jahia

January 2023 Webinar

# Customizing Jahia UI using React Components

# Speakers

## Serge Huber

CTO, Jahia Solutions

## Hervé Duchesne

Senior Solution Architect

jahia

# Agenda

01 Intro

02 Overview & Examples

03 Concepts

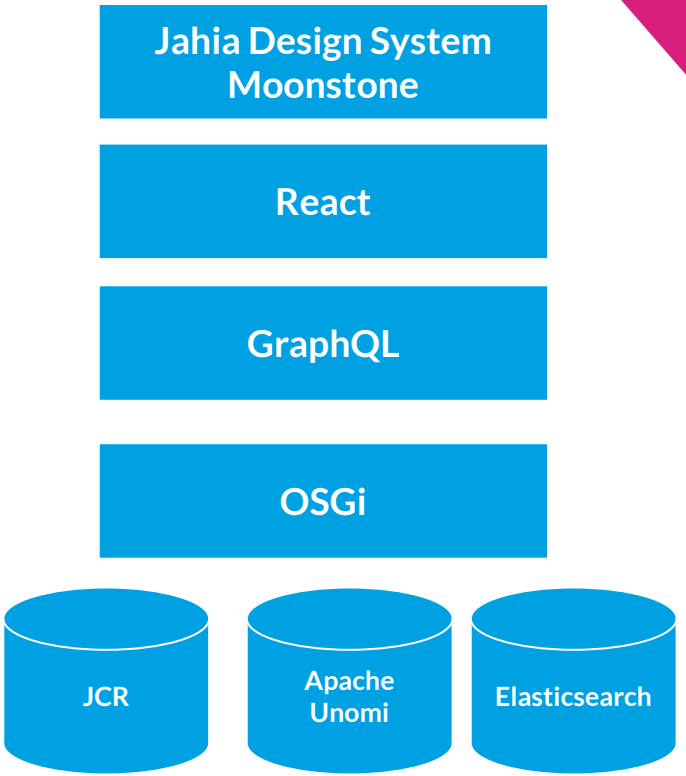
04 Live Coding of a content editor selector type

05 Q & A

# Jahia 8 UI

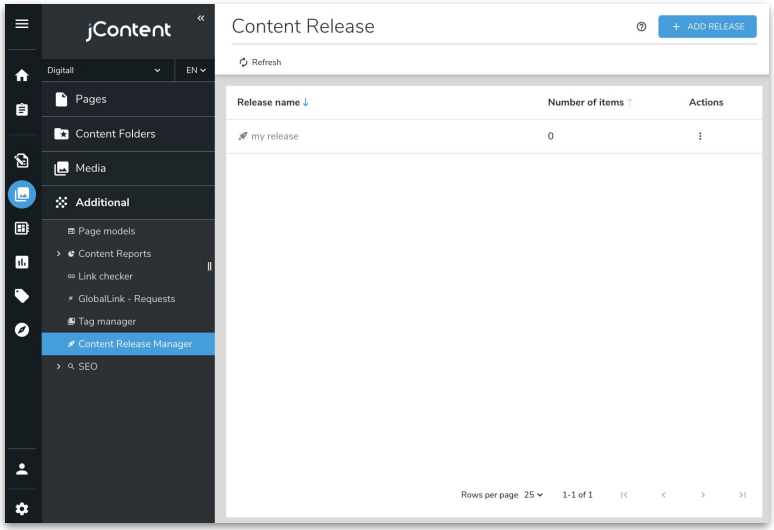
- Most of Jahia 8 UI is using React 16 instead of GWT
- Make it possible to extend the UI in a lot of ways
- Make it possible to modify the provided UI
- Make it all possible from modules / modular
- Use latest front-end technologies:


## Jahia UI Tech Stack



# Examples

# Adding a new panel in jContent

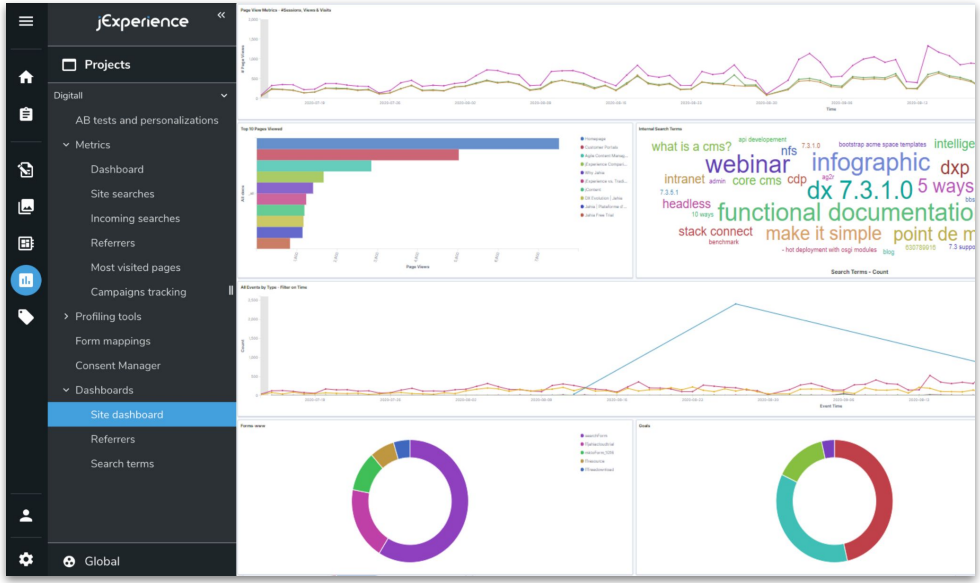



- Navigation entry is a React component
- The main display is a full React app using jahia-moonstone design system
- The app is built automatically when the module is installed
- Using yarn webpack react apollo
- Nice code sample for our datatable component usage

[See on Github]

[See on the store]

# Adding a new panel in jExperience

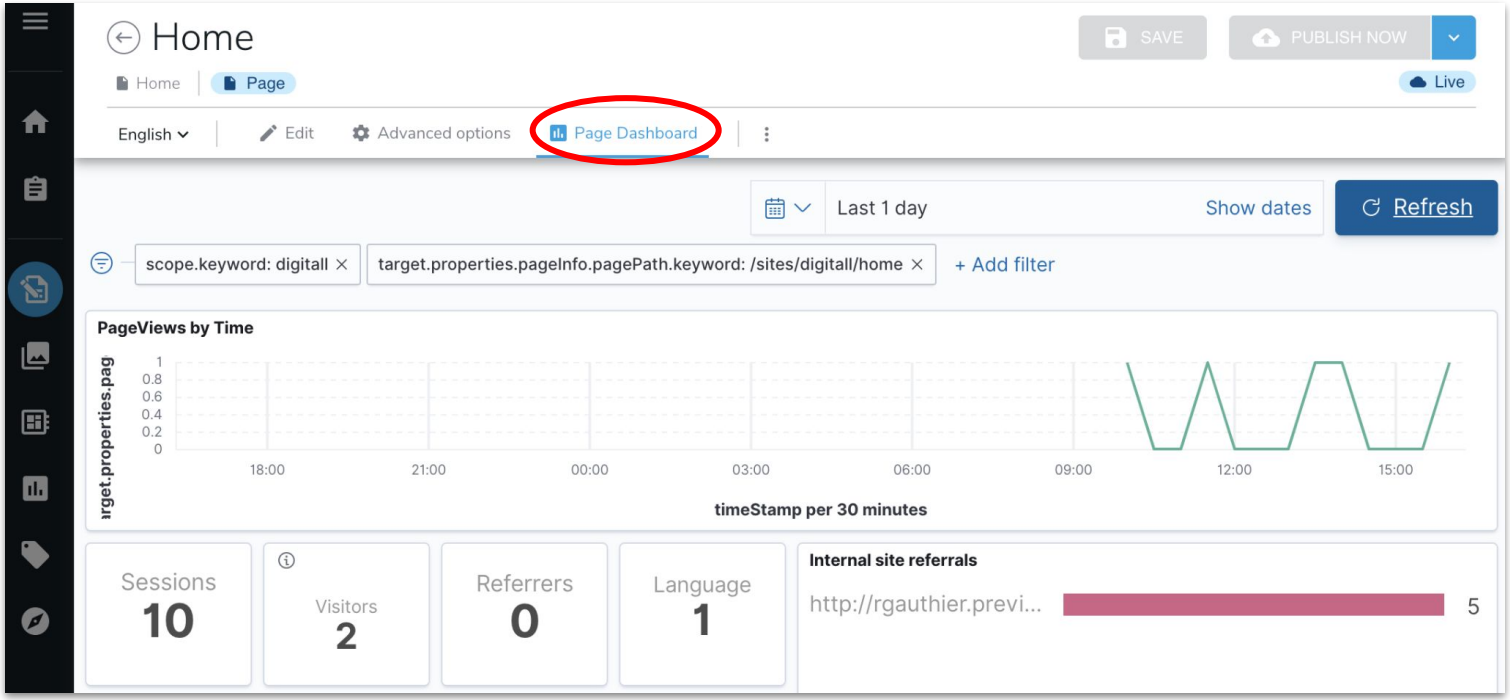



- Navigation entry is a React component
- The main display is an iFrame calling Elastic Kibana dashboard



