Intro to HTML/CSS

Table of Contents

[History and Tools](#_bx04tewmnuqh)

[HTML](#_q53cst8ybnjl)

[Tags](#_utn0jrvyk6sc)

[Paragraphs](#_ds49g6815wkf)

[Lists](#_1tx6u5yfmpb8)

[Links](#_9koz54g5nvly)

[Images](#_rjte3z5beehe)

[Block and Inline Tags](#_a22p1ljcbu2u)

[Semantic HTML5 Tags](#_l8jhz3e8llyo)

[CSS](#_fmiuvy2yr8zp)

[Selectors](#_ydoh3mt6skur)

[Properties](#_nq19zl9uijse)

[ID & Class](#_7rur9behj0vq)

[Pseudo Classes](#_3g5kxcksxaz8)

[CSS Box Model](#_wmmdu4y3sdm3)

[Shadows and Border Radius](#_riovv48et08m)

# History and Tools

**Hypertext Markup Language (HTML) and Cascading Styling Sheets (CSS) are 2 of the 3 main components in Web Development**. Both components together make webpages look presentable on the web.

**Every single webpage that you see on the World Wide Web, is written using HTML and CSS and Javascript**

We can write HTML and CSS to create web pages using a simple text editor. Some of the more popular editors are:

* **Sublime**
* **Atom**
* **VS Code**
* **Notepad++ (This is the extra editor)**

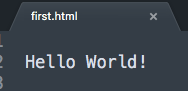
The other tool that we need is a web browser. Examples of web browsers are:

* **Chrome**
* **Firefox**
* **Edge**
* **Safari**
* **Opera**

**It’s a good idea to have a few browsers available for testing. Many times web pages will look one way in one browser, and quite different in another. In this course we’ll look at ways where we can mitigate and have some control over this.**

# HTML

**HTML is a markup language that uses tags to define its content. We use the browsers to interpret our HTML and render a viewable page**.



If we created an HTML page (e.g. first.html) in our editor, and just typed “Hello World”, the browser would still be able to display our content, even though there’s no actual HTML code.

## Tags

**Tags are used to define the content of our page**. If we added an **h1 tag**, for example, the browser would interpret our HTML differently when viewed. **Note that there is an opening (<h1>) *and* closing h1 tag (</h1>). This means that any text within these** opening and closing tags will be styled as an h1 tag as below.

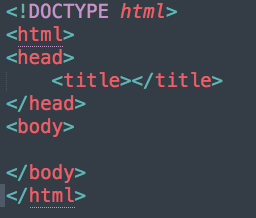


*The view in the browser*

**When the browser opened our HTML, it saw the <h1> tag. All tags have default formatting/styles applied to them. That may be font colour, size, height, etc**… **The h1 tag has styles that make it bigger and bolder than regular text. Also, there are 5 more heading tags (h2 to h6), with each getting smaller in size i.e. h6 tag has the smallest text size.**

There are many HTML tags, and we won’t be covering all of them here. But we’ll use some of the more common ones. Use [this reference](https://htmldog.com/references/html/tags/) to see all of the tags in HTML.

HTML is a very structured language**. Here’s an example of a basic HTML shell:**



Let’s run through these tags.

**The <!DOCTYPE> tag defines which version of HTML we’re using. By saying *html* that tells the browser we are using HTML5, which is the latest finalized standard**. This tag appears only once and at the very first line in the HTML code.

**The <head> tag is where we describe the content of our page. Or we can say it holds data about data (metadata). e.g. within the <head> tag we can specify the <title>, <link> ,<meta> tags etc.** The metadata is **not displayed on the web page**. These tags only hold data about the webpage. **It handles all the things in the Backend**

**What is Backend**

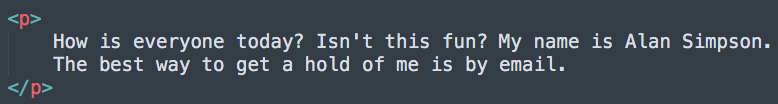
**The backend is the non-visible part of an application, website, or software system that handles the behind-the-scenes operations that make it work.**

After the closing </head> tag, **we see the <body> tag. This tag is where everything that is to be displayed in the browser is to go. So the <h1> we created earlier would have to go inside this <body> tag.**

You can see that there is a parent/child structure to HTML.

