Front-end Development

1. Where are you from?
2. Thing that makes you laugh?
3. What are your expectations for the course?

Web Development

What Do Web Developers Do?

- Web developers build and maintain websites. The work is typically very project focused and involves collaborating with a team that helps to coordinate the client's needs into the end product. The client could be a tech company, an organization, or a government. The work could involve front-end, back-end, or full-stack web development.
- Types of Web Developers:
  - **FRONT-END web developer**
  - **BACK-END web developer**
  - **FULL-STACK web developer**

How does the web work?

- What is the internet and how does it work?
- What is a network?
- What is an IP address, and what is a DNS request?
- What is a router?
- What are packets and how are they used to transfer data?
- What is a client?
- What is a server?
- What is a webpage and what is a website?
- What is a web server?
- What is a web browser, and what is a search engine?

Programing languages: Front-end

- With **HTML**, you will be able to create the structure of your website.
- **CSS** gives you the ability to make the website look more visually appealing.
- As for **JavaScript**, this is a robust programming language that allows you to effectively change the HTML and CSS components of your website to match your specifications precisely.

Installation & Setup

Installation & Setup

1. What OS(Operating System) are you using?(https://whatsmyos.com/)
2. What internet browser are you using? (https://www.whatismybrowser.com/)
3. VS Code installation(https://code.visualstudio.com/download)
4. Create a GitHub acc/Download GitHub Desktop (https://desktop.github.com/)

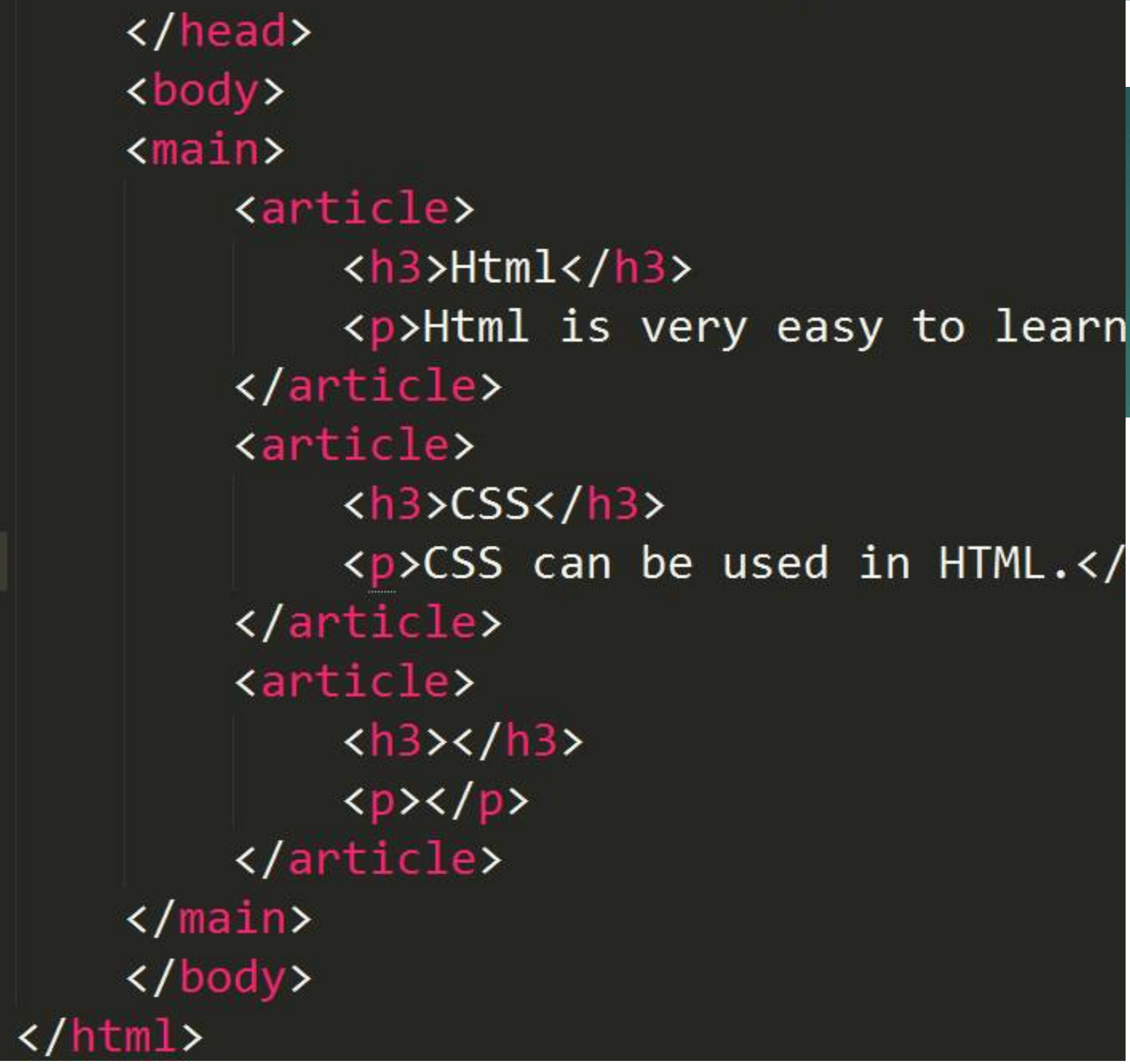
HTML

What is HTML?

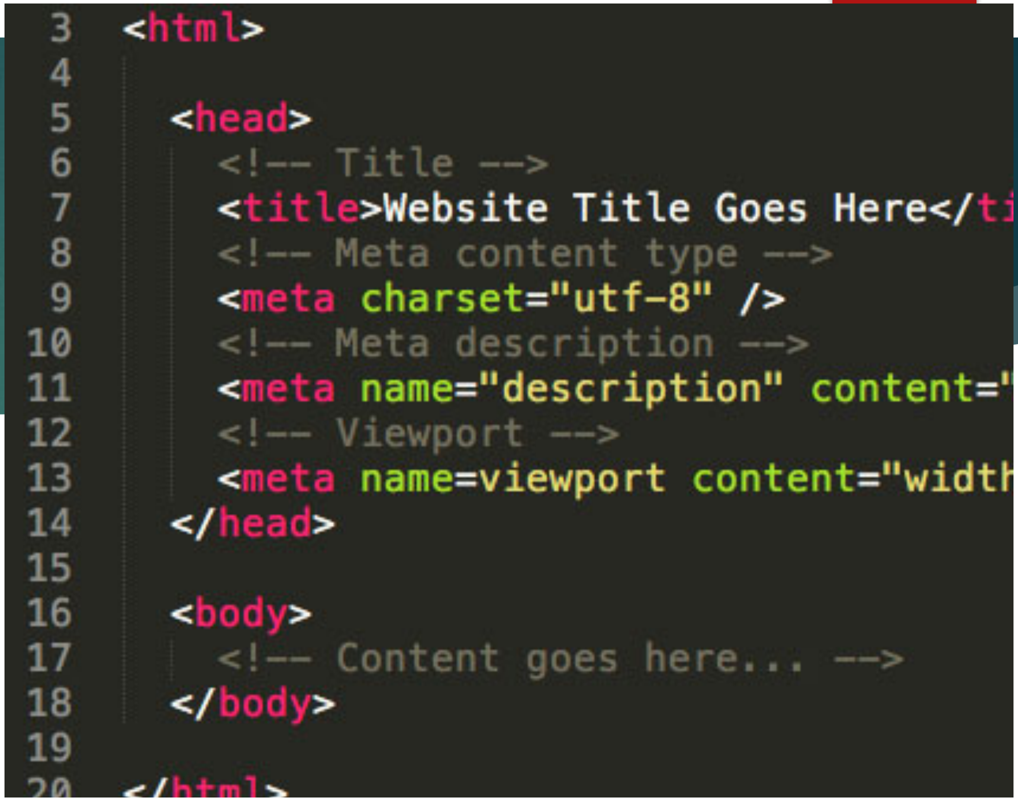
- HTML is a markup language that stands for Hypertext Markup Language, that allows web developers and website owners to create the structure of their websites. from which they can use any other coding language to enhance the design.

