

adaptTo()

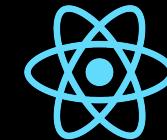
EUROPE'S LEADING AEM DEVELOPER CONFERENCE
28th – 30th SEPTEMBER 2020

Components as a Service
Tony Schumacher, Teclead



Agenda

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





Project Situation



Project Situation

Home

Time Zone

Select Your New Perfect Style

Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Aute irure.

SHOP NOW

New Arrivals



brand.com

Shop

Time Zone

NewestArrivals Price High To Low Most Popular 40 per page

Thermo Ball Etip Gloves \$ 45,743 Thermo Ball Etip Gloves \$ 45,743

SAP Hybris (x)

brand.com/shop

User Area

Time Zone

Welcome Back ! Please Sign in now

New to our Shop?

There are advances being made in science and technology everyday, and a good example of this is the

CREATE AN ACCOUNT

LOG IN

Forget Password?

Time Zone Quick Links New Products

About Offers & Discounts Woman Cloth

Get Coupon Fashion Accessories Man Accessories

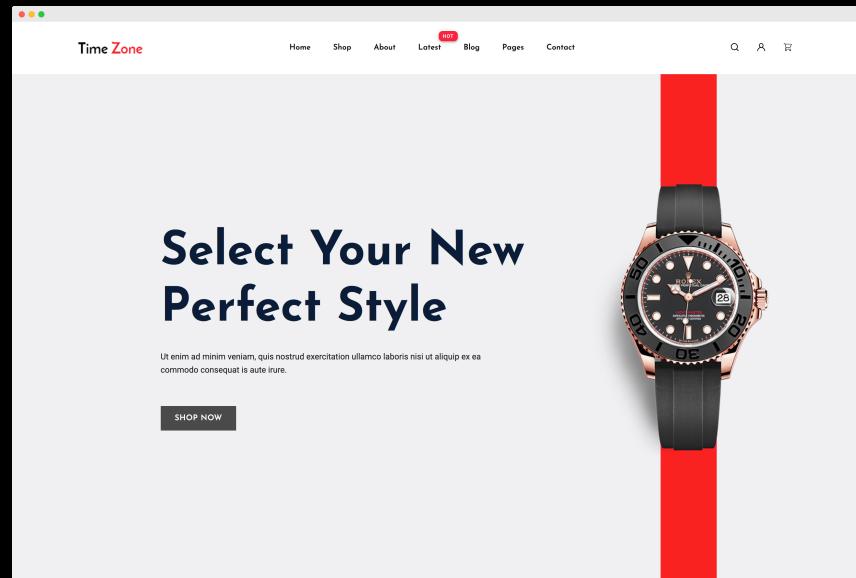
Contact Us Rubber made Toys



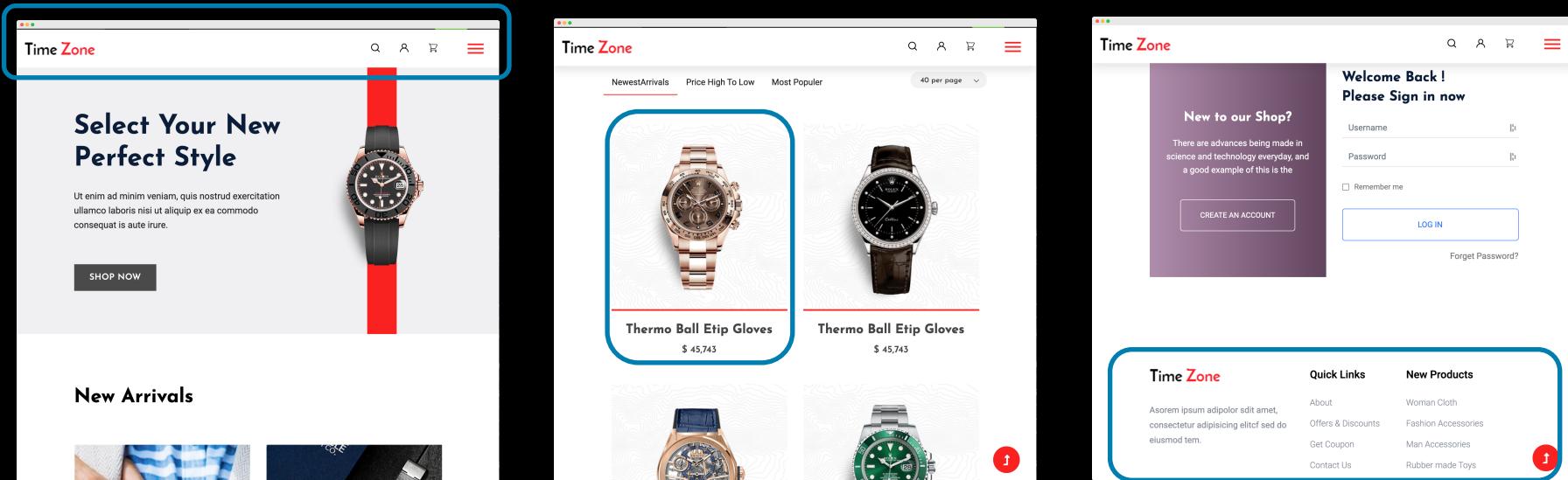
brand.com/user-area

Project Situation

- company uses multiple systems in one platform
- 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

