200 INTERVIEW Q&A

MAVEN & NPM

Handbook 2025

DevOps Shack

200 Maven and NPM Interview Questions & Answers

What is Maven, and how does it work?

- Maven is a build automation tool for Java projects, managing dependencies, builds, and plugins using the POM (Project Object Model) file.

2. **What is a POM file?**
   - The POM file is an XML configuration file defining project metadata and dependencies.
3. **Explain Maven's build lifecycle.**
   - Maven has three lifecycles: Clean (clean project), Default (build process), and Site (generate documentation).
4. **What are Maven goals?**
   - Specific tasks Maven executes, e.g., compile, test, or package.
5. **How does Maven differ from Ant?**
   - Maven is declarative with a standard lifecycle, while Ant is procedural and relies on manual configuration.

Dependency Management

6. **What are Maven dependencies?**
   - Libraries or frameworks required for a project to work.
7. **How does Maven resolve dependencies?**
   - It fetches them from local, central, or remote repositories.
8. **What are transitive dependencies?**
   - Dependencies of your project's dependencies.

9. **What is the scope of dependencies in Maven?**

- Scopes like compile, provided, runtime, test, system, and import define their availability.

10. **How do you handle conflicting dependency versions?**
    - Use <dependencyManagement> or exclusions in the POM.

Plugins

11. **What are Maven plugins?**
    - Extensions to execute tasks like compilation or testing.

12. **How do you use the Maven Compiler Plugin?**

```
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-compiler-plugin</artifactId>
<version>3.8.1</version>
<configuration>
<source>11</source>
<target>11</target>
</configuration>
</plugin>
```

13. **What is the Maven Surefire Plugin?**

- Runs unit tests during the test lifecycle phase.

14. **How do you skip tests in Maven?**

- Add -DskipTests=true to the Maven command.

15. **What is the Shade Plugin used for?**

- Creates Uber JARs containing all dependencies.

16. **What are Maven profiles?**

- Environment-specific configurations, e.g., dev or production.

17. **How do you activate a Maven profile?**

- Use `-P` option, e.g., `mvn clean install -Pdev`.

18. **How do you define profiles in a POM file?**

```
<profiles>
  <profile>
    <id>dev</id>
    <properties>
      <env>development</env>
    </properties>
  </profile>
```

</profiles>

19. **Can profiles be activated automatically?**

- Yes, using `<activation>` conditions like JDK version or OS.

20. **How do you integrate Maven with Jenkins?**

- Configure Jenkins to run Maven commands like `mvn clean install`.

21. **What happens if Maven Central is unavailable?**

- Maven uses the local repository or other configured repositories.

22. **How do you resolve dependency conflicts?**

- Use the `dependency:tree` command to analyze the hierarchy.

23. **What is a SNAPSHOT version?**

- Indicates a development version.

24. **How do you deploy artifacts to a remote repository?**

- Use `maven-deploy-plugin` with proper `<distributionManagement>` settings.

25. **What is the Maven Assembly Plugin?**

- Used to create ZIP or TAR distributions of the project.

26. **What is the purpose of `dependencyManagement`?**

- Centralize dependency version definitions in multi-module projects.

27. **What is the purpose of `parent` in Maven?**

- Allows inheritance of configurations and dependencies across modules.

28. **How do you customize the Maven lifecycle?**

- Add or override default plugins in the `<build>` section.

29. **What is the `mvn dependency:analyze` command?**

- Identifies unused or missing dependencies in your project.

30. **How do you improve Maven build performance?**

- Use local mirrors, parallel builds (`-T`), and avoid unnecessary plugins.

31. **How do you handle multiple repositories in Maven?**

```
<repositories>
    <repository>
      <id>custom-repo</id>
      <url>http://example.com/maven2</url>
    </repository>
  </repositories>
```

32. **What is the difference between `provided` and `runtime` scopes?**

- `provided`: Used only during compilation.
- `runtime`: Used during execution.

33. **How do you include resource files in a JAR?**

```
<resources>
    <resource>
      <directory>src/main/resources</directory>
    </resource>
  </resources>
```

34. **How do you manage environment-specific configurations?**

- Use Maven profiles and external property files.

