

## 2.3: Page Layout

### Tags, Attributes

In order to add content to our page, we will set up our file with some basic structure. First, we will use tags to identify information about where parts of our page start and end. We will do this by using our first set of tags, `<html>` and `</html>`. We refer to these as a set because both use the same predefined word (HTML), and the latter uses the backslash (/) to indicate that it marks the close of that tag set. The act of placing a set of tags around content and other code is often referred to as “wrapping” what is between the tags. This is a good way to create a mental picture of what you are doing.

Next, we will put in a few more tags and then save our file and see some results. In between (typically referred to this as “inside”) your HTML tags, add an empty set of tags labeled head, and another labeled body. Then create two more sets labeled header and footer inside your body tags. Your resulting file should look like this:

1. `<html>`
2. `<head>`
3. `</head>`
4. `<body>`
5. `<header>`
6. `</header>`
7. `<footer>`
8. `</footer>`
9. `</body>`
10. `</html>`

Now we will add some spacing (see [Spacing](#) from our [Good Practices](#) section), and the ubiquitous Hello World in between the close of our header and start of our footer tags:

1. `<html>`
2. `<head>`
3. `</head>`
4. `<body>`
5. `<header>`
6. `</header>`
7. Hello World
8. `<footer>`
9. `</footer>`
10. `</body>`
11. `</html>`

Now we can better see how elements can be placed within one another, called nesting. If we save our file again, and double click the icon for our file in our documents folder to open it, it should open in your computer’s default browser as a white page with Hello World in the upper left corner. Congratulations! You have just made your first web page. Before you ask, though, unless you have created your own server, you will not be able to show your page to someone unless they are at your computer or you send them the file(s) you created. Since only the browser is needed to view the output of HTML, JavaScript, and CSS files, we will be able to view our pages without a server until the PHP examples later on.

As we continue to make changes, you can keep your editor and browser open at the same time. After you save your file in your text editor, you can simply refresh the page in your browser to see your changes.

### Additional Notes

As we look at HTML elements, keep in mind that browsers will do their best to adjust for mistakes like missing tags. Different browsers will react to these problems in their own way, so just because it shows up correct in one browser does not mean it will in others.

Some tags, like those for including images, do not need a closing tag in order to work as no content is necessary inside the tags. In this case, we simply use the tag itself where we want it. In older versions of HTML one of the differences between document types

was how we closed single tag elements. HTML, for example, wanted a break written as `<br>` while an XHTML document wanted `<br/>`. HTML5 will recognize either of these.

## Head

Getting back into our document, we see that the first set of tags is called head. This is where we will put information about the page itself (called metadata) and where we will connect to any other resources like scripts and files that are not part of the page we are using. It is important that your `<head>` tags are always the first set of tags after `<html>`, as this is the order in which browsers expect the information.

To provide some basic information about our page or site, we will add some meta tags inside our head to give it a title, keywords, a description, and other details. These pieces of information help the user and bots understand what the page is about.

Each of these items are all parts of the meta tag (`<meta/>`). Some tags have special features that can be added to their definitions; these are called attributes. Meta tags support attributes like name and content. How we define these attributes will help the browser understand our page better. We will set our title as Our First Page, add the keywords beginning html, learning, and CSIT-107, and for a description we will put in a couple lines about what we are doing, and finally, add our name:

1. `<head>`
2. `<title>Our First Page</title>`
3. `<meta name="keywords" content="beginning html, learning, CSIT-107" />`
4. `<meta name="description" content="A beginning web page for CSIT-107 SUNY Fredonia" />`
5. `<meta name="author" content="Your Name Here" />`
6. `</head>`

There are a few things in this example for us to look at. First, you will note that title has its own tag. What you put in this tag is what will appear as the title in your browser window, and on the tab in your browser for this page (You can only have one set of these tags in a document). Keep in mind, this title will not appear on the page itself. If we want to include it as part of our page content, we would do that from within the body tags.

Next, we see that the values assigned to name and content are placed in quotes. The quotes identify what belongs to that attribute. While it is standard practice to use double-quoted strings, we can also use single quotes. These can be useful when creating HTML statements like those above as output from other languages (we will see examples of this in [PHP](#)).

You might also note that our keywords are comma separated, meaning we break them up by placing commas between different values; we do not need to add separate quotes around each word or phrase.