Focus points:

- Elements and Tags
- HTML Boilerplate
- Working with Text
- Lists
- Links and Images



```
    </head>
    <body>
    <main>
        <article>
            <h3>Html</h3>
            <p>Html is very easy to learn
        </article>
        <article>
            <h3>CSS</h3>
            <p>CSS can be used in HTML.</
        </article>
        <article>
            <h3></h3>
            <p></p>
        </article>
    </main>
    </body>
</html>
```



```
<html>

  <head>
    <!-- Title -->
    <title>Website Title Goes Here</ti
    <!-- Meta content type -->
    <meta charset="utf-8" />
    <!-- Meta description -->
    <meta name="description" content="
    <!-- Viewport -->
    <meta name=viewport content="widt
  </head>

  <body>
    <!-- Content goes here... -->
  </body>

</html>
```

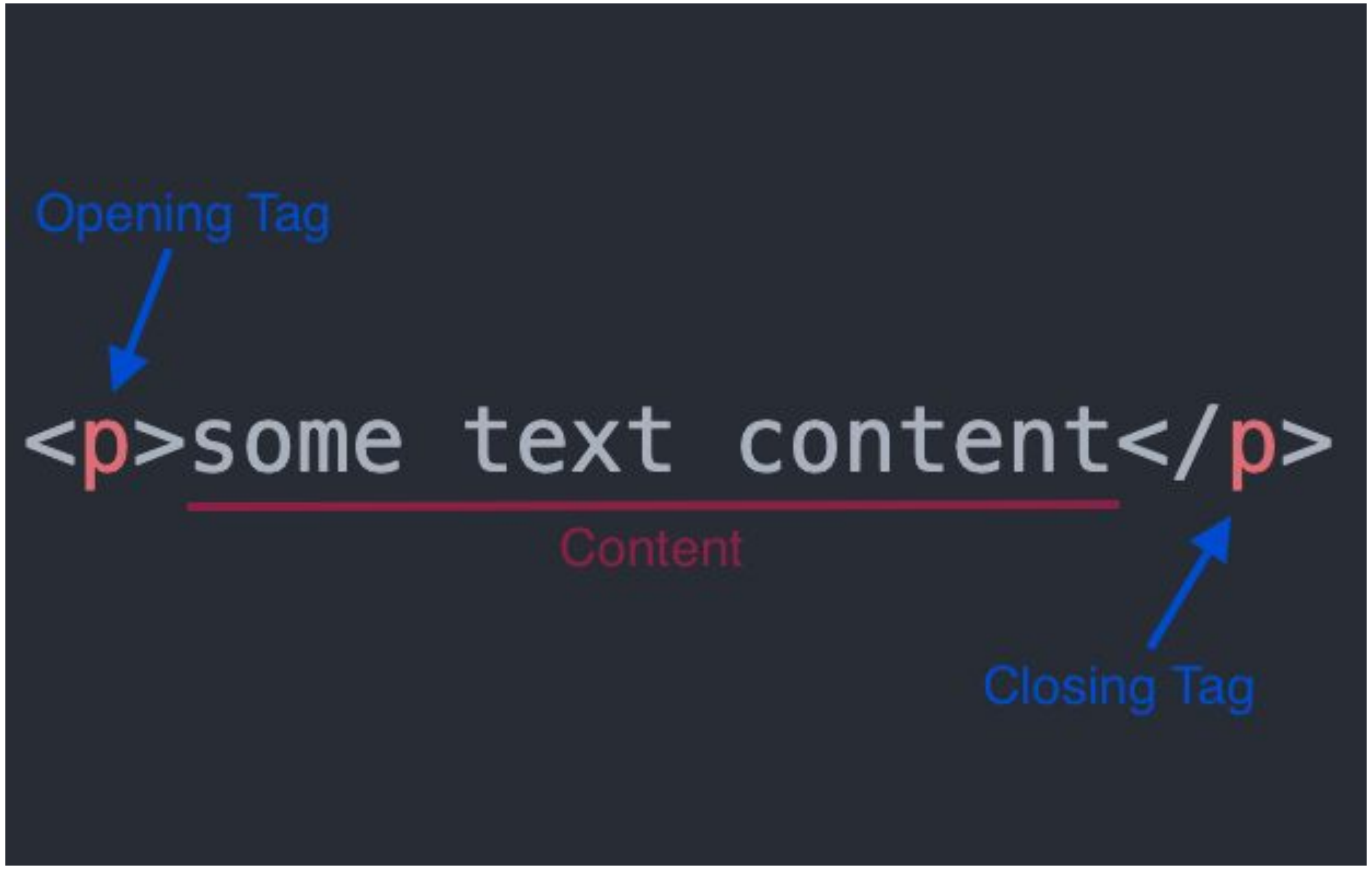
Elements and Tags

HTML

Elements and Tags

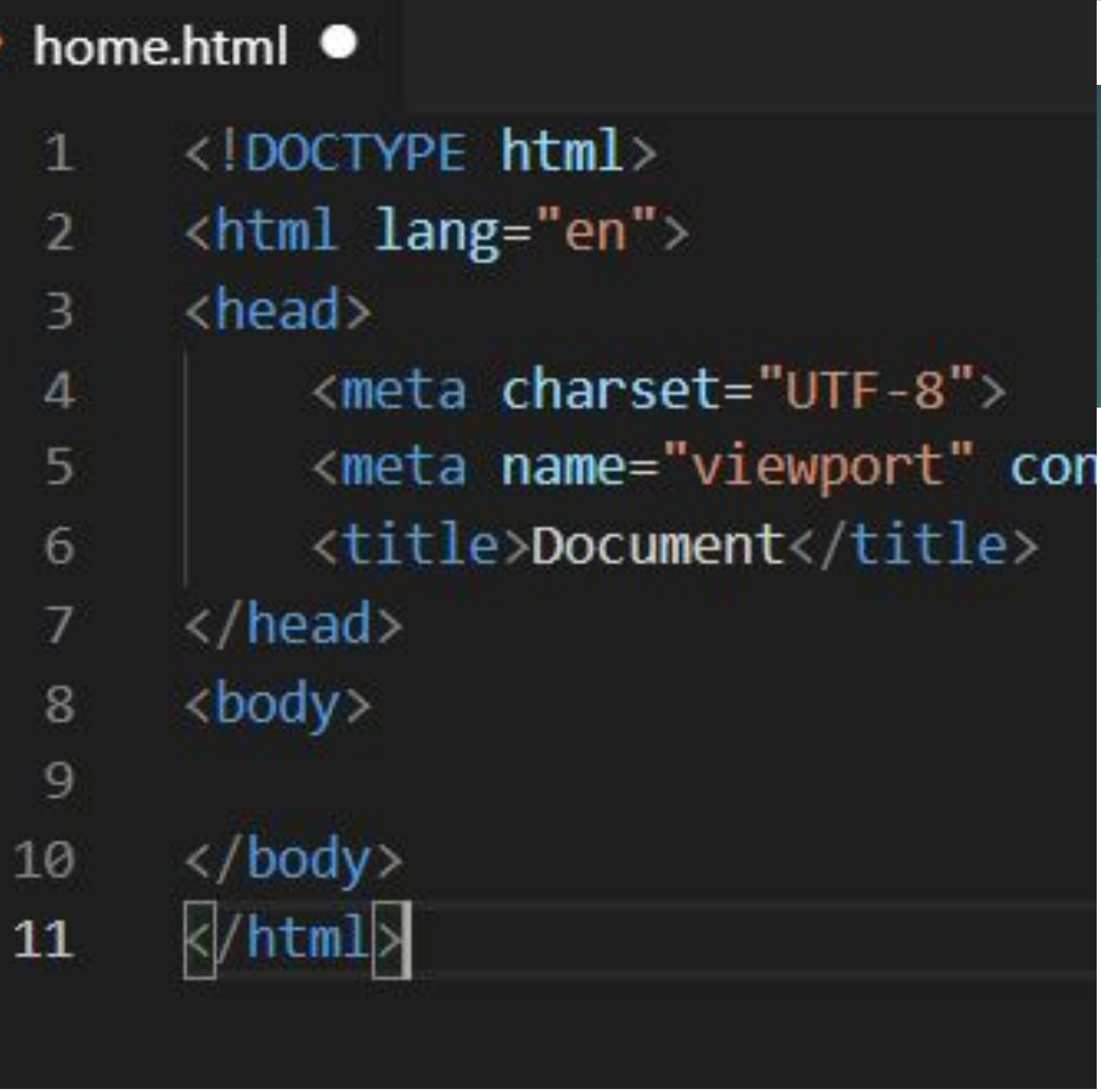
- HTML (HyperText Markup Language) defines the structure and content of webpages. We use HTML elements to create all of the paragraphs, headings, lists, images, and links that make up a typical webpage.
- Almost all elements on an HTML page are just pieces of content wrapped in opening and closing HTML tags.
- **Opening tags** tell the browser this is the start of an HTML element. They are composed of a keyword enclosed in angle brackets **<>**. For example, an opening paragraph tag looks like this: **<p>**.
- **Closing tags** tell the browser where an element ends. They are almost the same as opening tags; the only difference is that they have a forward slash before the keyword. For example, a closing paragraph tag looks like this: **</p>**.
- There are some HTML elements that do not have a closing tag. These are known as **empty elements** because they don’t wrap any content

