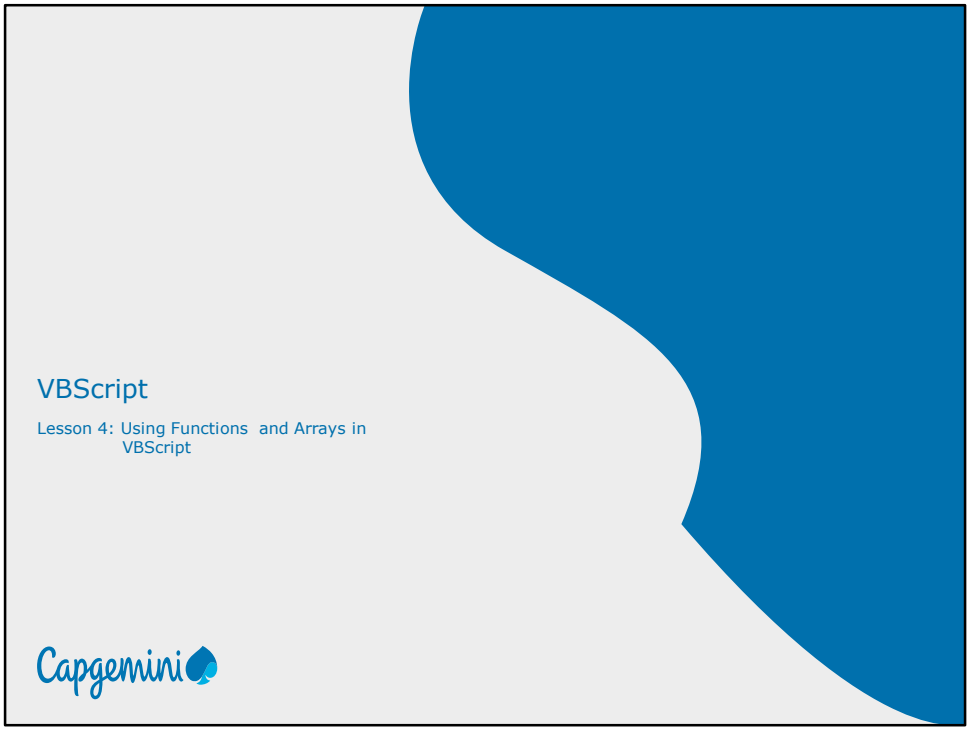
Instructor Notes:

Add instructor notes here.



Instructor Notes:

Explain the lesson coverage

Lesson Objectives

- Built In Functions: Date/Time Functions
- Built In Functions: Math Functions
- Built In Functions: String Functions
- Built In Functions: Format Functions
- Built In Functions: Other Functions
- Built In Functions: Conversion Functions
- Arrays
- Dynamic Arrays
- Array Functions
- Multidimensional Arrays
- Converting Variables to Arrays

Instructor Notes:

Additional notes for instructor

Functions and Arrays

Built In Functions
- Date and Time Functions
- Math Functions
- String Functions
- Format Functions
- Other Functions

Arrays
- Array Functions
- Dynamic Arrays
- Multidimensional arrays

Add the notes here.

Instructor Notes:

Built In Functions: Date/Time Functions

Function	Description	Syntax
Date	Returns the current system date	Date
DateAdd	Returns a date to which a specified time interval has been added	DateAdd(interval,number,date)
DateDiff	Returns the number of intervals between two dates	DateDiff(interval,date1,date2)
Day	Returns a number that represents the day of the month (between 1 and 31, inclusive)	Day(date)
FormatDateTime	Returns an expression formatted as a date or time	FormatDateTime(date,format)
Hour	Returns a number that represents the hour of the day (between 0 and 23, inclusive)	Hour(time)
IsDate	Returns a Boolean value that indicates if the evaluated expression can be converted to a date	IsDate(expression)
Minute	Returns a number that represents the minute of the hour (between 0 and 59, inclusive)	Minute(time)

Please note only those parameters mentioned in square brackets are optional.

