~IND

Chapter - 1
Introduction of HTML

Introduction to Internet

Internet means Inter connected Network. It is a Wide Area Network of over 150 countries, which connects multiple amount of different network. Internet connects thousands of network like different universities, government premises and many companies etc.

Internet is a global computer network which provide different types of information and many more facilities. There are different computers, mobile phones and other devices are connect with each other using internet. It’s works using TCP/IP (Transmission Control Protocol/Internet Protocol) to connect with each other.

All information found on Internet using Web page on web browser. Web browser is a software which can access and display web page. Web page is nothing more but is one type of a page, which provides an information. While two or more web pages connect with each other and make a directories, which is known as Web Site. To create this type of web page we need to learn its language and it is known as HTML (Hyper Text Markup Language).

What is HTML?

HTML is not a programming language but known as markup language. HTML is used to create or develop web page is called Hyper Text Markup Language. This language is interpreted by the web browser.

It's important to note that HTML is not a programming language. Technically it is known as a "markup" language, hence the name Hyper Text Markup Language. HTML is the language interpreted by a Browser. Web Pages are also called HTML documents.

HTML contains a collection of case-insensitive tags that starts and ends with angular brackets (<>). You can create hypertext documents using any text editor such as Notepad, FrontPage, Dream weaver and the documents are saved with the extension, .html or .htm.

- **HTML enables you to:**

  Create Hypertext documents that can contain headings, tables, text and images.

  Create hyperlinks that enable you to navigate through pages.

  Embed video, sound clips, style sheets and other applications in the hypertext documents.

Using HTML you can create two type of pages.

1) Static
2) Dynamic

(1) Static Page

The contents of a static Web page do not change after it is created. A static Web page displays the same content each time it is retrieved. The content in a static page is coded using HTML and can only be changed by editing the HTML. So, you can only see the content of web page. You cannot make any interaction with static page.

(2) Dynamic Page

The content of web page depends on user input. So interaction with the user is required to display a web page. A dynamic page can contain animations, graphics and sound.

HTML TAGS

HTML tags are usually English words or abbreviation. But they are different from the regular text because they are placed in small angle brackets. HTML tags are used to mark-up HTML elements. HTML tags are written into the two symbols '<' and '>'. The surrounding characters are called angle brackets.

HTML tag elements normally come in pairs like <body> and </body>. The first body tag in a pair is known as starting tag and the second one is the ending tag. The text between the starting and ending tag is known as element content in HTML. HTML tags are not case sensitive; <body> means the same as <BODY>.

There are two types of tags in HTML:

- Singular Tag
- Paired Tag

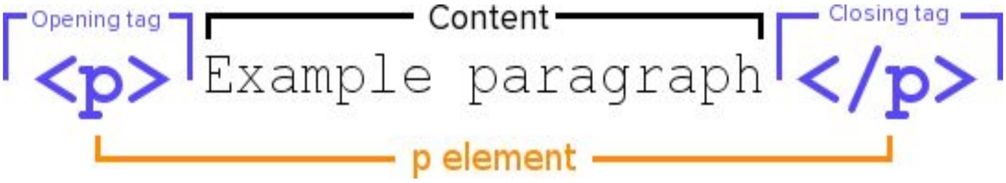
- **Singular Tag:**

  It is also called a standalone tag. A stand-alone tag does not have its companion tag. For e.g., <HR>, <BR>

- **Paired Tag:**

  The tag is said to be paired if it is along with its companion tag. In paired tag the first tag is called opening tag and the companion tag is called closing tag. The ending element always comes after the content of the element.

Here's a quick review of HTML tags:



For e.g., <B>ATMIYA University</B>

Structure of HTML Documents

- The first tag of an HTML document is always <HTML> and the last tag is always </HTML>.
- There are two main parts of HTML document.
  - Document Head
  - Document Body

Example:

```
<HTML>
        <HEAD>
                <TITLE>Document Title</TITLE>
        </HEAD>
        <BODY>
                - - - - - - - - - - - - - - - -
                Document Contents
                - - - - - - - - - - - - - - - -
        </BODY>
</HTML>
```

Document Head :

The header section of an HTML document contains information about the document (such as the title of the document, author, date of creation, software used to create the document). There should be no text or normal markup in the header section.

The information placed in this section is essential for inner processing. Within the document head there is one required header tag - the <TITLE>, </TITLE> field.

- **<TITLE> Tag**
  - You use the <TITLE> tag to set the title of an HTML document.
  - The <TITLE> tags are used inside the <HEAD> tags, as the, <TITLE> tag provides information about the document.
  - The title of the web page is normally shown in the title bar of the browser window.

Document Body :

<Body> …. </Body> is used to define document body. The body section contains the content of the document such as the actual text to display. Whatever content you want to display on web page are placed between <Body> ….

</Body> Within the document body you can place all tags that related to text, images, tables, lists are placed in this section. <Body> have various attributes.

Basic HTML Tags

1. **<b> Tag *or* <STRONG> Tag**

The <b> or <strong> tag displays a boldface type.

It is a paired tag.

Syntax:

```
<b> Web Scripting Language </b>
<strong> Web Scripting Language </strong>
```

2. **<i> Tag *or* <Em> tag**

The <i> or <Em> tag displays text in italic.

It is a paired tag.

Syntax:

```
<i> Web Scripting Language </i>
<EM> Web Scripting Language </EM>
```

3. **The <U> *or* <Ins>Tag**

It is used to display a text with underline.

It is a paired tag.

Syntax:

```
<U>Web Scripting Language</U>
```

4. **<BIG> Tag**

Big element shows the content in slightly bigger than normal size text. The <BIG> tag increases the font size from its current value by 1.

Syntax:

```
<BIG> Hello </BIG>
```

5. **<SMALL> Tag**

Small element shows the content in slightly smaller than normal size text. The <SAMLL> tag reduces the font size from its current value by 1.

Syntax:

```
<SMALL>……. </SMALL>
```

6. **The <SUP> Tag**

It is used to display a text as superscript.

It is a paired tag.

Superscript implies that the text is displayed in a text size smaller than the normal size and is positioned slightly above the normal body text.

Example:

```
x<sup>3</sup>
```

7. **The <SUB> Tag**

It is used to display a text as subscript

It is a paired tag.

Example:

```
H<SUB>2</SUB>O
```

8. The <BLOCKQUOTE> Tag

A web browser inserts space margin from left and right side of the content. It also inserts margins for the block quote element.

This tag is used to set left and right indents for the margin of text.

It is used for displaying a text as quoted text.

It is a paired tag.

Example:

```
<body>
     <blockquote>
          Web page contents.
     </blockquote>
</body>
```

9. The <CENTER> Tag

It is used to set center alignment of any content enclosed between <center></center> as per screen width.

Content may be text, image, table, list, form etc.... It is a paired tag.

Example:

```
<CENTER>Web Scripting Language</CENTER>
```

10. <STRIKE> or <s> or <Del> tag

It is used to display a text with a strikethrough character.

It is a paired tag.

Example:

```
<S>Web Scripting Language</S>
<del>Web Scripting Language</del>
```

11. The <BR> Tag

The <BR> Tag inserts a single line break.

The <BR> tag has no end tag.

When text needs to start from a new line and not continue on the same line, then <BR> tag should be used.

<BR> tag does not skip one blank line between previous line and new line.

It does not have any attributes.

Example:

```
<BODY>
        Yogidham Gurukul
        <BR>
        ATMIYA University
        <BR>
        Rajkot-360 005.
</BODY>
```

12. <DIV> tag

This element defines a division or a section of particular content in an HTML document. It is often used to create a group of one or more elements to format them with styles.

Example:

```
<body>
        <h1> ATMIYA University </h1>
        <p> This is a paragraph. </p>
        <div>
              This is a header.
              <p> This paragraph is displayed in division part.</p>
        </div>
</body>
```

13. The <PRE> Tag

This tag is used to display exactly same output as written in notepad. The pre element defines preformatted text. The text written into the opening and closing tag usually displays spaces and line breaks. The <PRE> tag displays the content in a new line. It is a paired tag.

Example:

```
<BODY>
    <Pre>
           #     #     # # # #      #              #             # # #
           #     #     #            #              #           #       #
           # # # #     # # # #      #              #           #       #
           #     #     #            #              #           #       #
           #     #     # # # #      # # # #        # # # #       # # #
    </PRE>
</BODY>
```

14. The Comment Tag:

This <!-- -- > tag is used to give a comment to your web page. This is a paired tag. The comment is used to insert a comment line for user in the source code. A comment will be ignored by the browser. You can also store program-specific information inside comments. In this situation comments text will not be visible for the user, but they are still available to the program.

Example:

```
<!-- This text is a comment -- >
```

Structured Elements

In Hyper Text Markup Language (HTML), structured tags are used to give structure to a document. For e.g., in a document, we can specify which is a heading and which part is the body of the document.

These tags are used in the body section of an HTML document. The structured tags provide visual effects to the content of an HTML document.

1. The <BODY> Tag

The <BODY> tag contains the content of the HTML document. The <BODY> tag is defined after the <HEAD> tag in an HTML document. So whether text, headings, textboxes, checkboxes, or any other possible content to be displayed must kept within the <Body> …. </Body>.

The attributes of the <BODY> tag enables you to use special features for the body content of an HTML document.

The various attributes of the <BODY> tag are:

- **Bgcolor**

  It can be used to change the background color of a web page. By default the background color is white. You can set the value for Bgcolor using constant character or using hexadecimal code.

  Example:

  <Body Bgcolor=”red”>

- **Background**

  It can be used to set an image in the background of an HTML document.

  You can set path for background using any one of the following way

  - Physical Path
  - Relative Path

  - Physical Path or Full Path:

    You have to specify the full path from the root directory to image file.

    Example:

    <Body background=”c:\images\mountain.jpg”>

  - **Relative Path:**

    Whenever the image file exists in current working directory at that time you do not need to specify full path. You can directly give the image file because by default Browser searches for file only within current working directory. For

    Example:

    <Body background=”mountain. jpg”>

- **Text**

  Enables you to specify the color of the text in hexadecimal code in an HTML document. By default the text color is black.

  Example:

  <Body text=”red”>

- **Link**

  Enables you to set the default color of the links in hexadecimal code and the color name in an HTML document.

  Here, Alink can be used to change the default color of a Hyperlink that is activated to whatever color is specified with this tag. With the use of this attribute user can specify the color name or a hexadecimal color code.

  Here, Vlink can be used to change the default color of a hyperlink that is really visited to whatever color is specified with this tag. The user can specify the color name or an equivalent hexadecimal number.

2. The <P> Tag

The <P> tag is used to start a paragraph from a new line. It is a paired tag. This tag is also used to insert a line break with an additional space. The ending </P> tag ensures that after the end of a paragraph some space is left blank.

Syntax:

```
<P align=”LEFT”|”RIGHT”|”CENTER”|”JUSTIFY”>Text</P>
```

- **Align**

  In <P> element align attribute specifies that the paragraph is aligned horizontally. The values assigned to the align attribute are:

  LEFT: Displays a left aligned
  paragraph. RIGHT: Displays a right aligned
  paragraph.
  CENTER: Displays a centrally aligned paragraph. This is the default value for the align attribute.
  JUSTIFY: This value displays a paragraph text as justified alignment.

  Example:

```
<HTML>
        <HEAD>
                <TITLE>Document Title</TITLE>
        </HEAD>
        <BODY>
                <p align=justify>
                        - - - - - - - - - - - - - - -
                        Document’s Paragraph
                        - - - - - - - - - - - - - - -
                </p>
        </BODY>
</HTML>
```

3. The <FONT> Tag

The <FONT> tag enables you to configure the various properties of the text to be displayed on a Web page such as style, size and color.

You can configure these properties using the following attributes:

Attributes:

- **SIZE**: Enables you to set the size of the text. You can specify the value of the SIZE attribute as an integer between 1 and 7. The default size of the font is +3.

- **COLOR:** Enables you to set the color of the text. You need to specify the hexadecimal value of the COLOR attribute to configure the color of the text. The default value of the COLOR attribute is black.
- **FACE:** Enables you to set the font style of the text. The default value of the FACE attribute is Times New Roman.

Example:

```
<HTML>
        <HEAD>
             <TITLE>Font</TITLE>
        </HEAD>
        <BODY>
                <FONT SIZE=7 COLOR=Blue FACE=arial>ATMYA University</FONT>
        </BODY>
</HTML>
```

4. The Standard Headings

Heading tags are used to define headings of contents in a document. The Line tags are used to implement styles on a particular line of text. They separate one block of text from another.

The standard heading tags, <H1>, <H2>, <H3>, <H4>, <H5>, <H6> are used to display headings. The <H1> tag is used to display the most dominant heading whereas <H6> tag is used to display the least dominant heading.

The headings are displayed from a new line in an emphasizing font style in contrast to the rest of the body content. All the style appears in Boldface and size of heading depending on level chosen.

These all are the paired tag, so you must have to write both tag start tag and end tag.

Attribute of Heading Tags:

- **Align:**

  This attribute is used to set alignment of headings. It have following four values.

Left – It align the heading in left hand side of the screen.

Right – It align the heading in Right hand side of the screen.

Center – It align the heading in Center of screen.

Justify – It align the heading in left hand side as well as right hand side of the screen.

```
<H1 align=”LEFT”|”RIGHT”|”CENTER”>…. </H1>
<H2 align=”LEFT”|”RIGHT”|”CENTER”>…..</H2>
<H3 align=”LEFT”|”RIGHT”|”CENTER”>…..</H3>
<H4 align=”LEFT”|”RIGHT”|”CENTER”>…..</H4>
<H5 align=”LEFT”|”RIGHT”|”CENTER”>…..</H5>
<H6 align=”LEFT”|”RIGHT”|”CENTER”>…..</H6>
```

5. The <HR> Tag

This tag is used to put a Horizontal line on the page. It is used to separate various sections of the document. The <HR> tag does not contain an ending tag.

The syntax of the <HR> tag is:

```
<HR
        color=”colorName”
        align=”CENTER”|”LEFT”|”RIGHT”
        size=”thickness”
        width=”width”
        noshade
>
```

In the above syntax, the various attributes of the <HR> tag are:

- **Color:** This attribute specifies the color of the horizontal line in web page. Value of this attribute comes in color name or 6 digits hexadecimal color code with “#” sign.
- **Size:** Specifies the thickness of a line in pixels. The default value is 2 pixels.
- **Width:** Specifies the horizontal width of a line. The default value for this attribute is the width of the Web browser window. The value for the attribute can be specified in either pixels or as a percentage of the width of the Web browser window.
- **Align:** Sets the horizontal alignment of a line. Three values can be assigned to this attribute, which are:
  - **CENTER:** Displays a center-aligned line. This is the default value for the align attribute.
  - **LEFT**: Displays a left-aligned line.
  - **RIGHT**: Displays a right-aligned line.
- **Noshade:** Specifies that no shading effect should be applied to the text included in the <HR> tag.

Example:

```
<HTML>
        <HEAD>
                <TITLE>Horizontal Line</TITLE>
        </HEAD>
        <BODY>
                Text before line
                <HR align=”LEFT” width=”50%” size=”8” color=”red”>
                Text after line
        </BODY>
</HTML>
```

6. <MARQUEE> Tag

The marquee element creates with scrolling text, similar to a live ticker tape from a stock brokerage firm. HTML provides the special feature which enables us to display scrolling and animated text in the web. It is a paired tag element.

Syntax:

```
<MARQUEE
        align="center" || “left” || “right”
        behavior=”scroll” || “slide” || “alternate”
        bgcolor=”color”
        direction=”left” || “right”||”top”||”Bottom”
        height=”height”
        width=”width”
        HSpace = “”
        Vspace =””
        Loop=2 >
            - - - - ATMIYA University - - - -
</MARQUEE>
```

- **ALIGN:**
  For vertical alignment of the marquee text.

- **Behavior:**
  It determines how marquee behaves scrolling text like a ticker tape, flying text in or having it move from side to side.

- **Bgcolor:**
  To set the background color of the marquee.

- **Direction:**
  To set the direction used with behavior attribute.

- **Height & width:**
  To set the vertical & horizontal size of the marquee.

- **Hspace, Vspace :**
  It sets the horizontal space around the marquee and vertical space around the marquee.

- **Scrolldelay :**
  This is used to specify the speed of text in a marquee moves. You have to type the delay, in milliseconds before the marquee text begins to move. Whether value is given in numeric value. Possible values are Delay or amount.

- **Scrollamount :**
  This is used to specify the amount of pixel in a marquee’s moves. You have to type the value in amount of pixel.

❖ **Loop :**

With the use of Loop attribute defines how many times the marquee will move. The default is INFINITE, which means loops endlessly. Also one of the problems with loop is that content disappears after the last loop. To overcome these problem use Behavior=Slide.

7. <A> Anchor Tag or Hyperlink (Create a link with other pages)

A hyper link connect to a document (or a resource) with content on a web page. Hyperlinks can point to any resource on the web: an HTML page, an Image, a sound file, a movie etc. A link in HTML documents can be basically considered as a reference to other resource. This reference establish a link between resource file and web page's content.

There are three different kinds of links:

1) **Internal Link:** Links to anchors on the current page.
(href="#anchorname")

2) **External or Local:** Links to other pages within your domain.
(href="D:/image/filename.jpg")

3) **Global Link:** Links to other domains outside of your site.
(href="http://www.google.com")

How to make a LINK?

The tag used to create links are the <A>…</A>. Everything between these two tags will work as a hyperlink. You can navigate between different HTML documents by clicking the hyperlinks. A hypertext link can navigate web pages and open other files also. A hypertext link can contain text and graphics. The value of the HREF attribute of anchor element must be a URL address. When the user clicks any content between <A HREF> and </A> tags, the link is activated.

Syntax:

```
<a
        HREF="location"
        TARGET="windowName">        ….
</a>
```

Attributes of Anchor tag are:

❖ **HREF="Path Name *or* Location"**

Specifies a destination URL for the link. With the use of HREF (Hyperlink Reference), provide the URL address of particular document file. This address could be either an absoulate or relative address.

Example of an absolute URL is:-
HREF="http://atmiyauniversity.edu/index.html"

Example of a relative URL is:-
HREF="d:/Atmiya/xyz.html"

TARGET="window's Name"
_blank = opens the linked file in new tab or a new browser window.
_self = This default value opens the linked file in the same frame as it was clicked.
_parent = This value opens the linked document in the parent frame.
_ top = with the use of this value opens the linked document in the full body of the window.

□ **Example:**

```
<html>
        <body>
          <a href="PathName" target="WindowName">Text</a>
        </body>
</html>
```

□ **Using Image as Hyperlink**

In HTML document Images can also work as Hyperlinks. To create hyperlink image, need to write the <IMG> tag between the opening and closing part of the anchor tag.

```
<A href="atmiya.htm">
          <IMG src="vsc.jpg">
</A>
```

□ **Creating a BOOKMARK**

The anchor also enables you to navigate within the current HTML document known as Bookmark. A Href attribute of the <A> tag is used to create bookmark.

The syntax is:
# anchor-name

In above syntax, the # symbol specifies that the path given, represents a link within the same document. The anchor-name represents the name of the destination anchor tag.
For e.g., you need to create a web page, bookmark.htm, which contains bookmarks.

- The coding of the Bookmarking is:

```
<body>
     <a href=#bookmark1>CLICK HERE TO SCROLL DOWN TO BOOKMARK1</a>
           <P>
                 Paragraph text Paragraph text Paragraph
                 text   Paragraph   text   Paragraph   text
                 Paragraph text Paragraph text Paragraph
                 text   Paragraph   text   Paragraph   text
                 Paragraph text Paragraph text Paragraph
                 text   Paragraph   text   Paragraph   text
                 Paragraph text Paragraph text Paragraph
                 text   Paragraph   text   Paragraph   text
                 Paragraph text Paragraph text Paragraph
                 text   Paragraph   text   Paragraph   text
                 Paragraph text Paragraph text Paragraph
                 text Paragraph text Paragraph text
           </P>
        <a name=bookmark1>BOOKMARK1</a>
</body>
```

8. <link> Tag

Use the <link> tag to link external files such as style sheets into your document. The <link> tag defines the relationship between a document and an external resource. The <link> tag is most used to link to style sheets. This tag is used in <head> section.

□ **Syntax:**

<link rel=”stylesheet” href=”url” type=”text/css”>

□ **Attributes:**

❖ **REL** = ”filetype”

- o Specifies the kind of file to be linked.
- o For an external style sheet, the value is “stylesheet”.

❖ **TYPE** = ”type”

- o Specifies more information about the particular type of file to be linked to, if that information is needed.
- o For style sheets the value can be “text/css” for cascading style sheets, or “text/javascript” for JavaScript style sheets.

❖ **HREF** = ”location”

Specifies a destination URL for the link.

Example:

```
<head>
        <link rel="stylesheet" type="text/css" href="styles.css" >
</head>
<body>
        <h1> ATMIYA University </h1>
        <p> I am formatted with a linked style sheet… </p>
</body>
```

9. CREATING LIST

A list represents a series of words one after another.
Lists are used when the data are to be mentioned in form list of items.
There are three types of list,

1. Ordered list
2. Unordered list
3. Definition list

1. ORDERED LIST <OL>

An ordered list can be represented by <OL> tag. List having the numbered items are known as ordered list. These lists have list items that have a natural number. The <LI> tag is nested inside the <OL> tag to represent every single element of an ordered list. <LI>…</LI> must be placed between <OL>…</OL>. It has two attributes.

```
<OL    Type=”a”|”A”|”i”|”I”|”1”    Start=value >
...
</OL>
```

Type attribute is used to set numbering format. The values may be one of the following:

a : Displays a series of lowercase letters.
A: Displays a series of uppercase letters.
i: Displays a series of lowercase Roman numerals.
I: Displays a series of uppercase Roman numerals.
1: Displays a series of numbers.

Start attribute is used to set starting position of the sequence. The values of this attribute are type of numeric.

2. UNORDERED LIST <UL>

These lists are also called bulleted list. These lists have list of items that do not have a number. This list is start with <UL> and end with </UL> tag. The <LI> tags are nested inside the <LI> tags to represent every single element of an unordered list.
<LI>…</LI> must be placed between <UL>…</UL> tag. It has one attribute.

```
<UL Type=”Circle”|”Square”|”Disc”> … </UL>
```

The Type attributes is used to set type of Bullets. The values are:

Circle – Represents a circular non-colored bullet.

