|  |  |
| --- | --- |
| **Week 1 - HTML Basics:** | |
| **Day 1-2** | Introduction to HTML, Basic Tags (head, title, body, h1-h6, p, br, hr) |
| **Day 3** | Lists (ordered, unordered), Links (anchor tag, absolute and relative URLs) |
| **Day 4** | Images (img tag, height, width, alt attributes) |
| **Day 5** | Tables (table, tr, td, th), Forms (form, input, label, button) |
| **Day 6-7** | Review and Practice |

|  |  |
| --- | --- |
| **Week 2 - CSS Basics:** | |
| **Day 8-9** | Introduction to CSS, Inline and Internal CSS, Selectors, Properties (color, background-color, font-size, font-family) |
| **Day 10** | External CSS, The class and id Selectors |
| **Day 11** | Box Model (margin, border, padding), Display Property (block, inline, inline-block) |
| **Day 12** | Positioning (static, relative, absolute, fixed), Float and Clear |
| **Day 13** | Flexbox and Grid |
| **Day 14-15** | Review and Practice |

HTML documents are defined by HTML tags. Each HTML tag describes different document content.

1. <!DOCTYPE html>: This declaration defines the document to be HTML5.
2. <html>: The HTML document itself begins with <html> and ends with </html>.
3. <head>: The head element contains meta-information about the HTML document, such as its title.
4. <title>: The title element specifies a title for the HTML document. It is shown in the browser’s title bar or tab.
5. <body>: The body element contains the content of the document, such as text, images, hyperlinks, tables, lists, etc.

Code: -

<!DOCTYPE *html*>

<html>

<head>

    <title>Page Title</title>

</head>

<body>

    <h1>This is a Heading</h1>

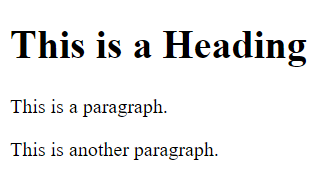
    <p>This is a paragraph.</p>

    <p>This is another paragraph.</p>

</body>

</html>

Output: -



1. Headings: Defined with the <h1> to <h6> tags. <h1> defines the most important heading, and <h6> defines the least important heading.
2. Paragraphs: Defined with the <p> tag.
3. Line Breaks: Defined with the <br> tag. This is an empty tag which means that it has no end tag.
4. Horizontal Rules: Defined with the <hr> tag. This is used to specify a thematic break in an HTML page to divide or separate document sections.

Code: -

<!DOCTYPE *html*>

<html>

<head>

    <title>Page Title</title>

</head>

<body>

    <h1>This is a Heading</h1>

    <p>This is a paragraph.</p>

    <p>This is another paragraph.</p>

    <br>

    <hr>

    <h2>This is a smaller Heading</h2>

    <p>This is a paragraph after a horizontal rule.</p>

</body>

</html>

Output: -



1. **HTML Lists:** HTML lists are used to group together related items in a list. There are two types of lists in HTML:
   * **Ordered lists (**<ol>**):** An ordered list starts with the <ol> tag. Each list item starts with the <li> tag. The list items will be marked with numbers by default.

|  |  |
| --- | --- |
| Code: | Output: |
| <ol>      <li>Coffee</li>      <li>Tea</li>      <li>Milk</li>  </ol> |  |

* + **Unordered lists (**<ul>**):** An unordered list starts with the <ul> tag. Each list item starts with the <li> tag. The list items will be marked with bullets by default.

|  |  |
| --- | --- |
| Code: | Output: |
| <ul>      <li>Coffee</li>      <li>Tea</li>      <li>Milk</li>  </ul> |  |

* + **Description lists (**<dl>**):** A description list list in HTML is a list of terms with their associated descriptions. It’s also known as a definition list

|  |  |
| --- | --- |
| Code: | Output: |
| <dl>      <dt>Term 1</dt>      <dd>Description for Term 1</dd>      <dt>Term 2</dt>      <dd>Description for Term 2</dd>  </dl> |  |

1. **HTML Links (**<a>**):** HTML links are defined with the <a> tag. The link’s destination is specified in the href attribute. Links are used to navigate between pages and within the same page.
   * **Absolute URLs:** An absolute URL contains the full URL including the protocol (http, https), domain name, and path. It points to another web site (like href="http://www.example.com/default.html").
   * **Relative URLs:** A relative URL points to a file within the same website. This is often simply a specified path (like href="default.html").

Here’s an example of how to use both types of URLs:

Code:

*<!-- Absolute URL -->*

<a *href*="https://www.google.com">Visit Google.com</a>

<br>

*<!-- Relative URL -->*

<a *href*="ordered list.html">ordered list</a>

Output:

****

The <img> tag is used to embed an image in an HTML page. Images are not technically inserted into an HTML page; images are linked to HTML pages. The <img> tag creates a holding space for the referenced image.

