Hyperiondev

TASK

HTML Overview

Visit our website

Introduction

Welcome to The 'HTML Overview' Task!

HTML is a markup language that all software developers need to have at least a basic understanding of. In this task, you will learn why HTML is important for software engineers. Additionally, you will learn to use HTML to create a basic, static webpage.

Get in touch

Connect for support

Remember that with our courses, you're not alone! You can contact your mentor to get support on any aspect of your course.

The best way to get help is to login to **www.hyperiondev.com/portal** to start a chat with your mentor. You can also schedule a call or get support via email.

Your mentor is happy to offer you support that is tailored to your individual career or education needs. Do not hesitate to ask a question or for additional support!

A note from the
Hyperion Team

Check out this infographic that compares front-end and back-end development.

WHY HTML?

HTML is used for front-end development. Front-end development focuses on providing the user with a good user experience. As a software engineer, your primary job is likely to be working on the back-end of an application. You will write the code that will run on the server and make everything work. You will design and code algorithms to solve problems. So why are you learning about HTML? For several reasons. First, you will have to work with front-end developers to create applications that aren't just technically great, but that users find easy and enjoyable to use. Therefore, it is good to have at least a basic understanding of the tools used by front-end developers. Also, your back-end code may have to use HTML tags to send data to the front-end. Besides, HTML is used in many popular development frameworks such as JavaServer Faces (JSF), Grails, Django, Ruby and Express. Given the preceding, all software engineers must have, and be able to demonstrate, at least a basic understanding of HTML.

INTRODUCTION TO HTML

HTML stands for Hypertext Markup Language. It is a language that we use to write files that tell the browser how to lay out the text and images on a page. We use HTML *tags* to define how the page must be structured.

HTML Tags

HTML ***tags*** are placed on the left and the right of the element you want to markup, to wrap around the element.

For example:

```
<opening tag>Some text here.</closing tag>
```

This is the general pattern that we follow for all tags in HTML. There are a few exceptions, which we will discuss later. The words 'opening tag' and 'closing tag' are just placeholders we use to illustrate the pattern. Instead of those words, we are going to use special keywords, or elements, that modify the appearance of our webpage.

Note that the opening and closing tags are not the same. The opening tag consists of an opening angled bracket, <, the name of the element, and a closing angled bracket, >. The closing tag consists of an opening angled bracket, <, a forward slash, /, then the name of the tag, and finally the closing angled bracket, >.

```
<!DOCTYPE html>

<html>
<head>
	<title>My first web page!</title>
</head>

<body>
	<p>I am learning to develop a dynamic web application.</p>
</body>
</html>
```

Example of HTML in a simple text file.

The HTML tags indicate to the browser what sort of structure the content is contained in. Note that HTML does not include the *style* of the content (e.g. font, colour, size, etc.), which is done using CSS (Cascading Style Sheets), but only the structure and content itself.

