Intro to HTML

While our previous website must've been an absolute adrenaline rush, there's much more to building a proper website! Our aim over the next couple of days is going to be building a blog site - Let's call it BrightBlog!

BrightBlog is going to be built using the best practices - including using semantic HTML and proper CSS practices to make sure that the site we are building is maintainable and performant!

First, let us talk a little bit about HTML!

HTML5 is the currently used version of HTML, which stands for *Hypertext Markup Language*. HTML is the code which browsers use to render the content of a web page. MDN describes HTML as:

> "...a series of elements which you use to enclose, wrap, or *mark up* different parts of the content to make it appear or act a certain way."

Almost every web page you've ever visited will have been made up with HTML. In this lesson we will be learning how to write HTML5 in the way it was intended to be written, known as *"semantic HTML5"*.

This will allow us to create well built web pages for our web apps. Today we will be creating a very simple web page to test and explore some of the most commonly used features of HTML.

HTML Elements

As you know, when we work with HTML, we wrap our content in a series of "elements". Elements take the form of opening and closing tags, with the wrapped content between them. There are many types of HTML elements available for us to use, and each has a different use case: - `<ol>` or `<ul>` and `<li>` for ordered or undordered lists with list items. - `<h1>` through `<h6>` for headings. - `<p>` for paragraphs.

For a full list of elements available in HTMl5 visit the HTML Cheat Sheet.

Use and Misuse

When building an HTML page, it can quickly take on a complex structure and become full of

many elements. Due to this the code for the page can become quite hard to read. One way in which we can mitigate that unreadability, as well as increase the accessibility for our web pages, is by using the appropriate tags in the right places.

Far too often do people overuse the infamous `<div></div>` tag when writing their content into a web page. When the `<div>` tag is misused in this way it can quick make our code this barely intelligible mess of the same tag wrapping the same tag wrapping the same tag.

The `<div>` tag does have actual uses. The tag exists not to wrap and lay out **content**, but rather to allow us to build more complex **aesthetic** layouts - allowing us to group various elements or sections of our page without implying anything about the content in the group. It's wrongly more often used to group **content**, which is ironically the exact opposite of its intended use.

HTML5 offers us many alternatives to this `<div>` tag for the purposes of grouping and describing content - many of which we will be exploring throughout the construction of our web page.

The Basic Skeleton

When we start writing an HTML document, we usually start with a standard skeleton.

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

</body>
</html>
```

This will do a few things for us:

- `<!DOCTYPE html>` will tell our browser that this is an HTML document, and should be treated as such.
- `<html lang="en">` will tell the browser that whatever is inside it is the main HTML in the document.
- `<head>` tells the browser that whatever is inside it isn't to be rendered and exists for non-visual reasons.
- `<meta>` will describe our page to the browser and search engines, dictating some of the behaviours of the page or providing information.

- `<title>` will tell the browser and search engines the name of the page:
    - Search engines will display this name on their results pages.
    - The browser will put this name in the tab at the top of the browser.
- `<body>` will tell the browser that whatever is inside it is to be rendered in the browser and shown to the user.

This skeleton is standard, and most text editors will offer a snippet to generate it for you. In VS Code, we can type `HTML` and select the HTML:5 option to autocomplete this skeleton.

Getting Started

We'll start by creating a file called "index.html", by convention this is the name of the HTML file for the main page of a website. We'll go through our wireframe from top to bottom, left to right: and thus will start with the header.

```
touch index.html
```

Open the file up with a text editor and let's get started...

When creating the HTML structure for a page, it can be difficult to know where to start. Taking the design that you want to achieve and breaking it into boxes can make this process much easier. Then start at the top of the design and work your way from top to bottom and left to right.

We will work through the page in the following order:

- header
    - logo
    - nav links
- main content
    - main article
    - headings
    - main content
    - footnote - author
    - filter navigation
    - filter by month
    - filter by category
- footer

If you feel like you couldn't have come to this conclusion on your own, don't worry! The ability to quickly break down a design into an HTML structure that makes sense will come with time

and practice.

First, we will need to insert the standard HTML boilerplate that we saw earlier. Fortunately, most text editors offer this as a snippet, so we don't have to type it out manually every time.

In VS Code, you can simply type `html` and select the `html:5` autocomplete option.

```
<!-- index.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>
```

This boilerplate code is fairly standard and requires minimal changes for each new project. The only thing that we will change is the `<title>`, so that something more appropriate is displayed in the tab bar in the browser. Let's change the `<title>` to "CodeClan Blog Site".

```
<title>BrightBlog Site</title> <!-- UPDATED -->
```

Great, now we're ready to move on and start building the actual content of the webpage. As we outlined earlier, the top level elements of our page will be the header, main content - which will include the main article and filter nav links, and the footer. Let's block those out.

```
<body>
  <header>
    <!-- Our header content will go here -->
  </header>
  <main>
    <!-- Our main content will go here -->
  </main>
  <footer>
    <!-- Our footer content will go here -->
  </footer>
</body>
```

`<header>` will contain the blue header at the top of the page `<main>` will include the white content in the middle of the page `<footer>` will contain the blue footer at the

bottom of the page

The Header

The header tag implies that the content inside it is in some way a higher level section of content which will render above the main (`<main>`) part of a group of related contents. This group of related pieces of content can constiture an entire HTML document, a blog post, a news article, or even a form. In our case our `<header>` is heading up the entire page. It will contain a title for the site, as well as some navigational links for moving around between pages on our site. We won't be creating any other pages today though, they're just for show.

The `<header>` will have two child elements - the site logo and the navigational links. We won't be using any `<div>` elements here because there are more appropriate tags that better describe the content.

We can consider the site logo to be the top level header on the page, so we will use an `<h1>` tag for this and a `<nav>` tag to contain the list of navigational links.

```
<header>
  <h1>CodeClan Blog site</h1> <!-- NEW -->
  <nav>                       <!-- NEW -->

  </nav>                      <!-- NEW -->
</header>
```

Navigational Links

Now that we've got our side details in the header, we'll move on to our navigational links. Generally speaking, when we create a list of navigational links in the header (yes, a "navbar" is technically still a list of links), we use a `<ul>` , inside a `<nav>` . This tells anyone else reading our code, and also our browser - that this is a list of links, which are going to be used for navigation.

```
<nav>
  <ul>
    <!-- Our links will go here -->
  </ul>
</nav>
```

This structure makes things highly readable to anyone with the most basic of understandings of HTML. Something which many people forget to do when creating a structure like this, is that because we are using a `<ul>` , we must wrap our links in `<li>` elements.

```
<nav>
  <ul>
    <li>
      <a href="/home">Home</a>
    </li>
    <li>
      <a href="/about">About</a>
    </li>
    <li>
      <a href="/blog">Blog</a>
    </li>
    <li>
      <a href="/contact">Contact</a>
    </li>
  </ul>
</nav>
```

> NOTE: For now we will leave the `href` HTML attribute as these placeholder values, as we don't actually have any other pages to navigate to.

Our header is complete, we have our site logo and a small navigational menu. It definitely doesn't look the best at the moment, but don't fret - CSS will take care of this later.

HTML Attributes

We just added an `href` attribute to our `<a>` elements. This tells the browser where we want the user to be navigated to when they click on our link.

HTML attributes are like annotations that we can put on our HTML opening tags in order to describe them in a little more context that their element name provides. There are many different HTML attributes available to us, and there are different ones for different elements.

If you're interested in what attributes are available for different HTML elements, the MDN docs for different elements will provide them for you.

We will continue next week with the `main` tag!