Here’s the basic syntax:

**HTML**

<img *src*="url" *alt*="alternatetext">

* src: This attribute stands for “source” which is used to specify the URL of the image. The URL can be either an absolute URL (like src="http://www.example.com/image.jpg") or a relative URL (like src="/path/to/image.jpg").
* alt: This attribute stands for “alternate”. It is used to specify alternate text to be displayed if the image cannot be loaded for any reason. This could be because the URL is incorrect, the image doesn’t exist, or the user is using a screen reader (which is a tool used by people with visual impairments to read the content of the screen).

Here’s an example:

|  |  |
| --- | --- |
| Code: | Output: |
| <img *src*="smiley.jpg" *alt*="Smiley face"> |  |

In addition to src and alt, there are two more attributes that are commonly used with the <img> tag:

* width: This attribute is used to specify the width of the image. The value is specified in pixels by default, but it can also be in percent (like width="50%").
* height: This attribute is used to specify the height of the image. The value is specified in pixels by default, but it can also be in percent (like height="50%").

Here’s an example that uses these attributes:

|  |  |
| --- | --- |
| Code: | Output: |
| <img *src*="smiley.jpg" *alt*="Smiley face" *width*="500" *height*="600"> |  |

1. **HTML Tables:** HTML tables are used to display data in a tabular format (like a spreadsheet). Here’s the basic syntax:

|  |  |
| --- | --- |
| Code: | Output: |
| <table>          <tr>              <th>Header 1</th>              <th>Header 2</th>          </tr>          <tr>              <td>Data 1</td>              <td>Data 2</td>          </tr>  </table> |  |

* + **<table>:** This tag is used to create a table.
  + **<tr>:** This tag stands for “table row”. It is used to define a row in the table.
  + **<td>:** This tag stands for “table data”. It is used to define a cell in the table.
  + **<th>:** This tag stands for “table header”. It is used to define a header cell in the table.

1. **HTML Forms:** HTML forms are used to collect user input. Here’s the basic syntax:

|  |  |
| --- | --- |
| Code: | Output: |
| <form *action*="submit.html" *method*="get">          <label *for*="name">Name:</label><br>          <input *type*="text" *id*="name" *name*="name"><br>          <input *type*="submit" *value*="Submit">  </form> |  |

* + **<form>**: This tag is used to create a form. The action attribute specifies where to send the form data when the form is submitted. The method attribute specifies how to send the form data.
  + **<label>**: This tag defines a label for many form elements. The for attribute of the <label> tag should be equal to the id attribute of the related element to bind them together.
  + **<input>**: This tag is used to create an input control. The type attribute specifies the type of input to display such as “text”, “email”, “password”, etc. The id attribute is a unique id for the input field. The name attribute specifies the name for the input element. The name attribute is used to reference form data after the form is submitted.
  + **<button>**: This tag is used to create a clickable button.

1

2

3

4

5

6

7

8

9

10

.

**CSS (Cascading Style Sheets)** is a language used to style HTML elements. It can control the layout of multiple web pages all at once. There are three types of CSS: Inline, Internal (or Embedded), and External.

**1. Inline CSS:** This method involves adding CSS directly to your HTML elements using the style attribute. Here’s an example:

|  |  |
| --- | --- |
| Code: | Output: |
| <h1 *style*="color: green; font-size: 24px;">Hello, world!</h1> |  |

**2. Internal CSS:** This method involves adding styles to the head section of your HTML document using the style tag. This is useful when a single HTML document needs to be styled uniquely. Here’s an example:

Code:

<!DOCTYPE *html*>

<html>

<head>

    <style>

        body {

            background-color: lightblue;

        }

        h1 {

            color: white;

            text-align: center;

        }

        p {

            font-family: verdana;

            font-size: 20px;

        }

    </style>

</head>

<body>

    <h1>Welcome to My Homepage</h1>

    <p>This is a paragraph.</p>

</body>

</html>

Output:



**Selectors:** CSS selectors are used to select the HTML element(s) you want to style. For example, in the above code, body, h1, and p are selectors.

**Properties:** CSS properties are used to set the style of the selected HTML elements. Some basic properties include:

* color: Sets the text color. For example, color: red;.
* background-color: Sets the background color of an element. For example, background-color: blue;.
* font-size: Sets the font size of the text. For example, font-size: 20px;.
* font-family: Specifies the font of the text. For example, font-family: Arial;.
* **External CSS** is a method where you link an external .css file to your HTML document. This method is beneficial when you want to apply the same style to multiple HTML pages.

