Essential Apps Script livecoding 3 walkthrough: Calling functions

This walkthrough guide accompanies the Essential Apps Script 3: Calling functions video and the Essential Apps Script guide. You can use the video, this document, or both, to help you do this livecoding exercise.

For this exercise, we are going to learn how to create multiple functions and then 'call' one function inside another, which means you can run another function from within the first function. We are also going to see how you can pass a variable from one function to another.

Livecoding instructions

1. As before, go into your Google Drive folder for the course and create a new Google Sheets file with a sensible name. The file doesn't need anything in it, so you can go straight to **Extensions** > **Apps Script** and rename the Apps Script project to the same name as the file.

2. We are going to have two functions in a single script file. Typically you'd create functions in different script files on the left hand side of the script editor, to make it easier to differentiate them, but for this example, we're doing them both at once. So rename the default function from 'myFunction' to **function1** and then copy and paste the whole of that function (including the curly braces) underneath and then change the name of that one to **function2** so you have something like:

```
function function1(){


}

function function2(){


}
```

3. We are going to create a variable, in this case a colour, in function1 and then pass it to function2. So create a variable in function 1 using **var** and call it **colour** then put an equals sign **=** to assign it a value. As it will be a string, put single or double quotes and then type a colour inside the quotes, e.g. **'red'** or **"blue"**. Put a semicolon to end the line and hit Save.

```
function function1(){

    var colour = 'red';


}
```

```
function function2(){


}
```

4. Next, we need to call function2, by writing its name in function1, so that it will run when we want it to. So hit enter in function1 and copy and paste **function2()** including the round brackets onto that line. We want to send the variable colour to function2, so put **colour** inside the brackets and end the line with a semicolon. Hit Save.

```
function function1(){

    var colour = 'red';
    function2(colour);


}

function function2(){


}
```

5. Now, we need to make sure function2 expects to get the variable colour as well. Put **colour** inside the brackets after function2 where we define it (before the curly brace {) so function2 knows it will be sent a variable.

```
function function1(){

    var colour = 'red';
    function2(colour);


}

function function2(colour){


}
```

6. Finally, we are going to put a Logger inside function2 to show us the value of colour, so on a new line inside the curly braces for function2, put **Logger.log()** and then inside the brackets we'll put our variable to be logged. We're actually going to put some text as well to be clearer what it is, so inside your brackets put **'Colour is: '** and then a plus sign **+** to concatenate that string with the text in our variable, then **colour** as the name of our variable. Outside the brackets, end the line with a semicolon.

```
function function1(){
```

```
    var colour = 'red';
    function2(colour);


}

function function2(colour){

    Logger.log('Colour is: ' + colour);

}
```

7. To run our code, we need to make sure we have function1 selected, as that is the main or 'parent' function and if we run function2 we will get an error because it won't know what colour is as it hasn't been created. Once it is set to function1, hit the **Run** button. There won't be an authorisation prompt because we're not using any Google apps, but you should see your message in the Execution log.

You're now ready to move on to the next part of section 2!

Reference: full code from exercise

```
function function1(){

    var colour = 'red';
    function2(colour);


}

function function2(colour){

    Logger.log('Colour is: ' + colour);

}
```