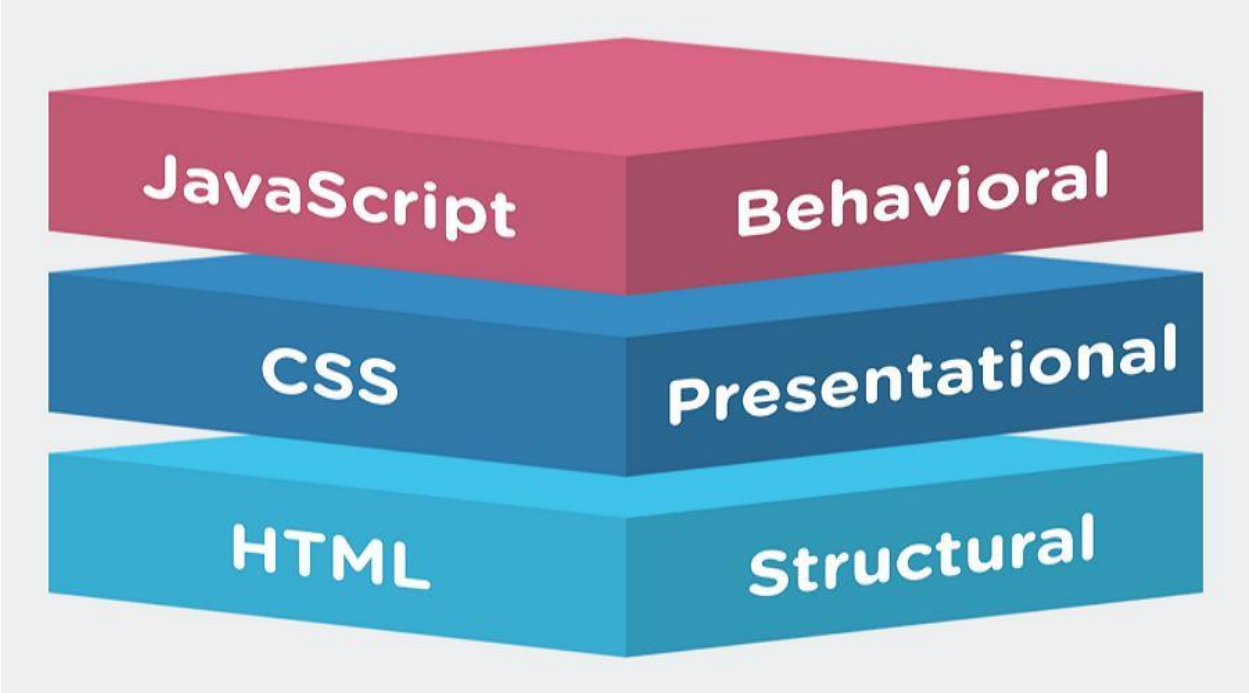
FRONT-END DEVELOPMENT



HTML5 Syntax: Style Guide and Coding Conventions:

CONTENTS

1. Introduction to HTML?
2. Basics of HTML Syntax
3. Uppercase or Lowercase?
4. Long Lines and Blank Spaces
5. Adding HTML Comments
6. HTML5 Semantic Tags:
    - 5.1 What Is Semantic Markup?
    - 5.2 The Origins of HTML5 Semantic Tags
    - 5.3 Semantic Markup for Document Structure
    - 5.4 Other Commonly Used Semantic Tags
    - 5.5 HTML5 Semantic Tags: Useful Tips
7. HTML Syntax: Useful Tips
8. The Ultimate HTML Cheat Sheet: A List of HTML Tags [A-Z]
9. HTML Attributes

Introducing HTML Basics

HTML is a markup language that forms a base for any website on the Internet. HTML5 is the latest version of HTML, finalized in 2014.

What is HTML: a Simple Explanation

To create polished and well-functioning websites, you will need **CSS** and **JavaScript**. However, to make them work as intended, you need to first understand what is HTML. Just like a house has a foundation, a **web page** has its base in HTML.

The answer to what does HTML stand for is simple: **HyperText Markup Language**. It is a system that allows you to define the structure of the content in your web page by using **elements** wrapped in universally supported **tags**.

History of HTML: Different Versions

HTML 1 was created by a CERN scientist Tim Berners-Lee. His initial goal was an Internet-based hypertext system that allows sharing and using documents in different computers. Introduced in 1991, HTML 1 only had **18 tags**, most of them based on the Standard Generalized Markup Language (SGML). **HTML 2** was presented in **1995** and had a few more features.

