## Animations, Transitions, Transformations & Web Accessibility (WCAG)

### CSS TRANSFORMATIONS

1. What are Transformations?
CSS transformations allow you to visually modify the appearance and position of elements—such as moving, rotating, scaling, or skewing them—without altering the normal document flow (meaning surrounding elements will not reflow or shift to accommodate the transformed element). Transformations can be 2D or 3D.

2. Core 2D Transform Functions
Transformations are applied using the transform property combined with one or more functions:
- translate(x, y): Moves an element horizontally (x) and vertically (y) from its original position (e.g., translate(50px, 100px)). Shorthands: translateX(n), translateY(n).
- scale(x, y): Resizes an element. Values greater than 1 enlarge it, while values between 0 and 1 shrink it (e.g., scale(1.2)). Shorthands: scaleX(n), scaleY(n).
- rotate(angle): Rotates an element clockwise or counter-clockwise around a fixed point specified by transform-origin (e.g., rotate(45deg)).
- skew(x-angle, y-angle): Distorts an element along the X and Y axes (e.g., skew(20deg, 10deg)). Shorthands: skewX(n), skewY(n).
- matrix(a, b, c, d, e, f): Combines all 2D transform methods into a single mathematical matrix.

3. Program: Interactive 2D Transformations Showcase

```
<!DOCTYPE html>
<html lang="">
<head>
  <meta charset="UTF-8">
  <title>CSS Transformations Demo</title>
  <style>
    body { font-family: Arial, sans-serif; display: flex; justify-content: center; gap: 20px;
margin-top: 50px; background: #f4f7f6; }
    .box {
      width: 100px; height: 100px; background-color: #3498db; color: white;
      display: flex; align-items: center; justify-content: center; font-weight: bold;
      transition: transform 0.4s ease; text-align: center; border-radius: 8px;
    }
    #translate-box:hover { transform: translate(30px, -30px); background-color: #2980b9;
}
    #scale-box:hover { transform: scale(1.25); background-color: #e74c3c; }
    #rotate-box:hover { transform: rotate(45deg); background-color: #27ae60; }
    #skew-box:hover { transform: skew(15deg, 15deg); background-color: #8e44ad; }
```

```
  </style>
</head>
<body>
  <div class="box" id="translate-box">Translate</div>
  <div class="box" id="scale-box">Scale</div>
  <div class="box" id="rotate-box">Rotate</div>
  <div class="box" id="skew-box">Skew</div>
</body>
</html>
```

## CSS TRANSITIONS

1. What are Transitions?

CSS transitions allow property changes in CSS values to occur smoothly over a specified duration rather than happening instantly. To trigger a transition, a state change must occur (such as a :hover, :focus pseudo-class, or JavaScript dynamic class toggling).

2. The Four Transition Properties

- transition-property: Specifies the name of the CSS property to transition (e.g., background-color, all, transform).
- transition-duration: Defines how long the transition takes (e.g., 0.3s, 500ms).
- transition-timing-function: Specifies the speed curve of the transition effect (e.g., ease, linear, ease-in, ease-out, ease-in-out).
- transition-delay: Defines when the transition will start (e.g., wait 0.2s before running).

Shorthand Property: transition: all 0.3s ease-in-out 0s;

3. Program: Smooth Card Hover Transition

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>CSS Transitions Demo</title>
  <style>
    body { font-family: Arial, sans-serif; display: flex; justify-content: center; align-items: center; height: 100vh; background: #eef2f3; }
    .card {
      width: 300px; padding: 20px; background: white; border-radius: 10px;
      box-shadow: 0 4px 6px rgba(0,0,0,0.1);
      transition: transform 0.3s ease, box-shadow 0.3s ease, background-color 0.3s ease;
    }
    .card:hover {
      transform: translateY(-10px);
      box-shadow: 0 12px 20px rgba(0,0,0,0.15);
      background-color: #fafbfc;
    }
```

