



January 2023 Webinar

# Customizing Jahia UI using React Components

# Speakers

**Serge Huber**

CTO, Jahia Solutions



**Hervé Duchesne**

Senior Solution Architect



# Agenda

01 Intro

02 Overview & Examples

03 Concepts

04 Live Coding of a content editor selector type

05 Q & A

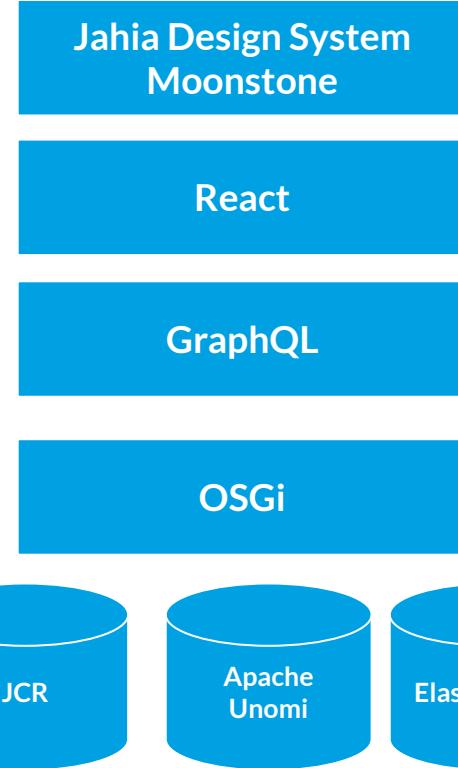


# Jahia 8 UI

- 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:



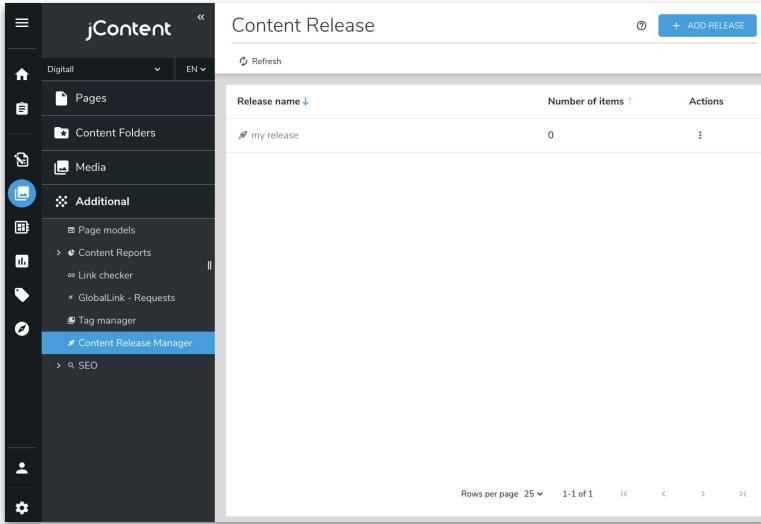
## Jahia UI Tech Stack



# Examples



# Adding a new panel in jContent

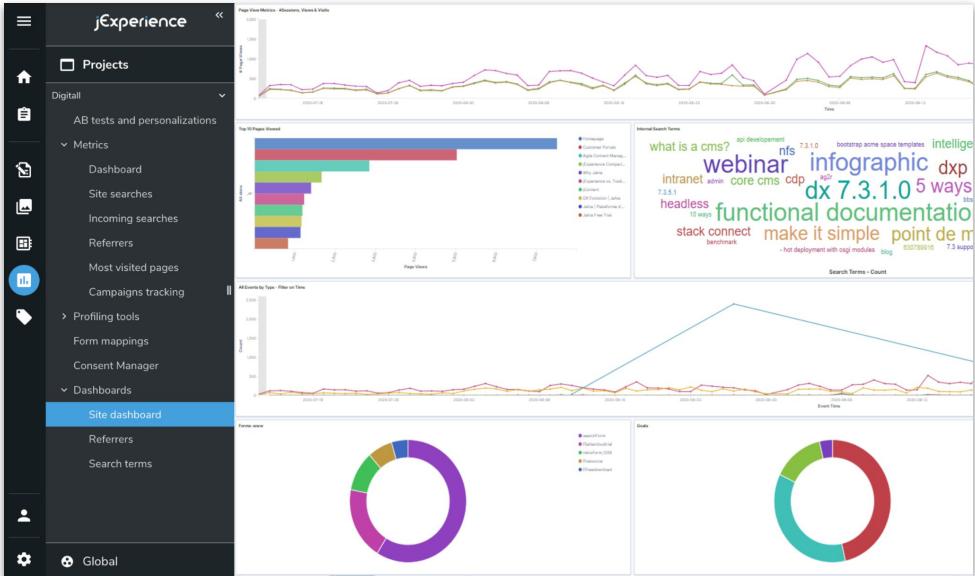


- Navigation entry is a React component
- The main display is a full React app using jahia-moonstone design system
- The app is built automatically when the module is installed
- Using yarn webpack react apollo
- Nice code sample for our datatable component usage

[See on Github](#)

