CTS Selenium Interview Questions with Answers Part 4 (2026) | Saravana Tech

100 Questions with Elaborate Answers and Practical Examples

> This guide preserves the supplied question order and repeated/section-heading prompts. Where a prompt refers to a missing example or unspecified project detail, the answer explains the limitation rather than inventing an exact project-specific fact.

HOW TO USE THIS GUIDE

- Personalize project, team-size, tool, client, and experience answers using facts from your own work.
- Use the selectors, URLs, credentials, and expected text configured for the application.
- Use current, compatible Selenium, TestNG, Maven, and Apache POI versions in your project.
- Do not share confidential customer data, passwords, or production connection details.

1. Number of user stories per sprint

There is no fixed number of user stories per sprint. The team selects work based on sprint capacity, story size, historical velocity, dependencies, and the Sprint Goal. A team completing 25 story points does not necessarily need to select the same number of stories every sprint because story sizes differ.

> **EXAMPLE / INTERVIEW TIP**
> Interview answer: “We estimate stories using our team’s agreed method, consider leave and capacity, and select a realistic amount during Sprint Planning. We track progress and discuss scope changes with the Product Owner rather than committing to an arbitrary story count.”

2. Complexity of a user story

Complexity is commonly estimated using story points, relative sizing, planning poker, or t-shirt sizes. Consider functional work, technical uncertainty, dependencies, data setup, integrations, non-functional requirements, and testing effort. Story points represent relative effort/complexity/uncertainty, not exact hours.

> **EXAMPLE / INTERVIEW TIP**
> Example: a simple label change may be 1 point; a new workflow with API integration and multiple validations may be 5 or 8 points, depending on team calibration. The team—not one individual—agrees the estimate.

3. Sprint duration

A sprint is a fixed-length event in Scrum, typically one month or less. Many teams use two-week sprints, but the duration is chosen to suit delivery cadence and feedback needs. The sprint duration should remain consistent enough to support planning and inspection.

> **EXAMPLE / INTERVIEW TIP**
> Explain your actual team’s duration, for example: “Our team works in two-week sprints, with planning at the start, daily Scrum, review, and retrospective near the end.”

4. Agile versus Waterfall

Waterfall follows a more sequential lifecycle, with phases commonly completed in order. Agile delivers in small increments, welcomes feedback, and adapts priorities as the team learns. Waterfall may suit stable, contract-driven requirements; Agile is useful when frequent feedback and evolving requirements matter. Real projects can use hybrid approaches.

> **EXAMPLE / INTERVIEW TIP**
> Testing in Waterfall may be concentrated after implementation; in Agile, QA participates throughout refinement, sprint planning, development, and continuous validation.

5. Defect tracking

Defect tracking records, assigns, prioritizes, investigates, fixes, retests, and closes defects. A defect usually includes summary, steps to reproduce, expected/actual result, environment/build, severity, priority, evidence, owner, and status.

> **EXAMPLE / INTERVIEW TIP**
> Tools include Jira, Azure DevOps, Bugzilla, and other defect systems. A good defect is reproducible, evidence-based, concise, and linked to a requirement or test where possible.

6. Team size

State the actual team size and composition. Explain roles such as Product Owner, Scrum Master, developers, QA/automation engineers, business analyst, DevOps, and shared specialists. Scrum does not prescribe a fixed number for every team; the team should be small enough to coordinate effectively.

> **EXAMPLE / INTERVIEW TIP**
> Example: “Our delivery team has [actual number] people, including developers, QA, and a Scrum Master/lead. QA works with developers and the Product Owner from refinement through release.” Do not invent your team's size.

7. Client interaction

Explain your real interaction model: requirement walkthroughs, sprint demos, defect triage, clarifications, test sign-off, status reporting, and release discussions. Keep communication factual and document decisions in the agreed tools.

> **EXAMPLE / INTERVIEW TIP**
> Example: “I clarify acceptance criteria early, raise risks during sprint execution, demonstrate completed scenarios, and share defect evidence and test status with client stakeholders.” Adjust to your actual responsibilities.

8. What is Selenium?

Selenium is an open-source suite of tools for automating web browsers. Selenium WebDriver controls browsers through browser-specific drivers or supported driver management. It supports languages such as Java, Python, C#, and JavaScript. Selenium primarily automates web applications, not native desktop applications.

