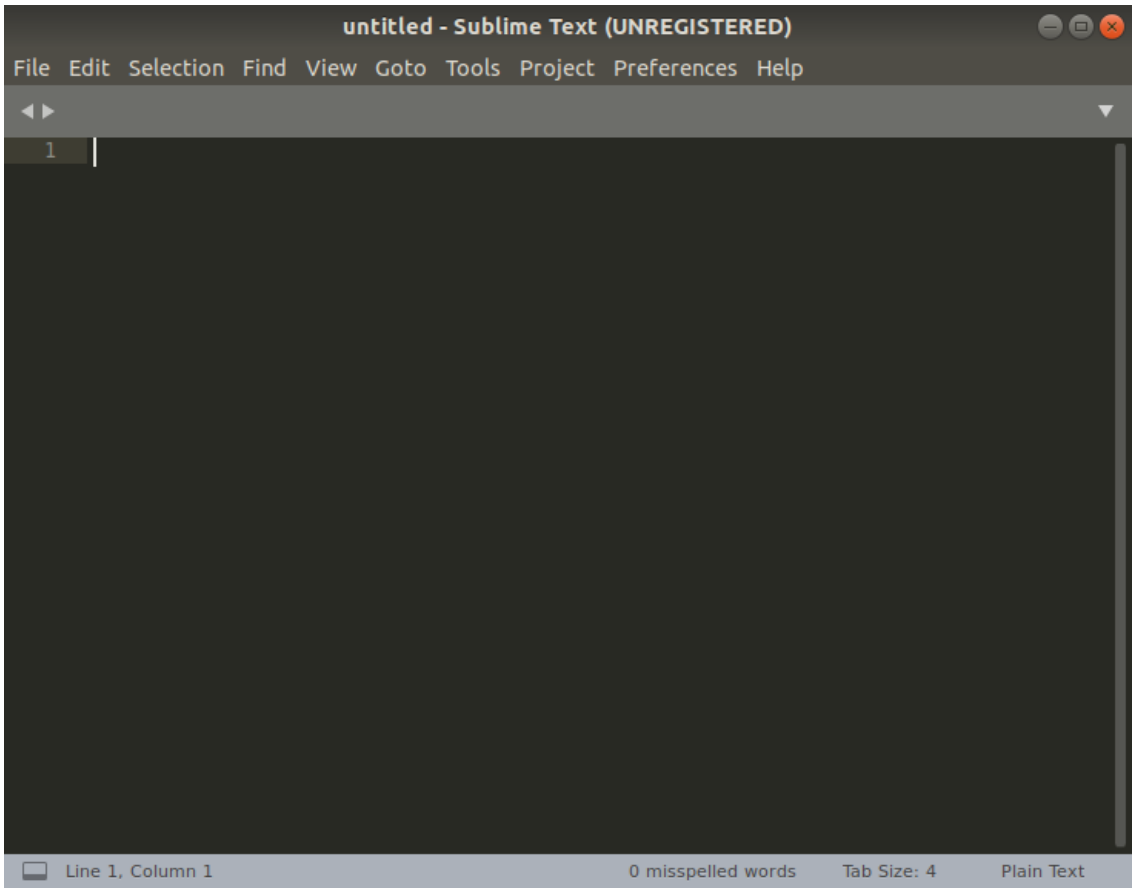
HTML Basics

We'll now use the tools that we downloaded and installed in the previous section to do our first development work, creating our first text on a webpage. We'll use HTML for this purpose. Let's talk about HTML quickly before we start creating our first page.

HTML stands for **H**yper**T**ext **M**arkup **L**anguage. In a nutshell:

- **Hypertext** simply means text that can jump from one point to the other. If you have ever clicked a link on a page you've used hypertext.
- **Markup** is simply a way to structure content so that we can distinguish between different blocks of text.
- **Language** Computer languages are similar to real-world languages like English and German, just very strict in their syntax.

HTML lets us structure our page and the data in it. Once we have a structure and data to work with, we can focus on style and functionality.



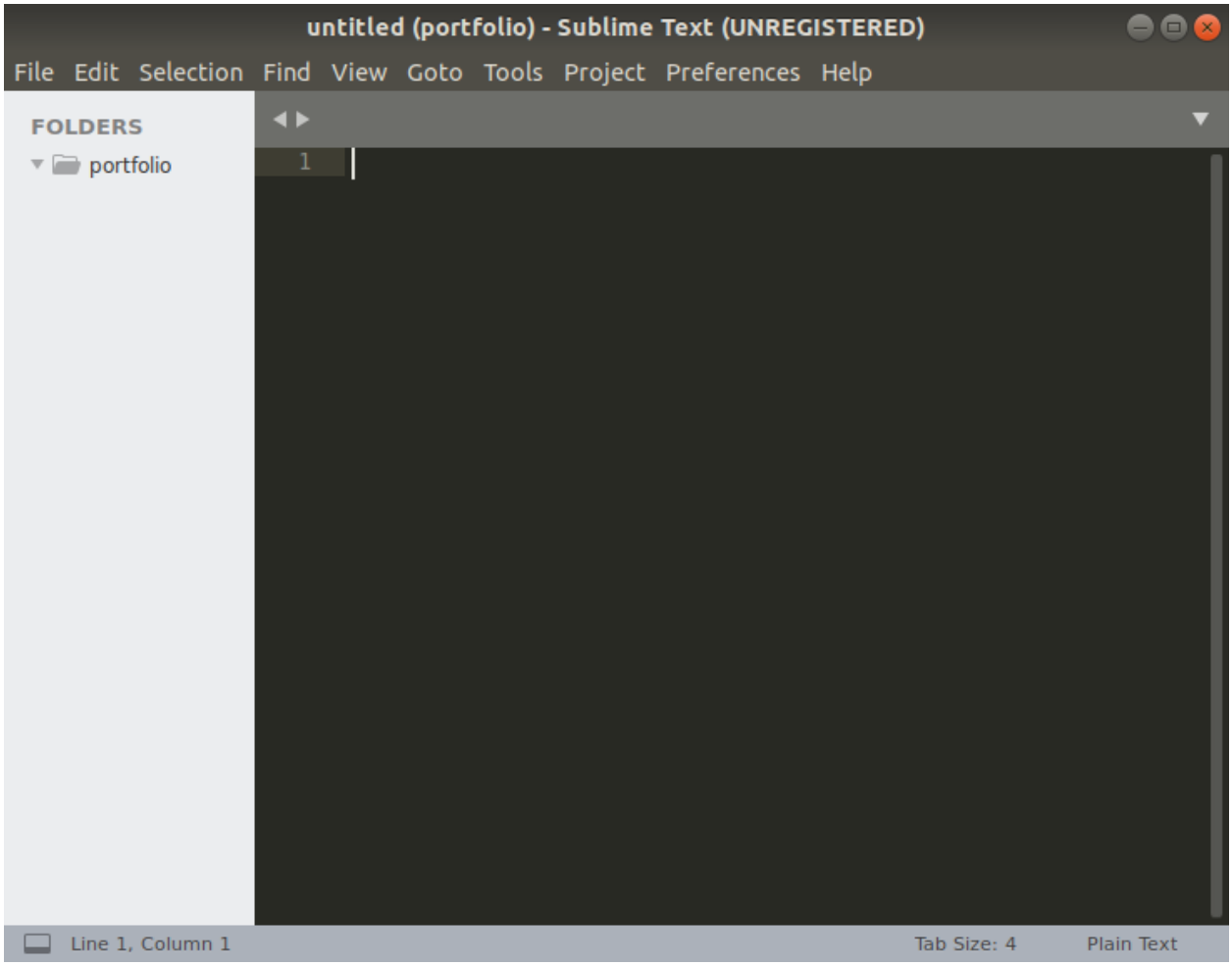
Click **File** > **Open Folder**

Once you select **File > Open / Open Folder,** it will open a new window that will allow you to select an existing folder or create a new folder. Navigate to your 'Desktop' and create the folder there. Creating the folder on your Desktop makes it easy to find later on.

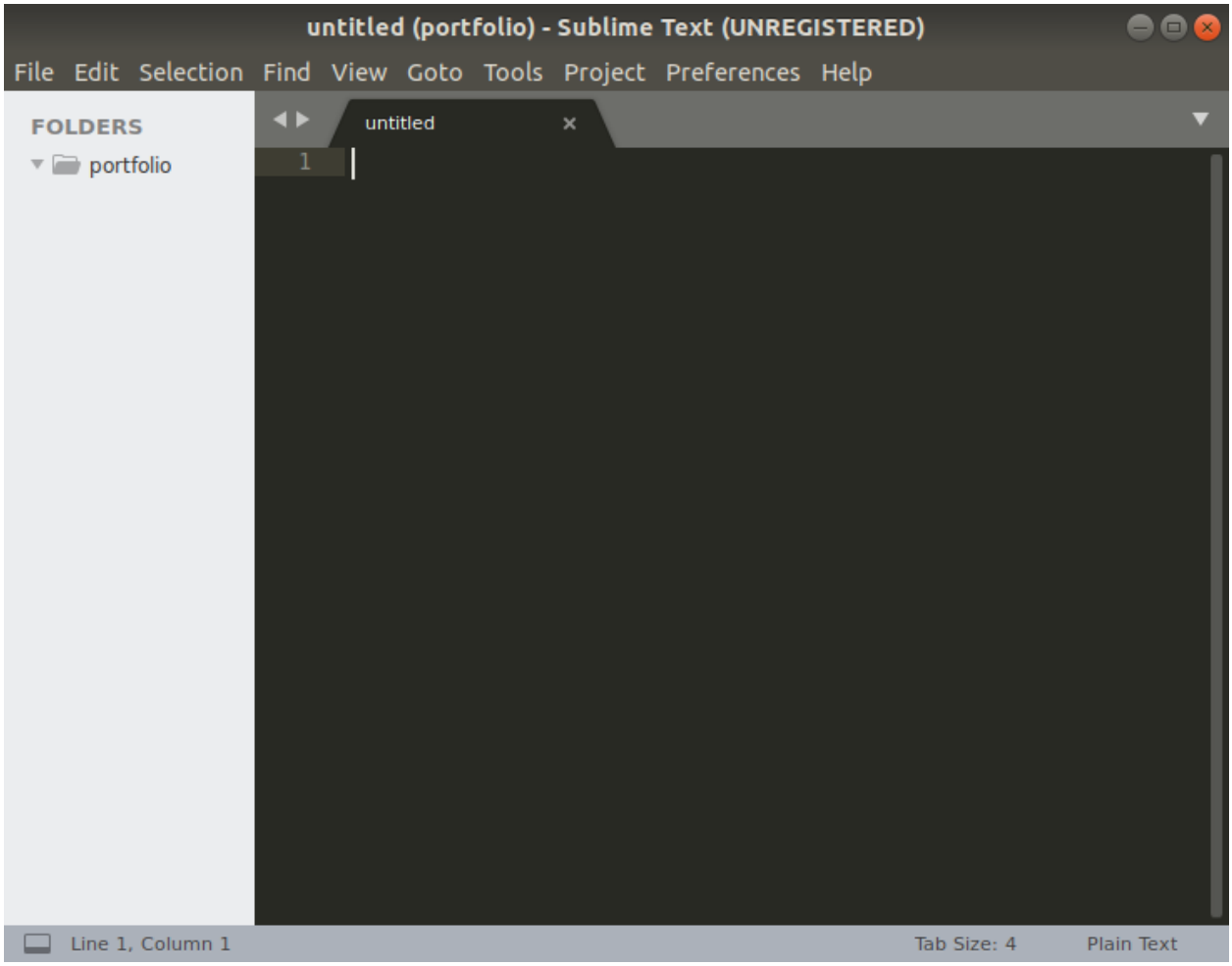
Click on the little icon that allows you to create a new folder. It can look different on different operating systems.

We'll name this folder **portfolio** because that's what our project is, but feel free to choose a different name.

If all goes well, you should see a screen like the following.



