

# HTML for Beginners

## Table of Contents

What is a Code Editor? .....	2
Popular Code Editors .....	2
HTML Introduction .....	2
HTML - What is it Exactly? .....	3
HTML Syntax .....	4
Saving Files in VS Code .....	5
HTML Anatomy .....	6
Headings .....	6
Paragraphs .....	7
HTML Headings, Paragraphs and Emphasis.....	8
Indenting Code in HTML .....	10
HTML lists.....	11
HTML links.....	13
Navigating Pages.....	15
Images in HTML .....	16
Creating tables using HTML .....	16
Introducing the Div .....	18
Semantic sectioning.....	19
HTML forms .....	19
The various input element types in HTML .....	20
The select dropdown in HTML.....	21
The Inspect tool .....	22
Using the data attribute in HTML .....	22
Commenting Out Code.....	23

## What is a Code Editor?

A code editor is a software application used to write and edit source code for computer programs. It provides a text editor with features and tools specifically designed for writing and editing code, such as syntax highlighting, auto-completion, code formatting, code navigation, debugging, and more. Code editors are essential tools for software developers and programmers working on projects in various programming languages such as Python, Java, C++, JavaScript, and many more. Popular examples of code editors include Visual Studio Code, Sublime Text, Atom, and Notepad++.

## Popular Code Editors

There are many code editor options available for different programming languages and platforms. Here are some of the most popular code editors:

Visual Studio Code - a free, open-source, cross-platform code editor developed by Microsoft that supports many programming languages and has a large community of users and extensions.

Sublime Text - a lightweight, fast, and powerful code editor with a customizable interface and a range of features, such as multi-selection editing and Python-based plugin API.

Atom - a free, open-source, cross-platform code editor developed by GitHub that is highly customizable, has a built-in package manager, and supports many programming languages.

Notepad++ - a free, open-source code editor for Windows that supports many programming languages, has syntax highlighting, and offers a range of plugins for added functionality.

IntelliJ IDEA - a powerful, commercial code editor designed specifically for Java and related technologies, such as Kotlin, Groovy, and Scala, with features such as code completion, refactoring, and debugging.

PyCharm - a powerful, commercial code editor designed specifically for Python development, with features such as intelligent code completion, debugging, and code analysis.

Eclipse - a free, open-source, cross-platform code editor primarily used for Java development, but supports other languages and technologies through plugins.

These are just a few examples of the many code editor options available. The best code editor for you will depend on your personal preferences, the programming language you are working with, and the specific features and tools you need for your work.

## HTML Introduction

HTML, which stands for HyperText Markup Language, is a markup language used to create and structure content on the web. It is a fundamental building block of the web and is used to create web pages and web applications.

HTML consists of a series of elements, which are enclosed in angled brackets (< >) and may have attributes that provide additional information about the element.

These elements and attributes are used to describe the structure and content of a web page.

HTML is used to define the basic structure of a web page, including headings, paragraphs, lists, and links. It also allows for the inclusion of images, videos, audio, and other multimedia content. HTML can be used in conjunction with other technologies such as CSS (Cascading Style Sheets) and JavaScript to create interactive and visually appealing web pages and applications.

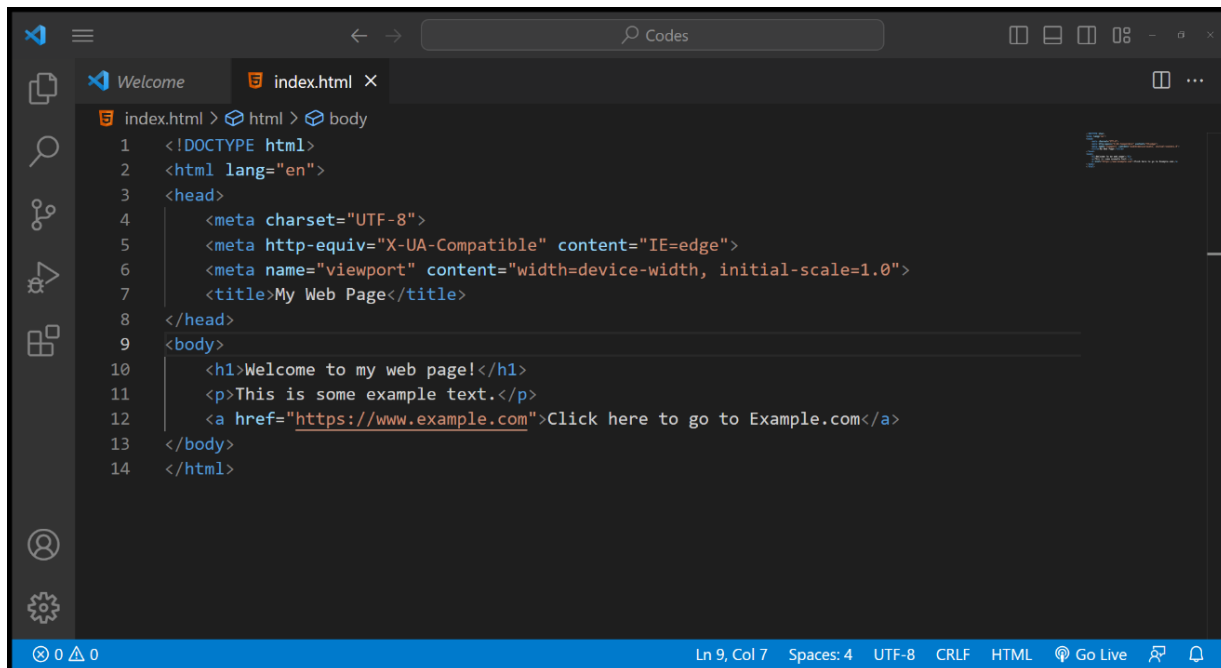
The latest version of HTML is HTML5, which includes many new features and improvements over previous versions, such as new semantic elements, multimedia support, and improved accessibility. HTML is an essential language for web developers and is a foundational skill for anyone interested in creating web content.

## HTML - What is it Exactly?

HTML, which stands for HyperText Markup Language, is a markup language used to create and structure content on the web. It is the standard markup language used to create web pages and web applications.

HTML works by using markup tags to define the structure and content of a web page. These tags are enclosed in angled brackets (< >) and provide additional information about the content they surround. HTML tags are used to create headings, paragraphs, lists, links, images, and other elements that make up a web page.

For example, the following HTML code creates a simple web page with a heading, a paragraph, and a link:

A screenshot of a code editor interface. The top bar shows a search icon and the text 'Codes'. Below the top bar, there's a tab labeled 'index.html'. The main editor area displays the following HTML code:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>My Web Page</title>
8 </head>
9 <body>
10  <h1>Welcome to my web page!</h1>
11  <p>This is some example text.</p>
12  <a href="https://www.example.com">Click here to go to Example.com</a>
13 </body>
14 </html>
```

The status bar at the bottom shows 'Ln 9, Col 7', 'Spaces: 4', 'UTF-8', 'CRLF', 'HTML', and a 'Go Live' button.

In this example, the `<html>` element defines the beginning and end of the HTML document, and the `<head>` element contains information about the document such as the title. The `<body>` element contains the visible content of the page, including the heading, paragraph, and link.

HTML is used in conjunction with other technologies such as CSS (Cascading Style Sheets) and JavaScript to create visually appealing and interactive web pages and applications.

## HTML Syntax

HTML syntax consists of a series of markup tags enclosed in angled brackets (`< >`) that define the structure and content of a web page. Here are some key elements of HTML syntax:

**Tags:** HTML markup tags define the elements of a web page. Tags are enclosed in angled brackets and may contain additional attributes that provide more information about the element. For example, the `<p>` tag is used to create a paragraph, and the `<img>` tag is used to insert an image.

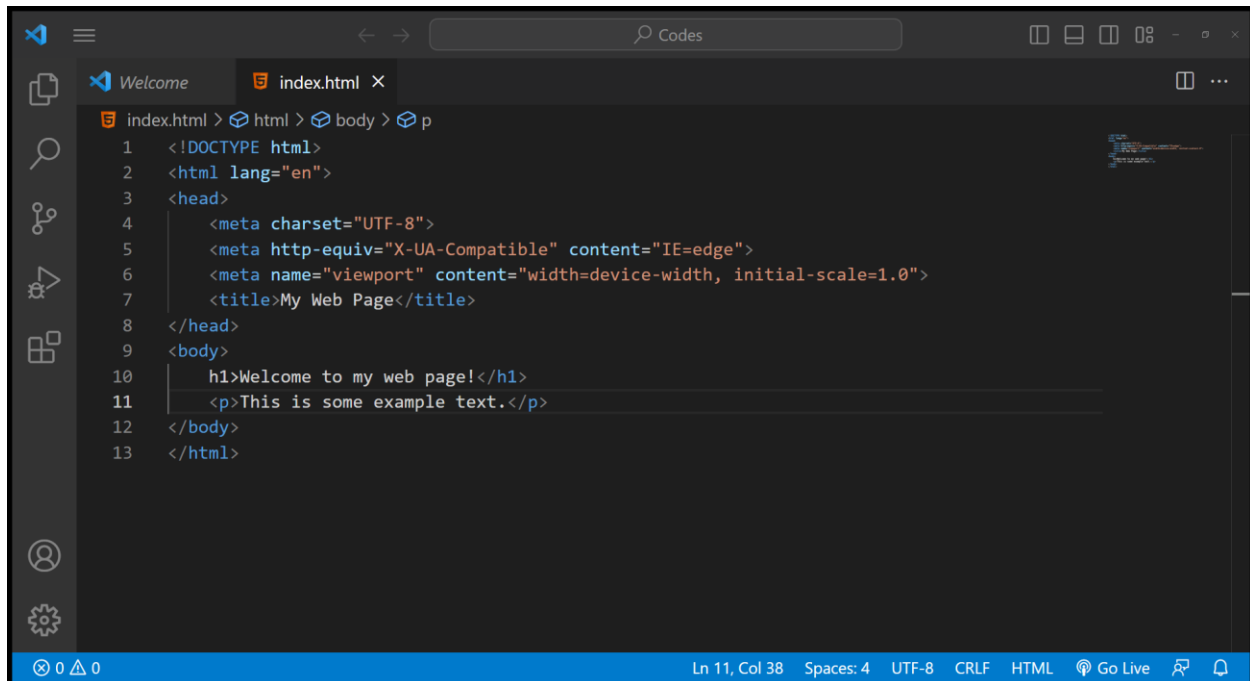
**Attributes:** HTML tags may have attributes that provide additional information about the element. Attributes are specified within the opening tag and take the form of name/value pairs. For example, the `src` attribute is used to specify the source file for an image, and the `href` attribute is used to specify the URL for a link.

**Elements:** HTML documents are structured using a hierarchical arrangement of elements. Elements can be nested within other elements, and each element has a specific purpose and meaning.

**DOCTYPE Declaration:** The DOCTYPE declaration is the first line of an HTML document and tells the browser which version of HTML is being used.

HTML document structure: An HTML document consists of an opening and closing `<html>` tag, which contains two sections: the `<head>` section and the `<body>` section. The `<head>` section contains information about the document, such as the page title and any links to external resources. The `<body>` section contains the visible content of the page.

Here is an example of the basic syntax for creating a simple HTML document:

A screenshot of the Visual Studio Code (VS Code) editor interface. The editor is open to a file named 'index.html'. The code is written in a dark theme. The HTML structure is as follows: Line 1: `<!DOCTYPE html>`; Line 2: `<html lang="en">`; Line 3: `<head>`; Line 4: `<meta charset="UTF-8">`; Line 5: `<meta http-equiv="X-UA-Compatible" content="IE=edge">`; Line 6: `<meta name="viewport" content="width=device-width, initial-scale=1.0">`; Line 7: `<title>My Web Page</title>`; Line 8: `</head>`; Line 9: `<body>`; Line 10: `<h1>Welcome to my web page!</h1>`; Line 11: `<p>This is some example text.</p>`; Line 12: `</body>`; Line 13: `</html>`. The VS Code interface includes a sidebar on the left with icons for Explorer, Search, Source Control, Run and Debug, and Extensions. The bottom status bar shows 'Ln 11, Col 38', 'Spaces: 4', 'UTF-8', 'CRLF', 'HTML', and 'Go Live'.

In this example, the DOCTYPE declaration specifies that this document is written in HTML5. The `<html>` tag encloses the entire document. The `<head>` section contains the document title, and the `<body>` section contains a heading and a paragraph of text.

## Saving Files in VS Code

To save files in VS Code, you can use either the keyboard shortcut or the File menu. Here are the steps:

1. To save a file using the keyboard shortcut, press `Ctrl+S` on Windows or `Cmd+S` on Mac. This will save the file to its current location.
2. To save a file using the File menu, click on the "File" menu in the top-left corner of the VS Code window, and then click "Save" or "Save As...".
3. If you click "Save", VS Code will save the file to its current location. If you click "Save As...", you can choose a new location and filename for the file.
4. If the file you are saving has not been saved before, VS Code will open a "Save As" dialog box, which allows you to choose a location and filename for the file.

5. Once you have chosen a location and filename for the file, click the "Save" button to save the file.

It's a good practice to save your files frequently as you work on them, to avoid losing any changes due to power outages, system crashes, or other issues.

## HTML Anatomy

HTML anatomy refers to the basic structure of an HTML document, which consists of several components that work together to create a web page. Here are the main components of an HTML document:

**DOCTYPE declaration:** This is the first line of an HTML document and specifies the version of HTML that the document is written in. For example, the DOCTYPE declaration for HTML5 is `<!DOCTYPE html>`.

**HTML tag:** This tag encloses the entire HTML document and is usually the first tag in the document. It consists of an opening `<html>` tag and a closing `</html>` tag.

**Head section:** This section contains metadata about the document, such as the page title, links to stylesheets or scripts, and other information that is not visible on the page itself. The head section is enclosed by the `<head>` and `</head>` tags.

**Body section:** This section contains the visible content of the page, such as headings, paragraphs, images, and other elements. The body section is enclosed by the `<body>` and `</body>` tags.

**Elements:** HTML documents are composed of various elements that define the structure and content of the page. Elements are enclosed in angled brackets, and may include attributes that provide additional information about the element. For example, the `<p>` element is used to create a paragraph of text, and the `<img>` element is used to insert an image.

**Attributes:** Attributes provide additional information about HTML elements. They are specified within the opening tag of an element and take the form of name/value pairs. For example, the `src` attribute is used to specify the URL of an image, and the `href` attribute is used to specify the URL of a link.

By understanding the anatomy of an HTML document, you can create well-structured and semantically meaningful web pages that are easy to read and understand.

## Headings

Headings in HTML are used to indicate the hierarchical structure of a web page's content. There are six levels of headings, each represented by a different HTML tag:

**<h1>:** This is the highest-level heading, used for main page headings. It is usually displayed in the largest font size.

**<h2>:** This is the second level heading, used for subheadings. It is usually displayed in a smaller font size than the h1 heading.

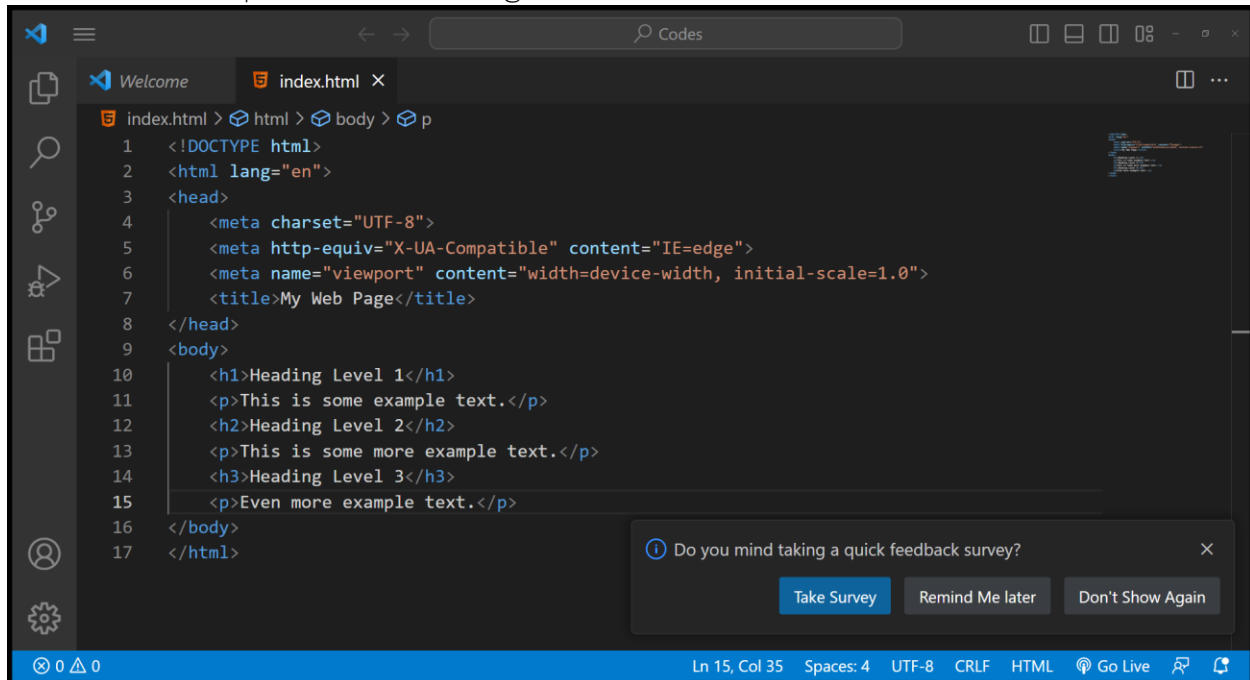
<h3>: This is the third level heading, used for sub-subheadings. It is usually displayed in a smaller font size than the h2 heading.

<h4>: This is the fourth level heading, used for sub-sub-subheadings. It is usually displayed in a smaller font size than the h3 heading.

<h5>: This is the fifth level heading, used for sub-sub-sub-subheadings. It is usually displayed in a smaller font size than the h4 heading.

<h6>: This is the lowest level heading, used for minor headings or labels. It is usually displayed in a smaller font size than the h5 heading.

Here is an example of how headings can be used in an HTML document:

A screenshot of a code editor window titled 'index.html'. The editor shows the following HTML code:

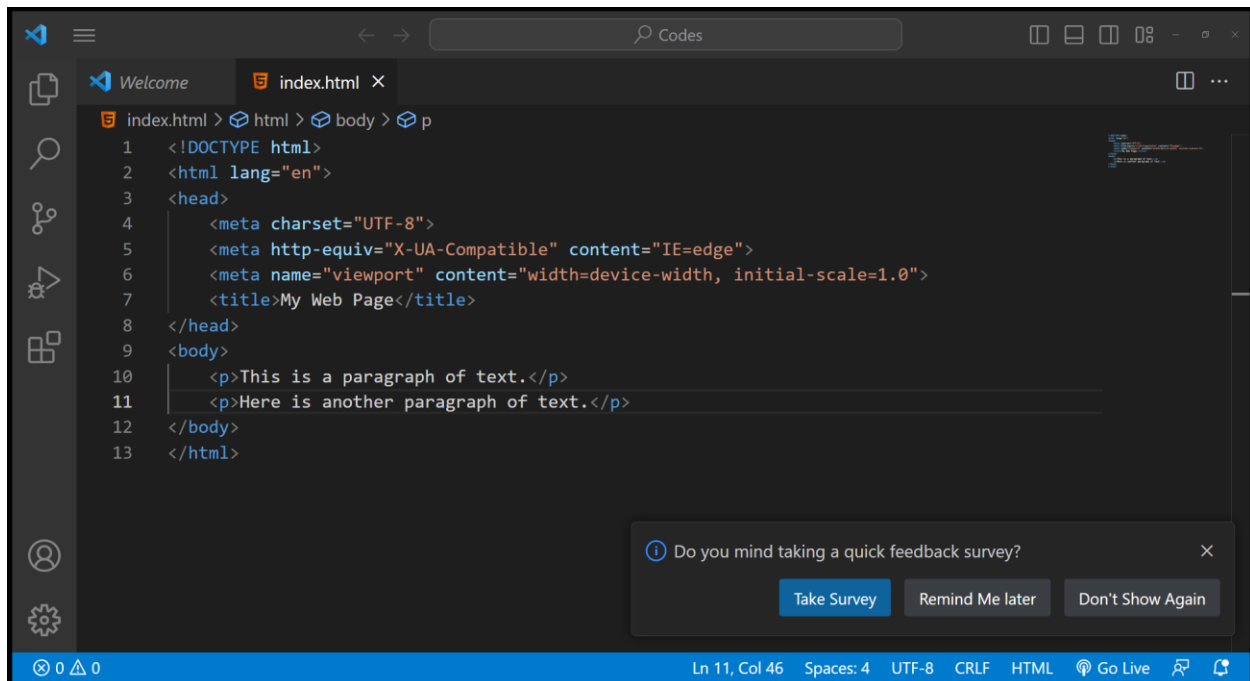
```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>My Web Page</title>
8 </head>
9 <body>
10  <h1>Heading Level 1</h1>
11  <p>This is some example text.</p>
12  <h2>Heading Level 2</h2>
13  <p>This is some more example text.</p>
14  <h3>Heading Level 3</h3>
15  <p>Even more example text.</p>
16 </body>
17 </html>
```

The editor interface includes a sidebar on the left with icons for file explorer, search, and other tools. A status bar at the bottom shows 'Ln 15, Col 35', 'Spaces: 4', 'UTF-8', 'CRLF', 'HTML', and a 'Go Live' button. A feedback survey prompt is visible in the bottom right corner.

In this example, we have used headings to create a hierarchical structure for the content of the page. The h1 heading is used for the main heading, while the h2 and h3 headings are used for subheadings. Using headings in this way makes the page easier to read and understand, and helps search engines to understand the content and structure of the page.

## Paragraphs

In HTML, paragraphs are created using the <p> tag. This tag is used to define a block of text that should be displayed as a separate paragraph. Here is an example of how to create a paragraph in HTML:



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>My Web Page</title>
8 </head>
9 <body>
10  <p>This is a paragraph of text.</p>
11  <p>Here is another paragraph of text.</p>
12 </body>
13 </html>
```