HTML Elements

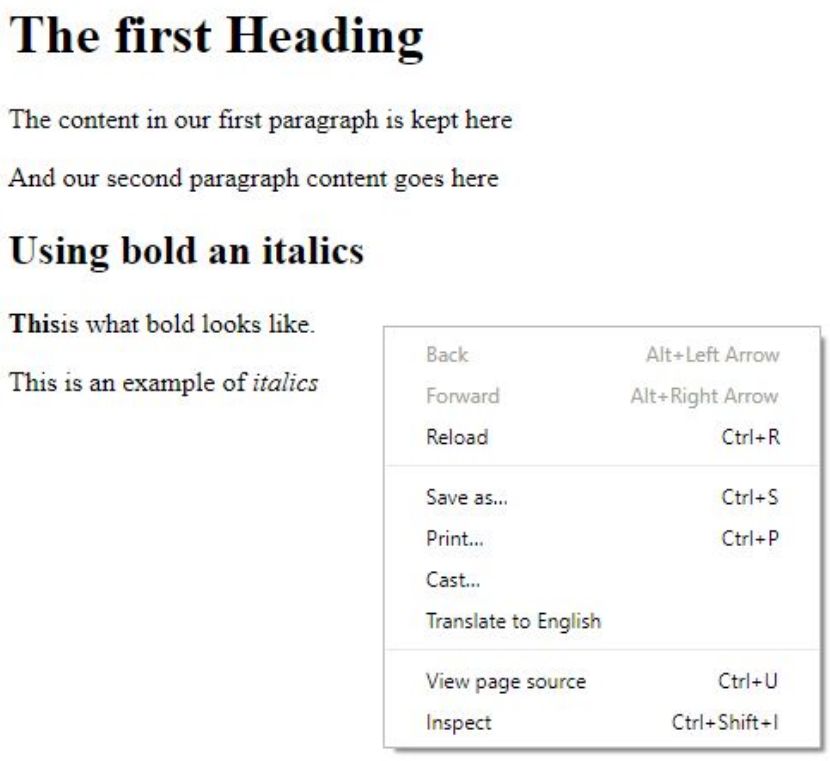
An element usually consists of an opening tag (<element_name>), a closing tag (</element_name>), which contain the element's name surrounded by angle brackets, and the content in between:
<element_name>...content...</element_name>

Example of HTML element:

```
<p>This element is going to result in this paragraph of text being displayed in the browser</p>
```

Try this:

- Double click on the file called 'example.html' (in the same Dropbox folder as this task) to open it in the browser.
- Examine how the HTML page renders in the browser.
- Now, right-click in the browser and select the option 'View page source.'



- You will see the HTML used to create this webpage that includes many HTML tags. For example, you will notice the tags shown below:

```
<h2> Using bold an italics </h2>

    <p> <b>This</b> is what bold looks like.</p> <!--This is also a tag and needs to be closed as shown here-->
    <p> This is an example of <em>italics</em> </p> <!--em stands for emphasis, and thus makes it in italics -->
```

  When the browser encounters the tag <h2> it knows to treat the information between the opening <h2> tag and the closing </h2> tag as a heading. Similarly, the browser will display the information between the tags and as a paragraph of text.
- You will learn more about specific HTML elements soon.

BASIC LAYOUT / TEMPLATE OF AN HTML PAGE

A typical HTML document consists of a **doctype** which indicates which version of HTML to load, a **head** which contains metadata about the page and a **body** which contains the actual content.

A general layout template that you can use before even starting to worry about what sort of content you want to display is set out below:

```
<!DOCTYPE html>

<html>

    <head>


    </head>


    <body>

    </body>

</html>
```

The Doctype is indicated at the top of the page and when typing 'html' it defaults it to HTML5. This is one of the only elements that does not need a closing tag. Note that throughout HTML, capitalisation is very important.

Next, we define what the content is to follow within the <html> tags (note the closing tag at the bottom). Within this <html> element, we introduce two other elements, namely <head> and <body>. Notice that although each of the tags is located on a separate line, we still have **opening tags** matching their **corresponding closing tags**. Notice how the <html> tag wraps around its contents. We use a nested order to structure tags on our web page; this means that tags are contained within other tags, which may themselves contain more tags. Above, the <html> tag *contains* the <head> and <body> tags. It is important to understand how elements are nested because one of the **most frequent mistakes that students make with HTML is getting the order all mixed up**. For example, it would be wrong to have a closing body tag (</body>) after a closing html tag (</html>) because the body element should be completely contained or nested within the <html> element. It should also be noted that white space is ignored by the browser, so you can lay out the physical spacing of the elements as you please.

ATTRIBUTES

Attributes are things that describe the objects created by HTML elements. For example, * This element is going to result in this paragraph of text being displayed in the browser *
would result in a paragraph that contains text. This paragraph can be described using various attributes including align, font-size etc.

