Lab 1 – Introduction to HTML

1 Introduction

Welcome to Web Development!

Each week will consist of a one-hour lecture and three-hour lab. While the lecture will focus on theory and understanding, the labs allow you to put what you have learnt into practice. The labs should be completed in order and in full before moving on. The labs build on each other, so it is important that you understand the concepts each week before progressing to the following weeks content.

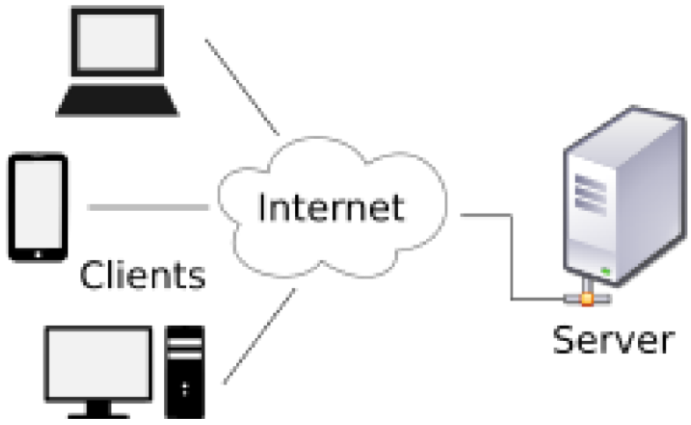
In this week's lab we introduce HTML. HTML is a markup language that allows you to define a websites structure and content. You will notice that raw HTML pages don't look pretty, and don't do anything interactive – we will cover that in upcoming weeks.

2 Learning Objectives

- To understand the fundamentals of how the web works
- To experience and understand the basics of HTML and what HTML is

3 How does the web work?

The web works off a client-server architecture. What this means is that clients, such as web browsers, make requests to servers. The servers then respond with content for the client to parse. In the case of the web, a web browser (the client) is used to send requests to a website (a server) located somewhere on the internet. The server responds with the web page that the browser then parses and displays on the screen.



Let's break it down:

1. You open a client-side web browser (e.g. Google Chrome)
2. You type the address of the website you wish to visit into the address bar
3. You hit enter
4. The browser sends a request across the internet
5. The request reaches the relevant server to be processed
6. Once processed, the server sends a response back to the client

7. The client receives the response, parses it, and displays some output to the user (e.g., displays the website on the screen)

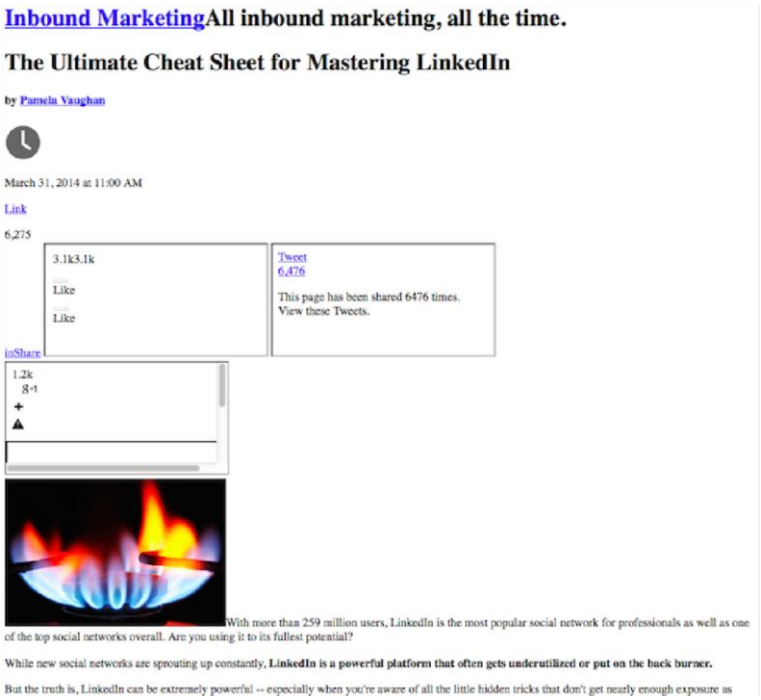
This transfer of information across the internet is done using the HyperText Transfer Protocol (HTTP), hence the “http://” you see at the start of URLs.