35. **How do you enforce code quality using Maven?**

- Use plugins like `checkstyle` and `spotbugs`.

36. **How do you clean the project?**

- `mvn clean`

37. **How do you compile the project?**

- `mvn compile`

38. **How do you test the project?**

- `mvn test`

39. **How do you package the project?**

- `mvn package`

40. **How do you generate project documentation?**

- `mvn site

41. **What is the difference between install and deploy?**
    - install: Copies the built artifact to the local repository.
    - deploy: Uploads the artifact to a remote repository.
42. **How do you override a dependency version in Maven?**
    - Use <dependencyManagement> to specify the desired version.
43. **How do you define multiple modules in Maven?**
    - Use a parent POM file with <modules> defined:

```
<modules>
  <module>module1</module>
  <module>module2</module>
</modules>
```

44. **How do you run a single test class in Maven?**
    - Use -Dtest=<TestClassName>, e.g., mvn test -Dtest=MyTest.
45. **How do you configure logging in Maven?**
    - Add -X for debug-level logs or -q for quiet mode.
46. **What are the best practices for writing a POM file?**
    - Avoid hardcoding versions; use properties.
    - Use dependency scopes judiciously.
    - Avoid redundant plugins.
47. **What is a reactor build in Maven?**
    - A build involving multiple modules within a project.
48. **How do you use Maven with Spring Boot?**
    - Add the spring-boot-starter-parent as the parent in your POM:

```
<parent>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-parent</artifactId>
<version>2.6.3</version>
</parent>
```

49. **How do you configure a custom artifact name in Maven?**
    - Use <finalName> in the <build> section:

```
<build>
<finalName>custom-artifact</finalName>
</build>
```

50. **What is a BOM (Bill of Materials) in Maven?**
    - A POM file containing dependency versions for consistent management across projects.

Dependency Management

51. **What is a snapshot dependency in Maven?**
    - A dependency under active development.
52. **How do you avoid including unnecessary transitive dependencies?**
    - Use <exclusions> for the specific dependency.

53. **What is the difference between compile and runtime scopes?**
    - compile: Available during both compile and runtime.
    - runtime: Available only during runtime.
54. **How do you specify a system-scoped dependency?**
    - Use <scope>system</scope> with the path specified:

```
<dependency>
<groupId>com.example</groupId>
  <artifactId>example-dependency</artifactId>
  <scope>system</scope>
  <systemPath>${basedir}/lib/example.jar</systemPath>
</dependency>
```

55. **What is the purpose of the mvn dependency:tree command?**
    - Displays the dependency hierarchy

56. **How do you restrict dependency versions?**
    - Use <dependencyManagement> in the parent POM.

Plugins

57. **What is the purpose of the Maven Enforcer Plugin?**
    - Enforces project standards like JDK version or dependency version ranges.
58. **How do you use the Maven Resources Plugin?**
    - It copies and filters resource files during the build process:

```
<plugin>
  <artifactId>maven-resources-plugin</artifactId>
  <version>3.2.0</version>
  <configuration>
    <encoding>UTF-8</encoding>
  </configuration>
</plugin>
```

59. **How do you configure the Maven Release Plugin?**
    - It automates project release preparation and deployment:

```
<plugin>
  <artifactId>maven-release-plugin</artifactId>
  <version>2.5.3</version>
</plugin>
```

60. **What is the Maven JAR Plugin?**
    - Creates JAR files during the build lifecycle.

61. **How do you use the Maven WAR Plugin?**
    - Configures and packages WAR files for web applications:

```
            <plugin>
   <groupId>org.apache.maven.plugins</groupId>
   <artifactId>maven-war-plugin</artifactId>
   <version>3.3.2</version>
</plugin>
```

Profiles and Customization

62. **How do you pass arguments to a Maven build?**
    - Use -D to define arguments, e.g., mvn package -Denv=prod.
63. **How do you enable a profile automatically?**
    - Use <activation> with conditions like property, JDK, or OS:

```
<activation>
<property>
     <name>env</name>
     <value>prod</value>
  </property>
</activation>
```

64. **Can Maven profiles inherit properties?**
    - Yes, profiles can inherit properties from the parent POM.
65. **How do you manage multiple environments in Maven?**
    - Define separate profiles for each environment and activate them with -P.

