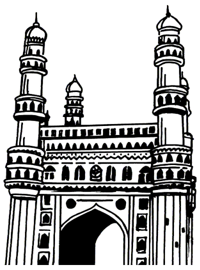
Rahul's ✔
Topper's Voice



M.C.A.

II Year III Sem

(Osmania University)

Latest **2024** Edition

WEB TECHNOLOGIES

- **Study Manual**
- **Important Questions**
- **Solved Model Papers**
- **Solved Previous Question Papers**

Price ` . 199-00

- by -

WELL EXPERIENCED LECTURER

Rahul Publications™
Hyderabad. Cell : 9391018098, 9505799122

M.C.A.

II Year III Sem

(Osmania University)

WEB TECHNOLOGIES

Price `. 199 -00

Sole Distributors : **Cell : 9391018098, 9505799122**

VASU BOOK CENTRE

Shop No. 2, Beside Gokul Chat, Koti, Hyderabad.

Maternity Hospital Opp. Lane, Narayan Naik Complex, Koti, Hyderabad.
Near Andhra Bank, Subway, Sultan Bazar, Koti, Hyderabad -195.

CONTENTS

WEB TECHNOLOGIES

Contents

Important Questions

UNIT - I

1. How can you insert an image in to your web document ?

Ans :

Refer Unit-I, Q.No.14

---

2. What is mean by Nested tables? How can you create ?

Ans :

Refer Unit-I, Q.No. 17

---

3. Define Listing. Explain types of list with sample html code.

Ans :

Refer Unit-I, Q.No. 20

---

4. What is DHTML ? What are the features of DHTML ?

Ans :

Refer Unit-I, Q.No. 25

---

5. Explain briefly the differences between HTML and DHTML.

Ans :

Refer Unit-I, Q.No. 28

---

6. Explain the different types of CSS with an examples.

Ans :

Refer Unit-I, Q.No. 32

---

7. Explain in detail the CSS box model.

Ans :

Refer Unit-I, Q.No. 33

UNIT - II

1. What is a object model ? Explain its Properties and functions ?

Ans :

Refer Unit-II, Q.No. 1

2. Describe about Java collections framework?

Ans :

Refer Unit-II, Q.No. 2

3. Discuss in brief about Navigator.

Ans :

Refer Unit-II, Q.No. 3

4. Define Event. Explain the various attributes supported by the JavaScript.

Ans :

Refer Unit-II, Q.No. 4

5. Explain briefly about Event handling mechanism?

Ans :

Refer Unit-II, Q.No. 5

6. What is Exception Handling? Explain how to handle the exceptions in Java Script?

Ans :

Refer Unit-II, Q.No. 15

7. Explain filters and transactions for multimedia effects.

Ans :

Refer Unit-II, Q.No. 16

UNIT - III

1. What are the different data types in Java Script?

Ans :

Refer Unit-III, Q.No.5

2. Explain different operators in JavaScript.

Ans :

Refer Unit-III, Q.No.6

3. What are the control statements in Java Script?

Ans :

Refer Unit-III, Q.No. 7

4. **What are arrays in JavaScript? Explain its Methods.**

Ans :

Refer Unit-III, Q.No. 12

5. **What are JavaScript Functions? Explain its types.**

Ans :

Refer Unit-III, Q.No.15

6. **What are objects? Explian how to create an object.**

Ans :

Refer Unit-III, Q.No. 17

UNIT - IV

1. **Explain the concept of VB Script ?**

Ans :

Refer Unit-IV, Q.No.1

2. **Discuss briefly about VB Script Loop Control Statements ?**

Ans :

Refer Unit-IV, Q.No. 5

3. **Write about VB Scripts Strings and Related Functions ?**

Ans :

Refer Unit-IV, Q.No. 7

4. **What are functions in VB Script ?**

Ans :

Refer Unit-IV, Q.No. 8

5. **What is an Array ? Explain in detail in VB Script ? Accessing methods?**

Ans :

Refer Unit-IV, Q.No.9

UNIT - V

1. **What is ASP? Explain its directories and working ?**

Ans :

Refer Unit-V, Q.No.1

2. Explain various variable and its constructs in ASP.

Ans :

Refer Unit-V, Q.No. 4

3. Discuss about different data types in CGI PERL.

Ans :

Refer Unit-V, Q.No.10

4. Explain the properties and methods of server objects ?

Ans :

Refer Unit-V, Q.No. 14

5. State the benefits and goals of XML benefits.

Ans :

Refer Unit-V, Q.No. 16

6. Discuss about different types DTD Element.

Ans :

Refer Unit-V, Q.No. 17

7. Explain how to Access XML Data in ASP.

Ans :

Refer Unit-V, Q.No. 19

8. What is the relation ship between XML and HTML?

Ans :

Refer Unit-V, Q.No. 20

UNIT I

HTML : Markup languages, common tags, header, text styling, linking images Formatting text,Unordered lists, nested and ordered list, Tabs-and formatting, Basic forms; Complex forms linking, Meta Tags.

Dynamic HTML: Cascading style sheets in line styles, style element External Style sheet, text flow and Box model, user style sheets.

1.1 Introduction to HTML

1.1.1 Markup languages

Q1. What is HTML ?

Ans : **(Imp.)**

Meaning

HTML is the standard markup language for creating Web pages.

- HTML stands for Hyper Text Markup Language
- HTML describes the structure of Web pages using markup
- HTML elements are the building blocks of HTML pages
- HTML elements are represented by tags
- HTML tags label pieces of content such as "heading", "paragraph", "table", and so on
- Browsers do not display the HTML tags, but use them to render the content of the page

The definition of HTML is HyperText Markup Language.

- HyperText is the method by which you move around on the web - by clicking on special text called **own** which bring you to the next page. The fact that it is hyper just means it is not linear - i.e. you can go to any place on the Internet whenever you want by clicking on links - there is no set order to do things in.
- Markup is what **HTML tags** do to the text inside them. They mark it as a certain type of text.
- HTML is a Language, as it has code-words and syntax like any other language.

Q2. Why HTML is a Markup language ?

(OR)

HTML is a Markup language - Explain.

Ans :

Hypertext means machine readable text and Markup means to structure it in a specific format. So, HTML is called hypertext markup language because it is a language that allows users to organize, improve the appearance of, and link text with data on the internet

HTML (Hypertext Markup Language) is the set of markup symbols or codes inserted in a file intended for display on a World Wide Web browser page. The markup tells the Web browser how to display a Web page's words and images for the user.

Q3. Explain different types of markup languages.

Ans :

There are three types of electronic markup language :

(i) Presentational Markup: Used by traditional word processing systems with WYSIWYG; it is hidden from human users.

(ii) Procedural Markup: Integrated with text to provide text processing instructions to programs. Such text is visibly manipulated by the author. Procedural markup systems include programming constructs, where macros or subroutines are defined and invoked by name.

(iii) Descriptive Markup: Used to label parts of a document as to how they should be treated. For example, the HTML <cite> tag is used to label citations in text.

Gencode was the first public markup language presentation in computer text processing. Some other major markup languages include:

- LaTex
- Extensible Markup Language (XML)
- Generalized Markup Language (GML)
- Standard Generalized Markup Language (SGML)
- HyperText Markup Language (HTML)

Q4. Discuss in brief the history of HTML?

Ans :

In 1986 the International Organization for Standardization released ISO 8879 entitled 'Information Processing - Text and Office Systems - Standard Generalized Markup Language (SGML). Although this defined a standard for the first time nothing much happened after that, mainly because no-one knew what to do with it. Then Tim Berners-Lee invented the World Wide Web and suddenly the demand was there.

Everybody wanted hypertext documents. More than that everybody wanted to produce software that would create hypertext documents, if only so they could control the standard.

In the software industry standards (the standard or industry accepted way of doing something that all other software manufacturers must conform to) is just about the ultimate prize. Should one company create a standard, known as a de facto standard in that every other company must follow it despite it never being ratified by a standards committee, then the financial rewards can be truly spectacular.

Not only does it have the prestige of being the market leader which can be easily translated into sales, but, if the technology can be patented or copyrighted in some way, then every other company in the business will have to buy a license to use it before they can even begin to compete. A lucrative affair in its own right made even more so by the fact that whoever controls the standard sets the standard. Practically speaking that means they can change it whenever they like, forcing the consumer to buy new software and the rest of the industry to buy new licenses.

Standards, and their control, is therefore an important issue which helps to explain why the W3 Consortium was set up in 1994 to oversee the process, at least as far as the Internet was concerned. Theoretically any web page written to conform to the W3 standard can be read by any Browser no matter where it comes from.

If only life was that simple. With such massive profits at stake the companies concerned are all pushing ahead with advancements to their own Browsers which, they hope, will become so popular that the W3 Consortium will have no choice but to include it in their next official standard release. A good business practice it might be, but along the way all it really produces is confusion the outcome of which is that some websites which include these new features are unreadable by other Browsers.

On a practical basis, then, before anyone is given the go-ahead to design a web page they should be made to state explicitly, and in writing, either that it will incorporate only those features officially approved by the W3 Consortium or if not they must provide a list of all Browsers that will support it. In this case they should also give the version number of each Browser and, just as importantly, the date it was released. That way the company who expects the website to work for them, and are paying for it, can make their own decisions about the size of the market being reached - or being excluded.

For example if a feature is supported by the latest version of Microsoft Internet Explorer the question is how many people are likely to be still using an earlier version which can only be decided when the date of its release is known. Not everyone downloads the latest version immediately it becomes available. A frequently recommended practice is to wait until the early adopters find the bugs and the software company fixes them. Then, when the package is known to be stable and safe, the upgrade can be performed without undue hazard.

Also, remember how the software industry works. What is leading edge today is standard technology tomorrow and almost obsolete the day after (and practically on that time scale). For the web design agencies this creates a tremendous pressure to be constantly working at the leading edge so that when it becomes standard technology they already have the experience and expertise in its use which they can then sell to other clients.

Of course for those first few clients using that leading edge technology this has several implications :

- Delivery could be delayed. If a system has to be learnt before it can be implemented no-one can be sure how long it will take.
- The final product could be flawed. If a system is new no-one can be certain it will work flawlessly, not without extensive testing leading to further delays. Alternatively the website could go live with its faults still undiscovered and still in place making the whole exercise useless.
- The site could be invisible to some users. If a Browser cannot support the features of a particular website it will be unable to read some or all of that website.

While a reputable design agency will make their client aware of this not every entire agency is reputable and so the questions should be asked. At the very least the agency should be able to explain the potential shortcomings of each and every feature included in a proposed website.

Better yet should there be even the slightest hint of a problem the onus should be on the design agency to justify that particular feature. Remember, the only people who score points for having fancy web pages are the web design agencies. Everyone else can make do with bog standard technology. And if the agency cannot create a good looking site with the tremendous variety of features which already exists as standard then find an agency that can.

Next, as if the entire business was not already over-complicated, there are yet more ways in which the waters can be muddied even further. In this case the culprit is a particular piece of software generally referred to as a plug-in.

Q5. Explain the Features of HTML

Ans :

1. It is a very easy and simple language. It can be easily understood and modified.
2. It is very easy to make effective presentation with HTML because it has a lot of formatting tags.
3. It is a markup language so it provides a flexible way to design web pages along with the text.
4. It facilitates programmers to add link on the web pages (by html anchor tag) , so it enhances the interest of browsing of the user.
5. It is platform-independent because it can be displayed on any platform like Windows, Linux and Macintosh etc.
6. It facilitates the programmer to add Graphics, Videos, and Sound to the web pages which makes it more attractive and interactive.

1.1.2 Common Tags

Q6. Explain the tags of HTML

Ans :

HTML tags are element names surrounded by angle brackets :

<tagname>content goes here...</tagname>

- HTML tags normally come in pairs like <p> and </p>
- The first tag in a pair is the start tag, the second tag is the end tag
- The end tag is written like the start tag, but with a forward slash inserted before the tag name

The start tag is also called the opening tag, and the end tag the closing tag.

Basic Structure of a HTML Program

```
<html>
    <head>
        <title>Page title</title>
    </head>
    <body>
    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>
    <p>This is another paragraph.</p>
    </body>
</html>
```

Mainly we have 4 basic elements to develop any html program

1. <HTML> – </HTML>

This tag is used to initiate the html program.

It has opening and closing tag. In between opening and closing mainly we should maintain 2 sub tags called <head> and <body > tags.by using <html> we can able to create web documents /pages.

Syntax :

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

2. <head> – </head>

This tag is used to maintain header section for the web documents or webpages. This tag can capable to declare many styles and rules for the document . this tag supports

syntax:

```
<head>
</head>
```

3. <title> – </title>

This tag is used to give a specific name for the web document

And which is displayed top left of the document. We should remember that always title tag should under head section only.

Syntax:

```
<head>
    <title>
    </title>
</head>
```

4. <body>—</body>

This tag is used to write actual body of the program. And which display the elements up on the web page which have been declared in between opening and closing body tags

Always we should remember that body tag has to declare after head tag only

Syntax:

```
<html>
<head>
        <title>
        </title>
  </head>
  <body>

</body>
</html>
```

Q7. Give and explain some basic tags supported by HTML ?

Ans :

Following are some basic tags supported by HTML, which will be helpful to develop the program in different views

1. Heading Tags

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

```
<html>
<head>
<title>Heading Example</title>
</head>
<body>
<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<h3>This is heading 3</h3>
<h4>This is heading 4</h4>
```

```
<h5>This is heading 5</h5>
<h6>This is heading 6</h6>
</body>
</html>
```

2. Paragraph Tag

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

```
<html>
<head>
<title>Paragraph Example</title>
</head>

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

</html>
```

3. Line Break Tag

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

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

```
<html>
<head>
<title>Line Break  Example</title>
</head>
<body>
    <p>Hello<br/>
You delivered your assignment ontime.<br>
Thanks<br/>
Mahnaz</p>
</body>
</html>
```

4. Centering Content

You can use **<center>** tag to put any content in the center of the page or any table cell.

Example

```
<html>
<head>
<title>Centring Content Example</title>
</head>
<body>
<p>This text is not in the center.</p>
<center>
<p>This text is in the center.</p>
</center>
</body>
</html>
```

5. Horizontal Lines

Horizontal lines are used to visually break-up sections of a document. The **<hr>** tag creates a line from the current position in the document to the right margin and breaks the line accordingly.

For example, you may want to give a line between two paragraphs as in the given example below

Example

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

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

1.1.3 Text Styling, Formatting Text

Q8. How can you format the text in HTML ?

Ans :

(i) HTML Formatting Elements

In the previous chapter, you learned about the HTML style attribute.

HTML also defines special elements for defining text with a special meaning.

HTML uses elements like <b> and <i> for formatting output, like bold or italic text.

Formatting elements were designed to display special types of text :

- <b> - Bold text
- <strong> - Important text
- <i> - Italic text
- <em> - Emphasized text
- <mark> - Marked text
- <small> - Small text
- <del> - Deleted text
- <ins> - Inserted text
- <sub> - Subscript text
- <sup> - Superscript text.

1. <b> and <strong>

The HTML <b> element defines **bold** text, without any extra importance.

Example

<b>This text is bold</b>

The HTML <strong> element defines **strong** text, with added semantic "strong" importance.

Example

<strong>This text is strong</strong>

2. <i> and <em>

The HTML <i> element defines italic text, without any extra importance.

Example

<i>This text is italic</i>

The HTML <em> element defines emphasized text, with added semantic importance.

Example

<em>This text is emphasized</em>

3. <small>

The HTML <small> element defines smaller text:

Example

<h2>HTML <small>Small</small> Formatting</h2>

4. <mark>

The HTML <mark> element defines marked or highlighted text:

Example

<h2>HTML <mark>Marked</mark> Formatting</h2>

5. <del>

The HTML <del> element defines ~~deleted~~ (removed) text.

Example

<p>My favorite color is <del>blue</del> red.</p>

6. <ins>

The HTML <ins> element defines inserted (added) text.

Example

<p>My favorite <ins>color</ins> is red.</p>

7. <sub>

The HTML <sub> element defines $_{\text{subscripted}}$ text.

Example

<p>This is <sub>subscripted</sub> text.</p>

8. <sup>

The HTML <sup> element defines $^{\text{superscripted}}$ text.

Example

<p>This is <sup>superscripted</sup> text.</p>

(ii) HTML Text Formatting Elements

Tag	Description
<b>	Defines blod text
<em>	Defines emphasized text
<i>	Defines italic text
<small>	Defines smaller text
<strong>	Defines important text
<sub>	Defines subscripted text
<sup>	Defines superscripted text
<ins>	Defines inserted text
<del>	Defines deleted text
<mark>	Defines marked/highlighted text

Q9. List and explain some Text Formatting Tags.

Ans :

The following HTML tags are used to format the appearance of the text on your web page.

(i) Header - <h?></h?>

There are 6 levels of headings available, from h1 for the largest and most important heading, down to h6 for the smallest heading.

(ii) Bold - <b></b>

The text in between the tags will be bold, and stand out against text around it, the same as in a word processor.

(iii) Italic - <i></i>

Also working the same way as a word processor, italics displays the text at a slight angle.

(iv) Underline - <u></u>

Again, the same as underline in a word processor. Note that html links are already underlined and don't need the extra tag.

(v) Strike-out - <strike></strike>

Puts a line right through the centre of the text, crossing it out.Often used to show that text is old and no longer relevant. Also works by using <s></s> instead.

(vi) Preformatted Text - <pre></pre>

Any text between the pre tags, including spaces, carriage returns and punctuation, will appear in the browser as it would in a text editor (normally browsers ignore multiple spaces)

(vii) Source Code - <code></code>

Text is displayed in a fixed-width font, commonly used when showing source code. I have used it on this site, along with stylesheets, to show all tags.

(viii) Typewriter Text - <tt></tt>

The text appears to have been typed by a typewriter, in a fixed-width font.

(ix) Block Quote - <blockquote></blockquote>

Defines a long quotation, and the quote is displayed with an extra wide margin on the left hand side of the block quote.

(x) Small - <small></small>

Instead of having to set a font size, you can use the small tag to render text slightly smaller than the text around it. Useful for displaying the 'fine-print'.

(xi) Font Colour - <font color="#??????"></font>

Change the colour of a few words or a section of text. The 6 question marks represent the hex color code,

(xii) Font Size - <font size="?"></font>

Replace the ?with a number from 1 to 7 to change the size of the font. One being the smallest and seven the largest.

(xiii) Font Size Change - <font size="+/-?"></font>

For an immediate change of font size with respect to the font size preceding it, this tag increase or decreases the size of the font by the number you specify. Eg: <font size="-1">Some Text</font>

(xiv) Change Font Face - <font face="?"></font>

To show text in a particular font, use the font name such "Helvetica" or "Arial" or "Courier". Be aware that using some fancy font from your computer means that the person viewing that page must also have that font installed on their computer too, otherwise it will look totally different to them.

(xv) Centre - <center></center>

A useful tag, as it says, it makes everything in between the tags centred (in the middle of the page).

(xvi) Emphasis - <em></em>

Used to emphasize text, which usually appears in italics, but can vary according to your browser.

(xvii) Strong Emphasis - <strong></strong>

Used to emphasize text more, which usually appears in bold, but can vary according to your browser.

Text Formatting

<h?> ... <,'h?>	Heading (?=1 for largest to 6 to smallest, e.g., h1)
<b> ... </b>	Bold Text
<i> ... </i>	Italic Text
<u> ... </u>	Underline Text
<strike> ... </strike>	Strikeout
<sup> ... </sup>	Superscript - Smaller text placed below normal text
<sub> ... </sub>	Subscript - Smaller text placed below normal text
<small> ... </small>	Small - Fineprint size text
<tt> ... < tt>	Typewriter Text
<pre> ... </pre>	Pre-formatted Text
<blockquote> ... </blockquote>	Text Block Quote
</strong> ... </strong>	Strong – Shown as Bold in most browsers
<em> ... </em>	Emphasis - Shown as Italics in most browsers
<font> ... </font>	Font tag obsolete

Q10. Explain the elements of forms?

(OR)

Explain form control methods.

Ans :

HTML Form Controls

There are different types of form controls that you can use to collect data using HTML form -

1. Text Input Controls
2. Checkboxes Controls
3. Radio Box Controls
4. Select Box Controls
5. File Upload Control
6. Button Controls
7. Hidden form Controls

1. Text Input Controls

There are three types of text input used on forms -

(i) Single-line text input controls " This control is used for items that require only one line of user input, such as search boxes or names. They are created using HTML **<input>** tag.

This control is used for items that require only one line of user input, such as search boxes or names. They are created using HTML <input> tag.

Example

Here is a basic example of a single-line text input used to take first name and last name -

```
<html>
<head>
<title>Text Input Control</title>
</head>
<body>
<form>
    First name: <inputtype="text" name="first_name"/>
<br>
    Last name: <inputtype="text" name="last_name"/>
</form>
</body>
</html>
```

This will produce the following result -

Attributes

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

Attribute	Description
type	Indicates the types of input control and for text output control it will be se to text.
name	Used to give a name to the control which is sent to the server to be recognized and get the value.
value	This can be used to provide an initial value inside the control.
size	Allows to specify the width of the text-input control in terms of characters.
maxlength	Allows to specify the maximum number of characters a user can enter into the text box.

(ii) Password input controls: This is also a single-line text input but it masks the character as soon as a user enters it. They are also created using HTMl<input> tag.

(iii) Multi-line text input controls: This is used when the user is required to give details that may be longer than a single sentence. Multi-line input controls are created using HTML **<textarea>** tag.

This is used when the user is required to give details that may be longer than a single sentence. Multi-line input controls are created using HTML <textarea> tag.

Example

Here is a basic example of a multi-line text input used to take item description:

```
<html>

<head>
<title>Multiple-Line Input Control</title>
</head>

<body>
<form>
          Description : <br/>
<textarearows= "5"   cols = "50"   name = "description">
          Enter description here...
</textarea>
</form>
</body>

</html>
```

Attributes

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

Attribute	Description
name	Used to give a name to the control which is sent to the server to be recognised and get the value.
rows	Indicates the number of rows of text area box.
cols	Indicates the number of columns of text area box.

2. Checkbox Control

Checkboxes are used when more than one option is required to be selected. They are also created using HTML <input> tag but type attribute is set to checkbox..

Example

Here is an example HTML code for a form with two checkboxes

```
<html>
<head>
<title>Checkbox Control</title>
</head>
<body>
<form>
<inputtype=”checkbox”  name=”maths”  value=”on”>Maths
<inputtype=”checkbox”  name=”physics”  value=”on”> Physics
</form>
</body>
</html>
```

This will produce the following result -

Attributes

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

Attribute	Description
type	Indicates the types of input control and for checkbox input control it will be set to checkbox.
name	Used to give a name to the control which is sent to the server to be recognized and get the value.
value	The value that will be used if the checkbox is selected.
checked	Set to checked if you want to select it by default.

3. Radio Box Control

Radio buttons are used when out of many options, just one option is required to be selected. They are also created using HTML <input> tag but type attribute is set to **radio**.

Example

```
<html>
<head>
<title>Radio Box Control</title>
</head>
<body>
<form>
<inputtype="radio" name="subject" value="maths">Maths
<inputtype="radio" name="subject" value="physics"> Physics
</form>
</body>

</html>
```

This will produce the following result -

Attributes

Following is the list of attributes for radio button.

Attribute	Description
type	Indicates the types of input control and for checkbox input control it will be set to radio.
name	Used to give a name to the control which is sent to the server to be recognized and get the value.
value	The value that will be used if the radio box is selected.
checked	Set to checked if you want to select it by default.

4. Select Box Control

A select box, also called drop down box which provides option to list down various options in the form of drop down list, from where a user can select one or more options.

Example

```
<html>
<head>
<title>Select Box Control</title>
</head>
      <body>
```

```
<form>
<selectname=”dropdown”>
<optionvalue=”Maths”selected>Maths</option>
<optionvalue=”Physics”>Physics</option>
</select>
</form>
</body>
</html>
```

Attributes

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

Attribute	Description
name	Used to give a name to the control which is sent to the server to be recognized and get the value.
size	This can be used to present a scrolling list box.
multiple	If set to “multiple” then allows a user to select multiple items from the menu.

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

Attribute	Description
value	The value that will be used if an option in the select box, box is selected.
selected	Specifies that this option should be the initially selected value when the page loads.
label	An alternative way of labeling options.

5. File Upload Box

If you want to allow a user to upload a file to your web site, you will need to use a file upload box, also known as a file select box. This is also created using the <input> element but type attribute is set to **file**.

```
<html>

<head>
<title>File Upload Box</title>
</head>
<body>
<form>
```

```
<inputtype="file" name="fileupload" accept="image/*"/>
</form>
</body>

</html>
```

Attributes

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

Attribute	Description
name	Used to give a name to the control which is sent to the server to be recognised and get the value.
accept	Specifies the types of files that the server accepts.

6. Button Controls

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

Type	Description
submit	This creates a button that automatically submits a form.
reset	This creates a button that automatically resets form controls to their initial values.
button	This creates a button that is used to trigger a client-side script when the user clicks that button.
image	This creates a clickable button but we can use an image as background of the button.

Example

```
<html>
<head>
<title>File Upload Box</title>
</head>
<body>
<form>
<inputtype="submit" name="submit" value="Submit"/>
<inputtype="reset" name="reset" value="Reset"/>
<inputtype="button" name="ok" value="OK"/>
<inputtype="image" name="imagebutton" src="/html/images/logo.png"/>
</form>
</body>
</html>
```

7. Hidden Form Controls

Hidden form controls are used to hide data inside the page which later on can be pushed to the server. This control hides inside the code and does not appear on the actual page. For example, following hidden form is being used to keep current page number. When a user will click next page then the value of hidden control will be sent to the web server and there it will decide which page will be displayed next based on the passed current page.

Example

```
<html>
<head>
<title>File Upload Box</title>
</head>
<body>
<form>
<p>This is page 10</p>
<inputtype=”hidden”   name=”pagename”   value=”10"/>
<inputtype=”submit”   name=”submit”   value=”Submit”/>
<inputtype=”reset”   name=”reset”   value=”Reset”/>
</form>
</body>
</html>
```

Q11. What is the use of anchor tag in html ?

(OR)

What is hyperlink? How can you link a page with another page.

Ans :

An anchor tag is an HTML tag. It is used to define the beginning and end of a hypertext link. It contains visible words within a text that can be clicked and the URL of the link’s target.

The HTML code for creating a link to another page or to a particular section within a page. It is also commonly called an “h-ref.” See HREF.

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

The <a> can be used in 2 ways

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

Syntax:

```
<a href=”url”>link text</a>
```

The href attribute specifies the destination of a link.

Example

```
<a href=”http//:www.w3cschools.com”>visit w3c schools</a>
```

By click on above link current page is directed to w3cschools page. This kind of lining between pages done by using <a>only. This mechanism is also called as “hyperlink”

Sample program

```
<html>
<head>
<title>creating hyperlink</title>
</head>
<body>
<center>
<h1>creating hyperlink</h1>
Here is a website to checkout
<a href=http://w3c.org>click here for w3cschools</a>
</center>
</body>
</html>
```

Q12. What is backgrouds? How can you set background for a html document ?

Ans :

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

- HTML Background with Colors
- HTML Background with Images

Html Background with Colors

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

Note : The *bgcolor* attribute deprecated in HTML5. Do not use this attribute.

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

```
<tagnamebgcolor = “color_value”...>
```

This color_value can be given in any of the following formats -

```
<!— Format 1 - Use color name —>
<table bgcolor = “lime” >
<!— Format 2 - Use hex value —>
<table bgcolor = “#f1f1f1” >
<!— Format 3 - Use color value in RGB terms —>
<table bgcolor = “rgb(0,0,120)” >
```

Example

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

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

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

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

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

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

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

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

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

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

Example

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

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

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

Q13. Explain the Patterned & Transparent Backgrounds.

Ans :

You might have seen many pattern or transparent backgrounds on various websites. This simply can be achieved by using patterned image or transparent image in the background.

It is suggested that while creating patterns or transparent GIF or PNG images, use the smallest dimensions possible even as small as 1x1 to avoid slow loading.

Example

Here are the examples to set background pattern of a table -

```
<html>
<head>
<title>HTML Background Images</title>
</head>
<body>
<!— Set a table background using pattern —>
<tablebackground=”/images/pattern1.gif”   width=”100%”   height=”100">
<tr>
<td>
This background is filled up with a pattern image.
</td>
</tr>
```

```
</table>
<!— Another example on table background using pattern —>
<tablebackground=”/images/pattern2.gif” width=”100%” height=”100">
<tr>
<td>
This background is filled up with a pattern image.
</td>
</tr>
</table>
</body>
</html>
```

1.1.4 linking images

Q14. How can you insert an image in to your web document ?

Ans : **(Imp.)**

Images are very important to beautify as well as to depict many complex concepts in simple way on your web page. This tutorial will take you through simple steps to use images in your web pages.

Insert Image

You can insert any image in your web page by using **<img>** tag. Following is the simple syntax to use this tag.

<imgsrc = “Image URL” ... attributes-list/>

The <img> tag is an empty tag, which means that, it can contain only list of attributes and it has no closing tag.

Example

To try following example, let’s keep our HTML file test.htm and image file test.png in the same directory -

```
<html>
<head>
<title>Using Image in Webpage</title>
</head>

<body>
<p>Simple Image Insert</p>
<imgsrc=”/html/images/test.png” alt=”Test Image”/>
</body>

</html>
```

You can use PNG, JPEG or GIF image file based on your comfort but make sure you specify correct image file name in **src** attribute. Image name is always case sensitive.

The **alt** attribute is a mandatory attribute which specifies an alternate text for an image, if the image cannot be displayed.

Q15. Explain the properties (or) attributes of an image.

Ans :

Following are the basic properties of images.

1. Set Image Location

Usually we keep all the images in a separate directory. So let's keep HTML file test.htm in our home directory and create a subdirectory **images** inside the home directory where we will keep our image test.png.

Example

Assuming our image location is "image/test.png", try the following example -

```
<html>
<head>
<title>Using Image in Webpage</title>
</head>
<body>
<p>Simple Image Insert</p>
<imgsrc="/html/images/test.png" alt="Test Image"/>
</body>
</html>
```

2. Set Image Width/Height

You can set image width and height based on your requirement using **width** and **height** attributes. You can specify width and height of the image in terms of either pixels or percentage of its actual size.

```
<html>
<head>
<title>Set Image Width and Height</title>
</head>
<body>
<p>Setting image width and height</p>
<imgsrc="/html/images/test.png" alt="Test Image" width="150" height="100"/>
</body>
</html>
```

3. Set Image Border

By default, image will have a border around it, you can specify border thickness in terms of pixels using border attribute. A thickness of 0 means, no border around the picture.

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

```
<title>Set Image Border</title>
</head>
<body>
<p>Setting image Border</p>
<imgsrc=”/html/images/test.png”  alt=”Test Image”  border=”3"/>
</body>
</html>
```

4. Set Image Alignment

By default, image will align at the left side of the page, but you can use **align** attribute to set it in the center or right.

```
<html>

<head>
<title>Set Image Alignment</title>
</head>

<body>
<p>Setting image Alignment</p>
<imgsrc=”/html/images/test.png”  alt=”Test Image”  border=”3"  align=”right”/>
</body></html>
```

1.2 Tables

Q16. Explain the properties (or) attributes of <table> in html ?

Ans :

1. Table Heading

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

Example

```
<html>
<head>
<title>HTML Table Header</title>
</head>
<body>
<tableborder=”1">
<tr>
```

```
<th>Name</th>
<th>Salary</th>
</tr>
<tr>
<td>Ramesh Raman</td>
<td>5000</td>
</tr>
<tr>
<td>Shabbir Hussein</td>
<td>7000</td>
</tr>
</table>
</body>
</html>
```

2. 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.

Example

```
<html>
<head>
<title>HTML Table Background</title>
</head>
<body>
<tableborder="1" bordercolor="green"
bgcolor="yellow">
<tr>
<th>Column 1</th>
<th>Column 2</th>
<th>Column 3</th>
</tr>
<tr>
<tdrowspan="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>
<tdcolspan="3">Row 3 Cell 1</td>
</tr>
</table>
</body>

</html>
```

3. Table Caption

The **caption** tag will serve as a title or explanation for the table and it shows up at the top of the table. This tag is deprecated in newer version of HTML/XHTML.

Example

```
<html>
<head>
<title>HTML Table Caption</title>
</head>
<body>
<tableborder="1" width="100%">
<caption>This is the caption</caption>
<tr>
<td>row 1, column 1</td><td>row 1,
columnn 2</td>
</tr>
<tr>
<td>row 2, column 1</td><td>row 2,
columnn 2</td>
</tr>
</table>
</body>
</html>
```

This is the caption	
row 1, column 1	row 1, column 2
row 2, column 1	row 2, column 2

4. Table Header, Body, and Footer

Tables can be divided into three portions “a header, a body, and a foot. The head and foot are rather similar to headers and footers in a word-processed document that remain the same for every page, while the body is the main content holder of the table.

The three elements for separating the head, body, and foot of a table are -

- **<thead>** - to create a separate table header.
- **<tbody>** - to indicate the main body of the table.
- **<tfoot>** - to create a separate table footer.

A table may contain several <tbody> elements to indicate *different pages* or groups of data. But it is notable that <thead> and <tfoot> tags should appear before <tbody>

Example

```
<html>

<head>
<title>HTML Table</title>
</head>
<body>
<tableborder=”1" width=”100%”>
<thead>
<tr>
<tdcolspan=”4">This is the head of the table</
td>
</tr>
</thead>
<tfoot>
<tr>
<tdcolspan=”4">This is the foot of the table</
td>
</tr>
</tfoot>
<tbody>
<tr>
<td>Cell 1</td>
<td>Cell 2</td>
<td>Cell 3</td>
<td>Cell 4</td>
</tr>
</tbody>

</table>
</body>

</html>
```

This will produce the following result -

This is the head of the table
This is the foot of the table

Q17. What is mean by Nested tables? How can you create ?

Ans : **(Imp.)**

You can use one table inside another table. Not only tables you can use almost all the tags inside table data tag <td>.

Example

Following is the example of using another table and other tags inside a table cell.

```
<html>

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

<body>
<tableborder=”1"width=”100%”>
<tr>
<td>
<tableborder=”1"width=”100%”>
<tr>
<th>Name</th>
<th>Salary</th>
</tr>
<tr>
<td>Ramesh Raman</td>
<td>5000</td>
</tr>
<tr>
```

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

This will produce the following result -

Name	Salary
Ramesh Raman	5000
Shabbir Hussein	7000

Q18. Explain the following terms related to table.

(a) Cell padding and cell spacing attributes

(b) Colspan and rowspan attributes.

Ans :

(a) 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.

Example

```
<html>
<head>
<title>HTML Table Cellpadding</title>
</head>
<body>
<tableborder="1" cellpadding="5" cellspacing="5">
<tr>
<th>Name</th>
<th>Salary</th>
</tr>
<tr>
<td>Ramesh Raman</td>
<td>5000</td>
</tr>
<tr>
<td>Shabbir Hussein</td>
<td>7000</td>
</tr>
</table>
</body>
</html>
```

(b) 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.

Example

```
<html>
<head>
<title>HTML Table Colspan/Rowspan</title>
</head>
<body>
<tableborder="1">
<tr>
<th>Column 1</th>
<th>Column 2</th>
<th>Column 3</th>
</tr>
<tr>
<tdrowspan="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>
<tdcolspan="3">Row 3 Cell 1</td>
</tr>
</table>
</body>
</html>
```

Q19. Write a program to demonstrate rowspan and colspan attributes in table shown below.

Name	Marks		
	FM	FIT	BOM
Rahul	68	68	75
Rakesh	60	69	60
Ravi	68	66	60

Ans :

Program

```
<html>
<head>
<title> Tables </title>
<body bgcolour = "pink" align = "centre"?
<table border = "1" width = "50%">
<col width = "75">
<col width = "25">
<tr>
<th rowspan = 2> Name </th>
<th colspan = 3> Marks </th>
</tr>
<tr>
<td> FM </td>
<td> FIT </td>
<td> BOM </td>
</tr>
<tr>
<align = "left">
<td> Rahul </td>
<td> 68 </td>
<td> 68 </td>
<td> 75 </td>
</tr>
<tr>
<td> Rakesh </td>
<td> 60 </td>
<td> 69 </td>
<td> 60 </td>
</tr>
<tr>
<td> Ravi </td>
<td> 68 </td>
<td> 66 </td>
<td> 60 </td>
</tr>
</table>
</body>
</html>
```

Output

Name	Marks		
	FM	FIT	BOM
Rahul	68	68	75
Rakesh	60	69	60
Ravi	68	66	60

1.3 Lists

1.3.1 Unordered lists, nested and Ordered list

Q20. What is meant by List ? How we can create the list in html ?

(OR)

Define Listing. Explain types of list with sample html code.

Ans : **(Imp.)**

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

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

(i) Unordered Lists

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

```
<html>

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

<body>
<ul>
<li>Beetroot</li>
<li>Ginger</li>
<li>Potato</li>
<li>Radish</li>
</ul>
</body>
</html>
```

Output

- ➢ Beetroot
- ➢ Ginger
- ➢ Potato
- ➢ Radish

The type Attribute

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

```
<ul type = "square">
<ul type = "disc">
<ul type = "circle">
```

(ii) Ordered Lists

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

```
<html>

<head>
<title>HTML Ordered List</title>
</head>
<body>
<ol>
<li>Beetroot</li>
<li>Ginger</li>
<li>Potato</li>
<li>Radish</li>
</ol>
</body>
</html>
```

Output

1. Beetroot
2. Ginger
3. Potato
4. Radish

The type Attribute

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

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

(iii) Definition Lists

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

Definition

List makes use of following three tags.

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

Example

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Definition List</title>
</head>

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

</html>
```

Output

HTML : This stands for Hyper Text Markup Language

HTTP : This stands for Hyper Text Transfer Protocol.

1.4 FRAMES

Q21. What is a frame? Explain different types of frames.

Ans :

Meaning

HTML frames are used to divide your browser window into multiple sections where each section can load a separate HTML document. A collection of frames in the browser window is known as a frameset. The window is divided into frames in a similar way the tables are organized: into rows and columns.

Disadvantages

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

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

Creating Frames

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

Note: The <frame> tag deprecated in HTML5. Do not use this element.

Example

Following is the example to create three horizontal frames -

```
<!DOCTYPE html>
    <html>
        <head>
        <title>HTML Frames</title>
        </head>
        <framesetrows="10%,80%,10%">
    <framename="top"src="/html/top_frame.htm"/>
    <framename="main"src="/html/main_frame.htm"/>
    <framename="bottom"src="/html/bottom_frame.htm"/>
    <noframes>
    <body>Your browser does not support frames.</body>
    </noframes>
    </frameset>
    </html>
```

Example

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

```
<!DOCTYPE html>
    <html>
    <head>
    <title>HTML Frames</title>
</head>
<framesetcols="25%,50%,25%">
<framename="left"src="/html/top_frame.htm"/>
<framename="center"src="/html/main_frame.htm"/>
<framename="right"src="/html/bottom_frame.htm"/>
<noframes>
<body>Your browser does not support frames.</body>
</noframes>
</frameset>
</html>
```

The <frameset> Tag Attributes

Following are important attributes of the <frameset> tag:

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

The <frame> Tag Attributes

Following are the important attributes of <frame> tag –

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

5.	**Marginheight:** This attribute allows you to specify the height of the space between the top and bottom of the frame's borders and its content. The value is given in pixels. For example margin height = "10".
6.	**Nore size:** By default, you can resize any frame by clicking and dragging on the borders of a frame. The nore size attribute prevents a user from being able to resize the frame. For example noresize = "noresize".
7.	**Scrolling:** This attribute controls the appearance of the scrollbars that appear on the frame. This takes values either "yes", "no" or "auto". For example scrolling = "no" means it should not have scroll bars.
8.	**Longdesc:** This attribute allows you to provide a link to another page containing a long description of the contents of the frame. For example long desc = "framedescription.htm".

Browser Support for Frames

If a user is using any old browser or any browser, which does not support frames then <noframes> element should be displayed to the user.

So you must place a <body> element inside the <noframes> element because the <frameset> element is supposed to replace the <body> element, but if a browser does not understand <frameset> element then it should understand what is inside the <body> element which is contained in a <noframes> element.

You can put some nice message for your user having old browsers. For example, Sorry!! your browser does not support frames as shown in the above example.

Frame's Name and Target Attributes

One of the most popular uses of frames is to place navigation bars in one frame and then load main pages into a separate frame.

Let's see following example where a test.htm file has following code -

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Target Frames</title>
</head>
<framesetcols="200, *">
<framesrc="/html/menu.htm"name="menu_page"/>
<framesrc="/html/main.htm"name="main_page"/>
<noframes>
<body>Your browser does not support frames.</body>
</noframes>
</frameset>
</html>
```

Here, we have created two columns to fill with two frames. The first frame is 200 pixels wide and will contain the navigation menu bar implemented by **menu.htm** file. The second column fills in remaining space and will contain the main part of the page and it is implemented by **main.htm** file. For all the three links available in menu bar, we have mentioned target frame as **main_page**, so whenever you click any of the links in menu bar, available link will open in main page. Following is the content of menu.htm file

```
<!DOCTYPE html>
<html>
<bodybgcolor="#4a7d49">
<ahref="http://www.google.com"target="main_page">Google</a>
<br/>
<br/>
<ahref="http://www.microsoft.com"target="main_page">Microsoft</a>
<br/>
<br/>
<ahref="http://news.bbc.co.uk"target="main_page">BBC News</a>
</body>
</html>
```

Following is the content of main.htm file -

```
<!DOCTYPE html>
<html>
<bodybgcolor="#b5dcb3">
<h3>This is main page and content from any link will be displayed here.</h3>
<p>So now click any link and see the result.</p>
</body>
</html>
```

Q22. Write a short notes on Iframes ?

Ans :

You can define an inline frame with HTML tag **<iframe>**. The <iframe> tag is not somehow related to <frameset> tag, instead, it can appear anywhere in your document. The <iframe> tag defines a rectangular region within the document in which the browser can display a separate document, including scrollbars and borders. An inline frame is used to embed another document within the current HTML document.

The **src** attribute is used to specify the URL of the document that occupies the inline frame.

Example

Following is the example to show how to use the <iframe> -

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Iframes</title>
</head>
<body>
<p>Document content goes here...</p>
<iframesrc="/html/menu.htm"width="555"height="200">
    Sorry your browser does not support inline frames.
</iframe>
<p>Document content also go here...</p>
```

</body>
</html>

The <Iframe> Tag Attributes

Most of the attributes of the <iframe> tag, including name, class, frameborder, id, longdesc, marginheight, marginwidth, name, scrolling, style,and title behave exactly like the corresponding attributes for the <frame> tag.

Note - The frameborder, marginwidth, longdesc, scrolling, marginheight attributes deprecated in HTML5. Do not use these attributes.

Sl. No.	Attribute and Description
1.	**Src:** This attribute is used to give the file name that should be loaded in the frame. Its value can be any URL. For example, src = "/html/top_frame.htm" will load an HTML file available in html directory.
2.	**Name:** This attribute allows you to give a name to a frame. It is used to indicate which frame a document should be loaded into. This is especially important when you want to create links in one frame that load pages into an another frame, in which case the second frame needs a name to identify itself as the target of the link.
3.	**Frameborder:** This attribute specifies whether or not the borders of that frame are shown in to verrides the value given int he frameborder attribute on the <frameset> tag if one is given, and this can take values either 1(yes) or 0(no).
4.	**Marginwidth:** This attribute allows you to specify the width of the space between the left and right of the frame's borders and the frame's content. The value is given in pixels. For example margin width = "10".
5.	**Marginheight:** This attribute allows you to specify the height of the space between the top and bottom of the frame's borders and its contents. The value is given in pixels. For example margin height = "10".
6.	**Height:** This attribute specifies the height of <iframe>.
7.	**Scrolling:** This attribute controls the appearance of the scrollbars that appear on the frame. This takes value either "yes", "no" or "auto". For example scrolling = "no" means it should not have scroll bars.
8.	**Longdesc:** This attribute allows you to provide a link to another page containing a long description of the contents of the frame. For example long desc = "framedescription.htm".
9.	**Width:** This attribute specifies the width of <iframe>.

1.5 Forms

1.5.1 Basic forms; Complex forms, linking

Q23. What are forms ? Explain different form elements applicable in HTML ?

Ans :

An HTML form is a section of a web document into which the user can enter information. This information is passed back to a web server where it might be recorded in a database for future use or perhaps used to control what information is returned to the user.

HTML forms can prompt the user to type in some text or choose from a number of options, collect several different items of information at once and can restrict user responses to a se of known values. Including forms in web pages is the most challenging feature for web designers. Forms are used for two-way communication between web pages and web sites.

Common uses of Forms include

- **Comment response:** It is generally used as a way to collect comments from web site viewers and have people suggest improvements.
- **On Line Order Forms:** These are now common on the web, provide a way for viewers to order goods from online stores. Order entry forms typically require the user to provide an address, credit card number and other information necessary to facilitate online commerce.
- **Subscription forms:** A subscribe Model is adopted by many sites, particularly those that attempt to generate revenue through direct subscriptions or by selling advertising space.
- **Registration forms:** These are used to collect information about a user and often are tied to an authentication system, which limits access to the site.
- **Customization forms:** Some sites allow the user to select the look and feel for the site itself, literally creating a custom site for each visitor. This form might allow users to specify what topics they are interested in within an online magazine. An HTML form consists of two parts. The first part, the user can see and which he fills out and next one is the part, the users cannot see.

The second part specifies how the server should process the user's information you should not that HTML forms cannot provide a fully interactive user interface, they can only construct a query or submission to be fetched like any other web page. There is no way of controlling what is typed into forms fields. Forms only prompt the user for information.

To handle the information entered by the user through the form usually requires the provider to write a CGI-based program designed specifically to process submissions from that form.

Most frequently used form attributes are:

Attribute	Description
name	The name of the form.
action	URL of the program that processes the information submitted via form.
method	The HTTP method that the browser uses to submit the form. Possible values are get and post.
target	A name or keyword indicating the target page where the result of the script will be displayed. The reserved keywords are_blank,_self,_parent and_top.

Form Controls or Widgets

A form is made up of fields or controls as well as the markup necessary to structure the form and control it presentation. The controls are the items filled in or manipulated by the user to indicate the state of the form. Form controls include:

- **Text Fields:** These are areas for brief text input.
- **Check boxes:** That allow visitors to select none, one or several items from a list use them to elicit multiple answers.
- **Submit and Reset buttons:** These send the form information to the server for processing and return the form to its original settings.

- **Radio buttons:** They give visitors an opportunity to choose only one item:
- **Text Areas:** These are areas for lengthy text input, as in open-ended comments or free form responses.
- **Select lists:** These are lists from which visitors can choose one or more items, use them to present long but finite list of choices.

The form tag: <form>....... </form>:

In the HTML source code, a form must start with a <FORM> tag and end with a </FORM>tag. It has mainly two attributes "action" and "method".

Syntax <Formaction="URL" method= get | post> form elements </Form>

When you have written a form handler (mainly a CGI program) to which the data in the form will be sent, you will be able to add on action= "URL" attributes to specify the location of the handler.

The "method" attribute tells that browser how to send the user's data to the server. It takes either "get" or "post" as its value. Note that the attributes can appear in any order. So you can put method first and then "action" or vice versa.

The Input Tag

This tag defines an input area within a form. It asks the user to input information in one of several ways. The different ways of input are specified by "type attribute. This attribute can accept text, radio, checkbox, passwords submit, reset, image, hidden, etc. as its value. Each of them will be discussed separately in the coming sections.

Text Fields

The most common "type" of form <INPUT> is text. It presents the user with a prompt for a single line of text. These fields are commonly used for a name address, email address, country, postal code etc. It is a blank area within a form and is the place for visitor-supplied information.

Syntax <INPUTtype="text" name="text-id" value="default-text" size="n" maxlength= "m"read only>

- **Name :** To label the content. This name is not visible to the user and must be unique within the form.
- **Value :** To supply initial text to the text field and is optional. The content of this attribute will be shown on the text field, unless the user changes it.
- **Size :** To specify the physical size of the text input box. The default value is usually a length of 20 characters.
- **Max length :** To set the number of characters a user can input. This value should not be less than the value of the size attribute otherwise user won't be able to type to the end of the box and might get confused. The default value of max length is unlimited.
- **Readonly** : To lock the predefined content within the text field and the content won't be editable further.

Passwords

Password fields are similar to text fields, except the contents of the field are not visible on the screen. It displays * * * * (asterisks) instead or the actual input.

These fields are mainly used for confidential contents, To establish a password field just set

type = "password" in the <INPUT> tag

Syntax: <INPUTtype="password" name="password-id" Value="default text" size="n"

Maxlength="m"readonly>

Radio Buttons

Radio buttons are mutually exclusive i.e. they allow the user to choose one of the several options. Selecting one turns the other options off. These buttons are viewed in the browser as small circles with a prompt and the selected one appears with a solid dot in it.

Syntax <INPUT type="radio" name="radio-id" Value="choice-id"checked>

- **Name:** Applies to the collection of buttons, not just to a particular item. Browsers use the name attribute to decide which buttons belong to the same group and therefore which ones are to set and unset as a group.
- **Value:** It is the value assigned by the user. These are predefined and they do not accept other input. Each radio button must be assigned a value.
- **Checked:** It causes the radio button to be on when the form is first created. It is the default selection.

Checkboxes

An INPUT tag with attribute type ="checkbox" offers the user an "on" or "off" switch. Each check box works independently from the others, visitors can select or deselect any combination of checkboxes use of these boxes is appropriate for open questions or questions that have more than one "right" answer. Checkboxes appear as square boxes and contain a checkmark when act as a checkbox is switch on when the form is submitted its value attribute is submitted as the form data for the named form component.

Syntax <INPUT type="Checkbox" name="box -id" Value="choice-id" checked>

- **Name:** To specify separate name for each checkbox. Unlike radio buttons, each item has a separate label. Although the checkboxes visually appear as a set, logically the items are completely separate.
- **Value:** To provide a descriptive value for each checkbox.
- **Checked:** To specify default selection.

Pull Down Lists(Select Option)

A pull down menu lets the user select one choice out of many possible choices. One nice aspect of pull-down menus is that all choices do not have to be seen on the screen and are normally hidden. They occupy minimal amount of space as it displays only on item of the list. In this kind of input field you use <SELECT>instead of <INPUT> and it has a closing tag.

Syntax

```
<SELECT name="text-id" size="n"multiple>
<OPTION  Value="Choice-id" selected>  Text Label -1  </OPTION>
   .............
   .............
<OPTIONValue="Choice id m"selected>  TEXT LABEL.  - m  </OPTION>
</SELECT>
```

- **Name:** To establish a name for the select field and is used for form processing.
- **Size:** The number of options you want to show in the window. By default its value is one, specifying size value greater than one turns the pull down list into a scrolling list. In the SELECT field, by default, the user can select only one field.

- **Multiple:** To set the SELECT field to accept any number of options. To select multiple options press ctrl key and click on the options.
- **<OPTION> Tag:** To include the list items. For each list item there must be an <option> tag "The closing" tag is optional.
- **Value:** Whenever a form is submitted, the options in the pull down list are passed to the form handles using the "Value" attribute. If the "Value" attribute is omitted the contents of the option are used instead.
- **Selected:** This attribute is used to specify the default other than the first option. More than the first option More than one option can be made default by using this attribute but in that case you just include the multiple attribute, otherwise the last option having selected attribute will be considered as default. The defaults will appear when the form is loaded or reset.

The TEXTAREA tag

This tag is used to set an area within a form in which the user can type a large amount of text. The textarea's are basically used for giving comments and free form feedback from visitors.

Syntax <TEXTAREA name = "text-id" rows = "n" Cols = "m" Wrap =" Virtual physically off" read only> Initial content </TEXT AREA>

- **TEXTAREA:** Sets an area in a form for lengthy visitor input. Initial content for the text area goes between the opening & closing tags. Rows: This attribute specifies the height of the text area. Cols: This attribute specifies the width of the text area.
- **Wrap:** If you turn wrap off the text is handled as one long sequence of text without line breaks. If you set it to virtual the text appears on your page as if it recognized line breaks but when the form is submitted the line breaks are turned off. If you set it to physical the text is submitted exactly as it appears on the screen line breaks included.
- **Readonly:** It prevents the user to alter the initial content.

Reset and Submit Buttons

SUBMIT- After filling up the form, the user needs to submit the information. An <INPUT> tag with type = submit provides a button that submits the information in the completed form to the URL, given as the action attribute of the FORM tag.

The information is submitted using the HTTP request type specified by Form's method attribute discussed in details in topic entitled ("Processing Forms")

Syntax <INPUT type="submit" Value="button label text">

- **Value:** This attribute will change the text mentioned on the button just substitute your text in the value attribute. By default "submit" as written over the button. The button size will be controlled directly by the text length.

RESET

The reset value of the type attribute clears all form entries to the default one or leaves them blank if no defaults are specified Not al HTML forms will use this feature, but it can help users to start afresh or if they want to reconsider the default options.

Syntax <INPUT type="reset" Value="button label text">

The "Value" attribute again is used to specify the text on the button. These buttons can be placed anywhere in the form. But the best place for them is below all the other input fields.

Special < input> Types Files

HTML also supports a special input field, a file field, to allow visitors to upload files. If you want visitors to submit information - say a picture, a spreadsheet, a word-processed document or a scanned document, they can use this field to simply upload the file with the hassle of using FTP or e-mail the file.

Syntax <INPUT type="file" name="field-id" size="n" accept="file-type">

- **Name:** To label the field
- **Size:** To specify the physical size of the field's input box.
- **Accept:** To restrict the file types allowed in a file upload. For example add accept = "image/ gif" to occupy only gif files

To include files of some specific types, just separate them by commas. For example to include BOTH .gif and .jpeg files, the following code can be used.

Accept = "image/gif, image/jpeg"

When the form is loaded or reset, a button labeled "Browse" together with a text box for writing the file name appears on the window. As the user clicks on this button a new window "choose file" will open and the user can select the desired file.

Hidden Value

The input "type" = hidden is unusual in the respect that they do not appear in the displayed form. By setting the "type" attribute to hidden, it is possible to transmit default or previously specified text that is hidden from the user to the handling program. It is a convenient tool for the web designer. It can be used to hold contextual information, such as the name of the form (useful when the same form handles is used for several different forms), where to send the passed data etc.

A hidden input does not show up anywhere on the web page. The visitors would not be able to see it unless they see your page source. This input is recognized only by the form handler program mainly CGI (scripts) that receives the user's information from the form.

Syntax <INPUT type="hidden" name="data-id" Value="hidden form data">

- **Name:** To label the hidden data
- **Value:** It specifies the destination where the hidden form data is to be sent. The value attribute is passed as the form data for the name attribute, but the user cannot see any representation of this form component on the screen and is not prompted to change the form component contents.
- One of the best use of the hidden input is to tell the form handler where to send the processed data and where to send your visitor after he fills out your form.

Examples

<INPUT type=hidden name="send processed data to" Value=" google.com">

The FIELDSET and LEGEND Tags

<FIELDSET>tag creates a box around a group of widgets and <LEGEND>tag provides a label to the field set. Both tags require closing tags.

Syntax

<FIELDSET> control 1 control 2.......Control n < /FIELDSET>

<LEGEND> Group label Text</**LEGEND**>

Disabled Attribute

This attribute disables a form element. The element contents are not only unalterable, but also unusable. It can be applied to any INPUT tag, as well as to the SELECT, OPTION, TEXTAREA tags. So to disable the File field.

Navigating Through a Form

Now we will be discussing how to navigate through a particular form. By default, you navigate through the form in the order in which elements appear in the form, using tab index attribute you can change this default order. Just place the tabindex attribute in the input field. Our sample form shows the following fields and they will be accessed in the order they are listed. Using the tabindex attribute the order will change.

If two elements have the same tabindex value, the orders will be same as the default order i.e. in which they appear. You can also remove an input field from the tab navigation order by assigning it a negative tabindex value. This attribute can be used with INPUT, SELECT and TEXTAREA elements.

Access Keys

To access the form elements the user either clicks the mouse or uses the tab key. You can give your visitors a direct keyboard access by using access key attribute. This attribute specifies the character that you assign to the form element as a particular hot key. When the key is pressed together with the “alt” key, the corresponding form element gets activated. When assigning characters for access key, always be careful that they do not override the browser’s own access keys. Also function keys (FI-F12) cannot be made access keys, as they have already been assigned a function. Otherwise the access key defined by you will get preference over the browser’s access key.

1.6 META TAGS

Q24. What is a Meta Tags?

Ans :

- HTML <meta> tag is very useful for Web developers. Meta Tag contain the information about web page keyword, description, focused key word etc. everything meta contain is optimize the web page in Search Engine.
- HTML Metadata is loosely defined as key data about your web page. Though this definition is easy to remember, it is not very precise. The strength of this definition is in recognizing that metadata is data. As such, metadata can be stored and managed in a database, often called a registry or repository. However, it is impossible to identify metadata just by looking at it. We don’t know when data is metadata or just data.
- HTML Metadata is a concept that applies mainly to electronically archived data and is used to describe the: **a)** definition, **b)** structure and **c)** administration of data files with all contents in context to ease the use of the captured and archived data for further use. Web pages often include metadata in the form of meta tags.
- Description and keywords meta tags are commonly used to describe the Web page’s content. Most search engines use this data when adding pages to their search index.

Example

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1"/>
<meta name="description" content="online web development tutorials Includes HTML Tutorial, Example codes." />
<meta name="keywords" content="html, free, tutorials, html5, tutorial in html,dynamic, online, web, design, tricks, development,site builder, for beginners, master, builder, howto, learn, pdf, editor" />
<meta name="robot" content="index,follow" />
<meta name="revisit-after" content="5" />
</head>
<body>
</body>
</html>
```

<meta> Tag Attributes

Attributes	Value	Description
content	"text"	Specifies the content of the web page.
http-equiv	"content-type" "content-style- type" "expires" "refresh" "set-cookie"	Specify HTTP header Information. Specify the time of expires. Specify the time of refresh. Specify the limit of the cookie.
name	author Description keywords	Specify the name for the author information. Specify the website content. Specify the website content keyword.

