Mozilla Developer Network Notes

Date: 28th March 2020

Contents

[1. Getting Started With HTML 1](#_Toc36470000)

[1.1. What Is HTML? 1](#_Toc36470001)

[1.2. Anatomy of a HTML Element. 1](#_Toc36470002)

[1.2.1. Nesting Elements 2](#_Toc36470003)

[1.2.2. Block Versus Inline Elements 2](#_Toc36470004)

[1.2.3. Empty Elements 4](#_Toc36470005)

[1.3. Attributes 4](#_Toc36470006)

[1.3.1. Adding Attributes to An Element 5](#_Toc36470007)

[1.3.2. Boolean Attributes 5](#_Toc36470008)

[1.3.3. Omitting Quotes Around Attribute Values 6](#_Toc36470009)

[1.3.4. Single or Double Quotes? 6](#_Toc36470010)

[1.4. Anatomy of an HTML Document 7](#_Toc36470011)

[1.4.1. Adding Some Features to an HTML Document 8](#_Toc36470012)

[1.4.2. Whitespace in HTML 10](#_Toc36470013)

[1.5. Entity Reference: Including Special Characters in HTML 10](#_Toc36470014)

[1.6. HTML Comments 11](#_Toc36470015)

# Getting Started With HTML

## What Is HTML?

HTML is NOT a programming language. It is a markup language that instructs the browser what content to show, and how it is to be structured.

The language itself depends on structuring content using a series of **elements** in order to enclose content and define their structure or function on the page. This is done via tags.

The MDN gives the following example. Take the following content:

*“My Cat Is Very Grumpy”*

In HTML, we can create a paragraph from this by writing the following in HTML:

<p>My Cat Is Very Grumpy</p>

Output:

My Cat Is Very Grumpy

**NOTE: Tags in HTML are case insensitive. For example, the title tag can be written as <title> or <TITLE> or <Title> or <TiTlE> and still operate correctly as HTML. HOWEVER all lower case is considered the standard practice for HTML.**

## Anatomy of a HTML Element.

HTML elements can be broken down into three components:

* **The Opening Tag (<>)**: Which consists of the name of the element opened and closed in angled brackets. This states where the element begins, and what content it’s effect is being applied to.
* **The Closing Tag (</>)**: Operate in a similar way to the opening tag, but includes a forward slash before the element name. This defines where the element ends. Failing to include the closing tag can create formatting errors on other content as it signifies where the effects of an opening tag should cease.
* **The Content**: The content of an element. More often than not, this will be text.
* **The Element:** The combination of all the aforementioned components.

Figure 1- Overview of a HTML Element. (Image Sourced From: https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction\_to\_HTML/Getting\_started)

### Nesting Elements

Elements can be nested within other elements. Using the previous grumpy cat example, we can say that the cat is **very** grumpy by nesting the content with an inner set of tags. For example:

<p>My cat is <strong>very</strong> grumpy. </p>

Output:

My cat is **very** grumpy.

Elements need to be closed in the order of which they appear. For example, the <strong> tag needs to be closed off before the <p> tag as it is the inner most element. Failing to do so results in the browser attempting to best guess what you were trying to implement, which doesn’t always yield the correct results. The following example demonstrates **INCORRECT** nesting:

<p>My cat is <strong>very grumpy. </p></strong>

### Block Versus Inline Elements

Elements in HTML can be separated into two categories. These are:

* **Block Level Elements**: These elements form a visible block on the page. The appear on a new line from the previous content stated in the HTML. Any content that follows on from a block element will also be placed on it’s own new line. These are very often the structural elements that represent things such as paragraphs, lists, navigation menus, footers so on so forth. Block level elements **can be nested inside** **other block level elements** but **can’t be nested inside inline elements!**
* **Inline Elements:** Elements contained within block-level elements which surround only a small portion of content. These will not form entire paragraphs and other groupings of content. Inline elements do not cause content to be placed on a newline, they simply carry on from the place they are positioned in text. Examples include the <a> (hyperlink) element, the emphasis <em> element or the bold <strong> element.

The following example demonstrates both types of element.

Inline Example:

<em>first</em><em>second</em><em>third</em>

Output: firstsecondthird

Block Example:

<p>fourth</p><p>fifth></p><p>sixth</p>

Output:

fourth

fifth

sixth

**NOTE: HTML5 has redefined the element categories. These definitions are more accurate and less ambiguous but slightly more complicated to understand than just “block” and “inline”.**

**NOTE: “block” and “inline” should not be confused with the CSS counterparts. Changing the CSS display type does not change the category of the element and doesn’t affect which elements it can contain or which elements it can be contained in. This was fundamentally one of the reasons why these terms were dropped from HTML 5, as they were often a source of confusion.**

### Empty Elements

Empty elements are elements that are made up of a single tag only. They are not followed by a closing tag. These elements are usually used to insert or embed something into the document at the particular place it is included. A good example is the <img> element, which embeds and image file onto a page at the position it is included in. For example:

<img src =<https://raw.githubusercontent.com/mdn/beginner-html-site/ghpages/images/firefox/icon.png>>

Output:



## Attributes

Attributes can be assigned to elements in order to assign extra information to the element that isn’t to appear in the actual content. For example, using the class attribute allows you to give the element a name which can then be used later to target the element with a particular style in CSS.

