JavaScript Dialog Fields

This module allows you to create custom form fields with JavaScript. No need for custom java code.

JIRA	N/A
Git	ui-framework-javascript

Installation

Maven is the easiest way to install the module. Add the following dependency to your bundle:

```
<dependency>
  <groupId>info.magnolia.ui</groupId>
  <artifactId>magnolia-ui-framework-javascript</artifactId>
  <version>${ui-framework-javascript.version}</version>
</dependency>
```

Versions

1.1.2	**Magnolia 6.2**

Configuration

Add a JavaScript field in your dialog definition as follows:

```
label: Home page
form:
  implementationClass: info.magnolia.ui.javascript.form.FormViewWithChangeListener
  properties:
    colorField:
      label: Background color
      $type: javascriptField
      fieldScript: /my-light-module/webresources/colorField/index.html
      height: 70
          defaultValue: "#00ff00"
      parameters:
        foo: bar
```

Properties:

- `$type`: Needs to be `javascriptField`
- `fieldScript`: Points to a valid HTML file located in any module (Maven & Light modules)
- `height`: The height of the field in the dialog
- `defaultValue`: The field default value
- `parameters`: Allows passing custom parameters to the Javascript field

> Don't forget to set your `defaultBaseUrl` in the `/server@defaultBaseUrl` configuration.

Integration

The only required element to write your own JavaScript field is an HTML file located in a Magnolia module.

This is an example of a color picker:

```
<html>
  <body>
    <input id="colorField" type="color" value="#ffffff">

    <script>
      let colorPicker = document.getElementById('colorField');
      let correlationId = null;

      // Add change event listener
      colorPicker.addEventListener("change", function (event) {
        parent.window.postMessage({
                  action: "changeValue",
          correlationId: correlationId,
          value: event.target.value
        }, "*");
      });

      // Assign handler to message event
      window.addEventListener("message", function (event) {
        if (event.data.correlationId) {
          let action = event.data.action;
          correlationId = event.data.correlationId;

          if (action==="init") {
            if (event.data.state.value) {
              colorPicker.value = event.data.state.value;
            } else if (event.data.state.defaultValue) {
              colorPicker.value = event.data.state.defaultValue;
            }
          } else if (action==="error") {
            colorPicker.style.backgroundColor = "red";
          }
        }
      }, false);
    </script>
  </body>
</html>
```

To init the field configuration and value, Magnolia sends a message to the iFrame loading the JavaScript field. That is done in the `window.addEventListener("message", function (event) {...}` statement. It is very important to store the `event.data.correlationId` for further communication with Magnolia.

- `action` can have different values:
    - `init` to initialize the field properties and values
    - `error` to notify the field failed the validation (either via the required flag or a validator)
- `correlationId` is very important, it will prevent mixing up messages between the different Javascript fields.
- `state` represents the data exchanged between the Backend field and the Javascript field.

To update the field value on the Magnolia side, the Javascript field needs to send back the value. That is done in the `colorPicker.addEventListener("change", function (event) {...}` statement.

- The `correlationId` is very important, it will prevent mixing up messages between the different Javascript fields
- The `value` hosts the updated value of the field. The type of the value is always a `String`. If you need to store complex objects, stringify it before returning it to Magnolia.

Warnings

- This module is at INCUBATOR level.

Changelog

- Version 1.1.1
    - Bug fixes
- Version 1.1
    - Add the message `action`
    - Add the error notification
- Version 1.0 - Initial release of the module.