Disc – Represents a solid bullet.

Square – Represents a square bullet.

3. DEFINITION LIST <DL>

The definition list is represented by the <DL> tag. The <DL> tag is used to represent the definition of terms. It is a paired tag.

<DT> and <DD> tags are nested inside the <DL> tag to represent the term and the definition of the term respectively. You can say that definition list consists of main two parts.

Definition Term.

- You can define definition term using <DT> tag.
- It is a paired tag.

Definition Data

- You can define any definition description using <DD> tag.
- It is a paired tag.

□ LIST ITEMS - <LI> tag

The <LI> tag is used to represent elements of both, the ordered and the unordered lists.

The <LI> tags are nested inside the <OL> and the <UL> tags.

<LI> tag does not need any ending tag.

Other HTML tags such as the <p> tag, can be nested inside the <LI> tag.

Syntax:

```
<LI
        Type = “DISC”|”CIRCLE”|”SQUARE”|”A”|”a”|”i”|”I”|”1”
        Value=”number”
>
```

□ IMAGE HANDLING IN HTML

Images used in HTML documents are classified into two classes:

Inline images and External images.

Inline images are displayed on a web page with the text and links and are loaded automatically when a HTML document is viewed on web browser.

External images are not displayed directly. They are stored separately from the HTML document and are loaded only when requested.

HTML allows placing of static or animated images in HTML page.

Commonly used format for images are PNG (portable N/w Graphics), GIF (Graphical Interchange Format), and JPEG (joint Photographic Experts Group)

10. <img>Tag

The <img> tag specifies an image to be displayed in an HTML document. An image can be a plain image that simply appears on the page. The <img> tag does not require a closing tag.

Syntax:

```
<img
          Src  =          "location"
          Height =        "tall"
          Width =         "wide"
          Alt      =      "alternative Text"
          Align    =      "alignment"
          Border =        "borderwidth"
          Hspace=         "horizmargin"
          Vspace=         "Vertimargin"
          Usemap=         "#mapname"

>
```

- **SRC**

  The src attribute is required for image location.
  Specifies the URL of the image to be displayed in the document.

- **Alt**

  Specifies text to be displayed if the browser does not support the <img> tag or if the use has superior image loading browser.

- **Align**

  Left align as image with the left margin.
  Right aligns as image with the right margin.
  Top aligns the top of an image with the top of the tallest item in the current item.
  Middle aligns the middle of an image with the baseline of the item in the current item.
  Bottom aligns as image with the bottom margin. Same as baseline.

- **Border**

  Specifies the width in pixels of a border around the image.
  The value must be an integer and pixel.

- **Height**

  Specifies the height of the image either in pixels or as a percentage of the height of the window block HTML that contains the image.
  It's indicate a number of pixels; specify the value as an integer.

- **Width**

  Specifies the weight of the image either in pixels or as a percentage of the weight of the window block HTML that contains the image.
  It's indicate a number of pixels; specify the value as an integer.

- **Hspace**

  Specifies the margins in pixels between the surrounding text and the right and left edges of an images.

- **Vspace**

  Specifies the margins in pixels between the surrounding text and the top and bottom edges of an images.

- **Ismap**

  To find out the coordinates pixels of an image ismap is used in <img> element.

- **Usemap**

  Represents an image map and specifies which Map tag is used to define the mapping for this image. It is associated with a MAP element's name attribute; and creates a relationship between the image and the MAP.

Example:

```
<html>
     <body>
           <img                          alt="Yogidham Gurukul"    width="32"
                 src="Atmiya.gif"
                 height="32">
     </body>
</html>
```

□ Creating Image Map

An image map refers to an image that is divided into different areas and each area links to a different HTML document. Thus image maps provide more functionality to hyper linked images.
You can click at a specific place within an image, to access the HTML document associated with that area.
Image maps can be created and applied to an image so specific portions of the image can be linked to a different file/image.

1. <MAP> Tag

The <MAP> tag contains information about the active area in an image map. The <MAP> tag contains one or more <AREA> tags. Each <AREA> tag describes a different area in the image and specifies the URL to which it connects.

Syntax:

```
<map name="mapName">
……
</map>
```

- **Name**="mapName"
  - Specifies the name of the map.
  - The NAME attribute is required.
  - Use this name preceded by # as the value of the USEMAP attribute of the <IMG> tag for the image map.

The <MAP> tag is used to define a client-side image map. An image map is an image with clickable areas. The Map element contains a number of area elements that defines the clickable areas in the image map.
Here the <AREA> tag defines a section of an image map. An image map can contain multiple sections where every section is associated with a different HTML document.

2. **<AREA> Tag**

The <AREA> tag describes an individual area in an image map.
The <AREA> tag must be used inside a <MAP> tag.

Syntax:

```
<AREA
        Shape=”shape”
        Coords=”coords”
        Href=”location”
        Target=”windowname”
         Nohref
>
```

- **Shape**

  Specifies the shape of the map.
  Each shape has its own form of the COORDS attribute.
  CIRCLE specifies a circle.
  RECT specifies a rect.
  Href=”location”
  Specifies the URL to invoke when a user clicks the area.

- **Coords**

  Specifies the shape of the map.
  The cords attribute is required.
  Codes for circle are (center-x,center-y, radius)
  Codes for RECT are (Left-Top and Right-bottom)
  Codes for Poly are (x1,y1,x2,y2,…..)

- **TARGET**=”frameName”

  Discussed previous in <A> element.

- **Nohref:** Specifies that the area has no links.

Example:

```
 <body>
          <img src="Atmiya.jpg" alt="Planets" usemap="#planetmap">
          <map name="planetmap">
            <area shape="rect" coords="5,7,27,36" href="sun.htm">
            <area shape="circle" coords="90,58,9" href="moon.htm">
         </map>
</body>
```

11. CREATING TABLES

□ Table Tag

Tables allow you to arrange items in a Grid pattern. It provides an organized and systematic display of data. So table is combination of Rows and Columns. Any table is made up of header, row, column and cells. Here <TABLE> tag is used to create a table.

Syntax:

```
<TABLE
    Align=”Left|Right|Center”
    Height=”height” Width=”width”
    Border=”value” bordercolor=”colorName”
    Cellpadding=”value” Cellspacing=”value”
    Bgcolor=”color”
    Hspace=”horizmargin” Vspace=”vertmargin”
    >
    …
</TABLE>
```

- ❖ **Align:** Specifies the Horizontal alignment of a table. The values are Left|Center|Right.
- ❖ **Height:** Specifies the height of a table. The value for the height attribute can be specified either in terms of number of pixels or as a percentage of the height of the web browser window.
- ❖ **Width:** Specifies the width of a table. The value for the width attribute can be specified either in terms of number of pixels or as a percentage of the width of the web browser window.
- ❖ **Border:** Specifies the thickness of a table border in pixels. By default, tables are displayed without borders. So its default value is 0.
- ❖ **Bordercolor:** Specifies the color of a table border in color name or a hexadecimal color code.
- ❖ **Cellpadding:** Specifies the space between the border and the content of the cell in pixels. The default value for this attribute is 1. Any text in the cells does not appear to be sticking to the border.
- ❖ **CellSpacing:** Specifies the space between the cells of a table in pixels or percentage. The default value for this attribute is 2.

- **Bgcolor:** Specifies the background color of a table. It can be specified either using color names, or using rrggbb hexa triplet. Default value is “white”. It can be used with all table tags.

- **Background:** Specifies the background picture of a table. We can specify an image to set background of HMTL page or table.

- **Hspace:** Specifies the space between the left and right edges of a table and the rest of the text on the page in pixels.

- **Vspace:** Specifies the space between top and bottom edges of a table and the rest of the text on the page in pixels.

□ <CAPTION> Tag

The <caption> tag is used to define the caption of a table. The <caption> tag is used inside the <TABLE> tag.

Syntax:

```
<caption align=”center”> ATMIYA University </caption>
```

The align attribute specifies the position of the caption in table. The values that can be assigned to the align values are top or bottom.

□ <TR> tag

The <TR> tag defines rows for a table. The <TR> tag is used inside the <TABLE> tag. Any special formatting like boldface or italics is done including appropriate formatting tags inside <tr>…</tr>

Syntax:

```
<TR   Bgcolor=”color” Align=”left|right|center”
      Valign=”top|bottom|middle|baseline”>
…
</TR>
```

- **Bgcolor:** The Bgcolor attribute sets the color of the background for the row.

- **Align:** The align attribute specifies the horizontal alignment of the table. The values that can be assigned to the align attribute are:
  - **Left:** Creates a left aligned table. This is the default value.
  - **Right:** Creates a right aligned table.
  - **Center:** Creates a centrally aligned table.

- **Valign:** The Valign attribute specifies the vertical alignment of the text within a cell. The values that can be assigned to the valign attribute are:

  Bottom: Displays the text at the bottom of the cell.

  Middle: Displays the text at the center of the cell. This is the default value for the valign attribute.

  Top: Displays the text at the top of the cell.

□ <TD> tag

The <TD> is used to specify the text in a cell of a table. The <TD> tag is used between the opening and closing <TR> tag. The content in a TH element is displayed BOLD and CENTERED. While the content in a TD element is look like REGULAR and LEFT-ALIGNED.

Syntax:

```
     <TD
          Align=”Center|Left|Right”
          Valign=”Bottom|Middle|Top”
          Bgcolor=”color”
          Colspan=”value”
          Rowspan=”value”
          Width=”pixelwidth”
          Height=”pixelheight”
          Nowrap
          >
          …
 </TD>
```

Attributes:

- **Colspan:**
  This attribute is used to merge more than columns cell into single cell for a single row. This attributes is useful when one row of the table needs to be certain number of columns wide. It is used with <TD> and <TR> tag. This attribute can be set by giving a numeric value.

- **Rowspan:**
  This attribute is used to merge more than rows cell into single cell for a single column. This attribute is used when one column of the table needs to be certain number of row wide. It is used with <TD> and <TR> tag. This attribute can be set by giving a numeric value.

- **Nowrap:**
  Specifies that the lines in a cell cannot be broken into the new lines.

□ **<TH> tag**

In HTML document the heading contents for a table are specified with the <TH> element. The content of a table header is automatically appeared in centered and in bold text. TH element is a paired tag.

Syntax:

```
<TH
        Align=”Center|Left|Right”
        Valign=”Bottom|Middle|Top”
        Bgcolor=”color”
        Colspan=”value”
        Rowspan=”value”
        Width=”pixelwidth”
        Height=”pixelheight”
        Nowrap
>
…
</TH>
```

HTML Character Entities:

Some characters like the <, >, & etc. characters, have a special meaning in HTML coding, and therefore it convert into markup language. So that to use this types of character compulsory used its character entity. To display a greater than or less than sign into web document, we have to use its character entity name or code.

Character Entities:

A sign like less than or greater than defines as an HTML element tag. When we use this sign, the browser convert as tag. To display this type of actual character sign we must writes its entities name or code in the HTML source.

A character entity has three parts:

(1) Starting with an ampersand sign (&)
(2) An entity name or a “#” sign
(3) An entity number, and finally a semicolon (;).

Ex.: To display a greater than (>) sign into web document, must write…
&gt; or &#62;

The character entities are case sensitive, so that it's written only in lowercase. Character entities name is easier to remember instead of its code but all the newest entity names are not support in the newest browser. At that time entity numbers is mostly prefer in almost all browsers.

The most common Character Entities:

Result	Description	Entity Name	Entity Number
	Non breaking	&nbsp;	&#160;
<	Less than	&lt;	&#60;
>	Greater than	&gt;	&#62;
&	Ampersand	&amp;	&#38;
"	Quotation mark	&quot;	&#34;
'	Apostrophe	&apos; (does not work In IE)	&#39;

Some other commonly used Character Entities:

Result	Description	Entity Name	Entity Number
¢	Cent	&cent;	&#162;
£	Pound	&pound;	&#163;
¥	Yen	&yen;	&#165;
€	Euro	&euro;	&#8364;
§	Section	&sect;	&#167;
©	Copyright	&copy;	&#169;
®	Registered Trademark	&reg;	&#174;
x	Multiplication	&times;	&#215;
÷	Division	&divide;	&#247;

Question Bank: (1 Mark Questions)

1) What is the full form of HTML? –**Hyper Text Mark-up Language.**
2) Which attribute is used to change the default font color using Body tag? –**Text**
3) Which font size is default in HTML document? –**3**
4) What is the range of font size in HTML document? –**1 to 7**
5) Which attribute is used to change font style in Font element? –**Face**
6) How many attributes are available in <HR> element? –**5**
7) How many values are available in behavior attributes of Marquee tag? –**3**
8) Which value is default of behavior attribute in Marquee element? –**Scroll**
9) How many values are available in direction attributes of Marquee tag? –**4**
10) Which attribute is used to scroll marquee text at left to right side? –**Direction**
11) Which attribute is used to specify the speed of text move in marquee? – **Scrolldelay**
12) Which type of link is used to connect the link with current page's content? – **Internal link**
13) Internal link is known as in HTML document. –**Bookmark**
14) Which attributes are used in <Link> element in HTML? –**Rel, Type, Href**
15) Which attribute is common in <OL> and <UL> element in HTML? –**Type**
16) Which elements are used to define Definition List in HTML? **<DL>, <DT>, <DD>**
17) Which values are used in type attribute of <UL> element? –**Disc, Circle, Square**
18) Which attribute is compulsory in <Img> tag? –**SRC**
19) Which 2 elements are used to create image map in HTML? –**<MAP> and <AREA>**
20) Which attribute is used to specify the coordinates of image in Image Map? – **Coords**
21) Which element is used to give a title in HTML Table? –**Caption**
22) How many values are used in align attribute of <Caption> tag? –**2**
23) Which attribute is used to give a space between two cells of HTML table? – **Cellspacing**
24) Which attribute is used to set the content at top, bottom or middle in HTML cell? – **Valign**
25) Which Character Entity name is used to display Registered Trademark in HTML document? – **&reg;**

True/False

26) There are only two values (Left & Right) are available in image alignment. (True/False) – **True**

27) We can't set the justify alignment in <H1> to <H6> element. (True/False) – **False**

28) Height attribute is used to specify the thickness of <HR> element in HTML. (True/False) – **False**

29) Nohref is the attribute of <Area> element. (True/False) – **True**

30) "&and;" entity name is used to display "&" character entity. (True/False) – **False**

31) Make the pair of element and attribute.

	(A) Elements	(B) Attributes
1	<HR>	Alt
2	<Table>	Href
3	<IMG>	Noshade
4	<Link>	Link
5	<Body>	BorderColor

Question Bank: (5 Marks Questions)

1) Introduced Internet with HTML. (Page No.1)

Ans. What is Internet? – 1 mark

HTML Introduction – 2 marks

2 types of HTML page – 2 marks

2) What is HTML? Describe types of web-page and types of tag. (Page No.1)

Ans. What is HTML? – 1 mark

Types of Web-page. – 2 marks

Types of Tag. – 2 marks

3) Explain Structure of HTML Documents. (Page No.3)

Ans. HTML Structure – 1 marks

Head Section – 2 marks

Body Section – 2 marks

4) Describe the basic HTML tag with detail. (Page No.4)

Ans. There are 14 basic html tags are available.

5) Describe the Body element with its attributes. (Page No.7)

Ans. About Body element – 1 mark

Attributes of body elements – 4 marks

6) Explain <P> tag, heading tag and <font> element with its attributes and an example. (Page No.8)

Ans. <P>, <H1> to <H6> and <Font> - Introduction – 1 mark

Attributes of above elements. – 4 marks

7) Describe the HR element with various attributes in detail. (Page No.10)

Ans. HR Description – 1 mark

Elements of HR element – 4 marks

8) Explain Marquee element with its attributes. (Page No.11)

Ans. Marquee explanation – 1 mark

Its attributes – 4 marks

9) What is Hyperlink? Describe Anchor tag with its various attributes. (Page No.12)

Ans. What is Hyperlink? – 1 marks

Different types of link – 1 mark

Hyperlink in Image – 1 mark

Attributes of Anchor tag – 2 marks

10) What is Bookmark? Describe internal link with an example. (Page No.13)

Ans. What is Bookmark? – 1 mark

Explanation of Internal link – 3 marks

Example of Internal link – 1 marks

11) Explain the elements and attributes for List. (Page No.15)

Ans. What is list? – 1 mark

Types of List with elements and attributes – 4 marks

12) Explain Image handling. (Page No.16)

Ans. Explanation about Image element – 1 mark

Image attributes – 4 marks

13) What is Image map? Explain related elements and attributes with an example. (Page No.18)

Ans. What is Image map? Explanation – 1 mark

Explanation about elements and its attributes used in Image map – 4 marks

14) Describe all elements and its attributes related to table tag. (Page No.20)

Ans. Explanation about Table element – 1 mark

All elements and its attributes related to table – 4 marks

15) Explain character entities in detail. (Page No.23)

Ans. What is character entities? Explanation – 3 marks

List of character entities with entities name and entities code. – 2 marks

Chapter - 2
Introduction of Form and Frame

What is Form?

- The <form> is a block-level element that's defines an interactive part of a webpage. HTML Forms are required, when we need to collect some data from the site visitor. For example, if you need to collection of information data during user registration such as name, email address, credit card, etc., at that time form is required.
- 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.
- There are various form elements are available like Textbox, Multiline textbox, pull-down menu, checkbox, radio button, buttons et.

<FORM>:

- The <form> tag creates an HTML form. The form can contain interface elements such as text fields, buttons, checkboxes, radio button, and selection lists that let users enter text and make choices.
- All elements in the form must be defined between the <form> and </form> tags.
- **Syntax:**

```
<form
        Action="url"
        Method="methodname"
        Name="formname">
        ……
</form>
```

□ Attributes:

❖ Action:

Action attribute is required to specify the URL address, where to store or submit the data information of concerned forms.

❖ Method:

Method attribute is specify that how to information is sent to the specified URL address.

- **GET**(the default) appends the input information to the URL which on most receiving systems becomes the value of environment variable QUERY_STRING.
- **POST** value specify types of method which sends the sensitive information like password to particular URL address. Value with POST request is not remember the data information in browser history.

❖ Name

Specifies the name of the form. The name is not displayed on the form. Scripting language can use the Name attribute to different forms if there are multiple forms on a page.

<INPUT> Element:

With the use of Input element form will receive the information data which are given by the user's input. The TYPE attribute determines to specific sort of form element to be created. The <input> element displays form element in several ways, depending on values of the **type** attribute.

❖ TYPE Attribute

A Type attribute is determine the various types of form controls.

1. Text

A value "text" is used to define a one-line text input field into HTML form. "Text" is a default value of type attribute.

Ex.:

Name : <input type="text">

Name: | ATMIYA University |

2. Password

A value "password" is used to define hidden (unreadable) text field into HTML form of web page. The characters in a password field are not readable and shown as asterisks or circles symbol.

Ex.:

Password : <input type="password">

Password : | * * * * * * |

3. Radio

A value "radio" defines a radio button. Which let a user to select only one option from limited options of choice.

❖ **Checked**

A checked attribute is used to set the control checked by default in HTML form. Its use only with checkbox or radio button.

Ex.:

Gender:<input type=”radio” checked>Male <input type=”radio”>Female

Output:

Gender: ●Male ○Female

4. **Checkbox**

- A value “checkbox” defines a checkbox in form. Checkboxes let a user to select zero or more options of a limited number of choices.

Ex.:

Hobbies: <input type=”checkbox” checked>Reading
<input type=”checkbox”> Music
<input type=”checkbox”> Chess

Output:

Hobbies : ☑Reading □Music □Chess

5. **File**

A value “file” is used to define a file-select field with a "Browse" button for file uploads.

Ex.:

Upload your file here:<input type=”file”>

Output:

Upload your file here: D:\Atmiya_University\file.jpg [Browse]

❖ **Multiple(Attribute)**

This attribute is a Boolean attribute. It define that the user is allowed to select more than one file. This attribute works with file input type. To select multiple files, hold down the CTRL or SHIFT key while selecting.

6. **Submit**

A value “submit” with type attribute defines a button for submitting form’s data to a form-handler data file.

Ex.: <input type=”submit”>

[Submit Query]

7. **Reset**

A value reset defines a button named with Reset, which will reset all the information as default form’s value.

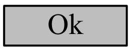
Ex.: <input type=”reset”>

[Reset]

8. Button

A value “button” is used to define a button in HTML form.

Ex.: <input type=”button” value=” Ok ”>



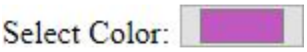
Ok

9. Color

A color value is used to provide a color picker interface element in web page. If you don't specify a value, the default is #000000, which is black.

Ex.: Select Color: <input type=color>

Output:



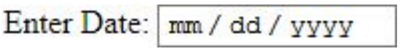
Select Color:

10. Date

A date value is used to define a date picker interface. The resulting value includes the year, month, and day.

Ex.: Enter Date: <input type=date>

Output:



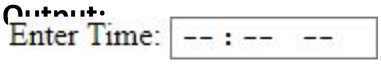
Enter Date: mm / dd / yyyy

11. Time

This Value allows the user to select a time from time controller from web page.

Ex.: Enter Time: <input type=time>

Output:



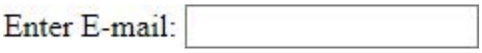
Enter Time: -- : -- --

12. Email

A value email is used for input fields that should contain an e-mail address. Depending on browser support, the e-mail address can be automatically validated when submitted.

Ex.: Enter E-mail: <input type=email>

Output:



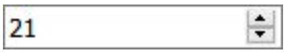
Enter E-mail:

13. Number

A value number provides a numeric input field in web page.

Ex.: <input type=number>

Output:



21

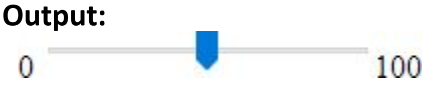
14. Range

A value range defines a control for entering a number using a slider control.

Default range is 0 to 100.

Ex.: 0 <input type=range> 100

Output:



❖ **Name Attribute**

The name attribute specifies the name of the input element. The name attribute is used to give a name to the control which is sent to the server to be recognized and get the value. The name is not displayed on the form.

