



NOTES

Modern HTML



Modern HTML

Introduction to HTML5

HTML was initially introduced in late 1991, and since then HTML has undergone many changes. The first version of HTML was written by Tim Berners-Lee in 1993. Since then, there have been many different versions of **HTML**. The most widely used version throughout the 2000s was **HTML 4.01**, which became an official standard in December 1999.

In 2014 **HTML 5** was published as a W3C recommendation, with the formation of the Web Hypertext Application Technology Working Group (**WHATWG**). The goal of the WHATWG was to create a new version of HTML that would meet the needs of modern web applications.

Features of HTML5

Some of the main features of HTML5 are as follows -

- **Improved Multimedia support** - HTML5 allows for integrating multimedia elements such as **<audio>** and **<video>** directly into web pages without the need for plugins like Flash.
- **Canvas element** - HTML5 introduced the **<canvas>** element, allowing dynamic, interactive graphics to be created and manipulated within a web page.
- **Geolocation API** is an API for obtaining the user's location, enabling web applications to offer location-based services.

Example: Get the user's current location.

JavaScript

```
<body>
  <script>
    if ("geolocation" in navigator) {
      navigator.geolocation.getCurrentPosition((position) => {
        console.log(
          `User current coordinates la=${position.coords.latitude},
lo=${position.coords.longitude}`
        );
      });
    } else {
      console.log("geolocation is not available");
    }
  </script>
</body>
```

One popup will appear, asking the browser to know your location.

Browser Output

User current coordinates la=13.0541276, lo=77.7566411
>

- **Local Storage** - HTML5 introduced the localStorage API, which allows for the storage of data on the user's device, improving performance and reducing the need for round-trips to the server.
- **New Structural Elements** - HTML5 introduced new semantic elements such as **<header>**, **<footer>**, **<nav>**, and **<article>**, etc. that make it easier to structure and organize content on a web page.

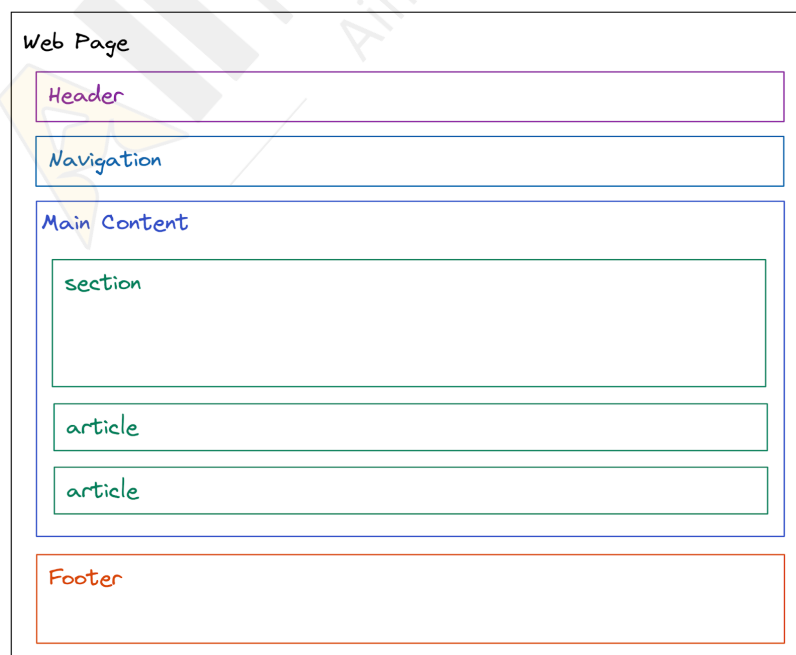
- **Form Improvements** - HTML5 introduced several improvements to forms, including new input types like **date**, **time**, and **color**, as well as new attributes such as **required** and **autofocus**.
- **Accessibility improvements** - HTML5 provides better support for accessibility, including the ability to provide alternative text for images and improved support for screen readers.

Introduction to semantic tags

Semantic tags are HTML tags that give **meaning to the content** of a web page beyond just its presentation. They provide information about the structure and meaning of the content, making it easier for **search engines, screen readers**, and other tools to understand the page and its content.

