Revision

What Are HTML & CSS?

HTML, HyperText Markup Language, gives content structure and meaning by defining that content as, for example, headings, paragraphs, or images. *CSS*, or Cascading Style Sheets, is a presentation language created to style the appearance of content—using, for example, fonts or colours.

The two languages—HTML and CSS—are independent of one another and should remain that way. CSS should not be written inside of an HTML document and vice versa. As a rule, HTML will always represent content, and CSS will always represent the appearance of that content.

Understanding Common HTML Terms

The three common HTML terms you should begin with are *elements*, *tags*, and *attributes*.

Elements

Elements are designators that define the structure and content of objects within a page. Some of the more frequently used elements include multiple levels of headings (identified as `<h1>` through `<h6>` elements) and paragraphs (identified as the `<p>`element); the list goes on to include the `<a>`, `<div>`, `<span>`, `<strong>`, and `<em>`elements, and many more.

Elements are identified by the use of less-than and greater-than angle brackets, `< >`, surrounding the element name. Thus, an element will look like the following:

```
<a>

```

Tags

The use of less-than and greater-than angle brackets surrounding an element creates what is known as a *tag*. Tags most commonly occur in pairs of opening and closing tags.

An *opening tag* marks the beginning of an element. It consists of a less-than sign followed by an element's name, and then ends with a greater-than sign; for example,`<div>`.

A *closing tag* marks the end of an element. It consists of a less-than sign followed by a forward slash and the element's name, and then ends with a greater-than sign; for example, `</div>`.

The content that falls between the opening and closing tags is the content of that element. An anchor link, for example, will have an opening tag of `<a>` and a closing tag of `</a>`. What falls between these two tags will be the content of the anchor link.

So, anchor tags will look a bit like this:

```
<a>...</a>

```

Attributes

Attributes are properties used to provide additional information about an element. The most common attributes include the `id` attribute, which identifies an element; the `class` attribute, which classifies an element; the `src` attribute, which specifies a source for embeddable content; and the `href` attribute, which provides a hyperlink reference to a linked resource.

Attributes are defined within the opening tag, after an element's name. Generally attributes include a name and a value. The format for these attributes consists of the attribute name followed by an equals sign and then a quoted attribute value. For example, an `<a>` element including an `href` attribute would look like the following:

```
<a href="http://tipsforyourwebsite.com/">Go to TipsForYourWebsite.com</a>

```

Common HTML Terms Demo

The preceding code will display the text "`my website`" on the web page and will take users to http:// tipsforyourwebsite.com/ upon clicking the "`Go to TipsForYourWebsite.com`" text. The anchor element is declared with the opening `<a>` and closing `</a>` tags encompassing the text, and the hyperlink reference attribute and value are declared with `href="http://mywebsite.com"` in the opening tag.

Fig 1
HTML syntax outline including an element, attribute, and tag

Now that you know what HTML elements, tags, and attributes are, let's take a look at putting together our first web page.

Setting Up the HTML Document Structure

HTML documents are plain text documents saved with an `.html` file extension rather than a `.txt` file extension. To begin writing HTML, you first need a plain text editor that you are comfortable using. Two of the more popular plain text editors for writing HTML and CSS are Dreamweaver (expensive) and Notepad++ (free).

All HTML documents have a required structure that includes the following declaration and elements: `<!DOCTYPE html>`, `<html>`, `<head>`, and `<body>`.

The document type declaration, or `<!DOCTYPE html>`, informs web browsers which version of HTML is being used and is placed at the very beginning of the HTML document. Because we'll be using the latest version of HTML, our document type declaration is simply `<!DOCTYPE html>`. Following the document type declaration, the`<html>` element signifies the beginning of the document.

Inside the `<html>` element, the `<head>` element identifies the top of the document, including any metadata (accompanying information about the page). The content inside the `<head>` element is not displayed on the web page itself. Instead, it may include the document title (which is displayed on the title bar in the browser window), links to any external files, or any other beneficial metadata.

All of the visible content within the web page will fall within the `<body>` element. A breakdown of a typical HTML document structure looks like this:

```
1 <!DOCTYPE html>
2 <html>
```

```
  <head>
        <title>Welcome</title>
  </head>
  <body>
    <h1>Hello World</h1>
    <p>This is a web page.</p>
  </body>
</html>


```

HTML Document Structure Demo

The preceding code shows the document beginning with the document type declaration, `<!DOCTYPE html>`, followed directly by the `<html>` element. Inside the`<html>` element come the `<head>` and `<body>` elements. The `<head>` element includes the title of the document via the `<title>` element.

