HTML

1. Introduction to HTML

HTML (HyperText Markup Language) is the standard markup language used to create web pages. It defines the **structure** of a webpage using **elements** (tags).

Key Features of HTML:

- **Not a programming language** (no logic, just structure).
- **Uses tags** (`<tag>`) to define elements.
- **Case-insensitive**, but lowercase is recommended.
- **Rendered by web browsers** (Chrome, Firefox, Edge).

2. Basic HTML Document Structure

Every HTML document follows a standard structure:

```
<!DOCTYPE html> <!-- Tells the browser this is HTML5 -->
<html lang="en"> <!-- Root element, wraps all content -->
<head> <!-- Contains meta-information -->
   <meta charset="UTF-8"> <!-- Defines character encoding -->
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>My First Webpage</title> <!-- Appears in browser tab -->
</head>
<body> <!-- Contains visible content -->
   <h1>Hello, World!</h1>
   <p>This is my first webpage.</p>
</body>
</html>
```

3. HTML Elements and Tags

An **HTML element** consists of:

- **Opening tag** (`<p>`)
- **Content** (text, images, etc.)

- **Closing tag** (`</p>`)

Example:

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

- `<p>` → Opening tag
- `This is a paragraph.` → Content
- `</p>` → Closing tag

Self-Closing Tags (Void Elements)

Some elements don't need closing tags:

- `<img>`
- `<br>` (line break)
- `<hr>` (horizontal rule)

4. Common HTML Tags with Examples

1. Headings (`<h1>` to `<h6>`)

- Define section headings (h1 = most important, h6 = least).

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

2. Paragraph (`<p>`)

- Defines a block of text.

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

3. Links (`<a>`)

- Creates hyperlinks using `href` attribute.

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

4. Images (<img>)

- Embeds an image (requires src and alt attributes).

```
<img src="cat.jpg" alt="A cute cat">
```

5. Lists

- **Unordered List (<ul>)**: Bullet points.
- **Ordered List (<ol>)**: Numbered list.
- **List Items (<li>)**: Each item in the list.

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

<ol>
  <li>First step</li>
  <li>Second step</li>
</ol>
```

6. Divisions ( ) & Spans ()

- → Block-level container (for layout).
- → Inline container (for styling text).

```
<div style="background: lightblue;">
  <p>This is inside a div.</p>
</div>

<p>This is <span style="color: red;">red text</span>.</p>
```

7. Line Break (
) & Horizontal Rule (<hr>)

-
 → Forces a line break.
- <hr> → Adds a horizontal line.

```
<p>First line.<br>Second line.</p>
<hr>
<p>Section after a line.</p>
```

5. HTML Attributes

Attributes provide **additional information** about an element.

Attribute	Description	Example
id	Unique identifier	
class	Groups elements for styling	<p class="highlight">
src	Source (for images, scripts)	<img src="logo.png">
href	Hyperlink reference	<a href="contact.html">
alt	Alternate text for images	<img src="dog.jpg" alt="A dog">
style	Inline CSS	<p style="color: red;">

Example:

```
<p id="intro" class="text-blue" style="font-size: 18px;">
   This is a styled paragraph.
</p>
```

6. Semantic HTML (Meaningful Tags)

Semantic HTML improves **accessibility** and **SEO**.

Tag	Purpose
<header>	Top section (logo, navigation)
<nav>	Navigation links
<main>	Main content of the page

Tag	Purpose
<section>	Defines a section
<article>	Independent content (blog post)
<footer>	Bottom section (copyright, links)

Example:

```
<body>
  <header>
    <h1>Website Title</h1>
    <nav>
      <a href="/home">Home</a>
      <a href="/about">About</a>
    </nav>
  </header>
  <main>
    <article>
      <h2>Blog Post</h2>
      <p>Content goes here...</p>
    </article>
  </main>
  <footer>
    <p>&copy; 2024 My Website</p>
  </footer>
</body>
```

7. HTML Forms (User Input)

Used for **login pages, surveys, search bars**.

Basic Form Structure:

```
<form action="/submit" method="POST">
  <label for="username">Username:</label>
  <input type="text" id="username" name="username" required>
```

```
    <label for="password">Password:</label>
    <input type="password" id="password" name="password" required>

    <input type="submit" value="Login">
</form>
```

Common Form Elements:

- `<input>` → Text, password, email, checkbox, radio, etc.
- `<textarea>` → Multi-line text input.
- `<select>` → Dropdown menu.
- `<button>` → Clickable button.

8. HTML Tables

Used to display **tabular data**.

html

```
<table border="1">
    <tr> <!-- Table Row -->
        <th>Name</th> <!-- Table Header -->
        <th>Age</th>
    </tr>
    <tr>
        <td>Alice</td> <!-- Table Data -->
        <td>25</td>
    </tr>
</table>
```

9. HTML Comments

Comments are ignored by the browser (for developers).

