***Module 1: Learn and Explore HTML as a Beginner***

HTML (**Hypertext Markup Language**) is the code that is used to structure a web page and its content. It is the basic scripting language used by web browsers to render pages on the World Wide Web.

**Hypertext** is text which contains links to other texts. HyperText allows a user to click a link and be redirected to a new page referenced by that link.

**Markup** refers to how you can structure or format content. **HTML** is called a ***Markup language*** because it allows users to organize the look and presentation of all the content that needs to go on a web page. It tells the web browser how to display the words and images on the page to the user. It also determines the appearance and structure of the text, i.e. font type, size and color, and even helps distinguish(পার্থক্য করা) HTML elements from text, which is done using HTML tags.

**Semantic Elements:** A semantic element clearly describes its meaning to both the browser and the developer. In programming, **Semantics** refers to the meaning of a piece of code. মানে জিনিসটা অর্থবহ।

For example, a <p> tag indicates that the enclosed (ঘেরা) text is a paragraph. This is both semantic and presentational because people know what paragraphs are, and browsers know how to display them. <form>, <table>, <bold> and <article> - Clearly defines its content.

On the flip side of this equation, tags such as <b> and <i> are not semantic. They define only how the text should look (bold or italic), and don't provide any additional meaning to the markup.

The benefit of writing semantic HTML stems from (থেকে উদ্ভূত) what should be the driving goal of any web page: the desire (আকাঙ্ক্ষিত বস্তু) to communicate. By adding semantic tags to your document, you provide additional information about that document, which aids (সাহায্য করা) in communication. Specifically, semantic tags make it clear to the browser what the meaning of a page and its content is. That clarity is also communicated with search engines, ensuring that the right pages are delivered for the right queries.

Semantic HTML tags provide information about the contents of those tags that goes beyond just how they look on a page. Text that is enclosed in the <code> tag is immediately recognized by the browser as some type of coding language. Instead of trying to render that code, the browser understands that you are using that text as an example of the code for the purposes of an article or online tutorial.

Using semantic tags gives you many more hooks for styling your content, too. Perhaps (সম্ভবত) today you prefer to have your code samples display in the default browser style, but tomorrow, you might want to call them out with a gray background color; later still, you might want to define the precise mono-spaced font family or font stack to use for your samples. You can do all of these things easily by using semantic markup and smartly applied CSS.

The <p> tag defines a **paragraph**.

<p> This is some text in a paragraph. </p>

Output: 

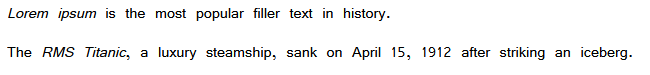
The <b> tag specifies **bold** text without any extra importance.

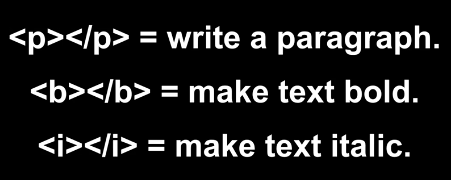
<p>This is normal text - <b>and this is bold text</b>.</p>

Output: 

The <i> tag defines a part of text in an alternate voice or mood. The content inside is typically displayed in **italic**. The <i> tag is often used to indicate a technical term, a phrase from another language, a thought, a ship name, etc.

<p><i>Lorem ipsum</i> is the most popular filler text in history.</p>  
  
<p>The <i>RMS Titanic</i>, a luxury steamship, sank on April 15, 1912 after striking an iceberg.</p>

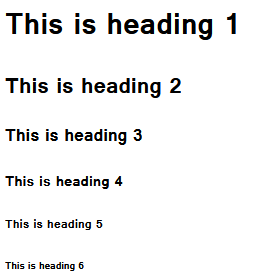
Output: 



The <h1> to <h6> tags are used to define HTML **headings**. <h1> defines the most important heading. <h6> defines the least important heading.

**Note:** Only use one <h1> per page - this should represent the main heading/subject for the whole page. Also, do not skip heading levels - start with <h1>, then use <h2>, and so on.

<h1>This is heading 1</h1>  
<h2>This is heading 2</h2>  
<h3>This is heading 3</h3>  
<h4>This is heading 4</h4>  
<h5>This is heading 5</h5>  
<h6>This is heading 6</h6>

Output: 

The <strong> tag is used to define text with strong importance. The content inside is typically displayed in **bold**.

<strong>This text is important!</strong>

Output: 

The <a> tag defines a **hyperlink**, which is used to link from one page to another. The most important attribute of the <a> element is the **href** attribute, which indicates the **link's destination**. If the <a> tag has no href attribute, it is only a placeholder for a hyperlink.

A linked page is normally displayed in the current browser window, unless you specify another target.

By default, links will appear as follows in all browsers:

* An unvisited link is underlined and blue
* A visited link is underlined and purple
* An active link is underlined and red

<a href="https://www.w3schools.com">Visit W3Schools.com!</a>

Output:  

The href (**Hypertext REFerence)** attribute specifies the URL of the page the link goes to. If the href attribute is not present, the <a> tag will not be a hyperlink.

**Tip:** You can use href="#top" or href="#" to link to the top of the current page!

An attribute is used to define the **characteristics** of an HTML element and is placed inside the element's opening tag. All attributes are made up of two parts − a **name** and a **value**

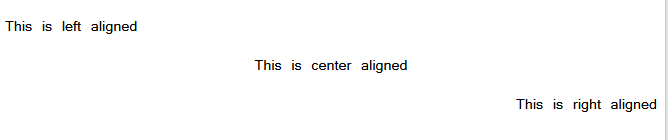
* The **name** is the property you want to set. For example, the paragraph **<p>** element in the example carries an attribute whose name is **align**, which you can use to indicate the alignment of paragraph on the page.
* The **value** is what you want the value of the property to be set and always put within quotations. The below example shows three possible values of align attribute: **left, center** and **right**.

