

**Unit-I**

Creating an HTML Document – creating list – creating links between document – linking to resources on the internet – working with hypertext attributes – working with fonts and text styles – tables – creating frameset – working with forms – working with cascading style sheets.

**Unit-II**

**JavaScript:** Introduction to javascript – Programming fundamentals – Functions and objects – Navigator object model

**Unit-III**

**JavaScript:** Form and form elements – Scripting frames and multiple windows – Event object – Functions and custom objects.

**Unit-IV**

**ASP:** Client side scripting vs. Server side scripting- Variables & Constants- Procedures – Forms – Cookies – Application - #include – Global.asa - Functions-ASP object model: Response-Request- Application- Session – Server – Error – Array

**Unit-V**

**ASP:** Collections & Control Structure-File system object: File System – Text Stream-Drive – File – Folder – Directory – ADO - sql & Databases for data driven applications-ASP Components: Ad Rotator – Browser Cap. – Content Linking – Content Rotator.

**Suggested Readings**

1. Patrick Carey, (2005). *New Perspectives on HTML and XHTML*, (1<sup>st</sup> ed.), Thomson Course Technology Publishing. **(Unit- I).**
2. Rohit Khurana's, (2002). *Javascript Professional edition*, (2<sup>nd</sup> ed.), A.P.H. Publishing company, NewDelhi.**(Unit -II)**
3. Danny Goodman, (2000). *Javascript Bible*, (3<sup>rd</sup> ed.), IDG Books India Pvt Ltd. **(Unit- III).**
4. Russell Jones, A. (2000). *Mastering ActiveServerPages 3*, (1<sup>st</sup> ed.), BPB Publishing, New Delhi.**( Unit- IV & Unit -V).**
5. David Flanagan, (2006). *JavaScript: The Definitive Guide*, O'Reilly,
6. Nicholas C. Zakas, Inc Ebrary and Ebrary,(2005). *Professional JavaScript for Web Developers*, New Delhi , John Wiley & Sons Inc.

**Web Sites:**

1. [www.w3schools.com/](http://www.w3schools.com/)
2. [www.javascriptkit.com](http://www.javascriptkit.com)
3. [www.aspfree.com](http://www.aspfree.com)
4. [www.aspnetutorials.com](http://www.aspnetutorials.com)



**KARPAGAM ACADEMY OF HIGHER EDUCATION**  
(Deemed to be University)  
(Established Under Section 3 of UGC Act, 1956)  
Coimbatore-21

**LECTURE PLAN**

**COMPUTER APPLICATIONS**

**WEB PROGRAMMING - 18CAU403**

S.No.	Lecture Duration Period	Topics to be Covered	Support Materials / Page No.
<b>Unit – I</b>			
1.	1	Creating an HTML Document Creating lists Creating Links between documents	T1: 9-21, W1 T1: 24-28 , W3 T1: 64-76
2.	1	Linking to Resources on the Internet Working with Hypertext attributes	T1: 76-87 R3: 19-33, W3
3.	1	Working with Fonts & Text Styles Creating a table Creating Frameset	T1: 116-130 T1: 174-187 T1: 248-258
4.	1	Working with Forms Working with Form Buttons	T1: 292-303, W1 T1: 323-329
5.	1	Working with Cascading Style Sheets Using an External Style Sheets	T1: 348-363 R3: 83-105
6.	1	Recapitulation and Discussion of important questions	-
<b>Total No. of Hours Planned for Unit-I</b>			<b>6hrs</b>

Unit – II			
1	1	Introduction of JavaScript Facts about JavaScript, What can you do with JavaScript	T2: 1-14, W3
2	1	Programming Fundamentals: Introduction, Datatypes	T2: 18-21, W3
3	1	Operators Statements , Comments	T2: 21-28,W4
4	1	Functions and Objects, Predefined Core Objects	T2: 50-60,W4
5	1	Array The String Object, Number Object,	T2: 53-56 R1: 104-110
6	1	Math Object, Date Object Navigator Object Model	T2: 60-63 T2: 83-87, R1: 111-118, W3
7	1	Recapitulation and Discussion of important questions	-
Total No. of Hours Planned for Unit-II			7hrs
Unit – III			
1.	1	Form and Form Elements, The Form Object, Text Object, Button	T3:99-101, R1: 439-447
2.	1	Check box, Radio, Select objects Submitting Forms	T3:102-105 T3: 109-110, W3
3.	1	Scripting Frames and Multiple Windows Frames: Parents and Children	T3: 123-127 W5
4.	1	Frame Scripting Tips Event Object: Event Handlers and Properties	T3: 126-129 T3: 685-686, W3
5.	1	New Navigator New Events	T3: 687-692, W5
6.	1	Functions Custom Objects	T3: 699-724, W4
7.	1	Recapitulation and Discussion of important questions	
Total No. of Hours Planned for Unit-III			7hrs

Unit - IV			
1.	1	Client Side Scripting Vs Server Side Scripting	T4: 80-85 R2: 9-27
2.	1	Variables and Constants Cookies Application	T4: 164-169, W1 T4: 198-200, W1 T4: 285-290, W6
3.	1	Procedures and Forms	T4: 170-172, 189-192
4.	1	#include Global.asa, Functions	T4: 258-265, W1
5.	1	ASP Object Model, Response, Request Application, Session	T4: 158-196, W7
6.	1	Server, Error	T4: 396-417
7.	1	Array	T4: 283-291, W7
8.	1	Recapitulation and Discussion of important questions	
Total No. of Hours Planned for Unit-IV			8hrs
Unit – V			
1.	1	ASP: Collection and Control Structure	T4: 184-220, 291- 298
2.	1	File System Object: File System, Text Stream	T4: 369-378
3.	1	Drive, File	T4: 378-382, W7
4.	1	Folder, Directory ADO	T4: 382-392 T4: 505-515
5.	1	SQL & Databases for data driven applications	T4: 516-526
6.	1	ASP Components	T4: 438- 443, W7
7.	1	Browser Cap	T4: 771-773, W7
8.	1	Content Linking, Content Rotator	T4: 773-774, W7
9.	1	Recapitulation and Discussion of important questions	-
10.	1	Recapitulation and Discussion of previous ESE question papers	-
11.	1	Recapitulation and Discussion of previous ESE question papers	-
12.	1	Recapitulation and Discussion of previous ESE question papers	-
Total No. of Hours Planned for Unit-V			12hrs
Total No. of Hours Planned for this Syllabus			44 hrs

**TEXT BOOKS:**

1. Patrick Carey 2011, New Perspectives on HTML and XHTML, 6th Edition, Cengage Learning. (UNIT I).
2. Rohit Khurana's, 2002, JavaScript Professional Edition, 2<sup>nd</sup> Edition, A.P.H Publishing Company, NewDelhi. (Unit II)
3. Danny Goodman, 2010, "JavaScript Bible", 7<sup>th</sup> Edition, Wiley India Publications; 7<sup>th</sup> Edition (November 9, 2010) (Unit III).
4. 4. A.Russell Jones. 2000. Mastering Active Server Pages 3, 1<sup>st</sup> Edition, BPB Publishing, New Delhi.( Unit IV & Unit V).

**REFERENCE BOOKS:**

1. Thau. 2007. The Book of JavaScript: A Practical Guide to Interactive WebPages,
2. Wendy Willard. 2007. HTML: A Beginner's Guide, Tata McGraw-Hill Professional, New Delhi.
3. Chuck Musciano and Bill Kennedy. 2006. HTML & XHTML: The Definitive \*Guide, O'Reilly.
4. David Flanagan. 2006. JavaScript: The Definitive Guide, O'Reilly,
5. Nicholas C. Zakas, Inc Ebrary and Ebrary. 2005. Professional JavaScript for Web Developers, John Wiley & Sons Inc, New Delhi.
6. Jude D'Souza and Monica D'Souza. 2002. Discover ASP, 1st Edition, TATA McGraw Hill Professional, New Delhi.

**WEB SITES:**

1. [www.w3schools.com](http://www.w3schools.com)
2. [www.tutorialpoint.com](http://www.tutorialpoint.com)
3. [www.javascriptkit.com](http://www.javascriptkit.com)
4. [www.tigzag.com](http://www.tigzag.com)
5. [www.pageresource.com](http://www.pageresource.com)
6. [www.aspfree.com](http://www.aspfree.com)
7. [www.aspnetutorials.com](http://www.aspnetutorials.com)

**UNIT-I**

**SYLLABUS**

**HTML** : Creating an HTML Document – creating list – creating links between document – linking to resources on the internet – working with hypertext attributes – working with fonts and text styles – tables – creating frameset – working with forms – working with cascading style sheets.

**Introduction:**

The HTML that Tim invented was strongly based on SGML (Standard Generalized Mark-up Language), an internationally agreed upon method for marking up text into structural units such as paragraphs, headings, list items and so on. SGML could be implemented on any machine. The idea was that the language was independent of the formatter (the browser or other viewing software) which actually displayed the text on the screen. The use of pairs of tags such as <TITLE> and </TITLE> is taken directly from SGML, which does exactly the same. The SGML elements used in Tim's HTML included P (paragraph); H1 through H6 (heading level 1 through heading level 6); OL (ordered lists); UL (unordered lists); LI (list items) and various others. What SGML does not include, of course, are hypertext links: the idea of using the anchor element with the HREF attribute was purely Tim's invention, as was the now-famous 'www.name.name' format for addressing machines on the Web.

Basing HTML on SGML was a brilliant idea: other people would have invented their own language from scratch but this might have been much less reliable, as well as less acceptable to the rest of the Internet community. Certainly the simplicity of HTML, and the use of the anchor element A for creating hypertext links, was what made Tim's invention so useful.

**Creating an HTML Document:**

**HTML Rules and Guidelines:**

There are **five important rules** for coding with HTML tags.

- Tags are always surrounded by angle brackets (less-than/greater-than characters), as in<HEAD>.

- Tags come in pairs and surround the material they affect. They work like a light switch: the first tag turns the action on, and the second turns it off. (There are some exceptions. For instance, the <BR> tag creates a blank line and doesn't have an "off switch." Once made a line break, we can't unmake it.)
- The second tag--the "off switch"--always starts with a forward slash. For example, we turn on bold with <B>, shout we piece, and then go back to regular text with </B>.
- First tag on, last tag off. Tags are embedded, so when we start a tag within another tag, we have to close that inner tag before closing the outer tag. For instance, the page will not display properly with the tags in this order:

<HEAD><TITLE>We text</HEAD></TITLE>.

The correct order is:

<HEAD><TITLE>We text</TITLE></HEAD>.

- Many tags have optional attributes that use values to modify the tag's behavior. The <P>(paragraph) tag's ALIGN attribute, for instance, lets we change the default (left) paragraph alignment. For example, <P ALIGN=CENTER> centers the next paragraph on the page.

### **Basic Structure of Html:**

Document Type <HTML> </HTML>

Title <TITLE></TITLE>

Header <HEAD> </HEAD>

Body <BODY> </BODY>

- HTML document begins and ends with HTML tag i.e. <HTML> </HTML>  
Here <HTML> indicates the browser that it is a HTML document and </HTML> tells the browser that HTML document is completed.

- Header Tag i.e. <HEAD></HEAD>

Header Tag does not contain any text, it only contains the Title Tag in it.

- Title tag i.e. <TITLE></TITLE>

Anything written between this tag is not displayed on the screen but it is used to identify the Webpage.

- Body tag i.e. <BODY></BODY>

This is the main part of HTML document. The content which is to be displayed on screen as webpage should be written here. Body Tag contains the text as well as various tags but only the text will be displayed on Webpage.

## Creating List:

### Un Ordered Lists: <ul> , <li>

Tag to create a list of items.

Code:

```
<ul>
<li>List1</li>
<li>List2</li>
</ul>
```

### Result:

- List1
- List2

### Ordered Lists: <ol> , <li>

Tag to create a list of numbered items. The numbering will be done automatically.

Code:

```
<ol>
<li>List1</li>
<li>List2</li>
</ol>
```

### Result:

1. List1
2. List2



**Nested Lists:** <ul> , <li>

Code:

```
<ul>
<li>List1</li>
  <ul>
    <li>Sub List1</li>
    <li>Sub List2</li>
  </ul>
<li>List2</li>
</ul>
```

**Result:**

- List1
  - Sub List1
  - Sub List2
- List2

**Definition List:**

A description list, with terms and descriptions:

- 
- <dl>
  - <dt>Coffee</dt>
  - <dd>Black hot drink</dd>
  - <dt>Milk</dt>
  - <dd>White cold drink</dd></dl>

**Result:**

Coffee

Black hot drink

Milk

White cold drink

## Creating Links between Document:

**Link:** <a> </a>

Consider we have two pages test.html and test1.html.

We want to give a link in test.html so that others can click on it and go to test1.html.  
The tag used is "a" with attribute named "href"

Example Code:

```
<a href="test1.html"> Click to test1.html </a>
```

### **Result:**

Click to test1.html\_

The tag should be used as above.

The text in between the tag will be displayed on the page.

On clicking the text the user will be taken to the page defined in the attribute href

### **Link using Images:**

Even images can be used for creating links.

Its simple.

Give the image in between the "a" tags.

Example Code:

```
<a href="test1.html">  </a>
```

The HTML <a> tag defines a hyperlink.

A hyperlink (or link) is a word, group of words, or image that we can click on to jump to another document.

When we move the cursor over a link in a Web page, the arrow will turn into a little hand.

The most important attribute of the <a> element is the href attribute, which indicates the link's destination.

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

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

### HTML Link Syntax

The HTML code for a link is simple. It looks like this:

```
<a href="url">Link text</a>
```

The href attribute specifies the destination of a link.

#### Example

```
<a href="http://www.w3schools.com/">Visit W3Schools</a>
```

Clicking on this hyperlink will send the user to W3Schools' homepage.

The "Link text" doesn't have to be text. It can be an image or any other HTML element.

#### HTML Links - The target Attribute

The target attribute specifies where to open the linked document.

The example below will open the linked document in a new browser window or a new tab:

#### Example

```
<a href="http://www.w3schools.com/" target="_blank">Visit W3Schools!</a>
```

#### HTML Links - The id Attribute

The id attribute can be used to create a bookmark inside an HTML document.

Bookmarks are not displayed in any special way. They are invisible to the reader.

#### Example

An anchor with an id inside an HTML document:

```
<a id="tips">Useful Tips Section</a>
```

Create a link to the "Useful Tips Section" inside the same document:

```
<a href="#tips">Visit the Useful Tips Section</a>
```

Or, create a link to the "Useful Tips Section" from another page:

```
<a href="http://www.w3schools.com/html_links.htm#tips">  
Visit the Useful Tips Section</a>
```

### HTML Images - The <img> Tag and the Src Attribute

In HTML, images are defined with the <img> tag.

The <img> tag is empty, which means that it contains attributes only, and has no closing tag.

To display an image on a page, we need to use the src attribute. Src stands for "source". The value of the src attribute is the URL of the image we want to display.

#### Syntax for defining an image:

```

```

The URL points to the location where the image is stored. An image named "boat.gif", located in the "images" directory on "www.w3schools.com" has the URL: <http://www.w3schools.com/images/boat.gif>.

The browser displays the image where the <img> tag occurs in the document. If we put an image tag between two paragraphs, the browser shows the first paragraph, then the image, and then the second paragraph.

### HTML Images - The Alt Attribute

The required alt attribute specifies an alternate text for an image, if the image cannot be displayed.

The value of the alt attribute is an author-defined text:

```

```

The alt attribute provides alternative information for an image if a user for some reason cannot view it (because of slow connection, an error in the src attribute, or if the user uses a screen reader).

### HTML Images - Set Height and Width of an Image

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

The attribute values are specified in pixels by default:

```

```

Specify both the height and width attributes for an image. If these attributes are set, the space required for the image is reserved when the page is loaded. Without these attributes, the browser does not know the size of the image. The effect will be that the page layout will change during loading (while the images load).

Example: image float to the left or right of a paragraph.

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<p>
```

```
 A  
paragraph with an image. The image will float to the left of this text.
```

```
</p>
```

```
<p>
```

```
 A  
paragraph with an image. The image will float to the right of this text.
```

```
</p>
```

```
<p><b>Note:</b> Here we have used the CSS "float" property to align the image; as the align  
attribute is deprecated in HTML 4, and is not supported in HTML5.</p>
```

```
</body>
```

```
</html>
```

### Result:

A paragraph with an image. The image will float to the left of this text.



A paragraph with an image. The image will float to the right of this text.

## Working with Fonts and Text Styles:

### Font Type and Font Size :

```
<html>
  <body bgcolor = "gray">
    <font face = "Arial","Times New Roman", size = 6
    color = Coral>COMPUTER SOFTWARE COLLEGE
  </font>
  <br>
  <br>
  <basefont face = "MsSansSerif" size = 7 color =
                                Cornflowerblue>MULTIMEDIA</basefont>
  <br>
  <font face = "Arial" size = 7 onmouseover="this.color='yellow'";
                                onmouseout="this.color='orange'";>COMPUTERS</font>
</body>
</html>
```

### Bold, Italic, Underline and Strikeout:

```
<html>
  <body bgcolor = "wheat" text="brown">
    <b>Computer</b>
  <br>
  <i>Software</i>
  <br>
  <u>College</u>
  <br>
  <strong>School</strong>
  <br>
  <strike>Chennai</strike>
```

```
<br>
<s>Calcutta</s>
<br>
<del>Mumbai</del>
<br>
</body>
</html>
```

**Result:**

**Computer**

*Software*

College

**School**

~~Chennai~~

~~Calcutta~~

| ~~Mumbai~~

**Big, Small, Bold, Italic, Superscript, Subscript and Striking out:**

HTML uses tags like `<b>` and `<i>` for formatting output, like **bold** or italic text. These HTML tags are called formatting tags (look at the bottom of this page for a complete reference).

`<strong>` renders as **`<b>`**, and `<em>` renders as *`<i>`*.

There is a difference in the meaning of these tags:

`<b>` or `<i>` defines bold or italic text only.

`<strong>` or `<em>` means that we want the text to be rendered in a way that the user understands as "important". Today, all major browsers render strong as bold and em as italics. However, if a browser one day wants to make a text highlighted with the strong feature, it might be cursive for example and not bold!

```
<!DOCTYPE html>
<html>
<body>
<p><b>This text is bold</b></p>
<p><strong>This text is strong</strong></p>
<p><em>This text is emphasized</em></p>
<p><i>This text is italic</i></p>
<p><small>This text is small</small></p>
<p>This is<sub> subscript</sub> and <sup>superscript</sup></p>
</body>
</html>
```

**Result:**

**This text is bold**

**This text is strong**

*This text is emphasized*

*This text is italic*

This text is small

This is <sub>subscript</sub> and <sup>superscript</sup>

**Preformatted Text**

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

```
<pre>
This is preformatted text.
It preserves    both spaces
and line breaks.
</pre>
```

<p>The pre tag is good for displaying computer code:</p>



```
<pre>
for i = 1 to 10
  print i
next i
</pre>

</body>
</html>
```

**Attribute: bgcolor** This sets the background color for marquee path

Example Code:

```
<marquee bgcolor=orange> Moving Text </marquee>
```

**Attribute: height,width** The attribute width sets the width of marquee area  
The attribute height sets the height of marquee area

**Example Code:**

```
<marquee bgcolor=orange width=100 height=20> Moving Text </marquee>
```

**Attribute: direction**

This sets the background color for marquee path.

It takes values **LEFT or RIGHT or UP or DOWN**

**Example Code:**

```
<marquee bgcolor=orange width=100 height=20 direction=right> Text will Move </marquee>
```

**Horizontal Rules:**

**Horizontal Line:** <hr>

This is special tag used to draw new horizontal lines.

It doesn't require closing tags.

This tag has the attribute "width" to specify the width of the line  
<hr width=60%> will give the below line

## **Tables :**

### **Creating Table:**

Tables are defined with the <table> tag.


A table is divided into rows (with the <tr> tag), and each row is divided into data cells (with the <td> tag). td stands for "table data," and holds the content of a data cell. A <td> tag can contain text, links, images, lists, forms, other tables, etc.

### **HTML Table Tags**

#### **Table Example**

```
<table border="1">
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>row 2, cell 2</td>
</tr>
</table>
```

The HTML code above looks in a browser:



row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2

### **Dividing Table into Rows**

---

### **HTML Tables and the Border Attribute**

If we do not specify a border attribute, the table will be displayed without borders. Sometimes this can be useful, but most of the time, we want the borders to show.

To display a table with borders, specify the border attribute:

```
<table border="1">
<tr>
<td>Row 1, cell 1</td>
<td>Row 1, cell 2</td>
</tr>
</table>
```

### **HTML Table Headers**

Header information in a table are defined with the <th> tag.

All major browsers display the text in the <th> element as bold and centered.

```
<table border="1">
<tr>
<th>Header 1</th>
<th>Header 2</th>
</tr>
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>row 2, cell 2</td>
```

```
</tr>
</table>
```

How the HTML code above looks in we browser:

Header 1	Header 2
row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2

**Table Without Borders:**

```
<html>
<body>

<h4>This table has no borders:</h4>
<table>
<tr>
  <td>100</td>
  <td>200</td>
  <td>300</td>
</tr>
<tr>
  <td>400</td>
  <td>500</td>
  <td>600</td>
</tr>
</table>

<h4>And this table has no borders:</h4>
<table border="0">
<tr>
  <td>100</td>
  <td>200</td>
```

```
<td>300</td>
</tr>
<tr>
  <td>400</td>
  <td>500</td>
  <td>600</td>
</tr>
</table>
```

```
</body>
</html>
```

### Result:

**This table has no borders:**

100 200 300

400 500 600

**And this table has no borders:**

100 200 300

400 500 600

### Dividing Table into Columns:

The <colgroup> tag is supported in all major browsers.

#### Definition and Usage

- The <colgroup> tag specifies a group of one or more columns in a table for formatting.
- The <colgroup> tag is useful for applying styles to entire columns, instead of repeating the styles for each cell, for each row.

- The <colgroup> tag must be a child of a <table> element, after any <caption> elements and before any <thead>, <tbody>, <tfoot>, and <tr> elements.
- To define different properties to a column within a <colgroup>, use the <col> tag within the <colgroup> tag

#### **//Dividing Columns and Rows//**

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<h4>Cell that spans two columns:</h4>
```

```
<table border="1">
```

```
<tr>
```

```
<th>Name</th>
```

```
<th colspan="2">Telephone</th>
```

```
</tr>
```

```
<tr>
```

```
<td>Bill Gates</td>
```

```
<td>555 77 854</td>
```

```
<td>555 77 855</td>
```

```
</tr>
```

```
</table>
```

```
<h4>Cell that spans two rows:</h4>
```

```
<table border="1">
```

```
<tr>
```

```
<th>First Name:</th>
```

```
<td>Bill Gates</td>
```

```
</tr>
```

```
<tr>
```

```
<th rowspan="2">Telephone:</th>
```

```
<td>555 77 854</td>
```

```
</tr>
```

```
<tr>
```

```
<td>555 77 855</td>
```

CLASS: II BCA  
COURSE CODE: 18CAU403

COURSE NAME: WEB PROGRAMMING  
UNIT: I(HTML) BATCH: 2018-2021

```
</tr>
</table>

</body>
</html>
```

### Result:

#### Cell that spans two columns:

Name	Telephone	
Bill Gates	555 77 854	555 77 855

#### Cell that spans two rows:

First Name:	Bill Gates
Telephone:	555 77 854
	555 77 855

### Creating Frameset:

#### Overview Of Frames

The <frameset> tag defines a frameset.

The <frameset> element holds one or more <frame> elements. Each <frame> element can hold a separate document.

The <frameset> element specifies HOW MANY columns or rows there will be in the frameset, and HOW MUCH percentage/pixels of space will occupy each of them.

**Example:**

**A simple three-framed page:**

```
<frameset cols="25%,*,25%">  
  <frame src="frame_a.htm">  
  <frame src="frame_b.htm">  
  <frame src="frame_c.htm">  
</frameset>
```

**Horizontal frameset:**

```
<!DOCTYPE html>  
<html>  
  
  <frameset rows="25%,*,25%">  
    <frame src="frame_a.htm">  
    <frame src="frame_b.htm">  
    <frame src="frame_c.htm">  
  </frameset>  
</html>
```

**Mixed Frameset:**

```
<!DOCTYPE html>  
<html>  
  
  <frameset rows="50%,50%">  
    <frame src="frame_a.htm">  
    <frameset cols="25%,75%">  
      <frame src="frame_b.htm">  
      <frame src="frame_c.htm">  
    </frameset>  
  </frameset>  
</html>
```



**Frameset with no resize:**

Use the "noresize" attribute. The frames are not resizable. Move the mouse over the borders between the frames and notice that we can not move the borders.

```
<!DOCTYPE html>
<html>

<frameset cols="50%,*,25%">
  <frame src="frame_a.htm" noresize="noresize">
  <frame src="frame_b.htm">
  <frame src="frame_c.htm">
</frameset>

</html>
```

**Definition and Usage:**

The target attribute specifies where to open the linked document.

**Syntax:**

```
<a target="_blank|_self|_parent|_top|framename">
```

**Attribute Values**

Value	Description
_blank	Opens the linked document in a new window or tab
_self	Opens the linked document in the same frame as it was clicked (this is default)
_parent	Opens the linked document in the parent frame
_top	Opens the linked document in the full body of the window

framename	Opens the linked document in a named frame
-----------	--

## Working with HTML Forms:

### HTML Forms

**HTML forms are used to pass data to a server.**

An HTML form can contain input elements like text fields, checkboxes, radio-buttons, submit buttons and more. A form can also contain select lists, textarea, fieldset, legend, and label elements.

