**Office Scripts Scrabble ® Solver**

**Video Commentary**

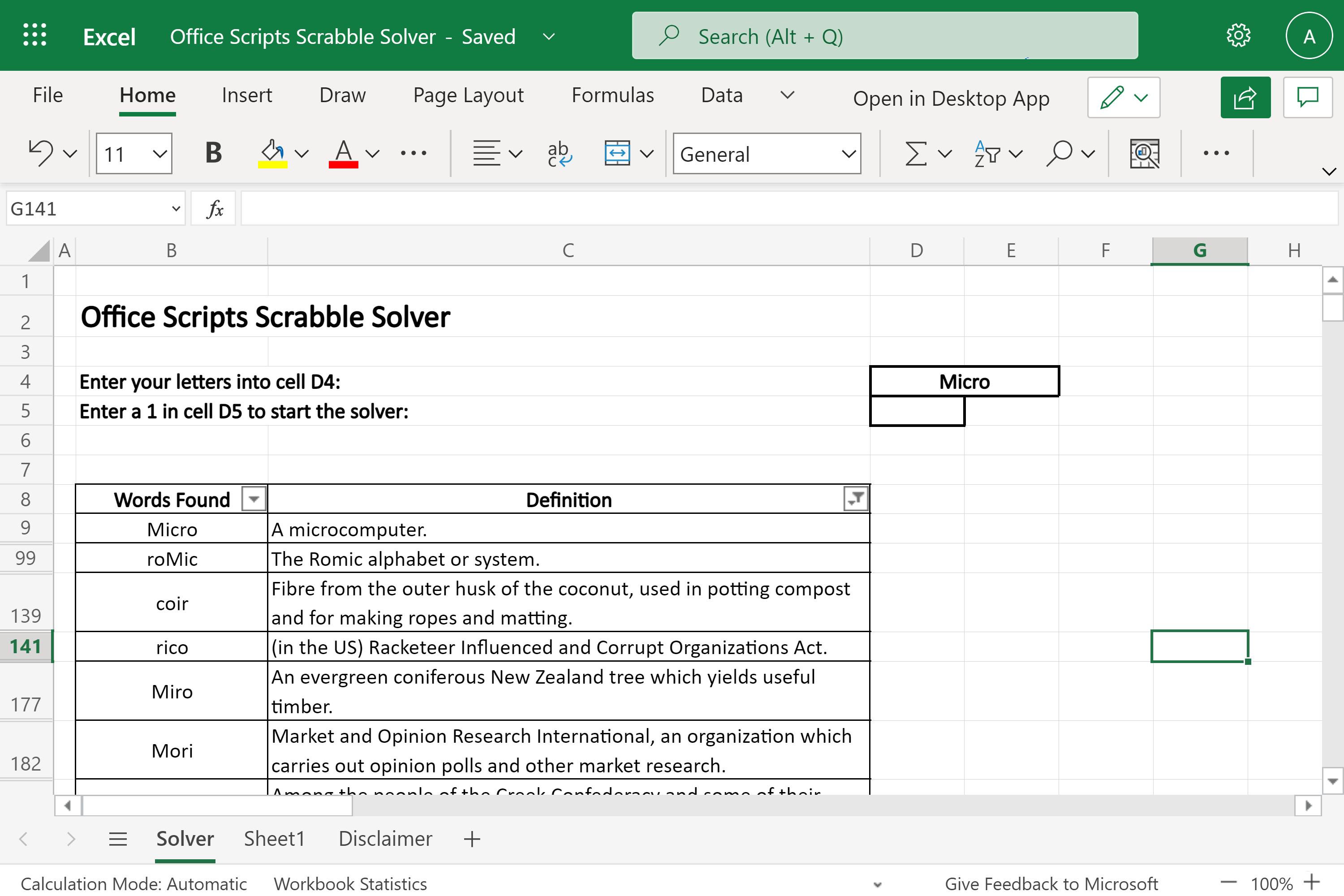
The Office Scripts Scrabble ® Solver uses the following Microsoft tools : Office Scripts, Power Automate, OneDrive for Business and Excel Online. Characters are added into cell D4. The solver finds all the anagrams, subwords and subsequent anagrams of all those subwords that have a English definition. A subword is any string of characters of any length derived from the original inputted string. For example if “microso” is added as the input string, one “subword” would be “mic”. The solver can be amended to solve for other languages, too.

Given medium strong players already know all the 2 letter words, this script, does not supply all the possible two letter words. So, the script ends at the three letter word level. The script generates all the possible word combinations for the characters supplied to it. The number of combinations possible from 7 characters is 5,568. For 6 characters it is 1,585.

In order to operate the Office Scripts Scrabble Solver enter a 1 in cell D5 to start a run. 4 character length strings take around 56 seconds to complete. 28 seconds of this is for the round-tripping to the dictionary API and back.

