Client-side web scripting

- Allows programs (i.e. scripts) to be downloaded from a web server and executed in the client environment (e.g. browser)

* Common client-side scripting technologies JavaScript, ECMAScript, Jscript, VBScript, ActionScriptJava Script, Java Applets, ActiveX Controls, Flash Animations, Microsoft Silverlight, Adobe Integrated Runtime (Adobe AIR)

Java Script:

* Developed circa 1995 by Brendan Eich at Netscape Communications as the scripting language for the Netscape Navigator Browser
* Formerly called Mocha, then LiveScript, then JavaScript
* Standardized by ECMA International as ECMAScript
* Latest version: JavaScipt 1.8.5, ECMAScript 6 [ECMA-262 7th Edition]
* JavaScript Frameworks:
  + script.aculo.us, jQuery, MooTools, Prototype, Dojo Toolkit, Angular-s, ReactJS, Vue, Ember, Meteor, etc.
* JavaScript + DOM/BOM + CSS + (X)HTML = DHTML
* JavaScript code is executed in response to document event
* JavaScript code in (X)HTML pages can be executed “on the fly” as the document is rendered.

3 ways of associating java script with HTML:

1. Externally Linked Script – JavaScript that is found outside of the HTML document and whose file extension name has .js property.
2. Embedded Script – JavaScript hat is found within the HTML document
3. Inline Script – Script that is found within the HTML document body and has the highest priority out of the three ways in associating javascript with HTML

2 Types of Code:

1. Global/Top-level Code – executes instantly when the java script has been executed
2. Function Block Code – it is a block of code that only executes when being called by its function name.

Attributes of JavaScript:

Attributes of <script> tag:

1. <script defer> – which waits until the document has been rendered then executes the script.
2. <script async> – is short for asynchronous, which runs the document and as soon as the script becomes available, run the script.
3. <no script> - is for security purposes of blind side scripting

Attributes of element.attributes:

window.document.innerText – edit the content of the specified target text without the markup element

window.document.outerText – edit the content of the specified target text with the markup element

window.document.textContent - edit the content of the specified target text content

window.document.innerHTML - edit the content of the specified target html tag without the markup element

window.document.outerHTML - edit the content of the specified target html tag with the markup element

* Window object – run time environment

Example:

* window.navigator.vendor
* “Google Inc”
* window.navigator.appName
* “Netscape”
* window.screen.width
* 1364

Attributes of document.getElementBy():

* document.getElementById()
* document.getElementByName()
* document.getElementByTagName()

HTML DOM querySelector() Method: this method returns first element that satisfies a specified CSS selector in the document.

* document.querySelector()
* document.querySelectorAll()

Basic Language Features:

1. Paradigm: Imperative Programming Language
2. Type System and Scoping rules:
   1. Loosely typed
   2. Dynamic Data typed
3. JavaScript uses JIT Compiler
4. 3 keywords inn JS Ecma Script:
   1. var – creating a global object property
   2. let – are standalone objects, declares a block scope local variable, optionally initializing it to a value.
   3. const – are standalone objects, the value declared in constant cannot by re-assigned a new value and it cant be re-declared

Data Types of JavaScript:

1. Simple/Primitive values:

* Boolean = true/false
  + Falsy - value is a value that translates to false when evaluated in a Boolean context. “not really false but equals to false”
  + Truthy - value is a value that translates to true when evaluated in a Boolean context. All values are truthy unless they are defined as falsy. “not really true but equals to true”
* Numbers = decimal, hexadecimal notation
  + Binary – Syntax: 0b<number>
  + Octal – Syntax: 0<number>
  + Hexadecimal – Syntax: 0x<number>
  + Exponential – Syntax: e<number>
* Strings
  + Single quote delimited
  + Double quote delimited
  + String interpolation syntax: ‘*String*’+a or ‘*String*:${a}’

Simple/Primitive types:

* Boolean type: Has two values, which is either true or false.
* Null Type: is an empty reference to an Object
* Undefined Type: anything that is not defined
* Number Type: has value/s that takes form of a digit or a sequence of digits
* String type: has value/s that take form of a letter or a sequence of letters
* Symbol Type: is a unique immutable primitive value and may be used as a key of an Objet Property

