Lecture Notes: HTML

HTML Elements:

Links

- To create a link on a web page, you need to wrap an anchor `(<a>)` element around text, then set its `href` attribute to the URL you want to link to.

```
<p>Check out <a href="https://www.freecodecamp.org/">freeCodeCar
```

- you can additionally also provide relative hrefs: exampleL If you have loaded up a web page : bristol.ac.uk/students/info.html

then :

```
<a href= "relative path">Our Courses</a>

where:
"/courses" ==> bristol.ac.uk/courses
"courses" ==> bristol.ac.uk/students/courses
"../courses" ==> bristol.ac.uk/courses
```

- The Target Attribute, this attribute tells the browser to open the link in a new tab, you can set the target attribute to _blank as follows:

```
<p>Check out <a href="https://www.freecodecamp.org/" target="_bl
```

Tabnabbing:

- It is recommended that you always add `rel="noopener noreferrer"` the `rel` attribute sets the relationship between your page and the linked URL. Setting it to `noopener noreferrer` is to prevent a type of phishing known as tabnabbing. This exploits the browser's default behaviour with `target="_blank"` to gain partial access to your page thorugh the `windon.object` API
- With tabnabbing, a page that you link to could cause your page to redirect to a fake login page. This would be hard for most users to notice because the focus would be on the tab that just opened – not the original tab with your page.
- Then when a person switches back to the tab with your page, they would see the fake login page instead and might enter their login details.

Forms:

- Form has two key attributes:
  - Method: What type of HTTP request should be sent as a result of interacting with the form
  - action: this specifies the URL that the form interacts with using that method, this can be specified relatively as with URLs
- Forms create key:value pairs, in the example above, the input id element name is where the user would type their name.

```
<form method= "post" action="/comment">

<p>
    <label for= "name">
        Name:
    </label>
    <input id = "name" name="name" type = "text"/>

    <label for= "age">
        Age:
```

```
    </label>
    <input id = "age" age="age" required type = "number"/>
</p>

<p>
    <button type= "submit">
        OK
    </button>
</p>


</form>
```

- A lot of the work can be done using the attribites of the elements in the field.
- e.g. Aage must be a number, you can set the attribute of the input which handles age to `type="number"`

Tables:

```
<table>

    <thead>
        <tr>
            <th>Name</th>
            <th>ID</th>
            <th>Age</th>
        </tr>
    </thead>

    <tbody>
        <tr>
            <td>Sarah</td>
            <td>14765B</td>
            <td>27</td>
        </tr>
```

```
        <tr>
            <td>Jon</td>
            <td>14765C</td>
            <td>29</td>
        </tr>
    </tbody>

</table>
```

- Output:

Name	ID	Age
Sarah	14765B	27
Jon	14765C	29

Tom Notes:

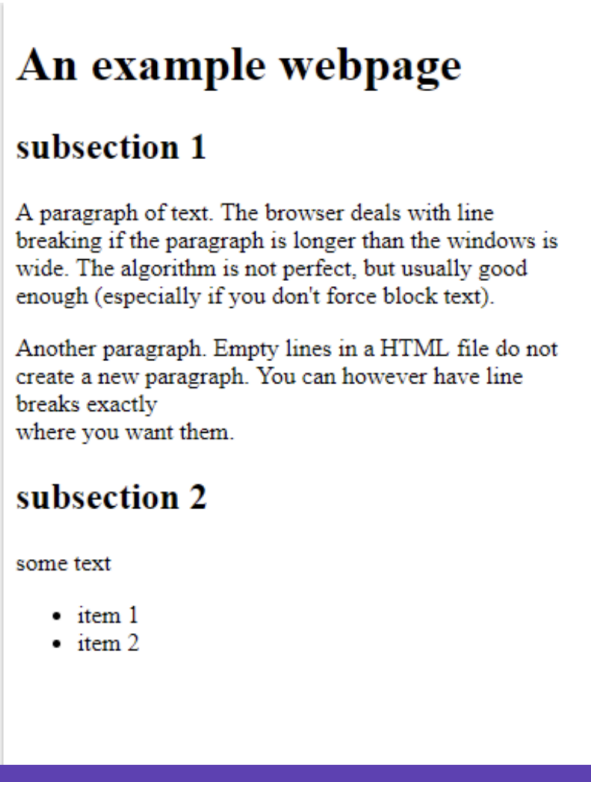
What is HTML?

