Notes:

Keep in Mind

* If we add a number with string it returns the concatenated value otherwise in multiplication or division or subtraction it is return NAN
* With strict mode, you cannot use undeclared variables.
* If we used a string with the quotation then must be used escape liked \" \"

Differ with Other OOP Programming Language

* JavaScript is Prototype based object oriented language instead of class based language.
* It is easy to define new properties and methods in JS at any moment which is difficult in class based language like C++, C# or others.
* JavaScript has no concepts of Classes like traditional Object Oriented languages.

JavaScript Use Strict

* "use strict"; Defines that JavaScript code should be executed in "strict mode".
* With strict mode, we can’t, use undeclared variables.
* Strict mode is declared by adding "use strict"; to the beginning of a JavaScript or a JavaScript function.
* Declared at the beginning of a JavaScript file, it has global scope (all code will execute in strict mode)
* Example

"use strict";

x = 3.14; // This will cause an error (x is not defined)

* Declared inside a function, it has local scope (only the code inside the function is in strict mode)
* Example

x = 3.14; // This will not cause an error.

myFunction();

function myFunction() {

"use strict";

y = 3.14; // This will cause an error (y is not defined)

}

Why Strict Mode?

* Strict mode makes it easier to write "secure" JavaScript.
* Strict mode changes previously accepted "bad syntax" into real errors.

About JS

* JavaScript is a cross-platform, object-oriented scripting language. JavaScript is a small, lightweight language; it is not useful as a standalone language, but is designed for easy embedding in other products and applications, such as web browsers.

Client-Side JavaScript

* File manipulation and all other activities done in user computer or browser or not in server.

Server-Side JavaScript

* All activities done in server not in host computer and manage a relational database.

1. JavaScript nature

* JavaScript is case sensitive.
* JavaScript is white space insensitive.
* JavaScript execute each code line by line.
* JavaScript is weekly typed language.

1. Output

* document.write()

1. JavaScript Input Function

* prompt()

1. Debugging

* For console
* Firefox – f12
* Chrome – ctrl+shift+i

1. JS popular Library

* jQuery
* Prototype
* MooTools

1. Debugging

* Debugging is the process of testing, finding, and reducing bugs (errors) in computer programs

1. Comment

* // for single line comment
* /\* \*/ is used for multiple comments

1. Different types of operator in JavaScript

* typeof operator is used to find the type of a JavaScript variable
* in operator checks whether a variable is exists or not.
* parseInt parses its argument and returns an integer

1. Concatenation

* + operator is used for concatenation in string variable.

1. Add JavaScript to the Script

* <script src=”myscripts.js”></script>

1. Special Operators in JS

* In Operator

1. The in operator returns true if the specified property is in the specified object.
2. console.log("Mainul" in window);//In operator checks whether a variable is exists in window or not

* Number

1. console.log(Number('123asflkj'));//Number returns a number,converted from its argument including floating point

* parseInt

1. console.log(parseInt('123asflkj111'));//parseInt parses its argument and returns an integer

* Instanceof

1. The instanceof operator returns true if the specified object is of the specified object type.

* TypeOf

1. typeof operator is used to find the type of a JavaScript variable

* toString()

1. The toString() method converts an array into a String and returns the result.
2. The returned string will separate the elements in the array with commas.

Example

var fruits = ["Banana", "Orange", "Apple", "Mango"];

fruits.toString();

* The result of fruits will be

Banana,Orange,Apple,Mango

1. JavaScript Events

* OnMouseOver
* Content change when we placed the mouse on the text.
* Content change when we placed out the mouse from the text
* Onfocus
* onfocus event attracts anyone attraction when hits on the related element.
* document.theform.myname.onfocus=function() {

document.getElementById('mynamehint').innerHTML = "(Enter last name, then first)";

}

* Blur
* Blur event stops the attraction when leave the related element.

1. Closure Funciton

Closure - Closure is just a memory from where we inherit some of the properties or memory from your origin.

Closure-> Function+Scope

* Closures make it possible to hide away functions and variables.
* No one can access it except for the closure. It is encapsulated.
* The closure also has a sense of state.
* Closures are a powerful tool.

1. Closures

* A closure is a function that returns a function. The function that is returned (the inner function) is created inside the called function (the outer) so - due to the scoping rules we’ve seen - the inner has access to the variables and arguments of the outer.

