Module 8: CSS Basics

8.1 Introduction to CSS

CSS is the language we use to style a Web page.

What is CSS?

- CSS stands for Cascading Style Sheets
- CSS describes how HTML elements are to be displayed on screen, paper, or in other media
- CSS saves a lot of work. It can control the layout of multiple web pages all at once
- External stylesheets are stored in CSS files

CSS, or Cascading Style Sheets, is a fundamental technology used for styling web pages. It allows developers to control the visual presentation of HTML elements, including layout, colors, fonts, and more. CSS works by targeting HTML elements and applying styling rules to them, which define how they should appear on the screen or in print. With CSS, developers can create visually appealing and responsive web designs, ensuring consistency and clarity across different devices and browsers. CSS is an essential tool in web development, enabling the creation of engaging and user-friendly interfaces that enhance the overall user experience.

8.2 CSS Syntax

CSS, or Cascading Style Sheets, uses a simple syntax to define styling rules for HTML elements. Here's the basic syntax along with an example:

```
selector {
   property1: value1;
   property2: value2;
}
```

Selector: Selects the HTML element(s) to which the styling rules should be applied.
Property: Defines the aspect of the element to be styled (e.g., color, font-size, margin).
Value: Specifies the value of the property (e.g., red, 16px, 10px 20px)

8.3 Selectors and Selecting HTML elements

A CSS selector selects the HTML element(s) you want to style.

We can divide CSS selectors into five categories:

1. Simple selectors (select elements based on name, id, class)
2. Combinator selectors (select elements based on a specific relationship between them)
3. Pseudo-class selectors (select elements based on a certain state)
4. Pseudo-elements selectors (select and style a part of an element)
5. Attribute selectors (select elements based on an attribute or attribute value)

CSS Simple Selectors:

The CSS element Selector:
The element selector selects HTML elements based on the element name.

The CSS id Selector:
The id selector uses the id attribute of an HTML element to select a specific element.

The id of an element is unique within a page, so the id selector is used to select one unique element!

To select an element with a specific id, write a hash (#) character, followed by the id of the element.

The CSS class Selector:
The class selector selects HTML elements with a specific class attribute.

To select elements with a specific class, write a period (.) character, followed by the class name.

The CSS Universal Selector:
The universal selector (*) selects all HTML elements on the page.

The CSS Grouping Selector:
The grouping selector selects all the HTML elements with the same style definitions.

To group selectors, separate each selector with a comma.

CSS Combinator Selectors:

Descendant Selector
The descendant selector matches all elements that are descendants of a specified element.

Child Selector (>)
The child selector selects all elements that are the children of a specified element.

Adjacent Sibling Selector (+)
The adjacent sibling selector is used to select an element that is directly after another specific element.

Sibling elements must have the same parent element, and "adjacent" means "immediately following".

General Sibling Selector (~)
The general sibling selector selects all elements that are next siblings of a specified element.

CSS Pseudo Selectors:

CSS Pseudo-classes:
A pseudo-class is used to define a special state of an element.

For example, it can be used to:

- Style an element when a user mouses over it
- Style visited and unvisited links differently
- Style an element when it gets focus

CSS Pseudo-elements:

A CSS pseudo-element is used to style specified parts of an element.

For example, it can be used to:

- Style the first letter, or line, of an element
- Insert content before, or after, the content of an element

8.4 CSS Properties and Values

CSS (Cascading Style Sheets) includes a wide range of properties and values that allow you to control the appearance and layout of HTML elements on a web page. Here's a list of some common CSS properties and their corresponding values:

Color:

- color: Specifies the text color.
- background-color: Sets the background color of an element.

Typography:

- font-family: Defines the font family for text.
- font-size: Sets the size of the font.
- font-weight: Specifies the thickness of the font (e.g., boldness).
- line-height: Defines the height of each line of text.

Layout:

- width: Sets the width of an element.
- height: Sets the height of an element.
- margin: Sets the margin around an element.
- padding: Sets the padding inside an element.
- display: Specifies the type of box used for an HTML element (e.g., block, inline, inline-block).
- position: Specifies the positioning method (e.g., static, relative, absolute, fixed).

Border:

- border: Sets the border properties (width, style, color) around an element.
- border-radius: Defines the radius of the element's corners.

Alignment:

- text-align: Specifies the alignment of text within an element.
- vertical-align: Aligns an element vertically with respect to its parent.

Visibility:

- visibility: Specifies whether an element is visible or hidden.

Background:

- background-image: Sets the background image of an element.
- background-repeat: Specifies how a background image should be repeated.
- background-position: Sets the starting position of a background image.
- background-size: Specifies the size of a background image.

Animation:

- animation: Specifies the animation properties (name, duration, timing function, delay, iteration count, direction).

Flexbox (for flexible layout):

- display: flex;: Enables Flexbox layout.
- flex-direction: Sets the direction of the flex container's main axis.
- justify-content: Aligns flex items along the main axis.
- align-items: Aligns flex items along the cross axis.

Grid (for grid layout):

- display: grid;: Enables CSS Grid layout.
- grid-template-columns: Defines the number and size of the columns in a grid layout.
- grid-template-rows: Defines the number and size of the rows in a grid layout.
- grid-gap: Sets the size of the gap between grid items.

8.5 Working with Text and Fonts

Choosing the right font for your website is important!

Font Selection is Important
Choosing the right font has a huge impact on how the readers experience a website.

The right font can create a strong identity for your brand.

