Java Script Tutorial

JavaScript is a programming languagecommonly used in web development. It was originally developed by Netscape as a means to add dynamic and interactive elements to websites. While JavaScript is influenced by Java, the syntax is more similar to C and is based on ECMAScript, a scripting language developed by Sun Microsystems.

JavaScript is a client-side scripting language, which means the source code is processed by the client's web browser rather than on the web server. This means JavaScript functionscan run after a webpage has loaded without communicating with the server. For example, a JavaScript function may check a web form before it is submitted to make sure all the required fields have been filled out. The JavaScript code can produce an error message before any information is actually transmitted to the server.

```
<script>
  function sum(a,b)
  {
    return a + b;
  }
  var total = sum(7,11);
  alert(total);
</script>
```

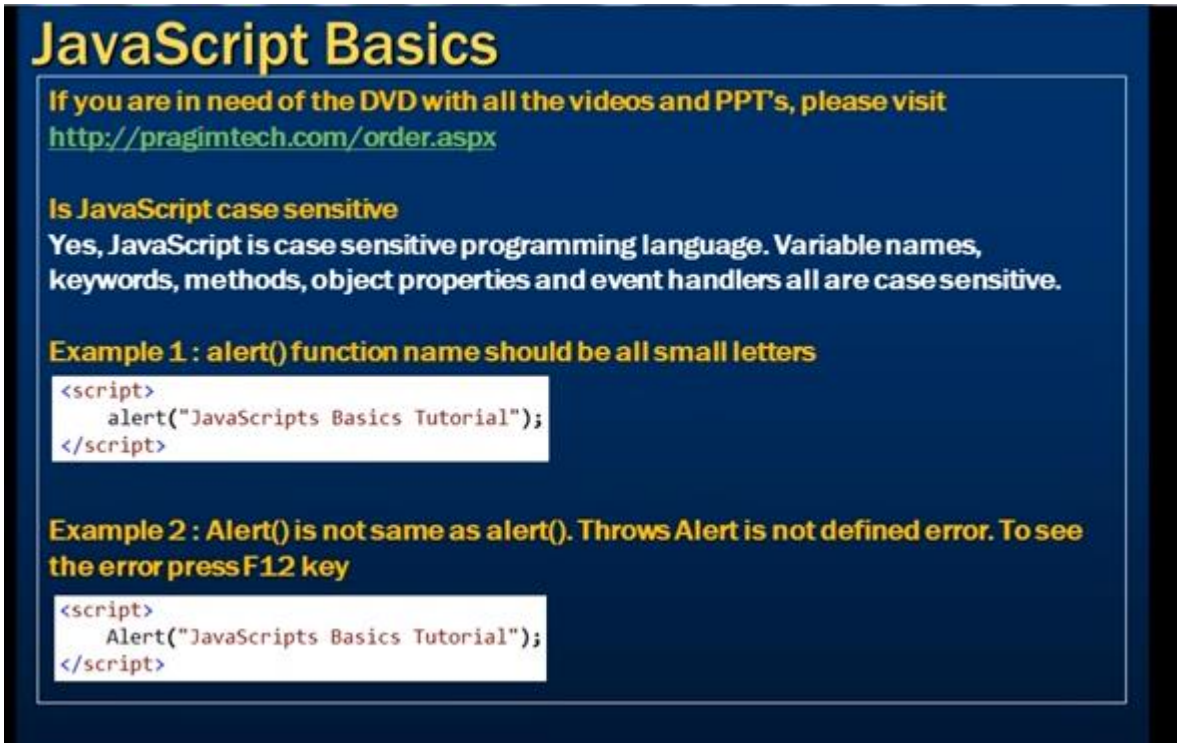
Advantages

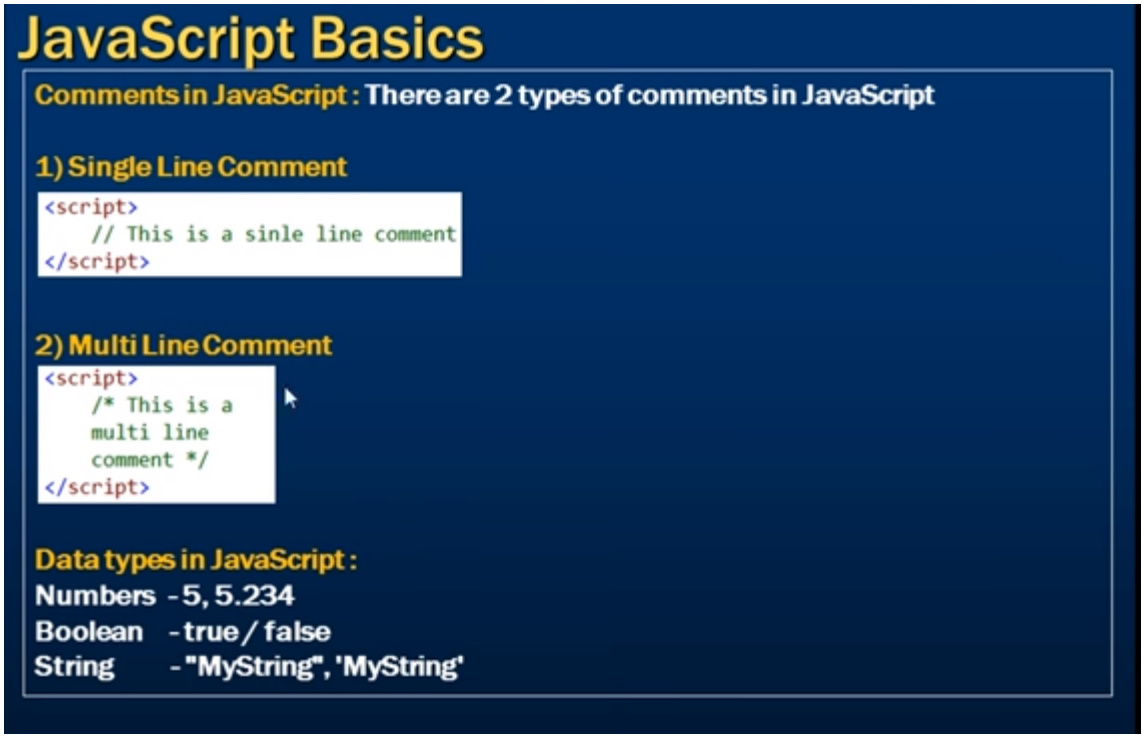
- **Javascript is executed on the client side**
  This means that the code is executed on the user's processor instead of the web server thus saving bandwidth and strain on the web server.

- **Javascript is a relatively easy language**
  The Javascript language is relatively easy to learn and comprises of syntax that is close to English. It uses the DOM model that provides plenty of prewritten functionality to the various objects on pages making it a breeze to develop a script to solve a custom purpose.

- **Javascript is relatively fast to the end user**
  As the code is executed on the user's computer, results and processing is completed almost instantly depending on the task (tasks in javascript on web pages are usually simple so as to prevent being a memory hog) as it does not need to be

processed in the site's web server and sent back to the user consuming local as well as server bandwidth.





