HTML

Structuring Your First Website

Classroom Course Manual

Written, designed, and produced by:
DoIT Software Training for Students

Last Updated 9/9/2016
About Software Training for Students

Software Training for Students is an organization on campus that provides free software training to all students and faculty. Our services include custom workshops, open-enrollment classes, one-on-one project help, and access to Lynda.com. For more information on the Software Training for Students (STS) program, visit our website at wisc.edu/sts.

STS is part of the Division of Information Technology (DoIT) - Academic Technology at UW-Madison. For more information regarding DoIT Academic Technology, visit at.doit.wisc.edu.

© University of Wisconsin Board of Regents.

This manual and any accompanying files were developed for use by current students at the University of Wisconsin-Madison. The names of software products referred to in these materials are claimed as trademarks of their respective companies or trademark holder.

If you are not a current member of the UW-Madison community and would like to use STS materials for self-study or to teach others, please contact sts@doit.wisc.edu. Thank you.
Topics Outline

1. Introduction
2. Web Technology Overview
3. Basic HTML Syntax
4. Getting Started
5. Document Setup
6. Formatting Text
7. Document Structure
8. Lists and Tables
9. Links
10. Pages
11. Images
12. Classes and IDs
13. CSS Preview
14. Quick Reference
Introduction

What is HTML?

HTML (Hyper Text Markup Language) is a formatting and markup language that is used to create webpages. HTML is a cornerstone technology on the web; it is the foundation practically every webpage and website on the internet.

The language works by labelling different pieces of content on your webpage with tags. When a web browser opens the webpage, it is able to interpret and differentiate between each piece of content, and display it correctly on the page.

A Brief Warning

If you are looking to build a fully functioning website from the ground up, then you will need to learn three key web development languages. These languages include HTML, CSS, and JavaScript. HTML serves as the starting point for building your own website, but a completed website will take time and practice with all three core languages.

If you wish to put together a website without writing code and starting from scratch, then there are plenty of services like Wordpress and Squarespace that allow you to do so. For most people, these services are sufficient, and may be the better option.

Prerequisites and Requirements

- Basic computer operation skills
- General familiarity with the internet and web browsers

Web Technology Overview

Before we dive in to working with HTML, it's important to understand the fundamentals of how the web works. So, what happens when you type a website’s URL into a web browser? Well, there are many steps that take place, but
Here's a general idea:

1. You type a website's URL into the browser.
2. The browser uses the internet to look up the IP address of the server where the website is hosted.
3. The server then transfers the requested data back through the internet to your web browser.
4. Your web browser then interprets the data and displays it to the screen.

To summarize, when you visit a website, your computer connects to a physical server somewhere in the world using the internet. It then downloads a series of files (HTML, CSS, and JavaScript files) to your computer, and displays them on screen.

**Terminology**

**The Internet**

A system of interconnected computer networks used to allow connections between any two computers in the world. These networks use standardized communication protocols to facilitate the exchange of data between millions of computers all around the world.

**Server**

A server is a computer that is used to hold a website's files and deliver (serve) these files to web browsers when the files are requested. Servers can run almost any operating system (e.g., Windows, Mac, Linux) and have innumerable features to tweak performance of the webpages they stores. They are often fast and expensive, but in reality, any computer can be used as a web server.

Most people, however, don't use their own computers as web servers. Instead, they use services (which can be found online) to host their websites. Some of these services are free to use, while others require a monthly hosting fee. The main benefit of using these services is that they're generally fast, affordable, and easy to set up while requiring little-to-no maintenance on your part.

**Web Browser**
A web browser (or browser for short) is a software application that takes the raw data from the Internet and converts it into a readable, graphic display (a process called rendering). Web browsers can also take input from the user and transmit it back to the server.

Common web browsers include Chrome, Firefox, Internet Explorer, Opera, and Safari. Although these browsers all serve the same purpose, they do have their differences, which can cause data to be displayed in unexpected ways. Throughout this series of courses, (HTML, CSS, etc.) we will be addressing some of these cross-browser issues and exploring a few ways to get around them.