Testing and Quality

66. **How do you use the Maven Failsafe Plugin?**
    - Runs integration tests during the integration-test phase.
67. **What is the difference between Surefire and Failsafe plugins?**
    - Surefire: Runs unit tests.\n- Failsafe: Runs integration tests.
68. **How do you enforce code style checks?**
    - Use the Checkstyle Plugin:

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-checkstyle-plugin</artifactId>
  <version>3.1.2</version>
</plugin>
```

69. **How do you generate a test report?**
    - Use the maven-surefire-report-plugin:

```
<plugin>
  <artifactId>maven-surefire-report-plugin</artifactId>
  <version>3.0.0-M5</version>
</plugin>
```

Advanced Real-World Scenarios

70. **How do you create a multi-module project in Maven?**
    - Use a parent POM and include modules.

71. **What is the difference between a parent and aggregator POM?**
    - Parent: Used for inheritance.\n- Aggregator: Used to build modules together.
72. **How do you manage logging during Maven builds?**
    - Use -X for debugging or -q for quiet logs.
73. **How do you clean up old SNAPSHOT versions from a repository?**
    - Use a repository manager like Nexus or Artifactory.
74. **How do you deploy Maven artifacts to AWS S3?**
    - Configure the maven-deploy-plugin with S3 details.

Continuous Integration

75. **How do you use Maven in a CI/CD pipeline?**
    - Define Maven commands (e.g., mvn clean install) in pipeline scripts.
76. **How do you integrate SonarQube with Maven?**
    - Use the sonar-maven-plugin to analyze code quality.
77. **How do you deploy a Maven artifact to Nexus?**
    - Use <distributionManagement> in the POM:

```
<distributionManagement>

  <repository>

    <id>nexus-repo</id>
```

```
    <url>http://nexus.example.com/repository/maven-releases</url>
  </repository>
</distributionManagement>
```

78. **How do you build a Docker image using Maven?**
    - Use the docker-maven-plugin to build and push Docker images.

Maven Commands and Utilities

79. **What does mvn validate do?**
    - Validates the project structure and POM file.
80. **How do you list dependencies?**
    - Use mvn dependency:list.
81. **How do you force an update of dependencies?**
    - Use mvn clean install -U.

82. **How do you create a Maven archetype?**

    - Use mvn archetype:create to define a project template.

Miscellaneous

83. **How do you troubleshoot Maven build issues?**
    - Use -X for debug logs and analyze dependency conflicts.
84. **What is the difference between clean install and clean deploy?**
    - install: Copies the artifact locally.\n- deploy: Uploads it to a remote repository.

85. **How do you integrate Maven with Docker?**
    - Use plugins like fabric8-maven-plugin or docker-maven-plugin.

86. **How do you handle large multi-module projects efficiently?**
    - Build specific modules using -pl and enable parallel builds.

87. **How do you test Maven commands locally?**
    - Use mvn verify to simulate the full build lifecycle.

88. **What is the site phase in Maven?**
    - Generates project documentation.

Final Tips

89. **How do you enforce Maven versioning policies?**
    - Use the Enforcer Plugin to define version ranges.
90. **How do you handle platform-specific builds?**
    - Use profiles with <activation> based on OS properties.
91. **How do you generate a POM file programmatically?**
    - Use Maven Archetypes or external tools.
92. **How do you configure third-party plugins?**
    - Add them to the <build> section.
93. **What is a mirror repository in Maven?**
    - A repository that acts as a proxy for Maven Central.
94. **How do you deal with network issues during dependency resolution?**
    - Configure mirrors or offline mode (mvn -o).
95. **How do you create a custom Maven plugin?**
    - Use the maven-plugin-archetype to create a new plugin.
96. **How do you enforce licensing checks in Maven?**

- Use the License Maven Plugin.

97. **How do you handle version conflicts across microservices?**
    - Centralize dependency management in a BOM.
98. **What is the role of Maven Wrapper (mvnw)?**
    - Ensures consistent Maven version across systems.
99. **How do you handle Maven deployment pipelines?**
    - Use CI/CD tools like Jenkins, GitLab CI, or GitHub Actions.
100. **How do you analyze dependency vulnerabilities in Maven?**
    - Use tools like OWASP Dependency-Check or Snyk.

NPM

1. **What is npm, and why is it used?**
    - Default package manager for Node.js.
    - Manages JavaScript packages and dependencies.
    - Used to install libraries, manage project dependencies, and run scripts.
2. **How do you initialize a new npm project?**
    - Run npm init to create a package.json file.
    - Use npm init -y to skip prompts and use default settings.
3. **What is the purpose of package.json?**
    - Contains project metadata (name, version, description).
    - Lists dependencies and devDependencies.
    - Defines custom scripts for automation.
4. **What's the difference between dependencies and devDependencies?**
    - dependencies: Required for application runtime.
    - devDependencies: Required only for development (e.g., testing tools).
5. **How do you install a specific version of a package?**
    - Run npm install <package-name>@<version> (e.g., npm install lodash@4.17.21).

Dependency Management

6. **What happens when you run npm install?**
    - Reads package.json and installs all listed dependencies.
    - Creates or updates node_modules and package-lock.json.

7. **What is package-lock.json?**
    - Records exact versions of installed dependencies.
    - Ensures consistent installations across environments.
8. **How do you update a specific package?**
    - Run npm update <package-name>.
    - Use npm install <package-name>@latest for major version updates.
9. **What's the difference between ^ and ~ in versioning?**
    - ^: Allows updates to minor and patch versions (e.g., ^1.2.3 includes 1.x.x).
    - ~: Allows updates to patch versions only (e.g., ~1.2.3 includes 1.2.x).
10. **How do you remove a package?**
    - Run npm uninstall <package-name>.

Scripts and Automation

11. **What is the scripts section in package.json?**
    - Defines custom npm commands.

Example:

```
"scripts": {
  "start": "node app.js",
  "test": "jest"
}
```

12. **How do you run npm scripts?**
    - Use npm run <script-name> (e.g., npm run start).

13. **How do you pass arguments to an npm script?**
    - Use -- followed by arguments (e.g., npm run script-name -- arg1 arg2).

14. **What is the purpose of npm start?**
    - Runs the start script defined in package.json.

15. **How do you run multiple scripts in parallel?**
    - Use npm-run-all or chain scripts with & (e.g., npm run script1 & npm run script2).

Real-Time Scenarios

16. **How do you handle version conflicts during npm install?**
    - Use npm dedupe to remove duplicate packages.
    - Manually adjust package.json for compatible versions.

17. **What should you do if node_modules grows too large?**
    - Check for unused dependencies with npm prune.
    - Use tools like Webpack to bundle assets.

18. **How do you debug a failing npm install?**
    - Run npm install --verbose for detailed logs.
    - Delete node_modules and package-lock.json, then reinstall.

19. **How can you globally install a package?**
    - Run npm install -g <package-name>.

20. **What is npm audit?**
    - Scans dependencies for security vulnerabilities.
    - Provides suggestions or fixes.

21. **What do you do if npm audit fix breaks the application?**
    - Manually resolve vulnerabilities by:
        - Checking version compatibility.
        - Reviewing changelogs and documentation.

22. **How do you install a package without adding it to package.json?**
    - Use npm install <package-name> --no-save.

23. **How do you clean the npm cache?**
    - Run npm cache clean --force.

24. **How do you test an npm package locally?**
    - Use npm link to create a symlink between the local package and the project.

25. **How do you publish a package to npm?**
    - Run npm login to authenticate.
    - Use npm publish to publish the package.

Advanced Scenarios

26. **How do you restrict access to a private npm package?**
    - Use scoped packages (e.g., @<scope>/<package-name>).
    - Publish to a private registry.

27. **How do you handle peer dependencies?**
    - Manually install the required version.
    - Use npm install --legacy-peer-deps to ignore conflicts

28. **What's the purpose of .npmrc?**
    - Configuration file for npm settings.
    - Used for registries, authentication, and proxies.

29. **How do you run a post-install script?**
    - Add the script under postinstall in package.json.

30. **How do you prevent a package from being updated accidentally?**
    - Use npm shrinkwrap to lock dependency versions.

Troubleshooting

31. **How do you resolve "Module not found" errors?**
    - Verify node_modules and reinstall dependencies.
    - Check import paths.
32. **What do you do if npm installs fail behind a proxy?**

Configure proxy settings in .npmrc:

```
npm config set proxy http://proxy.example.com:8080
```

33. **How do you handle circular dependencies?**
    - Refactor code to remove circular imports.
    - Use tools to identify and break the cycle.

34. **How do you switch between npm registries?**
    - Use npm config set registry <registry-url>.
35. **How do you set up a monorepo with npm?**
    - Use npm workspaces to manage multiple packages in a single repo.

Continuous Integration

36. **How do you cache node_modules in CI/CD pipelines?**

- Use tools like GitHub Actions or Jenkins.
- Cache the node_modules directory or package-lock.json.

37. **How do you ensure all developers use the same npm version?**

Specify the version in package.json:

```
"engines": {
  "npm": ">=7.0.0"
}
```

38. **What's the best way to handle large dependency trees in production?**
    - Use bundlers like Webpack or Rollup.

39. **How do you run npm tasks in Docker?**

Add the following to the Dockerfile:
RUN npm install

40. **What is the purpose of npm ci?**
    - Installs exact versions from package-lock.json.
    - Skips compatibility checks with package.json.

Security and Best Practices

41. **How do you check for outdated packages?**

- Use npm outdated to list outdated dependencies, showing current, wanted, and latest versions.

42. **How do you prevent installing vulnerable dependencies?**

- Use npm audit to identify vulnerabilities.
- Use npm audit fix to automatically resolve them.

43. **What is the purpose of npm shrinkwrap?**
- Creates npm-shrinkwrap.json to lock dependencies for deployment.
- Similar to package-lock.json but intended for publishing.

44. **How do you verify the integrity of a package?**
- Use npm audit or check the integrity field in package-lock.json.

45. **How do you block installation of specific packages?**
- Use .npmrc to block specific packages.
- Example: npm config set save-prefix=false.

46. **What is npx, and how is it different from npm?**
- Runs Node.js binaries directly.
- Does not require global installation of binaries.

47. **How do you set up a private npm registry?**
- Use tools like Verdaccio or Nexus.

48. **How do you ensure your npm package is secure before publishing?**
- Use npm audit, check dependencies, and remove sensitive data from the package directory.

49. **What is the purpose of .npmignore?**
- Prevents specific files or directories from being included when publishing a package.

50. **How do you troubleshoot an npm 403 Forbidden error?**
- Ensure you are logged in with npm login.
- Verify permissions and .npmrc settings.

Custom Package Development

51. **How do you create a custom npm package?**

- Create a project with npm init.
- Add your code.
- Publish using npm publish.

52. **How do you test a custom npm package locally before publishing?**

- Use npm link to link the local package globally for testing.

53. **What is semantic versioning, and why is it important for npm?**

- MAJOR: Breaking changes.
- MINOR: Backward-compatible features.
- PATCH: Bug fixes.

54. **How do you publish a beta version of a package?**

- Use a tag: npm publish --tag beta.

55. **How do you deprecate a package or version?**

- Use npm deprecate <package-name>@<version> "Deprecation message".

56. **How do you unpublish a package?**

- Use npm unpublish <package-name> --force.
- Note: Packages older than 72 hours cannot be unpublished.

57. **How do you enforce specific Node.js or npm versions in a package?**

Specify in package.json under engines:

```
"engines": {
  "node": ">=14",
```

```
  "npm": ">=7"
}
```

58. **How do you handle licensing for an npm package?**

Specify a license in package.json:
"license": "MIT"

59. **How do you automate npm publishing?**
- Use CI/CD pipelines with commands like npm login and npm publish.

60. **How do you version a package automatically?**
    - Use npm version <major|minor|patch>.

Performance Optimization

61. **How do you reduce the size of node_modules?**
- Use tools like modclean.
- Prune unused dependencies with npm prune.

62. **How do you bundle dependencies for production?**
- Use bundlers like Webpack, Rollup, or esbuild.

63. **What is the optionalDependencies field?**
- Specifies packages that are not critical.
- npm continues installation even if they fail.

64. **How do you split dependencies for different environments?**
- Use dependencies for production.
- Use devDependencies for development.

65. **How do you use tree-shaking with npm?**
- Ensure packages support ES module exports.
- Use bundlers like Webpack to eliminate unused code.

66. **What's the purpose of peerDependenciesMeta?**
- Marks peer dependencies as optional.

67. **How do you preload modules to improve performance?**

Use the --require flag:
node --require <module>

68. **How do you handle dependency duplication?**
- Use npm dedupe to flatten the dependency tree.

69. **How do you monitor package performance?**
- Use tools like BundlePhobia to analyze package size and performance.

70. **What is the purpose of npm install --production?**
    - Installs only dependencies, excluding devDependencies.

Team Collaboration

71. **How do you share npm configurations across a team?**
- Use .npmrc or share package.json and package-lock.json.

72. **How do you ensure all developers use the same dependency versions?**
- Commit the package-lock.json file to version control.

73. **What's the best way to handle global dependencies in a team?**
- Avoid global dependencies.

- Use project-specific dependencies.

74. **How do you enforce linting rules with npm?**

Define linting scripts in package.json:
"lint": "eslint ."

75. **How do you document npm package usage?**
- Use README.md with clear installation and usage instructions.

76. **How do you prevent accidental publishing of private packages?**
- Set "private": true in package.json.

77. **How do you enforce security standards across a team?**
- Use tools like npm audit and Snyk in CI pipelines.

78. **How do you prevent unused dependencies in a project?**
- Use tools like depcheck to identify unused dependencies.

79. **How do you test cross-platform compatibility with npm?**
- Use CI pipelines to test on different operating systems.

80. **How do you enforce dependency checks in CI?**
    - Run npm ci and npm audit as part of the build process.

Miscellaneous Scenarios

81. **What is the purpose of the bin field in package.json?**
- Specifies executable files for a package.

82. **How do you resolve EACCES permission errors in npm?**
- Fix file permissions.

- Use a Node Version Manager (e.g., nvm).

83. **How do you handle conflicting package binaries?**
- Use npx to run the correct binary version.

84. **How do you prevent dependency hijacking?**
- Lock versions in package-lock.json.
- Use private registries.

85. **How do you downgrade an npm package version?**
- Use npm install <package-name>@<version>.

86. **What's the difference between npm ls and npm list?**
- Both show installed dependencies.
- They are aliases.

87. **How do you generate documentation for a package?**
- Use tools like JSDoc or Typedoc.

88. **What is the purpose of bundledDependencies?**
- Ensures specific dependencies are bundled with the package.

89. **How do you install packages for multiple environments?**
- Use npm install --only=prod or --only=dev.

90. **How do you prevent typosquatting in npm?**
    - Verify the package source and author.

Advanced Troubleshooting

91. **How do you fix npm ERR! Code ELIFECYCLE?**
- Debug the failing script in package.json.

92. **What do you do if npm install hangs?**
- Clear the cache with npm cache clean --force.
- Delete node_modules and reinstall.

93. **How do you debug npm proxy issues?**
- Configure .npmrc with correct proxy settings.

94. **What is the purpose of npm doctor?**
- Diagnoses common issues with npm.
- Suggests fixes.

95. **How do you fix dependency tree conflicts?**
- Run npm dedupe.
- Manually resolve conflicts in package.json.

96. **How do you recover from a broken node_modules folder?**
- Delete node_modules and package-lock.json.
- Reinstall with npm install.

97. **How do you manage peer dependency conflicts?**
- Install compatible versions.
- Use --legacy-peer-deps.

98. **How do you debug an npm script failure?**
- Add --verbose to the npm command.

99. **How do you identify large dependencies in a project?**
- Use tools like webpack-bundle-analyzer or size-limit.

100. **How do you lock down npm package versions for production?**

- Use npm ci to install exact versions from package-lock.json.