



adaptTo()

EUROPE'S LEADING AEM DEVELOPER CONFERENCE
28th – 30th SEPTEMBER 2026

Hybrid AEM Architecture

Bridging server-side rendering with React at the component level



About Calix

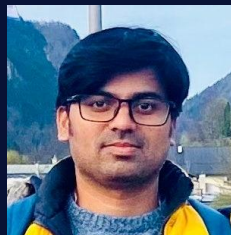
Calix — an AI-native cloud platform company helping Broadband Service Providers acquire, engage, and retain subscribers and grow their business across markets and communities





Daniel Jebaraj(*Danny*)

Engineering Manager, Web
Platform · Calix



Shashi Kumar

Web Technical Lead · Calix



Agenda

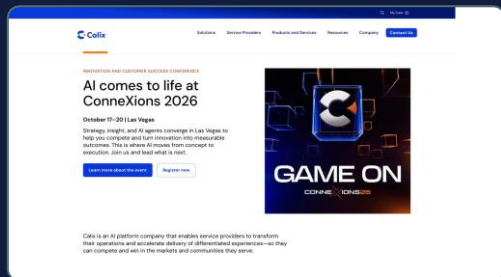
- 01 The Journey
- 02 The Dilemma
- 03 The Hybrid Approach
- 04 Architecture in Action
- 05 Demo
- 06 Under the hood
- 07 Outcome & Performance
- 08 Takeaway & QA



adaptTo()

The Journey

Calix Web Platform - Evolution



2024

Calix.com rebuilt

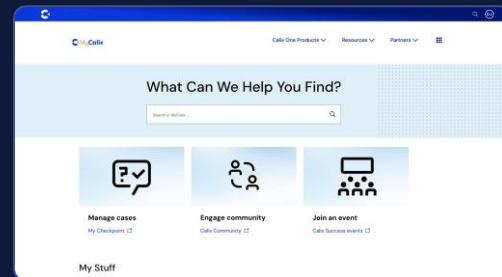
JSP to HTL, onto AEM as a Cloud Service.
50+ components, built for reuse across the
child portals.



UNTIL 2025

MyCalix as it was

The customer portal, still on the old build.
Different look, different component
vocabulary.



2026

MyCalix refreshed

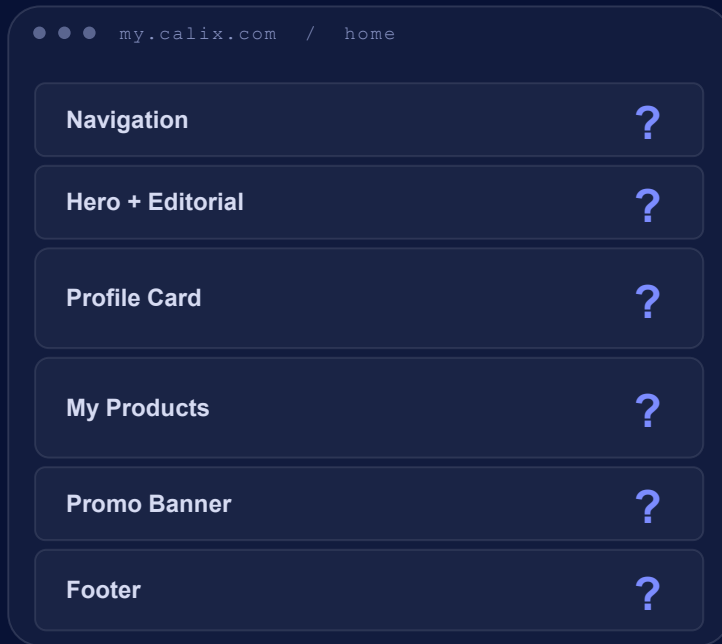
Same design language, same component set
— plus features the HTL components could
not carry.



adaptTo()

The Dilemma

CrossRoads - Which way to Go?



