Introduction to JavaScript

Module 1: Getting Started with JavaScript

1. What is JavaScript?

JavaScript is a cross-platform, object-based scripting language. It is not used as a general purpose standalone programming language, but is designed as client side Web programming language. There is also a server side version of JavaScript.

The Web designer cannot use HTML to perform dynamic tasks, such as changing the colors on a page after the page is loaded (you can do some of it using CSS), or doing certain computations. We need a programming language to do such tasks. JavaScript is ideally suited for doing such client side tasks.

JavaScript, originally called Live Script, is a simple, yet powerful programming language that was developed by Netscape in 1995 to create dynamic Web pages. Soon after that Microsoft developed its version called JScript. In 1997-99 the European Computer Manufacturers Association (ECMA) and the International Organization for Standardization and International Electro technical Commission (ISO/IEC) standardized JavaScript and JScript and called the standard version ECMAScript. Except for occasional differences, all the three JavaScript, JScript and ECMAScript are all the same. We will use the name JavaScript throughout our notes.

JavaScript code is embedded in an HTML page. When the HTML page is downloaded into a client machine, the browser on the client machine executes the JavaScript code embedded in the HTML page. Depending on what the JavaScript code is designed to do, the HTML page shows dynamic behavior.

Reference: https://developer.mozilla.org/en-US/docs/JavaScript/Guide

---

2. Some Salient features of JavaScript:

Here are some JavaScript highlights:
- JavaScript source code is directly embedded in an HTML document.
- Programs are event driven.
- It is compact and easy to learn.
- It is an object-based scripting language.
- It can be used to develop client and server internet applications.
- It is an interpreted language (interpreted by the browser)

---

3. JavaScript and Java

Java is another very important language today. JavaScript and Java are similar in some ways but fundamentally different in some others. The JavaScript language resembles Java but does not have Java's static typing and strong type checking. JavaScript follows most Java expression syntax, naming conventions and basic control-flow constructs which was the reason why it was renamed from LiveScript to JavaScript.

JavaScript is a very free-form language compared to Java. You do not have to declare all variables, classes, and methods. You do not have to be concerned with whether methods are public, private, or protected, and you do not have to implement interfaces. Variables, parameters, and function return types are not explicitly typed.

In summary, JavaScript language
• Supports Java's expression syntax and basic control flow constructs
• Does not have Java's static typing
• Type checking is not as strong as in Java
• Object-based (Java is object-oriented). Uses built-in, extensible objects
• No classes; no inheritance
• Variables are not declared - data type of variables is decided by context
• Cannot automatically write to hard disk

---

4. Writing and executing JavaScript programs

As mentioned earlier, JavaScript source code is embedded in HTML documents. When an HTML document is loaded using a browser, the JavaScript source code is also loaded along with the HTML code. The client browser interprets the JavaScript code.

Clearly, JavaScript, therefore, is platform independent. The platform dependent browser executes the code.

By using the "View Source" option from the browser, you may be able to see the source code for JavaScript.

---

5. The SCRIPT tag

JavaScript code is included between the HTML tags <script>.....</script>:

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

Note:

• You can specify the JavaScript version, such javascript1.8 as value of the attribute language:
<script language="JavaScript1.8">

• The script tag supports another attribute **type,** with value **text/javascript:**
<script type="text/javascript">

Note:

Statements in JavaScript are generally followed by a semicolon character, just as they are in C, C++, and Java. JavaScript, however, allows you to omit this semicolon if your

statements are each placed on a separate line. By habit (from Java), I always place a semicolon at the end of a JavaScript statement.

Note:
JavaScript is a case-sensitive language. This means that language keywords, variables, function names, and any other identifiers are all case-sensitive.

---

6. JavaScript objects

An object is a construct which groups together a set of related data and the functions that are used to process that data. Grouping data and functions in this way is the basic characteristic of object-oriented and object-based languages.

Objects are fundamental building blocks in JavaScript. Every important entity in JavaScript is an object. This includes the document, the browser's history, the window in which the document is displayed, every link and anchor, and every form element, such as a button or a text box.

A variable associated with an object is called a property.
A function associated with an object is called a method.

The properties of objects have values.

objectName.*propertyName* refers to the value of the property *propertyname* for the specified object *objectName*.

objectName.*methodName*(...) returns the value generated by the function *methodName* for the specified object *objectName*.

When a browser is up and running several built-in objects are already available to you.

---

7. The object hierarchy

As you know a JavaScript object is an entity which contains two elements: properties and methods.

Properties are names which hold certain values and methods are functions which do certain computations.

Now an extremely important fact:
Properties of objects can themselves be objects.
This important concept leads to the following chain:
An object can have a property which is an object; inside this object a property can be an object; inside which a property can be an object and so on. This is like having a box with a box inside, which contains a box inside, and so on.

Recall the syntax for accessing a property of an object. If A is an object and B is property of A, the syntax for accessing B is A.B.

Now consider the situation in which B is also an object with a property C, and C is also an object with property D with value 14.

The syntax for accessing the value of D is A.B.C.D.

---

8. The window and document object

As described earlier, when you write JavaScript programs, you will be using several built-in JavaScript objects.

One of the most important JavaScript objects is **window**. The **window** object has several properties and methods. A very important **window** property is **document**.

The **document** property itself is an object.

We use the usual dot notation to access properties and methods.

---

9. The document property write()

The **document** object has a very important method: **write**(). This method is used to write output on the screen.

Now, **write**() is a method of the object **document**, and **document** is a property of the object **window.** To access the **write**() method the dotted notation is **window.document.write**().

Complete syntax for using write():
window.document.write(*items to be printed separated by commas*)

Where items are typically strings and arithmetic expressions. This is an "output statement" in JavaScript. If you want the browser to display the output with special characteristics (such as bold, italics, different colors), you have to include appropriate HTML tags in the **write**() statement.

Important note:

Indicating the window object is optional. This means, **window.document.write()** can be written omitting window: **document.write().**
We will use this format all the time.

Practice Example 9:

```
<html>
<head><title>Document.write Example</title></head>
<body>
```

```
<script>
document.write("This is an example to illustrate document.write() method <P>");
document.write("Done <P>");
</script>
</body>
</html>
```

Try It: http://vulcan.seidenberg.pace.edu/~nmurthy/JavaScriptExamples/Example9.html

---

10. Two more properties of document object

We introduce two more properties of **document** object:
fgColor and **bgColor**.

fgColor refers to foreground color (meaning text color) and **bgColor** refers to background color.

Syntax for fgColor:
document.fgColor = "*color name*"
Sets the text to the specified color.

Example:
document.fgColor = "blue"
Sets the text to blue.

Syntax for bgColor:
document.bgColor = "*color name*"
Sets the background to the specified color.

Example:
document.bgColor = "red"
Sets the background to red.

Note: The letter C in **fgColor** and **bgColor** are capital C.

---

11. Color codes

Colors are indicated by their names or by using three numbers.

Specifying colors by name:
Here is a partial list of color names you can use to specify colors: aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white, and yellow.
This site gives 140 color names: http://www.html-color-names.com/color-chart.php.

Specifying colors by numbers:

Color names can be specified using three numbers, each between 0 and 255. The three numbers correspond to red, green and blue (R, G, and B). The numbers are typically specified in hexadecimal format. The syntax for specifying the three numbers for R, G, and B is: "#RRGGBB", where each R, G, and B is a hexadecimal digit. RR gives the red value, GG gives the green value and BB gives the blue value.

Example:
document.fgColor="#FF0000"" Sets the background color to "#FF0000"", which is red.

This site gives you a RGB color codes: http://www.globalrph.com/davescripts/colorcode.htm

---

Practice Example 11:

```
<html>
<head><title>Color Testing</title></head>
<body>
<script>
document.bgColor="green";
document.fgColor="#FF0000";
document.write("The back ground is green and text is in red <P>");
document.write("Done <P>");
</script>
</body>
</html>
```

Try It: http://vulcan.seidenberg.pace.edu/~nmurthy/JavaScriptExamples/Example11.html

---

12. Strings in JavaScript

A bunch of characters enclosed in double quotes is called string data. Even a number enclosed in double quotes is a string. For example, "421" is a string and not a number. A "numeric looking" string must be first converted into a number before it can be used in computations. We will see later how to do such conversions.

Note:
Strings are objects in JavaScript. One important property of a string object is **length**, which gives the length, the number of characters in the string.

Example:
city = "New York";
city.length gives 8 (space is a character too!).

---

13. Dual role of + in JavaScript

In JavaScript, the + operator has a dual role.
One use of +: To add numbers.
Second use of +: To concatenate strings.

If both operands are numbers, + performs addition. If at least one of them is a string, it performs concatenation. Furthermore, + is left associative.

Examples:

Expression	Result	Comment
2 + 3 + 4	9	Computation is done left to right
2 + 3 * 4	14	Multiplication has higher precedence
2+3 * 4 +5	19	Computation is done left to right – multiplication is done first.
3 + "abc"	3abc	Left to right - concatenation
"abc" + 3	Abc3	Left to right - concatenation
"abc" + 3 + 4	Abc34	Left to right - concatenation
3 + 4 + "abc"	7abc	Left to right, addition and then concatenation
3 + 4 +"abc" + 3 + 4	7abc34	Left to right, addition and then concatenation

---

14. Converting strings into numbers

JavaScript provides two functions that convert strings into numbers. One of these functions is used when you want an integer result, and the other is used to produce a decimal number.

The parseInt() method:

Syntax:
parseInt(*string*)

This method returns the *string* as an integer.

If the argument, string, does not start with a number, **parseInt**() gives an error message. If the argument starts with digits followed by some other characters, **parseInt**() converts the number part of the string.

parseInt() always returns a whole number.

Examples:

parseInt("235") returns the number 235.
parseInt("23.45'') returns the number 23.
parseInt("23.45abc") returns 23.
parseInt("ab354") results in an error.

The parseFloat() method:

Syntax:
parseFloat(*string*)

This method returns *string* as a decimal number (i.e., with a decimal point).
If the argument does not start with a number, **parseFloat**() gives an error message.
If the argument starts with a number followed by some other characters, **parseFloat**() converts the number part of the string.

Examples:
parseFloat("235") returns the number 235.
parseFloat("23.45'') returns the number 23.45.
parseFloat("23.45abc'') returns 23.45.
parseFloat("ab34.46'') results in an error.

---

15. The alert box

An **alert box** is a small pop up window JavaScript creates with an OK button.

Syntax:
alert("*message*")

The alert box will display the message.
When the user clicks the OK button, the alert dialog box is removed.

Practice Example 15:

```
<html>
<head><title> Alert Box Example</title></head>
<body>
<h3> ALERT DIALOG BOX </h3>
<script>
alert("Good Job!!")
</script>
<h2> We Are Done </h2>
</body>
</html>
```

Try It: http://vulcan.seidenberg.pace.edu/~nmurthy/JavaScriptExamples/Example15.html

Note:
The HTML document which comes after **alert**(), will be displayed only after the user clicks the OK button in the alert dialog box.

---

16. The confirm dialog box

The **confirm** dialog box is similar to the **alert** dialog box, except that the confirm dialog box displays two buttons: **OK** and **Cancel**. The user can choose to click either one. When the user clicks either button, the dialog box disappears.