The scripts used in this demo are supplied separately as .ts files, in the provided GitHiub repo and are included at the end of the supplied User Guide, also.

Here we can see the result of the input word “micro” after a run. Only those words that have a definition are visible. All others have been filtered using the Excel filter functionality.



Let’s now go to the demo. For reasons of brevity I have removed the portions of the video were nothing happens showing only the word generation and the definition input.

**Video**

**Let’s now look at the Power Automate Flow behind the script.**

In reference to my previous demo solution, the JD Salinger Word Generator, the flow is different only in that the two Office Scripts are different and the final Filter step has been removed. The filtering, now, takes place, in Excel, operated by the script rather than in the Flow. This was changed because all of the generated words needed to be output back into the Excel workbook rather than removed prior to input. The flow is triggered when a file, in this case the Excel Scrabble Solver workbook, changes. An Office Script, then, runs. testing the length of the inputted string. If it is greater than 7 then the entire Flow terminates with a message posted to Teams. Next, the ScrabbleWordGenerator7 script runs. This generates the anagrams and sub-words for the provided input string. Next, various variables are declared. The *InputArray* variable is populated with the result of the previous Office Script. InputArray1 is populated with a subset of values from the *InputArray* variable. Each subset contains 1,999 words. The *DefnArray* variable holds the list of retrieved definitions for each of the generated words. The *HHTPReponse* variable contains the HTTP code 404 which “indicates that the browser was able to communicate with a given server, but the server could not find what was requested”.

**A picture containing application

Description automatically generated**

In the bottom half of the Flow a sequence of Apply to each loops are run depending on the size of the InputArray. Each Apply to each loop is the same dealing with the next 1,999 words.

Table

Description automatically generated

In the first “Apply to each” loop, the definitions for each of the generated words are retrieved using the HTTP connector and added to the DefnArray. The HTTP connector connects with a free Dictionary REST API from Dicionaryapi.dev. Where no definition is available the response, from the HTTP connector is 404. This value is contained within the status code response. In the condition step ‘statusCode’ is extracted from the retrned JSON and compared to the value 404. If true then the “No Word Exists” text is added as item (value) to the DefnArray. If the condition is false then a definition has been found and it is extracted from the HTTP Body response value.

**Graphical user interface, application, Teams

Description automatically generated**

The HTTP Connector is shown with the url displayed. The array item is appended at the end.

Graphical user interface, text, application, email

Description automatically generated

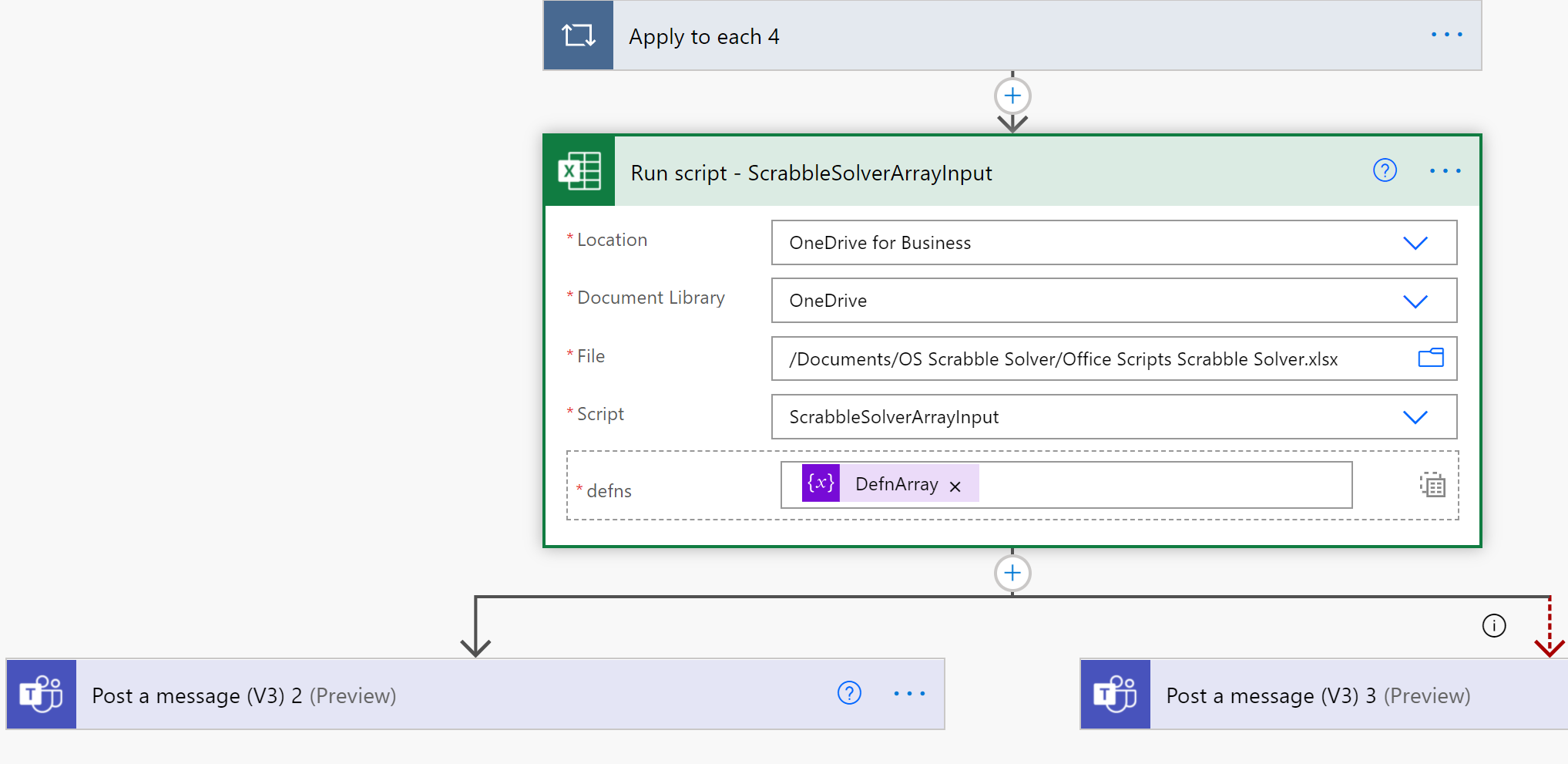
The definition string extraction formula is shown here. Even though the HTTP Response was in JSON format and Parse JSON connectors do operate on this response, they could not be used in this case for practical reasons. This was because the JSON syntax changes, dynamically, depending on the number of *partOfSpeech* items in the JSON response. That is, the JSON changes when a definition contains an adverb, verb, noun or adjective and so on. Each new word has a variable amount of these. The JSON can range from 1 to 7 pages long. See Appendix 1 and 2, of the user guide. So, there is a huge difference in the size and number of layers for each JSON response for each word definition retrieved. Creating a set of PARSE Json actions as well as the associated loops and variables to manage this would be onerous to create. An alternative, much easier, method was possible. The displayed string extraction formula was used, instead. It extracts the first definition it encounters ignoring all subsequent definitions with that same response body.