Consider the following:

```
<title id="myTitle">My first web page</title>
```

In this case, the element is of type 'title'. Next, we have an 'id' which is an attribute of the element (title), and has a value of "myTitle". Attributes like this are used mainly for CSS and JS (JavaScript) and will be covered later. Then there is a closing '>' which indicates that you have finished defining the attributes of the element.

COMMON HTML ELEMENTS

We have already encountered some commonly used elements that are used to create most web pages. Some of these (and some new elements) are summarised below:

- A piece of metadata that should be included in all web page is the **<title>** element.
  The <title> element:
  - defines a title in the browser tab
  - provides a title for the page when it is added to favorites
  - displays a title for the page in search engine results

  As noted before, metadata should be contained in the <head> of the HTML document.
  E.g. of a title element:
  <head>
  <title>Portfolio</title>
  </head>

- As you would with a word document, use **headings** to show the structure of your web page. This is important because search engines use the headings to index the structure and content of your web pages. There are 6 heading elements you can use: <h1> to <h6> where <h1> element is used for the most important headings and <h6> for the least important.
  E.g. of a heading element:
  <h1>Online Portfolio of work</h1>
  <h2>About me</h2>

- Add paragraphs of text using the ** ** element as follows:
  * This is an example of a paragraph. Paragraphs usually contain more text than headings and are not used by search engines to structure the content of your web page. *

- **Line breaks.** To do the equivalent of pressing enter to get a line break between text, use the
 element. This element does not have a matching closing tag. This should make sense because there is no content that you could put within a
 element. Elements like this, with no content or matching closing tags, are known as void elements.

- **Horizontal rule.** This is another void element. By adding the HTML element <hr> to your web page you will create a horizontal rule.

- **Lists.** Lists can either be ordered lists <ol> or unordered lists <ul>. Ordered simply means that the list is numbered, i.e. 1, 2, 3, etc. and unordered is in the form of bullet points. In lists keeping track of how far you are with nesting of the various elements is VERY important. We highly recommend that you use indentations to keep track of what elements fall under what other elements. Remember that indentation and "whitespace" does not affect the layout of the elements on the web page.

  Unordered Lists

  In an unordered list, as with most elements, we have to open and close the tags. Within this element, we now want to display some content in our list. This content is inputted in the form of *list items* and thus have the tag <li>. So, to create an unordered list with three items in it, we would write it out as follows:

```
<ul>
    <li> Item 1 </li>
    <li> Item 2 </li>
    <li> Item 3 </li>
</ul>
```

  Note how the indentation makes the entire structure a lot easier to read. The list items, as seen above, are also closed at the end of the content to indicate to the browser where the content of that specific item ends.

  Ordered Lists

  Ordered lists work almost the same as unordered lists, except that you use the tag, <ol>. You input list items in the same way as shown above. Instead of showing bullet points, these list items are numbered.

- **Tables.** Tables work similarly to lists in terms of nesting elements. First, define the fact that it's a table using the <table> tag, and then manually enter the data into the rows. Have a look at the example below:

```
<table>
    <tr>
        <td>Row 1, cell 1</td>
        <td>Row 1, cell 2</td>
        <td>Row 1, cell 3</td>
    </tr>
    <tr>
        <td>Row 2, cell 1</td>
        <td>Row 2, cell 2</td>
        <td>Row 2, cell 3</td>
    </tr>
    <tr>
        <td>Row 3, cell 1</td>
        <td>Row 3, cell 2</td>
        <td>Row 3, cell 3</td>
    </tr>
    <tr>
        <td>Row 4, cell 1</td>
        <td>Row 4, cell 2</td>
        <td>Row 4, cell 3</td>
    </tr>
</table>
```

The table element is defined within the opening and closing tags. Immediately within these tags, there is a *table row* indicated by <tr> which also has a closing tag. Within that first table row, there is a <td> tag which indicates that there is *table data*. A table is shown in the "example2.html" file so that you can try and correlate what elements contribute to what visual appearance on the web page.

