Getting Started with HTML & CSS

1.2: Internet & the Web

Learning Goals

* Explain the basic structure of the Internet

 Estimated Read Time: 45 Minutes.

Introduction

Welcome back! In the previous Exercise, you gathered up all the content you’ll need to build your portfolio site, which is what will allow you to dive straight into HTML code in this Exercise, learning how you can use HTML to construct the homepage of your new site. If that seems daunting to you, don’t worry! All will be explained shortly, and we’ll do our best to make things as simple and smooth as possible.

Getting Started

Everything you visit on the web, be it your favorite Nintendo blog, your mom’s Instagram account, or your online law textbook, is nothing but a collection of nicely sorted (hopefully) files, the same as anything you might have on your computer or laptop. These files are sorted in folders, which are themselves sorted into much larger directories, similar to the way books are sorted in a library. A page in a book contains text, similar to an HTML file, which also nothing more than text. A website, then, is like the entire book, which contains multiple pages (or, in this case, multiple HTML files).

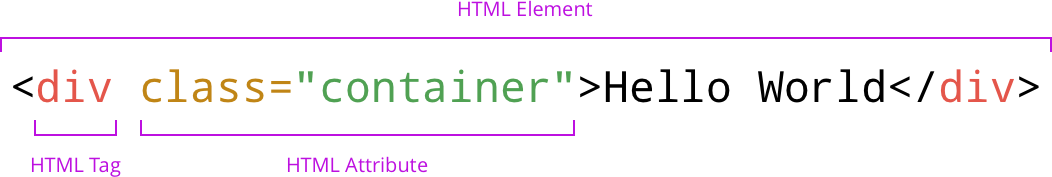
There’s a whole lot out there to learn about HTML and origins, and if you’re interested, you can learn more by checking out this [introduction to HTML on the W3Schools website](https://www.w3docs.com/learn-html/html-introduction.html).

But the best way to get acclimated with HTML is to simply start learning the elements that make it tick. Ready to take a look under the hood? Then, let’s get started!

NOTE!  
You will be using two types of interactive editors that are either embedded into our platform or linked out, as you go through the course. They are: repl.it and CodePen; they will help reinforce concepts you learn and get familiar with different code editors. If you interested to know more about them and how they work, you can visit official websites of [repl.it](https://repl.it/) and [CodePen](https://codepen.io/).

HTML Elements, Tags, and Attributes

Let’s start by defining a few foundational terms. Having a solid understanding of these basic HTML features and how they function within the browser will greatly benefit you in upcoming Exercises. In the image below, you can see an example HTML element.



HTML Elements

HTML web pages are full of **HTML elements**. An HTML element is an individual component of an HTML document or web page. On the page you’re looking at now, for example, the logo and menu items are all HTML elements, as well as the rest of the content you see. These elements are defined by **HTML tags**. In order for an HTML element to be displayed on a web page, you must present it within a tag.

HTML Tags

**HTML tags** always come between brackets: < and >. Think of HTML tags as the building blocks of HTML, defining how HTML elements should appear on a web page. HTML tags come in two forms—opening tags <tag> and closing tags </tag>—and use the following structure: <tag> content </tag>. Nearly every HTML element has **an opening tag and a closing tag** surrounding its content; however, you’ll find there are a few exceptions! These are marked with an \* in the list below.

The most important tags you’ll want to check out for your project are:

|  |  |  |
| --- | --- | --- |
| <!--...--> \* | <!DOCTYPE> \* | <html> |
| <head> | <meta> \* | <title> |
| <link> \* | <script> | <body> |
| <header> (HTML5) | <nav> (HTML5) | <main> (HTML5) |
| <section> (HTML5) | <div> | <h1> to <h6> |
| <a> | <br> \* | <button> |
| <picture> (HTML5) | <figure> (HTML5) | <figcaption> (HTML5) |
| <hr> \* | <img> \* | <p> |
| <span> | <strong> | <svg> |
| <footer> (HTML5) |  |  |

TIP!  
Now’s a good time to visit W3Schools for a look at the [full list of HTML tags](https://www.w3schools.com/tags/). You’ll find a description for each tag and, when selecting it, a new page will open with its definition and usage, including examples you can try out yourself. We recommend bookmarking this page as it will come in handy while you’re still learning to use all the tags.

