TASK

HTML and Semantic HTML

Visit our website

Introduction

WELCOME TO THE HTML AND SEMANTIC HTML TASK!

HTML is a markup language with which all front-end developers need to be comfortable. In this task, you will learn how to use HTML to create a basic static web page.

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 handled using CSS (Cascading Style Sheets), but only the structure and content itself.

HTML Elements

An element usually consists of an opening tag (`<element_name>`) and a closing tag (`</element_name>`), each containing the element's name surrounded by angled brackets, and the content in between these, as follows:

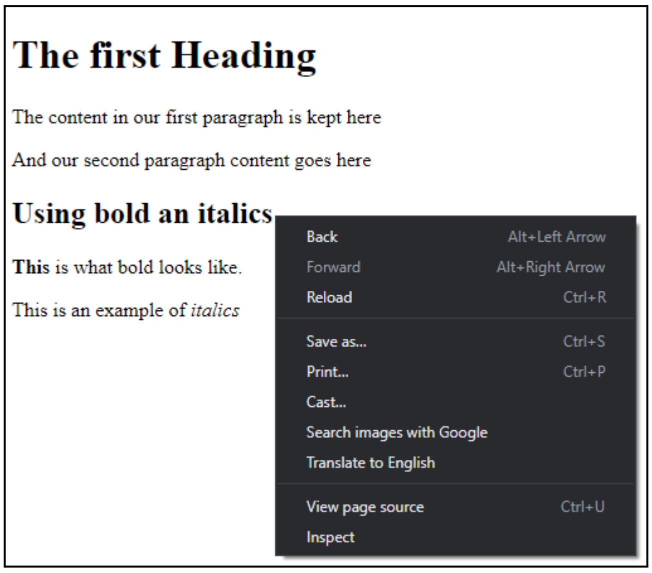
```
<element_name>...content...</element_name>
```

Example of the paragraph (**p**) HTML element:

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

Try this:

- Open the **example.html** file in your browser.
- Examine how the HTML page renders in the browser.
- Now, right-click in the browser and select the option 'View page source.'



The first Heading

The content in our first paragraph is kept here

And our second paragraph content goes here

Using bold an italics

This is what bold looks like.

This is an example of *italics*

- You will see the HTML used to create this webpage, which 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 `<p>` and `</p>` as a paragraph of text.

BASIC LAYOUT OF AN HTML PAGE

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:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>

</body>
</html>
```

A typical HTML document, as shown above, consists of the following elements:

- **`doctype`:** This indicates which version of HTML to load. The doctype is indicated at the top of the page and when typing 'html' it defaults to HTML5. This is one of the only elements that does not need a closing tag. Note that throughout HTML, capitalisation is very important.
- **`html`:** Next, we define what 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.
- **`head`:** This contains metadata about the page.
- **`body`:** This contains the actual content.

Note: 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.

Code Hack

Hack to prevent unexpected errors:
In VS Code, open a blank HTML file and type an exclamation mark "**!**" at the beginning of the document. Then, hit "Enter" and you will instantly generate the basic HTML skeleton to kickstart your web development.

ATTRIBUTES

Attributes describe the objects created by HTML elements. For example, `<p>This element is going to result in this paragraph of text being displayed in the browser</p>` would result in a paragraph that contains the text. This paragraph can be described using various attributes including alignment and font size.

Now let's look at page titles. Consider the following code:

```
<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 the value "myTitle". Attributes like this are used mainly for CSS and JavaScript (IDs are not compulsory, but they can come in very handy). Then there is a closing bracket `>` which indicates that you have finished defining the attributes of the element.

COMMON HTML ELEMENTS

We have already encountered some of the common 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 pages 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 favourites
    - displays a title for the page in search engine results

As noted before, metadata should be contained in the `<head>` of the HTML document.

Example of a title element:

```
<head>
      <title>Portfolio</title>
</head>
```

- Use **headings** to show the structure of your web page, as you would when creating a document with a word processor, although in HTML they serve an additional function. Headings enable search engines to index the structure and content of your web pages. There are six heading elements you can use, `<h1>` to `<h6>`, where `<h1>` is used for the most important headings and `<h6>` for the least important.