<tag\_name attribute\_name = "value"> Content </tag\_name>

<p align = "left">This is left aligned</p>

<p align = "center">This is center aligned</p>

<p align = "right">This is right aligned</p>

Output: 

The <img> tag is used to embed an **image** in an HTML page.

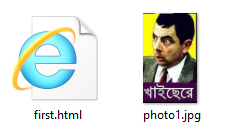
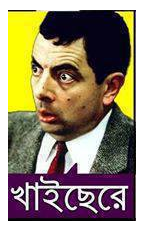
Images are not technically inserted into a web page; **images are linked to web pages**. The <img> tag creates a holding space for the referenced image.

The <img> tag has two required attributes:

* src - Specifies the path to the image
* alt - Specifies an alternate text for the image, if the image for some reason cannot be displayed

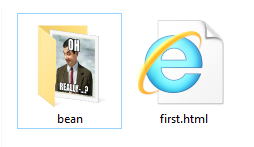
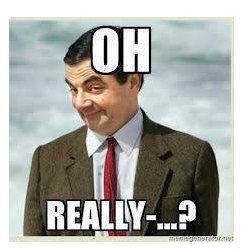
If html file and image are same directory.

<img src="photo1.jpg" alt="Picture of Mr. Bean">

Output:  

Html ফাইল যে ডিরেক্টরিতে আছে, তার ভিতরে কোন ফোল্ডারের ভিতর যদি ছবি থাকে।

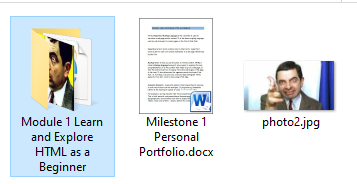
<img src="bean/photo3.jpg" alt="Picture of Mr. Bean">

Output:  

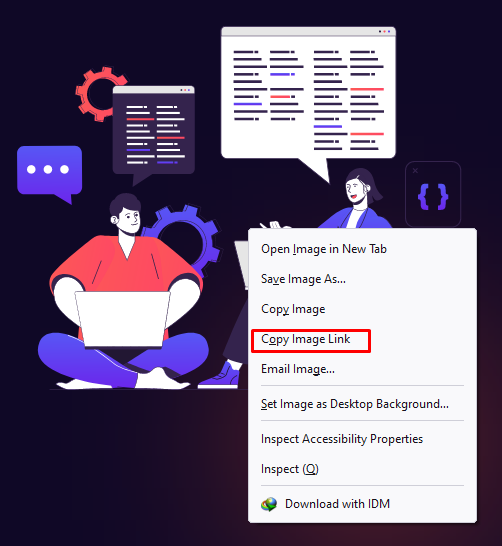
Html ফাইল যে ডিরেক্টরিতে আছে, তার বাহিরের কোথাও যদি ছবি থাকে।

<img src="../photo2.jpg" alt="Picture of Mr. Bean">

যত ফোল্ডার আগে থাকবে ততটা ../ দিতে হবে। এখানে একটা ফোল্ডার আগে তাই একবার ../; দুই ফোল্ডার আগে থাকলে দিতে হতো ../../ছবির নাম।

Output: 

প্রয়োজনে ইন্টারনেট সোর্সের কোন ছবিও ব্যবহার করা যাবে। সেক্ষেত্রে src–তে ওই ছবির ওয়েব লিংকটা দিতে হবে।



<img src="https://www.programming-hero.com/img/startup.svg" alt="software development team">

Output: 

লিস্ট দুই রকমের। একটা কোন অর্ডার বা সিরিয়াল অনুযায়ী আর আরেকটা অর্ডার বাদে র‌্যান্ডমলি তালিকা।

The <ol> tag defines an **ordered** list. An ordered list can be numerical or alphabetical.

The <ul> tag defines an **unordered** list. The list items will usually be displayed with bullet points.

The <li> tag is used to define each **list item**.

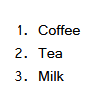
<ol>

<li>Coffee</li>

<li>Tea</li>

<li>Milk</li>

</ol>

Output: 

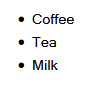
<ul>

<li>Coffee</li>

<li>Tea</li>

<li>Milk</li>

</ul>

Output: 

The <input> tag specifies an **input field** where the user can enter data. The <input> element is the most important form element. The <input> element can be displayed in several ways, depending on the type attribute.

The different input types are as follows:

* <input type = "button">
* <input type = "checkbox">
* <input type = "color">
* <input type = "date">
* <input type = "datetime-local">
* <input type = "email">
* <input type = "file">
* <input type = "hidden">
* <input type = "image">
* <input type = "month">
* <input type = "number">
* <input type = "password">
* <input type = "radio">
* <input type = "range">
* <input type = "reset">
* <input type = "search">
* <input type = "submit">
* <input type = "tel">
* <input type = "text"> (default value)
* <input type = "time">
* <input type = "url">
* <input type = "week">

input ট্যাগ মূলত form ট্যাগের ভিতর থাকে। The <form> tag is used to create an HTML form for user input.

<form>

First name: <input type="text">

Last name: <input type="password">

<input type="file">

<input type="date">

<input type="checkbox">

<input type="color">

<input type="radio">

<input type="range">

<input type="submit" value="Submit">

</form>

Output: 

The <button> tag defines a **clickable button**. Inside a <button> element you can put text (and tags like <i>, <b>, <strong>, <br>, <img>, etc.). That is not possible with a button created with the <input> element!

**Tip:** Always specify the type attribute for a <button> element, to tell browsers what type of button it is.

<button type="button">Click Me!</button>

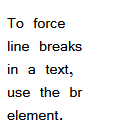
<button type="reset">Click Me for Reset!</button>

<button type="submit">Submit!</button>

Output: 

The <br> tag inserts a single line break.