```
    .card h3 { margin-top: 0; color: #2c3e50; }
    .card p { color: #7f8c8d; line-height: 1.5; }
  </style>
</head>
<body>
  <div class="card">
    <h3>Interactive Card</h3>
    <p>Hover over this card to see smooth transitions applied to the elevation (box-shadow) and position (translate).</p>
  </div>
</body>
</html>
```

## CSS ANIMATIONS

1. What are Animations?

While transitions require an explicit state trigger (like a hover), CSS Animations can run automatically, loop indefinitely, change multiple CSS properties across multiple distinct stages via Keyframes, and run independently without user interaction.

2. Key Concepts

- @keyframes: Defines the intermediate steps (keyframes) in a CSS animation sequence using percentages (from 0% to 100%) or keywords (from to to).
- Animation Properties: animation-name, animation-duration, animation-timing-function, animation-delay, animation-iteration-count, animation-direction, animation-fill-mode.

3. Program: Complex Loading Spinner & Pulse Animation

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>CSS Animations Demo</title>
  <style>
    body { font-family: Arial, sans-serif; display: flex; flex-direction: column; align-items: center; justify-content: center; height: 100vh; background: #2c3e50; color: white; gap: 40px; }
    .spinner {
      width: 60px; height: 60px;
      border: 6px solid rgba(255,255,255,0.2);
      border-top-color: #3498db;
      border-radius: 50%;
      animation: spin 1s linear infinite;
    }
    @keyframes spin {
      to { transform: rotate(360deg); }
```

```
    }
    .pulse-text {
      font-size: 24px; font-weight: bold;
      animation: pulse 1.5s ease-in-out infinite alternate;
    }
    @keyframes pulse {
      0% { opacity: 0.4; transform: scale(0.95); }
      100% { opacity: 1; transform: scale(1.05); }
      }
  </style>
</head>
<body>
  <div class="spinner"></div>
  <div class="pulse-text">Loading Content, Please Wait...</div>
</body>
</html>
```

## WEB ACCESSIBILITY (WCAG)

1. What is WCAG?
WCAG (Web Content Accessibility Guidelines) are international standards developed by the W3C (World Wide Web Consortium) to make web content accessible to people with a wide range of disabilities.

2. The POUR Principles
- Perceivable: Information and user interface components must be presentable to users in ways they can perceive.
- Operable: User interface components and navigation must be operable (full keyboard navigation, avoiding flashing animations).
- Understandable: Information and operation must be understandable (clear language, helpful error messages).
- Robust: Content must be robust enough to be interpreted reliably by assistive technologies like screen readers.

3. Key WCAG Implementation Rules
- Semantic HTML: Always use proper tags (<nav>, <main>, <button>, <label>) instead of generic  elements.
- Keyboard Accessibility: Every interactive element must be reachable and operable using the Tab key.
- ARIA Attributes: Use Accessible Rich Internet Applications attributes (aria-label, aria-hidden, role) when native HTML semantics are insufficient.
- Color Contrast: Maintain a minimum contrast ratio of 4.5:1 for normal text and 3:1 for large text.
- Reduced Motion: Respect user OS preferences for reduced motion via media queries (@media (prefers-reduced-motion: reduce)).

4. Program: Accessible Form with WCAG Best Practices

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>WCAG Accessible Form Demo</title>
  <style>
    body { font-family: Arial, sans-serif; margin: 40px; line-height: 1.6; color: #111;
background: #fff; }
    .form-group { margin-bottom: 20px; }
    label { display: block; font-weight: bold; margin-bottom: 6px; }
    input { width: 100%; max-width: 400px; padding: 10px; font-size: 16px; border: 2px
solid #555; border-radius: 4px; }
    input:focus { outline: 3px solid #005f73; outline-offset: 2px; border-color: #005f73; }
    button { background-color: #005f73; color: white; padding: 12px 24px; font-size: 16px;
border: none; border-radius: 4px; cursor: pointer; }
    button:hover, button:focus { background-color: #0a9396; }
    @media (prefers-reduced-motion: reduce) {
      * { animation: none !important; transition: none !important; }
    }
  </style>
</head>
<body>
  <main>
    <h1>User Registration Form</h1>
    <p>Please fill out the details below. All fields marked with an asterisk (*) are
required.</p>
    <form action="#" method="POST">
      <div class="form-group">
        <label for="fullname">Full Name <span aria-hidden="true">*</span></label>
        <input type="text" id="fullname" name="fullname" required aria-required="true"
autocomplete="name">
      </div>
      <div class="form-group">
        <label for="email">Email Address <span aria-hidden="true">*</span></label>
        <input type="email" id="email" name="email" required aria-required="true"
autocomplete="email">
      </div>
      <button type="submit">Register Account</button>
    </form>
  </main>
</body>
</html>
```