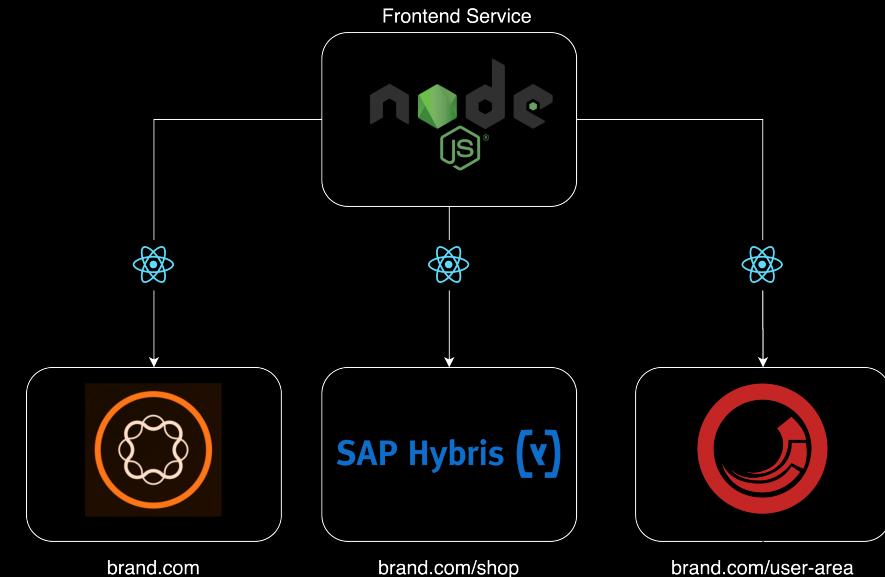


Solution Concept

Solution Concept

Specs:

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

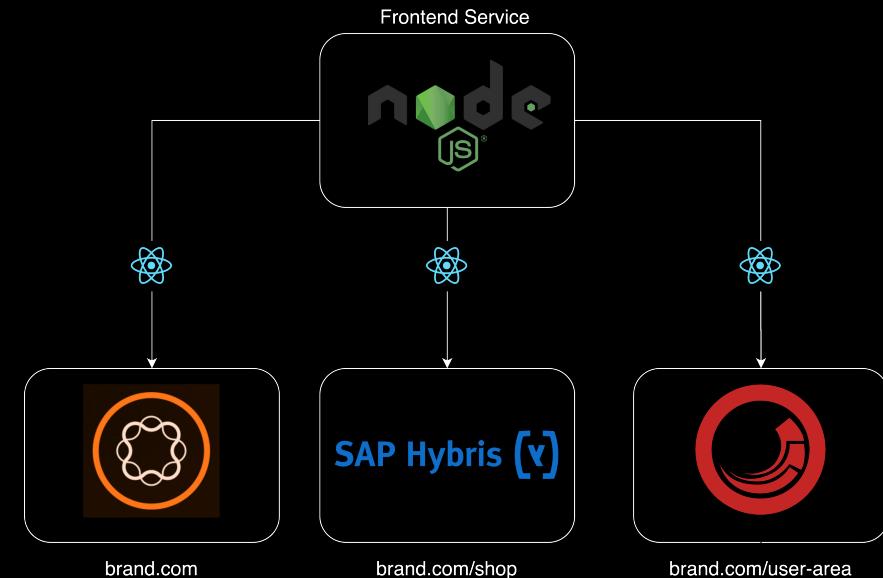


Solution Concept

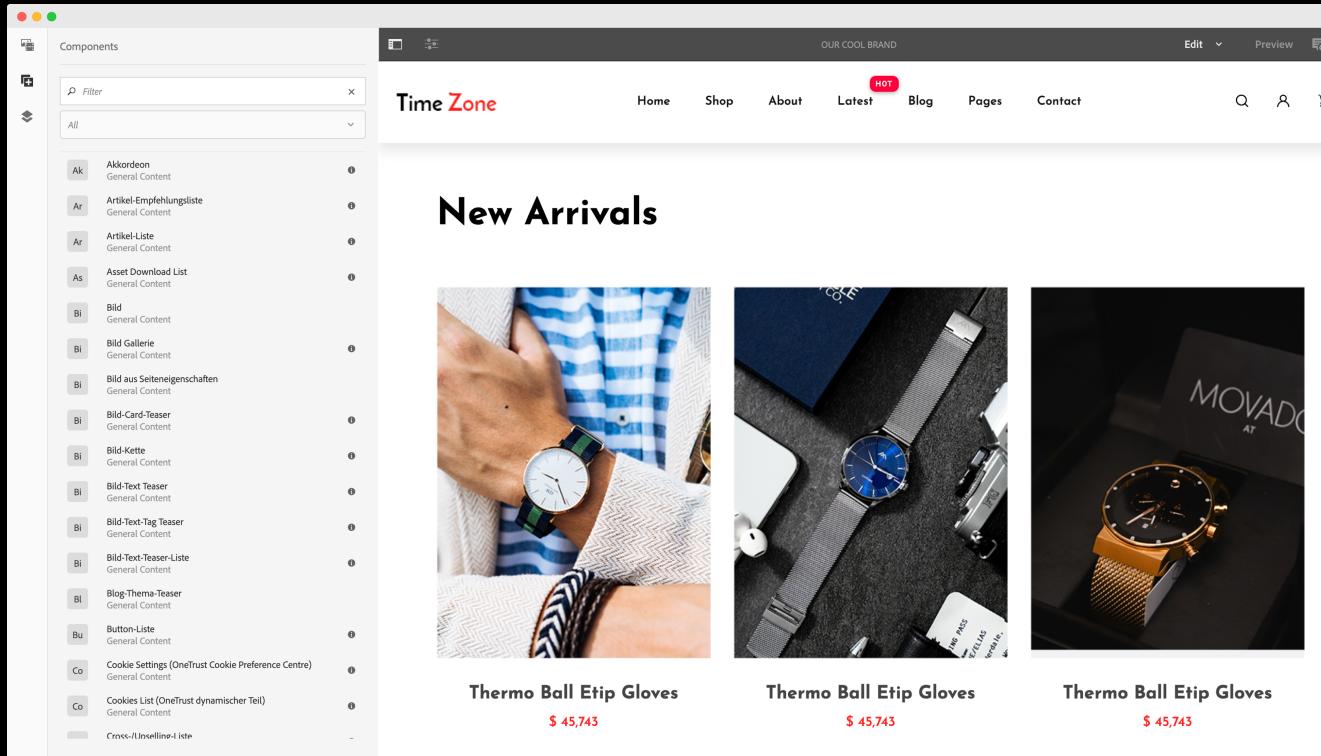
What it solves:

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

- ✓ no AEM deployment needed for new frontend components



Solution Concept



The screenshot illustrates the adaptTo() solution concept, showing a storefront interface and its corresponding component library.

Component Library: On the left, a sidebar titled "Components" lists various reusable components. The list includes:

- Ak: Akkordeon General Content
- Ar: Artikel-Empfehlungsliste General Content
- Ar: Artikel-Liste General Content
- As: Asset Download List General Content
- Bild: Bild General Content
- Bi: Bild Gallerie General Content
- Bi: Bild aus Seiteigenschaften General Content
- Bi: Bild-Card-Teaser General Content
- Bi: Bild-Kette General Content
- Bi: Bild-Text-Teaser General Content
- Bi: Bild-Text-Tag-Teaser General Content
- Bi: Bild-Text-Teaser-Liste General Content
- Bl: Blog-Thema-Teaser General Content
- Bu: Button-Liste General Content
- Co: Cookie Settings (OneTrust Cookie Preference Centre) General Content
- Co: Cookies List (OneTrust dynamischer Teil) General Content
- Cross-/Inselliste

Storefront: The main area shows a storefront for "Time Zone". The header includes "OUR COOL BRAND", "Edit", "Preview", and navigation links for Home, Shop, About, Latest (highlighted with a red "HOT" badge), Blog, Pages, and Contact. A search bar and user icons are also present.

The page content features a section titled "New Arrivals" with three product cards:

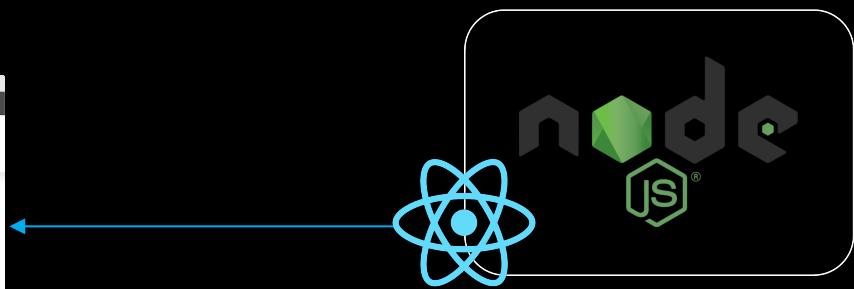
- Thermo Ball Etip Gloves** - \$ 45,743
- Thermo Ball Etip Gloves** - \$ 45,743
- Thermo Ball Etip Gloves** - \$ 45,743

Each card displays a product image (a watch), the product name, and the price.



Solution Concept

The screenshot displays a web application interface. On the left, there is a sidebar titled "Components" containing a list of various content types such as Accordion, Article, Asset Download List, Bild, Bild Gallerie, Bild mit Setzergenschaften, Bild Card-Poster, Bild-Kette, Bild-Trot-Teaser, Bild-Text-Tag-Teaser, Bild-Text-Teaser-Liste, Blog, Blog-Theme-Teaser, Button-Liste, and Cookies. The main content area features a "New Arrivals" section with three product cards. Each card includes a small image, the product name "Thermo Ball Etip Gloves", and a price of "\$ 45,743". The top navigation bar includes links for Home, Shop, About, Latest, Blog, Pages, Contact, and a search icon.

