Instructor Notes:

Add instructor notes here.

VBScript

Lesson 5: User-Defined Functions & Sub-Procedures in VBScript

Instructor Notes:

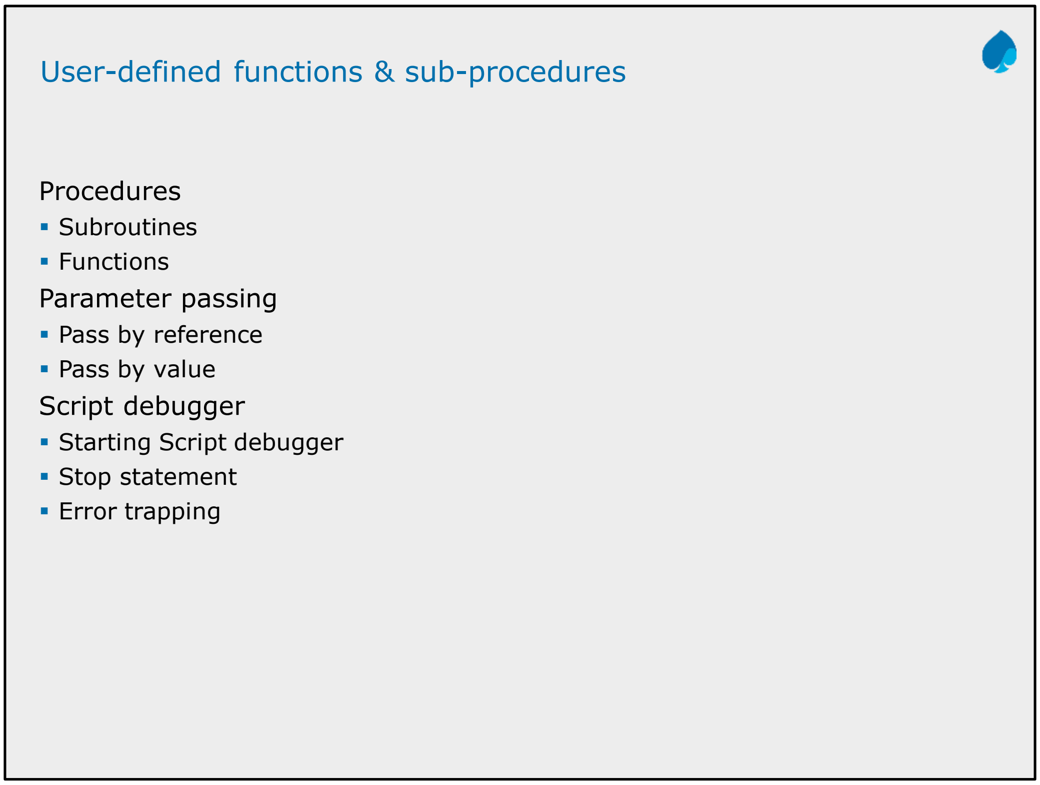
Explain the lesson coverage

Lesson Objectives

- Procedure and Function
- User Defined Functions
- Sub-procedures
- Passing by Reference Versus Passing by Value
- Example for Pass ByVal
- Example for Pass By Ref
- Dialog Boxes
- Dialog Box- Message Box
- Dialog Box: InputBox

Instructor Notes:

Additional notes for instructor



User-defined functions & sub-procedures

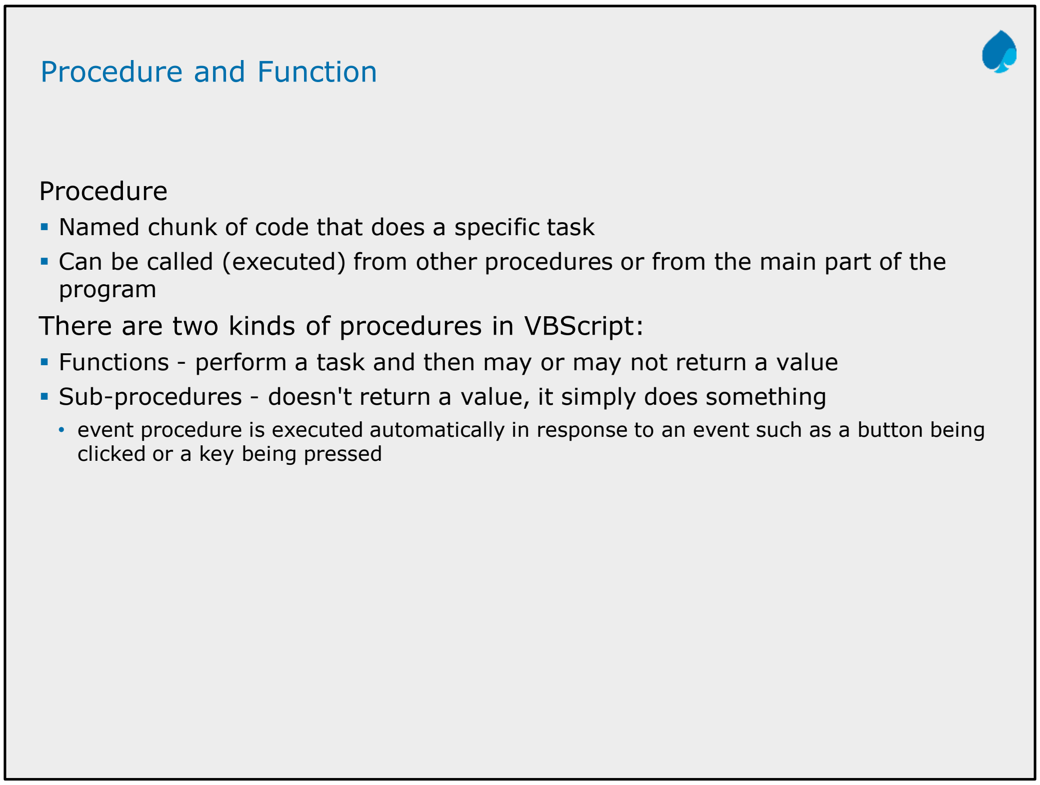
Procedures
- Subroutines
- Functions

Parameter passing
- Pass by reference
- Pass by value

Script debugger
- Starting Script debugger
- Stop statement
- Error trapping

Add the notes here.

Instructor Notes:



Procedure and Function

Procedure

- Named chunk of code that does a specific task
- Can be called (executed) from other procedures or from the main part of the program

There are two kinds of procedures in VBScript:

- Functions - perform a task and then may or may not return a value
- Sub-procedures - doesn't return a value, it simply does something
  - event procedure is executed automatically in response to an event such as a button being clicked or a key being pressed

You can send data to both functions and sub-procedures by passing them parameters.
Usually, functions don't change the data they are sent; they are written primarily to return a value. If you do write a function that changes the information sent to it, the value the function returns is normally used to indicate success or failure.
Sub-procedures that accept data, on the other hand, are usually written to process that data. Therefore, if you direct it to do so, a sub-procedure changes the data it is sent.