The draft of **HTML 3** was abandoned due to slow implementation of the newly created tags. Therefore, the World Wide Web Consortium(**W3C**) set out to standardize HTML. In **1997**, HTML 3.2 was released which became a standard at the time.

HTML 4 was a large step, as it separated styling from coding, leaving the former to CSS. A revised version called HTML 4.01 came out in 1999, correcting minor mistakes found in HTML 4.0 and introducing a few handy features.

HTML 5 is the HTML as we know it today.

> **Note:** you might come upon phrases like 'The difference between HTML and HTML5' or 'HTML vs HTML5'. Don't get confused: developers nowadays use the generic name HTML to refer to all versions prior to HTML5.

HTML5 was first released in 2008, though the specification process wasn't complete until 2014. As of now, this markup language is the base of every new website.

When comparing HTML vs HTML5, the first major difference is the level of support. Websites created with HTML5 can be used across all major browsers and different platforms as well. This is vital in the XXI century when the majority of users access the Web via mobile devices.

The **semantic tags** presented in **HTML5** help a great deal with search engine optimization (SEO). Their purpose is to allow the browser to understand the structure of the page. It also improves accessibility in mobile devices and simplifies the job for **assistive technologies** (e.g., screen readers).

HTML5 is also handy when you need to create dynamic and engaging content, as it allows you to include **video** and **audio** without relying on **Adobe Flash**.

A Basic HTML Document

Beginners might think: is HTML a programming language? It is actually not – remember, HTML stands for hypertext markup language. Therefore, working with HTML is coding, not programming.
The simplest HTML document consists of a few code lines:

```
<!DOCTYPE html>
<html>
<head>
 <title>Name of the website</title>
</head>
<body>
<div style="background-color: #333; color: white; padding: 10px;">
  <h3>This header has same color as a div</h3>
  <p>This paragraph should have same color as div</p>
</div>
</body>
</html>
```

Let's review what it contains in the following sections.

The <!DOCTYPE> declaration informs the browser about the document type in order to load a page successfully.

The <html> element is used to define an HTML document so web browsers could recognize and display it.

The <head> element contains all information about what is the HTML document. You can add a title for the website, some required meta information, styles, and more.

The <title> element describes a document's title which informs readers about the content of a website.

The <body> tag makes content visible on websites: without it, your website won't be functional. Every element that should be shown in a website belongs in this section.

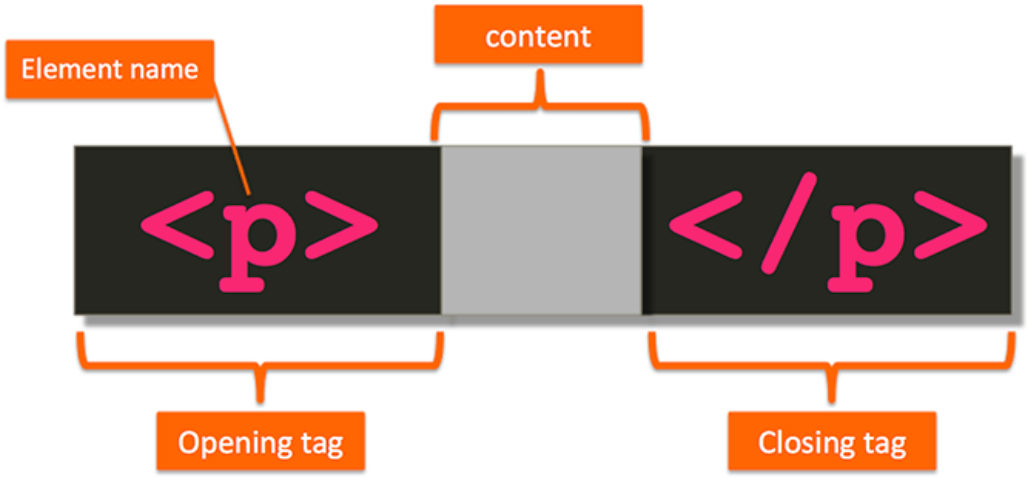
 is known as the division. It's a block level element, applied to divide a page into more user-friendly sections.

The <h1>–<h6> elements specify headings. A heading is similar to a title, but it has lower importance as it names parts of the document, not the whole website. Use them to point out content titles and names.

