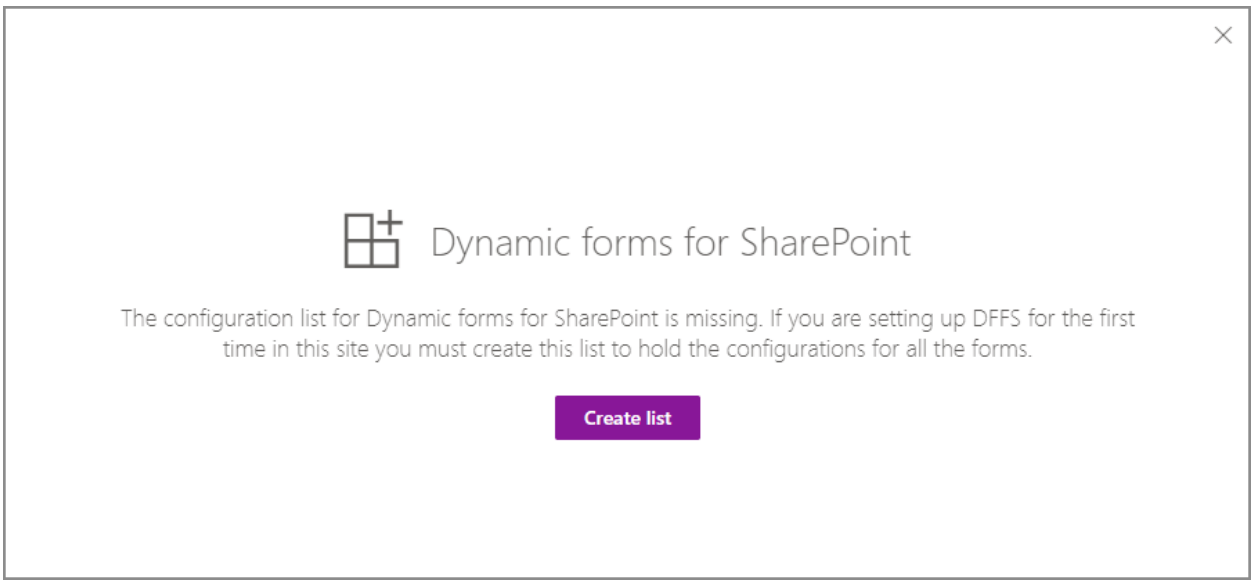
1 Contents

2 Enter setup

2.1 Create configuration list

When you have installed the Modern DFFS you will see a button “DFFS” in the banner of the list. This will only show for users with **Manage lists** permission. Click this button to enter setup.

The first time you enter setup you must create the configuration list:

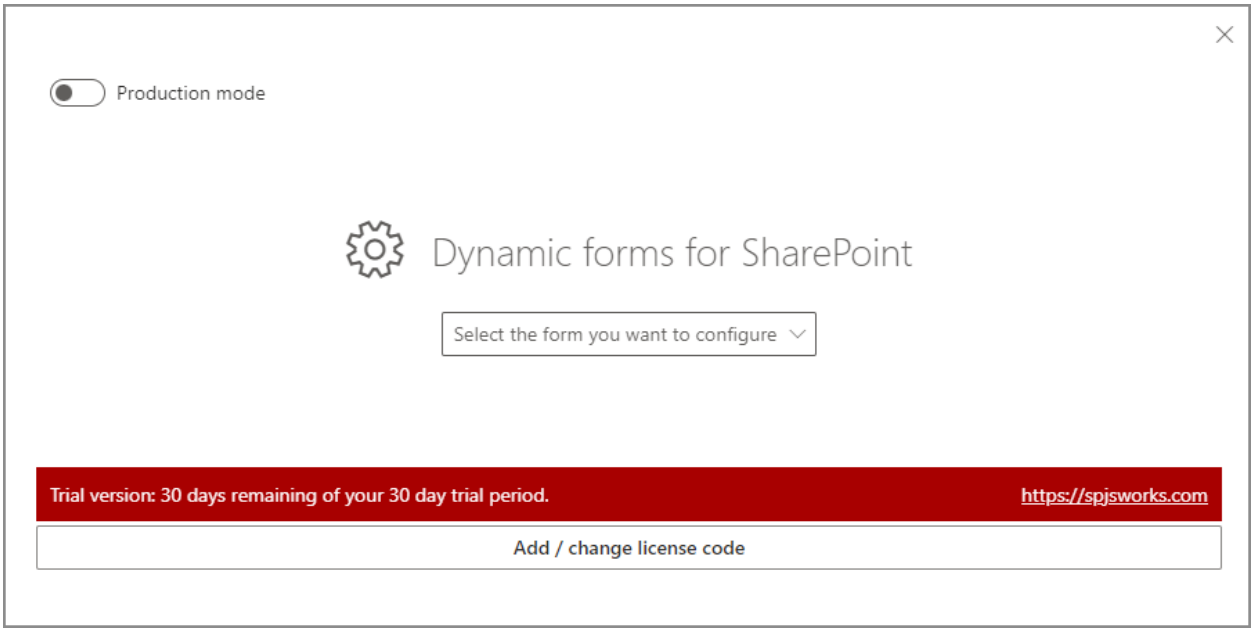


This configuration list is a standard *custom list*, but it is by default hidden from all site contents. You are not supposed to manually edit this list, but it can be accessed by typing in the list name in the URL like this: *…/Lists/DFFSConfigurationList*

Please note that all users must have read access to this list to access the configuration for the list.

2.2 Enter setup or add / change the license code

When you have created the configuration list, you will see this screen:



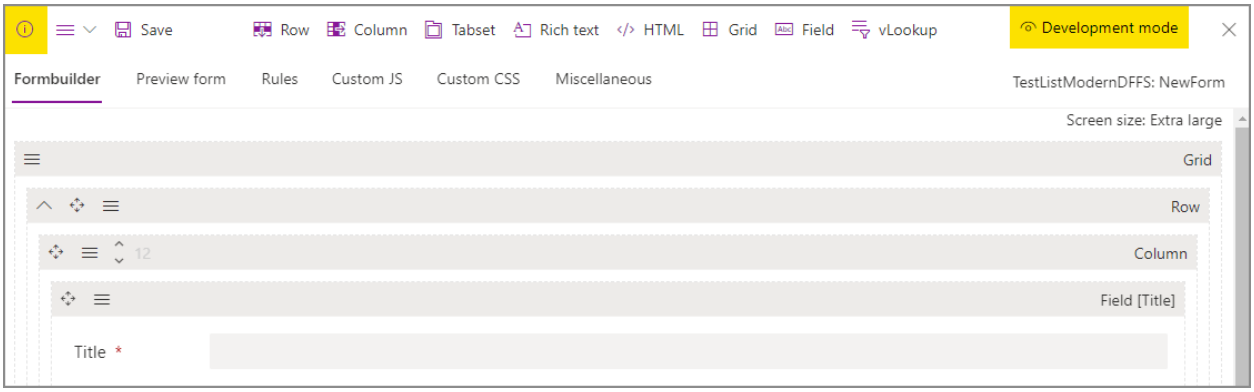
2.2.1 Enter setup

Here you can select the form you want to configure from the drop-down menu. You can choose from the following forms: NewForm, DispForm and EditForm.

You use NewForm to create new list items (a document library does not have an NewForm), DispForm to display an existing list item and EditForm to edit an existing list item.

