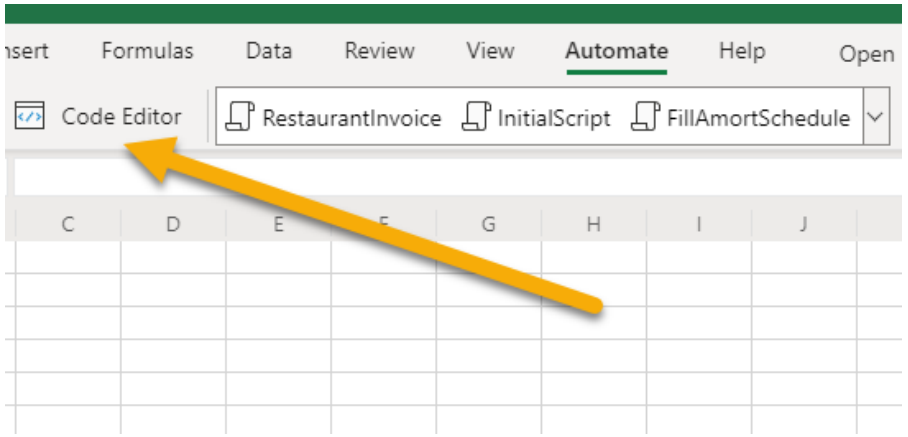
Adding New Code to an Office Script

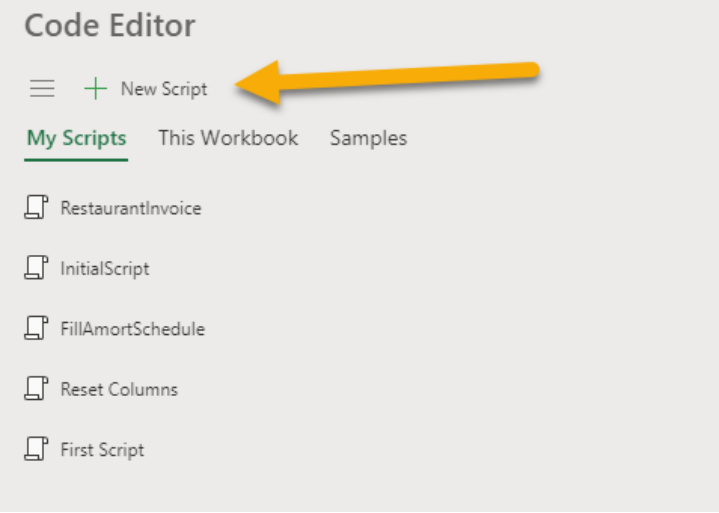
Brent Allen
FREELANCE CONSULTANT

@Macrordinary_CA http://macrordinary.ca

Getting Started With a New Script



Enter into the Code Editor



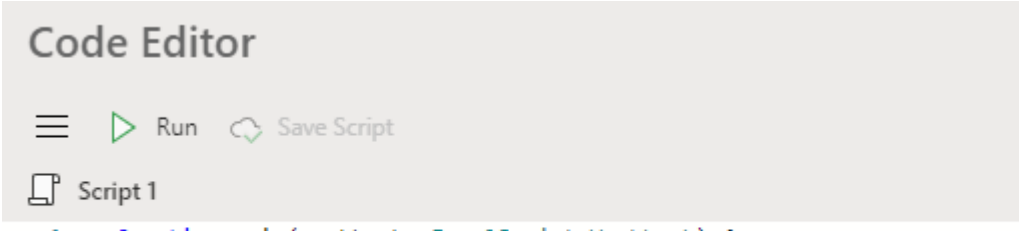
Click New Script

Structure of a Blank Script

Script name is Script 1

Function is named main

Workbook is defined



For the rest of the module,
no more code recorder!

