What are the modules available in AEM?

|  |  |
| --- | --- |
| 1. AEM **Sites** | Manages websites, landing pages, and other digital content. Supports **editable templates**, **content fragments**, and **experience fragments**. Provides **multi-site management (MSM)** for localization and translation. |
| 2. AEM **Assets** | A **digital asset management (DAM) system** for handling images, videos, and documents. Features include **metadata management**, **dynamic media**, and **smart tagging** (AI-powered). |
| 3. AEM **Forms** | Creating and managing **interactive forms** and documents. **PDF Forms, Adaptive Forms**, and **workflows for approvals**. |
| 4. AEM **Screens** | Author  Developer  System Administrator/Technician |
| 5. AEM **Commerce** | Integrates with e-commerce platforms like **Magento, SAP Commerce Cloud, React and others**. |
| 6. AEM **Communities** | Supports **social collaboration and engagement**. |
| 7. AEM **Workflows & Automation** | Provides workflow automation for content approval and publishing. Includes task management and **custom workflow development**. |
| 8. AEM **Cloud Service (Adobe Experience Cloud Integration)** | A cloud-native version of AEM with **auto-scaling, performance optimizations, and CI/CD pipelines**. |

Discuss on AEM Architecture.

**AEM Architecture Overview**

Adobe Experience Manager (AEM) follows a **modular, scalable, and component-based architecture**. It is built on **Apache Sling, JCR (Java Content Repository), and OSGi framework**.

|  |  |
| --- | --- |
| **1. Core Components of AEM Architecture** | |
| Apache Sling (Resource-Based Web Framework) | AEM uses **Apache Sling**, a RESTful framework that maps URLs to content stored in **JCR (Java Content Repository)**. It supports **HTL (Sightly) and JSP** for templating. |
| JCR (Java Content Repository) - Apache Jackrabbit Oak | AEM uses **JCR** to store and manage content hierarchically. Apache Jackrabbit Oak is the **latest implementation** of JCR, improving scalability and performance. Content is stored in **nodes and properties** instead of traditional relational databases. |
| OSGi (Open Service Gateway Initiative) Framework | AEM is **modular**, using OSGi-based **Apache Felix** for dependency management. Allows dynamic component loading and service-oriented architecture. Supports **OSGi bundles** for extending AEM functionality. |
| **2. AEM Deployment Instances** | |
| Author Instance | **content creation, editing, and management**. |
| Publish Instance | final **published content** to end users |
| Dispatcher (Caching & Load Balancer) | **A caching and security layer** in front of the Publish instance. |
| AEM Cloud Service (Optional for Cloud Deployment) | Adobe’s **Cloud-Native AEM solution** with **auto-scaling, CI/CD pipelines, and managed services**. Provides **GraphQL APIs for headless CMS integration**. |
| **3. AEM Workflow & Content Management** | |
| **4. AEM & Front-End (React/Headless Integration)** | |
| **5. AEM Deployment Models** | |

How to create client lib in AEM?

**Create ClientLib Folder**:

1. Navigate to /apps/<your-project>/clientlibs/ in CRXDE Lite.
2. Create a folder (e.g., custom-clientlib).

**Define .content.xml:**

* Add cq:ClientLibraryFolder properties:

|  |
| --- |
| <jcr:root  jcr:primaryType="cq:ClientLibraryFolder"  categories="[custom.clientlib]"  allowProxy="{Boolean}true"/> |

**Add Files:**

1. Create css/ and js/ folders.
2. Add css.txt and js.txt (list file names).
3. Add CSS (clientlib-base.css) and JS (clientlib-base.js) files.

**Include ClientLib in Components:**

In HTL (e.g., component.html), add:

|  |
| --- |
| <sly data-sly-call="${clientlib.css @ categories='custom.clientlib'}" />  <sly data-sly-call="${clientlib.js @ categories='custom.clientlib'}" /> |

**Test:** Open AEM Page Editor to check if the ClientLib is loaded correctly.

What is Adobe Experience Manager (AEM) and its key features?

Adobe Experience Manager (AEM) is a comprehensive **content management solution** for creating and delivering personalized digital experiences across websites, mobile apps, and other channels. It is part of the **Adobe Marketing Cloud**.

**Key Features of AEM:**

1. **Web Content Management (WCM)**: Manage and publish web content with **editable templates**.
2. **Digital Asset Management (DAM)**: Centralized storage for media assets with **AI tagging**.
3. **Personalization**: Deliver targeted, personalized content.
4. **Multi-Site Management (MSM)**: Manage multiple websites and localization.
5. **Headless CMS**: Serve content through APIs to any front-end.
6. **Mobile App Management**: Seamless content delivery to mobile apps.
7. **Form Management**: Create adaptive forms with workflows.
8. **Workflow Automation**: Automate content approvals and publishing.
9. **Analytics Integration**: Real-time insights with **Adobe Analytics**.
10. **Cloud-Native Deployment**: Scalable and secure with **AEM as a Cloud Service**.

What is Sling in the context of AEM?

**Apache Sling** is a **web application framework** used in **Adobe Experience Manager (AEM)** to map **JCR (Java Content Repository) content** to HTTP requests. It follows a **RESTful approach** and simplifies content retrieval and rendering in AEM.

What is the purpose of the Sling Model in AEM?

Sling Models **improve maintainability and performance** in AEM. They reduce boilerplate code, support dependency injection, and integrate seamlessly with HTL for clean, maintainable development.

Write a simple JavaScript code to fetch data from an AEM endpoint using AJAX.

|  |
| --- |
| let url = 'https://your-aem-instance.com/bin/querybuilder.json?path=/content/your-path';  fetch(url)  .then(response => {  if (!response.ok) {  throw new Error('Network response was not ok ' + response.statusText);  }  return response.json();  })  .then(data => {  console.log('Fetched data:', data);  })  .catch(error => {  console.error('There was a problem with the fetch operation:', error);  }); |

|  |
| --- |
| let url = 'https://jsonplaceholder.typicode.com/posts';  fetch(url)  .then(response => {  if (!response.ok) {  throw new Error('Network response was not ok ' + response.statusText);  }  return response.json();  })  .then(data => {  console.log('Fetched posts:', data);  })  .catch(error => {  console.error('There was a problem with the fetch operation:', error);  }); |

Explain the difference between a page and a template in AEM.

**Template**: Structure and design, reusable for multiple pages.

**Page**: Actual content instance created from a template, with specific content for that page.

How do you optimize AEM performance?

**1. Infrastructure**

* **Scale Servers** and use **CDNs** for faster content delivery.
* **Separate Author & Publish** instances for better resource management.

**2. Caching**

* Use **Dispatcher Cache** for static content and **CDNs** for assets.
* Implement **cache invalidation** for fresh content.

**3. Content Delivery**

* **Minify & Bundle** assets (JS, CSS).
* **Lazy load** images and resources for faster page loads.

**4. Component Design**

* **Cache components** and minimize **server API calls**.

**5. JCR Optimization**

* Optimize **queries** and implement proper **indexing**.

**6. Authoring Experience**

* **Limit concurrent editors** and streamline **workflows**.

**7. Monitoring & Maintenance**

* Use **AEM Health Checks** and clean up **unnecessary data** regularly.

**8. Updates -** Keep AEM **up-to-date** for better performance.

When Should You Use EDS Instead of Traditional AEM Sites?  
  
- When your focus is on performance and speed-to-market, not on complex custom workflows.  
- If you’re creating lean, content-driven sites that need to scale globally.  
- When you need to optimize resources without compromising user experience.  
- For teams looking for quick iterations and simplified deployments.