HTML basic notes

## index:

**HTML Tutorial Overview**

### **Part One: HTML Basics**

* **HTML Introduction**
* **HTML Basics**
* **HTML Elements**
* **HTML Attributes**
* **HTML Headings**
* **HTML Paragraphs**
* **HTML Styles**
* **HTML Formatting**
* **HTML Quotations**
* **HTML Comments**

### **Part Two: Colors and Styling**

* **HTML Colors**
* **HTML RGB Colors**
* **HTML RGB and RGBA**
* **HTML HSL and HSLA**
* **HTML CSS**
* Inline CSS
* Internal CSS
* External CSS
* CSS Colors, Fonts, and Sizes
* CSS Border
* CSS Padding
* CSS Margin
* Linking to External CSS

### **Part Three: Links and Images**

* **HTML Links**
* Links
* Link Colors
* Link Bookmarks
* **HTML Images**
* Images
* Image Map
* Background Images
* Picture Element

### **Part Four: Tables and Lists**

* **HTML Tables**
* **HTML Lists**
* Ordered Lists
* Unordered Lists
* Other Lists

### **Part Five: Structure and Identification**

* **HTML Block and Inline Elements**
* **HTML Classes**
* **HTML ID**
* **HTML Iframes**
* **HTML JavaScript**
* **HTML File Path**
* **HTML Head**
* **HTML Layout**
* **HTML Responsive Design**
* **HTML Computer Code**
* **HTML Semantics**
* **HTML Style Guide**

### **Part Six: Special Characters and Encoding**

* **HTML Entities**
* **HTML Symbols**
* **HTML Emojis**
* **HTML Charset**
* **HTML URL Encode**
* **HTML vs XHTML**

### **Part Seven: Forms**

* **HTML Forms**
* HTML Form Attributes
* HTML Form Elements
* HTML Input Types
* HTML Input Attributes
* HTML Input Form Attributes

### **Part Eight: Graphics**

* **HTML Canvas**
* **HTML SVG**

### **Part Nine: Media**

* **HTML Media**
* HTML Video
* HTML Audio
* HTML Plugins
* HTML YouTube Integration

### **Part Ten: APIs**

* **HTML Geolocation**
* **HTML Drag and Drop**
* **HTML Web Storage**
* **HTML Web Workers**
* **HTML SSE (Server-Sent Events)**

**Part One: HTML Basics**:

**HTML Introduction:**

### **HTML Webpage Structure Overview**

To define the structure of a webpage, HTML uses **tags** to mark different elements, helping the browser understand how to display the content.

### **Basic HTML Structure**

<!DOCTYPE html>

<html>

<head>

<title>Page Title</title>

</head>

<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

</body>

</html>

### **Explanation of Elements**

1. <!DOCTYPE html> – Defines that the document is written in **HTML5**.
2. <html> – The **root element** of an HTML document.
3. <head> – Contains **meta-information** about the HTML document.

* <title> – Specifies the title of the webpage (shown on the browser’s title bar or tab).

1. <body> – Contains the **visible content** of the webpage.

* <h1> – Defines a heading.
* <p> – Defines a paragraph.

### **Empty Elements**

* <br> – An **empty element** used to insert a line break.
* Elements like <img> and <input> are also examples of empty elements because they don’t have closing tags or content.

### **Character Encoding**

* UTF-8 is the recommended encoding for HTML files.
* Save HTML files as **"All Files"** with the .html or .htm extension — both are functionally identical.

### **✅ Best Practices**

✔️ Use lowercase tags for consistency.  
 ✔️ Always close tags properly to avoid rendering issues.  
 ✔️ Use UTF-8 encoding for better character support across languages.

## **HTML Notes**

**HTML Basics**

### **DOCTYPE Declaration**

* <!DOCTYPE> – Declares the **document type** and helps browsers display pages correctly.
* Must appear **once** at the top of the document.
* **Not case-sensitive**.

Example for HTML5:

<!DOCTYPE html>

### **Attributes**

* Provide **extra information** about an HTML element.

**Example:**

* <a> – Defines a **link**.
* href – Specifies the link's destination.
* <img> – Defines an **image**.
* src – Source file.
* alt – Alternative text (for accessibility).
* width and height – Set the image size.

### **Example**

<!DOCTYPE html>

<html>

<body>

<h2>HTML Images</h2>

<p>HTML images are defined with the img tag:</p>

<img src="w3schools.jpg" alt="W3Schools.com" width="104" height="142">

<a href="https://www.w3schools.com">This is a link</a>

</body>

</html>

### **Editing on the Fly**

* You can **inspect** and **edit** HTML and CSS directly in the browser using Developer Tools.

**HTML Elements**

## **HTML Notes**

### **HTML Elements**

* An **HTML element** includes: <tagname>Content goes here...</tagname>
* Example: <p>This is a paragraph.</p>

### **Root Element**

* <html> – The **root element** that defines the whole HTML document.

### **Closing Tags**

✅ Always close tags properly.  
 ❌ Skipping end tags can work but may cause **errors** and **unexpected results**.

Example (correct):

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

Example (incorrect):

<p>This is a paragraph.

### **Empty Elements**

* Empty elements **have no content** and **don't need closing tags**.  
   Example:

<br> <!-- Line break -->

<img src="image.jpg" alt="Sample Image">

### **Case Sensitivity**

* **HTML** is **not case-sensitive**.
* **XHTML** is case-sensitive and requires lowercase tags.  
   ✅ Best Practice: Use **lowercase tags** for consistency.

**HTML Attributes**

### **What are Attributes?**

* Attributes provide **extra information** about an HTML element.
* Syntax: <tagname attribute="value">Content</tagname>

### **Common Attributes**

1. **href** – Specifies the destination URL for a link.

<a href="https://www.w3schools.com">Visit W3Schools</a>

1. **src** – Specifies the source of an image or media.

* **Absolute URL** – Links to an image on another site:

1. <img src="https://www.w3schools.com/images/img\_girl.jpg" width="500" height="600">

* **Relative URL** – Links to an image on the same site:

1. <img src="img\_girl.jpg" width="500" height="600">
2. **alt** – Provides alternative text for an image (for accessibility).

<img src="img\_girl.jpg" alt="Girl with a jacket" width="500" height="600">

1. **width** and **height** – Define the size of an image (in pixels).
2. **style** – Adds inline styles like color, font, size, etc.

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

1. **lang** – Declares the language of the webpage.

* Without country code:

1. <html lang="en">

* With country code:

1. <html lang="en-US">
2. **title** – Provides extra information about an element (displays as a tooltip).

<p title="I'm a tooltip">This is a paragraph.</p>

### **Quotes in Attributes**

* Use **double quotes** or **single quotes** for attribute values.
* If the value contains one type of quote, use the other type:

<p title='John "ShotGun" Nelson'></p>

<p title="John 'ShotGun' Nelson"></p>

### **Attributes Summary**

✅ href – Link destination (used in <a>)  
 ✅ src – Source of an image (used in <img>)  
 ✅ width, height – Image size (in pixels)  
 ✅ alt – Alternate text for an image  
 ✅ style – Inline styling  
 ✅ lang – Language of the webpage  
 ✅ title – Extra information shown as a tooltip

### **✅ Best Practices**

✔️ Always use **lowercase** attributes.  
 ✔️ Use **quotes** around attribute values.  
 ✔️ Include the lang attribute for better SEO and accessibility.

### **HTML Headings**

* Headings range from <h1> to <h6>.
* <h1> is the **most important** heading, <h6> is the least important.
* Search engines use headings to understand the **structure** of a webpage.
* Example:

<h1>This is Heading 1</h1>

<h2>This is Heading 2</h2>

<h3>This is Heading 3</h3>

* **Styling Headings:** Use the style attribute to change the heading size:

<h1 style="font-size:60px;">Heading 1</h1>

### **HTML Paragraphs**

* <p> defines a **paragraph**.
* Extra spaces or new lines in the code are ignored by the browser.

Example:

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

### **Horizontal Rules**

* <hr> defines a **thematic break** (horizontal line).
* It is an **empty element** (no closing tag).

Example:

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

<hr>

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

### **Line Breaks**

* <br> inserts a **line break** without starting a new paragraph.
* It is an **empty element** (no closing tag).

Example:

<p>This is a line<br>with a break.</p>

### **The Poem Problem**

* Use <pre> for **preformatted text** to preserve spaces and line breaks.
* Displays text in a **fixed-width font** (like Courier).

Example:

<pre>

Roses are red,

Violets are blue,

Sugar is sweet,

And so are you.

</pre>

### **Tags Summary**

|  |  |
| --- | --- |
| **Tag** | **Description** |
| <p> | Defines a paragraph |
| <hr> | Defines a thematic change in content |
| <br> | Inserts a single line break |
| <pre> | Displays pre-formatted text |

### **✅ Best Practices**

✔️ Use <h1> only **once** per page.  
 ✔️ Use headings in **hierarchical order**.  
 ✔️ Avoid adding extra spaces for formatting — use CSS instead

### **HTML style Attribute**

* The style attribute is used to **add CSS styles** directly to HTML elements.
* Syntax:

<tagname style="property:value;">

### **Common CSS Properties in HTML**

1. **Background Color**

* Sets the **background color** of an element.

<body style="background-color:powderblue;">

<h1 style="background-color:powderblue;">This is a heading</h1>

<p style="background-color:tomato;">This is a paragraph.</p>

1. **Text Color**

* Changes the **text color** of an element.

<h1 style="color:blue;">This is a heading</h1>

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

1. **Fonts**

* Sets the **font family** for text.

<h1 style="font-family:verdana;">This is a heading</h1>

<p style="font-family:courier;">This is a paragraph.</p>

1. **Text Size**

* Changes the **size** of text.

<h1 style="font-size:300%;">This is a heading</h1>

<p style="font-size:160%;">This is a paragraph.</p>

1. **Text Alignment**

* Aligns text to the **left**, **center**, or **right**.

<h1 style="text-align:center;">Centered Heading</h1>

<p style="text-align:center;">Centered paragraph.</p>

### **Styles Summary**

|  |  |
| --- | --- |
| **Property** | **Description** |
| background-color | Sets the background color |
| color | Sets the text color |
| font-family | Defines the text font |
| font-size | Changes the text size |
| text-align | Aligns text (left, center, right) |

### **✅ Best Practices**

✔️ Use **external CSS** for better scalability.  
 ✔️ Use the style attribute for **quick testing** or **one-off changes**.  
 ✔️ Follow consistent styling for a clean design.

## **HTML Formatting Notes**

### **Common HTML Formatting Tags**

1. **Bold Text**

* <b> – Makes text **bold** (for styling only).

<b>This text is bold</b>

1. **Important Text**

* <strong> – Makes text bold and marks it as **important** for screen readers.

<strong>This text is important</strong>

1. **Italic Text**

* <i> – Makes text **italic** (used for technical terms or foreign phrases).

<i>This text is italic</i>

1. **Emphasized Text**

* <em> – Italicizes text and marks it as **emphasized** (used by screen readers).

<em>This text is emphasized</em>

1. **Marked Text**

* <mark> – Highlights text. You can customize the color with CSS.

<mark>This text is marked</mark>

1. **Small Text**

* <small> – Reduces text size.

<small>This is small text</small>

1. **Deleted Text**

* <del> – Strikes through text (used for deleted content).

<p>My favorite color is <del>blue</del>.</p>

1. **Inserted Text**

* <ins> – Underlines inserted text (used for added content).

<p>My favorite color is <del>blue</del> <ins>red</ins>.</p>

1. **Subscript Text**

* <sub> – Displays text **below the normal line** (used for chemical formulas).

H<sub>2</sub>O

1. **Superscript Text**

* <sup> – Displays text **above the normal line** (used for footnotes and exponents).

x<sup>2</sup>

### **Example**

<p>My favorite color is <del>blue</del> <ins>red</ins>.</p>

<p>This is <b>bold</b> and <i>italic</i> text.</p>

<p>Water formula is H<sub>2</sub>O</p>

<p>Square of x is x<sup>2</sup></p>

### **Tags Summary**

|  |  |
| --- | --- |
| **Tag** | **Description** |
| <b> | Bold text |
| <strong> | Important text |
| <i> | Italic text |
| <em> | Emphasized text |
| <mark> | Marked (highlighted) text |
| <small> | Small text |
| <del> | Deleted text (strikethrough) |
| <ins> | Inserted text (underlined) |
| <sub> | Subscript text |
| <sup> | Superscript text |

### **✅ Best Practices**

✔️ Use <strong> and <em> for **semantic meaning** (for screen readers).  
 ✔️ Use <b> and <i> only for **visual styling**.  
 ✔️ Use <mark> for **highlighting** key information.

## **HTML Quotations and Citations Notes**

### **1. Abbreviation (<abbr>)**

* Defines an **abbreviation** or **acronym**.
* Use the title attribute to display the full meaning when hovered over.

**Example:**

<p>The <abbr title="World Health Organization">WHO</abbr> was founded in 1948.</p>

### **2. Address (<address>)**

* Defines **contact information** (email, URL, phone number, etc.).
* Browsers add a **line break** before and after the element.

**Example:**

<address>

Written by John Doe.<br>

Visit us at:<br>

Example.com<br>

Box 564, Disneyland<br>

USA

</address>

### **3. Bi-Directional Override (<bdo>)**

* Overrides the **text direction**.
* Use dir="rtl" for right-to-left or dir="ltr" for left-to-right text.

**Example:**

<bdo dir="rtl">This text will be written from right to left</bdo>

### **4. Blockquote (<blockquote>)**

* Defines a **long quotation** from another source.
* Use the cite attribute to reference the source (not clickable).

**Example:**

<p>Here is a quote from WWF's website:</p>

<blockquote cite="http://www.worldwildlife.org/who/index.html">

For 50 years, WWF has been protecting the future of nature.

</blockquote>

### **5. Short Quotations (<q>)**

* Defines a **short inline quotation**.
* Browsers automatically add quotation marks.

**Example:**

<p>WWF's goal is to: <q>Build a future where people live in harmony with nature.</q></p>

### **6. Citation (<cite>)**

* Defines the **title of a creative work** (e.g., book, movie, painting).
* Browsers usually display it in **italic**.

**Example:**

<p><cite>The Scream</cite> by Edvard Munch. Painted in 1893.</p>

### **Tags Summary**

|  |  |
| --- | --- |
| **Tag** | **Description** |
| <abbr> | Defines an abbreviation or acronym |
| <address> | Defines contact information |
| <bdo> | Overrides text direction |
| <blockquote> | Defines a block quote |
| <q> | Defines a short inline quote |
| <cite> | Defines the title of a work |

### **✅ Best Practices**

✔️ Use <abbr> to improve **SEO** and **accessibility**.  
 ✔️ Use <cite> for **work titles** (not author names).  
 ✔️ Use <blockquote> for **long quotes** and <q> for **short quotes**.  
 ✔️ Keep <address> limited to contact info only.

## **HTML Comments Notes**

### **HTML Comments Syntax**

* HTML comments start with <!-- and end with -->.
* Comments are **ignored by the browser** and **not displayed** on the webpage.
* Useful for adding **notes** or **disabling code** during testing.

### **Example:**

<!-- This is a comment -->

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

<!-- Remember to add more information here -->

### **Use Cases:**

✅ Add notes for **clarity** in the code.  
 ✅ Temporarily **disable code** for testing.  
 ✅ Explain complex sections of code to other developers.

### **Best Practices:**

✔️ Keep comments **short and meaningful**.  
 ✔️ Avoid overusing comments — write **clean code** instead.  
 ✔️ Use comments to **explain why** rather than **what** the code does.

### **✅ Summary:**

|  |  |
| --- | --- |
| **Syntax** | **Purpose** |
| <!-- comment --> | Add notes or disable code |

## **HTML Colors Notes**

### **1. Color Types**

You can specify colors using:  
 ✅ **Color names** – 140 standard names (e.g., Tomato, DodgerBlue)  
 ✅ **RGB** – rgb(255, 99, 71) (Red, Green, Blue)  
 ✅ **HEX** – #ff6347  
 ✅ **HSL** – hsl(9, 100%, 64%) (Hue, Saturation, Lightness)  
 ✅ **RGBA** – rgba(255, 99, 71, 0.5) (Alpha controls transparency)  
 ✅ **HSLA** – hsla(9, 100%, 64%, 0.5)

### **2. Background Color**

Use background-color to set the background color of an element.

<h1 style="background-color:DodgerBlue;">Hello World</h1>

<p style="background-color:Tomato;">Lorem ipsum...</p>

### **3. Text Color**

Use color to change the text color.

<h1 style="color:Tomato;">Hello World</h1>

<p style="color:DodgerBlue;">Lorem ipsum...</p>

### **4. Border Color**

Use border to define the color of the border around an element.

<h1 style="border:2px solid Tomato;">Hello World</h1>

<h1 style="border:2px solid DodgerBlue;">Hello World</h1>

<h1 style="border:2px solid Violet;">Hello World</h1>

### **5. RGB and RGBA**

* rgb(r, g, b) – Each value ranges from **0 to 255**.
* rgba(r, g, b, a) – a defines **transparency** (0 = fully transparent, 1 = fully opaque).

**Example:**

<h1 style="background-color:rgb(255, 99, 71);">RGB Example</h1>

<h1 style="background-color:rgba(255, 99, 71, 0.5);">RGBA Example</h1>

|  |  |
| --- | --- |
| **Color** | **RGB Value** |
| **Red** | rgb(255, 0, 0) |
| **Green** | rgb(0, 255, 0) |
| **Blue** | rgb(0, 0, 255) |
| **Black** | rgb(0, 0, 0) |
| **White** | rgb(255, 255, 255) |
| **Gray** | rgb(x, x, x) where x = 0–255 |

### **6. HEX Colors**

* Uses **hexadecimal values** for Red, Green, and Blue (00 to ff).
* #ff0000 = Red, #00ff00 = Green, #0000ff = Blue

**Examples:**

<h1 style="background-color:#ff6347;">HEX Example</h1>

|  |  |
| --- | --- |
| **Color** | **HEX Code** |
| **Red** | #ff0000 |
| **Green** | #00ff00 |
| **Blue** | #0000ff |
| **Black** | #000000 |
| **White** | #ffffff |
| **Gray Shades** | #404040, #686868, #a0a0a0 |

### **7. HSL and HSLA**

* hsl(hue, saturation, lightness)
* hsla(hue, saturation, lightness, alpha)

**Example:**

<h1 style="background-color:hsl(9, 100%, 64%);">HSL Example</h1>

<h1 style="background-color:hsla(9, 100%, 64%, 0.5);">HSLA Example</h1>

#### **HSL: Hue, Saturation, and Lightness**

* **Hue** – Degree on the color wheel (0 to 360)
* 0 = Red, 120 = Green, 240 = Blue
* **Saturation** – Intensity of color (0% to 100%)
* 0% = Gray
* 100% = Full color (no gray)
* 50% = 50% gray, still showing some color
* **Lightness** – Brightness of color (0% to 100%)
* 0% = Black
* 50% = Normal color
* 100% = White

✅ **Example Shades of Gray:**

* hsl(0, 0%, 20%) → Darker gray
* hsl(0, 0%, 70%) → Lighter gray

#### **HSLA: Hue, Saturation, Lightness, and Alpha**

* **Alpha** – Controls transparency (0 to 1)
* 0 = Fully transparent
* 1 = Fully opaque

✅ **Example:**

<h1 style="background-color:hsla(240, 100%, 50%, 0.3);">HSLA Example</h1>

|  |  |
| --- | --- |
| **Component** | **Description** |
| **Hue** | Color type (0–360) → 0 = Red, 120 = Green, 240 = Blue |
| **Saturation** | Intensity of color (0% = gray, 100% = full color) |
| **Lightness** | Brightness of color (0% = black, 100% = white) |
| **Alpha** | Transparency (0 = fully transparent, 1 = fully opaque) |

### **✅ Summary**

|  |  |
| --- | --- |
| **Type** | **Example** |
| **Color Name** | Tomato, DodgerBlue |
| **RGB** | rgb(255, 99, 71) |
| **RGBA** | rgba(255, 99, 71, 0.5) |
| **HEX** | #ff6347 |
| **HSL** | hsl(9, 100%, 64%) |
| **HSLA** | hsla(9, 100%, 64%, 0.5) |

### **🚀 Best Practices**

✔️ Use **HEX** for consistency across browsers.  
 ✔️ Use **RGB/RGBA** when you need color flexibility.  
 ✔️ Use **HSL/HSLA** for better readability and control over saturation and lightness.

## **HTML CSS Notes**

### **1. What is CSS?**

✅ **CSS (Cascading Style Sheets)** is used to control the style and layout of HTML pages.  
 ✅ CSS allows you to define:

* Colors, fonts, and sizes
* Spacing between elements
* Positioning and layout of elements
* Background images and colors
* Styling for different devices and screen sizes

### **2. Why "Cascading"?**

* A style applied to a **parent element** is **inherited** by all child elements unless overridden.
* Example: If you set color: blue on the <body>, all text inside it will be blue unless styled differently.

