Greeting!

UNIT IV – JavaScript

Chapter 15: Control Structure in JavaScript

Conditional Statements in JavaScript:

Statements are executed in the order they are found in a script. Conditional statements execute or skip one or set of statements depending on the value of a specified conditional expression. There are two types of controls,

Branching / Selection

Looping / repetitive

Branching Statements:

JavaScript supports branching statements which are used to perform different actions based on different conditions. Branching is a transfer of control from the current statement to another statement or construct in the program unit. A branch alters the execution sequence. There are different branching statements. They are,

- **if** statement
- **if … else** statement
- **else if** statement
- **switch** statement

15.1.1.1 if and if..else Statement:

The **if** statement is the fundamental control statement that allows JavaScript to make decisions to execute statements conditionally. This statement has two forms. The form is for only true condition. The syntax is

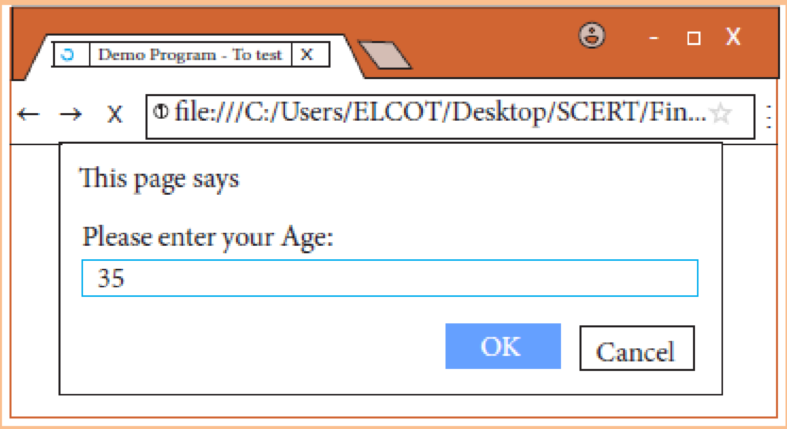
```
if (condition)
{
        True block;
}
```

In the **if** form, condition contains relational/logical expression is evaluated. If the resulting value is true the true block is executed. True block may contain one or more than one statement. For example

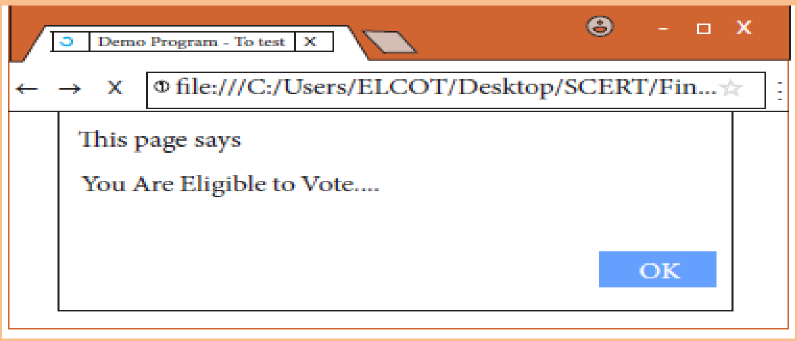
Illustration 15.1 Demo to Test if command

```
<Html>
<Head>
        <Title>Demo Program - To test if command in JavaScript </Title>
</Head>
<Body>
        <script language="javascript" type="text/javascript">
        var age = prompt("Please enter your Age :", "0");
        if(age>=16)
{
        alert("You Are Eligible to Vote ....");
}
</script>
</Body>
</Html>
```

The output will be



```
This page says
Please enter your Age:
35
```



```
This page says
You Are Eligible to Vote....
```

The second form of the **if** statement is an **else** clause that is the program to follow either of two branches depending on the condition. In the simple if construction, no special processing is performed when the condition evaluates to false. But if processing must follow one of two paths, hence need to use **if...else** format. Its **syntax** is:

if (expression)