It stands for **Hyper-Text Markup Language**

- HTML5 is the current best accepted standard
- *marking up* simply refers to the **annotation of a document**
    - *markup language* is a **standard scheme of annotations that you might apply to a document**
- below is the code for a HTML document, and how a typical browser would render this document

HTML5

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>A web page</title>
  </head>
  <body>
    <h1>An example webpage</h1>
    <h2>subsection 1</h2>
    <p>A paragraph of text. The browser deals
    with line breaking if the paragraph is
    longer than the windows is wide. The
    algorithm is not perfect, but usually
    good enough (especially if you don't force
    block text).</p>
    <p>Another paragraph. Empty lines in a
    HTML file do not create a new paragraph.
    You can however have line breaks
    exactly<br /> where you want them.</p>
    <h2>subsection 2</h2>
    <p>some text</p>
    <ul>
      <li>item 1</li>
      <li>item 2</li>
    </ul>
  </body>
</html>
```



An example webpage

subsection 1

A paragraph of text. The browser deals with line breaking if the paragraph is longer than the windows is wide. The algorithm is not perfect, but usually good enough (especially if you don't force block text).

Another paragraph. Empty lines in a HTML file do not create a new paragraph. You can however have line breaks exactly
where you want them.

subsection 2

some text

- item 1
- item 2

HTML was originally created for scientific paper navigation, to follow citations in when cited, by following hyperlinks which take you directly to the cited article

- this approach became higher popular

So at its core, HTML is a markup langauge for writing documents that you can interact with, and link the reader to other documents.

Basic Strucure of HTML Document

- it should always look like this:

