Website Designing Assignment

1. Introduction to Website Design and HTML Basics

Theory Assignments

1. Explain the structure of an HTML document and the purpose of DOCTYPE.
2. List and explain at least five common HTML tags used in website design.
3. Describe the difference between block-level and inline elements in HTML.

Assignment 1: Basic HTML Document Structure

- **Objective**: Understand the basic structure of an HTML document and practice using common tags.

Instructions:

1. Create a new HTML file.
2. Add the following basic structure:
    - `<!DOCTYPE html>` declaration.
    - HTML `<html>`, `<head>`, and `<body>` tags.
3. Inside the `<head>` section, add a title for your webpage.
4. Inside the `<body>` section:
    - Add a heading (using `<h1>`).
    - Add a paragraph (using `<p>`).
    - Add a horizontal line (`<hr>`).
    - Add another heading (using `<h2>`).
    - Add a second paragraph.

Assignment 2: Creating Lists and Links

- **Objective**: Practice using list tags and linking to other pages.

Instructions:

1. Inside a new HTML file:
    - Add a heading with the text “My Favorite Hobbies.”
2. Create an ordered list (`<ol>`) listing three hobbies.
3. Below the ordered list, create an unordered list (`<ul>`) with three items related to one of the hobbies.
4. Add a link (`<a>`) that directs to a webpage relevant to one of your hobbies.

Assignment 3: Adding Images and Embedding Videos

- **Objective**: Learn how to add images and embed videos in HTML.

Instructions:

1. Create an HTML page with the title “My Travel Memories.”
2. Inside the `<body>`, add:
    - A heading (using `<h1>`) that says “My Favorite Travel Destination.”
    - An image of a travel destination (use the `<img>` tag with an appropriate `src` and `alt`).
3. Below the image, add a short paragraph describing the place.
4. Embed a YouTube video of the destination using the `<iframe>` tag.

Assignment 4: Building a Simple HTML Form

- **Objective**: Learn how to create forms using HTML.

Instructions:

1. Create an HTML page with the title “Contact Me.”
2. Inside the `<body>`:
    - Add a form with the following input fields:
        - Name (text input)
        - Email (email input)
        - Message (textarea)
        - A submit button
3. Make sure each field has a label using the `<label>` tag for accessibility.

Assignment 5: Creating Tables

- **Objective**: Practice using table tags to organize data.

Instructions:

1. Create a new HTML page with a title “Course Schedule.”
2. Add a table with the following structure:
    - A header row with three columns: “Day,” “Course Name,” “Time.”
    - Add three more rows to represent a course schedule with information for each column.
3. Style the table to have a border and add padding to each cell.

Assignment 6: Creating a Personal Profile Page

- **Objective**: Combine different HTML tags to create a basic profile page.

Instructions:

1. Create an HTML file titled “Profile.”
2. Add the following sections:

    - **Header**: A heading with your name.
    - **Introduction**: A paragraph about yourself.
    - **Skills**: An unordered list of your skills.
    - **Hobbies**: An ordered list of your hobbies.
    - **Contact**: A link to your email address (using `mailto:`) and a phone number.
3. Add a profile picture using the `<img>` tag with an appropriate `alt` text.

Assignment 7: Creating a Navigation Bar

- **Objective**: Practice building a basic navigation bar.

Instructions:

1. Create an HTML page with a title "My Portfolio."
2. Inside the `<body>`, create a navigation bar at the top with the following links:
    - Home
    - About Me
    - Portfolio
    - Contact
3. Use an unordered list (`<ul>`) to create the navigation bar, and style it with inline CSS to display the links horizontally.

Assignment 8: Building a Multi-Page Website

- **Objective**: Practice linking between pages and organizing content across multiple pages.

Instructions:

1. Create a simple 3-page website with the following structure:
    - **Home**: Welcome message and a brief introduction about the website.
    - **About Me**: A paragraph about yourself and an image.
    - **Portfolio**: List of your projects or hobbies with descriptions.
2. Ensure each page has a link to the other two pages using `<a>` tags.
3. Include a heading on each page to identify it (e.g., "Welcome to My Homepage").

Assignment 9: Embedding Media and Maps

- **Objective**: Practice embedding media and external content.

