Control Structure

Control structure actually controls the flow of execution of a program. Following are the several control structure supported by javascript.

JavaScript has number of statements that control flow of the program. They are:

- Conditionals (if-else, switch) that perform different actions depending on the value of an expression,
- Loops (while, do-while, for, for-in, for-of), that execute other statements repetitively,
- Jumps (break, continue, labeled statement) that cause a jump to another part of the program.

1. CONDITIONAL STATEMENTS

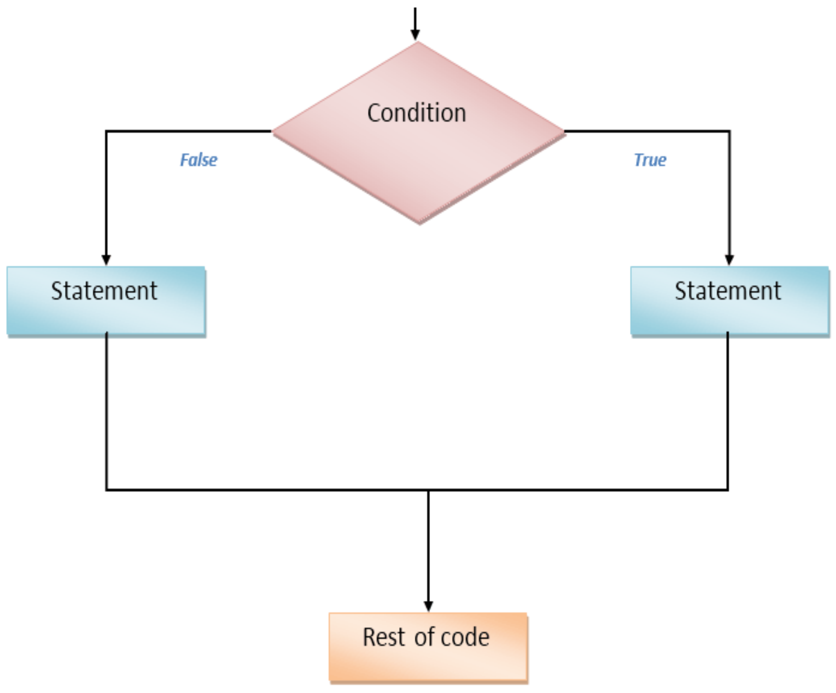
Very often when you write code, you want to perform different actions for different decisions.

You can use conditional statements in your code to do this.

In JavaScript we have the following conditional statements:

- Use if to specify a block of code to be executed, if a specified condition is true
- Use else to specify a block of code to be executed, if the same condition is false
- Use else if to specify a new condition to test, if the first condition is false
- Use switch to specify many alternative blocks of code to be executed

Conditional statements are used to decide the flow of execution based on different conditions. If a condition is true, you can perform one action and if the condition is false, you can perform another action.



Different Types of Conditional Statements

There are mainly three types of conditional statements in JavaScript.

1. If statement
2. If…Else statement

3. If…Else If…Else statement

If statement

The if statement is the fundamental control statement that allows JavaScript to make decisions and execute statements conditionally. The if statement allows to make decision and to execute statement conditionally. If the condition evaluates to true, statement is executed, otherwise it is not.

Syntax:

```
if (condition)

{

lines of code to be executed if condition is true

}
```

You can use If statement if you want to check only a specific condition.