Elements and Tags



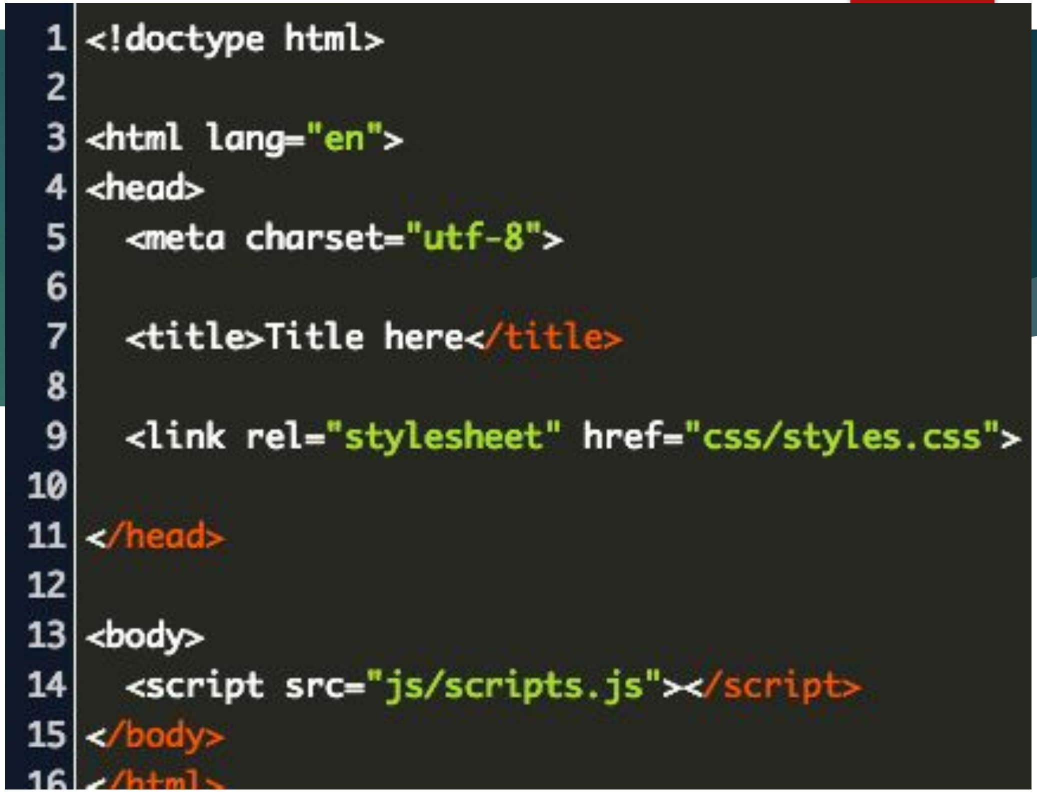
```
<p>some text content</p>
```

You can think of elements as containers for content. The opening and closing tags tell the browser what content the element contains. The browser can then use that information to determine how it should interpret and format the content.



home.html ●

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" con
    <title>Document</title>
</head>
<body>

</body>
</html>
```



```
<!doctype html>

<html lang="en">
<head>
  <meta charset="utf-8">

  <title>Title here</title>

  <link rel="stylesheet" href="css/styles.css">

</head>

<body>
  <script src="js/scripts.js"></script>
</body>
</html>
```

HTML Boilerplate

HTML

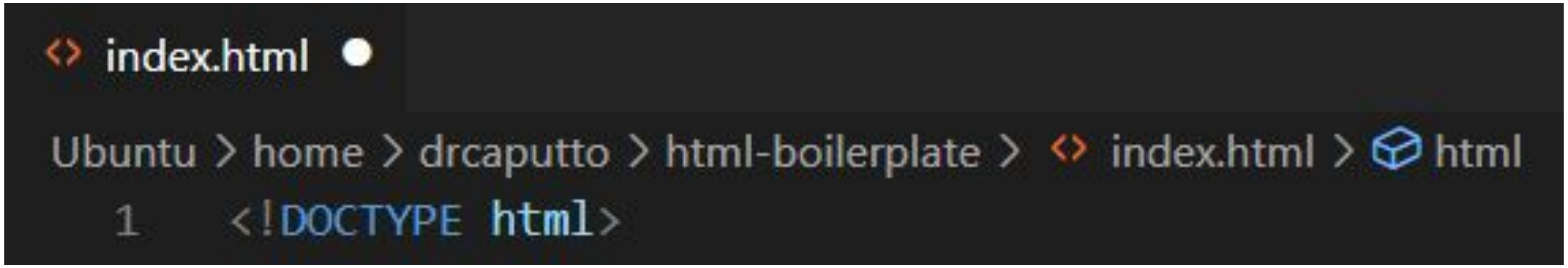
HTML Boilerplate

- All HTML documents have the same **basic structure or boilerplate** that needs to be in place before anything useful can be done. Let’s try it together:
  1. Create a new folder on your computer and name it ***html-boilerplate***. Within that folder create a new file and name it ***index.html.*** (you’re probably already familiar with a lot of different types of files, for example doc, pdf, and image files)
  2. To let the computer know you want to create an HTML file, we need to append the filename with the ***.html*** extension as we have done when creating the ***index.html*** file.
  3. It is worth noting that we named our HTML file index. We should always name the HTML file that will contain the homepage of our websites index.html. This is because web servers will by default look for an ***index.html*** page when users land on our websites - and not having one will cause big problems.

HTML Boilerplate - The DOCTYPE

- Every HTML page starts with a doctype declaration. The doctype’s purpose is to tell the browser what version of HTML it should use to render the document. The latest version of HTML is HTML5, and the doctype for that version is simply ***<!DOCTYPE html>.***

1. Open the index.html file created earlier in your text editor and add ***<!DOCTYPE html>*** to the very first line.



index.html ●
Ubuntu > home > drcaputto > html-boilerplate > index.html > html