```
<!DOCTYPE html>
<html lang="en>
    <head>
        <meta charset="utf-8" />
        <title>A web page</title>
    </head>
    <body>
        content here
    </body
</html>
```

- we start with a document type declaration: `<!DOCTYPE html>` **this refers to the *schema*** which should be used for understanding the rest of the document
    - ommiting this will result in many systems still making a decent guess on how the document will be interpreted

Next we have the html **root element:** `<html lang="en">`

- this goes around all of the content in the rest of the document
- everything else will be included inside this top-level html element

Then, there are two major sections to any html document:

1. **The head:** `<head>` - this contains *metadata* about the document
2. **The body:** `<body` - this contains the *content* of the document

Head element: Metadata

In the above example, there are two important pieces of metadata in the head:

the first is a meta tag with an attribute related to the character set for text encoding in the document - most of the time you want the character set to be `"utf-8"`

The second piece of metadata is the title of the document: `<title> A web page</title>`

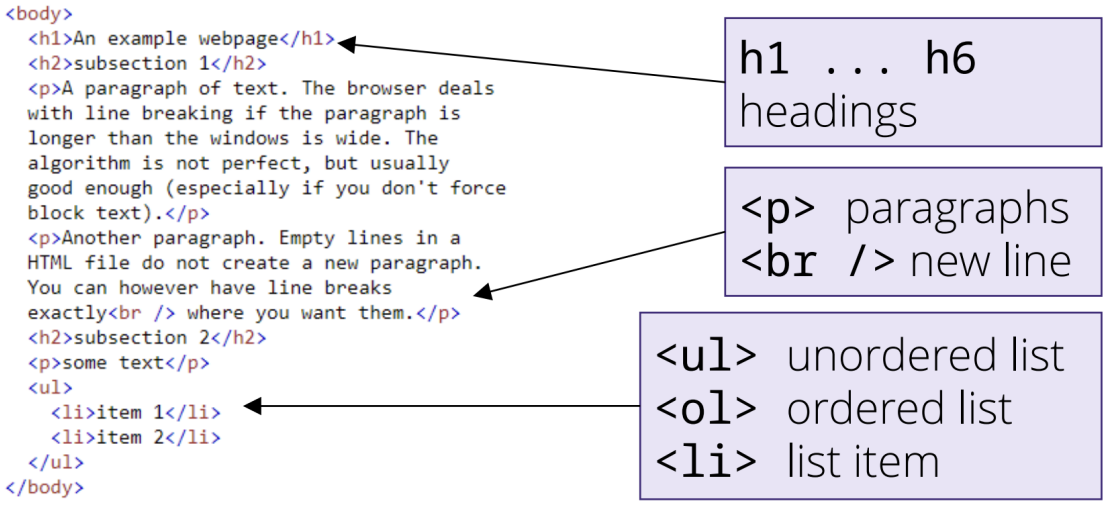
REMEMBER - this is only metadata about a document and not its content

- **I.e. the title wont be visible on the webpage itself, but in the title bar (tab)**
- Only the content **within** the `<body </body>` tags is included on the page itself

Tags

So how do you write your document body?

- HTML content should be contained in tags which express the strucutre of the document



  - To mark sections of text as headings you use `h` tag form 1 - 6 decreasing in size
  - most text in the document should be contained within paragraphs using the `p` tag
    - to seperate paragraphs you have to close one with `</p>` and open a new one, simply using a newline isnt enough

- if you want to include a *break* in the text within a paragraph you can use the `<br />` tag
    - it includes the closing / in the tag as it indicates there is nothing that goes inside a break element
- finally you can see a list element: `<ul>` for “unordered list” and then some list items
- these are denoted as list items with: `<li>` then the contents of the item followed by the closing tag `</li>`

Difference between an ordered list and unordered list:

Ordered list:

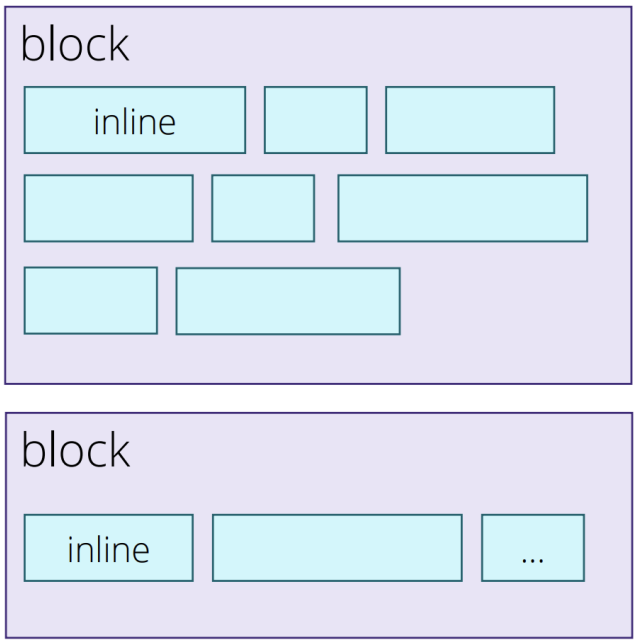
1. item1
2. item2

Unordered list:

- item1
- item2

Layout

- an important difference is between **block elements** and **inline elements**



Blocks are larger structures that can contain other elements, they are often used to control the layout of your text.

Inline elements. alter the interpretation of the document, but dont have an effect on document layout, for example `<em>` `</em>` tags which indicate *emphasis* should be placed on this part of the paragraph. This could mean a **screen reader** would place emphasis on this piece of text, or it would be presented in *italics* on the webpage:

```
<p>A paragraph
of <em>example
text</em>.</p>
<p>And another.
</p>
```

The distinction between block and inline tags is a remnant from older versions of HTML. HTML5 the question of whether an element affects the document layout is something which should be **handled by the stylesheet** and not the type of the element itself.

Semantic Tags

The ideal of a HTML5 document is that tags only affect the **meaning** of a document, not its presentation.

- this allows for different devices to render the same document in different ways
- e.g. a mobile device doesnt need to render a page in the same way as a laptop, as long as it conveys the same meaning
    - or a screen reader doesnt need to intepret the visual layout for a blind person

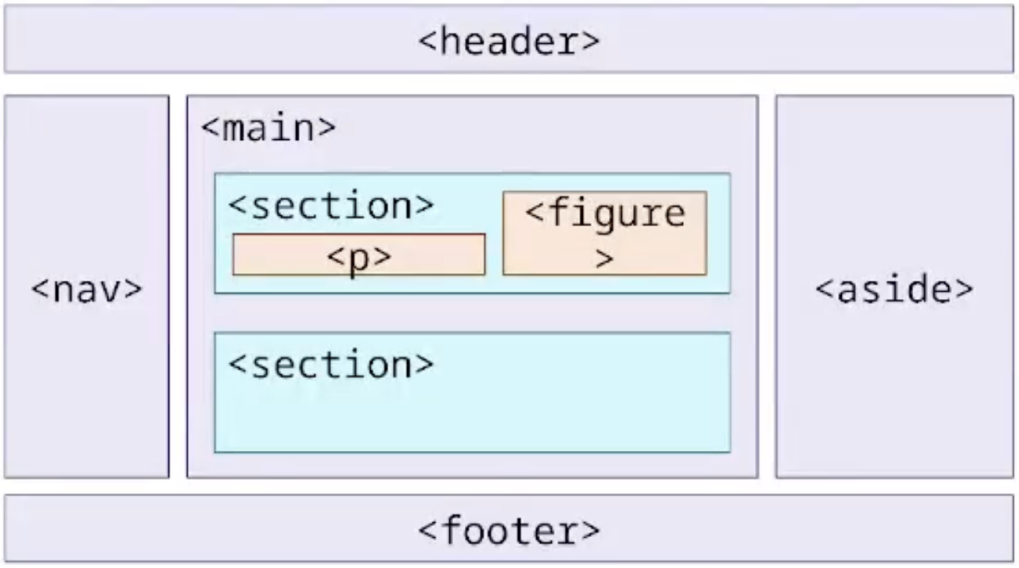
new		old	
`<em>`	emphasis	`<b>`	**bold**
`<strong>`	important	`<i>`	*italics*
`<q>`	quotation	`<u>`	underline
`<cite>`	citation	`<s>`	~~strike out~~
`<var>`	variable	`<tt>`	`monospace`
`<code>`	source code	`<small>`	small

This aim was generally compatible with how HTML worked, there are some elements in classic HTML which in practise refer tot eh presentation of a document, e.g. the old tags of `<b>` and `<i>` indicating bold or italics - these refer to visual presentaion rather than semantics

The HTML5 approach is to **focus on the meaning of the text** and let the stylesheet deal with presentation.

- you still see the old tags used however,

When you are writing in HTML5 compliant semantic tagging you will still be wanting to think about how the rendering of the document will look like. Below could reflect how elements may be layed out in a typical webpage:



- header for the webpage with intro content
- a navigation area containing links to navigate the page or website in generral
- main contains the main content of the page
- the side element will be content only indirectly related to the main content
- footer closes the document, typically with copyright notices or info about the author

Placeholder tags

- in additoin to section tags for structure and paragrpah layout, there are also other commonly used tags:

`<div>` - these are block tags to create divisions in the document layout

`<span>` - these are *inline* elements which are used to apply some custom interpretation

`<section>` - used to define sections of a document which group thematic content, making it easier for for browsers and developers to make sense of the structure and semantics of a webpage

Attributes

- alongside the default meanings of HTML elements, elements can also have attributes
- these are properties of the attribute to give some additional information
- two of the most common are its `id` and its `class` :

```
<p id="today">28 September</p>
<p class="info">Lecture 2</p>
<p class="info">QB 0.18</p>
```

- id allows you to **refer** to a specific element by its **unique id**, the above `p` will be unique to all other `p` based on its id
  - you often need ids for elements if you want to interact with them using **JavaScript**
- `class` is used to describe the **purpose** of an element
  - in the above both of the p elements have the same class with the purpose of providing information
  - `class` is important when we want to alter the presentation of a document using style sheets and you can apply the same style to all elements of the same class

Multiple Classes for the same element:

- To assign multiple classes to an HTML element, you simply separate each class name with a space within the class attribute of the element.