Webpage and Website

A webpage is an individual page and/or document of information on the web. Meanwhile, a website is a collection of related web pages, images, videos, etc.

Basic HTML Syntax

Tags and Elements

As a markup language, HTML uses markup tags to describe the contents on the page. These tags can be thought of as key words or descriptors, surrounded by angled brackets. Generally, a tag consists of three parts:

1. Start tag (also called the opening tag)
2. Element content, which is everything between the opening and closing tags
3. End tag (also called the closing tag)

All three of these parts combined form what’s known as an HTML element.

Commonly Used Tags

Below is a table containing a few common content tags in HTML.
<table>
<thead>
<tr>
<th>Opening Tag</th>
<th>Closing Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;h1&gt;</code> to <code>&lt;/h6&gt;</code></td>
<td><code>&lt;h1&gt;</code> to <code>&lt;/h6&gt;</code></td>
<td>Headings and subheadings. H1 is the main heading, H2 is the secondary heading, etc.</td>
</tr>
<tr>
<td><code>&lt;p&gt;</code></td>
<td><code>&lt;p&gt;</code></td>
<td>Paragraph text.</td>
</tr>
<tr>
<td><code>&lt;a&gt;</code></td>
<td><code>&lt;a&gt;</code></td>
<td>Anchor, or link to another webpage or file on the web.</td>
</tr>
<tr>
<td><code>&lt;ol&gt;</code></td>
<td><code>&lt;ol&gt;</code></td>
<td>An ordered list with numbers.</td>
</tr>
<tr>
<td><code>&lt;ul&gt;</code></td>
<td><code>&lt;ul&gt;</code></td>
<td>An unordered list with bullet points.</td>
</tr>
<tr>
<td><code>&lt;li&gt;</code></td>
<td><code>&lt;li&gt;</code></td>
<td>A list item nested inside an element such as <code>&lt;ol&gt;</code> or <code>&lt;ul&gt;</code>.</td>
</tr>
</tbody>
</table>

**Attributes and Values**

Within the opening tag of each HTML element, we can assign specific attributes and values, with the structure seen below. Each HTML element can have a vast variety of attributes assigned to it. We will specify their capabilities at a later point.

```
<tag class="class-name"> content </tag>
```

**Syntax Rules**

As with most computer languages, the structure and formatting must be strictly adhered to in order for the computer to interpret it properly. In HTML, there are a few basic guidelines you can follow to make sure all of your code has proper formatting:

**All tags must be closed.**

- Good example: `<p> some content </p>`
- Bad example: `<p> some content`  

**Tags must be completely enclosed within each other.**

- Good example: `<p><a> some content </a></p>`
- Bad example: `<p><a> some content </a></p>`

**All tags and attributes must be in lower case.**

- Good example: `<p attribute = "value"> some content </p>`
- Bad example: `<p ATTRIBUTE = "value"> some content </p>`

**All attribute values must be in quotes.**

- Good example: `<p attribute = "value"> some content </p>`
Getting Started

Root Folder and Site Organization

When creating a website, it is important to stay organized. A disorganized site can quickly become a burden to update and manage. Although everyone has their own organizational methods, the following is highly recommended.

A website is a combination of images, videos, HTML/CSS/JavaScript files, and much more. Every website has what's called a **root folder**, or a directory containing all of the content that builds the website.

Before moving on to the next section, create a root folder somewhere accessible on your computer to hold all of your website files.

![website-folder]

Inside of this folder, create two more folders. Name the first one *images*, and the second one *resources*. The *images* folder will be used for all pictures found on the website, meanwhile the *resources* folder will contain all of our CSS and JavaScript.

Again, this structure is optional, and only a recommendation. If you find a different organizational method to work better, then use it.

![images-resources]

Creating Our First File

Since HTML is a computer language, all you need to start writing is a simple text editor, such as Brackets, TextWrangler, TextEdit, or Notepad. Once you've written the code, you'll need to save the file as a `.html` file, so it can be opened and viewed by a web browser.