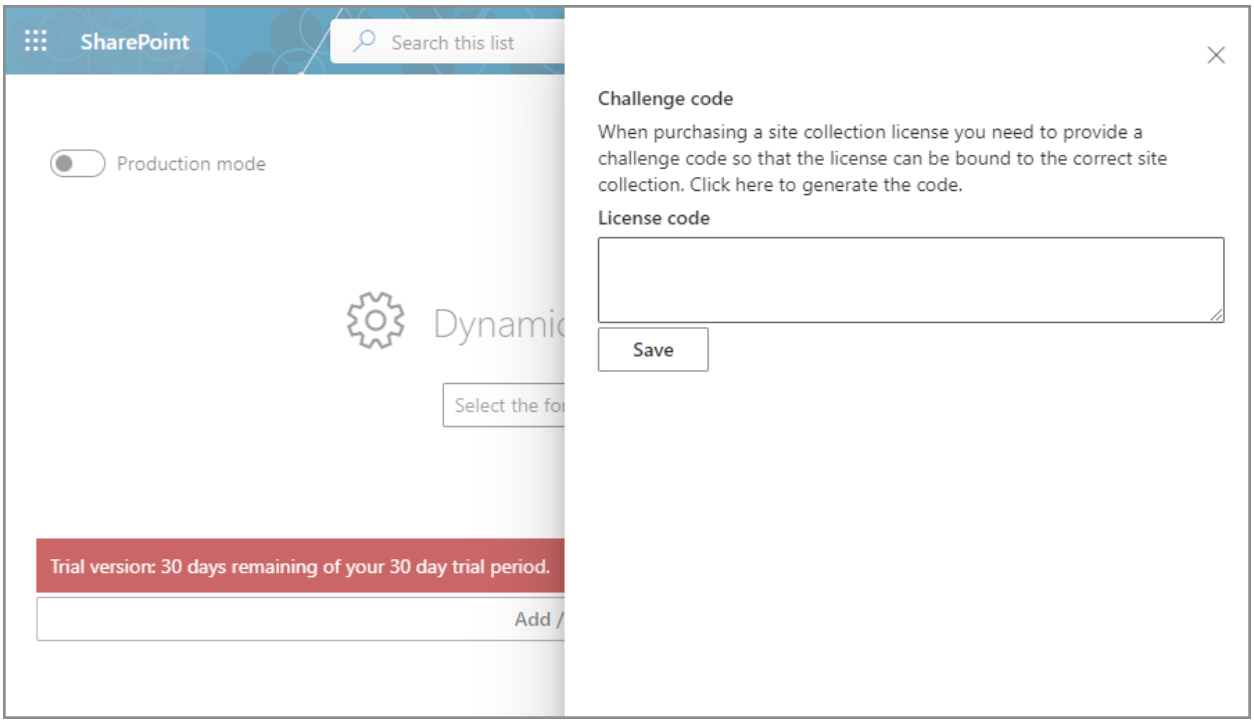
2.2.2 Production or development mode

You can toggle between production and development mode using the toggle in the top left corner. When toggling this to Development mode and entering a form configuration you will see a yellow button in the top right corner. If you click this button, you can assess your development mode configuration in a new window.



2.2.3 Generate challenge code

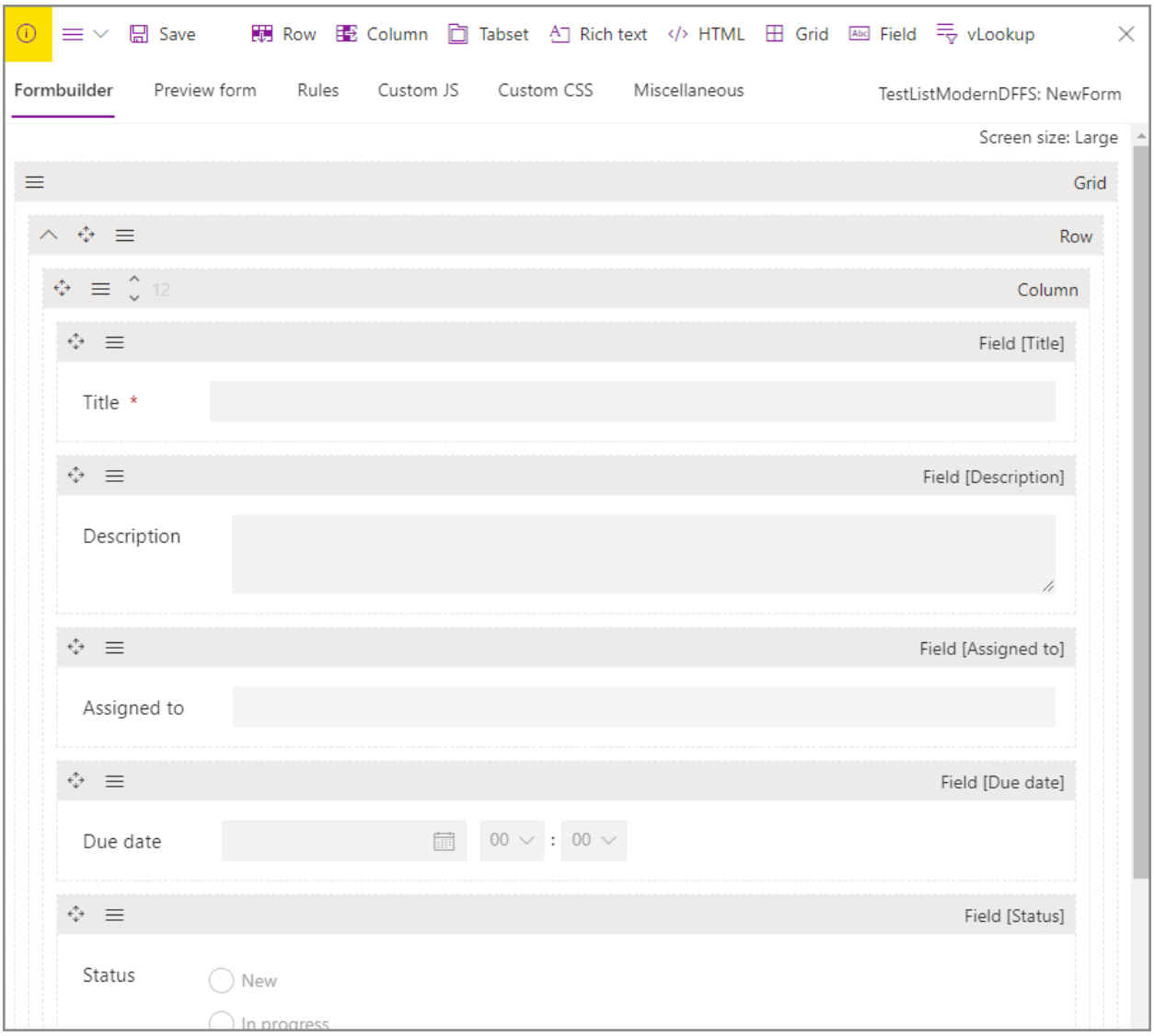
When you purchase a site collection license you must generate a challenge code to bind the license code to the current site collection. Click the **Add / change license code** to access the challenge code.



You get a 30-day trial when you first install the solution. You can purchase a license or ask for a quote from the here: https://spjsworks.com/purchase

3 Configuration of a form

When you open the configuration for the first time, all fields are automatically added to the form. You can delete the ones you do not want to have visible in the form. Deleting a field from the configuration for one form does not delete it from the list settings.



To add new fields to your list you must use the default list settings. You find a link to open list settings from the Miscellaneous tab.

You can rearrange the fields by drag-and-drop, and you can access the properties pane for all elements by using the burger-button or on the right click menu. The different form elements have different properties – you find a bit more information in the next section.

3.1 Banner buttons

3.1.1 Save

Saves the configuration.

3.1.2 Import /Export

This opens a panel where you can browse through other configurations and restore points created for this list, import configurations from the Classic DFFS (if you have the Classic DFFS version already in use in this site) or import / export as JSON from the Classic DFFS or the Modern DFFS.

See more details in the Upgrade from Classic DFFS section below.

3.1.3 Create restore point

Create a restore point that lets you change back to this configuration if needed. It is recommended that you create a restore point before doing major changes.

3.1.4 Switch form

Switch between NewForm, DispForm and EditForm of the list (NewForm is not available in document libraries).

3.1.5 Delete

Use this to send the configuration to the recycle bin and change back to the out of the box SharePoint form.

3.2 Formbuilder

Use the banner buttons to drag-and-drop form elements onto the form.

You can choose from the following elements:

- Row
- Column
- Tabset
- Rich text
- HTML
- Custom ROw
- Grid
- Field
- vLookup

You can configure all form elements by opening the properties pane using the burger-button or the right click menu.

3.2.1 Row

Rows are used as placeholders for columns. You can add an unlimited number of rows to your form.

3.2.2 Column

A row has 12 available sections. You can have one column that takes up all 12 section or add any number of columns (up to 12) and distribute the sections between them.

You can set the distribution of space depending on the screen size to make a more responsible for different screen sizes.

3.2.3 Tabset

You can create tabsets and add any number of tabs. If you do not want to use the default tab color (if is based on the color theme of your site) you can set the color of the individual tab, or add a default color for all tabs in the Miscellaneous tab. You can make a tabset "sticky" to ensure it stays on the top of the screen when scrolling a long form.

Each tab will have a placeholder where you can add rows and columns.

You can configure a tabset as a multi-step form where you hide the tabs and instead show a previous and next button at the bottom of the tabset and you can navigate between the tabs like in a multi-step form.

There are bullets below the tabset that show which tab you are currently on, and if a required field is missing data in that tab when you try to save the form.

