JavaScript Basics

JavaScript

- It is a scripting language used to manipulate the contents of a Web page. It can be both HTML and CSS.
- Using JavaScript, it can animate, move, transition, and hide HTML elements.
- JavaScript is different from Java. Java is a compiled, object-oriented programming language known for its ability to run on any platform with a Java Virtual Machine installed, while JavaScript only runs in a browser.
- JavaScript's file extension is **.js**

Comment

- The comment in JavaScript prevents the execution of the program or explain the JavaScript Code.
- There are two (2) styles of comment:
  - The single line or end-of-line comment starts with two (2) forward slashes //.
  - The block or multi-line comment begins with /* and ends with */.

Locations

- JavaScript can be linked in a number of ways in an HTML file. It can be inline, embedded (internal), and external JavaScript.
- **Inline JavaScript**
  - The inline JavaScript refers to including or placing the JavaScript code directly within certain attributes of an HTML.

```
<button onClick="showText()">Show</button>
<button onClick="removeText()">Remove</button>
```

- **Embedded (Internal) JavaScript**
  - This refers to including or placing the JavaScript code inside the **<script>** element.

```
<script>
    function showText(){
        document.getElementById("sample").innerHTML = "Hello World!";
        console.log("test 123");
    }
    function removeText(){
        document.getElementById("sample").innerHTML = "";
    }
</script>
```

- **External JavaScript**
  - This refers to the external file of a JavaScript that is placed within the **<head>** element, similar to the case of external CSS files. It is also possible to include this external file inside the **<body>** element, and it is recommended to place it at the bottom.

```
<head>
    <title>My First JavaScript</title>
    <script type="text/JavaScript" src="js/sample.js">
    </script>
</head>
```

Variables

- Variables in JavaScript are dynamically typed. It means that the variable can be a string, an integer, and then later an object.
- In declaring a variable, the **var** keyword is usually followed by the desired variable name and a semicolon. It simplifies the variable declarations, and it doesn't need or require the common data types, such as int, char, double or float, and string, once the value is set as a string (**`var str = "hello world!";`**). It is already set as a string variable, and if the value is a number (**`var num = 5;`**), then it is set as an integer.

```
var str = "hello world";
var integerr = 3;
var dbl = 5.22;
```

Data Type

- In JavaScript, data type specifies which type of value a variable contains.
- There are six (6) primitive data types that are supported in JavaScript:
  - String – The string data type represents textual data.
  - Boolean – The Boolean data type contains a logical entity that may return a value of TRUE or FALSE.
  - Number – The number data type can be an integer or a decimal.
  - Null – This data type represents an unknown or missing value.
  - Undefined – It refers to a variable that has no value. The **undefined** keyword can be set in the variable.

JavaScript Output

- In JavaScript, there are different ways to display the data in the browser:
  - **innerHTML**
    - The innerHTML is used to display the output in an HTML element.

```
//innerHTML
document.getElementById("ans").innerHTML = result;
```

  - **document.write();**
    - This syntax is used to display the data in the HTML document.

```
document.write("Answer: "+ result);
```

  - **window.alert();**
    - The alert function shows a pop-up message to the browser.

```
//alert
window.alert("Answer: "+result);
```

  - **console.log();**
    - This function displays the output in a console. The console for JavaScript can be seen in the developer tools of a Web browser.
    - The console object refers to the console of developer tools in a Web browser, while the log method prints the data.

```
//console
console.log("Answer: "+result);
```

Control Flow

Control Flow Statement

- The control flow statement refers to the decisions or conditions that the executed program may depend on its output.
- This control statement contains the following:
  - **if** – the **if** statement returns true if the specified condition is true.

```
if (condition){
 //block of code to be executed if
 //the condition is true
}
```

- **if-else** – the **if-else** statement returns true if the condition is true; otherwise, it returns false.

```
if (condition){
 //code to be executed if the
 //condition is true
}else{
 //code to be executed if the
 //condition is false
}
```

