Ministry of SMEs and Startups

KOSME
Korea SMEs and Startups Agency

LIKELION

JOBKOREA

# LESSON 01
# HTML / CSS Basics

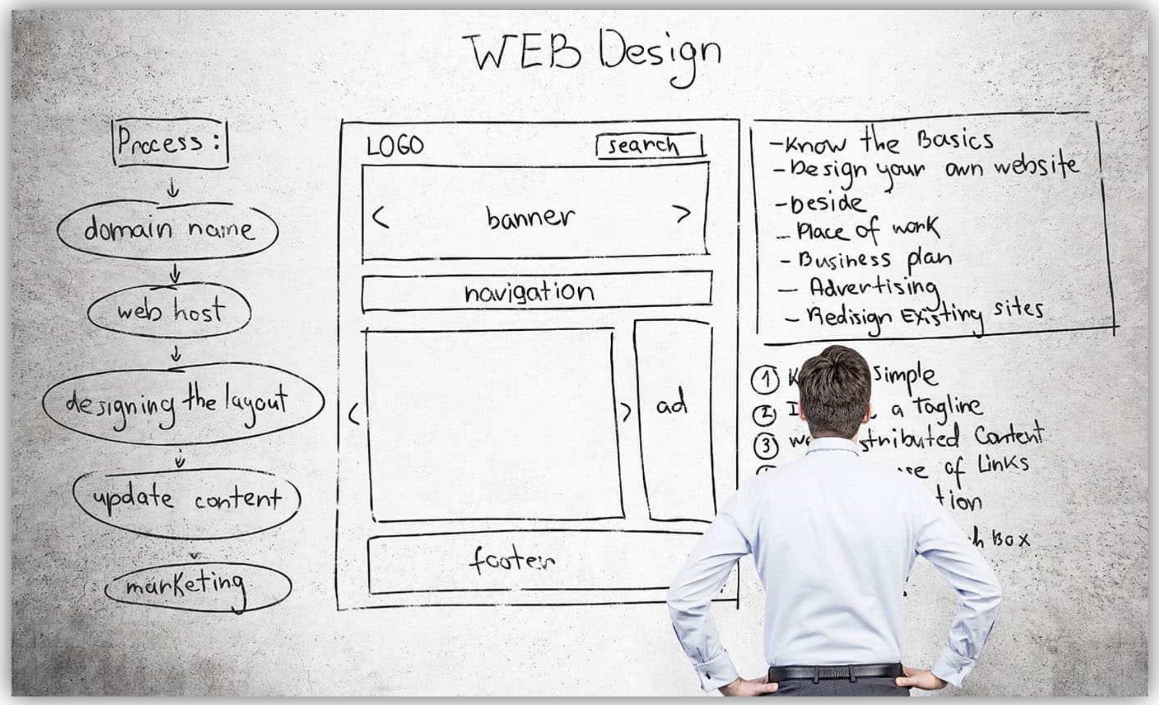
**WEEK 01**

## Objectives

- Web Design Overview
- What is HTML?
- Static vs Dynamic Webpages
- Layout of a Page in HTML5
- HTML Document Structure
- HTML Elements
- What is CSS?
- Introduction to CSS Animation
- Responsive UI with CSS

## Web Design Overview

- ❖ Target audience
- ❖ Multi-platform display
- ❖ Graphics
- ❖ Color
- ❖ Typography
- ❖ Sitemap
- ❖ Wireframe
- ❖ Mockup
- ❖ Prototype



## Web Design Overview

- Website purpose:
  - **Define Goals:** Identify the website's primary objectives (e.g., inform, sell, engage).
  - **Target Audience:** Understand user needs, demographics, and behaviors.
  - **Functionality:** Ensure features align with purpose (e.g., e-commerce, blog, portfolio).
  - **User Experience:** Prioritize intuitive navigation and accessibility.
  - **Brand Identity:** Reflect the brand’s values, tone, and visual style.

## What is HTML?

- ❖ Foundation of Web: HTML structures content, forming the backbone of all websites.
- ❖ Universal Standard: Supported by all browsers, ensuring consistent rendering.
- ❖ Semantic Structure: Enhances accessibility and SEO with meaningful tags.
- ❖ Easy to Learn: Simple syntax, ideal for beginners in web development.
- ❖ Integrates with CSS/JS: Combines with styling and interactivity for dynamic pages.

## Static vs. Dynamic Webpages

❖ **Static Pages**
Fixed content delivered as-is. Pre-built HTML, CSS, and JavaScript files without server-side processing.

❖ **Dynamic Pages**
Content generated by server-side scripts and databases. Displays personalized or real-time information to users.

❖ **Interactivity**
Static pages offer limited user interaction. Dynamic pages enable rich, personalized user experiences.

❖ **Data Source**
Static content comes from fixed files. Dynamic pages connect to databases, APIs, and user inputs for data.

❖ **Use Cases**
Static for portfolios or brochures.
Dynamic for e-commerce, social media, and complex web applications.

