Introduction to JavaScript

Topics

- Introduction to Java script
- Basics
- Variables
- String manipulation
- Mathematical Functions
- Operations
- Arrays
- Functions
- Objects in Java script
- Regular expressions
- Built- in objects
- Data validation
- Messages & Confirmation
- Status bar
- Writing to a different frame.

What is a Scripting Language

- In computing, a scripting language is a set of commands for controlling some specific piece of hardware, software, or operating system
- Used to write programs that produce inputs to another language processor
- Scripting languages intended to allow end users to extend the functionality of specific software packages are a subset commonly called extension languages simple program structure without complicated declarations.

What is JavaScript?

- Created by Netscape in 1995
  - Originally called LiveWire then LiveScript
- A client-side scripting language
  - Client-side refers to the fact that it is executed in the client (software) that the viewer is using. In the case of JavaScript, the client is the browser.
  - A server-side language is one that runs on the Web server. Examples: PHP, Python
- Interpreted on-the-fly by the client
  - Each line is processed as it loads in the browser

What is JavaScript?

- JavaScript is an object-based, client-side,light weighted scripting language that you can use to make Web pages more dynamic.
- Object Based
- Object based means that JavaScript can use items called objects.
- the objects are not class based (meaning no distinction is made between a class and an instance); instead, they are just general objects.

What is JavaScript?

- Client Side
- Client side means that JavaScript runs in the client (software) that the viewer is using, rather than on the Web server of the site serving the page.
- In this case, the client would be a Web browser.
- Lightweight
- A lightweight programming language is one that is designed to have very small memory footprint, is easy to implement (important when porting a language), and/or has minimalist syntax and features.

What is JavaScript?

- Server-Side Languages
- A server-side language needs to get information from the Web page or the Web browser, send it to a program that is run on the host's server, and then send the information back to the browser.
- Therefore, an intermediate step must send and retrieve information from the server before the results of the program are seen in the browser.

Java vs. JavaScript

- Requires the JDK to create the applet
- Requires a Java virtual machine to run the applet
- Applet files are distinct from the XHTML code
- Source code is hidden from the user
- Programs must be saved as separate files and compiled before they can be run
- Programs run on the server side

- Requires a text editor
- Required a browser that can interpret JavaScript code
- JavaScript can be placed within HTML and XHTML
- Source code is made accessible to the user
- Programs cannot write content to the hard disk
- Programs run on the client side

Quick look

- JavaScript was designed to add interactivity to HTML pages
- JavaScript is a scripting language (a scripting language is a lightweight programming language)
- A JavaScript consists of lines of executable computer code
- A JavaScript is usually embedded directly into HTML pages
- JavaScript is an interpreted language (means that scripts execute without preliminary compilation)
- Everyone can use JavaScript without purchasing a license

When to Use JavaScript

- Data entry validation.
  - If form fields need to be filled out for processing on the server, I let clientside scripts prequalify the data entered by the user.
- Use it to add multimedia elements
  - With JavaScript you can show, hide, change, resize images, and create image rollovers. You can create scrolling text across the status bar.
- Create pages dynamically
  - Based on the user's choices, the date, or other external data, JavaScript can produce pages that are customized to the user.

When to Use JavaScript

- JavaScript can react to events
  - A JavaScript can be set to execute when something happens, like when a page has finished loading or when a user clicks on an HTML element.
- JavaScript can be used to detect the visitor's browser
  - A JavaScript can be used to detect the visitor's browser, and - depending on the browser - load another page specifically designed for that browser
- JavaScript can be used to create cookies
  - A JavaScript can be used to store and retrieve information on the visitor's computer

Pros and Cons of JavaScript

- Advantages:
  - Wide browser support i.e. Most Browsers speak a dialect.
  - Easy access to document objects and their manipulation
  - No long download times w.r.t java or graphic animations
  - No Plug-in support required
  - Relatively Secure

Pros and Cons of JavaScript

- Disadvantages:
  - Not standard support for Javascript across browsers esp. DOM
  - Web page useless if script does not work!!
  - Javascript may be disabled by browser reading HTML file no control over this
  - JavaScripts can run slowly

Writing a JavaScript Program

- The Web browser runs a JavaScript program when the Web page is first loaded, or in response to an event.
- JavaScript programs can either be placed directly into the HTML file or they can be saved in external files.
  - placing a program in an external file allows you to hide the program code from the user
  - source code placed directly in the HTML file can be viewed by anyone

Writing a JavaScript Program

- A JavaScript program can be placed anywhere within the HTML file.
- Many programmers favor placing their programs between <head> tags in order to separate the programming code from the Web page content and layout.
- Some programmers prefer placing programs within the body of the Web page at the location where the program output is generated and displayed.

Writing a JavaScript Program

- That means Javascript can be located in the head, body or external file
- Head section
  - Ensures script is loaded before trigger event
- Body section
  - Script executes when body loads
- External
  - Allows scripts to run on several pages

Writing a JavaScript Program

- JavaScript code is typically embedded in the HTML, to be interpreted and run by the client's browser.
- Here are some tips to remember when writing JavaScript commands.
  - JavaScript code is case sensitive
  - White space between words and tabs are ignored
  - Line breaks are ignored except within a statement
  - JavaScript statements end with a semi- colon ;

Writing a JavaScript Program

- JavaScript Syntax:
- A JavaScript consists of JavaScript statements that are placed within the <script>... </script> HTML tags in a web page.
- You can place the <script> tag containing your JavaScript anywhere within you web page but it is preferred way to keep it within the <head> tags.
- The <script> tag alert the browser program to begin interpreting all the text between these tags as a script.

Writing a JavaScript Program

- simple syntax of your JavaScript will be as follows

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

The script tag takes two important attributes:

- language: This attribute specifies what scripting language you are using. Typically, its value will be javascript.
- Although recent versions of HTML (and XHTML, its successor) have phased out the use of this attribute.

