Let's explore **CSS Transform** and **CSS Transition** in even greater detail with practical examples and advanced techniques.

---

## 1. **CSS Transform (In-Depth)**

The `transform` property lets you apply various visual transformations to elements, including translation, rotation, scaling, skewing, and even 3D transformations.

### Transform Functions:

1. **Translate:**
   Moves an element along the X, Y, and Z axes (in 3D).

```
transform: translate(x, y);
```

- `translateX(50px)` — Moves 50px to the right.
- `translateY(50px)` — Moves 50px down.
- `translate(50px, 50px)` — Moves 50px right and 50px down.

**Example:**

```
<div class="translate-box">Translate</div>
<style>
  .translate-box {
    width: 100px;
    height: 100px;
    background: #f06;
    transition: transform 0.3s ease;
  }
  .translate-box:hover {
    transform: translate(30px, 20px);
  }
</style>
```

2. **Rotate:**
   Rotates an element around its origin.

```
transform: rotate(angle);
```

- `rotate(45deg)` — Rotates 45° clockwise.
- `rotate(-45deg)` — Rotates 45° counterclockwise.

**Example:**

```
<div class="rotate-box">Rotate</div>
<style>
  .rotate-box {
    width: 100px;
    height: 100px;
    background: #3498db;
    transition: transform 0.3s ease;
  }
  .rotate-box:hover {
    transform: rotate(45deg);
  }
</style>
```

3. **Scale:**
   Scales an element's size.

```
transform: scale(x, y);
```

- `scale(1.5)` — Increases the size by 50%.
- `scale(1, 0.5)` — Only scales the height by 50%.

**Example:**

```
<div class="scale-box">Scale</div>
<style>
  .scale-box {
    width: 100px;
    height: 100px;
    background: #2ecc71;
    transition: transform 0.3s ease;
  }
  .scale-box:hover {
    transform: scale(1.5);
  }
</style>
```

4. **Skew:**
   Skews an element along the X or Y axis.

```
transform: skew(x-angle, y-angle);
```

- `skew(20deg)` — Skews along the X-axis.
- `skew(20deg, 10deg)` — Skews both X and Y.

**Example:**

```
<div class="skew-box">Skew</div>
<style>
  .skew-box {
    width: 100px;
    height: 100px;
    background: #e74c3c;
    transition: transform 0.3s ease;
  }
  .skew-box:hover {
    transform: skew(20deg, 10deg);
  }
</style>
```

5. **3D Transformations:**
   You can apply 3D effects using `perspective` and `rotate3d`.

```
transform: perspective(500px) rotateY(45deg);
```

**Example:**

```
<div class="box-3d">3D Rotate</div>
<style>
  .box-3d {
    width: 100px;
    height: 100px;
    background: #9b59b6;
    transition: transform 0.3s ease;
    transform-style: preserve-3d;
  }
  .box-3d:hover {
    transform: perspective(500px) rotateY(45deg);
  }
</style>
```

6. **Combining Multiple Transforms:**
   You can combine multiple transforms in one declaration.

```
transform: translate(50px, 50px) rotate(45deg) scale(1.2);
```

**Example:**

```
<div class="combined-transform">Combine</div>
<style>
  .combined-transform {
    width: 100px;
    height: 100px;
    background: #f39c12;
    transition: transform 0.3s ease;
  }
  .combined-transform:hover {
    transform: translate(30px, 20px) rotate(45deg) scale(1.2);
  }
</style>
```

---

## 2. **CSS Transition (In-Depth)**

The `transition` property enables smooth changes between property values.

### Transition Properties:

1. **Property:**
   Specifies the property to animate.

```
transition: background-color 0.3s ease;
```

2. **Duration:**
   Specifies how long the transition takes.