HTML Attributes

**HTML attributes** are placed *within* an HTML element’s opening tag and are used to define the characteristics of the element. All HTML elements can have attributes, and they usually come in name/value pairs such as name="value". To check out all the different HTML attributes available, visit the [W3Schools attributes page](https://www.w3schools.com/tags/ref_attributes.asp).

Reviewing an HTML Document

Let’s start by inspecting some code. Below, you’ll find an HTML template, which includes the minimum code necessary to start building a web page.

<!DOCTYPE html>

<html lang="en" dir="ltr">

<head>

<meta charset="utf-8">

<title></title>

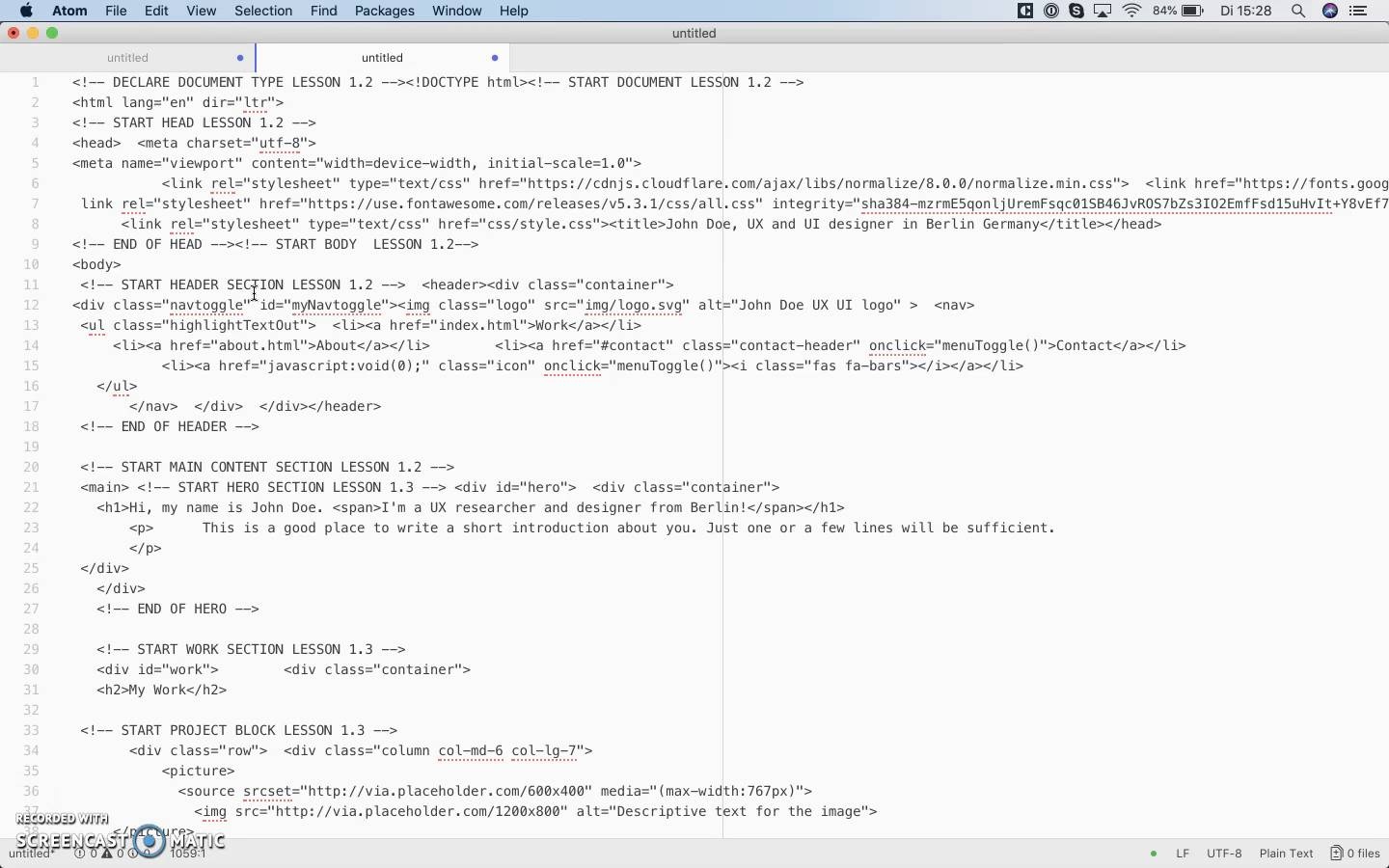
</head>

<body>

</body>

</html>

TRY IT!  
We recommend following along with all the code given in your own editor. This simple technique in active learning will help you better retain newly acquired information. We use [Atom](https://atom.io/) as our go-to editor in this course and advise you to do the same. (This will make it easy to follow along!) If you’re already comfortable with a different editor, however, feel free to use that. In the video below, we’ll show how to use the editor to create the default template given above. You can also try editing a code example with an online editor here: <https://www.w3docs.com/tools/editor/26>.



7:41

At the very top of the code, you should see a line: <!DOCTYPE html>. This is not HTML code per se, but a declaration used to inform the web browser that the document being rendered is an HTML document—this is also known as a browser directive. Every HTML document must begin with a DOCTYPE declaration in order to be compliant with HTML standards. Because a declaration isn’t really an HTML element, it doesn’t need an end tag.

Next, you’ll see: <html lang="en" dir="ltr">

The <html> tag used here has three purposes:

* It tells the browser that this is an HTML document.
* It represents the root of an HTML document.
* It acts as the container for all other HTML elements.

If you scroll all the way to the bottom of the code, you’ll see its end tag (</html>). Everything between these two tags is part of the HTML element.

Tacked onto your <html> tag, you’ll see: lang="en" dir="ltr". This lang is very useful as it declares the language of the document. Its value is currently set to en for English. The next attribute, dir, specifies the text direction of the document’s content, which in our case is ltr (or “left to right”).

