Intro to Modern HTML/CSS

By Hayden Walker (www.haywalk.ca)

This guide to HTML/CSS was born out of frustration about the lack of tutorials that include modern best practices. This guide is brief, but it only includes best practices and absolutely will not send your browser into quirk mode.

Table of Contents

Some Background

Before we begin to write webpages, some background may be useful.

What is HTML?

HTML is what is known as a *markup language*. In fact, HTML stands for *Hypertext Markup Language*. **Markup languages are not programming languages**, because all they do is define the formatting for documents. Another example of a markup language is *LaTeX*, which is used for paper documents.

HTML is the standard markup language for pages on the Internet that include *hypertext* (hence the name). Hypertext is text displayed on a computer that includes references (links) to other pages on the Internet.

What is CSS?

CSS stands for *Cascading Stylesheets*. While HTML defines the content of a webpage and tags things like headers and paragraphs, it does not define how things are *styled,* i.e. how they look. In the past, there were HTML tags for changing font colours, etc., but these are no longer acceptable to use. They usually still work, but will throw your browser into the dreaded *quirk mode*. *Quirk mode* is covered in the next section.

CSS has the added benefit of being able to change styling on multiple pages. That means that if you had a website with any number of pages that all used one stylesheet, changing the font colour in that stylesheet will change it on every page on your site. We will cover creating and writing stylesheets, and linking them to webpages.

Why Change my Practices?

If you don't already know HTML/CSS, you can gloss over this section.

Someone who has been writing HTML for any amount of time may read this guide and ask: "Why should I use CSS to centre text instead of using `<center>`?" or "Why should I use CSS to colour my background red instead of using `<body background='red'>`?"

The fact is that these tags are now considered bad practice, and will send your browser into what's called "quirk mode," where styling and markup aren't interpreted correctly because your browser is trying to 'guess' what you mean. The only reason these tags work is to maintain compatibility with old sites from before modern standards existed.

If you have a website, you can enter its URL into the W3C Markup Validation Service. If your site uses tags like `<center>` and `<font>`, among others, the validator will raise an error.

Making a Blank Webpage

In this section, we will get started by making our first blank webpage!

Start by making a folder to keep your site in. It can be named whatever you'd like. Now, open your preferred text editor (this must be a raw text editor, like Notepad or Vim, and not a word processor like Microsoft Word or LibreOffice). Create a blank file and save it as `index.html` in the folder that you just made. Create another blank file and save it as `style.css` in the same folder. For now we'll only be working with `index.html`. It is important to make sure that neither file has `.txt` at the end – in Notepad, make sure to select "all file types" when saving.

Go back to editing `index.html`. This will become your site's homepage! A homepage is always called 'index' because it is the 'index' of all the other pages on your site. In your index file, write the following lines at the top (we will go over what they mean in a minute):

```
<!DOCTYPE html>

<html lang='en'>

</html>
```

What does this mean? We'll start from the top. `<!DOCTYPE html>` is a statement that tells your browser that the following document is in standard HTML. `<html>` is the tag to *open* (i.e. begin) that HTML document, and when we add `lang='en'` inside that tag, it means the document is going to be in English. **These tags may seem redundant, but failure to add them will result in errors, and your browser will enter quirk mode.**

Now, if you save that page and open it in a browser, you will see a blank webpage. The goal of this section was to make a blank webpage, but we aren't finished yet. We need a couple more lines to bring our site up to standards, and to add a title and link our stylesheet. Add the following lines between your `<html>` tags (be sure to indent):

```
<head>

     <title>My Website</title>

     <link rel='stylesheet' type='text/css' href='style.css'>

     <meta name='viewport' content='with=device-width, initial-
scale=1.0'>

     <meta charset='utf-8'>

</head>
```

Now, what do these new tags do?

`<head>` is what's known as the 'header' of your document. It lays the foundation for the rest of the page by changing settings, defining metadata, linking to other documents, and giving the page a title. Like the `<html>` tag, this one gets closed at the end with a `</head>` tag.