**CSS Code:**

body {

    background-color: lightblue;

}

h1 {

    color: white;

    text-align: center;

}

p {

    font-family: verdana;

    font-size: 20px;

}

**HTML Code:**

<!DOCTYPE *html*>

<html>

<head>

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

</head>

<body>

    <h1>Welcome to My Homepage</h1>

    <p>This is a paragraph.</p>

</body>

</html>

**Output:**



**Class Selectors:** The class selector selects HTML elements with a specific class attribute. It starts with a period (.) character. For example:

**HTML**

<p *class*="center">This is a paragraph.</p>

**CSS**

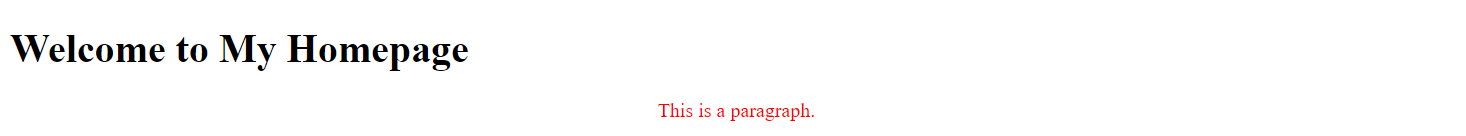
.center {

    text-align: center;

    color: red;

}

**Output:**



We can also specify that only specific HTML elements should be affected by a class. For example, only **<p>** elements with class **“center”** will be red and center-aligned:

**HTML**

<h1 *class*="center">Welcome to My Homepage</h1>

<p *class*="center">This is a paragraph.</p>

**CSS**

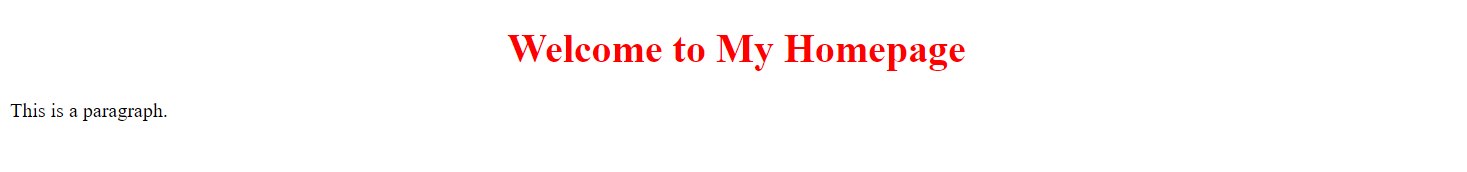
h1.center {

    text-align: center;

    color: red;

}

**Output:**

****

HTML elements can also refer to more than one class. For example:

**HTML**

<p *class*="center large">This paragraph refers to two classes.</p>

In this example, the <p> element will be styled according to both the “center” and “large” classes.

Remember, a class name cannot start with a number. The class attribute can specify one or more class names for an HTML element, and its elements can be used in any tag in HTML.

**1. CSS Box Model:** The CSS box model is essentially a box that wraps around every HTML element. It consists of: content, padding, borders, and margins.

* **Content:** This is where text and images appear.
* **Padding:** This clears an area around the content. The padding is transparent.
* **Border:** This goes around the padding and content.
* **Margin:** This clears an area outside the border. The margin is transparent.

Here’s an example:

**HTML**

<!DOCTYPE *html*>

<html>

<head>

    <link *rel*="stylesheet" *type*="text/css" *href*="BoxModel.css">

    <title>Box Model</title>

</head>

<body>

    <div>

        This is a div element with a width of 300px, a green border of 15px, padding of 50px, and a margin of 20px.

    </div>

</body>

</html>

**CSS**

div {

    width: 300px;

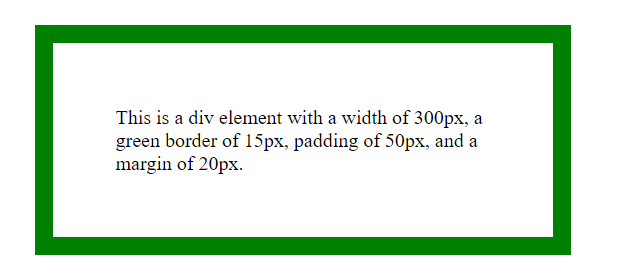
    border: 15px solid green;

    padding: 50px;

    margin: 20px;

}

Output:



**2. Display Property:** The display property specifies the display behavior of an element. It can be block, inline, or inline-block.

* **Block:** Elements like <div>, <p>, and <h1> are block elements. They start on a new line and take up the full width available.
* **Inline:** Elements like <span>, <a>, and <img> are inline elements. They do not start on a new line and only take up as much width as necessary.
* **Inline-Block:** This is a mix of both. Inline-block elements are like inline elements but they can have a width and height. This means you can use the element inline but still control its dimensions.

Here’s an example:

**HTML**

<!doctype *html*>

<html>

    <head>

        <title>Display Property</title>

        <link *rel*="stylesheet" *type*="text/css" *href*="DisplayProperty.css">

    </head>

    <body>

        <div *class*="block">

            This is a block-level div element.

