INTRODUCTION TO JAVASCRIPT PART TWO

INTRODUCTION TO JAVASCRIPT

- String Manipulation
- Math Object
- Date Object
- Functions
- Arrays

STRING MANIPULATION

- Examples of strings are as follows:

```
var string1 = "blue";
var string2= " Today is Monday";
var string3 = "12";
```

STRING MANIPULATION

- String Manipulation allows us to:
  - Combine these strings into a sentence i.e. take these strings and concatenate them into one.
  - Break a string into smaller ones.
  - Convert a string into upper case or lowercase.
  - See if a particular character exists in a string.
  - Find the length of a string.
  - Convert a string into a number.

STRING MANIPULATION

- In addition to the concatenation operator (+) JavaScript supports several advanced string operations as well.
- These functions are accessed by referring to various methods of the String object.
- Moreover, this object also contains the 'length' property.

EXAMPLE

```
name = "Bhola";
document.write(" The length of the string ‘name’ is ", name.length ) ;
```

```
The length of the string ‘name’ is 5
```

STRING METHODS

FORMAT: *string.methodName( )*

EXAMPLE:

```
name = "Bhola";
document.write(name);
document.write(name.toUpperCase()) ;
```

```
BholaBHOLA
```

STRING METHODS: ALL OTHERS

toLowerCase()

toUpperCase()

charAt(*n*)

substring(*n*, *m*)

indexOf(*substring*, *n*)

lastIndexOf(*substring*, *n*)

split(*delimiter*)

TOLOWERCASE(), TOUPPERCASE()

```
person = "Bhola" ;
document.write(person) ;
document.write(person.toLowerCase());
document.write(person.toUpperCase());
```

```
BholabholaBHOLA
```

CHARAT(*N*)

Returns a string containing the character at position *n* (the position of the 1st character is 0).

```
mister = "Bhola" ;
document.write( mister.charAt(0));
document.write( mister.charAt(2));
```

```
Bo
```

SUBSTRING(*N*, *M*)

Returns a string containing characters copied from positions *n* to *m* – *1*.

```
s = "Bhola" ;
document.write(s.substring(1, 3));
document.write(s.substring(0, s.length));
```

```
hoBhola
```

INDEXOF(*SEARCHSTRING, N*)

Returns the position of the first occurrence of *searchstring*. The search begins at character 0 unless specified by a value of N.

-1 is returned if the *searchstring* is **<u>not</u>** found.

```
s = "Bhola" ;
document.write(s.indexOf("ola"));
document.write(s.indexOf("z"));
```

```
2-1
```

SPLIT(*DELIMITER*)

Returns an array of strings, created by splitting string into substrings, at *delimiter* boundaries.

```
s = "Hello: I must be going!" ;
data = new Array() ;
data = s.split("  ") ;
document.write("<TABLE>") ;
for( I in data) {
      document.write("<TR><TD>", data[ i ], "</TD></TR>") ;
}
document.write("</TABLE>") ;
```

```
Hello:
I
must
be
going!
```

AUTOMATIC CONVERSION TO STRINGS

- Whenever a non-string is used where JavaScript is expecting a string, it converts that non-string into a string.
- Example:
  - The document.write() method expects a string (or several strings, separated by commas) as its argument.
  - When a number or a Boolean is passed as an argument to this method, JavaScript automatically converts it into a string before writing it onto the document.

THE ‘+’ OPERATOR

- When ‘+’ is used with numeric operands, it adds them.
- When it is used with string operands, it concatenates them.
- When one operand is a string, and the other is not, the non-string will first be converted to a string and then the two strings will be concatenated.

THE '+' OPERATOR: EXAMPLES

```
document.write(2 + 3) ;
```

```
5
```

```
document.write("2" + "3") ;
```

```
23
```

```
document.write("2" + 3) ;
```

```
23
```

STRINGS IN MATHEMATICAL EXPRESSIONS

When a string is used in a mathematical context, if appropriate, JavaScript first converts it into a number. Otherwise, a "NaN" is the result.

```
document.write("2" * 3) ;
```

```
6
```

```
document.write("2" + 3) ;
```

```
23
```

THE 'TOSTRING' METHOD
EXPLICIT CONVERSION TO A STRING

EXAMPLE:

Convert 100.553478 into a currency format

```
a = 100.553478 ;
b = a.toString() ;
decimalPos = b.indexOf(".", 0) ;
c = b.substring(0, decimalPos + 3) ;
document.write(c) ;
```

```
100.55
```

INTRODUCTION TO JAVASCRIPT

- String Manipulation
- Math Object
- Date Object
- Functions
- Arrays

JAVASCRIPT MATH OBJECT

- In addition to the simple arithmetic operations (e.g. +, *, etc.) JavaScript supports several advanced mathematical operations as well.

- These functions are accessed by referring to various methods of the **Math** object.

- Moreover, this object also contains several useful mathematical constants as its properties. For example Math.PI.

