UNIT-1:: Introduction to HTML
Module-1

1.1. Introduction

HTML stands for **Hyper Text Markup Language**, which is the most widely used language on Web to develop web pages. **HTML** was created by Berners-Lee in late 1991 but "HTML 2.0" was the first standard HTML specification which was published in 1995. HTML 4.01 was a major version of HTML and it was published in late 1999. Though HTML 4.01 version is widely used but now we are having HTML- 5 version which is an extension to HTML 4.01, and this version was published in 2012. Currently HTML 5.0 is the new version of the HTML and it was published in 2017.

1.2.1.What is HTML?

HTML stands for Hypertext Markup Language. This markup language is used to create web documents. A web document is viewed in a web browser, like the one you are using to read this document.

- **Hypertext** refers to the way in which Web pages (HTML documents) are linked together. Thus, the link available on a webpage is called Hypertext.
- As its name suggests, HTML is a **Markup Language** which means you use HTML to simply "mark-up" a text document with tags that tell a Web browser how to structure it to display.

  Now, HTML is being widely used to format web pages with the help of different tags available in HTML language.

1.2.2 .Basic Structure of HTML Document

An HTML Document is mainly divided into two parts:

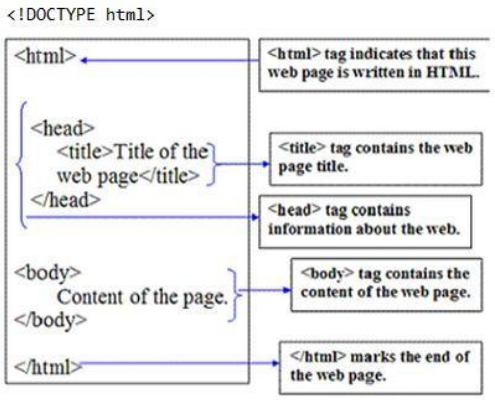
- **HEAD**: This contains the information about the HTML document. For Example, Title of the page, version of HTML, Meta Data etc.
- **BODY**: This contains everything you want to display on the Web Page.

Let us now have a look on the basic structure of HTML. That is the code which is must for every webpage to have:

[The <!DOCTYPE html> declaration defines this document to be HTML5

The <!DOCTYPE> declaration tag is used by the web browser to understand the version of the HTML used in the document. Current version of HTML is 5 and it makes use of the following declaration −

<!DOCTYPE html>]



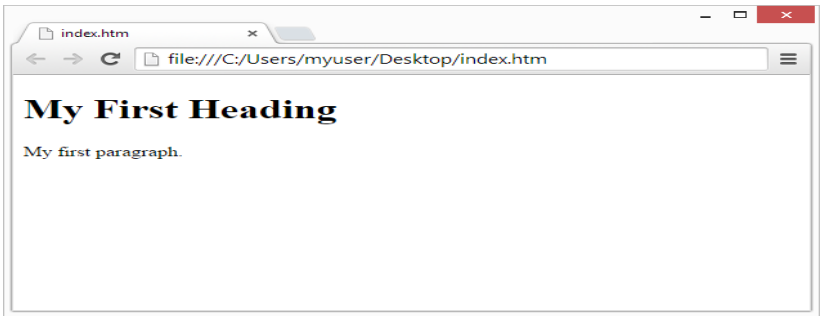
Example:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Page Title</title>
    </head>
    <body>
        <h1>Homepage Headline</h1>
        <p>This is a paragraph.</p>
    </body>
</html>
```

Example Explained

- The **DOCTYPE** declaration defines the document type to be HTML
- The text between **<html>** and **</html>** describes an HTML document
- The text between **<head>** and **</head>** provides information about the document
- The text between **<title>** and **</title>** provides a title for the document
- The text between **<body>** and **</body>** describes the visible page content
- The text between **<h1>** and **</h1>** describes a heading
- The text between ** ** and ** ** describes a paragraph

Using this description, a web browser can display a document with a heading and a paragraph.



1.2.3. HTML Editors

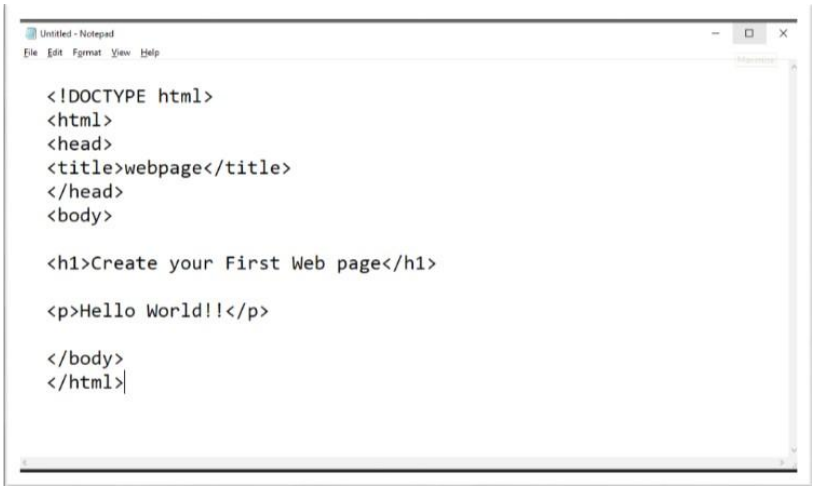
- Text editors are the programs which allow editing in a written text, hence to create a web page we need to write our code in some text editor.
- There are various types of text editors available which you can directly download, but for a beginner, the best text editor is Notepad (Windows) or TextEditor (Ubuntu).
- After learning the basics, you can easily use other professional text editors which are, **Notepad++, Sublime Text, Vim, etc**.

Step 1: Open TextEditor (Ubuntu)

Open up your computer's plain text editor and create a new file.

Open **Applications Menu > TextEditor**

Step 2: Write code in HTML



Step 3: Save the HTML file with .htm or .html extension.

Note: It is important that the extension .html is specified — some text editors, such as Notepad, will automatically save it as .txt otherwise.

There are a few important things to keep in mind when you save it the file:

1. Use the .htmlHTML file extension, i.e. about_me.html
2. Don't use any spaces or special characters in the file name. Use underscores (_) or dashes (-) instead.
3. Decide where in your computer you will save the file, and make sure to remember the location

Step 4: View the HTML Page in Your Browser

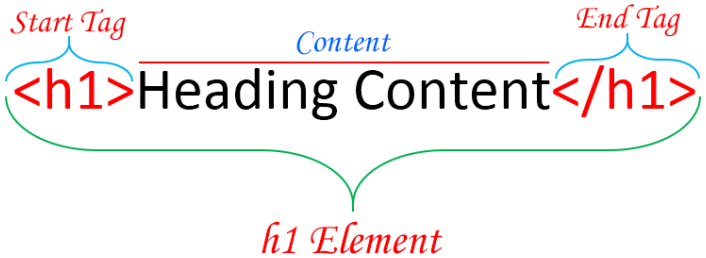
Open the saved HTML file in your favorite browser (double click on the file, or right-click - and choose "Open with").

The result will look much like this:

1.3. HTML – Tag and Elements

I am going to explain about HTML tags and elements together here. Because sometimes you may be confused about the difference between Tags and Elements. Let's have a look at the image below is actually showing which one is tag and which one is element.

The HTML **element** is everything from the start tag to the end tag:



1.3.1. What are HTML Tags

HTML Tags are used to mark up any text or graphics. According to the image above, the <h1> tag is surrounded by 2 angle brackets –<| and –>–; called start tag. And the end tag </h1> is almost same but it has only a forward slash —/| before the tag name (h1). So here <h1> is start tag and </h1> is end tag.

1.3.2. HTML Elements

An HTML file is made of elements. These elements are responsible for creating web pages and define content in that webpage. An element in HTML usually consist of a start tag <tag name>, close tag </tag name> and content inserted between them. Technically, an element is a collection of start tag, attributes, end tag, content between them.

```
<tagname>Content goes here...</tagname>
```

1.3.3. HTML Empty Elements

There are some HTML elements which don't need to be closed, such as <img.../>, <hr /> and
 elements. These are known as void elements.

Note: Some elements does not have end tag and content, these elements are termed as empty elements or self-closing element or void elements.

Start Tag	Description
** **	It produces a line break in text.
<hr>	Horizontal rule. It represents a thematic break in an HTML page.
<img>	It represents an image in a document.

1.3.4. Nested HTML Elements

It is very much allowed to keep one HTML element inside another HTML element −

Example

```
<!DOCTYPE html>
<html>

   <head>
      <title>Nested Elements Example</title>
   </head>

   <body>
      <h1>This is <i>italic</i> heading</h1>
      <p>This is <u>underlined</u> paragraph</p>
   </body>

</html>

```

This will display the following result:

This is *italic* heading

This is underlined paragraph

1.4. HTML - Basic Tags

1.4.1. Heading Tags

Any document starts with a heading. You can use different sizes for your headings. HTML also has six levels of headings, which use the elements **<h1>, <h2>, <h3>, <h4>, <h5>,** and **<h6>**. While displaying any heading, browser adds one line before and one line after that heading.

Example:

```
<!DOCTYPE html>
<html>

   <head>
      <title>Heading Example</title>
   </head>

   <body>
      <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>
   </body>

</html>
```

Result:

This is heading 1

This is heading 2

This is heading 3

This is heading 4

This is heading 5

This is heading 6

1.4.2. Paragraph Tag

The ** ** tag offers a way to structure your text into different paragraphs. Each paragraph of text should go in between an opening and a closing tag as shown below in the example −

Example:

```
<!DOCTYPE html>
<html>

   <head>
      <title>Paragraph Example</title>
   </head>

   <body>
      <p>Here is a first paragraph of text.</p>
      <p>Here is a second paragraph of text.</p>
      <p>Here is a third paragraph of text.</p>
   </body>

</html>
```

Result:

Here is a first paragraph of text.

Here is a second paragraph of text.

Here is a third paragraph of text.

1.4.3. Line Break Tag

Whenever you use the **
** element, anything following it starts from the next line. This tag is an example of an **empty** element, where you do not need opening and closing tags, as there is nothing to go in between them.

The
 tag has a space between the characters **br** and the forward slash. If you omit this space, older browsers will have trouble rendering the line break, while if you miss the forward slash character and just use
 it is not valid in XHTML.

Example:

```
<!DOCTYPE html>
<html>

   <head>
      <title>Line Break  Example</title>
   </head>

   <body>
      <p>Hello<br />
         You have uploaded your task.<br />
         Thanks<br />
         Ranaveer</p>
   </body>

</html>

```

Result:

Hello
You have uploaded your task.
Thanks
Ranaveer

1.4.4. Centering Content

You can use ** ** tag to put any content in the center of the page.

Example:

```
<!DOCTYPE html>
<html>

   <head>
      <title>Centring Content Example</title>
   </head>

   <body>
      <p>This text is not in the center.</p>

      <center>
         <p>This text is in the center.</p>
      </center>
   </body>

</html>

```

1.4.5. Horizontal Lines

You can use the <hr> tag to create horizontal rules or lines to visually separate content sections on a web page. Like
, the <hr> tag is also an empty element. Here's an example:

Example:

```
<!DOCTYPE html>
<html>

   <head>
      <title>Horizontal Line Example</title>
   </head>

   <body>
      <p>This is paragraph one and should be on top</p>
      <hr />
      <p>This is paragraph two and should be at bottom</p>
   </body>

</html>
```

Result:

This is paragraph one and should be on top

This is paragraph two and should be at bottom

1.4.6. Managing White Spaces

Normally the browser will display the multiple spaces created inside the HTML code by pressing the space-bar key or tab key on the keyboard as a single space. Multiple line breaks created inside the HTML code through pressing the enter key is also displayed as a single space. The following paragraphs will be displayed in a single line without any extra space:

Example:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Example</title>
    </head>
    <body>
        <p>This paragraph contains  multiple   spaces    in the source code.</p>
        <p>
            This paragraph


            contains multiple tabs and line breaks
            in the source code.
        </p>
    </body>
</html>
```

Result:

This paragraph contains multiple spaces in the source code.

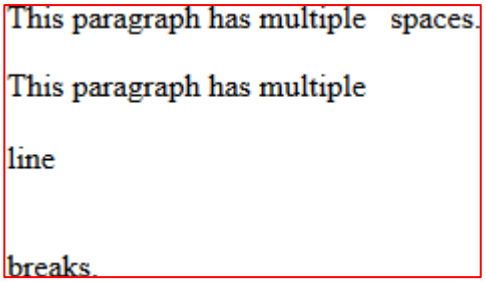
This paragraph contains multiple tabs and line breaks in the source code.

Insert   for creating extra consecutive spaces, while insert
 tag for creating line breaks on your web pages, as demonstrated in the following example:

Example:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Example</title>
    </head>
    <body>
        <p>This paragraph has multiple   spaces.</p>
        <p>This paragraph has multiple<br><br>line<br><br><br>breaks.</p>
    </body>
</html>
```

Result:



You can also use   for non-breakable lines. Suppose you want to use the phrase "12 Angry Men." Here, you would not want a browser to split the "12, Angry" and "Men" across two lines −

An example of this technique appears in the movie "12 Angry Men."

In cases, where you do not want the client browser to break text, you should use a no breaking space entity ** ** instead of a normal space. For example, when coding the "12 Angry Men" in a paragraph, you should use something similar to the following code –

Example:

```
<!DOCTYPE html>
<html>

    <head>
        <title>Nonbreaking Spaces Example</title>
    </head>

    <body>
        <p>An example of this technique appears in the movie "12 Angry 
            ;Men."</p>
    </body>

</html>
```

Result:

An example of this technique appears in the movie "12 Angry Men."

1.4.7. Preformatted Text

Sometimes, using ,
, etc. for managing spaces isn't very convenient. Alternatively, you can use the <pre> tag to display spaces, tabs, line breaks, etc. exactly as written in the HTML file. It is very helpful in presenting text where spaces and line breaks are important like poem or code.

The following example will display the text in the browser as it is in the source code:

Example:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Example</title>
    </head>
    <body>
        <pre>
            Twinkle, twinkle, little star,
            How I wonder what you are!
            Up above the world so high,
            Like a diamond in the sky.
        </pre>
    </body>
</html>
```

Result:

```
Twinkle, twinkle, little star,
How I wonder what you are!
Up above the world so high,
Like a diamond in the sky.
```

1.4.8. Comment Tag

You can add comments to your HTML source by using the following syntax:

```
<!-- Write your comments here -->
```

Comments are not displayed by the browser, but they can help document your HTML source code. With comments you can place notifications and reminders in your HTML:

Example:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Example</title>
    </head>
    <body>
         <!-- This is a comment -->
        <p>This is a paragraph.</p>
        <!-- Remember to add more information here -->
    </body>
</html>
```

Module-2

1.5. HTML – Formatting

HTML Formatting is a process of formatting text for better look and feel. HTML provides us ability to format text without using CSS. There are many formatting tags in HTML. These tags are used to make text bold, italicized, or underlined.

Following is the list of HTML formatting text.

Element name	Description
****	This is a physical tag, which is used to bold the text written between it.
****	This is a logical tag, which tells the browser that the text is important.
<i>	This is a physical tag which is used to make text italic.
****	This is a logical tag which is used to display content in italic.
<mark>	This tag is used to highlight text.
<u>	This tag is used to underline text written between it.
<tt>	This tag is used to appear a text in teletype. (not supported in HTML5)
<strike>	This tag is used to draw a strikethrough on a section of text. (Not supported in HTML5)
<sup>	It displays the content slightly above the normal line.
<sub>	It displays the content slightly below the normal line.
<del>	This tag is used to display the deleted content.
<ins>	This tag displays the content which is added
<big>	This tag is used to increase the font size by one conventional unit.
<small>	This tag is used to decrease the font size by one unit from base font size.

1.5.1.Bold Text

HTML and formatting elements. If you write anything within ... element, is shown in bold letters. If you write anything between ....., is shown important text.

Example

```
<!DOCTYPE html>
<html>
<head>
    <title>formatting elements</title>
</head>
<body>
<h1>Explanation of formatting element</h1>
<p><strong>This is an important content</strong>, and this is normal content</p>
<p><b>This is an important content</b>, and this is normal content</p>
</body>
</html>
```

Result:

Explanation of formatting element

This is an important content, and this is normal content

This is an important content, and this is normal content

1.5.2. Italic Text

HTML and formatting elements.

If you write anything within ..... element, is shown in italic letters. The HTML tag is a logical element, which will display the enclosed content in italic font, with added semantics importance.

Example

```
<!DOCTYPE html>
<html>
<head>
    <title>formatting elements</title>
</head>
<body>
<h1>Explanation of italic formatting element</h1>
<p><i>This is an Italic content</i>, which displayed in italic font.</p>
<p><em>This is an important content</em>, which displayed in italic font.</p>
</body>
</html>
```

Result

Explanation of italic formatting element

This is an Italic content, which displayed in italic font.

This is an important content, which displayed in italic font.

1.5.3.HTML Marked formatting

If you want to mark or highlight a text, you should write the content within
<mark>..... </mark>.

Example:

```
<h2> I want to put a <mark> Mark</mark> on your face</h2>
```

Result:

I want to put a Mark on your face

1.5.4. Underlined Text

Anything that appears within **<u>...</u>** element, is displayed with underline as shown below –

Example

```
<p>The word uses an <u>underlined</u> typeface.</p>
```

Result

The word uses an underlined typeface.

1.4.5. Strike Text

Anything that appears within **<strike>...</strike>** element is displayed with strikethrough, which is a thin line through the text as shown below

Example

```
<p>The word uses an <strike>strikethrough</strike> typeface.</p>
```

Result

The word uses an ~~strikethrough~~ typeface.

1.5.5. Monospaced Font

The content of a **<tt>...</tt>** element is written in monospaced font. Most of the fonts are known as variable-width fonts because different letters are of different widths (for example, the letter 'm' is wider than the letter 'i'). In a monospaced font, however, each letter has the same width.

Example

```
<p>Hello <tt>Write Your First Paragraph in monospaced font.</tt></p>
```

Result

```
Hello Write Your First Paragraph in monospaced font.
```

1.5.6. Superscript Text

If you put the content within ^{.....} element, is shown in superscript; means it is displayed half a character's height above the other characters.

Example

```
<p>The word uses RGUKT<sup>superscript</sup> typeface</p>
```

Result

The word uses RGUKT^{superscript} typeface

1.5.7. Subscript Text

If you put the content within _{.....} element, is shown in subscript ; means it is displayed half a character's height below the other characters.

Example

```
<p>The word uses as a <sub>subscript</sub> typeface</p>
```

Result

The word uses as a $_{\text{subscript}}$ typeface

1.5.8. Deleted Text

Anything that appears within **~~...~~** element is displayed as deleted text.

Example

```
<p>Hello <del>Delete your first paragraph.</del></p>
```

Result

Hello ~~Delete your first paragraph.~~

1.5.9. Inserted Text

Anything that appears within <**ins**>..... </**ins**> is displayed as inserted text.

Example

```
<p>Hello <ins>Write another paragraph.</ins></p>
```

Result

Hello Write another paragraph.

1.5.10. Larger Text

The content of the **<big>...</big>** element is displayed one font size larger than the rest of the text surrounding it as shown below

Example

```
<p>The following word uses a <big>big</big> typeface.</p>
```

Result

The following word uses a big typeface.

1.5.11. Smaller Text

The content of the **<small>. </small>** element is displayed one font size smaller than the rest of the text surrounding it as shown below

Example

```
<p>The following word uses a <small>small</small> typeface.</p>
```

Result

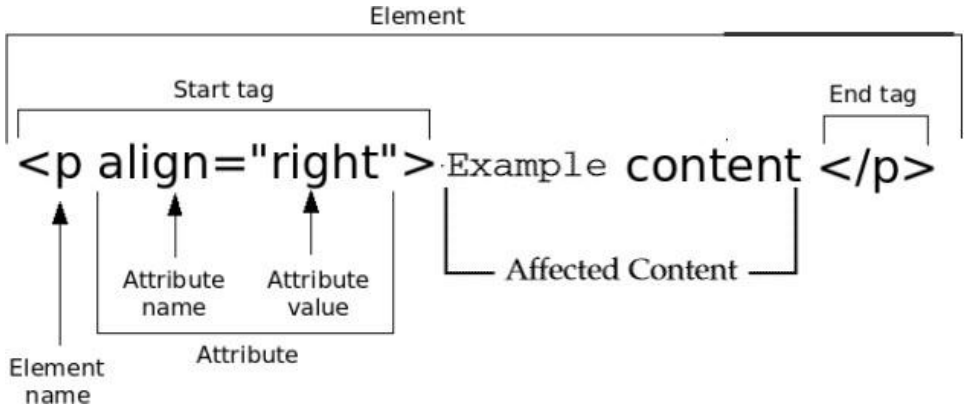
The following word uses a small typeface.

1.6. HTML - Attributes

We have seen few HTML tags and their usage like heading tags **<h1>, <h2>,** paragraph tag **<p>** and other tags. We used them so far in their simplest form, but most of the HTML tags can also have attributes, which are extra bits of information.

- HTML attributes are used to provide some more information about HTML elements.
- HTML Attributes are always specified in the start tag.
- HTML Attributes consist on name/value pairs like: i.e. name="value" and separated by an equals (=) sign.
- Attribute values always be enclosed in double/single quotes.
- You can add multiple attributes in one HTML element, but need to give space between two attributes.

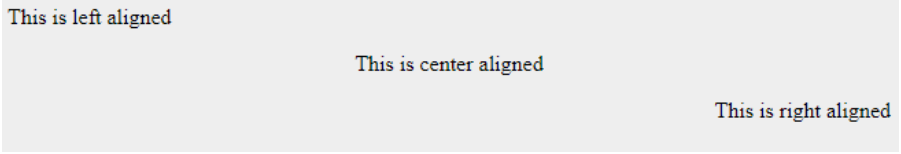
HTML Tag with Attributes and content is called **HTML Element**



Example

```
<!DOCTYPE html>
<html>

    <head>
        <title>Align Attribute  Example</title>
    </head>

    <body>
        <p align = "left">This is left aligned</p>
        <p align = "center">This is center aligned</p>
        <p align = "right">This is right aligned</p>
    </body>

</html>

```

Result:



Reference URL for Attributes

https://developer.mozilla.org/en-US/docs/Web/HTML/Attributes

1.6.1. Global Attributes

Many attributes are in HTML elements, some are common attributes and others can only be used on certain tags. Some of the more common attributes are:

Tag	Description
id	Often used with CSS to style a specific element. The value of this attribute must be unique.
class	Often used with CSS to style elements with common properties
style	CSS code specifies inline the HTML element is presented.
title	Text to be displayed in a tooltip when hovering over the element.

Example

```
<html>
<head>
  <title>HTML Attributes Example</title>
  <style type="text/css">
    #second {
      color:blue;
    }
    .abc{
    color:green;
    }
  </style>
</head>
<body>
  <p style="background-color:red;">Style Attributes</p>
  <p id="second">ID Attributes</p>
  <p class="abc">Class Attributes</p>
  <p title="Cascading Style Sheet">Title Attribute for CSS</p>
</body>
</html>
```

Result:

Style Attributes

ID Attributes

Class Attributes

Title Attribute for CSS

Cascading Style Sheet

1.6.2. Generic Attributes

Here's a table of some other attributes that are readily usable with many of the HTML tags.

Attribute Name	Value	Elements	Description	Example
Align	left, right, center	<hr> <img>	Specifies the horizontal alignment of the element.	RGUKT
background	Background Image of the document	<body>, <table>, <tr>	Specifies the URL of an image file.	<body background="img/bg.png">

bgcolor	numeric, hexidecimal, RGB values	<body>, <table>, <tbody>, <tfoot>, <td>, <th>, <tr>	Background color of the element.	<body bgcolor="blue">
border	Integer values	<img>, <object>, <table>	The border width.	<table border="1"> </table>
Color	numeric, hexidecimal, RGB values	, <hr>	This attribute sets the text color using either a named color or a color specified in the hexadecimal #RRGGBB format.	Color Attribute
colspan	Integer values	<td>, <th>	The colspan attribute defines the number of columns a cell should span.	<td colspan=”3”></td>
height	Integer values	<canvas>, <embed>, <iframe>, <img>, <input>, <object>,	Specifies the height of elements listed here. For all other elements, use the CSS height property..	
		<video>		
href	URL	<a>, <area>, <base>, <link>	The URL of a linked resource.	Text
Src	URL	<img>, <script>, <video>	The URL of the embeddable content.	
target	_blank, _self, _parent, _top	<a>, <area>, <base>, <form>	Specifies where to open a linked document.	<a href=”#” target=”_blank”>
Width	value	<canvas>, <embed>, <iframe>, <img>, <input>, <object>, <video>	For the elements listed here, this establishes the element's width..	

1.7. HTML – Backgrounds

By default, your webpage background is white in colour. You may not like it, but no worries. HTML provides you following two good ways to decorate your webpage background.

- HTML Background with Colours
- HTML Background with Images

Now let's see both the approaches one by one using appropriate examples.

1.7.1. Html Background with Colours

The bgcolour attribute is used to control the background of an HTML element, specifically page body and table backgrounds.

Following is the syntax to use bgcolour attribute with any HTML tag.

