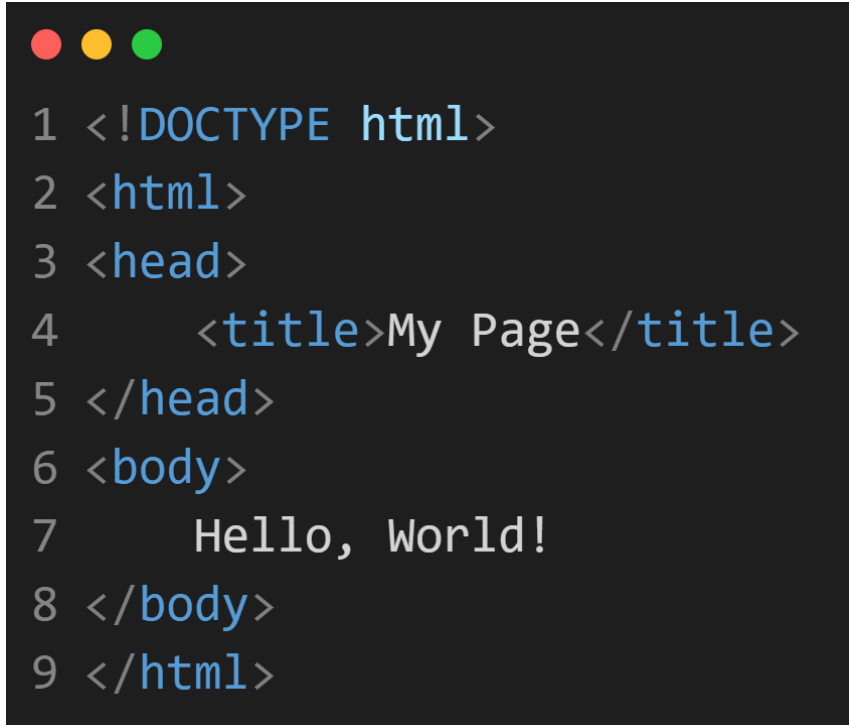


# 1 What is HTML?

## 1.1 Example

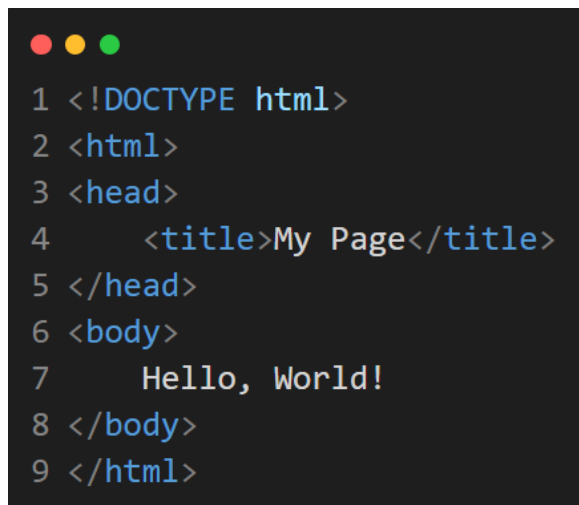


```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <title>My Page</title>
5 </head>
6 <body>
7     Hello, World!
8 </body>
9 </html>
```

This is a basic HTML structure. The `<!DOCTYPE html>` declares the document type and version, and the content inside `<body>` is what's displayed on the webpage.

## 2 Basic HTML Structure

### 2.1 Example



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <title>My Page</title>
5 </head>
6 <body>
7     Hello, World!
8 </body>
9 </html>
```

Every HTML document starts with a `DOCTYPE` declaration followed by the root `<html>` element. Inside, we have a `<head>` for meta information and a `<body>` for visible content.

## 3 HTML Tags, and Elements

The terms "tag" and "element" in the context of HTML are often used interchangeably, but they have distinct meanings.

### 3.1 Tag

- A tag is a markup construct that begins with < and ends with >.
- There are opening tags and closing tags. For example, <p> is the opening tag for a paragraph, and </p> is the closing tag.
- Some tags are self-closing, meaning they don't have a separate closing tag. For example, the line break tag is <br/> (or simply <br> in HTML5).

### 3.2 Element

- An element encompasses the opening tag, the content, and the closing tag.
- For instance, <p>Hello, world!</p> is a paragraph element. Here, <p> is the opening tag, `Hello, world!` is the content, and </p> is the closing tag.
- Elements can also have attributes that provide additional information about the element. For example, in the anchor element

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

