Sections

- What is Semantic HTML?
- What other tags did we cover today?
- What is CSS?
- What styles in CSS did we cover today?

What is Semantic HTML?

These are HTML elements that provide more information based on their name.

For example, the footer element (or tag) is exactly the same as a div element. It is a full width box that expands to the height of its contents. However, because of the name of the element, we know to put it at the foot, or the bottom, of our page. It often contains a copyright notice and a sitemap, and is the same for every page in our website. The header tag (not to be confused with the head tag) is the same; it is a div, but because of the name we know it belongs at the head, or top of the page, and will likely be the same across all pages of our website. Other examples of "semantic HTML" include: - the main tag, where we put the main content of our website (everything that isn't in the header tag or the footer tag) - section, where we organise sections inside our main content - nav, for all the buttons in a navigation bar

There are several more, which you can look up at any time.

This information is useful for us, as coders, to understand our code and the structure of our page better. It is also useful for other things that read our code; CSS, JavaScript, and accessibility tools (like screen readers).

Other tags

List tags

There are multiple kinds of list. We will look at two examples: unordered list (`ul`) and ordered list (`ol`) `. Each item in a list goes inside a` li` tag, and the whole list must go inside another tag that sets what kind of list it is. It looks like this:

```
<ul>
    <li>One item</li>
    <li>one more item</li>
    <li>another item</li>
</ul>
```

Span tag

The span tag is an inline tag (it can sit within text and next to other elements) that we use to style text. You will see how when you learn how to apply style on a tag, a class or an id, below.

```
<p> Hi, I have a <span>special</span> word!</p>
```

Input tags

An input tag is "self-closing", it does not have contents. You must set the `type` attribute on the input tag, but it defaults to text input: `<input type="text" />`

We tried the `type="date"` input, `radio` , `email` , `password` , `checkbox` , and `submit` . We will cover forms again later, but for now these are important things to know:

1. A radio button is used when the use must make a choice between several options. They have to select an option and they can only select one option. These elements have the same `name` attribute to link them together:

```
<input type="radio" name="breakfast"/> Eggs
<input type="radio" name="breakfast"/> Pancakes
<input type="radio" name="breakfast"/> Toast
```

2. An input element is implemented (designed and styled) by the browser - not by us, writing code. Not all types of input are supported by every browser. For example, the `week` type is not supported in FireFox, but IS supported in Chrome. This is because new elements are more recent, and browsers are not under any obligation to build in these elements. You can check that here: https://caniuse.com/?search=input%20date - test this element yourself in the browsers we installed.

```
<input type="week" />
```

3. Some input elements (e.g. email) have validation that prevents a user from submitting data we don't want. We will revisit this later, but it is a good reason to use inbuilt elements.
4. We use the `label` tag to create clickable text next to the input. The way we do this is using an `id` attribute and a `for` attribute, like so:

```
<label for="name-field">Please type in your name:</label><input type="text" id="name-field"/>
```

5. Use this reference to browse through other kinds of input tag: https://www.w3schools.com/tags/tag_input.asp

CSS

What is CSS? It stands for "Cascading Style Sheets". We use it to add style, design and positioning to our page. In the same way as we made an html file (`fileName.html`) we make a css file (`fileName.css`). We "link" this to our html using a `link` tag that goes in the `head` of our HTML document, like so:

```
    <link rel="stylesheet" type="text/css" href="index.css"/>
```

`href` points to the css file we created (this one is called `index.css`). `type` and `rel` are both mandatory attributes; they will always be set to the values in this example.

Inside your css file, the syntax looks like this:

```
selector {
    property: value;
}
```

Where selector is: - an element name (e.g. `div` or `p`) - a class - an id

A style applied to an element (using any kind of selector) is applied to all the children of that element by default. For example:

```
<div class="large-text">
    <p> example </p>
    <p> example </p>
</div>
```

```
.large-text {
    font-size: 20px;
}
```

Both the p tags will have `20px` size font, because they are inside the div that has 20px text.

Classes

The class attribute looks like this: `<p class="my-class-name"> example text </p>`

You may apply a class to as many elements as you like, and an element can have multiple classes, like this:

```
<p class="class1"> example text </p>
<p class="class1"> example text </p>
<p class="class1 class2 class3"> example text </p>
```

To access them in css, use a `.`, like so:

```
.my-class-name {
    color: #ff00aa;
}
.class1 {
    color: #0000ff;
}
```

You should almost always use a class to apply CSS, as it is more precise than an element and provides more reusability than an ID.