```
<tagname bgcolor = "color_value"...>
```

This colour_value can be given in any of the following formats −

```
<!-- Format 1 - Use color name -->
<table bgcolor = "lime" >

<!-- Format 2 - Use hex value -->
<table bgcolor = "#f1f1f1" >

<!-- Format 3 - Use color value in RGB terms -->
<table bgcolor = "rgb(0,0,120)" >
```

Example:

Here are the examples to set background of an HTML tag –

```
<!DOCTYPE html>
<html>

   <head>
      <title>HTML Background Colors</title>
   </head>

   <body>
      <!-- Format 1 - Use color name -->
      <table bgcolor = "yellow" width = "100%">
         <tr>
            <td>
               This background is yellow
            </td>
         </tr>
      </table>

      <!-- Format 2 - Use hex value -->
      <table bgcolor = "#6666FF" width = "100%">
         <tr>
            <td>
               This background is sky blue
            </td>
         </tr>
      </table>

      <!-- Format 3 - Use color value in RGB terms -->
      <table bgcolor = "rgb(255,0,255)" width = "100%">
         <tr>
            <td>
               This background is green
            </td>
         </tr>
      </table>
   </body>

</html>
```

Result:

This background is yellow

This background is sky blue

This background is green

1.7.2. Html Background with Images

The **background** attribute can also be used to control the background of an HTML element, specifically page body and table backgrounds. You can specify an image to set background of your HTML page or table.

> **Note** − The *background* attribute deprecated in HTML5. Do not use this attribute.

Following is the syntax to use background attribute with any HTML tag.

> **Note** − The *background* attribute is deprecated and it is recommended to use Style Sheet for background setting.

```
<tagname background = "Image URL"...>
```

The most frequently used image formats are JPEG, GIF and PNG images.

Example:

Here are the examples to set background images of a table.

```
<!DOCTYPE html>
<html>

   <head>
      <title>HTML Background Images</title>
   </head>

   <body>
      <!-- Set table background -->
      <table background = "/images/html.gif" width = "100%" height = "100">
         <tr><td>
            This background is filled up with HTML image.
         </td></tr>
      </table>
   </body>

</html>
```

Result:

1.8. HTML – Colours

Colours are very important to give a good look and feel to your website. You can specify colours on page level using <body> tag or you can set colours for individual tags using **bgcolour** attribute.
The <body> tag has following attributes which can be used to set different colours −

- **bgcolour** − sets a colour for the background of the page.
- **text** − sets a colour for the body text.
- **alink** − sets a colour for active links or selected links.
- **link** − sets a colour for linked text.
- **vlink** − sets a colour for *visited links* − that is, for linked text that you have already clicked on.

1.8.1. HTML Colour Coding Methods

There are following three different methods to set colours in your web page −

- **Colour names** − You can specify colour names directly like green, blue or red.
- **Hex codes** − A six-digit code representing the amount of red, green, and blue that makes up the colour.
- **Colour decimal or percentage values** − This value is specified using the rgb() property.

Now we will see these colouring schemes one by one.

1.8.2. HTML Colours - Colour Names

You can specify direct a colour name to set text or background colour. W3C has listed 16 basic colour names that will validate with an HTML validator but there are over 200 different colour names supported by major browsers.
Note − Check a complete list of HTML Colour Name.

Standard 16 Colours

Here is the list of Standard 16 Colours names and it is recommended to use them.

	Black		Gray		Silver		White
	Yellow		Lime		Aqua		Fuchsia
	Red		Green		Blue		Purple
	Maroon		Olive		Navy		Teal

Example:

Here are the examples to set background of an HTML tag by color name –

```
<!DOCTYPE html>
<html>

   <head>
      <title>HTML Colors by Name</title>
   </head>

   <body text = "blue" bgcolor = "green">
      <p>Use different color names for for body and table and see the result
         .</p>

      <table bgcolor = "black">
         <tr>
            <td>
               <font color = "white">This text will appear white on black
                  background.</font>
            </td>
         </tr>
      </table>
   </body>

</html>
```

1.8.3. HTML Colours - Hex Codes

A hexadecimal is a 6 digit representation of a colour. The first two digits (RR) represent a red value, the next two are a green value (GG), and the last are the blue value (BB).

A hexadecimal value can be taken from any graphics software like Adobe Photoshop, Paintshop Pro or MS Paint.

Each hexadecimal code will be preceded by a pound or hash sign #. Following is a list of few colours using hexadecimal notation.

Color	Color HEX
	#000000
	#FF0000
	#00FF00
	#0000FF
	#FFFF00
	#00FFFF
	#FF00FF
	#C0C0C0
	#FFFFFF

Example:

```
<!DOCTYPE html>
<html>

   <head>
      <title>HTML Colors by Hex</title>
   </head>

   <body text = "#0000FF" bgcolor = "#00FF00">
      <p>Use different color hexa for for body and table and see the result.</p
         >

      <table bgcolor = "#000000">
         <tr>
            <td>
               <font color = "#FFFFFF">This text will appear white on black
                  background.</font>
            </td>
         </tr>
      </table>
   </body>

</html>
```

1.8.4. HTML Colours - RGB Values

This colour value is specified using the **rgb( )** property. This property takes three values, one each for red, green, and blue. The value can be an integer between 0 and 255 or a percentage.

> **Note** – All the browsers does not support rgb() property of color so it is recommended not to use it.

Following is a list to show few colors using RGB values.

Color	Color RGB
	rgb(0,0,0)
	rgb(255,0,0)
	rgb(0,255,0)
	rgb(0,0,255)
	rgb(255,255,0)
	rgb(0,255,255)
	rgb(255,0,255)
	rgb(192,192,192)
	rgb(255,255,255)

Example

Here are the examples to set background of an HTML tag by colour code using rgb() values

```
<!DOCTYPE html>
<html>

   <head>
      <title>HTML Colors by RGB code</title>
   </head>

   <body text = "rgb(0,0,255)" bgcolor = "rgb(0,255,0)">
      <p>Use different color code for for body and table and see the result.</p
          >

      <table bgcolor = "rgb(0,0,0)">
         <tr>
            <td>
               <font color = "rgb(255,255,255)">This text will appear white on
                   black background.</font>
            </td>
         </tr>
      </table>
   </body>

</html>
```

1.9. HTML - Text Links

A webpage can contain various links that take you directly to other pages and even specific parts of a given page. These links are known as hyperlinks.

Hyperlinks allow visitors to navigate between Web sites by clicking on words, phrases, and images. Thus you can create hyperlinks using text or images available on a webpage.

1.9.1. Linking Documents

A link is specified using HTML tag <a>. This tag is called **anchor tag** and anything between the opening <a> tag and the closing tag becomes part of the link and a user can click that part to reach

```
<a href = "Document URL" ... attributes-list>Link Text</a>
```

to the linked document. Following is the simple syntax to use <a> tag.

Example

Let's try following example which links http://www.rguktrkv.ac.in at your page −

```
<!doctype html>
<html lang="en">
 <head>
  <title>Hyperlink Example</title>
 </head>
 <body>
    <p>Click following link to visit RKValley website</p>
    <a href="http://www.rguktrkv.ac.in/">RKValley Website</a>
 </body>
</html>
```

1.9.2. The target Attribute

We have used **target** attribute in our previous example. This attribute is used to specify the location where linked document is opened. Following are the possible options −

Sr.No	Option & Description
1	**_blank** Opens the linked document in a new window or tab.
2	**_self** Opens the linked document in the same frame.

Example:

Try following example to understand basic difference in few options given for target attribute

```
<!DOCTYPE html>
<html>

   <head>
      <title>Hyperlink Example</title>
      <base href = "https://www.tutorialspoint.com/">
   </head>

   <body>
      <p>Click any of the following links</p>
      <a href = "/html/index.htm" target = "_blank">Opens in New</a> |
      <a href = "/html/index.htm" target = "_self">Opens in Self</a> |
   </body>

</html>
```

1.9.3. Linking to a Page Section

You can create a link to a particular section of a given webpage by using **name** attribute. This is a two step process.

> **Note** − The *name* attribute deprecated in HTML5. Do not use this attribute. Use *id* and *title* attribute instead.

First create a link to the place where you want to reach with-in a webpage and name it using <a...> tag as follows −

```
<h1>HTML Text Links <a name = "top"></a></h1>
```

Second step is to create a hyperlink to link the document and place where you want to reach −

```
<a href = "/html/html_text_links.htm#top">Go to the Top</a>
```

This will produce following link, where you can click on the link generated **Go to the Top** to reach to the top of the HTML Text Link tutorial.

```
Go to the Top
```

1.10. HTML - Embed Multimedia

Sometimes you need to add music or video into your web page. The easiest way to add video or sound to your web site is to include the special HTML tag called <embed>. This tag causes the browser itself to include controls for the multimedia automatically provided browser supports <embed> tag and given media type.

You can also include a **<noembed>** tag for the browsers which don't recognize the<embed> tag. You could, for example, use <embed> to display a movie of your choice, and **<noembed>** to display a single JPG image if browser does not support <embed> tag.

Example:

Here is a simple example to play an embedded midi file −

```
<!DOCTYPE html>
<html>

   <head>
      <title>HTML embed Tag</title>
   </head>

   <body>
      <embed src = "/html/yourfile.mid" width = "100%" height = "60" >
         <noembed><img src = "yourimage.gif" alt = "Alternative Media" ></noembed>
      </embed>
   </body>

</html>
```

The <embed> Tag Attributes
Following is the list of important attributes which can be used with <embed> tag.

1.10.1. Supported Video Types

You can use various media types like Flash movies (.swf), AVI's (.avi), and MOV's (.mov) file types inside embed tag.

- **.swf files** − are the file types created by Macromedia's Flash program.
- **.wmv files** − are Microsoft's Window's Media Video file types.
- **.mov files** − are Apple's Quick Time Movie format.
- **.mpeg files** − are movie files created by the Moving Pictures Expert Group.

Example:

```
<!DOCTYPE html>
<html>

   <head>
      <title>HTML embed Tag</title>
   </head>

   <body>
      <embed src = "/html/yourfile.swf" width = "200" height = "200" >
         <noembed><img src = "yourimage.gif" alt = "Alternative Media" ></noembed>
      </embed>
   </body>

</html>
```

Objective Questions:

1. **HTML is stand for _____**
   a) **Hyper Text Markup Language**
   b) Holistick Technical Method Library
   c) Hyper Tax Makes Line
   d) None of the above
2. **ALL HTML tags are enclosed in what?**
   a) # and #
   b) ? and !
   c) < and >
   d) { and }
3. **To create HTML page, you need _____**
   a) Web browser
   b) text editor
   c) Both [A] and [B]
   d) None of the above
4. **<a> and are the tags used for _____**
   a) Adding image
   b) Aligning text
   c) Audio-voiced text
   d) **Adding links to your page**

5. To add a plain color background to your web page, use which of the following?

a) <body bgcolor= "36,24,35">
b) <body color= "# FF000">
c) <body bgcolor= "# FF000">
d) All of the above

6. HTML is stand for _____

a**) Hyper Text Markup Language**
b) Holistick Technical Method Library
c) Hyper Tax Makes Line
d) None of the above

7. HTML is a subset of _____

a) SGMD
b**) SGML**
c) SGMH
d) None of the above

8. ALL HTML tags are enclosed in what?

a) # and #
b) ? and !
c) < and >
d) { and }

9. To create HTML page, you need _____

a) Web browser
b) text editor
c) Both [A] and [B]
d) None of the above

10. <a> and are the tags used for _____

a) Adding image
b) Aligning text
c) Audio-voiced text
d) Adding links to your page

11. To add a plain color background to your web page, use which of the following?

a) <body bgcolor= "36,24,35">
b) <body color= "# FF000">
c) <body bgcolor= "# FF000">
d) All of the above

12. The BODY tag is usually used after _____

a) HTML tag
b) EM tag
c) TITLE tag
d) HEAD tag

13. Choose the correct HTML tag to make a text italic

a) <i>
b) <italic>
c) <it>
d) <il>

**14. What does the
 tag add to your webpage?**

a) Long break
b) Paragraph break
c) Line break
d) None of the above

15. Adding a border to your image helps the visitor to recognize it as what?

a) A frame
b) A link
c) A picture
d) None of the above

16. Which tag tells the browser where the page starts and stops?

a) <html>
b) <body>
c) <head>
d) <title>

17. Which program do you need to write HTML?

a) A graphics program
b) Any text editor
c) HTML -development suite 4
d) All of the above

18. In HTML, tags that include both on and off tag are called

a) comment tag
b) document tag
c) container tag
d) None of the above

19. What is the correct HTML for creating a hyperlink?

a) <a>https://w.w.w.gkseries.com
b) Gkseries.com
c) Gkseries.com
d) Gkseries.com

20. All normal webpages consists of _____

a) Top and bottom
b) Body and frameset
c) Head and body
d) None of the above

Problem set:

1. What is a HTML, explain it in briefly with example creating web page
2. Write short note on Attribute? Explain different kind of attributes with example source code and webpage
3. What are the formatting in HTML, explain them
4. How many types of heading does an HTML contain? Explain them
5. How to create a hyperlink in HTML? Explain with example code
6. What is the deference between HTML elements and tags?
7. How to make a picture of background image of a webpage?
8. Write HTML code that displays three hyperlinks to different websites. The websites should open in a new window when the user clicks on the hyperlinks

Unsolved Problem set:

1. Write the HTML code for the following text.

Basic protective measures against the new coronavirus

Stay aware of the latest information on the COVID-19 outbreak, available on the WHO website and through your national and local public health authority. COVID-19 is still affecting mostly people in China with some outbreaks in other countries. Most people who become infected experience mild illness and recover, but it can be more severe for others. Take care of your health and protect others by doing the following:

Wash your hands frequently

Regularly and thoroughly clean your hands with an alcohol-based hand rub or wash them with soap and water.

Why? Washing your hands with soap and water or using alcohol-based hand rub kills viruses that may be on your hands.

Maintain social distancing

Maintain at least 1 metre (3 feet) distance between yourself and anyone who is coughing or sneezing.

Why? When someone coughs or sneezes they spray small liquid droplets from their nose or mouth which may contain virus. If you are too close, you can breathe in the droplets, including the COVID-19 virus if the person coughing has the disease.

Avoid touching eyes, nose and mouth

Why? Hands touch many surfaces and can pick up viruses. Once contaminated, hands can transfer the virus to your eyes, nose or mouth. From there, the virus can enter your body and can make you sick.

Practice respiratory hygiene

Make sure you, and the people around you, follow good respiratory hygiene. This means covering your mouth and nose with your bent elbow or tissue when you cough or sneeze. Then dispose of the used tissue immediately.

Why? Droplets spread virus. By following good respiratory hygiene you protect the people around you from viruses such as cold, flu and COVID-19.

If you have fever, cough and difficulty breathing, seek medical care early

Stay home if you feel unwell. If you have a fever, cough and difficulty breathing, seek medical attention and call in advance. Follow the directions of your local health authority.

Why? National and local authorities will have the most up to date information on the situation in your area. Calling in advance will allow your health care provider to quickly direct you to the right health facility. This will also protect you and help prevent spread of viruses and other infections.

2. Design your own website with designing background color, inserting images and font format styles with headings.

Unit-2:: Lists
Module-1

2.1. Introduction

HTML offers web authors three ways for specifying lists of information. All lists must contain one or more list elements.

Lists may contain:

- ✓ - An unordered list. This will list items using plain bullets.
- ✓ - An ordered list. This will use different schemes of numbers to list your items.
- ✓ <dl> - A definition list. This arranges your items in the same way as they are arranged in a dictionary.

2.1.1. HTML Unordered Lists

An unordered list is a collection of related items that have no special order or sequence. This list is created by using HTML <ul> tag. Each item in the list is marked with a bullet.

```
<!DOCTYPE html>
<html>

   <head>
      <title>HTML Unordered List</title>
   </head>

   <body>
      <ul>
         <li>RGUKT RKV</li>
         <li>RGUKT NUZ</li>
         <li>RGUKT Srikakulam</li>
         <li>RGUKT Ongole</li>
      </ul>
   </body>

</html>
```

OUTPUT:

- RGUKT RKV
- RGUKT NUZ
- RGUKT Srikakulam
- RGUKT Ongole

The type Attribute

You can use type attribute for <ul> tag to specify the type of bullet you like. By default it is a disc. Following are the possible options:

- *<ul type="square">*
- *<ul type="disc">*
- *<ul type="circle">*

Let's see how <ul type="square"> is used

```
<!DOCTYPE html>
<html>

   <head>
      <title>HTML Unordered List</title>
   </head>

   <body>
      <ul type= square>
         <li>RGUKT RKV</li>
         <li>RGUKT NUZ</li>
         <li>RGUKT Srikakulam</li>
         <li>RGUKT Ongole</li>
      </ul>
   </body>

</html>
```

OUTPUT:

- RGUKTRKV
- RGUKT NUZ
- RGUKT Srikakulam
- RGUKT Ongole

2.1.2. HTML Ordered Lists

If you are required to put your items in a numbered list instead of bulleted then HTML ordered list will be used. This list is created by using <ol> tag. The numbering starts at '1' and is incremented by 1 for each successive ordered list element tagged with <li>.

```
<!DOCTYPE html>
<html>

   <head>
      <title>HTML Unordered List</title>
   </head>

   <body>
      <ol>
         <li>RGUKT RKV</li>
         <li>RGUKT NUZ</li>
         <li>RGUKT Srikakulam</li>
         <li>RGUKT Ongole</li>
      </ol>
   </body>

</html>

```

OUTPUT:

1. RGUKTRKV
2. RGUKT NUZ
3. RGUKT Srikakulam
4. RGUKT Ongole

The type Attribute

You can use type attribute for <ol> tag to specify the type of numbering you like. By default it is a number. Following are the possible options:

- *<ol type="1"> - Default-Case Numerals.*
- *<ol type="I"> - Upper-Case Numerals.*
- *<ol type="i"> - Lower-Case Numerals.*
- *<ol type="a"> - Lower-Case Letters.*
- *<ol type="A"> - Upper-Case Letters.*

Let's see how <ol type="A"> is used

```
<!DOCTYPE html>
<html>

   <head>
      <title>HTML Unordered List</title>
   </head>

   <body>
      <ol type="A">
         <li>RGUKT RKV</li>
         <li>RGUKT NUZ</li>
         <li>RGUKT Srikakulam</li>
         <li>RGUKT Ongole</li>
      </ol>
   </body>

</html>
```

OUTPUT:

A. RGUKTRKV
B. RGUKT NUZ
C. RGUKT Srikakulam
D. RGUKT Ongole

Now let's see how <ol type="I"> is used

```
<!DOCTYPE html>
<html>

   <head>
      <title>HTML Unordered List</title>
   </head>

   <body>
      <ol type="I">
         <li>RGUKT RKV</li>
         <li>RGUKT NUZ</li>
         <li>RGUKT Srikakulam</li>
         <li>RGUKT Ongole</li>
      </ol>
   </body>

</html>
```

OUTPUT:

A. RGUKTRKV
B. RGUKT NUZ
C. RGUKT Srikakulam
D. RGUKT Ongole

The start Attribute

You can use start attribute for <ol> tag to specify the starting point of numbering you need.Following are the possible options:

- *<ol type="1" start="4"> - Numerals starts with 4.*
- *<ol type="I" start="4"> - Numerals starts with IV.*
- *<ol type="i" start="4"> - Numerals starts with iv.*
- *<ol type="a" start="4"> - Letters starts with d.*
- *<ol type="A" start="4"> - Letters starts with D.*

Let's see how <ol type="i" start="4" > is used

```
<!DOCTYPE html>
<html>

    <head>
        <title>HTML Unordered List</title>
    </head>

    <body>
        <ol type="i" start="4">
            <li>RGUKT RKV</li>
            <li>RGUKT NUZ</li>
            <li>RGUKT Srikakulam</li>
            <li>RGUKT Ongole</li>
        </ol>
    </body>

</html>
```

OUTPUT:

A. RGUKTRKV
B. RGUKT NUZ
C. RGUKT Srikakulam
D. RGUKT Ongole

2.1.3. HTML Definition Lists

HTML and XHTML support a list style which is called definition lists where entries are listed like in a dictionary or encyclopedia. The definition list is the ideal way to present a glossary, list of terms, or other name/value list.

Definition List makes use of following three tags.

- *<dl> - Defines the start of the list*
- *<dt> - A term*
- *<dd> - Term definition*
- *</dl> - Defines the end of the list*

```
<!DOCTYPE html>
<html>

   <head>
      <title>HTML Definition List</title>
   </head>

   <body>
      <dl>
         <dt><b>HTML</b></dt>
         <dd>This stands for Hyper Text Markup Language</dd>
         <dt><b>HTTP</b></dt>
         <dd>This stands for Hyper Text Transfer Protocol</dd>
      </dl>
   </body>

</html>
```

OUTPUT:

HTML
This stands for Hyper Text Markup Language
HTTP
This stands for Hyper Text Transfer Protocol

Multiple choice Questions:

1. An ordered list in HTML document starts with a
   A. tag
   B. tag
   C. tag
2. An unordered list in HTML document starts with a
   A. tag
   B. tag
   C. <lu> tag.
   D. None
3. For arranging your list items in same way as they were arranged in dictionary which tag you will use?
   A.
   B.
   C.
   D. <dl>
4. From which tag descriptive list starts ?
   A. <LL>
   B. <DD>
   C. <DL>
   D. <DS>

5. The tag used to create a new list item and also include a hyperlink is
   A. <LI>
   B. <DL>
   C. <DD>
   D. <UL>
6. HTML <dl> tag defines the
   A. Unordered list
   B. Ordered list
   C. Description list
   D. Descriptive list

UNSOLVED PROBLEMS

Unordered lists

1. Unordered (bulleted) lists are used when a set of items can be placed in any order. An example is a shopping list:
   Milk
   Bread
   Butter
   Coffee beans
2. Although the items are all part of one list, you could put the items in any order and the list would still make sense:
   Milk
   Bread
   Butter
   Coffee Beans

Ordered lists

1. Ordered (numbered) lists are used to display a list of items that should be in a specific order. An example would be cooking instructions:
   1. Gather ingredients
   2. Mix ingredients together
   3. Place ingredients in a baking dish
   4. Bake in oven for an hour
   5. Remove from oven
   6. Allow to stand for ten minutes
   7. Serve
2. If the list items were moved around into a different order, the information would no longermake sense:
   1. Gather ingredients
   2. Bake in oven for an hour
   3. Serve
   4. Remove from oven
   5. Place ingredients in a baking dish
   6. Allow to stand for ten minutes
   7. Mix ingredients together

Description lists

Description list(previously called definition lists, but renamed in HTML5) associated specific names and values within a list.

Examples might be items in an ingredient list and their descriptions, article authors and brief bios,or competitions winners and the years in which they won. You can have as many names- value groups as you like, but there must be at least one name and at least one value in each pair.

Description lists are flexible: you can associate more than one value with a single name, or vice versa. For example, the term “coffee” can have several meanings, and you could show them oneafter the other.

```
coffee

  a beverage made from roasted, ground coffee beans
  a cup of coffee
  a social gathering at which coffee is consumed
  a medium to dark brown colour
```

Or, you can associate more than one name with the same value. This is useful to show variations ofa term, all of which have the same meaning:

```
soda
pop
fizzy drink
cola

  a sweet, carbonated beverage
```

3. Create the below mention page using the list?

Module - 2

2.2. HTML TABLES

The HTML tables allow web authors to arrange data like text, images, links, other tables, etc. into rows and columns of cells. The HTML tables are created using the <table> tag in which <tr> tag is used to create table rows and <td> tag is used to create data cells.

```
<!DOCTYPE html>
<html>

   <head>
      <title>HTML Tables</title>
   </head>

   <body>
      <table border = "1">
         <tr>
            <td>Row 1, Column 1</td>
            <td>Row 1, Column 2</td>
         </tr>

         <tr>
            <td>Row 2, Column 1</td>
            <td>Row 2, Column 2</td>
         </tr>
      </table>

   </body>
</html>
```

OUTPUT:

Row 1, Column 1	Row 1, Column 2
Row 2, Column 1	Row 2, Column 2

Here border is an attribute of <table> tag and it is used to Put a border across all the cells. If you do not need a border then you can use border="0".

2.2.1. Table Heading:

Table heading can be defined using <th> tag. This tag will be put to replace <td> tag, which is used to represent actual data cell. Normally you will put your top row as table heading as shown below, otherwise you can use <th> element in any row

```
<!DOCTYPE html>
<html>

   <head>
      <title>HTML Table Header</title>
   </head>

   <body>
      <table border = "1">
         <tr>
            <th>Name</th>
            <th>Salary</th>
         </tr>
         <tr>
            <td>Ramesh Raman</td>
            <td>5000</td>
         </tr>

         <tr>
            <td>Shabbir Hussein</td>
            <td>7000</td>
         </tr>
      </table>
   </body>

</html>
```

Output:

Name	Salary
Ramesh Raman	5000
Shabbir Hussein	7000

2.2.2. Cellpadding and Cellspacing Attributes

There are two attributes called cellpadding and cellspacing which you will use to adjust the white space in your table cells. The cellspacing attribute defines the width of the border, while cellpadding represents the distance between cell borders and the content within a cell.