Instructor Notes:

Additional notes for instructor

User Defined Functions

Here's how you can write a reusable function:

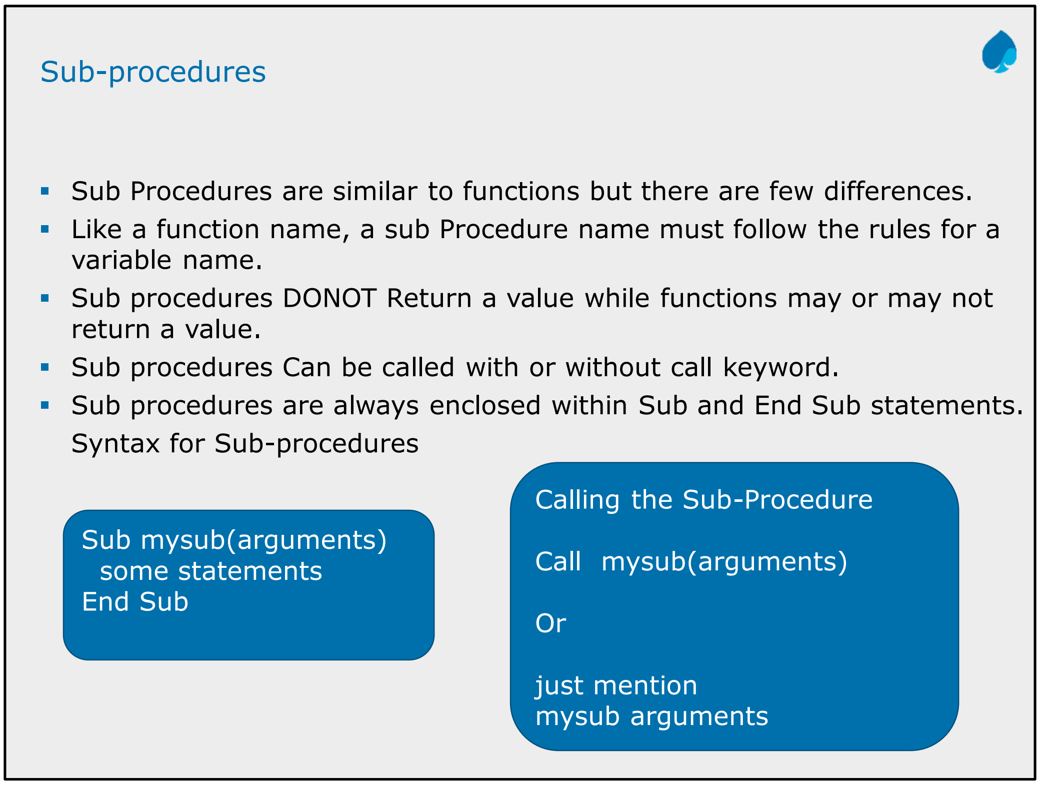
```
<html>
 <body>
   <script language="vbscript" type="text/vbscript">
               Dim variable1
               Function Functionname(parameter-list)
                    //Code
               End Function
             variable1=Functionname(parameter-list)
  </script>
</body>
</html>
```

Provided a function is enclosed within the pair of <SCRIPT> </SCRIPT> tags, VBScript is smart enough to find it no matter where you put it

Exiting a function prematurely

Occasionally, you will want to exit a function prematurely without assigning a return value. In that case, the default return value is 0 for numbers and "" for strings.
The statement that lets you do this is the Exit Function statement.

Instructor Notes:



Sub-procedures

- Sub Procedures are similar to functions but there are few differences.
- Like a function name, a sub Procedure name must follow the rules for a variable name.
- Sub procedures DONOT Return a value while functions may or may not return a value.
- Sub procedures Can be called with or without call keyword.
- Sub procedures are always enclosed within Sub and End Sub statements.

Syntax for Sub-procedures

```
Sub mysub(arguments)
  some statements
End Sub
```

Calling the Sub-Procedure

```
Call  mysub(arguments)

Or

just mention
mysub arguments
```

Sub-procedures

You write a function to automate a task and then receive a value.
Let's suppose, however, that you want to automate a task but you don't need to receive a value. In this case, you would write a sub-procedure.

For example, you might want to automate the display of text at a certain heading level. You've gotten tired of writing this:

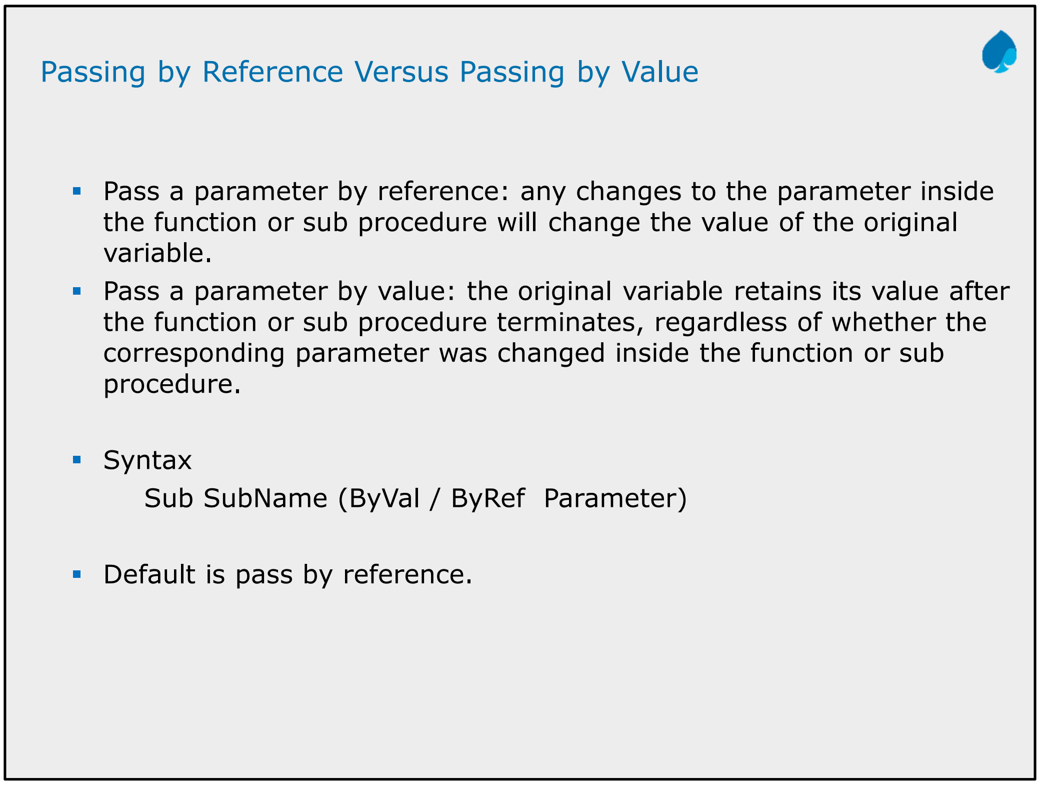
```
Document.Write "<H" & I & ">" & TheText & "</H>"
```

Here's a sub-procedure that has two parameters, one for the heading level and one for the text to display:

```
Sub WriteInHeadingLevel(HeadingLevel, TheText)
  Document.Write "<H" & (HeadingLevel & ">" & TheText & "</H>"
End Sub
```

In general, as this example shows, the first line of a subprocedure has the keyword Sub followed by the subprocedure's name.
Next comes the parameter list, enclosed in parentheses. (If your subprocedure uses no parameters, the parentheses are optional.) After the parameter list come the statements that make up the body of the subprocedure.
Finally there are the keywords End Sub, which end the subprocedure.

