Basic CSS

Assignment 1: **Selectors and Properties**
- Create a simple webpage with various HTML elements (headings, paragraphs, links, etc.).
- Apply different types of CSS selectors (element, class, id, attribute, pseudo-class, pseudo-element) to style these elements.
- For example, use an element selector to style all paragraphs, a class selector for a group of items, and an id selector for a unique element.

Assignment 2: Color and Backgrounds
- Create a webpage that demonstrates the use of different color properties in CSS.
- Use `color`, `background-color`, and gradients to style elements.
- Experiment with background properties like `background-image`, `background-repeat`, and `background-position`.

Assignment 3: Typography
- Design a webpage that showcases different typography styles.
- Use various font properties like `font-family`, `font-size`, `font-weight`, `font-style`, `line-height`, and `text-transform` to style the text.
- Include examples of different font families, sizes, and styles.

Layout

Assignment 4: Box Model
- Create a webpage to demonstrate the CSS box model.
- Include examples showing the effect of padding, border, and margin on different elements.
- Use `box-sizing` to switch between `content-box` and `border-box` models.

Assignment 5: Flexbox Basics
- Create a simple webpage layout using Flexbox.
- Arrange items in a row, center them, and distribute space between them.
- Use properties like `justify-content`, `align-items`, and `flex-direction`.

Assignment 6: Advanced Flexbox
- Build a more complex layout using Flexbox that includes nested flex containers.
- Implement features like wrapping (`flex-wrap`), alignment (`align-content`), and ordering of flex items (`order`).

Assignment 7: Grid Basics
- Create a simple grid layout with CSS Grid.
- Define rows and columns, and place items in specific grid areas.
- Use properties like `grid-template-rows`, `grid-template-columns`, and `grid-gap`.

Assignment 8: Advanced Grid
- Design a more complex grid layout with nested grids.
- Span items across multiple rows and columns.
- Use grid template areas to define named grid areas and place items within those areas.

Responsive Design

Assignment 9: Media Queries
- Create a responsive webpage that adjusts its layout based on different screen sizes using media queries.
- Include at least three breakpoints (e.g., for mobile, tablet, and desktop).
- Change styles like font size, layout direction, and visibility based on the screen width.

Assignment 10: Responsive Images
- Implement responsive images using the `srcset` attribute and CSS.
- Ensure images look good on various screen sizes and resolutions.
- Use different image sizes and resolutions for different screen conditions.

Positioning

Assignment 11: Positioning
- Create a webpage that demonstrates different CSS positioning techniques.
- Use `static`, `relative`, `absolute`, `fixed`, and `sticky` positioning on various elements.
- Show how these positioning methods affect the layout and flow of the document.

Assignment 12: Z-Index
- Create a webpage with overlapping elements to demonstrate the use of `z-index`.
- Use different values of `z-index` to control the stacking order of elements.
- Experiment with positioning methods to see how they interact with `z-index`.

Animations and Transitions

Assignment 13: Transitions
- Create a webpage that demonstrates CSS transitions.
- Apply transitions to various properties (e.g., `color`, `background-color`, `width`, `height`).
- Use different transition properties like `transition-duration`, `transition-timing-function`, and `transition-delay`.

Assignment 14: Animations
- Create a webpage that includes CSS animations.
- Define keyframes using `@keyframes` and apply these animations to elements.
- Experiment with properties like `animation-duration`, `animation-timing-function`, `animation-delay`, and `animation-iteration-count`.

Advanced Styling

Assignment 15: Transforms
- Create a webpage to demonstrate CSS transforms.
- Use properties like `translate`, `rotate`, `scale`, and `skew` to transform elements.
- Combine multiple transforms to create complex effects.

Assignment 16: Pseudo-Classes and Pseudo-Elements
- Create a webpage to demonstrate the use of pseudo-classes (e.g., `:hover`, `:active`, `:focus`, `:nth-child`) and pseudo-elements (e.g., `::before`, `::after`).
- Use these to create interactive and visually appealing elements.

Forms and UI Components

Assignment 17: Styling Forms
- Create a webpage with various form elements (input fields, checkboxes, radio buttons, select menus).
- Style these form elements using CSS to improve their appearance and usability.
- Include focus and validation styles for better user experience.

Assignment 18: CSS Variables
- Create a webpage that uses CSS variables.
- Define variables for common values like colors, font sizes, and spacing.
- Use these variables throughout your stylesheet to maintain consistency and ease of maintenance.

CSS Frameworks(Optional)

Assignment 19: Using a CSS Framework
- Create a webpage using a popular CSS framework (e.g., Bootstrap, Foundation).
- Use the framework's classes and components to build a responsive layout.
- Customize the framework's styles by overriding its default CSS.

Assignment 20: Building a Custom CSS Framework
- Design a simple custom CSS framework.
- Define a grid system, common UI components (buttons, cards, forms), and utility classes.
- Create documentation and examples to showcase how to use your custom framework.