




# HTML

Owner	 Sonsawan Ngamsom
Verification	
Tags	<a href="#">Codebase</a> <a href="#">Web Structure</a>
Last edited time	@July 5, 2023 8:44 PM

HTML stands for HyperText Markup Language. It is the standard markup language used for creating and structuring web pages and applications on the World Wide Web. HTML uses a set of tags to define the structure and layout of a web document, such as headings, paragraphs, lists, images, links, and more.

HTML files are plain text documents that consist of a series of elements enclosed in angle brackets (< >). These elements are used to define the structure and content of the web page. For example, the `<h1>` tag is used to define a heading, the `<p>` tag is used to define a paragraph, and the `<img>` tag is used to insert an image.

HTML tags can also include attributes, which provide additional information about an element. Attributes are defined within the opening tag and can specify things like the source of an image, the target of a link, or the dimensions of an element.

Web browsers interpret HTML documents and render them as visually appealing web pages. HTML is often used in conjunction with CSS (Cascading Style Sheets) and JavaScript to enhance the appearance and functionality of web pages. CSS is used to define the presentation and layout of the HTML elements, while JavaScript is used to add interactivity and dynamic behavior to web pages.

Overall, HTML is the backbone of the web and is essential for creating and structuring the content of web pages.

## Structure of an HTML Document

### The `<!DOCTYPE html>`

The `<!DOCTYPE html>` declaration is the very first line of an HTML document. It is not an HTML tag but rather an instruction to the web browser about the version of HTML being used in the document.

The `<!DOCTYPE html>` declaration is used in HTML5 to specify that the document is written in the latest version of HTML, which is simply referred to as HTML5. It informs the browser to interpret the document using the rules and specifications of HTML5.

```
<!DOCTYPE html>
```

In Visual Studio Code (VSCode), you can use a shorthand notation to generate the basic structure of an HTML document. By typing `!` and pressing the `Tab` key, VSCode will generate the following code:

```
<!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>
```

Please note that the `!` shorthand is specific to the VSCode editor and is not part of the actual HTML syntax. It is merely a convenient way to generate a starting point for an HTML document within the editor.

## The `<main>` element

The `<html>` element is the root element of an HTML page. It represents the entire HTML document and serves as the container for all other HTML elements. Every HTML document should have a single `<html>` element that wraps around all other elements.

The `<html>` element is usually paired with the closing `</html>` tag to define the beginning and end of the HTML document. All other HTML elements, such as `<head>` and `<body>`, are contained within the `<html>` element.

Here's an example of a basic HTML document structure with the `<html>` element:

```
<!DOCTYPE html>
<html>
<head>
  <!-- meta tags, title, stylesheets, etc. -->
</head>
<body>
  <!-- content of the web page -->
</body>
</html>
```

Within the `<html>` element, you will typically find two main sections:

1. `<head>`: This section contains meta-information about the document, such as the document title, links to external stylesheets or scripts, character encoding declaration ( `<meta charset="UTF-8">` ), and more. The content within the `<head>` section does not appear directly on the web page but provides important information for the browser and search engines.
2. `<body>`: This section contains the visible content of the web page, including headings, paragraphs, images, links, and other elements. The content within the `<body>` section is what users see and interact with when they visit the web page.

The `<html>` element encapsulates the entire HTML document and provides the structure and hierarchy for all the elements within it.

## The `<head>` element

The `<head>` element is a container element in HTML that contains meta-information about the document. It is located inside the `<html>` element but outside the `<body>` element. The content within the `<head>` element is not directly visible on the web page but provides important information and instructions for the browser and search engines.

Here are some common meta-information elements found within the `<head>` element

- `<title>`: The `<title>` element specifies the title of the web page, which is displayed in the browser's title bar or tab. It is also used by search engines when indexing the page.

```
<head>
  <title>My Website</title>
</head>
```

- `<meta charset="UTF-8">`: This meta tag declares the character encoding of the document, ensuring that the browser interprets and displays the text correctly. UTF-8 is a widely used character encoding that supports a broad range of characters from different languages.

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

- **External Stylesheets:** You can link external CSS (Cascading Style Sheets) files within the `<head>` element to define the styles and layout of your web page. The `<link>` element is used for this purpose.

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

- **External Scripts:** If you need to include external JavaScript files or other scripts, you can use the `<script>` element within the `<head>` element.