Note: the ‘s’ in https means that the website is using a secure version of HTTP.

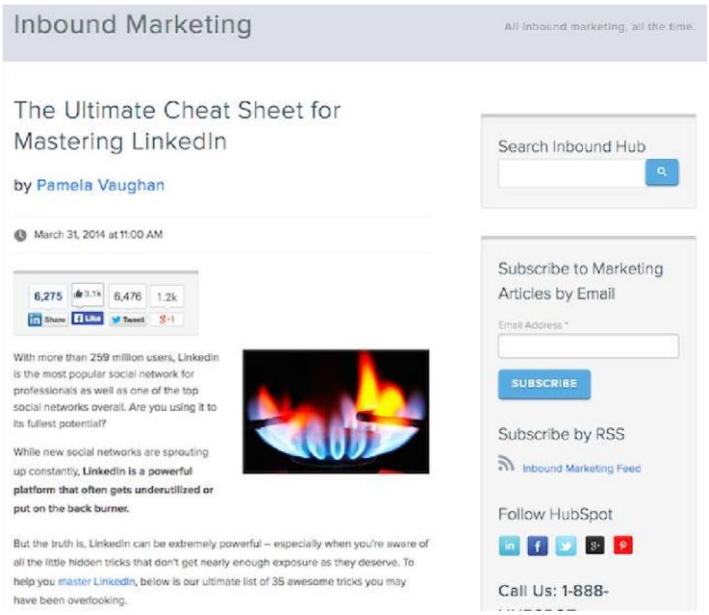
4 HTML (HyperText Markup Language)

HTML is the formatting language for the web and describes how web pages are structured. It includes all content for the page, but no styling or interactivity.

How many websites have you see that look like this?



This is what raw HTML looks like. All of the content is there (e.g. text, images, links) but there is no styling to make it look good. Normally, we see this when a browser has managed to receive the HTML from the server, but has failed to receive the CSS, maybe due to a poor connection. Hitting refresh a couple of times normally fixes it. Now the web page looks like this:

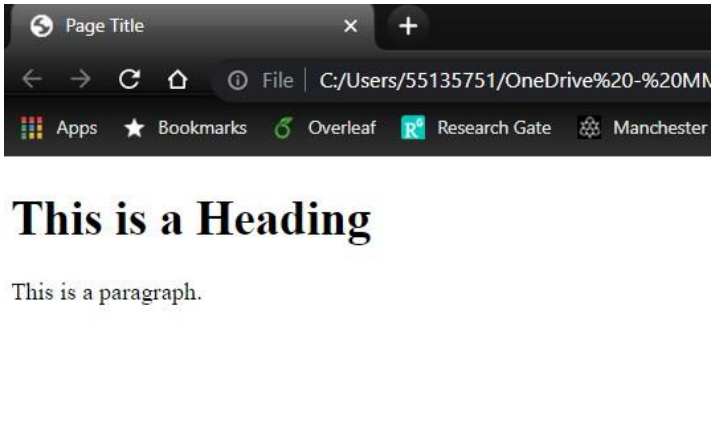


All the same content and structure is there (the text, images, links), but now the site is styled and looks how the developer intended it to be viewed. We will be looking at styling web pages next week.

4.1 Writing HTML

We write HTML to tell the browser how to parse the HTML in a graphically pleasing way. All browsers have to parse HTML the same way so that web pages look the same across all devices.

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>Page Title</title>
5      </head>
6
7      <body>
8
9          <h1>This is a Heading</h1>
10         <p>This is a paragraph.</p>
11
12     </body>
13 </html>
```



HTML is made up of HTML elements. Elements are defined by a starting tag, the content, and then a closing tag:

```
<tagname>Content goes here...</tagname>
```

Look at the above images. Can you identify the heading element and the paragraph element?

Some elements may contain other elements, such as the “body”, “head” and “html” elements in the above images.

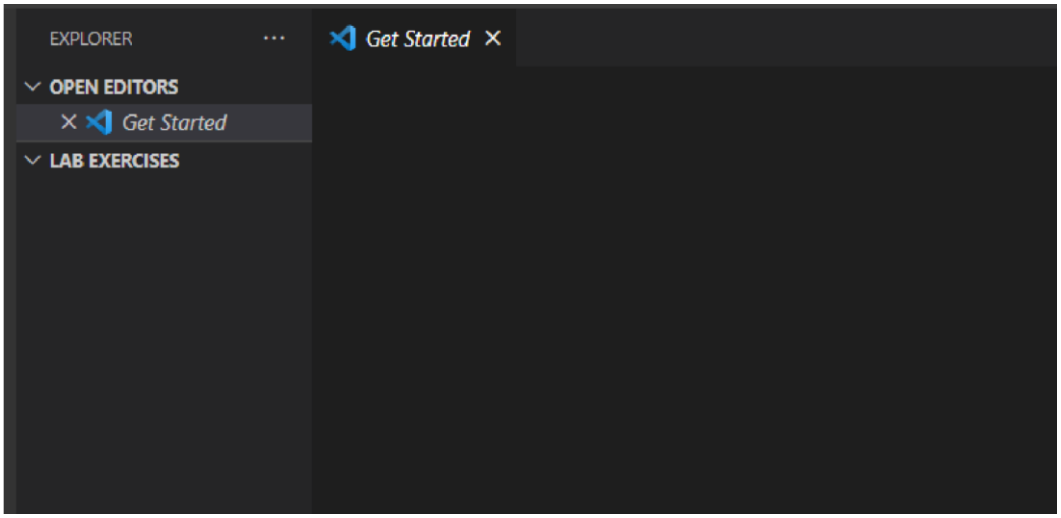
5 HTML Basics

Let's write some HTML pages.

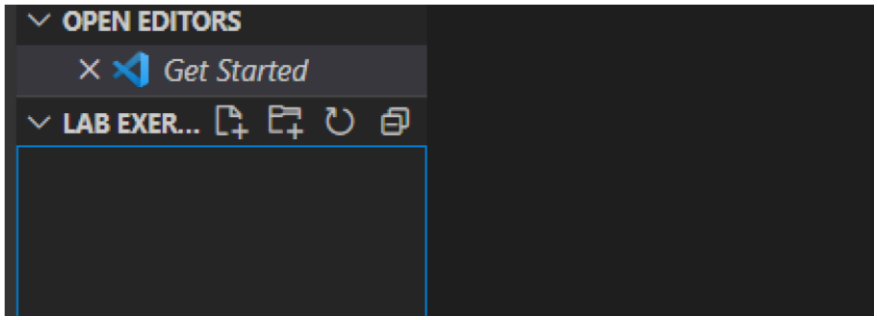
Note: the exercises assume that you are working on a campus lab machine. If you are on your own device, you will first need to install Visual Studio Code from:
https://code.visualstudio.com/download

5.1 Exercise 1: Creating your first HTML page

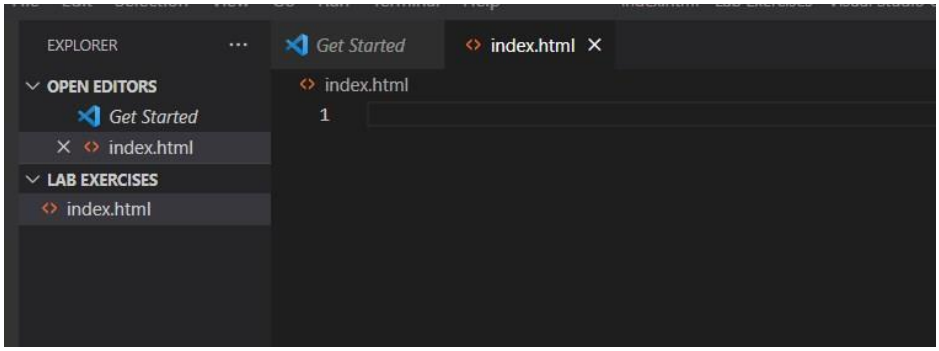
1. Open Visual Studio Code from the start menu on your machine
2. Click File -> Open Folder and select a directory on your machine where you want to store your lab exercises. I've opened a directory called "Lab Exercises". You can see that the directory is open on the left hand side (though it is currently empty)



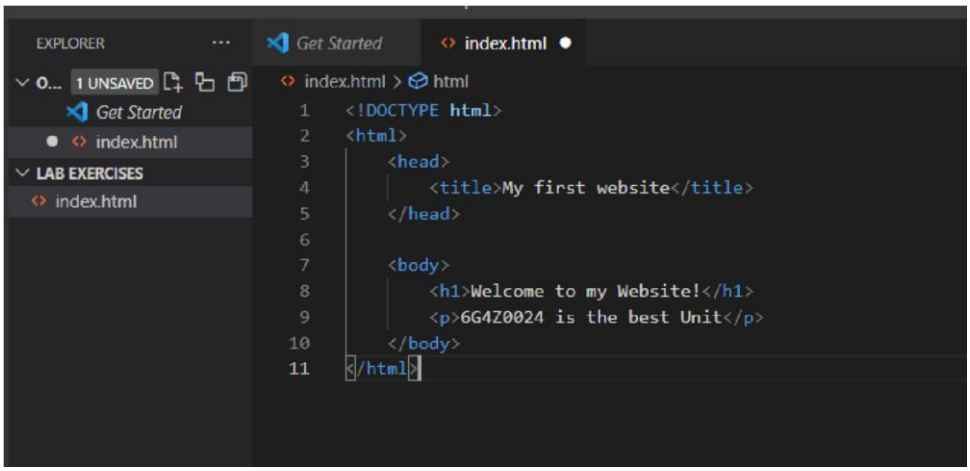
3. As you hover your mouse over the directory name, a series of icons appear. The first icon is for creating a new file – click it



4. Call your new file "index.html". When you hit enter, the file will open in the main window.
   Note: the "index" file is always the homepage for our website



5. Now we have our file, but we need to add some content. HTML pages all have the same structure:
    a. The <!DOCTYPE html> declaration defines that this document is a HTML5 document (HTML5 is the latest version of the HTML standard)
    b. The <html> element is the root element of a HTML page
    c. The <head> element contains meta information about the HTML page, such as the <title> of the page
    d. The <body> element defines the actual content of the page (i.e., what will be displayed to the browser)
    e. The <h1> element defines a large heading and the <p> element defines a paragraph Copy the HTML structure below into your file.



6. Save your page (hit the “ctrl” and “s” keys at the same time)