  Example of a heading element:

```
<h1>Online Portfolio of work</h1>
<h2>About me</h2>
```

- Add paragraphs of text using the `<p>` element as follows:

```
<p>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. </p>
```

- **Line breaks.** To do the equivalent of pressing enter to get a line break between text, use the break (`<br>`) element. This element does not have a matching closing tag. This should make sense because there is no content that you could put within the `<br>` 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>`. An ordered list is numbered, i.e. 1, 2, 3, etc., whereas an unordered list uses bullet points. In lists, keeping track of how deeply nested the various elements may be is VERY important. We highly recommend that you use indentations to keep track of which elements fall under which other elements. Remember that indentation and "whitespace" do not affect the layout of the elements on the web page, they just help to make your code more human–readable.

- **Unordered Lists:** An unordered list element includes *list items* and thus has the tag `<li>`. To create an unordered list with three items in it, we could write the following:

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

  Output Example:

  - Item 1
  - Item 2
  - Item 3

- **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 having bullet points, these list items are numbered.

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

  Output Example:

  1. Item 1
  2. Item 2
  3. Item 3

- **Tables.** Tables work similarly to lists in terms of nesting elements. First, define the table element using the `<table>` tag. You can then add a table header `<th>` inside the table row tag. Below the headers, you can enter the data into the rows. Have a look at the example below:

```
<table>
    <tr>
        <th>Header 1</th>
        <th>Header 2</th>
        <th>Header 3</th>
    </tr>
    <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>
</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 **html_example2.html** file provided to you. Have a look at the code and try to correlate which elements code for which visual effects on the web page. You can also try making small changes to the example file to see what effect this has when you refresh the displayed web page.

LINKS

You can add links to your web page as follows:

```
<a href="url" target="_blank">link text</a>
```

The `<a>` element stands for "**a**nchor" and is used to add links on a web page. The `href` attribute stands for "**h**ypertext **ref**erence", i.e. to anchor a link using HTML. Using this element, you can link to other pages within your own website, as well as to external web pages. You can also enable users to send an email by using the `mailto:` attribute within the opening anchor tag, as shown below. Note that the `mailto:` attribute takes an email address as a parameter, and the attribute and its parameter must all be enclosed in double quotes, like this:.

```
<a href="mailto:example@hyperiondev.com"">Contact us</a>
```

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 that 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 identify one of your HTML elements, such as a paragraph, heading, or table. We can then 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, 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/>
```

Open the **links_attributes_images_eg.html** example file. It contains the elements shown above. Notice how it makes 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 web page. This can be achieved 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. However, what if you wanted to open the link's destination address 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. Using `target="_blank"` will open the link in a new tab instead of changing your current tab to the specified web address. This is very useful when you want to keep a user on your website and don't want them to leave every time they follow an embedded link.

To open in a new tab, simply modify the link as follows:

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

IMAGES

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

```
<img src =
"http://hyperiondev.com/static/moocadmin/assets/img/hyperiondevlogo.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=` This attribute gives the path to the location of the image, i.e. the *source* of the image.
    - `alt=` The `alt` attribute defines the *alternate text* that will be displayed if the image won't display. This is also useful to extend the accessibility of your web page, as page-reader tools for the visually impaired will be able to read the alternate text to users. As such, it is good practice to use the `alt` attribute to provide a succinct description of the image.
    - 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 first example above, the first image uses a URL as the source of an image. The second example 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 quick note on the format for relative file paths:

- If `image1.jpg` is in the same folder as the page we're adding it to, we simply write `<img src = "image1.jpg">`

- If `image1.jpg` is located in the images folder at the root of the webpage, we write `<img src = "images/image1.jpg">`

- If `image1.jpg` is located in the folder one level up from the webpage, we write `<img src = "../image1.jpg">`

Take note:

When adding images to your web page, it is important to remember that the 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, creating responsive images (images that work well on devices with widely differing screen sizes and resolutions) are important. To see how to create responsive images, see **here** and consult Chapter 15 of "HTML5 notes for professionals" (the additional reading provided with this task).

FORMS

HTML forms allow users to submit information and interact with websites. We'll start by exploring the key concepts of a simple form. While our form won't be functional at this stage, we'll focus on understanding the fundamental structure and elements of HTML forms.