{ statements if true }

else

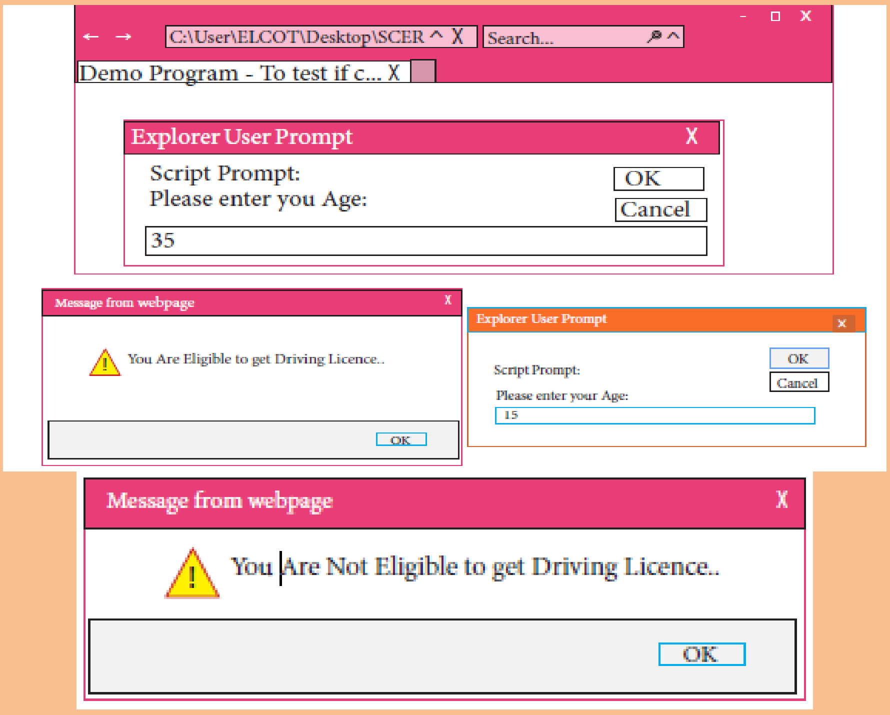
{ statements if false }

This form is similar to if statement but the only difference is the **else** keyword, which provides an alternate path for execution to follow if the condition evaluates to false.

Illustration 15.2 Using if.. else statement

```
<Html>
<Head>
        <Title>Demo Program - To test if..else command in JavaScript </Title>
</Head>
<Body>
        <script language="javascript" type="text/javascript">
        var age = prompt("Please enter your Age :", "0");
        if(age>=18)
        {               alert("You Are Eligible to get Driving Licence..");         }
        else
        {       alert("You Are Not Eligible to get Driving Licence..");   }
</script>     </Body>       </Html>
```

The output will be



```
Explorer User Prompt
Script Prompt:
Please enter you Age:
35
```

```
Message from webpage
You Are Eligible to get Driving Licence..
```

```
Explorer User Prompt
Script Prompt:
Please enter your Age:
15
```

```
Message from webpage
You Are Not Eligible to get Driving Licence..
```

else if Statement:

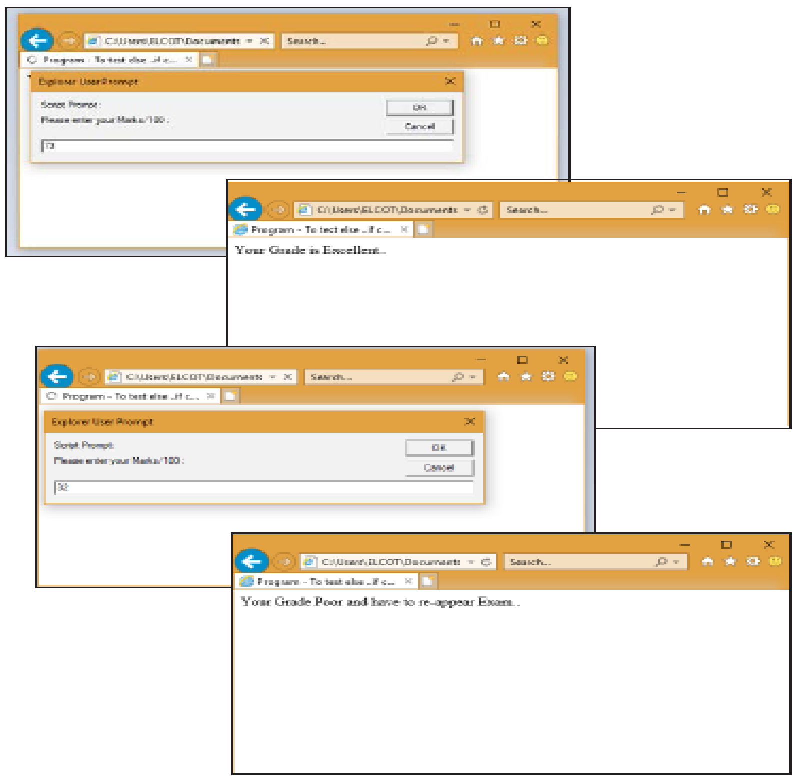
The **if ... else** statement evaluates an expression and executes one of two pieces of code, depending on the outcome. The **else if** statement to specify a new condition if the first condition is false.

```
        if (n == 10)
        {       // Execute code block #1    }
else if (n == 20)
        {       // Execute code block #2    }
else if (n == 30)
        {        // Execute code block #3    }
else
        {       // If all else fails, execute block #4    }
```

Illustration 15.3 Using Logical Operators and else if Statement

```
<Html>
<Head>
        <Title>Program - To test else ..if command in JavaScript </Title>
</Head>
<Body>
<script language="javascript" type="text/javascript">
        var marks = prompt("Please enter your Marks/100 :", "0");
        if(marks>90)
        {       document.write("Your Grade is Outstanding.."); }
        else if((marks>70) && (marks<=90))
        {       document.write("Your Grade is Excellent..");      }
        else if((marks>50) && (marks<=70))
        {       document.write("Your Grade is Good..");           }
        else if((marks>40) && (marks<=50))
        {       document.write("Your Grade is Satisfectory..");  }
        else
       {document.write("Your Grade Poor and  have to re-appear Exam..");   }
</script>      </Body>      </Html>
```

The output will be



```
Script Prompt:
Please enter your Marks/100 :
73
```

```
Your Grade is Excellent..
```

```
Script Prompt:
Please enter your Marks/100 :
32
```

```
Your Grade Poor and have to re-appear Exam..
```

There is nothing special about this code. It is just a series of **if** statements, where each following **if** is a part of the **else** clause of the previous statement. Using the else if idiom is preferable to, and more legible than, writing these statements out in their syntactically equivalent, fully nested form:

```
if (n == 10)
          {         // Execute code block #1    }
else
          {         if (n == 20)
          {         // Execute code block #2    }
else  {
if (n == 30)
          {         // Execute code block #3    }
else  {             // If all else fails, execute block #4
}
          }
```

switch case Statement:

JavaScripts offers the **switch** statement as an alternate to using **if...else** structure. The switch statement is especially useful when testing all the possible results of an expression. The syntax of a switch structure as the following:

```
switch(expression)
          {
          `         case label1:
                              statements1;
                              break;
                    case label2:
                              statements2;
                              break;
                    case labeln;
                              statements - N;
                              break;
                    default:
                              statements;
}
```