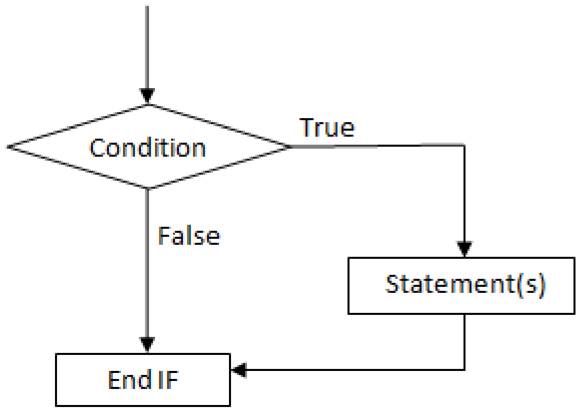
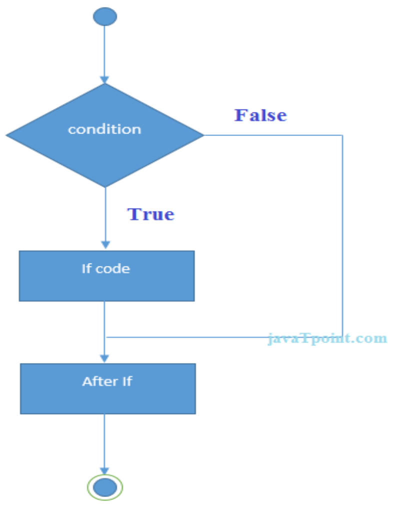


fig: Flowchart for if statement



Example:

```
<html>
<body>
<script type="text/javascript">
  var age = 20;
      if( age > 18 )
            {
                  document.write("<b>Qualifies for driving</b>");
             }
 </script>
</body>
</html>
```

Example 2:

```
<html>
    <body>
    <script type="text/javascript">
```

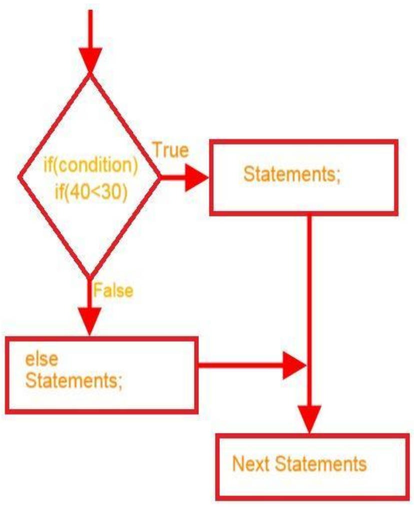
```
        var num = prompt("Enter Number");
        if (num > 0)
        {
            alert("Given number is Positive!!!");
        }
    </script>
    </body>
</html>
```

If…Else statement

The JavaScript If Else Statement is an extension of the If statement. We already saw the Js If Statement, it only executes the statements when the given condition is true. If the condition is false, it will not run any statements.

If – Else is a two-way decision statement.
It is used to make decisions and execute statements conditionally.

Flow Diagram of If – Else Statement



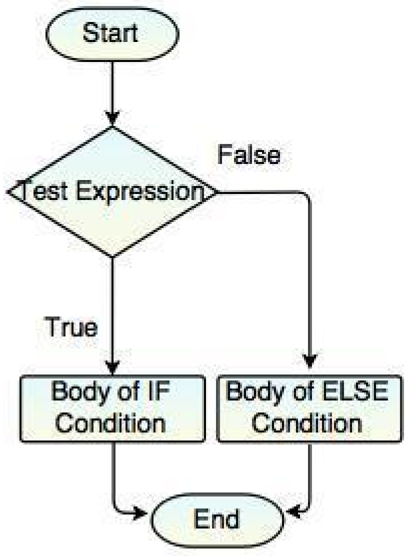


Fig. Flow Diagram of IF - ELSE Statement

The if..else statement evaluates the **condition** inside the parenthesis.
If the condition is evaluated to true,

1. the code inside the body of if is executed
2. the code inside the body of else is skipped from execution

If the condition is evaluated to false,

1. the code inside the body of else is executed
2. the code inside the body of if is skipped from execution

Syntax:

```
if (condition)

{

lines of code to be executed if the condition is true
}

else

{

lines of code to be executed if the condition is false

}
```

You can use If….Else statement if you have to check two conditions and execute a different set of codes.