### **3. Ways to Add CSS**

You can add CSS to HTML in **three ways**:

|  |  |  |
| --- | --- | --- |
| **Type** | **Description** | **Example** |
| **Inline CSS** | Directly inside an HTML element | <h1 style="color:blue;">Hello</h1> |
| **Internal CSS** | Inside a <style> tag in the <head> section | <style> h1 { color: blue; } </style> |
| **External CSS** | In a separate .css file linked via <link> | <link rel="stylesheet" href="styles.css"> |

### **4. Inline CSS**

* Use the style attribute to apply styles directly to a single element.

<h1 style="color:blue;">A Blue Heading</h1>

<p style="color:red;">A Red Paragraph</p>

### **5. Internal CSS**

* Use the <style> element inside the <head> section to apply styles to a single HTML document.

<!DOCTYPE html>

<html>

<head>

<style>

body { background-color: powderblue; }

h1 { color: blue; }

p { color: red; }

</style>

</head>

<body>

<h1>This is a heading</h1>

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

</body>

</html>

### **6. External CSS**

* Use the <link> element in the <head> to link to an external CSS file.
* Create a styles.css file and reference it like this:

<!DOCTYPE html>

<html>

<head>

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

</head>

<body>

<h1>This is a heading</h1>

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

</body>

</html>

✅ **styles.css**

body {

background-color: powderblue;

}

h1 {

color: blue;

}

p {

color: red;

}

### **7. CSS Fonts, Colors, and Sizes**

You can set:

* **Text color** using color
* **Font** using font-family
* **Font size** using font-size

<style>

h1 {

color: blue;

font-family: verdana;

font-size: 300%;

}

p {

color: red;

font-family: courier;

font-size: 160%;

}

</style>

### **8. CSS Borders**

* The border property defines a border around an HTML element.
* Format: border: width style color;

p {

border: 2px solid powderblue;

}

|  |  |  |
| --- | --- | --- |
| **Style** | **Description** | **Example** |
| solid | Solid line | border: 2px solid red; |
| dotted | Dotted line | border: 2px dotted red; |
| dashed | Dashed line | border: 2px dashed red; |
| double | Double line | border: 2px double red; |
| groove | Carved effect | border: 2px groove red; |

### **9. CSS Padding**

* The padding property sets space between the **content** and the **border**.
* Format: padding: top right bottom left;

p {

border: 2px solid powderblue;

padding: 30px;

}

✅ **Shortcut Format:**

* padding: 10px; → All sides
* padding: 10px 20px; → Top & Bottom, Left & Right
* padding: 10px 20px 30px; → Top, Left & Right, Bottom
* padding: 10px 20px 30px 40px; → Top, Right, Bottom, Left

### **10. CSS Margin**

* The margin property sets space **outside** the border.
* Format: margin: top right bottom left;

p {

border: 2px solid powderblue;

margin: 50px;

}

✅ **Shortcut Format:**

* margin: 10px; → All sides
* margin: 10px 20px; → Top & Bottom, Left & Right
* margin: 10px 20px 30px; → Top, Left & Right, Bottom
* margin: 10px 20px 30px 40px; → Top, Right, Bottom, Left

### **11. Linking External CSS**

You can reference an external style sheet with a **URL** or a **relative path**.

✅ **Example – Absolute URL**

<link rel="stylesheet" href="https://www.w3schools.com/html/styles.css">

✅ **Example – Relative Path**

<link rel="stylesheet" href="/html/styles.css">

### **12. CSS Best Practices**

✅ Use **External CSS** for consistent styling across pages.  
 ✅ Use **Internal CSS** for single pages or unique styles.  
 ✅ Use **Inline CSS** only for quick testing or unique elements.  
 ✅ Group similar rules together for better readability.

### **✅ Summary**

|  |  |  |
| --- | --- | --- |
| **Property** | **Description** | **Example** |
| **color** | Text color | color: red; |
| **font-family** | Font type | font-family: Arial; |
| **font-size** | Font size | font-size: 16px; |
| **border** | Border style | border: 1px solid black; |
| **padding** | Space inside the border | padding: 10px; |
| **margin** | Space outside the border | margin: 10px; |

### **🚀 Best Practices**

✔️ Use **external CSS** for large projects.  
 ✔️ Keep **CSS files organized** and modular.  
 ✔️ Use **comments** to explain complex rules.  
 ✔️ Use **CSS variables** to maintain consistency.  
 ✔️ Minify CSS for better performance in production.

## **HTML Links**

HTML links (or hyperlinks) allow users to navigate to other web pages, sections within a page, or external resources. When you hover over a link, the cursor changes to a hand icon.

✅ **Quick Facts:** ✔️ Links can be **text, images, buttons, or other HTML elements**.  
 ✔️ Links can point to **external websites**, **internal pages**, **files**, **email addresses**, and more.  
 ✔️ Links can be styled using **CSS** to change colors, hover effects, and more.

## **1. Basic HTML Link**

The <a> tag defines a hyperlink:

<a href="url">Link Text</a>

* href → The destination of the link.
* The **link text** is what the user sees and clicks.

✅ **Example:**

<a href="https://www.w3schools.com/">Visit W3Schools.com!</a>

## **2. Default Link Styles (All Browsers)**

By default, links appear as follows:

* **Unvisited link** → Blue and underlined
* **Visited link** → Purple and underlined
* **Active link** (while being clicked) → Red and underlined

## **3. Target Attribute**

The target attribute defines where to open the link:

|  |  |
| --- | --- |
| **Value** | **Description** |
| \_self | Default. Opens in the same window/tab |
| \_blank | Opens in a new window/tab |
| \_parent | Opens in the parent frame |
| \_top | Opens in the full body of the window |

✅ **Example:**

<a href="https://www.example.com" target="\_blank">Open in New Tab</a>

## **4. Absolute vs Relative URLs**

### **✅ Absolute URL – Full web address with protocol:**

<a href="https://www.w3.org/">W3C</a>

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

### **✅ Relative URL – Path relative to the current domain:**

<a href="html\_images.asp">HTML Images</a>

<a href="/css/default.asp">CSS Tutorial</a>

## **5. Link Without href Attribute**

If the href attribute is missing, the link will not be clickable:

<a>Non-clickable text</a>

## **6. Link to an Email Address**

Use mailto: inside href to create an email link:

<a href="mailto:someone@example.com">Send Email</a>

You can also add a **subject** and **body**:

<a href="mailto:someone@example.com?subject=Hello&body=How are you?">

Send Email with Subject

</a>

## **7. Link to a Phone Number**

Use tel: to create a clickable phone number:

<a href="tel:+1234567890">Call Us</a>

## **8. Open WhatsApp Chat**

You can create a direct link to open a WhatsApp chat:

<a href="https://wa.me/1234567890">Chat on WhatsApp</a>

## **9. Open Google Maps Location**

Create a link to a location on Google Maps:

<a href="https://www.google.com/maps?q=Eiffel+Tower">View on Google Maps</a>

## **10. Open YouTube Video at a Specific Time**

You can open a video at a specific timestamp:

<a href="https://www.youtube.com/watch?v=abcdefghij&t=60s">

Watch at 1 Minute

</a>

## **11. Use an Image as a Link**

An image can act as a clickable link:

<a href="default.asp">

<img src="smiley.gif" alt="HTML tutorial" style="width:42px;height:42px;">

</a>

## **12. Link as a Button**

You can make a button work as a link using JavaScript:

<button onclick="window.location.href='https://example.com'">

Go to Example

</button>

## **13. Link Title**

Use the title attribute to show a tooltip on hover:

<a href="https://www.w3schools.com/html/" title="Go to W3Schools HTML section">

Visit our HTML Tutorial

</a>

## **14. Downloadable Link**

Use the download attribute to trigger a file download:

<a href="files/sample.pdf" download="Sample.pdf">Download PDF</a>

## **15. FTP Link**

You can link to an FTP server:

<a href="ftp://ftp.example.com/file.zip">Download from FTP</a>

## **16. Data URL Link**

You can create a link to a base64-encoded data URL:

<a href="data:text/plain;base64,SGVsbG8gd29ybGQ=" download="hello.txt">

Download Text File

</a>

## **17. Create Bookmarks (Jump Links)**

You can create links to specific parts of a page using id.

✅ **Create Bookmark:**

<h2 id="C4">Chapter 4</h2>

✅ **Link to Bookmark Within the Same Page:**

<a href="#C4">Jump to Chapter 4</a>

✅ **Link to Bookmark on Another Page:**

<a href="html\_demo.html#C4">Jump to Chapter 4</a>

## **18. Remove Underline from Link**

You can remove the underline using text-decoration:

<a href="html\_images.asp" style="text-decoration:none">HTML Images</a>

## **19. Secure Links with noopener noreferrer**

Use rel="noopener noreferrer" with target="\_blank" to avoid security risks:

<a href="https://example.com" target="\_blank" rel="noopener noreferrer">

Open Securely

</a>

## **20. Open a PDF File in a New Tab**

<a href="file.pdf" target="\_blank">Open PDF</a>

## **21. Styling Links with CSS**

### **✅ Default CSS for Links:**

<style>

a {

color: blue;

text-decoration: none;

}

a:hover {

color: red;

text-decoration: underline;

}

</style>

### **✅ Style Link Like a Button:**

<style>

a {

display: inline-block;

padding: 10px 20px;

background-color: #008CBA;

color: white;

text-decoration: none;

border-radius: 5px;

}

a:hover {

background-color: #005f73;

}

</style>

## **✅ Summary**

|  |  |  |
| --- | --- | --- |
| **Type of Link** | **Example** | **Purpose** |
| **Text Link** | <a href="https://example.com">Link</a> | Basic hyperlink |
| **Image Link** | <a href="#"><img src="img.jpg"></a> | Link through an image |
| **Email Link** | <a href="mailto:email@example.com">Email</a> | Open email client |
| **Phone Link** | <a href="tel:+1234567890">Call</a> | Open phone dialer |
| **WhatsApp Link** | <a href="https://wa.me/1234567890">Chat</a> | Open WhatsApp |
| **JavaScript Link** | <a href="javascript:alert('Hi!')">Click</a> | Execute JS |
| **Bookmark Link** | <a href="#section">Jump</a> | Jump to a section |

💡 **Best Practice:** ✔️ Use noopener noreferrer with target="\_blank" for security.  
 ✔️ Add alt to images for accessibility.  
 ✔️ Avoid using javascript: links unless necessary.

## **HTML Images**

HTML images are used to display pictures on a webpage. Images are linked to a webpage, not embedded. The browser requests the image from a server and inserts it into the page when loaded.

## **1. Basic HTML Image**

The <img> tag defines an image:

<img src="pic\_trulli.jpg" alt="Italian Trulli">

* src → Specifies the path to the image.
* alt → Specifies alternate text displayed if the image fails to load.

✅ **Example:**

<img src="img\_girl.jpg" alt="Girl in a jacket" style="width:500px;height:600px;">

## **2. src Attribute**

The src attribute defines the **source** (path) of the image.

* **Local file:**

<img src="images/logo.png" alt="Logo">

* **Remote file:**

<img src="https://example.com/image.jpg" alt="Example">

✅ **Best Practices:** ✔️ Use relative paths for internal images.  
 ✔️ Use full URLs for external images.  
 ✔️ Ensure that the image source is accessible and not restricted by permissions.

## **3. alt Attribute**

The alt attribute provides alternate text for an image.

* Shown when the image fails to load.
* Improves accessibility for visually impaired users (used by screen readers).

✅ **Example:**

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

✅ **Best Practices:** ✔️ Always add a meaningful alt text.  
 ✔️ Use alt="" for decorative images that don’t need to be described.

## **4. Width and Height**

You can define the size of an image using:

1. **Inline style attribute** (preferred):

<img src="img.jpg" alt="Image" style="width:500px;height:300px;">

1. **HTML attributes**:

<img src="img.jpg" alt="Image" width="500" height="300">

✅ **Best Practices:** ✔️ Prefer style attribute or CSS over width and height attributes.  
 ✔️ Use responsive CSS (width: 100%) for better scaling on different screen sizes.

## **5. Responsive Images**

Use CSS to make images responsive:

<style>

img {

width: 100%; /\* Scales to parent container \*/

height: auto;

}

</style>

✅ **Example:**

<img src="img.jpg" alt="Responsive Image">

## **6. Images in Subfolders**

If the image is inside a folder, include the folder name:

<img src="/images/html5.gif" alt="HTML5 Icon" style="width:128px;height:128px;">

## **7. External Images**

If the image is hosted on another server, specify the full URL:

<img src="https://example.com/image.jpg" alt="Example">

✅ **Notes:** ✔️ External images may be removed or blocked by the site owner.  
 ✔️ Copyright permission may be required.

## **8. Animated Images (GIF)**

HTML supports animated images using the .gif format:

<img src="programming.gif" alt="Computer Man" style="width:48px;height:48px;">

## **9. Use an Image as a Link**

You can use an image as a clickable link:

<a href="default.asp">

<img src="smiley.gif" alt="HTML tutorial" style="width:42px;height:42px;">

</a>

## **10. Floating Images**

Use float to align images to the left or right of text:

✅ **Float Right:**

<p>

<img src="smiley.gif" alt="Smiley face" style="float:right;width:42px;height:42px;">

The image will float to the right of the text.

</p>

✅ **Float Left:**

<p>

<img src="smiley.gif" alt="Smiley face" style="float:left;width:42px;height:42px;">

The image will float to the left of the text.

</p>

## **11. Supported Image Formats**

|  |  |  |
| --- | --- | --- |
| **Abbreviation** | **File Format** | **Extensions** |
| **APNG** | Animated Portable Network Graphics | .apng |
| **GIF** | Graphics Interchange Format | .gif |
| **ICO** | Microsoft Icon | .ico, .cur |
| **JPEG** | Joint Photographic Experts Group | .jpg, .jpeg, .jfif, .pjpeg, .pjp |
| **PNG** | Portable Network Graphics | .png |
| **SVG** | Scalable Vector Graphics | .svg |

## **12. Image Maps**

An image map allows you to create **clickable areas** within an image.

✅ **Example:**

<img src="workplace.jpg" alt="Workplace" usemap="#workmap">

<map name="workmap">

<area shape="rect" coords="34,44,270,350" href="desk.html">

<area shape="circle" coords="337,300,44" href="coffee.html">

</map>

✅ **Shapes:**

|  |  |
| --- | --- |
| **Shape** | **Description** |
| rect | Rectangular clickable area |
| circle | Circular clickable area |
| poly | Polygonal clickable area |

✅ **Example with JavaScript:**

<map name="workmap">

<area shape="circle" coords="337,300,44" onclick="myFunction()">

</map>

<script>

function myFunction() {

alert("You clicked the coffee cup!");

}

</script>

## **13. Background Images**

You can add background images using the background-image property in CSS.

✅ **Example (Inline):**

<div style="background-image: url('img\_girl.jpg');">

Text over the background image

</div>

✅ **Example (CSS):**

<style>

body {

background-image: url('img\_girl.jpg');

}

</style>

✅ **Background Repeat:**

<style>

body {

background-image: url('img\_girl.jpg');

background-repeat: no-repeat;

}

</style>

✅ **Background Cover:**

<style>

body {

background-image: url('img\_girl.jpg');

background-repeat: no-repeat;

background-size: cover;

background-attachment: fixed;

}

</style>

✅ **Background Stretch:**

<style>

body {

background-image: url('img\_girl.jpg');

background-repeat: no-repeat;

background-size: 100% 100%;

}

</style>

## **14. Picture Element**

The <picture> element allows you to load different images based on the device size or resolution.

✅ **Example:**

<picture>

<source media="(min-width: 650px)" srcset="img\_large.jpg">

<source media="(min-width: 465px)" srcset="img\_medium.jpg">

<img src="img\_small.jpg" alt="Image">

</picture>

✅ **Best Practices:** ✔️ Use picture for responsive design.  
 ✔️ Ensure fallback using the <img> tag.

## **✅ Summary**

|  |  |
| --- | --- |
| **Tag** | **Purpose** |
| <img> | Defines an image |
| <map> | Defines an image map |
| <area> | Defines a clickable area inside an image map |
| <picture> | Defines a container for multiple image sources |

💡 **Best Practices:** ✔️ Use descriptive alt text.  
 ✔️ Avoid large images to improve load time.  
 ✔️ Ensure permission for external images to avoid copyright issues.  
 ✔️ Use responsive images for better user experience on all devices.

This version is more organized, clear, and professional. Let me know if you’d like to adjust or add anything!

:

## **HTML Tables**

HTML tables allow you to organize data into rows and columns.

## **1. Basic HTML Table**

The <table> tag defines a table.

* <tr> → Defines a row.
* <th> → Defines a table header (bold and centered by default).
* <td> → Defines table data/cell (regular and left-aligned by default).

✅ **Example:**

<table style="width:100%">

<tr>

<th>Firstname</th>

<th>Lastname</th>

<th>Age</th>

</tr>

<tr>

<td>Jill</td>

<td>Smith</td>

<td>50</td>

</tr>

<tr>

<td>Eve</td>

<td>Jackson</td>

<td>94</td>

</tr>

</table>

✅ **Output:**

|  |  |  |
| --- | --- | --- |
| **Firstname** | **Lastname** | **Age** |
| Jill | Smith | 50 |
| Eve | Jackson | 94 |

## **2. Add a Table Border**

Use the border property in CSS to add a border to the table and cells:

<style>

table, th, td {

border: 1px solid black;

}

</style>

✅ **Output:**

|  |  |  |
| --- | --- | --- |
| **Firstname** | **Lastname** | **Age** |
| Jill | Smith | 50 |
| Eve | Jackson | 94 |

## **3. Collapsed Table Border**

Use border-collapse to merge the borders of table cells:

<style>

table, th, td {

border: 1px solid black;

border-collapse: collapse;

}

</style>

## **4. Add Cell Padding**

Use the padding property to control the space inside table cells:

<style>

th, td {

padding: 15px;

}

</style>

## **5. Left-Align Headings**

By default, table headings are bold and centered. Use text-align to change alignment:

<style>

th {

text-align: left;

}

</style>

## **6. Add Border Spacing**

Use border-spacing to control the space **between** table cells:

<style>

table {

border-spacing: 5px;

}

</style>

✅ **Note:** border-spacing has no effect if border-collapse: collapse; is used.

## **7. Merge Cells Across Columns (colspan)**

The colspan attribute makes a cell span multiple columns:

<table style="width:100%">

<tr>

<th>Name</th>

<th colspan="2">Telephone</th>

</tr>

<tr>

<td>Bill Gates</td>

<td>55577854</td>

<td>55577855</td>

</tr>

</table>

✅ **Output:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Telephone** | **Telephone** |
| Bill Gates | 55577854 | 55577855 |

## **8. Merge Cells Across Rows (rowspan)**

The rowspan attribute makes a cell span multiple rows:

<table style="width:100%">

<tr>

<th>Name:</th>

<td>Bill Gates</td>

</tr>

<tr>

<th rowspan="2">Telephone:</th>

<td>55577854</td>

</tr>

<tr>

<td>55577855</td>

</tr>

</table>

✅ **Output:**

|  |  |
| --- | --- |
| **Name** | **Bill Gates** |
| Telephone | 55577854 |
|  | 55577855 |

## **9. Add a Table Caption**

The <caption> tag provides a description of the table:

<table style="width:100%">

<caption>Monthly Savings</caption>

<tr>

<th>Month</th>

<th>Savings</th>

</tr>

<tr>

<td>January</td>

<td>$100</td>

</tr>

<tr>

<td>February</td>

<td>$50</td>

</tr>

</table>

✅ **Output:  
Monthly Savings**

|  |  |
| --- | --- |
| **Month** | **Savings** |
| January | $100 |
| February | $50 |

## **10. Style a Specific Table**

You can use an id to target and style a specific table:

✅ **HTML:**

<table id="t01">

<tr>

<th>Firstname</th>

<th>Lastname</th>

<th>Age</th>

</tr>

<tr>

<td>Eve</td>

<td>Jackson</td>

<td>94</td>

</tr>

</table>

✅ **CSS:**

<style>

#t01 {

width: 100%;

background-color: #f1f1c1;

}

#t01 tr:nth-child(even) {

background-color: #eee;

}

#t01 tr:nth-child(odd) {

background-color: #fff;

}

#t01 th {

color: white;

background-color: black;

}

</style>

✅ **Output:**

|  |  |  |
| --- | --- | --- |
| **Firstname** | **Lastname** | **Age** |
| Eve | Jackson | 94 |

## **11. Responsive Table**

Use overflow-x: auto; to make tables scrollable on small screens:

<div style="overflow-x:auto;">

<table>

<tr>

<th>Firstname</th>

<th>Lastname</th>

<th>Age</th>

</tr>

<tr>

<td>Eve</td>

<td>Jackson</td>

<td>94</td>

</tr>

</table>

</div>

✅ **Best Practices:** ✔️ Add overflow-x: auto; to make the table scrollable.  
 ✔️ Keep tables simple for better readability on small screens.