1.7 Dynamic HTML

Q25. What is DHTML ? What are the features of DHTML ?

(OR)

Give a brief introduction about DHTML and its features.

Ans : **(Imp.)**

Meaning

DHTML stands for Dynamic hypertext mark up language. DHTML is not a language but a term used to describe the way of making dynamic and interactive web pages.

It is a combination of HTML, JavaScript, Cascading Style Sheets (CSS) and Document Object Model (DOM). Dynamic content is added to static HTML pages using scripts and styles. DHTML uses client side scripting languages like JavaScript to change the static attributes of a HTML page to generate a dynamic effect. This means all DHTML effects achieved are after loading of content on a page without interacting with server again.

Features

All features of dynamic HTML can be classified into four categories as below :

- **Dynamic Content:** Content on the page is modified dynamically based on the user input. Below is the example of the content change when hovering the mouse over the text.
- **Dynamic Style :** The appearance of an element on a web page is modified dynamically like color change or font change. Below is the example of dynamic font change on mouse hover.
- **Dynamic Positioning :** The position of an element is dynamically changed relative to other elements on the page.
- **Dynamic Binding :** Linking an object at run time based on the conditions at that moment.

Q26. What are the advantages of DHTML?

Ans :

Advantages

There are many advantages of using DHTML especially when the content is to be modified dynamically.

- DHTML supports adding styles to static content in various manners.
- It is dynamic so it can be changed even during the run time execution.
- Webmasters are often limited to use default fonts such as Arial or Times Roman. DHTML allows downloadable fonts which make the web pages looking more attractive.
- DHTML page is also saved as an .html file.

It is worth to note here that using multiple scripts on a web page will reduce the page loading speed and slower the site. Also dynamic pages may not perform well on search engines compared to static HTML pages.

Some of the general examples where DHTML is used include :

- Generating animated text
- Content slide from one position to another
- Collect user inputs through a form and processing through client site JavaScript
- Dynamic menu and mouse over events.

Q27. Give a basic introduction about DHTML?

Ans :

DHTML is the art of combining HTML, JavaScript, DOM, and CSS.

Should have a basic understanding of the following Languages :

(i) HTML

(ii) JavaScript

(iii) CSS

- DHTML is NOT a Language
- DHTML stands for Dynamic HTML.
- DHTML is NOT a language or a web standard.

To most people DHTML means the combination of HTML, JavaScript, DOM and CSS.

According to the World Wide Web Consortium (W3C)

"Dynamic HTML is a term used by some vendors to describe the combination of HTML, style sheets and scripts that allows documents to be animated."

(i) HTML

The W3C HTML 4 standard has rich support for dynamic content :

- HTML supports JavaScript
- HTML supports the Document Object Model (DOM)
- HTML supports HTML Events
- HTML supports Cascading Style Sheets (CSS)

DHTML is about using these features, to create dynamic and interactive web pages.

(ii) JavaScript

JavaScript is the most popular scripting language on the internet, and it works in all major browsers.

DHTML is about using JavaScript to control, access and manipulate HTML elements.

HTML DOM

The HTML DOM is a W3C standard. It describes the Document Object Model for HTML.

The HTML DOM defines a standard way for accessing and manipulating HTML documents.

DHTML is about using the DOM to access and manipulate HTML elements.

HTML Events

HTML events are a part of the HTML DOM.

DHTML is about creating web pages that reacts to (user)events.

(iii) CSS

CSS defines how to display HTML elements.

DHTML is about using JavaScript and the HTML DOM to change the style and positioning of HTML elements.

Q28. What are the differences between HTML and DHTML?

(OR)

Explain briefly the differences between HTML and DHTML.

Ans : **(Imp.)**

Sl.No.	HTML	Sl. No.	DHTML
1.	HTML is Hypertext Markup Language	1.	DHTML is Dynamic Hypertext Markup Language.
2.	HTML stands for only static pages.	2.	DHTML is Dynamic HTML means HTML + JavaScript.
3.	It is referred as a static HTML and static in nature.	3.	It is referred as a dynamic HTML and dynamic in nature.
4.	A plain page without any styles and scripts called as HTML.	4.	A page with HTML, CSS, DOM and Scripts called as DHTML.
5.	HTML sites will be slow upon client-side technologies.	5.	DHTML sites will be fast enough upon client-side technologies.

1.8 Cascading Style Sheets

Q29. Give a basic syntax and structure of CSS?

Ans :

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

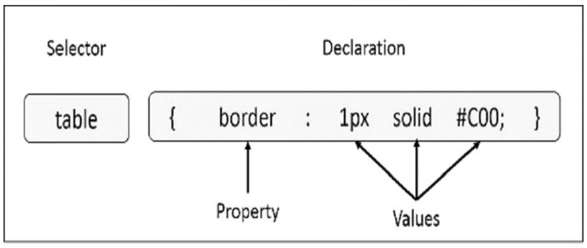
- **Selector** - A selector is an HTML tag at which a style will be applied. This could be any tag like <h1> or <table> etc.

➢ **Property** - A property is a type of attribute of HTML tag. Put simply, all the HTML attributes are converted into CSS properties. They could be color, border etc.

➢ **Value** - Values are assigned to properties. For example, color property can have value either red or #F1F1F1 etc.

Syntax

```
selector
{
property: value
 }
```



Example

You can define a table border as follows-

```
Table
{
border :1px solid #C00;
 }
```

Here table is a selector and border is a property and given value 1px solid #C00 is the value of that property.

You can define selectors in various simple ways based on your comfort. Let me put these selectors one by one.

Q30. Explain briefly about

(i) Class Selectors

(ii) ID Selectors

Ans :

(i) Class Selectors

You can define style rules based on the class attribute of the elements. All the elements having that class will be formatted according to the defined rule.

```
.black
{
   color: #000000;
}
```

This rule renders the content in black for every element with class attribute set to black in our document. You can make it a bit more particular. For example:

```
h1.black
{
   color: #000000;
}
```

This rule renders the content in black for only <h1> elements with class attribute set to black.

You can apply more than one class selectors to given element. Consider the following example:

<p class="center bold">

This para will be styled by the classes center and bold.

</p>

(ii) ID Selectors

You can define style rules based on the id attribute of the elements. All the elements having that id will be formatted according to the defined rule.

```
#black
{
   color: #000000;
}
```

This rule renders the content in black for every element with id attribute set to black in our document. You can make it a bit more particular. For example -

```
h1#black
{
   color: #000000;
}
```

This rule renders the content in black for only <h1> elements with id attribute set to black.

The true power of id selectors is when they are used as the foundation for descendant selectors, For example:

```
#black h2
 {
   color: #000000;
}
```

In this example all level 2 headings will be displayed in black color when those headings will lie with in tags having id attribute set to black.

Q31. What is meant by style sheet ?

Ans :

Cascading Style Sheets (CSS) provide easy and effective alternatives to specify various attributes for the HTML tags. Using CSS, you can specify a number of style properties for a given HTML element. Each property has a name and a value, separated by a colon (:). Each property declaration is separated by a semi-colon (;).

```
<html>
      <head>
            <title>HTML CSS</title>
                  </head>
                  <body>
                        <p><font color = “green” size = “5”>Hello, World!</font></p>
                  </body>
                  </html>
```

We can re-write above example with the help of Style Sheet as follows

```
<html>
<head>
      <title>HTML CSS</title>
</head>
<body>
<p style = “color:green; font-size:24px;” >Hello, World!</p>
</body>
</html>
```

OUTPUT :

```
Hello, World!
```

1.9 Types of CSS

1.9.1 In line styles, style element, External Style sheet

Q32. Explain the different types of CSS with an examples.

Ans : **(Imp.)**

CSS supports mainly 3 types of style sheets.

1. External style sheet
2. Internal style sheet (embedded style sheet)
3. Inline style sheet

1. External Style Sheet

If you need to use your style sheet to various pages, then its always recommended to define a common style sheet in a separate file. A cascading style sheet file will have extension as **.css** and it will be included in HTML files using <link> tag.

Example

Consider we define a style sheet file **style.css** which has following rules

```
.red
      {
      color: red;
      }
            .thick {
            font-size:20px;
      }
.green
{
color:green;
}
```

Here we defined three CSS rules which will be applicable to three different classes defined for the HTML tags. I suggest you should not bother about how these rules are being defined because you will learn them while studying CSS. Now let’s make use of the above external CSS file in our following HTML document -

```
<html>
      <head>
            <title>HTML External CSS</title>
            <link rel = “stylesheet” type = “text/css” href = “/html/style.css”>
      </head>
      <body>
      <p class = “red”>This is red</p>
      <p class = “thick”>This is thick</p>
```

```
<p class = "green">This is green</p>
<p class = "thick green">This is thick and
green</p>
</body>
</html>
```

OUTPUT

```
This is red
This is thick
This is green
This is thick and green
```

2. Internal Style Sheet (or) Embedded Style Sheet

If you want to apply Style Sheet rules to a single document only, then you can include those rules in header section of the HTML document using <style> tag.

Rules defined in internal style sheet overrides the rules defined in an external CSS file.

Example

Let's re-write above example once again, but here we will write style sheet rules in the same HTML document using <style> tag

```
<html>
    <head>
        <title>HTML Internal CSS</title>
        <style type = "text/css">
        .red {
        color: red;
    }
    .thick{
    font-size:20px;
    }
    .green {
    color:green;
    }
    </style>
    </head>
    <body>
    <p class = "red">This is red</p>
    <p class = "thick">This is thick</p>
    <p class = "green">This is green</p>
    <p class = "thick green">This is thick and green</p>
    </body>
    </html>
```

OUTPUT

```
This is red
This is thick
This is green
This is thick and green
```

3. Inline Style Sheet

You can apply style sheet rules directly to any HTML element using style attribute of the relevant tag. This should be done only when you are interested to make a particular change in any HTML element only.

Rules defined inline with the element overrides the rules defined in an external CSS file as well as the rules defined in <style> element.

Example

Let's re-write above example once again, but here we will write style sheet rules along with the HTML elements using style attribute of those elements.

```
<!DOCTYPE html>
<html>
        <head>
        <title>HTML Inline CSS</title>
        </head>
<body>
        <p style = "color:red;">This is red</p>
        <p style = "font-size:20px;">This is thick</p>
        <p style = "color:green;">This is green</p>
        <p style = "color:green;font-size:20px;">This is thick and green</p>
</body>
</html>
```

Output

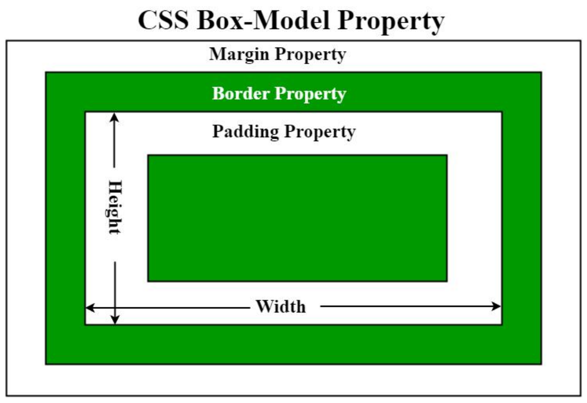
```
This is red
This is thick
This is green
This is thick and green.
```

1.10 Box Model

Q33. Explain in detail the CSS box model.

Ans : **(Imp.)**

The CSS box model is a container that contains multiple properties including borders, margin, padding, and the content itself. It is used to create the design and layout of web pages. It can be used as a toolkit for customizing the layout of different elements. The web browser renders every element as a rectangular box according to the CSS box model. Box-Model has multiple properties in CSS. Some of them are given below:



(i) Content Area: This area consists of content like text, images, or other media content. It is bounded by the content edge and its dimensions are given by content-box width and height.

(ii) Padding Area: It includes the element's padding. This area is actually the space around the content area and within the border-box. Its dimensions are given by the width of the padding-box and the height of the padding-box.

(iii) Border Area: It is the area between the box's padding and margin. Its dimensions are given by the width and height of the border.

(iv) Margin Area: This area consists of space between border and margin. The dimensions of the Margin area are the margin-box width and the margin-box height. It is useful to separate the element from its neighbors.

Q34. Discuss about properties and values in detail with their syntaxes?

Ans :

A number of properties of text can be altered. These can be grouped together. We list the properties in useful groups and give some of the options that can alter the bet way of discovering hoe styles work is to play around with some of these properties. Styles are normally saved in external.css files. Cascading style sheet(CSS) are a collection of formatting rules that control the appearance of Content in a web page, they are very useful for maintaining a website since its appearance (controlled by properties of HTML tags) can be managed by just one file.CSS STYLES also enhance your style look, accessibility and reduces file size. Another main advantage is reusability- instead of defining the properties of fonts, backgrounds, borders, bullets, uniform tags, etc.

COLOR AND URL'S

Color

A color value is a keyword or a numerical RGB specification. The original set of keyword colors was aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white and yellow. These have been enhanced as a result of the requirements from other specifications. The set of keywords with the corresponding RGB values given are likely to be supported by a browser.

<color> ::= <keyword> | # <hex> <hex> <hex> | # <hex> <hex> <hex><hex> <hex><hex> | rgb (<int><int><int>) | rgb (<percentage. <percentage > <percentage>)

RGB colors can be defined in one of four ways :

1. #rrggbb(e.g., #00ff00)
2. #rgb (e.g., #0f0)
3. rgb(x,x,x) where x is an integer between 0 and 255
4. rgb(y%,y%,y%) where y is a number between 0.0 and 100.0

URL

<url> ::= url(<whitespace>? [' | "]? <characters in>

A URL value is of the form: url(xyz) where xyz is the URL. The URL may be quoted with either single (') or double (") quotes and may have whitespace before the initial quote or after the final quote. Parentheses, commas, spaces, single quotes, or double quotes in the URL must be escaped with a backslash. Partial URLs are interpreted relative to the style sheet source, not to the HTML source.

body {background: url("circle.gif")}

body {background: url("http://www.clrc. images/monkey.gif") }

Q35. Explain font properties in CSS?

Ans :

Font Family

Font-family: [<family-name> | <generic-family>] [[<family-name> | <generic-family>]]*

<family-name>::= serif | sans-serif | cursive | fantasy | monospace

The font to use is set by the font-family property. Instead of defining a single font, a sequence of fonts should be listed. The browser will use the first if it is available but try the second and so on. The value can either be a specific font name or a generic font family.

Good practice is to define one or two specific fonts followed by the generic family name in case the first choices are not present. Thus a sample font-family declaration might look like

p { font-family: "New Century Schoolbook", Times, serif }

Any font name containing spaces (multiple words) must be quoted, with either single or double quotes. That is why the first font name is quoted but Times is not. The first two requests are for specific fonts **New Century Schoolbook** and **Times.** As both are serif fonts, the generic font family listed as a backup is the **serif** font family. In this case, Times will be used if New Century Schoolbook is not available, and a serif font if Times is not available.

FONT STYLE

Font-style : normal

The font-style property has one of three values: **normal,italic** or **oblique** (slanted). A sample style sheet might be

h3 {font-style: italic}
p {font-style: normal}

Font Size (set size of text)

font-size: <absolute-size> | <relative-size> | <length> | <percentage>

<absolute-size> ::= xx-small | x-small | small | medium | large | x-large | xx-large
<relative-size> ::= larger | smaller

The font-size property sets the size of the displayed characters. The absolute-size values are small, medium and large with additional possibilities like x-small (extra small) and xx-small . Relative sizes are smaller and larger . They are relative to any inherited value for the property. The length value defines the precise height in units like 12pt or 1in . Percentage values are relative to the inherited value.

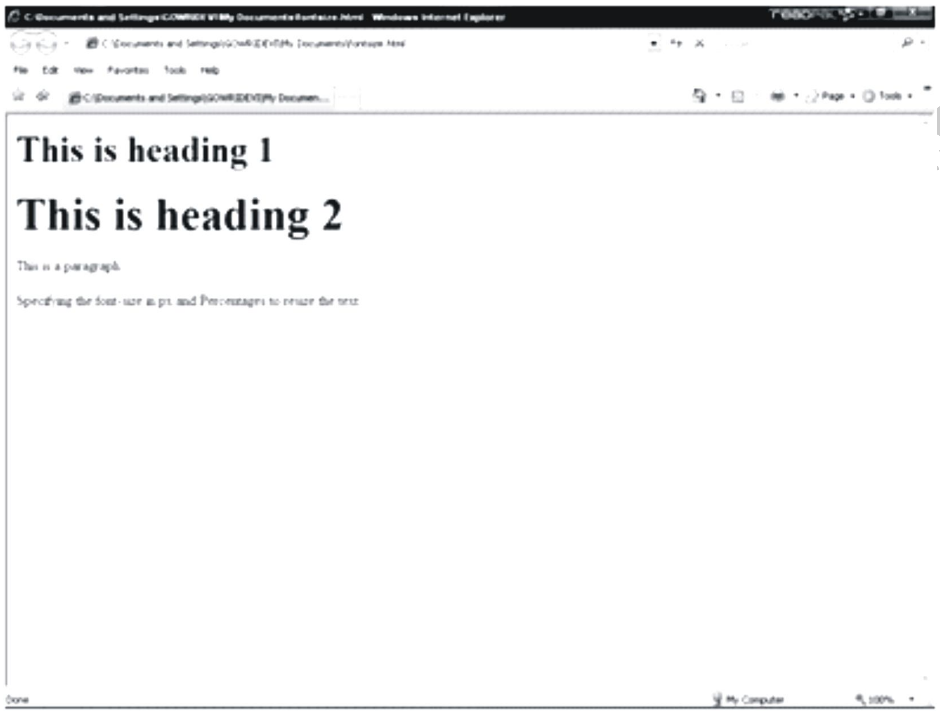
h1 { font-size: large }
h2 { font-size: 12pt }
h3 { font-size: 5cm }
p { font-size: 10pt }
li { font-size: 150%}

Example

```
<html>
<head>
<style type="text/css">
```

```
h1 {font-size:250%;}
h2 {font-size:50px;}
p {font-size:100%;}
</style>
</head>
<body text=blue>
<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<p>This is a paragraph.</p>
<p>Specifying the font-size in px and Percentages to resize the text </p>
</body>
</html>
```

Output



This is heading 1

This is heading 2

This is a paragraph.

Specifying the font-size in px and Percentages to resize the text

Q36. Discuss about color and background properties in CSS ?

Ans :

Color

color: <color>

The color property sets the foreground colour of a text element.

Background Color

Background-color: <color> | transparent

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

body { background-color: white }

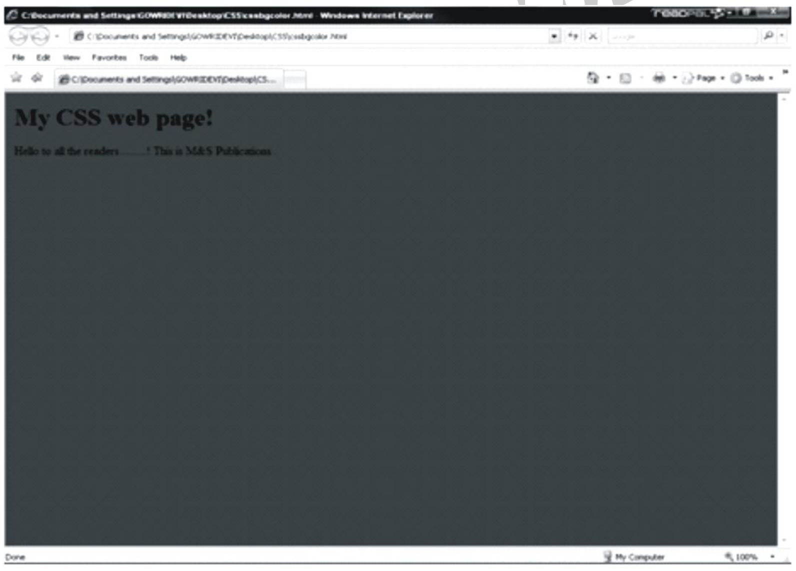
h1 { background-color: #000080 }

The value **transparent** indicates that whatever is behind the element can be seen. For example if the background to the **body** element was specified as being red, a block-level element with **background-color** set to **transparent** would appear with a red background.

Example

```
<html>
<head>
<style type="text/css">
body
{
    background-color:"#af2400";
}
</style>
</head>
<body>
<h1>My CSS web page!</h1>
<p>Hello to all the readers.........! This is M&S Publications .</p>
</body>
</html>
```

Output



My CSS web page!

Hello to all the readers.........! This is M&S Publications

Background Image

background-image: <url> | none

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

body { background-image: url("/images/monkey.gif") }

p { background-image: url("http://www.cclrc.com/pretty.png") }

h1 { background-image: none }

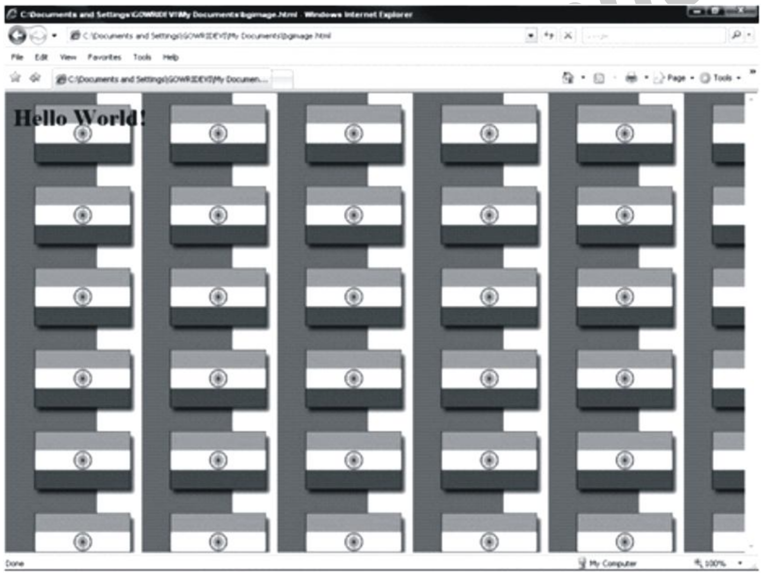
When a background image is defined, a compatible background colour should also be defined for those users who do not load the image or to cover the case when it is unavailable. If parts of the image are transparent, the background colour will fill these parts.

```
<ul style="background-image:url('wall.png')">
<li>An item</li>
<li>Another</li>
<li>And third</li>
</ul>
```

Example

```
<html>
<head>
<style type="text/css">
body {background-image:url('image.jpg');}
</style>
</head>
<body>
<h1>Hello World!</h1>
</body>
</html>
```

Output



Q37. Explain different ways to style the text in CSS?

Ans :

Word Spacing

The word-spacing property defines additional spacing between words. Negative values are permitted which allows the letters to be closed up or even overlap. Word spacing may be influenced by alignment.

p { word-spacing: 0.4em }

h1 { word-spacing: -0.2em }
h2 { word-spacing: normal }

Letter Spacing

letter-spacing: normal | <length>

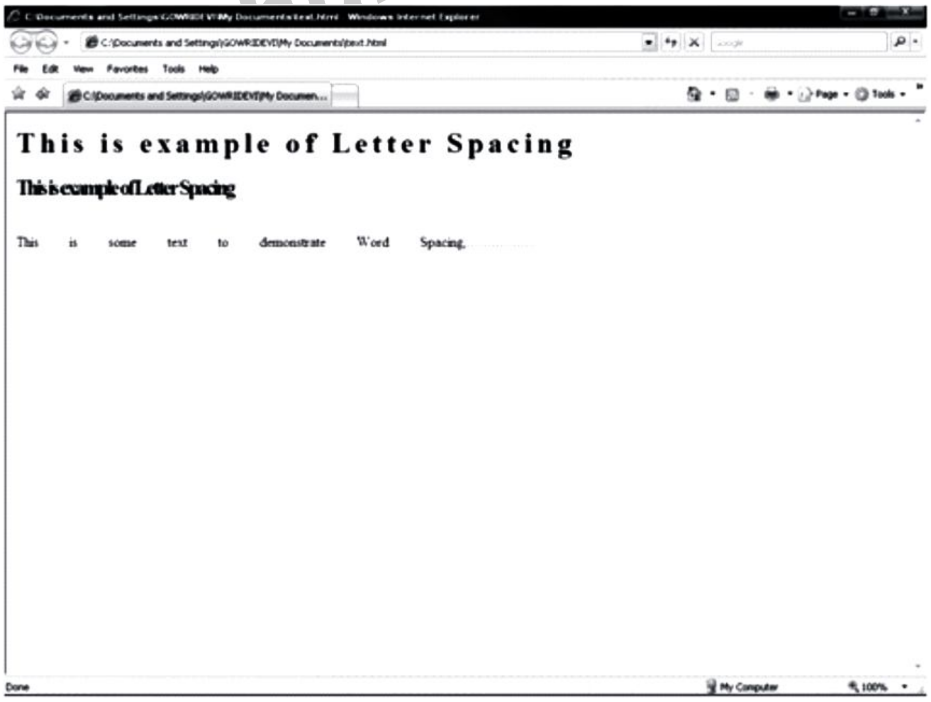
The letter-spacing property defines additional spacing between characters. Negative values are permitted. A value of normal will still allow justification to take place.

h {letter-spacing: 0.2em }
p {letter-spacing: -0.1cm }

Example

```
<html>
<head>
<style type=”text/css”>
h1 {letter-spacing:5px;}
h2 {letter-spacing:-3px;}
p { word-spacing:30px;}
</style>
</head>
<body>
<h1>This is example of Letter Spacing</h1>
<h2>This is example of Letter Spacing</h2>
<br>
<p> This is some text to demonstrate Word Spacing,.................. </p>
<br><br>
</body>
</html>
```

Output


C:\Documents and Settings\GOWRIDEVI\My Documents\text.html - Windows Internet Explorer

This is example of Letter Spacing

This is example of Letter Spacing

This is some text to demonstrate Word Spacing..........

Vertical Alignment

vertical-align: baseline | sub | super | top | text-top | middle | bottom | text-bottom | <percentage>

The vertical-align property defines the vertical positioning of an inline element (such as some text or even an image) relative to the current baseline. It could be used to produce x^2. The letter x in this case would be called the **parent** and the power 2 would be called the **child.**

The value may be a percentage of the element's line-height property. It specifies how much the element's baseline is raised above the parent's baseline. Negative values are permitted.

Text Transformation

Text-transform: capitalize | uppercase | lowercase | none

The text-transform property allows text to be transformed by one of four properties

Notation	Meaning/ Description
Capitalize	Capitalizes the first character of each word
Uppercase	Capitalizes all the character of each word
Lowercase	Uses small letters for all characters of each word
None	Leaves characters as defined

Example

```
h1 { text-transform: uppercase }
h2 { text-transform: capitalize }
```

Text Alignment (set justification)

text-align: left | right

The text-align property defines the horizontal alignment of text.

```
h1 {text-align: center}
h2 {text-align: left}
h3 {text-align: right}
p  {text-align: justify }
```

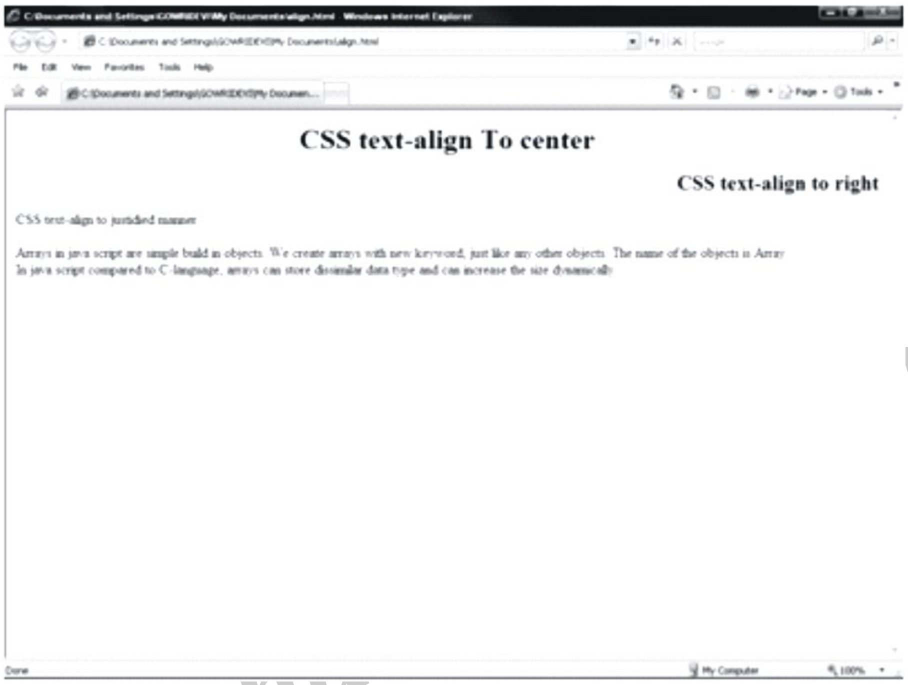
Example

```
<html>
<head>
<style type="text/css">
h1 {text-align:center;}
h2 {text-align:right;}
p {text-align:justify;}
</style>
</head>
<body>
<h1>CSS text-align To center</h1>
<h2>CSS text-align to right</h2>
<p> CSS text-align to justidied manner <br><br>
```

Arrays in java script are simple build in objects. We create arrays with new keyword, just like any other objects. The name of the objects is Array. <br>In java script compared to C-language, arrays can store dissimilar data type and can increase the size dynamically.

</p>
</body>
</html>

Output



C:\Documents and Settings\GOWRIDEVI\My Documents\align.html - Windows Internet Explorer
File Edit View Favorites Tools Help
Page Tools
Done
My Computer
100%

CSS text-align To center

CSS text-align to right

CSS text-align to justified manner

Arrays in java script are simple build in objects. We create arrays with new keyword, just like any other objects. The name of the objects is Array
In java script compared to C-language, arrays can store dissimilar data type and can increase the size dynamically

Width, Height of Images

width: <length> | <percentage> | auto
height: <length> | auto

These two properties apply to both block-level elements and replaced elements such as images. The width or height can be specified and the aspect ratio maintained by setting the other one to **auto** . If both are specified precisely, the scaling will not maintain the aspect ratio. If both are specified as auto, the intrinsic size of the image is used.

img {width: 100px}

img {height: auto}

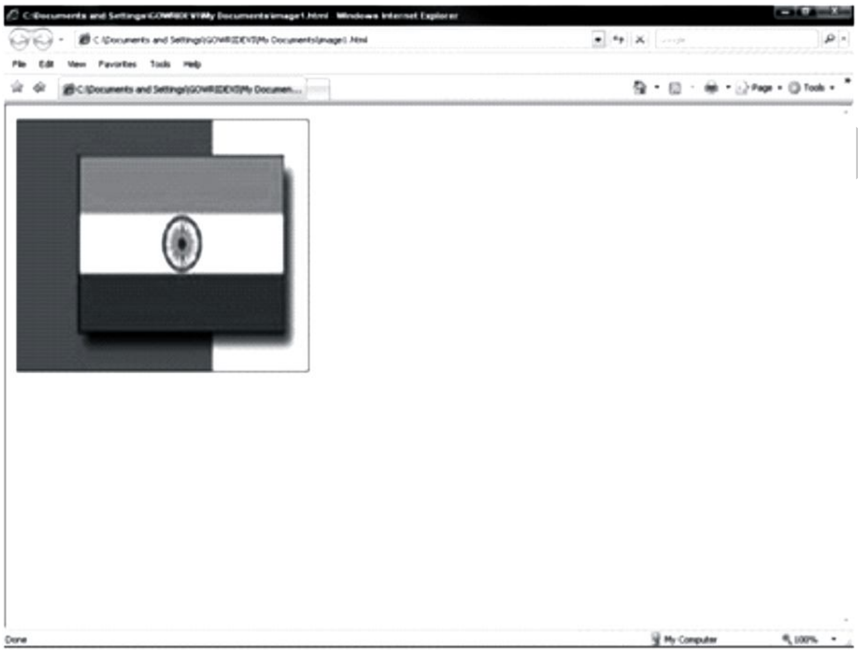
Percentage values for width refer to the parent element's width. Negative values are not allowed. Users should be aware that scaling by arbitrary amounts can severely degrade the image quality as can changing the aspect ratio so some care should be taken when using this property.

Example

<html>
<head>
<style type="text/css">
img

```
{
margin: 2px;
border: 1px solid #0000ff;
height: 300;
width: 350;
}
</style>
</head>
<body>
<img src="image.jpg">
</body>
</html>
```

Output



Clear

clear: none | left |

The clear property can be used by box-level elements to ensure that a previous floating element is not adjacent to one of its sides. A value of **left** applied to an element such as **h1** would ensure that any images with float set to **left** defined above it in the page would not appear to its left. The element would effectively move below the image (by adding additional space above the **h1** element).

h1: {clear: left }

The value **right** ensures that it is does not overlap with any images floated to the right. The value **both** is equivalent to setting both **left** and **right** . The property can also be used by inline elements.

Q38. Write a short notes on CSS Classes with Example ?

Ans :

A few lessons ago, we learned about *selectors*. You may recall that selectors are the things we apply a style against. In CSS, classes allow you to apply a style to a given *class* of an element. To do this, you link the element to the style by declaring a style for the class, then assigning that class to the element.

CSS Class Syntax

.class-name { property:value; }

We declare a CSS class by using a dot (.) followed by the class name. You make up the class name yourself. After the class name you simply enter the properties/values that you want to assign to your class.

If you want to use the same class name for multiple elements, but each with a different style, you can prefix the dot with the HTML element name.

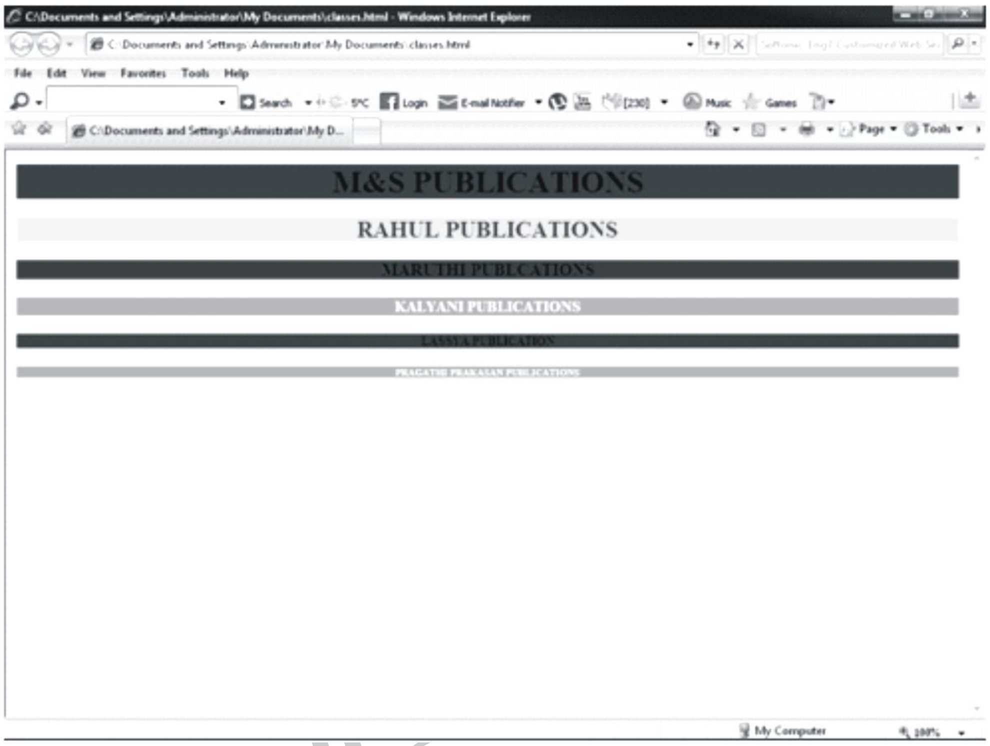
html-element-name.class-name { property:value; }

Example

```
<Html>
<Head>
<Style>
.first
{
background-color: yellow; color:red
}
.second
{
background-color: red;color: blue
}
.third
{
background-color: cyan;color: white;
}
.head
{
background-color: Green; color:blue;
}
</Style>
</Head>
<Body>
<Center>
<h1 class="head">M&S PUBLICATIONS</h1>
<h2 class="first">RAHUL PUBLICATIONS</h2>
<h3 class="second">MARUTHI PUBLCATIONS</h3>
<h4 class="third">KALYANI PUBLICATIONS</h4>
<h5 class="head">LASSYA PUBLICATION</h5>
```

```
<h6 class="third">PRAGATHI PRAKASAN PUBLICATIONS </h6>
</Center>
</Body>
</Html>
```

Output


C:\Documents and Settings\Administrator\My Documents\classes.html - Windows Internet Explorer
File Edit View Favorites Tools Help
My Computer
100%

M&S PUBLICATIONS

RAHUL PUBLICATIONS

MARUTHI PUBLCATIONS

KALYANI PUBLICATIONS

LASSYA PUBLICATION

PRAGATHI PRAKASAN PUBLICATIONS

CSS ID

IDs allow you to assign a unique identifier to an HTML element. This allows you to define a style that can only be used by the element you assign the ID to.

Syntax

#id-name { property:value; }

The syntax for declaring a CSS ID is the same as for classes - except that instead of using a dot, you use a hash (#).

similar to classes, if you want to use the same id name for multiple elements, but each with a different style, you can prefix the hash with the HTML element name.

1.10.1 User Style Sheets

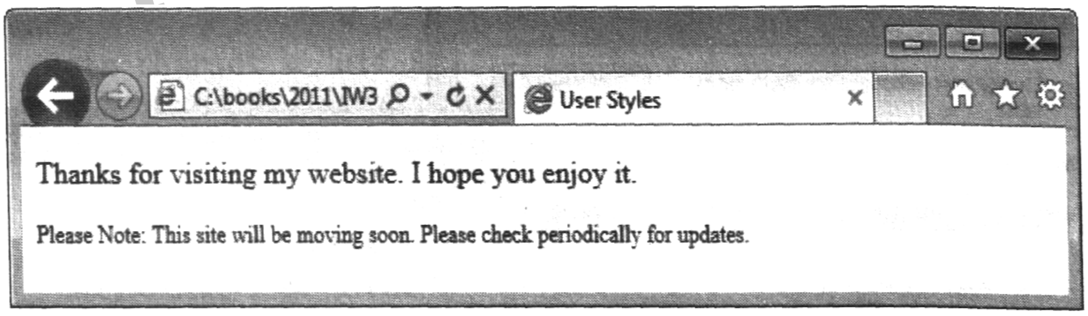
Q39. Explain different ways of user style sheets.

Ans :

Users can define their own user style sheets to format pages based on their preferences, tor example, people with visual impairments may want to increase the page's text size. You need to be careful not to inadvertently override user preferences with defined styles. This section discusses possible conflicts between author styles and user styles. For the purpose of this section, we demonstrate the concepts in Internet Explorer.

Figure contains an author style. The font-size is set to 9pt for all <p> tags that have class note applied to them.

```
<! DOCTYPE html>
<!— Fig. 4.18: user_absolute.html —>
<!— pt measurement for text size. —>
<html>
    <head>
        <meta charset = “utf-8”>
        <title>User Styles</title>
        <style type = “text/css”>
            .note { font-size: 9pt; }
        </style>
    </head>
    <body>
        <p>Thanks for visiting my website. I hope you enjoy it.
        </p><p class =, “note”>Please Note: This site will be
        moving soon. Please check periodically for updates.</p>
    </body>
</html>
```



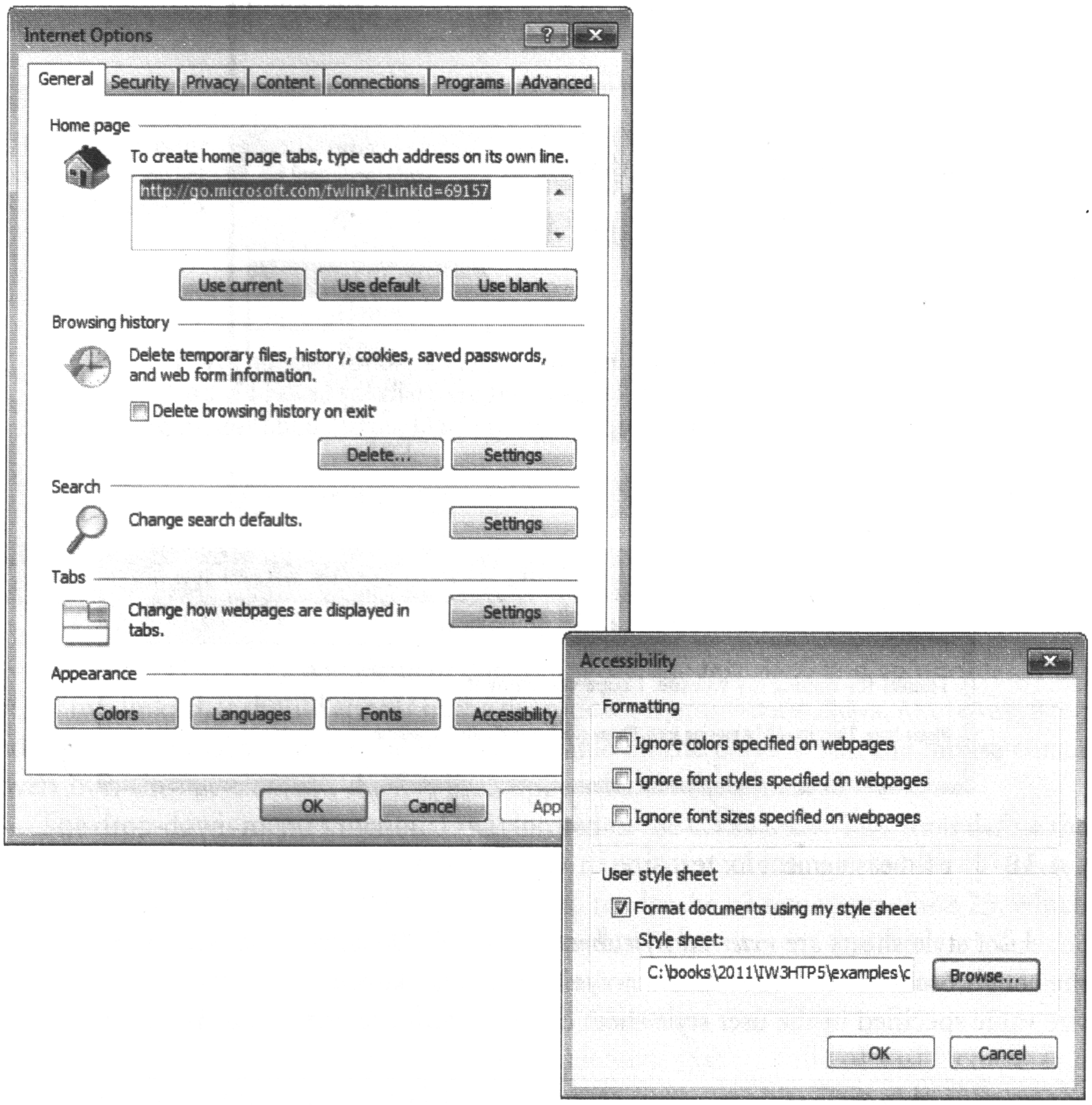
Thanks for visiting my website. I hope you enjoy it.

Please Note: This site will be moving soon. Please check periodically for updates.

Fig.: Measurement for text size

Adding a User Style Sheet

User style sheets are not linked to a document; rather, they're set in the browser's options. To add a user style sheet in IE9, select Internet Options..., located in the Tools menu. In the Internet Options dialog (Fig.) that appears, click Accessibility..., check the Format documents using my style sheet checkbox, and type the location of the user style sheet. IE9 applies the user style sheet to any document it loads.


Internet Options

General | Security | Privacy | Content | Connections | Programs | Advanced

Home page

To create home page tabs, type each address on its own line.

http://go.microsoft.com/fwlink/?LinkId=69157

Use current | Use default | Use blank

Browsing history

Delete temporary files, history, cookies, saved passwords, and web form information.

[ ] Delete browsing history on exit

Delete... | Settings

Search

Change search defaults. Settings

Tabs

Change how webpages are displayed in tabs. Settings

Appearance

Colors | Languages | Fonts | Accessibility

OK | Cancel | App

Accessibility

Formatting

[ ] Ignore colors specified on webpages

[ ] Ignore font styles specified on webpages

[ ] Ignore font sizes specified on webpages

User style sheet

[✓] Format documents using my style sheet

Style sheet:

C:\books\2011\IW3HTP5\examples\c

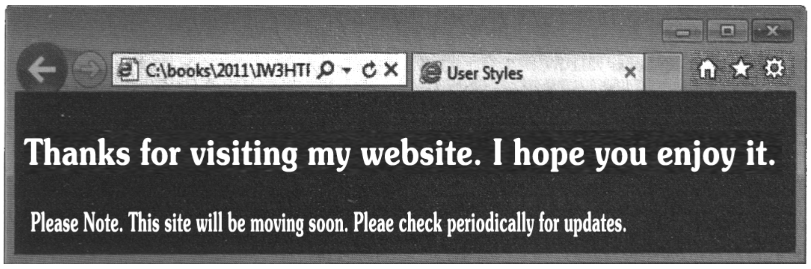
Browse...

OK | Cancel

Fig.: User style sheet in Internet explorer 9.

Defining font-size in a User Style Sheet

In the preceding example, if the user defines font-size in a user style sheet, the author style has a higher precedence and *overrides* the user style. The 9pt font specified in the author style sheet overrides the 20pt font specified in the user style sheet. This small font may make pages difficult to read, especially for individuals with *visual impairments* You can avoid this problem by using relative measurements (e.g., em or ex) instead of absolute measurements, such as pt. Figure changes the font-size property to use a relative measurement (line 10) that does *not* override the user style set in Figure. Instead, the font size displayed is relative to the one specified in the user style sheet. In this case, text enclosed in the <p> tag displays as 20pt, and <p> tags that have the class note applied to them are displayed in 15pt (.75 times 20pt).

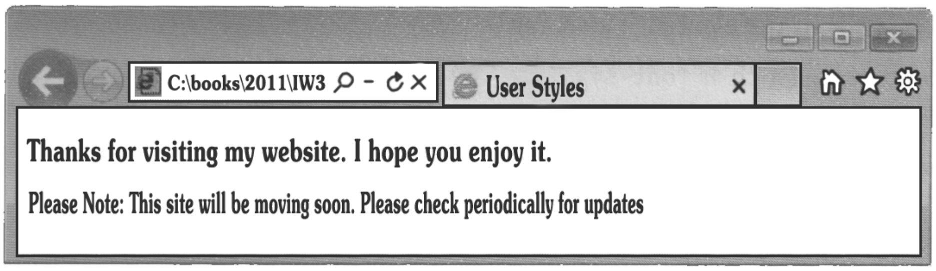


Thanks for visiting my website. I hope you enjoy it.

Please Note. This site will be moving soon. Pleae check periodically for updates.

Fig.: em measurement for text size

```
<!DOCTYPE html>
<! -- Figure user_relative.html -->
<!-- em measurement for text size. -->
<html>
    <head>
        <meta charset = "utf-8">
        <title>User  Styles</title>
        <style type = "text/css">
            .note { font-size: .75em; }
        </style>
    </head>
    <body>
        <p>Thanks for visiting my website. I hope you enjoy it.
        </pxp class = "note'VPlease Note: This site will be
        moving soon. Please check periodically for updates.</p>
    </body>
</html>
```

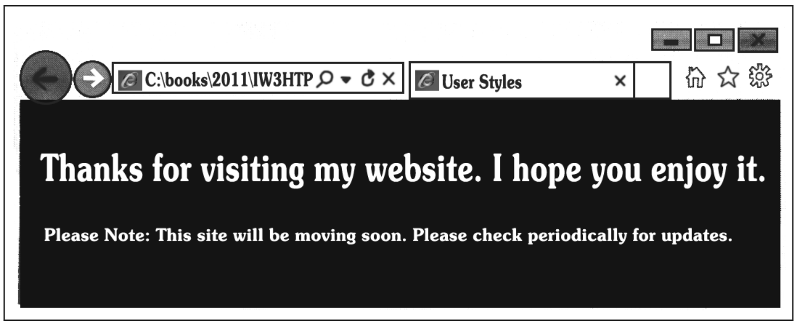


Thanks for visiting my website. I hope you enjoy it.

Please Note: This site will be moving soon. Please check periodically for updates

Fig.: em measurement for text size

Figure displays the web page from Figure in Internet Explorer with the user style sheet from Figure applied. Note that the second line of text displayed is larger than the same line of text in Figure.



Thanks for visiting my website. I hope you enjoy it.

Please Note: This site will be moving soon. Please check periodically for updates.

Fig.: User style sheet applied with em measurement

UNIT II	**Object Model and Collections :** Object referencing, collections all, children frames, navigator object. **Event Model:** ONCLICK, ONLOAD, Error Handling, ON ERRORS ONMUOUSEMOVE, ONMOUSEOVER, ONMOUSEOUT, ON FOCUS, ONBLUR, ONSUBMIT. **Dynamic HTML:** Filters and transitions, Data binding with Tabular data control binding to IMO, TABLE, Structured graphics, Active controls.

2.1 Object Model

2.1.1 Object referencing

Q1. What is a object model ? Explain its Properties and functions ?

Ans : **(Imp.)**

Java Script is based on a simple object-oriented paradigm. An object is a construct with properties that are Java Script variables. Properties can be other objects. Functions associated with an object are known as the object's *methods*.

In addition to objects that are built into the Navigator client and the Live Wire server, you can define your own objects.

- Objects and Properties
- Functions and Methods
- Creating New Objects

Properties

A JavaScript object has properties associated with it. You access the properties of an object with a simple notation:

object Name. property Name

Both the object name and property name are case sensitive. You define a property by assigning it a value. For example, suppose there is an object named *myCar* (we'll discuss how to create objects later-for now, just assume the object already exists). You can give it properties named make, model, and year as follows:

```
myCar.make = "Ford"
myCar.model = "Mustang"
myCar.year = 69;
```

You can also refer to these properties using an array notation as follows:

```
mycar["make"] = "Ford
myCar["model"] = "Mustang"
myCar["year"] = 69;
```

This type of an array is known as an associative array, because each index element is also associated with a string value. To illustrate how this works, the following function displays the properties of the object, when you pass the object and the object's name as arguments to the function:

```
function show_props(obj, obj_name) {
   var result = “”
   for (var i in obj)
      result += obj_name + “.” + i + “ = “ + obj[i] + “\n”
   return result;
}
```

So, the function call show_props(myCar, “myCar”) would return the following:

```
myCar.make = Ford
myCar.model = Mustang
myCar.year = 67
```

You may also define properties using ordinal numbers, for example:

```
temp[0] = 34
temp[1] = 42
temp[2] = 56
```

These statements create three properties of the object *temp*, and you must refer to these properties as temp[*i*], where *i* is an integer between 0 and 2.

Functions

Functions are one of the fundamental building blocks in JavaScript. A function is a JavaScript procedure a set of statements that performs a specific task.

A function definition consists of the function keyword, followed by

- The name of the function
- A list of arguments to the function, enclosed in parentheses, and separated by commas
- The JavaScript statements that define the function, enclosed in curly braces, {...}

In a Navigator application, you can use any functions defined in the current page. It is generally a good idea to define all your functions in the HEAD of a page. When a user loads the page, the functions will then be loaded first.

The statements in a function can include other function calls defined for the current application.

For example, here is the definition of a simple function named pretty_print:

```
function pretty_print(string) {
  document.write(“<HR><P>” + string)
}
```

This function takes a string as its argument, adds some HTML tags to it using the concatenation operator (+), then displays the result to the current document.

Defining a function does not execute it. You have to *call* the function for it to do its work. For example, you could call the pretty_print function as follows:

```
<SCRIPT>
pretty_print("This is some text to display")
</SCRIPT>
```

The arguments of a function are not limited to just strings and numbers. You can pass whole objects to a function, too. The show_props function from the previous section is an example of a function that takes an object as an argument.

