

Q. What is Java Script? Explain different ways of writing Java Script.

Ans: Java Script is a programming language for small routine and applications, and developed to serve a particular purpose. Scripting languages are usually interpreted rather than compiled. Programs written interpreted languages must be translated into machine code every time they are run, they are typically slower than compiled programs.

Being interpreted does have its advantages. One is platform independence. Because an interpreter performs the translation, you can write your program once and run it on a variety of platforms. In the case of JavaScript, the interpreter is built into Web Browsers. Browsers are available for a variety of platforms and operating systems. Another advantage is that scripting languages are often loosely typed and more forgiving than compiled languages.

As a scripting language, JavaScript is easy to learn and easy to use. You can embed command directly into your HTML code and the browser will interpret and run them at the appropriate time. JavaScript is loosely typed. You don't have to declare the type of data that will be stored in available before you use it.

Java Vs Java Script

JavaScript and Java are alike in more than just name. However, there are significant differences between these two languages.

JavaScript	Java
Interpreted by client	Compiled by the author, run on client
Code integrated in HTML documents	Applets distinct from HTML document
Loose typing of data types	Strong typing of data types
Dynamic binding	Static binding
Script limited to browser functions	Stand-alone applications
Works with HTML elements	Goes beyond HTML (for example, multimedia)
Access browser objects and functionality	No access to browser functionality objects or

Writing JavaScript

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

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

The SCRIPT Tag

The <SCRIPT> tag alerts a browser that JavaScript code follows. It is typically embedded in the HTML. Script code can be written in script element of HTML

```
<script>...</script>
```

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

```
statements
```

```
</SCRIPT>
```

SCRIPT Example

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

Attributes of script element are:**TYPE:** providing content type

Ex: type="text/javascript"

LANGUAGE: Specifying the which scripting language that script is using default is JavaScript

Ex: language="javascript"

SRC : A URL of externally linked script

Ex: SRC="filename.js"

Embedding JavaScript into a HTML-page

JavaScript code is embedded directly into the HTML-page. In order to see how this works we are going to look at an easy example:

```
<html>
```

```
<body>
```

```
<br>
```

This is a normal HTML document.

```
<br>
```

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

```
document.write("This is JavaScript!")
```

```
</script>
```

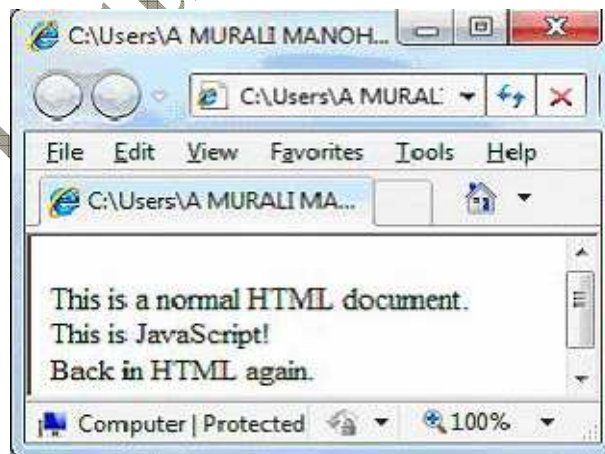
```
<br>
```

Back in HTML again.

```
</body>
```

```
</html>
```

Here is the output generated by the file (if you are using a JavaScript browser you will see 3 lines of output):



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

- Inline Scripting (Embedding code)
- Internal Scripting
- External Scripting (External file)

In Line Scripting

```
<html>
<head><title>Script Demo -1</title>
</head>
<body>
<script language="javascript">
document.write("<H2>Welcome to Java Script</H2> ");
</script></body></html>
```

Internal Scripting

```
<html>
<head><title>Script Demo - 2</title>
<script language="javascript">
document.write("<H2>Welcome to Java Script</H2> ");
</script>
</head><body></body>
</html>
```

External File

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

```
<SCRIPT SRC="filename.js">
</SCRIPT>
<SCRIPT language = "JavaScript" SRC = "external.js">
</SCRIPT>
```