```
<!DOCTYPE html>
<html>

   <head>
      <title>HTML Table Cellpadding</title>
   </head>

   <body>
      <table border = "1" cellpadding = "5" cellspacing = "5">
         <tr>
            <th>Name</th>
            <th>Salary</th>
         </tr>
         <tr>
            <td>Ramesh Raman</td>
            <td>5000</td>
         </tr>
         <tr>
            <td>Shabbir Hussein</td>
            <td>7000</td>
         </tr>
      </table>
   </body>

</html>
```

OUTPUT:

Name	Salary
Ramesh Raman	5000
Shabbir Hussein	7000

2.2.3. Colspan and Rowspan Attributes:

You will use colspan attribute if you want to merge two or more columns into a single column. Similarway you will use rowspan if you want to merge two or more rows.

```
<!DOCTYPE html>
<html>

   <head>
      <title>HTML Table Colspan/Rowspan</title>
   </head>

   <body>
      <table border = "1">
         <tr>
            <th>Column 1</th>
            <th>Column 2</th>
            <th>Column 3</th>
         </tr>
         <tr>
            <td rowspan = "2">Row 1 Cell 1</td>
            <td>Row 1 Cell 2</td>
            <td>Row 1 Cell 3</td>
         </tr>
         <tr>
            <td>Row 2 Cell 2</td>
            <td>Row 2 Cell 3</td>
         </tr>
         <tr>
            <td colspan = "3">Row 3 Cell 1</td>
         </tr>
      </table>
   </body>

</html>
```

OUTPUT:

Column 1	Column 2	Column 3
Row 1 Cell 1	Row 1 Cell 2	Row 1 Cell 3
	Row 2 Cell 2	Row 2 Cell 3
Row 3 Cell 1		

2.2.4. Border color

The color of the lines inside and outside the table can also be changed using the “Border Color” attribute. It accepts the value as name of the color. If you omit this attribute, the color of the table border is set to its default grey.

```
<!DOCTYPE html>
<html>

   <head>
      <title>HTML Table Background</title>
   </head>

   <body>
      <table border = "1" bordercolor = "green" bgcolor = "yellow">
         <tr>
            <th>Column 1</th>
            <th>Column 2</th>
            <th>Column 3</th>
         </tr>
         <tr>
            <td rowspan = "2">Row 1 Cell 1</td>
            <td>Row 1 Cell 2</td>
            <td>Row 1 Cell 3</td>
         </tr>
         <tr>
            <td>Row 2 Cell 2</td>
            <td>Row 2 Cell 3</td>
         </tr>
         <tr>
            <td colspan = "3">Row 3 Cell 1</td>
         </tr>
      </table>
   </body>

</html>
```

Output:

Column 1	Column 2	Column 3
Row 1 Cell 1	Row 1 Cell 2	Row 1 Cell 3
	Row 2 Cell 2	Row 2 Cell 3
Row 3 Cell 1		

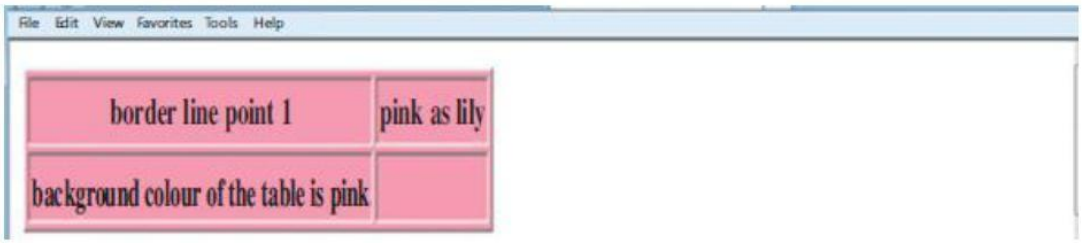
2.2.5. Bgcolor

The background color of a table can be set using the attribute bgcolor. This attribute takes the name of the color or hexadecimal number as value.

<TABLE border =1 bgcolor= pink>

Note: If you want to set individual cells of the table with different background colour, the attribute bgcolor can be used with either <TH> tag or <TD> tag.

OUTPUT:



2.2.6. Background

If you want to place an image or a picture at the background of the table, you can do so using the background attribute. This attribute takes the value as the address or the path of the picture. The picture may be a bitmap or a graphic image. In the following code, the image named "yelloww.jpg" is set as background to the entire table.

<TABLE border ="1" background="c:\yelloww.jpg">

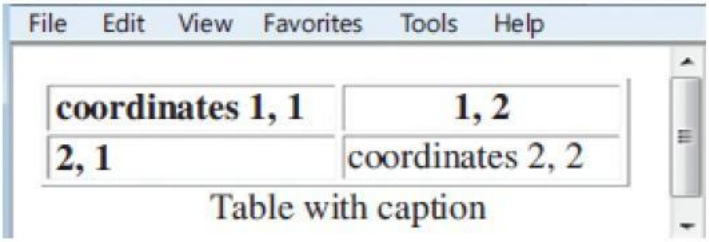
2.2.7. The CAPTION tag

The <caption> tag is used to provide a text to the table so as to explain the contents of the table. It is generally in bold, at center with respect to the table. However, the position of the caption can be on either the top or the bottom of the table using the 'align' attribute as shown below:

<TABLE BORDER = 1>

<CAPTION align=bottom>Table with caption</CAPTION>

OUTPUT:



Multiple Choice Questions

1. In HTML tables table row is defined by
   A. <th> tag
   B. <tr> tag
   C. <td> tag
   D. <row> tag
2. In HTML tables gap between two cells of same tables are known as
   A. Cell spacing
   B. Cell difference
   C. Cell padding
   D. All of above
3. In HTML tables table header is defined by
   A. <th> tag
   B. <tr> tag
   C. <td> tag
   D. <t head> tag
4. If you want to merge two or more rows in a table which attribute you can use?

A. Rowmerge
B. Rowspan
C. Colmerge
D. Colspan

5. Title element defines title of document at
   A. Header
   B. Web browser
   C. Middle of the Document
   D. Footer
6. In HTML tables space between cell content and cell border is called
   A. Cell spacing
   B. Cell difference
   C. Cell padding
   D. All of above

UNSOLVED PROBLEMS

1. Creating a Table

The objective of this Activity is to create a timetable for CSC5003 students to be displayed on a Webpage as shown below:

CSC503 timetable

	Monday	Tuesday	Wednesday	Thursday	Friday
6-7pm	look at website	free	Implementation	free	free
7-8pm	take some notes	free	Implementation	free	free

2. The <TR> tag is used to add rows. Each row is composed of several data cells. Their dimensions can be defined using width and height attributes: <TD width=25% height=20 bgcolor="darkred">Notice that the cell's color can also be defined. Try to create the table below before you look at the solution code under Discussion and Answers at the end of the chapter.

red cell	light blue cell

3. Reopen the file tab_ex1.html in your text editor and make the following amendments to <TABLE> and <tr> tags.

Note the <CENTER> tag centers the table horizontally and it also centers the text within each cell in therow.

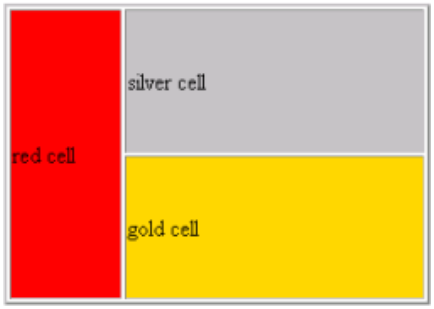
<TABLE style= "width: 80%" align = "center">

<tr align = "center">

4. Save this file as tab_ex2.html and view it in your browser. It should look as below

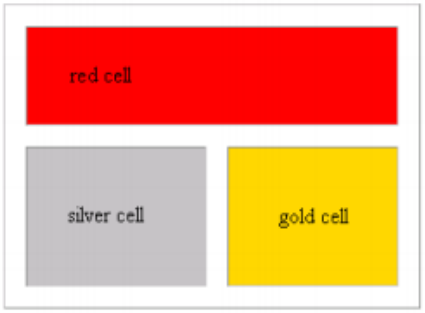
Using rowspan

5. This Activity introduces you to the attribute row span. The objective of this Activity is to createthe following table.



Using colspan, cellspacing and cellpadding

6. This Activity introduces you to the attributes colspan, cellspacing and cellpadding. The objectiveof this Activity is to create the following table



More cellspacing and cellpadding

7. The objective of this Activity is to create the table shown below.

Time Table

8. Write the necessary HTML code for your own study timetable. This should look similar to theone shown below. (Hint: use the colspan and rowspan attributes).

Ian's Timetable

		Morning			lunch	afternoon				evening		
		9-10 am	10-11 am	11-12 am	1-2 pm	2-3 pm	3-4 pm	4-5 pm	5-6 pm	6-7 pm	7-8 pm	8-9 pm
Work	Monday	Development Meeting				Client Meeting			commute	Free		
	Tuesday	in Office				in Office						
Lecture	Wednesday	CSC205				preparation						
	Thursday	MAM200				Tutorials for MAM200			Free			
Research	Friday	Research				Research						

9. Create the below mentioned page

Books Information

NAME	Available BOOKS	PRICE
Ramayanam	1000	75
Oxford Dictionary	900	500
Novels	500	200

10. Create the below mention timetable using tables in html page

COLLEGE TIME TABLE

	8:30-9:30	9:30-10:30	10:3-11:30	11:30-12:30	12:30-2:00	2:00-3:00	3:00-4:00	4:00-5:00
MONDAY	---	SUB1	SUB2	SUB3	LUNCH	SUB4	SUB5	counselling class
TUESDAY	SUB1	SUB2	SUB3	---		SUB4	SUB5	library
WEDNESDAY	SUB1	SUB2	SWA	---		lab		
THURSDAY	SUB1	SUB2	SUB3	---		SUB4	SUB5	library
FRIDAY	SUB1	SUB2	SUB3	---		SUB4	SUB5	library
SATURDAY	SUB1	seminar				SUB4	SUB5	library

Unit-3:: Forms
Module – 1

3.1 What is HTML Form?

HTML Forms are required, when you want to collect some data from the site visitor. For example, during user registration you would like to collect information such as name, email address, credit card, etc.

A form will take input from the site visitor and then will post it to a back-end application such as CGI, ASP Script or PHP script etc. The back-end application will perform required processing on the passed data based on defined business logic inside the application.

There are various form elements available like text fields, textarea fields, drop-down menus, radio buttons, checkboxes, etc.

The HTML **<form>** tag is used to create an HTML form and it has following syntax –

```
<form action = "Script URL" method = "GET|POST">
  form elements like input, textarea etc.
</form>
```

The <form> tag is used to create an HTML form. Here's a simple example of a login form:

```
<!DOCTYPE html>
<html>
<body>

<form>
  Username:<br>
  <input type="text" name="username">
  <br>
  Email id:<br>
  <input type="text" name="email_id">
  <br><br>
  <input type="submit" value="Submit">
</form>

</body>
</html>
```

3.2 Input Element

This is the most commonly used element within HTML forms.

It allows you to specify various types of user input fields, depending on the type attribute. An input element can be of type text field, password field, checkbox, radio button, submit button, reset button, file select box, as well as several new input types introduced in HTML.

The most frequently used input types are described below.

3.2.1 Text Fields

Text fields are one-line areas that allow the user to input text.

Single-line text input controls are created using an <input> element, whose type attribute has a value of text. Here's an example of a single-line text input used to take username:

Example

```
<form>
    <label for="username">Username:</label>
    <input type="text" name="username" id="username">
</form>
```

The output of the above example will look something like this:

Username:

Attributes

Following is the list of attributes for <input> tag for creating text field.

Sr. No	Attribute & Description
1	**type** Indicates the type of input control and for text input control it will be set to **text**.
2	**name** Used to give a name to the control which is sent to the server to be recognized and get the value.
3	**value** This can be used to provide an initial value inside the control.
4	**size** Allows to specify the width of the text-input control in terms of characters.
5	**maxlength** Allows to specify the maximum number of characters a user can enter into the text box.

3.2.2 Password Field

Password fields are similar to text fields. The only difference is; characters in a password field are masked, i.e. they are shown as asterisks or dots. This is to prevent someone else from reading the password on the screen. This is also a single-line text input controls created using an <input> element whose type attribute has a value of password.

Here's an example of a single-line password input used to take user password:

Example

```
<form>
    <label for="user-pwd">Password:</label>
    <input type="password" name="user-password" id="user-pwd">
</form>
```

The output of the above example will look something like this:

Password:

Attributes

Following is the list of attributes for <input> tag for creating password field.

Sr.No	Attribute & Description
1	**type** Indicates the type of input control and for password input control it will be set to **password**.
2	**name** Used to give a name to the control which is sent to the server to be recognized and get the value.
3	**value** This can be used to provide an initial value inside the control.
4	**size** Allows to specify the width of the text-input control in terms of characters.
5	**maxlength** Allows to specify the maximum number of characters a user can enter into the text box.

3.3 Radio Buttons

Radio buttons are used to let the user select exactly one option from a pre-defined set of options. It is created using an <input> element whose type attribute has a value of radio.

Here's an example of radio buttons that can be used to collect user's gender information:

Example

```
<form>
    <input type="radio" name="gender" id="male">
    <label for="male">Male</label>
    <input type="radio" name="gender" id="female">
    <label for="female">Female</label>
</form>
```

The output of the above example will look something like this:

Attributes

Following is the list of attributes for radio button.

Sr.No	Attribute & Description
1	**type** Indicates the type of input control and for checkbox input control it will be set to radio.
2	**name** Used to give a name to the control which is sent to the server to be recognized and get the value.
3	**value** The value that will be used if the radio box is selected.
4	**checked** Set to *checked* if you want to select it by default.

3.4 Checkboxes

Checkboxes allows the user to select one or more option from a pre-defined set of options. It is created using an <input> element whose type attribute has a value of checkbox.

Here's an example of checkboxes that can be used to collect information about user's hobbies:

Example

```
<form>
    <input type="checkbox" name="sports" id="soccer">
    <label for="soccer">Soccer</label>
    <input type="checkbox" name="sports" id="cricket">
    <label for="cricket">Cricket</label>
    <input type="checkbox" name="sports" id="baseball">
    <label for="baseball">Baseball</label>
</form>
```

The output of the above example will look something like this:

Attributes

Following is the list of attributes for <checkbox> tag.

Sr.No	Attribute & Description
1	**type** Indicates the type of input control and for checkbox input control it will be set to **checkbox.**.
2	**name** Used to give a name to the control which is sent to the server to be recognized and get the value.
3	**value** The value that will be used if the checkbox is selected.
4	**checked** Set to *checked* if you want to select it by default.

Note: If you want to make a radio button or checkbox selected by default, you can add the attribute checked to the input element, like <input type="checkbox" checked>.

3.5 File Select box

The file fields allow a user to browse for a local file and send it as an attachment with the form data. Web browsers such as Google Chrome and Firefox render a file select input field with a Browse button that enables the user to navigate the local hard drive and select a file.

This is also created using an <input> element, whose type attribute value is set to file

Example

```
<form>
    <label for="file-select">Upload:</label>
    <input type="file" name="upload" id="file-select">
</form>
```

The output of the above example will look something like this:

Attributes

Following is the list of important attributes of file upload box −

Sr.No	Attribute & Description
1	**name** Used to give a name to the control which is sent to the server to be recognized and get the value.
2	**Accept** Specifies the types of files that the server accepts.

3.6 Textarea

Textarea is a multiple-line text input control that allows a user to enter more than one line of text. Multi-line text input controls are created using an <textarea> element.

Example

```
<form>
    <label for="address">Address:</label>
    <textarea rows="3" cols="30" name="address" id="address"></textarea>
</form>
```

The output of the above example will look something like this:

Attributes

Following is the list of attributes for <textarea> tag.

Sr.No	Attribute & Description
1	**name** Used to give a name to the control which is sent to the server to be recognized and get the value.
2	**rows** Indicates the number of rows of text area box.
3	**cols** Indicates the number of columns of text area box

3.7 Select Boxes

A select box is a dropdown list of options that allows user to select one or more option from a pull-down list of options. Select box is created using the <select> element and <option> element.

The <option> elements within the <select> element define each list item.

Example

```
<form>
    <label for="city">City:</label>
    <select name="city" id="city">
        <option value="sydney">Sydney</option>
        <option value="melbourne">Melbourne</option>
        <option value="cromwell">Cromwell</option>
    </select>
</form>
```

The output of the above example will look something like this:

City: Sydney

Attributes

Following is the list of important attributes of <select> tag −

Sr.No	Attribute & Description
1	**name** Used to give a name to the control which is sent to the server to be recognized and get the value.
2	**size** This can be used to present a scrolling list box.
3	**multiple** If set to "multiple" then allows a user to select multiple items from the menu.

Following is the list of important attributes of <option> tag −

Sr.No	Attribute & Description
1	**value** The value that will be used if an option in the select box box is selected.
2	**selected** Specifies that this option should be the initially selected value when the page loads.
3	**label** An alternative way of labeling options

3.8 Submit and Reset Buttons

A submit button is used to send the form data to a web server. When submit button is clicked the form data is sent to the file specified in the form's action attribute to process the submitted data.

A reset button resets all the forms control to default values. Try out the following example by typing your name in the text field, and click on submit button to see it in action.

Example

```
<form action="action.php" method="post">
    <label for="first-name">First Name:</label>
    <input type="text" name="first-name" id="first-name">
    <input type="submit" value="Submit">
    <input type="reset" value="Reset">
</form>
```

First Name: Submit Reset

3.8.1. Button Controls

There are various ways in HTML to create clickable buttons. You can also create a clickable button using <input>tag by setting its type attribute to **button**. The type attribute can take the following values −

Sr.No	Type & Description
1	**submit** This creates a button that automatically submits a form.
2	**reset** This creates a button that automatically resets form controls to their initial values.
3	**button** This creates a button that is used to trigger a client-side script when the user clicks that button.
4	**image** This creates a clickable button but we can use an image as background of the button.

3.9 Grouping Form Controls

You also group logically related controls and labels within a web form using the <legend> element. Grouping form controls into categories makes it easier for users to locate a control which makes the form more user-friendly. Let's try out the following example to see how it works:

Example

```
<form>
    <fieldset>
        <legend>Contact Details</legend>
        <label>Email Address: <input type="email" name="email"></label>
        <label>Phone Number: <input type="text" name="phone"></label>
    </fieldset>
</form>
```

3.10 Attributes in HTML Forms

3.10.1 The Action Attribute:

The action to be performed after the submission of the form is decided by the action attribute. Generally, the form data is sent to a webpage on the web server after the user clicks on the submit button.

Example:

```
<!DOCTYPE html>
<html>
<h3>Example of a Submit And Reset Button</h3>
<body>
    <form action="test.php" method="post" id="users">
        <label for="username">Username:</label>
        <input type="text" name="username" id="Username">
        <input type="submit" value="Submit">
        <input type="reset" value="Reset">
    </form>
</body>
</html>
```

If you click the submit button, the form data would be sent to a page called test.php .

Specifies the URL of the program or script on the web server that will be used for processing the information submitted via form

3.10.2 The Target Attribute in HTML Forms:

The Target attribute is used to specify whether the submitted result will open in the current window, a new tab or on a new frame. The default value used is “self” which results in the form submission in the same window. For making the form result open in a new browser tab, the value should be set to “blank”.

Specifies where to display the response that is received after submitting the form. Possible values are _blank, _self, _parent and _top.

```
<!DOCTYPE html>
<html>
<body>

<form action="/test.php" target="_blank">
  Username:<br>
  <input type="text" name="username">
  <br>
  Password:<br>
  <input type="password" name="password">
  <br><br>
  <input type="submit" value="Submit">
</form>

</body>
</html>
```

After clicking on the submit button, the result will open in a new browser tab.

3.10.3 Name Attribute in Html Forms:

The name attribute is required for each input field. If the name attribute is not specified in an input field then the data of that field would not be sent at all.

```
<!DOCTYPE html>
<html>
<body>

<form action="/test.php" target="_blank">
  Username:<br>
  <input type="text">
  <br>
  Password:<br>
  <input type="password" name="password">
  <br><br>
  <input type="submit" value="Submit">
</form>

</body>
</html>
```

In the above code, after clicking the submit button, the form data will be sent to a page called /test.php. The data sent would not include the Username input field data since the name attribute is omitted.

3.10.4 The Method Attribute:

It is used to specify the HTTP method used to send data while submitting the form. There are two kinds of HTTP Methods, which are GET and POST.

The GET Method –

```
<!DOCTYPE html>
<html>
<body>

<form action="/test.php" target="_blank" method="GET">
  Username:<br>
  <input type="text" name="username">
  <br>
  Password:<br>
  <input type="password" name="password">
  <br><br>
  <input type="submit" value="Submit">
</form>

</body>
</html>
```

In the GET method, after the submission of the form, the form values

will be visible in the address bar of the new browser tab.

The Post Method –

```
<!DOCTYPE html>
<html>
<body>

<form action="/test.php" target="_blank" method="post">
  Username:<br>
  <input type="text" name="username">
  <br>
  Password:<br>
  <input type="password" name="password">
  <br><br>
  <input type="submit" value="Submit">
</form>

</body>
</html>
```

In the post method, after the submission of the form, the form values will not be visible in the address bar of the new browser tab as it was visible in the GET method.

Specifies the HTTP method used for sending the data to the web server by the browser. The value can be either get (the default) and post.

HTML – Frames

Module – 2

HTML provides programmers for dividing a single browser display into multiple window sections, where each section holds the capability to load individual URLs. This concept of HTML providing multiple frames at one browser display is called frameset, and all the frame tags are used within the container tag <frameset>. So the entire separation of HTML pages is possible using the concept of frames. In his chapter, you will be learning about the frames and how they are used for creating multiple sections in a single browser display.

3.11Creating Frames

To use frames on a page we use <frameset> tag instead of <body> tag. The <frameset> tag defines, how to divide the window into frames. The rows attribute of <frameset> tag defines horizontal frames and cols attribute defines vertical frames. Each frame is indicated by <frame> tag and it defines which HTML document shall open into the frame.

Example

Following is the example to create three horizontal frames –

```
<!DOCTYPE html>
<html>

   <head>
      <title>HTML Frames</title>
   </head>

   <frameset rows = "10%,80%,10%">
      <frame name = "top" src = "/html/top_frame.htm" />
      <frame name = "main" src = "/html/main_frame.htm" />
      <frame name = "bottom" src = "/html/bottom_frame.htm" />

      <noframes>
         <body>Your browser does not support frames.</body>
      </noframes>

   </frameset>

</html>
```

Example

Let's put the above example as follows, here we replaced rows attribute by cols and changed their width. This will create all the three frames vertically –

```
<!DOCTYPE html>
<html>

   <head>
      <title>HTML Frames</title>
   </head>

   <frameset cols = "25%,50%,25%">
      <frame name = "left" src = "/html/top_frame.htm" />
      <frame name = "center" src = "/html/main_frame.htm" />
      <frame name = "right" src = "/html/bottom_frame.htm" />

      <noframes>
         <body>Your browser does not support frames.</body>
      </noframes>
   </frameset>

</html>
```

3.12 The <frameset> Tag Attributes

Following are important attributes of the <frameset> tag −

Sr. No	Attribute & Description
1	**cols** Specifies how many columns are contained in the frameset and the size of each column. You can specify the width of each column in one of the four ways − Absolute values in pixels. For example, to create three vertical frames, use *cols = "100, 500, 100"*. A percentage of the browser window. For example, to create three vertical frames, use *cols = "10%, 80%, 10%"*. Using a wildcard symbol. For example, to create three vertical frames, use *cols = "10%, *, 10%"*. In this case wildcard takes remainder of the window. As relative widths of the browser window. For example, to create three vertical frames, use *cols = "3*, 2*, 1*"*. This is an alternative to percentages. You can use relative widths of the browser window. Here the window is divided into sixths: the first column takes up half of the window, the second takes one third, and the third takes one sixth.
2	**rows** This attribute works just like the cols attribute and takes the same values, but it is used to specify the rows in the frameset. For example, to create two horizontal frames, use *rows = "10%, 90%"*. You can specify the height of each row in the same way as explained above for columns.
3	**border** This attribute specifies the width of the border of each frame in pixels. For example, border = "5". A value of zero means no border.
4	**frameborder** This attribute specifies whether a three-dimensional border should be displayed between frames. This attribute takes value either 1 (yes) or 0 (no). For example frameborder = "0" specifies no border.
5	**framespacing** This attribute specifies the amount of space between frames in a frameset. This can take any integer value. For example framespacing = "10" means there should be 10 pixels spacing between each frames.

3.14 The <frame> Tag Attributes

Following are the important attributes of <frame> tag −

Sr.No	Attribute & Description
1	**src** This attribute is used to give the file name that should be loaded in the frame. Its value can be any URL. For example, src = "/html/top_frame.htm" will load an HTML file available in html directory.
2	**name** This attribute allows you to give a name to a frame. It is used to indicate which frame a document should be loaded into. This is especially important when you want to create links in one frame that load pages into an another frame, in which case the second frame needs a name to identify itself as the target of the link.
3	**frameborder** This attribute specifies whether or not the borders of that frame are shown; it overrides the value given in the frameborder attribute on the <frameset> tag if one is given, and this can take values either 1 (yes) or 0 (no).
4	**marginwidth** This attribute allows you to specify the width of the space between the left and right of the frame's borders and the frame's content. The value is given in pixels. For example marginwidth = "10".
5	**marginheight** This attribute allows you to specify the height of the space between the top and bottom of the frame's

	borders and its contents. The value is given in pixels. For example marginheight = "10".
6	**noresize** By default, you can resize any frame by clicking and dragging on the borders of a frame. The noresize attribute prevents a user from being able to resize the frame. For example noresize = "noresize".
7	**scrolling** This attribute controls the appearance of the scrollbars that appear on the frame. This takes values either "yes", "no" or "auto". For example scrolling = "no" means it should not have scroll bars.
8	**longdesc** This attribute allows you to provide a link to another page containing a long description of the contents of the frame. For example longdesc = "framedescription.htm"

3.15 Disadvantages of Frames

There are few drawbacks with using frames, so it's never recommended to use frames in your webpages −

- Some smaller devices cannot cope with frames often because their screen is not big enough to be divided up.
- Sometimes your page will be displayed differently on different computers due to different screen resolution.
- The browser's *back* button might not work as the user hopes.
- There are still few browsers that do not support frame technology.

HTML iFrame

Module – 3

In this Module you will learn how to use an iframe to display a web page within another web page.

3.16 What is iframe?

An iframe or inline frame is used to display external objects including other web pages within a web page. An iframe pretty much acts like a mini web browser within a web browser. Also, the content inside an iframe exists entirely independent from the surrounding elements.

The basic syntax for adding an iframe to a web page can be given with:

```
<iframe src="URL"></iframe>
```

The URL specified in the src attribute points to the location of an external object or a web page.

The following example display "hello.html" file inside an iframe in current document.