Explanation of some of the parameters:

- **DateAdd function**
  - the ‘interval’ parameter can take the following values:
    - yyyy - Year
    - q - Quarter
    - m - Month
    - y - Day of year
    - d - Day
    - w - Weekday
    - ww - Week of year
    - h - Hour
    - n - Minute
    - s - Second
  - The ‘number’ parameter is the number of interval you want to add. Can either be positive, for dates in the future, or negative, for dates in the past
  - The ‘date’ parameter is the Variant or literal representing the date to which interval is added

Instructor Notes:

Built In Functions: Date/Time Functions

Function	Description	Syntax
Month	Returns a number that represents the month of the year (between 1 and 12, inclusive)	Month(date)
MonthName	Returns the name of a specified month	MonthName(month[,abbreviate])
Now	Returns the current system date and time	Now
Second	Returns a number that represents the second of the minute (between 0 and 59, inclusive)	Second(time)
Time	Returns the current system time	Time
Weekday	Returns a number that represents the day of the week (between 1 and 7, inclusive)	Weekday(date[,firstdayofweek])
WeekdayName	Returns the weekday name of a specified day of the week	WeekdayName(weekday[, abbreviate[,firstdayofweek]])
Year	Returns a number that represents the year	Year(date)

- **Weekday and WeekdayName Function:**
  - The parameter 'firstdayofweek' is Optional. It specifies the first day of the week. It can take the following values:
    - 0 = vbUseSystemDayOfWeek - Use National Language Support (NLS) API setting
    - 1 = vbSunday - Sunday (default)
    - 2 = vbMonday - Monday
    - 3 = vbTuesday - Tuesday
    - 4 = vbWednesday - Wednesday
    - 5 = vbThursday - Thursday
    - 6 = vbFriday - Friday
    - 7 = vbSaturday - Saturday
- **MonthName and weekdayName Function:**
  - The 'abbreviate' parameter is optional. It can take a boolean value that indicates if the weekday name is to be abbreviated

Instructor Notes:

Additional notes for instructor

Demo

Date functions

- datefunctions_1
- datefunctions_2
- datefunctions_3
- datefunctions_4

Add the notes here.

Instructor Notes:

Built In Functions: Math Functions

Function	Description	Syntax
Abs	Returns the absolute value of a specified number	Abs(number)
Exp	Returns e raised to a power	Exp(number)
Int	Returns the integer part of a specified number	Int(number)
Log	Returns the natural logarithm of a specified number	Log(number)
Rnd	Returns a random number less than 1 but greater or equal to 0	Rnd[(number)]
Sgn	Returns an integer that indicates the sign of a specified number	Sgn(number)
Sqr	Returns the square root of a specified number	Sqr(number)
Abs	Returns the absolute value of a specified number	Abs(number)

Instructor Notes:

Additional notes for instructor

Demo

Math functions

- Mathfunctions

Add the notes here.

Instructor Notes:

Built In Functions: String Functions

Function	Description	Syntax
InStr	Returns the position of the first occurrence of one string within another. The search begins at the first character of the string	InStr([start,]string1,string2[,compare])
InStrRev	Returns the position of the first occurrence of one string within another. The search begins at the last character of the string	InStrRev(string1,string2[,start[,compare]])
LCase	Converts a specified string to lowercase	LCase(string)
Left	Returns a specified number of characters from the left side of a string	Left(string,length)

Explanation of some of the parameters:

➢ **For the Functions InStr and InStrRev :**

- The parameter 'start' is Optional. It Specifies the starting position for each search. The search begins at the first character position (1) by default. This parameter is required if compare is specified.
- The parameter 'string1' is required. This is the string to be searched.
- The parameter 'string2' is required. This is the string expression to search for .
- The 'compare' parameter is optional. It specifies the string comparison to use. Default is 0.

  It can have one of the following values:

  - 0 = vbBinaryCompare - Perform a binary comparison
  - 1 = vbTextCompare - Perform a textual comparison

•If string1 is "" - InStr and InStrRev returns 0
•If string1 is Null - InStr and InStrRev returns Null
•If string2 is "" - InStr and InStrRev returns start
•If string2 is Null -InStr and InStrRev returns Null
•If string2 is not found - InStr and InStrRev returns 0
•If string2 is found within string1 - InStr and InStrRev returns the position at which match is found
•If start > Len(string1) - InStr and InStrRev returns 0

Instructor Notes:

Built In Functions: String Functions

Function	Description	Syntax
Len	Returns the number of characters in a string	Len(string)
LTrim	Removes spaces on the left side of a string	LTrim(string)
RTrim	Removes spaces on the right side of a string	RTrim(string)
Trim	Removes spaces on both the left and the right side of a string	Trim(string)
Mid	Returns a specified number of characters from a string	Mid(string,start[,length])
Replace	Replaces a specified part of a string with another string a specified number of times	Replace(string,find,replace with[,start[,count[,compare]]])

➢ **For the Replace Function:**

- The ‘string’ parameter is required. It is the string to be searched.
- The ‘find’ parameter is required. This is the part of the string that will be replaced.
- The ‘replacewith’ parameter is required. It is the replacement substring.
- The ‘start’ parameter is optional. It specifies the start position. Default is 1. All characters before the start position will be removed.
- The ‘count’ parameter is also optional. It specifies the number of substitutions to perform.Default value is -1, which means make all possible substitutions
- The ‘compare’ parameter is optional. It specifies the string comparison to use. Default is 0

  It can have one of the following values:
  - 0 = vbBinaryCompare - Perform a binary comparison
  - 1 = vbTextCompare - Perform a textual comparison

Instructor Notes:

Built In Functions: String Functions

Function	Description	Syntax
Right	Returns a specified number of characters from the right side of a string	Right(string,length)
Space	Returns a string that consists of a specified number of spaces	Space(number)
StrComp	Compares two strings and returns a value that represents the result of the comparison	StrComp(string1,string2[,compare])
String	Returns a string that contains a repeating character of a specified length	String(number,character)
StrReverse	Reverses a string	StrReverse(string)
UCase	Converts a specified string to uppercase	UCase(string)

The StrComp function can return one of the following values:

- -1 (if string1 < string2)
- 0 (if string1 = string2)
- 1 (if string1 > string2)
- Null (if string1 or string2 is Null)

Instructor Notes:

Additional notes for instructor

Demo

String functions

- stringfunctions_1
- stringfunctions_2
- stringfunctions_3

Add the notes here.

Instructor Notes:

Built In Functions: Format Functions

Function	Description	Syntax
FormatCurrency	Returns an expression formatted as a currency value	FormatCurrency(Expression[, NumDigAfterDec[,IncLeadingDig[, UseParForNegNum[,GroupDig]]]])
FormatDateTime	Returns an expression formatted as a date or time	FormatDateTime(date,format)
FormatNumber	Returns an expression formatted as a number	FormatNumber(Expression[, NumDigAfterDec[,IncLeadingDig[, UseParForNegNum[,GroupDig]]]])
FormatPercent	Returns an expression formatted as a percentage	FormatPercent(Expression[, NumDigAfterDec[,IncLeadingDig[, UseParForNegNum[,GroupDig]]]])

Vbscript FormatCurrency function provides the functionality to format the simple number
expression into currency formatted numbers with grouping, specified number of digits after decimal point, leading zero for fractional numbers before decimal point etc.
You can also format the negative currency value numbers using vbscript FormatCurrency
function.

Syntax:
FormatCurrency (expression, NumDigitsAfterDecimal,IncludeLeadingDigit, UseParensForNegativeNumbers, GroupDigits)

FormatCurrency accepts 5 types of parameters as shown in the above syntax:

1. expression: first parameter accepts number expression that is to formatted into currency format.
2. NumDigitsAfterDecimal: accepts the number value to specify the number of digits after decimal points.
3. IncludeLeadingDigit: accepts 3 types of values to specify whether to display leading zero for fractional decimal numbers. e.g.: 0.9 or .9
 IncludeLeadingDigit accepts following values:
 -2 : Use the computer default regional settings.
 -1 : True
 0 : False
4. UseParensForNegativeNumbers: accepts 3 types of values to specify whether to display parenthesis () for negative numbers or not. E.g.: -9 or (9)
 UseParensForNegativeNumbers accept following values:
 -2 : Use the computer default regional settings.
 -1 : True

Instructor

o : False

Instructor Notes:

Additional notes for instructor

Demo

Format functions

- FormatCurrencyDemo
- FormatDateTimeDemo
- FormatNumberDemo
- FormatPercentDemo

Add the notes here.