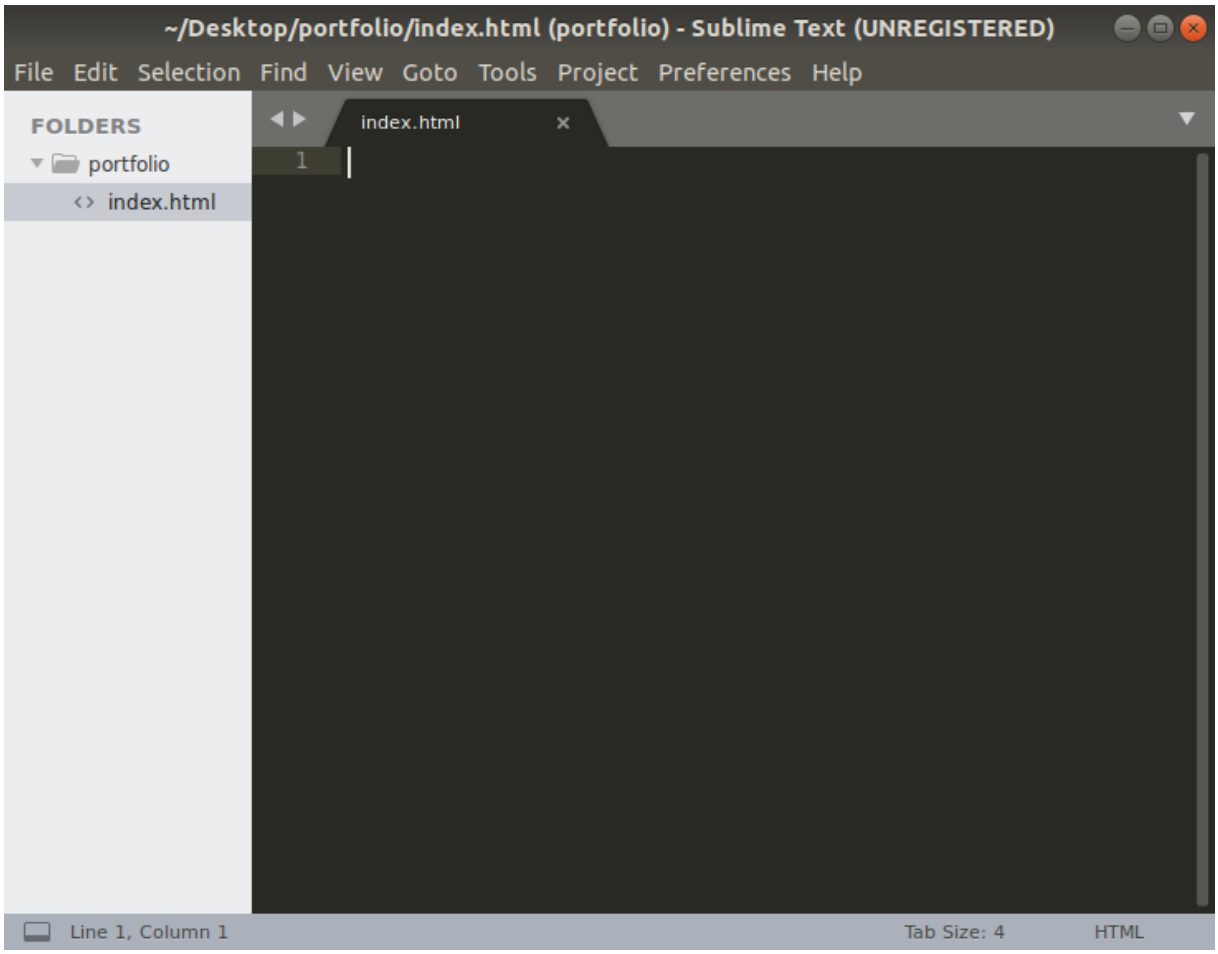
From here, right click on **portfolio** in the left sidebar, and click **New File**. You should see a new file labeled "**untitled"**. That's okay, since we've not named it yet.



Let's save the file. Click **File** > **Save** and you should get a window asking you to enter the file name. Here, enter **index.html** and click save.

We name it specifically 'index.html' as per convention. **By default, the entry point into a website is called the index page, hence the name.**

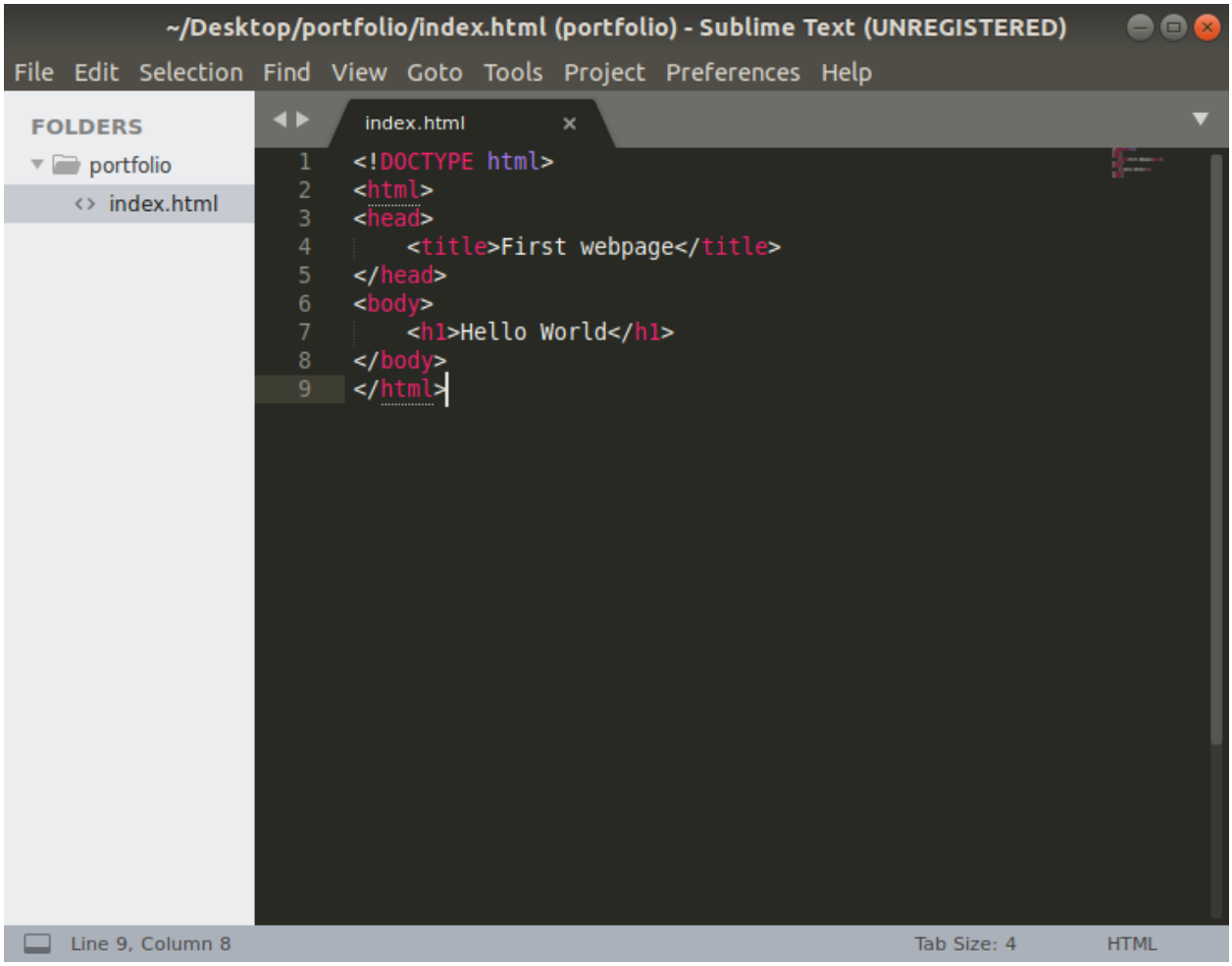
Also, make sure you save the file inside the **portfolio** folder. Here's what you should see now.



Now it's time to write our first few lines of code! For now, we'll just get our page up and running. Later, we will dig into the meaning of each line, talk more about what HTML is, and put more structure into place.

So let's get started coding your first webpage. Type the following code **as is** into your **index.html** file:

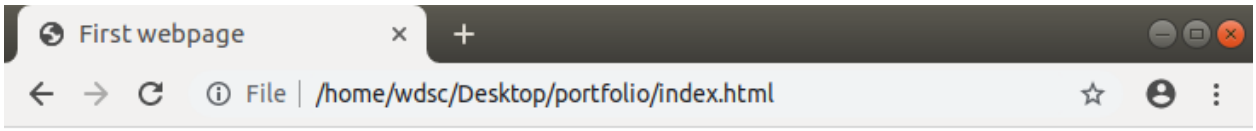
```
<!DOCTYPE html>

<html>

   <head>



   </head>



   <body>

       <h1>Hello World</h1>

   </body>

</html>
```



Good work! You just wrote some real HTML code. It's time to see how it looks in a browser. This is called **running the code** and we'll be doing it all the time in this training.

To run your HTML, open the Google Chrome web browser that we installed previously. Once there, press **Ctrl+O** on Windows and Linux or **Cmd+O** on Mac to open the file explorer dialog box.

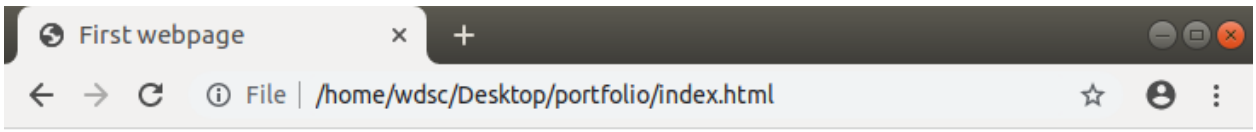
Navigate to the **Desktop** and then **portfolio** folder and you should see the **index.html** file that we just wrote some code in. Open that file and you should see something like the following:



Hello World

If you see the above page, give yourself a pat on the back. You just created your first functional webpage. Notice where the text "*First Webpage*" and "*Hello World*" appear.

Let's go back to Sublime Text and try to change the **Hello World** on line 7 to **Hello World, check out my website!** Save it and refresh the page in Chrome. You should see an updated version of the text.

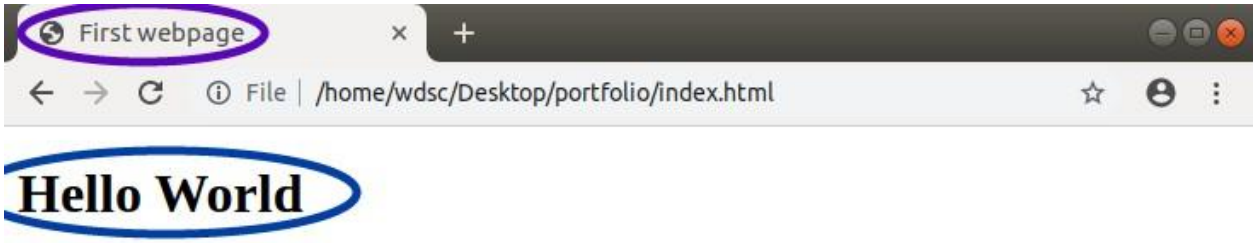


Hello World, check out my website!

Congratulations! You just created and edited your first webpage and viewed it on your browser. With this, you're well on your way to building your first website.

Task 1

Did you notice how the text on line 4 (<title>First webpage</title>) and line 7 (<h1>Hello World</h1>) appear in the tab name and in bold on the page respectively?

