Hyperiondev

TASK

HTML Overview

Visit our website

Copyright © 2021 HyperionDev. All rights reserved.

Introduction

WELCOME TO THE HTML OVERVIEW TASK!

It's time to learn to create a website! 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. In later tasks, we'll combine HTML with your JavaScript knowledge to create fully-fledged websites!

Get in touch

Connect for support

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

The best way to get help is to login to Discord at **https://discord.com/invite/hyperdev** where our specialist team is ready to support you.

Our expert code reviewers are 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!

Copyright © 2021 HyperionDev. All rights reserved.

A note from the

HyperionDev Team

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

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>
```


Copyright © 2021 HyperionDev. All rights reserved.

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

Copyright © 2021 HyperionDev. All rights reserved.

- 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 -->
```

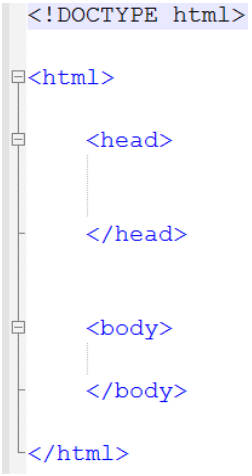
  When the browser encounters the tag `<h2>` it knows to treat the information between the opening `<h2>` tag and the closing `</h2>` tag as a heading. Similarly, the browser will display the information between the tags `<p>` and `</p>` as a paragraph of text.
- You will learn more about specific HTML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

Copyright © 2021 HyperionDev. All rights reserved.



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.

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. 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, `<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 text. This paragraph can be described using various attributes including alignment and font size.

Consider the following:

Copyright © 2021 HyperionDev. All rights reserved.

```
<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 JavaScript (IDs are not compulsory, but they can come in very handy, as you'll see in later tasks). 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 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.
  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>** 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 **<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>
```


Copyright © 2021 HyperionDev. All rights reserved.

- **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.

- **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 far you are with 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.

  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 showing bullet points, these list items are numbered.

Copyright © 2021 HyperionDev. All rights reserved.

```
<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, 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 which elements contribute to which visual appearance on the web page.

Copyright © 2021 HyperionDev. All rights reserved.

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 **example.html** and **example2.html**.

HTML SYNTAX

As a software engineer and web developer, you are going to learn many new languages. Each of these has its own rules which 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, or 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! 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="url" target="_blank">link text</a>
```

The `<a>` element is used to add all links on a web page, and stands for "**a**nchor"; `<a href>` stands for "**a**nchor **h**ypertext **ref**erence", i.e. to anchor a link using html. Using this element you can link to other pages in your website, to external web pages and to enable users to send an email.

Copyright © 2021 HyperionDev. All rights reserved.

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 = "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.

Copyright © 2021 HyperionDev. All rights reserved.

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. **target="_blank"** will open the link in a new tab instead of changing our current tab to the specified web address. This is very usefull when you want to keep a user on your website and don't want them to leave.
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 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=** 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.

Copyright © 2021 HyperionDev. All rights reserved.

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 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 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 and 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 this 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:

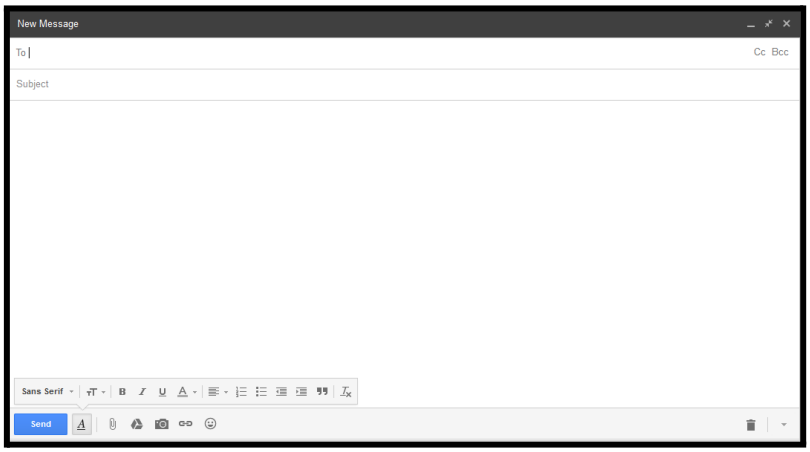


Figure 1: Here's a sophisticated form from Gmail (mail.google.com) — this is the popup text editor used to draft an email.