7. Now, navigate to the directory where your index file is stored. Right click on the file and select “Open with”. From the list, select Google Chrome. Your HTML page will open in a new browser window.

8. Note the <title> in the tab at the top of the page. Also, see how the header is a lot larger than the paragraph and has been bolded. Different HTML elements display in different ways.
9. Congratulations! You have just created your first website (and I'm honoured that your first ever website was to tell people how much you love Web Development).

5.2 Exercise 2: Adding more content to your HTML

Now that you have your basic structure in place, the rest of HTML is simply learning how different elements can help you with your development. For example, let's make some changes to our HTML page.

1. Navigate back to your VS Code window.
2. Add another paragraph element underneath your existing paragraph element. Inside your new element, add your name and an interesting fact about yourself.
3. Save your changes
4. Navigate back to your Chrome window and refresh your webpage. You should be able to see your new changes.

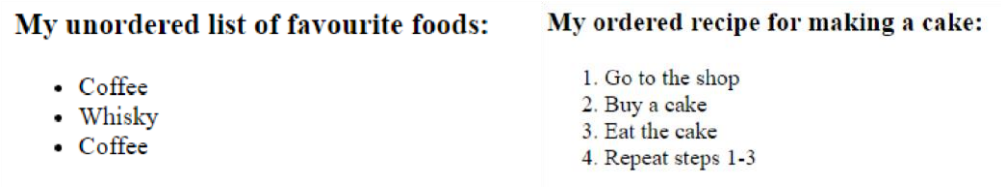
We have now made our first edit and have seen the changes updated graphically on the webpage. What else can we do? Well, we know that the <h1> tag is for a major heading, can we have a <h2> tag? What about <h3>? How far does it go?

1. Add multiple heading elements to your web page. How many different types of headers are there before they all just start looking like the paragraph text?
2. Make sure to save any changes and then refresh your browser window.

5.3 Exercise 3: HTML lists

We can add lists to our web pages. Typically, there are two types of lists: ordered and unordered.

Ordered lists are numbered and unordered lists are bulleted.



The HTML for both types of lists looks similar. Unordered lists are wrapped in a <ul> tag with each item in the list being wrapped in an <li> tag (li stands for list item). Ordered lists are the same but use an <ol> tag instead.