Break and Default Statement

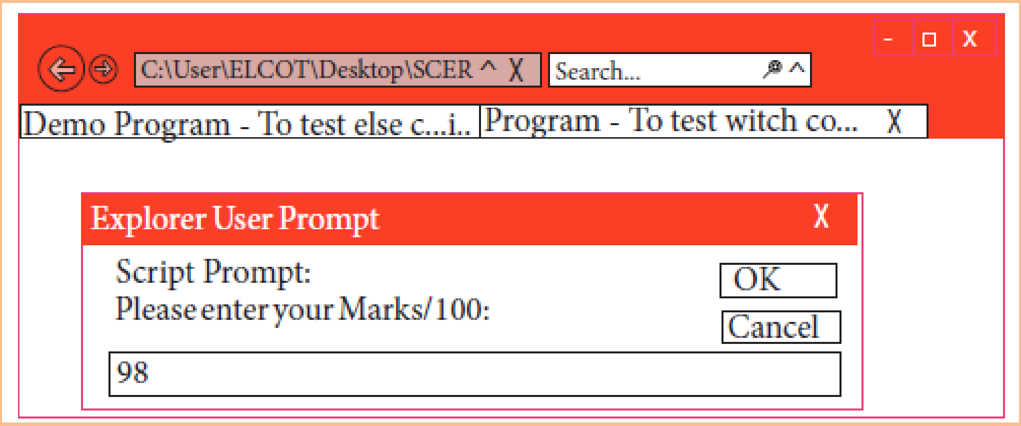
The switch statement begins by evaluating an expression placed between parenthesis, much like the if statement. The result compared to labels associated with case structure that follow the switch statement. If the result is equal to a label, the statements in the corresponding case structure are executed. The **default** structure is can be at the end of a switch structure if the result of the expression that do not match any of the case labels. The **break** statement is also used commonly within switch to exit the statement once the appropriate choice is found.

Illustration 15.4 Using Switch Statement

```
<Html>
            <Head>
            <Title>Program - To test witch command in JavaScript </Title>
            </Head>
      <Body>
      <script language="javascript" type="text/javascript">
      var grade=0;
      var marks=prompt("Please enter your marks/100:","0");
      if(marks>90)
      {grade=1;}
      else if(marks>70)&&(marks<=90)
      {grade=2;}
      else if(marks>50)&&(marks<=70)
      {grade=3;}
```

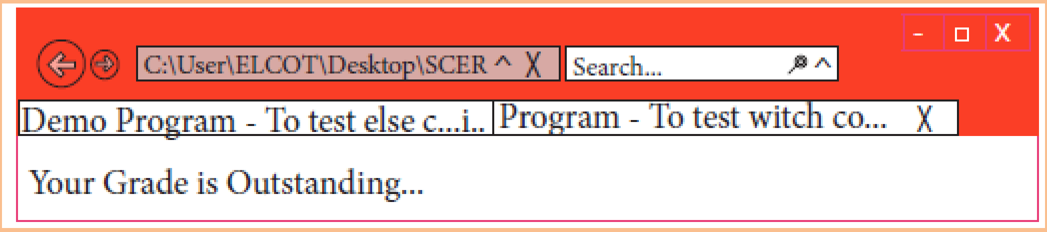
```
else if(marks>40v)&&(marks<=50)
{grade=4;}
else
{grade=5;}
switch(grade)
{
case 1:
document.write("Your Grade is Outstanding..");
      break;
      case 2:
document.write("Your Grade is Excellent..");
      break;
      case 3:
document.write("Your Grade is Good..");
break;
case 4:
document.write("Your Grade is Satisfectory..");
      break;
default:
document.write("Your Grade Poor and  have to re-appear Exam..");
}
</script>
</Body>
</Html>
```

The output will be

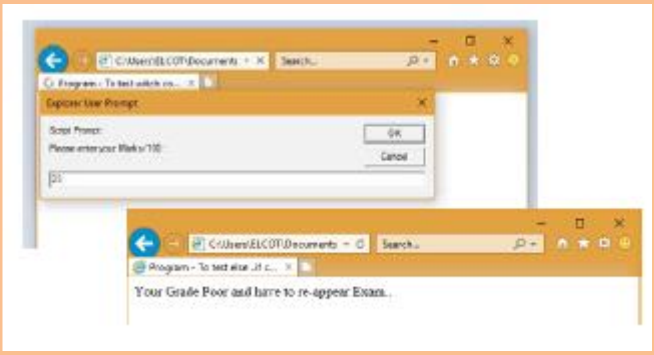

C:\User\ELCOT\Desktop\SCER
Search...
Demo Program - To test else c...i..
Program - To test witch co...
Explorer User Prompt
OK
Cancel