Syntax:
confirm("*message*")
The confirm dialog box is used to get true or false (yes/no) input from the user.

The **confirm**() dialog box returns true when the user clicks the OK button, otherwise, it returns false. To retrieve the value returned by the **confirm**() box, use an assignment statement as shown in the example.

Practice Example 16:

```
<html>
<head><title> JavaScript Example: confirm dialog box</title></head>
<body>
<h3> CONFIRM DIALOG BOX </h3>
<script>
answer = confirm("YES or NO?")
document.write("Your answer:",answer);
</script>
<h2> We Are Done </h2>
</body>
</html>
```

Tri It: http://vulcan.seidenberg.pace.edu/~nmurthy/JavaScriptExamples/Example16.html

---

17. The prompt dialog box

A prompt box can be used to receive an input value from the user.

Syntax:
prompt("*a string*","*default value*");

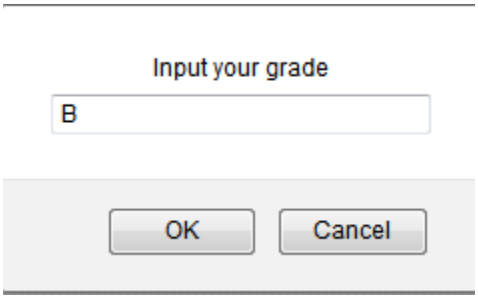
The first argument to **prompt**() is a string, which prompts the user to input a value. The second argument is a default value. If the user does not input any value, the default value is used.

The value can be received by assigning to a variable.

Example:

```
grade = prompt("Please input your letter grade","B");
```

When this line is executed, you will see,



Notice the default value in the box. If the user types a new value and clicks OK, the value of the variable grade will be the value typed. If the user clicks Cancel, the value of the variable grade will be null.

Note:
The second argument (default value), can be empty string, "".

Example:

```
<html>
<head><title> JavaScript</title></head>
<body>
<h3> CONFIRM DIALOG BOX Example</h3>
<script>
num1 = prompt("Input an integer number","");
num2 = prompt("Input another integer number","");
document.write("Sum of the number:", parseInt(num1)+parseInt(num2));
</script>
<H2> We Are Done </H2>
</body>
</html>
```

When you run this, it will show prompt dialog boxes twice. It assigns the values you type and converts the strings into numbers and displays their sum. Try it.

18. Comments in JavaScript

Comments are remarks placed in a program to make the program easily readable. These comments are ignored by the interpreter when the page is loaded and executed. JavaScript provides two kinds notations to place comments in programs:
// is used for one line comments. Everything after the two slashes is ignored.
/* is used for multiline comments. Everything between the slash-asterisk and the asterisk-slash is ignored. */
There is yet another comment: <!-- ... -->
This is really HTML comments tag. Everything between <!-- ... -->
Use these symbols to make non-JavaScript-enabled browser ignore JavaScript code.
Typically, type <!-- immediately after the <script> tag and --> just before the </script> tag. A JavaScript enabled browser will interpret the code between these symbols.

19. More Practice Examples

Practice Example 19:
In this example, a JavaScript program receives first name, last name and telephone number from a user and prints his/her name telephone number. It uses three prompt dialog boxes (one for each of first name, last name and telephone number) to receive data from the user. In the output, the first name and last name appear on the same line and the telephone number on a separate line.

Example19.html

```
<html>
<head><title>Practice Example 19</title></head>
<body>
<h3> This program inputs your first name, last name and telephone number</h3>
<h3> It prints the input data</h3>
<script>
firstName = prompt("Please type your first name","");
lastName = prompt("Please type your last name","");
telephoneNumber = prompt("Please type your telephone number","");
document.write("<b>Your Name:</b> ",firstName," ",lastName);
document.write("<P><b>Your Telephone Number: </b>",telephoneNumber);
</script>
</body></html>
```

Try It: http://vulcan.seidenberg.pace.edu/~nmurthy/JavaScriptExamples/Example19.html

---

Practice Example 19A:
In this example, we input two integers from the user and print their sum. Notice that the number input is a string. We need to convert it into a number before computation can be done.

Example19A.html

```
<html>
<head><title>Practice Example 19A</title></head>
<body>
<h3> This program inputs two integer numbers and prints their sum</h3>
<script>
firstNumber = prompt("Please type the first integer number","");
secondNumber = prompt("Please type the second integer number","");
document.write("<b>The sum of the numbers:</b>
",parseInt(firstNumber)+parseInt(secondNumber));
</script>
</body></html>
```

Try It: http://vulcan.seidenberg.pace.edu/~nmurthy/JavaScriptExamples/Example19A.html

---

Practice Example 19B:
In this example, we input two decimal numbers (numbers with decimal point) from the user and print their sum. Notice that the number input is a string. We need to convert it into a number before computation can be done.

Example19B.html

```
<html>
<head><title>Practice Example 19B</title></head>
<body>
<h3> This program inputs two integer numbers and prints their sum</h3>
<script>
firstNumber = prompt("Please type the first decimal number","");
secondNumber = prompt("Please type the second decimal number","");
document.write("<b>The sum of the numbers:</b>
",parseFloat(firstNumber)+parseFloat(secondNumber));
</script>
</body>
</html>
```

Try It: http://vulcan.seidenberg.pace.edu/~nmurthy/JavaScriptExamples/Example19B.html

---

Practice Example 19C:
This program inputs a color name (using a prompt dialog box) and makes it the background color for the page.

Example19C.html

```
<html>
<head><title>Practice Example 19C</title></head><body>
<h3> This program inputs a color name and makes it the background color</h3>
<script>
yourColor = prompt("Please type the background you prefer","");
document.bgColor=yourColor;
</script>
</body></html>
```

Try It: http://vulcan.seidenberg.pace.edu/~nmurthy/JavaScriptExamples/Example19C.html

---

Practice Example19D:
This program inputs a color name (using a prompt dialog box) and prints all text in that color.

Exercise19D.HTML

```
<html><head><title>Practice Example 19D</title></head><body>
<h3> This program inputs a color name and prints New York in that color</h3>
<script>
yourColor = prompt("Please type the text color you prefer","");
```

```
document.fgColor=yourColor;
document.write("New York");
</script>
</body></html>
```

Try It: http://vulcan.seidenberg.pace.edu/~nmurthy/JavaScriptExamples/Example19D.html

---

Module 1 Exercises

1. Getting used to objects in JavaScript is very important. If you come across a JavaScript object, you should immediately ask the question what are the useful properties and methods available with the object and how to use the methods.

   As an example, Math is a JavaScript object. The Math object has several properties and methods. One of the properties is PI, which gives you the value of π. One of Math methods is sqrt(), which takes one numeric argument and returns the square root of the argument.

   Write a JavaScript program to display the value of PI and the square root of 24.5.

---

2. Write a JavaScript program which displays your full name in blue color with screen color orange.

---

3. Write the values of the following expressions:

12+3+"West"	
12+3+"West"+3+4	
12+"West"+4*3	
12+3+"West"+3*5	
12+3+"West"+1+2+3	

---

4. Write the values of:

parseInt("23")	
parseInt("43.23")	
parseInt("12B4")	
parseInt("a23")	
parseInt("2356.7")	

---

5. Write the values of:

parseFloat("2305")	
parseFloat("00.23")	
parseFloat("12B4,56")	
parseFloat("a23")	
parseFloat("2356.7")	

---

6. Write a JavaScript program to display an alert box with the message, Good Morning!

---

7. Write a JavaScript program, when you run, it will show prompt dialog boxes twice. It assigns the floating-point (decimal) values you type and converts the strings into numbers and displays their product (use * for multiplication).

Answers to Module 1 Exercises:
http://vulcan.seidenberg.pace.edu/~nmurthy/JavaScriptExamples/ModuleExerciseAnswers/AnswerstoModule1Exercises.pdf

---

Module 2: Functions

20. Functions in JavaScript

Functions enable us to write certain code once, and use the code several times in the program at different parts.

Each function has a name. Whenever the function's name is used in proper syntax in the program, the code in the function is performed. We say that the program calls the function when it uses the function's name.

Also, each function may have a list of inputs, enclosed in parentheses, following the name. This list of inputs contains the data that the function needs to perform its computation. We call these inputs parameters or arguments.

Functions are the fundamental building blocks in JavaScript.

There are two kinds of functions: (1) Functions which return a value and (2) functions, which don't return a value but perform some actions.

An example of first kind of function is a function, which calculates certain value and returns the value. Examples of second kind is a functions are those just pop up a dialog box or start an animation or just activate printing something. The first kind of function returns an answer to the point in the program that called the function. This is done by including the **return** keyword in the function. The second kind of functions will not have the **return** keyword.

---

21. Functions that return a value

Syntax for writing functions, which return a value:

function *name(parameter list)*{
  computation code
  return *value*;
}

The keyword **function** is required, followed by a name – any name made up by you. Following the name of the function, you have list of variable names (called parameters) within parentheses. The computation result of the computation, which the function has to return is *value.* The function uses the values of the parameters in the computation.

In the body of such function, you actually compute some value and "return" it to be used at the point where it is called.

Example:
A simple example to make the points clearer. This example computes the average of three numbers and returns the average.

```
function average(num1, nm2, num3){
  sum = num1+num2+num3;
  return sum/3;
}
```

Here average is the name of the function. It has three parameters. Parameters will have values when you call the function (we will see below). The body of the function, calculates the average using the values of the parameters, and returns the average value.

Once, you write a function, you can use it any number of times from other places in the program. Using a function is called "calling the function." When you call a function, use the name of the function and provide values for the parameters.

For example,
average(11,15,25) is a call to the function. Here you have provided 11, 15 and 25 for the three parameters num1, num2, and num3 respectively. The function body uses three values for num1, num2 and num3 and computes the average.

As an example, you can call the average() as shown in below:
document.write(average(11,15,25));
This actually displays 17 (the average of 11, 15 and 25).

Here are two more examples of function calls,
document.write("The average of 27, 17 and 14 is ",average(27,17,14));
answer = average(34,28,10);

Example

In the following example the function interest() computes the interest using the arguments passed (two parameters: principal and interest rate) and returns the computed value. The function is called from the "main" program in the **document.write()** statement.

Practice Example 21:

```
<html><head><title> JavaScript Example: interest computation</title>
<script>
function interest(principal, rate) {
   return principal * (rate / 100)
}
</script> </head> <body><h3> Testing interest function </h3>
<script>
   document.write(7,"% interest on $",765," is ",interest(765, 7))
</script>
<H2> We Are Done </H2>
</body></html>
```

Try It: http://vulcan.seidenberg.pace.edu/~nmurthy/JavaScriptExamples/Example21.html

---

22. Functions that do not return a Value

Sometimes we don't want a function to return a value, instead we want the function to do some task such as printing. In such functions do not use the **return** statement in the body of the function.