In this example, we have created two paragraphs of text using the `<p>` tag. Each paragraph is enclosed in its own set of opening and closing `<p>` tags. The text between the tags is the content of the paragraph.

By default, paragraphs in HTML are separated by some vertical space, making them easier to read. However, the amount of space can vary depending on the browser and the CSS styling of the page. If you want more control over the spacing between paragraphs, you can use CSS to adjust the margin or padding of the `<p>` element. Using paragraphs in HTML is important for creating well-structured and readable web pages. By breaking up the content of the page into smaller, more manageable chunks, you make it easier for users to read and understand the information on the page.

## HTML Headings, Paragraphs and Emphasis

HTML provides several tags that are used for structuring and formatting content on a web page. Three of the most commonly used tags are headings, paragraphs, and emphasis.

**Headings:** As mentioned in the previous answer, headings are used to indicate the hierarchical structure of a page's content. There are six levels of headings, from `h1` to `h6`. You can use these tags to create headings of different sizes and levels of importance. For example:

```
<h1>Main Heading</h1>
```

```
<h2>Subheading</h2>
```

```
<h3>Sub-subheading</h3>
```



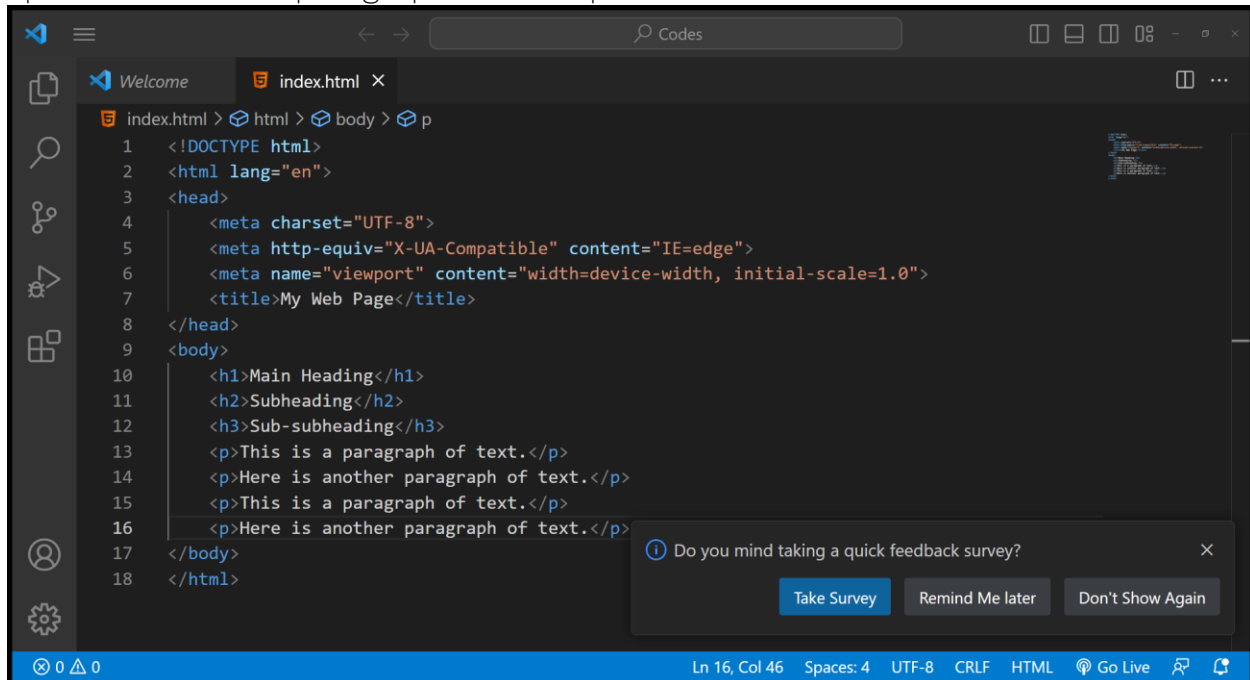
Paragraphs: Paragraphs are used to group blocks of text together. To create a paragraph in HTML, use the <p> tag. For example:

<p>This is a paragraph of text.</p>

<p>Here is another paragraph of text.</p>

<p>This is a paragraph of text.</p>

<p>Here is another paragraph of text.</p>



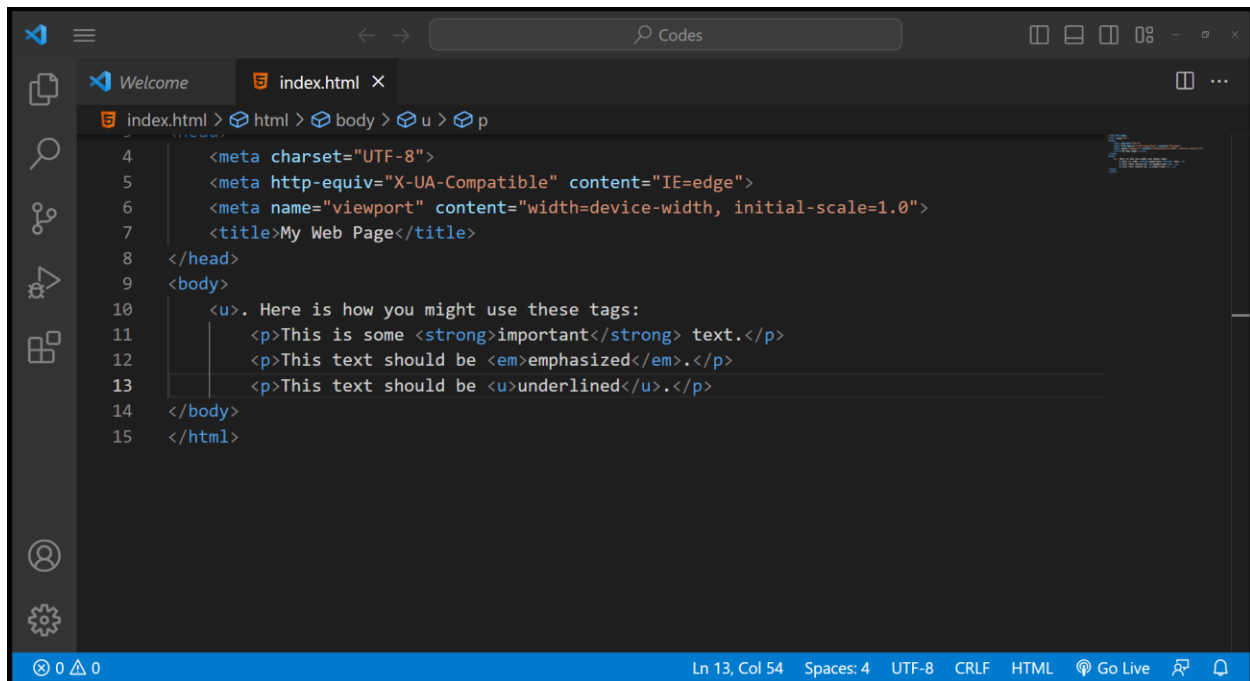
Emphasis: Emphasis is used to highlight text that should be given special attention. HTML provides several tags for creating emphasis, including <strong>, <em>, and <u>. Here is how you might use these tags:

<p>This is some <strong>important</strong> text.</p>

<p>This text should be <em>emphasized</em>.</p>

<p>This text should be <u>underlined</u>.</p>

In these examples, we have used the <strong> tag to indicate important text, the <em> tag to emphasize text, and the <u> tag to underline text. There are many other tags and attributes available in HTML for formatting and styling content, but headings, paragraphs, and emphasis are some of the most fundamental and widely used.