Variable Declaration

Programmers use variables to store values. A variable can hold several types of data. In JavaScript you don't have to declare a variable's data type before using it. Any variable can hold any JavaScript data type, including:

- String data
- Numbers
- Boolean values (T/F)

o To declare variables, use the keyword var and the variable name: var userName

o To assign values to variables, add an equal sign and the value: var userName = "Smith", var price = 100

Q. Explain about various dialogue boxes available in Java Script.

Ans: **JavaScript Input/Output:**

ALERT:

Alert boxes are used to display some text information and have the visitor click on the OK button to continue on. This script is usually placed in the HEAD area of the code which makes the box appear before the actual page load.

```
<script type="text/javascript">
alert ("Welcome to my friend");
</script>
```

ALERT creates the actual ALERT box to appear. Anything placed within the double quotes will become the message that appears on the ALERT

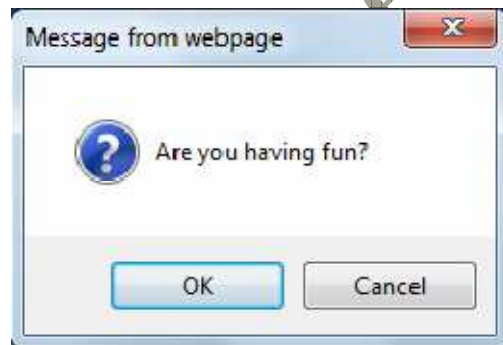
box. In this example a dialogue box will be displayed which will contain the message-Welcome my friend.



CONFIRM

The Javascript confirm function is very similar to the Javascript alert function. The following example shows how to make use of 'confirm' dialog box. This dialog box can be used for confirming something. The confirm box is different than the alert box in that it supplies the user with a choice, they can either press OK to confirm the popups message or they can press cancel and not agree to the popups request. This function returns a boolean value (true or false). If you press 'Ok', True is returned, if 'cancel' is pressed False is returned. After clicking OK or 'cancel', see the status bar for results.

```
<script language="javascript">
var ans = confirm ("Are you having fun?")
if (answer) alert ("Welcome to my friend");
</script>
```



PROMPT

A PROMPT is used to actually gather a bit of information when a simple TRUE/FALSE is not enough. This will take some information from the visitor and you can use it in other applications. PROMPT creates the actual pop-up box with a text area for the visitor to enter information based on the text following the PROMPT command. The empty set of quotes in the second part of the PROMPT is to clear the PROMPT text box. You can enter any text within the quotes to have a default answer appear.

```
<script type="text/javascript">
var answer = prompt ("please enter your name ?", "")
alert ("Hello there, " + answer)
</script>
```

In the PROMPT example if we entered nothing then it will read null If we enter any name in place of null it will display user entered name value



Q. Write short notes on Java Script functions.

Ans:

Functions: With functions, you can give a name to a whole block of code, allowing you to reference it from anywhere in your program. JavaScript has built-in functions for several predefined operations. Here are three some functions.

- alert("message")
- confirm("message")
- prompt("message")

User-Defined Functions

With user-defined functions, you can name a block of code and call it when you need it. You define a function in the HEAD section of a web page. It is defined with the function keyword, followed by the function name and any arguments.

```
function functionName(argument)
{
  statements
}
<SCRIPT language = "JavaScript">
function showAlert() {
alert("this is a user-defined function.")
}
</SCRIPT>
```

Q. Explain the properties and methods Java Script built-in Objects.

Ans:

Built-In Objects

Some of the built-in language objects of JavaScript offer more advanced operations such as:

- Math – provides for math calculations
- Date – provides date and time information
- String – provides for string manipulation

Objects

JavaScript supports programming with objects. Objects are a way of organizing the variables. The different screen elements such as Web pages, forms, text boxes, images, and buttons are treated as objects.

Properties and Methods

Every object has its own properties and methods.

- Properties define the characteristics of an object.