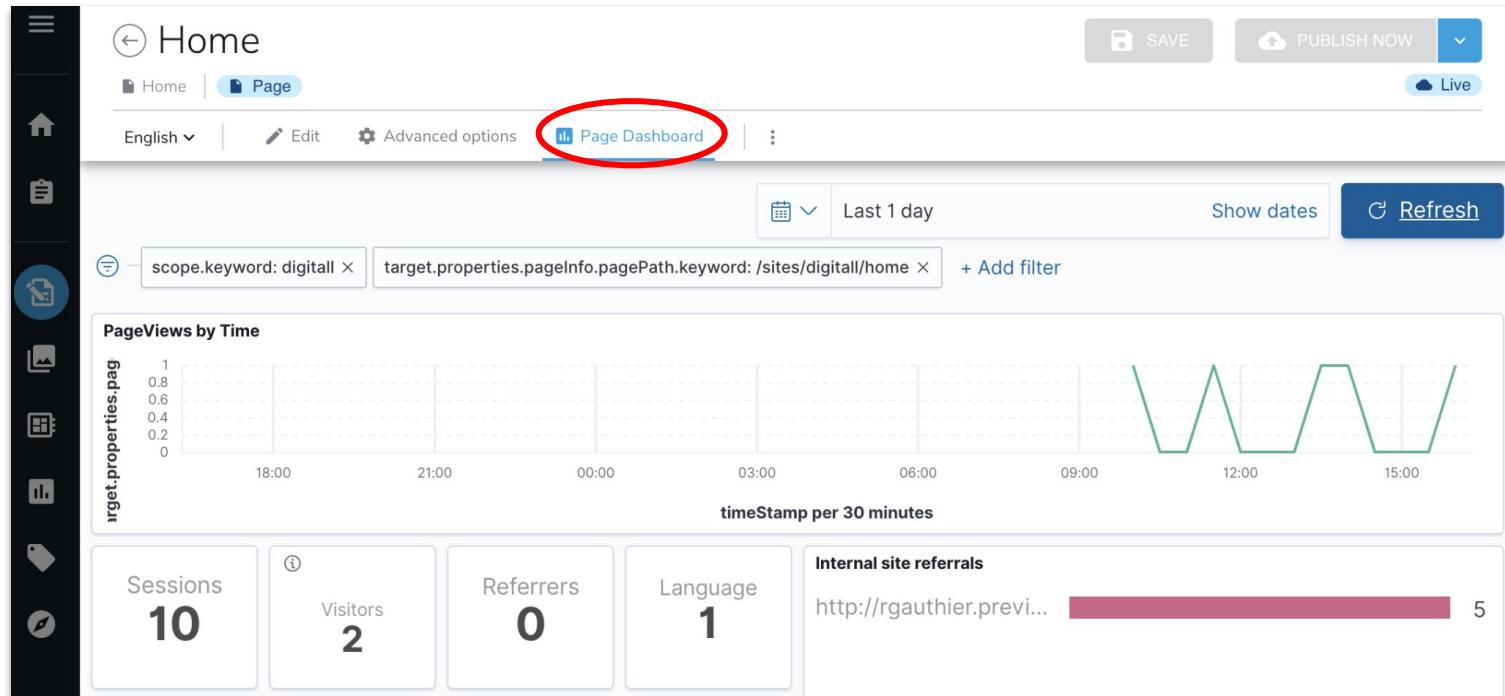
[See on the store](#)

# Adding a new panel in jExperience



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

# Adding a tab in Content Editor



The screenshot shows the Jahia Content Editor interface. On the left is a vertical toolbar with icons for navigation, creation, and management. The main area has a header with 'Home' and 'Page' tabs, and buttons for 'SAVE', 'PUBLISH NOW', and 'Live'. Below the header, there are language and edit options, followed by a red circle highlighting the 'Page Dashboard' tab. A search bar with filters ('scope.keyword: digitall', 'target.properties.pageInfo.pagePath.keyword: /sites/digitall/home') and a 'Refresh' button are also visible. The main content area displays a Kibana dashboard with a chart titled 'PageViews by Time' showing traffic over time, and summary statistics for 'Sessions' (10), 'Visitors' (2), 'Referrers' (0), 'Language' (1), and 'Internal site referrals' (a link to http://rgauthier.preview...).

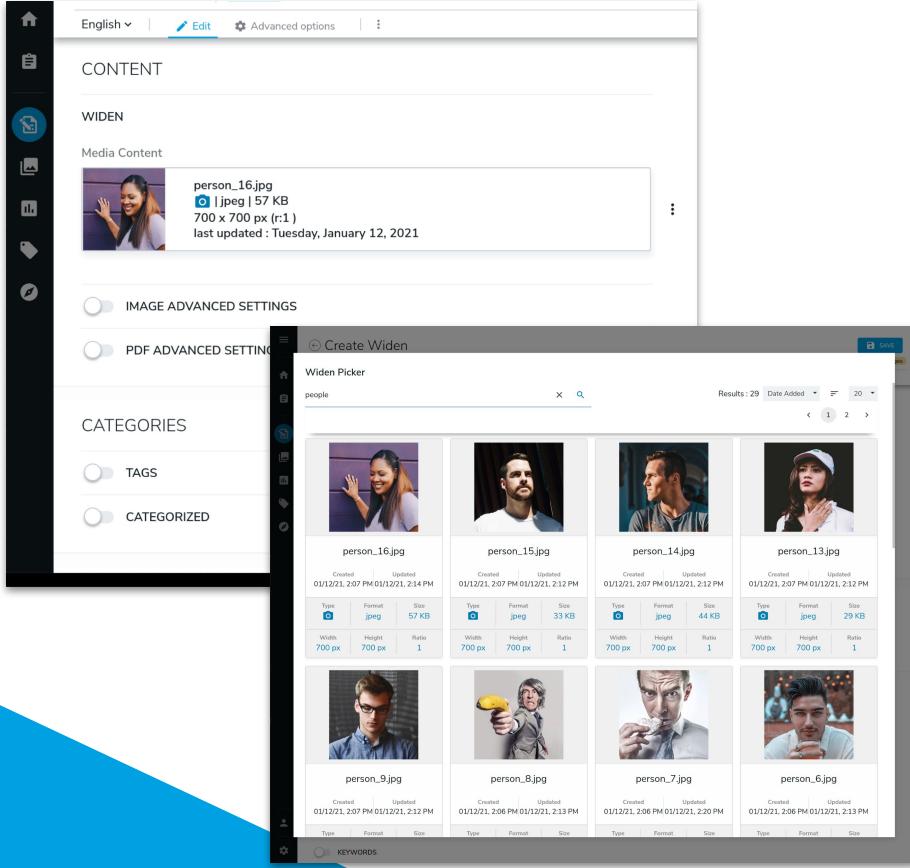
- The main display is an iframe calling an Elastic Kibana dashboard

# Adding an action in Content Editor

The screenshot shows the Jahia Content Editor interface for a news entry titled "Movies Can Determine Your Success". The "Edit" tab is selected. In the "CONTENT" section, there's a "NEWS ENTRY" form with fields for "News Title" (required) containing "Movies Can Determine Your Success", "System name" (required) containing "movies-can-determine-your-succes", and a "COPY TITLE" button. A modal dialog titled "Copy ‘News Title’ to other languages" is open, listing "5 languages selected" (Deutsch, italiano, français, español, Nederlands). A red box highlights the "Copy to other languages" button at the bottom right of the dialog.