```
<iframe src="hello.html"></iframe>
```

3.17 Setting Width and Height of an iFrame

The height and width attributes are used to specify the height and width of the iframe.

```
<iframe src="hello.html" width="400" height="200"></iframe>
```

You can also use CSS to set the width and height of an iframe, as shown here:

```
<iframe src="hello.html" style="width: 400px; height: 200px;"></iframe>
```

3.18 Removing Default Frameborder

The iframe has a border around it by default. However, if you want to modify or remove the iframe borders, the best way is to use the CSS border property.

The following example will simply render the iframe without any borders.

```
<iframe src="hello.html" style="border: none;"></iframe>
```

Similarly, you can use the border property to add the borders of your choice to an iframe. The following example will render the iframe with 2 pixels blue border.

```
<iframe src="hello.html" style="border: 2px solid blue;"></iframe>
```

3.19 Using an iFrame as Link Target

An iframe can also be used as a target for the hyperlinks.

An iframe can be named using the name attribute. This implies that when a link with a target attribute with that name as value is clicked, the linked resource will open in that iframe.

Let's try out an example to understand how it basically works:

```
<iframe src="demo-page.html" name="myFrame"></iframe>
<p><a href="https://www.tutorialrepublic.com" target="myFrame">Open
TutorialRepublic.com</a></p>
```

3.20 Difference between Frame and Iframe

Frames:

1. Frames are HTML tag which divides the browser's window into multiple parts where each part can load a separate HTML document.
2. <frame> tag specifies each frame within a frameset tag.
3. A collection of frames in the browser window is known as a frameset.
4. Each Frame can be linked with a single and different source.
5. Frame object represents an HTML frame

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Frames</title>
</head>
<frameset rows="20%,70%,10%">
   <frame name="top" src="/html/top.html" />
   <frame name="main" src="/html/main.html" />
   <frame name="bottom" src="/html/bottom.html" />
</frameset>
</html>
```

Iframe

1. Iframe is an inline frame and it is also used as <iframe> tag in HTML.
2. Iline frame means it is used to embed some other document within the current HTML document.
3. It is related to frameset but it can appear anywhere in your document.
4. <iframe> tag defines a rectangular region within the document in which the browser can display a separate document.
5. IFrame object represents an HTML inline frame.

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Iframes</title>
</head>
<body>
<p>See the video</p>
 <iframe width="854" height="480" src="https://www.youtube.com/embed/2eabXBvw4oI"
    frameborder="0"    allowfullscreen>
  </iframe>
</body>
</html>
```

	Frame	**IFrame**
Definition	Frame is a HTML tag that is used for dividing the web page into various frames/windows. Used as <frame> tag, it specifies each frame within a frameset tag.	Iframe as <iframe> is also a tag used in HTML but it specifies an inline frame, that means it is used to embed some other document within the current HTML document.
Tag frameset	Used	Not required
Placement of frames	Comparatively complicated	Easy
Width of frames/panes	Difficult to adjust	Easy to adjust

Multiple Choice Questions

1. Which one of the following is a form element?
    A. text box.
    B. radio button.
    C. submit button.
    D. All of these.
2. Choose the incorrect option.
    A. radio button allows to choose only one option from the given options.
    B. default option can be chosen using attribute "selected" in radio button
    C. default option can be chosen using attribute "checked" in radio button
    D. checkbox allows to choose one or more than one options from the given options.
3. Choose the correct option.
    A. Use of method attribute determines by which method the datas in the form will be submitted.
    B. Method can be POST or GET.
    C. Default method in HTML is GET.
    D. All of the above
4. Which one of the following does not hold true regarding GET method in HTML?
    A. Use of GET method in HTML is more secured.
    B. Use of GET method enables us to bookmark the page.
    C. GET has size limitation.
    D. None of the above
5. Which of the following tag is used for drop down list?
    A. <select>
    B. <text>
    C. <textarea>
    D. <dropdown>
6. How more than one option can be selected in drop down?
    A. Use of multiple attribute inside <option> tag.
    B. Use of multiple attribute inside <select> tag.
    C. use of multiple attribute inside <text> tag.
    D. It is not possible to select more than one option in drop down.
7. What is the default type of ‘type’ attribute of <input> element?
    A. Special characters
    B. Password
    C. Numerals
    D. Text
8. Which tag is used for grouping form controls?
    A. <label>
    B. <legend>
    C. <fieldset>
    D. <select>
9. In HTML form <input type="text"> is used for –
    A. One paragraph
    B. Block of text
    C. None
    D. One line text

10. The _____ attribute lets you turn off scrolling in a frame.
    A. OFFSCROLL
    B. none of the above
    C. SCROLLBAR
    D. SCROLLING
11. The _____ attribute indicates the number of rows in a frameset.
    A. ROWS
    B. TD
    C. HORIZONTAL
    D. TR
12. Which tag is not included in a frameset page?
    A. Div
    B. Body
    C. Title
    D. Head
13. Which of the following is a tag used in the creation of a frame definition file?
    A. All of the above
    B. <FRAMESET>
    C. <FRAME>
    D. <NOFRAMES>
14. What does the tag <NOFRAMES> do?
    A. Tells the browser to ignore the frame content.
    B. Allows information after the tag to be displayed with browsers that cannot handle frames.
    C. Specifies that no frames are included in the HTML file.
    D. None of these.
15. Which attribute of the frameset tag creates two horizontal frames?
    A. Rows
    B. None
    C. Both
    D. Cols
16. Which inline function embeds an independent HTML document into current document?
    A. <form>
    B.
    C. <iframe>
    D.
17. A group of frames is called as ___
    A. Index
    B. List
    C. Frameset
    D. Form
18. Which from the following is not a type of screen frames in HTML
    A. iframe
    B. noframe
    C. uframe
    D. frameset
19. The ___ tag and ___ attribute are used to force all links in a page to load in a particular frame.
    A. Frame and iframe
    B. Frame and Target
    C. Base and Frame
    D. Base and Target

20. Which tag embed an inline frame in a web page?
    A. Index
    B. Iframe
    C. Frame
    D. Object
21. The ____ attribute in frame tag specifies the web page to load into that frame.
    A. Id
    B. Href
    C. Src
    D. Name
22. Which of the following is not the value for frame attribute?
    A. none
    B. void
    C. above
    D. box
23. The _____ attribute indicates the number of columns in a frameset.
    A. COLS
    B. VERT
    C. COLUMNS
    D. TABLE
24. What is the use of iframe in HTML?
    A. to display a web page without browser
    B. to display a web page with animation effect
    C. to display a web page within a web page
    D. All of the Above

Unsolved Questions

1. What is HTML Form? Explain about form elements?
2. Explain about different types of form attributes?
3. Explain about Input Element in HTML Form and explain attributes supports for Input element? **Lear each element of attributes?**
4. Explain about HTML frames and how to create frames vertically and horizontally?
5. Explain about **frameset** tag attributes
6. Explain about **frame** tag attributes
7. Describe **iframe** with example of HTML source code
8. Explain difference between **frame** and **iframe** with example of HTML source code

Unit 4:: CSS
Module1

4.1. Introduction:

4.1.1. What is CSS?

- CSS stands for Cascading Style Sheets
- CSS describes how HTML elements are to be displayed on screen, paper, or in other media
- CSS saves a lot of work. It can control the layout of multiple web pages all at once
- External stylesheets are stored in CSS files

4.1.2. Why Use CSS?

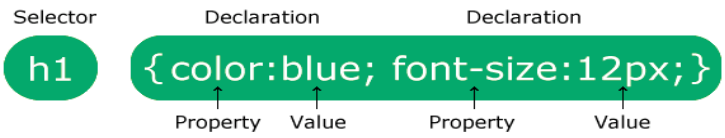
CSS is used to define styles for your web pages, including the design, layout and variations in display for different devices and screen sizes.

CSS Example:

```
<!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>My First CSS Example</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

Output:

CSS Syntax:



The selector points to the HTML element you want to style.
The declaration block contains one or more declarations separated by semicolons.
Each declaration includes a CSS property name and a value, separated by a colon.
Multiple CSS declarations are separated with semicolons, and declaration blocks are surrounded by curly braces.

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
  color: red;
  text-align: center;
}
</style>
</head>
<body>

<p>Hello World!</p>
<p>These paragraphs are styled with CSS.</p>

</body>
</html>
```

Output:

Hello World!

These paragraphs are styled with CSS.

Example Explained:

- p is a selector in CSS (it points to the HTML element you want to style: <p>).
- color is a property, and red is the property value
- text-align is a property, and center is the property value

4.2. CSS Selector:

A CSS selector selects the HTML element(s) you want to style.We can divide CSS selectors into five categories:

- Simple selectors (select elements based on name, id, class)
- Combinator selectors (select elements based on a specific relationship between them)
- Pseudo-class selectors (select elements based on a certain state)
- Pseudo-elements selectors (select and style a part of an element)
- Attribute selectors (select elements based on an attribute or attribute value)

4.2.1 The CSS element Selector:

The element selector selects HTML elements based on the element name.

Example:

Here, all <p> elements on the page will be center-aligned, with a red text color:

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
  text-align: center;
  color: red;
}
</style>
</head>
<body>

<p>Every paragraph will be affected by the style.</p>
<p id="para1">Me too!</p>
<p>And me!</p>

</body>
</html>
```

Output:

Every paragraph will be affected by the style.

Me too!

And me!

4.2.2. The CSS id Selector:

The id selector uses the id attribute of an HTML element to select a specific element.
The id of an element is unique within a page, so the id selector is used to select one unique element!
To select an element with a specific id, write a hash (#) character, followed by the id of the element.

Example:

The CSS rule below will be applied to the HTML element with id="para1":

```
<!DOCTYPE html>
<html>
<head>
<style>
#para1 {
  text-align: center;
  color: red;
}
</style>
</head>
<body>

<p id="para1">Hello World!</p>
<p>This paragraph is not affected by the style.</p>

</body>
</html>
```

Output:

Hello World!

This paragraph is not affected by the style.

4.2.3. The CSS class Selector:

The class selector selects HTML elements with a specific class attribute.
To select elements with a specific class, write a period (.) character, followed by the class name.

Example:

In this example all HTML elements with class="center" will be red and center-aligned:

```
<!DOCTYPE html>
<html>
<head>
<style>
.center {
  text-align: center;
  color: red;
}
</style>
</head>
<body>

<h1 class="center">Red and center-aligned heading</h1>
<p class="center">Red and center-aligned paragraph.</p>

</body>
</html>
```

Output:

Red and center-aligned heading

Red and center-aligned paragraph.

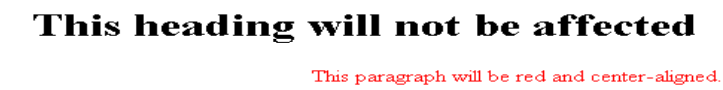
You can also specify that only specific HTML elements should be affected by a class.

Example:

In this example only <p> elements with class="center" will be red and center-aligned:

```
<!DOCTYPE html>
<html>
<head>
<style>
p.center {
  text-align: center;
  color: red;
}
</style>
</head>
<body>

<h1 class="center">This heading will not be affected</h1>
<p class="center">This paragraph will be red and center-aligned.</p>

</body>
</html>
```

Output:



HTML elements can also refer to more than one class.

Example:

In this example the <p> element will be styled according to class="center" and to class="large":

```
<!DOCTYPE html>
<html>
<head>
<style>
p.center {
  text-align: center;
  color: red;
}

p.large {
  font-size: 300%;
}
</style>
</head>
<body>

<h1 class="center">This heading will not be affected</h1>
<p class="center">This paragraph will be red and center-aligned.</p>
<p class="center large">This paragraph will be red, center-aligned, and in a
large font-size.</p>

</body>
</html>
```

Output:

Note: A class name cannot start with a number.

4.2.4. The CSS Universal Selector:

The universal selector (*) selects all HTML elements on the page.

Example:

The CSS rule below will affect every HTML element on the page:

```
<!DOCTYPE html>
<html>
<head>
<style>
* {
  text-align: center;
  color: blue;
}
</style>
</head>
<body>

<h1>Hello world!</h1>

<p>Every element on the page will be affected by the style.</p>
<p id="para1">Me too!</p>
<p>And me!</p>

</body>
</html>
```

Output:

4.2.5. The CSS Grouping Selector:

The grouping selector selects all the HTML elements with the same style definitions. Look at the following CSS code (the h1, h2, and p elements have the same style definitions). It will be better to group the selectors, to minimize the code. To group selectors, separate each selector with a comma.

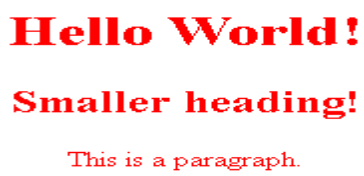
```
h1 {
  text-align: center;
  color: red;
}

h2 {
  text-align: center;
  color: red;
}

p {
  text-align: center;
  color: red;
}
```

Example:

In this example we have grouped the selectors from the code above:

```
<!DOCTYPE html>
<html>
<head>
<style>
h1, h2, p {
  text-align: center;
  color: red;
}
</style>
</head>
<body>

<h1>Hello World!</h1>
<h2>Smaller heading!</h2>
<p>This is a paragraph.</p>

</body>
</html>
```

Output:



4.2.6. All CSS Simple Selectors:

Selector	Example	Example description
#id	#firstname	Selects the element with id="firstname"
.class	.intro	Selects all elements with class="intro"
element.class	p.intro	Selects only <p> elements with class="intro"
*	*	Selects all elements
element	p	Selects all <p> elements
element,element,...	div, p	Selects all elements and all <p> elements

4.3. Font Properties:

4.3.1. Font Selection is Important

Choosing the right font has a huge impact on how the readers experience a website. The right font can create a strong identity for your brand. Using a font that is easy to read is important. The font adds value to your text. It is also important to choose the correct color and text size for the font.

4.3.2. Generic Font Families:

In CSS there are five generic font families:

1. **Serif** fonts have a small stroke at the edges of each letter. They create a sense of formality and elegance.
2. **Sans-serif** fonts have clean lines (no small strokes attached). They create a modern and minimalistic look.
3. **Monospace** fonts - here all the letters have the same fixed width. They create a mechanical look.
4. **Cursive** fonts imitate human handwriting.
5. **Fantasy** fonts are decorative/playful fonts.

All the different font names belong to one of the generic font families.

Difference Between Serif and Sans-serif Fonts

Note: On computer screens, sans-serif fonts are considered easier to read than serif fonts.

Some Font Examples:

Generic Font Family	Examples of Font Names
Serif	Times New Roman Georgia Garamond
Sans-serif	Arial Verdana Helvetica
Monospace	Courier New Lucida Console Monaco
Cursive	Brush Script MT Lucida Handwriting
Fantasy	Copperplate Papyrus

4.3.3. The CSS font-family Property:

In CSS, we use the font-family property to specify the font of a text. The font-family property should hold several font names as a "fallback" system, to ensure maximum compatibility between browsers/operating systems. Start with the font you want, and end with a generic family (to let the browser pick a similar font in the generic family, if no other fonts are available). The font names should be separated with comma.

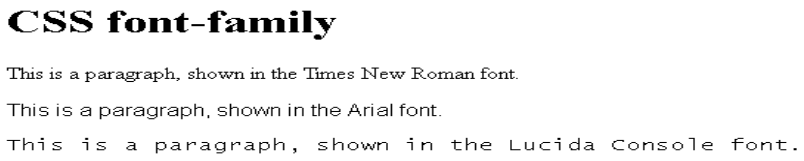
Note: If the font name is more than one word, it must be in quotation marks, like: “Times New Roman”.

Example:

Specify some different fonts for three paragraphs:

```
<!DOCTYPE html>
<html>
<head>
<style>
.p1 {
  font-family: "Times New Roman", Times, serif;
}

.p2 {
  font-family: Arial, Helvetica, sans-serif;
}

.p3 {
  font-family: "Lucida Console", "Courier New", monospace;
}
</style>
</head>
<body>

<h1>CSS font-family</h1>
<p class="p1">This is a paragraph, shown in the Times New Roman font.</p>
<p class="p2">This is a paragraph, shown in the Arial font.</p>
<p class="p3">This is a paragraph, shown in the Lucida Console font.</p>

</body>
</html>
```

Output:



4.3.4. Font Style:

The font-style property is mostly used to specify italic text.

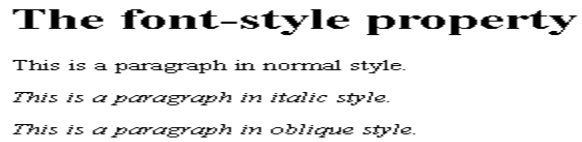
This property has three values:

- normal - The text is shown normally
- italic - The text is shown in italics
- oblique - The text is "leaning" (oblique is very similar to italic, but less supported)

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
p.normal {
  font-style: normal;
}

p.italic {
  font-style: italic;
}

p.oblique {
  font-style: oblique;
}
</style>
</head>
<body>

<h1>The font-style property</h1>

<p class="normal">This is a paragraph in normal style.</p>
<p class="italic">This is a paragraph in italic style.</p>
<p class="oblique">This is a paragraph in oblique style.</p>

</body>
</html>
```

Output:



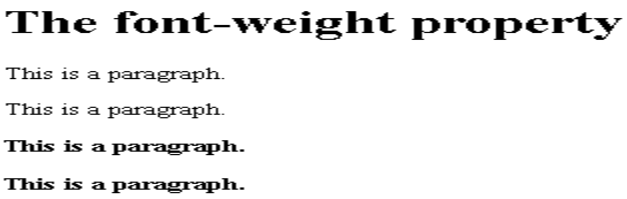
4.3.5. Font Weight:

The font-weight property specifies the weight of a font:

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
p.normal {
  font-weight: normal;
}

p.light {
  font-weight: lighter;
}

p.thick {
  font-weight: bold;
}

p.thicker {
  font-weight: 900;
}
</style>
</head>
<body>

<h1>The font-weight property</h1>

<p class="normal">This is a paragraph.</p>
<p class="light">This is a paragraph.</p>
<p class="thick">This is a paragraph.</p>
<p class="thicker">This is a paragraph.</p>

</body>
</html>
```

Output:



4.3.6. Font Variant:

The font-variant property specifies whether or not a text should be displayed in a small-caps font. In a small-caps font, all lowercase letters are converted to uppercase letters. However, the converted uppercase letters appears in a smaller font size than the original uppercase letters in the text.

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
p.normal {
  font-variant: normal;
}

p.small {
  font-variant: small-caps;
}
</style>
</head>
<body>

<h1>The font-variant property</h1>

<p class="normal">My name is Hege Refsnes.</p>
<p class="small">My name is Hege Refsnes.</p>

</body>
</html>
```

Output:

The font-variant property

My name is Hege Refsnes.

My name is Hege Refsnes.

4.3.7. Font Size:

The font-size property sets the size of the text. Being able to manage the text size is important in web design. However, you should not use font size adjustments to make paragraphs look like headings, or headings look like paragraphs. Always use the proper HTML tags, like <h1> - <h6> for headings and <p> for paragraphs.
The font-size value can be an absolute or relative size.

Absolute size:

- Sets the text to a specified size
- Does not allow a user to change the text size in all browsers (bad for accessibility reasons)
- Absolute size is useful when the physical size of the output is known

Relative size:

- Sets the size relative to surrounding elements
- Allows a user to change the text size in browsers

Note: If you do not specify font size, the default size for normal text, like paragraph, is 16px.

4.3.8. Set Font Size with Pixels

Setting the text size with pixels gives you full control over the text size:

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
  font-size: 40px;
}

h2 {
  font-size: 30px;
}

p {
  font-size: 14px;
}
</style>
</head>
<body>

<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>

</body>
</html>
```

Output:

This is heading 1

This is heading 2

This is a paragraph.

This is another paragraph.

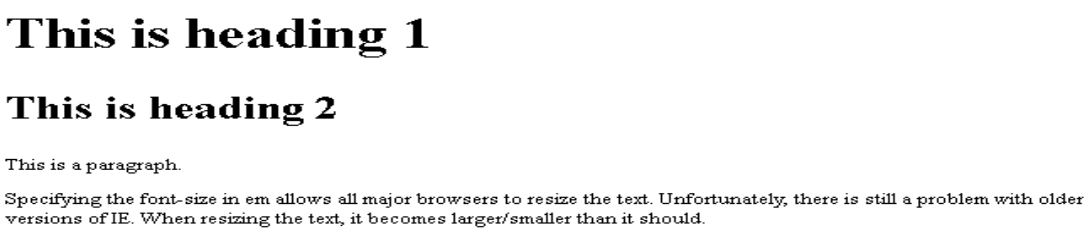
4.3.9. Set Font Size With Em:

To allow users to resize the text (in the browser menu), many developers use em instead of pixels.
1em is equal to the current font size. The default text size in browsers is 16px. So, the default size of 1em is 16px.
The size can be calculated from pixels to em using this formula: *pixels*/16=*em*

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
  font-size: 2.5em; /* 40px/16=2.5em */
}

h2 {
  font-size: 1.875em; /* 30px/16=1.875em */
 }

p {
  font-size: 0.875em; /* 14px/16=0.875em */
}
</style>
</head>
<body>

<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<p>This is a paragraph.</p>
<p>Specifying the font-size in em allows all major browsers to resize the text.
Unfortunately, there is still a problem with older versions of IE. When resizing
the text, it becomes larger/smaller than it should.</p>

</body>
</html>
```

Output:



In the example above, the text size in em is the same as the previous example in pixels. However, with the em size, it is possible to adjust the text size in all browsers. Unfortunately, there is still a problem with older versions of Internet Explorer. The text becomes larger than it should when made larger, and smaller than it should when made smaller.

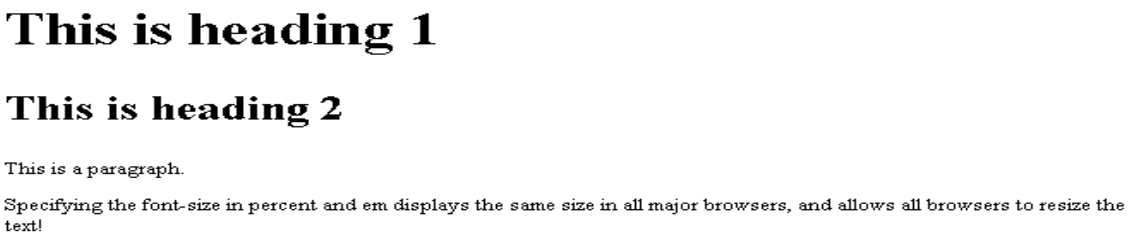
4.3.10. Use a Combination of Percent and 'Em':

The solution that works in all browsers is to set a default font-size in percent for the <body> element:

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  font-size: 100%;
}

h1 {
  font-size: 2.5em;
}

h2 {
  font-size: 1.875em;
}

p {
  font-size: 0.875em;
}
</style>
</head>
<body>

<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<p>This is a paragraph.</p>
<p>Specifying the font-size in percent and em displays the same size in all
major browsers, and allows all browsers to resize the text!</p>

</body>
</html>
```

Output:



4.4. Various Types of Style Sheets:

There are three ways of inserting a style sheet:

- External CSS
- Internal CSS
- Inline CSS

4.4.1. External CSS:

With an external style sheet, you can change the look of an entire website by changing just one file!
Each HTML page must include a reference to the external style sheet file inside the <link> element, inside the head section.

Example:

External styles are defined within the <link> element, inside the <head> section of an HTML page

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="mystyle.css">
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

Output:

An external style sheet can be written in any text editor, and must be saved with a .css extension.
The external .css file should not contain any HTML tags.
Here is how the "mystyle.css" file looks:

"mystyle.css"

```
body {
  background-color: lightblue;
}

h1 {
  color: navy;
  margin-left: 20px;
}
```

4.4.2. Internal CSS:

An internal style sheet may be used if one single HTML page has a unique style. The internal style is defined inside the <style> element, inside the head section.

Example:

Internal styles are defined within the <style> element, inside the <head> section of an HTML page

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-color: linen;
}

h1 {
  color: maroon;
  margin-left: 40px;
}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

Output:

4.4.3. Inline CSS:

An inline style may be used to apply a unique style for a single element. To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.

