Logical Operators

- ❖Used in while and if assertions true/false
- ❖There are three logical operators
  - ◆AND &&
  - ◆OR ||
  - ◆NOT !

A	B	A && B
F	F	F
F	T	F
T	F	F
T	T	T

A	B	A || B
F	F	F
F	T	T
T	F	T
T	T	T

A	!A
F	T
T	F

Note on Precedence: Evaluate relational first and then logical

Logical Operator Examples

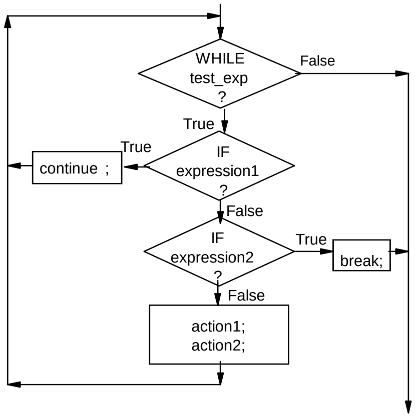
```
if(A==B && A==C)
if(!Valid)
if(A = 0)     // Error use ==
if(!(A || B))
if(!A && !B)
A <= B || C == D
A = B == 0;
if(sQuestion == "C" || sQuestion == "c")
if(sSSN > 999999999 || sSSN < 0)
if(fTax == 0 || fTax ==  15 || fTax == 28)
```

Operators Precedence

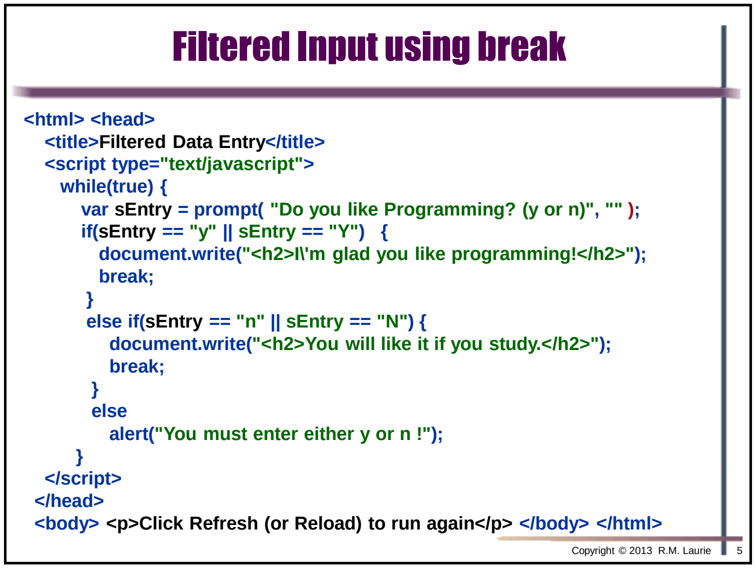
(Highest to Lowest)

.	Property access of an object
( )	Defines order of operation
- ++ --	Minus, Increment, Decrement
!	Logical NOT Operator
* / %	Multiply, Division, Remainder
+ -	Addition, Subtraction
< <= > >=	} Relational Operators
== !=	
&&	Logical AND Operator
| |	Logical OR Operator
= += -= *= /= %=	Compound Assignment

break; continue; Loop Control

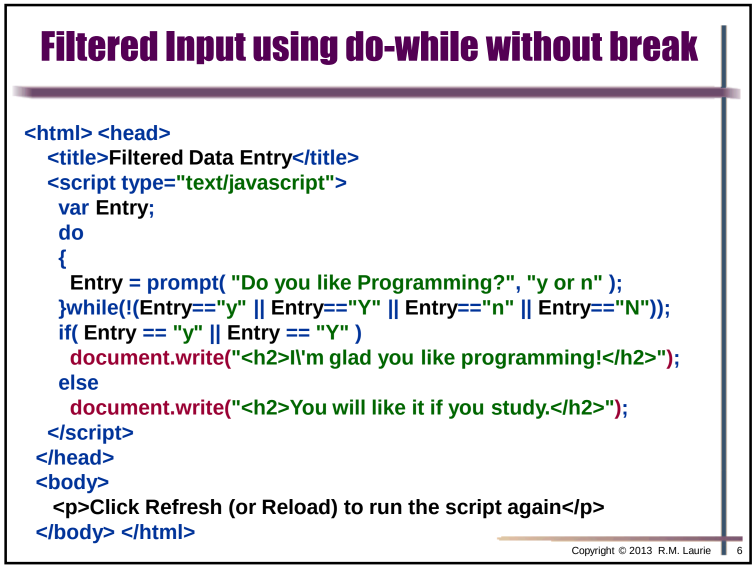


```
while(test_exp)
{
     if(expression1)
        continue;
     if(expression2)
        break;
     action1;
     action2;
}
```



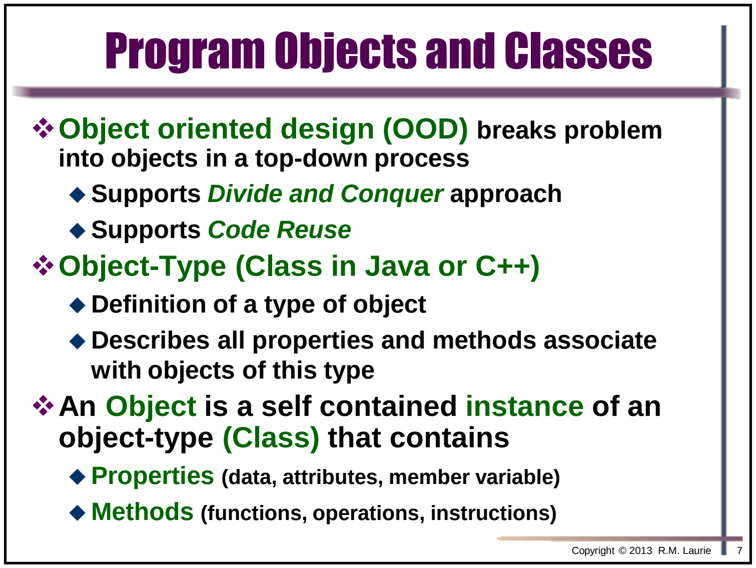
Filtered Input using break

```
<html> <head>
  <title>Filtered Data Entry</title>
  <script type="text/javascript">
    while(true) {
       var sEntry = prompt( "Do you like Programming? (y or n)", "" );
       if(sEntry == "y" || sEntry == "Y")   {
         document.write("<h2>I\'m glad you like programming!</h2>");
         break;
        }
        else if(sEntry == "n" || sEntry == "N") {
          document.write("<h2>You will like it if you study.</h2>");
          break;
        }
        else
          alert("You must enter either y or n !");
      }
  </script>
 </head>
 <body> <p>Click Refresh (or Reload) to run again</p> </body> </html>
```