Instructor Notes:

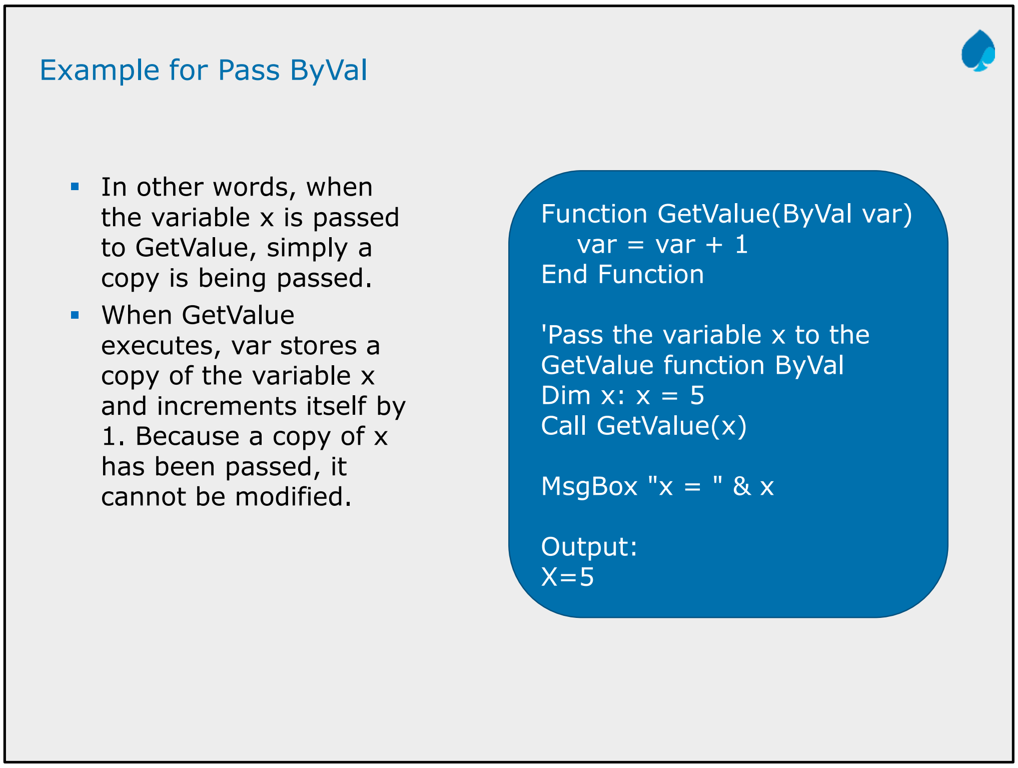


Passing by Reference Versus Passing by Value

- Pass a parameter by reference: any changes to the parameter inside the function or sub procedure will change the value of the original variable.
- Pass a parameter by value: the original variable retains its value after the function or sub procedure terminates, regardless of whether the corresponding parameter was changed inside the function or sub procedure.
- Syntax
  ```
  Sub SubName (ByVal / ByRef  Parameter)
  ```
- Default is pass by reference.

If **ByVa**l is specified, then the arguments are sent as byvalue when the function or procedure is called.
If **ByRef** is specified, then the arguments are sent as a reference when the function or procedure is called

Instructor Notes:



Example for Pass ByVal

- In other words, when the variable x is passed to GetValue, simply a copy is being passed.
- When GetValue executes, var stores a copy of the variable x and increments itself by 1. Because a copy of x has been passed, it cannot be modified.

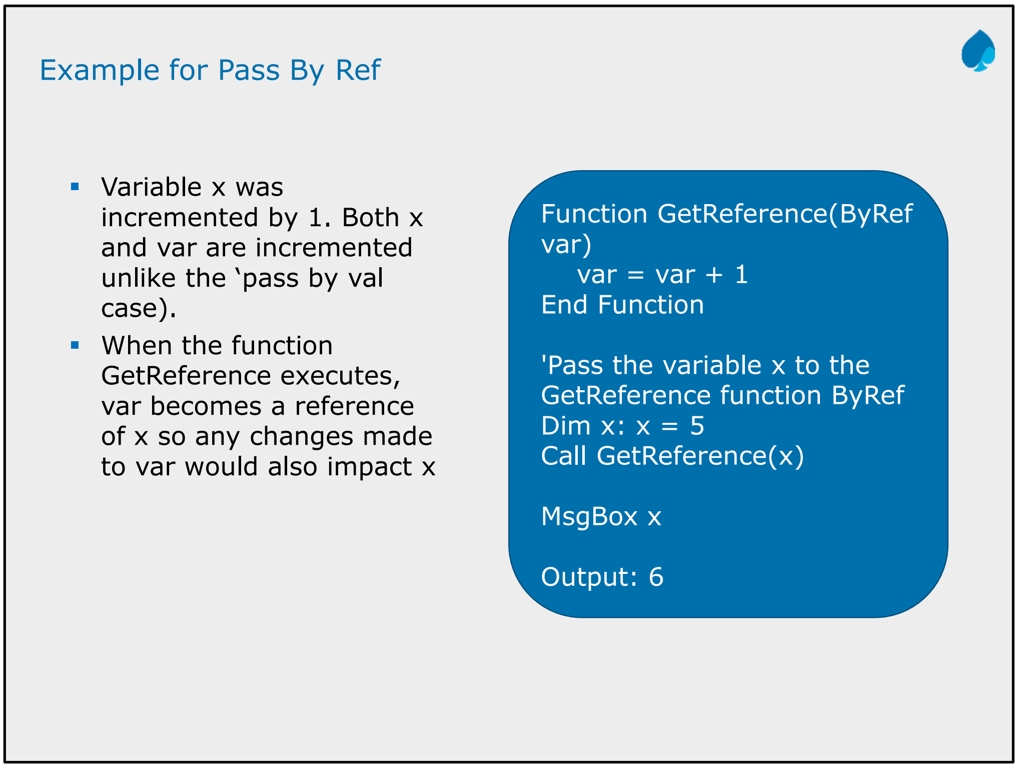
```
Function GetValue(ByVal var)
    var = var + 1
End Function

'Pass the variable x to the
GetValue function ByVal
Dim x: x = 5
Call GetValue(x)

MsgBox "x = " & x

Output:
X=5
```

Instructor Notes:



Example for Pass By Ref

- Variable x was incremented by 1. Both x and var are incremented unlike the 'pass by val case).
- When the function GetReference executes, var becomes a reference of x so any changes made to var would also impact x

```
Function GetReference(ByRef
var)
    var = var + 1
End Function

'Pass the variable x to the
GetReference function ByRef
Dim x: x = 5
Call GetReference(x)

MsgBox x

Output: 6
```

Instructor Notes:

Additional notes for instructor

Demo

Sub Procedures
- procedure_withoutCall

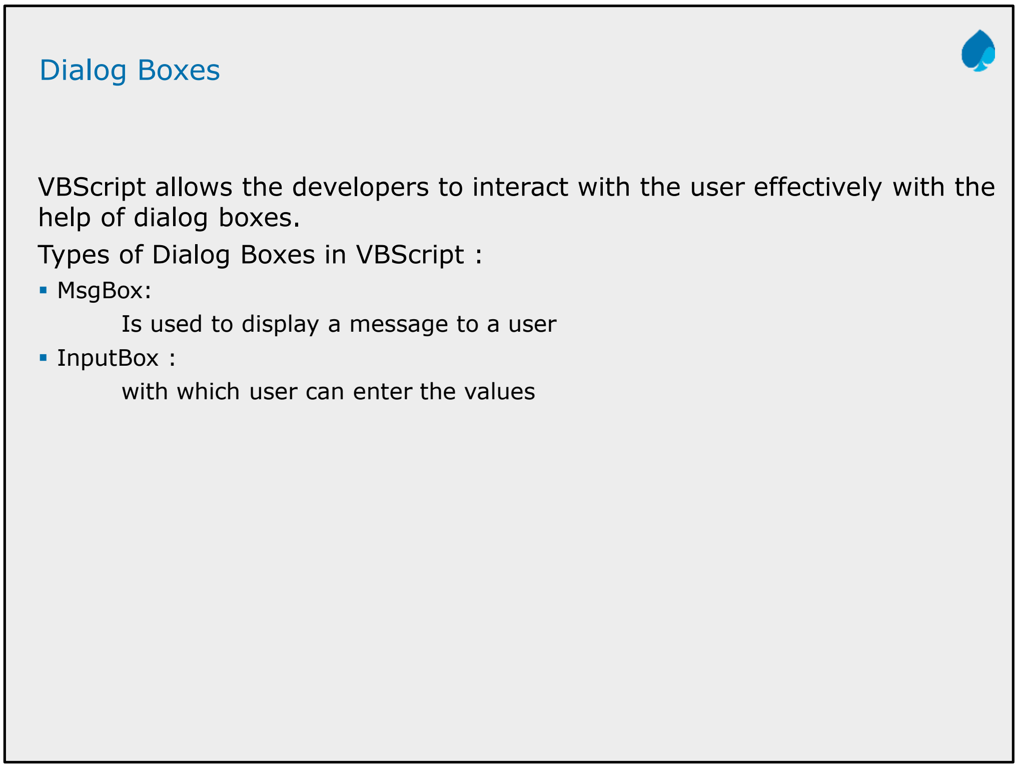
Functions
- Function
- function_return

Passby Value
- passbyvalue

PassBy Reference
- passbyreference

Add the notes here.

Instructor Notes:



Dialog Boxes

VBScript allows the developers to interact with the user effectively with the help of dialog boxes.

Types of Dialog Boxes in VBScript :

- MsgBox:
  - Is used to display a message to a user
- InputBox :
  - with which user can enter the values

Instructor Notes:

Dialog Box- Message Box

The MsgBox function displays a message box and waits for the user to click a button and then an action is performed based on the button clicked by the user.

Syntax:

```
MsgBox(prompt[,buttons][,title])
```

Example:

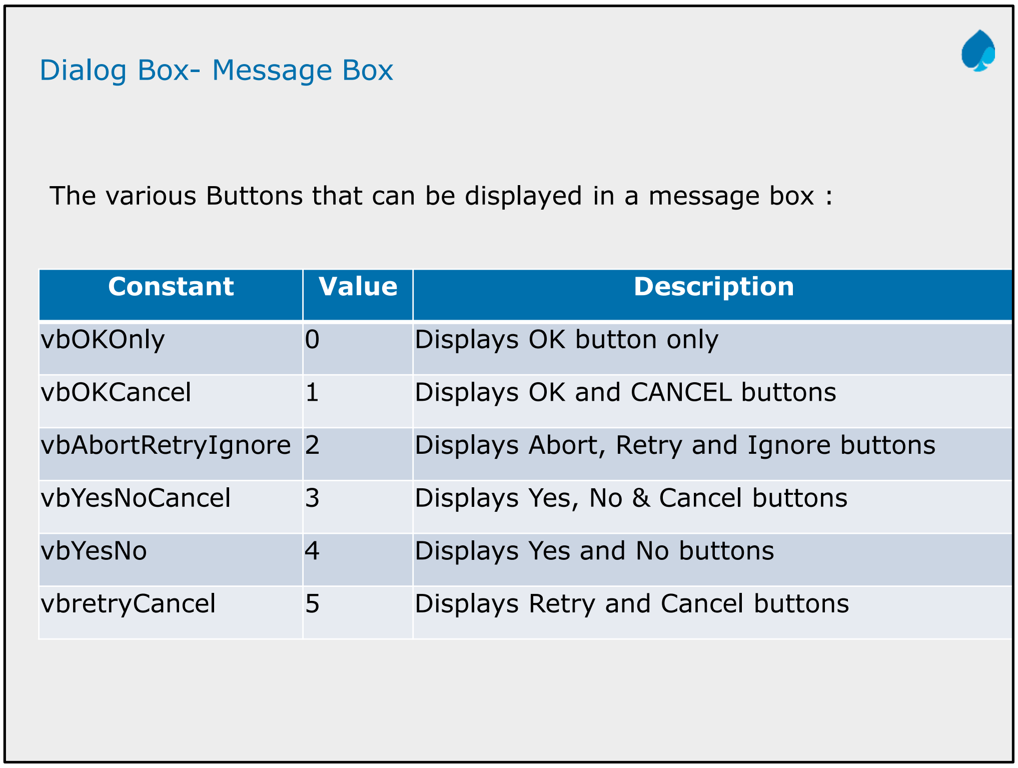
```
X = MsgBox(“message”, vbYesNo)

        If X = vbYes Then
             MsgBox “The YES button was clicked”
        Else
                MsgBox “The NO button was clicked”
        End if
```

Parameter explaination:

Prompt – It is a required parameter. A String that is displayed as a message in the dialog box. The maximum length of prompt is approximately 1024 characters. If the message extends to more than a line, then we can separate the lines using a carriage return character (Chr(13)) or a linefeed character (Chr(10)) between each line.
buttons – It is an optional parameter. A Numeric expression that specifies the type of buttons to display, the icon style to use, the identity of the default button, and the modality of the message box. If left blank, the default value for buttons is 0.
Title – It is an optional parameter. A String expression displayed in the title bar of the dialog box. If the title is left blank, the application name is placed in the title bar