```
href="https://www.example.com/"
```

is an attribute.

- Some elements, like the <img> or <br>, don't have content or a closing tag. They are still considered elements, but they are also self-closing.

In summary:

- Tags are the actual markup, the <...> constructs.
- Elements refer to the entire unit, from the start tag to the end tag (or just the self-closing tag) and everything in between.

### 3.3 Example

```
1 <h1>This is a heading</h1>
2 <p>This is a paragraph.</p>
```

HTML elements define the structure and content. Here, <h1> is a top-level heading, and <p> is a paragraph.

## 4 Headings in HTML

### 4.1 Example

```
1 <h1>Main Heading</h1>
2 <h2>Subheading</h2>
```

Headings range from ``<h1>`` (most important) to ``<h6>`` (least important). They structure content and improve accessibility.

## 5 Paragraphs in HTML

### 5.1 Example

```
1 <p>This is a sample paragraph of text.</p>
```

The ``<p>`` element defines a paragraph. It's a block-level element, meaning it starts on a new line and takes up the full width.

## 6 Links in HTML

### 6.1 Example

```
1 <a href="https://www.example.com/">Visit Example</a>
```

The ``<a>`` element defines a hyperlink. The ``href` attribute specifies the link's destination.`

## 7 Lists in HTML

### 7.1 Example

```
1 <ul>
2   <li>Item 1</li>
3   <li>Item 2</li>
4 </ul>
```

Here, we have an unordered list (``<ul>``) with list items (``<li>``). Each ``<li>`` represents a single item in the list.

## 8 Images in HTML

### 8.1 Example

```
1 
```

The `<img>` element embeds an image. The `src` attribute points to the image location, and the `alt` provides a text description for accessibility.

## 9 Forms in HTML

### 9.1 Example

```
1 <form action="/submit" method="post">
2   <input type="text" name="name" placeholder="Enter your name">
3   <input type="submit" value="Submit">
4 </form>
```

The `<form>` element collects user input. Here, we have a text input field and a submit button. The form's data will be sent to `/submit` when submitted.

## 10 What is CSS?

**CSS** stands for **Cascading Style Sheets**. It is a stylesheet language used to describe the look and formatting of a document written in HTML or XML. While HTML provides the structure of a web page (like defining headings, paragraphs, and links), CSS defines how these elements should appear on the screen.

### 10.1 Key Points About CSS

#### 10.1.1 Presentation

CSS is all about presentation. It controls the layout of multiple web pages all at once, the colours, the fonts, spacing between elements, and even animations or transitions.

#### 10.1.2 Separation of Content and Design

By using CSS, web developers can separate the content of a website (provided by HTML) from the design. This separation makes it easier to maintain websites, share styles across multiple pages, and tailor pages to different environments.

### 10.1.3 Cascading

The term "cascading" refers to the order of priority a browser should follow when it encounters conflicting styling rules. This means that some styles can "cascade" or fall from one style to another, allowing for a hierarchy of style rules.

### 10.1.4 Selectors and Properties

In CSS, styles are applied to HTML elements through selectors. Once an element is selected, one or more styles can be applied using properties. For example:

```
p {  
  color: red;  
  font-size: 16px;  
}
```

In this example, the selector is `p` (which targets all paragraph elements), and two properties are set: `color` and `font-size`.

### 10.1.5 External, Internal, and Inline Styles

CSS can be applied in three ways:

- External: By linking to an external `.css` file.
- Internal: By embedding styles in the `<head>` section of the HTML document using a `<style>` tag.
- Inline: By applying styles directly to individual HTML elements using the `style` attribute.

### 10.1.6 Responsive Design

With the rise of mobile devices, CSS has evolved to allow for the creation of responsive designs. This means that web pages can adjust and look well on screens of all sizes, from desktop monitors to smartphones.

### 10.1.7 Frameworks

There are many CSS frameworks, like Bootstrap, Foundation, and Tailwind CSS, that provide pre-written CSS to help speed up the development process and ensure consistent, responsive designs.

## 10.2 Example

Given a CSS file named `styles.css`, you can link it to your HTML document like this:

```
1 <head>  
2   <link rel="stylesheet" type="text/css" href="styles.css">  
3 </head>
```

This method of using an external stylesheet is beneficial because:

1. Separation of Concerns: It keeps the structure (HTML) separate from the presentation (CSS), making your code more organised and maintainable.
2. Reusability: The same `.css` file can be linked to multiple HTML pages, ensuring a consistent look and feel across different pages of a website.
3. Caching: Browsers can cache external CSS files, which can improve page load times for returning visitors.

While external stylesheets are common, remember that CSS can also be written directly in the HTML document using `<style>` tags (internal CSS) or inline within HTML elements using the `style` attribute (inline CSS).

However, for larger projects and better maintainability, external stylesheets are recommended.

### 10.3 Example

This CSS rule sets the text colour of all `<p>` elements to blue.

```
1 p {  
2     color: blue;  
3 }
```

## 11 Inline CSS

### 11.1 Example

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

Here, we're using inline CSS to style a paragraph element directly. The text will appear in red.

## 12 Properties VS Attributes

In the context of web development, "property" and "attribute" have specific meanings, and they are not interchangeable. Let's clarify:

### 12.1 Attribute

Context: HTML

Description: Attributes provide additional information about an HTML element. They are always specified in the start tag and usually come in name/value pairs like `name="value"`.

## Examples

- The `href` attribute in `[Example</a>`](https://www.example.com/)
- The `src` and `alt` attributes in `![Description of Image](image.jpg)- The `type` attribute in `

## 12.2 Property

Context: CSS and JavaScript

Description:

- In CSS, properties define how elements are displayed. They are part of a declaration in a stylesheet or a style block. Each property has a value, and they are paired as `property: value`.
- In JavaScript, when working with the Document Object Model (DOM), elements are objects with properties that can be set, read, and changed. Some of these properties correspond to the HTML attributes, but there are many other properties as well.

## Examples

- In CSS: `color: red;`, `font-size: 16px;`, and `display: block;`
- In JavaScript, when accessing the `href` attribute of a link: `document.querySelector('a').href`

To illustrate the difference further:

- You might set an `href` **attribute** in **HTML** like this:  
`<a href="https://www.example.com/">Example</a>`
- In **JavaScript**, you might access or modify this using the `href` **property**:  
`document.querySelector('a').href = "https://www.newlink.com/";`
- In **CSS**, you might set the `display` **property** to control how an element is shown:  
`a { display: block; }`

In summary:

- **Attributes** are in **HTML** and provide additional information about elements.
- **Properties** can refer to **CSS** settings that determine how elements look or **JavaScript**-accessible settings on **DOM** objects that represent those elements.

## 13 Internal CSS

### 13.1 Example

```
<style>
  h1 {
    font-size: 24px;
  }
</style>
```

This internal CSS sets the font size of ``<h1>`` elements to 24 pixels.

## 14 CSS Selectors

### 14.1 Example

```
p {  
  color: red;  
  font-size: 16px;  
}
```

This CSS rule makes the text of all `

` elements bold.

## 15 CSS Properties

### 15.1 Example

```
body {  
  background-color: #f5f5f5;  
}
```

This sets the background colour of the entire webpage (`<body>`) to a light Gray.

## 16 The Box Model

### 16.1 Example

