10 JavaScript Iterative Controls

Lesson Introduction

Previous lesson, you have leaned the main JavaScript programming concepts. During this week you will learn about control loops in JavaScript. Here, you will be able to apply loops with conditions, steps and nesting structures. Further, you will learn about Arrays and use of arrays in iterative scripting.

Learning Outcomes:

After completion of this lesson, the learner will be able to apply iterative loop structures in JavaScript Programs with arrays both in internal and external formats.

This lesson enables you to

- Write programs with array data structures
- Write JavaScript with For and For in loops
- Write JavaScript with while loop
- Write JavaScript with Do While loop

Lesson Outline

- Arrays in JavaScript
  - Elements and Indexes
- Control Loops
  - For Loop structure
  - For in loop and usage
  - While Loop
  - Do while loop
- Break and Continue with loops
- Internal and External JavaScript

10.1 Arrays in JavaScript

JavaScript arrays are used to store multiple values in a single variable. An array is a special variable, which can hold more than one value at a time. If you have a list of items (a list of names, for example), storing the names in single variables could look like this:

```
var names = "Saman";
var names = "Amara";
var names = "Seetha";
```

However, if you want to step through the names (one by one) and find a specific one, and if you had not 3 names, but 300, the most efficient (memory efficient) solution is an array.

An array can hold many values under a single name (array variable), and you can access the values by referring to an index number.

(var *array_name* = [*item1*, *item2*, ...];)

```
var names = ["Saman", "Amara", "Seetha"];
```

Spaces and line breaks are not important. A declaration can span multiple lines:

```
var names = [
    "Saman",
    "Amara",
    "Seetha"];
```

Using the JavaScript Keyword new

The following example also creates an Array, and assigns values to it:

```
var cars = new Array("Saab", "Volvo", "BMW");
```

In the above example, we use Array () function to create an array. It returns an array object with given parameter list.

Example:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Arrays</h2>

<p id="demo"></p>

<script>
var names = ["Saman", "Amara", "Seetha"];
document.getElementById("demo").innerHTML = names;
</script>

</body>
</html>
```

Activity 10.1: Write above web script and observe the output. Then understand the behavior of an Array.

10.1.1 Array Elements and Indexes

Access the Elements of an Array

You can refer to an array element by calling the index number of the array. Normally, array element indexes starts from 0. The following statement accesses the value of the first element in names:

```
var name1 = names[0];
```

The following statement modifies the first element in names:

```
names[0] = "Geetha";
```

Example:

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Arrays</h2>
<p>JavaScript array elements are accesses using numeric indexes (starting from 0).</p>

<p id="demo"></p>

<script>
var names = ["Saman", "Amara", "Seetha"];
document.getElementById("demo").innerHTML = names[0];
</script>
</body>
</html>
```

Activity 10.2: Write above web script and observe the output. Then understand the behavior of an Array indexes.

Modify the above script to replace names list “Seetha” with “Geetha”. Then add two another names “Senhas” and “Vanuja”. Finally, print the new names on the web.

Hint: Use document.write (*ARRAY [x]* + “
”) method for print on the web. Here, + uses for string concatenation; “
” uses to create line break.

Upload your modified script to the moodle under the given link.

10.2 Control Loops

Here, we are going to learn about repetition of statements, which is flow control part in JavaScript. Looping Array Elements is the way to access the elements one by one. There are four loops enable access to the array elements.

1. **for** - loops through a block of code a number of times
2. **for/in** - loops through the properties of an object
3. **while** - loops through a block of code while a specified condition is true
4. **do/while** - also loops through a block of code while a specified condition is true

10.2.1For Loop structure

In here you will learn for loop properly. The best way to loop through an array, is using a "for" loop:

“for” Repetition Structure

Operation of “for” structure in a program:

```
A          B  E          D
for (initial value; condition; increment/decrement) {

        command 1;
        command 2;                    D
}
C
```

A. Program execution flows first into initial value, which is executed once to initialize the loop counter.

B. Condition is then evaluated; recall that it is a logical expression containing the loop counter;

C. If condition evaluates to false, then the commands between the {} are skipped and the program continues to execute after the loop;

D. If condition evaluates to true, then the commands between the {} are executed and increment/decrement is performed to modify the value of the counter;