> **EXAMPLE / INTERVIEW TIP**
> A Java test can launch Chrome, navigate to a URL, interact with controls, and assert results. Selenium is often combined with TestNG/JUnit, Maven, POM, and CI tools.

9. Store employee names and domains from a web table; select dropdown only for Automation domain

Read the table rows, extract employee name and domain into objects or a Map, and interact with the dropdown only for rows whose domain equals Automation. The exact columns and dropdown locator depend on the supplied HTML.

> **EXAMPLE / INTERVIEW TIP**
>
> Example Java: List<WebElement> rows=driver.findElements(By.cssSelector("table tbody tr")); for(WebElement row:rows){ List<WebElement> cells=row.findElements(By.tagName("td")); String name=cells.get(0).getText().trim(); String domain=cells.get(1).getText().trim(); employeeData.put(name,domain); if(domain.equalsIgnoreCase("Automation")){ row.findElement(By.cssSelector("select")).click(); new Select(row.findElement(By.cssSelector("select"))).selectByIndex(0); } }. Declare employeeData as Map<String,String>; adapt cell indexes and option choice to the actual page.

10. Tell me about your current project

Use a structured 1–2 minute summary: business purpose, users, major workflows, your role, team/process, testing scope, automation stack, integrations, CI/reporting, and one meaningful outcome. Avoid confidential details.

> **EXAMPLE / INTERVIEW TIP**
>
> Template: “The project supports [business purpose]. I work as [role], handling requirement analysis, test planning, manual/automation testing, defect triage, and release reporting. Our stack includes [actual tools]. One challenge I addressed was [true example], by [actions], resulting in [truthful result].”

11. What challenges have you faced?

Choose one genuine challenge and explain it with Situation, Task, Action, Result. Common examples include flaky dynamic locators, unstable test data, slow regression runs, environment outages, intermittent API failures, or unclear acceptance criteria.

> **EXAMPLE / INTERVIEW TIP**
>
> Describe the evidence gathered, root cause, solution, and outcome. Use measurable results only if you can substantiate them.

12. Which framework are you currently working on?

Describe your actual framework and its layers: test/scenario layer, page objects or screen objects, reusable utilities, driver factory, configuration, test data, assertions, logging/reporting, build tool, and CI. If you use Cucumber, explain Feature files → Step Definitions → Page Objects/Utilities → WebDriver, plus hooks and reports.

> **EXAMPLE / INTERVIEW TIP**
>
> Do not claim a framework or feature you have not used. Be ready to show one test from scenario to execution and report.

14. How do you perform a right-click?

Use Selenium Actions.contextClick(). Example: WebElement target=driver.findElement(By.id("record")); new Actions(driver).contextClick(target).perform();. If the menu is a browser-native context menu, WebDriver usually cannot interact with its native entries; if it is an HTML custom menu, locate and click the resulting DOM option.

15. Implicit wait and explicit wait

Implicit wait applies globally to element-location calls. Explicit wait polls for a specific condition such as visibility or clickability. Example implicit: driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(5)); explicit: new WebDriverWait(driver,Duration.ofSeconds(10)).until(ExpectedConditions.elementToBeClickable(By.id("save"))).click();. Avoid mixing implicit and explicit waits because combined timeouts can be confusing.

16. What is FluentWait?

FluentWait lets you define timeout, polling interval, and ignored exception types. WebDriverWait is a WebDriver-focused subclass of FluentWait. It is useful when a page condition needs customized polling or transient exceptions must be ignored deliberately.

> **EXAMPLE / INTERVIEW TIP**
>
> Example: Wait<WebDriver> wait=new FluentWait<>(driver).withTimeout(Duration.ofSeconds(20)).pollingEvery(Duration.ofMillis(500)).ignoring(NoSuchElementException.class); WebElement e=wait.until(d->d.findElement(By.id("status")));. Ignore only exceptions that are expected during polling.

17. Read data from Excel

Apache POI is commonly used to read .xls/.xlsx files. Open a stream, create a Workbook, access a Sheet/Row/Cell, and close resources. Use DataFormatter to preserve displayed values and handle blank/numeric/date/formula cells intentionally.

