Course #13

- How to modify the order of the fields in a table
- A new function request : match
- Widget style....

Module sed_table

- Order or fields in edit
- Using tabs to group fields
- Using Number of columns for display
- Validations
- Help
- Display
- Javascript
- Rights

Introduction to the module "Style Editor"

- The Style Editor allows an Extenso user to style his widgets by using an option panel without having to go in the code
- The panel generate CSS in the page
- The options of the option panel are defined in a JSON
- The JSON is attached to a widget

Exercises Objectives

- Be able to attach a Style Editor JSON to a widget you want
- Be able to modify the JSON with what you want
- The JSON allows the user to properly modify the correct styles of the widget
- Understand where to look to search for new Style Editor options/functionalities

Style Editor Presentation

- Open file : /extenso/module/sed/style/widget/style.json
- This is a proper Style Editor structure : Array of Selector Group Objects -> Array of Item Selector Objects -> Array of Declaration Objects
- Selector Group Keys :
  - "name" : The name of the group as seen by the user, multilingual ex. "name" : { "en" : "Global parameters...", "fr" : "Paramètres globaux" }
  - (optional) "initial_display" : Close the group accordion on first access with the value "closed" (makes the interface less cluttered)

Style Editor Presentation

- (optional) "rule" : Add a CSS rule that will encapsulate all the selectors, example for media queries or animations ex. "rule" : "@media(max-width:767px)"
- (optional) "id" : Will override the automatic ID generator for the group in the backend (IDs are used in the backend to tie the values to the CSS properties, if you don't put an ID and you change the "rule" : "@media(max-width:767px)", the values the user already entered will be lost since the automatic ID generated will now be different)
- "item" : An array of item CSS selectors

Style Editor Presentation

- Item Selector Keys :
  - "name" : Name of the selector, ex. "name" : { "en" : "Send Button", "fr" : "Bouton Envoyer" }
  - "selector" : CSS selector, the scope is restricted to the widget ID ex. "selector" : "img" will only select <img> in that specific widget instance (in practice it will look like "#sn_widget_5060 img")
  - "declaration" : An array of CSS properties

Style Editor Presentation

- Declaration Keys :
  - (optional) "name" : Name of the property to display, if the property is already known by the style editor module and this key is empty, it will use the default translation
  - "property" : CSS property ex. "property" : "background-color"
  - (optional) "id" : Will override the automatic ID generator for the property in the backend, must be unique, will allow to keep the user value if you change the selector
  - (optional) "display" : Will override how the the field in the option panel will display, ex. "display" : "select", for unrecognized properties, the default display is a text input

Style Editor Presentation

  - Example of possible display : "select", "color_picker", "size_unit", "border", etc. refer to file "/extenso/module/sed/style/package/p_style.sn" for complete list (every function that starts with "call_")

- (optional) "help" : HTML that contains additional information for the property to give to the user, accessed by clicking the help icon

Exercise #1

- Make a style editor for your custom widget
- Go in the entry of your widget in Dev > Widget development > Widgets > Your widget
- Go in the "Css" tab, put a path of the file in the field "widget JSON css code", JSON files end with ".json" (you should probably just copy the path of your widget and replace the ".sn" with ".json")

Exercice #1 Paste this JSON code

```
[
    {
        "name" : { "en" : "Group", "fr" : "Groupe" },
        "item" : [
            {
                "name" : { "en" : "Affecting the whole widget",
                           "fr" : "Affecte tout le widget" },
                "selector" : "",
                "declaration" : [
                    {
                        "property" : "color"
                    },
                    {
                        "property" : "border"
                    }
                ]
            }
        ]
    }
]
```

Exercice #1

- Save widget
- Install widget on a page / go to a page with your widget (re-generate your page)
- Select your widget, in the top right corner of the right panel, open "Element specific styles", you can now edit some styles from this interface
- To make the changes last on the page, you must re-generate the page

Exercise #2
Modify your style editor

- Open your JSON file in your IDE
- Modify the selector key to ":hover"
- Add the "opacity" property
- Go back in Extenso, if you were already selecting your widget, select something else then re-select your widget, your Style Editor will now be updated with your new values

Exercise #3
Make default styles for your widget

- Input the values that you want to be present when your install the widget
- In the top menu in the Style Editor panel, click "Save all styles"
- Name your style (you can make multiple styles and load them for this widget)
- Go in Site Dev > Site data > List of saved style
- Usually the last one is your style, check the "Extenso style" and "Default style"
- Re-install your widget, you will now see it has already your styles

Exercise #4
Make styles apply only for certain breakpoints

- Open your JSON file in your IDE
- Add a group (you can copy your group object and separate them with a comma, ALWAYS properly indent)
- Add the "rule" key to the group and add your media query
  - Bootstrap breakpoint :
    - Phone portrait : @media(max-width:575px)
    - Phone landscape and smaller : @media(max-width:767px)
    - Tablet and smaller : @media(max-width:991px)
    - laptop and smaller : @media(max-width:1199px)
- Change the styles, your styles will apply only when equal or under the specified window width