Choice of options

01

Full SPA Editor

SCOPE

Whole page or template

HTL SET

Rewritten from scratch

AEM CAPABILITIES

Constrained by the SPA contract

COST

Migration risk, up front

✗ NOT FOR US

02

Full Headless

SCOPE

Whole platform

HTL SET

Abandoned

AEM CAPABILITIES

Rebuilt outside AEM

COST

Platform capability lost

✗ NOT FOR US

03

Hybrid - (React + HTL)

SCOPE

One component

HTL SET

Kept, untouched

AEM CAPABILITIES

Kept, fully native

COST

Dev governance

✓ WHAT WE SHIPPED

Every option has a tradeoff. We chose governance over complete overhaul

The Union - Each brings something the other cannot

HTL BRINGS

- **Content in the HTML source**
Content delivery
- **Native AEM authoring**
Policies, governance
- **Cache-friendly delivery**
Dispatcher and Edge
- **Site-like experience**
Client - server handshake



- + Modern UX without sacrificing SEO
- + Crawlable and Interactive experiences

REACT BRINGS

- **Stateful interaction**
user engagement
- **Per-user behaviour**
Dynamic
- **App-like experience**
Performant experiences
- **Modern frontend ecosystem**
High fidelity interactions

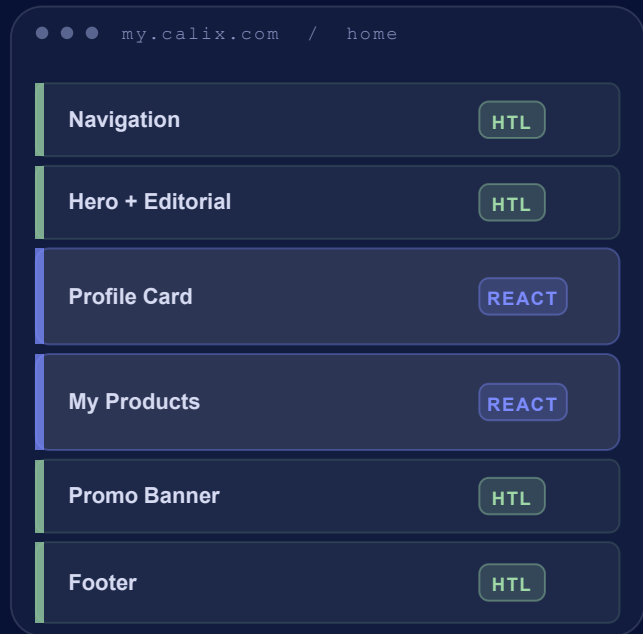


- + Preserve Native Authoring
- + Developer friendly Governance

The Hybrid Approach



Same page – Two Paths



Adobe Target

CUG

Experience Fragments

Multi-Site Manager

Adobe Data Collection

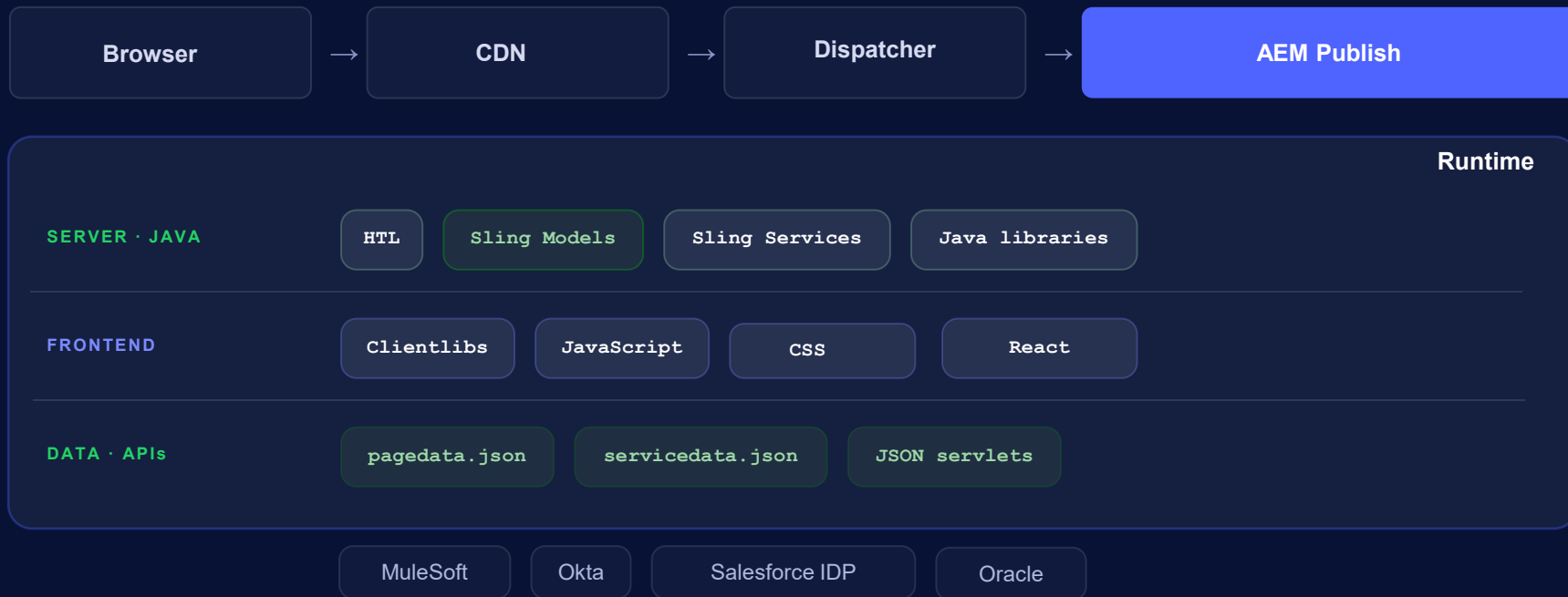
DAM workflows

Continue to leverage existing eco system

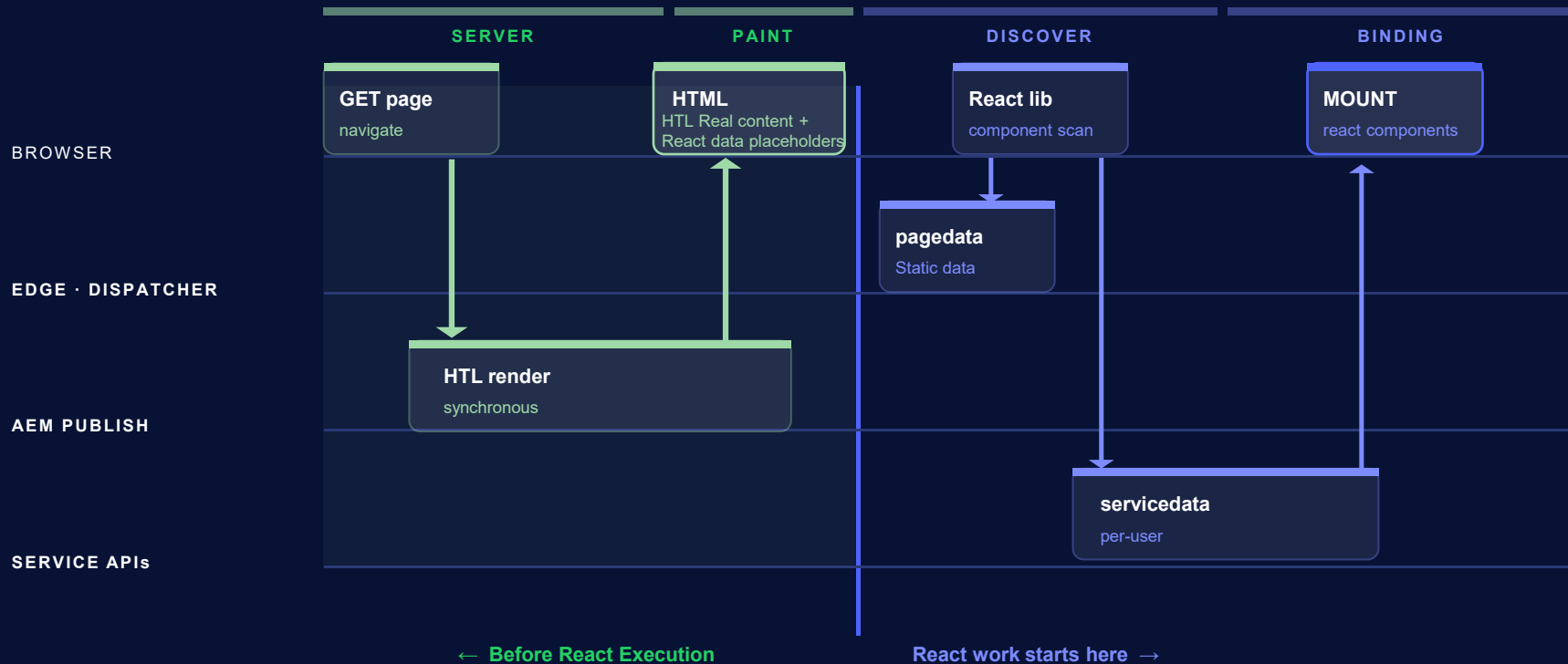
Architecture in Action



A peek into AEM Runtime



The interaction





adaptTo()