> **EXAMPLE / INTERVIEW TIP**
>
> Example: try(InputStream in=Files.newInputStream(Path.of("src/test/resources/data.xlsx")); Workbook wb=WorkbookFactory.create(in)){ Sheet s=wb.getSheet("Users"); DataFormatter fmt=new DataFormatter(); String username=fmt.formatCellValue(s.getRow(1).getCell(0)); }. Add the appropriate Apache POI dependency.

18. What is a Firefox profile?

A Firefox profile stores browser settings such as preferences, extensions, cookies, and other user data. Selenium can configure Firefox through FirefoxOptions and, when needed, a FirefoxProfile. Profiles can make tests resemble a configured user environment, but shared profiles may cause test interference.

> **EXAMPLE / INTERVIEW TIP**
>
> Modern Selenium example: FirefoxOptions options=new FirefoxOptions(); options.addPreference("intl.accept_languages","en-US"); WebDriver driver=new FirefoxDriver(options);. Profile APIs vary by Selenium/Firefox version; create isolated profiles for parallel tests.

19. How do you handle a proxy?

Configure browser options or system/network settings with the approved proxy host and port, and ensure the machine can reach it. Selenium provides a Proxy capability that can be supplied to browser Options. Authentication and corporate TLS inspection may need environment-specific setup; never bypass security controls without approval.

> **EXAMPLE / INTERVIEW TIP**
>
> Example: Proxy proxy=new Proxy(); proxy.setHttpProxy("proxy.company.local:8080").setSslProxy("proxy.company.local:8080"); ChromeOptions options=new ChromeOptions(); options.setProxy(proxy); WebDriver driver=new ChromeDriver(options);. Adapt host, scheme, authentication, and bypass rules to your organization.

20. Have you created a framework from scratch?

Answer honestly. If yes, describe requirements, architecture, driver lifecycle, POM, utilities, configuration, data, logging, reporting, build integration, CI, and standards. If you enhanced an existing framework, explain precisely what you designed or implemented.

> **EXAMPLE / INTERVIEW TIP**
>
> A credible walkthrough includes why you selected each design, how tests are executed, how failures are diagnosed, and how parallel runs are isolated.

21. What frameworks have you used other than TestNG and POM?

POM is a design pattern, while TestNG is a test runner. Other approaches/tools can include Cucumber BDD, data-driven, keyword-driven, hybrid frameworks, JUnit, Screenplay, or modular frameworks. Name only what you have actually used and explain how it differs.

> **EXAMPLE / INTERVIEW TIP**
>
> If using Cucumber, explain Gherkin feature files, step definitions, hooks, tags, and runner configuration; POM can still be used underneath.

22. Explain inheritance, extends, and using another class through an object

Inheritance lets a subclass reuse and specialize accessible members of a superclass; extends declares class inheritance. A class can also use another class by composition—creating/receiving an object and calling its public methods—without inheriting from it.

> **EXAMPLE / INTERVIEW TIP**
>
> Example: class BasePage { void waitForPage(){} } class LoginPage extends BasePage { void login(){waitForPage();} }. Composition: LoginPage holds Helper helper; then helper.click(...). Use inheritance for a genuine “is-a” relationship and composition for “has-a” reuse.

23. How do you find broken links?

Collect anchor href values, validate non-empty HTTP/HTTPS URLs, and send HTTP requests to check status. Some sites block automated HEAD requests, so retry with GET when appropriate. Avoid treating mailto:, javascript:, fragments, or authenticated links as ordinary web URLs.

> **EXAMPLE / INTERVIEW TIP**
>
> Example outline: for each a[href], resolve absolute URL; use Java HttpClient with a timeout; issue HEAD; if unsupported, retry GET; report status >=400 or connection failures. Use a polite rate limit, do not crawl beyond scope, and handle redirects/timeouts. Browser navigation alone is inefficient for many links.

24. How do you count all links on a page?

Use findElements(By.tagName("a")) and read the list size. Example: List<WebElement> links=driver.findElements(By.tagName("a")); System.out.println("Link count: "+links.size());. If you mean links with href only, use By.cssSelector("a[href]").