Examples: color, length, name, height, width

- Methods are the actions that the object can perform or that can be performed on the object.

Examples: alert, confirm, write, open, close

JavaScript uses dot notation to refer to an object and its associated properties and methods for instance, in dot notation we refer to the document object and its background property color we write

```
document.bgColor
```

We can also write directly to document using the document's write method

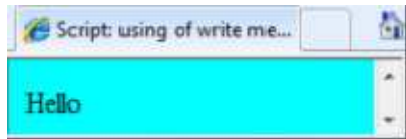
```
document.write("Hello")
```

```
<HTML>
```

```
<HEAD> <TITLE> Script: using of write method </TITLE>
```

```
</HEAD>
```

```
<BODY bgcolor="red" text="black">
<SCRIPT language = "JavaScript" type="text/JavaScript">
document.bgColor="cyan";
document.write("Hello");
</SCRIPT>
</BODY>
</HTML>
```



You can even use JavaScript to write HTML. Modify the above script write the greeting as ventured heading.

```
<SCRIPT language = "JavaScript" type="text/JavaScript">
document.bgcolor="cyan";
document.write("<CENTER><H1>Hello</H1></CENTER>");
</SCRIPT>
```



Hear what it looks like:

Document Object

The Document object represents the Web page that is loaded in the browser window, and the content displayed on that page, including text and form elements.

Document Methods

You can use the methods of the document object to work on a Web page. Here are the most common document methods:

- write() - write a string to the Web page
- open() - opens a new document
- close() - closes the document

Document Properties

Use the properties of the document object to set the colors of the page, the title and display the date the document was last modified. JavaScript has about 150 defined color words you can use or you can provide the hexadecimal RGB codes. Here are the most common document properties:

- bgColor – to set the background color of the document.
- fgColor – to set the foreground color of the document .
- linkColor – to set the color for the links of the document.
- vlinkColor
- title – to set the title for document.
- lastModified – gives the last modification time.

Window Object

The window object represents the browser window. You can use it to open a Web page in a new window and to set the attributes for the window. There are only two main window properties. They are:

- status - set the status bar message
- self - stores the name of the current window

Window Methods

The window methods are mainly for opening and closing new windows. The following are the main window methods. They are:

`open()` - Opens a new browser window

`close()` - Closes the current window

`focus()` - Sets focus to the current window

`blur()` - Removes focus from the current window

`moveBy()` - Moves a window relative to its current position

`moveTo()` - Moves a window to the specified position

`resizeBy()` - Resizes the window by the specified pixels

`resizeTo()` - Resizes the window to the specified width and height

`scrollBy()` - Scrolls the content by the specified number of pixels

`scrollTo()` - Scrolls the content to the specified coordinates

`print()` - Prints the content of the current window

`alert()` - Displays an alert box with a message and an OK button

`prompt()` - Displays a dialog box that prompts the visitor for input

`confirm()` - Displays a dialog box with a message and an OK and a Cancel button

Window Attributes:

If the default new window does not suit your needs, you can specify different features of the window when you open it. The complete syntax of the "window.open" is as follow:

`window.open(URL, windowName, featureList);`

By default, if you do not specify any features, then a window will have all of them. If you specify any one feature, then the window will have only those you set equal to 1.

The following are the window attributes:

- toolbar
- menubar
- scrollbars
- resizable
- status
- location
- directories

Window Properties:

name Sets or returns the name of a window

defaultStatus Sets or returns the default text in the statusbar of a window

status Sets the text in the statusbar of a window

length Returns the number of frames (including iframes) in a window

pageXOffset Returns the pixels the current document has been scrolled (horizontally) from the upper left corner of the window

pageYOffset Returns the pixels the current document has been scrolled (vertically) from the upper left corner of the window

screenX , **screenLeft** Returns the x coordinate of the window relative to the screen

screenY , **screenTop** Returns the y coordinate of the window relative to the screen

self Returns the current window

opener Returns a reference to the window that created the window

top Returns the topmost browser window

parent Returns the parent window of the current window

closed Returns a Boolean value indicating whether a window has been closed or not

frames Returns an array of all the frames (including iframes) in the current window