The most important elements for this course can be found in "example.html" and "example2.html".

HTML SYNTAX

As a web developer, you are going to learn many new languages. Each of these has their own rules which must be strictly followed for your instructions to be properly processed. The rules of a language are referred to as *syntax*. Examples of common HTML syntax errors include spelling the name of an element incorrectly or not closing tags properly or in the wrong order. You are bound to make mistakes that will violate these rules and that will cause problems when you try to view web pages in the browser! We all make syntax errors! Often! Being able to identify and correct these errors becomes easier with time and is an extremely important skill to develop.

To help you identify HTML syntax errors, copy and paste the HTML you want to check into this helpful tool.

LINKS

You can add links to your web page as follows:

link text

The <a> element is used to add all links on a web page. Using this element you can link to other pages in your website, to external web pages and to enable users to send an email.

Linking to other places on your web page

Often on your web page, you will want your users to be able to click on a link which will then take them to another part of the same page. Think about the "back to the top" button - you click on this and you are suddenly viewing the top of the page again!

To do this, we need to use *ID* attributes. An ID is used to uniquely identify one of your HTML elements, such as a paragraph, heading or table. Then we can use the link tag to make the text or image a link that the user clicks on to take them to whichever address we choose!

An ID can be assigned to any of your elements, and is done as follows:

```
<h1 id="theHeading">My first web page</h1>
```

Notice how the attribute "id" is within the opening tag.

Now that we have this heading, we can look at how to reference it within our text. We use the <a> tag which shows which address we are using. To reference a structure with an ID, we need to precede the value assigned to the id attribute with a "#" otherwise, the browser will think you are looking for a website.

```
<h1 id="theHeading">My first web page</h1>

<a href="#theHeading">Back to top</a>
```

Consider the "example.html file" that contains the elements shown above. If you open it you will see that it will make the text "Back to top" look like a hyperlink (blue and underlined). When this is clicked, it will take you to the Heading with the id "theHeading".

Linking to other web pages

Similarly, we can link to another page. This is done as follows:

```
<a href="http://www.hyperiondev.com">This cool place!</a>
```

The “http://” in front of the address lets the browser know that you are linking to an external website rather than a file on your system.

However, you aren’t limited to creating links through text! All the content that is between the <a> tags is what is to be clicked on to get to the destination address.

With the link specified above, if you click on the link it will change the window you’re currently on. What if you wanted to open the destination address of a link in a new tab? You can add an attribute to the link tag called *target* which specifies how the link should be opened, e.g. in the same window, new browser instance or new tab. To open in a new tab, simply modify the link as follows:

```
<a target = "_blank" href="http://www.hyperiondev.com"/>
    This cool place!
</a>
```

IMAGES

We add images to our website using the <img> element as shown below:

```
<img src="http://hyperiondev.com/static/moocadmin/assets/img/hyperionLogo.png">
<img src="images/image1.jpg">
```

There are a few things to note about the <img> element.

- Unlike most of the other elements we have explored so far, the <img> element doesn’t have a closing tag.
- The <img> element has several attributes that can describe it. These include:
    - src= The src attribute gives the path to where the image can be found or the *source* of the image.
    - alt= The alt attribute defines the *alternate text* that will be displayed if the image won't display.
    - Intuitively, the height and width attributes define the height and width of the image.
- The src attribute can point to a URL or a file location. In the example above, the first image uses a URL as the source of an image. The second image shows how the src attribute is defined to display an image named *image1.jpg* that is stored in a folder named *images* that resides in the same folder as your web page.

A note from our coding mentor

Sabir

When adding images to your web page, it is important to remember that this page may be viewed on many different devices with widely differing screen sizes, resolutions, etc. You want the images to look good independent of the device that is used to view the page. Thus responsive images (images that work well on devices with widely differing screen sizes, resolutions) are important. To see how to create responsive images, see here and Chapter 15 of "HTML5 notes for professionals" (additional reading in the Dropbox folder of the first task).

