**Lab Practice: Advanced CSS Techniques**

**Part 1: CSS Grid**

A powerful layout system in CSS that allows for the creation of complex designs for web pages using a grid made up of rows and columns. It simplifies the process of arranging and sizing elements, giving you control over how they are positioned within the grid.

**Objective:**

Students will understand and apply CSS Grid to create responsive web layouts.

**Step 1:**

1. Create a new folder for the lab practice.
2. Inside the folder, create an ***index.html*** file and a ***styles.css*** file.

**Step 2:**

1. Create a simple grid layout:

* In ***index.html***, create a structure with a header, main content area, sidebar, and footer.

*<div class="grid-container">*

*<header class="header">Header</header>*

*<aside class="sidebar">Sidebar</aside>*

*<main class="main">Main Content</main>*

*<footer class="footer">Footer</footer>*

*</div>*

1. Apply CSS Grid:

* In ***styles.css***, apply basic styles to set up a grid.

*.grid-container {*

*display: grid;*

*grid-template-areas:*

*"header header"*

*"sidebar main"*

*"footer footer";*

*grid-template-columns: 1fr 3fr;*

*grid-template-rows: auto 1fr auto;*

*height: 100vh;*

*}*

*.header { grid-area: header; background: lightblue; }*

*.sidebar { grid-area: sidebar; background: lightgray; }*

*.main { grid-area: main; background: white; }*

*.footer { grid-area: footer; background: lightblue; }*

1. Link the ***CSS file*** in the ***HTML head:***

*<link rel="stylesheet" href="styles.css">*

**Step 3:**

1. Make the Layout Responsive:

* Add media queries to adjust the layout for smaller screens.

*@media (max-width: 600px) {*

*.grid-container {*

*grid-template-areas:*

*"header"*

*"main"*

*"sidebar"*

*"footer";*

*grid-template-columns: 1fr;*

*}*

*}*

**Step 4:**

1. Explore Grid Gaps and Alignment:

* Modify the grid to add gaps and adjust item alignment.

*.grid-container {*

*gap: 10px; /\* Adding spaces between grid items \*/*

*}*

*.header, .sidebar, .main, .footer {*

*padding: 20px;*

*}*

1. Try Different grid-template-columns Values:

* Experiment with different column sizes using fixed and fractional units (e.g., grid-template-columns: 200px 1fr;).

**Part 2: CSS Flexbox**

CSS Flexbox is a layout system that makes it easier to create complex designs on web pages. It allows developers to arrange items within a container in a one-dimensional space (either in a row or a column), enabling easy alignment, spacing, and distribution of space among the flex items.

**Objective:**

Students will understand and implement CSS Flexbox to create flexible and responsive web layouts.

**Step 1:**

1. In ***index.html***, create a simple layout with several items in a flex container.

<*div class="grid-container">*

*<header class="header">*

*<nav class="navbar">*

*<div class="nav-item">Home</div>*

*<div class="nav-item">About</div>*

*<div class="nav-item">Services</div>*

*<div class="nav-item">Contact</div>*

*</nav>*

*</header>*

*<aside class="sidebar">Sidebar Content</aside>*

*<main class="main">*

*<div class="item">Main Content 1</div>*

*<div class="item">Main Content 2</div>*

*</main>*

*<footer class="footer">Footer Content</footer>*

*</div>*

1. Apply Flexbox Styles:

* In ***styles.css***, add the flex container properties.

*.navbar {*

*display: flex; /\* Use Flexbox for navigation items \*/*

*justify-content: space-around; /\* Distributes items evenly \*/*

*padding: 15px;*

*}*

*.main {*

*grid-area: main;*

*display: flex; /\* You can also use Flexbox for the main area if needed \*/*

*flex-direction: column; /\* Stack items vertically \*/*

*}*

*.item {*

*background: coral;*

*margin: 10px;*

*flex: 1; /\* Items will grow equally \*/*

*}*

**Part 3: CSS Variables**

**Objective:**

Students will understand and implement CSS variables to enhance maintainability and readability in their CSS code, alongside applying CSS Grid and Flexbox for layouts.