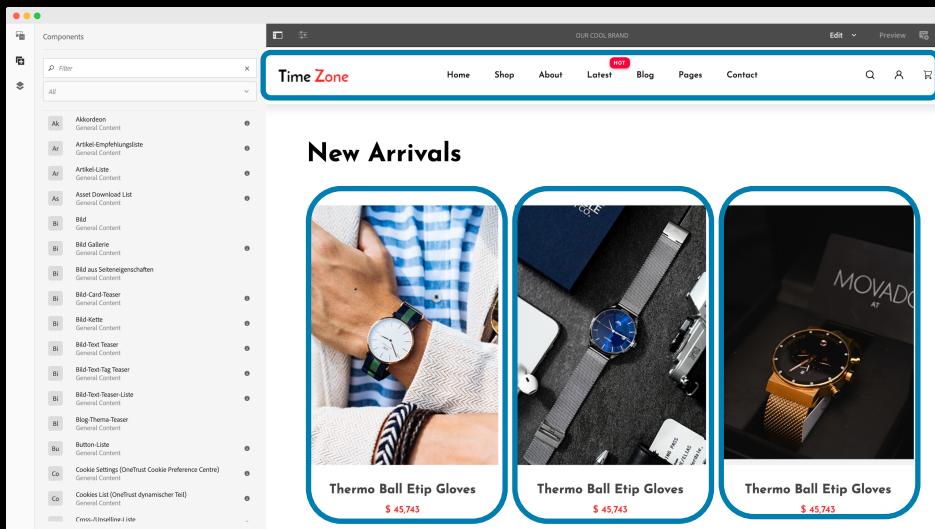


OSGI Service

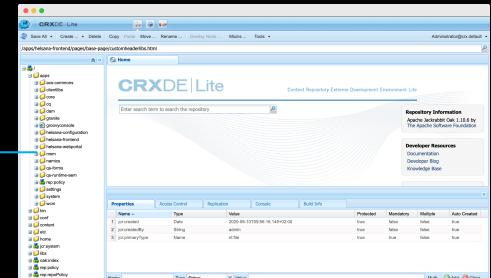
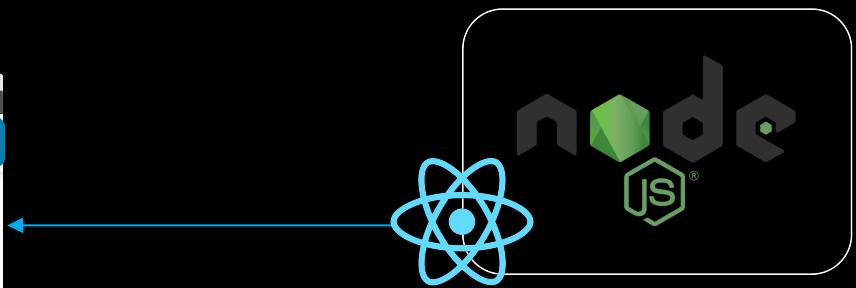
The screenshot shows the CRXDE Lite interface, which is a Content Repository Extensible Development Environment. It features a left-hand sidebar with a file tree and a central workspace. The workspace displays a properties table for a selected item. The table includes columns for Name, Type, Value, Protected, Mandatory, Multiple, and Auto Create. There are two entries: "processed" (String) and "processedType" (String). The "processed" entry has a value of "true". The "processedType" entry has a value of "true".

Solution Concept

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

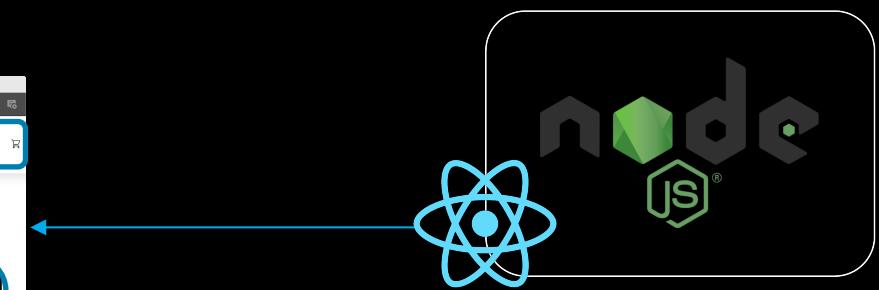
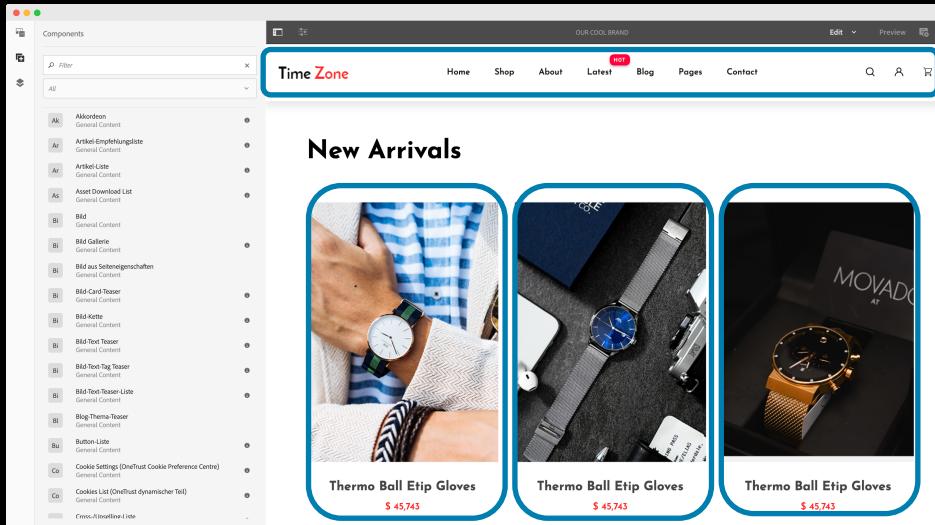