```
p {  
  padding: 10px;  
  border: 2px solid black;  
  margin: 20px;  
}
```

This styles a paragraph with 10 pixels of padding, a 2-pixel solid black border, and 20 pixels of margin on all sides.

## 17 CSS Layout Properties

### 17.1 Example

```
div {  
  display: flex;  
}
```

This sets a `

` element to use the flexible box layout, allowing its children to be laid out in any direction and have flexible sizes.



## 18 What is JavaScript?

JavaScript code is typically saved in a file with a `.js` extension. This external JavaScript file can then be linked to an HTML document using the `<script>` element.

For example, if you have a JavaScript file named `script.js`, you can link it to your HTML document like this:

```
<script src="script.js"></script>
```

It's common to place this `<script>` tag just before the closing `</body>` tag to ensure that the HTML content loads before the JavaScript executes, especially if the script manipulates the DOM or relies on the content being fully loaded.

Here are some benefits of using external JavaScript files:

1. **Separation of Concerns:** Just as with CSS, using external files keeps the structure (HTML) separate from the behaviour (JavaScript), making your code more organized and maintainable.
2. **Reusability:** The same `.js` file can be linked to multiple HTML pages, ensuring consistent behaviour across different pages of a website.
3. **Caching:** Browsers can cache external JavaScript files, which can improve page load times for returning visitors.
4. **Readability:** Keeping scripts in separate files can make the HTML document cleaner and easier to read, especially for larger projects.

While external JavaScript files are common, remember that JavaScript can also be written directly in the HTML document using `<script>` tags without the `src` attribute (known as internal or embedded JavaScript). However, for larger projects and better maintainability, external JavaScript files are recommended.

### 18.1 JavaScript VS Java

#### JavaScript

- Developed by Netscape as a way to add interactivity to websites. Brendan Eich created it in just 10 days in 1995, and it was originally named Mocha, then LiveScript, before finally being renamed to JavaScript.
- Primarily used for web development to make websites interactive. It's an essential part of the modern web development stack, often used alongside HTML and CSS. With the advent of Node.js, JavaScript can also be used for server-side programming.

#### Java

- Developed by Sun Microsystems (now owned by Oracle) as an object-oriented language that could run on any device or platform. It was introduced in 1995 by James Gosling.
- A general-purpose, object-oriented programming language used for a wide range of applications, from web applications to Android app development, desktop applications, and large-scale backend systems.

## 18.2 Example

```
alert("Hello, World!");
```

This JavaScript code displays an alert box with the message "Hello, World!".

## 19 Where to Place JavaScript?

### 19.1 Example

```
<script>  
    function showMessage() {  
        alert("Inline JavaScript");  
    }  
</script>
```

This inline JavaScript defines a function that, when called, will display an alert with the message "Inline JavaScript".

## 20 Basic JavaScript Syntax

### 20.1 Example

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

This code declares a variable named "name" with the value "John" and then logs it to the console.

## 21 Functions in JavaScript

### 21.1 Example

```
function greet() {  
    alert("Hello!");  
}
```

This defines a function named "greet" that displays an alert saying "Hello!" when called.

## 22 Conditions in JavaScript

### 22.1 Example

```
if (age > 18) {  
    console.log("Adult");  
} else {  
    console.log("Minor");  
}
```

This code checks if the variable "age" is greater than 18. If true, it logs "Adult" to the console; otherwise, it logs "Minor".

## 23 Loops in JavaScript

### 23.1 Example

```
for (var i = 0; i < 5; i++) {  
    console.log(i);  
}
```

This is a `for` loop that logs numbers 0 through 4 to the console.

## 24 Events in JavaScript

### 24.1 Example

