Getting Started with HTML

Here we cover the basics of HTML. We will define elements, attributes, and all the other important terms. It also explains where these fit into HTML, you'll learn how HTML elements are structured, how a typical HTML page is structured, and other important basic language features.

The objective here is to gain basic familiarity with HTML, and practice writing a few HTML elements.

What is HTML?

The HyperText Markup Language (HTML) is a markup language that tells web browsers how to structure the web pages. It can be as complicated or as simple as the web developer wants it to be. HTML consists of a series of elements, which you use to enclose, wrap, or markup different parts of content to make it appear or act in a certain way. The enclosing tags can make content into a hyperlink to connect to another page, italicize words, and so on.

For example: The following text *"I love my cat"* can be displayed if we specify a paragraph by enclosing it in a paragraph element, like that:

```html
<p> I love my cat </p>
```

Anatomy of an HTML element

Let's further explore our paragraph element from the previous section

- **<p>:** This is the opening tag, consisting of the name of the element, wrapped in opening and closing angle brackets. This opening tag marks where the element begins or starts to take effect;
- **</p>:** This is the same as the opening tag, except that it includes a forward slash before the element name. This marks where the element ends. Failing to include a closing tag is common beginner error that can produce peculiar results;
- **I love my cat:** This is the content of the element;

All this combined is the **element,** consisting in an opening tag, followed by content, followed by the closing tag.

Nesting Elements

Elements can be placed within other elements. This is called *nesting*. If we wanted to state that i **love** my cat, we could wrap the word *very* in a <strong> element, which means that the word is to have strong text formatting

```
<p> I <strong> love </strong> my cat </p>
```

There is a right and wrong way to do nesting. In the example above, we opened the *p* element first, then opened the *strong* element. For proper nesting, we should close the *strong* element first, before closing the *p.*

The tags have to open and close in a way that they are inside or outside one another. With the kind of overlap in the example bellow, the browsers have to guess at your intent. This kind of guessing can result in unexpected results

```
<p> I <strong> love my cat </p> </strong>
```

Void elements

Not all elements follow the pattern of an opening tag, content, and a closing tag. Some elements consist of a single tag, which is typically used to insert/embed something in the document. Such elements are called *void elements*. For example, the <*img*> element embeds an image file into a page.

```
<img
src="https://raw.githubusercontent.com/mdn/beginner-html-site/gh-p
ages/images/firefox-icon.png" alt="Firefox icon" />
```

This would output the following:

Attributes

Elements can also have attributes. Attributes look like this:

```
<p class="some-class"> I love my cat </p>
```

Attributes contain extra information about the element that won't appear in the content. In this example, the *class* attribute is an identifying name used to target the element with style information.

An attribute should have:

- A space between it and the element name (For an element with more than one attribute, the attributes should be separated by spaces too);
- The attribute name, followed by an equal sign;
- An attribute value, wrapped with opening and closing quote marks;

Boolean attributes

Sometimes you will see attributes written without values. This is entirely acceptable. These are called Boolean attributes. Boolean attributes can only have one value, which is generally the same as the attribute name. For example, consider the *disabled* attribute, which you can assign to form input elements. (You use this to disable the form input elements so the user can't make entries. The disabled elements typically have a grayed-out appearance)

```
<input type="text" disabled="disabled" />
```

As shorthand, it is acceptable to write this as follows:

```
<!-- using the disabled attribute prevents the end user from entering
text into the input box -->
<input type="text" disabled />

<!-- text input is allowed, as it doesn't contain the disabled attribute
-->
<input type="text" />
```

Omitting quotes around attribute values

If you look at code for a lot of other sites, you might come across a number of strange markup styles, including attribute values without quotes. This is permitted in certain circumstances, but it can also break your markup in other circumstances. The element in the code snippet below, <*a*>, is called an anchor. Anchors enclose text and turn them into links. The *href* attribute specifies the web address the link points to. You can write this basic version below with only the *href* attribute , like this:

```
<a href=https://www.mozilla.org/> Favorite website </a>
```

Anchors can also have a *title* attribute, a description of the linked page. However, as soon as we add the *title* in the same fashion as the *href* attribute there are problems:

```
<a href=https://www.mozilla.org/ title=The Mozilla homepage>
Favorite website </a>
```

As written above, the browser misinterprets the markup, mistaking the *title* attribute for three attributes: a title attribute with the value *The*, and two Boolean attributes, *Mozzila* and *homepage*. Obviously, this is not intended! It will cause errors on unexpected behavior.

