



Adobe

Adobe Experience Manager 6.5 – What's new for Developers

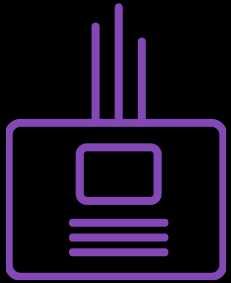
Gabriel Walt & Cedric Huesler – Product Management



Agenda

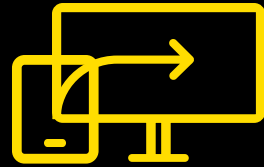
- **6.5 Focus and upgrade** (Cedric)
- **Java** (Cedric)
- **HTML Template Language** (Gabriel)
- **Core Components** (Gabriel)
- **HTTP APIs** (Cedric)
- **JS SDK for React & Angular** (Gabriel)
- **Practitioner UI improvements** (Cedric)

Focus areas for Adobe Experience Manager 6.5



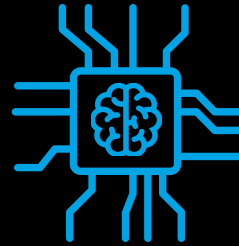
Content Velocity

Streamlined content delivery with easy-to-use development, authoring and delivery tools for marketers, creatives & IT.



Fluid Experiences

Channel-agnostic authoring with re-usable content blocks and headless delivery.



Experience Intelligence

Data-driven insights, content discovery and dynamic composition of next best experience powered by Adobe Sensei.



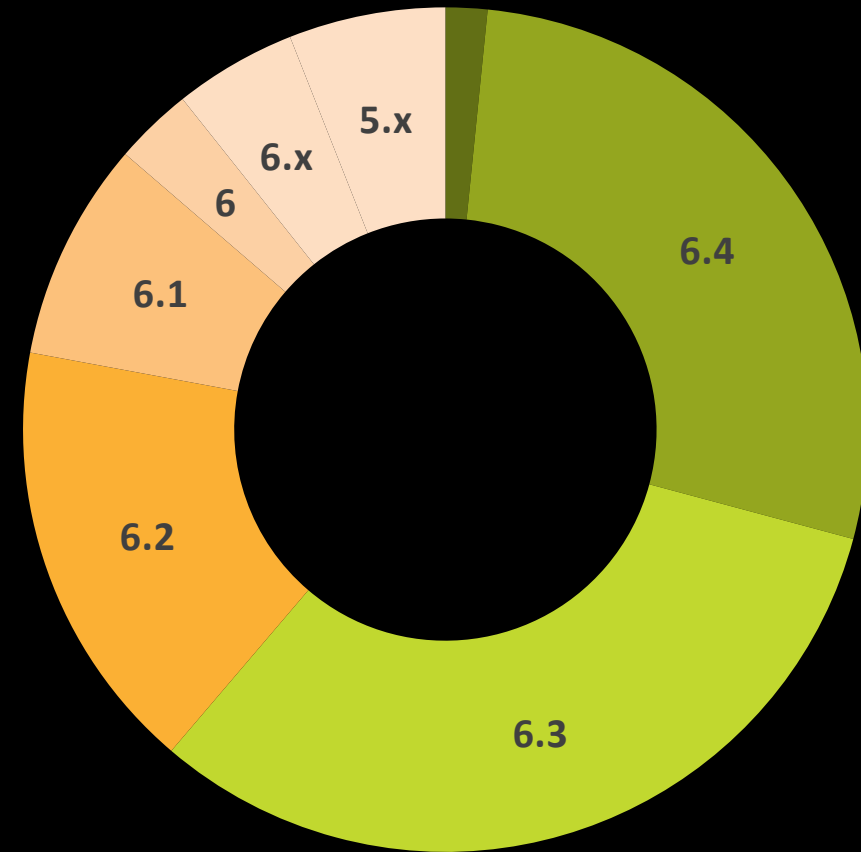
Cloud Agility

Cloud native capabilities to enable brands to quickly adapt and innovate their digital transformation of the customer experience.