OSGI Service

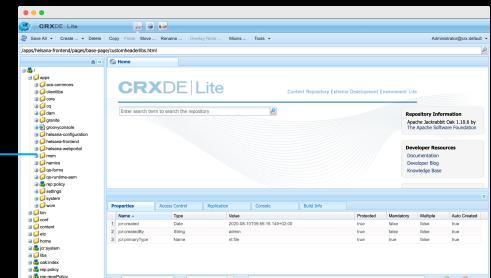


Solution Concept

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

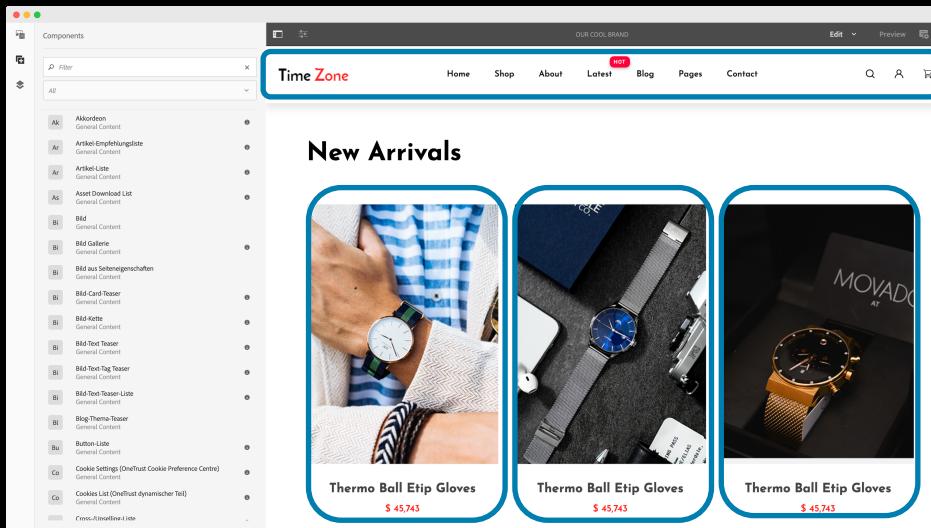


OSGI Service

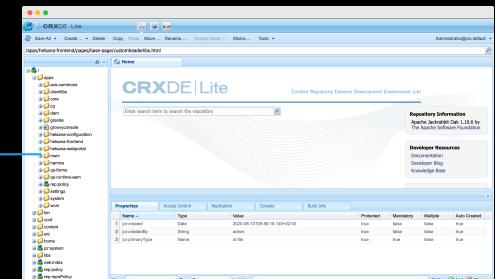
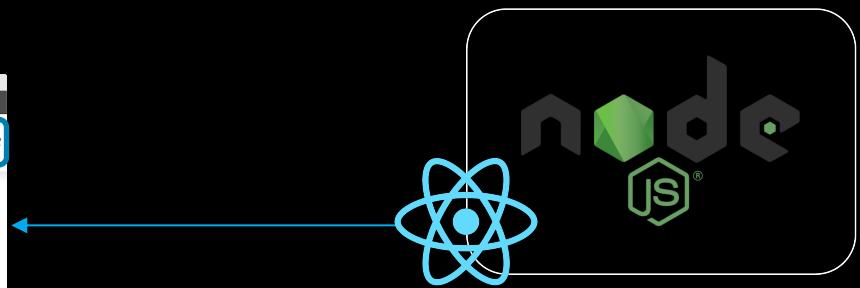


Solution Concept

- 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-fooooter" data-params="...>
        ${fs.getAppsJS}
</body>
```

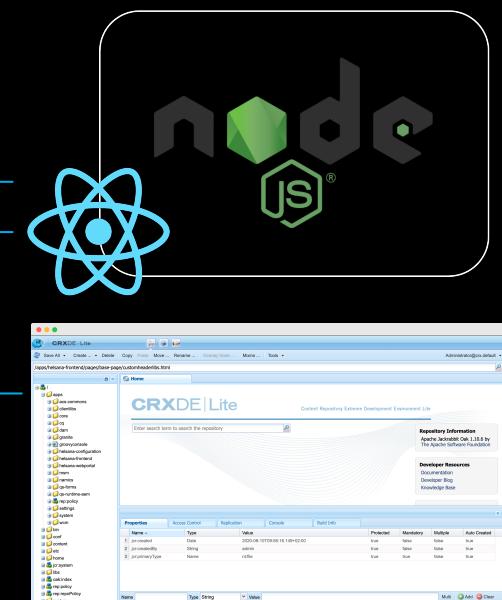
Frontend Architecture

- 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-fooooter.css">
</head>
<body>
    <div class="fs-navigation" data-params={...}>
    <div class="fs-product-teaser" data-params={...}><!-- dynamic by sightly -->
    <div class="fs-fooooter" data-params={...}>
        <script src="fs.brand.com/apps/navigations.js">
        <script src="fs.brand.com/apps/product-teaser.js" />
        <script src="fs.brand.com/apps/fs-fooooter.js" />
    </body>