The <p> element defines a paragraph of text. Do not use it to create titles, headings or other important text parts.

Basics of HTML Syntax

The first and most important thing to understand in HTML syntax is the usage of **elements** and **tags**. Every HTML element is defined by an **opening tag** and a **closing tag** wrapped around their **content**:



A pair of **angle brackets** (< and >) surround each tag. The difference between the opening tag and a closing tag is a **single forward slash** (/). See a simple example of using the HTML <***p***> tags.

These days, every developer is encouraged to use **semantic tags**. They help HTML5 structure a page better, and that in turn helps you create a nice HTML5 layout for your webpage.

Closing Elements

It is not necessary to close all elements in HTML5. However, if you're writing in XHTML or simply want a more readable and syntactically correct code, you should practice closing all the tags:Example

<***section***>
  <***p***>This element is not closed which can cause problems in the future.
 <***p***>This is a closed element. The range where it starts and where it ends is clearly defined</***p***>
</***section***>

An **empty element** only has an opening tag and doesn't require closing. However, XHTML requires all elements to be closed. If there is a chance that XML software will be accessing your page in the future, you should close your empty elements, for it may cause inconveniences.
You can make an empty element close itself (self-closing tags) by adding a forward slash (/) before the **second angle bracket** (>):
Example

HTML5: **<hr>** not closed

XHTML: **<hr/>** closed

Uppercase or Lowercase?

In the code snippet below, you see the **document type declaration**. It should always be the first line of the document. Although all the examples you see work, HTML syntax rules recommend to stick to either **uppercase** or **lowercase**:
Example:

```
<!DOCTYPE html>

<!DOCTYPE HTML>

<!doctype html>

<!Doctype Html>
```

Technically, you could mix uppercase and lowercase letters in HTML5. However, it's generally deemed good practice to use lowercase letters for element and attribute names. That improves both the look and the readability of your code:

```
<section>
  <p> I'm a paragraph that is easy to understand.</p>
</section>
```

Note: to prevent issues, it is recommended to always use **lowercase** when naming files too, as **web servers** are often **case-sensitive**.

Long Lines and Blank Spaces

All your HTML5 code should be viewable by scrolling **from top to bottom**. It is extremely inconvenient for a user to have to scroll sideways. It is also unprofessional regarding HTML syntax rules. Try not to make your code lines longer than **70 characters**.

You shouldn't add **blank lines** without a good reason. You can add a blank line to separate large code blocks, but don't overuse them.

White space is tricky too. You can **add two spaces** for indentation:

Example

```
<!DOCTYPE html>
<html>
<body>

<section>

<h1>Without indentation it is hard to read</h1>

<p>This example has unnecessary blank lines and no indentation.</p>

</section>

  <section>
    <h1>Indentation is cool</h1>
    <p>This example has indentation instead of blank lines. It is easy to read and saves space.</p>
  </section>
</body>
</html>
```

However, avoid unnecessary spaces between **symbols.** E.g., while you technically can put spaces around = signs, HTML syntax recommends to avoid them for better readability:

Example

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

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

Adding HTML Comments

Code comments are written between ***<!-- and --> tags.*** They do not affect the web page on the user side, as their purpose is to only make notes for the developer:

Example

```
<!--
This is commented out text.
This is commented out text.
-->
<body>
  <p>This is visible <!-- this is an invisible comment --> text.</p>
</body>
```

HTML5 Semantic Tags: What Is Semantic Markup?

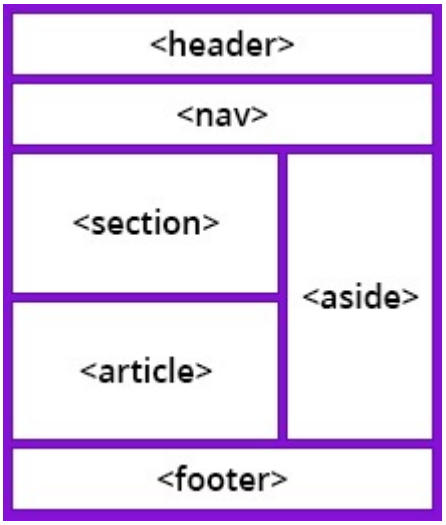
HTML5 semantic tags define the function and the category of your text, simplifying the work for browsers and search engines, as well as developers.