25. An element was removed in the previous release; how do you verify it is absent?

Use findElements and assert the result is empty, or wait for invisibility if the element might still be transitioning out of the DOM. Example: Assert.assertTrue(driver.findElements(By.id("oldElement")).isEmpty(), "Removed element is still present");. Presence and visibility are different: an element can exist in the DOM but be hidden.

26. Write XPath using following

The following axis selects nodes that occur after the context node in document order, excluding descendants. Example: //label[normalize-space()='Email']/following::input[1] finds the first input after the Email label. For a sibling in the same parent, following-sibling is usually more precise: //label[.='Email']/following-sibling::input[1]. Use the supplied DOM to choose correctly.

27. F2F questions

This is a section label. Prepare to explain your role, project, automation design, Agile workflow, defects, Java, Selenium, TestNG, SQL, Git, and debugging with practical examples.

28. Where are you currently working?

State your current employer and role truthfully, then briefly describe responsibilities relevant to the vacancy. Do not disclose confidential client information unless authorized.

29. Tell me about yourself

Give a 60–90 second summary: experience, current role, manual/automation skills, key tools, leadership responsibilities, and why this role fits. Keep details aligned with your resume.

30. How did you move from manual to automation?

Explain the progression: strong manual testing fundamentals, identifying repetitive/high-value cases, learning Java and Selenium, building small scripts, learning framework patterns, reviewing code, and integrating automation into regression/CI. Share a real example of a manual scenario you automated and how you maintained it.

31. Which methodology does your project follow?

State whether it uses Scrum, Kanban, Waterfall, or a hybrid approach. Explain the actual ceremonies/artifacts and how QA fits into the lifecycle. Do not call a project Agile merely because it has daily meetings.

32. What is Agile methodology?

Agile delivers software incrementally with collaboration, feedback, and adaptation. Scrum is one Agile framework; its events include Sprint Planning, Daily Scrum, Sprint Review, and Sprint Retrospective. The Product Backlog, Sprint Backlog, and Increment are Scrum artifacts.

33. What are the phases of STLC?

Common phases are requirement analysis, test planning, test case/test data development, test environment setup, test execution, defect reporting/retesting/regression, and test closure. In Agile, these activities often overlap rather than occur as a strictly linear sequence.

34. Which STLC phases do you actually work on?

Answer based on your responsibilities. A QA automation lead may participate in requirement analysis, estimation/test planning, test design, data/environment preparation, automation development, execution, defect triage, regression, reporting, and closure. Give one example for each phase you own.

35. Do you contribute to test planning?

If true, explain scope analysis, risk-based coverage, estimates, resource/capacity planning, environment/data needs, automation feasibility, entry/exit criteria, dependencies, and reporting. If your role is limited, be clear about what you contribute and who approves the plan.

36. Tell me about your organisation

Give a brief factual overview: company industry, service/product focus, your business unit, and your role. Avoid unsupported claims, confidential revenue/client information, or an overly long corporate history.

37. Explain your designation

State your official designation and explain actual responsibilities, not just the title. Examples may include test strategy, team coordination, automation reviews, defect triage, stakeholder communication, delivery tracking, and release sign-off support.

38. Which defect-management tool do you use?

Name the tool actually used, such as Jira, Azure DevOps, or another tracker. Explain how defects are logged, linked to requirements/test cases, assigned, prioritized, tracked, retested, and closed.

39. What are defect statuses?

Statuses vary by organization. A common workflow is New/Open → Assigned → In Progress → Fixed/Resolved → Ready for Retest → Closed. Other statuses can include Reopened, Rejected/Invalid, Duplicate, Deferred, or Cannot Reproduce. Follow your team's configured workflow.

40. What status do you set when assigning a defect to a developer?

Use the status configured in your workflow, commonly Assigned or Open with an assignee set to the developer. Some teams assign ownership without changing status. Explain the actual workflow rather than assuming all tools use the same status names.

41. If a defect is not fixed, who can move it to the next iteration?

The Product Owner typically prioritizes backlog items and helps decide what is included in a future sprint/release, in collaboration with the team and stakeholders. QA reports risk and impact; the team does not silently ignore a defect. Severity, release risk, and governance determine escalation.

42. If a defect is deferred, which status is used?

Common labels include Deferred, Postponed, or Open with a target release/sprint, depending on the defect workflow. Record the reason, risk, owner, and agreed target. Do not mark a defect Closed if it is not fixed unless the process explicitly defines a different disposition.