To understand semantics, we will build a web page using newly introduced structural-semantic elements in HTML.

See below web page structure,



Unlike traditional HTML tags, semantic tags focus on the meaning of the content, rather than on the visual appearance. Like in the above example, the **<header>** tag is a semantic tag indicating that the header section of a page begins, while the **<article>** tag indicates that the enclosed content is an **independent piece of content** that can stand alone, and so on.

Let's code a web page, using all common semantic tags to define a common web page structure.

JavaScript

```
<body>
  <!-- Header -->
  <header>
    <h1>Semantic HTML Tags</h1>
    <nav>
      <ul>
        <li><a href="#">Home</a></li>
        <li><a href="#">About Us</a></li>
        <li><a href="#">Services</a></li>
        <li><a href="#">Contact Us</a></li>
      </ul>
    </nav>
  </header>
  <!-- Main content -->
  <main>
    <!-- Article -->
    <article>
      <h2>What are Semantic HTML Tags</h2>

      <p>
        Semantic HTML tags are tags that give meaning to the content they
        surround. They provide information about the structure and meaning of the
        content on a web page, making it easier for search engines, screen readers,
        and other tools to understand the page and its content.
      </p>
    </article>

    <!-- Section -->
    <section>
      <h2>Semantic Tags</h2>
      <ul>
        <li>header: Indicates the beginning of a header section</li>
        <li>nav: Defines a section of navigation links</li>
```

```

        <li>
            article: Indicates an independent, self-contained piece of
content
        </li>
        <li>
            footer: Defines a footer section at the end of a page or article
        </li>
    </ul>
</section>
</main>

<!-- Footer -->
<footer>
<p>&copy; 2024 Example Company. All rights reserved.</p>
</footer>
</body>

```

Output:

Semantic HTML Tags

- [Home](#)
- [About Us](#)
- [Services](#)
- [Contact Us](#)

What are Semantic HTML Tags?

Semantic HTML tags are tags that give meaning to the content they surround. They are tools to understand the page and its content.

Semantic Tags

- header: Indicates the beginning of a header section
- nav: Defines a section of navigation links
- article: Indicates an independent, self-contained piece of content
- footer: Defines a footer section at the end of a page or article

© 2024 Example Company. All rights reserved.

In the above example, we've used many semantic tags, such as **<header>**, **<nav>**, **<main>**, **<article>**, **<footer>**, **<h1>**, **<p>** to

provide additional information about the structure and meaning of the content on the page.

These tags make the code more readable and easier to understand, both for humans and for web browsers and other tools that might be parsing the HTML.

In the below table, we have shown some semantic tags in HTML with their semantics.

Sr.no	Tags	Semantics
1	<aside>	Defines content aside from the page content.
2	<details>	Defines additional details that the user can view or hide
3	<figure>	Specifies self-contained content, like illustrations, Diagrams, photos, code listings, etc
4	<footer>	Defines a footer for a document or section
5	<header>	Specifies a header for a document or section
6	<main>	Specifies the main content of document
7	<section>	Defines a section in document
8	<nav>	Defines navigation links
9	<article>	Defines an article
10	<h1>	Defines a Heading
11	<p>	Defines a paragraph

Let's take an another example to understand better:

JavaScript

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Tech Job Listings</title>
  </head>
  <body>
    <!-- Header section contains the main heading and navigation -->
    <header>
      <h1>TechCareer Hub</h1>
      <!-- Navigation menu -->
      <nav>
        <ul>
          <li><a href="#home">Home</a></li>
          <li><a href="#jobs">Jobs</a></li>
          <li><a href="#companies">Companies</a></li>
          <li><a href="#resources">Resources</a></li>
        </ul>
      </nav>
    </header>