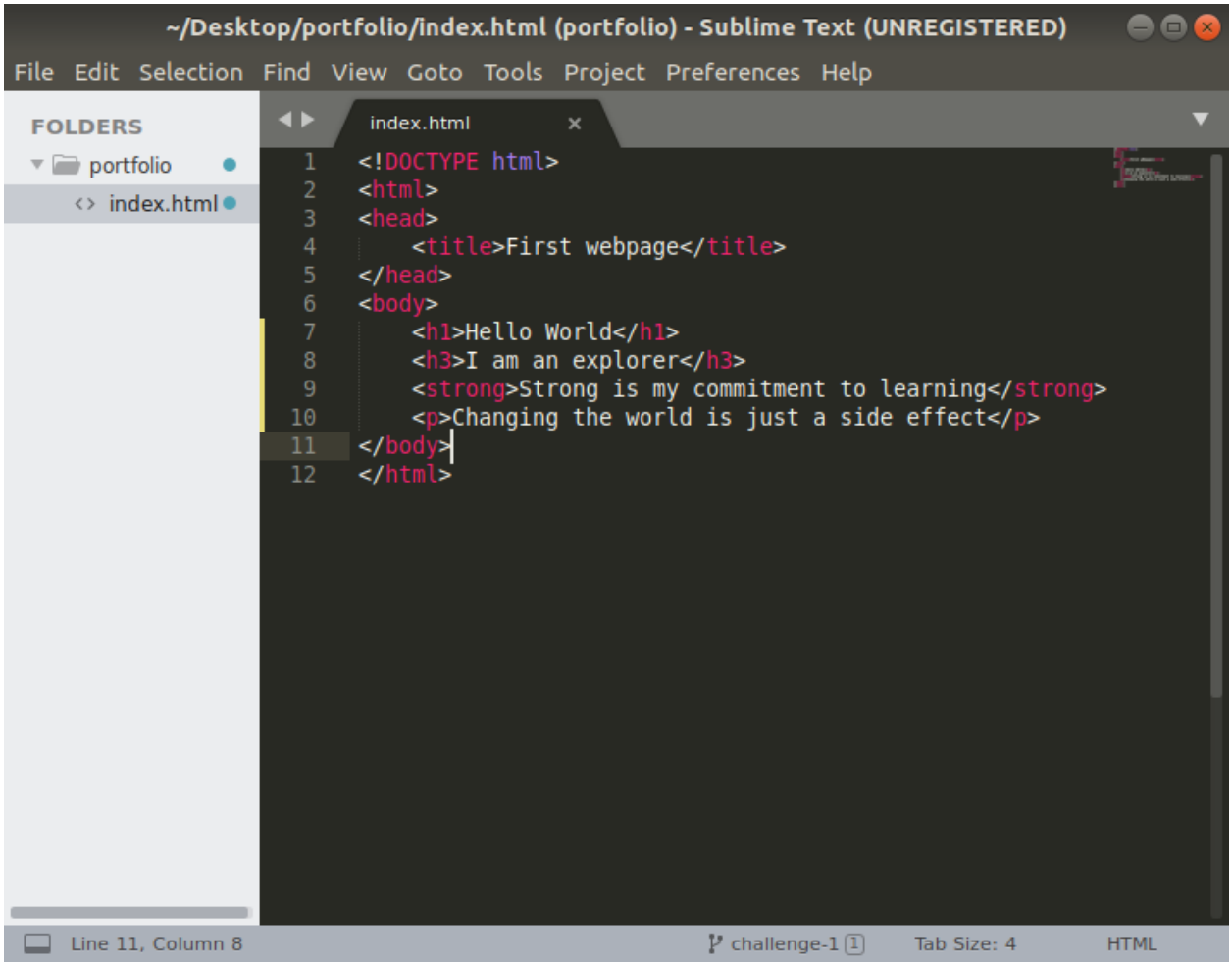


`title` and `h1` (heading 1) are what we call HTML elements. There are many more such elements, like `p` (paragraph), `h2`, `h3`..`h6` (heading 2, heading 3…heading 6) and `strong` (bold text).

For task 1, try adding the following tags to your page with your own content

1. h3
2. strong
3. p

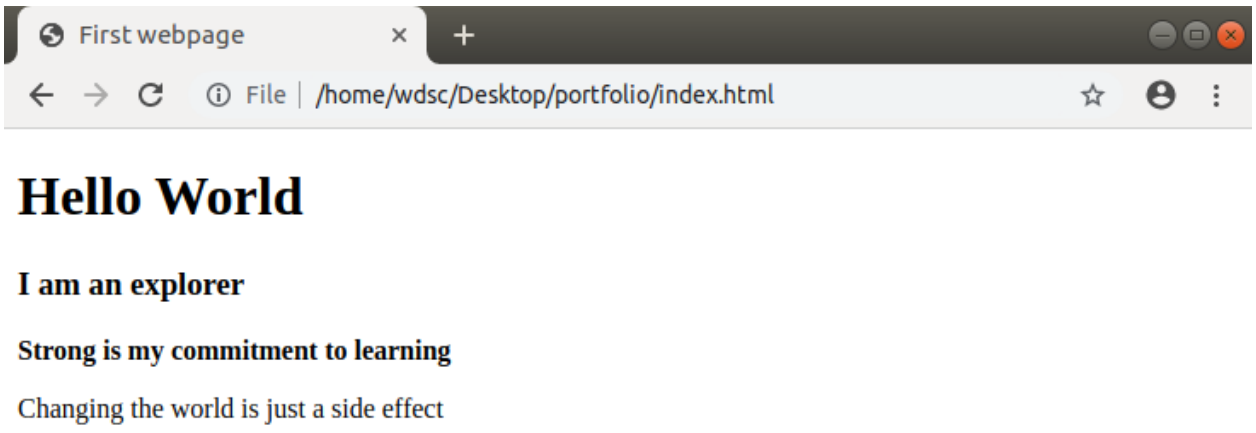
When you're done, your page should look something like this. The text inside the tags (*"I am an explorer"*, for example) can be customized to be whatever you want.



Your task is to write it as shown in the picture (maybe with your customized text) and explain to yourself what `h3`, `strong` and `p` elements did.

Did you figure out what each tag does?

`h3` is a heading, just like `h1`, but less important (and looks smaller). `strong` makes the text appear bold. `p` is short for paragraph, and is used for textual content.



HTML

We've already seen that HTML is a type of language that structures content; in other words, it labels different elements such as images and text in order to tell your browser how to display the content and in what order.

Earlier, we wrote some HTML and worked with a few HTML elements, too—but we haven't really understood them.

We'll look into what HTML is made up of—in other words, HTML elements—and then use them to add detail to our portfolio site.

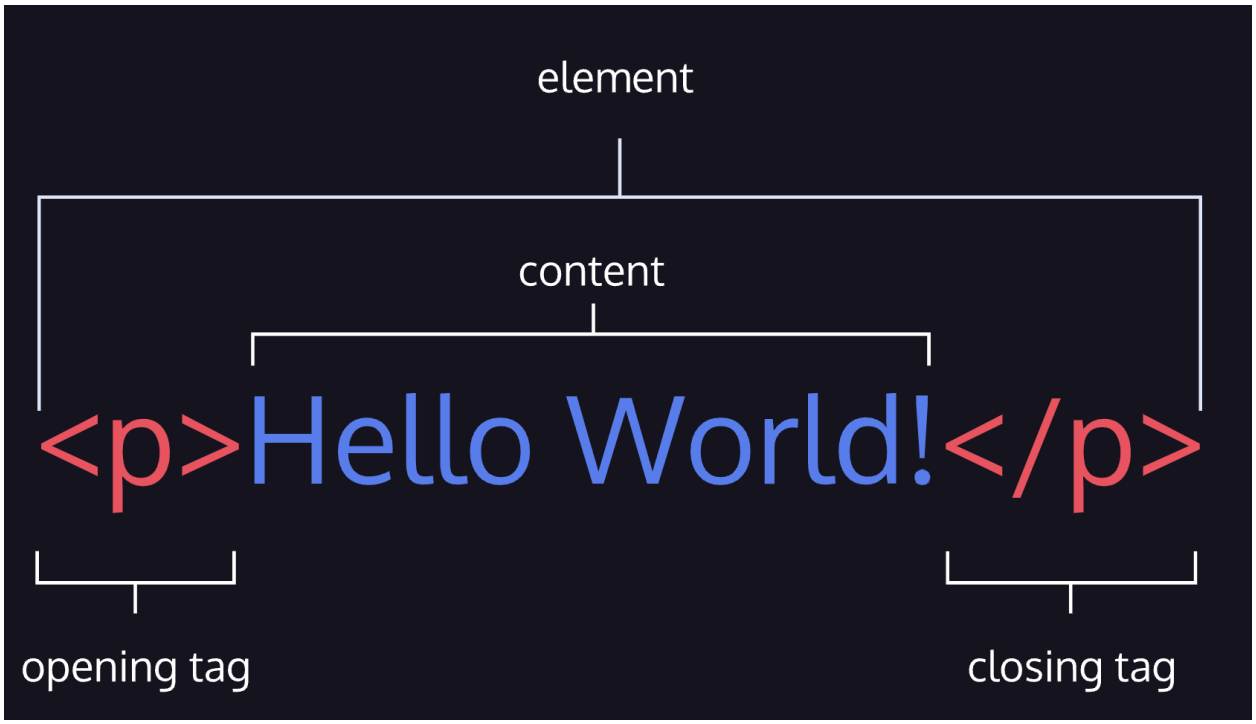
Element tags

We've already seen a few HTML elements. You can think of an HTML element as an individual piece of a webpage, like a block of text, an image, a header, and so on. In the previous lessons you used a heading element, h1, which looked like this:

Every HTML element has HTML tags, and a tag (<h1> or </h1>) is surrounded by angled brackets like < and >.

Tags define elements and tell the browser how to display them (for example, they can tell the browser whether an element is an image or a paragraph of text).

Most HTML elements have an opening tag and a closing tag that show where the element begins and ends. The closing tag is just the opening tag with a forward slash (/) before the element name. We can generalize the format of an HTML element as follows:



Here, content is something we add. It can be anything, like “Hello world” in our h1 example; ‘element name’, however, has to be one of the predefined tags like h1, h3, p or strong.

Element attributes

HTML elements can have certain attributes that modify their functionality and behavior. These attributes are written inside the opening tag. For example,

```
<img width=”300” height=”200” />
```

We have an image element with a “width” attribute with the value 300 and “height” attribute with value 200, which as you might guess, will make the image 300px wide and 200px tall.

Let’s look at another example.

```
<textarea rows="5" cols="20">
</textarea>
```

The very aptly named textarea element will display a text input field where our users can write text. In this example, rows and cols are attributes that define the number of rows and columns the textarea should span respectively.

Attributes like width and height for img, or rows and cols for textarea are useful directly within HTML.

But some attributes have a special meaning—meaning that they don't do anything on their own, but require us to write additional CSS or JavaScript, and thus connect the three pillars together—and we'll be learning more about them later in this training.

Note that **some elements don't have any content in them**, and hence they don't have to have a closing tag. Images, for example, only need a "src" attribute (short for source, or the location to find the image).

```
<img src="/path/to/image.png" />
```

Notice the /> at the end (instead of </img>). This is because image elements have a source attribute (src) which fetches the image to be displayed. There's no content that needs to go inside. There are other elements, similar to img, that don't require a closing tag.

Nesting elements

HTML elements can be nested inside each other; in other words, one element can hold other elements. Take a look at the following block of code.

```HTML
<p>I’m working as a <strong>Web
Developer</strong> at <strong>XYZ
Company</strong></p>

```

I’m working as a **Web Developer** at **XYZ Company**

Notice how we have two strong elements in our paragraph element. That’s totally legal.

For ease of reading, we can format the previous block of code as follows:

```HTML
<p>
  I’m working as a
  <strong>Web Developer</strong>
  at
  <strong>CareerFoundry</strong>
</p>
```

I’m working as a **Web Developer** at **CareerFoundry**

HTML doesn’t care how much space or how many new lines you use. The previous two examples will be displayed in the exact same way.

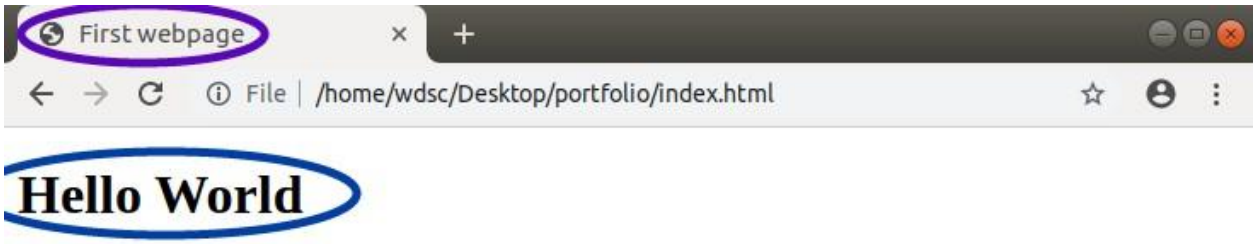
Other rules

Apart from these, there are a few basic rules that apply to all HTML pages. For example, The outermost HTML element needs to be <html> itself. Similarly, all 'visible' content goes into <body> while all configuration / metadata (data about the page itself) goes into <head>.

```
<!DOCTYPE html>
<html>

  <head>
    <title>First webpage</title>
  </head>

  <body>
    <h1>Hello World</h1>
  </body>

</html>
```

Remember our first webpage's code from earlier?

That was the reason <title> went into the <head>, and the browser picked it up and displayed it as the webpage's title (while it wasn't visible inside the page).