Using a font that is easy to read is important. The font adds value to your text. It is also important to choose the correct color and text size for the font.

Generic Font Families
In CSS there are five generic font families:

1. Serif fonts have a small stroke at the edges of each letter. They create a sense of formality and elegance.
2. Sans-serif fonts have clean lines (no small strokes attached). They create a modern and minimalistic look.
3. Monospace fonts - here all the letters have the same fixed width. They create a mechanical look.
4. Cursive fonts imitate human handwriting.
5. Fantasy fonts are decorative/playful fonts.

8.6 Box Model(Padding, Border, Margin)

The CSS Box Model
In CSS, the term "box model" is used when talking about design and layout.

The CSS box model is essentially a box that wraps around every HTML element. It consists of: content, padding, borders and margins.

Explanation of the different parts:

Content - The content of the box, where text and images appear
Padding - Clears an area around the content. The padding is transparent
Border - A border that goes around the padding and content
Margin - Clears an area outside the border. The margin is transparent
The box model allows us to add a border around elements, and to define space between elements.

8.7 CSS Display and Positioning

The position property specifies the type of positioning method used for an element.

There are five different position values:

- static
- relative
- fixed
- absolute
- sticky

Elements are then positioned using the top, bottom, left, and right properties. However, these properties will not work unless the position property is set first. They also work differently depending on the position value.

position: static;
HTML elements are positioned static by default.

Static positioned elements are not affected by the top, bottom, left, and right properties.

An element with position: static; is not positioned in any special way; it is always positioned according to the normal flow of the page:

position: relative;
An element with position: relative; is positioned relative to its normal position.

Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element.

position: fixed;
An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element.

A fixed element does not leave a gap in the page where it would normally have been located.

position: absolute;
An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).

However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.

position: sticky;
An element with position: sticky; is positioned based on the user's scroll position.

A sticky element toggles between relative and fixed, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like position:fixed).

Module 9: Integrating HTML with CSS

When a browser reads a style sheet, it will format the HTML document according to the information in the style sheet.

Three Ways to Insert CSS
There are three ways of inserting a style sheet:

External CSS
Internal CSS
Inline CSS

9.1 Inline CSS

An inline style may be used to apply a unique style for a single element.

To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.

9.2 Internal CSS

An internal style sheet may be used if one single HTML page has a unique style.

Internal styles are defined within the <style> element, inside the <head> section of an HTML page.

9.3 External CSS

With an external style sheet, you can change the look of an entire website by changing just one file!

External styles are defined within the <link> element, inside the <head> section of an HTML page.

An external style sheet can be written in any text editor, and must be saved with a .css extension.

The external .css file should not contain any HTML tags.

Multiple Style Sheets
If some properties have been defined for the same selector (element) in different style sheets, the value from the last read style sheet will be used.

Cascading Order
What style will be used when there is more than one style specified for an HTML element?

All the styles in a page will "cascade" into a new "virtual" style sheet by the following rules, where number one has the highest priority:

Inline style (inside an HTML element)
External and internal style sheets (in the head section)
Browser default
So, an inline style has the highest priority, and will override external and internal styles and browser defaults.

9.4 CSS Reset and Normalize

CSS Reset and Normalize are two techniques used to ensure consistent rendering of HTML elements across different web browsers by removing default browser styling and inconsistencies.

CSS Reset:

- CSS Reset is a technique used to reset the default styles applied by web browsers to HTML elements. It typically involves setting all CSS properties to a common baseline, effectively neutralizing any browser-specific styling.

- CSS Reset aims to provide a clean slate for styling, allowing developers to apply their own custom styles without interference from browser defaults.

- Common CSS Reset techniques include setting margins, padding, and borders to zero, removing list styles, and resetting font properties.

- CSS Reset can help ensure consistency in the appearance of web pages across different browsers, but it may require more effort to restyle elements from scratch.

Normalize.css:

- Normalize.css is a modern alternative to CSS Reset that aims to preserve useful default browser styles while normalizing styles across different browsers.

- Unlike CSS Reset, Normalize.css doesn't remove all default styles but rather ensures that styles are consistent and predictable across different browsers.

- Normalize.css addresses common inconsistencies in browser styling, such as font sizes, line heights, and spacing, making it easier to create consistent and visually appealing web designs.

- Normalize.css provides a more lightweight and less intrusive solution compared to CSS Reset, as it only targets specific elements and properties that need normalization, leaving other styles intact.

In summary, both CSS Reset and Normalize.css serve the purpose of ensuring consistent rendering of HTML elements across different browsers, but they approach it in slightly different ways. CSS Reset completely removes default browser styling, while Normalize.css normalizes styles to achieve consistency while preserving useful defaults.

9.5 Responsive Design Basics

Responsive web design (RWD) is an approach to designing and coding websites that ensures optimal viewing and interaction experiences across a wide range of devices and screen sizes. The goal of responsive design is to create websites that adapt fluidly to different screen resolutions and device capabilities, providing a seamless user experience regardless of whether the site is viewed on a desktop computer, tablet, or smartphone.

Key Principles of Responsive Web Design:

1. Fluid Grid Layouts: Use relative units like percentages (%) instead of fixed units like pixels (px) for layout widths, allowing content to adapt to different screen sizes.
2. Flexible Images and Media: Use CSS techniques like max-width: 100%; to ensure images and media elements scale proportionally and do not overflow their containers on smaller screens.
3. Media Queries: Apply CSS media queries to selectively apply styles based on the characteristics of the device, such as screen width, height, or orientation.