Instructor Notes:

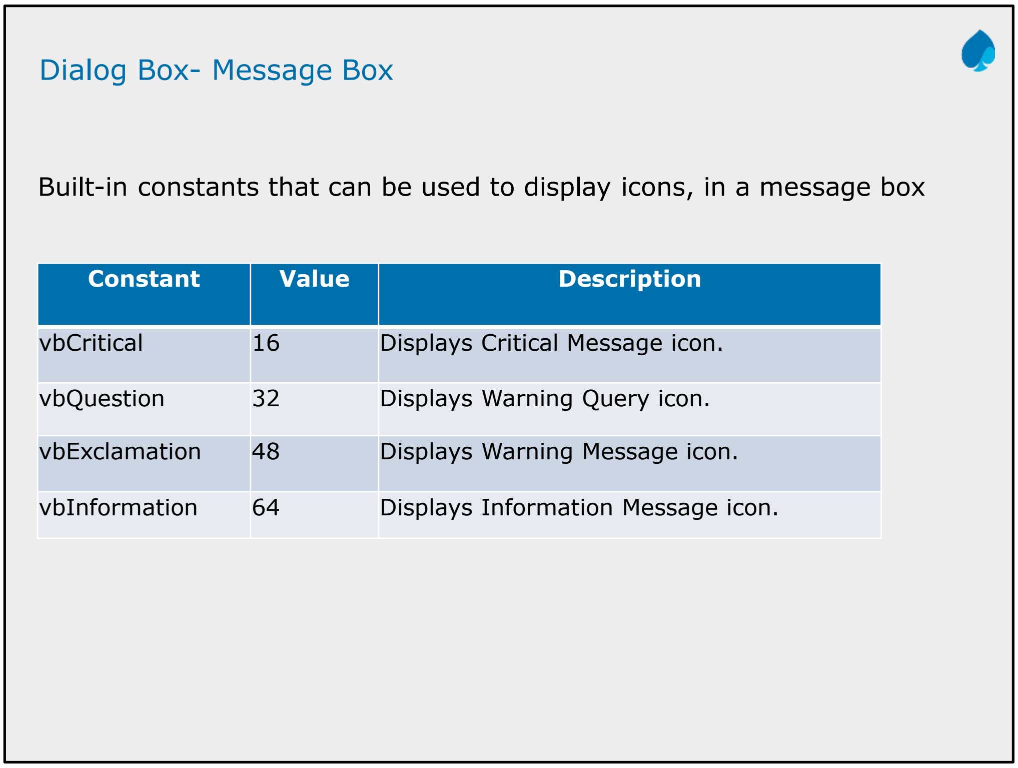


Dialog Box- Message Box

The various Buttons that can be displayed in a message box :

Constant	Value	Description
vbOKOnly	0	Displays OK button only
vbOKCancel	1	Displays OK and CANCEL buttons
vbAbortRetryIgnore	2	Displays Abort, Retry and Ignore buttons
vbYesNoCancel	3	Displays Yes, No & Cancel buttons
vbYesNo	4	Displays Yes and No buttons
vbretryCancel	5	Displays Retry and Cancel buttons

Instructor Notes:

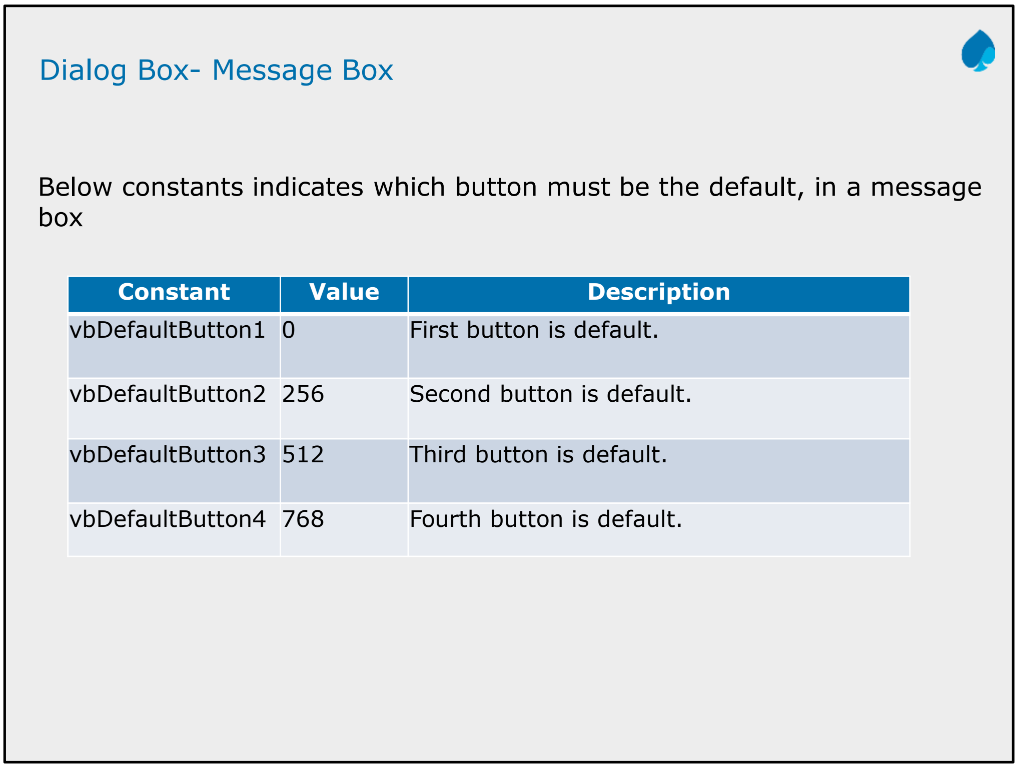


Dialog Box- Message Box

Built-in constants that can be used to display icons, in a message box

Constant	Value	Description
vbCritical	16	Displays Critical Message icon.
vbQuestion	32	Displays Warning Query icon.
vbExclamation	48	Displays Warning Message icon.
vbInformation	64	Displays Information Message icon.