See on the store

# Adding a new input type in Content Editor



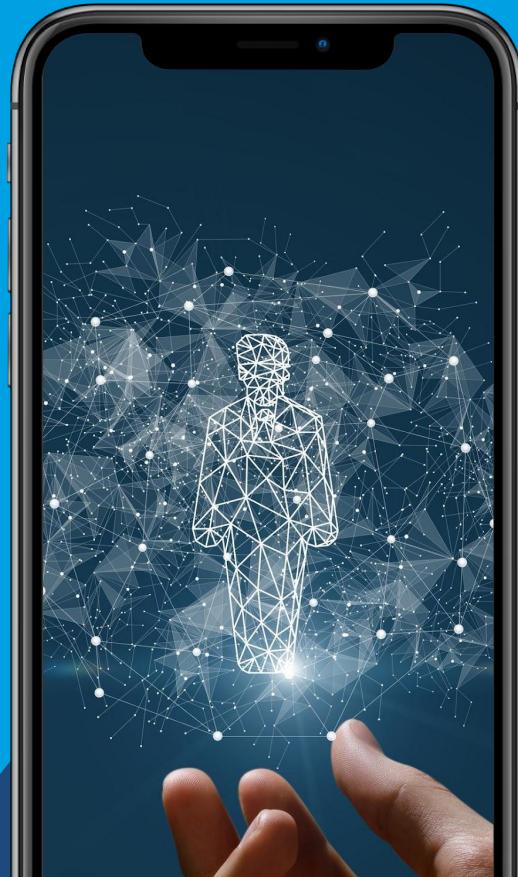
- The input field and the picker are rendered by a React app
- The picker browses the DAM using the DAM's rest api capabilities
- EDP is used to register in real time the selected image

[See on Github](#)

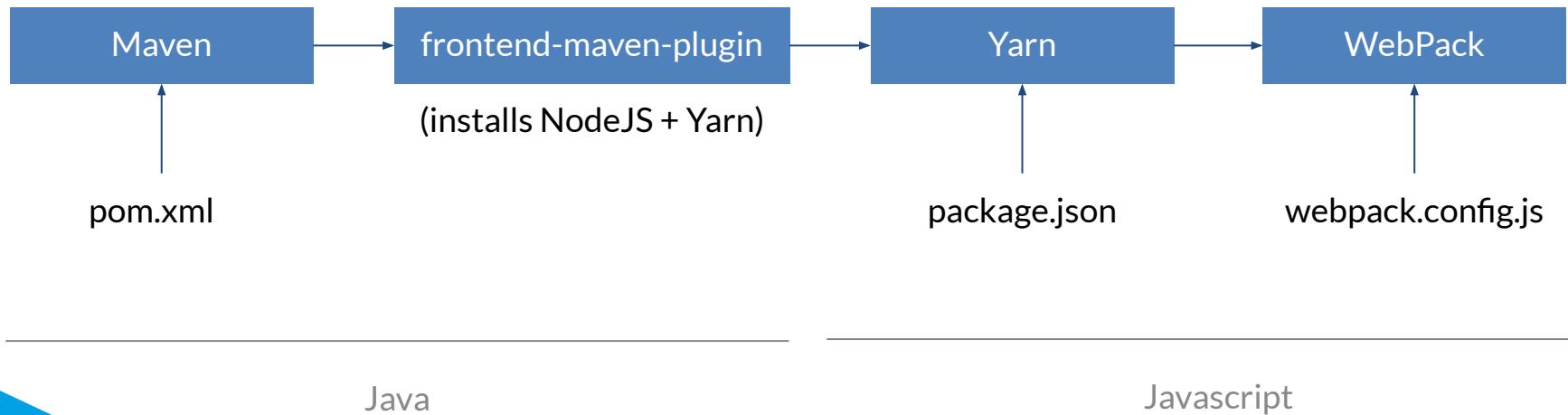
[See on store](#)

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

# Building and packaging



# Module building & packaging



# 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

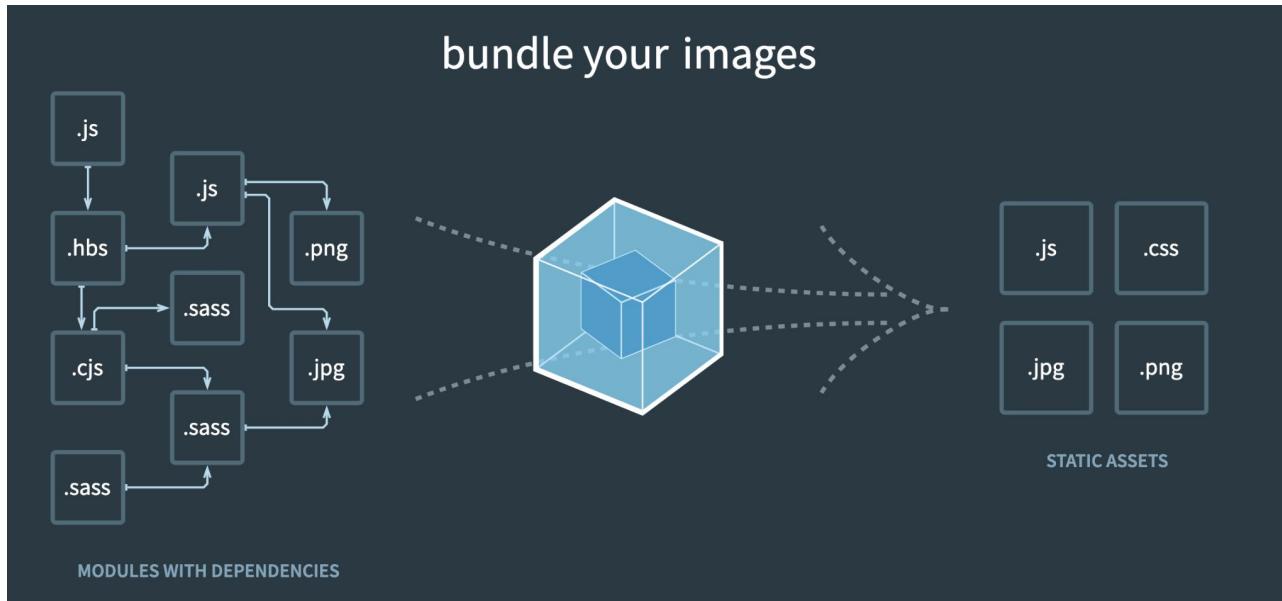
```
<plugin>
  <groupId>com.github.eirslett</groupId>
  <artifactId>frontend-maven-plugin</artifactId>
  <version>1.6</version>
  <executions>
    <execution>
      <id>npm install node and yarn</id>
      <phase>generate-resources</phase>
      <goals>
        <goal>install-node-and-yarn</goal>
      </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	yarn
Generating Lock Files	Automatically generated as <code>package-lock.json</code>	Automatically generated as <code>yarn.lock</code>
Using Workspaces	Supported	Supported
Remote Scripts	Supported, using the <code>npx</code> command	Supported, using the <code>yarn d1x</code> command
Plug'n'Play	Not supported	Uses a single <code>.pnp.cjs</code> file instead of the <code>node_modules</code> folder
Zero Installs	Not supported	Uses the <code>.pnp.cjs</code> file to quickly reinstall packages from the offline cache
License Check	Not supported	Checks each package license while downloading

# WebPack



# 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
      })
    ]
  }
}

export default config;
```

