Chapter 2

HTML

What is in This Chapter ?

This first chapter of the course explains how to use **HTML** to arrange content on a webpage. We will learn how to use headers, text, images and links. This is only a very basic introduction to **HTML**. There is a lot more that we can learn.

2.1 The Basics

In this section, we will explain how to create the simplest of webpages … which will be based solely on **HTML** … which is the standard language used to create webpages. It tells web browsers how to structure and organize the page content such as text, images, links, etc..

HTML *stands for* ***HyperText Markup Language****.*

The word ***hypertext*** simply means "text with links (or references) to other text". That means that hypertext can be clicked-on by the user and it will take the user to another location, which could be:

- part of the same document/file
- a different document/file
- a webpage on the internet

The word ***markup*** refers to the symbols (or tags) used to annotate text so that the web browser can understand how to structure and display it. It is essentially a way of labelling text with additional information for the browser to take note of so that it knows what to do with the content. These tags are not visible on the webpage, although we can view the source code of a webpage to see them.

The word ***language*** can be confusing to computer science professionals because when we hear that word, we think "*programming* language". However, **HTML** is not a "programming" language, it is a "*markup"* language which means something different. The term *language* is used because it follows a specific syntax, allows structured communication between the person and the browser, and in a way, it is how we "speak" to the browser to give it instructions.

An **HTML** file is just a text file and it has either a **.html** or a **.htm** extension. The default filename that most web servers look for when loading a website or folder is called **index.html**. So, for example, go to my personal webpage:

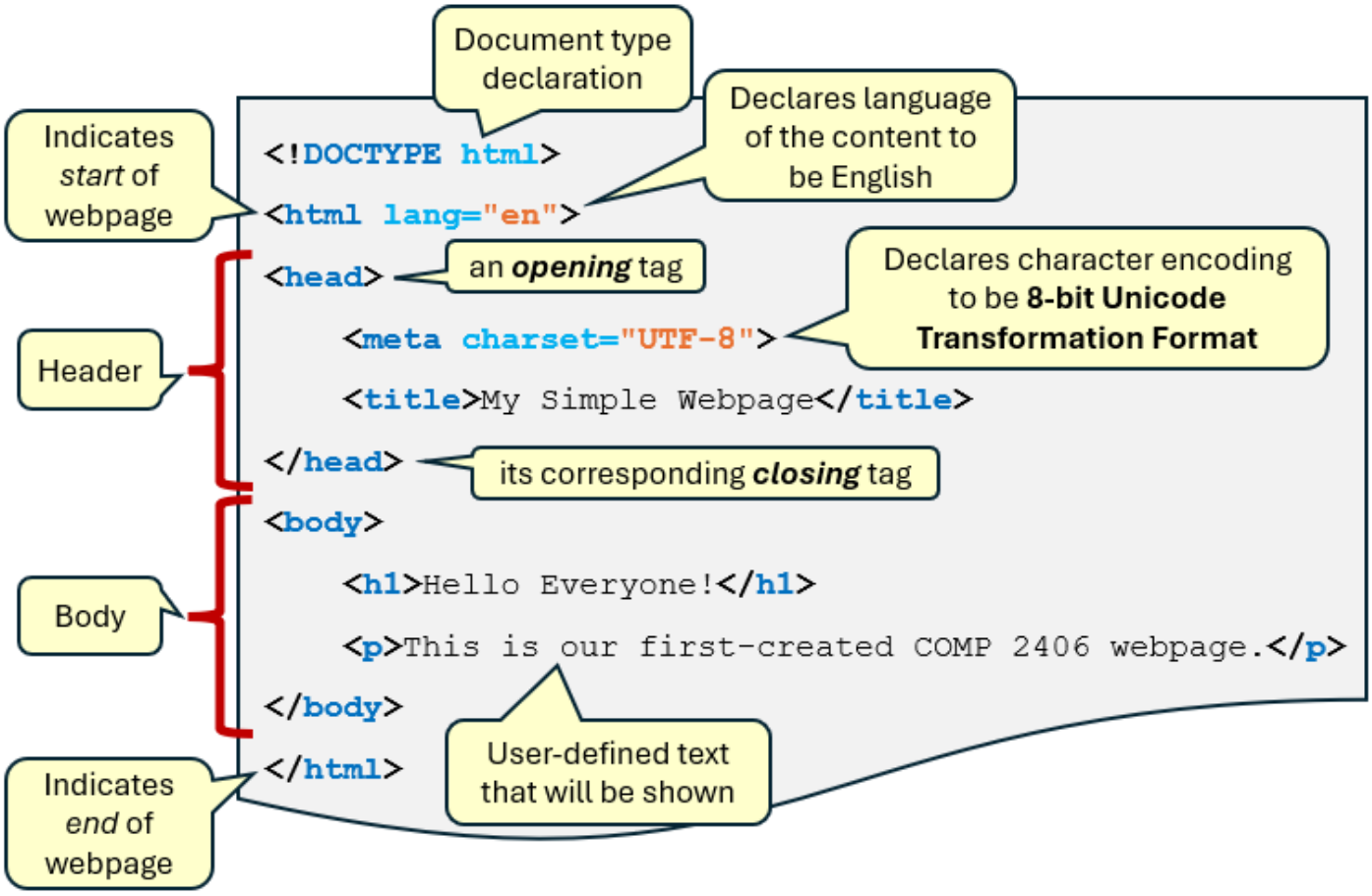
https://people.scs.carleton.ca/~lanthier/

The browser will automatically look for (and find) the **index.html** file that displays my webpage.