The arguments of a function are maintained in an array. Within a function, you can address the parameters passed to it as follows:

function Name. arguments[i]

Where *functionName* is the name of the function and *i* is the ordinal number of the argument, starting at zero. So, the first argument passed to a function named myfunc would be myfunc.arguments[0]. The total number of arguments is indicated by the variable arguments.length.

A function can even be recursive, that is, it can call itself. For example, here is a function that computes factorials:

```
function factorial(n) {
  if ((n == 0) || (n == 1))
    return 1
  else {
    result = (n * factorial(n-1) )
    return result
  }
}
```

You could then display the factorials of one through five as follows:

```
for (x = 0; x < 5; x++) {
   document.write(x, " factorial is ", factorial(x))
   document.write("")
}
```

The results would be:

```
0 factorial is 1
1 factorial is 1
2 factorial is 2
3 factorial is 6
4 factorial is 24
5 factorial is 120
```

Functions with Variable Numbers of Arguments

You can call a function with more arguments than it is formally declared to accept using the *arguments* array. This is often useful if you don't know beforehand how many arguments will be passed to the function. You can use arguments.length to determine the number of arguments actually passed to the function, a and then treat each argument using the arguments array.

For example, consider a function defined to create HTML lists. The only formal argument for the function is a string that is "U" if the list is to be unordered (bulleted) or "O" if the list is to be ordered (numbered). The function is defined as follows:

```
function list(type) {
  document.write("<" + type + "L>")              // begin list
  for (var i = 1; i < list.arguments.length; i++)  // iterate through arguments
    document.write("<LI>" + list.arguments[i])
  document.write("</" + type + "L>")             // end list
}
```

You can pass any number of arguments to this function and it will then display each argument as an item in the indicated type of list. For example, the following call to the function:

list("o", "one", 1967, "three", "etc, etc...")

results in this output:

1. one
2. 1967
3. three
4. etc, etc...

Defining Methods

A *method* is a function associated with an object. You define a method in the same way as you define a standard function. Then, use the following syntax to associate the function with an existing object:

object.methodname = function_name

where *object* is an existing object, *methodname* is the name you are assigning to the method, and *function_name* is the name of the function.

You can then call the method in the context of the object as follows:

object.methodname(params);

Using this for Object References

JavaScript has a special keyword, this, that you can use to refer to the current object. For example, suppose you have a function called *validate* that validates an object's value property, given the object, and the high and low values:

```
function validate(obj, lowval, hival) {
  if ((obj.value < lowval) || (obj.value > hival))
    alert("Invalid Value!")
}
```

Then, you could call *validate* in each form element's on Change event handler, using this to pass it the form element, as in the following example:

```
<INPUT TYPE = "text" NAME = "age" SIZE = 3 onChange="validate(this, 18, 99)">
```

In general, in a method this refers to the calling object.

Creating New Objects

Both client and server JavaScript have a number of predefined objects. In addition, you can create your own objects. Creating your own object requires two steps:

- Define the object type by writing a function.
- Create an instance of the object with new.

To define an object type, create a function for the object type that specifies its name, and its properties and methods. For example, suppose you want to create an object type for cars. You want this type of object to be called *car*, and you want it to have properties for make, model, year, and color. To do this, you would write the following function:

```
function car(make, model, year) {
  this.make = make;
  this.model = model;
  this.year = year;
}
```

Notice the use of this to assign values to the object's properties based on the values passed to the function.

Now you can create an object called *mycar* as follows:

```
mycar = new car("Eagle", "Talon TSi", 1993);
```

This statement creates *mycar* and assigns it the specified values for its properties. Then the value of mycar.make is the string "Eagle", mycar.year is the integer 1993, and so on.

You can create any number of *car* objects by calls to new.

For example,

```
kenscar = new car("Nissan", "300ZX", 1992)
```

An object can have a property that is itself another object. For example, suppose I define an object called *person* as follows:

```
function person(name, age, sex) {
  this.name = name;
  this.age = age;
  this.sex = sex;
}
```

And then instantiate two new *person* objects as follows:

```
rand = new person("Rand McNally", 33, "M")
```

```
ken = new person("Ken Jones", 39, "M")
```

Then we can rewrite the definition of *car* to include an owner property that takes a *person* object, as follows :

```
function car(make, model, year, owner) {
  this.make = make;
  this.model = model;
  this.year = year;
  this.owner = owner;
}
```

To instantiate the new objects, you then use the following:

```
car1 = new car("Eagle", "Talon TSi", 1993, rand);
car2 = new car("Nissan", "300ZX", 1992, ken)
```

Notice that instead of passing a literal string or integer value when creating the new objects, the above statements pass the objects rand and ken as the arguments for the owners. Then if you want to find out the name of the owner of car2, you can access the following property:

car2.owner.name

Note that you can always add a property to a previously defined object. For example, the statement:

car1.color = "black"

adds a property color to car1, and assigns it a value of "black". However, this does not affect any other objects. To add the new property to all objects of the same type, you have to add the property to the definition of the *car* object type.

Defining Methods

You can define methods for an object type by including a method defnition in the object type definition. For example, suppose you have a set of image GIF files, and you want to define a method that displays the information for the cars, along with the corresponding image. You could define a function such as:

```
function displayCar()
{
  var result = "A Beautiful " + this.year + " " + this.make + " " + this.model;
  pretty_print(result)
}
```

where *pretty_print* is the previously defined function to display a string. Notice the use of this to refer to the object to which the method belongs.

You can make this function a method of *car* by adding the statement

this.displayCar = displayCar;

to the object definition. So, the full definition of *car* would now look like:

```
function car(make, model, year, owner) {
  this.make = make;
  this.model = model;
  this.year = year;
  this.owner = owner;
  this.displayCar = displayCar;
}
```

Then you can call this new method as follows:

```
car1.displayCar()
car2.displayCar()
```

2.2 Collections

Q2. Describe about Java collections framework?

Ans : **(Imp.)**

Java Collections Framework is one of the core parts of the Java programming language. Collections are used in almost every programming language. Most of the programming languages support various type of collections such as List, Set, Queue, Stack, etc.

Collections are like containers that group multiple items in a single unit. For example, a jar of chocolates, a list of names, etc.

Collections are used in every programming language and when Java arrived, it also came with few Collection classes – Vector, Stack, Hashtable, Array

1. Interfaces

Java Collections Framework interfaces provides the abstract data type to represent collection.

Java.util.Collection is the root interface of Collections Framework. It is on the top of the Collections framework hierarchy. It contains some important methods such as size(), iterator(), add(), remove(), clear() that every Collection class must implement.

Some other important interfaces are java.util.List, java.util.Set, java.util.Queue and java.util.Map. The Map is the only interface that doesn't inherit from the Collection interface but it's part of the Collections framework. All the collections framework interfaces are present in java.util package.

2. Implementation Classes

Java Collections framework provides implementation classes for core collection interfaces. We can use them to create different types of collections in the Java program.

Some important collection classes are Array List, Linked List, Hash Map, Tree Map, Hash Set, and Tree Set. These classes solve most of our programming needs but if we need some special collection class, we can extend them to create our custom collection class.

Java 1.5 came up with thread-safe collection classes that allowed us to modify Collections while iterating over them. Some of them are Copy On Write Array List, Concurrent Hash Map, Copy On Write Array Set. These classes are in java.util.concurrent package.

All the collection classes are present in java.util and java.util.concurrent package.

3. Algorithms

Algorithms are useful methods to provide some common functionalities such as searching, sorting and shuffling.

Benefits

Java Collections framework have following benefits:

- **Reduced Development Effort:** It comes with almost all common types of collections and useful methods to iterate and manipulate the data. So we can concentrate more on business logic rather than designing our collection APIs.
- **Better Quality:** Using core collection classes that are well-tested increases our program quality rather than using any home-developed data structure.
- **Reusability and Interoperability**
- **Reduce effort:** to maintain because everybody knows Collection API classes.

2.3 Navigator Object

Q3. Discuss in brief about Navigator.

Ans : **(Imp.)**

The JavaScript window object property of *navigator* is used refer to the Navigator object, which contains all information related to browser, it vendor , version, plugins etc.

Earlier Navigator object was used only to determine whether clients systems were running, Internet Explorer or Netscape i.e *browser-sniffing*

Currently the navigator object can be used to check the Browser Version, whether Java is enabled, plugins attached etc.

Properties

Property	Description
appCodeName	The Code name of the Browser
appName	The name of the Browser
app Version	Specifies the Version of the Browser
mimeTypes	Array of MIME types registered with the browser
Platforms	The operation system on which the browser runs
userAgent	The HTTP user-agent header sent from the browser to the server

In the below demo, the special loop is used to assign each property of the navigator object to variable navigator Property.

```
<html>
<head>
<title>JavaScript Navigator Object</title>
<style>body{font-family:Helvetica;}
    b{color:#44c767;}
</style>
```

```
<script>
   for(var navigator Property in navigator){
     myString = "navigator"+"."+ navigator Property;
     document.write("<b>" + navigator Property + "</b> : <i>" + myString + "</i></br>");
   }
</script>
</head>
<body>
<p></p>
</body>
</html>
```

Plug-In Detection

The Navigator object is used mostly to detect the presence of plug-ins installed on the browser, it uses the plugins array for that purpose. The items in the array comprise of the following property.

name : The plug-in name.

description : The plug-in name.

file name : The file name of the plug-in.

length : The number of MIME types handled by the plugin.

Plug-Ins are programs which can be used by the browser to add abilities to play audio files, videos, animation, view PDF files etc. Eg: Adobe Acrobat Reader, Shockwave/Flash Player etc.

```
<html>
<head>
<title>JavaScript Navigator Object : Detecting Plugins</title>
<style>body{font-family:Helvetica;}
    b{color:#44c767;}
</style>
<script type="text/javascript">
   var plugins_no = navigator.plugins.length;
   for (var i=0; i < plugins_no; i++) {
      var number=i+1;
   document.write("<b>Plug-in No"+number+"</b>: "+navigator.plugins[i].name+" <i>[Location:
"+ navigator.plugins[i].filename + "]</i><p>");
     }
   alert("\n Your Browser has a total of " + number + " plugins installed on it");
</script>
</head>
<body>
```

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

Get all Browser Information.

The JavaScript navigator object has a set of properties which can be used to get all browser Information, these properties can help you customise the Web Page.

```
<!DOCTYPE html>
<html>
<head>
<title>JavaScript Navigator Object : Getting all Browser Information</title>
<style>body{font-family:Helvetica;}
   b{color:#44c767;}
</style>
<script type=”text/javascript”>
 var name Browser= navigator.app Name;
 var your Operation System=navigator. platform;
 var version Browser = navigator.app Version;
 var your Browser= navigator.user Agent;
 document.write(“<b>Browser Name</b> : “ + name Browser + “<br>”);
 document.write(“<b>Platform(i.e operating System)</b> : “ + your Operation System + “<br>”);
 document.write(“<b>Browser Version</b> : “ + version Browser + “<br>”);
 document.write(“<b>User Agent</b>: “ + your Browser + “<br>”);
</script>
</head>
<body>
<p></p>
</body>
```

2.4 EVENT

Q4. What is an event? List out the events supported by javascript?

(OR)

Define Event. Explain the various attributes supported by the JavaScript.

Ans : **(Imp.)**

JavaScript’s interaction with HTML is handled through events that occur when the user or the browser manipulates a page.

When the page loads, it is called an event. When the user clicks a button, that click too is an event. Other examples include events like pressing any key, closing a window, resizing a window, etc.

Developers can use these events to execute JavaScript coded responses, which cause buttons to close windows, messages to be displayed to users, data to be validated, and virtually any other type of response imaginable.

Events are a part of the Document Object Model (DOM) Level 3 and every HTML element contains a set of events which can trigger JavaScript Code.

Events are actions or occurrences that happen in the system you are programming, which the system tells you about so you can respond to them in some way if desired. For example, if the user clicks a button on a webpage, you might want to respond to that action by displaying an information box.

Following are basic events supported by java script along with html

Attribute	Value	Description
offline	script	Triggers when the document goes offline
onabort	script	Triggers on an abort event
onafterprint	script	Triggers after the document is printed
onbeforeonload	script	Triggers before the document loads
onbeforeprint	script	Triggers before the document is printed
onblur	script	Triggers when the window loses focus
oncanplay	script	Triggers when media can start play, but might has to stop for buffering
oncanplaythrough	script	Triggers when media can be played to the end, without stopping for buffering
onchange	script	Triggers when an element changes
onclick	script	Triggers on a mouse click
oncontextmenu	script	Triggers when a context menu is triggered
ondblclick	script	Triggers on a mouse double-click
ondrag	script	Triggers when an element is dragged
ondragend	script	Triggers at the end of a drag operation
ondragenter	script	Triggers when an element has been dragged to a valid drop target
ondragleave	script	Triggers when an element is being dragged over a valid drop target
ondragover	script	Triggers at the start of a drag operation
ondragstart	script	Triggers at the start of a drag operation
ondrop	script	Triggers when dragged element is being dropped
ondurationchange	script	Triggers when the length of the media is changed
onemptied	script	Triggers when a media resource element suddenly becomes empty.
onended	script	Triggers when media has reach the end

onerror	script	Triggers when an error occur
onfocus	script	Triggers when the window gets focus
onformchange	script	Triggers when a form changes
onforminput	script	Triggers when a form gets user input
onhaschange	script	Triggers when the document has change
oninput	script	Triggers when an element gets user input
oninvalid	script	Triggers when an element is invalid
onkeydown	script	Triggers when a key is pressed
onkeypress	script	Triggers when a key is pressed and released
onkeyup	script	Triggers when a key is released
onload	script	Triggers when the document loads
onloadeddata	script	Triggers when media data is loaded
onloadedmetadata	script	Triggers when the duration and other media data of a media element is loaded
onloadstart	script	Triggers when the browser starts to load the media data
onmessage	script	Triggers when the message is triggered
onmousedown	script	Triggers when a mouse button is pressed
onmousemove	script	Triggers when the mouse pointer moves
onmouseout	script	Triggers when the mouse pointer moves out of an element
onmouseover	script	Triggers when the mouse pointer moves over an element
onmouseup	script	Triggers when a mouse button is released
onmousewheel	script	Triggers when the mouse wheel is being rotated
onoffline	script	Triggers when the document goes offline
onoine	script	Triggers when the document comes online
ononline	script	Triggers when the document comes online
onpagehide	script	Triggers when the window is hidden
onpageshow	script	Triggers when the window becomes visible
onpause	script	Triggers when media data is paused
onplay	script	Triggers when media data is going to start playing
onplaying	script	Triggers when media data has start playing
onpopstate	script	Triggers when the window's history changes
onprogress	script	Triggers when the browser is fetching the media data
onratechange	script	Triggers when the media data's playing rate has changed
onreadystatechange	script	Triggers when the ready-state changes

onredo	script	Triggers when the document performs a redo
onresize	script	Triggers when the window is resized
onscroll	script	Triggers when an element's scrollbar is being scrolled
onseeked	script	Triggers when a media element's seeking attribute is no longer true, and the seeking has ended
onseeking	script	Triggers when a media element's seeking attribute is true, and the seeking has begun
onselect	script	Triggers when an element is selected
onstalled	script	Triggers when there is an error in fetching media data
onstorage	script	Triggers when a document loads
onsubmit	script	Triggers when a form is submitted
onsuspend	script	Triggers when the browser has been fetching media data, but stopped before the entire media file was fetched
ontimeupdate	script	Triggers when media changes its playing position
onundo	script	Triggers when a document performs an undo
onunload	script	Triggers when the user leaves the document
onvolumechange	script	Triggers when media changes the volume, also when volume is set to "mute"
onwaiting	script	Triggers when media has stopped playing, but is expected to resume

Q5. Explain briefly about Event handling mechanism?

Ans : **(Imp.)**

Event Handling

Any program that uses GUI (graphical user interface) such as Java application written for windows, is event driven. Event describes the change in state of any object.

For Example : Pressing a button, Entering a character in Textbox, Clicking or Dragging a mouse, etc.

Event handling is the receipt of an event at some event handler from an event producer and subsequent processes. The processes involved in event handling include: Identifying where an event should be forwarded. Making the forward

Components of Event Handling

Event handling has three main components,

- **Events :** An event is a change in state of an object.
- **Events Source :** Event source is an object that generates an event.
- **Listeners :** A listener is an object that listens to the event. A listener gets notified when an event occurs.

2.4.1 ONCLICK

Q6. Explain in detail about On Click event?

Ans :

The onclick event occurs when the user clicks on an element.

Syntax

In HTML :

```
<element onclick="myScript">
```

In JavaScript :

```
object.onclick = function(){myScript};
```

program:

```
<!DOCTYPE html>
<html>
<body>
<pid="demo"
 onclick="myFunction()">
Click me to change my text color.</p>
<script>
function myFunction() {
document.getElementById("demo").
style.color = "red";
}
</script>
</body>
</html>
```

Output :

```
Click me to change my text color.
```

2.4.2 ONLOAD

Q7. Explain briefly about Onload Event?

Ans :

The onload event occurs when an object has been loaded.

Onload is most often used within the <body> element to execute a script once a web page has completely loaded all content (including images, script files, CSS files, etc.).

The onload event can be used to check the visitor's browser type and browser version, and load the proper version of the web page based on the information.

The onload event can also be used to deal with cookies

Syntax

In HTML:

```
<element onload="myScript">
```

In JavaScript:

```
object.onload = function(){myScript};
```

Syntax

In HTML:

```
<element onload="myScript">
```

In JavaScript:

```
object.onload = function(){myScript};
```

Program :

```
<!DOCTYPE html>
    <html>
<body onload="myFunction()">
<h1>Hello World!</h1>
    <script>
function myFunction() {
alert("Page is loaded");
}
</script>
</body>
</html>
```

Output:

```
Page is loaded
Hello World
```

2.4.3 ONMOUSEMOVE

Q8. Explain briefly about Onmouse Move event?

Ans :

The onmousemove event occurs when the pointer is moving while it is over an element.

Occurs when the user moves the mouse over the element.

Use the onmouseover event to receive a notification when the user moves the mouse pointer into and the onmouseout event to receive a notification when the user moves the mouse pointer out of an element.

Syntax:

In HTML:

```
<element onmousemove=”myScript”>
```

In JavaScript:

```
object.onmousemove
            = function(){myScript};
```

Program:

```
<!DOCTYPE html>
      <html>
      <head>
      <style>
      div {
width: 200px;
height: 100px;
border: 1px solid black;
}
</style>
</head>
<body>
<div on mouse move =”
my Function(event)” on mouse out =” clear Coor()”></div>
<p>Mouse over the rectangle above, and get the coordinates of your mouse pointer.</p>
<p id=”demo”></p>
<script>
function myFunction(e) {
var x = e.clientX;
var y = e.clientY;
var coor = “Coordinates:
      (“ + x + “,” + y + “)”;
document. get Element By Id(“demo”).inner HTML = coor;
}
function clear Coor() {
document.get Element By Id(“demo”).
inner HTML = “”;
}
      </script>
      </body>
      </html>
```

Output:

```
Coordinates: (120,17)
```

2.4.4 ONMOUSEOVER

Q9. Explain about On mouse Over event?

Ans :

The onmouseover event occurs when the mouse pointer is moved onto an element, or onto one of its children.

This event is often used together with the onmouseout event, which occurs when a user moves the mouse pointer out of an element.

Syntax

In HTML:

```
<element onmouseover=”myScript”>
```

In JavaScript:

```
object.onmouseover = function(){myScript};
```

Program:

```
<!DOCTYPE html>
<html>
<body>
<img onmouseover=”bigImg(this)” on mouseout=”normalImg(this)” border=”0" src=”smiley.gif” alt=”Smiley” width=”32" height=”32">
<p>The function bigImg() is triggered when the user moves the mouse pointer over the image.</p>
<p>The function normalImg() is triggered when the mouse pointer is moved out of the image.</p>
<script>
      function bigImg(x) {
        x.style.height = "64px";
        x.style.width = "64px";
    }
        function normalImg(x) {
        x.style.height = "32px";
        x.style.width = "32px";
    }
        </script>
        </body>
        </html>
```

Output :

The function bigImg() is triggered when the user moves the mouse pointer over the image.

The function normalImg() is triggered when the mouse pointer is moved out of the image.

2.4.5 ONMOUSEOUT

Q10. Explain briefly about On mouse Out event?

Ans :

The onmouseout event occurs when the mouse pointer is moved out of an element, or out of one of its children.

This event is often used together with the onmouseover event, which occurs when the pointer is moved onto an element, or onto one of its children.

Occurs when the user moves the mouse pointer out of the element.

Use the onmouseover event to receive a notification when the user moves the mouse pointer into and the onmousemove event to receive a notification when the user moves the mouse pointer over an element.

Note: the onmouseout event is not fired during a drag operation. For that case, use the ondragleave event.

Program:

```
        <head>
        <scripttype="text/javascript">
        function OnMouseIn (elem) {
        elem.style.border = "2px solid blue";
    }
        function OnMouseOut (elem) {
        elem.style.border = "";
    }
        </script>
        </head>
        <body>
        <divstyle="background-color:
        #d0f0a0; width:200px"
        onmouseover="OnMouseIn(this)"
        onmouseout="OnMouseOut (this)">
    Move your mouse pointer into and out of
this element!
        </div>
        </body>
```

2.4.6 ONFOCUS

Q11. Explain about briefly Onfocus event?

Ans :

The onfocus event occurs when an element gets focus.

The onfocus event is most often used with <input>, <select>, and <a>. The onfocus event is the opposite of the onblur event. The onfocus event is similar to the on focus in event. The main difference is that the onfocus event does not bubble. Therefore, if you want to find out whether an element or its child gets the focus, you could use the on focus in event. However, you can achieve this by using the optional *use Capture* parameter of the add Event Listener() method for the onfocus event.

Syntax

In HTML:

```
<element onfocus="myScript">
```

In JavaScript:

```
object.onfocus = function(){myScript};
```

Program :

```
<!DOCTYPE html>
    <html>
    <body>
Enter your name: <input type="text"
onfocus="myFunction(this)">
<p>When the input field gets focus, a
function is triggered which changes the
background-color.</p>
    <script>
    function myFunction(x) {
    x.style.background = "yellow";
}
    </script>
    </body>
    </html>
```

Output:

Enter your name: [ ]

When the input field gets focus, a function is triggered which changes the background-color.

2.4.7 ONBLUR

Q12. Explain the concept of Onblur Event.

Ans :

The onblur event occurs when an object loses focus.

The onblur event is most often used with form validation code (e.g. when the user leaves a form field).

The onblur event is the opposite of the on focus event.

The onblur event is similar to the on focus out event. The main difference is that the onblur event does not bubble. Therefore, if you want to find out whether an element or its child loses focus, you could use the on focus out event. However, you can achieve this by using the optional use Capture parameter of the add Event Listener() method for the onblur event.

Syntax

In HTML:

```
<!DOCTYPE html>
<html>
<body>
<p>When you enter the input field, a
function is triggered which sets the
background color to yellow. When you leave
the input field, a function is triggered which
sets the background color to red.</p>
Enter your name: <input type="text"
id="myInput" onfocus="focusFunction()"
onblur="blurFunction()">
<script>
function focus Function() {
// Focus = Changes the background color of
input to yellow
document.get Element By Id ("myInput").
style. background = "yellow";
}
function blurFunction() {
// No focus = Changes the background color
of input to red
document.get Element By Id("myInput").
style.background = "red";
}
</script>
</body>
</html>
```

Output

When you enter the input field, a function is triggered which sets the background color to yellow. When you leave the input field, a function is triggered which sets the background color to red.

Enter your name: [ ]

2.4.8 ON SUBMIT

Q13. Explain about On submit event?

Ans :

The onsubmit event occurs when a form is submitted.

Syntax

In HTML:

```
<element onsubmit="myScript">
```

In JavaScript:

```
object.onsubmit = function(){myScript};
```

Program:

```
<!DOCTYPE html>
<html>
<body>
<p>When you submit the form, a function is triggered which alerts some text.</p>
<formaction="/action_page.php"onsubmit="myFunction()">
Enter name: <input type="text" name=" fname">
<inputtype="submit" value="Submit">
</form>
<script>
function myFunction() {
alert("The form was submitted");
}
</script>
</body>
</html>
```

Output:

When you submit the form, a function is triggered which alerts some text.

Top of Form

Enter name: Mahesh K submit

2.4.9 ON ERRORS

Q14. Explain the concept of On Errors Event

Ans :

The onerror event is triggered if an error occurs while loading an external file (e.g. a document or an image).

When used on audio/video media, related events that occurs when there is some kind of disturbance to the media loading process, are:

- onabort
- onemptied
- onstalled
- onsuspend

Syntax

In HTML:

```
<element onerror="myScript">
<!DOCTYPE html>
<html>
<body>
<p>This example demonstrates how to assign an "onerror" event to an img element.</p>
<img src="image.gif" onerror
="myFunction()">
<p id="demo"></p>
<script>
function myFunction() {
document.get Element By Id("demo").inner HTML = "The image could not be loaded.";
}
</script>
</body>
</html>
```

2.5 Exception Handling

Q15. What is Exception Handling? Explain how to handle the exceptions in Java Script?

Ans : **(Imp.)**

The more JavaScript we code the more errors we'll encounter. Now JavaScript Tutorial will guide us that his is a fact of life in any programming environment. Nobody's perfect and, once we build some complex JavaScript, we'll find there are sometimes scenarios that result in an error that we didn't think of. So we can't always prevent errors from occurring, but we can do something about to handle them.

Exception handling is a very important concept in programming technology. In earlier versions of JavaScript, the exceptions handling was not so efficient and programmers found it difficult to use.

Writing programs that work when everything goes as expected is a good start. Making your programs behave properly when encountering unexpected conditional Problems.

The problematic situations that a program can encounter fall into two categories:

1. Programmer mistakes and genuine problems. If someone forgets to pass a required argument to a function.
2. On the other hand, if a program asks the user to enter a name and it gets back an empty string that is something the programmer can not prevent.

When to Use Exceptions

Exceptions are a way of communicating an exception or error condition from a method to the caller of the method. They are favored over the older technique of returning a status code from a method because they cannot be ignored by the calling code. Without exceptions it is easy to write code that assumes every operation is successful. When an operation doesn't end in success, the error can show up in subsequent operations, and therefore be harder to diagnose.

In most implementations, throwing exceptions is a relatively expensive operation. Therefore, exceptions should be used to communicate truly exceptional conditions rather than conditions that may occur under the normal execution of a program.

The benefit of using exceptions usually increases with the size of the application. Exceptions require the developer to define the success and error conditions of methods and require the calling code to handle both possibilities. Exceptions in General Interface ignores error conditions altogether. Additionally, exceptions help with the principle of failing early. In general it is better to fail (raise an exception) at the first sign of trouble when the cause of the failure is well known. Otherwise errors might cascade to other parts of a program where they will be hardy to diagnose.

Catching errors in JavaScript:

It is very important that the errors thrown must be catched or trapped so that they can be handled more efficiently and conveniently and the users can move better through the web page.

There are mainly two ways of trapping errors in JavaScript.

1. Using try…catch statement
2. Using onerror event

try/catch/finally

The try..catch statement has two blocks in it:

➢ try block

➢ catch block

In the try block, the code contains a block of code that is to be tested for errors. The catch block contains the code that is to be executed if an error occurs.

Syntax

```
try
{
…………
…………
        // Block of code which is to be tested for errors
}
```

```
catch (err)
{
        …………
        …………   //Block of code which is to be executed if an error occurs
}
```

Handling run time errors in Java Script using try/catch/finally

try/catch/finally are so called exception handling statements in JavaScript. An exception is an error that occurs at run time due to an illegal operation during execution. try/catch/finally lets you deal with exceptions gracefully.

Whenever the browser runs into an exception somewhere in a JavaScript code, it displays an error message to the user while aborting the execution of the remaining code. You can put a lid on this behaviour and handle the error the way you see fit using try/catch/finally. At its simplest you'd just use try/catch to try and run some code, and in the event of any exceptions,

Syntax

```
        try
{
        undefinedfunction()
        alert('I guess you do exist')
}
        catch(e)
{
        alert('An error has occurred: '+e.message)
}
        finally
{
        alert('I am alerted regardless of the outcome above')
}
```

The Error object and throwing your own errors

As promised, we're going to take a closer look at the Error object that gets passed into the catch clause to see just what we can extract from it in an event of an error. The Error object in all browsers support the following two properties :

1. **name**

   The name of the error, or more specifically, the name of the constructor function the error belongs to.

2. **message**

   A description of the error, with this description varying depending on the browser.

Syntax

```
        try
{
        document.body.filters[0].apply()
}
        catch(e)
{
        alert(e.name + "\n" + e.message)
}
```

Six possible values can be returned by the name property, which as mentioned correspond to the names of the error's constructors. They are:

Error Name	Description
Eval Error	An error in the eval ( ) function has occurred
Range Error	Out of range number value has occurred
Reference Error	An illegal reference has occurred
Syntax Error	A syntax error within code inside the eval ( ) function has occurred. All other syntax errors are not caught by try/cach/finally, and will trigger the default browser error message associated with the error. To catch actual syntax errors, you may use the on error event.
Type Error	A syntax error within code inside the eval ( ) function has occurred.
URIError	An error when encoding or decoding the URI has occurred (ie: when calling encode URI ())

This level of detail may be useful when you wish to sniff out a specific type of error in your catch clause. In the below, no DIV on the page exists with ID=”mydiv”. When trying to set its .inner HTML property, a Type Error occurs, since we’re trying to assign the .inner HTML property to a null object:
This level of detail may be useful when you wish to sniff out a specific type of error in your catch clause.

Throwing your own errors (exceptions)

Instead of waiting for one of the 6 types of errors above to occur before control is automatically transferred from the try block to the catch block, you can also explicitly throw your own exceptions to force that to happen on demand. This is great for creating your own definitions of what an error is and when control should be transferred to catch.

To throw an error, invoke, well, the throw statement inside your try/catch/finally blocks. The syntax is :

Syntax

Throw my error object

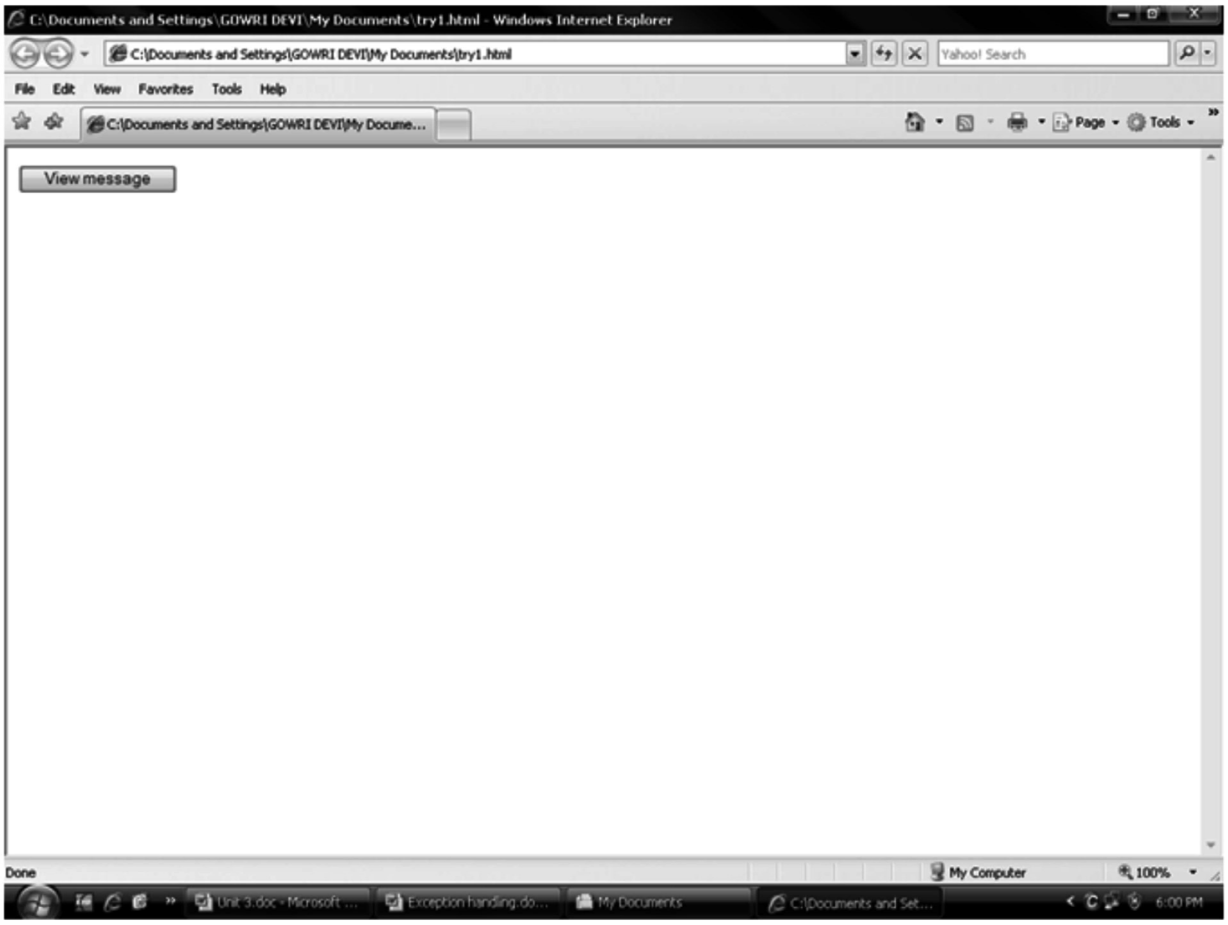
There are a number of other things you can throw, which changes the contents of the error object passed into catch. The following are all valid throws:

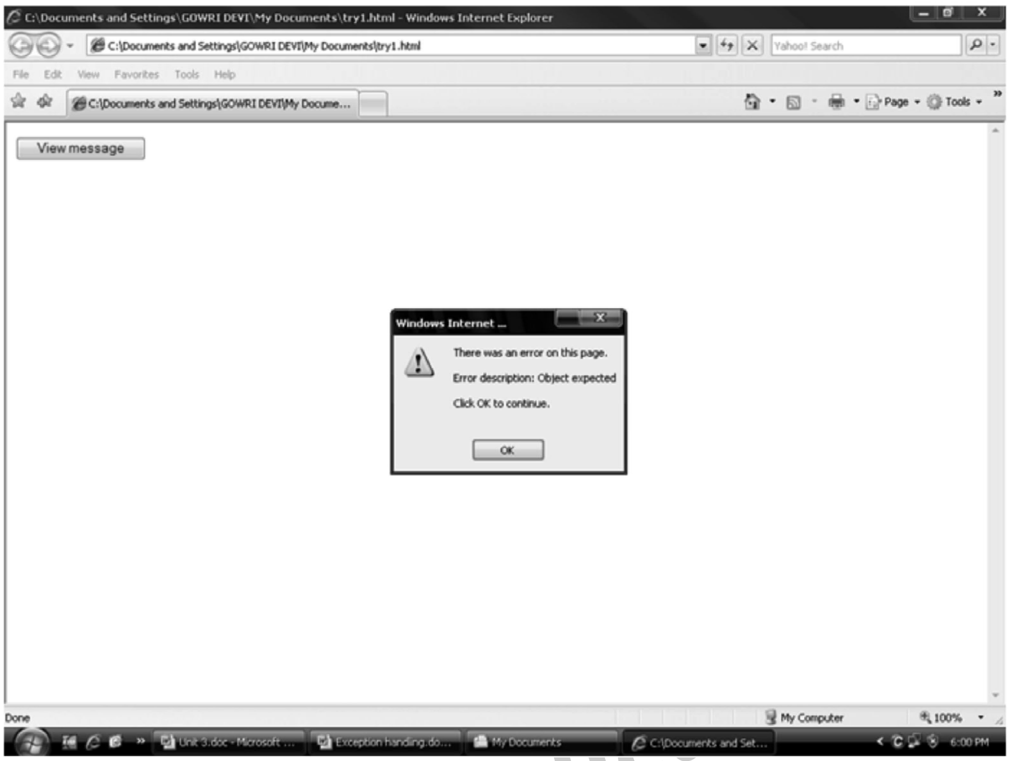
1. throw “An error has occurred”
2. throw true
3. throw new Error(“I detect an error!”)
4. throw new SyntaxError(“Your syntax is no good”)

Example

```
	<html>
	<head>
	<script type=”text/javascript”>
	var txt=””;
	function message()
{
	try
{
	adddlert(“Welcome guest!”);
}
        catch(err)
{
        txt=”There was an error on this page.\n\n”;
        txt+=”Error description: “ + err.description + “\n\n”;
        txt+=”Click OK to continue.\n\n”;
        alert(txt);
}
}
        </script>
        </head>
        <body>
        <input type=”button” value=”View message” onclick=”message()” />
        </body>
        </html>
```

Output


C:\Documents and Settings\GOWRI DEVI\My Documents\try1.html - Windows Internet Explorer
File Edit View Favorites Tools Help
View message
Done
My Computer
100%


C:\Documents and Settings\GOWRI DEVI\My Documents\try1.html - Windows Internet Explorer
C:\Documents and Settings\GOWRI DEVI\My Documents\try1.html
Yahoo! Search
File Edit View Favorites Tools Help
Page Tools
View message
Done
My Computer
100%
6:00 PM

Windows Internet ...

There was an error on this page.

Error description: Object expected

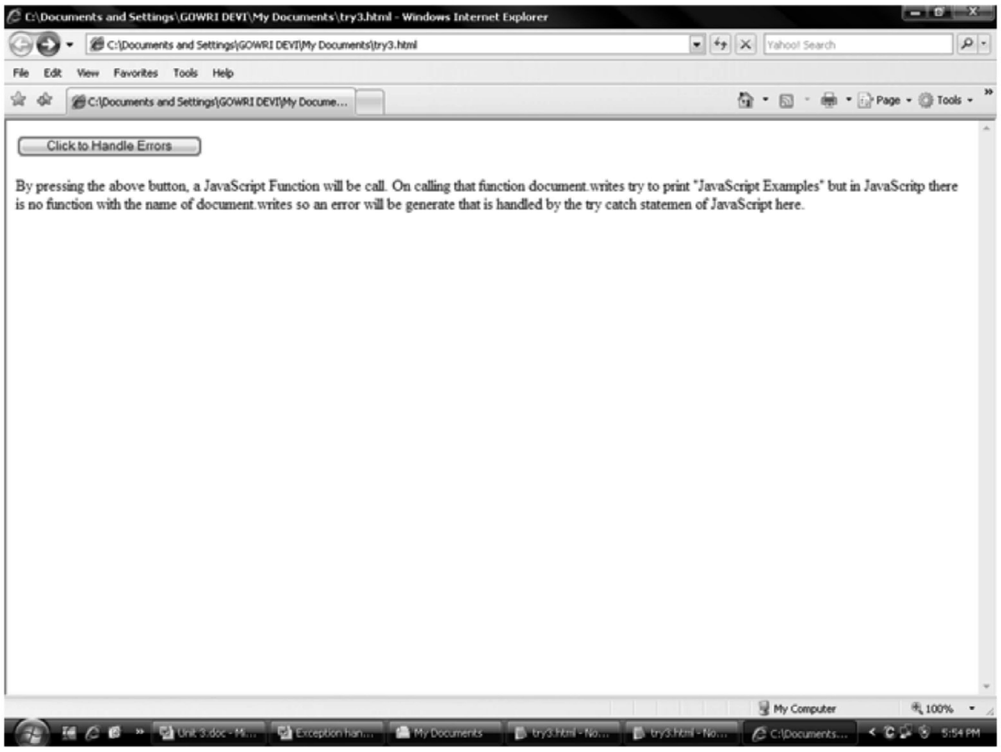
Click OK to continue.

OK

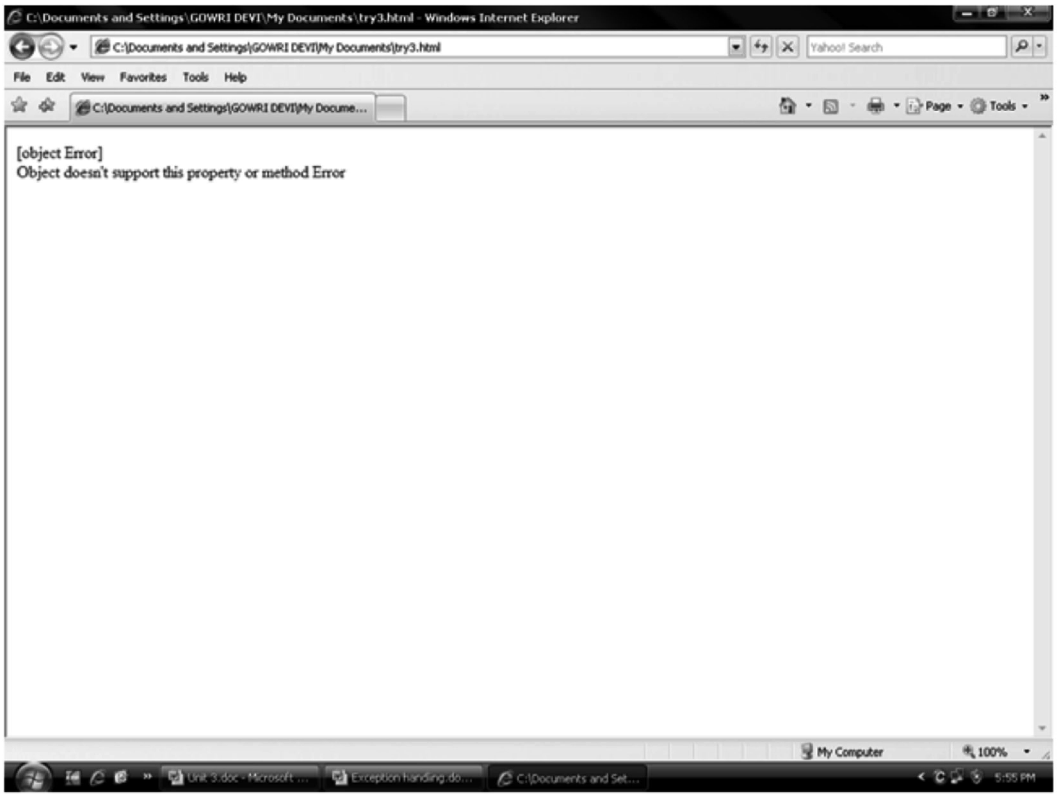
Example

```
        <html>
        <head>
        <script language="javascript">
        function handle Error()
{
        try
{
        document.writes("JavaScript Examples");
}
        catch(exception)
{
        document.write(exception+"<br>");
        document.write(exception.description + " Error");
}
}
        </script>
        </head>
        <body>
        <input type="button" value="Click to Handle Errors" onclick="handleError()">
        <p>By pressing the above button, a JavaScript Function will be call. On calling that function
document.writes try to print "JavaScript Examples" but in JavaScritp there is no function with the name
of document.writes so an error will be generate that is handled by the try catch statements of JavaScript
here.</p>
        </body>
        </html>
```

Output


C:\Documents and Settings\GOWRI DEVI\My Documents\try3.html - Windows Internet Explorer
Click to Handle Errors

By pressing the above button, a JavaScript Function will be call. On calling that function document.writes try to print "JavaScript Examples" but in JavaScritp there is no function with the name of document.writes so an error will be generate that is handled by the try catch statemen of JavaScript here.

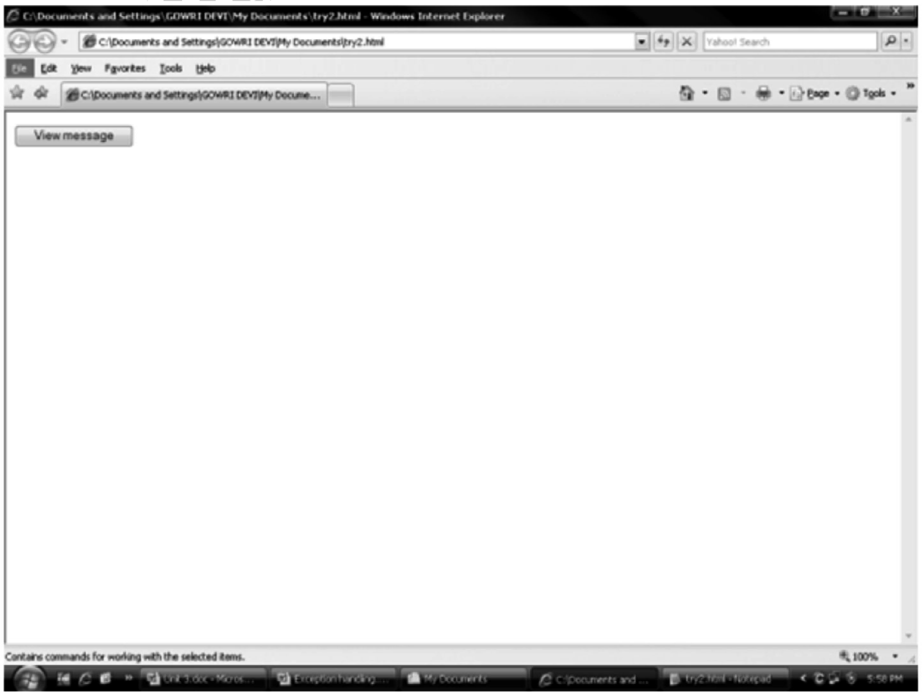

C:\Documents and Settings\GOWRI DEVI\My Documents\try3.html - Windows Internet Explorer

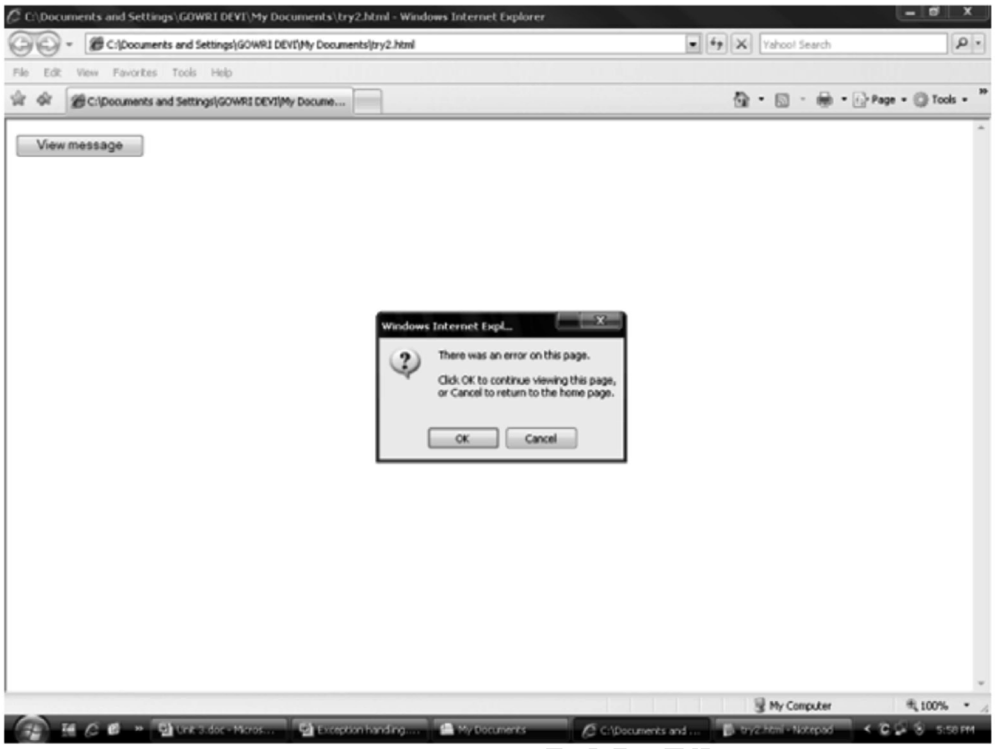
```
[object Error]
Object doesn't support this property or method Error
```

Example

```
        <html>
        <head>
        <script type="text/javascript">
        var txt="";
        function message()
{
        try
{
        adddlert("Welcome guest!");
}
        catch(err)
{
        txt="There was an error on this page.\n\n";
        txt+="Click OK to continue viewing this page,\n";
        txt+="or Cancel to return to the home page.\n\n";
        if(!confirm(txt))
 {
      document.location.href="http://www.w3schools.com/";
 }
 }
}
        </script>
        </head>
        <body>
        <input type="button" value="View message" onclick="message()" />
        </body>
        </html>
```

Output


C:\Documents and Settings\GOWRI DEVI\My Documents\try2.html - Windows Internet Explorer
C:\Documents and Settings\GOWRI DEVI\My Documents\try2.html
Yahoo! Search
File Edit View Favorites Tools Help
C:\Documents and Settings\GOWRI DEVI\My Docume...
Page Tools
View message
Contains commands for working with the selected items.
100%


C:\Documents and Settings\GOWRI DEVI\My Documents\try2.html - Windows Internet Explorer
View message
Windows Internet Expl...
OK
Cancel
My Computer
100%

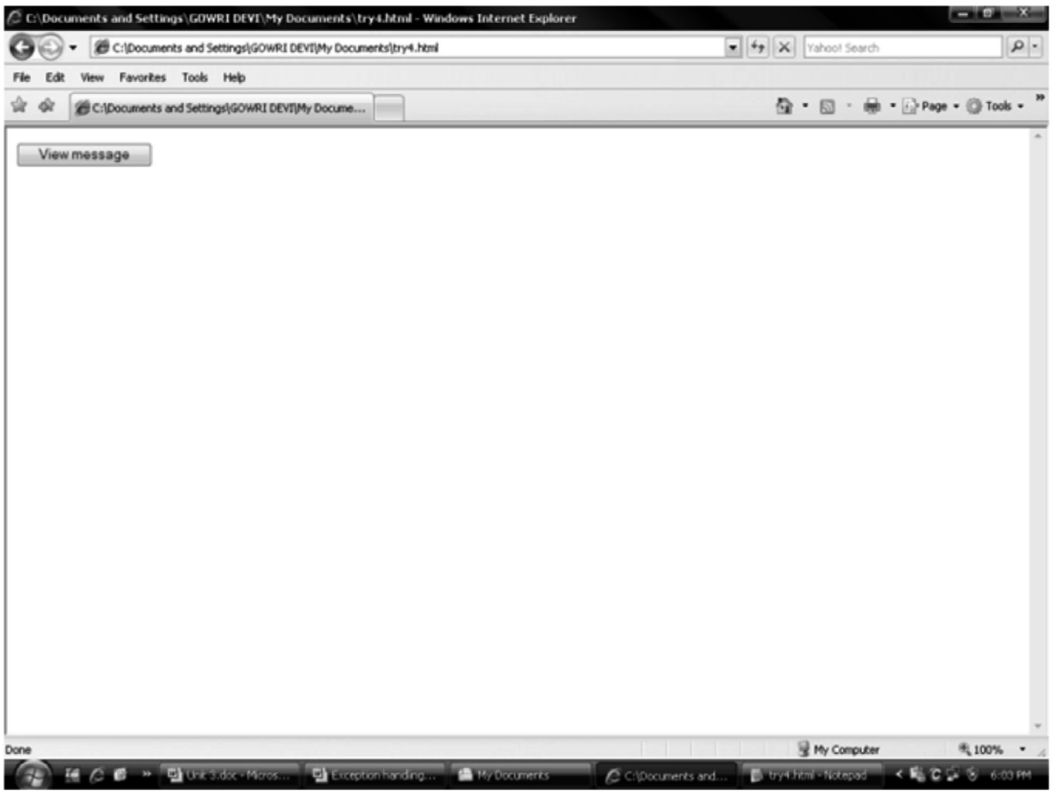
There was an error on this page.

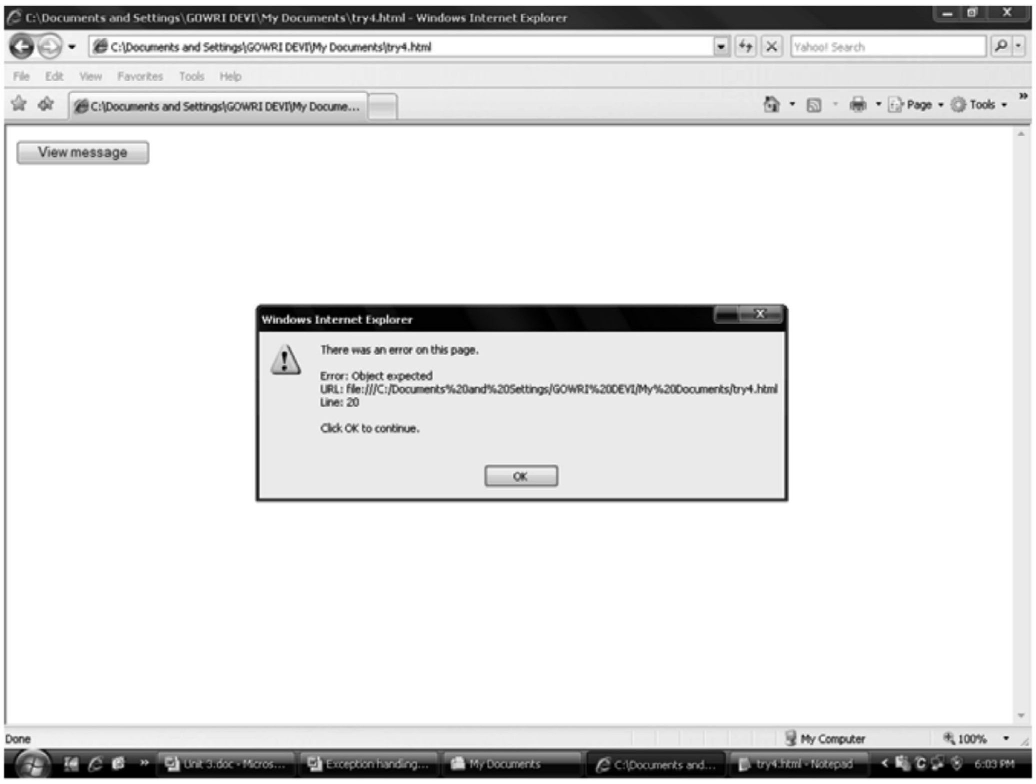
Click OK to continue viewing this page, or Cancel to return to the home page.

Example

```
        <html>
        <head>
        <script type="text/javascript">
        onerror=handleErr;
        var txt="";
        function handleErr(msg,url,l)
{
        txt="There was an error on this page.\n\n";
        txt+="Error: " + msg + "\n";
        txt+="URL: " + url + "\n";
        txt+="Line: " + l + "\n\n";
        txt+="Click OK to continue.\n\n";
        alert(txt);
        return true;
}
        function message()
{
        adddlert("Welcome guest!");
}
        </script>
        </head>
        <body>
        <input type="button" value="View message" onclick="message()" />
        </body>
        </html>
```

Output


C:\Documents and Settings\GOWRI DEVI\My Documents\try4.html - Windows Internet Explorer
View message
Done
My Computer
100%


C:\Documents and Settings\GOWRI DEVI\My Documents\try4.html - Windows Internet Explorer
View message
Windows Internet Explorer
OK

```
There was an error on this page.

Error: Object expected
URL: file:///C:/Documents%20and%20Settings/GOWRI%20DEVI/My%20Documents/try4.html
Line: 20

Click OK to continue.
```

2.6 Filters and Transactions

Q16. Explain filters and transactions for multimedia effects.