❖ **Size Attribute**

Specifies the length of the input length field, in characters. The value should be an integer. This is also used with text or password type filed.

❖ **Value Attribute**

A Value attribute is used to provide an initial value inside the control in HTML form.

❖ **Placeholder Attribute**

This attribute defines a short hint that describes the expected value of an input field from user. The short hint is displayed in the input field before the user enters a value. The placeholder attribute works with a value like text, email, password and textarea element etc.

Restricted Attributes in Input element:

❖ **Maxlength Attribute**

This attribute is used to specify the maximum number of characters a text box can accept. This is used with text box or password box controller.

❖ **Required Attribute**

Required attribute specifies that an input field must be filled out before submitting the form’s data. It is works with input types like text, password, email, date, time, number, checkbox, radio and file.

❖ **Min Attribute**

The min attribute determine the minimum value into particular input field.

❖ **Max Attribute**

The max attribute determine the maximum value into particular input field.

- **Readonly Attribute**

  The read only field cannot be modified by user. User can select, highlight and copy it's data. The readonly attribute specifies that the input field cannot be changed. User can read only. It is work with Input and textarea element.

- **Disabled Attribute**

  The disabled attribute specifies that the input field is disabled. A disabled field is unusable and un-clickable, and its value will not be sent when submitting the form.

<Button> Element:

The <button> tag is used to create clickable buttons on the web page. The difference between these elements and buttons created with the <input> tag is that you can place the content (images or text) inside the <button>.

The <button> tag comes in pairs, the content is written between the opening and the closing tags. When <button> element is usedinto the form, it works as the submit button. You can also use it as reset button. While use it outside the form, you can call JavaScript function on it.

Ex.:

```
<button>Ok</button>
<button><img
src="xyz.jpg"></button>
```

<TEXTAREA> Element:

The <TEXTAREA> is paired tag element. It defines a multiline input field on an HTML form. A text area field lets the user enter words, phrases or numbers.

- **Syntax:**

```
<TEXTAREA
        Cols="columns"
        Rows="rows"
        Name="name">
        ……
</TEXTAREA>
```

Attributes:

- **Cols**

  Defines the width the text area can accommodate without scrolling.

- **Rows**

  Defines the number of line text area can accommodate without scrolling.

- **Name**

  This attribute specifies the name of the text area element.

<SELECT> Element:

The <select> element is applied to create a pull down (drop-down list) menu. The <SELECT> tag defines a selection list on an HTML form. The <SELECT> tag should be used between <FORM> tags. The <option> tags inside the <select> element define the available options in the list.

- **Syntax:**

```
<SELECT
        Name=”selectName”
        Size=”listlength”>

……
</SELECT>
```

□ **Attributes:**

- **Name**

  This attribute specifies the name of the select element.

- **Size**

  This one specifies the number of option visible when the form is displayed.

- **Multiple**

  With the use of this attributes we can select multiple option from given list.

<OPTION> Element:

The <OPTION> tag define an option in a selection list. We can use the <OPTION>tag inside a <SELECT> tag.

- **Syntax:**

```
<OPTION
        Value=”optionvalue”
        Selected

>
….
</OPTION>
```

□ **Attributes:**

- **Value**

  It’s define to specify a value that is returned to the server when the option is selected and form is submitted.

- **Selected**

  This attribute is used to specify that the option is selected by default.

- **Disabled**

  With the use of disabled attribute user did not able to select specified option using mouse or keyboard event.

- **Example:**

```
<form>
        <select name="fruit">
                <option>Apple</option>
                <option>Banana</option>
                <option>Mango</option>
                <option>Orange</option>
        </select>
</form>
```

<FRAMESET>

The <FRAMESET> tag defines a set of frames that appear in a web browser window. The <FRAMESET> tag contains one or more <FRAME> tags that each describes a frame. An HTML document that contains a <FRAMESET> tag cannot contain a <BODY> tag.

- **Syntax:**

```
<frameset>
                Cols = “column size in %”
                Rows = “row size in %”
                Border = “size in pixel”
                Bordercolor=”color name”
                Frameborder=”Yes/No”
</frameset>
```

NOTE: You must supply at least one of the cols or rows attributes.

Attributes:

Cols

- Cols attribute specifies a comma-separated list of values giving the width of each frame in the frameset.
- It specifies how many columns are contained the frames into the frameset.
- Cols attribute specify the width of each column in one of the four ways –
  1. Absolute values in pixels.
     For example, to create two vertical frames, we can use like *cols = "300, 500"*.
  2. A percentage of the browser window.
     For example, to create three vertical frames, use *cols = "10%, 80%, 10%"*.
  3. Using a wildcard symbol.
     For example, to create three vertical frames, use *cols = "10%, *, 10%"*. In this case wildcard takes remain part of the window.
  4. As relative width of the browser window.
     For example, to create three vertical frames, use *cols = "3*, 2*, 1*"*.

- **Rows**
  This attribute specifies a comma-separated list of values, which display height of each frame on the screen. Row size could be same as cols size. If one of the values is missing, the browser did not display last frame of the list.

- **Border**
  This attribute specifies the size of the border thickness of each frame in pixels. For example, border = "5". A value of zero means no border.

- **Bordercolor**
  This one is used to specify the color of a frame's borders.

- **Frameborder**
  With the use of this attribute browser specifies whether border should be displayed between frames or not. This attribute have value either 1 (yes) or 0 (no). For example frameborder = "0" specifies no border.

<FRAME>:

The <FRAME> tag creates a frame, which is an individual independently scrollable region of a web browser. The <FRAME> tag must be used within a <FRAMESET> tag. Each frame has distinct URL that determines the content displayed by the frame.

- **Syntax:**

```
<frame
        SRC="URL"
        frameborder=Yes
        bordercolor=red
        scrolling=no
        name=frmName
        noresize
>
```

With the use of frames, you can display more than one web documents in to one browser window. Frames are the sub windows of a web browser window and each frame can display one HTML document.

For e.g., in a web browser window, one frame can display a static banner, second frame can display a navigation menu and third frame can display an HTML document based on the selected menu from the second frame.

□ **Attributes:**

- **Bordercolor**
  This attribute is used to specify the color of a frame's borders.

Frameborder

With the use of this attribute browser specifies whether border should be displayed between frames or not. This attribute have value either 1 (yes) or 0 (no). For example frameborder = "0" specifies no border.

Noresize:

All border of the frames can resize by clicking and dragging on the borders of a frameset. With the use of noresize attribute a user did not able to resize the frame's height or width. For example noresize = "noresize".

SRC="URL"

- This attribute is used to specify the URL address of the file, which should be loaded in the particular frame. Its value can be any URL.
- For example, SRC = "D:/html/Atmiya.html" will load an HTML file available in html directory.
- **Example:**

```
<html>
        <frameset>
                <frame src="Atmiya.html">
         </frameset>
</html>
```

HTML Multimedia

Multimedia comes in many different formats. It can be almost Images, music, sound, videos, records, films, animations, and many more, which u can see or listen. Web page contain many types of multimedia contents like audio or video. This types of elements are stored in media files.

HTML Plug-in

HTML Plug-in allows the extra functionality to the web browser. HTML Plug-ins are computer programs which extend more functionality of a web browser. Plug-ins can be used for many purposes: display maps, scan for viruses, verify your bank id, etc.

Plug-ins can be added to web pages with the <object> tag or the <embed> tag.

□ <Object>

The <object> tag defines an embedded object within an HTML document. This element is used to include different types of media files into HTML document such as images, videos, audio, Java applets, ActiveX controls, PDF document, Flash animations etc.

□ <Embed>

The <embed> element can be used to embed external application, typically multimedia content like audio or video into an HTML document.

- **Example**:

```
<object width="400" height="400">
        <embed src="Atmiya.pdf">
        <embed src="Atmiya.swf">
</object>
```

□ HTML Audio

The <audio> element is used to embed audio content in an HTML document without requiring any additional plug-in like Flash player.

- **Example:**

```
<audio controls autoplay>
        <source src="Atmiya.mp3">
</audio>
```

□ Attributes:

❖ **SRC:**
Specifies the location of the audio file. Alternatively, you can use the preferred <source> tag as it allows for multiple options.

❖ **Autoplay:**
This Boolean attribute specifies that the audio will automatically start playing as soon as it can do so without stopping to finish loading the data.

❖ **Controls:**
If specified, the browsers will display controls to allow the user to control audio playback, such as play/pause, volume, etc.

❖ **Loop:**
This Boolean attribute specifies that the audio will automatically start over again, upon reaching the end.

❖ **Muted:**
This Boolean attribute specifies whether the audio will be initially silenced. The default value is false, meaning that the audio will be played.

□ HTML Video

The <Video> element is used to create video content in to web page document without any type of additional plug-in. There are three types of formats are supported in video element, like mp4, ogg and webM.

- **Example:**

```
<video controls autoplay>
          <source src="Atmiya.mp4" type="video/mp4">
</video>
```

□ Attributes:

❖ **SRC:**
This attribute is used to specify the location of the video file. The sub tag <source> is preferred to allow the multiple file option.

❖ **Height:**
This attribute is used to specify the displayed height area of the video.

- **Width:**
  This attribute is used to specify the displayed width area of the video.

- **Autoplay:**
  This Boolean attribute is used to play the video automatically as soon as possible.

- **Controls:**
  This attribute is used to display the video controls and allow to the user to use it like play, pause, stop, volume etc.

- **Loop:**
  This attribute is used to specify that the specified video will play automatically again and again.

- **Muted:**
  This attribute is specifies that the video played with sound or silenced. The default value of this attribute is false.

- **Poster:**
  This attribute is specifies an image to be shown into the video screen until the user hits the play button. If the poster attribute is not specified, the first frame of the video is displayed as the poster.

Introduction to HTML5

HTML5 is the latest and most enhanced version of HTML. HTML5 is supported new and latest functionality of element and attribute, which works well and more user friendly.HTML5 is a cooperation between the World Wide Web Consortium (W3C) and the Web Hypertext Application Technology Working Group (WHATWG).

□ **Browser Support**

The latest versions of the browser such as Safari, Google Chrome, Mozilla Firefox, and Opera all support the future and functionality of HTML5. While Internet Explorer 9.0 will also have support some HTML5 functionality.

The mobile web browsers that come pre-installed on iPhones, iPads, and Android phones all have excellent support for HTML5.

- **The most interesting new HTML5 elements are:**

  New semantic elements like <header>, <footer>, <article>, and <section>.
  New attributes of form elements like number, date, time, calendar, and range.
  New graphic elements: <svg> and <canvas>.
  New multimedia elements: <audio> and <video>.

□ HTML5 API's (Application Programming Interfaces)

- **HTML Geolocation**
  Geolocation is an API that helps the user to identify the web browser's geographic location. Firefox was one of the first browsers to use geolocation. Geolocation is to connect a web search with map data to help users to locate places within walking distance from your current location.

- **HTML Drag and Drop**
  The HTML5 allows you to drag and drop API for dragging and dropping specified object within and between web sites.

- **Canvas 2D Context**
  HTML5 allows user to draw and manipulate of graphics in a browser using <canvas> functionality. Using **canvas** element without the 2D Context API, user not be able to draw any graphical object. There are brushes, palette and paint all rolled into one.

- **Full Screen API**
  This API controls the use of the user's full screen in to web pages, without the browser's different user interface.

- **Detecting device orientation**
  Get the information when the user's device like mobile or tablet change its orientation (portrait or landscape). Device are react different while its orientation is differ.

New Elements in HTML 5

1. **<Mark> element**
   The HTML <mark> tag is used for indicating text as marked or highlighted for reference purposes. Highlighting text on a web page can help bring important information immediately to the reader's attention. The <mark> element is used to highlight text inside of another element such as a paragraph, list, or table.
   Ex.:<mark>marking effect with yellow color</mark>

2. **<abbr> element**
   Abbr means Abbreviation. The <abbr> element is used along with a title attribute to associate a full-text explanation with acronym. This element give useful information to browsers, translation systems and search-engines, while user move the cursor upon the particular text or object.
   Ex.:<abbr title="Atmiya University">AU</abbr>

3. **<Details> element**
   The <**details**>**tag** specifies additional details, which can view or hide by user in web-browser. The details element can be used to create an interactive widget that the user can open and close. Any type sort content can be put inside the <details> tag. The content of a <details> element should not be visible unless the open attribute is set.

Attribute:

Open: The open attribute is a Boolean attribute. This attribute specifies that the details should be visible (open) to the user.

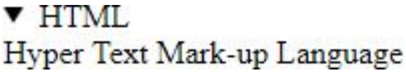
Sub Element: Summary

The <details> element is used to pair a <summary> statement with additional related details. The <summary> is displayed, and a user can view or hide additional details by clicking on the summary.
The <summary> tag is used to specify a visible heading for the details. The heading between opening and closing element <summary>can be clicked to view or hide the particular details.

Ex.:

```
<details>
        <summary>HTML</summary>
        Hyper Text Mark-up Language
</details>
```

Output:



▼ HTML
Hyper Text Mark-up Language

4. <Progress> element

The <progress> tag represents the progress of a task. It displays an indicator showing the completion progress of a task, typically displayed as a progress bar.

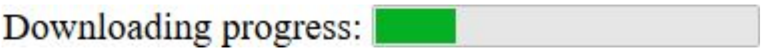
Attribute:

Value: This attribute specifies how much task has been completed in web-page. A value of this attribute must be between 0 to max, or between 0 and 1 if max is omitted. If there is no value attribute, the progress bar is indicates that the process is continuously ongoing.

Max: This attribute specifies that how much the task progresses completed. The max attribute must have a value greater than zero and be a valid floating point number. The default value is 1.

Ex.: <progress value=20 max=100>

Output:



Downloading progress:

5. <Meter> element

The <meter> is a new element in HTML5. The tag defines a scalar measurement in the known range or graphic representation of a fractional number. This types of element is used to display the battery level, disk space usage, etc. To use the <meter> tag, you need to know the maximum value.

Attributes:

Min: This attribute specifies the minimum value defined in the range.

Max: This attribute specifies the maximum value defined in the range.

Value: This is a mandatory attribute which is used to specify a value in numbers. The number may be integer or floating point number.

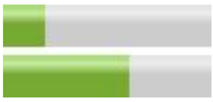
High: This attribute specifies a range that is considered as high value.

Low: This attribute specifies a range that is considered as low value.

Ex.:

```
<meter value=21 min=20 max=100 style="width:30%"></meter>
<meter value=0.7 style="width:30%"></meter>
```

Output:



6. <Fieldset>&<Legend> element

The <fieldset> tag draws a box around the related or group elements.

HTML <fieldset> tag is used to group the logically related fields/labels contained within an HTML form. The use of this tag is optional while creating an HTML form but using <filedset>, it is easy to understand the purpose of grouped elements of form. The <legend> tag is used with the <fieldset> element as a first child to define the caption for the grouped related fields.

The tag provides an opportunity to break forms down into logical sections. In the browser, a box around content is drawn.

Sub Element: Legend

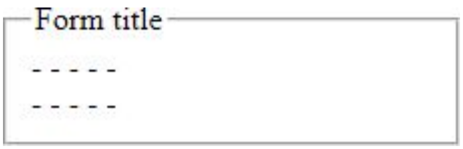
The <legend> tag is used to define a caption of a logical section defined by the <fieldset> element.

HTML <legend> tag used to add a caption (title) to a group of related form elements that are grouped together into the <fieldset> tag. The <legend> element always work with the <fieldset> element.

Ex.:

```
<fieldset>
        <legend>Form title</legend>
    </fieldset>
```

Output:



Form title

- - - - -

- - - - -

7. <OptGroup> element

OptGroup element is sub tag of <Select> element. This element is used to create a group of specific category options in your drop-down list within a select element. This one makes it easier for user to select their choices form multiple types of groups. If there are long list of options into drop-down control, groups of related options are easier to handle for a user.

Attribute: label

The **HTML** <optgroup> **element** creates a grouping of options within a <select> element. The <optgroup> is used to group related options in a drop-down list.

Value:

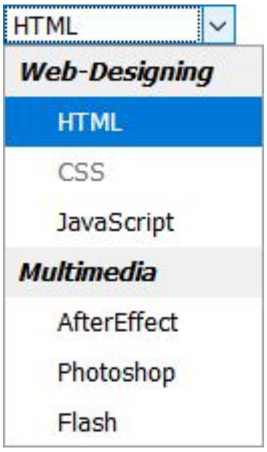
Disabled – Disabled attribute specifies that an option-group should be disabled. With the use of disabled attribute user did not able to select specified option using mouse or keyboard event.

Label – This attribute specifies a label for an option-group. The name of the group of options, which the browser can use when labeling the options in the user interface. This attribute is mandatory if this element is used.

Example:

```
<select required>
        <optgroup label="Web-Designing">
                <option value=html>HTML</option>
                <option value=css disabled>CSS</option>
                <option value=js>JavaScript</option>
        </optgroup>
        <optgroup label="Multimedia">
                <option value=AE>AfterEffect</option>
                <option value=PS>Photoshop</option>
                <option value=Fls>Flash</option>
        </optgroup>
</select>
```

Output:



Ch. 2 Introduction of Form & Frame

Question Bank: (1 Mark Questions)

1) Which attributes are used in <Form> of HTML? – **Action, Method, Name**
2) Which values are used in Method attribute of Form? – **Get, Post**
3) Which value is default for method attribute in form? – **Get**
4) What value is the default in type attribute of <Input>? – **Text**
5) Which value is used in type attribute to select one option from multiple choice in form? – **radio**
6) Which control is used to select zero or more option from multiple choice in form? – **Checkbox**
7) Which value is used in type attribute to upload file in form? – **File**
8) Which attribute is used to specify the limitation of input character in textbox? – **Maxlength**
9) Which element is used to input the multiple line into control field? – **Textarea**
10) Which attribute is used in <textarea> element? – **Cols, Rows, Name**
11) Which element is used to create a pull down menu in form? – **Select**
12) Which attribute is used to select more choice in pull down menu? – **Multiple**
13) Which attribute is used to specify that the option is selected by default in pulldown menu? – **Selected**
14) Which attribute is used to load html file in frame? – **SRC**
15) Which attribute is used to add source file in <source> element to insert audio? –**SRC**
16) Which attribute is used to insert video control like Play/Pause and Volume in Video element? –**Control**
17) Which attribute is used to set an image in starting of video clip in <video> element? – **Poster**
18) Which attribute of <video> start the preview of video automatically in web page? – **Autoplay**
19) Which attribute specifies that the video will automatically start over again, upon reaching the end? – **Loop**
20) is the latest and most enhanced version of HTML. – **HTML5**
21) Drawing and manipulation of graphics in a browser is possible using . – **Canvas**
22) Give the full form of W3C – **World Wide Web Consortium**
23) Give the full form of API –**Application Programming Interfaces**
24) <Audio> and <Video> are the new Multimedia elements of HTML5. (True/False) – **True**

Question Bank: (5 Marks Questions)

1) Explain Form element with its attributes. (Page No. 28)

   Ans. What is form? – 2 marks
   Form Attributes – 3 marks

2) Describe Form with its all controls. (Page No. 28 & 29)

   Ans. What is Form? – 1 marks
   Controls of Form – 4 marks

3) Explain Input element with type attribute.(Page No. 29)
Ans. About Input element – 1 marks
About type attribute with its values – 4 marks

4) Explain Input element with its attributes. (Page No. 29)
Ans. About Input element – 1 mark
Attributes of Input element – 4 marks

5) Explain restricted attributes in Input element with an example. (Page No. 32)
Ans. Explanation of 6 restricted attributes – 5 marks

6) Describe textarea and Pull down menu. (Page No. 33)
Ans. About Textarea and Select element – 1 mark
Textarea attributes – 1 marks
Attributes of Select and Option element – 3 marks

7) Explain Frameset with an example. (Page No. 35)
Ans. Frameset Explanation – 1 mark
Frameset Attributes – 4 marks

8) Explain Frame with an example. (Page No. 36)
Ans. Frame Explanation – 1 mark
Frame Attributes – 4 marks

9) Explain Multimedia in HTML. (Page No. 37)
Ans. Explanation about Multimedia & Plug-in – 2 marks
Object& Embed Element with attributes – 2 marks
Example – 1 mark

10) Explain Multimedia with Audio element. (Page No. 37 & 38)
Ans. What is Multimedia? – 1 mark
Audio element in HTML. – 1 mark
Attributes of Audio element. – 3 marks

11) Explain Multimedia with Video element. (Page No. 37 & 38)
Ans. What is Multimedia? – 1 mark
Video element in HTML. – 1 mark
Attributes of Video element. – 3 marks

12) Give the brief introduction of HTML5. *Or*
What is HTML5? Explain HTML5 API's. (Page No. 39 & 40)
Ans. What is HTML5? – 1mark
HTML5 browser support – 1 mark
New elements of HTML5. – 1 mark
HTML5 API's – 2 marks

13) Explain new element in HTML5. (Page No. 40)
Ans. 7 elements with an example – 5 marks

Chapter - 3
Cascading Style Sheet

What is CSS?

CSS stands for Cascading Style Sheets. CSS is used to control the style of a web document in a simple and easy way.

Styles are used generally for web-designing purpose so it works with HTML.

Styles are normally stored in Style Sheets.

External Style Sheets can save your lot of work. External Style Sheets are stored in CSS files. Multiple style definitions will cascade into one.

Styles Solve a Common Problem:

HTML tags were originally designed to define the content of a document. Supposed to say "This is a header", "This is a paragraph", "This is a table", by using tags like <h1>, <p>, <table>, and so on. The layout of the document was supposed to be taken care of by the browser, without using any formatting tags.

As the two major browsers - Netscape and Internet Explorer - continued to add new HTML tags and attributes (like the <font> tag and the color attribute) to the original HTML specification, it became more and more difficult to create Web sites where the content of HTML documents was clearly separated from the document's presentation layout.

To solve this problem, the World Wide Web Consortium (W3C) created STYLES in addition to HTML.

All major browsers support Cascading Style Sheets.