For now, let's look at a file called **basic.html** to see what its components are like. Here is a basic template for a simple webpage:

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>My Simple Webpage</title>
    </head>
    <body>
        <h1>Hello Everyone!</h1>
        <p>This is our first-created COMP 2406 webpage.</p>
    </body>
</html>
```

Notice that there are many things declared inside **< >** characters. These are called ***tags*** and they are identified as bold font in the above code. Here is a quick visual explanation of the contents:



Notice the document type declaration at the top. This is a requirement for the **HTML5** standard that is being used today by modern web browsers. It tells the browser that this document uses **HTML5** instead of older versions like **HTML4** or **XHTML**. **HTML5** is the current version of the **HTML** language, as defined by the **W3C** (World Wide Web Consortium) and **WHATWG** (Web Hypertext Application Technology Working Group).

Next, we notice the first **<html>** tag and the last **</html>** tag. These are examples of an ***opening*** tag and a ***closing*** tag, respectively. Closing tags are identified by the **</** sequence of characters. There are a couple of important rules about tags:

1. Every *opening* tag MUST have a corresponding *closing* tag if the tag *wraps* content.
2. Tags can be nested, but must open and close in the correct order.

How many pairs of opening/closing tags do we see in the above code?

We do not include the **<!DOCTYPE>** and **<meta>** tags in our count because those tags are self-contained and do not wrap content. Therefore, we have **6** pairs of opening/closing tags.

There are three opening/closing tag pairs that are important to explain:

1. **<html> </html>**
   - defines the start and end of the webpage content.
   - tells the browser that everything between it is considered **HTML** code.
   - without this our page could break on older or stricter browsers, developer tools and validators will flag it as incorrect and assistive technologies (such as screen readers) may behave unexpectedly.

2. **<head> </head>**
   - contains metadata (i.e., information about the document), settings and resources that are needed before rendering the visible content of the webpage.
   - tells the browser how to load and handle the page.
   - metadata improves our website's visibility and ranking.
   - metatags in here controls how the page displays on phones and tablets.
   - can contain links to **CSS** & **JavaScript** files that define the page's look & interaction (It is interesting to note that an **HTML** file may also contain **CSS** and **JavaScript** code … we will discuss this later).

3. **<body> </body>**
   - contains all the visible content of the webpage such as text, images, links, buttons, videos, etc..

Notice the `lang="en"` inside of the **<html>** tag. There are advantages of making it clear that the language is English:

- ✓ **Accessibility**:
  - Screen readers use it to choose the correct pronunciation or voice.

- ✓ **SEO** (Search Engine Optimization):
  - Search engines use it to better understand and index the page.

- ✓ **Translation tools**:
  - Helps auto-translators (such as Google Translate) to know the source language.

- ✓ **Browser behavior**:
  - Some browsers may adjust features like spellcheck based on the language.

There are other language codes (see IETF language tag). Here are a few others:

Arabic (`ar`), Chinese (`zh`), French (`fr`), Spanish (`es`), German (`de`), Hindi (`hi`)

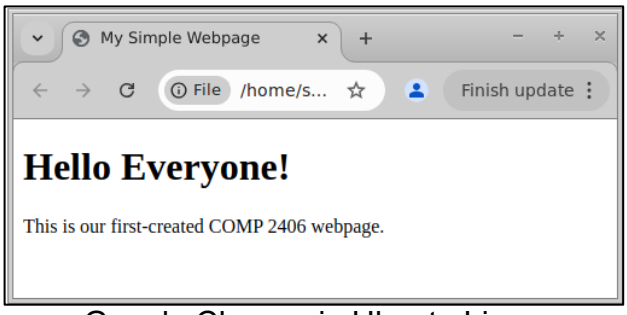
The **<meta charset="UTF-8">** inside the header indicates that the **Unicode Transformation Format - 8-bit** character set is to be used.

There are other options here such as UTF-16, UTF-32, ISO-8859-1, Windows-1252, ASCII, Shift_JIS, KUC-KR, GB2312, etc..

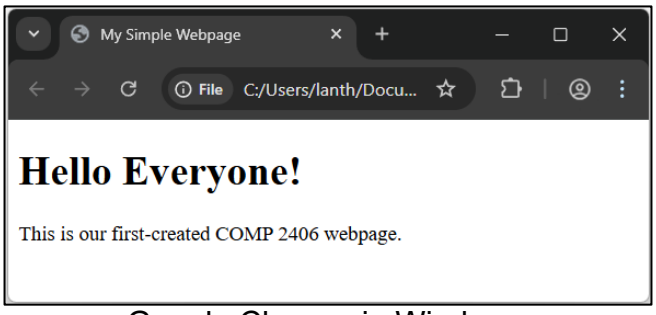
However, **UTF-8** is the most widely used encoding and it has these advantages:

- ✓ **Universal**:
  - Can represent every character in every language
- ✓ **Efficient**:
  - Uses just 1 byte for basic characters (e.g., English), more for others
- ✓ **Web Standard**:
  - Used by over 95% of websites today
- ✓ **Prevents Display Errors**:
  - Avoids weird symbols (□, Ã, etc.) when content has accents or special characters

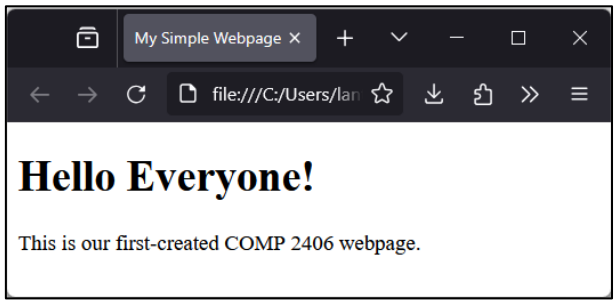
We can take our code and save it in a file called **basic.html**. Then we can open it with a browser. Here is what it looks like using 4 different web browsers:



Google Chrome in Ubuntu Linux



Google Chrome in Windows



Mozilla Firefox in Windows

Microsoft Edge in Windows

Notice that the tab contains the title of the webpage that we provided in the header:

```
<title>My Simple Webpage</title>
```

Also notice that the **Hello Everyone!** text is darker and bigger … because it is within a *header* tag:

```
<h1>Hello Everyone!</h1>
```

While the regular text is in a *paragraph* tag:

```
<p>This is our first-created COMP 2406 webpage.</p>
```

There are six different header tags, each varying in their font size, appearance and spacing. They allow for a consistent structure to webpage content. The largest is **<h1>** and the smallest is **<h6>**. Here is what they are typically used for:

<h1> = Main title or page title (one per page)
<h2> = Major section heading
<h3> = Subtopics under each section
<h4> = Deeper detail in a subtopic
<h5> = Minor subheadings
<h6> = Side notes

Here is some useful advice:

- Use only one **<h1>** per page.
- Use the headings consistently as if we were writing a report with various sections and subsections so that a clear hierarchy is visible.
- Don't skip levels (such as jumping from **<h1>** to **<h4>**) unless it makes sense semantically.

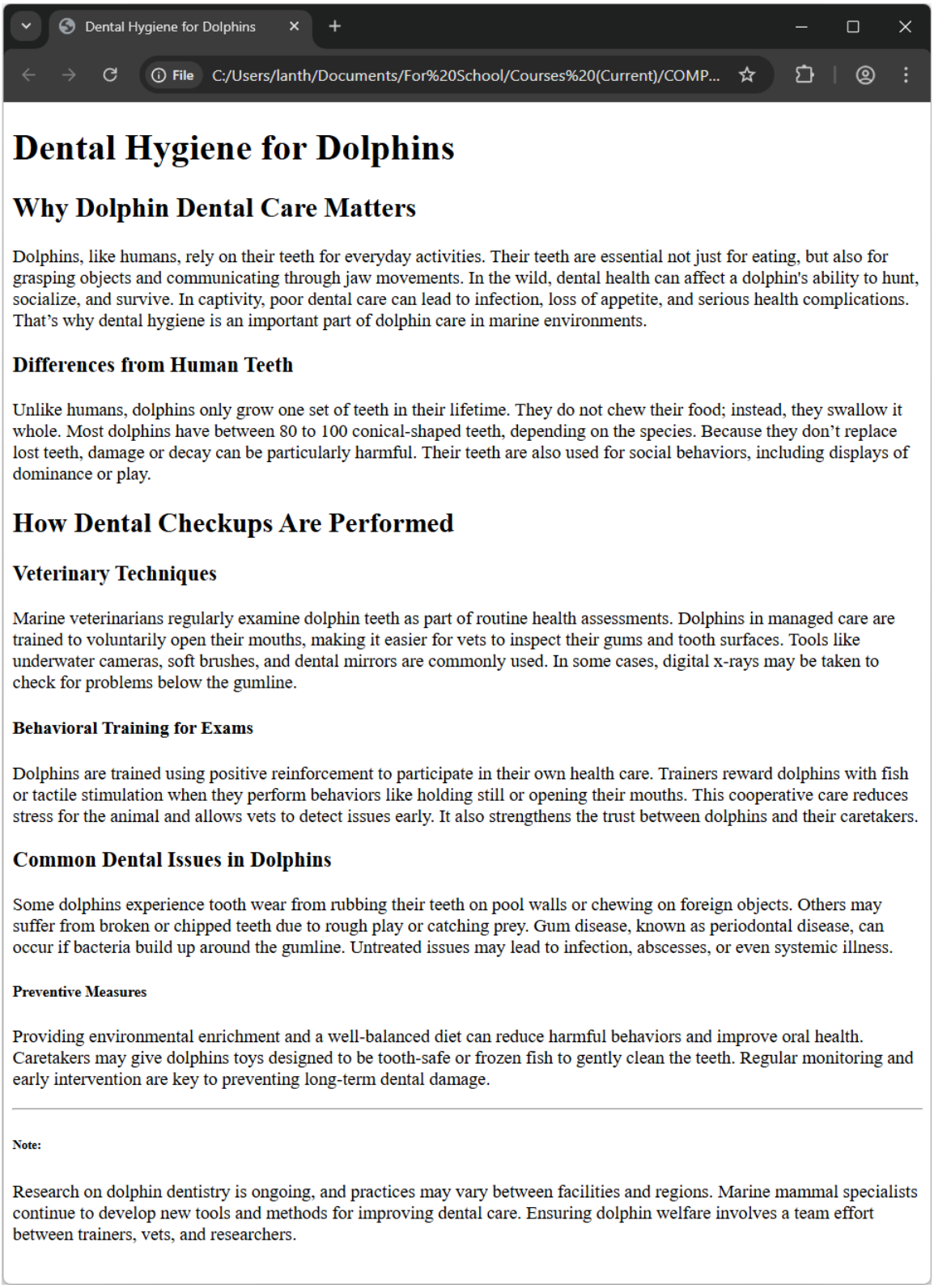
We can also use **<hr>** (i.e., *horizontal rule*) to make a visible horizontal line as a divider … which is good, for example, to add a section of notes at the end of the page.

Here is an example of how this can all be done in a file called **dentalHygeine.html**:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Dental Hygiene for Dolphins</title>
</head>
<body>

  <h1>Dental Hygiene for Dolphins</h1>

  <h2>Why Dolphin Dental Care Matters</h2>
  <p>
    Dolphins, like humans, rely on their teeth for everyday activities. Their teeth are
    essential not just for eating, but also for grasping objects and communicating through jaw
    movements.

    In the wild, dental health can affect a dolphin's ability to hunt, socialize, and survive.

    In captivity, poor dental care can lead to infection, loss of appetite, and serious health
    complications. That's why dental hygiene is an important part of dolphin care in marine
    environments.
  </p>

  <h3>Differences from Human Teeth</h3>
  <p>
    Unlike humans, dolphins only grow one set of teeth in their lifetime. They do not chew
    their food; instead, they swallow it whole. Most dolphins have between 80 to 100
    conical-shaped teeth, depending on the species. Because they don't replace lost teeth,
    damage or decay can be particularly harmful. Their teeth are also used for social
```

```
    behaviors, including displays of dominance or play.
  </p>

  <h2>How Dental Checkups Are Performed</h2>

  <h3>Veterinary Techniques</h3>
  <p>
    Marine veterinarians regularly examine dolphin teeth as part of routine health assessments.
    Dolphins in managed care are trained to voluntarily open their mouths, making it easier for
    vets to inspect their gums and tooth surfaces. Tools like underwater cameras, soft brushes,
    and dental mirrors are commonly used. In some cases, digital x-rays may be taken to check
    for problems below the gumline.
  </p>

  <h4>Behavioral Training for Exams</h4>
  <p>
    Dolphins are trained using positive reinforcement to participate in their own health care.
    Trainers reward dolphins with fish or tactile stimulation when they perform behaviors like
    holding still or opening their mouths. This cooperative care reduces stress for the animal
    and allows vets to detect issues early. It also strengthens the trust between dolphins and
    their caretakers.
  </p>

  <h3>Common Dental Issues in Dolphins</h3>
  <p>
    Some dolphins experience tooth wear from rubbing their teeth on pool walls or chewing on
    foreign objects. Others may suffer from broken or chipped teeth due to rough play or
    catching prey. Gum disease, known as periodontal disease, can occur if bacteria build up
    around the gumline. Untreated issues may lead to infection, abscesses, or even systemic
    illness.
  </p>

  <h5>Preventive Measures</h5>
  <p>
    Providing environmental enrichment and a well-balanced diet can reduce harmful behaviors
    and improve oral health. Caretakers may give dolphins toys designed to be tooth-safe or
    frozen fish to gently clean the teeth. Regular monitoring and early intervention are key to
    preventing long-term dental damage.
  </p>

  <hr>  ← this will make horizontal line across the page

  <h6>Note:</h6>
  <p>
    Research on dolphin dentistry is ongoing, and practices may vary between facilities and
    regions. Marine mammal specialists continue to develop new tools and methods for improving
    dental care. Ensuring dolphin welfare involves a team effort between trainers, vets, and
    researchers.
  </p>

</body>
</html>
```

