CS 110

Lab 7: CSS Animations and Positioning

Overview

In this lab, you will create an interactive nature gallery showcasing CSS animations, transitions, and positioning techniques. The lab focuses on creating engaging user interfaces through dynamic visual effects. You'll work with two main components: an interactive image grid with hover effects and a CSS-only carousel.

Getting Started

1. Download these starter files from Canvas and save them to your lab7 folder:
    - nature-gallery.html: HTML structure with some components already set up
    - nature-gallery-styles.css: CSS file with partial styling
    - images folder containing six nature images created by Copilot
2. Preview the starter code using Live Server:
    - If images are not displaying:
        - Verify the images folder is in the same directory as your HTML file
        - Check that image filenames match exactly with the HTML src attributes
        - Make sure all image files were downloaded successfully
3. Review the starter code:
    - Open nature-gallery.html in your code editor
    - Look for TODO comments indicating where you need to add code
    - Open nature-gallery-styles.css in your code editor
    - Find the sections marked with TODO comments
    - Note the existing styles that are already in place
4. Verify file structure:

```
lab7/
├── nature-gallery.html
├── nature-gallery-styles.css
└── images/
    ├── flowers.png
    ├── berries.png
    ├── nature1.png
    ├── nature2.png
```

```
        ├── nature3.png
        └── nature_background.png
```

Task 1: Complete the Grid Section

Step 1: Add Grid Item Styling

In `nature-gallery-styles.css`, you need to add styling for the grid items:

```css
.grid-item {
    position: relative;  /* Creates positioning context for overlay */
    border-radius: 8px;
    overflow: hidden;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
}
```

Understanding `position: relative`:

- Creates a positioning context for child elements

- The grid item remains in its normal document flow

- Serves as a reference point for absolute-positioned children

- Without this, the overlay would position itself relative to the nearest positioned ancestor (which could be the viewport)

- Keeps the overlay contained within each grid item

Step 2: Complete the Overlay Effect

Understanding `position: absolute` for overlays:

Add the missing overlay styling:

```css
.grid-item-overlay {
    position: absolute;  /* Positions overlay relative to grid-item */
    top: 0;             /* Align to top edge of grid-item */
    left: 0;            /* Align to left edge of grid-item */
    width: 100%;        /* Stretch full width of grid-item */
    height: 100%;       /* Stretch full height of grid-item */
    background-color: rgba(0, 0, 0, 0.7);  /* Semi-transparent black */
    color: white;
    display: flex;      /* Makes the overlay a flex container */
    align-items: center;    /* Centers content vertically */
    justify-content: center; /* Centers content horizontally */
    padding: 1rem;
    text-align: center;     /* Centers the text within the flex item */

    /* Understanding display: flex in the overlay:
     * 1. Without flex, vertically centering text would require complex calcu
```

```
lations
     * 2. Flexbox provides a simple way to center content both vertically and
horizontally
     * 3. align-items: center handles vertical alignment
     * 4. justify-content: center handles horizontal alignment
     * 5. This creates perfect centering regardless of text length
     */
    opacity: 0;         /* Initially invisible */
    transition: opacity 0.3s ease;  /* Smooth fade-in effect */
}
```

Why we use `position: absolute` for the overlay:

- Removes the overlay from normal document flow

- Positions it precisely relative to its nearest positioned ancestor (the grid-item)

- Allows it to cover the entire grid item without affecting layout

- Works with opacity for a clean fade-in effect

- Doesn't push other elements around when it appears

The relationship between `relative` and `absolute`:

1. The `position: relative` on the grid item creates a positioning context

2. The `position: absolute` on the overlay makes it position itself relative to the grid item

3. Using `top: 0, left: 0, width: 100%, height: 100%` makes the overlay cover the entire grid item

4. This combination allows each grid item to have its own independent overlay

5. Without this relationship, all overlays would position themselves relative to the entire page

Step 3: Add Missing Overlay Text

In `nature-gallery.html`, add descriptive overlay text for the remaining images:

- Forest Trail: "A serene forest path lined with evergreen trees and wildflowers"

- Mountain Valley: "Majestic mountain peaks rising above a valley filled with wildflowers"

- River Trail: "A winding trail along a crystal-clear mountain stream"

Task 2: Complete the Carousel

Step 1: Add Navigation Controls

In `nature-gallery-styles.css`, first understand how the carousel container creates the positioning context:

```
.carousel {
    position: relative;    /* Creates positioning context */
    overflow: hidden;      /* Clips slides outside viewport */
    width: 100%;
    max-width: 800px;
    height: 500px;
    margin: 0 auto;
}
```

Understanding the carousel container:

1. `position: relative`:

- Creates a positioning context for child elements

- Establishes the boundaries for absolutely positioned children

- Without this, navigation would position relative to the viewport

- Essential for containing the navigation dots

2. `overflow: hidden`:
    - Clips slides that extend beyond the carousel
    - Works with the positioning to create a clean interface
    - Prevents horizontal scrolling during transitions