Style Sheets Can Save a Lot of Work:

- Styles sheets define HOW HTML elements are to be displayed, just like the font tag and the color attribute in HTML 3.2. Styles are normally saved in external .css files.
- External style sheets enable you to change the appearance and layout of all the pages in your Web, just by editing one single CSS document!
- CSS is a breakthrough in Web design because it allows developers to control the style and layout of multiple Web pages all at once.
- As a Web developer you can define a style for each HTML element and apply it to as many Web pages as you want. To make a global change, simply change the style, and all elements in the Web are updated automatically.

Multiple Styles Will Cascade into One:

Style sheets allow style information to be specified in many ways.
Styles can be specified inside a single HTML element, inside the <head> element of an HTML page, or in an external CSS file.
Even multiple external style sheets can be referenced inside a single HTML document.

Uses of CSS:

1. **Text Formatting and color**

   CSS can be used to create number of text effects, such as:
   - Choosing specific fonts and font size.
   - Setting bold, italics, underline, and text shadows.
   - Changing text color and background color.
   - Changing the colors of link or removing underlining.
   - Aligning text.
   - Stretching and adjusting text size and line spacing.
   - Transforming sections of text to upper, lower or mixed case.

2. **Graphical Appearance and Layout**
   - Setting a background graphic, controlling its location, tiling, and scrolling.
   - Drawings borders and outlines around sections of a page.
   - Setting horizontal and vertical margins on all elements, as well as vertical and horizontal padding.
   - HTML tables, forms, and lists are presented.

3. **Dynamic Action**
   - Mouse over effects on links.
   - Dynamically inserted content before or after HTML tags.
   - Automatic numbering of page elements.

CSS Syntax:

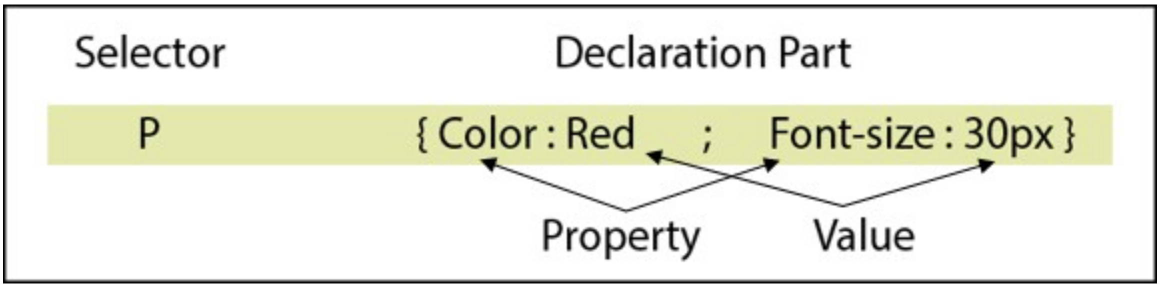
```
<STYLE type=”text/css” >
        Selector {property: value ; property: value ; ….}
</STYLE>
```

A CSS comprises of style rules that are interpreted by the browser and then applied to the corresponding elements in your document.

A style rule is made of three parts…

1) **Selector** − A selector is an HTML tag at which a style will be applied. This could be any tag like <h1> or <table> etc.
2) **Property** − A property is a type of attribute of HTML tag. Put simply, all the HTML attributes are converted into CSS properties. They could be color, border etc.
3) **Value** − Values are assigned to properties. For example, color property can have value either red or #F1F1F1 etc.

You can put CSS Style Rule Syntax as follows…



The style assign a process accomplished with the <STYLE>….</STYLE>tags.The <STYLE></STYLE> HTML tags are written within the <HEAD>…</HEAD> tags.In the <STYLE> Tag, the expression “TYPE=text/css” indicate that the style sheet Confirm to CSS syntax.

The CSS syntax is made up of three parts: a selector, a property and a value.The selector is normally the HTML element/tag you wish to define, the property is the Attribute you wish to change and each property can take a value. The property and value are separated by a colon and surrounded by curly braces:

- **Example**:

```
body {color: black}
```

CSS Comments:

You can insert comments into CSS to explain your code, which can help you when you edit the source code at a later date. A comment will be ignored by the browser.

A CSS comment begins with "/*", and ends with "*/", like this:

```
/* This is a comment */
  P{
        text-align: center;
        /* This is another comment */
        Color:red
  }
```

Types of CSS and How to Insert a Style Sheet:

- There are three ways of inserting a style sheet:
  1) External Style Sheet
  2) Internal Style Sheet
  3) Inline Styles

1) External Style Sheet

An external style sheet is ideal when the style is applied to many pages. With an external style sheet, you can change the look of an entire Web site by changing one file. Each page must link to the style sheet using the <link> tag.

The <link> tag goes inside the head section:

```
<head>
        <link rel="stylesheet" type="text/css" href="css file name">
</head>
```

The browser will read the style definitions from the css file and format the document according to it. An external style sheet can be written in any text editor. The file should not contain any html tags. Your style sheet should be saved with a .css extension.

An example of a style sheet file is shown below…

Style.css

```
P {
            Color: yellow;Font-family: arial;
}
```

Simple.html

```
<html>
      <head>
              <link rel=”stylesheet” type=”text/css” href=”Style.css”>
      </head>

      <body>
              <P>India is my Country.</P>
      </body>
<html>
```

2) Internal Style Sheet

An internal style sheet should be used when a single document has a unique style. You define internal styles in the head section by using the <style> tag, like this:

```
<html>
    <head>
            <style type="text/css">
            p { color: red; font-family: arial }
            </style>
    </head>
</html>
```

The browser will now read the style definitions, and format the document according to it.

3) Inline Styles

An inline style loses many of the advantages of style sheets by mixing content with presentation. Use this method sparingly, such as when a style is to be applied to a single occurrence of an element.

To use inline styles you use the style attribute in the relevant tag. The style attribute can contain any CSS property.

The example shows how to change the color & the left margin of a paragraph…

```
    <p style="color: orange; font-size: 20px">
    This is a paragraph
    </p>
    <p> Hello ATMIYA</p>
```

Multiple Style Sheets

If some properties have been set for the same selector in different style sheets, the values will be inherited from the more specific style sheet.

For example, an external style sheet has these properties for the h3 selector:

```
    h3{Color: red; text-align: left; font-size: 8pt}
```

And an internal style sheet has these properties for the h3 selector:

```
    h3{Text-align: right; font-size: 20pt}
```

If the page with the internal style sheet also links to the external style sheet the properties for h3 will be:

```
    Color: red; text-align: right; font-size: 20pt
```

The color is inherited from the external style sheet and the text-alignment and the font- size is replaced by the internal style sheet.

Types of Selector:

1) The class Selector

With the class selector you can define different styles for the same type of HTML element. Say that you would like to have two types of paragraphs in your document: one right-aligned paragraph, and one center-aligned paragraph.
Here is how you can do it with styles:

```
p.right {text-align: right}
p.center {text-align: center}
```

You have to use the class attribute in your HTML document:

```
<p class="right">
This paragraph will be right-aligned.
</p>
<p class="center">
</p>
```

This paragraph will be center-aligned. But only one class attribute can be specified per HTML element! The example below is wrong:

```
<p class="right" class="center"> this is a paragraph.     </p>
```

You can also omit the tag name in the selector to define a style that will be used by all HTML elements that have a certain class.
In the example below, all HTML elements with class="center" will be center-aligned:

```
.center {text-align: center}
```

In the code below both the h1 element and the p element have class="center". This means that both elements will follow the rules in the ".center" selector:

```
<h1 class="center">
        This heading will be center-aligned
</h1>
<p class="center">
        This paragraph will also be center-aligned.
</p>
```

2) The id Selector

With the id selector you can define the same style for different HTML elements.
The style rule below will match any element that has an id attribute with a value of "green":

```
#green {color: green}
```

The rule above will match both the h1 and the p element:

```
<h1 id="green">Some text</h1>
<p id="green">Some text</p>
```

The style rule below will match a p element that has an id with a value of "para1":

```
p#para1
{
        text-align: center;
        color: red
}
```

The style rule below will match any p element that has an id attribute with a value of "green":

```
        p#green {color: green}
```

The rule above will not match an h1 element:

```
<h1 id="green">Some text</h1>
```

CSS Font Properties:

The CSS font properties allow you to change the font family, boldness, size, and the style of a text.

- □ font-family property
- □ font-style property
- □ font-weight property
- □ font-size property
- □ font-variant property

□ Font-family property:

The font-family property is a prioritized list of font family names for an element. The browser will use the first value it recognizes. Separate each value with a comma. The font family of a text is set with the font-family property. The font-family property should hold several font names. If the browser does not support the first font, it tries the next font, and so on.

Note: If the name of a font family is more than one word, it must be in quotation marks, like: "Times New Roman".

□ **Example:**

```
Body{Font-family: courier, serif }
p{Font-family: arial, "lucida console", sans-serif}
<p style="font-family: arial, 'lucida console', sans-serif">
```

□ Font-style property:

The font-style property sets the style of a font. It is mostly used to specify italic text.

Normal : Value ‘Normal’ displays a normal font into content area.

Italic : Value ‘Italic’ displays a italic font into content area.

□ **Example:**

```
P {font-style: italic }
```

□ **Font-weight property:**

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

Normal	**:**	Value 'Normal' defines a normal character in selected content.
Bold	**:**	Value 'Bold' displays a thick characters into content area.
Bolder	**:**	Value 'Bolder' displays a thicker characters into content area.
Lighter	**:**	Value 'Lighter' displays a thick characters into selected tag.
100 to 900	**:**	It defines from thin to thick characters. Value 400 is the same as normal value and 700 is the same as bold value.

▯ **Example:**

```
P{ font-weight: bold }
```

□ **Font-size property:**

The font-size property sets the size of a 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.

The font-size value can be an absolute, or relative size.

xx-small/x-small/small/ medium/large/x-large/ xx-large	**:**	Sets the size of the font to different sizes, from xx-small to xx-large
Smaller	**:**	Sets the font-size to a smaller size than the parent element
Larger	**:**	Sets the font-size to a larger size than the parent element
Length	**:**	Sets the font-size to a fixed size
%	**:**	Sets the font-size to a % of the parent element

▯ **Example:**

```
Body{ font-size: x-large }
p{ font-size: 10px }
```

□ **Font-variant property:**

The font-variant property is used to display text in a small-caps font, which means that all the lower case letters are converted to uppercase letters, but all the letters in the small-caps font have a smaller font-size compared to the rest of the text.

Normal :	The browser displays a normal font
Small-caps :	The browser displays a small-caps font

▯ **Example:**

```
P { font-variant: small-caps }
```

CSS Text Properties:

The CSS text properties allow you to control the appearance of text. It is possible to change the color of a text, increase or decrease the space between characters in a text, align a text, decorate a text, indent the first line in a text, and more.

- □ color property
- □ text-align property
- □ text-decoration property
- □ text-transform property
- □ word-spacing property
- □ letter-spacing property
- □ line-height property

□ Color property

The color property sets the color of a text.

Color : The color value can be a color name (red), a rgb value (rgb(255,0,0)), or a hex number (#ff0000).

□ **Example**:

```
Body{color: rgb(255,255,0) }
```

□ Text-align property

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

Left	:	Aligns the text to the left
Right	:	Aligns the text to the right
Center	:	Centers the text
Justify	:	Define justified both margin

□ **Example**:

```
P { text-align: center }
```

□ Text-decoration property

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

None	:	Defines a normal text
Underline	:	Defines a line under the text
Overline	:	Defines a line over the text
Line-through	:	Defines a line through the text

□ **Example**:

```
p {text-decoration: underline}
```

□ **Text-transform property**

The text-transform property controls the letters in an element. It 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.

None	:	Defines normal text, with lower case letters and capital letters
Capitalize	:	Each word in a text starts with a capital letter
Uppercase	:	Defines only capital letters
Lowercase	:	Defines no capital letters, only lower case letters

▯ **Example**:

p{text-transform: uppercase}

□ **Word-spacing property**

The word-spacing property increases or decreases the white space between words. Negative values are allowed.

Normal	:	Defines normal space between words
Length	:	Defines a fixed space between words

▯ **Example**:

p{word-spacing: 30px}
p {word-spacing: -0.5px}

□ **Letter-spacing property**

The letter-spacing property increases or decreases the white space between characters. Negative values are allowed.

Normal	:	Defines normal space between characters
Length(px/pt/mm/cm…)	:	Defines a fixed space between characters

▯ **Example**:

p {Letter-spacing: 12px } p
{Letter-spacing: -0.5px}

□ **Line-height property**

The line-height property increases or decreases the spacing between two lines of paragraph. A value of line-height property should be in numeric value.

▯ **Example**:

p {Line-height: 2 }
p {Line-height: 70px }

CSS Multi-column Properties

□ **column-count:**

The column-count property specifies the number of columns of an element. All contents are divided into specifies number's columns. The following example will divide the text in the <div> element into 3 columns:

Ex.:

```
div{
column-count: 3;
}
```

□ **column-gap:**

The column-gap property specifies the gap between two columns. The following example specifies a 40 pixels gap between the columns:

Ex.:

```
div{
column-count: 3;
column-gap:40px;
}
```

□ **column-rule:**

The column-rule property is a shorthand property for setting all the column-rule like width, style and color properties. This property displays visible division between two columns. The following example sets the width, style, and color of the rule between columns:

```
div{
    column-rule: 1px solid lightblue;
}
```

CSS Text Over Flow Property

□ **OverFlow**

The overflow property specifies whether to clip the content or to add scrollbars when the content of an element is too big to fit in the specified area. The overflow property only works for block elements with a specified height.

The overflow property has the following values:

- **Visible :** Default. The overflow is not clipped. The content renders outside the element's box
- **Hidden :** The overflow is clipped, and the rest of the content will be invisible
- **Scroll :** he overflow is clipped, and a scrollbar is added to see the rest of the content
- **Auto :** Similar to scroll, but it adds scrollbars only when necessary.

□ **Example:**

```
div {
            overflow: auto;          // scroll / visible / hidden
  }
```

CSS Padding Properties

□ The CSS padding properties define the space between the element border and the element content.

□ Negative values are not allowed. The top, right, bottom, and left padding can be changed independently using separate properties.

□ A shorthand padding property is also created to control multiple sides at once.

- □ padding-top property
- □ padding-bottom property
- □ padding-left property
- □ padding-right property
- □ padding property

□ **Padding-top property**

The padding-top property sets the top padding (space) of an element. Negative values are not allowed.

Length(Pixel or %) : Defines atop padding in % of the height of the closest element.

□ **Example:**

```
h1{ padding-top:10px }
```

□ **Padding-bottom property, padding-left property, padding-right property**

The padding-bottom property, padding-left property, padding-right property sets the bottom padding (space), left padding (space), and right padding (space) of an element respectively. Negative values are not allowed.

Value : Same to padding-top property

□ **Example:**

```
h1 { padding-right:10px; padding-left: 10px;padding-bottom:10px}
```

□ Padding property

The padding property is a shorthand property for setting all of the padding properties in one declaration. Negative values are not allowed.

padding-top padding-right padding-bottom padding-left	Sets the padding. The values comes in%(defines padding in % of the width of the closest element) and length (defines affixed padding)

□ **Example**:

h1 {padding: 10px}

The padding will be 10px on all four sides h1

h1{padding: 10px2%}

Top and bottom padding willbe10px, left and rightpaddingwillbe2% of the width of the closest element.

h1 {padding: 10px2% 15px}

Top paddingwillbe10px, left and right padding will be 2%of the width of the closest element, bottom paddingwillbe15px

h1 {padding: 10px2% 15px20px}

Top paddingwillbe10px, right padding will be 2% of the width of the closest element, bottom padding will be15px, left padding will be20p.

CSS Margin Properties

The CSS margin properties define the space around elements. It is possible to use negative values to overlap content. The top, right, bottom, and left margin can be changed independently using separate properties. A shorthand margin property can also be used to change all of the margins at once. Netscape and IE give the body tag a default margin of 8px. Opera does not! Instead, Opera applies a default padding of 8px, so if one wants to adjust the margin for an entire page and have it display correctly in Opera, the body padding must be set as well!

- □ margin-top property
- □ margin-bottom property
- □ margin-left property
- □ margin-right property
- □ margin

□ Margin-top property

The margin-top property sets the top margin of an element. Negative values are allowed.

Auto	:	The browser sets atop margin
Length	:	Defines affixed top margin
%	:	Defines atop margin in % of the total height of the document

- **Example**:

  H1 { margin-top: 10px}
  H2 { margin-top: 20px}

□ **Margin-bottom property, margin-left property, margin-right property**The margin-bottom property, margin-left property & margin-right property sets the bottom margin, left margin & right margin of an element respectively. Negative values are allowed.

Value: Same to margin-top property

- **Example**:

  h1 {margin-left: 10px ; margin-right: 10px }
  h2 {margin-bottom: -20px }

□ **Margin property**

The margin property is a shorthand property for setting all of the properties for the four margins in one declaration. Negative values are allowed.

Margin-top : **Margin-right :** **Margin-bottom :** **Margin-left :**	Sets the properties for the margins. The values comes in%(defines a margin in % of the total height/width of the document), Length : (defines a fixed margin), and auto (the browser sets a margin).

- **Example**:

h1{margin:10px}

All four margins will be 10px.

h1 {margin: 10px 2%}

Top and bottom margin will be 10px, left and right margin will be 2%of the total width of the document.

h1 {margin:10px2% -10px}

Top margin will be 10px, left and right margin will be 2%of the total width of the document, bottom margin will be -10px

h1 {margin:10px2% -10px auto}

Top margin will be 10px, right margin will be 2%ofthe total width of the document, bottom margin will be -10px, left margin will be set by the browser

CSS Border Properties

The CSS border properties allow you to specify the style and color of an element's border. In HTML we use tables to create borders around a text, but with the CSS border properties we can create borders with nice effects, and it can be applied to any element.

□ Border-style property

The border-style property sets the style of the four borders, can have from one to four styles.

None	:	Defines no border
Hidden	:	The same as "none", except in border conflict resolution for table elements
Dotted	:	Defines a dotted border. Renders as solid in most browsers
Dashed	:	Defines a dashed border. Renders as solid in most browsers
Solid	:	Defines a solid border
Double	:	Defines two borders. The width of the two borders are the same as the border-width value
Groove	:	Defines a 3D grooved border. The effect depends on the border-color value
Ridge	:	Defines a 3D ridged border. The effect depends on the border-color value
Inset	:	Defines a 3D inset border. The effect depends on the border-color value
Outset	:	Defines a 3D outset border. The effect depends on the border-color value

□ **Example**:

table {border-style: dotted dashed solid double}

Top border will be dotted, right border will be dashed, bottom border will be solid, left border will be double.

□ Border-color property

□ The border-color property sets the color of the four borders. This property can take one to four colors. Always declare the border-style property before the border-color property. An element must have borders before you change the color of them.

Color	:	The color value can be a color name (red), A RGB value (RGB (255,0,0)), or a hex number (#FF0000).
Transparent	:	The border is transparent

- **Example:**

table {border-color: red}
All four borders will be red.

table {border-color: red green blue yellow}
Top border will be red, right border will be green, bottom border will be blue, left border will be yellow.

□ **Border-width property**

The border-width property is a shorthand property for setting the width of the four borders in one declaration, can have from one to four values.

Thin	:	Defines a thin border.
Medium	:	Defines a medium border.
Thick	:	Defines a thick border.
Length	:	Allows you to define the thickness of the borders.

- **Example:**

table {border-width: thin}
All four borders will be thin.

table {border-width: thin medium thick none}
Top border will be thin, right border will be medium, bottom border will be thick, left side will have no border.

□ **Border-top-width property**

The border-top-width property sets the width of an element's top border.

Thin	:	Defines a thin top border.
Medium	:	Defines a medium top border.
Thick	:	Defines a thick top border.
Length	:	Allows you to define the thickness of the top border.

- **Example:**

Table { border-top-width : thin }
Table { border-top-width : 0.5cm }

□ **Border-bottom-width**

The border-bottom-width sets the width of an element's bottom border.
Value: Same to border-top-width property

- **Example:**

Table{border-bottom-width: thin}

□ Border-left-width property

The border-left-width property sets the width of an element's left border.
Value : Same to border-top-width property

▯ **Example:**

```
Table{border-left-width: thin}
```

□ Border-right-width property

The border-right-width property sets the width of an element's right border.
Value : Same to border-top-width property

▯ **Example:**

```
Table{ Border-right-width: thin }
```

□ Border-top property

The border-top property is a shorthand property for setting all of the properties for the top border in one declaration.

Value	Description
border-top-width border-style border-color	Sets the properties for the top border

▯ **Example:**

```
Table { border-top : thin dotted #00FF00}
Table { border-top : solid #0000FF }
```

□ Border-left property, border-right property, border-bottom property

The border-left property, border-right property, border-bottom property is a shorthand property for setting all of the properties for the left border, right border& bottom border in one declaration respectively
Value: Same to border-top property

□ Border property

The border property is a shorthand property for setting all of the properties for the four borders in one declaration.
This property cannot set a different value for each side of the border, like "margin" and "padding".

Value	Description
border-width border-style border-color	Sets the properties for the four borders

▯ **Example:**

```
P { border: thin dotted #00FF00}
```

CSS Background Properties

The CSS background properties allow you to control the background color of an element, set an image as the background, repeat a background image vertically or horizontally, and position an image on a page.

- □ background-color property
- □ background-image property
- □ background-repeat property
- □ background-attachment property
- □ background-position property

□ Background-color property

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

Color : The color value can be a color name (red), a RGB value (RGB(255,0,0)), or a hex number (#ff0000)

Transparent : The background color is transparent

▯ **Example**:

```
P{background-color: Gray}
```

□ Background-image property

The background-image property sets an image as the background. Always set a background-color to be used if the image is unavailable.

Url : The path to an image

None : No background image

▯ **Example**:

```
Body {background-image: url(stars.gif);
        background-color: #000000}
```

□ Background-repeat property

The background-repeat property sets if/how a background image will be repeated.

Repeat : The background image will be repeated vertically and horizontally

repeat-x : The background image will be repeated horizontally

repeat-y : The background image will be repeated vertically

no-repeat : The background-image will be displayed only once

▯ **Example**:

```
Body {
        background-image : url(stars.gif);
        background-repeat : repeat-x
}
```

□ Background-attachment property

The background-attachment property sets whether a background image is fixed or scrolls with the rest of the page.