Adobe Experience Manager 6.5 Upgrade

- 3 media & entertainment sites that are in the Top 1000 websites in North America are live on AEM 6.5
- Total 9 sites live on AEM 6.5
- Customers on Adobe Managed Services get step-by-step upgrade to AEM 6.5 in Cloud Manager
- General Availability of AEM 6.5 in April 2019
- Hands-on Labs at Summit on AEM 6.5

~26.8k hostnames (author & publish)
March 6, 2019 looking back 30 days



Adobe Experience Manager – Java Support

Recommended Java version:

- AEM 6.2 - 6.4 Java 8
- AEM 6.5 Java 11

How to get the latest Java version?

1. Check for download on <http://java.sun.com> (will redirect to oracle.com)
2. If the version can't be downloaded – contact Adobe Support via ticket and mention what version of Java you need.

Public FAQ: adobe.ly/2SdHy66

HTML Template Language 1.4

- “in” operator for strings, arrays and objects:

```
`${'a' in 'abc'}`  
`${100 in myArray}`  
`${'a' in myObject}`
```

- Variable declarations with `data-sly-set`:

```
<sly data-sly-set.title="${currentPage.title}"/>${title}
```

- List and repeat control parameters: `begin`, `step`, `end`:

```
<h2 data-sly-repeat="${currentPage.listChildren @ begin = 1, step=2}">${item.title}</h2>
```

- Identifiers for `data-sly-unwrap`:

```
<div data-sly-unwrap.isUnwrapped="${myCondition || myOtherCondition}">  
  text <span data-sly-test="${isUnwrapped}>is unwrapped</span>  
</div>
```

- Support for negative numbers.

Demo



Core Components

AEM Sites components that cover the most common web content needs:

github.com/adobe/aem-core-wcm-components

Core Components are:

- **Pre-Configurable** Template policies can define how the page authors can use them.
- **Versatile** Authors can create most kind of content with them.
- **Easy to Use** Authors can efficiently create and manage content with them.
- **Production Ready** Useable as-is, because they are robust, well tested and perform well.
- **Accessible** They comply WCAG 2.0 standard, provide ARIA labels and support keyboard navigation.
- **Easy to Style** The components implement the Style System and the markup follows BEM CSS naming.
- **SEO Friendly** The HTML output is semantic and provides schema.org microdata annotations.
- **PWA/SPA/App Ready** rendering. The streamlined JSON output of the Sling Models can also be used for client-side rendering.
- **Extensible** extended. To also cover custom needs but without starting from scratch, everything can be
- **Open Sourced** If something is not as it should, contribute improvements on GitHub (Apache License).
- **Versioned** We won't break your site when improving things that might impact you

The Core Components

Template components

- UPDATE** **Page:** Responsive page that works well with the Template Editor.
- UPDATE** **Navigation:** Site navigation that handles globalized structures.
- 3.** **Language Navigation:** Displays the language structure of a site.
- UPDATE** **Breadcrumb:** Lists the hierarchy of parent pages.
- 5.** **Quick Search:** Incremental search field.

Atomic components

- UPDATE** **Title:** Headings configurable to allow levels 1 to 6.
- UPDATE** **Text:** Plain or rich text with configurable capabilities.
- UPDATE** **Image:** Smart image display with configurable capabilities.
- NEW** **Separator:** Visual spacer between the components

Social components

- 10.** **Sharing:** Facebook and Pinterest widgets.

Referencing components

- 11.** **List:** Lists pages that match the configured criteria.
- NEW** **Teaser:** Visual link to a page teasing its content with an image.
- 13.** **Content Fragment:** Displays reusable content.

Container components

- NEW** **Carousel:** Slideshow presentation of content.
- NEW** **Tabs:** Organizes content into accessible tabs.

Form components

- 16.** **Form Container:** Form paragraph system.
- 17.** **Form Text:** Text input field (text, text area, email, phone, date, ...).
- 18.** **Form Options:** Multi-input field (checkboxes, radios, drop-down, ...).
- 19.** **Form Button:** Submit or scriptable button.
- 20.** **Form Hidden:** Invisible input field, for sending information along.