Copyright © 2021 HyperionDev. All rights reserved.

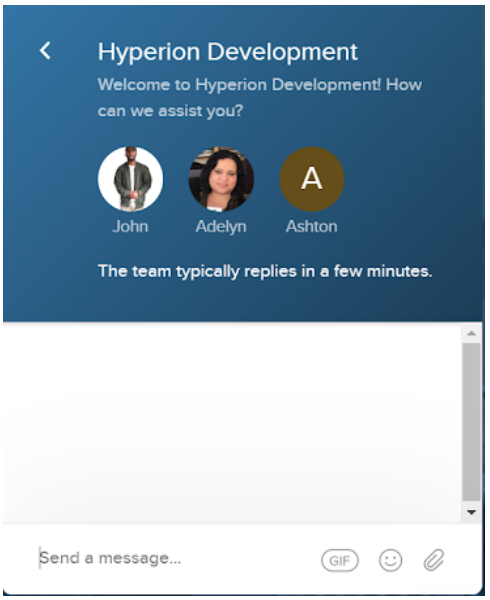


Figure 2: HyperionDev's (**www.hyperiondev.com**) chat box is a very sophisticated form, but a form nonetheless

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.

```
<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 create a form to capture our user's biographical information. It captures the following information:

- First Name
- Surname

Copyright © 2021 HyperionDev. All rights reserved.

- 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. 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 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>
```


Copyright © 2021 HyperionDev. All rights reserved.

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>
      <!--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 above, the indentation is 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

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.

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

Copyright © 2021 HyperionDev. All rights reserved.

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. It also makes webpages more informative and adaptable, allowing browsers and search engines to interpret content better. Elements such as `<div>` and `<span>` are not semantic elements since they provide no context as to what is inside of those tags.

LIST OF COMMON SEMANTIC HTML ELEMENTS

<article> - This defines independent and self-contained content in a webpage, which is indented to be reusable.

Examples use cases:

- A forum post
- A magazine or newspaper article
- A blog entry
- A product card
- A user-submitted comment

Copyright © 2021 HyperionDev. All rights reserved.

Here is a demo of how to use the `<article>` element.

```
<article class="forecast">
    <h1>Weather forecast for Seattle</h1>
    <article class="day-forecast">
        <h2>03 March 2022</h2>
        <p>Rain.</p>
    </article>
    <article class="day-forecast">
        <h2>04 March 2022</h2>
        <p>Periods of rain.</p>
    </article>
    <article class="day-forecast">
        <h2>05 March 2022</h2>
        <p>Heavy rain.</p>
    </article>
</article>
```

<aside> - This defines a portion of a webpage whose content is only indirectly related to the webpage's main content.

The following example uses `<aside>` to mark up a paragraph in an article which is only indirectly related to the main article content.

<cite> - This defines the title of a creative work.

```
<article>
  <p>
    The Disney movie entitled <cite>The Little Mermaid</cite> was
    first released in theatres in 1989.
  </p>
  <aside>
    <p>
      The movie earned $87 million during its initial release.
    </p>
  </aside>
  <p>
    More info about the movie...
  </p>
</article>
```


Copyright © 2021 HyperionDev. All rights reserved.

<details> - This creates a widget in which information is visible only when the widget is toggled into an “open” state. A summary or label must be provided using the `<summary>` element.