Practice Example 22:

```
<html><head><title> JavaScript Example: Another function example</title>
<script>
function address(name,street,city,state)
{
   document.write("Mailing address is <hr>")
   document.write(name,"<p>")
   document.write(street,city,state,"<p>")
   document.write("Thank you <p>")
}
</script></head>
<body><h3> address function</h3>
<script>
   address("John Smith "," 31 Post Road ","Pleasantville","NY10570")
</script>
<H2> Done </H2> </body> </html>
```

Try It: http://vulcan.seidenberg.pace.edu/~nmurthy/JavaScriptExamples/Example22.html

Note:

- Notice how the **return** keyword is absent in the function and the function is printing the address.
- Also notice how the function is called from the "main" part and is sending values for the parameters of the function.
- It is important to understand that a call to a function, which does not return a value is a standalone statement. And a call to a function, which returns a value is part of another statement. Observe this point in examples.

---

23. Various comments about functions

- Function definitions are written within <script>…</script> and are generally included within in the <head>…</head> portion of the HTML document.
- Primitive parameters (such as a number) are passed to functions **by value**; the value is passed to the function, but if the function changes the value of the parameter, this change is not reflected globally or in the calling function.

- If you pass an object (i.e. a non-primitive value, such as `Array` or a user-defined object) as a parameter, and the function changes the object's properties, that change is visible outside the function.

- Unlike some other languages (such as Java), a variable created in a function, by default is global. This means, a variable created in a function can be accessed outside the function (this is not the case in other programming languages). In order to make the variable purely local, "declare the variable" with the keyword **var**. That is, the first time the variable is used, prefix it with the keyword **var** followed by a space.

  Example:
  var sample = 10;
  Now, sample is a local variable: it is not available outside the function.

---

24. JavaScript as a link

It is possible to use a JavaScript function as a link from a regular HTML page.

Syntax:
<A HREF="JavaScript:*functionname*()"> text </A>
Where *functionname*() is a function call.
When the link is clicked, the function will be executed.

Practice Example 24:

```
<html><head>
<title>JavaScript function as a link</title>
</head><body>
   <A HREF="javascript:alert('You clicked Me!')">Click me</A>
</body></html>
```

In this example, I have used an **alert**() command, instead of a function.

Try It: http://vulcan.seidenberg.pace.edu/~nmurthy/JavaScriptExamples/Example24.html

---

25. Errors in a JavaScript program

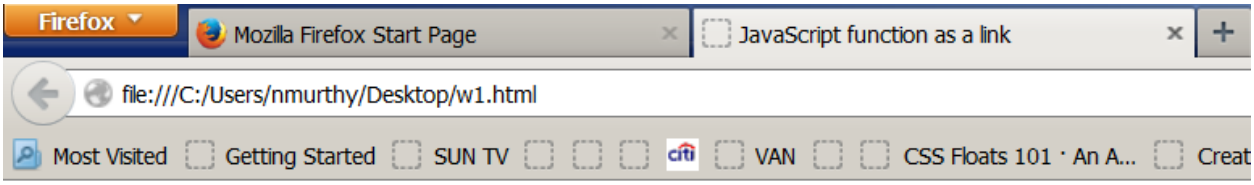
When there is an error in a JavaScript program, by default the browser does not display the error. To see errors in a JavaScript program, in Firefox, press CTRL + Shift + k to see "error console."

Example:
Take the previous example and make mistake: type alrt instead of alert:

```
<html><head>
<title>JavaScript function as a link</title>
</head><body>
```

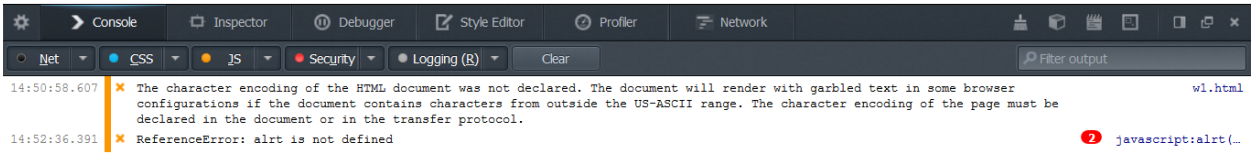
```
    <A HREF="javascript:alrt('You clicked Me!')">Click me</A>
</body></html>
```

When you load this in Firefox, you will see the following screen with a link:



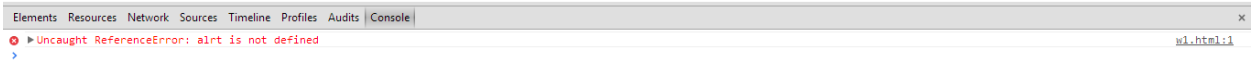
When you click the link, nothing happens on the screen. Clearly there is an error. To see the error, press CTRL+Shift+I. You will see the error console:

Click JS button. You will see:



Notice the error message, alrt is not defined.
Ignore the other comment above it.

CTRL+Shift+I works in Chrome as well.



---

26. More Practice Examples

Practice Example 26:
This example asks the user for a color name and displays a JavaScript link. When the user clicks the link, it changes the background to the input color.

```
<html>
<head>
<title>JavaScript function as a link</title>
<script>
function changeColor(colorName)
{
  document.bgColor = colorName;
}
</script>
</head>
<body>
<h3> COlor will change</h3>
<script>
```

```
colorValue = prompt("Please input a color name","");
</script>
<A HREF='JavaScript:changeColor(colorValue)'>change background to the input color</A>
</body>
</html>
```

Try It: http://vulcan.seidenberg.pace.edu/~nmurthy/JavaScriptExamples/Example26.html

---

Practice Example 26A:

This program includes a function, return Interest(principal, rate) - which returns the interest on an amount of money (principal) at a specified rate of interest. The program asks the user (using prompt boxes) for the principal and rate, calls the function, and displays the answer.

PracticeExample26A.html

```
<html><head><title> JavaScript Example: interest computation</title>
<script>
function returnInterest(principal, rate)
{
  return principal * (rate / 100)
}
</script>
</head>
<body>
<h3> Testing interest function </h3>
<script>
  principal = prompt("Type the principal","");
  interestRate = prompt("Type interest rate","");
  document.write(interestRate,"% interest on $",principal," is ", returnInterest
(principal,interestRate))
</script>
<H2> We Are Done </H2>
</body>
</html>
```

Try It: http://vulcan.seidenberg.pace.edu/~nmurthy/JavaScriptExamples/Example26A.html

---

Practice Example 26B

This program includes a function area(r), which calculates the area of a circle of radius r (input parameter) and returns the area. The program asks the user for the radius value, calls the function and prints the area of the circle of the given radius. The formula for the area is: area = πr^2. To get the value of π, use Math.PI.

PracticeExample26B.html

```
<html>
<head><title>Area of a circle</title>
<script>
function returnArea(r)
{
  return Math.PI * r * r;
}
</script>
</head>
<body>
<h3> area calculation</h3>
<script>
radius = prompt("Input radius","");
document.write("The area of a circle with radius ",radius," is ",returnArea(radius));
</script>
</body>
</html>
```

Try It: http://vulcan.seidenberg.pace.edu/~nmurthy/JavaScriptExamples/Example26B.html

---

Module 2 Exercises

1. Write a function, getSaleprice(original, salePercent), takes two arguments: the first one is the original price of an item and the second is the percentage of discount. The function returns the sale price of the item. Call this function twice to illustrate the use of it.
2. Write a function displayInfo(name, telNum), which takes two parameters: the first one for the name of a user and the second one for his/her telephone number, and displays the two values (name and telephone number). Write a JavaScript code, which uses two prompt dialog boxes to receive the name and telephone number of a user, passes the two values to the displayInfo() function to display the values.
3. Write a function displayRectInfo(width, height), which takes two parameters: the first one for wirth of a rectangle and the second for the height of the rectangle, and displays area and perimeter of the rectangle. Write a JavaScript code, which uses two prompt dialog boxes to receive the width and height of a rectangle, passes the two values to the displayRectInfo() function to display the area and perimeter of the rectangle.
4. Write a function displayCirInfo(radius), which takes one parameter: the radius of a circle, and displays area and circumference of the circle. Write a JavaScript code, which uses a prompt dialog box to receive the radius of a circle, passes the value to the displayCirInfo() function to display the area and perimeter of the circle. Formulas for area and circumference of a circle of radius r: πr^2 and $2\pi r$.

5. The cost of a ticket price for a sports event is $26 for adults and $15 for children. Write a function, getCost(adults, children), takes two arguments: the first one the number of

adult tickets and second one is the number of child tickets, and return the total cost of tickets. Call this function twice to illustrate the use of it.

6. Write a function displayInfo(), which does not take any arguments, but displays your name, course number and the semester. Call the function, as a link, that is, provide a link and when the user clicks the link, the function is called.

Answers to Module 2 Exercises:

http://vulcan.seidenberg.pace.edu/~nmurthy/JavaScriptExamples/ModuleExerciseAnswers/AnswerstoModule2Exercises.pdf

---

Module 3: Data Types and Arithmetic Operators

27. Data Types

All the data used in JavaScript is divided into four categories, called data types:

1. Numbers: All numbers come under this category.
Examples of numbers: 12, 4.21, 0.

2. Boolean: Only two values belong to this category: true and false.

3. Strings: A bunch of characters enclosed in double quotes or single quotes is called a string.
Examples of strings: "ABCD124", "New York", 'Westchester'.

4. null: This is a special keyword denoting null value. The word is case sensitive. It is NOT Null or NULL.

Note:
Data types are important in languages like Java and Pascal. In JavaScript data types are not very important. It is said, "JavaScript is not strongly typed."

Note:
There is another data type in JavaScript: undefined. When a value is not defined, it is said to be undefined.

Note:
The null value behaves as 0 in numeric contexts and as false in Boolean contexts.
The undefined behaves like false in Boolean contexts. It is NaN in numeric contexts.

---

28. Identifiers

Symbolic names like variable names and function names are called identifiers. These are names created by the programmer.

In JavaScript, an identifier
- is a sequence of letters, digits and underscore (_);
- must start with a letter (or an underscore or $);
- must not contain a space;
- cannot be a keyword.

Following are examples of valid identifiers:
sam
example4
sample_5
_trial
Following are examples of invalid identifiers:

6problem (invalid because it starts with a digit)
Example 43 (Invalid because it Includes a space)
problem#4 (Invalid because it Includes a special character)

---

29. Commonly used arithmetic operators

JavaScript provides the following commonly used arithmetic operators:

- **Arithmetic Operator +**

The binary + is the usual addition operator.
A + B yields the sum A + B.

- **Arithmetic subtraction operator –**

The binary – is the usual subtraction operator.
A – B yields the difference A – B.

- **The multiplication operator ***

The binary * is the usual multiplication operator.
A * B yields the product A * B.

- **Division operator /**

The binary / is the division operator.
Example:
½ yields 0.5
25/ 4 yields 6.25.

- **Increment operator ++**

The unary increment operator ++ is used with a variable, either postfix or prefix. The operator ++ increments the value of the variable by 1.

A++ or ++A increments the value of the variable A by 1.

Example:
A = 7; A++; A value now is 8.
A = 7; ++A; A value now is 8.

The difference between postfix and prefix usage of ++

There is an important difference between postfix and prefix usage of ++.
A++ means "use the current value of A first and then increment A."
++A means "increment A first and then use it."

Example:
Consider the statements:
A = 7;
B = A++;
The value of B is 7, and the value of A is 8.