METHODS

sin(r)
cos(r)
tan(r)
asin(x)
acos(x)
atan(x)
atan2(x, y)

sqrt(x)
pow(x, y)

exp(x)
log(x)

round(x)
floor(x)
ceil(x)

abs(x)

max(x, y)
min(x, y)

random()

sqrt(x)	pow(x, y)
Returns the square root of x	Returns x raised to the power y
Math.sqrt(9) → 3	Math.pow(2, 3) → 8

round(x)	floor(x)	ceil(x)
Returns integer nearest to x	Returns largest integer that is less than or equal to x	Returns smallest integer that is greater than or equal to x
$1.1 \rightarrow 1$ $12.5 \rightarrow 13$ $12.9 \rightarrow 13$	$1.1 \rightarrow 1$ $12.5 \rightarrow 12$ $12.9 \rightarrow 12$	$1.1 \rightarrow 2$ $12.5 \rightarrow 13$ $12.9 \rightarrow 13$

abs(x)

Returns the absolute value of x

$1.1 \rightarrow 1.1$

$-12.5 \rightarrow 12.5$

$0 \rightarrow 0$

min(x, y)	max(x, y)
Returns the smaller of x and y	Returns the larger of x and y
2, 4 $\rightarrow$ 2 -12, -5 $\rightarrow$ -12	2, 4 $\rightarrow$ 4 -12, -5 $\rightarrow$ -5

random()

Returns a randomly-selected, floating-point number between 0 and 1

Math.random( ) → 0.9601111965589273

RANDOM( ): EXAMPLE

Write JavaScript code that will display the result of the rolling of a 6-sided dice on user command.

If you want to get a random number between 1 and another number, just multiply the random() method by the uppermost number and add 1 to the total.

For Example: to generate a random number from 1 to 6:

var mynumber =Math.floor(Math.random()* 6 + 1);

INTRODUCTION TO JAVASCRIPT

- String Manipulation
- Math Object
- Date Object
- Functions
- Arrays

DATE OBJECT

Date()

Constructs an empty date object.

For example: var now=new Date();

getDate()	getDay()
Returns the day of the month.	Returns an integer representing the day of the week, Sunday is 0 and Saturday is 6.
`var dayNum = now.getDate();`	`var day = now.getDay();`

getMonth()	getFullYear()
Returns the month field of the Date object, represented by an integer, January is 0 and December is 11.	Returns the year as a four digit number.
`var month = now.getMonth();`	`var thisyear = now.getFullYear();`

DATE: OTHER RETRIEVAL METHODS

getHours()

getMinutes()

getSeconds()

getMilliseconds()

getTime()

INTRODUCTION TO JAVASCRIPT

- String Manipulation
- Math Object
- Date Object
- Functions
- Arrays

FUNCTIONS

- Functions:
  - consist of one or more *statements* (i.e., lines of program code that perform some operation).
  - are separated in some way from the rest of the program, for example, by being enclosed in curly brackets, {.....}
  - are given a unique name, so that they can be *called* from elsewhere in the program.
- Functions are used where the same operation has to be performed many times within a program.

FUNCTIONS

In JavaScript, functions are created in the following way:

```
function name()
{
    statement;
    statement;
}
```

FUNCTIONS

However, it is often necessary to supply information to a function so that it can carry out its task.

```
function addVAT(price)

{
   price *= 1.21;
   alert(price);
}
```

We would call this function in the following way:

```
  addVAT(costPrice);
```

FUNCTIONS

Sometimes we also need to get some information back from a function.

```
function addVAT(price)

{
   price *= 1.21;
   return(price);
}
```

We would call this function in the following way:

```
  var newPrice = addVAT(costPrice);
```

INTRODUCTION TO JAVASCRIPT

- String Manipulation
- Math Object
- Date Object
- Functions
- $\boxed{\text{Arrays}}$

ARRAYS

- The Array object is used to store a set of values in a single variable name.

```
var data= new Array();
data[0] = "Hurling" ;
data[1] = "Rugby";
data[2] = "Football";
data[3] = "Soccer";
data[4] = "Tennis";

document.write(data);
```

ARRAY MANIPULATION

```
for (count=0; count<len; count++) {
  document.write(data[count] + "<br>");
}
```

```
for (x in data) {
 document.write(data[x] + "<br>");
}
```

OBJECT BASED ARRAY FUNCTIONS

- Arrays have lots of nifty built in functions such as join(), push(), pop(), sort(), slice(), splice(), and more.

join()

The **join()** method is used to put all the elements of an array into a string. The elements will be separated by a specified separator.

```
data.join(',  ');
data.join(‘<br> ');
```

push()	unshift()
The **push()** method adds one or more elements to the end of an array and returns the new length.	The **unshift()** method adds one or more elements to the start of an array and returns the new length.
data.push('golf');	data.unshift('golf');

pop()	shift()
The **pop()** method is used to remove and return the last element of an array.	The **shift()** method is used to remove and return the first element of an array.
data.pop();	data.shift();

splice()-delete

The **splice()** command must specify where it should begin deleting (index number of first item to be deleted) and how many items it should delete.

```
data.splice(2,2);
```

splice()-add

The splice() command must specify where the new items should be located, 0 to indicate that you do not want to delete any items, then the list of items to be inserted: one or more values separated by commas.

```
data.splice(2,0, "Cricket", "Snooker");
```

splice()-replace

The process is the same as adding an item, but instead of specifying 0 for the second piece of information, you supply the number of items to be replaced. This is followed by the list of items that are replacing the deleted (replaced) items.

```
data.splice(2,2, "Cricket", "Snooker");
```

reverse()

The **reverse()** method is used to reverse the order of the elements in an array.

```
data.reverse();
```

concat()

The **concat()** method is used to join two or more arrays.

```
data.concat(data);
```

sort()	sort() - numeric
The sort() method is used to sort the elements of an array.	To sort numbers, you must add a function that compare numbers.
data.sort();	data.sort(function(a,b){return a - b});