1 - Introduction

Thanks for using the Quiver Workflow for **LaunchBar 6**. This action is a work in progress as I am continually adding new features. If there is a feature you would like me to add, please let me know at raguay@customct.com.

The main feature is the Template Expander using the Snippets notebook. With this, you can create **Handlebars** based templates and paste them to your writing application.

To use this workflow, please follow the **Installation** instructions first.

2 - Installation

The first step is to send the Quiver Library or the Snippet Notebook that is shared to this action. This will setup the default variables and load the help file and Snippets notebooks. You are ready to create and use snippets!

If you loaded the default Snippets, then you need to personalize them and fix the data in the Defaults note to fit your needs.

If you have **TextExpander** and would like to pass your snippets through it, you need to also install the **Paste Through TextExpander** action from the same source as the **Quiver Snippets**.

If you have **Keyboard Maestro** and would like to pass your snippets through it, you need to also install **Paste Throught KeyboardMaestro** action from the same source as the **Quiver Snippets**.

Once the **Paste Through KeyboardMaestro** action is loaded, you must run it without any text passing to it for it to load the macro file. It will load a macro called **Paste From Alfred**. I use the same macro with my **Alfred** workflow.

Currently, it only supports the cursor placement macro: %|

3 - The Default Snippet

The **Default** snippet isn't a snippet, but a **json** data structure that will be loaded into **Handlebars** for every expansion. This "snippet" will never show up in the list of snippets for expansion. Therefore, you can define globally used data that is inserted into your snippets.

Since you can cut/paste individual cells really easily, you can create a notebook of different data sets, and cut and paste fragments from that notebook to the **Defaults** note. All **code** cells set to **json** will be pulled into the expansion. Any other cell types will be ignored.

The **expandPlain** variable has a special meaning. If this variable is set to true, then the snippet will be pasted normally. Otherwise, either **TextExpander** or **Keyboard Maestro** will be used to further expand the snippet and place it in the top most application. If the variable **expander** is set to 1, then **TextExpander** is used. If the variable **expander** is set to 2, then Keyboard Maestro will be used. If these variables are set in the **Defaults** section, then that will be the default expansion style. This can also be set in each snippet to over-ride the **Defaults** setting.

JSON structure can be split up among many cells. They will be imported and added to the main structure. Any data redefined by a cell lower than the cell originally defined in will over-write the original definition in the final data structure.

Any **code** cells with a **json** data structure in a **snippet** will also be added to the main data structure and will override whatever is loaded from the **Defaults** data structure.

If you have a **JavaScript** cell in the **Defaults** notebook, it will be loaded as an extension. You have to format as:

```
function myHandlebar(Handlebars,moment) {
   // Your extension code.
}
myHandlebar
```

You add the **Handlebar** helper functions by calling `Handlebars.registerHelper()` with the name of the helper and a function to perform the action. Please refer to the **Handlebars Website** for more information.

4 - Creating Snippets

To create a snippet, go to the **Snippet** notebook and create a note. All cells that are designated as **code** and **handlebars** will be combined together to make the template. All cells that are designated as **code** and **json** will be combined with the **Defaults** data to make the database that **Handlebars** will use to fill in the macro expansions.

All other cell types are ignored. But, you can use them to add notes, explanations, or diagrams for the data and templates.

If you create a snippet with a data structure containing the keyword **versions**, then that snippet will be expanded and the variations sent to the user to pick. An example would be as follows:

This is the template:

```
My email:  {{email}}
```

This is the JSON data structure for doing versions:

```
{
   "versions": [
      {
         "name": "First Email",
         "email": "first@email.com"
      },
      {
         "name": "Second Email",
         "email": "second@email.com"
      },
      {
         "name": "Third Email",
         "email": "third@email.com"
      }
   ]
}
```

The **name** in each version will be shown to the user for picking the version to insert. Therefore, the **name** variable has to be set for each version. All the versions variables are used to expand the snippet. For example, if the user chooses the "First Email", then the phrase "My email: first@email.com" will be inserted.

Helper Functions

The four helper functions I defined are: **save**, **clipboard**, **cdate**, and **date**. All normal **Handlebars** helper functions are usable as well. This is not a complete guide to using **Handlebars**.

All **Handlebars** macros can be rapped in `{{ }}` for HTML escaping of characters, or `{{{ }}}` for raw outputing.

The **save** helper takes two parameters in quotation marks: the *name* and the *value*. A new helper is then created with the *name* that will produce the *value*. The helper also returns the value to be placed in it's place. From that point on, all macros that use the *name* will be replaced with the *value*.

The **clipboard** helper will be replaced with the current contents of the clipboard.

The **date** helper will be replaced with the current date formated by the **Moment.js** formatting variables given as the argument.

The **cdate** helper will be replaced with the date/time specifed by the first argument and the **Moment.js** formatting as the second argument. The date/time format has to follow **2013-02-08 24:00:00.000** format.

The **next** helper will be replaced with the date/time relative to the current date/time. It takes three arguments: **weeks**, **dow**, **fmat**. The weeks is the relative number of weeks. Therefore, this week is 0, next week is 1, in two weeks is 2, last week is -1. The **dow** is the day of the week of that week. Therefore, **weeks** set to 1 and **dow** set to "Sat" will give the Saturday of next week. The **fmt** is the formatting for the date/time as specified in the **Moment.js** documentation.

An example is:

{{next -1 "Mon" "YYYY-MM-DD"}} will give the monday of last week in Year-Month Number-two digit day. Example straight from the clipboard is: 2016-03-28 and the day I wrote this is 04/08/2016.

Predefined Macros

The following are the predefined macros for **Handlebars**:

Macro	Moment Format
cDateMDY	MMMM DD, YYYY
cDateDMY	DD MMMM YYYY

cDateDOWDMY	dddd, DD MMMM YYYY
cDateDOWMDY	dddd MMMM DD, YYYY
cDay	DD
cMonth	MMMM
cYear	YYYY
cMonthShort	MMM
cYearShort	YY
cDOW	dddd
cMDthYShort	MMM Do YY
cMDthY	MMMM Do YYYY
cHMSampm	h:mm:ss a
cHMampm	h:mm a
cHMS24	H:mm:ss
cHM24	H:mm

Note: The "Macro Test" snippet in the default snippets notebook gives an example of many of these macros. I used it to test the special macros, but you can see how they are used for your own snippets.

6 - Expanding the Snippets

To expand a snippet, type "qs" in **Launchbar** and select the **Quiver Snippet Action**. A list of snippet titles will be displayed. Select the snippet you want and it will be pasted in to your topmost application.

If you want to use a different abbreviation, then simply change the default abbreviation in the **Script Editor** or train **LaunchBar** to use a different one.