3.2.3.1 Select same tab when navigating from DispForm to EditForm

If you use the same tabs in DispForm and EditForm and want to select the same tab in EditForm when navigating from DispForm you can set up a rule in EditForm triggering on *Form is loaded,* add the *Select tab* action and select the option *When navigating from DispForm, select the same tab*.

3.2.3.2 Set selected tab in a tabset

When configuring a tabset you can set the default selected tab for each tabset.

If you want to create a link to a form and want to select a specific tab - that is not the default selected tab specified in the tabset - you can create a link with an URL query string key like this:

sTab=[id_of_the_tab]

To find the ID of your specific tab you must right click the tab you want to select and select *Inspect* to use the developer tools in your browser. This will open the developer tools and show something like this:

The number is used in the URL key like this: sTab= 9502543918899333

3.2.4 Rich text

You can use this control to add your own rich text to the form. This is not text that will be saved to the list, but instructions to the user that fill in the form.

3.2.5 HTML

You can add your own custom HTML. Accompany this with Custom JS and Custom CSS to create your own form elements. The values in any HTML-elements you create will not be saved to the list when saving your form element unless you use your own Custom JS to do so.

3.2.6 Grid

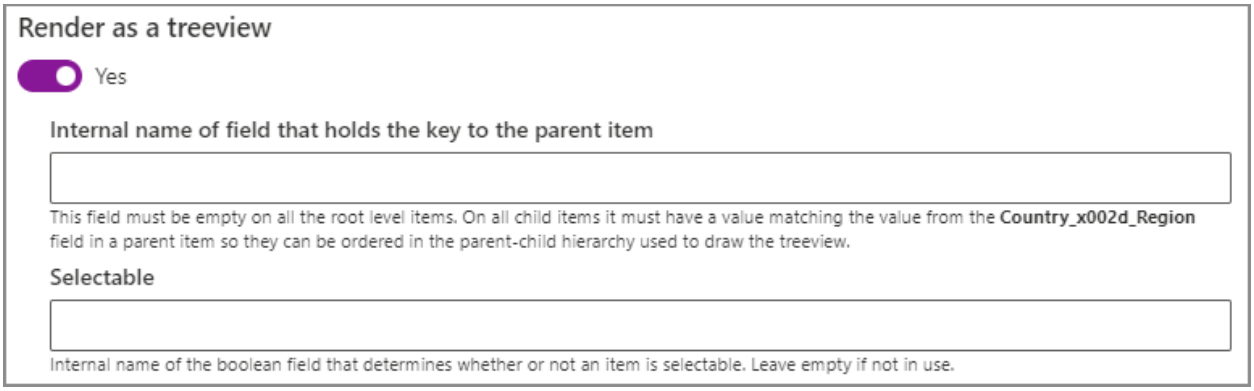
Use this to make a grid layout where you can add rows and columns.

3.2.7 Fields

You can select from all the built in SharePoint field types. To be able to add a field to the form, you must first crate it in the default list settings for that list. You find a button to open the list settings in the Miscellaneous tab.

3.2.7.1 Render a lookup field as a treeview

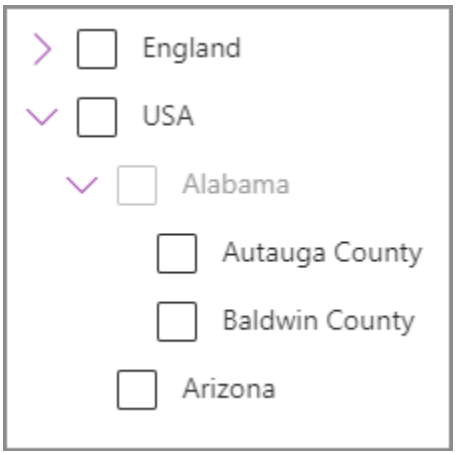
A lookup column can be rendered as a treeview. Open the field properties and toggle **Render as treeview** to Yes.



You must have a field that links the child items to the parent to maintain the parent-child relationship. For example, a field named Parent that for the child items holds the Title of the parent item.

You can also have a Boolean field in your list that determines whether you can select each item or not.

Here is an example where the Selectable option is set to false for Alabama.

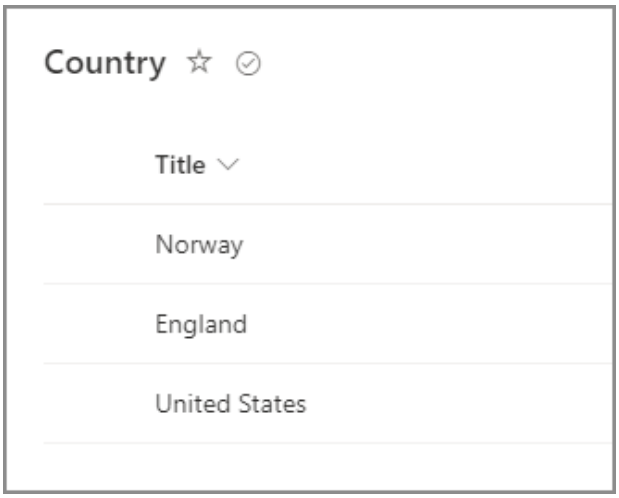


3.2.7.2 Create cascading lookups

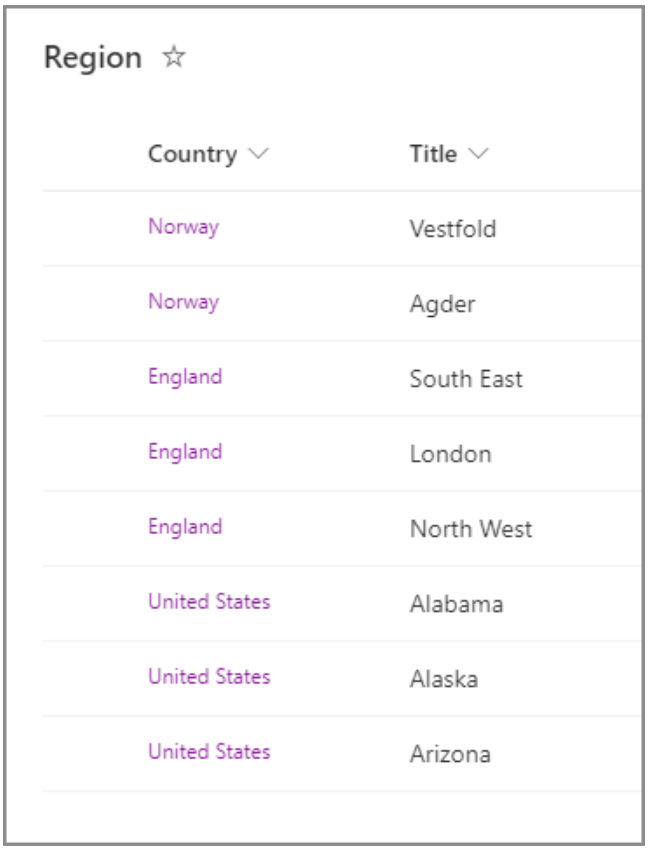
You can make your lookup fields cascading by using the Filter option. Use the **Add dynamic content** menu in the bottom right corner (where you focus the filter field) to pick the field you want to use as filter. The filter is updated when that field is changed.

This example shows how you can set up two lookup columns in your form *Country* and *Region* where the second one is filtered by the selection in the first. To do this you must create two lists that your lookup columns will get their values from – one for country and one for region.