Here is what it looks like:

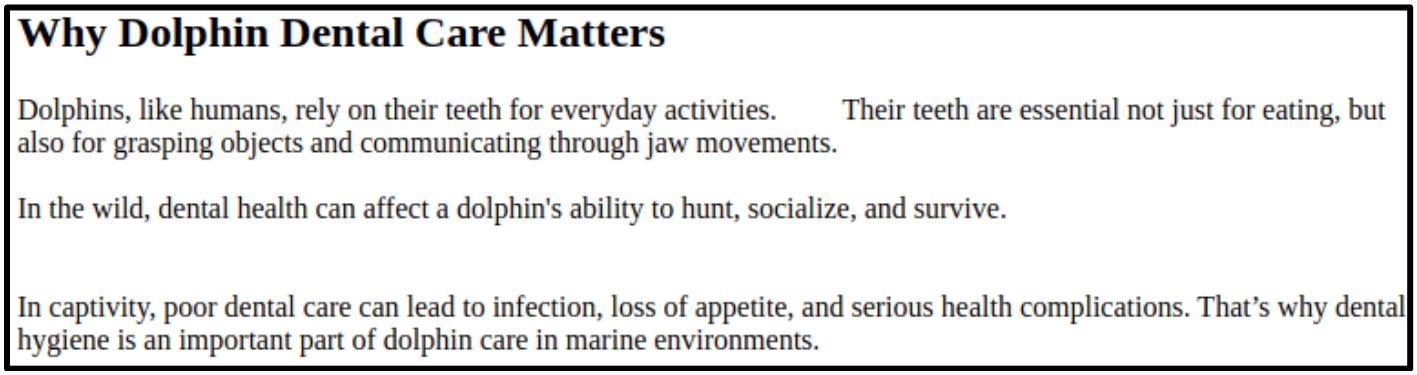


Notice something about the text in the first paragraph. The browser will automatically wrap text around and will ignore any *extra* blank spaces and blank lines that we add to the text. This allows the browser to have more freedom in laying out the content depending on the size of the view (e.g., pc, phone, tablet, etc..).

We can force a line break with a **
** tag and we can force single spaces by using ** ** (which stands for ***non-breaking space***). So, we can do this:

```
<p>
  Dolphins, like humans, rely on their teeth for everyday activities.
          Their teeth are essential not just for eating, but also for
  grasping objects and communicating through jaw movements.
  <br> <br>

  In the wild, dental health can affect a dolphin's ability to hunt, socialize, and survive.
  <br> <br> <br>

  In captivity, poor dental care can lead to infection, loss of appetite, and serious health
  complications. That’s why dental hygiene is an important part of dolphin care in marine
  environments.
</p>
```

Now, it will display like this:



Notice the extra spaces and the extra lines. However, it is not considered good/proper **HTML** style to use **
** tags and ** ** for layout. We will see later how to use **CSS** for proper spacing.

Whenever we are structuring webpage content such as articles, reports or multi-part documents, it would be good to also use **<section> … </section>** tags. This allows us to group related content into logical blocks (e.g., chapters, subsections, topics, etc..). Really, any time we have content that belongs together thematically (i.e., represents a standalone idea or topic), we should use **<section>** tags. Some advantages to using them are that it is …

- ✓ easier for browsers and screen readers to **navigate**
- ✓ easier for search engines to **understand** content hierarchy
- ✓ easier for us developers to **maintain** larger webpages

It is always best to use a heading tag (e.g., **<h2>**) for each section. In fact, if we don’t have a heading tag, then we likely shouldn’t be using a **<section>** tag. **<section>** tags should be used for *logical* divisions … not simply to wrap every paragraph in a section.

There is a similar tag called ** … ** that we will see later. It is also used to logically group together **HTML** code, but mainly for the purpose of styling. It is like saying “Everything within this tag will have the same style (e.g., color, font, alignment etc..)”.

Even though the appearance of the webpage will be the same, here is what the code looks like in a file called **dentalHygieneSections.html**:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Dental Hygiene for Dolphins</title>
</head>

<body>
    <h1>Dental Hygiene for Dolphins</h1>

    <section>
        <h2>Why Dolphin Dental Care Matters</h2>
        <p>
            Dolphins, like humans, rely on their teeth for everyday activities. Their teeth are
            essential not just for eating, but also for grasping objects and communicating
            through jaw movements. In the wild, dental health can affect a dolphin's ability to
            hunt, socialize, and survive. In captivity, poor dental care can lead to infection,
            loss of appetite, and serious health complications. That’s why dental hygiene is an
            important part of dolphin care in marine environments.
        </p>

        <section>
            <h3>Differences from Human Teeth</h3>
            <p>
                Unlike humans, dolphins only grow one set of teeth in their lifetime. They do
                not chew their food; instead, they swallow it whole. Most dolphins have between
                80 to 100 conical-shaped teeth, depending on the species. Because they don’t
                replace lost teeth, damage or decay can be particularly harmful. Their teeth
                are also used for social behaviors, including displays of dominance or play.
            </p>
        </section>
    </section>

