WK 4

**JAVASCRIPT PROGRAMMING FOR WEB APPLICATIONS**

-**JavaScript Language: Overview and Syntax**

-JavaScript derived from EMCAScript standard, originally designed to run on Netscape Navigator browser

-when JavaScript interpreter is embedded in browser, result Is ability to create dynamic web pgs

-adds behavior to otherwise static web content

-JavaScript code acts on document object model that web browser generates

-Ajax or Asynchronous JavaScript and XML

-ways that server programming and browser scripting work together in architecture

-term Ajax encompasses more than asynchronous server calls through JavaScript and XML

-represents techniques that provide richer, interactive web apps through HTML, JavaScript, Cascading style sheets, modifying web page through Document Object Model

-now a days- JSON more commonly used instead of XML

-in JavaScript there are 5 primitive types that are associated with various primitive values

-Number: all numbers (ex. 0 or 3.1412)

-represents both integer and floating point values

-ex. NaN (not a number) & infinity

-integers can be coded as base 10 (decimal), base 8 (octal, or base 16 (hexadecimal)

-ex. 16 (decimal), 020 (octal), 0x10 (hexadecimal)

-All strings (ex. “Hello World”)

-delimited bby either double or single quotation marks

-Boolean: values that are true or false

-same as primitive type number, string, and Boolean can be wrapped in counter parts

-Null: value is null

-Undefined: value undefined, since data type has not been assigned

-all other non-primitive data types are objects

-Wrapper objects have same name as primitive type, except start with uppercase letter

-use special methods such as valueOf & toString methods to convert between object and primitive literals

-typeOf function used to find out data type of supplied operand

-Arrays specialized collection objects that aid programmer in storage and retrieval of data by indexed keys

-use zero-based indexing scheme: meaning first element of array has index of zero

-can be declared by using array constructor or array literal

-when constructor: use new array keywords & specify array elements as parameters of new array

-literals: created by declaring array within square brackets

-then assign to variable

-Data object specialized object that is used to hold date & time

-constructor is format: new Date

-name property identifies type of error such as RangeError

-instance that is created when numeric value of parameter is outside of valid range

-besides generic error, 6 other core errors

-EvalError

-ReferenceError

-SyntaxError

-**JavaScript: Variables and Control Statements**

-variables declared with the “var” keyword followed by variable name

-ex. Var age

-can be declared and initialized in one step

-ex. Var age=54

-do not need to declare data type since JavaScript is loosely typed language

-variable assumed data type from field during assignment, type of variable can change during program execution

-variables that have not been assigned value or not initialized, is not zero, or empty string, it is “undefined”

-rules for variable names, or identifiers

-name must start with letter, underscore (\_), or dollar sign ($)

-subsequent character can be digits [0=9]

-identifiers are case-sensitive

-scope of variables

-declared within function have scope local to function

-declared outside of function have global scope, can be used elsewhere in JavaScript program

-variables declared without var keyword have global scope

-conditional statements are sets of commands that are used to perform different actions for different conditions

-ex. IF statement is way program logic decides which path to take based on current values of variables or object properties

-syntax for decision begins with keyword IF, followed by condition to test; then statements that run if conditions yields a true results

-true processing follows after test condition and is delimited by braces, unless single statement

-if conditions resolves to false, statement that follows “else” keyword are executed

-indentation of compound control not required, however programmer can find it helps in deciphering control statement

-no block statement in JavaScript, is in Java

-having none means variables declared inside one IF condition can be used outside scope of that condition

-JavaScript supports switch statement as alternative to IF then ELSE

-condition tested placed in parentheses that follow switch keyword

-parameter can evaluate any number or string

-labels in case statement enclosed in quotation marks when labels represent string values

-program looks for label that matches value then transfers control running program

-if no match, program looks for optional default clause, if found, transfers control, issues associated statement

-break keyword used to prevent code from automatically falling into next case clause

-FOR loop repeats series of statements for any number of times

-takes 3 parameter arguments

-namely initial value

-condition being tested

-increment expression

-when FOR loop runs following occur

-First: initial expression set

-Second: conditional expression evaluated

-if true, loop statements run and increment expression updated

-if false, FOR loop terminates

-if loop doesn’t terminate, control returns to second step and expression evaluated again

-WHILE loop another common

-loop repeats while conditions remains true

-assumes condition reaches conclusion and then exits loop

-make sure evaluates to false at some point; otherwise loop never terminates

**-JavaScript: Functions and Prototypes**

-function is block of code which can be called from any point in script after being declared

