Essential Apps Script livecoding 2 walkthrough: Ongoing livecoding project (part 1)

This walkthrough guide accompanies the Essential Apps Script 2: Ongoing livecoding project (part 1) video and the Essential Apps Script guide. You can use the video, this document, or both, to help you do this livecoding exercise.

We are going to build up a project over three parts, livecoding together the bits of Apps Script needed to expand our project as we learn more features of Apps Script. For the first part, we are going to learn how to get the data from a row of a spreadsheet, based on which row the user is clicked on.

Livecoding instructions

1. Firstly, make sure you're in the Google Drive folder you've set up for the Essential Apps Script course, then create a new spreadsheet using the **New** button > **Google Sheets**.
2. Once your spreadsheet has opened, rename it to something sensible like 'Ongoing livecoding project 1'. There also needs to be some data in the spreadsheet to work with, so make two columns of data, one with the heading 'Name' and one with 'Shoe size'. Put a row of data in (it doesn't matter what the data is!).
3. Open the Apps Script editor from **Extensions** > **Apps Script** then rename the project to the same name as the spreadsheet.
4. Now, rename the script file to getRowData by clicking on the three dots next to the file name, then **Rename**. Then, change the function name from myFunction() to getRowData() to match.
5. Put your cursor at the end of the line (after the opening curly brace **{**) and hit Enter a couple of times to make some space, then press the save button. You should have something like:

```
function getRowData(){


}
```

6. The first part of the code is going to get the spreadsheet and the sheet. On a new line, create a comment using two forward slashes **//** and then write something like 'get the sheet' so we know what these lines are going to do.
7. Hit Enter so we can create our variable to store the spreadsheet in, as with any Apps Script code bound to a spreadsheet. As it is a variable, write **var** followed by a space, then **ss** because we are working with a spreadsheet, then another space

and an equals sign. Then, we will use the **SpreadsheetApp** and then the command **.getActiveSpreadsheet()** and then end the line with a semi colon.

8. Hit Enter so we can now create a variable to store the sheet in. Usee **var** to create a variable and call it **sheet** then equals sign **ss.getSheetByName()** so we can get a specific sheet. Inside the brackets put your sheet name inside single or double quotes, as it is a string (at this point, it can be useful to rename the sheet in your spreadsheet to **data** to make it easier to recognise than 'Sheet1'). End the line with a semicolon and save your file.

```
function getRowData(){

    // get the sheet
    var ss = SpreadsheetApp.getActiveSpreadsheet();
    var sheet = ss.getSheetByName('data');

}
```

9. Next, hit Enter twice to leave some space and then put another comment using two forward slashes **//** for 'get the active row number so we can use it' as that's what we're going to do next.

10. Hit Enter to start a new line and then create a new variable using **var** and call it **rowNumber** then do an equals sign **=** to assign it a value. We're going to use the sheet and then find the active cell, so it'll be **sheet.getActiveCell()**. We don't just want the active cell, we want the row it is on, so add a **.getRow()** to chain together another command and then end the line with a semicolon. Hit save.

```
function getRowData(){

    // get the sheet
    var ss = SpreadsheetApp.getActiveSpreadsheet();
    var sheet = ss.getSheetByName('data');

    // get the active row number so we can use it
    var rowNumber = sheet.getActiveCell().getRow();

}
```

11. Hit Enter twice to do our third and final comment. Two forward slashes then 'get the row data and log it', then hit Enter again.

12. Our final variable is going to be **rowData**, so **var rowData =** and then we're going to use the sheet again to get the range and then get the specific row and its values.

    a. Write **sheet.getRange()** and then we're going to put values inside the brackets to define our range.

    b. We are going to start with the row and column we want the range to start from, which in this case with be our variable **rowData** and the first column, **1**,

so it will be **rowData, 1** separated by a comma (and a space, just for readability).

c. Next, we can define how many rows and columns we want, also separated by commas. So we only want a single row, **1**, and then we want as many columns as we have in our data, so that's **2**.

d. As getRange only gets the range, not the values themselves, we now need to go outside of the brackets and add **.getValues()** making sure there's the plural 'values' because we have more than one cell.

e. When using getValues, our data is automatically put into a two-dimensional array, which means one array filled with other arrays. As we only have a single row, we only want the first array from within that overall array, so we can end our line with index 0 using **[0]** to ensure we only have one array to work with (see our arrays guidance and exercises for more on this numbering).

13. Finally, hit Enter to start a new line and then log the value of rowData using **Logger.log(rowData)** and end the line with a semicolon. Your code should look something like (without the line break when creating rowData, that's just the limits of the page size here):

```
function getRowData(){

    // get the sheet
    var ss = SpreadsheetApp.getActiveSpreadsheet();
    var sheet = ss.getSheetByName('data');

    // get the active row number so we can use it
    var rowNumber = sheet.getActiveCell().getRow();

    // get the row data and log it
    var rowData = sheet.getRange(rowNumber, 1, 1, 2).getValues()[0];
    Logger.log(rowData);

}
```

14. Hit Save, then go back into your spreadsheet and check you are clicked on your row of data before you run the code, as otherwise it won't give the right output. Use the **Run** button at the top of the script editor to run your code, go through the authorisation prompt because this is a new script project, and then your code should run.

15. Check the execution log to see if the data in your row appears. If not, double check your code, and also make sure you are clicked on that row in the spreadsheet before running the code.

You're now ready to move on to adding a menu in section 1 and then the first project!

Reference: full code from exercise

```
function getRowData(){

    // get the sheet
    var ss = SpreadsheetApp.getActiveSpreadsheet();
    var sheet = ss.getSheetByName('data');

    // get the active row number so we can use it
    var rowNumber = sheet.getActiveCell().getRow();

    // get the row data and log it
    var rowData = sheet.getRange(rowNumber, 1, 1, 2).getValues()[0];
    Logger.log(rowData);

}
```