## **HTML Boilerplate**

### **Introduction**

All HTML documents have the same basic structure or boilerplate that needs to be in place before anything useful can be done. In this lesson, we will explore the different parts of this boilerplate and see how it all fits together.

### **Learning Outcomes**

* How to write the basic boilerplate for an HTML document
* How to open HTML documents in your browser

### **Creating a HTML File**

To demonstrate HTML boilerplate, we first need an HTML file to work with.

Create a new folder on your computer and name it html-boilerplate. Within that folder create a new file and name it index.html.

You’re probably already familiar with a lot of different types of files, for example doc, pdf, and image files.

To let the computer know we want to create an HTML file, we need to append the filename with the .html extension as we have done when creating the index.html file.

It is worth noting that we named our HTML file index. We should always name the HTML file that will contain the homepage of our website's index.html. This is because web servers will by default look for an index.html page when users land on our websites - and not having one will cause big problems.

### **The DOCTYPE**

Every HTML page starts with a doctype declaration. The doctype’s purpose is to tell the browser what version of HTML it should use to render the document. The latest version of HTML is HTML5, and the doctype for that version is simply <!DOCTYPE html>.

The doctypes for older versions of HTML were a bit more complicated. For example, this is the doctype declaration for HTML4:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

However, we probably won’t ever want to be using an older version of HTML, and so we’ll always use <!DOCTYPE html>.

Open the index.html file created earlier in your text editor and add <!DOCTYPE html> to the very first line.

### **HTML Element**

After we declare the doctype, we need to provide an <html> element. This is what’s known as the root element of the document, meaning that every other element in the document will be a descendant of it.

This becomes more important later on when we learn about manipulating HTML with JavaScript. For now, just know that the HTML element should be included on every HTML document.

Back in the index.html lets add the <html> element by typing out its opening and closing tags, like so:

<!DOCTYPE html> <html> </html>

### **Head Element**

The <head> element is where we put important meta-information about our web pages, and stuff required for our web pages to render correctly in the browser. Anything included within the head element will not be displayed to the user.

#### **Title Element**

One element we should always include in the head of an HTML document is the title element:

<title>My First Webpage</title>

The title element is used to give webpages a human-readable title which is displayed in our webpage’s browser tab.

If we didn’t include a title element, the webpage’s title would default to its file name. In our case that would be index.html, which isn’t very meaningful for users; this would make it very difficult to find our webpage if the user has many browser tabs open.

#### **The Charset Meta Element**

Another important element we should always have in the head element is the meta tag for the charset encoding of the webpage: <meta charset="utf-8"/>.

Setting the encoding is very important because it ensures that the webpage will display special symbols and characters from different languages correctly in the browser.

There are many more elements that can go within the head of an HTML document. However, for now it’s only crucial to know about the two elements we have covered here. We will introduce more elements that go into the head as and when we need them throughout the rest of the curriculum.

Back in our index.html, let’s add a head element with a title and a charset meta element within it. The head element goes within the HTML element and should always be the first element under the opening <html> tag:

<!DOCTYPE html> <html> <head> <title>My First Webpage</title> <meta charset="UTF-8"> </head> </html>

### **Body Element**

The final element needed to complete the HTML boilerplate is the <body> element. This is where all the content that will be displayed to users will go - the text, images, lists, links, and so on.

To complete the boilerplate, add a body element to the index.html file. The body element also goes within the HTML element and is always below the head element, like so:

<!DOCTYPE html> <html> <head> <title>My First Webpage</title> <meta charset="UTF-8"> </head> <body> </body> </html>

### **Viewing HTML Files in the Browser**

The HTML boilerplate in the index.html file is complete at this point, but how do you view it in the browser? There are a couple of different options:

1. You can drag and drop an HTML file from your text editor into the address bar of your favorite browser.
2. You can find the HTML file in your file system and then double click it. This will open up the file in the default browser your system uses.

Using one of the methods above, open up the index.html file we have been working on. You’ll notice the screen is blank. This is because we don’t have anything in our body to display.

Back in the index.html file, lets add a heading (more on these later) to the body, and save the file:

<!DOCTYPE html> <html> <head> <title>My First Webpage</title> <meta charset="UTF-8"> </head> <body> <h1>Hello World!</h1> </body> </html>

Now, if you refresh the page in the browser, you should see the changes take effect, and the heading “Hello World!” will be displayed.

### **VSCode Shortcut**

VSCode has a built-in shortcut you can use for generating all the boilerplate in one go. To trigger the shortcut, delete everything in the index.html file and just enter ! on the first line. This will bring up a couple of options. Press the enter key to choose the first one, and voila, you should have all the boilerplate populated for you.

But it’s still good to know how to write the boilerplate yourself in case you find yourself using a text editor like notepad (heaven forbid) which doesn’t have this shortcut. Try not to use the shortcut in your first few HTML projects, so you can build some muscle memory of how to write the boilerplate code.

### **Assignment**

1. Watch and follow along to Kevin Powell’s Brilliant [Building Your First Web Page video](https://www.youtube.com/watch?v=V8UAEoOvqFg&list=PL4-IK0AVhVjM0xE0K2uZRvsM7LkIhsPT-&index=2)
2. Build some muscle memory by deleting the contents of the index.html file and trying to write out all the boilerplate again from memory. Don’t worry if you have to peek at the lesson content the first few times if you get stuck. Just keep going until you can do it a couple of times from memory.
3. Run your boilerplate through an [HTML validator](https://validator.w3.org/). Validators ensure your markup is correct and are an excellent learning tool, as they provide feedback on syntax errors you may be making often and aren’t aware of, such as missing closing tags and extra spaces in your HTML.

### **Additional Resources**

This section contains helpful links to other content. It isn’t required, so consider it supplemental.

* Read through this article about what [charsets you should use with your HTML pages](https://www.bitdegree.org/learn/html-encoding)
* Another option for opening your HTML pages in the browser is using the [live server extension](https://marketplace.visualstudio.com/items?itemName=ritwickdey.LiveServer) with VSCode. This will open your HTML document and automatically refresh it every time you save the document. However, we recommend not using this extension and instead doing it the old fashioned way, by opening the page and refreshing the page manually in the browser for your first few HTML projects. In this way you can get used to that process and won’t become reliant on extensions right away.

### **Knowledge Check**

This section contains questions for you to check your understanding of this lesson. If you’re having trouble answering the questions below on your own, review the material above to find the answer.

* What is the purpose of the doctype declaration?
* What is the HTML element?
* What is the purpose of the head element?
* What is the purpose of the body element?