CSC 211 – INTRODUCTION TO WEB PROGRAMMING I

PART I – JAVASCRIPT

BY

MUHAMMAD S. ALI

JavaScript Versus ECMAScript

- ECMAScript is the official name for JavaScript. A new name became necessary because there is a trademark on Java (held originally by Sun, now by Oracle)
- JavaScript means the programming language.
- ECMAScript is the name used by the language specification. Therefore, whenever referring to versions of the language, people say ECMAScript. The current version of JavaScript is ECMAScript 6 (also ES6)

How to include JavaScript in an HTML document

- **In the <head> of a page:** These scripts will be called when an event triggers them.
- **In the <body> of a page:** These scripts will run as the page loads.
- **In an external file:** If the link is placed inside the <head> element, the script is treated the same as when the script lives inside the head of the document waiting for an event to trigger it, whereas if it is placed in the <body> element it will act like a script in the body section and execute as the page loads.

- **How to Code a link to JavaScript external file**
- <script type="JavaScript" src="path_to_file.js"></script>

The JavaScript syntax

The basic syntax rules

- JavaScript is case-sensitive.
- Each JavaScript statement ends with a semicolon.
- JavaScript ignores extra whitespace within statements.
- How to split a statement over two or more lines

• Split a statement after:

  - an arithmetic or relational operator such as +, -, *, /, =, ==, >, or <
  - an opening brace ({), bracket ([ ), or parenthesis
  - a closing brace (})

• Do not split a statement after:

  - an identifier, a value, or the return keyword
  - a closing bracket ( ]) or closing parenthesis

- A JavaScript statement has a syntax that’s similar to the syntax of Java.
- Whitespace refers to the spaces, tab characters, and return characters in the code, and it is ignored by the compiler. As a result, you can use spaces, tab characters, and return characters to format your code so it’s easier to read.
- In some cases, JavaScript will try to correct what it thinks is a missing semicolon by adding a semicolon at the end of a split line. To prevent this, follow the guidelines above for splitting a statement.

Identifiers

- Identifiers can only contain letters, numbers, the underscore, and the dollar sign.
- Identifiers can’t start with a number.
- Identifiers are case-sensitive.
- Identifiers can be any length.
- Identifiers cannot be the same as reserved words.
- Avoid using global properties and methods as identifiers. If you use one of them, you won’t be able to use the global property or method with the same name.

Identifiers naming recommendations

- Use meaningful names for identifiers. That way, your identifiers aren’t likely to be reserved words or global properties.
- Be consistent: Either use camel casing (**taxRate**) or underscores (**tax_rate**) to identify the words within the variables in your scripts.
- If you’re using underscore notation, use lowercase for all letters.

- **Valid identifiers in JavaScript**

subtotal		
index_1	$	
taxRate	calculate_click	$log

- **Camel casing versus underscore notation**

taxRate	tax_rate
calculateClick	calculate_click
emailAddress	email_address
futureValue	future_value

How to use comments

- JavaScript provides two forms of comments, block comments and single-line comments.
- Comments are ignored when the JavaScript is executed.
- During testing, comments can be used to comment out (disable) portions of code that you don't want tested. Then, you can remove the comments when you're ready to test those portions.
- You can also use comments to save a portion of code in its original state while you make changes to a copy of that code.
- **The basic syntax rules for JavaScript comments**
  - Block comments begin with /* and end with */.
  - Single-line comments begin with two forward slashes (//) and continue to the end of the line.
- **Guidelines for using comments**
  - Use comments to describe portions of code that are hard to understand.
  - Use comments to disable portions of code that you don't want to test.
  - Don't use comments unnecessarily.