```
<h3>My unordered list of favourite foods:</h3>
<ul>
    <li>Coffee</li>
    <li>Whisky</li>
    <li>Coffee</li>
</ul>


<h3>My ordered recipe for making a cake:</h3>
<ol>
    <li>Go to the shop</li>
    <li>Buy a cake</li>
    <li>Eat the cake</li>
    <li>Repeat steps 1-3</li>
</ol>
```

1. Experiment by adding some lists to your web page

5.4 Exercise 4: HTML formatting

There are multiple ways to format text in HTML. Let's try a few of them out.

1. Create a new paragraph that contains a single sentence. I've chosen song lyrics:

```
<p>A-well, a bird, bird, bird, bird is a word</p>
```

2. To make a portion of the text bold, we will wrap the relevant portion of text in the tag:

```
<p>A-well, a bird, bird, bird, <strong>bird is a word</strong></p>
```

Save the page and refresh the chrome window:

A-well, a bird, bird, bird, **bird is a word**

3. 4. Now try the other tags below. What do each of them do?

HTML Element	What does it do?

<i>	makes the text italics
<small>	reduce font size
<s>	puts a line through the text
<u>	underlines the text
<sub>	moves the text below the line
<sup>	displays text above line

5.5 Exercise 5: Break lines

Some HTML elements instruct the browser to do something without necessary adding content. For example, imagine that we wanted to add some spacing between two paragraphs. We might want to use a break line (the
 element).

```
<p>I'm a happy paragraph</p>
<br></br>
<p>We need some space</p>
```

I'm a happy paragraph

We need some space

While we can see that this has worked, having an opening and close tag without any content inbetween looks a bit messy. We can clean it up using a self-closing tag:

```
<p>I'm a happy paragraph</p>
<br />
<p>We need some space</p>
```

5.6

5.7

5.8

5.9

I'm a happy paragraph

We need some space

5.10

The HTML renders the same on the web page. However, the HTML in the second example is neater. You should always use self-closing tags where you can. We will see later that images also use selfclosing tags.

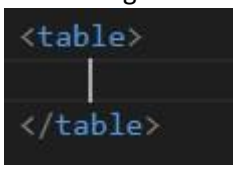
1. Add some break lines into your web page to space elements out a bit more

5.11 Exercise 6: HTML Tables

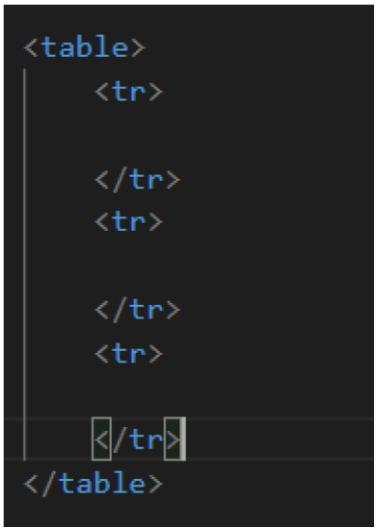
So, HTML is pretty easy right? Once you understand the concepts and basics, you can create basic websites quickly. In fact, you have probably spent more time this session reading than actually coding. That will change.

Tables are a little more complicated than what we have seen so far. They consist of multiple nested elements, but let's break it down:

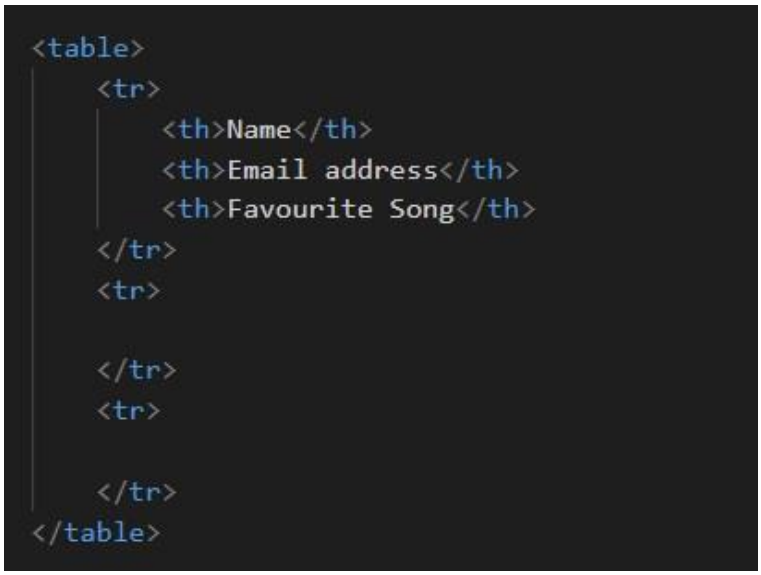
1. First create an empty table with the <table> tags. Nothing will appear in your browser yet.



2. For each row in the table, we want to add a <tr> tag (tr for table row). In this table, we're going to have three rows. Still, nothing shows in Chrome



3. In the first row, we want our column headers. These can be added with the <th> tag (th for table header). The headers now appear in Chrome when you save and refresh.



4. The rest of the rows contain our data. These can be added with the <td> tags (td for table data)