## **✅ Summary**

|  |  |
| --- | --- |
| **Tag/Attribute** | **Description** |
| <table> | Defines a table |
| <tr> | Defines a table row |
| <th> | Defines a table heading |
| <td> | Defines table data/cell |
| <caption> | Adds a caption to the table |
| colspan | Merges cells across columns |
| rowspan | Merges cells across rows |
| border | Adds a border to the table and cells |
| border-collapse | Merges adjacent borders into one |
| padding | Adds space inside the cell |
| border-spacing | Adds space between cells |
| text-align | Aligns cell text |
| nth-child() | Styles even or odd rows |

## **💡 Best Practices:**

✔️ Keep tables simple for better readability.  
 ✔️ Use colspan and rowspan for complex layouts.  
 ✔️ Make tables responsive for better mobile support.  
 ✔️ Use caption for accessibility and better understanding.  
 ✔️ Avoid using tables for page layout—use CSS Flexbox or Grid instead.

This version is more **organized**, **concise**, and **professional**. Let me know if you’d like to adjust or add anything!

## **HTML Lists**

HTML lists allow you to organize content in a structured format using bullets, numbers, or terms with definitions.

## **1. Unordered HTML Lists**

An unordered list starts with the <ul> tag.

* Each list item starts with the <li> tag.
* By default, list items are marked with **small black circles** (bullets).

✅ **Example:**

<ul>

<li>Coffee</li>

<li>Tea</li>

<li>Milk</li>

</ul>

✅ **Output:**

* Coffee
* Tea
* Milk

### **Set Custom List Markers**

Use the list-style-type property to change the marker style:

|  |  |  |
| --- | --- | --- |
| **Value** | **Description** | **Example** |
| disc | Filled circle (default) | list-style-type: disc; |
| circle | Open circle | list-style-type: circle; |
| square | Filled square | list-style-type: square; |
| none | No marker | list-style-type: none; |

✅ **Examples:**

1. **Disc (default):**

<ul style="list-style-type:disc;">

<li>Coffee</li>

<li>Tea</li>

<li>Milk</li>

</ul>

1. **Open Circle:**

<ul style="list-style-type:circle;">

<li>Coffee</li>

<li>Tea</li>

<li>Milk</li>

</ul>

1. **Square:**

<ul style="list-style-type:square;">

<li>Coffee</li>

<li>Tea</li>

<li>Milk</li>

</ul>

1. **No Marker:**

<ul style="list-style-type:none;">

<li>Coffee</li>

<li>Tea</li>

<li>Milk</li>

</ul>

## **2. Ordered HTML Lists**

An ordered list starts with the <ol> tag.

* Each list item starts with the <li> tag.
* By default, list items are numbered.

✅ **Example:**

<ol>

<li>Coffee</li>

<li>Tea</li>

<li>Milk</li>

</ol>

✅ **Output:**

1. Coffee
2. Tea
3. Milk

### **Custom Numbering Styles**

Use the type attribute to change the list numbering style:

|  |  |  |
| --- | --- | --- |
| **Type** | **Description** | **Example** |
| 1 | Numbers (default) | <ol type="1"> |
| A | Uppercase letters | <ol type="A"> |
| a | Lowercase letters | <ol type="a"> |
| I | Uppercase Roman numerals | <ol type="I"> |
| i | Lowercase Roman numerals | <ol type="i"> |

✅ **Examples:**

1. **Numbers (Default):**

<ol type="1">

<li>Coffee</li>

<li>Tea</li>

<li>Milk</li>

</ol>

1. **Uppercase Letters:**

<ol type="A">

<li>Coffee</li>

<li>Tea</li>

<li>Milk</li>

</ol>

1. **Lowercase Letters:**

<ol type="a">

<li>Coffee</li>

<li>Tea</li>

<li>Milk</li>

</ol>

1. **Uppercase Roman Numerals:**

<ol type="I">

<li>Coffee</li>

<li>Tea</li>

<li>Milk</li>

</ol>

1. **Lowercase Roman Numerals:**

<ol type="i">

<li>Coffee</li>

<li>Tea</li>

<li>Milk</li>

</ol>

### **Start Numbering at a Custom Value**

Use the start attribute to begin the numbering at a specific value:

<ol start="50">

<li>Coffee</li>

<li>Tea</li>

<li>Milk</li>

</ol>

✅ **Output:** 50. Coffee  
 51. Tea  
 52. Milk

## **3. Description Lists**

A description list allows you to define a list of terms and their descriptions.

* <dl> → Defines the description list
* <dt> → Defines the term
* <dd> → Defines the description

✅ **Example:**

<dl>

<dt>Coffee</dt>

<dd>- Black hot drink</dd>

<dt>Milk</dt>

<dd>- White cold drink</dd>

</dl>

✅ **Output:  
Coffee** – Black hot drink  
**Milk** – White cold drink

## **4. Nested Lists**

You can create nested lists (lists inside lists):

✅ **Example:**

<ul>

<li>Coffee</li>

<li>Tea

<ul>

<li>Black tea</li>

<li>Green tea</li>

</ul>

</li>

<li>Milk</li>

</ul>

✅ **Output:**

* Coffee
* Tea
* Black tea
* Green tea
* Milk

## **5. Horizontal Lists (Navigation Menus)**

Use float: left and display: block to create horizontal navigation menus:

✅ **Example:**

<style>

ul {

list-style-type: none;

margin: 0;

padding: 0;

overflow: hidden;

background-color: #333;

}

li {

float: left;

}

li a {

display: block;

color: white;

padding: 16px;

text-decoration: none;

}

li a:hover {

background-color: #111;

}

</style>

<ul>

<li><a href="#home">Home</a></li>

<li><a href="#news">News</a></li>

<li><a href="#contact">Contact</a></li>

<li><a href="#about">About</a></li>

</ul>

✅ **Output:** A responsive horizontal menu with hover effects.

## **✅ Summary**

|  |  |
| --- | --- |
| **Tag/Property** | **Description** |
| <ul> | Defines an unordered list |
| <ol> | Defines an ordered list |
| <li> | Defines a list item |
| <dl> | Defines a description list |
| <dt> | Defines a term in a description list |
| <dd> | Describes the term in a description list |
| list-style-type | Sets the type of marker for list items |
| type | Sets the numbering type for ordered lists |
| start | Sets the starting number for ordered lists |
| float | Aligns list items horizontally |

## **💡 Best Practices:**

✔️ Use <ul> for unordered content lists.  
 ✔️ Use <ol> for ordered steps or priority lists.  
 ✔️ Use <dl> for term and definition-style lists.  
 ✔️ Keep lists simple and readable.  
 ✔️ Use float and display for responsive and clean layouts.

This version is more **organized**, **concise**, and **professional**. Let me know if you’d like to adjust or add anything!

**HTML Block and Inline Elements**

### **Block-Level Elements**

* A block-level element **starts on a new line** and takes up the **full width** available.
* It stretches to the left and right edges of its container.
* Commonly used for structuring the layout of a webpage.

#### **Example:**

<div>

<h1>This is a block element</h1>

<p>This is another block element inside a div.</p>

</div>

#### **List of Block-Level Elements**

|  |  |
| --- | --- |
| **Tag** | **Description** |
| <address> | Defines contact information |
| <article> | Defines an article |
| <aside> | Defines content aside from the main content |
| <blockquote> | Defines a section quoted from another source |
| <canvas> | Used to draw graphics |
| <dd> | Describes a term in a description list |
| <div> | Defines a section in a document |
| <dl> | Defines a description list |
| <dt> | Defines a term in a description list |
| <fieldset> | Groups related elements in a form |
| <figcaption> | Defines a caption for a <figure> element |
| <figure> | Specifies self-contained content |
| <footer> | Defines a footer section |
| <form> | Defines an HTML form |
| <h1>–<h6> | Defines headings |
| <header> | Defines a header section |
| <hr> | Draws a horizontal line |
| <li> | Defines a list item |
| <main> | Defines the main content of a document |
| <nav> | Defines navigation links |
| <noscript> | Defines alternative content for users with JavaScript disabled |
| <ol> | Defines an ordered list |
| <p> | Defines a paragraph |
| <pre> | Defines preformatted text |
| <section> | Defines a section in a document |
| <table> | Defines a table |
| <tfoot> | Defines a footer in a table |
| <ul> | Defines an unordered list |
| <video> | Embeds video content |

### **Inline Elements**

* An inline element **does not start on a new line**.
* It only takes up as much width as necessary.
* Commonly used for formatting text and adding links or images.

#### **Example:**

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

#### **List of Inline Elements**

|  |  |
| --- | --- |
| **Tag** | **Description** |
| <a> | Defines a hyperlink |
| <abbr> | Defines an abbreviation |
| <acronym> | Defines an acronym |
| <b> | Defines bold text |
| <bdo> | Overrides the text direction |
| <big> | Defines large text (deprecated) |
| <br> | Inserts a line break |
| <button> | Defines a clickable button |
| <cite> | Defines the title of a work |
| <code> | Defines a piece of code |
| <dfn> | Defines a definition term |
| <em> | Defines emphasized text |
| <i> | Defines italic text |
| <img> | Embeds an image |
| <input> | Defines an input field |
| <kbd> | Defines keyboard input |
| <label> | Defines a label for an input element |
| <map> | Defines an image map |
| <object> | Embeds an external resource |
| <output> | Represents the result of a calculation |
| <q> | Defines a short quotation |
| <samp> | Defines sample output from a program |
| <script> | Embeds a client-side script |
| <select> | Defines a drop-down list |
| <small> | Defines smaller text |
| <span> | Defines an inline section |
| <strong> | Defines important text |
| <sub> | Defines subscript text |
| <sup> | Defines superscript text |
| <textarea> | Defines a multi-line input field |
| <time> | Defines a time or date |
| <tt> | Defines teletype text (deprecated) |
| <var> | Defines a variable |

### **The <div> Element**

* A **block-level** element used to group other HTML elements.
* Common attributes: style, class, id.
* Used with CSS to style content blocks.

#### **Example:**

<div style="background-color: black; color: white; padding: 20px;">

<h2>London</h2>

<p>London is the capital city of England.</p>

</div>

### **The <span> Element**

* An **inline** element used to style parts of text or document.
* Common attributes: style, class, id.
* Used with CSS to apply inline styling.

#### **Example:**

<p>My mother has <span style="color: blue; font-weight: bold;">blue</span> eyes and my father has <span style="color: green; font-weight: bold;">green</span> eyes.</p>

### **Key Differences Between <div> and <span>**

|  |  |  |
| --- | --- | --- |
| **Feature** | **<div>** | **<span>** |
| Display Type | Block-level | Inline |
| Purpose | Groups elements together | Styles specific text parts |
| Width Behavior | Takes full width available | Takes only the necessary width |

### **Chapter Summary**

✅ Block-level elements start on a new line and take up the full width.  
 ✅ Inline elements do not start on a new line and take up only as much width as necessary.  
 ✅ <div> is a block-level container used to group HTML elements.  
 ✅ <span> is an inline container used to style parts of text.

### **HTML Tags Summary**

|  |  |
| --- | --- |
| **Tag** | **Description** |
| <div> | Defines a block-level section |
| <span> | Defines an inline section |

**HTML Classes**

### **Definition**

* The class attribute is used to define a class for an HTML element.
* Multiple elements can share the same class to apply consistent styling or be manipulated using JavaScript.

### **Example**

In the example below, all <div> elements with the city class are styled equally based on the .city CSS definition:

<!DOCTYPE html>

<html>

<head>

<style>

.city {

background-color: tomato;

color: white;

border: 2px solid black;

margin: 20px;

padding: 20px;

}

</style>

</head>

<body>

<div class="city">

<h2>London</h2>

<p>London is the capital of England.</p>

</div>

<div class="city">

<h2>Paris</h2>

<p>Paris is the capital of France.</p>

</div>

<div class="city">

<h2>Tokyo</h2>

<p>Tokyo is the capital of Japan.</p>

</div>

</body>

</html>

### **Creating a Class**

* Use a **period (.)** followed by the class name in CSS.
* Define the CSS properties within curly braces {}.

#### **Syntax:**

.classname {

property: value;

}

### **Example:**

<style>

.city {

background-color: tomato;

color: white;

padding: 10px;

}

</style>

### **Using class with <span>**

The class attribute can also be used with inline elements like <span>.

<!DOCTYPE html>

<html>

<head>

<style>

.note {

font-size: 120%;

color: red;

}

</style>

</head>

<body>

<h1>My <span class="note">Important</span> Heading</h1>

<p>This is some <span class="note">important</span> text.</p>

</body>

</html>

### **Case Sensitivity**

* Class names are **case-sensitive**.
* city and City are treated as two different class names.

### **Multiple Classes**

* An element can belong to **multiple classes**.
* Separate multiple class names with a **space**.
* The element will be styled according to **all specified classes**.

#### **Example:**

In the example below, London is aligned to the center using the main class and styled using the city class:

<!DOCTYPE html>

<html>

<head>

<style>

.city {

background-color: tomato;

color: white;

padding: 10px;

}

.main {

text-align: center;

}

</style>

</head>

<body>

<h2>Multiple Classes</h2>

<p>Here, all three h2 elements belong to the "city" class. In addition, London also belongs to the "main" class, which center-aligns the text.</p>

<h2 class="city main">London</h2>

<h2 class="city">Paris</h2>

<h2 class="city">Tokyo</h2>

</body>

</html>

### **Sharing Styles Across Different Elements**

Different HTML elements (like <h2> and <p>) can share the same class and styles:

<h2 class="city">Paris</h2>

<p class="city">Paris is the capital of France</p>

### **Using class with JavaScript**

JavaScript can target elements with a specific class using the getElementsByClassName() method.

#### **Example:**

In this example, the myFunction() function hides all elements with the city class when called:

<script>

function myFunction() {

var x = document.getElementsByClassName("city");

for (var i = 0; i < x.length; i++) {

x[i].style.display = "none";

}

}

</script>

### **Key Points**

✅ The class attribute allows elements to share styles.  
 ✅ Classes are defined using a **period (.)** in CSS.  
 ✅ Multiple classes can be assigned to a single element.  
 ✅ JavaScript can manipulate elements with the same class using getElementsByClassName().  
 ✅ Class names are **case-sensitive**.

### **HTML Tags Summary**

|  |  |
| --- | --- |
| **Tag** | **Description** |
| <div> | Defines a section in a document (block-level) |
| <span> | Defines a section in a document (inline) |

**HTML ID**

### **Definition**

* The id attribute specifies a **unique identifier** for an HTML element.
* An id value **must be unique** within a document.
* The id is used to target an element for:
* **CSS styling**
* **JavaScript manipulation**
* **HTML bookmarks**

### **Creating an ID**

* Use a **hash (#)** followed by the id name in CSS.
* Define the CSS properties within {}.

#### **Syntax:**

#idname {

property: value;

}

### **Example:**

In this example, the #myHeader ID styles the <h1> element:

<!DOCTYPE html>

<html>

<head>

<style>

#myHeader {

background-color: lightblue;

color: black;

padding: 40px;

text-align: center;

}

</style>

</head>

<body>

<h1 id="myHeader">My Header</h1>

</body>

</html>

### **Case Sensitivity**

* The id value is **case-sensitive** (myHeader ≠ MyHeader).
* The id value:
* Must contain **at least one character**.
* Cannot contain **whitespaces** (spaces, tabs, etc.).

### **Difference Between id and class**

|  |  |  |
| --- | --- | --- |
| **Feature** | **id** | **class** |
| **Uniqueness** | Must be unique within a document | Can be used by multiple elements |
| **CSS Syntax** | Use # to define an id (#myHeader) | Use . to define a class (.city) |
| **JavaScript Access** | getElementById() | getElementsByClassName() |

#### **Example:**

An element with a unique id and multiple elements sharing the same class:

<!DOCTYPE html>

<html>

<head>

<style>

#myHeader {

background-color: lightblue;

color: black;

padding: 40px;

text-align: center;

}

.city {

background-color: tomato;

color: white;

padding: 10px;

}

</style>

</head>

<body>

<h2>Difference Between Class and ID</h2>

<p>A class name can be used by multiple HTML elements, while an id name must only be used by one HTML element within the page:</p>

<!-- Unique ID -->

<h1 id="myHeader">My Cities</h1>

<!-- Multiple elements with same class -->

<h2 class="city">London</h2>

<p>London is the capital of England.</p>

<h2 class="city">Paris</h2>

<p>Paris is the capital of France.</p>

<h2 class="city">Tokyo</h2>

<p>Tokyo is the capital of Japan.</p>

</body>

</html>

### **HTML Bookmarks Using id**

* id can be used to create **bookmarks** for navigating long pages.
* To create a bookmark:

1. Add an id to an element.
2. Create a link using #idname.

#### **Example:**

<h2 id="C4">Chapter 4</h2>

<a href="#C4">Jump to Chapter 4</a>

To link to a bookmark on **another page**:

<a href="page.html#C4">Jump to Chapter 4</a>

### **Complete Example:**

In this example, clicking the links will scroll to the corresponding chapter:

<!DOCTYPE html>

<html>

<body>

<p><a href="#C4">Jump to Chapter 4</a></p>

<p><a href="#C10">Jump to Chapter 10</a></p>

<h2>Chapter 1</h2>

<p>This chapter explains ba bla bla</p>

<h2>Chapter 2</h2>

<p>This chapter explains ba bla bla</p>

<h2>Chapter 3</h2>

<p>This chapter explains ba bla bla</p>

<h2 id="C4">Chapter 4</h2>

<p>This chapter explains ba bla bla</p>

<h2 id="C10">Chapter 10</h2>

<p>This chapter explains ba bla bla</p>

</body>

</html>

### **Using id with JavaScript**

JavaScript can target elements with an id using getElementById().

#### **Example:**

This example changes the content of the h1 element with id="myHeader":

<script>

function displayResult() {

document.getElementById("myHeader").innerHTML = "Have a nice day!";

}

</script>

### **Key Points**

✅ The id attribute defines a **unique identifier** for an HTML element.  
 ✅ The id value is case-sensitive and must be unique.  
 ✅ Use #idname in CSS to target an element with a specific id.  
 ✅ JavaScript uses getElementById() to select an element with a specific id.  
 ✅ id is useful for creating **bookmarks** and **direct links**.

### **HTML Tags Summary**

|  |  |
| --- | --- |
| **Tag** | **Description** |
| <div> | Defines a section in a document (block-level) |
| <span> | Defines a section in a document (inline) |
| <h1> – <h6> | Define headings (block-level) |
| <p> | Defines a paragraph (block-level) |

**HTML Iframes**

### **Definition**

* The <iframe> tag is used to display a **webpage within another webpage**.
* It creates an **inline frame** inside the current HTML document.

### **Syntax**

<iframe src="url" title="description"></iframe>

✅ src – URL of the embedded page.  
 ✅ title – Describes the iframe content (important for accessibility).

### **Example:**

Embed an external webpage:

<iframe src="https://www.example.com" title="Example Website"></iframe>

### **Size Adjustment**

Use width and height attributes to define the size of the iframe:

<iframe src="demo\_iframe.htm" height="200" width="300" title="Iframe Example"></iframe>

Alternatively, use **CSS** for more control:

<iframe src="demo\_iframe.htm" style="height:200px; width:300px;" title="Iframe Example"></iframe>

### **Remove the Border**

To remove the iframe border using **CSS**:

<iframe src="demo\_iframe.htm" style="border:none;" title="Iframe Example"></iframe>

To add or style the border:

<iframe src="demo\_iframe.htm" style="border:2px solid red;" title="Iframe Example"></iframe>

### **Targeting Links to an Iframe**

* An iframe can act as the **target** for a link using the name attribute.
* The target attribute in the link should match the name of the iframe.

#### **Example:**

The link opens in the iframe:

<iframe src="demo\_iframe.htm" name="iframe\_a" title="Iframe Example"></iframe>

<p><a href="https://www.w3schools.com" target="iframe\_a">Open W3Schools</a></p>

✅ When the target value matches the name attribute of an iframe, the link opens in that iframe.  
 ✅ The target attribute can also open a link in:

* \_blank – New window or tab
* \_self – Current window
* \_parent – Parent frame
* \_top – Topmost frame

### **Complete Example:**

This example shows how to create an iframe, adjust its size, remove the border, and use it as a link target:

<!DOCTYPE html>

<html>

<head>

<style>

iframe {

width: 400px;

height: 300px;

border: 2px solid blue;

}

</style>

</head>

<body>

<h2>Iframe Example</h2>

<iframe src="https://www.example.com" name="myIframe" title="Example Iframe"></iframe>

<p>

<a href="https://www.wikipedia.org" target="myIframe">Open Wikipedia</a>

</p>

</body>

</html>

### **Key Points**

✅ <iframe> embeds another webpage within the current document.  
 ✅ src specifies the URL of the embedded page.  
 ✅ title is important for accessibility.  
 ✅ Use height and width to set the size.  
 ✅ Use name to make an iframe the target for links.  
 ✅ border can be styled or removed using CSS.