Scroll **:** The background image moves when the rest of the page scrolls
Fixed **:** The background image does not move when the rest of the page scrolls.

▯ **Example:**

```
Body{
          background-image: url(stars.gif); background-attachment: scroll
         }
```

□ Background-position property

The background-position property sets the starting position of a background image. By default, a background-image is placed at the top-left corner of an element, and repeated both vertically and horizontally

1. Top Left
2. Top Center
3. Top Right
4. Center Left
5. Center Center
6. Center Right
7. Bottom Center
8. Bottom Right

If you only specify one keyword, the second value will be “center”.

x- % y-% **:** The first value is the horizontal position and the second value is the vertical. The top left corner is 0% 0%. The right bottom corner is 100%.if you only specify one value, the other value will be 50%.

x-pos y-pos **:** The first value is the horizontal position and the second value is the vertical. The top left corner is 0 0. Units can be pixels (0px 0px) or any other CSS units. If you only specify one value, the other value will be 50%. You can mix % and positions.

▯ **Example:**

```
Body {
              background-image: url(stars.gif);
              background-repeat: no-repeat;
              background-position: top left
         }
```

CSS Tables

The look of an HTML table can be greatly improved with CSS:

□ Table Borders

To specify table borders in CSS, use the border property. The example below specifies a black border for TABLE, TH, and TD elements:

- **Example**:

```
table, th, td
{
    border: 1px solid black;
}
```

Notice that the table in the example above has double borders. This is because both the table and the th/td elements have separate borders.
To display a single border for the table, use the border-collapse property.

□ Collapse Borders

The border-collapse property sets whether the table borders are collapsed into a single border or separated:

- **Example**:

```
Table{         border-collapse : collapse
} table, th, td{ border: 1px solid black }
```

□ Table Width and Height

Width and height of a table is defined by the width and height properties. The example below sets the width of the table to 100%, and the height of the "th" elements to 50px:

- **Example**:

```
table { width:100% }
th{ height:50px }
```

□ Table Text Alignment

The text in a table is aligned with the text-align and vertical-align properties. The text-align property sets the horizontal alignment, like left, right, or center:

- **Example**:

```
Td{        text-align: right }
```

The vertical-align property sets the vertical alignment, like top, bottom, or middle:

- **Example**:

```
Td{ height:50px; vertical-align: bottom}
```

□ Table Padding

To control the space between the border and content in a table, use the padding property on 'TD' and 'TH' elements:

▯ **Example:**

```
Td{ padding:15px }
```

□ Table Color

The example below specifies the color of the borders, and the text and background color of TH elements:

▯ **Example:**

```
table, td, th{ border:1px solid green }
th{ background-color : green; color : white }
```

CSS Element Positioning

The CSS positioning properties allow you to position an element. It can also place an element behind another, and specify what should happen when an element's content is too big.

Elements can be positioned using the top, bottom, left, and right properties. However, these properties will not work unless the position property is set first. They also work differently depending on the positioning method.

There are four different positioning methods.

□ Static Positioning

HTML elements are positioned static by default. A static positioned element is always positioned according to the normal flow of the page.

Static positioned elements are not affected by the top, bottom, left, and right properties.

□ Fixed Positioning

An element with fixed position is positioned relative to the browser window.

It will not move even if the window is scrolled:

▯ **Example:**

```
p.pos_fixed
{
position: fixed;
    top : 30px;
    right : 5px;
}
```

Note: IE7 and IE8 support the fixed value only if a !DOCTYPE is specified. Fixed positioned elements are removed from the normal flow. The document and other elements behave like the fixed positioned element does not exist. Fixed positioned elements can overlap other elements.

- **Relative Positioning**

A relative positioned element is positioned relative to its normal position.

- **Example**:

```
h2.pos_left{ position : relative; left:-20px }
h2.pos_right{position : relative; left:20px }
```

The content of relatively positioned elements can be moved and overlap other elements, but the reserved space for the element is still preserved in the normal flow.

- **Example**:

```
h2.pos_top{position : relative; top:-50px }
```

Relatively positioned elements are often used as container blocks for absolutely positioned elements.

- **Absolute Positioning**

An absolute position element is positioned relative to the first parent element that has a position other than static. If no such element is found, the containing block is <html>.

- **Example**:

```
h2{
        position : absolute;
        left:100px;
        top:150px;
}
```

Absolutely positioned elements are removed from the normal flow. The document and other elements behave like the absolutely positioned element does not exist. Absolutely positioned elements can overlap other elements.

- **Overlapping Elements**

When elements are positioned outside the normal flow, they can overlap other elements. The z-index property specifies the stack order of an element (which element should be placed in front of, or behind, the others).An element can have a positive or negative stack order:

- **Example**:

```
Img{
        position : absolute;
        left:0px ; top:0px;
        z-index:-1;
}
```

An element with greater stack order is always in front of an element with a lower stack order.

Note: If two positioned elements overlap, without a z-index specified, the element positioned last in the HTML code will be shown on top.

Styling Links

Links can be styled with any CSS property (e.g. color, font-family, background, etc.).Special for links are that they can be styled differently depending on what state they are in.

The four links states are…

□ a:link - a normal, unvisited link

□ a:visited - a link the user has visited

□ a:hover - a link when the user mouse over it

□ a:active - a link the moment it is clicked

□ **Example:**

```
a:link {color:#FF0000;}/* unvisited link */
a:visited {color:#00FF00;} /* visited link */
a:hover {color:#FF00FF;} /* mouse over link */
a:active {color:#0000FF;} /* selected link */
```

- **When setting the style for several link states, there are some order rules:**
  a:hover MUST come after a:link and a:visited
  a:active MUST come after a:hover

- **Common Link Styles**
  In the example above the link changes color depending on what state it is in.
  Let's go through some of the other common ways to style links:

- **Text Decoration**
  The text-decoration property is mostly used to remove underlines from links:

□ **Example:**

```
a:link {text-decoration: none;}
a:visited {text-decoration: none;}
a:hover {text-decoration: underline;}
a:active {text-decoration: underline;}
```

- **Background Color**
  The background-color property specifies the background color for links:

□ **Example:**

```
a:link {background-color:#B2FF99;}
a:visited   {background-color:#FFFF85;}
a:hover   {background-color:#FF704D;}
a:active {background-color:#FF704D;}
```

Image Style Property

Images play an important role in any webpage. Though it is not recommended to include a lot of images, but it is still important to use good images wherever required. CSS plays a good role to control image display. You can set the following image properties using CSS.

□ Border-radius property (Rounded corner of image)

□ Opacity property (Transparency in Image)

□ Filter property (Filter effect in Image)

□ Border-radius property (Rounded corner of image)

Border-radius property is used to add rounded corner effect in image's corner or any other blocks like table and div tag in CSS style.

With the use of this elements, we can set the four corner's radius separately.

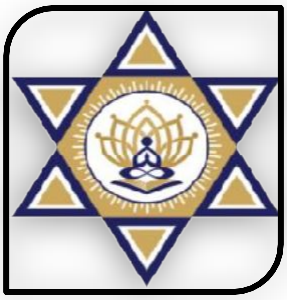
1) Border-top-left-radius
2) Border-top-right-radius
3) Border-bottom-left-radius
4) Border-bottom-right-radius

▯ **Example**:

Img { border-radius: 30% 0%30%0% }or
Img { border-top-left-radius:30% ; border-bottom-right-radius:30% }

<img src="ATMIYA_LOGO.jpg">

Output:



□ Opacity property (Transparency in Image)

The opacity property of an image is used to set the opacity of an image. This property is used to create a transparent image in Mozilla. IE uses filter:alpha(opacity=x) to create transparent images. In IE (filter:alpha(opacity=x)) x can be a value from 0 - 100. A lower value makes the element more transparent.

□ **Example:**

In Mozilla: img{opacity:0.4}

In IE: img { filter:alpha(opacity=40)}
<img src = "ATMIYA_LOGO.jpg">

□ Filter property (Photo Filter effect)

The CSS filter property adds visual effects (like blur and saturation) to an element. The filter property is not supported in Internet Explorer or Safari 5.1 and earlier. To perform visual effect operations like blur, balancing contrast, brightness, color saturation, there are filter is used in to the web page.

□ **Example:**

```
.blur { filter: blur(4px) }
.brightness { filter: brightness(250%) }
.contrast { filter: contrast(180%) }
.grayscale { filter: grayscale(100%) }
.huerotate { filter: hue-rotate(180deg)}
.invert { filter: invert(100%) }
.opacity { filter: opacity(50%) }
.saturate { filter: saturate(7) }
.sepia { filter: sepia(100%) }
.shadow { filter: drop-shadow(8px 8px 10px red)}

<img src=”ATMIYA_LOGO.jpg” class=blur>
```

Introduction of DHTML

DHTML stands for Dynamic HTML, it is totally different from HTML. The browsers which support the dynamic HTML are some of the versions of Netscape Navigator and Internet Explorer of version higher than 4.0. The DHTML is based on the properties of the HTML, JavaScript, CSS, and DOM (Document Object Model which is used to access individual elements of a document) which helps in making dynamic content. It is the combination of HTML, CSS, JS, and DOM. The DHTML make use of Dynamic object model to make changes in settings and also in properties and methods. It also makes uses of Scripting and it is also part of earlier computing trends.

□ **Advantages:**

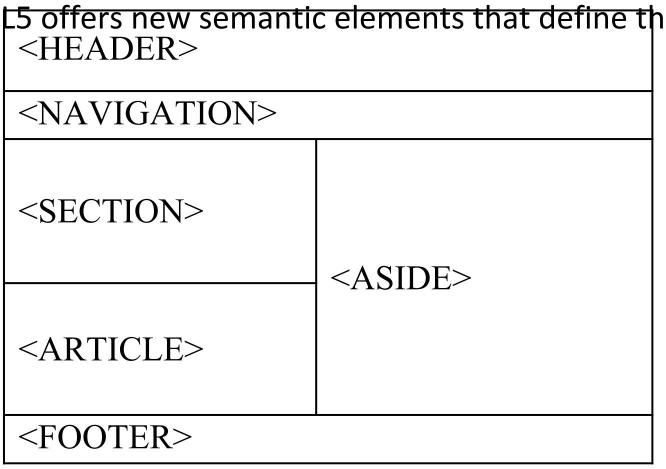
- ✔ It is supported by big browser manufacturers like Microsoft and Netscape.
- ✔ Highly flexible and easy to make changes.
- ✔ Viewer requires no extra plug-ins for browsing through the webpage that uses DHTML, they do not need any extra requirements or special software to view it.
- ✔ It is possible to modify and replace elements even after a page is loaded, it is not required to create separate pages for changing styles which in turn saves time in building pages and also reduces the number of requests that are sent to the server.

□ **Disadvantages:**

- ✔ It is not supported by all the browsers. It is supported only by recent browsers such as Netscape 6, IE 5.5, and Opera 5 like browsers.
- ✔ Learning of DHTML requires a lot of pre-requisites languages such as HTML, CSS, JS, etc. should be known to the designer before starting with DHTML which is a long and time-consuming in itself.
- ✔ Implementation of different browsers are different. So if it worked in one browser, it might not necessarily work the same way in another browser.

Page Layout

Websites often display content in multiple columns (like a magazine or newspaper). HTML5 offers new semantic elements that define the different parts of a web page:



□ **Floating Property**

- ✔ It is common to do entire web layouts using the CSS float property. Float is easy to learn - you just need to remember how the float and clear properties work.
- ✔ Floating a box will shift it to the right or left of a line, with surrounding content flowing around it.
- ✔ Floating is normally used to shift around smaller chunks within a page, such as pushing a navigation link to the right of a container, but it can also be used with bigger chunks, such as navigation columns.
- ✔ Using float, you can float: left or float: right.

□ **Example:**

div { float : left ; width : 200 }

Ch. 3 | Cascading Style Sheet

Question Bank: (1 Mark Questions)

1) Which extension is used in external CSS file? – **CSS**

2) Which tag is used for insert style sheet in HTML file? –**<Style>**

3) Which style type can apply directly into HTML tag? –**Inline style**

4) Which symbol is used to create Id selector? –**'#'**

5) Which symbol is used to create Class selector? –**' . ' (dot)**

6) Which property is used to create bold text? –**Font-weight**

7) Which property is used to define first character capital of all text of paragraph?

   –**Text-transform**

8) Which value is used to create first character capital of all text of paragraph? –**Capitalize**

9) Which value is the default value in background-repeat property? –**repeat**

10) Which value is used to repeat image vertically in CSS? –**repeat-y**

11) Which value is the default in background-attachment property in CSS? –**Scroll**

12) In which element we can apply the border-collapse property of CSS? –**Table**

13) Which three values are used with position property in CSS? –**relative, absolute, fixed**

14) Which property is used to change the stack order of overlapped element in CSS? –**Z-index**

15) Which property is used to remove the underline from hyperlink in CSS? –**Text-decoration**

[TRUE / FALSE]

16) Using CSS we can set different four side padding in HTML element. –**TRUE**

17) Small-Caps is the value of Text-transform property in CSS. –**FALSE**

18) Text-decoration is a value of Underline property in CSS. –**FALSE**

19) Letter-spacing cannot be applied in heading tag. –**FALSE**

20) Text-transform is used to remove underline from hyperlink. –**FALSE**

Question Bank: (5 Marks Questions)

1) Describe about types of Style sheet.(Page No. 46, 49)

Ans. What is CSS? – 1 marks
Three types of CSS – 4 marks

2) What is CSS? Explain style rule with syntax. (Page No. 46, 48)

Ans. What is CSS? – 2 marks
Style rules with 3 parts and syntax – 3 marks

3) Explain types of Selector in CSS. (Page No. 51)

Ans. Explanation of 2 types of Selector. – 5 marks

4) Write CSS Font properties in detail. (Page No. 52)

Ans. 5 types of font property – 5 marks

5) Write CSS Text properties in detail. (Page No. 54)

Ans. 6 types of text property – 5 marks

6) Explain multi-column properties in CSS. (Page No. 56)

Ans. 3 types of properties with an example – 5 marks

7) Explain Margin property with an example. (Page No. 58)

Ans. Explanation of Margin property. – 5 marks

8) Explain CSS Border properties in brief. (Page No. 60, 61, 62)

Ans. Three Border properties – 5 marks

9) Explain about CSS background property in brief. (Page No. 63)

Ans. Five types of background properties – 5 marks

10) Explain CSS element positioning method. Describe Overlapping elements in CSS. (Page No. 66, 67)

Ans. Element positioning method – 3 marks
Overlapping elements – 2 marks

11)Explain Styling Links in CSS with an example. (Page No. 68)

Ans. Explanation of Styling Links? – 4 marks
Example – 1 marks

12)What is Border Collapse in table? Explain Border radius property with an example in CSS.(Page No. 65, 69)

Ans. Border collapse in table – 1 marks
Different border radius property with an example – 4 marks

13)Explain image style properties using CSS. (Page No. 69)

Ans. Explanation of Image Radius, Opacity and Filter property – 5 marks

Chapter – 4
Introduction to JavaScript

What is JavaScript?

JavaScript was designed to add interactivity to HTML pages. The web pages developed in HTML contains certain limitations that, they can't be embedded with multimedia components, such as audio-video, graphics etc. To remove these limitations, we are using JavaScript in HTML. JavaScript is a client-side 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. i.e. It's an open source.

□ Client Side JavaScript:

It supplies objects to control browser.

For eg. It allows application to place elements on HTML form and give response to user events such as mouse clicks, form inputs and page navigation etc.

□ Server Side JavaScript:

It supplies objects relevant to running JavaScript on server. For eg. It allows application Relational Database (RDB), providing information from one page to another page.

□ Difference between JAVASCRIPT AND JAVA:

JAVASCRIPT	JAVA
JavaScript is Scripting Language.	Java is programming language.
It supports runtime system.	It supports compile-time system.
No distinction between types of Objects.	Objects are divided into classes and instances.
Added to any object dynamically.	Classes and instances cannot have properties and methods added dynamically.
No need to declare all variables, classes and methods.	Its needed.
JavaScript is not pure OOP.	Java is pure OOP.

- □ **How to insert JavaScript in to page?**

  To insert a JavaScript into an HTML page, we use the <script> tag. Inside the <script> tag we use the type attribute to define the scripting language.

- □ **Where to put JavaScript?**

  1) You can add JavaScript in <Head> section
  2) You can add JavaScript in <Body>section
  3) You can add JavaScript in both <Head> and <Body> section

  NOTE: You can also add External JavaScript File having extension ".js".

1) **JavaScript in <Head> Section.**

   Scripts to be executed when they are called, or when an event is triggered, go in the head section. Scripts in head section will be executed immediately while page loads into browser. If you place a script in the head section, you will ensure that the script is loaded before anyone uses it.

   ❖ **Example:**

```
<html>
        <head>
                <script type="text/JavaScript">
                        function message()
                        {
                                alert("ATMIYA Univeristy");
                        }
                </script>
        </head>
        <body onload="message()">
        </body>
</html>
```

2) **JavaScript in <Body> Section.**

   Scripts to be executed when the page loads go in the body section. If you place a script in the body section, it generates the content of a page.

   ❖ **Example:**

```
<html>
        <head>
        </head>
        <body>
                <script type="text/JavaScript">
                        document.write("This message is written by JavaScript");
                </script>
        </body>
</html>
```

3) JavaScript in both <Head> and <Body>

You can place an unlimited number of scripts in your document, so you can have scripts in both the body and the head section.

❖ **Example:**

```
<html>
        <head>
                <script type="text/JavaScript">
                        ....
                </script>
        </head>
        <body>
                <script type="text/JavaScript">
                        ....
                </script>
        </body>
</html>
```

□ **Comments in JavaScript:**

The Unexcitable part of any program is known as "Comment". Comments are generally used to give instruction about the Program, and only useful for programmer. Comments can be added to explain the JavaScript, or to make the code more readable.

```
//This is single line comment.
/* This is Multi
 line comments */
```

Some important things to know when scripting with JavaScript.

✔ **JavaScript is case sensitive:**

A function named "myfunction" is not same as "myFunction" and a variable named "my Var" is not same as "myVar".

✔ **Symbols:**

Opening symbols, like…

({ [" ' must always have a matching closing symbols, like ' "] })

✔ **White space:**

JavaScript ignores extra spaces. You can add white space to your script to make it more readable. The following lines equivalent:

```
Name="Atmiya"
Name=" Atmiya"
```

✔ **Insert special characters:**

You can also insert special characters like ("': &) with a backslash.

❖ **Example:**

document.write (“You \& I sing \”Happy Birthday\”.”);

 Output:

You & I sing “Happy Birthday”.

□ **Variables**

✔ **What is variable?**

A variable is a name given to the memory location. A variable is a “container” for information you want to store. During the execution of the program the value of the variable will be changed.

✔ **JavaScript Variables**

JavaScript variables are used to hold values or expressions. A variable can have a short name like “x” or a more descriptive name like “car_name”.

✔ **Rules for JavaScript variable names:**

Variable names are case sensitive (y and Y are two different variables)
Variable names must begin with a letter or the underscore character.

✔ **Declaring (Creating) JavaScript Variables**

You can create a variable with the “var” statement.

```
var variable-name = some value
```

❖ **Example:**

var x = 10 (now variable x holds value 10)

□ **Data Types & Literals**

The predefined identification of any variable is known as Data type. Or we can say that the data type defines the characteristics of variables. JavaScript can’t support data types. But it supports Literals. Literals are used to represent values of variables of particular data type in JavaScript. These are fixed values not variables that you literally provide in your script.

✔ **Integer Literals:**

Integer Literals can be expressed as Whole Numbers without any decimal points. It can be positive or negative number.

❖ **Example:**

```
var a = 112;
var b = - 44;
```

✔ **Floating Point Literals:**

Floating Point Literals can be expressed as numbers with decimal points. It can be positive or negative number.

❖ **Example:**

```
var a = 1.23;
var b = -4.9
```

- ✔ **Boolean Literals:**
  Boolean Literals can be expressed as only with two values i.e. TRUE or FALSE. You can just assign true or false values, it will automatically take 0 as true value and 1 as false value.
  - ❖ **Example:**

```
var a = true;
var b = false;
```

- ✔ **String Literals:**
  String Literals can be expressed as a sequence of characters. It always enclose with either single quote(‘) or double quotes(“).
  - ❖ **Example:**

```
var str1 = “Wel-Come”
var str2 = ‘Atmiya’
```

- ✔ **Array Literals:**
  Array Literals can be expressed as a list of values that are called elements of Array. The values may be numbers, strings or any type of Literal.
  - ❖ **Example:**

```
var arr = new Array(3)
arr[0] = “One”
arr[1] = “Two”
arr[2] = “Three”
```

- ✔ **Object Literals:**
  Object Literals can be expressed as a list of pairs of property names and associated values of an Object enclosed in { }.
  NOTE: It is just a User Defined Object which is shown later on.

- ✔ **NULL Literals:**
  Null Literal consists of single value, which identifies a NULL i.e. empty. It is used to set initial value of variable that is different from other valid values.
  - ❖ **Example:**
    - var temp = null;
    - document. write(“Value of temp is :”+temp);

Dialog Boxes

“Dialog Box” is nothing but a small separate window that contents provided by end user. In JavaScript, Dialog Boxes are very useful because they allow the script to interact with visitors where the visitor must provide a response to Dialog Box before script can continue processing. All Dialog Boxes in JavaScript are Modal Dialog Boxes and don’t permit anything else to happen until Dialog Box receives a response from the user. JavaScript provides the ability to get user input or display small amount of text to the user by using Dialog Boxes.

There are three types of Dialog Boxes provided by JavaScript:

1) Alert Dialog Box.
2) Confirm Dialog Box.
3) Prompt Dialog Box.

1. ALERT Dialog Box:

Alert Dialog Box is use to display some information on web browser window. Alert Dialog Box is used to give warning messages to end users. The alert Dialog Box displays string passed to the alert () method as well as OK button. The JavaScript and HTML, in which this code is held, will not continue processing until OK button pressed.

Use of Alert dialog box:

- ✔ A message to display when user inputs invalid information.
- ✔ An invalid result from Calculation.
- ✔ A warning that a service is not available.