```
<table>
    <tr>
        <th>Name</th>
        <th>Email address</th>
        <th>Favourite Song</th>
    </tr>
    <tr>
        <td>A</td>
        <td>a@mmu.ac.uk</td>
        <td>Hot Legs</td>
    </tr>
    <tr>

    </tr>
</table>
```

5. It doesn't format very nicely in the browser, but we will fix that with styling in later weeks

6. In the last row, add your details. Save and view in your browser window.

As with most elements in HTML, there are more advanced features that we could add. With tables, we can add captions, as well as grouping the headers, body, and footer into their own separate elements. However, basic tables are enough for now.

6 The elephant in the room

You will have noticed by now that your websites are not publicly available on the internet. Usually, developers will create their websites locally (as we are doing here). Then when ready, they will push the website up to a hosted server on the internet. We will talk about how to do this in a later lecture.

7 HTML Attributes and linking between multiple pages

So far in the lab, we have seen and used a lot of different HTML elements. All HTML elements can also contain HTML attributes. Attributes are important as they provide additional information about elements. Attributes usually come in name/value pairs.

For example:

```
<p id="welcome-intro">
    Hi, I am a lecturer of Software Engir
</p>
```

Here, our paragraph tag has been provided with an "id" attribute. I can use this ID elsewhere to identify this specific paragraph from others (e.g., for applying styles to just one paragraph and not all). The attribute isn't visible in the browser.

Some elements require certain attributes in order to be useful.

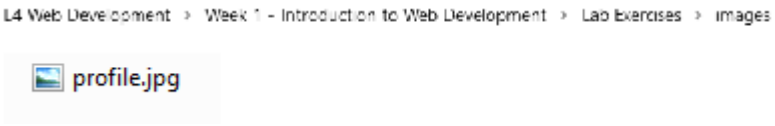
7.1 Exercise 7: Adding images to your web page

For adding images, we use the self-closing <img /> tag. However, we also need to provide it with some attributes in order to work:

- The “src” (source) attribute gives a path to the image to display
- The “alt” (alternative text) attribute provides some text to display in the event that the image can’t load

Let’s give it a go:

1. Find an (appropriate) image yourself online and download it into the directory where your website is stored. I’ve put mine into a dedicated “images” directory



dot before the path to the image instructs the HTML to start from the current directory (e.g., where the HTML is stored)

```
<img src="./images/profile.jpg" alt="A terrible photo of my face" />
```

3. Save and refresh your browser. Can you see the image?

4. Nobody needs to see that massive forehead on their browsers. Let’s make it smaller by setting the width of the image

```
<img src="./images/profile.jpg" alt="A terrible photo of my face" width="150px" />
```

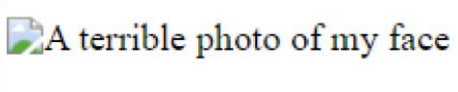
5. Save and refresh your browser. Phew – that’s better.

6. We can simulate the image not loading by changing the file path in the src attribute to a

location that doesn't exist