43. How do you get requirements?

Sources may include Product Owner/business analyst walkthroughs, user stories, acceptance criteria, business requirement documents, designs, API contracts, refinement sessions, and client discussions. Record questions and confirm decisions in the agreed tool.

44. How do you analyze requirements?

Check business rules, user roles, inputs/outputs, positive/negative paths, boundary conditions, validations, integrations, security, performance, accessibility, test data, dependencies, and acceptance criteria. Identify ambiguities early and derive traceable test scenarios.

45. Rate yourself in Java

Give an honest rating and support it with examples of what you can do independently: OOP, collections, exceptions, strings, loops, test utilities, and debugging. Mention an area you are improving. Do not inflate your rating.

46. What are Java features?

Common features include object-oriented design, platform independence through bytecode/JVM, portability, robustness, security features, multithreading, garbage collection, rich standard libraries, exception handling, and dynamic class loading. Avoid claiming Java is completely platform-independent in every operational detail; the JVM/runtime still matters.

47. Which Java features do you use in Selenium?

Examples: classes/objects for page objects, interfaces for WebDriver, inheritance for shared page/base classes, encapsulation for private locators, polymorphism through WebDriver implementations, collections for elements/data, exceptions for failure handling, strings/regex for parsing, and lambdas for wait conditions or streams.

48. Explain inheritance

Inheritance lets a child class reuse and specialize accessible fields/methods from a parent class. Java uses extends for class inheritance and implements for interfaces. Constructors are not inherited, and private members are not directly accessible in subclasses.

49. Explain encapsulation

Encapsulation bundles data and behavior and restricts direct access to internal state, commonly using private fields and public methods. It enables validation and protects invariants. Example: private balance with deposit() and getBalance() methods instead of allowing arbitrary direct changes.

50. Explain overloading and overriding

Overloading uses the same method name with different parameter lists and is resolved at compile time. Overriding provides a subclass implementation of an inherited instance method and is resolved at runtime. Return type alone cannot distinguish overloaded methods.

51. Explain polymorphism

Polymorphism means one interface/type can represent different concrete implementations. Compile-time polymorphism is commonly illustrated by method overloading; runtime polymorphism by method overriding and dynamic dispatch.

> **EXAMPLE / INTERVIEW TIP**
>
> Example: WebDriver driver=new ChromeDriver(); the reference is WebDriver while the runtime implementation is ChromeDriver. Calling driver methods uses the actual implementation.

52. Explain abstract class

An abstract class cannot be instantiated directly and may include abstract methods, concrete methods, state, and constructors. Subclasses implement abstract methods unless they remain abstract. Use it when related classes share state or implementation.

53. Explain interface

An interface defines a contract. A class can implement multiple interfaces, supporting loose coupling and multiple inheritance of type. Modern Java interfaces can include abstract, default, and static methods, plus private helper methods.

54. Rate yourself in Selenium

Rate yourself honestly and explain your practical strengths: locators, waits, frames/windows, dropdowns, actions, file upload, test isolation, debugging, and framework integration. Mention areas still being developed.

55. What are TestNG annotations?

Common annotations include @BeforeSuite/@AfterSuite, @BeforeTest/@AfterTest, @BeforeClass/@AfterClass, @BeforeMethod/@AfterMethod, @Test, @BeforeGroups/@AfterGroups, @DataProvider, @Parameters, @Factory, and @Listeners. Explain lifecycle annotations in order and note that execution can vary with groups, dependencies, inheritance, and parallelism.

56. Explain your framework

Describe architecture, test flow, page objects, utilities, driver factory, configuration, test data, assertions, reporting, logging, Maven, TestNG/Cucumber, and Jenkins. Walk through one real scenario from test definition to report and explain how failures are diagnosed.

57. What is a hybrid framework?

A hybrid framework combines approaches such as POM, data-driven testing, keyword-driven utilities, and reusable components. Explain the exact combination in your project and how it improves reuse, readability, and test-data coverage.

58. What reports do you use to verify results?

Name your actual reporting tools. TestNG/Surefire can generate XML/text results; ExtentReports or Allure may produce richer HTML reports if configured. Reports should include pass/fail/skip, errors, timing, environment, and useful screenshots/logs for failures.