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

3. **Timing Function:**
   Controls the speed curve.

```
transition-timing-function: ease | linear | ease-in | ease-out | ease-in-out;
```

4. **Delay:**
   Delays the transition.

```
transition-delay: 0.2s;
```

5. **Shorthand:**

```
transition: all 0.3s ease 0.2s;
```

---

Advanced Transition Examples:

1. **Color Change:**

```
<button class="color-btn">Hover Me</button>
<style>
  .color-btn {
    padding: 10px 20px;
    background: #3498db;
    color: white;
    border: none;
    cursor: pointer;
    transition: background-color 0.3s ease;
  }
  .color-btn:hover {
    background-color: #2ecc71;
  }
</style>
```

2. **Slide In/Out:**

```
<div class="slide-box">Slide Me</div>
<style>
  .slide-box {
    width: 100px;
    height: 100px;
    background: #e74c3c;
    transition: transform 0.3s ease;
  }
  .slide-box:hover {
    transform: translateX(100px);
  }
</style>
```

3. **Card Hover Effect:**

```
<div class="card">
  <h3>Hover Me!</h3>
</div>
<style>
  .card {
    width: 200px;
    height: 150px;
    margin: 50px auto;
    background: white;
    border-radius: 15px;
    box-shadow: 0 5px 15px rgba(0, 0, 0, 0.2);
    transition: transform 0.3s ease, box-shadow 0.3s ease;
  }
  .card h3 {
    text-align: center;
    padding-top: 50px;
  }
  .card:hover {
    transform: translateY(-10px) scale(1.05);
    box-shadow: 0 10px 25px rgba(0, 0, 0, 0.3);
  }
</style>
```

4. **Text Animation:**

```
<h1 class="text-animate">Hover Me!</h1>
<style>
  .text-animate {
    font-size: 40px;
    color: #2c3e50;
    transition: color 0.3s ease, transform 0.3s ease;
  }
  .text-animate:hover {
    color: #e74c3c;
    transform: scale(1.2);
  }
</style>
```

---

Let's dive deep into **CSS Shadows** and **Border Radius**. These two properties can make your UI look polished and professional when used well. I'll cover everything from syntax to advanced techniques, with practical examples!

---

## 3. **CSS Shadows (Deep Dive)**

In CSS, you can create shadows for elements and text using `box-shadow` and `text-shadow`.

### A. **Box Shadow**

The `box-shadow` property adds a shadow effect to the entire box (element).

**Syntax:**

```
box-shadow: offsetX offsetY blur-radius spread-radius color inset;
```

**Parameters:**

- **offsetX:** Moves shadow horizontally (positive = right, negative = left).
- **offsetY:** Moves shadow vertically (positive = down, negative = up).
- **blur-radius:** Optional — Controls the blur (higher = softer edges, default = 0).
- **spread-radius:** Optional — Expands or contracts the shadow size (default = 0).
- **color:** Specifies the shadow color (can use `rgba()` for transparency).
- **inset:** Optional — Makes the shadow appear inside the element.

**Example:**

```
<div class="box-shadow">Box Shadow</div>
<style>
  .box-shadow {
    width: 200px;
    height: 100px;
    margin: 50px;
    background: #3498db;
    box-shadow: 10px 10px 20px rgba(0, 0, 0, 0.3);
  }
</style>
```

**Inset Shadow:**

```
box-shadow: inset 5px 5px 10px rgba(0, 0, 0, 0.5);
```

**Multiple Shadows:**
You can apply multiple shadows by separating them with commas.

```
box-shadow: 2px 2px 5px rgba(0, 0, 0, 0.2), -5px -5px 10px rgba(255, 255, 255,
0.8);
```

---

## B. **Text Shadow**

The `text-shadow` property adds shadow effects to text.

**Syntax:**

```
text-shadow: offsetX offsetY blur-radius color;
```

**Example:**

```
<h1 class="text-shadow">Text Shadow</h1>
<style>
  .text-shadow {
    color: #2c3e50;
    text-shadow: 2px 2px 5px rgba(0, 0, 0, 0.3);
  }
</style>
```

**Multiple Text Shadows:**

```
text-shadow: 2px 2px 5px rgba(0, 0, 0, 0.3), -2px -2px 5px rgba(255, 255, 255,
0.8);
```

**Glow Effect:**

```
text-shadow: 0 0 10px #ff0000;
```

---

## 4. **CSS Border Radius (Deep Dive)**

The `border-radius` property rounds the corners of elements, creating smooth curves and circular shapes.