```
<!DOCTYPE html>
```

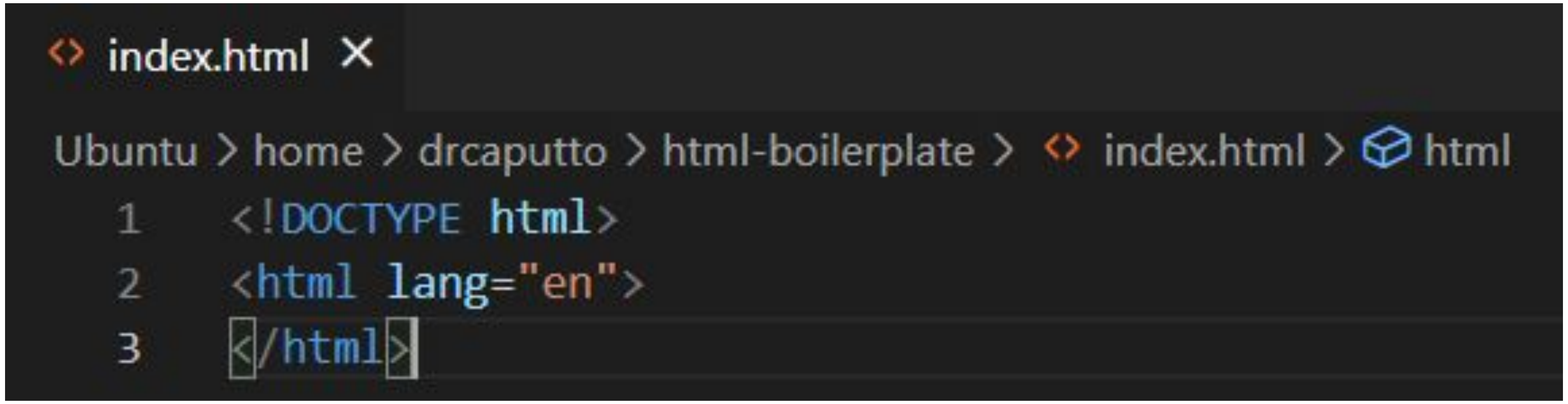
HTML Boilerplate - HTML Element

- After we declare the doctype, we need to provide an **<html>** element. This is what's known as the root element of the document, meaning that every other element in the document will be a descendant of it.
- Back in the **index.html** file, let's add the **<html>** element by typing out its opening and closing tags, like so:

<!DOCTYPE html>

<html lang="en">

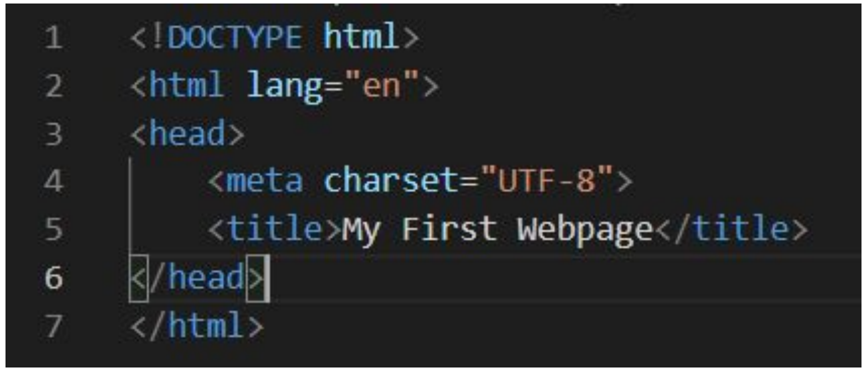
</html>


index.html
Ubuntu > home > drcaputto > html-boilerplate > index.html > html

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

HTML Boilerplate - Head Element

- The **<head>** element is where we put important meta-information about our webpages, and stuff required for our webpages to render correctly in the browser. Inside the **<head>**, we should not use any element that displays content on the webpage.
- We should always have the meta tag for the charset encoding of the webpage in the head element: **<meta charset="utf-8">**.
- Setting the encoding is very important because it ensures that the webpage will display special symbols and characters from different languages correctly in the browser.
- Another element we should always include in the head of an HTML document is the **title** element: **<title>My First Webpage</title>**

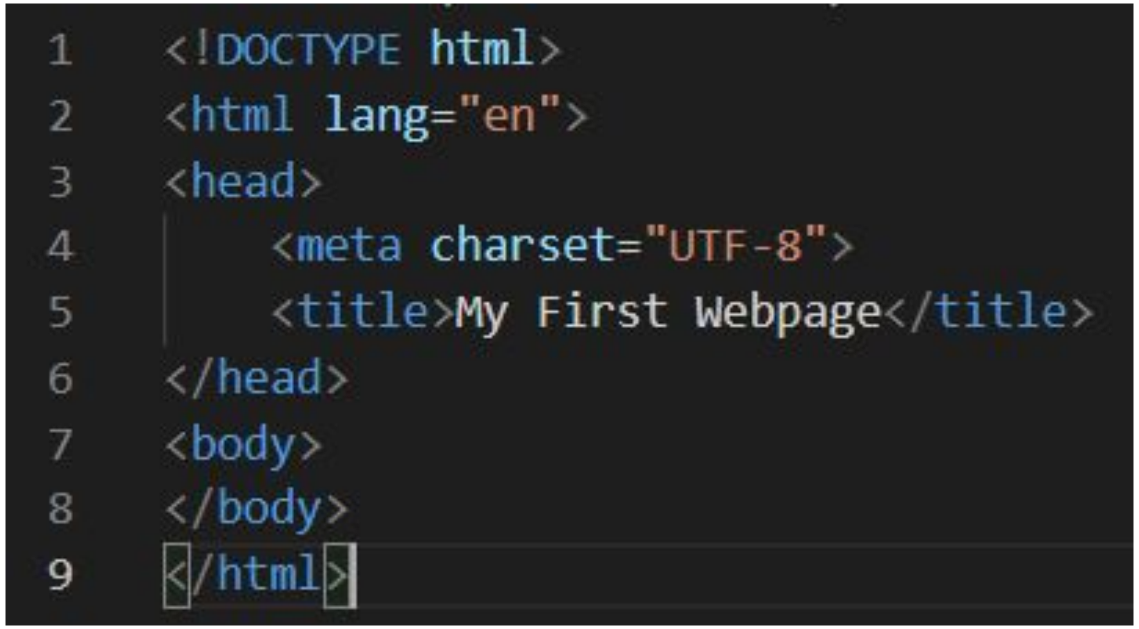


```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>My First Webpage</title>
</head>
</html>
```

HTML Boilerplate - Body Element

- The final element needed to complete the HTML boilerplate is the **<body>** element. This is where all the content that will be displayed to users will go - the text, images, lists, links, and so on.
- To complete the boilerplate, add a **body** element to the index.html file. The body element also goes within the HTML element and is always below the head element, like so:

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="UTF-8">
  <title>My First Webpage</title>
 </head>
 <body>
 </body>
</html>
```