<p>To force<br> line breaks<br> in a text,<br> use the br<br> element.</p>

Output: 

The <hr> HTML element represents a **thematic break (**বিষয়গত বিরতি) between paragraph-level elements: for example, a change of scene in a story, or a shift of topic within a section.

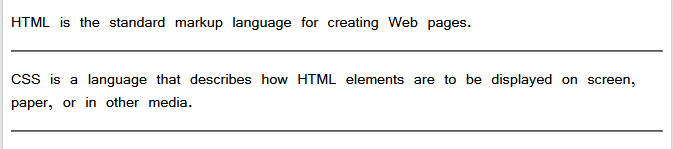
The <hr> element is most often displayed as a **horizontal rule** that is used to separate content (or define a change) in an HTML page.

<p>HTML is the standard markup language for creating Web pages.</p>

<hr>

<p>CSS is a language that describes how HTML elements are to be displayed on screen, paper, or in other media.</p>

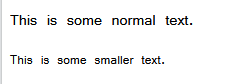
<hr>

Output: 

The <small> tag defines **smaller text** (like copyright and other side-comments).

<p>This is some normal text.</p>

<p><small>This is some smaller text.</small></p>

Output: 

The <div> tag defines **a division or a section** in an HTML document. The <div> tag is used as a **container** for HTML elements - which is then styled with CSS or manipulated with JavaScript.

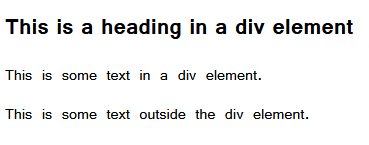
<div>

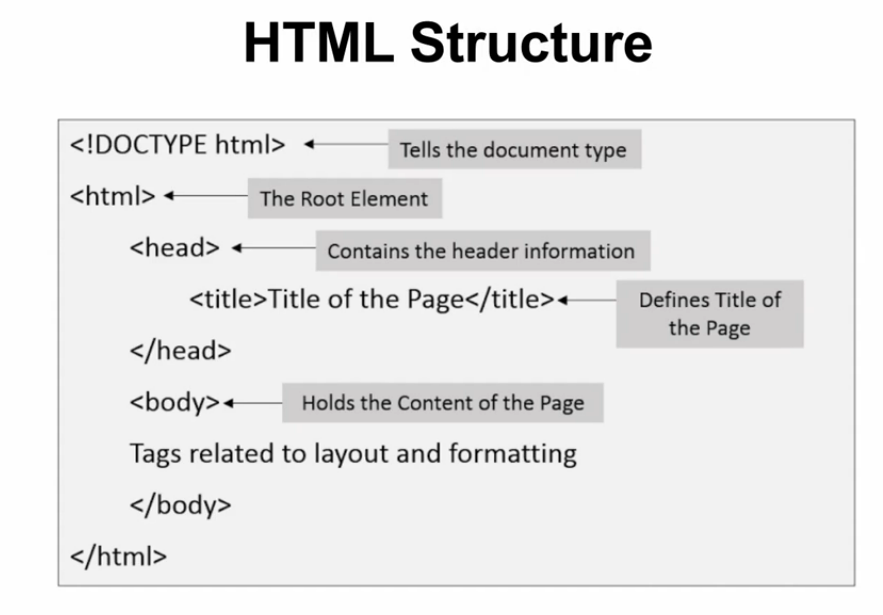
<h2>This is a heading in a div element</h2>

<p>This is some text in a div element.</p>

</div>

<p>This is some text outside the div element.</p>

Output: 



<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta name="author" content="Md. Faysal Alam Riyad">

    <title>Document</title>

<link rel="shortcut icon" href="photo1.jpg" type="image/x-icon">

</head>

<body>

</body>

</html>

All HTML documents must start with a <!DOCTYPE> declaration. The declaration is not an HTML tag. It is an "**information**" to the browser about what document type to expect.

The <meta> tag defines **metadata** about an HTML document. Metadata is data (**information**) about data.

<meta> tags always go inside the <head> element, and are typically used to specify character set, page description, keywords, author of the document, and viewport settings.

The <title> tag defines the **title of the document**. The title must be text-only, and it is shown in the browser's title bar or in the page's tab. The <title> tag is **required** in HTML documents!

<head>

<title>HTML Elements Reference</title>

</head>

Output: 

To add a favicon to your website, either save your **favicon image** to the root directory of your webserver, or create a folder in the root directory called images, and save your favicon image in this folder. A common name for a favicon image is "favicon.ico".

Next, add a <link> element to your "index.html" file, after the <title> element, like this:

<link rel="shortcut icon" href="photo1.jpg" type="image/x-icon">

Output: 

***Module 2 Learn and Explore CSS as a Beginner***

CSS stands for **Cascading Style Sheets**. CSS describes how HTML elements are to be displayed on screen, paper, or in other media. The style definitions are normally saved in external **.css** files.

CSS has an inherent hierarchy and styles of a higher precedence will overwrite rules of a lower precedence. Cascading defines level of existence and makes action accordingly. First styles will be checked at inline css level, if not exists it will check internal level in same file, if it’s also not exists then go with external level. Hence inline styles cascades internal styles and external styles, internal styles cascade external styles.

There are three types of CSS which are given below: 

* Inline CSS
* Internal or Embedded CSS
* External CSS

**Inline CSS:** Inline CSS contains the CSS property in the body section attached with element is known as inline CSS. This kind of style is specified within an HTML tag using the style attribute.