Writing a JavaScript Program

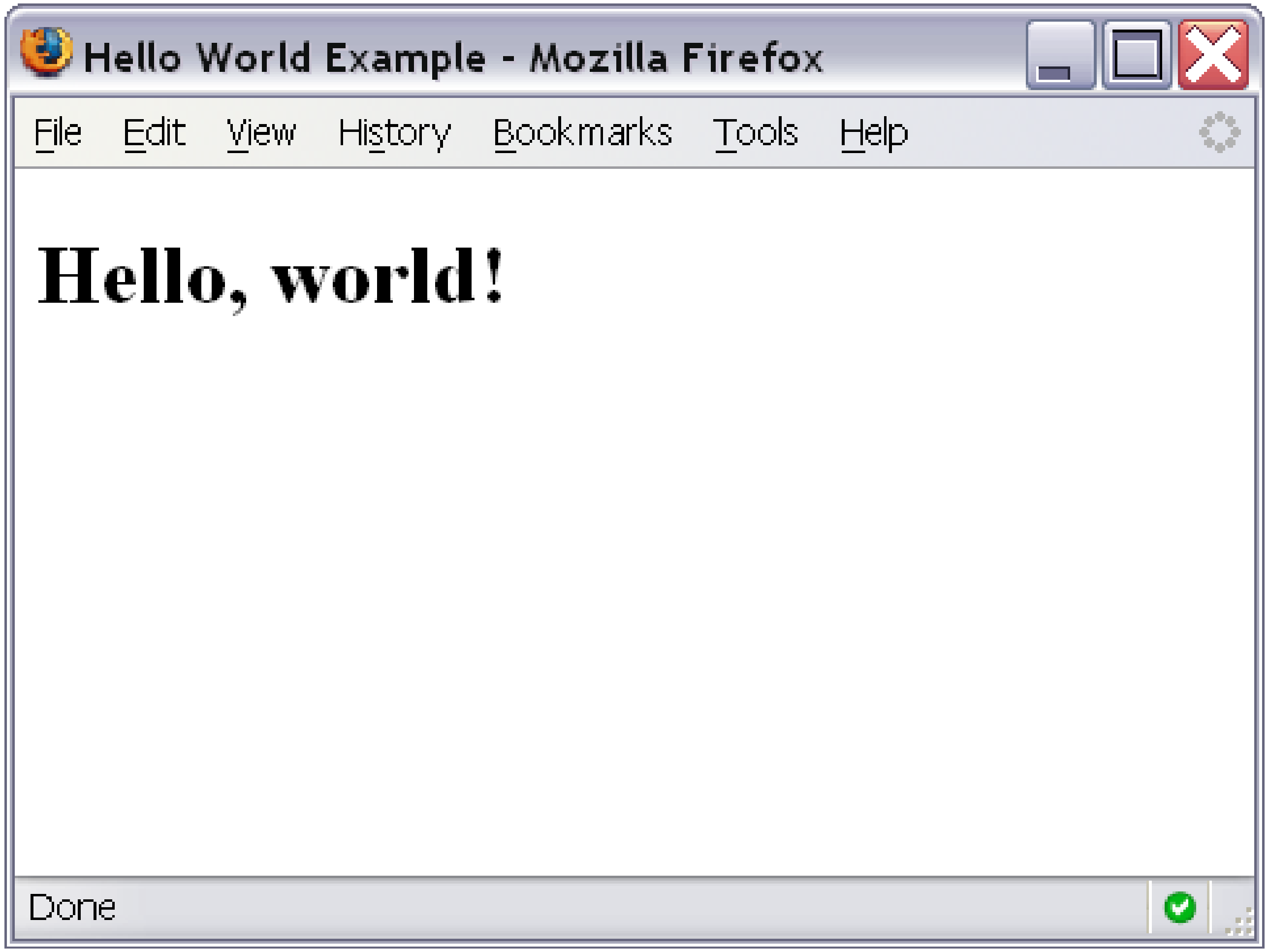
- type: This attribute is what is now recommended to indicate the scripting language in use and its value should be set to "text/javascript".
- So your JavaScript segment will look like:
- <script language="javascript" type="text/javascript">
-                     JavaScript code
-         </script>

Hello, World!

- Typically, in any programming language, the first example you learn displays “Hello, World!”

```
<html>
<body>
<script type="text/javascript">
document.write("<h1>Hello,world!</h1>");
</script>
</body>
</html>
```

Hello World Screenshot


Hello World Example - Mozilla Firefox
File Edit View History Bookmarks Tools Help

Hello, world!

The <script>…</script> tag

- The code for the script is contained in the <script>…</script> tag

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

                  .

                  .

                  .

</script>
```

Hiding JavaScript from Older Browsers

- Some older browsers do not support JavaScript
- We need to tell those browsers to ignore what is in the <script> tag

```
<script type="text/javascript">
  <!--
       some JavaScript code
  //-->
</script>
```

Writing Output to a Web Page

- JavaScript provides two methods to write text to a Web page:
  - **document.write(“text”);**
  - **document.writeln(“text”);**
- The document.writeln() method differs from document.write() in that it attaches a carriage return to the end of each text string sent to the Web page.

```
document.write("<h3>News Flash!</h3><br />");
```

document.write()

Ends in a semicolon

