ALLAN THRAEN | 4 years ago | PDF |

Contentful CMS Tech Talk Integrations Addon Development

GETTING STARTED WITH CONTENTFUL UI EXTENSIONS - PART 1

Contentful has a handful of extension points, where you in a fairly straightforward and simple manner can extend the editorial experience with minimal development effort. In this post-series I'll show some examples of this.

I was recently invited to give a talk at a Contentful meetup in Copenhagen. As always I'm passionate about integrating and building addons for content management, the first topic that came to mind was to introduce newbies to the fun of building UI extensions that enhances the editorial experience within Contentful. This series of blog posts is based on that presentation.

Why extend

"Now, why would you want to extend a perfectly good editorial interface like Contentfuls in the first place?", I hear you ask. But the reason is pretty straight forward. The better editorial experience you can provide your content workers (=editors) with, the more efficient they'll be - and the better content they'll produce in the end. And better content SEO and/or more conversions which in the turn will lead to improved profits - just as the improved efficiency can lead to saved costs. The one thing to keep in mind, though is to ensure that your improvements are just that...Improvements. There are plenty of examples of addons and extensions that simply complicates things without solving any problems and results in monster of a CMS with dependencies through the roof.

Customizing editorial process

The process of customizing the editorial process goes beyong UI Extensions. In fact, one of the simplest ways to improve content quality is simply to add the proper validation and appearance to your fields. If a field _has_ to be one of a short list of values, don't just have a textbox, have a dropdown list to avoid confusion and save typing time. And if a field should always hold en email, ensure that it's validated to contain just that - and nothing else. However, it can be a tricky balance to maintain just enough validation and lockdown, while still empowering the editors to achieve their goals.

In Contentful there is quite an impressive API available, so it's also possible to achieve many editorial goals using that. For instance by connecting the webhooks and API to a service like Zapier or Microsoft Flow and thereby have content pushed to/from other sources. Or maybe connecting your blog section to a tool like Open Live Writer so the bloggers can use a desktop app to create content.

Finally, of course there is the approach of building UI Extensions which I'll describer in more details now.

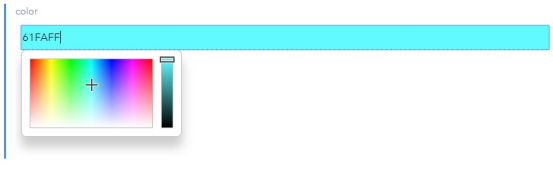
UI Extension Types

Since Contentful is a complete multitenant SaaS based system, all UI extensions are essentially relying on iframes with code executed in the browser to function. As this blog posts is being written there are essentially 4 main UI Extension Types:

- **Field Extension.** By far the most common. This is simply where you change the apperance of fields of a certain type used on a certain Content Type.
- **Entry Extension.** A fairly recent addition is the ability to change the apperance of the entire edit-interface for a certain content type.
- **Sidebar Extension.** Started off being just field extensions, visually placed in the right-hand sidebar. But now, it's also possible of adding them to a custom sidebar for any content type without being attached to a specific field.
- **Dialog Extension.** Not really an extension point in itself, but deserves a mention. Any of the above extensions can use custom dialogs to improve the user experience.

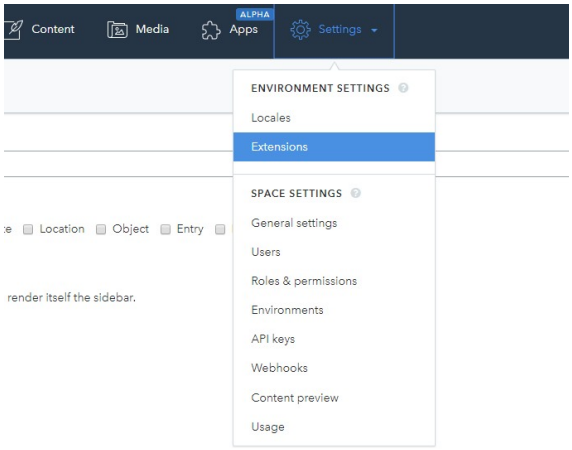
Extensions typically consist of at least 1 html file and 1 json file (and whatever dependencies they might need) and can be stored either in a storage provided by Contentful or hosted by yourself (or on a service such as Github).

Example: Color Picker Field



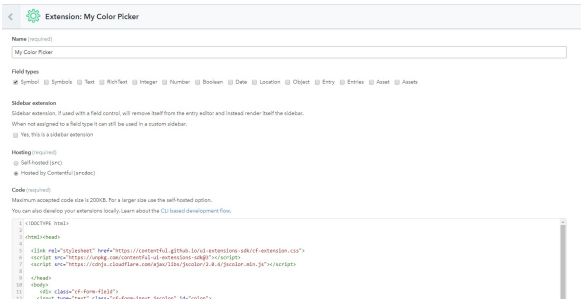
The first example I'll share is an ultra-simple color picker. Sometimes you want to give the editors the power to pick a color - for example a background color or maybe a text color. The best approach is probably to give them a dropdown with limited options that fit with your corporate visual identity. But we don't have time for rational solutions and this is more fun to make :-)

The simplest approach is to create the extension directly in Contentful. Simply go to Settings|Extensions and you can create a new extension just as easy as you would create a piece of content.