<body>

<p style = "color:#009900; font-size:50px; font-style:italic; text-align:center;">

Md. Faysal Alam Riyad </p>

</body>

#### Advantages of Inline CSS:

* You can easily and quickly insert CSS rules to an HTML page. That’s why this method is useful for testing or previewing the changes, and performing quick-fixes to your website.
* You don’t need to create and upload a separate document as in the external style.

#### Disadvantages of Inline CSS:

* Adding CSS rules to every HTML element is time-consuming and makes your HTML structure messy.
* Styling multiple elements can affect your page’s size and download time.

**Internal or Embedded CSS:** This can be used when a single HTML document must be styled uniquely. The CSS rule set should be within the HTML file in the head section i.e the CSS is embedded within the HTML file.

<head>

<style>

p {

text-align:center;

}

.GFG {

color:#009900;

font-size:50px;

font-weight:bold;

}

#geeks {

font-style:bold;

font-size:20px;

}

</style>

</head>

#### Advantages of Internal CSS:

* You can use class and ID selectors in this style sheet. Here’s an example:

.class {

property1 : value1;

property2 : value2;

property3 : value3;

}

#id {

property1 : value1;

property2 : value2;

property3 : value3;

}

* Since you’ll only add the code within the same HTML file, you don’t need to upload multiple files.

#### Disadvantages of Internal CSS:

* Adding the code to the HTML document can increase the page’s size and loading time.

**External CSS:** External CSS contains separate CSS file which contains only style property with the help of tag attributes (For example class, id, heading, … etc). CSS property written in a separate file with .css extension and should be linked to the HTML document using **link** tag. This means that for each element, style can be set only once and that will be applied across web pages.

<head>

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

</head>

#### Advantages of External CSS:

* Since the CSS code is in a separate document, your HTML files will have a cleaner structure and are smaller in size.
* You can use the same **.css** file for multiple pages.

#### Disadvantages of External CSS:

* Your pages may not be rendered correctly until the external CSS is loaded.
* Uploading or linking to multiple CSS files can increase your site’s download time.

**Properties of CSS:** Inline CSS has the highest priority, then comes Internal/Embedded followed by External CSS which has the least priority. Multiple style sheets can be defined on one page. If for an HTML tag, styles are defined in multiple style sheets then the below order will be followed.

* As Inline has the highest priority, any styles that are defined in the internal and external style sheets are overridden by Inline styles.
* Internal or Embedded stands second in the priority list and overrides the styles in the external style sheet.
* External style sheets have the least priority. If there are no styles defined either in inline or internal style sheet then external style sheet rules are applied for the HTML tags.

# Colors:

CSS Color property is used to set the color of HTML elements. This property is used to set font color, background color, border-color and other decorative effects.

Color of an element can be defined in the following ways:

* Built-In Color
* RGB Format
* RGBA Format
* Hexadecimal Notation
* HSL
* HSLA

**Built-In Color:** These are a set of predefined colors which are used by its name. For example: red, blue, green etc.

h1 {

color: color-name;

}

**RGB Format:** The RGB (Red, Green, and Blue) format is used to define the color of an HTML element by specifying the R, G, B values range between 0 to 255. Values can either be in percentage or integer. RGB format is the short form of '**RED GREEN** and **BLUE**'. For example: RGB value of Red color is (255, 0, 0), Green color is (0, 255, 0), Blue color is (0, 0, 255) etc.

h1 {

color: rgb (R, G, B);

}

**RGBA Format:** The RGBA format is similar to the RGB, but the difference is RGBA contains A (**Alpha**) which specify the transparency of elements. The value of alpha lies between 0.0 to 1.0; where 0.0 represents fully transparent and 1.0 represents not transparent.

h1 {

color:rgba(R, G, B, A);

}

**Hexadecimal Notation:** The hexadecimal notation begins with # symbol followed by 6 characters each range from 0 to F. In hexadecimal notation, the first two digits represent the **red (RR)** color value, the next two digits represent the **green (GG)** color value, and the last two digits represent the **blue (BB)** color value. For example: Red #FF0000, Green #00FF00, Blue #0000FF etc.

h1 {

color:#(0-F)(0-F)(0-F)(0-F)(0-F)(0-F);

}

## Short Hex Codes:

It is a short form of hexadecimal notation in which every digit is recreated to arrive at an equivalent hexadecimal value. For example, #7B6 becomes #77BB66 in hexadecimal.

The black color notation in hexadecimal is #000000, and the white color notation in hexadecimal is #FFFFFF. The black color notation in short hex is #000, and the white color notation in short hex is #FFF.

**HSL:** HSL stands for Hue, Saturation, and Lightness respectively. This format uses the cylindrical coordinate system.

* **Hue:** Hue is the degree of the color wheel. Its value lies between 0 to 360 where 0 represents red, 120 represents green and 240 represents blue color.
* **Saturation:** It takes percentage value, where 100% represents completely saturated, while 0% represents completely unsaturated (gray).
* **Lightness:** It takes percentage value, where 100% represents white, while 0% represents black.

h1 {

color:hsl(H, S, L);

}

**HSLA:** The HSLA color property is similar to HSL property, but the difference is HSLA contains A (**Alpha**) which specify the transparency of elements. The value of alpha lies between 0.0 to 1.0; where 0.0 represents fully transparent and 1.0 represents not transparent.

h1 {

color:hsla(H, S, L, A);

}

**Background Color:** It is used to set the background color of an HTML element.

h1 {

background-color:color\_name;

}

**Border Color:** It is used to set the border color of an element. Border-style is used to set the border type.

h1 {

border-style:solid/dashed/dotted

border-color:color\_name;

}

**Height and Width**: Height and Width in CSS are used to set the height and width of boxes.

It does not include padding, borders or margins, whereas it sets the height/width of the area inside the padding, border, and margin of the element. It can accept the length and percentage values. But it does not allow negative values.