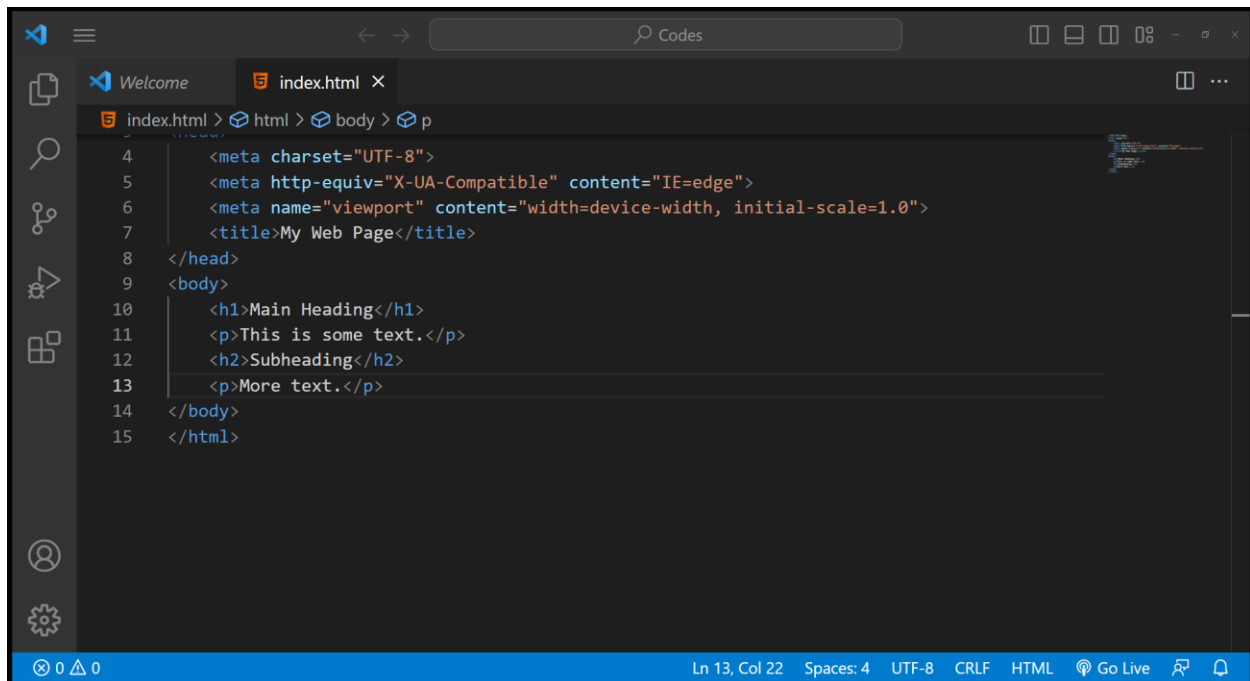


```
4 <meta charset="UTF-8">
5 <meta http-equiv="X-UA-Compatible" content="IE=edge">
6 <meta name="viewport" content="width=device-width, initial-scale=1.0">
7 <title>My Web Page</title>
8 </head>
9 <body>
10   <u>. Here is how you might use these tags:
11     <p>This is some <strong>important</strong> text.</p>
12     <p>This text should be <em>emphasized</em>.</p>
13     <p>This text should be <u>underlined</u>.</p>
14 </body>
15 </html>
```

## Indenting Code in HTML

Indenting code in HTML is not strictly necessary, but it can make your code easier to read and understand. Indentation is used to visually group related lines of code together and to indicate the structure of your HTML document.

The most common way to indent HTML code is to use tabs or spaces. You can choose whichever method you prefer, but it is important to be consistent throughout your document. Here is an example of how you might indent an HTML document using spaces:

A screenshot of a code editor window showing an HTML document named 'index.html'. The editor has a dark theme and a sidebar on the left with icons for Explorer, Search, Source Control, Run and Debug, and Extensions. The main editor area displays the following HTML code with consistent four-space indentation for nested elements:

```
4 <meta charset="UTF-8">
5 <meta http-equiv="X-UA-Compatible" content="IE=edge">
6 <meta name="viewport" content="width=device-width, initial-scale=1.0">
7 <title>My Web Page</title>
8 </head>
9 <body>
10 <h1>Main Heading</h1>
11 <p>This is some text.</p>
12 <h2>Subheading</h2>
13 <p>More text.</p>
14 </body>
15 </html>
```

The status bar at the bottom indicates 'Ln 13, Col 22', 'Spaces: 4', 'UTF-8', 'CRLF', 'HTML', and includes a 'Go Live' button.

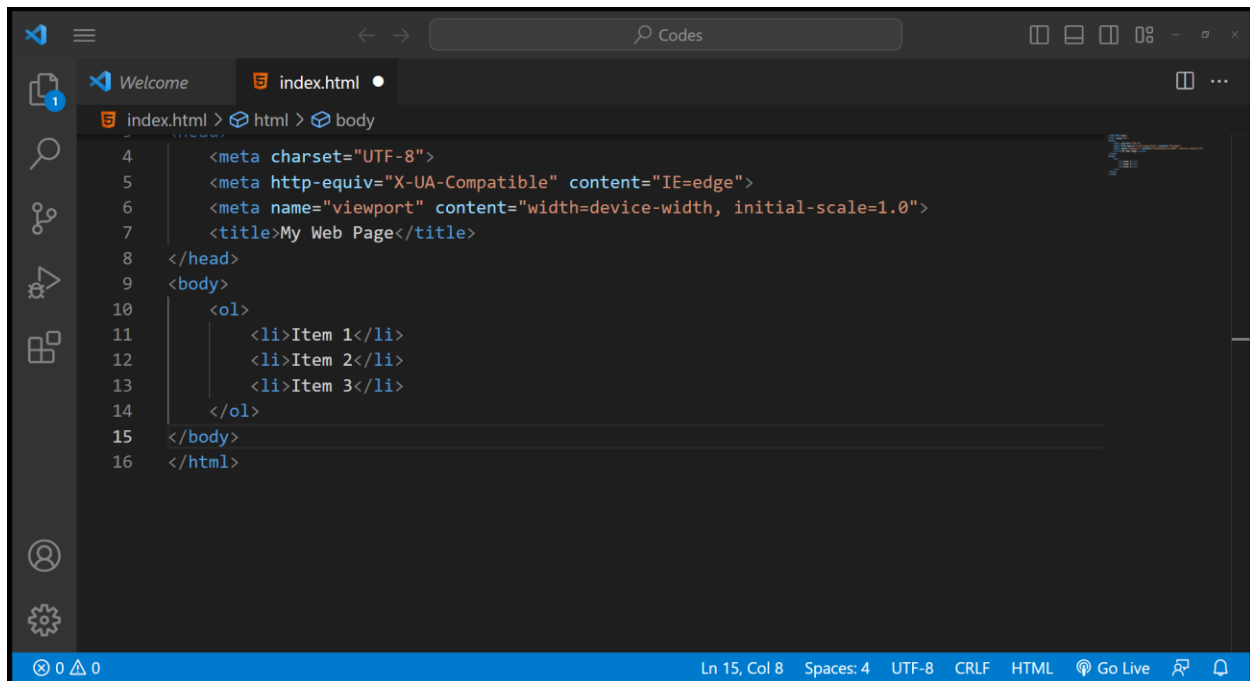
In this example, each level of indentation is four spaces. The `<head>` and `<body>` tags are indented one level deeper than the `<html>` tag, and the `<title>`, `<h1>`, `<p>`, and `<h2>` tags are indented one level deeper than the `<head>` and `<body>` tags, respectively.

You can also use an HTML formatter or code editor that supports auto-indentation to automatically format your code with proper indentation. This can save you time and ensure that your code is consistently formatted.

## HTML lists

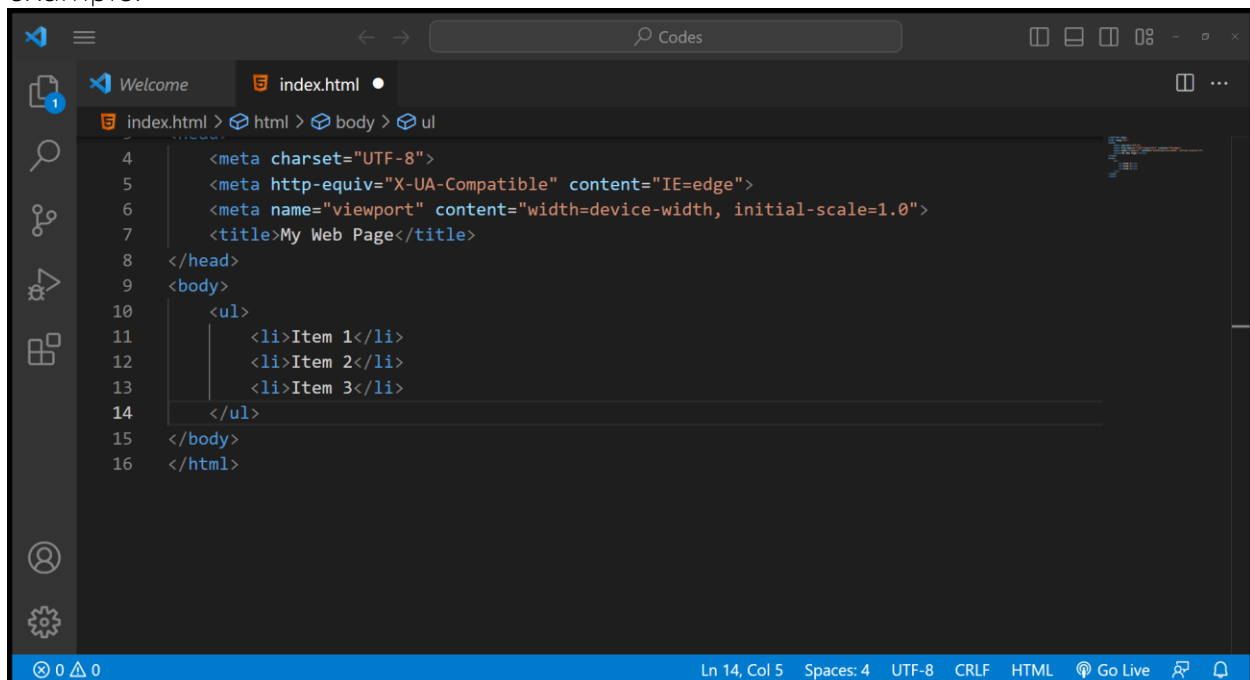
HTML provides two types of lists: ordered lists and unordered lists.