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

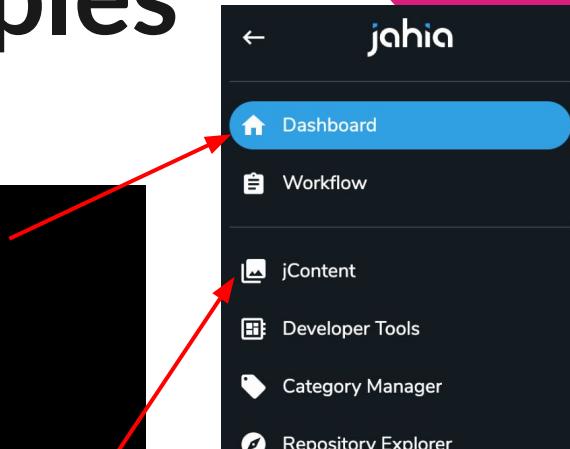
- Common types are:
  - action: use to add an UI action in content editor
  - selectorType : add a selector type in content editor
  - adminRoute : add a route in the administration
  - route : add a route in for a custom panel
  - primary-nav-item : add an entry in the primary level navigation
  - callback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

# Component registry examples

- Dashboard navigation entry

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

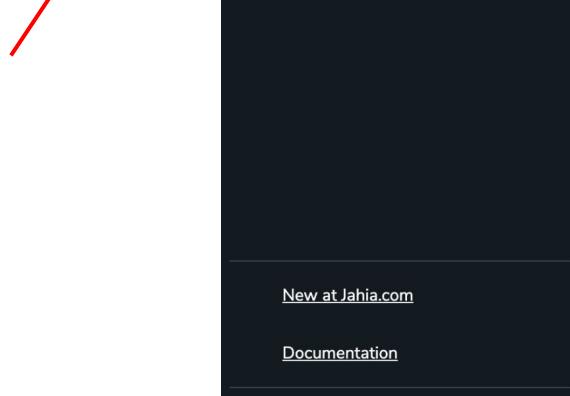
registry.add('primary-nav-item', 'dashboardGroupItem', {
    targets: ['nav-root-tasks:1'],
    render: () => <DashboardGroup/>
}) ;
```



- 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'});
```

# Navigation iFrame URLs

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

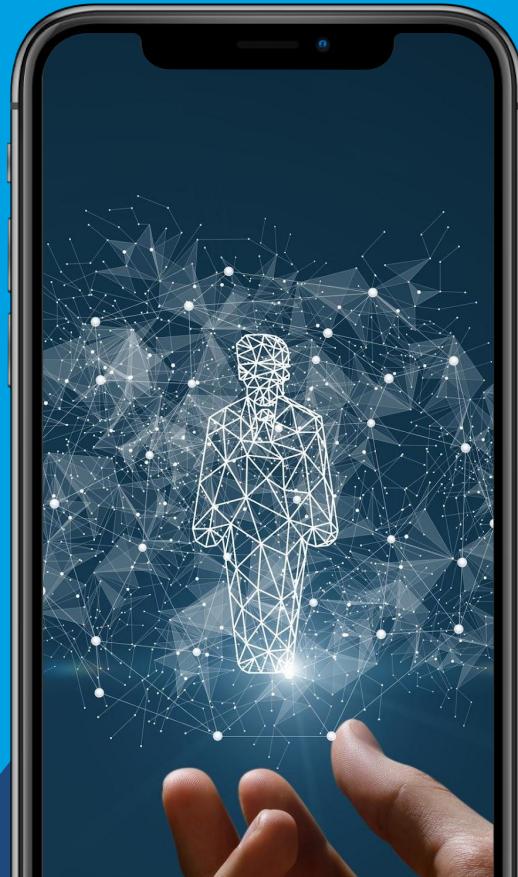
```
registry.addOrReplace('jExperienceMenuEntry', 'siteDashboard', {
    targets: ['siteMetrics:1'],
    id: 'siteDashboard',
    labelKey: 'jexperience:jexperience.siteMetrics.siteDashboard.title',
    isSelectable: true,
    iFrameUrl: window.contextJsParameters.contextPath +
    '/cms/editframe/default/${lang/sites/${site-key}.marketing-10-dashboard.html}',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

# Component registry debugging

```
> jahia.uiExtender.registry.find({type: "selectorType"})
< ▷ (17) [{...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}]

