**JavaScript**

* Interpreted programming language
* E.g. Browser is the best interpreter for JS
* Used for adding programming login in HTML pages
* To make a html page dynamic (DHTML)
* Original name: LiveScript
* Standardized by ECMA (Europian Computers Manufacture’s Association)
  + ECMAScript 4
  + ECMAScript 5:
    - Getters/setters
    - Function overloading
  + ECMAScript 6
* To write JS Code
  + Using HTML
    - Use <script> tag to write js code
    - <script> may appear in both head and body sections
  + Using command line (console)
* Case sensitive language
  + firstName and FirstName are different variables
* Types
  + Internal
  + External
* JS DOES NOT support pointers ☺
* JS is loosely typed language

**Data Types**

* All data types in JS are inferred
* Automatically decided by JS by looking at the variable’s **CURRENT** value
* Types
  + Number:
    - used to keep both integer and float values
    - E.g.
      * num = 100;
      * salary = 10.15;
  + String:
    - E.g.
      * firstName = “steve”;
      * firstName = ‘steve’;
      * firatName = `steve`;
  + Boolean:
    - Can have only true or false value
    - E.g.
      * canVote = true;
  + undefined
  + object
    - Everything in JS is an object
    - Functions are also considered objects

**Pre-Defined Objects**

* console: used to represent the browser console
* window: used to represent the browser GUI
* document:

**Pre-Defined Values**

* NaN: Not a Number
  + “test” \* “test1” = NaN
  + data type: number
* Infinity:
  + Only when a number is divided by zero (0)
* undefined

**Function**

* function keyword is used to define a function
* function declaration and definitions CAN NOT separated
* Syntax:

function <function name>(<param list>) {

// function body here

}

* IMP
  + Function can not supply the param data type
  + There is no change in function signature (prototype) even if the function is returning a value
    - return keyword is used to return a value
  + function **CAN NOT** decide the number of parameters and data type of parameters

function function1(n1) {

// body

}

// n1 is declared as undefined

function1();

// n1 is declared as number (20)

function1(20);

// n1 is declared as string (“test”)

function1(“test”);

// n1 is declared as number (10)

// 20 and 30 will be ignored (**\***)

function1(10, 20, 30);

* + function **CALLER** decides the number of parameters and data type of parameters
  + function can be called before its declaration (definition)
  + may define multiple functions with same name
    - **ONLY THE LAST (bottom most)** definition will be kept and remaining definitions will be ignored
  + Every function receives a hidden parameter called as **arguments**
    - arguments is an array of all the values passed to the function (irrespective of the named parameters)
    - E.g.

function add() {

console.log(arguments);

}

add(10, 20); // arguments = [10, 20]

add(10, 20, 30, 40); // arguments = [10, 20, 30, 40]

* + Every function receives a hidden member named **this**
    - In case if the function is called without using an object this reference will refer to Object function

function test() {

console.log(this);

}

test();

* + - In case if the function is called by using an object then the object becomes this

function test() {

console.log(this);

}

var p = new Object();

p.myTest = test;

// p becomes this reference

p.myTest();

// prints

// p object {…}

* Function Alias
  + Another name given to a function
  + Variable of type function
  + E.g.

function function1() {}

// function alias

var myfunction1 = function1;

myfunction1();

* Anonymous Function
  + Function without a name
  + E.g.

var multiply = function(p1, p2) {

console.log(“p1 \* p2 = “ + (p1 \* p2));

};

**Array**

* Collection of objects
* May contain similar or dis-similar objects (values)
  + E.g.

var array = [1, 2, 3, 4]

var array = [1, “test”, 2, true, false, “test2”, 3, 4]

* to print / process all members in the array

for (var index = 0; index < array.length; index++ )

// code here

}

* to manipulate the values
  + push: to add a value at the last position
  + pop: to remove the last element from array
  + splice: to remove an item at an index

**Variable Scope**

* Local scope:
  + Declare a variable with **var** keyword
  + Function parameters are **ALWAYS** declared as local variables
  + E.g.

function function1 () {

// local

var num = 100;

}

* Global scope:
  + Declare a variable without using **var** keyword
  + Global variables CAN be declared inside a function
  + E.g.

// global

salary = 10.15;

function function1 () {

// global

num = 100;

}

**Conversion**

* string to number:
  + parseInt()
    - converts string to number in integer format
    - E.g.
      * parseInt(“10.20”); // 10
  + parseFloat()
    - converts string to number in float format
    - E.g.
      * parseInt(“10.20”); // 10.20
* number to string:
  + E.g.
    - “” + 20; // “20” (string)
    - ‘’ + 20; // “20” (string)

**JSON**

* JavaScript Object Notation
* Types
  + Object
    - Collection of Key-value pairs separated by Comma(,)
    - Use {}
    - E.g.

var movie = {

“title”: ”cars”,

“price”: 30

}

* + - Where title and price are keys while cars and 30 are values
    - To access values

console.log(“title: “ + movie.title);

console.log(“price: “ + movie.price);

* + Array
    - Collection of objects
    - Use []
    - E.g.

var movies = [

{ “title”:”cars”, “price”: 30 },

{ “title”:”moana”, “price”: 40 }

]

* + - To access all objects in array

for (var index = 0; index < movies.length; index++) {

var item = movies[index];

console.log(“title :” + item.title);

console.log(“price :” + item.price);

}

**jQuery**

* External JS library (collection of functions)
* Open source and free library
* Large community
* To load jQuery js
  + <script src=”js/jquery.js”></script>
* To use jQuery
  + Use $ to load jQuery functions
  + Syntax:
    - $(<css selector>).<function>()
    - E.g.

$(‘#div1’).hide();

* Animation functions
  + hide(), show()
  + fadeIn(), fadeOut()
  + slideUp(), slideDown()
  + animate()
* text related functions
  + html(): set/get html to/from an element
  + text(): set/get text to/from an element
  + val(): set/get value to/from an input element

**OOP JavaScript**

* Everything in JS is an Object, even functions are also considered as objects
* collection of properties and functions/methods
* To create an object/instance
  + Using Object function
    - E.g.

var person = new Object();

person.name = “person1”; // name is a property

* + Using Constructor Function
  + Using JSON
  + Using class: TypeScript
* Instance / object
  + Collection of properties and methods
    - Where a property is used to store the data (state)
      * To add/access property in an instance
        + Use dot syntax (.)

var person = new Object();

person.name = “steve”;

person.address = “USA”;

console.log(“Name: “ + person.name);

console.log(“Address: “ + person.address);

* + - * + Use []

var car = new Object();

car[“model”] = “Nano”;

car[“company”] = “Tata”;

console.log(“Model: “ + car[“model”];

console.log(“Company: “ + car[“company”];

* + - Where a method is a function alias added inside an object (behavior)

function printRecord() {}

var person = new Object();

person.name = “person1”;

person.myPrintRecord = printRecord;

console.log(“name : “ + person.name);

person.myPrintRecord();

**Object**

* Built-in function
* Using Object JS allows developer to create an instance
* Root function