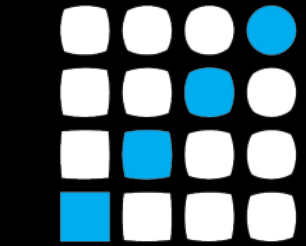


EUROPE'S LEADING AEM DEVELOPER CONFERENCE

28th – 30th SEPTEMBER 2020

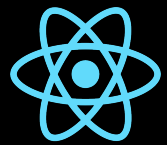
Components as a Service

Tony Schumacher, Teclead

adaptTo()

Agenda

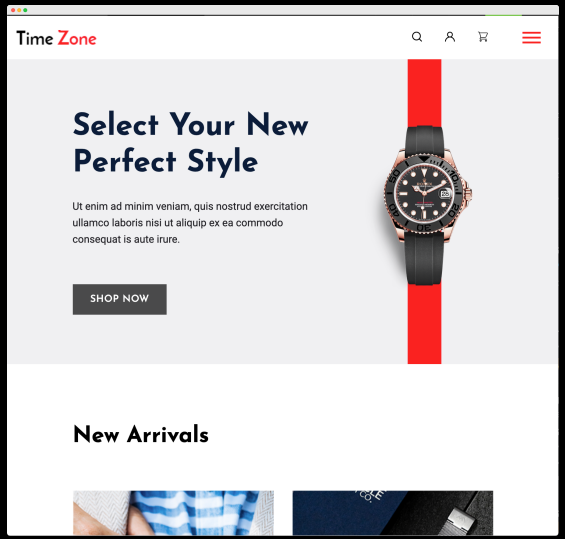
- Project Situation
- Solution Concept
- Frontend Architecture
- Authoring
- Demo / Live coding
- FAQ