Script Prompt:
Please enter your Marks/100:

98


C:\User\ELCOT\Desktop\SCER
Search...
Demo Program - To test else c...i..
Program - To test witch co...

Your Grade is Outstanding...


C:\Users\ELCOT\Documents
Search...
Program - To test witch co...
Explorer User Prompt
OK
Cancel
Program - To test else ..if c...

Script Prompt:
Please enter your Marks/100

Your Grade Poor and have to re-appear Exam..

Looping / repetitive

In JavaScript there are times when the same portion of code needs to be executed many times with slightly different values is called Loops. JavaScript supports three kinds of looping statements. They are

- **for** loop
- **while** loop
- **do..while** loop

for loop

The **for** loop is a very rigid structure that loops for a pre-set number of times. In JavaScript **for** structure is very flexible, which makes this type is very useful. The syntax of the **for** loop looks like the following:

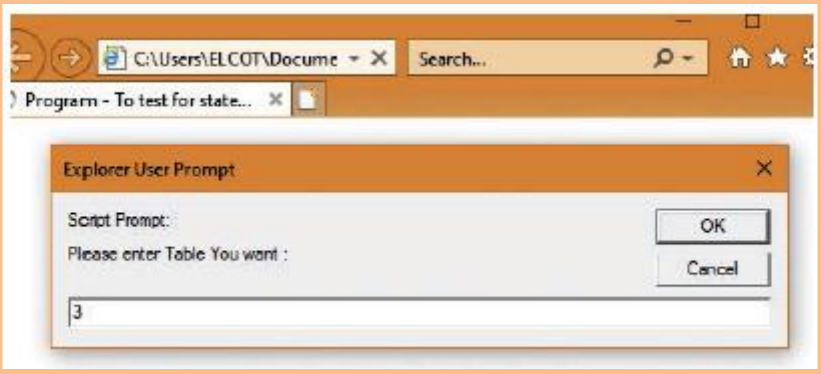
```
for(initialization; condition; increment/decrement)
        {
                Body of the loop;
        }
```

The for structure within parenthesis there are three parts each separated by semicolon. They are,

1. The first part of the loop initialize a variable which is also called as control variable. In most case the control variable is declared as well as initialized.
2. The second part is the conditional statement that determines how many times the loop will be iterated.
3. The third and final part determines how the value of control variable is changed (Incremented/Decremented)

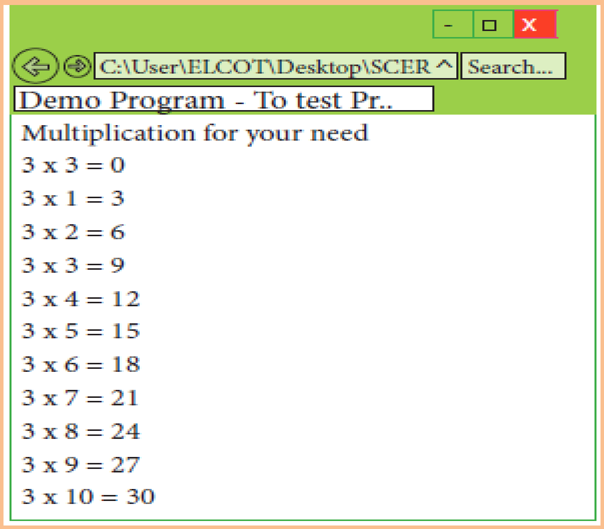
Illustration 15.5 Using for loop

```
<Html>
<Head>
        <Title> Program - To test for statement in JavaScript </Title>
</Head>
<Body>
        <script language="javascript" type="text/javascript">
        var no1 = prompt("Please enter Table You want :", "0");
        document.write("<h2> Multiplication for your need </h2>");
        for( var no2=0;no2<=10;no2++)
{
        document.write(no1+" x "+no2+" = "+no1*no2+"<br>");
}
</script>
</Body>
</Html>
```