We will begin by setting up a workspace in which we can edit and preview our code side-by-side. This process should be similar on Mac and Windows machines alike.

1. Open a blank document in a text editor of your choice. In our examples, we will be using a program called TextWrangler, which you can find for free online.

2. In the menu bar at the top of the screen, select File > Save As. Name the file *index.html* then save it directly inside your website's root folder. Your root folder should now contain the following:
Congratulations! You now have your very first HTML file.

**Naming Conventions**

You may have noticed that we named our file `index.html`. But why this name? Well, there are a few reasons:

1. When your web browser downloads your website files from a server, it immediately searches for the first file to open and display. By default, your web browser will look for a file named “index”. And so, it is common practice to have your “index” file contain the homepage of your website.

2. The `.html` extension is necessary for the computer to recognize that the file is an HTML document. In fact, almost every file on your computer has some sort of extension signifying what kind of file it is. For example, pictures typically have a `.jpg` extension, Word documents have a `.docx` extension. etc.

Throughout the process of creating a website, you will accumulate many files within the root folder of your website. It's important that you name your files properly.

When naming files, here are a few things you may want to avoid:

- Any sort of strange symbol that is not a letter, dash, or underscore. For example: `wisc.edu/my|dog's#1*fan!.html`.
- Overly long or complex names.
- Periods that don't denote the file extension: `wisc.edu/my.files/first.webpage.html`.
- Spaces within your filename.

Most of these symbols will work if you use them in file names, but in general, they're annoying and will cause you pain at some point in the future. They are easily avoidable, so avoid them.

**Setting up a Workspace**

To edit your HTML file, open it with a text editor. To preview your HTML file, open it in a web browser. To open a file using a specific program, right-click on the file, then select **Open With**. This will allow you to choose whether you want to open the HTML file in a web browser or a text editor.

We recommend that you open the text editor and the web browser side-by-side. This way, you can edit the HTML, save the document, then refresh the web browser to see how the actual webpage looks.
Now that our website's root folder is ready, and we have created our first HTML file, it's time to begin writing a webpage. Every webpage follows a specific outline, or document structure in order for it to be read properly. In this next exercise we will go through step by step, explaining and writing out the document structure necessary for our webpage.

1. In your text editor, begin by writing the following line at the top of your document:

```html
<!DOCTYPE html>
```

This statement is a declaration describing what type of document we are writing. When the web browser opens our file, it will see this statement and understand to read it as a regular HTML file.

2. We will now need to mark the beginning and end of our code in the document. We can do this by enclosing all of our code within the `<html>` tag. The browser will not read any HTML code written outside this tag.

```html
<!DOCTYPE html>
<html>
<body>
</body>
</html>
```

3. Inside of our `<html>` tags, there are two more sections we would like to add:
   1. **Head**: The head contains information about the webpage that is usually not displayed. It is often used for linking external scripts and stylesheets, adding metadata, and changing the title and description of the webpage.
   2. **Body**: The body holds all of the information that is displayed on the webpage. Any content that is written outside of the body tags will not show up in the browser window.

To add these sections into our document, we can use the `<head>` and `<body>` tags.

```html
<!DOCTYPE html>
<html>
  <head>
    <title></title>
  </head>
  <body>
  </body>
</html>
```

4. The final piece we will add to our document structure is a `<title>` tag within the head of our document. This will set the title of our webpage.
Congratulations! You have officially structured your first HTML document. Now we are ready to add some content.

### Formatting Text

Earlier in this manual, we previewed a series of basic HTML tags to add content to our page. In this section, we will utilize these tags to begin building our webpage.

#### Paragraph Text

The most essential element we can add to our page is basic text. In order to add paragraph text, we can enclose it within the `<p>` tag we saw earlier.

