----:CSS CONCEPTS:----

**CSS Table of Contents**

1. **Introduction to CSS**
   * What is CSS?
   * How CSS Works (CSS Selectors and Declarations)
   * CSS Syntax and Rules
   * External, Internal, and Inline CSS

**Practice Project:** Style a simple HTML page with a title, headings, and paragraphs using different methods (inline, internal, external).

1. **CSS Selectors**
   * Basic Selectors (Tag, Class, ID)
   * Grouping Selectors
   * Universal Selector
   * Attribute Selectors
   * Pseudo-Classes & Pseudo-Elements (e.g., :hover, :first-child)

**Practice Project:** Create a list with various list items and use CSS selectors to apply different styles to specific items.

1. **CSS Box Model**
   * Understanding the Box Model (Content, Padding, Border, Margin)
   * Box-Sizing Property (border-box, content-box)
   * Overflow Property

**Practice Project:** Create a card layout using the box model to control padding, borders, and margins.

1. **CSS Layouts**
   * Display Property (block, inline, inline-block, none)
   * Positioning (Static, Relative, Absolute, Fixed, Sticky)
   * Z-Index
   * CSS Flexbox
   * CSS Grid

**Practice Project:** Build a two-column layout with a header and footer using CSS Grid and Flexbox.

1. **Styling Text**
   * Font Properties (Font Family, Size, Weight, Style)
   * Text Alignment and Decoration
   * Line Height, Letter Spacing, Word Spacing
   * Google Fonts Integration

**Practice Project:** Create a styled blog post page with custom fonts, headers, paragraphs, and links.

1. **Colors and Backgrounds**
   * CSS Colors (Named Colors, Hex, RGB, RGBA, HSL)
   * Backgrounds (Images, Gradients)
   * Background Attachment (Fixed, Scroll)
   * Opacity

**Practice Project:** Design a website section with gradient backgrounds and text overlays.

1. **Borders, Shadows, and Shapes**
   * Border Properties (Width, Style, Color)
   * Border Radius (Rounded Corners)
   * Box Shadows
   * CSS Shapes (Circles, Ovals, etc.)

**Practice Project:** Create a card component with rounded corners, shadows, and different border styles.

1. **CSS Transitions and Animations**
   * Transitions (Transition Property, Duration, Timing Functions)
   * Keyframe Animations (@keyframes)
   * Animation Properties (Timing Functions, Iterations, Direction)

**Practice Project:** Animate a button's hover state with smooth transitions and keyframe animations.

1. **Responsive Web Design**
   * Media Queries
   * Fluid Layouts
   * CSS Units (px, em, rem, vw, vh, %)
   * Responsive Typography

**Practice Project:** Build a simple responsive webpage that adapts to different screen sizes using media queries.

1. **CSS Flexbox in Depth**
   * Flex Container and Flex Items
   * Flex Direction, Justify Content, Align Items, Align Self
   * Order and Flex-Basis

**Practice Project:** Create a responsive navigation bar using Flexbox.

1. **CSS Grid in Depth**
   * Grid Container and Grid Items
   * Grid Template Columns, Rows
   * Grid Gaps, Grid Areas
   * Implicit vs Explicit Grids

**Practice Project:** Design a portfolio layout using CSS Grid.

1. **CSS Variables and Custom Properties**
   * Declaring and Using CSS Variables (var())
   * Global vs Local Scope
   * CSS Variables in Media Queries

**Practice Project:** Create a themed webpage where colors and styles are controlled by CSS variables.

1. **CSS Frameworks (Overview)**
   * Introduction to Bootstrap, Tailwind CSS
   * Using Pre-Built CSS Frameworks
   * Advantages & Disadvantages

**Practice Project:** Rebuild a simple webpage using a CSS framework like Bootstrap.

**Final Project: Complete Responsive Portfolio Website**

**Objective:**

Create a fully functional and responsive portfolio website with multiple sections:

1. Home
2. About
3. Portfolio/Projects
4. Blog
5. Contact

**Requirements:**

* **Responsive Layout**: The website should work well on different screen sizes (mobile, tablet, desktop).
* **Header and Navigation**: Create a sticky navigation bar with links to various sections.
* **Home Section**: Hero image with a welcome message and call-to-action button.
* **About Section**: Short bio with images and styling using Flexbox.
* **Portfolio Section**: Grid layout showcasing your previous projects.
* **Blog Section**: List of recent blog posts (can be dummy text) styled with responsive typography.
* **Contact Form**: A functional contact form using CSS forms (styling only, no backend required).
* **Animations**: Add CSS transitions and animations to elements like buttons and images on hover.
* **Custom Fonts and Colors**: Use custom fonts and colors, leveraging CSS variables for theme management.
* **CSS Grid and Flexbox**: Combine CSS Grid for the portfolio section and Flexbox for layout management elsewhere.

