HTML

What is HTML?
HTML (Hypertext Markup Language) is the standard language for creating webpages. It provides the structure of a webpage using elements enclosed in tags.

Why "HyperText" and "Markup"?
HyperText → Text that contains links to other documents or web pages.
Markup Language → A system for annotating a document so that it can be displayed in a structured way by web browsers.

History of HTML

Early Development (1989-1995)
- 1989: Tim Berners-Lee, a British scientist at CERN, invented HTML while working on the first web browser.
- 1991: The first version of HTML (HTML 1.0) was released with 18 tags.
- 1993: HTML 2.0 introduced more features like tables and forms.

Expansion (1995-2000)
- 1995: HTML 3.2 became the standard, adding support for tables, images, and scripts.
- 1997: HTML 4.0 introduced Cascading Style Sheets (CSS) and better multimedia support.

Modern HTML (2000-Present)
- 2000: XHTML was introduced, making HTML more strict and XML-compliant.
- 2014: HTML5 became the official standard, adding support for audio, video, and animations without plugins like Flash.

Current Version: HTML5 (most widely used).

Why Use HTML?
- Defines the structure of a webpage.
- Allows adding text, images, links, tables, forms, and multimedia content.
- Enables navigation through hyperlinks.
- Works with CSS and JavaScript to enhance design and functionality.
- Ensures compatibility across web browsers.

Who Uses HTML?

- Web Developers → To create websites and apps.
- Content Writers → To structure blog posts.
- SEO Specialists → To optimize web content.

Basic HTML Structure

Every HTML document follows this structure:

```
<!DOCTYPE html
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>My First Webpage</title>
</head>
<body>
    <h1>Welcome to My Website</h1>
    <p>This is a paragraph.</p>
</body>
</html>
```

Breakdown of Components:

Element	Description
<!DOCTYPE html>	Declares the HTML version (HTML5).
<html>	The root element of the HTML document.
<head>	Contains meta information, links to CSS, and the title.
<meta>	Provides metadata like character encoding and viewport settings.
<title>	Sets the webpage title (appears in browser tabs).
<body>	Contains all visible content.

HTML Elements and Their Uses

Elements and Tags

HTML consists of elements represented by tags enclosed within angle brackets (<>). Elements may have attributes to provide additional information.

Example:

```
<p>This is a paragraph.</p>
<a href="https://www.example.com">Visit Example</a>
```

Headings (<h1> to <h6>)

Defines headings for different sections.

```
<h1>Main Heading</h1>
<h2>Subheading</h2>
<h3>Smaller Heading</h3>
```

Paragraph (<p>)

Defines a block of text.

```
<p>This is a paragraph.</p>
```

Links (<a>)

Creates hyperlinks.

```
<a href="https://www.google.com">Visit Google</a>
```

Images (<img>)

Displays an image.

```
<img src="image.jpg" alt="Description of image">
```

Lists (<ul>, <ol>, <li>)

Used to create lists.

```
<ul>
    <li>Apple</li>
    <li>Banana</li>
</ul>
```

Tables (<table>, <tr>, <th>, <td>)

Organizes data in tabular form.

```
<table border="1">
<tr>
    <th>Name</th>
    <th>Age</th>
</tr>
```

```
<tr>
    <td>John</td>
    <td>25</td>
</tr>
</table>
```

Forms (<form> & <input>)

Used to collect user data.

```
<form>
    <label for="name">Name:</label>
    <input type="text" id="name">
    <input type="submit" value="Submit">
</form>
```

Divs and Spans

- is a block-level container used for layout structuring.
- is an inline container used for styling small sections of text.

```
<div style="background-color: lightblue; padding: 10px;">
    <h2>This is inside a div</h2>
</div>

<p>This is a <span style="color: red;">red</span> word in a paragraph.</p>
```

HTML Attributes

Attributes provide extra information about elements.

Attribute	Description	Example
href	Specifies link URL	Google
src	Specifies media source	<img src="image.jpg">
alt	Alternate text for images	<img src="logo.png" alt="Company Logo">
id	Unique identifier	Hello
class	Groups elements for styling	Hello

HTML Semantic Elements (Better for SEO & Accessibility)

Semantic elements clearly describe their purpose.