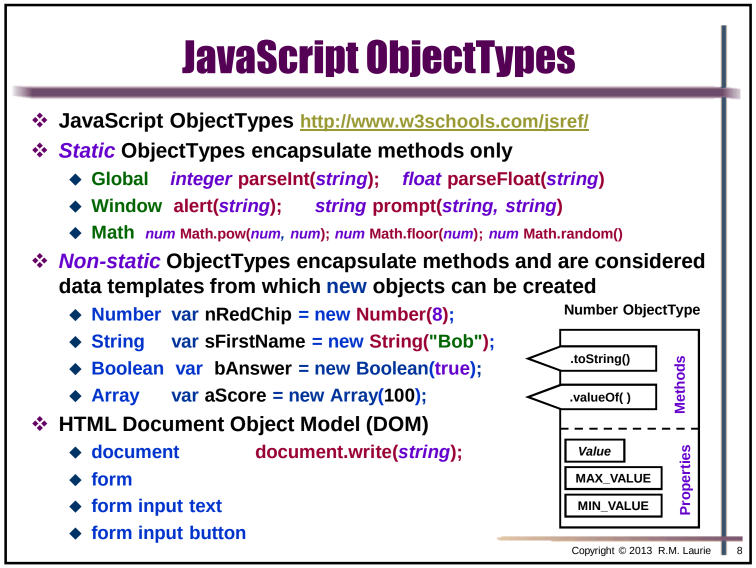
Filtered Input using do-while without break

```
<html> <head>
  <title>Filtered Data Entry</title>
  <script type="text/javascript">
   var Entry;
   do
   {
     Entry = prompt( "Do you like Programming?", "y or n" );
   }while(!(Entry=="y" || Entry=="Y" || Entry=="n" || Entry=="N"));
   if( Entry == "y" || Entry == "Y" )
     document.write("<h2>I\'m glad you like programming!</h2>");
   else
     document.write("<h2>You will like it if you study.</h2>");
  </script>
 </head>
 <body>
   <p>Click Refresh (or Reload) to run the script again</p>
 </body> </html>
```



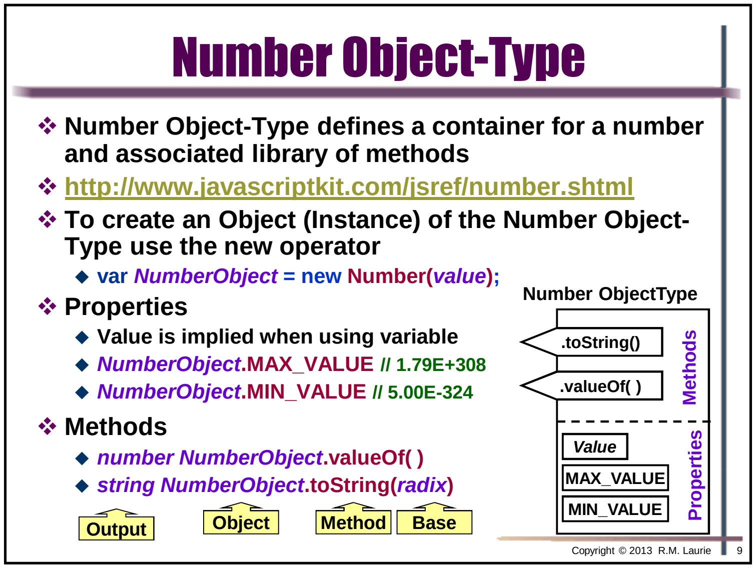
Program Objects and Classes

- Object oriented design (OOD) breaks problem into objects in a top-down process
  - Supports *Divide and Conquer* approach
  - Supports *Code Reuse*
- Object-Type (Class in Java or C++)
  - Definition of a type of object
  - Describes all properties and methods associate with objects of this type
- An Object is a self contained instance of an object-type (Class) that contains
  - Properties (data, attributes, member variable)
  - Methods (functions, operations, instructions)



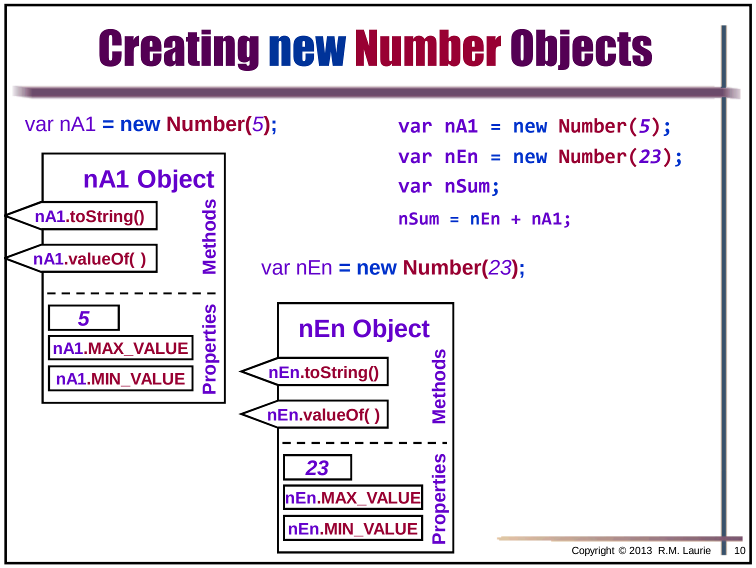
JavaScript ObjectTypes

- JavaScript ObjectTypes http://www.w3schools.com/jsref/
- *Static* ObjectTypes encapsulate methods only
  - Global *integer* parseInt(*string*); *float* parseFloat(*string*)
  - Window alert(*string*); *string* prompt(*string, string*)
  - Math *num* Math.pow(*num, num*); *num* Math.floor(*num*); *num* Math.random()
- *Non-static* ObjectTypes encapsulate methods and are considered data templates from which new objects can be created
  - Number var nRedChip = new Number(8);
  - String var sFirstName = new String("Bob");
  - Boolean var bAnswer = new Boolean(true);
  - Array var aScore = new Array(100);
- HTML Document Object Model (DOM)
  - document document.write(*string*);
  - form
  - form input text
  - form input button