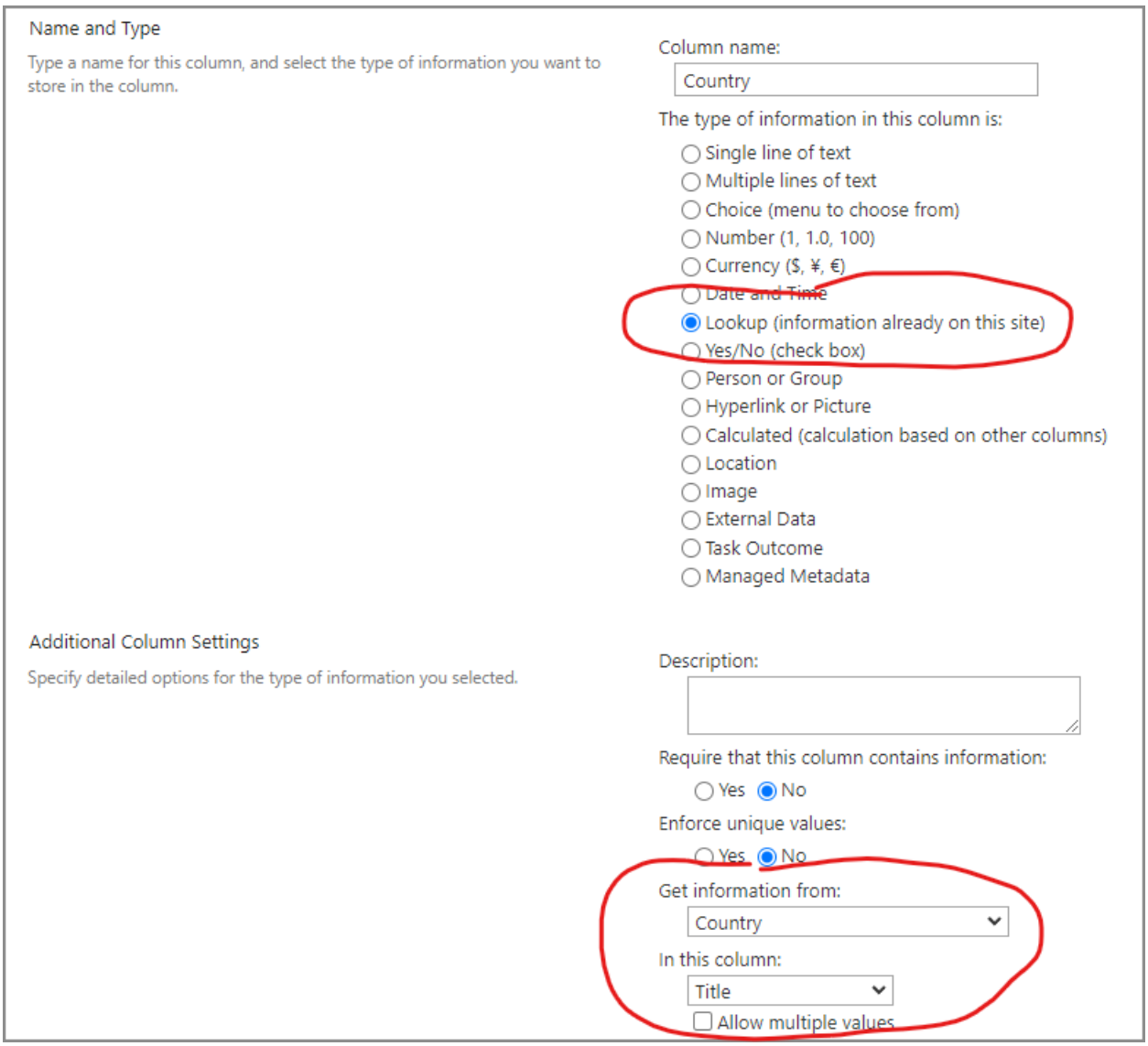
The first list *Country* is set up with only the Title field:



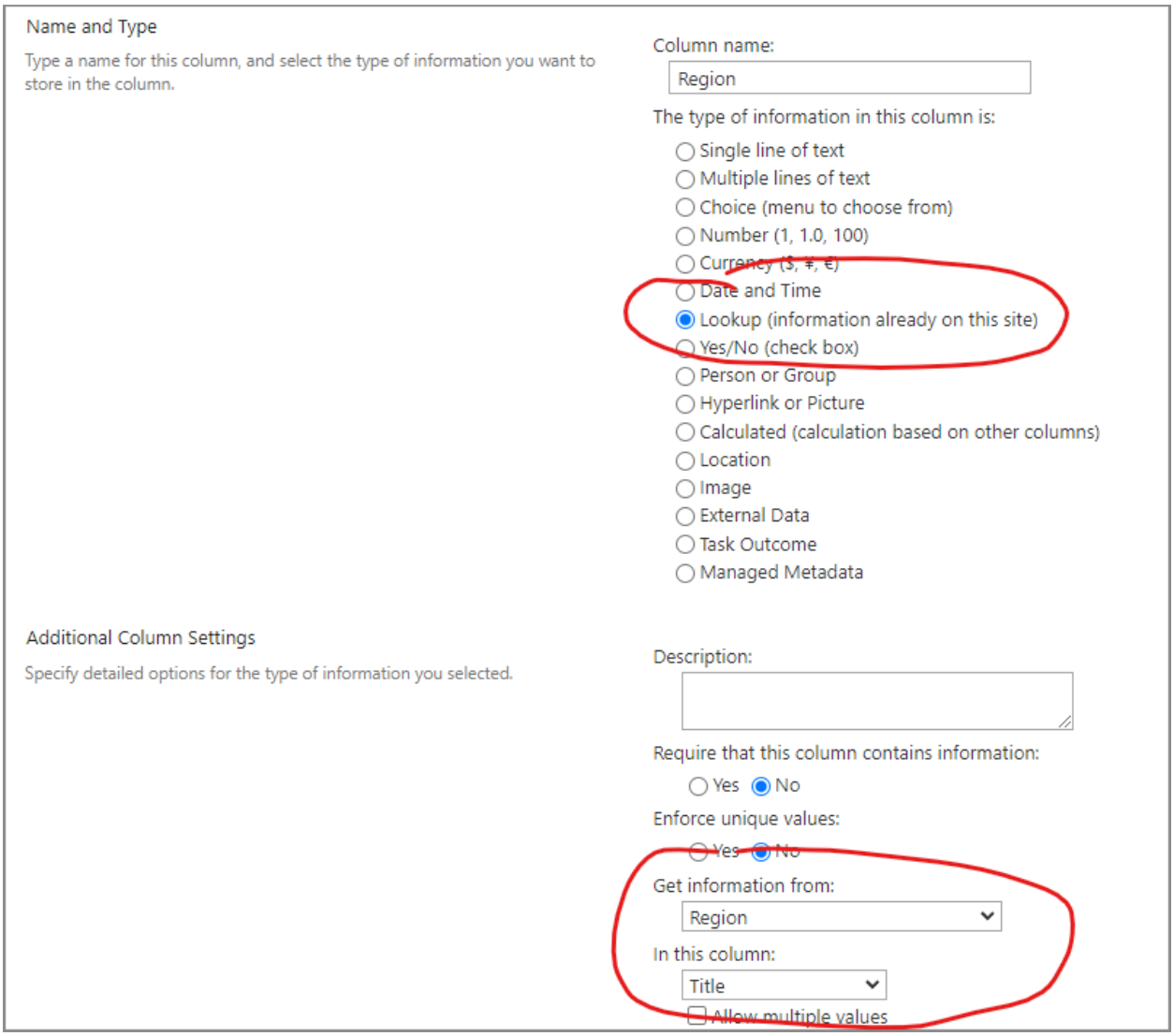
The second list *Region* is set up with a lookup column that looks up the Title field from the *Country* list. The Region name is specified in the Title field:



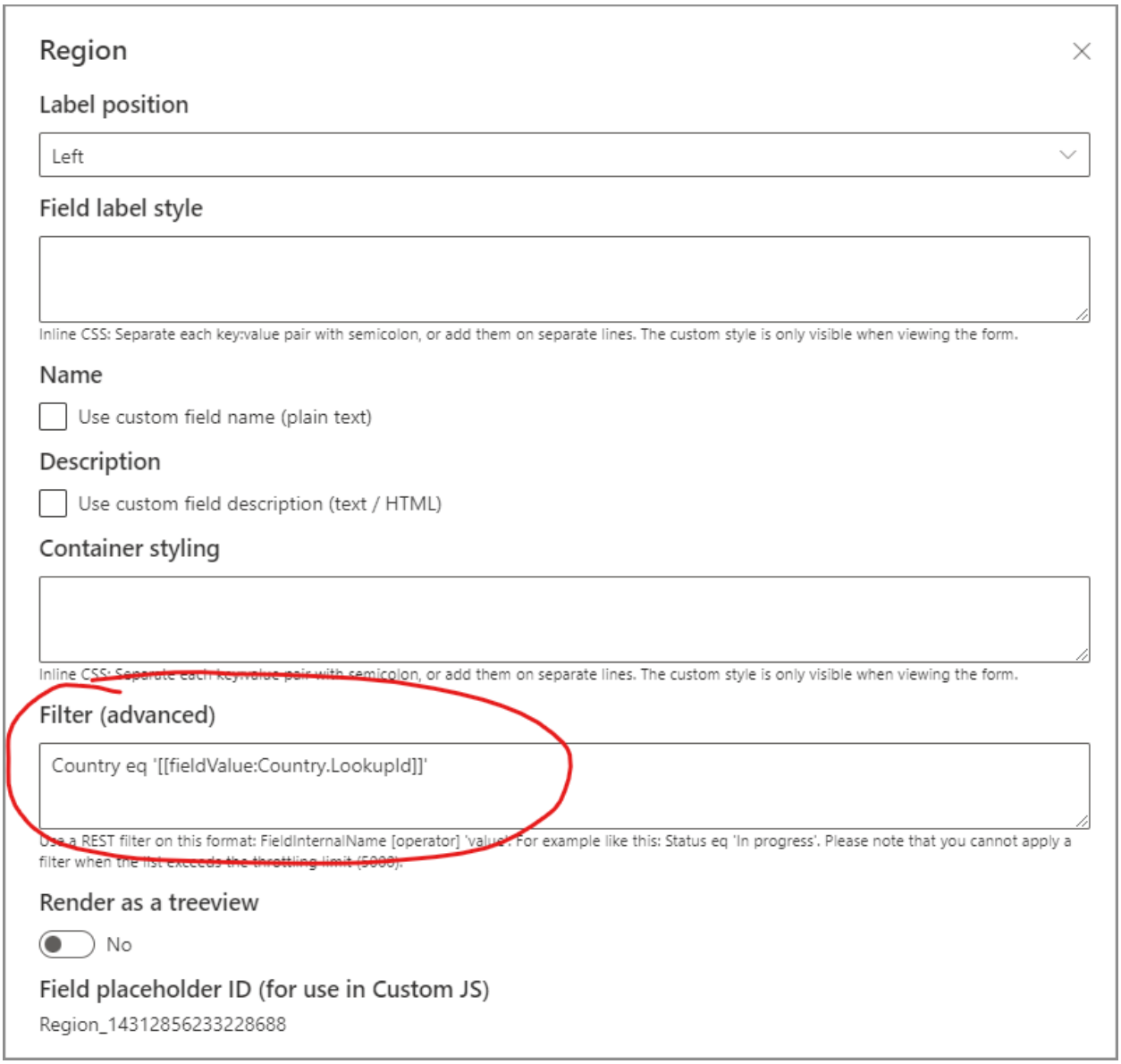
Now you can create the two lookup columns in your form – one for *Country*:


Name and Type
Type a name for this column, and select the type of information you want to store in the column.
Column name:
Country
The type of information in this column is:
Single line of text
Multiple lines of text
Choice (menu to choose from)
Number (1, 1.0, 100)
Currency ($, ¥, €)
Date and Time
Lookup (information already on this site)
Yes/No (check box)
Person or Group
Hyperlink or Picture
Calculated (calculation based on other columns)
Location
Image
External Data
Task Outcome
Managed Metadata
Additional Column Settings
Specify detailed options for the type of information you selected.
Description:
Require that this column contains information:
Yes
No
Enforce unique values:
Yes
No
Get information from:
Country
In this column:
Title
Allow multiple values

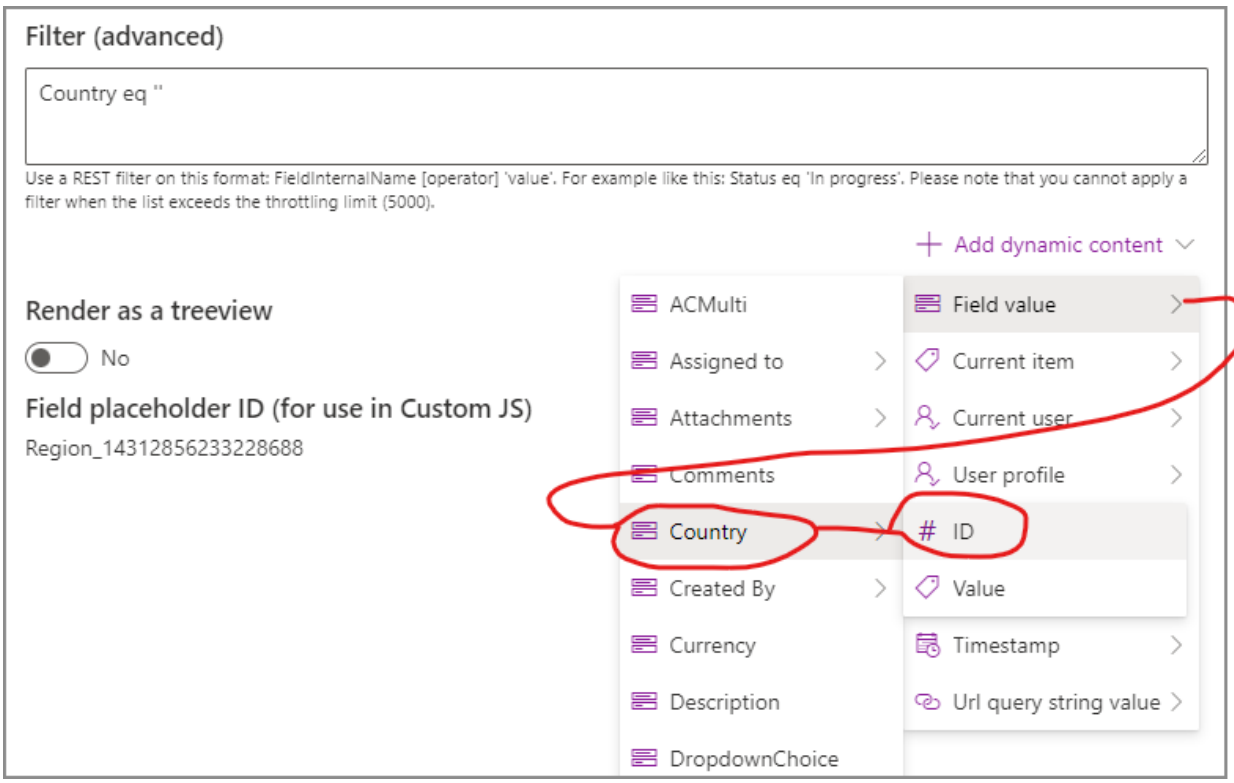
And one for *Region*:



Now you can enter the Modern DFFS configuration to set up the filter on the *Region* lookup column:



The field name *Country* in the filter is the internal name of the lookup field *Country* in the list *Region*. The filter value *[[fieldValue:Country.LookupId]]* can be found by clicking the Add dynamic content below the textarea (shown when you focus on the textarea) and selecting the lookup field for Country in the current list – like this:



Your Regions column will now be filtered to show only the regions that fall under the Country selected in the first lookup field.

3.2.7.3 vLookup

Use this control to show a table view of items from a child list. You configure the control by selecting the web, list, and filter to retrieve the desired items.

You can select which fields to populate the table and configure **Add new** to be able to create a new child item that is automatically linked to the current item by prefilling values from the current item (the same values as you use in the filter).

Linking parent and child forms using _DFFSID or _vLookupID

Like in the classic DFFS version you can link the “parent” and “child” lists using an automatically generated unique id in your form instead of having to use a lookup column to link the two.

If you add a field named **_DFFSID** to your list, it will automatically be populated with an auto-generated id (a timestamp in milliseconds). This can, using the **Prefill field values** functionality, be transferred to a single line of text field that you create in the child for.

You can use this field in the filter using a format like this:

```
ParentId eq '[[fieldValue:_DFFSID]]'
```

Where “ParentId” is the field that you created in the “child” list.

For backwards compatibility with older lists created with the classic DFFS version you can also use the _vLookupID field – if added to the list, it will auto-generate a similar unique id as for the classic DFFS version.

Add a callback function when vLookup table is ready

You can add a custom function that triggers every time a vLookup table is loaded (on form load, or if a new item is added) by using code like this in Custom JS:

```
function dffs_vLookup_callback(id, items){
  if(id === "vLookup_7943515991751215"){
      // Loaded vLookup with id vLookup_7943515991751215 - you can find the ID when editing the vLookup configuration
     console.log(id, items);
  }
}
```

Override configuration for a vLookup in Custom JS

This is for advanced users and can for example be used to switch between different document libraries depending on a selection in the form.

To use this functionality, you must configure the vLookup using the vLookup control and then add an object like this in the Custom JS editor in the form to override the values:

```
var vLookup_config_override = {};
if (getFieldValue("DocLibSelect") === "DocLibA") {
 vLookup_config_override["vLookup_13389631338251728"] = {
  "list": {
    "override": "DocLibA",
    "guid": "f31dfde0-3299-4415-80fd-64806cf091bc",
    "title": "DocLibA",
    "rootfolder": "DocLibA",
    "BaseTemplate": 101
  }
 };
} else {
 vLookup_config_override["vLookup_13389631338251728"] = {
  "list": {
    "override": "DocLibB",
    "guid": "91c5e295-ccdd-4855-8419-06ec284c0c22",
    "title": "DocLibB",
    "rootfolder": "DocLibB",
    "BaseTemplate": 101
  }
 };
}
```

The ID used in the vLookup_config_override object can be found in the first tab in the vLookup configuration. You can override all the values, but because of how the override object is merged with the current configuration in the code you must add all properties and not for example only “guid” in “list” in the example above.