Element	Purpose
<header>	Represents the top section of a webpage.
<nav>	Contains navigation links.
<section>	Defines a section of content.
<article>	Represents an independent article or post.
<aside>	Defines side content (like a sidebar).
<footer>	Represents the bottom section of a webpage.

Example:

```
<header>
    <h1>My Website</h1>
</header>

<nav>
    <a href="#">Home</a>
    <a href="#">About</a>
</nav>

<section>
    <h2>About Us</h2>
    <p>We provide web development services.</p>
</section>

<footer>
    <p>© 2025 My Website</p>
</footer>
```

HTML5 Features (Why It's Better!)

New Multimedia Support:

- <audio> and <video> tags (no need for Flash).

<video controls>
<source src="video.mp4" type="video/mp4">
</video>

Canvas & SVG (For Graphics)

- Draw shapes and graphics dynamically.

<canvas id="myCanvas"></canvas>

Better Forms (New Input Types)
<input type="email">
<input type="date">

Complete List of HTML Tags

Basic Document Structure Tags

Tag	Description
<!DOCTYPE>	Declares the HTML version (HTML5).
<html>	Root element of an HTML document.
<head>	Contains meta information and links.
<title>	Defines the webpage title.
<body>	Contains the visible webpage content.

Text Formatting Tags

Tag	Description
<h1> to <h6>	Headings (H1 is the largest, H6 is the smallest).
<p>	Defines a paragraph.
 	Inserts a line break.
<hr>	Inserts a horizontal line.
	Bold text (semantic).
<b>	Bold text (non-semantic).
	Italicized text (semantic).
<i>	Italicized text (non-semantic).
<u>	Underlined text.
<mark>	Highlights text.

Links & Navigation Tags

Tag	Description
<a>	Creates a hyperlink.
<nav>	Defines a section for navigation links.

Links & Navigation Tags

Tag	Description
	Creates an unordered (bulleted) list.
	Creates an ordered (numbered) list.
	List item (used inside and).

Table Tags

Tag	Description
<table>	Creates a table.
<tr>	Defines a row in a table.
<th>	Defines a table header.
<td>	Defines a table cell.

Media Tags

Tag	Description
<img>	Displays an image.
<audio>	Embeds an audio file.
<video>	Embeds a video file.

Forms & Input Tags

Tag	Description
<form>	Creates a form.
<input>	Creates an input field.
<textarea>	Creates a multi-line text field.
<button>	Creates a clickable button.
<select>	Creates a dropdown list.

Semantic Tags

Tag	Description
<header>	Defines the header section of a page.
<nav>	Defines a section for navigation links.
<section>	Defines a section of content.
<article>	Defines independent, self-contained content.
<aside>	Defines side content (e.g., sidebar).
<footer>	Defines the footer section of a page.

CSS

CSS (Cascading Style Sheets)

What is CSS?

CSS (Cascading Style Sheets) is a stylesheet language used to style HTML documents. It controls the layout, colors, fonts, and overall presentation of web pages.

Why Use CSS?

1. Separation of Content and Design: HTML defines structure, while CSS handles styling.
2. Consistency: Ensures a uniform look across multiple web pages.
3. Efficiency: One CSS file can style multiple HTML pages.
4. Improved Page Load Speed: Reduces inline styling, making pages load faster.
5. Responsive Design: Enables web pages to be mobile-friendly.
6. Accessibility: Enhances readability and usability for all users.

History of CSS

1. **CSS1 (1996):** Basic styling like colors, fonts, and text alignment.
2. **CSS2 (1998):** Added positioning, media types, and table styling.
3. **CSS3 (1999-Present):** Introduced modularization, animations, flexbox, grid, and more.
4. **CSS4 (Future Concept):** Not officially released but includes advanced selectors and features.

