The Main Content

We have a header, so now it's time to flesh our page out with something a bit more substantial. Let's have a look at adding the main part of our blog page: the blog post. In our case, the main part of our content consists of the blog post itself, and the sidemenu adjacent to it. We will communicate that this is the main part of our content by using the `<main>` tag to wrap all this content.

We will also be using an `<article>` tag and an `<aside>` tag to wrap our blog post, and our sidemenu, respectively.

```
<main>
  <article>
    <!-- Our blog post will go here -->
  </article>
  <aside>
    <!-- Our side menu will go here -->
  </aside>
</main>
```

An article tag groups related content of a **single, self-contained context, which could be independently reusable or redistributable**. A blog post would definitely match this description.

The reason you may not necessarily jump to an `<article>` tag every time you have some pieces of content which are related to each other is that sometimes the content isn't **self-contained**.

Our Blog Post

Now that we've outlined a place for our blog post, we can start looking at its internal structure. The temptation is to go simple: maybe an `<h2>`, a few `<p>` tags, not much more... however there is more we can do here to properly structure our content here.

Our blog post is not only going to have a title, but a subtitle for the date and read time as well: thus will require two heading tags; it will have the main content of the post; and it will also have an area at the bottom for the author.

This is starting to sound familiar... a piece of content containing a three part structure: one acting as an introduction, one as a main container for the bulk of the content, and another to round things off at the end. Here we see another place we can use our `<header>`, `<main>`, and `<footer>`. Due to their containment inside the `<article>` elements, these three elements will act as parts of the blog post, while the **same elements** elsewhere act as parts of the page as a whole: it's all about **context**.

In this case we have a heading and a subheading, rather than a single heading, so we will use an `<hgroup>` to contain the heading elements.

```
<!-- index.html -->
<!-- ... -->
<article>
  <hgroup>
    <!-- Our headings will go here -->
  </hgroup>
  <main>
    <!-- Our blog post will go here -->
  </main>
  <footer>
    <!-- Our author details will go here -->
  </footer>
</article>
<!-- ... -->
```

Once again we will progress top to bottom: so let's have a look at the hgroup. It will have two headings in it.

We already have an `<h1>` on the page, so we will need to think about the 'heading hierarchy' to keep things semantic. We will use an `<h2>` for the blog post title as it follows the page's `<h1>` in terms of scope; and we will use an `<h3>` for the read time subtitle, as it acts as a subtitle for an `<h2>`. Further than that, we want to very clearly communicate that the combination of our `<h2>` and `<h3>` constitute **one** overall heading for a piece of content, and thus will wrap them in an `<hgroup>` elements. The `<hgroup>` element exists for this exact purpose: grouping headings (*h-tags*, e.g. `<h1>` or `<h2>`).

```
<hgroup>
  <h2>Ten Amusing Facts About Dogs</h2>
  <h3>Read time: 5 minutes</h3>
</hgroup>
```

With our brief intro complete for our blog post it's time to move onto the main content: the post. Depending on the content of the post here, this could be incredibly simple, or incredibly complex. Let's aim for somewhere on the simple end.

A `<blockquote>` is used to highlight a particular part of a piece of content, usually a relative direct quote. When it's being used for a direct quote, we use a `<cite>` tag to credit the quote personally, and we also use a `cite="source"` attribute on the blockquote to credit the source of the quote.

Of course we can't deglect the mighty `<p>` element, we will use one of these elements **for each** paragraph in the content of our post.

> Feel free to cut & paste, rather than type this one out. Alternatively, you can use the `lorem` + `tab` shortcut!

```
<main>
  <p>
    Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eius
    magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ulla
    consequat.
  </p>
  <blockquote cite="https://www.brightnetwork.co.uk/">
    "Puppies." - <cite>Zsolt, 2024</cite>
  </blockquote>
  <p>
    Duis aute irure dolor in reprehenderit in voluptate velit esse cillu
    Excepteur sint occaecat cupidatat non proident, sunt in culpa qui of
  </p>
  <p>
    Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eius
    magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ulla
    consequat.
  </p>
  <p>
    Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eius
    magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ulla
    consequat.
    Duis aute irure dolor in reprehenderit in voluptate velit esse cillu
    Excepteur sint occaecat cupidatat non proident, sunt in culpa qui of
  </p>
  <p>
    Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eius
    magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ulla
    consequat.
  </p>
</main>
```

This will be enough content for the main content of our post for now.

Finally let's finish off this blog post and fill in our author profile section:

We used a `<footer>` here as it's a footnote to the article. This footer will have two child elements, the image of the author and an hgroup for grouping the headings of the footnote.