Here is our first repl. Take some time to play around with the HTML tags like <html>, <head>, <title><body>, <p>, and <h1> (Don’t worry if you’re familiar with only a few tags out of them, we’ll discuss them shortly). You will see that the repl has a few lines of code already. Read those lines carefully to get a sense of their purpose. Press the green run button to find the output in the bottom half of the repl. Next, modify the existing code, run the repl, and check the output. Is the output different this time? Compare the new and the previous outputs and try to understand what caused the difference. We suggest you practice whatever you will learn in this Exercise a lot so that you are confident enough to move to the next Exercise.

HTML Document Head

After the html tag, we see a <head> tag. The <head> element is a container for metadata, which is data concerning the HTML document, and is placed between the <html> tag and the <body> tag. Metadata typically defines the document’s title, character set, styles, links, scripts, and other meta information. Let’s take a look at what’s in our example:

<meta charset="utf-8">

This charset tag specifies the character encoding for our HTML. This is important for the browser to understand the characters in the document. There are many other meta tags out there. Check out this page for a [complete list of all meta tags](https://gist.github.com/lancejpollard/1978404). The ones you’ll likely be working with the most are name and content. For example:

* <meta name="author" content="YourName">
* <meta name="description" content="Short description including keywords related to the pages content for SEO purposes">

Next, you’ll find the <title> tag, which defines the title of the document and is required in all HTML documents. The title appears in several places: the browser tab, the favorites list, and search engine results. Give it a meaningful name, for example: MyName UX/UI Designer CityName

TIP!  
Choosing an appropriate title for each page is very important as it’s one of the first things that search engines take into account when indexing your website. Keep in mind that it shouldn’t be too long and that it should describe the content of the page as succinctly and accurately as possible. You can find more information on titles in [Google’s Webmaster Central Blog](https://webmasters.googleblog.com/2006/10/target-visitors-or-search-engines.html).

Later in the course, you’ll also add the following tags in the head: <link> and <script>. Check the link we gave you with the tag list to find out what these are used for.