1. In the body of the document, add some text enclosed by two `<p>` tags. (Remember that `<p>` acts as the opening tag, and `</p>` acts as the closing tag.)

```html
<body>
  <p>Hello world!</p>
</body>
```

2. Save your HTML file, then refresh your browser to see the result. You should now see "Hello world!" displayed in your web browser.

![Hello world!](image)

3. Each `<p>` element in HTML will act as its own paragraph, and space itself accordingly. Continue to add paragraph text to your document, until you have a significant amount of content. If you need filler text to fill the page, look to the file called `sample-text.txt` in the class files.

### Headings
Now that we've added some basic content to our page, we want to title and properly mark each section. In HTML we can use the heading tags to create headings and subheadings, for different sections and sub-sections of our website.

There are six different levels of headings we can use, ranging from `<h1>` to `<h6>`. The first three are previewed to the right.

Headings are used to establish hierarchy within a webpage. For example, an `<h1>` tag may mark the beginning of a section of content, then a series of `<h2>` tags may be used to mark sub-sections.

```html
<h1>Section Title</h1>
<h2>Sub-section Title</h2>
<p>Paragraph text here.</p>
<h2>Sub-section Title</h2>
<p>Paragraph text here.</p>
```

Take some time to add a title to the top of your webpage, then add titles for each sub-section of content on your page. When finished, your HTML document should look similar to the following. (Note, the paragraph text is cut off in the preview below.)

```html
<!DOCTYPE html>
<html>
<head>
  <title>Fuzzy and the Blue Tones</title>
</head>
<body>
  <h1>Fuzzy and the Blue Tones</h1>
  <h2>History</h2>
  <p>Fuzzy and the Blue Tones was founded in 2003 by banjo aficionado Bob Fonz.</p>
  <h2>Unbelievable Success</h2>
  <p>In 2004, the band released its first single ever, 'Illegally Fuzzy. Only'</p>
  <h2>Top Songs</h2>
  <h2>Sponsors</h2>
</body>
</html>
```

**Bold, Italics, and Line Breaks**

In order to make a specific word, sentence, or paragraph bold, you can surround it with a `<strong>` tag. Likewise, to italicize your text, you may surround it with the `<em>` tag. An example of each are given below.

```html
<p>This word is <strong>bold</strong>.</p>
<p>This word is <em>italisized</em>.</p>
```

Within a paragraph of text, you may also add a line break. To do so, you may use the `<br/>` tag. This tag is unique, because it does not have an opening or a closing. Instead, it is a singular tag that will break text to a new line wherever it is placed. An example of its use is demonstrated below.
Document Structure

With the introduction of HTML5, we received new tags that make it easy to divide our webpage into sections in order to stay more organized and give additional meaning to certain parts of the page. These tags allow us to structure the document, also making it easier for us down the road when we interact with our webpage using CSS and JavaScript.

Below is a list of commonly used document structure tags. To find a more comprehensive list, visit the quick reference guide at the end of this manual.

<table>
<thead>
<tr>
<th>Opening Tag</th>
<th>Closing Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;header&gt;</td>
<td>&lt;/header&gt;</td>
<td>Contains introductory content for a webpage (e.g. banner, navigation), or a section of a page.</td>
</tr>
<tr>
<td>&lt;nav&gt;</td>
<td>&lt;/nav&gt;</td>
<td>Contains navigation bar, usually linking to other pages.</td>
</tr>
<tr>
<td>&lt;aside&gt;</td>
<td>&lt;/aside&gt;</td>
<td>Contains content that is tangentially related to the main content of the page, as one would find in a sidebar.</td>
</tr>
<tr>
<td>&lt;footer&gt;</td>
<td>&lt;/footer&gt;</td>
<td>Contains footer of a page. Typically the footer contains information about the content, such as the author and a copyright statement.</td>
</tr>
</tbody>
</table>

1. At the top of your webpage, there should be an <h1> element that acts as the page title. Surround this element in a <header> container.

2. Inside of the header, add a <nav> container. Keep it empty for now, we will utilize it later.

3. Add a <footer> container at the bottom of the body. Feel free to write in the footer with whatever content you wish. We will be utilizing the footer of our website to contain copyright information.
In HTML, some symbols need to be represented using a specific code for that symbol. In the example above, the copyright symbol (©) is written as &copy;.