```
document.write("<h1>Hello,world!</h1>");
```

Enclosed in quotes -- denotes a "string"

JavaScript Syntax Issues

- JavaScript commands and names are case-sensitive.
- JavaScript command lines end with a semicolon to separate it from the next command line in the program.
  - in some situations, the semicolon is optional
  - semicolons are useful to make your code easier to follow and interpret

Identifier

- Identifier– The name of a variable (or function)
  - Starts with a letter, can contains digits & underscores
  - Case Sensitive!!
  - Should be meaningful to someone reading your code
- Good: accountBalance, amountDue
- Variables
- A variable represents or holds a value. The actual value of a variable can be changed at anytime.

Variables

- A variable name must start with a letter, underscore (_), or dollar sign ($).
- A variable name cannot start with a number.
- A variable name can only contain alpha-numeric characters (A-z, 0-9) and underscores.
- A variable name cannot contain spaces.
- A variable name cannot be a JavaScript keyword or a JavaScript reserved word.
- Example – Note Assignments
  - `var candyBarPrice = 2.50;`
  - `var taxRate = .075;`
  - `var candyBarsPurchased;`

Variables

- To declare variables, use the keyword var and the variable name:
  - var statement;
  - var a;
- Declaration precedes variable usage in program

  var a;
- a = 5;document.write(a);
- More than one variable can be declared in a statement.
  - var a, b;
- Is the same as…
  - var a;var b;

Using variable in java script

- <html>
- <body>

  <script type="text/javascript">

  var a;

  a = 5;

  document.write(a);

- </script>
- </body>
- </html>

Three data types

- Numbers
- - No units (%, $)
- Strings
- Booleans

1. Number

- JavaScript does not require numbers to be declared as integers, floating-point (decimal) numbers, or any other number type.
- Instead, any number is seen as just another number, whether it is 7, –2, 3.453, or anything else.)

Three data types

- The number will remain the same type unless you perform a calculation to change the type.
- For instance, if you use an integer in a variable, it won't suddenly have decimal places unless you perform a calculation of some sort to change it (dividing unevenly, for instance).

2. String

- String are data type that represent a string of text.
- The string may contain letters,words, spaces, numbers, symbols, or most anything you like.

Three data types

- Strings are defined in a slightly different way than numbers, using this format:
- var variablename="stringtext";
- Must be surrounded by quotes
- Return, backspace, tab – not allowed
- Double quoted strings can contain single quotes and vice versa.
    - eg. ' "We can leave now", she said.'
    - Or "CS 170's assignment"
- Any number of characters allowed

Three data types

3. Boolean

- A *Boolean Data type is one with a value of true or false. Here are examples:*
- var Value=true;
- var Value=false;
- Notice that the words *true and false do not need to be enclosed in quotes.*
- Instead of using the words true and false, JavaScript also allows you to use the number 1for true and the number 0 for false, as shown here:
- var Value=1;var Value=0;

Escape sequences

Output Character	Special Code to Use
Backslash (\)	\\
Double quote (")	\"
Single quote (')	\'
Backspace	\b
Form feed	\f
Newline	\n
Carriage return	\n
Tab	\t
Vertical tab	\v

Comments in JavaScript

- Two types of comments
  - Single line
    - Uses two forward slashes (i.e. ***//***)
  - Multiple line
    - Uses ***/**** and ****/***

Single Line Comment Example

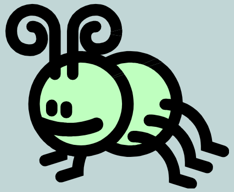
```
<script type="text/javascript">
  <!--
    // This is my JavaScript comment
    document.write("<h1>Hello!</h1>");
  //-->
</script>
```

Multiple Line Comment Example

```
<script type="text/javascript">
 <!--
   /* This is a multiple line comment.
    * The star at the beginning of this line is optional.
    * So is the star at the beginning of this line.
    */
    document.write("<h1>Hello!</h1>");
 //-->
</script>
```

Find the Bug!

```
<script type="text/javascript">
  <!--
    /* This is my JavaScript comment
     * that spans more than 1 line.
     *
    document.write("<h1>Hello!</h1>");
  //-->
</script>
```



Operators

- The Arithmetic Operators:
- There are following arithmatic operators supported by JavaScript language:
- Assume variable A holds 10 and variable B holds 20 then:

Operator	Description	Example
+	Adds two operands	A + B will give 30
-	Subtracts second operand from the first	A - B will give -10
*	Multiply both operands	A * B will give 200
/	Divide numerator by denominator	B / A will give 2

Operators

Operato r	Description	Example
%	Modulus Operator and remainder of after an integer division	B % A will give 0
++	Increment operator, increases integer value by one	A++ will give 11
--	Decrement operator, decreases integer value by one	A-- will give 9

Operators

- The Comparison Operators:
- There are following comparison operators supported by JavaScript language .Assume variable A holds 10 and variable B holds 20 then:

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

Operators

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

Operators

- The Logical Operators:
- There are following logical operators supported by JavaScript language.Assume variable A holds 10 and variable B holds 20 then:

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

Conditional Statements

- Conditional statements are used to perform different actions based on different conditions.
- In JavaScript we have the following conditional statements:
- if statement - use this statement to execute some code.only if a specified condition is true
- Syntax
- if (condition){

    code to be executed if condition is true
- }

Conditional Statements

- <html>
- <body>
- <script type = "text/javascript">
- <!—
- var x=5;
  if (x==5) {
          document.write('x is equal to 5!');
  }
- // -->
- </script>
- </body>
- </html>

Conditional Statements

- If...else Statement
- Use the if....else statement to execute some code if a condition is true and another code if the
- condition is not true.
- Syntax
- if (condition){
  - code to be executed if condition is true
- }
- else{
  - code to be executed if condition is not true
- }

Conditional Statements

- <html><body>
- <script type = "text/javascript">
- <!— var x=5;
- if (x==5) {
  - document.write('x is equal to 5!');
- }
- else {
  - document.write('x is not equal to 5!');
- }// -->
- </script>
- </body></html>

Conditional Statements

- If...else if...else Statement
- Use the if....else if...else statement to select one of several blocks of code to be executed.
- Syntax
- if (condition1){
  - code to be executed if condition1 is true
- }
- else if (condition2){
  - code to be executed if condition2 is true
- }
- else{
- code to be executed if condition1 & condition2 are not true
- }

Conditional Statements

- <html>
- <body>
- <script type = "text/javascript">
- <!—
- var x=5;
- if (x==1) {
- document.write('x is equal to 1!');
- } else if (x==2) {
- document.write('x is equal to 2!');

Conditional Statements

- } else if (x==5) {
- document.write('x is equal to 5!');
- } else {
- document.write('x isn't 1, 2 or 5!');
- }
- // -->
- </script>
- </body>
- </html>

Conditional Statements

- Switch Statement
- Use the switch statement to select one of many blocks of code to be executed.
- Syntax
- switch(n)
- {
- case 1: execute code block 1

            break;

  case 2: execute code block 2

            break;

Conditional Statements

- default: code to be executed if n is different from case 1 and 2
- }
- This is how it works: First we have a single expression n (most often a variable), that is evaluated once.
- The value of the expression is then compared with the values for each case in the structure.
- If there is a match, the block of code associated with that case is executed.
- Use break to prevent the code from running into the next case automatically.

Conditional Statements

- var x=5;
- switch (x) {
- case 1: document.write ('x is equal to 1!'; break;
- case 2: document.write ('x is equal to 2!'; break;
- case 5: document.write ('x is equal to 5!'; break;
- default: document.write ("x isn't 1, 2 or 5!");
- }

Conditional Statements

- The switch is a conditional statement like if statement.
- A switch statement includes literal value or is expression based
- A switch statement includes multiple cases that include code blocks to execute.
- A break keyword is used to stop the execution of case block.
- A switch case can be combined to execute same code block for multiple cases.

ITERATION

- Using Loops
- Execute a block of code a specified number of times while a specified condition is true
- Loops in JavaScript :
- for
- while
- do…while

ITERATION

- for Loops

The for loop is the most compact form of looping and includes the loop initialization, test statement, and iteration statement all in one line.

- The basic syntax is