Ans : **(Imp.)**

CSS filters to add special effects to text, images and other aspects of a webpage without using images or other graphics. Filters only work on Internet Explorer 4.0+,. If you are developing your site for multi browsers, then it may not be a good idea to use CSS filters because there is a possibility that it would not give any advantage.

(i) Filters

Following are the filters/effects generally in use.

1. Alpha Channel

The Alpha Channel filter alters the opacity of the object, which makes it blend into the background. The following parameters can be used in this filter.

Parameter	Description
opacity	Level of the opacity. 0 is fully transparent, 100 is fully opaque.
finishopacity	Level of the opacity at the other end of the object.
style	The shape of the opacity gradient. 0 = uniform 1 = linear 2 = radial 3 = rectangular
startX	X coordinate for opacity gradient to begin.
startY	Y coordinate for opacity gradient to begin.
finishX	X coordinate for opacity gradient to end.
finishY	Y coordinate for opacity gradient to end.

Example

```
<html>
  <head>
  </head>
    <body>
     <p>Image Example:</p>
    <img src="/css/images/logo.png" alt="CSS Logo"
       style="Filter: Alpha(Opacity=100,
       FinishOpacity=0,
       Style=2,
       StartX=20,
       StartY=40,
       FinishX=0,
    FinishY=0)" />
   <p>Text Example:</p>
   <div style="width: 357;
    height: 50;
    font-size: 30pt;
    font-family: Arial Black;
    color: blue;
   Filter: Alpha(Opacity=100, FinishOpacity=0, Style=1, StartX=0, StartY=0, FinishX=580,
FinishY=0)">CSS Tutorials</div>
  </body>
</html>
```

2. Motion Blur

Motion Blur is used to create blurred pictures or text with the direction and strength. The following parameters can be used in this filter.

Parameter	Description
add	True or false. If true, the image is added to the blurred image; and if false, the image is not added to the blurred image.
direction	The direction of the blur, going clockwise, rounded to 45-degree increments. The default value is 270 (left). 0 = Top 45 = Top right 90 = Right 135 = Bottom right 180 = Bottom 225 = Bottom left 270 = Left 315 = Top left
strength	The number of pixels the blur will extend. The default is 5 pixels.

Example

```
<html>
  <head>
  </head>
  <body>
    <p>Image Example:</p>
```

```
        <img src="/css/images/logo.png" alt="CSS Logo"
          style="Filter: Blur(Add = 0, Direction = 225, Strength = 10)">
        <p>Text Example:</p>
        <div style="width: 357;
          height: 50;
          font-size: 30pt;
          font-family: Arial Black;
          color: blue;
          Filter: Blur(Add = 1, Direction = 225, Strength = 10)">CSS Tutorials</div>
      </body>
    </html>
```

3. Chroma Filter

Chroma Filter is used to make any particular color transparent and usually it is used with images. You can use it with scrollbars also. The following parameter can be used in this filter -

Parameter	Description
color	The color that you'd like to be transparent.

4. Drop Shadow Effect

Drop Shadow is used to create a shadow of your object at the specified X (horizontal) and Y (vertical) offset and color.

The following parameters can be used in this filter -

Parameter	Description
color	The color, in #RRGGBB format, of the dropshadow.
offX	Number of pixels the drop shadow is offset from the visual object, along the x-axis. Positive integers move the drop shadow to the right, negative integers move the drop shadow to the left.
offY	Number of pixels the drop shadow is offset from the visual object, along the y-axis. Positive integers move the drop shadow down, negative integers move the drop shadow up.
positive	If true, all opaque pixels of the object have a dropshadow. If false, all transparent pixels have a dropshadow. The default is true.

5. Flip Effect

Flip effect is used to create a mirror image of the object. The following parameters can be used in this filter -

Parameter	Description
FlipH	Creates a horizontal mirror image
FlipV	Creates a vertical mirror image

6. Glow Effect

Glow effect is used to create a glow around the object. If it is a transparent image, then glow is created around the opaque pixels of it. The following parameters can be used in this filter “

Parameter	Description
Color	The color you want the glow to be
Strength	The intensity of the glow (from 1 to 255)

7. Grayscale Effect

Grayscale effect is used to convert the colors of the object to 256 shades of gray. The following parameter is used in this filter -

Parameter	Description
gray	Converts the colors of the object to 256 shades of gray.

8. Invert Effect

Invert effect is used to map the colors of the object to their opposite values in the color spectrum, i.e., to create a negative image. The following parameter is used in this filter “

Parameter	Description
Invert	Maps the colors of the object to their opposite value in the color spectrum.

9. Mask Effect

Mask effect is used to turn transparent pixels to a specified color and makes opaque pixels transparent. The following parameter is used in this filter -

Parameter	Description
Color	The color that the transparent areas will become

10. Shadow Filter

Shadow filter is used to create an attenuated shadow in the direction and color specified. This is a filter that lies in between Drop shadow and Glow. The following parameters can be used in this filter

Parameter	Description
color	The color that you want the shadow to be.
direction	The direction of the blur, going clockwise, rounded to 45-degree increments. The default value is 270 (left). 0 = Top 45 = Top right 90 = Right 135 = Bottom right 180 = Bottom 225 = Bottom left 270 = Left 315 = Top left

11. Wave Effect

Wave effect is used to give the object a sine wave distortion to make it look wavy. The following parameters can be used in this filter -

Parameter	Description
Add	A value of 1 adds the original image to the waved image, 0 does not.
Freq	The number of waves.
Light	The strength of the light on the wave (from 0 to 100).
Phase	At what degree the sine wave should start (from 0 to 100).
strength	The intensity of the wave effect.

12. X-Ray Effect

X-Ray effect grayscales and flattens the color depth. The following parameter is used in this filter:

Parameter	Description
xray	Grayscales and flattens the color depth.

(ii) Transition

Transitions are filters that provide effects to the context of web page. They are responsible for visually changing the control/moving a page from one state to another. They provide time-based effects thereby allowing the user to create animation in images.

1. Reveal transition filter
2. Blend transition filter

1. Reveal transition filter

The reveal transition filter written as Reveal? Trans is applied to multiple visual objectives together inorder to show/hide them.

Syntax

STYLE = "filter:revealtrans(duration = duration, transition = transitionshape)

Here, duration is the value/time taken duration by the transition. It is exposed in "seconds.milliseconds".

2. Blend transition filter

The blend transition filter written as BlendTrans is applied to a visual object to fadein/fadeout for certain time limit.

Syntax

STYLE = "filter: blendtrans(duration = duration";

Here, duration is the value/time duration taken by the transition. It is also expressed in "seconds.milliseconds".

2.7 Data binding with Tabular data control binding to IMO

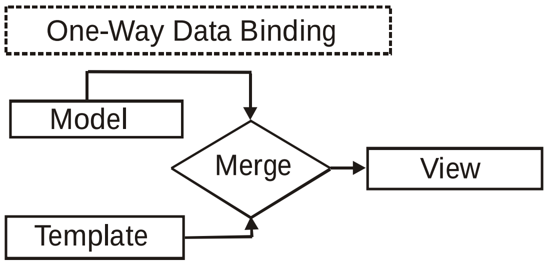
Q17. What is DataBinding ? Explain.

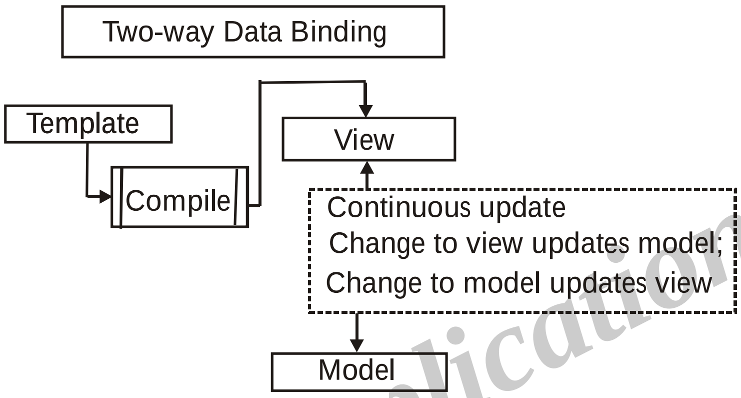
Ans : **(Imp.)**

The user interface or UI is the piece of our web applications that we present to the end user. Sometimes we want to update the UI to reflect changes in input or vice versa. For this task, we can use data binding. This means that we can connect changes to an object to the UI.

For example, if you have an Employee Name input field, then you should be able to have the underlying data change as well. Most data binding is done between an external application and an underlying database system. However, we now have tools that let us complete two-way binding right in the web browser. Two-way binding means that updates to the UI and model are kept in sync.

The graphic below shows how one-way and two-way binding work.





The word model in the graphic simply means the data model. In our example, we will be using an Employee Name and Rate as fields in our model. When they are updated in the UI, they get updated in the model (and vice versa). There are tools out there such as AngularJS, Knockout, Backbones, Derby, or Meteor. These tools provide built-in data binding and their own syntax.

Data Binding in JavaScript

You can also accomplish binding in native JavaScript! First, let's build a simple HTML page with some inputs. When the user enters data in the fields, we want to dynamically update the UI. This is why you will see that each field is repeated. Don't worry, it will make sense when we get to the actual JavaScript.

```
<html>
<head>
<title>Data Binding in JavaScript</title>
<head>
<body>
<p>Employee Name: <input class="emp" type="text"></p>
<p>Output: <input class="emp" type="text"></p>
<p>Rate: <input class="rate" type="text"></p>
<p>Output: <input class="rate" type="text"></p>
<script src="bindMe.js"></script>
</body>
</html>
```

Now for the fun part. The following JavaScript may look intimidating, but when examined carefully, we can see how the data is kept in sync with the UI. Recall that we have two fields, each of the same class and name for both Employee Name and Rate. The JavaScript code looks at these and ensures that both are kept up to date.

2.8 Structured graphics, Active controls

Q18. Explain Structured Graphics of Active Controls.

Ans :

The Structured Graphics control is an effective and versatile tool for adding lightweight graphics to a Web document. Structured graphics are vector-based (meaning that they are constructed out of a series of lines and curves, rather than pixel by pixel like a JPG or GIF image) and tend to have very small file sizes compared to conventional graphics.

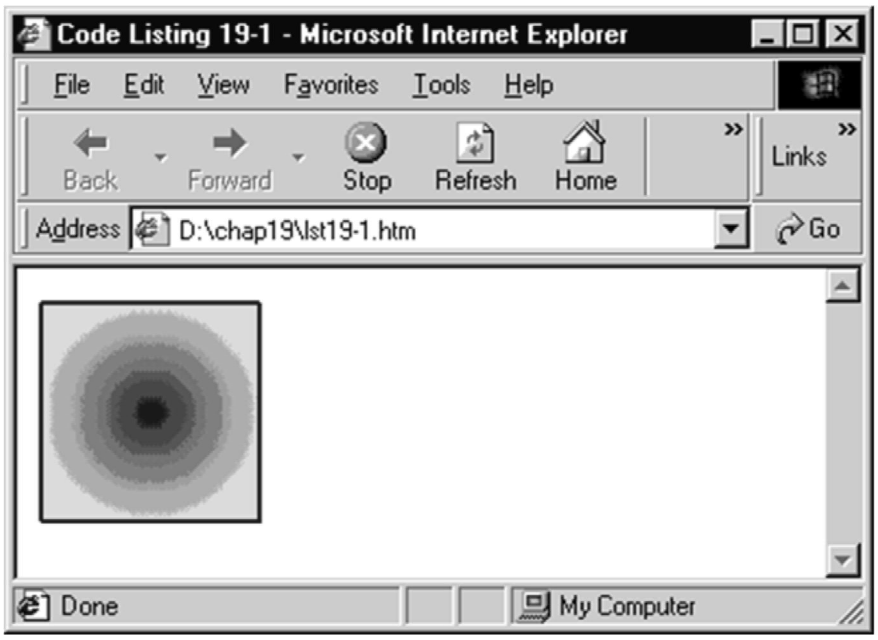
The Direct Animation controls, like other ActiveX controls, must be identified on the page. First that everything is enclosed in a set of <OBJECT></OBJECT> tags, an HTML mechanism for identifying a non-HTML item. The ID attribute gives the object a name so that it can be referenced easily by script. Although an ID is not necessary, you will probably find that an object is of limited use without one since it becomes more difficult to address through script.

Every ActiveX control has a unique CLASSID a required identifier that tells the browser which exact type of object is being used. Fortunately, you don't need to memorize those long hexadecimal number sequences. The easiest way to code the correct CLASSID is simply to copy it from another page that uses that control. To help you get started, we have included a text file named da_id.txt in the chap19 folder on the companion CD that contains all the CLASSIDs for the DirectAnimation controls, along with their friendly names. Many of the HTML editing programs, such as Microsoft FrontPage, Microsoft Visual Interdev, and others make adding objects (and their CLASSIDs) to a page as simple as selecting a name from a list.

The STYLE attribute is not completely necessary either, but it is very useful. It can be used to set a height and width for the overall structured graphic object (independent of any shapes created inside it). It is important that you size a structured graphic to be large enough to hold any items you create inside it. For example, if the object is only 100 pixels wide but it holds a shape 300 pixels wide, the shape inside will be clipped. Any shapes created inside the structured graphic object are normally drawn starting at its center. Without planning for this, you might find a shape being clipped by the right and bottom edges of the structured graphic object. You can avoid this situation by using the Extent Height and Extent Width parameters, discussed later in this chapter. The STYLE attribute also can be used to position an object on the page or to set its z-index to put the object behind or in front of other elements.

The PARAM tags allow you to set values for many of the properties that affect a control. <PARAM> tags can also be used to call methods of a particular control. The name portion of the PARAM element refers either to a property name, such as High Quality, or to a line number. The value portion indicates either the value assigned to the property or a particular method that should be used. These properties and methods are different for every object. Although many objects might use properties or methods of the same name, the effects often differ for each object.

```
<HTML>
<HEAD>
 <TITLE>Code Listing 19-1
</TITLE>
</HEAD>
<BODY>
<OBJECTID="square" CLASSID="CLSID:369303C2-D7AC-11D0-89D5-00A0C90833E6"
STYLE="height: 102; width: 102">
<PARAM NAME="Line0001" VALUE="SetLineColor(0,0,255)">
<PARAM NAME="Line0002" VALUE="SetLineStyle(1,2)">
<PARAM NAME="Line0003" VALUE="SetFillStyle(10)">
<PARAM NAME="Line0004" VALUE="SetFillColor(0,0,255,255,196,196)">
<PARAM NAME="Line0005" VALUE="Rect(1,1,100,100,0)">
<PARAM NAME="ExtentTop" VALUE="0">
<PARAM NAME="ExtentLeft" VALUE="0">
</OBJECT>
</BODY>
</HTML>
```


Code Listing 19-1 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Links
Address D:\chap19\lst19-1.htm Go
Done My Computer

UNIT III

Introduction to scripting: Java Script, Data types, Arithmetic's Equality relational, assignment increment, decrement operators, Java Script Control Structures- if, if-else, while. Java Script.

Control Structures: For, Switch, Do/while, break. Programming modules, recursion, recursion vs iteration global functions arrays,. using arrays, Reference and reference parameters, passing arrays to functions, multiplesubscripted arrays, objects-math, string. Boolean and number.

3.1 Introduction to Scripting

Q1. What is scripting language?

Ans :

A scripting language is a programming language designed for integrating and communicating with other programming languages. Some of the most widely used scripting languages are JavaScript, VBScript, PHP, Perl, Python, Ruby, ASP and TCL. Since a scripting language is normally used in conjunction with another programming language, they are often found alongside HTML, Java or C++.

One common distinction between a scripting language and a language used for writing entire applications is that, while a programming language is typically compiled first before being allowed to run, scripting languages are interpreted from source code or bytecode one command at a time.

Although scripts are widely employed in the programming world, they have recently become more associated with the World Wide Web, where they have been used extensively to create dynamic Web pages. While technically there are many client-side scripting languages that can be used on the Web, in practice it means using JavaScript.

Scripting Languages :

- AppleScript
- ColdFusion
- DCL
- Embeddable Common Lisp
- ecl
- Erlang
- JCL
- CoffeeScript
- JScript and JavaScript
- Lua
- m4
- Perl
- PHP
- Pure
- Python
- Rebol
- Red
- Rexx
- Ruby
- Scheme
- Tcl
- Unix Shell scripts (ksh, csh, bash, sh and others)
- VBScript
- Work Flow Language
- Windows PowerShell
- XSLT

Q2. Differentatiate between Scripting Language and Programming Language.

Ans :

Scripting languages are programming languages that don't require an explicit compilation step.

For example, in the normal case, you have to compile a C program before you can run it. But in the normal case, you don't have to compile a JavaScript program before you run it. So JavaScript is sometimes called a "scripting" language.

This line is getting more and more blurry since compilation can be so fast with modern hardware and modern compilation techniques. For instance, V8, the JavaScript engine in Google Chrome and used a lot outside of the browser as well, actually compiles the JavaScript code on the fly into machine code, rather than interpreting it. (In fact, V8's an optimizing two-phase compiler.)

Also note that whether a language is a "scripting" language or not can be more about the environment than the language. There's no reason you can't write a C interpreter and use it as a scripting language (and people have). There's also no reason you can't compile JavaScript to machine code and store that in an executable file (and people have). The language Ruby is a good example of this: The original implementation was entirely interpreted (a "scripting" language), but there are now multiple compilers for it.

Some examples of "scripting" languages (e.g., languages that are traditionally used without an explicit compilation step):

- Lua
- JavaScript
- VBScript and VBA
- Perl
- And a small smattering of ones traditionally used with an explicit compilation step:
- C
- C++
- Java (but note that Java is compiled to bytecode, which is then interpreted and/or recompiled at runtime)
- Pascal

...and then you have things like Python that sit in both camps: Python is widely used without a compilation step, but the main implementation (CPython) does that by compiling to bytecode on-the-fly and then running the bytecode in a VM, and it can write that bytecode out to files (.pyc, .pyo) for use without recompiling.

3.2 Java Script

Q3. What is Java Script? What are the Features of Java Script?

Ans :

Java Script is a Scripting. Object based language developed by Netscape Communications. The Original name is Live Script & due to the association of Netscape with Sun Microsystems, it is renamed as Java Script. Java Script & Java, even though both are languages, differ a lot, Java Script is an object based & java is an Object Oriented language.

Java Script is easier to code & increases the functionality of HTML. With java script, we can design dynamic & interactive Web Pages easily. Java Script comes with many Web-pages building functions that can manipulate HTML Forms, buttons, test areas, textboxes and data.

Java Script is speedy language. It is interpreted & not compiled. Browsers include the Java Script interpreter. If errors exist in the program, the program runs & gives output units it doesn't come across a mistake. The remaining part of the program from where mistake occur is not interpreted.

Java Script is written inside an HTML document. i.e., Java Script code & HTML code exists in the same file. But separated from HTML code by enclosing the Java Script with in <Script>.... </Scripts tags. Within the opening tag, the 'language' attribute is set to 'Java Script or Text'.

The Script tag is typically used as follows.

< Script Language "Java Script/text" >

Features

1. Java Script is an object-based language that is confined to run within the Web-browsers only.
2. Java Script is an interpreted language &requires no compilation steps.
3. Java Script can directly be embedded in HTML files. The HTML files with embedded JavaScript commands.
4. Java script is loosely typed language i.e. one data type can be automatically converted into other types without explicit conversion.
5. Java Script is Platform independent but browser dependent.
6. Java Script supports event-based programming
7. Performance is good since the java script programs are included in the same files as the HTML code for a webpage.
8. Java Script is multifunctional.

Java Vs Java Script

Java Script	Java
Interpreted (Not Compiled) by client	Compile byte codes downloaded from server, executed on client
Object based. Uses built-in, extensible object, but no classes or inheritance.	Object Oriented, Applet consists of object classes with inheritance.
Code integrated with and embedded in HTML.	Applets distinct from HTML
Variable data types not declared	Variable data type must be declared
Dynamic binding. Object references checked at runtime.	Static binding, object references must exit at compile time

Q4. State the Advantages and Disadvantages of Java Script.

Ans :

Advantages

1. JavaScript is supported by most of the web browsers.
2. It provides an easy way of accessing document object as well as it can manipulate almost all of these objects.
3. It gives attractive animation with a less download time for the multimedia data types.
4. No, special plug-ins are required to use java Script.
5. Security is the biggest advantage in java script.

Disadvantages

1. If your script doesn't work then your page is useless.
2. Most of the web suffers disable their java script support due to the problems associated with broken scripts.

3. Complex scripts take longer starting and running time.
4. Most of the java script depends on the manipulated on Dom's elements. And different browsers give different type of accessing to object
5. It does not support the standards set of objects.

Structure & Basic Syntax of Java Script

Java Script code is written in HTML files along with HTML code. To separate the code of Java Script from HTML<code, we write Java Script code with in <Script> & </Scripts tags. Java Script is a Scripting language like other languages it defines variables functions, control structures etc. Html code can contain any no of <Script>tags.

Syntax Declaration in Body

```
<Html>
<Head>
<Title>     </Title>
<Body>
<Script type = Javascript/text>
.
.
</Script>
</Body>
</Html>
```

We define functions in head tags & call them in body tag. By the time we call them in body they will be already defined & executed in head tags.

Syntax Declaration in Head

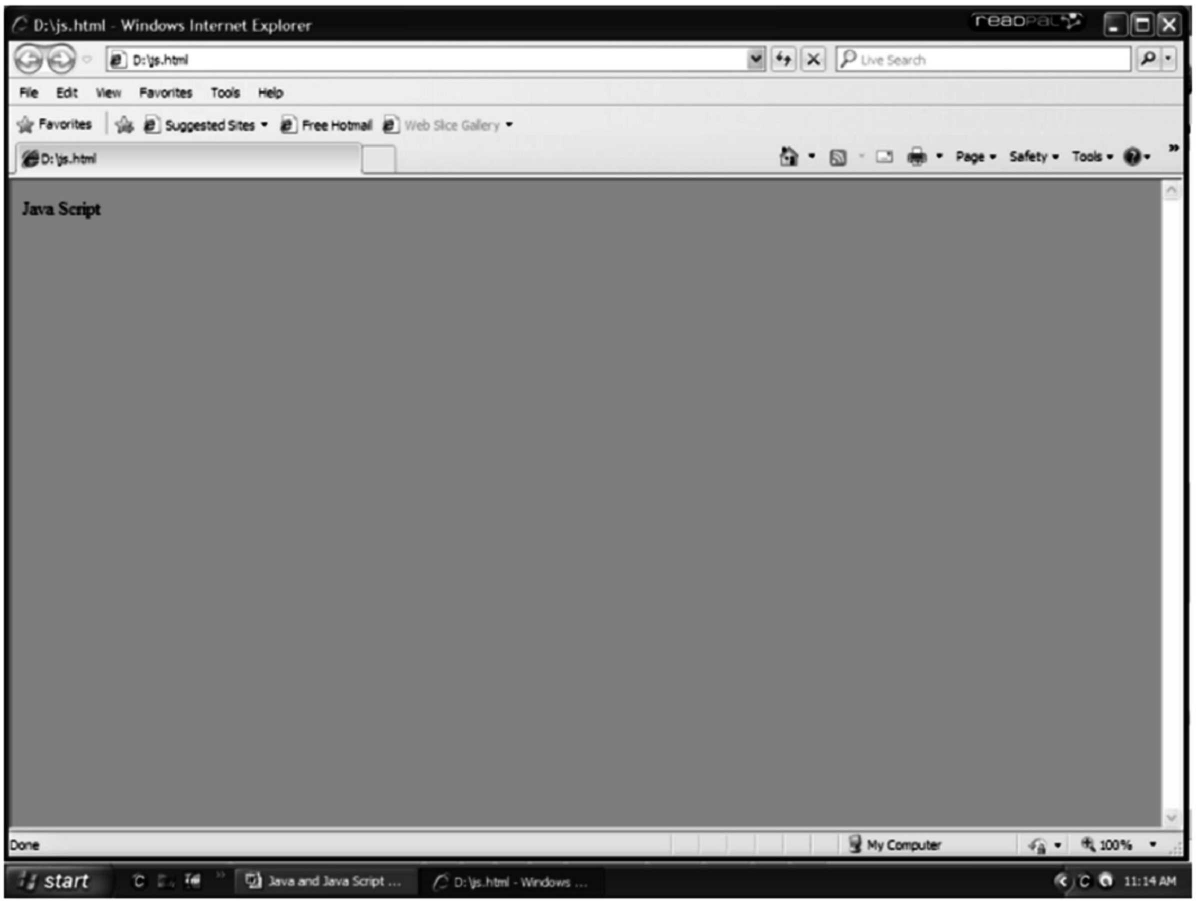
```
<Html>
<Head>
<Title> ………… </Title>
<Script type = Javascript/text>
.
.
</Script>
<Body>
.
.
</Body>
</Html>
```

Program to demonstrate java Script

```
<Html>
<Head<
<Title>Java Script</Title>
</head>
```

```
<Body bgcolor=”cyan”>
<script type=”javascript/text”>
document.writeln(“Hai,JavaScript.....”);
</script>
</Body>
</Html>
```

Output



Java Script

The output statements are written in JavaScript as **document.writeln()** method like printf(). “Document” is an object of java script. If any code of HTML is specified it should be written in double codes.

document.writeln(“ “+”<br>” +” “);

3.3 Data Types

Q5. What are the different data types in Java Script?

Ans : **(Imp.)**

Every language identifies the information in the form of data types. Compared to other languages, java script has a small number of data types. A variable’s purpose is to store information so that it can be used later. A variable is a symbolic name that represents some data that you set. To think of a variable name in real world terms, picture that the name is a grocery bag and the data it represents are the groceries. Java script supports 4 different data types. They are as follows :

Data Types

1. **Numbers:** Numbers are that can be processes and calculated. You don't enclose them in quotation marks. The numbers can be either positive or negative.
2. **Strings:** Strings are a series of letters and numbers enclosed in quotations marks. Java script uses the strings literally, it doesn't process it. You will use strings fro text.
3. **Boolean:** Lets you evaluate whether a condition meets or doesn't meet specific criteria.
4. **Null:** Null is an empty value, Null is not the same as 0—0 is a real, calculated number, where as null is the absence of any value.

Data Type	**Description**
Number	Any number with out quotes
String	A series of characters enclosed in quotes
Boolean	A logical value True/false
Null	A keyword meaning no value

Variables

A variable is a name value that you can use in your programs. A variable is a container for storing values. A variable value can change during the script's execution.

Creating a variable

To create variable, the key word "var" is preceded the variable name, and is used only once for declaration. Variable can be assigned values late on or immediately followed by the name with an equal sign the value they represent.

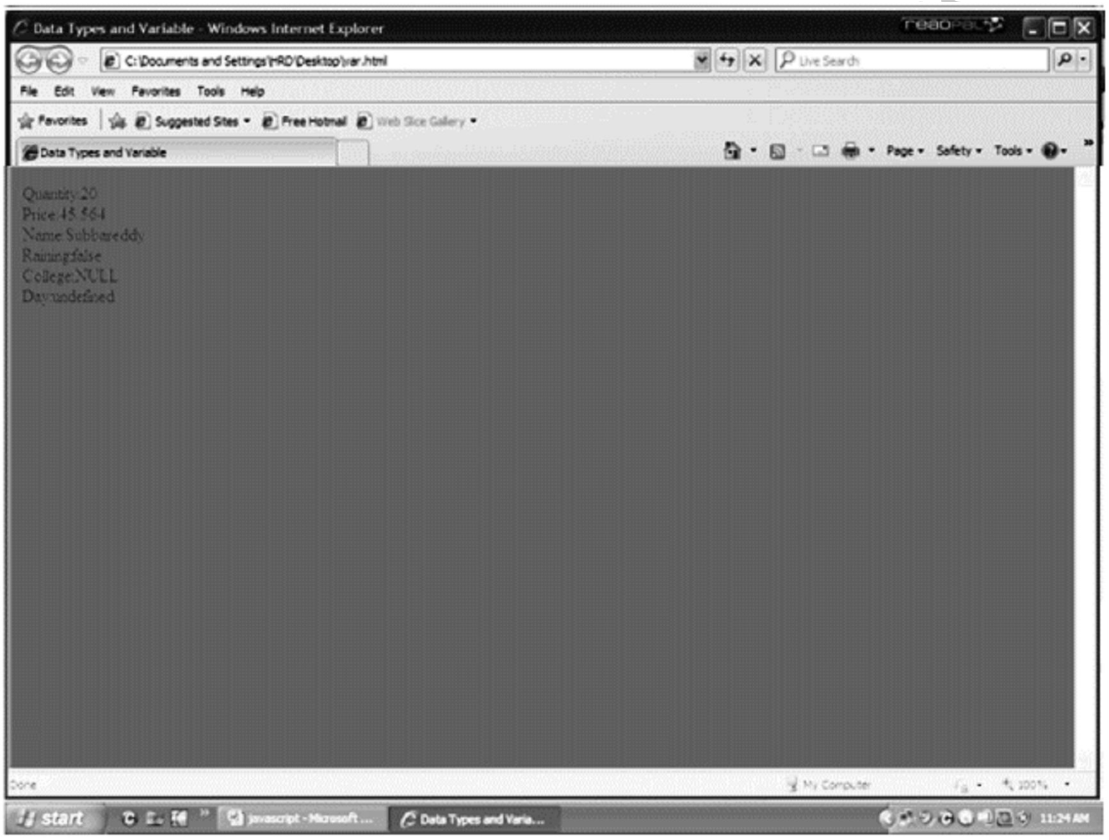
Observe the following examples

- Var qty=20; àNumber Data Type
- Var price=45.56;àNumber Data Type
- Var name=”Ramu”;àCharacter/String data Type
- Var raining=”true”;àBoolean Data Type
- Var ID=null;àNull Data Type
- Var deptUn-Defined

Example

```
<Html>
<Head>
<Title> Data Types and Variable</Title>
</Head>
<Body bgcolor=red text=blue>
<script language=”javascript”>
var qty=20;
var  price=45.564;
var name=”Rahul Publications”;
var raining=”false”;
var college=”NULL”;
var day;
document.writeln(“Quantity:”+qty+”<br>”);
document.writeln(“Price:”+price+”<br>”);
document.writeln(“Name:”+name+”<br>”);
document.writeln(“Raining:”+raining+”<br>”);
document.writeln(“College:”+college+”<br>”);
document.writeln(“Day:”+day+”<br>”);
</script>
</Body>
</Html>
```

Output



```
Quantity:20
Price:45.564
Name:Subbareddy
Raining:false
College:NULL
Day:undefined
```

Rules for Variable Declaration

1. A JavaScript variable must start with a letter or underscore (“_”).
2. Subsequent characters can also be digits (0-9) or any letters.
3. Java Script is case sensitive letter include the characters “A-Z” and “a-z”.

Variable Scope

The scope of the variable is the region of your program in which it is defined. Java script variables will have only two scopes.

- **Global Variable :** A global variable has global scope which it is defined everywhere in your java script 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.

3.4 Operators

3.4.1 Assignment increment, decrement operators

Q6. Explain different operators in JavaScript.

Ans : **(Imp.)**

Operators in java script are very similar to that appears in both other programming languages. The definition of an operator is a symbol that is used to perform an operation. Most often these operations are arithmetic but not always.

Operators in java script are classified into

- Arithmetic operators
- Logical operators
- Assignment operators
- Comparison operators
- Bitwise operators
- String operators

Arithmetic Operators

Arithmetic Operator takes numerical values, literals or variables as their operands and returns a single numerical value. The standard arithmetic operator are listed as below

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
++	Increment
--	Decrement

Comparison Operator

A comparison operator compares its operand and return a logical value based on weather the comparison is true or not. The comparison operators are listed below

Operator	Description
&&	AND returns if both operands are true
||	OR returns true if either is true
!	Not returns true if the negation of the operant is true

Bitwise Operator

Bitwise Operator treat their operands as a set of 32 bits, rather than as decimals, hexadecimal, or octal numbers. Bitwise operators perform their operations on such binary representations, but they return standard java script numerical values. The bitwise operators are listed below

Operator	Description
&	Bitwise AND
|	Bitwise OR
^	Bitwise XOR
<<	Shift Left
>>	Shift right
>>>	Shift right, zero fill

Assignment Operators

The assignment operators (=) lets you assign a value to a variable. You can assign and value to a variable, including another variable. The assignment operator are listed below

Operator	Description
=	Assign the value to the right hand operands to the variable on the left
+= Similarly -=,*=,/=	Add the value of the right hand operand to the left hand variable
&= Similarly! =	Assign the result of right hand to the left hand operand

String Operators

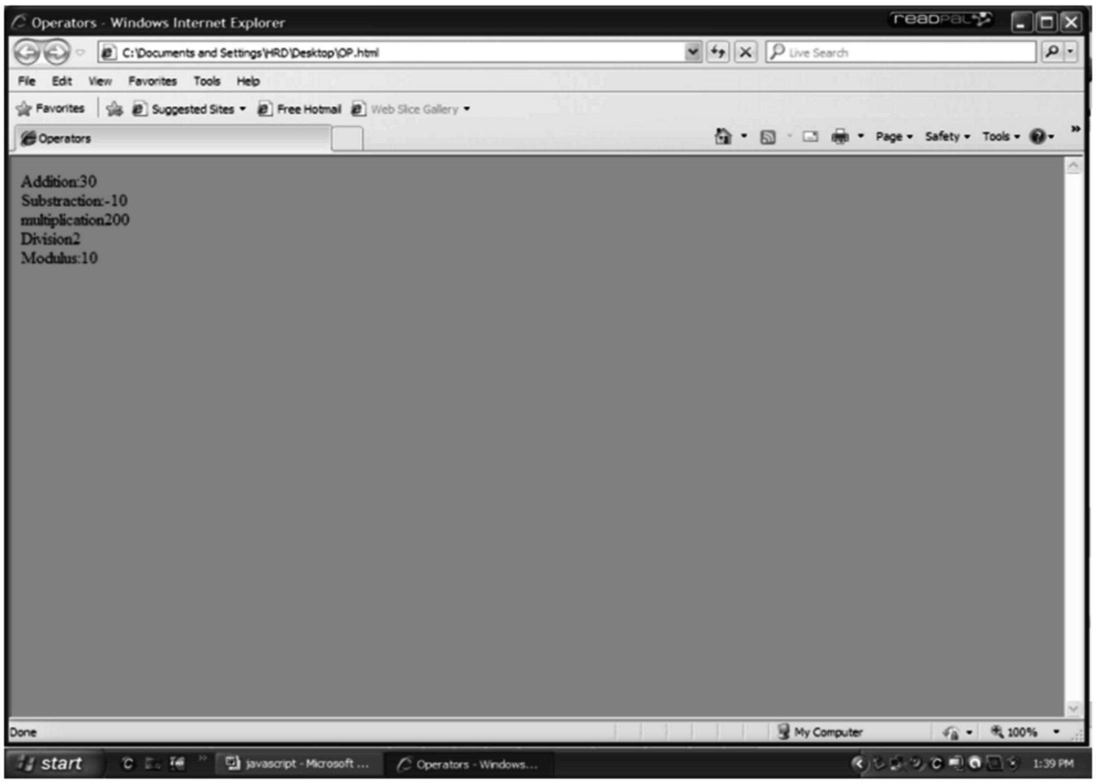
In addition to the comparison operator which can be used on string values, the concatenation operator (+) adds two string values together, returning another string. The short hand assignment operators += can also be used to concatenation string

Example

```
<Html>
<Head>
<Title>Operators</Title>
</Head>
<Body bgcolor=cyan text=blue>
<script language=”javascript”>
var a=10,b=20,c,d,e,f,g;
c=a+b;
d=a-b;
```

```
e=a*b;
f=b/a;
g=a%b;
document.writeln("Addition:"+c+"<br>");
document.writeln("Substraction:"+d+"<br>");
document.writeln("multiplication"+e+"<br>");
document.writeln("Division"+f+"<br>");
document.writeln("Modulus:"+g+"<br>");
</script>
</Body>
</Html>
```

Output


Operators - Windows Internet Explorer
C:\Documents and Settings\HRD\Desktop\OP.html
File Edit View Favorites Tools Help
Favorites
Suggested Sites
Free Hotmail
Web Slice Gallery
Operators
Page Safety Tools
Done
My Computer
100%
start
javascript - Microsoft ...
Operators - Windows...
1:39 PM

```
Addition:30
Substraction:-10
multiplication200
Division2
Modulus:10
```

3.5 Control Structures

3.5.1 If, If-else, While, For, Switch, Do/while

Q7. What are the control statements in Java Script?

Ans : **(Imp.)**

Statement

All java script is essentially a series of commands that are passed to the interpreter to be carried out sequentially. Java script usually requires that each one be placed on a separate line. These lines of code are referred to as statements.

Usually statements end with a line break, but if two statements are placed on the same line a semicolon must be used to separate them. The semicolon tells the java script interpreter that it has reached the end of a statement and should finish processing the code in front of the semicolon before it proceeds.

Control Statements

Control statements are designed to allow the user to create scripts that can decide which lines of code are evaluated, or how many times to evaluate them. Conditional statements are used to make necessary decisions.

There are two types of control statements

- Conditional Statements
- Loop Statements

Conditional Statements

There are 5 different types of conditional statements they are

- If
- If else
- Switch case
- Break
- Continue

(i) If Conditional Statement

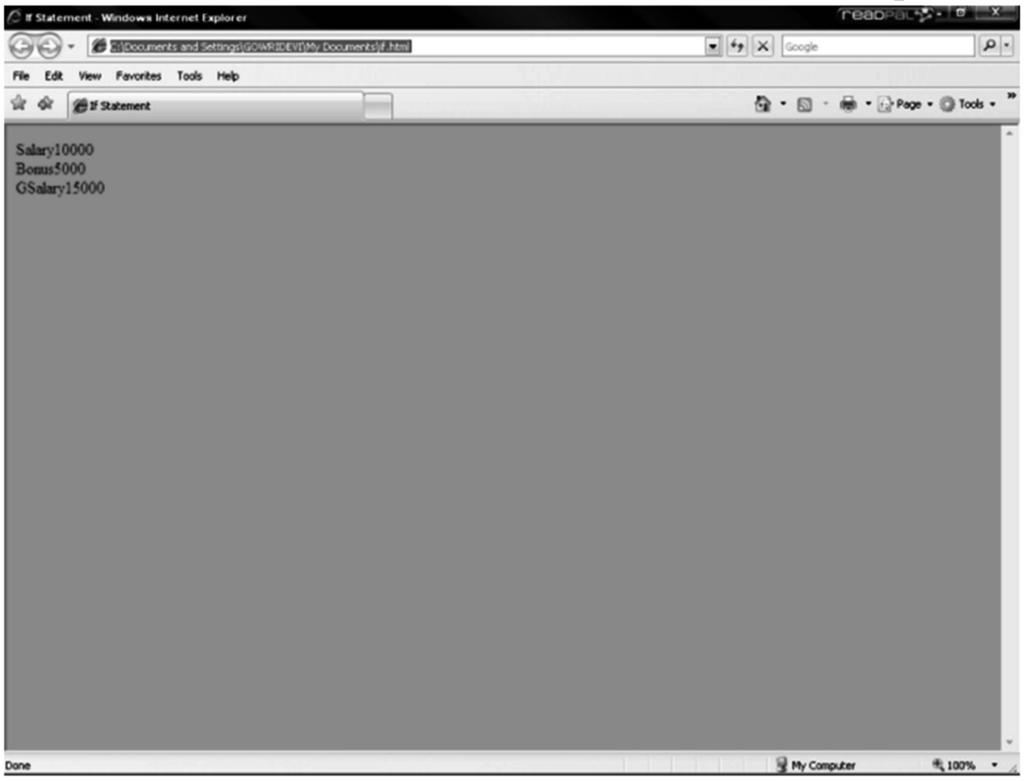
The **if statement** lets us to put the decision making in our script. A script without any decisions does the same procedure each time it is executed. Java script enables decision making using an if statement. If statement associated a single statement with a true condition, that condition is only executed if the condition expression is true.

```
If(Condition)
{
    Statements1;
    Statements2;
    .
    .
    .
}
```

Example

```
<Html>
<Head>
<Title>If Statement</Title>
</Head>
<Body bgcolor="cyan" text="blue">
<script language="javascript">
var salary=10000;
var bonus=5000;
var Gsalary;
if(salary>=10000)
{
        Gsalary=salary+bonus;
        document.writeln("Salary"+salary+"<br>");
        document.writeln("Bonus"+bonus+"<br>");
        document.writeln("GSalary"+Gsalary+"<br>");
}
</script>
</Body>
</Html>
```

Output



```
Salary10000
Bonus5000
GSalary15000
```

(ii) If else Conditional Statement

If statements are executed when the if condition is evaluate to true the additional else statement specifies a statement that is executed when the if condition is false this construction is specified as an alternative.

if(Condition)

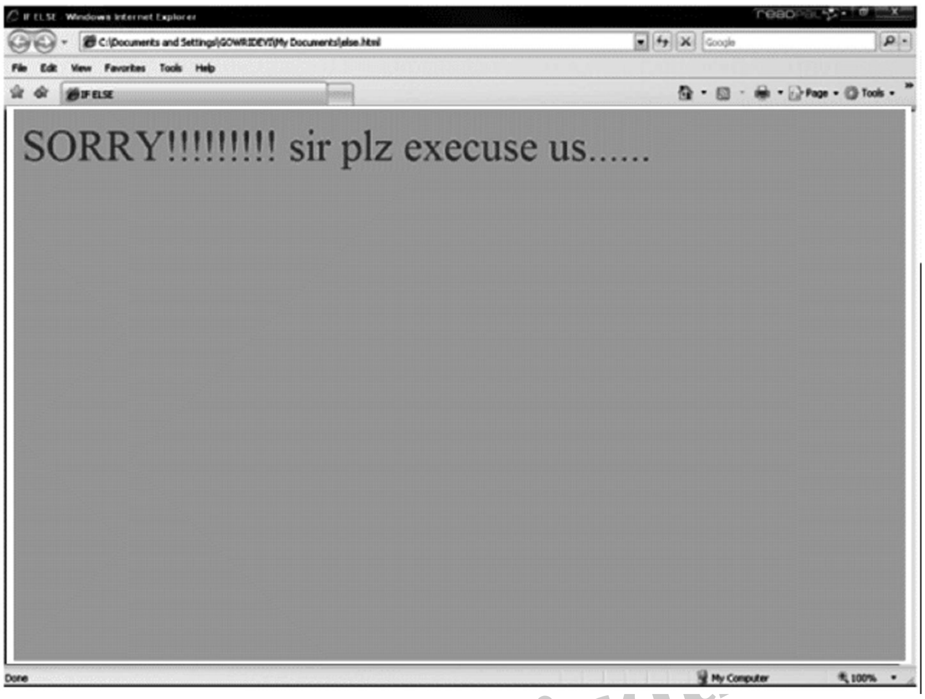
{

```
        Statements1;
        Statements2;
        .
        .
        .
}
else
{
        Statements1;
        Statements2;
        .
        .
        .
}
```

Example

```
<Html>
<Head>
<Title> IF ELSE</Title>
</Head>
<body bgcolor=green>
<script language=”javascript”>
var salary=1000;
if(salary>10000)
{
      document.writeln(“<font color=blue size=7>Pay Your Tax........      </font>”);
}
else
{
document.writeln(“<font color=blue size=7>SORRY!!!!!!!!! sir plz execuse us......</font>”);
}
</script>
</Body>
</Html>
```

Output


IF ELSE - Windows Internet Explorer
C:\Documents and Settings\GOWRIDEVI\My Documents\else.html
File Edit View Favorites Tools Help
IF ELSE
Page Tools
Done
My Computer
100%

```
SORRY!!!!!!!!! sir plz execuse us......
```

(iii) Switch Control Statement

When the **if else** structure build up, it becomes difficult to get a clear picture of the script. A simple system would solve the problem by offering multiple choices or action. A switch statement solves the problem which consists of 2 parts

- ➢ The declaration of the switch, which includes the value to be tested.
- ➢ A list of cases which associates actions with certain values. A specific action is taken when the switch value matches the values in the corresponding cases.

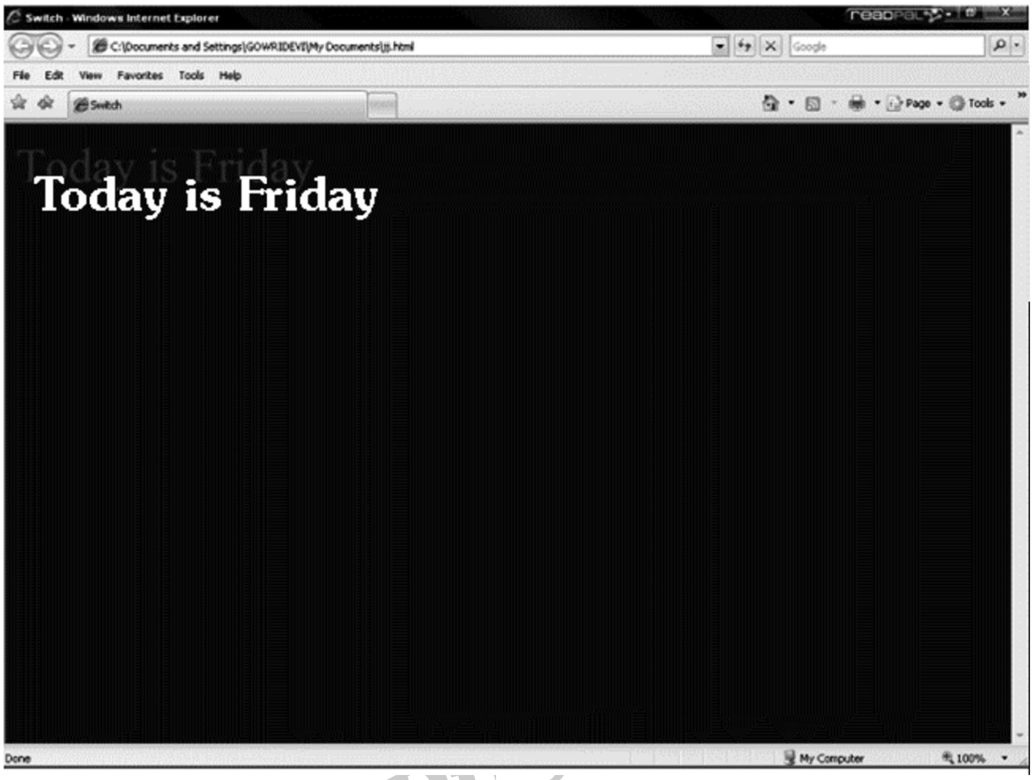
```
Switch (Condition)
{
        Cases value1:
                Statements;
                Break;
        Cases value2:
                Statements;
                Break;
        Cases value3:
                Statements;
                Break;
        Cases value4:
                Statements;
                Break;
        .
        .
        .
        Default:
              Statements
}
```

Break is an optional statement that tells "switch" to exit, should the proceeding statement be executed. Default allows you to specify the statement to execute if none of the above statements are executed.

Example

```
<Html>
<Head>
<Title>Switch</Title>
</Head>
<Body bgcolor=magnita>
<script>
var n=5;
switch(n)
{
case 1:
document.writeln("<font color=red size=7>"+"Today is Monday" +" </font>");
break;
case 2:
document.writeln("<font color=brick red size=6>"+"Today is Tuesday "+ "</font>");
break;
case 3:
document.writeln("<font color=voilet size=6>"+"Today is Wednesay"+ "</font>");
break;
case 4:
document.writeln("<font color=yellow size=6>"+"Today is Thusday"+ "</font>");
break;
case 5:
document.writeln("<font color=green size=7>"+"Today is Friday"+ "</font>");
break;
case 6:
document.writeln("<font color=blue size=6>"+"Today is Saturday"+ "</font>");
break;
default:
document.writeln("<font color=grey size=7>"+"ENTER THE CORRECT VALUE"+"</font>");
}
</script>
</Body>
</Html>
```

Output



```
Today is Friday
```

Q8. Define Break and Continue statements?

Ans :

Break

The break statement terminates the current process and transfer the control to the statement that follows the terminated statement.

Break

Continue

A continue statement terminates the execution of a block of statements and continue the execution of a block of statements with the next iteration.

Continue

3.5.2 Loop Control Statements

Q9. Explain about Loop control Statements?

Ans :

Loops are the control statements that performs asset of actions more than once. Every one agrees that a computer can calculate faster than a human being. Using loops you can repeat calculations and take advantage of the computer ability to do them faster, a loop repeats only one statement. However, you already know that the repetition of many statements, perhaps the whole programs.

Java script has 3 different features of loop control statements they are

(i) For loop

(ii) While loop

(iii) Do-while loop

Each looping statement has its own advantages.

(i) FOR Loop Statement

The most commonly used loop is FOR, because a loop usually repeats more than one statement. We can use a command block in the following format

```
for(initialization;condition;increment/decrement)
{
Statements;
}
```

Initialization is usually a statement or a variable declaration. It should evaluate to a single value is typically used to initialize a counter variable. This expression may optionally declare new variable keyword. Condition is a condition that is executed after each successive pass through the loop. The statement is executed only of the condition evaluate to true.

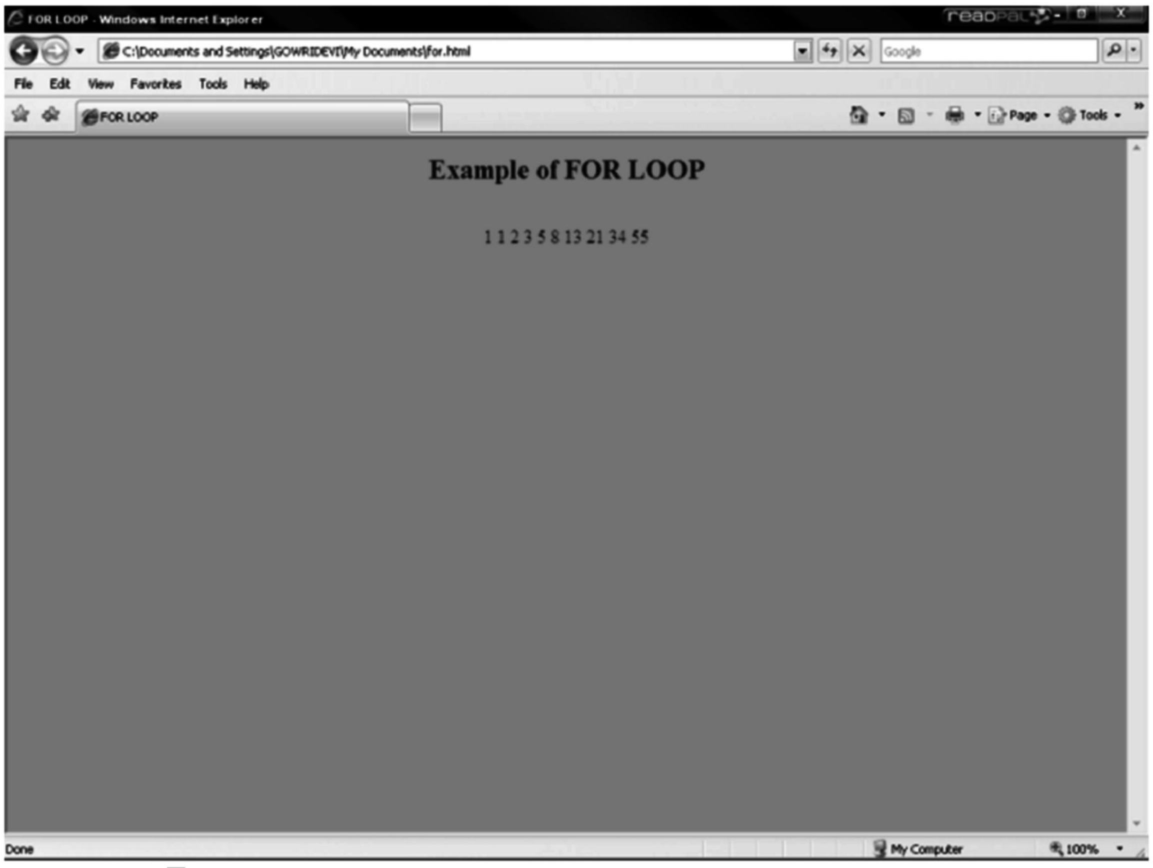
Increment/decrement is a statement that is executed after each consecutive pass a statement that is executed after each consecutive pass through the loops body. It is typically used to update or increment the counter variable, which counts the number of passes through the loop.

Example

```
<Html>
<Head>
<Title> FOR LOOP</Title>
</Head>
<Body bgcolor=cyan text=voilet>
<center>
<h2>Example of FOR LOOP</h2><br>
<script language=”javascript”>
var number1=1;
var number2=1;
var counter;
for(counter=1;counter<=10;counter++)
{
document.writeln(number1+”   “);
number2=number2+number1;
number1=number2-number1;
}
</script>
```

```
</center>
</body>
</Html>
```

Output



```
Example of FOR LOOP

1 1 2 3 5 8 13 21 34 55
```

(ii) While Loop Statement

The while loop continues to loop while the condition is true. As opposed to the for loop the loop is not executed at all. If the condition is false at the beginning. The condition is evaluated before each iteration statement can also be a common block, allowing multiple statements.