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

- **Other meta tags:** The `<head>` element can contain additional meta tags that provide information such as the author of the document, a description, keywords, viewport settings for responsive design, and more. These meta tags are not typically displayed directly on the page but are used by search engines, social media platforms, and other tools to understand and process the content.

```
<head>
  <meta name="author" content="John Doe">
  <meta name="description" content="This is my website.">
  <meta name="keywords" content="HTML, CSS, JavaScript">
  <!-- Additional meta tags... -->
</head>
```

These are just a few examples of the types of meta-information that can be included within the `<head>` element. It serves as a container for various elements that provide information about the document and help control its presentation and behavior.

## The `<body>` element

The `<body>` element is a container element in HTML that holds the visible content of a web page. It is located within the `<html>` element and follows the `<head>` element. The content placed within the `<body>` element is what users see and interact with when they visit the web page.

Here are some examples of common elements and content that can be placed within the `<body>` element:

- **Text Content:** Headings, paragraphs, lists, and other textual content are typically placed within the `<body>` element. For instance:

```
<body>
  <h1>Welcome to My Website</h1>
  <p>This is a paragraph of text.</p>
  <ul>
    <li>Item 1</li>
    <li>Item 2</li>
    <li>Item 3</li>
  </ul>
</body>
```

- **Images:** The `<img>` element is used to display images within the `<body>` element. The `src` attribute specifies the source URL or file path of the image. For example:

```
<body>
  <h1>Welcome to My Website</h1>
```

```

</body>
```

- Links: Hyperlinks are created using the `<a>` element within the `<body>` element. The `href` attribute specifies the URL or destination of the link. Here's an example:

```
<body>
  <h1>Welcome to My Website</h1>
  <p>Visit our <a href="https://www.example.com">website</a> for more information.</p>
</body>
```

- Forms: If you want to collect user input, you can use the `<form>` element within the `<body>` element. It contains input fields, checkboxes, radio buttons, and more. Here's a simple form example:

```
<body>
  <h1>Contact Form</h1>
  <form>
    <label for="name">Name:</label>
    <input type="text" id="name" name="name">
    <br>
    <label for="email">Email:</label>
    <input type="email" id="email" name="email">
    <br>
    <input type="submit" value="Submit">
  </form>
</body>
```

These examples demonstrate how the `<body>` element is used to contain the visible content of an HTML document. It holds the text, images, links, forms, and other elements that make up the actual content and structure of the web page.

## HTML Elements and Tags

In HTML, elements are the individual components or parts that make up a web page's structure and content. Each element serves a specific purpose and contributes to the overall layout and functionality of the page. Elements can represent various things such as headings, paragraphs, images, links, forms, tables, and more.

These elements are represented in HTML by using tags. A tag is a set of characters enclosed in angle brackets `<` and `>`. Tags define the beginning and end of an element, creating a container that holds the content associated with that element.

Here's an example to illustrate the concept:

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

In this example, the `<p>` tag represents the paragraph element. The content, which is the text "This is a paragraph element," is placed between the opening `<p>` tag and the closing `</p>` tag. Together, the tags define the paragraph element.

HTML tags can also include additional attributes that provide extra information or specify certain properties of an element. Attributes are written within the opening tag and are typically in the form of name-value pairs. For example:

```

```

In this example, the `<img>` tag represents an image element. The `src` attribute specifies the source URL or file path of the image, while the `alt` attribute provides alternative text that is displayed if the image cannot be loaded.

It's important to note that some HTML elements do not require closing tags because they are self-contained or have no content within them. These are known as self-closing or void elements. Instead of a closing tag, self-closing elements end with a forward slash `/` before the closing angle bracket.