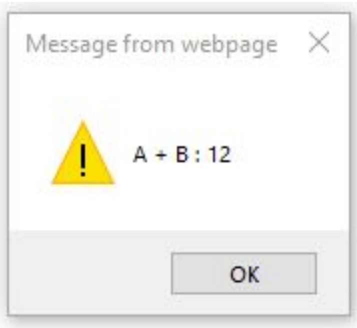
⇨ **Syntax:**

```
alert("message");
```

❖ **Example:**

```
<html>
<body>
        <script language="JavaScript">
                var a=5,b=7
                var c = a+b;
                alert("A + B : "+c);
        </script>
</body>
</html>
```

□ **Output:**



A + B : 12

2. CONFIRM Dialog Box:

A Confirm Dialog Box is often used if we want the user to verify or accept something. A Confirm Dialog Box display predefined message with OK and CANCEL button. When a confirm Dialog Box pop up, the user will have to click either OK or CANCEL to proceed. If you press the OK button then it will return TRUE if you press CANCEL button then it will return FALSE.

Use of Confirm dialog box:

A message will display when we want any confirmation from end user.

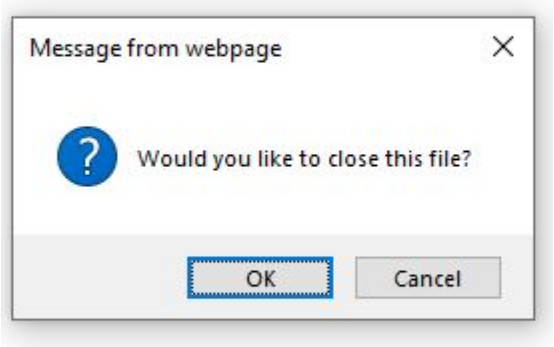
⇨ **Syntax:**

```
confirm ("Message");
```

❖ **Example:**

```
<html>
<body>
        <script language="JavaScript">
                confirm("Would you like to close this file?");
        </script></body></html>
```

 Output:



Message from webpage

Would you like to close this file?

3. PROMPT Dialog Box:

When you want to get the value from user at that time, the prompt dialog box is used. Prompt dialog box will display with a pre-defined message, a text box for user input and with two command buttons.(OK,CANCEL). When a prompt dialog box pops up, the user will have to click either OK or CANCEL button to proceed after entering input value. If the user clicks OK, then the value that user entered will stored inside the variable. If the user clicks CANCEL, "NULL" will stored inside the variable.

Use of Prompt dialog box:

In online games, ask for end user to enter name.

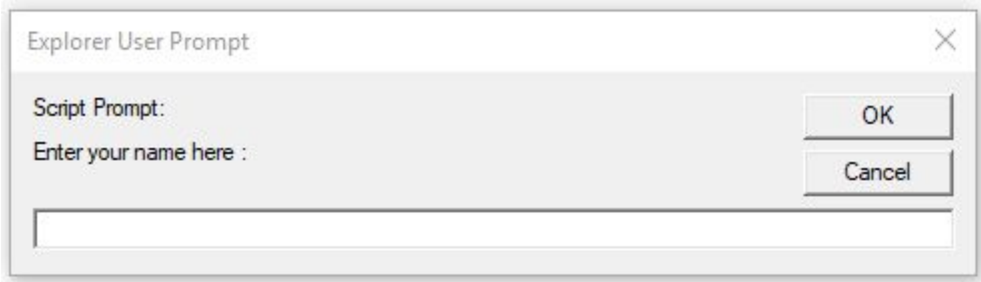
⇨ **Syntax:**

```
var name=prompt("Message","default value");
```

❖ **Example:**

```
<html>
        <body>
         <script language=”JavaScript”>
                  nm=prompt(“Enter your name here :”,””)
           </script>
        </body>
</html>
```

 Output:



Script Prompt:
Enter your name here :

Decision Statements

Some times when you write code, you want to perform different actions for different decisions, at that time you can use conditional statements in your code. JavaScript provides a complete range of basic programming constructs. They are as given below:

If statement	Use this statement if you want to execute some code only if a specified condition is true.
If …else statement	Use this statement if you want to execute some code if the true and another code if the condition is false.
If…else if…else statement	Use this statement if you want to select one or many blocks of code to be executed.
Switch statement	Use this statement if you want to select one of many blocks of code to be executed.

□ **If statement:**

You can use the if statement if you want to execute some code only when the condition is true. The block of statements is executed only when the specified condition is true.

⇨ **Syntax:**

```
if (Condition)
{
        Block of statements
}
```

If the condition is evaluated to true than the block of statements will be executed, then control will transfer out of block. If the condition is evaluated to false than the block of code will be skipped and the control will transfer out of block by skipping the block of statements.

❖ **Example:**

```
<script language="JavaScript">
        var a=10,b=20;
        if(a>b)
        {
                document.write("A is big");
        }
</script>
```

□ If… else statement:

The 'if else' construct is an extension of simple if statement. If…else construct is used to execute some code when the condition is true or false. You can handle the true block or false block with the help of if… else construct.

⇨ **Syntax:**

```
if(condition)
{
        Code for true part
}
else
{
        Code for false part
}
```

If the condition is evaluated to true then the code for true block will be executed. If the condition is evaluated to false then by skipping the true block the control will transfer in else part and the code for false block will be executed.

❖ **Example:**

```
<script language="JavaScript">
        var a=20,b=10;
        if (a>b)
        {
                document.write("A is Big");
        }
        else
        {
                document.write("B is Big");
        }
</script>
```

□ **if…else if…else statement:**

You can use the if…else if…else statement if you want to select one of many sets of lines to execute. This structure gives you facility to specify the condition with each else statement.

⇨ **Syntax:**

```
if(condition1)
{
        code for block 1
}
else if (condition 2)
{
        code for block 2
}
else
{
        code to be executed if both condition are not true.
}
```

In this structure the condition 1 will be evaluated first, if the condition 1 is evaluated to true then the code for block 1 will be execute. If the condition 1 is evaluated to true then and then the control proceeds to the condition 2. As soon as the true condition is found then the respected block of code will be executed and then control will go out of the whole if..else if…else block. If all the conditions are evaluated to false then the final else block will be executed.

□ **Switch statement**

You should use the switch statement if you want to select one of many blocks of code to be executed. First we have a single expression that is evaluated once. The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed. Use break to prevent the code from running into the next case automatically. If expression is not match with any case then default block is executed.

⇨ **Syntax:**

```
switch(n)
{
     case 1:
            Code block 1
            break;
     case 2:
            Code block 2
            break;
     default:
            Code to be executed if n is different from case 1 & 2.
}
```

❖ **Example:**

```
<script language=”JavaScript”>
          switch(ch)
          {
                    case ‘1’:
                              c=a+b;
                              document.write(“Addition is”+c);
                              break;
                    case ‘2’:
                              c=a-b;
                              document.write(“Subtraction is”+c);
                              break;
                    case ‘3’:
                              c=a*b;
                              document.write(“Mulitplication is”+c);
                              break;
                    default:
                              document.write(“Invalid choice”);
          }
</script>
```

Looping Structure in Java script:

□ **What is Loop?**

When you want to perform a particular task no. of times at that time the concepts of loop is used. JavaScript supports 3 types of looping structure they are as given below:

1) While()
2) Do…while()
3) For()

4) While () Statement:

The while() statement is the simplest one from all among the looping structure. The while statement is used to execute the statements when the condition is evaluated to true. This is also known as an “Entry Controlled Looping Statement” because the condition is evaluated first and if the condition is evaluated to true then & then the body of the loop will be executed.

⇨ **Syntax:**

```
While (Condition)
{
          Body of the Loop
}
```

❖ **Example:**

```
<script language=”JavaScript”>
        var x=1;
        while(x<=10)
        {
                document.write(“<br>”+x);
                x++ ;
        }
</script>
```

2) Do…while() statement:

The do…while() statement is used to execute the statements no. of times as same as while() statement. But the difference is that the location of condition. This is the reason that’s why it is called “Exit control Loop statement”.

⇨ **Syntax:**

```
 do
 {
        Body of the Loop
 }while (Condition);
```

❖ **Example:**

```
<script language=”JavaScript”>
        var x=1;
        do
        {
                document.write(“<br>”+x);
                x+= 2;
        }while(x<=20);
</script>
```

3) For() statement:

The for() statement is another “Entry Controlled Loop” statement. It is used when you know in advanced how many times you want to iterate the loop. It allows to specify intitialization, testing and increment/decrement, just within a single line of statement.

⇨ **Syntax:**

```
for (initialization;condition;incre/decre )
{
        body of the loop
}
```

❖ **Example:**

```
<script language="JavaScript">
        for(i=0;i<=5;i++)
        {
                document.write("Atmiya <br>")
        }
</script>
```

Array in Java script:

An Array is an ordered set of values that we refers with its name and index. JavaScript does not have any explicit array data type. However, we can use the predefined Array Object and its methods to work with arrays in our Application. The array object has so many methods for manipulating arrays in various ways. The array object is used to store set of values in single variable name. The array Literals are also Array Object.

Creating an Array:

We can create array object in either of three ways:

⇨ **Syntax:**

1) arrayObjectName = new Array()
2) arrayObjectName = new Array(arrayLength)
3) arrayObjectName = new Array(Element0, Element1,..,Element N)

arrayObjectName : It is either the name of a new Object or existing Object.
arrayLength : It is an initial length of array.
Element 0, Element 1 : It is a list of values for array elements.

❖ **Example:**

```
var emp = new Array()
emp[0] = "Atmiya"
emp[1] = "Virani"
document.write("<br>First Employee Name : "+emp[0])
document.write("<br>Second Employee Name : "+emp[1])
document.write("<br>Third Employee Name : "+emp[2])
```

Output:

```
Atmiya
Virani
Undefine
```

✔ **Populating an Array:**

We can populate an array by assigning values to its elements. The values which we can assign to its elements may be string and numbers.

❖ **Example:**

```
var myArray = new Array("AAA",12,"bbb");
```

✔ **Accessing an Array:**

We can refer to a particular element in an array by referring to the name of array and index number.

User Define Functions in Java script:

□ What is Function?

"Function" is self-contain block of statements that perform a particular task. Function provides the ability to combine no. of statements in single unit. JavaScript provides Functions through which you can perform certain task. JavaScript allows users to create our own Functions, are called "User Defined Functions". User Defined Function first need to be declared after that function can be invoked by calling it using its name when it was declared. The data needed to execute a function is passed as a Parameter.

⇨ **Syntax:**

```
function function-name(parameter list)
{
        block of statements
        [return statement]
}
```

- Functions are declared and created using "**function"** keyword.
- "**function-name**" is the appropriate name of the function.
- **parameter-list**" are optional and passed to the function appears in parentheses and commas separate member of list.
- "**return**" statement can be used to return any valid expression that evaluates a single value.

❖ **Example:**

```
<html>
  <head>
       <script language="javacript">
              function displayMsg( )
              {
                     alert("Welcom to ATMIYA University");
              }
       </script>
  </head>
  <body>
       <script language="JavaScript">
              displayMsg()
       </script>
  </body>
</html>
```

✔ **Note:** A function code will be executed only by …
When user fires any event.
When a call to that function (anywhere within a page).

□ Scope of Variables in Functions:

The variable declared or assigned outside the function, is called “Global variable”.
The variable declared or assigned inside the function, is called “Local variable”.

❖ **Example:**

```
<script language=”JavaScript”>
        var x = 10;     // global variable
        function scope()
        {
                var y = 20; // local variable
                document.write(y);          // 20
                document.write(x);          // 10
        }
        document.write(y);                  //undefined.
        document.write(x);                  //10
</script>
```

✔ **Passing parameters and Return values:**
We can pass no. of parameters inside the function and we can return values also.

❖ **Example:**

```
<html>
        <head>
                <script>
                        function scope(a,b)
                        {
                                c = a+b;
                                return c ;
                        }
                </script>
        </head>
        <body>
                <script>
                        c = scope(10,20);
                        document.write("Sum : "+c);
                </script>
        </body>
</html>
```

□ TYPES OF FUNCTION

- Function Without Argument and Without Return Value
- Function With Arguments Without Return Value
- Function With Arguments With Return Value
- Function Without Arguments With Return Value

✔ Function without argument and without return value

```
<body>
      <script language="JavaScript">
            var name=""
            function hello()
            {
                  name = prompt("Enter your name")
                  alert("Welcome " + name)
            }
            function bye()
            {
                  alert("Good bye " + name)
            }
            hello()
            bye()
      </script>
  </body>
```

✔ Function with arguments and without return value

```
<body>
    <script language="JavaScript">
          var name=""
          function hello(name)
          {
                alert("Welcome " + name)
          }
          function bye(name)
          {
                alert("Good bye " + name)
          }
          hello("World")
          name="Friends"
          bye(name)
    </script>
</body>
```

- ✔ **Function with arguments with return value**

```
<body>
        <script language="JavaScript">
                var name=""
                function hello(usrname)
                {
                        return usrname+" Atmiya"
                }
                name=hello("Welcome")
                document.write(name)
        </script>
</body>
```

- ✔ **Function without arguments with return**

```
<body>
        <script language="JavaScript">
                var name=""
                function hello()
                {
                        return "Atmiya"
                }
                name=hello()
                document.write(name)
        </script>
</body>
```

Built-in Functions

- **eval() :**

The eval() function evaluates a string without reference to a particular objects.

⇨ **Syntax:**

```
eval(string-expression)
```

❖ **Example:**

```
<script language=”JavaScript”>
    var no1 = prompt(“Enter First Value”);
    var no2 = prompt(“Enter Second Value”);
    sum = eval(no1) + eval(no2);
    document.write(“Sum : ”+sum);
</script>
```

- **parseInt():**

The parseInt() function parses a string and returns an integer.

⇨ **Syntax:**

```
parseInt(string)
```

❖ **Example:**

```
<script language=”JavaScript">
     document.write(parseInt("15") + "<br>");
     document.write(parseInt("15.34") + "<br>");
     document.write(parseInt("25 35 45") + "<br>");
     document.write(parseInt(" 100 ") + "<br>");
     document.write(parseInt("30 years") + "<br>");
     document.write(parseInt("Hello") + "<br>");
     document.write(parseInt("015") + "<br>");
</script>
```

□ **Output:**

```
15
15
25
100
30
NaN
15
```

□ **parseFloat():**

The parseFloat() function parses a string and returns a floating point number. This function determines if the first character in the specified string is a number. If it is, it parses the string until it reaches the end of the number, and returns the number as a number, not as a string.

⇨ **Syntax:**

```
parseFloat(string);
```

❖ **Example:**

```
<script type="text/JavaScript">
          document.write(parseFloat("15") + "<br>");
          document.write(parseFloat("15.46") + "<br>");
          document.write(parseFloat("35 45 55") + "<br>");
          document.write(parseFloat(" 100 ") + "<br>");
          document.write(parseFloat("30 years") + "<br>");
          document.write(parseFloat("Hello") + "<br>");
</script>
```

□ **Output:**

```
15
15.46
35
100
30
NaN
```

JavaScript Objects

□ **What is Object?**

"Object" is just a special kind of data. Each Object has its own state and behavior. State of the Object refers as Property. Behavior of the Object refers as Methods.

Properties:

Properties are used to define the characteristics of an Object. Properties are used to access the data values present in the object.

To access the property of an Object

objectName.propertyName

Methods:

Methods are the actions that can be performed on Objects. Object's behavior, we can change by using accessing their methods.

To access the method of an Object

objectName.methodName

JavaScript is object-oriented scripting language. So JavaScript allows you to use with objects. JavaScript provides two types of Objects.

Built-In Objects: JavaScript provides two types of Built-In Objects.

Native Objects

- String Object
- Math Object
- Array Object
- Date Object

Browser Objects

- Window Object
- History Object
- Location Object
- Screen Object
- Navigator Object
- Document Object

□ **Built-In Objects :**

□ **String Object:**

This object is used to manipulate the String.

Property:

Length	Returns the no. of characters in a String.

Methods:

- bold:
- italics:
- strike:
- sup:
- sub:
- fontcolor:
- fontsize:
- link:
- blink:
- charAt:
- concat:
- indexOf:
- lastIndexOf:
- match:
- replace:
- search:
- slice:
- substr:
- substring:
- toLowercase:
- toUppercase:

Bold: This method is used to display a string in a bold font.

⇨ **Syntax:**

Stringobject.bold()

Italics: This method is used to display a string in italics font.

⇨ **Syntax:**

Stringobject.italics()

Big: This method is used to display a string in big font.

⇨ **Syntax:**

Stringobject.big()

Small: This method is used to display a string in small font.

⇨ **Syntax:**

Stringobject.small()

Sub: This method is used to give subscript effect to the string.

⇨ **Syntax:**

Stringobject.sub()

Sup: This method is used to give superscript effect to the string

⇨ **Syntax:**

Stringobject.sup()

Fontsize: This method is used to specify the size of font.

⇨ **Syntax:**

Stringobject.fontsize(size)

Fontcolor: This method is used to specify the color of font.

⇨ **Syntax:**

Stringobject.fontcolor(color)

Blink: This method is used to display a blinking string. It can't support by the internet explorer.

⇨ **Syntax:**

Stringobject.blink()

- **Link:** This method is used to display a string as a hyperlink.
  - ⇨ **Syntax**:

    ```
    Stringobject.link()
    ```

- **Strike:** This method is used to give strikethrough effects to the string.
  - ⇨ **Syntax**:

    ```
    Stringobject.strike()
    ```

- ❖ **Example of all above String method:**

```
<script language=”JavaScript”>
            var myString = "ATMIYA University"
            var myString2 = " Happy World”
            var str=”st”;
            var str1=”2”;
            document.write("My String is: "+myString);
            document.write("Bold: "+myString.bold());
            document.write("Italics: "+myString.italics());
            document.write("Strike: "+myString.strike());
            document.write("Superscript: 1"+str.sup());
            document.write("Subscript: H"+str1.sub()+”O”);
            document.write("FontColor:"+myString.fontcolor("red”));
            document.write("FontSize: "+myString.fontsize(15));
            document.write("Link:"+myString.link("JavaScript") );
   </script>
```

- **CharAt:** The charAt () method is used to find the character at specified position. This method is case sensitive.
  - ⇨ **Syntax**:

    ```
    Stringobject.charAt(index)
    ```

  - ❖ **Example**:

    ```
    Var str=”My string”
    document. write(charAt(3));
    ```

  - **Output**:

    ```
    S
    ```

- **Concat:** This method is used to concat (merge) two or more strings.
  - ⇨ **Syntax**:

    ```
    Stringobject.concat (stringX,stringX,……X)
    ```

  - ❖ **Example**:

    ```
    Var st1=”This is“
    Var st2=” Computer”
    document.write(st1.concat(st2))
    ```

  - **Output**:

    ```
    This is computer
    ```

- **indexOf:** This method returns the position of the first occurrence of a specified string value in a string object. It is case sensitive. Returns -1 if no match found.

⇨ **Syntax:**

```
Stringobject.indexOf(stringvalue,startIndex)
```

Here the string value is required and start index is optional.

❖ **Example:**

```
var str = "ATMIYA";
document.write(str.indexOf(“M”));
```

 Output:

```
2
```

- **lastIndexOf:** This method returns position of the last occurrence of a specified string value. It is case sensitive. It returns -1 if no match found.

⇨ **Syntax:**

```
Stringobject.lastIndexOf(stringValue,startIndex)
```

❖ **Example:**

```
var str=”Hello World”
document.write(str.lastIndexOf(“o”))
```

 Output:

```
7
```

- **Match:** It searches for specified string value in a string. It is similar to indexOf() and lastIndexOf(), but returns specified string, instead of position of the string. It is case sensitive method. Returns null if no match found.

⇨ **Syntax:**

```
stringObject.match(search value)
```

❖ **Example:**

```
var str=”Have A Nice Day”;
document.write(str.match(“Nice”))
document.write(str.match(“nice”))
```

 Output:

```
Nice
Null
```

- **Replace:** The replace() method is used to replace some characters with some other characters in a string. The replace() method is case sensitive.

⇨ **Syntax:**

```
stringObject.replace (findstring, newstring)
```

❖ **Example:**

```
var str=”Play Cricket!”;
document.write (str.repalce (“Cricket”,”Chess”))
```

 Output:

```
Play Chess!
```

- **Search:** The search() method is used to search a string for specified value. The search() method returns the position of the specified value in the. If no match was found it returns -1. This method is case sensitive.

⇨ **Syntax**:

```
StringObject.search(searchvalue)
```

❖ **Example**:

```
var str=”Have a nice day”
document.write (str.search (“day”))
```

□ **Output**:

```
12
```

- **Slice:** The slice() method extracts a part of a string and returns the extracted part in a new string. You can use negative numbers to select from the end of the string. If the end is not specified, slice() selects all characters from the specified start position and to the end of the string.

⇨ **Syntax**:

```
stringObject.slice(start,end)
```

❖ **Example**:

```
var str=”Hello, How are you?”
document.write(str.slice(7));
document.write(str.slice(7,12));
```

□ **Output**:

```
How are you?
How a
```

- **Substr:** The substr() method extracts a specified number of characters in a string. To extracts from the end of the string, use a negative start number. The start index starts at 0. If the length parameter is omitted, this method extracts to the end of the string.

⇨ **Syntax**:

```
stringObject.substr(start,length)
```

❖ **Example**:

```
var str=”Good Morning”;
document.write(str.substr(3));
document.write(str.substr(3,7));
```

□ **Output**:

```
d Morning
d Morni
```

- **Substring:** The substring () method extracts the characters in a string between two specified indices. To extract the character from the end of the string, use negative start number. The start index starts at 0. If the stop parameter is omitted, this method extracts to the end of the string.

⇨ **Syntax**:

```
stringObject.substring(start,stop)
```

❖ **Example**:

```
var str=”Hello world”;
document.write(str.substr(3))
document.write(str.substr(7,3))
```

□ **Output**:

```
lo world
lo w
```

- **toLowerCase:** The toLowerCase() method is used to display a string in lowercase letters.

⇨ **Syntax**:

```
stringObject.toLowerCase()
```

❖ **Example**:

```
var str=”Hello World!”
```

□ **Output**:

```
hello world!
```

- **toUpperCase:** The toUpperCase() method is used to display a string in uppercase letters.