    <section>
        <h2>How Dental Checkups Are Performed</h2>

        <section>
            <h3>Veterinary Techniques</h3>
            <p>
                Marine veterinarians regularly examine dolphin teeth as part of routine health
                assessments. Dolphins in managed care are trained to voluntarily open their
                mouths, making it easier for vets to inspect their gums and tooth surfaces.
                Tools like underwater cameras, soft brushes, and dental mirrors are commonly
                used. In some cases, digital x-rays may be taken to check for problems below
                the gumline.
            </p>
        </section>

        <section>
            <h4>Behavioral Training for Exams</h4>
            <p>
                Dolphins are trained using positive reinforcement to participate in their own
                health care. Trainers reward dolphins with fish or tactile stimulation when
                they perform behaviors like holding still or opening their mouths. This
                cooperative care reduces stress for the animal and allows vets to detect issues
                early. It also strengthens the trust between dolphins and their caretakers.
            </p>
        </section>
```

```
    </section>

    <section>
        <h3>Common Dental Issues in Dolphins</h3>
        <p>
            Some dolphins experience tooth wear from rubbing their teeth on pool walls or
            chewing on foreign objects. Others may suffer from broken or chipped teeth due to
            rough play or catching prey. Gum disease, known as periodontal disease, can occur
            if bacteria build up around the gumline. Untreated issues may lead to infection,
            abscesses, or even systemic illness.
        </p>

        <section>
            <h5>Preventive Measures</h5>
            <p>
                Providing environmental enrichment and a well-balanced diet can reduce harmful
                behaviors and improve oral health. Caretakers may give dolphins toys designed
                to be tooth-safe or frozen fish to gently clean the teeth. Regular monitoring
                and early intervention are key to preventing long-term dental damage.
            </p>
        </section>

        <hr>

        <section>
            <h6>Note:</h6>
            <p>
                Research on dolphin dentistry is ongoing, and practices may vary between
                facilities and regions. Marine mammal specialists continue to develop new tools
            and methods for improving dental care. Ensuring dolphin welfare involves a team
                effort between trainers, vets, and researchers.
            </p>
        </section>
    </section>

</body>
</html>
```

Notice how much more readable the document is with the extra sections added and tabbed nicely … although it will display the same in the browser.

2.2 Other Common HTML Tags

Another often-used tag pair in **HTML** is **<table> </table>**, which allows us to organize our content into rows and columns. Within these tags, we can add a caption, a header and a body to the table using their own wrapping tags as follows:

```
<table>
    <caption> ... </caption>
    <thead> ... </thead>
    <tbody> ... </tbody>
</table>
```

Then we use **<tr> </tr>** tag pair to indicate a row of the table. For each cell of the table, we can use the **<th> </th>** tag pair to wrap information that goes in the cell of the table's header and the **<td> </td>** tag pair to wrap information that goes in a cell of the body of the table.

Here is an example (in the **movieTable.html** file) that shows a table of some movies and some statistics about the movie:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Movie List</title>
</head>

<body>
    <table border="1">
        <caption>Popular Movies and Their Ratings</caption>
        <thead>
            <tr>
                <th>Title</th>
                <th>Genre</th>
                <th>Rating</th>
                <th>Release Year</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>Inception</td>
                <td>Science Fiction</td>
                <td>8.8</td>
                <td>2010</td>
            </tr>
            <tr>
                <td>The Godfather</td>
                <td>Crime, Drama</td>
                <td>9.2</td>
                <td>1972</td>
            </tr>
            <tr>
                <td>Jurassic Park</td>
                <td>Action, Adventure, Sci-Fi</td>
                <td>8.2</td>
                <td>1993</td>
            </tr>
            <tr>
                <td>Toy Story</td>
                <td>Animation, Family</td>
                <td>8.3</td>
                <td>1995</td>
            </tr>
        </tbody>
    </table>
</body>
</html>
```

Popular Movies and Their Ratings

Title	Genre	Rating	Release Year
Inception	Science Fiction	8.8	2010
The Godfather	Crime, Drama	9.2	1972
Jurassic Park	Action, Adventure, Sci-Fi	8.2	1993
Toy Story	Animation, Family	8.3	1995

As we can see, the table keeps things nicely organized into cells. Notice as well that we can specify the number of pixels of thickness for the table (and cell) borders. In **HTML5**, we don't usually do this now, we will use **CCS** (more later).

The table header items are, by default, displayed in **bold** and centered, although this is all adjustable. It is interesting to note that it is not necessary to wrap the header cells within **<thead> </thead>** tags, but it is good to use it because:

- ✓ **Semantic clarity** - it tells the browser and assistive tech: "This is the header"
- ✓ **Sticky headers** - in some **CSS/JS** frameworks, it helps implement sticky table headers
- ✓ **Consistency when printing** - browsers may repeat **<thead>** rows on each printed page for multi-page tables
- ✓ **Styling** – we can easily style just the header rows via **CSS** (more later)

We can also specify that some data should span multiple rows or columns by inserting **rowspan** and **colspan** within the **<td>** tag. For example, how would we produce this table:

<table>
<tr><th>Time</th><th>Monday</th><th>Tuesday</th><th>Wednesday</th><th>Thursday</th><th>Friday</th></tr>
<tr><td>9:00–10:00</td><td rowspan="2">Math</td><td>English</td><td rowspan="3">Science</td><td>History</td><td rowspan="2">PE</td></tr>
<tr><td>10:00–11:00</td><td rowspan="2">Art</td><td>Geography</td></tr>
<tr><td>11:00–12:00</td><td>Computer</td><td colspan="2">Free Period</td></tr>
<tr><td>12:00–1:00</td><td colspan="5">Lunch Break</td></tr>
<tr><td>1:00–2:00</td><td>Drama</td><td rowspan="2">Biology</td><td>Music</td><td rowspan="2">Chemistry</td><td>Math</td></tr>
<tr><td>2:00–3:00</td><td>English</td><td>PE</td><td>History</td></tr>
</table>

Well, we would need to indicate that **Math**, **Art**, **Biology**, **PE** and **Chemistry** all span 2 rows, **Science** spans 3 rows, **Free Period** spans 2 columns and **Lunch Break** spans 5 columns.