    <!-- Main content area -->
    <main>
      <!-- Section for featured job listings -->
      <section id="featured-jobs">
        <h2>Featured Job Listings</h2>
        <!-- Each job listing is wrapped in an article tag -->
        <article class="job-listing">
          <h3>Senior Software Engineer</h3>
          <!-- Using itemprop for structured data -->
          <p>
            <strong>Company:</strong>
            <span itemprop="hiringOrganization">TechCorp Inc.</span>
          </p>
          <p>
            <strong>Location:</strong>
            <span itemprop="jobLocation">San Francisco, CA</span>
          </p>
          <p>
            <strong>Salary:</strong>
            <span itemprop="baseSalary">$120,000 - $160,000</span>
          </p>
          <!-- Details and summary for expandable content -->
```



```

<details>
  <summary>Job Description</summary>
  <p>
    We are seeking an experienced software engineer to join our
    dynamic team...
  </p>
</details>
<!-- Time tag for machine-readable dates -->
<time datetime="2024-07-15">Posted on: July 15, 2024</time>
<a href="#apply" class="cta-button">Apply Now</a>
</article>

<!-- Second job listing article -->
<article class="job-listing">
  <!-- Structure similar to the first job listing -->
  <!-- ... (content remains the same) ... -->
</article>
</section>

<!-- Aside for related but not main content -->
<aside>
  <h2>Job Search Tips</h2>
  <ul>
    <li>Tailor your resume to each job application</li>
    <li>Research the company before applying</li>
    <li>Prepare for common interview questions</li>
  </ul>
</aside>
</main>

<!-- Footer section -->
<footer>
  <p>&copy; 2024 campncorp. All rights reserved.</p>
  <!-- Address tag for contact information -->
  <address>
    Contact us at:
    <a href="mailto:info@aimerz.ai">info@aimerz.ai</a>
  </address>
</footer>
</body>
</html>

```

TechCareer Hub

- [Home](#)
- [Jobs](#)
- [Companies](#)
- [Resources](#)

Featured Job Listings

Senior Software Engineer

Company: TechCorp Inc.

Location: San Francisco, CA

Salary: \$120,000 - \$160,000

► Job Description

Posted on: July 15, 2024 [Apply Now](#)

Job Search Tips

- Tailor your resume to each job application
- Research the company before applying
- Prepare for common interview questions

© 2024 campncorp. All rights reserved.

Contact us at: info@aimerz.ai

Benefits of using semantic tags

- By using Semantic tags in our code, we can **provide additional information** about that document by defining the layout and sections of the webpage.
- Semantic elements are of great help to people using screen readers. The additional information provided by semantic tags **helps screen readers** understand the content better and help them to determine the different sections within a page more efficiently.
- HTML Semantic tags help the browser determine the purpose of the page and its content. Semantic tags also help in **Search Engine Optimization** as they help browsers interpret the content more easily by making content more adaptive.

Introduction to HTML entities

- The reserved characters used in HTML documents are known as HTML entities. They are not seen on the typical keyboard. They provide a wide range of characters, allowing you to add icons, geometric shapes, mathematical operators, and so on.
- HTML entities are useful when you want to display special characters that may conflict with HTML syntax or when you want to ensure proper rendering across different platforms and browsers that may have different character encoding settings.
- It's important to note that with the advancement of web technologies, the need for using HTML entities has diminished to some extent.

Let us discuss more about the HTML entities

Purpose & Usage

The main two purposes are:

1. Displaying special characters
2. Handling reserved characters

Displaying special characters:

HTML entities are used to display characters that have special meanings in HTML or characters that cannot be easily typed or displayed directly.

For example, if you want to display the copyright symbol (©) or the trademark symbol (™) on a webpage, you would use the corresponding HTML entities "©" and "™", respectively.

Handling Reserved characters

HTML uses certain characters for its syntax and structure, such as the less-than symbol (<), the greater-than symbol (>), and the ampersand symbol (&). These reserved characters can cause issues if used directly in HTML code, as they may be interpreted as part of the markup.

By using HTML entities like "<", ">", and "&", you can display these characters as text on a webpage without affecting the HTML structure.

The entity code is enclosed in ampersands (&) and semicolons (;) to differentiate it from regular text.