⇨ **Syntax**:

```
stringObject.toUpperCase()
```

❖ **Example**:

```
var str=”Hello World!”
```

□ **Output**:

```
HELLO WORLD!
```

STRING OBJECT PROPERTY:

- **length:** This property is used to count the length of the specified string.

⇨ **Syntax**:

```
stringObject.length
```

❖ **Example**:

```
var str=”Hello World”;
```

□ **Output**:

```
11
```

□ **Math Object:**

Math Object provides standard library of mathematical constants and functions. The “Constants” are defined as properties of Math Object. The “Functions” are defined as methods of Math Object.

Methods:

- **abs:** It returns absolute value of a number.
  - ⇨ **Syntax**:
    ```
    Math.abs(number)
    ```
  - ❖ **Example**:
    ```
    document.write(Math.abs(-7.25));
    document.write(Math.abs(7.25));
    ```
- **ceil:** It returns value of number rounded UPWARDS to the nearest integer.
  - ⇨ **Syntax**:
    ```
    Math.ceil(number)
    ```
  - ❖ **Example**:
    ```
    document.write(Math.ceil(0.60))
    document.write(Math.ceil(0.40))
    document.write(Math.ceil(-  5.1))
    document.write(Math.ceil(5.1))
    ```
- **floor:** It returns the value of number rounded DOWNWARDS to the nearest integer.
  - ⇨ **Syntax**:
    ```
    Math.floor(number)
    ```
  - ❖ **Example**:
    ```
    document.write(Math.  floor  (0.40))
    document.write(Math.  floor  (- 5.1))
    document.write(Math. floor (5.1))
    ```
- **cos:** It returns the value of cosine of number. It returns numeric value between -1 to 1.
  - ⇨ **Syntax**:
    ```
    Math.cos (number)
    ```
  - ❖ **Example**:
    ```
    document.write(Math.cos (3))
    ```
- **sin:** It returns the value of sine of a number. It returns the numeric value between -1 to 1.
  - ⇨ **Syntax**:
    ```
    Math.sin(number)
    ```
  - ❖ **Example**:
    ```
    document.write(Math.sin (3))
    ```
- **tan:** It returns the number that represents tangent of an angle.
  - ⇨ **Syntax**:
    ```
    Math.tan(number)
    ```
  - ❖ **Example**:
    ```
    document.write(Math.tan (1))
    ```

- **log:** It returns the natural log (base E) of number. If the number is negative(-), NaN is returned.
  - ⇨ **Syntax:**

    ```
    Math.log(number)
    ```
  - ❖ **Example:**

    ```
    document.write(Math.log(1))
    ```

- **pow:** It returns the value of X to power of Y.
  - ⇨ **Syntax:**

    ```
    Math.pow(x,y)
    ```
  - ❖ **Example:**

    ```
    document.write(Math.pow(2,3));
    document.write(Math.pow(5,2));
    ```

- **random:** It returns a random number between 0 to 1.
  - ⇨ **Syntax:**

    ```
    Math.random()
    ```
  - ❖ **Example:**

    ```
    document.write(Math.random());
    ```

- **max:** It returns the number with highest value of two specified numbers.
  - ⇨ **Syntax:**

    ```
    Math.max(x,y)
    ```
  - ❖ **Example:**

    ```
    document.write(Math.max(10,15));
    ```

- **min:** It returns the number with lowest value of two specified numbers.
  - ⇨ **Syntax:**

    ```
    Math.min(x,y)
    ```
  - ❖ **Example:**

    ```
    document.write(Math.min(10,15));
    ```

- **round:** It rounds the number to nearest integer.
  - ⇨ **Syntax:**

    ```
    Math.round(number)
    ```
  - ❖ **Example:**

    ```
    document.write(Math.round(0.60));
    document.write(Math.round(0.49));
    ```

- **sqrt:** It returns the square root of a number.
  - ⇨ **Syntax:**

    ```
    Math.sqrt(number)
    ```
  - ❖ **Example:**

    ```
    document.write(Math.sqrt(16));
    ```

Array Object

An array is the collection of elements having similar datatypes or we can say that array provides the facility to store multiple values in single variable name.

Property:

- **length:**
  - It returns the length of the Array.

⇨ **Syntax:**

```
ArrayObject.length
```

❖ **Example:**

```
<script   languagae=”javacript”>
          var subArr = new Array()
          arr[0] = “PHP”
          arr[1] = “C”
          arr[2] = “VB”
          document.write(subArr.length)
</script>
```

Output:

```
3
```

Methods:

- **concat():** It is used to join two or more Arrays. This method does not change the original arrays. It only returns a copy of joined arrays.

⇨ **Syntax:**

```
ArrayObject.concat()
```

❖ **Example:**

```
<script languagae=”javacript”>
          var subArr = new Array()
          subArr[0] = “PHP”
          subArr[1] = “C”
          subArr [2] = “VB”
          var moreArr = new Array()
          moreArr[0] = “Java”
          moreArr[1] = “Oracle”
          document.write(subArr.concat(moreArr))
          document.write(subArr.length)
</script>
```

Output:

```
PHP,C,VB,Java,Oracle
3
```

- **join():** It is used to put all the elements of array into string. The elements will be seprated by Seprator, comma(,) is default if you omitted the parameter.

⇨ **Syntax:**

```
ArrayObject.join(seprator)
```

❖ **Example**:

```
    <script languagae=”javacript”>
        Var subArr = new Array()
        arr[0] = “PHP”
        arr[1] = “C”
        arr[2] = “VB”
        document.write(subArr.join(“:”))
  </script>
```

 Output:

```
PHP: C :VB
```

pop(): It is used to remove and return “Last Element” of Array. This method changes the length of original Array.

⇨ **Syntax**:

```
ArrayObject.pop()
```

❖ **Example**:

```
<script   languagae=”javacript”>
        var subArr = new Array()
        arr[0] = “PHP”
        arr[1] = “C”
        arr[2] = “VB”
        document.write(subArr.pop())
</script>
```

 Output:

```
VB
```

shift(): It is used to remove and return “First Element” of Array. This method changes the length of original Array.

⇨ **Syntax**:

```
ArrayObject.shift()
```

❖ **Example**:

```
    <script languagae=”javacript”>
        var subArr = new Array()
        arr[0] = “PHP”
        arr[1] = “C”
        arr[2] = “VB”
        document.write(subArr.shift())
   </script>
```

 Output:

```
PHP
```

push(): It is used to add one or more elements to end of the Array and returns new length.

⇨ **Syntax**:

```
ArrayObject.push(newElement1,newElement2,…)
```

❖ **Example**:

```
<script   languagae=”javacript”>
          var subArr = new Array()
          arr[0] = “PHP”
          arr[1] = “C”
          arr[2] = “VB”
          document.write(subArr.push(“java”))
</script>
```

 Output:

```
4
```

unshift(): It is used to add one or more elements to the beginning of the Array and returns new length.

⇨ **Syntax**:

```
ArrayObject.unshift(newElement1,newElement2,…)
```

❖ **Example**:

```
<script   languagae=”javacript”>
          var subArr = new Array()
          arr[0] = “PHP”
          arr[1] = “C”
          arr[2] = “VB”
          document.write(subArr.unshift(“oracle”))
</script>
```

 Output:

```
4
```

reverse(): It is used to reverse the order of elements in Array. It changes the original Array.

⇨ **Syntax**:

```
ArrayObject.reverse()
```

❖ **Example**:

```
<script   languagae=”javacript”>
          var subArr = new Array()
          arr[0] = “PHP”
          arr[1] = “C”
          arr[2] = “VB”
          document.write(subArr.reverse())
</script>
```

 Output:

```
VB,C,PHP
```

slice(): It is used to extract a section of an Arrya and returns a new Array.

⇨ **Syntax**:

```
ArrayObject.slice(start_index,end_index)
```

❖ **Example**:

```
<script   languagae=”javacript”>
        var subArr = new Array()
        arr[0] = “PHP”
        arr[1] = “C”
        arr[2] = “VB”
        document.write(subArr.slice(0,1))
</script>
```

□ **Output**:

```
PHP
```

- **sort():** It is used to sort the elements of an Array. This method will sort the elements alphabetically by default. Numbers will not be sorted correctly. To sort numbers, we must create function that compare numbers. After sort() method, Array is changed.

⇨ **Syntax**:

```
ArrayObject.sort()
```

❖ **Example**:

```
    <script languagae=”javacript”>
        var subArr = new Array()
        arr[0] = “PHP”
        arr[1] = “C”
        arr[2] = “VB”
        document.write(subArr.sort())
</script>
```

□ **Output**:

```
C,PHP,VB
```

□ **DATE OBJECT:**

JavaScript does not have the datatype as Date. However, we can use Date object & it’s methods to work with dates and times. Some important things about methods of date object.

Seconds & minutes	0 to 59
Hours	0 to 23
Day	0(Sunday) to 6(Saturday)
Date	1 to 31
Month	0(January) to 11 (December)
Year	Years since 1900

Methods:

- **Date():** This method returns today’s date and time.

⇨ **Syntax**:

```
dateObject.Date()
```

- **Example**:

```
var d=new Date()
document.write(d.Date())
```

- **getDate():** The getDate() method returns the day of the month. The value returned by getDate() us a number between 1 and 31. This method is always used in conjunction with a Date object.

  ⇨ **Syntax**:

  ```
  dateObject.getDate()
  ```

  - **Example**:

  ```
  Var d=new Date()
  document.write(d.getDate())
  ```

- **getDay():** The getDay() method returns a number that represents the day of the week. The value returned by getDay() is a number between 0 and 6. Sunday is 0, Monday is 1 and so on. This method is always used in conjunction with a Date.

  ⇨ **Syntax**:

  ```
  dateObject.getDay()
  ```

  - **Example**:

  ```
  Var d=new Date()
  document.write(d.getDay())
  ```

- **getMonth():**The getMonth() method reurns the month as a number.
  - The value returned by the getMonth() is a number between 0 and 11. January is 0 and February is 1 and so on..
  - This method is always used in conjunction with a Date object.

  ⇨ **Syntax**:

  ```
  dateObject.getMonth()
  ```

  - **Example**:

  ```
  Var d=new Date()
  document.write(d.getMonth())
  ```

- **getYear():** The getYear() method returns the month, as a number. The value returned by getMonth() is a number between 0 & 11. January is 0, February is 1 and So on. This method is always used in conjunction with date object.

  ⇨ **Syntax**:

  ```
  dateObjecct.getYear()
  ```

  - **Example**:

  ```
  var d=new Date()
  document.write(d.getYear())
  ```

- **getFullYear():** The getFullYear() method returns a four digit number that represents a year. This method is always used in conjunction with a date object.

⇨ **Syntax**:

```
dateObject.getFullYear()
```

❖ **Example**:

```
Var d=new Date()
document.write(d.getFullYear())
```

- **getHours():** The getHours() method returns the hour of a time. The value returned by this method is a number that represents the Hour of a day. This method is always used in conjunction with date object.

  ⇨ **Syntax**:

  ```
  dateObject.getHours()
  ```

  ❖ **Example**:

  ```
  var d=new Date()
  document.write(d.getHours())
  ```

- **getMinutes():** The getMinutes() method returns the minutes of a time. The value returned by the getMinutes() is a number that represents minutes of a current Hour. This method is used with conjunction with the date object.

  ⇨ **Syntax**:

  ```
  dateObject.getMinutes();
  ```

  ❖ **Example**:

  ```
  var d=new Date();
  document.write(d.getMinutes());
  ```

- **getSeconds():** This method returns the seconds of a time. The value returned by this method is a number that represents the second of a current time. This method is always used in conjunction with date object.

  ⇨ **Syntax**:

  ```
  dateObject.getSeconds();
  ```

  ❖ **Example**:

  ```
  var d=new Date();
  document.write(d.getSeconds());
  ```

- **getMilliseconds():** The getMilliseconds() returns the milliseconds of a time. The value returned by this method is a number that represents a milliseconds of a current time. This method is always used in conjunction with date object.

  ⇨ **Syntax**:

  ```
  dateobject.getMilliseconds()
  ```

  ❖ **Example**:

  ```
  var d=new Date()
  document.write(d.getMilliseconds())
  ```

- **getTime():** The getTime() returns the number of milliseconds since midnight of January 1, 1970. This method is always used in conjunction with Date object.

  ⇨ **Syntax**:

  ```
  dateObject.getTime()
  ```

❖ **Example**:

```
var d=new Date()
document.write(d.getTime())
```

- **setDate():** The setDate() method is used to set the day of the month. The value set by setDate() is a number between 1 and 31, that represents the day number of month. This method is always used in conjunction with Date object.

⇨ **Syntax**:

```
dateObject.setDate(day)
```

❖ **Example**:

```
var d=new Date()
d.setDate(4)
```

- **setMonth():** The setMonth() method is used to set the month. You can set month by specifying the value of month and value of day. The month value in setMonth() is a number between 0 and 11. January is 0, February is 1 and so on… and the day value in setMonth() is a number between 1 and 31, represents the day of month. This method is always used in conjunction with Date object. Here the parameter day is optional and month is required.

⇨ **Syntax**:

```
dateObject.setMonth(month,day);
```

❖ **Example**:

```
var d=new Date()
d.setMonth(8)
```

- **setFullYear():** The setFullYear() method is used to set the year. You can set the year by specifying the year, month and day value. The year value represents the four digit value specifies the year. The month value represents the number between the 0 and 11 specifies the month. The day value represents the number between 1 and 31, specifies the date. Here the parameter year is required and month and day are optional.

⇨ **Syntax**:

```
dateObject.setFullYear(year,month,day);
```

❖ **Example**:

```
var d=new Date();
d.setFullYear(2009);
```

- **setYear():** The setYear() method is used to set the year. The value set by the setYear() is a numeric value that represents the year value. Always used in conjunction with date Object.

⇨ **Syntax**:

```
dateObject.setYear(year);
```

❖ **Example**:

```
var d=new Date();
d.setYear(year);
```

- **setHours():** The setHours() method is used to set the hour of a specified time. You can set hours by specifying the values of hours, minutes, seconds and milliseconds. Here the parameter hours is required. Whereas minutes, seconds, milliseconds are optional. Always used in conjunction with date object.

⇨ **Syntax**:

```
dateObject.setHours(hour,minutes,seconds,millisec);
```

❖ **Example**:

```
var d=new Date();
d.setHours(10);
```

- **setMinutes():** The setMinutes() method is used to set the minutes of a specified time. You can set the minutes by specifying the values of minutes, seconds and milliseconds. Here the parameter minute is required whereas seconds and milliseconds are optional. Always used in conjunction with date object.

⇨ **Syntax**:

```
dateObject.setMinutes(minute,seconds,milliseconds)
```

❖ **Example**:

```
var d=new Date()
d.setMinutes(20)
```

- **setSeconds():** The setSeconds() method is used to set the seconds of a specified time. You can set the seconds by specifying the values of seconds and milliseconds. Here the parameter second is required and millisecond is optional. Always used in conjunction with date Object.

⇨ **Syntax**:

```
dateObject.setSeconds(seconds,milliseconds)
```

❖ **Example**:

```
var d=new Date()
d.setSeconds(20)
```

□ Window Object

When you load your browser application, a window immediately appears. This window is known as browser window or window in short. You can use JavaScript to open a window, as the result of a button click or any other operation. Since each window is represented as a distinct window object, opening a new window actually creates another window object.

- **CREATE A WINDOW**
  windowVar=[window].open(“URL”,”WindowName”, [Window Attributes],height,width)

Window Attribute	**Description**
Toolbar	Back, forward and other buttons button
Location	Address bar displaying the current URL
Directories	What’s new, What’s Cool and other buttons in that row
Status	Status bar at the bottom of the window
Menubar	Displays menu bar to the window (File, Edit, View…)
Scrollbars	Displays scroll bar to the window
Resizable	Allows resizing of the window
Width	Windows width in pixel
Height	Windows height in pixel

❖ **Example**:

```
<html>
 <head>
     <script type="text/JavaScript">
        function open_win()
        {
            window.open("","my1","menubar,status,toolbar,height=
            100,width=500");
            window.open("","my2","fullscreen,menubar,status,toolbar");
        }
     </script>
 </head>
 <body>
       <form>
          <input type=button value="OpenWindow"onclick="open_win()">
     </form>
 </body>
</html>
```

- **Example 1:**

```
<head>
        <script type="text/JavaScript">
                function closeWin()
                {
                        xyz.close();
                }
        </script>
</head>
<body>
        <script type="text/JavaScript">
                xyz=window.open('','',width=200,height=100);
                xyz.document.write("This is my Window");
        </script>
        <form>
                <input type="button" value="Close 'myWindow'"
                onclick="closeWin()">
        </form>
</body>
```

- **Example 2:**

```
<head>
<script type="text/JavaScript">
        function resizeWindow1()
        {
                window.resizeBy(-100,-100);
   /*method is used to resize a window by the specified pixels*/
        }
        function resizeWindow2()
        {
                window.resizeTo(500,300)
    /*method is used to resize the window to the specified  width
    and height.*/
        }
</script>
</head>
<body>
        <form>
                <input type="button" onclick="resizeWindow1()"
                        value="Resize window By">
                <input type="button" onclick="resizeWindow2()"
                        value="Resize window To">
        </form>
</body>
```

History Object

- The browser maintains a list of most recent URL's, which can be viewed in IE as well as Netscape. The history list behaves like a LIFO queue (Last In First Out).
- The history list is represented in JavaScript by the window.history object. It allows u to deal with list but not with the data.

History Attribute	Description
length[property]	Returns the number of elements in the history list
back()	Loads the previous URL in the history list
forward()	Loads the next URL in the history list
go()	Loads a specific page in the history list

❖ **Example:**

```
<html>
   <head>
      <script language = "JavaScript">
           function ShowHistory()
          {
                 alert("U have accessed " + history.length +  " Web Pages ")
          }
     </script>
   </head>
   <body onload="ShowHistory()">
   </body>
</html>
```

❖ **Example:**

```
<html>
  <head>
        <script type="text/JavaScript">
                 function goBack()
                 {
                          history.forward()
                 }
        </script>
</head>
<body>
                 <input type="button" value="Back" onclick=goBack()>
</body>
</html>
```

❖ **Example:**

```
<html>
  <head>
    <script type="text/JavaScript">
      function goBack()
       {
            history.go(-3)
       }
    </script>
  </head>
   <body>
       <input type="button" value="Back" onclick=goBack()>
   </body>
</html>
```

 Location Object

The Location object is actually a JavaScript object, not an HTML DOM object. The Location object is automatically created by the JavaScript runtime engine and contains information about the current URL.

The Location object is part of the Window object and is accessed through the window.location property.IE: Internet Explorer, F: Firefox, O: Opera.

Attribute	Description
hash	Sets or returns the URL from the hash sign (#)
host	Sets or returns the hostname and port number of the current URL
hostname	Sets or returns the hostname of the current URL
href	Sets or returns the entire URL
pathname	Sets or returns the path of the current URL
port	Sets or returns the port number of the current URL
protocol	Sets or returns the protocol of the current URL
search	Sets or returns the URL from the question mark (?)
assign()	Loads a new document
reload()	Reloads the current document
replace()	Replaces the current document with a new one

- **Example**:

```
<html>
  <head>
    <script type="text/JavaScript">
      function newDoc()
      {
                window.location.assign("http://www.yahoo.com")
      }
    </script>
  </head>
  <body>
      <input type=button value="Load new document” onclick=newDoc()>
  </body>
</html>
```

- **Example**:

```
<html>
      <body>
            <script type="text/JavaScript">
                  function reloadPage()
                  {
                        window.location.reload()
                  }
            </script>
            <input type=button value="Reload page" onclick=reloadPage()>
      </body>
</html>
```

- **Example**:

```
<html>
      <body>
            <script type="text/JavaScript">
                  function replaceDoc()
                  {
                     window.location.replace("http://www.yahoo.com");
                  }
            </script>
      <input type=button value="Replace document" onclick=replaceDoc()>
  </body>
</html>
```

❖ **Example:**

```
<html>
        <body>
                <script type="text/JavaScript">
                        document.write(location.search);
                </script>
        </body>
</html>
```

❖ **Example:**

```
<html>
        <body>
                <script type="text/JavaScript">
                        document.write(location.href);
                </script>
        </body>
</html>
```

Question Bank: (1 Mark Questions)

1) JavaScript is a Client Side Scripting Language. (True/False) – **True**
2) There are no need to declare variables, classes or methods in JavaScript. (True/False) – **True**
3) We can add JavaScript in both <Head> and <Body> section. (True/False) – **True**
4) Which tag is used to describe JavaScript statement? – **<script>**
5) Which Literal has only two values like True & False? – **Boolean**
6) How many types of dialog boxes are available in JavaScript? – **3**
7) What is used to break line in message boxes? – **'\n'**
8) Which dialog box is used to display some information on web browser window? – **Alert**
9) Which dialog box is used to verify or accept something from user? – **Confirm**
10) Which dialog box is used to get some input from user? – **Prompt**

11) Which dialog box is return only Boolean datatype value? –**Confirm**

12) Which block is executed in Switch Case when expression is not match with any case? – **Default**

13) Which method is used to merge two or more string in JavaScript? –**Concat**

14) Which property is used in string object? –**Length**

15) What is the document in document.write()?–**Object**

16) What is the write in document.write()? -**Method**

17) What is the output of statement - Math.ceil(-9.7)? –**"–9"**

18) What is the output of statement - Math.floor(-0.9)? –**"–1"**

19) What is the output of statement - Math.random()? – **0.n (any random no)**

20) What is the output of statement - document.write(1 + 2 + "3")? –**33**

21) What is the output of statement - document.write("1" + 2 + 3)? –**123**

22) What is the output of statement - document.write("10" * 2 + 3)? –**23**

23)parseFloat(1 / 2 + "2")=? – **0.52**

24)parseInt(0 * 1 + "2")=? –**2**

25) If x="GUJARAT", What will the output in document.write(x.substr(3,2))? –**AR**

26) Which method is used to remove and return first element of array? –**shift()**

27) Which method is used to remove and return last element of array? –**pop()**

28) Which method is used to add element into end of the array? –**push()**

29) Which method is used to add element into beginning of the array? –**unshift()**

30) If x=”ATMIYA University”, then output of alert(x.substr(5,9)) is . – **A Univers**

Question Bank: (5 Marks Questions)

1) What is JavaScript? How to insert JavaScript into page? (Page No. 74,75)
Ans. What is JavaScript? – 2 marks
How to insert – 1 marks
Where to put JS into web-page. – 2 marks

2) Describe Data types (Literals) in JavaScript. (Page No. 77)
Ans. Different Literals. – 5 marks

3) Explain Dialog boxes in JavaScript. (Page No. 78)
Ans. What is dialog box? – 2 marks
3 types of dialog boxes with an example. – 3 marks

4) Explain four Decision making statement in JavaScript. (Page No. 81)
Ans. Explanation about decision statement with an example. – 5 marks

5) Explain Looping structure in JavaScript. (Page No. 84)
Ans. Explanation about looping structure with an example. – 5 marks

6) Explain Array in JavaScript. (Page No. 86)
Ans. Explanation about Array with an example. – 5 marks

7) Explain any 5 methods of String object with an example. (Page No. 92)
Ans. Description of methods with example – 5 marks

8) What is Object, Method and Property? Explain method of Math object with an example. (Page No. 92, 97)
Ans. Object, Method, Property – 2 marks
Method of Math Object. – 3 marks

9) Explain any 5 methods of Array object with an example. (Page No. 100)
Ans. Description of methods with example – 5 marks

10) Explain Window and History Object with an example. (Page No. 107,110)
Ans. Explanation of Window Object – 2 marks
Explanation of History Object – 3 marks

11) Explain Location Object with its attributes in JS. Describe any two examples of it. (Page No. 111)
Ans. About Location Object. – 1 marks
All attributes – 2 marks
Two examples – 2 marks

Chapter – 5

Events with DOM in JavaScript

EVENTS

□ **What Is an Events?**

It describes actions that occur as a result of user interaction with the web page or other browser related activities. When a button is clicked or a mouse has been moved or even when a key has been pressed an event is said to be occurred.