**Tools and Techniques Used in Final Project:**

* **HTML5 and CSS3**
* **Flexbox and CSS Grid**
* **Media Queries for Responsiveness**
* **Custom Properties (CSS Variables)**
* **CSS Transitions and Animations**
* **Google Fonts and Icons**
* **CSS Box Model for Spacing and Layout**
* **Optional: CSS Framework like Bootstrap (if you choose to use it)**

**Submission Guidelines:**

1. The website should be fully functional and responsive.
2. The final project should be hosted on a platform like **GitHub Pages**.
3. Write clean, well-organized CSS, using comments to describe different sections.
4. Optionally, submit a short readme file explaining the structure and design choices.

**Cascading Style Sheets**, is a language used to style and present web documents written in HTML or XML. It allows you to control the layout, colors, fonts, and overall appearance of your web pages, making them visually appealing and user-friendly.

Here are some key points about CSS:

* **Styling**: CSS can change the color, size, and font of text, as well as the spacing and positioning of elements.
* **Layout**: It helps in creating layouts, such as multi-column designs, grids, and flexible layouts for different screen sizes.
* **Consistency**: By using external stylesheets, you can apply consistent styling across multiple web pages.
* **Efficiency**: CSS reduces the need for repetitive styling in HTML, making your code cleaner and easier to maintain.

**CSS selectors** are patterns used to select the elements you want to style in your HTML document. They are a fundamental part of CSS, allowing you to apply styles to specific elements based on various criteria.

**Types of CSS Selectors**

1. **Element Selector**: Selects all elements of a specific type.

**CSS**

p {

color: blue;

}

AI-generated code. Review and use carefully.

This will style all <p> elements with blue text.

1. **ID Selector**: Selects a single element with a specific ID.

**CSS**

#header {

background-color: yellow;

}

AI-generated code. Review and use carefully.

This will style the element with id="header".

1. **Class Selector**: Selects all elements with a specific class.

**CSS**

.highlight {

font-weight: bold;

}

AI-generated code. Review and use carefully.

This will style all elements with class="highlight".

1. **Universal Selector**: Selects all elements.

**CSS**

\* {

margin: 0;

padding: 0;

}

AI-generated code. Review and use carefully.

This will reset the margin and padding for all elements.

1. **Attribute Selector**: Selects elements based on an attribute or attribute value.

**CSS**

input[type="text"] {

border: 1px solid black;

}

AI-generated code. Review and use carefully.

This will style all text input fields.

1. **Pseudo-class Selector**: Selects elements in a specific state.

**CSS**

a:hover {

color: red;

}

AI-generated code. Review and use carefully.

CSS pseudo-classes are used to define the special state of an element. They can be combined with CSS selectors to add an effect to existing elements based on their states. Here is a comprehensive list of commonly used pseudo-classes:

**Structural Pseudo-classes**

1. :first-child: Selects the first child of a parent element.

**CSS**

p:first-child {

color: blue;

}

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

1. :last-child: Selects the last child of a parent element.

**CSS**

p:last-child {

color: green;

}

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

1. :nth-child(n): Selects the nth child of a parent element.

**CSS**

li:nth-child(2) {

color: orange;

}

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

1. :nth-of-type(n): Selects the nth child of a specific type within a parent.

**CSS**

p:nth-of-type(2) {

color: red;

}

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

1. :first-of-type: Selects the first child of a specific type within a parent.

**CSS**

p:first-of-type {

color: purple;

}

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

1. :last-of-type: Selects the last child of a specific type within a parent.

**CSS**

p:last-of-type {

color: brown;

}

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

1. :only-child: Selects an element that is the only child of its parent.

**CSS**

p:only-child {

color: pink;

}

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

1. :only-of-type: Selects an element that is the only child of its type within its parent.

**CSS**

p:only-of-type {

color: teal;

}

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

1. :empty: Selects elements that have no children (including text nodes).