Example:

Inline styles are defined within the style attribute of the relevant element.

```
<!DOCTYPE html>
<html>
<body>

<h1 style="color:blue;text-align:center;">This is a heading</h1>
<p style="color:red;">This is a paragraph.</p>

</body>
</html>
```

Output:

4.4.4. Multiple Style Sheets:

If some properties have been defined for the same selector (element) in different style sheets, the value from the last read style sheet will be used. Assume that an external style sheet has the following style for the <h1> element.

```
h1 {
  color: navy;
}
```

Then, assume that an internal style sheet also has the following style for the <h1>

```
h1 {
  color: orange;
}
```

Example:

If the internal style is defined after the link to the external style sheet, the <h1> elements will be "orange"

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
<style>
h1 {
  color: orange;
}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>The style of this document is a combination of an external stylesheet, and
internal style</p>

</body>
</html>
```

Output:

Example:

However, if the internal style is defined before the link to the external style sheet, the <h1> elements will be navy

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
  color: orange;
}
</style>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
<body>

<h1>This is a heading</h1>
<p>The style of this document is a combination of an external stylesheet, and
internal style</p>

</body>
</html>
```

Output:

4.4.5. Cascading Order:

What style will be used when there is more than one style specified for an HTML element?

All the styles in a page will "cascade" into a new "virtual" style sheet by the following rules, where number one has the highest priority:

1. Inline style (inside an HTML element)
2. External and internal style sheets (in the head section)
3. Browser default

So, an inline style has the highest priority, and will override external and internal styles and browser defaults.

Module 2

4.5. Formatting Text:

CSS has a lot of properties for formatting text.

4.5.1. Text Color:

The color property is used to set the color of the text. The color is specified by:

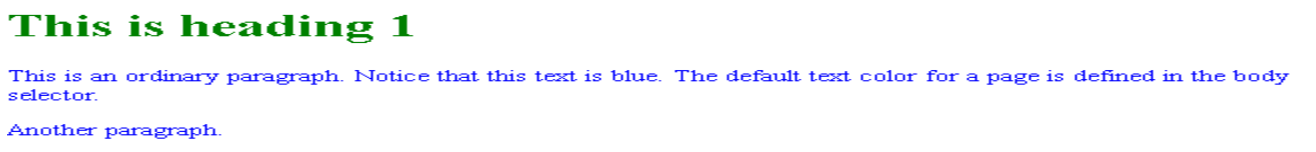
- a color name - like "red"
- a HEX value - like "#ff0000"
- an RGB value - like "rgb(255,0,0)"

The default text color for a page is defined in the body selector.

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  color: blue;
}

h1 {
  color: green;
}
</style>
</head>
<body>

<h1>This is heading 1</h1>
<p>This is an ordinary paragraph. Notice that this text is blue. The default text color for a page is defined in the body selector.</p>
<p>Another paragraph.</p>

</body>
</html>
```

Output:



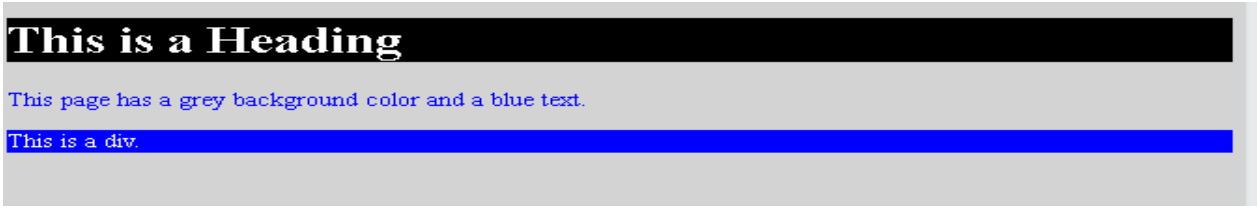
4.5.2. Text Color and Background Color:

In this example, we define both the background-color property and the color property:

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-color: lightgrey;
  color: blue;
}

h1 {
  background-color: black;
  color: white;
}

div {
  background-color: blue;
  color: white;
}
</style>
</head>
<body>

<h1>This is a Heading</h1>
<p>This page has a grey background color and a blue text.</p>
<div>This is a div.</div>

</body>
</html>
```

Output:



4.5.3. Text Alignment:

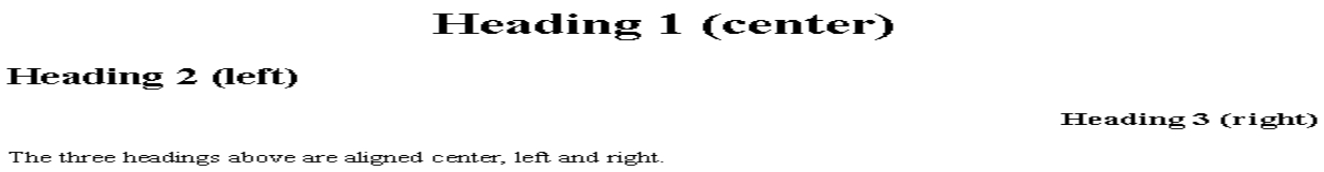
The text-align property is used to set the horizontal alignment of a text. A text can be left or right aligned, centered, or justified.

The following example shows center aligned, and left and right aligned text (left alignment is default if text direction is left-to-right, and right alignment is default if text direction is right-to-left):

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
  text-align: center;
}

h2 {
  text-align: left;
}

h3 {
  text-align: right;
}
</style>
</head>
<body>

<h1>Heading 1 (center)</h1>
<h2>Heading 2 (left)</h2>
<h3>Heading 3 (right)</h3>

<p>The three headings above are aligned center, left and right.</p>

</body>
</html>
```

Output:

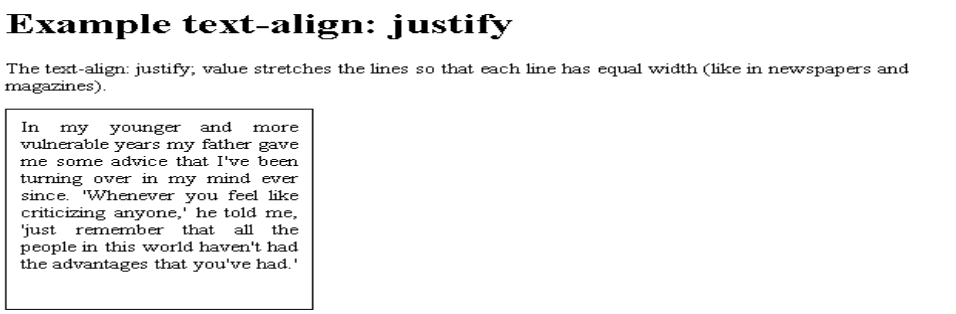


When the text-align property is set to "justify", each line is stretched so that every line has equal width, and the left and right margins are straight (like in magazines and newspapers).

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  border: 1px solid black;
  padding: 10px;
  width: 200px;
  height: 200px;
  text-align: justify;
}
</style>
</head>
<body>

<h1>Example text-align: justify</h1>

<p>The text-align: justify; value stretches the lines so that each line has
equal width (like in newspapers and magazines).</p>

<div>
In my younger and more vulnerable years my father gave me some advice that I've
been turning over in my mind ever since. 'Whenever you feel like criticizing
anyone,' he told me, 'just remember that all the people in this world haven't
had the advantages that you've had.'
</div>

</body>
</html>
```

Output:



4.5.4. Text Direction:

The direction and unicode-bidi properties can be used to change the text direction of an element

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
p.ex1 {
  direction: rtl;
  unicode-bidi: bidi-override;
}
</style>
</head>
<body>

<p>This is the default text direction.</p>

<p class="ex1">This is right-to-left text direction.</p>

</body>
</html>
```

Output:

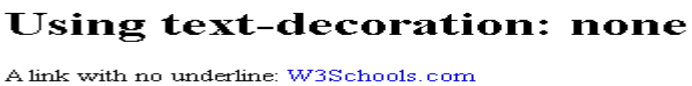
4.5.5. Text Decoration:

The text-decoration property is used to set or remove decorations from text. The value text-decoration: none; is often used to remove underlines from links:

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
a {
  text-decoration: none;
}
</style>
</head>
<body>

<h1>Using text-decoration: none</h1>

<p>A link with no underline: <a
href="https://www.w3schools.com">W3Schools.com</a></p>

</body>
</html>
```

Output:



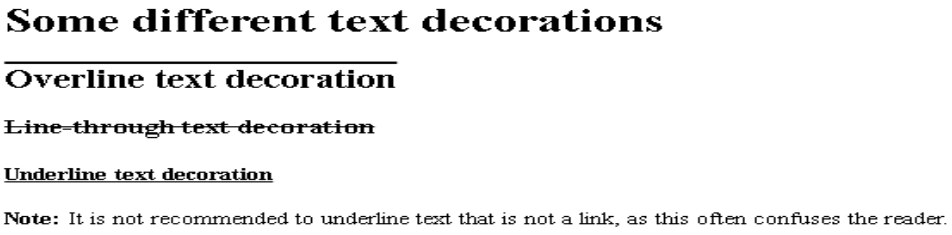
A link with no underline: W3Schools.com

The other text-decoration values are used to decorate text:

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
h2 {
  text-decoration: overline;
}

h3 {
  text-decoration: line-through;
}

h4 {
  text-decoration: underline;
}
</style>
</head>
<body>

<h1>Some different text decorations</h1>

<h2>Overline text decoration</h2>
<h3>Line-through text decoration</h3>
<h4>Underline text decoration</h4>

<p><strong>Note:</strong> It is not recommended to underline text that is not a
link, as this often confuses
the reader.</p>

</body>
</html>
```

Output:



4.5.6. Text Transformation

The text-transform property is used to specify uppercase and lowercase letters in a text.
It can be used to turn everything into uppercase or lowercase letters, or capitalize the first letter of each word:

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
p.uppercase {
  text-transform: uppercase;
}

p.lowercase {
  text-transform: lowercase;
}

p.capitalize {
  text-transform: capitalize;
}
</style>
</head>
<body>

<h1>Using the text-transform property</h1>

<p class="uppercase">This text is transformed to uppercase.</p>
<p class="lowercase">This text is transformed to lowercase.</p>
<p class="capitalize">This text is capitalized.</p>

</body>
</html>
```

Output:

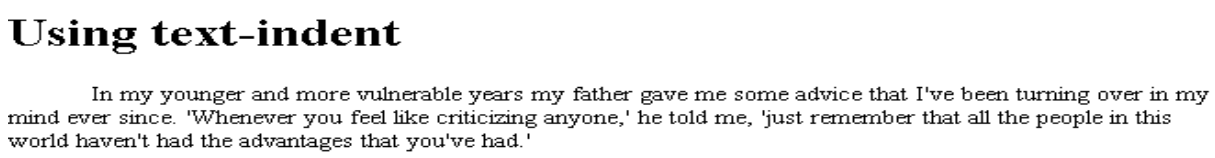
4.5.7. Text Indentation:

The text-indent property is used to specify the indentation of the first line of a text:

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
  text-indent: 50px;
}
</style>
</head>
<body>

<h1>Using text-indent</h1>

<p>In my younger and more vulnerable years my father gave me some advice that
I've been turning over in my mind ever since. 'Whenever you feel like criticizing
anyone,' he told me, 'just remember that all the people in this world haven't had
the advantages that you've had.'</p>

</body>
</html>
```

Output:



4.5.8. Letter Spacing:

The letter-spacing property is used to specify the space between the characters in a text. The following example demonstrates how to increase or decrease the space between characters:

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
h2 {
  letter-spacing: 5px;
}

h3 {
  letter-spacing: -2px;
}
</style>
</head>
<body>

<h1>Using letter-spacing</h1>

<h2>This is heading 1</h2>
<h3>This is heading 2</h3>

</body>
</html>
```

Output:

Using letter-spacing

This is heading 1

This is heading 2

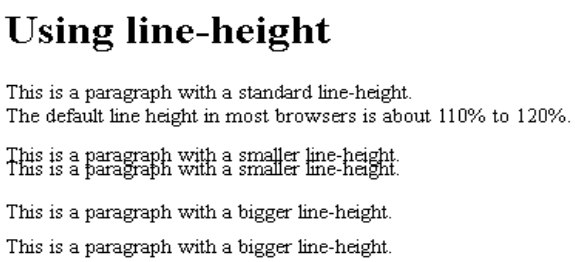
4.5.9. Line Height:

The line-height property is used to specify the space between lines

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
p.small {
  line-height: 0.7;
}

p.big {
  line-height: 1.8;
}
</style>
</head>
<body>

<h1>Using line-height</h1>

<p>
This is a paragraph with a standard line-height.<br>
The default line height in most browsers is about 110% to 120%.<br>
</p>

<p class="small">
This is a paragraph with a smaller line-height.<br>
This is a paragraph with a smaller line-height.<br>
</p>

<p class="big">
This is a paragraph with a bigger line-height.<br>
This is a paragraph with a bigger line-height.<br>
</p>

</body>
</html>
```

Output:



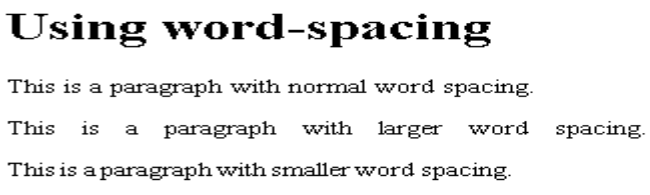
4.5.10. Word Spacing:

The word-spacing property is used to specify the space between the words in a text.
The following example demonstrates how to increase or decrease the space between words.

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
p.one {
  word-spacing: 10px;
}

p.two {
  word-spacing: -2px;
}
</style>
</head>
<body>

<h1>Using word-spacing</h1>

<p>This is a paragraph with normal word spacing.</p>

<p class="one">This is a paragraph with larger word spacing.</p>

<p class="two">This is a paragraph with smaller word spacing.</p>

</body>
</html>
```

Output:



4.5.11. White Space:

The white-space property specifies how white-space inside an element is handled.
This example demonstrates how to disable text wrapping inside an element

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
  white-space: nowrap;
}
</style>
</head>
<body>

<h1>Using white-space</h1>

<p>
This is some text that will not wrap.
This is some text that will not wrap.
This is some text that will not wrap.
This is some text that will not wrap.
This is some text that will not wrap.
This is some text that will not wrap.
This is some text that will not wrap.
This is some text that will not wrap.
This is some text that will not wrap.
</p>

<p>Try to remove the white-space property to see the difference!</p>

</body>
</html>
```

4.5.12. Text Shadow:

The text-shadow property adds shadow to text.

In its simplest use, you only specify the horizontal shadow (2px) and the vertical shadow (2px).

Text Shadow Effect:

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
  text-shadow: 2px 2px;
}
</style>
</head>
<body>

<h1>Text-shadow effect!</h1>

</body>
</html>
```

Output:

Text-shadow effect!

Next, add a color (red) to the shadow

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
  text-shadow: 2px 2px red;
}
</style>
</head>
<body>

<h1>Text-shadow effect!</h1>

</body>
</html>
```

Text-shadow effect!

Then, add a blur effect (5px) to the shadow

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
  text-shadow: 2px 2px 5px red;
}
</style>
</head>
<body>

<h1>Text-shadow effect!</h1>

</body>
</html>
```

Text-shadow effect!

All CSS Text Properties

Property	Description
color	Sets the color of text
direction	Specifies the text direction/writing direction
letter-spacing	Increases or decreases the space between characters in a text
line-height	Sets the line height
text-align	Specifies the horizontal alignment of text
text-decoration	Specifies the decoration added to text
text-indent	Specifies the indentation of the first line in a text-block
text-shadow	Specifies the shadow effect added to text
text-transform	Controls the capitalization of text
text-overflow	Specifies how overflowed content that is not displayed should be signaled to the user
unicode-bidi	Used together with the direction property to set or return whether the text should be overridden to support multiple languages in the same document
vertical-align	Sets the vertical alignment of an element
white-space	Specifies how white-space inside an element is handled
word-spacing	Increases or decreases the space between words in a text

4.6. Colors:

Colors are specified using predefined color names, or RGB, HEX, HSL, RGBA, HSLA values.

4.6.1. CSS color names:

In CSS, a color can be specified by using a predefined color name

CSS/HTML supports 140 standard color names.

4.6.2. CSS Background Color

You can set the background color for HTML elements:

```
<!DOCTYPE html>
<html>
<body>

<h1 style="background-color:DodgerBlue;">Hello World</h1>

<p style="background-color:Tomato;">
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh
euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.
Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit
lobortis nisl ut aliquip ex ea commodo consequat.
</p>

</body>
</html>
```

Hello World

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

4.6.3. CSS Text Color

You can set the color of text

```
<!DOCTYPE html>
<html>
<body>

<h3 style="color:Tomato;">Hello World</h3>

<p style="color:DodgerBlue;">Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.</p>

<p style="color:MediumSeaGreen;">Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.</p>

</body>
</html>
```

Hello World

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

4.6.4. CSS Border Color

You can set the color of borders

```
<!DOCTYPE html>
<html>
<body>

<h1 style="border: 2px solid Tomato;">Hello World</h1>

<h1 style="border: 2px solid DodgerBlue;">Hello World</h1>

<h1 style="border: 2px solid Violet;">Hello World</h1>

</body>
</html>
```

Hello World

Hello World

Hello World

4.6.5. CSS Color Values

In CSS, colors can also be specified using RGB values, HEX values, HSL values, RGBA values, and HSLA values:

Same as color name "Tomato":

```
<!DOCTYPE html>
<html>
<body>

<p>Same as color name "Tomato":</p>

<h1 style="background-color:rgb(255, 99, 71);">rgb(255, 99, 71)</h1>
<h1 style="background-color:#ff6347;">#ff6347</h1>
<h1 style="background-color:hsl(9, 100%, 64%);">hsl(9, 100%, 64%)</h1>

<p>Same as color name "Tomato", but 50% transparent:</p>
<h1 style="background-color:rgba(255, 99, 71, 0.5);">rgba(255, 99, 71, 0.5)</h1>
<h1 style="background-color:hsla(9, 100%, 64%, 0.5);">hsla(9, 100%, 64%, 0.5)
</h1>

<p>In addition to the predefined color names, colors can be specified using RGB,
HEX, HSL, or even transparent colors using RGBA or HSLA color values.</p>

</body>
</html>
```

Same as color name "Tomato":

rgb(255, 99, 71)

#ff6347

hsl(9, 100%, 64%)

Same as color name "Tomato", but 50% transparent:

rgba(255, 99, 71, 0.5)

hsla(9, 100%, 64%, 0.5)

In addition to the predefined color names, colors can be specified using RGB, HEX, HSL, or even transparent colors using RGBA or HSLA color values.

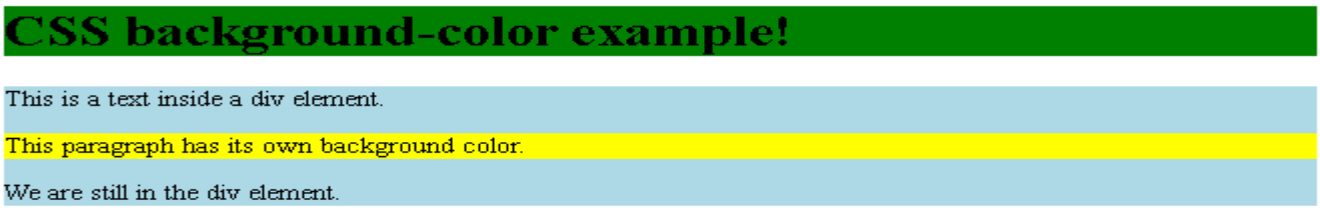
4.7.Backgrounds:

The CSS background properties are used to add background effects for elements.

4.7.1. CSS background-color

The background-color property specifies the background color of an element.

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
  background-color: green;
}

div {
  background-color: lightblue;
}

p {
  background-color: yellow;
}
</style>
</head>
<body>

<h1>CSS background-color example!</h1>
<div>
This is a text inside a div element.
<p>This paragraph has its own background color.</p>
We are still in the div element.
</div>

</body>
</html>
```



4.7.2. CSS background-image

The background-image property specifies an image to use as the background of an element.
By default, the image is repeated so it covers the entire element.

```
body {
  background-image: url("paper.gif");
}
```

4.7.2.1.CSS background-repeat

By default, the background-image property repeats an image both horizontally and vertically.
Some images should be repeated only horizontally or vertically, or they will look strange, like this

```
body {
  background-image: url("gradient_bg.png");
}
```

If the image above is repeated only horizontally (background-repeat: repeat-x;), the background will look better

```
body {
  background-image: url("gradient_bg.png");
  background-repeat: repeat-x;
}
```

Showing the background image only once is also specified by the background-repeat property

```
body {
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
}
```

The background-position property is used to specify the position of the background image

```
body {
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
  background-position: right top;
}
```

4.7.2.2.Background-attachment

The background-attachment property specifies whether the background image should scroll or be fixed (will not scroll with the rest of the page):
Specify that the background image should be fixed.

```
body {
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
  background-position: right top;
  background-attachment: fixed;
}
```

Specify that the background image should be scroll with the rest of the page.

```
body {
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
```

```
  background-position: right top;
  background-attachment: scroll;
}
```

4.7.2.3.Background - Shorthand property

To shorten the code, it is also possible to specify all the background properties in one single property. This is called a shorthand property. Instead of writing:

```
body {
  background-color: #ffffff;
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
  background-position: right top;
}
```

You can use the shorthand property background

```
body {
  background: #ffffff url("img_tree.png") no-repeat right top;
}
```

When using the shorthand property the order of the property values is:

- background-color
- background-image
- background-repeat
- background-attachment
- background-position

It does not matter if one of the property values is missing, as long as the other ones are in this order. Note that we do not use the background-attachment property in the examples above, as it does not have a value.

Objective Questions

1. Which HTML tag is used to define an internal style sheet?
   - **a. <style>**
   - b. <css>
   - c. <script>
   - d. <body>
2. Where to insert an external style sheet into an HTML document?
   - a. At the top of the document
   - **b. In the <head> section**
   - c. In the <body> section
   - d. At the end of the document
3. Which HTML attribute is used to define inline styles?
   - a. Font
   - b. Class
   - c. Styles
   - **d. Style**
4. Which is the correct CSS syntax?
   - a. {body;color:black}
   - b. body:color=black
   - **c. body{color:black}**
   - d. {body:color=black(body)}

5. We use …….. to inset comments in a CSS file.
    a. // ….//
    b. /* …. */
    c. ‘ …. ‘
    d. //
6. How do you change the text color of an element?
    a. Textcolor:
    b. Fgcolor:
    c. color:
    d. Text-color:
7. How to add background color for all <h1> elements?
    a. All.h1{background-color:#FFFFFF}
    b. h1.all{background-color:#FFFFFF}
    c. <h1>all{background-color:#FFFFFF}
    d. **h1{background-color:#FFFFFF}**
8. Which CSS property is used to increase/decrease the text size?
    a. font-size
    b. font-style
    c. text-size
    d. text-style
9. What is the correct syntax for making all the <p> elements bold?
    a. <p style=”text-size:bold>
    b. p{font-weight:bold}
    c. p{text-size:bold]
    d. <p style=”text-weight:bold”>
10. How do you display hyperlinks without an underline?
    a. a{text-decoration:nounderline}
    b. a{decoration:nounderline}
    c. a{text-decoration:none}
    d. a{underline:none}
11. How do you make each word in a text start with a capital letter?
    a. Text-transform:capitalize
    b. Text-transform:uppercare
    c. Text-transform:lowercase
    d. None of the above
12. CSS code to change the font of an element.
    a. Font:
    b. F
    c. Fontstyle
    d. Font-family
13. How do you change the left margin of an element?
    a. Margin:
    b. Indent:
    c. Margin-left:

d. Text-indent:

14. How do you make a list that lists its items with squares?
    a. List-type:square
    b. Type:square
    c. Type:list-square
    d. List-style-type:square
15. Which of the following is used to specify the subscript of text using CSS?
    a. vertical-align:sub
    b. vertical-align:sup
    c. vertical-align:subscript
    d. none

Descriptive Questions

1. What is CSS? What are the advantages and limitations of CSS?
2. How many methods are there for applying CSS to an HTML document? And explain them.
3. Explain about all the font properties (family, style, size, variant and weight) with suitable examples
4. Explain about all the text properties (indent, align, decoration, letter spacing and text transform) with suitable examples
5. How to apply borders for an element?
6. Explain about short form of border property.
7. How to apply background color?
8. How to insert background image and explain about background image properties?

Unit: 5 Introductions to JavaScript
Module-1

5.1.1 Introduction

JavaScript is the globally used client-side scripting language for the web. Most browsers support the language by default, so you can get started using JavaScript and HTML with a simple text editor and browser for testing. Client-side, programming languages make web pages dynamic without making calls to your web servers for every button clicked, character typed, or mouse moved. The JavaScript language is so popular that thousands of developers have made customized libraries that make development even easier for other programmers and web designers. If you design web pages, you'll certainly need to know JavaScript to make a custom UI (user interface).

This course gets you started with an introduction to JavaScript. We assume that you're new to the language, so it gets you started with basic functionality such as creating functions, creating variables, and calling these lines of code from your standard HTML pages. We talk about events and triggers for custom event handling. We also discuss pattern matching, searching for text within a page, flow control and the document object model (DOM).

5.1.2 History:

HTML's first version, designed by **Tim Berners-Lee** from 1989 to 1991, was fairly static in nature. Except for link jumps with the element, web pages simply displayed content, and the content was fixed. In 1995, the dominant browser manufacturer was Netscape, and one of its employees, **Brendan Eich**, thought that it would be useful to add dynamic functionality to web pages. So he designed the JavaScript programming language, which adds dynamic functionality to web pages when used in conjunction with HTML. For example, JavaScript provides the ability to update a web page's content when an event occurs, such as when a user clicks a button. It also provides the ability to retrieve a user's input and process that input.

In 1996, Netscape submitted JavaScript to the Ecma International1 standards organization to promote JavaScript's influence on all browsers (not just Netscape's browser). Ecma International used JavaScript as the basis for creating the ECMAScript standard. As hoped, ECMAScript now serves as the standard for the interactive programming languages embedded in all of today's popular browsers. The most recent ECMAScript version is version 7, published in 2016. Coming up with the name ECMAScript was a difficult process, with different browser manufacturers having strong opposing views. JavaScript creator **Brendan Eich** has stated that the result, ECMAScript, is "an unwanted trade name that sounds like a skin disease. In 1998, Netscape formed the Mozilla free-software community, which eventually implemented Firefox, one of today's premier browsers. Brendan Eich moved to Mozilla, where he and others have continued to update JavaScript over the years, following the ECMAScript standard as set forth by Ecma International.
Other browser manufacturers support their own versions of JavaScript. For their Internet Explorer and Edge browsers, Microsoft uses JScript. For their Chrome browser, Google uses the V8 JavaScript Engine. Fortunately, all the browser manufacturers attempt to follow the ECMAScript standard, so for most tasks, programmers can write one version of their code and it will work for all the different browsers.
Before you continue you should have a basic understanding of the following:

- HTML / XHTML

5.1.3 JavaScript is:

- JavaScript was designed to add interactivity to HTML pages
- JavaScript is a scripting language
- A scripting language is a lightweight programming language

- JavaScript is usually embedded directly into HTML pages JavaScript is an interpreted language (means that scripts execute without preliminary compilation)
- Everyone can use JavaScript without purchasing a license

5.1.4 Advantages of JavaScript

The merits of using JavaScript are:

- **Less server interaction:** You can validate user input before sending the page off to the server. This saves server traffic, which means fewer loads on your server.
- **Immediate feedback to the visitors:** They don't have to wait for a page reload to see if they have forgotten to enter something.
- **Increased interactivity:** You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.
- **Richer interfaces:** You can use JavaScript to include such items as drag-and-drop components and sliders to give a Rich Interface to your site visitors.

5.1.5 Limitations of JavaScript

- We cannot treat JavaScript as a full-fledged programming language. It lacks the following important features:
- Client-side JavaScript does not allow the reading or writing of files. This has been kept for security reason.
- JavaScript cannot be used for networking applications because there is no such support available.
- JavaScript doesn't have any multithreading or multiprocessor capabilities.
- Once again, JavaScript is a lightweight, interpreted programming language that allows you to build interactivity into otherwise static HTML pages.

5.1.6 Javascript – Syntax

JavaScript can be implemented using JavaScript statements that are placed within the **<script>... </script>** HTML tags in a web page.

You can place the **<script>** tags, containing your JavaScript, anywhere within you web page, but it is normally recommended that you should keep it within the **<head>** tags.
The <script> tag alerts the browser program to start interpreting all the text between these tags as a script. A simple syntax of your JavaScript will appear as follows.