## Some common elements

- **Headings** (`<h1>` to `<h6>`): Headings are used to represent different levels of headings or titles on a web page. HTML provides six levels of headings, from `<h1>` (the highest level) to `<h6>` (the lowest level).
  - **Paragraphs** (`<p>`): The `<p>` element is used to define paragraphs of text. It represents a block of text that is separated from other elements.
  - **Lists** (`<ul>`, `<ol>`, `<li>`): Lists allow you to present items in an ordered (numbered) or unordered (bulleted) format. The `<ul>` element represents an unordered list, the `<ol>` element represents an ordered list, and the `<li>` element represents individual list items.
  - **Links** (`<a>`): The `<a>` element is used to create hyperlinks, allowing users to navigate to other web pages or specific sections within the same page. The `href` attribute specifies the destination URL or target location.
  - **Images** (`<img>`): The `<img>` element is used to insert images into a web page. The `src` attribute specifies the source URL or file path of the image, while the `alt` attribute provides alternative text that is displayed if the image cannot be loaded.
  - **`<div>`**: The `<div>` element is a generic container that is used to group and organize other elements on a web page. It is often used to create sections or divisions of content.
  - **`<span>`**: The `<span>` element is an inline container that is used to apply styling or manipulate specific portions of text within a larger block of content.
  - **`<table>`, `<tr>`, `<td>`**: These elements are used to create tables for displaying tabular data. The `<table>` element represents the table itself, `<tr>` represents a table row, and `<td>` represents a table cell.
  - **`<form>`, `<input>`**: These elements are used to create interactive forms for user input. The `<form>` element represents the form itself, and the `<input>` element represents various form input fields such as text fields, checkboxes, radio buttons, and more.
  - **`<iframe>`**: The `<iframe>` element is used to embed another web page or external content within the current HTML document. It is commonly used to display maps, videos, or other external content.
  - **`<header>`**: The `<header>` element represents the introductory content or a container for the top section of a web page. It often includes site branding, navigation menus, or page-level headings.
  - **`<footer>`**: The `<footer>` element represents the closing content or a container for the bottom section of a web page. It typically includes information like copyright notices, links to terms of service or privacy policy, and contact details.
  - **`<nav>`**: The `<nav>` element is used to define a section containing navigation links. It is commonly placed within the header or footer, but it can be used anywhere on the page.
  - **`<button>`**: The `<button>` element is used to create clickable buttons on a web page. It can be used to trigger actions or submit forms.
- 

## Text Formatting

**The text formatting tags available in HTML. These tags allow you to apply specific formatting styles to your text. Here's an explanation of the mentioned tags:**

- **`<strong>`**: The `<strong>` tag is used to indicate strong importance or emphasis on the enclosed text. By default, the text wrapped in `<strong>` is displayed in a bold font.
- **`<em>`**: The `<em>` tag is used to emphasize or highlight text. By default, the text wrapped in `<em>` is displayed in italics.
- **`<u>`**: The `<u>` tag is used to underline text. It is typically used to indicate a hyperlink, but it can also be used for other purposes.
- **`<s>`**: The `<s>` tag is used to create a strikethrough effect on text. It is commonly used to indicate deleted or deprecated content.
- **`<code>`**: The `<code>` tag is used to represent a fragment of computer code or program. It is typically displayed in a monospace font to distinguish it from regular text.

These tags provide a way to visually differentiate and format text in HTML. By using these formatting tags appropriately, you can enhance the readability and presentation of your content. It's important to note that the actual visual styling of these tags may vary depending on the browser and CSS styles applied to your HTML document.

## Hyperlinks

Hyperlinks in HTML are created using the `<a>` tag (anchor tag). Hyperlinks allow you to link to other web pages or resources. Here's an explanation of the concepts mentioned:

→ `<a>` tag: The `<a>` tag is used to create hyperlinks in HTML. It stands for anchor and serves as the container for the link. The opening `<a>` tag starts the link, and the closing `</a>` tag marks the end of the link. For example:

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

→ `href` attribute: The `href` attribute is used within the `<a>` tag and specifies the URL or destination of the link. It indicates where the link should navigate to when clicked. The `href` attribute value can be an absolute URL (e.g., "<https://www.example.com>") or a relative URL (e.g., "about.html"). For example:

```
<a href="about.html">About</a>
```

## Relative and Absolute links

HTML supports both relative and absolute links.

- **Relative links:** A relative link specifies the path or location of the linked resource relative to the current page. It can be a file in the same directory, a file in a subdirectory, or a file in a parent directory. Relative links are useful when linking within the same website or referencing resources within the project structure.
- **Absolute links:** An absolute link specifies the complete URL or web address of the linked resource, including the protocol (e.g., "http://" or "https://"). Absolute links are used when linking to external websites or resources.

```
<a href="contact.html">Contact</a> <!-- Relative link -->  
<a href="https://www.example.com">Visit Example Website</a> <!-- Absolute link -->
```

## Images

Adding images in HTML using the `<img>` tag. By using the `<img>` tag and specifying the `src` attribute, you can insert images into your HTML documents, enhancing their visual content. The `alt` attribute provides important accessibility information, ensuring that users can understand the image's content even if it cannot be displayed. Here's an explanation of the concepts mentioned:

- `<img>` tag: The `<img>` tag is used to insert images into an HTML document. It is a self-closing tag, meaning it doesn't require a closing tag. The `<img>` tag is used to define the location and display settings of the image. For example:

```