The <form> tag is used to create an HTML form:

```
<form>  
.  
input elements  
.  
</form>
```

### The Input Element

The most important form element is the <input> element.

The <input> element is used to select user information.

An <input> element can vary in many ways, depending on the type attribute. An <input> element can be of type text field, checkbox, password, radio button, submit button, and more.

The most common input types are described below.

### Text Fields

<input type="text"> defines a one-line input field that a user can enter text into:

```
<form>  
First name: <input type="text" name="firstname"><br>  
Last name: <input type="text" name="lastname">  
</form>
```

The HTML code above looks in a browser:

First name:   
Last name:

The form itself is not visible. Also note that the default width of a text field is 20 characters.

### Password Field

`<input type="password">` defines a password field:

```
<form>  
Password: <input type="password" name="pwd">  
</form>
```

The HTML code above looks in a browser:

Password:

The characters in a password field are masked (shown as asterisks or circles).

### Simple drop-down list

Creating a simple drop-down list.

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

```
<form action="">
<select name="cars">
<option value="volvo">Volvo</option>
<option value="saab">Saab</option>
<option value="fiat">Fiat</option>
<option value="audi">Audi</option>
</select>
</form>

</body>
</html>
```

### Drop-down list with a pre-selected value

Creating a drop-down list with a pre-selected value.

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

<form action="">
<select name="cars">
<option value="volvo">Volvo</option>
<option value="saab">Saab</option>
<option value="fiat" selected>Fiat</option>
<option value="audi">Audi</option>
</select>
</form>

</body>
</html>
```

### Radio Buttons

`<input type="radio">` defines a radio button. Radio buttons let a user select ONLY ONE of a limited number of choices:

```
<form>
<input type="radio" name="sex" value="male">Male<br>
<input type="radio" name="sex" value="female">Female
</form>
```

the HTML code above looks in a browser:

- ☐ Male  
☒ Female

### Checkboxes

`<input type="checkbox">` defines a checkbox. Checkboxes let a user select ZERO or MORE options of a limited number of choices.

```
<form>
<input type="checkbox" name="vehicle" value="Bike">I have a bike<br>
<input type="checkbox" name="vehicle" value="Car">I have a car
</form>
```

How the HTML code above looks in a browser:

- ☐ I have a bike  
☐ I have a car

### Submit Button

`<input type="submit">` defines a submit button.

A submit button is used to send form data to a server. The data is sent to the page specified in the form's action attribute. The file defined in the action attribute usually does something with the received input:

```
<form name="input" action="html_form_action.asp" method="get">  
Username: <input type="text" name="user">  
<input type="submit" value="Submit">  
</form>
```

The HTML code above looks in a browser:

Username:

### Resetting the Form:

HTML <button> type Attribute

### HTML <button> tag

### Definition and Usage

The type attribute specifies the type of button.

Specify the type attribute for the <button> element. Different browsers may use different default types for the <button> element.

### Syntax

```
<button type="button|submit|reset">
```

### Attribute Values

Value	Description
Button	The button is a clickable button

Submit	The button is a submit button (submits form-data)
Reset	The button is a reset button (resets the form-data to its initial values)

### Example

Two button elements that act as one submit button and one reset button (in a form):

```
<form action="demo_form.asp" method="get">  
  First name: <input type="text" name="fname"><br>  
  Last name: <input type="text" name="lname"><br>  
  <button type="submit" value="Submit">Submit</button>  
  <button type="reset" value="Reset">Reset</button>  
</form>
```

### Form Action Attribute:

#### Definition and Usage

The action attribute specifies where to send the form-data when a form is submitted.

### Example

On submit, send the form-data to a file named "demo\_form.asp" (to process the input):

```
<form action="demo_form.asp" method="get">  
  First name: <input type="text" name="fname"><br>  
  Last name: <input type="text" name="lname"><br>  
  <input type="submit" value="Submit">  
</form>
```

The action attribute is supported in all major browsers.

### Syntax

```
<form action="URL">
```

### Attribute Values

Value	Description
-------	-------------

**URL** Where to send the form-data when the form is submitted.

**Possible values:**

- An absolute URL - points to another web site (like action="http://www.example.com/example.htm")
- A relative URL - points to a file within a web site (like action="example.htm")

**Form Method Attribute:**

**Definition and Usage**

The method attribute specifies how to send form-data (the form-data is sent to the page specified in the action attribute).

The form-data can be sent as URL variables (with method="get") or as HTTP post transaction (with method="post").

**Notes on GET:**

- Appends form-data into the URL in name/value pairs
- The length of a URL is limited (about 3000 characters)
- Never use GET to send sensitive data! (will be visible in the URL)
- Useful for form submissions where a user want to bookmark the result
- GET is better for non-secure data, like query strings in Google

**Notes on POST:**

- Appends form-data inside the body of the HTTP request (data is not shown in URL)
- Has no size limitations
- Form submissions with POST cannot be bookmarked

**Syntax**

<form method="get|post">

**Attribute Values**



Value	Description
Get	Default. Appends the form-data to the URL in name/value pairs: URL?name=value&name=value
Post	Sends the form-data as an HTTP post transaction

**Submit a Form Using Post Method**

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<form action="demo_form_method_post.asp" method="post" target="_blank">
```

```
  First name: <input type="text" name="fname"><br>
```

```
  Last name: <input type="text" name="lname"><br>
```

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

```
</form>
```

```
<p>Click on the submit button, and the input will be sent to a page on the server called  
"demo_form_method_post.asp".</p>
```

```
</body>
```

```
</html>
```

**Result:**First name: Last name: 

Click on the submit button, and the input will be sent to a page on the server called  
"demo\_form\_method\_post.asp".

## **Form Enctype Attribute**

### **Definition and Usage**

The formenctype attribute specifies how the form-data should be encoded when submitting it to the server (only for forms with method="post")

The formenctype attribute overrides the enctype attribute of the <form> element.

**Note:** The formenctype attribute is used with type="submit" and type="image".

### **Syntax:**

<input formenctype="value">

### **Attribute Values**

Value	Description
application/x-www-form-urlencoded	Default. All characters are encoded before sent (spaces are converted to "+" symbols, and special characters are converted to ASCII HEX values)
multipart/form-data	No characters are encoded
text/plain	Spaces are converted to "+" symbols, but no special characters are encoded

### **Example:**

Send form-data encoded as "multipart/form-data":

```
<form action="demo_post_enctype.asp"
method="post" enctype="multipart/form-data">
  First name: <input type="text" name="fname"><br>
  Last name: <input type="text" name="lname"><br>
```

```
<input type="submit" value="Submit">
</form>
```

## Working with Cascading Style Sheet:

### CSS syntax:

CSS is a stylesheet language that describes the presentation of an HTML (or XML) document. CSS describes how elements must be rendered on screen, on paper, or in other media.

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
    background-color: #d0e4fe;
}

h1 {
    color: orange;
    text-align: center;
}

p {
    font-family: "Times New Roman";
    font-size: 20px;
}
</style>
</head>
<body><h1>My First CSS Example</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

# My First CSS Example

This is a paragraph.

**Possible Questions**

**PART-B**

**(Each Question carries 2 Marks)**

1. What is HTML?
2. Write the structure of HTML.
3. Name the different list in HTML.
4. How do you create link in HTML?
5. Define absolute path.
6. Define relative path.
7. Define CSS.

**PART-C**

**(Each Question carries 6 Marks)**

1. Reveal the advantages of java.
2. Discuss about Java byte code.
3. Explain J2EE multitier architecture with a neat sketch.
4. Give a detailed note on J2EE and J2SE.
5. Discuss about distributive systems in multi-tier architecture
6. Discuss about EJB tier implementation.
7. Explain the birth of J2EE and why J2EE is important?
8. What are the classifications of client?
9. Give a detailed note on beginning of Java.
10. Explain the following
  - i) Web Tier Implementation
  - ii) Client Tier Implementation

**UNIT- I**

S.No	Question	Option1	Option2
1	HTML stands for _____ In HTML _____ commands are used to tell the browser software, the structure of the document & how you want the content to be	HyperText Markup Language	HyperText Manual Language
2	displayed If you want to display a section of the text in bold	Manual	Automatic
3	face, then _____ tags are used _____ document is simply a text file that	<P> & </P>	<H1> & </H1>
4	contains the information you want to publish HTML also contains embedded instructions called _____ that indicate how a web browser	HTML	Browser
5	should structure the document HTML Elements generally consist of a pair of	Attributes	Elements
6	_____ tags surrounding some text. The end tag is just like the start tag except that it	Square Brackets	Curly Brackets
7	has a _____ _____	line Horizontal	an arrow
8	The <HR> tag is used for _____ In HTML, some elements such as _____	Rule tag	HyperRule Tag
9	do not have a corresponding end tag	<HEAD> World Wide Web Consortium	<B> Worldly Wide Consortium
10	The W3C stands for _____ The _____ is the primary organisation that	Consortium	Consortium
11	attempts to standardise HTML	DTD Standard Geometric Manual Language Document	SGML Standard Generalised Measurable Language
12	SGML stands for _____	Terminal Definition	Document Type Definition
13	The DTD stands for _____ The W3C has defined HTML as an application of	DTD	SGML
14	the _____ _____ is a language used to define other languages by specifying the allowed document	DTD	SGML
15	structure in the form of a DTD _____ is a document that indicates the	HTML	SGML
16	syntax that can be used for elements	DOC	DTD

17	All proper HTML files should begin with _____	<!DOCTYPE	>	<TITLE>
18	Most HTML files begin with the <HTML> elements which indicates that the content of the file includes _____	Command		Tags
19	The HTML Element contains 2 primary sections such as _____ & _____	Head & Title		Head & Body
20	The _____ element includes supplementary information about the document	<TITLE>		<HTML>
21	The information contained within the <HEAD> element is the information about information generally referred to as _____	Meta Information		Small Information
22	The search engines such as Lycos & Hotbot use _____ information to index webpages	Web		Internet
23	The _____ tags contain the actual content and the appropriate markup tags needed to render the page	<HTML>		<HEAD>
24	The _____ header tag creates a headline	<HR>		<H1>
25	The _____ element inserts a bar across the screen	<HR>		<H1>
26	The _____ tag indicates a paragraph of text	<P>		<B>
27	The _____ tag specifies the title of the document	<HTML>		<TITLE>
28	If you are using a word processing program, you can type the example & save it with a file name such as _____	first.shtml		first.html
29	A markup language such as HTML is simply a collection of _____	elements		attributes
30	CITE stands for _____	Character Extraction		Citation
31	The Alphanumeric tags are _____	Not Case Sensitive		Case Sensitive
32	Attribute value should be enclosed within _____	single quotes		double quotes
33	_____ means that HTML elements can surround each other	crossing extensible		switching
34	Expand XML _____	markup lg		xtra markup lg
35	_____ HTML refers to using HTML to make pages look a particular way	physical		logical
36	_____ HTML refers to using HTML to specify the structure of a document	physical		special

		what you see	
		is what you	what you get is
37	expand wysiwyg	get	what you see
38	In HTML, you can make something _____	italics	underlined
39	Expand for DHTML	design HTML	dynamic HTML
	The structure of HTML document is specified by a		
40	_____	SGML	DHTML
	_____ is a structured markup language that is used to		
41	create webpages	SGML	HTML
	The HTML specification defines the type of		
	content that an element can enclose known as		definition
42	elements	content model	model
43	HTML documents are _____ documents	non structured	structured
44	Element name cannot contain _____	quotes	spaces
45	Browsers _____ known elements & attributes	do not ignore	ignore
	An HTML document begins with a _____		
46	declaration	<HTML>	<HEAD>
	The head can contain identifying & other		
	supplementary information about the document or		
47	_____ information	structure	format
	The _____ element encloses a document		
	section that contains identification &		
48	supplementary information about the document	<body>	<html>
	The _____ element specifies an absolute URL		
	address that is used to provide server & directory		
49	information	<isindex>	<link>
	The _____ element indicates that the		
50	document contains a searchable index	<ISINDEX>	<FORINDEX>
	The _____ element specifies a special		
	relationship between the current document &		
51	another document	<ALINK>	<Attribute>
	The _____ element uses name/value pairs to		
52	provide meta-information about a document	<LINK>	<Base>
	A document's _____ section contains all the		
53	descriptive information about the document	<head>	<body>
	The _____ element allows programs & other		
	binary objects to be directly embedded in a web		
54	page	<base>	<script>
	The _____ element allows programs written		
	in a scripting language to be directly embedded in		
55	a web page	<style>	<base>

According to HTML 4.0 specification, only one _____ element should appear in every		
56 document	<head>	<html>
Common attributes for the _____ element affect the colors for a documents text,		
57 background & links	<head>	<body>
58 A link stands for _____	Active links	Allone links
59 V link stands for _____	Variety link	Visited link
netscape navigator & internet explorer display around _____ characters of a title in their		
60 bookmark list	10 to 20	25 to 50



Option3	Option4	Answer
HigherText Markup Language	HeavyText Markup Language	HyperText Markup Language
Markup	Machine	Markup
<UL> & </UL>	<B> & </B>	<B> & </B>
Text	Doctype	HTML
Markup Commands	tags	Elements
Angle Brackets	Less than	Angle Brackets
slash	asterisk	slash
HighRule Tag	Heavy Rule Tag	Horizontal Rule tag
<HR>	<I>	<HR>
Web World Consortium	Wide Web Consoetium	World Wide Web Consortium
W3C Secured Generalised Markup Language	XML Standard Generalised Markup Language	W3C Standard Generalised Markup Language
Dictionary To Dictionary	Definition To Definition	Document Type Definition
DOCTYPE	HTML	SGML
XML	ASP	SGML
D To D	XML	DTD

<BODY>	<HEAD>	<!DOCTYPE>
--------	--------	------------

Markup	Attribute	Markup
--------	-----------	--------

Title & Body	Head & Title	Head & Body
--------------	--------------	-------------

<HEAD>	<body>	<HEAD>
--------	--------	--------

Mini Information	Wide information	Meta Information
---------------------	---------------------	---------------------

Meta	Mini	Meta
------	------	------

<BODY> <P>	<TITLE> <HEAD>	<BODY> <H1>
---------------	-------------------	----------------

<P>	<HEAD>	<HR>
-----	--------	------

<H1>	<HR>	<P>
------	------	-----

<HEADING>	<H1>	<TITLE>
-----------	------	---------

first.html	first.doc	first.html
------------	-----------	------------

codes	contents	codes
-------	----------	-------

Coding Enable	Context	Citation
Uppercase Only	Lowercase Only	Not Case Sensitive

dotted lines	exclamatory	double quotes
--------------	-------------	---------------

nesting exclusive model lg	block extra-modeling language	nesting extensible markup lg
----------------------------------	-------------------------------------	------------------------------------

secondary	primary	physical
-----------	---------	----------

secondary	logical	logical
-----------	---------	---------

whichever you see is whichever you get bold dynamo HTML	get highlight dialogue HTML	what you see is what you get bold dynamic HTML
---	-----------------------------------	---

XML	DTD	DTD
-----	-----	-----

DHTML	XML	HTML
-------	-----	------

document model	element model	content model
-------------------	---------------	---------------

both a & b comments both a & b	unified blanks blanks	structured spaces ignore
--------------------------------------	-----------------------------	--------------------------------

<DOCTYPE>	<SCRIPT>	<DOCTYPE>
-----------	----------	-----------

meta	mini	meta
------	------	------

<title>	<head>	<head>
---------	--------	--------

<base>	<meta> <METAINDEX <ISSEARCH>	<base> <ISINDEX>
--------	------------------------------------	---------------------

<LINK>	<Base>	<LINK>
--------	--------	--------

<meta>	<style>	<meta>
--------	---------	--------

<title>	<html>	<head>
---------	--------	--------

<style>	<object>	<object>
---------	----------	----------

<script>	<meta>	<script>
----------	--------	----------

<body>	<title>	<title>
<title>	<style>	<body>
Accept links	Unvisted Link	Active links
Very high link	Unvisted Link	Visited link
20 to 30	30 to 40	20 to 30

**UNIT-II**

**SYLLABUS**

**JavaScript:** Introduction to javascript – Programming fundamentals – Functions and objects – Navigator object model

## Introduction to JavaScript:

JavaScript is a scripting language that will allow us to add real programming to our webpages. We can create small application type processes with javascript, like a calculator or a primitive game of some sort.

There are more serious uses for javascript:

- **Browser Detection**  
Detecting the browser used by a visitor at web page. Depending on the browser, another page specifically designed for that browser can then be loaded.
- **Cookies**  
Storing information on the visitor's computer, then retrieving this information automatically next time the user visits we page. This technique is called "cookies".
- **Control Browsers**  
Opening pages in customized windows, where we specify if the browser's buttons, menu line, status line or whatever should be present.
- **Validate Forms**  
Validating inputs to fields before submitting a form.  
An example would be validating the entered email address to see if it has an @ in it, since if not, it's not a valid address.

## JavaScript Origins

JavaScript was released by Netscape and Sun Microsystems in 1995. JavaScript is not the same thing as Java.

### What is JavaScript?

- It is a programming language.
- It is an interpreted language.
- It is object-based programming.
- It is widely used and supported

- It is accessible to the beginner.

### ***Uses of JavaScript***

- Use it to add multimedia elements

With JavaScript we can show, hide, change, resize images, and create image rollovers. We can create scrolling text across the status bar.

### **• Create pages dynamically**

Based on the user's choices, the date, or other external data, JavaScript can produce pages that are customized to the user.

### **• Interact with the user**

It can do some processing of forms and can validate user input when the user submits the form.

### **Writing JavaScript**

JavaScript code is typically embedded in the HTML, to be interpreted and run by the client's browser. Here are some tips to remember when writing JavaScript commands.

- JavaScript code is case sensitive
- White space between words and tabs are ignored
- Line breaks are ignored except within a statement
- JavaScript statements end with a semi- colon ;

### **The SCRIPT Tag**

The <SCRIPT> tag alerts a browser that JavaScript code follows. It is typically embedded in the HTML.

```
<SCRIPT language = "JavaScript">  
statements  
</SCRIPT>
```

### **SCRIPT Example**

- Open "script\_tag.html" in a browser.
- View the Source
- Put the cursor after <! – Enter code below Æ and enter the following:

```
<SCRIPT language = "JavaScript">  
alert("Welcome to the script tag test page.")  
</SCRIPT>
```

- Save the changes by choosing Save from the File menu.
- Then Refresh the browser by clicking the Refresh or Reload button

### Implementing JavaScript

There are three ways to add JavaScript commands to we Web Pages.

- Embedding code
- Inline code
- External file

#### External File

We can use the SRC attribute of the <SCRIPT> tag to call JavaScript code from an external text file. This is useful if we have a lot of code or we want to run it from several pages, because any number of pages can call the same external JavaScript file. The text file itself contains no HTML tags. It is call by the following tag:

```
<SCRIPT SRC="filename.js">  
</SCRIPT>
```

#### External Example

- Open "external.html" in a browser
- View the Source
- Put the cursor after <! – Enter code here Æ and enter:  
<SCRIPT language = "JavaScript" SRC = "external.js">  
</SCRIPT>
- Save the changes and Refresh the browser.

## Programming Fundamentals:

### JavaScript Data Types

String, Number, Boolean, Array, Object, Null, Undefined.

JavaScript Has Dynamic Types

JavaScript has dynamic types. This means that the same variable can be used as different types:

Example

```
var x;           // Now x is undefined
var x = 5;       // Now x is a Number
var x = "John";  // Now x is a String
```

### JavaScript Strings

A string is a variable which stores a series of characters like "John Doe".

A string can be any text inside quotes. We can use single or double quotes:

**Example**

```
var carname="Volvo XC60";
var carname='Volvo XC60';
```

Can use quotes inside a string, as long as they don't match the quotes surrounding the string:

**Example**

```
var answe="It's alright";
var answe="He is called 'Johnny'";
var answe='He is called "Johnny"';
```

### JavaScript Numbers

JavaScript has only one type of numbers. Numbers can be written with, or without decimals:

**Example**

```
var x1=34.00;    // Written with decimals
var x2=34;       // Written without decimals
```

Extra large or extra small numbers can be written with scientific (exponential) notation:

**Example**

```
var y=123e5;     // 12300000
var z=123e-5;    // 0.00123
```



### JavaScript Booleans

Booleans can only have two values: true or false.

```
var x=true;
```

```
var y=false;
```

Booleans are often used in conditional testing.

### JavaScript Arrays

The following code creates an Array called cars:

```
var cars=new Array();
```

```
cars[0]="Saab";
```

```
cars[1]="Volvo";
```

```
cars[2]="BMW";
```

or (condensed array):

```
var cars=new Array("Saab","Volvo","BMW");
```

or (literal array):

#### Example

```
var cars=["Saab","Volvo","BMW"];
```

### JavaScript Variables:

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

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

```
<script type="text/javascript">
```

```
<!--
```

```
var money;
```

```
var name;
```

```
//-->
```

```
</script>
```

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

```
<script type="text/javascript">
```

```
<!--
```

```
var money, name;
```

```
//-->
```

```
</script>
```

Storing a value in a variable is called variable initialization. We can do variable initialization at the time of variable creation or later point in time when we need that variable as follows:

For instance, we might create a variable named *money* and assign the value 2000.50 to it later.

For another variable we can assign a value the time of initialization as follows:

```
<script type="text/javascript">
```

```
<!--
```

```
var name = "Ali";
```

```
var money;
```

```
money = 2000.50;
```

```
//-->
```

```
</script>
```

### JavaScript Variable Scope:

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

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

Within the body of a function, a local variable takes precedence over a global variable with the same name. If we declare a local variable or function parameter with the same name as a global variable, we effectively hide the global variable. Following example explains it:

```
<script type="text/javascript">
```

```
<!--
```

```
var myVar = "global"; // Declare a global variable
```

```
function checkscope( ) {
```

```
    var myVar = "local"; // Declare a local variable
```

```
    document.write(myVar);
```

```
}
```

```
//-->
```

```
</script>
```

This produces the following result:

Local

**JavaScript Variable Names:**

While naming variables in JavaScript keep following rules in mind.

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

**Operators in Java Script:**

What is an operator?

It can be given by using the expression *4 + 5 is equal to 9*. Here 4 and 5 are called operands and + is called operator. JavaScript language supports following type of operators.

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

Lets have a look on all operators one by one.

**The Arithmetic Operators:**

There are following arithmetic operators supported by JavaScript language:

Assume variable A holds 10 and variable B holds 20 then:

Operator	Description	Example
+	Adds two operands	A + B will give 30
-	Subtracts second operand from the first	A - B will give -10
*	Multiply both operands	A * B will give 200
/	Divide numerator by denominator	B / A will give 2
%	Modulus Operator and remainder of after an integer	B % A will give 0

	division	
++	Increment operator, increases integer value by one	A++ will give 11
--	Decrement operator, decreases integer value by one	A-- will give 9

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

### **The Comparison Operators:**

There are following comparison operators supported by JavaScript language  
Assume variable A holds 10 and variable B holds 20 then:

<b>Operator</b>	<b>Description</b>	<b>Example</b>
==	Checks if the value of two operands are equal or not, if yes then condition becomes true.	(A == B) is not true.
!=	Checks if the value of two operands are equal or not, if values are not equal then condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true.

### **The Logical Operators:**

There are following logical operators supported by JavaScript language  
Assume variable A holds 10 and variable B holds 20 then:

Operator	Description	Example
&&	Called Logical AND operator. If both the operands are non zero then then condition becomes true.	(A && B) is true.
	Called Logical OR Operator. If any of the two operands are non zero then then condition becomes true.	(A    B) is true.
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!(A && B) is false.

**The Bitwise Operators:**

There are following bitwise operators supported by JavaScript language  
Assume variable A holds 2 and variable B holds 3 then:

Operator	Description	Example
&	Called Bitwise AND operator. It performs a Boolean AND operation on each bit of its integer arguments.	(A & B) is 2 .
	Called Bitwise OR Operator. It performs a Boolean OR operation on each bit of its integer arguments.	(A   B) is 3.
^	Called Bitwise XOR Operator. It performs a Boolean exclusive OR operation on each bit of its integer arguments. Exclusive OR means that either operand one is true or operand two is true, but not both.	(A ^ B) is 1.
~	Called Bitwise NOT Operator. It is a unary operator and operates by reversing all bits in the operand.	(~B) is -4 .
<<	Called Bitwise Shift Left Operator. It moves all bits in its first operand to the left by the number of places specified in the second operand. New bits	(A << 1) is 4.

	are filled with zeros. Shifting a value left by one position is equivalent to multiplying by 2, shifting two positions is equivalent to multiplying by 4, etc.	
>>	Called Bitwise Shift Right with Sign Operator. It moves all bits in its first operand to the right by the number of places specified in the second operand. The bits filled in on the left depend on the sign bit of the original operand, in order to preserve the sign of the result. If the first operand is positive, the result has zeros placed in the high bits; if the first operand is negative, the result has ones placed in the high bits. Shifting a value right one place is equivalent to dividing by 2 (discarding the remainder), shifting right two places is equivalent to integer division by 4, and so on.	(A >> 1) is 1.
>>>	Called Bitwise Shift Right with Zero Operator. This operator is just like the >> operator, except that the bits shifted in on the left are always zero,	(A >>> 1) is 1.