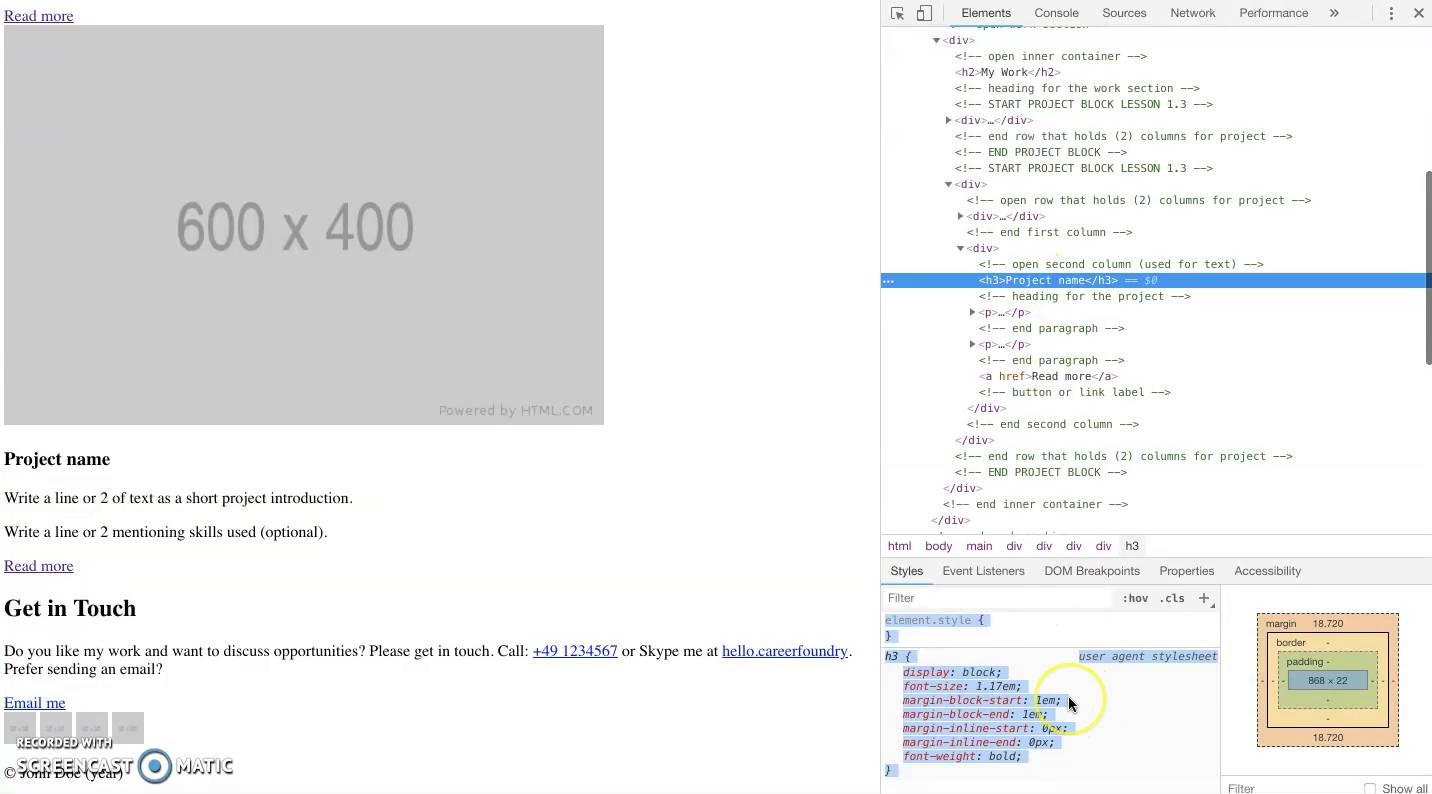
Before we move onto the visual part of the page, let’s check out a very helpful tool first, the **Web Inspector**.

Web Inspection

From a designer’s point of view, wireframes aren’t exceedingly complicated. What’s more complicated, however, is the code work that goes into translating the outfacing design into something clickable—in other words, the stage between our wireframes/mockups and the completed web page with which users can interact. Web pages are written in Hypertext Markup Language (HTML), which is a descriptive language for structuring web pages, but where is that code?

This is where the **Web Inspector** comes in! The Web Inspector is a handy tool that comes with every browser and gives you a sneak peek into the code behind the pages. Simply viewing the code via this tool would give you key insight into the differences between your world and that of a developer’s—something that could greatly improve project communication. You might even be able to identify framework and plugin code and do a bit of your own troubleshooting!