```

- `src` attribute: The `src` attribute is used within the `<img>` tag and specifies the source URL or file path of the image. It indicates where the browser should retrieve the image from. The `src` attribute value can be a relative or absolute URL, or a file path on the local system. For example:

```
  
<!--In this example, the image is located in the "images" directory relative to the HTML file.-->
```

- `alt` attribute: The `alt` attribute provides alternative text for the image. It is displayed if the image cannot be loaded or when using assistive technologies such as screen readers. The `alt` text should describe the content or purpose of the image. For example:

- Displaying images: In addition to the `src` and `alt` attributes, you can also use other attributes to control the display of the image, such as width and height. For example:

```

```

## Forms

By using the `<form>` tag and its associated form elements, you can create interactive forms to collect user input. The `action` attribute specifies where the form data will be sent, and the `method` attribute determines how the data will be transmitted. Server-side scripts can then handle the form data for processing or storage.

- **`<form>` tag:** The `<form>` tag is used to create a form in HTML. It acts as a container for form elements that collect user input. The opening `<form>` tag starts the form, and the closing `</form>` tag marks the end of the form. For example:

```
<form action="submit.php" method="post">
  <!-- form elements go here -->
</form>
```

- **action attribute:** The `action` attribute within the `<form>` tag specifies the URL or file that will handle the form data when it is submitted. The `action` attribute value typically points to a server-side script that processes the form data.
- **method attribute:** The `method` attribute within the `<form>` tag specifies the HTTP method used to send the form data to the server. The two most common methods are `GET` and `POST`.

Form elements: Form elements are used to collect specific types of user input within a form. Some commonly used form elements include:

- **`<input>`:** The `<input>` tag is used to create various types of input fields, such as text fields, checkboxes, radio buttons, and more. The `type` attribute determines the specific type of input field. For example:

```
<input type="text" name="username" placeholder="Enter your username">
<input type="checkbox" name="agree" value="1">
<input type="radio" name="gender" value="male">
```

- **`<select>` and `<option>`:** The `<select>` tag is used to create a dropdown list, and the `<option>` tag is used to define the available options within the dropdown list. For example:

```
<select name="country">
  <option value="usa">United States</option>
  <option value="canada">Canada</option>
  <option value="uk">United Kingdom</option>
</select>
```

- **`<label>` element in HTML** is used to associate a text label with a form element, typically an input field. It provides a visual description or caption for the corresponding form element, enhancing the usability and accessibility of the form.

```
<label for="username">Username:</label>
<input type="text" id="username" name="username">
```

## Semantic HTML

Semantic HTML refers to the practice of using HTML markup in a way that conveys the meaning and structure of the content on a web page, rather than focusing solely on its visual appearance. By using semantic HTML, web

developers can provide more context and clarity to both human users and automated systems that interact with the web page.

There are several benefits to using semantic HTML. First, it improves search engine optimization (SEO) by enabling search engines to better understand the content and purpose of different elements on the page. This can lead to improved rankings in search results.

Second, semantic HTML enhances web accessibility. People with disabilities often rely on assistive technologies such as screen readers to navigate the web. Semantic HTML helps these technologies interpret and present the content in a meaningful way, making it easier for users with disabilities to access and understand the information on the page.

Lastly, using semantic HTML makes the code more readable and maintainable. By choosing appropriate semantic elements (e.g., `<header>`, `<nav>`, `<main>`, `<footer>`) to represent the different sections of a webpage, developers can quickly understand the structure and purpose of each element. This improves collaboration among developers and makes it easier to update or modify the code in the future.

## `<header>`

The `<header>` tag is indeed used to mark up the top section of a web page. It helps create a clear separation between the header and the main content of the page, making it easier for users to understand and navigate the website.

By placing common elements such as the main title or logo, navigation links, and other relevant information within the `<header>` element, you establish a consistent layout across multiple pages of your site. This ensures that visitors can easily identify and access these important elements regardless of which page they are on.

```
<!-- Non-semantic header -->
<div class="header">
  <h1>My Site</h1>
  <nav>
    <a href="#">Home</a>
    <a href="#">About</a>
    <a href="#">Contact</a>
  </nav>
</div>

<!-- Semantic header -->
<header>
  <h1>My Site</h1>
  <nav>
    <ul>
      <li><a href="#">Home</a></li>
      <li><a href="#">About</a></li>
      <li><a href="#">Contact</a></li>
    </ul>
  </nav>
</header>
```

## `<nav>`

The `<nav>` tag is used to mark up a section of a web page that contains navigation links. It helps indicate to both users and search engines that the enclosed content is intended for navigation purposes. By wrapping your navigation links with the `<nav>` tag, you provide a semantic structure to the navigation section of your webpage.