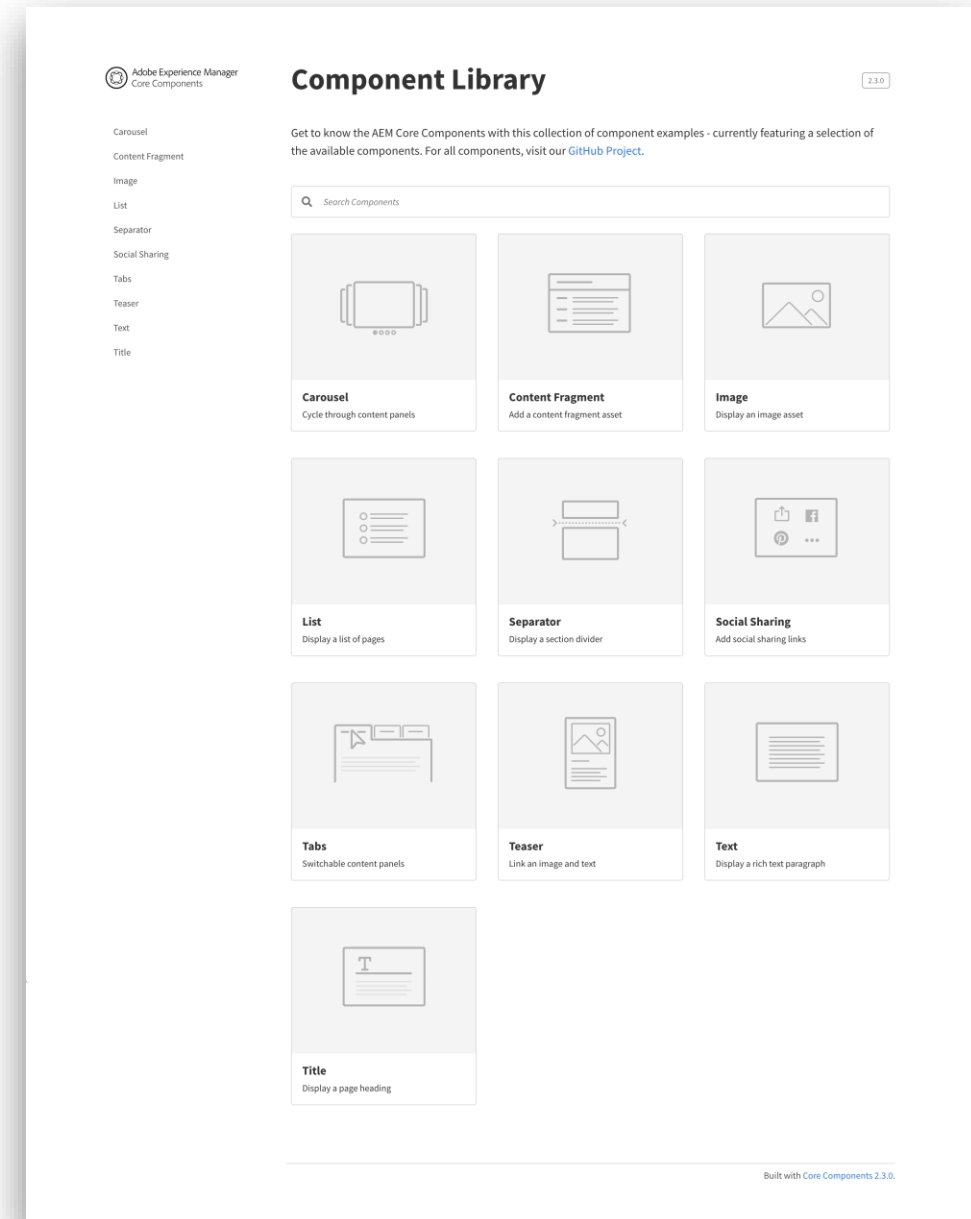
Component Library & Project Archetype

[NEW] The **Component Library** is a reference for front-end developers to style components without customizing their markup, so that implementation effort can be greatly reduced.