Always include the attribute quotes. It avoids such problems, and results in more readable code.

Single or double quotes?

Attributes are wrapped in double quotes. However, you might see single quotes in some HTML code. This is a matter of style. You can feel free to choose which one you prefer. Both of these lines are equivalent:

```
<a href='https://www.example.com'>A link to my example.</a>
<a href="https://www.example.com">A link to my example.</a>
```

Make sure you don't mix single quotes and double quotes. This example below shows a kind of mixing of quotes that will go wrong

```
<a href="https://www.example.com'>A link to my example.</a>
```

However, if you use one type of quote, you can include the other type of quote **inside** your attribute values:

```
<a href="https://www.example.com" title="Isn't this fun?">
  A link to my example.
</a>
```

To use quote marks inside other quote marks of the same type, use HTML entities. For example, this will break:

```
<a href="https://www.example.com" title="An "interesting"
reference">A link to my example.</a>
```

Instead, you need to do this:

```
<a href="https://www.example.com" title="An
&quot;interesting&quot; reference">A link to my example.</a>
```

Anatomy of an HTML document

Individual HTML elements aren't very useful on their own. Next, let's examine how individual elements combine to form an entire HTML page:

```
<!doctype html>
<html lang="en-US">
  <head>
    <meta charset="utf-8" />
    <title>My test page</title>
  </head>
  <body>
    <p>This is my page</p>
  </body>
</html>
```

- **<!doctype html>:** The doctype. When HTML was young, doctypes were meant to act as links to a set of rules that the HTML page had to follow to be considered good HTML. Doctypes used to look something like this:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

  More recently, the doctype is a historical artifact that needs to be included for everything else to work right. <!DOCTYPE html> is the shortest string of characters that counts as a valid doctype.

- **<html> </html>:** This element wraps all the content on the page. It is sometimes known as the root element.
- **<head> </head>:** This element acts as a container for everything you want to include on the HTML page, **that isn't the content** the page will show to viewers. This

includes keywords and a page description that would appear in search results, CSS to style content, character set declarations, and more.

- **<meta charset="utf-8">:** This element represents metadata that cannot be represented by other HTML meta-related elements, like <base>, <link>, <script>, <style> or <title>. The charset attribute specifies the character encoding for your document as UTF-8, which includes most characters from the vast majority of human written languages. With this setting, the page can now handle any textual content it might contain. There is no reason not to set this, and it can help avoid some problems later.
- **<title> </title>:** This sets the title of the page, which is the title that appears in the browser tab the page is loaded in. The page title is also used to describe the page when it is bookmarked.
- **<body> </body>:** This contains all the content that displays on the page, including text, images, videos, games, playable audio tracks or whatever else.

Whitespace in HTML

This two code snippets are equivalent:

```
<p id="noWhitespace">Dogs are silly.</p>

<p id="whitespace">Dogs
    are
        silly.</p>
```

No matter how much whitespace you use inside HTML element content, the HTML parser reduces each sequence of whitespace to a single space when rendering the code. So why use so much whitespace? The answer is readability.

It can be easier to understand what is going on in your code if you have it nicely formatted. In our HTML we've got each nested element indented by two spaces more than the one it is sitting inside. It is up to you to choose the style of formatting, but you should consider formatting it.

Entity references: including special characters in HTML

The characters: <, >, “, ‘, and & are special characters. They are parts of the HTML syntax itself. So how do you include one of these special characters in your text?