Moreover, using the `<nav>` tag also benefits users by providing a clear indication of where the navigation elements are located on the page. It helps them quickly identify and access the navigation menu, improving the overall user experience and making it easier for them to navigate through your website.

```
<!-- Non-semantic navigation -->
<div class="navigation">
  <a href="#">Home</a>
  <a href="#">About</a>
  <a href="#">Contact</a>
</div>

<!-- Semantic navigation -->
<nav>
  <ul>
    <li><a href="#">Home</a></li>
    <li><a href="#">About</a></li>
    <li><a href="#">Contact</a></li>
  </ul>
</nav>
```



```
</ul>
</nav>
```

## <main>

The <main> tag is used to mark up the main content area of a web page. It helps indicate to both users and search engines which section of your page contains the primary content.

Additionally, using the <main> tag benefits users by clearly identifying the primary content area of the page. It helps them focus on the essential information or the core message of the webpage. Screen readers and other assistive technologies also recognize the <main> tag, allowing users with disabilities to easily navigate to the main content of the page.

```
<!-- Non-semantic main content -->
<div class="main-content">
  <h2>About Us</h2>
  <p>Welcome to our website. We are a company that specializes in widgets.</p>
</div>

<!-- Semantic main content -->
<main>
  <h2>About Us</h2>
  <p>Welcome to our website. We are a company that specializes in widgets.</p>
</main>
```

## <article>

The <article> tag is used to represent a standalone piece of content within an HTML document. It is typically used for content that can be independently distributed or syndicated, such as blog posts, news articles, forum posts, or product reviews.

The <article> tag helps to encapsulate and define a self-contained and meaningful unit of content within a larger document. It provides a semantic structure that aids in understanding the purpose and significance of the enclosed content. This is beneficial for both users and search engines, as it allows them to easily identify and distinguish individual articles or pieces of content within a webpage.

```
<!-- Non-semantic article -->
<div class="article">
  <h2>How to Make a Widget</h2>
  <p>Widgets are great. Here's how to make one.</p>
</div>

<!-- Semantic article -->
<article>
  <h2>How to Make a Widget</h2>
  <p>Widgets are great. Here's how to make one.</p>
</article>
```

## <aside>

The <aside> tag in HTML is used to mark up content that is related to the main content of a web page but is not considered an integral part of it. This section often contains supplementary or supporting information that is tangentially related to the primary content.

The **<aside>** element is commonly used for things like:

1. Sidebar content: Additional information or links that provide context or related resources to the main content.
2. Advertisements: Ads that are related to the content but are not a part of the main article or webpage.
3. Related articles: Links to other articles or content on the same website that might be of interest to the reader.

```
<div>
  <article>
    <h2>Article Title</h2>
    <p>Article content goes here</p>
  </article>
  <aside>
    <h3>Related Articles</h3>
    <ul>
```

```

<li><a href="#">Article 1</a></li>
<li><a href="#">Article 2</a></li>
<li><a href="#">Article 3</a></li>
</ul>
</aside>
</div>

```

## <section>

The `<section>` tag in HTML is used to mark up sections of a web page that are thematically grouped together. It helps organize and structure the content of a webpage into logical and cohesive sections.

The `<section>` element is commonly used for:

1. Chapters or sections of an article: When a lengthy article or document is divided into distinct parts or chapters, each section can be marked up using the `<section>` tag.
2. Different parts of a product page: On a product page, different sections such as product description, features, specifications, reviews, and related products can be marked up as separate `<section>` elements.
3. Grouping related content: Any content that belongs together and can be identified as a distinct section can be enclosed within a `<section>` tag.

```

<section>
  <h2>Section Title</h2>
  <p>Section content goes here</p>
</section>

```

## <footer>

The `<footer>` tag in HTML is used to mark up the bottom section of a web page. It represents the footer area of the webpage, which typically contains information related to the page itself, such as copyright notices, contact details, legal information, navigation links, or links to social media profiles.

The `<footer>` element provides a semantic structure to the footer section of a webpage, allowing you to distinguish it from the main content. It helps in organizing and presenting important information related to the page or website.

```

<!-- Non-semantic footer -->
<div class="footer">
  <p>&copy; 2021 My Site</p>
</div>

<!-- Semantic footer -->
<footer>
  <p>&copy; 2021 My Site</p>
</footer>

```

## <details> and <Summary>

The `<details>` and `<summary>` tags in HTML are used together to create a collapsible section of content, allowing users to show or hide the details as needed.