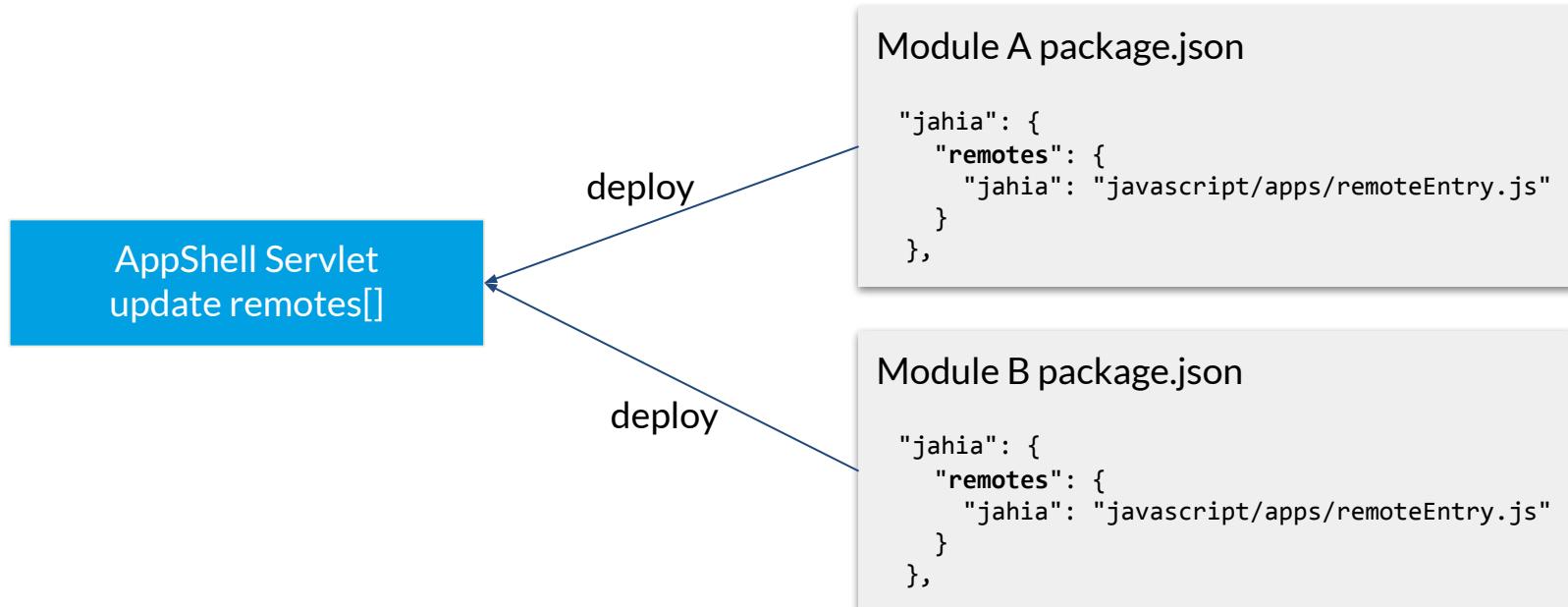
▶ 0: {multiple: false, type: 'selectorType', key: 'KibanaTargetField', cmp: f}
▶ 1: {supportMultiple: true, type: 'selectorType', key: 'AutomatedTags', cmp: f}
▶ 2: {dataType: Array(1), labelKey: 'content-editor:label.contentEditor.selectorTypes.category.displayValue', properties: Array(2), supportMultiple: true, cmp: f, ...}
▶ 3: {dataType: Array(1), labelKey: 'content-editor:label.contentEditor.selectorTypes.tag.displayValue', properties: Array(2), supportMultiple: true, cmp: f, ...}
▶ 4: {dataType: Array(3), labelKey: 'content-editor:label.contentEditor.selectorTypes.text.displayValue', properties: Array(2), supportMultiple: false, cmp: f, ...}
▶ 5: {dataType: Array(1), labelKey: 'content-editor:label.contentEditor.selectorTypes.textArea.displayValue', properties: Array(2), supportMultiple: false, cmp: f, ...}
▶ 6: {dataType: Array(1), labelKey: 'content-editor:label.contentEditor.selectorTypes.richText.displayValue', properties: Array(2), supportMultiple: false, cmp: f, ...}
▶ 7: {dataType: Array(1), labelKey: 'content-editor:label.contentEditor.selectorTypes.color.displayValue', properties: Array(2), supportMultiple: false, cmp: f, ...}
▶ 8: {dataType: Array(1), labelKey: 'content-editor:label.contentEditor.selectorTypes.dateTimePicker.displayValue', properties: Array(2), supportMultiple: false, cmp: f, ...}
▶ 9: {dataType: Array(1), labelKey: 'content-editor:label.contentEditor.selectorTypes.datePicker.displayValue', properties: Array(2), supportMultiple: false, cmp: f, ...}
▶ 10: {dataType: Array(1), containerStyle: 'uvmWULEguySIGQHSuBDaUg==', labelKey: 'content-editor:label.contentEditor.selectorTypes.checkbox.displayValue', properties: Array(2), cmp: f, ...}
▶ 11: {dataType: Array(1), labelKey: 'content-editor:label.contentEditor.selectorTypes.picker.displayValue', properties: Array(2), type: 'selectorType', resolver: f, ...}
▶ 12: {dataType: Array(1), supportMultiple: true, labelKey: 'content-editor:label.contentEditor.selectorTypes.dropdown.displayValue', properties: Array(2), cmp: f, ...}
▶ 13: {dataType: Array(1), supportMultiple: false, type: 'selectorType', key: 'SystemName', cmp: f}
▶ 14: {dataType: Array(1), labelKey: 'content-editor:label.contentEditor.selectorTypes.multipleLeftRightSelector.displayValue', properties: Array(2), supportMultiple: true, cmp: f, ...}
▶ 15: {supportMultiple: false, type: 'selectorType', key: 'contentTypeProperties', cmp: f}
▶ 16: {supportMultiple: false, type: 'selectorType', key: 'CodeMirror', cmp: f}
length: 17
▶ [[Prototype]]: Array(0)
```

# Loading & routing



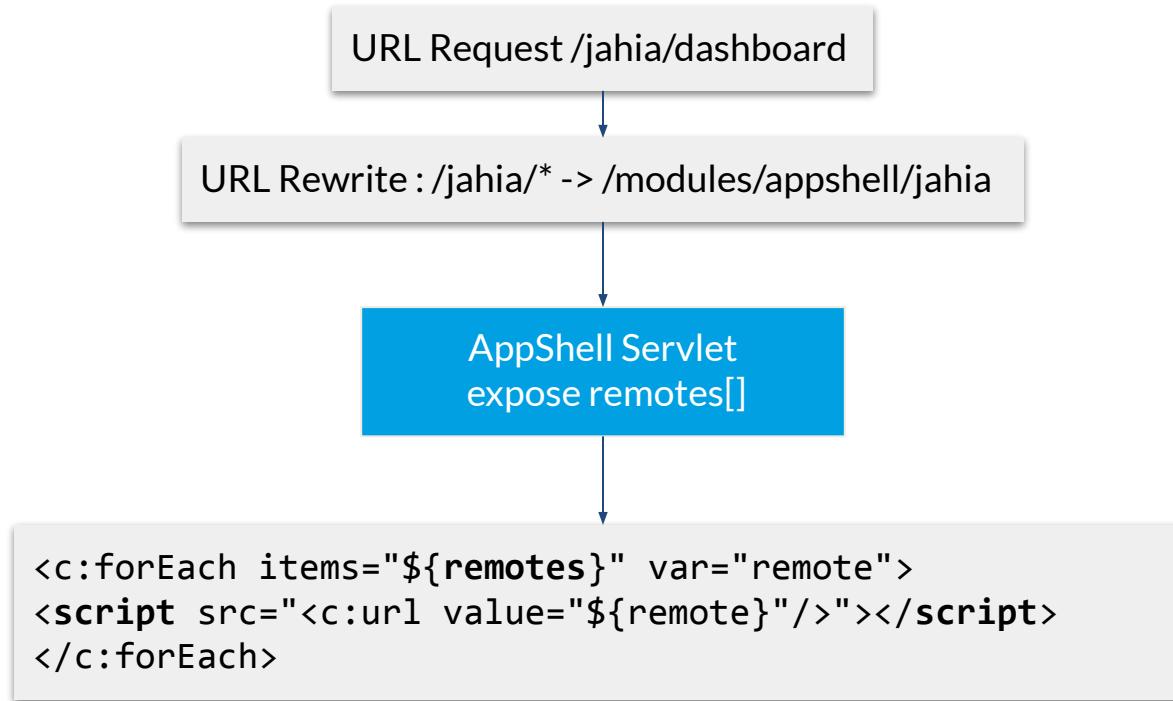
# Javascript UI Registration

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



# 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>
                }
            ) : null)}
          </Switch>
        </ErrorBoundary>
      </ContentNavigation>
    </LayoutModule>
  );
}
```