```
for(initialization; condition; increment)
{…}
```

Eg. `for(i=0;i<5;i=i+1)`

```
{…}
```

ITERATION

- The loop initialization where we initialize our counter to a starting value. The initialization statement is executed before the loop begins.
- The test statement which will test if a given condition is true or not. If the condition is true, then the code given inside the loop will be executed, otherwise the control will come out of the loop.
- The iteration statement where you can increase or decrease your counter.

ITERATION

- While loop
- The purpose of a while loop is to execute a statement or code block repeatedly as long as an expression is true. Once the expression becomes false, the loop terminates.
- while (condition)
- {

  code to be executed
- }

ITERATION

- <body>
```
        <script type="text/javascript">
        var i = 0;
        while (i <= 10){
                document.write("<b>Hello</b>");
                document.write("<br>");
                i = i + 1;
        }
```
- </script>
- </body>

ITERATION

- do...whileLoop
- The do...while loop is similar to the while loop except that the condition check happens at the end of the loop.
- This means that the loop will always be executed at least once, even if the condition is false.

-
  ```
  do  {       ...
  
          ...
  
  }while (condition);
  ```
- Don’t miss the semicolon used at the end of the do...while loop.

ITERATION

```
<body>
        <script type="text/javascript">
                var i = 100;
                do
                {
                  document.write(“Hello");
                  document.write("<br>");
                   i = i + 1;
                } while (i <= 10);
        </script>
   </body>
```

Array

- The Array object is used to store a set of values in a single variable name.”
- An Array is thus a collection of variables of the same data type and (usually) the same size.
- The array element index starts with 0 and the last element index number is one less than the length of the array.
- The individual element of an array is accessed by using the name the array followed by value of array element enclosed in square brackets.

Array

- The syntax for creating an array variable is:

  var variable = new Array(size);
- variable is the name of the array variable
- size is the number of elements in the array (optional)
- To populate the array with values, use: variable[i]=value;

  where i is the ith item of the array. The 1st item has an index value of 0.

Array

- There are 3 ways to construct array in JavaScript

1. By array literal
2. By creating instance of Array directly (using new keyword)
3. By using an Array constructor (using new keyword)

1) JavaScript array literal

The syntax of creating array using array literal is given below:

```
var arrayname=[value1,value2.....valueN];
```

Array

- Defining Arrays
- var team = new Array();
- team[0] = “Red Sox”;
- team[1] = “Mets”;
- team[2] = “Phillies”;

  Or

  ```
  var team = new Array(3);
  team[0] = “Red Sox”;
  team[1] = “Mets”;
  team[2] = “Phillies”;
  ```

- Or var team = new Array(“Red Sox”, “Mets”, “Phillies”);

Array

- <html>
```
        <body>
        <script>
        var emp=["Sonoo","Vimal","Ratan"];
        for (i=0;i<emp.length;i++){
                document.write(emp[i] + "<br/>");
        }
        </script>
```
- </body>
- </html>

Array

- <html>
- <body>
- <script type="text/javascript">

  var myArray = new Array();

  myArray[0] = "Bob";

  myArray[1] = "Pete";

  myArray[2] = "Paul";
- document.write("myArray[0] = " + myArray[0] + "<BR>");
- document.write("myArray[2] = " + myArray[2] + "<BR>");
- document.write("myArray[1] = " + myArray[1] + "<BR>");

  myArray[1] = "Mike";
- document.write("myArray[1] changed to " + myArray[1]);
- </script>
- </body>
- </html>

Array

2) JavaScript Array directly (new keyword)

- The syntax of creating array directly is given below:

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

- <script>

```
var i;
var emp = new Array();
emp[0] = "Arun";
emp[1] = "Varun";
emp[2] = "John";
for (i=0;i<emp.length;i++){
        document.write(emp[i] + "<br>");
}
```

- </script>

Array

3) JavaScript array constructor (new keyword)

- Here, you need to create instance of array by passing arguments in constructor so that we don't have to provide value explicitly.
- The example of creating object by array constructor is given below.
- <script>