Here is the code (in a file called **scheduling.html**), with a few extra features as well:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Movie List</title>
</head>
<body>
    <table border="1" cellpadding="8" cellspacing="0">
        <thead>
            <tr>
                <th>Time</th>
```

```
                <th>Monday</th>
                <th>Tuesday</th>
                <th>Wednesday</th>
                <th>Thursday</th>
                <th>Friday</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>9:00–10:00</td>
                <td rowspan="2">Math</td>
                <td>English</td>
                <td rowspan="3">Science</td>
                <td>History</td>
                <td rowspan="2">PE</td>
            </tr>
            <tr>
                <td>10:00–11:00</td>
                <td rowspan="2">Art</td>
                <td>Geography</td>
            </tr>
            <tr>
                <td>11:00–12:00</td>
                <td>Computer</td>
                <td colspan="2">Free Period</td>
            </tr>
            <tr>
                <td>12:00–1:00</td>
                <td colspan="5" style="text-align:center;">Lunch Break</td>
            </tr>
            <tr>
                <td>1:00–2:00</td>
                <td>Drama</td>
                <td rowspan="2">Biology</td>
                <td>Music</td>
                <td rowspan="2">Chemistry</td>
                <td>Math</td>
            </tr>
            <tr>
                <td>2:00–3:00</td>
                <td>English</td>
                <td>PE</td>
                <td>History</td>
            </tr>
        </tbody>
    </table>
</body>
</html>
```

Time	Monday	Tuesday	Wednesday	Thursday	Friday
9:00–10:00	Math	English	Science	History	PE
10:00–11:00		Art		Geography	
11:00–12:00	Computer			Free Period	
12:00–1:00	Lunch Break				
1:00–2:00	Drama	Biology	Music	Chemistry	Math
2:00–3:00	English		PE		History

The **rowspan** and **colspan** values should be intuitive, according to what we needed. However, while the **9:00-10:00** timeslot row has **6 <td>** tags, notice that the **10:00-11:00** timeslot row has only **3 <td>** tags. That is because **Math**, **Science** and **PE** "extend into" the next row, so it is implied that there will be no data cells for those columns in the **10:00-11:00** timeslot row. Only **Art** and **Geography** are the data items for that row.

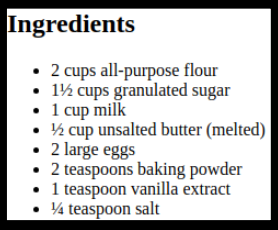
Notice at the beginning of the table we are using `cellpadding="8"`. This allows **8** pixels of spacing all around the inside of the table cells, which spaces them out nicely. The `cellspacing="0"` allows no spacing around the outside of the cells, whereas the default was **2** pixels as we can see in our earlier movie ratings table.

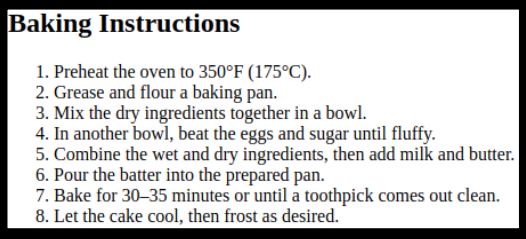
Lastly, notice the use of `style="text-align:center;"` in the **Lunch Break** row. This allows the words to be centered across the **5** columns that it spans.

There are so many (other) options available. As it turns out, in **HTML5**, the `border=`, `cellpadding=` and `cellspacing=` are all deprecated, so it is not good to use them. Instead, we will use **CSS** options. The use of `style="text-align:center;"` is actually considered ***inline CSS***. We will talk more about this soon.

Other useful formatting tags are ones used for creating lists. We can simply wrap our list items in the `<ul> </ul>` tags if we want an unordered (i.e., bulleted) list and `<ol> </ol>` if we want a numbered list. Then we wrap each list item with the `<li> </li>` tags . Here is **cakeBake.html**:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Cake Bake</title>
</head>
<body>
    <h1>Vanilla Sponge Cake</h1>
    <p>
        This recipe is for a classic vanilla sponge cake. It's light, fluffy, and perfect for
        any occasion. Whether you're serving it plain, dusted with powdered sugar, or topped
        with your favorite frosting, this cake is a timeless treat that's easy to make and
        always delicious.
    </p>

    <h2>Ingredients</h2>
    <ul>
        <li>2 cups all-purpose flour</li>
        <li>1½ cups granulated sugar</li>
        <li>1 cup milk</li>
        <li>½ cup unsalted butter (melted)</li>
        <li>2 large eggs</li>
        <li>2 teaspoons baking powder</li>
        <li>1 teaspoon vanilla extract</li>
        <li>¼ teaspoon salt</li>
    </ul>

    <h2>Baking Instructions</h2>
    <ol>
        <li>Preheat the oven to 350°F (175°C).</li>
        <li>Grease and flour a baking pan.</li>
        <li>Mix the dry ingredients together in a bowl.</li>
        <li>In another bowl, beat the eggs and sugar until fluffy.</li>
        <li>Combine the wet and dry ingredients, then add milk and butter.</li>
        <li>Pour the batter into the prepared pan.</li>
```





```
        <li>Bake for 30-35 minutes or until a toothpick comes out clean.</li>
        <li>Let the cake cool, then frost as desired.</li>
    </ol>

    <h2>Serving</h2>
    <p>
        Allow the cake to cool completely before serving. For best results, slice with a
        serrated knife to maintain the cake's fluffy texture. Serve plain, dusted with
        powdered sugar, or topped with whipped cream and fresh fruit. This cake pairs
        wonderfully with tea or coffee and can also be layered with frosting or jam for
        a more festive presentation.
    </p>

    <hr>

    <h6> Disclaimer </h6>
    <p>
        This recipe is provided for informational purposes only. Cooking times and
        results may vary depending on your equipment and ingredients. Please use
        caution when handling hot ovens and sharp utensils. We are not responsible
        for any accidents or damages that may occur while following this recipe.
    </p>
</body>
</html>
```

2.3 Images, Links and Forms

Text-only pages are boring. How do we add images to our webpages? We simply use the self-closing **<img>** tag which has this format:

```
<img src="imageFileName.jpg" alt="Description of image">
```

The `src` indicates the name of the file, which should also include the path to the file or the **URL**. Here are a common options for the `src` value:

- **Relative URL (i.e., relative path)** - points to an image file relative to current **HTML** file location

```
src="cake.jpg"                ← file is in same folder as the html file
src="images/cake.jpg"         ← file is in images subfolder from where the html file is
src="./images/cake.jpg"       ← ./ means current directory, so same as above
src="../images/cake.jpg"      ← back up one folder then go into images folder
src="../../images/cake.jpg"   ← back up two folders then go into images folder
```

- **Absolute URL (full web address)** - points to an image hosted anywhere on the web

```
src="https://madeup-example.com/images/cake.jpg"
```

- **Local file system path (not recommended for web)** - points to a local file on computer

```
src="C:/Users/Me/Pictures/cake.jpg"
```

The `alt` is optional. It lets us specify alternative text to be shown if the image cannot load. We should always provide the `alt` description because it:

- ✓ it improves **accessibility** for screen readers.
- ✓ displays **something** if the image fails to load (e.g., **URL** is invalid or file type not supported).
- ✓ helps with **search engine optimization** so that image shows up in image search results.
- ✓ provides text description if browser turns off image display .
  (e.g., to reduce data cost when using a mobile plan or to speed up loading)

What goes into an `alt` description? We should try to keep it concise and yet informative, ensuring that it is descriptive and relevant to the image, yet avoid stuffing keywords together unnaturally.

Since images can vary greatly in size and resolution, it is common to adjust the size of an image on our webpage so that it fits into a specific area. We can use the `width` and `height` options withing our **<`img`>** tag to specify the number of pixels that the width and/or height of the image will be displayed with. We can also supply an optional `title` value that with be a descriptive text that will appear when the cursor hovers over the image.

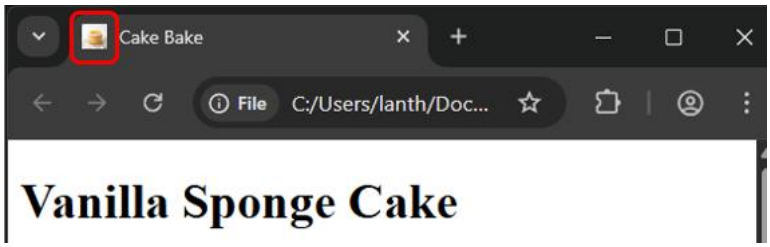
In our cake-baking example, we can add an image of a finished cake right under the heading as follows:

```
<h1>Vanilla Sponge Cake</h1>

<img src="spongecake.jpg" width=200 height=200 title="Vanilla Sponge Cake"
     alt="Vanilla sponge cake with icing and peach slice topping">

<p>
    This recipe is for a classic vanilla sponge cake.
    It's light, fluffy, and …
```

For this to work, we would need to ensure that the **spongecake.jpg** image is in the same directory as the **cakeBake.html** file. Otherwise, we would have to adjust the `src`= value to be the path that we would need to follow to get to the cake image.

The image will be displayed as shown here →

We can also tell the browser to display a ***favicon*** (i.e., favorite icon) at the left of our webpage tab as shown here in the image below (i.e., **spongecakeIcon.jpg**):



To add the favicon, we add a **<link>** tag in the **<head>** section of the webpage, often just below the **<title>** tag as follows (check it out in the **cakeBakeImage.html** file):

```
<head>
    <meta charset="UTF-8">
    <title>Cake Bake</title>
    <link rel="icon" href="spongecakeIcon.png">
</head>
```

Here, the **rel=** attribute stands for the ***relationship*** between the current html document and the linked resource. In our case, we are specifying that we want to supply the icon for the webpage. The **href=** attribute stands for ***hypertext reference***. In the case of specifying an icon, we should supply the path to the image file.

Icon images are typically one of three types:

- **.ico** - traditional, supported everywhere
- **.png** - modern, supports transparency
- **.svg** - scalable vector, newer browsers

The standard size of an icon is **16×16** pixels, but modern browsers support larger/responsive versions too … meaning that the icon can adapts well to different screen sizes, resolutions, and devices.

In the above example, we used **href=** to refer to an external image file. In general, we can refer to any file, which may not be an image.

Anywhere on our webpage file, we could insert a ***link*** to a resource that the user can click on. A ***link*** uses the ***anchor*** tag pair (i.e., **<a> **) which has this format:

```
<a href="pathToFilename.xxx">linkText</a>
```

Once again, the path to the filename/resource can be relative or absolute, just as with image files. The `linkText` is the text that appears on the webpage, which the user would click on to get to the resource.

Consider the following tree structure for folders and files being used by a website (directories / folders are shown in purple) →

```
food-website
    about.html        ← about the food brand
    contact.html      ← contact page
    index.html        ← main homepage

    icons ← favicons or SVG icon files
        apple-touch-icon.png
        favicon.ico

    images ← images for main page
        background-pattern.png
        banner.png
        logo.png

    products ← product category pages
        baked-goods.html
        beverages.html
        fruits.html

        details ← specific product detail pages
            apple.html
            chocolate-cake.html
            lemonade.html

    scripts ← JavaScript files
        interactivity.js

    styles ← CSS files
        main.css
```

In the **index.html** file, how would we set up the favicon which is in the **icons** subdirectory?

```
<link rel="icon" href="icons/favicon.ico">
```

How would we include/display the **logo.png** image file as an image on the **index.html** webpage?

```
<img src="images/logo.png"
     title="Company Logo"
     alt="Delicious Food company logo">
```

Assume that we had the following code somewhere on the **index.html** webpage:

```
<body>
    <h1>Products</h1>
    <ul>
        <li>Baked Goods</li>
        <li>Beverages</li>
        <li>Fruits</li>
    </ul>
</body>
```

Products

- Baked Goods
- Beverages
- Fruits

Assume also, that we had separate webpages (i.e., **.html** files) for each of these products within a **products** subfolder as shown in the above hierarchy. How can we turn the three bullet items (in the list above) into links to each of these three webpages?

We would simply make use of the anchor **<a>** tags and reference each page accordingly:

```
<body>
    <h1>Products</h1>
    <ul>
        <li><a href="products/baked-goods.html">Baked Goods</a></li>
        <li><a href="products/beverages.html">Beverages</a></li>
        <li><a href="products/fruits.html">Fruits</a></li>
    </ul>
</body>
```

Products

- Baked Goods
- Beverages
- Fruits

What if the **baked-goods.html** page required the **logo.png** file to be used, which is in the **images** folder? What would the image **src** look like if we wanted to reference that same image file?

```
<img src="../images/logo.png" title="Company Logo"
     width=50 height=50 alt="Delicious Food company logo">
```

Baked Goods

- pies
- cakes
- muffins

What if we wanted the user to be able to click on the logo to go back to the main **index.html** page?

