

Biyani's Think Tank

Concept based notes

Web Technology

(MCA)

Nitasha Jain

Dept. of MCA(IT)
Biyani Girls College, Jaipur



Published by :

Think Tanks

Biyani Group of Colleges

Concept & Copyright :

Biyani Shikshan Samiti

Sector-3, Vidhyadhar Nagar,

Jaipur-302 023 (Rajasthan)

Ph : 0141-2338371, 2338591-95 □ Fax : 0141-2338007

E-mail : acad@biyanicolleges.org

Website :www.gurukpo.com; www.biyanicolleges.org

Edition: 2011

Price :

While every effort is taken to avoid errors or omissions in this Publication, any mistake or omission that may have crept in is not intentional. It may be taken note of that neither the publisher nor the author will be responsible for any damage or loss of any kind arising to anyone in any manner on account of such errors and omissions.

Leaser Type Setted by :

Biyani College Printing Department

Preface

I am glad to present this book, especially designed to serve the needs of the students. The book has been written keeping in mind the general weakness in understanding the fundamental concepts of the topics. The book is self-explanatory and adopts the "Teach Yourself" style. It is based on question-answer pattern. The language of book is quite easy and understandable based on scientific approach.

Any further improvement in the contents of the book by making corrections, omission and inclusion is keen to be achieved based on suggestions from the readers for which the author shall be obliged.

I acknowledge special thanks to Mr. Rajeev Biyani, *Chairman* & Dr. Sanjay Biyani, *Director (Acad.)* Biyani Group of Colleges, who are the backbones and main concept provider and also have been constant source of motivation throughout this Endeavour. They played an active role in coordinating the various stages of this Endeavour and spearheaded the publishing work.

I look forward to receiving valuable suggestions from professors of various educational institutions, other faculty members and students for improvement of the quality of the book. The reader may feel free to send in their comments and suggestions to the under mentioned address.

Author

□ □ □

Syllabus

The internet: history of the world wide web, hardware and software trend, object technology - java script object, scripting for the web-browser portability.

Introduction of HTML: introduction, markup language, editing HTML : common tags, headers, text styles, linking, images, formatting text, horizontal rules and more line breaks, unordered lists, nested and ordered lists, basic HTML tables : intermediate HTML tables and formatting : basic HTML forms, more complex HTML forms, internal linking, creating and using image maps.

Java script - introduction to scripting: introduction- memory concepts- arithmetic- decisionmaking. Java script control structures, Java script functions: introduction - program modules in java script - function definitions, duration of identifiers, scope rules, recursion, java script global functions.

Java script arrays: introduction, array-declaring and allocating arrays, references and reference parameters - passing arrays to functions, multiple subscripted arrays. Java script objects: introduction, math, string, data, boolean and number objects.

Dynamic HTML : CSS : introduction - inline styles, creating style sheets with the style element, conflicting styles, linking external style sheets, positioning elements, backgrounds, element dimensions, text flow and the box model, user style sheets.

Dynamic HTML: object model and collections: introduction, object referencing, collections all and children, dynamic style, dynamic positioning, using the frames collection, navigator object.

Dynamic HTML: event model : introduction, event ON CLICK, event ON LOAD - error handling with ON ERROR, tracking the mouse with event, more DHTML events. Filters and Transitions: Dynamical HTML: Client side scripting with VB script: Introduction - operators- data types and control structures - VB script functions - arrays -string manipulation classes and objects.

Introduction to PHP - Advantages of PHP - Functions - Data types - Arrays - SQL - Connecting Databases using ODBC - Files - Forms - Images -Imap objects.

Contents

S.No	Chapter Name	Page No
1	HTML With Tags and Attribute	7 - 31
2	Java Script	32 - 65
3	Arrays	66 - 77
4	Number Name	78 - 85
5	Object Oriented Programming	86 - 91
6	DHTML	92 - 95
7	Dhtml - CSS	96 - 100
8	PHP	101 - 120
9	ODBC Connectivity	121 - 123

Chapter 1

HTML With Tags and Attribute

Q1. What Exactly Is HTML?

Ans: The name *HTML* stands for *Hypertext Markup Language*. That's a mouthful. Many people who create Web pages and work in HTML often forget what the letters stand for. The term's hypertext portion refers to the cross-links, also called hyperlinks, between Web pages. The term's markup language portion refers to the commands that format the Web pages that the users see. Knowing how to write and use HTML is the goal, not remembering the archaic abbreviation.

Note: The term *HTML language* is as redundant as *ATM machine* and *PIN number*. Literally, *HTML language* means *Hypertext Markup Language*. Redundant or not, *HTML language* is often the phrase used, even by experienced HTML programmers.

The Internet is more than just a bunch of Web pages. The Internet consists of Web pages, e-mail, text, voice, video chat sessions, and an assortment of other tasks that often hide behind the scenes from typical Internet users. Amidst the array of Internet components, a Web page comprises the most important piece of the Internet because a Web page is the user interface to the information that resides on the Internet. Close to one billion Web pages comprise the World Wide Web (WWW). Virtually every Web page that you've ever visited has two things in common:

- ❖ They contain formatted text and graphic images.
- ❖ They are created, in whole or in part, using the HTML language.

It may surprise you to learn that HTML is a language that has absolutely no formatted text or graphic images. The HTML language consists solely of unformatted text. That text, however, contains instructions, called *tags* or *command tags* that define exactly how formatted text and graphics appear on Internet Web pages. In other words, HTML determines how a

Web page browser displays the information your HTML-based Web pages produce.

Q2. What is HTML?

Ans: HTML is a language for describing web pages.

- HTML stands for Hyper Text Markup Language
- HTML is not a programming language, it is a markup language
- A markup language is a set of markup tags
- HTML uses markup tags to describe web pages

Q3: Define HTML filename extension.

Ans: A Web page, defined in an HTML file, always has the filename extension html (Or htm if you want to be compatible with Windows 3x users, although fewer and Fewer of them exist). The html extension separates the file type from ordinary, unformatted text files whose extensions might be txt. Many browsers, such as Internet Explorer, will refuse to open your file with an extension such as txt, except by starting another program such as Notepad and loading the text file into that secondary program for your viewing and editing work. Some browsers will open a file whose name does not end with the html extension, but will refuse to interpret any HTML command tags. In such a case, the file will appear inside the browser window displaying the nitty-gritty command tags themselves instead of performing the formatting actions that the command tags request.

Q4: Define <html><html> tags.

Ans: *Beginning and Ending <html> and </html>Tags* every Web page should begin with the following HTML start tag:

<html>

Every Web page should end with the following HTML end tag:

</html>

The poem, therefore, looks like this with those two enclosing tags:

<html>

Roses are red,

The Web is sure growing.

You can use HTML,

To keep your page flowing.

```
</html>
```

More is needed to make this an appealing Web page. All HTML tags are enclosed between angled brackets. Often, related tags appear in pairs with one beginning the formatting process and the other terminating that format. The `<html>` and `</html>` tags indicate the very beginning and ending of a Web page. The *end tag* contains the same command name as the start tag except it begins with a forward slash to distinguish where the tag pair begins and ends.

Q5: Define `<head>` and `<title>` tag.

Ans: The title command tag must appear inside a special section of your Web page called the *header section*. Before adding the title's tags, you must first create the header section with the `<head>` and `</head>` command tags. Start these tags immediately after the opening `<html>` tag, like this:

```
<html>
```

```
<head>
```

```
</head>
```

Roses are red,

The Web is sure growing.

You can use HTML,

To keep your page flowing.

```
</html>
```

Add ample spacing to make your HTML files readable and to make the command tags and HTML sections pronounced. Subsequent HTML examples in this weekend course include plenty of this *whitespace* to make the file readable and easy to maintain. The value that you type between the title tags becomes the actual title you want the Web browser to display in the browser window's title bar.

```
<html>
```

```
<head>
```

```
<title>Poem to make you feel good</title>
```

```
</head>
```

Roses are red,

The Web is sure growing.

You can use HTML,

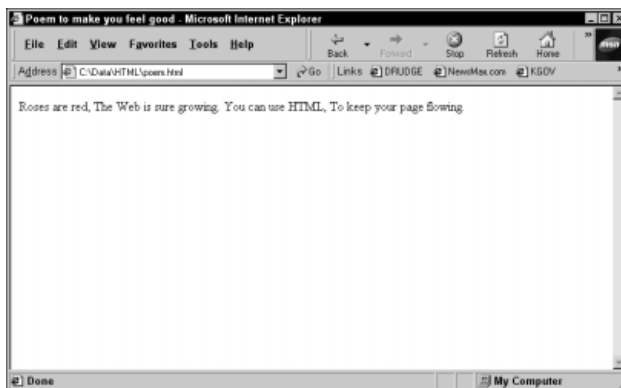
To keep your page flowing.

</html>

Figure shows the resulting browser window. The browser window displays

the poem's title, "Poem to make you feel good."

The window's title



**Q6: Define
 tag?**

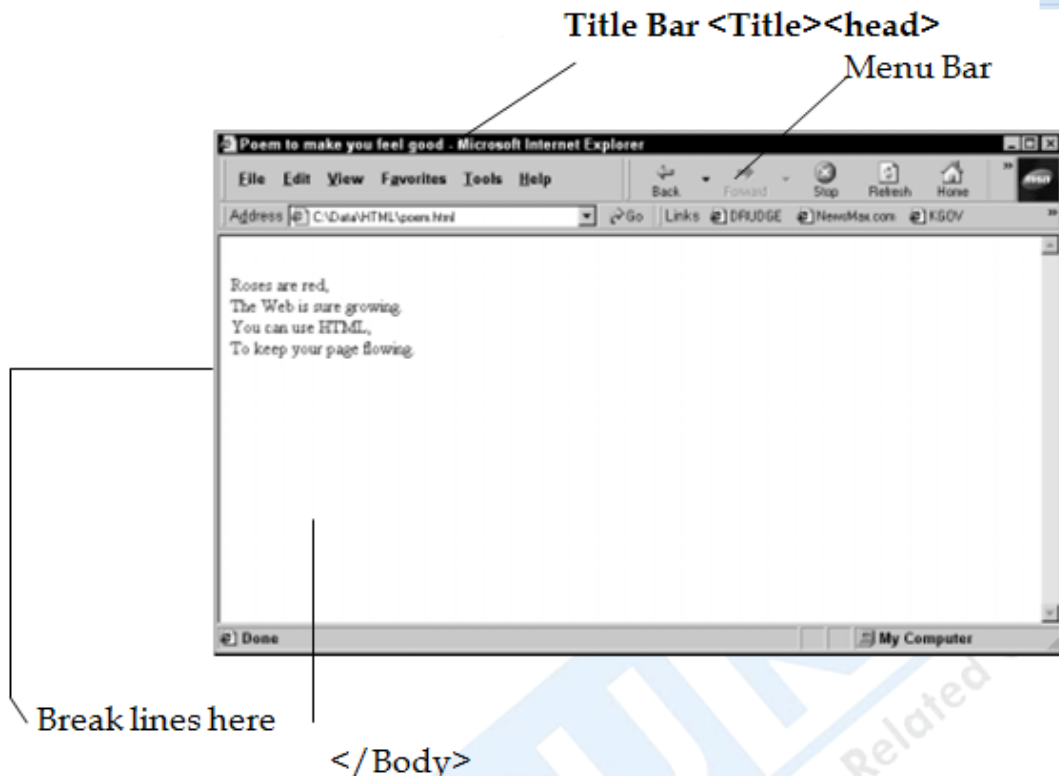
Ans: Use the *break tag* to break lines. The format of the tag is as follows:

Text that appears on its own line. The
 tag is special because, unlike so many other command tags,
 has no corresponding end tag. The
 tag is a stand-alone tag because it requests that the browser move down to the next line on the screen before displaying the text that follows. Adding
 to the beginning of each line in the poem produces a four-line poem. Here is the complete HTML file:

```
<html>
<head>
<title>Poem to make you feel good</title>
</head>
<body>
<br>Roses are red,
<br>The Web is sure growing.
<br>You can use HTML,
<br>To keep your page flowing.
</body>
</html>
```

Output

<Title>
<Body>



Note:- The
 tag creates a line break at each location in which it is laced. The first line of the poem would appear one line higher without the
 tag in front of it. You can put the
 tag at the end of a line to force a line break for subsequent text.

Q7: Which resolution displays the most information: 640×480 or 800×600?

Ans: A Web page displayed at 800 × 600 pixels of resolution. the same Web page displayed at 640 × 480. Notice how much less of the Web page the lower resolution displays. actually, the resolutions 800 × 600 and 640 × 480 are misleading because the user's browsers consume much of the screen because of the menu, toolbars, and status bar. A more realistic design area, if you want to hit virtually every Web user in the world, is only 580 × 315 on a Mac and 635 × 314 on a PC.

Q8: Define with example HTML Documents (Web Pages)?

Ans: HTML Documents (Web Pages)

- HTML documents describe web pages
- HTML documents contain HTML tags and plain text
- HTML documents are also called web pages

The purpose of a web browser (like Internet Explorer or Firefox) is to read HTML documents and display them as web pages. The browser does not display the HTML tags, but uses the tags to interpret the content of the page:

```
<html>
<body>
<h1>My First Heading</h1>
<p>My first paragraph.</p>
</body>
</html>
```

Example Explained

- The text between <html> and </html> describes the web page
- The text between <body> and </body> is the visible page content
- The text between <h1> and </h1> is displayed as a heading
- The text between <p> and </p> is displayed as a paragraph

Q9. Define Body <body></body>tag.

Ans: The <body> element defines the body of the HTML document.

The element has a start tag <body> and an end tag </body>.

Example

```
<body>
<p>This is my first paragraph.</p>
</body>
```

The element content is another HTML element (a p element).

Q10: Define HTML Headings.

Ans: HTML headings are defined with the <h1> to <h6> tags.