-function made up of following:

-keyword function

-name of the function

-parentheses, with optional parameter arguments

-curly braces, with logic

-lastly, function block is the optional return statement that returns control back to whatever called function

-this function is example of ‘add’

-add function takes 2 parameter arguments and returns sum of arguments or concatenates the 2 arguments if they are strings

-do not specify data type for arguments of function

-they are determined by values of arguments that are being passed to function

-if no specific return type declared, function returns whatever type is required

-ex: function add(n,m){return n+m}

Var x – add(1,2); // returns 3

X = add(“hello”,”world”); // returns “helloworld”

-by using prototypes, can easily define properties and methods for all instances of particular object

-prototypes exist for all JavaScript objects that can be created with new keyword

-all object constructors create objects that inherit properties and methods that are defined by prototype for that object

-object that gets instantiated inherits current state of prototype

-Scripts can override prototype properties and functions

-if do so, changes affect current working instances of objects that match prototype

-functions usually declared first, not run until you specifically ask them to

-auto-invocation or self-executing functions start running immediately after being declared

-can also be unnamed or anonymous functions

-have format that is shown on code block

-often used to initialize data or declare DOM elements on page

-Client-Side JavaScript: with HTML

-client-side script is program that accompanies HTML document or might be embedded directly in HTML document itself

-script program runs on client device when document loads, or some other time such as when link is activated or when button is clicked

-scripts offer authors means to modify and extend HTML documents in highly interactive ways

-can run after HTML document is loaded

-can be used to validate forms or to process input as it is typed

-can be triggered by events that occur on web page: such as clicking on button

-can be used to dynamically create document elements on HTML page

-2 ways to used Script tag to include scripts in HTML document

-1: <script>

// JavaScript code

</script>

Shows how can include script directly inside HTML

Method is good for short scripts

-2: <script src-“/source/script.js”></script>

Uses src attribute to point to external script file

-method has several use cases

-importing JavaScript libraries for complex interactions

-using same script across several HTML documents

-when scripting is not available- user disables or page does not support

-to accommodate place content for alternative path within <noscript>

-scripts can be run on detection of certain events that happen when page is running in browser session

-ex. Onload even can run script when browser finishes loading page or function can be performed when onclick event occurs

-even occurs when pointing device is clicked over element, such as button, that declares handler for even

**-Client-Side JavaScript: with DOM**

-Document Object Model (DOM) is programming interface between HTML or XHTML and JavaScript

-is browser-based interface for apps and script to dynamically access and update content, structure, style of docs

-World Wide Web released 4 levels of DOM specs

-Level 1

-representation of basic DOM for browsers

-window object is top of hierarchy and controls environment that contains doc

-history object keeps internal details about recent history in browser

-location object contains info about URL

-navigator is object representation of client internet browser or user agent

-no standard applies to navigator object, so property values returned when running queries on navigator object are not consistent across browser

-screen object used to derive info about users screen

-ex. Dimensions of display screen

-useful for determining screen size to run on mobile devices

-doc object provides access to all HTML elements within page

-each doc that gets loaded becomes doc object

-window object outermost global container of all object in DOM hierarchy

-when loading page, automatically created for you

-can now access properties and functions from JavaScript code

-in client-side JavaScript, window object serves as global object and everything in DOM takes place in window

-wind.alert, window.confirm, window.prompt dialongs come from this

-window.alert method can be coded more simply as alert with message argument

-Object diagram can also be represented as tree

-brancehs of tree termed nodes

-2 types of nodes in W3C DOM: element nodes, text nodes

-element nodes: html, head, meta, title, body

-text nodes: nodes that contain actual text that go between element star tag and end tag

-DOM level 2

-includes line feed text node before paragraph and input elements

-input elements include text node that contains all text that follows input tag, additional line follows input element

-DOM level 0 for form portion of doc would only have form, p, input boxes

-DOM level 2 adds carriage returns, tabs, spaces: sometimes referred to as “white space”

**-JavaScript DOM Objects**

-W3C DOM level 2 defines 12 different nodes

-7 have direct applicability in HTML docs

-each node type named constant that is also represented by integer value

-ex. ELEMENT\_NODE type represented by integer 1

-ATTRIBUTE\_NODE represented by integer 2

-TEXT\_NODE represented by integer 3

-in DOM tree, node name for ELEMENT\_NODE type is name of element or tag

-you see these properties when you view DOM tree for an HTML page in browsers developers tools, like Chrome’s DevTools

-How to access elements on HTML page with DOM API?