If we set the height to a numeric value (like in px, %, etc.), the content can be overflow if it does not fit in the given height.

**CSS Units:** There are various units in CSS to express the measurement and length. A CSS unit is used to determine the property size, which we set for an element or its content. The units in CSS are required to define the measurement such as margin: 20px; in which the **px** (or pixel) is the CSS unit. They are used to set margin, padding, lengths, and so on.

We cannot apply the whitespace between the number and the unit. The unit can be omitted for the value 0. Some properties of CSS allow the negative values of length.

The length unit in CSS is of two types:

* Absolute length.
* Relative length.

**Absolute lengths:** These are the fixed-length units, and the length expressed using the absolute units will appear as exactly that size. It is not recommended to use on-screen, because the size of the screen varies too much. So, the absolute units should be used when the medium of output is known, such as the print layout.

Absolute units are useful when the responsiveness is not considered in a project. They are less favorable for the responsive sites because they do not scale when the screen changes.

Generally, absolute lengths are considered to be the same size always. The absolute length units are tabulated as follows:

## 

# Pixels (px) are relative to the viewing device. For low-dpi devices, 1px is one device pixel (dot) of the display. For printers and high resolution screens 1px implies multiple device pixels.

**Relative lengths**: Relative units are good to style the responsive site because they scale relative to the window size or the parent. They specify the length, which is relative to another length property.

Depending on the device, if the size of the screen varies too much, then the relative length units are the best because they scale better between the different rendering mediums. We can use the relative units as the default for the responsive units. It helps us to avoid update styles for different screen sizes.

The relative length units are tabulated as follows:

## 

The em and rem units are practical in creating perfectly scalable layout!

Viewport = the browser window size. If the viewport is 50cm wide, 1vw = 0.5cm.

**font-size:** It is used to set the font size of an HTML element. The font-size can be set in different ways like in “pixels, percentage, em or we can set values like small, large” etc.

#### If we do not define a font-size, then for the normal text such as paragraphs, the default size is 16px, which is equal to 1em. Most of the developers prefer ****em**** instead of ****pixels****. It is recommended by the World Wide Web consortium (W3C).

font-size: font size value;

## Property Values:

## 

**id Attribute**: The HTML id attribute is used to specify a unique id for an HTML element. You cannot have more than one element with the same id in an HTML document.

## Using the id Attribute:

The id attribute specifies a unique id for an HTML element. The value of the id attribute must be unique within the HTML document.

The id attribute is used to point to a specific style declaration in a style sheet. It is also used by JavaScript to access and manipulate the element with the specific id.

**The syntax for id is:** Write a hash character (#), followed by an id name. Then, define the CSS properties within curly braces {}.

In the following example we have an <h1> element that points to the id name "myHeader". This <h1> element will be styled according to the #myHeader style definition in the head section:

<style>  
#myHeader {  
  background-color: lightblue;  
  color: black;   
}   
</style>  
  
<h1 id="myHeader">My Header</h1>

**Note:** The id name is case sensitive! The id name must contain at least one character, cannot start with a number, and must not contain whitespaces (spaces, tabs, etc.).

**class Attribute**: The HTML class attribute is used to specify a class for an HTML element. Multiple HTML elements can share the same class.

**The syntax for class is:** Write a dot character (.), followed by a class name. Then, define the CSS properties within curly braces {}.

## Using the class Attribute:

The class attribute is often used to point to a class name in a style sheet. It can also be used by a JavaScript to access and manipulate elements with the specific class name.

In the following example we have three <div> elements with a class attribute with the value of "city". All of the three <div> elements will be styled equally according to the .city style definition in the head section:

<style>  
.city {  
   background-color: tomato;  
  color: white;  
  border: 2px solid black;  
  margin: 20px;  
  padding: 20px;  
}   
</style>

<div class="city">  
  <h2>London</h2>  
  <p>London is the capital of England.</p>  
</div>  
  
<div class="city">  
  <h2>Paris</h2>  
  <p>Paris is the capital of France.</p>  
</div>  
  
<div class="city">  
  <h2>Tokyo</h2>  
  <p>Tokyo is the capital of Japan.</p>  
</div>

## Output:

## 

## ****Note**:** The class name is case sensitive!

<h2 class="city main">London</h2>

একটা ট্যাগের ভিতর প্রয়োজনে একাধিক ক্লাসও ব্যবহার করা যাবে। এখানে ***city*** একটা ক্লাস, ***main*** আরেকটা ক্লাস। দুটোই এপ্লাই হবে।

এবং চাইলে একটা ট্যাগের ভিতর আইডির পাশাপাশি ক্লাসও ব্যবহার করা যাবে।

<h3 id="city" class="hot">Paris is the capital of France</h3>

**Difference between id and class attribute:** The only difference between them is that “id” is unique in a page and can only apply to at most one element, while “class” selector can apply to multiple elements.

# span Tag: HTML <span> tag is used as a generic container of inline elements. It is used for styling purpose to the grouped inline elements (using class and id attribute or inline style).

The <span> tag does not have any default meaning or rendering.

The <span> tag can be useful for the following task:

* To change the language of a part of the text.
* To change the color, font, background of a part of text using CSS
* To apply the scripts to the particular part of the text.

#### Note: HTML <span> is much similar as <div> tag, but <div> is used for block-level elements and <span> tag is used for inline elements.