Here we have:

1. **<!DOCTYPE html>**: When HTML was young (1991-1992), doctypes were meant to act as links to a set of rules that the HTML page had to follow to be considered good HTML.

More recently, the doctype is a historical artifact that needs to be included for everything else to work right. <!DOCTYPE html> is the shortest string of characters that counts as a valid doctype.

2. **<html></html>**: This element wraps all the content on the page. It is sometimes known as the root element.
3. **<head></head>**: This element acts as a container for everything you want to include on the HTML page, that isn't the content the page will show to viewers. This includes
    - keywords and a page description that would appear in search results
    - CSS to style content,
    - JS (Javascript) to interact programmatically
    - character set declarations and more.

4. **<title></title>**: This sets the title of the page, which is the title that appears in the browser tab the page is loaded in. The page title is also used to describe the page when it is bookmarked.
5. **<body></body>:** This contains all the content that displays on the page, including text, images, videos, games, playable audio tracks, or whatever else.

HTML comments

HTML has a mechanism to write comments in the code. **Browsers ignore comments**, effectively making comments invisible to the user.
The purpose of comments is to allow you to include notes in the code to explain your logic or coding. This is very useful if you return to a code base after being away for long enough that you don't completely remember it. Likewise, comments are invaluable as different people are making changes and updates.

To write an HTML comment, wrap it in the special markers <!-- and -->. For example:

```
<p>I'm not inside a comment</p>

<!-- <p>I am inside a coment so i am not
visible!</p> -->
```

I'm not inside a comment

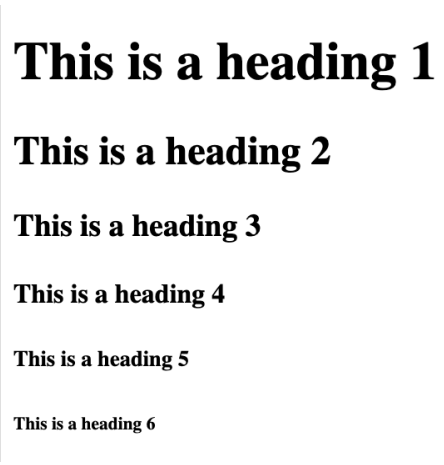
HTML elements

Now that we have a basic understanding of HTML elements, let's look at some of the basic elements.

Headings

Headings are exactly as the name suggests. In HTML, there are six headings, ranging from h1 to h6. Heading 1, or h1, is the largest and most significant heading; it signals that this is the most important text on the page. The significance decreases gradually as we move towards h6.

```
<h1>This is a heading 1</h1>
<h2>This is a heading 2</h2>
<h3>This is a heading 3</h3>
<h4>This is a heading 4</h4>
<h5>This is a heading 5</h5>
<h6>This is a heading 6</h6>
```



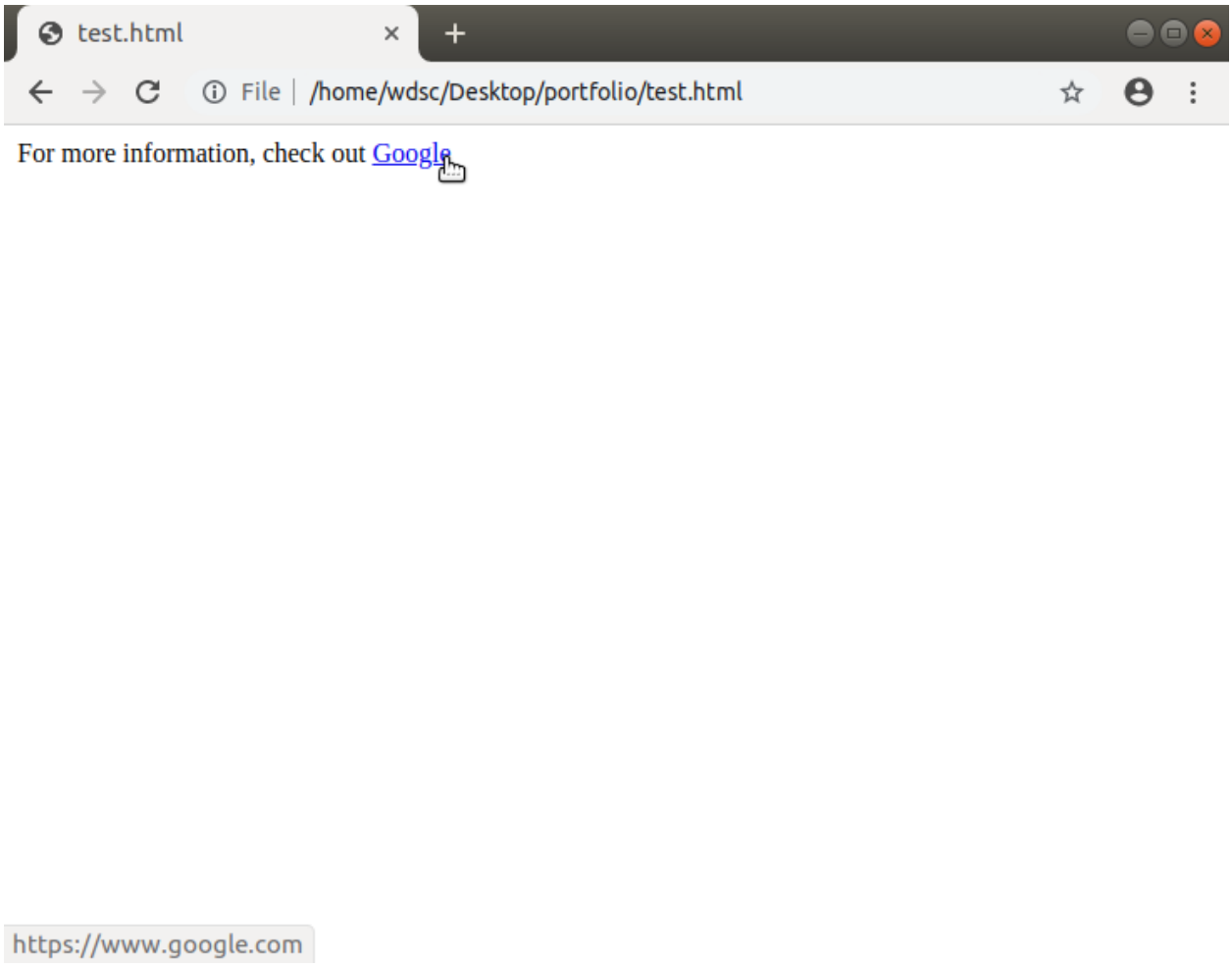
Anchor Links

The anchor element, a, enables the HyperText in HTML. It can link to another page on the same or a different website. Here's how to create an anchor link to Google's homepage:

```
For more information, check out
<a href="https://www.google.com">Google</a>
```

For more information, check out Google

This code will display as follows. Notice how hovering the mouse pointer on the link shows the anchor href value in the bottom left corner of the page. You must've clicked a couple of such anchor links to reach this very page!



Hyperlinks are one of the most exciting innovations the Web has to offer. They've been a feature of the Web since the beginning, and are what makes the Web a web.

Hyperlinks allow us to

- link documents to other documents or resources,
- link to specific parts of documents, or
- make apps available at a web address.

Almost any web content can be converted to a link so that when clicked or otherwise activated the web browser goes to another web address (URL).

A URL can point to HTML files, text files, images, text documents, video and audio files, or anything else that lives on the Web.

A basic link is created by wrapping the text or other content inside an <a> element and using the href attribute, also known as a Hypertext Reference, or target, that contains the web address.

```
<p>
  I'm creating a link to
  <a href="https://google.com">Google</a>.
</p>
```

I'm creating a link to Google.

Block-level link: Almost any content can be made into a link, even block-level elements. If you want to make a heading element a link then wrap it in an anchor (<a>) element as shown in the following code snippet:

```
<a href="https://google.com/">
  <h1>Google </h1>
</a>

<p>
  Best Search Engine the world has
when compared with yahoo, being or
other search engines .
</p>
```

Google

Best Search Engine the world has when compared with yahoo, being or other search engines .

Adding supporting information with the title attribute: The title contains additional information about the link, such as which kind of information the page contains, or things to be aware of on the website.

```
<p>
  I'm creating a link to
  <a
    href="https://google.com"
    title="Best Search Engine the
world has when compared with yahoo,
being or other search engines .">

    Google Search Engine</a>.
</p>
```

I'm creating a link to Google Search Engine.

Best Search Engine the world has when compared with yahoo, being or other search engines .

URLs and paths: Paths specify where the file you're interested in is located in the filesystem.

Same directory: If you want to include a hyperlink to another page or resource inside the same folder, just use the resource name with its extension:

```
<p>
  Do you Want to contact us?
  <a href="contacts.html">contacts page</a>.
</p>
```

Do you Want to contact us? contacts page.

Moving down into subdirectories: If you wanted to include a hyperlink pointing to projects/project1.html,

```
<p>Visit my <a href="projects/project1.html">project</a>.</p>
```

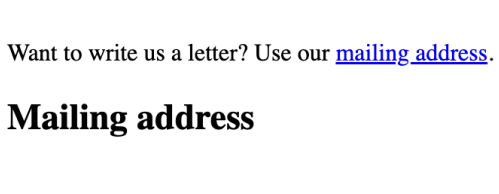
Moving back up into parent directories: If you wanted to include a hyperlink inside projects/index.html pointing to pdfs/project-brief.pdf, you'd have to go up a directory level, then back down into the pdfs directory.

```
<p>
  A link to my <a href="../pdfs/project-brief.pdf">project brief</a>.
</p>
```

Document fragments: It's possible to link to a specific part of an HTML document, known as a document fragment.

To do this you first have to assign an **id** attribute to the element you want to link to.

```
<p>
  Want to write us a letter? Use our
  <a href="#Mailing_address">mailing address</a>.
</p>
<h2 id="Mailing_address">Mailing address</h2>
```



Absolute versus relative URLs:

Absolute URL: Points to a location defined by its absolute location on the web, including protocol and domain name. An absolute URL will always point to the same location, no matter where it's used.

```
<p>
  Visit
  <a href="https://www.example.com/projects/index.html">
    My Project.
  </a>
</p>
```

Relative URL: Points to a location that is relative to the file you are linking from, more like what we looked at in the previous section.

```
<p>
  Visit
  <a href="projects/index.html">
    My Project.
  </a>
</p>
```

Paragraphs

The paragraph element, p, is used for text blocks. We usually style paragraphs such that they have a nice space between one another and between the first paragraph and its heading.

```
<p>This is a paragraph</p>
```

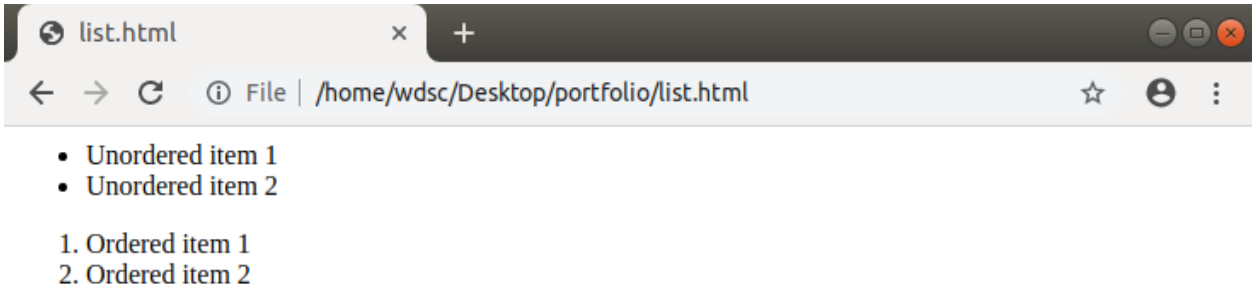
This is a paragraph

Lists

Lists are very useful for displaying data in an ordered or unordered list. For ordered lists (a list that uses numbers) we use <ol> and for unordered lists (a list with bullet points), we use <ul>. Within one of these elements, each list item is denoted by <li>. Here's an example:

```
<ul>
  <li>Unordered item 1</li>
  <li>Unordered item 2</li>
</ul>
<ol>
  <li>Ordered item 1</li>
  <li>Ordered item 2</li>
</ol>
```

Here's how our example 'renders' (which is just a fancy word meaning how it looks in our browser when we refresh the page).