```
JavaScript
&entity_name;
// OR
```

```
&#entity_number;
```

For example, to display the less-than symbol (<), you would use "<". The browser then interprets the HTML entity and renders the appropriate character on the webpage.

HTML entities are especially useful when working with characters that are not available on the keyboard or when you want to ensure consistent rendering of characters across different platforms and browsers. They help in maintaining the integrity of HTML code and displaying special characters accurately in HTML documents.

Character Reference chart - [Link](#)

Commonly used HTML entities

The reserved characters used in HTML documents are known as HTML entities. They are not seen on the typical keyboard. They provide a wide range of characters, allowing you to add icons, geometric shapes, mathematical operators, and so on.

HTML entities are useful when you want to display special characters that may conflict with HTML syntax or when you want to ensure proper rendering across different platforms and browsers that may have different character encoding settings.

It's important to note that with the advancement of web technologies, the need for using HTML entities has diminished to some extent.

Let us discuss more about the HTML entities

Result	Description	Entity Name	Entity Number
<	Less than	<	<
>	Greater than	>	>
&	Ampersand	&	&
"	Double quotation mark	"	"
'	Single quotation mark	'	'
©	Copyright	©	©
®	Registered	®	®

Numeric Character References vs. Named Character Entities

Numeric Character References represent characters using their Unicode (code point) in decimal or hexadecimal format.

Ex:

JavaScript

```
&#60; // for less than (<)
&#62; // for greater than (>)
&#38; // for ampersand (&)
```

Named Character Entities represent characters using predefined names that correspond to specific characters.

Ex:

JavaScript

```
&lt; // for less than (<)
&gt; // for greater than (>)
&amp; // for ampersand (&)
```

ASCII character set and Unicode

ASCII

The ASCII (American Standard Code for Information Interchange) was one of the first encoding standards to offer a truly universal character set for basic electronic communications. It is a widely used character encoding standard that represents characters using 7 bits and 8 bits, allowing for a total of 128 and 256 characters respectively. The ASCII character set indeed uses 8 bits to represent a total of 256 characters. Originally, ASCII was defined using 7 bits, allowing for 128 characters. However, an extended version of ASCII known as "extended ASCII" or "ASCII-8" uses the 8th bit to represent an additional 128 characters, bringing the total to 256.

ASCII includes basic Latin letters (A-Z, a-z), digits (0-9), punctuation marks, control characters, and some special characters.

Link to ASCII characters reference table: [Link](#)

Unicode

Unicode is a character encoding standard that aims to represent almost every character from every writing system in the world including over 149,000 characters. Unicode provides a unique numerical value (code point) for every character including punctuation marks, mathematical symbols, technical symbols, arrows, emojis, and various scripts, including Latin, Greek, Cyrillic, Chinese, Japanese, Korean, Arabic, Hebrew, and many more, allowing for a much larger range of characters to be represented compared to ASCII. Since its inception, Unicode has been adopted by all modern software providers, allowing the transportation of data through devices, applications, and platforms without corruption.

It is now used in all major operating systems, browsers, search engines, laptops, smartphones, and across the internet.

The Unicode standard is maintained by the Unicode Consortium, a non-profit organization that exists to develop and promote the Unicode Standard.

The Unicode Standard supports three encoding forms: UTF-8, UTF-16, and UTF-32.

What is UTF?

UTF stands for "Unicode Transformation Format". Unicode can be implemented by different character sets.

The most commonly used encodings are UTF-8 and UTF-16. The default character encoding in HTML- 5 is UTF-8, HTML 4 supports UTF-8. HTML 5 supports both UTF-8 and UTF-16.

What is the difference between Unicode and UTF-8?

Unicode is a character set while UTF-8 IS encoding.

Unicode is basically a list of characters with unique decimal numbers (code point)

Let's look at how the code point of Alphabets

A = 65

B = 66

C = 67

D = 68 ...etc

Encoding is how these numbers are translated into binary numbers which the computer can understand and store.

Example

Let's convert **"hey"** to Unicode

hey = 104 101 121

Now let's convert the **"hey"** to UTF-8 encoding

hey = 01101000 01100101 01111001

So we can say that Encoding converts numbers into binary. Character sets convert characters to numbers.

Special text tags (<kbd>, <code>)

Let's discuss about some special text tags in HTML.

• < kbd>

The <kbd> tag is used to indicate user input, typically representing keyboard input or commands.

By default, the content wrapped in the < kbd> tags is displayed in a monospace font

Ex:

JavaScript

