# 

**Name:** Lakshmisai.T

# **CSS Documentation**

## **1. Introduction to CSS**

CSS (Cascading Style Sheets) is a stylesheet language used to describe the presentation of HTML documents. CSS controls how HTML elements are displayed on screen, paper, or in other media.

**Benefits of CSS:**

* Separates content from design
* Reduces code duplication
* Enhances reusability and maintainability

**Types of CSS:**

* Inline CSS
* Internal CSS
* External CSS

<!-- Inline CSS -->

<p style="color: blue;">Hello World</p>

<!-- Internal CSS -->

<style>

p { color: blue; }

</style>

<!-- External CSS -->

<link rel="stylesheet" href="style.css">

## **2. CSS Syntax**

CSS is written using selectors and declaration blocks.

selector {

property: value;

property2: value2;

}

Example:

h1 {

color: red;

font-size: 24px;

}

## **3. CSS Selectors**

Selectors are used to select HTML elements for styling.

**Common Selectors:**

* Type: p, h1, div
* Class: .classname
* ID: #idname
* Grouping: h1, p
* Universal: \*

**Advanced Selectors:**

* Descendant: div p
* Child: div > p
* Adjacent sibling: h1 + p
* Attribute: input[type="text"]

## **4. Colors and Backgrounds**

### **Colors**

CSS supports color names, hex, RGB, RGBA, HSL, and HSLA formats.

p { color: blue; }

div { color: #00ff00; }

h1 { color: rgb(255, 0, 0); }

### **Backgrounds**

div {

background-color: lightgray;

background-image: url('bg.jpg');

background-repeat: no-repeat;

background-position: center;

background-size: cover;

}

## **5. Box Model**

Each element is a box consisting of:

* Content
* Padding
* Border
* Margin

div {

width: 200px;

padding: 10px;

border: 1px solid black;

margin: 15px;

}

## **6. Text and Fonts**

p {

font-family: Arial, sans-serif;

font-size: 16px;

font-weight: bold;

color: darkblue;

text-align: justify;

text-decoration: underline;

}

**Additional Properties:**

* text-transform, letter-spacing, line-height, word-spacing

## **7. Display and Positioning**

### **Display**

div { display: block; }

span { display: inline; }

.container { display: flex; }

### **Position**

.box {

position: relative;

top: 10px;

left: 20px;

}

Values: static, relative, absolute, fixed, sticky

## **8. Flexbox**

.container {

display: flex;

flex-direction: row;

justify-content: space-between;

align-items: center;

}

.item {

flex: 1;

}

## **9. Grid Layout**

.grid {

display: grid;

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

grid-gap: 10px;

}

.item {

grid-column: 1 / 3;

grid-row: 1 / 2;

}

## **10. Transitions and Animations**

### **Transitions**

div {

transition: background-color 0.3s ease-in-out;

}

### **Animations**

@keyframes slidein {

from { margin-left: 100%; }

to { margin-left: 0%; }

}

div {

animation: slidein 2s;

}

## **11. Responsive Design with Media Queries**

@media (max-width: 600px) {

.container {

flex-direction: column;

}

}

## **12. Pseudo-Classes and Pseudo-Elements**

### **Pseudo-Classes**

a:hover { color: red; }

p:first-child { font-weight: bold; }

### **Pseudo-Elements**

p::first-line { color: blue; }

div::before { content: 'Note: '; }

## **13. Forms and Inputs Styling**

input[type="text"] {

border: 1px solid #ccc;

padding: 10px;

}

input:focus {

border-color: blue;

}

## **14. CSS Variables**

:root {

--main-color: coral;

}

div {

color: var(--main-color);

}

## **15. Conclusion**

CSS is essential for building modern, responsive, and visually engaging websites. Mastery of selectors, layout techniques like Flexbox and Grid, and advanced topics like transitions and media queries will significantly improve your frontend development skills.

# **HTML Documentation**

## **Table of Contents**

1. Introduction to HTML
2. Basic Structure of an HTML Document
3. HTML Elements and Tags
4. Headings and Paragraphs
5. Formatting Text
6. Comments in HTML
7. Hyperlinks
8. Images
9. Lists
10. Tables
11. Forms
12. Semantic Elements
13. Multimedia (Audio and Video)
14. HTML APIs
15. Best Practices

## **1. Introduction to HTML**

HTML (HyperText Markup Language) is the standard language for creating webpages. It structures the content using elements represented by tags.

* **HyperText** refers to links that connect web pages.
* **Markup** refers to tags used to describe the structure of the page.

## **2. Basic Structure of an HTML Document**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>My First Page</title>

</head>

<body>

<h1>Hello, World!</h1>

</body>

</html>

* <!DOCTYPE html>: Defines the document as HTML5
* <html>: Root element
* <head>: Metadata (title, links, meta info)
* <body>: Visible page content

## **3. HTML Elements and Tags**

Elements usually have an opening and closing tag:

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

Some elements are self-closing:

<img src="image.jpg" alt="An image" />

## **4. Headings and Paragraphs**

HTML has six heading levels:

<h1>Main Heading</h1>

<h2>Subheading</h2>

...