To find the complete vLookup configuration object you must export the configuration (using the Import, export and restore menu item in the top left corner) and parse the data from the FormJSON property.

Refresh a vLookup table from Custom JS

If you want to refresh a vLookup table from Custom JS you can use this function:

```
vLookup_loadItems("vLookup_06588267196671516").then(() => {
   console.log("done");
});
```

The id can be found in the vLookup configuration at the bottom of the first tab.

Open a vLookup form from Custom JS

To open a vLookup from form Custom JS you can use code like shown below. Opening NewForm uses only 2 arguments – the ID for the vLookup and the form type “new” like this:

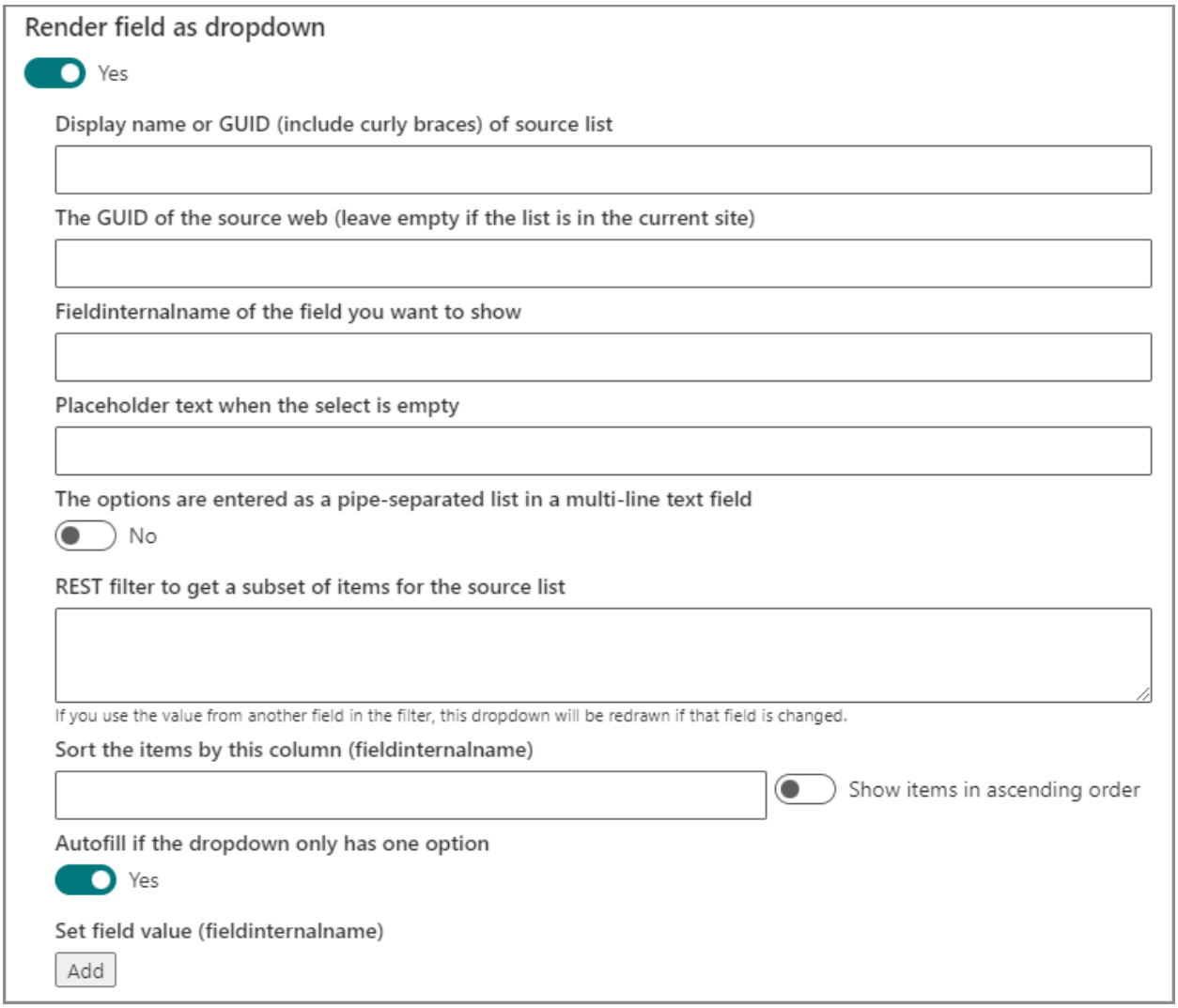
```
vlookup_open("vLookup_06588267196671516", "new");
```

Opening DispForm or EditForm uses three arguments – the id of the vLookup, the form type and the Item ID like this:

```
vlookup_open("vLookup_06588267196671516", "disp", 123);
vlookup_open("vLookup_06588267196671516", "edit", 123);
```

3.2.7.4 Render a single line of text field as a Dropdown

You can render a single line of plain text field as a dropdown that lists the options from another list in your SharePoint site collection. Open the field properties and toggle **Render field as dropdown** to Yes. Configuration is done in the field properties like this:



Now specify the display name or the GUID of the source list, the internal name of the field you want to use as the selectable options, the placeholder text, and any REST filter you want to use to filter the data source. You can also change the sort order. You can also check **Autofill if the dropdown only has one option** to have it automatically fill in the value when it is the only option.

You can use “dynamic content” in most of the textfields.

You can make your custom dropdowns cascading by using the **REST filter to get a subset of items for the source list** option. Use the **Add dynamic content** menu in the bottom right corner (where you focus the filter field) to pick the field you want to use as filter. The filter is updated when that field is changed.

Pipe-separated options

Check this box if you have the options for the dropdown in a multiline textfield in this format:

```
Red|Green|Yellow|Blue
```

Please note that if you check this option, the Set field value functionality will be disabled.

Set field value

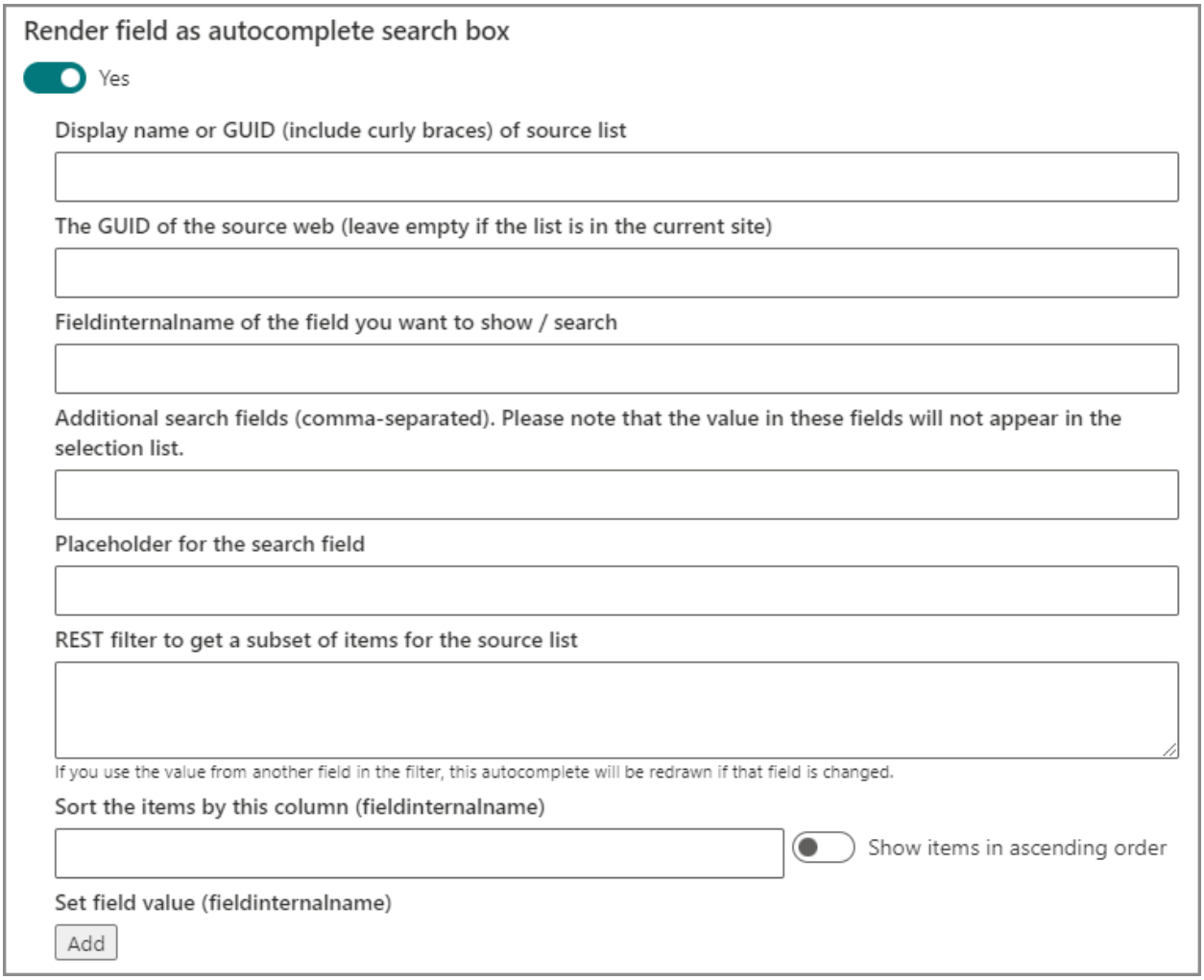
If you use a single line of text field (single choice autocomplete) you can set additional values by specifying the field to pull the data from and the field to write it to. You can write the From field like this to pull from a complex field type like a people picker or a lookup column:

```
FieldInternalName/Title
```