JavaScript Basics

To create a variable in JavaScript use var keyword. Variable names are case sensitive.

In c# to create an integer variable we use int keyword

```
int X = 10;
```

to create a string variable we use string keyword

```
string str = "Hello"
```

With JavaScript we always use var keyword to create any type of variable. Based on the value assigned the type of the variable is inferred.

```
var a = 10;
var b = "MyString";
```

In C#, you cannot assign a string value to an integer variable

```
int X = 10;
X = "Hello"; // Compiler error
```

JavaScript Basics

JavaScript is a dynamically typed language. This means JavaScript data types are converted automatically as needed during script execution. Notice that, in myVariable we are first storing a number and then a string later.

```
<script>

    var myVariable = 100;
    alert(myVariable);

    myVariable = "Assigning a string value";
    alert(myVariable);

</script>
```

When a + operator is used with 2 numbers, JavaScripts adds those numbers.

```
<script>
    var a = 10;
    var b = 20;
    var c = a + b;
    alert(c);
</script>
```

Output → 30

JavaScript Basics

When a + operator is used with 2 strings, JavaScript concatenates those 2 strings

```
<script>
    var a = "Hello "
    var b = "JavaScript";
    var c = a + b;
    alert(c);
</script>
```

Output → Hello JavaScript

When a + operator is used with a string and a number, JavaScript converts the numeric value to a string and performs concatenation.

```
<script>
    var a = "Number is : "
    var b = 10;
    var c = a + b;
    alert(c);
</script>
```

Output → Number is 10

```
<script>
    var a = "50"
    var b = 10;
    var c = a + b;
    alert(c);
</script>
```

Output → 5010

JavaScript Basics

If you use a minus operator, numeric value is not converted to string

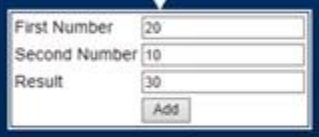
```
<script>
    var a = "50"
    var b = 10;
    var c = a - b;
    alert(c);
</script>
```