E. Back to step B.

Form of execution of for loop

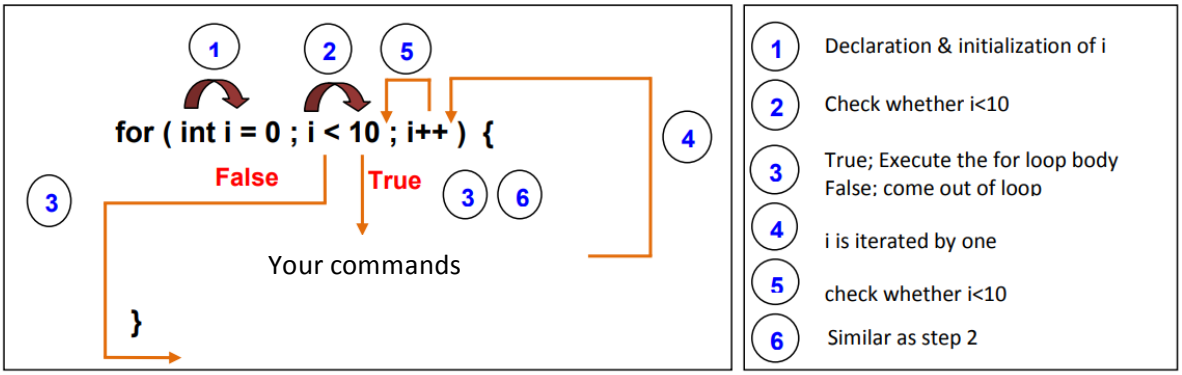

Figure 10.1: For loop illustration

10.2.2For in loop and usage

In the following code snippet it shows how to loop through an array, by using a "for" loop:

```
var fruits, text, fLen, i;

fruits = ["Banana", "Orange", "Apple", "Mango"];
fLen = fruits.length;
text = "<ul>";
for (i = 0; i < fLen; i++) {
    text += "<li>" + fruits[i] + "</li>";
}
```

There is one important thing you have to remember, that is the use of html tags inside JavaScript is to be done as text print into the document. In the above example snippet, <ul> and <li> tags are used as strings for printing.

In the following example, you can see the use of above text fro printing, then when you view it; you will see the interpretation of <li> and <ul> tags.

When you use the ***array_name.length*** for any array, it will return the size or number of elements in the array.

Example:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Arrays</h2>

<p>The best way to loop through an array is using a standard for loop:</p>

<p id="demo"></p>

<script>
var fruits, text, fLen, i;

fruits = ["Banana", "Orange", "Apple", "Mango"];
fLen = fruits.length;
text = "<ul>";
for (i = 0; i < fLen; i++) {
   text += "<li>" + fruits[i] + "</li>";
}
text += "</ul>";
document.getElementById("demo").innerHTML = text;
</script>

</body>
</html>
```

Activity 10.3: Write above web script and observe the output. Then understand the behavior of a for loop with array indexes.

Modify the above script to take five names from input prompt window and store inside an array. Finally, print the name list on the web.

Upload your script to the moodle under the given link.

10.2.3 Use of While Loop in JavaScript

Lets see, how the “while loop” iterate through a block of code as long as a specified condition is true. The following code snippet shows the syntax of a while loop. When the condition is true, code block inside {} will be executed repeatedly until it breaks (false).

```
while (condition) {
    code block to be executed
}
```

```
while (i < 10) {
    text += "The number is " + i;
    i++;
}
```

The following example shows the print of list of numbers through a while loop. Here the condition is given as (i<10). As I starts from 0, there are 10 iterations as I increments and break the condition.

Example:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript while</h2>

<p id="demo"></p>
<script>
var text = "";
var i = 0;
while (i < 10) {
  text += "<br>The number is " + i;
  i++;
}
document.getElementById("demo").innerHTML = text;
</script>
</body>
</html>
```

Activity 10.4: Write above web script and observe the output. Then understand the behavior of a while loop with numeric index printing.

Modify the above script to take five names from input prompt window and store inside an array. Finally, print the name list on the web. Here, for input acquisition and printing, use while loops.

Upload your script to the moodle under the given link.

10.2.4 Do while loop

The "do/while" loop also operates similar to the "while" loop. This loop will execute the code block once, before checking if the condition is true, this happens because of the condition is evaluated at the end of the code block. The loop will repeat the code block as long as the condition is true.
The following code snippet shows syntax of do .. while loop. The condition is placed at the end.

```
do {
    code block to be executed
}
while (condition);
```

```
do {
    text += "The number is " + i;
    i++;
}
while (i < 10);
```

The following example shows the use of do .. while loop to print 10 numbers in a sequence. Text variable concatenate the new line of string in each iteration.