Types of CSS

- Inline CSS (Applied directly to an HTML element)
  Hello World
- Internal CSS (Within the <style> tag inside <head>)
  <style>
  p { color: blue; font-size: 18px; }
  </style>
- External CSS (Using an external .css file)
  <link rel="stylesheet" href="styles.css">

CSS Syntax

```
selector {
    property: value;
}
```

Example:

```
h1 {
    color: blue;
    font-size: 24px;
}
```

CSS Selectors

1) Basic Selectors

- element → Targets specific HTML elements.

```
p { color: red; }
```

- #id → Targets elements by their unique ID.

```
#main { background-color: yellow; }
```

- .class → Targets elements with a specific class.

```
.highlight { font-weight: bold; }
```

2) Grouping and Combinators

- , (Grouping Selector) → Applies styles to multiple elements.

```
h1, h2, h3 { color: blue; }
```

- > (Child Selector) → Targets direct children.

```
div > p { font-size: 16px; }
```

- + (Adjacent Sibling Selector) → Targets immediate next sibling.

```
h1 + p { color: green; }
```

3) Pseudo-Classes

- :hover → Styles an element when hovered over.

```
a:hover { color: red; }
```

- :nth-child(n) → Targets the nth child of a parent.

```
tr:nth-child(even) { background-color: #f2f2f2; }
```

4) Pseudo-Elements

- ::before → Inserts content before an element.

p::before { content: "★ "; }

- ::after → Inserts content after an element.

p::after { content: " ✓"; }

CSS Properties

1) Text & Font Properties

font-size: 16px;
font-family: Arial, sans-serif;
color: #333;
text-align: center;
line-height: 1.5;
letter-spacing: 1px;

2) Background & Borders

background-color: lightblue;
background-image: url('image.jpg');
border: 2px solid black;
border-radius: 10px;
box-shadow: 5px 5px 10px gray;

3) Layout & Positioning

display: flex;
flex-direction: column;
justify-content: center;
align-items: center;
margin: 20px;
padding: 15px;
position: absolute;
top: 50px;
left: 100px;

4) Responsive Design

@media (max-width: 768px) {
body { background-color: lightgray; }
}

CSS Layouts

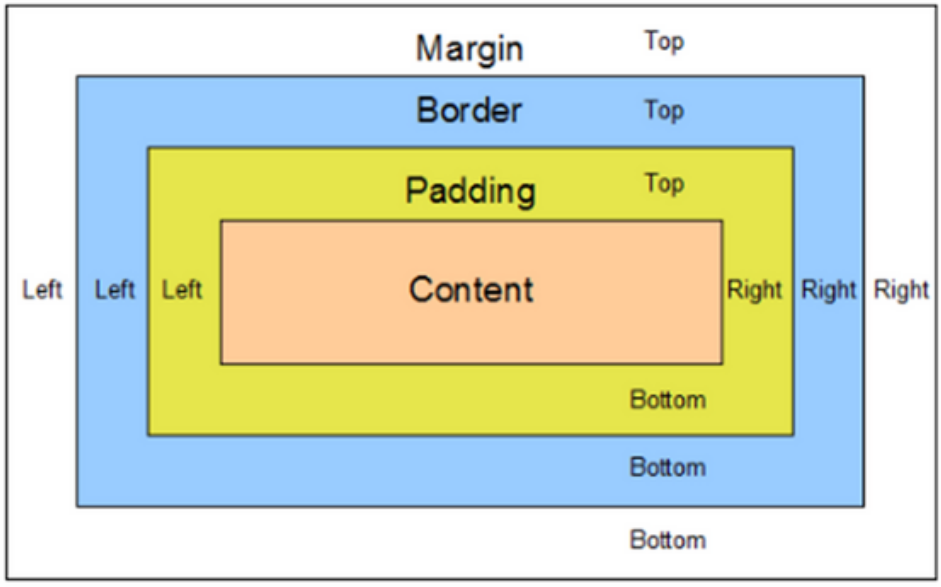
1) Flexbox (1D Layout - Row or Column)

```
.container {
display: flex;
flex-direction: row;
justify-content: space-between; }
```

2) Grid (2D Layout - Rows & Columns)

```
.container {
display: grid;
grid-template-columns: 1fr 2fr;
gap: 10px;
}
```

Margin, Border, Padding



JS

Introduction to JavaScript

- JavaScript (JS) is a programming language used for web development.
- It runs in the browser and allows interactive web pages.
- It is used for animations, form validation, event handling, and more.