**CSS**

p:empty {

display: none;

}

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

**Link Pseudo-classes**

1. :link: Selects unvisited links.

**CSS**

a:link {

color: blue;

}

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

1. :visited: Selects visited links.

**CSS**

a:visited {

color: purple;

}

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

1. :hover: Selects elements when the user designates them (e.g., by placing a cursor over them).

**CSS**

a:hover {

color: red;

}

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

1. :active: Selects elements when they are being activated (e.g., during a mouse click).

**CSS**

a:active {

color: green;

}

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

**Form Pseudo-classes**

1. :focus: Selects elements that have focus.

**CSS**

input:focus {

border-color: green;

}

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

1. :enabled: Selects enabled form elements.

**CSS**

input:enabled {

background-color: white;

}

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

1. :disabled: Selects disabled form elements.

**CSS**

input:disabled {

background-color: gray;

}

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

1. :checked: Selects checked elements (e.g., checkboxes or radio buttons).

**CSS**

input:checked {

background-color: yellow;

}

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

1. :indeterminate: Selects elements that are in an indeterminate state.

**CSS**

input:indeterminate {

background-color: lightgray;

}

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

1. :placeholder-shown: Selects input elements that are displaying placeholder text.

**CSS**

input:placeholder-shown {

color: gray;

}

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

1. :valid: Selects elements with valid contents.

**CSS**

input:valid {

border-color: green;

}

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

1. :invalid: Selects elements with invalid contents.

**CSS**

input:invalid {

border-color: red;

}

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

1. :in-range: Selects elements with values within a specified range.

**CSS**

input:in-range {

border-color: blue;

}

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

1. :out-of-range: Selects elements with values outside a specified range.

**CSS**

input:out-of-range {

border-color: orange;

}

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

**Other Pseudo-classes**

1. :not(selector): Selects elements that do not match the selector.

**CSS**

p:not(.highlight) {

color: gray;

}

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

1. :root: Selects the document’s root element.

**CSS**

:root {

--main-color: #06c;

}

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

1. :lang(language): Selects elements with a specific language attribute.

**CSS**

p:lang(en) {

color: blue;

}

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

1. :target: Selects the target element of a URL containing a fragment identifier.

**CSS**

#section:target {

background-color: yellow;

}

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

1. :nth-last-child(n): Selects the nth child from the end of a parent element.

**CSS**

li:nth-last-child(2) {

color: red;

}

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

1. :nth-last-of-type(n): Selects the nth child of a specific type from the end of a parent element.

**CSS**

p:nth-last-of-type(2) {

color: blue;

}

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

1. :dir(direction): Selects elements based on the directionality of the text.

**CSS**

p:dir(rtl) {

text-align: right;

}

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

1. :fullscreen: Selects elements that are in fullscreen mode.

**CSS**

:fullscreen {

background-color: black;

}

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

1. :read-only: Selects elements that are read-only.

**CSS**

input:read-only {

background-color: lightgray;

}

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

1. :read-write: Selects elements that are editable.

**CSS**

input:read-write {

background-color: white;

}

This will style links when they are hovered over.

1. **Pseudo-element Selector**: Selects a part of an element.

**CSS**

p::first-line {

font-weight: bold;

}

AI-generated code. Review and use carefully.

This will style the first line of all <p> elements.

1. **Descendant Selector**: Selects elements that are descendants of another element.

**CSS**

div p {

color: green;

}

AI-generated code. Review and use carefully.

This will style all <p> elements inside <div> elements.

1. **Child Selector**: Selects elements that are direct children of another element.

**CSS**

ul > li {

list-style-type: none;

}

AI-generated code. Review and use carefully.

This will style all <li> elements that are direct children of <ul> elements.

1. **Adjacent Sibling Selector**: Selects an element that is immediately preceded by another element.

**CSS**

h1 + p {

margin-top: 0;

}

AI-generated code. Review and use carefully.

This will style the first <p> element immediately following an <h1> element.

1. **General Sibling Selector**: Selects all elements that are siblings of a specified element.

**CSS**

h1 ~ p {

color: purple;

}

AI-generated code. Review and use carefully.

This will style all <p> elements that are siblings of an <h1> element.

1. **Grouping Selector**: Groups multiple selectors to apply the same styles.

**CSS**

h1, h2, p {

color: red;

}

This will style all <h1>, <h2>, and <p> elements with red text.