Number Object-Type

- ❖ Number Object-Type defines a container for a number and associated library of methods
- ❖ http://www.javascriptkit.com/jsref/number.shtml
- ❖ To create an Object (Instance) of the Number Object-Type use the new operator
  - ◆ var *NumberObject* = new Number(*value*);
- ❖ Properties
  - ◆ Value is implied when using variable
  - ◆ *NumberObject*.MAX_VALUE // 1.79E+308
  - ◆ *NumberObject*.MIN_VALUE // 5.00E-324
- ❖ Methods
  - ◆ *number NumberObject*.valueOf( )
  - ◆ *string NumberObject*.toString(*radix*)

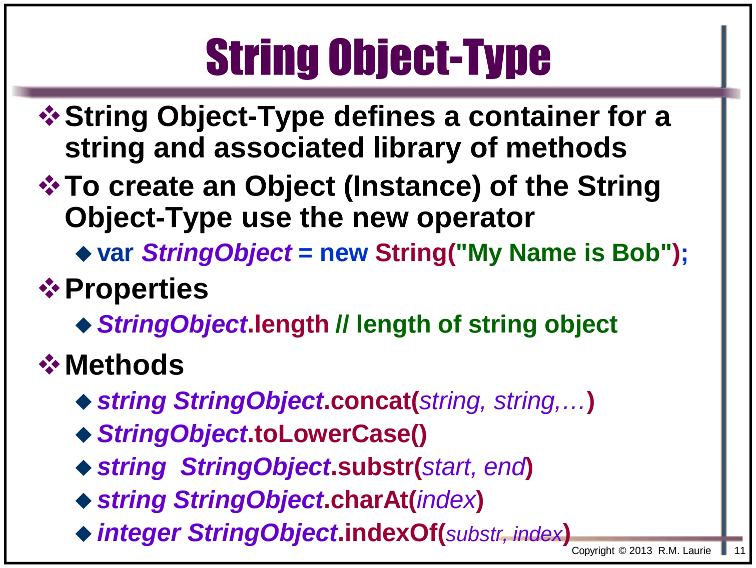


Creating new Number Objects

var nA1 = new Number(*5*);

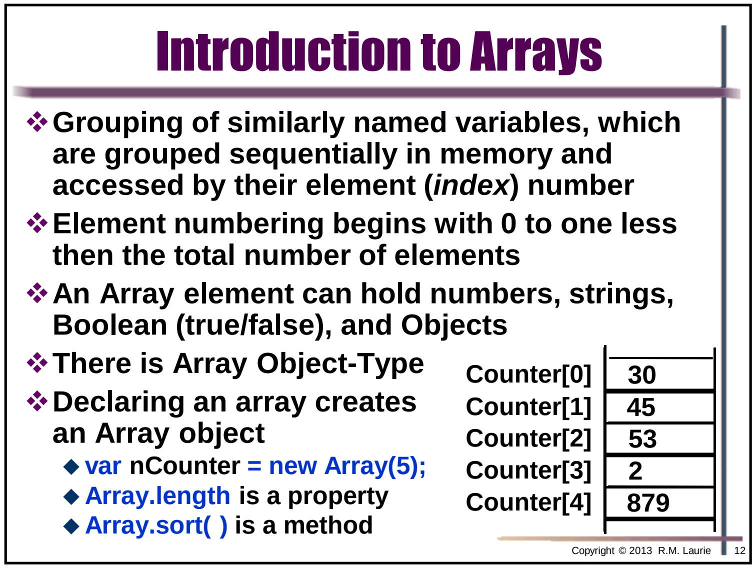
```
var nA1 = new Number(5);
var nEn = new Number(23);
var nSum;
nSum = nEn + nA1;
```

var nEn = new Number(*23*);



String Object-Type

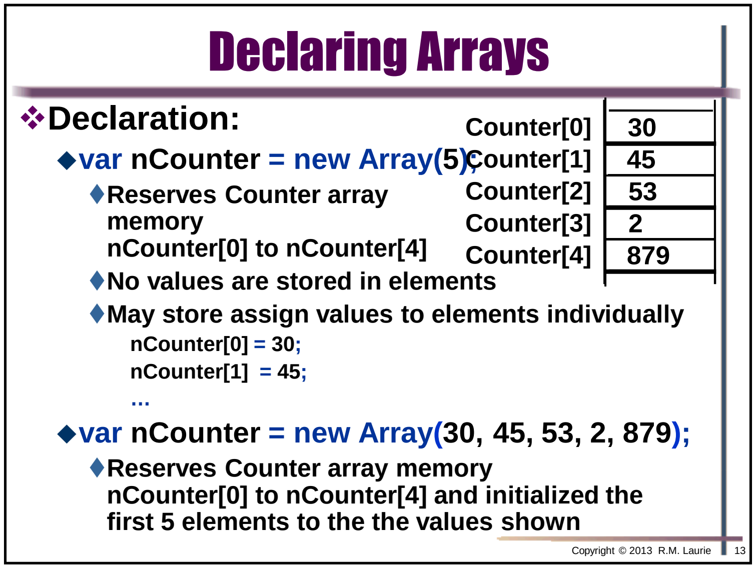
- ❖ String Object-Type defines a container for a string and associated library of methods
- ❖ To create an Object (Instance) of the String Object-Type use the new operator
  - ◆ var *StringObject* = new String("My Name is Bob");
- ❖ Properties
  - ◆ *StringObject*.length // length of string object
- ❖ Methods
  - ◆ *string StringObject*.concat(*string, string,…*)
  - ◆ *StringObject*.toLowerCase()
  - ◆ *string StringObject*.substr(*start, end*)
  - ◆ *string StringObject*.charAt(*index*)
  - ◆ *integer StringObject*.indexOf(*substr, index*)



