HyperionDev

HTML Overview

Task

Visit our website

Introduction

It is time to learn to create a website! HyperText Markup Language (HTML) is a markup language that all developers need to be comfortable with for front-end web development. In this task, you will learn how to use HTML to create a basic static web page.

Extra resource

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

HTML

HTML, which stands for HyperText Markup Language, 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 content you want to markup, wrapping around it.

For example:

```
<opening_tag>Some text here.</closing_tag>
```

This is the general pattern that we follow for all elements 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 web page.

Note that the opening and closing tags are not the same. The opening tag consists of an opening angled bracket (<), the name of the tag, 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, (>).

Look at the example code below:

```
<!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>
```

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 (`<tag_name>`) and a closing tag (`</tag_name>`), with each containing the element's name surrounded by angle brackets, and the content in between these, as follows:

`<tag_name>...content...</tag_name>`

Here's another example of an HTML element:

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

Try this:

1. Double-click on the file called **html_example1.html** (in the same folder as this task) to open it in the browser.
2. Examine how the HTML page renders in the browser.
3. Now, right-click in the browser and select the option "View page source".

4. You will see the HTML used to create this web page, which includes many HTML tags. For example, you will notice the tags shown below:

```
<p>The content in our first paragraph is kept here</p>
<p>And our second paragraph content goes here</p>
<h2>Using bold and italics</h2>
```

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.

You will soon learn more about specific HTML elements.

Extra resource

Accompanying this task, you will find an ebook titled *HTML5 Notes for Professionals*, which is a great resource for learning how to use HTML.

HTML code comments

Comments are used to explain and annotate your HTML code. They are not visible in the browser but can be seen in the source code. Use the following syntax for comments:

```
<!-- This is a comment -->
```

Example:

```
<!-- Main heading of the page -->
<h1>Welcome</h1>
```

In this example, the comment indicates that the `<h1>` element serves as the main heading for the page. Comments help make your code clearer and easier to maintain.

Basic layout/template of an HTML page

A typical HTML document consists of a **doctype** that indicates which version of HTML to load, a **head** that contains metadata about the page, and a **body** that 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 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.
- **html**: Next, we define what content is to follow within the <html> tags (note the closing tag at the bottom). The `<html>` tag typically includes a `lang` attribute, which specifies the language of the document, helping search engines and browsers to understand the primary language used on the page.

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. Metadata is information about the data on a page that helps define its content and structure.
- **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.

Attributes

Attributes are things that 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>
```

This would result in a paragraph that contains text. This paragraph can be described using various attributes including alignment and font size.

Consider the following:

```
<title id="my-title">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 `my-title`. Attributes like this are used mainly for CSS and JavaScript (using an `id` is not compulsory, but they can come in very handy, as you will see in later tasks). Then there is a closing bracket (>) that 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:

Titles

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, and
- 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>
```

Headings

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 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>
```

Paragraphs

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 `<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**.

Example of the line break element:

```
<p>
  First line of text.<br>
  Second line of text after a line break.<br>
  Third line of text with another line break.
</p>
```

Horizontal rule

This is another void element. By adding the HTML element `<hr>` to your web page you will create a horizontal line across the width of your web page.

Lists

Lists can either be **unordered lists** `<ul>` or **ordered lists** `<ol>`. An ordered list is numbered, i.e., 1, 2, 3, etc., whereas an unordered list uses bullet points. In lists, keeping track of how far you are with the nesting of the various elements 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.