        </div>

        <span *class*="inline">

            This is an inline span element.

        </span>

        <span *class*="inline-block">

            This is an inline-block span element.

        </span>

    </body>

</html>

**CSS**

div.block {

    display: block;

    width: 60%;

    height: 200px;

    border: 2px solid blue;

}

span.inline {

    display: inline;

    border: 2px solid red;

}

span.inline-block {

    display: inline-block;

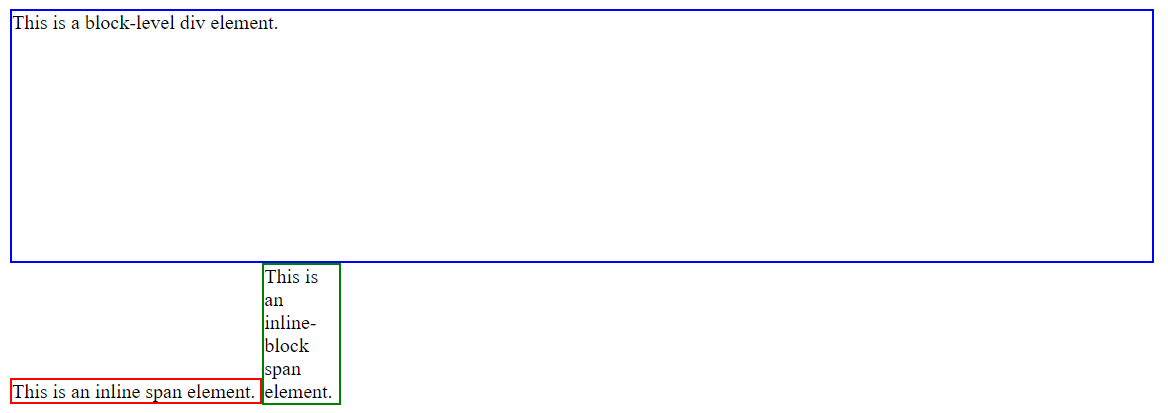
    width: 60px;

    height: auto;

    border: 2px solid green;

}

**Output:**



In this example, the div with class “block” will behave as a block element, the span with class “inline” will behave as an inline element, and the span with class “inline-block” will behave as an inline-block element.

1. **Positioning**: The position property in CSS determines how an element is positioned in a document. There are five different position values:
   * **Static**: This is the default value. The element is positioned according to the normal flow of the document. The top, right, bottom, left, and z-index properties have no effect. Here is an example:

**CSS**

div.static {

position: static;

border: 3px solid #73AD21;

}

* + [**Relative**: The element is positioned according to the normal flow of the document, and then offset relative to itself based on the values of top, right, bottom, and left](https://developer.mozilla.org/docs/Web/CSS/position)[1](https://developer.mozilla.org/docs/Web/CSS/position). Here is an example:

**CSS**

div.relative {

position: relative;

left: 30px;

border: 3px solid #73AD21;

}

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

* + **Absolute**: The element is removed from the normal document flow, and no space is created for the element in the page layout. [It is positioned relative to its closest positioned ancestor](https://developer.mozilla.org/docs/Web/CSS/position)[1](https://developer.mozilla.org/docs/Web/CSS/position). Here is an example:

**CSS**

div.relative {

position: relative;

width: 400px;

height: 200px;

border: 3px solid #73AD21;

}

div.absolute {

position: absolute;

top: 80px;

right: 0;

width: 200px;

height: 100px;

border: 3px solid #73AD21;

}

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

* + **Fixed**: The element is removed from the normal document flow, and no space is created for the element in the page layout. [It is positioned relative to the viewport](https://developer.mozilla.org/docs/Web/CSS/position)[1](https://developer.mozilla.org/docs/Web/CSS/position). Here is an example:

**CSS**

div.fixed {

position: fixed;

bottom: 0;

right: 0;

width: 300px;

border: 3px solid #73AD21;

}

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

* + [**Sticky**: The element is positioned according to the normal flow of the document, and then offset relative to its nearest scrolling ancestor and containing block](https://developer.mozilla.org/docs/Web/CSS/position)[1](https://developer.mozilla.org/docs/Web/CSS/position).

1. **Float**: The float property is used for positioning and formatting content e.g. let an image float left to the text in a container.
2. **Clear**: The clear property is used to control the behavior of floating elements. Elements after a floating element will flow around it. To avoid this, use the clear property. The clear property can have one of the following values:
   * none: Allows floating elements on both sides. This is default
   * left: No floating elements allowed on the left side
   * right: No floating elements allowed on the right side
   * both: No floating elements allowed on either the left or the right side
   * initial: Sets this property to its default value
   * inherit: Inherits this property from its parent element

.