```

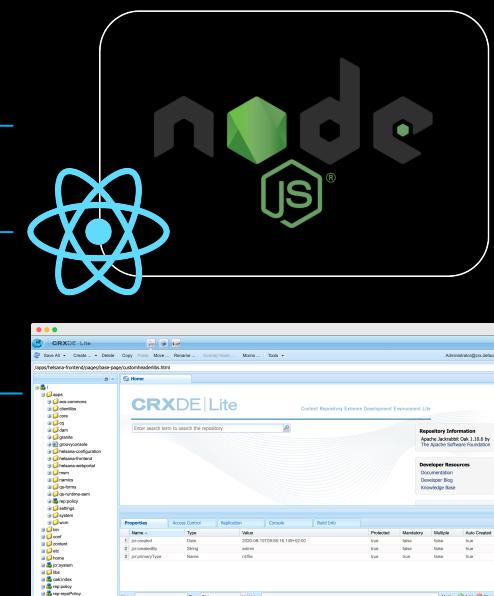


Frontend Architecture

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

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

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



Frontend Architecture

- OSGI Config available
- Inline options for internal loading

adobe Felix Frontend Service Configuration

Teclead Frontend Service

X

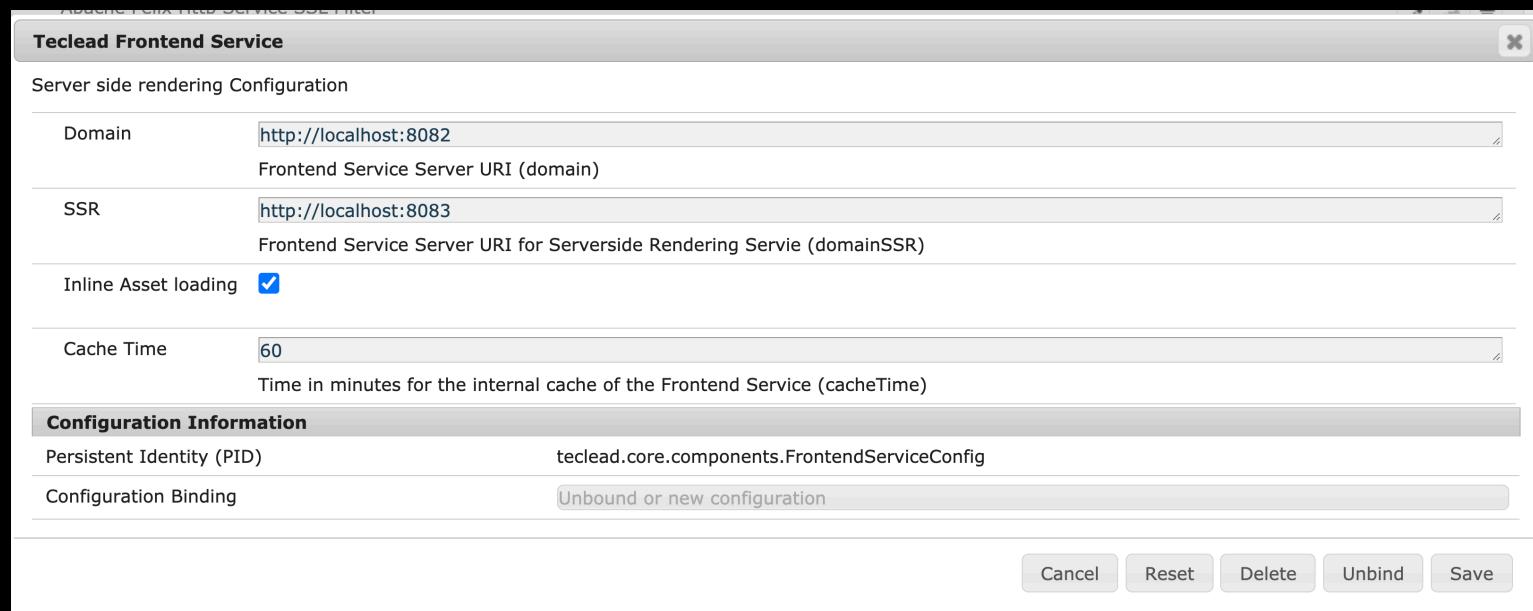
Server side rendering Configuration

Domain	<input type="text" value="http://localhost:8082"/>
Frontend Service Server URI (domain)	
SSR	<input type="text" value="http://localhost:8083"/>
Frontend Service Server URI for Serverside Rendering Servie (domainSSR)	
Inline Asset loading	<input checked="" type="checkbox"/>
Cache Time	<input type="text" value="60"/>
Time in minutes for the internal cache of the Frontend Service (cacheTime)	

Configuration Information

Persistent Identity (PID)	teclead.core.components.FrontendServiceConfig
Configuration Binding	Unbound or new configuration