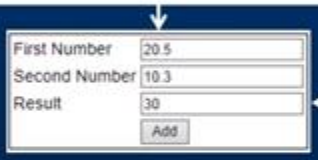
Output → 40

Converting Strings to Numbers

We need to explicitly do the conversion. This is when parseInt() function is useful

```
function addNumbers()
{
    var firstNumber = parseInt(document.getElementById("txtFirstNumber").value);
    var secondNumber = parseInt(document.getElementById("txtSecondNumber").value);
    document.getElementById("txtResult").value = firstNumber + secondNumber;
}
```

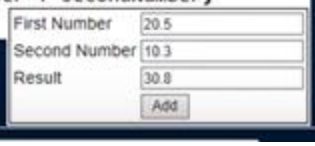




Decimal part is lost

To retain the decimal places, use parseFloat() function

```
function addNumbers()
{
    var firstNumber = parseFloat(document.getElementById("txtFirstNumber").value);
    var secondNumber = parseFloat(document.getElementById("txtSecondNumber").value);
    document.getElementById("txtResult").value = firstNumber + secondNumber;
}
```



Converting Strings to Numbers

NaN in JavaScript stands for Not-a-Number. In JavaScript we have isNaN() function which determines whether a value is an illegal number. This function returns true if the value is not a number, and false if not.

```
function addNumbers()
{
    var firstNumber = parseFloat(document.getElementById("txtFirstNumber").value);
    if (isNaN(firstNumber)) {
        alert("Please enter a valid number in the first number textbox");
        return;
    }

    var secondNumber = parseFloat(document.getElementById("txtSecondNumber").value);
    if (isNaN(secondNumber)) {
        alert("Please enter a valid number in the second number textbox");
        return;
    }

    document.getElementById("txtResult").value = firstNumber + secondNumber;
}
```

Now, when you leave first number and second number textboxes blank or if you enter text instead of a number, and when you click the Add button, you get relevant validation error messages as expected

Converting Strings to Numbers

Let's make the validation error message a little more relevant:

If the first number and second number textboxes are left blank, then we want to display the following validation messages

a) First Number is required
b) Second Number is required

If you enter text instead of number

a) Please enter a valid number in the first number textbox
b) Please enter a valid number in the second number textbox

Strings in JavaScript

If you are in need of the DVD with all the videos and PPT's, please visit http://pragimtech.com/order.aspx

A string is any text inside quotes. You can use either single or double quotes

```
var string1 = "string in double quotes" // string in double quotes
var string2 = 'string in single quotes' // string in single quotes
```

Concatenating strings : There are 2 options to concatenate strings in JavaScript. You could either use + operator or concat() method

Example : Concatenating strings using + operator

```
var string1 = "Hello"
var string2 = "JavaScript"
var result = string1 + " " + string2;
alert(result);
```

→ Hello JavaScript

Example : Concatenating strings using concat() method

```
var string1 = "Hello"
var string2 = "JavaScript"
var result = string1.concat(" ", string2);
alert(result);
```

→ Hello JavaScript

Strings in JavaScript

If you want single quotes inside a string, there are 2 options

Option 1 : Place your entire string in double quotes, and use single quotes inside the string wherever you need them

```
var myString = "Welcome to 'JavaScript' Training";
alert(myString);
```

→ Welcome to 'JavaScript' Training

Option 2 : If you prefer to place your entire string in single quotes, then use escape sequence character \ with a single quote inside the string

```
var myString = 'Welcome to \'JavaScript\' Training';
alert(myString);
```

→ Welcome to 'JavaScript' Training

Please Note : You can use the above 2 approaches if you need double quotes instead of single quotes inside a string

Strings in JavaScript

Converting a string to uppercase : Use toUpperCase() method

```
var upperCaseString = "JavaScript";
alert(upperCaseString.toUpperCase());
```

→ JAVASCRIPT

Converting a string to lowercase : Use toLowerCase() method

```
var lowerCaseString = "JavaScript";
alert(lowerCaseString.toLowerCase());
```

→ javascript

Length of string in JavaScript : Use length property

```
alert("JavaScript".length);
```

→ 10

```
var myString = "Hello JavaScript";
alert(myString.length);
```

→ 16

Strings in JavaScript

Remove whitespace from both ends of a string : Use trim() method

```
var string1 = " AB ";
var string2 = " CD ";
var result = string1.trim() + string2.trim();
alert(result);
```

→ ABCD

Replacing strings in javascript : Use replace() method. This method searches a given string for a specified value or a regular expression, replaces the specified value with the replacement value and returns a new string. This method does not change the original string

```
var myString = "Hello JavaScript";
var result = myString.replace("JavaScript", "World");
alert(result);
```