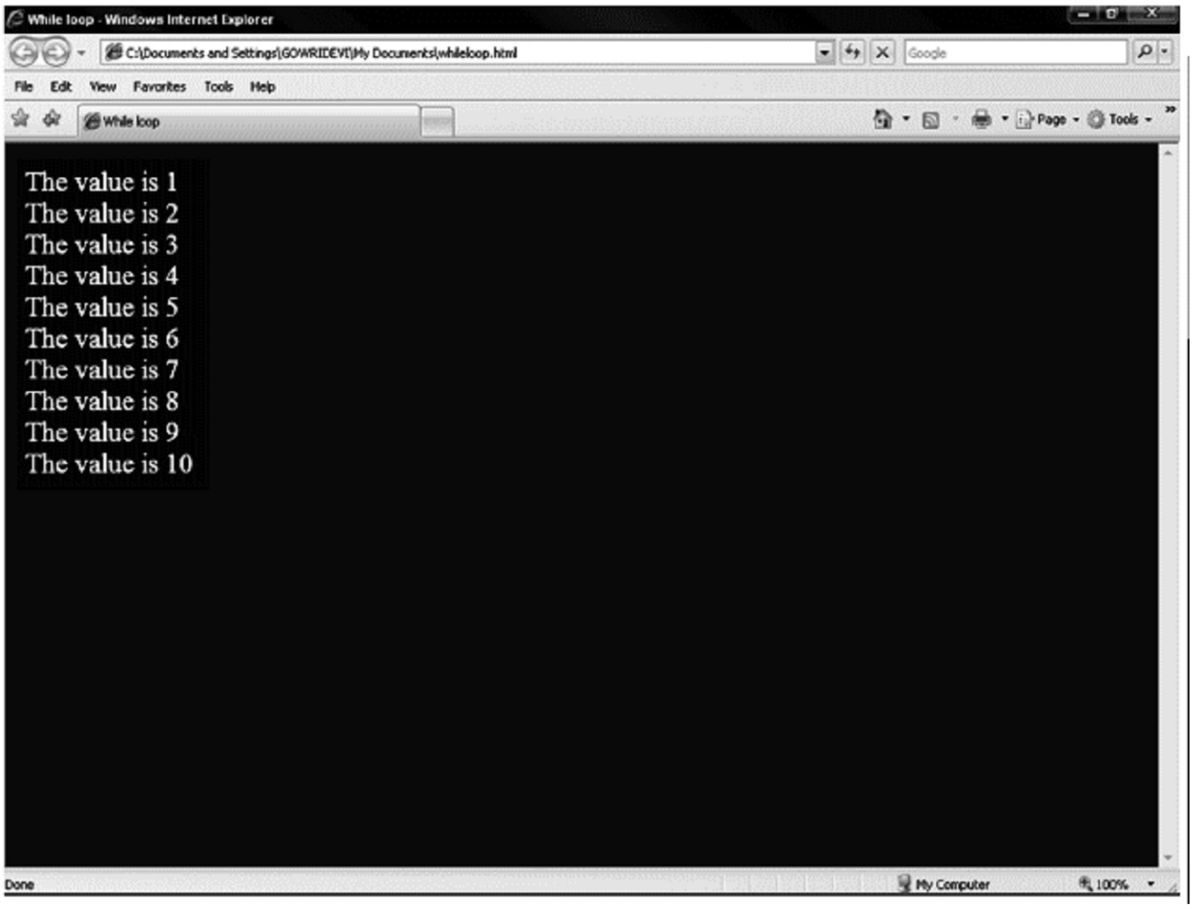
```
while (condition)
{
    Statements;
    Increment/Decrement;
}
```

Example

```
<Html>
<Head>
<Title>While loop</Title>
</Head>
```

```
<Body bgcolor=blue text=yellow>
<Script language="javascript">
var n=1;
while(n< =10)
{
document.writeln("<font size=5>"+"The value is "+n+ "<br> <font>");
n++;
}
</script>
</Body>
</Html>
```

Output



```
The value is 1
The value is 2
The value is 3
The value is 4
The value is 5
The value is 6
The value is 7
The value is 8
The value is 9
The value is 10
```

(iii) Do-While loop

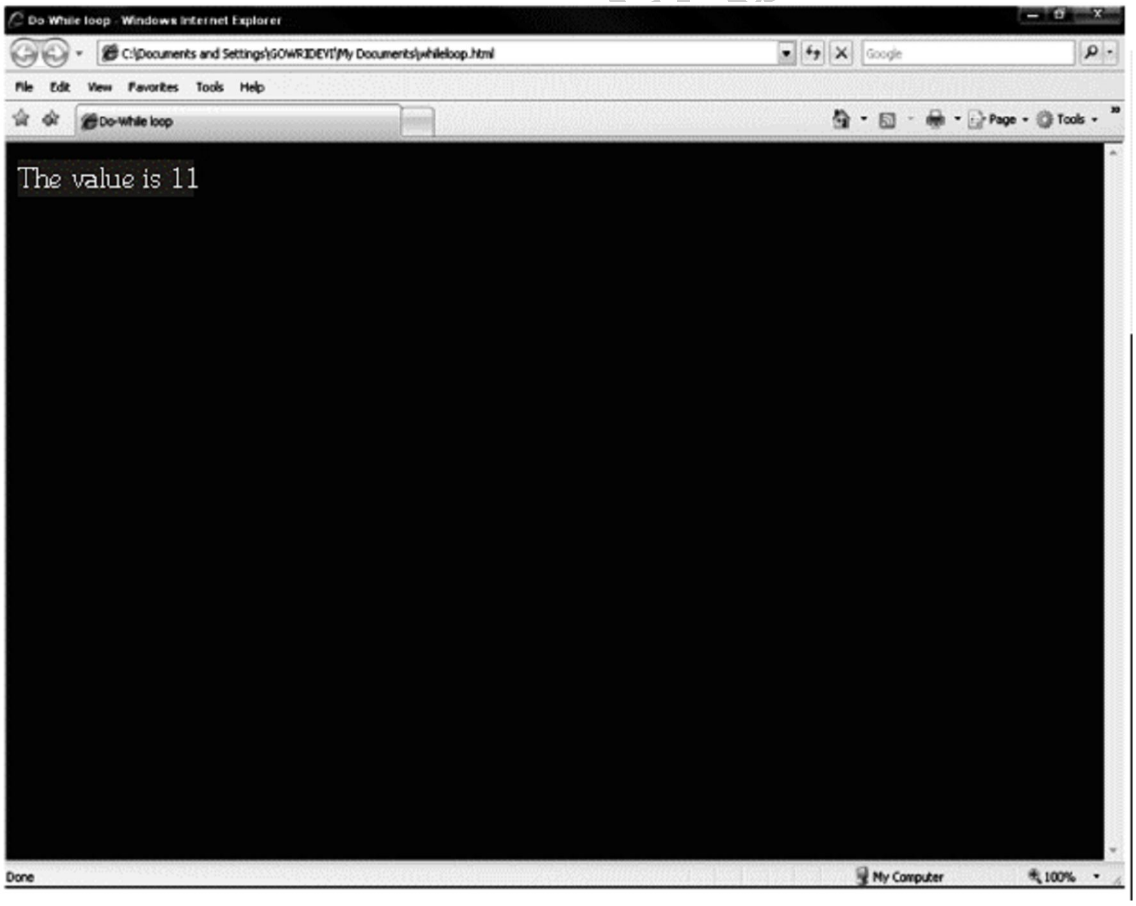
Do while loop runs the same block of statements while or until a condition is false, because the statements are executed before the condition is tested.

```
do
{
    Statements;
}while(condition);
```

Example

```
<Html>
<Head>
<Title>Do-While loop</Title>
</Head>
<Body bgcolor=blue text=yellow>
<Script language=”javascript”>
var n=11;
do
{
document.writeln(“<font size=5>”+”The value is “+n+”<br> <font>”);
n++;
}while(n<=10);
</script>
</Body>
</Html>
```

Output


Do While loop - Windows Internet Explorer
C:\Documents and Settings\GOWRIDEVI\My Documents\whileloop.html
Google
File Edit View Favorites Tools Help
Do-While loop
Page Tools
Done
My Computer
100%

```
The value is 11
```

3.6 Recursion

Q10. Explain the concept of Recursion.

Ans :

A recursive function is a function that calls itself, either directly, or indirectly through another function. Recursion is an important computer science topic. In this section, we present a simple example of recursion.

We consider recursion conceptually first; then we examine several programs containing recursive functions. Recursive problem-solving approaches have a number of elements in common. A recursive function is called to solve a problem. The function actually

knows how to solve only the simplest case(s), or base case(s). If the function is called with a base case, the function returns a result. If the function is called with a more complex problem, it divides the problem into two conceptual pieces a piece that the function knows how to process (the base case) and a piece that the function does not know how to process. To make recursion feasible, the latter piece must resemble the original problem but be a simpler or smaller version of it. Because this new problem looks like the original problem, the function invokes (calls) a fresh copy of itself to go to work on the smaller problem; this invocation is referred to as a recursive call, or the recursion step. The recursion step also normally includes the keyword return, because its result will be combined with the portion of the problem the function knew how to solve to form a result that will be passed back to the original caller.

The recursion step executes while the original call to the function is still open (i.e., it has not finished executing). The recursion step can result in many more recursive calls as the function divides each new subproblem into two conceptual pieces. For the recursion eventually to terminate, each time the function calls itself with a simpler version of the original problem, the sequence of smaller and smaller problems must converge on the base case. At that point, the function recognizes the base case, returns a result to the previous copy of the function, and a sequence of returns ensues up the line until the original function call eventually returns the final result to the caller. This process sounds exotic when compared with the conventional problem solving we've performed to this point.

As an example of these concepts at work, let's write a recursive program to perform a popular mathematical calculation. The factorial of a non negative integer n, written n! (and pronounced "n factorial"), is the product

n.(n – 1).(w - 2) 1

Where 1! is equal to 1 and 0! is defined as 1. For example, 5! is the product 5 . 4 . 3 . 2 . 1, which is equal to 120.

The factorial of an integer (number in the following example) greater than or equal to zero can be calculated iteratively (non-recursively) using a for statement, as follows:

```
var factorial = 1;

for ( var counter = number; counter >= 1; -- counter )

  factorial *= counter;
```

A recursive definition of the factorial function is arrived at by observing the following relationship:

n! = n. (n – 1)!

For example, 5! is clearly equal to 5 * 4!, as is shown by the following equations:

5! = 5 . 4 . 3 . 2. 1

5! = 5 . (4 . 3 . 2 . 1)

5! = 3. (4!)

The evaluation of 5! would proceed as shown in Fig. 9.11. Figure 9.11 (a) shows how the succession of recursive calls proceeds until 1! is evaluated to be 1, which terminates the recursion. Figure 9.11 (b) shows the values returned from each recursive call to its caller until the final value is calculated and returned.

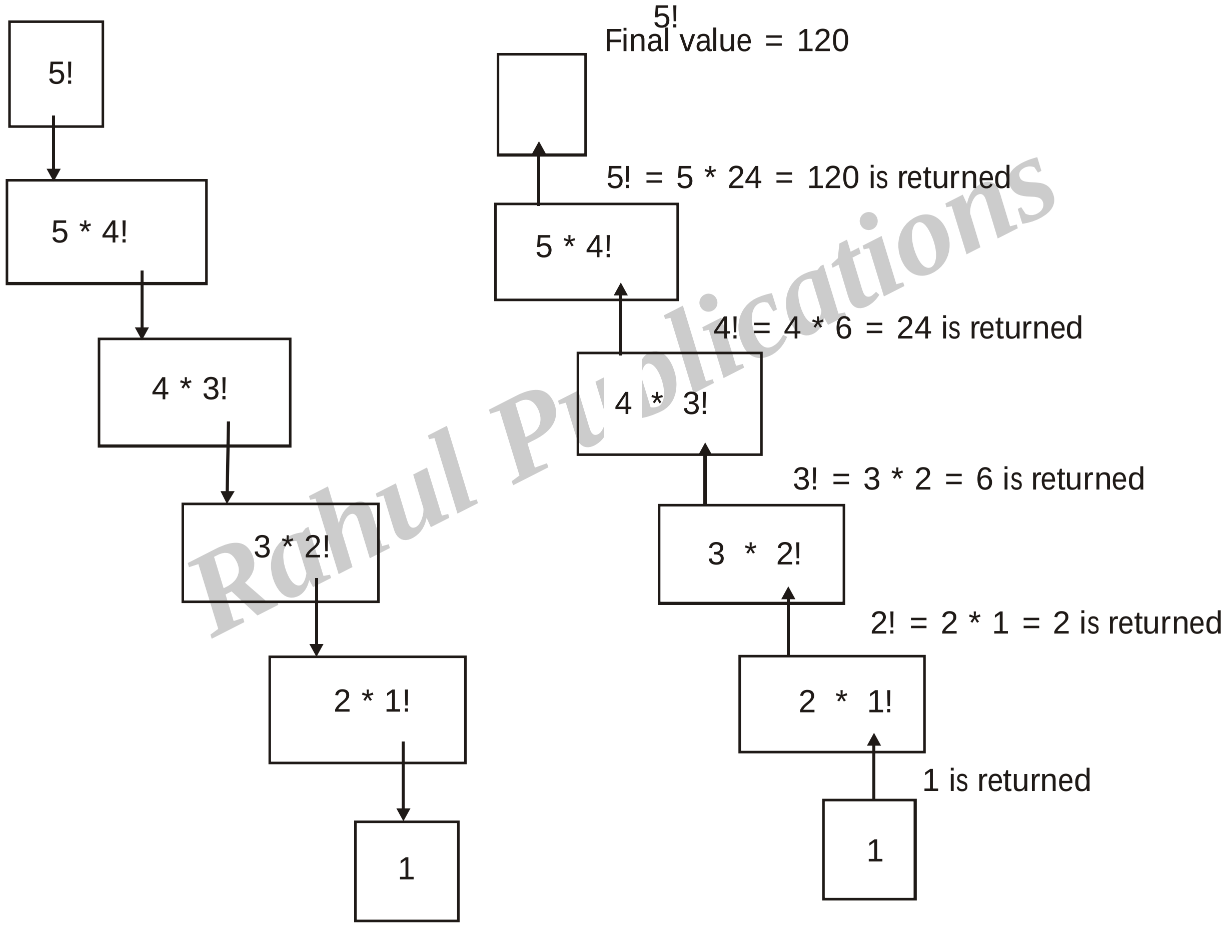

5!
5 * 4!
4 * 3!
3 * 2!
2 * 1!
1
5!
Final value = 120
5! = 5 * 24 = 120 is returned
5 * 4!
4! = 4 * 6 = 24 is returned
4 * 3!
3! = 3 * 2 = 6 is returned
3 * 2!
2! = 2 * 1 = 2 is returned
2 * 1!
1 is returned
1

Fig.: (a) Sequence of recursive calls **Fig.: (b) Values returned from each recursive call**

Uses recursion to calculate and print the factorials of the integers 0 to 10. The recursive function factorial first tests (line 27) whether a terminating condition is true, i.e., whether number is less than or equal to 1. If so, factorial returns 1,no further recursion is necessary and the function returns. If number is greater than 1, line 30 expresses the problem as the product of number and the value returned by a recursive call to factorial evaluating the factorial of number –1. Note that factorial (number –1) is a simpler problem than the original calculation, factorial (number).

```
<!DOCTYPE html>

<!-- Fig. 9.12: FactorialTest.html -->
<!-- Factorial calculation with a recursive function. -->
<html>
    <head>
        <meta charset = "utf-8">
        <title>Recursive Factorial Function</title>
        <style type = "text/css">
        p      { margin: Opx; }
        </style>
        <script>
        var output =   // stores the output

        // calculates factorials of 0 - 10
        function calculateFactorialsO
        {
```

Fig.: Factorial calculation with a recursive function. (Part I of 2.)

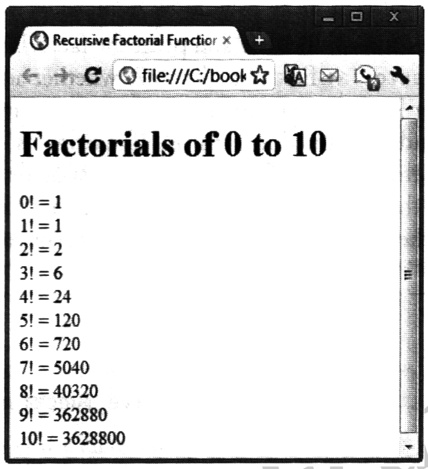
```
    for ( var i =0; i <=10; ++i )
    output += "<p>" + i + "! = " + factorial( i ) + "</p>";

    document.getElementById( "results" ).innerHTML = output;
    } // end function calculateFactorials

    // Recursive definition of function factorial
    function factorial( number )
    {
        if ( number <= 1 )     // base case
            return 1;
        else
            return number * factorial( number - 1 );
    }   //    end  function   factorial

        window.addEventListener( "load", calculateFactorials, false );
    </script>
```

```
        </head>
        <body>
              <hl>Factorials of 0 to 10</hl>
              <div id     =  "results"></div>
        </body>
  </html>
```



Factorials of 0 to 10

```
0! = 1
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
7! = 5040
8! = 40320
9! = 362880
10! = 3628800
```

3.6.1 Recursion vs Iteration

Q11. Compare and contrast Recursion vs Iteration.

Ans :

In the preceding section, we studied a function that can easily be implemented either recursively or iteratively. In this section, we compare the two approaches and discuss why j might choose one approach over the other in a particular situation.

Both iteration and recursion are based on a control statement: Iteration uses a repetition statement (e.g., for, while or do...while); recursion uses a selection statement (e.g., if, if...else or switch).

Both iteration and recursion involve repetition: Iteration explicitly uses a repetition cement; recursion achieves repetition through repeated function calls.

Iteration and recursion each involve a termination test: Iteration terminates when the loop-continuation condition fails; recursion terminates when a base case is recognized.

Iteration both with counter-controlled repetition and with recursion gradually approaches termination: Iteration keeps modifying a counter until the counter assumes a value that makes the loop-continuation condition fail; recursion keeps producing simpler versions of the original problem until the base case is reached.

Both iteration and recursion can occur infinitely: An infinite loop occurs with iteration if the loop-continuation test never becomes false; infinite recursion occurs if the recursion step does not reduce the problem each time via a sequence that converges on the base case or if the base case is incorrect.

One negative aspect of recursion is that function calls require a certain amount of time md memory space not directly spent on executing program instructions. This is known as function-call overhead. Because recursion uses repeated function calls, this overhead greatly beets the performance of the operation. In many cases, using repetition statements in place of recursion is more efficient. However, some problems can be solved more elegantly md more easily) with recursion.

3.7 Arrays in Java Script

Q12. What are arrays in JavaScript? Explain its Methods.

Ans : **(Imp.)**

An array is an ordered set of data elements which can be accessed through a single variable name. Conceptually an array is made of a set of slots with each slot assigned to the single data element. You access the data element either sequentially by reading from the start of the array, or by their indexes is the position of the element in the array with first element being at position 0 and the last at (array length-1).

Item 1	Item 2	Item 3	Item 4	. . . .	Item N
0	1	2	3	. . . .	n

Structure of an Array

In java script an array is slightly different because it is a special type of objects and has functionality which is not normally available in other languages. It sole function is to hold data until the script requires it. The data inside an array is ordered, because elements are added and accessed in a particular order, but is not sorted. There is no relationship between the way the data is held and any external meaning it has. If the words are being added to the array they are not necessary going to be stored in alphabetical order. The content of the array to the may have a particular order as an artifact of the way that they were presented to the array but not as a result of being in an array.

An array can hold all your variables 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.

Creating an Array

An array can be defined in 3 different ways

1. var array_name=new Array(); àregular array (add an optional integer).

Array_name[0]=”value”;

Array_name[1]=”value”;

Array_name[2]=”value”;

2. var array_name=new Array(value1,value2,value3,………);

3. var array_name=[value1,value2,value3,…….];

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

Adding Elements to an Array

Array elements are accesses by their indexes. The index denotes the position of the element in the array and, as in for loops, this start from 0. Adding an element uses the square brackets.

var array_name[index]=”value”;

array_name[index]=”value”;

Accessing Array Members

The elements in the array are accessed through their indexes. The same access method is used to find elements and to change their value.

var array_name=[“value1”,”value2",”value3",..........];

for(var variable=index value; variable=condition; variable++)

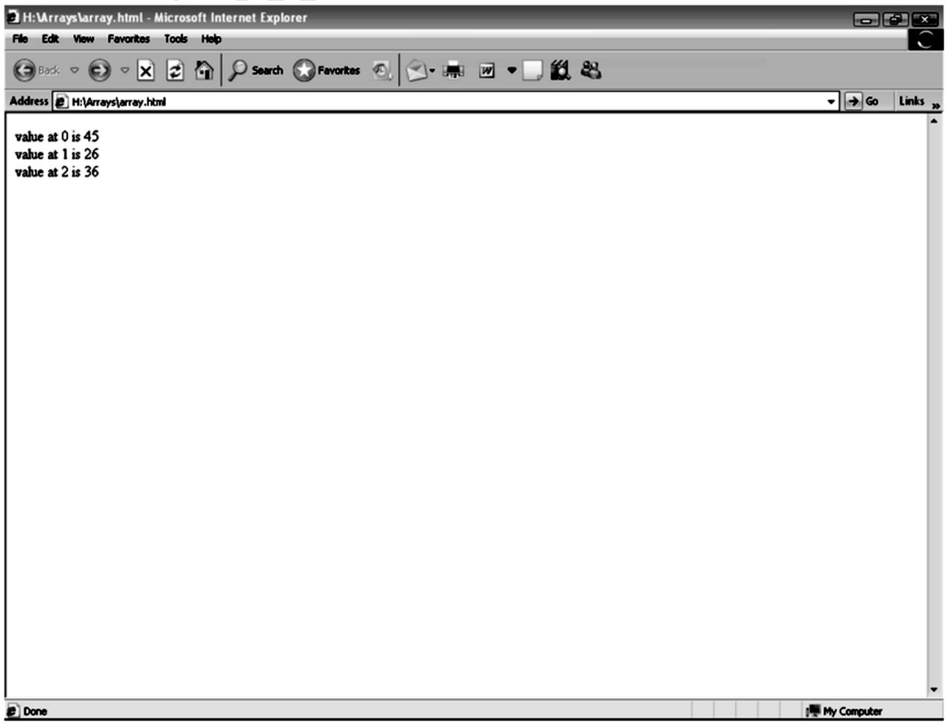
{

```
        Stmts;
        document.write(array_name[index value]+" ,");
}
```

Program to demonstrate Arrays in Java Script

```
<html>
<head>
<title>Arrays</title>
</head>
<body bgcolor=magnita text=yellow>
<script language="javascript">
var mycars = new Array();
mycars[0] = 45;
mycars[1] = 26;
mycars[2] = 36;
for (i=0;i<mycars.length;i++)
{
document.write("value at "+i+" is "+ mycars[i] + "<br />");
}
</script>
</body>
</html>
```

Output


H:\Arrays\array.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address H:\Arrays\array.html
Go Links
Done
My Computer

```
value at 0 is 45
value at 1 is 26
value at 2 is 36
```

Object –Based Array Functions

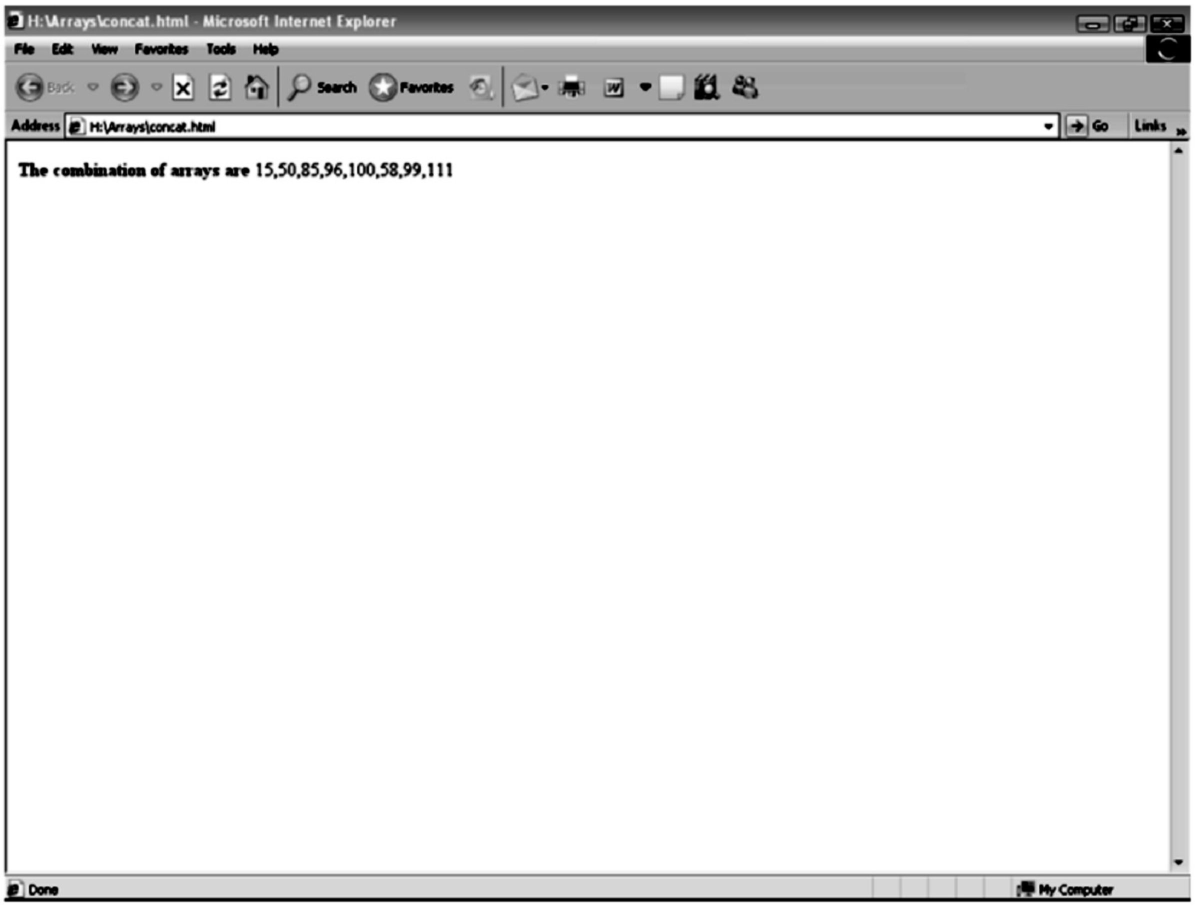
The following below table summarizes the method available in the arrays.

Method	Description
concat(value1,....)	Concatenates all the arguments values to the existing array, and returns the new array. Values can be another array.
join([separator])	Converts each element with the array the array to a string and joins them into one large string. Pass in an optional separator as arguments to be used to separate each array element.
pop()	Deletes the last element with in array and return the deleted element. Original array is modified.
push()	Adds the arguments values to the end of the array, and modify the original array with the new additions. Returns the bew length of arrays
reverse()	Reverse the order of all elements within the array. Original array is modified.
valueOf()	Returns the primitive value of the array.
shift()	Deletes and returns the first element with in the array. Original array is modified.

Program to concatenation of two arrays

```
<Html>
<Head>
<Title></title>
</Head>
<Body>
<script language="javascript">
var parents=[15,50,85,96];
var children = [100,58,99,111];
var family = parents.concat(children);
document.write("<b>The combination of arrays are </b>"+family);
</Script>
</Body>
</Html>
```

Output

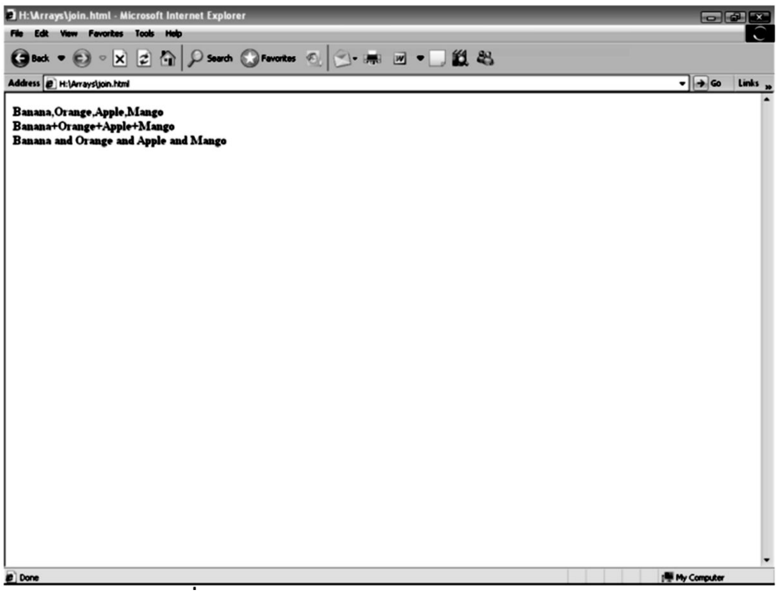

H:\Arrays\concat.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address H:\Arrays\concat.html

```
The combination of arrays are 15,50,85,96,100,58,99,111
```

Program to demonstrate join()

```
<html>
<head>
<title>Join </title>
</head>
<body>
<script language="javascript">
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write("<b>"+fruits.join() + "</b><br>");
document.write("<b>"+fruits.join("+") + "</b><br>");
document.write("<b>"+fruits.join(" and ")+"</b>");
</script>
</body>
</html>
```

Output

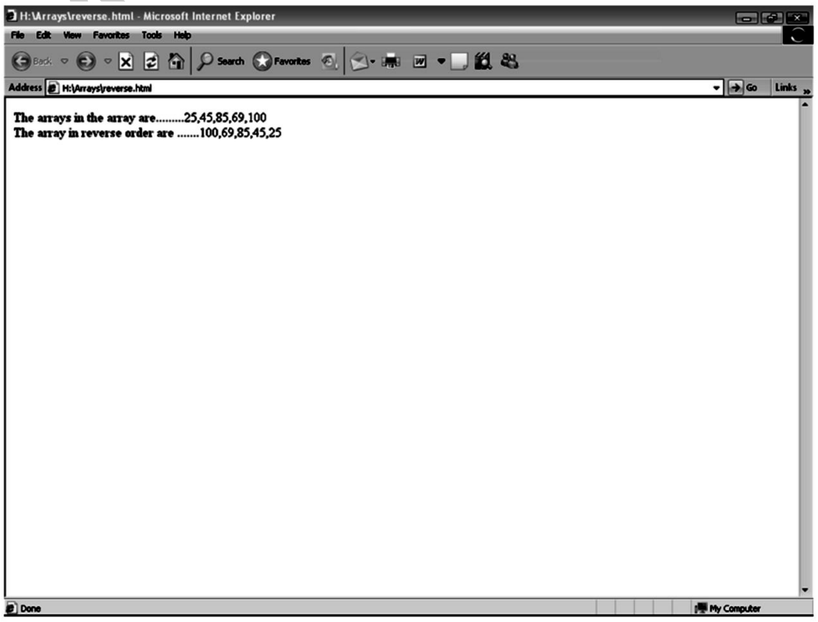


```
Banana,Orange,Apple,Mango
Banana+Orange+Apple+Mango
Banana and Orange and Apple and Mango
```

Program to print the array in the reverse order

```
<html>
<body>
<script language=”javascript”>
var fruits = [25,45,85,69,100];
document.write(“<b>The arrays in the array are......... </b>” +fruits + “<br>”);
document.write(“<b>The array in reverse order are .......</b>” + fruits.reverse());
</script>
</body>
</html>
```

Output

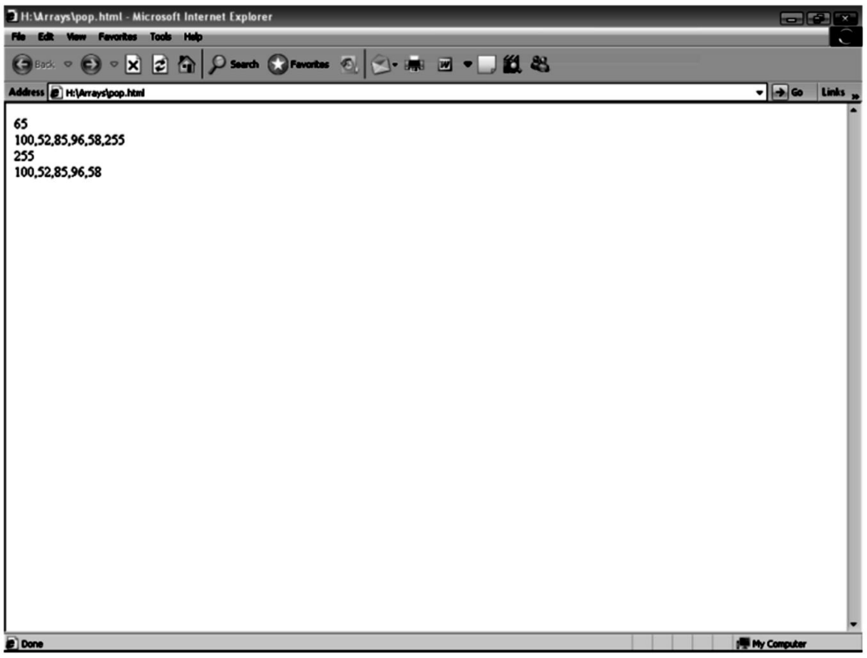


```
The arrays in the array are.........25,45,85,69,100
The array in reverse order are .......100,69,85,45,25
```

Program to remove the elements from the last

```
<html>
<head>
<title> </title>
</head>
<body>
<script language="javascript">
var fruits = [100,52,85,96,58,255,65];
document.write(fruits.pop() + "<br >");
document.write(fruits + "<br>");
document.write(fruits.pop() +"<br>");
document.write(fruits);
</script>
</body>
</html>
```

Output



```
65
100,52,85,96,58,255
255
100,52,85,96,58
```

3.7.1 Reference and reference parameters

Q13. Explain the concept of Reference and reference parameters in Java Script.

Ans :

Two ways to pass arguments to functions (or methods) in many programming languages are pass-by-value and pass-by-reference. When an argument is passed to a function by value, a copy of the argument's value is made and is passed to the called function. In Java-Script, numbers, boolean values and strings are passed to functions by value.

With pass-by-reference, the caller gives the called function access to the caller's data and allows the called function to modify the data if it so chooses. This procedure is accomplished by passing to the called function the address in memory where the data resides. Pass-by-reference can improve performance because it can eliminate the overhead of copying large amounts of data, but it can weaken security because the called function can access the caller's data. In JavaScript, all objects (and thus all arrays) are passed to functions by reference.

The name of an array actually is a reference to an object that contains the array elements and the length variable. To pass a reference to an object into a function, simply specify the reference name in the function call. The reference name is the identifier that the program uses to manipulate the object. Mentioning the reference by its parameter name in the body of the called function actually refers to the original object in memory, and the original object is accessed directly by the called function.

3.7.2 Passing arrays to functions

Q14. Explain the concept of Passing arrays to functions.

Ans :

To pass an array argument to a function, specify the array's name (a reference to the array without brackets. For example, if array hourly Temperatures has been declared as

var hourlyTemperatures = new Array(24);

then the function call

modifyArray (hourlyTemperatures);

passes array hourlyTemperatures to function modifyArray. As stated in Section 10.2, ev-ery array object in JavaScript knows its own size (via the 1 ength attribute). Thus, when we pass an array object into a function, we do not pass the array's size separately as an argument. Figure demonstrated this concept.

Although entire arrays are passed by reference, individual numeric and boolean arrc elements are passed by value exactly as simple numeric and boolean variables are passec. Such simple single pieces of data are called scalars, or scalar quantities. Objects referred to by individual array elements are still passed by reference. To pass an array element to a function, use the indexed name of the element as an argument in the function call.

For a function to receive an array through a function call, the function's parameter lb: must specify a parameter that will refer to the array in the body of the function. JavaScript does not provide a special syntax for this purpose it simply requires that the identifier for the array be specified in the parameter list. For example, the function header for function modify Array might be written as

function modifyArray (b)

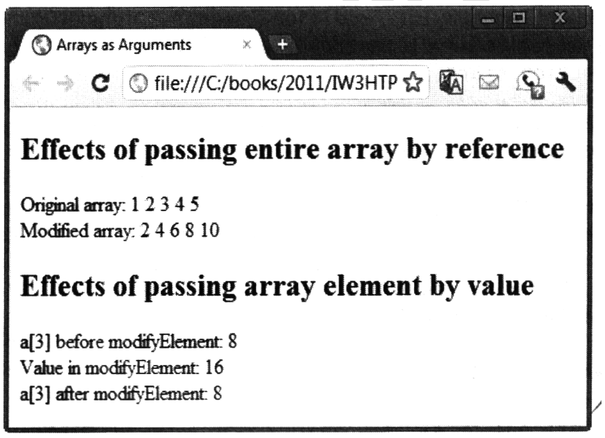
indicating that modi fyArray expects to receive a parameter named b. Arrays are passed b reference, and therefore when the called function uses the array name b, it refers to the ac tual array in the caller (array hourlyTemperatures in the preceding call). The script in Figures demonstrates the difference between passing an entire array an: passing an array element. The body of the document in Fig. contains the p element that the script in Fig. uses to display the results.

```
<!DOCTYPE html>

<!-- Fig. 10.13: PassArray.html -->
<!-- HTML document that demonstrates passing arrays and -->
<!-- individual array elements to functions. -->
<html>
```

Fig.: HTML document that demonstrates passing arrays and individual array elements functions. (Part I of 2.)

```
<head>
      <meta charset = "utf-8">
      <title>Arrays as Arguments</title>
      <link rel = "stylesheet" type = "text/css" href = "style.css">
      <script src = "PassArray.js"></script>
</head>
<body>
      <h2>Effects of passing entire array by reference</h2>
      <p id = "originalArray"></p>
      <p id = "modifiedArray"></p>
      <h2>Effects of passing array element by value</h2>
      <p id = "originalElement"></p>
      <p id = "inModifyElement"></p>
      <p id = "modifiedElement"></p>
      </body>
</html>
```



Arrays as Arguments

file:///C:/books/2011/IW3HTP

Effects of passing entire array by reference

Original array: 1 2 3 4 5
Modified array: 2 4 6 8 10

Effects of passing array element by value

a[3] before modifyElement: 8
Value in modifyElement: 16
a[3] after modifyElement: 8

3.8 Functions in Java Script

Q15. What are JavaScript Functions? Explain its types.

Ans : **(Imp.)**

Java script Function is nothing but it is a reusable code-block that is execute when the function is called. Function is defined in the head section of the code. The syntax of JavaScript function is as follows:

function fname(prameter1,parameter2, ...)
{
JavaScript code 1
JavaScript code 2
....

}

A function is a set of statements under a single name. This allows to conveniently “CALL” the function when ever its action is required. Functions are fundamental buildings blocks of most java script programs.

Function Properties

Functions in JavaScript are actually an object. Therefore any created function created using the “function” declaration will have the properties of a function. These properties are

- **Arguments:** An array of arguments passed to the function. This is an array object which has a property of length which enables the programmer to tell how many arguments (or variables) are passed to the function.
- **Caller:**The name of the function that called the function.
- **Prototype:** Used to make more properties.

We can break down the use of functions then, into two logical categories

1. Defining a function
2. calling a function

Defining a Function

The function definition is a statement which describes the function like name, any value (arguments) which it accepts, and the statements which the function comprises of.

```
Function function_name(arguments)
{
        Statements();
}
```

function should be in lowercase other wise JavaScript won’t understand it as function. If you write function in uppercase the code will generate error. In the JavaScript semicolon is optional but its better to put semi-colon as part of best practices. All the java script code is written inside the curly braces.

Calling a function

A function is the waits until it is called on to the stage. We call a function simply by specifying its name followed by a list of arguments.

Function_name(arguments)

Using a function call inside event handler. Call the function through the onclick() event handler.

Onclick=”function_name()

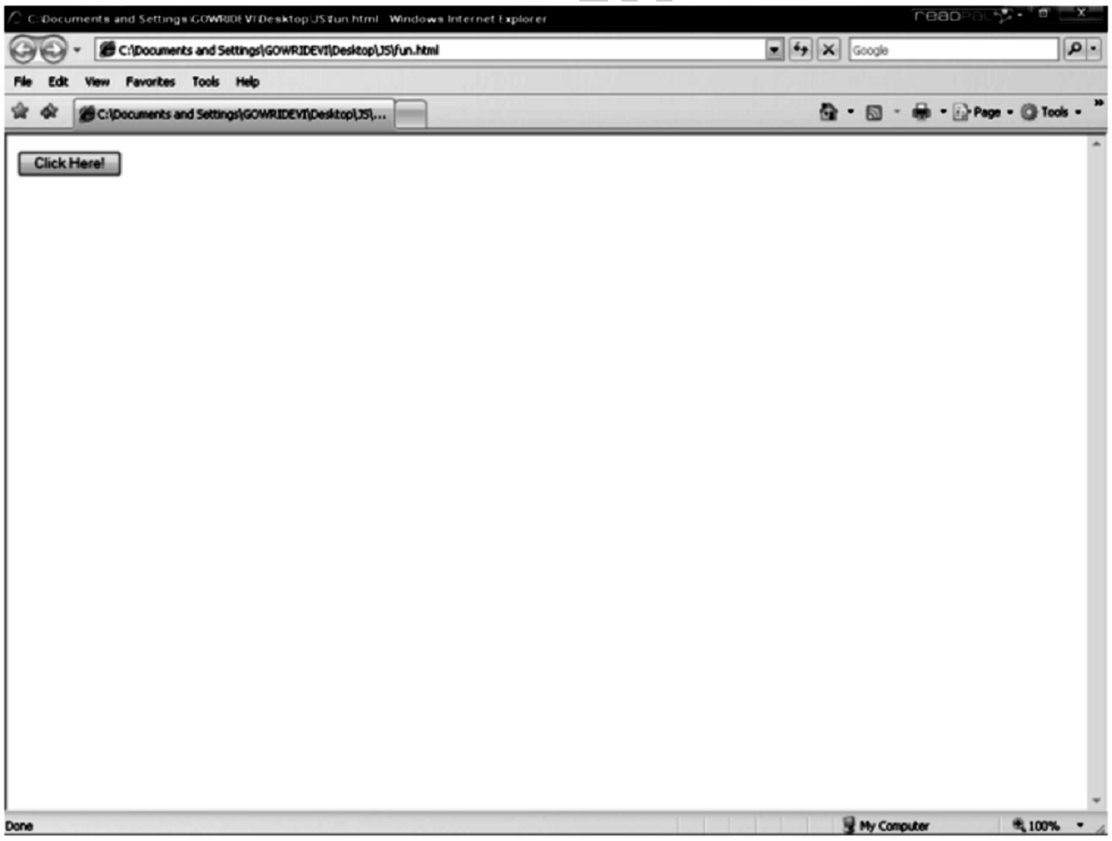
Use of JavaScript Functions

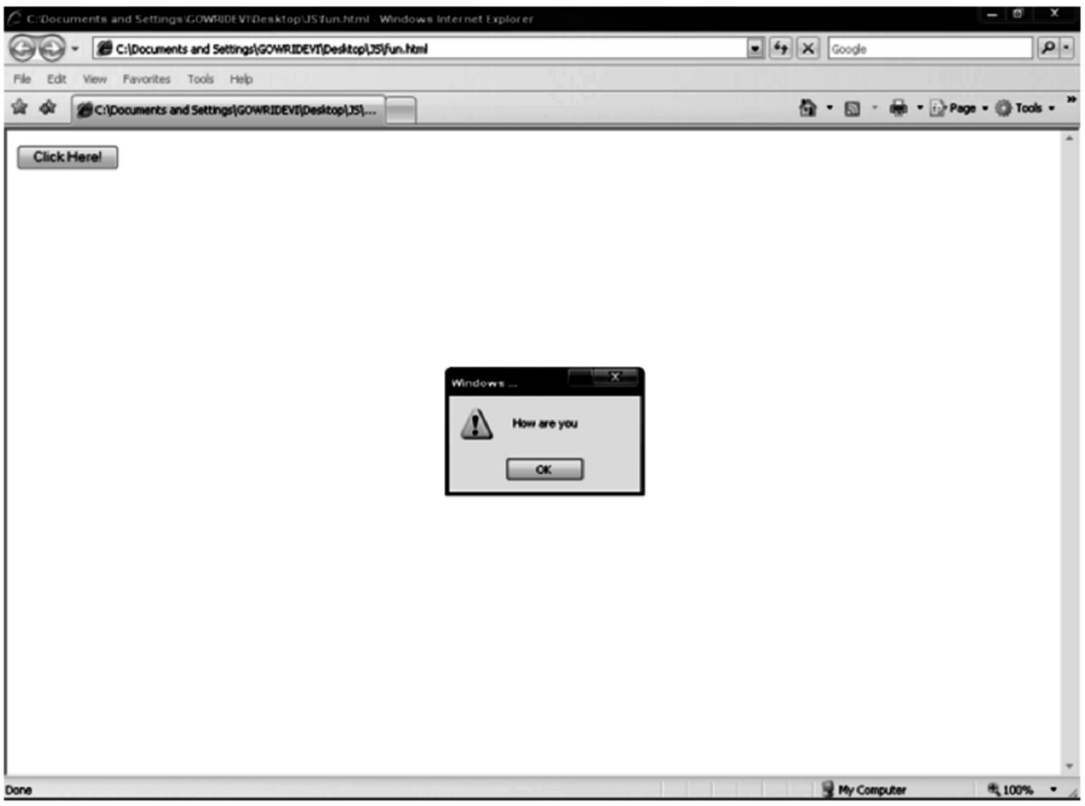
The java Script Function is very useful in writing the JavaScript code. It is used to group many JavaScript codes under one name.

Example

```
<html>
<head>
<script language="javascript">
function showmessage()
{
alert("How are you");
}
</script>
</head>
<body>
<form>
<input type="button" value="Click Here!" onclick="showmessage()" >
</form>
</body>
</html>
```

Output


C:\Documents and Settings\GOWRIDEVI\Desktop\JS\fun.html - Windows Internet Explorer
C:\Documents and Settings\GOWRIDEVI\Desktop\JS\fun.html
Google
File Edit View Favorites Tools Help
C:\Documents and Settings\GOWRIDEVI\Desktop\JS\...
Page Tools
Click Here!
Done
My Computer
100%



Built-in functions

Build in Function	Function Description
alert()	The alert() built-in function displays the alert dialog box.
confirm()	The confirm() built-in function display the confirmation dialog box. and ask the user to determine from the two option .
focus()	The focus() built -in function built the pointed object active and put the curser on the text field.
prompt()	The prompt() built -in function display the prompt dialog box. Inquiring the user for input.
select()	The select () built -in function used to select the pointed object.
write()	The write() built in function used to write something on the document.

Function Arguments

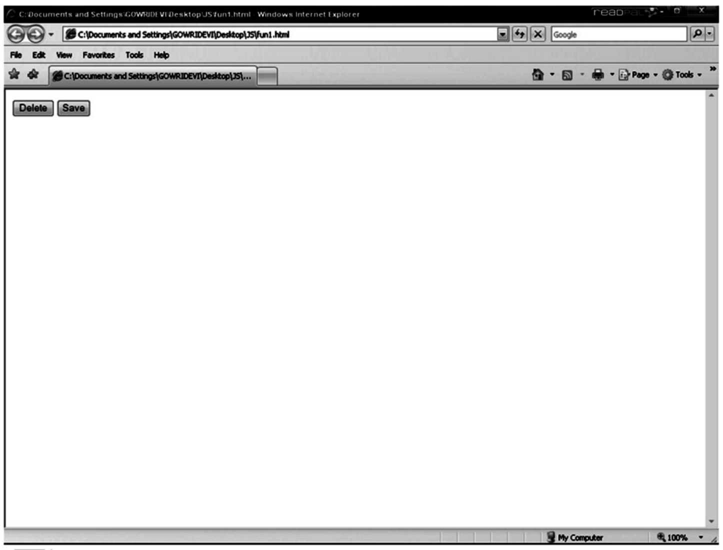
Through variable you can pass argument to function. The out put of the function looks on the arguments given by you.

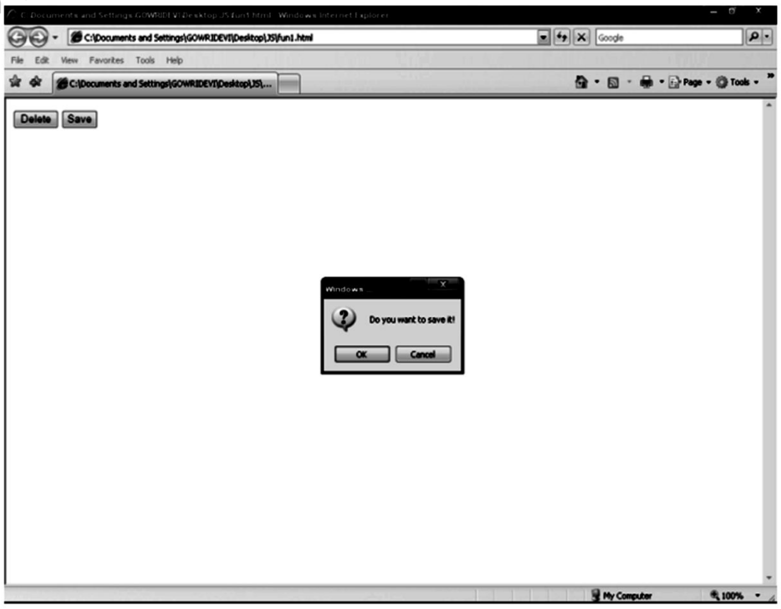
Example

```
<html>
<head>
<script language=”javascript”>
function myfunction(text)
```

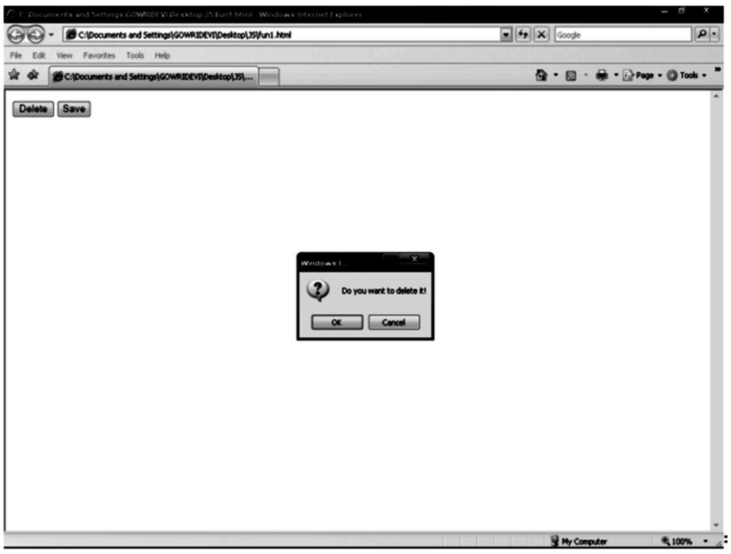
```
{
confirm(text)
}
</script>
</head>
<body>
<form>
<input type=”button” onclick=”myfunction(‘Do you want to delete it!’)”  value=”Delete”>
<input type=”button”  onclick=”myfunction(‘Do you want to save it!’)” value=”Save”>
</form>
</body>
</html>
```

Output





Do you want to save it!



Do you want to delete it!

3.8.1 Mathematical Functions

Q16. List and Explain Some Mathematical Functions related to math object.

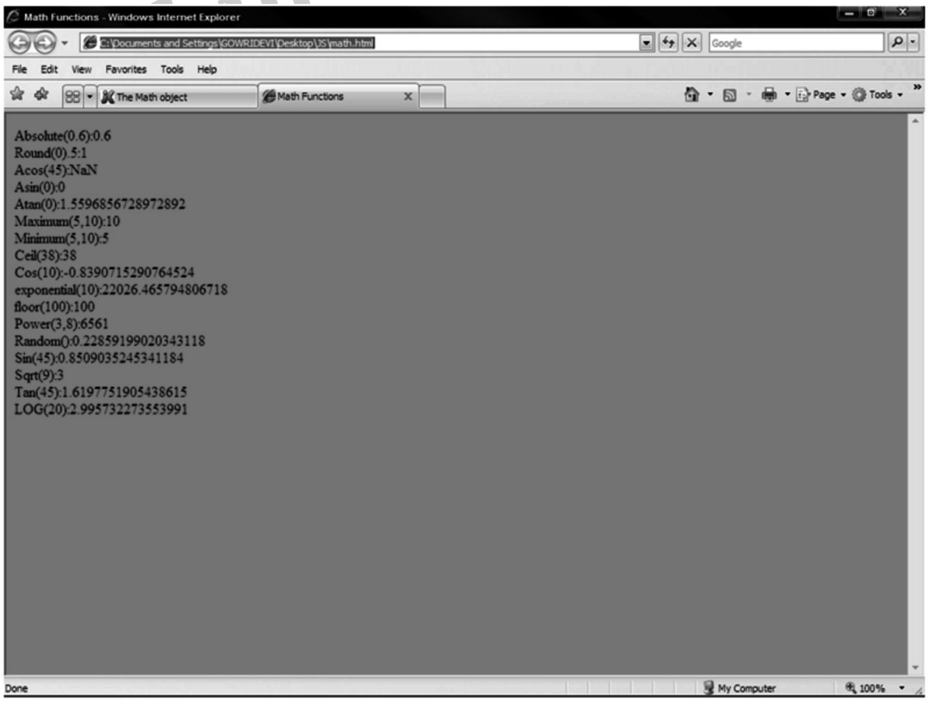
Ans :

Method	Description
abs()	Returns the absolute value of a number.
acos()	Returns the arccosine (in radians) of a number.
asin()	Returns the arcsine (in radians) of a number.
atan()	Returns the arctangent (in radians) of a number.
atan2()	Returns the arctangent of the quotient of its arguments.
ceil()	Returns the smallest integer greater than or equal to a number.
cos()	Returns the cosine of a number.
exp()	A RETURN E^N, where N is the argument and E is Euler's constant, the base of the natural logarithm.
floor()	Returns the largest integer less than or equal to a number.
log()	Returns the natural logarithm (base E) of a number.
max()	Returns the largest of zero or more numbers.
min()	Returns the smallest of zero or more numbers.
pow()	Returns base to the exponent power, that is, base exponent.
random()	Returns a pseudo-random number between 0 and 1.
round()	Returns the value of a number rounded to the nearest integer.
sin()	Returns the sine of a number.
sqrt()	Returns the square root of a number.
tan()	Returns the tangent of a number.