The `<summary>` tag is used to mark up the title or heading of the collapsible section. It provides a brief description or label for the content that can be toggled.

The `<details>` tag is used to wrap the content that should be collapsible. It contains the actual details or additional information that can be expanded or collapsed.

```

<details>
  <summary>Click to expand</summary>
  <p>Content goes here</p>
</details>

```

## <figure> and <figcaption>

The `<figure>` and `<figcaption>` tags in HTML are used together to mark up a self-contained piece of content that is referenced from the main content of a web page.

The `<figure>` tag is used to mark up the self-contained content, which can include images, illustrations, diagrams, videos, code snippets, or any other media that stands alone and is referenced from the main content. It's worth noting that the `<figure>` and `<figcaption>` tags are not limited to images. They can be used for any type of self-contained content that requires a caption or description.

```
<figure>
  
  <figcaption>Image caption</figcaption>
</figure>
```

## `<mark>`

The `<mark>` tag in HTML is used to mark up text that has been highlighted or marked for some reason. It is commonly used to indicate search results, keyword matches, or text that has been highlighted by a user.

The `<mark>` tag is a semantic element that helps visually distinguish and draw attention to specific sections of text. By wrapping the highlighted text with the `<mark>` tag, you can apply styling or other visual cues to make it stand out.

```
<p>Here is some <mark>highlighted</mark> text.</p>
```

## `<time>`

The `<time>` tag in HTML is used to mark up a specific date or time on a web page. It provides semantic meaning to the content and helps search engines and other technologies interpret and understand the information.

The `<time>` tag is typically used to mark up dates, times, or durations, and it can also include additional attributes to provide further context, such as `datetime`, `pubdate`, or `timezone`.

```
<h1>Web Page Title</h1>

<p>Article published: <time datetime="2023-07-05T10:30:00Z">July 5, 2023</time></p>

<p>Event starts: <time datetime="2023-07-10T15:00:00+03:00" timezone="Europe/Helsinki">July 10, 2023 at 3:00 PM (Helsinki T
```

In this example, the `<time>` tag is used to mark up a published date and an event start time. The `datetime` attribute is used to specify the date and time in a machine-readable format (ISO 8601) for better understanding by search engines and other technologies. The `timezone` attribute is used to indicate the timezone of the event start time.

By using the `<time>` tag, you provide semantic meaning to the date and time information, allowing search engines to recognize and interpret it as temporal data. This can be helpful for search engines to display relevant search results, enable users to filter content based on specific dates or times, or provide accessibility benefits for users who rely on assistive technologies.

It's important to note that the visual presentation of the `<time>` tag is not defined by default. You can style it using CSS to match your desired design.

## `<progress>`

The `<progress>` tag in HTML is used to mark up a progress bar, indicating the completion status or progress of a task or process. It provides a visual representation of the progress, making it easier for users to understand the current state of an ongoing operation.

The `<progress>` tag requires two attributes: `value` and `max`. The `value` attribute specifies the current progress value, while the `max` attribute defines the maximum value or completion point of the task.

```
<h1>Web Page Title</h1>

<p>File Upload Progress:</p>
<progress value="50" max="100"></progress>
```

## `<div>` and `<span>`

The `<div>` and `<span>` tags in HTML are generic container elements that do not have any inherent semantic meaning. They are commonly used as layout elements or wrappers to group and style other elements or content on a web page.

Here's a brief description of each tag:

- `<div>`: The `<div>` tag is a block-level element that is used to create a division or section within an HTML document. It is a versatile container that allows you to group and structure content, apply styling, or target specific sections with CSS or JavaScript.
- `<span>`: The `<span>` tag is an inline element that is used to mark up small portions of text or inline elements within a larger block of content. It is typically used to apply styling, target specific sections with CSS or JavaScript, or provide hooks for manipulating the content programmatically.

While the `<div>` and `<span>` tags are not semantic in nature, they serve as essential tools for organizing and styling content on a web page. They provide flexibility and control over the layout and presentation of elements, allowing you to create custom designs and structure your content as needed.

However, it's important to prioritize the use of semantic HTML elements, such as `<header>`, `<nav>`, `<main>`, `<article>`, `<section>`, and others we discussed earlier, whenever they are applicable and convey the meaning of the content more accurately. Semantic HTML helps improve accessibility, search engine optimization, and code maintainability.

In summary, while the `<div>` and `<span>` tags are not semantic tags themselves, they play a crucial role in organizing and styling content within an HTML document. However, it's best to use semantic tags whenever possible to provide meaningful structure and enhance the accessibility and understandability of your web pages.