Instructions:

1. Create an HTML page with the title "Explore New York City."
2. Add a paragraph with a short description of New York City.
3. Embed a Google Maps view of New York City using the `<iframe>` tag.
4. Embed a YouTube video about New York City.

Assignment 10: Using Semantic HTML Elements

- **Objective**: Learn and practice using semantic HTML elements.

Instructions:

1. Create an HTML file with the title "Tech News."
2. Use the following HTML5 semantic tags to structure the page:
    - **Header**: Title of the website.
    - **Nav**: Links to "Home," "Latest News," and "Contact."
    - **Main**: A section with a few news articles.
        - Each article should use the `<article>` tag, with headings and short descriptions.
    - **Footer**: Add contact information in the footer.

Assignment 11: Basic HTML Styling with Inline CSS

- **Objective**: Practice adding inline CSS to style HTML elements.

Instructions:

1. Create an HTML page called "Styled Page."
2. Use inline CSS to:
    - Change the background color of the page.
    - Center-align the main heading and set a custom font size.
    - Add padding and a border to an image.
3. Add a paragraph with a custom font color and italicized text.

Assignment 12: Creating an Image Gallery

- **Objective**: Practice working with images and structuring an image gallery layout.

Instructions:

1. Create an HTML page titled "My Gallery."
2. Add a heading that says "My Image Gallery."
3. Insert at least six images into the gallery, each with an `alt` attribute.
4. Use a table or an unordered list to arrange the images into a 2x3 grid layout.

Practical Assignments

1. Create a basic HTML webpage that includes:
    - A header, main content, and footer section.
    - At least three headings, paragraphs, a list, and an image.
2. Build a simple form with fields for name, email, and a submit button.

2. Introduction to CSS and Styling Basics

Theory Assignments

1. Explain the difference between inline, internal, and external CSS.
2. Describe CSS selectors and list the types of selectors (e.g., element, class, id).
3. Discuss the CSS box model and its components.

Assignment 1: Basic Text Styling

- **Objective**: Practice basic text styling using CSS.

Instructions:

1. Create an HTML page with a title and a paragraph.
2. In a separate CSS file or within `<style>` tags:
    - Set the font size of the title to 24px.
    - Change the font family of the paragraph text to Arial.
    - Set the color of the text to navy.
    - Center-align the title and justify the paragraph text.

Assignment 2: Adding Backgrounds and Colors

- **Objective**: Practice using colors and background properties.

Instructions:

1. Create an HTML page with three sections (`<div>` elements) labeled “Introduction,” “Content,” and “Footer.”
2. Apply CSS to:
    - Give each section a different background color (use HEX or RGB values).
    - Add padding and set a specific width for each section.
    - Use a gradient background for the “Introduction” section.
    - Set a light gray background color for the entire webpage.

Assignment 3: Working with Borders and Shadows

- **Objective**: Learn how to use borders, rounded corners, and shadows.

Instructions:

1. Create a simple HTML page with a card-like structure.
2. Add CSS to style the card:
    - Add a border with a color, style, and width of your choice.
    - Round the corners using `border-radius`.
    - Apply a box shadow to give it a raised look.

Assignment 4: Styling Lists

- **Objective**: Practice customizing list styles.

Instructions:

1. Create an HTML page with both an ordered list and an unordered list.
2. In your CSS file:
    - Remove default list markers using `list-style-type: none`.
    - Customize each list by adding a custom marker or symbol.
    - Change the font color and style for list items.

Assignment 5: Creating a Simple Navigation Bar

- **Objective**: Style a basic navigation bar using CSS.

Instructions:

1. Create a navigation bar with links to “Home,” “About,” “Services,” and “Contact.”
2. In your CSS:
    - Make the links horizontal by setting `display: inline-block`.
    - Add padding and a background color to each link.
    - Remove the underline and set a hover effect to change the link color.
    - Center-align the entire navigation bar on the page.

Assignment 6: Using Flexbox for Layout

- **Objective**: Learn to use Flexbox for creating responsive layouts.

Instructions:

1. Create an HTML file with a header, main content area, and a footer.
2. Style the layout using Flexbox:
    - Set up the main container as a Flexbox container.
    - Arrange the elements in a column layout.
    - Center-align the content horizontally.
    - Adjust the layout to a row for screens wider than 768px.

Assignment 7: Grid Layout

- **Objective**: Use CSS Grid for creating a gallery layout.