The `<body>` element includes a heading via the `<h1>` element and a paragraph via the `<p>` element. Because both the heading and paragraph are nested within the `<body>` element, they are visible on the web page.

When an element is placed inside of another element, also known as nested, it is a good idea to indent that element to keep the document structure well organized and legible. In the previous code, both the `<head>` and `<body>` elements were nested—and indented—inside the `<html>` element. The pattern of indenting for elements continues as new elements are added inside the `<head>` and `<body>` elements.

Self-Closing Elements

In the previous example, the `<meta>` element had only one tag and didn't include a closing tag. Fear not, this was intentional. Not all elements consist of opening and closing tags. Some elements simply receive their content or behavior from attributes within a single tag. The `<meta>` element is one of these elements. The content of the previous `<meta>` element is assigned with the use of the charset attribute and value. Other common self-closing elements include

- `<br> or <br />`
- `<hr> or <hr />`
- `<img> or <img />`
- `<link> or <link />`

- `<meta> or <meta />`

The structure outlined here, making use of the `<!DOCTYPE html>` document type and`<html>`, `<head>`, and `<body>` elements, is quite common. We'll want to keep this document structure handy, as we'll be using it often as we create new HTML documents.

Code Validation

No matter how careful we are when writing our code, we will inevitably make mistakes. Thankfully, when writing HTML and CSS we have validators to check our work. The W3C has built both HTML and CSS validators that will scan code for mistakes. Validating our code not only helps it render properly across all browsers, but also helps teach us the best practices for writing code.

In Practice

1. Let's open our text editor, create a new file named `index.html`, and save it to a location we won't forget. I'm going to create a folder on my Desktop named "styles-conference" and save this file there; feel free to do the same.
2. Within the index.html file, let's add the document structure, including the`<!DOCTYPE html>` document type and the `<html>`, `<head>`, and `<body>` elements.

```
<!DOCTYPE html>
<html>
  <meta charset="utf-8" />
  <head>
  </head>
  <body>
  </body>
</html>
```

3. Inside the `<head>` element, let's add `<title>` elements. The `<title>`element should contain the title of the page—let's say "Styles Conference."

```
<head>
  <title>Welcome</title>
</head>
```

4. Inside the `<body>` element, let's add `<h1>` and `<p>` elements. The `<h1>` element should include the heading we wish to include—let's use "Styles Conference" again—and the `<p>` element should include a simple paragraph to introduce our conference.

```
<body>
  <h1>Certificate 3 Web</h1>
  <p>Glad to see you all back!</p>
</body>

```

Building Structure

For the longest time the structure of a web page was built using divisions. The problem was that divisions provide no semantic value, and it was fairly difficult to determine the intention of these divisions. Fortunately HTML5 introduced new structurally based elements, including the `<header>`, `<nav>`, `<article>`, `<section>`, `<aside>`, and`<footer>` elements.

All of these new elements are intended to give meaning to the organization of our pages and improve our structural semantics. They are all block-level elements and do not have any implied position or style. Additionally, all of these elements may be used multiple times per page, so long as each use reflects the proper semantic meaning.

Let's take a closer look.