```
        var emp=new Array("Jai","Vijay","Smith");
        for (i=0;i<emp.length;i++){
                document.write(emp[i] + "<br>");
        }
```

- </script>

Using delete operator

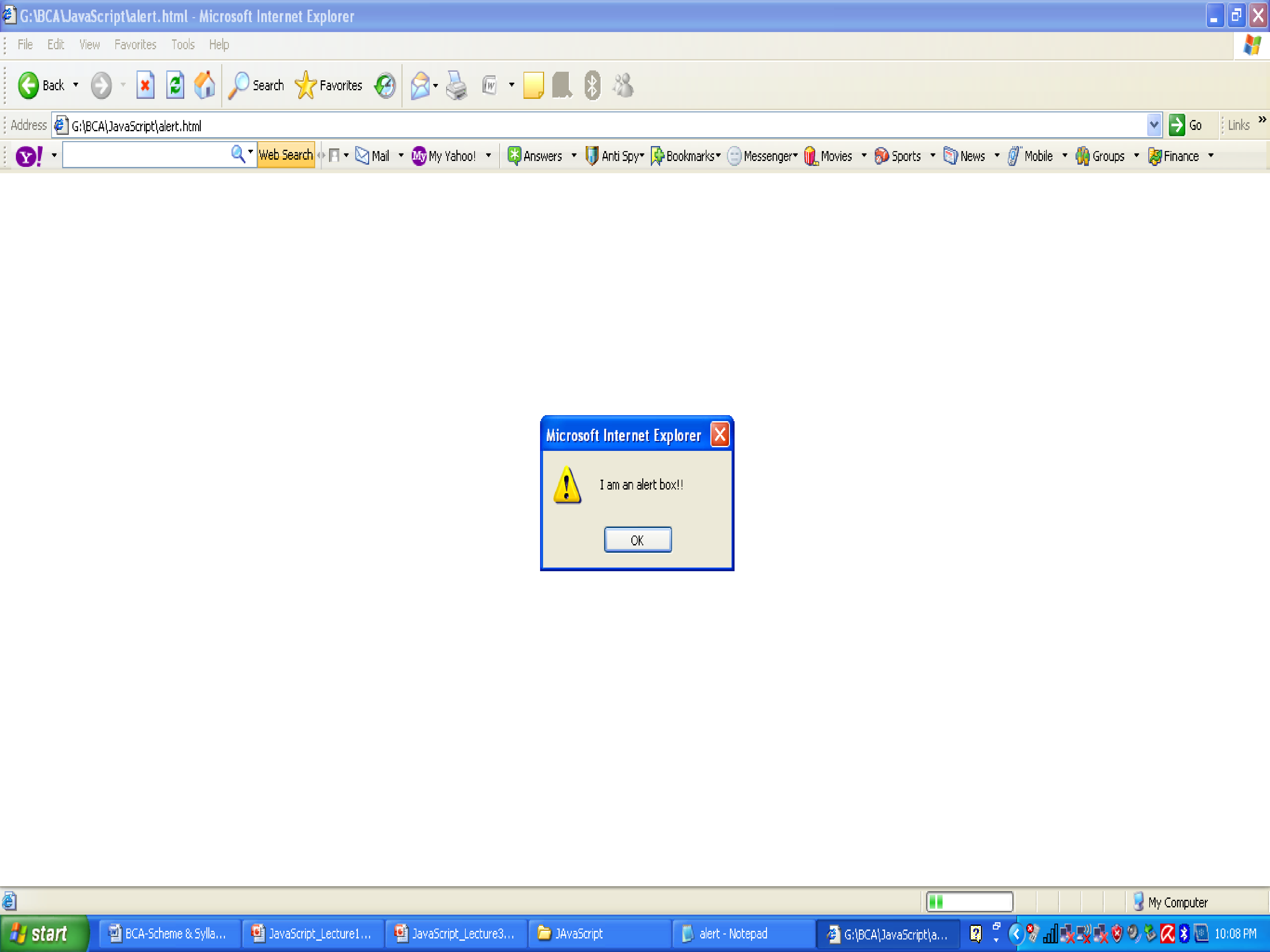
```
<html>
    <head><title>"Array1"</title></head>
    <body>
            <script type="text/javascript">
                        var team = new Array()
                        team[0] = "Red Sox";
                        team[1] = "Mets";
                        team[2] = "Phillies";
                        delete team[0];
                        for (i=0;i<3;i++)
                        {
                                    document.write("<b>  "  + team[i]);
                                    document.write("<br>");
                        }
            </script>
            </body>
</html> output : Undefined Mets Phillies
```

Dialog Boxes or Pop-up Boxes

- It is used to display or input small amount of text to the user.
- Alert Box
  - simplest way to direct small amount of text to a browser window
- Confirm Box
  - Used to verify something
- Prompt Box
  - Get user to input value

Dialog Boxes or Pop-up Boxes

- ALERT BOX
- Display some output to the user
  - alert(“text”)
  - alert(“text1” + “\n”+ “text2”)
  - or alert (“text1 \n text2”)
- <html>
-         <head>
          <script type="text/javascript">
                    alert("I am an alert box!!");
-         </script>
-         </head>
-         <body> </body>
- </html>

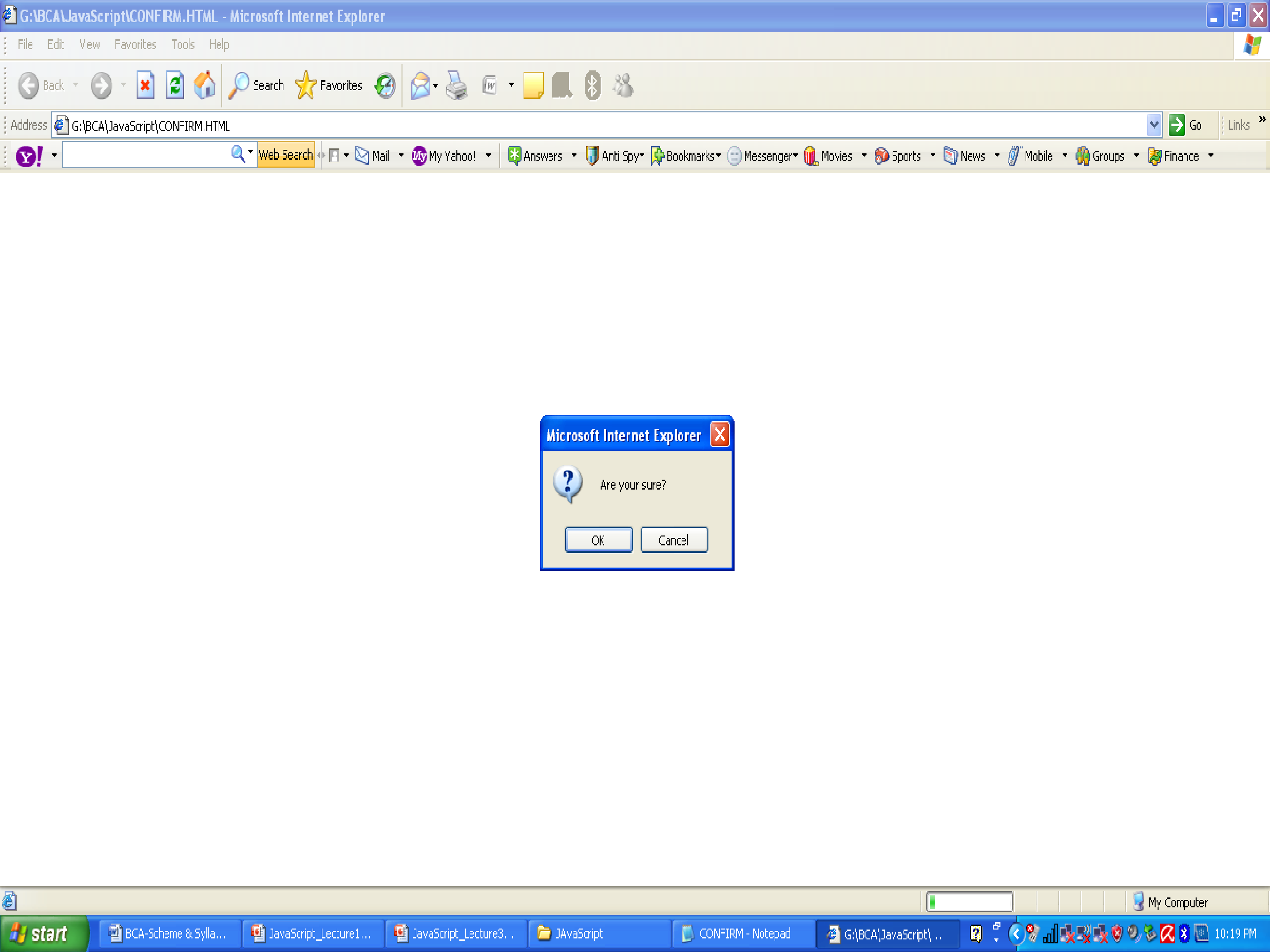


I am an alert box!!

Dialog Boxes or Pop-up Boxes

2. CONFIRM BOX

- Used to verify something
- User clicks OK, Box returns true
- User clicks Cancel, Box returns false
- confirm(“text”)
- <html><head>
-                          <script type="text/javascript">
-   confirm("Are your sure?")
-                          </script>
-          </head>
-          <body>   </body>
- </html>


G:\BCA\JavaScript\CONFIRM.HTML - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address G:\BCA\JavaScript\CONFIRM.HTML
Microsoft Internet Explorer
OK
Cancel
start
10:19 PM

Are your sure?

Dialog Boxes or Pop-up Boxes

3. PROMPT BOX

- Get user to input value
- After inputting value if user clicks on

  OK – the box returns input value

  Cancel – the box returns null
- prompt("text")
- prompt("text", "default value")

Events & Event Handlers

- Every element on a web page has certain events which can trigger invocation of event handlers
- When the page loads, it is called an event. When the user clicks a button, that click too is an event.
- Other examples include events like pressing any key, closing a window, resizing a window, etc.
- Attributes are inserted into HTML tags to define events and event handlers
- Examples of events
  - A mouse click, A web page or an image loading
  - Mousing over a hot spot on the web page, Selecting an input box in an HTML form, Submitting an HTML form
  - A keystroke

Events & Event Handlers

Events

1.	onabort	- Loading of an image is interrupted
2.	onblur	- An element loses focus
3.	onchange	- The content of a field changes
4.	onclick	- Mouse clicks an object
5.	ondblclick	- Mouse double-clicks an object
6.	onerror	- An error occurs when loading a document or an image
7.	onfocus	- An element gets focus