Project Situation

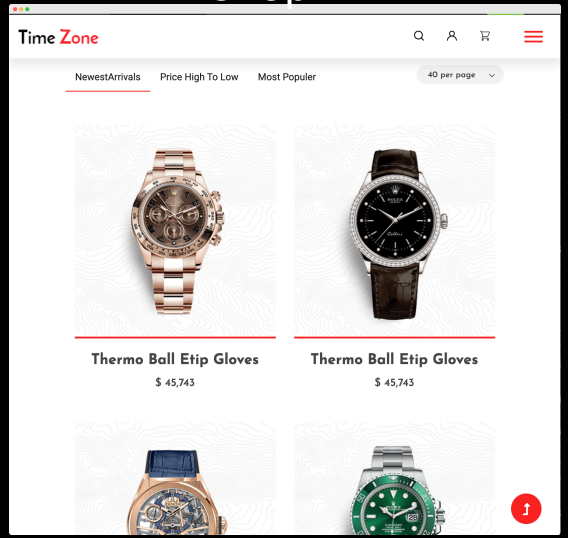
Project Situation

Home



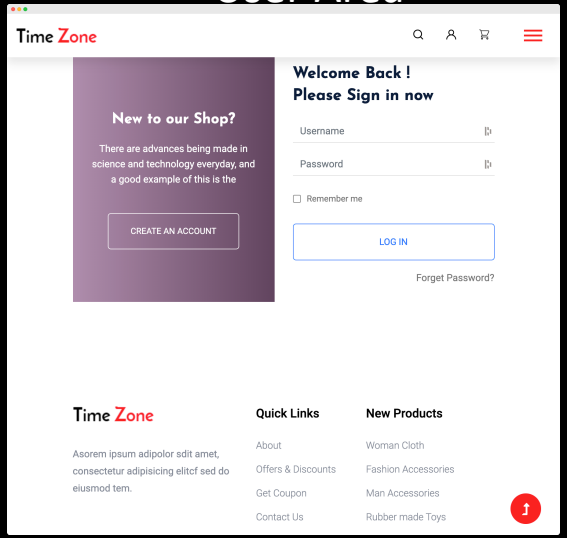
brand.com

Shop



brand.com/shop

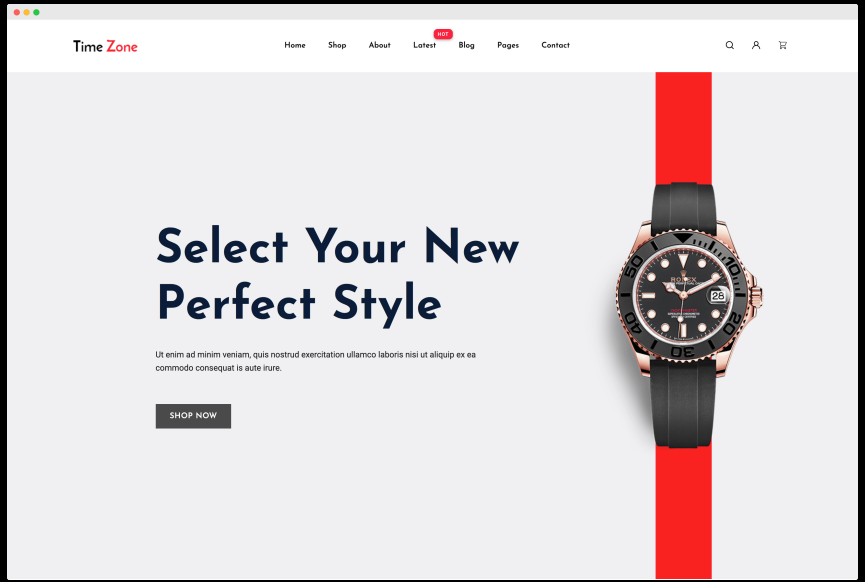
User Area



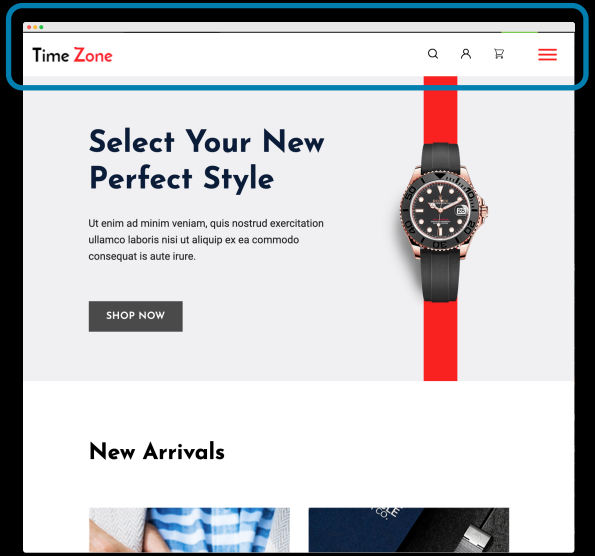
brand.com/user-area

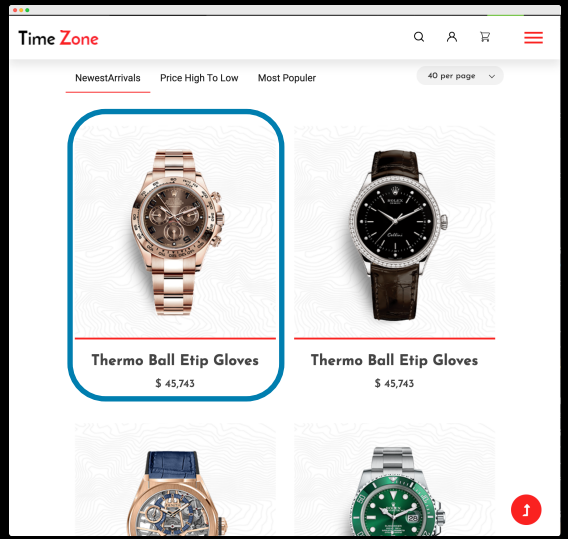
Project Situation

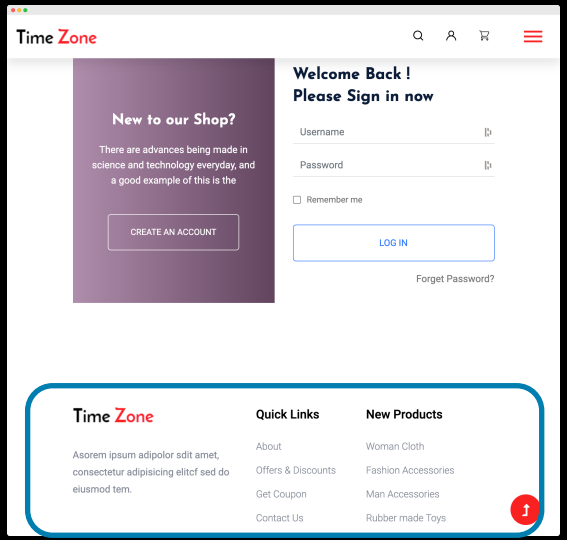
- company uses multiple systems in one plattform
- redundant development work
- slow and synchronized release processes
- no option for system migration
- independent teams



Project Situation







- Redundant implementations for multiple components
- Synced deployment processes

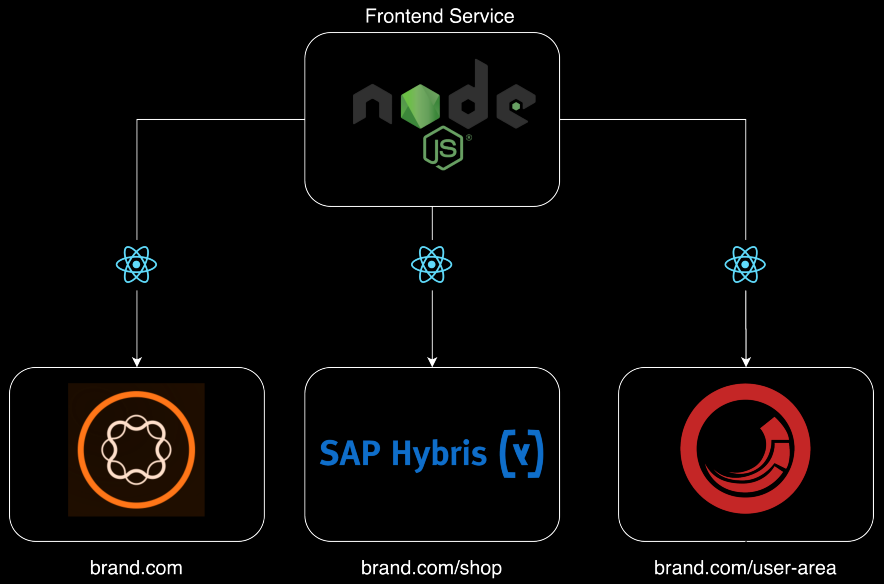
adaptTo()

Solution Concept

Solution Concept

Specs:

- decouple frontend development (Frontend Service)
- One common frontend stack through the systems
- Frontend components ship dialogs
- „Hybrid headless“ set up