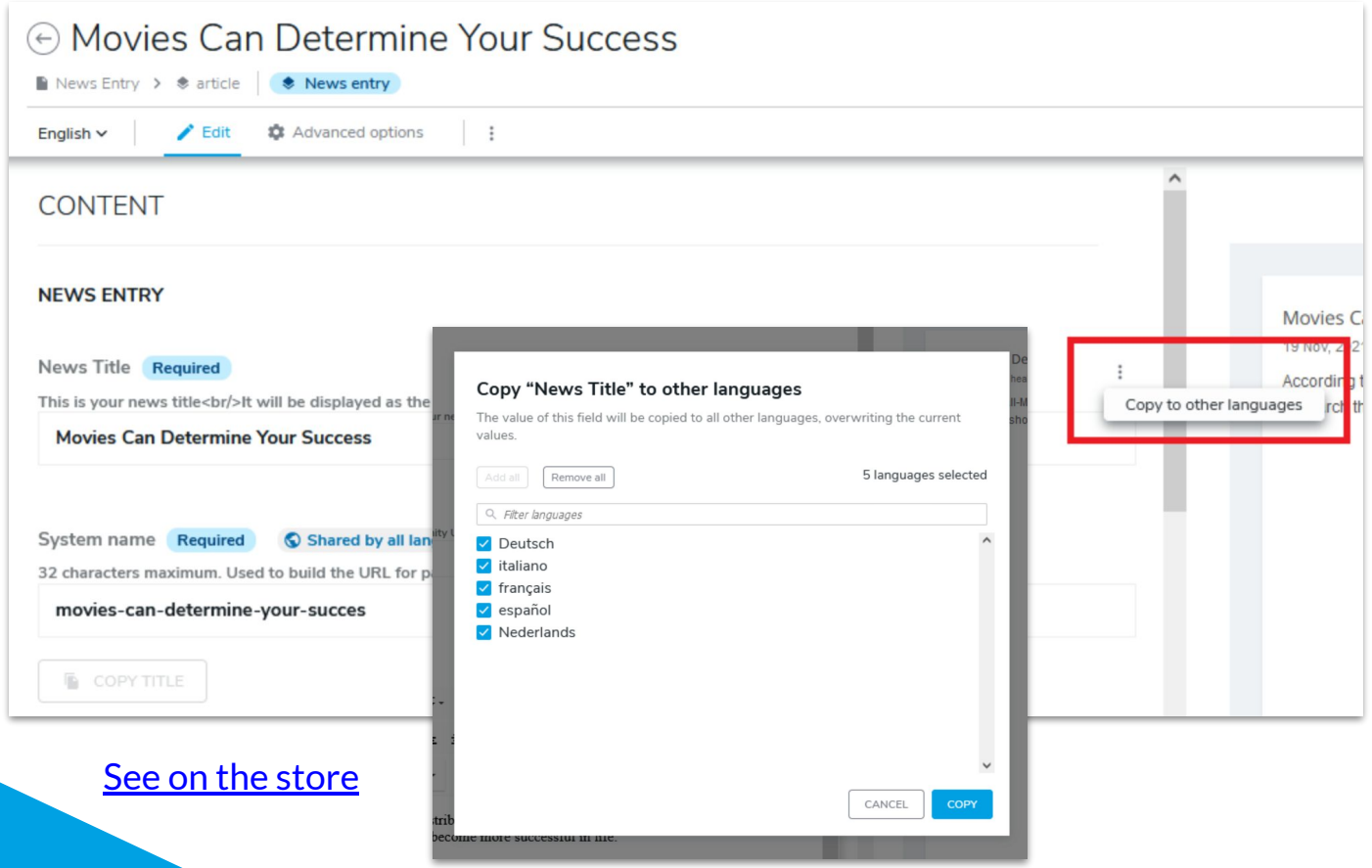
jahia

# Adding a tab in Content Editor



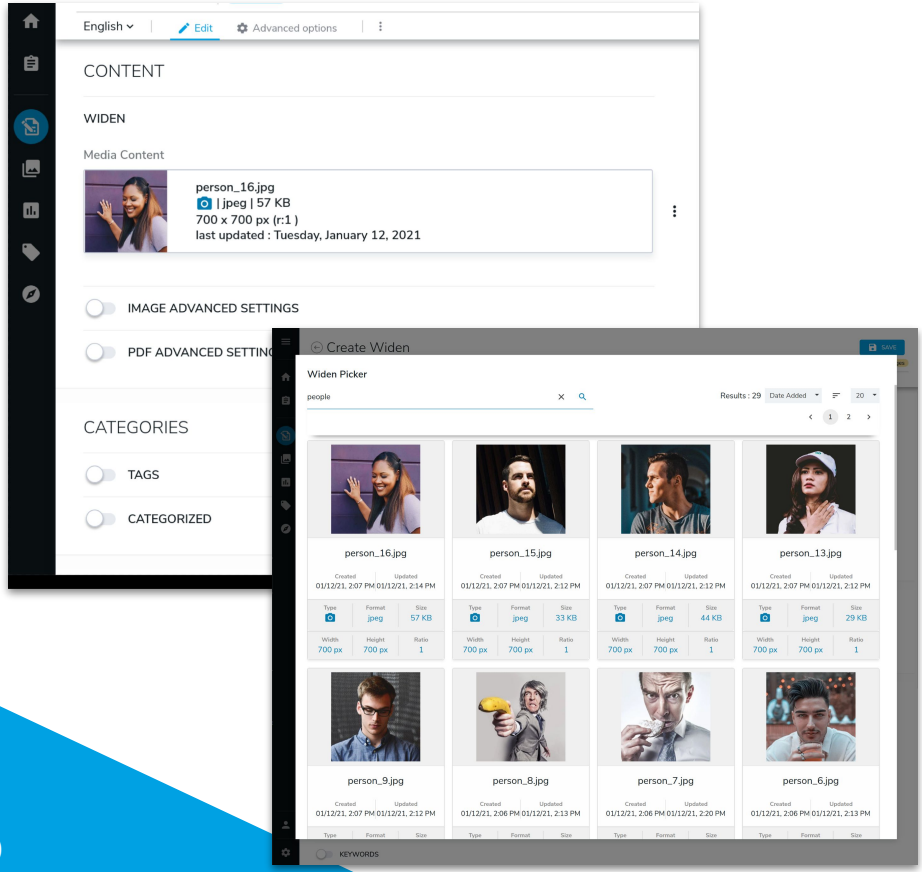

- The main display is an iframe calling an Elastic Kibana dashboard

# Adding an action in Content Editor




[See on the store]

# Adding a new input type in Content Editor




- The input field and the picker are rendered by a React app
- The picker browses the DAM using the DAM's rest api capabilities
- EDP is used to register in real time the selected image

[See on Github]

[See on store]

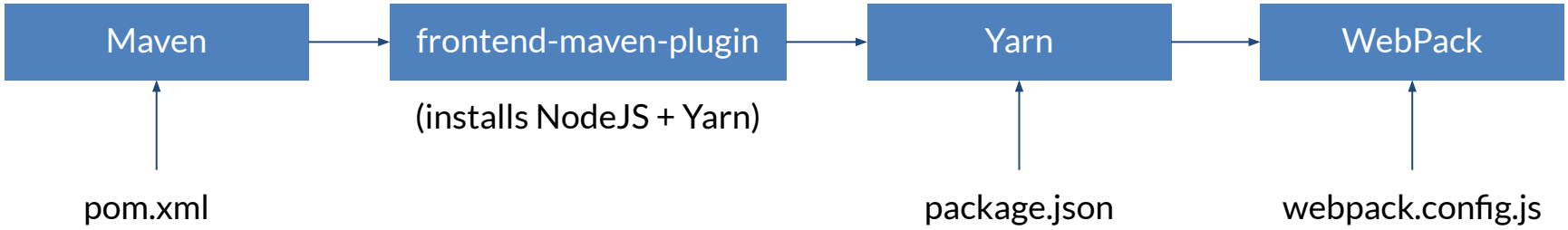
To know more, have a look at a dedicated video in the last section of our [partner portal]



jahia

# Building and packaging

# Module building & packaging




Java

Javascript

jahia

# frontend-maven-plugin

- Acts as a bridge between Maven builds and Yarn/NPM builds
- Will automatically install the configured NodeJS/NPM/Yarn versions
- Will run npm or yarn install to download dependencies & build project

```
<plugin>
    <groupId>com.github.eirslett</groupId>
    <artifactId>frontend-maven-plugin</artifactId>
    <version>1.6</version>
    <executions>
        <execution>
            <id>npm install node and yarn</id>
            <phase>generate-resources</phase>
            <goals>
                <goal>install-node-and-yarn</goal>
            </goals>
            <configuration>
                <nodeVersion>v16.15.0</nodeVersion>
                <yarnVersion>v1.22.10</yarnVersion>
            </configuration>
        </execution>
        <execution>
            <id>yarn install</id>
            <phase>generate-resources</phase>
            <goals>
                <goal>yarn</goal>
            </goals>
        </execution>
        <execution>
            <id>yarn post-install</id>
            <phase>generate-resources</phase>
            <goals>
```