**Syntax:**

```
border-radius: top-left top-right bottom-right bottom-left;
```

**Single Value:** Applies the same radius to all corners.

```
border-radius: 15px;
```

**Two Values:**

- First value applies to top-left and bottom-right.
- Second value applies to top-right and bottom-left.

```
border-radius: 15px 30px;
```

**Four Values:** Each value applies to a specific corner, in clockwise order:

- Top-left → Top-right → Bottom-right → Bottom-left.

```
border-radius: 10px 20px 30px 40px;
```

**Creating Circles:**
For perfect circles, use `50%` on a square element.

```
border-radius: 50%;
```

**Creating Ellipses:**
For ovals, use percentage values on rectangles.

```
border-radius: 50% / 20%;
```

**Example:**

```
<div class="rounded-box">Rounded Box</div>
<style>
  .rounded-box {
    width: 200px;
    height: 100px;
    background: #e74c3c;
    border-radius: 20px;
  }
</style>
```

## ✨ Combining Shadows and Border Radius

```
<div class="card">
  <h3>Card</h3>
</div>

<style>
  .card {
    width: 250px;
    height: 150px;
    background: #fff;
    margin: 50px auto;
    border-radius: 15px;
    box-shadow: 0 5px 15px rgba(0, 0, 0, 0.2);
    transition: box-shadow 0.3s ease;
  }
  .card:hover {
    box-shadow: 0 10px 25px rgba(0, 0, 0, 0.3);
  }
  h3 {
    text-align: center;
    line-height: 150px;
    margin: 0;
    text-shadow: 2px 2px 5px rgba(0, 0, 0, 0.1);
  }
</style>
```

---

## Real-World Example: **Button with Hover Effects**

```
<button class="fancy-btn">Click Me</button>

<style>
  .fancy-btn {
    background: #3498db;
    color: white;
    padding: 15px 30px;
    border: none;
    border-radius: 30px;
    font-size: 16px;
    cursor: pointer;
    box-shadow: 0 5px 15px rgba(0, 0, 0, 0.2);
    transition: background 0.3s, box-shadow 0.3s;
  }
  .fancy-btn:hover {
    background: #2980b9;
    box-shadow: 0 8px 20px rgba(0, 0, 0, 0.3);
  }
</style>
```

---

Let's dive deep into **@keyframes** and the **animation** property in CSS!

---

### 1. **What is @keyframes?**

`@keyframes` is used to define a set of animation steps that describe how an element should change over time. It breaks down the animation into stages and specifies the styles at each stage.

### Syntax:

```
@keyframes animationName {
  0% {
    /* Starting state */
  }
  50% {
    /* Midpoint */
  }
  100% {
    /* End state */
  }
}
```

- `0%` = Starting point.
- `50%` = Midpoint.
- `100%` = Ending point.
- You can use any percentage between `0%` and `100%` to create multiple stages.

### Example:

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

In this example:

- At **0%**, the element is off-screen to the left and invisible.
- At **100%**, it slides into its normal position and becomes visible.

---

## 2. **The animation Property**

The `animation` property is a shorthand for applying animations to elements. It combines several sub-properties into one line.

### Syntax:

```
animation: name duration timing-function delay iteration-count direction
  fill-mode play-state;
```

### Breakdown of Properties:

1. **animation-name:**
   Specifies the name of the `@keyframes` to use.

   ```
   animation-name: slideIn;
   ```

2. **animation-duration:**
   Defines how long the animation takes to complete (e.g., `2s`, `500ms`).

   ```
   animation-duration: 1s;
   ```

3. **animation-timing-function:**
   Controls the speed curve of the animation.