## Lists and Tables

### Lists

Lists are an easy way to display information in a nice, structured format. HTML supports two types of lists, **ordered lists** and **unordered lists**. Ordered lists are enumerated, meaning they display with numbers next to each item. Unordered lists use bullet points instead. Let’s outline the structure used to make lists.

- The `<ol>` tag is used to begin and end an *ordered* list.
- The `<ul>` tag is used to begin and end an *unordered* list.
- The `<li>` tag is used to contain each individual item within a list.

For further clarity, take a look at the example below.

```
<ul>
  <li>First list item</li>
  <li>Second list item</li>
  <li>Third list item</li>
</ul>
```

Let’s go ahead and add a few lists to our webpage to practice.

1. Near the bottom of the document, but before the footer, add a new section to your webpage. In our example website, we will be adding a section called “Top Songs”, to list the most popular songs for our imaginary band.

2. Add opening and closing `<ol>` tags to begin creating a list.

```
<h2>Top Songs</h2>
<ol>
</ol>
```

3. Begin populating the list by adding `<li>` tags for each list item.
Tables

Much like lists, tables have a unique structure that make them slightly more difficult. We will not actually be adding any tables to our example site, so feel free to skip this section if it is of no interest.

We will be re-creating the table below piece-by-piece to learn how to make tables.

<table>
<thead>
<tr>
<th>First Name</th>
<th>Favorite Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>George</td>
<td>Orange</td>
</tr>
<tr>
<td>Allan</td>
<td>Blue</td>
</tr>
</tbody>
</table>

1. Tables open and close with the `<table>` tag. Add these tags to your document to create a table element.

2. Each row of a table is opened and closed with a `<tr>` tag. Our table has three rows, so add four opening and closing `<tr>`.

3. To add information into specific columns, we can add `<td>` tags inside of each table row. Because we have two columns in our table, add two `<td>` tags inside of each table row tag.

   Once this is done, you may populate the table with information.
In most tables, the top row acts as a header for the information being displayed. The header (top row) of the table is usually styled differently than the rest of the table for clarity.

In HTML, we can mark a table header by using `<th>` tags instead of `<td>` tags in the top row.

```
<table>
  <tr>
    <th>First Name</th>
    <th>Favorite Color</th>
  </tr>
  <tr>
    <td>George</td>
    <td>Orange</td>
  </tr>
  <tr>
    <td>Allan</td>
    <td>Blue</td>
  </tr>
</table>
```

Now our table is complete. When previewing in a web browser, it will look different from the example pictured at the top of this section. This is because the table above has a small amount of CSS, or styling applied to it for readability. **HTML simply provides the structure of the table, and CSS is what is used to change it visually.**

## Links

By far, links are one of the most important aspects of a webpage. In fact, links are the only way we can navigate and move between pages on the web. In the website we are building, we will use links in the navigation bar to bounce between multiple pages on our website. First, let's learn how to create links.

### Link Structure
Links begin and end with the `<a>` tag.

```
<a>This is my link</a>
```

However, the format above is incorrect. There is a crucial piece of information we are missing: *where will the link take us?* The whole point of a link is to move a user from one page to another, and if the link does not have a destination, it cannot take us there.

In an earlier section of this manual, we learned that HTML elements can have **attributes and values**. In order to give our link a destination, we can add the `href` (Hypertext Reference) attribute to our element. Take a look at the example below.

```
<a href="#">This is my link!</a>
```

We have successfully added this attribute to our link tag! Great! The value of the attribute is currently a symbol, "#". This is where we place the destination, or the **path** to where we would like to go. For example, if we wanted our link to take us to Google:

```
<a href="http://google.com">This is my link!</a>
```

**Relative and Absolute Referencing**

There are two ways we can write references (destinations) for our links, that are crucial to understand: using **relative paths** and **absolute paths**. Both are described in-depth below.