- **else if** – the **else if** statement is used if the first condition is false; otherwise, the second condition is true and so on.

```
if (condition){
 //code to be executed if the
 //condition is true
}else if(condition){
 //code to be executed if the
 //the first condition is false and second condition
 //is true
}else{
 //code to be executed if first and second
 //condition is false.
}
```

- **switch** – The **switch** statement is the same as the **if-else** statement. It evaluates an expression and matches the expression's value but in the case label.

```
function chooseFruit(myColor) {
    switch (myColor)
    {
        case "Apple":
             console.log("Fruit: Apple");
             break;
        case "Orange":
             console.log("Fruit: Orange");
             break;
        case "Mango":
             console.log("Fruit: Mango");
             break;
        default:
             console.log("All of the above");
    }
}
```

- **try…catch** – The **try…catch** statement is used to block a statement(s) to try and respond with an exception.
  - This statement consists of the following:
    - **try** – The try block contains one (1) or more statements.
    - **catch** – The catch block contains a statement of what to do when the try block throws an exception.

```
try{
    //try the block of code
}catch(exception){
  //error handler
}
```

Operator

- It is used to assign values in a variable.
- The following types are operators in JavaScript:
  - **Assignment operator**
    - The assignment operator is used to assign the value to its left operand or variable.
    - The following are the common assignment operators in JavaScript:
      - **=**
        - Assigns the value to a variable
      - **+=**
        - Adds the value to a variable
      - **-=**
        - Subtracts the value from a variable
      - ***=**
        - Multiplies the value by a variable
      - **/=**
        - Divides the value by a variable
  - **Comparison operators**
    - The comparison operator compares the two (2) variables or operands.
    - The following comparison operators are:
      - **==**
        - Equal to the value of the variable or operand.
      - **===**
        - Equal value and equal type
      - **< , >**
        - Less than and greater than
      - **<= , >=**
        - Less than or equal to and greater than or equal to
      - **!=**
        - Not equal to the value of the variable or operand.
  - **Arithmetic operators**
    - The arithmetic operator performs addition, subtraction, multiplication, and division.
    - The following arithmetic operators are:
      - **Addition (+)**
        - This arithmetic operator adds two (2) operands or two (2) variables that contain number values.
      - **Subtraction (-)**
        - This arithmetic operator subtracts two (2) operands or two (2) variables that contain number values.
      - **Multiplication (*)**
        - This arithmetic operator multiplies two (2)operands or two (2) variables that contain number values.

- **Division (/)**
  - This arithmetic operator divides two (2) operands or two (2) variables that contain number values.
- **Modulus (%)**
  - The modulus operator gets the remainder of the division of the two (2) numbers.
- **Increment (++)**
  - This operator adds one (1) in the operand.
- **Decrement (--)**
  - This operator subtracts one (1) from the operand.

- **Logical operators**
  - The logical operator uses Boolean values.
  - There are three (3) logical operators, and they are the following:
    - **AND (&&)**
      - The AND operator returns true if both values are true.
    - **OR (||)**
      - The OR operator returns true if either *expression1* or *expression2* is true.
    - **NOT (!)**
      - The NOT operator will return true if the *expression1* is false.
- **Ternary operator**
  - The ternary operator is the same as the **`if-else`** statement that if the condition is true, then execute the statement; otherwise, return false.

```
(condition) ? value1 : value2;
```

```
<input id="test" type="text" value=""/><br>
<button onclick="show()">Click me</button>
<script>
     function show(){
         var fruit = document.getElementById("test").value;
         var analyzeFruit = (fruit == "Apple") ? "An apple":"Not an apple";
         console.log(analyzeFruit);
     }
</script>
```