1. Array Functions

* Array means one name store multiple values
* Array index is start from 0 or computers counting start from 0

Join and Pop

* The join method is used to join all array elements in a string
* We can also separate string element by usnig our customize separator like comma semicolon etc.

Pop

* The pop method is used to remove last element from the array.

Length

* The pop method is used to remove last element from the array

1. Difference between ID & CLASS

ID

* ID should be unique
* Like <div id = “widget”>My Widget</div>
* Only one element can have the name widget in the script.

CLASS

* Classes multiple element can have the same class name.
* like <p classes = “error”>An error occured</p>
* <p classes = “error”> An error occured </p>

1. Accessing Forms

* If we have 8 different JavaScript forms creates 8 arrays against 8 different forms the first form array 0 the second form array index 1 and so on.
* Forms collection is also an associative array that automatically associates a form’s id with the form itself. This means that you can also use the id of the form as an index to the forms collection.
* Access form using index number or using id as the key of the index.

var formByIndex = document.forms[0];

var formById = document.forms["contactForm"];

* If you have multiple forms on a page, it’s safest to use the id of a form as an index to the forms collection. Otherwise, if the order of the forms changes in the document source, you will have to change any numerical indices in your JavaScript code to match it.
* Avoid Accessing Forms from the document Object Although it’s possible to access a form directly from the document object (document.formId), this is an extremely inefficient method of locating an element, as it requires the browser to search the entire DOM tree for the corresponding node.
* To access the elements contained within a form, you just access its individual elements collection either by index or name/id:

var firstNameElement = document.forms["contactForm"].elements[0];

var lastNameElement = document.forms["contactForm"].elements["lastName"];

1. Overloading:

* Overloading is the ability of a single function to have multiple signatures.
* A function signature is made up of the function name plus the number and type of parameters the function expects. Thus, a single function can have one signature that accepts a single string argument and another that accepts two numeric arguments. The language determines which version of a function to call based on the arguments that are passed in.

1. Objects:

* Objects in JavaScript are class-free.
* Objects are useful for collecting and organizing data.
* Objects can contain other objects, so they can easily represent tree or graph structures.
* An object is an unordered list of properties consisting of a name (always a string) and a value.
* When the value of a property is a function, it is called a method.
* Objects are flexible they are hold many other data types like strings, array, functions, other objects and so many.
* Objects in JavaScript are dynamic, meaning that they can change at any point during code execution. Whereas class-based languages lock down objects based on a class definition, JavaScript objects have no such restrictions.
* In JS object is creating in two ways
* Constructor Function
* Constructors in JavaScript begin with a capital letter to distinguish them from non-constructor functions.

var object = new Object();

* Example of object look like by object initializer

var info = {

full\_name : “Ray Villalobos”,

title: “Staff Author”,

links: [

{ blog : “<http://iviewsource.com>”},

{ twitter : “http://twitter.com/planetoftheweb”}

]

};

1. Error Handling

* Errors can be coding errors made by the programmer, errors due to wrong input, and other unforeseeable things.
* Error handling is important the developer error.
* Display meaningful information to happy the user.
* Try->trying to execute some codes.
* The try statement lets you test a block of code for errors.
* The catch statement lets you handle the error.
* The throw statement lets you create custom errors.
* The finally statement lets you execute code, after try and catch, regardless of the result.
* In try block any code will not execute after getting an error
* Catch->if some reason fails the execution of code catch try to catch and handle the error.
* Catch (err) ->has a parameter and parameter name can be anything.
* Catch ->after catch block code execution will be continued.
* The throw statement allows you to create a custom error.
* The technical term for this is: throw an exception.
* The catch block catches the error, and executes code to handle it.
* Finally->Code block will always be executed regardless any error in try block or not.

1. JavaScript Form Validations:

* For attribute is used identify the input field which name is the name of input field id.
* Action:

Tells the browser the location of the form processor

Where to send the data to process

* Enctype:

How to encode the data

* Method – How to send the data
* Get – Send data thorough a url.
* Is default so it does necessary to include in the method section.
* Post -- Sends data in the background
* Used for longer forms.
* It’s necessary when sending file.
* More secure than get.

Defining methods

* A method is a function associated with an object, or, simply put, a method is a property of an object that is a function. Methods are defined the way normal functions are defined, except that they have to be assigned as the property of an object.

Using this for object references

* this, that you can use within a method to refer to the current object.
* In general, this refers to the calling object in a method.

Defining getters and setters

* A getter is a method that gets the value of a specific property.
* A setter is a method that sets the value of a specific property.