change` for better performance
2. **Browser Support**

```
.element {
  -webkit-transform: rotate(45deg);
  -moz-transform: rotate(45deg);
  -ms-transform: rotate(45deg);
  transform: rotate(45deg);
}
```

3. **Accessibility**
    - Ensure transformed content remains readable
    - Provide alternatives for users who experience motion sickness
    - Consider adding `prefers-reduced-motion` media query

❓ CSS Transitions – Interview Questions (Copy-Paste Ready)

🟢 Beginner Level

1. What is the `transition` property in CSS?
2. What are the main components/parameters of a CSS transition?
3. How do you apply a hover animation using CSS transitions?
4. Which units are required for specifying transition duration?
5. Can CSS transitions work without `:hover` or `:focus`?
6. What is the default timing function in CSS transitions?

🟡 Intermediate Level

7. Which CSS properties are best suited for transitions and why?
8. Why should you avoid using `transition: all`?
9. Can the `display` property be transitioned? Why or why not?
10. What does `ease-in-out` do in a transition?
11. What is the difference between `transition` and `animation` in CSS?

🔴 Advanced Level

12. How does `cubic-bezier()` work in CSS transitions?
13. How can you delay a CSS transition and trigger it using JavaScript?
14. Why is `transform: scale()` preferred over changing `width` or `height` in transitions?
15. How do you transition multiple properties with different durations or timings?
16. What are some common reasons why a transition might not work?
17. How can you improve performance while using transitions in large-scale applications?
18. What happens if you set `transition` on a non-animatable property?

❓ CSS `transform` – Interview Questions (In English)

🟢 Beginner Level

1. What is the `transform` property in CSS?
2. Name the different values of the `transform` property.
3. What is the difference between `transform` and `position` properties in CSS?
4. How does `transform` affect the layout of an element in a webpage?
5. What is the syntax for applying a `rotate` transformation in CSS?

🟡 Intermediate Level

6. How do you apply a `scale` transformation using the `transform` property?
7. Explain the difference between `translate()` and `translate3d()` functions in CSS.

8. What does `transform-origin` do in CSS, and how does it affect transformations?
9. How do `transform` and `transition` work together in animations?
10. Can you animate a `transform` property? If so, how?

🔴 Advanced Level

11. What is the difference between `matrix()` and `matrix3d()` in the `transform` function?
12. How do you center an element using `transform` in CSS?
13. Explain how you can apply multiple transformations on an element.
14. What is the impact of using `transform` on performance, and how can you optimize it?
15. How can you use `transform` to create 3D animations in CSS?

✅ Bonus Questions

16. Can you animate `rotate`, `scale`, and `translate` properties individually? How?
17. What happens when you apply `transform: rotate(90deg)` to a block element in CSS?
18. How do you combine multiple transformations (like rotate, scale, and translate) in one line?