Indium Software Selenium Interview Questions with Answers (2026) | Saravana Tech

13 common interview questions with explanations, sample answers, and code

Answers about Selenium and browser versions should reflect your project history. Use versions declared in your project pom.xml, browser About screen, and CI environment.

1. What version of Selenium did you start with?

This is a question about your own experience, so do not guess. Give the Selenium version you actually used in your first automation project, then describe how you upgraded.

Sample answer to personalize: “I started with Selenium [2.x/3.x/4.x] in my early automation work. At that time, we used it for browser automation such as locating elements, entering data, clicking buttons, and validating results. As the framework evolved, I worked with Selenium [your current project version]. I check compatibility among Selenium, Java, browser versions, and the driver setup before upgrading.”

If you do not remember the exact version, say: “I don’t want to give an inaccurate patch number. I started with Selenium [major version, if known], and I can confirm the exact version from the old project’s dependency file.”

Where to check: Maven pom.xml for org.seleniumhq.selenium:selenium-java; Gradle build.gradle for the Selenium dependency; old source-control tags or build logs for historical versions.

2. What current versions of Selenium, Firefox, and Chrome are you using?

Answer with the versions in your actual project, not simply the newest versions available online. Browser releases change frequently, and a project may deliberately pin an earlier stable version.

Sample answer: “Our project currently uses Selenium [x.y.z], Java [version], Chrome [version], and Firefox [version]. We run the supported browser versions in our test environment and validate compatibility in CI. Before upgrading, we check release notes, driver management, browser policies, and the impact on existing tests.”

How to verify:

```
// Selenium version: inspect pom.xml
<dependency>
  <groupId>org.seleniumhq.selenium</groupId>
  <artifactId>selenium-java</artifactId>
  <version>YOUR_PROJECT_VERSION</version>
</dependency>

// In Chrome: open chrome://settings/help
// In Firefox: open about:support and check Application Basics
// In Maven: mvn dependency:tree
```

Selenium 4 uses the W3C WebDriver standard and includes Selenium Manager in modern releases to help manage browser drivers. Do not claim a particular current browser version unless you have checked the machine or CI agent on the day of the interview.

3. How do you sort an array of elements?

For a Java array, use Arrays.sort(). For strings, the default sort is lexicographical (dictionary-like by character values). For numeric arrays, it sorts ascending. If the values are web elements, first collect their displayed text, sort the text, and compare it with the original order.

```
import java.util.Arrays;

public class ArraySortExample {
    public static void main(String[] args) {
        int[] numbers = {40, 10, 30, 20};
        Arrays.sort(numbers);
        System.out.println(Arrays.toString(numbers)); // [10, 20, 30, 40]

        String[] names = {"Ravi", "Anu", "Kumar"};
        Arrays.sort(names, String.CASE_INSENSITIVE_ORDER);
        System.out.println(Arrays.toString(names)); // [Anu, Kumar, Ravi]
    }
}
```

Interview point: Arrays.sort() changes the original array. If you need to preserve the original order, copy the array first.

4. How do you check whether dropdown elements are sorted in ascending order?

Get all options, read their visible text, create a sorted copy, and compare the sorted copy with the original list. Do not sort the original list before checking, because that would hide whether the UI was already sorted.

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import org.openqa.selenium.By;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.ui.Select;

WebElement dropdown = driver.findElement(By.id("country"));
Select select = new Select(dropdown);

List<String> actual = new ArrayList<>();
for (WebElement option : select.getOptions()) {
    actual.add(option.getText().trim());
}

List<String> expected = new ArrayList<>(actual);
expected.sort(String.CASE_INSENSITIVE_ORDER);

if (actual.equals(expected)) {
    System.out.println("Dropdown is sorted in ascending order");
} else {
    System.out.println("Dropdown is NOT sorted in ascending order");
}
```

For a custom Angular/React dropdown that is not a native <select>, locate the visible option elements and apply the same text-list comparison. Decide how to treat placeholders such as “Select” and blank options; usually exclude them if they are not part of the sortable data.

5. How do you count rows and columns in a web table?

Use the table's rows and cells. Usually, count rows with tr and columns using the first row's th/td cells. If the table has multiple header rows, merged cells (colspan/rowspan), or nested tables, simple counts may need adjustment.

