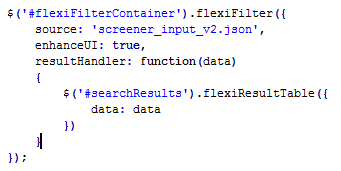
**FlexiFilter - Flexible Filter UI Component**

Daniel Llewellyn (Kenji776)

**The Basics**

FlexiFilter is a jQuery plugin for creating a flexible search/form widget from JSON data, which can then send the results to an endpoint specified and return the results. It optionally has the ability to use the jQuery ui library to style the form in a pleasant way.

Using FlexiFilter is easy. Just call include the css and JavaScript library. Then invoke FlexiFilter on any div you want to populate with the widget.



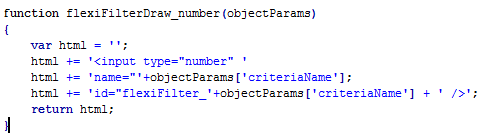
The plugin takes 3 parameters.

* source: The URL (either relative or absolute) from which to load the JSON content to build the widget. The source must be on the same domain as the page (as it is with all ajax requests).
* enhanceUI: Should the jQuery UI library be used to enhance the form elements created. This includes creating an accordion container for each section. If you prefer to make your own enhancements you can pass false for this attribute. If you wish to use most of the UI enhancements but want to make a few small adjustments it is recommended to adjust the CSS file.
* resultHander: a function to which the data from search will be sent to. When the user clicks the submit button the function specified will be passed the resulting JSON fetched from the remote source. FlexiFilter comes with a basic plugin for creating a data table from the resulting JSON, but you may do anything you like with it.

**Supported Form Elements and Creating Extensions**

The JSON fetched from the URL specified by the source attribute it expected to have the same format as the sample file included (screener\_input\_v2.json). Each searchCriteria object is expected to have a criteriaType property that tells FlexiFilter what kind of form field to draw. There are by default five supported elements. Those being: text, dropmenu, checkbox, radio, and slider.

If you wish to add support for more types of elements, you must create a function who’s name takes the form ‘flexiFilterDraw\_[criteriaType]’ where [criteriaType] is the value of the criteriaType string you want to create a form element for. Your function will receive all the data found in the searchCriteria object (such as criteriaType, criteriaName, criteriaLabel, etc) that is responsible for calling the function. You may use that data to customized the returned HTML element. For example if wanted to create a special handler for numerical inputs, you could add the following in the FlexiFilter.js file



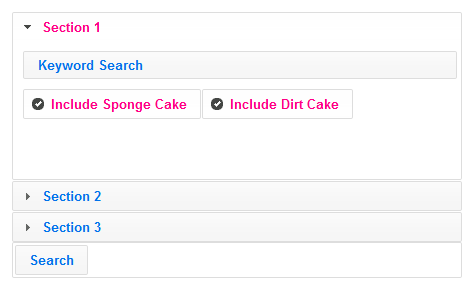
This will return a simple HTML5 numerical input box whenever a searchCriteria object with the criteriaType ‘number’ is encountered. This is a basic example, you’d likely want to add a label, and handling for additional parameters to the input field.

**Testing it Out**

FlexiFilter comes with a demo HTML file where you can see it in action. However, due to browser security issues when fetching content via ajax requests it must be hosted on a webserver. For this reason FlexiFilter comes with a deployment of the mongoose webserver software. If you don’t have a webserver available to test on, you can use mongoose instead. To test FlexiFilter on your local machine start Mongoose (all it through your firewall if prompted), then open your web browser and enter the following URL.

<http://127.0.0.1:8080/flexifilter_demo.html>

You should be presented with a webpage that looks like the following.



You will also see a few debugging text boxes below that show the fetched JSON content used to generate the widget. When you click search you will also see the other two boxes populate with the request and response for the search action. After you click search you should also see a results table show up that looks like this.



That is the included FlexiFilter ResultTable plugin processing the fetched search result JSON into a simple table. Of course in your production version you can replace that handler with anything you desire.

**Additional Customization**

FlexiFilter is of course totally open source. You are free to gut it down to its very core to make whatever changes you like. There are however a few things FlexiFilter does that may make some of your customization easier.

1. jQuery UI library
   1. You can use any jQuery UI css stylesheet you like to radically change the look and feel of the FlexiFilter widets. You can easily try the different stylesheets here <http://stackoverflow.com/a/1349238> or use the jQuery UI themeroller to make your own.
2. FlexiFilter CSS
   1. The CSS for FlexiFilter is intentionally fairly minimalistic outside of the main jQuery UI css. If you wish to adjust margins, spacing, or some of the font formatting though the CSS sheet is the place to do it. Classes are labeled and should be pretty easy to understand.
3. Created Elements contain all source data
   1. When the source JSON is being parsed to create the HTML form elements, all the data about that element is passed to the function responsible for creating the element. All the data is then encoded as attributes on the form element. So you can pass in additional data in your JSON and it will be included in the elements. This makes it easier to do any post processing or tweaks you may want to perform. You can also pass in some of the valid HTML attributes (excluding id, name, and type) to further affect the behavior of the created elements. For example if your criteriaType was set to ‘text’ and you included a key called ‘maxLength’ that would be included in your created textbox, therefor setting the maximum length of the textbox to the value indicated by the maxLength attribute. You can also pass in custom CSS classes, or custom created attributes.
4. Easy extensions for additional data types
   1. As covered above, the creation of the form elements is handled in a fairly modular fashion. Your custom functions can do whatever you like when a certain criteriaType is encountered including drawing custom input types.

**Troubleshooting**

1. Nothing is being drawn
   1. Make sure you have included the the jQuery, jQuery UI, and FlexiFilter javascript libraries and stylesheets. See the demo file for everything required.
   2. Your JSON content is on a different domain than the FlexiFilter page. In this version the JSON content source must be on the same domain as the FlexiFilter page. If this is an issue you could modify the code to use jQuery JSONP and callbacks, or instead create a server side proxy that requests the data from your remote source.
2. Clicking Search does nothing
   1. Make sure the URL specified by the searchEndpoint variable in the source JSON is valid and on the same domain as the FlexiFilter page. See above (1b) for alternatives.

And there you have it, a dynamic, modular, flexible plugin for creating search widgets and handling the resulting data. I hope you enjoy using it as much as I enjoyed writing it ☺