History of JavaScript

- **1995** – Created by Brendan Eich at Netscape, originally named Mocha.
- **1996** – Renamed JavaScript to market it with Java (but they are different).
- **1997** – Standardized as ECMAScript (ES).
- **2009** – Node.js introduced, allowing JavaScript to run on servers.
- **2015 (ES6)** – Introduced modern features like let, const, and arrow functions.

Why Use JavaScript?

✅ Adds interactivity to web pages.
✅ Reduces server load by running in the browser.
✅ Works across all browsers and platforms.
✅ Allows client-side and server-side development.
✅ Has a huge ecosystem (React, Angular, Node.js).

Variables in JavaScript

Variables store values and allow us to reuse them in code. JavaScript has three ways to declare variables:

1) var (Old way - Avoid using)

- Can be re-declared and updated.
- Function-scoped (limited to the function it is declared in).
- May cause issues in larger programs.

Example:

```
var name = "John";
console.log(name); // Output: John
```

2) let (Recommended for changeable values)

- Can be updated but not re-declared within the same scope.
- Block-scoped (only exists within {} it is declared in).

Example:
let age = 25;
age = 30;// Allowed (updating value)
console.log(age); // Output: 30

3) const (Recommended for constant values)

- Cannot be updated or re-declared.
- Block-scoped.
- Must be assigned a value when declared.

Example:
const pi = 3.14;
console.log(pi); // Output: 3.14
// pi = 3.14159; // This will cause an error

Data Types in JavaScript

JavaScript has different data types:

1. String – Text values ("Hello", 'JavaScript')
2. Number – Numeric values (10, 3.14)
3. Boolean – True/false values (true, false)
4. Array – Collection of values (["Apple", "Banana", "Cherry"])
5. Object – Key-value pairs ({name: "John", age: 25})

Example:
let name = "Alice"; // String
let age = 28; // Number
let isStudent = true; // Boolean
let fruits = ["Apple", "Banana"]; // Array
let person = {name: "John", age: 30}; // Object

Functions in JavaScript

Functions allow us to write reusable code.

1) Regular Function

A block of code that executes when called.

Example:
```
function greet() {
    console.log("Hello, World!");
}
greet(); // Output: Hello, World!
```

2) Function with Parameters
Functions can take inputs (parameters) and return outputs.

Example:
```
function add(a, b) {
    return a + b;
}
console.log(add(5, 10)); // Output: 15
```

3) Arrow Functions (ES6 Feature)
A shorter way to write functions.

Example:
```
const multiply = (x, y) => x * y;
console.log(multiply(4, 5)); // Output: 20
```

Conditionals in JavaScript

Used to make decisions based on conditions.

1) if Statement
Executes a block of code only if a condition is true.

Example:
```
let num = 10;
if (num > 5) {
    console.log("Number is greater than 5");
}
```

2) if-else Statement
Executes one block if true, another if false.

Example:
```
let age = 18;
if (age >= 18) {
    console.log("You can vote.");
} else {
    console.log("You cannot vote.");
}
```

3) if-else if-else Statement
Checks multiple conditions.

Example:
```
let marks = 85;
if (marks >= 90) {
    console.log("Grade: A");
} else if (marks >= 80) {
    console.log("Grade: B");
} else {
    console.log("Grade: C");
}
```

Loops in JavaScript
Loops allow repeating tasks multiple times.

1) for Loop
Runs a block of code a specific number of times.

Example:
```
for (let i = 1; i <= 5; i++) {
    console.log("Iteration:", i);
}
```

2) while Loop
Runs as long as a condition is true.

Example:
```
let i = 1;
while (i <= 5) {
    console.log("While Loop:", i);
    i++;
}
```

3) do...while Loop
Runs at least once, even if the condition is false.

Example:
```
let j = 1;
do {
    console.log("Do While:", j);
    j++;
} while (j <= 5);
```

Arrays in JavaScript
Arrays store multiple values in one variable.

Example:
```
let fruits = ["Apple", "Banana", "Cherry"];
console.log(fruits[0]); // Output: Apple
```

1) Array Methods
- push() – Adds an item to the end
- pop() – Removes last item
- length – Returns the number of elements

Example:
```
fruits.push("Mango"); // Adds Mango
console.log(fruits.length); // Output: 4
```

Objects in JavaScript
Objects store key-value pairs.