We could make a link from the image by supplying the logo image inside the link instead of text as follows:

```
<a href="../index.html">
    <img src="../images/logo.png" title="Company Logo"
         width=50 height=50 alt="Delicious Food company logo">
</a>
```

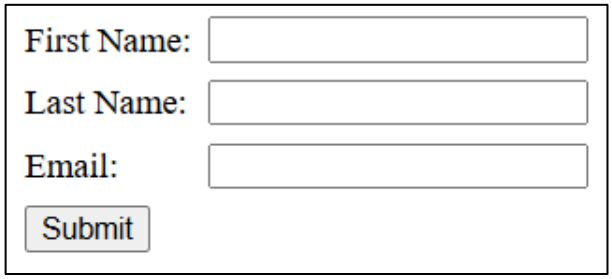
Clicking on this link will allow the **index.html** file to open in the same tab/window as the **baked-goods.html** file. If we want the **index.html** file to open in a new window, we can set the `target` to `"_blank"` as follows:

```
<a href="../index.html" target="_blank">
    <img src="../images/logo.png" title="Company Logo"
         width=50 height=50 alt="Delicious Food company logo">
</a>
```

The default (to open in same tab/window) is `target="_self"`, which we never have to specify.

FORMS

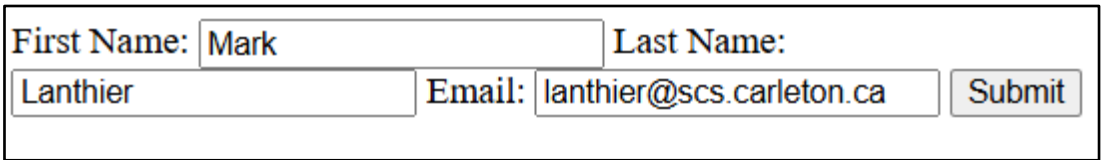
One last **HTML** item that we will mention is that of forms. An ***HTML form*** allows users to input data and submit it to a server for processing (e.g., signing up, logging in, searching, providing feedback, etc..). We will talk more about this later in the course but for now, how can we make a form like this shown here with 3 labels and their corresponding text fields along with a **Submit** button? →



Here is the code:

```
<form action="https://example.com/apply" method="post">
    <label for="first">First Name:</label>
    <input type="text" id="first" name="firstname">

    <label for="last">Last Name:</label>
    <input type="text" id="last" name="lastname">

    <label for="email">Email:</label>
    <input type="email" id="email" name="email">

    <input type="submit">
</form>
```

Unfortunately, it looks like this though? …



We need to use a table to contain all the input items as follows:

```
<form action="https://example.com/apply" method="post">
    <table cellpadding="4" cellspacing="0">
        <tbody>
            <tr>
                <td><label for="first">First Name:</label></td>
                <td><input type="text" id="first" name="firstname"></td>
            </tr>
            <tr>
                <td><label for="last">Last Name:</label></td>
                <td><input type="text" id="last" name="lastname"></td>
            </tr>
            <tr>
                <td><label for="email">Email:</label></td>
                <td><input type="email" id="email" name="email"></td>
            </tr>
            <tr>
                <td><input type="submit"></td>
            </tr>
        </tbody>
    </table>
</form>
```

Getting back to the **<form>** tag … notice the `action` and `method` attributes. The `action` attribute lets us specify the **URL** where we want to send the form data to when the **Submit** button is pressed. The `method` is either `"get"` or `"post"` (more on this later in the course).

There are three `label`/`input` pairs in the body of the form. The value of the `for`= in the label should match the value of the `id`= in the input so that a connection is made between the two.

There are many options for the **<`input`>** tag `type` attribute:

- `"text"` – a regular text field
- `"password"` – a password text field (text will be hidden)
- `"email"` – an email address text field
- `"checkbox"` – a checkbox
- `"radio"` – a radio button
- `"submit"` – a submit button
- `"file"` – a file upload

What happens when we submit the form by pressing the **Submit** button?

1. The data is collected from the form fields.
2. The browser sends the data to the server using the specified method (i.e., get or post).
3. The server processes the data and returns a response.

We will look much more at this later in the course.

MUCH MORE…

When it comes to **HTML**, there is a LOT more to know. We can look up online references when we are trying to do something specific that we have not discussed.

We should bookmark this website and use it throughout the course: https://www.w3schools.com

It contains tutorials, guides and references for **HTML**, **CSS**, **Javascript** … and many other things.

There is a nice style guide there for **HTML**. Please look that over because all submitted code in this course is expected to follow proper/valid **HTML** coding practices.

Here are some tips mentioned there:

- Use lowercase tag names (e.g., <body> … not <BODY>)
- Use lowercase names (e.g., <a href="…"> … not <a HREF="…">)
- Close all **HTML** tags that need to be closed (e.g., …)
- Always quote attribute values (e.g., <a src="logo.png">… not <a src=logo.png>)
- Use lowercase file names (e.g., **logo.png** … not … **Logo.png**)
- Always specify `alt`, `width` and `height` for images.
- Don't put spaces around equal signs (e.g., <a href="…"> … not <a href = "…">)
- Do not add blank lines and indentation without a reason.
- For readability, add blank lines to separate large or logical code blocks.
- Make sure that we indent and align nested levels of tags.
- Always include the <title>, <html>, <head>, and <body> tags.

We can check all our **HTML** code to see if our code is valid by going to this website:

https://validator.w3.org/#validate_by_input+with_options

Now that we have learned the basics of **HTML**, we can think of it as the backbone (or "skeleton") of every web page. That is … **HTML** gives structure and meaning to the content of our page. But a plain **HTML** page still looks pretty bare (and lame). To make those pages visually appealing and easier to read, we need a way to style and format what we've built. In the next chapter, we will explore how to control colors, fonts, layouts, and overall design so our pages won't just "work" … they will actually look polished and professional.