Then, add the navigation controls styling:

```
.navigation {
    position: absolute;    /* Removes from normal document flow */
    bottom: 20px;         /* Positions 20px from bottom of carousel */
    left: 50%;           /* Moves to horizontal center */
    transform: translateX(-50%);  /* Centers the navigation dots */
    display: flex;       /* Arranges dots horizontally */
    gap: 10px;          /* Spaces between dots */
}
```

Understanding `position: absolute` for carousel navigation:

1. Why Absolute Positioning is Necessary:

- Navigation dots need to overlay the carousel images

- Dots must maintain a fixed position regardless of slide content

- Must not affect the layout or spacing of slides

- Needs to stay at the bottom even when slides change

2. Centering the Navigation:
    - `left: 50%` moves the left edge to center
    - `transform: translateX(-50%)` shifts back by half width
    - This achieves perfect horizontal centering
    - Works dynamically regardless of number of dots
3. What Would Happen Without Absolute Positioning:
    - Navigation would push slides down
    - Dots would appear below the carousel
    - Would break the overlay effect
    - Could cause unwanted scrolling
4. Benefits of This Approach:
    - Clean visual hierarchy
    - Navigation stays fixed during transitions
    - Works responsively at all screen sizes
    - No impact on slide layout
    - Easy to adjust position with bottom/left properties

Step 2: Complete Slide Transitions

Add the missing slide transitions for slides 4-6:

```
#slide4:checked ~ .slides { transform: translateX(-49.998%); }
#slide5:checked ~ .slides { transform: translateX(-66.664%); }
#slide6:checked ~ .slides { transform: translateX(-83.33%); }
```

Understanding the translateX percentages:

1. Basic Carousel Structure:

```
.slides {
    display: flex;
    width: 600%;        /* 6 slides × 100% */
    height: 100%;
}

.slide {
    width: 16.666%;     /* 100% ÷ 6 slides */
    transition: all 0.5s;
}
```

2. How the Percentages Work:
    - Slide 1: 0% (starting position)

        - No translation needed
        - Shows first slide in view
    - Slide 2: -16.666%
        - Moves left by one slide width
        - Calculation: -(100% ÷ 6)
    - Slide 3: -33.332%
        - Moves left by two slide widths
        - Calculation: -(100% ÷ 6 × 2)
    - Slide 4: -49.998%
        - Moves left by three slide widths
        - Calculation: -(100% ÷ 6 × 3)
    - Slide 5: -66.664%
        - Moves left by four slide widths
        - Calculation: -(100% ÷ 6 × 4)
    - Slide 6: -83.33%
        - Moves left by five slide widths
        - Calculation: -(100% ÷ 6 × 5)
3. Why These Specific Numbers:
    - Base unit is 16.666% (100% ÷ 6 slides)
    - Each subsequent slide multiplies this by its position (1-5)
    - Negative values move slides to the left
    - Precise decimals ensure smooth alignment
4. Visual Breakdown:

```
[Slide 1][Slide 2][Slide 3][Slide 4][Slide 5][Slide 6]
↑ 0%    ↑-16.6% ↑-33.3% ↑-50.0% ↑-66.6% ↑-83.3%
```

5. Why This Approach Works:
    - Each slide is exactly one viewport width
    - translateX moves the entire container
    - Percentage-based movement ensures responsiveness
    - Transitions create smooth animations between states

Step 3: Add Missing Radio Buttons

In `nature-gallery.html`, add the remaining radio buttons and navigation labels for slides 4-6:

```
<input type="radio" name="slide" id="slide4">
<input type="radio" name="slide" id="slide5">
<input type="radio" name="slide" id="slide6">
```

```
<!-- In the navigation section -->
<label class="nav-btn" for="slide4"></label>
<label class="nav-btn" for="slide5"></label>
<label class="nav-btn" for="slide6"></label>
```

Testing Your Implementation

Grid Section Testing

1. Hover over each grid item:
    - Images should scale up slightly
    - Text overlay should fade in smoothly
    - Overlay should be centered and readable
    - All six images should have proper descriptions

Carousel Testing

1. Test navigation:
    - All six slides should be accessible
    - Transitions should be smooth
    - Navigation dots should highlight correctly
    - Images should maintain proper aspect ratio

Validation Requirements

HTML Validation

1. Use the W3C Markup Validation Service (https://validator.w3.org):
    - Go to “Validate by File Upload”
    - Upload your nature-gallery.html file
    - Fix any validation errors or warnings

CSS Validation

1. Use the W3C CSS Validation Service (https://jigsaw.w3.org/css-validator/):
    - Go to “File Upload” tab
    - Upload your nature-gallery-styles.css file
    - Fix any validation errors or warnings

Submission Requirements

1. Files to Submit:
    - Completed nature-gallery.html
    - Completed nature-gallery-styles.css
    - Images folder with all required images
2. Code Requirements:
    - All TODO comments addressed

    - Proper indentation
    - Consistent naming conventions
    - Meaningful comments
    - No broken image links

Grading Criteria

- Grid Implementation (45%)
- Carousel Implementation (45%)
- Code Organization and Comments (10%)

Notes

- All images are provided and created by Copilot
- The gallery uses a dark theme with light text
- Transitions are set to 0.3s for optimal smoothness
- The carousel uses pure CSS without JavaScript