1. Composite (Objects) Types
   * Core JavaScript Objects
     + Object, Number, Boolean, String, Date, Math, Global, RegExp, Error
     + Arrays (Array)
     + Functions (Function, Arguments)
   * DOM Objects
     + Anchor, Applet, Attr, Comment, DOMException, DOMImplementation, DocumentFragment,
     + Element, Event, Form, Image, Input, Layer, Link, Node, Option, Select, Style, Text, TextArea

Functions in Java Script

Functions

* is the building blocks in JavaScript.
* Functions is a procedural abstraction
* Function is also treated as object
* Functions can be recursive; functions that call themselves
* Functions can be nested
* Function arguments can have default values;
* Function has Rest parameters; which allows us to represent an indefinite number of arguments as an array (…others)

Function Expression

* No function name
* Enclosed in a parenthesis
* Can be used as a Immendiately Invoked Function Expression(IIFE)

Anonymous Function Syntax

* Are function expressions that are not hoisted onto the beginning of the scope.
* Cannot be used before they appear in the code
* Fat Arrow Syntax: “=>”
  + Has a shorter syntax compared to function expressions and lexically binds the *this* value
  + Are always anonymous
  + Note: the *this* keyword – is a reference to the current object

Arrays

2 ways of using arrays:

1. Using array as objects
   * Arrays in javascript is considered to be a list objects that has proper indexes in order to access each individual value in the list.
   * Array’s length nor types of elements is not fixed and it can change anytime in the run-time duration.
   * Eg:
     + var students = [“Student1”,”Student2”,”Student3”,”Student4”];
     + console.log(students[0]);
     + > Student1
2. Using array as literals
   * Arrays can also be a treated as a key and value relationship.
   * This means that you can call that specific string value if you don’t know the index of the value in the array list.
   * Eg:
     + var students = [“Student1”:”You”,”Student2”:”me”,”Student3”:”Male”,”Student4”:”Female”];
     + console.log(students[‘Student1’]);
     + >You

Array Destructuring – is an expression that lets you assign variables to each of the element in the array and therefore making it possible to call them one by 1 but not calling the array itself.

Eg:

* + - var students = [“Student1”,”Student2”,”Student3”,”Student4”];
    - var [a,b,c,d] = students;
    - console.log(a); //Student1
    - console.log(b); //Student2
    - console.log(c); //Student3

Array indices can be non-contiguous – array values can be non-contiguous after it has been declared if you call the index where you skip, the expression will just return *Undefined*.

3 General Methods:

1. Mutator Methods: This method modify the array
   * Array.prototype.copyWithin()
   * Array.prototype.fill()
   * Array.prototype.pop()
   * Array.prototype.push()
   * Array.prototype.reverse()
   * Array.prototype.shift()
   * Array.prototype.sort()
   * Array.prototype.splice()
   * Array.prototype.unshift()
2. Accessor Methods: This method do not modify the array and return some representation of the array
   * Array.prototype.concat()
   * Array.prototype.includes()
   * Array.prototype.indexOf()
   * Array.prototype.join()
   * Array.prototype.lastIndexOf()
   * Array.prototype.slice()
   * Array.prototype.toSource()
   * Array.prototype.toString()
   * Array.prototype.toLocalString()
3. Iteration Methods: This method takes the length of the array and any element added beyond this length from within the callback is not visited.
   * Array.prototype.entries()
   * Array.prototype.every()
   * Array.prototype.filter()
   * Array.prototype.find()
   * Array.prototype.findIndex()
   * Array.prototype.forEach()
   * Array.prototype.keys()
   * Array.prototype.map()
   * Array.prototype.reduce()
   * Array.prototype.reduceRight()
   * Array.prototype.some()
   * Array.prototype.values()
   * Array.prototype[@@iterator]()

Objects

* Are prototype based objects.
* Uses the curly braces ( { } ) in order to denote a list of objects

Object Destructuring – is an expression that lets you assign variables to each of the element in the object and therefore making it possible to call them one by 1 but not calling the object itself.

Eg:

* + - var students = {“Name”:”John”,”Age”:21,”Gender”:”Male”,”Course”:”BSCS”};
    - var {“Name”:a,”Age”:b,”Gender”:c,”Course”:d} = students;
    - console.log(a); //”John”
    - console.log(b); //21
    - console.log(c); //”Male”
    - console.log(d); //”BSCS”