Instructor Notes:

Built In Functions: Other Functions

Function	Description	Syntax
Eval	Evaluates an expression and returns the result	Eval(expression)
IsEmpty	Returns a Boolean value that indicates whether a specified variable has been initialized or not	IsEmpty(expression)
IsNull	Returns a Boolean value that indicates whether a specified expression contains no valid data (Null)	IsNull(expression)
IsNumeric	Returns a Boolean value that indicates whether a specified expression can be evaluated as a number	IsNumeric(expression)

Instructor Notes:

Additional notes for instructor

Demo

Other functions

- otherfunctions

Add the notes here.

Instructor Notes:

Arrays

- Dim SalesInMonth (11)
- This Dim statement would tell VBScript to set aside 12 slots in memory
- Those slots would be identified as:

```
SalesInMonth(0) = some number    'for sales in January
SalesInMonth(1) = some number    'for sales in February
```

- The SalesInMonth array is an example of a fixed array

In general, a list (or one-dimensional array) is a way to group items so that you can refer to each item individually by means of its index. The index is simply the number you place inside the parentheses; it is the position of the item in the array.
The name of the list has to follow VBScript's rules for variable names. You tell VBScript that you will be working with a list by using a variation of the Dim statement that you have already seen. The only difference is that you need to add the parentheses with the maximum index. For example, the following statement could be used to identify the sales for a year: Dim SalesInMonth(11)
In general, a list (or one-dimensional array) is a way to group items so that you can refer to each item individually by means of its index. The index is simply the number you place inside the parentheses; it is the position of the item in the array. The name of the list has to follow VBScript's rules for variable names. You tell VBScript that you will be working with a list by using a variation of the Dim statement that you have already seen. The only difference is that you need to add the parentheses with the maximum index. For example, the following statement could be used to identify the sales for a year:

```
Dim SalesInMonth(11)
```

This Dim statement would tell VBScript to set aside 12 slots in memory. Those slots would be identified as:

```
SalesInMonth(0) = some number    'for sales in January
SalesInMonth(1) = some number    'for sales in February
. .
SalesInMonth(11) = some number   'for sales in December
```

Instructor Notes:

Dynamic Arrays

- Dynamic array whose size can change inside any event procedure
- Dim gThingsToDo()
- To change the size of the array each time the user adds or removes an item. This can be done with the ReDim statement:
- ReDim gThingsToDo(lstBox1.ListCount - 1)
- To know the current upper bound, if array size keeps changing. It is done with the UBound function
- For I = 0 To UBound(AnArray)

Dynamic Versus Fixed Arrays

The SalesInMonth array is an example of a fixed array. That's because we fixed its size in the Dim statement. However, you might need an array whose size can change while the program is running. This is called a dynamic array. To create a dynamic array whose size you can change inside any event procedure, simply use parentheses without the maximum index and make sure that the array is a script-level variable Dim gThingsToDo( )

Now suppose, for example, we want to activate the Web page and we want to store the contents of the text-box in an array. We need to change the size of the array each time the user adds or removes an item. This can be done with the

ReDim statement:
ReDim gThingsToDo(lstBox1.ListCount - 1)

When you start enlarging or shrinking the size of an array, it becomes vital (in For-Next loops, for example) to have a way of knowing the current upper bound. This is done with the UBound function. A For-Next loop whose code starts like this will go through all the elements in the array:

For I = 0 To UBound(AnArray)

Instructor Notes:

Array Functions

Function	Description	Syntax
Array	Returns a variant containing an array	Array(arglist)
Filter	Returns a zero-based array that contains a subset of a string array based on a filter criteria	Filter(inputstrings,value[,include[, compare]])
IsArray	Returns a Boolean value that indicates whether a specified variable is an array	IsArray(variable)
Join	Returns a string that consists of a number of substrings in an array	Join(list[,delimiter])

➢ **For the Filter Function:**

- The ‘inputstrings’ parameter is required. It is one-dimensional array of strings to be searched
- The ‘value’ parameter is required. It is the string to search for
- The ‘include’ parameter is optional. A Boolean value that indicates whether to return the substrings that include or exclude value. True returns the subset of the array that contains value as a substring. False returns the subset of the array that does not contain value as a substring. Default is True.
- The ‘compare’ parameter is optional. Specifies the string comparison to use.