→ Hello World

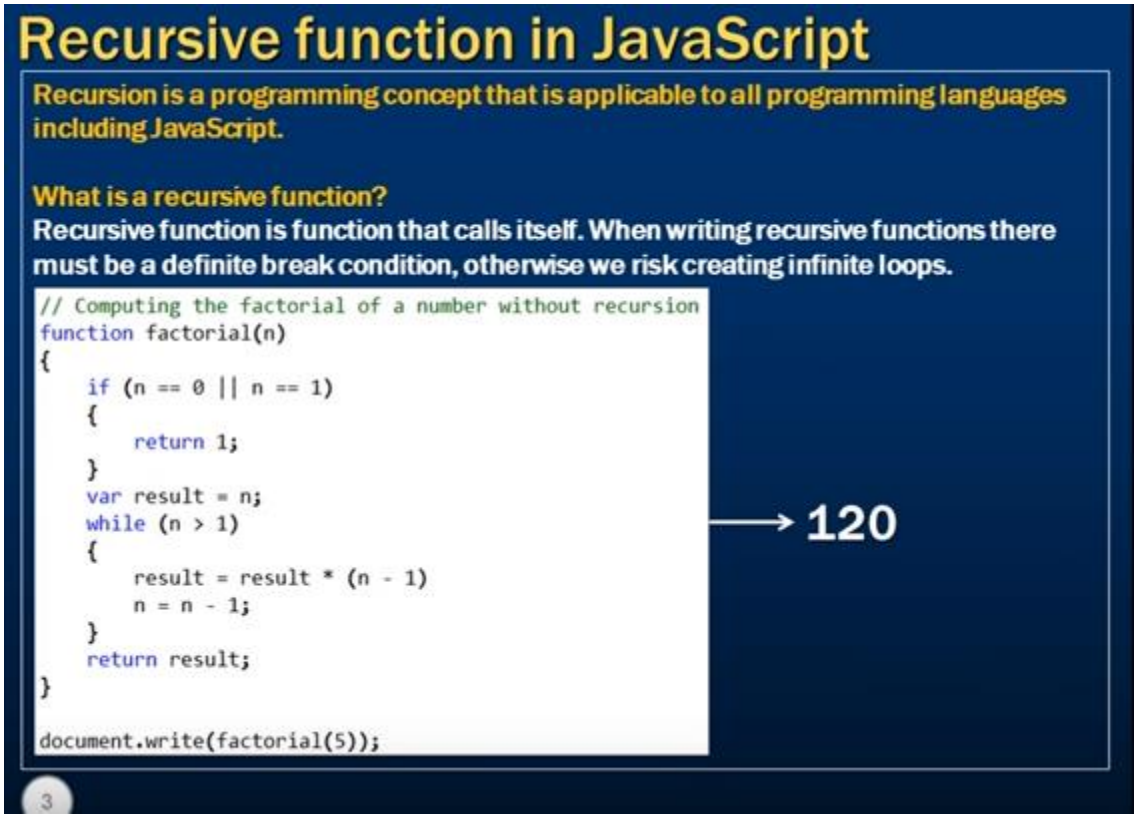
Conditional Statements in JavaScript

If you are in need of the DVD with all the videos and PPT's, please visit http://pragimtech.com/order.aspx

JavaScript code is executed from the first line to the last line. If for some reason you want to interrupt this flow and execute certain statements, only, if certain condition is met, then we use conditional statements.

JavaScript has the following conditional statements

if
if else
if else if else
switch
ternary operator - shortcut for an if...else statement



Creating Objects using JavaScript

1) **Using Literal**
2) **Using Construction**

Object Literal vs Object Constructor

Object defined with a function constructor lets you have multiple instances of that object. This means change made to one instance, will not affect other instances.

```
<script type="text/javascript">
    var emp = function ()
    {
        this.name = "John";
    }

    // Create an instance of employee
    // employee.name will return John
    var employee = new emp();

    // Create an other instance of employee
    // newEmployee.name will return John
    var newEmployee = new emp();

    // Change the name property of the newEmployee object
    newEmployee.name = "Mary";

    // Retrieve the name property from the original employee object
    // Notice that name is not changed to Mary, it is still John
    document.write(employee.name);
</script>
```

So, when to use one over the other: If you need multiple instances of the object use constructor function where as if you need just one instance of the object then use literal notation.

JavaScript & OOP

JavaScript is object oriented programming language. The following are the 4 pillars of any object oriented programming language. We will discuss examples of these in a later video session.