3.2.7.5 Render a single line or multi line of text field as an Autocomplete

You can render a single line (for single choice) or multiline (for multi choice) plain text field as an autocomplete lookup field that lists the options from another list in your SharePoint site collection.

Open the field properties and toggle **Render field as autocomplete search box** to Yes. Configuration is done in the field properties like this:



Now specify the display name or the GUID of the source list, the internal name of the field you want to use as the selectable options, the placeholder text, and any REST filter you want to use to filter the data source. You can also change the sort order. Additional search fields can be used if you want to let the user search for records using multiple search fields.

You can use “dynamic content” in most of the textfields.

You can make your autocompletes cascading by using the **REST filter to get a subset of items for the source list** option. Use the **Add dynamic content** menu in the bottom right corner (where you focus the filter field) to pick the field you want to use as filter. The filter is updated when that field is changed.

Set field value

If you use a single line of text field (single choice autocomplete) you can set additional values from the lookup source list by specifying the field to pull the data from and the field to write it to. You can write the From field like this to pull from a complex field type like a people picker or a lookup column:

```
FieldInternalName/Title
```

3.2.7.6 Render a single line of text field using a custom render function [advanced]

A single line of text field can be rendered using a custom function if you want to add your own customized rendering.

Start by adding the field to your form and then open the field properties. Check the **Use a custom render function** and add the name of the function – for example *myTxtFieldRenderFn*.

Open the Custom JS tab and add a function with the name you specified. This example will render a dropdown menu with two options. You can render any control you like as long as the value can be stored in a single line of text field.

```
function myTxtFieldRenderFn(f) {
 let options = [{ "val": "Option 1", "text": "Option one" }, { "val": "Option 2", "text": "Option two" }];
 let currFieldValue = getFieldValue(f.InternalName);
 let b = [];
 b.push("<select onchange='changeCustomDropdown(\"" + f.InternalName + "\", this)' style='padding: 6px 10px'>");
 b.push("<option value=''>---</option>");
 options.forEach(opt => {
  b.push("<option value='" + opt.val + "'" + (currFieldValue === opt.val ? " selected=true'" : "") + ">" + opt.text + "</option>");
 });
 b.push("</select>");
 return b.join("");
}

function changeCustomDropdown(fin, elm) {
 var value = elm.value;
 // Set the value to ensure it is saved with the form
 setFieldValue(fin, value);
}
```

3.3 Preview form

You can preview the form configuration from within the configurator. Some fields are rendered as read-only, and lookup columns rendered as treeview will have dummy content.

3.4 Rules

You can create rules to make your form dynamic. You can combine different triggers to create complex triggers by using a combination of and / or.

You can use the form fields as trigger using these operators:

- Is equal to
- Is not equal to
- Contains
- Does not contain
- Is greater than
- Is greater than or equal to
- Is less than
- Is less than or equal to
- Starts with
- Does not start with
- Ends with
- Does not end with

In addition to these triggers:

- Form is loaded
- Before save of the form
- After save of the form
- Using a mobile device
- SharePoint group
- Selected tab
- URL query string value
- Run based on the result of another rule
- Run based on the result of a Custom JS function (on form load)
- Run from a Custom JS function

When you have set up the trigger you must select the actions. You can set up different actions based on the result of the rule – **If yes** and **If No**.

You can select among these actions:

- Required fields
- Optional fields
- Visible fields
- Hidden fields
- Editable fields

- Readonly fields
- Show elements
- Hide elements
- Show / hide form buttons
- Call a function
- Set field value
- Prepare email
- Remove email
- Show / hide tabs
- Select tab
- vLookup readonly / editable
- Show a message box
- Show a field message
- Remove a field message

3.4.1 Run based on the result of a Custom JS function (on form load)

Add a function like this in Custom JS:

```
function checkFromRule(ruleId) {
  let result = false;
  // Add your custom logic here and set result to true or false to trigger the if yes or the if no section of the rule
  return result;
}
```

3.4.2 Run from a Custom JS function

When you select this trigger you can an example of how to trigger the rule. It takes two arguments, one is the Rule ID and the second is a Boolean value to select the if yes or the if no part of the rule actions. The function call is like this (change the Rule ID to match your rule).

```
dffs_triggerRule("43796707680074887", true);
```

Please note that only a rule configured with the trigger “Run from a Custom JS function” can be triggered like this.

3.5 Custom JS

In this tab you can load external *.js files or add custom js directly to an editor.

You can find code examples in the forum here: https://spjsblog.com/forums/topic/custom-js-examples/

3.6 Custom CSS

You can add your own custom css style your custom HTML sections or to override any styling on your form.

You can load external CSS files by writing this in the Custom CSS field:

```
@import "/path_to_file/filename.css";
```

If you want to override any of the default classnames you must replace the auto-generated ID with the text SUFFIX. Example: change formLabel_a04d0572 to formLabel_SUFFIX. The reason for this is that the suffix number is automatically generated between each load of the page.

3.7 Miscellaneous

This tab contains various settings.

3.7.1 Password

You should set a password to protect your configuration from others.

3.7.2 Form panel type

Choose from the following list of form types:

- Full width
- Large
- Medium
- Small

3.7.3 Comments

Check the **Show comments button in the top right corner of display form and edit form** to let the users comment on the list items. This uses the out-of-the-box SharePoint comment functionality.

3.7.4 Tab colors

You can set the default colors for the tabsets you add to the form.

3.7.5 Load the configuration from another site or from a file

This feature lets you load the configuration from another site in your tenant, or from a file. This can be used when you have multiple sites created from a template, and you want to manage the configuration for all forms centrally.

You can find more information in the contextual information text in the Miscellaneous tab.

In addition to these two options that must be applied to each form you can also set a general “Load all configurations from another site” for all lists in the site by manually adding an item to the configuration list (see 2.1 for information on how to access the list).

Set the Title field to “LoadAllConfigsFromOtherSite” and the Misc field to the fully qualified URL of the site where the configurations are stored.

Please note that this is the URL to the site and not to the list. For example:
https://contoso.com/sites/your_site

4 Printing a DFFS form

Because of how the DFFS form is rendered, printing the form using the default print functionality in the browser does not work 100%. To print properly you must add a button to your form using a HTML section with code like this:

```
<div style="text-align:right">
<span style="font-size:3em;cursor:pointer;" onclick="dffs_print()">&#128438;</span>
```

```
</div>
```

If you cannot see the icon, replace 🖶 with an image tag with your preferred print icon.