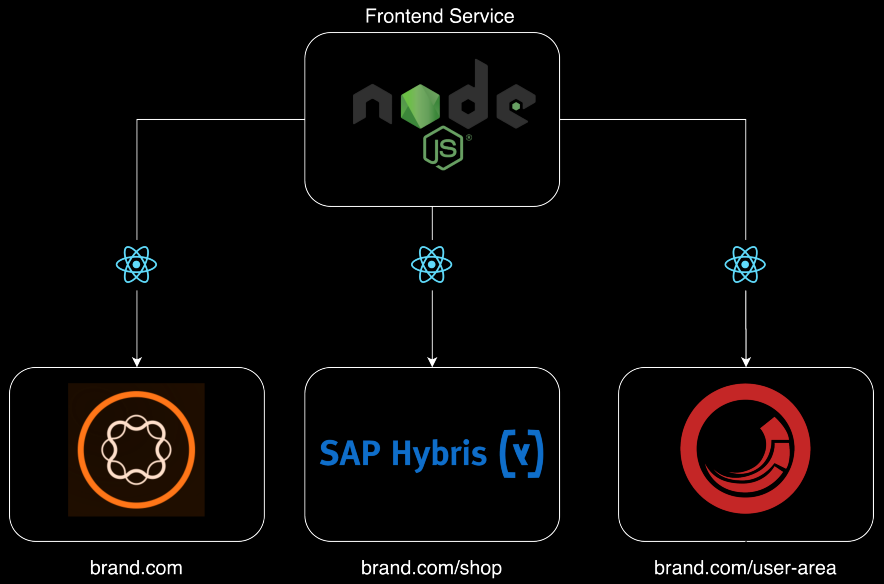


Solution Concept

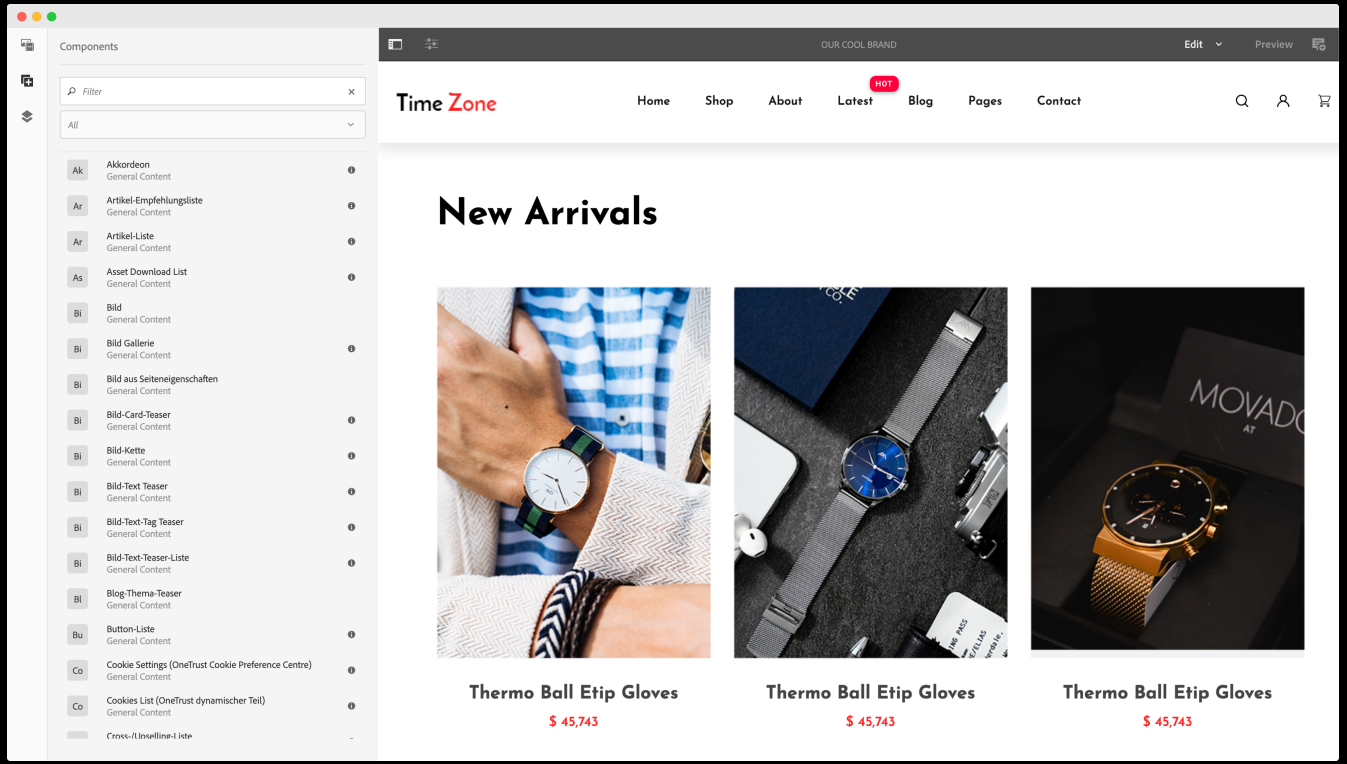
What it solves:

- ✓ One release process for all systems
- ✓ deployment is synced
- ✓ no redundancies
- ✓ Technology agnostic