Data Types and Variables Declaration

- JavaScript provides for three primitive data types (number, string and Boolean).
- **The number data type**
  - A number value can be an integer, which is a whole number, or a decimal value, which can have one or more decimal positions to the right of the decimal point.
  - In JavaScript, decimal values are stored as floating-point numbers. In that format, a number consists of a positive or negative sign, one or more significant digits, an optional decimal point, optional decimal digits, and an optional exponent.
  - If a result is stored in a number data type that is larger or smaller than the data type can store, it will be stored as the value Infinity or -Infinity.
- **The string data type**
  - A string value is surrounded by double quotes or single quotes. The string must start and end with the same type of quotation mark.
  - An empty string is a string that contains no characters. It is entered by typing two quotation marks with nothing between them.
- **The Boolean data type**
  - A Boolean value can be true or false. Boolean values are often used in conditional statements.

- **Examples of number values**
  - ✓15 // an integer
  - ✓-21 // a negative integer
  - ✓21.5 // a decimal number
  - ✓-124.82 // a negative decimal value
  - ✓-3.7e-9 // scientific notation for -0.0000000037
- **Examples of string values**
  - ✓"JavaScript" // a string with double quotes
  - ✓'String Data' // a string with single quotes
  - ✓"" // an empty string
- **The two Boolean values**
  - ✓true
  - ✓false

Variables

- **How to declare and assign values to variables**
- A variable stores a value that can change as the program executes.
- To declare a variable, code the keyword **var** and a variable name.
- To declare more than one variable in a single statement, code var and the variable names separated by commas.
- To assign a value to a variable, use an *assignment* statement that consists of the variable name, an equals sign (the assignment operator), and a value. The type of a variable is determined by the value that's assigned to it.
- To assign values to two or more variables in a single statement, code the assignment statements separated by commas.
- The value that's assigned to a variable can be a literal value, another variable, or an expression like the arithmetic and string expressions.

- To code a string literal, you enclose it in single or double quotes. To code a numeric literal, you code an integer or decimal value that isn't enclosed in quotes.
- You can also declare a variable and assign a value to it in a single statement.
- Before you can use a variable, you must assign a value to it. A variable that hasn't been assigned a value has the value *undefined* and its type is *undefined*.
- To assign a Boolean value to a variable, you use the *true* and *false* keywords.
- **How to declare and assign a value to a variable in two statements**
  - **Syntax**
    - var variableName;
    - variableName = value;
  - **Examples**
    - var counter;

      counter = 1;
    - var sum, average;

      sum = 0, average = 0;

- How to declare and assign a value to a variable in one statement
- **Syntax**
  - ✓var variableName = value;
- **Examples**
  - ✓var count = 1;
  - ✓var subtotal = 74.95;
  - ✓var name = "Joseph";
  - ✓var email = "";
  - ✓var isValid = false;
  - ✓var total = subtotal;
  - ✓var x = 0, y = 0;

How to code arithmetic expressions

- An arithmetic expression consists of one or more operands that are operated upon by arithmetic operators.
- An arithmetic expression is evaluated based on the order of precedence of the operators. To override the order of precedence, you can use parentheses.
- **Arithmetic Operators**

  + - addition

  - - subtraction

  ***** - multiplication

  / - division

  % - modulus

  ++ - increment

  -- - decrement

- The order of precedence of arithmetic expressions

1	++	L - R
2	--	L - R
3	* / %	L - R
4	+ -	L - R

- **Examples**

5 + 7	12
5 - 12	-7
6 * 7	42
13 / 4	23.25
13 % 4	1
counter++	counter = counter + 1
counter--	counter = counter - 1
3 + 4 * 5	23
(3 + 4) * 5	35
13 % 4 + 9	10
13 % (4 + 9)	0

Assignment Operators

- The assignment operator (**=**) is used to write (assign) value to a variable.
- Besides the assignment operator, JavaScript provides for compound assignment operators.

  += Adds the result of the expression to the variable.

  -= Subtracts the result of the expression from the variable.

  ***=** Multiplies the variable value by the result of the expression.
- These operators are a shorthand way to code common assignment operations.
- JavaScript also offers **/=** and **%=** compound operators which are not often used.
- When you do some types of arithmetic operations with decimal values, the results aren't always precise, although they are extremely close. That's because decimal values are stored internally as *floating-point* numbers. The primary problem with this is that an equality comparison may not return true.