Task

Create a Food Recipies Web Page using the following code you can change the food type and the recipes to one of Ethiopian food SHIRO WET.

```
<!doctype html>
<html lang="en-US">
    <head>
      <title>Ethiopian Shiro Wat recipe</title>
    </head>
    <body>
        <h1>Quick Ethiopian Shiro Wat recipe</h1>


      <p>Shiro is a staple vegetarian dish in Ethiopian cuisine.
      It’s basically a stew or curry made from ground dried
      chickpeas and various spices.  The chickpeas give the stew
      a beautiful texture and nutty flavor.  It’s very smooth
      and spreadable, almost the consistency of a thickened
      pureed soup.  Just mix and stir! </p>


      <p>The flavors that enhance the Shiro are the classic
      Ethiopian ingredients- Berbere spice , Niter Kibbe
      (Ethiopian spiced clarified butter), onions, tomatoes, and
      garlic. Getting these ingredients singing in the right
      combination is the art of this dish.</p>

       <h2>Ingredients</h2>
       <ul>
            <li>½ cup oil</li>
            <li>½ cup chickpea flour</li>
            <li>2 medium onions pureed</li>
            <li>1 tomato</li>
            <li>4 cloves of garlic chopped</li>
            <li>2 tablespoons niter kibbeh </li>
            <li>2 to 2 ½ cups of water</li>
            <li>3 tablespoons berbere spice</li>
            <li>1 teaspoon garlic powder</li>
            <li>Salt to taste</li>
       </ul>

       <h2>Instructions</h2>
       <ol>
```

```
            <li>Bring a heavy bottom stockpot to medium heat. Add
            pureed onions to the dry pan, and saute until they
            become dry and start to take on color- about 4-5
            minutes. Add the oil and berbere spice. Saute for 1-2
            minutes until fragrant.</li>

            <li>Next add tomato and chopped garlic. Saute for 2-3
            minutes more.</li>

            <li>Now start whisking in about half of the chickpea
            flour. Gradually start to add about 1 cup of water.
            Whisk in the remaining chickpea flour and an
            additional 1 cup of water. Whisk until mixture is
            very smooth. Add remaining ½ cup of water if you
            prefer your shiro a little thinner.</li>

            <li>Heat until the shiro begins to pop (simmer). Then
            add the niter kibbeh, garlic powder, sugar, and salt
            to taste, stirring until combined.</li>

            <li>Simmer for about 5-10 minutes over low heat until
            the flavors combine and the oil separates slightly
            from the shiro.</li>

            <li>Serve with fresh injera.</li>
        </ol>

         <h2>An aside</h2>
         <p>if anyone is at all interested- you can acquire
Shiro Powder to make this dish as well. Shiro powder is
different from chickpea flour/ besan.   It is actually a
mixture of chickpea flour, spices, and seasonings.  It is a
just-add-water type dish that is an even faster process to
make.</p>

    </body>

 </html>
```

The resulting webpage should look like this

Quick Ethiopian Shiro Wat recipe

Shiro is a staple vegetarian dish in Ethiopian cuisine. It's basically a stew or curry made from ground dried chickpeas and various spices. The chickpeas give the stew a beautiful texture and nutty flavor. It's very smooth and spreadable, almost the consistency of a thickened pureed soup. Just mix and stir!

The flavors that enhance the Shiro are the classic Ethiopian ingredients- Berbere spice , Niter Kibbe (Ethiopian spiced clarified butter), onions, tomatoes, and garlic. Getting these ingredients singing in the right combination is the art of this dish.

Ingredients

- ½ cup oil
- ½ cup chickpea flour
- 2 medium onions pureed
- 1 tomato
- 4 cloves of garlic chopped
- 2 tablespoons niter kibbeh
- 2 to 2 ½ cups of water
- 3 tablespoons berbere spice
- 1 teaspoon garlic powder
- Salt to taste

Instructions

1. Bring a heavy bottom stockpot to medium heat. Add pureed onions to the dry pan, and saute until they become dry and start to take on color- about 4-5 minutes. Add the oil and berbere spice. Saute for 1-2 minutes until fragrant.
2. Next add tomato and chopped garlic. Saute for 2-3 minutes more.
3. Now start whisking in about half of the chickpea flour. Gradually start to add about 1 cup of water. Whisk in the remaining chickpea flour and an additional 1 cup of water. Whisk until mixture is very smooth. Add remaining ½ cup of water if you prefer your shiro a little thinner.
4. Heat until the shiro begins to pop (simmer). Then add the niter kibbeh, garlic powder, sugar, and salt to taste, stirring until combined.
5. Simmer for about 5-10 minutes over low heat until the flavors combine and the oil separates slightly from the shiro.
6. Serve with fresh injera.

We can also Nest Lists

```
<ol>
  <li>Remove the skin from the garlic, and chop coarsely.</li>
  <li>Remove all the seeds and stalk from the pepper, and chop
coarsely.</li>
  <li>Add all the ingredients into a food processor.</li>
  <li>
    Process all the ingredients into a paste.
    <ul>
      <li>
        If you want a coarse "chunky" hummus, process it for a short
time.
      </li>
      <li>If you want a smooth hummus, process it for a longer time.
</li>
    </ul>
  </li>
</ol>
```

1. Remove the skin from the garlic, and chop coarsely.
2. Remove all the seeds and stalk from the pepper, and chop coarsely.
3. Add all the ingredients into a food processor.
4. Process all the ingredients into a paste.
    - If you want a coarse "chunky" hummus, process it for a short time.
    - If you want a smooth hummus, process it for a longer time.

Italic, bold, underlie

<i> is used to convey a meaning traditionally conveyed by italic: foreign words, taxonomic designation, technical terms, a thought…

<b> is used to convey a meaning traditionally conveyed by bold: keywords, product names, lead sentence…

<u> is used to convey a meaning traditionally conveyed by underline

```
<!-- scientific names -->
<p>
  The Ruby-throated Hummingbird (<i>Archilochus colubris</i>) is the most common
  hummingbird in Eastern North America.
</p>

<!-- foreign words -->
<p>
  The menu was a sea of exotic words like <i>vatrushka</i>,
  <i>nasi goreng</i> and <i>soupe à l'oignon</i>.
</p>

<!-- a known misspelling -->
<p>Someday I'll learn how to <u>spel</u> better.</p>
```

The result should look like this

The Ruby-throated Hummingbird (*Archilochus colubris*) is the most common hummingbird in Eastern North America.

The menu was a sea of exotic words like *vatrushka*, *nasi goreng* and *soupe à l'oignon*.

Someday I'll learn how to spel better.

Multimedia

The web would be really boring only using text. Let's start looking at how to make the web come alive with more interesting content! We will now see how to use HTML to include multimedia in your web pages, including the different ways that images can be included, and how to embed video and audio.

Images in HTML:

<img> element: requires two attributes to be useful: src and alt.

src : attribute contains a URL pointing to the image you want to embed in the page.

alt : It is a textual description of the image, and will be displayed when the image is not found on the **url** noted on **src**

```
<img
 src="https://www.example.com/images/dinosaur.jpg"
 alt="Dinosaur" />
```

Dinosaur

```
<img
src="https://media.sciencephoto.com/c0/53/97/88/c05
39788-400px-wm.jpg"
alt="Dinosaur" />
```

width and height to set the intrinsic size of the image, allowing it to take up space before it loads, to mitigate content layout shifts.

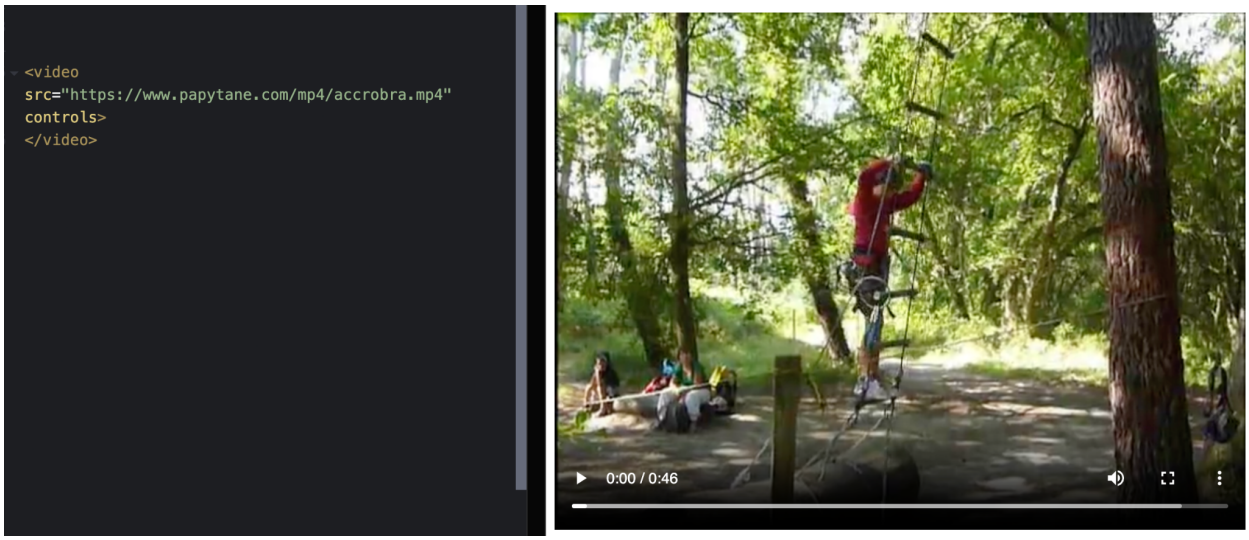
title: The attribute is usually presented to the user as a tooltip, which appears shortly after the cursor stops moving over the image.

```
<img
 src="https://media.sciencephoto.com/c0/53/97/88/c0
539788-400px-wm.jpg"
 alt="The head and torso of a dinosaur skeleton;
      it has a large head with long sharp teeth"
 width="300"
 height="200"
 title="A T-Rex on display in the Manchester
University Museum" />
```

Video In HTML

<video> element allows you to embed a video very easily. A really simple example looks like this

```
<video
src="https://www.papytane.com/mp4/accrobra.mp4"
controls>
</video>
```



src: in the same way as for the <img> element, the src (source) attribute contains a path to the video you want to embed.

controls: Users must be able to control video and audio playback. You must either use the controls attribute to include the browser's own control interface or build your interface using the appropriate JavaScript API.

```
<video
  controls
  width="400"
  height="400"
  autoplay
  loop
  muted
  preload="auto"
  poster="poster.png">
  <source src="rabbit320.mp4" type="video/mp4" />
  <source src="rabbit320.webm" type="video/webm" />
</video>
```

<source> elements that point to their own sources. In this case the browser will go through the <source> elements and play the first one that it has the codec to support.

width and height : same as image you can decide on the video width and size

autoplay: Makes the audio or video start playing right away.

loop: Makes the video (or audio) start playing again whenever it finishes.

Muted: Causes the media to play with the sound turned off by default.

poster: The URL of an image which will be displayed before the video is played.

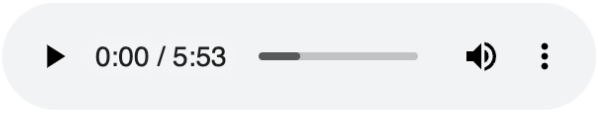
preload: Used for buffering large files; it can take one of three values:

- "none" does not buffer the file
- "auto" buffers the media file
- "metadata" buffers only the metadata for the file

The <audio> element

The <audio> element works just like the <video> element. A typical example might look like so:

```
<audio
controls
autoplay
loop
src="https://aveclagare.org/mp3/050717Alphaze-EP-NOW-%20Ladzik.mp3"
      >
</audio>
```



Divisions and spans

Everything on a webpage can be imagined to be contained in a series of boxes. Our job as web developers is to arrange these boxes so that the whole page looks nice on all screens. These boxes contain text, images, and everything else that we see on webpages.

**The ** HTML element is the generic container for flow content. It has no effect on the content or layout until styled in some way using CSS.

As a "pure" container, the element does not inherently represent anything. Instead, it's used to group content so it can be easily styled using the class or id attributes

```
<div>
  <p>
    Any kind of content here.
  </p>
</div>
```

Any kind of content here.