1. Inheritance
2. Encapsulation
3. Abstraction
4. Polymorphism

Objects in JavaScript can be broadly classified into 2 categories

1. Standard built-in objects
2. Custom objects

Standard built-in objects : So far in this video series, we have already seen many of the JavaScript standard built-in objects. Examples include string, array, RegExp, Date etc.

Example:

```
var currentDate = new Date();
document.write(currentDate.getFullYear());
```

JavaScript & OOP

Custom objects : In C#, to create a custom object, we create a Custom class and then create an instance of a class.

In JavaScript we don't have classes. Instead we use functions. In JavaScript there are two ways to create a custom object

1. Constructor function
2. Literal notation

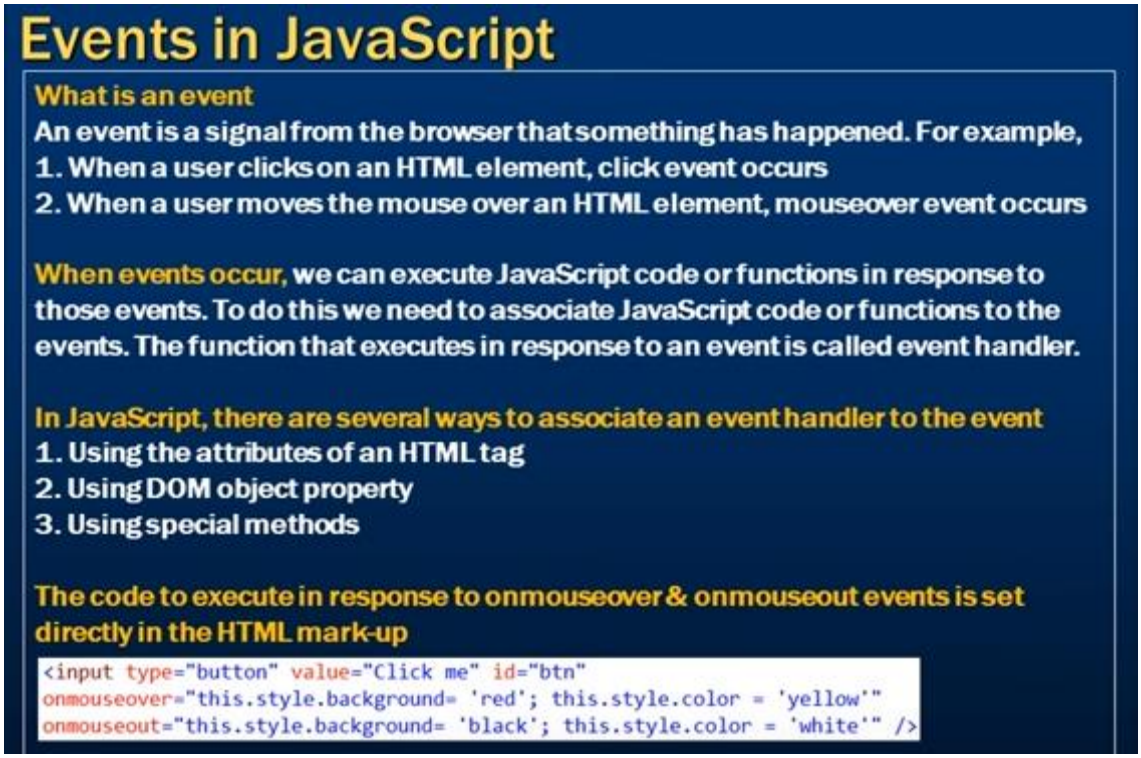
```
// Creating an object using constructor function
function Employee(firstName, lastName)
{
    this.firstName = firstName;
    this.lastName = lastName;

    this.getFullName = function ()
    {
        return this.firstName + " " + this.lastName;
    }
}

var employee = new Employee("Pragim", "Tech");

document.write("FirstName = " + employee.firstName + "<br/>");
document.write("LastName = " + employee.lastName + "<br/>");
document.write("FullName = " + employee.getFullName() + "<br/>");
```

→
FirstName = Pragim
LastName = Tech
FullName = Pragim Tech

Inheritance in JavaScript

Object oriented programming languages support inheritance. Since JavaScript is also an object oriented programming language, it supports inheritance

In Object oriented programming languages like C# and Java to implement inheritance, a class inherits from another class. In JavaScript, we don't have the concept of classes, so inheritance in JavaScript is prototype-based. This means to implement inheritance in JavaScript, an object inherits from another object.



Regular Expressions