Cancel Reset Delete Unbind Save





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>

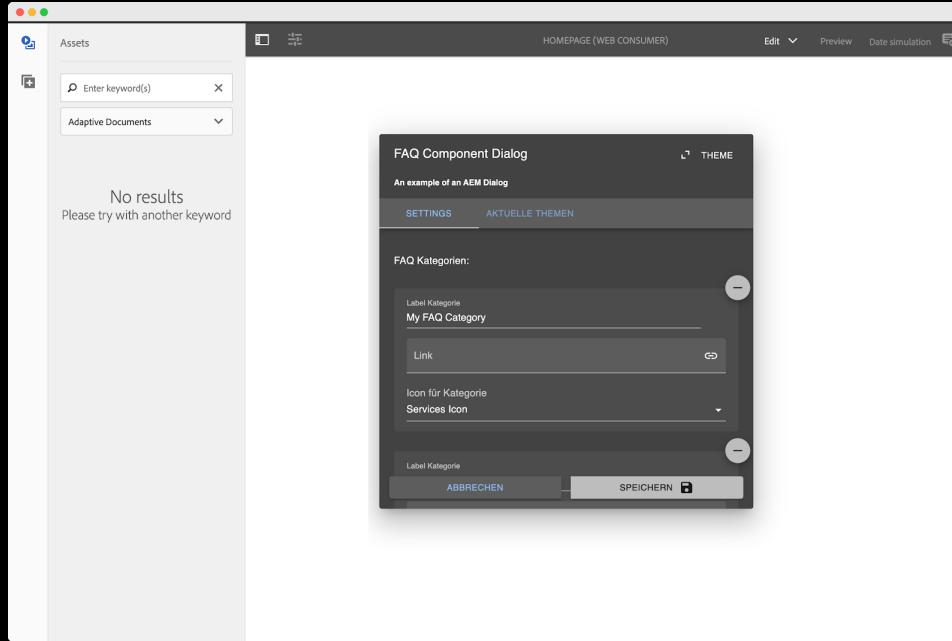
```
<div data-sly-use.fs="de.teclead.FrontendService" class="aem-react-component ${fs.componentName}"  
    data-params="${fs.data}">  
    <!-- Component render here, SSR is optional-->  
</div>
```



Authoring

Authoring

One Dialog to rule them all



The screenshot shows the AEM authoring interface with a search bar for 'Assets' and a dropdown for 'Adaptive Documents'. A modal dialog titled 'FAQ Component Dialog' is open, displaying the text 'An example of an AEM Dialog'. The dialog has tabs for 'SETTINGS' and 'AKTUELLE THÄMEN'. It contains fields for 'FAQ Kategorien' (with 'My FAQ Category' and 'Link' inputs), 'Icon für Kategorie' (with 'Services Icon' selected), and a 'Label Kategorie' section with 'ABBRECHEN' and 'SPEICHERN' buttons.