**The ** HTML element is a generic inline container for phrasing content, which does not inherently represent anything until used by css or js.

```
<p>
  <span>Some text</span>
</p>
```

Some text

HTML Table

The HTML table tag (<table>) is used to represent data in a structured way by creating a table. For example,

```
<table border="1" >
  <tr>
    <th>Name</th>
    <th>Age</th>
    <th>Country</th>
  </tr>
  <tr>
    <td>Harry Depp</td>
    <td>28</td>
    <td>Britain</td>
  </tr>
  <tr>
    <td>John Smith</td>
    <td>35</td>
    <td>USA</td>
  </tr>
  <tr>
    <td>Ram Krishna</td>
    <td>19</td>
    <td>Nepal</td>
  </tr>
</table>
```

Browser Output

Name	Age	Country
Harry Depp	28	Britain
John Smith	35	USA
Ram Krishna	19	Nepal

In the above example, you can see we have used multiple tags to create a table in HTML.

- <table>
- <tr>
- <td>
- <th>

Table tag <table> in HTML

The <table> tag is used to define a table. For example,

```
<table>
  ….
<table>
```

Table Row <tr> in HTML

The <tr> tag is used to define a row in a table. For example,

```
<table>
<tr>
  ...
</tr>
</table>
```

The table row can include either table heading, <th> or table data, <td>.

```
<tr>
  <th>Name</th>
  <th>Country</th>
</tr>
<tr>
  <td>Prasanna</td>
  <td>Nepal</td>
</tr>
<tr>
  <td>Simon</td>
  <td>USA</td>
</tr>
```

In a table, there can be any number of rows.

Table Heading, <th> in HTML

The <th> tag is used to define a table header. It is generally the top row of the table. For example,

```
<table>
  <tr>
    <th>Item</th>
    <th>Count</th>
  </tr>
  <tr>
    <td>Mango</td>
    <td>125</td>
  </tr>
  <tr>
    <td>Orange</td>
    <td>75</td>
  </tr>
</table>
```

Browser Output

Item	Count
Mango	125
Orange	75

In the above example, Item and Count are table headers and they are used to represent the category of data in a particular row.
Here, the styling of the table headers is bold and center-aligned. This is because the <th> tag has some default styling.

Table Cell <td> in HTML

The <td> tag is used to define table cells (data). The table cells store data to be displayed in the table. For example,

```
<tr>
  <td>Apple</td>
  <td>Mango</td>
  <td>Orange</td>
</tr>
```

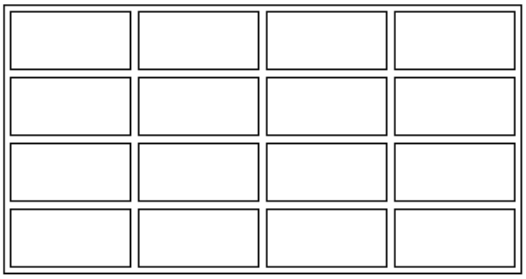
In the above example, <td>Apple</td>, <td>Mango</td> and <td>Orange</td> are table cells.
Table cells are generally inside the table row or table headers.

Table Border

Remember we have used the border attribute in our first example.

```
<table border="1">
  …
</table>
```

In HTML, the border attribute is used to add a border to a table and all the cells.



Note: We can have borders of various styles in tables, however for more specific borders, we need to use CSS.

To prevent double borders like the one in the example above, we can set the border-collapse property of the table. For example,

```
<table border="1" style="border-collapse: collapse;">
  ...
</table>
```

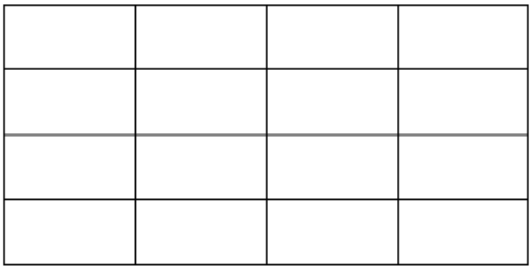


Table Head, Body, and Footer

The HTML table can be divided into three parts: a header, a body, and a footer.

1. Table Header

We use the <thead> tag to add a table head. The <thead> tag must come before any other tags inside a table. For example,

```
<table>
  <thead>
    <tr>
        <th>Head1</th>
        <th>Head2</th>
    </tr>
  </thead>

  ...
  ...
</table>
```

The content of <thead> is placed on the top part of the table and we usually place the rows with table headers inside the <thead> tag.

2. Table Body

We use the <tbody> tag to add a table body. The <tbody> tag must come after <thead> and before any other tags inside a table. For example,

```
<table>
  <thead>
    ...
  </thead>
  <tbody>
    <tr>
      <td>Cell 1</td>
      <td>Cell 2</td>
    </tr>
  </tbody>

  ...
  ...
</table>
```

The content of <tbody> is placed on the center part of the table and we usually place the rows with the content we want to represent in the <tbody>.

3. Table Footer

We use the <tfoot> tag to add a table footer. The <tfoot> tag must come after <tbody> and before any other tags inside a table. For example,

```
<table>
  <thead>
    ...
  </thead>
  <tbody>
    ,,,,
  </tbody>
  <tfoot>
    <tr>
      <td>foot 1</td>
      <td>foot 2</td>
    </tr>
  </tfoot>
</table>
```

The content of <tbody> is placed on the bottom part of the table and we usually place the rows with the footer in the <tfoot>.

All these tags must be placed inside a <table> tag and must contain at least one <tr>. For example,

Example: HTML Table Head, Body, and Footer

```
<table>
  <thead>
    <tr>
      <th>S.N</th>
      <th>Item</th>
      <th>Quantity</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>1</td>
      <td>Apple</td>
      <td>2</td>
    </tr>
    <tr>
      <td>2</td>
      <td>Mango</td>
      <td>2</td>
    </tr>
    <tr>
      <td>3</td>
      <td>Orange</td>
      <td>1</td>
    </tr>
  </tbody>
  <tfoot>
    <tr>
      <td></td>
      <td>Total</td>
      <td>5</td>
    </tr>
  </tfoot>
</table>
```

Browser Output

S.N	Item	Quantity
1	Apple	2
2	Mango	2
3	Orange	1
	Total	5

Colspan and Rowspan

Colspan

The colspan attribute merges cells across multiple columns. For example,

```
<table>
  <tr>
    <th>S.N</th>
    <th>Item</th>
    <th>Quantity</th>
  </tr>
  <tr>
    <td>1</td>
    <td>Apple</td>
    <td>2</td>
  </tr>
  <tr>
    <td>2</td>
    <td>Mango</td>
    <td>2</td>
  </tr>
  <tr>
    <td>3</td>
    <td>Orange</td>
    <td>1</td>
  </tr>
  <tr>
    <td colspan="2">Total</td>
    <td>5</td>
  </tr>
</table>
```

Browser Output

S.N	Item	Quantity
1	Apple	2
2	Mango	2
3	Orange	1
Total		5

In the above example, you can see that the last row only has 2 cells with one cell occupying 2 columns.

The value of the colspan attribute determines how many columns the cell occupies.

Rowspan

The rowspan attribute merges cells across multiple rows. For example,

```
      <table>
        <tr>
          <th>Name</th>
          <th>Subject</th>
          <th>Marks</th>
        </tr>
        <tr>
          <td rowspan="3">Mark Smith</td>
          <td>English</td>
          <td>67</td>
        </tr>
        <tr>
          <td>Maths</td>
          <td>82</td>
        </tr>
        <tr>
          <td>Science</td>
          <td>91</td>
  </tr>
</table>
```

Browser Output

Name	Subject	Marks
Mark Smith	English	67
	Maths	82
	Science	91

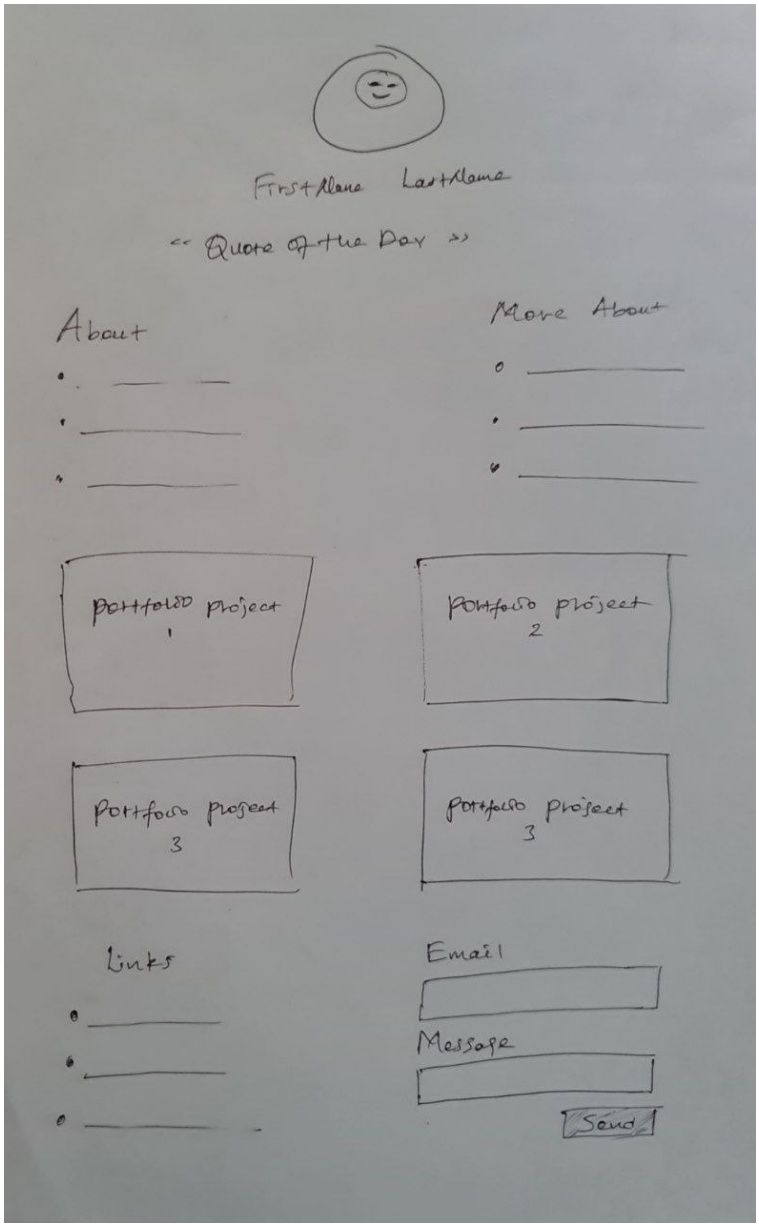
In the above example, you can see that the first column only has 2 cells with one cell occupying 2 rows.

The value of the rowspan attribute determines how many rows the cell occupies.

Creating the basic page layout

Now we know enough HTML elements to start adding HTML to our portfolio page project! Before we get into writing code, let's take a look at our wireframe mockup. A wireframe is a very simple design that we use as a reference to code our website.

In the real world, your team may have dedicated designers who'll come up with a design that is then handed over to you, the developer, for implementation (converting into actual code). In our case, we'll use a hand drawn design as a starting point. The purpose it serves is similar: it gives us a broad outline for how our end result should look.



Looking at the mockup, we can roughly compartmentalize our page into sections.

- Introduction
    - Profile picture
    - Name
    - Professional title
    - Quote
- About
    - Who I am
    - What I like
- Portfolio
- Links and contact form

It is generally helpful to think in terms of sections, because as you'll see, each of these bullet points will become a box in itself, with the sub points getting nested inside the main points. Let's take each of these points and tackle them separately.

Introduction

The introduction section contains

- an image (profile picture),
- a heading (name),
- a subheading (professional title) and
- a line of text (quote).

We can start with the introduction box and add each of the nested elements into it.