Window Objects:

document Returns the Document object for the window (See Document object)

history Returns the History object for the window (See History object)

location Returns the Location object for the window (See Location object)

navigator Returns the Navigator object for the window (See Navigator object)

screen Returns the Screen object for the window (See Screen object)

History Object:

length Returns the number of URLs in the history list

back() Loads the previous URL in the history list

forward() Loads the next URL in the history list

go() Loads a specific URL from the history list (..-1,1..)

Navigator Object Properties and method

appName Returns the name of the browser

appVersion Returns the version information of the browser

cookieEnabled Determines whether cookies are enabled in the browser

platform Returns for which platform the browser is compiled

userAgent Returns the user-agent header sent by the browser to the server

javaEnabled() Specifies whether or not the browser has Java enabled

The location-object:

Besides the window- and document-objects there is another important object: the location-object. This object represents the address of the loaded HTML-document. So if you loaded the page *http://www.xyz.com/page.html* then *location.href* is equal to this address. What is more important is that you can assign new values to location.

Hash - Returns the anchor portion of a URL

Host - Returns the hostname and port of a URL

Hostname - Returns the hostname of a URL

Href - Returns the entire URL

Pathname - Returns the path name of a URL

Port - Returns the port number the server uses for a URL (for port 80 nothing returns)

Protocol - Returns the protocol of a URL

Search - Returns the query portion of a URL (returns after ?)

assign() - Loads a new document

reload() - Reloads the current document

replace() - Replaces the current document with a new one

The Date-object

JavaScript lets you use some predefined objects. We are going to have a look at the Date-object first. As the name implies this object lets you work with time and date. For example you can easily calculate how many days are left until next Festival. Or you can add the actual time to your HTML-document. The Date-object is created using the new operator.

```
var today;
```

```
today = new Date();
```

This creates a new Date-object called today. If you do not specify a certain date and time when creating a new Date-object the actual date and time is used. The Date-object offers some methods which can now be used with our object today.

This is for example `getHours()`, `setHours()`, `getMinutes()`, `setMinutes()`, `getMonth()`, `setMonth()` and so on.

In order to get another date and time we can use another constructor (this is the Date() method which is called through the new operator when constructing a new Date-object):

```
today= new Date(1997, 0, 1, 17, 35, 23)
```

Specify the date and time like this:

```
Date(year, month, day, hours, minutes, seconds)
```

Please note that you have to use 0 for January - and not 1 as you might think. 1 stands for February and so on.

The code looks like this:

```
<script language="JavaScript">
now= new Date();
document.write("Time: " + now.getHours() + ":" + now.getMinutes() + "<br>");
document.write("Date: " + (now.getMonth() + 1) + "/" + now.getDate() + "/" +
(1900 + now.getYear()));
</script>
```

The Array-object

Arrays are very important. Just think of an example where you want to store 100 different names. How could you do this with JavaScript? Well, you could define 100 variables and assign the different names to them. This is quite complicated. Arrays can be seen as many variables bundled together. You can access them through one name and a number.

Let's say out array is called names. Then we can access the first name through myArray[0]. The second name is called myArray[1] and so on.

You can create a new array through myArray= new Array(). Now you can assign values to this array:

```
myArray[0]= 17;
myArray[1]= "Stefan";
myArray[2]= "Koch";
```

JavaScript arrays are really flexible. You do not have to bother about the size of the array its size is being set dynamically.