With all that in mind, let’s go ahead and watch the video below to learn how to use the Web Inspector. We’ll be using Chrome as our default browser throughout this course (you may need to alter the directions slightly if you’re using a different browser). As for the pages we’ll be viewing, these are simple web pages we’ve created based on the wireframes we’ve prepared for your portfolio site.



13:44

HTML Document Body

As this Exercise is quite long, feel free to take a break here if needed. We’ve already covered a lot of information! Otherwise, let’s move on to the HTML code for the body of our document.

The <body> tag starts right below the ending </head> tag. As its name implies, the <body> tag refers to the body of the document. It contains all the content within an HTML document—text, hyperlinks, images, tables, lists, and more.

As a designer, you may be wondering when we’ll start examining the more visual elements of an HTML web page. Well, now is that time! To do so, we’re going to use the same wireframes we explored up in the Web Inspector video (the wireframes you’ll be using to code your new portfolio page). Let’s take a look at the wireframe for our homepage and see if we can break it up in meaningful sections.

Go ahead open up the [homepage wireframe](https://images.careerfoundry.com/public/courses/frontend/New%20Update/Project%20Brief%20Wireframes/Desktop/home_work%20page%20desktop.png). You can see that we’ve already split it up into four different sections. Using the tags list at the beginning of this Exercise, can you identify the four sections? Great! But just in case you need to double-check:

* 1 = <header> section
* 2 and 3 = <main> section
* 4 = <footer> section

Note that all these elements (which form the visual part of your page) are inside the <body> tag. As its name implies, the <body> tag refers to the body of the document. It contains all the content within an HTML document—text, hyperlinks, images, tables, lists, and more. This is why you’ll see the entire web page light up if you hover over the <body> tag using the Web Inspector.

Header

When we build a site, we follow a general left-right-up-down order, much like when we read a book. Thus, the first element we would want to take care of in the <body> is the <header>. The <header> element is most often located at the top of a page, and it usually contains the logo and navigation element. It can also be used inside other elements such as <section> or <article>, where it contains one or more heading elements. There are a few exceptions, however, you can read about in the [W3Schools article on tags](https://www.w3schools.com/tags/).

For our portfolio site, the header would be structured in this way (note: your header may look different, this is just a representation of the HTML tags):



You can see, here, that the <header> (the orange area) contains different tags. It starts with a <div> tag (yellow area). The <div> tag is a generic container used to group elements together.

Run the repl below to check out the complete structure of your portfolio site:

TIP!  
It’s a good idea to always use the same format when creating “sections” like the header, hero, work, and footer. Start with the main container for the whole block, which would be <header> for the header, <div> for the hero, <div>for work or projects, and <footer> for the footer section. (In all example graphics in this course, this “main container” will be represented in orange). Also, make a habit out of closing your tags immediately after creating them (so you don’t forget to later). Following the main container, you then create a second <div> inside of it. (In all example graphics in this course, this second container will be represented in yellow. You can see that it’s INSIDE the orange container in the image.). This will be the div that actually holds the content in each section. The final result should look like this:

For the header: <header><div></div></header>

For the hero section: <div><div></div></div>

For your work/projects: <div><div></div></div>

For the footer: <footer><div></div></footer>

Setting up each of the sections of your page in this fashion will enable you to more easily change the width and position of the content inside later without changing the width and position of any other content on the page. It will also allow you to use a background color or image for each section if you so choose. This will all become clear when we discuss CSS in Exercise 1.7.

You see that, on the left side of this <div>, there’s an <img> (image for our logo), and on the right side, there’s a <nav> (navigation) tag that contains three <a> tags of its own. These tags allow us to style the different sections of the page. The <div> tag itself allows us to resize and position the entire container (including all the tags inside of it) on the page, giving it a certain width or centering it, for instance. We’ll be looking at this in more detail in the CSS lessons.

The <nav> tag is used for your page’s navigation. It’s used only for major blocks of navigation links such as your primary and secondary navigation. Using **HTML5** semantic tags like <header> and <nav> gives your code more meaning than simply using generic <div> tags. Why does this make a difference? Well, first of all, it makes your code easier to read and maintain. Second, it improves your site’s accessibility. Browsers with screen readers often use this element to determine whether to exclude the initial rendering of the content. You can find a full list of all new HTML5 elements on the [W3Schools HTML5 page](https://www.w3schools.com/html/html5_new_elements.asp).