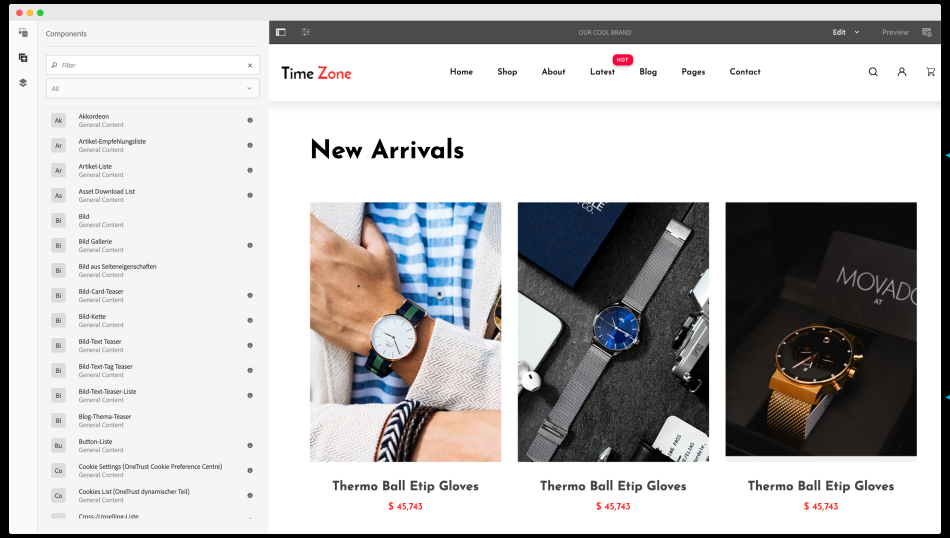
- ✓ no AEM deployment needed for new frontend components



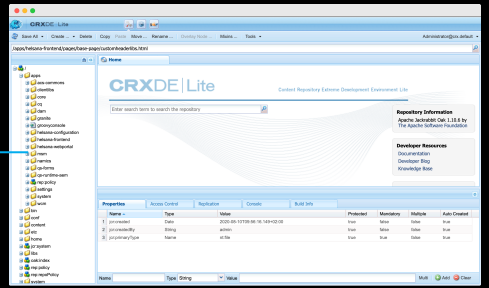
Solution Concept



Solution Concept

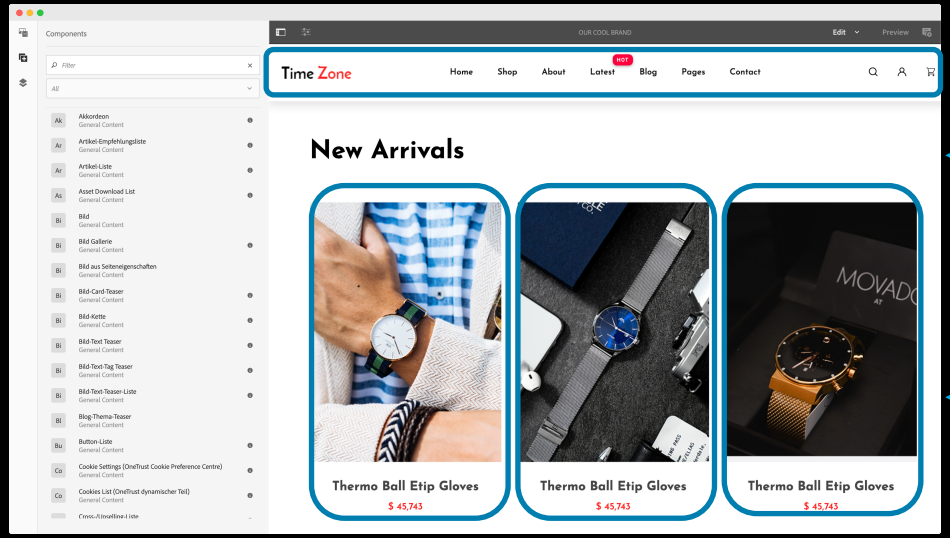


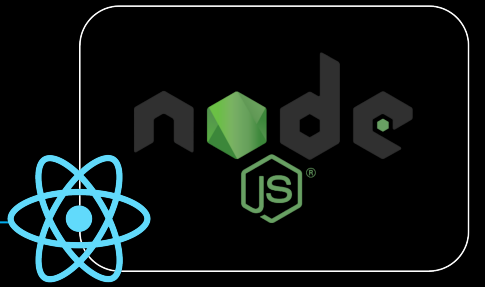
OSGI Service



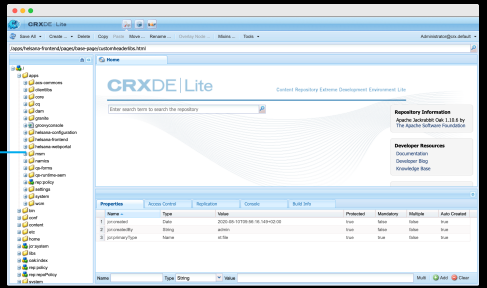
Solution Concept

- OSGI Service checks which components are used (in AEM)