**Absolute Paths:** These paths point to a specific page or file on the web with zero ambiguity. The path will lead to the same place, no matter what page the user may be on. Examples:

- http://google.com/about
- http://wisc.edu/sts
- http://mywebsite.com/absolute-references.html

**Relative Paths:** These paths lead to a location that is relative to the user's location. Examples:

- index.html
- images/2017/my-photo.jpg
- ../..documents/my-file.pdf

To clarify this further, consider this analogy. If I provide you with GPS coordinates of my exact location, you would be able to find me no matter what your location. GPS coordinates would be an absolute reference to my location.

On the other hand, I could give you instructions: To find me, drive for one mile, turn left, then turn right. In this case, the destination is dependent upon the location of the user. This would be a relative reference to my location.
Let's place this in context of the website we are building. When the user is viewing our homepage, (the current page we are building)

### Adding Links

We would now like to put this into practice, and add links to our own website.

1. At the bottom of our webpage, before the footer, create a new section. In this section, create a list, then add links inside of each list element.

   In the example site we're building, we decided to add links to our band's biggest sponsors.

   ```html
   <h2>Sponsors</h2>
   <ul>
     <li><a href="http://www.bluegrassmuseum.org/">Bluegrass Museum</a></li>
     <li><a href="http://www.spbgma.com/">SPBMA</a></li>
   </ul>
   ```

2. We now want to form a navigation bar at the top of our webpage. Inside the `<nav>` element, create a list and populate it with links, like in the example below.

   ```html
   <!-- nav bar example -->
   <ul>
     <li><a href="index.html">Home</a></li>
     <li><a href="albums.html">Albums</a></li>
     <li><a href="tour.html">Tour</a></li>
   </ul>
   ```

### Pages

We now want to add new pages for our website. In our navigation bar, we added links to an "Albums" and "Tour" page, so these are the pages we can create.

1. In your text editor, navigate to the menu bar and select File > New.
2. Name your file `albums.html`, then save it inside the root folder of your website.
3. Repeat the same process to create a page titled `tour.html`.
4. Take some time to populate these HTML files with content with the same structure we used to make our homepage.

Your website's root folder should now look similar to the following:
Each HTML file now represents a different webpage. In the previous section, we have already written the links to each of these HTML files in our navigation bar. Test your navigation bar by clicking between the three different pages.

Images

The final important topic we'll cover in HTML is adding images. Images are added to your document using a single tag, with no opening or closing. The image tag also has an attribute that will allow us to point to the image's location in our website's root folder.

1. Find an image by searching online, or by using the image provided in the class files folder.

2. Place this image into the images folder within your website's root folder.

3. Within the header, directly below the navigation bar, place an image tag and link it to the image in our website's root folder. Our example is given below.

   `<img src = "images/concert.jpg">

As you can see, we used a relative reference to link our image.

4. Refresh your web browser to preview the image on your page. Remember that the image will show up at its full size, with zero formatting. We will be able to better position our images using CSS in our next class.

5. To complete our image tag, there’s one more attribute we can add: the alternative text. The alt (alternative text) attribute contains a small, written description of the image. It will not show up on your page, however, it will help search engines find your images, and it will also help in the case that the link to your image breaks.

   `<img src = "images/concert.jpg" alt = "Fuzzy and the Blue Tones Concert">

Classes and IDs

One final concept we would like to introduce, is the idea of classes and IDs. Essentially, you can think of classes and IDs as labels or descriptors for certain elements in your webpage. In HTML, every element can be labelled with a
class or ID to categorize it.

You may use classes and IDs interchangeably. However, it is recommended you use classes for recurring elements, and IDs for singular elements that will not repeat again.

First, let's look at some examples of elements with classes attached to them.

```html
<p class = "highlight">Sample text here</p>
<h4 class = "highlight">Sample title here</h4>
<div class = "highlight"></div>
```

Each one of these elements has the `highlight` class attached to it. The name of the class is arbitrary, you may label your elements whatever you like. IDs are structured in a very similar way.

```html
<span id = "hidden">
```

This `<span>` element is labeled with an ID called `hidden`.