**HTML JavaScript**

### **Definition**

JavaScript is used to make HTML documents **interactive** and **dynamic**.

### **Example:**

A button that shows the current date and time:

<!DOCTYPE html>

<html>

<body>

<h1>My First JavaScript</h1>

<button type="button" onclick="document.getElementById('demo').innerHTML = Date()">

Click me to display Date and Time.

</button>

<p id="demo"></p>

</body>

</html>

✅ onclick – Executes JavaScript when the button is clicked.  
 ✅ document.getElementById('demo').innerHTML – Selects the element and changes its content.

### **HTML <script> Tag**

* The <script> element allows JavaScript to be included in HTML.
* It can contain:
* **Inline JavaScript**
* **External JavaScript** (via the src attribute)

### **Example:**

Write text into an HTML element:

<h2>Use JavaScript to Change Text</h2>

<p>This example writes "Hello JavaScript!" into an HTML element with id="demo":</p>

<p id="demo"></p>

<script>

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

</script>

✅ innerHTML changes the HTML content of an element.

### **Changing HTML Style with JavaScript**

You can change the **CSS style** of an element using JavaScript:

#### **Example:**

Change font size, color, and background:

<h1>My First JavaScript</h1>

<p id="demo">JavaScript can change the style of an HTML element.</p>

<script>

function myFunction() {

document.getElementById("demo").style.fontSize = "25px";

document.getElementById("demo").style.color = "red";

document.getElementById("demo").style.backgroundColor = "yellow";

}

</script>

<button type="button" onclick="myFunction()">Click Me!</button>

✅ style.property – Modifies CSS properties directly.

### **Changing the src Attribute of an Image**

JavaScript can dynamically update the src attribute of an image:

#### **Example:**

Switch between two images:

<script>

function light(sw) {

var pic;

if (sw == 0) {

pic = "pic\_bulboff.gif"

} else {

pic = "pic\_bulbon.gif"

}

document.getElementById('myImage').src = pic;

}

</script>

<img id="myImage" src="pic\_bulboff.gif" width="100" height="180">

<p>

<button type="button" onclick="light(1)">Light On</button>

<button type="button" onclick="light(0)">Light Off</button>

</p>

✅ src – Changes the source of the image.

### **HTML <noscript> Tag**

* Displays alternative content if JavaScript is **disabled** or **unsupported**.

#### **Example:**

<p id="demo"></p>

<script>

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

</script>

<noscript>Sorry, your browser does not support JavaScript!</noscript>

<p>A browser without support for JavaScript will show the text written inside the noscript element.</p>

✅ <noscript> content is shown only if JavaScript is disabled.

### **Complete Example**

Combining JavaScript to modify HTML, CSS, and handle events:

<!DOCTYPE html>

<html>

<head>

<style>

#myHeader {

color: blue;

text-align: center;

}

</style>

</head>

<body>

<h1 id="myHeader">My First JavaScript</h1>

<button type="button" onclick="showDate()">Show Date</button>

<p id="demo"></p>

<script>

function showDate() {

document.getElementById("demo").innerHTML = Date();

}

function changeStyle() {

document.getElementById("myHeader").style.color = "red";

}

</script>

<button type="button" onclick="changeStyle()">Change Header Color</button>

</body>

</html>

### **Key Points**

✅ JavaScript is included using the <script> tag.  
 ✅ document.getElementById() is used to access HTML elements.  
 ✅ innerHTML modifies HTML content.  
 ✅ style.property modifies CSS properties.  
 ✅ src changes the source of media elements.  
 ✅ <noscript> shows content when JavaScript is disabled.

**HTML File Path**

### **Definition**

A **file path** describes the location of a file in a website’s folder structure.  
 👉 File paths are used when linking to:

* Web pages
* Images
* Style sheets
* JavaScripts

## **Types of File Paths**

### **1. Absolute File Path**

* Specifies the **full URL** to a file.
* Always starts with https:// or http://.

✅ **Example:**

<img src="https://www.w3schools.com/images/picture.jpg" alt="Mountain">

➡️ Links directly to an external website.

### **2. Relative File Path**

* Points to a file **relative to the current page**.
* Commonly used in web development for better portability.

✅ **Examples:**

|  |  |
| --- | --- |
| **Syntax** | **Description** |
| <img src="picture.jpg"> | File is in the **same folder** as the current page. |
| <img src="images/picture.jpg"> | File is in the **"images" folder** in the current folder. |
| <img src="/images/picture.jpg"> | File is in the **"images" folder** at the root of the current web. |
| <img src="../picture.jpg"> | File is in the folder **one level up** from the current folder. |

### **Example Folder Structure:**

/website

├── index.html

├── images

│ └── picture.jpg

├── css

│ └── styles.css

#### **✅ Example Code:**

1. **File in the same folder:**

<img src="picture.jpg" alt="Mountain">

1. **File in a subfolder:**

<img src="images/picture.jpg" alt="Mountain">

1. **File in the root directory:**

<img src="/images/picture.jpg" alt="Mountain">

1. **File one level up:**

<img src="../picture.jpg" alt="Mountain">

### **Best Practices**

✅ Use **relative paths** whenever possible for better portability.  
 ✅ Relative paths ensure that links work on both **localhost** and **public domains**.  
 ✅ Use absolute paths only when linking to **external resources**.

### **Complete Example**

<!DOCTYPE html>

<html>

<head>

<title>File Path Example</title>

</head>

<body>

<h2>Example of File Paths</h2>

<!-- File in the same folder -->

<img src="picture.jpg" alt="Mountain">

<!-- File in a subfolder -->

<img src="images/picture.jpg" alt="Mountain">

<!-- File in the root folder -->

<img src="/images/picture.jpg" alt="Mountain">

<!-- File one level up -->

<img src="../picture.jpg" alt="Mountain">

</body>

</html>

### **✅ Summary**

|  |  |  |
| --- | --- | --- |
| **Type** | **Description** | **Example** |
| **Absolute Path** | Full URL to the file | https://example.com/images/pic.jpg |
| **Relative Path** | Relative to the current file's location | "images/pic.jpg" |
| **Root Relative Path** | Starts from the root of the project | "/images/pic.jpg" |
| **Parent Folder Path** | Points to a file in the parent directory | "../pic.jpg" |

## **HTML <head> Element**

### **Definition**

The <head> element is a container for **metadata** (data about the HTML document).

* Placed between the <html> tag and the <body> tag.
* Metadata includes information like the document title, character set, styles, scripts, and more.
* Metadata is **not displayed** on the webpage but is used by browsers, search engines, and other web services.

## **HTML <head> Elements**

|  |  |
| --- | --- |
| **Tag** | **Description** |
| <title> | Defines the title of the document |
| <style> | Defines style information for the document |
| <link> | Links to an external resource (usually a stylesheet) |
| <meta> | Provides metadata about the document |
| <script> | Embeds or links to client-side JavaScript |
| <base> | Sets the base URL and target for all relative links |

## **1. <title> – Document Title**

* Defines the **title** shown in the browser tab or search engine results.
* Improves **SEO** (Search Engine Optimization).
* Required in all HTML documents.

✅ **Example:**

<head>

<title>My Web Page</title>

</head>

👉 **Purpose:**

* Title in browser toolbar
* Title in search engine results
* Title in bookmarks

## **2. <style> – Internal CSS**

* Used to define CSS styles directly in the HTML document.
* Placed inside the <head> element.

✅ **Example:**

<head>

<style>

body { background-color: powderblue; }

h1 { color: red; }

p { color: blue; }

</style>

</head>

👉 **Purpose:**

* Define styling for a single HTML page.
* Overwritten by external stylesheets (if present).

## **3. <link> – External CSS**

* Links an external CSS file to the HTML document.
* rel="stylesheet" specifies that the linked file is a stylesheet.

✅ **Example:**

<head>

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

</head>

👉 **Purpose:**

* Keep CSS separate from HTML.
* Easier to maintain and update styling.

## **4. <meta> – Metadata**

* Provides information about the document (charset, viewport, description, author, etc.).
* Not displayed on the page but used by browsers and search engines.

✅ **Examples:**

|  |  |
| --- | --- |
| **Purpose** | **Example** |
| **Set character set** | <meta charset="UTF-8"> |
| **Define keywords** | <meta name="keywords" content="HTML, CSS, JavaScript"> |
| **Set page description** | <meta name="description" content="Free Web tutorials"> |
| **Set author** | <meta name="author" content="John Doe"> |
| **Set viewport for responsive design** | <meta name="viewport" content="width=device-width, initial-scale=1.0"> |
| **Auto-refresh every 30 seconds** | <meta http-equiv="refresh" content="30"> |

👉 **Purpose:**

* Search engine optimization (SEO)
* Mobile responsiveness
* Automatic refresh or redirection

## **5. <script> – JavaScript Inclusion**

* Embeds or links JavaScript files.
* Can be placed in <head> or at the end of <body>.
* Use async or defer for performance optimization.

✅ **Example:**

<head>

<script>

function showMessage() {

alert("Hello, JavaScript!");

}

</script>

</head>

<body>

<button onclick="showMessage()">Click Me</button>

</body>

👉 **Purpose:**

* Add interactivity to the page
* Perform dynamic content changes

## **6. <base> – Base URL for Links**

* Specifies a **base URL** for all relative links on the page.
* Can define a **default target** for links (\_self, \_blank, \_parent, \_top).

✅ **Example:**

<head>

<base href="https://www.w3schools.com/" target="\_blank">

</head>

<body>

<img src="images/stickman.gif" alt="Stickman">

<a href="tags/tag\_base.asp">HTML Base Tag</a>

</body>

👉 **Purpose:**

* If <base> is defined, the image path will become:  
  https://www.w3schools.com/images/stickman.gif
* The link target will open in a **new tab** (\_blank).

## **Complete Example**

<!DOCTYPE html>

<html lang="en">

<head>

<!-- Title -->

<title>My Web Page</title>

<!-- Meta Tags -->

<meta charset="UTF-8">

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

<meta name="keywords" content="HTML, CSS, JavaScript">

<meta name="description" content="Free Web tutorials">

<meta name="author" content="John Doe">

<!-- CSS -->

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

<!-- Base URL -->

<base href="https://www.example.com/" target="\_blank">

<!-- Internal CSS -->

<style>

body {

background-color: lightgray;

font-family: Arial, sans-serif;

}

h1 {

color: navy;

}

</style>

<!-- JavaScript -->

<script>

function showMessage() {

alert("Hello, JavaScript!");

}

</script>

</head>

<body>

<h1>Welcome to My Web Page</h1>

<p>This is a sample webpage.</p>

<button onclick="showMessage()">Click Me</button>

</body>

</html>

## **✅ Summary**

|  |  |  |
| --- | --- | --- |
| **Element** | **Purpose** | **Example** |
| <title> | Sets the document title | <title>My Page</title> |
| <style> | Internal CSS styles | <style>body {background-color: gray;}</style> |
| <link> | Links external CSS | <link rel="stylesheet" href="styles.css"> |
| <meta> | Provides metadata | <meta name="description" content="Web tutorials"> |
| <script> | Embeds or links JavaScript | <script src="script.js"></script> |
| <base> | Sets base URL and target for links | <base href="https://example.com/"> |

## **HTML Layout**

HTML layout defines the structure and organization of a webpage using semantic elements and CSS techniques.

## **HTML Semantic Elements**

HTML5 introduced **semantic elements** to define different parts of a web page, improving **accessibility** and **SEO**.

|  |  |
| --- | --- |
| **Element** | **Description** |
| <header> | Defines a header for a document or section |
| <nav> | Defines a navigation bar or set of links |
| <section> | Defines a section of related content |
| <article> | Defines an independent, self-contained piece of content |
| <aside> | Defines side content (e.g., sidebar) |
| <footer> | Defines a footer for a document or section |
| <details> | Defines additional information that can be toggled open or closed |
| <summary> | Defines a heading for the <details> element |

### **✅ Example of HTML Semantic Layout**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>HTML Layout Example</title>

<style>

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

}

header, footer {

background-color: #333;

color: white;

text-align: center;

padding: 1em;

}

nav {

background-color: #555;

padding: 0.5em;

text-align: center;

}

nav a {

color: white;

padding: 0.5em;

text-decoration: none;

display: inline-block;

}

section {

padding: 20px;

}

aside {

background-color: #f4f4f4;

padding: 10px;

margin-top: 10px;

}

</style>

</head>

<body>

<header>

<h1>My Website</h1>

</header>

<nav>

<a href="#home">Home</a>

<a href="#about">About</a>

<a href="#contact">Contact</a>

</nav>

<section>

<h2>Main Content</h2>

<p>This is the main content of the page.</p>

</section>

<aside>

<h3>Sidebar</h3>

<p>This is some additional content.</p>

</aside>

<footer>

<p>&copy; 2025 My Website. All rights reserved.</p>

</footer>

</body>

</html>

## **Techniques for Creating HTML Layouts**

HTML layouts are structured using **CSS** techniques:

### **1. CSS Frameworks**

* **Bootstrap** and **W3.CSS** provide ready-made styles and responsive grids.
* Easy to use, responsive, and consistent styling.

✅ **Example (Bootstrap):**

<!-- Bootstrap CDN -->

<link href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">

<div class="container">

<div class="row">

<div class="col-sm-4">Column 1</div>

<div class="col-sm-4">Column 2</div>

<div class="col-sm-4">Column 3</div>

</div>

</div>

### **2. CSS Float Layout**

* Uses the float and clear properties to create layouts.
* **Outdated** and replaced by Flexbox and Grid.

✅ **Example:**

<div style="float: left; width: 50%; background-color: lightgray;">

Left column

</div>

<div style="float: left; width: 50%; background-color: lightblue;">

Right column

</div>

<div style="clear: both;"></div>

👉 **Problem:** Tied to document flow, causing unpredictable behavior when resizing.

### **3. CSS Flexbox**

* One-dimensional layout (row or column).
* Useful for responsive designs and flexible alignment.

✅ **Example:**

<div style="display: flex;">

<div style="flex: 1; background-color: lightgray; padding: 20px;">Flex Item 1</div>

<div style="flex: 2; background-color: lightblue; padding: 20px;">Flex Item 2</div>

</div>

👉 **Best for:**

* Aligning items in a row or column.
* Equal or proportional sizing.

### **4. CSS Grid**

* Two-dimensional layout (rows and columns).
* Best for complex layouts.

✅ **Example:**

<div style="

display: grid;

grid-template-columns: 1fr 2fr;

gap: 10px;

">

<div style="background-color: lightgray; padding: 20px;">Grid Item 1</div>

<div style="background-color: lightblue; padding: 20px;">Grid Item 2</div>

</div>

👉 **Best for:**

* Complex designs.
* Aligning items along both axes (row and column).

## **Complete Example (Flexbox + Grid + Semantic Elements)**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Complete HTML Layout Example</title>

<style>

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

box-sizing: border-box;

}

header, footer {

background-color: #333;

color: white;

text-align: center;

padding: 1rem;

}

nav {

display: flex;

justify-content: center;

background-color: #444;

padding: 0.5rem;

}

nav a {

color: white;

padding: 0.5rem;

text-decoration: none;

}

.container {

display: grid;

grid-template-columns: 3fr 1fr;

gap: 20px;

padding: 20px;

}

article {

background-color: #f4f4f4;

padding: 1rem;

border-radius: 5px;

}

aside {

background-color: #eee;

padding: 1rem;

border-radius: 5px;

}

</style>

</head>

<body>

<header>

<h1>My Website</h1>

</header>

<nav>

<a href="#home">Home</a>

<a href="#about">About</a>

<a href="#contact">Contact</a>

</nav>

<div class="container">

<article>

<h2>Main Content</h2>

<p>This is the main content area where most of the information will be displayed.</p>

<details>

<summary>More Info</summary>

<p>This is additional information that can be hidden or shown on demand.</p>

</details>

</article>

<aside>

<h3>Sidebar</h3>

<p>This is a sidebar for related content or links.</p>

</aside>

</div>

<footer>

<p>&copy; 2025 My Website. All Rights Reserved.</p>

</footer>

</body>

</html>

## **✅ Summary**

|  |  |  |
| --- | --- | --- |
| **Technique** | **Description** | **Use Case** |
| **CSS Framework** | Predefined styles and responsive grid | Fast prototyping (Bootstrap, W3.CSS) |
| **CSS Float** | Old method using float and clear | Simple layouts, avoid if possible |
| **CSS Flexbox** | One-dimensional (row or column) | Flexible alignment and spacing |
| **CSS Grid** | Two-dimensional (rows and columns) | Complex layouts and structured designs |
| **HTML Semantic Elements** | Improves SEO and accessibility | Logical page structure |

## **HTML Responsive Design**

Responsive web design ensures that a webpage **adjusts automatically** to look good on **all devices** (desktops, tablets, and smartphones).

## **✅ Key Techniques for Responsive Design**

### **1. Viewport Settings**

Use the <meta> viewport tag to make sure the page scales correctly on different screen sizes.

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

* width=device-width – Sets the page width to the screen width.
* initial-scale=1.0 – Sets the initial zoom level to 100%.

### **2. Responsive Images**

To make images scale automatically based on the screen size, use **CSS width** or **max-width**.

✅ **Using width: 100% for full scaling:**

<img src="img\_girl.jpg" style="width:100%;">

✅ **Using max-width to prevent upscaling:**

<img src="img\_girl.jpg" style="max-width:100%; height:auto;">

✅ **Using <picture> to display different images based on screen size:**

<picture>

<source srcset="img\_small.jpg" media="(max-width: 600px)">

<source srcset="img\_medium.jpg" media="(max-width: 1200px)">

<source srcset="img\_large.jpg">

<img src="img\_large.jpg" alt="Example">

</picture>

### **3. Responsive Text (Viewport Units)**

Use vw (viewport width) for responsive font sizes.

* 1vw = 1% of the viewport width.

✅ **Example:**

<h1 style="font-size: 10vw;">Responsive Text</h1>

👉 If the screen is 500px wide, 10vw will be 50px.

### **4. Media Queries**

Use media queries to apply different styles based on screen width.

✅ **Example:**

<style>

body {

font-family: Arial, sans-serif;

}

.container {

display: flex;

gap: 10px;

}

.box {

flex: 1;

padding: 20px;

background-color: lightgray;

}

/\* Change layout when viewport is smaller than 600px \*/

@media screen and (max-width: 600px) {

.container {

flex-direction: column;

}

}

</style>

<div class="container">

<div class="box">Box 1</div>

<div class="box">Box 2</div>

<div class="box">Box 3</div>

</div>

👉 When the screen width is **600px** or less, the .container becomes a **column** layout.

### **5. CSS Flexbox for Responsive Layouts**

* Flexbox aligns elements horizontally or vertically.
* Automatically resizes based on screen size.

✅ **Example:**

<div style="display: flex; gap: 10px; flex-wrap: wrap;">

<div style="flex: 1; min-width: 150px; background-color: lightblue; padding: 20px;">Item 1</div>

<div style="flex: 1; min-width: 150px; background-color: lightcoral; padding: 20px;">Item 2</div>

<div style="flex: 1; min-width: 150px; background-color: lightgreen; padding: 20px;">Item 3</div>

</div>

👉 flex-wrap: wrap ensures that the elements will wrap to a new row if the screen is too small.

### **6. CSS Grid for Responsive Layouts**

* Grid allows defining rows and columns.
* Elements adjust based on screen size.

✅ **Example:**

<div style="

display: grid;

grid-template-columns: repeat(auto-fit, minmax(150px, 1fr));

gap: 10px;

">

<div style="background-color: lightgray; padding: 20px;">Grid 1</div>

<div style="background-color: lightblue; padding: 20px;">Grid 2</div>

<div style="background-color: lightcoral; padding: 20px;">Grid 3</div>

</div>

👉 repeat(auto-fit, minmax(150px, 1fr)) makes the grid items auto-adjust.

## **✅ CSS Frameworks for Responsive Design**

### **1. W3.CSS**

* Lightweight, fast, and mobile-first.
* No dependencies on JavaScript libraries.

✅ **Example:**

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">

</head>

<body>

<div class="w3-container w3-teal">

<h1>Responsive Example</h1>

</div>

<div class="w3-row-padding">

<div class="w3-third">

<div class="w3-card">

<p>Box 1</p>

</div>

</div>

<div class="w3-third">

<div class="w3-card">

<p>Box 2</p>

</div>

</div>

<div class="w3-third">

<div class="w3-card">

<p>Box 3</p>

</div>

</div>

</div>

</body>

</html>

### **2. Bootstrap**

* Most popular CSS framework.
* Includes a 12-column grid and responsive components.