Comparing Variables in OfficeScript to VBA

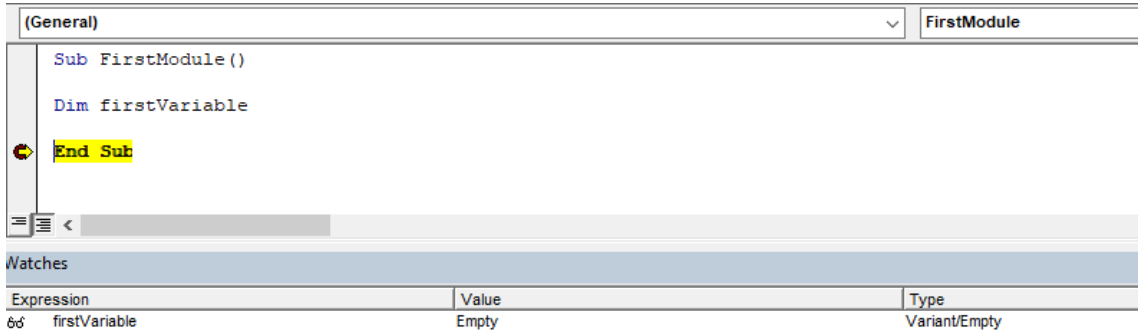
If you're comfortable with JavaScript or TypeScript, skip to Assigning Sheets and Ranges!

Unassigned Type Variables

VBA

Can be Dim, Private or Public

Defaults to the Variant type

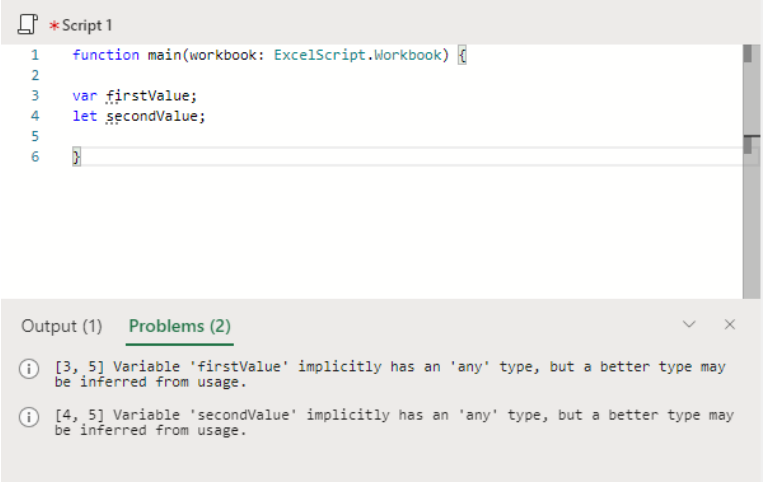


Office Script / TypeScript

Declared with let (alternatively, var)

Defaults to the ‘any’ variable type

Shows up as a problem in the log



Declared Type Variables

VBA

Use the ‘As’ keyword to assign

Can be variable types such as String, Long Integer, Double Float

```
Sub FirstModule()

Dim firstVariable As String
Dim secondVariable As Double

End Sub
```

Watches

Expression	Value	Type
firstVariable	""	String
secondVariable	0	Double

Office Script / TypeScript

Use a colon to assign

Normally are number, string or boolean

Script 1

```
function main(workbook: ExcelScript.Workbook) {

let firstValue: string;
let secondValue: number;

}
```

Output (1) Problems

Assigning Objects

VBA

First declare variable

Assigned using the ‘Set’ keyword

```
Sub FirstModule()

Dim currentWorksheet As Worksheet
Set currentWorksheet = ThisWorkbook.Sheets("Sheet1")

End Sub
```

Watches

Expression	Value	Type
currentWorksheet		Worksheet/Sheet1

Office Script / TypeScript

Variable declared on same line

Assigned using equals sign

Script 1

```
function main(workbook: ExcelScript.Workbook) {

let currentWorksheet = workbook.getWorksheet("Sheet1")

}
```

Arrays

VBA

Created using curved brackets ()

Number of elements must be specified

Elements can be changed using ReDim

```
Sub FirstModule()

Dim groupOfNumbers() As Integer
ReDim groupOfNumbers(3)
groupOfNumbers(3) = 4
groupOfNumbers(2) = 3
groupOfNumbers(1) = 2
groupOfNumbers(0) = 1

End Sub
```

Office Script / TypeScript

Created using square brackets []

Number of elements are not specified

Array functions to add / remove data

Script 1

```
function main(workbook: ExcelScript.Workbook) {

let groupOfNumbers: number[];
groupOfNumbers = [4,3,2,1];

}
```

Referring to Array Elements

VBA

One based

First sheet is Sheets(1)

Range A1 is at index (1,1)

```
Sub Main()
    ThisWorkbook.Sheets(1).Cells(1, 1) = 1
End Sub
```

Office Script / TypeScript

Zero-based

The first spreadsheet is element[0]