Headings Are Important->Use HTML headings for headings only. Don't use headings to make text BIG or bold.

Search engines use your headings to index the structure and content of your web pages. Since users may skim your pages by its headings, it is

important to use headings to show the document structure. H1 headings should be used as main headings, followed by H2 headings, then the less important H3 headings, and so on.

Example

```
<h1>hello</h1>
```

```
<h2>hello</h2>
```

```
<h3>hello</h3>
```

```
<h4>hello</h4>
```

```
<h5>hello</h5>
```

```
<h6>hello</h6>
```

Q11. Define HTML Paragraphs

Ans: HTML paragraphs are defined with the <p> tag.

Example:

```
<p>this is my first paragraph writing</p>
```

Q12 Explain the html elements

Ans: HTML Element Syntax

- An HTML element starts with a **start tag / opening tag**
- An HTML element ends with an **end tag / closing tag**
- The **element content** is everything between the start and the end tag
- Some HTML elements have **empty content**
- Empty elements are **closed in the start tag**
- Most HTML elements can have **attributes**

Q13 What is HTML Attributes?

Ans: HTML Attributes are:

- HTML elements can have **attributes**
- Attributes provide **additional information** about an element
- Attributes are always specified in **the start tag**
- Attributes come in name/value pairs like: **name="value"**

Attribute Example

HTML links are defined with the <a> tag. The link address is specified in the href attribute:

```
<a href="http://www.gurukpo.com">This is a link</a>
```

Q14: Define HTML Lines

Ans: HTML Lines The `<hr />` tag creates a horizontal line in an HTML page. The `hr` element can be used to separate content:

Example

```
<p>hello students</p>
```

```
<hr />
```

```
<p>How Are You</p>
```

```
<hr />
```

```
<p>God Bless you</p>
```

Q15: How to use comment in html document?

Ans: **HTML Comments**-> Comments can be inserted into the HTML code to make it more readable and understandable. Comments are ignored by the browser and are not displayed.

Comments are written like this:

Example

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

Q16: Define HTML Formatting tags?

Ans: HTML Formatting Tags like `` and `<i>` for formatting output, like bold or *italic* text.

These HTML tags are called formatting tags (look at the table below)

Tag	Description
<code></code>	Defines bold text
<code><big></code>	Defines big text
<code></code>	Defines emphasized text
<code><i></code>	Defines italic text
<code><small></code>	Defines small text
<code></code>	Defines strong text
<code><sub></code>	Defines subscripted text
<code><sup></code>	Defines superscripted text

Q17 How can we use linking in html document?**Ans: HTML Hyperlinks (Links)**

A hyperlink (or link) is a word, group of words, or image that you can click on to jump to a new document or a new section within the current document. When you move the cursor over a link in a Web page, the arrow will turn into a little hand.

Links are specified in HTML using the <a> tag.

The <a> tag can be used in two ways:

1. To create a link to another document, by using the href attribute
2. To create a bookmark inside a document, by using the name attribute

Syntax of linking

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

The target Attribute: The target attribute specifies where to open the linked document.

Example

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

For example External linking

```
<Html>
<Head><title>linking internal</title>
</head>
<Body>
<a href="www.gurukpo.com">
Gurukpo data</a>
<h1>hello</h1>
<h2>hello</h2>
<h3>hello</h3>
<h4>hello</h4>
<h5>hello</h5>
<h6>hello</h6>
</body>
</html>
```

The name Attribute: The name attribute specifies the name of an anchor. The name attribute is used to create a bookmark inside an HTML document.

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

For example Internal linking

```
<Html>
<Head><title>linking internal</title>
</head>
<Body>
<a href="#gurukpo">
Gurukpo data</a>
<h1>hello</h1>
<h2>hello</h2>
<h3>hello</h3>
<h4>hello</h4>
<h5>hello</h5>
<h6>hello</h6>
<a name="gurukpo">
Useful tips define in this paragraph
</a>
</body>
</html>
```

Q18 How to define an image in html document?**Ans: HTML Images - The Tag and the Src Attribute**

In HTML, images are defined with the tag. The tag is empty, which means that it contains attributes only, and has no closing tag. To display an image on a page, you need to use the src attribute. Src stands for "source". The value of the src attribute is the URL of the image you want to display.

Syntax for defining an image:

```

```

The URL points to the location where the image is stored.

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:

```

```

Q19: Explain table tag

Ans: HTML Tables

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.

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>
```

How the HTML code above looks in a browser:

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

HTML Tables and the Border Attribute

If you 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 will 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 your browser:

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

Q20: Explain Ordered and unordered list in html?

Ans: HTML Unordered Lists

An unordered list starts with the tag. Each list item starts with the tag. The list items are marked with bullets (typically small black circles).

```
<ul>
<li>Coffee</li>
```

```
<li>Milk</li>
</ul>
```

How the HTML code above looks in a browser:

- Coffee
- Milk

HTML Ordered Lists

An ordered list starts with the tag. Each list item starts with the tag. The list items are marked with numbers.

```
<ol>
<li>Coffee</li>
<li>Milk</li>
</ol>
```

How the HTML code above looks in a browser:

1. Coffee
2. Milk

HTML Definition Lists

A definition list is a list of items, with a description of each item. The <dl> tag defines a definition list.

The <dl> tag is used in conjunction with <dt> (defines the item in the list) and <dd> (describes the item in the list):

```
<dl>
<dt>Coffee</dt>
<dd>- black hot drink</dd>
<dt>Milk</dt>
<dd>- white cold drink</dd>
</dl>
```

How the HTML code above looks in a browser:

Coffee

- black hot drink

Milk

- white cold drink

HTML List Tags

Tag	Description
<u></u>	Defines an ordered list

<u></u>	Defines an unordered list
<u></u>	Defines a list item
<u><dl></u>	Defines a definition list
<u><dt></u>	Defines an item in a definition list
<u><dd></u>	Defines a description of an item in a definition list

Q21: Explain html Forms

Ans: HTML Forms

HTML forms are used to pass data to a server.

A 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>
```

HTML Forms - 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 used 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'><br/>
</form>
```

How the HTML code above looks in a browser:

First name:

Last name:

Note: 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><input type="password" name="pass" />
</form>
```

How the HTML code above looks in a browser:

Password:

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

Radio Buttons

<input type="radio" /> defines a radio button. Radio buttons let a user select ONLY ONE 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>
```

How the HTML code above looks in a browser:

Male

Female

Checkboxes

<input type="checkbox" /> defines a checkbox. Checkboxes let a user select ONE 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>
```

How the HTML code above looks in a browser:

Username:

If you type some characters in the text field above, and click the "Submit" button, the browser will send your input to a page called "html_form_action.asp". The page will show you the received input.

Q22: Create a Form with suitable information?

Ans:

```
<html>
<body background="c:\lucky\a3.jpg">
<p align=center><font color=red><font size=+1>biyani girls
college</font></p>
<p align=center><font color=red><font size=+1>computer education
center</font></p>
<p><font color=red><font size=+1>application form</font></p>
<form>
name <input type="text"width=30><br><br>
father's name <input type="text"width=30><br><br>
date of birth <select name="dd"><br>
<option>1</option>
<option>2</option>
<option>3</option>
<option>4</option>
<option>5</option>
<option>6</option>
```

```
<option>7</option>
<option>8</option>
<option>9</option>
<option>10</option>
<option>11</option>
<option>12</option>
<option>13</option>
<option>14</option>
<option>15</option>
<option>16</option>
<option>17</option>
<option>18</option>
<option>19</option>
<option>20</option>
<option>21</option>
<option>22</option>
<option>23</option>
<option>24</option>
<option>25</option>
<option>26</option>
<option>27</option>
<option>28</option>
<option>29</option>
<option>30</option>
<option>31</option>
</select>
<select name="mm">
<option>january</option>
<option>february</option>
<option>march</option>
<option>aprial</option>
<option>may</option>
<option>june</option>
<option>july</option>
<option>august</option>
<option>september</option>
<option>octomber</option>
```

```
<option>november</option>
<option>december</option>
</select>
<select name="yy">
<option>1980</option>
<option>1981</option>
<option>1982</option>
<option>1983</option>
<option>1984</option>
<option>1985</option>
<option>1986</option>
<option>1987</option>
<option>1988</option>
<option>1989</option>
<option>1990</option>
<option>1991</option>
<option>1992</option>
<option>1993</option>
<option>1994</option>
<option>1995</option>
<option>1996</option>
<option>1997</option>
<option>1998</option>
<option>1999</option>
<option>2000</option>
<option>2001</option>
<option>2002</option>
<option>2003</option>
<option>2004</option>
<option>2005</option>
<option>2006</option>
<option>2007</option>
<option>2008</option>
<option>2009</option>
<option>2010</option>
</select>
<br>
```

```

<br>
address <input type="text" width=50><br><br>
sex<br><br>
male<input type="radio" name="a" value=0>
female<input type="radio" name="a" value=0>
<br>
<br>
hobbies<br><br>
sports<input type="checkbox" name="s" value=0>
shopping<input type="checkbox" name="sh" value=0>
<br>
computer<input type="checkbox" name="c" value=0>
reading<input type="checkbox" name="r" value=0>
<br>
<br>
password<input type="password"><br><br>
<p align=center><input type="submit" name="yy" value="submit"></p>
<p align=center><input type="reset" name="b" value="reset"></p>
</form>
</html>

```

Biyani Girls COLLEGE
COMPUTER EDUCATION CENTER

APPLICATION FORM

NAME

FATHER'S NAME

DATE OF BIRTH 1 | JANUARY | 1980

ADDRESS

SEX

MALE FEMALE

HOBBIES

SPORTS SHOPPING

COMPUTER READING

PASSWORD

Q23 Explain frames in HTML

Ans HTML Frames with frames, we can display more than one HTML document in the same browser window. Each HTML document is called a frame, and each frame is independent of the others.

The disadvantages of using frames are:

- Frames are not expected to be supported in future versions of HTML
- Frames are difficult to use. (Printing the entire page is difficult).
- The web developer must keep track of more HTML documents

The HTML frameset Element

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

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

The HTML frame Element

The <frame> tag defines one particular window (frame) within a frameset.

In the example below we have a frameset with two columns.

The first column is set to 25% of the width of the browser window. The second column is set to 75% of the width of the browser window. The document "frame_a.htm" is put into the first column, and the document "frame_b.htm" is put into the second column:

```
<frameset cols="25%,75%">
  <frame src="frame_a.htm" />
  <frame src="frame_b.htm" />
</frameset>
```

Note: The frameset column size can also be set in pixels (cols="200,500"), and one of the columns can be set to use the remaining space, with an asterisk (cols="25%,*").

Basic Notes - Useful Tips

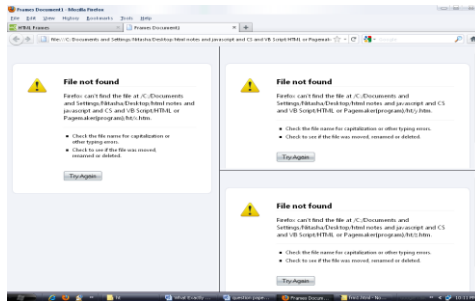
Tip: If a frame has visible borders, the user can resize it by dragging the border. To prevent a user from doing this, you can add noresize="noresize" to the <frame> tag.

Note: Add the <noframes> tag for browsers that do not support frames.

Important: You cannot use the <body></body> tags together with the <frameset></frameset> tags! However, if you add a <noframes> tag containing some text for browsers that do not support frames, you will have to enclose the text in <body></body> tags! See how it is done in the first example below.

Q24: Create a Frame

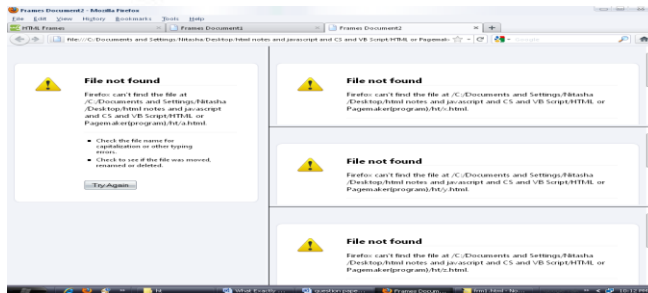
Ans



```
<html>
<head>
<title>Frames Document1</title>
</head>
<frameset cols="40%,50%">
<frame src="x.htm">
<frameset rows="25%,25%">
<frame src="y.htm">
<frame src="z.htm">
</frameset>
</frameset>
</html>
```

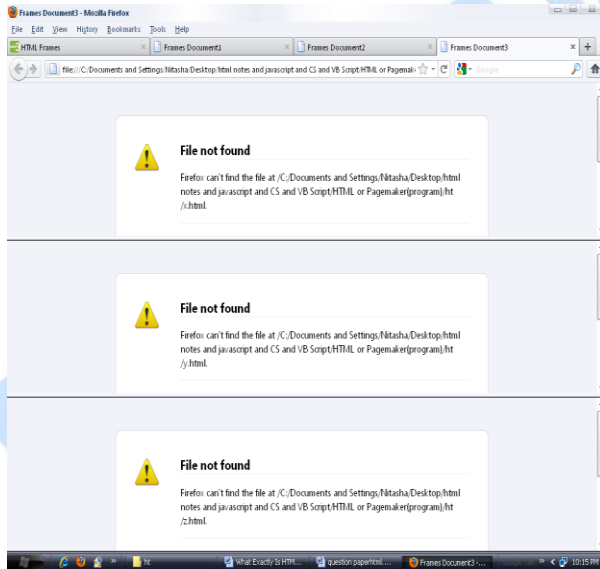
Q25: Create a frame like that

Ans



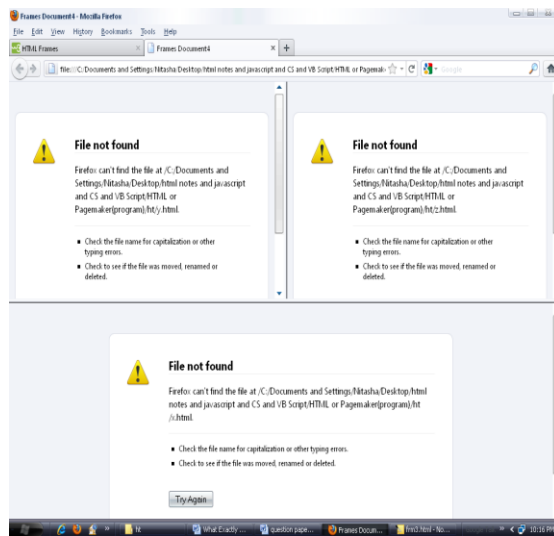
```
<html>
<head>
<title>Frames Document2</title>
</head>
<frameset cols="40%,60%">
<frame src=" a.html">
<frameset rows="20%,20%,20%">
<frame src="x.html">
<frame src="y.html">
<frame src="z.html">
</frameset>
</html>
```