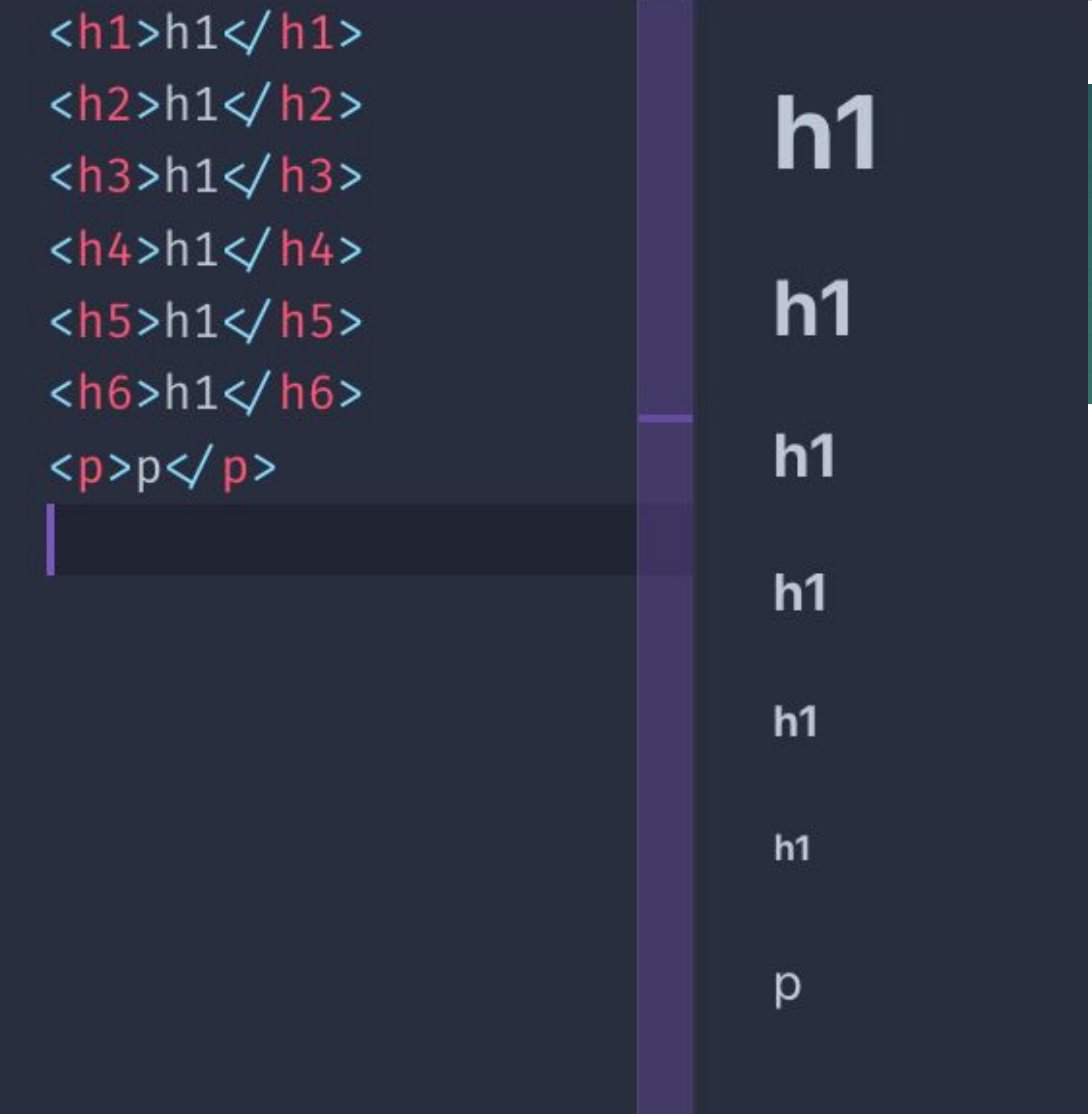
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>My First Webpage</title>
</head>
<body>
</body>
</html>
```

HTML Boilerplate - Viewing HTML Files in the Browser

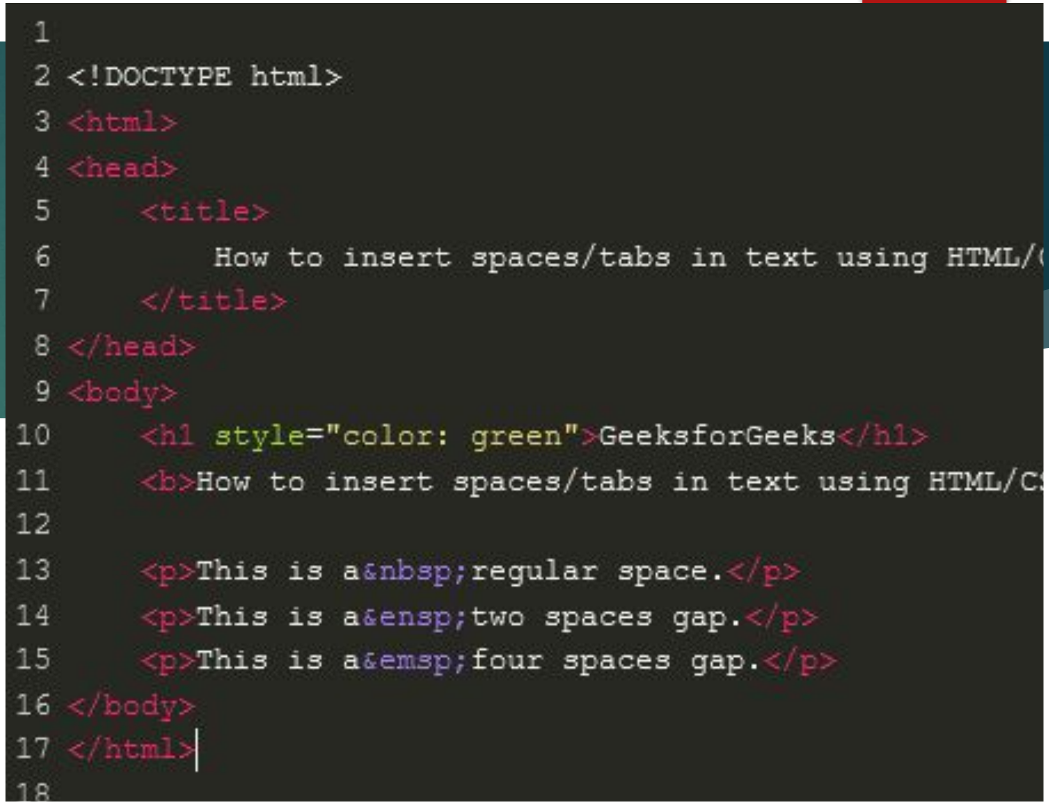
The HTML boilerplate in the index.html file is complete at this point, to view it in the browser there are a couple of different options:

- drag and drop an HTML file from your text editor into the address bar of your browser.
- find the HTML file in your file system and then double click it. This will open up the file in the default browser your system uses.
- use the terminal to open the file in your browser.

*** **VSCode** has a built-in shortcut you can use for generating all the boilerplate in one go. Please note that this shortcut only works while editing a file with the ‘.html’ extension or a text file with the HTML language already selected. To trigger the shortcut, just enter **!** on the first line. This will bring up a couple of options. Press the enter key to choose the first one, you should have all the boilerplate populated for you.



```
<h1>h1</h1>
<h2>h1</h2>
<h3>h1</h3>
<h4>h1</h4>
<h5>h1</h5>
<h6>h1</h6>
<p>p</p>
```



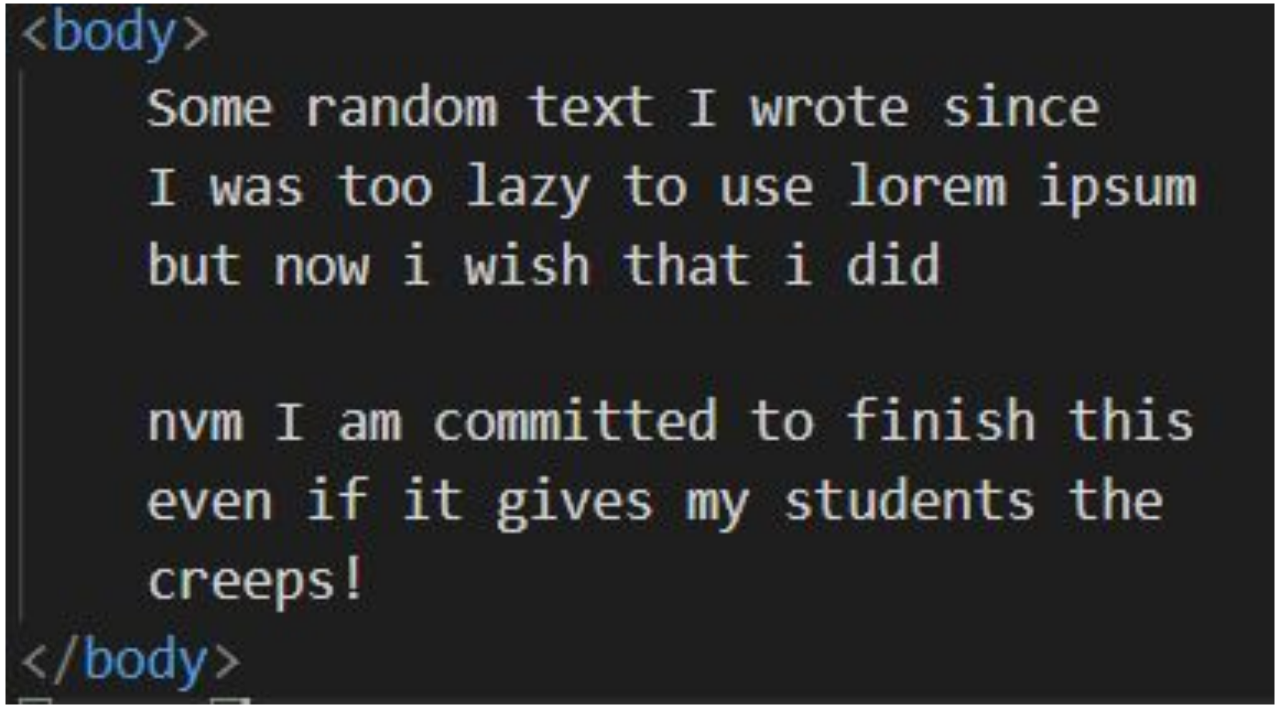
```

<!DOCTYPE html>
<html>
<head>
    <title>
        How to insert spaces/tabs in text using HTML/(
    </title>
</head>
<body>
    <h1 style="color: green">GeeksforGeeks</h1>
    <b>How to insert spaces/tabs in text using HTML/C