In the HTML5 specs, the meta tag can also have attributes named charset and http-equiv. Charset allows us to specify a particular set of characters we want to use for the page, an http-equiv supports content-type, default-style, and refresh as options. This lets us tell the browser what type of page we have (in our case, text/html), name a default style sheet (we will get to this later) and specify in terms of seconds how often we want to refresh the page, if at all.

Since our content will not be changing unless we change the page ourselves, and we do not have a style sheet yet, we will just add our content type declaration for now:

1. `<meta http-equiv="content-type" content="text/html" />`

Meta is also where we will define cache items like how long our page can be cached before it expires, or if we even want content to expire, as in these examples:

1. `<META HTTP-EQUIV="EXPIRES" CONTENT="Mon, 22 Jul 2002 11:12:01 GMT">`
2. `<META HTTP-EQUIV="CACHE-CONTROL" CONTENT="NO-CACHE">`

## Body

All of the content that we wish to have on the screen should be encompassed by a set of body tags. The header, content, footer, and div (an all-purpose tag derived from “division”) sections are examples of what we can put in our body tags, which we will see in the next example. Keep in mind that using these tags does not prevent us from seeing content that is not within the body. Tags are used to guide the interpreter in how to display the document. Unless it is specifically within the `<head>` tags, browsers may elect to display any content that is not properly nested in different ways.

## Header, Footer

The header and footer tags are new in HTML5. They were added due to the volume of sites that define a top and bottom section to their pages. Allowing these tags makes it easier to define and find those parts of the layout. The header and footer should be nested within your body tags, but are not a requirement. For our example, we will put a screen title in our header and a copyright in our footer:

1. `<body>`
2. `<header>`
3. `<h1>This is our first page!</h1>`
4. `</header>`
5. Hello World
6. `<footer>`
7. `&copy; 2013 Your Name Here`
8. `</footer>`
9. `</body>`

After saving your file and refreshing your browser, you should see our sentence in the top left, followed by Hello World, and then © 2013 Your Name Here. We made our title text extra-large by wrapping it in `<h1>` tags. H1 stands for heading one, the largest heading by default. We can also use h2 through h6 to access additional styles. Just as in a written document, we use these headings to distinguish different portions of our text. Your browser is applying a default style to make h1 look as it does on your screen. Later, we will see how to override this default style to make our headings look how we want them to. Using these headings allows us to quickly identify different portions of content not only for the reader, but also for search engines, which typically consider content in these tags as indicators of what your site is about, reducing our SEO efforts later.

## Div/Span

While `<p>` helps us split up our text, we also need a mechanism to separate different pieces of content like we did when we used header and footer. This will allow us to define more than just a top, middle and bottom to our page. To do this, we can wrap those sections in `<div>` tags. Div stands for divide—it defines a section of content that should be treated as separate from other content. Span is very similar to div, except that it should identify inline content, meaning material that is within a block of text. Ultimately, a div will place a line break before and after its tags, while a span will not. Aside from this, these tags are functionally equivalent. While these tags seem very plain now, they are very useful when creating complex layouts, and are the tags we will use most often.

While div and span are effective for styling, we should strive to use the best set of tags available so browsers, users, and bots are able to understand our site and its layout, like the `<header>` and `<footer>` that we have already used in our code. Just keep in mind, not all of these are supported in all browsers. If you are wondering which browsers will work with what tags, you can refer to [caniuse.com](http://caniuse.com) to see what is available.

To create organization for our site, we will add some more conceptual sections to our file. You will notice the layout does not actually go left or right as we are labeling our divs (everything will organize top-to-bottom). This is because we need to add CSS for the full effect. We will pick this example up again later to add the CSS needed to create the layout we want:

1. `<body>`
2. `<header>`
3. `<h1>This is our first page!</h1>`
4. `</header>`
5. `<div id="left">`
6. some menu items
7. `</div>`
8. `<div id="content">`
9. Hello World
10. `</div>`
11. `<div id="right">`
12. and some content on the right
13. `</div>`
14. `<footer>`

15. &copy; 2013 Your Name Here
16. </footer>
17. </body>

---

2.3: Page Layout is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

- **2.3: Page Layout** by [Michael Mendez](#) has no license indicated.