Demo

Live data after paint

Network panel

Author's view

Under the Hood

profile-card.html

1

```
<div class="react-component"
    data-component="${resource.resourceType}"
    data-component-id="${resource.name}">
```

2

```
<!-- Skeleton - hidden while an author edits or previews -->
<sly data-sly-test="${!wcmmode.edit && !wcmmode.preview}">
  <div class="cmp-profile-card--skeleton" aria-hidden="true">
    <span class="skeleton-block skeleton__image"></span>
    <span class="skeleton-block skeleton__name"></span>
  </div>
</sly>
</div>
```

3

```
<!--SEO or SSR Need-add meta data or component level structured data(JSON-LD)outside react-component-->
<meta name="description" content="${properties.metaDescription}/>
```

class

react-component : What the loader looks for

data-component

resourceType : React mount point

data-component-id

resourceName : Json binding key



adaptTo()

Two Sling Models. Zero boilerplate.

ProfileCardStaticModel.java

```
@Model(adapters = GenericStructureStaticModel.class,
      resourceType = ".../react/profile-card")
public class ProfileCardStaticModel
    implements GenericStructureBaseModel {
}
```

ProfileCardServiceModel.java

```
@Model(adapters = GenericStructureServiceModel.class,
      resourceType = ".../react/profile-card")
public class ProfileCardServiceModel
    implements GenericStructureServiceModel {
    // read UserManager / external API
}
```

home.pagedata.json - authored static data

```
{
  "components": [
    {
      "resourceType": ".../profile-card",
      "componentId": "profile_card",
      "order": 0,
      "isReact": true,
      "data": {
        "greetingPrefix": "Welcome back,",
        "sectionLabel": "My Profile"
      }
    }
  ]
}
```

home.servicedata.json - per-user dynamic data

```
{
  "components": [
    {
      "resourceType": ".../profile-card",
      "componentId": "profile_card",
      "order": 0,
      "data": {
        "user": {
          "fullName": "Administrator",
          "userId": "admin"
        }
      }
    }
  ]
}
```

Shared key: **componentId**, **resourceType**



Register once. The loader does the rest.

UI Frontend

```
ui.frontend/  
├─ site/  -> clientlib.site  (HTL Components)  
└─ react/ -> clientlib.react (React components)
```

React -> index.js

```
const COMPONENTS = {  
  ".../react/profile-card": ProfileCard;  
  
const [pageData, serviceData] = await Promise.all([  
  loadPageData(), loadServiceData(),  
]);  
  
document.querySelectorAll(".react-component").forEach((el) => {  
  const Component = COMPONENTS[el.dataset.component];  
  if (!Component) return; // Not registered: remains HTL  
  
  const id = el.dataset.componentId;  
  const props = {  
    ...pageData.components[id],  
    dynamicData: serviceData.components[id],  
  };  
  createRoot(el).render(<Component {...props} />);  
});
```

Outcome & Performance

One Page – Two Rendering options

HTL

Server-side rendering

Sling Model → HTML

React

Client-side rendering

Sling Model → JSON

What Stays

AEM Runtime

Existing Platform Service

Authoring Experience



Performance

0 KB

Page without react

2 API Call

servicedata & pagedata call per page

On Demand

Invoke libs on demand of react components

Measured on same templates(*Lab tested*)

Metric

vs. the old portal

Performance

FCP, LCP, TBT

~21% faster

Best Practices

~10% better

Key Takeaways

Takeaways & Recommendations

1 Decide at component level, not per page

2 Enables SEO to work with React

3 Authored and User data stay apart

4 Register once – the loader does the rest

5 Existing AEM investment stays untouched

USE IT WHEN

- A mature AEM investment already exists
- React is on demand, not forced
- SEO and server-rendered HTML needed

HTL

+

React

Let each component decide.

Thank you · Questions?