So, how are classes and IDs useful? When it comes to styling our page using CSS, or writing JavaScript that interacts with our page, we can use them as selectors.

For example, in CSS you could select every element with the class `highlight` and change it to have a light blue color. Instead of writing code for each individual element, you may select them all at once using the class applied to them. Likewise, you could write JavaScript code that selects elements with the `hidden` ID, and hide them on your webpage.

Unfortunately, because we have not yet learned CSS or JavaScript, classes and IDs are not much use to us.

**CSS Preview**

By the end of this class, you website should look similar to the following:

```
# Fuzzy and the Blue Tones

- Name
- Albums
- Songs

# History

Forming the Band

Fuzzy and the Blue Tones was founded in 2005 by banjo aficionado Bob Fuzzy Mungstein. In 1995 Fuzzy was all set to realize his childhood dream of becoming a skilled pianist for the U.S. Air Force, when tragedy struck. Fuzzy was volunteering as a chef at the local veterans home cooking hot chili pepper stew when suddenly a wasp flew in through an open window. Fuzzy was so startled by the wasp that he knocked the pan of boiling chili pepper stew off of the burner, splashing himself in the eyes. He was quickly rushed to the hospital where they mistakenly treated his right eye, but tragically, the left eye was lost. Devastated, Fuzzy withdrew from society.
```

Of course, this site is far from ideal. The next step is to add color, fonts, formatting, and styling to our page in order to make it appealing. We can do this using another web language called CSS, which has been referenced throughout this
Just for show, we would like to preview how exactly CSS can change a webpage, and how dramatic of a difference it can make. In your class files folder, open up the finished-website-example folder, and open its index.html file. You should see something similar to the following:

The two sites are exactly the same, only the latter has a small amount of CSS applied to it.

To continue learning web development, we strongly recommend the following classes/manuals provided by Software Training for Students:

- CSS 1
- CSS 2
- JavaScript 1
- JavaScript 2
- Adobe Dreamweaver

**Quick Reference**

This section is a reference for commonly used HTML tags, to use as you work on your own website. There are also a number of tags in here that were not discussed during the course, and may be useful to know.

**Content Tags**
### Headings and subheadings

- `<h1>` to `<h6>`: Headings and subheadings.
- `<p>`: Paragraph text.
- `<a href="#">`: Link.
- `<ul>` and `<ol>`: Unordered and ordered lists.
- `<li>`: Individual list elements.
- `<!-- and -->`: Comment. Anything in-between these tags is ignored by the computer.

### Container Tags

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;div&gt;</code></td>
<td>Generic container for a block of content.</td>
</tr>
<tr>
<td><code>&lt;span&gt;</code></td>
<td>Generic container for in-line content.</td>
</tr>
</tbody>
</table>

### Document Structure Tags

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;html&gt;</code></td>
<td>Opens and closes HTML document.</td>
</tr>
<tr>
<td><code>&lt;head&gt;</code></td>
<td>Contains information about the webpage, like title and metadata.</td>
</tr>
<tr>
<td><code>&lt;body&gt;</code></td>
<td>Contains all content displayed on the webpage.</td>
</tr>
</tbody>
</table>

### Semantic Tags

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;header&gt;</code></td>
<td>Contains introductory content, such as website name and navigation.</td>
</tr>
<tr>
<td><code>&lt;nav&gt;</code></td>
<td>Contains primary navigation for website.</td>
</tr>
<tr>
<td><code>&lt;main&gt;</code></td>
<td>Contains main content of page.</td>
</tr>
<tr>
<td><code>&lt;aside&gt;</code></td>
<td>Contains content tangentially related to main content, like a sidebar.</td>
</tr>
<tr>
<td><code>&lt;nav&gt;</code></td>
<td>Contains the footer at the bottom of a page.</td>
</tr>
</tbody>
</table>