Example:

```
<html>
<body>


<script>
  var greeting;
  if (time < 20)
       {
                  greeting = "Good day";
       }
else
      {
                   greeting = "Good evening";
       }
  document.write(greeting)
}
</script>
</body>
</html>
```

Example 2:

```
<html>
<body>
<script type="text/javascript">
  var marks = 60;
  if( marks >= 50 )
  {
    document.write("<b> Congratulations </b>");
    document.write("<br\> You Passed the subject" );
  }
  else
  {
    document.write("<b> You Failed </b>"); //s3
    document.write("<br\> Better Luck Next Time" ); //s4
  }
</script>
</body>
</html>
```

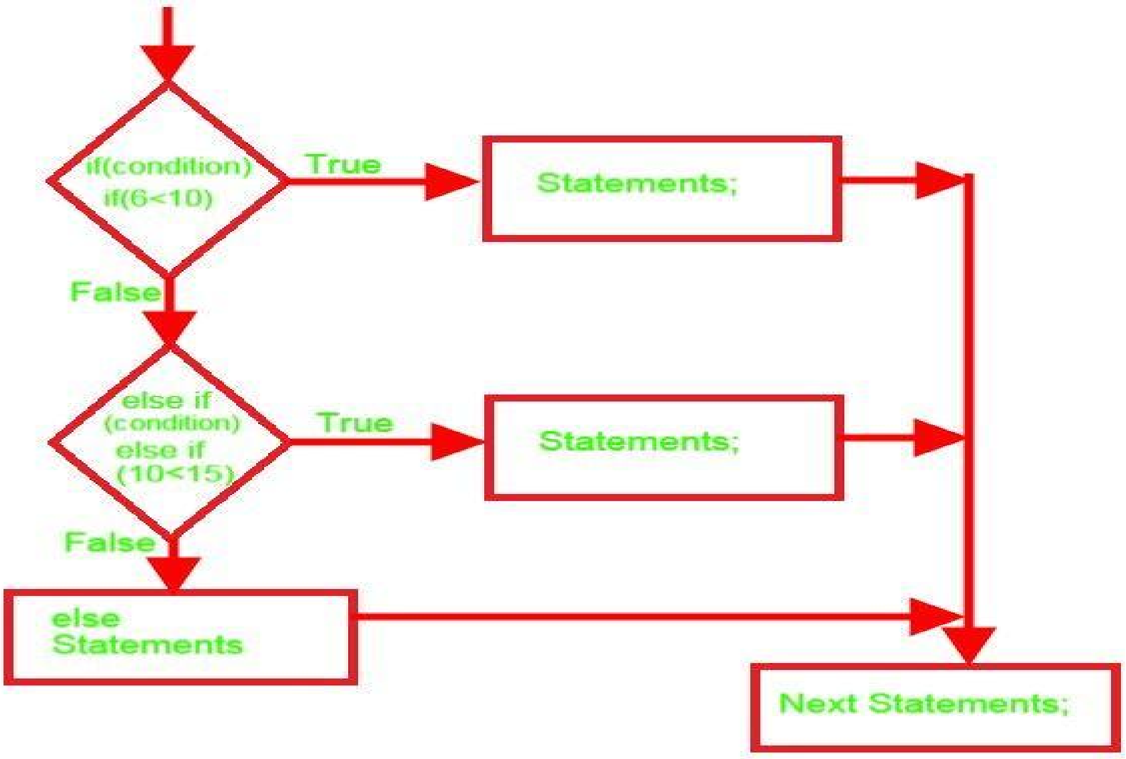
If…Else If…Else statement

It evaluates the content only if expression is true from several expressions. The signature of JavaScript if else if statement is given below.

Syntax:

```
if (condition1)

{

lines of code to be executed if condition1 is true

}

else if(condition2)

{

lines of code to be executed if condition2 is true

}

else

{

lines of code to be executed if condition1 is false and condition2 is false

}
```

You can use If….Else If….Else statement if you want to check more than two conditions.



Example:

<html>

```
<body>
<script type="text/javascript">

Var year = prompt('In which year was the ECMAScript-2015 specification published?', '')

if (year < 2015)
 {
 alert( 'Too early...' )
  }
 else if (year > 2015)
{
 alert( 'Too late' )
}
else
{
 alert( 'Exactly!' )
 }
</script>
</body>
</html>
```

Switch Statement

The switch statement is similar to the if-else statement. If all of the branches depend on the value of the same expression, the switch control structure can be used instead of the if-else.