Introduction to Arrays

- ❖ Grouping of similarly named variables, which are grouped sequentially in memory and accessed by their element (*index*) number
- ❖ Element numbering begins with 0 to one less then the total number of elements
- ❖ An Array element can hold numbers, strings, Boolean (true/false), and Objects
- ❖ There is Array Object-Type
- ❖ Declaring an array creates an Array object
  - ◆ var nCounter = new Array(5);
  - ◆ Array.length is a property
  - ◆ Array.sort( ) is a method

Counter[0]	30
Counter[1]	45
Counter[2]	53
Counter[3]	2
Counter[4]	879



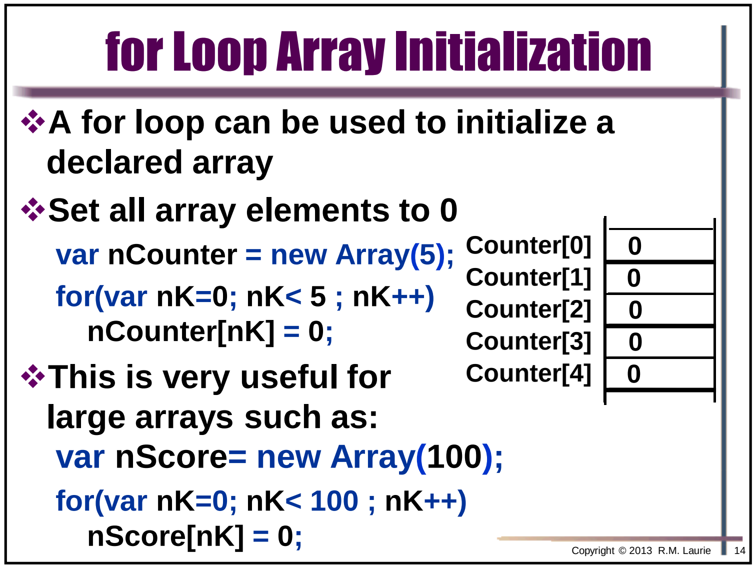
Declaring Arrays

- Declaration:
  - var nCounter = new Array(5);
    - Reserves Counter array memory nCounter[0] to nCounter[4]
    - No values are stored in elements
    - May store assign values to elements individually

```
nCounter[0] = 30;
nCounter[1]  = 45;
...
```

Counter[0]	30
Counter[1]	45
Counter[2]	53
Counter[3]	2
Counter[4]	879

- var nCounter = new Array(30, 45, 53, 2, 879);
  - Reserves Counter array memory nCounter[0] to nCounter[4] and initialized the first 5 elements to the the values shown



for Loop Array Initialization

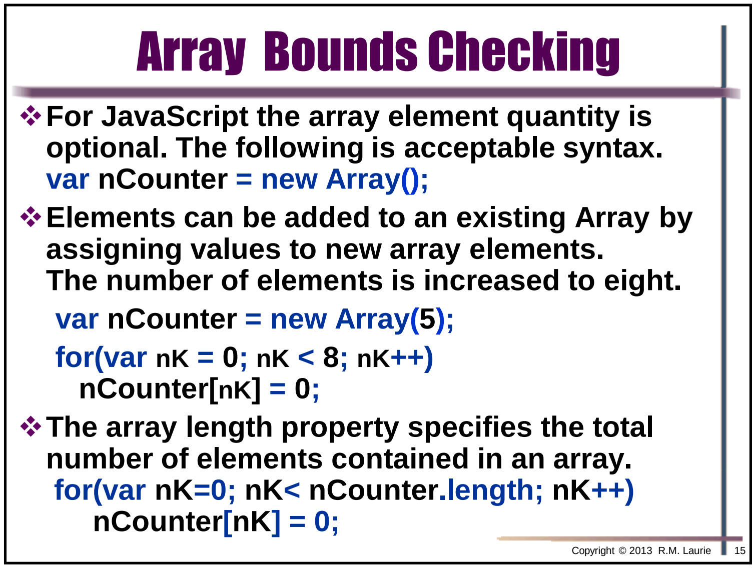
- A for loop can be used to initialize a declared array
- Set all array elements to 0

```
var nCounter = new Array(5);
for(var nK=0; nK< 5 ; nK++)
   nCounter[nK] = 0;
```

Counter[0]	0
Counter[1]	0
Counter[2]	0
Counter[3]	0
Counter[4]	0

- This is very useful for large arrays such as:

```
var nScore= new Array(100);
for(var nK=0; nK< 100 ; nK++)
   nScore[nK] = 0;
```



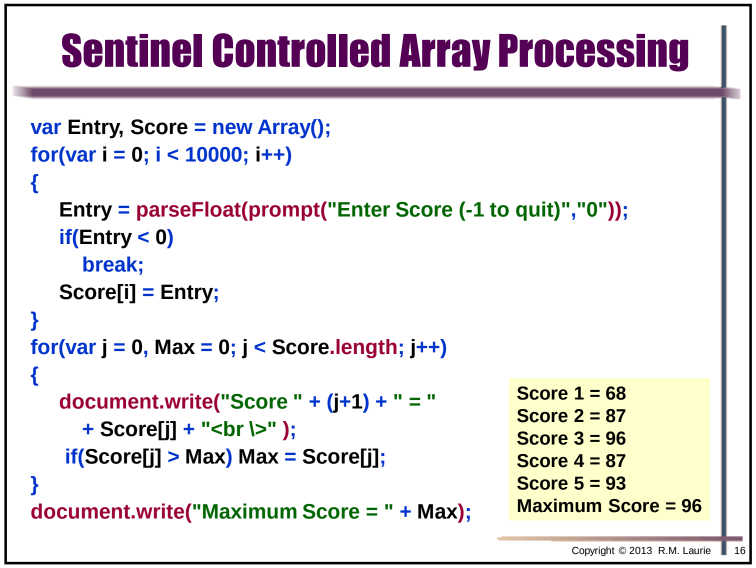
Array Bounds Checking

- For JavaScript the array element quantity is optional. The following is acceptable syntax.
  var nCounter = new Array();
- Elements can be added to an existing Array by assigning values to new array elements. The number of elements is increased to eight.

```
var nCounter = new Array(5);
for(var nK = 0; nK < 8; nK++)
  nCounter[nK] = 0;
```

- The array length property specifies the total number of elements contained in an array.