## Paragraphs

**Paragraphs of text are one of the most common components of any webpage. To define a paragraph, we use the <p> tag.**

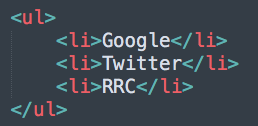


**When displayed in the browser, it has space above and below, like the <h1> tag, but the text is a normal reading size. These are the default styles for a paragraph**.

## Lists

**Lists are another commonly used tag in HTML. There are 2 very common ones: the <ul> and <ol> tags**. The first is an unordered or bulleted list, and the second is an ordered or numbered list.

Here’s an example of an unordered list and what it looks like in the browser:

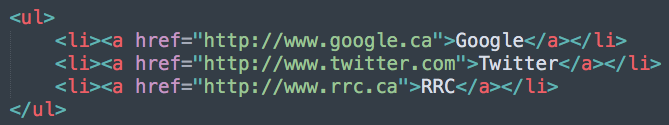


**Notice that the list has child tags; the <li> tags. <li> tags are used in <ol> tags as well. <li> stands for *list item*.** These have to be used to define each individual point in the list.

## Links

If we had a list like the one above, you’d expect to be able to click on them and be taken to that website. To do that we need to create *hyperlinks.*

**Hyperlinks are created using the <a> tag, which is short for *anchor.*** Unlike other tags we’ve seen, this one doesn’t have space above or below, rather **it stays inline** with its surrounding content.



We can Observe that here <a> Tag is done and inside the <a> Tag we put a reference for the <a> tag.

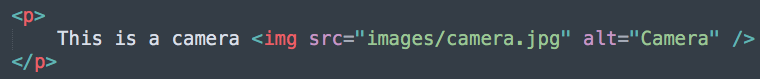
*Brower results*

**Again, notice the closing tag and how it’s used. If we didn’t have a closing tag for the first <a>, then all the following content would be a hyperlink to the google.ca website.**

Also notice the styles. **The blue links are unvisited links, and the purple one means that I’ve visited that link before**. This is the **default style for <a> tags.** We’ll talk more about changing these styles later on.

## Images

**Using images in pages is a great way to break up content and make it more visually appealing for the reader**. To add an image in HTML**, we use the <img> tag**, but how we use it is a little **different** than what we’ve seen.



*As viewed in the browser*

Let’s start with the HTML. [Attributes](https://htmldog.com/references/html/globalattributes/) are commonly used in HTML to further describe our content. Here we have 2 attributes: src and alt. The src attribute defines where the image is located (a path) that we’re going to display, and the alt attribute (which is required) defines what text will be displayed in the case that the image can’t be displayed.

**Notice in the src attribute that the path is relative. This is very important and should always be used in web development.** Here, we’re referring to an image located in an *images* folder located at the same level as our webpage.

**One more note about the <img> tag, notice that it doesn’t have a closing </img> tag. That’s because there is no content that would be needed inside the opening and closing <img> tag**. Everything needed is defined within its attributes**. So to close the tag a forward slash (/) is put at the end of the <img> tag.** These are called **empty tags**, and there are a few others, but <img> is the most commonly used one.

## Block and Inline Tags

**As we’ve seen now with <a> and <img>, these tags don’t have styles that put space above and below them. They are *inline* tags. They don’t do any kind of line breaking.**

**Most tags are *block* tags.** This means that they put space above and below them, such as the <h1> and <p> tags that we’ve seen previously.

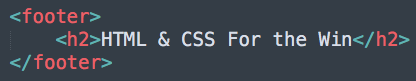
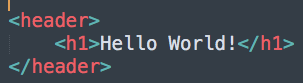
**This is an important concept to remember, and something we’ll work with more when we get into CSS.**

## Semantic HTML5 Tags

Previously to HTML5, tags were used that often didn’t do a good job in describing their content. For example, <div> tags were used whenever you needed a block tag. This meant that your markup was very hard to read, as you couldn’t easily differentiate one section of your page to another one.