brand.com



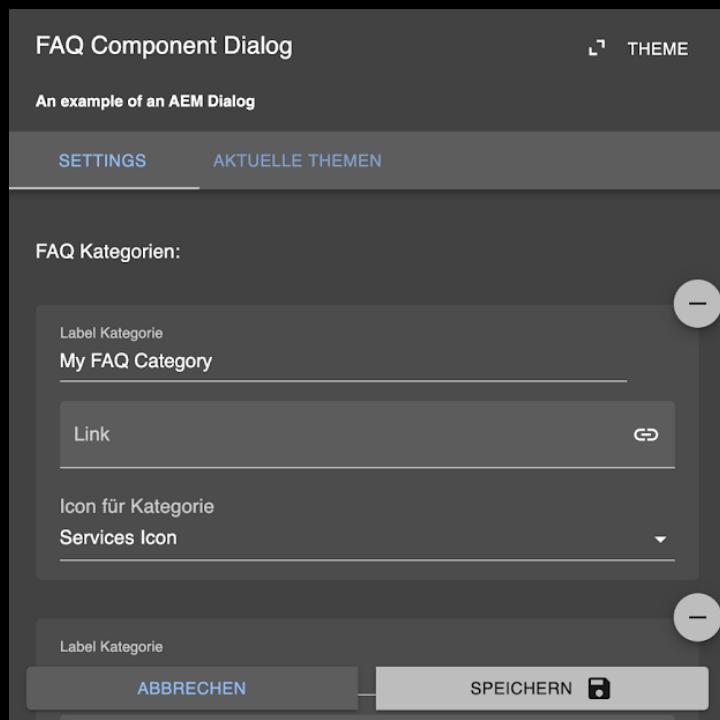
brand.com/shop



brand.com/user-area

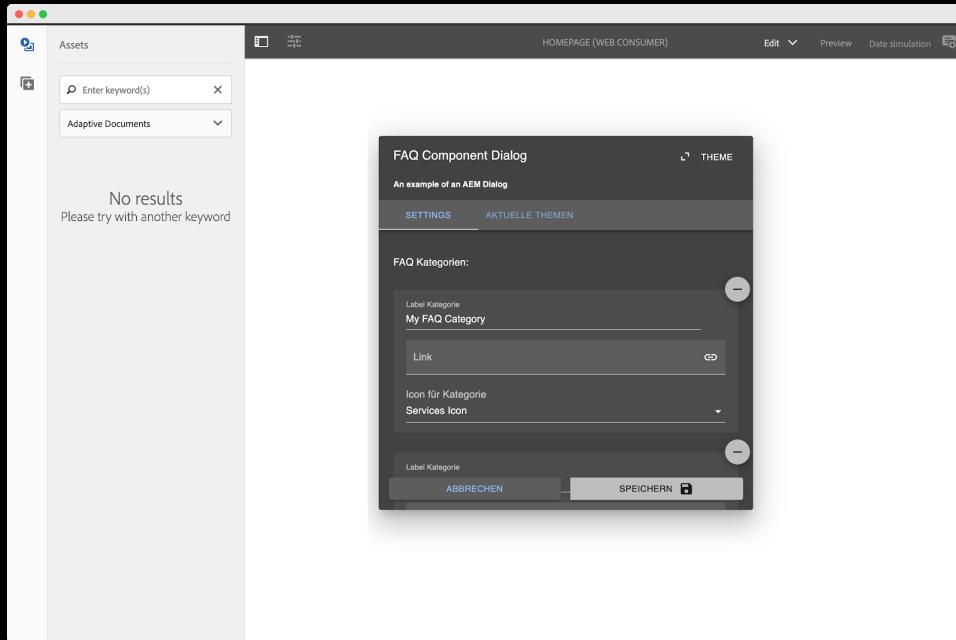
Authoring

- 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>



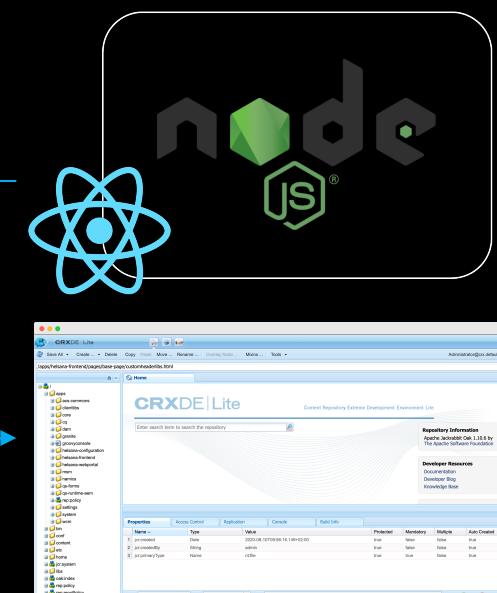
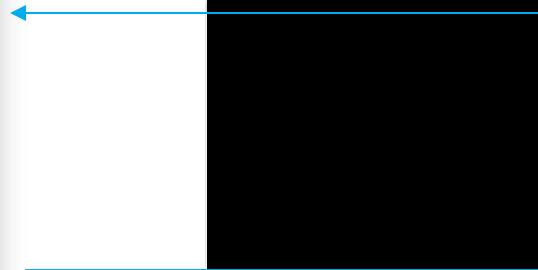
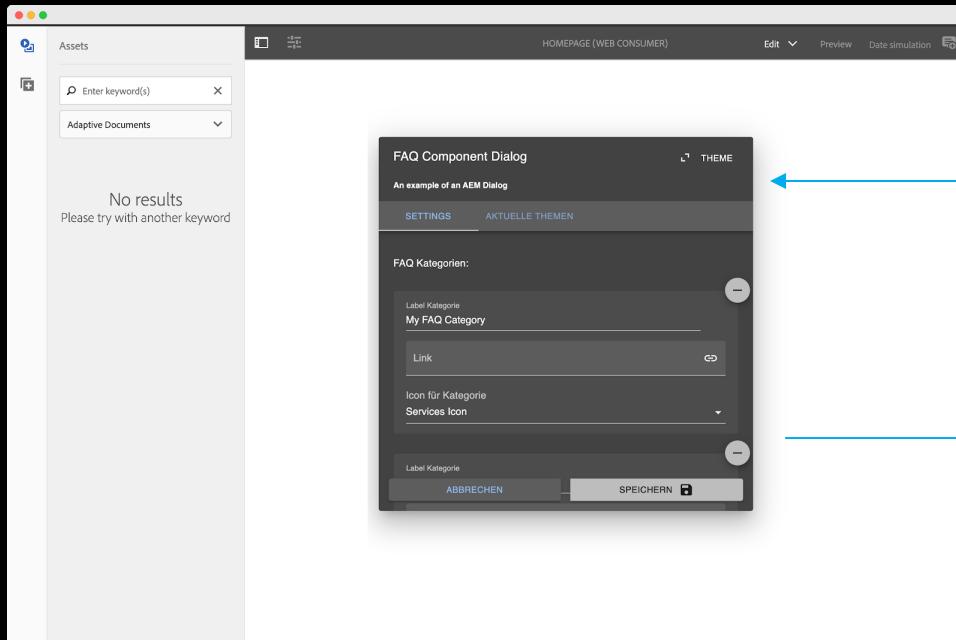
Authoring

- 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





adaptTo()

Demo



adaptTo()

FAQ



FAQ

Does everything needs to be in the Frontend Service?

- architecture can be mixed with normal AEM development



FAQ

Does everything 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

Does everything 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

Does everything 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