**Ordered lists:** Ordered lists are used to create numbered lists. To create an ordered list, use the `<ol>` tag and enclose each list item in an `<li>` tag. Here is an example:

A screenshot of the Visual Studio Code editor interface. The file explorer on the left shows 'index.html' selected. The editor window displays the HTML code for 'index.html'. The code includes a head section with meta tags for charset, http-equiv, and viewport, and a title 'My Web Page'. The body section contains an ordered list with three items: 'Item 1', 'Item 2', and 'Item 3'. The status bar at the bottom indicates 'Ln 15, Col 8', 'Spaces: 4', 'UTF-8', 'CRLF', 'HTML', and 'Go Live' button.

```
4 <meta charset="UTF-8">
5 <meta http-equiv="X-UA-Compatible" content="IE=edge">
6 <meta name="viewport" content="width=device-width, initial-scale=1.0">
7 <title>My Web Page</title>
8 </head>
9 <body>
10   <ol>
11     <li>Item 1</li>
12     <li>Item 2</li>
13     <li>Item 3</li>
14   </ol>
15 </body>
16 </html>
```

Unordered lists: Unordered lists are used to create bulleted lists. To create an unordered list, use the `<ul>` tag and enclose each list item in an `<li>` tag. Here is an example:

A screenshot of the Visual Studio Code editor interface, similar to the previous one. The file explorer shows 'index.html' selected. The editor window displays the HTML code for 'index.html'. The code is identical to the previous example, but the body section contains an unordered list instead of an ordered list. The status bar at the bottom indicates 'Ln 14, Col 5', 'Spaces: 4', 'UTF-8', 'CRLF', 'HTML', and 'Go Live' button.

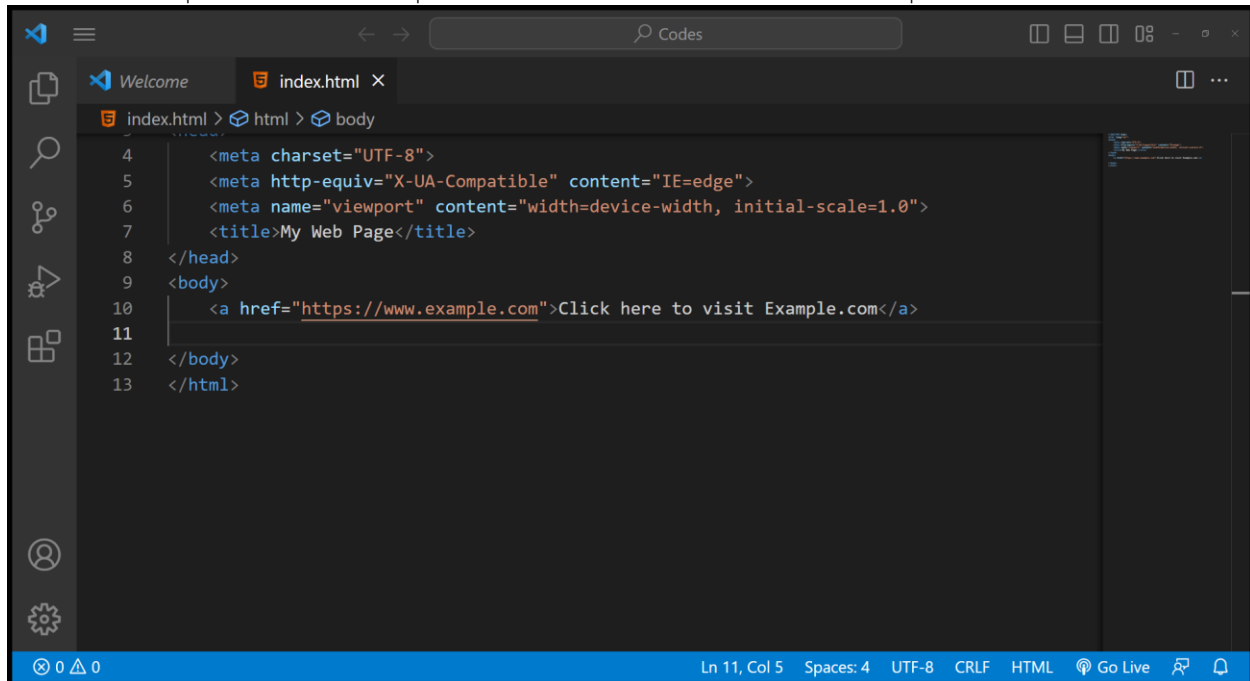
```
4 <meta charset="UTF-8">
5 <meta http-equiv="X-UA-Compatible" content="IE=edge">
6 <meta name="viewport" content="width=device-width, initial-scale=1.0">
7 <title>My Web Page</title>
8 </head>
9 <body>
10   <ul>
11     <li>Item 1</li>
12     <li>Item 2</li>
13     <li>Item 3</li>
14   </ul>
15 </body>
16 </html>
```

You can also customize the appearance of your lists using CSS. For example, you can change the color, size, and style of the bullets or numbers, or you can add background colors or images to the list items.

## HTML links

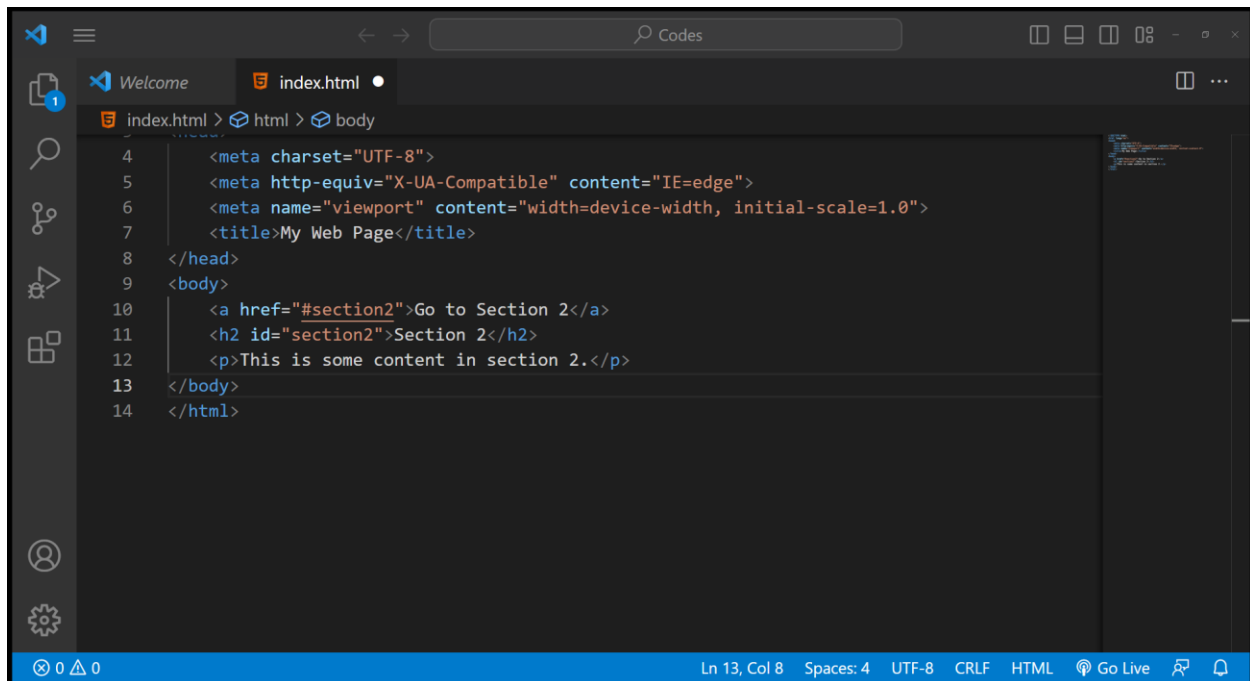
HTML links are used to create clickable hyperlinks that take the user to another web page or a specific location on the same page. Here is an example of how to create a basic link in HTML:

```
<a href="https://www.example.com">Click here to visit Example.com</a>
```



In this example, the `<a>` tag is used to create the link, and the `href` attribute specifies the URL that the link should point to. The text between the opening and closing `<a>` tags is the visible link text that the user will click on.

You can also create links to other locations within the same web page by using anchor tags and the `id` attribute to specify the target location. Here is an example:

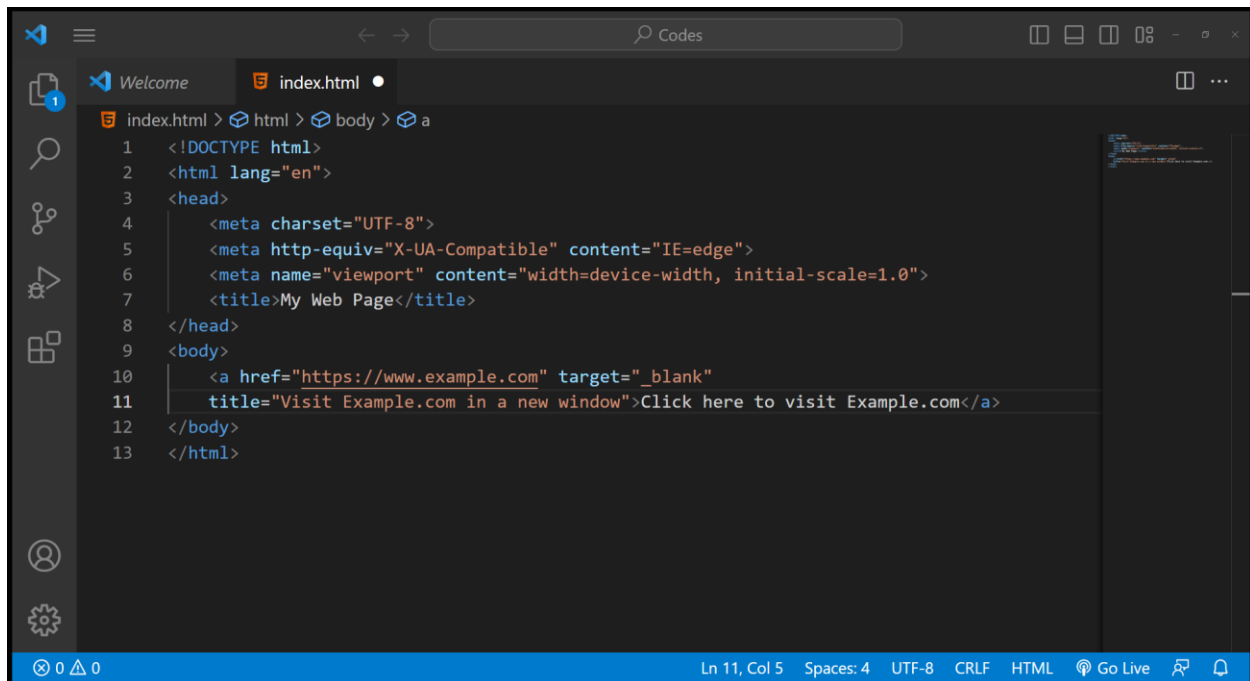


```
4 <meta charset="UTF-8">
5 <meta http-equiv="X-UA-Compatible" content="IE=edge">
6 <meta name="viewport" content="width=device-width, initial-scale=1.0">
7 <title>My Web Page</title>
8 </head>
9 <body>
10 <a href="#section2">Go to Section 2</a>
11 <h2 id="section2">Section 2</h2>
12 <p>This is some content in section 2.</p>
13 </body>
14 </html>
```

In this example, the link text will take the user to the location on the page with the id attribute of "section2". The target location is specified by including the # symbol followed by the id attribute value in the href attribute of the anchor tag.