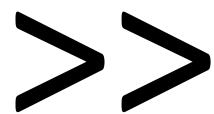
opensource.adobe.com/aem-core-wcm-components/library.html

[UPDATE] The **Project Archetype** is the best way to start a new Sites projects, so that they use the latest and greatest AEM features, like the Core Components, Template Editor, Style System & Responsive Layout, etc.

github.com/adobe/aem-spa-project-archetype

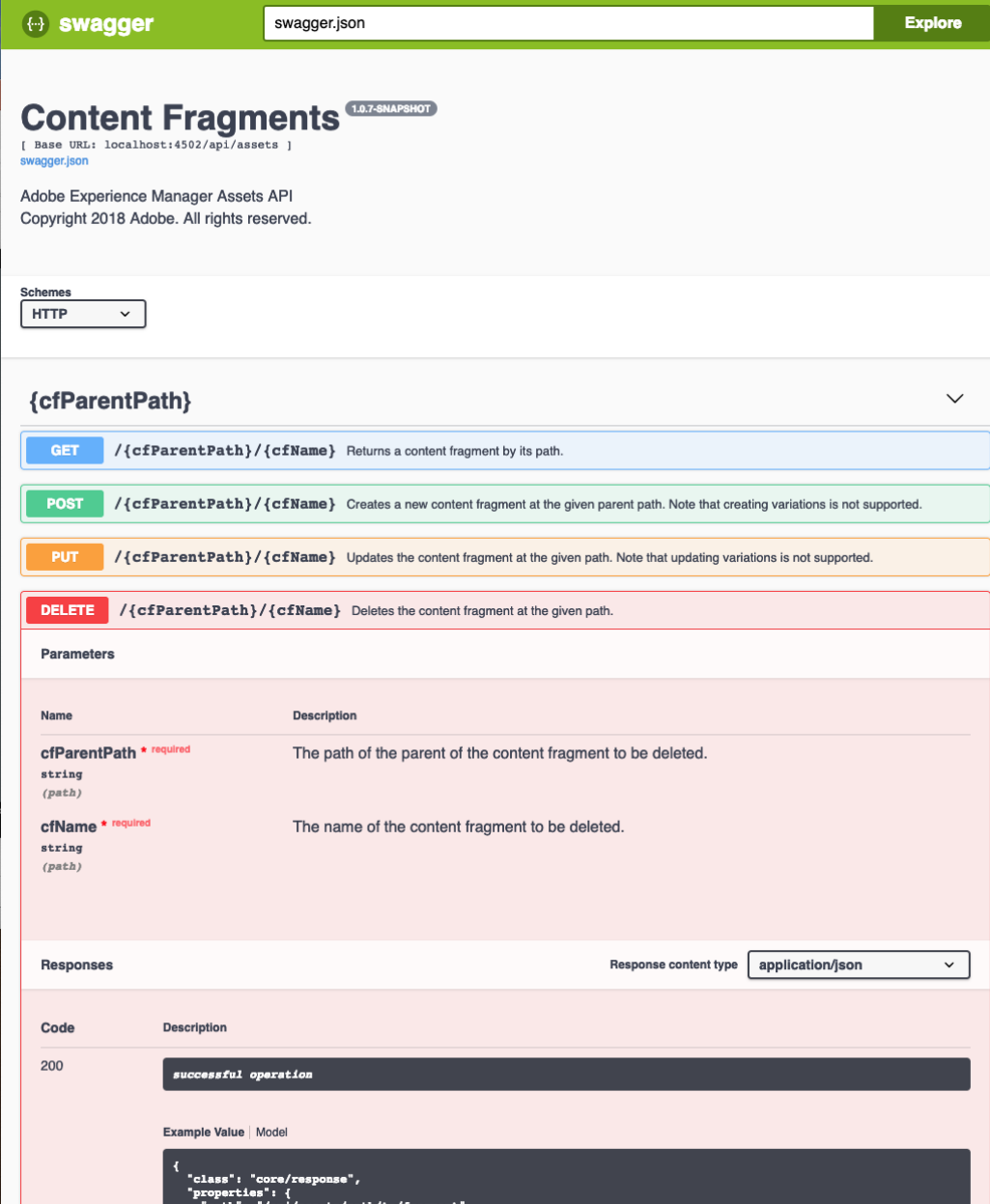


Demo



HTTP APIs – using AEM headless

- Content Fragments: extended Assets HTTP API
 - Create
 - Read
 - Update
 - Delete
- Content
 - more Core Components with default JSON output
 - select JSON as format when exporting Experience Fragment to Adobe Target
- Commerce: extended the CIF and added GraphQL end-point