    <p>This is a&nbsp;regular space.</p>
    <p>This is a&ensp;two spaces gap.</p>
    <p>This is a&emsp;four spaces gap.</p>
</body>
</html>
```

Working with Text

HTML

Working with Text - Paragraphs

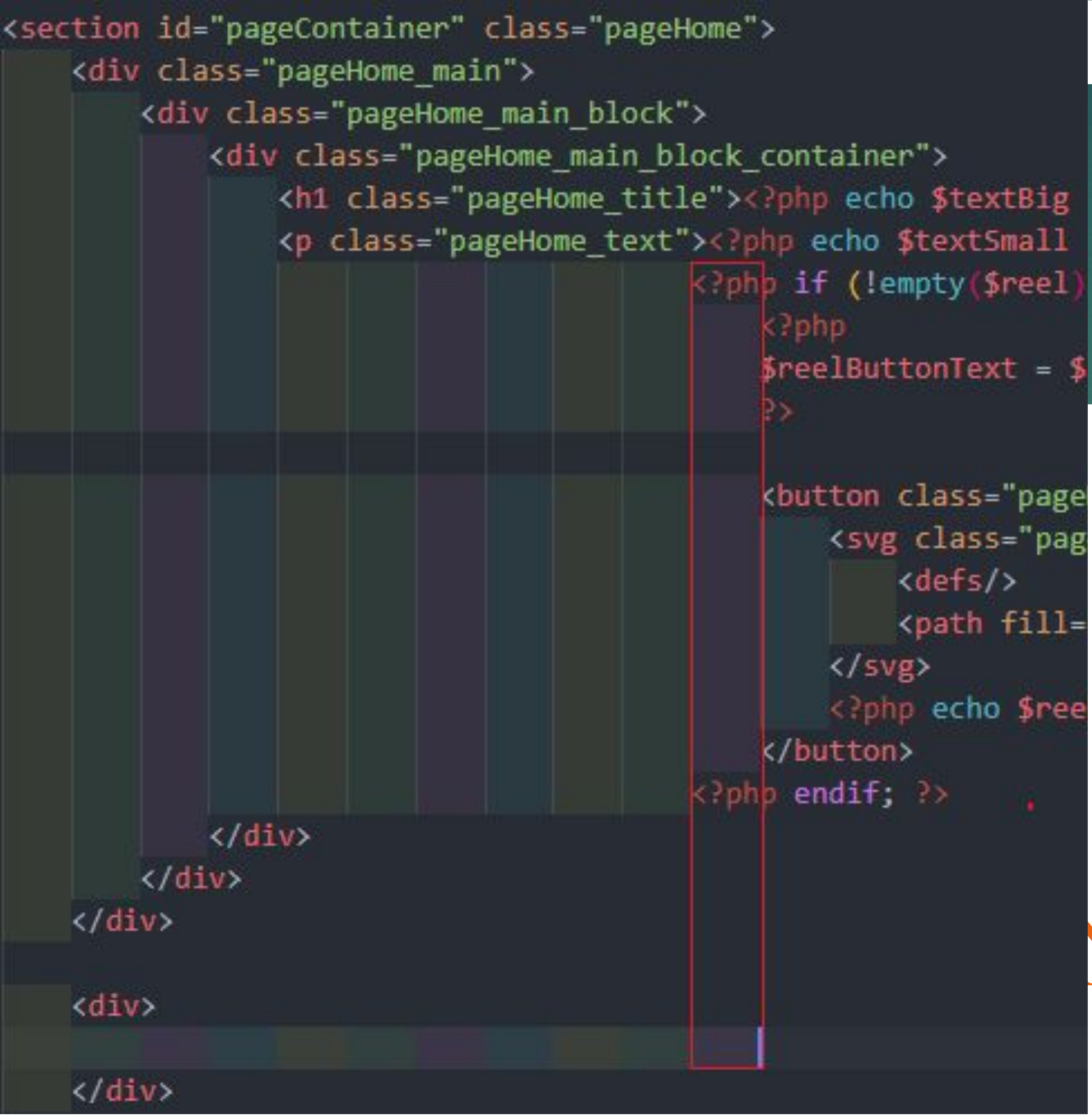
- If we want to create paragraphs in HTML, we need to use the paragraph element, which will add a newline after each of our paragraphs. A paragraph element is defined by wrapping text content with a ***<p>*** tag.



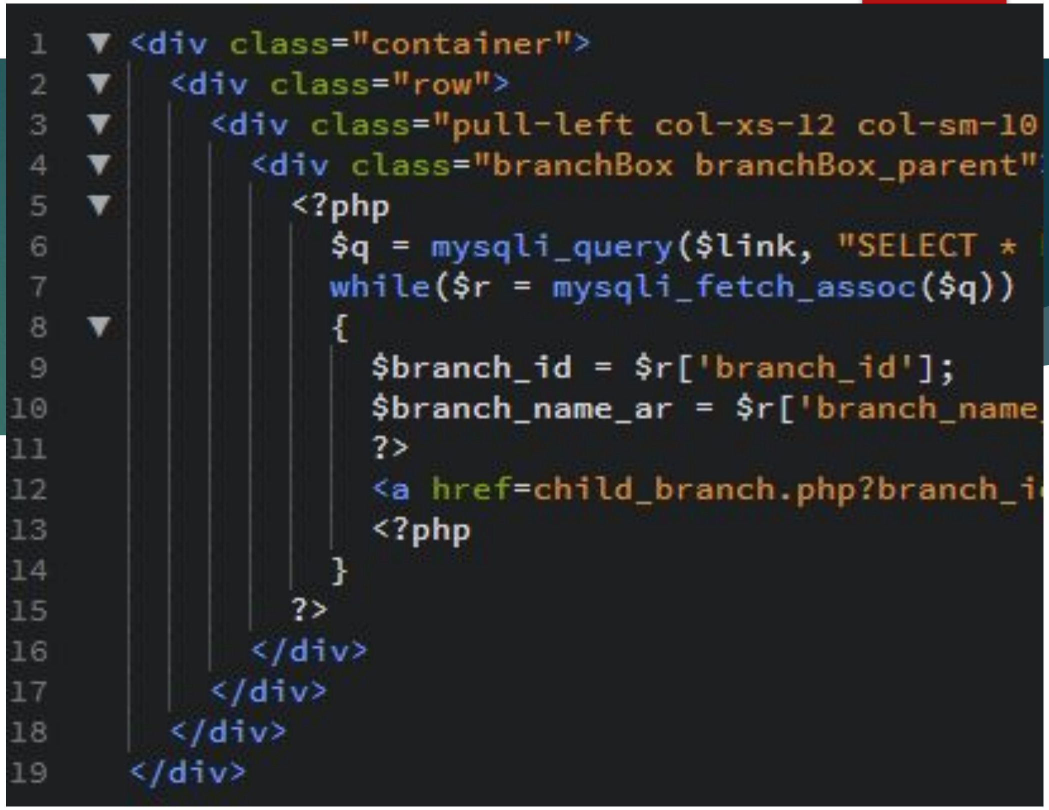
```
<body>
    Some random text I wrote since
    I was too lazy to use lorem ipsum
    but now i wish that i did

    nvm I am committed to finish this
    even if it gives my students the
    creeps!
</body>
```

Most common ways to influence text:

- Headings **<h1> - <h6>**
- Strong element **<strong></strong>**
- Em element *<em></em>*
- Comments <!– some text -->



```
<section id="pageContainer" class="pageHome">
    <div class="pageHome_main">
        <div class="pageHome_main_block">
            <div class="pageHome_main_block_container">
                <h1 class="pageHome_title"><?php echo $textBig
                <p class="pageHome_text"><?php echo $textSmall
                                        <?php if (!empty($reel)
                                            <?php
                                            $reelButtonText = $
                                            ?>

                                            <button class="page
                                                <svg class="pag
                                                    <defs/>
                                                    <path fill=
                                                </svg>
                                                <?php echo $ree
                                            </button>
                                        <?php endif; ?>
            </div>
        </div>
    </div>