An attribute should have the following:

* A space between it and the element name (or the previous attribute, if the element already has one or more attributes).
* The attribute name, followed by an equal sign.
* An attribute value, with opening and closing quote marks wrapped around it.

The following example from the MDN demonstrates this:



Figure 2- An example of an attribute. (Imaged Sourced From: https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction\_to\_HTML/Getting\_started)

### Adding Attributes to An Element

An example of an element that can support attributes is <a>. This is the “anchor” element and allows any wrapped text to become a hyperlink. This element takes many attributes, however some of the more commonly used are:

* **“href”**: The href attribute specifies the web address that you want the link to point to; or where the browser navigates too upon clicking the element. For example, “href=<https://ww.mozilla.org/>”.
* **“title”**: The title attribute specifies extra information about the link such as what page the link sends the user to. For example, “title=”The Mozilla Homepage”. This will appear as a tooltip when the user hovers over the link on a website.
* **“target”**: The target attribute specifies the browsing context that will be used to display the link. For example, target=”\_blank” will display the link in a new tab. To display the link in the current tab, just omit this attribute.

The example below demonstrates how to use the <a> tag to set up a link that displays in a new browser tab:

<p><a href="https://www.bbc.co.uk" target="\_blank">A link</a> to my favorite website.</p>

### Boolean Attributes

Sometimes attributes don’t have values. This is allowed. These are called Boolean attributes, and they can only have one value, which is generally the same as the attribute name. An example of this is the “disabled” attribute, which can be assigned to form input elements. These can be greyed out (i.e. If you don’t want the user to use them)

For example:

<input type=”text” disabled=”disabled>

Alternatively, the above can be written as shorthand as follows:

<!—Using the disabled attribute prevents the end user from entering text into the input box -->

<input type=”text” disabled>

<!—The user can enter text into the following input as it doesn’t contain the disabled attribute -->

<input type=”text”>

### Omitting Quotes Around Attribute Values

Some markup style use attribute values without quotes. In simple elements with single attributes, this works fine, however it creates all sorts of issues when used in multi-attribute elements. **IT IS BEST TO AVOID THIS PRACTICE ALTOGETHER.**

An example of this style is as follows:

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

Using no quotes in this instance works fine, however, when we add a title to the anchor, we start to encounter issues:

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

In the above example, the browser misinterprets the mark-up. It thinks the title attribute is three attributes. A title attribute with the value “The”, and two Boolean attributes, Mozilla and homepage. HTML often tries to help the user by doing what it thinks the user intended and running markup no matter what, however, more often than not, it often assumes incorrectly and causes issues for the user.

### Single or Double Quotes?

In the above examples all attributes are wrapped in double quotes. Single quotes can also be used to a similar effect. There are no consequences for opting for either style, it all boils down to a matter of personal coding style. Both lines are equivalent:

<a href=”<http://www.example.com>”>A link to my example. </a>

<a href=’<http://www.example.com>’>A link to my example. </a>

Quotes should, however, not be mixed, as this causes errors. The following example yields errors:

<a href=”http://www.example.com’>A link to my example.</a>

In HTML, if the user has used one type of quotation mark to enclose an attribute value, the other type can then be used as part of the value itself. For example:

<a href=<http://ww.example.com> title=”Isn’t This Fun?”> A link to my example.</a>

Including a quotation mark inside an attribute where the same type of quotation mark has been used to enclose the attribute value will result in errors as the value is delimited prematurely as the quote inside the value is treated as a delimiter, not as part of the string. For example:

<a href=’http://www.example.com’ title=’Isn’t this fun?’>A link to my example.</a>

Instead, HTML entities (special values used to denote the presence of a special character) must be used instead. For example, a single quote can be displayed as part of the value using the following code:

<a href=’http://www.example.com’ title=’Isn&#39;t this fun?’>A link to my example.</a>

The code used is then recognised and the appropriate character is inserted as text.

## Anatomy of an HTML Document

The previous section covered what individual elements do, and how they are implemented in HTML. The following section focuses on how these elements can be combined together to produce pages. The following is an example of a basic webpage:

<!DOCTYTPE html>

<html>  
 <head>  
 <meta charset=”utf-8”>  
 <title>My test page</title>  
 </head>  
 <body>  
 <p> This is my page</p>  
 </body>  
</html>

Here we have:

1. **<!DOCTYPE html>**: This element is known as the Doctype and has very limited use in modern HTML documents, other than so far as to be required when producing them to make sure everything runs properly. The Doctype originated from the early days of HTML programming, where the element would include a link to a set of rules that the HTML page had to follow in order to be considered good HTML. They often provided automatic error checking and used to look something like:

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

However, these days doctypes are merely an artefact from days gone by. The only thing worth noting is that <!DOCTYPE html> is the shortest string of characters that counts as a valid Doctype.