You do this with character references. These are special codes that represent characters, to be used in these exact circumstances. Each character reference starts with an ampersand (&), and ends with a semicolon (;)

Character	Character Reference
<	&lt;
>	&gt;
“	&quot;
‘	&apos;
&	&amp;

The character reference equivalent could be easily remembered because the text it uses can be seen as less than for ‘&lt.’, quotation for ‘&quot; and similarly for others.

HTML comments

HTML has a mechanism to write comments in the code. Browsers ignore comments, effectively making comments invisible to the user. The purpose of comments is to allow you to include notes in the code to explain your logic or coding. This is very useful if you return to a code base after being away for long enough that you don’t completely remember it. Likewise, comments are invaluable as different people are making changes and updates.

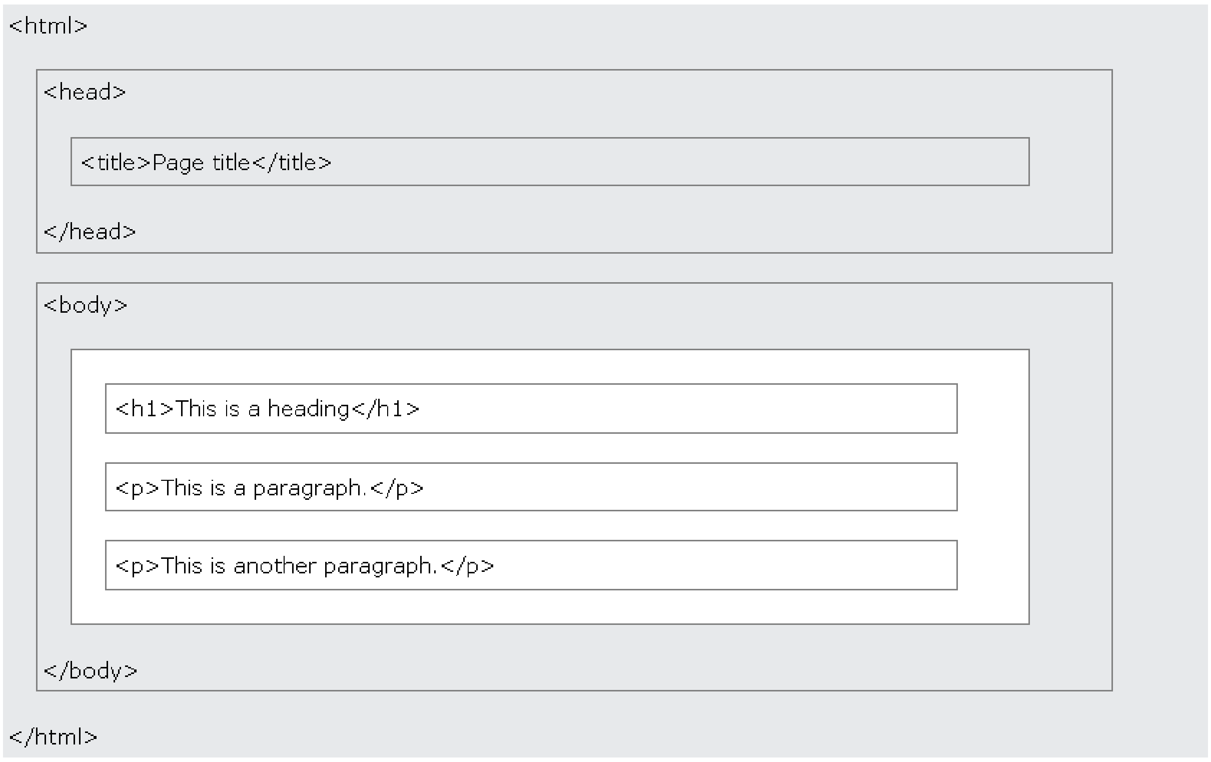
To write an HTML comment, wrap it in the special markers <!-- and →

```
<p>I'm not inside a comment</p>

<!-- <p>I am!</p> -->
```

HTML page structure

Below is a visualization of an HTML page structure


<html>
<head>
<title>Page title</title>
</head>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
</body>
</html>

References

[Getting started with HTML - Learn web development | MDN (mozilla.org)]

[Introduction to HTML (w3schools.com)]