Loops and Iterations

`for` Loop

- A **`for`** loop specifies the number of loops of an object.
- The **`for`** loop contains three (3) expressions:
  - *Initialization*
    - The first expression is known as the counter of the **`for`** loop.
  - *Condition*
    - The second expression is condition; it will check if the condition will return true or false.
  - *Increment/Decrement*
    - The third expression specifies the number of loops based on the condition; it may increment or decrement this loop.

```
for(initialization; condition; increment/decrement){
    //code to be executed
}
```

```
<h3 id="test"></h3>
<button onclick="show()">Click me</button>
<script>
    function show(){
        var val;
        for(var y = 0; y < 10; y++){
            console.log(y);
        }
    }
</script>
```

for in

- It is used to loop the properties of an object or an array.

```
for(variable in object){
}
```

```
<h3 id="test"></h3>
<button onclick="show()">Click me</button>
<script>
    function show(){
        var val="";
        var list = {fruit1:"Apple", fruit2:"Orange", fruit3:"Grapes"};
        for(var x in list){
            val += list[x];
        }
        document.getElementById("test").innerHTML = val;
    }
</script>
```

do-while

- It is used to loop the statement until it evaluates to false.
- The **do-while** will execute the block of code first before checking if the condition will evaluate to true.

```
<h3 id="test"></h3>
<button onclick="show()">Click me</button>
<script>
    function show(){
        var val = " ";
        var x = 0;
        do{
            val += "Count: " + x + " \n";
            x++;
        }while(x < 10);
        console.log(val);
        document.getElementById("test").innerHTML = val;
    }
</script>
```

while

- The **while** loop is the most basic loop. It only uses a condition. The **while** loop is like an **if** statement that repeatedly runs until a condition is satisfied.

```
<h3 id="test"></h3>
<button onclick="show()">Click me</button>
<script>
    function show(){
       var val = " ";
       var x = 0;
       while(x < 10){
          val += "Count: " + x + " \n";
          x++;
       }
       console.log(val);
       document.getElementById("test").innerHTML = val;
    }
</script>
```

label

- The **label** statement identifies a statement and uses a **break** or **continue** statement to indicate whether the executed code will interrupt a loop or continue its execution.

```
fruit:{
    val += list["fruit1"] + "<br>";
    break fruit;
    val += list["fruit2"] + "<br>";
}
```

```
fruit:{
    for(var x = 0; x < 4; x++){
       if(x == 1){
         continue;
       }
       val += list[x] + "<br>";
    }
}
```

break

- The break statement is used to terminate a loop, **switch**, and a **labeled** statement.

```
<script>
function show(){
    var val = " ";
    var list = {fruit1:" Apple", fruit2:" Orange"};
    fruit:{
        val += list["fruit1"] + "<br>";
        break fruit;
        val += list["fruit2"] + "<br>";
    }
    console.log(val);
    document.getElementById("test").innerHTML = val;
}
</script>
```

continue

- The continue statement is used to repeat a loop, condition, or a labeled statement.

```
<script>
function show(){
var val = " ";
var list = [" Apple"," Orange",“ Grapes","Mango"];
    fruit:{
       for(var x = 0; x < 4; x++){
          if(x == 1){
            continue;
          }
          val += list[x] + "<br>";
       }
    }
    console.log(val);
    document.getElementById("test").innerHTML = val;
}
</script>
```

Functions

Functions

- A function is a block of code or modular code in JavaScript that performs a specific task. The function is defined in the
  function keyword followed by the function name and parameters enclosed in parentheses.
- Parameters in a function are accessible within the body of the function.

```
function functionName(param1, param2…){
    //code to be executed……
}
```

- The function may also return a value once you declare the parameters.

```
<script>
//return
function sum(num1, num2){
   return num1 + num2;
}
function show(){
   document.getElementById("test").innerHTML = "Total: "
   + sum(4,5);
   console.log("Total: " + sum(4,5));
    }
</script>
```

REFERENCES:

1. Connolly, R. & Hoar, R. (2015). *Fundamentals of web development*. New Jersey: Pearson Education, Inc.
2. Lemay, L., Colburn, R., & Kyrnin, J. (2016). *Sams teach yourself html, CSS and JavaScript web publishing in one hour a day* (7th Ed.). New Jersey: Pearson Education, Inc.
3. Krause, J. (2016). *Introducing web development*. California: Apress Media, LLC.