Example:
Consider the statements:
A = 7;
B = ++A;
The value of B is 8, and the value of A is 8.

Exercise:
Consider the statements:
x = 3;
y = 5;
z = x++ * y++;
What is the value of z?

Exercise:
Consider the statements:
x = 3;
y = 5;
z = ++x * y++;
What is the value of z?

When can you NOT use ++?

You can only use ++ with variables. It is illegal to use ++ with a constant:
4++ and ++4 are illegal.

It is illegal to use ++ with expressions:
(x+y)++ and ++(x+y) are illegal.

- **Decrement operator –**

The unary decrement operator – is used with a variable, either postfix or prefix. The operator – decrements the value of the variable by 1.

A–or –A decrements the value of A by 1.

Example:
A = 7; A--; A value now is 6.
A = 7; --A; A value now is 6.

Exercise:
Assume x is 15.
What is the value of x, after each of the following statements?
x--;
--x;

The difference between postfix and prefix usage of –

There is an important difference between postfix and prefix usage of --.
A–means "use the current value of A first and then decrement A."
--A means "decrement A first and then use it."

Example:
Consider the statements:
A = 7;
B = A--;
The value of B is 7, and the value of A is 6.

Example:
Consider the statements:
A = 7;
B = --A;
The value of B is 6, and the value of A is 6.

Exercise:
Consider the statements:
x = 3;
y = 5;
z = x-- * y--;
What is the value of z?

Exercise:
Consider the statements:
x = 3;
y = 5;
z = --x * y--;
What is the value of z?

When can you not use --?

You can only use – with variables. It is illegal to use – with a constant:
4–and –4 are illegal.

It is illegal to use – with expressions:
(x+y)–and –(x+y) are illegal.

30. Modulus operator %

The binary % is the remainder operator.
If A and B are integers, A % B produces the remainder obtained when A is divided by B.

Examples:
17 % 5 is 2
10 % 3 is 1
200 % 10 is 0
278 % 2 is 0
271 % 2 is 1
7 % 10 is 7
0 % 5 is 0

---

31. Assignment operator

The usual = is the assignment operator. The left side of = must be a variable and the right side can be a constant, a variable or an arithmetic expression.

The value of the right side is computed and is assigned to the variable on the left side.
Example:
number1 = 3+2*5
city = "New York"
firstName = "John "
lastName = "Smith"
name = firstName+lastName

Note:
You should be aware of a special kind of assignment statement.
In many situations, we use the variable appearing on the left side in the expression on the right side. In this case, use the current value of the variable in the expression on the right and the computed value becomes the new value of the variable.

Example:
Assume x = 4.
Consider x = 2*x – 3
Use the current value of 4 for x and do computation 2*4-3, which is 5. Thus 5 will be the new value of x.

Note: Unlike other leading languages, such as Java, JavaScript is a dynamically typed language. That means you do not have to specify the data type of a variable when you declare it, and data types are converted automatically as needed during script execution. The same variable can be assigned a number at one point and a string at the other.

Example:
var value = 54;

At a later time, you can assign the variable value a string:
value = "New York";

Shorthand assignment:
An assignment of the form
variable op= expression;
is a shorthand assignment, and it is equivalent to
variable = variable op expression;
Where *op* is any operator.

The following table shows some of the shorthand assignments:

Shorthand operator	Meaning
x += y	x = x + y
x -= y	x = x – y
x *= y	x = x * y
x /= y	x = x / y
x %= y	x = x % y

Example:
Consider the following statements:
x = 9;
x += 4; //is equivalent to x = x+4;
x now is 13.

Exercise:
Consider the following statements:
x = 9;
x *= 3;
What is the value of x?

---

32. Operator precedence

Consider the following expression with several arithmetic operators:
3 + 2 * 5

We all know that multiplication is done before addition. Thus in the expression 2 * 5 is done first resulting in 10, and then 3+ 10 is performed. The final result will be 13.

Some basic precedence rules:

- Multiplication is said to have higher precedence over addition.
- Multiplication, division and modulus all have the same precedence.
- Addition and subtraction have the same precedence.
- Each of *, / and % has higher precedence over each of + and -.

Example:
3 + 2 * 5 + 4:
first: 2*5 = 10, next 3+10 = 13, next 13+4 = 17.
Example:
7%3+5:
first: 7%3 = 1, next 1+5 = 6.

Example:
8+10/5-4:
first: 10/5 = 2, next 8+2 =10, next 10-4=6.

Module 3 Exercises

1. Write the values of the following arithmetic expressions:

Arithmetic expression	Value
17%4	
17/4	
4+8 / 4-2	
4 *7+2* 5	
1/4*(10+8+6)	
15%3*24	
10+12 % 3	

2. Write the value of x after each assignment statement:
   (If the statement is invalid, just say Invalid)

Assignment statement	x value
a = 4; x = ++a;	
a = 4;x = a++;	
a = 4; b = 3; x = ++a * ++b;	
a = 4; b = 3; x = a++ * ++b;	
a = 4; b = 3; x = a++ * b++;	
a = 4; b = 3; x = ++a * b++;	
x = 7++;	
a=4; b=2; x = ++(a+b);	

Answers to Module 3 Exercises:

http://vulcan.seidenberg.pace.edu/~nmurthy/JavaScriptExamples/ModuleExerciseAnswers/AnswerstoModule3Exercises.pdf

Module 4: Relational Operators and Branching

33. Relational operators

The following operators, called relational operators, are used to compare two items that are ordered. Each of the operators takes two operands - a left operand and a right operand, and returns true or false depending on the result of the comparison.

Operator	Meaning	Comments
==	Equal	True if the two operands are equal; false otherwise.
!=	Not equal	True if the two operands are not equal; false otherwise.
<	Less than	True if the left operand is less than the right operand.
<=	Less than or equal to	True if the left operand is less than or equal to the right operand.
>	Greater than	True if the left operand is greater than the right operand.
>=	Greater than or equal to	True if the left operand is greater than or equal to the right operand.

Note:
- An expression involving arithmetic operators is called an arithmetic expression.
- An expression involving relational operators is called a boolean expression.

---

34. Logical operators

Boolean expressions, typically use relational operators and have either true or false as a value. An example of use of such boolean expression is, to check if age is less than 27, the boolean expression for this would be age<21.

Now suppose you need to check two conditions. For example you need to check if the age is less than 27 and also height is more than 6 feet.

We need logical operators to check multiple conditions.
There are three logical operators:
- Logical NOT - symbol !
- Logical AND - symbol &&
- Logical OR -- symbol ||

As relational operators work on arithmetic expressions, logical operators work on boolean expressions. An expression which involves a logical operator also returns true or false. An expression which contains a logical operator is called a compound boolean expression, sometimes just a boolean expression.

Thus a boolean expression is an expression with relational operators and/or an expression with logical operators.

---

35. The logical operator NOT

The symbol ! is used for logical operator NOT.
The operator ! takes one boolean operand on the right.
The following table (called the truth table) shows the effect of the not operator:

!(true) is false
!(false) is true

Note: Arithmetic operators have higher precedence over logical operators.

Example:
Assume a = 7, b = 4 and c= 10.
!(a < b) is true.
!(a < b+c) is false
!(a*b >= 2*c) is false
!(a+b != 0) is false.
!(a+b+c == 21) is false

---

36. The logical operator AND

The symbol for logical operator AND is &&
The operator && takes two operands - a left operand and a right operand, each is a boolean condition.

The following is the truth table for logical AND:

(true) && (true) is true
(true) && (false) is false
(false) && (true) is false
(false) && (false) is false

Notice that a logical && returns true only when both of its operands are true. In all other cases it returns false.

Example:
Assume a = 7, b = 4 and c= 10.
(a<c) && (a+b !=11) is false.

(2*c < a+b+c) && (a+b != 0) is true.
(a - b <= a-c) && (c <=10) is false.

---

37. The logical operator OR

The symbol for logical operator OR is || (keymate of \)
The operator || takes two operands - a left operand and a right operand, each is a boolean expression.

The following is the truth table for logical OR:

(true) || (true) is true
(true) || (false) is true
(false) || (true) is true
(false) || (false) is false

Notice that a logical || returns false only when both of its operands are false. In all other cases it returns true.

Example:
Assume a = 7, b = 4 and c= 10.
(a<c) || (a+b !=11) is true.
(2*c < a+b+c) || (a+b != 0) is true.
(a - b <= a-c) || (c <10) is false.

JavaScript Special Operators
The following are some of the special operators provided by JavaScript:

- **Conditional operator**

The conditional operator, also called ternary operator, is a special operator with two symbols ? and :.

Syntax:
condition **?** *expression1* **:** *expression2*

Where *condition* is a boolean expression.

The ternary expression returns *expression1*, if condition is **true**; otherwise returns *expression2*.

Example:
Consider the following statements:
x = 15;
answer = x<=10 ? 100: 200;
Assigns answer value 200.

Exercise:
Consider the following statements:
a = 3;
b = a<=10?0.25:1.8;
What is the value of b?

---

38. The if statement

This is a very important JavaScript statement.

The **if** statement is a branching instruction.

Syntax for if:
if (*condition)*
{
   statements
}
Where *condition* is a boolean expression or a compound boolean expression.
If *condition* is true, the statements between { ... } will be executed. The group of statements between { ... } is called the if block. Any statements may be placed in this block, including other if statements.

Example:
```
if ( x <= 14 )
{
   statement 1
   statement 2
   statement 3
}
```
If the condition x<=14 is true all the statements in the **if** block will be executed.

Note: The following values will evaluate to false:

- false
- undefined
- null
- 0
- NaN
- the empty string ("")

All other values, including all objects evaluate to true when passed to a conditional statement.

---

39. if ... else statement

An **if** statement can have an **else** clause. This will enable a two way branch.

Syntax for if ... else:

```
if (condition )
{
   statements1
}
else
{
   statements2
}
```

If condition is true, statements1 (if block) will be executed and statements2 will not be executed; otherwise statements1 will not be executed and statements2 (else block) will be executed.

Example:

```
if(x<=14)
{
   statement 1
   statement 2
   statement 3
}
else
{
   statement 4
   statement 5
   statement 6
}
```

In this example, if x has a value that is less than or equal to 14, then statements 1, 2 and 3 will be executed and statements 4, 5 and 6 will not. If x has a value greater than 14, statements 1, 2 and 3 will be skipped and statements 4, 5 and 6 will be executed.

Note:
The **if** block and/or **else** block can have only one statement. In such a case, the braces can be omitted. In this case, the statement must appear on the same line as the **if** and/or else.

---

40. The switch Statement

The switch is a multi-branch statement. Depending on the value of an expression, you can choose to execute any of a number of sets of statements.

A **switch** statement allows a program to evaluate an expression and attempt to match the expression's value to a case label. If a match is found, the program executes the associated statement. A `switch` statement looks as follows:

Syntax:

switch (*expression*)
{
  case *label1*: { *statements* ; **break**;}
  case *label2*: { *statements*; **break**;}
  …
  …
  case *labelk*: {*statements*; **break**;}
  default: {*statements* }
}

The **switch** statement matches the value of *expression* to labels, and executes the statements corresponding to the matching label. If no label matches expression, the statements corresponding to default label will be executed.