**The Assignment Operators:**

There are following assignment operators supported by JavaScript language:

<b>Operator</b>	<b>Description</b>	<b>Example</b>
=	Simple assignment operator, Assigns values from right side operands to left side operand	C = A + B will assign value of A + B into C
+=	Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand	C += A is equivalent to C = C + A
-=	Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand	C -= A is equivalent to C = C - A
*=	Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand	C *= A is equivalent to C = C * A

/=	Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand	C /= A is equivalent to C = C / A
%=	Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand	C %= A is equivalent to C = C % A

### The Conditional Operator ( ? : )

There is an operator called conditional operator. This first evaluates an expression for a true or false value and then execute one of the two given statements depending upon the result of the evaluation. The conditional operator has this syntax:

Operator	Description	Example
? :	Conditional Expression	If Condition is true ? Then value X : Otherwise value Y

### The *typeof* Operator

The *typeof* is a unary operator that is placed before its single operand, which can be of any type. Its value is a string indicating the data type of the operand.

The *typeof* operator evaluates to "number", "string", or "boolean" if its operand is a number, string, or boolean value and returns true or false based on the evaluation.

Here is the list of return values for the *typeof* Operator :

Type	String Returned by <i>typeof</i>
Number	"number"
String	"string"
Boolean	"boolean"
Object	"object"
Function	"function"

Undefined	"undefined"
Null	"object"

**Data Type Conversion:**

The following data type conversions are possible

- a) String to Integer (int)
- b) Float to Integer (int)
- c) String to Float
- d) Integer (int) to String
- e) Float to String

**a) Converting String to Integer**

To convert a value from string to integer we have to use the function or method "**parseInt()**". Passing a correct argument to the function (e.g: "4") will return correct value else (e.g: "sd") 0 will be returned.

Example	Result
parseInt("4");	4
parseInt("5aaa");	5
parseInt("4.33333");	4
parseInt("aaa");	NaN (means "Not a Number")

**b) Converting Float to Integer**

To convert a float value in to integer we have to use the function or method parseInt.

Example	Result
parseInt(5.133);	5



**Example Code:**

```
<script language="javascript">
var a = "334";
var b = 3;
var c = parseInt(a)+b;
var d = a+b;
document.write("parseInt(a)+b = "+c);
document.write(" a+b = "+d);
</script>
```

**Result:**

parseInt(a)+b = 337  
a+b = 3343

**c) Converting String to Float**

To convert a value from string to float, we have to use the function or method "**parseFloat()**" to convert a string value to float.

Example	Result
parseFloat("4.333");	4.333
parseFloat("5aaa");	5
parseFloat("6e2");	600
parseFloat("aaa");	NaN (means "Not a Number")

**d) Converting Integer/Float to String**

Integer (int) or float value can be converted to string by using the function or method toString().

Example	Result

**CLASS: II BCA**  
**COURSE CODE: 18CAU403**

**COURSE NAME: WEB PROGRAMMING**  
**UNIT: II (JavaScript)      BATCH: 2018-2021**

var a = 3.22; a.toString();	"3.22"
var a = 5; a.toString();	5

**Example Code:**

```
<script language="javascript">
var a = 32;
var b = 333;
var c = a.toString()+b;
document.write(" to String function "+c);
</script>
```

**Result:**

to String function -> 32333

**c) Converting String to Float**

To convert a value from string to float, we have to use the function or method "**parseFloat()** to convert a string value to float.

<b>Example</b>	<b>Result</b>
parseFloat("4.333");	4.333
parseFloat("5aaa");	5
parseFloat("6e2");	600
parseFloat("aaa");	NaN (means "Not a Number")

**d) Converting Integer/Float to String**

Integer (int) or float value can be converted to string by using the function or method toString().

Example	Result
var a = 3.22; a.toString();	"3.22"
var a = 5; a.toString();	5

**Example Code:**

```
<script language="javascript">  
var a = 32;  
var b = 333;  
var c = a.toString()+b;  
document.write(" to String function "+c);  
</script>
```

**Result:**

to String function -> 32333

**Comments in Javascript:**

JavaScript supports three different types of comments:

Multiple-line C-style comments. Everything between `/*` and `*/` is a comment, for example:

```
/* This is a comment */
```

```
/* C-style comments can span
```

```
as many lines as we like,
```

```
as shown in this example */
```

One-line comments of C++ style. These comments begin with `//` and continue up to the next line break:

```
// This is a one-line comment
```

One-line comments with the *HTML comment-opening sequence* (`<!--`). Note that the JavaScript interpreter ignores the closing characters of HTML comments (`-->`). Consider this example:

```
<!-- This is treated as a one-line JS comment
```

```
<!-- It works just like a comment beginning with //
```

```
<! ----> This is also a one-line JS comment
```

```
<! ----> because JS ignores the closing characters
```

```
<! ----> of HTML-style comments
```

## Conditional Statements

If statement is used to check or verify a condition and execute a set of statement only if the condition is true. This should be referred as a statement and not as a function.

### Syntax:

```
if(condition)
{
    // set of statements that will be executed
    // only if the condition is true or satisfied
}
```

### Example Code:

```
<script language="javascript">
var a = "if check";
if(a == "if check")
{
    document.write(" inside if statement ");
}
</script>
```

### Result:

inside if statement

### Nested If:

Nested if statement is nothing but using an if statement inside another if statement. Nested if is used when we check a condition only when another condition is true. For an example when we purchase a car, first we verify is the car looks good, only if it satisfies we go to the next condition color, then next and so on..

### Syntax: Nested if

```
if(condition 1)
{
    if(condition 2)
    {
        // set of statements that will be executed
    }
}
```

```
    }  
}
```

**If else statement:**

If else statement also has the same format as that of "if" statement. Only additional thing is that an else part is added to if statement. So if the condition satisfies the statements inside if part will be executed else the statement inside else part will be executed.

**Syntax:**

```
if(condition){  
    // set of statements if condition satisfies  
}  
else{  
    // set of statements if condition fails  
}
```

**Example Code:**

```
<script language="javascript">  
var a = "1234abc";  
if(a == "adcdefa"){  
    document.write(" inside if statement ");  
}else{  
    document.write(" inside else part of statement ");  
}  
</script>
```

**Result:**

inside else part of statement

In the above example the condition is to check if variable 'a' equals (==) "adcdefa". The condition fails as we have assigned a as "1234abc". So the else part is executed.

**Switch case:**

Switch case is used to when a condition may have multiple results and a different set of operation is done based on each result..

**Syntax:**

**Switch(condition)**

```
{  
case result1:  
    // Operation for result1  
  
case result2:  
    // Operation for result2  
.  
.  
.  
default :  
    // If result belongs to none of the case specified }
```

**Example Code:**

```
<script language="javascript">  
for(var i=1; i<5; i++)  
{  
switch(i)  
{  
case 1:  
    document.write("message for case 1 <br>");  
    break;  
case 2:  
    document.write("message for case 2 <br>");  
    break;  
case 3:  
    document.write("message for case 3 <br>");  
    break;  
default:  
    document.write("message for case default<br>");  
    break;  
}  
}  
</script>
```

**Result:**

message for case 1  
message for case 2

message for case 3

message for case default

"with" statement is used when numerous function of an object is used or a function of an object is to be used numerous times.

### With Statement:

#### Syntax:

```
with(object)
{
    // Calling the functions or methods of the object
}
```

For this example we will use the object "document" and its methods (functions) "write" and attribute "title".

#### Example Code:

```
<script language="javascript">
with(document)
{
    write(" inside with statement <br>");
    write(" using with(object) we can call its functions directly <br>");
    write (" TITLE - "+title);
}
</script>
```

#### Result:

inside with statement

using with(object) we can call its functions directly

TITLE - With Statement Code - Javascript (JS) Tutorial

In the above example we can clearly see that the write function is executed numerous times without calling document.write() and also title (document.title) is called. We can use any objects with "with" statement. e.g: date, math, etc.

**LOOPING STATEMENTS:****for LOOP:**

A set of statements are executed as a loop until a condition is satisfied, the condition is based on an incremental or decremental counter. In other words "Looping statements in javascript are used to execute the same set of code a specified number of times".

**Syntax:**

```
for(initialvalue; condition; increment)
{
    // set of statements that will be executed
}
```

As defined in the syntax, for loop takes three parameters, the initial value (e.g i=0), condition - the statements inside "for" will be executed until this condition is satisfied (e.g i<7), increment - this is where we set the initial value to be increased or decreased after each loop.

All the three parameters are separated by semicolon ";".

For an example, we will consider a situation where we want to add all numbers between one and ten.

**Example Code:**

```
<script language="javascript">
var i=0; var total=0;
for(i=1; i<11; i++)
{
    total = total+i;
}
document.write("----- The total ----- "+total);
</script>
```

**Result:**

----- The total-----45

**For... in Statement:**

There is one more loop supported by JavaScript. It is called **for...in** loop. This loop is used to loop through an object's properties.



Because we have not discussed Objects yet, so we may not feel comfortable with this loop. But once we will have understanding on JavaScript objects then we will find this loop very useful.

**Syntax:**

```
for (variablename in object){  
    statement or block to execute  
}
```

In each iteration one property from *object* is assigned to *variablename* and this loop continues till all the properties of the object are exhausted.

**Example:**

Here is the following example that prints out the properties of a Web browser's **Navigator** object:

```
<script type="text/javascript">  
<!--  
var aProperty;  
document.write("Navigator Object Properties<br /> ");  
for (aProperty in navigator)  
{  
    document.write(aProperty);  
    document.write("<br />");  
}  
document.write("Exiting from the loop!");  
//-->  
</script>
```

This will produce following result:

Navigator Object Properties

appCodeName

appName

appMinorVersion

cpuClass

platform

plugins

opsProfile

userProfile

systemLanguage

userLanguage

appVersion

userAgent

onLine

cookieEnabled

mimeTypes  
Exiting from the loop!

### While Loop:

#### Explanation

'while' loop is used to execute a set of statements repeatedly until a condition works true. The difference between 'for' and 'while' loop is that 'while' does not take counter as an argument.

#### Syntax:

```
while(condition)
{
    // set of statements that will be executed
}
```

As defined in the syntax, while loop has only one parameter, condition to be validated. The statements inside "while" will be executed until this condition becomes false.

For an example, we will consider a situation where we want to print first 5 number.

#### Example Code:

```
<script language="javascript">
var i=0;
while(i<5)
{
    document.write("The value of i is - "+i+" ");
    i++;
}
</script>
```

#### Result:

The value of i is - 0  
The value of i is - 1  
The value of i is - 2  
The value of i is - 3  
The value of i is - 4

### Do...While

'do-while' loop is similar to while loop and the only difference here is that the set of statements are executed first and the condition is checked next.

#### Syntax:

```
do
{
    // set of statements that will be executed
}
while(condition)
```

Here the statements are added under do loop and while condition is checked at the end of loop. In 'Do While' the statements are executed once even if the condition will fail.

An example

#### Example Code:

```
<script language="javascript">
var i=0;
do
{
    document.write("Testing DO-While loop");
}
while(i!=0)
</script>
```

#### Result:

Testing DO-While loop

#### Break statement :

Break is a statement used to exit or terminate a loop in execution. It is used in "for, while, do-while" looping statements. **Break statement** is used mostly with a conditional statement inside the loop. When the condition satisfies the control breaks/terminates from the loop and moves to the next line below the loop.

For an example, we will use a for loop that prints 1 to 5 but will use break or exit the loop iteration when i is 3.

**Example Code:**

```
<script language="javascript">
for(var i=0; i<5; i++)
{
    if(i == 3)
        break;
    document.write("i is - "+i);
}
document.write(" ----- After Looping -----");
</script>
```

**Result:**

i is - 0  
i is - 1  
i is - 2

**Continue statement :**

**Continue statement** is used to stop or terminate one iteration of the loop in execution. It is used in "for, while, do-while" looping statements. Continue statement unlike break statement does not completely terminate the loop. It stops processing for only one iteration and brings the control back to the beginning of the loop.

For an example, we will try to stop processing inside a for loop when i is 2.

**Example Code:**

```
<script language="javascript">
for(var i=0; i<5; i++)
{
    if(i == 2)
        continue;
    document.write("i is - "+i);
}
</script>
```

**Result:**

i is - 0  
i is - 1  
i is - 3  
i is - 4

**Functions and Objects:**

A function is nothing but a group or block of statements doing a specific task.

**Syntax:**

```
function name()
{
    // set of statements that will be executed
}
```

A function has to be defined in the above syntax.

The name "function" followed by the name we choose for the function and then open and close brackets. The statements that will do the specific operation are then group together under this name using braces.

A function may or may not return a value. A function may or may not have parameters value.

**Invoking a function:**

The statements inside the function will not be executed automatically. We have to invoke or call the function to execute the statements. Just calling the name of the function will invoke the function. i.e. if we write a function with the name "test" calling it as "test();" will invoke the function.

**Example Code:**

```
<script language="javascript">
function test()
{
    document.write(" ---- This is a test function ---- ");
}
```

```
}  
  
test();  
</script>
```

**Result:**

---- This is a test function ----

**Syntax:**

```
function name(parameter 1,parameter 2,...)  
{  
    // set of statements that will be executed  
}
```

**Passing Parameters to Functions:**

In many cases we pass parameters or arguments to functions, these arguments will be used inside the function for required calculations. For an example we will use two numbers to add, subtract using function.

Here we write separate function for each operations add, subtract.

**Example Code:**

```
<script language="javascript">  
  
function add(number1, number2)  
{  
    var c = number1+number2;  
    document.write(" ---- This added value is ---- "+c;  
}  
  
function sub(number1, number2)  
{  
    var c = number1-number2;  
    document.write(" ---- This subtracted value is ---- "+c;  
}  
  
var a = 7;
```

```
var b = 3;  
add(a,b);  
sub(a,b);  
</script>
```

**Result:**

---- This added value is ---- 10  
---- This subtracted value is -----4

Here we can clearly see that the two functions were invoked as "add(a,b);" and "sub(a,b);" where a and b are defined variables. We can even call the function directly with the variables as say "add(9,1);".

A function can be invoked any number of times with any proper value.

**Returning Values:**

Syntax:

```
function name(parameter 1,parameter 2,...)  
{  
    // set of statements that will be executed  
    return thevalue;  
}
```

A function can also **return a value** after doing the specified operations. To explain, we would consider a requirement where we have to calculate  $x^2/2$ .

We will calculate  $x^2$  in a function and return the result of  $x^2$ . After getting the return value, we will divide it by 2.

**Example Code:**

```
<script language="javascript">
```

```
function square(number1)  
{  
    var c = number1*number1;  
    // Now we will return the result  
    return c;  
}
```

```
var x = 4;  
// Here we invoke the function and capture the result  
var des = square(x);  
var res = des/2;  
document.write(" The result - "+res);  
</script>
```

**Result:**

The result - 8

**Predefined Functions:**

Javascript is an object oriented programming language. It supports the concept of objects in the form of attributes. If an object attribute consists of function, then it is a method of that object, or if an object attribute consists of values, then it is a property of that object. For example,

```
var status=document.readyState;
```

readyState is a property of the document object which can contain values such as "uninitialized", "loading", "interactive", "complete" whereas,

```
document.write("Hello World");
```

write() is a method of the document object that writes the content "Hello World" on the web page. There are several Javascript built-in objects such as,

- Number
- String
- RegExp
- Array
- Math
- Date
- Boolean

Each of the above objects hold several built-in functions to perform object related functionality. Apart from these methods, Javascript provides few predefined functions which do not stick to a particular object type but are global. These global built-in functions are explained below with examples.



### isNaN()

isNaN() method determines whether value of a variable is a legal number or not.

```
document.write(isNaN(0));  
document.write(isNaN("Javascript"));  
document.write(isNaN(-2.45));  
document.write(isNaN("77"));  
document.write(isNaN("2012/4/8"));
```

Results:

**false**

**true**

**false**

**false**

**true**

*Note: There is a property called NaN of the Number Object which can be used to assign a variable with 'not a number' value.*

```
var year= Number.NaN;  
document.write(year);
```

Result: **NaN**

### isFinite()

As the name indicates, this function is used to find whether a number is a finite legal number.

```
document.write(isFinite("5678"));  
document.write(isFinite("ABCD"));  
document.write(isFinite("123_456"));
```

Result:

**true**

**false**

**false**

### **eval()**

eval() is used to execute Javascript source code. It evaluates or executes the argument passed to it and generates output.

```
eval("var number=2;number=number+2;document.write(number)");
```

Result:

4

### **Number()**

Number() method takes an object as an argument and converts it to the corresponding number value. If the object passed cannot be converted to a number, that is if the object is not in a format to be represented as a number, then it returns NaN(not a number).

```
var obj1=new String("123");  
var obj2=new Boolean("false");  
var obj3=new Boolean("true");  
var obj4=new Date();  
var obj5=new String("9191 9999");  
document.write(Number(obj1));  
document.write(Number(obj2));  
document.write(Number(obj3));  
document.write(Number(obj4));  
document.write(Number(obj5));
```

Result:

123  
0  
1  
1342720050291  
NaN

### String()

String() function converts the object argument passed to it to a string value.

```
var obj1=new Boolean(0);  
var obj2=new Boolean(1);  
var obj3=new Date();  
document.write(String(obj1));  
document.write(String(obj2));  
document.write(String(obj3));
```

Result:

**false**

**true**

**Thu Jul 19 2012 23:28:08 GMT+0530 (India Standard Time)\**

### parseInt()

parseInt() function takes string as a parameter and converts it to integer.

```
document.write(parseInt("50"));  
document.write(parseInt("77 days"));  
document.write(parseInt("this is 7"));
```

Result:

**50**

**77**

**NaN**

An optional radix parameter can also be used to specify the number system to be used to parse the string argument. For example,

```
document.write(parseInt("10",16));
```

Result:

**16**

If the radix parameter is not specified, then Javascript

- Assumes radix to be 16(hexadecimal), if the string starts with "0x"
- Assumes radix to be 8(octal), if the string starts with 0
- Assumes radix to be 10(decimal), if the string starts with any other number.

### parseFloat()

parseFloat() function takes a string as parameter and parses it to a floating point number.

```
document.write(parseFloat("10.33"));  
document.write(parseFloat("15 66 75"));  
document.write(parseFloat("this is 77"));  
document.write(parseFloat(" 77 "));
```

Result:

**10.33**  
**15**  
**NaN**  
**77**

This function allows leading and trailing spaces. If the first character in the string is not a number, then it returns NaN. If the string has more than one set of number separated by delimiters such as spaces, semicolons, commas then it returns only the first set of number before the first delimiter.

### escape()

escape() function encodes the string passed to it so that it can be used across any network, say for example in query strings.

```
document.write(escape("testing escape function!!"));
```

**Result:**

**testing%20escape%20function%21%21**

**Alert Dialog Box:**

An alert dialog box is mostly used to give a warning message to the users. Like if one input field requires to enter some text but user does not enter that field then as a part of validation we can use alert box to give warning message as follows:

```
<head>
<script type="text/javascript">
<!--
    alert("Warning Message");
//-->
</script>
</head>
```

An alert box can still be used for friendlier messages. Alert box gives only one button "OK" to select and proceed.

**Array object:**

The **Array** object let's we store multiple values in a single variable.

Syntax:

Creating a **Array** object:

```
var fruits = new Array( "apple", "orange", "mango" );
```

The **Array** parameter is a list of strings or integers. When we specify a single numeric parameter with the Array constructor, we specify the initial length of the array. The maximum length allowed for an array is 4,294,967,295.

We can create array by simply assigning values as follows:

```
var fruits = [ "apple", "orange", "mango" ];
```

We will use ordinal numbers to access and to set values inside an array as follows:

- fruits[0] is the first element
- fruits[1] is the second element
- fruits[2] is the third element

**Boolean object:**

The **Boolean** object represents two values either "true" or "false".

Syntax:

Creating a **boolean** object:

```
var val = new Boolean(value);
```

If *value* parameter is omitted or is 0, -0, null, false, NaN, undefined, or the empty string (""), the object has an initial value of false.

Boolean Properties:

Here is a list of each property and their description.

Property	Description
<u>constructor</u>	Returns a reference to the Boolean function that created the object.
<u>prototype</u>	The prototype property allows we to add properties and methods to an object.

### String object:

#### Syntax:

Creating a **String** object:

```
var val = new String(string);
```

The *string* parameter is series of characters that has been properly encoded.

#### String Properties:

Here is a list of each property and their description.

Property	Description
constructor	Returns a reference to the String function that created the object.
<u>length</u>	Returns the length of the string.
<u>prototype</u>	The prototype property allows we to add properties and methods to an object.

### Number object:

The **Number** object represents numerical date, either integers or floating-point numbers. In general, we do not need to worry about **Number** objects because the browser automatically converts number literals to instances of the number class.

Syntax:

Creating a **number** object:

```
var val = new Number(number);
```

If the argument cannot be converted into a number, it returns NaN (Not-a-Number).

Number Properties:

Here is a list of each property and its description.

Property	Description
<u>MAX_VALUE</u>	The largest possible value a number in JavaScript can have 1.7976931348623157E+308
<u>MIN_VALUE</u>	The smallest possible value a number in JavaScript can have 5E-324
<u>NaN</u>	Equal to a value that is not a number.
<u>NEGATIVE_INFINITY</u>	A value that is less than MIN_VALUE.
<u>POSITIVE_INFINITY</u>	A value that is greater than MAX_VALUE
<u>prototype</u>	A static property of the Number object. Use the prototype property to assign new properties and methods to the Number object in the current document

### Date object

The Date object is a datatype built into the JavaScript language. Date objects are created with the **new Date( )** as shown below.

Once a Date object is created, a number of methods allow we to operate on it. Most methods simply allow we to get and set the year, month, day, hour, minute, second, and millisecond fields of the object, using either local time or UTC (universal, or GMT) time.

The ECMAScript standard requires the Date object to be able to represent any date and time, to millisecond precision, within 100 million days before or after 1/1/1970. This is a range of plus or minus 273,785 years, so the JavaScript is able to represent date and time till year 275755.

### Syntax:

Here are different variant of Date() constructor:

```
new Date( )  
new Date(milliseconds)  
new Date(datestring)  
new Date(year,month,date[,hour,minute,second,millisecond ])
```

**Note:** Paramters in the brackets are always optional

Here is the description of the parameters:

- **No Argument:** With no arguments, the Date( ) constructor creates a Date object set to the current date and time.
- **milliseconds:** When one numeric argument is passed, it is taken as the internal numeric representation of the date in milliseconds, as returned by the getTime( ) method. For

example, passing the argument 5000 creates a date that represents five seconds past midnight on 1/1/70.

- **datestring:** When one string argument is passed, it is a string representation of a date, in the format accepted by the Date.parse( ) method.
- **7 arguments:** To use the last form of constructor given above, Here is the description of each argument:
  1. **year:** Integer value representing the year. For compatibility (in order to avoid the Y2K problem), we should always specify the year in full; use 1998, rather than 98.
  2. **month:** Integer value representing the month, beginning with 0 for January to 11 for December.
  3. **date:** Integer value representing the day of the month.
  4. **hour:** Integer value representing the hour of the day (24-hour scale).
  5. **minute:** Integer value representing the minute segment of a time reading.
  6. **second:** Integer value representing the second segment of a time reading.
  7. **millisecond:** Integer value representing the millisecond segment of a time reading.

## Window Object

The window object represents an open window in a browser.

If a document contains frames (<frame> or <iframe> tags), the browser creates one window object for the HTML document, and one additional window object for each frame.

**Note:** There is no public standard that applies to the Window object, but all major browsers support it.

### Window Object Properties

Property	Description
<u>closed</u>	Returns a Boolean value indicating whether a window has been closed or not
<u>defaultStatus</u>	Sets or returns the default text in the statusbar of a window
<u>document</u>	Returns the Document object for the window ( <u>See Document object</u> )
<u>frames</u>	Returns an array of all the frames (including iframes) in the current window
<u>history</u>	Returns the History object for the window ( <u>See History object</u> )



## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: II BCA  
COURSE CODE: 18CAU403

COURSE NAME: WEB PROGRAMMING  
UNIT: II (JavaScript) BATCH: 2018-2021

<u>innerHeight</u>	Sets or returns the inner height of a window's content area
<u>innerWidth</u>	Sets or returns the inner width of a window's content area
<u>length</u>	Returns the number of frames (including iframes) in a window
<u>location</u>	Returns the Location object for the window ( <a href="#">See Location object</a> )
<u>name</u>	Sets or returns the name of a window
<u>navigator</u>	Returns the Navigator object for the window ( <a href="#">See Navigator object</a> )
<u>opener</u>	Returns a reference to the window that created the window
<u>outerHeight</u>	Sets or returns the outer height of a window, including toolbars/scrollbars
<u>outerWidth</u>	Sets or returns the outer width of a window, including toolbars/scrollbars
<u>pageXOffset</u>	Returns the pixels the current document has been scrolled (horizontally) from the upper left corner of the window
<u>pageYOffset</u>	Returns the pixels the current document has been scrolled (vertically) from the upper left corner of the window
<u>parent</u>	Returns the parent window of the current window
<u>screen</u>	Returns the Screen object for the window ( <a href="#">See Screen object</a> )
<u>screenLeft</u>	Returns the x coordinate of the window relative to the screen
<u>screenTop</u>	Returns the y coordinate of the window relative to the screen
<u>screenX</u>	Returns the x coordinate of the window relative to the screen
<u>screenY</u>	Returns the y coordinate of the window relative to the screen
<u>self</u>	Returns the current window
<u>status</u>	Sets or returns the text in the statusbar of a window
<u>top</u>	Returns the topmost browser window

**The Document Object:**

When an HTML document is loaded into a web browser, it becomes a **document object**.

The document object is the root node of the HTML document and the "owner" of all other nodes: (element nodes, text nodes, attribute nodes, and comment nodes).

The document object provides properties and methods to access all node objects, from within JavaScript.

- The document is a part of the window object and can be accessed as window.document.