Here is an example of the `<details>` element.

```
<details>
    <summary>Details</summary>
    Something small enough to escape casual notice.
</details>
```

<summary> - This specifies a summary, caption, or legend for a `<details>` elements’s disclosure box. Clicking on the `<summary>` element toggles the state of the parent `<details>` element between *open* and *closed*. See the example above under the `<details>` heading for an example of the `<summary>` element.

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

Here is an example of the `<figcaption>` element.

```
<figure>
    <img
src="https://hips.hearstapps.com/hmg-prod/images/womanyellingcat-1573233850.jpg"
         alt="A cat meme template">
    <figcaption>A cat meme template</figcaption>
</figure>
```

<figure> - This represents a self contained content like illustrations, diagrams, photos, and code listings which is specified using the `<figcaption>` element. You can see an example of using the `<figure>` element above.

Copyright © 2021 HyperionDev. All rights reserved.

<footer> - This defines a footer for a webpage 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 `<footer>` tag.

```
<article>
    <h1>How to be a wizard</h1>
    <ol>
        <li>Grow a long, majestic beard.</li>
        <li>Wear a tall, pointed hat.</li>
        <li>Have I mentioned the beard?</li>
    </ol>
    <footer>
        <p>(c) 2018 Gandalf</p>
    </footer>
</article>
```

<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

Here is an example of the `<header>` element

```
<header class="page-header">
    <h1>Cute Cat Express!</h1>
</header>

<main>
    <p>I love cats <em>so</em> much! Like, really, a lot. They're adorable and
so, so snuggly soft!</p>
</main>
```

<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 under the `<header>` element for an example of the `<main>` element.

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

Example use cases of navigation sections:

Copyright © 2021 HyperionDev. All rights reserved.

- Menus
- Tables of contents
- Indexes

```
<nav>
    <a href="/">Home</a>
    <a href="#about">About us</a>
    <a href="#articles">Articles</a>
    <a href="#contact">Contacts</a>
</nav>
```

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

Here is an example of the `<section>` element.

```
<section>
  <h1>Pink Flamingo</h1>
  <p>Plastic flamingos have been used as garden ornaments since the late
1950s.</p>
</section>
```


Copyright © 2021 HyperionDev. All rights reserved.

WHY USE SEMANTIC HTML?

ACCESSIBILITY

Screen readers and browsers can interpret Semantic HTML better, which makes webpages 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
Hyperion Team

For more information about semantic HTML, please refer to this **playlist** of video tutorials. The entire tutorial series takes just over an hour, and should provide a lot of helpful context!

Copyright © 2021 HyperionDev. All rights reserved.

Compulsory Task 1

Follow these steps:

- Use Visual Studio Code to create an HTML file called **childhood.html** in this folder.
- Set out the basic document template.
- Make the title “My Childhood”
- Make two headings (‘h1’s), with the content being “Hobbies” and “Toys”.
- Each of these headings should have two subheadings of type h2 (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**.

Compulsory 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 3 columns, with column names in bold. These names should be: “Topic”, “Name of the website” and “URL”.

Copyright © 2021 HyperionDev. All rights reserved.

- Populate this table with the details of 3 resources that you have found on the web that provide useful advice for coding in JavaScript. Feel free to use resources referred to in earlier tasks.
- Before submitting your code, check it with the HTML validator **here**.

Compulsory Task 3

Follow these steps:

- Open the **childhood.html** file that you created.
- If you have not yet done so, make sure that your webpage contains appropriate headings and paragraphs.
- Add at least 3 relevant pictures (either from your PC or online) to your webpage. Don't worry about the sizes and position for now — we will deal with that in the CSS section.
- Add a "back to top" link at the bottom of your webpage that will return the user to the top of the webpage when clicked.
- Make the following semantic elements: <footer>, <header>, <main>, <nav>, and <section>.
- Before submitting your code, check it with the HTML validator **here**.

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?

Copyright © 2021 HyperionDev. All rights reserved.

Click here to share your thoughts anonymously.

Copyright © 2021 HyperionDev. All rights reserved.