5 Modern DFFS WebPart

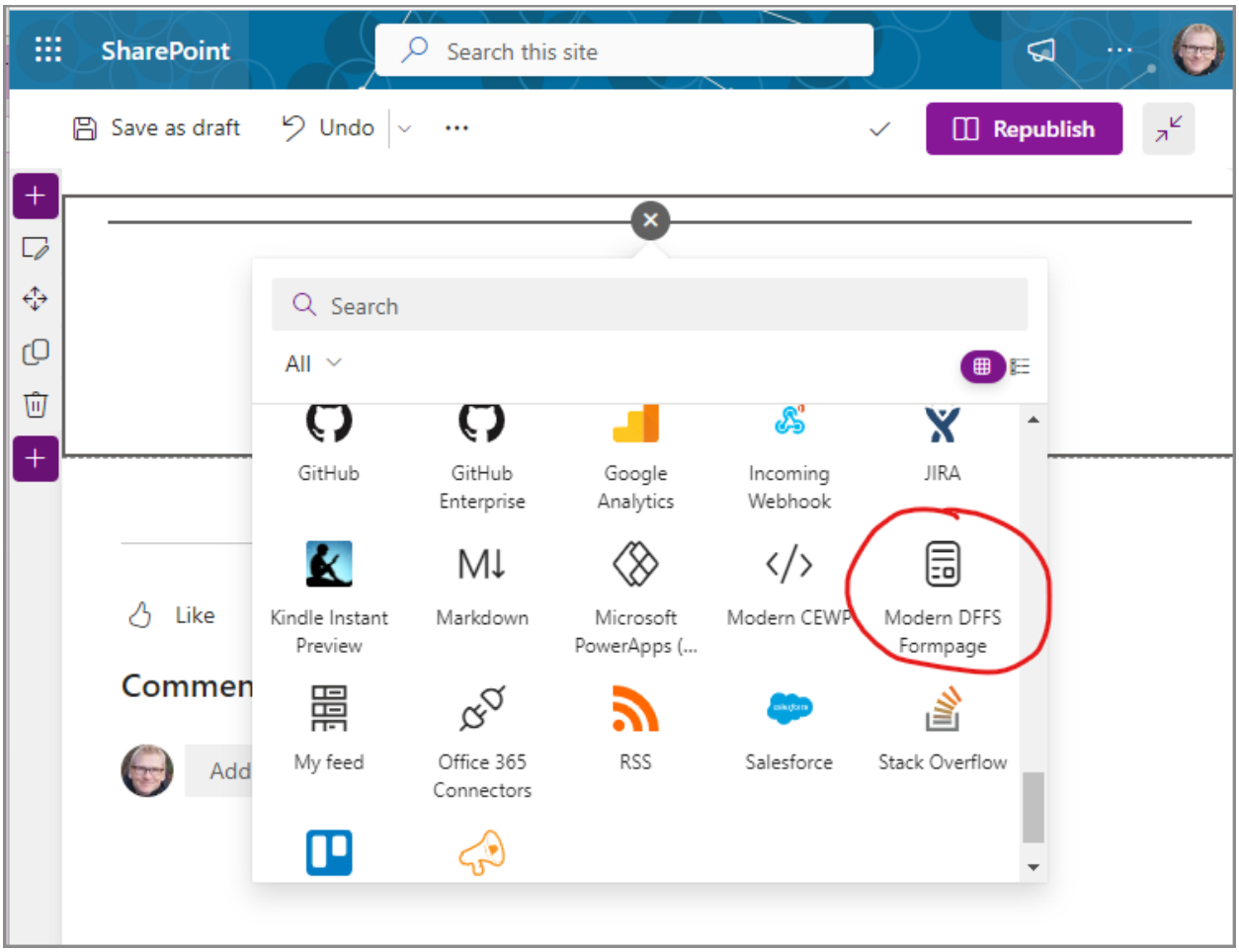
Requires Modern DFFS solution v1.0.14.0 or later.

This web part is used to show a Modern DFFS form in a page. This can be handy if you want to show a new item form, or you want to let users edit a list item by for example passing them a link to an item in an email.

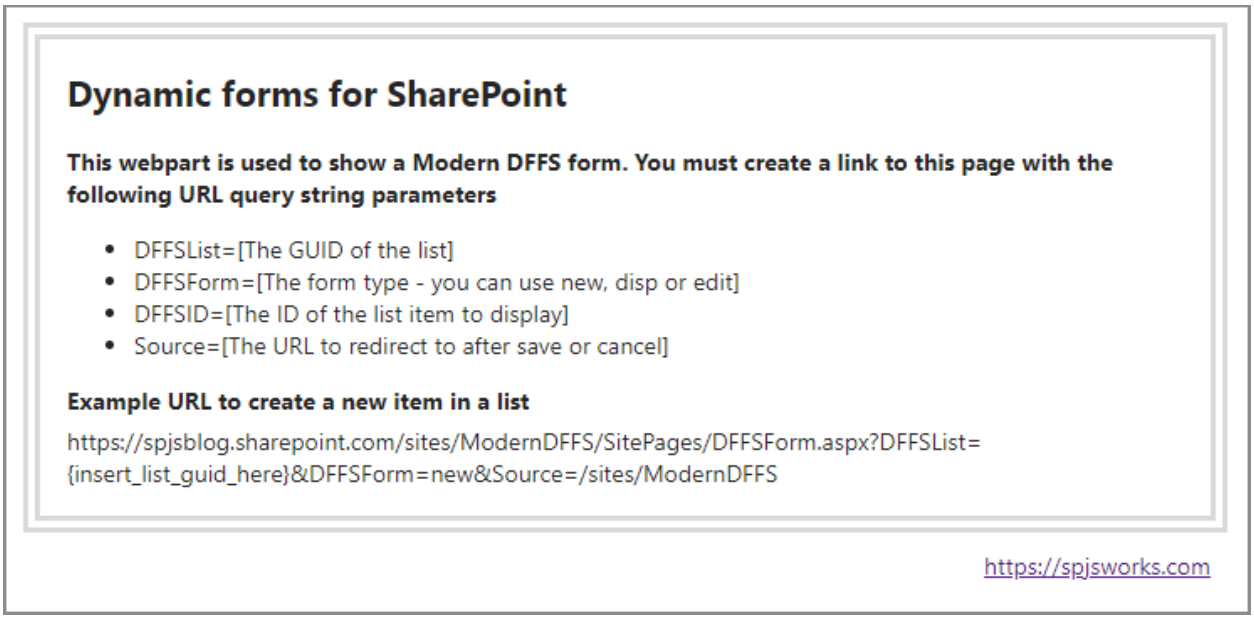
You can use the Moder DFFS WebPart to add, view or edit items in a list that has Modern DFFS Forms configured. To use it you must pass the **DFFSList** (GUID of the target list), the **DFFSForm** (new, disp or edit) and the **DFFSID** (if you want to display or edit an existing item). You can also add a **Source** attribute to have the user redirected to that address after they cancel or save the form.

See instructions when adding the web part.

Add the web part like this:



When you have added it, you can read the instructions on how to use it in the placeholder:



It is currently not possible to use this formpage when adding, viewing or editing a list item from a list view – only by using a custom link created on the format described above.

5.1 Setting a default redirect in the form URL

When using the Modern DFFS Formpage webpart you can use an URL parameter to se a default redirect. This can still be overridden in Custom JS if you use the function dffs_PostSaveAction or dffs_PostCancelAction.

Create the URL like this

https://spjsblog.sharepoint.com/sites/ModernDFFS/SitePages/DFFSForm.aspx?DFFSList={insert_list_guid_here}&DFFSForm=new&DFFSSource=/The_url_to_redirect_to

6 Upgrade from Classic DFFS

Use the ***Import / Export*** button to open the ***Import, export and restore configurations*** panel.

If you have the Classic DFFS version already in use on this site, you can browse the list of configurations and import them directly.

If you don't have the Classic DFFS version on this site, you must use the export functionality in the Classic DFFS and the import functionality in the Modern DFFS. This must be done one form at the time

Please note that because the Classic DFFS and the Modern DFFS are built with completely different technologies, the Custom JS and Custom CSS parts will not be imported. You can do all the same things in Custom JS and Custom CSS in the Modern version, but code that interacts with SharePoint lists (using CRUD) must be rewritten using the REST API because the *spjs.dffs* and *spjs.utility* namespaces are not available.

There are several functions built into the Modern DFFS that can be used, but the format is a bit different than in the Classic version. You can find details about these functions in the forum here: https://spjsblog.com/forums/topic/custom-js-examples/

Tabs and rules are imported, but because the rule configuration is a bit different you must manually go through the rules to ensure that they do what you want them to do.

Please note that you must manually uninstall the Classic DFFS version before configuring the list with the Modern DFFS to avoid conflicts.

7 Troubleshooting

7.1 Blocked web resources

If you are behind a company firewall you might have trouble loading some of the resources used in the Modern DFFS. For example, the Monaco-editor (the code editor used in Custom JS and Custom CSS) is loaded from cdn.jsdelivr.net. To use the code editor (and not have to edit the code as plain text) you must ensure that this CDN source is not blocked.

The license is loaded from an Azure CDN on this address: https://spjs.blob.core.windows.net. To be able to use the Modern DFFS you must also unblock this resource.

7.2 Other blocked resources

Because firewall rules are individually set up for different companies there might be other resources that must be unblocked in your company. If you have trouble loading the Modern DFFS you can use the developer tools to look for error messages in the Console or the Network tab.

If you have any questions please contact support: https://spjsworks.com/support