The image shows a Swagger UI interface for the Content Fragments API. At the top, there is a search bar containing 'swagger.json' and an 'Explore' button. Below this, the title 'Content Fragments' is displayed with a '1.0.7-SNAPSHOT' tag. A small text block indicates the base URL as 'localhost:4502/api/assets' and provides a link to the 'swagger.json' file. The API description reads: 'Adobe Experience Manager Assets API Copyright 2018 Adobe. All rights reserved.' A 'Schemes' dropdown menu is set to 'HTTP'. The main section is titled '{cfParentPath}' and lists four HTTP methods: GET, POST, PUT, and DELETE, each with a description of its function. Below the methods, a 'Parameters' table lists 'cfParentPath' and 'cfName' as required string parameters. The 'Responses' section shows a 200 status code with a description of 'successful operation' and an example JSON response.

swagger.json Explore

Content Fragments 1.0.7-SNAPSHOT

[Base URL: localhost:4502/api/assets]
[swagger.json](#)

Adobe Experience Manager Assets API
Copyright 2018 Adobe. All rights reserved.

Schemes
HTTP

{cfParentPath}

- GET** `/{cfParentPath}/{cfName}` Returns a content fragment by its path.
- POST** `/{cfParentPath}/{cfName}` Creates a new content fragment at the given parent path. Note that creating variations is not supported.
- PUT** `/{cfParentPath}/{cfName}` Updates the content fragment at the given path. Note that updating variations is not supported.
- DELETE** `/{cfParentPath}/{cfName}` Deletes the content fragment at the given path.

Parameters

Name	Description
cfParentPath * required string (path)	The path of the parent of the content fragment to be deleted.
cfName * required string (path)	The name of the content fragment to be deleted.

Responses Response content type: application/json

Code	Description
200	successful operation

Example Value | Model

```
{  
  "class": "core/response",  
  "properties": {  
    "path": "/en/assets/path/en/fragment"  }  
}
```

Demo



JS SDK for React & Angular

AEM Experience Management UI

(SPA Editor & Site Admin)

Practitioner focus

Developer focus

Your Single-Page Experiences

(using the AEM JS SDK)