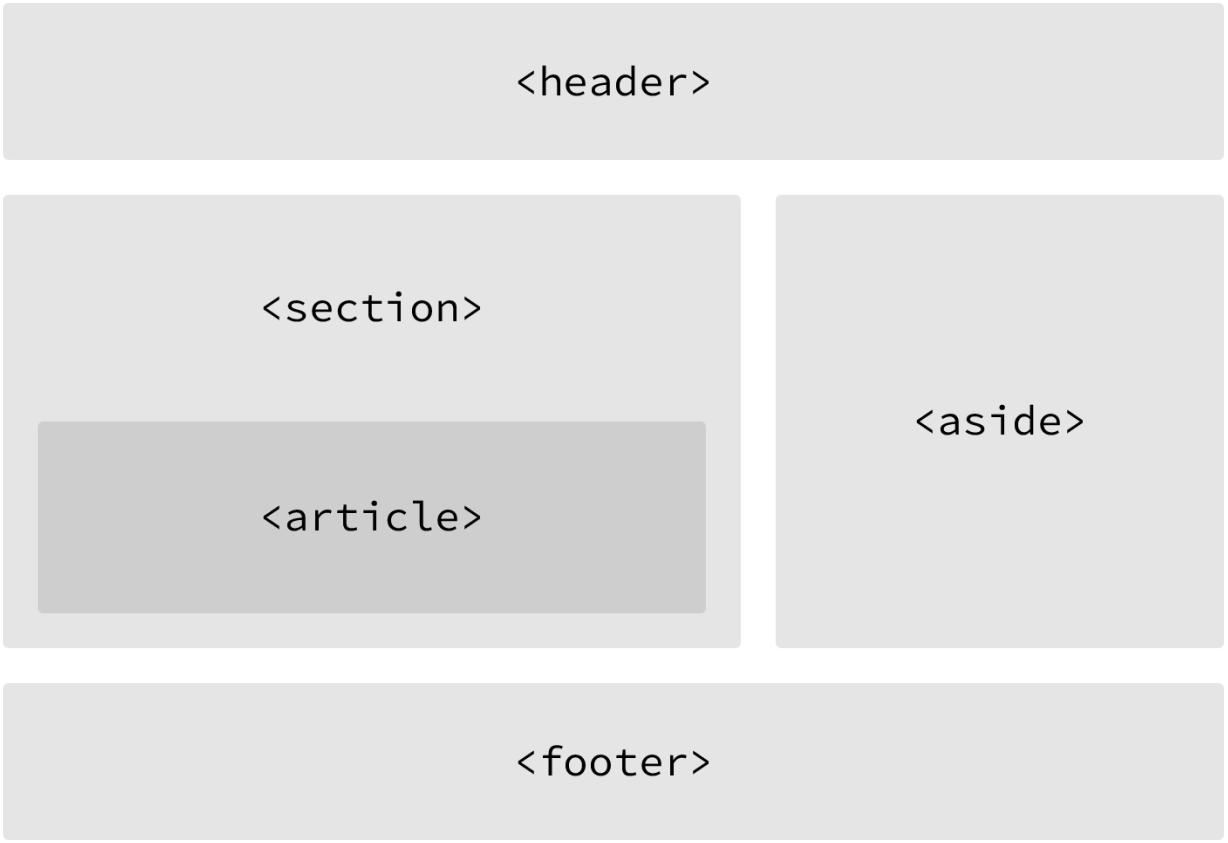

Fig 2

One possible example of HTML5 structural elements giving meaning to the organization of our pages

Header

The `<header>` element, like it sounds, is used to identify the top of a page, article, section, or other segment of a page. In general, the `<header>` element may include a heading, introductory text, and even navigation.

```
<header>...</header>

```

`<header>` vs. `<head>` vs. `<h1>` through `<h6>` Elements

It is easy to confuse the `<header>` element with the `<head>` element or the heading elements, `<h1>` through `<h6>`. They all have different semantic meanings and should be used according to their meanings. For reference...

The `<header>` element is a structural element that outlines the heading of a segment of a page. It falls within the `<body>` element.

The <head> element is not displayed on a page and is used to outline metadata, including the document title, and links to external files. It falls directly within the <html> element.

Heading elements, <h1> through <h6>, are used to designate multiple levels of text headings throughout a page.

Navigation

The <nav> element identifies a section of major navigational links on a page.
The <nav>element should be reserved for primary navigation sections only, such as global navigation, a table of contents, previous/next links, or other noteworthy groups of navigational links.

Most commonly, links included within the <nav> element will link to other pages within the same website or to parts of the same web page. Miscellaneous one-off links should not be wrapped within the <nav> element; they should use the anchor element, <a>, and the anchor element alone.

```
<nav>...</nav>

```

Article

The <article> element is used to identify a section of independent, self-contained content that may be independently distributed or reused. We'll often use the <article>element to mark up blog posts, newspaper articles, user-submitted content, and the like.

When deciding whether to use the <article> element, we must determine if the content within the element could be replicated elsewhere without any confusion. If the content within the <article> element were removed from the context of the page and placed, for example, within an email or printed work, that content should still make sense.

```
<article>...</article>

```

Section

The <section> element is used to identify a thematic grouping of content, which generally, but not always, includes a heading. The grouping of content within the<section> element may be generic in nature, but it's useful to identify all of the content as related.

The <section> element is commonly used to break up and provide hierarchy to a page.

```
<section>...</section>

```

Deciding Between <article>, <section>, or Elements

At times it becomes fairly difficult to decide which element—<article>, <section>, or —is the best element for the job based on its semantic meaning. The trick here, as with every semantic decision, is to look at the content.

Both the <article> and <section> elements contribute to a document's structure and help to outline a document. If the content is being grouped solely for styling purposes and doesn't provide value to the outline of a document, use the element.

If the content adds to the document outline and it can be independently redistributed or syndicated, use the <article> element.