ID

The id attribute looks like this: `<p id="my-id"> example text </p>`

An ID must always be unique on a page. You can't have two ids with the same value. You can only have one id on an element. It is a unique identified for that element. We rarely use them.

To access a class in css, use a `#`, like so:

```
#my-id {
    color: #eeeeee;
}
```

Precedence

Styles "cascade". That means if a style is applied to an element in two places, the **bottom** one will override the top one. In the following example, the text will be **green**:

```
<p class="danger-text success-text"> example text </p>
```

```
.danger-text {
    color: red;
}
.success-text {
    color: green;
}
```

However, you can override this by using more selectors (element name, class, or id). It has an order of precedence like this: - element name: 1 - class: 10 - id: 100 So you will have to use 11 classes to override one id (which is another reason we don't use IDs).

If we modify our example above:

```
<p class="danger-text success-text"> example text </p>
```

```
p.danger-text {
    color: red;
}
.success-text {
    color: green;
}
```

The first style in our CSS now has a precedence of 11, while the second one only has a precedence of 10. Even though it is above, it will override `success-text`; the text will be red.

CSS styles

Sizes: `px` vs `%`

`px` means "pixels". To set a `width` or `height` that is relative to the parent element, use `%`. When there is no parent element, it will be a percentage of the window. The best usecase is this:

```
<div class="column">
</div>
<div class="column">
</div>
<div class="column">
</div>
```

```
.column {
    width: 33%;
    height: 200px;
}
```

Now (when we learn positioning tomorrow) we will be able to put all these columns next to each other on the page, as they are a third of the width of the page (or parent element).

This can also be used so that a box always remains a percentage of the screen when you change the size of the browser window (try it!).

Font-size: `px` vs `em` vs `rem`

A lot of font sizes are set in pixels (`px`). I prefer `rem`.

`em` means "the letter m" - it sets your font to be a size based on the letter m. `font-size: 2em;` sets your font to be the width of two letters m - this is relative to the parent element. So:

```
<div class="large-text">
    <p class="normal-text"> example </p>
</div>
```

```
.large-text {
    font-size: 2em;
}
.normal-text {
    font-size: 1em;
}
```

In this example, `normal-text` is actually the same size as the large text - 2em - because it is inside the `large-text` div.

To avoid this, we use `rem` - which means "relative m". This will always use the default font size of the page, not the parent element.

```
<div class="large-text">
    <p class="normal-text"> example </p>
</div>
```

```
.large-text {
    font-size: 2rem;
}
.normal-text {
    font-size: 1rem;
}
```

In this example the large text is twice as large as normal text, and the normal text is the normal size you expect text to be.

Colours: hex vs rgb/rgba vs words

We usually set colours (`color` , `background-color`) using hex codes. A hex code looks like this: `color: #00ff00;` . It is comprised of hexadecimals (base-16 numbers running from 0 to 9, and through a to f). You can read more about them, but all you need to know is: if you want a colour, use a colour picker (www.google.com/search?q=colour+picker or download the program I use, https://annystudio.com/software/colorpicker/) to get the Hex code.

We can also use rgb. It looks like this: `color: rgb(0, 255, 0);` - there isn't much benefit to using rgb. However, if you want to make an element transparent, you should use `rgba` . It has an extra value; 0 means an item is totally transparent, and 1 means an item is visible. `color: rgb(0, 255, 0, 0.5);` would set the font of an element to be semi-transparent.

Lastly, you can use words - like `red` , `magenta` , or `springgreen` . These aren't very precise and can be quite limited; do not use these.

border

The `border` style has three properties: - type of line (e.g. `solid` , `dashes` , `dotted`) - the thickness of the line (i.e. pixels) - the colour of the border. It is set like this:

```
<div class="box">
</div>
```

```
.box {
    height: 200px;
    width: 200px;
    border: solid 2px #ff9214;
}
```

Font-family

You can set the font family to any font, but either you must provide this font (link it in your HTML head) or you they must have it installed. Some fonts cannot be shown on some systems (that is why you get the weird empty box when people use a character your computer does not know how to show). For that reason, you should always set a default for each browser/system your application may be used on, such as Arial, in case a computer cannot show the font you are using. You can set it like this:

```
<p class="spooky-font"> example text </p>
```

```
.spooky-font {
    font-family: Spooky, "Comic Sans", Arial;
}
```

It will display the first font it can, running left-to-right. Font names with a space (e.g. Comic Sans, Times New Roman) must be written in double quotes.

Comments

Comment in CSS (single line or multi-line) using this syntax:

```
/*
Comment goes here
*/
```

More

You can find a full list of styles here: https://www.w3schools.com/cssref/ - play around with them!