8.	onkeydown	- A keyboard key is pressed

Events & Event Handlers

9.	onkeypress	- A keyboard key is pressed or held down
10.	onkeyup	- A keyboard key is released
11.	onload	- A page or an image is finished loading
12.	onmousedown	- A mouse button is pressed
13.	onmousemove	- The mouse is moved
14.	onmouseout	- The mouse is moved off an element
15.	onmouseover	- The mouse is moved over an element
16.	onmouseup	- A mouse button is released

Events & Event Handlers

17.	onreset	- The reset button is clicked
18.	onresize	- A window or frame is resized
19.	onselect	- Text is selected
20.	onsubmit	- The submit button is clicked
21.	onunload	- The user exits the page

Events & Event Handlers

- onload & onUnload Events
- The onload and onUnload events are triggered when the user enters or leaves the page
- The onload event is often used to check the visitor's browser type and browser version, and load the proper version of the web page based on the information
- Both the onload and onUnload events are also often used to deal with cookies that should be set when a user enters or leaves a page.

Events & Event Handlers

- onFocus, onBlur and onChange
- The onFocus, onBlur and onChange events are often used in combination with validation of form fields.
- Example: The checkEmail() function will be called whenever the user changes the content of the field:
- <input type="text" size="30“ id="email" onchange="checkEmail()">;

Events & Event Handlers

- **Example & Demo: onblur**
- <html>
  <head>
  <script type="text/javascript">
  function upperCase() {
      var x=document.getElementById("fname").value
      document.getElementById("fname").value=x.toUpperCase()
  }
  </script>
- </head>
- <body>
- Enter your name:
- <input type="text" id="fname" onblur="upperCase()">
- </body>
- </html>

Events & Event Handlers

- onSubmit
- The onSubmit event is used to validate all form fields before submitting it.
- Example: The checkForm() function will be called when the user clicks the submit button in the form. If the field values are not accepted, the submit should be canceled. The function checkForm() returns either true or false. If it returns true the form will be submitted, otherwise the submit will be cancelled:
- <form method="post" action=“hello.html“ onsubmit="return checkForm()">

Events & Event Handlers

- onMouseOver and onMouseOut
- onMouseOver and onMouseOut are often used to create "animated" buttons.
- Below is an example of an onMouseOver event. An alert box appears when an onMouseOver event is detected:
- <a href=http://www.w3schools.com onmouseover="alert('An onMouseOver event');return false">
- <img src="w3schools.gif" width="100" height="30"> </a>

Events & Event Handlers

- <html>
- <head>
- <script>
- function myFunction() {
-   alert("Page is loaded");
- }
- </script>
- </head>

- <body onload="myFunction()">

Functions

- A function is a group of reusable code which can be called anywhere in the program.
- This eliminates the need of writing the same code again and again.
- It helps programmers in writing modular codes.
- Functions allow a programmer to divide a big program into a number of small and manageable functions.

Functions

- Syntax
- The basic syntax is shown here.
- function function_name(parameters) {

  JavaScript commands