If the content adds to the document outline and represents a thematic group of content, use the <section> element.

Aside

The <aside> element holds content, such as sidebars, inserts, or brief explanations, that is tangentially related to the content surrounding it. When used within an<article> element, for example, the <aside> element may identify content related to the author of the article.

We may instinctively think of an <aside> element as an element that appears off to the left or right side of a page. We have to remember, though, that all of the structural elements, including the <aside> element, are block-level elements and as such will appear on a new line, occupying the full available width of the page or of the element they are nested within, also known as their parent element.

```
<aside>...</aside>

```

Footer

The `<footer>` element identifies the closing or end of a page, article, section, or other segment of a page. Generally the `<footer>` element is found at the bottom of its parent. Content within the `<footer>` element should be relative information and should not diverge from the document or section it is included within.

```
<footer>...</footer>

```

With structural elements and text-based elements under our belts, our HTML knowledge is really starting to come together. Now is a good time to revisit our Styles Conference website and see if we can provide it with a little better structure.

In Practice

Currently, our practice example lacks real structure—and content for that matter.

1. Using our existing index.html file, let's add in a `<header>` element. Our `<header>`element should include our existing `<h1>` element; let's also add an `<h3>` element as a tagline to support our `<h1>` element.

```
<header>
  <h1>Certificate 3 Web</h1>
  <h2>Lismore TAFE Sem 2 2016</h2>
</header>

```

1. After our `<header>` element, let's add a new group of content, using the`<section>` element, that introduces our conference. We'll begin this section with a new `<h2>` element and end it with our existing paragraph.

```
<section>
  <h2>Dedicated to the Craft of Building Websites</h2>
  <p>From this course you could/should go onto the Cert 4!</p>


</section>

```

2. Now let's add another group of content that teases a few of the pages we'll be adding, specifically the **Topics** and **Teachers** pages. Each of the pages we're teasing should also reside within its own section and include supporting text.

   We'll group all of the teasers inside a `<section>` element, and each individual teaser will be wrapped within a `<article>` element. In all, we'll have two`<article>` elements inside a `<section>` element, which is fine.

```
<section>
….
  <article>
    <h2>Topics</h2>
     <p>Topic1.</p>
  </article>
  <article>
    <h2>Teachers</h2>
     <p>Teacher 1</p>
  </article>

</section>
```

3. Next we will add a sidebar for the timetable. The aside element can go inside a section or article or div depending on what the content of the aside relates to so in this instance it will go inside the section but after the article elements

```
<section>
 ……
  <article>
    <h2>Topics</h2>
     <p>Topic1.</p>
  </article>
  <article>
    <h2>Teachers</h2>
     <p>Teacher 1</p>
  </article>

  <aside>
    <h2>Timetable</h2>
     <p>Monday</p>
  </aside>

</section>
```

4. Lastly, let's add our copyright within the `<footer>` element at the end of our page. To do so let's use the `<small>` element, which semantically represents side comments and small print—perfect for our copyright.

   Generally, content within the `<small>` element will be rendered as, well, small, but our CSS reset will prevent that from happening.

```
<footer>
  <small>&copy; Your Name 2017</small>
</footer>

```

Now we can see our home page beginning to come to life.

Encoding Special Characters

The `<small>`element within our `<footer>` element, has some interesting things going on. Specifically, a few special characters within these elements are being encoded.

Special characters include various punctuation marks, accented letters, and symbols. When typed directly into HTML, they can be misunderstood or mistaken for the wrong character; thus they need to be encoded.

Each encoded character will begin with an ampersand, `&`, and end with a semicolon, `;`. What falls between the ampersand and semicolon is a character's unique encoding, be it a name or numeric encoding.

For example, we would encode the word "resumé" as `resum&eacute;`. Within our header we have encoded both en and em dashes, and within our footer we have encoded the copyright symbol. For reference, a long list of character encodings may be found at Copy Paste Character.

With our home page taking shape, let's take a look at creating hyperlinks so that we may add additional pages and build out the rest of our website.

Creating Hyperlinks

Along with text, one of the core components of the Internet is the hyperlink, which provides the ability to link from one web page or resource to another. Hyperlinks are established using the anchor, `<a>`, inline-level element. In order to create a link from one

page (or resource) to another, the `href` attribute, known as a hyperlink reference, is required. The `href` attribute value identifies the destination of the link.

For example, clicking the text "Google," which is wrapped inside the anchor element with the `href` attribute value of `http://google.com`, will take users to Google.

```
<a href="http://google.com">Google</a>

```

Relative & Absolute Paths

The two most common types of links are links to *other pages* of the same website and links to *other websites*. These links are identified by their `href` attribute values, also known as their paths.