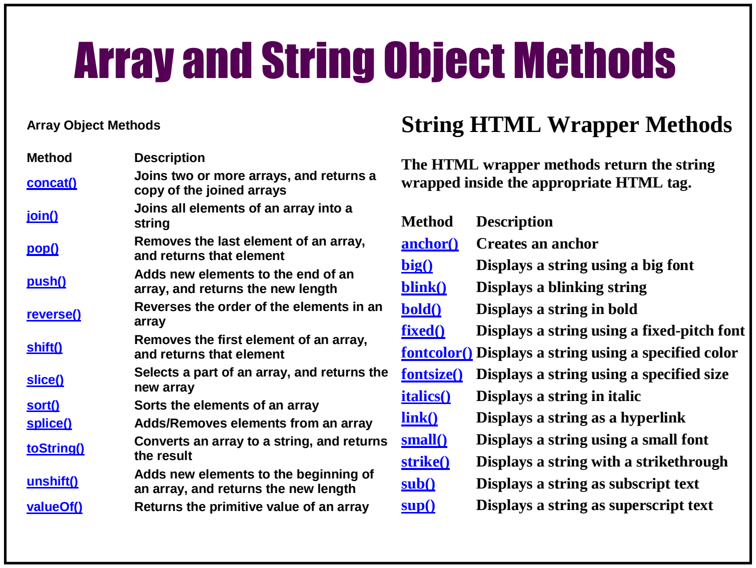
```
for(var nK=0; nK< nCounter.length; nK++)
    nCounter[nK] = 0;
```



Sentinel Controlled Array Processing

```
var Entry, Score = new Array();
for(var i = 0; i < 10000; i++)
{
    Entry = parseFloat(prompt("Enter Score (-1 to quit)","0"));
    if(Entry < 0)
       break;
    Score[i] = Entry;
}
for(var j = 0, Max = 0; j < Score.length; j++)
{
    document.write("Score " + (j+1) + " = "
       + Score[j] + "<br \>" );
     if(Score[j] > Max) Max = Score[j];
}
document.write("Maximum Score = " + Max);
```

```
Score 1 = 68
Score 2 = 87
Score 3 = 96
Score 4 = 87
Score 5 = 93
Maximum Score = 96
```



Array and String Object Methods

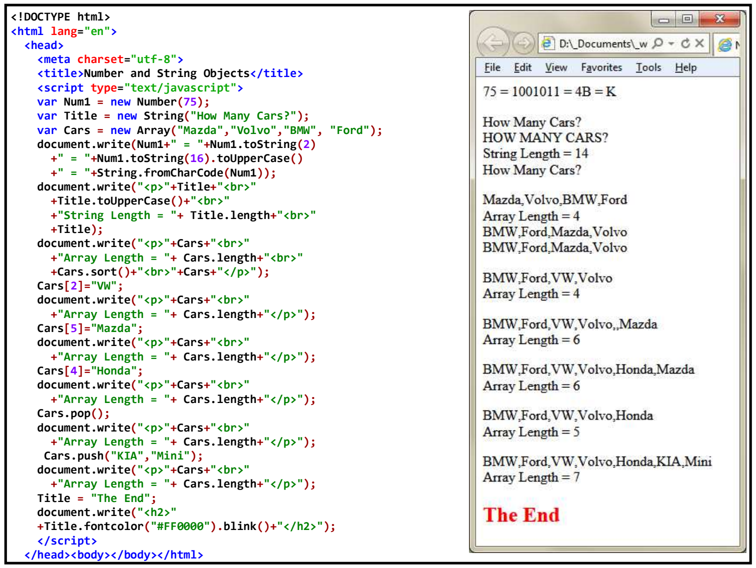
Array Object Methods

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

String HTML Wrapper Methods

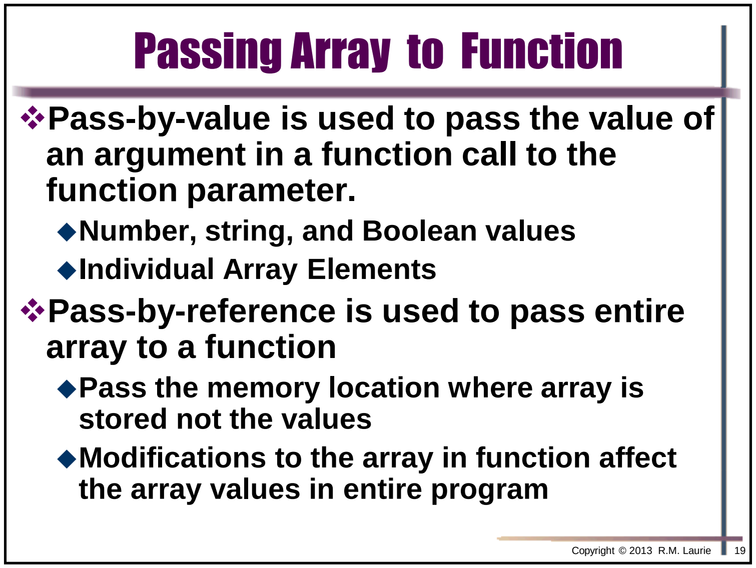
The HTML wrapper methods return the string wrapped inside the appropriate HTML tag.