Program to demonstrate the above math functions

```
<html>
<head>
<title>Math Functions</title>
</head>
<body bgcolor=cyan>
<script type="text/javascript">
document.write("Absolute(0.6):"+Math.abs(0.60) + "<br>");
document.write("Round(0).5:"+Math.round(0.50) + "<br>");
document.write("Acos(45):"+Math.acos(45) + "<br>");
document.write("Asin(0):"+Math.asin(0) + "<br>");
document.write("Atan(0):"+Math.atan(90)+"<br>");
document.write("Maximum(5,10):"+Math.max(5,10) + "<br>");
document.write("Minimum(5,10):"+Math.min(5,10) + "<br>")
document.write("Ceil(38):"+Math.ceil(38) + "<br>");
document.write("Cos(10):"+Math.cos(10) + "<br>");
document.write("exponential(10):"+Math.exp(10) + "<br>");
document.write("floor(100):"+Math.floor(100)+"<br>");
document.write("Power(3,8):"+Math.pow(3,8) + "<br>");
document.write("Random():"+Math.random() + "<br>");
document.write("Sin(45):"+Math.sin(45) + "<br>");
document.write("Sqrt(9):"+Math.sqrt(9) + "<br>");
document.write("Tan(45):"+Math.tan(45)+"<br>");
document.write("LOG(20):"+Math.log(20) + "<br>");
</script>
</body>
</html>
```

Output

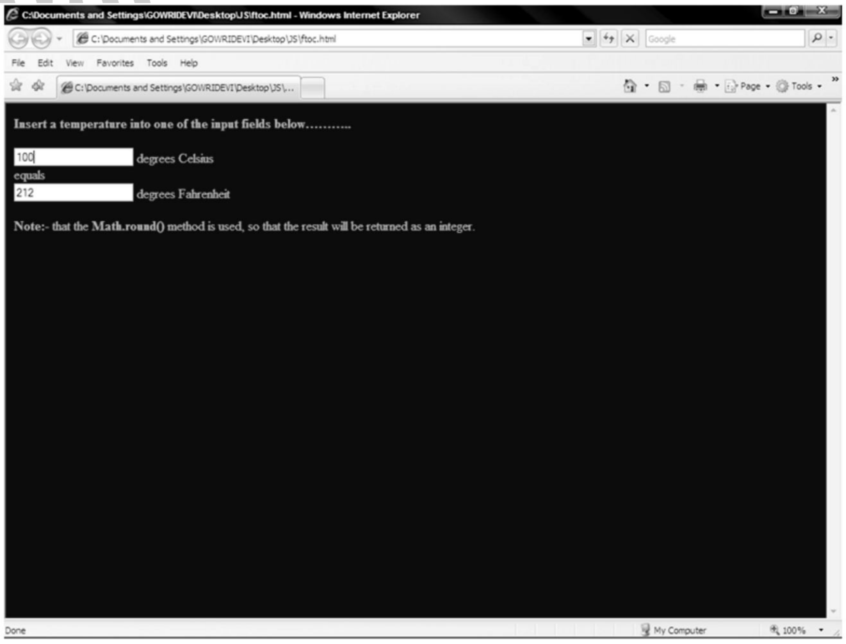

Math Functions - Windows Internet Explorer
C:\Documents and Settings\GOWRIDEVI\Desktop\JS\math.html
Google
File Edit View Favorites Tools Help
The Math object
Math Functions
Page
Tools
Done
My Computer
100%

```
Absolute(0.6):0.6
Round(0).5:1
Acos(45):NaN
Asin(0):0
Atan(0):1.5596856728972892
Maximum(5,10):10
Minimum(5,10):5
Ceil(38):38
Cos(10):-0.8390715290764524
exponential(10):22026.465794806718
floor(100):100
Power(3,8):6561
Random():0.22859199020343118
Sin(45):0.8509035245341184
Sqrt(9):3
Tan(45):1.6197751905438615
LOG(20):2.995732273553991
```

Program for converting temperature Fahrenheit to centigrade and vise versa using java script

```
<html>
<head>
<script type=”text/javascript”>
function convert(degree)
{
if (degree==”C”)
          {
 F=document.getElementById(“c”).value * 9 / 5 + 32;
 document.getElementById(“f”).value=Math.round(F);
 }
else
          {
 C=(document.getElementById(“f”).value -32) * 5 / 9;
 document.getElementById(“c”).value=Math.round(C);
 }
}
</script>
</head>
<body bgcolor=magnita text=cyan>
<b>Insert a temperature into one of the input fields below………..</b></p>
<form>
<input id=”c” name=”c” onkeyup=”convert(‘C’)”> degrees Celsius<br/>
equals<br/>
<input id=”f” name=”f” onkeyup=”convert(‘F’)”> degrees Fahrenheit
</form>
<p><b>Note:-</b> that the <b>Math.round()</b> method is used, so that the result will be returned as an integer.</p>
</body>
</html>
```

Output



```
Insert a temperature into one of the input fields below………..

100   degrees Celsius
equals
212   degrees Fahrenheit

Note:- that the Math.round() method is used, so that the result will be returned as an integer.
```

C:\Documents and Settings\GOWRIDEVI\Desktop\JS\ftoc.html - Windows Internet Explorer
File Edit View Favorites Tools Help
Page Tools
Done
My Computer
100%

3.9 JavaScript - Objects

Q17. What are objects? Explian how to create an object.

Ans : **(Imp.)**

JavaScript is an Object Oriented Programming (OOP) language. A programming language can be called object-oriented if it provides four basic capabilities to developers -

- **Encapsulation:** the capability to store related information, whether data or methods, together in an object.
- **Aggregation:** the capability to store one object inside another object.
- **Inheritance:** the capability of a class to rely upon another class (or number of classes) for some of its properties and methods.
- **Polymorphism:** the capability to write one function or method that works in a variety of different ways.

Objects are composed of attributes. If an attribute contains a function, it is considered to be a method of the object, otherwise the attribute is considered a property.

Object Properties

Object properties can be any of the three primitive data types, or any of the abstract data types, such as another object. Object properties are usually variables that are used internally in the object's methods, but can also be globally visible variables that are used throughout the page.

The syntax for adding a property to an object is –

objectName.objectProperty = propertyValue;

For example: The following code gets the document title using the "title"property of the document object.

var str = document.title;

Object Methods

Methods are the functions that let the object do something or let something be done to it. There is a small difference between a function and a method – at a function is a standalone unit of statements and a method is attached to an object and can be referenced by the this keyword.

Methods are useful for everything from displaying the contents of the object to the screen to performing complex mathematical operations on a group of local properties and parameters.

For example - Following is a simple example to show how to use the write() method of document object to write any content on the document.

document.write("This is test");

User-Defined Objects

All user-defined objects and built-in objects are descendants of an object called Object.

The new Operator

The new operator is used to create an instance of an object. To create an object, the new operator is followed by the constructor method.

In the following example, the constructor methods are Object(), Array(), and Date(). These constructors are built-in JavaScript functions.

```
var employee = new Object();

var books = new Array("C++", "Perl", "Java");

var day = new Date("August 15, 1947");
```

The Object() Constructor

A constructor is a function that creates and initializes an object. JavaScript provides a special constructor function called **Object()** to build the object. The return value of the **Object()** constructor is assigned to a variable.

The variable contains a reference to the new object. The properties assigned to the object are not variables and are not defined with the **var** keyword.

Example

```
<html>
  <head>
    <title>User-defined objects</title>
    <script type="text/javascript">
      var book = new Object();   // Create the object
      book.subject = "Perl"; // Assign properties to the object
      book.author  = "Mohtashim";
    </script>
  </head>
  <body>
    <script type="text/javascript">
      document.write("Book name is : " + book.subject + "<br>");
      document.write("Book author is : " + book.author + "<br>");
    </script>
  </body>
</html>
```

Output

```
Book name is : Perl

Book author is : Mohtashim
```

Example

This example demonstrates how to create an object with a User-Defined Function. Here **this** keyword is used to refer to the object that has been passed to a function.

```
<html>
  <head>
  <title>User-defined objects</title>
    <script type="text/javascript">
       function book(title, author){
         this.title = title;
         this.author  = author;
       }
    </script>
  </head>
  <body>
    <script type="text/javascript">
       var myBook = new book("Perl", "Mohtashim");
       document.write("Book title is : " + myBook.title + "<br>");
       document.write("Book author is : " + myBook.author + "<br>");
    </script>
  </body>
</html>
```

Output

```
Book title is : Perl
Book author is : Mohtashim
```

Defining Methods for an Object

The examples demonstrate how the constructor creates the object and assigns properties. But we need to complete the definition of an object by assigning methods to it.

Example

```
<html>
  <head>
  <title>User-defined objects</title>
```

```
    <script type="text/javascript">
      // Define a function which will work as a method
      function addPrice(amount){
        this.price = amount;
      }
      function book(title, author){
        this.title = title;
        this.author  = author;
        this.addPrice = addPrice; // Assign that method as property.
      }
    </script>

  </head>
  <body>
    <script type="text/javascript">
      var myBook = new book("Perl", "Mohtashim");
      myBook.addPrice(100);
      document.write("Book title is : " + myBook.title + "<br>");
      document.write("Book author is : " + myBook.author + "<br>");
      document.write("Book price is : " + myBook.price + "<br>");
    </script>
  </body>
</html>
```

Output

```
Book title is : Perl
Book author is : Mohtashim
Book price is : 100
```

The 'with' Keyword

The **'with'** keyword is used as a kind of shorthand for referencing an object's properties or methods.

The object specified as an argument to with becomes the default object for the duration of the block that follows. The properties and methods for the object can be used without naming the object.

Syntax

The syntax for with object is as follows -

```
with (object){
   properties used without the object name and dot
}
```

Example

Try the following example.

```
<html>
   <head>
   <title>User-defined objects</title>
      <script type="text/javascript">
         // Define a function which will work as a method
         function addPrice(amount){
            with(this){
               price = amount;
            }
         }
         function book(title, author){
            this.title = title;
            this.author = author;
            this.price = 0;
            this.addPrice = addPrice; // Assign that method as property.
         }
      </script>
   </head>
   <body>
      <script type="text/javascript">
         var myBook = new book("Perl", "Mohtashim");
         myBook.addPrice(100);
         document.write("Book title is : " + myBook.title + "<br>");
         document.write("Book author is : " + myBook.author + "<br>");
         document.write("Book price is : " + myBook.price + "<br>");
```

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

Output

```
Book title is : Perl
Book author is : Mohtashim
Book price is : 100
```

JavaScript Native Objects

JavaScript has several built-in or native objects. These objects are accessible anywhere in your program and will work the same way in any browser running in any operating system.

Here is the list of all important JavaScript Native Objects -

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

3.9.1 Math Object

Q18. Explain the functions involved in math Object.

Ans :

The **math** object provides you properties and methods for mathematical constants and functions. Unlike other global objects, Math is not a constructor. All the properties and methods of Math are static and can be called by using Math as an object without creating it.

Thus, you refer to the constant pi as Math.PI and you call the *sine* function as Math.sin(x), where x is the method's argument.

Syntax

The syntax to call the properties and methods of Math are as follows :

```
var pi_val = Math.PI;
var sine_val = Math.sin(30);
```

Math Properties

Here is a list of all the properties of Math and their description.

Sr.No	Property & Description
1	**E ** Euler's constant and the base of natural logarithms, approximately 2.718.
2	**LN2** Natural logarithm of 2, approximately 0.693.
3	**LN10** Natural logarithm of 10, approximately 2.302.
4	**LOG2E** Base 2 logarithm of E, approximately 1.442.
5	**LOG10E** Base 10 logarithm of E, approximately 0.434.
6	**PI** Ratio of the circumference of a circle to its diameter, approximately 3.14159.
7	**SQRT1_2** Square root of 1/2; equivalently, 1 over the square root of 2, approximately 0.707.
8	**SQRT2** Square root of 2, approximately 1.414.

In the following sections, we will have a few examples to demonstrate the usage of Math properties.

Math Methods

Here is a list of the methods associated with Math object and their description.

S.No	Method & Description
1	**abs()** Returns the absolute value of a number.
2	**acos()** Returns the arccosine (in radians) of a number.
3	**asin()** Returns the arcsine (in radians) of a number.
4	**atan()** Returns the arctangent (in radians) of a number.
5	**atan2()** Returns the arctangent of the quotient of its arguments.
6	**ceil()** Returns the smallest integer greater than or equal to a number.
7	**cos()** Returns the cosine of a number.
8	**exp()** Returns E^N, where N is the argument, and E is Euler's constant, the base of the natural logarithm.
9	**floor()** Returns the largest integer less than or equal to a number.

10	**log()** Returns the natural logarithm (base E) of a number.
11	**max()** Returns the largest of zero or more numbers.
12	**min()** Returns the smallest of zero or more numbers.
13	**pow()** Returns base to the exponent power, that is, base exponent.
14	**random()** Returns a pseudo-random number between 0 and 1.
15	**round()** Returns the value of a number rounded to the nearest integer.
16	**sin()** Returns the sine of a number.
17	**sqrt()** Returns the square root of a number.
18	**tan()** Returns the tangent of a number.
19	**toSource()** Returns the string "Math".

3.9.2 Strings Object

Q19. Discuss in detail about string Object ?

Ans :

The String object lets you work with a series of characters; it wraps Javascript's string primitive data type with a number of helper methods.

As JavaScript automatically converts between string primitives and String objects, you can call any of the helper methods of the String object on a string primitive.

Syntax : Use the following syntax to create a String object -

var val = new String(string);

The String parameter is a series of characters that has been properly encoded.

String Properties

Here is a list of the properties of String object and their description.

Sr.No	Property & Description
1	**constructor** Returns a reference to the String function that created the object.
2	**length** Returns the length of the string.
3	**prototype** The prototype property allows you to add properties and methods to an object.

In the following sections, we will have a few examples to demonstrate the usage of String properties.

String Methods

Here is a list of the methods available in String object along with their description.

Sr.No	Method & Description
1	**charAt()** Returns the character at the specified index.
2	**charCodeAt()** Returns a number indicating the Unicode value of the character at the given index.
3	**concat()** Combines the text of two strings and returns a new string.
4	**indexOf()** Returns the index within the calling String object of the first occurrence of the specified value, or -1 if not found.
5	**lastIndexOf()** Returns the index within the calling String object of the last occurrence of the specified value, or -1 if not found.
6	**localeCompare()** Returns a number indicating whether a reference string comes before or after or is the same as the given string in sort order.
7	**match()** Used to match a regular expression against a string.
8	**replace()** Used to find a match between a regular expression and a string, and to replace the matched substring with a new substring.
9	**search()** Executes the search for a match between a regular expression and a specified string.
10	**slice()** Extracts a section of a string and returns a new string.
11	**split()** Splits a String object into an array of strings by separating the string into substrings.
12	**substr()** Returns the characters in a string beginning at the specified location through the specified number of characters.
13	**substring()** Returns the characters in a string between two indexes into the string.
14	**toLocaleLowerCase()** The characters within a string are converted to lower case while respecting the current locale.
15	**toLocaleUpperCase()** The characters within a string are converted to upper case while respecting the current locale.
16	**toLowerCase()** Returns the calling string value converted to lower case.
17	**toString()** Returns a string representing the specified object.
18	**toUpperCase()** Returns the calling string value converted to uppercase.
19	**valueOf()** Returns the primitive value of the specified object.

String HTML Wrappers

Here is a list of the methods that return a copy of the string wrapped inside an appropriate HTML tag.

S.No	Method & Description
1	**anchor()** Creates an HTML anchor that is used as a hypertext target.
2	**big()** Creates a string to be displayed in a big font as if it were in a <big> tag.
3	**blink()** Creates a string to blink as if it were in a <blink> tag.
4	**bold()** Creates a string to be displayed as bold as if it were in a <b> tag.
5	**fixed()** Causes a string to be displayed in fixed-pitch font as if it were in a <tt> tag
6	**fontcolor()** Causes a string to be displayed in the specified color as if it were in a <font color="color"> tag.
7	**fontsize()** Causes a string to be displayed in the specified font size as if it were in a <font size="size"> tag.
8	**italics()** Causes a string to be italic, as if it were in an <i> tag.
9	**link()** Creates an HTML hypertext link that requests another URL.
10	**small()** Causes a string to be displayed in a small font, as if it were in a <small> tag.
11	**strike()** Causes a string to be displayed as struck-out text, as if it were in a <strike> tag.
12	**sub()** Causes a string to be displayed as a subscript, as if it were in a <sub> tag
13	**sup()** Causes a string to be displayed as a superscript, as if it were in a <sup> tag

3.9.3 Number Object

Q20. What are the methods involved in Number Object

Ans :

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

Syntax

The syntax for creating a **number** object is as follows "

var val =newNumber(number);

In the place of number, if you provide any non-number argument, then the argument cannot be converted into a number, it returns NaN (Not-a-Number).

Number Properties

Here is a list of each property and their description.

Sr.No	Property & Description
1	**MAX_VALUE** The largest possible value a number in JavaScript can have 1.7976931348623157E+308
2	**MIN_VALUE** The smallest possible value a number in JavaScript can have 5E-324
3	**NaN** Equal to a value that is not a number.
4	**NEGATIVE_INFINITY** A value that is less than MIN_VALUE.
5	**POSITIVE_INFINITY** A value that is greater than MAX_VALUE
6	**prototype** 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
7	**constructor** Returns the function that created this object's instance. By default this is the Number object.

In the following sections, we will take a few examples to demonstrate the properties of Number.

Number Methods

The Number object contains only the default methods that are a part of every object's definition.

Sr.No	Method & Description
1	**toExponential()** Forces a number to display in exponential notation, even if the number is in the range in which JavaScript normally uses standard notation.
2	**toFixed()** Formats a number with a specific number of digits to the right of the decimal.
3	**toLocaleString()** Returns a string value version of the current number in a format that may vary according to a browser's local settings.
4	**toPrecision()** Defines how many total digits (including digits to the left and right of the decimal) to display of a number.
5	**toString()** Returns the string representation of the number's value.
6	**valueOf()** Returns the number's value.

3.9.4 Boolean Object

Q21. Explain the representation of Boolean Object?

Ans :

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

Syntax

Use the following syntax to create a boolean object.

var val =newBoolean(value);

Boolean Properties

Here is a list of the properties of Boolean object –

Sr.No	Property & Description
1	**constructor** Returns a reference to the Boolean function that created the object.
2	**prototype** The prototype property allows you to add properties and methods to an object.

In the following sections, we will have a few examples to illustrate the properties of Boolean object.

Boolean Methods

Here is a list of the methods of Boolean object and their description.

Sr.No	Method & Description
1	**toSource()** Returns a string containing the source of the Boolean object; you can use this string to create an equivalent object.
2	**toString()** Returns a string of either "true" or "false" depending upon the value of the object.
3	**valueOf()** Returns the primitive value of the Boolean object.

UNIT IV

Client side scripting with VB Script, operations, Data types and control structures, Functions, Arrays, String manipulations, classes and objects.

Web Servers : Personal Web server, Internet information server, Apache Web Server, Installation of a Web Server.

4.1 Client Side Scripting with VB Script

Q1. Explain the concept of VBScript ?

Ans : **(Imp.)**

- VBScript is a client-side scripting language like JavaScript.
- VB Script stands for Visual Basic Script and it is a light version of Microsoft Visual Basic.
- The syntax of VBScript is very similar to that of Visual Basic.
- VBScript was developed by Microsoft with the intention of developing dynamic web pages.

In other words, if you want your webpage to be more lively and interactive, then you can incorporate VBScript in your code.

VBScript is just a scripting language.

So, it cannot run its code on its own.

It needs a bigger programming language to host it.

Right now, there are 3 environments where VB Script can run.

1. **IIS (Internet Information Server):** Microsoft's web server
2. **WSH (Windows Script Host)**: The native hosting environment of the Windows OS
3. **IE (Internet Explorer):** The simplest hosting environment we can use to run VBScripts.

Disadvantage of VBScript

The main disadvantage of VBScript is that most browsers except Internet Explorer will not process VBScript code.

In other words, if your site has visitors who use a web browser other than Internet Explorer like Chrome, Firefox or Opera, then VBScript will not be useful.

Moreover, VBScript will not run on computers that run on operating systems other than Microsoft Windows including Linux, Mac etc.

Like any other scripting language, VBScript has gone through many changes over the years.

Now, VB Script is used as the default scripting language of ASP.

How to Create a Simple VBScript ?

You only need 2 simple tools to create and run VBScript code throughout this tutorial:

1. **Internet Explorer:** any version, but it is good to use IE6 or above.
2. **Text Editor:** You can use normal text editors like Notepad++ or Microsoft Expression Web or even Notepad to write VBScript code.

Let's start by developing a simple VB Script program.

We will embed our VBScript code within a very basic HTML code.

This way, we can see VBScript in action by running the particular HTML file on Internet Explorer web browser.

```
<html>
<head>
<title>My First VBScript Code!!!</title>
</head>
<body>
<script type="text/vbscript">
document.write("Yes!!! I have started learning VBScript.")
</script>
</body>
</html>
```

VBScript stands for Visual Basic Scripting that forms a subset of Visual Basic for Applications (VBA).

VBA is a product of Microsoft which is included NOT only in other Microsoft products such as MS Project and MS Office but also in Third Party tools such as AUTO CAD.

Features of VBScript

- VBScript is a lightweight scripting language, which has a lightning fast interpreter.
- VBScript, for the most part, is case insensitive. It has a very simple syntax, easy to learn and to implement.
- Unlike C++ or Java, VBScript is an object-based scripting language and NOT an Object-Oriented Programming language.
- It uses Component Object Model (COM) in order to access the elements of the environment in which it is executing.
- Successful execution of VBScript can happen only if it is executed in Host Environment such as Internet Explorer (IE), Internet Information Services (IIS) and Windows Scripting Host (WSH).

VBscript – Version History and Uses

VBScript was introduced by Microsoft way back in 1996 and the first version was 1.0. The Current Stable version of VBScript is 5.8, which is available as part of IE8 or Windows 7.

The VBScript usage areas are aplenty and not restricted to the below list.

- VBScript is used as a scripting language in one of the popular Automation testing tools – Quick Test Professional abbreviated as QTP.
- Windows Scripting Host, which is used mostly by Windows System administrators for automating the Windows Desktop.
- Active Server Pages (ASP), a server side scripting environment for creating dynamic webpages which uses VBScript or Java Script.
- VBScript is used for Client side scripting in Microsoft Internet Explorer.
- Microsoft Outlook Forms usually runs on VBScript; however, the application level programming relies on VBA (Outlook 2000 onwards).

Disadvantages

- VBscript is used only by IE Browsers. Other browsers such as Chrome, Firefox DONOT Support VBScript. Hence, JavaScript is preferred over VBScript.
- VBScript has a Limited command line support.
- Since there is no development environment available by default, debugging is difficult.

Where VBScript is Today ?

The current version of VBScript is 5.8, and with the recent development of .NET framework, Microsoft has decided to provide future support of VBScript within ASP.NET for web development.

Hence, there will NOT be any more new versions of VBScript engine but the entire defect fixes and security issues are being addressed by the Microsoft sustaining Engineering Team.

However, VBScript engine would be shipped as part of all Microsoft Windows and IIS by default.

Enabling VBScript in Browsers

NOT All the modern browsers support VBScript. VBScript is supported just by Microsoft's Internet Explorer while other browsers(Firefox and Chrome) just support JavaScript. Hence, the developers prefer JavaScript over VBScript.

Though Internet Explorer (IE) supports VBScript, many a times you may need to enable or disable this feature manually. This tutorial will make you aware of the procedure of enabling and disabling VBScript support in Internet Explorer.

VBScript in Internet Explorer

Here are simple steps to turn on or turn off VBScript in your Internet Explorer :

- Follow Tools-> Internet Options from the menu
- Select Security tab from the dialog box
- Click the Custom Level button
- Scroll down till you find Scripting option
- Select Enable radio button under Active scripting
- Finally click OK and come out

To disable VBScript support in your Internet Explorer, you need to select Disable radio button under Active scripting.

Q2. Write briefly about VB Script variables?

Ans :

A variable is a fundamental part of any programming language. Variables are best visualized as a container. This is because a variable's purpose is to hold something - a value.

How does the variable get the value? You, as a programmer, assign the value to it. This value could be dynamically produced, depending on your program/script.

Once the variable has a value, you can use subsequent code to check its value, and do something (or not do something) depending on its value.

Some programming languages require that you declare your variable before assigning a value to it. Others (such as VBScript) make this optional. In any case, it is good practice to declare all your variables first.

Declare a VBScript Variable

VBScript variables are declared using the dim statement :

```
<script type="text/vbscript">
    Dim firstName
    Dim age
</script>
```

These variables could also have been declared on the same line, separated by a comma:

```
<script type="text/vbscript">
    Dim firstName, age
</script>
```

Option Explicit

As mentioned, it's good practice to declare all variables before using them. Occasionally, if you're in a hurry, you might forget to do this. Then, one day, you might misspell the variable name and all sorts of problems could start occurring with your script.

VBScript has a way of ensuring you declare all your variables. This is done with the Option Explicit statement :

```
<script type="text/vbscript">
    Option Explicit
    Dim firstName
    Dim age
</script>
```

Assign Values to your Variables

You assign values using an equals operator (equals sign). The variable name goes on the left, the value on the right.

```
<script type="text/vbscript">
    Option Explicit
    Dim firstName
    Dim age
    firstName = "Borat"
    age = 25
</script>
```

Display your Variables

You can output the value of a variable by including it within the document.write()method.

```
<script type="text/vbscript">
    Option Explicit
    Dim firstName
    Dim age
    firstName = "Borat"
    age = 25
    document.write("Firstname: " & firstName & "</br />")
    document.write("Age: " & age)
</script>
```

4.2 Operators

Q3. Discuss different types of Operators in VB Script ?

Ans :

Simple answer can be given using expression 4 + 5 is equal to 9. Here, 4 and 5 are called operands and + is called operator. VBScript language supports following types of operators:

i) Arithmetic Operators

ii) Comparison Operators

iii) Logical (or Relational) Operators

iv) Concatenation Operators

i) Arithmetic Operators

There are following arithmetic operators supported by VBScript language :

Assume variable A holds 5 and variable B holds 10, then :

Operator	Description	Example
+	Adds two operands	A + B will give 15
–	Subtracts second operand from the first	A - B will give -5
*	Multiply both operands	A * B will give 50
/	Divide numerator by denumerator	B / A will give 2
%	Modulus Operator and remainder of after an integer division	B MOD A will give 0
^	Exponentiation Operator	B ^ A will give 100000

ii) Comparison Operators

There are following comparison operators supported by VBScript language:

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

Operator	Description	Example
= =	Checks if the value of two operands are equal or not, if yes then condition becomes true.	(A = = B) is False
< >	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 False
<	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 False
< =	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

iii) Logical Operators

There are following logical operators supported by VBScript language:

Assume variable A holds 10 and variable B holds 0, then:

Operator	Description	Example
AND	Called Logical AND operator. If both the conditions are True then Expression becomes true.	a< >0 AND b< >0 is False.
OR	Called Logical OR Operator. If any of the two conditions are True then condition becomes true.	a< >0 OR b< >0 is true.
NOT	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.	NOT(a< >0 OR b< >0) is false.
XOR	Called Logical Exclusion. It is the combination of NOT and OR Operator. If one, and only one, of the expressions evaluates to True, result is True.	(a< >0 XOR b< >0) is true.

iv) Concatenation Operators

There are following Concatenation operators supported by VBScript language:

Assume variable A holds 5 and variable B holds 10 then:

Operator	Description	Example
+	Adds two Values as Variable Values are Numeric	A + B will give 15
&	Concatenates two Values	A & B will give 510

Assume variable A= “Microsoft” and variable B=”VBScript”, then:

Operator	Description	Example
+	Concatenates two Values	A + B will give MicrosoftVBScript
&	Concatenates two Values	A & B will give MicrosoftVBScript

Note: Concatenation Operators can be used for numbers and strings. The Output depends on the context if the variables hold numeric value or String Value.

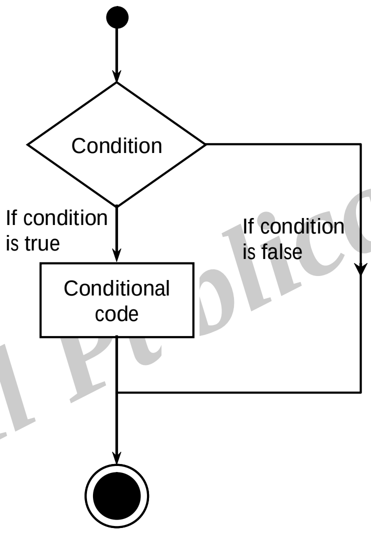
4.3 Control Structures

Q4. What are decision making statements in VB Script ?

Ans :

Decision making allows programmers to control the execution flow of a script or one of its sections. The execution is governed by one or more conditional statements.

Following is the general form of a typical decision making structure found in most of the programming languages :



VBScript provides following types of decision making statements. Click the following links to check their details.

Statement	Description
if statement	An if statement consists of a boolean expression followed by one or more statements.
if..else statement	An if else statement consists of a boolean expression followed by one or more statements. If the condition is True, the statements under Ifstatements are executed. If the condition is false, Else part of the script is Executed
if...elseif..else statement	An if statement followed by one or more ElseIfStatements, that consists of boolean expressions and then followed by an optional else statement, which executes when all the condition becomes false.
nested if statements	An if or elseif statement inside another if or elseifstatement(s).
switch statement	A switch statement allows a variable to be tested for equality against a list of values.

i) If Statements

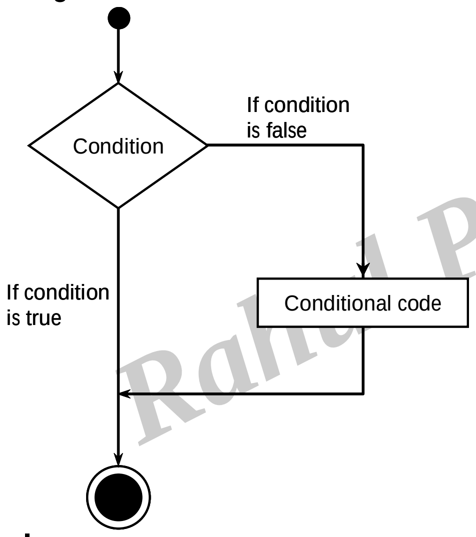
An If statement consists of a boolean expression followed by one or more statements. If the condition is said to be True, the statements under Ifcondition(s) are Executed. If the Condition is said to be False, the statements after the If loop are executed.

Syntax

The syntax of an If statement in VBScript is:

```
If(boolean_expression)Then
Statement1
      .....
      .....
Statement n
EndIf
```

Flow Diagram



Example

```
<!DOCTYPE html>
<html>
<body>
<scriptlanguage="vbscript"type="text/vbscript">
Dim a : a =20
Dim b : b =10
If a > b Then
Document.write "a is Greater than b"
EndIf

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

ii) If Else Statements

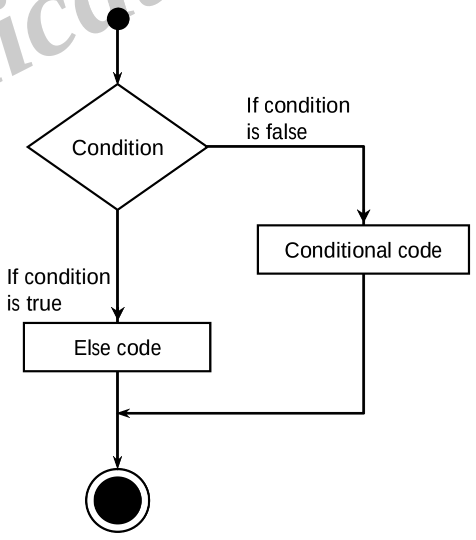
An If statement consists of a boolean expression followed by one or more statements. If the condition is said to be True, the statements under Ifcondition(s) are Executed. If the Condition is said to be False, the statements under Else Part would be executed.

Syntax

The syntax of an if statement in VBScript is:

```
If(boolean_expression)Then
Statement1
      .....
      .....
Statement n
Else
Statement1
      .....
      ....
Statement n
EndIf
```

Flow Diagram



Example

```
<!DOCTYPE html>
<html>
<body>
<scriptlanguage="vbscript"type="text/vbscript">
Dim a : a =5
Dim b : b =25
If a > b Then
```

```
Document.write “a is Greater”
Else
Document.write “b is Greater”
EndIf
</script>
</body>
</html>
```

iii) Switch Statements in VBScript

When a User want to execute a group of statements depending upon a value of an Expression, then Switch Case is used. Each value is called a Case, and the variable being switched ON based on each case. Case Else statement is executed if test expression doesn’t match any of the Case specified by the user.

Case Else is an optional statement within Select Case, however, it is a good programming practice to always have a Case Else statement.

Syntax

The syntax of a Switch Statement in VBScript is :

```
SelectCase expression
Case expressionlist1
    statement1
    statement2
      ....
      ....
    statement1n
Case expressionlist2
    statement1
    statement2
      ....
      ....
Case expressionlistn
    statement1
    statement2
      ....
      ....
CaseElse
    elsestatement1
    elsestatement2
      ....
      ....
EndSelect
```

Example

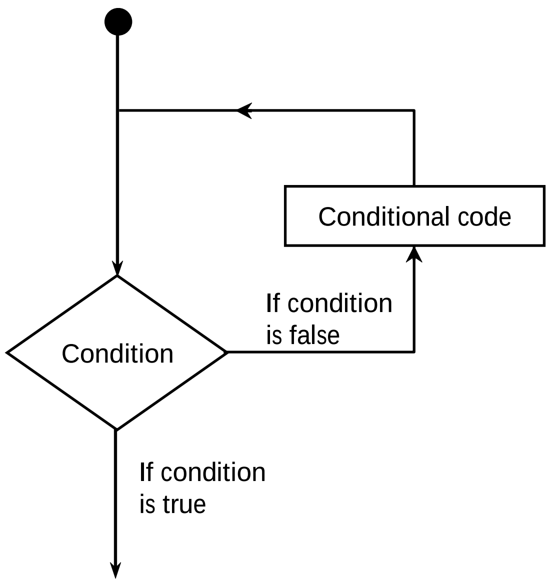
```
<!DOCTYPE html>
<html>
<body>
<script language=”vbscript”type=”text/vbscript”>
Dim MyVar
MyVar=1
Select caseMyVar
case 1
        Document.write “The Number is the Least Composite Number”
case 2
        Document.write “The Number is the only Even Prime Number”
case 3
        Document.write “The Number is the Least Odd Prime Number”
case else
        Document.write “Unknown Number”
End select
</script>
</body>
</html>
```

Q5. Discuss briefly about VB Script Loop Control Statements ?

Ans : **(Imp.)**

There may be a situation when you need to execute a block of code several number of times. In general, statements are executed sequentially: The first statement in a function is executed first, followed by the second, and so on. Programming languages provide various control structures that allow for more complicated execution paths.

A loop statement allows us to execute a statement or group of statements multiple times and following is the general from of a loop statement in VBScript.



VBScript provides the following types of loops to handle looping requirements. Click the following links to check their detail.

Loop Type	Description
for loop	Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.
for ..each loop	This is executed if there is at least one element in group and reiterated for each element in a group.
while..wend loop	This tests the condition before executing the loop body.
do..while loops	The do..While statements will be executed as long as condition is True.(i.e.,) The Loop should be repeated till the condition is False.
do..until loops	The do..Until statements will be executed as long as condition is False.(i.e.,) The Loop should be repeated till the condition is True.

Loop Control Statements

Loop control statements change execution from its normal sequence. When execution leaves a scope, all the remaining statements in the loop are NOT executed.

VBScript supports the following control statements. Click the following links to check their detail.

Control Statement	Description
Exit For statement	Terminates the For loop statement and transfers execution to the statement immediately following the loop
Exit Do statement	Terminates the Do While statement and transfers execution to the statement immediately following the loop

i) For Loops

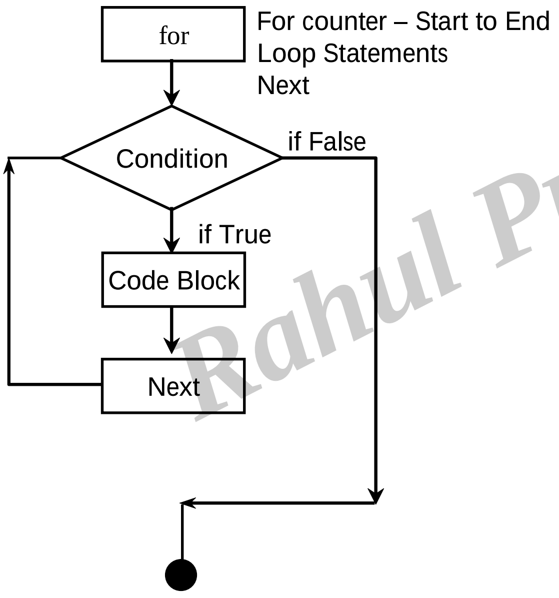
A for loop is a repetition control structure that allows a developer to efficiently write a loop that needs to execute a specific number of times.

Syntax

The syntax of a for loop in VBScript is:

```
For counter = start Toend[Step stepcount]
[statement 1]
[statement 2]
....
[statement n]
[ExitFor]
[statement 11]
[statement 22]
....
[statement n]
Next
```

Flow Diagram



Here is the flow of control in a For Loop:

- The For step is executed first. This step allows you to initialize any loop control variables and increment the step counter variable.
- Secondly, the condition is evaluated. If it is true, the body of the loop is executed. If it is false, the body of the loop does not execute and flow of control jumps to the next statement just after the For Loop.
- After the body of the for loop executes, the flow of control jumps to the Next statement. This statement allows you to update any loop control variables. It is updated based on the step counter value.
- The condition is now evaluated again. If it is true, the loop executes and the process repeats itself (body of loop, then increment step, and then again condition). After the condition becomes false, the For Loop terminates.

Example

```
<!DOCTYPE html>
<html>
<body>
<scriptlanguage="vbscript"type="text/vbscript">
Dim a : a=10
        For i=0 to a Step2'i is the counter variable
and it is incremented by 2
        document.write("The value is i is : "& i)
        document.write("<br></br>")
Next
        </script>
        </body>
        </html>
```

When the above code is compiled and executed, it produces the following result:

```
The value is i is:0
The value is i is:2
The value is i is:4
The value is i is:6
The value is i is:8
The value is i is:10
```

ii) While...Wend Loop

In a While..Wend loop, if the condition is True, all statements are executed until Wend keyword is encountered.

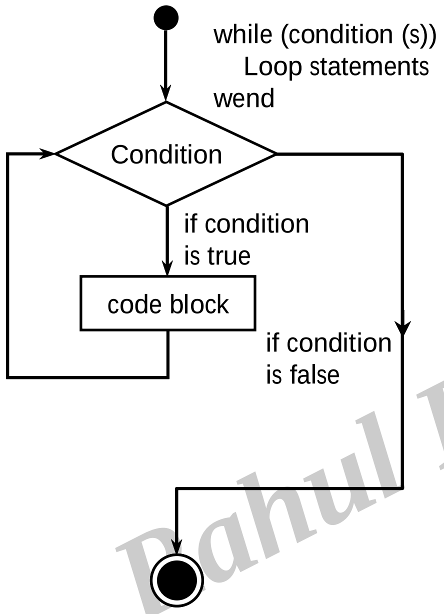
If the condition is false, the loop is exited and the control jumps to very next statement after Wend keyword.

Syntax

The syntax of a While..Wend loop in VBScript is:

```
While condition(s)
        [statements 1]
        [statements 2]
        ...
        [statements n]
        Wend
```

Flow Diagram



Example

```
<!DOCTYPE html>
<html>
<body>
<scriptlanguage="vbscript"type="text/vbscript">
DimCounter:Counter=10
WhileCounter<15'Test value of Counter.
Counter=Counter+1'IncrementCounter.
   document.write("The Current Value of the Counter is : "&Counter)
        document.write("<br></br>")
Wend'While loop exits ifCounterValue becomes 15.
</script>
</body>
</html>
```

When the above code is executed, it prints the following output in the console.

```
TheCurrentValue of the Counteris:11
TheCurrentValue of the Counteris:12
TheCurrentValue of the Counteris:13
TheCurrentValue of the Counteris:14
TheCurrentValue of the Counteris:15
```

iii) Do..While statement

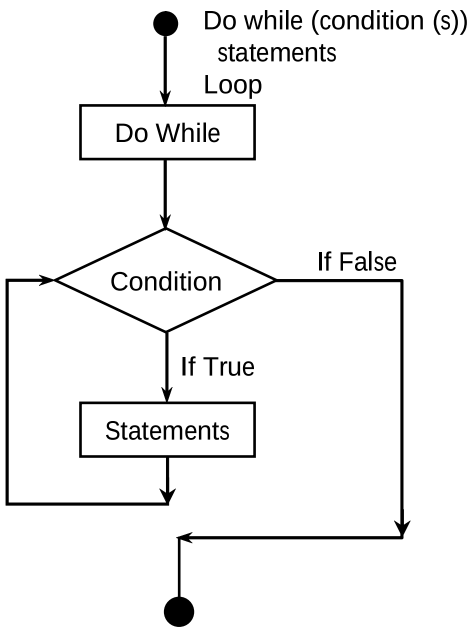
A Do..While loop is used when we want to repeat a set of statements as long as the condition is true. The Condition may be checked at the beginning of the loop or at the end of the loop.

Syntax

The syntax of a Do..While loop in VBScript is:

```
DoWhile condition
[statement 1]
[statement 2]
...
[statement n]
[ExitDo]
[statement 1]
[statement 2]
...
[statement n]
Loop
```

Flow Diagram



Example

The below example uses Do..while loop to check the condition at the beginning of the loop. The statements inside the loop are executed only if the condition becomes True.

```
<!DOCTYPE html>
<html>
<body>
<script language="vbscript" type="text/vbscript">
DoWhile i <5
   i = i +1
Document.write("The value of i is : "& i)
Document.write("<br></br>")
Loop
</script>
</body>
</html>
```

When the above code is executed, it prints the following output in the console.

```
The value of i is:1
The value of i is:2
The value of i is:3
The value of i is:4
The value of i is:5
```

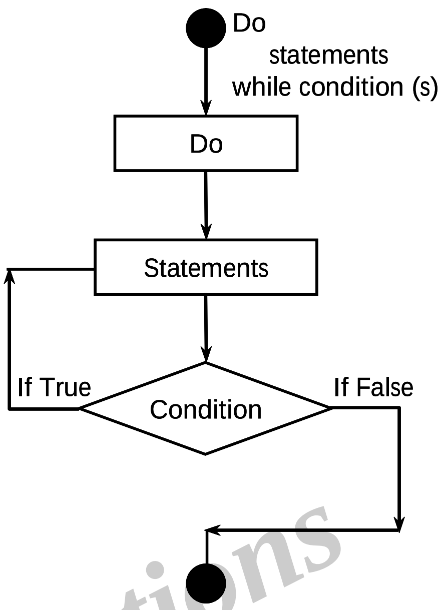
Alternate Syntax

There is also an alternate Syntax for Do..while loop which checks the condition at the end of the loop. The Major difference between these two syntax is explained below with an example.

```
Do
[statement 1]
[statement 2]
...
[statement n]
[ExitDo]
[statement 1]
[statement 2]
...
[statement n]
```

iv) LoopWhile condition

Flow Diagram



Example

The below example uses Do..while loop to check the condition at the end of the loop. The Statements inside the loop are executed atleast once even if the condition is False.

```
<!DOCTYPE html>
<html>
<body>
<scriptl anguage="vbscript" type="text/vbscript">
 i=10
Do
   i = i +1
Document.write("The value of i is : "& i)
Document.write("<br></br>")
Loop While i<3'Condition is false.Hence loop is executed once.
</script>
</body>
</html>
```

When the above code is executed, it prints the following output in the console.

```
The value of i is:11
```

Q6. Write about Do-Until Loop in VB Script?

Ans :

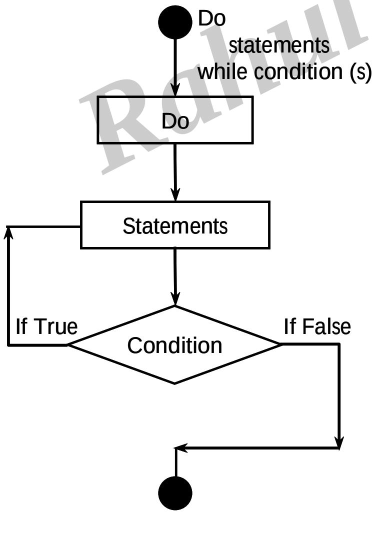
A Do..Until loop is used when we want to repeat a set of statements as long as the condition is false. The Condition may be checked at the beginning of the loop or at the end of loop.

Syntax

The syntax of a Do..Until loop in VBScript is:

```
DoUntil condition
[statement 1]
[statement 2]
...
[statement n]
[ExitDo]
[statement 1]
[statement 2]
...
[statement n]
Loop
```

Flow Diagram



Example

The below example uses Do..Until loop to check the condition at the beginning of the loop. The Statements inside the loop are executed only if the condition is false. It exits out of the loop when the condition becomes true.

```
<!DOCTYPE html>
<html>
<body>
<scriptlanguage="vbscript"type="text/vbscript">
  i=10
DoUntil i>15'Condition is False.Hence loop will be executed
  i = i +1
Document.write("The value of i is : "& i)
Document.write("<br></br>")
Loop
</script>
</body>
</html>
```

When the above code is executed, it prints the following output in the console.

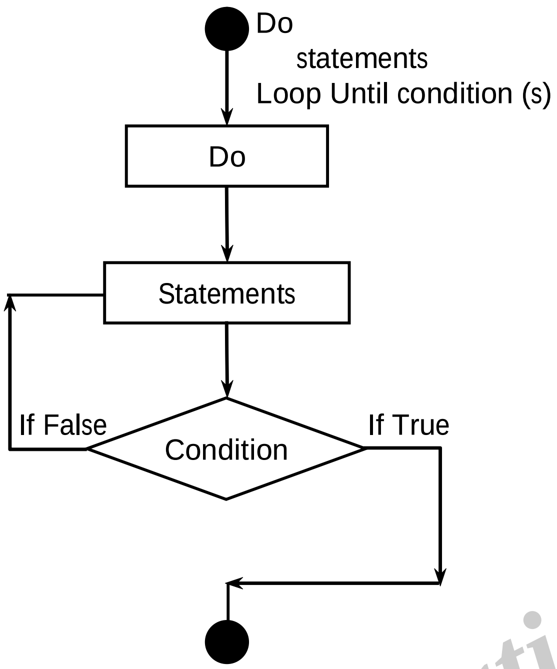
```
The value of i is:11
The value of i is:12
The value of i is:13
The value of i is:14
The value of i is:15
The value of i is:16
```

Alternate Syntax

There is also an alternate Syntax for Do..Until loop which checks the condition at the end of the loop. The Major difference between these two syntax is explained below with an example.

```
Do
[statement 1]
[statement 2]
...
[statement n]
[ExitDo]
[statement 1]
[statement 2]
...
[statement n]
LoopUntil condition
```

Flow Diagram



Example

The below example uses Do..Until loop to check the condition at the end of the loop. The Statements inside the loop are executed atleast once even if the condition is True.

```
<!DOCTYPE html>

<html>

<body>

<script language=”vbscript” type=”text/vbscript”>

 i=10

Do

   i = i +1

Document.write(“The value of i is : “& i)

Document.write(“<br></br>”)

Loop Until i<15'Condition is True.Hence loop is executed once.

</script>

</body>

</html>
```

When the above code is executed, it prints the following output in the console.

```
The value of i is:11
```

4.4 Strings

Q7. Write about VB Scripts Strings and Related Functions ?

Ans : **(Imp.)**

Strings are a sequence of characters, which can consist of alphabets or numbers or special characters or all of them. A variable is said to be a string if it is enclosed within double quotes "".

Syntax

variablename = "string"

Examples

str1 = "string"' Only Alphabets

str2 = "132.45" 'OnlyNumbers

str3 = "!@#$;*"' Only Special Characters

Str4 = "Asc23@#" 'Has all the above

String Functions

There are predefined VBScript String functions, which help the developers to work with the strings very effectively. Below are String methods that are supported in VBScript. Please click on each one of the methods to know in detail.

Function Name	Description
InStr	Returns the first occurrence of the specified substring. Search happens from left to right.
InstrRev	Returns the first occurrence of the specified substring. Search happens from Right to Left.
Lcase	Returns the lower case of the specified string.
Ucase	Returns the Upper case of the specified string.
Left	Returns a specific number of characters from the left side of the string.
Right	Returns a specific number of characters from the Right side of the string.
Mid	Returns a specific number of characters from a string based on the specified parameters.
Ltrim	Returns a string after removing the spaces on the left side of the specified string.
Rtrim	Returns a string after removing the spaces on the right side of the specified string.
Trim	Returns a string value after removing both leading and trailing blank spaces.
Len	Returns the length of the given string.
Replace	Returns a string after replacing a string with another string.
Space	Fills a string with the specified number of spaces.
StrComp	Returns an integer value after comparing the two specified strings.
String	Returns a String with a specified character the specified number of times.
StrReverse	Returns a String after reversing the sequence of the characters of the given string.

i) **InStr :** VBS InStr is used to find the position value of a substring at its first occurrence inside the main string. This function requires 2 strings to be specified to perform this search operation and the search operation starts right from the first character.

The syntax of this function is: InStr(name of string1, name of string2)

If the name of string1 or string2 is null or “” then this function will return null and 0 respectively. In case, if the string is not found then the value of this function will be >=1 and 0.

ii) **InStrRev :** InStrRev is just the reverse of the above function. This is also used to find the position value of a substring at its first occurrence inside the main string. This function requires 2 strings to be specified to perform this search operation but with a minor difference that the search operation starts from the last character and even the position count is starts from the beginning character only.

The syntax of this function is: InStrRev (name of string1, name of string2)

If the name of string1 or string2 is null or “” then this function will return null and 0 respectively. In case, if the string is not found then the value of this function will be >=1 and 0.

iii) **LCase:** LCase is used to convert the specified string into a lower case.

The syntax of this is: LCase(name of the string)

iv) **UCase:**UCase is used to convert the specified string into an upper case.

The syntax of this is: UCase(name of the string)

v) **Left:**Left is used to fetch/get the mentioned number of characters(as per length parameter) from the left-hand side of the specified String.

The syntax of this is: Left(name of the string, length)

vi) **Len:**Len is used to get the length of a specified String i.e. the total number of characters of a specified String.

The syntax of this is: Len(name of the string)

vii) **StrReverse:**StrReverse is used to reverse the specified string i.e. this will return the characters of a specified string in a reverse order starting from end to the beginning.

The syntax of this is: StrReverse(name of the string)

viii) **LTrim:**LTrim is used to trim/remove the spaces from the left-hand side of the specified String.

The syntax of this is: LTrim(name of the string)

ix) **Trim:**Trim is used to trim/remove the spaces from both the sides of the specified String.

The syntax of this is: Trim(name of the string)

x) **Right:**Right is used to fetch/get the mentioned number of characters(as per length parameter) from the right-hand side of the specified String.

The syntax of this is: Right(name of the string, length)

xi) **RTrim:**RTrim is used to trim/remove the spaces from the right-hand side of the specified String.

xii) **The syntax of this is:** RTrim(name of the string)

xiii) **Mid:**Mid is used to fetch the mentioned number of characters from the string by specifying the starting position.

The syntax of this is: Mid(name of the string, starting position)

xiv) **Space:**Space is used to fetch the String containing the required number of spaces as specified inside the parenthesis.

The syntax of this is: Space(number of spaces)

xv) Replace: Replace is used to replace the specified portion of a string with some other text as specified.

xvi) The syntax of this is: Replace(name of the string, name of the string to be replaced, name of the new replaced string)

xvii) StrComp: StrComp is used to compare the 2 strings and return values on the basis of comparison. This returns 0 if string1 = string2,-1 if string1<string2,1 if string1> string2 and null if any of the strings is null.

4.5 Functions

Q8. What are functions in VB Script ?

Ans : **(Imp.)**

VBScript functions are self contained blocks of code that perform a given "function", then return a value.

Sometimes when coding, you may find yourself having to write the same piece of code over and over again. If this happens, chances are, you'd be better off writing a function. That way, you can write the code once, then simply call the function whenever you need it.

Creating VBScript Functions

VBScript functions are defined using the Function and End Function keywords. The code that makes up the function resides between these two keywords.

Here's the method for creating VBScript functions:

1. Start by using the Function keyword. This tells the browser that a function is about to be defined.
2. Provide a name for the function (make it concise, but descriptive)
3. Follow the name with opening and closing brackets
4. Add the names of any arguments that the function requires (optional)
5. Starting on a new line, write the code that makes up the function
6. Finish with End Function

Example

```
Function sum(number1,number2)
    sum = number1 + number2
End Function
```

Calling VBScript Functions

A function doesn't actually do anything until it is called. Once it is called, it takes any arguments, then performs it's function (whatever that may be).

You call a VBScript function like this:

1. Write the function name, followed by opening and closing brackets
2. Between the brackets, provide all arguments that the function requires

Example

```
sum(100,9)
```

Combined

Creating, then Calling a VBScript FunctionSo, if we combine them together, and add a tiny bit more code, we might end up with something that looks like this:

```
    <script type="text/vbscript">
Function sum(number1,number2)
    sum = number1 + number2
End Function
Dim total
total = sum(100,9)
document.write(total)
</script>
```

4.6 Arrays

Q9. What is an Array ? Explain in detail in VB Script ? Accessing methods?

Ans : **(Imp.)**

We know very well that a variable is a container to store a value. Sometimes, developers are in a position to hold more than one value in a single variable at a time. When a series of values are stored in a single variable, then it is known as array variable.

Array Declaration

Arrays are declared the same way a variable has been declared except that the declaration of an array variable uses parenthesis. In the below example, the size of the array is mentioned in the brackets.

'Method 1 : Using Dim

Dim arr1() 'WithoutSize

'Method 2 : Mentioning the Size

Dim arr2(5) 'Declaredwith size of 5

'Method 3 : using 'Array' Parameter

Dim arr3

arr3 = Array("apple","Orange","Grapes")

- Although, the Array size is indicated as 5, it can hold 6 values as array index starts from ZERO.
- Array Index Cannot be Negative.
- VBScript Arrays can store any type of variable in an array. Hence, an array can store an integer, string or characters in a single array variable.

Assigning Values to an Array

The values are assigned to the array by specifying array index value against each one of the values to be assigned. It can be a string.

Example

```
<!DOCTYPE html>
<html>
<body>
<script language="vbscript" type="text/vbscript">
Dim arr(5)
arr(0)="1"'Number as String
arr(1)="VBScript"'String
arr(2)=100 'Number
arr(3)=2.45 'DecimalNumber
arr(4)=#10/07/2013#'Date
arr(5)=#12.45 PM#'Time
     document.write("Value stored in Array index 0 : "& arr(0)&"<br />")
     document.write("Value stored in Array index 1 : "& arr(1)&"<br />")
     document.write("Value stored in Array index 2 : "& arr(2)&"<br />")
document.write("Value stored in Array index 3 : "& arr(3)&"<br />")
document.write("Value stored in Array index 4 : "& arr(4)&"<br />")
document.write("Value stored in Array index 5 : "& arr(5)&"<br />")
</script>
</body>
</html>
```

When the above code is saved as .HTML and executed in Internet Explorer, it produces the following result :

```
Value stored inArray index 0:1
Value stored inArray index 1:VBScript
Value stored inArray index 2:100
Value stored inArray index 3:2.45
Value stored inArray index 4:7/10/2013
Value stored inArray index 5:12:45:00 PM
```

Multi Dimension Arrays

Arrays are not just limited to single dimension and can have a maximum of 60 dimensions. Two-dimension arrays are the most commonly used ones.

Example : In the below example, a multi-dimension array is declared with 3 rows and 4 columns.

```
<!DOCTYPE html>
<html>
<body>
<scriptlanguage="vbscript"type="text/vbscript">
Dim arr(2,3) 'Which has 3 rows and 4 columns
arr(0,0)="Apple"
arr(0,1)="Orange"
arr(0,2)="Grapes"
arr(0,3)="pineapple"
arr(1,0)="cucumber"
arr(1,1)="beans"
arr(1,2)="carrot"
arr(1,3)="tomato"
arr(2,0)="potato"
arr(2,1)="sandwitch"
arr(2,2)="coffee"
arr(2,3)="nuts"
document.write("Value in Array index 0,1 : "&  arr(0,1)&"<br />")
document.write("Value in Array index 2,2 : "&  arr(2,2)&"<br />")
```

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

When the above code is saved as .HTML and executed in Internet Explorer, it produces the following result :

```
Value stored inArray index :0,1:Orange
Value stored inArray index :2,2: coffee
```

Redim Statement

ReDim Statement is used to Declare dynamic-array variables and allocate or reallocate storage space.

ReDim[Preserve] varname(subscripts)[, varname(subscripts)]

- **Preserve:** An Optional parameter used to preserve the data in an existing array when you change the size of the last dimension.
- **varname:** A Required parameter, which denotes Name of the variable, which should follow the standard variable naming conventions.
- **subscripts:** A Required parameter, which indicates the size of the array.

Example : An array has been redefined and then preserved the values when the existing size of the array is changed.

Note: Upon resizing an array smaller than it was originally, the data in the eliminated elements will be lost.