Instructor Notes:

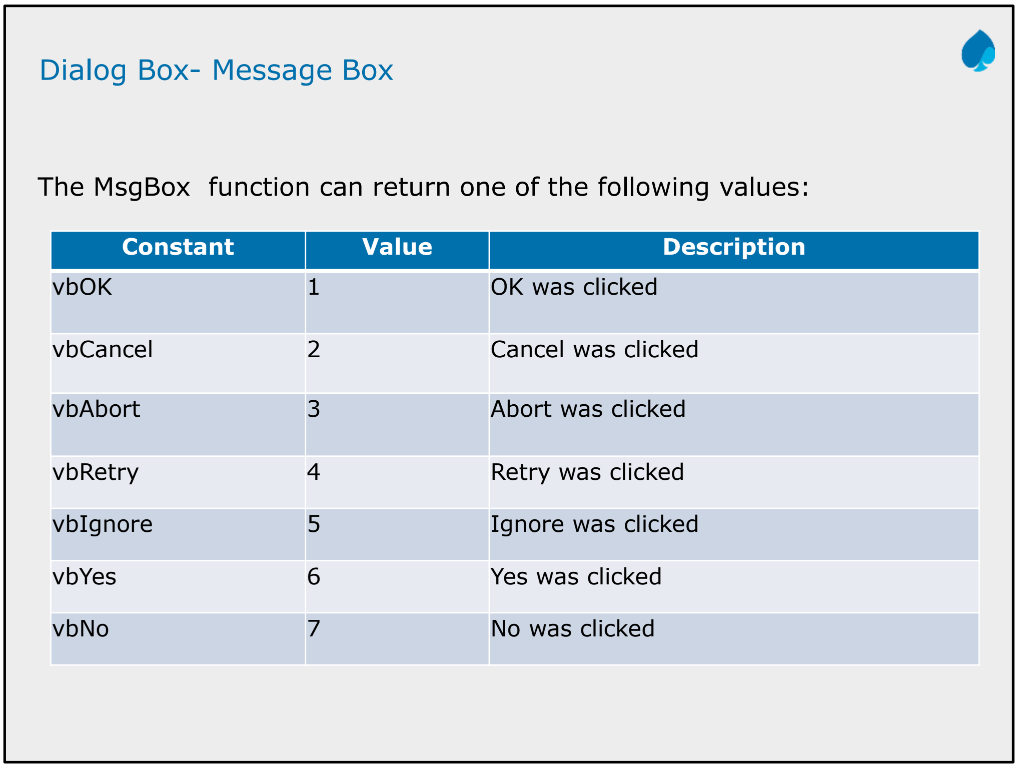


Dialog Box- Message Box

Below constants indicates which button must be the default, in a message box

Constant	Value	Description
vbDefaultButton1	0	First button is default.
vbDefaultButton2	256	Second button is default.
vbDefaultButton3	512	Third button is default.
vbDefaultButton4	768	Fourth button is default.

Instructor Notes:

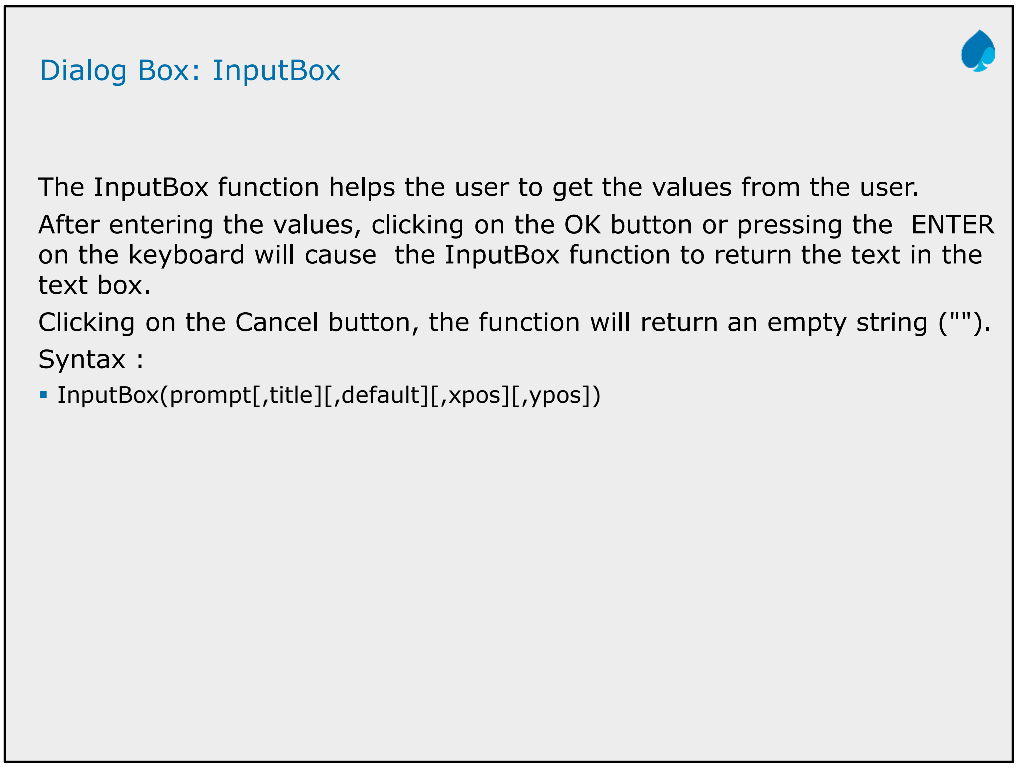


Dialog Box- Message Box

The MsgBox function can return one of the following values:

Constant	Value	Description
vbOK	1	OK was clicked
vbCancel	2	Cancel was clicked
vbAbort	3	Abort was clicked
vbRetry	4	Retry was clicked
vbIgnore	5	Ignore was clicked
vbYes	6	Yes was clicked
vbNo	7	No was clicked

Instructor Notes:



Dialog Box: InputBox

The InputBox function helps the user to get the values from the user.

After entering the values, clicking on the OK button or pressing the ENTER on the keyboard will cause the InputBox function to return the text in the text box.

Clicking on the Cancel button, the function will return an empty string ("").

Syntax :

- InputBox(prompt[,title][,default][,xpos][,ypos])

Prompt - A Required Parameter. A String that is displayed as a message in the dialog box. The maximum length of prompt is approximately 1024 characters. If the message extends to more than a line, then we can separate the lines using a carriage return character (Chr(13)) or a linefeed character (Chr(10)) between each line.

Title - An Optional Parameter. A String expression displayed in the title bar of the dialog box. If the title is left blank, the application name is placed in the title bar.
Default - An Optional Parameter. A default text in the text box that the user would like to be displayed.

XPos - An Optional Parameter. The Position of X axis which represents the prompt distance from left side of the screen horizontally. If left blank, the input box is horizontally centered.

YPos - An Optional Parameter. The Position of Y axis which represents the prompt distance from left side of the screen Vertically. If left blank, the input box is Vertically centered.