59. How do you read Excel?

Use Apache POI with FileInputStream/Files.newInputStream, WorkbookFactory or XSSFWorkbook, and Sheet/Row/Cell. Close resources with try-with-resources. Use DataFormatter when displayed values matter and handle empty cells and formulas.

60. What repository do you use for files and folders?

Clarify whether the interviewer means source control or artifact storage. Git repositories store versioned source code; artifact/file repositories such as Nexus or Artifactory can store build artifacts and packages. Teams may also use shared drives or cloud storage for approved files. State your actual tool.

61. If you use Git, how do you add files to the cloud repository?

Typical workflow: git status; git add path/to/file; git commit -m "Add automation update"; git push origin branch-name. First ensure the remote is configured and you are on the intended branch. A pull/rebase may be required before pushing if the remote has new commits.

62. Why might correct code not be added to Git?

Possible causes: file not staged, ignored by .gitignore, wrong repository/branch, untracked file, merge conflict, failed commit, push rejected, authentication/permission error, or file outside the repository. Diagnose with git status, git check-ignore -v filename, git branch --show-current, git remote -v, and push output.

63. When do you use Selenium Grid?

Use Grid when you need remote browser execution, parallelism, or coverage across browser/OS combinations. For small local runs it may add unnecessary complexity. Isolate test data and use a compatible browser capability configuration.

64. Who is the Hub in your project?

In legacy Selenium Grid, the Hub is the central coordinator that receives session requests and routes them to registered Nodes. In Selenium Grid 4 the architecture is more distributed, with Router, Distributor, Session Map, Event Bus, and Nodes; standalone mode packages components together. Describe the version and topology actually used.

65. What is the use of Grid?

Grid supports remote and parallel test execution across browser/OS configurations, reducing total suite time and expanding compatibility coverage. It also centralizes browser session allocation. Benefits depend on available capacity and test isolation.

66. When do you run regression testing?

Run regression after code changes that may affect existing functionality, bug fixes, integrations, configuration changes, and before releases. Select scope by risk and impact. In CI, run a fast smoke suite frequently and broader regression at suitable milestones/nightly.

67. Verification versus validation

Verification asks whether the product is being built according to specifications—reviews, inspections, and static checks. Validation asks whether the product meets user needs—executing tests and evaluating real behavior. A concise phrase is “building the product right” versus “building the right product.”

68. How do you verify a page title?

Use driver.getTitle() and compare it with the expected value. Example: String actual=driver.getTitle(); Assert.assertEquals(actual,"Dashboard - Portal","Unexpected page title");. Wait for navigation first if the title changes asynchronously.

69. Difference between assert and verify

An assertion commonly stops the current test at the first failed check by throwing an assertion error. “Verify” often means a soft assertion that records failures and continues, though exact semantics depend on the framework. In TestNG, Assert is hard assertion; SoftAssert collects failures until assertAll() is called.

70. Rate yourself in SQL

Give a realistic rating and name the operations you can perform: SELECT, WHERE, JOINs, GROUP BY, aggregates, subqueries, INSERT/UPDATE, and data validation. Be clear about whether you have used Oracle or another database hands-on.

71. How do you connect to a database?

Use JDBC: add the database driver dependency, connect with a JDBC URL and credentials, execute a PreparedStatement, read ResultSet, and close resources. Use try-with-resources, parameterized queries, least-privilege access, and a secure secret store.

72. Write a query to fetch data

Example: SELECT employee_id, employee_name FROM employees WHERE department_id = 10 ORDER BY employee_name;. Adapt table/column names to the schema. Use a WHERE clause to limit the result and avoid SELECT * when only specific columns are needed.

73. Verify title, select third dropdown option, submit, accept alert

Illustrative Java Selenium code: String title=driver.getTitle(); Assert.assertEquals(title,"Expected Title"); WebElement dropdown=driver.findElement(By.id("choice")); new Select(dropdown).selectByIndex(2); driver.findElement(By.id("submit")).click(); Alert alert=new WebDriverWait(driver,Duration.ofSeconds(10)).until(ExpectedConditions.alertIsPresent()); alert.accept();. Index 2 is the third option because indexing starts at 0. Use the IDs and expected title defined by the application.