Note:
1. Labels must be constants.
2. The **default** label is optional.
3. If you do not place a **break**; statement at the end of the blocks, control will fall to the next set of statements.
4. The braces grouping statements is optional.

Example:

```
switch (grade)
{
  case 'A': {document.write("Excellent score .. You have an A <P>"); break;}
  case 'B': {document.write("Very good score .. You have a B <P>"); break;}
  case 'C': {document.write("Good score .. You have a C <P>"); break;}
  case 'D': {document.write("Passing score .. You have a D <P>"); break;}
  default: {document.write("Sorry, you have not completed the course.<P>"); }
}
```

Depending on the value of the char variable grade, appropriate group of statements will be executed.

Practice Example 40A:
Write a program to input a number representing the number of doors in a car and then print a comment depending on the input. If the value input for doors is 2, print "Small car", if it is 4, print "Four door sedan", if it is 5, print "Hatchback", if it is 6, print "Family Van", if it is 8, print "Bus", otherwise print "It is an alien ship."

```
<html>
<head><title>Another example using switch</title></head>
<body>
Input the number of doors in the input box.  The program prints a comment.<P>
<script>
var doors = prompt("Input number of doors","");
```

```
switch (doors)
{
case "2":{document.write("Small car <P>"); break;}
case "4":{document.write("Four door sedan <P>"); break;}
case "5":{document.write("Hatchback <P>"); break;}
case "6":{document.write("Family van <P>"); break;}
case "8":{document.write("Bus <P>"); break;}
default: {document.write("It is an alien ship <P>"); break;}
}
</script>
</body>
</html>
```

Try It: http://vulcan.seidenberg.pace.edu/~nmurthy/JavaScriptExamples/Example40A.html

---

Practice Example 40B:
Write a program to input four integers between 1 and 6 (inclusive) and print the nature of numbers input as one of the following categories: (1) Four of a kind (2) Three of a kind (3) Two pairs (4) One pair (5) All distinct

```
<html>
<head><title>Four numbers </title></head>
<body>
<h3>Input four numbers between and 6 in the four prompt boxes:</h3>
<script>
var a = prompt("Input a number between 1 and 6","");
var b = prompt("Input a number between 1 and 6","");
var c = prompt("Input a number between 1 and 6","");
var d = prompt("Input a number between 1 and 6","");
document.write("Four numbers input: ",a," ",b," ",c," ",d);
document.write("<P>");
var counter = 0;
if (a==b) counter++;
if (a==c) counter++;
if (a==d) counter++;
if (b==c) counter++;
if (b==d) counter++;
if (c==d) counter++;
     switch (counter)
     {
       case 6: {answer = "FOUR OF A KIND";break;}
       case 3: {answer = "THREE OF A KIND";break;}
       case 2: {answer = "TWO PAIRS";break;}
       case 1: {answer = "ONE PAIR";break;}
       case 0: {answer = "ALL DISTINCT";break;}
```

```
    }
document.write("Result: ",answer);
</script>
</body>
</html>
```

Try It: http://vulcan.seidenberg.pace.edu/~nmurthy/JavaScriptExamples/Example40B.html

---

Including multiple labels

You can include several constants in the same label. The syntax is shown in the following example.

Example:

```
switch (value)
{
    case "1": case "2":{System.out.println("You input 1 or 2");break;}
    case "3":case "4": case 5:{System.out.println("You input 3 or 4 or 5");break;}
    case "6": case "7":{System.out.println("You input 6 or 7");break;}
    default: {System.out.println("default");}
}
```

The meaning is self-explanatory.

Note:

Clearly you cannot place the same constants on two labels.

Practice Exercise:

Write a program to input a month, as a number 1-12, and a year as a four-digit number, print the number of days in the month. Make sure you take care of the leap year appropriately. A year that is divisible by 4 but not by 100 unless it is by 400 is called a leap year.

Module 4 Exercises

1. Assume x = 3 and y = -2. Write the value of each of the following boolean expressions (write true or false).

Boolean expression	**true or false**
2*x-3*y <= x+y	
!(x+y <= 2.5)	
x+y<2.5 && 2*x ==14	
x+y<2.5 || 2*x ==14	
2*x <= x*2	

---

2. Assume x = 4.21, y = 15.2 and z = 0. Write the value of each of the following boolean expressions (write true or false).

Boolean expression	**true or false**
(x <=10)&&(y<10)&&(z>=0)	
(x<5)&&(y<20)&&(z<10)	
(x<5)||(y<20)||(z<10)	
(x>4)||(y<20)&&(z<10)	
(x <=10)&&(y<15)||(z<0)	

3. Write a program that inputs an integer using a prompt dialog box and prints whether the number is odd or even.

4. Write a function, which takes three parameters – three numbers and returns the largest of the three. Input three numbers using three prompt dialog boxes, pass the three numbers to the function to get the largest of the three. Display the largest number.

5. Each student in a class takes five exams. He/she passes the course if he/she gets at least 60 points on at least three of the exams. Write a function that takes five numeric parameters for the five exam scores and returns boolean true if the student has passed the course, false otherwise. Input five exam scores using five prompt dialog boxes, pass the five numbers to the function to get the result. Display the student passed or failed the course.

6. First understand the definition of a leap year: A leap year is a year which is divisible by 4 but not 100 unless it divisible by 400.

   Write a program which inputs (using a prompt dialog box) a year as a four-digit integer and prints if it is a leap year or not.

7. Individual roses cost $1.50 each; a bunch of 10 roses costs $12.00. Write a function to input the number of roses a customer likes to buy as a parameter and return the total cost.

   Note: You have to give the customer the bunch price first. For example, if a customer buys 34 roses (3 bunches + 4 individual), he pays @ $12.00 for 3 bunches and then @ $1.50 for 4.

Module 5: Loops

41. What are loops?

You can make the program repeatedly execute a set of statements. The language construct that enables this is called a loop. The set of statements, which is repeatedly executed, is called the body of the loop.

JavaScript provides three kinds of loops: the **for** loop, the **while** loop and the **do** loop.
We will discuss **for** and **while** loops in detail and skip discussion on **do** loops.

---

42. The for loop

The **for** loop is JavaScript statement enables you to specify how many times the body of the loop has to execute.

Syntax:
for (*expression1*; *expression2*; *expression3*)
{
   statements
}

Where,
expression1 is normally used to initialize a variable called the control variable.
expression2 is a boolean expression.
expression3 is normally used to modify the control variable.

This is how a **for** loop works:

step 1: initializes the control variable in *expression1*
step 2: if the *expression2* is true, executes the body of the loop.
step 3: updates the control variable in *expression3*.
Repeats steps 2 and 3, until expression2 becomes false.

Example:
for (n = 0; n <=100 ; n++)
{
   Body of loop
}
Here n is the loop control variable.
The body of the loop is executed for each value of n=0, 1, 2, ..., 100.

Note: If the body of the loop has only one statement, the braces ({...}) can be omitted.

Note: n++ is equivalent to n = n+1.

Example:
This example uses a for loop to compute sum 1 through 10.

```
sum = 0;
for (i=1;i<=100;i++){
  sum = sum + i;
}
```

Example:
This example uses a for loop to compute the sum of all even numbers and sum of all odd numbers between 1 and 1000.

```
<html>
<head><title>Loop example </title>
</head>
<body>
<h3>loop example</h3>
<script>
evensum = 0;
oddsum = 0;
for(i=1;i<=1000;i++){
  if(i%2 == 0) evensum += i; else oddsum +=i;
}
document.write("Even sum 1 through 1000 is ",evensum);
document.write("<p>Even sum 1 through 1000 is ",oddsum);
</script>
</body>
</html>
```

Example:
It is possible that a loop is never entered.
Notice in the following example, the loop is never entered.

```
for ( i = 100; i < 100; i++)
{
    .............
}
```

There is nothing wrong with this. This will compile but the body loop is not executed even once.

Example:
Infinite loop.
Something every programmer must avoid is the dreaded infinite loop.

A loop that never terminates is called an infinite loop.
The following example illustrates an infinite loop:
Notice in the following example the loop never terminates:

```
for (int k = 0; k<=100; k--)
{
    ..............
}
```

The programmer has to take the responsibility that such a loop never exists in the program.

Example:
JavaScript object Math (which we have used before) provides a method to generate random numbers.

Math.random()
Generates a number like: 0.49957117391750216.

You have to do some arithmetic manipulation if you want a random number between two numbers.

For example,
Math.floor((Math.random()*10)+1
Generates a number between 1 and 10 (inclusive).

Math.floor((Math.random()*6)+1)
Generates a random number between 1 and 6 (inclusive).

The following code generates 10 sets of 4 numbers between 1 and 6.

```
for(i=1;i<=10;i++){
document.write(Math.floor((Math.random()*6)+1),"  ");
document.write(Math.floor((Math.random()*6)+1),"  ");
document.write(Math.floor((Math.random()*6)+1),"  ");
document.write(Math.floor((Math.random()*6)+1),"  ");
document.write("<p>");
}
```

Output will be something like this:

```
1 2 6 6
6 1 5 3
5 4 2 2
3 4 6 3
2 6 6 6
1 2 6 2
5 3 2 2
```

6 2 2 2
2 3 5 5
1 1 6 5

Example:
The following example generates 10 sets of four of numbers between 1 and 6, and displays each time what kind of numbers were generated (four of a kind, three of a kind, two pairs, one pair or all distinct). The example uses a function.

```
<html>
<head><title>Four Numbers </title>
<script>
function whatKind(a,b,c,d){
      document.write("Four numbers are ",a," ",b," ",c," ",d,"  ");
        counter = 0;
        if(a == b) counter++;
        if(a == c) counter++;
        if(a == d) counter++;
        if(b == c) counter++;
        if(b == d) counter++;
        if(c == d) counter++;
        switch (counter)
        {
            case 6: {answer = "FOUR OF A KIND";break;}
            case 3: {answer = "THREE OF A KIND";break;}
            case 2: {answer = "TWO PAIRS";break;}
            case 1: {answer = "ONE PAIR";break;}
            case 0: {answer = "ALL DISTINCT";break;}
        }
      document.write(answer);
}
</script></head>
<body>
<h3>Four Numbers</h3>
<script>
for(i=1;i<=10;i++){
num1=Math.floor((Math.random()*6)+1);
num2=Math.floor((Math.random()*6)+1);
num3=Math.floor((Math.random()*6)+1);
num4=Math.floor((Math.random()*6)+1);
whatKind(num1,num2,num3,num4);
document.write("<p>");
}
</script>
</body></html>
```

---

43. The while loop

The while loop controlled by a boolean expression. As long as the boolean expression is true, the body of the while loop keeps executing.

Syntax:
while (*boolean expression*)
{
   statements
}
Where *boolean expression* is an usual boolean expression with relational operators or a compound boolean expression with logical operators.

The condition is tested to see whether it is true or false. If it is true, the body of the loop is executed and then the entire process is repeated. If the condition is false, the body of the loop is not executed, and JavaScript passes to the statement after the while loop.
The body of the loop is repeatedly executed as long as *boolean expression* is true.