Instructor Notes:

Additional notes for instructor

Demo

Message Box

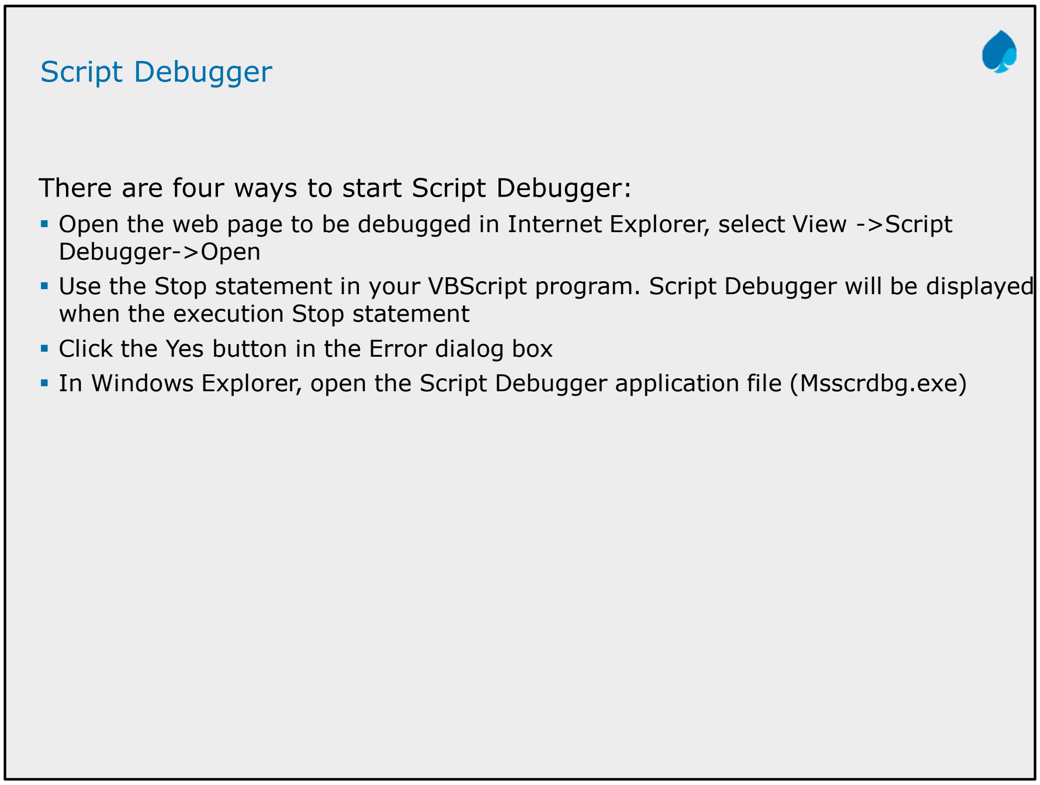
- messageBox

Input box

- InputBox

Add the notes here.

Instructor Notes:



Script Debugger

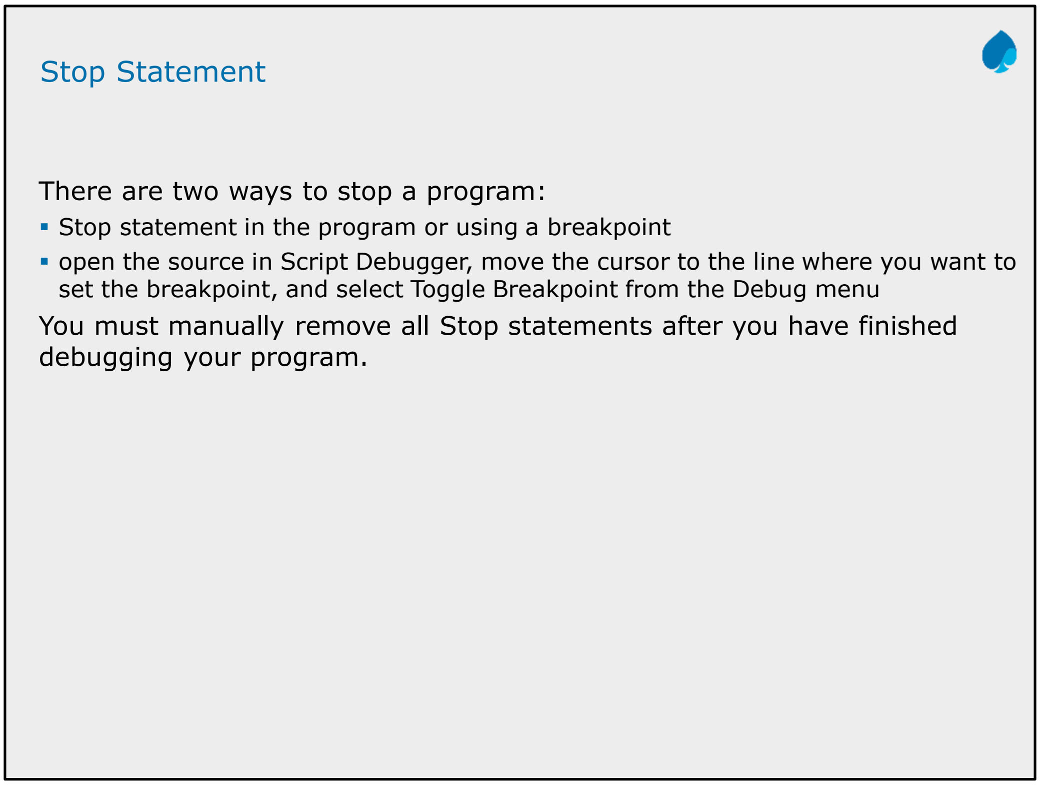
There are four ways to start Script Debugger:

- Open the web page to be debugged in Internet Explorer, select View ->Script Debugger->Open
- Use the Stop statement in your VBScript program. Script Debugger will be displayed when the execution Stop statement
- Click the Yes button in the Error dialog box
- In Windows Explorer, open the Script Debugger application file (Msscrdbg.exe)

Or you can put a shortcut to this file on your desktop, which is a lot easier to ge to.

In all but the last case, you will be working with the Web page that is currently displayed in Internet Explorer. In the last case, you can open the source for an existing Web page by selecting Open from Script Debugger's File menu, or you can create a new page from scratch by selecting New from the File menu. Once Script Debugger is running, you can debug any page that is currently running in Internet Explorer by choosing Running Documents from the View menu.

Instructor Notes:



Stop Statement

There are two ways to stop a program:

- Stop statement in the program or using a breakpoint
- open the source in Script Debugger, move the cursor to the line where you want to set the breakpoint, and select Toggle Breakpoint from the Debug menu

You must manually remove all Stop statements after you have finished debugging your program.

There are two ways to stop a program:

- placing a Stop statement in the program.
- Using a breakpoint.

When Internet Explorer reaches a Stop statement, it stops executing the script at that point. The other way is to open the source in Script Debugger, move the cursor to the line where you want to set the breakpoint, and select Toggle Breakpoint from the Debug menu.