---

## Forms and Validations

Forms and validations are concepts commonly used in web development to ensure the integrity and accuracy of data entered by users into online forms.

### Forms:

In web development, a form is a user interface element that allows users to submit data to a website or application. Forms typically consist of input fields, checkboxes, radio buttons, dropdown menus, and buttons. They are used for various purposes such as user registration, data entry, feedback submission, and more. When users fill out a form and submit it, the entered data is sent to the server for processing.

### Validations:

Form validations are mechanisms employed to verify the correctness and completeness of user input before it is accepted and processed by the server. Validations help ensure that the data entered into the form meets specific criteria and is suitable for the intended purpose. They prevent users from submitting incorrect or incomplete data, improving the quality of the information and reducing potential errors in the system.

Common types of form validations include:

1. Required Fields: Ensuring that essential fields are filled out before submitting the form.
2. Data Types: Validating that the data entered matches the expected type, such as numbers, dates, email addresses, etc.
3. Length Constraints: Checking that the entered data meets specific length requirements, such as minimum and maximum character limits.
4. Format Validations: Verifying that the entered data follows a specific format, such as a valid phone number, ZIP code, or credit card number.
5. Comparison Validations: Comparing the values of different fields to ensure consistency, such as password confirmation.
6. Unique Entries: Ensuring that the entered data is unique, such as a unique username or email address.
7. Custom Validations: Implementing additional checks based on specific business rules or requirements.

Validations can be performed both on the client-side (using JavaScript) and on the server-side (using server-side programming languages like PHP, Python, or Ruby). It is recommended to perform validations on both sides to provide a better user experience and maintain data integrity.

---

## HTML best practices

Here are some HTML best practices to follow when developing web pages:

- **Use a Valid Doctype:** Begin your HTML document with a valid doctype declaration to ensure proper rendering and compatibility across different browsers.
- **Indentation and Formatting:** Use consistent indentation and formatting to enhance readability and maintainability of your code. Indent nested elements with spaces or tabs, and use line breaks to separate logical sections.
- **Use Semantic HTML:** Utilize semantic HTML elements to give meaning to the structure of your web page. This improves accessibility, search engine optimization (SEO), and makes your code more readable.
- **Provide Alternative Text for Images:** Always include descriptive alternative text (alt attribute) for images. This is important for accessibility and helps users with visual impairments understand the content of the image.
- **Use External CSS and JavaScript:** Place your CSS styles and JavaScript code in external files and link them to your HTML document. This promotes separation of concerns and allows for better code organization and maintenance.
- **Maintain Consistent Naming Conventions:** Use descriptive and meaningful names for your classes, IDs, and other attributes to improve code readability and maintainability.
- **Minimize Inline Styles:** Minimize the use of inline styles (style attributes) within your HTML. Instead, utilize external CSS files to keep styling separate from your markup.
- **Provide Clear and Concise Comments:** Use comments in your HTML code to explain complex sections, provide context, or document important information. However, avoid excessive comments that clutter the code unnecessarily.
- **Optimize Page Load Speed:** Optimize your HTML and associated resources (images, scripts, stylesheets) to improve page load speed. Minify your HTML code, compress images, and reduce the number of HTTP requests by combining and minifying CSS and JavaScript files.
- **Test Cross-Browser Compatibility:** Ensure your HTML code works correctly and renders consistently across different web browsers by testing your web page on multiple browsers and versions.
- **Accessibility Considerations:** Follow accessibility best practices by using appropriate HTML elements, providing alternative text for images, ensuring proper heading structure, and using ARIA attributes when necessary.
- **Regularly Validate Your HTML:** Use HTML validators (e.g., W3C Markup Validation Service) to validate your HTML code and fix any errors or warnings. This helps ensure your code adheres to HTML standards and reduces potential issues.