***Syntax:***

*--main-color: blue; /\* Define a variable \*/*

*background-color: var(--main-color); /\* Use a variable \*/*

**Step 1:**

1. In ***styles.css***, define CSS variables within the ***:root*** selector to create global scope:

*:root {*

*--main-color: lightblue;*

*--secondary-color: lightgray;*

*--accent-color: coral;*

*--font-size: 16px;*

*--padding: 20px;*

*--margin: 10px;*

*}*

**Step 2:**

1. Style Grid Layout with CSS Variables:

* Update ***styles.css*** to use the defined CSS variables.

*body {*

*margin: 0;*

*font-family: Arial, sans-serif;*

*}*

*.header {*

*background-color: var(--main-color);*

*padding: var(--padding);*

*}*

*.sidebar {*

*background-color: var(--secondary-color);*

*padding: var(--padding);*

*text-align: left;*

*}*

*.main {*

*padding: var(--padding);*

*}*

*.footer {*

*background-color: var(--main-color);*

*padding: var(--padding);*

*text-align: center;*

*}*

*.navbar {*

*padding: var(--padding);*

*}*

*.item {*

*background: var(--accent-color);*

*padding: var(--padding);*

*}*

**Part 4: CSS Animation and Transition**

**Objective:**

Students will understand and implement CSS animations and transitions to enhance the interactivity and visual appeal of web layouts.

***Syntax:***

*Transition example:*

*.element {*

*transition: all 0.3s ease;*

*}*

*Animation example:*

*@keyframes example {*

*from { background-color: red; }*

*to { background-color: blue; }*

*}*

**Step1:**

1. Add new elements in ***index.html*** to incorporate animations and transitions:

*<aside class=”sidebar”>*

*<div class="animated-item">Hover over me!</div>”*

*</aside>*

**Step 2: Apply CSS Transition**

1. Style the Animated Container and Item in CSS:

* In ***styles.css***, apply styles that will transition properties when they change.

*.animated-item {*

*background-color: var(--accent-color);*

*color: white;*

*padding: var(--padding);*

*border-radius: 5px;*

*transition: background-color 0.3s ease; /\* Add transition effects \*/*

*font-size: var(--font-size);*

*cursor: pointer;*

*}*

*.animated-item:hover {*

*background-color: darkorange; /\* Change background color on hover \*/*

*transform: scale(1.1); /\* Slightly increase size \*/*

*}*

**Step 3: Implement CSS Animation**

1. Define Keyframes for Animation:

* Create a keyframe animation in styles.css to animate the item.

*@keyframes pulse {*

*0% { transform: scale(1); }*

*50% { transform: scale(1.1); }*

*100% { transform: scale(1); }*

*}*

*.animated-item:hover {*

*animation: pulse 1.5s infinite; /\* Apply the pulse animation \*/*

*}*

**Part 5: CSS Preprocessors: SASS and LESS**

A CSS preprocessor is a scripting language that extends the capabilities of traditional CSS by adding features such as variables, nesting, mixins, and functions. These features help developers write more maintainable, organized, and efficient stylesheets.

**Objective:**

Students will learn the fundamentals of CSS preprocessors, specifically SASS and LESS, and how to use their features to write more maintainable and modular CSS.

**I. SASS:**

**Step 1: Install Node.js (If Not Installed)**

1. Before you start, ensure you have Node.js installed on your machine, as it allows you to run the SASS CLI.

* Download and install Node.js from the [official website](https://nodejs.org/).
* Check Node.js installation.

*node -v*

**Step 2: Install SASS Globally**

1. To use the SASS command-line tool, you’ll need to install it globally using npm (Node Package Manager):
2. Open your terminal or command prompt:
   * On Windows: Press Win + R, type cmd, and hit Enter.
   * On macOS: Press Command + Space, type Terminal, and hit Enter.
   * On Linux: Usually, you can open the terminal using Ctrl + Alt + T.
3. Run the following command:

*npm install -g sass*

This command installs SASS globally on your system, allowing you to access the sass command from anywhere.