```
<button onclick="alert('Button clicked!')">Click Me</button>
```

This button, when clicked, will trigger an alert with the message "Button clicked!".

## 25 Introduction to the DOM

- The DOM, or Document Object Model, is a programming interface for web documents.
- It represents the structure of a document as a tree of objects.
- Each object corresponds to a part of the document, such as an element or an attribute.
- The DOM provides a way for programs to manipulate the structure, style, and content of web documents.

### 25.1 Breakdown of the DOM

**Tree Structure:** The DOM represents a document as a tree of nodes. These nodes can be elements, attributes, text content, and more. The tree structure mirrors the structure of the HTML (or XML) document.

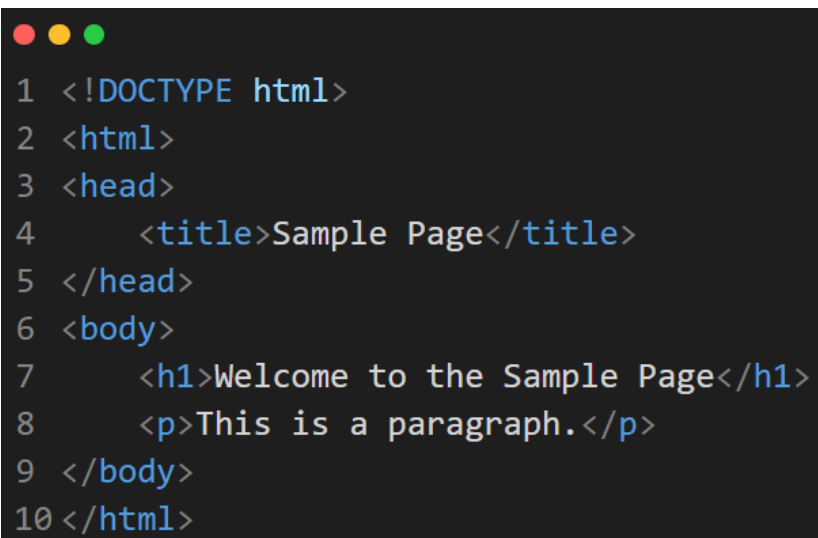
**Language-Neutral:** While the DOM is often accessed and manipulated using JavaScript, it's a language-neutral interface. This means that other programming languages can also interact with the DOM, although JavaScript is the most common in the context of web browsers.

**Dynamic:** The DOM is "live" and dynamic. Changes to the DOM immediately reflect on the rendered page. For example, if you use JavaScript to change the text of an element in the DOM, the visible page will update instantly.

**Standardized:** The DOM is standardized by the World Wide Web Consortium (W3C). This ensures consistency across different web browsers. However, there can still be some browser-specific quirks.

## 25.2 Example

Consider this simple HTML



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Sample Page</title>
5 </head>
6 <body>
7   <h1>Welcome to the Sample Page</h1>
8   <p>This is a paragraph.</p>
9 </body>
10 </html>
```

In the DOM, this would be represented as a tree:

- **Document** node at the root.
  - **HTML** element node.
    - **head** element node.
      - **title** element node.
        - Text node: "Sample Page"
    - **body** element node.
      - **h1** element node.
        - Text node: "Welcome to the Sample Page"
      - **p** element node.
        - Text node: "This is a paragraph."

Using JavaScript, you can access and manipulate any part of this tree.

For example, you might change the text of the `<h1>` element, add a new element, or apply a style.

## 25.3 Example

```
document.querySelector("button").addEventListener("click", function() {  
    alert("Button was clicked!");  
});
```

This code adds an event listener to a button. When the button is clicked, an alert will be displayed.

## 25.4 Example

```
var title = document.title;  
console.log(title);
```

This code retrieves the title of the document and logs it to the console.

## 26 Accessing Elements

### 26.1 Example

```
var heading = document.querySelector("h1");  
console.log(heading.textContent);
```

This code selects the first `<h1>` element on the page and logs its text content to the console.

## 27 Modifying Elements

### 27.1 Example

```
heading.textContent = "New Heading";
```

This code changes the text content of the previously selected `<h1>` element to "New Heading".

## 28 Adding and Removing Elements

### 28.1 Example

```
var newPara = document.createElement("p");  
newPara.textContent = "New paragraph."  
document.body.appendChild(newPara);
```

This code creates a new paragraph element, sets its text content, and then appends it to the body of the document.