HTML FORMS

A dynamic website is driven by user interaction. For users to be able to interact with your website, you need to provide them with the means to enter the information that is to be used and displayed on the pages. Forms are the instruments that we use to allow users to enter data in HTML. Forms can be structured in various ways; in fact, web designers often try to make them as cool as possible to encourage users to interact with the site. Here are some examples of different kinds of forms on the Web:

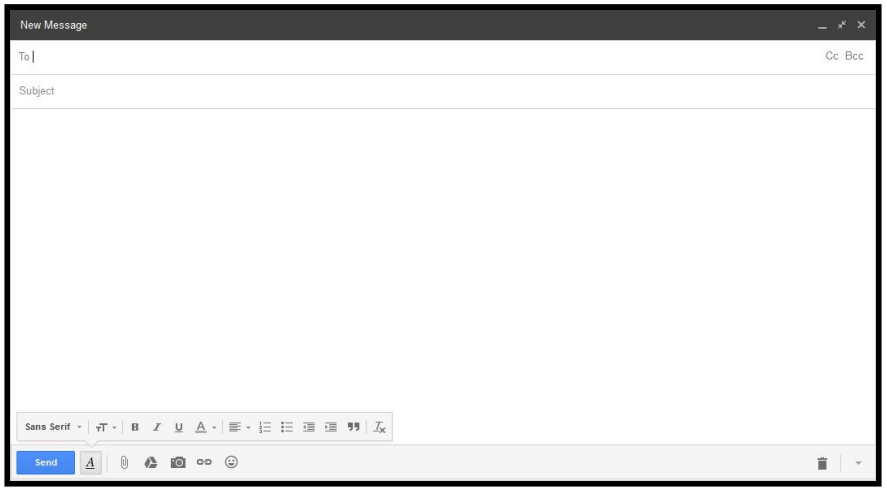


Figure 1: Here's another sophisticated form from gmail (mail.google.com) - this is the popup text editor used to draft an email.

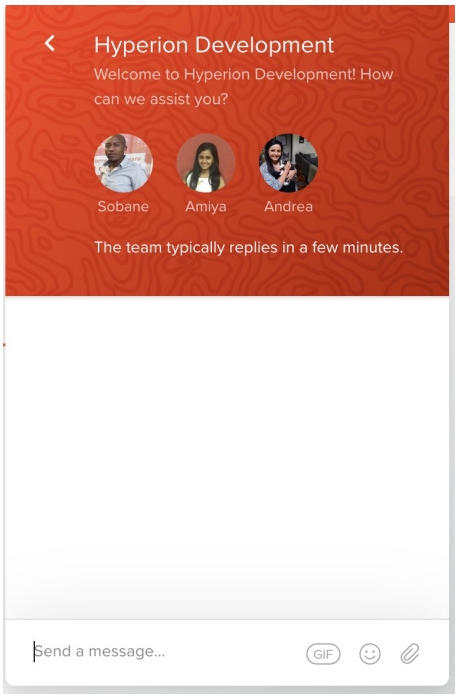


Figure 2: Hyperion Development's (www.hyperiondev.com) chat box is a very sophisticated form, but a form nonetheless

CREATING A FORM

We won't begin with complex forms like the ones you see above. First, we're going to build a simple form and focus on investigating some of the components of a form. At this stage, our forms won't be functional. You will create forms that function correctly later when you learn JavaScript.

```
<form>
    <label>First Name:</label>
    <input type="text"><br>
    <label>Surname:</label>
    <input type="text"><br>
    <label>Gender:</label>
    <select>
        <option value="male">Male</option>
        <option value="female">Female</option>
    </select>
    <label>Age:</label>
    <input type="text">
</form>
```

In the example above we create a form to capture our user's biographical information. It captures the following information:

- First Name

- Surname
- Gender
- Age

We expect the user to enter text for their name and surname. We, therefore, use the **input** element. This element has a **type** attribute with the **text** property assigned to it. This displays text boxes in the browser into which users can type input. We add labels to tell our visitors what information we want them to enter into the boxes.