- **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 input in the form of list items and thus has the tag `<li>`. So, to create an unordered list with three items in it, we would write the following:

  ```
  <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. Each list item, as seen above, has a closing tag at the end of the content to indicate to the browser where the content of that specific item ends. Here's the output:

  - Item 1
  - Item 2
  - Item 3

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

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

Again, here is the output:

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, 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>` that also has a closing tag. Within that first table row, there is a `<td>` tag that indicates there is table data. A table is shown in

the **html_example2.html** file so that you can try to correlate which elements contribute to which visual appearance on the web page.

Table headers: To add a table header we use the `<th>` tag inside the table row tag.

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

The most important elements for this task can be found in **html_example1.html** and **html_example2.html**.

Take note

During your career in tech, you are likely to learn many new coding languages. Each of these has its own rules that must be strictly followed 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 that will cause problems when you try to view web pages in the browser. We all make syntax errors, and 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:

```
<a href="example.com">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 **r**eference, i.e. to anchor a link using HTML. Using this element, you can link to other pages within your website, as well as to external web pages. You can also enable users to send an email by using the mailto: scheme within the `href` attribute of the `<a>` element. For instance, `mailto:` is used within the `href` attribute to create a link that opens the user's default email application with a new email addressed to the specified email address. Both `mailto:` and the email address must be enclosed in double quotes.

```
<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. 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="the-heading">My first web page</h1>
```

Notice how the `id` attribute 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 that 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 = "the-heading">My first web page</h1>
<a href = "#the-heading">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 `"the-heading"`.

Linking to other web pages

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

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

The `https://` 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 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 are currently on. But 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` that specifies how the link should be opened, e.g., in the same window, a new browser instance, or a new tab. Using `target="_blank"` will open the link in a new tab instead of changing our 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.

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

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

Images

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

```
<img
  src="https://www.hyperiondev.com/static/images/couple.png"
  alt="Image of two people sitting cross-legged in front of a laptop."
/>
<img
  src="image1.jpg"
  alt="Image displaying a closing HTML tag with an orange background"
  height="200px"
  width="300px"
/>
```

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 the image. The second image has the `src` attribute defined to display an image named **image1.jpg** that is stored 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 web page, we write: `<img src = "images/image1.jpg">`
- If `image1.jpg` is in the folder one level up from the web page, we write: `<img src = "../image1.jpg">`

Take note

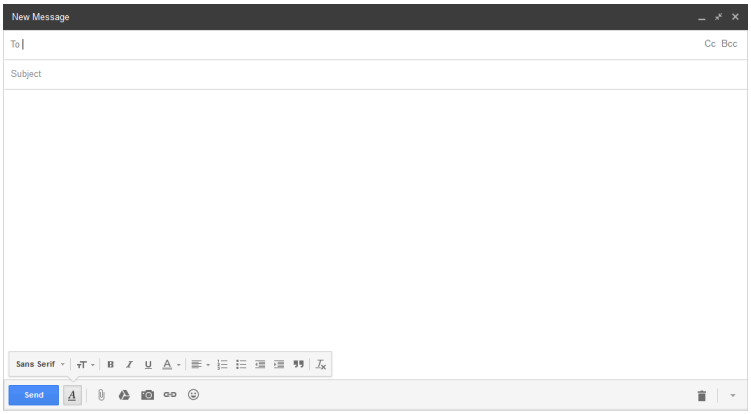
When adding images to your web page, it's essential to consider that the page may be viewed on various devices with different screen sizes and resolutions. To ensure that your images appear well on all devices, it's important to use responsive images—images that adjust to look good regardless of the device's screen size or resolution. For guidance on creating responsive images, click **here**.

HTML forms

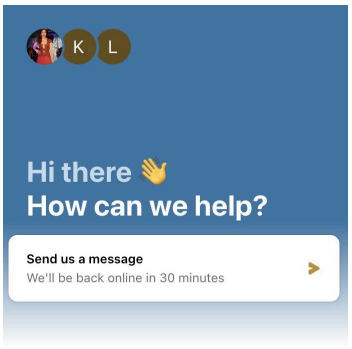
A dynamic website is driven by user interaction. For users to be able to interact with your website, you need to provide them with the means to enter the information that is to be used and displayed on the pages. Forms are the instruments that we use to allow users to enter data in HTML. Forms can be structured in various ways; in fact, web

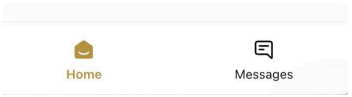
designers often try to make them as cool as possible to encourage users to interact with the site.