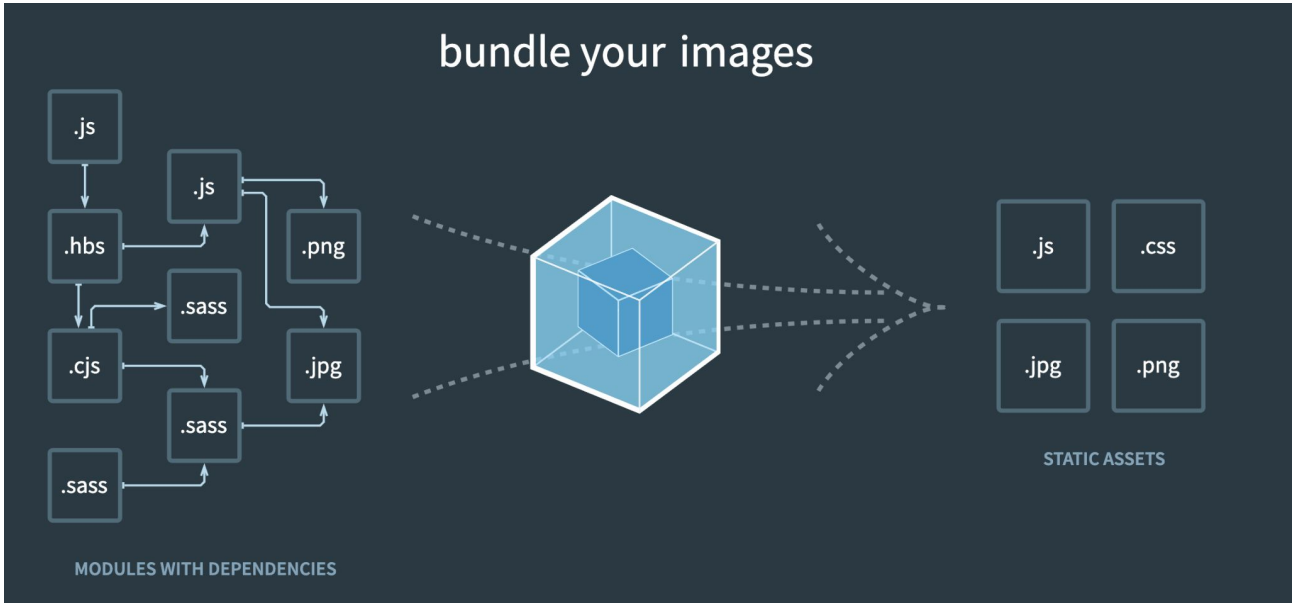
# Yarn

- (Javascript) Package manager similar to NPM
- Optimized for speed & build reliability
- NPM has evolved to include some of Yarn improvements

| Features | npm | |
|---|---|---|
| **Generating Lock Files** | Automatically generated as **package-lock.json** | Automatically generated as **yarn.lock** |
| **Using Workspaces** | Supported | Supported |
| **Remote Scripts** | Supported, using the **npx** command | Supported, using the **`yarn dlx`** command |
| **Plug'n'Play** | Not supported | Uses a single **.pnp.cjs** file instead of the *node_modules* folder |
| **Zero Installs** | Not supported | Uses the **.pnp.cjs** file to quickly reinstall packages from the offline cache |
| **License Check** | Not supported | Checks each package license while downloading |

jahia

# WebPack




jahia

# Webpack configuration

```
// const requires

module.exports = (env, argv) => {
   let config = {
       entry: // Javascript app entry points for dependency analysis
       output: // Javascript app output bundle locations
       module: {
           rules: [
...
               {
                   test: /\.jsx?$/,
                   include: [path.join(__dirname, 'src')],
                   use: {
                       loader: 'babel-loader',
                       options: { ... }
                   }
               },
...
           ]
       },
       plugins: [
           new ModuleFederationPlugin({
               name: "dashboard",
               library: { type: "assign", name: "appShell.remotes.dashboard" },
               filename: "remoteEntry.js",
               exposes: {
                   './init': './src/javascript/init',
               },
               remotes: {
                   '@jahia/app-shell': 'appShellRemote',
               },
               shared
           }),
```

Module federation, makes it possible to build “modules” with Webpack that will avoid repackaging code and build reusable libraries instead

[Learn more about module federation]

jahia

# Focus on the Component Registry

# Javascript UI Component registry

- Javascript Dynamic registry of UI components
- Inspired by OSGi services registry used in the Java modules
- Jahia navigation uses component registry to know what to display
- jContent uses registry to know which actions to display for the current user
- Content Editor uses registry to know how to render fields
- Other components use registry components to be extendable/modifiable
- Accessible through a Javascript import statement or globally using `window.jahia.uiExtender.registry`

```
class Registry {

    addOrReplace(type, key);

    add(type, key);

    get(type, key);

    remove(type, key);

    find(filters);

}

let registry = new Registry();

export {registry};
```

# Main component registry types

- Common types are:
  - action: use to add an UI action in content editor
  - selectorType : add a selector type in content editor
  - adminRoute : add a route in the administration
  - route : add a route in for a custom panel
  - primary-nav-item : add an entry in the primary level navigation
  - callback : adds a function to be called when the app shell initializes
- Modules may add their own types for their own needs



jahia

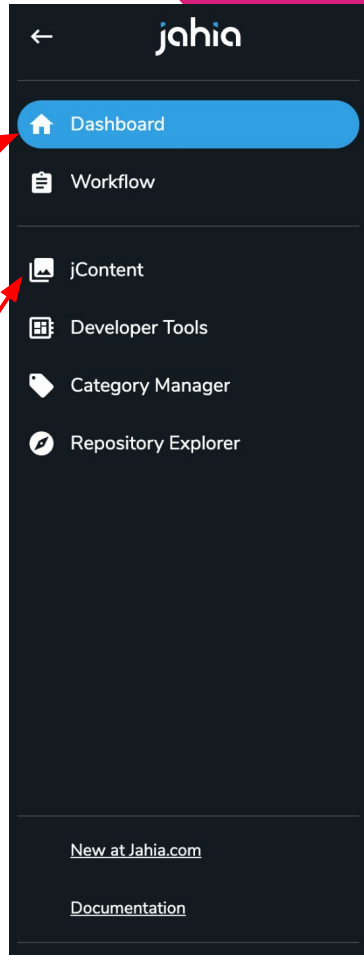
# Component registry examples

- Dashboard navigation entry

```
import {registry} from '@jahia/ui-extender';

registry.add('primary-nav-item', 'dashboardGroupItem', {
   targets: ['nav-root-tasks:1'],
   render: () => <DashboardGroup/>
});
```

- jContent navigation entry

```
import {registry} from '@jahia/ui-extender';

registry.add('primary-nav-item', 'jcontentGroupItem', {
   targets: ['nav-root-top:2'],
   requiredPermission: 'jContentAccess',
   render: () => <CmmNavItem/>
});
```



# Registry targets

```
registry.addOrReplace('action', 'createContent', createContentAction, {
   buttonIcon: <AddCircle/>,
   buttonLabel:
       'content-editor:label.contentEditor.CMMActions.createNewContent.menu'
   targets: ['createMenuActions:3', 'contentActions:3', 'headerPrimaryActions:1'],
   showOnNodeTypes: ['jnt:contentFolder', 'jnt:content'],
   hideOnNodeTypes: ['jnt:navMenuText', 'jnt:page'],
   requiredPermission: ['jcr:addChildNodes'],
   isModal: true
});
```

- Allows registration of components in multiple targets
- Allows sorting components within targets

```
...registry.find({type: 'action', target: 'headerPrimaryActions'}).map(action => action.key),
```

```
let contentActions =registry.find({type: 'action', target: 'contentActions'});
```

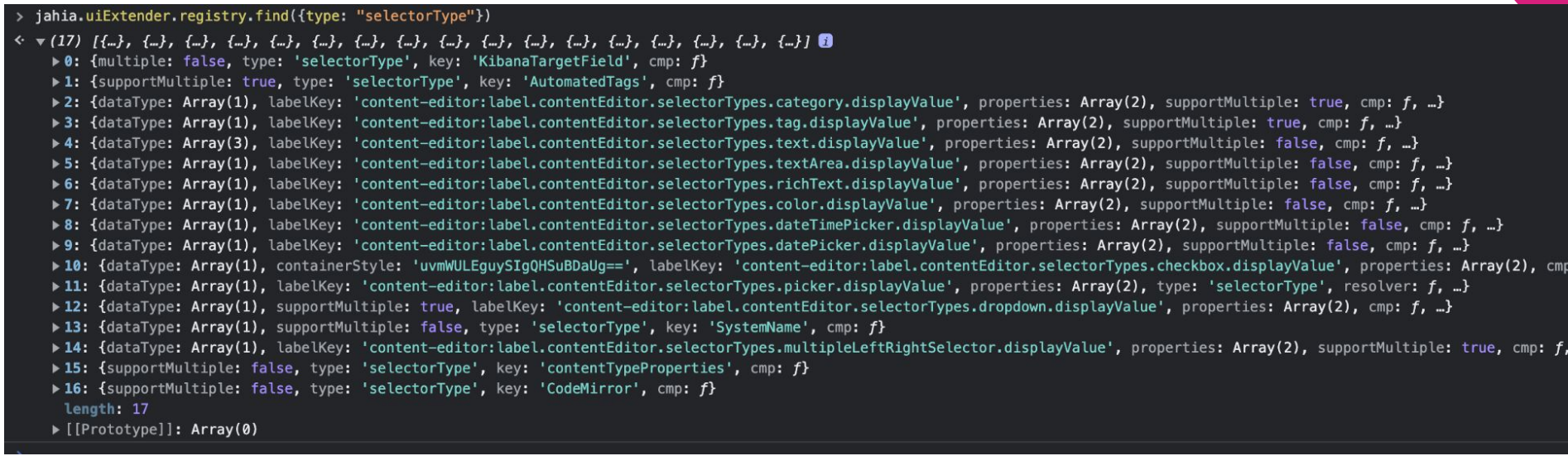
jahia

# Navigation iFrame URLs

- Use in Jahia for:
  - Site settings
  - Server settings
  - jExperience UI

```
registry.addOrReplace('jExperienceMenuEntry', 'siteDashboard', {
    targets: ['siteMetrics:1'],
    id: 'siteDashboard',
    labelKey: 'jexperience:jexperience.siteMetrics.siteDashboard.title',
    isSelectable: true,
    iFrameUrl: window.contextJsParameters.contextPath +
'/cms/editframe/default/$lang/sites/$site-key.marketing-10-dashboard.html',
    treeItemProps: {
        'data-sel-role': 'siteDashboard'
    }
});
```

