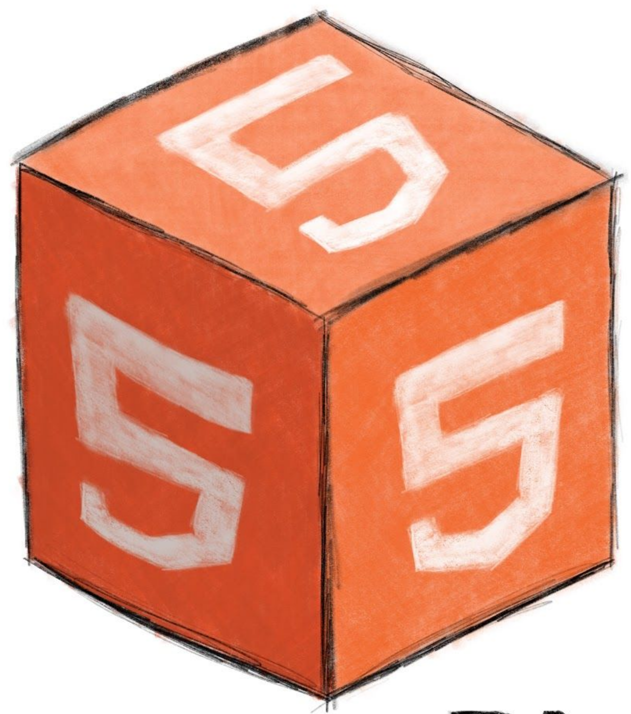
HTML



Beginner Blocks

HTML Beginner Blocks
Version 1.0
Written by Jared York

Introduction

This course is intended for those wanting to read and write a little bit of code, to those who wish to build the front-end to huge enterprise website. Learning HTML is so important, and isn't that difficult at all to learn. It really is the fundamental building blocks of the web.

HTML stands for *Hypertext Markup Language*. Markup languages allow you to define elements, or the structure of a page. In regards to HTML, you are able to define the title, page content, paragraphs, headers, footers, etc. HTML is not meant to be a language for styling a website, for that, CSS is needed. Since this course is meant for interaction, feel free to open up a text editor or code editor of your choice. Even Notepad or Text Edit will work just fine! Use what is most comfortable for you. The first step to writing HTML is defining the file as an HTML file. Let's create a new folder, name it anything you wish, then save the file as index.html. The .html extension is important here. The name, "index", can be anything you wish, however the index file is usually the default homepage for most websites out there. It's a great idea to get used to good practices, no matter what programming/scripting language you use. Once you have saved the file, we will need to add our first bit of code to tell the browser this is an HTML document:

```
<!DOCTYPE html>
<html>

</html>
```

There we go! Fantastic, we have now defined our file to be interpreted as an HTML file. The first line of an HTML file should be <!DOCTYPE html>, which is the HTML5 syntax for defining the file as HTML. HTML, XML, and many other markup languages use what are known as an *element*. Elements are generally defined with an *opening tag* and a *closing tag*. In the above bit of code <html> is known as an opening tag. The line </html> is a closing tag. In many cases, you can also add elements between an opening and closing tag. Let's put this in action. To start, add a new opening and closing tag named "head", within the HTML tags:

```
<head>

</head>
```

We will want to define the charset and the language of the HTML document. Inside the head tags add the following two lines:

```
<meta charset=”UTF-8”>
<meta lang=”en-US”>
```

This will tell the browser to interpret this HTML file with the UTF-8 character set, and also specify we want it parsed as using US English. Many language tags can be found on this Wikipedia page. Meta tags are known as empty elements, because they don't use closing tags. Inside the opening tag for the meta element, we set charset to "UTF-8". Charset is what is known as an *attribute*. Attributes allow you to set properties of an HTML element, and are defined in the opening tag of an element. There may be multiple attributes that can be set, all of which are to be added to an opening tag of the element being added. We can also add a title to our page with the <*title*> tag, and add a title between the <*head*> opening and closing tags:

```
<title>My Super Cool Website</title>
```

For now, we are finished working with the *head* tags. Let's create a new element after the *head* element, still within the HTML tags, named "body".

Our HTML file should now look like so:

```
<!DOCTYPE html>
<html>
        <head>
                <meta charset="UTF-8">
                <meta lang="en-US">
                <title>My Super Cool Website</title>
        </head>

        <body>

        </body>
</html>
```

Websites generally have text. Most of the time, text is added to a website via paragraph tags, or <*p*> tags as defined in HTML. Text can be added between an opening and closing <*p*> tag. Let's add the following between the body tags:

```
<p>Hello world!</p>
```

If we open our HTML file with the browser, we should see the text:

Hello world!

There we have it! Though, what if we want to bold the word "Hello"? Not a problem! Encapsulate the world "Hello" with an open and closing <*b*> tag. Let's run the page. We should now see:

Hello world!

Perhaps we also want to italicize the word, “world”. Just add an opening and closing <*i*> tag around “world”. If we refresh the page, we should see:

Hello *world*!