The origins of HTML5 Semantic Tags

In earlier versions of HTML, there were **no globally accepted names** for structural elements, and each developer used their own. That made it very hard for **search engines** to index web page content correctly.

When a browser communicates with the code, it looks for some specific information to help with the display. Hence, HTML5 introduced a **consistent** list of **semantic elements** to help search engines and developers.

HTML5 semantic tags define the **purpose of the element**. By using semantic markup, you help the browser understand the **meaning** of the content instead of just displaying it. By providing this extra level of clarity, HTML5 semantic elements also help **search engines** to read the page and find the **required information faster**.



Semantic Markup for Document Structure

First, we will discuss the HTML5 elements you can see in the illustration above. They are used to convey the structure of the document in a clear manner.

The **<header>** element defines a header of your document. It is always visible for the users at the top of the page:

Example

```
<header>
 <h1>JavaScript</h1>
 <h3>What is JavaScript?</h3>
 <p>Today we are going to talk about JavaScript</p>
</header>
```

The <nav> depicts the space for navigation links on your website:

Example

```
<nav>
  <ul>
    <li><a href="#"> Services </a></li>
    <li><a href="#"> Products </a></li>
    <li><a href="#"> Contacts </a></li>
  </ul>
</nav>
```

The **<*section*>** tags are used to define a separate section within a webpage, with its own content:

Example

```
<section>
  <h1>Section Heading</h1>
  <p>The section tag can contain any elements.</p>
  <img src="image.png" alt="section example">
</section>
```

The **<article>** element can be used to define the article content on your website:

Example

```
<article>
  <h1>Fun Fact</h1>
  <p>Fun fact: most of the fun facts on the Internet are not actually fun.</p>
</article>
```

The **<*aside*>** semantic element defines the content which will be set to the side. It is occasionally used for creating sidebars, but can also be used for displaying less important content:

Example

```
<aside>
  <h4>Lake</h4>
  <p>Oxford lake is a lake in the state.</p>
</aside>
```

The **HTML5** ***<footer>*** element describes the footnote for your website or part of the content:

Example

```
<footer>
  <address>
    Postal Address: Door No.00, Street, City, State, Country.
  </address>
  <p>Copyright © 2018 All rights reserved.</p>
</footer>
```

Other Commonly Used Semantic Tags

- The ***<figure>*** element depicts space for separated content, such as photos, diagrams, etc. To provide a caption for this element, use the ***<figcaption>*** tags:

Example

```
<figure>
  <figcaption> Dog </figcaption>
  <img src="image.png" alt="The Bread Dog" width="300" height="300">
</figure>
```

- The ***<details>*** element defines the details on your website that can either be visible or hidden. To add a summary for this element, use the ***<summary>*** element:

Example

```
<details>
   <summary>Some details</summary>
   <p>Provide more info about the details here.</p>
</details>
```

- The content of the ***<main>*** tags is the main content of a page. It can be an article, regular paragraph or anything else:

Example

```
<main id="content" class="group" role="main">
```

- The ***<mark>*** element highlights the text to emphasize its meaning:

Example

```
<p>The mark tag is <mark>useful</mark> when you need to highlight important information</p>
```

- The ***<time>*** element is used to define and display time and date on your web page:

Example

```
<h2>The premiere show starts at <time>21:00 </time> today.</h2>
```

- The **<*img*>** tags are used to add an image on your website:

Example

```
<img src="image.png" alt="Space Doggo" width="50" height="50">
```

- The **<*form*>** element allows you to include user input:

Example

```
<form action="search" method="GET">
  Search Term: <input type="text" name="search_query">
  <input type="submit" value="Search">
</form>
```

- The **<*table*>** tags are used to add a table with rows and columns on a web page:

Example

```
<table>
  <tr>
    <th>Place</th>
    <th>Animal</th>
  </tr>
  <tr>
    <td>Space</td>
    <td>Dog</td>
  </tr>
</table>
```

HTML Syntax: Useful Tips

- If you worry about forgetting to close your tags, start by typing both of them and then fill in the content.
- You can also try syntax highlighting (e.g., from highlight.js (***ctrl* + *click***) if you think it will help you understand the HTML5 document structure easier.
- If you are still unsure about your **HTML syntax**, try a simple syntax validator such as Freeformatter ((***ctrl* + *click***) which will display all **syntax errors**.