The first tag inside your header, `<title>`, may seem obvious – it defines your site's title, which appears in your browser tab! The next tag, `<link>` (with additional parameters), tells the HTML document that it will get its styling from the `style.css` document we made earlier.

The next two are `<meta>` tags, each with different parameters. The first of the two will make your site scale depending on if it's being viewed from a computer or a phone (so that it looks good on both), and the next tells the browser that the document is written in the UTF-8 character set.

"What does that mean?", you may ask. A document written in UTF-8 can contain special characters (like 'é' and 'ß') that aren't necessarily from English. Failure to tell your browser that your document is in UTF-8 can result in special characters appearing as jumbled boxes, and **regardless of whether or not you have any**, it will definitely raise an error with the W3C validator that was mentioned in the previous section.

But now, we have finished our first blank webpage, and it is completely up to standard. Congratulations!

Adding Content

Now that we've laid the foundation for our webpage, we can begin to add content. Inside your HTML tag and below your header, add the following lines:

```
<body>
</body>
```

These two tags open and close the body of your page. Everything added between them will appear as content.

Adding Headings

Just like in this document, different content is separated by headings. Headings are important for the user to be able to understand your page's content, but they're also important for search engines trying to look for keywords on your page.

A heading is put between two heading tags (one to open the heading, and one to close it). HTML has six different heading tags, with `<h1>` being the largest and most important and `<h6>` being the smallest and least important. The main title of your page should be formatted as `<h1>`, sections should be `<h2>`, subheadings `<h3>`, etc., all the way down to 6.

Let's add a heading to our document, by adding this line between our body tags:

```
<h1>Welcome to my Website!</h1>
```

When you save the file and refresh your browser, you'll notice that "Welcome to my Website" is now displayed in large text. That's your first heading! Now let's add a subheading. Add the following below your first heading:

```
<h2>Let's Write a Paragraph!</h2>
```

Refresh again, and you'll now see a subheading just below your page title.

Adding Paragraphs

Just like headings, HTML also has a tag for paragraphs. When you're writing a paragraph, put it between `<p>` tags. Let's add a short sentence just below our second heading:

```
<p>This sentence is properly formatted!</p>
```

Some people neglect to put their body text into tagged paragraphs, **which is bad practice.** Now that we have a pretty good idea about how paragraphs work, let's add another heading below our paragraph, and start another section. Add this:

```
<h2>Adding Images</h2>
```

Refresh the page and you'll notice your sentence and a new heading.

Adding Images

For this section, you'll need to find an image to add to your website. This can be anything, but nothing too large, as we haven't gotten to image scaling yet. Also take note of the image's extension (`.png`, `.jpg`, etc.). For the purposes of this guide, it will be assumed that the image is a `.png` file.

Now, in the same folder as `index.html` and `style.css`, make a subfolder called `images`. Save/move the picture you've chosen into our new folder, and for simplicity's sake, name it `image1.png`.

Back in `index.html`, under the heading we just added, add the following lines:

```
<img src='images/image1.png' alt='My Image'>

<p>This is my image!</p>
```

What does this line mean? First, the `<img>` tag defines an image, and does not have to be closed. `src` means 'source', i.e. the path to the image file, which is `images/image1.png`.

The `alt` attribute tells the browser what to display if the image is missing or unable to be loaded. That means that if, for whatever reason, the image you chose is unable to be displayed, the browser will display 'My Image' instead. **Not having an `alt` for an image will raise an error.**

Hyperlinks

Linking between pages

As previously mentioned, HTML is the *Hypertext Markup Language,* and hypertext is all about links.

Before we can link to another page on our site, we need to make one. Copy your `index.html` file and name the copy `page2.html`. Edit it and change the title (in the document header) to "Page 2", and get rid of everything between the body tags. Now we have a new blank page to link to!

Back in your index file, add the following line to make a link to the new page:

```
<a href='page2.html'>Page 2</a>
```

An `<a>` tag defines a hyperlink, and anything between the opening and closing `<a>` tags acts as one. By putting "Page 2" inside those tags, it will be our link. The `href` attribute tells the link where to send someone who clicks on it. In this case, it's sending them to `page2.html`.