74. How do you locate an element with no attributes?

Use stable text or its relationship to a uniquely identifiable neighboring element/container. Examples: //button[normalize-space()='Continue']; //label[.='Email']/following::input[1]; or locate a unique parent then use .//span[normalize-space()='Ready']. If neither attributes, text, nor stable structure exists, request a test ID or improve accessibility/markup.

75. Types of XPath

Interviewers commonly mean absolute XPath and relative XPath. Absolute starts from the root, such as /html/body/div/...; relative searches from matching nodes, commonly //input[@id='email']. XPath also supports axes (parent, child, ancestor, descendant, following-sibling, preceding-sibling, following, preceding, self) and functions such as contains(), starts-with(), and normalize-space().

76. Which locator is faster?

In general, a stable ID is a good first choice and is typically efficient; CSS selectors are also commonly efficient. XPath can be equally practical in modern browsers, and actual performance differences are usually less important than stability and readability. Choose a unique, robust locator and measure only if performance is a real concern.

77. Which is faster: absolute or relative XPath?

There is no reliable universal rule that one is always faster. Browser implementation and DOM complexity matter, while locator robustness is usually more important. Relative XPath is typically easier to maintain and less brittle than an absolute path.

78. Disadvantages of absolute XPath

Absolute XPath is tied to the exact DOM hierarchy, so adding a wrapper or changing layout can break it. It is hard to read and maintain and often depends on positional indexes. Prefer a stable ID, CSS selector, or scoped relative XPath.

79. What is Maven?

Maven is a build and dependency-management tool for Java projects. It uses pom.xml to define dependencies, plugins, project coordinates, properties, and build configuration. It supports a standard lifecycle and CI integration.

80. Explain Maven lifecycle

The main default lifecycle phases include validate, compile, test, package, verify, install, and deploy. Running a later phase runs the earlier phases in that lifecycle first. clean is a separate lifecycle that removes build outputs; site generates project documentation. Plugins bind goals to phases.

81. Methods available for JavaScript alerts

Selenium's Alert interface provides accept(), dismiss(), getText(), and sendKeys() (sendKeys is for prompts that accept input). Example: Alert a=driver.switchTo().alert(); String msg=a.getText(); a.accept();. Wait for alert presence before switching.

82. How do you handle a Yes/No alert?

If it is a JavaScript alert/confirm, Selenium can usually accept or dismiss it; the buttons are browser-defined. If it is an HTML modal with Yes and No buttons, locate and click the corresponding DOM element. Confirm which kind of popup the question describes.

83. How do you interact with a web table?

Locate rows, then cells within each row. Use row/cell text to identify the target record rather than relying only on row indexes. Example: for each tr, read td cells; if the employee cell matches a name, click the action button within that row. Handle pagination and dynamic tables explicitly.

84. A two-month calendar shows July and August; date 4 is enabled. How do you select it?

Scope the date locator to the intended month panel; otherwise two calendars may contain a day numbered 4. Example: locate the calendar header whose text is “August”, find its day-cell button with text 4, verify it is enabled, then click it. Avoid global //button[text()='4'] if both months show the same day number. Adapt selectors to the calendar DOM.

85. What wait methods are used in Selenium?

Common approaches are implicit wait, explicit wait (WebDriverWait), and FluentWait (custom timeout/polling/ignored exceptions). Thread.sleep is a fixed Java pause, not a condition-based Selenium wait, and should generally be avoided for synchronization.

86. FluentWait syntax and an example

Example: Wait<WebDriver> wait=new FluentWait<> (driver).withTimeout(Duration.ofSeconds(15)).pollingEvery(Duration.ofMillis(300)).ignoring(NoSuchElementException.class); WebElement status=wait.until(d->{WebElement e=d.findElement(By.id("status")); return e.isDisplayed()?e:null;});. Use it when the condition may become true after repeated polling; do not ignore unrelated exceptions.

87. Parallel execution through TestNG

TestNG supports parallel execution via testng.xml attributes such as parallel="methods", "classes", or "tests" and thread-count. A DataProvider can also use parallel=true. Ensure each test/thread has its own WebDriver instance, avoid shared mutable state, isolate data, and make reports thread-safe.