Maybe we even want to underline both words, while maintaining the bold and italicized styling. Between the opening paragraph tag and the opening <*b*> tag, add the opening tag <*u*>. After that, just before the closing <*p*> tag, let’s add the closing underline tag, <*ul*>. The whole line of code relevant to our text, “Hello world!” should appear as:

<p><u><b>Hello</b> <i>world</i>!</u></p>

Refreshing the browser, the text should be displayed as:

<u>**Hello** *world*!</u>

At some point, you will probably want to add a line break. Let’s try adding one. First, add another paragraph element after the previous one we made:

<p></p>

Between the opening and closing tags, copy and paste the following paragraph:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam et ultrices elit. Pellentesque non est vel tellus cursus rutrum non in libero. Curabitur ut eros nec massa scelerisque mollis nec non odio. Vestibulum dapibus odio eget feugiat tempor.

After the third sentence, add a <*br*> tag. This is the line break tag, and is another empty element. This will insert a line break just after the third sentence. It is discouraged to use multiple line breaks in a row to create separate paragraphs, however. Set the margin of multiple paragraph elements to add spacing, rather than using line breaks to create separate paragraphs. This will allow a much more seamless experience for users using screen readers.

Awesome! Now what if we want to add a heading to our page? That’s no problem either. There are six sized headings accessible via the tags <*h1*> through <*h6*>. The element <*h1*> is the largest heading, and <*h6*> is the smallest.

Let’s try adding an <*h1*> heading. Just before our paragraph tag, add the following line:

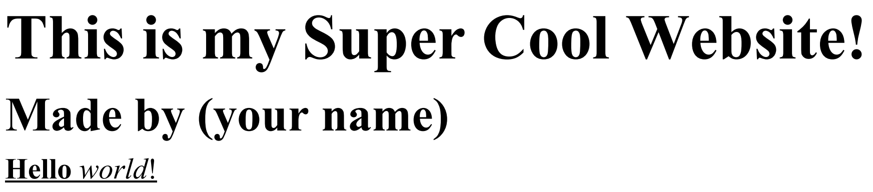
<h1>This is my Super Cool Website!</h1>

If we refresh the page, we should see a large, bold, title displaying “This is my Super Cool Website”, with our “Hello world!” text under it.

Next, let’s try using a smaller heading like *h3*. Add the following line right underneath our *<h1>* tags:

```
<h3>Made by (your name)</h3>
```

If we refresh the page we should see something like:



This is my Super Cool Website!

Made by (your name)

Hello *world*!

At some point or another, you will probably want to add a list to your website. There are two types of lists in HTML: unordered lists, and ordered lists. First, let’s start with unordered lists. An unordered list can be defined using the *<ul>* tag. Items in a list are defined with *<li>* tags and are placed between the opening and closing *<ul>* tags.

Let’s try it, at the bottom of our document’s *body* element, add the following to create an unordered list:

```
<p>Ingredients:</p>
<ul>
        <li>butter</li>
        <li>flour</li>
        <li>baking powder</li>
        <li>salt</li>
        <li>sugar</li>
        <li>vanilla extract</li>
        <li>milk</li>
</ul>
```

If we refresh the page, we should see our list with bullet points next to each item. We can change the symbol next to each item of the list using an inline CSS style. In our *<ul>* opening tag, add the following attribute:

```
style=”list-style-type: square;”
```

There are a few values commonly used for the *list-style-type* CSS property:

- disc

- circle
- square
- None

We can define inline CSS styles in the opening tag of any element. However, inline CSS styles are discouraged, and it's preferable to link CSS stylesheets within the *head* element. For our purposes, it'll be fine to use an inline style for our list.

Let's try it out! Perhaps we want a list of items for dessert. Add the following code:

```
<ol>
        <li>cake</li>
        <li>cookies</li>
        <li>coffee</li>
</ol>
```

If we navigate to the page in the browser, we should see:

1. cake
2. cookies
3. Coffee

Got it? Great! Feel free to experiment setting the *type* attribute of our ordered list. The following values can be assigned to the type attribute:

- 'a' uses lowercase letters
- 'A' uses uppercase letters
- 'i' uses lowercase Roman numerals
- 'I' uses uppercase Roman numerals
- '1' uses numbers (default value)

Lists can also be nested. In other words, lists can be inside lists, effectively allowing for multiple levels of items in a list. Let's try creating a nested list! Add the following after the previous order list:

```
<ol>
	<li>first item</li>
	<li>second item</li>
	<li>third item
		<ol>
			<li>first subitem</li>
			<li>second subitem</li>
			<li>third subitem</li>
		</ol>
	</li>
	<li>fourth item</li>
</ol>
```

Refreshing the page, we should see our nested list!

The other type of list, ordered lists, are quite similar to unordered lists. Ordered lists are defined with the *ol* element. The method of ordering items can be specified with the *type* attribute. If the *type* attribute is not specified, items will be numbered starting from one.