# Component registry debugging

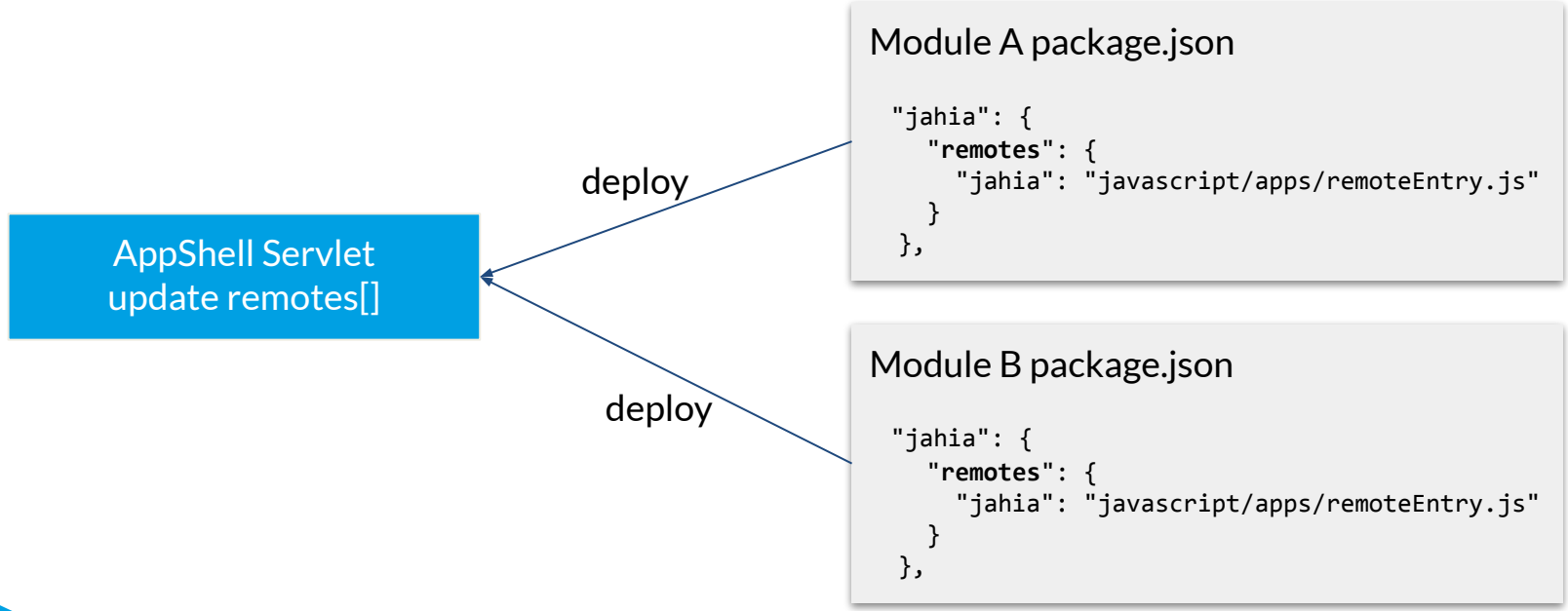



jahia

# Loading & routing

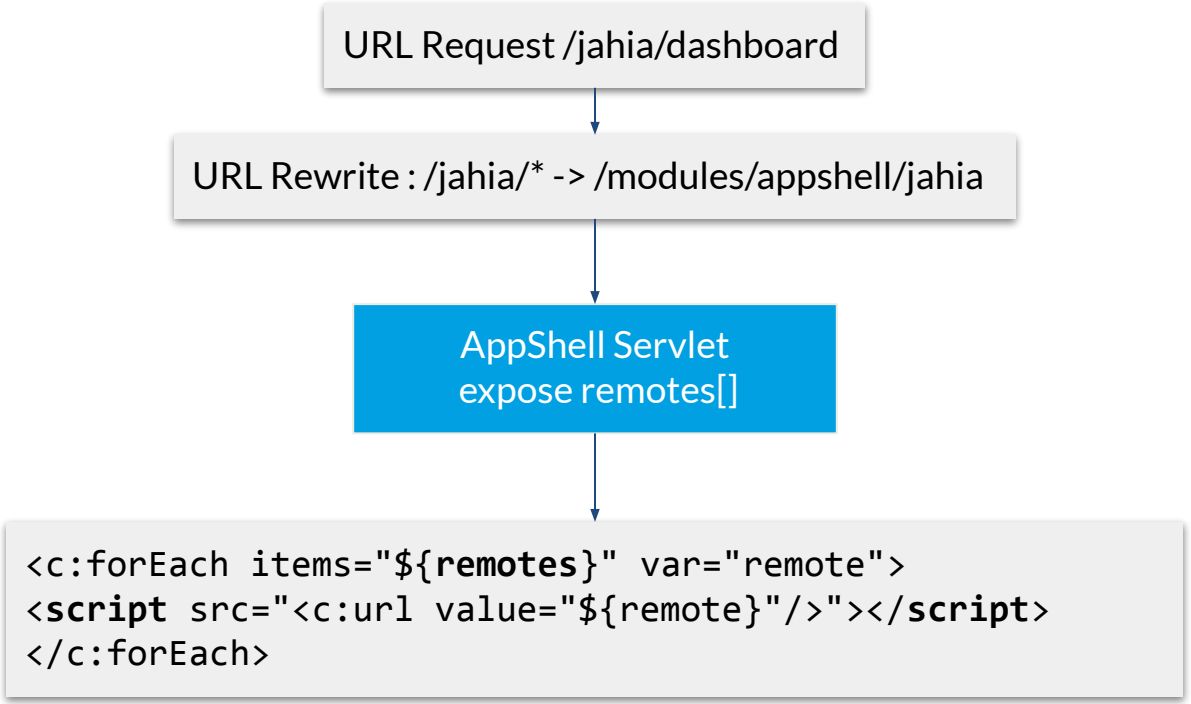
# Javascript UI Registration

or how modules declare new Javascript scripts to be used in UI



# Javascript UI Loading

or how Jahia knows which Javascript files to load on a page




jahia

# Javascript UI routing

react-router v5 used

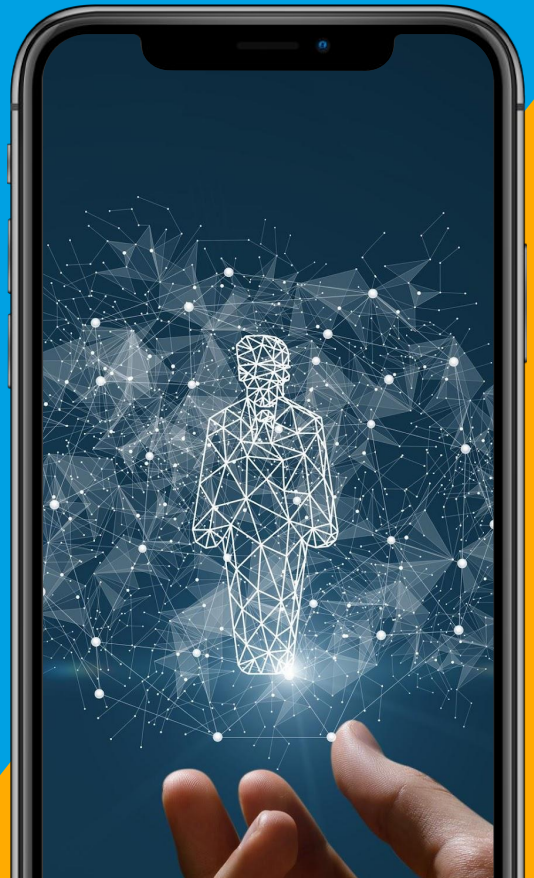
```
export const jContentRoutes = registry => {
   registry.add('route',
'jcontent-search-route', {
       targets: ['jcontent:99'],
       path:
'/jcontent/:siteKey/:lang/search',
       component: ContentRoute
   });

   registry.add('route',
'jcontent-sql2Search-route', {
       targets: ['jcontent:99'],
       path:
'/jcontent/:siteKey/:lang/sql2Search',
       component: ContentRoute
   });
```

```
export const JContent = () => {
   const routes = registry.find({type: 'route', target:
'jcontent'});

   return (
       <LayoutModule
           navigation={<ContentNavigation/>}
           content={
               <LoaderSuspense>
                   <ErrorBoundary>
                       <Switch>
                           {routes.map(r => r.component ? (
                               <Route key={r.key}
                                      path={r.path}
                                      render={p =>
<ErrorBoundary>{React.createElement(r.component,
p)}</ErrorBoundary>}
                                      />
```