The Ultimate HTML Cheat Sheet: A List of HTML Tags [A-Z]

A

Tag	Description
<a>	Sets a hyperlink to another page
<abbr>	Represents an abbreviation or acronym and provides a full-text explanation to it (optional)
<acronym>	DEPRECATED. Defined an acronym and associated an explanation to it (optional)
<address>	Defines the contact information of the document's author.
<applet>	DEPRECATED. Embedded a Java applet into an HTML document
<area>	Defines a clickable area in an image map
<article>	Defines independent content
<aside>	Describes content to be displayed aside other content
<audio>	Includes sound in the web page (music, streams, etc.)

B

Tag	Description
<b>	Displays enclosed text in bold typeface without conveying any added importance
<base>	Defines the base URI or URL for all the relative links in the HTML document
<basefont>	DEPRECATED. Defined a default font-family, font-size, text-color for all the text in a document
<bdi>	Defines bidirectional text isolation
<bdo>	Define bidirectional text override
<big>	DEPRECATED. Defined a bigger font size in HTML
<blink>	DEPRECATED. Defined text to blink
<blockquote>	Defines a quoted section
<body>	Defines the HTML container that contains visible contents of an HTML document
** **	Defines a line break
<button>	Creates a button

C

Tag	Description
<canvas>	Defines an area of the webpage that becomes a space for rendering graphics
<caption>	Defines an HTML table caption
** **	DEPRECATED. Set the alignment of content to the center
<cite>	Defines a title for a work
<code>	Defines text to be displayed in the computer output style
<col>	Specifies grouped attributes for columns
<colgroup>	Specifies a common formatting style for a group of columns
<comment>	Specifies a comment in the document's source code

D

Tag	Description
<datalist>	Specifies an autocomplete feature to be used with a form element
<dd>	Describes a term in a description list
<del>	Defines deleted text in a document and crosses it out
<details>	Specifies additional details that can be shown or hidden manually
<dfn>	Represents a term that needs an explanation
<dialog>	Creates pop-up dialog models
<dir>	DEPRECATED. Defined a directory list
** **	Defines a section or a division in HTML document by grouping elements
<dl>	Defines a description list
<dt>	Defines a description term

E

Tag	Description
****	Defines emphasized text
<embed>	Defines an embedded resource

F

Tag	Description
<fieldset>	Defines a group of <form> elements
<figcaption>	Sets a caption for a <figure> element
<figure>	Defines a self-contained piece of content
<font>	DEPRECATED. Defined text style
<footer>	Defines footer context
<form>	Defines a group of submittable inputs
<frame>	DEPRECATED. Defined one single frame in a web page
<frameset>	DEPRECATED. Defined a container for holding multiple frames

H

Tag	Description
<h1> - <h6	Defines headings in a document
<head>	Defines a space for declaring title, stylesheets, scripts and other document related information
<header>	Specifies a container for one or more headings
<hr>	Defines a thematic division between content
<html>	Defines a document as an HTML document

I

Tag	Description
<i>	Defines an italic styled text
<iframe>	Defines a frame of inline nature
<img>	Sets an area for an image
<input>	Specifies an element used to take values from the user
<ins>	Defines inserted text in the document

K

Tag	Description
<kbd>	Identifies a user keyboard entry
<keygen>	DEPRECATED. Defined a key-pair used while submitting an HTML form

L

Tag	Description
<label>	Specifies a label for a given input element
<legend>	Sets a caption for the fieldset
** **	Defines a list item within a list
<link>	Sets a link between a document and other external resources

M

Tag	Description
<main>	Defines the primary content of the document
<map>	Defines a client-side image-map (an image with clickable places)
<mark>	Defines marked text
<menu>	Defines a list/menu of commands
<menuitem>	DEPRECATED. Defined a command, which can be invoked from the popup menu by the user
<meta>	Defines supplementary information about a website
<meter>	Displays a scalar measurement within a known range

N

Tag	Description
<nav>	Defines a block of navigational links to the main sections of a website
<noframes>	DEPRECATED. Defined fallback content for the browsers which did not support the <frame> tag
<noscript>	Defines HTML to be inserted as replacement if the browser does not support scripting

O

Tag	Description
<object>	Defines an external source that is embedded into the HTML document
** **	Defines an ordered list of items in the HTML document
<optgroup>	Groups the related options within a select element
<option>	Sets an HTML option in select element list
<output>	Displays the result of a calculation performed