## Layout of a Page in HTML5

- ❖ **Semantic Structure**: Use <header>, <nav>, <main>, <aside>, <footer> for meaningful layouts.
- ❖ **Container Elements**: <div> and <section> group content for styling and organization.
- ❖ **Navigation**: <nav> ensuring accessible site navigation.
- ❖ **Content Areas**: <main> for primary content, <article> for self-contained sections.
- ❖ **Responsive Design**: Combine HTML5 with CSS (Flexbox/Grid) for adaptive layouts.

# HTML Document Structure (#1)

❖ **HTML Document Structure Overview**
HTML documents are structured using specific elements to define the content and layout.

❖ **The Doctype Declaration**
The <!DOCTYPE html> declaration defines the document type and version (HTML5).

❖ **Root Element: <html>**
The root element of the HTML document that wraps all content.

❖ **Head Section: <head>**
Contains metadata such as title, character set, links to stylesheets, and scripts.

❖ **Title Element: <title>**
Specifies the title of the document, displayed on the browser tab.

## HTML Document Structure (#2)

- **Body Section: <body>**
  Contains all the content that is visible on the webpage, such as text, images, and links.
- **HTML Elements**
  HTML elements are enclosed in opening and closing tags, like <h1> or <p>.
- **Attributes**
  Elements can have attributes like id, class, style to provide additional information.
- **Nesting of Elements**
  HTML elements can be nested inside one another to create complex structures.
- **Closing Tags**
  Most HTML elements require closing tags to ensure proper structure and rendering.

## HTML Elements Overview

❖ **What are HTML Elements?**

HTML elements are the building blocks of an HTML document. They define the structure and content.

❖ **Basic Structure of an Element**

An element consists of an opening tag, content, and a closing tag. For example: <p>Content</p>.

❖ **Types of Elements**

HTML elements can be structural (e.g., <div>, <header>) or inline (e.g., <span>, <a>).

## Common HTML Elements

- **Text Elements**

  <h1>, <h2>, <p>, <strong>, and <em> are commonly used for headings, paragraphs, and text formatting.

- **Link and Image Elements**

  <a> defines hyperlinks, while <img> is used to embed images.

- **List Elements**

  <ul>, <ol>, and <li> are used to create unordered and ordered lists.

## HTML Form Elements

❖ **Form Elements Overview**

Forms allow users to submit data to a server. Key elements include <input>, <textarea>, and <button>.

❖ **Input Types**

The <input> element supports various types like text, password, email, and checkbox.

❖ **Form Structure**

Forms are created using the <form> element, with actions and methods for submitting data.

## What is CSS?

## What is CSS?

❖ **CSS Overview**

CSS (Cascading Style Sheets) is used to style and design the layout of web pages.

❖ **Separation of Concerns**

CSS separates the structure of HTML content from its presentation (style, color, fonts, etc.).

❖ **How CSS Works**

CSS applies styles to HTML elements using selectors, properties, and values.

## CSS Syntax

❖ **Basic Structure**

CSS consists of selectors and declaration blocks.

Example: selector { property: value; }

❖ **Selectors**

CSS selectors target HTML elements to apply styles (e.g., element, class, ID selectors).

❖ **Declaration**

A declaration includes a property and its value (e.g., color: red;).

## CSS Selectors

- **Element Selector**

  Targets HTML elements by their name. Example: p { color: red; }

- **Class Selector**

  Selects elements with a specific class. Example: .myClass { font-size: 16px; }

- **ID Selector**

  Targets elements with a specific id. Example: #myId { background-color: yellow; }

## CSS Properties for Formatting

❖ **Text Formatting**

Control text appearance using properties like color, font-size, font-family, and text-align.

❖ **Box Model**

Defines element's width, height, padding, margin, and border.

❖ **Backgrounds**

Style element backgrounds with properties like background-color, background-image, and background-repeat.

## Types of CSS

- ❖ **Inline CSS**

  Applied directly within HTML tags using the style attribute.

- ❖ **Internal CSS**

  Defined within the <style> tags in the <head> section of an HTML document.

- ❖ **External CSS**

  Stored in a separate .css file linked to the HTML document using the <link> tag.

## Why Use CSS?

- **Visual Styling**
  CSS allows you to control the appearance of text, colors, spacing, and layout.
- **Consistency Across Pages**
  By using CSS, you can ensure consistent styling across multiple pages of a website.
- **Improved User Experience**
  Well-designed CSS enhances usability and the overall look of the web page.

# CSS Animation

## Introduction to CSS Animation

- **What is CSS Animation?**

  CSS animation allows you to create smooth transitions and movements on web elements without using JavaScript.

- **Why Use CSS Animation?**

  It enhances user experience by adding interactivity and visual effects, making the website more engaging.

- **Key Concepts**

  CSS animations are controlled by @keyframes and the animation property.

# CSS Keyframes

❖ **What are Keyframes?**

Keyframes define the starting and ending points of an animation, along with intermediate steps.

❖ **Basic Syntax**

@keyframes animationName { from { property: value; } to { property: value; } }

Example: @keyframes move { from { left: 0; } to { left: 100px; } }

❖ **Specifying Animation Phases**

You can define multiple points in the animation to control movement and transitions.