Instructions:

1. Create an HTML page with a gallery of 6 images.

2. Apply CSS Grid to:
    - Arrange the images in a 3x2 grid layout.
    - Set a gap between the grid items.
    - Adjust the grid to be 2x3 for screens narrower than 600px.
    - Center the entire gallery on the page.

Assignment 8: Styling Buttons

- **Objective**: Practice styling buttons with CSS.

Instructions:

1. Create three buttons in HTML: "Primary," "Secondary," and "Danger."
2. Style each button in CSS:
    - Set distinct background colors for each button.
    - Add padding, rounded corners, and a shadow effect.
    - Change the button color on hover.
    - Add transitions to make the hover effect smooth.

Assignment 9: Responsive Design with Media Queries

- **Objective**: Learn how to make a responsive layout using media queries.

Instructions:

1. Create an HTML page with a main container containing a header, three content sections, and a footer.
2. Apply CSS to:
    - Make the layout a single column on screens narrower than 768px.
    - Change the layout to two columns for screens wider than 768px.
    - Adjust font sizes and padding at different screen widths.

Assignment 10: Styling a Contact Form

- **Objective**: Practice styling forms with CSS.

Instructions:

1. Create a simple contact form with fields for name, email, and message.
2. In your CSS file:
    - Style each form field with padding, border, and background color.
    - Style the submit button with a distinct color, padding, and rounded corners.
    - Add focus effects for input fields.
    - Center the form on the page.

Assignment 11: Using Pseudo-Classes and Pseudo-Elements

- **Objective**: Learn how to use pseudo-classes and pseudo-elements for styling.

Instructions:

1. Create an HTML page with several links and a paragraph.
2. Apply CSS to:
    - Style links with different colors for hover, active, and visited states.
    - Use `::first-line` to style only the first line of the paragraph.
    - Add a decorative `::before` element to headings.

Assignment 12: Using CSS Variables

- **Objective**: Practice using CSS variables to maintain color and theme consistency.

Instructions:

1. Create a simple web page with a header, main content, and footer.
2. Define CSS variables for primary color, secondary color, and text color.
3. Apply the variables to style the background, text, and border colors.
4. Adjust the variables to see how the page's color theme changes.

Assignment 13: Creating Animations

- **Objective**: Learn how to create basic CSS animations.

Instructions:

1. Create an HTML page with a box (using a `<div>`).
2. Create an animation in CSS:
    - Make the box move from left to right.
    - Add a color change as the box moves.
    - Set the animation to loop indefinitely.

Assignment 14: Building a Product Card

- **Objective**: Combine various CSS skills to style a product card.

Instructions:

1. Create a product card in HTML with an image, title, description, and a price.
2. In your CSS:
    - Set the layout using Flexbox or Grid.
    - Style the title, price, and description.
    - Add a hover effect to scale up the card slightly.

- Add a shadow to the card.

Assignment 15: Styling a Landing Page with Custom Fonts and Icons

- **Objective**: Use custom fonts and icons to enhance a page's look.

Instructions:

1. Create an HTML landing page with sections for "Hero," "Features," and "Contact Us."
2. In your CSS:
   - Use Google Fonts to import and apply a custom font to headings.
   - Use Font Awesome (or a similar library) to add icons to the "Features" section.
   - Style each section with padding, background colors, and font adjustments.

Assignment 16: Advanced CSS with Transforms and Transitions

- **Objective**: Practice using CSS transformations and transitions for interactive effects.

Instructions:

1. Create an HTML page with an image gallery.
2. In CSS:
   - Apply a transition effect to each image that changes its scale and adds a shadow when hovered.
   - Use `transform` properties to rotate or scale the images.
   - Ensure the effects are smooth by adding a duration to the transition.

Practical Assignments

1. Style the HTML webpage created in Topic 1 by:
   - Adding colors, font styles, and padding/margin to each section.
   - Experimenting with background colors and borders.
2. Create a navigation menu with horizontal and vertical layouts using CSS.

---

3. Responsive Design with Media Queries

Theory Assignments

1. Define responsive design and its importance in modern web development.
2. Explain the role of media queries in responsive design.
3. Describe how `viewport` settings affect mobile displays.