Example:
```
let person = {
    name: "John",
    age: 30,
    city: "New York"
};
console.log(person.name); // Output: John
```

Events in JavaScript

Events allow user interactions like clicks and keypresses.

Example:

```
document.getElementById("btn").addEventListener("click", function() {
    alert("Button Clicked!");
});
```

DOM Manipulation

JavaScript can modify HTML and CSS dynamically.

Example:

```
document.getElementById("demo").innerHTML = "Hello, JavaScript!";
```

WORDPRESS

Introduction to WordPress

WordPress is a content management system (CMS) that allows users to create, manage, and customize websites without coding. It is widely used for blogs, business websites, eCommerce stores, and more.

Key Features:
✅ Easy to use (No coding required)
✅ Customizable themes and plugins
✅ SEO-friendly
✅ Supports blogs, business sites, and eCommerce
✅ Open-source and free (self-hosted version)

History of WordPress

- **2003** – Created by Matt Mullenweg and Mike Little as a blogging platform.
- **2005** – Introduction of WordPress themes and plugins.
- **2010** – Custom post types introduced, making WordPress more flexible.
- **2013** – WordPress powers 20% of the web.
- **2022+** – WordPress powers over 40% of all websites worldwide.

Difference Between WordPress.com & WordPress.org

Feature	WordPress.com (Hosted)	WordPress.org (Self-hosted)
Hosting	Hosted by WordPress (limited control)	Self-hosted (full control)
Customization	Limited themes & plugins	Full access to themes & plugins
Monetization	Limited (Ads by WordPress)	Full monetization freedom
Cost	Free (with paid plans)	Hosting and domain costs apply
Best For	Beginners who want an easy setup	Advanced users who want full control

Use WordPress.org if you want full control, customization, and monetization options.
Use WordPress.com for a hassle-free experience but with limited flexibility.

How to Use WordPress?

Step 1: Install WordPress

- For WordPress.com → Sign up at wordpress.com
- For WordPress.org → Download from wordpress.org and install on a web host

Step 2: Choose a Theme

- Go to Appearance > Themes
- Install and activate a theme

Step 3: Install Plugins

- Go to Plugins > Add New
- Install plugins like Yoast SEO, Elementor, WooCommerce

Step 4: Create Pages & Posts

- Pages → Used for static content (Home, About, Contact)
- Posts → Used for blog entries (News, Updates)

Step 5: Customize Website

- Use Appearance > Customize to change fonts, colors, layouts
- Add widgets and menus

Step 6: Publish Your Website

- Once designed, click Publish and make it live!

WIX

Introduction to Wix

Wix is a website builder that allows users to create websites easily using a drag-and-drop interface. It requires no coding skills and is ideal for small businesses, personal websites, and online stores.

Key Features:
✅ Drag-and-drop editor (No coding needed)
✅ Customizable templates for different industries
✅ SEO-friendly tools
✅ E-commerce support (for online stores)
✅ Free & paid plans available

History of Wix

- **2006** – Founded by Avishai Abrahami, Nadav Abrahami, and Giora Kaplan.
- **2013** – Introduced the Wix App Market for additional features.
- **2016** – Launched Artificial Design Intelligence (ADI) for automatic website creation.
- **2020+** – Wix powers millions of websites worldwide.

How to Use Wix?

Step 1: Sign Up
- Go to wix.com and create an account.

Step 2: Choose a Website Type
- Select Business, Portfolio, Blog, eCommerce, etc.

Step 3: Select a Method to Build Your Website
- Wix ADI (Artificial Design Intelligence) – Automatically creates a website based on your answers.
- Wix Editor – Full control with drag-and-drop customization.

Step 4: Customize Your Website
- Change fonts, colors, layouts, and images.
- Add sections like About, Services, and Contact.

Step 5: Add Features Using Apps
- Install apps like Wix Stores (for eCommerce), Wix Bookings (for appointments), and Wix SEO.

Step 6: Connect a Domain

- Use a free Wix domain (example: yourname.wixsite.com).
- Upgrade to paid plans for a custom domain (yourname.com).

Step 7: Publish Your Website

- Click Publish and go live!