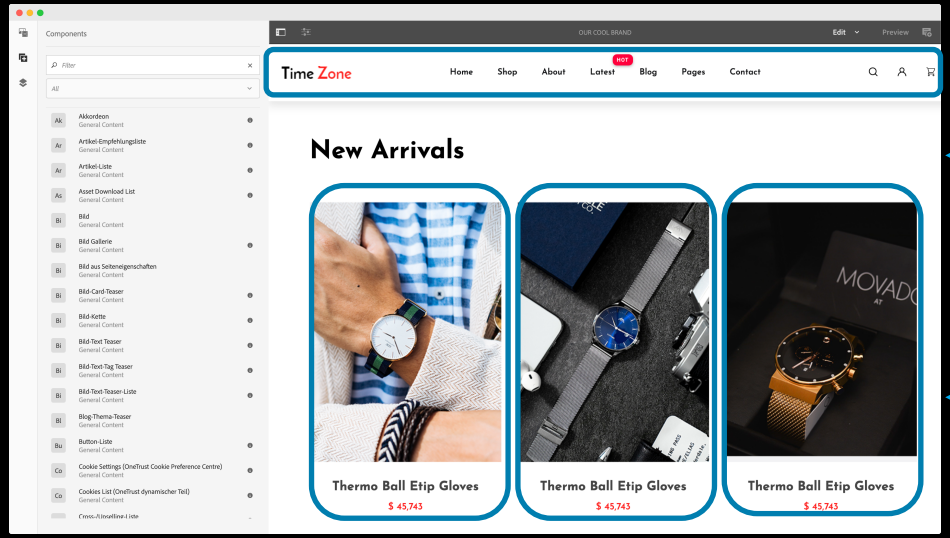


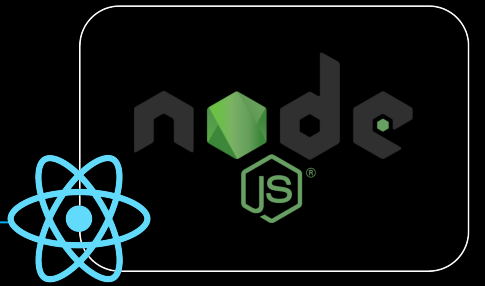
OSGI Service



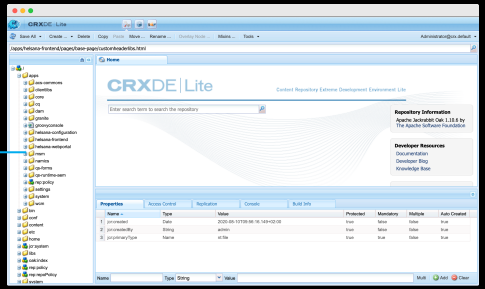
Solution Concept

- OSGI Service checks which components are used (in AEM)
- Calls Frontend Service to get needed assets (JS / CSS)



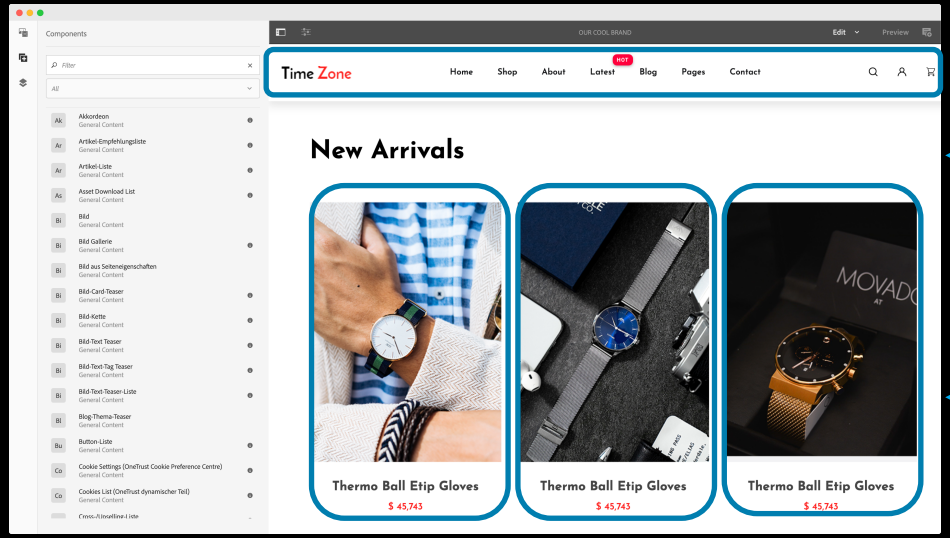


OSGI Service

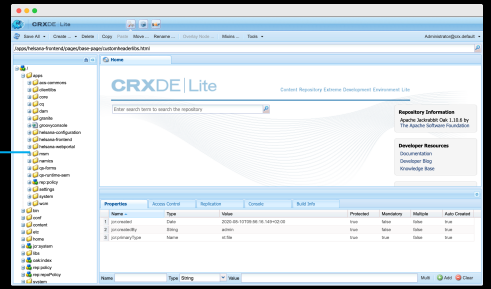


Solution Concept

- OSGI Service checks which components are used (in AEM)
- Calls Frontend Service to get needed assets (JS / CSS)
- Same Concept for all other systems



OSGI Service



Frontend Architecture

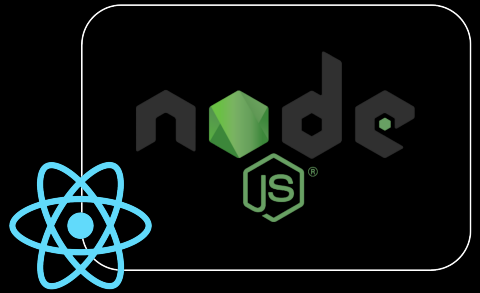
- OSGI Service reads page content and gets all apps
- Calls Frontend Service to get assets

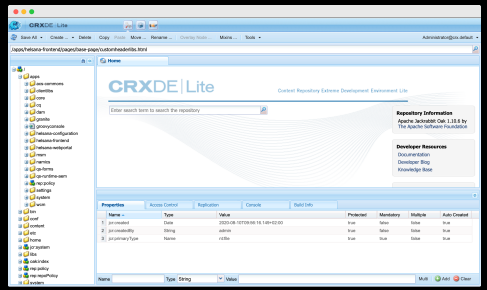
```
<sly data-sly-use.fs="de.teclead.FrontendService"/>
<head>
    ${fs.getAppsCSS}
</head>

<body>
    <div class="fs-navigation" data-params={...}>
    <div class="fs-product-teaser" data-params={...}>
    <div class="fs-foooter" data-params={...}>
    ${fs.getAppsJS}
</body>
```