```
<!DOCTYPE html>
<html>
<body>
<script language="vbscript" type="text/vbscript">
Dim a()
      i=0
      redim a(5)
      a(0)="XYZ"
      a(1)=41.25
      a(2)=22
REDIM PRESERVE a(7)
For i=3 to 7
      a(i)= i
Next
'to Fetch the output
For i=0 to ubound(a)
Msgbox a(i)
Next
</script>
</body>
</html>
```

When we save the above script as HTML and execute it in Internet Explorer, it produces the following result.

```
XYZ
41.25
22
3
4
5
6
7
```

Array Methods

There are various inbuilt functions within VBScript which help the developers to handle arrays effectively. All the methods that are used in conjunction with arrays are listed below. Please click on the method name to know in detail.

Function	Description
LBound	A Function, which returns an integer that corresponds to the smallest subscript of the given arrays.
UBound	A Function, which returns an integer that corresponds to the Largest subscript of the given arrays.
Split	A Function, which returns an array that contains a specified number of values. Splitted based on a Delimiter.
Join	A Function, which returns a String that contains a specified number of substrings in an array. This is an exact opposite function of Split Method.
Filter	A Function, which returns a zero based array that contains a subset of a string array based on a specific filter criteria.
IsArray	A Function, which returns a boolean value that indicates whether or not the input variable is an array.
Erase	A Function, which recovers the allocated memory for the array variables.

i) L. Bound

The LBound Function returns the smallest subscript of the specified array. Hence, LBound of an array is ZERO.

- ArrayName, a Required parameter. This parameter corresponds to the name of the array.
- Dimension, an Optional Parameter. This takes an integer value that corresponds to the dimension of the array. If it is '1', then it returns the lower bound of the first dimension; if it is '2', then it returns the lower bound of the second dimension and so on.

Example

```
<!DOCTYPE html>
<html>
<body>
<scriptlanguage="vbscript"type="text/vbscript">
```

```
Dim arr(5)
arr(0)=”1"’Number as String
arr(1)=”VBScript’String
arr(2)=100          ‘Number
arr(3)=2.45         ‘DecimalNumber
arr(4)=#10/07/2013#’Date
arr(5)=#12.45 PM#’Time
document.write(“The smallest Subscript value of  the given array is : “&LBound(arr))
‘ForMultiDimensionArrays:
Dim arr2(3,2)
document.write(“The smallest Subscript of the first dimension of arr2 is : “&LBound(arr2,1)&”<br />”)
document.write(“The smallest Subscript of the Second dimension of arr2 is : “&LBound(arr2,2)&”<br />”)
</script>
</body>
</html>
```

ii) UBound

The UBound Function returns the Largest subscript of the specified array. Hence, this value corresponds to the size of the array.

- ArrayName, a Required parameter. This parameter corresponds to the name of the array.
- dimension, an Optional Parameter. This takes an integer value that corresponds to dimension of the array. If it is ‘1’, then it returns the lower bound of the first dimension; if it is ‘2’, then it returns the lower bound of the second dimension, and so on.

Example

```
<!DOCTYPE html>
<html>
<body>
<script language=”vbscript” type=”text/vbscript”>
Dim arr(5)
arr(0)=”1"’Number as String
arr(1)=”VBScript”’String
arr(2)=100          ‘Number
arr(3)=2.45         ‘DecimalNumber
arr(4)=#10/07/2013#’Date
```

```
        arr(5)=#12.45 PM#'Time
        document.write("The Largest Subscript value of  the given array is : "&UBound(arr))
        'ForMultiDimensionArrays:
        Dim arr2(3,2)
        document.write("The Largest Subscript of the first dimension of arr2 is : "&UBound(arr2,1)&"
        <br />")
        document.write("The Largest Subscript of the Second dimension of arr2 is :
        "&UBound(arr2,2)&"<br />")
        </script>
        </body>
        </html>
```

iii) Split

A Split Function returns an array that contains a specific number of values splitted based on a Delimiter.

Syntax

Split(expression[,delimiter[,count[,compare]]])

- expression, a Required parameter. The String Expression that can contain strings with delimiters.
- delimiter, an Optional Parameter. The Parameter, which is used to convert into arrays based on a delimiter.
- count, an Optional Parameter. The number of substrings to be returned, and if specified as -1, then all the substrings are returned.
- compare, an Optional Parameter. This parameter specifies which comparison method to be used.
  - 0 = vbBinaryCompare - Performs a binary comparison
  - 1 = vbTextCompare - Performs a textual comparison

Example

```
<!DOCTYPE html>
<html>
<body>
<script language="vbscript" type="text/vbscript">
' Splitting based on delimiter comma '$'
a=Split("Red $ Blue $ Yellow","$")
b=ubound(a)
For i=0 to b
   document.write("The value of array in "& i &" is :"& a(i)&"<br />")
Next
</script>
</body>
</html>
```

iv) Join

A Function, which returns a String that contains a specified number of substrings in an array. This is an exact opposite function of Split Method.

Syntax

Join(List[,delimiter])

- List, a Required parameter. An Array that contains the substrings that are to be joined.
- delimiter, an Optional Parameter. The Character, which used as a delimiter while returning the string. The Default delimiter is Space.

Example

```
<!DOCTYPE html>
<html>
<body>
<script language=”vbscript” type=”text/vbscript”>
‘Join using spaces
a = array(“Red”,”Blue”,”Yellow”)
b = join(a)
document.write(“The value of b “&” is :”& b &”<br />”)
‘Join using $
b = join(a,”$”)
document.write(“The Join result after using delimiter is : “& b &”<br />”)
</script>
</body>
</html>
```

v) Filter

A Filter Function, which returns a zero-based array that contains a subset of a string array based on a specific filter criteria.

Syntax

Filter(inputstrings,value[,include[,compare]])

- **inputstrings, a Required parameter.** This parameter corresponds to the array of strings to be searched.
- **value, a Required Parameter.** This parameter corresponds to the string to search for against the inputstrings parameter.
- **include, an Optional Parameter**. This is a Boolean value, which indicates whether or not to return the substrings that include or exclude.
- **compare, an Optional Parameter.** This Parameter describes what string comparison method to be used.
  - 0 = vbBinaryCompare - Performs a binary comparison
  - 1 = vbTextCompare - Performs a textual comparison

Example

```
<!DOCTYPE html>
<html>
<body>
<script language="vbscript" type="text/vbscript">
a= array("Red","Blue","Yellow")
b =Filter(a,"B")
c =Filter(a,"e")
d =Filter(a,"Y")
For each x in b
Document.write("The Filter result 1: "& x &"<br />")
Next
For each y in c
Document.write("The Filter result 2: "& y &"<br />")
Next
For each z in d
Document.write("The Filter result 3: "& z &"<br />")
Next
</script>
</body>
</html>
```

vi) Erase

The Erase Function is used to reset the values of fixed size arrays and free the memory of the dynamic arrays. It behaves depending upon the type of the arrays.

Syntax

EraseArrayName

- Fixed numeric array, each element in an array is resetted to Zero.
- Fixed String array, each element in an array is resetted to Zero length.
- Array of Objects, each element in an array is resetted to special value Nothing.

Example

```
<!DOCTYPE html>
<html>
<body>
<script language="vbscript" type="text/vbscript">

DimNumArray(3)
NumArray(0)="VBScript"
NumArray(1)=1.05
NumArray(2)=25
NumArray(3)=#23/04/2013#

DimDynamicArray()
ReDimDynamicArray(9)'Allocate storage space.
EraseNumArray'Each element is reinitialized.
EraseDynamicArray'Free memory used by array.
'All values would be erased.
```

```
Document.write("The value at Zeroth index of NumArray is "&NumArray(0)&"<br />")
Document.write("The value at First index of NumArray is "&NumArray(1)&"<br />")
Document.write("The value at Second index of NumArray is "&NumArray(2)&"<br />")
Document.write("The value at Third index of NumArray is "&NumArray(3)&"<br />")
</script>
</body>
</html>
```

4.7 Classes & Objects

Q10. What is a Class and Object ?

Ans :

Class is a construct that is used to define a unique type. Like Object Oriented Programming, VbScript 5.0 supports the creation of classes and it is very similar to writing COM objects with VB.

Class is simply the template for an object and we instantiate an object to access the properties and methods of it. Classes can contain variables, properties, methods or events.

Syntax

VBScript classes are enclosed within Class .... End Class

```
'Defining the Class
Class classname   'Declare the object name
...
EndClass


' Instantiation of the Class
Set objectname = new classname
```

Class Variables

Classes can contain variables, which can be of private or public. Variables within classes should follow VBScript naming conventions. By default, the variables in class are Public. That is why they can be accessed outside the class.

Dim var1 , var2.

Private var1 , var2.

Public var1 , var2.

Class Properties

Class properties, such as Property Let, which handles the process of data validation and assigning the new value to the private variable. Property set, which assigns the new property value to the private object variable.

Read-only properties have only a Property Get procedure while write-only properties (which are rare) have only a Property Let or a Property Set procedure.

Example

In the below example, we are using Properties to wrap private variables.

```
Class Comp
Private modStrType
Private OS
Public Property  LetComputerType(strType)
        modStrType = strType
EndProperty
PublicPropertyGetComputerType()
ComputerType= modStrType
EndProperty
PublicPropertySetOperatingSystem(oObj)
Set OS = oObj
EndProperty
PublicPropertyGetOperatingSystem()
SetOperatingSystem= OS
EndProperty
EndClass
```

Class Methods

Methods allow the class to perform the operation that the developer wants. The Methods are nothing but Functions or Subroutines.

Example

In the below example, we are using Proper-ties to wrap private variables.

```
ClassCar
Private Model
Private Year
Public Start()
Fuel=2.45
        Pressure=4.15
EndFunction
EndClass
```

Class Events

There are two events that are automatically associated with every class by default. Class_Initialize and Class_Terminate.

Class_Initialize is triggered whenever you instantiate an object based on the class. Class_Terminate event is fired when the object goes out of scope or when the object is set to Nothing.

Example

In the below example, we will make you understand how the events work in VBScript.

```
'Instantation of the Object
Set objectname = New classname
        Private Sub Class_Initialize( )
        Initalization code goes here
End Sub
'When ObjectisSet to Nothing
Private SubClass_Terminate()
Termination code goes here
End Sub
```

What is an Object

VBScript runtime objects help us to accomplish various tasks. This section will help you understand how to instantiate an object and work with it.

Syntax

In order to work with objects seamlessly, we need to declare the object and instantiate it using Set Keyword.

```
    Dim objectname 'Declare the object name
    Set objectname = CreateObject(object_type)
```

Example

In the below example, we are creating an object of type Scripting.Dictionary.

```
Dim obj
    Set obj =CreateObject ("Scripting. Dictionary")
```

Destroying the Objects

The significance of destroying the Object is to free the memory and reset the object variable.

Syntax

In order to destroy the objects, we need to use Set Keyword followed by the object name and point it to Nothing.

```
    Set objectname =Nothing'Destroy the object.
```

Example

In the below example, we are creating an object of type Scripting.Dictionary.

```
Dim obj
    Set obj =CreateObject("Scripting.Dictionary")
    Set obj =Nothing.
```

Object Usage

Please click on each one of the given object types to know more.

Object Type	Description
Class	Class is a container, which holds methods and variables associated with it and accessed by creating an object of Type Class.
Scripting.FileSystemObject	It is the group of objects with which we can work with file system.
Scripting.Dictionary	A Group of objects, which are used for creating the dictionary objects.
Debug	A Global Object with which we can send output to the Microsoft script debugger.

4.8 Web Server

Q11. What isa web Server? Explain its Working ?

Ans :

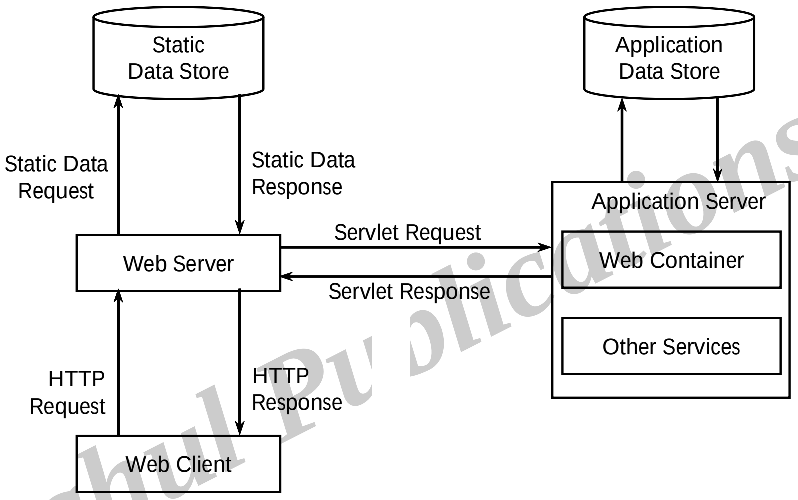
Web server is a computer where the web content is stored. Basically web server is used to host the web sites but there exists other web servers also such as gaming, storage, FTP, email etc.

Web site is collection of web pages while web server is software that responds to the request for web resources.

Web Server Working

Web server respond to the client request in either of the following two ways:

- Sending the file to the client associated with the requested URL.
- Generating response by invoking a script and communicating with database



Key Points

- When client sends request for a web page, the web server search for the requested page if requested page is found then it will send it to client with an HTTP response.
- If the requested web page is not found, web server will the send an HTTP response:Error 404 Not found.
- If client has requested for some other resources then the web server will contact to the application server and data store to construct the HTTP response.

Architecture

Web Server Architecture follows the following two approaches:

1. Concurrent Approach
2. Single-Process-Event-Driven Approach.

Concurrent Approach

Concurrent approach allows the web server to handle multiple client requests at the same time. It can be achieved by following methods:

- Multi-process
- Multi-threaded
- Hybrid method.

Multi-processing

In this a single process (parent process) initiates several single-threaded child processes and distribute incoming requests to these child processes. Each of the child processes are responsible for handling single request.

It is the responsibility of parent process to monitor the load and decide if processes should be killed or forked.

- **Multi-threaded:** Unlike Multi-process, it creates multiple single-threaded process.
- **Hybrid:** It is combination of above two approaches. In this approach multiple process are created and each process initiates multiple threads. Each of the threads handles one connection. Using multiple threads in single process results in less load on system resources.

Examples

Following table describes the most leading web servers available today:

S.N.	Web Server Description
1	**Apache HTTP Server** This is the most popular web server in the world developed by the Apache Software Foundation. Apache web server is an open source software and can be installed on almost all operating systems including Linux, UNIX, Windows, FreeBSD, Mac OS X and more. About 60% of the web server machines run the Apache Web Server.
2.	**Internet Information Services (IIS)** The Internet Information Server (IIS) is a high performance Web Server from Microsoft. This web server runs on Windows NT/2000 and 2003 platforms (and may be on upcoming new Windows version also). IIS comes bundled with Windows NT/2000 and 2003; Because IIS is tightly integrated with the operating system so it is relatively easy to administer it.
3.	**Lighttpd** The lighttpd, pronounced lighty is also a free web server that is distributed with the FreeBSD operating system. This open source web server is fast, secure and consumes much less CPU power. Lighttpd can also run on Windows, Mac OS X, Linux and Solaris operating systems.
4.	**Sun Java System Web Server** This web server from Sun Microsystems is suited for medium and large web sites. Though the server is free it is not open source. It however, runs on Windows, Linux and UNIX platforms. The Sun Java System web server supports various languages, scripts and technologies required for Web 2.0 such as JSP, Java Servlets, PHP, Perl, Python, and Ruby on Rails, ASP and Coldfusion etc.
5.	**Jigsaw Server** Jigsaw (W3C's Server) comes from the World Wide Web Consortium. It is open source and free and can run on various platforms like Linux, UNIX, Windows, and Mac OS X Free BSD etc. Jigsaw has been written in Java and can run CGI scripts and PHP programs.

4.9 Personal Web Server

Q12. Explain the concept of Personal Web Server.

Ans :

A personal web server (PWS) is system of hardware and software that is designed to create and manage a web server on a desktop computer. It can be used to learn how to set up and administer a website, and it can also serve as a site for testing dynamic web pages. One of the main functions of PWS is to provide an environment where web programmers can test their programs and web pages. Therefore, a PWS supports the more common server-side programming approaches that can be used with production web servers.

A personal web server, or personal server in short, allows users to store, selectively share, or publish information on the web or on a home network. Unlike other types of web servers, a personal web server is owned or controlled by an individual, and operated for the individual's needs, instead of by a company. It can be implemented in different ways:

- As a computer appliance
- As a general-purpose server, such as a Linux server, which may be located at the owner's home or in a data center
- In a shared hosting model, where several users share one physical server by means of virtualization, or virtual hosting.
- As one feature of a computer that is otherwise also used for other purposes.

A personal web server is conceptually the opposite of a web server, or website, operated by third parties, in a software as a service (SaaS) or cloud model.

4.10 Internet Information Server

Q13. Write a short note on IIS Server ?

Ans :

IIS (which stands for Internet Information Services or Internet Information Server) also known as Windows web server is available on most versions of Microsoft Windows operating systems and takes second place in overall usage behind Apache HTTP Server on the internet. It will host websites, web applications and services needed by users or developers. Many versions have shipped as far back as IIS 1 on Windows 3 and with nearly every new Windows OS a new IIS version follows.

Versions and History

Microsoft Windows Server 2003 or IIS 6 is the oldest version you would want to run for anything outside of a hobby or testing, which does supports IPV6 as well as modern security measures. However in a professional environment IIS 8.5 or 10 (Still in Beta) will receive official software updates for years to come and support more modern applications and needs.

- IIS 6 or Windows Server 2003 is no longer receiving any updates from Microsoft but supports IPV6 and most security measures needed for simple hosting needs on a budget.
- IIS 7 shipped with Windows Vista and has better support for the .NET framework and some security enhancements over IIS 6.
- IIS 7.5 Shipped with Windows 7 and added support for TLS 1.1 and 1.2. Extended support will end in 2020 this is the oldest version receiving any support officially from microsoft.

- IIS 8 also known as Microsoft Web Server 2012 began supporting SNI or associating SSL to hostnames instead of IP addresses and multicore scaling. Support will last until 2023.
- IIS 8.5 shipped with Windows 8.1 and has new features such as Enhanced logging capabilities and Dynamic Site Activation.
- IIS 10 is currently in beta and will support modern technology such as HTTP/2 and powershell 5.0.

If you are a business owner consider purchasing the newest version your hardware can run. IIS 8.5 is currently the most stable and secure version as of this writing, however once out of beta ISS 10 will become your best bet. If you are hosting a basic website on your own and cannot afford a newer version consider Apache Server instead of anything older than IIS 6.

Virtual Directories

IIS allows you to create sites, applications, and virtual directories to share information with users over the Internet or internally on an intranet such as a home network. This concept did exist in older versions of IIS, but several changes took place in IIS 7 and changed the definition and functionality of this concept.

A virtual directory is a name that you specify in IIS and that maps to a physical directory on a server similar to how DNS maps a URL to an IP address. The directory name becomes part of the application allowing users to navigate to a website or application and gain access to the content hosted on the server. This content could be a website itself or media such as photos or videos within a web application or site.

In IIS 6.0, virtual directories and applications were considered to be separate objects even though they were the same thing. An application was not a physically separate object from a virtual directory instead an app was really just a virtual directory on its own with one of the following properties in its metabase: AppFriendlyName, AppRoot, AppIsolated, and AppPoolID. The only issue was creating a system where applications in one pool would not be allowed to communicate with applications in another pool on the same server.

In IIS 7.0 and above virtual directories and applications are separate objects and functioned in that manner. They exist in a hierarchical relationship such as a website may contain one or more applications, an application contains one or more virtual directories, and a virtual directory maps to a physical directory on a computer.

Q14. How to set up your first IIS Web site ?

Ans :

When you install IIS, it is preconfigured to serve as a default Web site; however, you may want to change some of the settings. To change the basic settings for the Web site and to emulate the steps that are required to set up Apache for the first time by using the configuration file :

1. Log on to the Web server computer as an administrator.
2. Click Start, point to Settings, and then click Control Panel.
3. Double-click Administrative Tools, and then double-click Internet Services Manager.
4. Right-click the Web site that you want to configure in the left pane, and then click Properties.
5. Click the Web site tab.
6. Type a description for the Web site in the Description box.
7. Type the Internet Protocol (IP) address to use for the Web site or leave the All (Unassigned) default setting.
8. Modify the Transmission Control Protocol (TCP) port as appropriate.
9. Click the Home Directory tab.
10. To use a folder on the local computer, click A directory on this computer, and then click Browse to locate the folder that you want to use.

11. To use a folder that has been shared from another computer on the network, click A share located on another computer, and then either type the network path or click Browse to select the shared folder.

12. Click Read to grant read access to the folder (required).

13. Click OK to accept the Web site properties.

Create a new Web site

To create a new Web site in Apache, you must set up a virtual host and configure the individual settings for the host. If you are using IIS, you can create a new Web site by translating the following terms to the IIS equivalents :

Apache term	IIS term
DocumentRoot	IIS Web Site Home Directory
ServerName	IIS Host Header
Listen	IIS IP Address and TCP Port

To create a new Web site in IIS, follow these steps :

1. Log on to the Web server computer as an administrator.
2. Click Start, point to Settings, and then click Control Panel.
3. Double-click Administrative Tools, and then double-click Internet Services Manager.
4. Click Action, point to New, and then click Web Site.
5. After the Web Site Creation Wizard starts, click Next.
6. Type a description for the Web site.

This description is used internally to identify the Web site in Internet Services Manager only.

1. Select the IP address to use for the site.If you select All (unassigned), the Web site is accessible on all interfaces and all configured IP addresses.
2. Type the TCP port number to publish the site on.
3. Type the Host Header name (the real name that is used to access this site).
4. Click Next.
5. Either type the path to the folder that is holding the Web site documents or click Browse to select the folder, and then click Next.
6. Select the access permissions for the Web site, and then click Next.
7. Click Finish.

4.11 Apache Web Server

Q15. What is HTTP Apache Server? How to install Apache ?

Ans :

There are numerous ways of installing the package or application. There are enlisted below:

1. One of the features of this open source web application is that anyone can make installer as per their own environment. This has allowed various vendors like Debian, Red Hat, FreeBSD, Suse etc. to customize the file location and configuration of apache taking into account other installed applications and base OS.

2. Apart from installing it from a vendor based installer, there is always the option of building and installing it from the source code. Installing Apache from source file is a platform independent & works for all OS.

The apache web server is a modular application where the administrator can choose the required functionality and install different modules as per his/her requirement.

All modules can be compiled as a Dynamic Shared Objects (DSO is an object file that could be shared by multiple apps while they are executing) that exists separately from the main apache file. The DSO approach is highly recommended, it makes the task of adding/ removing/updating modules from the servers configuration very simple.

Install Apache:Linux Platform

On Red Hat or rpm based systems

If you are using an rpm (RedHat Package Manager is a utility for installing application on Linux systems) based Linux distribution i.e. Red Hat, Fedora, CentOs, Suse, you can install this application by either vendor specific Package Manager or directly building the rpm file from the available source tarball.

You can install Apache via the default Package Manager available on all Red Hat based distributions like CentOs, Red Hat and Fedora.

```
[root@amsterdam ~]# yum install httpd
```

The apache source tarball could be converted into an rpm file using the following command.

```
[root@amsterdam ~]# rpmbuild -tb httpd-2.4.x.tar.bz2
```

It is mandatory to have -devel package installed on your server for creating .rpm file from source.

Once you convert the source file into an rpm installer, you could use the following command to install Apache.

```
[root@amsterdam ~]# rpm –ivh httpd-2.4.4-3.1.x86_64.rpm
```

After the installation the server does not start automatically, in order to start the service, you have to use any of the following command on Fedora, CentOs or Red Hat.

```
[root@amsterdam ~]# /usr/sbin/apachectl start
[root@amsterdam ~]# service httpd start
[root@amsterdam ~]# /etc/init.d/httpd start
```

Install Apache from Source

Installing apache from the source require the –devel package to be installed on your server. .You can find the latest available version of Apache, you can download it here. Once you download the source file move it to the /usr/local/src folder.

```
[root@amserversterdam ~] cd /usr/local/src
[root@amserversterdam ~] gzip -d httpd-2.2.26.tar.gz
[root@amserversterdam ~] tar xvf httpd-2.2.26.tar
[root@amserversterdam ~] httpd-2.2.26
```

In order to see all configuration option available for Apache, you can use ./configure –help option. The most common configuration option is –prefix={install directory name}.

```
[root@amserversterdam ~]./configure —help
[root@amserversterdam ~]./configure –prefix=/usr/local/apache –enable-so
[root@amserversterdam ~] make
[root@amserversterdam ~] make install
```

The above example shows the compilation of Apache within the /usr/local/apache directory with the DSO capability. The –enable-so option, can load required modules to apache at run time via the DSO mechanism rather than requiring a recompilation.

Once the installation completes, you can browse the web servers default page with your favorite browser. If firewall is enabled on your server, you must have to make exception for port 80 on your OS firewall. You can use the following command to open port 80.

```
iptables -I INPUT -p tcp —dport 80 -j ACCEPTservice iptables save
```

What is Virtual Host ?

An Apache web server can host multiple websites on the SAME server. You do not need separate server machine and apache software for each website. This can achieved using the concept of Virtual Host or VHost.

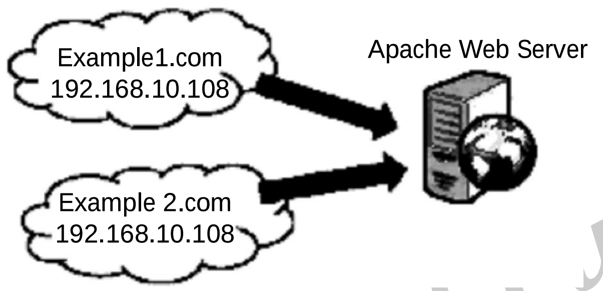
Any domain that you want to host on your web server will have a separate entry in apache configuration file.

Types of Apache Virtualhost

1. Name-based Virtual host
2. Address-based or IP based virtual host and.

Name-based Virtual Host

Name based virtual hosting is used to host multiple virtual sites on a single IP address.



In order to configure name based virtual hosting, you have to set the IP address on which you are going to receive the Apache requests for all the desired websites. You can do this by NameVirutalHost directive within the apache configuration i.e. httpd.conf/apache2.conf file.

Apache virtual host Example:

```
NameVirtualHost *:80
<VirtualHost 192.168.0.108:80>
ServerAdmin webmaster@example1.com
DocumentRoot /var/www/html/example1.com
ServerName www.example1.com
</VirtualHost>
<VirtualHost 192.168.0.108:80>
ServerAdmin admin@example2.com
DocumentRoot /var/www/html/example2.com
ServerName www.example2.com
</VirtualHost>
```

You can add as many virtual hosts, as per your requirement. You can check your web configuration files with:

```
[root@amsterdam ~]#httpd –t
```

Syntax ok

If the configuration file has some wrong syntax, it will throw an error

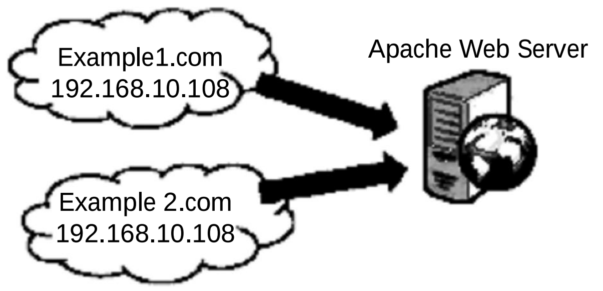
```
[root@115 conf.d]# httpd -t
```

Syntax error on line 978 of /etc/httpd/conf/httpd.conf:

Invalid command '*', perhaps misspelled or defined by a module not included in the server configuration

IP-based Virtual host

In order to setup IP based virtual hosting, you need more than one IP address configured on your server. So, the number of vhost apache will depend on number of IP address configured on your server. If your server has 10 IP addresses, you can create 10 IP based virtual hosts.



What Apache needs to Run Php File ?

Running Php files on Apache needs mod _php enabled on your server. It allows Apache to interpret .Php files. It has Php handlers that interpret the Php code in apache and send HTML to your web server.

If mod_php is enabled on your server, you will have a file named php.conf in /etc/httpd/conf.d/ directory. You can also check it with:

```
httpd -M | grep "php5_module"
```

4.12 Installation of a Web Server

Q16. How to install a Web Server?

Ans :

The Web Server (IIS) role in Windows Server 2016 provides a secure, easy-to-manage, modular and extensible platform for reliably hosting websites, services, and applications. With IIS, you can share information with users on the Internet, an intranet, or an extranet. IIS is a unified web platform that integrates IIS, ASP.NET, FTP services, PHP, and Windows Communication Foundation (WCF).

When you deploy server certificates, your Web server provides you with a location where you can publish the certificate revocation list (CRL) for your certification authority (CA). After publication, the CRL is accessible to all computers on your network so that they can use this list during the authentication process to verify that certificates presented by other computers are not revoked.

If a certificate is on the CRL as revoked, the authentication effort fails and your computer is protected from trusting an entity that has a certificate that is no longer valid.

1. In Server Manager, click Manage, and then click Add Roles and Features. The Add Roles and Features Wizard opens.
2. In Before You Begin, click Next.

   Note The Before You Begin page of the Add Roles and Features Wizard is not displayed if you have previously run the Add Roles and Features Wizard and you selected Skip this page by default at that time.
3. On the Installation Type page, click Next.
4. On the Server selection page, click Next.
5. On the Server roles page, select Web Server (IIS), and then click Next.
6. Click Next until you have accepted all of the default web server settings, and then click Install.
7. Verify that all installations were successful, and then click Close.

UNIT V

Active Sever Pages, Client side Scripting vs Server side Scripting, Server side Active X Component, ADO, file system objects, Session tracking, CGI and PERL5, String. Processing and Regular Expressions, Server side includes, Cookies and PERL XML Document Type Definition, XML Parsers, Using XML with HTML.

5.1 Active Server Pages

5.1.1 Client side Scripting vs Server side Scripting

Q1. What is ASP? Explain its directories and working ?

Ans : **(Imp.)**

Active Server Pages (ASPs) are Web pages that contain server-side scripts in addition to the usual mixture of text and HTML tags. Server-side scripts are special commands you put in Web pages that are processed before the pages are sent from the server to the web-browser of someone who's visiting your website. If that file is a normal HTML file, it looks the same when your web-browser receives it as it did before the server sent it. After receiving the file, your web-browser displays its contents as a combination of text, images, and sounds.

Active Server Pages is an open, compile-free application environment in which you can combine HTML, scripts, and reusable ActiveX server components to create dynamic and powerful Web-based business solutions. Active Server Pages enables server side scripting for IIS with native support for both VBScript and JavaScript.

In the case of an Active Server Page, the process is similar, except there's an extra processing step that takes place just before the server sends the file. Before the server sends the Active Server Page to the browser, it runs all server-side scripts contained in the page. Process information the user has just typed into a form, such as a page in the website's guestbook. And you can write your own code to put in whatever dynamic information you want.

To distinguish Active Server Pages from normal HTML pages, Active Server Pages are given the ".asp" extension.

Server Side Scripting

Server-side scripts typically start with <% and end with %>. The <% is called an opening tag, and the %> is called a closing tag. In between these tags are the server-side scripts. You can insert server-side scripts anywhere in your webpage - even inside HTML tags.

Requirements to Run ASP Files

The server must do additional processing on the ASP scripts, it must have the ability to do so. The only servers which support this facility are Microsoft Internet Information Services & Microsoft Personal Web Server.

IIS (Internet Information Server)

This is Microsoft's web server designed for the Windows NT platform. It can only run on Microsoft Windows NT 4.0, Windows 2000 Professional, & Windows 2000 Server. The current version is 5.0, and it ships as a part of the Windows 2000 operating system.

PWS (Personal Web Server)

This is a stripped-down version of IIS and supports most of the features of ASP. It can run on all Windows platforms, including Windows 95, Windows 98 & Windows Me. ASP developers use PWS to develop their sites on their own machines and later upload their files to a server running IIS.

Steps to Install Servers

You cannot view Active Server Pages without running a web-server. To test your own pages, you should save your pages in a directory mapped as a virtual directory, and then use your web-browser to view the page.

1. From the CD, run the SETUP.EXE program for starting the web-server installation.
2. After the installation is complete, go Start> Settings > Control Panel > Personal Web Manager. and click the “Start” button under
3. Now your web-server is up & running.

Creating Virtual Directories

After you have installed the web-server, you can create virtual directories as follows:

- Right-Click on the folder that you wish to add as a virtual
- Select “Properties” from the context-menu
- In the second tab titled “Web Sharing,” click “Share this folder,” then “Add Alias”.

Steps to navigate to this directory:

1. Open up “My Computer” or “Windows Explorer” so that you can view your system’s files and directories.
2. Select and Open the C drive (C:)
3. Double click the Inetpub folder
4. Double click the wwwroot folder - The full path to this location is “C:\Inetpub\wwwroot” for you advanced users.
5. Within the wwwroot directory locate the “localstart.asp” file.

Creating ASP Testing Grounds

1. Throughout this tutorial we will be referring to the folder “vyomASP” that you should create if you want to follow along with these ASP Tutorials.
2. Inside the wwwroot folder create a New Folder and rename it “vyomASP”
3. To access this directory you would type the following into Internet Explorer:
4. http://localhost/vyomASP/FILENAME.asp
5. Where FILENAME is name of your ASP file
6. All of the files you create while reading the ASP Tutorial should go into this directory.

ASP Syntax

```
<%
Response.Write(“Hello world”)
%>
```

These special tags <% and %> will encapsulate your ASP code. Some things about the ASP tag that sets it apart from normal HTML tags: An opening ASP tag is <% while an HTML tag normally looks like <tagname> A closing ASP tag looks like %> while an HTML tag looks like </tagname> ASP code can occurr anywhere, even within an HTML tag opening tag

Accessing Web page

Your server is completely configured and ready to use, why not give it a try? Start your web-browser, and enter the following address into the address-bar.

http://localhost/

You should see a page come up that tells you more about Microsoft IIS.

Local Host

What we mean by a hostname. Whenever you connect to a remote computer using it’s URL, you are in effect calling it by its hostname.

For example

http://www.google.com/

You are really asking the network to connect to a computer named www.google.com. It is called the “hostname” of that computer.

Local host is a special hostname. It always references your own machine. What you just did, was to try to access a webpage on your own machine for testing all your pages, you will need to use localhost as the hostname. By the way, there is also a special IP address associated with localhost.

Q2. Explain the representation of ASP in VB Script and Java Script?

Ans :

We have our ASP Code working; say that we wanted to use our ASP code along with some client side JavaScript code. We are just going to use the JavaScript write function and have our ASP Code fill in the Date. All the JavaScript client code is in the default color, but ASP Server Side Code is in Red.

ASP with VBScript

You should be able to follow along with our ASP tutorial with little or no VBScript knowledge,

ASP in JavaScript

VBScript is the default scripting language that ASP is coded in, so if you want to specify a different scripting language you have to state which scripting language you will be using at the very beginning of your code. Below is the line of code that must be your first line of ASP code or else your page will break and you'll get a boring error message

```
<%@Language=”javascript”
‘The rest of your ASP Code....
%>
```

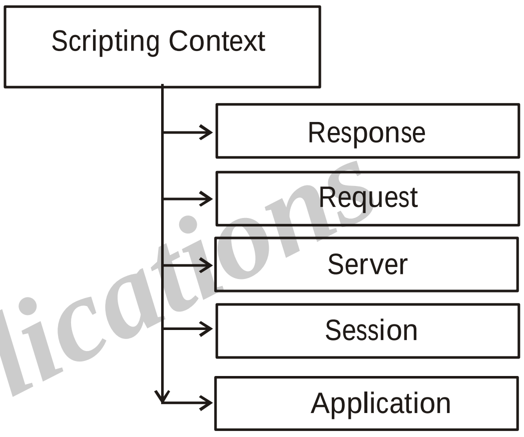
Q3. Discuss different Components of ASP ?

Ans :

- The important points about ASP is that the script runs entirely on a serve, thus protecting your intellectual property and shielding you from browser differences you had to account for when writing client side code.
- Your entire server side code is processed and only plain HTML or whatever kind of content you chose to generate is sent to the client.
- When the web server receives the request for the 123.asp page, it invokes the ASP engine, which in turn invokes the appropriate scripting language to execute the server side code you added to 123.asp.
- One of the big advantages of ASP is that it is language agnostic, meaning that the program isn't limited to the Microsoft provided standard scripting languages, such as VB script.
- It can be extended by the scripting languages of your choice, as long as this scripting language supports the ActiveX scripting model.
- Because a scripting language by itself doesn't know what a web server is. The Asp engine supplies objects and makes them available to your code allowing you to interact with the client's browser, accessing server functionality and more.

We have various built in objects available like,

1. Application
2. Session
3. Server
4. Request
5. Response
6. Object context



ASP Technology:

- ASP stands for Active Server Pages
- ASP is a program that runs inside the IIS (Internet Information Services)
- IIS comes as a free component with windows
- PWS, personal web server, is a smaller – but fully functional version of IIS.

ASP compatibility

- ASP is a Microsoft Technology.
- To run IIS you must have window NT 4.0 or later.
- To run PWS you must have windows 95 or later.

Components and Objects

Components and objects are the tools used to communicate with the server's environment and systems. Components can contain one or more objects. An object can have one or more methods and one or more properties. By creating the instance of an object you can use its methods to perform tasks. Changing objects properties will cause its methods to perform task differently.

Several objects are built into ASP, some are built into VBScript and some are built into the server's system. Other components are can be created to further customize an application.

Objects Or Components	**Functions**	**Sources**
Request Object	Handles information coming from	Built into ASP
Response	Handles information sent from	Built into ASP
Server Object	Provides Access to some basic Services	Built into ASP
Application Object	Maintain Information for the life application	Built into ASP
Text Object	Manipulate the Text Files	VBScript Object
Error Object	Provides for error analysis	VBScript Object
Content Linking	Impart the order to the page	Server Component
Active Database	Provides an information	Server Component
File System Object	Provides access to the page	VBScript Object
Browser Capable	Specifies what a user browse	Server Component
Voting	Collect information from a user	Server Component

- **Request -** To get the information For the user
- **Response -** To send the information To the user
- **Server -** To control the internet information Server
- **Session -** To store information about and change settings for the user's current Web-Server Session
- **Application -** To share application-level information and control settings for the life time of the application

Q4. Explain various variable and its constructs in ASP.

Ans : **(Imp.)**

Variable in ASP

1. Always declare all your variables before you use them, even though it is not required. Nearly all programming languages require you to declare variables so that it will increase your program's readability.
2. In ASP you declare a variable with the use of the Dim keyword, which means Dimension.
3. Dimension refers to the amount of space something takes up in the real world, but in computer terms it refers to space in computer memory.
4. Variables can be declared all at once or one at a time . Below is an example of both methods.

Syntax

```
<%
'Single Variable Declarations
Dim myVar1
Dim myVar2
'Multiple Variable Declarations
Dim myVar6, myVar7, myVar8
%>
```

ASP Variable Naming Conventions

Default language for ASP is VBScript and so it also uses VBScripts variable naming conventions. These rules are :

1. Variable name must start with an alphabetic character.
2. Variables cannot contain a period.
3. Variables cannot be longer than 255 characters.
4. Variables must be unique in the scope in which it is declared.

Assigning Values to ASP Variables

You can assign values in ASP using equals "=" operator. Below we have set a variable equal to a number and a separate variable equal to a string.

```
myNum = 25
myString = Hello
myGarbage = I changed my variable
```

Example

```
<html>
<body>
<%
dim name
name="AlbertEinstein"
response.write("My name is: " & name)
%>
</body>
</html>
```

Dim

To "Dim" it means to Dimension it. That's VB lingo. A variable is declared in VBScript using the Dim keyword.

```
<%
Dim myVar
%>
```

VB programmers will notice here, that we have not included any indication of the type of the said variable. For example: Dim myString as String or Dim myString$.

In VBScript, all variables are variants. Their type is determined automatically by the runtime interpreter, and the programmer need not (and should not) bother with them.VBScript does not force requiring variable declaration. That is, it is allowed to use a variable directly without declaring it first. programmers know the importance of making it compulsory to declare all your variables first – without that, the bugs that may result are very difficult to detect.

Remember that Option Explicit must necessarily be the first statement of your ASP page, otherwise a server error is generated.

```
<%
Pi = 3.141592654 Response.Write Pi
%>
```

This is a perfectly valid page. You will get the value of Pi written back to the page, as you really expected.

```
<%
Option Explicit Pi = 3.141592654
%>
```

Now you have an error that says: Microsoft VBScript runtime (0x800A01F4) Variable is undefined: 'Pi' /asp/test.asp. In this case, the correct script must read:

```
<%
Option Explicit Dim Pi = 3.141592654
%>
```

Q5. Define array. Explain its working mechanism.

Ans :

One of the most common Array definitions says that an Array is a set of sequentially indexed elements (variables or objects) having one and the same data type. The array elements are indexed and each element of an array has its own unique index number.

The VBScript arrays are 0 based, meaning that the array element indexing starts always from 0. 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.

Syntax

Dim array_name(size)

Example

```
<%
Dim myFixedArray(3) 'Fixed size array
Dim myDynArray() 'Dynamic size array
%>
```

There are several ways to create a dynamic VBScript array. If you use the Dim statement along with the array's name, without specifying upper bound, you will create a dynamic array.

Syntax

Dim array_name

The other way is to use the VBScript Array function along with a Dim statement to create and populate your dynamic array:

Syntax

Dim array_name

array_name = Array("Peter", "Stephen", "Tracy", "Jerry")

Example

```
<%
Dim myDynArray() 'Dynamic size array
ReDim myDynArray(1)
myDynArray(0) = "Albert Einstein"
myDynArray(1) = "Mother Teresa"
ReDim Preserve myDynArray(3)
myDynArray(2) = "Bill Gates"
myDynArray(3) = "Martin Luther King Jr."
For Each item In myDynArray
Response.Write(item & "<br />")
Next
%>
```

Example

```
<html>
<head>
<title>arraysredimcorrect.asp</title>
</head>
<body bgcolor="#FFFFFF">
<%
Dim my_array ()
x=100
ReDim preserve my_array(x)
my_array(20)="Hi!"
my_array(40)="How are You"
lowcount=LBound(my_array)
highcount=UBound(my_array)
Response.Write "lowcount=" & lowcount &
";highcount=" & highcount & "<p>"
For counter=lowcount To highcount
Response.Write counter & "   "
Response.Write my_array(counter) &
"<br>"
Next
%>
</body>
</html>
```

5.2 Active X Component

5.2.1 ADO, file system objects

Q6. Whata are the Componenets of Active X Control ?

Ans :

Unlike the session, application, response, and request objects, ActiveX server components (which act like objects) must be explicitly instantiated in your ASP programs. The syntax for instantiating other objects is: <% Set My Object = Server. Create Object ("OBJECTNAME") %>. You then use the instance of the object, MyObject, in the rest of your program. OBJECTNAME is replaced with the name of an object or component. For example, to include a browser-capabilities component, you would replace OBJECTNAME with MSWC.BrowserType.

The Database-Access Component

Microsoft's ActiveX Data Object (ADO) provides access to ODBC databases. This is the most significant server component for typical Web-based database applications. To use a database in your ASP application, you must first create an ODBC data source for the database on your Web server. It should be created as a System Data Source Name (use the System DSN section in your ODBC).

To use this data source in an application, you must instantiate a connection object and open the data source identified in the ODBC Data Source Administrator.

More powerful ways exist to connect to data sources, but this method will get you started. To get some specific data, you could execute the appropriate SQL commands. Queries against the Listings table in the CatalogData database. To display the data, you refer to the result-set object. Output can be passed to the browser one value at a time using the = operator, as in. This is a simple alternative to the Response.Write method described earlier. "Title" is the name of a field in the Listings table.

The underlying complexity and robustness of the ADO model makes using databases in ASP quite complex. As employed in ASP, this model provides one distinctly nice feature: the ability to store commonly used data in the application or session objects. This means that if your application needs to reference the same set of data on a variety of pages, you needn't rerun the same query on each; instead, run the query once and associate the result set with either the session or application object.

For example, to check a user's security level at different places within an application, you would execute the query in one place and create session variables that can be referred to later.

Complex database applications will require that you roll your own data-validation routines, as ASP has no built-in mechanism for specifying that only certain types of data be allowed in certain fields.

Other Server Components

ASP also includes components that enable you to:

- Read files from and write files to the server file system.
- Rotate ads (GIFs) at user-defined frequencies.
- Determine browser capabilities.
- Specify URLs used for navigational purposes in a separate text file.

These components provide examples from which you can learn more about programming ASP apps. The tutorials and samples use these ActiveX objects extensively and informatively.

Conspicuously absent is a mechanism for producing email. ASP itself contains nothing that would allow you, for example, to send email to a selected group of people stored in a database, but several third-party tools exist for this purpose. The tools I've seen, ASPMail for example, dovetail right into ASP's object model.

Server Side Includes

All modern Web servers and some Web-based programming environments provide mechanisms for including one HTML file in another. This feature is known as server side includes (SSI). Unfortunately, the SSI syntax doesn't resemble VBScript at all. SSIs require a separate tag, like this:

<!—#INCLUDE FILE="myfile.htm"—>

This makes your code a bit harder to write (and read) as you must close off one chunk of code with a delimiter (%>) in order to execute an SSI, then open another with another delimiter (<%);This gets even more complex if your program needs to set the name of the include file at run time. It would be much simpler if there was a simple VBScript command for including files.

An Example

To demonstrate some ASP in action, we'll create a simple guest-book application that presents the entry form Submitting a form adds your entries to an Access database (in the CatalogData datasource, which has been set up as an ODBC System DSN on our server). An email confirmation of your entry is generated and sent to you. Listing Two contains the PostGuestBook.

The data validation requires all three fields in the form: FName, LName, and EMail. A special object — dctErrMsg — of type Scripting.Dictionary is used to store validation error messages. A Scripting. Dictionary is like an associative array in Perl. Think of it as a bag: An entry is made in the bag for each validation error. Before performing any of this script's functions, the bag is checked for entries using If dctErrMsg.Count=0 Then...If the bag is empty, we continue with the database insert and create the email confirmation. If the bag isn't empty, control flows to the section in which an error-message page is produced. This is done by looping through the entries in the bag after they've been stored in a regular array.

5.2.2 Session tracking in ASP

Q7. What are various sessions in ASP? Explain.

Ans :

To store information about, or change settings for a user session, the Session object is used. Session object is a Variables that holds information about one single user, and are available to all pages in one application. In ASP,the Session Object is a great tool for the modern web site. It allows you to keep information specific to each of your site's visitors. You don't have to worry about passing information page to page because information like username, shopping cart, and location can be stored for the life of the session.

When you are working with an application, you open it, do some changes and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the HTTP address doesn't maintain state so that the web server does not know who you are and what you do.

By creating a unique cookie for each user, ASP solves this problem. The cookie is sent to the client and it contains information that identifies the user. This interface is called the Session object. The Session object is used to store information about, or change settings. Session object is a variable that holds information about one single user, and are available to all pages in one application. Common information stored in session variables is name, id, and preferences. The server creates a new Session object for each new user, and destroys the Session object when the session expires.

After a new user requests an ASP file, and the Global.asa file includes a Session_OnStart procedure. In a Session variable, a value is stored. To instantiate an object with session scope a user requests an ASP file, and the Global.asp file uses the <object> tag.

When does a Session End?

If a user has not requested or refreshed a page in the application for a specified period,a session ends . By default, this is 20 minutes. you can set the Timeout property if you want to set a timeout interval that is shorter or longer than the default,

Syntax

```
<%
Session.Timeout=10
%>
```

or

```
<%
Session.Abandon  (You may use the
Abandon method to end a session
immediately)
%>
```

Store and Retrieve Session Variables

You can store variables in it is the most important thing about the Session object.

Example

```
<%
Session("username")="Martin Luther"
Session("age")=40
%>
```

It can be reached from ANY page in the ASP application if the value is stored in a session variable:

Example

Welcome <%Response.Write(Session("username"))%>

In the Session object, you can also store user preferences and then access that preference to choose what page to return to the user.

If the user has a low screen resolution, the example below specifies a text-only version of the page.

Example

```
<%If Session("screenres")="low" Then%>
This is the text version of the page
<%Else%>
This is the multimedia version of the page
<%End If%>
```

Remove Session Variables

All session variables are stored in Contents collection, To remove a session variable with the Remove method is possible.

Example

```
<%
If Session.Contents("age")<18 then
Session.Contents.Remove("sale")
End If
%>
```

Use the RemoveAll method to remove all variables in a session

```
<%
Session.Contents.RemoveAll()
%>
```

Loop Through the Contents Collection

All session variables are stored in Contents collection .To see what's stored in it, you can loop through the Contents collection

Example

```
<%
Session("username")="Donald Duck"
Session("age")=50
dim i
For Each i in Session.Contents
Response.Write(i & "<br />")
Next
%>
```

Loop Through the StaticObjects Collection

To see the values of all objects stored in the Session object you can loop through the StaticObjects collection

Example

```
<%
dim i
For Each i in Session.StaticObjects
Response.Write(i & "<br />")
Next
%>
```

5.3 CGI

Q8. What is a CGI Script ?

Ans :

Meaning

The Common Gateway Interface, or CGI, is a set of standards that define how information is exchanged between the web server and a custom script.

The CGI specs are currently maintained by the NCSA and NCSA defines CGI is as follows:

The Common Gateway Interface, or CGI, is a standard for external gateway programs to interface with information servers such as HTTP servers.

The current version is CGI/1.1 and CGI/1.2 is under progress.

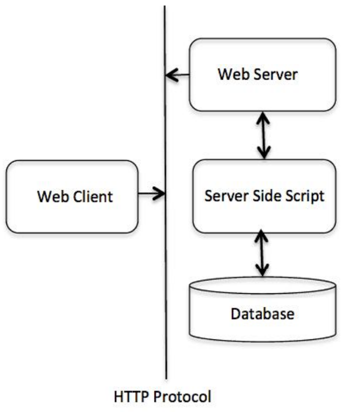
Web Browsing

To understand the concept of CGI, lets see what happens when we click a hyper link to browse a particular web page or URL.

- Your browser contacts the HTTP web server and demand for the URL ie. filename.
- Web Server will parse the URL and will look for the filename in if it finds that file then sends back to the browser otherwise sends an error message indicating that you have requested a wrong file.
- Web browser takes response from web server and displays either the received file or error message.

However, it is possible to set up the HTTP server so that whenever a file in a certain directory is requested that file is not sent back; instead it is executed as a program, and whatever that program outputs is sent back for your browser to display. This function is called the Common Gateway Interface or CGI and the programs are called CGI scripts. These CGI programs can be a PERL Script, Shell Script, C or C++ program etc.

CGI Architecture Diagram



Web Server Support & Configuration

Before you proceed with CGI Programming, make sure that your Web Server supports CGI and it is configured to handle CGI Programs. All the CGI Programs be executed by the HTTP server are kept in a pre-configured directory. This directory is called CGI Directory and by convention it is named as /cgi-bin. By convention PERL CGI files will have extention as.cgi.

First CGI Program

```
#!/usr/bin/perl
print"Content-type:text/html\r\n\r\n";
print'<html>';
print'<head>';
print'<title>Hello Word - First CGI Program</title>';
print'</head>';
print'<body>';
print'<h2>Hello Word! This is my first CGI program</h2>';
print'</body>';
print'</html>';
1;
```

Output

```
Hello Word! This is my first CGI program
```

HTTP Header

The line Content-type:text/html\r\n\r\n is part of HTTP header which is sent to the browser to understand the content. All the HTTP header will be in the following form

HTTP Field Name: Field Content

Example

Content-type:text/html\r\n\r\n

There are few other important HTTP headers which you will use frequently in your CGI Programming.

S.No.	Header & Description
1	**Content-type: String** A MIME string defining the format of the file being returned. Example is Content-type:text/html
2	**Expires: Date String** The date the information becomes invalid. This should be used by the browser to decide when a page needs to be refreshed. A valid date string should be in the format 01 Jan 1998 12:00:00 GMT.
3	**Location: URL String** The URL that should be returned instead of the URL requested. You can use this filed to redirect a request to any file.
4	**Last-modified: String** The date of last modification of the resource.
5	**Content-length: String** The length, in bytes, of the data being returned. The browser uses this value to report the estimated download time for a file.
6	**Set-Cookie: String** Set the cookie passed through the *string*

CGI Environment Variables

All the CGI program will have access to the following environment variables. These variables play an important role while writing any CGI program.

S.No.	Variable Name & Description
1	**CONTENT_TYPE** The data type of the content. Used when the client is sending attached content to the server. For example file upload etc.
2	**CONTENT_LENGTH** The length of the query information. It's available only for POST requests.
3	**HTTP_COOKIE** Return the set cookies in the form of key & value pair.
4	**HTTP_USER_AGENT** The User-Agent request-header field contains information about the user agent originating the request. Its name of the web browser.
5	**PATH_INFO** The path for the CGI script.
6	**QUERY_STRING** The URL-encoded information that is sent with GET method request.
7	**REMOTE_ADDR** The IP address of the remote host making the request. This can be useful for logging or for authentication purpose.

8	**REMOTE_HOST** The fully qualified name of the host making the request. If this information is not available then REMOTE_ADDR can be used to get IR address.
9	**REQUEST_METHOD** The method used to make the request. The most common methods are GET and POST.
10	**SCRIPT_FILENAME** The full path to the CGI script.
11	**SCRIPT_NAME** The name of the CGI script.
12	**SERVER_NAME** The server's hostname or IP Address.
13	**SERVER_SOFTWARE** The name and version of the software the server is running.