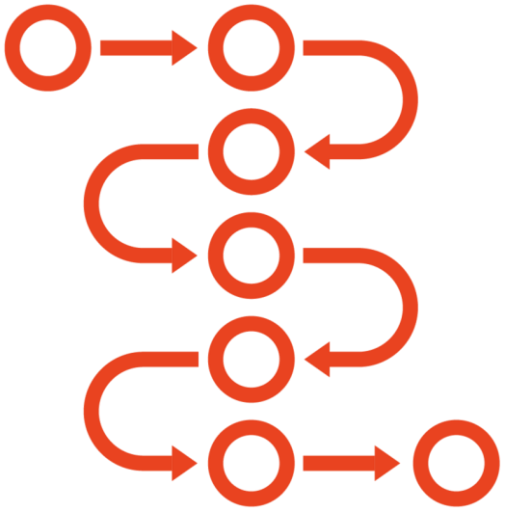
Range A1 is at index [0,0]

```
function main(workbook: ExcelScript.Workbook) {
    workbook.getWorksheets()[0].getRangeByIndexes(0,0,1,1).setValue(1)

}
```

Program Flow – Office Script vs. VBA

Program Flow



Iterative Loops (for, for each)

Conditional Statements (if, else if, else)

Two Different Types of Iterative Loops

Used for iteration based on a single number

Used for iteration through elements of an array or collection

Iteration – The For Loop

VBA

Variable does not have to be declared.

Step of 1 is implied.

For loop ends with a Next statement.

```
Sub main()

For x = 1 To 10
    Cells(x, 1).Value = x
Next

End Sub
```

Office Script

Variable must be declared.

Step count must be stated.

For loop uses curled brackets.

```
function main(workbook: ExcelScript.Workbook) {
let activeSheet = workbook.getWorksheet("Sheet1");

    for (let x = 1; x <= 10; x++) {
        activeSheet.getRangeByIndexes(x-1,0,1,1).setValue(x);
    }

}
```

Iteration Through An Array Or Collection

VBA

Syntax is similar to the for loop syntax

Iterator (Sheet) type is defined by the collection (ActiveWorkbook.Sheets)

```
Sub main()

For Each Sheet In ActiveWorkbook.Worksheets
    Sheet.Range("A1") = 6

Next

End Sub
```

Office Script

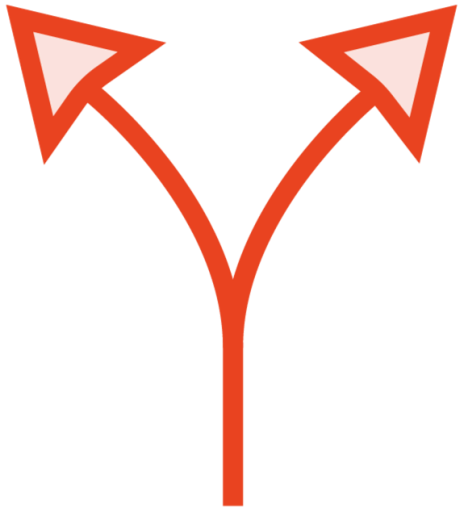
Syntax is very different than for loop syntax

Iterator (sheet) is an argument to the forEach method

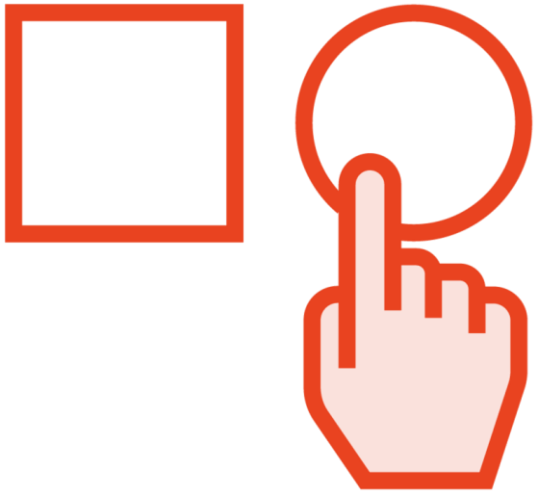
Arguments are in a function

```
function main(workbook: ExcelScript.Workbook) {
    workbook.getWorksheets().forEach((sheet) =>
        {
            sheet.getRange("A1").setValue(6)
        }
    )
}
```