Method	Description
anchor()	Creates an anchor
big()	Displays a string using a big font
blink()	Displays a blinking string
bold()	Displays a string in bold
fixed()	Displays a string using a fixed-pitch font
fontcolor()	Displays a string using a specified color
fontsize()	Displays a string using a specified size
italics()	Displays a string in italic
link()	Displays a string as a hyperlink
small()	Displays a string using a small font
strike()	Displays a string with a strikethrough
sub()	Displays a string as subscript text
sup()	Displays a string as superscript text



```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Number and String Objects</title>
    <script type="text/javascript">
    var Num1 = new Number(75);
    var Title = new String("How Many Cars?");
    var Cars = new Array("Mazda","Volvo","BMW", "Ford");
    document.write(Num1+" = "+Num1.toString(2)
      +" = "+Num1.toString(16).toUpperCase()
      +" = "+String.fromCharCode(Num1));
    document.write("<p>"+Title+"<br>"
      +Title.toUpperCase()+"<br>"
      +"String Length = "+ Title.length+"<br>"
      +Title);
    document.write("<p>"+Cars+"<br>"
      +"Array Length = "+ Cars.length+"<br>"
      +Cars.sort()+"<br>"+Cars+"</p>");
    Cars[2]="VW";
    document.write("<p>"+Cars+"<br>"
      +"Array Length = "+ Cars.length+"</p>");
    Cars[5]="Mazda";
    document.write("<p>"+Cars+"<br>"
      +"Array Length = "+ Cars.length+"</p>");
    Cars[4]="Honda";
    document.write("<p>"+Cars+"<br>"
      +"Array Length = "+ Cars.length+"</p>");
    Cars.pop();
    document.write("<p>"+Cars+"<br>"
      +"Array Length = "+ Cars.length+"</p>");
     Cars.push("KIA","Mini");
    document.write("<p>"+Cars+"<br>"
      +"Array Length = "+ Cars.length+"</p>");
    Title = "The End";
    document.write("<h2>"
    +Title.fontcolor("#FF0000").blink()+"</h2>");
    </script>
  </head><body></body></html>
```

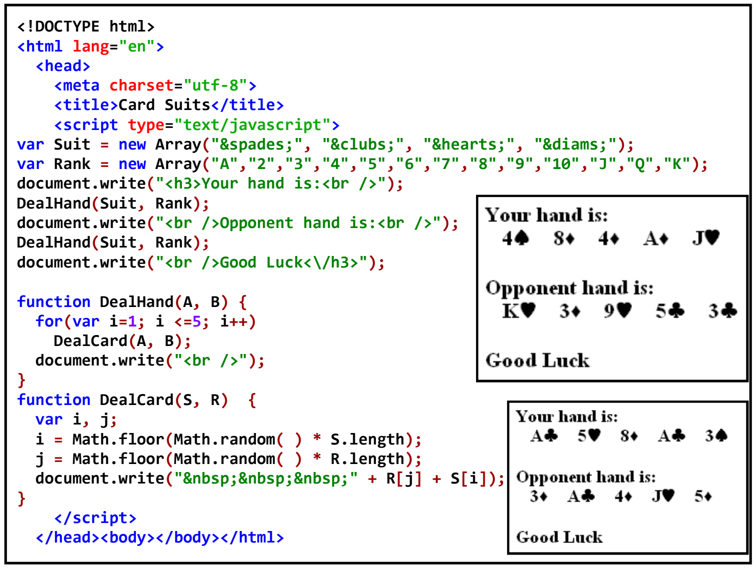
```
75 = 1001011 = 4B = K

How Many Cars?
HOW MANY CARS?
String Length = 14
How Many Cars?

Mazda,Volvo,BMW,Ford
Array Length = 4
BMW,Ford,Mazda,Volvo
BMW,Ford,Mazda,Volvo

BMW,Ford,VW,Volvo
Array Length = 4

BMW,Ford,VW,Volvo,,Mazda
Array Length = 6

BMW,Ford,VW,Volvo,Honda,Mazda
Array Length = 6

BMW,Ford,VW,Volvo,Honda
Array Length = 5

BMW,Ford,VW,Volvo,Honda,KIA,Mini
Array Length = 7

The End
```



Passing Array to Function

- Pass-by-value is used to pass the value of an argument in a function call to the function parameter.
  - Number, string, and Boolean values
  - Individual Array Elements
- Pass-by-reference is used to pass entire array to a function
  - Pass the memory location where array is stored not the values
  - Modifications to the array in function affect the array values in entire program

Copyright © 2013 R.M. Laurie 19



```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Card Suits</title>
    <script type="text/javascript">
var Suit = new Array("&spades;", "&clubs;", "&hearts;", "&diams;");
var Rank = new Array("A","2","3","4","5","6","7","8","9","10","J","Q","K");
document.write("<h3>Your hand is:<br />");
DealHand(Suit, Rank);
document.write("<br />Opponent hand is:<br />");
DealHand(Suit, Rank);
document.write("<br />Good Luck<\/h3>");

function DealHand(A, B) {
  for(var i=1; i <=5; i++)
    DealCard(A, B);
  document.write("<br />");
}
function DealCard(S, R)  {
  var i, j;
  i = Math.floor(Math.random( ) * S.length);
  j = Math.floor(Math.random( ) * R.length);
  document.write("&nbsp;&nbsp;&nbsp;" + R[j] + S[i]);
}
    </script>
  </head><body></body></html>
```

```
Your hand is:
   4♠   8♦   4♦   A♦   J♥

Opponent hand is:
   K♥   3♦   9♥   5♣   3♣

Good Luck
```

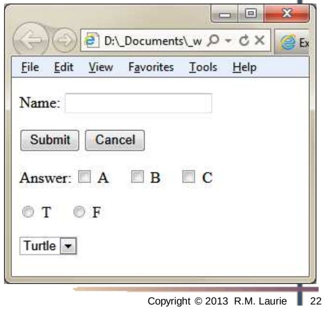
```
Your hand is:
   A♣   5♥   8♦   A♣   3♠

Opponent hand is:
   3♦   A♣   4♦   J♥   5♦

Good Luck
```

Event Driven Programming

- Procedural Program Paradigm
  - *Command line programming* is DOS style programming
  - Sequential processing modeled using flowcharts
  - Programs may include:
    - Sequential, selection, and repetition structures
    - Functions calls to user defined or library procedures
    - Arrays
- Event Driven Program Paradigm
  - Microsoft Windows and Mac OSX are operating system environments that designed around event driven concepts
  - Program execution is determined by user actions or Events (onclick, onkeyup, onchange) on a Graphical User Interface
  - Functions can read and write to DOM Document Object Model
  - Program divided into three sections:
    - Graphical User Interface = GUI created using HTML forms
    - Events triggered by user interacting with GUI
    - Event handling calls JavaScript functions