```
<!-- This is a comment -->
<p>Visible content</p>
```

10. HTML Best Practices

Use **semantic HTML** (`<header>`, `<nav>`, etc.).

Always include `alt` **text** for images.

Use **lowercase** for tags and attributes.
Validate HTML using W3C Validator.

1. Introduction to CSS3

CSS3 (Cascading Style Sheets Level 3) is the latest version of CSS, introducing new features for styling web pages, including animations, transitions, gradients, and responsive design enhancements.

2. CSS3 Selectors

CSS3 introduces advanced selectors for more precise targeting of HTML elements.

2.1 Attribute Selectors

- Select elements based on attributes.

```
/* Selects all <a> tags with a "target" attribute */
a[target] {
    color: red;
}

/* Selects <input> elements with type="text" */
input[type="text"] {
    border: 1px solid blue;
}
```

2.2 Pseudo-classes

- `:nth-child(n)` – Selects the nth child element.

```
li:nth-child(2) {
    color: green;
}
```

- `:hover` – Applies style when mouse hovers.

```
button:hover {
    background-color: yellow;
}
```

2.3 Pseudo-elements

- ::before & ::after – Insert content before/after an element.

```
p::before {
    content: "Note: ";
    font-weight: bold;
}
```

3. Box Model Enhancements

3.1 box-sizing

- Controls how width/height are calculated.

```
* {
    box-sizing: border-box; /* Includes padding & border in width */
}
```

3.2 box-shadow

- Adds shadow effects.

```
iv {
    box-shadow: 5px 5px 10px rgba(0, 0, 0, 0.5);
}
```

4. Backgrounds & Gradients

4.1 Multiple Backgrounds

- Apply multiple background images.

```
div {
    background-image: url("img1.png"), url("img2.png");
    background-position: left top, right bottom;
}
```

4.2 Gradients

- Linear Gradient:

```
body {
    background: linear-gradient(to right, red, yellow);
}
```

- Radial Gradient:

```
div {
  background: radial-gradient(circle, red, yellow, green);
}
```

5. Transitions & Animations

5.1 Transitions

- Smoothly change property values over time.

```
button {
  transition: background-color 0.5s ease;
}
button:hover {
  background-color: blue;
}
```

5.2 Animations

- Define keyframe animations.

```
@keyframes slide {
  from { transform: translateX(0); }
  to { transform: translateX(100px); }
}
div {
  animation: slide 2s infinite;
}
```

6. Transformations

- Modify element shape, size, and position.

```
div {
  transform: rotate(45deg) scale(1.2);
}
```

7. Flexbox (Flexible Box Layout)

- A modern layout model for responsive designs.

```
.container {
  display: flex;
```

```
    justify-content: center;
    align-items: center;
}
```

8. Grid Layout

- Two-dimensional grid-based layout system.

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

9. Responsive Design (Media Queries)

- Adjust styles based on screen size.

```
@media (max-width: 600px) {
    body {
        font-size: 14px;
    }
}
```

10. Custom Properties (CSS Variables)

- Define reusable variables.

```
:root {
    --main-color: #3498db;
}
button {
    background-color: var(--main-color);
}
```

11. Example: Complete CSS3 Styled Card

```
<!DOCTYPE html>
<html>
<head>
    <style>
        .card {
            width: 300px;
            padding: 20px;
```

```
            border-radius: 10px;
            box-shadow: 0 4px 8px rgba(0,0,0,0.2);
            background: linear-gradient(135deg, #f5f7fa, #c3cfe2);
            transition: transform 0.3s;
        }
        .card:hover {
            transform: scale(1.05);
        }
        .card h2 {
            color: #333;
        }
        .card p {
            color: #666;
        }
    </style>
</head>
<body>
    <div class="card">
        <h2>CSS3 Card</h2>
        <p>Hover over me to see an effect!</p>
    </div>
</body>
</html>
```

12. Browser Support & Prefixes

Some CSS3 features require vendor prefixes:

```
button {
    -webkit-transition: all 0.3s; /* Chrome/Safari */
    -moz-transition: all 0.3s;    /* Firefox */
    -o-transition: all 0.3s;      /* Opera */
    transition: all 0.3s;         /* Standard */
}
```

1. Introduction to JavaScript

JavaScript is a high-level, interpreted programming language used to make web pages interactive. It is a core technology of the web alongside HTML and CSS.

Example: Hello World in JavaScript

```
console.log("Hello, World!");
```

2. Variables and Data Types

Variables store data values. JavaScript is dynamically typed (no need to declare variable types).

Variable Declaration

- `var` (old, function-scoped)
- `let` (block-scoped, can be reassigned)
- `const` (block-scoped, cannot be reassigned)

Data Types

1. **Primitive Types**:
   - `String` (`"Hello"`, `'World'`)
   - `Number` (`10`, `3.14`)
   - `Boolean` (`true`, `false`)
   - `Null` (`null`)
   - `Undefined` (`undefined`)
   - `Symbol` (`Symbol('id')`)
   - `BigInt` (`1234567890123456789n`)
2. **Non-Primitive (Reference) Types**:
   - `Object` (`{ name: "John" }`)
   - `Array` (`[1, 2, 3]`)
   - `Function` (`function() {}`)

Examples

```
let name = "Alice"; // String
const age = 25; // Number
let isStudent = true; // Boolean
let car = null; // Null
```