Now, let’s discuss the other elements found in the header: **images** and **hyperlinks**.

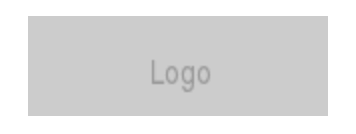
Adding an Image

Text can convey a great deal of information, but there’s nothing like a well-placed image to add that extra punch. The <img> tag places an image on a web page. It has **two required attributes**: src, which specifies the URL of the image file, and alt, which shows the alternate text for an image in the event the image can’t be displayed. The alt (alternative) text is also used by screen readers, hence the importance of describing the image well.

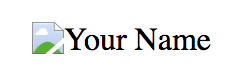
Let’s use our logo as an example since it’s the next element in our header. To add a logo in the header, use the following code:

<img src="https://via.placeholder.com/150x50?text=Logo" alt="Your Name" >

In the browser, the image would look like this:



Now, if the link were incorrect, you’d see the broken link and the alt text:



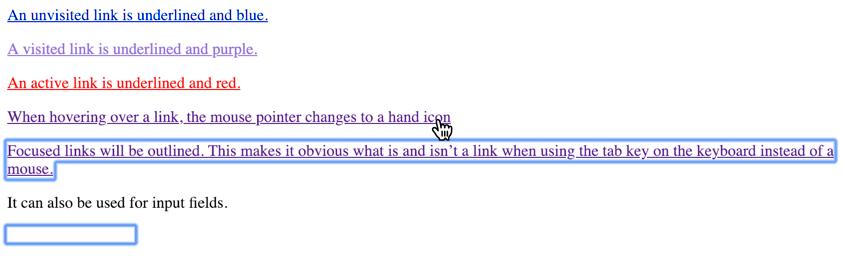
You may have noticed that the <img> tag has no end tag. As the image file is specified as an attribute in the start tag (meaning that no content falls outside of the tag itself), the <img> tag can be closed without an </img> at the end. It’s completely self-contained!

NOTE!  
You may have noticed that all the images in the provided files are gray blocks. We call these placeholder images, and they’re used to “hold” the “place” of an image before we have the final image asset ready to place in the page. There are a number of resources out there that can provide these images, or you could code them yourself. In our examples, we used [Placeholder.com](https://placeholder.com/) to create the placeholder images. In the upcoming Exercises, we’re going to replace them with real images.

Hyperlinks

Next in our header, we see the menu (navigation) bar and the menu items. There are different tags we can use for the menu bar. As we learned, the <nav> is the semantic tag used for the navigation. Inside the <nav>, we find each menu item as an individual link—a hyperlink.

A hyperlink is defined by the <a> tag, which is used to link from one page to another, or to another part of the same page (which we’ll discuss later in the course). The href attribute in the <a> element is the most important attribute as it points to the link’s destination. By default, links will appear as follows in all browsers:



We can leave the href attribute of the links blank for now as we haven’t created their corresponding pages. In the end, our menu bar might look something like this:

<nav>

<a href="#">Work</a>

<a href="#">About</a>

<a href="#">Contact</a>

</nav>

TIP!  
When specifying an href for an external link (link to a page or file outside your website), always remember to use the full URL, including the protocol (e.g., https://). For example, the link <a href="www.google.com">Google</a> wouldn't work! Instead, you’d need to use <a href="https://www.google.com">Google</a>.

Want to practice whatever you've just learned? You can use the following repl to understand hyperlinks better:

Summary

In this Exercise, we familiarized ourselves with a handy little tool—the Web Inspector—that allows us to examine the HTML code behind every web page. We also began talking about HTML tags. We learned that the <head> tag contains metadata for an HTML document and that the <body> tag contains all the content in an HTML document. We also started working with Atom, a code editor we’ll be using throughout this course. Finally, we looked at a few common HTML tags—<div>, <a>, and <img>—as well as a few HTML5 tags—<header>, <nav>, <main>, and <footer>. Once finished, we began using those tags to build our portfolio site.

That was certainly a lot of information to take in, so for now, let’s take what you’ve learned and put it into practice by creating the first part of the mockup for your portfolio site! In the next Exercise, we’ll begin looking at more advanced and complex HTML tags and create the content for the <main> and <footer> sections of your homepage.

