BONNIE COMPUTER HUB

Frontend Web Development Course

MODULE 3/WEEK 3: ADVANCED CSS + UI LIBRARIES

SESSION 1: CSS TRANSITIONS & ANIMATIONS

1. Introduction

Objective:

By the end of this lesson, students will understand how to create smooth transitions and engaging animations using CSS to enhance user experience on websites.

Relevance:

In modern web development, static websites are a thing of the past. Users expect interactive and dynamic experiences. CSS transitions and animations allow you to create movement, visual feedback, and engaging effects without relying on JavaScript, making your websites feel more polished and professional.

2. Lesson Overview

Topics Covered:

- CSS Transitions: properties, timing functions, and delays
- CSS Animations: keyframes, properties, and controls
- Best practices for performance and accessibility
- Practical implementation examples

3. Core Content

CSS Transitions

What are CSS Transitions? CSS transitions allow elements to change values over a specified duration, animating the change in property values. Instead of having property changes take effect immediately, you can make the changes occur smoothly over a period of time.

Basic Syntax:

CSS

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

Shorthand Syntax:

CSS

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

Properties You Can Transition: Most CSS properties can be transitioned, including:

- color
- background-color
- width, height
- margin, padding
- opacity
- transform (for movement, scaling, rotation)

Example: Button Hover Effect

CSS

```
.button {
  background-color: #3498db;
  color: white;
  padding: 10px 20px;
  border: none;
  border-radius: 4px;
```

```
  cursor: pointer;
  transition: background-color 0.3s ease;
}

.button:hover {
  background-color: #2980b9;
}
```

Timing Functions:

- ease: Starts slow, speeds up, and then slows down (default)
- linear: Constant speed throughout
- ease-in: Starts slow, then speeds up
- ease-out: Starts fast, then slows down
- ease-in-out: Starts and ends slow, faster in the middle
- cubic-bezier(n,n,n,n): Custom timing function

Example: Different Timing Functions

CSS

```
.box-1 { transition: transform 1s ease; }
.box-2 { transition: transform 1s linear; }
.box-3 { transition: transform 1s ease-in; }
.box-4 { transition: transform 1s ease-out; }
.box-5 { transition: transform 1s ease-in-out; }

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

CSS Animations

What are CSS Animations? While transitions are great for simple state changes, animations allow for more complex, multi-step animations using keyframes.

Basic Syntax:

1. Define the animation using @keyframes:

CSS

```css
@keyframes animation-name {
  0% {
    /* properties at start */
  }
  50% {
    /* properties at middle */
  }
  100% {
    /* properties at end */
  }
}
```

2. Apply the animation to an element:

CSS

```css
.element {
  animation-name: animation-name;
  animation-duration: time;
  animation-timing-function: timing-function;
  animation-delay: time;
  animation-iteration-count: number | infinite;
  animation-direction: normal | reverse | alternate | alternate-reverse;
  animation-fill-mode: none | forwards | backwards | both;
  animation-play-state: running | paused;
}
```

Shorthand Syntax:

CSS

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

Example: Simple Pulsing Button

CSS

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

.pulse-button {
  padding: 10px 20px;
  background-color: #e74c3c;
  color: white;
  border: none;
  border-radius: 4px;
  cursor: pointer;
  animation: pulse 2s infinite ease-in-out;
}
```

Example: Loading Spinner

CSS

```css
@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 2s linear infinite;
}
```

Best Practices

1. **Performance Considerations:**
    - Only animate transform and opacity properties when possible for better performance
    - Avoid animating properties that trigger layout recalculations (like width, height, top, left)
    - Use will-change property to hint browsers about properties that will change
2. **Accessibility:**
    - Consider users who prefer reduced motion:

CSS

```css
@media (prefers-reduced-motion: reduce) {
  .animated-element {
    animation: none;
```

```
    transition: none;
  }
}
```

3. **Don't Overdo It:**
    - Animations should enhance user experience, not distract from it
    - Use purposeful animations that guide users, provide feedback, or draw attention

4. Assignments

Practical Tasks:

1. **Interactive Card Design:** Create a product card that has smooth hover effects using transitions:
    - Scale slightly on hover
    - Change shadow to give "lifting" effect
    - Transition the background color
    - Add a smooth reveal of additional information
2. **Animated Navigation Menu:** Design a navigation menu that uses transitions for hover states and animations for dropdown menus.
3. **Loading Animation:** Create a custom loading animation using CSS keyframes to be displayed while content loads.

Mini-Project: Interactive Story Page

Create a simple webpage that tells a short story with at least 3 animated elements that enhance the storytelling:

- Characters that move or change using CSS animations
- Transition effects for scene changes
- Interactive elements that respond with animations when clicked

5. Knowledge Check

Reflection Questions:

1. How do CSS transitions differ from CSS animations, and when would you use each?
2. Why is it generally better to animate transform and opacity instead of properties like width and left?
3. How can you ensure your animations are accessible to all users?

Quick Quiz:

1. Which CSS function allows you to define custom animation stages?
   a) @animate
   b) @keyframes
   c) @transition
   d) @stages
2. Which of these properties cannot be transitioned?
   a) color
   b) display
   c) opacity
   d) transform
3. To make an animation repeat forever, what value should you use for animation-iteration-count?
   a) repeat
   b) loop
   c) infinite
   d) continuous
4. What's the purpose of the transition-timing-function property?
   a) To set how long the transition takes
   b) To control the acceleration curve of the transition
   c) To delay when the transition starts
   d) To determine which properties will transition

6. Additional Resources

Further Reading:

- CSS Tricks: A Complete Guide to CSS Transitions
- MDN Web Docs: Using CSS Animations
- Web.dev: Animations and Performance

Tools:

- Animista - CSS animation library and generator
- Cubic-Bezier.com - Tool for creating custom timing functions
- Keyframes.app - Visual timeline for creating CSS animations