Memo	**IMED 2351**
Subject	JavaScript Basics, Strings, Variables, Data types, Constants, Comments Operators

Today we will cover the following:

- **Part I**
  - Overview
  - Syntax
  - Comments
  - Variables
  - document.write(), strings
    - single quotes
    - double quotes
  - Escape Characters
- **Part II**
  - Operators
    - Assignment
    - Concatenation
    - Arithmetic
    - Comparison
    - Logical
  - Operator Precedence
  - Incrementing & Decrementing

<u>**JavaScript Basics**</u>

JavaScript scripts start with <script> and always end with </script>

Where the JavaScript code lives on a page depends on the circumstance.

It can live at the top of the page in the <head> tag while performing a specific action:

```
<html>
<head>
<script>
        (JavaScript code here)
</script>
</head>
<body>
        (HTML code)
</body>
</html>
```

Or

It can live within the main HTML page itself, performing specific action(s) related to changing or interacting with the HTML

```
<html>
<body>
        (HTML code)
        <script>
                (JavaScript code)
        </script>
</body>
</html>
```

Or

You can have a page that is completely JavaScript. This is usually a file with the extension .js
To include the JavaScript file from within an html document use the following syntax:

```
<script type="text/javascript" src="myfile.js"></script>
```

Most JavaScript statements end with a ; (semi-colon)

When outputting some text or doing a calculation or assigning a value to something you will need a ; at the end. A statement is programming code that performs a task.

Comments

There are 2 ways to comment code in JavaScript.

//

/* … */

The comment tags in JavaScript work similar to the HTML comment tags in that the information is not displayed to the end user. The comments are also available in the View Source of an HTML page.

Comments are primarily used to help organize your code and explain any complicated sections of your page.

Comments can also be used to hide from the interpreter any JavaScript code that you do not want to run because you want to use it later.

Example:

```
<script>
// Comment code here for single line

/*
	Comment code
	On multiple lines
*/
<?script>
```

<u>document.write()</u>

- used to display the results of a php script within a web page to the client (web browser)
- will return/display a text string contained in either
    - double “
    - single ‘

document.write(“explore America!”);

document.write(‘explore America!’);

- Can embed HTML within document.write() and it will be rendered by the browser

<u>Variables</u>

- a variable name can only contain alpha-numeric characters and underscores (a-Z, 0-9, and _)
- a variable **name** can not contain spaces
- a variable name is case-sensitive
- I suggest naming all variables in lower-case or keep good track of them if they are mixed case.

JavaScript variables are loosely typed. Meaning the type of variable (string/text or number) is determined when the variable is assigned a value.

var myname = “thomas”;
    (this variable becomes a string)

var mycost = 3.50;
    (this variable becomes a floating point number—number with decimal)

var mycount = 50;
    (this variable becomes an integer or whole number—number without decimal)

- before you can use a variable in code → it must be “declared” or “initialized”
- to “declare’ a variable → you simply use the **var** keyword
- in JavaScript you can assign a value and initialize at the same time by giving it a value
    - *var variablename = value;*

You can display a variable using document.write()

document.write(mymoney);

You can combine with regular text using the document.write()

document.write(‘<br>The bank balance is ‘ + $mymoney);

Escape Characters

- Tells the JavaScript interpreter that the **character** that follows the **escape sequence** has a special purpose
- An **escape sequence** in JavaScript is the \ character
- Can be used within ‘ or “
- Used to:
    - Prevent a JavaScript error
    - Display text that has a special meaning to JavaScript as the meaning to your output

Escape Characters	
\\	backslash
\r	carriage return
\t	horizontal tab
\n	new line
\”	double quote
\’	single quote

Example

```
<script>
        // This will allow the use of “ within existing double quotes
        var myvar = “This is an \”escaped\” string”;
        document.write(“<br />” + myvar);

        // This allows you to display a Windows directory name
        // which would normally be a problem
        document.write(“c:\\windows\\system32\\myfile.txt”);
</script>
```

Data Types

Both variables and constants can have certain types of data stored within them. These are known as “data types”

They include:

- Integer Numbers
    - Positive or Negative number
    - No decimal places
- Floating Point Numbers
    - Number that contains a decimal place
    - Can be negative or positive
    - Can be exponential
- Boolean
    - Is either True or False
- String
    - Any alphanumeric text value
    - Surrounded by quotes (either ‘ or “) when assigned to a variable
- NULL
    - An empty value
- Undefined
    - An object that does not exist
- isNAN
    - A value or variable that is not a number

JavaScript is a loosely typed language; meaning the value of a variable can simply be assigned any specific type of value, and you can change that value type at any time.

Part II

Operators

Operators are used in JavaScript to assign or manipulate values within the program. The different types of operators include:

Assignment:

- used for assigning a value to a variable
- most common assignment =
- other assignment operators are called "compound assignment" operators
    - they perform mathematical operations

Assignment Operators		
Operator	**Name**	**Description**
=	Assignment	assigns the value of the right operand to the left operand
+=	compound addition	Add the value of the right operand to the value of the left operand and assigns the new value to the left operand
-=	compound subtraction	subtracts the value of the right operand to the value of the left operand and assigns the new value to the left operand
*=	compound multiplication	multiplies the value of the right operand by the value of the left operand and assigns the new value to the left operand
/=	compound division	divides the value of the left operand by the value of the right operand and assigns the new value to the left operand
%=	compound modulus	divides the value of the left operand by the value of the right operand and assigns the remainder (modulus) to the left operand

Concatenation:

- used with strings to combine values together
- can combine two strings
- uses a + (plus)

Arithmetic:

- perform mathematical calculations
- addition, subtraction, multiplication, division
- Types include:
  - Binary
  - Unary

Arithmetic/Mathematical Operators	
-	Negation
*	Multiplication
/	Division
%	Modulus (division remainder)
+	Addition
-	Subtraction
+	String Concatenation
++	Increment
--	Decrement

Comparison:

- used to compare two operands together and determine how one compares to another
- always returns the Boolean value of true or false
- primarily used with control structures: **if - while - for**
- can use with numbers or strings

Comparison Operators	
==	Equality
!=	Inequality
<	Less Than
>	Greater Than
<=	Less than or equal to
>=	Greater than or equal to
===	Strict Equality/Identity
!==	Non-identity / strict not equal

Logical:

- used for comparing two Boolean operands for equality
- a Boolean value of true or false is always returned
- used with comparison operators

Logical Operators	
&&	And
||	Or
!	Not

Operator Precedence (Order of Operations):

- the order in which operations in an expression are evaluated
- the order can be left to right or right to left in the evaluation (depending on the operator)

Precedence of Relational and Logical Operators	
Operator	**Associativity**
! ++ -- @	Right to left
* / %	Left to right
+ -	Left to right
< <= > >=	Left to right
== != ===	Left to right
&&	Left to right
||	Left to right
= += -= *= /=	Right to left