   - `ease` (default): Starts slow, speeds up, then slows down.
   - `linear`: Same speed throughout.
   - `ease-in`: Starts slow, then speeds up.
   - `ease-out`: Starts fast, then slows down.
   - `ease-in-out`: Slow start, fast middle, slow end.
   - `cubic-bezier`: Custom curve.

   ```
   animation-timing-function: ease-in-out;
   ```

4. **animation-delay:**
   Delays the animation start by a specified time.

   ```
   animation-delay: 0.5s;
   ```

5. **animation-iteration-count:**
   Defines how many times the animation repeats.

   - `1` (default): Runs once.
   - `infinite`: Runs forever.

   ```
   animation-iteration-count: infinite;
   ```

6. **animation-direction:**
   Controls the direction of the animation.

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

   ```
   animation-direction: alternate;
   ```

7. **animation-fill-mode:**
   Specifies what styles apply before and after the animation.

   - `none` (default): The element returns to its original state.
   - `forwards`: The element keeps the last frame of the animation.
   - `backwards`: Applies the first frame before the animation starts.
   - `both`: Applies both `forwards` and `backwards`.

   ```
   animation-fill-mode: forwards;
   ```

8. **animation-play-state:**
   Controls whether the animation is running or paused.

   - `running` (default): The animation plays normally.
   - `paused`: Pauses the animation.

   ```
   animation-play-state: paused;
   ```

---

## 3. **Combining Everything:**

Let's combine these properties into a single `animation` declaration:

```
@keyframes bounce {
  0% {
    transform: translateY(0);
  }
  50% {
    transform: translateY(-20px);
  }
  100% {
    transform: translateY(0);
```

```
    }
  }

  .ball {
    width: 100px;
    height: 100px;
    background: #3498db;
    border-radius: 50%;
    animation: bounce 1s ease-in-out infinite alternate;
  }
```

In this example:

- The ball "bounces" continuously.
- It moves upward at `50%` and returns to its original position at `100%`.

**HTML:**

```
<div class="ball"></div>
```

---

## 4. **Practical Examples**

### A. **Loading Spinner:**

```
@keyframes spin {
  0% {
    transform: rotate(0deg);
  }
  100% {
    transform: rotate(360deg);
  }
}

.loader {
  width: 50px;
  height: 50px;
  border: 5px solid #f3f3f3;
  border-top: 5px solid #3498db;
  border-radius: 50%;
  animation: spin 1s linear infinite;
}
```

**HTML:**

```
<div class="loader"></div>
```

B. **Text Color Change:**

```
@keyframes colorChange {
  0% {
    color: red;
  }
  50% {
    color: green;
  }
  100% {
    color: blue;
  }
}

h1 {
  animation: colorChange 3s ease-in-out infinite;
}
```

**HTML:**

```
<h1>Color Changing Text</h1>
```

C. **Button Hover Animation:**

```
@keyframes pulse {
  0% {
    transform: scale(1);
    box-shadow: 0 0 5px rgba(0, 0, 0, 0.1);
  }
  50% {
    transform: scale(1.1);
    box-shadow: 0 0 15px rgba(0, 0, 0, 0.3);
  }
  100% {
    transform: scale(1);
    box-shadow: 0 0 5px rgba(0, 0, 0, 0.1);
  }
}

button {
  padding: 15px 30px;
  background: #e74c3c;
  color: white;
```

```
    border: none;
    border-radius: 30px;
    font-size: 16px;
    cursor: pointer;
    animation: pulse 1.5s ease infinite;
  }
```

**HTML:**

```
<button>Pulse Button</button>
```

---

## 5. **Pro Tips:**

- Use `animation` **shorthand** to simplify your code.
- Use `infinite` iteration for loaders or spinners.
- Combine **multiple animations** on one element by separating them with commas:

```
animation: bounce 1s ease infinite, spin 1s linear infinite;
```

- Use **DevTools** in the browser to fine-tune your animations in real time.

---