Graphical user interface, application

Description automatically generated

Substring(substring(Substring(string(body('HTTP')),indexOf(string(body('HTTP')),'definitions'),sub(lastIndexOf(string(body('HTTP')),']'),indexOf(string(body('HTTP')),'definitions'))),29,sub(lastIndexOf(Substring(string(body('HTTP')),indexOf(string(body('HTTP')),'definitions'),sub(lastIndexOf(string(body('HTTP')),']'),indexOf(string(body('HTTP')),'definitions'))),']'),29)),0,indexOf(substring(Substring(string(body('HTTP')),indexOf(string(body('HTTP')),'definitions'),sub(lastIndexOf(string(body('HTTP')),']'),indexOf(string(body('HTTP')),'definitions'))),29,sub(lastIndexOf(Substring(string(body('HTTP')),indexOf(string(body('HTTP')),'definitions'),sub(lastIndexOf(string(body('HTTP')),']'),indexOf(string(body('HTTP')),'definitions'))),']'),29)),'"'))

The final Office Script returns the found definitions back into Excel and places them alongside the generated words.



**The Disclaimer**

SCRABBLE® is a registered trademark. All intellectual property rights in and to the game are owned in the U.S.A and Canada by Hasbro Inc., and throughout the rest of the world by J.W. Spear & Sons Limited of Maidenhead, Berkshire, England, a subsidiary of Mattel Inc.

This solver is not affiliated with SCRABBLE®, Mattel Inc, Hasbro Inc.

**GitHub Repo location**

[**https://github.com/MrAnalyticals/OfficeScripts/tree/main/ScrabbleSolver**](https://github.com/MrAnalyticals/OfficeScripts/tree/main/ScrabbleSolver)

**References**

Ref : Free REST API Dictionaries [Free Dictionary API](https://dictionaryapi.dev/), [WordsAPI](https://www.wordsapi.com/)

Ref : [How Many Letter Tiles Are in Scrabble? (thesprucecrafts.com)](https://www.thesprucecrafts.com/how-many-letter-tiles-are-in-scrabble-410933)

Ref : [Words With 'dsfghy' - ScrabbleWordFinder.org](https://scrabblewordfinder.org/solver)

Ref : [(1) Office Scripts: Update large Excel range in performant way - YouTube](https://www.youtube.com/watch?v=BP9Kp0Ltj7U)

By Sudhi Ramamurthy

Ref : [JavaScript and HTML DOM Reference (w3schools.com)](https://www.w3schools.com/jsref/)

Ref : [javascript - Generate all possible combinations of letters in a word - Code Review Stack Exchange](https://codereview.stackexchange.com/questions/57161/generate-all-possible-combinations-of-letters-in-a-word)