```
#!/usr/bin/perl
print"Content-type: text/html\n\n";
print"<font size=+1>Environment</font>\n";
foreach(sort keys %ENV)
{
print"<b>$_</b>: $ENV{$_}<br>\n";
}
1;
```

5.4 PERL

Q9. What is a Perl ?

Ans :

Perl is a general-purpose programming language originally developed for text manipulation and now used for a wide range of tasks including system administration, web development, network programming, GUI development, and more.

- Perl is a stable, cross platform programming language.
- Though Perl is not officially an acronym but few people used it as Practical Extraction and Report Language.
- It is used for mission critical projects in the public and private sectors.
- Perl is an Open Source software, licensed under its Artistic License, or the GNU General Public License (GPL).
- Perl was created by Larry Wall.
- Perl 1.0 was released to usenet's alt.comp.sources in 1987.
- At the time of writing this tutorial, the latest version of perl was 5.16.2.

- Perl is listed in the Oxford English Dictionary.
- PC Magazine announced Perl as the finalist for its 1998 Technical Excellence Award in the Development Tool category.

Features

- Perl takes the best features from other languages, such as C, awk, sed, sh, and BASIC, among others.
- Perls database integration interface DBI supports third-party databases including Oracle, Sybase, Postgres, MySQL and others.
- Perl works with HTML, XML, and other mark-up languages.
- Perl supports Unicode.
- Perl is Y2K compliant.
- Perl supports both procedural and object-oriented programming.
- Perl interfaces with external C/C++ libraries through XS or SWIG.
- Perl is extensible. There are over 20,000 third party modules available from the Comprehensive Perl Archive Network (CPAN).
- The Perl interpreter can be embedded into other systems.
- Perl and the Web
- Perl used to be the most popular web programming language due to its text manipulation capabilities and rapid development cycle.
- Perl is widely known as "the duct-tape of the Internet".
- Perl can handle encrypted Web data, including e-commerce transactions.

u Perl can be embedded into web servers to speed up processing by as much as 2000%.

- Perl's mod_perl allows the Apache web server to embed a Perl interpreter.
- Perl's DBI package makes web-database integration easy.

Perl is Interpreted

Perl is an interpreted language, which means that your code can be run as is, without a compilation stage that creates a non portable executable program.

Traditional compilers convert programs into machine language. When you run a Perl program, it's first compiled into a byte code, which is then converted (as the program runs) into machine instructions. So it is not quite the same as shells, or Tcl, which are strictly interpreted without an intermediate representation.

It is also not like most versions of C or C++, which are compiled directly into a machine dependent format. It is somewhere in between, along with Python and awk and Emacs .elc files.

5.4.1 Data types

Q10. Discuss about different data types in CGI PERL.

Ans : **(Imp.)**

Perl is a loosely typed language and there is no need to specify a type for your data while using in your program. The Perl interpreter will choose the type based on the context of the data itself.

Perl has three basic data types: scalars, arrays of scalars, and hashes of scalars, also known as associative arrays. Here is a little detail about these data types.

Sr.No.	Types & Description
1	**Scalar** Scalars are simple variables. They are preceded by a dollar sign ($). A scalar is either a number, a string, or a reference. A reference is actually an address of a variable, which we will see in the upcoming chapters.
2	**Arrays** Arrays are ordered lists of scalars that you access with a numeric index, which starts with 0. They are preceded by an "at" sign (@).
3	**Hashes** Hashes are unordered sets of key/value pairs that you access using the keys as subscripts. They are preceded by a percent sign (%).

Numeric Literals

Perl stores all the numbers internally as either signed integers or double-precision floating-point values. Numeric literals are specified in any of the following floating-point or integer formats “

Type	Value
Integer	1234
Negative integer	-100
Floating point	2000
Scientific notation	16.12E14
Hexadecimal	0xffff
Octal	0577

String Literals

Strings are sequences of characters. They are usually alphanumeric values delimited by either single (‘) or double (“) quotes. They work much like UNIX shell quotes where you can use single quoted strings and double quoted strings.

Double-quoted string literals allow variable interpolation, and single-quoted strings are not. There are certain characters when they are proceeded by a back slash, have special meaning and they are used to represent like newline (\n) or tab (\t).

You can embed newlines or any of the following Escape sequences directly in your double quoted strings –

Escape sequence	Meaning
\\	Backslash
\'	Single quote
\"	Double quote
\a	Alert or bell
\b	Backspace
\f	Form feed
\n	Newline
\r	Carriage return
\t	Horizontal tab
\v	Vertical tab
\0nn	Creates Octal formatted numbers
\xnn	Creates Hexideciamal formatted numbers
\cX	Controls characters, x may be any character
\u	Forces next character to uppercase
\l	Forces next character to lowercase
\U	Forces all following characters to uppercase
\L	Forces all following characters to lowercase
\Q	Backslash all following non-alphanumeric characters
\E	End \U, \L, or \Q

Example

Let’s see again how strings behave with single quotation and double quotation. Here we will use string escapes mentioned in the above table and will make use of the scalar variable to assign string values.

```
#!/usr/bin/perl
# This is case of interpolation.
$str = ”Welcome to \ntutorialspoint.com!”;
print”$str\n”;
# This is case of non-interpolation.
$str = ’Welcome to \ntutorialspoint.com!’;
print”$str\n”;
# Only W will become upper case.
$str = ”\uwelcome to tutorialspoint.com!”;
print”$str\n”;
```

```
# Whole line will become capital.
$str = "\UWelcome to tutorialspoint.com!";
print"$str\n";
# A portion of line will become capital.
$str = "Welcome to \Ututorialspoint\E.com!";
print"$str\n";
# Backsalash non alpha-numeric including spaces.
$str = "\QWelcome to tutorialspoint's family";
print"$str\n";
```

5.4.2 Regular Expressions

Q11. What are Regular Expressions in PERL ?

Ans :

A regular expression is a string of characters that defines the pattern or patterns you are viewing. The syntax of regular expressions in Perl is very similar to what you will find within other regular expression.supporting programs, such as sed, grep, and awk.

The basic method for applying a regular expression is to use the pattern binding operators =~ and !~. The first operator is a test and assignment operator.

There are three regular expression operators within Perl.

- Match Regular Expression - m//
- Substitute Regular Expression - s///
- Transliterate Regular Expression - tr///

The forward slashes in each case act as delimiters for the regular expression (regex) that you are specifying. If you are comfortable with any other delimiter, then you can use in place of forward slash.

(i) The Match Operator

The match operator, m//, is used to match a string or statement to a regular expression. For example, to match the character sequence "foo" against the scalar $bar, you might use a statement like this -

```
#!/usr/bin/perl
$bar = "This is foo and again foo";
if($bar = ~/foo/)
{
print"First time is matching\n";
}else{
print"First time is not matching\n";
}
$bar = "foo";
if($bar = ~/foo/)
```

```
{
print"Second time is matching\n";
}else{
print"Second time is not matching\n";
}
```

When above program is executed, it produces the following result "

```
First time is matching
Second time is matching
```

The m// actually works in the same fashion as the q// operator series.you can use any combination of naturally matching characters to act as delimiters for the expression. For example, m{}, m(), and m>< are all valid. So above example can be re-written as follows -

```
#!/usr/bin/perl
$bar ="This is foo and again foo";
if($bar =~ m[foo])
{
print"First time is matching\n";
}else{
print"First time is not matching\n";
}
$bar ="foo";
if($bar =~ m{foo})
{
print"Second time is matching\n";
}
else
{
print"Second time is not matching\n";
}
```

You can omit m from m// if the delimiters are forward slashes, but for all other delimiters you must use the m prefix.

Note that the entire match expression, that is the expression on the left of =~ or !~ and the match operator, returns true (in a scalar context) if the expression matches. Therefore the statement "

```
$true =($foo =~ m/foo/);
```

will set $true to 1 if $foo matches the regex, or 0 if the match fails. In a list context, the match returns the contents of any grouped expressions. For example, when extracting the hours, minutes, and seconds from a time string, we can use "

```
my($hours, $minutes, $seconds)=($time =~ m/(\d+):(\d+):(\d+)/);
```

(ii) Match Operator Modifiers

The match operator supports its own set of modifiers. The /g modifier allows for global matching. The /i modifier will make the match case insensitive. Here is the complete list of modifiers.

Sr.No.	Modifier & Description
1	**i -** Makes the match case insensitive.
2	**m -** Specifies that if the string has newline or carriage return characters, the ^ and $ operators will now match against a newline boundary, instead of a string boundary.
3	**o -** Evaluates the expression only once.
4	**s -** Allows use of . to match a newline character.
5	**x -** Allows you to use white space in the expression for clarity.
6	**g -** Globally finds all matches.
7	**cg -** Allows the search to continue even after a global match fails.

(iii) Matching Only Once

There is also a simpler version of the match operator - the ?PATTERN? operator. This is basically identical to the m// operator except that it only matches once within the string you are searching between each call to reset.

For example, you can use this to get the first and last elements within a list “

```
#!/usr/bin/perl
@list= qw/food foosball subeo footnote terfoot canic footbrdige/;
foreach(@list){
    $first = $1 if/(foo.*?)/;
    $last = $1 if/(foo.*)/;
}
print"First: $first, Last: $last\n";
```

When above program is executed, it produces the following result “

```
First: food, Last: footbrdige
```

(iv) Regular Expression Variables

Regular expression variables include $, which contains whatever the last grouping match matched; $&, which contains the entire matched string; $‘, which contains everything before the matched string; and $’, which contains everything after the matched string. Following code demonstrates the result -

```
#!/usr/bin/perl
$string =”The food is in the salad bar”;
$string =~ m/foo/;
print”Before: $‘\n”;
print”Matched: $&\n”;
print”After: $’\n”;
```

When above program is executed, it produces the following result “

```
Before: The
Matched: foo
After: d is in the salad bar
```

(v) The Substitution Operator

The substitution operator, s///, is really just an extension of the match operator that allows you to replace the text matched with some new text. The basic form of the operator is “

s/PATTERN/REPLACEMENT/;

The PATTERN is the regular expression for the text that we are looking for. The REPLACEMENT is a specification for the text or regular expression that we want to use to replace the found text with. For example, we can replace all occurrences of dog with cat using the following regular expression “

```
#/user/bin/perl
$string = ”The cat sat on the mat”;
$string = ~ s/cat/dog/;
print”$string\n”;
```

When above program is executed, it produces the following result -

```
The dog sat on the mat
```

(vi) The Translation Operator

Translation is similar, but not identical, to the principles of substitution, but unlike substitution, translation (or transliteration) does not use regular expressions for its search on replacement values. The translation operators are -

```
tr/SEARCHLIST/REPLACEMENTLIST/cds
y/SEARCHLIST/REPLACEMENTLIST/cds
```

The translation replaces all occurrences of the characters in SEARCHLIST with the corresponding characters in REPLACEMENTLIST. For example, using the “The cat sat on the mat.” string we have been using in this chapter -

```
#/user/bin/perl
$string = ’The cat sat on the mat’;
$string = ~ tr/a/o/;
print”$string\n”;
```

When above program is executed, it produces the following result “

```
The cot sot on the mot.
```

Standard Perl ranges can also be used, allowing you to specify ranges of characters either by letter or numerical value. To change the case of the string, you might use the following syntax in place of the uc function.

```
$string = ~ tr/a-z/A-Z/;
```

(vii) Translation Operator Modifiers

Following is the list of operators related to translation.

Sr.No.	Modifier & Description
1	**c -** Complements SEARCHLIST.
2	**d -** Deletes found but unreplaced characters.
3	**s -** Squashes duplicate replaced characters.

The /d modifier deletes the characters matching SEARCHLIST that do not have a corresponding entry in REPLACEMENTLIST. For example -

```
#!/usr/bin/perl
$string = ’the cat sat on the mat.’;
$string = ~ tr/a-z/b/d;
print”$string\n”;
```

When above program is executed, it produces the following result “

```
b b   b.
```

The last modifier, /s, removes the duplicate sequences of characters that were replaced, so “

```
#!/usr/bin/perl
$string = ’food’;
$string = ’food’;
$string = ~ tr/a-z/a-z/s;
print”$string\n”;
```

When above program is executed, it produces the following result -

```
fod
```

(viii) More Complex Regular Expressions

You don’t just have to match on fixed strings. In fact, you can match on just about anything you could dream of by using more complex regular expressions. Here’s a quick cheat sheet “

Following table lists the regular expression syntax that is available in Python.

Sr.No.	Description
1	**^ -** Matches beginning of line.
2	**$ -** Matches end of line.
3	**. -** Matches any single character except newline. Using m option allows it to match newline as well.
4	**[...] -** Matches any single character in brackets.
5	**[^...] -** Matches any single character not in brackets.
6	* - Matches 0 or more occurrences of preceding expression.
7	**+ -** Matches 1 or more occurrence of preceding expression.
8	**? -** Matches 0 or 1 occurrence of preceding expression.
9	**{ n} -** Matches exactly n number of occurrences of preceding expression.
10	**{ n,} -** Matches n or more occurrences of preceding expression.
11	**{ n, m} -** Matches at least n and at most m occurrences of preceding expression.
12	**a| b -** Matches either a or b.
13	**\w -** Matches word characters.
14	**\W -** Matches nonword characters.
15	**\s -** Matches whitespace. Equivalent to [\t\n\r\f].
16	**\S -** Matches nonwhitespace.
17	**\d -** Matches digits. Equivalent to [0-9].
18	**\D -** Matches nondigits.
19	**\A -** Matches beginning of string.
20	**\Z -** Matches end of string. If a newline exists, it matches just before newline.
21	**\z -** Matches end of string.
22	**\G -** Matches point where last match finished.
23	**\b -** Matches word boundaries when outside brackets. Matches backspace (0x08) when inside brackets.
24	**\B -** Matches nonword boundaries.
25	**\n, \t, etc. -** Matches newlines, carriage returns, tabs, etc.
26	**\1...\9 -** Matches nth grouped subexpression.
27	**\10 -** Matches nth grouped subexpression if it matched already. Otherwise refers to the octal representation of a character code.
28	**[aeiou] -** Matches a single character in the given set
29	**[^aeiou] -** Matches a single character outside the given set

The ^ metacharacter matches the beginning of the string and the $ metasymbol matches the end of the string. Here are some brief examples.

```
# nothing in the string (start and end are adjacent)
/ ^ $/
# a three digits, each followed by a white space
# character (eg "3 4 5 ")
/(\d\s){3}/
# matches a string in which every
# odd-numbered letter is a (eg "abacadaf")
/(a.)+/
# string starts with one or more digits
/ ^ \d+/
# string that ends with one or more digits
/\d+$/
```

Lets have a look at another example.

```
#!/usr/bin/perl
$string ="Cats go Catatonic\nWhen given Catnip";
($start)=($string =~/\A(.*?)/);
@lines= $string =~/ ^ (.*?)/gm;
print"First word: $start\n","Line starts: @lines\n";
```

When above program is executed, it produces the following result -

```
First word: Cats
Line starts: Cats When
```

(ix) Matching Boundaries

The \b matches at any word boundary, as defined by the difference between the \w class and the \W class. Because \w includes the characters for a word, and \W the opposite, this normally means the termination of a word. The \Bassertion matches any position that is not a word boundary.

Example

```
/\bcat\b/# Matches 'the cat sat' but not 'cat on the mat'
/\Bcat\B/# Matches 'verification' but not 'the cat on the mat'
/\bcat\B/# Matches 'catatonic' but not 'polecat'
/\Bcat\b/# Matches 'polecat' but not 'catatonic'
```

(x) Selecting Alternatives

The | character is just like the standard or bitwise OR within Perl. It specifies alternate matches within a regular expression or group. For example, to match "cat" or "dog" in an expression, you might use this -

```
if($string =~/cat|dog/)
```

You can group individual elements of an expression together in order to support complex matches. Searching for two people's names could be achieved with two separate tests, like this "

```
if(($string =~/MartinBrown/)||($string =~/SharonBrown/))
```

This could be written as follows :

```
if($string =~/(Martin|Sharon)Brown/)
```

(xi) Grouping Matching

From a regular-expression point of view, there is no difference between except, perhaps, that the former is slightly clearer.

```
$string =~/(\S+)\s+(\S+)/;
```

and

```
$string =~/\S+\s+\S+/;
```

However, the benefit of grouping is that it allows us to extract a sequence from a regular expression. Groupings are returned as a list in the order in which they appear in the original. For example, in the following fragment we have pulled out the hours, minutes, and seconds from a string.

```
my($hours, $minutes, $seconds)=($time =~ m/(\d+):(\d+):(\d+)/);
```

As well as this direct method, matched groups are also available within the special $x variables, where x is the number of the group within the regular expression. We could therefore rewrite the preceding example as follows -

```
#!/usr/bin/perl
$time =”12:05:30";
$time =~ m/(\d+):(\d+):(\d+)/;
my($hours, $minutes, $seconds)=($1, $2, $3);
print”Hours : $hours, Minutes: $minutes, Second: $seconds\n”;
```

When above program is executed, it produces the following result “

```
Hours : 12, Minutes: 05, Second: 30
```

When groups are used in substitution expressions, the $x syntax can be used in the replacement text. Thus, we could reformat a date string using this -

```
#!/usr/bin/perl
$date =’03/26/1999';
$date =~ s#(\d+)/(\d+)/(\d+)#$3/$1/$2#;
print”$date\n”;
```

When above program is executed, it produces the following result -

```
1999/03/26
```

(xii) The \G Assertion

The \G assertion allows you to continue searching from the point where the last match occurred. For example, in the following code, we have used \G so that we can search to the correct position and then extract some information, without having to create a more complex, single regular expression “

```
#!/usr/bin/perl
$string =”The time is: 12:31:02 on 4/12/00";
$string =~/:\s+/g;
($time)=($string =~/\G(\d+:\d+:\d+)/);
$string =~/.+\s+/g;
```

($date)=($string =~ m{\G(\d+/\d+/\d+)});

print"Time: $time, Date: $date\n";

When above program is executed, it produces the following result -

Time: 12:31:02, Date: 4/12/00

The \G assertion is actually just the metasymbol equivalent of the pos function, so between regular expression calls you can continue to use pos, and even modify the value of pos (and therefore \G) by using pos as an lvalue subroutine.

(xiii) Literal Characters

Sr.No.	Example & Description
1	**Perl -** Match "Perl".

(xiv) Character class

Sr.No.	Example & Description
1	**[Pp]ython** Matches "Python" or "python"
2	**rub[ye]** Matches "ruby" or "rube"
3	**[aeiou]** Matches any one lowercase vowel
4	**[0-9]** Matches any digit; same as [0123456789]
5	**[a-z]** Matches any lowercase ASCII letter
6	**[A-Z]** Matches any uppercase ASCII letter
7	**[a-zA-Z0-9]** Matches any of the above
8	**[^ aeiou]** Matches anything other than a lowercase vowel
9	**[^ 0-9]** Matches anything other than a digit

(xv)Special Character Class

Sr.No.	Example & Description
1	**.** Matches any character except newline
2	**\d** Matches a digit: [0-9]
3	**\D** Matches a nondigit: [^ 0-9]
4	**\s** Matches a whitespace character: [\t\r\n\f]
5	**\S** Matches nonwhitespace: [^ \t\r\n\f]
6	**\w** Matches a single word character: [A-Za-z0-9_]
7	**\W** Matches a nonword character: [^ A-Za-z0-9_]

(xvi) Repetition Cases

Sr.No.	Example & Description
1	**ruby?** Matches "rub" or "ruby": the y is optional
2	**ruby*** Matches "rub" plus 0 or more ys
3	**ruby+** Matches "rub" plus 1 or more ys
4	**\d{3}** Matches exactly 3 digits
5	**\d{3,}** Matches 3 or more digits
6.	**\d{3,5}** Matches 3, 4, or 5 digits

5.4.3 String Processing with Perl

Q12. Explain String Functions in Processing PERL?

Ans :

Many predefined functions that are useful in information manipulations. In Perl, you use a function as an expression. As soon as Perl sees a function call in the script, the function line is executed. Perl functions can be grouped as the following :

- String functions
- Numeric functions
- Array functions
- Time functions
- Pattern Matching Functions

String functions : Strings are a combination of characters. The string has no limit and it can be of any size and may contain any characters, symbols or words. In Perl, it is defined by placing them in double quotes or with the ‘q’ functions.

- **Chop - Removes the last character from a string or array of strings.**

  chop STRING

- **eval- Evaluates perl code, then executes it.**

  eval STRING

  Any errors are returned in the @a variable.

- **Index -** This function returns the position of the first occurance of the specified SEARCH string. If POSITION is specified, the occurance at or after the position is returned. The value -1 is returned if the SEARCH string is not found.

  rindex STRING,SEARCH,POSITION

  rindex STRING,SEARCH

- **length- Returns the length of the string in bytes.**

  length STRING

- **lc - Converts all characters in the string to lower case.**

  lc Str

- **lcfirst - Takes a string and retruns it with the first character in lower case.**

  lcfirst Str1

- **quotemeta - rindex**

  This function returns the position of the last occurance of the specified SEARCH string. If POSITION is specified, the occurance at or before the position is returned. The value -1 is returned if the SEARCH string is not found.

  rindex STRING,SEARCH,POSITION

  rindex STRING,SEARCH

- **substr - This function supports three sets of passed values as follows:**

  substr (STRING,OFFSET,LEN,REPLACEMENT)

  substr (STRING,OFFSET,LEN)

  substr (STRING,OFFSET)

  The function:

  substr (STRING,OFFSET)

returns all characters in the string after the designated offset from the start of the passed string. The function:

substr (STRING,OFFSET,LEN)

returns all characters in the string after the designated offset from the start of the passed string up to the number of characters designated by LEN. The function:

substr (STRING,OFFSET,LEN,REPLACEMENT)

Replaces the part of the string beginning at OFFSET of the length LEN with the REPLACEMENT string.

- **Uc - Converts all characters in the string to upper case.**

  uc Str

- **ucfirst - Takes a string and retruns it with the first character in upper case.**

  ucfirst Str1

5.4.4 Serve side includes, coockies

Q13. What are the different functions in ASP ?

Ans :

Functions and procedures provide a way to create re-usable modules of programming code and avoid repeating the same block of code every time you do the same task.

The ASP pages are executed from top to bottom and if you don't have any functions/procedures in your ASP page, the ASP parsing engine simply processes your entire file from the beginning to the end.

ASP (VBScript) functions and procedures, on the other hand, are executed only when called, from within the ASP code (they might be called from another function/procedure or directly from inline code).

The ones returning a value start with Function keyword and end with End Function. The second type of function in VBScript doesn't return a value starts with the Sub keyword and ends with End Sub, defining the function as a subroutine.

Syntax

```
Function function_name(arguments)
     Statements
End Function
```

Example

```
Function MultNum(iNum1, iNum2)

MultNum = iNum1 * iNum2

End Function
```

The function above gets 2 arguments and returns them multiplied.

Example

```
<%

Function Factorial(ByVal intNumber)
```

```
If intNumber <= 1 Then
Factorial = 1
Else
Factorial = intNumber * Factorial(intNumber - 1)
End If
End Function
%>
'And here we test the function with 2 values 5 and then 10
<%
Response.Write ("Factorial of 5 is " & Factorial(5))
Response.Write ("<br>")
Response.Write ("Factorial of 7 is " & Factorial(7))
%>
```

Cookies in ASP

A cookie is a small text file that is stored on clientmachine that is embeded by the server. Each time the same computer requests a page with a browser, it will send the cookie too. To store information specific to a visitor of your website, ASP Cookies are used.

Create a Cookie

To create cookies,the "Response.Cookies" command is used . BEFORE the <html> tag, the Response.Cookies command must appear. We will create a cookie named and assign the value to it

Example

```
<%
dim numvisits
response.cookies("CountVisitors").Expires
=date+365
numvisits=request.cookies("CountVisitors")
if numvisits="" then
response.cookies("CountVisitors")=1
response.write("Welcome! This is the first time you are visiting this Web page.")
else
response.cookies("CountVisitors")= numvisits+1
response.write("You have visited this ")
response.write("Web page " & numvisits)
```

```
if numvisits=1 then
response.write " time before!"
else
response.write " times before!"
end if
end if
%>
```

Cookie Value

To retrieve a cookie value,the "Request. Cookies" command is used. We retrieve the value of the cookie name and display it on a page

Example

```
<%
gname=Request.Cookies("goodname")
response.write("Goodname=" & gname)
%>
```

A Cookie with Keys

We say that the cookie has Keys if a cookie contains a collection of multiple values. We will create a cookie collection named. The cookie has Keys that contains information about a user

Example

```
<%
Response.Cookies("visitor")("firstname")="John"
Response.Cookies("visitor")("lastname")="Smith"
Response.Cookies("visitor")("country")="Norway"
Response.Cookies("visitor")("age")="25"
%>
```

What if a Browser Does NOT Support Cookies?

You will have to use other methods to pass information from one page to another in your application if your application deals with browsers that do not support cookies. There are two methods of doing this,

1. Add parameters to a URL.
2. Use a form

Add parameters to a URL

You can add parameters to a URL

<a href="welcome.asp?fname=John&lname=Smith"> Go to Welcome Page</a>

And retrieve the values in the web page

```
<%
fname=Request.querystring("fname")
lname=Request.querystring("lname")
response.write("<p>Hello " & fname & " " & lname & "!</p>")
response.write("<p>Welcome to my Web site!</p>")
%>
```

Use a form

You can use a form. When the user clicks on the Submit button, the form passes the user input.

```
<form method="post" action="welcome.asp">
First Name: <input type="text" name="fname" value="">
Last Name: <input type="text" name="lname" value="">
<input type="submit" value="Submit">
</form>
```

Retrieve the values in the web page

```
<%
fname=Request.form("fname")
lname=Request.form("lname")
response.write("<p>Hello " & fname & " " & lname & "!</p>")
response.write("<p>Welcome to my Web site!</p>")
%>
```

5.4.5 ASP Server Object

Q14. Explain the properties and methods of server objects ?

Ans : **(Imp.)**

Server Object

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

Properties

Property	Description
ScriptTimeout	Sets or returns the maximum number of seconds a script can run before it is terminated

Methods

Method	Description
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

Example to Check when this file was last modified

```
<html>
<body>
<%
Set fs = Server.CreateObject("Scripting.FileSystemObject")
Set rs = fs.GetFile(Server.MapPath("demo_lastmodified.asp"))
modified = rs.DateLastModified
%>
This file was last modified on: <%response.write(modified)
Set rs = Nothing
Set fs = Nothing
%>
</body>
</html>
```

Application Object

A group of ASP files may be called an application on the Web. Some purpose can be achieved when the ASP file work together. In ASP, the Application object is used to tie these files together.

For storing and accessing variables from any page, just like the Session object,the Application object is used . The difference is that in Sessions there is one Session object for EACH user while ALL users share one Application object.

In the Particular application (like database connection information), the Application object should hold information that will be used by many pages. This means it is possible access the information from any page. In an Application object, you can change the information in one place and the changes will automatically be reflected on all pages.

Store and Retrieve Application Variables

Application variables can be accessed and changed by any page. You can create Application variables

```
<script language="vbscript" runat="server">
Sub Application_OnStart
application("vartime")=""
application("users")=1
End Sub
</script>
```

Application variables can be accessed like this

```
<%
Response.Write(Application("users"))
%>
```

active connections

Loop Through the Contents Collection

All application variables will be stored in Contents collection. To see what's stored in it ,you can loop through the Contents collection,

```
<%
dim i
For Each i in Application.Contents
Response.Write(i & "<br />")
Next
%>
```

You can use the Count property if you do not know the number of items in the Contents collection,

```
<%
dim i
dim j
j=Application.Contents.Count
For i=1 to j
Response.Write(Application.Contents(i) & "<br />")
Next
%>
```

Lock and Unlock

Using "Lock" method, you can lock an application. The users cannot change the Application variables(other than the one currently accessing it),when an application is locked. Using "Unlock" method, you can unlock an application. This method is used to remove the lock from the Application variable.

Example

```
<%
Application.Lock
'do some application object operations
Application.Unlock
%>
```

5.5 Extensible Markup Language (XML)

Q15. Explain about XML.

Ans : **(Imp.)**

In the 1970's, GML was used to describe a way of marking up technical documents with structural tags.

Goldfarb invented the term "mark-up language" to make better use of the initials and it became the Standard Generalised Markup Language. In 1986, SGML became an international standard for defining the markup languages. It was used to create other languages, including HTML, which is very popular for its use on the web.

While on one hand SGML is very effective but complex, on the other, HTML is very easy, but limited to a fixed set of tags. This situation raised the need for a language that was as effective as SGML and at the same time as simple as HTML. This gap has now been filled by XML.

The development of XML started in 1996 at Sun Microsystems. The World Wide Web Consortium also contributes to the creation and development of the standard for XML.

If you are familiar with HTML, you have some concept of markup language. If you write a plain text file, it is composed of simple ASCII characters. When a program (like notepad) is used to display the file, all characters in the text file will be displayed using the same font size, type, and boldness. There are no special characteristics to present such type of file.

Markup languages, like HTML or XML, allow special markup to be embedded with the remaining text that will enable the program that displays the file to determine how to show the text. In this way, special text like paragraph may be justify, have a larger and bolder font, or specific display colors may be set. Also additional elements may be added to the file such as numbered lists andtables.

Markup languages use different elements to set aside one area of content from other content. The display of these elements (e.g. color, size, and font type) may be determined within the markup file or outside the file using a style sheet.

Normally, there is a predetermined set of display characteristics (default) for each element which may be modified locally or using style sheets. Authors are encouraged to separate the determination of display characteristics (style) from the markup file. This makes management of display style much easier but the separation is not required.

XML Definition

XML (eXtensible Markup Language) is a meta-language; that is, it is a language in which other languages are created. In XML, data is "marked up" with tags, similar to HTML tags. In fact, the latest version of HTML, called XHTML, is an XML-based language, which means that XHTML follows the syntax rules ofXML.

XML is used to store data or information. This data might be intended to be by read by people or by machines. It can be highly structured data such as data typically stored in databases or spreadsheets, or loosely structured data, such as data stored in letters or manuals.

Structured information contains any type of content (words, pictures, etc.) and some indication of what role that content plays (for example, content in a section heading has a different significance from content in a footnote, which means something different than content in a figure caption or content in a database table, etc.). Almost all documents have some structure.

A markup language is a mechanism to identify the document structures. The XML is used to define a standard way to add markup to documents.

Q16. State the benefits and goals of XML benefits.

Ans : **(Imp.)**

Initially XML received a lot of excitement, which has now died down some. This isn't because XML is not as useful, but rather because it doesn't provide the Wow! factor that other technologies, such as HTML do. When you write an HTML document, you see a nicely formatted page in a browser - instant gratification. When you write an XML document, you see an XML document not so exciting. However, with a little more effort, you can make that XML document sing!

- XML Holds Data, NothingMore
- XML Separates Structure fromFormatting
- XML Separates Structure fromFormatting
- XML Promotes DataSharing
- XML isHuman-Readable
- XML is Free

Goals

XML was initially "developed by a World Wide Web Consortium Generic SGML Editorial Review Board formed under the auspices of the W3 Consortium in 1996 and chaired by Jon Bosak of Sun Microsystems, with the active participation of a Generic SGML Working Group also organized by theW3C."

The Extensible Markup Language (XML) became a W3C Recommendation 10.February 1998.

1. To use XML over the Internet, users must be able to view XML documents as quickly and easily as HTML documents. In practice, this will only be possible when XML browsers are as robust and widely available as HTML browsers, but the principle remains.
2. XML support a wide variety of applications. XML should be beneficial to a wide variety of diverse applications: browsing, authoring, content analysis, etc. Although the first focus is on serving structured documents over theweb.
3. XML should be compatible with SGML. Most of the people involved in the XML effort come from organizations that have a large, in some cases staggering, amount of material in SGML.
4. Standards while solving the relatively new problem of sending richly structured documents over the web.
5. It should be easy to write programs that process XML documents. The informal way of expressing this goal while the spec was being developed was that it ought to take about two weeks for a competent computer science graduate student to build a program that can process XMLdocuments.
6. Many optional features in XML is to be kept to an absolute minimum, ideally zero. Optional features inevitably raise compatibility problems when users want to share documents and sometimes lead to confusion andfrustration.
7. XML documents should be human-legible and reasonably clear. If you don't have an XML browser and you've received a hunk of XML from somewhere, you ought to be able to look at it in your favorite text editor and actually figure out what the content means.
8. The XML design should be prepared quickly. Standards efforts are disreputably slow. XML was needed immediately and was developed as quickly aspossible.

9. The design of XML shall be formal and concise. In many ways a notoriously to rule 4, it essentially means that XML must be expressed in Extended Backus-Naur Form (EBNF) and must be amenable to modern compiler tools andtechniques.

10. There are a number of technical reasons why the SGML grammar cannot be expressed in Extended Backus-Naur Form (EBNF) writing a proper SGML parser requires handling a variety of rarely used and difficult to parse language features. XML doesnot.

11. It is easy to create XML documents. Although there will eventually be complicated editors to create and edit XML content, they won't appear immediately. In the interim, it must be possible to create XML documents in other ways: directly in a text editor, with simple shell and Perl scripts,etc.

12. Shortness in XML markup is of minimal importance. Several SGML language features were designed to minimize the amount of typing required to manually key in SGML documents. These features are not supported in XML. From an abstract point of view, these documents are indistinguishable from their more fully specified forms, but supporting these features adds a considerable burden to the SGML parser. In addition, most modern editors offer better facilities to define shortcuts when enteringtext.

5.6 XML Document Type Definition

Q17. Discuss about different types DTD Element.

Ans : **(Imp.)**

Markup languages require a Document Type Definition which defines the elements that are allowed in the document. The DTD also defines how elements may be used with relationship to each other. It define how many and which elements may be included inside another element. The DTD is a text file written in a specific format to define the document.

The DTD is based on the Standardized Generalized Markup Language (SGML). SGML is the parent language of all markup languages. Although XML may use a DTD, it is not required for those documents that are considered "well formed". A well formed document follows a set of rules for XML and this subject is discussed in more detail later. The DTD also defines another characteristic of the element such as whether or not it requires a beginning or ending tag along with various possible attributes of eachelement.

The purpose of a DTD is to define the structure of an XML document. It defines the structure with a list of legal elements: A DTD can be declared inline inside an XML document, or as an external reference. With a DTD, each of your XML files can carry a description of its own format. With a DTD, independent groups of people can agree to use a standard DTD for interchanging data. XML application can use a standard DTD to verify that the data we receive from the outside world is valid. We can also use a DTD to verify our own data.

DTD - Elements

In a DTD, elements are declared with an ELEMENT declaration.

<! ELEMENT element-name category> or <! ELEMENT element-name (element-content)>

Elements with only parsed character data are declared with #PCDATA inside parentheses:

<! ELEMENT element-name (#PCDATA)> Example

<! ELEMENT from (#PCDATA)>

Elements with one or more children are declared with the name of the children elements inside parentheses:

<! ELEMENT element-name (child1)> or <! ELEMENT element-name (child1, child2...)>

Example

<! ELEMENT note (to, from, heading, body)>

When children are declared in a sequence separated by commas, the children must appear in the same sequence in the document. In a full declaration, the children must also be declared, and the children can also have children. The full declaration of the "note" element is :

```
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
<!DOCTYPE note [
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
```

Internal DTD Declaration

If the DTD is declared inside the XML file, it should be wrapped in a DOCTYPE definition with the following syntax:

```
<!DOCTYPE root-element [element-declarations]>
<?xml version=”1.0"?>
<!DOCTYPE note [
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
```

```
<body>Don’t forget me this weekend</body>
</note>
```

External DTD Declaration

If the DTD is declared in an external file, it should be wrapped in a DOCTYPE definition with the following syntax:

```
<!DOCTYPE root-element SYSTEM “filename”>

<?xml version=”1.0"?>

<!DOCTYPE note SYSTEM “note.dtd”>

<note>

<to>Tove</to>

<from>Jani</from>

<heading>Reminder</heading>

<body>Don’t forget me this weekend!</body>

</note>
```

And this is the file “note.dtd” which contains the DTD:

```
<!ELEMENT note (to,from,heading,body)>

<!ELEMENT to (#PCDATA)>

<!ELEMENT from (#PCDATA)>

<!ELEMENT heading (#PCDATA)>

<!ELEMENT body (#PCDATA)>
```

5.7 XML PARSER

Q18. What is XML Parsers ?

Ans :

XML parsers are libraries used for checking and validating XML document schema.

DOM Parser

- DOM (Document Object Model) provides a programming interface for manipulating XML documents.
- DOM includes a set of objects and interfaces, which represent the content and structure of an XML document and enables a program to traverse XML tree structure.
- DOM allows to create new XML document with programming interfaces, for example, we can create XML document through ASP.net.
- SAX Parser
- SAX stands for Simple API for XML.
- API (Application Programming interface) allows programmer to read and write XML data.

- SAX parser is based on events.
- SAX Parser follows a push model which allows sequential access.

SAX Parser vs DOM Parser

SAX Parser	DOM Parser
Instead of creating internal structure in the document, it takes the occurrences of components as events.	DOM parser creates a tree structure in memory from an input document and waits for client request.
SAX Parser allows to extract the information in the document according to the users interest.	User can not extract the information of his interest.
SAX parser is space efficient.	DOM Parser is space inefficient.
Users have to create their own data structures.	DOM parser allows user to access any part of the document and modify the DOM tree.

Q19. Explain how to Access XML Data in ASP.

Ans : **(Imp.)**

The variety of ways that XML data can be accessed from an ASP page. Before we delve into these, let's look at some of the various ways XML data can be stored. Some of these storage methods include:

- **Simple XML**-formatted text files stored in file system
- Data stored in RDBMS that you want to convert to and from XML only when processing the database information
- XML data built dynamically in memory during processing.
- We will consider the following cases with code samples of each:
- Accessing stand alone XML files - You can access and manipulate XML files already presented on the server
- Creating XML files from ground up in memory - You can create entire XML document in memory and then manipulate it.
- **Sending XML data to client** - Once done you can send the result of your processing to the client as XML document or XML Data Islands
- **Storing XML data** - The data send by client in the form of HTML forms or XML can be processed and stored at server end in the form of XML files. The data can also be saved in Response object directly (which is then send back to the client).

Document Object Model

Document Object Model or DOM plays an important role in accessing XML data. DOM is governed by W3C and hence is a standard for accessing XML documents. DOM views XML document as a tree of nodes. Each node of the tree can be accessed randomly. The main advantage of DOM is that it provides all the functionality in an Object based way which make it very easy to use. Microsoft has created XML parser based on DOM known as MSXML. We will be using this component to access the XML documents. For working with examples presented in tis article you must have Internet Explorer 5+ installed on your machine(this automatically installs MSXML.dll on the machine). It is assumed that you are familier with XML DOM objects and how to use them via some scripting language.

Accessing stand alone XML files

Your XML data can be stored in separate disk files. In this case we need to perform following steps:

- Create an object instance of the Microsoft.XMLDOM object
- Set the async property of document object
- Load the XML file
- Check for any errors
- Manipulate the XML file

Let's look at a code example!

Step 1: Create object of Microsoft.XMLDOM

```
Dim mydoc
Set mydoc=Server.CreateObject("Microsoft.XMLDOM")
```

Step 2: Set async property

If you set this property to false then DOM will prevent access to the document while it is in process of change or update. If you set this property to true then you need to use onreadystatechangeevent and check the readyState property for complete(4).

```
mydoc.async=false
```

Step 3: Load XML file

```
mydoc.load("file_path")
```

Note: With IIS 5.0 you can also load XML data directly from ASP Request object if it is being send from client as follows :

```
'Load the specified XML file
mydoc.load(Request)
```

Step 4: Check for any errors

Here, you check the errorcode property and decide the success or failure of the operation.

```
    if mydoc.parseError.errorcode< >0 then
    error handling code
else
    proceed
end if
```

Now, you can use various XMLDOM methods and properties to manipulate the document.

Creating XML files from ground up

Some times you may want to build entire XML document in memory by yourself. This can be the case if your data is stored in RDBMS and you want to convert in into XML only for processing. You can create XML in memory in two ways:

- Using XMLDOM methods like createElement() and appendChild()
- Directly loading XML data in the form of a string

Let's look at using the XMLDOM first :

```
Dim mydoc,myelement
Set mydoc=Server.CreateObject("Microsoft.XMLDOM")
set myelement=mydoc.createElement("rootelement")
mydoc.appendChild(myelement)
```

The code for loading the XML data from a string appears as follows:

```
Dim mydoc,strXML
Set mydoc=Server.CreateObject("Microsoft.XMLDOM")
strXML="<book><author>author1</author><title>title1</title></book>"
mydoc.loadXML(strXML)
```

Sending XML data to client

We will consider following three cases :

1. You have generated a full XML document and sending it to the client as is. Here you can include a stylesheet reference in the XML itself and browser will apply that stylesheet to your document. This requires that the client browser supports XML.
2. You have generated XML data but want to send it as HTML to the client. Here you can generate a simple HTML by applying a stylesheet at server side.This do not enforce client browser support for XML
3. You want to send XML data as Data Islands to the client. Here you can use <XML> tag with an ID attribute or <SCRIPT LANGUAGE=XML>. IE also provides a way to directly bind data islands to HTML elements, liks TABLEs.

Now, let's look at code for each of the three scenarios described above. Code for the first method follows below :

```
Dimmydoc,myelement
Set mydoc=Server.CreateObject("Microsoft.XMLDOM")
mydoc.load("xml_file_path")
Response.ContentType = "text/xml"
Response.write mydoc.xml
Or
Response.write "xml_as_string"
```

Here we set ContentType to text/xml so that an XML-aware client browser knows what to do with it. We can also send raw XML in the form of string:

```
Dim myXML,myXSL

myXML= Server.CreateObject("Microsoft.XMLDOM")

myXSL= Server.CreateObject("Microsoft.XMLDOM")

'Fill myXML with one of the methods mentioned above

'Load a stylesheet

myXSL.load "xsl_path"

strHTML = mydoc.transformNode(myXSL.documentElement)

Response.write strHTML
```

Finally, using the third method (using the <XML> tag with an ID attribute or <SCRIPT LANGUAGE=XML>), we have:

```
<XML ID=mydata>

<node1>

<child1>Some data</child1>

<child2>Some other data</child2>

...

</node1>

</XML>
```

Now, you can bind a table to this data island like this:

```
<table datasrc="#mydata">

<tr>

<td>

<div datafld="child1"></div>

</td>

<td>

<div datafld="child2"></div>

</td>
```

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

The table will display all the rows from the data island automatically.

Storing XML data

Suppose you want to save XML data - generated by you or posted by a client - to a disk file. This is not too difficult, simply call the save() method of Document object as follows.

```
Dim mydoc,strXML
Set mydoc=Server.CreateObject("Microsoft.XMLDOM")
strXML="<book><author>author1</author><title>title1</title></book>"
mydoc.loadXML(strXML)
mydoc.save("path_goes_here")
```

Note :

With IIS 5.0 you can also save the XML data to ASP Response object directly with mydoc.save(Response). This is useful if you are processing the XML data and sending it back to client.

5.8 Using XML with HTML

Q20. What is the relation ship between XML and HTML?

Ans : **(Imp.)**

XML can be used to store data inside HTML documents. XML data can be stored inside HTML pages as "Data Islands". As HTML provides a way to format and display the data, XML stores data inside the HTML documents. The data contained in an XML file is of little value unless it can be displayed, and HTML files are used for that purpose.

The simple way to insert XML code into an HTML file is to use the <xml> tag. The XML tag informs, the browser that the contents are to be parsed and interpreted using the XML parser. Like most other HTML tags, the <xml> tag has attributes. The most important attribute is the ID, which provides for the unique naming of the code. The contents of the XML tag come from one of two sources : inline XML code or an imported XML file.

If the code appears in the current location , it's said to be inline.

➢ **Embedding XML code inside an HTML File.**

```
<html>
<xml Id = msg>
```

```
<message>
<to> Visitors </to>
<from> Author </from>
<Subject> XML Code Islands </Subject>
<body> In this example, XML code is embedded inside HTML code </body>
</message>
</xml>
</html>
```

- The efficient way is to create a file and import it. You can easily do so by using the SRC attribute of the XML tag.

```
<xml Id = msg SRC = "example1.xml">
</xml>
```

Data Binding

Data binding involves mapping, synchronizing, and moving data from a data source, usually on a remote server, to an end user's local system where the user can manipulate the data. Using data binding means that after a remote server transmits data, the user can perform some minor data manipulations on their own local system. The remote server does not have to perform all the data manipulations nor repeatedly transmit variations of the same data.

- Data binding involves moving data from a data source to a local system, and then manipulating the data, such as, searching, sorting, and filtering, it on the local system.
- When you bind data in this way, you do not have to request that the remote server manipulate the data and then retransmit the results; you can perform some data manipulation locally.
- In data binding, the data source provides the data, and the appropriate applications retrieve and synchronize the data and present it on the terminal screen.
- If the data changes, the applications are written so they can alter their presentation to reflect those changes.
- Data binding is used to reduce traffic on the network and to reduce the work of the Web server, especially for minor data manipulations.
- Binding data also separates the task of maintaining data from the tasks of developing and maintaining binding and presentation programs.

Q21. What are the differences between HTML and XML?

Ans :

S.No.	XML	S.No.	HTML
1.	The full form is eXtensible Markup Language.	1.	The full form is Hypertext Markup Language.
2.	The main purpose is to focus on the transport of data and saving the data.	2.	Focusses on the appearance of data. Enhances the appearance of text.
3.	XML is dynamic because it is used in the transport of data.	3.	HTML is static because its main function is in the display of data.
4.	It is case sensitive. The upper and lower case needs to be kept in mind while coding.	4.	It is not case sensitive. Upper and lower case are of not much importance in HTML.
5.	You can define tags as per your requirement but closing tags are mandatory.	5.	It has its own pre-defined tags and it is not necessary to have closing tags.
6.	XML can preserve white spaces.	6.	White spaces are not preserves in HTML.
7.	eXtensible Markup Language is content-driven and not many formatting features are available.	7.	Hypertext Markup Language, on the other hand, is presentation driven. How the text appears is of utmost importance.
8.	Any error in the code shall not give the final outcome.	8.	Small errors in the coding can be ignored and the outcome can be achieved.
9.	The size of the document may be large No lengthy documents.	9.	Only the syntax needs to be added for best-formatted output.

FACULTY OF INFORMATICS

M.C.A II Year III - Semester Examination

MODEL PAPER - I

WEB TECHNOLOGIES

Time : 3 Hours] [Max. Marks : 70

Note: Answer all the question according to the internal choice **(5 × 14 = 70)**

			ANSWERS
1.	(a)	What is HTML ?	**(Unit-I, Q.No. 1)**
	(b)	What is hyperlink? How can you link a page with another page.	**(Unit-I, Q.No. 11)**
		OR	
2.	(a)	Write a short notes on Iframes ?	**(Unit-I, Q.No. 22)**
	(b)	Explain different ways to style the text in CSS?	**(Unit-I, Q.No.37)**
3.	(a)	Describe about Java collections framework?	**(Unit-II, Q.No. 2)**
	(b)	Explain about On mouse Over event?	**(Unit-II, Q.No. 9)**
		OR	
4.		Explain Structured Graphics of Active Controls.	**(Unit-II, Q.No.18)**
5.		What is Java Script? What are the Features of Java Script?	**(Unit-III, Q.No.3)**
		OR	
6.		Explain about Loop control Statements?	**(Unit-III, Q.No.9)**
7.		Explain the concept of VBScript ?	**(Unit-IV, Q.No. 1)**
		OR	
8.		What are functions in VB Script ?	**(Unit-IV, Q.No.8)**
9.	(a)	Explain the representation of ASP in VB Script and Java Script?	**(Unit-V, Q.No.2)**
	(b)	What are Regular Expressions in PERL ?	**(Unit-V, Q.No.11)**
		OR	
10.		What are the differences between HTML and XML?	**(Unit-V, Q.No.21)**

FACULTY OF INFORMATICS

M.C.A II Year III - Semester Examination

MODEL PAPER - II

WEB TECHNOLOGIES

Time : 3 Hours] [Max. Marks : 70

Note: Answer all the question according to the internal choice (5 × 14 = 70)

			ANSWERS
1.	(a)	HTML is a Markup language - Explain.	**(Unit-I, Q.No. 2)**
	(b)	What is a frame? Explain different types of frames.	**(Unit-I, Q.No.21)**
		OR	
2.	(a)	What is a Meta Tags?	**(Unit-I, Q.No. 24)**
	(b)	What is mean by Nested tables? How can you create ?	**(Unit-I, Q.No.17)**
3.	(a)	What is a object model ? Explain its Properties and functions ?	**(Unit-II, Q.No.1)**
	(b)	Explain briefly about On mouse Out event?	**(Unit-II, Q.No. 10)**
		OR	
4.		Explain filters and transactions for multimedia effects.	**(Unit-II, Q.No. 16)**
5.		What are arrays in JavaScript? Explain its Methods.	**(Unit-III, Q.No.12)**
		OR	
6.		What are objects? Explian how to create an object.	**(Unit-III, Q.No. 17)**
7.		Discuss briefly about VB Script Loop Control Statements ?	**(Unit-IV, Q.No. 5)**
		OR	
8.		How to install a Web Server?	**(Unit-IV, Q.No.16)**
9.	(a)	Define array. Explain its working mechanism.	**(Unit-V, Q.No.5)**
	(b)	What are the different functions in ASP ?	**(Unit-V, Q.No.13)**
		OR	
10.		Discuss about different types DTD Element.	**(Unit-V, Q.No. 17)**

FACULTY OF INFORMATICS

M.C.A II Year III - Semester Examination

MODEL PAPER - III

WEB TECHNOLOGIES

Time : 3 Hours] [Max. Marks : 70

Note: Answer all the question according to the internal choice (5 × 14 = 70)

ANSWERS

1. (a) Explain the Features of HTML **(Unit-I, Q.No. 5)**

   (b) Explain in detail the CSS box model. **(Unit-I, Q.No.33)**

OR

2. (a) How can you insert an image in to your web document ? **(Unit-I, Q.No.14)**

   (b) Discuss about properties and values in detail with their syntaxes? **(Unit-I, Q.No. 34)**

3. (a) Discuss in brief about Navigator. **(Unit-II, Q.No. 3)**

   (b) Explain about briefly Onfocus event? **(Unit-II, Q.No. 11)**

OR

4. What is Exception Handling? Explain how to handle the exceptions in Java Script? **(Unit-II, Q.No.15)**

5. Explain the functions involved in math Object. **(Unit-III, Q.No.18)**

OR

6. What are the different data types in Java Script? **(Unit-III, Q.No.5)**

7. What isa web Server? Explain its Working ? **(Unit-IV, Q.No. 11)**

OR

8. Discuss diferent types of Operators in VB Script ? **(Unit-IV, Q.No. 3)**

9. (a) Discuss different Components of ASP ? **(Unit-V, Q.No.3)**

   (b) What is a CGI Script ? **(Unit-V, Q.No.8)**

OR

10. Explain how to Access XML Data in ASP. **(Unit-V, Q.No.19)**

FACULTY OF INFORMATICS

M.C.A. (2 years Course) III - Semester (CBCS) (Main) Examination

April/May - 2023

WEB TECHNOLOGY

Time : 3 Hours Max. Marks : 70

Note : I. Answer one question from each unit. All questions carry equal marks.

II. Missing data, if any, may be suitably assumed.

Answers

Unit-I

1. (a) Define Listing. Explain types of lists. **(Unit-I, Q.No. 20)**

   (b) What is HTML? What are the features of HTML? **(Unit-I, Q.No. 1, 5)**

(OR)

2. (a) Explain Table. Table Property with Example. **(Unit-I, Q.No. 16)**

   (b) Explain the different types of CSS with an example. **(Unit-I, Q.No. 32)**

Unit-II

3. (a) Discuss in brief about Event. Types of Events. **(Unit-II, Q.No. 4, 6, 7, 8)**

   (b) What is an object model? Explain its properties and functions. **(Unit-II, Q.No. 1)**

(OR)

4. (a) Explain about Exception Handing Mechanization. **(Unit-II, Q.No. 15)**

   (b) Describe about Java collections Frame work. **(Unit-II, Q.No. 2)**

Unit-III

5. (a) Explain different operators in Java Script. **(Unit-III, Q.No. 6)**

   (b) What are the different data types in Java Script? **(Unit-III, Q.No. 5)**

(OR)

6. (a) What are the Java Scripts Function? Explain its types. **(Unit-III, Q.No. 15)**

   (b) What are objects? Explain how to create an object. **(Unit-III, Q.No. 17)**

Unit-IV

7. (a) Write about VB Scripts Looping Statements. **(Unit-IV, Q.No. 5)**

   (b) Discuss about Apache Web Server. **(Unit-IV, Q.No. 15)**

(OR)

8. (a) What is an array? Explain in detail the VBScript. **(Unit-IV, Q.No. 9)**

   (b) Describe about Personal Web Server. **(Unit-IV, Q.No. 12)**

Unit - V

9. (a) Discuss about Different data types in CGI PERL **(Unit-V, Q.No. 10)**

   (b) Discuss about different types of DTD Element. **(Unit-V, Q.No. 17)**

(OR)

10. (a) Discuss about the ASP Server Object. **(Unit-V, Q.No. 14)**

    (b) What is XML Parser? **(Unit-V, Q.No. 18)**

FACULTY OF INFORMATICS

M.C.A. (2 years Course) III - Semester (CBCS) (Main) Examination

October/November - 2023

WEB TECHNOLOGIES

Time : 3 Hours Max. Marks : 70

Note : I. Answer one question from each unit. All questions carry equal marks.

II. Missing data, if any, may be suitably assumed.

Unit-I

1.	(a)	Explain in detail the CSS Box model.	**(Unit-I, Q.No. 33)**
	(b)	What is DHTML? What are the features of DHTML?	**(Unit-I, Q.No. 25)**
		(OR)	
2.	(a)	Explain briefly the difference between HTML and DHTML.	**(Unit-I, Q.No. 28)**
	(b)	Explain the different types of CSS with an example.	**(Unit-I, Q.No. 32)**

Unit-II

3.	(a)	Discuss in brief about Navigate.	**(Unit-II, Q.No. 3)**
	(b)	What is an object model? Explain its properties and functions.	**(Unit-II, Q.No. 1)**
		(OR)	
4.	(a)	Explain files and translates for multimedia effects.	**(Unit-II, Q.No. 16)**
	(b)	Explain briefly about event handling mechanism.	**(Unit-II, Q.No. 15)**

Unit-III

5.	(a)	What are the control status in Java Servlets?	**(Unit-III, Q.No. 7)**
	(b)	What are the different data types in JavaScript?	**(Unit-III, Q.No. 5)**
		(OR)	
6.	(a)	What are the objects? Explain how to create an objective.	**(Unit-III, Q.No. 17)**
	(b)	What are arrays in JavaScript? Explain its methods.	**(Unit-III, Q.No. 12)**

Unit-IV

7.	(a)	Explain the concept of VBScript.	**(Unit-IV, Q.No. 1)**
	(b)	What are the functions in VBScript?	**(Unit-IV, Q.No. 8)**
		(OR)	
8.	(a)	What is an array? Explain in detail the VBScript.	**(Unit-IV, Q.No. 9)**
	(b)	Write about VBScript strings and related functions.	**(Unit-IV, Q.No. 7)**

Unit-V

9.	(a)	Explain how to access XML data in ASP.	**(Unit-V, Q.No. 19)**
	(b)	What is the relationship between the XML and HTML?	**(Unit-V, Q.No. 21)**
		(OR)	
10.	(a)	Discuss about the different types of DTD elements.	**(Unit-V, Q.No. 17)**
	(b)	State the benefits and goals of XML.	**(Unit-V, Q.No. 16)**