## CSS Animation Property

- **Animation Properties**

  animation-name: Specifies the animation defined in @keyframes.

  animation-duration: Defines how long the animation runs.

  animation-timing-function: Controls the speed of the animation (e.g., ease, linear).

- **Complete Syntax**

  Example: animation: move 2s ease-in-out infinite;

## Animation Effects

❖ **Common Animation Effects**

Fade In/Out: Changes opacity.

Slide: Moves an element across the screen.

Bounce: Makes the element bounce back and forth.

❖ **Combining Multiple Animations**

Multiple animations can be applied to the same element by separating them with commas.

## Responsive UI with CSS

## Introduction to Responsive UI with CSS

❖ **What is Responsive UI?**

Responsive UI ensures that a website looks good and functions well across all screen sizes and devices (desktops, tablets, and smartphones).

❖ **Why is Responsive Design Important?**

It improves user experience and accessibility by adapting the layout and content to different screen sizes.

❖ **Key Concepts**

Uses flexible grid layouts, media queries, and flexible images to adjust design for different devices.

## CSS Media Queries

❖ **What Are Media Queries?**

Media queries allow you to apply CSS rules based on the screen size, resolution, or device characteristics.

❖ **Basic Syntax**

Example: @media (max-width: 600px) { ... }

This targets devices with a screen width of 600px or less.

❖ **Common Media Query Features**

Width, height, orientation (portrait/landscape), resolution, and more.

## Flexible Grid Layout

❖ **What is a Grid Layout?**

A grid layout allows you to create a flexible design using rows and columns that adjust to screen size.

❖ **CSS Grid vs Flexbox**

CSS Grid is used for 2-dimensional layouts (both rows and columns), while Flexbox is better for 1-dimensional layouts (rows or columns).

❖ **Basic Grid Example**

Example: display: grid; grid-template-columns: repeat(3, 1fr);

This creates a grid with 3 equal-width columns.

## Flexible Images and Media

- **Responsive Images**

  Use the max-width: 100%; CSS property to make images scale relative to their container.

- **Picture Element**

  The <picture> element allows specifying different images based on device characteristics (e.g., different resolutions for retina displays).

- **Aspect Ratio**

  Maintain the aspect ratio of images using height: auto; and width: 100%;.

# Mobile-First Approach

- **What is Mobile-First Design?**

  Mobile-first design starts by designing for the smallest screen size first and then adding styles for larger screens with media queries.

- **Benefits of Mobile-First**

  Prioritizes performance and load times by ensuring the mobile experience is optimized before scaling to larger devices.

- **Example of Mobile-First CSS**

  Start with the default mobile styles, then use media queries to adjust for larger screens:

```
body { font-size: 14px; }
@media (min-width: 768px) {
   body {font-size: 18px;}
}
```

## Introduction to Bootstrap

## Introduction to Bootstrap

- **Bootstrap**
  a popular CSS framework
- **Key features**
  grid system, components, utilities
- **Website**
  https://getbootstrap.com

## Why Use Bootstrap for Responsive UI?

- ❖ Predefined grid system (12 columns)
- ❖ Ready-to-use responsive components
- ❖ Mobile-first approach
- ❖ Community support and extensive docs

## Bootstrap Grid System

- **What is the Grid?**
  12-column layout for responsive design.
- **Key Classes**
  container, row, col.
- **Breakpoints**
  sm, md, lg, xl, xxl.

## Responsive Containers

- **Container Types**
  container, container-fluid, container-{breakpoint}.
- **Use Cases**
  Fixed-width vs. full-width layouts.

## Video tutorials

❖ https://www.youtube.com/watch?v=z6tJ5ngiF14&list=PLC3y8-rFHvwg6rjbiMadCILrjh7QkvzoQ

## Responsive UI with TailwindCSS

## Introduction to Responsive UI with TailwindCSS

- **What is Responsive UI?**
  UI that adapts to various screen sizes and devices.
- **What is TailwindCSS?**
  Utility-first CSS framework for rapid UI development.
- **Key Features**
  Highly customizable, responsive utilities, no predefined components.

## Tailwind’s Utility-First Approach

- ❖ **What is Utility-First?**
  Apply styles directly via classes, no custom CSS needed.
- ❖ **Benefits**
  Fast prototyping, consistent design.
- ❖ **Common Utilities**
  Margin (m-), padding (p-), text (text-).

## Responsive Design with Tailwind

❖ **Responsive Utilities**
Prefix classes with breakpoints (sm:, md:, lg:, xl:).

❖ **Breakpoints**
Default: 640px, 768px, 1024px, 1280px.
<div class="text-base md:text-lg lg:text-xl"> Responsive Text </div>

## Grid System in Tailwind

- **Grid Layout**
  Use grid, grid-cols-, gap- for responsive grids.
- **Flex vs. Grid**
  Grid for 2D layouts, Flex for 1D.

## Flexbox for Layouts

- **Flex Utilities**
  flex, flex-row, justify-, items-.
- **Responsive Flex**
  Apply breakpoints for dynamic layouts.

## Video tutorials

❖ https://www.youtube.com/watch?v=bxmDnn7lrnk&list=PL4cUxeGkcC9gpXORlEHjc5bgnIi5HEGhw