The **select** element is used to create a drop-down menu that users can select from instead of typing out their gender.

To see a list of other HTML input types, see here and Chapter 17 of "HTML5 notes for professionals" (additional reading in the Dropbox folder of the first task).

READABILITY

As you start to create HTML pages with more elements, it becomes increasingly important to make sure that your HTML is easy to read. In software development, readability is an important principle! Code and markup that is easy to read are easier to debug and maintain than code or markup that is not easy to read.

Indenting your HTML is an important way of improving the readability of your code. For example, consider the HTML below:

```
<!DOCTYPE html><html><head>
<title>My first web page</title>
</head>
<body>
<form><label>First Name:</label><input type="text"><br>
<label>Surname:</label><input type="text"><br>
<label>Gender:</label>
<select><option value="male">Male</option>
<option value="female">Female</option></select>
<label>Age:</label>
<input type="text">
</form></body></html>
```

The above is perfectly correct HTML that will render properly in the browser but it is certainly not as easy to read and understand as the code below which is properly indented:

```html
<!DOCTYPE html>
<html>

<head>
    <title>My first web page</title>
    <!-- This is a comment, by the way -->
</head>

<body>
    <form>
        <label>First Name:</label>
        <input type="text"><br>
        <label>Surname:</label>
        <input type="text"><br>
        <label>Gender:</label> <br>
        <select>
            <option value="male">Male</option>
            <option value="female">Female</option>
        </select><br>
        <label>Age:</label><br>
        <input type="text"><br>
        <input type="submit" value="Add user">
    </form>
</body>

</html>
```

As you can see above, indentation should be used to show which HTML elements are nested within other HTML elements. As shown above, all the other elements are nested within the <html> element.

A note from the **Hyperion Team**

Remember that with our courses, you're not alone! To become a competent software developer, it is important to know where to get help when you get stuck.

Here you can find resources that provide extra information about HTML, CSS and JavaScript.

Compulsory Task

In this task, you are going to be creating the content for your personal webpage. Don't worry too much about what the webpage looks like at this stage. You will use CSS to add some style and perfect the layout in the next task. For now, focus on the content of the webpage. Does it contain all the information that you would like that introduces you to the world? Strike a balance in your content — this webpage should show more of your personality than a typical CV but it should still be professional.

Follow these steps:

- Create an HTML page called **index.html**. You can use most code editors to do this, including Eclipse, Sublime, Visual Studio Code, etc.
- On this page, add any elements you would like to create a webpage that acts as an online CV. This is your personal webpage, so feel free to customise it to suit your needs, but make sure that you include at least the following:
    - A short bio. Add a short (no more than about three paragraphs) description of yourself. Who are you? What is your experience? What are you passions? What motivates you? What is it that you would most like to do? Etc.
    - Your contact details. E.g. name, contact number, email address, links to any of your (professional) social media including LinkedIn. An image of yourself.
    - A list of your skills and competencies.
    - Describe your education.
    - Describe your work experience.
    - Describe some projects that you have worked on. Add links to the code (in GitHub).
    - Add links to any blog posts or articles that you have written.

Once you have completed the task in line with the instructions above, click the button below to request your mentor to review your work and provide feedback. If you have any questions while attempting the task, leave your mentor a note on the comments.txt file in your Student Dropbox folder.

Completed the task(s)?

Ask your mentor review your work!

Review work

Rate us

Share your thoughts

Hyperion strives to provide internationally-excellent course content that helps you achieve your learning outcomes.

Think the content of this task, or this course as a whole, can be improved or think we've done a good job?

Click here to share your thoughts anonymously.