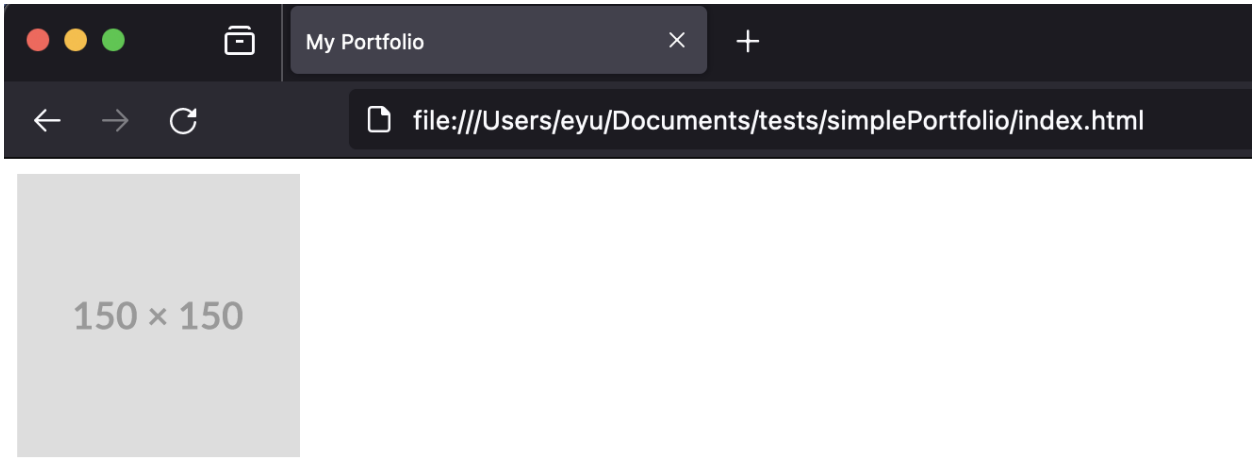
Note that this code goes inside the body tag, that is, in between the opening and closing body tag (<body> and </body>).

Remember what we said about the div element? It's just like a box that holds our content together. Inside the box, we have all the elements we mentioned above.

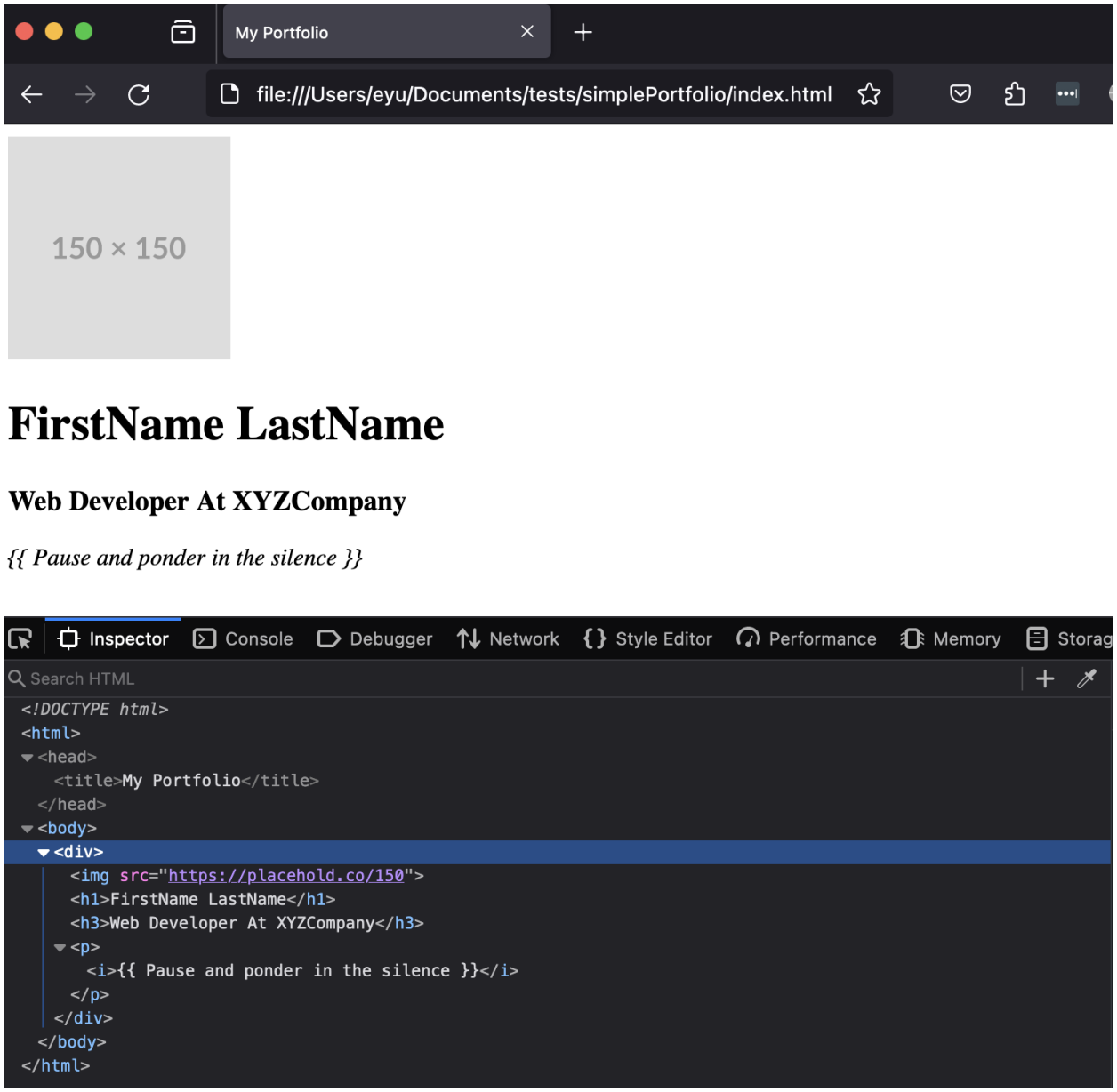
Notice the https://placehold.co/150 in image source (<img src=)? That is a placeholder image that we use while we're developing this website. We can replace it with our own image later once we're happy with our design.

```
<!DOCTYPE html>
<html>

    <head>
        <title>My Portfolio</title>
    </head>

    <body>

        <div>
            <img src="https://placehold.co/150" />
            <h1>FirstName LastName</h1>
            <h3>Web Developer At XYZCompany</h3>
            <p> <i> {{ Pause and ponder in the silence }} </i> </p>
        </div>

    </body>
</html>
```

Let’s see how that looks in a browser.



Open the developer tool and check the code that the browser interpreted



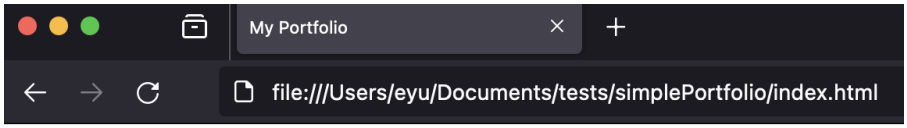
This is the code that your browser interpreted. You can try clicking on each element (img, h1, etc) and see that the browser highlights them for you.

About

Next, let's work on the About section.

```
<div>
      <div>
          <h3>I am</h3>
          <ul>
          <li>An engineer</li>
          <li>A reader</li>
          <li>A hobby cook</li>
          </ul>
      </div>
      <div>
          <h3>I like to</h3>
          <ul>
          <li>Meet new people</li>
          <li>Play guitar</li>
          <li>Eat Icecream</li>
          </ul>
      </div>
  </div>
```

Let's look at the result of that in our browser.



FirstName LastName

Web Developer At XYZCompany

{{ Pause and ponder in the silence }}

I am

- An engineer
- A reader
- A hobby cook

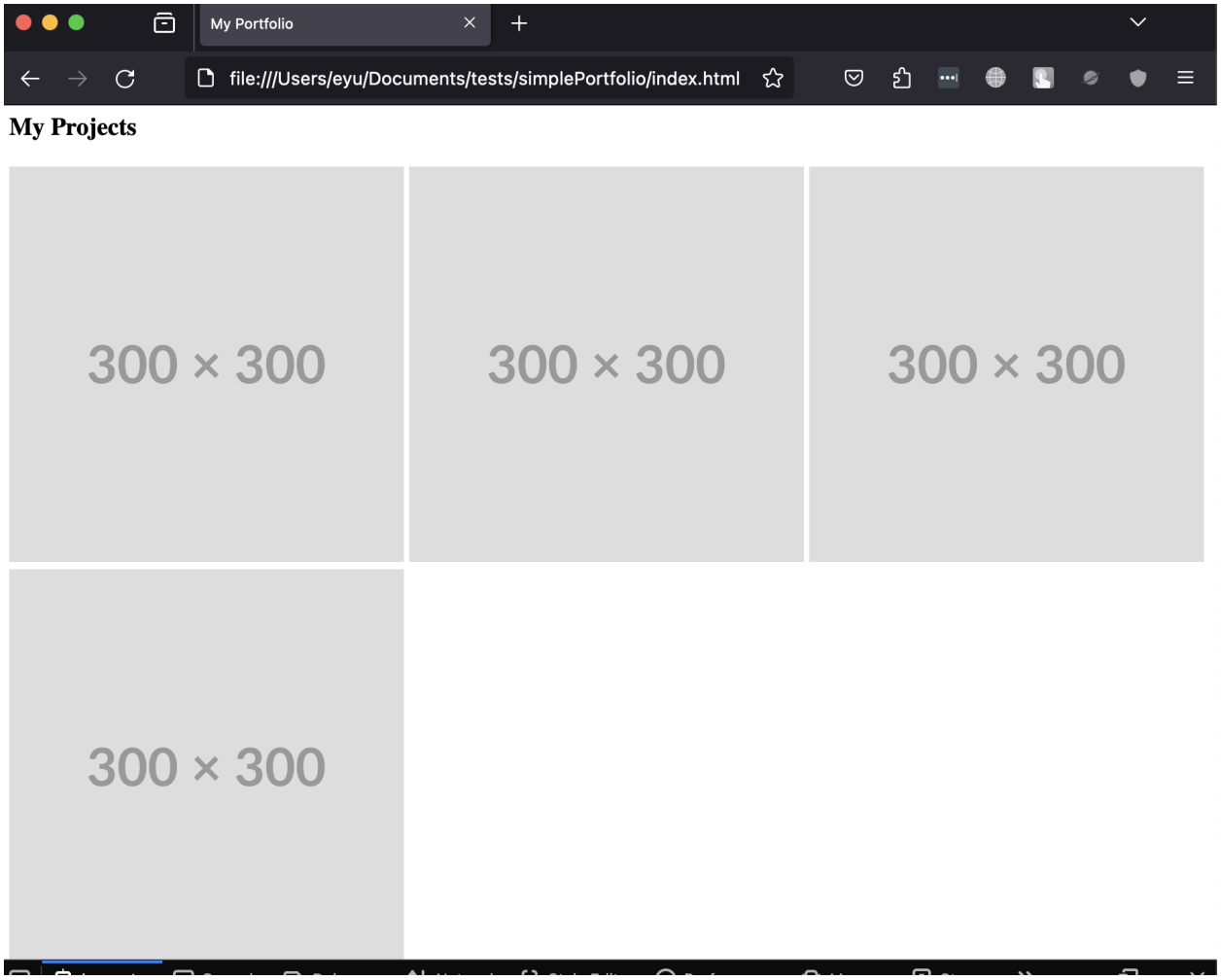
I like to

- Meet new people
- Play guitar
- Eat Icecream

Portfolio

Great! Now let's tackle the Portfolio section. This section will contain four of our chosen project screenshots. You'll see in our wireframe that we're planning to arrange them in a 2x2 grid. We'll be able to do that with CSS later in the training. For now, let's add a heading and four images using the <img> tag just below the previous section.

The result, after adding to our page, should look as follows:



Links and footer

Our final section is our footer (so called because it is the vertical end of the webpage). It contains some links to our online social profiles, like LinkedIn, Github, and Twitter, but you can replace them with your own custom links if you'd like!

```
<div>
      <h3>Links</h3>
      <ul>
        <li>
          <a href="https://github.com/<user>">Github</a>
        </li>
        <li>
          <a href="https://twitter.com/<user>">Twitter</a>
        </li>
        <li>
          <a href="https://linkedin.com/in/<user>">Linkedin</a>
        </li>
      </ul>
</div>
```

Links

- Github
- Twitter
- Linkedin

Contact form

Before we dive into creating the contact form in our portfolio page lets first learn a little bit about HTML Form.

<form> is an HTML element to collect input data containing interactive controls. Form is a container that contains input elements like text, email, number, radio buttons, checkboxes, submit buttons, etc. Forms are generally used when you want to collect data from the user.

Basic Syntax:

```
<form>
      <!--form elements-->
</form>
```

Form elements

These are one of the most commonly used HTML <form> elements:

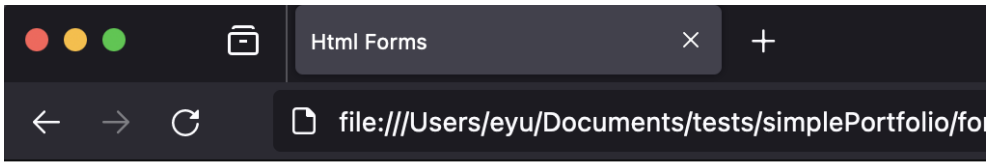
- <label>: It defines label for <form> elements.
- <input>: It is used to get input data from the form in various types such as text, password, email, etc by changing its type.
- <button>: It defines a clickable button to control other elements or execute a functionality.
- <select>: It is used to create a drop-down list.
- <option>: It is used to define options in a drop-down list.
- <textarea>: It is used to get input long text content.