HTML Forms and JavaScript Processing

- HTML Forms can be utilized to implement a (GUI) Graphical User Interface that interacts with JavaScript
  - Form element event triggers call to JavaScript function
  - JavaScript functions can read input data from form elements
  - JavaScript functions can write output data to form elements
  - Formatting of form elements can be done using CSS styles
- Common form elements available in HTML
  - Text Field
  - Buttons
  - Check boxes
  - Radio buttons
  - Select Menus
  - Text Area



Name:

Submit Cancel

Answer: □ A □ B □ C

○ T ○ F

Turtle ▾

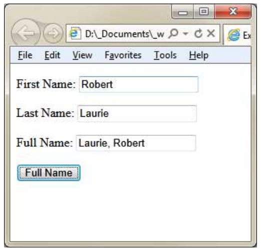
Form and Input Elements

- Form is a block level element
  `<form name="frmName" action="#"></form>`
  - name attribute is identifier of the form for older browsers
  - id attribute is identifier of the form for newer browsers & DOM
  - action specifies the Server script on web server to process the sent data; for JavaScript "#" works well
  - <u>Don't forget </u>to close your form elements
- Text input element is for single line text input
  `<input type="text" name="txtFirstName" tabindex="1">`
  - type="text" defines as a text box
  - name attribute is identifier of the form for older browsers
  - id attribute is identifier of the form for newer browsers & DOM
  - size attribute specifies character width of element
  - maxlength attribute specifies maximum number of characters entered
  - tabindex="1" is the first tab stop. Set to -1 to disallow tab
  - readonly="readonly" For results only not input
- Input button usually used to call function

```
<input type="button" name="btCalc" value="Calculate"
onclick="calculate()">
```

Old DOM Access method utilizes document element name attribute for access of element

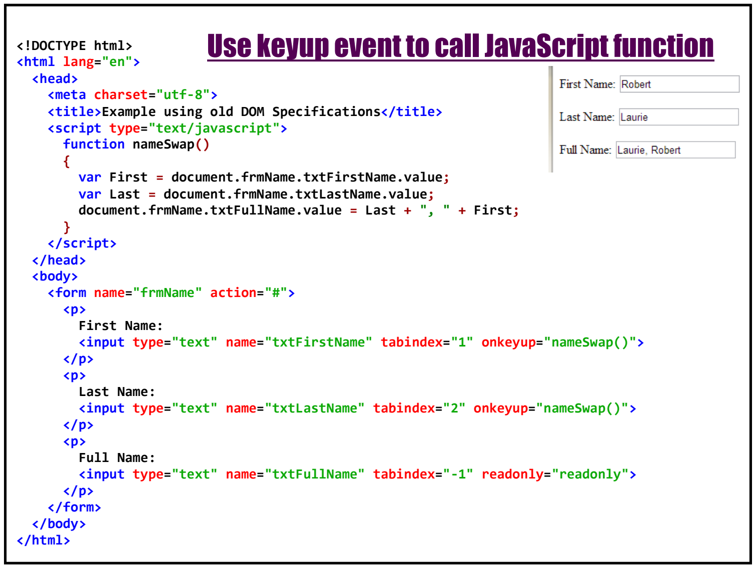
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Example using old DOM Specifications</title>
    <script type="text/javascript">
      function NameSwap()
      {
        var First = document.frmName.txtFirstName.value;
        var Last = document.frmName.txtLastName.value;
        document.frmName.txtFullName.value = Last + ", " + First;
      }
    </script>
  </head>
  <body>
    <form name="frmName" action="#">
      <p>
        First Name:
        <input type="text" name="txtFirstName" tabindex="1">
      </p>
      <p>
        Last Name:
        <input type="text" name="txtLastName" tabindex="2">
      </p>
      <p>
        Full Name:
        <input type="text" name="txtFullName" tabindex="-1" readonly="readonly">
      </p>
      <p>
        <input type="button" name="btnFullName" tabindex="3"
         value="Full Name" onclick="NameSwap();">
      </p>
    </form>
  </body>
</html>
```



First Name: Robert

Last Name: Laurie

Full Name: Laurie, Robert



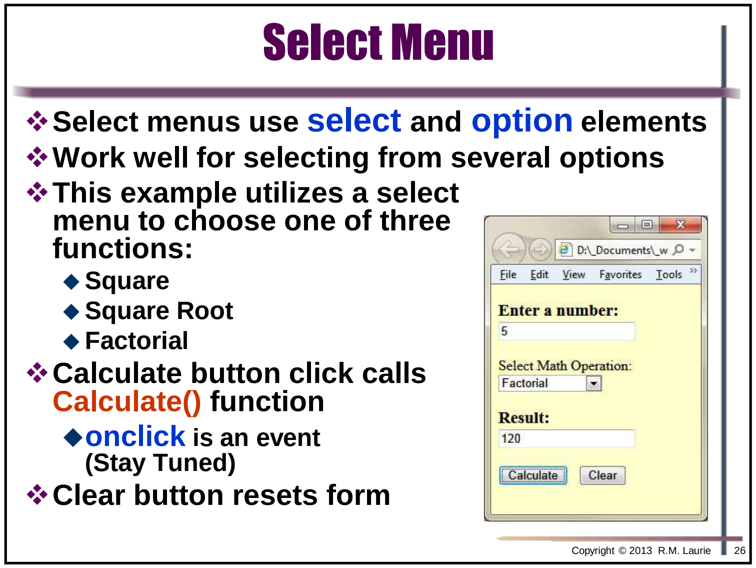
Use keyup event to call JavaScript function

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Example using old DOM Specifications</title>
    <script type="text/javascript">
      function nameSwap()
      {
        var First = document.frmName.txtFirstName.value;
        var Last = document.frmName.txtLastName.value;
        document.frmName.txtFullName.value = Last + ", " + First;
      }
    </script>
  </head>
  <body>
    <form name="frmName" action="#">
      <p>
        First Name:
        <input type="text" name="txtFirstName" tabindex="1" onkeyup="nameSwap()">
      </p>
      <p>
        Last Name:
        <input type="text" name="txtLastName" tabindex="2" onkeyup="nameSwap()">
      </p>
      <p>
        Full Name:
        <input type="text" name="txtFullName" tabindex="-1" readonly="readonly">
      </p>
    </form>
  </body>
</html>
```