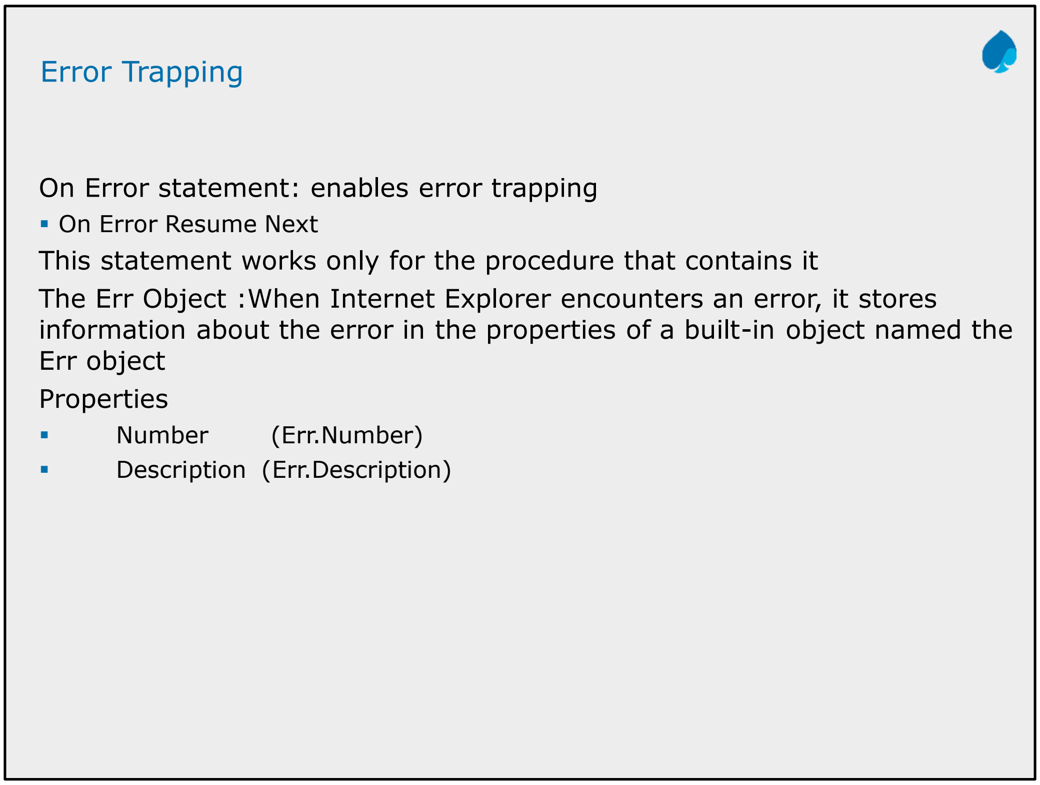
Unfortunately, clicking the Refresh button in Internet Ex plorer cancels the breakpoint in Script Debugger. For this reason, It is preferred to use Stop statements a lot more than breakpoints.

You must manually remove all Stop statements after you have finished debugging your program.

Now let's assume that we used Notepad to add a Stop statement right after the statements that get the input. Here's what the code would look like:

Option Explicit
Dim I, A(4), Average
For I = 1 To 4
A(I) = InputBox("What is the student's grade on exam " _ & I & "?") Next Stop I would save the file, click the Refresh button, and enter four 100s again. When Internet Explorer encounters the Stop statement, it automatically opens Script Debugger.

Instructor Notes:



Error Trapping

On Error statement: enables error trapping

- On Error Resume Next

This statement works only for the procedure that contains it

The Err Object :When Internet Explorer encounters an error, it stores information about the error in the properties of a built-in object named the Err object

Properties

- Number (Err.Number)
- Description (Err.Description)

Error Trapping

The On Error statement, which enables error trapping in VBScript, looks like the following: On Error Resume Next This tells Internet Explorer that if it encounters an error to simply forget about it and move to the next statement. This statement works only for the procedure that contains it. This means that you might want to include an On Error statement in each procedure in your program. If you don't do this and an error occurs in a called procedure, Internet Explorer will move to the statement following the call (not to the next statement in the called procedure).

The Err Object

When Internet Explorer encounters an error, it stores information about the error in the properties of a built-in object named the Err object. The value of the Description property is a string describing the error, and the value of the Number property is an integer representing the error. For example, if a user attempts to divide by 0, the Description property would be set to "Division by zero" and the Number property would be set to 11.
If you do use error trapping in a procedure you always add a fragment similar to the one.

Instructor Notes:

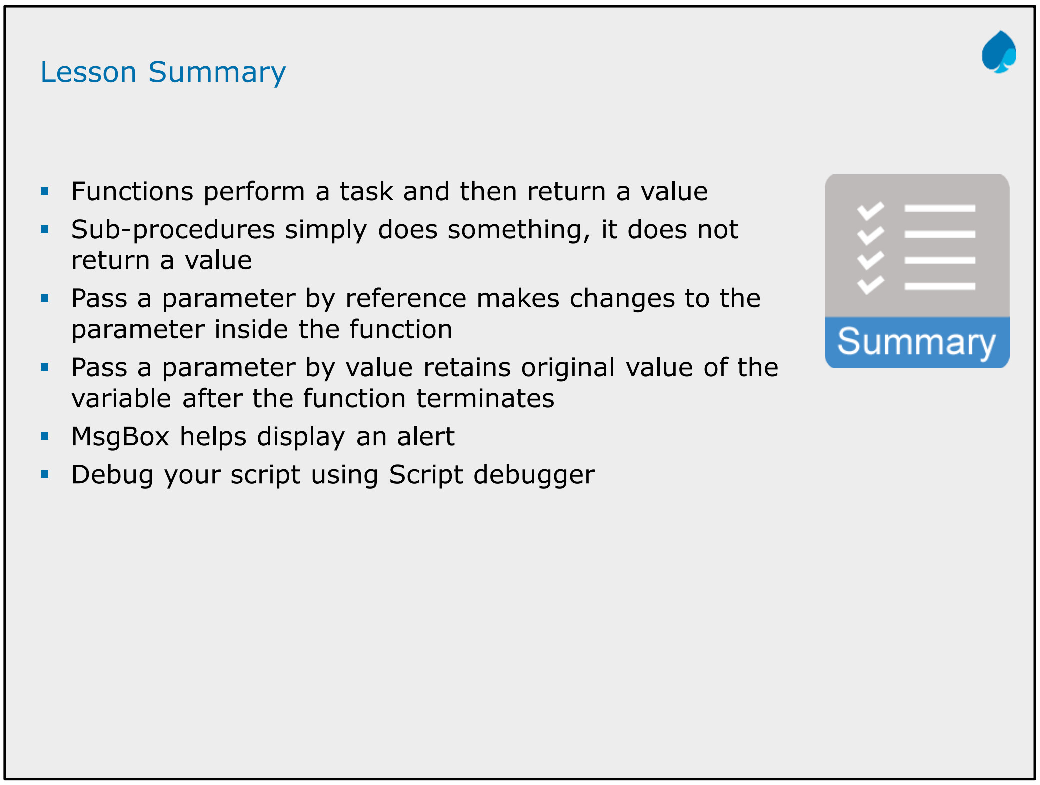
Additional notes for instructor

Lab

Lab

Instructor Notes:

Additional notes for instructor



Lesson Summary

- Functions perform a task and then return a value
- Sub-procedures simply does something, it does not return a value
- Pass a parameter by reference makes changes to the parameter inside the function
- Pass a parameter by value retains original value of the variable after the function terminates
- MsgBox helps display an alert
- Debug your script using Script debugger

Instructor Notes:

1. function
2. stop
3. Err object

Review - Questions

- Which of the following returns a value?
  - Option 1: function
  - Option 2: Sub routine
  - Option 3: procedure
  - Question 2: While debugging a program the execution stops at ________ statement.
- Question 3: Error information is stored in _________.