https://v5.reactrouter.com/web/guides/quick-start

jahia

# Leveraging Jahia features



# Leverage UI features

Custom components can leverage

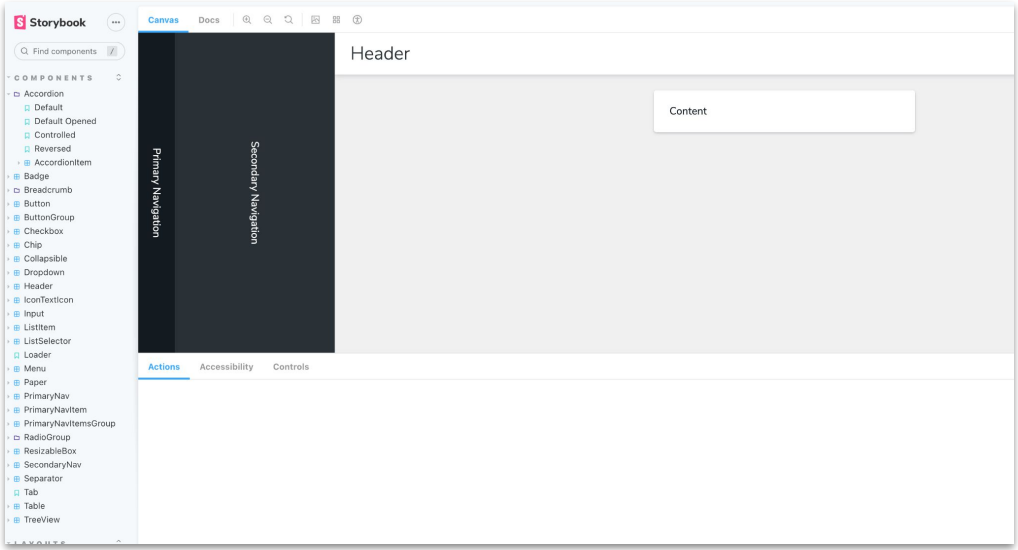
- Authentication
- Permissions
- (Minor) Transparent upgrades of Jahia UI
- Component registry
- Existing Jahia components
- Maven packaging
- Internationalization
- Content Editor form overrides

```
import {registry} from
'@jahia/ui-extender';

registry.add('primary-nav-item',
'jcontentGroupItem', {
   targets: ['nav-root-top:2'],
   requiredPermission: 'jContentAccess',
   render: () => <CmmNavItem/>
});
```



jahia

# Moonstone: Jahia UI Design system




https://jahia.github.io/moonstone/

- Still under active development
- APIs may still evolve, but mostly stable
- Not covered by Jahia Support

jahia

# Internationalization

Component JSX

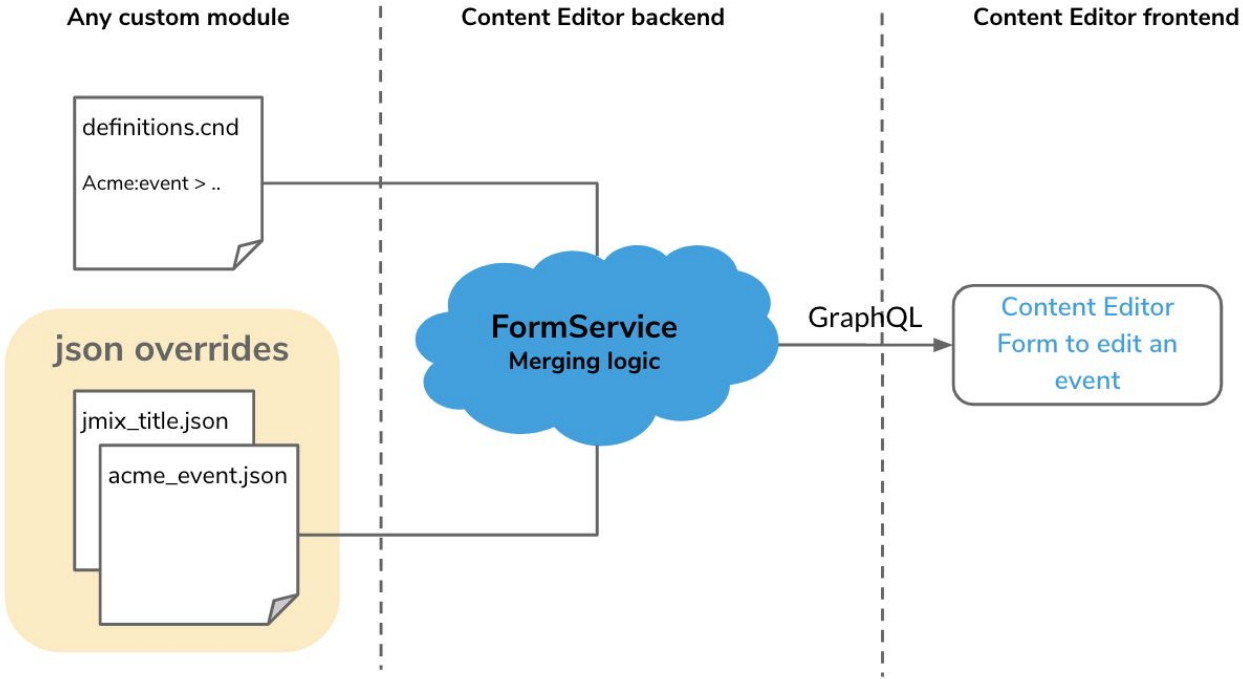
```
import React from 'react';
import {useTranslation} from 'react-i18next';
import {PrimaryNavItem} from '@jahia/moonstone';
import Constants from './Dashboard.constants';

export const DashboardGroup = () => {
   const {t} = useTranslation('jahia-dashboard');
   …
   return (
       <PrimaryNavItem key={Constants.ROUTE}
                       label={t('jahia-dashboard.label')}
       />
   );
};
```

src/main/resources/javascript/locales/en.json

```
{
 "jahia-dashboard": {
   "label": "Dashboard",
```

jahia

# Content Editor Form overrides



# Examples

- Remove fields from content editor UI
- Re-order fields in content editor
- Replace content editor field selector types

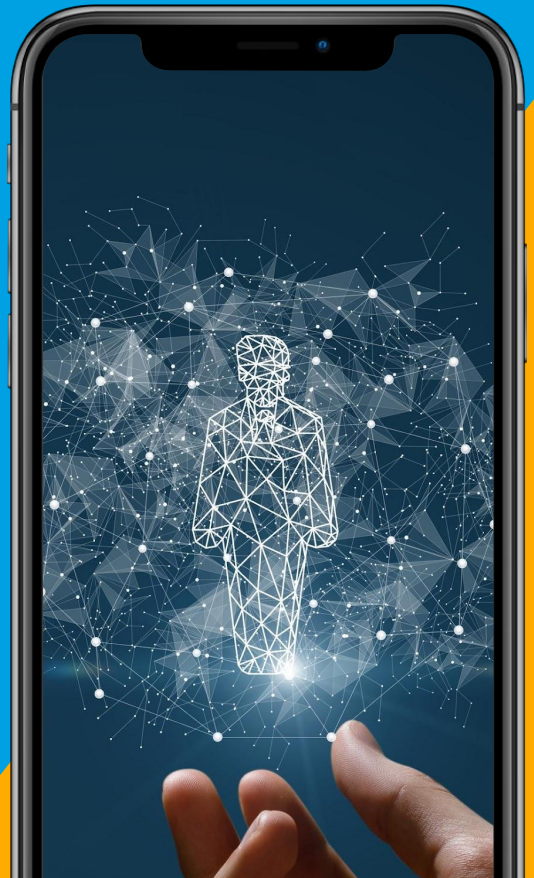
```
{
  "name": "game4mix:damVideoLink",
  "fields": [
    {
      "name": "game4:videoIntPath",
      "selectorType": "WidenPicker"
    }
  ]
}
```

```
{
 "name": "jnt:page",
 "priority": 1.1,
 "fields": [
   {
     "name": "j:templateName",
     "target": {
       "rank": -1
     }
   }]
}
```



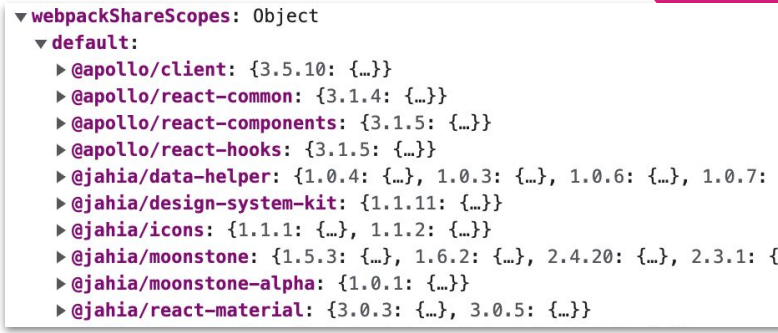
jahia

# Development tips & tricks



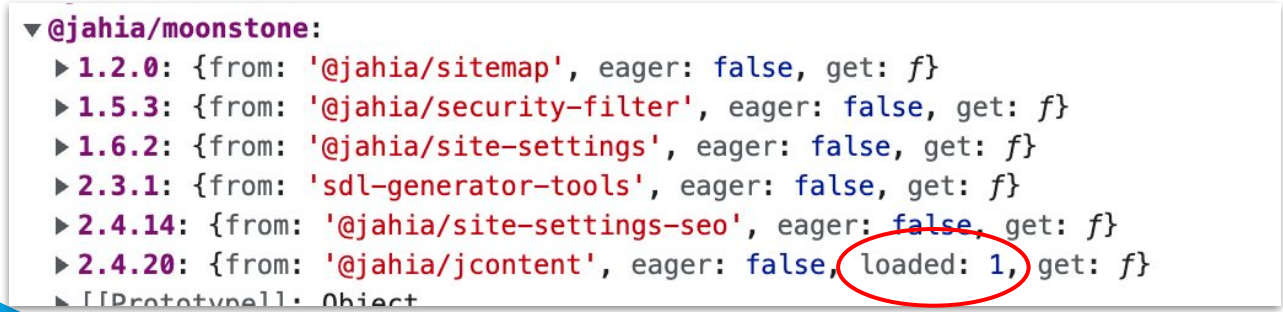
# Frameworks & versions

- Check which library are registered and where they are coming from
  - In browser Javascript console inspect `jahia.webpackShareScopes.default`