Let us create a form that utilizes the above major form elements,

```
<!DOCTYPE html>
<html>
<head>
    <title>Html Forms</title>
</head>

<body>
    <h2>Form and Form Elements</h2>
    <form>
        <p>Gender :
            <label>Male</label>
            <input type="radio" name="gender" value="male" />
            <label>Female</label>
            <input type="radio" name="gender" value="female" />
        </p>
        <p>
```

```
            Language:
            <ul style="list-style-type:none;">
                <li>
                    <input type="checkbox" name="language" value="english" /> English
                </li>
                <li>
                    <input type="checkbox" name="language" value="german" /> German
                </li>

                <li>
                    <input type="checkbox" name="language" value="arabic" /> Arabic
                </li>
            </ul>
        </p>
        <p>
            <label>Nationality: </label>
            <select name="nationality">
                <option value="british">British</option>
                <option value="german">German</option>
                <option value="uae">UAE</option>
                <option value="others">Others</option>
            </select>
        </p>
        <p>
            <label>Username : </label>
            <input type="text" />
        </p>
        <p>
            <label>Password : </label>
            <input type="password" />
        </p>
        <p>
            <label>Brief Info About You: </label>
            <textarea name="briefInfo" rows="5" cols="40">
            </textarea>
        </p>
        <p>
            <button type="submit">Submit</button>
        </p>
    </form>
</body>
</html>
```

It should result in this



Form and Form Elements

Gender : Male ○ Female ○

Language:

- ☐ English
- ☐ German
- ☐ Arabic

Nationality: British

Username :

Password :

Brief Info About You:

Submit

Input types in html

- <input type="text"> : defines a single-line text input field
- <input type="checkbox"> : defines a checkbox, Checkboxes let a user select ZERO or MORE options of a limited number of choices.
- <input type="radio"> : defines a radio button, Radio buttons let a user select ONLY ONE of a limited number of choices:
- <input type="color"> : is used for input fields that should contain a color, Depending on browser support, a color picker can show up in the input field.
- <input type="date"> : is used for input fields that should contain a date, Depending on browser support, a date picker can show up in the input field.
- <input type="email"> : is used for input fields that should contain an e-mail address, Depending on browser support, the e-mail address can be automatically validated when submitted.

- <input type="file"> : defines a file-select field and a "Browse" button for file uploads.
- <input type="hidden"> : defines a hidden input field (not visible to a user).
- <input type="number"> : defines a numeric input field.
- <input type="password"> : defines a password input field, which will not show the password to the user but instead shows *
- <input type="url"> : is used for input fields that should contain a URL address, Depending on browser support, the url field can be automatically validated when submitted.

Drop-down menu is used to create a drop-down menu in your form which contains multiple options. we use the <select> tag with <option> tag.

Syntax:

```
<select name="select_box_name">
      <option value="value1">option1</option>
      <option value="value2">option2</option>
      <option value="value3">option3</option>
</select>
```

Submit button : is used to submit the details of the form to the form handler. A form handler is a file on the server with a script that is used to process input data.

Syntax:

```
<button type="submit">submit</button>
```

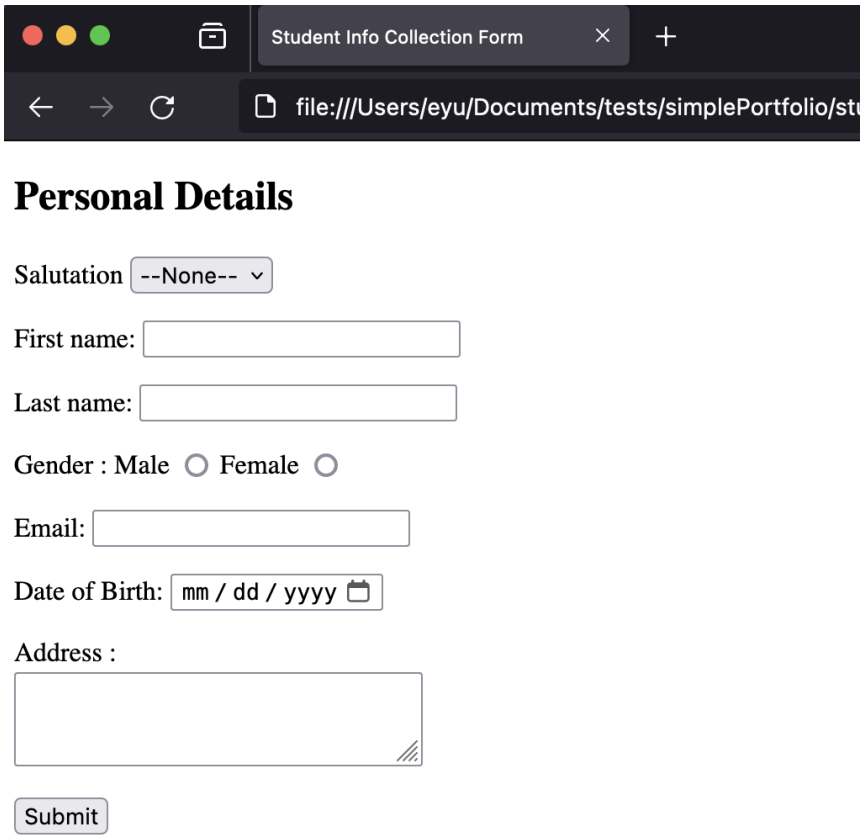
The <textarea> : element represents a multi-line plain-text editing control, useful when you want to allow users to enter a sizeable amount of free-form text

```
<textarea id="story" name="story" rows="5" cols="33">

</textarea>
```

rows and cols attributes to allow you to specify an exact size for the <textarea> to take.

Task

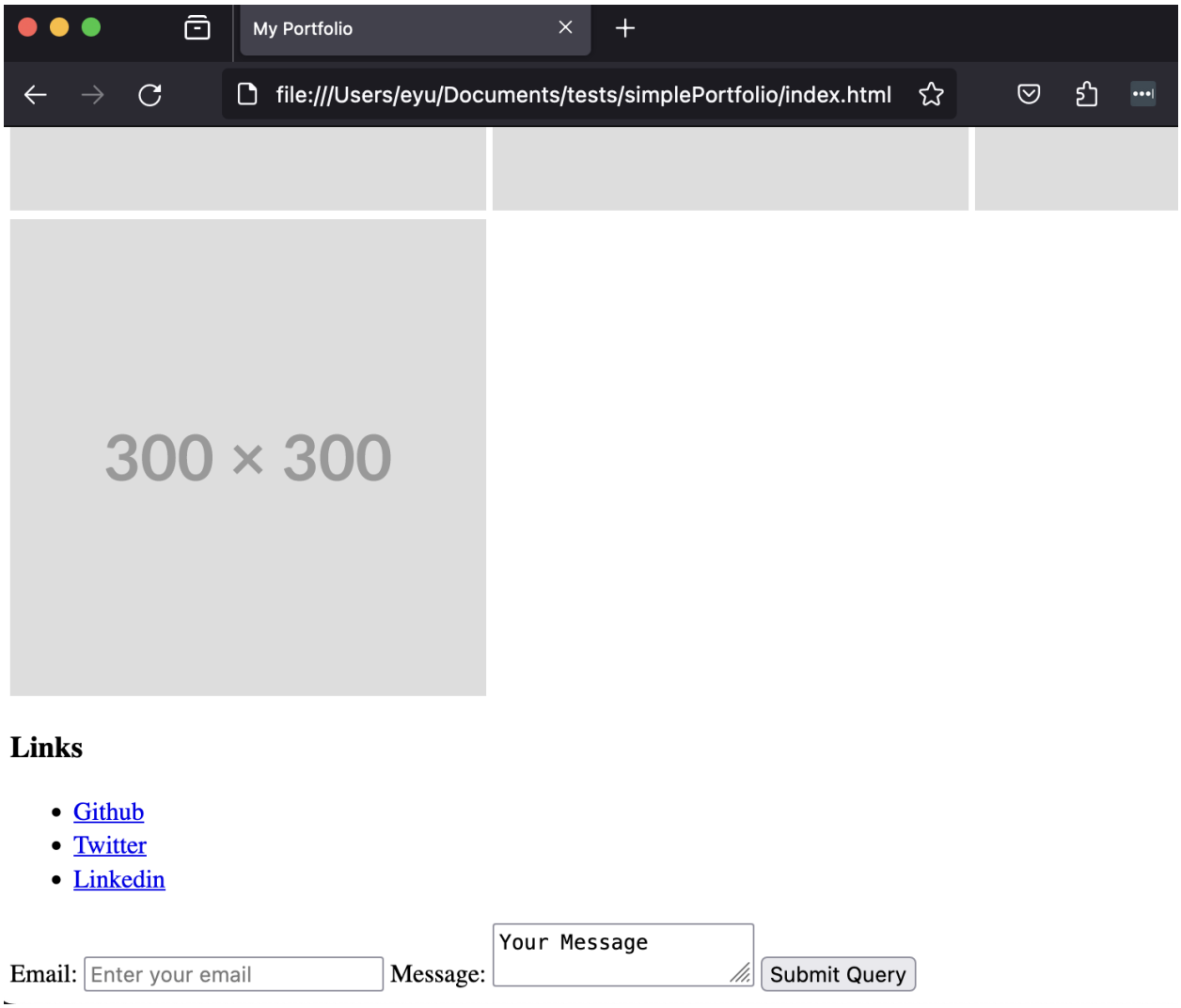
Create a student information collection for that looks like this



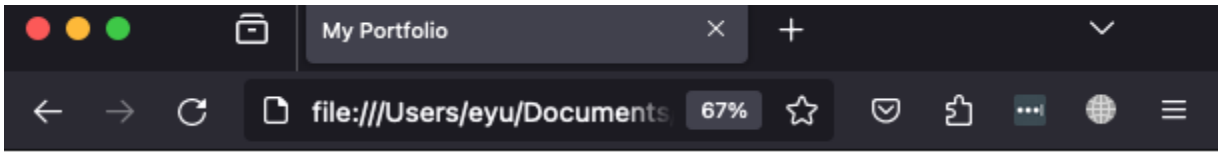
NOW Let continue to our Contact me section of our Portfolio site

On a real website, it would enable people to send us a message. For now, the form that we're writing is just on the front end. It won't work since we do not have a backend for it yet, we can create the backend when learn about php later on in this training

```
<div>
   <form action="#">
      <label for="email">
      Email: <input type="email" id="email" placeholder="Enter your email" />
      </label>
      <label for="message">
      Message: <textarea id="message">Your Message</textarea>
      </label>
      <input type="submit" valid="Send Message" />
   </form>
</div>
```



Putting it all Together



FirstName LastName

Web Developer At XYZCompany

{{ Pause and ponder in the silence }}

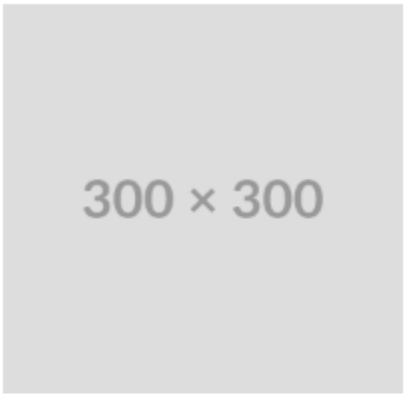
I am

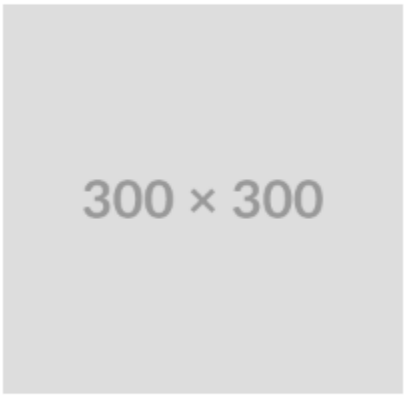
- An engineer
- A reader
- A hobby cook

I like to

- Meet new people
- Play guitar
- Eat Icecream

My Projects





Links

- Github
- Twitter
- Linkedin

Email: Enter your email Message: Your Message Submit Query