The basic syntax of the switch statement is to give an expression to evaluate and several different statements to execute based on the value of the expression. The interpreter checks each case against the value of the expression until a match is found. If nothing matches, a default condition will be used.

Syntax

```
switch (expression)
 {
   case condition 1:
                        statement(s)
                        break;
   case condition 2:
                        statement(s)
                        break;
   ...
   case condition n:
                        statement(s)
                         break;
   default:
                        statement(s)
}
```

When the switch statement is executed, the program attempts to match the value of expression to a case valuel. If the value of expression is equal to a case value, the program executes the associated statements. If matching is not found, the program looks for the optional default clause. If it is found, associated statements are executed, if not the program skips all statements within the switch statement.

The optional break keyword associated with each case ensures that the program breaks out of the switch statement once the matched statement is executed. If the break is omitted all subsequent statements will also be executed.

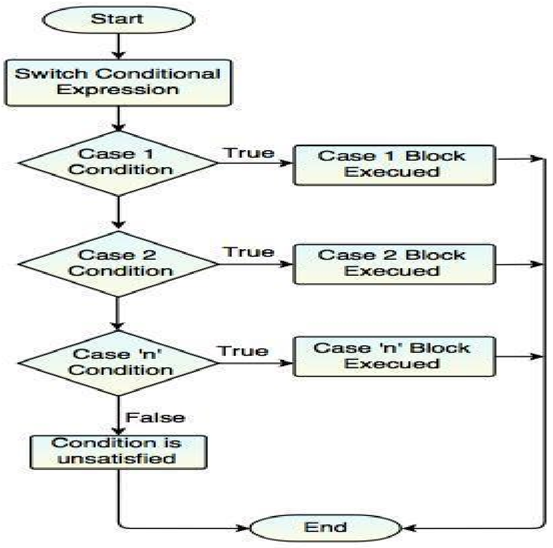


Fig. Flow Diagram of Switch Statement

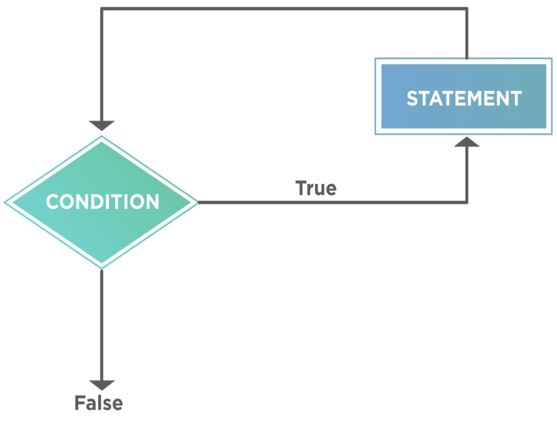
Example:

```
<html>
<body>
<script type="text/javascript">

      var grade='A';
      document.write("Entering switch block<br/>");
      switch (grade) {
         case 'A':
               document.write("Good job<br/>");
                break;
         case 'B':
               document.write("Pretty good<br/>");
               break;
         case 'C':
               document.write("Passed<br/>");
                break;
         case 'D':
                document.write("Not so good<br/>");
                break;
         case 'F':
                document.write("Failed<br/>");
                 break;
         default:
               document.write("Unknown grade<br/>")
      }
      document.write("Exiting switch block");

</script>
</body>
</html>
```

ITERATIONAL STATEMENTS

Loops are one of the most fundamental concepts available in all ***programming languages***. The loop will execute the set of code repeatedly until the given condition is satisfied. Loop will ask a question; if the given answer is satisfied, then it will perform some actions, again it will ask a question, and this repeats

until no further action required. Each time the question is asked, it is called an ***iteration***. We can depict it with the help of the following diagram:
A programmer who needs to execute the same lines of code, again and again, can use loops. The basic idea behind a loop is to automate the repetitive tasks within a program to save time and effort.



In computer programming, a loop is a sequence of instructions that is repeated until a certain condition is reached.