As mentioned, simply create a new extension, then give it a name and select which field types it should apply to, assuming your a creating a Field extension, that alters the apperance of a specific field. You can also leave the field types unchecked, and instead use your extension as an entry extension - or check the 'Sidebar' checkbox and it would be available in a custom sidebar.

Regarding the field types, one thing that had me baffled a little was the terminology: A symbol is just a short string (<256 chars). Text is a long string. Object can hold any json object and entry and asset are really just references. The plural form (Symbols, Entries, Assets) can hold an array of that type (Matching with the "This is a list" checkbox in the content type modelling tool).



Notice how you can pick to have it self hosted or hosted by Contentful (the default).

If it's hosted by Contentful, you can edit the source directly below, in the editor.

The code you are editing is basically an HTML file. The file typically as a minimum contain a javascript reference to the contentful-ui-extensions-sdk (hosted on a CDN). And it comes with a premade piece of javascript code that gives you an 'api' object with full access to all the operations you could want.

In the case of my Color Picker, I found the jscolor library and reference to that, as well as a reference to the default contentful UI extensions css file. I find this the simplest approach - but you can also chose to go down the path of using Forma 36 to get the full Contentful design.

This is my html/js code I entered:

```
<!DOCTYPE html>

<html><head>

  <!-- Our dependencies -->
  <link rel="stylesheet" href="https://contentful.github.io/ui-extensions-sdk/cf-extension.css">
  <script src="https://unpkg.com/contentful-ui-extensions-sdk@3"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/jscolor/2.0.4/jscolor.min.js"></script>

  </head>
  <body>
    <div class="cf-form-field">
    <!-- Here goes the input field, that we manipulate through javascript -->
    <input type="text" class="cf-form-input jscolor" id="color">
    <div class="cf-form-hint">Pick the background color </div>
  </div>


            window.contentfulExtension.init(function(api) {
              //api is the object giving us access to the Contentful API.

              //First, let's adjust the height of the iframe we are in, so it fits with what we need.
              //We could call api.window.startAutoResizer() but in this case I prefer to set a specif
              api.window.updateHeight(200);

              //For debugging purposes we can also output the value of the current field:
              console.log(api.field.getValue());

              //Let's get the input field from the DOM:
              var inputfield=document.getElementById('color');

              //We should set up an event listener for when it changes, so we can send the new value
              inputfield.addEventListener('change',function() {
                api.field.setValue(inputfield.value);
              });
```

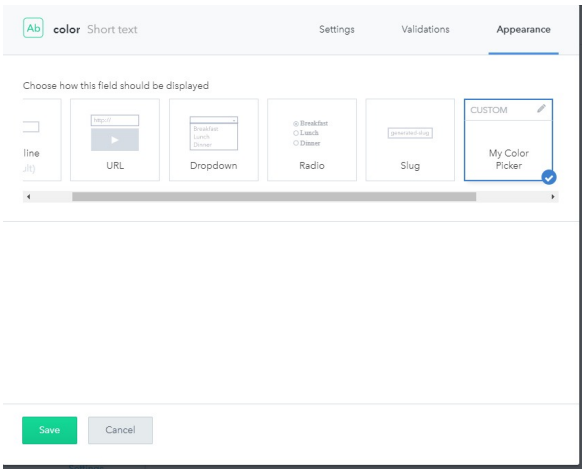
```
            //Also, what if something or someone else changed this value while we were editing it?
            //Better listen for that as well:
            api.field.onValueChanged(function(value) {
              if (value !== inputfield.value) {
                inputfield.value=value;
                inputfield.style.backgroundColor = '#' + inputfield.value; //Set the background colo
              }
            });

            //Finally, as we are still initializing here, let's update the input field with the valu
            if(api.field.getValue()!=undefined){
              inputfield.value=api.field.getValue();
              //And update the background color!
              inputfield.style.backgroundColor = '#' + inputfield.value;
            }
  <script>});
</body>
</html>
```

As you can see I've tried to fill the code with step by step comments. Pretty straight forward, right?

Now we can save our extension and start using it.

In order to use it on a field, we basically have to go to a content type, add a field of the right type (in this case short string) and select the Color Picker under apperance - and we're in business!



I hope you enjoyed this post! In the next post in this series we'll have a look at another Field extension that also takes certain predefined parameters - and is uploaded and managed through the CLI. Later, in the series we'll explore the sidebar extensions and see just how much power the javascript SDK gives you! Drop a comment below with questions or feedback - it's very much appreciated!

Contentful CMS Tech Talk Integrations Addon Development

RECENT POSTS

Second post in the series
Getting started with Contentful UI Extensions - Part 3

CodeArt ApS
Teknikerbyen 5, 2830 Virum, Denmark
Email: info@codeart.dk
Phone: +45 26 13 66 96
CVR: 39680688