These headings are at the same level as headings of the `<main>` article above, so we will use `<h2>` and `<h3>` tags for the heading and subheading respectively.

```
<footer>
  <hgroup>
    <h2>Author: A. Dog Jr.</h2>
    <h3>Last posted: 2024-08-01</h3>
    <a href="https://www.brightnetwork.co.uk/" target="_blank">https://ww
  </hgroup>
</footer>
```

You'll notice we added a the HTML attribute `target="_blank"` to our `<a>` element, this is because we want the destination of the link to open in a new tab when clicked - meaning the user won't be navigated off of our blog site.

Just like that, the main part of our content is complete and properly structure. Just two more sections to go.

The Side Menu

We've got our header for the major navigation around pages and our article area to show off out blog post; so how about a side-menu for navigation within the blog section of our site.

For this we will use an `<aside>` tag: they contain content which **indirectly** related to the context which the main content relates to. In our case the menu will sit to the right of the article (once it's been CSS'd) and is indirectly related to our blog post because it relates to all the blog posts as a whole!

These `<aside>` elements aren't always for menus though, they can also be used for extra content which will side next to some of the main stuff - such as quotes.

We will start, as always, with the basic structure for what we want. We will use `<section>` tags to wrap each of the filter lists.

```
<aside>
  <section>
    <h2>Filter by Month</h2>
    <nav>
      <ul>
        <li><a href="/blog/month/january">January</a></li>
        <li><a href="/blog/month/february">February</a></li>
        <li><a href="/blog/month/march">March</a></li>
      </ul>
    </nav>
  </section>
  <section>
    <h2>Filter by Category</h2>
    <nav>
      <ul>
        <li><a href="/blog/puppies">Puppies!</a></li>
        <li><a href="/blog/dogs">Dogs</a></li>
        <li><a href="/blog/morepuppues">More Puppies!</a></li>
        <li><a href="/blog/cats">Cats</a></li>
      </ul>
    </nav>
  </section>
</aside>
```

That's our side-menu done - it'll look a little weird as it is right now... not very **side** for a side-menu. Positioning the elements can be done later, using CSS.

A lot of people work on their HTML and CSS at the same time - this is a completely fine way to work through things. However it can be highly useful to focus purely on one, and then the other: as it will give you an idea of what your site will look like if for any reason the CSS fails to load. It can help you ensure that your site is still useable without any styling.

The Footer

The last area of the page we want to work on is naturally the footer. Our footer will act as another overall summary of the site, and will also contain a list of important files or notices that must be accessible to the user. Ours will have another showing of the site logo, a list of pages available to be navigated to (sometimes called a sitemap), and a file list as mentioned.

Let's start by blocking out the basic structure of our `<footer>` content:

```
<footer>
  <h1>BrightBlog Site</h1>
  <section>
    <!-- Our sitemap will go here -->
  </section>
  <section>
    <!-- Our important files will go here -->
  </section>
</footer>
```

We know that we will want these sections to appear side by side, rather than one above the other, so we may have finally found a use-case for a `<div>` element!

We need to group these `<sections>` purely for formatting reasons, which is exactly what the `<div>` tag should be used for. We will wrap a `<div>` element around the two sections.

We will also add an `id` to this `<div>`, which will make it easier to access for styling later on.

```
<footer>
  <h1>BrightBlog Site</h1>
  <div id="footer-container"> <!-- NEW -->
    <section>
    </section>
    <section>
    </section>
  </div>                      <!-- NEW -->
</footer>
```

We'll flesh out these sections in order, starting with the site map as it appears closest to the left.

```
<section>
  <h2>Our Site</h2>
  <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>
</section>
```

The important files section will follow exactly the same pattern. We'll leave `href` attributes of the links blank to save ourselves a bit of typing. I'm sure you get the idea by now.

```
<section>
  <h2>Important Files</h2>
  <nav>
    <ul>
      <li><a href="/">Privacy Policy</a></li>
      <li><a href="/">Terms & Conditions</a></li>
      <li><a href="/">Cookie Policy</a></li>
    </ul>
  </nav>
</section>
```

Voila!

With that the HTML structure of our blog page is complete. It looks super basic at the moment: no colours, no layouts, not even a font. This is what users will see if they don't allow CSS, or if our CSS doesn't load for someone.

Our page is semantically structured: the code is nice, easy to read, and expressive; and tools like screenreaders or search engines crawlers will find this structure very easy to read/preferable for your SEO ranking.

In the next few lessons we will work on making our web pages look and feel like the real deal: firstly with pure aesthetics, then with an added layout to ensure our content sits where we want it to.

Recap

▶ What is an HTML attribute?

▶ What is the purpose of the alt HTML attribute on an image tag?

▶ What is a Progressive Web App?

▶ What's the difference between a `&#60section&#62` element, and an `&#60article&#62` element?

Conclusion

In this lesson we learned how to use some of the most common HTML lessons in ways that would be considered semantically correct. When writing HTML, it's important to think about the context of the content within the page when deciding what type of elements to use to structure it. Next we will learn how to use CSS to style our web page and make it look a bit more presentable.

Further Resources

- MDN docs on HTML