Frontend Architecture

- Micro Apps concept is used
- OSGI Service injects JCR data into the components as JSON
- Sling Model can be used for adding business logic

```
<head>
    <link rel="stylesheet" href="fs.brand.com/apps/navigations.css">
    <link rel="stylesheet" href="fs.brand.com/apps/product-teaser.css">
    <link rel="stylesheet" href="fs.brand.com/apps/fs-foooter.css">
</head>
<body>
    <div class="fs-navigation" data-params={...}>
    <div class="fs-product-teaser" data-params={...}>
    <div class="fs-foooter" data-params={...}>

    <!-- dynamic by sightly -->
    <script src="fs.brand.com/apps/navigations.js">
    <script src="fs.brand.com/apps/product-teaser.js" />
    <script src="fs.brand.com/apps/fs-foooter.js" />
</body>
```

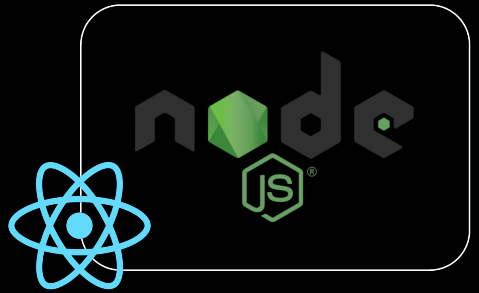


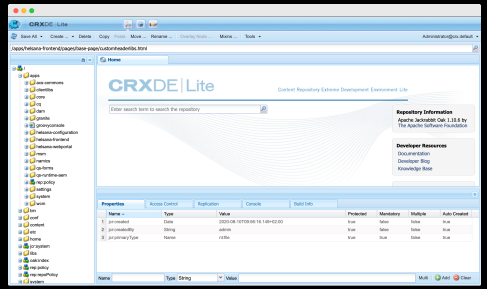


Frontend Architecture

- Inline option available => no external Domain calls
- OSGI service injects assets into the page directly

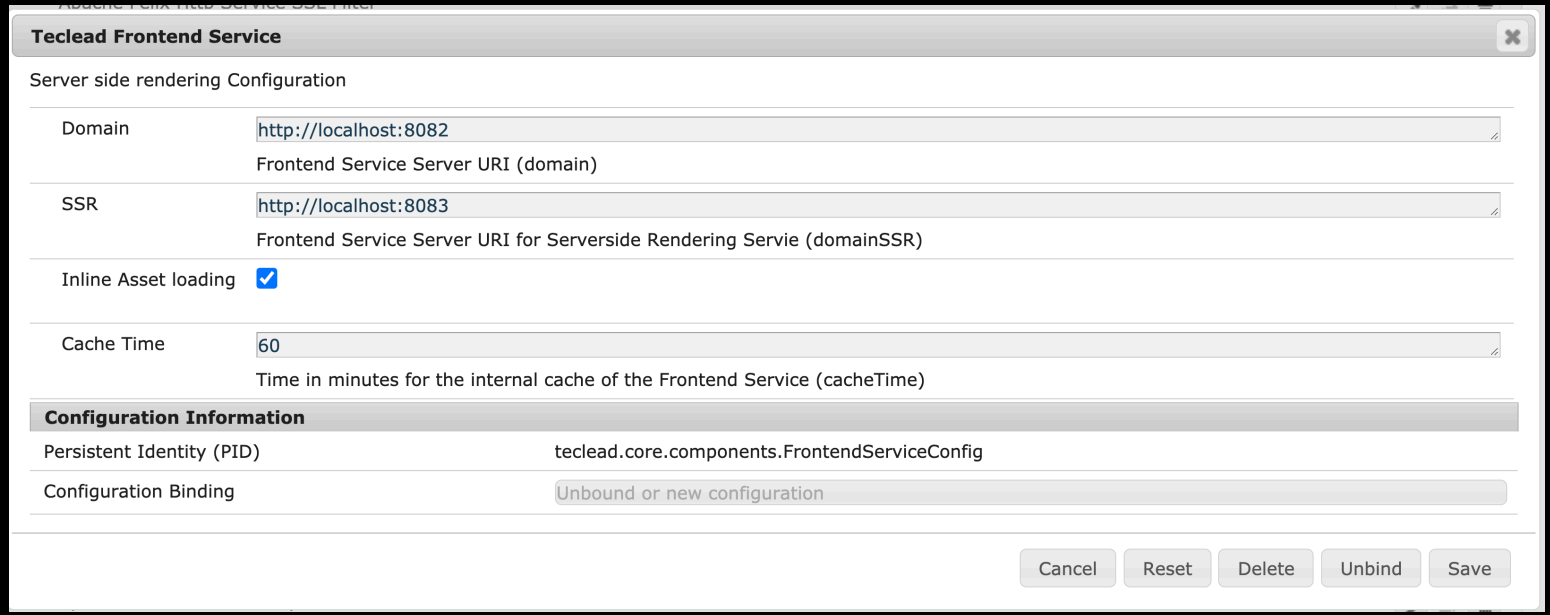
```
<head>
    <stlye>
        {inline css from service}
    </stlye>
</head>
<body>
    <div class="fs-navigation" data-params={...}>
    <div class="fs-product-teaser" data-params={...}>
    <div class="fs-foooter" data-params={...}>

    <!-- dynamic by sightly -->
    <script>
        {inline js from service}
    </script>
</body>
```