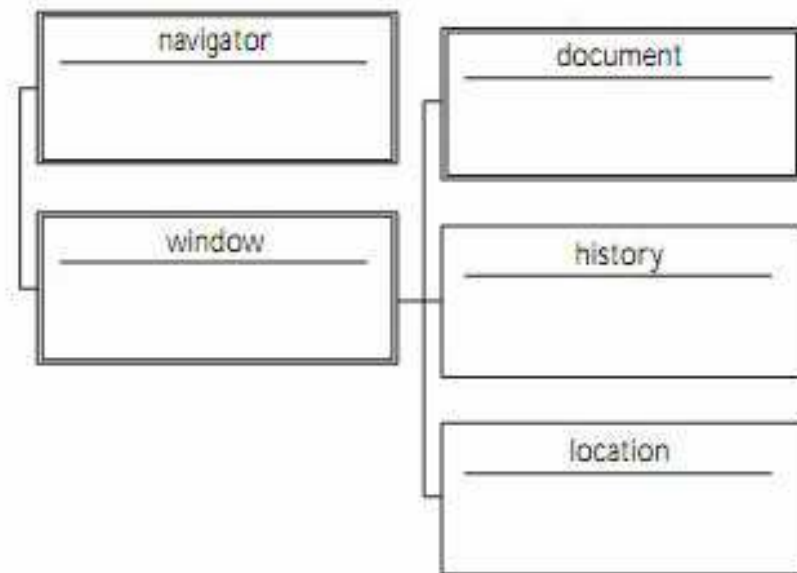
The Math-object

If you need to do mathematical calculations you will find some methods in the Math object which might help you further. There is for example a sine-method sin(). You will find a complete reference in the Netscape documentation. I want to demonstrate the random() method. If you have read the first version of this tutorial you know that there have been some problems with the random() method. We wrote a function which let us create random numbers. We don't need that anymore as the random() method now works on all platforms. If you call Math.random() you will receive a random number between 0 and 1. Here is one possible output of document.write(Math.random()).

Q. Explain about DOM in Java Script.

Ans:

DOM: Web documents are made available to Client-Side JavaScript via the Document Object Model(DOM). The DOM illustration represents the DOM common to all major browsers. The items contained in a Web document become objects in the browser's memory as the browser loads and interprets the HTML code that defines them. Notice that the DOM is arranged in a hierarchical way with window as the highest-level object.



DOM (Document Object.)

anchors[] -Returns an array of all the anchors in the document

forms[] -Returns an array of all the forms in the document

images[] -Returns an array of all the images in the document

links[] -Returns an array of all the links in the document

title -Sets or returns the title of the document

URL -Returns the full URL of the document

close() -Closes the output stream previously opened with document.open()

getElementById() -Accesses the first element with the specified id

getElementsByName() -Accesses all elements with a specified name

getElementsByTagName() -Accesses all elements with a specified tagname

open() -Opens an output stream to collect the output from document.write() or document.writeln()

write() -Writes HTML expressions or JavaScript code to a document

writeln() -Same as write(), but adds a newline character after each statement

Q: Explain about Java Script Events.

Ans:

Events

The objects in a Web pages are organized in a hierarchy. Events and event handlers are very important for JavaScript programming. Events are mostly caused by user actions. If the user clicks on a button a Click-event occurs. If the mousepointer moves across a link a MouseOver-event occurs. There are several different events. All objects have properties and methods. In addition, some objects also have "events". Events are things that happen, usually user actions, that are associated with an object. The "event handler" is a command that is used to specify actions in response to an event. Below are some of the most common events:

- **onLoad** - occurs when a page loads in a browser
- **onUnload** - occurs just before the user exits a page
- **onMouseOver** - occurs when you point to an object
- **onMouseOut** - occurs when you point away from an object
- **onSubmit** - occurs when you submit a form
- **onClick** - occurs when an object is clicked

Events and Objects

Events are things that happen, actions, that are associated with an object. Below are some common events and the object they are associated with:

Event Object

onLoad Body

onUnload Body

onMouseOver Link, Button

onMouseOut Link, Button

onSubmit Form

onClick Button, Checkbox, Submit,

Reset, Link

We want our JavaScript program to react to certain events. This can be done with the help of event-handlers. A button might create a popup window when clicked. This means the window should pop up as a reaction to a Click-event. The event-handler we need to use is called onClick. This tells the computer what to do if this event occurs. The following code shows an easy example of the event-handler onClick:

```
<form>
<input type="button" value="Click me" onClick="alert('Yo')">
</form>
```

You can see that we create a form with a button. The new part is onClick="alert('Yo')" inside the <input> tag. If a Click-event occurs the computer shall execute alert('Yo').