<p>My mother has <span style="color:blue;font-weight:bold">blue</span> eyes and my father has <span style="color:darkolivegreen;font-weight:bold">dark green</span> eyes.</p>

Output: 

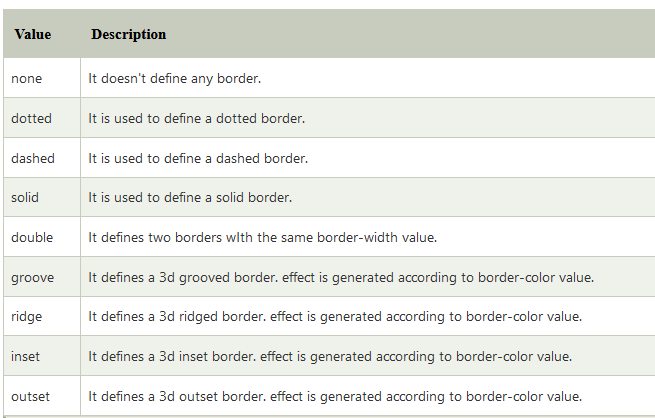
# CSS Border: The CSS border is a shorthand property used to set the border on an element.

The CSS border properties are use to specify the style, color and size of the border of an element. The CSS border properties are given below

* border-style
* border-color
* border-width
* border-radius

## CSS border-style: The Border style property is used to specify the border type which you want to display on the web page. There are some border style values which are used with border-style property to define a border.

 {border-style: dotted;}



**CSS border-width**: The border-width property specifies the width of the four borders. The width can be set as a specific size (in px, pt, cm, em, etc) or by using one of the three pre-defined values: thin, medium, or thick**.**

border-width: 25px; /\* 25px top right bottom and left\*/

border-width: 25px 10px; /\* 25px top and bottom, 10px right and left \*/

border-width: 25px 10px 4px; /\* 25px top, 10px right and left, 4px bottom \*/

border-width: 25px 10px 4px 35px; /\* 25px top, 10px right, 4px bottom and 35px left \*/

**Border Color:** The border-color property is used to set the color of the four borders.

The color can be set by:

* name - specify a color name, like "red"
* HEX - specify a HEX value, like "#ff0000"
* RGB - specify a RGB value, like "rgb(255,0,0)"
* HSL - specify a HSL value, like "hsl(0, 100%, 50%)"
* transparent

**Note:** If border-color is not set, it inherits the color of the element.

{border-color: green;}

Border width style Color একসাথে ব্যবহার করা যাবে।

{border: 5px dotted yellow;}

### border-radius: With border-radius, you can remove sharp edges from the borders in order to make them into rounded corners. The value is specified in pixels (px), and percentage (%) too, depending on how you prefer it.

{border-radius: 10px;}

border-width এর মতন border-radius ও ১, ২, ৩ এবং ৪টি ভ্যালু ব্যবহার করা যাবে।

{border-radius: 10px;}

{border-radius: 10px, 25px;}

{border-radius: 10px, 15px, 25px;}

{border-radius: 10px, 25px, 15px, 18px;}

**CSS Margins:** The margin property defines the outermost portion of the box model, creating space around an element, outside of any defined borders. With CSS, you have full control over the margins. There are properties for setting the margin for each side of an element (top, right, bottom, and left).

**Margin - Individual Sides**

CSS has properties for specifying the margin for each side of an element:

* margin-top
* margin-right
* margin-bottom
* margin-left

All the margin properties can have the following values:

* auto - the browser calculates the margin
* *length* - specifies a margin in px, pt, cm, etc.
* *%* - specifies a margin in % of the width of the containing element
* inherit - specifies that the margin should be inherited from the parent element

**Tip:** Negative values are allowed.

{  
  margin-top: 100px;  
  margin-bottom: 100px;  
  margin-right: 150px;  
  margin-left: 80px;  
}

To shorten the code, it is possible to specify all the margin properties in one property.

{margin: 25px 50px 75px 100px;} /\*top, right, bottom, left\*/

## CSS Padding: The CSS padding properties are used to generate space around an element's content, inside of any defined borders.

With CSS, you have full control over the padding. There are properties for setting the padding for each side of an element (top, right, bottom, and left).

All the padding properties can have the following values:

* *length* - specifies a padding in px, pt, cm, etc.
* *%* - specifies a padding in % of the width of the containing element
* inherit - specifies that the padding should be inherited from the parent element

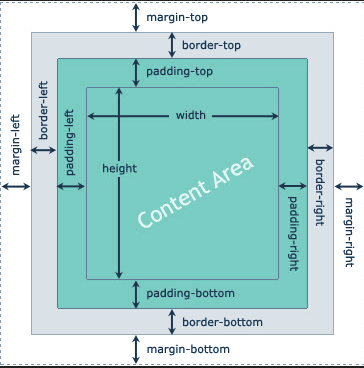
**Note:** Negative values are not allowed.

{padding: 25px 50px 75px 100px;} /\*top, right, bottom, left\*/

# CSS Box Model:

Every element that can be displayed on a web page is comprised of one or more rectangular boxes. CSS box model typically describes how these rectangular boxes are laid out on a web page. These boxes can have different properties and can interact with each other in different ways, but every box has a ***content area*** and optional surrounding ***padding***, ***border***, and ***margin areas***.

The following diagram demonstrates how the width, height, padding, border, and margin CSS properties determines how much space an element can take on a web page.



Padding is the transparent space between the element's content and its border (or edge of the box, if it has no border), whereas margin is the transparent space around the border.

Also, if an element has the [background color](https://www.tutorialrepublic.com/css-tutorial/css-background.php) it will be visible through its padding area. The margin area is always remain transparent, it is not affected by the element's background color, however, it causes the background color of the parent element to be seen through it.

## Width and Height of the Elements:

## Usually when you set the width and height of an element using the CSS [width](https://www.tutorialrepublic.com/css-reference/css-width-property.php) and [height](https://www.tutorialrepublic.com/css-reference/css-width-property.php) properties, in reality you are only setting the width and height of the content area of that element. The actual width and height of the element's box depends on the several factors.