Resources

If you’re curious to read more about the topics covered in this Exercise, then we recommend taking a look at the resources below. Note that this reading is optional and not required to complete the course.

* [How Does the Internet Work?](https://web.stanford.edu/class/msande91si/www-spr04/readings/week1/InternetWhitepaper.htm)
* [Chrome DevTools Overview](https://developer.chrome.com/devtools)
* [HTML Element Reference](https://www.w3schools.com/Tags/)
* [Meta Tags and SEO](http://www.wordstream.com/meta-tags)
* [The <a> Tag](https://www.w3schools.com/tags/tag_a.asp)

Take the quiz to test your knowledge on this Exercise.

Take Quiz

Task

* [DIRECTIONS](https://careerfoundry.com/en/course/frontend-development-for-designers/exercise/internet-the-web#directions)
* [SUBMISSION HISTORY](https://careerfoundry.com/en/course/frontend-development-for-designers/exercise/internet-the-web#step_submission_history)

 Estimated Task Time: 1 - 3 hours.

NOTE!  
As this course has been recently updated, many of the Tasks or instructions within the Tasks have changed. For this reason, the example student submissions shown at the bottom will **no longer match the current Task instructions**. Please keep this in mind and don't use them as a guideline when working on your own submission.

Now that you’ve learned about the powerful Web Inspector and Atom, a text-editor for developers, it’s time to get some practice in! Are you ready to create the very first iteration of your new portfolio site? This will be the site you share with future clients and employers alike, so be sure you’re putting in the effort now to make it shine when you’re done!

**Directions**

1. Create a folder on your desktop called "Portfolio Site (YourName)".
   * **Note:** This is simply to make your project easier to identify once you’ve submitted the folder for review.
2. Open this folder in Atom and create a new file. If you need a refresher, watch the video at the beginning of this Exercise again.
3. Using what you’ve learned, create the default html template in this file. It should look something like [this](https://s3.amazonaws.com/coach-courses-us/public/courses/frontend/1.1/default_html.png).
4. In the <head> tag, think of a meaningful title for your <title> tag and add it to your code.
5. Write down the HTML code for the 4 sections created in the <body> based on the mockup. Feel free to refer to the sample code found in this Exercise (or do a bit of your own research if you created a different layout than was introduced in [Exercise 1.1](https://careerfoundry.com/en/steps/getting-started-with-frontend)).
6. In the header section, insert the <div> , the logo <img> , and the <nav>, as well as 3 menu links (<a>) for “Work” (which will be both your homepage as well as your projects page), “About,” and “Contact”. For the logo, you can simply use a placeholder image for now.
7. Once you’re done writing the code for all these elements, save the file as “home.html”.
8. Go to the folder in your file explorer and double-click on the file. It should open in your default browser as a web page. If you followed the mockup and code above, it should look something like [this](https://s3.amazonaws.com/coach-courses-us/public/courses/frontend/1.1/example_1.2_task_8.png).
9. While it doesn’t look quite like the mockup yet, so long as all the elements are there, it will work for now. We’ll complete this page in the next Exercise and learn how to style the elements to make them look like the mockup later on in the course.
10. Once you’re happy with your code, save it, zip the folder you created and submit it here for review. Feel free to share additional thoughts or ask questions on your submission page.

**Bonus Task:**

As a designer, you might feel there’s no reason to have your own logo, which is fine; however, it’s still a good idea to have your name visible in the header on each page. This way, regardless of which page is viewed first, it will always be clear to a visitor whose site they’re navigating. A logo with your name would be a perfect solution for this.

There are two simple ways you can go about making a logo:

**Method 1:**

Add your name as a hyperlink. You’ll be able to go in with CSS later and style this using a nice web font. Why a hyperlink? You can have the logo link directly back to your homepage (which is a common navigation pattern in modern websites).

* **Task:** Create the html code for your name as a hyperlink in the html page you created above.

**Method 2:**

Create a picture logo of your name. You can do this using a variety of different tools. If you create a pixel-based image, save it as a (transparent) PNG (24). A better option, however, would be to create a logo using a tool like Sketch, Illustrator, or Affinity Designer. Make sure the text is set to outlined before saving (so that each letter is an image), then save it as an svg.

* **Task:** Create a picture logo and add it to the zip file in your submission.