With HTML5, a major goal was to create tags that were more descriptive of their content, thus the term *semantic tags*. There are quite a few new tags, and you can [read more about them here](https://www.bitdegree.org/learn/html5-semantic-tags), but for now we’ll start with some of the more common ones. As the course progresses we’ll use more as well.

<header> and <footer> are HTML5 semantic tags that are commonly used to define areas at the top and bottom of our pages.



These tags don’t really do anything visually, but what they do is make our markup easier to read and more semantic.

# CSS

HTML is responsible for the data and also describes the structure of the page, but all the styles/formatting should be defined in Cascading Style Sheets (CSS). The syntax is different from HTML, but they work together to present a webpage.

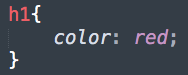
To start, create a new file with a .css extension. This is an external style sheet, where ALL styles should be stored. A common name for stylesheets is *styles.css*.

But first, we have to let the HTML know that there is a CSS file. We do this by using the <link> tag in our HTML. The <link> tag has to go within the <head> tag.



[There are a lot of CSS properties](https://htmldog.com/references/css/properties/) that we can use, so as the course progresses we’ll look at more and more.

## Selectors

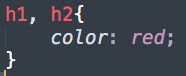
**CSS uses *selectors* to define which tag is supposed to be styled. If we want to style an <h1>** for example:

*The selector*

*The property:value*

**Here we’re changing the colour of the <h1> from its default black to red. By the way, it’s *color* because CSS was created in the USA, so we’re stuck with their spellings.**

If we wanted to do the same to the <h2> tags on the page, we don’t have to create a whole new selector and that can also reduce the amount of the duplicate code. We can just add to the one above:

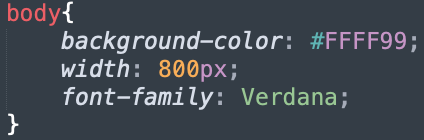


**The comma signifies that the styles contained in here will be applied to all of the <h1> and <h2> tags on the page.**

**Note that in CSS Semicolon(;) is a must**

## Properties

Like was said earlier, there are a lot of properties in CSS. Many we’ll cover as the course continues. Here’s an example of a few, within a selector for the <body> tag:



You’ll notice we’re using [hexadecimal colour codes](https://www.rapidtables.com/web/color/html-color-codes.html) for the background-color. This is a very common practice in web development, as it gives us access to many more colours than just the literal ones we can write out (like when we used *red* above).

## ID & Class

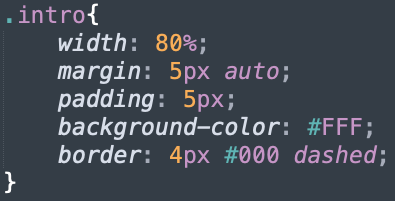
What happens if we just want to add a style to one paragraph in our page, but not to all the paragraphs?

We can use attributes like *id* and *class* in the HTML, and use CSS to create selectors just for those id or class values.

For example, if I want to style this paragraph a little differently from the rest, I would add a class in HTML first:



Now to access this in the CSS, we have to prefix a period on to the start of our selector, like so:

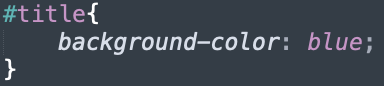


Classes can be used multiple times in HTML code. If we wanted to reuse our *intro* class, we could just use the class attribute and set it to the same value on any other HTML tag.

ID is another attribute that we can use, but ID values can **only be used once per page in HTML.** So they are reserved for HTML components that would only appear once on the page, like a main navigation area for example.



To target this in the CSS, we prefix a pound (#) sign before the value of the ID attribute:



As the course progresses we’ll talk about using these more.

## Pseudo Classes

**A *pseudo class* in CSS is used to define the special state of an element. It can be combined with a CSS selector to add an effect to existing elements based on their states**.

**Or in other words we can say that pseudo class is the class to change the default properties of a particular tag**

For Example, changing the style of an element when the user hovers over it, or when a link is visited.