```
<p>
  press
  <kbd>Ctrl</kbd> + <kbd>S</kbd> to save a file or document
</p>
```

Output

press Ctrl + S to save a file or document

In the above example output, you can notice that the Ctrl and S is in different font

• <code>

The <code> tag is used to represent code snippets of code within an HTML document.

By default, the content wrapped in the <code> tags is displayed in a monospace font.

JavaScript

```
<code>
let name = Prabir; <br />
let age = 11;
</ code>
```

```
let name = Prabir;
```

```
let age = 11;
```

You can see that the change in the font and the readability of the code also increases

Also there are many special text tags present.

Accessibility in HTML

What is accessibility

Accessibility in HTML refers to the practice of designing and coding web content in a way that ensures it can be easily accessed, understood, and interacted with by people of diverse abilities, including those with disabilities. This includes using semantic markup, providing alternative text for images and videos, and making sure your web pages are navigable using a keyboard.

There are some assistive devices that play a major role in providing accessibility.

1. **Screen Reader:** A screen reader is software that reads out loud the content of a web page to individuals who are visually impaired. It can also interpret and communicate

information about graphics, multimedia, and other elements on the page.

2. **Voice recognition software:** Voice recognition software enables users to navigate web pages and input text using voice commands. This technology is particularly useful for individuals with mobility impairments or those who have difficulty using a keyboard or mouse.
3. **Keyboard alternatives:** Keyboard alternatives such as sip-and-puff devices, head-tracking devices, and eye-tracking devices allow individuals with physical disabilities to navigate and interact with web pages without the use of a traditional keyboard or mouse.

Why to use accessibility

There are many reasons why you should use accessibility. Here are some of the important points

- To identify accessibility issues: - It can help you to identify accessibility issues in your website or web application before users with disabilities encounter them.
- To meet accessibility standards - it can help you to ensure that our website or web application meets accessibility standards.
- To improve usability - Accessibility automated testing software tools can help us to improve the usability of your website or web application for everyone, not just users with disabilities, as a result, it can help you to increase user satisfaction and engagement.

Best practices to follow for accessibility

Understanding Accessibility Guidelines and familiarisation of Web Content Accessibility Guidelines (WCAG), that provide the international standard for web accessibility. Understanding the different levels of standard rules and the principles, guidelines, and success criteria outlined in the guidelines of best practices to follow for accessibility.

Following the guidelines of the WCAG includes the best practices to follow for accessibility.

Some of the best practices are as follows -

1. Image alternative text - Add descriptive alternative text (alt text) to all images using the alt attribute

Example -

JavaScript

```

```

Browser output



If the above code is not working due to connectivity issues or a mistake in the URL link or Path. The alternate text will be displayed.



Random Pic

2. Ensuring proper heading structure or Semantic Structure of HTML- practice the use of proper semantic structure, by using appropriate HTML elements like **<h1>**, **<nav>**, **<main>**, **<article>**, **<button>**, and **<label>** to convey the purpose and role of each element.

Example of HTML with proper semantic structure.

JavaScript

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Delicious Chocolate Chip Cookies</title>
  </head>
  <body>

    <!-- Heading section -->
    <header>
      <h1>Delicious Chocolate Chip Cookies</h1>
    </header>

    <!-- navbar section -->
    <nav>
      <ul>
        <li><a href="#ingredients">Ingredients</a></li>
        <li><a href="#instructions">Instructions</a></li>
      </ul>
    </nav>

    <!-- Main section -->
    <main>
      <article id="ingredients">
        <h2>Ingredients</h2>
        <p>Lorem ipsum dolor sit.....</p>
      </article>

      <article id="instructions">
        <h2>Instructions</h2>
        <p>Lorem ipsum dolor sit sciunt distinctio unde....</p>
      </article>
    </main>

    <!-- Footer section -->
    <footer>
      <p>&copy; 2023 Delicious Cookie Recipes</p>
    </footer>
  </body>
</html>
```

3. Making links descriptive - to make a link descriptive in HTML, meaning full and descriptive text in the anchor **<a>** element should be used.