Links can also have additional attributes, such as target, which specifies where to open the linked page (e.g., in a new window or tab), and title, which provides additional information about the link when the user hovers over it with the mouse. Here is an example:

```
<a href="https://www.example.com" target="_blank" title="Visit Example.com in a new window">Click here to visit Example.com</a>
```



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>My Web Page</title>
8 </head>
9 <body>
10   <a href="https://www.example.com" target="_blank"
11     title="Visit Example.com in a new window">Click here to visit Example.com</a>
12 </body>
13 </html>
```

In this example, the target attribute is set to `_blank`, which will cause the linked page to open in a new window or tab, and the title attribute provides additional information about the link.

## Navigating Pages

Navigating between pages is an important part of creating a website. In HTML, you can create links that allow the user to move from one page to another by using the `<a>` (anchor) tag.

To create a link to another page, you need to specify the URL of the destination page in the href attribute of the anchor tag. Here's an example:

```
<a href="page2.html">Go to Page 2</a>
```

In this example, the link text is "Go to Page 2", and the destination URL is "page2.html". When the user clicks on the link, the browser will navigate to the specified page.

If the destination page is in a different folder or directory, you need to include the path to the file in the href attribute. Here's an example:

```
<a href="articles/article1.html">Read Article 1</a>
```

In this example, the link text is "Read Article 1", and the destination URL is "articles/article1.html". This means that the "article1.html" file is located in a subdirectory called "articles".

You can also create links that open in a new window or tab by using the target attribute. Here's an example:

```
<a href="page2.html" target="_blank">Go to Page 2</a>
```

In this example, the target attribute is set to "\_blank", which will cause the linked page to open in a new window or tab.

When creating links between pages, it's important to make sure that the URLs are accurate and up-to-date, and that the linked pages are easy to navigate and understand.

## Images in HTML

In HTML, you can add images to your web pages using the `<img>` tag. Here's an example:

```

```

In this example, the `src` attribute specifies the URL of the image file, and the `alt` attribute provides a description of the image that can be used by screen readers or if the image cannot be displayed for some reason. When the browser encounters this code, it will display the image on the page.

You can also specify additional attributes for the `<img>` tag to control the display of the image, such as `width` and `height`, which set the dimensions of the image, and `title`, which provides additional information about the image when the user hovers over it with the mouse. Here's an example:

```

```

In this example, the `width` and `height` attributes set the dimensions of the image to 500 pixels wide by 300 pixels high, and the `title` attribute provides additional information about the image when the user hovers over it.

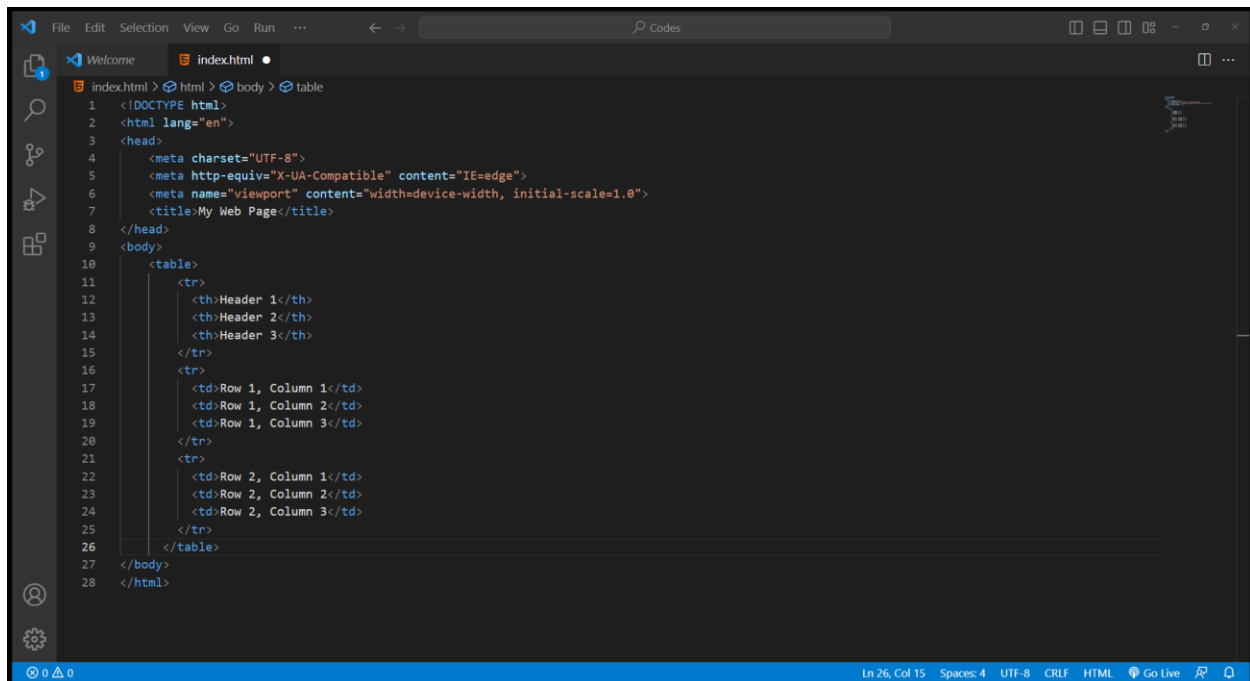
It's important to make sure that your image files are properly optimized for the web to ensure fast loading times and good performance. This includes using the appropriate file format (such as JPEG or PNG), compressing the file size as much as possible without sacrificing quality, and using descriptive file names and alt text for accessibility.

## Creating tables using HTML

In HTML, you can create tables to display data using the `<table>`, `<tr>`, `<th>`, and `<td>` tags.

Here's an example of a basic table:

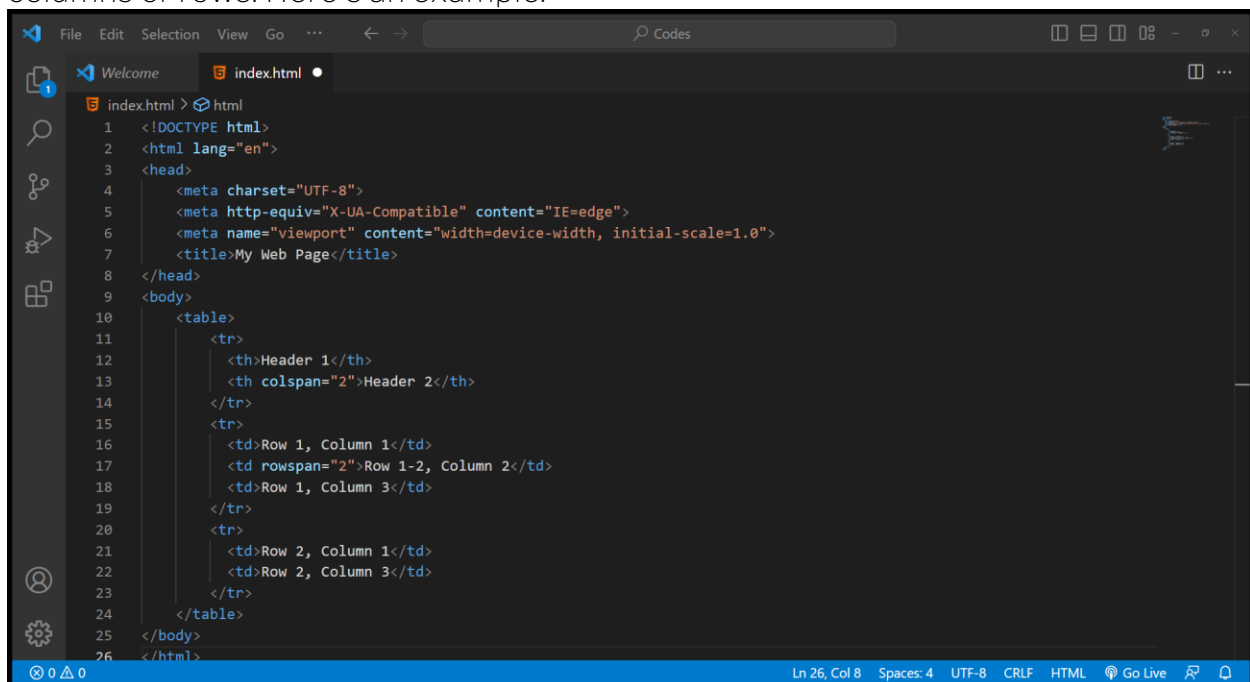