✅ **Example:**

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1">

<!-- Bootstrap CDN -->

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

</head>

<body>

<div class="container">

<div class="row">

<div class="col-md-4">

<div class="border p-3">Box 1</div>

</div>

<div class="col-md-4">

<div class="border p-3">Box 2</div>

</div>

<div class="col-md-4">

<div class="border p-3">Box 3</div>

</div>

</div>

</div>

</body>

</html>

👉 col-md-4 creates a responsive grid where each column takes **4 out of 12 parts**.

## **✅ Complete Responsive Example (Flexbox + Grid + Media Queries)**

<!DOCTYPE html>

<html lang="en">

<head>

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

<title>Complete Responsive Example</title>

<style>

body {

font-family: Arial, sans-serif;

margin: 0;

}

.header {

background-color: #333;

color: white;

padding: 20px;

text-align: center;

}

.container {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(150px, 1fr));

gap: 10px;

padding: 20px;

}

.box {

padding: 20px;

background-color: lightgray;

text-align: center;

border-radius: 5px;

}

@media (max-width: 600px) {

.header {

font-size: 18px;

}

}

</style>

</head>

<body>

<div class="header">

<h1>My Responsive Site</h1>

</div>

<div class="container">

<div class="box">Box 1</div>

<div class="box">Box 2</div>

<div class="box">Box 3</div>

</div>

</body>

</html>

## **✅ Summary**

|  |  |
| --- | --- |
| **Technique** | **Use Case** |
| **Flexbox** | One-dimensional, for flexible rows or columns |
| **Grid** | Two-dimensional, for complex layouts |
| **Media Queries** | Define different styles for different screen sizes |
| **Viewport Units** | Scale elements based on screen size |

## **HTML Computer Code Elements**

HTML provides special elements to define **computer code** and **user input**. These elements are displayed using the **browser's default monospace font** to distinguish them from regular text.

## **✅ 1. <code> – For Code Snippets**

* The <code> element is used to define a piece of code.
* Content inside it is shown in a **monospace font**.

✅ **Example:**

<p>JavaScript code example:</p>

<code>

x = 5;

y = 6;

z = x + y;

</code>

👉 Output:

x = 5;  
 y = 6;  
 z = x + y;

## **✅ 2. <pre> + <code> – For Preserving Whitespace**

* <pre> preserves **whitespace**, **line breaks**, and **indentation**.
* <code> inside <pre> formats it as code.

✅ **Example:**

<pre>

<code>

x = 5;

y = 6;

z = x + y;

</code>

</pre>

👉 Output:

x = 5;

y = 6;

z = x + y;

## **✅ 3. <kbd> – For Keyboard Input**

* The <kbd> element defines **keyboard input**.
* It's displayed in a monospace font.

✅ **Example:**

<p>Press <kbd>Ctrl + S</kbd> to save the document.</p>

👉 Output:

Press **Ctrl + S** to save the document.

## **✅ 4. <samp> – For Program Output**

* The <samp> element defines **sample output** from a program.
* It's displayed in a monospace font.

✅ **Example:**

<p>Message from the system:</p>

<p><samp>File not found.<br>Press F1 to continue.</samp></p>

👉 Output:

Message from the system:  
**File not found.  
Press F1 to continue.**

## **✅ 5. <var> – For Variables**

* The <var> element defines **variables** in programming or math.
* Content is usually shown in **italic**.

✅ **Example:**

<p>The area of a triangle is: 1/2 × <var>b</var> × <var>h</var></p>

👉 Output:

The area of a triangle is: 1/2 × *b* × *h*

## **✅ 6. Combined Example**

<!DOCTYPE html>

<html>

<head>

<title>HTML Code Elements</title>

</head>

<body>

<h2>HTML Code Elements</h2>

<p>Example of JavaScript code:</p>

<pre>

<code>

function add(a, b) {

return a + b;

}

</code>

</pre>

<p>Save the document by pressing <kbd>Ctrl + S</kbd></p>

<p>Program output:</p>

<p><samp>Operation successful</samp></p>

<p>Formula for a circle's area: π × <var>r</var>²</p>

</body>

</html>

## **✅ Summary**

|  |  |  |  |
| --- | --- | --- | --- |
| **Element** | **Purpose** | **Example** | **Display Style** |
| <code> | Defines inline code | <code>x = 5;</code> | Monospace font |
| <pre> + <code> | Preserves whitespace for code blocks | <pre><code>...</code></pre> | Monospace + Line breaks |
| <kbd> | Keyboard input | <kbd>Ctrl + S</kbd> | Monospace font |
| <samp> | Sample output from a program | <samp>File not found</samp> | Monospace font |
| <var> | Variables | <var>b</var> | Italics |

## **HTML Semantic Elements**

HTML **semantic elements** clearly describe the **meaning** of the content to both the **browser** and the **developer**.

### **✅ 1. What Are Semantic Elements?**

* **Semantic elements**: Elements that convey **meaning** and **structure** to the content.
* **Non-semantic elements**: <div> and <span> — They provide no meaningful information about the content.

|  |  |  |
| --- | --- | --- |
| **Type** | **Example** | **Meaning** |
| **Semantic Elements** | <article>, <header>, <footer> | Describe the content |
| **Non-Semantic Elements** | <div>, <span> | Provide no meaning or structure |

## **✅ 2. Common Semantic Elements**

### **(a) <section> – Grouping Thematic Content**

* Represents a **section** of content.
* Can include headings (<h1>–<h6>) and other related elements.

✅ **Example:**

<section>

<h1>About WWF</h1>

<p>The World Wide Fund for Nature (WWF) is an international organization working on environmental conservation.</p>

</section>

### **(b) <article> – Independent Self-Contained Content**

* Represents an independent piece of content (e.g., blog post, news article).
* Can be distributed or reused independently.

✅ **Example:**

<article>

<h2>Google Chrome</h2>

<p>Google Chrome is a web browser developed by Google, released in 2008.</p>

</article>

### **(c) <header> – Introductory Content or Navigation**

* Contains introductory content like a **logo**, **headings**, or **navigation links**.
* Can be used **multiple times** in a document.

✅ **Example:**

<header>

<h1>My Website</h1>

<p>Welcome to my site!</p>

</header>

### **(d) <footer> – Footer for a Document or Section**

* Contains metadata about the document like:
* Author information
* Copyright info
* Related links

✅ **Example:**

<footer>

<p>Author: John Doe</p>

<p>&copy; 2025 My Website</p>

</footer>

### **(e) <nav> – Navigation Links**

* Represents a block of **navigation links**.
* Helps assistive technologies identify the **main navigation**.

✅ **Example:**

<nav>

<a href="/home">Home</a> |

<a href="/about">About</a> |

<a href="/contact">Contact</a>

</nav>

### **(f) <aside> – Content Aside from the Main Content**

* Represents **content that is related** but not essential to the main content (e.g., sidebar).

✅ **Example:**

<aside>

<h4>Related News</h4>

<p>Check out the latest tech news!</p>

</aside>

### **(g) <figure> and <figcaption> – Grouping Media and Captions**

* <figure> – Represents self-contained content like images, diagrams, or charts.
* <figcaption> – Provides a caption for a <figure> element.

✅ **Example:**

<figure>

<img src="nature.jpg" alt="Nature">

<figcaption>Beautiful Nature</figcaption>

</figure>

### **(h) <details> and <summary> – Expandable Content**

* <details> – Creates a collapsible block of content.
* <summary> – Defines a visible heading for <details>.

✅ **Example:**

<details>

<summary>More Info</summary>

<p>This is additional information that can be revealed or hidden.</p>

</details>

### **(i) <mark> – Highlighted Text**

* Highlights text for reference or attention.

✅ **Example:**

<p>Don't forget to <mark>submit the form</mark> by Friday.</p>

### **(j) <time> – Date and Time**

* Represents a date, time, or duration.
* Can help search engines and browsers understand time-related content.

✅ **Example:**

<p>The event is scheduled for <time datetime="2025-04-15">April 15, 2025</time>.</p>

## **✅ 3. Example: Complete Semantic Layout**

<!DOCTYPE html>

<html>

<head>

<title>HTML Semantic Example</title>

</head>

<body>

<header>

<h1>My Blog</h1>

<nav>

<a href="#home">Home</a>

<a href="#about">About</a>

<a href="#contact">Contact</a>

</nav>

</header>

<main>

<article>

<h2>Semantic HTML</h2>

<p>Using semantic elements improves SEO and accessibility.</p>

</article>

<aside>

<h4>More Articles</h4>

<p>Check out related articles.</p>

</aside>

</main>

<footer>

<p>&copy; 2025 My Blog. All rights reserved.</p>

</footer>

</body>

</html>

## **✅ 4. Summary**

|  |  |  |
| --- | --- | --- |
| **Tag** | **Purpose** | **Usage Example** |
| <section> | Thematic grouping of content | Group related content |
| <article> | Independent, self-contained content | Blog post, news article |
| <header> | Introductory content | Logo, navigation |
| <footer> | Footer section | Author info, copyright |
| <nav> | Navigation links | Menu or navigation bar |
| <aside> | Content aside from the main content | Sidebar |
| <figure> | Grouping media and captions | Image with caption |
| <figcaption> | Caption for <figure> | Caption for an image |
| <details> | Expandable content | Collapsible content |
| <summary> | Heading for <details> | Summary for expandable section |
| <mark> | Highlighted text | Important or relevant text |
| <time> | Date and time | Event date |

## **🚀 Key Takeaways**

✔️ Use **semantic elements** to improve SEO and accessibility.  
 ✔️ Semantic elements provide **clear structure** and **meaning** to content.  
 ✔️ They help **screen readers** and **search engines** better understand the document.

## **HTML Style Guide and Coding Conventions**

Following a consistent HTML style guide improves **readability**, **maintainability**, and helps avoid common issues.

## **✅ 1. Basic Structure**

✔️ Always declare the DOCTYPE at the beginning of the document:

<!DOCTYPE html>

✔️ Include the <html> and <body> tags even if browsers can infer them:

<!DOCTYPE html>

<html lang="en-us">

<head>

<meta charset="UTF-8">

<title>Page Title</title>

</head>

<body>

<h1>This is a heading</h1>

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

</body>

</html>

✔️ **Set language and encoding** to improve search engine ranking and avoid display issues:

<html lang="en-us">

<meta charset="UTF-8">

✔️ **Set the viewport** for responsive design:

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

## **✅ 2. Syntax Rules**

### **🔹 (a) Use Lowercase for Elements and Attributes**

✅ Good:

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

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

❌ Bad:

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

<A HREF="https://example.com">Example</A>

### **🔹 (b) Close All HTML Elements**

✅ Good:

<section>

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

</section>

❌ Bad:

<section>

<p>This is a paragraph.

### **🔹 (c) Quote All Attribute Values**

✅ Good:

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

❌ Bad:

<a href=https://example.com>Visit</a>

### **🔹 (d) Avoid Spaces Around the = Sign**

✅ Good:

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

❌ Bad:

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

### **🔹 (e) Use Lowercase File Names**

✅ Good:

index.html

styles.css

script.js

❌ Bad:

Index.html

Styles.CSS

Script.JS

## **✅ 3. Indentation and Spacing**

✔️ Use **2 spaces** for indentation (avoid tabs).  
 ✔️ Avoid trailing spaces and blank lines.  
 ✔️ Add blank lines between logical code blocks.

✅ Example:

<body>

<h1>Title</h1>

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

<ul>

<li>Item 1</li>

<li>Item 2</li>

</ul>

</body>

## **✅ 4. Tables and Lists**

### **🔹 (a) Tables**

✔️ Align table tags consistently and keep indentation clean.  
 ✔️ Keep table headers (<th>) and data cells (<td>) aligned.

✅ Example:

<table>

<tr>

<th>Name</th>

<th>Age</th>

</tr>

<tr>

<td>John</td>

<td>25</td>

</tr>

</table>

### **🔹 (b) Lists**

✔️ Use proper indentation.  
 ✔️ Keep list items aligned.

✅ Example:

<ul>

<li>Apple</li>

<li>Banana</li>

<li>Cherry</li>

</ul>

## **✅ 5. Comments**

✔️ Use single-line comments for short notes:

<!-- This is a comment -->

✔️ Use multi-line comments for longer explanations:

<!--

This is a multi-line comment.

It can span over multiple lines.

-->

## **✅ 6. CSS Rules**

✔️ Link external CSS files without the type attribute (not required):

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

✔️ Short rules can be inline:

p.intro { font-family: Arial; font-size: 16px; }

✔️ Long rules should be split into multiple lines:

body {

background-color: #f4f4f4;

font-family: Arial, sans-serif;

font-size: 16px;

}

**✅ CSS Best Practices:**

* Opening bracket { on the same line as the selector.
* One space before {.
* Indentation with **2 spaces**.
* Always end with a ;.
* Quotes only around values with spaces.

## **✅ 7. JavaScript Rules**

✔️ Load external JS files at the bottom of the page (for faster rendering):

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

✔️ Case sensitivity matters in JavaScript:  
 ✅ Good:

document.getElementById("demo").innerHTML = "Hello";

❌ Bad:

document.getElementById("Demo").innerHTML = "Hello";

## **✅ 8. File Naming Conventions**

|  |  |  |
| --- | --- | --- |
| **Type** | **Example** | **Notes** |
| **HTML** | index.html | Use lowercase; .html preferred over .htm |
| **CSS** | style.css | Use lowercase; .css extension |
| **JavaScript** | script.js | Use lowercase; .js extension |
| **Images** | logo.png | Use lowercase; avoid spaces (logo image.png → logo-image.png) |

## **✅ 9. Title and Meta Data**

✔️ **Title:** Keep it descriptive and relevant.  
 ✔️ **Meta charset:** Define early in the document.  
 ✔️ **Meta viewport:** Ensure responsive behavior.

✅ Example:

<head>

<meta charset="UTF-8">

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

<title>My Website</title>

</head>

## **✅ 10. Closing Empty Elements**

✔️ HTML5 allows self-closing tags or unclosed tags, but it's better to close them for consistency:  
 ✅ Good:

<meta charset="UTF-8">

✅ XHTML:

<meta charset="UTF-8" />

## **✅ 11. Good Example of a Clean HTML Document**

<!DOCTYPE html>

<html lang="en-us">

<head>

<meta charset="UTF-8">

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

<title>HTML Style Guide</title>

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

</head>

<body>

<header>

<h1>My Website</h1>

<nav>

<a href="#home">Home</a>

<a href="#about">About</a>

<a href="#contact">Contact</a>

</nav>

</header>

<main>

<section>

<h2>About</h2>

<p>This is a sample website to demonstrate HTML styling rules.</p>

</section>

<aside>

<h3>Side Note</h3>

<p>Additional information goes here.</p>

</aside>

</main>

<footer>

<p>&copy; 2025 My Website</p>

</footer>

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

</body>

</html>

## **✅ 12. Summary**

|  |  |
| --- | --- |
| **✅ Rule** | **✅ Example** |
| Use lowercase for tags | <p> not <P> |
| Close all elements | <div></div> |
| Quote attribute values | href="example.com" |
| No space around = | rel="stylesheet" |
| Lowercase file names | index.html, not Index.html |
| Indentation | 2 spaces |
| Title and metadata | Always present |
| Consistent CSS | Simple and readable |

## 🚀 Following these guidelines ensures cleaner, more professional HTML code!

## **HTML Entities**

HTML entities allow you to display **reserved characters**, **special symbols**, and **invisible characters** in HTML without causing rendering issues.

## **✅ 1. Basic Syntax**

An HTML entity can be written in two ways:

1. **Entity name:** &entity\_name;
2. **Entity number:** &#entity\_number;

|  |  |  |
| --- | --- | --- |
| **Type** | **Example** | **Example Code** |
| **Entity Name** | Less than < | &lt; |
| **Entity Number** | Greater than > | &#62; |

### **🔹 Use entity numbers for better browser support.**

✅ Example (better support):

<p>10 &lt; 20</p> <!-- Displays: 10 < 20 -->

✅ Example (less reliable):

<p>10 &#60; 20</p> <!-- Displays: 10 < 20 -->

## **✅ 2. Reserved Characters**

Some characters are reserved because they are used to define HTML structure.

|  |  |  |  |
| --- | --- | --- | --- |
| **Character** | **Entity Name** | **Entity Number** | **Example** |
| < (less than) | &lt; | &#60; | 10 &lt; 20 → 10 < 20 |
| > (greater than) | &gt; | &#62; | 10 &gt; 5 → 10 > 5 |
| & (ampersand) | &amp; | &#38; | Me &amp; You → Me & You |
| " (double quote) | &quot; | &#34; | "Hello" → "Hello" |
| ' (apostrophe) | &apos; | &#39; | It's mine → It's mine |

## **✅ 3. Non-Breaking Space**

A **non-breaking space** (&nbsp; or &#160;) prevents line breaks between two words or symbols.

✅ Example:

<p>10&nbsp;km/h</p> <!-- Will stay on the same line -->

✅ Example with a non-breaking hyphen (&#8209;):

<p>state&#8209;of&#8209;the&#8209;art</p>

<!-- Displays: state-of-the-art without breaking -->

## **✅ 4. Commonly Used HTML Entities**

|  |  |  |  |
| --- | --- | --- | --- |
| **Character** | **Description** | **Entity Name** | **Entity Number** |
| © | Copyright | &copy; | &#169; |
| ® | Registered Trademark | &reg; | &#174; |
| ™ | Trademark | &trade; | &#8482; |
| € | Euro | &euro; | &#8364; |
| £ | Pound | &pound; | &#163; |
| ¥ | Yen | &yen; | &#165; |
| ¢ | Cent | &cent; | &#162; |

✅ Example:

<p>&copy; 2025 My Company. All rights reserved.</p>

**Output:** © 2025 My Company. All rights reserved.

## **✅ 5. Mathematical Symbols**

|  |  |  |  |
| --- | --- | --- | --- |
| **Character** | **Description** | **Entity Name** | **Entity Number** |
| ± | Plus-minus | &plusmn; | &#177; |
| ÷ | Division | &divide; | &#247; |
| × | Multiplication | &times; | &#215; |
| √ | Square root | &radic; | &#8730; |
| ∞ | Infinity | &infin; | &#8734; |
| ≈ | Approximately equal | &asymp; | &#8776; |

✅ Example:

<p>5 &times; 3 = 15</p> <!-- Displays: 5 × 3 = 15 -->

## **✅ 6. Currency Symbols**

|  |  |  |  |
| --- | --- | --- | --- |
| **Character** | **Description** | **Entity Name** | **Entity Number** |
| € | Euro | &euro; | &#8364; |
| £ | Pound | &pound; | &#163; |
| $ | Dollar | &#36; | &#36; |
| ¥ | Yen | &yen; | &#165; |
| ¢ | Cent | &cent; | &#162; |

✅ Example:

<p>Price: &euro;19.99</p> <!-- Displays: Price: €19.99 -->

## **✅ 7. Arrows and Punctuation**

|  |  |  |  |
| --- | --- | --- | --- |
| **Character** | **Description** | **Entity Name** | **Entity Number** |
| ← | Left Arrow | &larr; | &#8592; |
| → | Right Arrow | &rarr; | &#8594; |
| ↑ | Up Arrow | &uarr; | &#8593; |
| ↓ | Down Arrow | &darr; | &#8595; |
| “” | Double quotes | &ldquo; &rdquo; | &#8220; &#8221; |
| ‘’ | Single quotes | &lsquo; &rsquo; | &#8216; &#8217; |

✅ Example:

<p>&larr; Back | Next &rarr;</p> <!-- Displays: ← Back | Next → -->

## **✅ 8. Diacritical Marks**

You can combine **diacritical marks** with letters to create accented characters.

|  |  |  |  |
| --- | --- | --- | --- |
| **Mark** | **Example** | **Code** | **Output** |
| Grave ̀ | à | a&#768; | à |
| Acute ́ | á | a&#769; | á |
| Circumflex ̂ | â | a&#770; | â |
| Tilde ̃ | ã | a&#771; | ã |

✅ Example:

<p>a&#769; = á</p> <!-- Displays: á -->

✅ Yoruba Example:

<p>O&#768; = Ò</p> <!-- Displays: Ò -->

## **✅ 9. Miscellaneous Characters**

|  |  |  |  |
| --- | --- | --- | --- |
| **Character** | **Description** | **Entity Name** | **Entity Number** |
| • | Bullet | &bull; | &#8226; |
| … | Ellipsis | &hellip; | &#8230; |
| – | En dash | &ndash; | &#8211; |
| — | Em dash | &mdash; | &#8212; |
| ™ | Trademark | &trade; | &#8482; |

✅ Example:

<p>This is an example &ndash; with an en dash.</p>

## **✅ 10. Good Example of HTML Entities Usage**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>HTML Entities Example</title>

</head>

<body>

<h1>HTML Entities Example</h1>

<p>&copy; 2025 &mdash; All Rights Reserved</p>

<p>Price: &pound;10.00 &ndash; Special Offer!</p>

<p>&larr; Previous | Next &rarr;</p>

<p>a&#768; = à, o&#769; = ó</p>

</body>

</html>

## **✅ 11. Summary**

✔️ Use **entity numbers** for better browser support.  
 ✔️ Use **entity names** when easier to remember.  
 ✔️ Use **non-breaking spaces** to keep text on the same line.  
 ✔️ Combine diacritical marks for accented characters.  
 ✔️ Keep HTML entity usage consistent and clean.