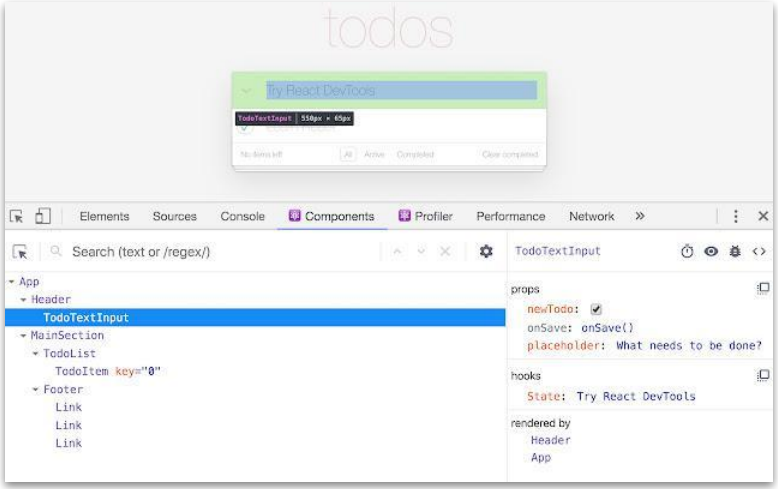

- Check which version is actually loaded using
  - In browser Javascript console, execute:

```
Object.entries(jahia.webpackShareScopes.default['@j
ahia/moonstone']).find(f => f[1].loaded)
```



# JS Developer tools

- Chrome Dev Tools
- React Developer Tools
- Redux Tools
- webpack development build (default) vs production
- Jahia Deploy Free coding & webpack watch
- Apollo GraphQL Chrome Tools



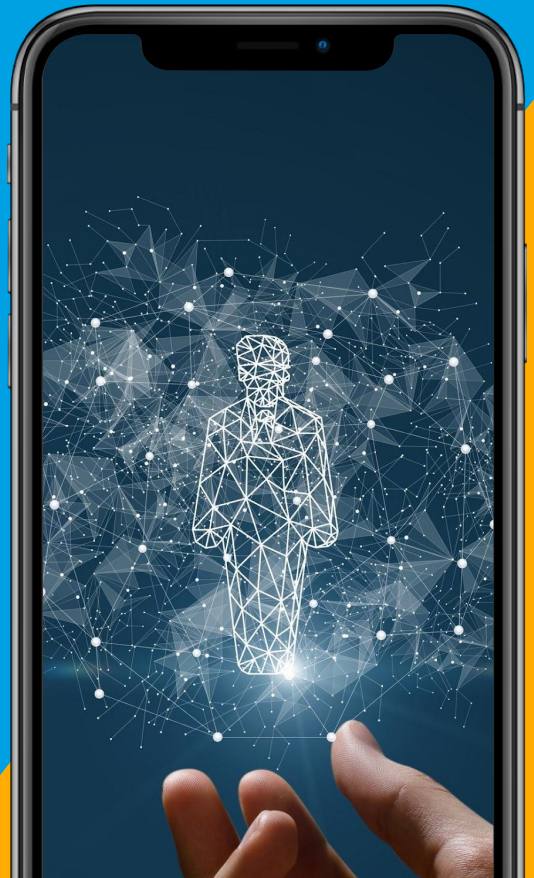

```
"scripts": {
  "test": "env-cmd --no-override jest",
  "testcli": "jest",
  "build": "yarn lint:fix && yarn webpack",
  "build:nolint": "yarn webpack",
  "dev": "yarn webpack --watch",
  "watch": "yarn webpack --watch",
  "webpack": "node --max_old_space_size=2048 ./node_modules/webpack/bin/webpack.js",
  "build:analyze": "yarn build --analyze",
  "build:production": "yarn build --mode=production",
```

# Steps to build a UI extension

1. Create/copy/clone a Maven project
2. Configure Webpack using webpack.config.js
3. Configure package.json
4. Add init.js file to register your components
5. Implement your components
6. Register them in the component registry
7. If your components are using in the content editor forms, provide JSON override files

jahia

# Live coding



# Context

## Marketing team wants to use Jahia to create and deliver a Quiz

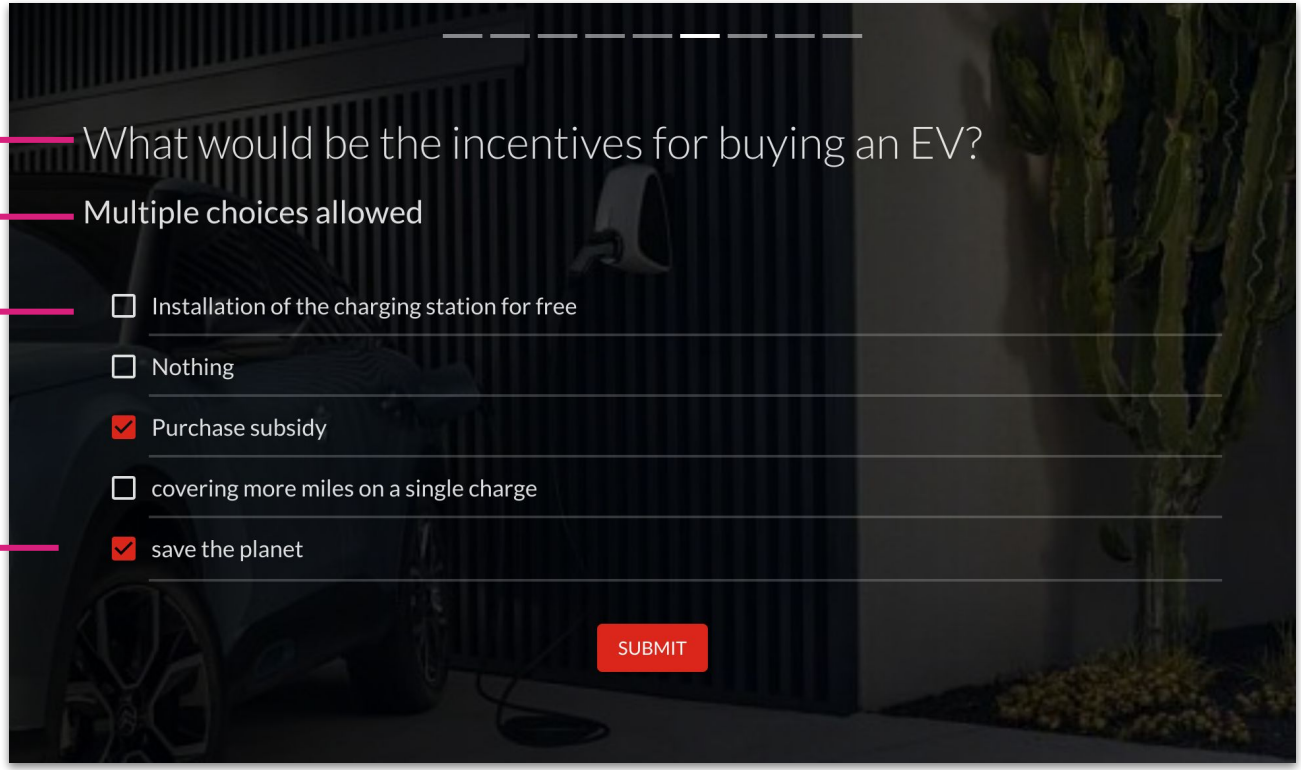
**QnA Content Properties**

Question

Help

Answers

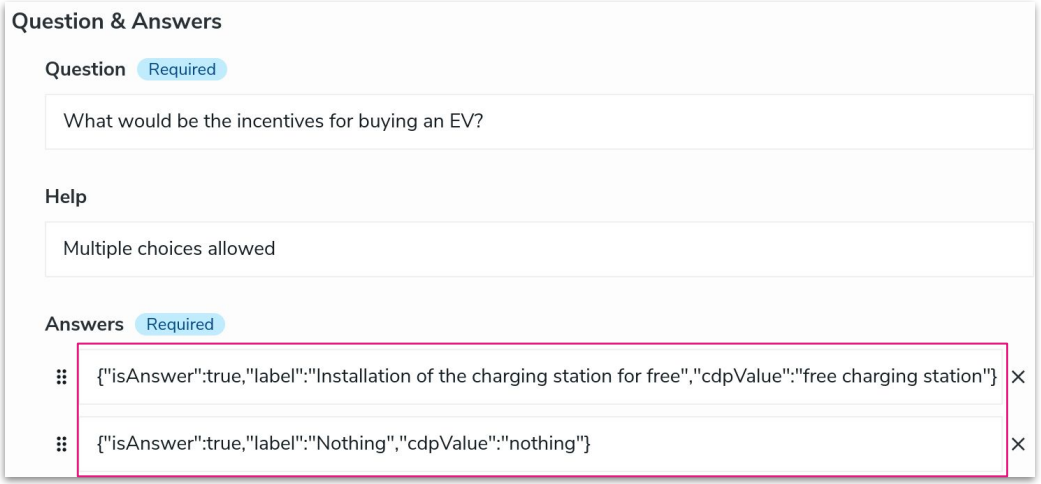
- isAnswer
- label
- cdpValue



# Context

## Content definition

```
[game4nt:qna] > jnt:content, ...
 - game4:question (string) mandatory internationalized
 - game4:help (string) internationalized
 - game4:answers (string) mandatory multiple internationalized
 - ...
```