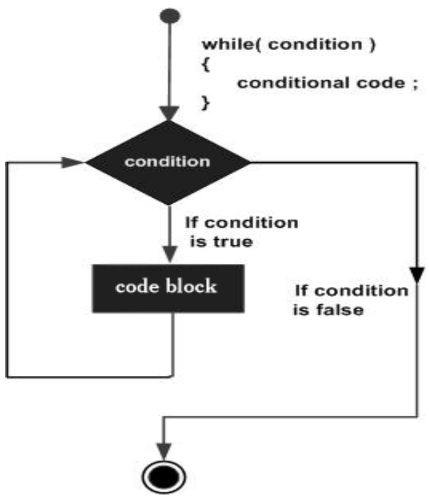
- An operation is done, such as getting an item of data and changing it, and then some condition is checked such as whether a counter has reached a prescribed number.
- **Counter not Reached:** If the counter has not reached the desired number, the next instruction in the sequence returns to the first instruction in the sequence and repeat it.
- **Counter reached:** If the condition has been reached, the next instruction "falls through" to the next sequential instruction or branches outside the loop.

There are mainly two types of loops:

1. **Entry Controlled loops**: In this type of loops the test condition is tested before entering the loop body. **For Loop** and **While Loop** are entry controlled loops.
2. **Exit Controlled Loops**: In this type of loops the test condition is tested or evaluated at the end of loop body. Therefore, the loop body will execute atleast once, irrespective of whether the test condition is true or false. **do – while loop** is exit controlled loop.

a) The While Loop

A while loop is a control flow statement that allows code to be executed repeatedly based on a given Boolean condition. The while loop can be thought of as a repeating if statement.
The purpose of a while loop is to execute a statement or code block repeatedly as long as an expression is true. Once the expression becomes false, the loop terminates.



```
while( condition )
{
   conditional code ;
}
```

- While loop starts with the checking of condition. If it evaluated to true, then the loop body statements are executed otherwise first statement following the loop is executed. For this reason it is also called **Entry control loop**
- Once the condition is evaluated to true, the statements in the loop body are executed. Normally the statements contain an update value for the variable being processed for the next iteration.
- When the condition becomes false, the loop terminates which marks the end of its life cycle.

Syntax :

```
while (condition)
{
        Statement(s) to be executed if expression is true
}
```

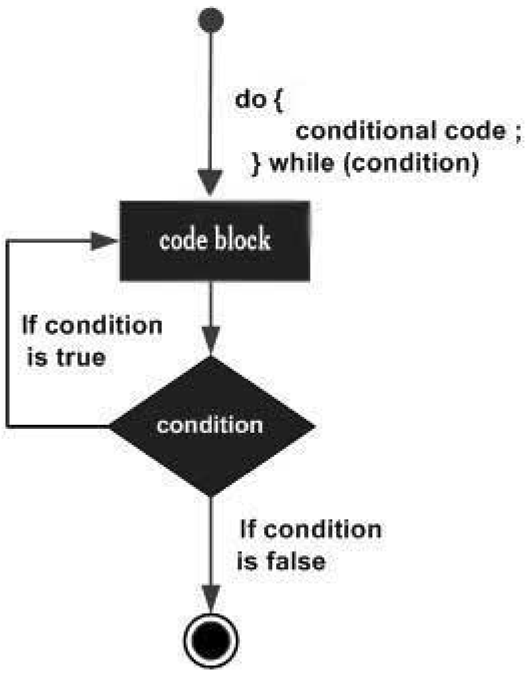
Example:

```
<html>
  <body>

    <script type = "text/javascript">
        var count = 0;
        document.write("Starting Loop ");

        while (count < 10)
          {
                  document.write("Current Count : " + count + "<br />");
                  count++;
          }
       document.write("Loop stopped!");
    </script>

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

The Do…..While Loop

The **do...while** loop is similar to the **while** loop except that the condition check happens at the end of the loop. This means that the loop will always be executed at least once, even if the condition is **false**.

1. First, we initialize our variables. Next, it will enter into the Do While loop in JavaScript.
2. It will execute the group of statements inside the loop.
3. Next, we have to use JavaScript Increment and Decrement operators inside the loop to Increment and Decrement value.
4. Now it will check the condition. If the condition is True, then the statement inside the JavaScript Do while loop will execute again. It will continue the process as long as the condition is True.
5. If the condition is False, it will exit from the loop



- do while loop starts with the execution of the statement(s). There is no checking of any condition for the first time.
- After the execution of the statements, and update of the variable value, the condition is checked for true or false value. If it is evaluated to true, next iteration of loop starts.
- When the condition becomes false, the loop terminates which marks the end of its life cycle.
- It is important to note that the do-while loop will execute its statements atleast once before any condition is checked, and therefore is an example of exit control loop.

Syntax:

```
do
{
   statements..
}while (condition);
```

Example:

```
<html>
   <body>
      <script type = "text/javascript">
         var count = 0;

         document.write("Starting Loop" + "<br />");

         do
         {
            document.write("Current Count : " + count + "<br />");
            count++;
         } while (count < 5);

         document.write ("Loop stopped!");
      </script>
      <p>Set the variable to different value and then try...</p>
   </body>
</html>
```

For Loop

For loop provides a concise way of writing the loop structure. Unlike a while loop, a for statement consumes the initialization, condition and increment/decrement in one line thereby providing a shorter, easy to debug structure of looping.

The '**for**' loop is the most compact form of looping. It includes the following three important parts −

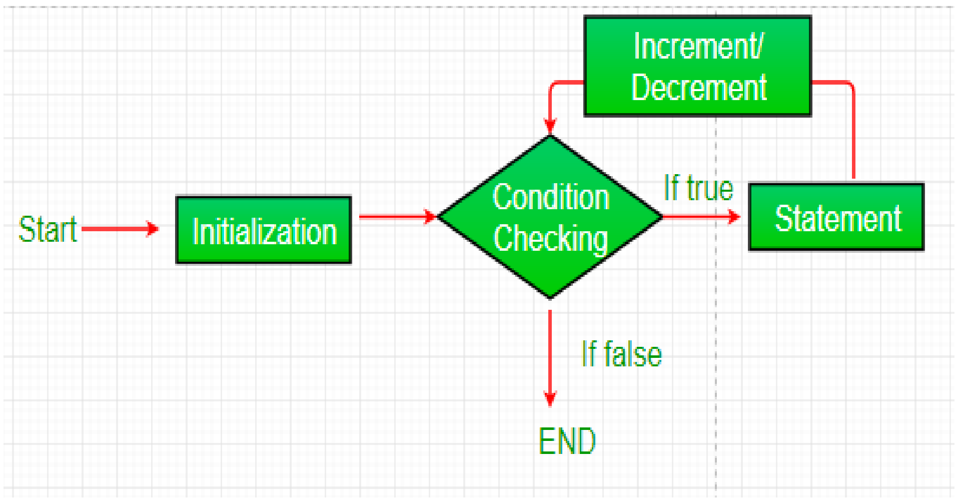
- The **loop initialization** where we initialize our counter to a starting value. The initialization statement is executed before the loop begins.
- The **test statement** which will test if a given condition is true or not. If the condition is true, then the code given inside the loop will be executed, otherwise the control will come out of the loop.
- The **iteration statement** where you can increase or decrease your counter.

Syntax:

```
for (Initialization condition; test condition; Increment/Decrement)
{
Body of loop
}
```

The Workflow of For Loop in JavaScript

1. **Initialization Condition:** The condition states the start of the for a loop. The variable can be initialized at a loop, or the variable can be declared separately.
2. **Testing Condition:** For loop is an Entry Control Loop, the condition is checked before the program executes. It also tests the exit condition of the loop.
3. **Statement Execution:** Only if the test condition is true then the body of the loop is executed.
4. **Increment/ Decrement:** For every cycle, after the loop executes, the control goes to the increment statement. Here it Increments or decrement the control variables.
5. **Loop Termination:** The loop terminates as soon as the condition becomes false.



Example:

```
<html>

  <body>

    Demonstrating For loop statement

    </br>

    <script type = "text/javascript">

        var n = 5;

        var i=0;

        for(i=1;i<=n; i++)
        {

          document.write(i);

          document.write("</br>");

        }

        document.write("</br>");

    </script>

  </body>

</html>
```