## 🚀 Using HTML entities ensures consistent, clean, and accessible content!

## **HTML Symbols**

HTML allows you to display **special symbols** (like mathematical symbols, currency signs, and Greek letters) using **HTML entities**.

## **✅ 1. Basic Syntax**

You can add symbols using either:

* **Entity name** → &entity\_name;
* **Entity number** → &#entity\_number;

✅ Example:

<p>&copy; 2025 &mdash; All Rights Reserved</p>

<!-- Output: © 2025 — All Rights Reserved -->

## **✅ 2. Mathematical Symbols**

|  |  |  |  |
| --- | --- | --- | --- |
| **Symbol** | **Description** | **Entity Name** | **Entity Number** |
| + | Plus | &plus; | &#43; |
| − | Minus | &minus; | &#8722; |
| × | Multiply | &times; | &#215; |
| ÷ | Divide | &divide; | &#247; |
| = | Equals | &equals; | &#61; |
| ≠ | Not equal to | &ne; | &#8800; |
| ≈ | Approximately equal to | &asymp; | &#8776; |
| ∞ | Infinity | &infin; | &#8734; |
| √ | Square root | &radic; | &#8730; |
| ∑ | Summation | &sum; | &#8721; |
| ∫ | Integral | &int; | &#8747; |

✅ Example:

<p>5 &times; 4 = 20</p>

<!-- Output: 5 × 4 = 20 -->

## **✅ 3. Greek Letters**

|  |  |  |  |
| --- | --- | --- | --- |
| **Symbol** | **Description** | **Entity Name** | **Entity Number** |
| α | Alpha | &alpha; | &#945; |
| β | Beta | &beta; | &#946; |
| γ | Gamma | &gamma; | &#947; |
| Δ | Delta (uppercase) | &Delta; | &#916; |
| δ | Delta (lowercase) | &delta; | &#948; |
| ε | Epsilon | &epsilon; | &#949; |
| θ | Theta | &theta; | &#952; |
| λ | Lambda | &lambda; | &#955; |
| μ | Mu | &mu; | &#956; |
| π | Pi | &pi; | &#960; |
| Ω | Omega (uppercase) | &Omega; | &#937; |
| ω | Omega (lowercase) | &omega; | &#969; |

✅ Example:

<p>&alpha; + &beta; = &gamma;</p>

<!-- Output: α + β = γ -->

## **✅ 4. Currency Symbols**

|  |  |  |  |
| --- | --- | --- | --- |
| **Symbol** | **Description** | **Entity Name** | **Entity Number** |
| $ | Dollar | &#36; | &#36; |
| € | Euro | &euro; | &#8364; |
| £ | Pound | &pound; | &#163; |
| ¥ | Yen | &yen; | &#165; |
| ¢ | Cent | &cent; | &#162; |

✅ Example:

<p>Price: &euro;50</p>

<!-- Output: Price: €50 -->

## **✅ 5. Arrows and Punctuation**

|  |  |  |  |
| --- | --- | --- | --- |
| **Symbol** | **Description** | **Entity Name** | **Entity Number** |
| ← | Left arrow | &larr; | &#8592; |
| → | Right arrow | &rarr; | &#8594; |
| ↑ | Up arrow | &uarr; | &#8593; |
| ↓ | Down arrow | &darr; | &#8595; |
| “ | Left double quote | &ldquo; | &#8220; |
| ” | Right double quote | &rdquo; | &#8221; |
| ‘ | Left single quote | &lsquo; | &#8216; |
| ’ | Right single quote | &rsquo; | &#8217; |
| … | Ellipsis | &hellip; | &#8230; |
| – | En dash | &ndash; | &#8211; |
| — | Em dash | &mdash; | &#8212; |

✅ Example:

<p>&larr; Previous | Next &rarr;</p>

<!-- Output: ← Previous | Next → -->

## **✅ 6. Miscellaneous Symbols**

|  |  |  |  |
| --- | --- | --- | --- |
| **Symbol** | **Description** | **Entity Name** | **Entity Number** |
| © | Copyright | &copy; | &#169; |
| ® | Registered trademark | &reg; | &#174; |
| ™ | Trademark | &trade; | &#8482; |
| § | Section sign | &sect; | &#167; |
| ¶ | Paragraph sign | &para; | &#182; |
| • | Bullet | &bull; | &#8226; |
| † | Dagger | &dagger; | &#8224; |
| ‡ | Double dagger | &Dagger; | &#8225; |

✅ Example:

<p>&copy; 2025 My Company &trade;</p>

<!-- Output: © 2025 My Company ™ -->

## **✅ 7. Good Example of HTML Symbols**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>HTML Symbols Example</title>

</head>

<body>

<h1>HTML Symbols Example</h1>

<!-- Mathematical Symbols -->

<p>5 &times; 4 = 20</p>

<p>&radic;16 = 4</p>

<p>&infin; is the symbol for infinity</p>

<!-- Greek Letters -->

<p>&alpha; + &beta; = &gamma;</p>

<!-- Currency Symbols -->

<p>Price: &euro;19.99</p>

<!-- Arrows -->

<p>&larr; Previous | Next &rarr;</p>

<!-- Miscellaneous -->

<p>&copy; 2025 My Company &trade;</p>

</body>

</html>

## **✅ 8. Summary**

✔️ Use **entity names** when they are easy to remember.  
 ✔️ Use **entity numbers** for better browser compatibility.  
 ✔️ Combine symbols with other HTML elements for enhanced formatting.  
 ✔️ Keep HTML entity usage consistent and clean.

## 🚀 HTML symbols make it easy to display complex characters and ensure consistency across browsers!

## **HTML Emojis**

Emojis in HTML are **Unicode characters** from the **UTF-8** character set.

* They are **not images**—they are treated as **text characters**.
* UTF-8 supports almost all existing characters and symbols, including emojis.

## **✅ 1. Basic Emoji Syntax**

You can add emojis using either:

* **Decimal code** → &#128512;
* **Hexadecimal code** → &#x1F600;
* **Direct emoji** → 😀

✅ Example:

<p>&#128512; &#x1F600; 😀</p>

<!-- Output: 😀 😀 😀 -->

## **✅ 2. Common Emoji Categories**

### **😊 Smiley and Emotion Emojis**

|  |  |  |  |
| --- | --- | --- | --- |
| **Emoji** | **Description** | **Decimal Code** | **Hexadecimal Code** |
| 😀 | Grinning Face | &#128512; | &#x1F600; |
| 😃 | Grinning Face with Big Eyes | &#128513; | &#x1F603; |
| 😁 | Beaming Face with Smiling Eyes | &#128513; | &#x1F601; |
| 😂 | Face with Tears of Joy | &#128514; | &#x1F602; |
| 🥰 | Smiling Face with Hearts | &#129392; | &#x1F970; |

### **🍎 Food and Drink Emojis**

|  |  |  |  |
| --- | --- | --- | --- |
| **Emoji** | **Description** | **Decimal Code** | **Hexadecimal Code** |
| 🍎 | Red Apple | &#127822; | &#x1F34E; |
| 🍕 | Pizza | &#127829; | &#x1F355; |
| 🍔 | Hamburger | &#127828; | &#x1F354; |
| 🍦 | Ice Cream | &#127846; | &#x1F366; |
| ☕ | Hot Beverage | &#9749; | &#x2615; |

### **🌍 Nature and Animal Emojis**

|  |  |  |  |
| --- | --- | --- | --- |
| **Emoji** | **Description** | **Decimal Code** | **Hexadecimal Code** |
| 🌍 | Earth Globe Europe-Africa | &#127757; | &#x1F30D; |
| 🌞 | Sun with Face | &#127774; | &#x1F31E; |
| 🌧️ | Cloud with Rain | &#127783; | &#x1F327; |
| 🐶 | Dog Face | &#128054; | &#x1F436; |
| 🐱 | Cat Face | &#128049; | &#x1F431; |

### **🎯 Activity and Object Emojis**

|  |  |  |  |
| --- | --- | --- | --- |
| **Emoji** | **Description** | **Decimal Code** | **Hexadecimal Code** |
| ⚽ | Soccer Ball | &#9917; | &#x26BD; |
| 🏆 | Trophy | &#127942; | &#x1F3C6; |
| 🎸 | Guitar | &#127928; | &#x1F3B8; |
| 🎯 | Bullseye | &#127919; | &#x1F3AF; |
| 📸 | Camera with Flash | &#128248; | &#x1F4F8; |

### **📅 Time and Calendar Emojis**

|  |  |  |  |
| --- | --- | --- | --- |
| **Emoji** | **Description** | **Decimal Code** | **Hexadecimal Code** |
| 🕛 | Clock Face 12:00 | &#128347; | &#x1F55B; |
| 🗓️ | Calendar | &#128197; | &#x1F4C5; |
| ⏳ | Hourglass Not Done | &#9203; | &#x23F3; |
| 📅 | Calendar | &#128197; | &#x1F4C5; |
| ⏰ | Alarm Clock | &#9200; | &#x23F0; |

### **🚗 Travel and Transportation Emojis**

|  |  |  |  |
| --- | --- | --- | --- |
| **Emoji** | **Description** | **Decimal Code** | **Hexadecimal Code** |
| 🚗 | Car | &#128663; | &#x1F697; |
| ✈️ | Airplane | &#9992; | &#x2708; |
| 🏖️ | Beach with Umbrella | &#127958; | &#x1F3D6; |
| 🚀 | Rocket | &#128640; | &#x1F680; |
| 🚲 | Bicycle | &#128690; | &#x1F6B2; |

### **❤️ Hearts and Symbols**

|  |  |  |  |
| --- | --- | --- | --- |
| **Emoji** | **Description** | **Decimal Code** | **Hexadecimal Code** |
| ❤️ | Red Heart | &#10084; | &#x2764; |
| 💔 | Broken Heart | &#128148; | &#x1F494; |
| 💖 | Sparkling Heart | &#128150; | &#x1F496; |
| 💘 | Heart with Arrow | &#128152; | &#x1F498; |
| 💕 | Two Hearts | &#128149; | &#x1F495; |

## **✅ 3. Example of Emojis in HTML**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>HTML Emojis Example</title>

</head>

<body>

<h1>HTML Emojis Example</h1>

<!-- Smiley Emojis -->

<p>Feeling happy 😀 😂 🥰</p>

<!-- Food Emojis -->

<p>My favorite food: 🍎 🍕 🍔 🍦 ☕</p>

<!-- Nature Emojis -->

<p>Today's weather: 🌞 🌧️</p>

<!-- Activity Emojis -->

<p>Playing sports: ⚽ 🏆 🎯 🎸</p>

<!-- Travel Emojis -->

<p>Travel plans: ✈️ 🚗 🚲 🚀</p>

<!-- Heart Emojis -->

<p>I ❤️ coding!</p>

</body>

</html>

## **✅ 4. Emoji Best Practices**

✔️ Use **decimal codes** or **hexadecimal codes** for better browser support.  
 ✔️ Test emojis on different platforms since rendering may vary.  
 ✔️ Always include the UTF-8 encoding in the <head>:

<meta charset="UTF-8">

✔️ For SEO and accessibility, add descriptive text next to emojis.

## **✅ 5. Summary**

| ✅ Unicode characters are part of UTF-8.  
 | ✅ Emojis are supported in most modern browsers.  
 | ✅ Use **decimal or hexadecimal codes** for compatibility.  
 | ✅ Include UTF-8 encoding in the HTML document.

## 🚀 Emojis make your content more engaging and expressive! 😎

## **HTML Charset**

To display an HTML page correctly, a browser needs to know the **character set** used in the page.

## **✅ 1. Declaring Charset in HTML**

You define the charset in the <head> section using the <meta> tag:

<meta charset="UTF-8">

* UTF-8 is the default character set in HTML5.
* If no charset is specified, most browsers assume **UTF-8**.

✅ Example:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Charset Example</title>

</head>

<body>

<p>I will display A B C</p>

<p>I will display &#65; &#66; &#67;</p>

</body>

</html>

💡 A, B, C are represented by **65, 66, and 67** respectively in Unicode.

## **✅ 2. What is UTF-8?**

* **UTF-8** (Unicode Transformation Format) is the most widely used character set.
* Covers **all characters** and **symbols** across languages.
* UTF-8 is **backward-compatible** with ASCII.

✅ Example with UTF-8 characters:

<p style="font-size:48px">

&#128512; &#128516; &#128525; &#128151;

</p>

Output: 😀 😄 😍 💗

## **✅ 3. Character Encoding Overview**

|  |  |  |  |
| --- | --- | --- | --- |
| **Encoding** | **Description** | **Characters Supported** | **Notes** |
| **ASCII** | First encoding standard | 128 characters (0–127) | Basic Latin letters, digits, and symbols |
| **ISO-8859-1** | Western European encoding | 256 characters (0–255) | Extended Latin characters |
| **ANSI (Windows-1252)** | Windows-based extension of ISO-8859-1 | 256 characters | Adds 32 extra characters |
| **UTF-8** | Universal encoding | 1,112,064 characters | Covers all languages and symbols |

## **✅ 4. Why Use UTF-8?**

✔️ Supports almost all characters and symbols  
 ✔️ Compatible with ASCII  
 ✔️ Required for handling emojis and special characters  
 ✔️ Recommended in **HTML5**

## **✅ 5. Example with Different Encodings**

### **Example using UTF-8:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>UTF-8 Example</title>

</head>

<body>

<p>Emoji: 😎</p>

<p>Greek letter: &#945; (α)</p>

<p>Copyright: &copy;</p>

</body>

</html>

✅ Works perfectly in modern browsers ✅

### **Example using ISO-8859-1:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="ISO-8859-1">

<title>ISO-8859-1 Example</title>

</head>

<body>

<p>&euro; &pound; &yen;</p>

</body>

</html>

🚫 May not display emojis or certain symbols properly 🚫

## **✅ 6. ASCII to UTF-8 Transition**

|  |  |  |
| --- | --- | --- |
| **Encoding** | **Characters** | **Notes** |
| **ASCII** | 128 characters | Basic Latin characters |
| **ISO-8859-1** | 256 characters | Extended Latin characters |
| **UTF-8** | Over 1 million characters | Global standard, includes emojis and symbols |

## **✅ 7. Best Practices**

✔️ Always use **UTF-8** for modern web development.  
 ✔️ Place <meta charset="UTF-8"> at the **top of the <head> section**.  
 ✔️ Use **HTML entities** for special characters when needed:

* &#169; → ©
* &#128512; → 😀

## **✅ 8. Summary**

| ✅ Use UTF-8 for universal character support.  
 | ✅ Include the <meta charset="UTF-8"> tag early in the document.  
 | ✅ UTF-8 supports emojis and special characters.  
 | ✅ UTF-8 ensures consistent rendering across browsers.

## 🚀 UTF-8 = Universal Compatibility! 😎

## **HTML URL Encoding**

A **URL (Uniform Resource Locator)** is used by web browsers to request resources from a web server.

## **✅ 1. URL Structure**

A typical URL follows this format:

scheme://prefix.domain:port/path/filename

### **🔎 Explanation:**

|  |  |  |
| --- | --- | --- |
| **Part** | **Description** | **Example** |
| **scheme** | Type of internet service | http, https, ftp |
| **prefix** | Domain prefix | www |
| **domain** | Internet domain name | example.com |
| **port** | Port number | 80 (HTTP), 443 (HTTPS) |
| **path** | Path on the server | /products/ |
| **filename** | Name of the resource | index.html |

### **✅ Example:**

https://www.example.com:443/products/index.html

* **scheme** → https (secure)
* **prefix** → www
* **domain** → example.com
* **port** → 443 (default for HTTPS)
* **path** → /products/
* **filename** → index.html

## **✅ 2. Common URL Schemes**

|  |  |  |
| --- | --- | --- |
| **Scheme** | **Full Form** | **Purpose** |
| **http** | HyperText Transfer Protocol | Unsecured web pages |
| **https** | Secure HyperText Transfer Protocol | Secure web pages (encrypted) |
| **ftp** | File Transfer Protocol | Downloading and uploading files |
| **file** | Local file reference | Accessing files on a local device |

## **✅ 3. URL Encoding**

URLs can only use **ASCII characters** (letters, numbers, and basic symbols).

* Characters outside ASCII must be **encoded** using % followed by a hexadecimal value.
* Spaces are replaced with **%20** or + in query strings.

### **✅ Example of URL encoding:**

|  |  |
| --- | --- |
| **Character** | **Encoded Value** |
| **Space** | %20 or + |
| **!** | %21 |
| **#** | %23 |
| **$** | %24 |
| **%** | %25 |
| **&** | %26 |
| **(** | %28 |
| **)** | %29 |
| **+** | %2B |
| **/** | %2F |
| **:** | %3A |
| **?** | %3F |
| **@** | %40 |

### **✅ Example of URL Encoding:**

1. **Original URL:**

https://www.example.com/search?q=hello world!

1. **Encoded URL:**

https://www.example.com/search?q=hello%20world%21

* Space → %20
* ! → %21

## **✅ 4. URL Encoding in Forms**

When data is sent through a form using GET or POST, browsers automatically encode the values.

✅ Example:

<form action="/search" method="get">

<input type="text" name="q" value="hello world">

<input type="submit" value="Search">

</form>

Encoded URL:

/search?q=hello%20world

## **✅ 5. Why URL Encoding Matters**

✔️ Ensures compatibility across different systems and browsers.  
 ✔️ Prevents misinterpretation of characters.  
 ✔️ Encodes special characters like &, ?, and / to avoid breaking the URL structure.

## **✅ 6. Summary**

| ✅ Always encode special characters in URLs.  
 | ✅ Use %20 instead of + for encoding spaces in general URLs.  
 | ✅ URLs should follow the proper scheme and format.  
 | ✅ HTTPS is preferred for secure communication.

## 🚀 Use URL encoding to keep your links clean and functional!

## **HTML vs XHTML**

**XHTML** stands for **Extensible HyperText Markup Language** and is a stricter, more XML-based version of HTML.

## **✅ 1. What is XHTML?**

✔️ XHTML is HTML defined as an **XML application**.  
 ✔️ XHTML is supported by all major browsers.  
 ✔️ XHTML ensures that web pages are **well-formed** and follow stricter rules for compatibility with XML parsers.

## **✅ 2. Key Differences Between HTML and XHTML**

|  |  |  |
| --- | --- | --- |
| **Feature** | **HTML** | **XHTML** |
| **Doctype** | Optional | Mandatory |
| **Namespace (xmlns)** | Not required | Required |
| **Element Nesting** | Not strict | Must be properly nested |
| **Element Closure** | Optional for some elements | Mandatory |
| **Case Sensitivity** | Not case sensitive | Must be lowercase |
| **Attribute Quotation** | Optional | Mandatory |
| **Minimized Attributes** | Allowed | Forbidden |
| **Empty Elements** | Can be self-closing or not | Must be self-closing |

## **✅ 3. XHTML Rules**

### **🔎 1. Doctype is Mandatory**

XHTML requires a proper <!DOCTYPE> declaration.  
 ✅ Example:

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"

"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

### **🔎 2. XML Namespace (xmlns) is Mandatory**

The xmlns attribute defines the XML namespace.  
 ✅ Example:

<html xmlns="http://www.w3.org/1999/xhtml">

### **🔎 3. <html>, <head>, <title>, and <body> are Mandatory**

✅ Example:

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"

"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title>XHTML Example</title>

</head>

<body>

<p>Hello, world!</p>

</body>

</html>

### **🔎 4. Elements Must be Properly Nested**

✅ Correct:

<b><i>Some text</i></b>

❌ Incorrect:

<b><i>Some text</b></i>

### **🔎 5. Elements Must Always Be Closed**

✅ Correct:

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

❌ Incorrect:

<p>This is a paragraph.