What if we want to add an image? I like donuts. Cinnamon donuts. So I found a public domain photo of one. Let's add it to our page! Adding images is as easy as using the <*img*> tag, an empty element, with two attributes: *src*, and *alt*. The source attribute requires a path to the image, either relative, or absolute like a URL. Instead of downloading an image and tossing it in our HTML file's directory or another child directory such as an images folder, we can using the URL:

https://images.unsplash.com/photo-1534380615157-f5df9a796818?ixlib=rb-1.2.1&ixid=eyJhcHBfaWQiOjEyMDd9&auto=format&fit=crop&w=400&q=60

Let's append the following line to the bottom of the body element:

```
<img
src="https://images.unsplash.com/photo-1534380615157-f5df9a796818?ixlib=rb-1.2.1&ixid=eyJ
hcHBfaWQiOjEyMDd9&auto=format&fit=crop&w=400&q=60" alt="Donut!">
```

Don't worry about the line breaks above, it can all be on a single line in your code. If you refresh the page, you should see the usual text we've added previously, in addition to the awesome donut image. (Thanks Charles "Duck" Unitas for taking the photo!) The *alt* attribute allows you to specify text to be displayed in place of the image if it fails to load.

Usually, when a website is created, elements usually aren't just dumped between the opening and closing <*body*> tags. Many web developers use <*div*> tags to create a sort of "container" to add elements to. Elements such as *div* provide a way to group elements in a section together.

These containers can also be nested inside each other. Nesting is pretty much putting an element inside another element. There can be as many levels of nesting as you wish, though after too many levels, code can be very hard to read.

Another common element to utilize on websites, are tables! Tables can be very useful for displaying data. There are usually two elements inside a *table* element, *thead*, and *tbody*. The *thead* element represents the table header. The *tbody* element is used to represent the body of the table. Table bodies are made up of rows, and the table header can contain a row to show column headers. Table rows are defined with the *tr* element. Values inside a table cannot be directly added within a *tr* element, but rather inside a *td* element, representing table data. Table data is a single cell that appears under the corresponding column header. You can have as many *td* elements in a table row as you have columns. Adding an example to our HTML file might make more sense. Let's say we want a table keeping track of the weather conditions on the first week of March. Add the following to create our table, table header, and our table data:

```
<table>
        <thead>
                <tr>
                        <td>Date:</td>
                        <td>Weather Condition:</td>
                </tr>
        </thead>

        <body>
                <tr>
                        <td>3/1/19</td>
                        <td>Sunny</td>
                </tr>


                <tr>
                        <td>3/2/19</td>
                        <td>Partly Cloudy</td>
                </tr>

                <tr>
                        <td>3/3/19</td>
                        <td>Scattered Showers</td>
                </tr>

                <tr>
                        <td>3/4/19</td>
                        <td>Mostly Cloudy</td>
                </tr>

                <tr>
                        <td>3/5/19</td>
                        <td>Partly Sunny</td>
                </tr>

                <tr>
                        <td>3/6/19</td>
                        <td>Light Rain</td>
                </tr>

                <tr>
                        <td>3/7/19</td>
                        <td>Mostly Sunny</td>
                </tr>
        </body>
</table>
```

If we refresh the page, we should see at the bottom of the page two columns with date and weather conditions. We should see the column of dates filled with dates, and their corresponding weather condition next to each date. With a lot more data, it could be difficult to read the table. To fix this, we can prettify the table a bit. In the *head* element, let's add an opening and closing *style* <tag> under our <*title*> tag. Inside these <*style*> tags, we can add CSS rules. We can add a rule affecting tables, in which we can collapse borders around each table cell into a single line. So far, we don't see any border around each table cell, but we will add that shortly. This is another method to add styles to elements. Let's add the CSS rule for tables:

```
table {
        border-collapse: collapse;
}
```

Next, we can add our CSS rule for table data elements. Inside our CSS rule, let's add eight pixels of padding in each table data, and a single pixel black border around each cell.

Let's finish up by learning how to create a link. Links can be added via an anchor tag, in code specified as an <*a*> tag. Let's add a link the directs to google.com. Add the following line to the bottom of the <*body*> element:

```
<a href="https://www.google.com">Go to Google</a>
```

If you want the link to open a new tab when clicked, you can add the *target* attribute, and set it to _*blank*.

Remember when I explained <*div*> elements? Let's add one to wrap around all elements betweens the body tags. Add an opening <*div*> tag between the opening <*body*> tag and the opening <*h1*> tag. We will also have to add a closing <*div*> tag after the closing anchor tag and just before the closing <*body*> tag.

With that, you now know how to read and write the majority of HTML you will encounter in the real world! If you have any questions, suggestions, or general feedback about this course, feel free to contact me via email, or tweet at @jaredyork_. I'd love to hear what you're able to build with HTML after utilizing this course! Also if you found this course valuable, and would like to receive future news and updates on future courses I create, be sure to fill out the form.

The full source code for this course can be found on GitHub.