Script Prompt:
Please enter Table You want :

```
3
```



```
Multiplication for your need
3 x 3 = 0
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
3 x 10 = 30
```

break and continue statement

JavaScript also supports statements used to modify flow control, specifically **break** and **continue**. The **break** statement will terminate the loop early. For example,

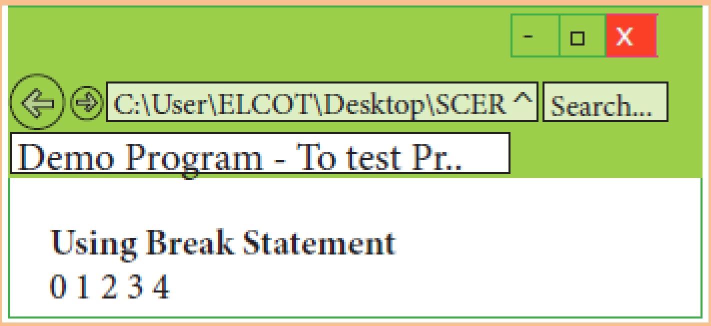
```
for(var n=0;n<=10;n++)
        {       if(n==5)
        {        break;        }
        document.write(n+"<br>");
        }
```

In the above example, which writes out the value of n starting from 0, when n is equal to 5 the break statement is executed and the loop is terminated and the output is as follows,

Illustration 15.6 Using break statement

```
<Html>
  <Head>
      <Title>Demo Program - To test Break command in JavaScript </Title>
  </Head>
 <Body>
   <script language="javascript" type="text/javascript">
   document.write("<h2> Using Break Statement </h2>");
    for( var no2=0;no2<=10;no2++)
    {
     if(no2==5)
     {break;}
     document.write(no2+" ");
    }
    </script>

 </Body>
</Html>
```



```
Using Break Statement
0 1 2 3 4
```

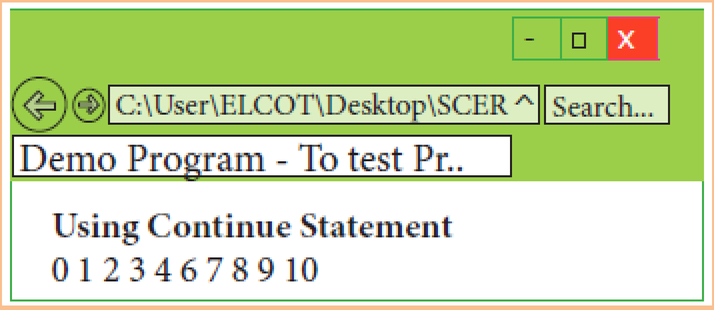
The **continue** statement will skip back to the loop condition check. When the **continue** statement is executed, the current iteration of the enclosing loop is terminated, and the next iteration begins. For example,

```
for(var n=0;n<=10;n++)
        {
                if(n==5)
        document.write(n+"<br>");
        }
```

In the above example, which writes out the value of n starting from 0, when n is equal to 5 the continue statement is executed and the **continue** statement continues the loop without printing the value 5 and the output is as follows,