    <div>

    </div>
```



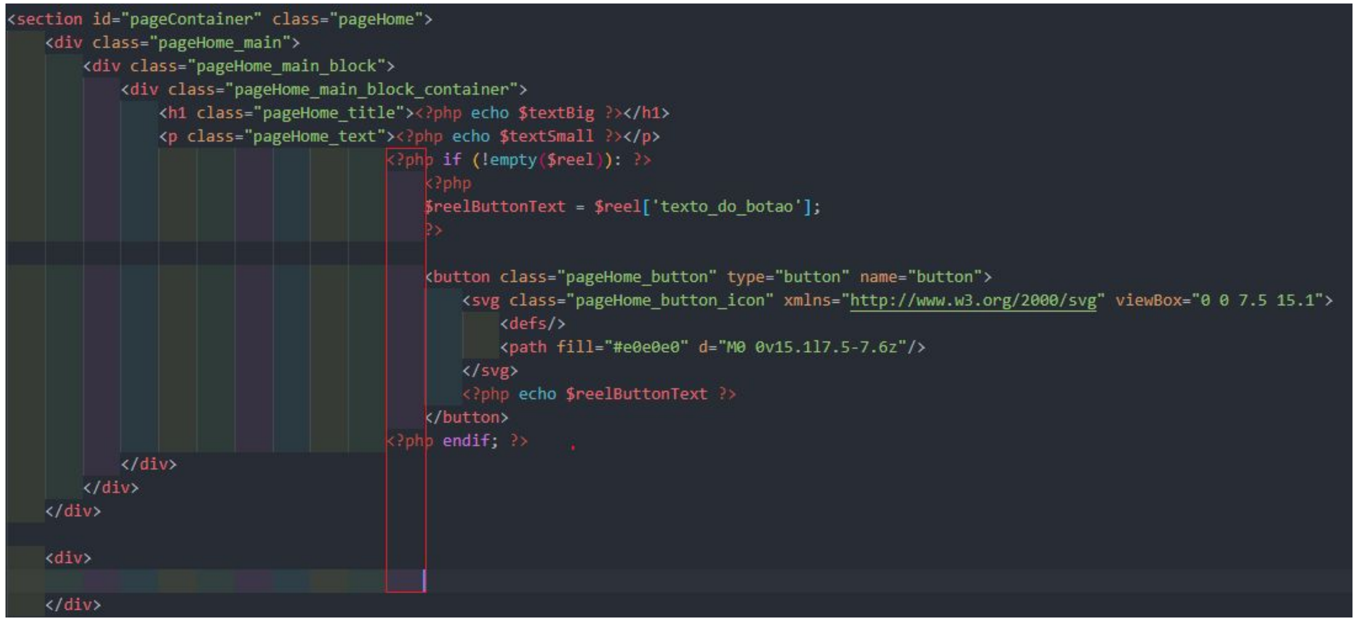
```
<div class="container">
  <div class="row">
    <div class="pull-left col-xs-12 col-sm-10
      <div class="branchBox branchBox_parent"
        <?php
          $q = mysqli_query($link, "SELECT *
          while($r = mysqli_fetch_assoc($q))
          {
            $branch_id = $r['branch_id'];
            $branch_name_ar = $r['branch_name
            ?>
            <a href=child_branch.php?branch_i
            <?php
          }
        ?>
      </div>
    </div>
  </div>
</div>
```

Nesting and Indentation

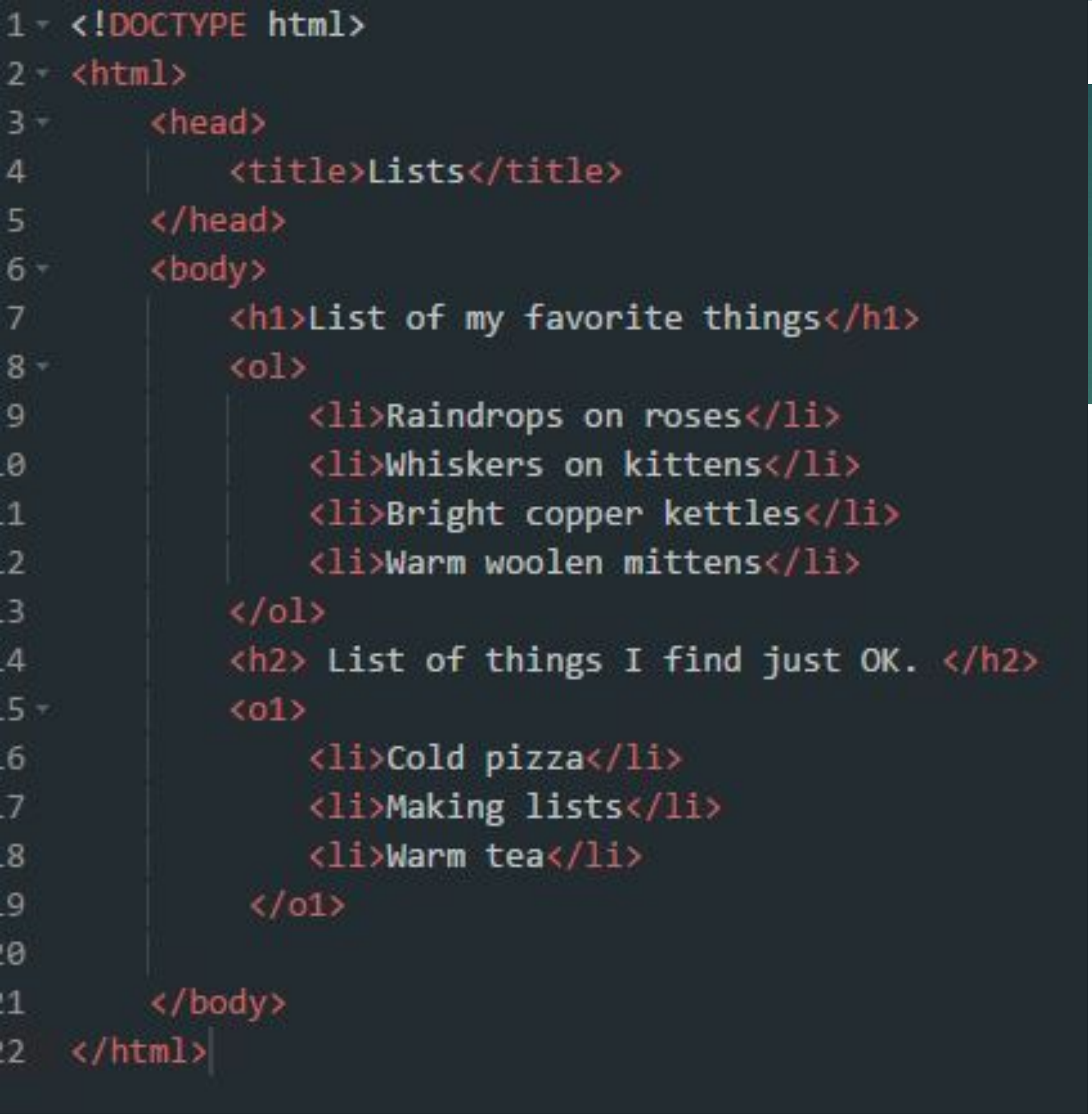
HTML

Nesting and Indentation

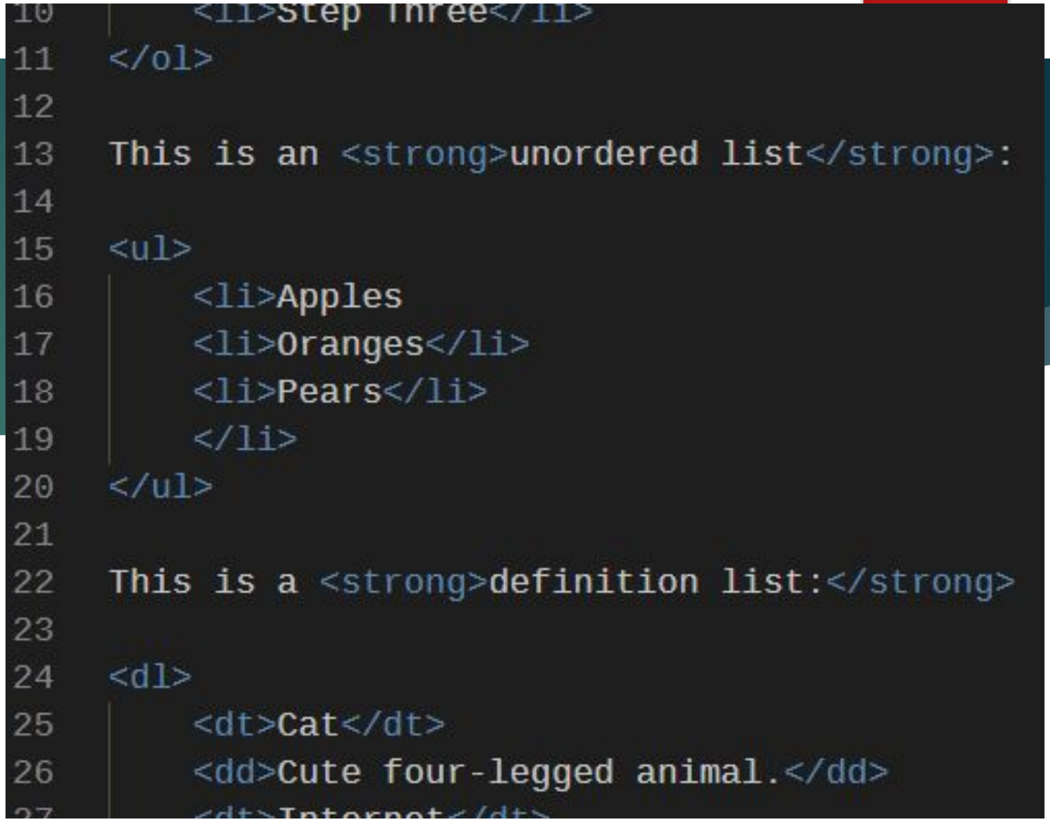
- When we nest elements within other elements, we create a parent and child relationship between them. The nested elements are the children and the element they are nested within is the parent.
- Just as in human relationships, HTML parent elements can have many children. Elements at the same level of nesting are considered to be siblings.
- We use indentation to make the level of nesting clear and readable for ourselves and other developers who will work with our HTML in the future. It is recommended to indent any child elements by two spaces(one tab).