```
<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>
            <option value = "non-binary" > Non-binary</option>
            <option value = "other" > Other</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 can capture 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 from which users can select an option, instead of typing out their gender.

To see a list of other HTML input types, see **here** and consult Chapter 17 of "HTML5 notes for professionals" (the additional reading provided with this task).

SEMANTIC HTML

The latest and most enhanced version of HTML is called Semantic HTML. Let's have a look at this and find out what Semantic HTML is and how to improve the way you create the markup for all your web pages.

One of the most important features of HTML5 is its semantics. The word semantic means *"relating to meaning"*. HTML was originally designed to semantically describe scientific documents; it has since evolved to describe much more. Semantic HTML refers to writing HTML in a way that is more comprehensible, by better defining the different sections and layout of webpages.

When using Semantic HTML, we choose HTML elements based on their meaning, not on how they are presented. This approach also makes web pages more informative and adaptable, allowing browsers and search engines to interpret content better. Elements such as **`<div>`** and **`<span>`** are not semantic elements because they provide no context as to what is inside those tags.

LIST OF COMMON SEMANTIC HTML ELEMENTS

`<header>` - The **`<header>`** element represents introductory content, typically a group of introductory or navigational aids.

The **`<header>`** element may contain:
- heading elements
- a logo
- a search form
- an author name
- navigational links

`<nav>` - The `<nav>` element represents a section of a web page the purpose of which is to provide navigation links, either within the current webpage or to other web pages.

Example use cases of navigation sections:

- menus
- tables of contents
- indexes

`<main>` - The `<main>` element represents the content of the `<body>` of a webpage. The main content area consists of content that is directly related to, or expands upon, the central topic of a webpage, or the central functionality of an application. Please look at the example (coming up) under the `<header>` element for an example of the `<main>` element.

`<section>` - This represents a generic standalone section of a webpage, which doesn't have a more specific semantic element to represent it. A `<section>` should always have a heading, with very few exceptions.

`<figure>` - This represents self-contained content that is specified using the `<figcaption>` element (things like illustrations, diagrams, photos, and code listings). You can see an example of the `<figure>` element in the example below.

`<figcaption>` -This represents a caption or legend describing the rest of the contents of its parent `<figure>` element.

`<footer>` - This defines a footer for a web page or section. A `<footer>` typically contains information about the author of the section, copyright data, or links to related web pages.

Here is an example of the above semantic elements used to create a webpage:

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport"
      content="width=device-width, initial-scale=1 shrink-to-fit=no"/>
    <title>Home</title>
  </head>

  <body>
    <header>
      <a href="#">
        <img src="../images/logo.png" alt="my cat logo" />
      </a>
      <nav>
        <ul>
          <li><a href="#">Home</a></li>
          <li><a href="#cats">Cute Cat Express</a></li>
        </ul>
      </nav>
    </header>

    <main>
      <section id="cats">
        <h1>Cute Cat Express</h1>
        <p>I love cats <em>so</em> much! Like, really, a lot.</p>

        <figure>
          <img src="../images/cat.jpg" alt="a cat meme template" />
          <figcaption>A cat meme template</figcaption>
        </figure>
      </section>
    </main>