□ **Event Handling**

The response in conjunction to the occurrence of events by the web browser is called as event handling.

□ **Event Handler**

The code that performs this processing is called event handler.

Mouse Events

onClick:

The onClick event occurs when the user clicks on an element. The onClick event is the most frequently used event type, which occurs when a user clicks the left button of the mouse. You can put your validation, warning etc., against this event type.

- **Example:**

```
<script>
        function atmiya()
        {
                alert(“Wel-come to ATMIYA University”);
        }
</script>
<input type=button value=”Atmiya” onClick=atmiya()>
```

onMouseOver

The onMouseOver event occurs when the mouse pointer is moved onto an element of web page.

- **Example:**

```
<script>
        function atmiya()
        {
              alert(“Wel-come to ATMIYA University”);
        }
</script>
<span onMouseOver=atmiya()>Atmiya</span>
```

onMouseOut

The onmouseout event occurs when the mouse pointer is moved out of an element of web page.

- **Example:**

```
<script>
        function atmiya()
        {
              alert(“Wel-come to ATMIYA University”);
        }
</script>
<span onMouseOut=atmiya()>Atmiya</span>
```

onMouseDown

The onmousedown event occurs when a user presses a mouse button over an element. The order of events related to the onmousedown event (for the left/middle mouse button).

- **Example:**

```
<script>
        function atmiya()
        {
              alert(“Wel-come to ATMIYA University”);
        }
</script>
<span onMouseDown=atmiya()>Atmiya</span>
```

onMouseUp

The onmouseup event occurs when a user releases a mouse button over an element after pressing a mouse button. This event trigger after onMouseOver and onMouseDown events.

- **Example:**

```
<script>
        function atmiya()
        {
                alert("Wel-come to ATMIYA University");
        }
</script>
<span onMouseUp=atmiya()>Atmiya</span>
```

onMouseMove

The onmousemove event occurs when the pointer is moving while it is over an element of web page.

- **Example:**

```
<script>
        function atmiya()
        {
                alert("Wel-come to ATMIYA University");
        }
</script>
<div onMouseMove=atmiya()>Atmiya</div>
```

Keyboard Events

onKeyDown

The onKeyDown event occurs when the user is pressing any key from the keyboard.

- **Example:**

```
<script>
        function atmiya()
        {
                alert("Wel-come to ATMIYA University");
        }
</script>
<input onKeyDown=atmiya()>
```

onKeyPress

The onKeyDown event occurs when the user is pressing any key from the keyboard. The onkeypress event is not fired for all keys (e.g. ALT, CTRL, SHIFT, ESC) in all browsers. To detect only whether the user has pressed a key, use the onKeyDown event instead, because it works for all keys.

- **Example**:

```
<script>
        function atmiya()
        {
              alert(“Wel-come to ATMIYA University”);
        }
</script>
<input onKeyPress=atmiya()>
```

onKeyUp

The onKeyUp event occurs when the user releases a key from the keyboard. This event triggers after onkeydown or onkeypress event.

- **Example**:

```
<script>
        function atmiya()
        {
              alert(“Wel-come to ATMIYA University”);
        }
</script>
<input onKeyUp=atmiya()>
```

Other Events

onFocus

The onFocus event occurs when an element gets focus. The onFocus event is most often used with <input>, <select>, and <a>. The onFocus event is the opposite of the onBlur event.

- **Example**:

```
<script>
        function atmiya()
        {
              alert(“Wel-come to ATMIYA University”);
        }
</script>
<input onFocus=atmiya()>
```

onBlur

The onblur event occurs when an object loses focus. The onblur event is most often used with form validation code (e.g. when the user leaves a form field). The onblur event is the opposite of the onfocus event.

- **Example**:

```
<script>
        function atmiya()
        {
                alert("Wel-come to ATMIYA University");
        }
</script>
<input onBlur=atmiya()>
```

onChange

The onchange event occurs when the value of an element has been changed. For radiobuttons and checkboxes, the onchange event occurs when the checked state has been changed. This event triggers also on <select> elements and also while elements loses focus.

- **Example**:

```
<script>
        function atmiya()
        {
                alert("Wel-come to ATMIYA University");
        }
</script>
<input onChange=atmiya()>
```

onError

The onerror event is triggered if an error occurs while loading an external file (e.g. a document or an image).

- **Example**:

```
<script>
        function atmiya()
        {
                alert("Error..!! \n Image could not load.");
        }
</script>
<img src="e:\wrong\path\image.jpg" onError=atmiya()>
```

onLoad

The onload event occurs when an object has been loaded. onload is most often used within the <body> element to execute a script once a web page has completely loaded all content (including images, script files, CSS files, etc.). The onload event can be used to check the visitor's browser type and browser version, and load the proper version of the web page based on the information.

- **Example:**

```
<body onLoad=atmiya()>
        <script>
                function atmiya()
                {
                        alert("Web-page loading process is completed.");
                }
        </script>
</body>
```

- **onUnload**

The onunload event occurs once a page has unloaded (or the browser window has been closed). onunload occurs when the user navigates away from the page (by clicking on a link, submitting a form, closing the browser window, etc.). The onunload event is also triggered when a user reloads the page (and the onload event).

- **Example:**

```
<body onUnload=atmiya()>
        <img src="e:\wrong\path\image.jpg">
        <script>
                function atmiya()
                {
                        alert("Web-page could not load successfully.");
                }
        </script>
</body>
```

- **onResize**

The onResize event occurs when the browser window has been resized.

- **Example:**

```
<body onResize=atmiya()>
        <script>
                function atmiya()
                {
                        alert("You resize your browser window.");
                }
        </script>
</body>
```

onReset

The onReset event occurs when a form is reset using reset button in web page.

- **Example:**

```
<form onReset=atmiya()>
        Name : <input><br>
        <input type=reset>
</form>
<script>
        function atmiya()
        {
                alert("Web-page form will be reset.");
        }
</script>
```

onSelect

The onSelect event occurs after some text has been selected in an element. The onSelect event is mostly used on <input type="text"> or <textarea> elements.

- **Example:**

```
<input onSelect=atmiya()>
<script>
        function atmiya()
        {
                alert("You select some text from input box.");
        }
</script>
```

onSubmit

The onSubmit event occurs when a form is submitted. This event is triggered only from <form> element.

- **Example:**

```
 <form onSubmit=atmiya()>
         Name : <input><br>
        <input type=submit>
</form>
<script>
        function atmiya()
        {
                alert("Web-page form data submitted successfully.");
        }
</script>
```

Document Object Model

An HTML page is rendered in a browser. The browser assembles all the elements contained in the html page, downloaded from the web server in its memory. Once done the browser then renders these objects in the browser window. One the HTML page is rendered in the browser window, the browser can no longer recognize individual HTML elements. To create an interactive web page it is imperative that the browser continues to recognize individual HTML objects even after they are rendered in the browser window.

This allows the browser to access the properties of these objects using the built in methods of the objects. Once the properties of an object are accessible then the functionality of the object can be controlled at will.

JavaScript enabled browsers are capable of recognizing individual objects in an HTML page, after the page has been rendered in the browser, because the JavaScript enabled browser recognize and uses the Document Object Model.

Using the Document Object Model JavaScript enabled browsers identify the collection of web page objects (web page elements) that have to be dealt with while rendering an HTML based, web page in the browser window.

□ DOM methods and property (Working With Forms in JavaScript)

In the DOM, all HTML elements are defined as objects. The programming interface is the properties and methods of each object. A method is an action you can do (like add or deleting an HTML element). A property is a value that you can get or set (like changing the content of an HTML element).

1. getElementById()
2. getElementByName()
3. innerHTML property

4. getElementById()

If you want to quickly access the value of an HTML input, than this method can be used. It return a reference of the HTML element (control).

⇨ **Syntax:**

```
getElementById(Reference Id);
```

❖ **Example:**

```
<body>
        Welcome <span id=x>Guest</span>,
        <script type=text/JavaScript>
                function abc()
                {
                        document.getElementById(‘x’).innerHTML=”ATMIYA”;
                }
        </script>
        <input type=button value=” Ok ” onClick=abc()>
```

```
</body>
```

Here we print a message in web page like “Welcome Guest,” And call a function abc() using onClick event from input button. Function abc() get element by id ‘x’ and replace the content ‘ATMIYA’ using innerHTML property.

2. getElementByName()

It returns a collection of objects with the specified NAME.

⇨ **Syntax**:

```
document.getElementByName(Reference Name);
```

❖ **Example**:

```
<script language=JavaScript>
        function abc()
        {
                var x=document.getElementByName(“a”);
                alert(x.length);
        }
</script>
<form>
        Science : <input type=radio name=”a”>
        Commerce : <input type=radio name=”a”>
        Arts : <input type=radio name=”a”>
        <input type=button onClick=abc() value=” Ok ”>
</form>
```

3. innerHTML property

The easiest way to get or modify the content of an element is by using the innerHTML property.

When the innerHTML property is set, the given string completely replaces the existing content of the object. If the string contains HTML tags, the string is parsed and formatted as it is placed into the document. This property is accessible at run time as the document is being parsed.

Each HTML element has an innerHTML property that defines both the HTML code and the text that occurs between that element's opening and closing tag. By changing an element's innerHTML after some user interaction, you can make much more interactive pages.

However, using innerHTML requires some preparation if you want to be able to use it easily and reliably. First, you must give the element you wish to change an id. With that id in place you will be able to use the getElementById function, which works on all browsers.

After you have that set up you can now manipulate the text of an element. To start off, let's try changing the text inside a bold tag.

❖ **Example:**

```
<body>
        Welcome <span id=x>Guest</span>,
        <script type=text/JavaScript>
                function abc()
                {
                        document.getElementById('x').innerHTML="ATMIYA";
                }
        </script>
        <input type=button value=" Ok " onClick=abc()>
</body>
```

Timer [*Events/functions*]

The window object allows execution of code at specified time. The timer event is used to execute the statement[s] after particular time. The time will be in milliseconds. Once the timer event is started it will execute the statement[s] after specified time, which is given by user. There are two ways to use the timer events.

- setTimeout();
- setInterval();

These timer functions are called timing events.

□ setTimeout();

The window.setTimeout() method can be written without the window prefix.

The first parameter in setTimeout function is an execution part and the second parameter indicates the number of milliseconds before execution.

⇨ **Syntax:**

```
setTimeout(Execution part, miliseconds);
```

❖ **Example:**

```
<script type=text/javascript>
        setTimeout("alert('Welcome to ATMIYA University')",3000);
</script>
```

□ setInterval();

The setInterval() method repeats a given expression[s] at every specified time-intervals (in milliseconds). The setInterval() method will continue calling the function until clearInterval() is called, or the window is closed. The ID value returned by setInterval() is used as the parameter for the clearInterval() method.

⇨ **Syntax:**

```
setInterval(Execution part, miliseconds);
```

The window.setInterval() method can be written without the window prefix. The first parameter is the function to be executed. The second parameter indicates the length of the time-interval between each execution.

Note: *1000 miliseconds = 1 Second*

❖ **Example:**

```
<script type=text/javascript>
        setInterval("alert('Welcome to ATMIYA University')",3000);
</script>
```

□ How to Stop the Execution?

The clearTimeout() or clearInterval() method stops the execution of the function specified in timer event.

⇨ **Syntax:**

```
window.clearTimeout(setInterval VariableName)
window.clearInterval(setInterval VariableName)
```

The window.clearInterval() method can be written without the window prefix. The clearInterval() method uses the variable name returned from setInterval() function.

❖ **Example:**

```
<input type=button value=Stop onClick=abc()>
<script>
        var x=setInterval("alert('Welcome to ATMIYA University')",3000);
        function abc()
        {
                clearInterval(x);
        }
</script>
```

Error-Handling in JavaScript

□ Try...Catch Statement

The try...catch statement allows you to test a block of code for errors. The try block contains the code to be run, and the catch block contains the code to be executed if an error occurs. When users see errors, they often leave the Web page. This will teach you how to catch and handle JavaScript error messages.

⇨ **Syntax:**

```
try
{
    Some JS statements
 }
catch(variable name)
{
    Handle errors message
}
```

When executing JavaScript code, different errors can occur. Errors can be coding errors made by the programmer, errors due to wrong input, and other unforeseeable things.

Note that try...catch is written in lowercase letters. Using uppercase letters will generate a JavaScript error!

❖ **Example:**

```
<script>
        try
        {
                Document.write(“Welcome to ATMIYA Univeristy”);
        }
        catch(z)
        {
                alert(z.description);
        }
</script>
```

The example above determines statement into try block generate some error, because of object ‘Document’ is written in capital letter. Try block sent error to the catch() statement. In catch statement, parameter ‘z’ store this error and finally display it in alert box as a parameter description.

□ **The throw Statement**

The throw statement can be used together with the try...catch statement, to create an exception for the error. The throw statement allows you to create an exception. If you use this statement together with the try...catch statement, you can control program flow and generate accurate error messages.

⇨ **Syntax:**

```
throw (exception)
```

The exception can be a string, integer, Boolean or an object. Note that throw is written in lowercase letters. Using uppercase letters will generate a JavaScript error.

- **Example:**

```
<script>
        var x=prompt("Enter a number between 0 and 9.","");
        try
        {
                if(x>9)
                {
                        throw "Err1";
                }
                else if(x<0)
                {
                        throw "Err2";
                }
                else if(isNaN(x))
                {
                        throw "Err3";
                }
                else
                {
                        alert("Entered Value is "+x);
                }
         }
        catch(er)
        {
                switch(z)
                {
                        Case 'Err1':
                                alert("Too High Value");
                                break;
                        Case 'Err2':
                                alert("Too Low Value");
                                break;
                        Case 'Err3':
                                alert("Value is not a number");
                }
        }
</script>
```

The example above determines the value of a variable called x. If the value of x is higher than 9, lower than 0, or not a number, we are going to throw an error. The error is then caught by the catch argument and the proper error message is displayed.

Question Bank: (1 Mark Questions)

1) Which property is used in DOM? –**innerHTML**
2) Which method is used to access the set (group) of objects in DOM? – **getElementsByName()**
3) Which event is occur when input element lose the focus? –**onBlur()**
4) Which event is triggered while release the mouse button? –**onMouseUp()**
5) Which event is triggered while release the key from keyboard? –**onKeyUp()**
6) Which event is written only into form element? –**onSubmit()**
7) Stands for : DOM –**Document Object Model**
8) Which event is used to execute statement only once after particular time? – **setTimeout()**
9) Which function is used to stop output of interval function? –**clearInterval()**
10) Which function is used to repeats an expression at specified time. –**setInterval()**
11) Which two statements are used for error handling in JavaScript? –**try...Catch, throw**
12) A throw() statement is used with   statement. –**try…catch()**
13) Which statement is compulsory used with try…catch statements? –**throw()**
14) onSubmit event is written in Submit button controller only. (True/False) –**False**
15) onReset event is written in Form element. (True/False) –**True**
16) onBlur event is triggered while image got blur pixel on web page. (True/False) – **False**
17) innerHTML is a method of Document Object Model. (True/False) –**False**
18) innerHTML is not a property of Timer event. (True/False) –**True**
19) innerHTML is written with setTimeout function. (True/False) –**False**
20) A throw statement is used with try…catch() statement. (True/False) –**True**
21) The exception of throw() can be a string, integer, Boolean or an object. (True/False) – **True**

Question Bank: (5 Marks Questions)

1) What is an Event? Describe mouse events with an example. (Page No. 116)
Ans. What is Event? – 1 marks
Mouse events with an example – 4 marks

2) What is an Event? Describe Keyboard events with an example. (Page No. 116, 118)
Ans. What is Event? – 1 marks
Keyboard events with an example – 4 marks

3) What is an Event? Describe onFocus, onError, onReset, onSelect with an example. (Page No. 119)
Ans. What is Event? – 1 marks
Events description with an example – 4 marks

4) What is DOM? Describe its methods with an example. (Page No. 123)
Ans. What is DOM? – 2 marks
Description of methods with example– 3 marks

5) What is DOM? Describe its property with an example. (Page No. 123, 124)
Ans. What is DOM? – 2 marks
Description of property with example– 3 marks

6) What is Timer event? Describe its functions with an example. (Page No. 125)
Ans. What is Timer? – 1 marks
Description of its 2 functions with example– 4 marks

7) Explain setInterval? How to stop the execution of Interval? (Page No. 125, 126)
Ans. Explanation of setInterval. – 1 marks
How to stop the execution? With example– 4 marks

8) Explain Error Handling statements with an example. (Page No. 126)
Ans. Explanation of try…catch() statement with example – 2 marks
Explanation of throw statement with example – 3 marks

~PAPER STYLE~

Enrollment No.

ATMIYA University, Rajkot

Semester End Examination - October 2018

Program : **BCA/B.Sc. Information Technology**	Semester : **II**	Batch **:**
Course : **18BCACC203/18BITCC203 - Web Scripting Languages**		
Duration of Examination : **3 Hrs.**	Max. Marks : **70**	Date : / /

Part A ***(10X1= 10 marks)***

Answer **ALL** questions

1. xxxxx xxxx xxxxx xxxx xxxxx xxxx xxxxx xxxx

↓

10. xxxxx xxxx xxxxx xxxx xxxxx xxxx xxxxx xxxx

Part B ***(5X5= 25 marks)***

Answer **ALL** questions

11a. xxxxxxxxx xxxxx xxxx xxxxx xxxx xxxxx xxxx

OR

11b. xxxxxxxxx xxxxx xxxx xxxxx xxxx xxxxx xxxx

↓

15a. xxxxxxxxx xxxxx xxxx xxxxx xxxx xxxxx xxxx

OR

15b. xxxxxxxxx xxxxx xxxx xxxxx xxxx xxxxx xxxx

Part C ***(5X7= 35 marks)***

Answer **ALL** questions

16a. xxxxxxxxx xxxxx xxxx xxxxx xxxx xxxxx xxxx

OR

16b. xxxxxxxxx xxxxx xxxx xxxxx xxxx xxxxx xxxx

↓

20a. xxxxxxxxx xxxxx xxxx xxxxx xxxx xxxxx xxxx

OR

20b. xxxxxxxxx xxxxx xxxx xxxxx xxxx xxxxx xxxx

Enrollment No.

ATMIYA UNIVERSITY, RAJKOT

MODEL QUESTION PAPER

Program **: BCA/B.Sc.I.T.** Semester : **II** Batch **:**

Course : **18BCACC203/18BITCC203 - Web Scripting Languages**

Duration of Examination : **3 Hrs.** Max. Marks : **70** Date : / /

Part A ***(10X1= 10 marks)***

Answer **ALL** questions

1. Which attribute is used to change the default font color using Body tag?
2. Full form of HTML.
3. Which attribute is used to specify the coordinates of image in Image Map?
4. Which element is used to give a title in HTML Table?
5. Which extension is used in external CSS file?
6. Which symbol is used to create Id selector?
7. Javascript is Case sensitive language. (True/False)
8. Javascript is side Scripting Language.
9. Which symbol is used to create Class Selector?
10. If parseInt(8.4) then output = .

Part B ***(5X5= 25 marks)***

Answer **ALL** questions

11a. Describe the Body element with its attributes.

OR

11b. Explain <P> tag, heading tag and <font> element with its attributes and an example.

12a. Explain Marquee element with its attributes.

OR

12b. What is Image map? Explain related elements and attributes with an example.

13a. Write CSS Font properties in detail.

OR

13b. Describe about types of Style sheet.

14a. Explain Decision statements with example.

OR

14b. Explain Looping Structure with example.

15a. Explain JavaScript Dialog boxes with an example.

OR

15b. Explain all JS Literals with an example.

Part C ***(5X7= 35 marks)***

Answer **ALL** questions

16a. What is Hyperlink? Describe Anchor tag with its various attributes.

OR

16b. Describe all elements and its attributes related to table tag.

17a. Describe all elements and its attributes related to Form.

OR

17b. Describe the Image element with various attributes in detail.

18a. Explain CSS Font Properties with example.

OR

18b. Explain CSS Margin & Padding Properties with example.

19a. Write CSS Text properties in detail.

OR

19b. Explain Border Properties with example.

20a. Explain method for String object in JS.

OR

20b. Explain method for Array object in JS.