Day 3:

Instruction for Students:

1. **Watch all the tutorial videos** provided on the given websites. These will give you a solid understanding of the advanced CSS and JavaScript concepts needed to complete these exercises.
2. **Complete at least 12 exercises** from the above list. You must choose at least 3 exercises from each of the following areas:
    - Advanced CSS: Flexbox, CSS Grid, Media Queries, Animations/Transitions
    - JavaScript Basics: Setting up JS, Syntax, and Variables

Advanced CSS Exercises:

Flexbox Layout Design Exercises:

1. **Create a responsive navigation bar using Flexbox:**
    - Design a navigation bar with links that adjust in layout on smaller screens (e.g., from a horizontal layout to a vertical one).
    - Use `justify-content`, `align-items`, and `flex-wrap` properties to manage the layout.
2. **Create a photo gallery using Flexbox:**
    - Arrange multiple images in a grid-like structure using Flexbox.
    - Ensure that the images are responsive, resizing based on the screen size.
3. **Design a card layout with Flexbox:**
    - Create a responsive grid of cards where each card has an image, title, and description.
    - Use Flexbox to control the spacing and alignment of the cards within the container.
4. **Create a layout with Flexbox that includes a header, main content area, and sidebar:**
    - Use Flexbox to create a page with a header at the top, sidebar to the left, and main content area to the right.
    - Make the layout responsive so that the sidebar moves below the content on smaller screens.
5. **Create a pricing table using Flexbox:**
    - Create a responsive pricing table with three columns that adjust to one or two columns on smaller screens.
    - Style the table items to display pricing and features in a clean, well-aligned way using Flexbox.

CSS Grid Layout Exercises:

1. **Create a two-column page layout using CSS Grid:**
    - Set up a simple layout with a header, a two-column content section, and a footer.
    - Use `grid-template-columns` to control the column layout and `grid-gap` for spacing between the columns.

2. **Design a responsive grid layout for a product catalog:**
    - Create a product catalog with multiple items arranged in a grid.
    - Ensure the grid adjusts to 1 or 2 columns on smaller screens and displays multiple columns on larger screens using media queries.
3. **Create a CSS Grid for a 2D photo gallery:**
    - Create a photo gallery with images arranged in rows and columns using CSS Grid.
    - Make the layout responsive by adjusting the number of columns based on screen size.
4. **Create a layout with multiple sections using CSS Grid:**
    - Create a layout with a header, navigation, content area, and footer.
    - Use CSS Grid to arrange these sections and control their positions and sizing.
5. **Design a calendar using CSS Grid:**
    - Create a monthly calendar layout with days of the week as the header and dates arranged in a grid.
    - Style each grid item to represent a day, and ensure it is responsive.

Media Queries for Responsive Design Exercises:

1. **Create a responsive webpage layout with different background colors for different screen sizes:**
    - Use media queries to change the background color of the webpage based on the viewport width (e.g., blue for desktop, pink for tablet, and yellow for mobile).
2. **Design a webpage with a flexible grid layout:**
    - Create a layout with two columns on larger screens and a single column on smaller screens.
    - Use media queries to adjust the number of columns based on screen size.
3. **Create a responsive layout with Flexbox and media queries:**
    - Create a three-column layout that switches to a single column on small screens (e.g., less than 600px wide).
    - Use Flexbox for layout and media queries to handle the responsive changes.
4. **Build a responsive image gallery:**
    - Create a grid of images that changes the number of columns based on the viewport width.
    - Use media queries to change the grid layout for different screen sizes (e.g., 4 columns for large screens, 2 columns for medium, and 1 column for small screens).
5. **Use media queries to create a mobile-first navigation bar:**
    - Design a navigation bar with a dropdown menu that appears only on small screens.
    - On larger screens, the menu items should be displayed inline.

CSS Animations and Transitions Exercises:

1. **Create a simple hover effect using CSS transitions:**
    - Style a button with a smooth color change when hovered using the transition property.
    - Use :hover to change the background color and transition to animate the change.
2. **Create an animated loading spinner using CSS keyframes:**
    - Use @keyframes to create a rotating spinner animation.
    - Control the animation speed and make it repeat indefinitely.
3. **Create a fade-in effect for an element using CSS animations:**
    - Use @keyframes to animate an element's opacity from 0 to 1, creating a fade-in effect when the page loads.
4. **Animate the movement of an element on the page:**
    - Use CSS keyframes to animate an element's position, such as moving a box from left to right across the page.
    - Use animation to control the timing and ease of the movement.
5. **Create a bouncing ball animation:**
    - Use @keyframes to animate a ball element bouncing up and down.
    - Control the timing function to make the animation smooth.

JavaScript Basics Exercises:

Setting up JavaScript in an HTML File:

1. **Add a JavaScript script to an HTML file to display an alert when the page loads:**
    - Write a simple alert() message in the script that pops up when the page is opened.
2. **Link an external JavaScript file to an HTML page and display a greeting message in the console:**
    - Create an external JavaScript file and link it to your HTML file.
    - Display a message in the browser's console using console.log().
3. **Write an inline script in an HTML file to change the content of a paragraph when clicked:**
    - Write JavaScript directly within an HTML tag to change the text of a paragraph when it is clicked using an onclick event.
4. **Create a simple HTML page with JavaScript that prompts the user for their name and displays a welcome message:**
    - Use the prompt() method to ask the user for their name and then display a personalized message.

JavaScript Syntax and Variables Exercises:

1. **Declare variables using let, const, and var and demonstrate their differences:**
    - Create three variables using let, const, and var, assign them values, and log them to the console.
    - Explain the scope and reassignment rules for each type of variable.
2. **Write a JavaScript function to calculate the area of a rectangle and display the result in the console:**
    - Declare two variables for the length and width of a rectangle, calculate the area, and output the result.
3. **Create a program that swaps the values of two variables and displays the swapped values:**
    - Declare two variables and swap their values using a temporary third variable, then display the new values.
4. **Write a JavaScript function that takes two numbers as arguments and returns their sum:**
    - Create a function that takes two parameters and returns the sum of the numbers, then call the function and display the result.
5. **Create a JavaScript program that converts a string to uppercase and displays it:**
    - Declare a string variable, use the toUpperCase() method to convert it to uppercase, and log the result.