-when doc is loaded, browser creates arrays for forms, images, anchors, links, applets, and embeds

-then places all objects of each type into these arrays

-arrays are indexed as they occur in source doc

-first index of each array starts at 0

-Each of array types, such as forms[], contains array elements[] with each index element being fields or buttons that occur in that form

-ID attribute identifies element in a doc

-used by scripts to refer to the element with a name that matches value of ID attribute

-in order to assign scriptable reference name to HTML element with ID attribute, use:

-ID must be unique name in doc

-name must be in quotation marks when assigned

-name must not start with numeric digit

-document.getElementById- used to return node object, name is parameter argument

-suggested same value is used for both ID and nae attribute

**-JavaScript APIs**

-to retrieve node reference for element of doc give ID

-use document.getElementById, pass ID value as argument

-getElementsByTagName retrieves node list of elements with specified tag

-node list contains array of elements in doc

-src attribute common property for img tag

-indicates location of image source

-document.write() functions adds script-generated HTML to doc

-can use DOM API function document.createElement(TagName) to create element I current doc

-after creation, can use any number of functions to place element in appropriate location within doc

-ex’s. include insertBefore, appendChild, replaceChild

-function element.innerHTML retrieves or sets contents of HTML element

-innterHTML returns all child elements as txt string

-can use to change contents of HTML element by setting txt string that can include HTML tags

-this removes all current child elements

-can use element.style method to retrieve or set incline CAA style for particular element

-if use, it overrides any setting form CSS style sheet with one specific style

-function element.getAttribute(attrName) retrieves value of specified element, if exists

-to open new browser window

-use window.open()

-returns reference to new window object

-use window.onload

-an event handler that executes method when web page has finished rendering doc

-use window.dump(“message”)

-writes string into console for web browser

-use window.scrollTo(x-value, y-value)

-scrolls browser to specific set of coordinates

**-Hands-On Lab: JavaScript**

-JavaScript- client side scripting commonly used to create dynamic web pages

-helps to change web page contents dynamically as well as enabling to validate forms and perform other actions

-Lab will accept person’s name and email ID then perform simple validation on entered input

-<script> used to embed executable code, usually JavaScript into HTML page

-can contain scripting statements or refer to external script file

-use type attribute to specify scripting language

-can put tag anywhere in doc

-a function can be called any number of times

<!DOCTYPE html>

<html>

<head>

<title>Contact Details</title>

<script type="application/javascript">

function checkdata(){

//Create references to the input elements we wish to validate

var username = document.getElementById("name");

var emailid = document.getElementById("email");

//Check if username field is empty

if(username.value == ""){

alert("Please enter the name");

username.focus();

return false;

}

//Check if email field is empty

if(emailid.value == ""){

alert("Please enter the email");

emailid.focus();

return false;

}

//If all is well return true.

alert("Form validation is successful.")

return true;

}

</script>

</head>

<body>

<h2>Enter your contact Details:</h2> <br>

<form id="form1" onsubmit="return checkdata()">

<label for="name">Name :</label>

<input type="text" id="name" name="name">

<br>

<br>

<label for="email">E-mail ID :</label>

<input type="text" id="email" name="email">

<br>

<br>

<input type="submit" value="Submit">

<input type="reset" value="Reset">

</form>

</html>

-with Zipcode label

<!DOCTYPE html>

<html>

<head>

<title>Contact Details</title>

<script type="application/javascript">

function checkdata(){

//Create references to the input elements we wish to validate

var username = document.getElementById("name");

var emailid = document.getElementById("email");

var zipcode =document.getElementById("zipcode");

//Check if username field is empty

if(username.value == ""){

alert("Please enter the name");

username.focus();

return false;

}

//Check if email field is empty

if(emailid.value == ""){

alert("Please enter the email");

emailid.focus();

return false;

}

//Check if zipcode field is empty

if(zipcode.value == ""){

alert("Please enter the zipcode");

zipcode.focus();

return false;

}

//If all is well return true.

alert("Form validation is successful.")

return true;

}

</script>

</head>

<body>

<h2>Enter your contact Details:</h2> <br>

<form id="form1" onsubmit="return checkdata()">

<label for="name">Name :</label>

<input type="text" id="name" name="name">

<br>

<br>

<label for="email">E-mail ID :</label>

<input type="email" id="email" name="email">

<br>

<br>

<label for="zipcode">Zipcode ID :</label>

<input type="zipcode" id="zipcode" name="zipcode">

<br>

<br>

<input type="submit" value="Submit">

<input type="reset" value="Reset">

</form>

</html>