Q26: Create a frame
Ans



```
<html>
<head>
<title>Frames Document3</title>
<head>
<frameset rows="30%,30%,30%">
<frame src="x.html">
<frame src="y.html">
```

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

Q27: Create a Frame**Ans.**

```
<html>
<head>
<title>Frames Document4</title>
</head>
<frameset rows="50%,50%">
<frameset cols="25%,25%">
<frame src="y.html">
<frame src="z.html">
</frameset>
<frame src="x.html">
</frameset>
</html>
```

Q28 Create a Image Map**Ans:**

```

<html>
<body>
<p>Click on the sun or on one of the planets to watch it closer:</p>

<map name="planetmap">
  <area shape="rect" coords="0,0,82,126" alt="Sun" href="sun.htm" />
  <area shape="circle" coords="90,58,3" alt="Mercury" href="mercur.htm"
/>
  <area shape="circle" coords="124,58,8" alt="Venus" href="venus.htm" />
</map>
</body>
</html>

```

Q29 Create a different types of Bullets

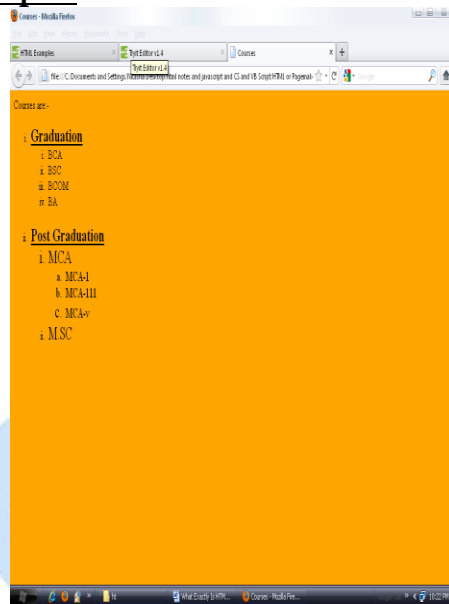
Ans:

```

<html>
<title>Courses</title>
<body bgColor=orange>
Courses are:-
<font size="+2">
<ol type=i>
  <li><b><u>Graduation</u></b>
</font>
<ol type=i>
  <li>BCA
  <li>BSC
  <li>BCOM
  <li>BA
</ol>
<br>
<font size="+2">
<li><b><u>Post Graduation</u></b>
</font>
<font size="+2">
<ol type=i>
  <li>MCA

```

```
<font size="+1">
<ol type=a>
  <li>MCA-1
  <li>MCA-111
  <li>MCA-v
</font>
</ol>
  <li>M.SC
</font>
</ol>
</body>
</html>
```

Output:

Chapter 2

Java Script

Q1 Explain Java Script ?

Ans: HTML pages which we have discussed so far are considered as static HTML pages, these pages layout remains same. But, if you want to create the dynamic page, with the programming support then you have to make use of the scripting languages. Scripting languages are not the complete programming languages then work within the HTML code. The Scripting Language in the market are **VBScript, Javascript** etc..

The Scripting code is written in the **<script>** tag, and the general form is ,
<script language="JavaScript/VBScript">
 Body of the code
</script>

Q2: How to Displaying the Information Using JavaScript Code

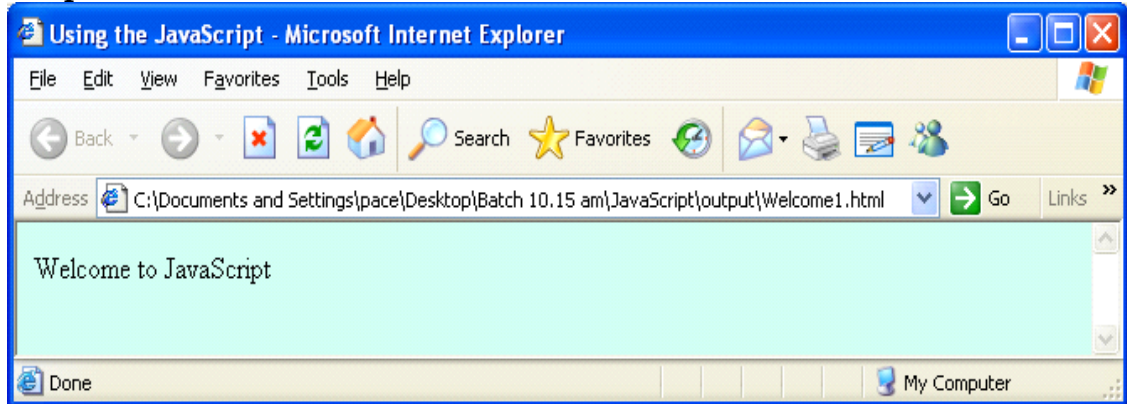
Ans : (a) **document. write()** The document object is used to identify the webpage and it provides the method or the function known as write() to display the information on the browser windows.

Note : JavaScript is case sensitive.

Consider the following code,

```
<html>
<head>
<title>Using the JavaScript </title>
</head>
<body bgcolor="#ccffee">
<script language="JavaScript">
document.write("Welcome to JavaScript");
```

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

Output :

(b) **alert()** This function will display a dialog box on the screen, via, which we can display the information on the screen. The general form is ,
`alert("message");`

Consider the following code ,

```
<html>  
<head>  
<title>Using the JavaScript </title>  
</head>  
<body bgcolor="#ccffee">  
<script language="JavaScript">  
    alert("Welcome to JavaScript");  
</script>  
</body>  
</html>
```

Output :

**Q3: How to Declaring the variables:**

Ans: Variables are the named value holders. They act as a container to store the value. To declare the variable we will use the var keyword.

The general form is ,

```
var variablelist;
```

e.g. var a,b,c;

Types of values :

- 1) **integer values** : The values without the fractional part .e.g. 10,-45 etc...
- 2) **real values** : The values which contains the fractional part e.g. 5.6,7.89 etc...The real values are also known as the floating point values.
- 3) **character values** : It refers to the single character. e.g. 'a','%','3' etc...
- 4) **string values** : It refers to the group of characters. "ajay" etc...

Note that the var type variable is capable of holding any type of value.

Q4: How to Read the Information from the User :

Ans: 1) **prompt()** : This function is used to read the information from the user.

The general form is ,

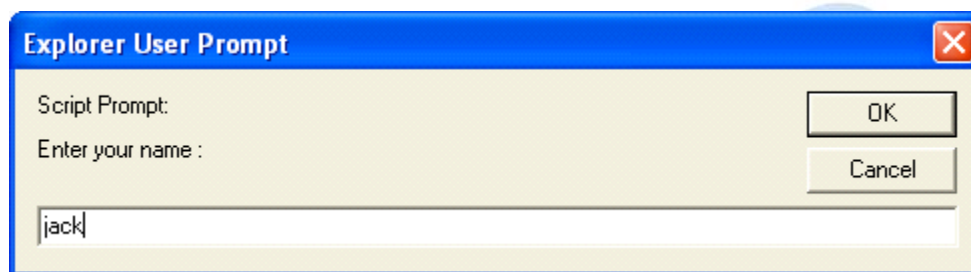
```
prompt("message");
```

Consider the following code ,

```
<html>
<head>
<title>Using JavaScript</title>
</head>
<body bgcolor="#ccffee">
<script language="JavaScript">
var sname;
```

```
sname=prompt("Enter your name :");
document.write("<center><h1>Your Name is : "+sname+"
</h1></center>");
</script>
</body>
</html>
```

Output :



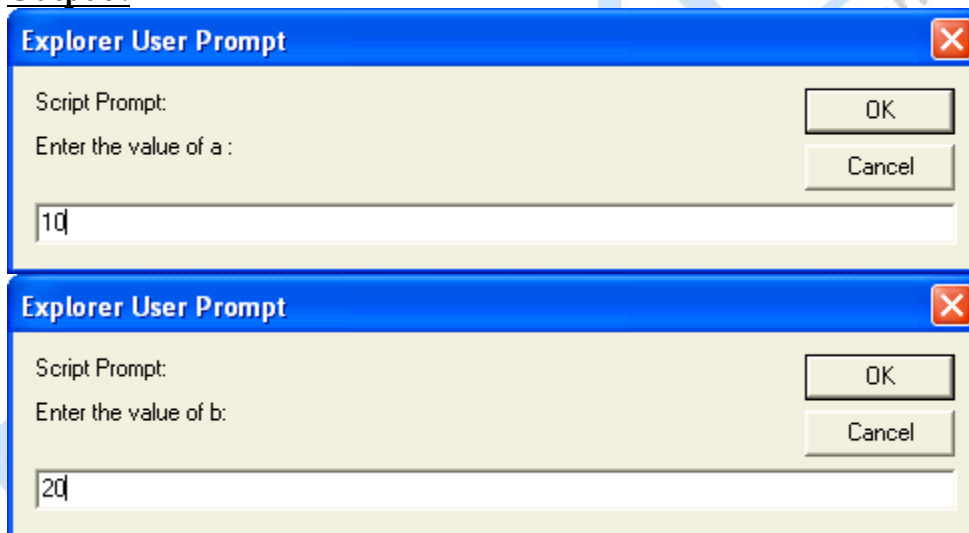
Note : Whatever you enter or input using the `prompt()` will be considered as a **string**.

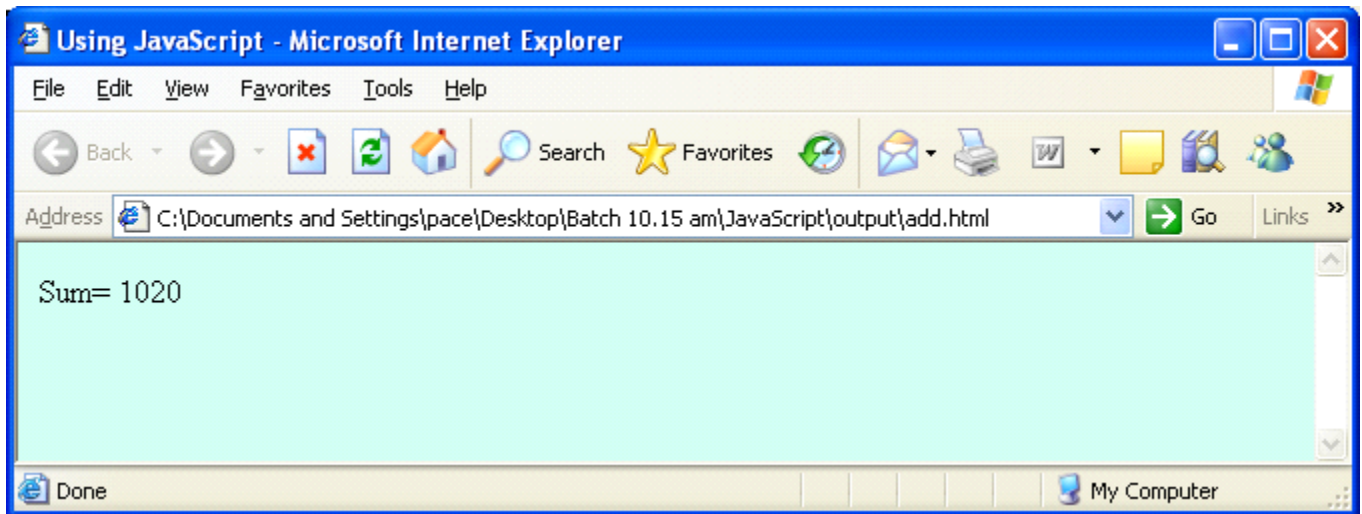
Q5: Write a JavaScript code to add two numbers.