```
<script ...>
  JavaScript code
</script>
```

The script tag takes two important attributes:
Language: This attribute specifies what scripting language you are using. Typically, its value will be Javascript. Although recent versions of HTML (and XHTML, its successor) have phased out the use of this attribute. **Type:** This attribute is what is now recommended to indicate the scripting language in use and its value should be set to "text/Javascript".

So your JavaScript syntax will look as follows.

```
<script language="javascript" type="text/javascript">
  JavaScript code
</script>
```

5.1.7 Your First JavaScript Code

Let us take a sample example to print out "Hello World". We added an optional HTML comment that surrounds our JavaScript code. This is to save our code from a browser that does not support JavaScript. The comment ends with a "//-->". Here "//" signifies a comment in JavaScript, so we add that to prevent a browser from reading the end of the HTML comment as a piece of JavaScript code. Next, we call a function **document. write** which writes a string into our HTML document.

This function can be used to write text, HTML, or both. Take a look at the following code.

```
<html>
<body>
<script language="javascript" type="text/javascript">
<!--
   document.write ("Hello World!")
//-->
</script>
</body>
</html>
```

This code will produce the following result:

```
Hello World!
```

5.1.8 Whitespace and Line Breaks

JavaScript ignores spaces, tabs, and newlines that appear in JavaScript programs. You can use spaces, tabs, and newlines freely in your program and you are free to format and indent your programs in a neat and consistent way that makes the code easy to read and understand.

5.1.9 Semicolons are Optional

Simple statements in JavaScript are generally followed by a semicolon character, just as they are in C, C++, and Java. JavaScript, however, allows you to omit this semicolon if each of your statements is placed on a separate line. For example, the following code could be written without semicolons.

```
<script language="javascript" type="text/javascript">
<!--
  var1 = 10
  var2 = 20
//-->
</script>
```

But when formatted in a single line as follows, you must use semicolons:

```
<script language="javascript" type="text/javascript">
<!--
  var1 = 10; var2 = 20;
//-->
</script>
```

Note: It is a good programming practice to use semicolons.

5.1.10 Case Sensitivity

JavaScript is a case-sensitive language. This means that the language keywords, variables, function names, and any other identifiers must always be typed with a consistent capitalization of letters.

So the identifiers **Time** and **TIME** will convey different meanings in JavaScript.

NOTE: Care should be taken while writing variable and function names in JavaScript.

5.1.11 Comments in JavaScript

JavaScript supports both C-style and C++-style comments. Thus:

- Any text between a // and the end of a line is treated as a comment and is ignored by JavaScript.
- Any text between the characters /* and */ is treated as a comment. This may span multiple lines.
- JavaScript also recognizes the HTML comment opening sequence <!--. JavaScript treats this as a single-line comment, just as it does the // comment.
- The HTML comment closing sequence --> is not recognized by JavaScript so it should be written as //-->.

Example

The following example shows how to use comments in JavaScript.

```
<script language="javascript" type="text/javascript">
<!--
// This is a comment. It is similar to comments in C++
/*
 * This is a multiline comment in JavaScript
 * It is very similar to comments in C Programming
 */
//-->
</script>
```

5.1.12 JavaScript in Firefox

Here are the steps to turn on or turn off JavaScript in Firefox:

- Open a new tab -> type **about: config** in the address bar.
- Then you will find the warning dialog. Select **I'll be careful, I promise!**
- Then you will find the list of **configure options** in the browser.
- In the search bar, type **javascript.enabled**.
- There you will find the option to enable or disable javascript by right-clicking on the value of that option -> **select toggle**.

If javascript.enabled is true; it converts to false upon clicking **toogle**. If javascript is disabled; it gets enabled upon clicking toggle.

5.1.13 JavaScript in Chrome

Here are the steps to turn on or turn off JavaScript in Chrome:

- Click the Chrome menu at the top right hand corner of your browser.
- Select **Settings**.
- Click **Show advanced settings** at the end of the page.
- Under the **Privacy** section, click the Content settings button.
- In the "Javascript" section, select "Do not allow any site to run JavaScript" or "Allow all sites to run JavaScript (recommended)".

Module-2

5.2.1 Interactivity in HTML

There is a flexibility given to include JavaScript code anywhere in an HTML document. However the most preferred ways to include JavaScript in an HTML file are as follows:

- Script in <head>...</head> section.
- Script in <body>...</body> section.
- Script in <body>...</body> and <head>...</head> sections.
- Script in an external file and then include in <head>...</head> section.

In the following section, we will see how we can place JavaScript in an HTML file in different ways.

5.2.2 JavaScript in <head>...</head> Section

If you want to have a script run on some event, such as when a user clicks somewhere, then you will place that script in the head as follows.

```
<html>
<head>
<script type="text/javascript">
<!--
function sayHello() {
   alert("Hello World")
}
//-->
</script>
</head>
<body>
Click here for the result
<input type="button" onclick="sayHello()" value="Say Hello" />
</body>
</html>
```

This code will produce the following results:

Click here for the result

Say Hello

5.2.3 JavaScript in <body>...</body> Section

If you need a script to run as the page loads so that the script generates content in the page, then the script goes in the <body> portion of the document. In this case, you would not have any function defined using JavaScript. Take a look at the following code.

```
<html>
<head>
</head>
<body>
<script type="text/javascript">
<!--
document.write("Hello World")
//-->
</script>
<p>This is web page body </p>
</body>
</html>
```

This code will produce the following results:

```
Hello World
This is web page body
```

5.2.4 JavaScript in <body> and <head> Sections

You can put your JavaScript code in <head> and <body> section altogether as follows.

```
<html>
<head>
<script type="text/javascript">
<!--
function sayHello() {
   alert("Hello World")
}
//-->
</script>
</head>
<body>
<script type="text/javascript">
<!--
document.write("Hello World")
//-->
</script>
<input type="button" onclick="sayHello()" value="Say Hello" />
</body>
</html>
```

This code will produce the following result.

5.2.5 Javascript – Variables

5.2.5.1 JavaScript Data types

One of the most fundamental characteristics of a programming language is the set of data types it supports. These are the type of values that can be represented and manipulated in a programming language.

JavaScript allows you to work with three primitive data types:

- Numbers, e.g., 123, 120.50 etc.
- Strings of text, e.g. "This text string" etc.
- Boolean, e.g. true or false.

JavaScript also defines two trivial data types, null and undefined, each of which defines only a single value. In addition to these primitive data types, JavaScript supports a composite data type known as object. We will cover objects in detail in a separate chapter.

Note: Java does not make a distinction between integer values and floating-point values. All numbers in JavaScript are represented as floating-point values. JavaScript represents numbers using the 64-bit floating-point format defined by the IEEE 754 standard.

5.2.5.2 JavaScript Variables

Like many other programming languages, JavaScript has variables. Variables can be thought of as named containers. You can place data into these containers and then refer to the data simply by naming the container.

Before you use a variable in a JavaScript program, you must declare it. Variables are declared with the **var** keyword as follows.

```
<script type="text/javascript">
<!--
var money;
var name;
//-->
</script>
```

You can also declare multiple variables with the same **var** keyword as follows:

```
<script type="text/javascript">
<!--
var money, name;
//-->
</script>
```

Storing a value in a variable is called **variable initialization**. You can do variable initialization at the time of variable creation or at a later point in time when you need that variable.

For instance, you might create variable named **money** and assign the value 2000.50 to it later. For another variable, you can assign a value at the time of initialization as follows.

```
<script type="text/javascript">
<!--
var name = "Ali";
var money;
money = 2000.50;
//-->
</script>
```

Note: Use the **var** keyword only for declaration or initialization, once for the life of any variable name in a document. You should not re-declare same variable twice.

JavaScript is **untyped** language. This means that a JavaScript variable can hold a value of any data type. Unlike many other languages, you don't have to tell JavaScript during variable declaration what type of value the variable will hold. The value type of a variable can change during the execution of a program and JavaScript takes care of it automatically.

5.2.5.3 JavaScript Variable Scope

The scope of a variable is the region of your program in which it is defined. JavaScript variables have only two scopes.

- **Global Variables:** A global variable has global scope which means it can be defined anywhere in your JavaScript code.
- **Local Variables:** A local variable will be visible only within a function where it is defined. Function parameters are always local to that function.

Within the body of a function, a local variable takes precedence over a global variable with the same name. If you declare a local variable or function and parameter with the same name as a global variable, you effectively hide the global variable. Take a look into the following example.

```
<script type="text/javascript">
<!--
var myVar = "global"; // Declare a global variable
function checkscope( ) {
   var myVar = "local";  // Declare a local variable
   document.write(myVar);
}
//-->
</script>
```

It will produce the following result:

```
Local
```

5.2.5.4 JavaScript Variable Names

While naming your variables in JavaScript, keep the following rules in mind.

- You should not use any of the JavaScript reserved keywords as a variable name. These keywords are mentioned in the next section. For example, **break** or **boolean** variable names are not valid.
- JavaScript variable names should not start with a numeral (0-9). They must begin with a letter or an underscore character. For example, **123test** is an invalid variable name but **_123test** is a valid one.
- JavaScript variable names are case-sensitive. For example, **Name** and **name** are two different variables.

5.2.6 Javascript – Operators

What is an Operator?

Let us take a simple expression 4 + 5 is equal to 9. Here 4 and 5 are called operands and '+' is called the operator. JavaScript supports the following types of operators.

- Arithmetic Operators
- Comparison Operators
- Logical (or Relational) Operators
- Assignment Operators
- Conditional (or ternary) Operators

Let's have a look at all the operators one by one.

5.2.6.1 JavaScript Arithmetic Operators

Arithmetic operators are used to perform arithmetic between variables and/or values.

Given that **y=5**, the table below explains the arithmetic operators:

Operator	Description	Example	Result
+	Addition	x=y+2	x=7
-	Subtraction	x=y-2	x=3
*	Multiplication	x=y*2	x=10
/	Division	x=y/2	x=2.5
%	Modulus (division remainder)	x=y%2	x=1
++	Increment	x=++y	x=6
--	Decrement	x=--y	x=4

Note: Addition operator (+) works for Numeric as well as Strings. e.g. "a" + 10 will give "a10".

```
<html>
<body>

<script type="text/javascript">
<!--
var a = 33;
var b = 10;
var c = "Test";
var linebreak = "<br />";

document.write("a + b = ");
result = a + b;
document.write(result);
document.write(linebreak);
```

```
document.write("a - b = ");
result = a - b;
document.write(result);
document.write(linebreak);
document.write("a / b = ");
result = a / b;
document.write(result);
document.write(linebreak);

document.write("a % b = ");
result = a % b;
document.write(result);
document.write(linebreak);

document.write("a + b + c = ");
result = a + b + c;
document.write(result);
document.write(linebreak);
```

```
a = a++;
document.write("a++ = ");
result = a++;
document.write(result);
document.write(linebreak);

b = b--;
document.write("b-- = ");
result = b--;
document.write(result);
document.write(linebreak);
//-->
</script>

<p>Set the variables to different values and then try...</p>
</body>
</html>
```

Output

```
a + b = 43
a - b = 23
a / b = 3.3
a % b = 3
a + b + c = 43Test
a++ = 33
b-- = 10

Set the variables to different values and then try...
```

5.2.6.2 Comparison Operators

Comparison operators are used in logical statements to determine equality or difference between Variables or values.
Given that **x=5**, the table below explains the comparison operators:

Operator	Description	Example
==	is equal to	x==8 is false
===	is exactly equal to (value and type)	x===5 is true x==="5" is false
!=	is not equal	x!=8 is true
>	is greater than	x>8 is false
<	is less than	x<8 is true
>=	is greater than or equal to	x>=8 is false
<=	is less than or equal to	x<=8 is true

5.2.6.3 Logical Operators

Logical operators are used to determine the logic between variables or values.
Given that **x=6 and y=3**, the table below explains the logical operators:

Operator	Description	Example
&&	and	(x < 10 && y > 1) is true
||	or	(x==5 || y==5) is false
!	not	!(x==y) is true

5.2.6.4 JavaScript Assignment Operators

Assignment operators are used to assign values to JavaScript variables.
Given that **x=10** and **y=5**, the table below explains the assignment operators:

Operator	Example	Same As	Result
=	x=y		x=5
+=	x+=y	x=x+y	x=15
-=	x-=y	x=x-y	x=5
*=	x*=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

5.2.6.5 Conditional Operator

JavaScript also contains a conditional operator that assigns a value to a variable based on some condition.
Syntax

```
variablename=(condition)?value1:value2
```

Example

```
greeting=(visitor=="PRES")?"Dear President ":"Dear ";
```

If the variable **visitor** has the value of "PRES", then the variable **greeting** will be assigned the value "Dear President " else it will be assigned "Dear".

Module-3

5.3.1 Conditional Statements

Very often when you write code, you want to perform different actions for different decisions. You can use conditional statements in your code to do this.

In JavaScript we have the following conditional statements:

- **if statement** - use this statement to execute some code only if a specified condition is true
- **if...else statement** - use this statement to execute some code if the condition is true and another code if the condition is false
- **if...else if....else statement** - use this statement to select one of many blocks of code to be executed
- **switch statement** - use this statement to select one of many blocks of code to be executed

5.3.1.1 If Statement

Use the if **statement** to execute some code only if a specified condition is true.

Syntax

```
if (condition)
  {
  code to be executed if condition is true
  }
```

Note that if is written in lowercase letters. Using uppercase letters (IF) will generate a JavaScript error!

Example

```
<script type="text/javascript">
//Write a "Good morning" greeting if
//the time is less than 10

var d=new Date();
var time=d.getHours();

if (time<10)
  {
  document.write("<b>Good morning</b>");
  }
</script>
```

Notice that there is no ..else.. in this syntax. You tell the browser to execute some code **only if the Specified condition is true**.

5.3.1.2 If...else Statement

Use the if....else statement to execute some code if a condition is true and another code if the condition is not true.

Syntax

```
if (condition)
  {
  code to be executed if condition is true
  }
else
  {
  code to be executed if condition is not true
  }
```

Example

```
<script type="text/javascript">
//If the time is less than 10, you will get a "Good morning" greeting.
//Otherwise you will get a "Good day" greeting.

var d = new Date();
var time = d.getHours();

if (time < 10)
  {
  document.write("Good morning!");
  }
else
  {
  document.write("Good day!");
  }
</script>
```

5.3.1.3 If...else if...else Statement

Use the if....else if...else statement to select one of several blocks of code to be executed.

Syntax

```
if (condition1)
  {
  code to be executed if condition1 is true
  }
else if (condition2)
  {
  code to be executed if condition2 is true
  }
else
  {
  code to be executed if condition1 and condition2 are not true
  }
```

Example

```
<script type="text/javascript">
var d = new Date()
var time = d.getHours()
if (time<10)
  {
  document.write("<b>Good morning</b>");
  }
else if (time>10 && time<16)
  {
  document.write("<b>Good day</b>");
  }
else
  {
  document.write("<b>Hello World!</b>");
  }
</script>
```

5.3.2 JavaScript Popup Boxes

JavaScript has three kinds of popup boxes: Alert box, Confirm box, and Prompt box.

5.3.2.1 Alert Box

An alert box is often used if you want to make sure information comes through to the user. When an alert box pops up, the user will have to click "OK" to proceed.

Syntax

```
alert("sometext");
```

Example

```
<html>
<head>
<script type="text/javascript">
function show_alert()
{
alert("I am an alert box!");
}
</script>
</head>
<body>

<input type="button" onclick="show_alert()" value="Show alert box" />

</body>
</html>
```

5.3.2.2 Confirm Box

A confirm box is often used if you want the user to verify or accept something. When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed. If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.

Syntax

```
confirm("sometext");
```

Example

```
<html>
<head>
<script type="text/javascript">
function show_confirm()
{
var r=confirm("Press a button");
if (r==true)
  {
  alert("You pressed OK!");
  }
else
  {
  alert("You pressed Cancel!");
  }
}
</script>
</head>
<body>

<input type="button" onclick="show_confirm()" value="Show confirm
box" />

</body>
</html>
```

5.3.2.3 Prompt Box

A prompt box is often used if you want the user to input a value before entering a page. When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.

If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.

Syntax

```
prompt("sometext","defaultvalue");
```

Example

```
<html>
<head>
<script type="text/javascript">
function show_prompt()
{
var name=prompt("Please enter your name","Harry Potter");
if (name!=null && name!="")
  {
  document.write("Hello " + name + "! How are you today?");
  }
}
</script>
</head>
<body>

<input type="button" onclick="show_prompt()" value="Show prompt box" />

</body>
</html>
```

5.3.3 JavaScript Functions

A function will be executed by an event or by a call to the function.

To keep the browser from executing a script when the page loads, you can put your script into a function.

A function contains code that will be executed by an event or by a call to the function.

You may call a function from anywhere within a page (or even from other pages if the function is embedded in an external .js file).

Functions can be defined both in the <head> and in the <body> section of a document. However, to assure that a function is read/loaded by the browser before it is called, it could be wise to put functions in the <head> section.

How to Define a Function

Syntax

```
function functionname(var1,var2,...,varX)
{
some code
}
```

The parameters var1, var2, etc. are variables or values passed into the function. The { and the } defines the start and end of the function.

Note: A function with no parameters must include the parentheses () after the function name.

Note: Do not forget about the importance of capitals in JavaScript! The word function must be written in lowercase letters, otherwise a JavaScript error occurs! Also note that you must call a function with the exact same capitals as in the function name.

JavaScript Function Example

Example

```
<html>
<head>
<script type="text/javascript">
function displaymessage()
{
alert("Hello World!");
}
</script>
</head>

<body>
<form>
<input type="button" value="Click me!" onclick="displaymessage()" />
</form>
</body>
</html>
```

If the line: alert("Hello world!!") in the example above had not been put within a function, it would have been executed as soon as the line was loaded. Now, the script is not executed before a user hits the input button. The function displaymessage() will be executed if the input button is clicked.

The return Statement

The return statement is used to specify the value that is returned from the function. So, functions that are going to return a value must use the return statement.

The example below returns the product of two numbers (a and b):

Example

```
<html>
<head>
<script type="text/javascript">
function product(a,b)
{
return a*b;
}
</script>
</head>

<body>
<script type="text/javascript">
document.write(product(4,3));
</script>

</body>
</html>
```

5.3.4 JavaScript While Loop

Loops execute a block of code a specified number of times, or while a specified condition is true.

The while loop loops through a block of code while a specified condition is true.

Syntax

```
while (var<=endvalue)
  {
  code to be executed
  }
```

Note: The <= could be any comparing statement.

Example

The example below defines a loop that starts with i=0. The loop will continue to run as long as **i** is less than, or equal to 5. **i** will increase by 1 each time the loop runs:

Example

```
<html>
<body>
<script type="text/javascript">
var i=0;
while (i<=5)
  {
  document.write("The number is " + i);
  document.write("<br />");
  i++;
  }
</script>
</body>
</html>
```

5.3.5 JavaScript Events

By using JavaScript, we have the ability to create dynamic web pages. Events are actions that can be detected by JavaScript.

Every element on a web page has certain events which can trigger a JavaScript. For example, we can use the onClick event of a button element to indicate that a function will run when a user clicks on the button. We define the events in the HTML tags.

Examples of events:

- A mouse click
- A web page or an image loading
- Mousing over a hot spot on the web page
- Selecting an input field in an HTML form
- Submitting an HTML form
- A keystroke

Note: Events are normally used in combination with functions, and the function will not be executed before the event occurs!

You can use JavaScript to create a complex animation having, but not limited to, the following elements

- Fireworks, Fade Effect, Roll-in or Roll-out, Page-in or Page-out and Object movements

You might be interested in existing JavaScript based animation library:

This module provides a basic understanding of how to use JavaScript to create an animation.

JavaScript can be used to move a number of DOM elements (<img />, or any other HTML element) around the page according to some sort of pattern determined by a logical equation or function.

JavaScript provides the following two functions to be frequently used in animation programs.

- **setTimeout(function, duration)** − This function calls **function** after **duration** milliseconds from now.
- **setInterval(function, duration)** − This function calls **function** after every **duration** milliseconds.
- **clearTimeout(setTimeout_variable)** − This function calls clears any timer set by the setTimeout() functions.

JavaScript can also set a number of attributes of a DOM object including its position on the screen. You can set *top* and left attribute of an object to position it anywhere on the screen. Here is its syntax.

// Set distance from left edge of the screen.

object.style.left = distance in pixels or points;

or
// Set distance from top edge of the screen.
object.style.top = distance in pixels or points;
Manual Animation
So let's implement one simple animation using DOM object properties and JavaScript functions as follows. The following list contains different DOM methods.

- We are using the JavaScript function **getElementById()** to get a DOM object and then assigning it to a global variable **imgObj**.
- We have defined an initialization function **init()** to initialize **imgObj** where we have set its **position** and **left** attributes.
- We are calling initialization function at the time of window load.
- Finally, we are calling **moveRight()** function to increase the left distance by 10 pixels. You could also set it to a negative value to move it to the left side.

Example:

```
<html><head>
<title>JavaScript Animation</title>
<script type = "text/javascript">
<!--
varimgObj = null;
function init()
{
imgObj = document.getElementById('myImage');
imgObj.style.position= 'relative';
imgObj.style.left = '0px';
}
function moveRight()
 {
imgObj.style.left = parseInt(imgObj.style.left) + 10 + 'px';
}
window.onload = init;
//-->
</script>
</head>
<body>
<form>
<img id = "myImage" src = "/images/html.gif" />
<p>Click button below to move the image to right</p>
<input type = "button" value = "Click Me" onclick = "moveRight();" />
</form>
</body></html>
```

Output: Automated Animation
In the above example, we saw how an image moves to right with every click. We can automate this process by using the JavaScript function **setTimeout()** as follows −
Here we have added more methods. So let's see what is new here −

- The **moveRight()** function is calling **setTimeout()** function to set the position of *imgObj*.

- We have added a new function **stop()** to clear the timer set by **setTimeout()** function and to set the object at its initial position.

Multiple Choice Questions

1) Which type of JavaScript language is ___

a) Object-Oriented
b) Object-Based
c) Assembly-language
d) High-level

2) Which of the following is the correct output for the following JavaScript code:

```
varx=5,y=1
var obj ={ x:10}
with(obj)
{
    alert(y)
}
```

a) 1
b) Error
c) 10
d) 5

3) In JavaScript, what is a block of statement?

a. Conditional block
b. block that combines a number of statements into a single compound statement
c. both conditional block and a single statement
d. block that contains a single statement

4) The "function" and " var" are known as:

```
var x=3;
var y=2;
var z=0;
If(x==y)
document.write(x);
elseif(x==y)
document.write(x);
else
document.write(z);
```

a. Keywords
b. Data types
c. Declaration statements
d. Prototypes

5) Which of the following is the correct output for the following JavaScript code

a. 3
b. 0
c. Error
d.2

Descriptive Questions

1) What is client side java script and server side java script?
2) Write java script history and features of it?
3) What are the limitations of java script
4) Write any simple java script in html
5) Write java script example for if, if..else and if..else if
6) Write any 3 basic effects using java script

Unit 6:: Basics of Algorithms and Flow charts

Objectives:

- Students will come to Know Introduction to Problem solving and algorithms
- Students can understand about the algorithms
- Students will cover the : Qualities of good algorithms
- Students will be able to know about the Advantages and Disadvantages of Algorithm,
- Properties and efficiency of an algorithm and Tracing of an algorithm
- Types of Control Structures and Nested Control Structures in Algorithms

Module 1
Introduction to problem solving and algorithms

6.1 Introduction:

Intelligence is one of the key characteristics which differentiate a human being from other living creatures on the earth. Basic intelligence covers day to day problem solving and making strategies to handle different situations which keep arising in day to day life. One person goes Bank to withdraw money. After knowing the balance in his account, he/she decides to with draw the entire amount from his account but he/she has to leave minimum balance in his account. Here deciding about how much amount he/she may with draw from the account is one of the examples of the basic intelligence. During the process of solving any problem, one tries to find the necessary steps to be taken in a sequence. In this Unit you will develop your understanding about problem solving and approaches.

6.1.1 Problem Solving:

Can you think of a day in your life which goes without problem solving? Answer to this question is of course, No. In our life we are bound to solve problems. In our day to day activity such as purchasing something from a general store and making payments, depositing fee in school, or withdrawing money from bank account. All these activities involve some kind of problem solving. It can be said that whatever activity a human being or machine do for achieving a specified objective comes under problem solving. To make it clearer, let us see some other examples.

Example1: If you are watching a news channel on your TV and you want to change it to a sports channel, you need to do something i.e. move to that channel by pressing that channel number on your remote. This is a kind of problem solving.

Example 2: One Monday morning, a student is ready to go to school but yet he/she has not picked up those books and copies which are required as per timetable. So here picking up books and copies as per timetable is a kind of problem solving.

Example 3: If someone asks to you, what is time now? So seeing time in your watch and telling him is also a kind of problem solving.

Example 4: Some students in a class plan to go on picnic and decide to share the expenses among them. So calculating total expenses and the amount an individual have to give for picnic is also a kind of problem solving.

- Now, we can say that problem is a kind of barrier to achieve something and problem solving is a process to get that barrier removed by performing some sequence of activities.
- Here it is necessary to mention that all the problems in the world cannot be solved. There are some problems which have no solution and these problems are called Open Problems.

- If you can solve a given problem then you can also write an algorithm for it. In next section we will learn what is an algorithm?

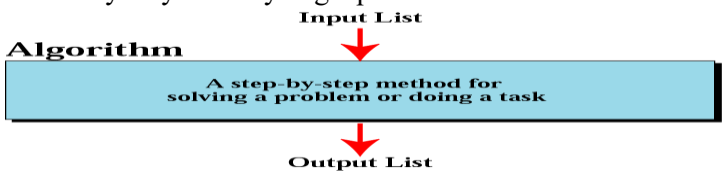
ALGORITHM

6.1.2 Introduction:

Algorithm is a step-by-step process of solving a well-defined computational problem. In practice, in order to solve any complex real life problems, first we have to define the problem and then, design algorithm to solve it. Writing and executing a simple program may be easy; however, for executing a bigger one, each part of the program must be well organized. In short, algorithms are used to simplify the program implementation. The study of algorithms is one of the key foundations of computer science. Algorithms are essential to the way computers process information, because a computer program is essentially an algorithm that tells the computer what specific steps to perform (in what specific order) in order to carry out a specified task, such as calculating employees' paychecks or printing students' report cards. Thus , an algorithm can be considered to be any sequence of operations that can be performed by a computing device such as a computer.

6.1.3 What is an Algorithm?

- Algorithm is a step by step procedure to solve a particular problem or "A sequence of activities to be processed for getting desired output from a given input.
- It's a easy way of analyzing a problem.



6.1.4 Characteristics of Algorithm:

1) **Input:** An algorithm has zero or more inputs, taken from a specified set of objects.
2) **Output:** An algorithm has one or more outputs, which have a specified relation to the inputs.
3) **Definiteness:** Each step must be precisely defined; the actions to be carried out must be rigorously and unambiguously specified for each case.
4) **Effectiveness:** All operations to be performed must be sufficiently basic that they can be done exactly and in finite length.
5) **Finiteness or Termination:** The algorithm must always terminate after a finite number of steps.

"A formula or set of steps for solving a particular problem. To be an algorithm, a set of rules must be unambiguous and have a clear stopping point". There may be more than one way to solve a problem, so there may be more than one algorithm for a problem. Now, if we take definition of algorithm as: "A sequence of activities to be processed for getting desired output from a given input." Then we can say that:

1. Getting specified output is essential after algorithm is executed.
2. One will get output only if algorithm stops after finite time.
3. Activities in an algorithm to be clearly defined in other words for it to be unambiguous.

6.1.5 Structure of an algorithm:

START

- **STEP 1**
- **SETP 2**
- **SETP 3**

- **SETP 4**
- **……so on**

STOP

Example 1: A simple algorithm to print „Good Morning".

Step 1: Start
Step 2: Print „Good Morning"
Step 3: Stop

Example 2: Let us take one simple day-to-day example by writing algorithm for making "Maggi Noodles" as a food.

Step 1: Start
Step 2: Take pan with water
Step 3: Put pan on the burner
Step 4: Switch on the gas/burner
Step 5: Put magi and masala
Step 6: Give two minutes to boil
Step 7: Take off the pan
Step 8: Take out the magi with the help of fork/spoon
Step 9: Put the maggi on the plate and serve it
Step 10: Stop

Example3: Find the area of a Circle of radius r.

Inputs to the algorithm: Radius r of the Circle.
Expected output: Area of the Circle
Algorithm:
Step1: Read\input the Radius r of the Circle
Step2: Area PI*r*r // calculation of area
Step3: Print Area
Step4: stop

Example4: Write an algorithm to read two numbers and find their sum.

Inputs to the algorithm: First num1. Second num2.
Expected output: Sum of the two numbers.
Algorithm:

Step1: Start
Step2: Read\input the first num1.
Step3: Read\input the second num2.
Step4: Sum num1+num2 // calculation of sum
Step5: Print Sum
Step6: End

6.1.6 Expressing Algorithms:

An algorithm is a set of steps designed to solve a problem or accomplish a task. Algorithms are usually written in pseudocode, or a combination of your speaking language and one or more programming languages, in advance of writing a program**.** An algorithm may be expressed in a number of ways, including:

1. **Natural language:** usually verbose and ambiguous.
2. **Flow charts:** avoid most (if not all) issues of ambiguity; difficult to modify w/o specialized tools; largely standardized.
3. **Pseudo – code:** also avoids most issues of ambiguity; vaguely resembles common elements of programming languages; no particular agreement on syntax.

4. **Programming language:** Tend to require expressing low – level details that are not necessary for a high – level understanding.

6.1.7 Qualities of a good algorithm:

1) Inputs and outputs should be defined precisely.
2) A good algorithm should produce correct and accurate results for any set of legal or correct inputs.
3) Each step in algorithm should be clear and unambiguous.
4) Algorithm should be most effective among many different ways to solve a problem.
5) An algorithm shouldn't have computer code. Instead, the algorithm should be written in such a way that it can be used in similar programming languages whenever we write an algorithm, we should make sure that all of these parameters given for a good algorithm are incorporated.

6.1.8 Advantage and disadvantages of algorithm:

1) It is a step-wise representation of a solution to a given problem, which makes it easy to understand.
2) An algorithm uses a definite procedure.
3) It is not dependent on any programming language, so it is easy to understand for anyone even without programming knowledge.
4) Every step in an algorithm has its own logical sequence so it is easy to debug.
5) By using algorithm, the problem is broken down into smaller pieces or steps hence, it is easier for programmer to convert it into an actual program

6.1.9 Disadvantages of algorithm:

1) Writing algorithm takes a long time.
2) An Algorithm is not a computer program, it is rather a concept of how a program should be
3) Big tasks are difficult to put in Algorithms.
4) Difficult to show Branching and Looping in Algorithms.

6.2.0 Properties of algorithm:

Donald Ervin Knuth has given a list of five properties for a algorithm, these properties are:

1) **Finiteness:** An algorithm must always terminate after a finite number of steps. It means after every step one reach closer to solution of the problem and after a finite number of steps algorithm reaches to an end point.
2) **Definiteness:** Each step of an algorithm must be precisely defined. It is done by well thought actions to be performed at each step of the algorithm. Also the actions are defined unambiguously for each activity in the algorithm.
3) **Input:** Any operation you perform need some beginning value/quantities associated with different activities in the operation. So the value/quantities are given to the algorithm before it begins.
4) **Output:** One always expects output/result (expected value/quantities) in terms of output from an algorithm. The result may be obtained at different stages of the algorithm. If some result is from the intermediate stage of the operation then it is known as intermediate result and result obtained at the end of algorithm is known as end result. The output is expected value/quantities always have a specified relation to the inputs
5) **Effectiveness:** Algorithms to be developed/written using basic operations. Actually operations should be basic, so that even they can in principle be done exactly and in a finite amount of time by a person, by using paper and pencil only.

6.2.1 Algorithm Efficiency

Algorithmic efficiency is a property of an algorithm which relates to the number of computational resources used by the algorithm. An algorithm must be analyzed to determine its resource usage, and the efficiency of an algorithm can be measured based on usage of different resources.

- Speed - Method that takes the least time
- Space - Method that uses the least memory
- Code — Method that is shortest to describe

Speed is now the most important factor Example

A real world example: Travelling Vempalli to Hyderabad in Vehicle

Case 1:

1. Take vehicle
2. Check vehicle condition if not get it repair
3. Drive from vempalli to Proddutur
4. Then Proddutur to Hyderabad
5. Travel is successfully completed.

Case 2:

1. Take vehicle
2. Check vehicle condition if not get it repair
3. Drive from Vempalli to kadapa
4. Then Kadapa to Kurnool
5. Then Kurnool to Hyderabad
6. Travel is successfully completed

While comparing both cases Algorithm executed successfully with same output But in case 1 travelling time is lesser than the case 2 travelling time.

6.2.2 Tracing an Algorithm:

Tracing of an Algorithm means showing how the value of a variable changes during the execution of an algorithm.

6.2.3 Steps in tracing:

1) Identify the variables in the algorithm that need to be traced.
2) Examine the value of the variables at each step in the execution of the algorithm.
3) Determine if the algorithm is giving the correct outputs for a set of legitimate/legal input values.
4) Analyze and determine what the purpose of the algorithm

In this module we will see examples of computational algorithms and the techniques used for tracing algorithms. A computational algorithm is a set of precise instructions expected to be expressed as a computer program that can execute on a computer.

Let us now get started with an example of adding two numbers.

Example Algorithm: Add two numbers

1. Input x
2. Input y
3. Set z to x+ y
4. Output z

If you notice the algorithm it takes two inputs, performs an addition operation and gives the output. Let us see how the algorithm executes step by step.

Input x /* Takes input from the user by using input devices like keyboard. After entering input from the key board, for ex: 5 the value is stored in a memory location. The name of this memory location is x in this particular case. Any time this variable x can store only one value. For ex: if you set x value 6 you will find the new value is referred by x is 6 but not 5 that means the old value which was stored in

x values is 0++++. This is a very good example that shows a variable can store only one value at a time. Input value :5 */
Input y /* Similar to previous statement where new memory location is created for variable y and the user inputs to y is stored. Input value :2 */
Set z to x+ y /* in this statement actual operation that is performed on the input values which is to add the numbers referred to by the variables x and y. In this example you will notice after the addition operation is executed and the values 5 and 2 are added and set to new variable z. */
Output z //the result value i.e. z is displayed to an output device like monitor?
You will notice that algorithm also can be visualized as functions. A function performs a specific operation like addition and multiplication or it could be more complex operations like finding square roots or sorting list of numbers.
Variable:

- Variables are used to handle the data by the algorithm.
- Variables refer to a memory location where the actual value is stored.
- Variable can store one value at a time.
- Old values of the variable are lost when a new value is assigned.

6.2.4 Control structure:

Control Structures are just a way to specify flow of control in programs. Any algorithm or program can be more clear and understood if they use self-contained modules called as logic or control structures. It basically analyzes and chooses in which direction a program flows based on certain parameters or conditions.

6.2.5 Types of control structures:

There are three types of control structures

1. Sequence
2. Selection or Branching
3. Loop or Repetition

1. Sequence Control Structure:

This refers to the line – by – line execution, in which statements are executed sequentially, in the same order in which they appear in the script. They might, for example, carry out a series of read or write operations, arithmetic operations, or assignments to variables.

Example problem:

Write algorithm to find the greater number between two numbers
Step1: Start
Step2: Read/input A and B
Step3: If A greater than B then C=A
Step4: if B greater than A then C=B
Step5: Print C
Step6: End

2. Selection or Decision Control Structure:

The branch refers to a binary decision based on some condition. Depending on whether a condition is true or false. If the condition is true, one of the two branches is explored; if the condition is false, the other alternative is taken. This is usually represented by the 'if-then' construct in pseudo-codes and programs.. This structure is also known as the selection structure

Example problem:

A person whose age is more than or equals to 18 years is eligible to vote. Now write an algorithm to check whether he is eligible to vote?

Step 1: Start
Step2: Declare the variable age // Declaration of variable
Step 3: Take age and store it in age // Accepting input.
Step 4: Check if age >= 18 then go to step 5
else go to step 6 //Decision Making
Step 5: Print "Eligible to vote" and go to step 7 // Output
Step 6: Print "Not eligible to vote" //Output
Step 7: Stop

3. Repetition or Loop Control Structure:

This is a control structure that allows the execution of a block of statements multiple times until a specified condition is met. The loop allows a statement or a sequence of statements to be repeatedly executed based on some loop condition. It is represented by the 'while' and 'for' constructs in most programming languages, for unbounded loops and bounded loops respectively. (Unbounded loops refer to those whose number of iterations depends on the eventuality that the termination condition is satisfied; bounded loops refer to those whose number of iterations is known before-hand.) In the flowcharts, a back arrow hints the presence of a loop. A trip around the loop is known as iteration. You must ensure that the condition for the termination of the looping must be satisfied after some finite number of iterations, otherwise it ends up as an infinite loop, a common mistake made by inexperienced programmers. The loop is also known as the repetition structure.

Example problem:

Algorithm to print all natural numbers upto "n" (Here you need to accept a number from the user, and to print all the natural numbers till 'n' i.e. 1 to n)
Step 1: Start
Step2: Declare the variable n // Declaration of variable
Step 3: Take any number and store it in n. // accepting input from user
Step 4: Store 1 in "I" //I=1
Step 5: Check I value, if I<=n then go to step 6
else go to step 9
Step 6: Print I // output value of I
Step 7: Increment I value by 1 //I = I +1
Step 8: Go to step 5
Step 9: Stop
//In the above example, steps 4, 5, 6 and 7 are executed more than one time

4. Nested control structures:

Combining the use of these control structures, for example, a loop within a loop (nested loops), a branch within another branch (nested if), a branch within a loop, a loop within a branch, and so forth, is not uncommon. Complex algorithms may have more complicated logic structure and deep level of nesting, in which case it is best to demarcate parts of the algorithm as separate smaller modules. Beginners must train themselves to be proficient in using and combining control structures appropriately, and go through the trouble of tracing through the algorithm before they convert it into code

Module 2

Introduction to Flow chart in programming

Objectives:

- Students will come to Know Introduction to flowchart