P

Tag	Description
<p>	Defines an HTML paragraph of text
<param>	Defines parameters for an object element
<pre>	Defines preformatted text. Preserves spaces and line breaks
<progress>	Displays an HTML progress bar representing the progress of a task

Q

Tag	Description
<q>	Indicates that the enclosed text is an inline quotation

R

Tag	Description
<rp>	Defines a fallback parenthesis for a ruby annotation
<rt>	Defines ruby text for Asian characters
<ruby>	Represents ruby annotation for Asian characters

S

Tag	Description
<s>	Draws a line across the text making a strikethrough
<samp>	Defines sample output from a computer program
<script>	Defines a client-side script (JavaScript)
<section>	Defines the primary content of the document
<select>	Creates an HTML dropdown list with one or more options for a web form
<small>	Reduces the HTML text size down by one size
<source>	Defines sources for media elements like audio, video and picture
****	Sets an inline container for formatting an HTML document
<strike>	DEPRECATED. Defined a portion of a text to be struck out
****	Makes an important part of text appear bold
<style>	Defines styling properties for different HTML elements
<sub>	Defines subscript text
<summary>	Sets the summary content for the <details> element
<sup>	Defines superscript text

T

Tag	Description
<table>	Includes a table with rows and columns in a web page
<tbody>	Associates the body content in a table
<td>	Defines table data
<tfoot>	Defines table footer content
<th>	Defines an HTML header table cell
<thead>	Defines the title of the HTML document
<time>	Defines human and machine readable date and time
<title>	Defines the title of the HTML document
<tr>	Defines a row in an HTML table
<track>	Defines text tracks for all the media elements
<tt>	DEPRECATED. Defined text to be displayed in teletype font

U

Tag	Description
<u>	Underlines the text
** **	Defines an unordered list

V

Tag	Description
<var>	Defines a variable
<video>	Includes a video in the web page (a video clip, streams, etc.)

W

Tag	Description
<wbr>	Defines a possible line-break in a text

HTML Attributes

HTML attributes are additional values used to customize HTML elements.

In HTML, attributes can be applied to basically **any element**. They modify the **default behavior** of the element or specify certain characteristics (e.g., dimensions).

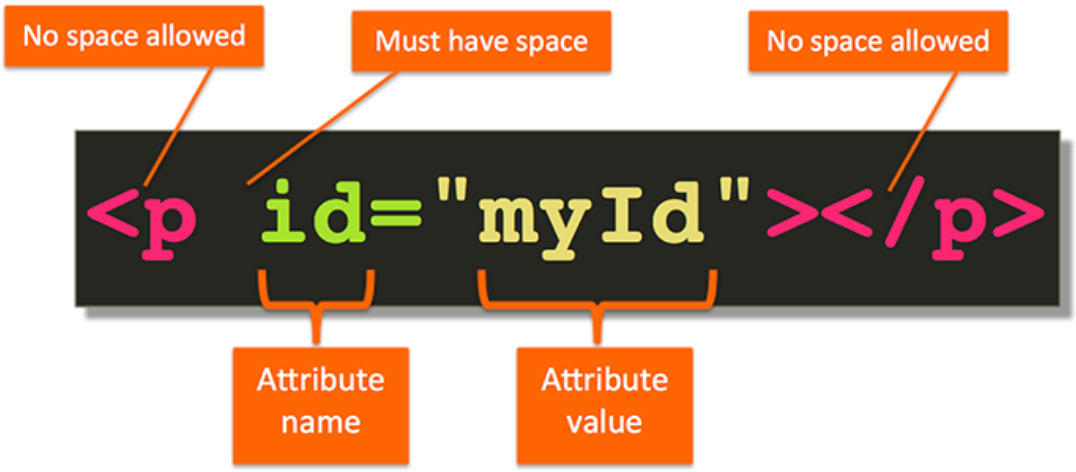
HTML attributes come in four basic types:

Type	Function	Supported by
Required	Required for an element to work as intended	One or multiple specific elements
Optional	Changes the element's default functionality	One or multiple specific elements
Global	Changes the element's default functionality	All elements
Event	Specifies conditions for a script to run	All elements

How to Write HTML Attributes?

HTML attributes are always included in the opening tag. You may add various attributes in the opening tag. They are written in name-value pairs with the value wrapped in double quotes:
The syntax is rather simple:
<tag name="value">
As you can see, attributes are specified in name-value pairs:

- **name:** defines the name for the HTML attribute
- **value**: specifies the value you wish to set for it



HTML **attributes** are **case-insensitive**. However, World Wide Web Consortium **recommends** to write them in **lowercase for a neater look**.