Ans: First code display addition in string format:

```
<html>
<head>
```

```
<title>Using JavaScript</title>
</head>
<body bgcolor="#ccffee">
<script language="JavaScript">
    var a,b,c;
    a=prompt("Enter the value of a :");
    b=prompt("Enter the value of b:");
    c=a+b;
    document.write("Sum= " + c);
</script>
</body>
</html>
```

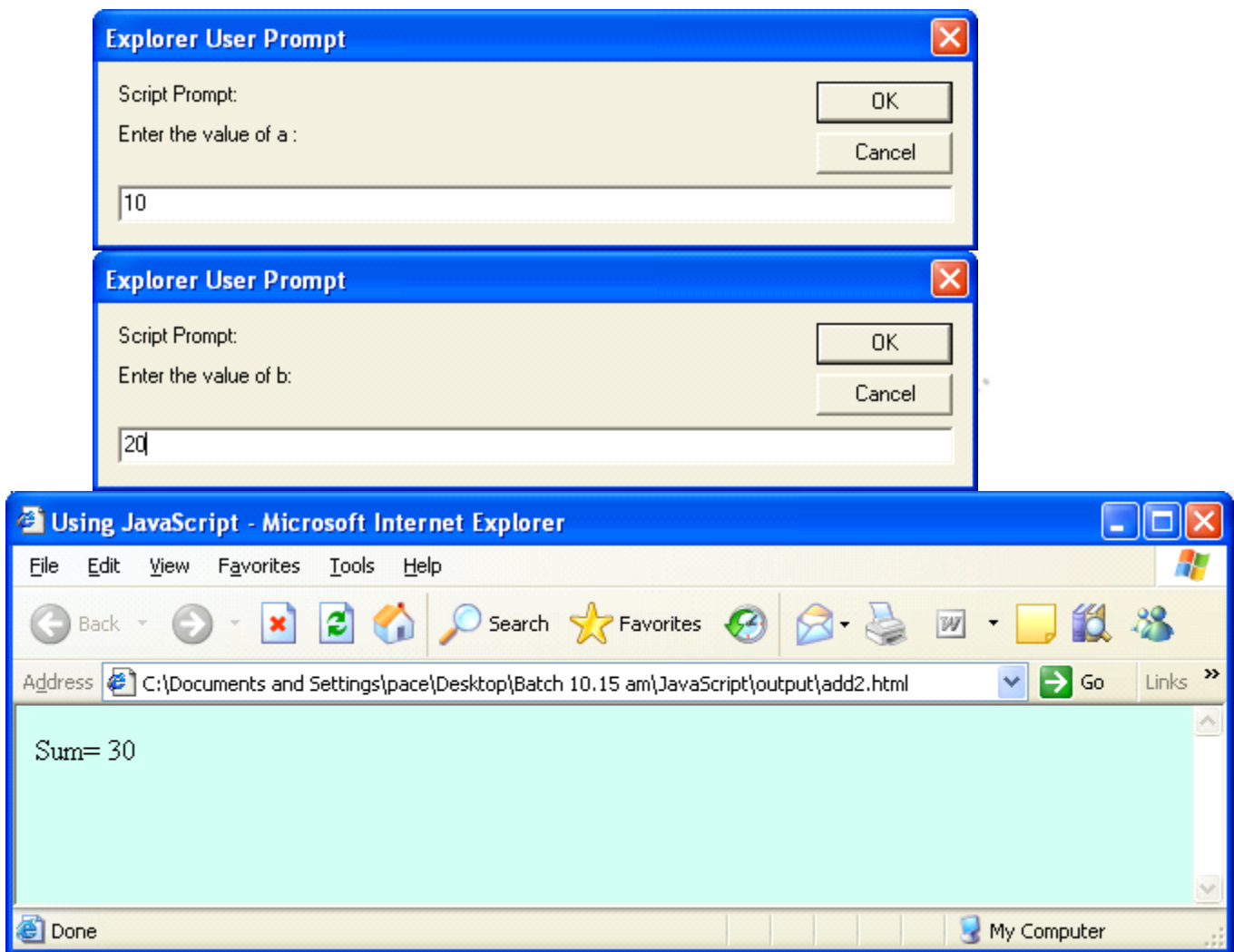
Output :



Now, consider the following program, Which display addition in integer format

```
<html>
<head>
<title>Using JavaScript</title>
</head>
<body bgcolor="#ccffee">
<script language="JavaScript">
    var a,b,c;
    a=prompt("Enter the value of a :");
    b=prompt("Enter the value of b:");
    a=parseInt(a);
    b=parseInt(b);
    c=a+b;
    document.write("Sum= " + c);
</script>
</body>
</html>
```

Note: **parseInt()** : This function is used to convert the string into integer.
parseFloat() : This function is used to convert the string into Decimal number.



Conditional statements

Q6: How to use conditional statements in Java Script?

Ans: The general form is ,
if(condition)
 statement1;
else

statement2;

where,statement1 will get executed when the condition is true and the statment2 will get executed when the condition is false.

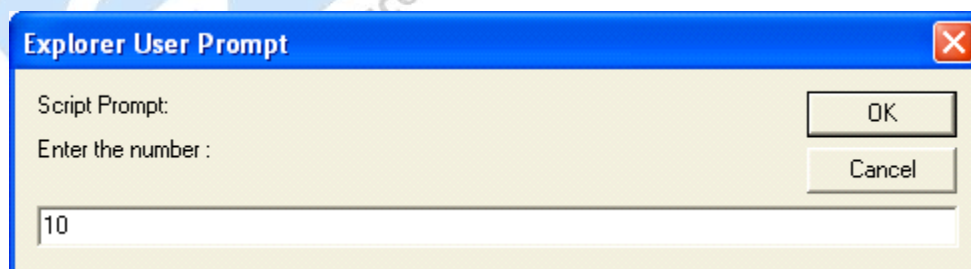
Q7. Write a JavaScript code to check whether the number is even or odd.

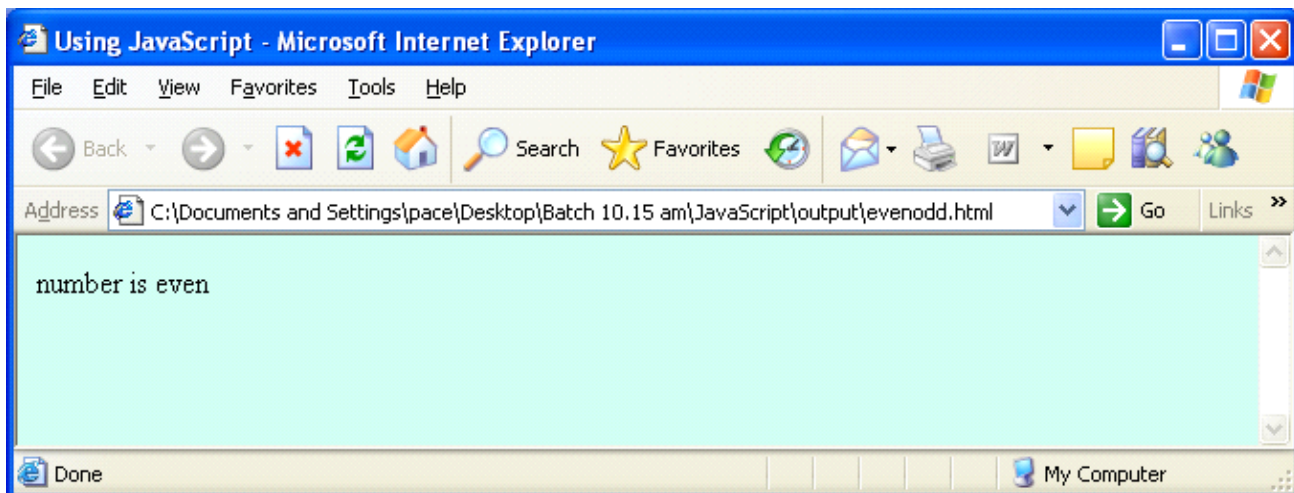
Ans

```
<html>
<head>
<title>Using JavaScript</title>
</head>
<body bgcolor="#ccffee">
<script language="JavaScript">
    var num;
    num=prompt("Enter the number :");
    num=parseInt(num);

    if(num%2==0)
        document.write("number is even");
    else
        document.write("number is odd");
</script>
</body>
</html>
```

Output :





Q8: Write a JavaScript code to find the largest of three numbers.

Ans:

```
<html>
<head>
<title>Using JavaScript</title>
</head>
<body bgcolor="#ccffee">
<script language="JavaScript">
    var a,b,c;
    a=prompt("Enter the value of a :");
    b=prompt("Enter the value of b:");
    c=prompt("Enter the value of c:");
    a=parseInt(a);
    b=parseInt(b);
    c=parseInt(c);
    if(a>b)
        if(a>c)
            document.write("a is largest");
        else
            document.write("c is largest");
    else
        if(b>c)
            document.write("b is largest");
    else
```

```

        document.write("c is largest");
    </script>
</body>
</html>

```

Loops :

Q9: What are the Loop structure use in Java script?

Ans: The term Loops refers to executing a statement or a group of statement till the specified condition is true. Their are three types of loop structure

1. for loop
2. while loop
3. do while loop

(i) for loop : The general form is ,
 for(initialise the variable;condition;increment/ decrement)
 {
 body of loop
 }

Consider the following Example:

```

n=4
1
12
123
1234

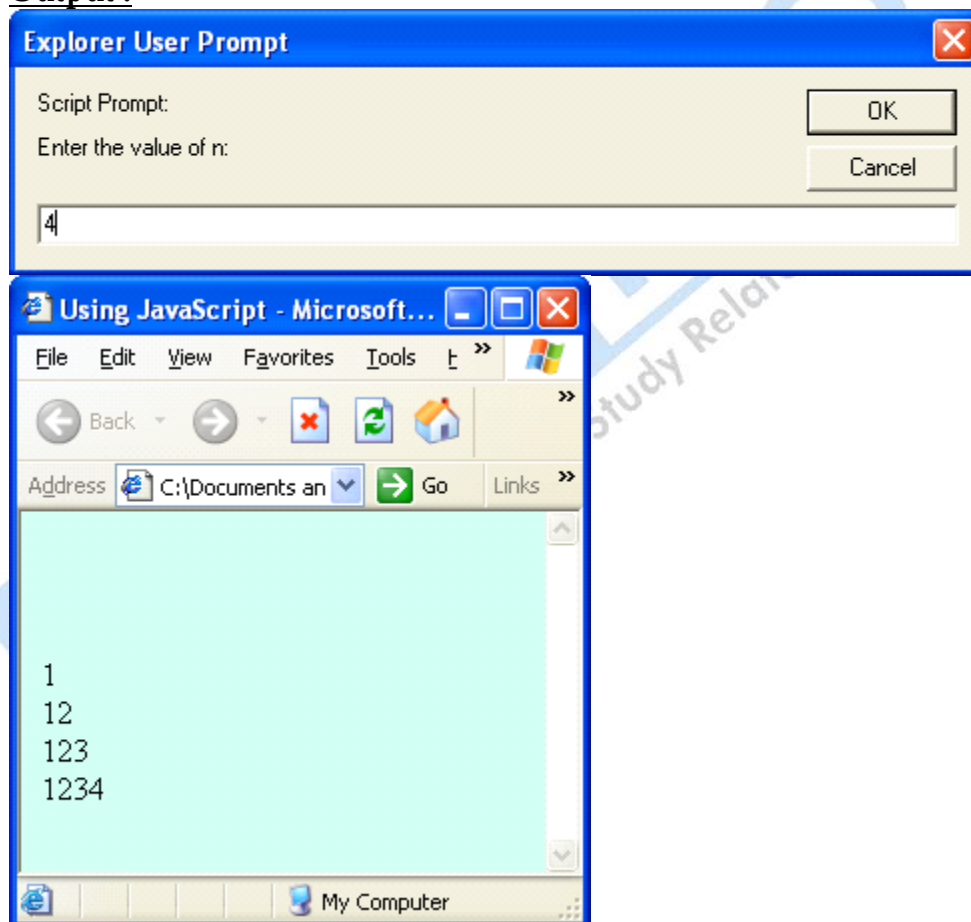
```

```

<html>
<head>
<title>Using JavaScript</title>
</head>
<body bgcolor="#ccffee">
<script language="JavaScript">
var i,j,n;
n=prompt("Enter the value of n:");
document.write("<br><br><br>");
for(i=1;i<=n;i++)
{

```

```
        for(j=1;j<=i;j++)
        {
            document.write(j);
        }
        document.write("<br>");
    }
</script>
</body>
</html>
```

Output :**(ii) while loop**

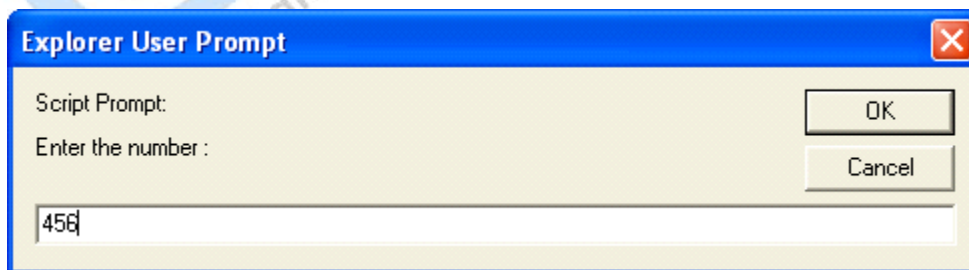
```
/*initialise the variable*/
while(condition)/*test the condition*/
```

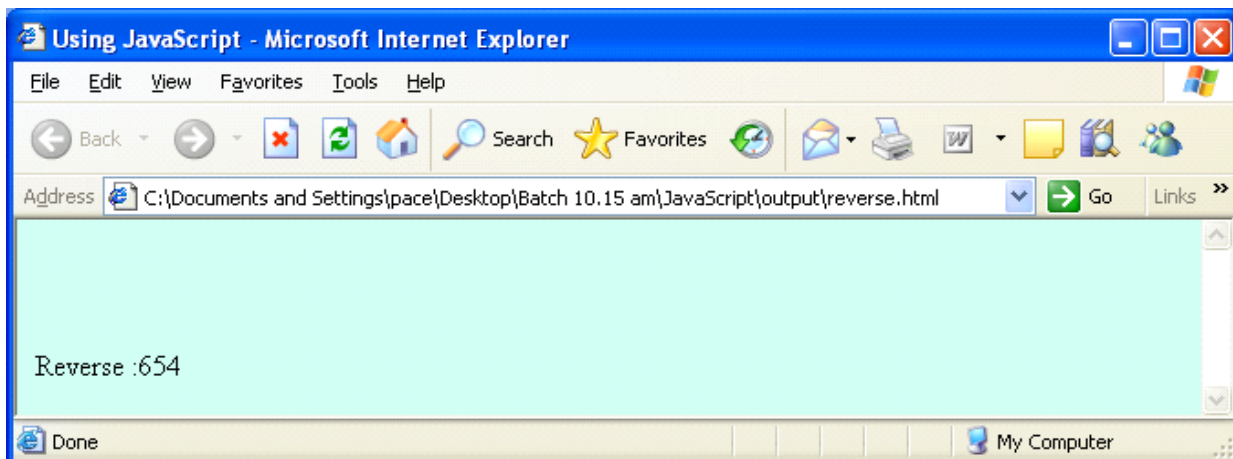
```
{
    /*body of the loop*/
    /*increment or decrement the variable*/
}
```