**Document Object Properties and Methods**

The following properties and methods can be used on HTML documents:

Property / Method	Description
document.adoptNode(node)	Returns an adapted node from another document to this document.
document.anchors	Returns a collection of all the anchors in the document
document.applets	Returns a collection of all the applets in the document
document.baseURI	Returns the absolute base URI of a document
document.body	Returns the body element of the document
document.close()	Closes the output stream previously opened with document.open()
document.cookie	Returns all name/value pairs of cookies in the document
document.createAttribute()	Creates an attribute node
document.createComment()	Creates a Comment node with the specified text
document.createDocumentFragment()	Creates an empty DocumentFragment node
document.createElement()	Creates an Element node

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: II BCA  
COURSE CODE: 18CAU403

COURSE NAME: WEB PROGRAMMING  
UNIT: II (JavaScript) BATCH: 2018-2021

<u>document.createTextNode()</u>	Creates a Text node
<u>document.doctype</u>	Returns the Document Type Declaration associated with the document
<u>document.documentElement</u>	Returns the Document Element of the document (the HTML element)
<u>document.documentMode</u>	Returns the mode used by the browser to render the document
<u>document.documentURI</u>	Sets or returns the location of the document
<u>document.domain</u>	Returns the domain name of the server that loaded the document
<u>document.domConfig</u>	Returns the configuration used when normalizeDocument() is invoked
<u>document.forms</u>	Returns a collection of all the forms in the document
<u>document.getElementById()</u>	Returns the element that has the ID attribute with the specified value
<u>document.getElementsByName()</u>	Accesses all elements with a specified name
<u>document.getElementsByTagName()</u>	Returns a NodeList containing all elements with the specified tagname
<u>document.images</u>	Returns a collection of all the images in the document
<u>document.implementation</u>	Returns the DOMImplementation object that handles this document
<u>document.importNode()</u>	Imports a node from another document

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: II BCA  
COURSE CODE: 18CAU403

COURSE NAME: WEB PROGRAMMING  
UNIT: II (JavaScript) BATCH: 2018-2021

<u>document.inputEncoding</u>	Returns the encoding, character set, used for the document
<u>document.lastModified</u>	Returns the date and time the document was last modified
<u>document.links</u>	Returns a collection of all the links in the document
<u>document.normalize()</u>	Removes empty Text nodes, and joins adjacent nodes
<u>document.normalizeDocument()</u>	Removes empty Text nodes, and joins adjacent nodes
<u>document.open()</u>	Opens an HTML output stream to collect output from document.write()
<u>document.readyState</u>	Returns the (loading) status of the document
<u>document.referrer</u>	Returns the URL of the document that loaded the current document
<u>document.renameNode()</u>	Renames the specified node

**PART-B**

**(Each Question carries 2 Marks)**

1. What is JavaScript?
2. What is EMCA Script?
3. Define <noscript> tag.
4. Differentiate Java and Javascript.
5. How javascript is embedded into HTML?
6. Define Comments in javascript.

**PART-C**

**(Each Question carries 6 Marks)**

1. Explain how to create Table border with caption with example.
2. Explain about Row Span and Col Span Attributes with example.
3. Explain Frame targeting attribute with example.
4. Explain how to reset a form in HTML with example.
5. Explain how to create Radio Button and Check Boxes in HTML with example.
6. Explain about Action attribute of a form in HTML with example.
7. Explain how to set the width of table cells and alignment with example.
8. Explain Form elements in HTML with example.
9. Explain table layouts with example.
10. Explain form elements with example.

## UNIT - II

S.No	Question	Option1	Option2	Option3
1	JavaScript is a _____ language	Object-Oriented	High-level	Assembly language
2	JavaScript can be written _____	directly into JS file and included into HTML	directly on the server page	directly into HTML pages
3	What is the return type of standard JavaScript objects?	xml	object	DOM
4	JavaScript code between a pair of “script” tags are called _____	Non-inline	External	Referenced
5	When does JavaScript code appear inline within an HTML file?	Between the “script” tag	Outside the “script” tag	Outside the “Javascript” tag
6	The JavaScript’s syntax calling a function or method is called	Primary expression	Functional expression	Invocation expression
7	What kind of an expression is “new Point(2,3)”?	Primary Expression	Object Creation Expression	Constructor Calling Expression
8	Which of the operator is used to test if a particular property exists or not?	in	exist	within
9	The script tag must be placed in	head	head and body	title and head
10	A JavaScript program developed on a Unix Machine	will throw errors and exceptions	must be restricted to a Unix Machine only	will work perfectly well on a Windows Machine
11	JavaScript is ideal to	make computations in HTML simpler	minimize storage requirements on	increase the download time for the client
12	JavaScript Code can be called by using	RMI	Triggering Event	Preprocessor
13	A hexadecimal literal begins with	0	0x	0X
14	When there is an indefinite or an infinity value during an arithmetic value computation, javascript	Prints an exception error	Prints an overflow error	Displays “Infinity”
15	The snippet that has to be used to check if “a” is not equal to “null” is	if(a!=null)	if (!a)	if(a!=null)

16	The statement a===b refers to	Both a and b are equal in value, type and reference address	Both a and b are equal in value	Both a and b are equal in value and type
17	A conditional expression is also called a _____ A statement block is a _____	Alternate to if-else conditional block	Immediate if block that contains a single statement	If-then-else statement block that contains no statements
18				
19	When an empty statement is encountered, a JavaScript interpreter	Ignores the statement	Prompts to complete the statement	Throws an error
20	The enumeration order becomes implementation dependent and non-interoperable if :	If the object inherits enumerable properties	The object does not have the properties present in the integer array indices	The delete keyword is never used
21	What are the three important manipulations done in a for loop on a loop variable? One of the special feature of an interpreter in reference with the for loop is that	Updation, Incrementation, Initialization Before each iteration, the interpreter evaluates the variable expression and assigns the name of the property	Initialization, Testing, Updation The iterations can be infinite when an interpreter is used	Testing, Updation, Testing The body of the loop is executed only once
22				
23	Among the keywords below, which one is not a statement?	debugger	with	if
24	The unordered collection of properties, each of which has a name and a value is called The object has three object attributes namely	String Class, parameters, object's extensible flag	Object Prototype, class, objects' parameters	Serialized Object Prototype, class, object's extensible flag
25				
26	book[datatype]=assignment_value; In this syntax, the datatype within the square brackets must be _____	An integer	A String	An object

27	To determine whether one object is the prototype of another object, one should use the	isPrototypeOf() method	equals() method	=== operator
28	function f() {}; This prototype represents a	Function f	A custom constructor	Prototype of a function
29	What will happen if reverse() and join() methods are used simultaneously ?	Reverses and stores in the same array	Reverses and concatenates the elements of the array	Reverses the array
30	36. The function definitions in JavaScript begins with	Identifier and Parentheses	Return type and Identifier	Return type, Function keyword, Identifier and Parentheses
31	A function with no return value is called	Procedures	Method	Static function
32	Do functions in JavaScript necessarily return a value ?	It is mandatory	Not necessary	Few functions return values by default
33	Which of the following are global functions that are not part of core JavaScript?	spawn(f);	trim();	exit();
34	Which of the following reads the textual contents of a URL and returns as a string?	spawn(f);	load(filename,...);	readFile(file);
35	Which is a more formal way of importing packages and classes as JavaScript objects?	import(jav util.*);	importClass(jav util.*);	import.Class(jav util.*);
36	The JavaScript classes can be instantiated using _____ operator?	create	new	instantiate
37	The new Java arrays can be created into a JavaScript programs using which of the following classes?	jav Array	jav lang.*	jav lang.Array
38	Which of the following is the descendant operator?	.	...	*
39	Which function is a synonym for on()?	addListener()	listeners()	once()
40	What is the method used to pause “data” events?	s.pause();	s.stop();	s.halt();



41	Why the “this” keyword forbidden in JavaScript?	Functions should access the global objects	Functions should not access the global objects	Highly memory consuming
42	The maximum number of global symbols a module can define is	2	1	3
43	The scope of a function is also called as	The function’s scope	Private function	public function
44	What does /[^(]* regular expression indicate ?	Match one or more characters that are not open parenthesis	Match zero or more characters that are open parenthesis	Match zero or more characters that are not open parenthesis
45	What does the sub expression /java(script)?/ result in ?	It matches “java” followed by the optional “script”	It matches “java” followed by any number of “scripts”	It matches “java” followed by a minimum of one “script”
46	What is the most essential purpose of parentheses in regular expressions?	Define pattern matching techniques	Define sub patterns within the complete pattern	Define portion of strings in the regular expression
47	The basic difference between JavaScript and Java is _____	There is no difference	Variables are specific	Functions are considered as fields
48	When a class B can extend another class A, we say that	A is the superclass and B is the subclass	B is the super class and A is the subclass	Both A and B are the super class
49	What kind of scoping does JavaScript use?	Literal	Segmental	Lexical
50	When are the keyboard events fired?	When user manually calls the button	When user clicks a key	When the user calls the modifier
51	How does a user generate multiple key down events?	Repeating the same process	Pressing multiple keys	Pressing the key double times
52	66. Which property is used to specify the key type when pressed?	keyCode	keyType	keyName
53	A JavaScript program can traverse and manipulate document content through	Element Object	Class	Document Object
54	Which character in JavaScript code will be interpreted as XML markup?	!	>	&

55	What is the code for getting the current time?	<code>var now = new Date();</code>	<code>now = new Date();</code>	<code>var now = Date();</code>
56	What will be done if more than one page requires a file of JavaScript code?	Downloads that many times	Retrieves from the browser cache	Must be re executed
57	What is the default value of the type attribute?	text/css	text	text/javascript
58	Which is a fast C++ based JavaScript interpreter?	Node	Sockets	Processors
59	What is the command to run the node programs?	<code>node(program.js)</code>	<code>program.js</code>	<code>node program.js</code>
60	What is the purpose of the event handlers in the JavaScript?	Adds inner HTML page to the code	Performs handling of exceptions and occurrences	No event handlers available

<b>Option4</b>	<b>Answer</b>
Object-Based	Object-Based
directly into JS file and without included into HTML html	directly into JS file and included into HTML object
Inline	Inline
Outside the “javascript” tag	Between the “script” tag
Property Access Expression	Invocation expression
Invocation Expression between	Object Creation Expression in
title	head and body
will be displayed as a JavaScript text on the browser	will work perfectly well on a Windows Machine
decrease the download time for the client	minimize storage requirements on the web server
Function/Method	Function/Method
0x or 0X	0x or 0X
Prints the value	Displays “Infinity”
if(a!==null)	if(a!==null)

There is no such statement

Both a and b are equal in value and type

switch statement

Immediate if

block that combines multiple statements into a single compound statement

block that combines multiple statements into a single compound statement

Throws an exception

Ignores the statement

Object.defineProperty() is not used

If the object inherits enumerable properties

Initialization, Testing, Incrementation

Initialization, Testing, Updation

The body of the loop is not executed

Before each iteration, the interpreter evaluates the variable expression and assigns the name of the property

use strict

use strict

Class

Object

Native object, Classes and Interfaces and Object's extensible flag

Prototype, class, object's extensible flag

A Character

A String

prototype( )  
method

isPrototypeOf()  
method

Not valid

A custom constructor

Reverses and  
stores in the another  
array

Reverses and stores in  
the same array

Identifier and  
Return type

Identifier and  
Parentheses

Dynamic function

Procedures

Few functions  
may not return  
values

Few functions return  
values by default

load();

spawn(f);

readUrl(url);

readUrl(url);

Class.import(jav  
util.\*);

importClass(jav  
util.\*);

create.new

new

jav  
lang.reflect.Array

jav lang.reflect.Array

@

...

add()

addListener()

s.wait();

s.pause();

Very inefficient to use

4

Functions should not access the global objects

3

Module function

Module function

Match one or more characters that are open parenthesis

Match zero or more characters that are not open parenthesis

It matches “java” followed by without “script”

It matches “java” followed by the optional “script”

Define the precedence

Define sub patterns within the complete pattern

functions are values, and there is no hard distinction between methods and fields

functions are values, and there is no hard distinction between methods and fields

Both A and B are the subclass

A is the supe rclass and B is the subclass

Sequential

Lexical

When the user moves the mouse

When user clicks a key

Pressing the key longer than usual  
keyProperty

Pressing the key longer than usual  
keyCode

Element &  
Document Object

Element &  
Document Object

.

&

`var now = new  
Date(current);`  
Must be in one  
page

xml

Closures

node.program.js

Allows JavaScript  
code to alter the  
behaviour of  
windows

`var now = new  
Date();`  
Retrieves from the  
browser cache

text/javascript

Node

node program.js

Allows JavaScript  
code to alter the  
behaviour of windows

**UNIT-III**

**SYLLABUS**

**JavaScript:** Form and form elements – Scripting frames and multiple windows – Event object – Functions and custom objects.

**Working with Form and Form Elements:**

**Form Object:**

The Form object represents an HTML form.

For each <form> tag in an HTML document, a Form object is created.

Forms are used to collect user input, and contain input elements like text fields, checkboxes, radio-buttons, submit buttons and more. A form can also contain select menus, textarea, fieldset, legend, and label elements. Forms are used to pass data to a server.

**Form Object Collections**

Collection	Description
<u>elements</u>	Returns a collection of all elements in a form

**Form Object Properties**

Property	Description
<u>acceptCharset</u>	Sets or returns the value of the accept-charset attribute in a form
<u>action</u>	Sets or returns the value of the action attribute in a form
<u>enctype</u>	Sets or returns the value of the enctype attribute in a form



<u>length</u>	Returns the number of elements in a form
<u>method</u>	Sets or returns the value of the method attribute in a form
<u>name</u>	Sets or returns the value of the name attribute in a form
<u>target</u>	Sets or returns the value of the target attribute in a form

**Form Object Methods**

<b>Method</b>	<b>Description</b>
reset()	Resets a form
submit()	Submits a form

**Form action Property:**

**Example**

Return where to send the form-data when a form is submitted:

```
var x = document.getElementById("myForm").action;
```

**The result of *x* will be:**

form\_action.asp

**Definition and Usage:**

The action property sets or returns the value of the action attribute in a form.

The action attribute specifies where to send the form-data when a form is submitted.

**Syntax:**

Return the action property:

*formObject.action*

Set the action property:

*formObject.action=URL*

Value	Description
URL	<p>Specifies where to send the form-data when the form is submitted.</p> <p>Possible values:</p> <ul style="list-style-type: none"><li>• An absolute URL - points to another web site (like action="http://www.example.com/example.htm")</li><li>• A relative URL - points to a file within a web site (like action="example.htm")</li></ul>

**Return Value**

Type	Description
String	The value of the action attribute in the form element

**Form enctype Property**

**Example**

Return how form-data should be encoded before sending it to the server:

```
var x = document.getElementById("myForm").enctype;
```

**The result of *x* will be:**

application/x-www-form-urlencoded

**//Source Code //**

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

<form id="myForm" action="demo_post_encype.asp" method="post">
First name: <input type="text" name="fname" value="Donald"><br>
Last name: <input type="text" name="lname" value="Duck"><br>
<input type="submit" value="Submit">
</form>

<p>Click the "Try it" button to return the value of the enctype attribute in the form element.</p>

<p id="demo"></p>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction()
{
var x = document.getElementById("myForm").enctype;
document.getElementById("demo").innerHTML=x;
}
</script>

</body>
</html>
```

### Form reset() Method

#### Example:

#### Reset a form:

```
document.getElementById("myForm").reset();
```

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

<p>Enter some text in the fields below, then press the "Reset form" button to reset the form.</p>

```
<form id="myForm">  
First name: <input type="text" name="fname"><br>  
Last name: <input type="text" name="lname"><br><br>  
<input type="button" onclick="myFunction()" value="Reset form">  
</form>
```

```
<script>  
function myFunction()  
{  
document.getElementById("myForm").reset();  
}  
</script>
```

```
</body>  
</html>
```

### Definition and Usage:

The reset() method resets the values of all elements in a form (same as clicking the Reset button).

- Use the submit() method to submit the form.

### Browser Support

The reset() method is supported in all major browsers.

### Syntax

*formObject*.reset()

### Parameters

None.

### Return Value

No return value.

## Form submit() Method

### Example

#### Submit a form:

```
document.getElementById("myForm").submit();
```

### Definition and Usage

The submit() method submits the form (same as clicking the Submit button).

- Use the reset() method to reset the form.

The submit() method is supported in all major browsers.

### Syntax

```
formObject.submit()
```

### Parameters

None.

### Return Value

No return value.

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

```
<p>Enter some text in the fields below, then press the "Submit form" button to submit the  
form.</p>
```

```
<form id="myForm" action="form_action.asp">  
First name: <input type="text" name="fname"><br>  
Last name: <input type="text" name="lname"><br><br>  
<input type="button" onclick="myFunction()" value="Submit form">
```

```
</form>

<script>
function myFunction()
{
document.getElementById("myForm").submit();
}
</script>

</body>
</html>
```

### **Location Object:**

The location object contains information about the current URL.

The location object is part of the window object and is accessed through the window.location property.

- There is no public standard that applies to the location object, but all major browsers support it.

### **Location Object Properties**

Property	Description
<u>hash</u>	Sets or returns the anchor portion of a URL
<u>host</u>	Sets or returns the hostname and port of a URL
<u>hostname</u>	Sets or returns the hostname of a URL
<u>href</u>	Sets or returns the entire URL

**CLASS: II BCA**  
**COURSE CODE: 18CAU403**

**COURSE NAME: WEB PROGRAMMING**  
**UNIT: III (JavaScript)      BATCH: 2018-2021**

<u>pathname</u>	Sets or returns the path name of a URL
<u>port</u>	Sets or returns the port number the server uses for a URL
<u>protocol</u>	Sets or returns the protocol of a URL
<u>search</u>	Sets or returns the query portion of a URL

### Location Object Methods

Method	Description
<u>assign()</u>	Loads a new document
<u>reload()</u>	Reloads the current document
<u>replace()</u>	Replaces the current document with a new one

### Location href Property

#### Example

Return the entire URL (of the current page):

```
var x = location.href;
```

**The result of *x* will be:**

[http://www.w3schools.com/jsref/prop\\_loc\\_href.asp](http://www.w3schools.com/jsref/prop_loc_href.asp)

### Definition and Usage

The href property sets or returns the entire URL of the current page.

## Browser Support

The href property is supported in all major browsers.

## Syntax

Return the href property:

```
location.href
```

Set the href property:

```
location.href=URL
```

HTML DOM allows JavaScript to react to HTML events.

## Example

**Mouse Over Me**

**Click Me**

**Click Me Again**

## Reacting to Events

A JavaScript can be executed when an event occurs, like when a user clicks on an HTML element. To execute code when a user clicks on an element, add JavaScript code to an HTML event attribute:

```
onclick=JavaScript
```

## Examples of HTML events:

- When a user clicks the mouse
- When a web page has loaded
- When an image has been loaded
- When the mouse moves over an element
- When an input field is changed
- When an HTML form is submitted



- When a user strokes a key

In this example, the content of the <h1> element is changed when a user clicks on it:

### Example

```
<!DOCTYPE html>
<html>
<body>
<h1 onclick="this.innerHTML='Oops!'">Click on this text!</h1>
</body>
</html>
```

In this example, a function is called from the event handler:

### Example

```
<!DOCTYPE html>
<html>
<head>
<script>
function changetext(id)
{
id.innerHTML="Oops!";
}
</script>
</head>
<body>
<h1 onclick="changetext(this)">Click on this text!</h1>
</body>
</html>
```

## Event Objects:

### HTML Event Attributes

To assign events to HTML elements we can use event attributes.

### Example

Assign an onclick event to a button element:

```
<button onclick="displayDate()">Try it</button>
```

In the example above, a function named *displayDate* will be executed when the button is clicked.

### Assign Events Using the HTML DOM

The HTML DOM allows us to assign events to HTML elements using JavaScript:

#### Example

Assign an onclick event to a button element:

```
<script>  
document.getElementById("myBtn").onclick=function(){displayDate()};  
</script>
```

In the example above, a function named *displayDate* is assigned to an HTML element with the id="myBtn".

The function will be executed when the button is clicked.

### The onload and onunload Events

The onload and onunload events are triggered when the user enters or leaves the page.

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.

The onload and onunload events can be used to deal with cookies.

#### Example

```
<body onload="checkCookies()">
```

### The onchange Event

The onchange event are often used in combination with validation of input fields.

Below is an example of how to use the onchange. The upperCase() function will be called when a user changes the content of an input field.

### Example

```
<input type="text" id="fname" onchange="upperCase()">
```

### The onmouseover and onmouseout Events

The onmouseover and onmouseout events can be used to trigger a function when the user mouses over, or out of, an HTML element.

### Example

A simple onmouseover-onmouseout example:

#### Mouse Over Me

### The onmousedown, onmouseup and onclick Events

The onmousedown, onmouseup, and onclick events are all parts of a mouse-click. First when a mouse-button is clicked, the onmousedown event is triggered, then, when the mouse-button is released, the onmouseup event is triggered, finally, when the mouse-click is completed, the onclick event is triggered.

### Example

A simple onmousedown-onmouseup example:

#### Click Me

### Examples

#### onmousedown and onmouseup

Change an image when a user holds down the mouse button.

#### onload

Display an alert box when the page has finished loading.

**onfocus**

Change the background-color of an input field when it gets focus.

**Mouse Events**

Change the color of an element when the cursor moves over it.

**Window Events:**

**HTML DOM Events**

HTML DOM events allow JavaScript to register different event handlers on elements in an HTML document.

Events are normally used in combination with functions, and the function will not be executed before the event occurs (such as when a user clicks a button).

- The event model was standardized by the W3C in DOM Level 2.

**HTML DOM Events**

**DOM:** Indicates in which DOM Level the property was introduced.

**Mouse Events**

Property	Description
<u>onclick</u>	The event occurs when the user clicks on an element
<u>ondblclick</u>	The event occurs when the user double-clicks on an element
<u>onmousedown</u>	The event occurs when a user presses a mouse button over an element
<u>onmousemove</u>	The event occurs when the pointer is moving while it is over an element
<u>onmouseover</u>	The event occurs when the pointer is moved onto an element

<u>onmouseout</u>	The event occurs when a user moves the mouse pointer out of an element
<u>onmouseup</u>	The event occurs when a user releases a mouse button over an element

**Keyboard Events**

Attribute	Description
<u>onkeydown</u>	The event occurs when the user is pressing a key
<u>onkeypress</u>	The event occurs when the user presses a key
<u>onkeyup</u>	The event occurs when the user releases a key

**Scripting frames and multiple windows:****Frame/Object Events**

Attribute	Description
<u>onabort</u>	The event occurs when an image is stopped from loading before completely loaded (for <object>)
<u>onerror</u>	The event occurs when an image does not load properly (for <object>, <body> and <frameset>)
<u>onload</u>	The event occurs when a document, frameset, or <object> has been loaded
<u>onresize</u>	The event occurs when a document view is resized

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: II BCA  
COURSE CODE: 18CAU403

COURSE NAME: WEB PROGRAMMING  
UNIT: III (JavaScript) BATCH: 2018-2021

onscroll	The event occurs when a document view is scrolled
<u>onunload</u>	The event occurs once a page has unloaded (for <body> and <frameset>)

### Form Events

Attribute	Description
<u>onblur</u>	The event occurs when a form element loses focus
<u>onchange</u>	The event occurs when the content of a form element, the selection, or the checked state have changed (for <input>, <select>, and <textarea>)
<u>onfocus</u>	The event occurs when an element gets focus (for <label>, <input>, <select>, <textarea>, and <button>)
onreset	The event occurs when a form is reset
<u>onselect</u>	The event occurs when a user selects some text (for <input> and <textarea>)
onsubmit	The event occurs when a form is submitted

### Event Object

#### Constants

Constant	Description
CAPTURING_PHASE	The current event phase is the capture phase (3)

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: II BCA  
COURSE CODE: 18CAU403

COURSE NAME: WEB PROGRAMMING  
UNIT: III (JavaScript) BATCH: 2018-2021

AT_TARGET	The current event is in the target phase, i.e. it is being evaluated at the event target (1)
BUBBLING_PHASE	The current event phase is the bubbling phase (2)

### Properties

Property	Description
<u>bubbles</u>	Returns whether or not an event is a bubbling event
<u>cancelable</u>	Returns whether or not an event can have its default action prevented
<u>currentTarget</u>	Returns the element whose event listeners triggered the event
eventPhase	Returns which phase of the event flow is currently being evaluated
<u>target</u>	Returns the element that triggered the event
<u>timeStamp</u>	Returns the time (in milliseconds relative to the epoch) at which the event was created
<u>type</u>	Returns the name of the event

### Methods

Method	Description
initEvent()	Specifies the event type, whether or not the event can bubble, whether or not the event's default action can be prevented

preventDefault()	To cancel the event if it is cancelable, meaning that any default action normally taken by the implementation as a result of the event will not occur
stopPropagation()	To prevent further propagation of an event during event flow

**EventTarget Object**

## Methods

Method	Description
addEventListener()	Allows the registration of event listeners on the event target (IE8 = attachEvent())
dispatchEvent()	Allows to send the event to the subscribed event listeners (IE8 = fireEvent())
removeEventListener()	Allows the removal of event listeners on the event target (IE8 = detachEvent())

**EventListener Object**

## Methods

Method	Description
handleEvent()	Called whenever an event occurs of the event type for which the EventListener interface was registered



**MouseEvent/KeyboardEvent Object****Properties**

<b>Property</b>	<b>Description</b>
<u>altKey</u>	Returns whether or not the "ALT" key was pressed when an event was triggered
<u>button</u>	Returns which mouse button was clicked when an event was triggered
<u>clientX</u>	Returns the horizontal coordinate of the mouse pointer, relative to the current window, when an event was triggered
<u>clientY</u>	Returns the vertical coordinate of the mouse pointer, relative to the current window, when an event was triggered
<u>ctrlKey</u>	Returns whether or not the "CTRL" key was pressed when an event was triggered
keyIdentifier	Returns the identifier of a key
keyLocation	Returns the location of the key on the device
<u>metaKey</u>	Returns whether or not the "meta" key was pressed when an event was triggered
<u>relatedTarget</u>	Returns the element related to the element that triggered the event
<u>shiftKey</u>	Returns whether or not the "SHIFT" key was pressed when an event was triggered

**Methods**

Method	Description
initMouseEvent()	Initializes the value of a MouseEvent object
initKeyboardEvent()	Initializes the value of a KeyboardEvent object

**Functions and Custom Objects:**

A function is nothing but a group or block of statements doing a specific task.

**Syntax:**