Conditional Statements



If statement – code runs if condition is met



Else if statement – code runs if a secondary condition is met

Else – code runs if no conditions are met

Conditional Statements

VBA

Checks using a single equal sign

Blocks end with the next statement (an else or else if) or an end if

```
Sub main()

Dim checkVariable As Integer
checkVariable = 2

If checkVariable = 1 Then
    Range("A1") = "First Hit"
ElseIf checkVariable = 2 Then
    Range("A1") = "Second Hit"
End If

End Sub
```

Office Script

Checks using a double or triple equal sign

Blocks are in curled brackets

```
function main(workbook: ExcelScript.Workbook) {
    let activeSheet = workbook.getWorksheet("Sheet1");
    let checkVariable = 2;

    if (checkVariable == 1) {
        activeSheet.getRange("A1").setValue("First Hit");
    }
    else if (checkVariable == 2) {
        activeSheet.getRange("A1").setValue("Second Hit");
    }

}
```

Assigning Sheets and Ranges

Setting a Value for a Range

Microsoft® Visual Basic® for Applications

```
Sub Main()
    Range("A1") = 1
End Sub
```

TS

```
function main(workbook: ExcelScript.Workbook) {
    getRange("A1").setValue(1)

}
```

TS

```
function main(workbook: ExcelScript.Workbook) {
    workbook.getActiveWorksheet().getRange("A1").setValue(1)

}
```

Setting Worksheet Variables

getWorkbook – sets a variable by sheet name

getWorksheets – sets a variable by index

getActiveWorksheet – sets a variable based on the active worksheet

```
function main(workbook: ExcelScript.Workbook) {
    let outputSheetbyName = workbook.getWorksheet("Sheet1");
        // Refer by name.
    let outputSheetbyIndex = workbook.getWorksheets()[0];
        // Refer by index.
    let outputActiveSheet = workbook.getActiveWorksheet();
        // Refer by active sheet.

}
```

Adding a Worksheet

Sheet can be added without assigning to a variable

Like VBA Add method

Adding name is optional

Always added at the end

```
workbook.addWorksheet();
let thirdSheet = workbook.addWorksheet();
let fourthSheet = workbook.addWorksheet("Sheet No4");
```

Setting Range Variables

There are 6 different methods:

getSelectedRange()

getRange()

getRanges()

getRangeByIndexes()

getCell()

getUsedRange()

```
function main(workbook: ExcelScript.Workbook) {
    let outputSheetbyName = workbook.getWorksheet("Sheet1");
        // Refer to sheet by name.

    let selectedRange = workbook.getSelectedRange();
        // Get the selected range (does not use worksheet)
    let singleRange = outputSheetbyName.getRange("A1:A2");
        // Gets a single range on the worksheet.
    let multiRange = outputSheetbyName.getRanges("A4:A5,C4:C5");
        // Gets multiple ranges on the worksheet.
        // Not all methods can be used!
    let indexedRange = outputSheetbyName.getRangeByIndexes(4,6,2,1);
        // Gets a range specified by index coordinates.
    let indexedCell = outputSheetbyName.getCell(3,5);
        // Gets a single cell specified by index coordinates.
    let wholeRange = outputSheetbyName.getUsedRange();
        // Gets the range encompassing all values and formatting on a sheet.
}
```

getSelectedRange()

```
let selectedRange = workbook.getSelectedRange();
    // Get the selected range (does not use worksheet)
```

Uses only the workbook to get the range.

Cannot refer to any worksheet.

getRange()

```
let singleRange = outputSheetbyName.getRange("A1:A2");
    // Gets a single range on the worksheet.
```

Uses a worksheet where the range is stored.

Range must be contiguous.

getRanges()

```
let multiRange = outputSheetbyName.getRanges("A4:A5,C4:C5");
    // Gets multiple ranges on the worksheet.
```

,

Ranges must be separated by a comma.

Not all methods can be applied to this range.

getRangeByIndexes()

```
let indexedRange = outputSheetbyName.getRangeByIndexes(4,6,2,1);
    // Gets a range specified by index coordinates.
```

$\begin{bmatrix} 4 & 7 \\ 1 & 5 \end{bmatrix}$

Index is row, column, # of rows, # of columns.

[0]

Row and columns are zero indexed (cell A1 is 0,0)

getCell()

```
let indexedCell = outputSheetbyName.getCell(3,5);
    // Gets a single cell specified by index coordinates.
```

Single Cell - Index is row and column

[0]

Row and column are zero indexed (cell A1 is 0,0)

getUsedRange()

```
let wholeRange = outputSheetbyName.getUsedRange()
    // Gets the range encompassing all values and formatting.
```

Range is from top left cell with data to bottom right cell.