Note: *don't forget to enclose the value in quotes.*

Most Common HTML Attributes:

- id
- class
- style
- title
- alt
- href
- width
- height
- src
- type

The HTML **id** attribute helps you to add a unique ID for an element to be used for identification:

Example

```
<nav id="bahir-dar”>
  <a href="index.html">Your Home Page</a>
</nav>
```

The HTML **class** attribute creates a relation between the element and a stylesheet:

Example

```
<p class="paragraph1">This is paragraph text.</p>
```

The HTML **style** attribute allows you to provide inline styling for an element:

Example

```
<div style="background-color: black; color: white;">
  This text was styled using the style attribute
</div>
```

The HTML **title** attribute adds information related to the element. Hovering on it will cause a tooltip with the title you created to appear:

Example

```
<h1 title="This will show up after hovering the mouse over this element">Some random text</h1>
```

Note: *id, class,* ***style*** *and* ***title*** *attributes are supported* ***globally****.*

The **alt** attribute sets up some alternative text to be displayed in case an element (e.g., an image) cannot be loaded:

Example

```
<img src="image.png" alt="Ethiopia">
```

Note: you can use alt with <applet>, <area>, <img> and <input> elements.

The HTML **href** attribute adds an URL destination for a link, creating a hyperlink which can take you to any other webpage:

Example

```
<a href="#">Click this link to start learning. </a>
```

Note: *you can use HTML href with <a>, <area>, <base> and <link> elements.*

Using HTML **width** and **height** attributes, you can change the element's dimensions:

Example

```
<img src="image.png" width="100" height="100">
```

Note: *size is usually defined in* ***pixels (px).***

src is an abbreviation for source. By using this HTML attribute, you can define an external source for an element:

Example

```
<img src="image.png" alt="My Ethiopia" width="50" height="50">
```

Note: *you can use* **src** *with <frame>, <iframe>, <img>, <input> and <script> elements.*

HTML Attributes: Useful Tips

- When using multiple HTML attributes, you can list them **in any order** - just make sure to **separate them with spaces**.
- **alt** helps to make your website accessible for disabled users who use screen readers. When assistive technologies have an alternative text to read, people with disabilities can understand the content better.

- The HTML **title** attribute can help to optimize your website for search engines, as this supplementary information can also contain keywords.

HTML Inline and Block Elements

Based on how they are **displayed** by the **browser by default**, HTML elements are divided into **two groups: inline** and **block-level** elements.

While **block elements** move to a **new line** and take its whole width, inline elements stay in the line they were put in and don't take any more space than is needed for their content:

A good example of **inline elements** are **text formatting elements**, such as or . They only need to affect the look of text. Meanwhile, HTML **headings** and **paragraphs** are block elements, as they help to create the structure of the page.

Block Elements: Examples

HTML Block elements are the elements that **always** start with a **new line**, take up all available **width** of the website and are displayed in a column. It is the default value for a bunch of elements which we will review in this section.
HTML block:

- headings
- paragraphs
- form

 is used to specify an area for other elements on a website. It does not require any attributes, but it often includes id, class or style:

Example

```
<div style="background-color: lightpink;">
  This div is on the top.
</div>
<div style="background-color: cornsilk;">
  This div is on the bottom.
</div>
```

Inline Elements: Examples

HTML inline elements don't start with a **new line**. They also only take up as much space as their **content**. A few of them would be displayed side by side in the page.
HTML inline:

- Links
- Images
- Span

HTML Inline and Block Elements: Useful Tips

- The chances to **style** HTML inline elements with CSS are rather limited. However, you can set height, width, margin, padding and more for block elements.
- HTML elements only come as block or inline elements by default. However, in CSS, the **display** property also accepts an **inline-block** value. Such element stays in the line it was put in, but can be **styled** like a block element.

Brief Introduction to CSS3

The Cascading Style Sheets (**CSS**) is a language for styling documents of markup languages such as HTML.

List of CSS3 major modules explaining in the near future:

- ≈ CSS Introduction
- ≈ CSS Typography
- ≈ CSS Backgrounds & Colors
- ≈ CSS Effects
- ≈ CSS Layout
- ≈ CSS Responsive
- ≈ CSS Element Styling
- ≈ CSS References
- ≈ CSS Properties