1. Check npm installation:

*node -v*

**Step 3:**

1. Create a new folder for the SASS and LESS lab practice.
2. Create a ***styles.scss*** file for SASS.
3. Add some SASS code to your styles.scss file.

*$main-color: lightblue;*

*$padding: 20px;*

*.container {*

*padding: $padding;*

*background-color: $main-color;*

*.header {*

*font-size: 24px;*

*}*

*.footer {*

*font-size: 16px;*

*}*

*}*

**Step 4: Convert SCSS to CSS**

To convert the ***styles.scss*** file to CSS, follow these steps:

1. Open your terminal or command prompt.
2. Navigate to the directory where your SCSS file is located. You can use the cd (change directory) command. For example:

*cd path/to/your/project-directory*

1. Run the following SASS command to compile the SCSS file into a CSS file:

*sass styles.scss styles.css*

**Step 5: Check the Output CSS File**

1. After running the command, you should see a new styles.css file in the same directory.
2. Open the styles.css file using a text editor to see the compiled CSS. It should look similar to this:

*.container {*

*padding: 20px;*

*background-color: lightblue;*

*}*

*.container .header {*

*font-size: 24px;*

*}*

*.container .footer {*

*font-size: 16px;*

*}*

**II. LESS**

**Step 1: Install LESS Globally**

1. Open your terminal or command prompt:
   * On Windows: Press Win + R, type cmd, and hit Enter.
   * On macOS: Press Command + Space, type Terminal, and hit Enter.
   * On Linux: Usually, you can open the terminal using Ctrl + Alt + T.
2. Install LESS:
   * In the terminal, run the following command to install LESS globally:

*npm install -g less*

**Step 2: Create Your LESS File**

1. Create a new file called ***styles.less*** and add some LESS code

*@main-color: lightblue;*

*@padding: 20px;*

*.container {*

*padding: @padding;*

*background-color: @main-color;*

*.header {*

*font-size: 24px;*

*}*

*.footer {*

*font-size: 16px;*

*}*

*}*

**Step 4: Convert LESS to CSS**

To convert the ***styles.less*** file to CSS, follow these steps:

1. Open your terminal or command prompt.
2. Navigate to the directory where your SCSS file is located. You can use the cd (change directory) command. For example:

*cd path/to/your/project-directory*

1. Run the following SASS command to compile the SCSS file into a CSS file:

*lessc styles.less styles.css*

**Step 5: Check the Output CSS File**

1. After running the command, you should see a new ***styles.css*** file in the same directory.
2. Open the styles.css file using a text editor to see the compiled CSS. It should look similar to this:

*.container {*

*padding: 20px;*

*background-color: lightblue;*

*}*

*.container .header {*

*font-size: 24px;*

*}*

*.container .footer {*

*font-size: 16px;*

*}*

**Exercise:**

**Scenario: Build a Personal Portfolio Website**

You are a web developer looking to showcase your skills and projects through a personal portfolio website. Your goal is to create a visually appealing and responsive website using advanced CSS techniques such as CSS Grid, Flexbox, CSS Variables, Media Queries and CSS Animations/Transitions.

**Completion Checklist:**

* Create an HTML file with a structured layout that includes a header, navigation bar, main content area, and footer.
* Implement CSS styles using CSS Grid for the project items and Flexbox for the navigation.
* Use CSS Variables for consistent theming throughout the styles.
* Add hover effects for the navigation items and project items.
* Include a simple fade-in animation for the header.
* Implement media queries to ensure a responsive design:
  + Adjust the grid layout for smaller screens (e.g., stack items vertically).
  + Change font sizes or paddings for better readability on mobile devices.
* (Optional) Add JavaScript for interactivity and test its functionality.