Instructor Notes:

Add instructor notes here.

Instructor Notes:

Explain the lesson coverage

Lesson Objectives

- Conditional Statements
- If Statements
- If Else Statements
- If..ElseIf..Else Statements
- Nested If Statement
- Select Case Statements
- Loops
- Do..While Loop and Do…Until Loop
- While…Wend Loop
- For….Next Loop
- For-Step-Next Loop
- For...Each Loops

Instructor Notes:

Additional notes for instructor

Using Conditional Statements and Loops in VBScript

- Conditional Statements:
  - If..Then ,
  - If …Then.. Else..,
  - If …Then.. ElseIf..,
  - Select Case
- Loops
  - Do While Loop
  - Do Until Loop
  - For Next
  - For-Step-Next
  - For Each Next

Add the notes here.

Instructor Notes:

Conditional Statements

- In VBScript there are four conditional statements :
  - If statement - executes a set of code when a condition is true
  - If...Then...Else statement - select one of two sets of lines to execute
  - If...Then...ElseIf statement - select one of many sets of lines to execute
  - Select Case statement - select one of many sets of lines to execute

Instructor Notes:

If Statements

- An If statement consists of a Boolean expression followed by one or more statements. If the condition is said to be True, the statements under If condition(s) are Executed
- If the Condition is said to be False, the statements after the If loop are executed.
- The syntax of an If statement in VBScript is:

```
If(boolean_expression) Then
Statement 1 ..... .....
Statement n
End If
```

Instructor Notes:

If Else Statements

- An If statement consists of a boolean expression followed by one or more statements. If the condition is said to be True, the statements under If condition(s) are Executed. If the Condition is said to be False, the statements under Else Part would be executed.
- The syntax of an if statement in VBScript is:

```
If(boolean_expression) Then
        Statement 1
        ......
        ....
        Statement n
Else
        Statement 1
        .....
        ....
        Statement n
End If
```

Instructor Notes:

If..ElseIf..Else Statements

- An If statement followed by one or more ElseIf Statements that consists of boolean expressions and then followed by a default else statement, which executes when all the condition becomes false.
- The syntax of an If-ElseIf-Else statement in VBScript is:

```
If(boolean_expression) Then
      Statement 1
       .....
      Statement n
 ElseIf (boolean_expression) Then
      Statement 1
      .....
      Statement n
ElseIf (boolean_expression) Then
      Statement 1
       .....
      Statement n
 Else
      Statement 1
      .....
      Statement n
End If
```

Instructor Notes:

Nested If Statement

- An If or ElseIf statement inside another If or ElseIf statement(s).
- The Inner If statements are executed based on the Outermost If statements. This enables VBScript to handle complex conditions with ease
- The syntax of a Nested if statement in VBScript is:

```
If(boolean_expression) Then
Statement 1
Statement n
If(boolean_expression) Then
Statement 1
Statement n
 ElseIf (boolean_expression) Then
 Statement 1
Statement n
 Else
Statement 1
Statement n
 End If
 Else
Statement 1
Statement n
 End If
```

Instructor Notes:

Select Case Statements

- When a User want to execute a group of statements depending upon a value of an Expression, then Switch Case is used.
- Each value is called a Case, and the variable being switched ON based on each case.
- Case Else statement is executed if test expression doesn't match any of the Case specified by the user.
- Case Else is an optional statement within Select Case, however, it is a good programming practice to always have a Case Else statement

The Select-Case Statement

Suppose you were designing a program to decide whether to hire people for programming jobs based on their grades on the final exam in their VBScript programming course. You'll make an offer to those who received an A; you'll bring those who received a B in for an interview; you won't consider anybody else. For example, you could write this:

```
If Grade = "A" Then
MsgBox "We would like to hire you!"
If Grade = "B" Then
MsgBox "Please schedule an interview."
If (Grade <> "A") And (Grade <> "B") Then
MsgBox  "I am sorry, we won't be able to consider your application."
```

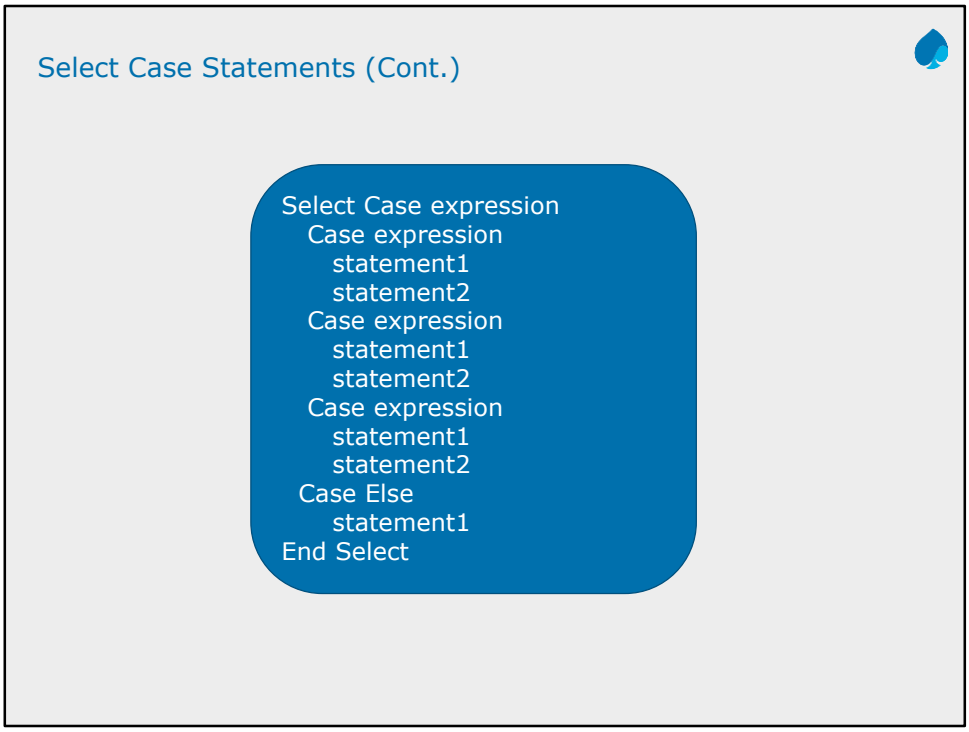
Using the Select-Case statement, however, you can write this:

```
Select Case Grade
  Case "A”
   MsgBox "We would lik e to hire you!"
Case "B"
   MsgBox "Please s chedule an interview.“
 Case "C"
    MsgBox "I am sorry, we won't be able to consider your application."
End Select
```

