JAVASCRIPT - LOOP CONTROL

JavaScript provides full control to handle loops and switch statements. There may be a situation when you need to come out of a loop without reaching at its bottom. There may also be a situation when you want to skip a part of your code block and start the next iteration of the look.

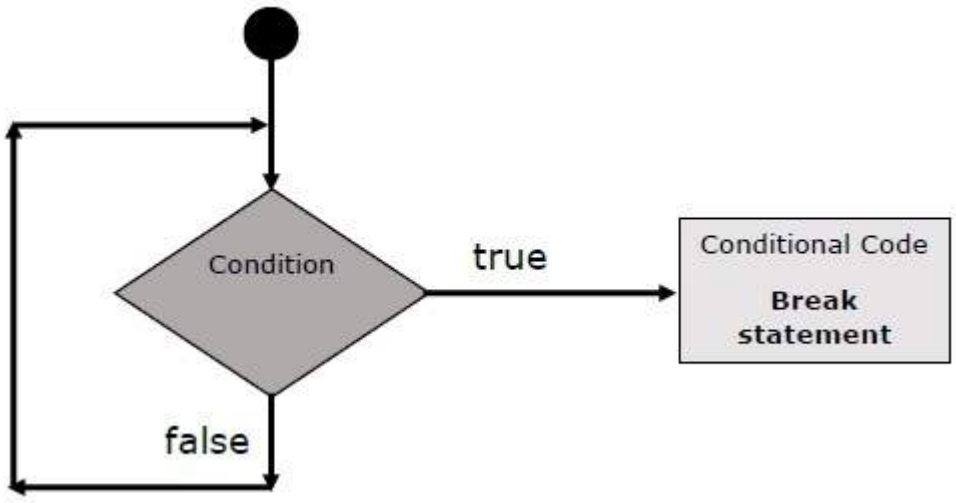
To handle all such situations, JavaScript provides **break** and **continue** statements. These statements are used to immediately come out of any loop or to start the next iteration of any loop respectively.

The break Statement

The **break** statement, which was briefly introduced with the *switch* statement, is used to exit a loop early, breaking out of the enclosing curly braces.

Flow Chart

The flow chart of a break statement would look as follows −



Example

The following example illustrates the use of a **break** statement with a while loop. Notice how the loop breaks out early once **x** reaches 5 and reaches to **document.write . .** statement just below to the closing curly brace −

```
<html>
   <body>

      <script type="text/javascript">
         <!--
         var x = 1;
         document.write("Entering the loop<br /> ");

         while (x < 20)
         {
            if (x == 5){
               break; // breaks out of loop completely
            }
            x = x + 1;
            document.write( x + "<br />");
         }

         document.write("Exiting the loop!<br /> ");
         //-->
```

```
      </script>

      <p>Set the variable to different value and then try...</p>
   </body>
</html>
```

Output

```
Entering the loop
2
3
4
5
Exiting the loop!
Set the variable to different value and then try...
```

We already have seen the usage of **break** statement inside **a switch** statement.

The continue Statement

The **continue** statement tells the interpreter to immediately start the next iteration of the loop and skip the remaining code block. When a **continue** statement is encountered, the program flow moves to the loop check expression immediately and if the condition remains true, then it starts the next iteration, otherwise the control comes out of the loop.

Example

This example illustrates the use of a **continue** statement with a while loop. Notice how the **continue** statement is used to skip printing when the index held in variable **x** reaches 5 −

```
<html>
   <body>

      <script type="text/javascript">
         <!--
            var x = 1;
            document.write("Entering the loop<br /> ");

            while (x < 10)
            {
               x = x + 1;

               if (x == 5){
                  continue; // skill rest of the loop body
               }
               document.write( x + "<br />");
            }

            document.write("Exiting the loop!<br /> ");
         //-->
      </script>

      <p>Set the variable to different value and then try...</p>
   </body>
</html>
```

Output

```
Entering the loop
2
3
4
6
7
8
```

```
9
10
Exiting the loop!
```

Using Labels to Control the Flow

Starting from JavaScript 1.2, a label can be used with **break** and **continue** to control the flow more precisely. A **label** is simply an identifier followed by a colon : that is applied to a statement or a block of code. We will see two different examples to understand how to use labels with break and continue.

Note − Line breaks are not allowed between the **'continue'** or **'break'** statement and its label name. Also, there should not be any other statement in between a label name and associated loop.

Try the following two examples for a better understanding of Labels.

Example 1

The following example shows how to implement Label with a break statement.

```
<html>
   <body>

      <script type="text/javascript">
         <!--
            document.write("Entering the loop!<br /> ");
            outerloop: // This is the label name

            for (var i = 0; i < 5; i++)
            {
               document.write("Outerloop: " + i + "<br />");
               innerloop:
               for (var j = 0; j < 5; j++)
               {
                  if (j > 3 ) break ; // Quit the innermost loop
                  if (i == 2) break innerloop; // Do the same thing
                  if (i == 4) break outerloop; // Quit the outer loop
                  document.write("Innerloop: " + j + " <br />");
               }
            }

            document.write("Exiting the loop!<br /> ");
         //-->
      </script>

   </body>
</html>
```

Output

```
Entering the loop!
Outerloop: 0
Innerloop: 0
Innerloop: 1
Innerloop: 2
Innerloop: 3
Outerloop: 1
Innerloop: 0
Innerloop: 1
Innerloop: 2
Innerloop: 3
Outerloop: 2
Outerloop: 3
Innerloop: 0
Innerloop: 1
```

```
Innerloop: 2
Innerloop: 3
Outerloop: 4
Exiting the loop!
```

Example 2

```
<html>
   <body>

      <script type="text/javascript">
         <!--
         document.write("Entering the loop!<br /> ");
         outerloop: // This is the label name

         for (var i = 0; i < 3; i++)
         {
            document.write("Outerloop: " + i + "<br />");
            for (var j = 0; j < 5; j++)
            {
               if (j == 3){
                  continue outerloop;
               }
               document.write("Innerloop: " + j + "<br />");
            }
         }

         document.write("Exiting the loop!<br /> ");
         //-->
      </script>

   </body>
</html>
```

Output

```
Entering the loop!
Outerloop: 0
Innerloop: 0
Innerloop: 1
Innerloop: 2
Outerloop: 1
Innerloop: 0
Innerloop: 1
Innerloop: 2
Outerloop: 2
Innerloop: 0
Innerloop: 1
Innerloop: 2
Exiting the loop!
```