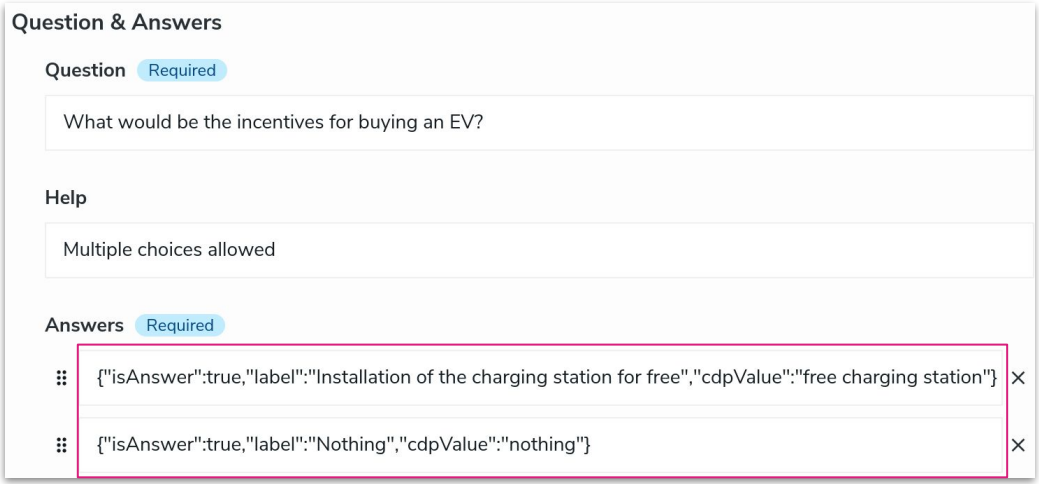

**Default** selector type for a multivalues string property

# Context

## Content definition

```
Answers = [‘{
  "isAnswer":true,  "label":"Nothing",
"cdpValue":"nothing"
}’]
```

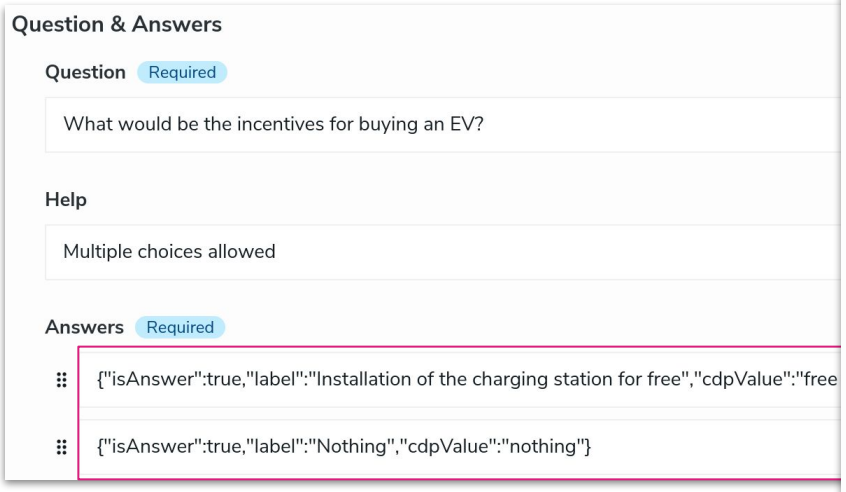
[**game4nt:qna**] > jnt:content, ...
- game4:question (string) mandatory internationalized
- game4:help (string) internationalized
- **game4:answers** (**string**) mandatory **multiple** internationalized
- **...**



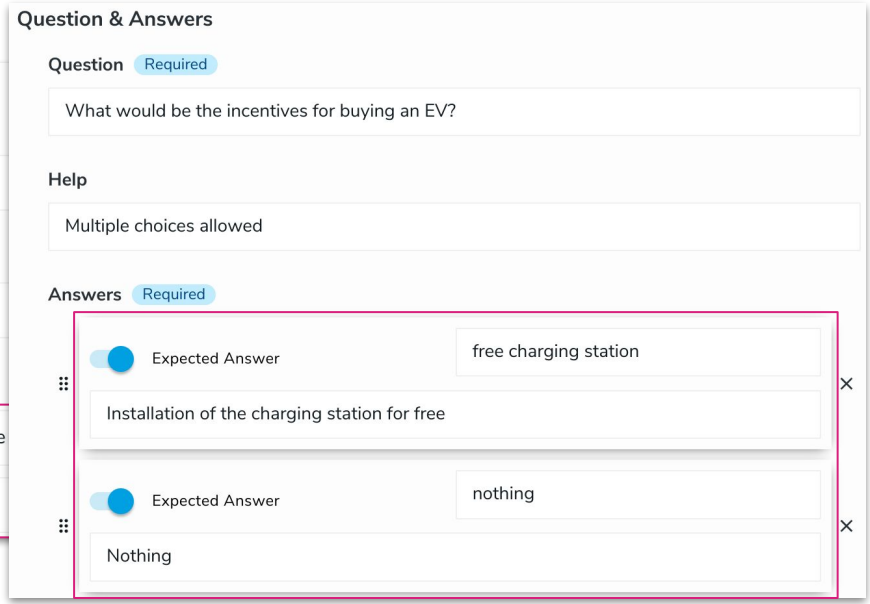

**Default** selector type for a multivalues string property

# What we want to achieve

Add a new selector type in the Content Editor form to properly display and contribute a property value stored as a stringify json object.

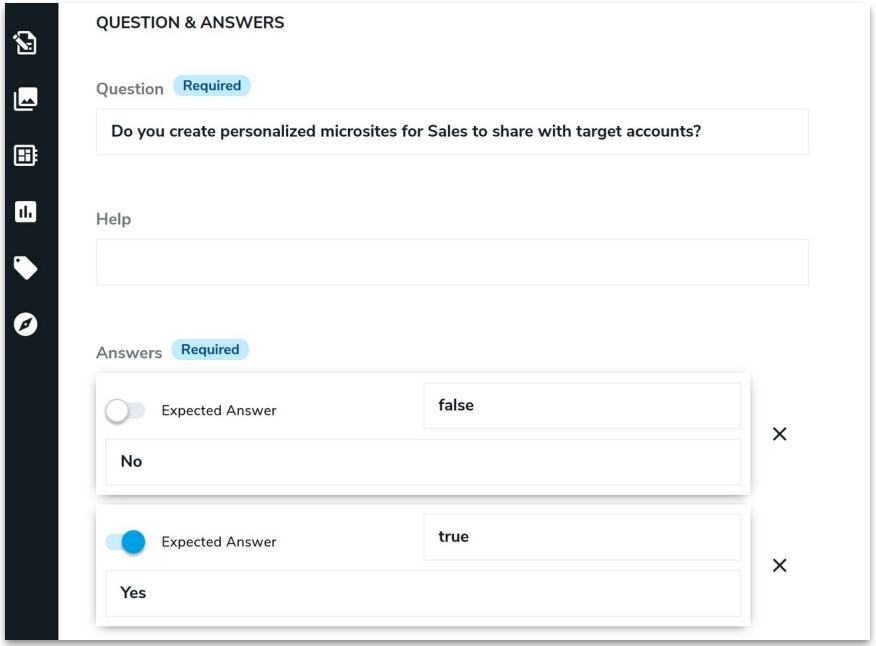



**Default** selector type for a multivalues string property




**Custom** selector type to manage the QnA stringify json object

# Adding new type of input in the CEditor form



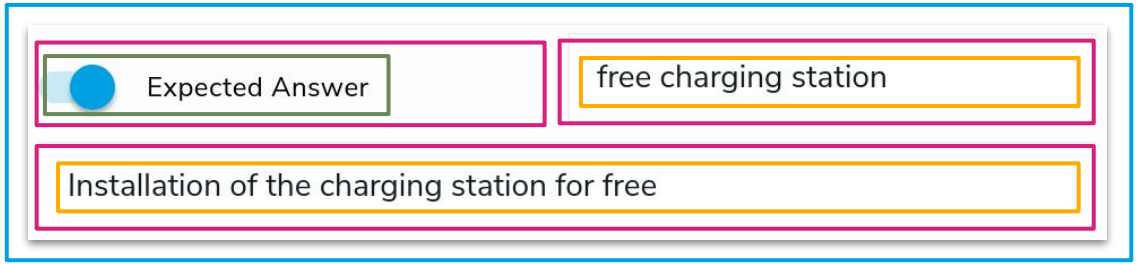

- The input field is render by a react app
- Content stored in the Answers property is a set of json String which can be used as it is by the quiz react app

[See on Github]

[See on the store]