Example:
```
n = 0
while (n <=100)
{
   body of the loop
   n=n+i
}
```
Notice that the execution of the loop is controlled by the variable n. To start with the value of n is 0. The condition, n<=100, is true. The body of the loop is executed. In the body of the loop the value of n is modified. Loop gets executed. But ultimately, n becomes 101 in which case the condition becomes false and loop stops.

Important note:
Observe that if you forget to modify n within the body of the loop, the condition never becomes false. This leads to a very undesirable situation that the loop never stops. This situation is called infinite loop. Programmers must make sure that they don't have infinite loops in their programs.

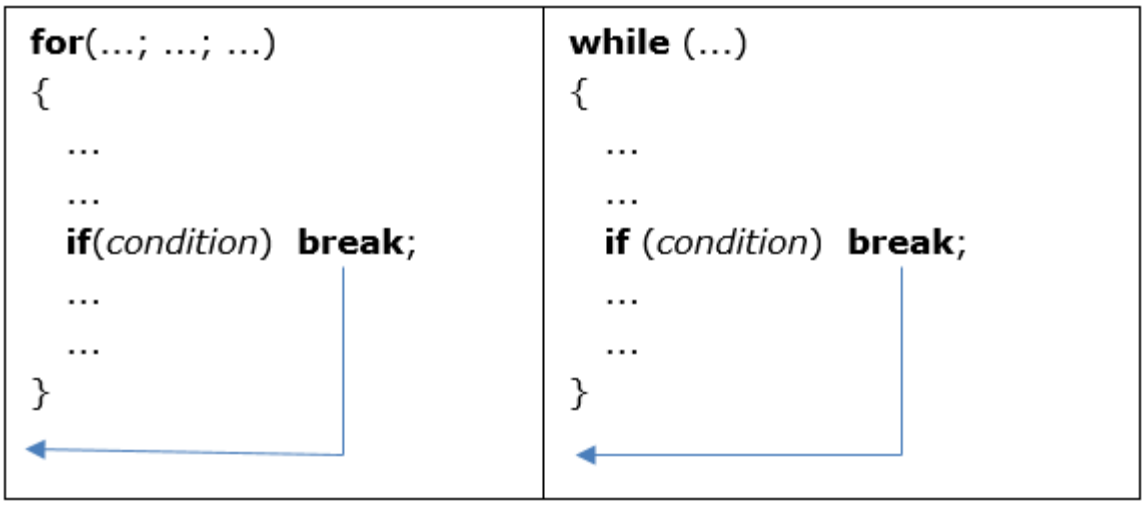
---

44. The break statement

The **break** statement is included in loops: **for** and **while** loops.
When a break statement is encountered in a loop, control comes out of the loop. In other words, a break will terminate the loop.

Generally, a break statement is used with an **if** statement:

When the condition in **if** is true, **break** is executed which results in control coming out of the loop.



In each case, if the *condition* is true, the **break** statement is executed resulting in the termination of the loop. (i.e., control comes out of the loop to the statement indicated by the arrow.)

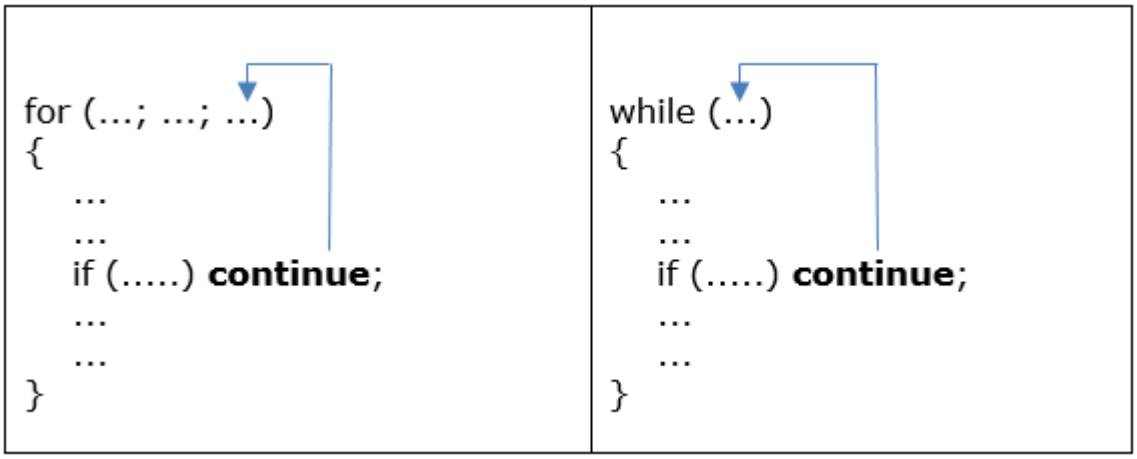
---

45. The continue statement

The continue statement is used in loops: both **for** and **while** loops.

A **continue** statement can only be used in a loop. When **continue** statement is encountered in a loop, control jumps to the beginning of the loop skipping the rest of the loop for the current iteration.

A continue statement is normally used with an **if** statement. When the condition in **if** is true, **continue** is executed which results in control coming to the beginning of the loop skipping the rest of the loop for the current iteration:



Module 5 Exercises

1. How many times does the following code display New York in each case?

for(int i = 10; i<10; i++) document.write("New York");	
for(i = 10; i<=10; i++)document.write("New York");	
for(i = 1; i<=10; i++)document.write("New York");	
for(i = 0; i<=10; i++)document.write("New York");	
for(i = 0; i<10; i++)document.write("New York");	
for(i = 1; i<10; i++)document.write("New York");	
i=0; while (i <= 100) document.write("New York");	

2. Consider the following code segment:

```
A=0; B=0;
for( i = 1; i<=10; i++)
     A += i;
     B += A;
document.write("A = "+A);
document.write("B = "+B);
```

3. What is wrong with the following code?

```
sum = 0;
      for (i = 1; i<=10; i++);
      sum = sum + i;
      document.write("Sum = "+sum);
```

4. Write program which finds the sum of all integers 1000,1005,1010,…,10000.

5. Write a program which generates 1000 random integers between 1 and 10, and finds how many are less than or equal to 5 and how many are more than 5.

6. The formula Math.floor(100*Math.random())+1 generates a random number between 1 and 100. Write a program to play a number guessing game. The program generates a number between 1 and 100, and prompts the user to guess the number. If the user guesses a number larger than the generated number, the program hints "Your guess is larger"; otherwise it says "Your guess is smaller". The program gives 10 chances to the user. If the user does not guess in 10 attempts, the program prints the right number and terminates. (Use prompt dialog boxes.)

7. Write a program that helps a first grader to practice addition problems. The program displays an addition problem of two one-digit numbers, and asks the student to type the answer. The two numbers are randomly generated using the formula Math.floor(10*Math.random())+1. Your program checks the answer and announces if it is correct or wrong. After ten problems, the program displays the score. (Use prompt dialog boxes.)

8. Write a program that helps a first grader to practice subtraction problems. The program displays a subtraction problem of two one-digit numbers, and asks the student to type the answer. The two numbers are randomly generated using the formula Math.floor(10*Math.random())+1. Make sure the first number is larger than or equal to the second number, so that the answer is never negative. Your program checks the answer and announces if it is correct or wrong. After ten problems, the program displays the score. (Use prompt dialog boxes.)

9. Write a program that plays a word game. The program displays (using a prompt dialog box) a clue to a word, and in a text box within the prompt box, the first two or three letters of the word. If the user inputs the wrong answer, display the right answer. After providing feedback, display the next clue until all 10 words have been displayed. Finally score the correct answers. Keep the question and answers in an array.

10. Write a program which generates 1000 integers between 1 and 100, using a for loop. Count how many odd and how many even numbers were generated. Your loop, however, skips numbers 25, 50, 75 and 90. Display results (how many odd how many even) outside the loop. Also display how many numbers were generated.

Module 6: Arrays

46. What are arrays?

An array is a structure used to hold a list of data. For example, an array of Strings might be used to hold all the names of the customers of a business.

An array of numbers might be used to hold all the product prices of a business.

The distinguishing characteristic of an array is that it is composed of a number of cells that are stored in order. Each cell holds one piece of data. Each cell is associated with an index number called the subscript of the cell. Subscripts start from 0.

Arrays are typically processed by loops.

Arrays in JavaScripts are objects.

---

47. Creating arrays

Basic syntax for declaring an array is by using the keyword **new** and predefined object called **Array**:
arrayname = **new Array()**

This creates an array without any elements in it. Once created this way, you can start using the array.

The Array object has methods for manipulating arrays in various ways, such as joining, reversing, and sorting them. It has a property for determining the array length and other properties for use with regular expressions.

To refer to an array element, you use the usual syntax: *arrayname*[*subscript*].
Subscripts start from 0.

Notice there is no mention of the size of the array in the declaration. The array size grows as you use it.

You can initialize an array at the time of declaration:

You can create an array and initialize it at the same time using any of the following methods:

```
arrayName = new Array("a","b","c","d","e");
arrayName = Array("a","b","c","d","e");
arrayName = ["a","b","c","d","e"];
```

All the three statements are equivalent.
These statements initialize the array with elements:

arrayName[0]=a, arrayName[1]=b, arrayName[2]=c, arrayName[3]=d, arrayName[4]=e.

To use this array, we use a loop. For example, to print out the elements of the array, we use the loop:

```
for(i=0; i<5; i++)
{
   document.write(arrayName[i])
}
```

Notice that in this loop we use a variable i to refer to each element of the array. When this variable is zero, then arrayName[i] refers to arrayName[0], which is the first element of the Array. As i is increased, we refer to successive elements of the array.

Note:

- arrayName = []; creates an array without any elements and size 0.
- arrayName = [10]; creates an array with one element 10; size 1.
- arrayName = Array(10); creates an array without any elements; size 10.
- arrayName = Array(10,20); creates an array with elements 10 and 20; size 2.
- arrayName = Array("a"); creates an array with one element; size 1.

Observe that Array(n) will create an array of size n – not an array of one element n. Array("a") creates an array with one element a.

How do you create an array with one numeric element using Array(). You cannot. To create an array with one numeric element, you must use arrayName = [n].

Array(10.5) results in an error because the size is not an integer.

Note:
If you assign an array to a variable, the value of the variable will be a string, which is a list of all the elements of the array separated by commas.

Example:

```
array1 = ["abc","def","pqr","xyz"];
temp=array1;
```

The value of temp is abc,def,pqr,xyz.

Example

The following example first creates an array with four elements and then prints the elements using a loop.

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

```
<h3>Here are the array elements</h3>
<script>
 array1 = ["aaa","bbb","ccc","ddd"];
 for (i=0;i<4;i++)
 {
   document.write(i," ",array1[i],"<P>");
 }
</script>
</body>
</html>
```

- **You can have a "blank" element:**

When you specify elements separated by commas, you can leave a blank between commas.

Example:

```
array3 = ["xxx","yyy", ,"zzz","ppp"];
```

Notice that array3[2] is undefined.

Example

```
<html>
<head><title>Arrays</title></head>
<body>
<h3>Here are the array elements</h3>
<script>
 array3 = ["xxx","yyy", ,"zzz","ppp"];
 for (i=0;i<5;i++)
 {
   document.write(i," ",array3[i],"<P>");
 }
</script>
</body>
</html>
```

---

48. Arrays are objects

In JavaScript, arrays are objects.
As all objects have properties and methods, every array has properties and methods associated with it.