Example -

JavaScript

```
<p>
  Check out our latest blog post about
  <a
    href="https://www.example.com/latest-blog"
    title="Read the latest blog post" >web-development</a>
</p>
```

4. Accessibility in Form - use of **<label>** tag or **aria-label** attribute and Form fields should be properly grouped with **<fieldset>** and **<legend>** elements.

Example

JavaScript

```
<form action="/">
  <fieldset>
    <legend>Personal Form Details</legend>
    <label for="fname">First name:</label>
    <input type="text" id="fname" name="fname" /><br /><br />
    <label for="lname">Last name:</label>
    <input type="text" id="lname" name="lname" /><br /><br />
    <label for="email">Email:</label>
    <input type="email" id="email" name="email" /><br /><br />
    <label for="birthday">Birthday:</label>
    <input type="date" id="birthday" name="birthday" /><br /><br />
    <input type="submit" value="Submit" aria-label="Submit form" />
  </fieldset>
</form>
```


Browser output -

Personal Form Details

First name:

Last name:

Email:

Birthday: 

5. Text resizes and Zoom testing - The website pages should be checked by resizing text or zooming in/out to ensure that the content remains readable and usable without causing layout or functionality issues.
6. Video and Audio Accessibility - Provide captions, transcripts, and audio descriptions for multimedia content. Ensure that video players are accessible and controllable with the keyboard.
7. HTML tables - Ensure that tables are used for tabular data and properly marked up with appropriate headers and scope attributes to provide context to screen readers.

Example -

JavaScript

```
<table>
  <caption>
    Monthly Employee Attendance
  </caption>
  <thead>
    <tr>
      <th scope="col">Month</th>
```



```

        <th scope="col">Product A</th>
        <th scope="col">Product B</th>
        <th scope="col">Product C</th>
    </tr>
</thead>
<tbody>
    <tr>
        <th scope="row">January</th>
        <td>19 days</td>
        <td>24 days</td>
        <td>19 days</td>
    </tr>
    <tr>
        <th scope="row">February</th>
        <td>24 days</td>
        <td>21 days</td>
        <td>19 days</td>
    </tr>
    <!-- More rows go here -->
</tbody>
</table>

```

Browser output -

Monthly Employee Attendance			
Month	Product A	Product B	Product C
January	19 days	24 days	19 days
February	24 days	21 days	19 days

8. Colour contrast - Verify that the colour contrast between text and background meets the WCAG contrast requirements. Ensure that information conveyed through colour is also available via other means (e.g., labels, symbols).
9. Tab-index - The tab-index is an HTML attribute that can be used to specify the order of navigating an HTML element when the tab button is pressed. The tab-index can be used on any HTML element but it should be used only on specific elements where it is needed.

It's best not to use the **"tabindex"** attribute on non-interactive elements to make them work like interactive elements when users use the keyboard, For example, don't use a **<div>** to represent a button when you should use the **<button>** element

It accepts an integer as a value, with different results depending on the integer's value.

- `tabindex="-1,"` indicates that the element cannot be accessed using standard keyboard navigation in a sequential manner.
- `tabindex="0"` makes the element focusable through keyboard navigation, following any elements with positive **tabindex** values
- A positive **tabindex** value determines the order of keyboard focus. Lower values are focused before higher ones. When elements share the same positive **tabindex**, their order follows the document source.

Recommend using only "0" and "-1" as **tabindex** values to maintain a logical and accessible tab order. Avoid using

values greater than 0 or CSS properties that alter the order of focusable elements, as it can hinder keyboard navigation and accessibility for users who rely on it.

Example

Index.html (Basic example)

JavaScript

```
<body>
  <div tabindex="0">
    Click here!
  </div>
  <div tabindex="0" >
    Click right here!
  </div>
  <div tabindex="0" >
    Submit here
  </div>
</body>
```

From the above example, **<div>** buttons have the ability to be focused (including via tab) by giving each on the attribute `tabindex= "0"`. Basically, the `tabindex` attribute is primarily intended to allow tabbable elements to have a custom tab order.