<h6>Smallest Heading</h6>

Paragraphs are added using:

<p>This is a paragraph of text.</p>

## **5. Formatting Text**

<b>Bold</b>, <strong>Strong</strong>

<i>Italic</i>, <em>Emphasis</em>

<u>Underline</u>, <mark>Highlight</mark>

<del>Deleted</del>, <ins>Inserted</ins>

<sub>Subscript</sub>, <sup>Superscript</sup>

## **6. Comments in HTML**

Comments are not displayed:

<!-- This is a comment -->

## **7. Hyperlinks**

<a href="https://example.com" target="\_blank">Visit Example</a>

* href: Destination URL
* target="\_blank": Opens link in a new tab

## **8. Images**

<img src="image.jpg" alt="Description" width="300" height="200" />

* alt: Text for screen readers or if image fails to load
* width and height: Dimensions

## **9. Lists**

**Unordered List:**

<ul>

<li>Item 1</li>

<li>Item 2</li>

</ul>

**Ordered List:**

<ol>

<li>First</li>

<li>Second</li>

</ol>

**Description List:**

<dl>

<dt>HTML</dt>

<dd>Markup language for web pages</dd>

</dl>

## **10. Tables**

<table>

<tr>

<th>Header</th>

<th>Header</th>

</tr>

<tr>

<td>Data 1</td>

<td>Data 2</td>

</tr>

</table>

## **11. Forms**

<form action="/submit" method="post">

<label for="name">Name:</label>

<input type="text" id="name" name="name" required>

<button type="submit">Submit</button>

</form>

* action: Target URL
* method: GET or POST

## **12. Semantic Elements**

<header></header>

<nav></nav>

<main></main>

<article></article>

<section></section>

<footer></footer>

These improve accessibility and SEO.

## **13. Multimedia (Audio and Video)**

<video controls width="400">

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

</video>

<audio controls>

<source src="audio.mp3" type="audio/mpeg">

</audio>

## **14. HTML APIs**

HTML integrates with JavaScript APIs:

* Geolocation API
* Drag and Drop API
* Web Storage API (localStorage & sessionStorage)

Example:

<script>

localStorage.setItem('name', 'Lakshmisai.T');

</script>

# **JavaScript Documentation**

## **1. Introduction to JavaScript**

JavaScript is a high-level, interpreted programming language used to create dynamic and interactive content on web pages. It runs in the browser and enhances user interaction.

### **Features:**

* Lightweight and interpreted
* Object-oriented
* Event-driven
* Versatile (client-side and server-side)

<script>

alert("Hello, JavaScript!");

</script>

## **2. Variables and Data Types**

### **Declaring Variables:**

var x = 10; // function scoped

let y = 20; // block scoped

const z = 30; // constant block scoped

### **Data Types:**

* Number: let num = 42;
* String: let str = "Hello";
* Boolean: let isTrue = true;
* Null: let a = null;
* Undefined: let b;
* Object: let obj = { name: "John" };
* Array: let arr = [1, 2, 3];

## **3. Operators**

### **Arithmetic Operators:**

+ - \* / % \*\* ++ --

### **Comparison Operators:**

== === != !== > < >= <=

### **Logical Operators:**

&& || !

### **Assignment Operators:**

= += -= \*= /=

### **Ternary Operator:**

let result = age >= 18 ? "Adult" : "Minor";

## **4. Control Structures**

### **if-else:**

if (condition) {

// code

} else {

// code

}

### **switch:**

switch(value) {

case 1:

// code

break;

default:

// code

}

### **Loops:**

for (let i = 0; i < 5; i++) {

console.log(i);

}

while (condition) {

// code

}

do {

// code

} while (condition);

## **5. Functions**

### **Function Declaration:**

function greet(name) {

return `Hello ${name}`;

}

### **Function Expression:**

const greet = function(name) {

return `Hello ${name}`;

}

### **Arrow Functions:**

const greet = (name) => `Hello ${name}`;

## **6. Objects and Arrays**

### **Objects:**

let person = {

name: "Alice",

age: 25,

greet() {

return `Hello, ${this.name}`;

}

};

### **Arrays:**

let numbers = [1, 2, 3];

console.log(numbers[0]); // 1

### **Array Methods:**

* push(), pop(), shift(), unshift()
* map(), filter(), reduce(), forEach()

## **7. DOM Manipulation**

### **Accessing Elements:**

document.getElementById('id');

document.querySelector('.class');

### **Modifying Elements:**

document.getElementById('id').textContent = "New text";

### **Creating Elements:**

const div = document.createElement('div');

document.body.appendChild(div);

## **8. Events**

### **Adding Event Listeners:**

document.getElementById('btn').addEventListener('click', function() {

alert("Button clicked");

});

## **9. Callback Functions**

function greetUser(callback) {

console.log("Hello");

callback();

}

greetUser(function() {

console.log("User");

});

## **10. Promises**

const fetchData = () => {

return new Promise((resolve, reject) => {

setTimeout(() => resolve("Data received"), 2000);

});

};

fetchData().then(data => console.log(data));

## **11. Async/Await**

async function getData() {

const data = await fetchData();

console.log(data);

}

getData();

## **12. ES6+ Features**

* Template literals: `Hello ${name}`
* Destructuring: const {name} = person;
* Spread/Rest: ...array
* Modules: import/export

## **13. Error Handling**

try {

throw new Error("Something went wrong");

} catch (e) {

console.error(e);

} finally {

console.log("Cleanup");

}

## **14. Classes**

class Animal {

constructor(name) {

this.name = name;

}

speak() {

return `${this.name} makes a noise.`;

}

}

const dog = new Animal("Dog");

dog.speak();