---

## Accessibility

Web accessibility is a fundamental aspect of web development that ensures people with disabilities can perceive, understand, navigate, and interact with websites and web applications effectively. It aims to remove barriers and provide equal access and opportunities for individuals with disabilities. Here are key points to understand about web accessibility:

1. **Inclusive Design:** Web accessibility promotes inclusive design practices, considering diverse user needs and abilities from the start of the development process. By designing with accessibility in mind, websites can be usable by a wide range of users, including those with visual, auditory, physical, cognitive, or neurological disabilities.
2. **Standards and Guidelines:** Accessibility standards and guidelines provide a framework for creating accessible web content. The Web Content Accessibility Guidelines (WCAG) developed by the World Wide Web Consortium (W3C) are widely accepted as the international standard for web accessibility. WCAG provides a set of principles, guidelines, and success criteria to make web content more accessible.
3. **Assistive Technologies:** Assistive technologies, such as screen readers, screen magnifiers, speech recognition software, and alternative input devices, help individuals with disabilities access and interact with web content. Ensuring compatibility with assistive technologies is essential for web accessibility.
4. **Key Accessibility Considerations:** To improve web accessibility, consider the following aspects:

- **Text alternatives:** Provide alternative text for images, captions for videos, and transcripts for audio content to ensure that users with visual or auditory impairments can understand the information.
  - **Keyboard accessibility:** Ensure that all functionality can be operated using a keyboard alone, without relying on mouse interactions. This helps individuals with motor disabilities or those who cannot use a mouse.
  - **Color contrast:** Use sufficient color contrast between text and background elements to make content readable for people with visual impairments or color vision deficiencies.
  - **Heading structure:** Use proper heading tags (h1, h2, h3, etc.) to create a logical and well-structured document outline. This assists users in navigating and understanding the content.
  - **Form accessibility:** Include descriptive labels for form fields, provide clear instructions, and ensure that form controls are operable and understandable for all users.
  - **Multimedia accessibility:** Include captions, transcripts, or audio descriptions for multimedia content to make it accessible to individuals with hearing or visual impairments.
  - **Responsive design:** Create websites that are responsive and adapt to different screen sizes and devices. This helps users with disabilities who may rely on specific devices or assistive technologies.
5. **Testing and Evaluation:** Conduct regular accessibility testing to identify and fix any accessibility issues. Manual testing, automated tools, and user testing with individuals with disabilities can help ensure that your website meets accessibility standards.

Web accessibility is not only a legal and ethical responsibility but also improves the user experience for all users. By making your website accessible, you can reach a broader audience, enhance usability, and demonstrate a commitment to inclusivity and equal access to information and services.

---

## Basics of SEO

SEO (Search Engine Optimization) is the practice of optimizing your website to improve its visibility and rankings in search engine results pages (SERPs). It involves various techniques and strategies aimed at making your website more attractive to search engines, thereby increasing its chances of appearing higher in relevant search queries. Here are some basic principles and components of SEO:

1. **Keyword Research:** Identify the relevant keywords and phrases that users are likely to search for when looking for content related to your website. Use keyword research tools to find popular and relevant keywords that align with your website's content and goals.
2. **On-Page Optimization:** Optimize various elements on your website to make it search engine-friendly. This includes optimizing page titles, meta descriptions, heading tags (H1, H2, etc.), URL structure, and keyword usage in the content. Ensure that your content is well-structured, informative, and relevant to the target keywords.
3. **Off-Page Optimization:** Develop a strong online presence and build high-quality backlinks from reputable websites. Backlinks are links from other websites that point to your site, indicating its credibility and authority. The more quality backlinks you have, the more search engines perceive your website as valuable and trustworthy.
4. **Technical SEO:** Ensure that your website is technically optimized for search engines. This includes improving website loading speed, optimizing mobile responsiveness, using descriptive URLs, implementing structured data markup, creating XML sitemaps, and managing robots.txt files.
5. **User Experience (UX):** Provide a positive user experience on your website, as search engines consider user satisfaction when ranking websites. Ensure your website is easy to navigate, has a clear site structure, is mobile-friendly, and provides valuable and relevant content to users.
6. **Content Optimization:** Create high-quality, unique, and engaging content that satisfies user intent. Use your target keywords naturally throughout the content, but avoid keyword stuffing. Provide informative and valuable content that answers users' questions and solves their problems.
7. **Local SEO:** If your business has a physical location, optimize your website for local search. Include your business name, address, and phone number (NAP) on your website and in online directories. Create a Google My Business profile to improve your visibility in local searches.
8. **Analytics and Monitoring:** Track and analyze your website's performance using tools like Google Analytics. Monitor key metrics such as organic traffic, rankings, bounce rate, and conversion rates. This data helps you understand how your SEO efforts are impacting your website's visibility and user engagement.

9. Regular Updates and Maintenance: SEO is an ongoing process. Stay updated with search engine algorithm changes and adapt your strategies accordingly. Regularly audit your website for any issues or opportunities for improvement.

Remember, SEO is a complex and ever-evolving field. It's essential to stay informed about the latest best practices and guidelines to optimize your website effectively and improve its visibility in search engine results.

---