1. **<html></html>**: The <html> element wraps all content of the entire page. It is known as the **root element**.
2. **<head></head>**: The <head> element acts as a container for all the stuff to be included on the webpage that isn’t content to be shown to the user. This includes things such as keywords, a page description that appears in search results, CSS to style content, character set declarations and more.
3. **<meta charset=”utf-8”>**: This element specifies the character set to be used for the document. In the above instance, it’s the “utf-8” character set which includes most of the characters from most human written languages. It can handle any textual content inputted onto a page with this element embedded in the head. As such, there is usually no reason not to set this, and it generally helps avoid problems further down the road.
4. **<title></title>**: The <title> element specifies the title of the page. This is the string that appears in the browser tab the page is loaded in, and is used to describe the page when a user bookmarks or favourites it.
5. **<body></body>**: The body element contains **ALL** the content that is to be shown to a web user accessing the page. This includes text, images, videos, games, audio or whatever else is to be delivered to the end user.

### Adding Some Features to an HTML Document

The following example details the basic layout of a HTML page:

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">   
<title>My test page</title>

</head>

<body>

<p>This is my page</p>

</body>

</html>

The above code has been rewritten to produce a webpage with slightly more detailed content:

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>My test page</title>

</head>

<body>

<h1>Jake's Page</h1>

<p><strong>Welcome to Jake's page.</strong> This web page has been produced as part of the <a href="https://developer.mozilla.org/en-US/" title="The Mozilla Developer Network Webpage" target="\_blank">Mozilla Developer Network's</a> tutorial series on HTML.</p>

<p>Here Is A Bob Ross Meme:</p>

<img src ="https://cdn.acidcow.com/pics/20181109/bob\_ross\_07.jpg">

</body>

</html>

### Whitespace in HTML

The HTML parser will reduce all space separating text down to a single space. This means the developer can insert large amounts of space, or space their code however they like, and the HTML file will still be readable. Tabs are nice ways of formatting HTML and ensuring the code is as readable as possible.

For example, the following markup is valid, albeit, badly formatted:

<p>Dogs are silly.</p>

<p>Dogs are

silly.</p>

## Entity Reference: Including Special Characters in HTML

In HTML, the characters <, >, “, ‘ and & are all special characters which are included in the HTML syntax to perform some kind of operation. Trying to include them into string literals as they are normally, results in the characters being implemented as code and not text.

To get around this, HTML supports character references. These are special codes that represent characters and can be represented by and ampersand, followed by the character code, which is then terminated by a semicolon (;).

The following list defines the literal character and their equivalent character reference:

|  |  |
| --- | --- |
| Literal Character | Character Reference Equivalent |
| < | &lt; |
| > | &gt; |
| “ | &quot; |
| ‘ | &apos; |
| & | &amp; |

Character references closely resemble the name of their literal character representation. More examples can be found on the HTML Wiki.

The following examples demonstrates how to use character references. The first example is incorrect and results in a new paragraph element wrapping “element”. The second line is correct and outputs “<p>” in the text:

<p>In HTML, you define a paragraph using the <p> element.</p>

<p>In HTML, you define a paragraph using the &lt;p&gt; element.</p>

## HTML Comments

HTML supports the use of comments. These are notes written by the programmer which allow them to annotate their markup and explain how it functions without being processed by the browser.

Comments can be implemented by using “<!—( comment) -->”.

The following is an example of how to implement comments:

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

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

The first line of code will be read by the browser and output a paragraph. The second line will not be read by the browser as it is a comment. No paragraph will be output.

# What’s in The Head? Metadata in HTML

The head of an HTML document is the part that is not displayed in the web browser when the page is loaded. It contains information such as the page <title>, links to the CSS (if the content is styled), links to custom favicons, and other metadata (data about the HTML, such as the author, and important keywords that describe the document.). The following section details what the head is and the various elements used within to define the meta-data needed to support webpages.

## What is the HTML Head

Take the previous example given in the last section:

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>My test page</title>

</head>

<body>

<p>This is my page</p>

</body>

</html>

The HTML head is the contents of the <head> element – unlike the contents of the <body> element (which are displayed when the webpage is loaded in a browser), the head’s content is not displayed on the page. Instead, the head’s job is to contain metadata (data that describes data) about the document. The above example shows a document with a small head:

<head>

<meta charset="utf-8">

<title>My test page</title>

</head>

In larger pages, the head can become quite sizable. The following section, however, does not aim to detail every element that can be used in the head. It is there to explain the fundamentals elements that provide the major details and components most users expect when using a website (such as a title for the webpage).

## Adding a Title

As previously mentioned, the <title> element is used to add a title to the document which is displayed in the browser tab. This is NOT the same as the <h1> element, which is used to add a top level heading the document