Q: Explain how java script provides provision for form validation? Explain with an example.

Ans:

Validating form input: Forms are widely used on the Internet. The form input is often being sent back to the server or via mail to a certain e-mail account. But how can you be certain that a valid input was done by the user? With the help of JavaScript the form input can easily be checked before sending it over the Internet.

First of all we want to create a simple script. The HTML-page shall contain two text elements.

The user has to write his name into the first and an e-mail address into the second element. If the user has entered his name (for example 'Kanth') into the first text-field the script creates a popup window with the text 'Hi Kanth!'.

Concerning the first input element you will receive an error message when nothing is entered. Any input is seen as valid input. Of course, this does not prevent the user from entering any wrong name. The browser even accepts numbers. So if you enter '17' you will get 'Hi 17!'. So this might not be a good check. The second form is a little bit more sophisticated. Try to enter a simple string - your name for example. It won't work (unless you have a @ in your name...). The criteria for accepting the input as a valid e-mail address is the @. A single @ will do it – but this is certainly not very meaningful. Every Internet e-mail address contains a @ so it seems appropriate to check for a @ here.

```
<html>
<head>
<script language="JavaScript">
<!-- Hide
function test1(form) {
if (form.text1.value == "")
alert("Please enter a string!")
else {
```

```
alert("Hi "+form.text1.value+"! Form input ok!");
}
}
function test2(form) {
if (form.text2.value == "" ||form.text2.value.indexOf('@', 0) == -1)
    alert("No valid e-mail address!");
else
    alert("OK!");
}
</script>
</head>
<body>
<form name="first">
Enter your name:<br>
<input type="text" name="text1">
<input type="button" name="b1" value="Test Input" onClick="test1(this.form)">
<P>
Enter your e-mail address:<br>
<input type="text" name="text2">
<input type="button" name="b2" value="Test Input" onClck="test2(this.form)">
</body>
</html>
```

Two buttons, call the functions test1(...) or test2(...) depending on which button is pressed. We pass this.form to the functions in order to be able to address the right elements in the functions later on.

The function test1(form) tests if the string is empty. This is done via if(form.text1.value == "");.

'form' is the variable which receives the 'this.form' value in the function call. We can get the value of the input element through using 'value' in combination with form.text1. In order to look if the string is empty we compare it with "". If the input string equals "" then no input was done. The user will get an error message. If something is entered the user will get an ok.

Sometimes you want to restrict the form input to certain characters or numbers. Just think of a telephone number the input should only contain digits (we assume that the telephone number does not contain any characters). We could check if the input is a number. But most people use different symbols in their telephone number – for example: 01234-56789, 01234/56789 or 01234 56789 (with a space in between). The user should not be forced to enter the telephone number without these symbols. So we have to extend our script to check for digits and some symbols. This is demonstrated in the next example which is taken.

```
<html>
<head>
<script language="JavaScript">function check(input) {
var ok = true;
for (var i = 0; i < input.length; i++) {
var chr = input.charAt(i);
var found = false;
for (var j = 1; j < check.length; j++) {
if (chr == check[j]) found = true;
```

```
}  
if (!found) ok = false;  
}  
return ok;  
}  
function test(input) {  
if (!check(input, "1", "2", "3", "4", "5", "6", "7", "8", "9", "0", "/", "-", " ")) {  
alert("Input not ok.");  
}  
else alert("Input ok!");  
}  
</script>  
</head>  
<body>  
<form>  
Telephone: <input type="text" name="telephone" value="">  
<input type="button" value="Check"  
onClick="test(this.form.telephone.value)">  
</form>  
</body>  
</html>
```

The function test() specifies which characters are valid.