### **🔎 6. Empty Elements Must Always Be Closed**

✅ Correct:

<br />

<hr />

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

❌ Incorrect:

<br>

<hr>

<img src="image.jpg" alt="Example">

### **🔎 7. Elements Must Be in Lowercase**

✅ Correct:

<body>

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

</body>

❌ Incorrect:

<BODY>

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

</BODY>

### **🔎 8. Attribute Names Must Be in Lowercase**

✅ Correct:

<a href="https://example.com">Visit our site</a>

❌ Incorrect:

<A HREF="https://example.com">Visit our site</A>

### **🔎 9. Attribute Values Must Be Quoted**

✅ Correct:

<input type="checkbox" name="vehicle" value="car" checked="checked" />

❌ Incorrect:

<input type=checkbox name=vehicle value=car checked>

### **🔎 10. Attribute Minimization is Forbidden**

✅ Correct:

<input type="checkbox" checked="checked" />

❌ Incorrect:

<input type="checkbox" checked />

## **✅ 4. Example of Valid XHTML**

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"

"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title>XHTML Example</title>

</head>

<body>

<h1>XHTML Rules</h1>

<p>This is a valid XHTML document.</p>

<img src="example.jpg" alt="Example" />

</body>

</html>

## **✅ 5. Validate Your XHTML**

You can validate XHTML documents using the W3C Validator:  
 👉 [https://validator.w3.org](https://validator.w3.org/)

## **✅ 6. Why XHTML Matters**

✔️ Improves compatibility with XML-based tools.  
 ✔️ Ensures consistent browser rendering.  
 ✔️ More structured and predictable code.

## 🚀 Stick to XHTML rules for better compatibility and cleaner code!

## **HTML Forms**

HTML forms are used to **collect user input** and send the data to a server for processing.

## **✅ 1. Basic Structure of a Form**

A basic form is created using the <form> element.

✅ Example:

<h2>HTML Forms</h2>

<form action="/action\_page.php">

<label for="fname">First name:</label><br>

<input type="text" id="fname" name="fname" value="John"><br>

<label for="lname">Last name:</label><br>

<input type="text" id="lname" name="lname" value="Doe"><br><br>

<input type="submit" value="Submit">

</form>

### **🔎 Explanation**

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Description** | **Example** |
| action | URL where form data will be sent | /action\_page.php |
| method | HTTP method (GET/POST) | method="post" |
| id | Uniquely identifies the element | id="fname" |
| name | Assigns a name to the input | name="fname" |
| value | Initial value of the input field | value="John" |

## **✅ 2. <input> Types**

The <input> element is the most common form element.

### **➡️ Text Field**

Displays a single-line text input field.  
 ✅ Example:

<form>

<label for="username">Username:</label>

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

</form>

### **➡️ Password Field**

Hides the characters entered by the user.  
 ✅ Example:

<form>

<label for="password">Password:</label>

<input type="password" id="password" name="password">

</form>

### **➡️ Radio Buttons**

Allows the user to select **one option** from multiple choices.  
 ✅ Example:

<form>

<input type="radio" id="male" name="gender" value="male">

<label for="male">Male</label><br>

<input type="radio" id="female" name="gender" value="female">

<label for="female">Female</label><br>

<input type="radio" id="other" name="gender" value="other">

<label for="other">Other</label>

</form>

* **Only one option** can be selected at a time (because of the same name attribute).

### **➡️ Checkboxes**

Allows the user to select **multiple options**.  
 ✅ Example:

<form>

<input type="checkbox" id="vehicle1" name="vehicle1" value="Bike">

<label for="vehicle1">I have a bike</label><br>

<input type="checkbox" id="vehicle2" name="vehicle2" value="Car">

<label for="vehicle2">I have a car</label><br>

<input type="checkbox" id="vehicle3" name="vehicle3" value="Boat">

<label for="vehicle3">I have a boat</label>

</form>

* Multiple checkboxes **can be selected** at the same time.

### **➡️ Submit Button**

Sends the form data to the server.  
 ✅ Example:

<form action="/action\_page.php">

<input type="submit" value="Submit">

</form>

* Clicking the button will trigger the **form submission**.

### **➡️ Reset Button**

Clears all form inputs.  
 ✅ Example:

<form>

<input type="reset" value="Reset">

</form>

### **➡️ Button**

Displays a clickable button (does not submit form by default).  
 ✅ Example:

<form>

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

</form>

## **✅ 3. <label> Tag**

The <label> element defines a label for an input field.

✅ Example:

<label for="email">Email:</label>

<input type="email" id="email" name="email">

* The for attribute of <label> matches the id of the input field.
* Improves **accessibility** and **clickable area** for users.

## **✅ 4. name Attribute**

The name attribute is **required** for input fields to submit data to the server.

✅ Example:

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

* If the name is missing, the value will **NOT be submitted**.

## **✅ 5. Form Submission with GET and POST**

### **➡️ GET**

* Data is appended to the URL as query parameters.
* Should be used for **retrieving data** (not sensitive).

✅ Example:

<form action="/search" method="get">

<input type="text" name="q">

<input type="submit" value="Search">

</form>

🔎 URL Example:

https://example.com/search?q=value

### **➡️ POST**

* Data is sent in the **body** of the request.
* Should be used for **sending sensitive data** (e.g., passwords).

✅ Example:

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

<input type="text" name="username">

<input type="password" name="password">

<input type="submit" value="Login">

</form>

## **✅ 6. Input Validation**

Use the required attribute to **validate input** before submission.

✅ Example:

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

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

<input type="email" name="email" required>

<input type="submit" value="Submit">

</form>

## **✅ 7. Placeholder Attribute**

Provides a short hint inside the input field.

✅ Example:

<input type="text" placeholder="Enter your name">

## **✅ 8. disabled Attribute**

Disables the input field (user cannot interact with it).

✅ Example:

<input type="text" value="Read-only" disabled>

## **✅ 9. readonly Attribute**

Prevents user input but allows the value to be submitted.

✅ Example:

<input type="text" value="Read-only" readonly>

## **✅ 10. Example of a Complete Form**

✅ Example:

<h2>Registration Form</h2>

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

<label for="username">Username:</label>

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

<label for="password">Password:</label>

<input type="password" id="password" name="password" required><br>

<label>Gender:</label><br>

<input type="radio" id="male" name="gender" value="male">

<label for="male">Male</label>

<input type="radio" id="female" name="gender" value="female">

<label for="female">Female</label><br>

<label>Interests:</label><br>

<input type="checkbox" id="sports" name="interest" value="sports">

<label for="sports">Sports</label>

<input type="checkbox" id="music" name="interest" value="music">

<label for="music">Music</label><br>

<input type="submit" value="Register">

</form>

## **🚀 Form Best Practices**

✅ Always add name attributes to inputs.  
 ✅ Use POST for sensitive data.  
 ✅ Use label for accessibility.  
 ✅ Add required for essential fields.  
 ✅ Keep forms clean and simple.

## 🎯 Master HTML Forms for Better UX!

## **HTML Form Attributes**

HTML form attributes define the behavior and functionality of a form when it's submitted or interacted with.

## **✅ 1. action Attribute**

Specifies where to send the form data when the form is submitted.

### **🔎 Syntax:**

<form action="URL">

### **✅ Example:**

Send form data to a file on the server:

<form action="/action\_page.php">

<label for="fname">First name:</label>

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

<label for="lname">Last name:</label>

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

<input type="submit" value="Submit">

</form>

### **🔥 Notes:**

* If action is omitted, the form data will be sent to the **current page**.

## **✅ 2. method Attribute**

Defines the HTTP method used to send form data to the server.

|  |  |
| --- | --- |
| **Method** | **Description** |
| GET | Appends form data to the URL as query strings. |
| POST | Sends form data in the body of the request. |

### **➡️ GET Method**

* Suitable for **non-sensitive data**.
* Data is **visible** in the URL.
* URL length limit (~2048 characters).  
   ✅ Example:

<form action="/action\_page.php" method="get">

<input type="text" name="query" value="Search">

<input type="submit" value="Submit">

</form>

🔎 Example URL:

https://example.com/action\_page.php?query=Search

### **➡️ POST Method**

* Suitable for **sensitive data**.
* Data is **hidden** from the URL.
* No size limit for data.  
   ✅ Example:

<form action="/action\_page.php" method="post">

<input type="text" name="username">

<input type="password" name="password">

<input type="submit" value="Submit">

</form>

## **✅ 3. target Attribute**

Specifies where to display the response after submitting the form.

|  |  |
| --- | --- |
| **Value** | **Description** |
| \_self | Open response in the **same window** (default). |
| \_blank | Open response in a **new window/tab**. |
| \_parent | Open response in the **parent frame**. |
| \_top | Open response in the **full window**. |
| framename | Open response in a specific iframe. |

### **✅ Example:**

Open the response in a **new tab**:

<form action="/action\_page.php" target="\_blank">

<input type="text" name="query">

<input type="submit" value="Search">

</form>

## **✅ 4. autocomplete Attribute**

Enables or disables **auto-filling** of form fields based on the browser's history.

|  |  |
| --- | --- |
| **Value** | **Description** |
| on | Enable autofill (default). |
| off | Disable autofill. |

### **✅ Example:**

Enable autocomplete:

<form action="/action\_page.php" autocomplete="on">

<input type="text" name="username">

<input type="submit" value="Submit">

</form>

Disable autocomplete:

<form action="/action\_page.php" autocomplete="off">

<input type="text" name="username">

<input type="submit" value="Submit">

</form>

## **✅ 5. novalidate Attribute**

Prevents the browser from **validating form data** before submission.

✅ Example:

<form action="/action\_page.php" novalidate>

<input type="email" name="email" required>

<input type="submit" value="Submit">

</form>

* The required attribute will be **ignored** when novalidate is present.

## **✅ 6. accept-charset Attribute**

Defines the **character encoding** for the submitted form data.

|  |  |
| --- | --- |
| **Value** | **Description** |
| UTF-8 | Default encoding (covers most characters). |
| ISO-8859-1 | Western European encoding. |

✅ Example:

<form action="/action\_page.php" accept-charset="UTF-8">

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

<input type="submit" value="Submit">

</form>

## **✅ 7. enctype Attribute**

Defines how the form data should be encoded when submitting a POST request.

|  |  |
| --- | --- |
| **Value** | **Description** |
| application/x-www-form-urlencoded | Default encoding (used for text data). |
| multipart/form-data | Used for uploading files. |
| text/plain | Data sent as plain text. |

### **✅ Example:**

For uploading files:

<form action="/upload" method="post" enctype="multipart/form-data">

<input type="file" name="file">

<input type="submit" value="Upload">

</form>

## **✅ 8. name Attribute**

Defines a unique name for the form to reference it in **JavaScript** or **server-side code**.

✅ Example:

<form action="/submit" name="myForm">

<input type="text" name="username">

<input type="submit" value="Submit">

</form>

* In JavaScript:

document.forms["myForm"].submit();

## **✅ 9. rel Attribute**

Specifies the relationship between the current document and the form action target.

|  |  |
| --- | --- |
| **Value** | **Description** |
| noopener | Prevents opening pages from accessing the original window. |
| noreferrer | Prevents sending the Referer header. |
| external | Indicates the target is an external resource. |

✅ Example:

<form action="https://example.com" rel="noopener noreferrer">

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

<input type="submit" value="Submit">

</form>

## **✅ 10. Complete Form Example**

✅ Example:

<h2>Registration Form</h2>

<form action="/register" method="post" target="\_blank" autocomplete="on" enctype="multipart/form-data" accept-charset="UTF-8" name="registerForm" rel="noopener noreferrer">

<label for="username">Username:</label>

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

<label for="password">Password:</label>

<input type="password" id="password" name="password" required><br>

<label for="gender">Gender:</label><br>

<input type="radio" id="male" name="gender" value="male"> Male<br>

<input type="radio" id="female" name="gender" value="female"> Female<br>

<label for="file">Upload Profile Picture:</label>

<input type="file" id="file" name="file"><br>

<input type="submit" value="Register">

</form>

## **✅ Summary of Form Attributes**

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| action | URL to send form data to |
| method | HTTP method (GET or POST) |
| target | Where to display response |
| autocomplete | Enable/disable autofill |
| novalidate | Disable form validation |
| accept-charset | Character encoding for form submission |
| enctype | How to encode form data |
| name | Name of the form |
| rel | Relationship between form and linked resource |

## 🚀 Master HTML Forms to Build Powerful Web Apps!

### **HTML Form Elements Summary**

HTML forms collect user input using various form elements:

#### **Basic Elements**

|  |  |
| --- | --- |
| **Tag** | **Description** |
| <form> | Defines a form for user input. |
| <input> | Defines an input control (various types). |
| <label> | Labels an input element for accessibility. |
| <textarea> | Multiline text input field. |
| <button> | Defines a clickable button. |
| <select> | Drop-down list. |
| <option> | Defines an option in a <select>. |
| <optgroup> | Groups related <option> elements. |
| <fieldset> | Groups related form elements. |
| <legend> | Caption for a <fieldset>. |
| <datalist> | Provides a list of predefined input options. |
| <output> | Displays the result of a calculation. |

#### **Common Input Types**

|  |  |
| --- | --- |
| **Type** | **Description** |
| text | Single-line text input. |
| password | Masked text input. |
| submit | Submits form data. |
| reset | Resets form values. |
| radio | Select one option from a group. |
| checkbox | Select multiple options. |
| button | Clickable button. |
| color | Color picker. |
| date | Date picker. |
| datetime-local | Date and time picker (no time zone). |
| email | Email input with validation. |
| file | File upload. |
| month | Month and year picker. |
| number | Numeric input (with min/max/step). |
| range | Slider control. |
| search | Search input. |
| tel | Phone number input (with pattern). |
| time | Time picker. |
| url | URL input with validation. |
| week | Week and year picker. |

#### **Special Attributes**

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| required | Makes the field mandatory. |
| disabled | Disables the input. |
| readonly | Makes input read-only. |
| maxlength | Limits character count. |
| pattern | Validates using a regex pattern. |
| min / max | Sets minimum and maximum values. |
| step | Sets value intervals (e.g., 10, 20, 30). |
| value | Sets default value. |
| size | Sets input width in characters. |

### **✅ Basic Form Structure**

A form is defined using the <form> tag. The action attribute specifies where to send the form data when submitted.

html

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

<!-- Form elements go here -->

</form>

### **🔥 Complete Code Example Using Various Elements**

This example includes **input fields**, **labels**, **dropdowns**, **buttons**, and other form elements:

html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

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

<title>HTML Form Example</title>

</head>

<body>

<h2>Registration Form</h2>

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

<!-- Text input -->

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

<input type="text" id="name" name="name" required minlength="3" maxlength="30" /><br><br>

<!-- Email input -->

<label for="email">Email:</label>

<input type="email" id="email" name="email" required /><br><br>

<!-- Password input -->

<label for="password">Password:</label>

<input type="password" id="password" name="password" required minlength="6" /><br><br>

<!-- Radio buttons -->

<label>Gender:</label><br>

<input type="radio" id="male" name="gender" value="male">

<label for="male">Male</label><br>

<input type="radio" id="female" name="gender" value="female">

<label for="female">Female</label><br><br>

<!-- Checkbox -->

<input type="checkbox" id="subscribe" name="subscribe" checked>

<label for="subscribe">Subscribe to newsletter</label><br><br>

<!-- Select dropdown -->

<label for="country">Country:</label>

<select id="country" name="country" required>

<option value="">Select your country</option>

<option value="US">United States</option>

<option value="IN">India</option>

<option value="UK">United Kingdom</option>

</select><br><br>

<!-- Datalist for autocomplete -->

<label for="browser">Preferred Browser:</label>

<input list="browsers" id="browser" name="browser">

<datalist id="browsers">

<option value="Chrome">

<option value="Firefox">

<option value="Safari">

<option value="Edge">

</datalist><br><br>

<!-- Number input -->

<label for="age">Age:</label>

<input type="number" id="age" name="age" min="18" max="100" required><br><br>

<!-- Date input -->

<label for="dob">Date of Birth:</label>

<input type="date" id="dob" name="dob" min="1900-01-01" max="2025-12-31" required><br><br>

<!-- Time input -->

<label for="appt">Preferred Time:</label>

<input type="time" id="appt" name="appt"><br><br>

<!-- File input -->

<label for="resume">Upload Resume:</label>

<input type="file" id="resume" name="resume" accept=".pdf,.docx"><br><br>

<!-- Color input -->

<label for="color">Favorite Color:</label>

<input type="color" id="color" name="color" value="#ff0000"><br><br>

<!-- Range slider -->

<label for="satisfaction">Satisfaction (0-100):</label>

<input type="range" id="satisfaction" name="satisfaction" min="0" max="100" step="5" value="50"><br><br>

<!-- Output element -->

<label for="result">Calculated Result:</label>

<output name="result" id="result" for="satisfaction"></output><br><br>

<!-- Textarea -->

<label for="comments">Comments:</label>

<textarea id="comments" name="comments" rows="4" cols="50"></textarea><br><br>

<!-- Submit button -->

<input type="submit" value="Submit">

<!-- Reset button -->

<input type="reset" value="Reset">

</form>

<script>

// Example of calculating value with output element

const satisfaction = document.getElementById('satisfaction');

const result = document.getElementById('result');

satisfaction.addEventListener('input', () => {

result.value = satisfaction.value;

});

</script>

</body>

### **✅ Best Practices**

* Use <label> with the for attribute for better accessibility.
* Always set the type attribute on <input> elements.
* Use required, pattern, and other attributes to improve validation.
* Group related inputs using <fieldset> and <legend>.

## **HTML Input Form Attributes**

HTML forms support various attributes that allow fine control over how data is collected and submitted. Here’s a complete guide with **examples** and **tips** for each attribute:

## **✅ 1. form Attribute**

The form attribute specifies which <form> element the input belongs to.

* The value must match the id of the target <form> element.
* Useful when an input is placed **outside** the <form> tag but still belongs to that form.

### **💡 Example**

<form action="/submit" id="myForm">

<label for="fname">First name:</label>

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

<input type="submit" value="Submit">

</form>

<!-- Input outside the form but still linked to it using the form attribute -->

<label for="lname">Last name:</label>

<input type="text" id="lname" name="lname" form="myForm">

**🔎 How it works:**

* The lname input is not inside the <form> but will still be submitted with myForm because of form="myForm".

## **✅ 2. formaction Attribute**

The formaction attribute specifies a **URL** to process the form data when submitted.

* Overrides the action attribute of the <form>.
* Only works with <input type="submit"> and <input type="image">.

### **💡 Example**

<form action="/default\_action">

<label for="fname">First name:</label>

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

<input type="submit" value="Submit">

<input type="submit" formaction="/admin\_action" value="Submit as Admin">

</form>

**🔎 How it works:**

* The first submit button will send data to /default\_action.
* The second submit button will send data to /admin\_action.

## **✅ 3. formenctype Attribute**

The formenctype attribute specifies how to encode form data when submitted.

* Only applies when method="post".
* Overrides the enctype attribute of the <form>.

### **Encoding Options:**

|  |  |
| --- | --- |
| **Value** | **Description** |
| application/x-www-form-urlencoded | Default. Encodes form data as key-value pairs. |
| multipart/form-data | Used for file uploads. |
| text/plain | Sends data as plain text. |

### **💡 Example**

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

<label for="file">Upload File:</label>

<input type="file" id="file" name="file"><br><br>

<input type="submit" value="Submit">

<input type="submit" formenctype="multipart/form-data" value="Submit as Multipart/form-data">

</form>

**🔎 How it works:**

* First button sends form data using the default encoding.
* Second button sends data as multipart/form-data (required for file uploads).

## **✅ 4. formmethod Attribute**

The formmethod attribute defines the HTTP method (GET or POST) to send form data.

* Overrides the method attribute of the <form>.

### **💡 Example**

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

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

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

<input type="submit" value="Submit using GET">

<input type="submit" formmethod="post" value="Submit using POST">

</form>

**🔎 How it works:**

* First button sends data using GET (visible in URL).
* Second button sends data using POST (secure, not visible in URL).

## **✅ 5. formtarget Attribute**

The formtarget attribute specifies where to display the response after submitting the form.

* Overrides the target attribute of the <form>.

### **Target Options:**

|  |  |
| --- | --- |
| **Value** | **Description** |
| \_self | (Default) Opens in the same window. |
| \_blank | Opens in a new tab/window. |
| \_parent | Opens in the parent frame. |
| \_top | Opens in the full body of the window. |

### **💡 Example**

<form action="/submit" target="\_self">

<label for="email">Email:</label>

<input type="email" id="email" name="email" required><br><br>

<input type="submit" value="Submit">

<input type="submit" formtarget="\_blank" value="Submit in New Tab">

</form>

**🔎 How it works:**

* First button opens the response in the same window.
* Second button opens the response in a new tab.

## **✅ 6. formnovalidate Attribute**

The formnovalidate attribute specifies that form validation should be **skipped** for the specific button.

* Works only with <input type="submit">.

### **💡 Example**

<form action="/submit">

<label for="email">Email:</label>

<input type="email" id="email" name="email" required><br><br>

<input type="submit" value="Submit (Validation)">

<input type="submit" formnovalidate="formnovalidate" value="Submit without Validation">

</form>

**🔎 How it works:**

* First button will validate the form before submission.
* Second button will submit without validating.