Frontend Architecture

- OSGI Config available
- Inline options for internal loading



Frontend Architecture

Sightly Component

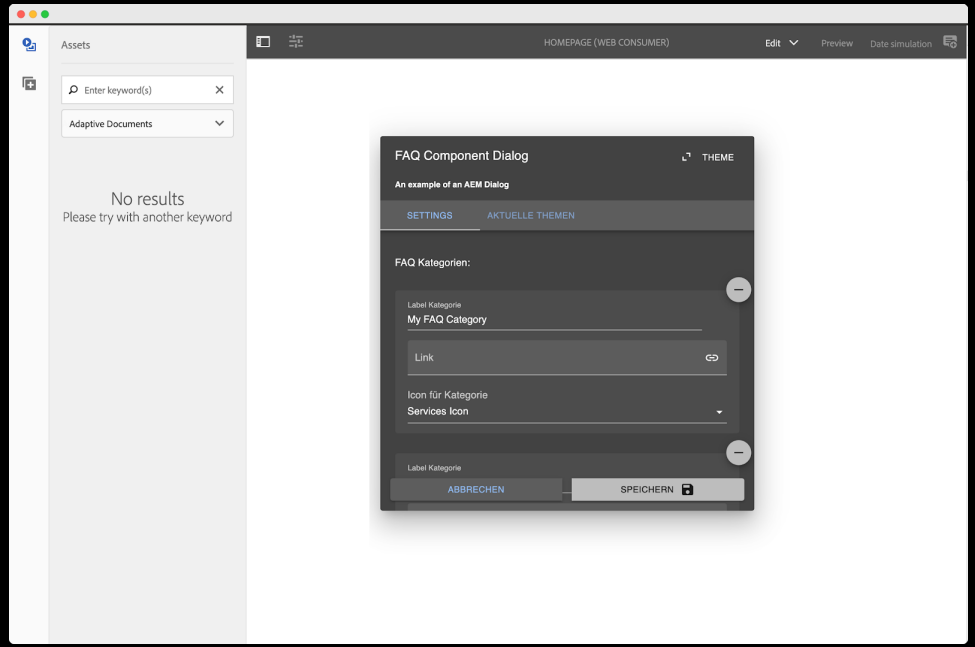
- All components use the same ResourceSuperType
- Removes complexity out of Sightly templates
- Can be used together with Serverside Renering
- More Details can be found in last years presentation:
- https://www.youtube.com/watch?v=Byfs7ptHhiU

```
<div data-sly-use.fs="de.teclead.FrontendService" class="aem-react-component ${fs.componentName}"
    data-params="${fs.data}">
    <!-- Component render here, SSR is optional-->
</div>
```

Authoring

Authoring

One Dialog to rule them all



brand.com

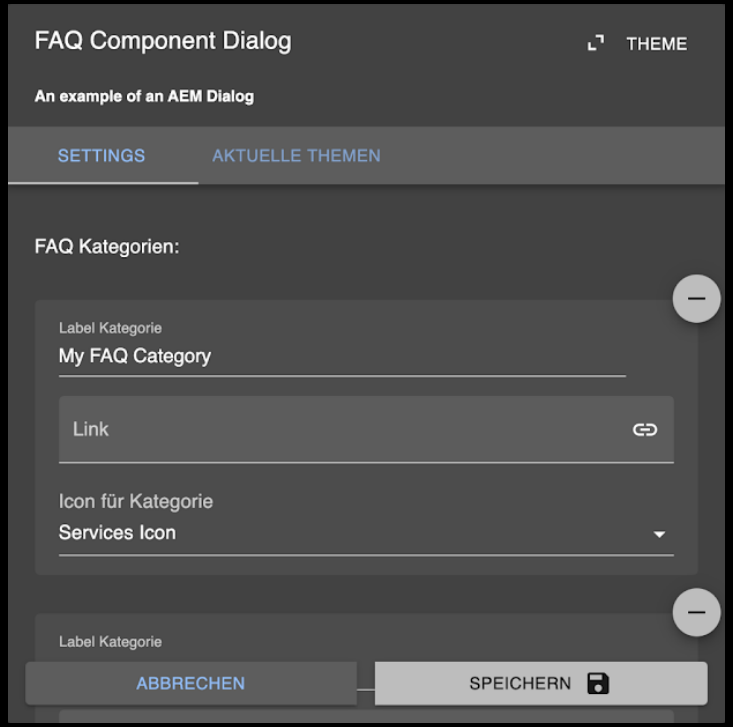
SAP Hybris

brand.com/shop

brand.com/user-area

Authoring

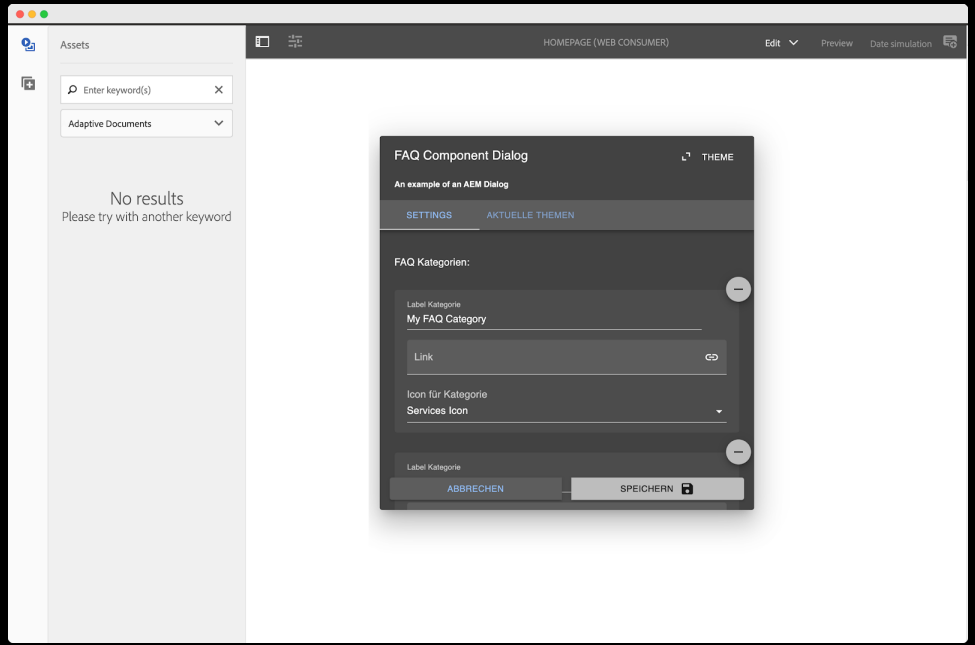
- Authors can decide: Unified vs. Touch UI
- All dialogs are shipped from the Frontend Service
- Data is stored in the JCR
- Works with the standard Dialog as well
- Dialog can be used in all systems