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

jahia

# Leveraging Jahia features



# Leverage UI features

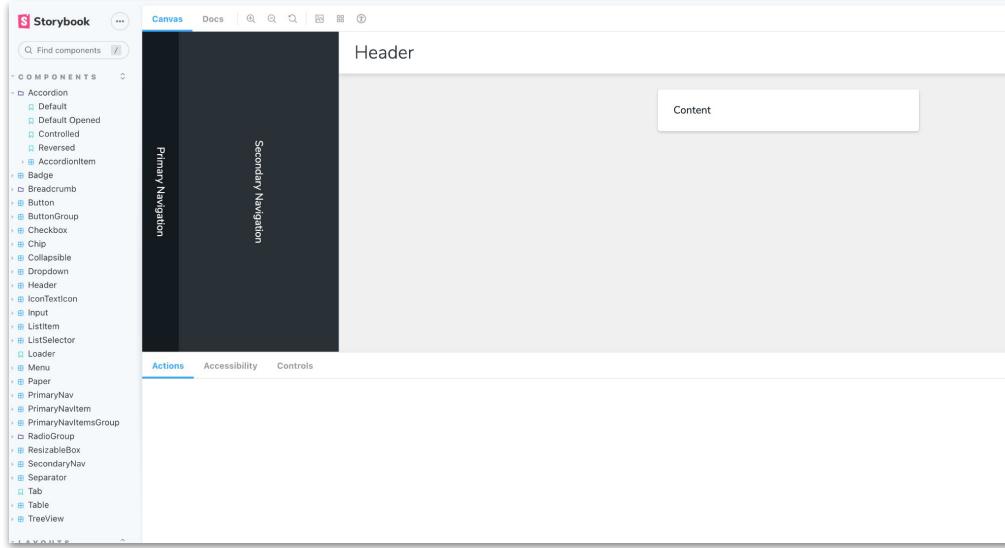
Custom components can leverage

- 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/>
}) ;
```

# 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

# Internationalization

## Component JSX

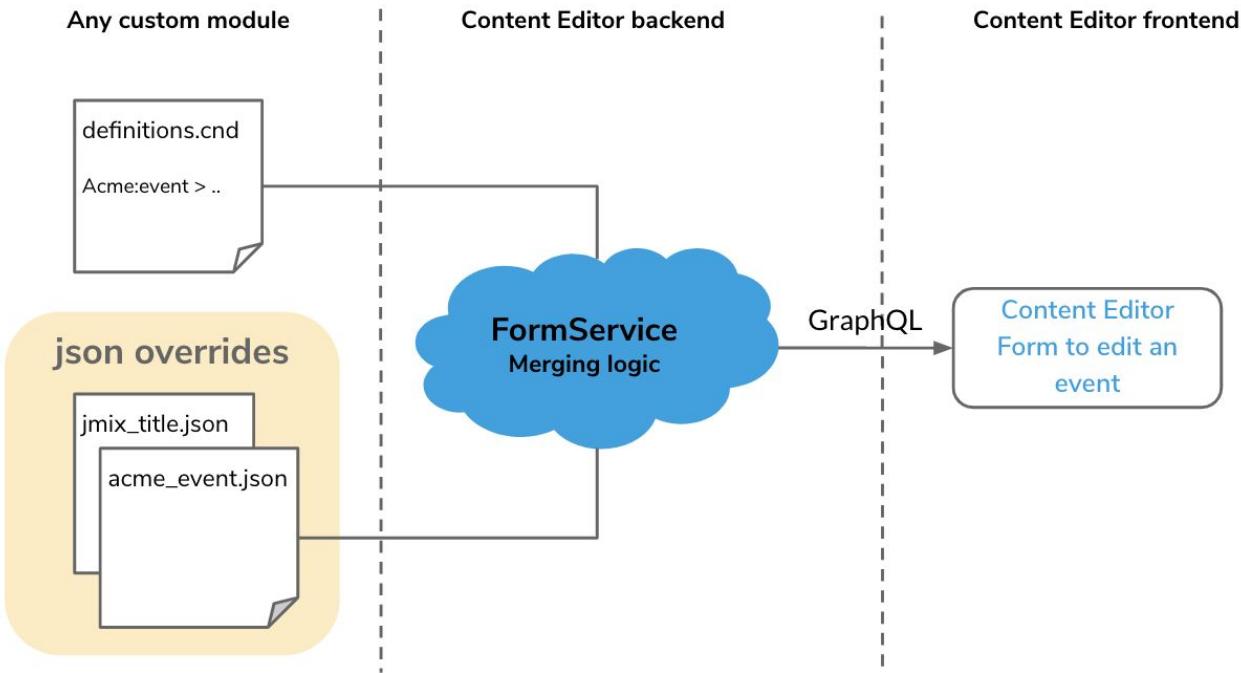
```
import React from 'react';
import {useTranslation} from 'react-i18next';
import {PrimaryNavItem} from '@jahia/moonstone';
import Constants from './Dashboard.constants';