Illustration 15.7 Using continue statement

```
    <Html>
  <Head>
     <Title>Demo Program - To test Continue command in JavaScript </Title>
  </Head>
 <Body>
   <script language="javascript" type="text/javascript">
    document.write("<h2> Using continue Statement </h2>");
    for( var no2=0;no2<=10;no2++)
    {
     if(no2==5)
     {continue;}
     document.write(no2+" ");
    }
    </script>
 </Body>
</Html>
```



```
Using Continue Statement
0 1 2 3 4 6 7 8 9 10
```

while loop

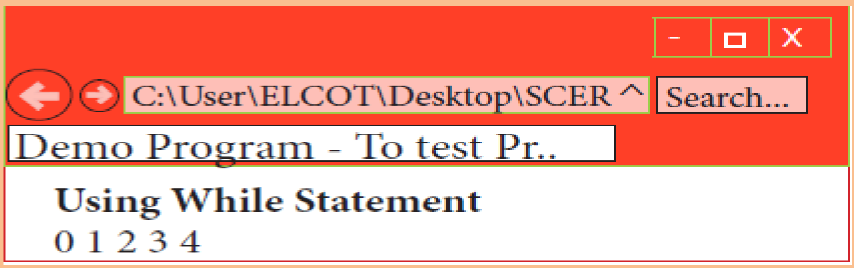
In JavaScript **while** loop is another most basic loop. The purpose of a **while** loop is to execute a statement /block of statement repeatedly as long as an expression is true. The while statement creates a loop that executes a specified statement as long as the test condition evaluates to true. The condition is evaluated before executing the statement.

The syntax is:

```
while (condition)
{
        body of the loop
}
```

Illustration 15.8 Using while loop

```
<Html>
<Head>
        <Title>Program - To test while statement in JavaScript </Title>
</Head>
<Body>
        <script language="javascript" type="text/javascript">
        document.write("<h2> Using while Statement </h2>");
        var no2=0;
        while(no2<=5)
{
        document.write(no2+" ");
        no2=no2+1;
}
</script>
</Body>
</Html>
```



```
Using While Statement
0 1 2 3 4
```

To execute a while statement, the interpreter first evaluates expression. If the value of the expression is true the interpreter executes the statement and repeats, jumping back to the top of the loop and evaluating expression again. In the above example let us see how the while command is executed,

1. Initial value of the variable no2 is set to 0
2. The expression in the while statement is executed

3. If the condition is true then body of the loop will be executed once otherwise body of the loop will be skipped and control will jump to next statement to end of the loop.
4. Then the value of the control variable is executed and control jumps to condition again go to step 3

do .. while loop

The **do..while** loop is like a while loop, except that the loop expression is tested at the end of the loop rather than at the beginning. This means that the body of the loop is always executed at least once. The syntax is:

do

{

body of the loop

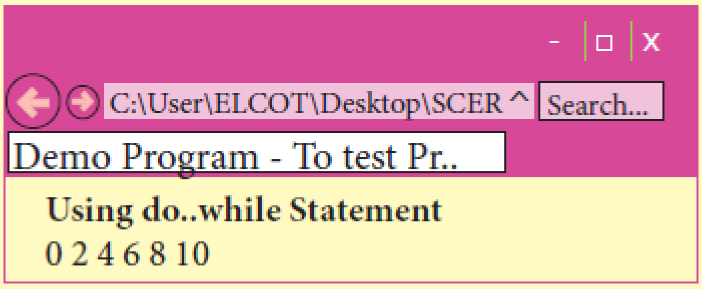
} while (expression);

An important difference between while and do..while statement is in the do..while loop body of the loop always executed at least once before the condition can be executed. In a while loop, first condition will be evaluated and then only based on the result of the condition the body of the loop will be executed or not.

Illustration 15.9 Using do..while loop

```
<Html>
<Head>
        <Title>Program - To test do..while statement in JavaScript </Title>
</Head>
<Body>
        <script language="javascript" type="text/javascript">
        document.write("<h2> Using do..while Statement </h2>");
        var no2=0;
        do
{
document.write(no2+" ");
no2=no2+2;
}while(no2<=10);
</script>
</Body>
</Html>
```

Output:



```
Using do..while Statement
0 2 4 6 8 10
```