```
<section id="pageContainer" class="pageHome">
    <div class="pageHome_main">
        <div class="pageHome_main_block">
            <div class="pageHome_main_block_container">
                <h1 class="pageHome_title"><?php echo $textBig ?></h1>
                <p class="pageHome_text"><?php echo $textSmall ?></p>
                                        <?php if (!empty($reel)): ?>
                                            <?php
                                            $reelButtonText = $reel['texto_do_botao'];
                                            ?>

                                            <button class="pageHome_button" type="button" name="button">
                                                <svg class="pageHome_button_icon" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 7.5 15.1">
                                                    <defs/>
                                                    <path fill="#e0e0e0" d="M0 0v15.1l7.5-7.6z"/>
                                                </svg>
                                                <?php echo $reelButtonText ?>
                                            </button>
                                        <?php endif; ?>
            </div>
        </div>
    </div>

    <div>

    </div>
```



```html
<!DOCTYPE html>
<html>
    <head>
        <title>Lists</title>
    </head>
    <body>
        <h1>List of my favorite things</h1>
        <ol>
            <li>Raindrops on roses</li>
            <li>Whiskers on kittens</li>
            <li>Bright copper kettles</li>
            <li>Warm woolen mittens</li>
        </ol>
        <h2> List of things I find just OK. </h2>
        <o1>
            <li>Cold pizza</li>
            <li>Making lists</li>
            <li>Warm tea</li>
         </o1>

    </body>
</html>
```



```html
    <li>Step Three</li>
</ol>

This is an <strong>unordered list</strong>:

<ul>
    <li>Apples
    <li>Oranges</li>
    <li>Pears</li>
    </li>
</ul>

This is a <strong>definition list:</strong>

<dl>
    <dt>Cat</dt>
    <dd>Cute four-legged animal.</dd>
    <dt>Internet</dt>
```

Lists
Unordered & Ordered

HTML

Unordered Lists

- Unordered lists are created using the **<ul>** element, and each item within the list is created using the list item element **<li>.**
- Each list item in an unordered list begins with a **bullet point**.
- If you want to have a list of items where the order doesn't matter, like a shopping list of items that can be bought in any order, then you can use an unordered list.

```
<ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```

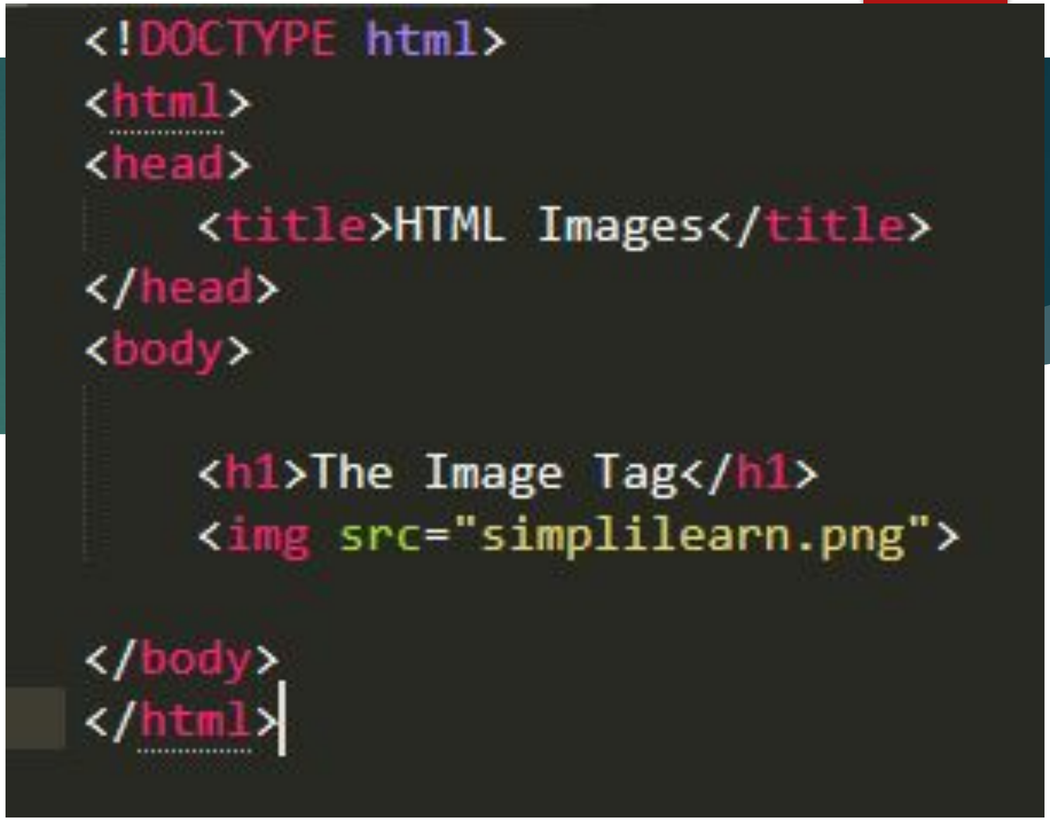
- Coffee
- Tea
- Milk

Ordered Lists

- Ordered lists are created using the **<ol>** element. Each individual item in them is again created using the list item element **<li>**. However, each list item in an ordered list begins with a **number** instead.
- If you instead want to create a list of items where the order does matter, like step-by-step instructions for a recipe, or your top 10 favorite TV shows, then you can use an ordered list.

```html
<ol>
  <li>China</li>
  <li>Serbia</li>
  <li>Disneyland</li>
</ol>
```

1. China
2. Serbia
3. Disneyland



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

      <h1>The Image Tag</h1>
      <img src="simplilearn.png">

</body>
</html>
```

Links and Images

HTML

Links and Images

- Links are one of the key features of HTML. They allow us to link to other HTML pages on the web. In fact, this is why it’s called the web.
- Websites would be fairly boring if they could only display text. Luckily, HTML provides a wide variety of elements for displaying all sorts of different media. The most widely used of these is the image element.

<a> + <img>

Links

- **Anchor Elements -** To create a link in HTML, we use the anchor element. An anchor element is defined by wrapping the text or another HTML element we want to be a link with an **<a>** tag
- An HTML attribute gives additional information to an HTML element and always goes in the element's opening tag. An attribute is made up of two parts, a name, and a value. In our case, we need to add a **href** (hyperlink reference) attribute to the opening anchor tag. The value of the **href** attribute is the destination we want our link to go to.

```
<a href="https://theuselessweb.com/">click me</a>
```

Links

- **Anchor Elements -** To create a link in HTML, we use the anchor element. An anchor element is defined by wrapping the text or another HTML element we want to be a link with an **<a>** tag
- An HTML attribute gives additional information to an HTML element and always goes in the element's opening tag. An attribute is made up of two parts, a name, and a value. In our case, we need to add a **href** (hyperlink reference) attribute to the opening anchor tag. The value of the **href** attribute is the destination we want our link to go to.

```
<a href="https://theuselessweb.com/">click me</a>
```

- **Absolute Links -** Links to pages on other websites on the internet are called absolute links. A typical absolute link will be made up of the following parts: **protocol://domain/path**. An absolute link will always contain the protocol and domain of the destination. **(https://theuselessweb.com)**
- **Relative Links -** Links to other pages within our own website are called relative links. Relative links do not include the domain name, since it is another page on the same site, it assumes the domain name will be the same as the page we created the link on.

Images

- To display an image in HTML we use the **<img> element**. Unlike the other elements we have encountered so far, the **<img>** element is **empty**. Which means it doesn’t have a closing tag.
- Instead of wrapping content with an opening and closing tag, it embeds an image into the page using a **src** attribute which tells the browser where the image file is located. The **src** attribute works much like the **href** attribute for anchor tags. It can embed an image using both absolute and relative paths

```
<img src="image location, absolute or relative">
```

- **Alt attribute -** Besides the **src** attribute, every image element should also have an **alt** (alternative text) attribute.
- The **alt** attribute is used to describe an image. It will be used in place of the image if it cannot be loaded. It is also used with screen readers to describe what the image is to visually impaired users.

Assignment

aleksa.fd.tutor@gmail.com

Thank you for your attention!