Selected array properties and methods:

➢ **The array property length**

arrayName.**length** refers to the number of elements in the array.

It is the largest subscript defined + 1.

Example:

```
arr1 = new Array()
arr1[0] = 12
arr1[1] = 23
arr1[10] = 45
```

Length of the array: arr1.length is 11.

Array methods:

Array has a large number of methods. We will explore selected methods here:

- **The concat() method**

The concat() method joins two arrays and returns a new array.

Example:

```
array1 = ["abc","def","pqr","xyz"];
array2 = ["123","456","789"];
array3 = array1.concat(array2);
```

array3 is abc,def,pqr,xyz,123,456,789.

- **The join() method**

The **join()** method takes one argument, which is a delimiter. The **join()** method concatenates the elements of the array into a string with specified delimiter.

Example:

```
array1 = ["abc","def","pqr","xyz"];
temp=array1.join("-");
```

The value of the variable temp is abc-def-pqr-xyz.

- **The push() method**

The push() methods inserts arguments to the end of the array and returns the new length of the array.

Example:

```
array1 = [10,20,30,40];
temp = array1.push(100,200,300);
```

array1 now is 10,20,30,40,100,200,300.
The value of temp (new length of the array) is 7.

- **The pop() method**

The pop() method removes the last element from the array and returns that element.

Example:

```
array1 = [10,20,30,40];
temp = array1.pop();
```

The array now has the elements 10,20,30.
The value of temp is 40.

- **The shift() method**

The shift() method removes the first element from the array and returns that element.

Example:

```
array1 = [10,20,30,40];
temp = array1.shift();
```

The array now has the elements 20, 30, 40.
The value of temp is 10.

- **The unshift() method**

The **unshift()** method inserts the arguments into the array in the beginning of the array, and returns the new length of the array.

Example:

```
array1 = [10,20,30,40];
temp = array1.unshift(100,200,300);
```

The elements of the array now are 100, 200, 300, 10, 20, 30, 40.
The value of temp (the new length of the array) is 7.

- **The slice() method**

The **slice()** method takes two arguments (m,n), and extracts all the elements from the array starting index **m** up to index **n-1**. It will not alter the original array.

Example:

```
array1 = [10,20,30,40,50,60,70,80];
temp = array1.slice(3,6);
```

The slice(3,6) method extracts array1[3],array1[4], and array1[5] and creates an array temp. Thus, temp is an array with elements 40, 50, 60. The original array, array1 in not affected.

- **The splice() method**

The **splice()** method takes two arguments (m.n) and removes **n** elements from the array starting from index **m**. It returns an array which contains the removed elements.

Example:

```
array1 = [10,20,30,40,50,60,70,80];
temp = array1.splice(3,2);
```

The splice(3,2) method removes 2 elements from array1 starting from index 3: array1[3], and array1[4] and returns the removed elements. Thus temp is an array consisting of 40,50. The new contents of the array array1 are 10,20,30,60,70,80.

- **The reverse() method**

The **reverse()** method transposes the elements of an array: the first array element becomes the last and the last becomes the first.

Example:

```
array1 = [10,20,30,40,50,60,70,80];
array1.reverse();
```

array1 now consists of 80,70,60,50,40,30,20,10.

- **The sort() method**

The **sort()** method sorts the elements of the array in ascending order (dictionary order for strings).

- **The indexOf() method**

The **indexOf()** method takes one argument and returns the index of the argument in the array.

Example:

```
array1 = [10,20,30,40,50,60,70,80];
temp = array1.indexOf(40);
```

The **indexOf(40)** returns the index of the element 40 in the array1, which is 3. Thus, the value of the variable temp is 3.

Note: If the specified argument does exist in the array, **indexOf()** returns -1.

---

49. Multidimensional arrays

Multidimensional arrays are created by nesting arrays. That is, an array is contained another array as an element. Using this characteristic of JavaScript arrays, multi-dimensional arrays can be created.

The following code creates a two-dimensional array.

```
<html>
<head><title>Array example</title></head>
<body>
<script>
var array1 = new Array(4);
for (i=0;i<4;i++){
 array1[i] = new Array(4);
 for(j=0;j<4;j++){
   array1[i][j] = i+j;
   }
}
for( i=0;i<4;i++){
  for (j=0;j<4;j++){
   document.write(array1[i][j]," ");}
document.write("<P>");
}
</script>
</body>
</html>
```

Output:

0 1 2 3

1 2 3 4

2 3 4 5

3 4 5 6

Module 7: Multimedia in JavaScripts

50. Including images

To include an image, we use the IMG tag. The IMG tag has several attributes; many of them are optional.

HTML Syntax for IMG tag:

```
<IMG
NAME="Imagename"
SRC="imagefilepath"
LOWSRC="location"
WIDTH="pixels"|"value"%
HEIGHT="pixels"|"value"%
HSPACE="pixels"
VSPACE="pixels"
BORDER="pixels"
ALIGN="left"|"right"|"top"|"absmiddle"|"absbottom"|"texttop"|
"middle"|"baseline"|"bottom"
>
```

Description of attributes:

1. NAME

Programmer assigns a name to the NAME attribute. This name becomes the image name in the program.

Example:

NAME="image1"

2. SRC

This is a very important attribute. You need to assign the SRC attribute the image file name on the disk, including the path to the file if the file is in the same directory of the HTML page.

Example:

SRC="tajmahal.gif"

Here the assumption is that the image file tajmahal.gif is in the same directory as the HTML page.

3. LOWSRC

This is not an important attribute. LOWSRC="location" URL or path to a low resolution image file. The browser loads this first and then loads the image specified in SRC.

4. WIDTH

Assign the WIDTH attribute a value specifying width of the rectangle for the image. You can specify the value in number of pixels or percentage of the window.

Example:

WIDTH=200

Width of the image is specified be 200 pixels.
WIDTH=30%
Width of the image is specified to be 30% of the window.

5 HEIGHT
Assign the HEIGHT attribute a value specifying height of the rectangle for the image. You can specify the value in number of pixels or percentage of the window.
Example:
HEIGHT=200
In this case you have specified height to be 200 pixels.
HEIGHT=30%
In this case you specified height to be 30% of the window.

6. HSPACE
Assign the HSPACE attribute a number specifying the number of pixels of space on either side of the image rectangle.

7. VSPACE
Assign the VSPACE attribute a number specifying the number of pixels of space above and below the image rectangle.

8. BORDER
Assign the BORDER attribute a number specifying the number of pixels of space around the image rectangle.

9. ALIGN
Assign the ALIGN attribute one of the following values, to have the specified alignment of the image. Default is left.
You can specify any one of the values for the ALIGN attribute. The meaning name of the value is self-explanatory.
"left", "right", "top", "absmiddle", "absbottom", "texttop", "middle", "baseline","bottom".

Example:
```
<html>
<head><title>An image example</title><head>
<body>
<h3>Here is a picture of Lord Buddha</h3>
<IMG NAME="Buddha" SRC="buddha.gif "WIDTH=50%" HEIGHT="50%" HSPACE="20"
VSPACE="20" BORDER="20" ALIGN="right" />
</body>
</HTML
```

---

51. Image object

Once you include an image using IMG HTML tag, it becomes a JavaScript object.
The image object is "the location rectangle on the window," not the physical image.

The following HTML code,
<IMG NAME="image1" WIDTH=250 HEIGHT=250>
Is not associated with any physical image. It displays an empty square on the browser window.

The rectangle defined by the HTML code is a JavaScript object. The name of the image object is image1. The object refers to the rectangle on the browser screen. Once an image object is created (the rectangle is defined), you can place any image in the rectangle.

JavaScript syntax for doing this:
document.*imageObject*.**src**="*image file name*"
The right side must include the path for the file if the image file in not in the same directory as the HTML file.

Practice Example 71:

This example first loads an image and provides a JavaScript link. When you click the link, a function is executed, which changes the image.

```
<html>
<head><title>An image example</title>
<script>
function changePicture()
{
 document.Buddha.src="WhiteHouse1.jpg";
}
</script>
<head>
<body>
<h3>Here is a picture of Lord Buddha</h3>
<IMG NAME="Buddha" SRC="buddha.gif "WIDTH=30%" HEIGHT="30%" HSPACE="20"
VSPACE="20" BORDER="20" ALIGN="right" />
<A HREF="JavaScript:changePicture()"> Change Picture </A>
</body>
</html>
```

Try It: http://vulcan.seidenberg.pace.edu/~nmurthy/JavaScriptExamples/Example71.html

---

52. SetInterval() method

The setIntervl() method is used to start the execution of a function after a specified number of milliseconds. The function will be executed repeatedly every specified number of milliseconds.

Syntax:
setInterval("*functionName*()",*n*)

Starts Execution of the specified function *functionName()* after *n* milliseconds. The function will be executed repeatedly every *n* milliseconds.

Practice Example 72:
This code first loads buddha.gif and after 5000 milliseconds, replaces the image by WhiteHouse1.jpg.

```
<html>
<head><title>An image example</title>
<script>
function changePicture()
{
 document.Buddha.src="WhiteHouse1.jpg";
}
</script>
<head>
<body>
<h3>Here is a picture of Lord Buddha</h3>
<IMG NAME="Buddha" SRC="buddha.gif "WIDTH=30%" HEIGHT="30%" HSPACE="20"
VSPACE="20" BORDER="20" ALIGN="right" />
<script>
setInterval("changePicture()",5000);
</script>
</body>
</html>
```

Try It: http://vulcan.seidenberg.pace.edu/~nmurthy/JavaScriptExamples/Example72.html

Note: setInterval() returns a value, which can be used to stop execution of the function *functionName()*.

The clearInterval() method

As described, setInterval() method makes a function execute every specified number of milliseconds. To cancel this the clearInterval() method is used.

The setInterval() method returns a code. This code is need to apply clearInterval() method.

First assign the return code a variable:
timerId = setInterval("*functionName*()",*n*);

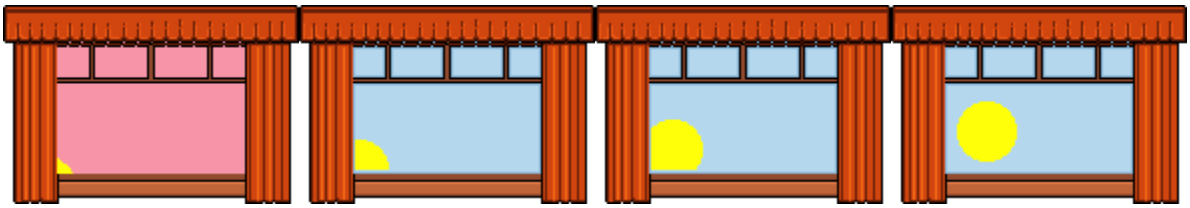
Now, to cancel this, the syntax is:
clearInterval(*timerId*)
Where *timerId* is the value returned by the **setInterval**() call.

---

53. Displaying of a sequence of images

We can display a sequence of images one after the other with a small time delay between displaying images. This delay is created using the **setInterval**() method.