Remember the default styles for our hyperlinks? We can change those based on their state:

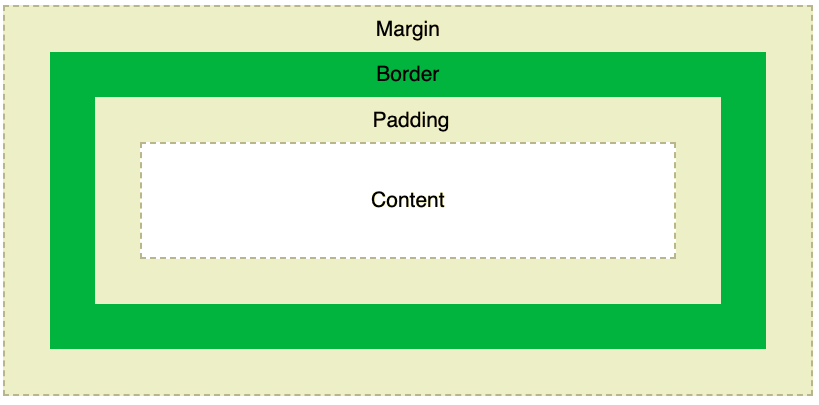


* The **:link** pseudo class is the default state of the link.
* The **:visited** pseudo class is when a link has been clicked
* The **:hover** pseudo class is when a mouse is hover over the link
* And the **:active** pseudo class is the style applied when a link is being clicked

## CSS Box Model

**All HTML elements can be considered as boxes. In CSS, the term "box model" is used when talking about design and layout.**

**The CSS box model is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content.** The image below illustrates the box model:



Explanation of the different parts:

* Content - The content of the box, where text and images appear
* Padding - Clears an area around the content. The padding is transparent
* Border - A border that goes around the padding and content
* Margin - Clears an area outside the border. The margin is transparent

**The box model allows us to add a border around elements, and to define space between elements**.

We can define a border, for instance, in just one step in CSS:

**This creates a 4px width border that is black, and dashed. A more common border style is *solid,* but *dashed* and *dotted* can be used as well.**

Padding and margin can be declared similarly. We’ll start with padding, which is the area outside of the content and before the border.



Padding and margin have 4 sides: top, right, bottom, and left. In the example above we’ve used some CSS shorthand to declare all 4 sides at once. Each of them has been given a 10px value. This is the equivalent of writing out the values like this:



There are also properties for the individual sides: padding-top, padding-right, padding-bottom, and padding-left, where each could be declared on their own line.

Margins act similarly, with 4 sides, and some shorthand options.

Here’s an example of declaring margins, with something a little different that we’ll talk about:



We’ve applied 10px of margin, the area outside the border, to the top and bottom margins.

But what does *auto* do?

**Centering HTML components can sometimes be challenging. We don’t know the size of the browser or resolution the user has on their screen. But with *auto* for left and right margins*,* we’re telling the browser to distribute the leftover space evenly** between the left and right margins. This will make our HTML tag line up in the middle of the browser window, regardless of resolution.

## Shadows and Border Radius

Over the years, CSS has gotten quite good at being able to handle visual effects. A couple of these are, shadows and radius, used to be done using images, which would then increase the digital size and perhaps loading time of the page. But newer CSS techniques have made this easier.

Let’s look at box and text shadows, as their syntax is identical.



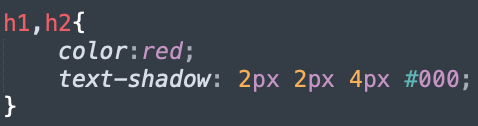
Here we’ve added a grey box shadow to an HTML element (element is a pseudonym often used in place of tag).

These values mean that we are adding a shadow that starts 5px to the right of the box, 5px to the bottom, and the length of the shadow is 5px. The colour will be #888, which is grey. Here’s the result in the browser:



If you wanted to go to the left, you would use a negative number, like -5px.

Text shadows work the same way, but only apply to the text.



Here we’re adding a shadow that starts 2px to the right, 2px down, and 4px long with a colour of black.



**There’s more you can do with shadows, such as stacking them with multiple shadows. Experiment and see what you can come up with!**