Practical Assignments

1. Make your HTML page from Topic 1 responsive by:
   - Using media queries to adjust the layout for mobile screens.
   - Hiding/showing elements or adjusting font sizes for smaller screens.

2. Create a simple layout with a sidebar that moves below the main content on screens smaller than 768px.

---

4. Introduction to Bootstrap

Theory Assignments

1. What is Bootstrap, and why is it useful for website design?
2. Explain the Bootstrap grid system and how it helps create responsive layouts.
3. List and explain at least three Bootstrap components (e.g., navbar, cards, buttons).

Practical Assignments

1. Redesign the HTML webpage using Bootstrap to:
    - Implement a responsive grid layout for the header, main content, and footer.
    - Add Bootstrap buttons and a styled form using Bootstrap classes.
2. Create a simple portfolio page with Bootstrap's card component to display portfolio items in a grid format.

---

5. Advanced Bootstrap Components

Theory Assignments

1. Explain how modals and carousels work in Bootstrap.
2. Describe the purpose of utility classes in Bootstrap and give examples.
3. Discuss the importance of customizing Bootstrap variables for unique styling.

Practical Assignments

1. Add a Bootstrap carousel to showcase multiple images on your portfolio page.
2. Create a contact form in a modal that opens on a button click.
3. Customize Bootstrap using variables (e.g., changing primary colors and button styles).

---

6. Introduction to Tailwind CSS

Theory Assignments

1. Explain what Tailwind CSS is and how it differs from traditional CSS frameworks.
2. Describe the concept of utility-first CSS and its advantages.
3. List and explain at least five common Tailwind classes.

Practical Assignments

1. Redesign your HTML webpage using Tailwind CSS by:
    - Applying utility classes for styling and layout adjustments.
    - Ensuring it is responsive with Tailwind's responsive utilities.
2. Create a pricing table using Tailwind with three columns for different pricing options, including buttons and card elements.

7. Advanced Tailwind CSS Components

Theory Assignments

1. Explain how Tailwind's configuration file works and its role in customizing Tailwind.
2. Describe how to create responsive designs using Tailwind's breakpoints.
3. Discuss using custom colors and spacing with Tailwind's configuration.

Practical Assignments

1. Customize Tailwind's configuration to include a new color scheme and spacing values.
2. Create a custom button and card component using Tailwind and your custom configurations.
3. Design a login page layout with Tailwind, including a form and a styled submit button.

8. Advanced CSS: Flexbox

Theory Assignments

1. Explain the purpose of Flexbox and its benefits for responsive design.
2. Describe the main properties of Flexbox (`flex-direction`, `justify-content`, `align-items`).
3. Discuss the difference between `flex-grow`, `flex-shrink`, and `flex-basis`.

Practical Assignments

1. Create a responsive navigation bar with Flexbox that aligns items horizontally and adjusts to a vertical layout on mobile screens.
2. Design a three-column layout using Flexbox with each column equally spaced.
3. Build a product card layout with Flexbox where elements (image, title, description, and price) align and space evenly.

9. Advanced CSS: CSS Grid

Theory Assignments

1. What is CSS Grid, and how is it different from Flexbox?
2. Describe the CSS Grid properties `grid-template-columns` and `grid-template-rows`.
3. Explain the purpose of `grid-area` and how it is used to create complex layouts.

Practical Assignments

1. Create a responsive grid layout for an image gallery with CSS Grid, displaying images in a 3-column layout on desktop and a 1-column layout on mobile.
2. Design a multi-section page layout using CSS Grid, where sections like header, sidebar, main content, and footer are arranged in a grid.
3. Build a blog post layout using Grid, with a main content area, related posts sidebar, and footer section.

10. Advanced CSS: Sass and Less

Theory Assignments

1. Describe the benefits of using a CSS preprocessor like Sass or Less.
2. Explain variables, nesting, and mixins in Sass.
3. Describe the purpose of inheritance and partials in Sass.

Practical Assignments

1. Convert the CSS styling of one of your previous assignments (e.g., portfolio page) to Sass by:
    - Using variables for colors, font sizes, and spacing.
    - Applying nesting and mixins to simplify the CSS structure.
2. Create a Sass file with partials for typography, layout, and colors, then import them into a main stylesheet.
3. Write a mixin in Sass for a responsive card component that can adjust its layout based on screen size.