```
function name()
{
    // set of statements that will be executed
}
```

A function has to be defined in the above syntax.

The name "function" followed by the name we choose for the function and then open and close brackets. The statements that will do the specific operation are then group together under this name using braces.

A function may or may not return a value. A function may or may not have parameters value.

**Invoking a function:**

The statements inside the function will not be executed automatically. We have to invoke or call the function to execute the statements. Just calling the name of the function will invoke the function. i.e. if we write a function with the name "test" calling it as "test();" will invoke the function.

**Example Code:**

```
<script language="javascript">
function test()
```

```
{  
  document.write(" ---- This is a test function-----");  
}  
  
test();  
</script>
```

**Result:**

---- This is a test function ----

**Syntax:**

```
function name(parameter 1,parameter 2,...)  
{  
  // set of statements that will be executed  
}
```

**Passing Parameters to Functions:**

In many cases we pass parameters or arguments to functions, these arguments will be used inside the function for required calculations. For an example we will use two numbers to add, subtract using function.

Here we write separate function for each operations add, subtract.

**Example Code:**

```
<script language="javascript">  
  
function add(number1, number2)  
{  
  var c = number1+number2;  
  document.write(" ---- This added value is ---- "+c;  
}  
  
function sub(number1, number2)  
{  
  var c = number1-number2;  
  document.write(" ---- This subtracted value is ---- "+c;
```

```
}  
  
var a = 7;  
var b = 3;  
add(a,b);  
sub(a,b);  
</script>
```

**Result:**

---- This added value is ---- 10  
---- This subtracted value is -----4

Here we can clearly see that the two functions were invoked as "add(a,b);" and "sub(a,b);" where a and b are defined variables. We can even call the function directly with the variables as say "add(9,1);".

A function can be invoked any number of times with any proper value.

**Returning Values:**

Syntax:

```
function name(parameter 1,parameter 2,...)  
{  
    // set of statements that will be executed  
    return thevalue;  
}
```

A function can also **return a value** after doing the specified operations. To explain, we would consider a requirement where we have to calculate  $x^2/2$ .

We will calculate  $x^2$  in a function and return the result of  $x^2$ . After getting the return value, we will divide it by 2.

**Example Code:**

```
<script language="javascript">
```

```
function square(number1)  
{
```

```
var c = number1*number1;  
// Now we will return the result  
return c;  
}  
  
var x = 4;  
// Here we invoke the function and capture the result  
var des = square(x);  
var res = des/2;  
document.write(" The result - "+res);  
</script>
```

**Result:**

The result - 8

**Predefined Functions:**

Javascript is an object oriented programming language. It supports the concept of objects in the form of attributes. If an object attribute consists of function, then it is a method of that object, or if an object attribute consists of values, then it is a property of that object. For example,

```
var status=document.readyState;
```

readyState is a property of the document object which can contain values such as "uninitialized", "loading", "interactive", "complete" whereas,

```
document.write("Hello World");
```

write() is a method of the document object that writes the content "Hello World" on the web page. There are several Javascript built-in objects such as,

- Number
- String
- RegExp
- Array

- Math
- Date
- Boolean

Each of the above objects hold several built-in functions to perform object related functionality. Apart from these methods, Javascript provides few predefined functions which do not stick to a particular object type but are global. These global built-in functions are explained below with examples.

**PART-B**

**(Each Question carries 2 Marks)**

1. Define functions in Javascript.
2. What is event?
3. What is event handler?
4. What is object in Javascript?
5. Define Array.
6. What are the properties of Number object?
7. Write the Syntax for creating a Date Object.

**PART-C**

**(Each Question carries 6 Marks)**

1. Explain Arithmetic and Logical operators in JavaScript with example.
2. Explain about If . . . Else Statement with example.
3. Explain Data type conversions in JavaScript with example.
4. Explain about do . . . While Statement in JavaScript with example.
5. Explain Looping Statements in JavaScript with example.
6. Explain Window Methods in JavaScript with example.
7. Explain Data types in JavaScript with example.
8. Explain Data type conversions in JavaScript with example.
9. Explain conditional statements in JavaScript with example.
10. Explain functions in Java script with example.

### UNIT-III

S.No	Question	Option1	Option2	Option3
1	Why so JavaScript and Java have similar name?	JavaScript is a stripped-down version of Java	JavaScript's syntax is loosely based on Java's	They both originated on the island of Java
2	_____ JavaScript is also called client-side JavaScript.	Microsoft	Navigator	LiveWire
3	What are variables used for in JavaScript Programs?	Storing numbers, dates, or other values	Varying randomly	Causing high-school algebra flashbacks
4	_____ JavaScript is also called server-side JavaScript.	Microsoft	Navigator	LiveWire
5	Which of the following are capabilities of functions in JavaScript?	Return a value	Accept parameters and Return a value	Accept parameters
6	_____ tag is an extension to HTML that can enclose any number of JavaScript statements.	<SCRIPT>	<BODY>	<HEAD>
7	What is the correct JavaScript syntax to write "Hello World"?	System.out.println ("Hello World")	println ("Hello World")	document.write("Hello World")
8	Inside which HTML element do we put the JavaScript?	<js>	<scripting>	<script>
9	JavaScript entities start with _____ and end with _____.	Semicolon, colon	Semicolon, Ampersand	Ampersand, colon
10	Choose the server-side JavaScript object?	FileUpLoad	Function	File
11	Choose the client-side JavaScript object?	Database	Cursor	Client
12	Which of the following is not considered a JavaScript operator?	new	this	delete
13	JavaScript is interpreted by _____	Client	Server	Object
14	Using _____ statement is how you test for a specific condition.	Select	If	Switch
15	How to create a Date object in JavaScript?	dateObjectName = new Date([parameters] )	dateObjectName.new Date([parameters] )	dateObjectName := new Date([parameters])



16	The _____ method of an Array object adds and/or removes elements from an array.	Reverse	Shift	Slice
17	To set up the window to capture all Click events, we use which of the following statement?	window.captureEvents(Event.CLICK);	window.handleEvents(Event.CLICK);	window.routeEvents(Event.CLICK);
18	Which tag(s) can handle mouse events in Netscape?	<IMG>	<A>	 
19	_____ is the tainted property of a window object.	Pathname	Protocol	Defaultstatus
20	What is mean by "this" keyword in javascript?	It refers current object	It refers previous object	It is variable which contains value
21	Choose the client-side JavaScript object:	Database	Cursor	Client
22	Which best explains getSelection()?	Returns the VALUE of a selected OPTION	Returns document.URL of the window in focus.	Returns the value of cursor-selected text
23	Scripting language are	High Level Programming language	Assembly Level programming language	Machine level programming language
24	The syntax of close method for document object is _	Close(doC.	Close(object)	Close(val)
25	The syntax of capture events method for document object is	captureEvents( )	captureEvents(args eventType)	captureEvents(eventType)
26	The syntax of a blur method in a button object is	Blur( )	Blur(contrast)	Blur(value)
27	The JavaScript exception is available to the Java code as an instance of	netscape.javascript.JSObject	netscape.javascript.JSException	netscape.plugin.JSException
28	A _____ object is a reference to one of the classes in a Java package, such as netscape.javascript .	JavaArray	JavaClass	JavaObject
29	_____ is a wrapped Java array, accessed from within JavaScript code.	JavaArray	JavaClass	JavaObject
30	When a JavaScript object is sent to Java, the runtime engine creates a Java wrapper of type _____	ScriptObject	JSObject	JavaObject

_____method evaluates a string of JavaScript code in the context of the specified object	Eval	ParseInt	parseFloat
JavaScript is _____ language.	Programming	Application	machine
JavaScript is _____ Side Scripting Language.	ISP	Browser	Server
JavaScript is designed for following purpose -	To Perform Server Side Scripting Operation	To add interactivity to HTML Pages.	To Execute Query Related to DB on Server
JavaScript Code is written inside file having extension _____	.javascript	.jsc	.jvs
_____ attribute is used to specify the character encoding used in an external script file.	chartype	type	charset
Which built-in method combines the text of two strings and returns a new string?	append( )	concat( )	attach( )
Which of the following code creates an object?	var book = Object( );	var book = new Object( );	var book = new OBJECT( );
Which of the following function of Array object returns a string representing the array and its elements?	toSource( )	sort( )	splice( )
Which built-in method returns the characters in a string beginning at the specified location?	substr( )	getSubstring( )	slice( )
Which of the following function of String object returns the character at the specified index?	charAt( )	charCodeAt( )	concat( )
Which of the following code creates an object?	var book = Object( );	var book = new Object( );	var book = new OBJECT( );
Which built-in method calls a function for each element in the array?	while( )	loop( )	forEach( )
Which built-in method sorts the elements of an array?	change(order)	order( )	sort( )

45	The type of a variable that is volatile is	Volatile variable	Mutable variable	Immutable variable
46	Which of the following is not considered as an error in JavaScript?	Syntax error	Missing of semicolons	Division by zero
47	The escape sequence ‘\f’ stands for	Floating numbers	Representation of functions that returns a value	\f is not present in JavaScript
48	A function definition expression can be called	Function prototype	Function literal	Function definition
49	Which of the operator is used to test if a particular property exists or not?	in	exist	within
50	Among the following, which one is a ternary operator?		:	?:
51	The “var” and “function” are	Keywords	Declaration statements	Datatypes
52	What does the subexpression /java(script)?/ result in ?	It matches “java” followed by the optional “script”	It matches “java” followed by any number of “script”	It matches “java” followed by a minimum of one “script”
53	Which function among the following lets to register a function to be invoked once?	setTimeout( )	setTotaltime( )	setInterval( )
54	Which property is used to obtain browser vendor and version information?	modal	version	browser
55	The setTimeout( ) belongs to which object?	Element	Window	Location
56	To which object does the location property belong?	Window	Position	Element
57	Which method receives the return value of setTimeout( ) to cancel future invocations?	clearTimeout( )	clearInterval( )	clearSchedule( )
58	The Cookie manipulation is done using which property?	cookie	manipulation	manipulate
59	Which of the following explains Cookies nature?	Non Volatile	Volatile	Intransient
60	Which of the following is a boolean cookie attribute?	bool	secure	lookup

**Option4**

Both are  
developed by the  
same person

Native

Varying  
sequentially

Native

does not return a  
value

<TITLE>

response.write("H  
ello World")

<javascript>

Ampersand,  
semicolon

Date

FileUpload

typeof

Class

For

dateObjectName  
Date([parameters]  
)

**Answer**

JavaScript's syntax  
is loosely based on  
Java's

Navigator

Storing numbers,  
dates, or other  
values

LiveWire

Accept parameters

<SCRIPT>

document.write("H  
ello World")

<script>

Ampersand,  
semicolon

File

FileUpload

this

Client

If

dateObjectName =  
new  
Date([parameters])

Splice

window.raiseEvents(Event.CLICK );

<P>

Host

It refers the previous value

FileUpload

Returns the VALUE of a checked radio input.

Low Level Programming Language

Close( )

captureEvents(eventVal)

Blur(depth)

netscape.plugin.Exception

JavaPackage

JavaPackage

Object

Splice

window.captureEvents(Event.CLICK);

<A>

Defaultstatus

It refers current object

FileUpload

Returns the value of cursor-selected text

High Level Programming language

Close( )

captureEvents(eventType)

Blur( )

netscape.javascript.JSException

JavaClass

JavaArray

JSObject

Efloat

Eval

Scripting

Scripting

IIS

Browser

To Style HTML  
Pages

To add interactivity  
to HTML Pages.

.js

.js

character

charset

join( )

concat( )

var book = new  
Book( );

var book = new  
Object( );

toString( )

toSource( )

substring( )

substr( )

indexOf( )

charAt( )

var book = new  
Book( )

var book = new  
Object( );

for( )

forEach( )

arrange( )

sort( )

|                      |   |
|----------------------|---|
| Dynamic variable     | Mutable variable                                    |
| Array out of bound   | Division by zero                                    |
| Form feed            | Form feed   |
| Function declaration | Function literal                                    |
| exists               | in  |
|                      | ?:  |
| Prototypes           | Declaration statements                              |
| Not Applicable       | It matches “java” followed by the optional “script” |
| setTime( )           | setTimeout( )                                       |
| navigator            | navigator   |
| control              | Window  |
| Location             | Window  |
| cancelTimeOut( )     | clearTimeout( )                                     |
| cookieManipulate     | cookie  |
| Transient            | Transient   |
| domain               | secure  |

**UNIT-IV**

**SYLLABUS**

**ASP:** Client side scripting vs. Server side scripting- Variables & Constants- Procedures – Forms – Cookies – Application - #include – Global.asa - Functions-ASP object model: Response-Request- Application- Session – Server – Error – Array

**Active Server Pages (ASP):**

The web server is not only responsible for processing HTML tags but also the ASP scripts. The ASP engine, installed on the web server, is responsible for processing ASP scripts. Because the browser can only display HTML compatible text, it is the responsibility of ASP engine to process the ASP scripts and send the results back as HTML text to the client. In other words, the Web server sends HTML text, part of which may be the result of the ASP scripts.

How the Web server can determine whether to call ASP engine or not. Recall from your HTML experience that HTML files have the .html or .htm extension. Similarly, .asp extension is used for ASP scripts. An .asp file extension indicates to the Web server to call the ASP engine. HTML tags inside an ASP file are processed by the ASP engine; however, an html file (meaning a file that ends with .htm or .html) containing ASP code won't run the ASP code!

Once the Web server sends the results to the client, it is the client's browser responsibility to display HTML text. Again, the client's browser only receives HTML text. This can be confirmed, for an asp file, by viewing the source code in the browser window. Note server-side scripts are processed before a page is downloaded on a browser, and client-side scripts are processed after a page has been downloaded. When we are writing ASP scripts, we are writing pages for server-side processing. Table 1 summarizes the differences between client and server side scripting.

- **Server-side Scripting**

A web server is responsible for processing **ASP scripts** and returning output to the client.

- The server-side scripting is done **before** a page is downloaded on the browser.
- ASP scripts is a server-side technology.



**Advantages:**

- ASP code remains on the server. Because ASP code does not leave the server, the user does not learn about how the page is built. This provides a measure of security of your ASP code and other files used in conjunction with your ASP code.
- ASP is a stable and matured technology. Microsoft, third-party vendors, and developers, are some of the players providing support (i.e., tools, components, scripts, discussion boards, etc.) for the ASP technology.
- An ASP file can use databases or other files available to the web server to produce the desired output.

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

<%
response.write("My first ASP script!")
%>
</body>
</html>
```

Result:

My first ASP script!

If a webpage content can be customised or used to complete an activity, interaction is either client-side or server-side.

**supports interaction within a webpage**

Client-side scripting enables interaction *within* a webpage. The code required to process user-input is downloaded and compiled by the browser or plug-in. An example of a client-side interaction is a rollover (typically triggered when choosing a navigation option).

Client-side scripting languages include JavaScript.

**Client-Side Vs Server-Side Scripting Language:****Client-side interaction**

Response to interaction may be more immediate (once the program code has been downloaded) services are secure (as no information is sent from the browser)

reliant on the user having using a specific browser and/or plug-in on their computer  
affected by the processing speed of the user's computer

**Server-side interaction**

Complex processes are often more efficient (as the program and the associated resources are not downloaded to the browser)

there are security considerations when sending sensitive information

does not rely on the user having specific browser or plug-in

affected by the processing speed of the host server. How the user connects to the internet affects both forms of interaction. For client-side scripting, the connection type affects the time it takes program code to be downloaded. For server-side processing, it affects the time taken for information to be sent to the server and the response downloaded.

**Variables and constants:**

A variable is a qualified name to a specific part of computer's memory. "Qualified" means the name of the variable must be legal in VBScript. Like other programming languages, you cannot name your variable to be one of the reserved words in that language. With variables, we can change and store data values. Without variables, we would only be able to display results; we won't be able to store the results for future use.

Variable example:

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

<%
dim name
name="Donald Duck"
response.write("My name is: " & name)
%>
```

```
</body>
</html>
```

Result:

My name is: Donald Duck

### **ASP Procedures:**

In ASP you can call a JavaScript procedure from a VBScript and vice versa.

```
<!DOCTYPE html>
<html>
<head>
<%
sub vbproc(num1,num2)
response.write(num1*num2)
end sub
%>
</head>

<body>
<p>You can call a procedure like this:</p>
<p>Result: <%call vbproc(3,4)%></p>
<p>Or, like this:</p>
<p>Result: <%vbproc 3,4%></p>
</body>
</html>
```

Result:

You can call a procedure like this:

Result: 12

Or, like this:

Result: 12

Example:2

```
<% @ language="javascript" %>
<!DOCTYPE html>
<html>
<head>
<%
function jsproc(num1,num2)
{
Response.Write(num1*num2)
}
```

```
%>
</head>

<body>
<p>Result: <%jsproc(3,4)%></p>
</body>
</html>
```

### **Constants:**

Constants just as variables are used to store information. The main difference between constants and variables is that constant value can not be changed in the process of running program. If we attempt to re-assign the value of the constant we'll get a run time error.

It can be mathematic constants, passwords, paths to files, etc. By using a constant you "lock in" the value which prevents you from accidentally changing it. If you want to run a program several times using a different value each time, you do not need to search throughout the entire program and change the value at each instance. You only need to change it at the beginning of the program where you set the initial value for the constant.

To declare a constant in VBScript we use the Const keyword. Have a look at the following example:

```
Const myConst = "myText"
```

'it is allowed to declare a few constants on the one line

```
Const PI = 3.14159, Wg = 2.78
```

Next example demonstrates the advantage of constants: during the calculations with number PI, you can type in your code only constants names instead of typing this number each time.

```
...
```

```
vCircle = PI * vRadius^2
```

### **Forms in ASP:**

The Request.QueryString and Request.Form commands are used to retrieve user input from forms.

**Get Method:**

```
<!DOCTYPE html>
<html>
<body>
<form action="demo_requery.asp" method="get">
Your name: <input type="text" name="fname" size="20" />
<input type="submit" value="Submit" />
</form>
<%
dim fname
fname=Request.QueryString("fname")
If fname<>"" Then
    Response.Write("Hello " & fname & "!<br>")
    Response.Write("How are you today?")
End If
%>
</body>
</html>
```

Your name:

Hello

How are you today?

EdwinJeyanthi!

**Post Method:**

```
<!DOCTYPE html>
<html>
<body>
<form action="demo_simpleform.asp" method="post">
Your name: <input type="text" name="fname" size="20" />
<input type="submit" value="Submit" />
</form>
<%
dim fname
fname=Request.Form("fname")
```

```
If fname<>" " Then
    Response.Write("Hello " & fname & "!<br>")
    Response.Write("How are you today?")
End If
%>
</body>
</html>
```

Your name:

Hello EdwinJeyanthi!  
How are you today?

**Form with Radio Buttons:**

```
<!DOCTYPE html>
<html>
<%
dim cars
cars=Request.Form("cars")
%>
<body>
<form action="demo_radiob.asp" method="post">
<p>Please select your favorite car:</p>

<input type="radio" name="cars"
<%if cars="Volvo" then Response.Write("checked")%>
value="Volvo">Volvo</input>
<br>
<input type="radio" name="cars"
<%if cars="Saab" then Response.Write("checked")%>
value="Saab">Saab</input>
<br>
<input type="radio" name="cars"
<%if cars="BMW" then Response.Write("checked")%>
value="BMW">BMW</input>
<br><br>
<input type="submit" value="Submit" />
</form>
```

```
<%  
if cars<>"" then  
    Response.Write("<p>Your favorite car is: " & cars & "</p>")  
end if  
%>  
</body>  
</html>
```

---

Please select your favorite car:

- ☒ Volvo  
☐ Saab  
☐ BMW

Your favorite car is: Volvo

### Cookies:

Cookies expire after a certain amount of time which you can set. Also, the clients browser must have cookies enabled for them to work.

Here is a very simple example of creating a cookie called "MYCOOKIE" with a key called "BgColor" and setting the cookie to expire in one year. In this example there is only one key, but cookies can have multiple keys.

```
<%  
Response.Cookies ("MYCOOKIE")("BgColor") = "Blue"  
Response.Cookies ("MYCOOKIE").Expires = DATE + 365  
%>
```

If you don't set the "expires" time the cookie will expire right away so don't leave that part out.

Create the cookie before the HTTP headers are written to the client browser. Basically that means you need to create cookies before the <HTML> tag on your page.

However if we turn on buffering like so <% Response.Buffer = True %> then we can create the cookie wherever we want within our code because when buffering is turned on no information is written to the client browser until the all the code is finished running.

The <% Response.Buffer = True %> needs to go right under the <% @LANGUAGE="VBSCRIPT" %> on your page.

Don't use underscores or other strange characters when naming your cookies because they may not work.

Examples of retrieving the cookie.

Do this if you don't care if the cookie exists or not.

```
<%  
Response.Write(Request.Cookies ("MYCOOKIE")("BgColor"))  
%>
```

- Check first to see if the cookie exists which is usually a good idea.

```
<%  
If Request.Cookies ("MYCOOKIE")("BgColor") <> "" Then  
Response.Write(Request.Cookies ("MYCOOKIE")("BgColor"))  
Else
```

```
We could do something like ask them to pick a BgColor and then set the cookie again  
'or we could just give them a default value for the Bgcolor since they don't have a cookie  
End If  
%>
```

When we create a cookie the info is saved on the clients computer.

With IE a text file is created in the cookies directory of the clients computer. It keeps track of the cookie. If they are using netscape there is a file called cookies.txt where the cookie info piles



up. Either way if they delete the cookie info the cookies will be gone when they return to wer site.

There is a lot more to using cookies but this is the basic info we need to know to create and retrieve them.

To read the names and values of all the cookies available to the page, you can loop through the Cookies collection using code such as the following.

```
System.Text.StringBuilder output = new System.Text.StringBuilder();
HttpCookie aCookie;
for(int i=0; i<Request.Cookies.Count; i++)
{
    aCookie = Request.Cookies[i];
    output.Append("Cookie name = " + Server.HtmlEncode(aCookie.Name)
        + "<br />");
    output.Append("Cookie value = " + Server.HtmlEncode(aCookie.Value)
        + "<br /><br />");
}
Label1.Text = output.ToString();
```

The following code example shows two ways to get the value of a cookie named username and display its value in a Label control:

```
if(Request.Cookies["userName"] != null)
    Label1.Text = Server.HtmlEncode(Request.Cookies["userName"].Value);

if(Request.Cookies["userName"] != null)
{
    HttpCookie aCookie = Request.Cookies["userName"];
    Label1.Text = Server.HtmlEncode(aCookie.Value);
}
```

### **The Global.asa file:**

The Global.asa file is an optional file that can contain declarations of objects, variables, and methods that can be accessed by every page in an ASP application.

All valid browser scripts (JavaScript, VBScript, JScript, PerlScript, etc.) can be used within Global.asa.

The Global.asa file can contain only the following:

- Application events
- Session events
- <object> declarations
- TypeLibrary declarations
- the #include directive
- **Note:** The Global.asa file must be stored in the root directory of the ASP application, and each application can only have one Global.asa file.

### Events in Global.asa

- In Global.asa you can tell the application and session objects what to do when the application/session starts and what to do when the application/session ends. The code for this is placed in event handlers. The Global.asa file can contain four types of events:
- **Application\_OnStart** - Occurs when the FIRST user calls the first page in an ASP application. This event occurs after the Web server is restarted or after the Global.asa file is edited. The "Session\_OnStart" event occurs immediately after this event.
- **Session\_OnStart** - This event occurs EVERY time a NEW user requests his or her first page in the ASP application.
- **Session\_OnEnd** - This event occurs EVERY time a user ends a session. A user-session ends after a page has not been requested by the user for a specified time (by default this is 20 minutes).
- **Application\_OnEnd** - This event occurs after the LAST user has ended the session. Typically, this event occurs when a Web server stops. This procedure is used to clean up settings after the Application stops, like delete records or write information to text files.
- A Global.asa file could look something like this:
- <script language="vbscript" runat="server">

```
sub Application_OnStart
'some code
end sub
```

```
sub Application_OnEnd
'some code
```

```
end sub
```

```
sub Session_OnStart  
'some code  
end sub
```

```
sub Session_OnEnd  
'some code  
end sub
```

```
</script>
```

**Note:** Because we cannot use the ASP script delimiters (<% and %>) to insert scripts in the Global.asa file, we put subroutines inside an HTML <script> element

### **Functions in ASP:**

We can write functions in ASP similar to the way you write them in VisualBASIC. It is good programming practice to use functions to modularize your code and to better provide reuse. To declare a subroutine (a function that returns no value), you simply type:

```
<% @ LANGUAGE="VBSCRIPT" %>  
<%  
sub SubroutineName( Parameters to Pass In )  
    'Code for Sub...  
end sub  
>%
```

A function differs from a subroutine in the fact that it returns data. To declare a function, the syntax is similar:

```
<% @ LANGUAGE="VBSCRIPT" %>  
<%  
function FunctionName( Parameters to Pass In )  
    'Code for Function...  
end function  
>%
```

Let's look at the code for a function that takes an integer value and returns the square of that value. Also included is code to call the function.