```
<div class="unit y1-tb1 cp20">
    <b>COMS10014</b>
    <p>Mathematics A</p>
</div>
```

- In this example, the `<div>` element has three classes: `unit`, `y1-tb1`, and `cp20`. This means it inherits the styling rules from all three of these classes. You can then define CSS styles that apply to any combination of these classes

---

HTML Elements

Links

- these make the hyperlinks possible

The `href` attribute is used in `<a>` (anchor) tags to specify the link's destination URL. The path you provide in an `href` attribute can be either absolute or relative.

- **Absolute paths** specify the full URL, including the protocol (e.g., `http` or `https`), domain, and full path to the resource.
- **Relative paths** are more common for internal links, referring to a resource on the same domain. These paths are relative to the current document's location or a base URL if one is set in the document's `<base>` tag.
- links are implemented through **hyper references:** `href` these are anchortags and the text between these is what is being linked to the reader

```
<a href="/courses">Our Courses</a>
```

- the `href` provides a *mechanical* path to that document
- something that could be included in a href is a URL which can be accessed over HTTP

- the link would give an absolute reference to that document
- in other cases like the above example it is providing a *relative address* with no domain name specified

```
bristol.ac.uk/students/info.html :

"/courses"    => bristol.ac.uk/courses
"courses"     => bristol.ac.uk/students/courses
"../courses" => bristol.ac.uk/courses
```

- href's can navigate up file directory trees the same way the terminal handles relative addressing

NEED MORE NOTES OR CLARIFY UNDERSTANDING

Forms

- crucial for interactivity on the web

- allows user typed text which when you press a button it is posted to the webpage
- forms are just a collection of HTML elements
- the crucial main element being `<form>` :

```
<form method="post" action="/comment">
  <p>
    <label for="name">Name:</label>
    <input id="name" name="name"/>
  </p>
  <p>
    <button type="submit">OK</button>
  </p>
</form>
```

Forms have two key attributes:

1. the `method` - this specifies what sort of **http request** should be sent as a result of interacting with the form
2. `action` - this specifies the URL that the form should interact with using the method

Forms create **key-value pairs** for the user input

- They keys are the name for input options
- values contain what the user has inputted for this element

For example in the above:

```
<label for="name">Name:</label>
<input id="name" name="name"/>
```

- so the input element with the id name, if you type your name in and then submit the form, a data payload will be created mapping the name id and the value you entered, e.g. “Tom”
- this is then sent to the website using a HTTP `POST` request (the method)
- submission of a form is usually handled by a **button** with a type attribute of `submit`

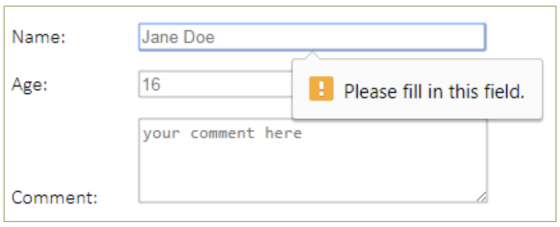
```
<button type="submit">OK</button>
```

- the browser will know that when this is clicked, the form should be submitted

Form Validation

While there will be **server-side** code that handles input from a user, forms need features to make sure requests are properly formatted on the **client-side**

E.g. attributes can be set on elements which indicate that a field requries input before it can be submitted:




```
<input required type="number">
```

Keywords can also inform browsers than input can be autocompleted using info that the browser may have saved, .e.g a name or address

```
<input type="text"
autocomplete="name">
```

name, email, address-line1, country, tel, ...

cc-name, cc-number, ...

From Input Types

- there are many input types which can be used in forms, these typically have standard controls implemented in the browser to make inputs easier

button	month	text
checkbox	number	time
color	password	url
date	radio	week
datetime-local	range	
email	reset	<textarea>
file	search	
hidden	submit	*(source: MDN)*
image	tel	

Dropdown Boxes using Select

- this lets the user pick from some predefined options for that input

