CSS: A Complete Guide

Compiled by: Muslim Helalee

Introduction to CSS

CSS (Cascading Style Sheets) is used to control the layout and appearance of web pages. It provides the ability to separate content (HTML) from visual design, allowing for cleaner, more maintainable code. The core of CSS lies in its properties, which define how elements should be styled, such as their color, size, spacing, and positioning.

Basic CSS Concepts

---

1. Selectors

CSS selectors target HTML elements for styling. The most common types include:

- **Element Selector**: Targets all elements of a specific type.
- **Class Selector**: Targets elements with a specific class attribute.
- **ID Selector**: Targets the element with a specific ID attribute.
- **Universal Selector**: Targets all elements on a page.
- **Attribute Selector**: Targets elements based on an attribute and its value.
- **Pseudo-Class**: Targets elements in a specific state (e.g., :hover).
- **Pseudo-Element**: Targets specific parts of an element, like ::before or ::after.

---

2. Properties

CSS properties control how elements are displayed. Here are common categories:

- **Color and Background**:
    - color: Sets the text color.
    - background-color: Sets the background color.
    - background-image: Adds a background image.
    - background-repeat, background-size: Controls image repetition and sizing.
- **Text**:

    - font-family, font-size: Sets font family and size.
    - font-weight: Controls the boldness of text.
    - line-height: Sets the space between lines of text.
    - text-align: Aligns text horizontally (left, right, center).
    - text-transform: Changes text case (uppercase, lowercase, etc.).
- **Box Model**:
    - margin: Space outside the element's border.
    - padding: Space inside the element's border.
    - border: Defines the border size, style, and color.
    - width, height: Sets the element's size.
- **Layout**:
    - display: Determines how an element is displayed (e.g., block, inline, none).
    - position: Positions an element (static, relative, absolute, fixed, sticky).
    - top, bottom, left, right: Specifies an element's position when using absolute, fixed, or relative.
    - z-index: Controls stacking order of elements.

---

3. The CSS Box Model

The CSS box model defines the rectangular boxes that HTML elements generate. It consists of:

- **Content**: The content area where text and images are displayed.
- **Padding**: Space inside the element between the content and the border.
- **Border**: A surrounding line around the padding and content.
- **Margin**: Space outside the border.

Understanding and managing the box model is key to controlling the spacing and alignment of elements.

---

4. Display and Positioning

- **Display**:
    - block: Elements take up the full width available.
    - inline: Elements take only as much width as necessary.
    - inline-block: Acts like inline elements but allows width and height to be set.

    - none: Hides the element from the page.
- **Positioning**:
    - static: Normal flow, default positioning.
    - relative: Positioned relative to its normal position.
    - absolute: Positioned relative to its nearest positioned ancestor.
    - fixed: Positioned relative to the viewport.
    - sticky: Switches between relative and fixed based on scroll position.

Intermediate CSS Concepts

---

1. Advanced Selectors

- **Child Selector (>)**: Targets only direct children of an element.
- **Adjacent Sibling Selector (+)**: Targets an element that is immediately next to a specific element.
- **General Sibling Selector (~)**: Targets all sibling elements following a specific element.
- **Attribute Selectors**: Select elements based on their attributes or attribute values.

---

2. CSS Variables (Custom Properties)

CSS variables, also known as **custom properties**, enable more maintainable and reusable code by storing values that can be reused throughout the stylesheet. This is especially useful for things like color schemes and spacing.

---

3. Responsive Design

Responsive design ensures web pages adapt to various screen sizes and devices. Techniques include:

- **Media Queries**: Apply different styles based on screen size or device characteristics.
    - Common breakpoints: 600px (tablets), 900px (small desktops), 1200px (large desktops).
- **Viewport Units**: Use units like vw (viewport width) and vh (viewport height) to create responsive layouts that adjust with the screen size.
- **Fluid Layouts**: Use percentages, em, or rem units instead of fixed units like pixels to allow elements to scale with the page.
- **Flexible Images**: Use max-width: 100% to ensure images scale appropriately within their containers.

---

4. Flexbox

Flexbox simplifies complex layouts by providing a flexible way to align items in one-dimensional layouts.

- **Container Properties**:
    - flex-direction: Determines the direction of flex items (row, column).
    - justify-content: Aligns items horizontally.
    - align-items: Aligns items vertically.
    - flex-wrap: Allows items to wrap onto multiple lines.
- **Item Properties**:
    - flex-grow: Defines how much an item should grow relative to others.
    - flex-shrink: Defines how much an item should shrink relative to others.
    - order: Changes the order of the flex items.

---

5. CSS Grid

CSS Grid is a two-dimensional layout system that provides more control over the layout of elements.

- **Grid Container**:
    - grid-template-columns, grid-template-rows: Define the structure of rows and columns.
    - grid-gap: Sets the spacing between grid items.
- **Grid Item**:
    - grid-column, grid-row: Specify how many columns or rows an item should span.
    - align-self, justify-self: Adjust alignment of individual grid items.

Advanced CSS Concepts

---

1. CSS Transitions and Animations

- **Transitions**: Allow you to smoothly change property values over a given time frame.
    - Example: Change background color on hover with a gradual transition.
- **Animations**: Define keyframes to animate an element from one style to another.

- @keyframes: Define the stages of an animation.
- animation: Specify the animation duration, timing function, and other properties.

---

2. CSS Grid: Advanced Features

- **Named Grid Areas**: You can name grid areas to make layouts more intuitive.
- **Auto-Fill and Auto-Fit**: Automatically adjust grid items to fit available space using the repeat() function in grid-template-columns or grid-template-rows.
- **Minmax()**: Allows for responsive design by setting a minimum and maximum size for grid items.

---

3. Pseudo-Classes and Pseudo-Elements

- **Pseudo-Classes**: Style elements based on a specific state (e.g., :hover, :focus, :nth-child).
- **Pseudo-Elements**: Target specific parts of an element (::before, ::after, ::first-letter).

---

4. CSS Functions

CSS includes several useful functions for dynamic styling:

- **Calc()**: Perform calculations to dynamically determine property values (e.g., calc(100% - 50px)).
- **Min(), Max(), Clamp()**: Define flexible values based on conditions, like minimum and maximum allowed sizes.

---

5. Preprocessors (Sass, LESS)

CSS preprocessors like **Sass** and **LESS** introduce features such as variables, nesting, mixins, and functions to make CSS more maintainable and extendable. Preprocessors compile into standard CSS that can be used in the browser.

---

6. Specificity and Inheritance

CSS follows a **specificity** hierarchy to determine which styles apply when multiple rules target the same element. The order of specificity is as follows:

1. Inline styles.
2. ID selectors.
3. Class selectors, pseudo-classes, and attribute selectors.
4. Element selectors and pseudo-elements.

In addition to specificity, some CSS properties are **inherited** by child elements (e.g., color, font-family), while others are not (e.g., margin, padding).

---

7. Browser Compatibility

Cross-browser compatibility can be a challenge with CSS. Always test styles across multiple browsers and use feature detection (e.g., @supports) to apply specific rules when certain features are available.

Conclusion

This guide covers a broad range of CSS concepts, from the basics of styling elements to advanced layout systems like Flexbox and Grid. Understanding how to structure your styles, manage responsive design, and utilize modern CSS features will empower you to create flexible, maintainable, and visually appealing web pages.