```
<% @ LANGUAGE="VBSCRIPT" %>  
<%
```

```
function Square(num)
    Square = num * num    end function
```

```
'Returns 25
Response.Write(Square(5))
```

```
'Should print "40 is less than 8^2"
if 40 < Square(8) then
    Response.Write("40 is less than 8^2")
else
    Response.Write("8^2 is less than 40")
end if
%>
```

If you do not understand an if statement, this tutorial should help!

To return a value from a function, you need to say the function's name = some value. That value is what is returned. In this case, we are returning num (the number passed in) times itself, or essentially num<sup>2</sup>.

**Important:** Whenever you call a function and expect it to return a value, you **must** use parenthesis to pass in the parameter(s). For example, we used Square(8). If you are calling a subroutine, you **cannot** use parenthesis. If we want to pass in value(s), we need to put a space after the sub name and then each parameter separated by a comma. Observe the example below:

```
<% @ LANGUAGE="VBSCRIPT" %>
<%
    sub PrintProfit(Revenue, Overhead, COGS, Admin)
        'Create a variable to store our profit
        Dim Profit

        Profit = CDbI(Revenue - (Overhead + COGS + Admin))

        Response.Write("$" & Profit)
    end sub

    'Call the sub with various values
    'Will output $100.0
```

PrintProfit 150, 25, 10, 15

'Will output \$0.5 (a bad year!!)

PrintProfit 200, 150, 45, 4.5

%>

### **ASP Object Model:**

Microsoft Internet Information Server (IIS) implements Active Server Pages (ASP) as an OLE automation server that has a hierarchical object framework. Figure A presents the ASP object model. The primary object in the ASP programming model is the ScriptingContext object, which exposes the interaction of the client browser. Because the ScriptingContext object is always available to ASP applications, you don't need to explicitly create a reference to it. The ScriptingContext object contains the six primary ASP objects, five built-in objects, and theObjectContext object.

The five built-in objects are the Application object, the Request object, the Server object, the Sessions object, and the Response object.

All active Web sessions use the Application object to share information among all users of an ASP application.

The Application object contains the Contents collection and the StaticObjects collection. Each Contents object in the Contents collection contains all the items for which you used ActiveX script commands to add them to the Web application. The StaticObjects collection contains all the objects for which you used the HTML <Object> tag to add them to the Web application. In addition, the Application object can contain user-defined objects that the Web application creates, and multiple users can share it.

The Request object receives requests from the Web clients. The Request object can receive all the data on the form, plus information about the current user. The Request object contains several collections, each of which represents a different set of information that can be returned for the Web client. Each ClientCertificate object in the ClientCertificate collection represents a certificate field that the Web client returns and that identifies the client.

The Cookies collection contains a set of Web cookies, where each cookie contains a small amount of information about the Web user. The Forms collection contains a set of Form objects,

and each object represents an HTML form. The QueryString collection contains a set of added URL arguments, and the ServerVariables collection contains a set of server environment variables.

You use the Server object to create other OLE objects that you want your Web application to use. For instance, the Server object's CreateObject method creates the ADO Connection and Recordset objects, which access SQL Server.

The Session object maintains information that relates to the current Web session. The Session object is much like the Application object, but the Application object pertains to all Web users, whereas the Session object refers only to the current Web session. The collection of Contents objects in the Session object contains all the items for which you used script commands to add them to the Web session.

TheObjectContext object provides access to the current object's context. It typically instantiates MTS objects or controls database transactions.

The Response object writes information into the HTML stream and sends that information to the client browser.

The Response object also supports a collection of cookies, where each cookie object contains information that can be written to the client system. The Request object can later read these cookies.

- Built-in objects
  - Communicate with a Web browser
  - Gather data sent by HTTP request
  - Distinguish between users
  - **Request**
    - Get or post information
    - Data provided by the user in an XHTML form
    - Access to information stored on client machine
      - Cookies
      - File upload (binary)
  - **Response**
    - Sends information to client
      - XHTML, text
  - **Server**

- Access to server methods or properties

### **Server Object**

The ASP Server object is used to access properties and methods on the server. Its properties and methods are described below:

#### **Properties**

| <b>Property</b>      | <b>Description</b>   |
|----------------------|--|
| <u>ScriptTimeout</u> | Sets or returns the maximum number of seconds a script can run before it is terminated |

#### **Methods**

| <b>Method</b>  | <b>Description</b>   |
|----------------|--|
| CreateObject   | Creates an instance of an object   |
| Execute        | Executes an ASP file from inside another ASP file                                  |
| GetLastError() | Returns an ASPError object that describes the error condition that occurred        |
| HTMLEncode     | Applies HTML encoding to a specified string  |
| MapPath        | Maps a specified path to a physical path   |
| Transfer       | Sends (transfers) all the information created in one ASP file to a second ASP file |
| URLEncode      | Applies URL encoding rules to a specified string                                   |

### **The ASPError Object**

The ASPError object displays information about errors in scripts

**The ASPError object was implemented in ASP 3.0 and is available in IIS5 and later.**

The ASPError object is used to display detailed information of any error that occurs in scripts in an ASP page.

The ASPError object is created when Server.GetLastError is called, so the error information can only be accessed by using the Server.GetLastError method.

The ASPError object's properties are described below (all properties are read-only):

Properties

| <b>Property</b>       | <b>Description</b>   |
|-----------------------|--|
| <u>ASPCode</u>        | Returns an error code generated by IIS   |
| <u>ASPDescription</u> | Returns a detailed description of the error (if the error is ASP-related)                                |
| <u>Category</u>       | Returns the source of the error (was the error generated by ASP? By a scripting language? By an object?) |
| <u>Column</u>         | Returns the column position within the file that generated the error                                     |
| <u>Description</u>    | Returns a short description of the error   |
| <u>File</u>           | Returns the name of the ASP file that generated the error  |
| <u>Line</u>           | Returns the line number where the error was detected   |
| <u>Number</u>         | Returns the standard COM error code for the error  |
| <u>Source</u>         | Returns the actual source code of the line where the error occurred                                      |

The VBScript arrays are 0 based, meaning that the array element indexing starts always from 0. The 0 index represents the first position in the array, the 1 index represents the second position in the array, and so forth.

There are two types of VBScript arrays - static and dynamic. Static arrays remain with fixed size throughout their life span. To use static VBScript arrays you need to know upfront the maximum number of elements this array will contain. If you need more flexible VBScript arrays with



variable index size, then you can use dynamic VBScript arrays. VBScript dynamic arrays index size can be increased/decreased during their life span.

### **Static Arrays**

Let's create an array called 'arrCars' that will hold the names of 5 cars:

```
<% @ LANGUAGE="VBSCRIPT" %>
```

```
<%
```

```
'Use the Dim statement along with the array name
```

```
'to create a static VBScript array
```

```
'The number in parentheses defines the array's upper bound
```

```
Dim arrCars(4)
```

```
arrCars(0)="BMW"
```

```
arrCars(1)="Mercedes"
```

```
arrCars(2)="Audi"
```

```
arrCars(3)="Bentley"
```

```
arrCars(4)="Mini"
```

```
'create a loop moving through the array
```

```
'and print out the values
```

```
For i=0 to 4
```

```
response.write arrCars(i) & "<br>"
```

```
Next    'move on to the next value of i
```

```
%>
```

Here is another way to define the array in VBScript:

```
<%
```

```
'we use the VBScript Array function along with a Dim statement
```

```
'to create and populate our array
```

```
Dim arrCars
```

```
arrCars = Array("BMW","Mercedes","Audi","Bentley","Mini") 'each element must be separated  
by a comma
```

```
'again we could loop through the array and print out the values
```

```
For i=0 to 4
```

```
response.write arrCars(i) & "<br>"
```

Next

%>

### **Dynamic Arrays**

Dynamic arrays come in handy when you aren't sure how many items your array will hold. To create a dynamic array you should use the Dim statement along with the array's name, without specifying upper bound:

<%

Dim arrCars

arrCars = Array()

%>

In order to use this array you need to use the ReDim statement to define the array's upper bound:

<%

Dim arrCars

arrCars = Array()

Redim arrCars(27)

%>

If in future you need to resize this array you should use the Redim statement again. Be very careful with the ReDim statement. When you use the ReDim statement you lose all elements of the array. Using the keyword PRESERVE in conjunction with the ReDim statement will keep the array we already have and increase the size:

<%

Dim arrCars

arrCars = Array()

Redim arrCars(27)

Redim PRESERVE arrCars(52)

%>

**PART-B**

**(Each Question carries 2 Marks)**

1. Define ASP.
2. What are the advantages of using java?
3. What is variable in ASP?
4. Define subroutine.
5. What is a cookie in ASP?
6. What are the types of ASP?

**PART-C**

**(Each Question carries 6 Marks)**

1. Explain Procedures and Forms in ASP with example.
2. Write about session object with example.
3. Explain server side scripting in ASP with example.
4. Write an ASP script to track cookies information.
5. Explain response and request object with example.
6. Explain server and error handling objects in ASP with example.
7. Explain about functions in ASP with example.
8. Explain Array objects in ASP with example.
9. Discuss about forms and application object with example.
10. Write an ASP Script to handle user request and get response from server with example.

## UNIT - IV

| S.No | Question   | Option1              | Option2                | Option3                 |
|------|--|----------------------|------------------------|-------------------------|
| 1    | _____ is a program that runs inside IIS  | ASP                  | HTML                   | JavaScript              |
| 2    | ADO stands for _____   | ActiveX Data Objects | ActiveX Date Objects   | Active Data Objects     |
| 3    | _____ is used to access the file system on the server  | Drives               | The FileSystemObject   | Folders                 |
| 4    | "<%= " is the same as:   | <%Equal              | <% Write               | <%Document.Write        |
| 5    | _____ can be used to access databases from your web pages.   | ADO                  | AAO                    | ADA                     |
| 6    | _____ is a collection of html documents reside on the server.  | Web Site             | Page                   | Program                 |
| 7    | _____ event is fired every time the web application is stopped.  | OnEnd                | OnStart                | OnPause                 |
| 8    | _____ is the property of session object that identifies the current session                              | Session ID           | Session                | Application ID          |
| 9    | _____ is the method which Gets and displays a content string   | ChooseContent        | Choosecontent          | ChooseCnt               |
| 10   | _____ is the method which Moves a specified file from one location to another                            | move()               | mov                    | Moves()                 |
| 11   | _____ is the method which Returns a Folder object for a specified path                                   | GetFolder            | Folder                 | getFolder               |
| 12   | _____ is the property of drive object which Returns the total size of a specified drive or network share | AvailableSpace       | TotalSpace             | TotalSize               |
| 13   | _____ is used to send output to the user from the server   | The Request Object   | The Session Object     | The ASP Response object |
| 14   | _____ is used to store information in name/value pairs   | The Folder Object    | The Drive Object       | The Dictionary object   |
| 15   | _____ is the method which Copies a specified file from one location to another                           | Copy()               | copy()                 | cpy()                   |
| 16   | _____ is the method which Retrieves and displays all of the content strings in the text file             | GetAllContent        | GetallContent          | GetContent              |
| 17   | _____ is used to return information about a specified file.  | The Drive            | The Folder Object      | The File object         |
| 18   | _____ is used to store and access variables from any page  | Requst Object        | The Application object | Session Object          |

|    |   |                       |                        |                         |
|----|---|-----------------------|------------------------|-------------------------|
| 19 | _____ is the method which Deletes a specified file  | Delete                | Del                    | Deletefile              |
| 20 | _____ is the property of server object Which Sets or returns the maximum number of seconds a script can run before it is terminated | Scripttimeout         | ScriptTimeout          | scriptTimeout           |
| 21 | _____ is used to access properties and methods on the server.   | The ASP Server object | The ASP Session object | The ASP response object |
| 22 | _____ is the collection of request object that holds variables value in the query.  | QueryString           | String                 | Querystring             |
| 23 | _____ is the event of session object which Occurs when a session starts   | Session_OnEnd         | Session_Onend          | Session_OnStart         |
| 24 | _____ is the property of drive object which Returns the amount of available space to a user on a specified drive or network share   | AvailableSpace        | Availablespace         | TotalSpace              |
| 25 | _____ is used to return information about a specified folder.   | The Folder Object     | The Drive Object       | The File Object         |
| 26 | _____ is the method which Returns the complete path from the root of the drive for the specified path                               | GetAbsoultePathName   | GetabsoultePathName    | GetPath                 |
| 27 | _____ is the method which Applies HTML encoding to a specified string   | HTMLEncode            | HTMLdecode             | HTMLcode                |
| 28 | _____ is the method which Creates an instance of an object  | CreateObject          | Createobject           | createObject            |
| 29 | _____ is the method which Executes an ASP file from inside another ASP file   | Execute               | execute                | Exe()                   |
| 30 | _____ is a method used to end a session immediately.  | Abandon               | Aband                  | Quit                    |
| 31 | _____ is the method which Copies one or more files from one location to another   | CopyFile              | Copyfile               | copyFile                |
| 32 | _____ is the method which Returns HTML that displays the advertisement in the page  | GetAdvertisement      | Getadvertisement       | GetAdvertisement        |
| 33 | _____ is the method which Checks if a specified file exists   | FileExists            | Fileexists             | Exists                  |
| 34 | _____ is the method which Sends (transfers) all the information created in one ASP file to a second ASP file                        | Send                  | Transfer               | sends                   |

|    |  |  |   |  |
|----|--|--|---|--|
| 35 | _____ is used to return information about a local disk drive or a network share.                               | Folder Object                                | File Object   | DriveObject  |
| 36 | _____ is the component displays a different HTML content string each time a user visits or refreshes the page. | Content Rotator                              | AdRotator   | Content Browser  |
| 37 | _____ is the method which Returns the index number of the current item in the Content Linking List file.       | GetListIndex                                 | GetListCount  | getListCount   |
| 38 | _____ is the method which Creates a new folder   | CreateFolder                                 | Createfolder  | createfolder   |
| 39 | _____ is the method which executes the specified SQL Statement or stored Procedure                             | Execute                                      | Start   | Close  |
| 40 | An ASP file has the file extension   | .asp   | .ap   | .asa   |
| 42 | ASP is a   | Microsoft Technology                         | Sun Micro System  | IBM  |
| 43 | ASP is a _____   | Server Side Scripting Language               | Client Side Javascript                                  | Cilent and Server Side Scripting                         |
| 44 | ASP server scripts are surrounded by delimiters, which?  | <%...%>                                      | <%>...</%>  | <script>...</script>                                     |
| 45 | How can you script your ASP code in JavaScript?  | JavaScript is the default scripting language | Start the document with:<br><% language="javascript" %> | Start the document with:<br><% @ language="javascript" % |
| 46 | How do you create a FileSystemObject?  | Server.CreateObject("FileSystemObject")      | Server.CreateObject("Scripting.FileSystemObject")       | Create("FileSystemObject")                               |
| 47 | How do you create a FileSystemObject?  | Server.CreateObject("FileSystemObject")      | Server.CreateObject("Scripting.FileSystemObject")       | Create("FileSystemObject")                               |
| 48 | How do you get information from a form that is submitted using the "get" method?                               | Request.Form                                 | Request.QueryString                                     | request object   |
| 49 | How do you get information from a form that is submitted using the "post" method?                              | Request.Form                                 | Request.QueryString                                     | request object   |
| 50 | How do you write "Hello World" in ASP  | "Hello World"                                | Document.Write("Hello World")                           | Response.Write("Hello World")                            |
| 51 | IIS need _____ to run asp programs   | Windows 95                                   | windows   | Windows NT 4.0   |

|    |  |                                    |   |                                     |
|----|--|------------------------------------|---|-------------------------------------|
| 52 | IIS stands for _____   | Internet Information Server        | Intranet Information Server                 | Internet Information Services       |
| 53 | PWS Stands for   | Personal Web Server                | Personal Web Service                        | Personal Web Socket                 |
| 54 | What does ASP stand for?   | Active Server Pages                | All Standard Pages                          | A Server Page                       |
| 55 | What is the default scripting language in ASP?                         | JavaScript                         | PERL  | EcmaScript                          |
| 56 | Which of the following is the correct syntax to include a file in asp. | <!--include file="somefilename"--> | <include file="somefilename"                | <!--#include file="somefilename"--> |
| 57 | Which one of these events is a standard Global.asa event?              | Application_OnStart                | Global.asa doesn't have any standard events | Session_Start                       |
| 58 | Choose the form in which Postback occur                                | HTMLForms                          | Webforms                                    | Winforms                            |
| 59 | Which of the following object is not an ASP component?                 | LinkCounter                        | Counter                                     | AdRotator                           |
| 60 | The first event triggers in an aspx page is                            | Page_Init()                        | Page_Load()                                 | Page_click()                        |

**Option4****Answer**

|                        |  |  |                         |
|------------------------|--|--|-------------------------|
| DHTML                  |  |  | ASP                     |
| AgentX Data Objects    |  |  | ActiveX Data Objects    |
| files                  |  |  | The FileSystemObject    |
| <%Response.Write       |  |  | <%Response.Write        |
| AVO                    |  |  | ADO                     |
| instruction            |  |  | Web Site                |
| OnStop                 |  |  | OnEnd                   |
| ObjectID               |  |  | Session ID              |
| ChooseCollection       |  |  | ChooseContent           |
| Match()                |  |  | move()                  |
| folder                 |  |  | GetFolder               |
| Space                  |  |  | TotalSize               |
| application object     |  |  | The ASP Response object |
| The File Object        |  |  | The Dictionary object   |
| Copied()               |  |  | Copy()                  |
| GetActivate            |  |  | GetAllContent           |
| The Directories Object |  |  | The File object         |
| The File object        |  |  | The Application object  |



|                        |  |  |                       |
|------------------------|--|--|-----------------------|
| Dele                   |  |  | Delete                |
| Script                 |  |  | ScriptTimeout         |
| The ASP request object |  |  | The ASP Server object |
| QueryLink              |  |  | QueryString           |
| Session_onStart        |  |  | Session_OnStart       |
| EmptySpace             |  |  | AvailableSpace        |
| The Directores Object  |  |  | The Folder Object     |
| GetFilesystem          |  |  | GetAbsoultePath Name  |
| HTMLCode               |  |  | HTMLEncode            |
| Create                 |  |  | CreateObject          |
| exe()                  |  |  | Execute               |
| Terminate              |  |  | Abandon               |
| Cpy                    |  |  | CopyFile              |
| GetAd                  |  |  | GetAdvertisemen t     |
| exists                 |  |  | FileExists            |
| Transfers              |  |  | Send                  |

|   |  |  |  |
|---|--|--|--|
| DrirectoryObject  |  |  | DriveObject  |
| Collection<br>Browser   |  |  | Content Rotator  |
| ListIndex   |  |  | GetListIndex   |
| Create  |  |  | CreateFolder   |
| Stop  |  |  | Execute  |
| .aspa   |  |  | .asp   |
| Infosys   |  |  | Microsoft<br>Technology  |
| Assembly<br>Language  |  |  | Server Sise<br>Scripting<br>Language                               |
| <&>...</&>  |  |  | <%...%>  |
| End the<br>document with:<br><%<br>language="javas<br>cript" %> |  |  | Start the<br>document with:<br><% @<br>language="javasc<br>ript" % |
| Create<br>Object:"Scriptin<br>g.FileSystemObj<br>ect"           |  |  | Server.CreateObj<br>ect("FileSystemO<br>bject")                    |
| Create<br>Object:"Scriptin<br>g.FileSystemObj<br>ect"           |  |  | Server.CreateObj<br>ect("FileSystemO<br>bject")                    |
| requestall  |  |  | Request.QueryString  |
| requestall  |  |  | Request.Form   |
| response  |  |  | Response.Write("<br>Hello World")                                  |
| Win 32  |  |  | Windows NT 4.0   |

|   |  |  |   |
|---|--|--|---|
| Information<br>Internet Services            |  |  | Internet<br>Information<br>Services         |
| Procedures Web<br>System                    |  |  | Personal Web<br>Server                      |
| Active Standard<br>Pages                    |  |  | Active Server<br>Pages                      |
| VBScript                                    |  |  | VBScript                                    |
| <!--%include file<br>="somefilename<br>"--> |  |  | <!--#include file<br>="somefilename"-<br>-> |
| Session_id                                  |  |  | Application_OnS<br>tart                     |
| webpage                                     |  |  | Webforms                                    |
| File Access                                 |  |  | LinkCounter                                 |
| page_ begin()                               |  |  | Page_Init()                                 |

**UNIT-V**

**SYLLABUS**

**ASP:** Collections & Control Structure-File system object: File System – Text Stream-Drive – File – Folder – Directory – ADO - sql & Databases for data driven applications-ASP Components: Ad Rotator – Browser Cap. – Content Linking – Content Rotator.

**ASP Collections:**

**IIS 6.0:**

Most of the ASP built-in objects provide collections. Collections are data structures similar to arrays that store strings, numbers, objects and other values. Unlike arrays, collections expand and contract automatically as items are retrieved or stored. The position of an item will also move as the collection is modified. You can access an item in a collection by its unique string key, by its index (position) in the collection, or by iterating through all the items in the collection.

**Accessing an Item by Name or Index:**

You can access a specific item in a collection by referencing its unique string key, or name. For example, the **Contents** collection holds any variables stored in the **Session** object. It can also hold any objects instantiated by calling **Server.CreateObject**. Suppose you have stored the following user information in the **Session** object:

```
<%  
Session.Contents("FirstName") = "Meng"  
Session.Contents("LastName") = "Phua"  
Session.Contents("Age") = 29  
%>
```

You can access an item by using the string key you associated with the item when you stored it in the collection. For example, the following expression returns the string "Meng":

```
<%= Session.Contents("FirstName") %>
```

You could also access an item by using the index, or number, associated with an item. For example, the following expression retrieves the information stored in the second position of the **Session** object and returns "Phua":

```
<%= Session.Contents(2) %>
```

ASP collections are numbered starting with 1. The index associated with an item might change as items are added to or removed from the collection. You should not depend on an item's index remaining the same. Indexed access is generally used to iterate through a collection, as described in the following topics, or to access an item in a read-only collection.

You can also use a shorthand notation to access items by name. ASP searches the collections associated with an object in a particular order. If an item with a particular name appears only once in an object's collections, you can eliminate the collection name (although doing so may affect performance):

```
<%= Session("FirstName") %>
```

Eliminating the collection name is generally safe when you are accessing items stored in the **Application** or **Session** object. For the **Request** object, however, it is safer to specify the collection name because the collections could easily contain items with duplicate names.

### **Iterating through a Collection:**

You can iterate through all the items in a collection to see what is stored in the collection or to modify the items. You must supply the collection name when you iterate through a collection. For example, you can use the VBScript **For...Each** statement to access the items you stored in the **Session** object:

```
<%
```

```
    'Declare a counter variable.
```

```
    Dim strItem
```

```
    'For each item in the collection, display its value.
```

```
For Each strItem In Session.Contents
    Response.Write Session.Contents(strItem) & "<BR>"
Next
%>
```

You can also iterate through a collection by using the VBScript **For...Next** statement. For example, to list the three items stored in **Session** by the previous example, use the following statements:

```
<%
'Declare a counter variable.
Dim intItem

'Repeat the loop until the value of counter is equal to 3.
For intItem = 1 To 3
    Response.Write Session.Contents(intItem) & "<BR>"
Next
%>
```

Because you do not usually know how many items are stored in a collection, ASP supports the **Count** property for a collection, which returns the number of items in the collection. You use the **Count** property to specify the end value of the counter.

```
<%
'Declare a counter variable.
Dim lngItem, lngCount

lngCount = Session.Contents.Count

'Repeat this loop until the counter equals the number of items
'in the collection.
For lngItem = 1 To lngCount
    Response.Write Session.Contents(lngItem) & "<BR>"
Next
%>
```

In JScript, you use the **for** statement to loop through a collection. For greater efficiency when using the **Count** property with a JScript **for** statement, you should assign the value of **Count** to a local variable and use that variable to set the end value of the counter. That way, the script engine does not have to look up the value of **Count** each time through the loop. The following example demonstrates this technique:

```
<%  
var intItem, intNumItems;  
  
intNumItems = Session.Contents.Count;  
  
for (intItem = 1; intItem <= intNumItems; intItem++)  
{  
    Response.Write(Session.Contents(intItem) + "<BR>")  
}  
%>
```

Microsoft JScript supports an **Enumerator** object that you can also use to iterate through an ASP collection. The **atEnd** method indicates whether there are any more items in the collection. The **moveNext** method moves to the next item in the collection.

```
<%  
Session.Contents("Name") = "User Name"  
Session.Contents("Department") = "Hardware"  
.  
.  
.  
  
