1. What is the purpose of the core module in AEM?

The core module in AEM is where all the backend logic is written. It contains Java classes that handle business logic, Sling models, and OSGi services. These classes interact with AEM’s JCR (Java Content Repository) and provide data to the frontend components. Basically, if a component needs some dynamic content, the logic for fetching or processing that data is written in the core module.

1. What kind of files and code can be found in the core folder?

The core folder mainly contains Java files. These include Sling Models (which help in fetching data from JCR), OSGi services (which handle backend operations), and servlets (which process requests). It also has unit test files and the pom.xml file for managing dependencies. Everything inside this folder is responsible for making components work dynamically.

3. Explain the role of ui.apps in AEM projects.

The ui.apps module is where all the frontend-related stuff is stored. This includes components, templates, dialogs, and client libraries. It's the place where we define how components look and behave in the AEM site. If the core module is the brain, then ui.apps is the face of the application.

4. How are components structured in the ui.apps folder?

Inside ui.apps, components are usually found in ui.apps/src/main/content/jcr\_root/apps/<project>/components/. Each component has its own folder containing an HTL file (.html), a cq:dialog for authoring, a content.xml for properties, and sometimes a JavaScript or CSS file for styling and interactions.

5. Where is the Hello World component located in both core and ui.apps?

In core, the Hello World component’s Java class is in core/src/main/java/<project>/models/.  
In ui.apps, the component is inside ui.apps/src/main/content/jcr\_root/apps/<project>/components/helloworld/.

6. Explain the Java class (in core) for the Hello World component.

The Java class for Hello World is a Sling Model that retrieves some text from JCR and provides it to the frontend. It usually has @Model annotation and uses @Inject to fetch properties like text from the component’s configuration.

7. How does the HTL script work in ui.apps for Hello World?

The HTL (.html file) inside ui.apps simply prints the data coming from the Sling Model. It looks something like:

html

CopyEdit

<sly data-sly-use.model="com.example.models.HelloWorld">

<h1>${model.text}</h1>

</sly>

This pulls the text property from the Java model and displays it in an <h1> tag.

8. How are properties and dialogs defined for this component?

The properties are stored in content.xml, and the dialog is defined in cq:dialog.xml. The dialog lets authors enter values (like text) in AEM’s editor, and these values get stored in JCR, which the Sling Model then fetches.

9. What are the different types of AEM modules (core, ui.apps, ui.content, etc.)?

* core – Java logic (backend).
* ui.apps – Frontend components, templates, client libraries.
* ui.content – Sample content and site structure.
* all – Combines all modules into one deployable package.

10. How does Maven build these modules?

Maven uses the pom.xml in each module to define dependencies and build steps. When we run mvn clean install, it compiles Java code, processes templates, and packages everything into .zip file for AEM.

11.Explain the build lifecycle of Maven in the context of AEM.

Maven follows these main steps in its lifecycle:

1. Validate – Checks if everything is correctly set up.
2. Compile – Compiles Java code.
3. Test – Runs unit tests.
4. Package – Bundles the code into a deployable format (.jar for core, .zip for ui.apps).
5. Install – Stores the built package locally.
6. Deploy – Installs the package into AEM.

12. How are dependencies managed in pom.xml?

Dependencies (like AEM APIs, libraries, etc.) are declared inside the pom.xml file. Maven automatically downloads them from repositories like Adobe’s Maven repo or Maven Central when we build the project.

13. Why is Maven used instead of other build tools?

Maven is preferred because it automates the build process, handles dependencies efficiently, and follows a structured lifecycle. It also integrates well with AEM’s package structure.

14. What advantages does Maven offer for AEM development?

* Standardized Build Process – Consistent across projects.
* Dependency Management – No need to manually download libraries.
* Integration with AEM – Can deploy directly to an AEM instance.

15. How does Maven help in managing dependencies and plugins in AEM projects?

Maven automatically fetches the required dependencies (like AEM APIs) based on what’s specified in pom.xml. It also manages plugins for packaging, testing, and deployment.

16. What does mvn clean install do in an AEM project?

mvn clean install deletes old build files (clean), compiles the code, runs tests, packages everything, and installs it locally (install), making it ready to deploy.

17. How to deploy packages directly to AEM using Maven commands?

Run:

mvn clean install -PautoInstallPackage

This builds and deploys the package to a running AEM instance.

18. Explain the purpose of different Maven profiles in AEM (autoInstallPackage, autoInstallBundle).

* autoInstallPackage – Installs the full AEM package (components, templates, content).
* autoInstallBundle – Installs only the OSGi bundle (Java logic).

19. What is the purpose of dumplibs in AEM?

dumplibs helps debug client libraries by listing all CSS and JS files available in the system. It ensures the right libraries are getting loaded.

20. How can you view client libraries using dumplibs?

Go to:

ruby

http://localhost:4502/libs/granite/ui/content/dumplibs.html

This page shows all available client libraries.

21. Explain how client libraries are structured in AEM.

Client libraries (/etc.clientlibs) group CSS, JS, and dependencies together using categories. Each library has a .content.xml file and a cq:clientLibraryFolder node.