More example of the `tabindex`
index.html

JavaScript

```
<body>
  <div>Not in the tab order</div>
  <div tabindex="0">In the tab order</div>
  <div tabindex="-1">Not in the tab order (but is
focusable)</div>
  <div tabindex="5">Ahead of everything in the tab
order</div>
```

```
<button>Totally focusable</button>
</body>
```

Browser output

Not in the tab order

In the tab order

Not in the tab order (but is focusable)

Ahead of everything in the tab order

Totally focusable

-

In the above example, the **tabindex="0"** option allows elements that are not usually able to be focused via the keyboard to become focusable. This value of the tab index is particularly beneficial. The **tabindex="-1"** option enables elements that are not typically focusable to receive focus programmatically, such as through JavaScript or as the target of links. The **tabindex="5"** moves the order ahead of everything. Additionally, the button is focusable by default.

What is SEO

SEO stands for Search Engine Optimization. It helps you optimize your websites so that they have better rankings on Search Engines like Google, Bing and Yahoo.

The ultimate goal of SEO is to rank higher in search engine results pages for relevant keywords and phrases, which can increase visibility, traffic, and ultimately revenue for the website.

It is essential for businesses that want to succeed in the digital age. In HTML, a common way to help search engines is to include document titles and some important meta tags.

Meta tags are important for SEO because they provide information to search engines about the content on your site and can help improve your website's visibility and ranking in search engine results pages

Introduction to head tag

The <head> tag in HTML is indeed an element that contains important information and metadata about the

document. It is placed between the <html> tag and the <body> tag. The

<head> tag typically includes various elements that provide additional information and instructions to the web browser and search engines.

The head tag in HTML can consist of a meta tag, title tag, script tag, and so on.

JavaScript

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>

  </body>
</html>
```

Tags used inside the head tag(title,link,style,meta)

The title, link, style, and meta are some of the tags that are commonly used inside the head tag.

Let's understand them (title, link, style, and meta) in detail

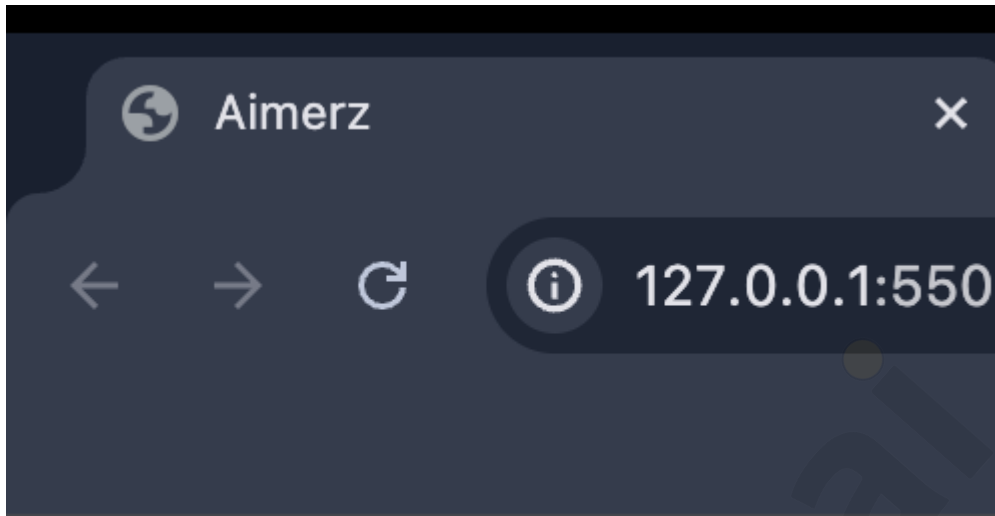
The title tag "<title>": The <title> tag specifies the title of the document, which is displayed as the page title in the browser's title bar or tab. It helps users identify the webpage and search engines understand the page's content

JavaScript

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Aimerz</title>
  </head>
  <body>

  </body>
</html>
```

This title appears in the Browser Tab as shown below:



The <link> tag: The <link> tag is used to reference external resources such as CSS stylesheets, icon files(favicon), or alternative versions of the page for different devices or languages

JavaScript

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

style tag(<style>) : The style tag is used to define internal CSS styles that are applied to the document. It allows us to specify the custom styling rules for elements on the page

JavaScript

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Aimerz</title>
    <style>
      body{
        background-color: #f0f0f0;
      }
    </style>
  </head>
  <body>
  </body>
</html>
```

```

    </style>
  </head>
  <body></body>
</html>

```

Introduction to Meta Tags

Meta tag < meta > in HTML is used to provide information about a web page to search engines, browsers, and other web services.

Usage of Meta tags

Some of the uses of meta tags are as follows -

- **charset** - This Meta tag specifies the characters encoding for the document.

Example meta tag (charset)

JavaScript

```
<meta charset="UTF-8" />
```

- **viewport** - This viewport in meta tag helps in making the web pages responsive on different devices by setting the viewport width to the device's width and ensuring the initial zoom is 1.0

JavaScript

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

- **description**: This meta tag Provides a brief description of the page's content, often used by search engines to display a snippet in search results.

JavaScript

```
<meta name="description" content="Write something about the website">
```

- **author:** This meta tag indicates the author of the web page.

JavaScript

```
<meta name="author" content=" Enter website author name">
```

- **keywords:** This meta tag specifies a list of keywords or phrases that are relevant to the page's content. However, major search engines like Google no longer consider this meta tag for ranking purposes.

JavaScript

```
<meta name="keywords" content="keyword1, keyword2, keyword3" >
```

- **refresh** - This meta tag redirects the user to another URL or refreshes the page itself after a specific time

JavaScript

```
<meta http-equiv="refresh" content="5;URL=https://example.com/">
```

Benefits of using meta tag

Using meta tags in HTML documents provides several benefits for web developers, search engines, and users.

Some of the key benefits include

- **SEO (Search Engine Optimization):** Meta tags, such as the "description" and "keywords" tags, can help improve a webpage's visibility on search engine result pages (SERPs). A well-crafted meta description can attract users to click on the link, while the keywords tag, although less impactful than in the past,

can still provide search engines with relevant information about the page's content.

- **Social Media Sharing:** Meta tags like the Open Graph Protocol tags allow developers to control how web pages are displayed when shared on social media platforms like Facebook, Twitter, and LinkedIn. They define the title, description, and image that appear in social media posts, making shared links more attractive and **informative**.
- **Character Encoding and Language:** The meta tag specifying the character encoding (e.g., UTF-8) ensures that the browser interprets the text correctly, especially when dealing with special characters or multilingual content.
- **Viewport Control:** The viewport meta tag is crucial for responsive web design. It allows developers to set the initial scale and width of the viewport, making sure the web page adapts well to different screen sizes and resolutions.
- **HTTP-Equiv Tags:** Meta tags with http-equiv attribute (e.g., `< meta http-equiv="refresh">`) enable specific HTTP header functionalities, like refreshing or redirecting the page after a certain time. While overusing them can be detrimental, there are valid use cases for handling page behavior.
- **Web Accessibility:** Certain meta tags can improve web accessibility by providing additional information to assistive technologies, making it easier for users with disabilities to navigate and understand the content.

Favicon

A favicon (favourite icon) is a tiny icon included along with a website, which is displayed in places like the browser's address bar, page tabs and bookmarks menu.

Usually, a favicon is **16 x 16 pixels** in size and stored in the GIF, PNG, or ICO file format.

They are used to improve user experience and enforce brand consistency. When a familiar icon is seen in the browser's address bar, for example, it helps users know they are in the right place.

Example of meta tags

1. **Create a favicon image:** Create a favicon image using some photo editor software or an online favicon generator like "https://favicon.io/".
2. **Save the Image:** name the image with the desired name and save it with the extension of ico png or gif.

Ex: favicon.ico or favicon.png or favicon.gif

3. Add the favicon.ico image using the link tag

JavaScript

```
<link rel="icon" href="image/favicon.ico" />
```

THANK YOU

