Control statements, Loops and Popup boxes in JavaScript

Objectives : At the end of this lesson you shall be able to
- **explain control statements**
- **discuss about various Loops**
- **explain the uses of Popup boxes.**

Control Statements: When we write code for a particular program, we sometimes takes various decisions for executing different action. These can be done through conditional/control statements.

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.

The if Statement

Use the if statement to specify a block of JavaScript code to be executed if a condition is true.

Syntax

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

Example 1

Make a "Good day" greeting if the time is less than 18:00:

```
if (time < 18) {
   greeting = "Good day";
}
```

The **result** of greeting will be:

Good day

The else Statement

Use the else statement to specify a block of code to be executed if the condition is false.

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

Example 2

If the time is less than 18:00, create a "Good day" greeting, otherwise "Good evening":

```
if (time < 18) {
   greeting = "Good day";
} else {
   greeting = "Good evening";
}
```

The **result** of greeting will be:

Good day

The else if Statement

Use the else if statement to specify a new condition if the first condition is false.

Syntax

```
if (condition1) {
   block of code to be executed if condition1 is true
} else if (condition2) {
   block of code to be executed if the condition1 is false and condition2 is true
} else {
   block of code to be executed if the condition1 is false and condition2 is false
}
```

Example 3

If time is less than 10:00, create a "Good morning" greeting, if not, but time is less than 18:00, create a "Good day" greeting, otherwise a "Good evening":

```
if (time < 10) {
   greeting = "Good morning";
} else if (time < 18) {
   greeting = "Good day";
} else {
```

```
  greeting = "Good evening";
}
```

The **result** of x will be:

Good day

The JavaScript Switch Statement

Use the switch statement to select one of many blocks of code to be executed.

Syntax

```
switch(expression) {
  case n1:
    code block
    break;
  case n2:
    code block
    break;
  default:
    default code block
}
```

This is how it works:

- The switch expression is evaluated once.
- The value of the expression is compared with the values of each case.
- If there is a match, the associated block of code is executed.

Example 4

Use today's weekday number to calculate weekday name: (Sunday=0, Monday=1, Tuesday=2, ...)

```
switch (new Date().getDay()) {
  case 0:
    day = "Sunday";
    break;
  case 1:
    day = "Monday";
    break;
  case 2:
    day = "Tuesday";
    break;
  case 3:
    day = "Wednesday";
    break;
  case 4:
    day = "Thursday";
    break;
  case 5:
    day = "Friday";
    break;
  case 6:
    day = "Saturday";
    break;
}
```

The **result** of day will be:

Tuesday

The break Keyword

When the JavaScript code interpreter reaches a break keyword, it breaks out of the switch block.

This will stop the execution of more execution of code and/or case testing inside the block.

The default Keyword

The default keyword specifies the code to run if there is no case match:

Example 5

If today is neither Saturday nor Sunday, write a default message:

```
switch (new Date().getDay()) {
  case 6:
    text = "Today is Saturday";
    break;
  case 0:
    text = "Today is Sunday";
    break;
  default:
    text = "Looking forward to the Weekend";
}
```

The **result** of text will be:

Looking forward to the Weekend

Common Code and Fall-Through

Sometimes, in a switch block, you will want different cases to use the same code, or fall-through to a common default.

Note from the next example, that cases can share the same code block and that the default case does not have to be the last case in a switch block:

Example 6

```
switch (new Date().getDay()) {
    case 1:
    case 2:
    case 3:
   default:
       text = "Weekend is coming";
       break;
 case 4:
    case 5:
      text = "Weekend is soon";
       break;
    case 0:
    case 6:
      text = "Now in Weekend";
}
```

JavaScript Loops

Loops are handy, if you want to run the same code over and over again, each time with a different value.

Often this is the case when working with arrays:

Instead of writing:

```
text += train[0] + "<br>";
text += train [1] + "<br>";
text += train [2] + "<br>";
text += train [3] + "<br>";
text += train [4] + "<br>";
text += train [5] + "<br>";
```

You can write:

```
for (i = 0; i < train.length; i++) {
    text += train [i] + "<br>";
}
```

Different Kinds of Loops

JavaScript supports different kinds of loops:

- for - loops through a block of code a number of times
- for/in - loops through the properties of an object
- while - loops through a block of code while a specified condition is true
- do/while - also loops through a block of code while a specified condition is true

The For Loop

The for loop is often the tool you will use when you want to create a loop.