```
let job; // Undefined
const id = Symbol("id"); // Symbol
const bigNum = 100n; // BigInt

const person = { name: "Bob", age: 30 }; // Object
const numbers = [1, 2, 3]; // Array
function greet() { console.log("Hi!"); } // Function
```

3. Operators

Arithmetic Operators

```
let x = 10, y = 5;
console.log(x + y); // 15 (Addition)
console.log(x - y); // 5 (Subtraction)
console.log(x * y); // 50 (Multiplication)
console.log(x / y); // 2 (Division)
console.log(x % y); // 0 (Modulus)
console.log(x ** y); // 100000 (Exponentiation)
```

Comparison Operators

```
console.log(10 == "10"); // true (Loose equality)
console.log(10 === "10"); // false (Strict equality)
console.log(10 != "10"); // false
console.log(10 !== "10"); // true
console.log(10 > 5); // true
console.log(10 <= 5); // false
```

Logical Operators

```
console.log(true && false); // false (AND)
console.log(true || false); // true (OR)
console.log(!true); // false (NOT)
```

4. Control Flow

Conditional Statements

```
let age = 18;

if (age >= 18) {
  console.log("Adult");
```

```
} else if (age >= 13) {
  console.log("Teen");
} else {
  console.log("Child");
}

// Ternary Operator
let canVote = age >= 18 ? "Yes" : "No";
console.log(canVote); // "Yes"
```

Loops

javascript

```
// For Loop
for (let i = 0; i < 5; i++) {
  console.log(i); // 0, 1, 2, 3, 4
}

// While Loop
let j = 0;
while (j < 3) {
  console.log(j); // 0, 1, 2
  j++;
}

// Do-While Loop
let k = 0;
do {
  console.log(k); // 0, 1, 2
  k++;
} while (k < 3);
```

5. Functions

Functions are reusable blocks of code.

Function Declaration

```
function greet(name) {
  return `Hello, ${name}!`;
}
console.log(greet("Alice")); // "Hello, Alice!"
```

Function Expression

```
const greet = function(name) {
  return `Hello, ${name}!`;
};
console.log(greet("Bob")); // "Hello, Bob!"
```

Arrow Function (ES6)

```
const greet = (name) => `Hello, ${name}!`;
console.log(greet("Charlie")); // "Hello, Charlie!"
```

6. Arrays

Arrays store multiple values in a single variable.

Array Methods

```
const fruits = ["Apple", "Banana"];

fruits.push("Orange"); // Adds to end
fruits.pop(); // Removes from end
fruits.unshift("Mango"); // Adds to start
fruits.shift(); // Removes from start

console.log(fruits.includes("Apple")); // true
console.log(fruits.indexOf("Banana")); // 1

const numbers = [1, 2, 3];
const doubled = numbers.map(num => num * 2); // [2, 4, 6]
const filtered = numbers.filter(num => num > 1); // [2, 3]
const sum = numbers.reduce((acc, num) => acc + num, 0); // 6
```

7. Objects

Objects store key-value pairs.

Object Example

```
const person = {
  name: "John",
  age: 30,
  greet: function() {
```

```
    console.log(`Hi, I'm ${this.name}`);
  }
};

console.log(person.name); // "John"
person.greet(); // "Hi, I'm John"

// Adding a new property
person.job = "Developer";
console.log(person.job); // "Developer"
```

8. Classes (OOP)

Classes are blueprints for creating objects.

Class Example

```
class Person {
  constructor(name, age) {
    this.name = name;
    this.age = age;
  }

  greet() {
    console.log(`Hello, ${this.name}`);
  }
}

const alice = new Person("Alice", 25);
alice.greet(); // "Hello, Alice"
```

9. Asynchronous JavaScript

Callbacks (Old Way)

```
function fetchData(callback) {
  setTimeout(() => {
    callback("Data received");
  }, 1000);
}

fetchData((data) => {
```

```
  console.log(data); // "Data received" after 1 sec
});
```

Promises (Better Way)

```
function fetchData() {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      resolve("Data received");
    }, 1000);
  });
}

fetchData()
  .then(data => console.log(data)) // "Data received"
  .catch(err => console.log(err));
```

Async/Await (Modern Way)

```
async function getData() {
  try {
    const data = await fetchData();
    console.log(data); // "Data received"
  } catch (err) {
    console.log(err);
  }
}
getData();
```

10. Error Handling

```
try {
  let x = y + 10; // y is not defined
} catch (err) {
  console.log("Error:", err.message); // "y is not defined"
} finally {
  console.log("Execution completed");
}
```

11. DOM Manipulation

JavaScript can change HTML content, styles, and attributes.

Example: Changing HTML Content

```
<button id="btn">Click Me</button>
<p id="text">Hello</p>
```

javascript

```
const btn = document.getElementById("btn");
const text = document.getElementById("text");

btn.addEventListener("click", () => {
  text.textContent = "Button Clicked!";
});
```

12. LocalStorage & SessionStorage

Store data in the browser.

Example: Using localStorage

```
localStorage.setItem("name", "Alice");
console.log(localStorage.getItem("name")); // "Alice"
localStorage.removeItem("name");
```