Example:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript do ... while</h2>

<p id="demo"></p>

<script>
var text = ""
var i = 0;

do {
   text += "<br>The number is " + i;
   i++;
}
while (i < 10);

document.getElementById("demo").innerHTML = text;
</script>

</body>
</html>
```

Activity 10.5: Write above web script and observe the output. Then understand the behavior of a do .. while loop with numeric index printing.

Modify the above script to take five names from input prompt window and store inside an array. Finally, print the name list on the web. Here, for input acquisition and printing, use do .. while loops. Compare the difference of while loop and do .. while loop.

Upload your script to the moodle under the given link.

10.3 Break and Continue with loops

10.3.1 The Break Statement

You have already seen the break statement in switch .. case conditions. It was used to "exit " of a switch() statement. The break statement can also be used to exit from a loop. The break statement breaks from the immediate loop and continues executing the code after the loop (if any):

```
for (i = 0; i < 10; i++) {
    if (i == 3) { break; }
    text += "The number is " + i + "<br>";
}
```

In the above code snippet, when i ==3, it break out from the for loop.

The following example shows the use of break statement inside a loop. Same way, you may use it in other loop bodies as well.

Example

```
<!DOCTYPE html>
<html>
<body>
<p>A loop with a break.</p>
<p id="demo"></p>
<script>
var text = "";
var i;
for (i = 0; i < 10; i++) {
   if (i === 3) { break; }
   text += "The number is " + i + "<br>";
}
document.getElementById("demo").innerHTML = text;
</script>
</body>
</html>
```

Activity 10.6: Write above web script and observe the output. Then understand the behavior of break statement inside a loop.

10.3.2 The Continue Statement

The continue statement helps you to ignore one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

This example skips the value of i== 3:

```
for (i = 0; i < 10; i++) {
    if (i === 3) { continue; }
    text += "The number is " + i + "<br>";
}
```

Example:

```
<!DOCTYPE html>
<html>
<body>
<p>A loop which will skip the step where i = 3.</p>
<p id="demo"></p>

<script>
var text = "";
var i;
for (i = 0; i < 10; i++) {
   if (i === 3) { continue; }
   text += "The number is " + i + "<br>";
}
document.getElementById("demo").innerHTML = text;
</script>

</body>
</html>
```

Activity 10.7: Write above web script and observe the output. Then understand the behavior of continue statement inside a loop.

10.4 Internal and External JavaScript

10.4.1Internal JavaScript

In order for JavaScript code to work its program, it must be included in the proper location on a particular web content as the page is loaded. Internal JavaScript code is code that's placed anywhere within the web page between the HTML tags. Many web developers choose to place their JavaScript code before the html tag. Some situations, it is placed at <head> section.

10.4.2 External JavaScript

JavaScript code placed in a file separate from the HTML code is called external JavaScript. External JavaScript code is written in the same way as internal JavaScript is written but without <script> tag. The file should save with ".js" extension.
Then the written external script can be called from any web page to execute it. The web browser must know that it needs to load an external code. The web page must have the following HTML tags referencing the script.
The src tells the web server where to locate and load the JavaScript code.
The following example shows the use of external .js file.

Write the following code and save the document with an extension of .js .

```
document.write("This script is an external JavaScript")
```

Save the external file as xxx.js.
Now you can call this script, using the "src" attribute, from any of your pages

```
<html>
<head> </head>
<body>
 <script src="xxx.js"> </script>
</body>
</html>
```

Any of the previously written codes can be written as external js files and call them

10.4.3 Advantages of Internal JavaScript

If the number of lines of JavaScript is relatively small, a web page with internal JavaScript loads faster than pages that must reference external code. This is because, as the web browser loads the page and encounters the reference to the external code, it must make a separate request to the web server to fetch the code.

10.4.4 Advantages of External JavaScript

If identical code is to be used on several pages of a website, external JavaScript files are beneficial. These js files improve the reusability of codes. You only need to include a reference to the external code in those pages that actually require that code. If the JavaScript code changes, only one file has to be edited; changes are instantly available to all pages referred to the code. Storing JavaScript in external files also makes it easier to maintain websites.
Finally, if a visitor moves to another page containing the same external JavaScript references, the second page will load faster, because the user's browser will have cached or stored the JavaScript file locally.

This external JavaScript concept is popular as js libraries where many usable codes were written and distributed to reuse them where coding time can be reduce in greater extent.

Activity 10.8: Write an external JavaScript to perform, addition, subtraction, multiplication and division functions for two numerical inputs (parameters) and a function to calculate the sum of all the numerical values up to given number (summation of all positive integers up to the given number). Then call the above functions by using a web interface. Use input form to perform those operations.

Upload your exercise to the moodle on the given link.