Will return error if no data on sheet.

Changing Properties For a Range

Properties In VBA vs. Office Script

VBA

Uses equal sign to assign value

Assign value directly to property

```
Sub main()

Dim rngTestValue As Range
Set rngTestValue = Sheets("Sheet1").Range("A1")

rngTestValue.Value = "Test Value"
rngTestValue.Borders(xlEdgeBottom).ColorIndex = 0

End Sub
```

Office Script

Identify properties using get functions

Assigns value to a property using a set function

```
function main(workbook: ExcelScript.Workbook) {
  let selectedSheet = workbook.getActiveWorksheet();
  // Set range A1 on selectedSheet
  selectedSheet.getRange("A1").setValue("Test Value");
  // Set border for range Sheet1!A1
  selectedSheet.getRange("A1")
    .getFormat()
    .getRangeBorder(ExcelScript.BorderIndex.edgeBottom)
    .setColor("black");

}
```

Value Set Methods

Type of Adjustment	Method	Example
Same value for whole range.	.setValue()	getRange(“A1:A3”) .setValue(2)
Different values for every cell	.setValues()	getRange(“C1:D2”) .setValues([[1,2],[3,4]])
Same formulas for whole range	.setFormula() or .setFormulaLocal()	getRange(“G1”) .setFormula(“=sum(A1:A3)”)
Different formulas for every cell	.setFormulas() or .setFormulasLocal()	getRange(“H2:I2”) .setFormulas([[“=sum(A1:A3”), [“=sum(C1:D2)”]])

.setValue() and .setFormula() Example

setValue – the same number is entered in each cell

setFormula – the formula is not absolute unless specified

```
function main(workbook: ExcelScript.Workbook) {
    let outputSheetbyName = workbook.getWorksheet("Sheet1");
        // Refer to sheet by name.
    let singleRange = outputSheetbyName.getRange("A1:A3");
    singleRange.setValue(1);

    let differentCell = outputSheetbyName.getRange("A4:A5");
    differentCell.setValue(5);

    let sumRangeRelative = outputSheetbyName.getRange("C1:C3");
    sumRangeRelative.setFormula("=SUM(A1:A2)")

    let sumRangeAbsolute = outputSheetbyName.getRange("C4:C5");
    sumRangeAbsolute.setFormula("=SUM($A$1:$A$2)")

    let formulaCheckRange = outputSheetbyName.getRange("D1:D5")
    formulaCheckRange.setFormula("=FORMULATEXT(C1)")

}
```

	A	B	C	D	E
1	1		2	=SUM(A1:A2)	
2	1		2	=SUM(A2:A3)	
3	1		6	=SUM(A3:A4)	
4	5		2	=SUM($A$1:$A$2)	
5	5		2	=SUM($A$1:$A$2)	
6					

.setValues() Example

Takes an array of arrays

Each row is an array

Columns are comma separated

A value must be entered for every cell

```
function main(workbook: ExcelScript.Workbook) {
    let outputSheetbyName = workbook.getWorksheet("Sheet1");
        // Refer to sheet by name.
    let multipleRange = outputSheetbyName.getRange("A1:B4");
    multipleRange.setValues([
        [1,2],
        ["",null],
        [3,4],
        ["five","six"]
    ]);
}
```

	A	B
1	1	2
2		
3	3	4
4	five	six

Formatting Methods

Type of Adjustment	Method	Example
Change number format	.setNumberFormat()	.getRange(“A1:A3”) .setNumberFormat(“$0.00”)
Change bold formatting	.getFormat() .getFont() .setBold()	.getRange(“A1:A3”) .getFormat().getFont() .setBold(true)
Add border to bottom of cell	.getFormat() .getRangeBorder() .setColor()	.getRange(“A1:A3”) .getFormat() .getRangeBorder(ExcelScript.BorderIndex. edgeBottom) .setColor("black")

Demo

Returning to the “Winger Place”

Copy transactions from Order tab to the Invoice tab

Apply specific formatting

Unlike the previous example, the number of rows needs to be dynamic!

Summary

- Created and assigned variables
- Iterated through array / sheet elements
- Used multiple methods for getting ranges
- Changed properties of a range
- Wrote Office Script without recorder