For example

```
<html>
<head>
<title>Using JavaScript</title>
</head>
<body bgcolor="#ccffee">
<script language="JavaScript">
    var num,rev;
    num=prompt("Enter the number :");
    document.write("<br><br><br>");
    rev=0;
    while(num!=0)
    {
        rev=rev*10+num%10;
        num=parseInt(num/10);
    }
    document.write("Reverse :"+rev);

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

output :



(iii) do...while loop :The general form is ,

```
/*initialise the variable*/
```

```
do
```

```
{
```

```
    /*body of the loop*/
```

```
    /*increment or decrement the variable*/
```

```
}while(condition);
```

For example

```
<html>
```

```
<head>
```

```
<title>Using JavaScript</title>
```

```
</head>
```

```
<body bgcolor="#ccffee">
```

```
<script language="JavaScript">
```

```
    var num,sum,org;
```

```
    num=prompt("Enter the number :");
```

```
    num=parseInt(num);
```

```
    org=num;
```

```
    document.write("<br><br><br>");
```

```
    sum=0;
```

```
    do
```

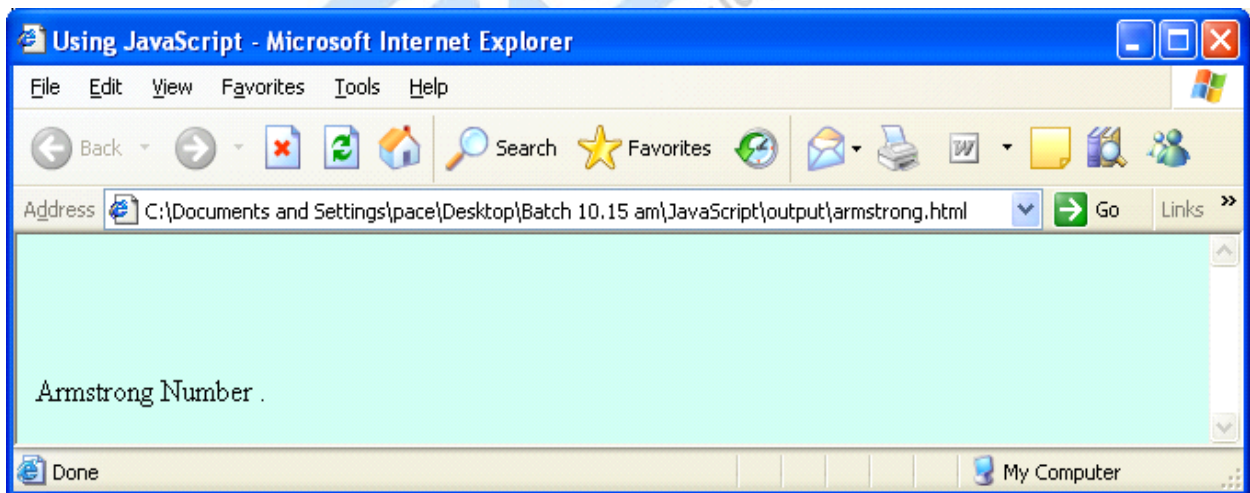
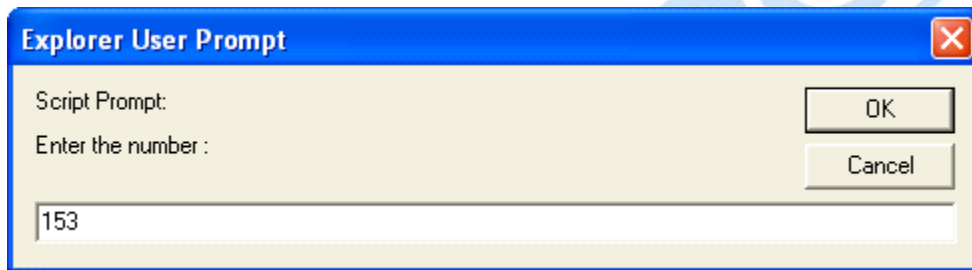
```
    {
```

```
        sum=sum+(num%10)*(num%10)*(num%10);
```

```
        num=parseInt(num/10);
```

```
}while(num!=0);  
if(sum==org)  
    document.write("Armstrong Number .");  
else  
    document.write("Not an armstrong number.");  
</script>  
</body>  
</html>
```

output :



Functions

Q11. Define JavaScript Functions.

Ans: A function will be executed by an event or by a call to the function. To keep the browser from executing a script when the page loads, you can put your script into a function. A function contains code that will be executed by an event or by a call to the function. You may call a function from anywhere within a page (or even from other pages if the function is embedded in an external .js file). Functions can be defined both in the <head> and in the <body> section of a document. However, to assure that a function is read/loaded by the browser before it is called, it could be wise to put functions in the <head> section.

Q12: How to Define a Function

Ans: Function is define like

Syntax

```
function functionname(var1,var2,...,varX)
{
  some code
}
```

The parameters var1, var2, etc. are variables or values passed into the function. The { and the } defines the start and end of the function.

Note: A function with no parameters must include the parentheses () after the function name.

Note: Do not forget about the importance of capitals in JavaScript! The word *function* must be written in lowercase letters, otherwise a JavaScript error occurs! Also note that you must call a function with the exact same capitals as in the function name.

Q13: Define JavaScript Function with Example.

Ans:

Example

```
<html>
<head>
<script type="text/javascript">
function displaymessage()
{
  alert("biyani girls colleges");
}
```

```

</script>
</head>

<body>
<form>
<input type="button" value="Click me!" onclick="displaymessage()" />
</form>
</body>
</html>

```

If the line: `alert("biyani girls colleges")` in the example above had not been put within a function, it would have been executed as soon as the page was loaded. Now, the script is not executed before a user hits the input button. The function `displaymessage()` will be executed if the input button is clicked.

Q14: What is the return Statement?

Ans: The return statement is used to specify the value that is returned from the function.

So, functions that are going to return a value must use the return statement.

The example below returns the product of two numbers (a and b):

Example

```

<html>
<head>
<script type="text/javascript">
function product(a,b)
{
return a*b;
}
</script>
</head>

```

```

<body>
<script type="text/javascript">
document.write(product(4,3));
</script>

```

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

Q15: Explain the procedure of the Lifetime of JavaScript Variables

Ans: If you declare a variable, using "var", within a function, the variable can only be accessed within that function. When you exit the function, the variable is destroyed. These variables are called local variables. You can have local variables with the same name in different functions, because each is recognized only by the function in which it is declared.

If you declare a variable outside a function, all the functions on your page can access it. The lifetime of these variables starts when they are declared, and ends when the page is closed.

Q16: How many types of functions in JavaScript :

Ans: There are three types of functions:

1. When the function will return a value
2. When the function will not return a value

The general form is,

```
function functionname(argument list)
{
    body of the function
}
```

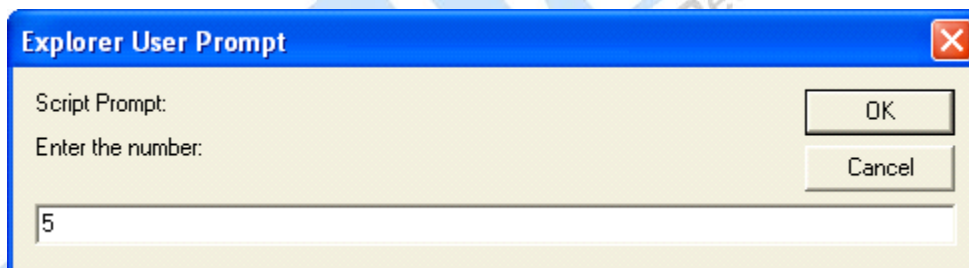
Consider the following program ,

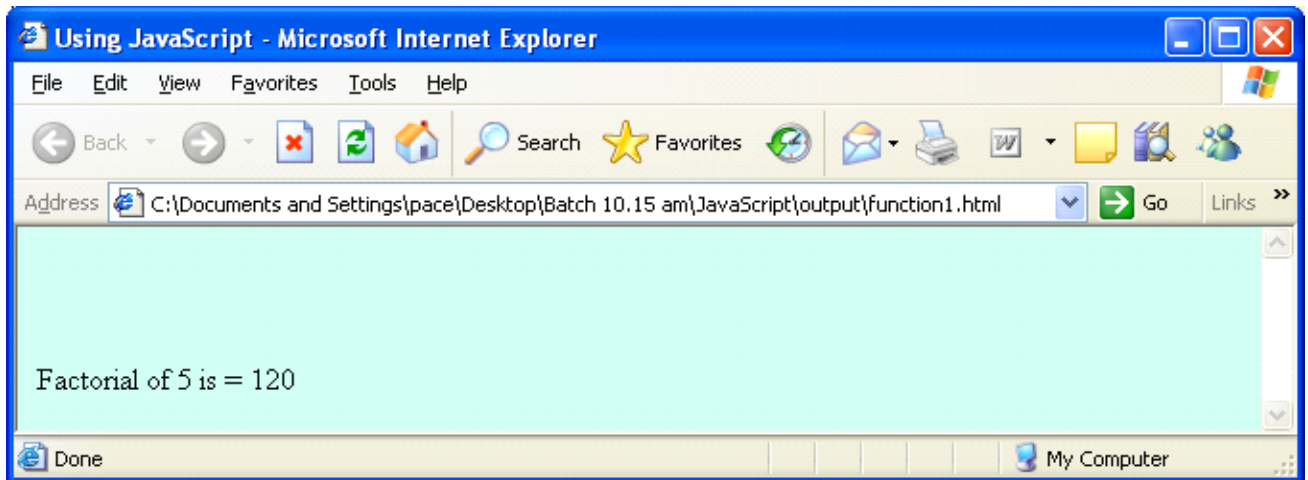
(i) When the function will return a value

```
<html>
<head>
<title>Using JavaScript</title>
<script language="JavaScript">
    function factorial(num)
    {
        var fact,i;
        fact=1;
        for(i=1;i<=num;i++)
```

```
        fact=fact*i;
    return fact;
    }
</script>
</head>
<body bgcolor="#ccffee">
<script language="JavaScript">
    var num,fact;
    num=prompt("Enter the number:");
    document.write("<br><br><br>");
    fact=factorial(num);
    document.write("Factorial of " + num + " is = " + fact);
</script>
</body>
</html>
```

output :



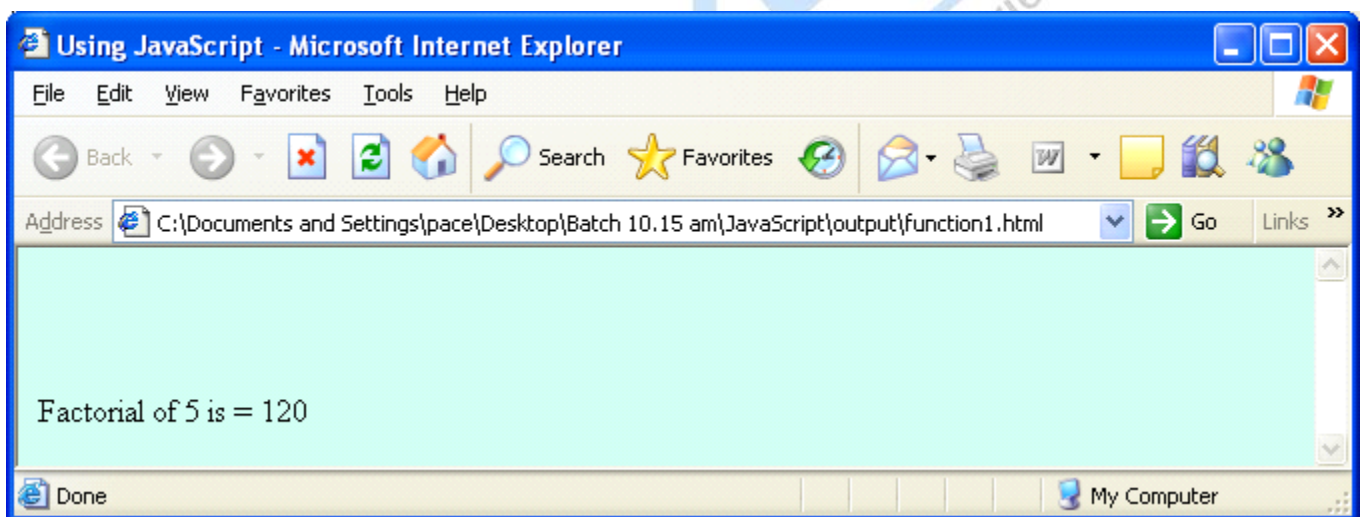
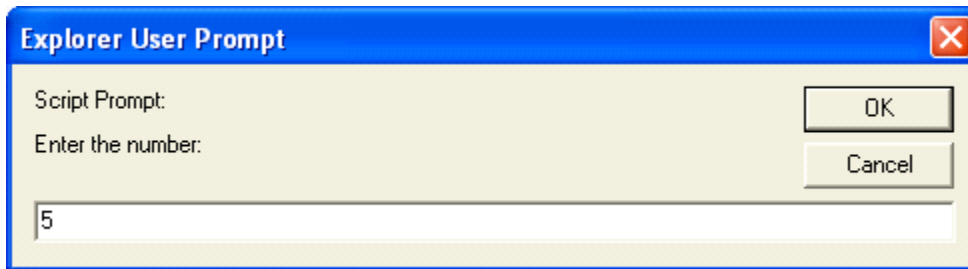


(ii) When the function will not return a value