Here's a sophisticated form from Gmail (mail.google.com) – this is the popup text editor used to draft an email:



HyperionDev's chat box on **www.hyperiondev.com** is another advanced form.





Creating a form

We will not begin with complex forms like the ones you see above. First, we are going to build a simple form and focus on investigating some of the components of a form. At this stage, our forms will not be functional.

```
<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="other">Other</option>
  </select>
  <label>Age:</label>
  <input type="text" />
</form>
```

In the example above we created 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**.

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, 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 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><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, but it is certainly not as easy to read and understand as the code below, which is properly indented:

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

  <!-- This is a comment. It is not interpreted as code and, thus
  will not affect your web page. Comments (along with indentation)
  can be used to improve the readability of your code. -->
  <body>
    <form>
      <label>First name:</label>
      <input type="text" />
      <label>Surname:</label>
      <input type="text" />
      <label>Gender:</label>
      <select>
        <option value="male">Male</option>
        <option value="female">Female</option>
        <option value="other">Other</option>
      </select>
      <label>Age:</label>
      <input type="number" />
      <input type="submit" value="Add user" />
    </form>
  </body>
</html>
```

As you can see above, the indentations are used to show which HTML elements are nested within other HTML elements. Also, all the other elements are nested within the `<html>` element.

Take note

HyperionDev has **code style guides**. Style is one of the metrics your tasks are graded on. If you want to score the highest possible grade, be sure to adhere to the provided style guides.

Extra resource

Remember that with our courses you are not alone! To become a competent 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** and **CSS**.

Take note

The task(s) below is/are **auto-graded**. An auto-graded task still counts towards your progression and graduation. Give it your best attempt and submit it when you are ready.

When you select “Request Review”, the task is automatically complete, you do not need to wait for it to be reviewed by a mentor.

You will then receive an email with a link to a model answer, as well as an overview of the approach taken to reach this answer.

Take some time to review and compare your work against the model answer. This exercise will help solidify your understanding and provide an opportunity for reflection on how to apply these concepts in future projects.

In the same email, you will also receive a link to a survey, which you can use to self-assess your submission.

Once you've done that, feel free to progress to the next task.

Auto-graded task 1

Follow these steps:

- Create an HTML file called **index.html**.
- Set out the basic document template.
- Make the title "My Childhood".
- Make two headings (`h1`), with the content being "Hobbies" and "Toys".
- Each of these headings should have two subheadings of type `h2` (with a total of four). List two hobbies you had as a child and two of your favourite toys.
- Within these subheadings, create two paragraphs per `h2` heading. In each of these paragraphs, mention a memory that you have about that particular hobby or toy. You don't need to make this too long – a sentence is fine (unless you're feeling very creative!).
- Be sure to use italics and bold text on your page – you can choose where.
- Before submitting your code, check it with the HTML validator **here**.

Be sure to place files for submission inside your task folder and click "Request review" on your dashboard.

Auto-graded task 2

Follow these steps:

- Create an HTML file called **tables.html** in this folder.
- Set out the basic document template, giving it a title and headings as you see fit.
- Create a table with three columns, with column names in bold. These names should be: "Topic", "Name of the website", and "URL".

- Populate this table with the details of three resources that you have found on the web that provide useful advice for coding in HTML. Feel free to use resources referred to in earlier tasks.
- Before submitting your code, check it with the HTML validator **here**.

Be sure to place files for submission inside your task folder and click "Request review" on your dashboard.

Auto-graded task 3

Follow these steps:

- Open the **index.html** file that you created.
- If you have not yet done so, make sure that your web page contains appropriate headings and paragraphs.
- Add at least three relevant pictures (either from your PC or online) to your web page.
- Add a “back to top” link at the bottom of your web page that will return the user to the top of the web page when clicked.
- Before submitting your code, check it with the HTML validator **here**.

Be sure to place files for submission inside your task folder and click "Request review" on your dashboard.

Share your thoughts

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

Do you think we’ve done a good job or do you think the content of this task, or this course as a whole, can be improved?

Share your thoughts anonymously using this **form**.