- Students can understand about the symbols and its use
- Students will be able to know about the Advantages and Disadvantages of Flowchart
- Types of Control Structures and Nested Control Structures in Flowcharts
- Difference between algorithm and flowchart & Pseudo code.

6.2 Flowchart:

The diagrammatic representation of an algorithm is called “Flowchart”. Before you start coding a program it is necessary to plan the step by step solution to the task your program will carry out. Such a plan can be symbolically developed using a diagram. This diagram is then called a flowchart. Hence a flowchart is a symbolic representation of a solution to a given task. A flowchart can be developed for practically any job. Flowcharting is a tool that can help us to develop and represent graphically program logic sequence.

For example suppose you are going for a picnic with your friends then you plan for the activities you will do there. If you have a plan of activities then you know clearly when you will do what activity. Similarly when you have a problem to solve using computer or in other word you need to write a computer program for a problem then it will be good to draw a flowchart prior to writing a computer program. Flowchart is drawn according to defined rules.

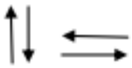
6.2.1 Features of flowchart:

The features of a flowchart are: It is an easy method of communication.

1) It is independent of a programming language.
2) It is the key to correct programming.
3) It helps to plan and design a new system.
4) It clearly indicates the task to be performed at each level.

6.2.2 Flowchart Symbols:

There are 6 basic symbols commonly used in flowcharting of assembly language Programs: Terminal, Process, and input/output, Decision, Connector and Predefined Process. This is not a complete list of all the possible flowcharting symbols; it is the ones used most often in the structure of Assembly language programming.

Symbol	Name	Function
	**Process**	Indicates any type of internal operation inside the Processor or Memory
	input/output	Used for any Input / Output (I/O) operation. Indicates that the computer is to obtain data or output results
	Decision	Used to ask a question that can be answered in a binary format (Yes/No, True/False)
	Connector	Allows the flowchart to be drawn without intersecting lines or without a reverse flow.
	Predefined Process	Used to invoke a subroutine or an Interrupt program.
	Terminal	Indicates the starting or ending of the program, process, or interrupt program
	Flow Lines	Shows direction of flow.

1. **Process Symbol:**
   - A process symbol is used to represent arithmetic and data movement instructions in the flowchart. All arithmetic processes of addition, subtraction, multiplication and division are indicated in the process symbol.
   - The logical process of data movement from one memory location to another is also represented in the process box. If there are more than one process
2. **Input/ Output Symbol:**
   - This symbol is used to denote any input/output function in the program. Thus if there is any input to the program via an input device, like a keyboard, tape, card reader etc. it will be indicated in the flowchart with the help of the Input/output symbol.
   - Similarly, all output instructions, for output to devices like printers, plotters, magnetic tapes, disk, monitors etc. are indicated in the Input/ Output symbol.
3. **Decision Symbol:**
   - The decision symbol is used in a flowchart to indicate the point where a decision is to be made and branching done upon the result of the decision to one or more alternative paths. The criteria for decision making are written in the decision box.
   - All the possible paths should be accounted for. During execution, the appropriate path will be followed depending upon the result of the decision.
4. **Connectors:**

   This symbol shows continuation of the flow chart from one page to another. When you reach the bottom of the page or need to jump to another page, draw flow chart connector symbol and connect it to the last item on the chart. Label the inside of the symbol with a letter, typically beginning with an "A".

5. **Predefined process:**
   This symbol indicates a sequence of actions that perform a specific task embedded within a larger process.
6. **Terminal Symbol:**
   - Every flowchart has a unique starting point and an ending point.
   - The flowchart begins at the start terminator and ends at the stop terminator.
   - The Starting Point is indicated with the word START inside the terminator symbol. The Ending Point is indicated with the word STOP inside the terminator symbol. There can be only one START and one STOP terminator in your entire flowchart..
7. **Flow lines:**
   - Flow lines are solid lines with arrowheads which indicate the flow of operation. They show the exact sequence in which the instructions are to be executed. The normal flow of the flowchart is depicted from top to bottom and left to right.

6.2.3 General Rules for flowcharting:

1) All boxes of the flowchart are connected with Arrows. (Not lines)
2) Flowchart symbols have an entry point on the top of the symbol with no other entry points. The exit point for all flowchart symbols is on the bottom except for the Decision symbol.
3) The Decision symbol has two exit points; these can be on the sides or the bottom and one side.
4) Generally a flowchart will flow from top to bottom. However, an upward flow can be shown as long as it does not exceed 3 symbols.
5) Connectors are used to connect breaks in the flowchart. Examples are:
   a. From one page to another page.
   b. From the bottom of the page to the top of the same page.
   c. An upward flow of more than 3 symbols
6) Subroutines and Interrupt programs have their own and independent flowcharts.
7) All flow charts start with a Terminal or Predefined Process (for interrupt programs or subroutines) symbol.
8) All flowcharts end with a terminal or a contentious loop.

Flowcharting uses symbols that have been in use for a number of years to represent the type of operations and/or processes being performed. The standardized format provides a common method for people to visualize problems together in the same manner. The use of standardized symbols makes the flow charts easier to interpret; however, standardizing symbols is not as important as the sequence of activities that make up the process.

6.2.4 Advantages of using flowchart:

As we discussed flow chart is used for representing algorithm in pictorial form. This pictorial representation of a solution/system is having many advantages. These advantages are as follows:

1) **Communication:** A Flowchart can be used as a better way of communication of the logic of a System and steps involve in the solution, to all concerned particularly to the client of system.
2) **Effective analysis:** A flowchart of a problem can be used for effective analysis of the problem.
3) **Documentation of Program/System:** Program flowcharts are a vital part of a good program documentation. Program document is used for various purposes like knowing the components in the program, complexity of the program etc.
4) **Efficient Program Maintenance:** Once a program is developed and becomes operational it needs time to time maintenance. With help of flowchart maintenance become easier.
5) **Coding of the Program:** Any design of solution of a problem is finally converted into computer

program. Writing code referring the flowchart of the solution become easy.

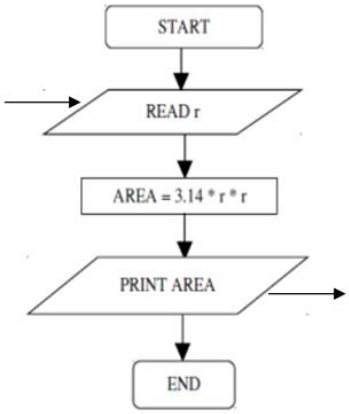
6.2.5 Disadvantages of Flowchart:

1) Drawing flowchart is a time-consuming task.
2) Manual tracing is needed to check correctness of flowchart drawn on paper.
3) Simple modification in problem logic may leads to complete redraw of flowchart.
4) Showing many branches and looping in **flowchart** is difficult.
5) In case of complex program/algorithm, **flowchart** becomes **very complex** and **clumpsy.**

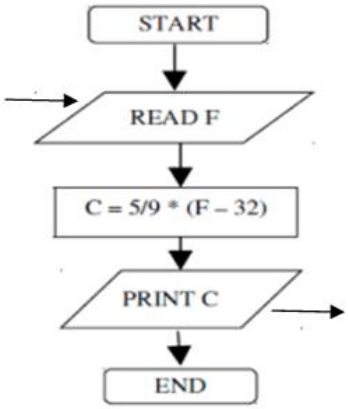
6.2.6 Some examples of Flowcharts:

Now, we will discuss some examples on flowcharting. These examples will help in proper understanding of flowcharting technique. This will help you in program development process in next unit of this block.

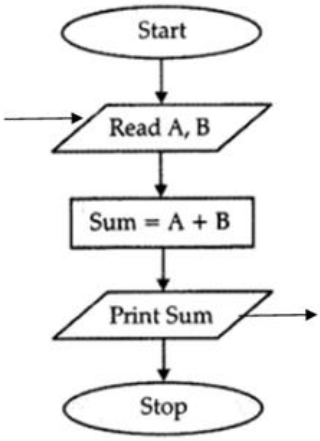
Problem1: Find the area of a circle of radius r



Problem 2: Convert temperature Fahrenheit to Celsius.



Problem3: Flowchart for an algorithm which gets two numbers and prints sum of their value.



6.2.7 Types of control structures:

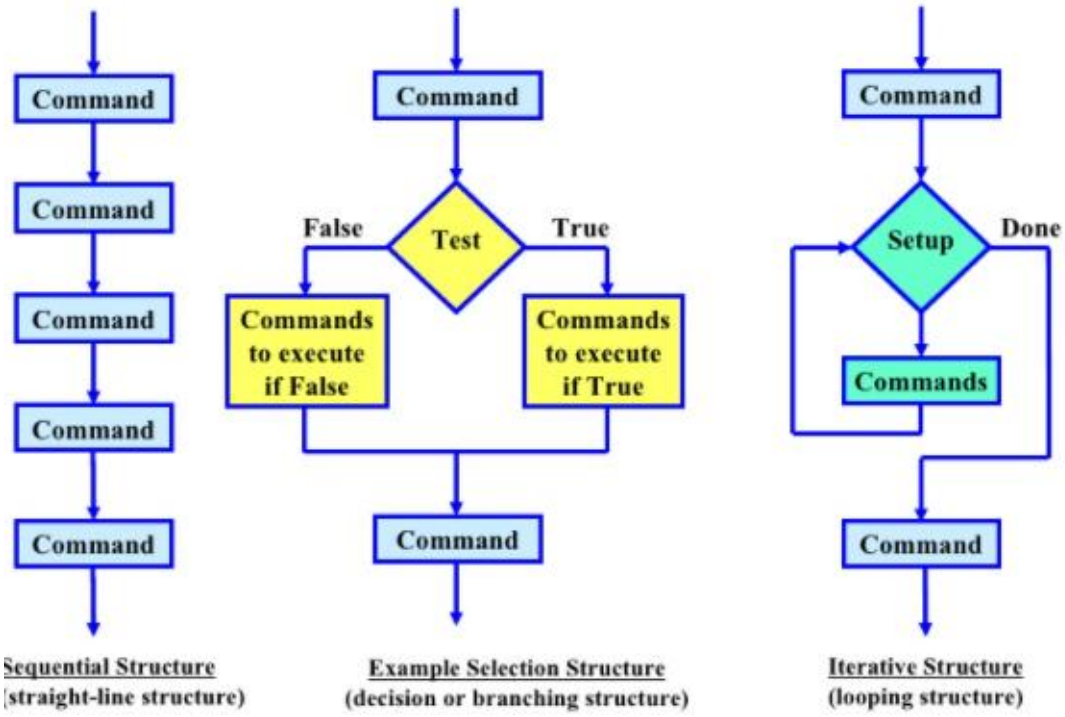
6.2.7.1 Definition:

- ❖ The logic of a program may not always be a linear sequence of statements to be executed in that order.
- ❖ The logic of the program may require execution of a statement based on a decision.
- ❖ Control structures specify the statements to be executed and the order of execution of statements.

6.2.7.2 Where is the usage of control structure?

Flowchart and Pseudo Code use Control structures for representation

Flowcharts for sequential, selection, and iterative control structures



6.2.7.3 There are three kind of Control Structure:

1) Sequential
2) Selection(Branch or conditional)
3) Iterative(loop)

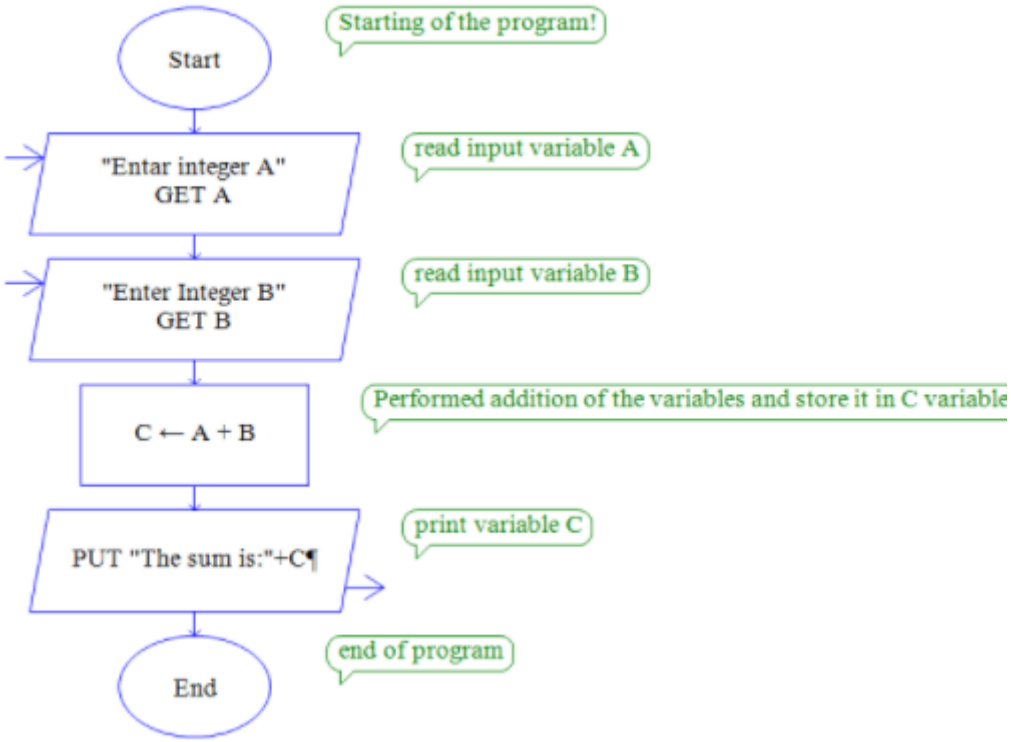
1). **Sequential:**

- ❖ Instructions are executed in linear order.
- ❖ Statements are executed in a specified order. No statement is skipped and no statement is executed more than once.

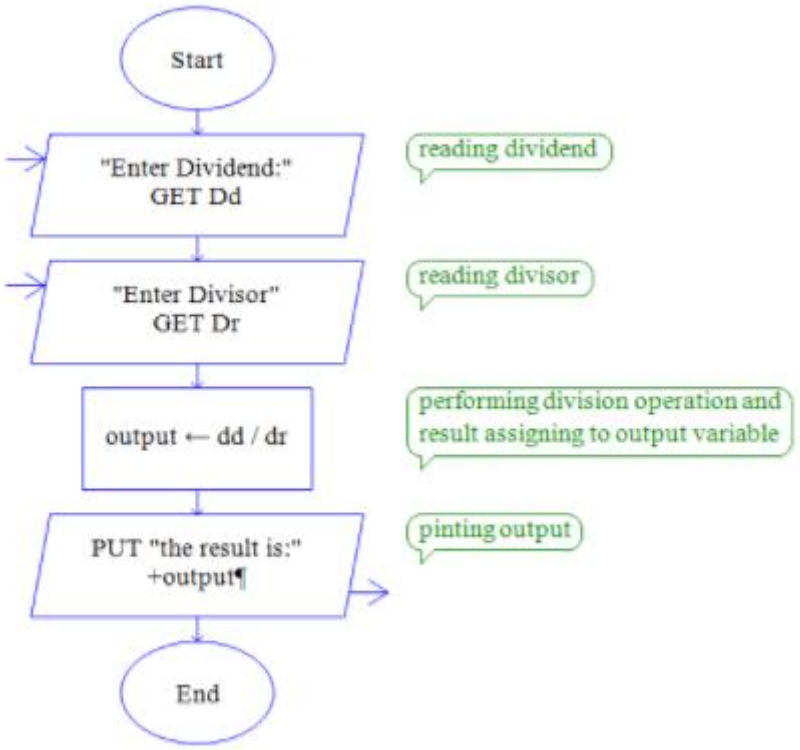
Sequential Structure

Example problem:

1. Take two numbers from the user print their sum?



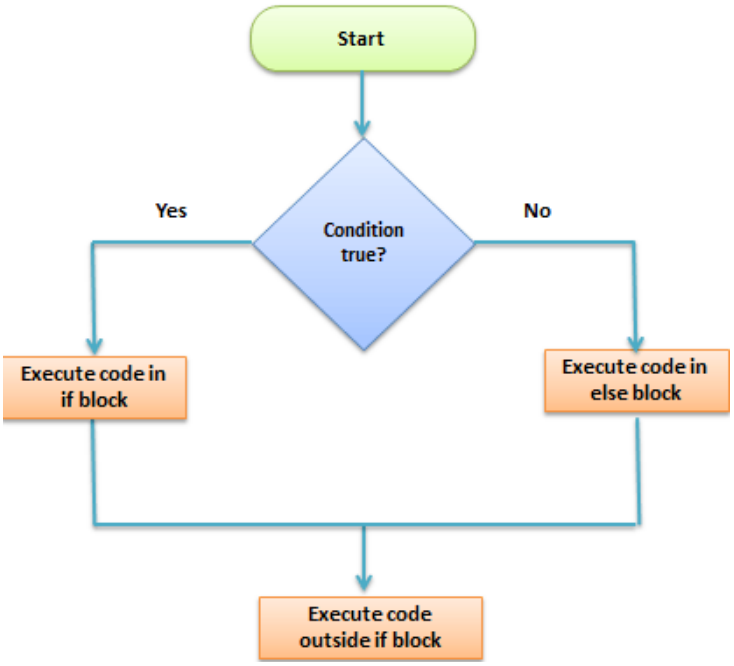
2. Take input of two numbers from the user and do the division of two numbers?



2) Selection (Branch or conditional):

1) It's ask a true/false question THEN select the next instruction based on the answer.
2) It selects a statement to execute on the basis of **condition**. Statement is executed when the condition is true and ignored when it is false

Example: if, if else, switch structures.



Example problem:

1. Draw a flowchart to determine greatest among two numbers?

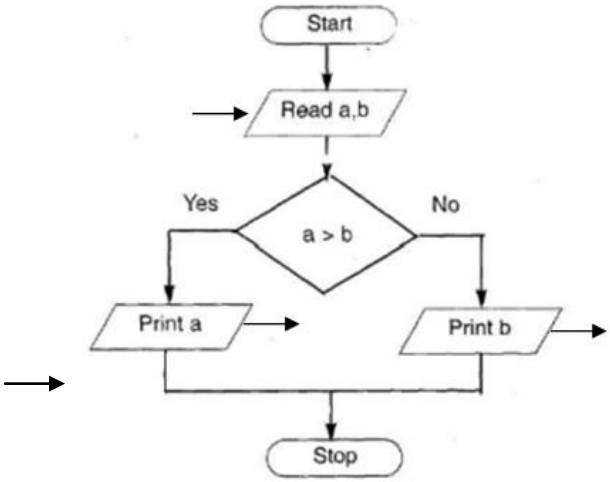


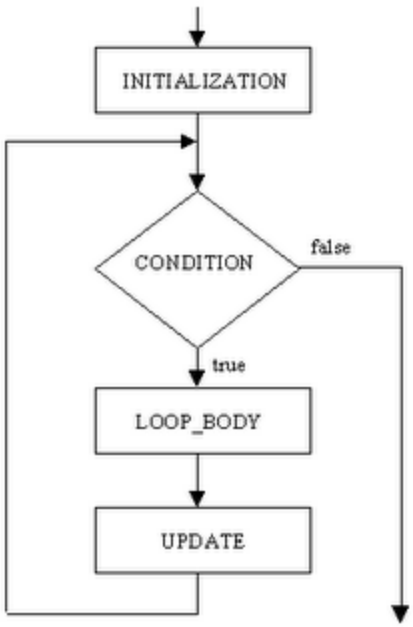
Fig. 2b) Flowchart to determine the greater of two numbers a and b

3) **Iterative (loop)**:

- It's repeat the execution of a block of instruction.
- In this structure the statements are executed more than one time. It is also known as iteration or loop
- A loop structure is used to execute a certain set of actions for a predefined number of times or until a particular condition is satisfied

Example: while loop, for loop do-while loops etc.

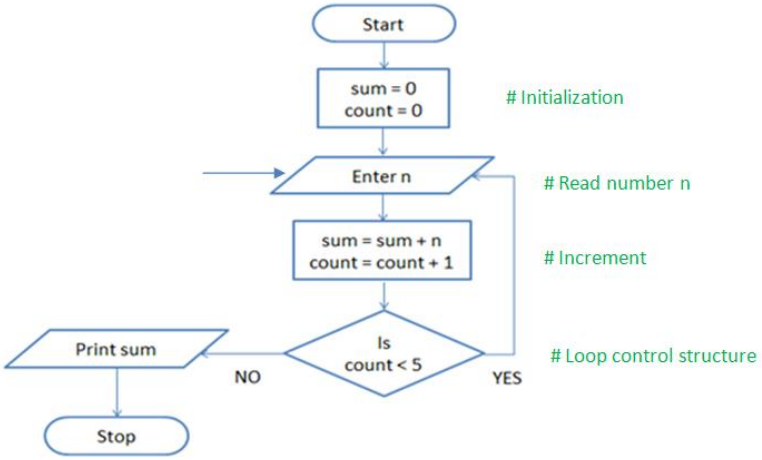
Iterative Structure
(looping structure)



Iterative Structure

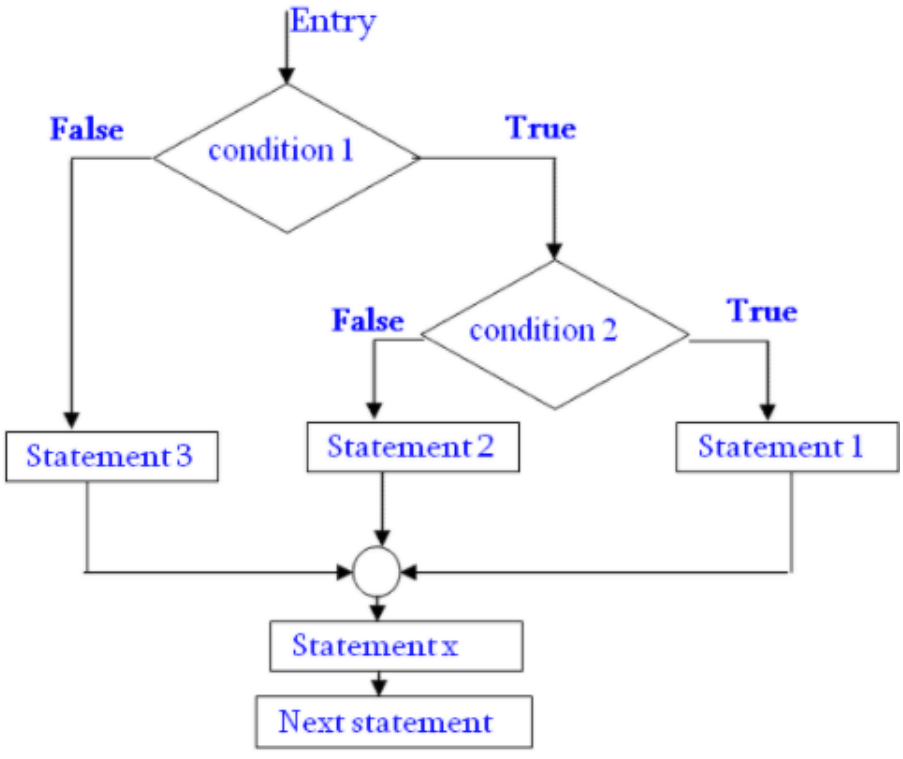
Example problem:

1. Find the sum of 5 numbers:



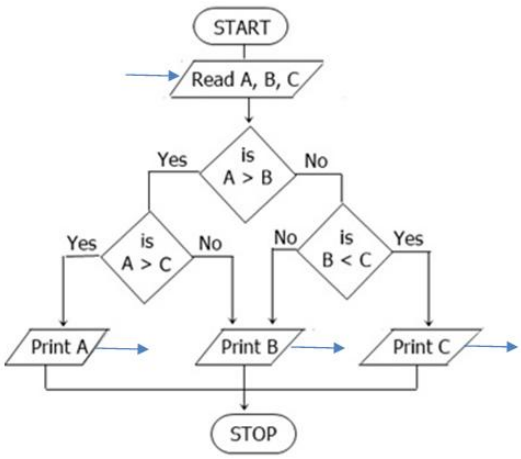
4) Nested Control Structures:

A nested control statement is a control statement that is contained within another control statement. You can do this to many levels. You can place control statements inside other control statements, for example an If...Then...Else block within a For...Next loop. A control statement placed inside anothecontrol statement is said to be ***nested.***



Example problem:

1. Draw a flowchart to determine largest among three numbers?



Solved problems

1. **Write an algorithm to go for class picnic.**

   Algorithm:

   Step 1: Start
   Step 2: Decide the picnic venue, date and time
   Step 3: Decide the picnic activities
   Step 4: Hire a vehicle to reach to the venue and comeback
   Step 5: Goto to the picnic venue on the decided date
   Step 6: Do the activities planned for the picnic
   Step 7: Come back to school in the hired vehicle
   Step 8: Stop

2. **To celebrate Teachers' Day**

   Algorithm:

   Step 1: Start
   Step 2: Decide the activities for teachers' day like dance performances, plays, etc.
   Step 3: Form groups of students and assign the decided activities from step 2 to each group.
   Step 4: Decide the practice timings for each group.
   Step 5: Each group to practice as per the timings decided in step 4.
   Step 6: Invite the teachers to Teachers' Day celebrations.
   Step 7: Perform the activities planned in step 2 on Teachers' Day
   Step 8: Stop

3. To celebrate New Year

   Algorithm:

   Step 1: Start
   Step 2: Prepare a guest list for New Year party
   Step 3: Decide the venue, food menu, games and fun activities for the party
   Step 4: Invite the guests for the party

Step 5: On New Year eve, get ready and enjoy the party
Step 6: Stop

4. Convert temperature Fahrenheit to Celsius

Inputs to the algorithm: Temperature in Fahrenheit
Expected output: Temperature in Celsius

Algorithm:

Step1: Start
Step 2: Read Temperature in Fahrenheit F
Step 3: C 5/9*(F32)
Step 4: Print Temperature in Celsius: C
Step5: End

5. Write an algorithm to read two numbers and find their product?

Inputs to the algorithm: First num1. Second num2.
Expected output: Sum of the two numbers.

Algorithm:

Step1: Start
Step2: Read\input the first num1.
Step3: Read\input the second num2.
Step4: Sum num1 * num2 // calculation of product
Step5: Print product.
Step6: End

6. An algorithm to calculate even numbers between 0 and 99

Step 1:Start
Step 2: I ← 0
Step3: Write I in standard output
Step 4: I ← I+2
Step 5:If (I <=98) then go to line 3
Step 6: End

7. A algorithm to find the largest value of any three numbers.

Step1: Start
Step2: Read/input A,B and C
Step3: If (A>=B) and (A>=C) then Max=A
Step4: If (B>=A) and (B>=C) then Max=B
Step5:If (C>=A) and (C>=B) then Max=C
Step6: Print Max
Step7: End

8. Design an algorithm which generates even numbers between 1000 and 2000 and then prints them in the standard output. It should also print total sum:

Solution:

1. Start
2. I ← 1000 and S ← 0
3. Write I
4. S ← S + I
5. I ← I + 2
6. If (I <= 2000) then go to line 3 else go to line 7
7. Write S
8. End

6. Design an algorithm with a natural number, n, as its input which calculates the following formula and writes the result in the standard output: S = ½ + ¼ + … +1/n

Solution:

1. Start
2. Read n
3. I ← 2 and S ← 0
4. S= S + 1/I
5. I ← I + 2
6. If (I <= n) then go to line 4 else write S in standard output
7. End

9. Define the following symbol and use?

Answer:

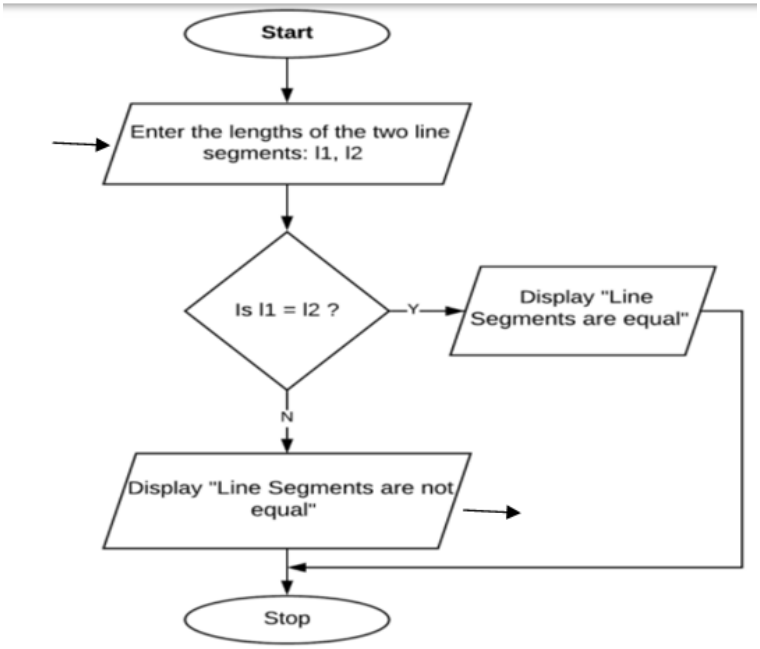
Process Box: A process box is used to represent all types of mathematical tasks like addition, subtraction, multiplication, division, etc.

10. Accept the length of two different line segments and check whether they are equal or unequal. Display the message accordingly.

Algorithm:

Step 1: Start
Step 2: Accept the length of the two line segments as l1 and l2.
Step 3: If l1 and l2 are equal, then display 'Line Segments are equal'.
Step 4: If l1 and l2 are not equal, then display 'Line Segments are not equal'.
Step 5: Stop

Flow chart:



11. Algorithm to find area of a rectangle?

Algorithm:

Step 1: Start
Step2: Declare the variables L, B and area // Declaration of variables
Step 3: Take length and breadth and store them as L and B? // Input
Step 4: Multiply L and B and store it in area //Area of Rectangle = L*B
Step 5: Print area //Output
Step 6: Stop.

12. **Print 1 to 20**

Algorithm:
Step 1: Initialize X as 0,
Step 2: Increment X by 1,
Step 3: Print X,
Step 4: If X is less than 20 then go back to step 2

Unsolved Problems

1) Write an algorithm and flow chart to add four numbers?
2) Write an algorithm to make tea/coffee
3) Write 6 No's Algorithms for performing day to day activities.
4) Write an algorithm and flow chart to find large number among given two numbers.
5) Draw the flow chart to convert distance entered in Km to Meters.
6) Accept the age of a person and check whether he/she is eligible to vote or not. A person is eligible to vote only when he/she is 18 years or more.
7) Draw the flow chart to find the area of a rectangle whose length and breadth are given
8) Draw the flow chart to find the average of three numbers a, b and c.
9) Write an algorithm and flow chart Print 1 to 100?
10) Write an algorithm to find average of three numbers?

Multiple Choice Questions:

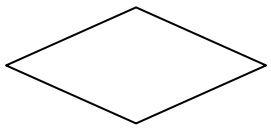
1) An Algorithm is expressed by
   a) flowchart
   b) common language
   c) pseudo code
   d) all
2) A good algorithm having finite steps?
   a) True
   b) False

3) Algorithm efficiency is measured by?
   a) speed
   b) space
   c) code
   d) all of the above
4) How many control structures are there in algorithm?
   a) 3
   b) 2

c) 1

5) Which control structure used weather a condition is true or false?
   a) selection
   b) iteration
   c) sequence
   d) none
6) Which control structure used to check a condition more than one time?
   a) iteration
   b) loop
   c) a & b
   d) None of the above

7) In computer science, algorithm refers to a pictorial representation of a flowchart.
   a) True
   b) False
8) The process of drawing a flowchart for an algorithm is called _____
   a) Performance
   b) Evaluation
   c) Algorithmic Representation
   d) Flowcharting
9) Actual instructions in flowcharting are represented in _____
   a) Circles
   b) Boxes
   c) Arrows
   d) Lines
10) A box that can represent two different conditions.
   a) Rectangle
   b) Diamond
   c) Circle
   d) Parallelogram
11) The following box denotes?



   a) Decision
   b) Initiation
   c) Initialization
   d) I/O
12) There should be certain set standards on the amount of details that should be provided in a flowchart
   a) True
   b) False

   Answer: b

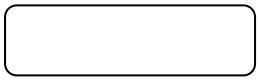
   Explanation: The statement is false. There should be no set standards on the amount of details that should be provided in a flowchart.
13) Which of the following is not an advantage of a flowchart?
   a) Better communication

b) Efficient coding
c) Systematic testing
d) Improper documentation

Answer: d

Explanation: Flowcharts provide a proper documentation. It also provides systematic debugging.

14) The symbol denotes _____



a) I/O
b) Flow
c) Terminal
d) Decision

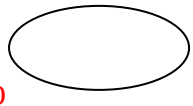
15) Pseudocode is an informal high-level description of an **algorithm**.

a) True
b) false

16) In a flowchart a calculation(process) is represented by

a) A rectangle
b) A rhombus
c) A parallelogram
d) A circle

17) The symbol denotes_____



a) I/O
b) Flow
c) Loop
d) Decision

18) To repeat a task a number times

a) Loop statement
b) Input statement
c) Conditional statement
d) Output statement

19) In a flow chart how are the symbols connected?

a) Symbols do not get connected together in a flowchart
b) With lines and arrow to show the direction of flow
c) With dashed lines and numbers
d) With solid lines to link events

20) A flow chart does need to have a start?

a) True
b) False

. Descriptive questions

1. Define Algorithm. What are the characteristics of a good algorithm?
2. What are advantages and disadvantages of algorithm?
3. Discuss about types of control structures?
4. Discuss about algorithm efficiency?
5. What are the characteristics of a good algorithm?
6. What is a flowchart? What are the features of a flowchart?
7. Explain about flow chart symbols?
8. What are control structures? Explain about its types in flowchart?
9. What is Pseudo code? Explain it?
10. What are difference between Pseudo code and algorithm?
11. What are difference between algorithm and flowchart?

Reference Links:

https://www.geeksforgeeks.org/difference-between-algorithm-and-flowchart/
https://www.edrawsoft.com/explain-algorithm-flowchart.html
https://www.youtube.com/watch?v=jwG5gaD3rU4
https://www.youtube.com/watch?v=XVGggCc-d4k