- Code that calculates sales tax
  - ✓var subtotal = 200;
  - ✓var taxPercent = .05;
  - ✓var taxAmount = subtotal * taxPercent;
  - ✓var total = subtotal + taxAmount;
- Code that calculates the perimeter of a rectangle
  - ✓var width = 4.25;
  - ✓var length = 8.5;
  - ✓var perimeter = (2 * width) + (2 * length)
- Statements that use the compound assignment operators
  - ✓var subtotal = 74.95;
    subtotal **+=** 20.00; // subtotal = 94.95
  - ✓var counter = 10;
    counter **-=** 1;
  - ✓var price = 100;
    price ***=** .8;

- Three ways to increment a variable named counter by 1
  - ✓var counter = 1;
  - ✓counter = counter + 1;
  - ✓counter += 1;
  - ✓counter++;
- A floating-point result that isn't precise
  - ✓var subtotal = 74.95; // subtotal = 74.95
  - ✓var salesTax = subtotal * .1; // salesTax = 7.495000000000001

How to concatenate strings

- To concatenate, or join, two or more strings, you can use the + or += operator.
- If you use a plus sign in an expression and both values are strings, JavaScript concatenates them. But if one value is a number and one is a string, JavaScript converts the number to a string and then concatenates the strings.
- Escape sequences can be used to insert special characters within a string like a return character that starts a new line or a quotation mark.
- Concatenation Operators
  - `+` - Concatenates two values.
  - `+=` - Adds the result of the expression to the end of the variable.
- Escape sequence characters
  - `\n` - new line
  - `\"` - double quotes
  - `\'` - single quote
  - `\t` - tab space
  - `\r` - carriage return

Introduction to objects, methods, and properties

- In simple terms, an **object** is a collection of *methods* and *properties*. A **method** performs a function or does an action. A **property** is a data item that relates to the object. When you develop JavaScript applications, you will often work with *objects*, *methods*, and *properties*.
- An object has methods that perform functions that are related to the object as well as properties that represent the data or attributes that are associated with the object.
- When you call a method, you may need to pass one or more parameters to it by coding them within the parentheses after the method name, separated by commas.
- The window object is the global object for JavaScript, and JavaScript lets you omit the object name and dot operator (period) when referring to the window object.
- **Common methods of the window object**

  alert(string) - Displays a dialog box that contains the string that's passed to it by the parameter along with an OK button.

  prompt(string, default) - Displays a dialog box that contains the string in the first parameter, the default value in the second parameter, an OK button, and a Cancel button. If the user enters a value and clicks OK, that value is returned as a string. If the user clicks Cancel, null is returned to indicate that the value is unknown.

  print() - Issues a print request for the current web page.

- One property of the window object

  location - The URL of the current web page

- The syntax for calling a method of an object

  objectName.methodName(parameters)

- A statement that calls the alert() method of the window object

  window.alert("Welcome to JavaScript window objects");

- A statement that calls the prompt() method with the object name omitted

  var dataEntry = prompt("This is a test of the prompt method", 100);

- The syntax for accessing a property of an object

  objectName.propertyName

- A statement that displays the location property of the window object

  alert(window.location); // Displays the URL of the current page

- **Other methods of window objects**

  parseInt(string) - Converts the string that's passed to it to an integer data type and returns that value. If it can't convert the string to an integer, it returns **NaN**.

  parseFloat(string) - Converts the string that's passed to it to a decimal data type and returns that value. If it can't convert the string to a decimal value, it returns **NaN**.

- **Examples**

```
var entryA = prompt("Enter any value", 12345.6789);
alert(entryA);
entryA = parseInt(entryA);
alert(entryA);

var entryB = prompt("Enter any value", 12345.6789);
alert(parseFloat(entryB));

var entryC = prompt("Enter any value", "Hello");
alert(parseFloat(entryC));
```