# FUNSourcing: Changes from FUNemployed

## Overview: This document summarizes changes from FUNemployed. The major change is to expand the current “qualities” pool into four, or more if needed, mutually exclusive pools of qualifications so that there will be no conflicting qualifications of the same player due to random draw from the script. An example is given based on [a sourcing context](https://docs.google.com/spreadsheets/d/1Xj3uoybsWUzZhPOUiF-u4q08_2ZjsHdfLlPNtTYfcOQ/edit#gid=0) where Yi-Su Chen adopted and developed. The coding and this document is primarily accomplished by Ryan Sutton.

## Qualification Pools

The base game had a number of qualifications that were not mutually exclusive, but this version contains categories of qualifications in which receiving two from the same category will result in conflicting results. The pools can be defined as follows:

* Quantity Discount (shown in cell **A6**)
* Performance (shown in cell **B6**)
* Relationship (shown in cell **C6**)
* Other (shown in cell **D6**)

An example of the conflict can be seen by examining two of the “Quantity Discount” qualifications:

* We do offer quantity discount but there is no fixed price/discount and everything is all up to the negotiations.
* Quantity discount is not available but we ship in batch, which save you logistics costs.

The base game used a single array on **line 5** of **vars.js** to hold all qualifications (**quals**) and a single array on **line 6** to hold all used qualifications (**quals\_used**). Our modification to the program uses a two-dimensional array, or a series of arrays within one array, to separate the qualifications into discrete pools but still maintain the ease of access.

A spreadsheet was previously used to create the input for **quals** and is similarly used here. The formatting of the **Quals** worksheet now requires that each qualification be placed in its respective column. A listing is as follows:

* Quantity Discount = Cells **A8:A54**
* Performance = Cells **B8:B54**
* Relationship = Cells **C8:C54**
* Other = Cells **D8:D54**

The output in cell **B2** becomes an array of four arrays, one for each type of qualification. This can be copied and pasted (as **Values Only**) to cell **C4** for ease of access. Additionally, because the program recycles qualifications once all of them have been used (e.g., like shuffling a deck of discarded cards), it is necessary to keep track of the used qualifications in discrete pools. The empty arrays are generated in cell **B3**, which can be copied and pasted (as **Values Only**) to cell **C5**.

The **Qualities** worksheet is formatted such that, if a decision was made to expand the number of qualification categories past four, all the user would need to do is enter in the new set of qualifications in cells **E8:E54**, **F8:F54**, and so on for each unique type, and name the category in cells **E6**, **F6**, and so on accordingly.

Once the values have been pasted into cells **C4** and **C5**, they can be copied and pasted into **vars.js** on **line 5** and **line 6**, respectively.

The only function that required modification to meet the demand of the work pool was the **getQual()** function in **vars.js** (**line 26**). Previously, this function did not require any parameters and simply pulled a random qualification from the array. Since we have a series of arrays, we needed a value to determine which of the four internal arrays must be accessed. Thusly, the **getQual()** function was modified to take the **qualClass** variable. The **qualClass** variable will track how many qualifications have been drawn for each player, such that:

* If 0 qualifications had been selected, a qualification will be selected from the Quantity Discount array.
* If 1 qualification had been selected, a qualification will be selected from the Performance array.
* If 2 qualifications had been selected, a qualification will be selected from the Relationship array.
* If 3 qualifications had been selected, a qualification will be selected from the Other array.

This required modification of any references to the **quals** and **quals\_used** arrays in **vars.js** such that operations on those arrays were done on the internal arrays based on the type of qualification selected. This resulted in the inclusion of the **qualClass** variable on **lines 27-29** and **lines 31-33**.

In **script.js**, the **getQual()** function is called twice: once on **line 69** for the start of the round and once on **line 78** for every subsequent selection of qualifications. Passing the function a 0 on **line 69** works for the start of the round, as the Quantity Discount is always the first qualification to be pulled. On **line 78**, the **qualCounter** variable is passed – this variable counts how many qualifications have been selected to determine when the next player is selected.

Adding a new type of qualification would require the user to increment the **qualCounter** Boolean expression on **line 80** such that 4 is changed to the new total number of qualification categories.

## Other Changes

* A header row was added to the table of past qualifications, requiring the addition of **line 55** to create the row and **line 90** and **line 92** to shift the appending of qualifications one row down (as the header row is considered a row that data could be appended to) in **script.js**.
* Changes were made to the **strings.js** file to initialize all string values to match the changes in the **Game Interface** tab of the spreadsheet.
* Previously in the **strings.js** file, **line 34** saw **gameRules** passed into a **parseHTML()** function. This prevented the game rules from being seen on the “How to Play” screen. This function was removed, and the initialization of the **gameRules** text on **lines 4-13** was changed to reflect the new supply chain focus of the game.
* Cosmetic changes (text size and position) were made in the **style.css** file, as the new length of each Job meant that the text would overlap the timer without any changes.

# Quick Guide to Modifying the Game

## Modifying the Jobs

1. Make the necessary changes to the jobs on the **Jobs** tab of the spreadsheet.
2. Copy and paste cell **B2** into cell **C3** as **Values Only**.
3. Copy the value of cell **C3** into the **vars.js** file on **line 3**.

## Modifying the Qualifications

1. Make the necessary changes to the qualifications on the **Qualities** tab of the spreadsheet. Remember that each qualification type must be in its own column, including new types of qualifications.
2. Copy and paste cell **B2** into cell **C4** as **Values Only**.
3. Copy and paste cell **B3** into cell **C5** as **Values Only**.
4. Copy the value of cell **C4** into the **vars.js** file on **line 5**.
5. Copy the value of cell **C5** into the **vars.js** file on **line 6**.
6. If the total number of qualification types were changed, go to **script.js** and:
   1. Change **line 55** to match the new headings.
   2. Change **line 80** to match the new number of qualification types.

## Making Cosmetic Changes

1. Determine what table has the content you’re attempting to change by looking at **script.js**.
2. Go to **style.css** and modify the corresponding section or create one for the table being modified.