First Name: Robert

Last Name: Laurie

Full Name: Laurie, Robert



Select Menu

- Select menus use select and option elements
- Work well for selecting from several options
- This example utilizes a select menu to choose one of three functions:
  - Square
  - Square Root
  - Factorial
- Calculate button click calls Calculate() function
  - onclick is an event (Stay Tuned)
- Clear button resets form

Enter a number:

5

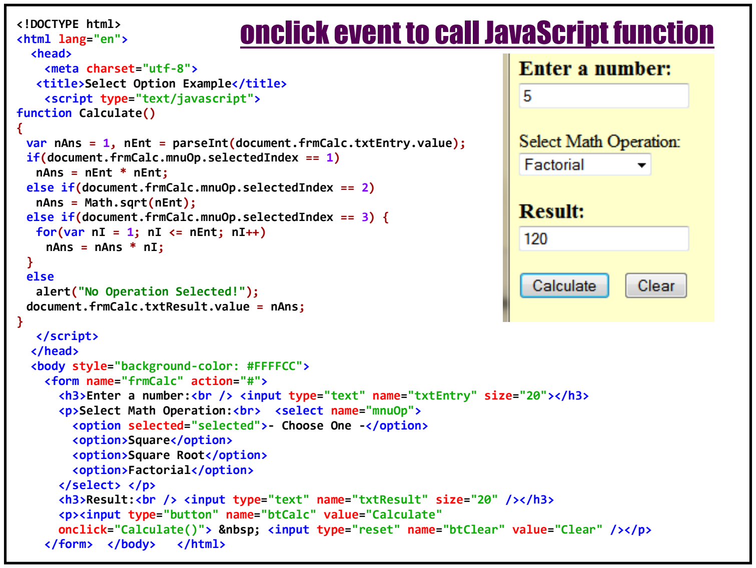
Select Math Operation:

Factorial

Result:

120

Copyright © 2013 R.M. Laurie 26



onclick event to call JavaScript function

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
   <title>Select Option Example</title>
    <script type="text/javascript">
function Calculate()
{
 var nAns = 1, nEnt = parseInt(document.frmCalc.txtEntry.value);
 if(document.frmCalc.mnuOp.selectedIndex == 1)
  nAns = nEnt * nEnt;
 else if(document.frmCalc.mnuOp.selectedIndex == 2)
  nAns = Math.sqrt(nEnt);
 else if(document.frmCalc.mnuOp.selectedIndex == 3) {
  for(var nI = 1; nI <= nEnt; nI++)
   nAns = nAns * nI;
 }
 else
  alert("No Operation Selected!");
 document.frmCalc.txtResult.value = nAns;
}
   </script>
  </head>
  <body style="background-color: #FFFFCC">
    <form name="frmCalc" action="#">
      <h3>Enter a number:<br /> <input type="text" name="txtEntry" size="20"></h3>
      <p>Select Math Operation:<br>  <select name="mnuOp">
        <option selected="selected">- Choose One -</option>
        <option>Square</option>
        <option>Square Root</option>
        <option>Factorial</option>
      </select> </p>
      <h3>Result:<br /> <input type="text" name="txtResult" size="20" /></h3>
      <p><input type="button" name="btCalc" value="Calculate"
      onclick="Calculate()"> &nbsp; <input type="reset" name="btClear" value="Clear" /></p>
    </form>  </body>   </html>
```

Enter a number:

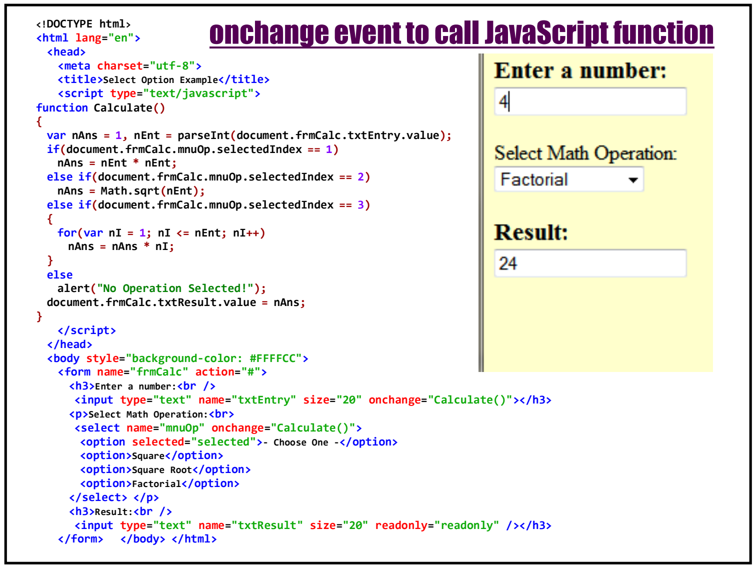
5

Select Math Operation:

Factorial

Result:

120



onchange event to call JavaScript function

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Select Option Example</title>
    <script type="text/javascript">
function Calculate()
{
  var nAns = 1, nEnt = parseInt(document.frmCalc.txtEntry.value);
  if(document.frmCalc.mnuOp.selectedIndex == 1)
    nAns = nEnt * nEnt;
  else if(document.frmCalc.mnuOp.selectedIndex == 2)
    nAns = Math.sqrt(nEnt);
  else if(document.frmCalc.mnuOp.selectedIndex == 3)
  {
    for(var nI = 1; nI <= nEnt; nI++)
      nAns = nAns * nI;
  }
  else
    alert("No Operation Selected!");
  document.frmCalc.txtResult.value = nAns;
}
    </script>
  </head>
  <body style="background-color: #FFFFCC">
    <form name="frmCalc" action="#">
      <h3>Enter a number:<br />
       <input type="text" name="txtEntry" size="20" onchange="Calculate()"></h3>
      <p>Select Math Operation:<br>
       <select name="mnuOp" onchange="Calculate()">
        <option selected="selected">- Choose One -</option>
        <option>Square</option>
        <option>Square Root</option>
        <option>Factorial</option>
      </select> </p>
      <h3>Result:<br />
       <input type="text" name="txtResult" size="20" readonly="readonly" /></h3>
    </form>   </body> </html>
```

Enter a number:

4

Select Math Operation:

Factorial

Result:

24