```
<html>
<head>
<title>Using JavaScript</title>
<script language="JavaScript">
    function factorial(num)
    {
        var fact,i;
        fact=1;
        for(i=1;i<=num;i++)
            fact=fact*i;
        document.write("Factorial of " + num + " is = " + fact);
    }
</script>
</head>
<body bgcolor="#ccffee">
<script language="JavaScript">
    var num;
    num=prompt("Enter the number:");
    document.write("<br><br><br>");
    factorial(num);
```

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

output :



JavaScript Global Properties And Global functions

Q17 Define the JavaScript global properties.

Ans: JavaScript Global Properties

Property	Description
Infinity	A numeric value that represents positive/negative infinity
NaN	"Not-a-Number" value

undefined

Indicates that a variable has not been assigned a value

Q18: Define eval() function and its usage with example.

Ans: Definition eval() :-The eval() function evaluates or executes an argument.If the argument is an expression, eval() evaluates the expression. If the argument is one or more JavaScript statements, eval() executes the statements.

Syntaxeval(*string*)

Parameter	Description
<i>string</i>	A JavaScript expression, variable, statement, or sequence of statements

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

```
eval("x=10;y=20;document.write(x*y)");
document.write("<br />" + eval("2+2"));
document.write("<br />" + eval(x+17));
```

```
</script>
```

The output of the code above will be:

200

4

27

Q19: Define String() function and its usage with example.

Ans: Definition of String():-The String() function converts the value of an object to a string.

Syntax

String(object)

Parameter	Description
object	Required. A JavaScript object

```
<script type="text/javascript">
var test1 = new Boolean(1);
var test2 = new Boolean(0);
var test3 = new Boolean(true);
var test4 = new Boolean(false);
var test5 = new Date();
var test6 = new String("999 888");
var test7 = 12345;
```

```
document.write(String(test1)+ "<br />");
document.write(String(test2)+ "<br />");
document.write(String(test3)+ "<br />");
document.write(String(test4)+ "<br />");
document.write(String(test5)+ "<br />");
document.write(String(test6)+ "<br />");
document.write(String(test7)+ "<br />");
</script>
```

The output of the code above will be:

```
true
false
true
false
```

```
Tue Aug 23 2011 18:38:43 GMT+0530 (India Standard Time)
999 888
12345
```

Q20: Define escape() function and its usage with example.

Ans: Definition of escape():-The escape() function encodes a string. This function makes a string portable, so it can be transmitted across any network to any computer that supports ASCII characters. This function encodes special characters, with the exception of: * @ - _ + . /

Note: Use [unescape\(\)](#) to decode strings.

Syntax

```
escape(string)
```

Parameter	Description
String	Required. The string to be encoded

Example

Encode a string:

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

```
document.write(escape("Need Notes? Visit BISMA!"));
```

```
</script>
```

The output of the code above will be:

```
Need%20Notes%3F%20Visit%20BISMA%21
```

Q21: Define parseFloat() function and its usage with example.

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

Syntax

```
parseFloat(string)
```

Parameter	Description
string	Required. The string to be parsed

Example

Parse different strings:

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

```
document.write(parseFloat("10") + "<br />");
```

```
document.write(parseFloat("10.33") + "<br />");
```

```
document.write(parseFloat("34 45 66") + "<br />");
```

```
document.write(parseFloat(" 60 ") + "<br />");
```

```
document.write(parseFloat("40 years") + "<br />");
```

```
document.write(parseFloat("He was 40") + "<br />");
```

```
</script>
```

The output of the code above will be:

```
10
10.33
34
60
40
NaN
```

Q22: Define parseInt() function and its usage with example.

Ans: Definition of parseInt():-The parseInt() function parses a string and returns an integer. The radix parameter is used to specify which numeral system to be used, for example, a radix of 16 (hexadecimal) indicates that the number in the string should be parsed from a hexadecimal number to a decimal number. If the radix parameter is omitted, JavaScript assumes the following:

- If the string begins with "0x", the radix is 16 (hexadecimal)
- If the string begins with "0", the radix is 8 (octal). This feature is deprecated
- If the string begins with any other value, the radix is 10 (decimal)

Syntax

```
parseInt(string, radix)
```

Parameter	Description
string	Required. The string to be parsed
radix	Optional. A number (from 2 to 36) that represents the numeral system to be used

Example

Parse different strings:

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

```
document.write(parseInt("10") + "<br />");
document.write(parseInt("10.33") + "<br />");
```

```
document.write(parseInt("34 45 66") + "<br />");
document.write(parseInt(" 60 ") + "<br />");
document.write(parseInt("40 years") + "<br />");
document.write(parseInt("He was 40") + "<br />");
```

```
document.write("<br />");
document.write(parseInt("10",10)+ "<br />");
document.write(parseInt("010")+ "<br />");
document.write(parseInt("10",8)+ "<br />");
document.write(parseInt("0x10")+ "<br />");
document.write(parseInt("10",16)+ "<br />");
```

```
</script>
```

The output of the code above will be:

10
10
34
60
40
NaN

10
8
8
16
16

Event Handling :

Q23: Explain Event Handling in Java Script?

Ans: Events as we know are the actions performed by the computer system or the user.

In case of the event handling , we have ,

- (a) Type of event which is to be raised.
- (b) Code which is to be executed when the event is raised.

1. Type of Events ::There are various events which we can handle.

- (i) onLoad() : This will occur when we load the body or other tag.
 - (ii) onClick() : This will occur when we click on the item.
 - (iii) onDbClick() : This will occur when we double click the item.
 - (iv) onFocus() : This will occur when the control receive the focus.
 - (v) onBlur() : This will occur when the control will lost the focus.
 - (vi) onMouseOver() : This will occur when we place the mouse over an object.
 - (vii) onMouseOut() : This will occur when we move the mouse out of an object.
- etc....

2.Code which is to be executed when the event is raised.

Here , we have to write a function which is to be executed when a particular event is raised.

Q24: Design the JavaScript code to display the dialog box when the page is loaded in the memory.

Ans:

```
<html>
<head>
<title>Using Event Handling</title>
<script language="JavaScript">
function wel()
{
    alert("Welcome to JavaScript Event Handling");
}
</script>
</head>
<body onLoad="wel()" bgcolor="red">
</body>
</html>
```

Output :

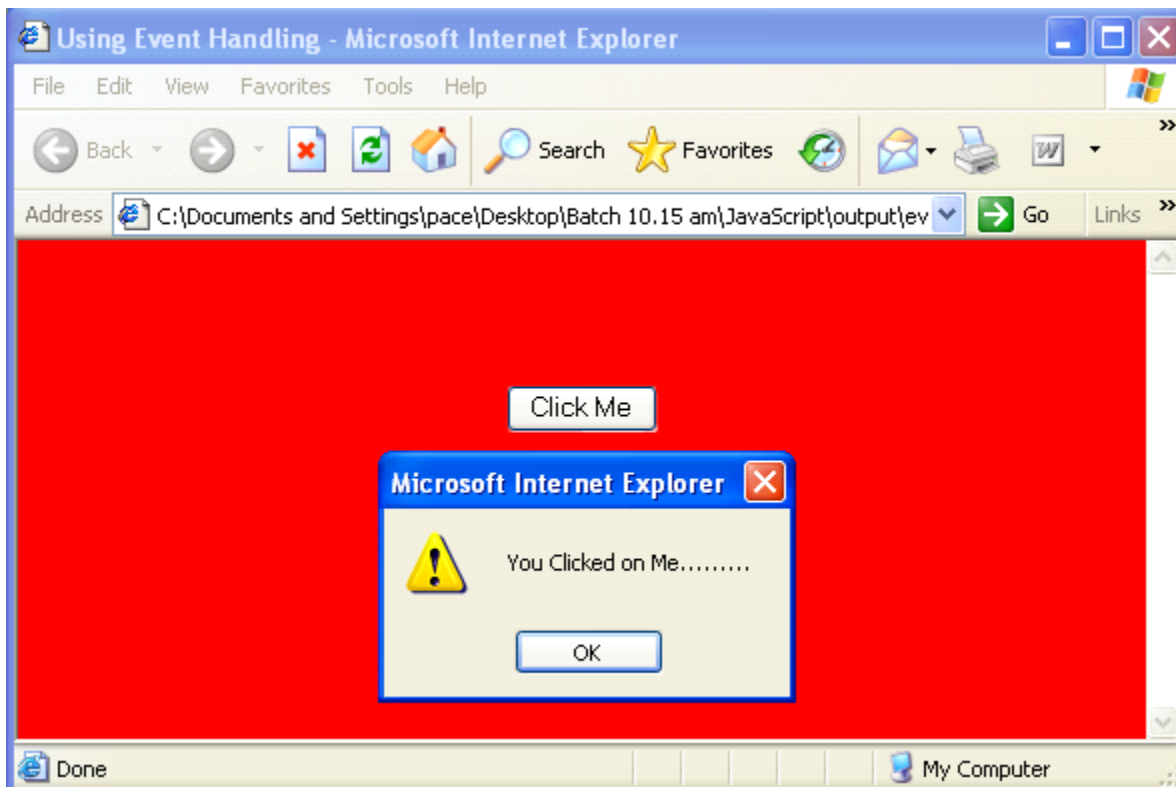


Q25 Design the JavaScript code to create the button and when the user click on it a message is displayed.

Ans:

```
<html>
<head>
<title>Using Event Handling</title>
<script language="JavaScript">
function msg( )
{
    alert("You Clicked on Me.....");
}
</script>
</head>
<body bgcolor="red">
    <br>
    <br>
    <br>
    <center>
    <form name="myform">
        <input type="button" name="cmdbutton" value="Click Me"
onClick="msg()">
    </form>
    </center>
</body>
</html>
```

Output :



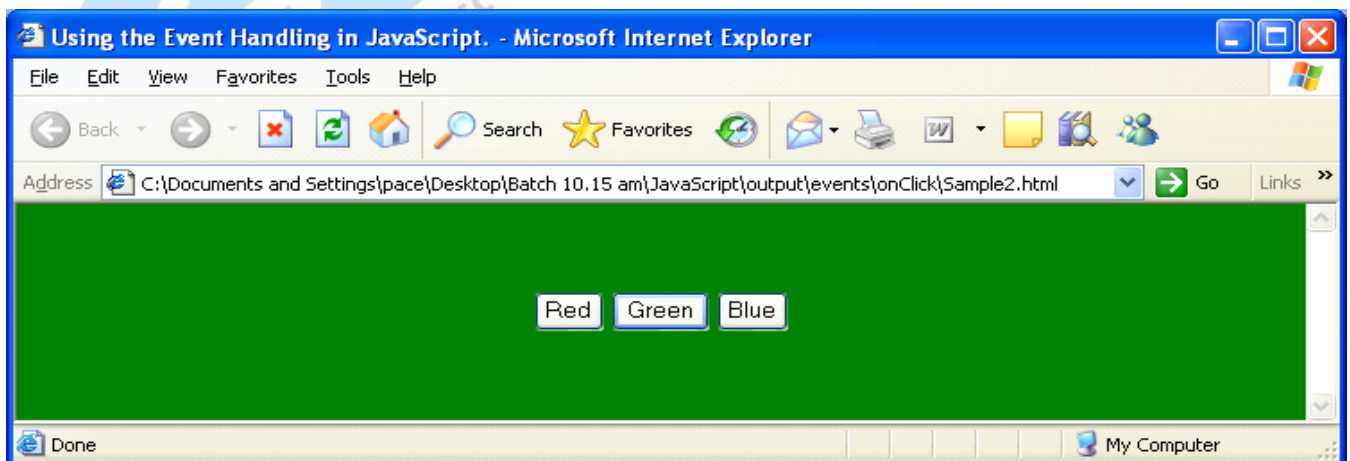
Q26 Design the JavaScript code for creating the application which contains, three buttons, Red, Green and Blue to change the background color accordingly.

Ans

```
<html>
<head>
<title>Using the Event Handling in JavaScript.</title>
<script language="JavaScript">
function changeRed()
{
    document.bgColor="red";
}
function changeGreen()
{
    document.bgColor="green";
```