- https://www.npmjs.com/package/@teclead/aem-generator
- https://www.npmjs.com/package/@teclead/dialog-generator-react

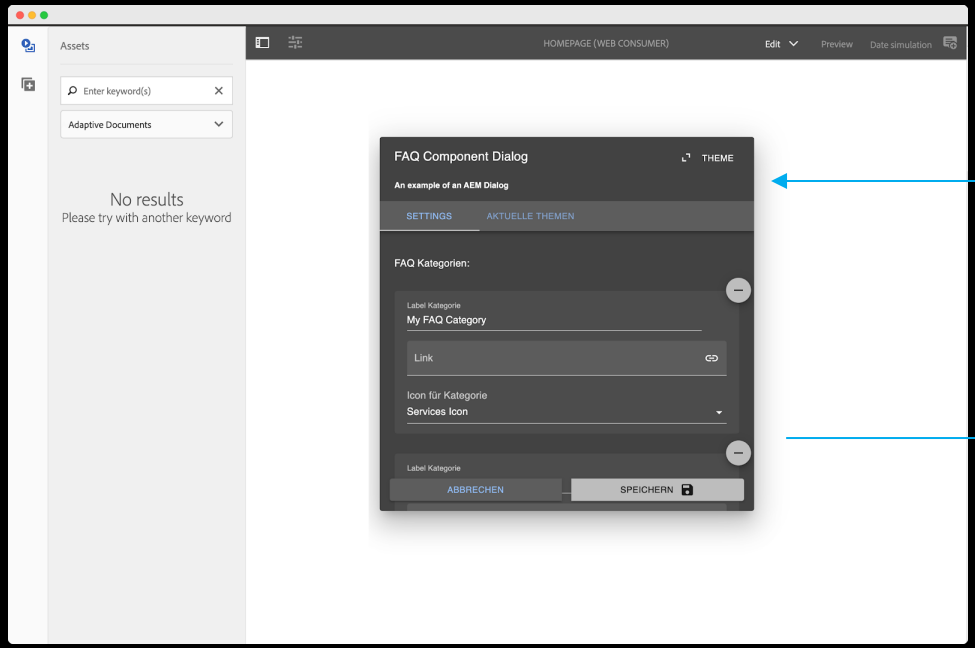
Authoring

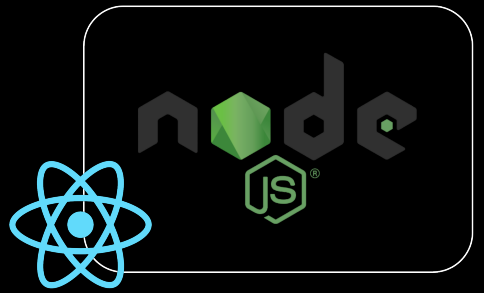
- Dialogs POSTs data to an AEM Servlet
- Stores it like the normal AEM dialog in the JCR

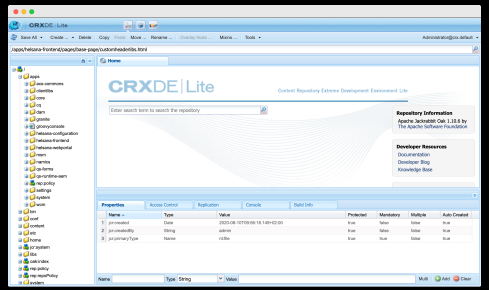


Authoring

- Dialogs POSTs data to an AEM Servlet
- Stores it like the normal AEM dialog in the JCR







Demo

FAQ

adaptTo()

FAQ

Does evertything needs to be in the Frontend Service?

- architecture can be mixed with normal AEM development

adaptTo()

FAQ

Does evertything needs to be in the Frontend Service?

- architecture can be mixed with normal AEM development

Does it work with Core Components?

- React components still use Core Components and child parsys

adaptTo()

FAQ

Does evertything needs to be in the Frontend Service?

- architecture can be mixed with normal AEM development

Does it work with Core Components?

- React components still use Core Components and child parsys

What about performance?

- components are cached in AEM / Dispatcher, no performance bottleneck

adaptTo()

FAQ

Does evertything needs to be in the Frontend Service?

- architecture can be mixed with normal AEM development

Does it work with Core Components?

- React components still use Core Components and child parsys

What about performance?

- components are cached in AEM / Dispatcher, no performance bottleneck

Can we use touch UI Dialog only?

- Yes! The external dialog is optional for 3rd party systems

Contact

Tony Schumacher

Partner

Mail: tony.schumacher@teclead.de

HQ: Berlin