JSON APIs

JSON APIs

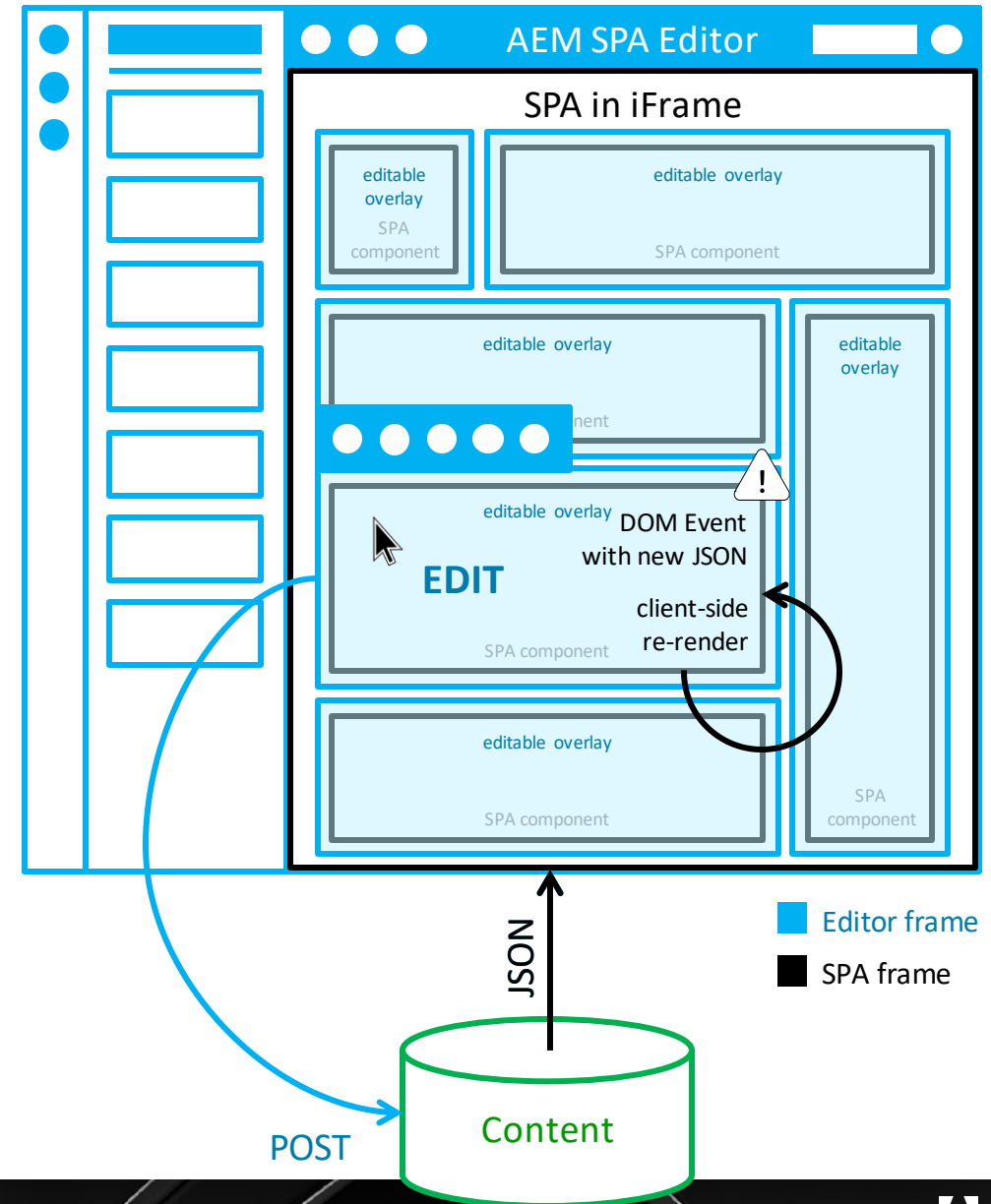
AEM as a
Headless CMS
(Content Services)

Your other
(micro) services

How the SPA Editor Works

1. SPA Editor loads.
2. SPA is loaded in a separated frame.
3. SPA requests JSON content and renders components client-side.
4. SPA Editor detects rendered components and generates overlays.
5. Author clicks overlay, displaying the component's edit toolbar.
6. SPA Editor persists edits with a POST request to the server.
7. SPA Editor requests updated JSON to the SPA Editor, which is sent to the SPA with a DOM Event.
8. SPA re-renders the concerned component, updating its DOM.

- **The SPA is isolated from the editor and always in charge of its display.**
- **In production (publish), the SPA editor is never loaded.**



JS Components with React or Angular

JS components can be implemented AEM agnostic and follow framework-specific best practices.

➔ What is needed is to map them to the AEM resource types via **MapTo()**



```
import React, { Component } from "react";
import { MapTo } from
  "@adobe/cq-react-editable-components";

class MyComponent extends Component {
  render() {
    return (
      <p>
        { this.props.text }
      </p>
    );
  }
}

MapTo("myResourceType")(MyComponent);
```



```
import { Component, Input } from "@angular/core";
import { MapTo } from
  "@adobe/cq-angular-editable-components";

@Component({
  templateUrl: "./my-component.component.html"
})
class MyComponent {
  @Input() text:string;
}

MapTo("myResourceType")(MyComponent);

<!-- my-component.component.html -->
<ng-template>
  <p>{{text}}</p>
</ng-template>
```


Demo



Practitioner UI improvements - Highlights

- Asset Search with Dynamic Facets
- Use assets from central Assets instance in Page Editor
- “Select All” shortcut in list view
- List Inbound Links per page in Reference Rail
- Comments in Content Fragments
- Compare Versions for Content Fragments
- Promote Launch only with approved pages

Demo





Adobe