```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>My Web Page</title>
8 </head>
9 <body>
10   <table>
11     <tr>
12       <th>Header 1</th>
13       <th>Header 2</th>
14       <th>Header 3</th>
15     </tr>
16     <tr>
17       <td>Row 1, Column 1</td>
18       <td>Row 1, Column 2</td>
19       <td>Row 1, Column 3</td>
20     </tr>
21     <tr>
22       <td>Row 2, Column 1</td>
23       <td>Row 2, Column 2</td>
24       <td>Row 2, Column 3</td>
25     </tr>
26   </table>
27 </body>
28 </html>
```

In this example, the `<table>` tag creates the table, and each row is defined using the `<tr>` tag. The first row contains the headers of the table, which are defined using the `<th>` tag. The remaining rows contain the data for each cell, which are defined using the `<td>` tag.

You can also use the `colspan` and `rowspan` attributes to span cells across multiple columns or rows. Here's an example:



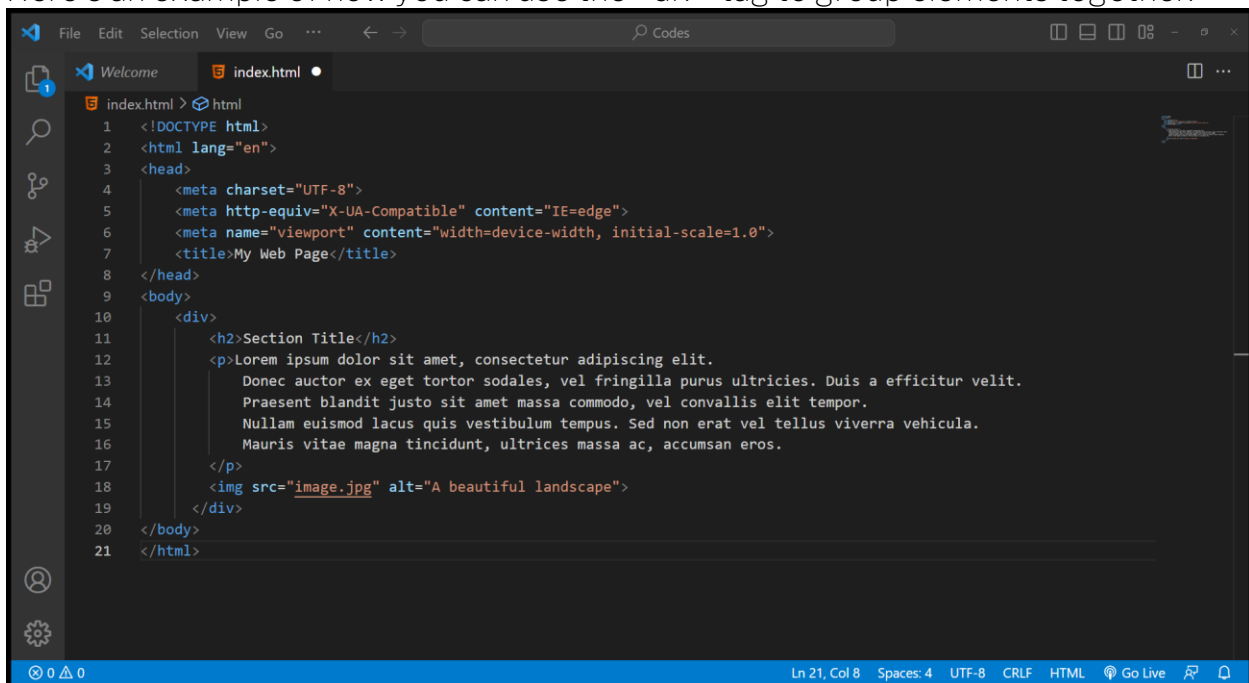
```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>My Web Page</title>
8 </head>
9 <body>
10   <table>
11     <tr>
12       <th>Header 1</th>
13       <th colspan="2">Header 2</th>
14     </tr>
15     <tr>
16       <td>Row 1, Column 1</td>
17       <td rowspan="2">Row 1-2, Column 2</td>
18       <td>Row 1, Column 3</td>
19     </tr>
20     <tr>
21       <td>Row 2, Column 1</td>
22       <td>Row 2, Column 3</td>
23     </tr>
24   </table>
25 </body>
26 </html>
```

In this example, the second header spans two columns using the `colspan` attribute. The second cell in the first-row spans two rows using the `rowspan` attribute. It's important to properly structure your tables using the appropriate tags to ensure that they are accessible and easy to read. This includes using `<thead>`, `<tbody>`, and `<tfoot>` tags to group table headers, data, and footers, and using CSS to style your tables for a consistent look and feel.

## Introducing the Div

In HTML, the `<div>` tag is used to create a division or a container for a section of a web page. The `<div>` tag itself doesn't have any specific visual or semantic meaning, but it allows you to group related elements together and apply styles or JavaScript effects to them as a unit.

Here's an example of how you can use the `<div>` tag to group elements together:



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>My Web Page</title>
8 </head>
9 <body>
10   <div>
11     <h2>Section Title</h2>
12     <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.
13       Donec auctor ex eget tortor sodales, vel fringilla purus ultricies. Duis a efficitur velit.
14       Praesent blandit justo sit amet massa commodo, vel convallis elit tempor.
15       Nullam euismod lacus quis vestibulum tempus. Sed non erat vel tellus viverra vehicula.
16       Mauris vitae magna tincidunt, ultrices massa ac, accumsan eros.
17     </p>
18     
19   </div>
20 </body>
21 </html>
```

In this example, the `<div>` tag contains a heading, a paragraph, and an image that are related to each other. You can apply CSS styles to the `<div>` tag to change the background color, border, padding, or margin of the entire section.

The `<div>` tag can also be used to create layout structures for your web page by dividing it into sections. You can use multiple `<div>` tags to create a grid-like layout or to create columns for your content.

It's important to use the `<div>` tag in a way that makes sense for the content and purpose of your web page. Overusing the `<div>` tag can make your code harder to read and maintain, and can also affect the accessibility and usability of your web page.

## Semantic sectioning

Semantic sectioning refers to the practice of using HTML tags to create meaningful sections in your web page content. By using semantic sectioning, you can provide structure and context to your content, which can improve the accessibility, usability, and search engine optimization (SEO) of your web page.

Here are some examples of semantic sectioning tags in HTML:

`<header>` - used to define the header or top section of a web page or a section within a web page.

`<nav>` - used to define a navigation section of a web page.

`<main>` - used to define the main content section of a web page.

`<section>` - used to define a section of content within a web page.

`<article>` - used to define a self-contained article or blog post within a web page.

`<aside>` - used to define a section of content that is related but not central to the main content of a web page.

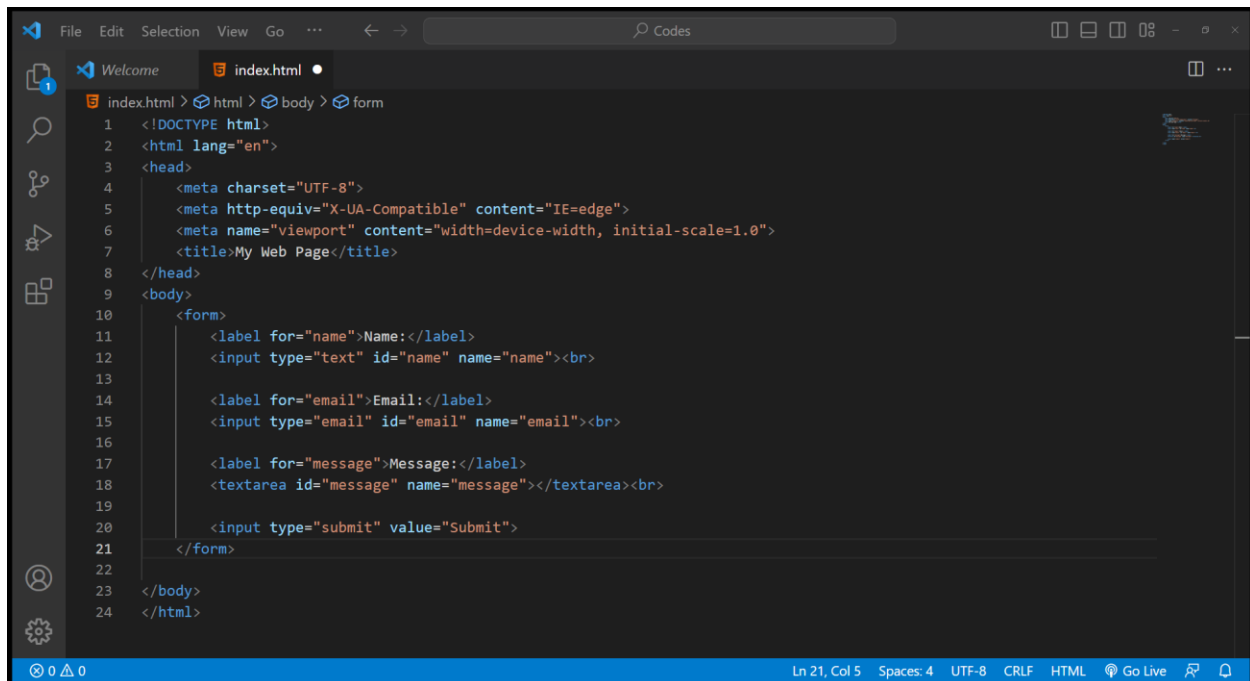
`<footer>` - used to define the footer or bottom section of a web page or a section within a web page.

By using these semantic sectioning tags, you can provide clear and meaningful structure to your content, which can benefit both users and search engines. For example, screen readers can use the semantic structure of your content to provide a better experience for visually impaired users, and search engines can use the semantic structure to better understand and index your content for search results.

## HTML forms

HTML forms are used to collect information from users on a web page. Forms can contain a variety of input fields, such as text boxes, checkboxes, radio buttons, and drop-down lists, which allow users to input different types of data.