  It can have one of the following values:
  - 0 = vbBinaryCompare - Perform a binary comparison
  - 1 = vbTextCompare - Perform a textual comparison

➢ **For The Join Function:**

- The ‘list’ parameter is required. A one-dimensional array that contains the substrings to be joined
- The ‘delimiter’ parameter is optional. The character(s) used to separate the substrings in the returned string. Default is the space character

➢ **For the LBound and UBound Functions:**

- The LBound for any dimension is ALWAYS 0.
- We can use the LBound function with the UBound function to determine the size of an array

Instructor Notes:

Array Functions

Function	Description	Syntax
LBound	Returns the smallest subscript for the indicated dimension of an array	LBound(arrayname[,dimension])
UBound	Returns the largest subscript for the indicated dimension of an array	UBound(arrayname[,dimension])
Split	Returns a zero-based, one-dimensional array that contains a specified number of substrings	Split(expression[,delimiter[,count[,compare]]])

➢ **For the Spilt Function:**

- The ‘expression’ parameter is required. A string expression that contains substrings and delimiters
- The ‘delimiter’ parameter is optional. A string character used to identify substring limits. Default is the space character
- The ‘count’ parameter is optional. The number of substrings to be returned. -1 indicates that all substrings are returned
- The ‘compare’ parameter is optional. Specifies the string comparison to use. It can have one of the following values:
    - 0 = vbBinaryCompare - Perform a binary comparison
    - 1 = vbTextCompare - Perform a textual comparison

Instructor Notes:

Additional notes for instructor

Demo

Array functions

- arrayfunctions

Add the notes here.

Instructor Notes:

Multidimensional Arrays

Two-dimensional array for the multiplication tables:

Dim MultiplicationTable(9,9)

This sets aside 10 rows and 10 columns for a total of 100 lot

Multidimensional Arrays

Just as lists lead to one-dimensional arrays, tables lead to multidimensional arrays. For example, to make a two-dimensional array for the multiplication tables, you would write this: Dim MultiplicationTable(9,9)

This sets aside 10 rows and 10 columns for a total of 100 slots. (Remember that arrays start with a slot at position 0.).

To actually fill the table, use a nested For-Next loop:

```
For I = 0 To 9
   For J = 0 To 9
         MultiplicationTable(I,J) = (I+1) * (J+1)
    Next
 Next
```

Instructor Notes:

Converting Variables to Arrays

We can assign an array to a variable without parentheses
Then treat the variable just like an ordinary array

```
Dim A
A = Array(1, 2, 3, 4)
```

Now A(0) gives you 1, A(1) gives you 2

Converting Variables to Arrays

One of VBScript's most amazing abilities regarding arrays is that you can assign an array to a variable that doesn't use parentheses in its definition, and then treat the variable just like an ordinary array. VBScript provides two ways to do this. The first is with the Array function, which simply turns a bunch of data into an array:

```
Dim A
A = Array(1, 2, 3, 4)
```

Now A(0) gives you 1, A(1) gives you 2, and so on. You can also declare an array and then simply assign it to another variable:

```
Dim ATable, Multiplic ationTable(9,9)
ATable = MultiplicationTable
```

You can even use a variable to temporarily hold an array in order to "swap" two arrays.
Although List property of a list box is an array, this is true only in a restricted sense.
It works exactly like an array as far as element access (via the index), but you cannot assign the List property of a list box to a variable.

Instructor Notes:

Additional notes for instructor

Lab

Lab 2
Lab 3
Lab 4

Add the notes here.

Instructor Notes:

Additional notes for instructor

Lesson Summary

- String functions: Lcase, Ucase, Lan, Mid,Replace, InStr
- Numeric functions: Int, Round, Sgn, Abs
- Date Time functions: IsDate(), IsNumeric()
- Array is a way to group items so that you can refer to each item individually by means of its index
- Functions for Parsing and Building Strings: Join, Split, Filter

Add the notes here.

Instructor Notes:

1. InStr
2. -1
3. Redim

Review - Questions

To search a string or a part of string we can use:

- Option 1: Find
- Option 2: InStr
- Option 3: InStrRev

Question 2: Sgn function returns _____ for negative number.

Question 3: _____ is used to change the size of an array dynamically.

Add the notes here.