- }
  - parameters are the values sent to the function (note: not all functions require parameters)
  - { and } are used to mark the beginning and end of the commands in the function.

Functions

- Function names are case-sensitive.
- The function name must begin with a letter or underscore (_) and cannot contain any spaces.
- There is no limit to the number of function parameters that a function may contain.
- The parameters must be placed within parentheses, following the function name, and the parameters must be separated by commas.

Functions

- All variables declared in function are called local
  - Do not exist outside current function
- Parameters
  - Also local variables
- Returning control
  - return statement
  - Can return either nothing, or a value
  - return expression;
  - No return statement same as return;
  - Not returning a value when expected is an error

What are Cookies

- Cookies are data, stored in small text files, on your computer.
- When a web server has sent a web page to a browser, the connection is shut down, and the server forgets everything about the user.
- Cookies were invented to solve the problem "how to remember information about the user":
- When a user visits a web page, his/her name can be stored in a cookie.
- Next time the user visits the page, the cookie "remembers" his/her name.

What are Cookies

- Cookies are a plain text data record of 5 variable-length fields −

1. Expires − The date the cookie will expire. If this is blank, the cookie will expire when the visitor quits the browser.
2. Domain − The domain name of your site.
3. Path − The path to the directory or web page that set the cookie. This may be blank if you want to retrieve the cookie from any directory or page.
4. Secure − If this field contains the word "secure", then the cookie may only be retrieved with a secure server. If this field is blank, no such restriction exists.
5. Name=Value − Cookies are set and retrieved in the form of key-value pairs

Built-In Objects

- Build-In JavaScript Objects

Object	Description
Array	Creates new array objects
Boolean	Creates new Boolean objects
Date	Retrieves and manipulates dates and times
Function	Creates new function objects
Math	Contains methods and properties for performing mathematical calculations
Number	Contains methods and properties for manipulating numbers.
String	Contains methods and properties for manipulating text strings

String Manipulation

- The String object lets you work with a series of characters;
- it wraps Javascript's string primitive data type with a number of helper methods.
- Syntax
- Use the following syntax to create a String object:
    - var val = new String(string);
- The string parameter is a series of characters that has been properly encoded.

String methods

1. charAt()

- charAt() is a method that returns the character from the specified index.
- Characters in a string are indexed from left to right. The index of the first character is 0, and the index of the last character in a string, called stringName, is stringName.length – 1.
- Syntax
- Use the following syntax to find the character at a particular index.
- string.charAt(index)

String methods

- <html>
- <head>
- <title>JavaScript String charAt() Method</title>
- </head>
- <body>