    <footer>Cat Memes &copy; 2023. All Rights Reserved</footer>
  </body>
</html>
```

WHY USE SEMANTIC HTML?

ACCESSIBILITY

Screen readers and browsers can interpret Semantic HTML better, which makes web pages more accessible for people with disabilities.

SEO

Using Semantic HTML can improve website Search Engine Optimisation (SEO). SEO refers to the process of increasing the number of people that visit your webpage. With better SEO, search engines are better able to identify the content of your website and weigh the most important content appropriately.

EASY TO UNDERSTAND

Writing Semantic HTML makes code easier to understand, making the source code more readable for other developers.

A note from the

HyperionDev 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.

Take a look at the following resources that provide extra information about **HTML** and **CSS**.

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. As you know, in software development, readability is an important principle! Code and markup that are easy to read are easier to debug and maintain than code or markup that are difficult 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><br>
<select><option value = "male" > Male</option>
<option value = "female" > Female</option>
<option value = "other" > Other</option>
</select><br>
<label> Age: </label><br>
<input type = "text"><br>
<input type="submit" value ="Add user">
</form></body></html>
```

The above is perfectly correct HTML that will render properly in the browser. However, 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>
                  <option value = "other" > Other</option>
            </select><br>
            <label> Age: </label><br>
            <input type = "text"><br>
            <input type="submit" value ="Add user">
      </form>
</body>
</html>
```

As you can see in the example, the indentation is used to show which HTML elements are nested within which other HTML elements. All the elements on the web page are nested within the `<html>` element which provides the outer structure for the code.

Code Hack

Install the code formatter **Prettier** from the Extension tab in VS Code. This extension will automatically and consistently format your code each time you save the file.

Once this extension is installed, change the settings by typing "default" in the search box and selecting the "Text Editor" folder. Look for the option to set Prettier as the default formatter. Next, type "format" in the search box and navigate to "Text Editor -> Formatting" and check the box "Format on Save".

HTML SYNTAX

As a developer, you are going to learn many new languages. Each of these has its own rules which must be followed strictly for your instructions to be properly processed. As you know, the rules of a language are referred to as *syntax*. Examples of common HTML syntax errors include spelling the name of an element incorrectly, not closing tags properly, or closing tags in the wrong order. You are bound to make mistakes that will violate these rules and 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**.

Extra resource

An **Extra Resource** for this task includes the eBook entitled "**HTML5 notes for professionals**" by the 'beautiful people of Stack Overflow' which is included in this task's Dropbox folder.

Instructions

Read and run the example files provided to you along with this task before doing the Compulsory Task. This will help you to become more comfortable with the concepts covered in the task before you start to apply them in the Compulsory Task.

Compulsory Task

In this task, you will create your own personal webpage to showcase your HTML skills and serve as an online CV/résumé.

Although the webpage's visual design is not the primary focus at this stage of your learning, pay attention to creating well-structured HTML code and presenting the content in a clear and organised manner.

Follow these steps:

- In Visual Studio Code, create a new HTML file called **index.html**.
- Write HTML code to create a basic CV, including the following sections:
  - **Menu**: Include navigational links to direct users to different sections within your CV.
  - **Profile**: Display your name and a profile photo.
  - **Bio**: Write a short description of yourself, including your strengths, motivations, and aspirations.
  - **Skills**: Provide a list of both your soft and hard skills.
  - **Education**: List and describe your educational background.
  - **Work Experience**: List and describe your roles and responsibilities. If you lack work experience, include any relevant volunteer work or completed projects instead. Provide links to the company's website and/or to your projects or portfolio, if applicable.
  - **Contact Details**: Include your full name, contact number, email address, and links to your professional social media accounts.
- Ensure that your code includes the following mandatory elements:
  - `<nav>`: Navigation element for linking to different sections.
  - `<img>`: Image element to display your profile photo.
  - `<a>`: Anchor elements for linking to external websites or social media profiles.
  - `<ol>`, `<ul>`, and `<li>`: Ordered and unordered lists containing list elements for listing skills, education, and work experience.
  - `<hr>`: Horizontal rule element for visual separation.

    - `<h1>` to `<h6>` (any): Heading elements for proper content hierarchy.
- Feel free to customise the page and include additional elements as needed to suit your preferences, requirements, and creativity.

Rate us

Share your thoughts

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

Think that 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.