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",
        "icon": "fa fa-tachometer"
    }
}
```

# 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  
      }  
    }  
  ]  
}
```

# 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['@jahia/moonstone']).find(f => f[1].loaded)`

```
▼ webpackShareScopes: Object
  ▼ default:
    ► @apollo/client: {3.5.10: {...}}
    ► @apollo/react-common: {3.1.4: {...}}
    ► @apollo/react-components: {3.1.5: {...}}
    ► @apollo/react-hooks: {3.1.5: {...}}
    ► @jahia/data-helper: {1.0.4: {...}, 1.0.3: {...}, 1.0.6: {...}, 1.0.7: ...}
    ► @jahia/design-system-kit: {1.1.11: {...}}
    ► @jahia/icons: {1.1.1: {...}, 1.1.2: {...}}
    ► @jahia/moonstone: {1.5.3: {...}, 1.6.2: {...}, 2.4.20: {...}, 2.3.1: ...}
    ► @jahia/moonstone-alpha: {1.0.1: {...}}
    ► @jahia/react-material: {3.0.3: {...}, 3.0.5: {...}}
```

```
▼ @jahia/moonstone:
  ► 1.2.0: {from: '@jahia/sitemap', eager: false, get: f}
  ► 1.5.3: {from: '@jahia/security-filter', eager: false, get: f}
  ► 1.6.2: {from: '@jahia/site-settings', eager: false, get: f}
  ► 2.3.1: {from: 'sdl-generator-tools', eager: false, get: f}
  ► 2.4.14: {from: '@jahia/site-settings-seo', eager: false, get: f}
  ► 2.4.20: {from: '@jahia/jcontent', eager: false, loaded: 1, get: f}
  ► [[Prototypal: Object]]
```

# 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

# Live coding



# Context

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

## QnA Content Properties

Question What would be the incentives for buying an EV?

Help Multiple choices allowed

Answers  Installation of the charging station for free

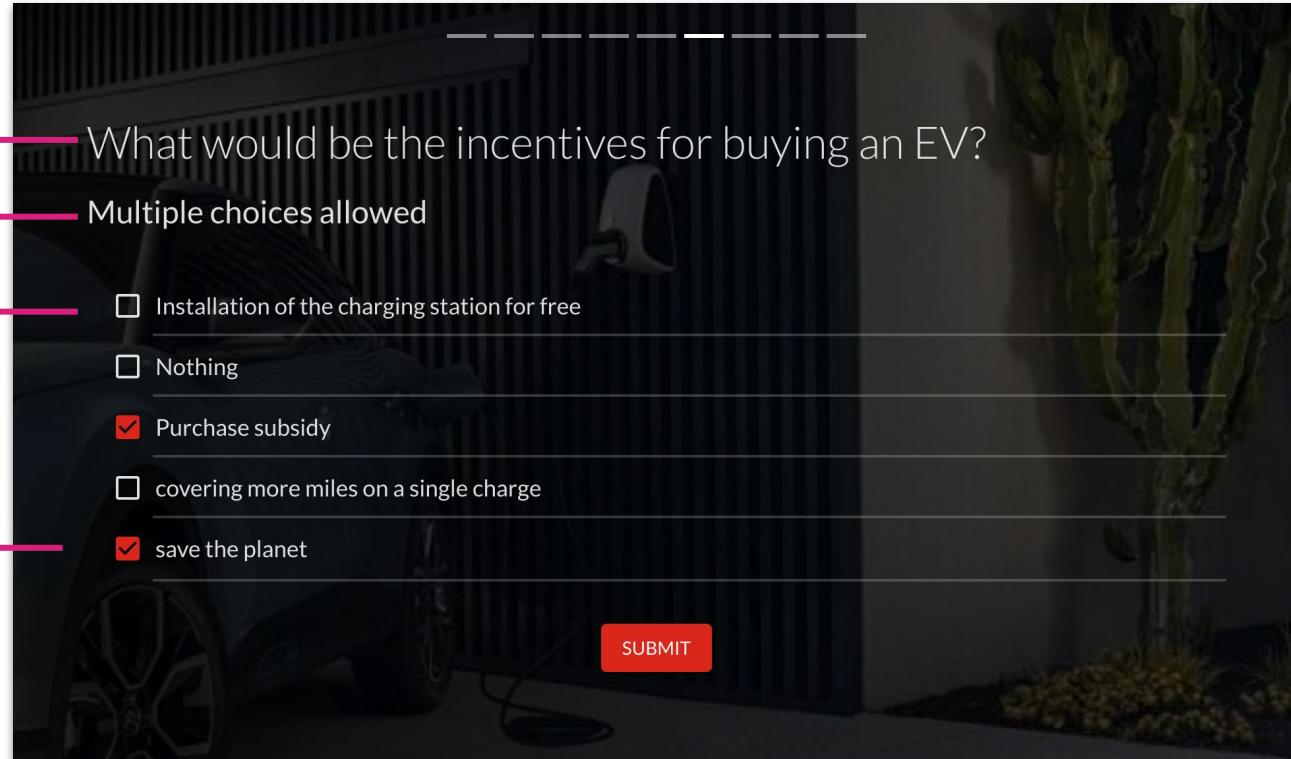
Nothing

Purchase subsidy

covering more miles on a single charge

save the planet

SUBMIT



# Context

## Content definition

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

**Question & Answers**

**Question** Required

What would be the incentives for buying an EV?