The for loop has the following syntax:

```
for (statement 1; statement 2; statement 3) {
    code block to be executed
}
```

Statement 1 is executed before the loop (the code block) starts. It is called Initialisation Part

Statement 2 defines the condition for running the loop (the code block).It is called condition part.

Statement 3 is executed each time after the loop (the code block) has been executed. It is called increment/ decrement part.

Example 7

```
for (i = 0; i < 5; i++) {
    text += "The number is " + i + "<br>";
}
```

From the example above, you can read:

Statement 1 sets a variable before the loop starts (var i = 0).

Statement 2 defines the condition for the loop to run (i must be less than 5).

Statement 3 increases a value (i++) each time the code block in the loop has been executed.

Initialisation Part

Normally you will use statement 1 to initiate the variable used in the loop (var i = 0).

This is not always the case, JavaScript doesn't care. Statement 1 is optional.

You can initiate many values in statement 1 (separated by comma):

Example 8

```
for (i = 0, len = train.length, text = ""; i < len; i++) {
    text += train [i] + "<br>";
}
```

And you can omit statement 1 (like when your values are set before the loop starts):

Example 9

```
var i = 2;
var len = train.length;
var text = "";
for (; i < len; i++) {
    text += train [i] + "<br>";
}
```

Condition Part

Often statement 2 is used to evaluate the condition of the initial variable.

This is not always the case, JavaScript doesn't care. Statement 2 is also optional.

If statement 2 returns true, the loop will start over again, if it returns false, the loop will end.

If you omit statement 2, you must provide a break inside the loop. Otherwise the loop will never end. This will crash your browser. Read about breaks in a later chapter of this tutorial.

Increment/Decrement Part

Often statement 3 increases the initial variable.

This is not always the case, JavaScript doesn't care, and statement 3 is optional.

Statement 3 can do anything like negative increment (i--), or larger increment (i = i + 15), or anything else.

Statement 3 can also be omitted (like when you increment your values inside the loop):

Example 10

```
var i = 0;
len = train.length;
for (; i < len; ) {
    text += train [i] + "<br>";
    i++;
}
```

For/In Loop

The JavaScript for/in statement loops through the properties of an object:

```
var person = {fname:"Raja", lname:"Sen", age:35};
var text = "";
var x;
for (x in person) {
    text += person[x];
}
```

While Loop

The while loop loops through a block of code as long as a specified condition is true.

Syntax

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

In the following example, the code in the loop will run, over and over again, as long as a variable (i) is less than 10:

Example 11

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

If you forget to increase the variable used in the condition, the loop will never end. This will crash your browser.

The Do/While Loop

The do/while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

Syntax

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

The example below uses a do/while loop. The loop will always be executed at least once, even if the condition is false, because the code block is executed before the condition is tested:

Example 12

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

Do not forget to increase the variable used in the condition, otherwise the loop will never end!

Comparing For and While

If you have read the previous chapter, about the for loop, you will discover that a while loop is much the same as a for loop, with statement 1 and statement 3 omitted.

The loop in this example uses a for loop to collect the car names from the train array:

Example 13

```
train = ["Duronto","Satabdi","Garib Rath","Rajdhani"];
var i = 0;
var text = "";
for (;train[i];) {
    text += train[i] + "<br>";
    i++;
}
```

The loop in this example uses a while loop to collect the car names from the train array:

```
train = ["Duronto","Satabdi","Garib Rath","Rajdhani"];
var i = 0;
var text = "";
while (train[i]) {
text += train[i] + "<br>";
    i++;
}
```

The Break Statement in Loop

Break statement is used to terminate a loop before its completion. It saves machine time for not iterating a loop uselessly.

For example: In linear search, if we find the item then we can break the loop as no point of runnign it unnecessary.

Example 14

```
for(i=0;i<l;i++ {
        if(arr[i]==item) {
                alert("Found at :"+i);
                fl=1;
                break;
                }
if(fl==0) alert("Not Found");
```

Here, if the item is found, loop breaks and CPU time is saved.

Popup Boxes

JavaScript has three kind of popup boxes. They are

1 Alert box

2 Confirm box and

3 Prompt box.

Alert Box

An alert box is often used if you want to make sure information comes through to the user. When an alert box pops up, the user will have to click “OK” to proceed.