88. Explain your project framework if it uses Cucumber

Explain Feature files written in Gherkin, step definitions binding steps to Java code, hooks for setup/teardown, page objects/utilities for UI actions, runner configuration, tags, test data, reporting, Maven, and CI. Walk through a feature scenario to show how a step invokes a page object and how results are reported.

89. Write XPath code for textbox, combo box, and input inside a web table

Use the table row as context, identify it by a unique cell, and then locate controls inside that row. Example: WebElement row=driver.findElement(By.xpath("//tr[td[normalize-space()='Employee A']]")); WebElement textBox=row.findElement(By.xpath(".//input[@type='text']")); WebElement combo=row.findElement(By.xpath(".//select")); textBox.sendKeys("QA"); new Select(combo).selectByVisibleText("Automation");. Adapt selectors to the actual HTML.

90. Roles and responsibilities in automation

Common responsibilities include analyzing requirements, choosing automation candidates, designing/maintaining framework components, writing/reviewing scripts, managing test data, executing tests, triaging failures, reporting metrics, integrating CI, collaborating with developers/BA/product owners, and mentoring/reviewing code. Tailor to your actual role.

91. What is a data-driven framework?

It separates test logic from test data and executes the same scenario with multiple input sets. Data may come from TestNG DataProvider, Excel, CSV, JSON, or a database. Validate data, avoid hardcoded secrets, and ensure unique data and cleanup.

92. Explain Page Object Model

POM represents each page/component with a class that encapsulates locators and actions. Test classes focus on business flows and assertions. Benefits include reuse, readable tests, and centralized maintenance. POM can coexist with Cucumber and data-driven testing.

93. What is NullPointerException?

NullPointerException occurs when code dereferences a null reference, such as calling a method on null. Find where the value should have been initialized, validate method returns, and use clear preconditions. Example: String name=null; name.length(); throws NullPointerException. Avoid masking the problem with broad catch blocks.

94. Which wait methods does WebDriver support?

WebDriver exposes timeouts for implicit wait, page-load timeout, and script timeout through manage().timeouts(). Explicit/Fluent waits are classes from Selenium support APIs rather than WebDriver timeout settings. Example: driver.manage().timeouts().pageLoadTimeout(Duration.ofSeconds(30));.

95. FluentWait and its syntax

FluentWait configures maximum timeout, polling interval, and exceptions to ignore. Example: new FluentWait<> (driver).withTimeout(Duration.ofSeconds(10)).pollingEvery(Duration.ofMillis(500)).ignoring(NoSuchElementException.class).until(d -> d.findElement(By.id("ready")).isDisplayed());. WebDriverWait is a convenient specialization.

96. What is a default class / abstract class?

Java does not have a general “default class” keyword. A class with no access modifier has package-private access. If the interviewer means a default method, interfaces can provide default method implementations. An abstract class cannot be instantiated and can mix abstract and concrete methods. Ask which meaning is intended.

97. Database connection

Use JDBC and a suitable database driver. Obtain a Connection with DriverManager or a connection pool, use PreparedStatement for parameterized SQL, read results through ResultSet, and close resources. Keep credentials outside code and use least-privilege accounts.

98. Git commands to push code

Typical sequence: git status; git add . (or a specific path); git commit -m "Describe change"; git push origin feature-branch. Before pushing, confirm branch and remote. If rejected, fetch/pull and reconcile changes safely; do not force-push shared branches unless the team explicitly permits it.

99. ArrayList versus List with code

List is an interface; ArrayList is one implementation. Declare to the interface for flexibility: List<String> names=new ArrayList<>(); names.add("Asha"); names.add("Ravi"); System.out.println(names.get(0));. ArrayList uses a resizable array and offers O(1) average indexed access; List itself cannot be instantiated.

100. Handling dynamic web elements with XPath examples

Use stable attributes/text, scope locators to a unique parent, and wait for the right state. Examples: //button[normalize-space()='Submit']; //div[contains(@class,'result') and .//span[normalize-space()='Approved']]; //label[normalize-space()='Email']/following::input[1]; //tr[td[normalize-space()='Asha']]/td[3]; and //button[contains(@data-testid,'save')]. For changing IDs, use a stable prefix only if the pattern is guaranteed. Pair dynamic locators with explicit waits and re-locate after DOM refreshes.