```
<select name="animal">
  <option value="dog">Dog</option>
  <option value="cat">Cat</option>
</select>
```

Tables

```
<table>
  <thead>
    <tr><th>Name</th><th>ID</th></tr>
  </thead>
  <tbody>
    <tr><td>Sarah</td><td>100</td></tr>
    <tr><td>Jon</td><td>101</td></tr>
  </tbody>
</table>
```

- tart with a table element `<table>` which everyting will be inside of
- table head `<thead>` will contain column names etc
  - in the header you specify a column heading using `<th>` tags
- table body `<tbody>` will include the data contained in the table
  - `<td>` stands for table data which each td tag containing one piece of data

You will align the td with the column name for example, at the end of a row close it with `</tr>`

You then proceed by creating rows using `<tr>` and each row should contain the same number of cells.

Table benefits

- you can interact with tables using JavaScript
- there are lots of pre built libraries to make these interactions very easy

Top tips:

1. **Nesting of Block Elements Inside `<p>` Tags**: In HTML, the `<p>` (paragraph) tag cannot properly contain block-level elements such as `<ul>` (unordered list). This is a common error where a `<p>` tag is opened before a block element and closed after it, which is not allowed.
2. **Missing Closing Tag for `<a>` Element**: Although the closing tag is optional for some tags in HTML5 when followed by certain elements, it's generally a good practice to close all tags explicitly to avoid confusion and ensure compatibility.

Example HTML Code

1. Basic HTML Structure

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initia
l-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>Hello, World!</h1>
    <p>This is a simple HTML document.</p>
</body>
</html>
```

This snippet shows the basic structure of an HTML document, including doctype declaration, language attribute, and meta tags for character set and responsive design.

2. Embedding Images and Links

```
<body>
```

```
    <h1>Embedded Content</h1>
    <img src="image.jpg" alt="Sample Image" width="500">
    <p>Check out more at <a href="https://www.example.com">Ex
ample Site</a>.</p>
</body>
```

This demonstrates how to embed an image and create a hyperlink.

3. Tables

```
<table border="1">
    <tr>
        <th>Name</th>
        <th>Email</th>
    </tr>
    <tr>
        <td>John Doe</td>
        <td>johndoe@example.com</td>
    </tr>
    <tr>
        <td>Jane Smith</td>
        <td>janesmith@example.com</td>
    </tr>
</table>
```

This snippet is an example of an HTML table with headers and multiple rows.

4. Forms

```
<form action="/submit_form" method="post">
    <label for="name">Name:</label>
    <input type="text" id="name" name="name"><br><br>
```

```
    <label for="email">Email:</label>
    <input type="email" id="email" name="email"><br><br>
    <input type="submit" value="Submit">
</form>
```

Here's an HTML form for collecting user inputs like name and email, using POST method.

5. Lists

```
// Bullet point list:

<ul>
    <li>Coffee</li>
    <li>Tea</li>
    <li>Milk</li>
</ul>


// Numbered list:

<ol>
    <li>Coffee</li>
    <li>Tea</li>
    <li>Milk</li>
</ol>
```

This snippet shows an unordered list, a common way to display items in a bullet format.

6. Video Embedding

```
htmlCopy code
<video width="320" height="240" controls>
    <source src="movie.mp4" type="video/mp4">
```

```
    Your browser does not support the video tag.
</video>
```

This code embeds a video file with controls for play, pause, and volume.

7. Using CSS for Styling

```
htmlCopy code
<head>
    <style>
        body { font-family: Arial, sans-serif; }
        h1 { color: navy; }
        p { color: olive; }
    </style>
</head>
<body>
    <h1>Styled Content</h1>
    <p>This paragraph is styled using internal CSS.</p>
</body>
```

This example includes CSS directly within the HTML to style elements like headers and paragraphs.

8. JavaScript Interaction

```
htmlCopy code
<script>
    function showAlert() {
        alert("Hello from JavaScript!");
    }
</script>
<button onclick="showAlert()">Click Me!</button>
```