The Select-Case statement makes it clear that a program has reached a point with many branches; multiple If-Then statement's do not.
(And the clearer a program is, the easier it is to debug.)

The elimination of all cases that require special testing is so common that VBScript has a way of lumping all the remaining cases into one. This is represented (naturally enough) by the keywords Case Else

Instructor Notes:



The Select-Case Statement

Suppose you were designing a program to decide whether to hire people for programming jobs based on their grades on the final exam in their VBScript programming course. You'll make an offer to those who received an A; you'll bring those who received a B in for an interview; you won't consider anybody else. For example, you could write this:

```
If Grade = "A" Then
MsgBox "We would like to hire you!"
If Grade = "B" Then
MsgBox "Please schedule an interview."
If (Grade <> "A") And (Grade <> "B") Then
MsgBox  "I am sorry, we won't be able to consider your application."
```

Using the Select-Case statement, however, you can write this:

```
Select Case Grade
  Case "A”
   MsgBox "We would lik e to hire you!"
Case "B"
   MsgBox "Please s chedule an interview.“
 Case "C"
    MsgBox "I am sorry, we won't be able to consider your application."
End Select
```

The Select-Case statement makes it clear that a program has reached a point with many branches; multiple If-Then statement's do not.
(And the clearer a program is, the easier it is to debug.)

The elimination of all cases that require special testing is so common that VBScript has a way of lumping all the remaining cases into one. This is represented (naturally enough) by the keywords Case Else

Instructor Notes:

Additional notes for instructor

Demo

- Conditional statements demo
- ifstatement
- Ifelsestatement
- Ifelseifelsestatement
- nestedifstatement
- selectcasestatement

Add the notes here.

Instructor Notes:

Loops

- A loop statement allows us to execute a statement or group of statements multiple times.
- VBScript provides the following types of loops to handle looping requirements:
  - Do While Loop
  - Do Until Loop
  - While...Wend Loop
  - For Next
  - For-Step-Next
  - For Each Next

Instructor Notes:

Do..While Loop and Do…Until Loop

- A Do..While and Do…Until loop is used when one wants to repeat a set of statements as long as the condition is true. The Condition may be checked at the beginning of the loop or at the end of the loop.
- The syntax of a Do..While loop in VBScript is:

```
Do While/Until condition
[statement 1]
[statement 2]
 ...
[statement n]
[Exit Do]
[statement 1]
[statement 2]
[statement n]
Loop
```

Instructor Notes:

While…Wend Loop

- In a While…Wend loop, if the condition is True, all statements are executed until Wend keyword is encountered.
- If the condition is false, the loop is exited and the control jumps to very next statement after Wend keyword
- The syntax of a While…Wend loop in VBScript is:

```
While condition(s)
 [statements 1]
 [statements 2]
 ...
 [statements n]
Wend
```

Instructor Notes:

For….Next Loop

- The For...Next statement is used to run a block of code a specified number of times.
- The For statement specifies the counter variable, and its start and end values. The Next statement increases the counter variable by one.

Instructor Notes:

For-Step-Next Loop

- With the Step keyword, you can increase or decrease the counter variable by the value you specify.
- In the example below, the counter variable (m) is Increased by 3, each time the loop repeats

```
For m=1 To 10 Step 3
    some code
Next
```

- To decrease the counter variable, you must use a negative Step value. You must specify an end value that is less than the start value.
- In the example below, the counter variable m is decreased by 3, each time the loop repeats.

```
For m=10 To 3 Step -3
    some code
Next
```

Instructor Notes:

For...Each Loops

- A For Each loop is used when we want to execute a statement or a group of statements for each element in an array or collection.
- The syntax of a For Each loop in VBScript is:

```
For Each element In Group
 [statement 1]
 [statement 2]
 [Exit For]
 [statement 11]
 [statement 22]
Next
```

A **For Each** loop is similar to For Loop; however, the loop is executed for each element in an array or group. Hence, the step counter won't exist in this type of loop and it is mostly used with arrays or used in context of File system objects in order to operate recursively

Instructor Notes:

Additional notes for instructor

Demo

- Loops demo
- Doloop
- do_untilloop
- while_wend_loop
- for_next
- for_step_next
- for_each_loop

Add the notes here.

Instructor Notes:

Additional notes for instructor

Lab

Lab 3

Instructor Notes:

Additional notes for instructor

Lesson Summary

- Decision making allows programmers to control the execution flow of a script or one of its sections. The execution is governed by one or more conditional statements.
- When you need to execute a block of code several number of times we use loops.

Instructor Notes:

1.Case Else
2. True
3. Wend

Review - Questions

- Question 1:The optional statement in Select Case is _____. (Case or Select Case)
- Question 2: In For–step-Next loop ,you must specify an end value that is less than the start value.
- True or False
- Question 3: In a While..Wend loop, if the condition is True, all statements are executed until _____ keyword is encountered.