JavaScript

# About JavaScript

* Scripting language. It’s fast becoming lingua franca of web
* Popularity due to cross browser support & creating rich clients with AJAX
* Increasing usage on server side with Node.js
* Simple & fast & light
* <https://developer.mozilla.org/en/Javascript/Guide>

# Basic

* **//** single line comment
* **/\*** multi line comment **\*/**
* **var varName;** //declares a variable
  + **var varName=value;** //declares & assigns the value of variable.
  + **Typeof(varName)** // returns the dataType of the variable.
  + **a=1;** // a is a global variable & is part of window object. window.a is true
  + Use local variables & not global to avoid conflicting situations
* **null** // absence of value. It evaluates to false
* **undefined** // unknown value. It evaluates to false. Returned if non-existent object property is called.
* Objects // Everything in JS is object except string, Boolean, number , null & undefined
  + **var myObject = { property1 :value1, property2:value2,…};** //literal notation
  + **var emptyObject={};** // empty object. It evaluates to true
  + **var myObject =new Object(); myObject.name=”Smita”;** //using Object() constructor
* Use **=== & !==** for comparison. As == & != are type coercion operators
  + Objects are only equal to themselves even though their contents may be same.
  + 1==”1” or “”==0 will return true as type conversion occurs
* **Block {…}** 
  + Variables defined inside a block are visible outside as well unlike other languages
* **if (condition) {true block} else {false block}** //control flow
* **switch (var) { case “str” : statements break; ….; default : statements}** //multi-way control flow
* **for ( init var ; cond; inc var) {…}** //iterate
* **for(var propertyKey in Object) { …}** //iterate objects. To access value **Object[propertyKey]**
* **while (condition) {…}** //pre tested loop
* **do {…}while(condition)** // post tested loop. Executed at least once
* Error Handling
  + Throw an exception when unusual condition occurs
  + **try{**

**throw { name: errNAme, message:errMsg};**

**}catch(e){**

**Handle exception here //can have multiple catch blocks**

**}finally { executed irrespective of exception}**

* **eval** // Interprets JS strings. slow & insecure so never use it

# Function

* Function is inherited from object
* **function FName (PName1,PName2,…) {**

**code // can have return statement. “;” is the delimiter**

**} ;**//parameters are optional

* + **var FuncVar = function F1 (p1,…) { code };** //named & assigned function
  + **var FuncVar = function (p1,…) { code };** //Anonymous & assigned function. No name while debugging
  + **(function F1 (p1,…) { code })(A1,…);** // Anonymous & immediately invoked. No namespace pollution. Ex implement plugin for jquery
* Function Invocatiobn : **FName(A1,A2,….); or FuncVar(A1,A2,…);**
  + If more arguments are passed than parameters, JS ignores more arguments.
  + If less arguments are passed than parameters, JS sets extra parameters to undefined.
  + No Function overloading in JS
  + Object parameters are passed by reference
  + Functions can be called recursively. Break condition is important for recursion to end.
* **arguments** // local Object of all function parameters available in all functions. Can act like array
* Nested Function : Inner function has access to variables defined outside of inner function (Closure)

# Types and Libraries

## String

* var str1 =”double quotes” or var str2=’single quotes’ //include other in string
* No multi line string
* **\n** – New Line **\” or \’** – string literal delimiter **\u[code]** – Unicode symbol ex \u00A9
* **“+”** // String concatenation
* charAt(index) // returns the character (as a string) for the specified position. 1st char=0
* indexOf(string) // returns the index of the specified string. 1st char=0
* replace(from, to) // replaces the 1st occurrence of string from in string to
  + use regex to replace all instance (g) ex str.replace(/3/g,4);
* search (regex) // returns the index of the regex search pattern
* slice(index1,index2) // returns a substring of a string from index1 to index2
* split(separator) // splits a string on separator
* toLowerCase() //converts to lowercase
* toUpperCase() //converts to uppercase

## Numbers

* All Numbers are floating point. Decimal fractions are not exact
* Standard operator : **-,+,\*,/,%**
* **.toFixed(n)** // terminate to given decimal places.
  + Another technique to round esp for money values is \* 10 before operation starts
* isNan(p) // returns true if parameter is NaN or string or invalid operation & false for valid number
* parseFloat(string) //converts string to a number
* Math //Class provides standard mathematical functions
  + abs(n) floor(n) ceil(n) pow(a,n) random() //between 0 &1 round() //rounds to integer
  + Math.PI //returns value of PI

## Array

* Indexed collection of any datatypes
  + **var arr=[v1,v2,…];** //array declaration
  + **arr[i]** //access element
* **arr.length** //gets array length
* **arr.push(val) & arr.pop()** //array behaves like a stack
* **arr.reverse()** //reverses the array
* **arr.splice(start,count)** //removes part of array
* **arr.sort()** //sorted alphabetically even number array
  + **arr.sort(function(first,second){return first-second});**
* underscore.js // lots of functions to work with array <http://documentcloud.github.io/underscore/>
  + **\_.each(arrayObject,function(p1){ code});** //functional usage
  + **\_(arrayObject).each(function(p1){ code});** // object-oriented usage

## Regular Expression

* Tool for string pattern matching (search, replace & extract strings)
* **\w** //any word **\d** // any digit **.** //any character **\** //escapes special characters
* **\*** // 0 or more
* **[]** //set of characters
* **{}** // quantifies a match ex .{2} is match any 2 characters
* **()** //capturing group
* **/pattern/** //Regular expression literal
* Examples
  + “abcde”.search(/c/); //returns 2
* **Res=exp.exec(input)** where input is **/<b>(.\*)<\/b>/**
  + //returns all characters between bold tag in exp to an array
  + res[0] is matched string (with tag) while res[1] is captured group
* **“Smita Vishal”.replace(/(\w+) (\w+)/g, function (match,capture1,capture2){**

**return capture2.toUpperCase() + “,” +capture1;**

**}** //returns matching 2 words & replaces all occurences

## Date

* var d =new Date(yyyy,mm,dd); //sets d as an date
* Date() // gives current date
* .toUTCString() //converts to UTC string
* Date1-date2 gives number of seconds elapsed
* Datejs // <http://www.datejs.com> help with date functions

## JSON (JavaScript Object Notation)

* Lightweight, readable, alternative to xml. Used with AJAX web apps
* Use json2.js (<http://json.org/js.html>)
  + Parse //converts JSON to JS Objects
  + Stringify //converts JS Objects to JSON

# Testing & Debugging

* Fire bug for debugging on Firefox
* YSlow for improving performance based on <http://developer.yahoo.com/performance/rules.html>
* QUnit//tool to test JS Pages <http://docs.jquery.com/Qunit>
  + module(moduleName);
  + test(testName,callback);
  + Assertions:
    - ok(Boolean,message);
    - equal(actual,expected,message);
    - deepEqual(actual,expected,message); //deep recursive comparison of 2 objects
* BDD (Behavior Driven Development) //given when then tests
  + Jasmine is BDD framework for testing JS code <http://pivotal.github.com/jasmine/>