Syntax

window.alert(“*sometext*”);

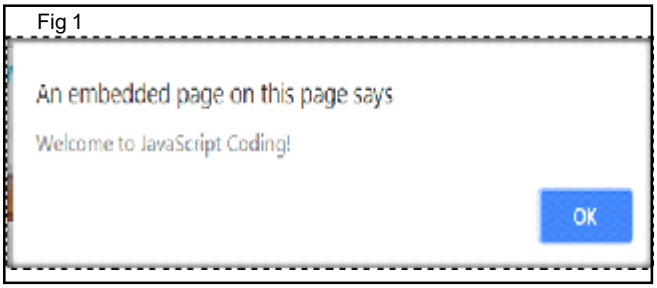
Note: The window.alert() method can be written without the window prefix.

Example 15

alert (“Welcome to Java Script Coding!;)

The result is shown in Fig 1.

Fig 1



Confirm Box

A confirm box is often used to verify or accept something.When a confirm box pops up, the user will have to click either “OK” or “Cancel” to proceed.If the user clicks “OK”, the box returns **true**. If the user clicks “Cancel”, the box returns **false**.

Syntax

window.confirm(“*sometext*”);

Note: The window.confirm() method can be written without the window prefix.

Example 16

```
if (confirm(“Click a button!”))
{
    txt = “ You clicked OK!”;
}
else
{
    txt = “You clicked Cancel!”;
}
```

The result is is shown in Fig 2.

Fig 2

Note: When click on OK button it displays the message "You clicked OK!" and when click on Cancel button it displays the message "You clicked Cancel!"

Prompt Box

A prompt box is often used if the user to input a value before entering a page.When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.

Syntax

window.prompt("*sometext*","*default text*");

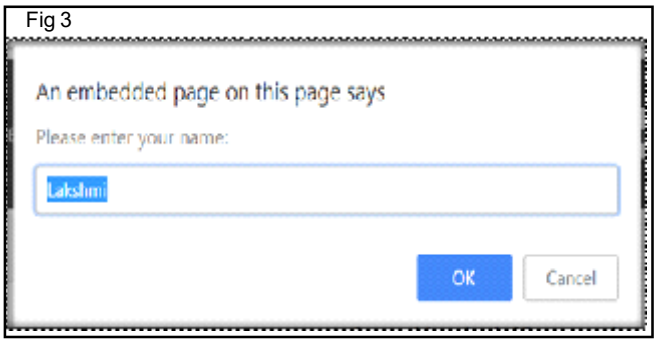
Note: The window.prompt() method can be written without the window prefix.

Example 17

```
var tname = promp t("Please Enter your Name", "Lakshmi");
    if (tname == null || tname == "")
      {txt = "User cancelled the prompt.";
    }
    else
       {txt = "Hello " + tname + "! Congratulations!!!!!");
    }
```

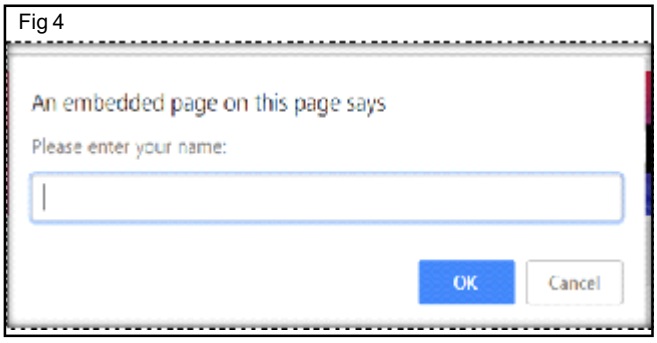
The result is is shown in Fig 3.

Fig 3



Note: If click on OK button it displays the message "Hello Lakshmi! Congratulations!!!!!" If cancelled the name 'Lakshmi' as shown in Fig 4 it gives the message "User cancelled the Prompt". Also whenclick the Cancel button even when if the box has text 'Lakshmi' it gives the message "User cancelled the Prompt".

Fig 4



Line Breaks

To display line breaks inside a popup box, use a back-slash followed by the character n.

Example 18

alert("Hello\nWelcome!");

The result is shown in Fig 5.

Fig 5