## **✅ 7. novalidate Attribute**

The novalidate attribute on the <form> element disables validation for **all inputs**.

### **💡 Example**

<form action="/submit" novalidate>

<label for="email">Email:</label>

<input type="email" id="email" name="email" required><br><br>

<input type="submit" value="Submit without Validation">

</form>

**🔎 How it works:**

* Form data will be submitted without any client-side validation.

## **🚀 Complete Example Using All Attributes**

<form action="/submit" method="get" target="\_self" id="mainForm">

<!-- form attribute -->

<label for="fname">First Name:</label>

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

<!-- formaction and formmethod -->

<input type="submit" value="Submit with GET">

<input type="submit" formmethod="post" formaction="/admin" value="Submit as POST to Admin">

<!-- formtarget -->

<input type="submit" formtarget="\_blank" value="Submit in New Tab">

<!-- formnovalidate -->

<input type="submit" formnovalidate value="Submit without Validation">

</form>

<!-- form attribute linking -->

<label for="lname">Last Name:</label>

<input type="text" id="lname" name="lname" form="mainForm">

## **🎯 Summary of Key Differences**

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Purpose** | **Overridden Element** |
| form | Links input to a form by id | <form> |
| formaction | URL for submission | action |
| formenctype | Encoding type | enctype |
| formmethod | HTTP method | method |
| formtarget | Where to display response | target |
| formnovalidate | Skips validation | novalidate |
| novalidate | Disables validation for the whole form | N/A |

## **💡 Tips:**

✅ Use POST for sensitive data (more secure).  
 ✅ Use multipart/form-data for file uploads.  
 ✅ formnovalidate is helpful for quick tests or non-critical inputs.  
 ✅ formtarget="\_blank" is useful for external links or downloads.

## **HTML Graphics (Canvas and SVG)**

HTML provides two main methods for creating graphics on a webpage:

1. **Canvas** – For pixel-based graphics, drawn using JavaScript.
2. **SVG (Scalable Vector Graphics)** – For vector-based graphics, defined using XML.

## **✅ 1. HTML Canvas**

The <canvas> element is used to draw graphics on the fly using **JavaScript**.

* Canvas is resolution-dependent and does not support event handlers directly.
* Graphics are drawn pixel by pixel, meaning once drawn, they cannot be easily modified.

### **💡 Canvas Example**

Create a canvas and draw a red rectangle:

<canvas id="myCanvas" width="300" height="150" style="border:1px solid #000;"></canvas>

<script>

const canvas = document.getElementById('myCanvas');

const ctx = canvas.getContext('2d');

// Draw a red rectangle

ctx.fillStyle = 'red';

ctx.fillRect(20, 20, 150, 100);

</script>

### **🔎 Explanation:**

* getContext('2d') → Returns a drawing context on the canvas.
* fillStyle → Sets the fill color.
* fillRect(x, y, width, height) → Draws a rectangle.

### **✅ Adding a Border**

You can add a border using the style attribute:

<canvas id="myCanvas" width="300" height="150" style="border:2px solid black;"></canvas>

### **✅ Draw a Line**

Use beginPath() and moveTo() to define the path:

<canvas id="myCanvas" width="300" height="150"></canvas>

<script>

const canvas = document.getElementById('myCanvas');

const ctx = canvas.getContext('2d');

ctx.beginPath();

ctx.moveTo(20, 20);

ctx.lineTo(180, 120);

ctx.strokeStyle = 'blue';

ctx.lineWidth = 2;

ctx.stroke();

</script>

### **✅ Draw a Circle**

Use arc() to draw a circle:

<canvas id="myCanvas" width="300" height="150"></canvas>

<script>

const canvas = document.getElementById('myCanvas');

const ctx = canvas.getContext('2d');

ctx.beginPath();

ctx.arc(100, 75, 50, 0, 2 \* Math.PI); // x, y, radius, startAngle, endAngle

ctx.fillStyle = 'green';

ctx.fill();

</script>

### **✅ Draw Text**

Use fillText() to draw text:

<canvas id="myCanvas" width="300" height="150"></canvas>

<script>

const canvas = document.getElementById('myCanvas');

const ctx = canvas.getContext('2d');

ctx.font = '30px Arial';

ctx.fillStyle = 'purple';

ctx.fillText('Hello Canvas', 10, 50);

</script>

### **✅ Draw an Image**

Use drawImage() to add an image:

<canvas id="myCanvas" width="300" height="150"></canvas>

<script>

const canvas = document.getElementById('myCanvas');

const ctx = canvas.getContext('2d');

const img = new Image();

img.onload = () => {

ctx.drawImage(img, 10, 10);

};

img.src = 'https://via.placeholder.com/100';

</script>

### **✅ Create a Linear Gradient**

Use createLinearGradient():

<canvas id="myCanvas" width="300" height="150"></canvas>

<script>

const canvas = document.getElementById('myCanvas');

const ctx = canvas.getContext('2d');

const gradient = ctx.createLinearGradient(0, 0, 300, 0);

gradient.addColorStop(0, 'red');

gradient.addColorStop(1, 'blue');

ctx.fillStyle = gradient;

ctx.fillRect(0, 0, 300, 150);

</script>

## **✅ 2. HTML SVG**

The <svg> element is used to create vector-based graphics.

* SVG is **resolution-independent** and supports **event handling**.
* SVG elements are treated like DOM elements, meaning you can apply CSS and JavaScript directly.

### **💡 SVG Example**

Draw a red rectangle using SVG:

<svg width="300" height="150" style="border:1px solid black;">

<rect x="20" y="20" width="150" height="100" fill="red" />

</svg>

### **✅ Draw a Circle**

Use the <circle> element:

<svg width="300" height="150" style="border:1px solid black;">

<circle cx="100" cy="75" r="50" fill="green" />

</svg>

### **✅ Draw a Line**

Use the <line> element:

<svg width="300" height="150" style="border:1px solid black;">

<line x1="10" y1="10" x2="100" y2="100" stroke="blue" stroke-width="2" />

</svg>

### **✅ Draw Text**

Use the <text> element:

<svg width="300" height="150" style="border:1px solid black;">

<text x="10" y="50" fill="purple" font-size="30">Hello SVG</text>

</svg>

### **✅ SVG with JavaScript**

You can modify SVG with JavaScript:

<svg id="mySvg" width="300" height="150" style="border:1px solid black;">

<circle id="myCircle" cx="100" cy="75" r="50" fill="blue" />

</svg>

<script>

document.getElementById('myCircle').setAttribute('fill', 'red');

</script>

## **✅ Comparison Between Canvas and SVG**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Canvas** | **SVG** |
| **Rendering** | Pixel-based | Vector-based |
| **Resolution** | Resolution-dependent | Resolution-independent |
| **Event Handling** | No | Yes |
| **Animation** | Manual (JavaScript) | Built-in support |
| **Performance** | Faster for complex graphics | Slower for complex graphics |
| **Best for** | Games, real-time rendering | Static graphics, scalable graphics |
| **Modifiability** | Redraw needed | DOM-based (modification possible) |

## **🚀 Complete Example Using Canvas and SVG**

### **Canvas Example**

<canvas id="canvas" width="200" height="100" style="border:1px solid black;"></canvas>

<script>

const canvas = document.getElementById('canvas');

const ctx = canvas.getContext('2d');

ctx.fillStyle = 'orange';

ctx.fillRect(20, 20, 150, 60);

</script>

### **SVG Example**

<svg width="200" height="100" style="border:1px solid black;">

<rect x="20" y="20" width="150" height="60" fill="orange" />

</svg>

## **🎯 When to Use Canvas vs SVG**

|  |  |
| --- | --- |
| **Use Case** | **Best Option** |
| **High-performance graphics** (games, real-time updates) | ✅ Canvas |
| **Interactive graphics** (charts, diagrams) | ✅ SVG |
| **Static graphics** (icons, logos) | ✅ SVG |
| **Complex scenes with dynamic updates** | ✅ Canvas |
| **Scalable graphics** | ✅ SVG |

## **💡 Tips:**

✅ Use Canvas for high-performance, game-like graphics.  
 ✅ Use SVG for graphics that need to be scalable and interactive.  
 ✅ Canvas is faster for thousands of objects; SVG handles complex shapes better.

## **HTML Media**

HTML allows embedding multimedia content such as **audio, video, and interactive media** directly in a webpage.

## **✅ 1. HTML Video**

Use the <video> element to embed video files on a webpage.  
 Supported formats: **MP4**, **WebM**, and **Ogg**.

### **Example – Basic video element:**

<video width="320" height="240" controls>

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

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

Your browser does not support the video tag.

</video>

### **🔎 Explanation:**

* controls → Shows play, pause, and volume controls.
* <source> → Allows multiple video formats (browser picks the first supported one).
* Text inside <video> → Shown if the browser doesn’t support video.

### **✅ Autoplay and Mute**

Start video playback automatically (muted autoplay works in most browsers).

<video width="320" height="240" autoplay muted>

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

</video>

### **✅ Loop**

Make the video repeat continuously:

<video width="320" height="240" autoplay loop muted>

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

</video>

### **✅ Disable Controls**

Hide play and pause controls:

<video width="320" height="240" autoplay muted controls="false">

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

</video>

### **✅ Custom Video Controls with JavaScript**

<video id="myVideo" width="320" height="240" controls>

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

</video>

<button onclick="document.getElementById('myVideo').play()">Play</button>

<button onclick="document.getElementById('myVideo').pause()">Pause</button>

<button onclick="document.getElementById('myVideo').volume += 0.1">Volume Up</button>

<button onclick="document.getElementById('myVideo').volume -= 0.1">Volume Down</button>

## **✅ 2. HTML Audio**

Use the <audio> element to play sound or music files.  
 Supported formats: **MP3**, **WAV**, and **Ogg**.

### **Example – Basic audio element:**

<audio controls>

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

<source src="sound.ogg" type="audio/ogg">

Your browser does not support the audio element.

</audio>

### **🔎 Explanation:**

* controls → Shows play, pause, and volume controls.
* <source> → Allows multiple audio formats (browser picks the first supported one).

### **✅ Autoplay and Loop**

Start audio playback automatically and loop:

<audio autoplay loop>

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

</audio>

### **✅ Disable Controls**

<audio autoplay loop controls="false">

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

</audio>

### **✅ Custom Audio Controls with JavaScript**

<audio id="myAudio" controls>

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

</audio>

<button onclick="document.getElementById('myAudio').play()">Play</button>

<button onclick="document.getElementById('myAudio').pause()">Pause</button>

<button onclick="document.getElementById('myAudio').volume += 0.1">Volume Up</button>

<button onclick="document.getElementById('myAudio').volume -= 0.1">Volume Down</button>

## **✅ 3. HTML YouTube Embedding**

You can embed YouTube videos using the <iframe> element.

### **Example – Basic YouTube Embed:**

<iframe width="560" height="315" src="https://www.youtube.com/embed/tgbNymZ7vqY" allowfullscreen></iframe>

### **✅ Autoplay and Mute**

<iframe width="560" height="315" src="https://www.youtube.com/embed/tgbNymZ7vqY?autoplay=1&mute=1" allowfullscreen></iframe>

### **✅ Loop**

Loop a video:

<iframe width="560" height="315" src="https://www.youtube.com/embed/tgbNymZ7vqY?playlist=tgbNymZ7vqY&loop=1" allowfullscreen></iframe>

### **✅ Hide Controls**

<iframe width="560" height="315" src="https://www.youtube.com/embed/tgbNymZ7vqY?controls=0" allowfullscreen></iframe>

## **✅ 4. HTML Plugins**

HTML allows embedding third-party plugins like Flash, Java Applets, and PDFs using <object> or <embed> (not widely supported anymore).

### **Example – Embed a PDF:**

<object data="file.pdf" type="application/pdf" width="100%" height="500px"></object>

### **✅ Embed an Image**

<object data="image.jpg" type="image/jpeg"></object>

### **✅ Use <embed> for Browser-Supported Plugins**

<embed src="file.swf" width="300" height="200">

## **✅ 5. Supported Media Formats**

### **🎥 Supported Video Formats**

|  |  |  |  |
| --- | --- | --- | --- |
| **Format** | **Extension** | **Description** | **Supported by HTML** |
| **MP4** | .mp4 | Best for modern web browsers | ✅ |
| **WebM** | .webm | Open-source, widely supported | ✅ |
| **Ogg** | .ogg | Open-source format | ✅ |
| **AVI** | .avi | Windows-based, not supported in browsers | ❌ |
| **WMV** | .wmv | Windows-based, not supported in browsers | ❌ |
| **MOV** | .mov | Apple-based, not supported in browsers | ❌ |

### **🎧 Supported Audio Formats**

|  |  |  |  |
| --- | --- | --- | --- |
| **Format** | **Extension** | **Description** | **Supported by HTML** |
| **MP3** | .mp3 | Best for music and small file size | ✅ |
| **WAV** | .wav | High-quality audio, larger file size | ✅ |
| **Ogg** | .ogg | Open-source format | ✅ |
| **WMA** | .wma | Windows-based | ❌ |
| **AAC** | .aac | Apple-based, not supported in browsers | ❌ |

## **✅ 6. Media Attributes and Events**

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| **controls** | Shows built-in controls (play, pause, volume) |
| **autoplay** | Starts playing the media automatically |
| **loop** | Repeats the media after it ends |
| **muted** | Starts the media with the sound muted |
| **poster** | Sets an image to show before video plays |
| **preload** | Loads the media file before it is played |
| **width/height** | Sets the display size of the video or audio player |

|  |  |
| --- | --- |
| **Event** | **Description** |
| **play** | Triggered when playback starts |
| **pause** | Triggered when playback is paused |
| **ended** | Triggered when playback ends |
| **volumechange** | Triggered when volume is changed |
| **timeupdate** | Triggered when playback position changes |

## **🚀 Best Practices**

✅ Use **MP4** for video and **MP3** for audio for maximum browser support.  
 ✅ Always provide **multiple formats** in <source> for compatibility.  
 ✅ Use **YouTube** for complex video hosting to reduce load time.  
 ✅ Include controls to give users control over playback.  
 ✅ Avoid autoplay unless muted to avoid annoying users.

## **HTML Geolocation and Related APIs**

## **✅ 1. HTML Geolocation**

The **Geolocation API** allows a website to access the user's geographical location (latitude and longitude).

### **Example – Get User's Location:**

<p id="demo"></p>

<button onclick="getLocation()">Get Location</button>

<script>

var x = document.getElementById("demo");

function getLocation() {

if (navigator.geolocation) {

navigator.geolocation.getCurrentPosition(showPosition);

} else {

x.innerHTML = "Geolocation is not supported by this browser.";

}

}

function showPosition(position) {

x.innerHTML = "Latitude: " + position.coords.latitude +

"<br>Longitude: " + position.coords.longitude;

}

</script>

### **🔎 Explanation:**

* navigator.geolocation.getCurrentPosition() → Gets the current location.
* showPosition() → Displays latitude and longitude.
* Works only over **HTTPS** for security reasons.
* User must grant **permission** to access location.

### **✅ Error Handling**

Use a second parameter to handle errors.

<script>

function getLocation() {

if (navigator.geolocation) {

navigator.geolocation.getCurrentPosition(showPosition, showError);

} else {

x.innerHTML = "Geolocation is not supported by this browser.";

}

}

function showError(error) {

switch(error.code) {

case error.PERMISSION\_DENIED:

x.innerHTML = "User denied the request for Geolocation.";

break;

case error.POSITION\_UNAVAILABLE:

x.innerHTML = "Location information is unavailable.";

break;

case error.TIMEOUT:

x.innerHTML = "The request to get user location timed out.";

break;

case error.UNKNOWN\_ERROR:

x.innerHTML = "An unknown error occurred.";

break;

}

}

</script>

|  |  |
| --- | --- |
| **Error Code** | **Meaning** |
| PERMISSION\_DENIED | User denied the location request. |
| POSITION\_UNAVAILABLE | Location information is unavailable. |
| TIMEOUT | Request to get location timed out. |
| UNKNOWN\_ERROR | An unknown error occurred. |

### **✅ Display Location on Google Maps**

You can use Google Maps API to display the location on a map.

<p id="demo"></p>

<div id="mapholder"></div>

<script>

function showPosition(position) {

var latlon = position.coords.latitude + "," + position.coords.longitude;

var img\_url = "https://maps.googleapis.com/maps/api/staticmap?center=" + latlon +

"&zoom=14&size=400x300&sensor=false&key=YOUR\_API\_KEY";

document.getElementById("mapholder").innerHTML = "<img src='" + img\_url + "'>";

}

</script>

### **✅ Track User Location Continuously**

Use watchPosition() to update the location continuously:

<script>

function getLocation() {

if (navigator.geolocation) {

navigator.geolocation.watchPosition(showPosition);

}

}

function showPosition(position) {

console.log("Latitude: " + position.coords.latitude +

" Longitude: " + position.coords.longitude);

}

</script>

* watchPosition() → Tracks location as the user moves.
* clearWatch() → Stops tracking.

### **✅ Data Returned by Geolocation API**

|  |  |
| --- | --- |
| **Property** | **Description** |
| coords.latitude | Latitude (decimal) |
| coords.longitude | Longitude (decimal) |
| coords.accuracy | Accuracy in meters |
| coords.altitude | Altitude in meters (if available) |
| coords.heading | Direction of movement in degrees |
| coords.speed | Speed in meters/second |
| timestamp | Time of position retrieval |

## **✅ 2. HTML Drag and Drop**

The **Drag and Drop API** allows you to make HTML elements draggable.

### **Example – Drag and Drop:**

<p>Drag the image into the box:</p>

<div id="box" ondrop="drop(event)" ondragover="allowDrop(event)"

style="width:200px;height:100px;border:1px solid black;"></div>

<img id="drag" src="example.jpg" draggable="true"

ondragstart="drag(event)" width="100" height="50">

<script>

function allowDrop(event) {

event.preventDefault();

}

function drag(event) {

event.dataTransfer.setData("text", event.target.id);

}

function drop(event) {

event.preventDefault();

var data = event.dataTransfer.getData("text");

event.target.appendChild(document.getElementById(data));

}

</script>

### **🔎 Explanation:**

* draggable="true" → Makes the element draggable.
* dragstart → Fires when dragging starts.
* drop → Fires when the element is dropped.
* allowDrop() → Prevents the default handling of the element.

## **✅ 3. HTML Web Storage**

The **Web Storage API** allows storing data in the browser (per domain).

* localStorage → Persistent storage (data stays after closing the browser).
* sessionStorage → Temporary storage (data clears when the tab is closed).

### **Example – Save and Retrieve Data:**

<button onclick="saveData()">Save</button>

<button onclick="getData()">Get</button>

<p id="result"></p>

<script>

function saveData() {

localStorage.setItem("name", "John Doe");

}

function getData() {

document.getElementById("result").innerHTML = localStorage.getItem("name");

}

</script>

### **🔎 Explanation:**

* localStorage.setItem() → Stores data.
* localStorage.getItem() → Retrieves data.
* localStorage.removeItem() → Removes data.
* localStorage.clear() → Clears all data.

## **✅ 4. HTML Web Workers**

Web Workers run JavaScript in the **background** without blocking the main thread.

### **Example – Start a Worker:**

**worker.js**:

onmessage = function(e) {

let result = e.data \* 2;

postMessage(result);

}

**Main File**:

<script>

let worker = new Worker("worker.js");

worker.postMessage(10);

worker.onmessage = function(e) {

console.log('Result:', e.data);

};

</script>

### **🔎 Explanation:**

* new Worker() → Creates a new worker.
* postMessage() → Sends data to the worker.
* onmessage() → Captures the data returned from the worker.

## **✅ 5. HTML SSE (Server-Sent Events)**

SSE allows a server to push data to a webpage in real-time.

### **Example – Receive Updates:**

**Server (PHP Example):**

<?php

header('Content-Type: text/event-stream');

header('Cache-Control: no-cache');

echo "data: " . date('H:i:s') . "\n\n";

flush();

**Client:**

<script>

const source = new EventSource("server.php");

source.onmessage = function(event) {

console.log(event.data);

};

</script>

### **🔎 Explanation:**

* EventSource → Listens for server events.
* onmessage → Handles incoming data.
* Requires a server that can send updates (e.g., PHP, Node.js).

## **🚀 Best Practices**

✅ Use **HTTPS** for geolocation and secure APIs.  
 ✅ Provide **error handling** for geolocation failures.  
 ✅ Use **localStorage** for persistent data and **sessionStorage** for temporary data.  
 ✅ Offload heavy processing to **Web Workers** to keep the UI responsive.  
 ✅ Use **SSE** for real-time updates.  
 ✅ Avoid autoplay and intrusive notifications for a better user experience.

Reference:https://github.com/misterola/W3SCHOOLS-HTML-NOTES