//Create an Enumerator object.  
var myCollection = new Enumerator(Session.Contents);  
  
//Iterate through the collection and display each item.  
while (!myCollection.atEnd())  
{  
    var x = myCollection.item();  
    Response.Write(Session.Contents(x) + "<BR>");  
}
```

```
myCollection.moveToNext();  
}  
%>
```

### Iterating through a Collection with Subkeys:

Scripts might embed several related values in a single cookie to reduce the number of cookies passed between the browser and the Web server. The **Cookies** collection of the **Request** and the Web server. The Cookies collection of the Request and **Response** objects can thus hold multiple values in a single item. These subitems, or subkeys, can be accessed individually. Subkeys are supported only by the **Request.Cookies** and the Web server. The Cookies collection of the Request and Response objects can thus hold multiple values in a single item. These subitems, or subkeys, can be accessed individually. Subkeys are supported only by the **Request.Cookies** and **Response.Cookies** collections. **Request.Cookies** supports only read operations; **Response.Cookies** supports only write operations. The following creates a regular cookie and a cookie with a subkeys:

```
<%  
'Send a cookie to the browser.  
Response.Cookies("Food") = "Pineapple"  
  
'Send a cookie with subkeys to browser.  
Response.Cookies("Mammals")("Elephant") = "African"  
Response.Cookies("Mammals")("Dolphin") = "Bottlenosed"  
%>
```

The cookie text in the HTTP response sent to the browser would appear as the following:

HTTP\_COOKIE= Mammals=ELEPHANT=African&DOLPHIN=Bottlenosed; Food=Pineapple;

You can also enumerate all the cookies in the **Request.Cookies** collection and all the subkeys in a cookie. However, iterating through subkeys on a cookie that does not have subkeys will not produce an item. You can avoid this situation by first checking to see whether a cookie has subkeys by using the **HasKeys** attribute of the **Cookies** collection. This technique is demonstrated in the following example.

```
<%  
'Declare counter variables.
```



Dim Cookie, Subkey

```
'Display the entire cookie collection.
For Each Cookie In Request.Cookies
Response.Write Cookie
If Request.Cookies(Cookie).HasKeys Then
    Response.Write "<BR>"
    'Display the subkeys.
    For Each Subkey In Request.Cookies(Cookie)
        Response.Write " ->" & Subkey & " = " & Request.Cookies(Cookie)(Subkey) & "<BR>"
    Next
Else
    Response.Write " = " & Request.Cookies(Cookie) & "<BR>"
End If
Next
%>
```

This script would return the following results:

```
Mammals
->ELEPHANT = African
->DOLPHIN = Bottlenosed
Food = Pineapple
```

The Case of the Key Name

The **Cookies**, **Request**, **Response**, and **ClientCertificate** collections can reference the same, unique string key name. For example, the following statements reference the same key name, User, but return different values for each collection:

```
strUserID = Request.Cookies("User")
strUserName = Request.QueryString("User")
```

The case of the key name is set by the first collection to assign a value to the key. Examine the following script:

```
<%
'Retrieve a value from QueryString collection using the UserID key.
strUser = Request.QueryString("UserID")
```

```
'Send a cookie to the browser, but reference the key, UserId, which has a different spelling.  
Response.Cookies("UserId")= "1111-2222"  
Response.Cookies("UserId").Expires="December 31, 2000"  
%>
```

Although it may appear that key name `UserId` was assigned to the cookie, in actuality, the key name `UserID` (which is capitalized differently) was assigned to the cookie. The **QueryString** collection was first to define the case of this key.

Because references to collection values are independent of the case of the key name, this behavior will not affect your scripts unless your application uses case-sensitive processing of key names.

### **Iterating through a Collection of Objects:**

The **Session** and **Application** collections can hold either scalar variables or object instances. The **Contents** collection holds both scalar variables and object instances created by calling **Server.CreateObject**. The **Contents** collection holds both scalar variables and object instances created by calling **Server.CreateObject**. The **StaticObjects** collection holds objects created by using the HTML `<OBJECT>` tag within the scope of the **Session** object. For more information about instantiating objects in this manner, see **Setting Object Scope**.

When you iterate through a collection that contains objects, you can either access the object's **Session** or **Application** state information or access the object's methods or properties. For example, suppose your application uses several objects to create a user account, and each object has an initialization method. You could iterate through the **StaticObjects** collection to retrieve object properties:

```
<%  
For Each Object in Session.StaticObjects  
    Response.Write Object & ": " & Session.StaticObjects(Object)  
Next  
%>
```

## **ASP Control Structures:**

### **Branches**

#### **If**

The "if" statement is a fundamental control statement. It allows your program to perform a test, and act based on the results of that test. For example:

```
if ( (x = 1) and (y = 3) ) then  
sum = y - x  
end if
```

In this statement the value of the x variable is compared to 1 to see if it is equal, and the value of the y variable is compared with 3 to see if it is equal. The use of the "and" operator, adds an additional logical condition that says that the first comparison must be true and the second comparison must be true for the overall result of the test to be true. If the test results in an overall true condition then the statements that follow the if statement will be executed. If the test results are false nothing will occur.

An additional clause you can add to the "if" statement is the "else", an example:

```
if (sum = 0) then  
sum = x + y  
else  
subtotal = sum  
end if
```

This statement is read as: if sum equals 0 then sum equals x plus y, or else subtotal equals sum.

#### **ElseIf**

ElseIf can be used to refine your code so that you can reduce the number of statements and make your program code easier to read.

```
If (sum > 10) then  
Alert( "Sum is greater than ten")
```

```
ElseIf (sum > 5 ) then  
Alert( "Sum is greater than five")  
End If
```

```
If (sum > 10) then  
Alert( "Sum is greater than ten")  
ElseIf (sum > 5 ) then  
Alert( "Sum is greater than five")
```

### **Nested Ifs**

You can also nest your If statements. Nesting is where one statement is contained within another one. You would do this when you need to meet one condition before you test for the second one.

```
If (x>10) then  
If (y>5) then  
Rem do something  
ElseIf (y>10) then  
Rem do something else  
End if  
End if
```

### **Select Case**

The select case statement is handy when a variable may take on a number of values and you want to test for some of those values. The use of "select case" is shorter and easier to read than a number of "if" statements.

```
Select case n  
case 1  
REM start here if n equals 1.  
REM place code here  
case 2  
REM start here if n equals 2.  
REM place code here  
case 3  
REM start here if n equals 3.
```

REM place code here

Case else

REM if n is not equal to 1, 2 or 3

REM case else is optional

End Select

A case can also take multiple values:

Case 5, 10, 15

A case can also take strings instead of numeric values:

Case "red"

### **Loops**

A loop is a programming structure that forces the statements contained within its delimiters to execute over and over again until a condition is met at which point the loop ends.

#### **Do while ... Loop do until ... Loop :**

While a condition is true, execute one or more statements. "While loops" are especially useful when you do not know how many times you have to loop, but you know you should stop when you meet the condition.

dim x

x=1

do while x <= 10 : REM loop until x is greater than 10

until x is greaterthan 10

x = x + 1 : REM add one to the value of x

loop

x=1

do until x = 10 : REM loop until x is greater than 10

until x is greaterthan 10

x = x + 1 : REM add one to the value of x

loop

Loops that execute at least once.

x=1

do

until x is greater than 10

    x = x + 1 : REM add one to the value of x

loop until x = 10 : REM loop until x is greater than 10

### **For ... Next :**

"For loops" are useful when you know exactly how many times you want the loop to execute.

for x = 1 to 10 : REM loop while x is <= 10

do something ten times

Next

The loops normally increment by 1, you can alter increment using the ""step"" statement. You can change the step to be positive or negative.

i.e. for x = 0 to 10 step 2

i.e. for x = 10 to 0 step -2

Remember that your step value should increment and decrement to the final value.

Exit For

You can also exit out of a for loop before you have reached the end of the loop.

The "Exit For" statement can be used with a condition, like below.

for x = 1 to 10 : REM loop while x is <= 10

REM do something ten times

If (sum>10) then

Exit For

End if

Next

Learning VBScript

24

### **For Each ... Next :**

The For Each ... Next statement makes working with collections easier. An easy way to think about it is; loop through all the items in a collection. It is exceptionally handy because you don't need to determine the beginning and ending points of the collection to construct your loop. In the example below rs is the object and fields is a collection contained in that object. We will discuss collections in detail when we cover objects.

Dim customer\_field

```
For Each customer_field in rs.fields
    Rem Display the field
Next
```

### **While ... Wend:**

While ... Wend is an older loop construct. Microsoft recommends using the Do ... Loop because it affords more flexibility.

```
Dim X
X=1
While ( X < 10 )
    Rem do something while C < 10
    X=X+1
Wend
```

### **The File System Object:**

The File object is used to return information about a specified file. To work with the properties and methods of the File object, you will have to create an instance of the File object through the FileSystemObject object. First; create a FileSystemObject object and then instantiate the File object through the GetFile method of the FileSystemObject object or through the Files property of the Folder object. The following code uses the GetFile method of the FileSystemObject object to instantiate the File object and the DateCreated property to return the date when the specified file was created:

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

<%
dim fs, f
set fs=Server.CreateObject("Scripting.FileSystemObject")
set f=fs.GetFile(Server.MapPath("testread.txt"))
Response.Write("The file testread.txt was created on: " & f.DateCreated)
set f=nothing
set fs=nothing
```

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: II BCA  
COURSE CODE: 18CAU403

COURSE NAME: WEB PROGRAMMING  
UNIT: V (ASP) BATCH: 2018-2021

%>

</body>

</html>

Result:

The file testread.txt was created on: 5/14/2012 8:46:44 AM

The File object's properties and methods are described below:

Properties

| Property                | Description  |
|-------------------------|--|
| <u>Attributes</u>       | Sets or returns the attributes of a specified file                             |
| <u>DateCreated</u>      | Returns the date and time when a specified file was created                    |
| <u>DateLastAccessed</u> | Returns the date and time when a specified file was last accessed              |
| <u>DateLastModified</u> | Returns the date and time when a specified file was last modified              |
| <u>Drive</u>            | Returns the drive letter of the drive where a specified file or folder resides |
| <u>Name</u>             | Sets or returns the name of a specified file                                   |
| <u>ParentFolder</u>     | Returns the folder object for the parent of the specified file                 |
| <u>Path</u>             | Returns the path for a specified file  |
| <u>ShortName</u>        | Returns the short name of a specified file (the 8.3 naming convention)         |
| <u>ShortPath</u>        | Returns the short path of a specified file (the 8.3 naming convention)         |
| <u>Size</u>             | Returns the size, in bytes, of a specified file                                |
| <u>Type</u>             | Returns the type of a specified file   |



**Methods**

| <b>Method</b>           | <b>Description</b>  |
|-------------------------|---|
| <u>Copy</u>             | Copies a specified file from one location to another                      |
| <u>Delete</u>           | Deletes a specified file  |
| <u>Move</u>             | Moves a specified file from one location to another                       |
| <u>OpenAsTextStream</u> | Opens a specified file and returns a TextStream object to access the file |

**Drive property:**

The Drive property is used to return the drive letter of the drive where the specified file or folder resides.

Syntax

FileObject.Drive

FolderObject.Drive

Example for the File object

```
<%  
dim fs,f  
set fs=Server.CreateObject("Scripting.FileSystemObject")  
set f=fs.GetFile("c:\test.txt")  
Response.Write("File resides on drive: ")  
Response.Write(f.Drive)  
set f=nothing  
set fs=nothing  
%>
```

**Output:**

File resides on drive: c:

Example for the Folder object

```
<%  
dim fs,fo  
set fs=Server.CreateObject("Scripting.FileSystemObject")  
set fo=fs.GetFolder("c:\test")  
Response.Write("Folder resides on drive: ")  
Response.Write(fo.Drive)  
set fo=nothing  
set fs=nothing  
%>
```

**Output:**

Folder resides on drive: c:

The Copy method copies the specified file or folder from one location to another.

Syntax

FileObject.Copy(destination[,overwrite])

FolderObject.Copy(destination[,overwrite])

| Parameter   | Description  |
|-------------|--|
| destination | Required. Where to copy the file or folder. Wildcard characters are not allowed  |
| overwrite   | Optional. A Boolean value indicating whether an existing file or folder can be overwritten. True indicates that the file/folder can be overwritten, false indicates that the file/folder cannot be overwritten. Default is true. |

Example for the File object

```
<%  
dim fs,f  
set fs=Server.CreateObject("Scripting.FileSystemObject")
```

```
set f=fs.GetFile("c:\test.txt")
f.Copy("c:\new_test.txt",false)
set f=nothing
set fs=nothing
%>
```

The Delete method deletes a specified file or folder.

Syntax

FileObject.Delete[(force)]

FolderObject.Delete[(force)]

| Parameter | Description   |
|-----------|---|
| force     | Optional. A Boolean value that indicates whether a read-only file or folder are to be deleted. True indicates that a read-only file/folder will be deleted and false indicates that it will not be deleted. Default is false. |

Example for the File object

```
<%
dim fs,f
set fs=Server.CreateObject("Scripting.FileSystemObject")
set f=fs.GetFile("c:\test.txt")
f.Delete
set f=nothing
set fs=nothing
%>
```

Example for the Folder object

```
<%
dim fs,fo
set fs=Server.CreateObject("Scripting.FileSystemObject")
set fo=fs.GetFolder("c:\test")
fo.Delete
set fo=nothing
set fs=nothing
%>
```

The Move method moves the specified file or folder from one location to another.

Syntax

FileObject.Move(destination)

FolderObject.Move(destination)

| Parameter   | Description   |
|-------------|---|
| destination | Required. Where to move the file or folder. Wildcard characters are not allowed |

Example for the File object

```
<%  
dim fs,f  
set fs=Server.CreateObject("Scripting.FileSystemObject")  
Set f=fs.GetFile("c:\test.txt")  
f.Move("c:\test\test.txt")  
set f=nothing  
set fs=nothing  
>%
```

Example for the Folder object

```
<%  
dim fs,fo  
set fs=Server.CreateObject("Scripting.FileSystemObject")  
set fo=fs.GetFolder("c:\test")  
fo.Move("c:\asp\test")  
set fo=nothing  
set fs=nothing  
>%
```

Syntax

FileObject.OpenAsTextStream(mode,format)

| Parameter | Description  |
|-----------|--|
| mode      | Optional. How to open the file. <ul style="list-style-type: none"><li>• 1 = ForReading - Open a file for reading. You cannot write to this file</li><li>• 2 = ForWriting - Open a file for writing</li><li>• 8 = ForAppending - Open a file and write to the end of the file</li></ul> |

|        |   |
|--------|---|
| format | Optional. The format of the file. <ul style="list-style-type: none"><li>• 0 = TristateFalse - Default. Open the file as ASCII</li><li>• -1 = TristateTrue - Open the file as Unicode</li><li>• -2 = TristateUseDefault - Open the file using the system default</li></ul> |
|--------|---|

Example

```
<%  
dim fs,f,ts  
set fs=Server.CreateObject("Scripting.FileSystemObject")  
Set f=fs.GetFile("c:\test.txt")  
Set ts=f.OpenAsTextStream(ForWriting)  
ts.Write("Hello World!")  
ts.Close
```

```
Set ts=f.OpenAsTextStream(ForReading)  
Response.Write(ts.ReadAll)  
ts.Close  
set ts=nothing  
set f=nothing  
set fs=nothing  
>
```

Output:

Hello World!

### **ASP AdRotator Component:**

The ASP AdRotator component creates an AdRotator object that displays a different image each time a user enters or refreshes a page. A text file includes information about the images.

The AdRotator does not work with Internet Information Server 7 (IIS7).

Syntax

```
<%  
set adrotator=server.createobject("MSWC.AdRotator")  
adrotator.GetAdvertisement("textfile.txt")  
>
```

ASP AdRotator Example:

Assume that we have the following text file, named "ads.txt":

REDIRECT banners.asp

\*

w3s.gif

<http://www.w3schools.com>

Free Tutorials from W3Schools

50

xmlspy.gif

<http://www.altova.com>

XML Editor from Altova

50

The lines below the asterisk in the text file above specifies the name of the images (ads) to be displayed, the hyperlink addresses, the alternate text (for the images), and the display rates (in percent).

The first line in the text file above specifies what to happen when a visitor clicks on one of the images. The redirection page (banners.asp) will receive a querystring with the URL to redirect to.

To specify the height, width, and border of the image, you can insert the following lines under REDIRECT:

REDIRECT banners.asp

WIDTH 468

HEIGHT 60

BORDER 0

\*

w3s.gif

...

The "banners.asp" file looks like this:

Example

<%

url=Request.QueryString("url")

If url<>"" then Response.Redirect(url)

%>

<!DOCTYPE html>

```
<html>
<body>
<%
set adrotator=Server.CreateObject("MSWC.AdRotator")
response.write(adrotator.GetAdvertisement("textfile.txt"))
%>
</body>
</html>
```

//Source Code – AdRotator //

```
<%
url=Request.QueryString("url")
If url<>"" then Response.Redirect(url)
%>
<!DOCTYPE html>
<html>
<body>
```

```
<%
set adrotator=Server.CreateObject("MSWC.AdRotator")
adrotator.TargetFrame="target='_blank'"
response.write(adrotator.GetAdvertisement("text/advertisements.txt"))
%>
```

```
<p>
NOTE: Because images are changed randomly, and because this page has few images to choose
from, it will
often display the same advertisement twice in a row.
</p>
```

```
<p>NOTE: The AdRotator does not work with Internet Information Server 7 (IIS7).</p>
```

```
<p>
<a href="text/advertisements.txt">
</a>
</p>
```

</body>  
</html>



■ View TextFile

REDIRECT demo\_adrotator.asp  
\*  
w3s.gif  
http://www.w3schools.com  
Free Tutorials from W3Schools  
50  
xmlspy.gif  
http://www.altova.com  
XML Editor from Altova  
50

### ASP Browser Capabilities Component:

The ASP Browser Capabilities component creates a `BrowserType` object that determines the type, capabilities and version number of a visitor's browser.

When a browser connects to a server, a User Agent header is also sent to the server. This header contains information about the browser.

The `BrowserType` object compares the information in the header with information in a file on the server called "Browscap.ini".

If there is a match between the browser type and version number in the header and the information in the "Browscap.ini" file, the `BrowserType` object can be used to list the properties



of the matching browser. If there is no match for the browser type and version number in the Browscap.ini file, it will set every property to "UNKNOWN".

### Syntax

```
<%  
Set MyBrow=Server.CreateObject("MSWC.BrowserType")  
%>
```

### ASP Browser Capabilities Example

The example below creates a BrowserType object in an ASP file, and displays some of the capabilities of your browser:

### Example

```
<!DOCTYPE html>  
<html>  
<body>  
<%  
Set MyBrow=Server.CreateObject("MSWC.BrowserType")  
%>  
  
<table border="0" width="100%">  
<tr>  
<th>Client OS</th><th><%=MyBrow.platform%></th>  
</tr><tr>  
<td>Web Browser</td><td><%=MyBrow.browser%></td>  
</tr><tr>  
<td>Browser version</td><td><%=MyBrow.version%></td>  
</tr><tr>  
<td>Frame support?</td><td><%=MyBrow.frames%></td>  
</tr><tr>  
<td>Table support?</td><td><%=MyBrow.tables%></td>  
</tr><tr>  
<td>Sound support?</td><td><%=MyBrow.backgroundsounds%></td>  
</tr><tr>  
<td>Cookies support?</td><td><%=MyBrow.cookies%></td>  
</tr><tr>
```

```
<td>VBScript support?</td><td><%=MyBrow.vbscript%></td>
</tr><tr>
<td>JavaScript support?</td><td><%=MyBrow.javascript%></td>
</tr>
</table>

</body>
</html>
```

Output:

Client OS	WinNT
Web Browser	IE
Browser version	5.0
Frame support?	True
Table support?	True
Sound support?	True
Cookies support?	True
VBScript support?	True
JavaScript support?	True

### **ASP Content Linking Component:**

The ASP Content Linking component is used to create a quick and easy navigation system!

The Content Linking component returns a Nextlink object that is used to hold a list of Web pages to be navigated.

Syntax

```
<%
```

```
Set nl=Server.CreateObject("MSWC.NextLink")
```

```
%>
```

ASP Content Linking Example

First we create a text file - "links.txt":

asp\_intro.asp ASP Intro

asp\_syntax.asp ASP Syntax

asp\_variables.asp ASP Variables

asp\_procedures.asp ASP Procedures

The text file above contains the pages to be navigated. The pages must be listed in the same order you want them to be displayed, and it must also contain a description for each file name (use the tab key to separate file name from description).

If you want to add a page, or change the order of the pages in the list; you only have to modify the text file! The navigation will automatically be corrected!

Then we create an include file, "nlcode.inc". The .inc file creates a NextLink object to navigate between the pages listed in "links.txt".

"nlcode.inc":

```
<%  
dim nl  
Set nl=Server.CreateObject("MSWC.NextLink")  
if (nl.GetListIndex("links.txt")>1) then  
    Response.Write("<a href='" & nl.GetPreviousURL("links.txt")<br>    Response.Write(">Previous Page</a>")  
end if  
Response.Write("<a href='" & nl.GetNextURL("links.txt")<br>Response.Write(">Next Page</a>")  
%>
```

In each of the .asp pages listed in the text file "links.txt", put one line of code: **<!-- #include file="nlcode.inc"-->**. This line will include the code in "nlcode.inc" on every page listed in "links.txt" and the navigation will work.

### **ASP Content Rotator Component:**

The ASP Content Rotator component creates a ContentRotator object that displays a different content string each time a visitor enters or refreshes a page.

A text file, called the Content Schedule File, includes the information about the content strings. The content strings can contain HTML tags so you can display any type of content that HTML can represent: text, images, colors, or hyperlinks.

Syntax

```
<%  
Set cr=Server.CreateObject("MSWC.ContentRotator")  
%>
```

#### **ASP Content Rotator Example**

The following example displays a different content each time a visitor views the Web page. First, create a text file named "textads.txt" and place it in a subfolder called "text".

```
"textads.txt":
%% #3
<h2>This is a great day!!</h2>

%% #3


%% #4
<a href="http://www.w3schools.com">Visit W3Schools.com</a>
```

Notice the #number at the beginning of each content string. This number is an optional parameter that indicates the relative weight of the HTML content string. In the text file above, the Content Rotator will display the first and second content string three-tenth of the time, and the third string four-tenths of the time.

Then, create an ASP file, and insert the following code:

Example

```
<html>
<body>
<%
set cr=server.createobject("MSWC.ContentRotator")
response.write(cr.ChooseContent("text/textads.txt"))
%>
</body>
</html>
<!DOCTYPE html>
<html>
<body>
<p>
<a href="demo_textads.asp"></a>
</p>
<%
set cr=server.createobject("MSWC.ContentRotator")
response.write(cr.ChooseContent("text/textads.txt"))
%>

<p>
```

Because the content strings are changed randomly in the text file, and this page has only four content strings to choose from, sometimes the page will display the same content strings twice in a row.

</p>

</body>

</html>

■ View TextFile



Because the content strings are changed randomly in the text file, and this page has only four content strings to choose from, sometimes the page will display the same content strings twice in a row.

**PART-B**

**(Each Question carries 2 Marks)**

1. Define file in ASP?
2. What is Text Stream?
3. Define Ad Rotator.
4. What is ODBC?
5. What are the components of ASP?

**PART-C**

**(Each Question carries 6 Marks)**

1. Explain control structures in ASP with example.
2. Explain File system objects in ASP with example.
3. Explain File Operations in ASP with example.
4. Explain Text streams and drives in ASP.
5. Explain about ADO in ASP with example.
6. Write an ASP script to perform database connectivity using ADO.
7. Explain Browser capabilities and its components with example.
8. Explain Content Rotator in ASP with example.
9. Explain File handling features in ASP.
10. Explain content linking in ASP with example.