Links pointing to other pages of the same website will have a *relative path*, which does not include the domain (.com, .org, .edu, etc.) in the `href` attribute value. Because the link is pointing to another page on the same website, the `href` attribute value needs to include only the filename of the page being linked to: `about.html`, for example.

Should the page being linked to reside within a different directory, or folder, the `href` attribute value needs to reflect this as well. Say the `about.html` page resides within the `pages` directory; the relative path would then be `pages/about.html`.

Linking to other websites outside of the current one requires an *absolute path*, where the `href` attribute value must include the full domain. A link to Google would need the `href` attribute value of `http://google.com`, starting with http and including the domain, `.com` in this case.

Here clicking on the text "About" will open the `about.html` page inside our browser. Clicking the text "Google," on the other hand, will open `http://google.com/` within our browser.

```
<!-- Relative Path -->
<a href="about.html">About</a>

<!-- Absolute Path -->
<a href="http://www.google.com/">Google</a>

```

Linking to an Email Address

Occasionally we may want to create a hyperlink to our email address—for example, hyperlink text that says "Email Me," which when clicked opens a user's default email client and pre-populates part of an email. At a minimum the email address to which the email is being sent is populated; other information such as a subject line and body text may also be included.

To create an email link, the `href` attribute value needs to start with `mailto:` followed by the email address to which the email should be sent. To create an email link to `info@awesome.com`, for example, the `href` attribute value would be`mailto:info@awesome.com`.

Here's the full breakdown:

```
<a href="mailto:info@awesome.com">Email Me</a>
```

Opening Links in a New Window

One feature available with hyperlinks is the ability to determine where a link opens when clicked. Typically, links open in the same window from which they are clicked; however, links may also be opened in new windows.

To trigger the action of opening a link in a new window, use the `target` attribute with a value of `_blank`. The `target` attribute determines exactly where the link will be displayed, and the `_blank` value specifies a new window.

To open `http://google.com/` in a new window, the code would look like this:

```
<a href="http://google.com/" target="_blank">Google</a>
```

In Practice

It's time to take our exercise from a single-page website to a full-blown website with multiple pages, all of which will be linked together using hyperlinks.

1. We'll begin by making our "Certificate 3 Web" text inside the `<h1>` element within our `<header>` element link to the `index.html` page.

   Because we are already on the `index.html` page, this may seem a little odd—and rightfully so—but as the header will be replicated on other pages, linking back to the home page will make sense.

```
<h1>
  <a href="index.html"> Certificate 3 Web </a>
</h1>

```

2. In order to navigate across all of the different pages, we're going add in a navigation menu, using the `<nav>` element, within our `<header>` element. We'll be creating teachers, topics, and Register pages to go with our home page, so we should create links for all of them.

```
<header>

  ...

  <nav>
    <a href="index.html">Home</a>
    <a href="teachers.html">Teachers</a>
    <a href="topics.html">Topics</a>
     <a href="contact.html">Contact</a>
  </nav>

</header>

```

3. Let's also add the same navigation menu from our `<header>` element to our`<footer>` element for convenience.

```
<footer>

  ...

    <nav>
    <a href="index.html">Home</a>
    <a href="teachers.html">Teachers</a>
    <a href="topics.html">Topics</a>
     <a href="contact.html">Contact</a>
  </nav>
</footer>
```

```
12
13
14
```

4. We can't forget to add links to all of the sections teasing our other pages. Inside each section, let's wrap the `<h2>` elements within an anchor element linking to the proper page.

   We'll want to make sure we do this for every article accordingly.

```
<section>


   <article>
   <a href="topics.html">
     <h2>Topics</h2>
   </a>
    <p>Topic1.</p>
 </article>

 ...

</section>

```

5. # Independent challenges

   -Add the text "READ MORE" to the bottom of each article and make it a link to the appropriate page
   -Add the rest of days of the week to the aside onto their own lines
   -Add the text "teacher2" and "teacher 3" below "teacher 1" on their own lines
   -Add the text "email me" to the end of the navigation links and make it a mailto link
   -there are also text presentation tags such as   for emphasis and   for bold text. Use them on your page for at least 2 words

Summary

Semantics, as discussed within this lesson, are essential for providing our HTML with structure and meaning. Moving forward we'll periodically introduce new elements, all of which will come with their own semantic meaning. It is the meaning of all of these elements that will provide our content with the most value.

Next up, we'll take at CSS and "presenting" our html content.