The actual space that an element's box might take on a web page is calculated like this:

|  |  |
| --- | --- |
| Box Size | CSS Properties |
| Total Width | width + padding-left + padding-right + border-left + border-right + margin-left + margin-right |
| Total Height | height + padding-top + padding-bottom + border-top + border-bottom + margin-top + margin-bottom |

# text-align Property:

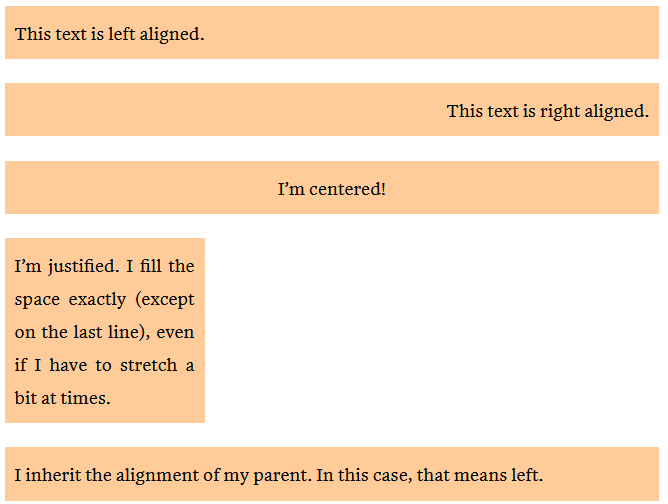
The **text-align** property in CSS is used to specify the horizontal alignment of text in an element ie., it is used to set the alignment of the content horizontally, inside a block element or table-cell box.

{ text-align: justify; }

These are the traditional values for text-align:

* left – The default value. Content aligns along the left side.
* right – Content aligns along the right side.
* center – Content centers between the left and right edges. White space on the left and right sides of each line should be equal.
* justify – Content spaces out such that as many blocks fit onto one line as possible and the first word on that line is along the left edge and the last word is along the right edge.
* inherit – The value will be whatever the parent element’s is.

“Content” is used here as as term instead of “text”, because while text-align certainly affects text, it affects all inline or inline-block elements in that container.



# font-family Property:

The font-family property specifies the font for an element.

The font-family property can hold several font names as a "fallback" system. If the browser does not support the first font, it tries the next font.

There are two types of font family names:

* **family-name** - The name of a font-family, like "times", "courier", "arial", etc.
* **generic-family** - The name of a generic-family, like "serif", "sans-serif", "cursive", "fantasy", "monospace".

Start with the font you want, and always end with a generic family, to let the browser pick a similar font in the generic family, if no other fonts are available.

**Note:** Separate each value with a comma.

**Note:** If a font name contains white-space, it must be quoted. Single quotes must be used when using the "style" attribute in HTML.

{font-family: Cambria, Cochin, Georgia, Times, 'Times New Roman', serif;}

Let's define the generic-family categories.

**serif:** It is mainly used when we are writing the text for printing, such as books, magazines, newspapers, etc. It includes the font-family such as Georgia, Garamond, Times New Roman, Minion, and many more.

**sans-serif:** It is a modern, formal, and simple style. It is widely used but most often in the digital form of text. It includes the font-family that are Arial, Calibri, Verdana, Futura, Lato, and many more.

**cursive:** It is mainly used for writing the invitation letter, informal messages, etc. It is like a handwritten text which is written by a pen or a brush. The font-family that it includes is Insolente, Corsiva, Flanella, Belluccia, Zapfino, and many more.

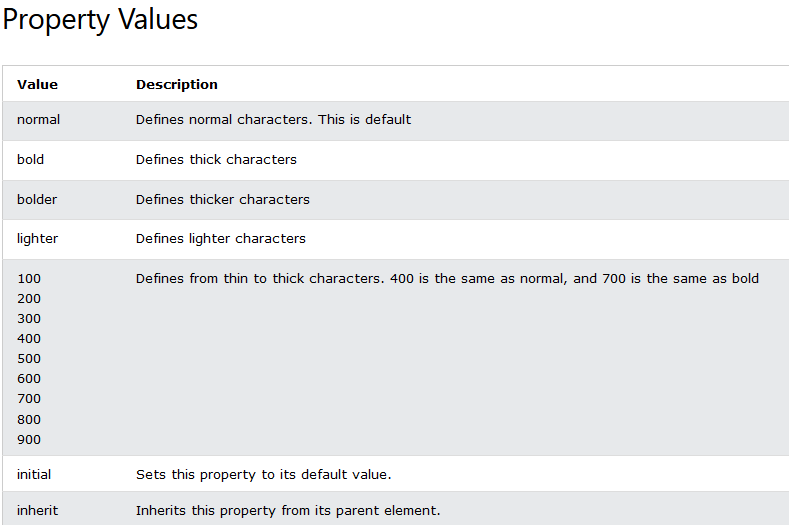
**monospace:** It is for instructions, mailing address, typewritten text, etc. It includes the font-family that is Monaco, SimSun, Courier, Consolas, Inconsolata, and many more.

**fantasy:** It makes the text expressive, decorative, and impactful. It includes the font-family that is Impact, Copperplate, Cracked, Critter, and many more.

# font-weight Property:

The font-weight property sets how thick or thin characters in text should be displayed.

{ font-weight: bold;}



# font-style Property:

The font-style property specifies the font style for a text.

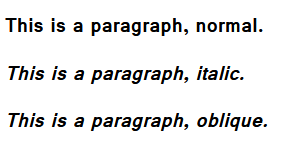
{

font-style: normal;

font-style: italic;

font-style: oblique;

}

Output: 

# display Property:

The display property specifies the display behavior (the type of rendering box) of an element. In HTML, the default display property value is taken from the HTML specifications or from the browser/user default style sheet. Every element has a default display value according to its nature. Every element on the webpage is a rectangular box and the CSS property defines the behavior of that rectangular box.