## UNIT - V

S.No	Question	Option1	Option2	Option3
1	ASP stands for _____	Active Session Page	Active Server Page	Application Session Page
2	X in ASPX stands for _____	Extensible	Extended	Extreme
3	What class does the ASP.NET Web Form class inherit from by default?	System.Web.UI.Page	System.Web.UI.Form	System.Web.UI.Page
4	We can manage states in asp.net application using	Session Objects	Application Objects	Viewstate
5	Attribute must be set on a validator control for the validation to work.	ControlToValidate	ControlToBind	ValidateControl
6	Caching type supported by ASP.Net	Output Caching	DataCaching	Output and Data Caching
7	What is used to validate complex string patterns like an e-mail address?	Extended expressions	Basic expressions	Regular expressions
8	File extension used for ASP.NET files.	.Web	.ASP	.ASPX
9	asp:label is used to as _____	alternative way of displaying text on web page	alternative way of displaying image on web page	alternative way of displaying table on web page
10	Default Session data is stored in ASP.Net.	StateServer	Session Object	InProcess
11	How do you get information from a form that is submitted using the "post" method?	Request.QueryString	Request.Form	Response.write
12	Which object can help you maintain data across users?	Application object	Session object	Response object
13	Which of the following ASP.NET object encapsulates the state of the client?	Session object	Application object	Response object
14	Which of the following control is used to validate that two fields are equal?	RegularExpressionValidator	CompareValidator	equals() method

15	Mode of storing ASP.NET session	InProc	StateServer	SQL Server
16	In ASP.NET in form page the object which contains the user name is _____ ?	Page.User.Identity	Page.User.IsInRole	Page.User.Name
17	Which of the following transfer execution directly to another page?	Server.Transfer	Response.Redirect	Server.Redirect
18	The actual work process of ASP.NET is taken care by _____?	inetinfo.exe	aspnet_isapi.dll	aspnet_wp.exe
19	Which of the following allow writing formatted output?	Response.Write()	Response.Output.WriteLine()	Server.Write( )
20	Which of the following denote the property in every validation control?	Caption property	Text property	Height
21	Which of the following languages can be used to write server side scripting in ASP.NET?	Java	VB	C++
22	When an .aspx page is requested from the web server, the out put will be rendered to browser in following format.	HTML	XML	WML
23	What's the difference between Response.Write() and Response.Output.WriteLine()?	Response.Output.WriteLine() allows you to flush output	Response.Output.WriteLine() allows you to buffer output	Response.Output.WriteLine() allows you to write formatted output
24	Which property of the session object is used to set the local identifier?	SessionId	LCID	Item
25	Select the caching type is not supported by ASP.Net	Output Caching	DataCaching	Display Caching
26	Which one of the following namespaces contains the definition for IDbConnection?	System.Data.Interf aces	System.Data.Comm on	System.Data
27	Which of the following languages are used to write server side scripting in ASP.NET?	C-sharp	VB	Both C-sharp and VB



28	How do you get information from a form that is submitted using the "post" method?	Request.QueryString	Request.Form	Response.Write
29	Web.config file is used...	Configures the time that the server-side codebehind module is called	To store the global information and variable definitions for the application	To configure the web server
30	Which of the following method must be overridden in a custom control?	The Paint() method	The Control_Build() method	The default constructor
31	How do we create a FileSystemObject?	Server.CreateObject("Scripting.FileSystemObject")	Create("FileSystemObject")	Create Object:"Scripting.FileSystemObject"
32	Which of the following tool is used to manage the GAC?	RegSvr.exe	GacUtil.exe	GacSvr32.exe
33	Why is Global.asax is used?	Declare Global variables	Implement application and session level events	No use
34	Which of the following is not a member of ADODBCommand object?	ExecuteScalar	ExecuteStream	Open
35	Which DLL translate XML to SQL in IIS?	SQLISAPI.dll	SQLXML.dll	LISXML.dll
36	Which of the following object is used along with application object in order to ensure that only one process accesses a variable at a time?	Synchronizing()	Synchronize()	ThreadLock()
37	Which of the following is not the way to maintain state?	View state	Cookies	Hidden fields
38	How many Global.asax file per project?	Only one	Any number	No Need
39	Where do we include the user lists for windows authentication?	< Credential>	< authorization>	< validation>

40	Which of the following authentication is best suited for a corporate network?	Windows	Form	User
41	What is the base class from which all Web forms inherit?	Master Class	Page Class	Session Class
42	WSDL stands for _____ ?	Web Server Description Language	Web Server Descriptor Language	Web Services Description Language
43	If one has two different web form controls in a application and if one wanted to know whether the values in the above two different web form control match what control must be used?	DataList	GridView	CompareValidator
44	What tags one need to add within the asp:datagrid tags to bind columns manually?	Set AutoGenerateColumns Property to false on the datagrid tag	Set AutoGenerateColumns Property to true on the datagrid tag	It is not possible to do the operation
45	Which method do you invoke on the DataAdapter control to load your generated dataset with data?	Load ( )	Fill( )	DataList( )
46	Which of the following provides an alternative way of displaying text on web page, is	< asp:label >	< asp:listitem >	< asp:button >
47	How ASP.Net Different from classic ASP?	Scripting is separated from the HTML, Code is interpreted seperately	Scripting is separated from the HTML, Code is compiled as a DLL, the DLLs can be executed on server	Code is separated from the HTML and interpreted Code is interpreted separately
48	What is the extension of a web user control file?	.Asmx	.Ascx	.Aspx
49	The number of forms that can be added to a aspx page is.	1	2	3

50	What is the maximum number of cookies that can be allowed to a web site?	1	10	20
51	How do you explicitly kill a user session?	Session.Close( )	Session.Discard( )	Session.Abandon()
52	How to implement authentication via we config?	Include the authentication element.	Include the authorization element.	Include the identity element.
53	_____ element in the we config file to run code using the permissions of a specific user	< credential> element	< authentication> element	< authorization> element
54	In which of the following format, output will be rendered to browser When an .aspx page is requested from the web server?	JSP	WML	HTML
55	C in CSS stands for	Continuous	Cascading	Cascaded
56	Which of these classes maps to the tag <input type="checkbox"/>	HtmlCheckBox	HtmlInputCheckBox	HtmlControl
57	A web application running on multiple servers is called as _____	Webfarm	Webform	Website
58	What is the protocol used to call a web service?	HTTP	SOAP	TCP
59	The default session out time is _____	20 seconds	20 minutes	30 minutes
60	The operation of reading metadata and using its contents is known as _____?	Reflection	Enumeration	Binding

**Option4****Answer**

Application  
Server Page

Active Server Page

External

Extended

System.Web.For  
m

System.Web.UI.Page

Session,  
Application and  
View State  
Objects

Session, Application  
and View State  
Objects

Validate

ControlToValidate

No Caching  
supported

Output and Data  
Catching

Irregular  
expressions

Regular expressions

.AS

.ASP

alternative way  
of displaying  
animations on  
web page

alternative way of  
displaying text on  
web page

Viewstate

InProcess

Response.writeIn

Request.Form

Server object

Application object

Server object

Session object

RequiredFieldVa  
lidator

CompareValidator

InProc, StateServer, SQL Server	InProc, StateServer, SQL Server
Page.UserName	Page.User.Identity
Response.Transf er	Server.Transfer
aspnet_wpinfo.ex e	aspnet_wp.exe
Server.FWrite( )	Response.Output.Writ e()
Width	Text property
C	VB
JSP	HTML
Response.Output .Write() allows you to stream output	Response.Output.Writ e() allows you to buffer output
Key	Key
No Caching supported	Display Caching
System.Data.Con nection	System.Data.Connecti on
C++	Both C-sharp and VB

Response.WriteLine

Request.Form

To configure the web browser

To store the global information and variable definitions for the application

The Render() method

The Render() method

Server.CreateObject("FileSystemObject")

Server.CreateObject("Scripting.FileSystemObject")

GacMgr.exe

GacUtil.exe

Declare local variables

Implement application and session level events

ExecuteReader

Open

SQLIIS.dll

SQLISAPI.dll

ThreadLock()

Synchronize()

Request object

Request object

Two

Only one

< authentication>

< authorization>

server

Windows

Cookie Class

Page Class

Web Services  
Descriptor  
Language

Web Services  
Description Language

Listview

CompareValidator

Set  
AutoGenerateColumns  
Property to  
true on the  
datagrid tag

Set  
AutoGenerateColumns  
Property to false on  
the datagrid tag

DataBind()

Fill( )

<  
asp:disbutton >

< asp:label >

No difference

Code is separated  
from the HTML and  
interpreted Code is  
interpreted separately

.Asdx

.Ascx

more than 3

3

more than 30

1

Session.End

Session.Abandon()

Include the  
deny element.

Include the  
authorization element.

< identity>  
element

< identity> element

XML

HTML

Continuous

Cascading

HtmlCheck

HtImInputCheckBox

Webpage

Webfarm

SMTP

SOAP

1 hour

20 minutes

Serialization

Reflection



**KARPAGAM ACADEMY OF HIGHER EDUCATION***(Established under Section 3 of UGC Act, 1956)***Coimbatore - 641 021****BCA Degree Examination****(For the candidates admitted from 2018 onwards)****Fourth Semester****FIRST INTERNAL EXAMINATION****WEB PROGRAMMING****Class: II BCA****Maximum: 50 Marks****Date:****Time: 2 Hrs****Part - A (20 X 1 =20 Marks)****(Answer all the questions)**

- HTML stands for\_\_\_\_  
 a) **Hyper Text Markup Language**      b) Hyper Text Model Language  
 c) Hyper Text Markup Link      d) Hyper Text Model Link
- \_\_\_\_\_ is the core building block of HTML.  
 a) Text      b) Image      **c) Tag**      d) Objects
- Elements that employ one-sided tags are called \_\_\_\_\_.  
 a) opening tag      b) closing tag      **c) empty elements**      d) two-sided tag
- Which of the following is the syntax of the comment tag?  
 a) <--comment-->      **b) <!--comment-->**      c) <-comment->      d) <--comment--!>
- The tag used in HTML to link it with other URL's is: \_\_\_\_\_.  
**a) <a>...</a>**      b) <b>...</b>      c) <u>...</u>      d) <li>...</li>
- Which tag allows you to add a row in a table?  
 a) <td> and </td>      b) <cr> and </cr>      c) <th> and </th>      **d) <tr> and </tr>**
- \_\_\_\_\_ is the tag used define a cell.  
 a) <tdata>      b) <table>      **c) <td>**      d) <cell>
- URL stands for \_\_\_\_\_.  
 a) Universal Resource Location      **b) Uniform Resource Location**  
 c) Universal Resource Link      d) Uniform Resource Link
- Ordered lists are preceded by \_\_\_\_\_.  
**a) numerals**      b) closed circle      c) open circle      d) square
- \_\_\_\_\_ attribute is used to provide the thickness of borders around each table cell  
 a) bgcolor      b) thick      **c) border**      d) light
- The data of definition list is provided using \_\_\_\_\_ tag  
 a) <DL>      b) <LI>      c) <DT>      **d) <DD>**
- The output of a HTML file can be viewed using \_\_\_\_\_.  
 a) document      b) slide show      **c) browser**      d) interpreter
- Which of the following is not a type of list in html?  
 a) ordered list      b) unordered list      c) definition list      **d) graphics list**
- \_\_\_\_\_defines the largest heading.  
 a) <h2>      **a) <h1>**      c) <h6>      d) <h5>

15. \_\_\_\_\_ is the tag used to define horizontal line  
 a) **<hr>**                      b) <hrule>                      c) <br>                      d) <hline>
16. HTML file is saved with extension \_\_\_\_\_  
 a) .ht                      **b) .html**                      c) .txt                      d) .doc
17. CSS stands for \_\_\_\_\_  
 a) Computing Style Sheet                      **b) Cascading Style Sheet**  
 c) Computer Style Sheet                      d) Commercial Style Sheet
18. JavaScript is a \_\_\_\_\_ language  
 a) Object-Oriented                      b) High-level                      c) Assembly                      **d) Object-Based**
19. What is the return type of standard JavaScript objects?  
 a) Xml                      **b) object**                      c) DOM                      d) html
20. JavaScript code between a pair of “script” tags are called \_\_\_\_\_  
 a) Non-inline                      b) External                      c) Referenced                      **d) Inline**

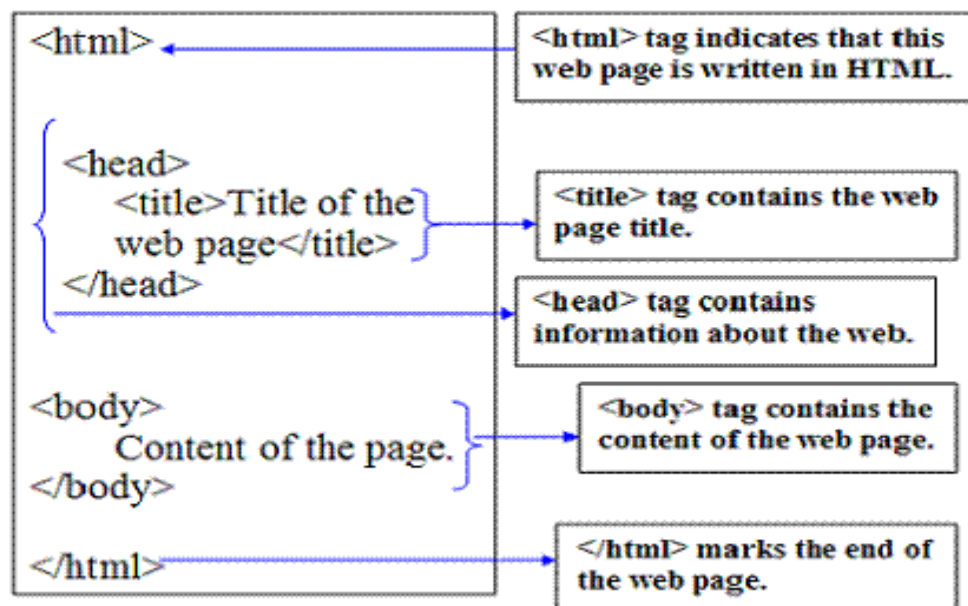
**Part - B (3 X 2=6 Marks)**  
**(Answer all the questions)**

21. Write the structure of HTML document.

**Answer:**

An HTML document has two main parts:

1. **Head:** The head element contains title and meta data of a web document.
2. **Body:** The body element contains the information that you want to display on a web page.



22. Name the different lists in HTML?

**Answer:**

1. Ordered List
2. Un-Ordered List
3. Definition List

23. How do you create a link in HTML?

**Answer:**

HTML links are hyperlinks. You can click on a link and jump to another document. When you move the mouse over a link, the mouse arrow will turn into a little hand.

In HTML, links are defined with the anchor `<a>` tag:

**Syntax:**

```
<a href="url">link text</a>
```

**Example:**

```
<a href="https://www.w3schools.com/html/">Visit our HTML tutorial</a>
```

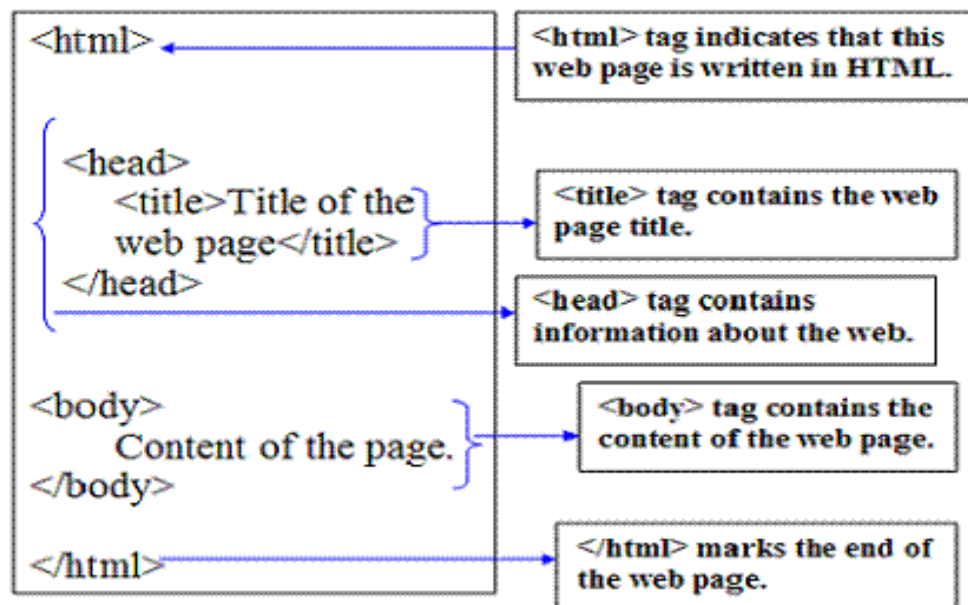
**Part - C (3 X 8=24 Marks)**  
**(Answer all the questions)**

24. a) Explain about the Basic Structure of HTML.

**Answer:**

An HTML document has two main parts:

1. **Head:** The head element contains title and meta data of a web document.
2. **Body:** The body element contains the information that you want to display on a web page.



Document Type `<HTML>` `</HTML>`

Title `<TITLE></TITLE>`

Header `<HEAD>` `</HEAD>`

Body `<BODY>` `</BODY>`

- HTML document begins and ends with HTML tag i.e. `<HTML>` `</HTML>`  
Here `<HTML>` indicates the browser that it is a HTML document and `</HTML>` tells the browser that HTML document is completed.

- Header Tag i.e. <HEAD></HEAD>

Header Tag does not contain any text, it only contains the Title Tag in it.

- Title tag i.e. <TITLE></TITLE>

Anything written between this tag is not displayed on the screen but it is used to identify the Webpage.

- Body tag i.e. <BODY></BODY>

This is the main part of HTML document. The content which is to be displayed on screen as webpage should be written here. Body Tag contains the text as well as various tags but only the text will be displayed on Webpage.

**Example:**

```
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

(Or)

- b) Write about Unordered Lists in HTML with example.

**Answer:**

HTML lists are used to present list of information in well formed and semantic way. There are three different types of list in HTML and each one has a specific purpose and meaning:

- Unordered list — Used to group a set of related items, in no particular order.
- Ordered list — Used to group a set of related items, in a specific order.
- Description list — Used to display a list of terms and their descriptions.

**Unordered Lists:**

An unordered list created using the <ul> tag, and each list item starts with the <li> tag.

The list items in unordered lists are marked with bullets (small black circles), by default.

The CSS list-style-type property is used to define the style of the list item marker:

Disc - Sets the list item marker to a bullet (default)

Circle - Sets the list item marker to a circle

square - Sets the list item marker to a square

none - The list items will not be marked

## Example

```
<ul style="list-style-type:disc" >
  <li>Chocolate Cake</li>
  <li>Black Forest Cake</li>
  <li>Pineapple Cake</li>
</ul>
```

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

- Chocolate Cake
- Black Forest Cake
- Pineapple Cake

25. a) Explain how to create table in HTML with example.

### Answer:

The HTML tables are created using the **<table>** tag in which the **<tr>** tag is used to create table rows and **<td>** tag is used to create data cells. The elements under **<td>** are regular and left aligned by default

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

### Table Heading

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

### Cellpadding and Cellspacing Attributes

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

### Colspan and Rowspan Attributes

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

### Tables Backgrounds

You can set table background using one of the following two ways –

- **bgcolor** attribute – You can set background color for whole table or just for one cell.
- **background** attribute – You can set background image for whole table or just for one cell.

You can also set border color also using **bordercolor** attribute.

### Table Height and Width

You can set a table width and height using **width** and **height** attributes. You can specify table width or height in terms of pixels or in terms of percentage of available screen area.

### Example

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML Table Background</title>
  </head>
  <body>
    <table border = "1" bordercolor = "green" bgcolor = "yellow" height="400" width="200" >
      <tr>
        <th>Column 1</th>
        <th>Column 2</th>
        <th>Column 3</th>
      </tr>
      <tr>
        <td rowspan = "2">Row 1 Cell 1</td>
        <td>Row 1 Cell 2</td>
        <td>Row 1 Cell 3</td>
      </tr>
      <tr>
        <td>Row 2 Cell 2</td>
        <td>Row 2 Cell 3</td>
      </tr>
      <tr>
        <td colspan = "3">Row 3 Cell 1</td>
      </tr>
    </table>
  </body>
</html>
```

(Or)

b) Explain Form elements in HTML with example.

### Answer:

An HTML form can contain input elements like text fields, checkboxes, radio-buttons, submit buttons and more. A form can also contain select lists, textarea, fieldset, legend, and label elements.

The <form> tag is used to create an HTML form:

```
<form>
.
input elements
.
</form>
```

### The Input Element

The most important form element is the <input> element.

The <input> element is used to select user information.

An `<input>` element can vary in many ways, depending on the type attribute. An `<input>` element can be of type text field, checkbox, password, radio button, submit button, and more.

The most common input types are described below.

### Text Fields

`<input type="text">` defines a one-line input field that a user can enter text into:

```
<form>
First name: <input type="text" name="firstname"><br>
Last name: <input type="text" name="lastname">
</form>
```

The HTML code above looks in a browser:

First name:   
Last name:

The form itself is not visible. Also note that the default width of a text field is 20 characters.

### Password Field

`<input type="password">` defines a password field:

```
<form>
Password: <input type="password" name="pwd">
</form>
```

The HTML code above looks in a browser:

Password:

The characters in a password field are masked (shown as asterisks or circles).

### Simple drop-down list

Creating a simple drop-down list.

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

<form action="">
<select name="cars">
<option value="volvo">Volvo</option>
<option value="saab">Saab</option>
<option value="fiat">Fiat</option>
<option value="audi">Audi</option>
</select>
```

```
</form>
```

```
</body>
```

```
</html>
```

## Drop-down list with a pre-selected value

Creating a drop-down list with a pre-selected value.

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<form action="">
```

```
<select name="cars">
```

```
<option value="volvo">Volvo</option>
```

```
<option value="saab">Saab</option>
```

```
<option value="fiat" selected>Fiat</option>
```

```
<option value="audi">Audi</option>
```

```
</select>
```

```
</form>
```

```
</body>
```

```
</html>
```

## Radio Buttons

`<input type="radio">` defines a radio button. Radio buttons let a user select ONLY ONE of a limited number of choices:

```
<form>
```

```
<input type="radio" name="sex" value="male">Male<br>
```

```
<input type="radio" name="sex" value="female">Female
```

```
</form>
```

the HTML code above looks in a browser:



Male



Female

## Checkboxes

`<input type="checkbox">` defines a checkbox. Checkboxes let a user select ZERO or MORE options of a limited number of choices.

```
<form>
```

```
<input type="checkbox" name="vehicle" value="Bike">I have a bike<br>
```

```
<input type="checkbox" name="vehicle" value="Car">I have a car
```

```
</form>
```

How the HTML code above looks in a browser:



- ☐ I have a bike
- ☐ I have a car

## Submit Button

`<input type="submit">` defines a submit button.

A submit button is used to send form data to a server. The data is sent to the page specified in the form's action attribute. The file defined in the action attribute usually does something with the received input:

```
<form name="input" action="html_form_action.asp" method="get">  
Username: <input type="text" name="user">  
<input type="submit" value="Submit">  
</form>
```

The HTML code above looks in a browser:

Username:

## Resetting the Form:

HTML `<button>` type Attribute

## HTML `<button>` tag

### Definition and Usage

The type attribute specifies the type of button.

Specify the type attribute for the `<button>` element. Different browsers may use different default types for the `<button>` element.

### Syntax

```
<button type="button|submit|reset">
```

### Attribute Values

Value	Description
Button	The button is a clickable button
Submit	The button is a submit button (submits form-data)
Reset	The button is a reset button (resets the form-data to its initial values)

### Example

Two button elements that act as one submit button and one reset button (in a form):

```
<form action="demo_form.asp" method="get">
  First name: <input type="text" name="fname"><br>
  Last name: <input type="text" name="lname"><br>
  <button type="submit" value="Submit">Submit</button>
  <button type="reset" value="Reset">Reset</button>
</form>
```

### **Form Action Attribute:**

#### **Definition and Usage**

The action attribute specifies where to send the form-data when a form is submitted.

#### **Example**

On submit, send the form-data to a file named "demo\_form.asp" (to process the input):

```
<form action="demo_form.asp" method="get">
  First name: <input type="text" name="fname"><br>
  Last name: <input type="text" name="lname"><br>
  <input type="submit" value="Submit">
</form>
```

The action attribute is supported in all major browsers.

#### **Syntax**

```
<form action="URL">
```

#### **Attribute Values**

<b>Value</b>	<b>Description</b>
<i>URL</i>	Where to send the form-data when the form is submitted.

#### **Possible values:**

- An absolute URL - points to another web site (like action="http://www.example.com/example.htm")
- A relative URL - points to a file within a web site (like action="example.htm")

### **Form Method Attribute:**

#### **Definition and Usage**

The method attribute specifies how to send form-data (the form-data is sent to the page specified in the action attribute).

The form-data can be sent as URL variables (with method="get") or as HTTP post transaction (with method="post").

#### **Notes on GET:**

- Appends form-data into the URL in name/value pairs
- The length of a URL is limited (about 3000 characters)

- Never use GET to send sensitive data! (will be visible in the URL)
- Useful for form submissions where a user want to bookmark the result
- GET is better for non-secure data, like query strings in Google

#### Notes on POST:

- Appends form-data inside the body of the HTTP request (data is not shown in URL)
- Has no size limitations
- Form submissions with POST cannot be bookmarked

#### Syntax

<form method="get|post">

Attribute Values

Value	Description
Get	Default. Appends the form-data to the URL in name/value pairs: URL?name=value&name=value
Post	Sends the form-data as an HTTP post transaction

26. a) Explain Arithmetic and Logical operators in JavaScript with example.

#### Answer:

JavaScript supports the following types of operators.

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

Lets have a look on all operators one by one.

#### Arithmetic Operators

JavaScript supports the following arithmetic operators –

Assume variable A holds 10 and variable B holds 20, then –

Sr.No	Operator and Description
1	<p><b>+</b> (<b>Addition</b>)</p> <p>Adds two operands</p> <p><b>Ex:</b> A + B will give 30</p>

2	<b>- (Subtraction)</b> Subtracts the second operand from the first <b>Ex:</b> A - B will give -10
3	<b>* (Multiplication)</b> Multiply both operands <b>Ex:</b> A * B will give 200
4	<b>/ (Division)</b> Divide the numerator by the denominator <b>Ex:</b> B / A will give 2
5	<b>% (Modulus)</b> Outputs the remainder of an integer division <b>Ex:</b> B % A will give 0
6	<b>++ (Increment)</b> Increases an integer value by one <b>Ex:</b> A++ will give 11
7	<b>-- (Decrement)</b> Decreases an integer value by one <b>Ex:</b> A-- will give 9

## Logical Operators

JavaScript supports the following logical operators –

Assume variable A holds 10 and variable B holds 20, then –

Sr.No	Operator and Description
1	<b>&amp;&amp; (Logical AND)</b> If both the operands are non-zero, then the condition becomes true.

	<b>Ex:</b> (A && B) is true.
2	<b>   (Logical OR)</b> If any of the two operands are non-zero, then the condition becomes true. <b>Ex:</b> (A    B) is true.
3	<b>! (Logical NOT)</b> Reverses the logical state of its operand. If a condition is true, then the Logical NOT operator will make it false. <b>Ex:</b> ! (A && B) is false.

(Or)

b) Explain about If . . . Else Statement with example.

**Answer:**

**if Statement:**

Use the **if** statement to specify a block of JavaScript code to be executed if a condition is true.

**Syntax**

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

**if** is in lowercase letters. Uppercase letters (If or IF) will generate a JavaScript error.

**Example:**

Make a "Good day" greeting if the hour is less than 18:00:

```
if (hour < 18) {
    greeting = "Good day";
}
```

**The if..else Statement**

Use the **else** statement to specify a block of code to be executed if the condition is false.

```
if (condition)
{
    block of code to be executed if the condition is true
}
else
{
    block of code to be executed if the condition is false
}
```

### **Example**

If the hour is less than 18, create a "Good day" greeting, otherwise "Good evening":

```
if (hour < 18) {  
    greeting = "Good day";  
} else {  
    greeting = "Good evening";  
}
```