Example:
This example uses 15 images:sunrise1.gif,…,sunrise15.gif. Each is an image of sunrise with slight variations. Here are the first four images:



This program displays these fifteen images one after the other with a slight delay to produce animation of sunrise.

Practice Example 73:

```
<html>
<head>
<title> Repeated images</title>
<script>
function seti()
{
i = 2;
}
function repeatImages()
{
document.picture1.src="../SPACE"+i+".bmp";
i=i+1;
if (i>10) i=1;
}
</script>
</head>
<BODY onLoad="seti();setInterval('repeatImages()',1000)">
<h3> Repeating sequence of images</h3>
<IMG NAME="picture1" SRC="../SPACE1.bmp" WIDTH=200 HEIGHT=200>
</body>
</html>
```

Try It: http://vulcan.seidenberg.pace.edu/~nmurthy/JavaScriptExamples/Example73.html

---

54. The seTimeout() method

The setTimeout() method is used to execute a function after a specified number of milliseconds.

Syntax:
timerId = setTimeout("functionName()",n)

Executes the specified function functionName() after n milliseconds.

setTimeout() returns a value, which can be used to cancel the execution of the function functionName() before the expiration of n milliseconds.

- **The clearTimeout() method**

The clearTimeout() method is used to cancel the seTimeout() call.

Syntax:
clearTimeout(timerId)

Where timerId is the value returned by the seTimeout() call.

Note:
The setTimeout() method does not suspend loading of the page. It just enables the execution of the specified function after the specified number of seconds. The statements following setTimeout() are executed are not affected by this wait.

Practical Example 74

The principle of animation: Select a sequence of images any two of which differ in a minor way. Display the sequence of images one after the other with a small delay of time between them. The following example first loads a sequence of images into an array and displays one of the other with a small delay of time between them.

```
<html>
<head><title> JavaScript animation </title>
<script>
delay = 100
next = 1
images = new Array()
for (i=1; i<13; i++){
images[i] = new Image()
images[i].src = "cats/cat" + i + ".gif"
}
function animate(){
document.animation.src = images[next].src
```

```
next = next +1;
if (next > 12) next = 1
}
function slower(){
delay = delay + 20;
if (delay > 8000) delay = 8000;
}
function faster(){
delay = delay - 20;
if(delay < 0) delay = 100;
}
function stop()
{
clearTimeout(timerId);
}
</script>
</head>
<BODY >
<IMG NAME="animation" SRC="cats/cat1.gif"
onLoad="timerId=setTimeout('animate()',delay)" >
<FORM>
<INPUT TYPE="button" Value="slower" onClick="slower()">
<INPUT TYPE="button" Value="faster" onClick="faster()">
<INPUT TYPE="button" Value="STOP" onClick="stop()">
</FORM>
</body>
</html>
```

Try It: http://vulcan.seidenberg.pace.edu/~nmurthy/JavaScriptExamples/Example74.html

---

Module 8: Selected JavaScript Objects

55. The Math object

The following table gives the properties of Math object. The syntax for using them is **Math**.*propertyname*.

Property name	Value
E	Euler's constant (approximately 2.718)
LN2	ln(2) (approximately 0.693)
LN10	ln(10) (approximately 2.302)
LOG2E	log2(e) (approximately 1.442)
LOG10E	log10(e) (approximately 0.434)
PI	pi (approximately 3.1415926)
SQRT1_2	1/squareroot of 2 (approximately 0.707)
SQRT2	squareroot of 2(approximately 1.414)

Math object methods:
The following is a selected list of methods available in Math object. The syntax for using is **Math**.*methodName*(*arguments*).

Method name	Arguments	Returns
abs(arg1)	arg1 - a number	|arg1|
ceil(arg1)	arg1 - a number	The least integer greater than or equal to arg1
floor(arg1)	arg1 - a number	The greatest integer less than or equal to arg1
max(arg1,arg2)	arg1, arg2 - numbers	The greater of arg1 and arg2
min(arg1,arg2)	arg1, arg2 - numbers	The smaller of the arg1 and arg2
log(arg1)	arg1 - number>0	ln(arg1)
pow(arg1,arg2)	arg1, arg2 - numbers	$arg1^{arg2}$
random()	none	pseudo-random number between 0 and 1
round(arg1)	arg1 - a number	Rounded off integer
sqrt)arg1)	arg1-a number >=0	squarerrot of arg1
sin(arg1)	arg1 - a number (in radians)	sin(arg1)
cos(arg1)	arg1 - a number (in radians)	cos(arg1)
tan(arg1)	arg1 - a number (in radians)	tan(arg1)

---

56. The Date object

The **Date** object is useful when working with dates and times. The Date object has a large number of methods, but no properties.

To start with we have to create our own instance of the object.

There are four ways you can do this, but we will discuss the most useful one:
dateobjectName = **new Date**()

Creates the current date and time.
Once created, many methods (no properties) are available to be used with a Date object.

Date object methods
There are two sets of methods: set and get. Practically, the get methods are more useful.
The get methods:
These methods do not take any arguments:

Name of method	Returns
getDate()	Date - as an integer between 1 and 31
getDay()	Day - aas an integer between 0 and 6; 0 = Sunday
getHours()	Hour - as an integer between 0 and 23
getMinutes()	Minutes - as an integer between 0 and 59
getMonth()	Month - as an integer between 0 and 11; 0 = January
getSeconds()	Seconds - as an integer between 0 and 59
getYear()	Year - a two digit integer

---

57. A digital clock example

This example displays a digital clock, we use the Date object to get the current time and date.

```
<head><title>Digital Clock</title>
<script>
function rundate(){
var dt=new Date()
var mm,dd,yy,h,m,s
mm = dt.getMonth() + 1
dd = dt.getDate()
yy = dt.getFullYear();
h = dt.getHours();
m = dt.getMinutes()
s = dt.getSeconds()
document.dateform.month.value = mm
document.dateform.day.value = dd
document.dateform.year.value = yy
document.dateform.hours.value = h
```

```
document.dateform.minutes.value = m
document.dateform.seconds.value = s
id = setTimeout("rundate()",1000)
}

</script>
</head>
<BODY onLoad="rundate()" onUnload="stop()">
<FORM NAME="dateform">
<TABLE border=2 WIDTH=25% cellpadding=2 cellspacing=2>
<TR align="center">
<TD>Month</TD><TD>Day</TD><TD>Year</TD><TD>Hours</TD>
<TD>Minutes</TD><TD>Seconds</TD>
</TR>
<TR align="center">
<TD><INPUT type=text name=month size=2></TD>
<TD><INPUT type=text name=day size=2></TD>
<TD><INPUT type=text name=year size=2></TD>
<TD><INPUT type=text name=hours size=2></TD>
<TD><INPUT type=text name=minutes size=2></TD>
<TD> <INPUT type=text name=seconds size=2></TD>
</TR>
</TABLE>
</FORM>
</CENTER>
</body>
</html>
```

Try It: http://vulcan.seidenberg.pace.edu/~nmurthy/JavaScriptExamples/Example77.html

---

58. The eval function

The **eval** takes one string argument and returns the value of the argument.
Example:

```
a = "3 +2*5"
document.write(eval(a))
```

Prints 13.

---

59. Another digital clock

```
<head><title>Another digital clock</title><script>
digits = new Array()
for (i=0; i<=9; i++){
digits[i]=new Image()
digits[i].src ="digits/"+ i+"Red.gif"
```

```
}
function runtime(){
var dt=new Date()
h = dt.getHours()
m = dt.getMinutes()
s = dt.getSeconds()
if (h<10){h1 = 0} else {h1 = parseInt(h/10)}
h2 = h%10
document.hour1.src = digits[h1].src
document.hour2.src = digits[h2].src
if (m < 10){ m1 = 0} else { m1 = parseInt(m/10)}
m2 = m%10
document.min1.src = digits[m1].src
document.min2.src = digits[m2].src
if (s < 10){ s1=0} else{ s1 = parseInt(s/10)}
s2 = s%10
document.sec1.src = digits[s1].src
document.sec2.src = digits[s2].src
}
</script> </head><BODY >
<IMAGE NAME="hour1" SRC="digits/0Red.gif">
<IMAGE NAME="hour2" SRC="digits/0Red.gif">
<script> document.write("<b>:</b>")</script>
<IMAGE NAME="min1" SRC="digits/0Red.gif">
<IMAGE NAME="min2" SRC="digits/0Red.gif">
<script> document.write("<b>:</b>") </script>
<IMAGE NAME="sec1" SRC="digits/0Red.gif"> <IMAGE NAME="sec2"
SRC="digits/0Red.gif" onLoad="setTimeout('runtime()',1000)">
</body>
</html>
```

Try It: http://vulcan.seidenberg.pace.edu/~nmurthy/JavaScriptExamples/Example79.html

---

60. Creating new windows

The **open** method of window object is used to create new windows.

Syntax:
variableName = open("URL","windowname", "window features")

Here,
variableName is the name of the window which is used to access the new window's properties;
URL is the URL of the site to get the HTML document.

Windowname is just a name associated with this window for future use with TARGET attribute (in <FORM> and <A> HTML tags).
window features is optional. We will see details in the next section.
open() will create a new window and loads the document from the specified URL.

Example:

```
<html>
<head>
<title> Window object </title>
</head>
<h3> Open cs.pace.edu in a window called firstwin </h3>
<script>
win1 = window.open("http://cs.pace.edu","firstwin")
</script>
</html>
```

---

61. Window features

The following features can be included in open method :

Features	Effect
width=pixels	Defines the width of the new window
height=pixels	Defines the height of the new window
toolbar	New window includes toolbar (row with HOME, RELOAD etc)
location	New window will include location box(the box showing location)
directories	New window will include directory row(the row with what's is new etc)
status	New window will include status bar(the status bar at the bottom)
menubar	New window will contain menubar(row at the top with file, edit, etc)
scrollbars	New window will contain scroll bars
resizable	New window will be resizable by user

---

62. Writing to new window

After you create a new window, use the document object of the new window object.
Example:

```
<html>
<title> A simple window example </title>
<h2> We create a new window and write something on it. </h2>
<script>
mywindow = open("","window1","width=250,height=200")
mywindow.document.write("New York","<P>")
mywindow.document.write("Pace University<BR>")
```

```
mywindow.document.bgColor = "cyan"
</script>
<h3> We are done </h3>
</html>
```

Note:
When writing to a new window, it is important to remember
(1) to place a
 tag at the end of on the last write to window, otherwise the browser will keep looking for more text to write.
(2) to define width and height when including an image on the new window.

---

63. The window property status

The window property status refers to the contents of status bar.
Example:

```
<html> <head>
<title> windows object properties </title>
</head>
<h3> See the default status bar </h3>
<script>
timerid = setTimeout("status='Well Done!!!'",1000)
</script> </html>
```

---

64. The string object

Strings in JavaScript are objects. Whenever you assign a string, enclosed in double quotes or single quotes, to a variable, the variable is a string object.
String is a built-in JavaScript object.

Example:
sample = “New York”
A string object sample is created.
A string literal is also a string object.

Example:
“White Plains" is a string object.
The string object has a large number of properties and methods. We will discuss only two here.