Here's an example of a simple form in HTML:

A screenshot of a code editor window with a dark theme. The editor shows an HTML file named 'index.html'. The code defines a form with three input fields: a text input for 'Name', an email input for 'Email', and a text area for 'Message'. A 'Submit' button is at the bottom of the form. The code is as follows:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>My Web Page</title>
8 </head>
9 <body>
10  <form>
11    <label for="name">Name:</label>
12    <input type="text" id="name" name="name"><br>
13
14    <label for="email">Email:</label>
15    <input type="email" id="email" name="email"><br>
16
17    <label for="message">Message:</label>
18    <textarea id="message" name="message"></textarea><br>
19
20    <input type="submit" value="Submit">
21  </form>
22
23 </body>
24 </html>
```

The editor interface includes a sidebar on the left with icons for Explorer, Search, and Run and Debug. The top bar shows 'File Edit Selection View Go' and a search bar. The bottom status bar indicates 'Ln 21, Col 5 Spaces: 4 UTF-8 CRLF HTML Go Live'.

In this example, the `<form>` tag defines the start and end of the form, and the various input fields are defined within the form using different HTML tags. Each input field is given a unique name and ID, which can be used to identify and access the data entered by the user.

The `<label>` tag is used to associate a text label with an input field, making it more accessible to users who are using assistive technology. The `<input>` tag is used to define the type of input field, such as text, email, or submit button, and the `<textarea>` tag is used to define a multi-line input field for longer messages.

When the user submits the form, the data is sent to a server-side script, which can then process the data and take appropriate action, such as sending an email or storing the data in a database.

HTML forms are a powerful tool for collecting data from users, but they also require careful design and implementation to ensure that they are easy to use and secure. Proper validation of user input, protection against spam and malicious attacks, and clear feedback to users are all important considerations when designing and implementing HTML forms.

## The various input element types in HTML

HTML provides a variety of input element types that can be used in forms to allow users to input different types of data. Here are some of the most common input types:

`<input type="text">` - used for single-line text input.

`<input type="password">` - used for password input, with the entered characters hidden.

`<input type="email">` - used for email address input, with basic validation.

`<input type="number">` - used for numeric input, with optional minimum and maximum values.

`<input type="date">` - used for date input, with a date picker widget in some browsers.

`<input type="checkbox">` - used for checkboxes, allowing users to select one or more options.

`<input type="radio">` - used for radio buttons, allowing users to select one option from a group.

`<input type="file">` - used for file uploads, allowing users to select and upload files.

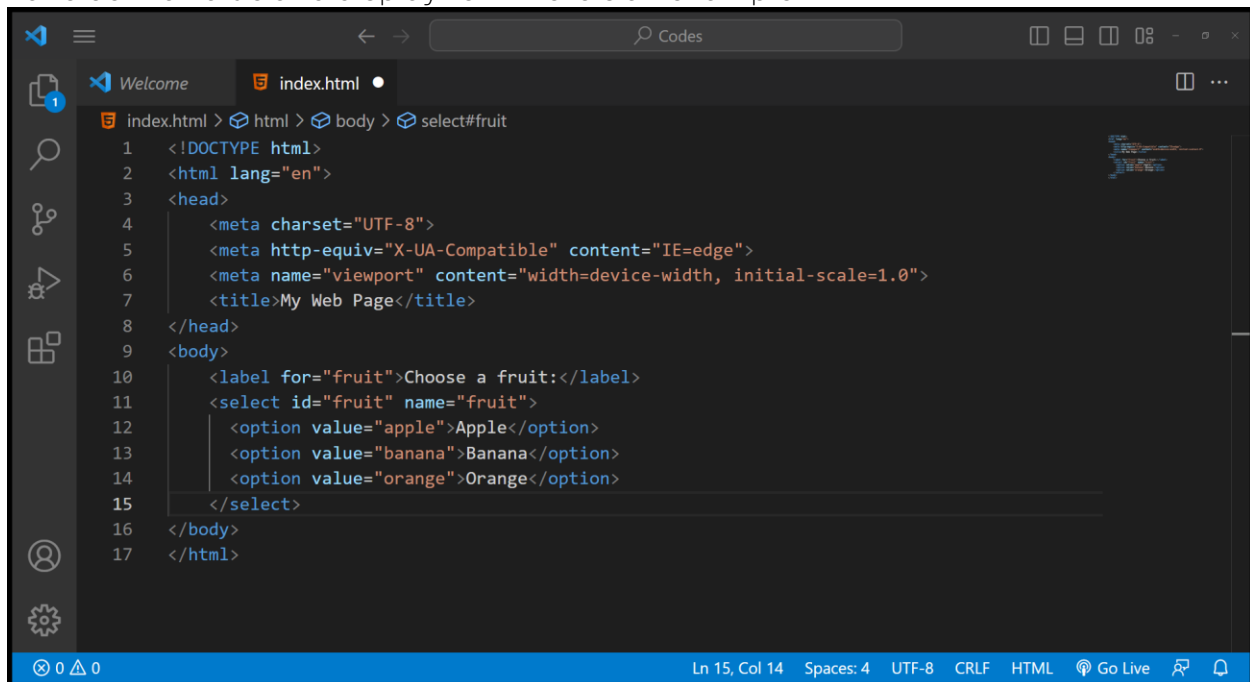
`<input type="submit">` - used for submitting the form to the server.

`<input type="reset">` - used for resetting the form to its initial state.

These input types can be combined with other attributes, such as name, id, value, placeholder, and required, to create more complex and customized input fields. For example, the name attribute can be used to identify the input field when the form is submitted, and the required attribute can be used to require that the user enters a value before the form can be submitted. By choosing the appropriate input types and attributes, you can create forms that are easy to use, secure, and effective at collecting the data you need from your users.

## The select dropdown in HTML

The `<select>` element in HTML is used to create a drop-down list of options that users can select from. Each option is defined using the `<option>` element, which can have both a value and display text. Here's an example:



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>My Web Page</title>
8 </head>
9 <body>
10  <label for="fruit">Choose a fruit:</label>
11  <select id="fruit" name="fruit">
12    <option value="apple">Apple</option>
13    <option value="banana">Banana</option>
14    <option value="orange">Orange</option>
15  </select>
16 </body>
17 </html>
```

In this example, the `<select>` element defines the start and end of the drop-down list, and the `<option>` elements define the different options that the user can choose from. The value attribute of each option specifies the value that will be sent to the server when the form is submitted, while the text between the opening and closing `<option>` tags is the text that will be displayed to the user.

The id and name attributes of the `<select>` element is used to identify the field and its value when the form is submitted. The for attribute of the `<label>` element is used to associate the label with the input field, making it more accessible to users. When the user selects an option from the drop-down list and submits the form, the value of the selected option will be sent to the server as part of the form data. The server-side script can then process the data and take appropriate action based on the selected option.

The `<select>` element can also be used in combination with JavaScript to create more dynamic and interactive drop-down lists, such as ones that change based on user input or that pull data from an external source. Overall, the `<select>` element is a powerful and flexible tool for creating user-friendly forms and interfaces in HTML.

## The Inspect tool

The Inspect tool is a feature in web browsers that allows you to view the HTML, CSS, and JavaScript code of a web page and make real-time changes to the code and see how they affect the page. It is a powerful tool for web developers and designers to debug, test, and optimize their web pages.

To open the Inspect tool in most web browsers, you can right-click on any element on a web page and select "Inspect" or "Inspect Element" from the context menu.

This will open the Inspect tool in a separate window or pane, showing the HTML code of the selected element and its corresponding CSS styles.

From here, you can edit the code and styles in real time, either by clicking and typing directly in the Inspect tool, or by editing the code in your favorite code editor and refreshing the web page. This allows you to experiment with different design and layout options, test how your web page responds to different screen sizes and devices, and identify and fix any bugs or errors in your code.

The Inspect tool also provides a wealth of other features and functions, such as:

- Viewing and modifying the properties and values of CSS styles.
- Analyzing network requests and response times.
- Debugging JavaScript code and setting breakpoints.
- Simulating different user agents and devices.
- Testing accessibility and performance metrics.

Overall, the Inspect tool is an essential tool for anyone working with web development and design, and can greatly speed up the process of building, testing, and refining your web pages.

## Using the data attribute in HTML

The data attribute in HTML is a powerful and flexible way to store custom data in HTML elements, which can then be accessed and manipulated by JavaScript or CSS. The data attribute can be added to any HTML element, and can be used to store any kind of data, such as strings, numbers, objects, arrays, or even JSON data.

The syntax for using the data attribute is simple: you simply add the attribute to the HTML element, with a descriptive name for the data value, like this:

```
<div data-mydata="42">This is a div with data attribute</div>
```

```
<div data-mydata="42">This is a div with data attribute</div>
```

In this example, we have added a data attribute to a `<div>` element, with a name of "mydata" and a value of "42". To access this data using JavaScript, you can use the `getAttribute()` method, like this:

```
var mydiv = document.querySelector('div[data-mydata="42"]');
```

```
var mydata = mydiv.getAttribute('data-mydata');
```

```
console.log(mydata); // Output: 42
```

In this example, we use the `querySelector()` method to select the `<div>` element with the `data-mydata` attribute set to "42", and then use the `getAttribute()` method to retrieve the value of the attribute.

Using the data attribute can be particularly useful for storing and manipulating data that is specific to a particular HTML element, such as data for a tooltip, a custom attribute for tracking user interactions, or data for a custom widget or plugin. By using the data attribute, you can keep your HTML code clean and semantic, while still providing a powerful and flexible way to store and manipulate data in your web applications.

## Commenting Out Code

Commenting out HTML code is a way to temporarily disable or hide a section of code from being displayed or executed in a web browser. This can be useful for testing or debugging purposes, or for leaving notes and reminders in your code for future reference.

In HTML, you can comment out a section of code by using the `<!-- -->` syntax. Any text or code inside these tags will be treated as a comment and ignored by the browser:

```
<!-- This is a comment in HTML -->
```

```
<p>This is some visible text.</p>
```

```
<!-- <p>This is some commented-out text.</p> -->
```

In this example, we have added a comment to the first line using the `<!-- -->` syntax. The text "This is a comment in HTML" will not be displayed in the web browser, but can be seen in the HTML source code.

We have also commented out a `<p>` element on the third line by surrounding it with the `<!-- -->` tags. This means that the text "This is some commented-out text" will not be displayed in the web browser, but can be seen in the HTML source code. This can be useful for temporarily hiding code that you want to keep for future reference, or for testing and debugging purposes.

When you are ready to enable the commented-out code, you can simply remove the `<!-- -->` tags to restore the original code. By using comments in your HTML code, you can make your code more readable, maintainable, and easy to debug and test.

---

Instructor:

**ABUBAKAR SIDDIQUE**

Full-Stack Developer

MS in Computer Science (MDS, WSN, MCS, BED), LCC (AMAP, Online), BTI (CIT, Australia), LDT (open SAP, Online), CLC (Peking University, China), CAC (TEVTA, PAK), ICSC (The Open University, UK), Data Science (SAP, Online), RPJ (VU, PAK)

[dr.sagher@gmail.com](mailto:dr.sagher@gmail.com), +923017362696