Common HTML Elements

There are many different HTML elements. This can feel overwhelming. The truth is that we don't use *that* many on a daily basis. Here are the common ones.

- **<!DOCTYPE html>** It's important to let the browser know exactly what type of HTML you're writing. The most common, and modern, is `<!DOCTYPE html>`
- **<html>** The first element we need in every document is the `html` element. It's the box that we put everything else into. Inside the `html` element we split our code into hidden information and visual information with the `head` and `body` elements.
- **<head>** Inside the `head` element we add invisible elements and information browsers and search engines use.
- **<title>** We put the `title` element inside the `head` element. It's the page's title and gets displayed in the title of a tab and other useful places.
- **<link>** We put `link` elements in the `head` element too. We can use the element to link CSS files, and add favicons. All you need to do is change the `rel` attribute.
- **<body>** Inside the `body` element we add elements that will translate into something visual on our page.
- ** ** The most basic HTML element is a `div` element. It's like a default box. It doesn't do anything special. By default, it's as high as its contents and as wide as its parent box. And if you put 2 or more divs next to each other in code they'll stack themselves below each other in the browser.
- **<nav> <header> <footer>** Similar to the `div` tag are the `nav`, `header` and `footer` elements. You could use a `div` element instead of each of these and it wouldn't make much difference. But for SEO and search engine reasons, it's good to structure your content into different containers. And it will help you separate your content better. You put navigation links inside the nav element. You put header stuff inside the `header` element. And footer stuff inside the `footer` element.

- **<h1> <h2> <h3> <h4> <h5> <h6>** Heading elements are great for, well, headings! There are 6 types of headings, ranging from the most important heading on a page, the `h1` element, all the way down to an `h6` element. By default, all headings will make the text inside them bold. The `h1` font-size is pretty big, but the font-size will get smaller and smaller the less important the heading is.
- **<p>** Next are paragraphs elements. We use them quite a bit, so thankfully all we need to do is type a `p` for the tag name. Paragraph elements are similar to divs. They are as high as their contents and as wide as their parent element. Any text that is considered part of the page's content will normally be put inside of a paragraph element. The thing about paragraph elements to keep in mind is that when you put them next to each other in code they leave gaps between each other. We'll cover how this works later in the class, but it's good to know from the outset.
- **<blockquote>** Similar to a paragraph element is a `blockquote` element. It's usually used for quotes. And by default, they are as high as their contents and as wide as their parent element.
- **** The next is a `span` element. It's a utility box, mainly used for text. By default, it's as high and wide as its contents, and acts the same as the text around it. By default, you won't be able to tell when a piece of text is inside of a `span` element and when it's not.
- **<b> <i> ** Similar to the span element is the bold, strong, emphasised and italic elements. Bold and strong do similar things: they make the text inside them look bold. Emphasised and italic also do similar things: they make the text inside them look italicised. The tag names are `b`, `strong`, `i` and `em`.
- **<a>** Next up is the anchor element. We type `a` for the tag name. Most of the world calls this a link. It's one of the most important elements in the HTML language. It's how we get from page to page. And the way we do this is by giving it an attribute of `href` and specifying an address relative to the current address or by specifying an absolute address which is characterised by a full website domain name at the beginning, complete with the `http://` or `https://` part. Along with this we can tell the browser how to open this address by giving it an attribute of `target`. If we want to open it in a new tab or window we can give the attribute a value of `_blank`. The default is to open the new address in the current window. To explicitly

set this, give the `target` attribute a value of `_self`. And you can jump to another part of the same page by specifying the `id` attribute value of the element you want to jump to, prefixed by a hash. You'll need to make sure that there is an element with an `id` attribute with that value, otherwise nothing will happen when the user clicks on the link. Another attribute we can give an anchor element is the `title` attribute. When a user hovers over the anchor element this piece of text will show itself. It's quite useful for providing more detail about what will happen when they click the anchor element. It's only really useful when the user has the ability to hover. By default, an anchor element is an element that is only as high and wide as its contents. And if you put 2 or more anchors next to each other in code they'll sit next to each other in the browser and act as if they were normal text.

- ** ** In HTML there is an easy way to create bulleted lists and numbered lists – without you having to put the numbers in! If you want a list without numbers you use an unordered list element. We type `ul` for the tag name. And if you want an ordered list element, type `ol` for the tag name. Now, whether you've got an ordered or an unordered list, we put list item elements inside them. Each item in a list must go inside a list item element. We type `li` for the tag name. Ordered, unordered and list item elements are all as high as their contents and as wide as their parent element.
- **<img>** The image element acts like text, and by default, is as high and as wide as the image it is displaying. And if you put an image next to text in code, it will appear next to it in the browser. We type `img` for the tag name and can use the `src` attribute to tell the image element what file to display. This works exactly the same as the anchor tag's `href` attribute. We can use a relative address or an absolute address to tell the image element where to find the file to display. The other attribute you can use on image elements is the `alt` attribute. This tells the browser and SEO spiders what this image is about. And if the image doesn't load for some reason this piece of text will show up. The weird thing about an image element is that it doesn't have a closing tag. You write the opening tag and that's it! Because you can't put anything inside it.
- **
** The next special element is a line break element. We type `br` for the tag name and it also only has an opening tag. In HTML if you write more than one space between 2 pieces of text the browser treats it as a single

space. If you hit return, and again, and again, and again, it will also treat this is a single space. So what does this mean? It means browsers treat whitespace in HTML, however big, as a single space. This is where the line break element comes in. You can force text onto a new line by putting a line break element in amongst the text.

- **<iframe>** A really cool element is the `iframe` element. You can display a whole other webpage in this element. You'll often use an `iframe` element for embedding videos. `iframe` elements act similar to images. They'll sit next to text and image alike. The only thing that's different is that you have to tell them how wide and high to be. One way to do this is to use `width` and `height` attributes. The other is by styling them.
- **<!-- comments -->** The final thing this lesson covers is HTML comments. Comments are notes to leave yourself or others. And you can comment out code you don't want the browser to interpret and display. Browsers don't understand HTML comments. You may find using comments helpful during this class. You can write an HTML comment by beginning with `<!--`, adding whatever comment you want, and ending with `-->`. Most code editors will have a keyboard shortcut for this.