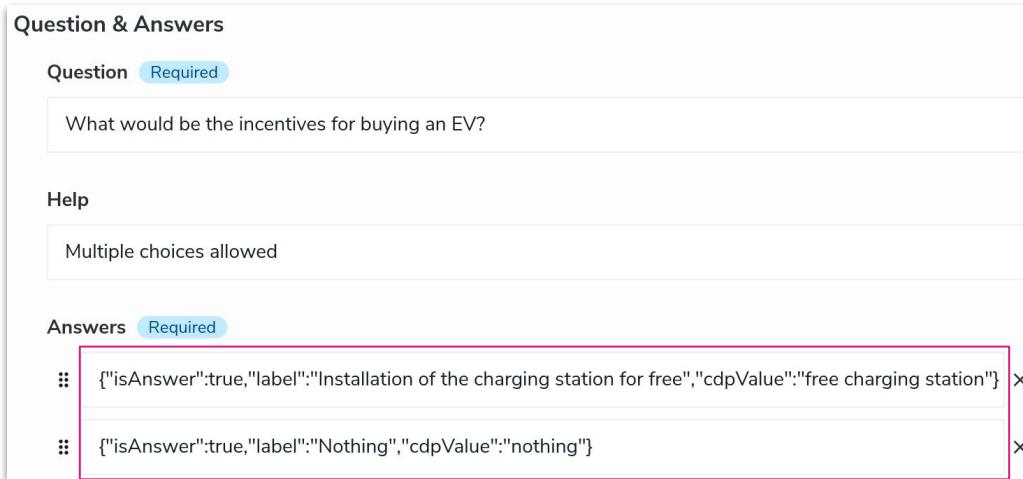
**Help**

Multiple choices allowed

**Answers** Required

⌘ {"isAnswer":true,"label":"Installation of the charging station for free","cdpValue":"free charging station"} x

⌘ {"isAnswer":true,"label":"Nothing","cdpValue":"nothing"} x



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

# Context

## Content definition

[game4nt:qna] > jnt:content, ...

- game4:question (string) mandatory internationalized
- game4:help (string) internationalized
- **game4:answers (string) mandatory multiple internationalized**
- ...

**Question & Answers**

Question Required

What would be the incentives for buying an EV?

Help

Multiple choices allowed

Answers Required

⌘ {"isAnswer":true,"label":"Installation of the charging station for free","cdpValue":"free charging station"} ×

⌘ {"isAnswer":true,"label":"Nothing","cdpValue":"nothing"} ×

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

**Default selector type for a multivalue 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.

**Question & Answers**

**Question** Required

What would be the incentives for buying an EV?

**Help**

Multiple choices allowed

**Answers** Required

```
[:] {"isAnswer":true,"label":"Installation of the charging station for free","cdpValue":"free"}  
[:] {"isAnswer":true,"label":"Nothing","cdpValue":"nothing"}
```

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

**Question & Answers**

**Question** Required

What would be the incentives for buying an EV?

**Help**

Multiple choices allowed

**Answers** Required

<input checked="" type="checkbox"/> Expected Answer	free charging station	X
Installation of the charging station for free		X
<input checked="" type="checkbox"/> Expected Answer	nothing	X
Nothing		X

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

# Adding new type of input in the CEditor form

QUESTION & ANSWERS

Question Required

Do you create personalized microsites for Sales to share with target accounts?

Help

Answers Required

<input type="radio"/> Expected Answer	false	x
No		
<input checked="" type="radio"/> Expected Answer	true	x
Yes		

- 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'
```

# 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`

# Time to code

```
const QnAJsonCmp = ({field, id, value, onChange, classes}) => {
  const maxLength = field.selectorOptions.find(option => option.name === 'm');
  // Note do a convert here, because I need a unique format for the app!
  const controlledValue = formatValue(value);
  // ControlledValue.id=id;

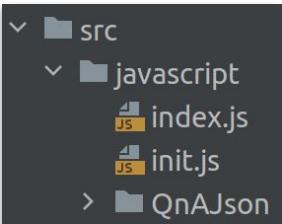
  const handleChangeLabel = e => {
    controlledValue.label = e?.target?.value;
    onChange(JSON.stringify(controlledValue));
  };

  const handleChangeCdpValue = e => {
    controlledValue.cdpValue = e?.target?.value;
    onChange(JSON.stringify(controlledValue));
  };
};
```

# 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

# Documentation

Extending Jahia UI on the Academy:

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

The screenshot shows a navigation bar with links for Downloads, Jahia 8, Documentation (which is active), Community, Customer Center, Training & KB, and Search. The main content area has a sidebar on the left containing a tree view of documentation topics under 'Extending and customizing Jahia UI'. The main content area displays the title 'Extending Jahia UI' with a publish date of 'Published a year ago'. Below the title is a section titled 'To declare a new module, you have two options:' with two bullet points.

jahia  
ACADEMY

Downloads    Jahia 8 ▾    Documentation ▾    Community ▾    Customer Center ▾    Training & KB ▾    Search

Extending and customizing Jahia UI

- › Extending Jahia UI
  - Settings pages
- Extending jContent UI
- Customizing jContent
- › Customizing Content Editor forms
- Customizing Content Editor pickers
- Extending Content Editor UI
- Creating custom selector types for Content Editor
- › Configuring and customizing CKEditor
- Using Content Editor from a custom UI

Developer ▾ / Jahia ▾ / 8 ▾ / Extending and customizing Jahia UI ▾ / Extending Jahia UI ▾

## Extending Jahia UI

Published a year ago

To declare a new module, you have two options:

- › A simple module with very little to no JavaScript (for example, a legacy site settings)
- › More complex modules with React and using Jahia moonstone design system

# 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

Were you able to find the information you were looking for?

Yes

No

 hotjar

Next

Sorry about that, what can we improve about this page?

Please type here...

 hotjar

Next

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

# Q & A



# Thank you!