# What we want to achieve

## UI Component HTML architecture




```
<Grid container>
  <Grid item sm={6}> <Toggle /> </Grid>
  <Grid item sm={6}> <Input /> </Grid>
  <Grid item> <Input /> </Grid>
</Grid>
```

```
import {Grid} from '@material-ui/core'
import {Input, Toggle} from '@jahia/design-system-kit'
```



jahia

# How to proceed

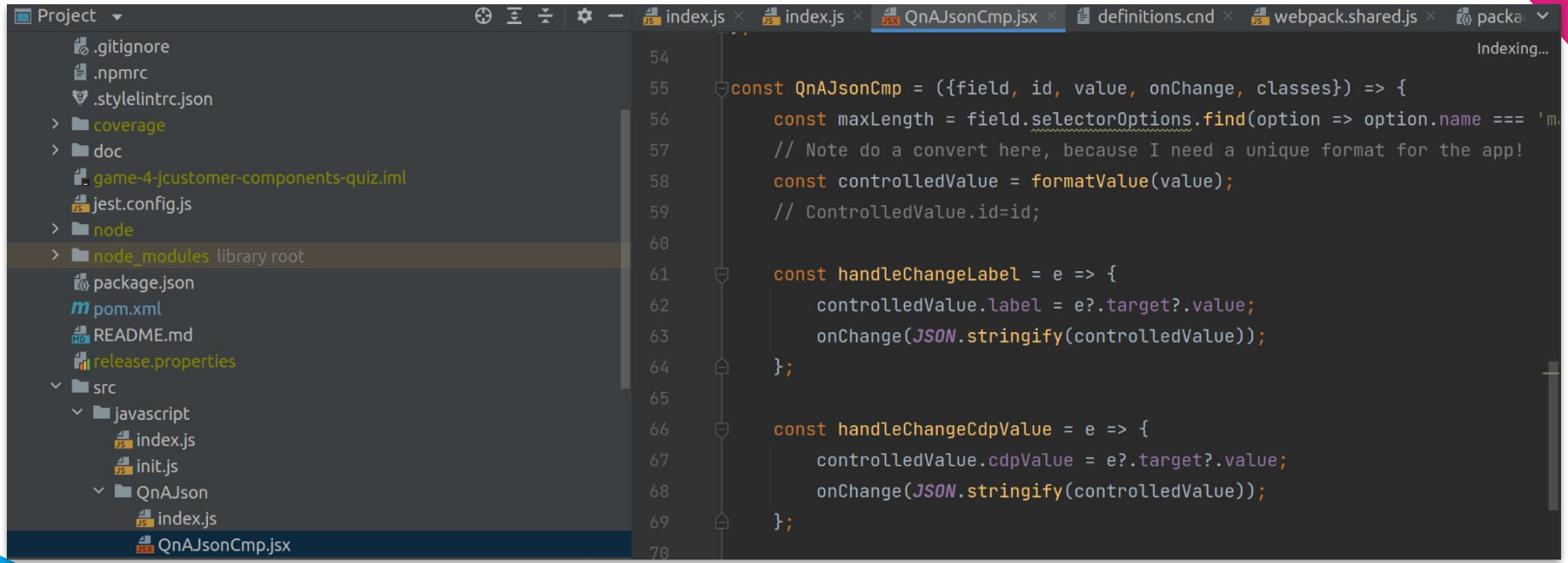
## Live coding steps to create and register a React component for Content Editor

- Create your react component in a Jahia module :
  **QnAJsonCmp.jsx**
- Create a file to register your component in the jahia UI :
  **init.js**
- Create and configure a file to build the component using the ModuleFederationPlugin :
  **webpack.config.js** *(already configured in this demo)*
- Informs Jahia about your new remote entry :
  **package.json**
- Configure the **pom.xml** to build and deploy your app *(already configured in this demo)*
- Create a json file to override the default input text used to render the string property :
  **game4nt_qna.json**



jahia

# Time to code



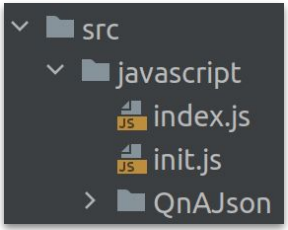



jahia

# Adding new type of input in the CEditor form

## How to proceed

- Create your react app in a Jahia module




- Create an init.js file to register your app in the jahia UI

```
registry.add('callback', 'QnAJsonEditor', {
    targets: ['jahiaApp-init:20'],
    callback: () => {
        registry.add('selectorType', 'QnAJson', {
            cmp: QnAJson, supportMultiple: false
        });
```

- Create and configure a webpack.config.js file to build the app using the ModuleFederationPlugin

```
plugins: [
    new ModuleFederationPlugin({
        name: "quizQnAJsonEditor",
        library: { type: "assign", name: "appShell.remotes.quizQnAJsonEditor" },
        filename: "remoteEntry.js",
        exposes: {
            './init': './src/javascript/init'
```

- In your package.json file informs Jahia about your new remote entry

```
"jahia": {
    "remotes": {
        "jahia": "javascript/apps/remoteEntry.js"
```

- Configure the pom.xml to build and deploy your app at run time

# Adding new type of input in the CEditor form

## How to proceed

- Create a content type in .cnd file with a string property

```
[game4nt:qna]
 - game4:answers (string) mandatory multiple internationalized
```

- Create a json file to override the default input text used to render a string

```
{
  "name": "game4nt:qna",
  "fields": [
    {
      "name": "game4:answers",
      "selectorType": "QnAJson"
```

[See more about json override]

/src/main/resources/META-INF/jahia-content-editor-forms/fieldsets/game4nt_qna.json

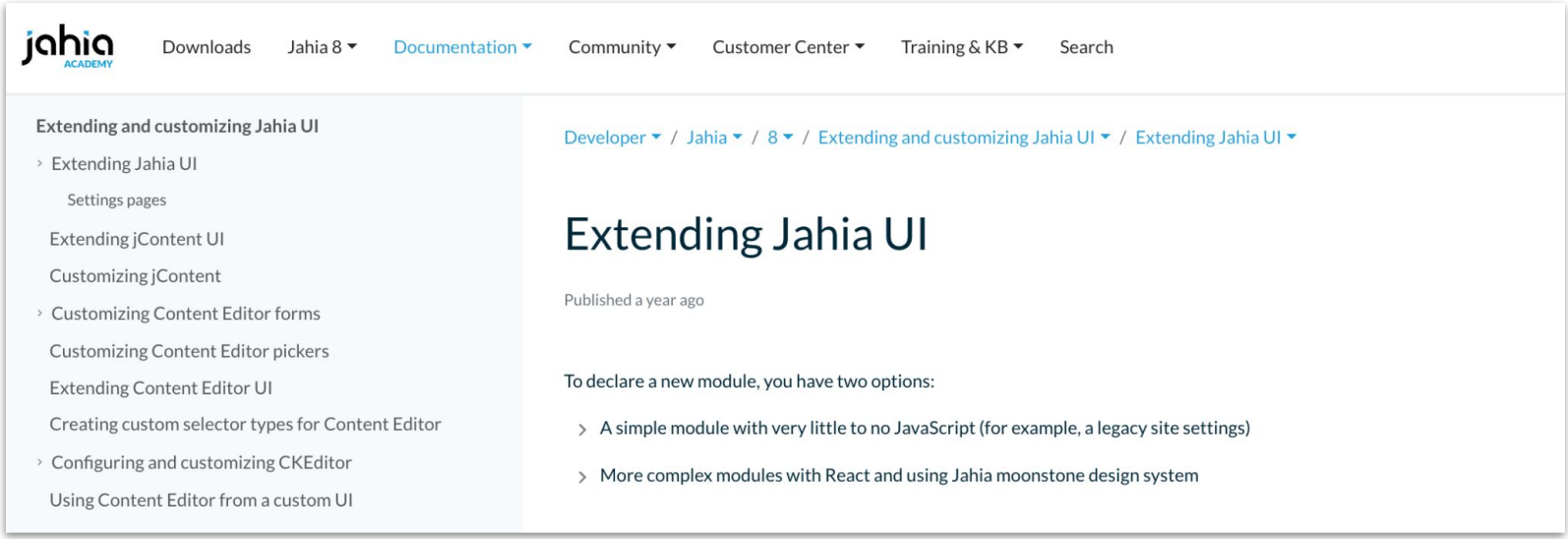


jahia

# Documentation

Extending Jahia UI on the Academy:

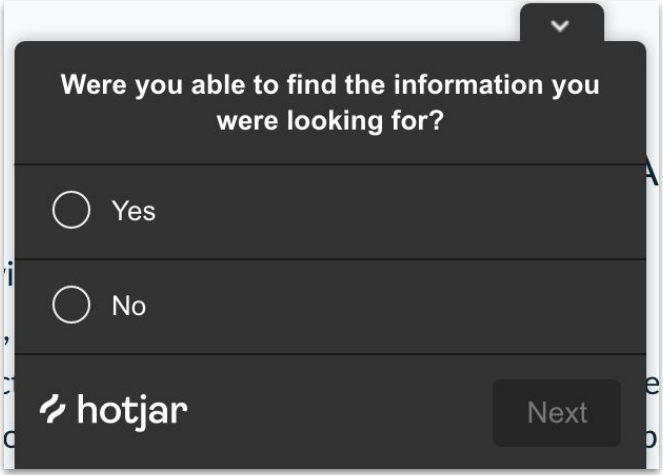
https://academy.jahia.com/documentation/developer/jahia/8/extending-and-customizing-jahia-ui/extending-jahia-ui

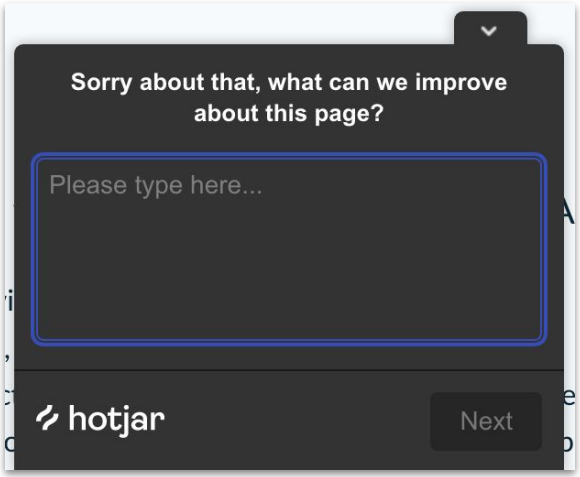



jahia

# Our documentation needs your feedback

- Product Managers and R&D team want to improve Jahia Academy!
- If you don’t find the information you’re looking for, share feedback via hotjar or open a ticket to our support team



jahia

# Next Webinar February 28th, 2023

# Content Editor 4: New Pickers and much more

- Discover the new version of Content Editor, the form to edit any content in our CMS.
  This webinar will cover every changes, from UX and functional to the deep technicalities such as:
  - Calling Content Editor from custom applications
  - Configuring custom pickers
  - json overrides

jahia

# Q & A

# Thank you!