```
<img src="./images/dog.jpg" alt="A terrible photo of my face" width="150px" />
```

7. Now when we refresh, we can see that the "alt" text is displayed instead

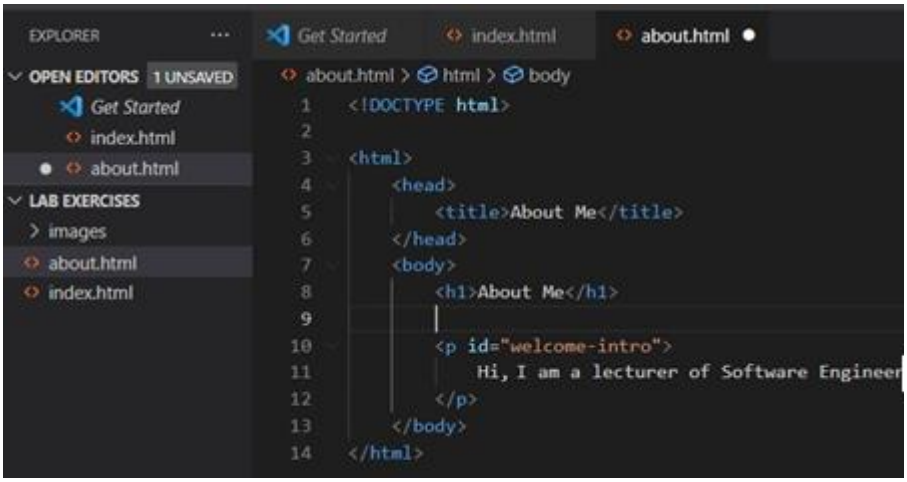


8. As well as downloading images and using file paths, we can also use the src attribute to link to images already on the web.

7.2 Exercise 8: Adding multiple pages to your website

The lab so far has used only a single page. However, one of the main features of the web is that we can have multiple pages and link between them for navigation. Let's have a go at creating a new page:

1. In your directory, create a new file called "about.html"
2. Open your new HTML file, provide it with the basic HTML structure from Exercise 1. However, your header should say "About Me" and your paragraph should contain the short bio that you wrote earlier.



3. Save this new file
4. In your index.html file, add an <a> tag (a stands for anchor). The <a> tag should include a "href" attribute which contains the path for where you want to send the user to.

```
<a href="./about.html">Read more About Me</a>
```

5. Save and refresh. The link should appear in your browser and clicking it should send you to the about page.
6. Oh no! We are now stuck on your about page! Add a link to your about page so that visitors to your site can navigate back to your home page.
7. We can also link to external websites in the same way. Try replacing your href text with the following link (https://www.youtube.com/). Refresh your screen and see where the link now takes you.

8 Validating HTML

We have covered a lot today. I'm sure that you can see how as we add more and more content, and pages, to our website, it can become difficult to manage, maintain, and easy to break. Unlike the programming you have done in previous Units, there is no debugger in HTML to help us.

Instead, we can pass our HTML through a validator to check for errors.

8.1 Exercise 9: Validating your HTML

1. Navigate to https://validator.w3.org/ in your browser
2. Select the "Validate by Direct Input" tab
3. Copy and paste the HTML from your homepage into the textbox provided.
4. Click "Check" – do you have any errors? What suggestions for improving your HTML are provided?
5. Try breaking your HTML by removing the closing tag from one of your elements. What error do you get when you check the page again?

9 You should always make sure that all your HTML pages are valid. Make sure that you also validate your new About page.

10 Resources

By now you should be realising that the HTML elements you learn and use largely depend on what you are creating. No course on HTML will expect you to learn all elements. Instead, you will just learn the most important overtime as you practice writing web pages. For everything, you need to know how to research new elements online.

There are lots of resources for learning web development online. Perhaps one of the best reference sites for beginners is W3Schools.

10.1 Exercise 10: Adding comments

1. Add some code comments to your HTML so that you and other developers will understand the HTML when reading later. Use the W3Schools site to help you. (https://www.w3schools.com/html).
2. <!-- comments go here -->

11 Bringing it all together

11.1 Exercise 11: Build a website

Using all your new HTML skills, create a website version of your CV. Your website needs the following:

- An index page that includes your name, a picture of yourself, and a short paragraph that provides a brief bio.
- An education page that contains a list of educational institutes that you have attended. Each institute should be accompanied with a HTML table of grades
- An experience page that describes your professional experience. The experience should be displayed in a HTML list
- All three pages should be linked to from each other so that users can navigated around your website.
- All three pages should be valid HTML

Make sure that you save this website. We will be using it in later weeks.