There are following CSS display values which are commonly used.

1. display: inline;
2. display: inline-block;
3. display: block;
4. display: run-in;
5. display: none;

## display inline: The inline element takes the required width only. It doesn't force the line break so the flow of text doesn't break in inline.

## display inline-block: The CSS display inline-block element is very similar to inline element but the difference is that you are able to set the width and height.

## display block: The CSS display block element takes as much as horizontal space as they can. Means the block element takes the full available width. They make a line break before and after them.

## display run-in: This property doesn't work in Mozilla Firefox. These elements don't produce a specific box by themselves.

* If the run-in box contains a bock box, it will be same as block.
* If the block box follows the run-in box, the run-in box becomes the first inline box of the block box.
* If the inline box follows the run-in box, the run-in box becomes a block box.

## display none: The "none" value totally removes the element from the page. It will not take any space.

# Inline vs Inline-Block Display in CSS:

**display: inline-block** brought a new way to create side by side boxes that collapse and wrap properly depending on the available space in the containing element. It makes layouts that were previously accomplished with floats easier to create. No need to clear floats anymore.

Compared to display: inline, the major difference is that **inline-block** allows to set a width and height on the element. Also, with display: inline, top and bottom margins & paddings are not respected, and with display: inline-block they are.

Now, the difference between display: inline-block and display: block is that, with **display: block**, a line break happens after the element, so a block element doesn’t sit next to other elements. Here are some visual examples:

## display: inline

Notice here how the width and height are not respected, and how the padding top and bottom are present, but overlap over the lines above and under.

span.box {

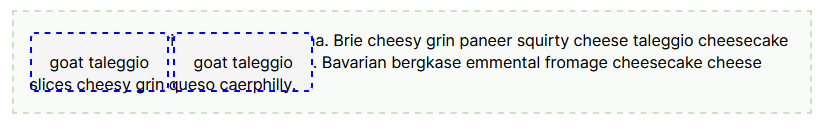
display: inline; /\* the default for span \*/

width: 100px;

height: 160px;

padding: 18px;

}

Output: 

## display: inline-block

Here the width, height and padding are respected, but the two copies of the element can still sit side by side.

span.box {

display: inline-block;

width: 100px;

height: 160px;

padding: 18px;

}

## 

## display: block

Here again everything is respected, but the elements don’t sit side by side.

span.box {

display: block;

width: 100px;

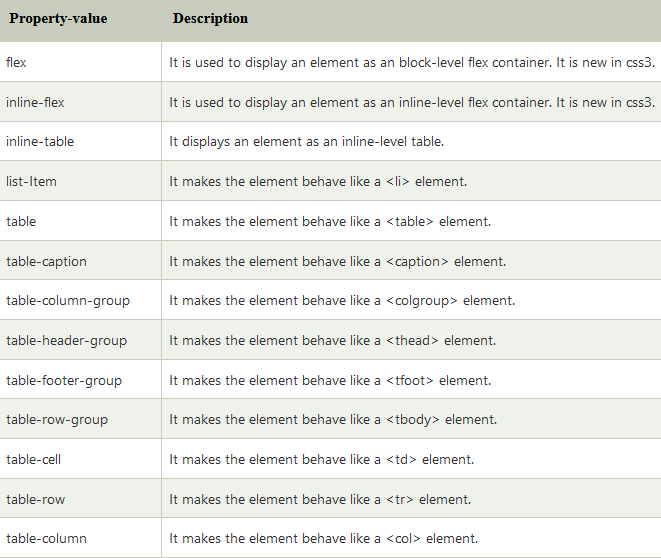
height: 160px;

padding: 18px;

}

## 

## Other CSS display values:



**Module 3: Git, Source Control, Github and Hosting**

**Git** is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

**Source code management** (SCM) is used to track modifications to a source code repository. SCM tracks a running history of changes to a code base and helps resolve conflicts when merging updates from multiple contributors.

**GitHub** is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere.

A **repository** is usually used to organize a single project. Repositories can contain folders and files, images, videos, spreadsheets, and data sets -- anything your project needs. Often, repositories include a README file, a file with information about your project. README files are written in the plain text Markdown language.

#### git branch, merge branches:

ব্রাঞ্চের তালিকা দেখার জন্য (তারকা বর্তমান ব্রাঞ্চ নির্দেশ করে):

git branch

নতুন ব্রাঞ্চ তৈরি:

git branch [branch name]

ব্রাঞ্চ ডিলিট:

git branch -d [branch name]

স্পেসিফিক ব্রাঞ্চে স্যুইচ করতে চাইলে মানে এক ব্রাঞ্চ থেকে আরেক ব্রাঞ্চে যেতে চাইলে:

git checkout [branch name]

কোন একটা ব্রাঞ্চে পুশ করতে চাই, কিন্তু ওই ব্রাঞ্চটা গিটহাবে তৈরী করা নাই। তাই ব্রাঞ্চটা তৈরী করা এবং পুশ করার জন্য কমান্ড হল:

git push --set-upstream origin [branch name]

অ্যাক্টিভ ব্রাঞ্চের সাথে আরেকটি ব্রাঞ্চ মার্জ:

git merge [branch name]

যেমন, main ব্রাঞ্চ-এর সাথে image-add নামের আরেকটা ব্রাঞ্চ মার্জ করব। মানে image-add ব্রাঞ্চটার কোডগুলা main ব্রাঞ্চ-এ যুক্ত হবে। এর জন্য আমাদের প্রথমে main ব্রাঞ্চ-এর মধ্যে থাকতে হব। তারপর কমান্ড হল: git merge image-add

***Module 5.5 [Bonus] Box Model, Pseudo Class, Position***