```
<script type="text/javascript">
      var str = new String( "This is string" );
      document.writeln("str.charAt(0) is:" + str.charAt(0));
      document.writeln("<br />str.charAt(1) is:" + str.charAt(1));
      document.writeln("<br />str.charAt(2) is:" + str.charAt(2));
</script>
```

- </body>
- </html>

String methods

2. charCodeAt ()

- This method returns a number indicating the Unicode value of the character at the given index.
- Unicode code points range from 0 to 1,114,111. The first 128 Unicode code points are a direct match of the ASCII character encoding.
- charCodeAt() always returns a value that is less than 65,536.
- Syntax

```
string.charCodeAt(index)
```

- Argument Details
- index: An integer between 0 and 1 less than the length of the string; if unspecified, defaults to 0.

String methods

- <html>
- <head>
- <title>JavaScript String charCodeAt() Method</title>
- </head> <body>
- <script type="text/javascript">

  var str = new String("This is string");

  document.write("str.charCodeAt(0) is:" + str.charCodeAt(0));

  document.write("<br /> str.charCodeAt(1) is:" + str.charCodeAt(1));
- </script>
- </body>
- </html> **Output :** str.charCodeAt(0) is:84 str.charCodeAt(1) is:104

String methods

3. contact ()

- This method adds two or more strings and returns a new single string.
- Syntax
- Its syntax is as follows:
- string.concat(string2, string3[, ..., stringN]);
- Argument Details
- string2...stringN: These are the strings to be concatenated.
- Return Value :Returns a single concatenated string.

String methods

- <body>
- <script type="text/javascript">
- var str1 = new String("This is string one");
- var str2 = new String("This is string two");
- var str3 = str1.concat(str2);
- document.write("Concatenated String :" + str3);
- </script>
- </body>
- Output: Concatenated String :This is string one This is string two

String methods

4) indexOf ()

- This method returns the index within the calling String object of the first occurrence of the specified value, starting the search at fromIndex or -1 if the value is not found.
- Syntax
- Use the following syntax to use the indexOf() method.
- string.indexOf(searchValue[, fromIndex])

String methods

- Argument Details
- searchValue: A string representing the value to search for.
- fromIndex: The location within the calling string to start the search from. It can be any integer between 0 and the length of the string. The default value is 0.
- Return Value
- Returns the index of the found occurrence, otherwise -1 if not found.

String methods

- <body>
- <script type="text/javascript">
- var str1 = new String("This is string one");
- var index = str1.indexOf("string");
- document.write("indexOf found String :" + index);
- document.write("<br />");
- var index = str1.indexOf("one");
- document.write("indexOf found String :" + index);
- </script> Output
- indexOf found String :8 indexOf found String :15

String methods

- lastIndexOf ()
- This method returns the index within the calling String object of the last occurrence of the specified value, starting the search at fromIndex or -1 if the value is not found.
- Syntax
- Its syntax is as follows:
- string.lastIndexOf(searchValue[, fromIndex])

String methods

- Argument Details
- searchValue : A string representing the value to search for.
- fromIndex : The location within the calling string to start the search from. It can be any integer between 0 and the length of the string. The default value is 0.
- Return Value
- Returns the index of the last found occurrence, otherwise -1 if not found.

String methods

- <body>
- <script type="text/javascript">
- var str1 = new String("This is string one and again string");
- var index = str1.lastIndexOf("string");
- document.write("lastIndexOf found String :" + index);
- document.write("<br />");
- var index = str1.lastIndexOf("one");
- document.write("lastIndexOf found String :" + index);
- </script> Output
- lastIndexOf found String :29lastIndexfound String :15

The Math Object & Math Methods

- Another way of performing a calculation is to use the JavaScript built-in Math methods.
- These methods are applied to an object called the Math object.
- The syntax for applying a Math method is:

  value = Math.*method*(*variable*);
- For example,

  AbsValue = Math.abs(NumVar);

The Math Object & Math Methods

Math Method	Description
math.round()	use to round number
math.max()	use to return the number with the highest value of two specified numbers.
math.min()	use to return the number with the lowest value of two specified numbers.
math.ceil()	use to return least integer greater than or equal to argument
math.floor()	use to returns greatest integer less than or equal to argument

Built-In Objects - Number

- The Number object represents numerical date, either integers or floating-point numbers.
- In general, you do not need to worry about Number objects because the browser automatically converts number literals to instances of the number class.
- Syntax
- The syntax for creating a number object is as follows:
- var val = new Number(number);
- In the place of number, if you provide any non-number argument, then the argument cannot be converted into a number, it returns NaN (Not-a-Number).

Built-In Objects - Number

- Number Properties
- Here is a list of each property and their description.
- Property Description
- MAX_VALUE : The largest possible value a number in JavaScript can have 1.7976931348623157E+308
- MIN_VALUE : The smallest possible value a number in JavaScript can have 5E-324

Built-In Objects - Number

- NaN :Equal to a value that is not a number.
- NEGATIVE_INFINITY :A value that is less than MIN_VALUE.
- POSITIVE_INFINITY : A value that is greater than MAX_VALUE

Built-In Objects - Number

- MAX_VALUE
- The Number.MAX_VALUE property belongs to the static Number object.
- It represents constants for the largest possible positive numbers that JavaScript can work with.
- The actual value of this constant is 1.7976931348623157 x 10308.
- Syntax
- The syntax to use MAX_VALUE is:
- var val = Number.MAX_VALUE;

Built-In Objects - Number

- <html>
- <head>
- <script type="text/javascript">
- <!--
- function showValue()
- {
- var maxval = Number.MAX_VALUE;
- document.write ("Value of Number.MAX_VALUE : " + maxval);
- }
- //-->

Built-In Objects - Number

- </script>
- </head>
- <body>
- <p>Click the following to see the result:</p>
- <form>
- <input type="button" value="Click Me" onclick="showValue();" />
- </form>
- </body>
- </html>

Built-In Objects - Number

- MIN_VALUE
- The Number.MIN_VALUE property belongs to the static Number object.
- It represents constants for the smallest possible positive numbers that JavaScript can work with.
- The actual value of this constant is 5 x 10-324.
- Syntax
- The syntax to use MIN_VALUE is:
- var val = Number.MIN_VALUE;

Built-In Objects - Number

- <html>
- <head>
- <script type="text/javascript">
- <!--
- function showValue()
- {
- var val = Number.MIN_VALUE;
- alert("Value of Number.MIN_VALUE : " + val);
- }
- //-->

Built-In Objects - Number

- </script>
- </head>
- <body>
- <p>Click the following to see the result:</p>
- <form>
- <input type="button" value="Click Me" onclick="showValue();" />
- </form>
- </body>
- </html>

Built-In Objects - Number

- NaN
- Unquoted literal constant NaN is a special value representing Not-a-Number.
- Since NaN always compares unequal to any number, including NaN, it is usually used to indicate an error condition for a function that should return a valid number.
- Note: Use the isNaN() global function to see if a value is an NaN value.
- Syntax
- The syntax to use NaN is:
- var val = Number.NaN;

Built-In Objects - Number

- <html><head>
- <script type="text/javascript">
- <!--
- function showValue() {
- var dayOfMonth = 50;
- if (dayOfMonth < 1 || dayOfMonth > 31) {
- dayOfMonth = Number.NaN
- alert("Day of Month must be between 1 and 31.")
- }
- Document.write("Value of dayOfMonth : " + dayOfMonth); }

Built-In Objects - Number

- //-->
- </script>
- </head>
- <body>
- <p>Click the following to see the result:</p>
- <form>
- <input type="button" value="Click Me" onclick="showValue();" />
- </form>
- </body>
- </html>

Built-In Objects - Number

- Number Methods

Method	Description
toExponential()	Forces a number to display in exponential notation, even if the number is in the range in which JavaScript normally uses standard notation.
toFixed()	Formats a number with a specific number of digits to the right of the decimal.
toLocaleString()	Returns a string value version of the current number in a format that may vary according to a browser's local settings.
toPrecision()	Defines how many total digits (including digits to the left and right of the decimal) to display of a number.

Built-In Objects - Number

- Number Methods

Method	Description
toString()	Returns the string representation of the number's value.
valueOf()	Returns the number's value.

Built-In Objects - Boolean

- The Boolean object represents two values, either "true" or "false". If value parameter is omitted or is 0, -0, null, false, NaN, undefined, or the empty string (""), the object has an initial value of false.
- Syntax
- Use the following syntax to create a boolean object.
- var val = new Boolean(value);

Built-In Objects - Boolean

- Boolean Methods

Method	Description
toSource()	Returns a string containing the source of the Boolean object; you can use this string to create an equivalent object.
toString()	Returns a string of either "true" or "false" depending upon the value of the object.
valueOf()	Returns the primitive value of the Boolean object.