```
}  
function changeBlue()  
{  
    document.bgColor="blue";  
}  
</script>  
</head>  
<body bgcolor="#ccddee">  
<center>  
<br>  
<br>  
<form name="myform">  
<input type="button" name="cmdred" onClick="changeRed()" value="Red">  
<input type="button" name="cmdgree" onClick="changeGreen()" value="Green">  
<input type="button" name="cmdblue" onClick="changeBlue()" value="Blue">  
</form>  
</center>  
</body>  
</html>
```

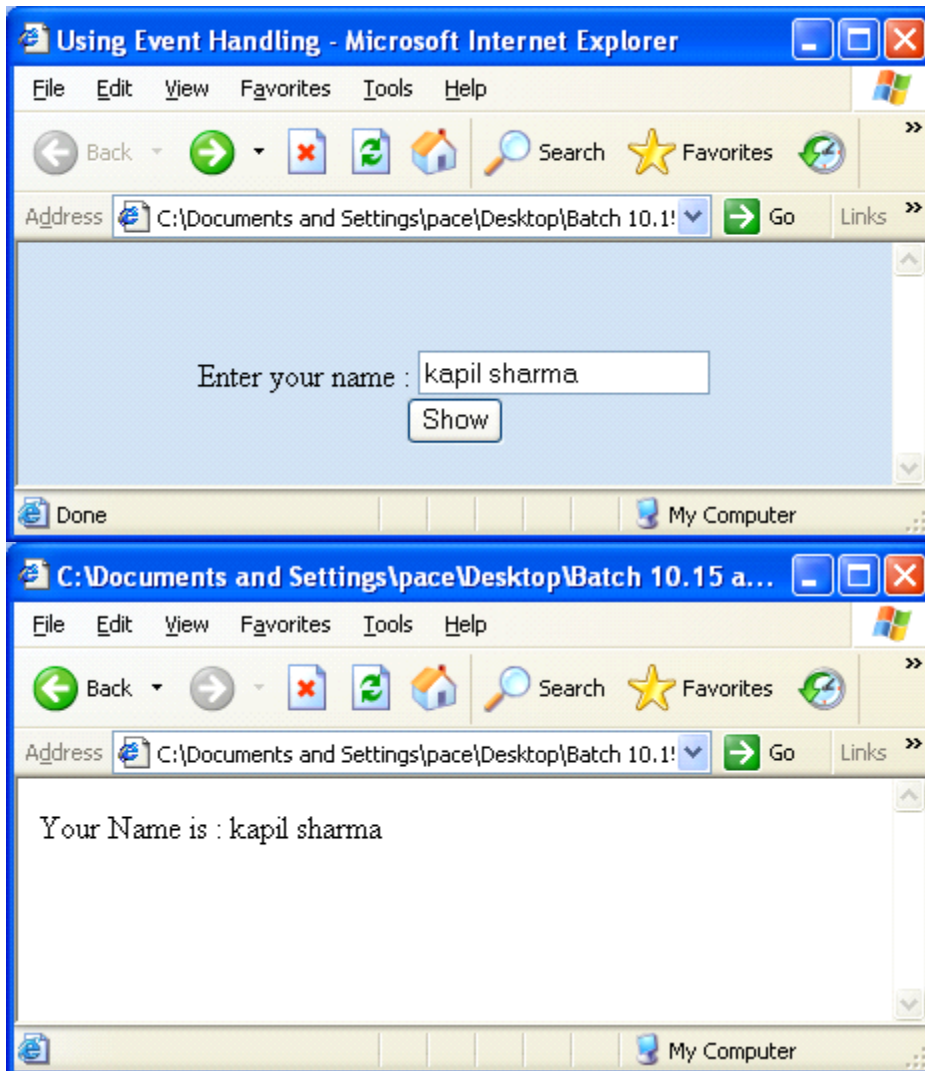
output :



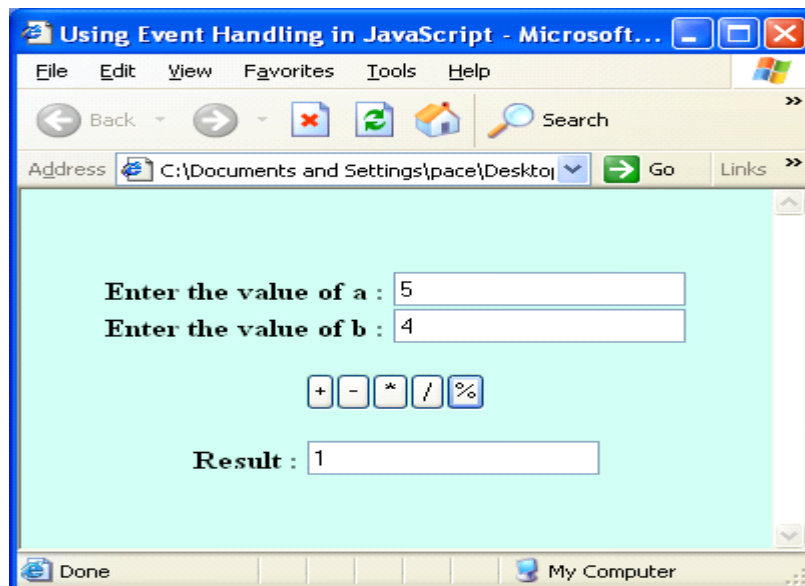
Q27 Design the application to access the value of the text box

Ans

```
<html>
<head>
<title>Using Event Handling</title>
<script language="JavaScript">
function show()
{
    var sname;
    sname=document.myform.txtname.value;
    document.write("Your Name is : " + sname);
}
</script>
</head>
<body bgcolor="#ccddee">
<center>
<br>
<br>
<form name="myform">
Enter your name : <input type="text" name="txtname">
<br>
<input type="button" name="cmdshow" value="Show"
onClick="show()">
</form>
</center>
</body>
</html>
```



Q28: Design the application which contains three textboxes for two number and a result , and perform +,-,*/ and %.



```
<html>
<head>
<title>Using Event Handling in JavaScript</title>
<script language="JavaScript">
function add()
{
    var a,b,c;
    a=document.myform.txta.value;
    b=document.myform.txtb.value;
    a=parseInt(a);
    b=parseInt(b);
    c=a+b;
    document.myform.txtc.value=c;
}
function subtract()
{
    var a,b,c;
    a=document.myform.txta.value;
    b=document.myform.txtb.value;
    a=parseInt(a);
    b=parseInt(b);
    c=a-b;
}
```

```
    document.myform.txtc.value=c;
}
function multiply( )
{
    var a,b,c;
    a=document.myform.txta.value;
    b=document.myform.txtb.value;
    a=parseInt(a);
    b=parseInt(b);
    c=a*b;
    document.myform.txtc.value=c;
}
function divide( )
{
    var a,b,c;

    a=document.myform.txta.value;
    b=document.myform.txtb.value;
    a=parseInt(a);
    b=parseInt(b);
    c=a/b;
    document.myform.txtc.value=c;
}
function mod( )
{
    var a,b,c;
    a=document.myform.txta.value;
    b=document.myform.txtb.value;
    a=parseInt(a);
    b=parseInt(b);
    c=a%b;
    document.myform.txtc.value=c;
}
</script>
</head>
<body bgcolor="#ccffee">
<center>
```

```
<br>
<br>
<form name="myform">
<b> Enter the value of a : </b><input type="text" name="txta"><br>
<b> Enter the value of b : </b><input type="text" name="txtb"><br>
<br>
<input type="button" name="cmdplus" value="+" onClick="add(
)><input type="button" name="cmdminus" value="-" onClick="subtract(
)><input type="button" name="cmdmult" value="*" onClick="multiply(
)><input type="button" name="cmddiv" value="/" onClick="divide(
)><input type="button" name="cmdmod" value="%" onClick="mod(
)><br>
<br>
<b> Result : </b><input type="text" name="txtc">
<br>
</form>
</center>
</body>
</html>
```

Chapter 3

Arrays

Q1: What is an Array?

Ans: An array is a special variable, which can hold more than one value, at a time.

If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
var car1="Saab";
var car2="Volvo";
var car3="BMW";
```

However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?

The best solution here is to use an array!

An array can hold all your variable values under a single name. And you can access the values by referring to the array name.

Each element in the array has its own ID so that it can be easily accessed.

Q2 How to create an Array?

Ans: An array can be defined in three ways.

The following code creates an Array object called myCars:

1:

```
var myCars=new Array(); // regular array (add an optional integer  
myCars[0]="Saab"; // argument to control array's size)  
myCars[1]="Volvo";  
myCars[2]="BMW";
```

2:

```
var myCars=new Array("Saab","Volvo","BMW"); // condensed array
```

3:

```
var myCars=["Saab","Volvo","BMW"]; // literal array
```

Note: If you specify numbers or true/false values inside the array then the variable type will be Number or Boolean, instead of String.

Q3. How to Access an Array?

Ans: We can refer to a particular element in an array by referring to the name of the array and the index number. The index number starts at 0.

The following code line:

```
document.write(myCars[0]);
```

will result in the following output:

Saab

Q4. How to use array in JavaScript?

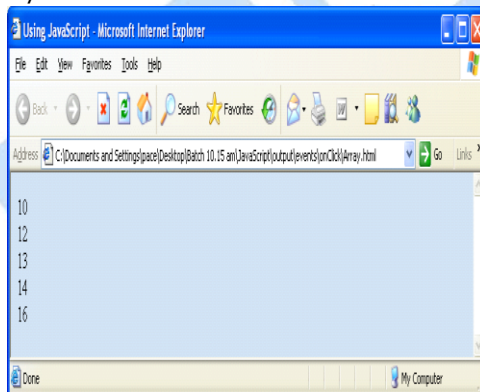
Ans: Arrays : An array is the collection of elements. In JavaScript we can declare the array by,

```
arrayname=new Array(size of array);
```

e.g. x=new Array(5);

Consider the following code,

```
<html>
<head>
<title>Using JavaScript</title>
</head>
<body bgcolor="#ccddee">
<script language="JavaScript">
x=new Array(5);
x[0]=10;
x[1]=12;
x[2]=13;
x[3]=14;
x[4]=16;
var i;
for(i=0;i<5;i++)
{
    document.write(x[i]+"<br>");
}
</script>
</body>
</html>
```



Q5: How to modify values in an Array

Ans: To modify a value in an existing array, just add a new value to the array with a specified index number:

```
myCars[0]="Opel";
```

Now, the following code line:
`document.write(myCars[0]);`
 will result in the following output:
 Opel

Object Properties and Array Object Methods

Q6: Define Array Object Properties and Array Object Methods

Ans: Array Object Properties

Property	Description
<u>constructor</u>	Returns the function that created the Array object's prototype
<u>length</u>	Sets or returns the number of elements in an array

Array Object Methods

Method	Description
<u>concat()</u>	Joins two or more arrays, and returns a copy of the joined arrays
<u>join()</u>	Joins all elements of an array into a string
<u>pop()</u>	Removes the last element of an array, and returns that element
<u>push()</u>	Adds new elements to the end of an array, and returns the new length
<u>reverse()</u>	Reverses the order of the elements in an array
<u>shift()</u>	Removes the first element of an array, and returns that element
<u>sort()</u>	Sorts the elements of an array
<u>toString()</u>	Converts an array to a string, and returns the result
<u>unshift()</u>	Adds new elements to the beginning of an array, and returns the new length
<u>valueOf()</u>	Returns the primitive value of an array

Q7: Define constructor and its usage with example.

Ans: Definition of [constructor](#):- The constructor property returns the function that created the array object's prototype.

Syntax

`array.constructor`

Example

Return the function that created the Array object's prototype:

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

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.constructor);
```

```
</script>
```

The output of the code above will be:

```
function Array() { [native code] }
```

Q8: Define length and its usage with example.

Ans: Definition of length:-The length property sets or returns the number of elements in an array.

Syntax

```
array.length
```

Example

Return and set the length of an array:

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

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write("Original length: " + fruits.length);
document.write("<br />");
fruits.length=5;
document.write("New length: " + fruits.length);
```

```
</script>
```

The output of the code above will be:

Original length: 4

New length: 5

Q9: Define concat and its usage with example.

Ans: Definition of concat:-The concat() method is used to join two or more arrays.

This method does not change the existing arrays, it only returns a copy of the joined arrays.

Syntax

```
array.concat(array2, array3, ..., arrayX);
```

Parameter	Description
array2, array3, ..., arrayX	Required. The arrays to be joined

Example 2

Join three arrays:

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

```
var parents = ["Jani", "Tove"];
var brothers = ["Stale", "Kai Jim", "Borge"];
var children = ["Cecilie", "Lone"];
var family = parents.concat(brothers, children);
document.write(family);
```

```
</script>
```

The output of the code above will be:

Jani,Tove,Stale,Kai Jim,Borge,Cecilie,Lone

Q10: Define join and its usage with example.

Ans: Definition of join: The join() method joins all elements of an array into a string, and returns the string. The elements will be separated by a specified separator. The default separator is comma (,).

Syntax

```
array.join(separator)
```

Parameter	Description
separator	Optional. The separator to be used. If omitted, the elements are separated with a comma

Example

Join all elements of an array into a string:

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

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.join() + "<br />");
document.write(fruits.join("+") + "<br />");
document.write(fruits.join(" and "));
```

```
</script>
```

The output of the code above will be:
 Banana,Orange,Apple,Mango
 Banana+Orange+Apple+Mango
 Banana and Orange and Apple and Mango

Q11: Define POP() and its usage with example.

Ans: Definition of POP():-The pop() method removes the last element of an array, and returns that element.

Note: This method changes the length of an array!

Syntax

```
Array.pop()
```

Example

Remove the last element of an array (this will also change the length of the array):

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

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.pop() + "<br />");
document.write(fruits + "<br />");
document.write(fruits.pop() + "<br />");
document.write(fruits);
```

```
</script>
```

The output of the code above will be:

Mango
 Banana,Orange,Apple
 Apple
 Banana,Orange

Q12: Define Push() and its usage with example.

Ans: Definition of Push():-The push() method adds new elements to the end of an array, and returns the new length.

Note: This method changes the length of an array!

Syntax

```
array.push(element1, element2, ..., elementX)
```

Parameter	Description
element1, element2, ..., elementX	Required. The element(s) to add to the end of the array

Example

Add new elements to the end of an array, and return the new length:

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

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.push("Kiwi") + "<br />");
document.write(fruits.push("Lemon","Pineapple") + "<br />");
document.write(fruits);
```

```
</script>
```

The output of the code above will be:

5

7

Banana,Orange,Apple,Mango,Kiwi,Lemon,Pineapple

Q13 : Define Reverse and its usage with example.

Ans: Definition of Reverse:-The reverse() method reverses the order of the elements in an array (makes the last element first, and the first element last).

Note: This method changes the original array!

Syntax

```
array.reverse()
```

Example

Reverse the order of the elements in an array:

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

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.reverse());
```

```
</script>
```

The output of the code above will be:

Mango,Apple,Orange,Banana

Q14: Define sort and its usage with example.

Ans: Definition of Sort:- The sort() method sorts the elements of an array.

Note: This method changes the original array!

