Essential Apps Script livecoding 1 walkthrough: My First Apps Script

This walkthrough guide accompanies the Essential Apps Script 1: My First Apps Script video and the Essential Apps Script guide. You can use the video, this document, or both, to help you do this livecoding exercise.

We're going to have a go at writing our first bit of Apps Script code. All you're going to need is a Google account that you can access Google Drive with (if you're a member of the University of York, use your UoY Google account).

For this exercise, we're going to have a go at using Apps Script to get some data out of a spreadsheet and tell us what that data is.

Livecoding instructions

1. Firstly, you're going to need to make sure you're logged into your Google account and have opened Google Drive. Make sure you're not logged into multiple Google accounts at the same time as this can affect things later on.
    a. Create a folder in Google Drive for the Essential Apps Script course so you can keep all your files relating to the course in the same place.
2. We're going to create a spreadsheet as we are going to create an Apps Script project that is **bound** to a Google Sheets file and use our code to access data in the spreadsheet. In your Google Drive folder, use the **New** button and choose **Google Sheets** to create a new file.
3. When your new file opens, rename it something sensible, like 'My First Apps Script'. It doesn't matter what you call it, as long as it makes sense to you.
4. In your Sheets file, write something in cell **B2** (If you're not sure what to write, you could write 'Hello', or 'Hello World' which is typically used for a first coding project).
5. Now we're going to open the Apps Script editor. At the menu at the top of Sheets, **Extensions** > **Apps Script**. It will open in a new window, but leave the spreadsheet open - if you close that spreadsheet, the Apps Script window will also close.
6. Click on the project name ('Untitled project' by default) to rename it - typically it is helpful to give it the same name as the spreadsheet.
7. Next, we're going to rename the code file (code.gs) on the side. Click on the three dots and choose **Rename**. Call it 'firstScript' - you'll notice there's no spaces and it uses camel case, which means the first letter of any subsequent words are capitalised, like a hashtag, to make it easier to read. Click elsewhere in the script editor to save that change.

8. The final bit of renaming is to rename the function in the area where you can write Apps Script code. By default it is called 'myFunction', but change that to 'firstScript', making sure that you leave the word 'function' at the start and the round brackets () at the end (as those are an important part of the syntax). Now hit the **Save** icon (you have to save Apps Script files). Your code should now look something like:

```
function firstScript(){
}
```

9. Put your cursor at the end of line 1, after the brace (the curly looking bracket), and hit Enter a couple of times to get some space (you can use blank lines in Apps Script to space out your code and the computer will ignore them).

10. Now we're going to write a comment, which is information that the computer ignores, but humans can read. Do two forward slashes **//** and then write the comment 'get the sheet', so we know what the code that comes afterwards will do.

11. Hit Enter and then we're going to create our first variable, saving something in the computer's memory.

    a. Write **var** then a space and then give it a name. As we're going to be storing the spreadsheet in this variable, we are going to call it **ss** as that is the convention. Then do another space and an equals sign, to assign to variable a value

    b. We are to use the SpreadsheetApp, so write **SpreadsheetApp** making sure to capitalise the S and the A (the autosuggest will suggest this once you start typing it). Next, do a period **.** and immediately write **getActiveSpreadsheet()**. If you use the autosuggest feature to complete this it will not put the open and closed brackets at the end, so make sure you add that.

    c. Finally, finish your line with a semicolon **;**

```
function firstScript(){

    // get the sheet
    var ss = SpreadsheetApp.getActiveSpreadsheet();
}
```

12. Now, we're going to get the specific sheet of the spreadsheet (the tabs along the bottom of a Sheets file). Create another variable using **var** and give it the name **sheet** this time. Put an equals sign as before, and then write **ss** to access the spreadsheet and all the things we can do with it. Next, write a period **.** and then **getActiveSheet()** and then end the line with a semicolon **;**

```
function firstScript(){

  // get the sheet
```

```
    var ss = SpreadsheetApp.getActiveSpreadsheet();
    var sheet = ss.getActiveSheet();
}
```

13. Next, hit Enter twice more to leave another blank line if you'd like some space, then do another comment using two forward slashes **//** and write something like 'get the cell value and log it'.

14. Hit Enter again and now we're going to get the value from the cell that we wrote a message in.

    a. Write **var** to create a variable to store the value in, then give it the name **cellValue** (using camel case with the capital V to make it easier to read the two words without a space). Then, as before, a space, an equals sign **=** and another space.

    b. We're going to work with the sheet to access the range of cells within it, so write **sheet** then **.getRange()** (make sure you include the period at the start and the brackets at the end).

    c. To get the range, we need to put numbers inside the brackets, which are basically coordinates in the spreadsheet for the range we want. As we only want one cell, we just need to give it a row number and a column number. Cell B2 is on row 2 and column B, but it has to be a numerical column not a letter, so that is 2 as well. So inside the brackets write **2, 2** (the comma separating them is important).

    d. Now we have the range, but that isn't the value in the cell, just the cell itself, which can have other properties too, like the font or background colour. So do another period **.** and then write **getValue()** and end the line with a semicolon **;**

15. Finally, we want to log the value we now have stored in cellValue so we can check it got the right cell and did the right thing. Logging means showing a value on screen in the **Execution Log**. Hit Enter and then write **Logger.log()** - we're going to put our value to log inside the brackets, so write **cellValue** inside the brackets and then end the line with a semicolon. Your final code should look something like:

```
function firstScript(){

    // get the sheet
    var ss = SpreadsheetApp.getActiveSpreadsheet();
    var sheet = ss.getActiveSheet();

    // get the cell value and log it
    var cellValue = sheet.getRange(2, 2).getValue();
    Logger.log(cellValue);
}
```

16. Now hit save and run the code. The **Run** button at the top will run whichever function you have selected from the drop down list. As this is the first time you've run this Apps Script project, you will get an authorisation prompt, which is Google checking the permissions that your script might need to run. It should say that it can view, edit and delete your spreadsheets, but that doesn't mean the code you've written will - just that commands using the SpreadsheetApp could do that. Hit **Allow** and then your script will run.
17. When the script runs, you should see the execution log appear at the bottom of the screen. It will tell you the time the script starts and finishes running and it should also show you the logged value. If you see the correct value in there, then celebrate because your code worked! If not, double check your code against what we're got in this document and also check you've written a message in cell B2!

You're now ready to move on to the next part of section 1!

Reference: full code from exercise

```
function firstScript(){

    // get the sheet
    var ss = SpreadsheetApp.getActiveSpreadsheet();
    var sheet = ss.getActiveSheet();

    // get the cell value and log it
    var cellValue = sheet.getRange(2, 2).getValue();
    Logger.log(cellValue);
}
```