Now if you refresh the page, you should see a link that, when clicked on, takes you to our (still blank) second page!

On the second page, you can make a button to take you back to the homepage by applying the same principles. Add this line:

```
<a href='index.html'>Home</a>
```

External Links

The same process applies to external links, but instead of adding the path to a page, you use a URL. Let's add a link to the W3C Validator! Add the following lines to your homepage:

```
<br>

<a href='https://validator.w3.org'>W3C Markup Validator</a>
```

When you save and refresh, you should see that link and it should take you to the W3C Markup Validator. The `<br>` tag means add a new line. Since this isn't really body text, it doesn't need to be in a paragraph.

Styling your Pages

Now that we have some content, we'll begin to add some style to the pages we've made by setting some style 'rules'.

Setting Colours

In CSS, rules are made for different tags. If we want to change the background and text colours of our page body, then we need to make a rule for the `<body>` tag. Let's start by making our background black and our text white, as that's an easy-to-read colour combination.

Edit `style.css` and add the following lines:

```
body{
	background-color: black;
	color: white;
}
```

We've just made our first style rule, which applies to the body tag. `background-color` tells the browser what colour the page's background should be, and `color` always defines the *text* colour. You can also use hexadecimal colour codes (Like `#FF0000` for red) to pick colours instead of using words. You can find many colour codes online.

When you save the file and refresh both pages, you should notice that the backgrounds are now both black, and that the text on both pages is now white.

We can also change the colour of specific elements. Let's make all the links lime green by adding the following rule:

```
a{
	color: lime;
}
```

We can also make the colour of the links change when we hover over them, by adding this rule:

```
a:hover{
     color: red;
}
```

The colon in that rule means it only applies to links being hovered over.

Centring Text

In the past, there was an HTML tag called `<center>` that could be used to centre text, but that is no longer supported in HTML 5 and will raise an error in the markup validator. Instead, we can use CSS! Let's use CSS to centre our title heading. Let's also change its colour – why not red? Add this rule:

```
h1{
     text-align: center;
     color: red;
}
```

Using Classes

Instead of making rules that apply to all of a certain tag, you can use classes to make groups of items that follow the same rules.

Let's make the first paragraph a different colour. Why not pink? Start by changing the line so it looks like this:

```
<p class='firstparagraph'>This sentence is properly formatted!</p>
```

This makes that paragraph part of the class `firstparagraph`. Now, in your stylesheet, add the following rule (note the dot at the beginning):

```
.firstparagraph{
     color: pink;
}
```

Now, that text should be pink, but the rest should still be white.

Working with Images

Scaling Images

What if you want to add a picture to your document, but it’s absolutely massive and takes up the whole page? You just need to use CSS to scale it! A good-practice rule with images is to always include this rule in your stylesheet:

```
img{
     max-width: 100%;
}
```

That rule makes it so that any image that is wider than the screen it’s being displayed on, will be scaled down to fit. You can also set use classes to size specific images. With images, you only need to set its width, as the height will adjust automatically.

Wrapping Text Around Images

To wrap text around an image with CSS, we use the `float` property on the image, to make it ‘float’ to one side or the other. The text will automatically wrap, but we should add some margins so that it doesn’t get too close. Let’s make all images stay on the right, with text wrapping around them. Change your `img` rule:

```
img{
     max-width: 100%;
     float: right;
     margin-left: 10px;
}
```

This will make images on our site float to the right, and leave 10 pixels of space between the image and any text.

Centring the Whole Page

If you want to have your content centred on the page, you can do that with CSS. Add the following statements to your body rule:

```
max-width: 800px;
margin-left: auto;
margin-right: auto;
```

Conclusion

Now you have most of the tools you need to make and style a webpage. Now you can apply them and begin to customize your site.

More Resources

If you're looking for more advanced guides to HTML/CSS, check out w3schools.com.

Copyright

Revision History

- Written 30 September 2020 by Hayden Walker (www.haywalk.ca)
- On 2 October 2020, Hayden Walker (www.haywalk.ca) changed the section *Using Classes*, *Wrapping Text Around Images*, and *Conclusion.* Comment: “Made the guide more straightforward.”