Syntax

```
array.sort(sortfunc)
```

Parameter	Description
sortfunc	Optional. A function that defines the sort order

Example 1

Sort an array (alphabetically and ascending):

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

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.sort());
```

```
</script>
```

The output of the code above will be:

Apple,Banana,Mango,Orange

Sort numbers (numerically and ascending):

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

```
function sortNumber(a,b)
{
return a - b;
}
```

```
var n = ["10", "5", "40", "25", "100", "1"];
document.write(n.sort(sortNumber));
```

```
</script>
```

The output of the code above will be:

1,5,10,25,40,100

Sort numbers (numerically and descending):

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

```
function sortNumber(a,b)
{
return b - a;
}
```

```
var n = ["10", "5", "40", "25", "100", "1"];
document.write(n.sort(sortNumber));
```

```
</script>
```

The output of the code above will be:

100,40,25,10,5,1

Q15: Define ToString and its usage with example.

Ans: Definition of ToString:- The toString() method converts an array to a string and returns the result.

Note: The returned string will separate the elements in the array with commas.

Syntax

```
array.toString()
```

Example

Convert an array to a string:

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

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.toString());
```

```
</script>
```

The output of the code above will be:

Banana,Orange,Apple,Mango

Q16: Define Valueof and its usage with example.

Ans: Definition of Valueof:-The valueOf() method returns the primitive value of an array.

Note: This method is usually called automatically by JavaScript behind the scenes, and not explicitly in code.

Syntax

`array.valueOf()`

Example

Return the primitive value of an array:

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

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.valueOf());
```

```
</script>
```

The output of the code above will be:

Banana,Orange,Apple,Mango

Q17: Define shift and its usage with example.

Ans: Definition of Shift:-The shift() method removes the first element of an array, and returns that element.

Note: This method changes the length of an array!

Syntax

`array.shift()`

Example

Remove the first element of an array (this will also change the length of the array):

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

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.shift() + "<br />");
document.write(fruits + "<br />");
document.write(fruits.shift() + "<br />");
document.write(fruits);
```

```
</script>
```

The output of the code above will be:

```
Banana
Orange,Apple,Mango
Orange
Apple,Mango
```

Q18: Define unshift and its usage with example.

Ans: Definition of Unshift:-

The unshift() method adds new elements to the beginning of an array, and returns the new length.

Note: This method changes the length of an array!

Syntax

```
array.unshift(element1,element2, ..., elementX)
```

Parameter	Description
element1,element2, ..., elementX	Required. The element(s) to add to the beginning of the array

Example

Add new elements to the beginning of an array, and return the new length:

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

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.unshift("Kiwi") + "<br />");
document.write(fruits.unshift("Lemon","Pineapple") + "<br />");
document.write(fruits);
```

```
</script>
```

The output of the code above will be:

```
5
7
```

Lemon,Pineapple,Kiwi,Banana,Orange,Apple,Mango

Chapter 4

Number Object

Q1 Define Number Object.

Ans: The Number object is an object wrapper for primitive numeric values. Number objects are created with new Number().

Syntax

Var num = new Number(value);

Note: If the value parameter cannot be converted into a number, it returns NaN (Not-a-Number).

Q2. Define number object properties and methods list.

Ans:

Number Object Properties

Property	Description
constructor	Returns the function that created the Number object's prototype
MAX_VALUE	Returns the largest number possible in JavaScript
MIN_VALUE	Returns the smallest number possible in JavaScript
NEGATIVE_INFINITY	Represents negative infinity (returned on overflow)
POSITIVE_INFINITY	Represents infinity (returned on overflow)
prototype	Allows you to add properties and methods to an object

Number Object Methods

Method	Description
toExponential(x)	Converts a number into an exponential notation
toFixed(x)	Formats a number with x numbers of digits after the decimal point
toPrecision(x)	Formats a number to x length

toString()	Converts a Number object to a string
valueOf()	Returns the primitive value of a Number object

Q3: Define MAX_VALUE and its usage with example.

Ans: Definition of MAX_VALUE:-The MAX_VALUE property returns the largest number possible in JavaScript.

This static property has a value of 1.7976931348623157e+308.

Note: Numbers larger than this are represented as infinity.

Syntax

Number.MAX_VALUE

Example

Return the largest number possible in JavaScript.

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

```
document.write(Number.MAX_VALUE);
```

```
</script>
```

The output of the code above will be:

1.7976931348623157e+308

Q4: Define MIN_VALUE and its usage with example.

Ans: Definition of MIN_VALUE:-The MIN_VALUE property returns the smallest number possible in JavaScript. This static property has a value of 5e-324.

Note: Numbers smaller than this are converted to 0.

Syntax

Number.MIN_VALUE

Example

Return the smallest number possible in JavaScript:

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

```
document.write(Number.MIN_VALUE);
```

```
</script>
```

The output of the code above will be:

5e-324

Q5: Define NEGATIVE_INFINITY and its usage with example.

Ans: Definition NEGATIVE_INFINITY:The NEGATIVE_INFINITY property represents negative infinity, returned on overflow.

Syntax

```
Number.NEGATIVE_INFINITY;
```

Example

Create overflow:

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

```
var x=(-Number.MAX_VALUE)*2;
if (x==Number.NEGATIVE_INFINITY)
{
document.write(x);
}
```

```
</script>
```

The output of the code above will be:

-Infinity

Q6: Define POSITIVE_INFINITY and its usage with example.

Ans: POSITIVE_INFINITY:The POSITIVE_INFINITY property represents infinity, returned on overflow.

Syntax

```
Number.POSITIVE_INFINITY;
```

Example

Create overflow:

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

```
var x=(Number.MAX_VALUE)*2;
if (x==Number.POSITIVE_INFINITY)
{
document.write(x);
}
```

```
</script>
```

The output of the code above will be:

Infinity

Q7 Define prototype property and its usage with example.

Ans: **Definition prototype property:** The prototype property allows you to add properties and methods to an object.

Note: Prototype is a global property which is available with almost all JavaScript objects.

Syntax

object.prototype.name=value

Example

Use the prototype property to add a property to an object:

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

```
function employee(name,jobtitle,born)
{
this.name=name;
this.jobtitle=jobtitle;
this.born=born;
}
```

```
var fred=new employee("Fred Flintstone","Caveman",1970);
employee.prototype.salary=null;
fred.salary=20000;
```

```
document.write(fred.salary);
```

```
</script>
```

The output of the code above will be:

20000

Q8 Define toExponential() and its usage with example.

Ans: The toExponential() method converts a number into an exponential notation.

Syntax

number.toExponential(x)

Parameter	Description
x	Optional. An integer between 0 and 20 representing the number of digits in the notation after the decimal point. If omitted, it is set to as many digits as necessary to represent the value

Example

Convert a number into an exponential notation:

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

```
var num = new Number(13.3714);
document.write(num.toExponential()+"<br />");
document.write(num.toExponential(2)+"<br />");
document.write(num.toExponential(3)+"<br />");
document.write(num.toExponential(10));
```

```
</script>
```

The output of the code above will be:

```
1.33714e+1
1.34e+1
1.337e+1
1.3371400000e+1
```

Q9 Define toFixed() and its usage with example.

Ans: Definition toFixed():The toFixed() method formats a number to use a specified number of trailing decimals.

The number is rounded up, and nulls are added after the decimal point (if needed), to create the desired decimal length.

Syntax

number.toFixed(x)

Parameter	Description
x	Optional. The number of digits after the decimal point. Default is 0 (no digits after the decimal point)

Example

```

Format a number:
<script type="text/javascript">

var num = new Number(13.3714);
document.write(num.toFixed()+"<br />");
document.write(num.toFixed(1)+"<br />");
document.write(num.toFixed(3)+"<br />");
document.write(num.toFixed(10));

</script>

```

The output of the code above will be:

```

13
13.4
13.371
13.3714000000

```

Q10 Define toPrecision() and its usage with example.

Ans: Definition toPrecision():-The toPrecision() method formats a number to a specified length.

A decimal point and nulls are added (if needed), to create the specified length.

Syntax

number.toPrecision(x)

Parameter	Description
X	Optional. The number of digits. If omitted, it returns the entire number (without any formatting)

Example

Format a number to a specified length:

```

<script type="text/javascript">

var num = new Number(13.3714);
document.write(num.toPrecision()+"<br />");
document.write(num.toPrecision(2)+"<br />");
document.write(num.toPrecision(3)+"<br />");

```

```
document.write(num.toPrecision(10));
```

```
</script>
```

The output of the code above will be:

```
13.3714
13
13.4
13.37140000
```

Q11 Define toString() and its usage with example.

Ans: Definition toString():-The toString() method converts a Number object to a string.

Note: This method is automatically called by JavaScript whenever a Number object needs to be displayed as a string.

Syntax

```
number.toString(radix)
```

Parameter	Description
radix	Optional. Which base to use for representing a numeric value. Must be an integer between 2 and 36. <ul style="list-style-type: none"> • 2 - The number will show as a binary value • 8 - The number will show as an octal value • 16 - The number will show as a hexadecimal value

Example

Convert a number to a string, with different bases:

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

```
var num=new Number(15);
document.write(num.toString()+"<br />");
document.write(num.toString(2)+"<br />");
document.write(num.toString(8)+"<br />");
document.write(num.toString(16)+"<br />");
```

```
</script>
```

The output of the code above will be:

```
15
```

1111
17
f

Q12 Define valueOf() and its usage with example.

Ans: Definition valueOf():-The valueOf() method returns the primitive value of a Number object.

Note: This method is usually called automatically by JavaScript behind the scenes, and not explicitly in code.

Syntax

number.valueOf()

Example

Return the primitive value of a Number object:

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

```
var num=new Number(15);  
document.write(num.valueOf());
```

```
</script>
```

The output of the code above will be:

15

Chapter 5

Object Oriented Programming

Q1. Define Object Oriented Programming in Java Script?

Ans: JavaScript is an Object Oriented Programming (OOP) language. An OOP language allows you to define your own objects and make your own variable types.

JavaScript is an Object Oriented Programming (OOP) language. An OOP language allows you to define your own objects and make your own variable types.

Note that an object is just a special kind of data. An object has properties and methods.

Q2. Define the Properties of an object?

Ans: Properties are the values associated with an object. In the following example we are using the length property of the String object to return the number of characters in a string:

```
<script type="text/javascript">
var name="biyani's!";
document.write(name.length);
</script>
```

The output of the code above will be:

9

Q3. What are the Methods of javascript?

Ans: Methods are the actions that can be performed on objects. In the following example we are using the toUpperCase() method of the String object to display a text in uppercase letters:

```
<script type="text/javascript">
var name="biyani";
```

```
document.write(name.toUpperCase());  
</script>
```

The output of the code above will be:

BIYANI

Q4: Define String object in JavaScript.

Ans: The String object is used to manipulate a stored piece of text. The following example uses the length property of the String object to find the length of a string:

```
var txt="Hello world!";  
document.write(txt.length);
```

The code above will result in the following output:

12

The following example uses the toUpperCase() method of the String object to convert a string to uppercase letters:

```
var txt="Hello world!";  
document.write(txt.toUpperCase());
```

The code above will result in the following output:

HELLO WORLD!

Q5: How to Create a Date Object in javascript?

Ans: The Date object is used to work with dates and times. Date objects are created with the Date() constructor. There are four ways of instantiating a date:

```
New Date() // current date and time  
new Date(milliseconds) // milliseconds since  
1970/01/01  
new Date(dateString)  
new Date(year, month, day, hours, minutes, seconds,  
milliseconds)
```

Most parameters above are optional. Not specifying, causes 0 to be passed in.

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

All dates are calculated in milliseconds from 01 January, 1970 00:00:00 Universal Time (UTC) with a day containing 86,400,000 milliseconds.

Some examples of instantiating a date:

```
var today = new Date()
var d1 = new Date("October 13, 1975
11:13:00")
var d2 = new Date(79,5,24)
var d3 = new Date(79,5,24,11,33,0)
```

Q6: How to Set Dates in JavaScript?

Ans: We can easily manipulate the date by using the methods available for the Date object.

In the example below we set a Date object to a specific date (14th January 2010):

```
var myDate=new Date();
myDate.setFullYear(2010,0,14);
```

And in the following example we set a Date object to be 5 days into the future:

```
var myDate=new Date();
myDate.setDate(myDate.getDate()+5);
```

Note: If adding five days to a date shifts the month or year, the changes are handled automatically by the Date object itself!

Q7: How to Compare Two Dates in JavaScript ?

Ans: The Date object is also used to compare two dates. The following example compares today's date with the 14th January 2100:

```
Var x=new Date();
x.setFullYear(2100,0,14);
var today = new Date();
if (x>today)
{
  alert("Today is before 14th January
2100");
}
else
{
  alert("Today is after 14th January 2100");
}
```

Q8: How to Create a Boolean Object in JavaScript?

Ans: The Boolean object represents two values: "true" or "false". The following code creates a Boolean object called myBoolean:

```
var myBoolean=new Boolean();
```

If the Boolean object has no initial value, or if the passed value is one of the following:

- 0
- -0
- null
- ""
- false
- undefined
- NaN

the object it is set to false. For any other value it is set to true (even with the string "false")!

Q9: How to Create a Math Object in JavaScript?

Ans: The Math object allows you to perform mathematical tasks. The Math object includes several mathematical constants and methods.

Syntax for using properties/methods of Math:

```
var x=Math.PI;
var y=Math.sqrt(16);
```

Note: Math is not a constructor. All properties and methods of Math can be called by using Math as an object without creating it.

Q10: What is Mathematical Constants in javascript

Ans: JavaScript provides eight mathematical constants that can be accessed from the Math object. These are: E, PI, square root of 2, square root of 1/2, natural log of 2, natural log of 10, base-2 log of E, and base-10 log of E.

You may reference these constants from your JavaScript like this:

Math.E

Math.PI

Math.SQRT2

Math.SQRT1_2

Math.LN2

Math.LN10

Math.LOG2E

Math.LOG10E

Send your requisition at

info@biyanicolleges.org