```
WebElement table = driver.findElement(By.id("employeeTable"));

List<WebElement> rows =
    table.findElements(By.xpath(".//tr"));
System.out.println("Rows: " + rows.size());

if (!rows.isEmpty()) {
    List<WebElement> cells =
        rows.get(0).findElements(By.xpath("./th | ./td"));
    System.out.println("Columns in first row: " + cells.size());
}
```

If you want only body rows, locate tbody tr. If you need the number of data columns, inspect a representative data row rather than assuming the first row contains only data cells.

6. What Selenium exceptions have you encountered?

Mention exceptions you have actually encountered and explain the cause and resolution. Common examples:

- NoSuchElementException — locator did not find an element; verify locator, page state, and whether the element is inside a frame.
- StaleElementReferenceException — the DOM changed after the element was located; locate it again after the update.
- TimeoutException — an explicit wait condition was not satisfied within the timeout; check the expected condition and application response.
- ElementClickInterceptedException — another element, overlay, or loader intercepted the click; wait for it to disappear and ensure the target is interactable.
- ElementNotInteractableException — the element exists but cannot be interacted with in its current state; check visibility, enabled state, and the correct locator.
- NoSuchFrameException — the requested frame could not be found; verify the frame locator and page state.
- InvalidSelectorException — malformed XPath or CSS selector; validate the selector syntax.
- SessionNotCreatedException — browser/driver/configuration incompatibility or browser startup problem; check versions, capabilities, and environment.

```
WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));
WebElement save = wait.until(
    ExpectedConditions.elementToBeClickable(By.id("saveButton")));
save.click();
```

Avoid Thread.sleep() as the default fix. Prefer explicit waits that describe the condition you need. Import java.time.Duration and the relevant Selenium support classes for Selenium 4.

7. Can static methods be overridden?

No. A static method belongs to the class, so it is hidden—not overridden—when a subclass declares a static method with the same signature. Instance methods can be overridden and use runtime polymorphism.

```
class Parent {
    static void show() { System.out.println("Parent static"); }
    void display() { System.out.println("Parent instance"); }
}
class Child extends Parent {
    static void show() { System.out.println("Child static"); } // hiding
    @Override
    void display() { System.out.println("Child instance"); }   // overriding
}
Parent obj = new Child();
obj.show();    // Parent static (resolved using reference type)
obj.display(); // Child instance (runtime dispatch)
```

8. There are 50 names, each with an Edit link. How do you click Edit for your name if it is on row 39?

Do not hard-code row 39 unless the requirement specifically says row 39. Find the row containing the exact name, then click the Edit link inside that row. This is more robust if row order changes.

```
String targetName = "Your Name";

WebElement targetRow = driver.findElement(
    By.xpath("//table[@id='users']//tr[td[normalize-space()='" +
              targetName + "']]"));

targetRow.findElement(By.linkText("Edit")).click();
```

If the name is in a different column or the link is a button, adjust the row XPath and child locator. For dynamic text, use parameterized locator helpers and escape XPath values safely rather than concatenating arbitrary user input.

Alternative when you know the exact row number: locate the 39th row using XPath indexing carefully. XPath positions are 1-based, and headers may count as a row depending on the locator.

```
// Example only: if the 39th tr really is the target data row
driver.findElement(By.xpath(
    "(//table[@id='users']//tr)[39]//a[normalize-space()='Edit']")).click();
```

9. The question repeats: “if it is present somewhere in the 39th row?”

Treat this as a clarification of Question 8: if the name is known to be in row 39, locate that row and click Edit within it. If the row is not guaranteed, search by the name instead. Always scope the Edit locator to the matching row, because a page with 50 identical Edit links can otherwise cause the wrong record to be edited.

```
WebElement row39 = driver.findElement(
    By.xpath("(//table[@id='users']//tr)[39]"));
WebElement editLink = row39.findElement(
    By.xpath(".//a[normalize-space()='Edit']"));
editLink.click();
```

10. Tell me about yourself

Use a 60–90 second structure: experience, core technical skills, framework, responsibilities, leadership, and what role you are looking for. Keep claims accurate and supportable.

Sample answer tailored to a senior QA automation profile: “I have around 13 years of experience in software testing across manual and automation testing. My core automation experience includes Selenium with Java, Cucumber, TestNG, and the Page Object Model, along with API automation and tools such as RestAssured and Playwright. I have worked on designing and maintaining reusable automation frameworks, preparing test strategies and test scenarios, reviewing test results, investigating failures, and coordinating with developers and business teams to resolve defects. In my current role, I also contribute to test planning, execution tracking, quality reporting, and team coordination. I focus on improving regression coverage, maintainability, and reliable feedback in CI. I’m now looking for a role where I can use both hands-on automation and QA leadership experience to improve product quality.”

Personalize the tools, project domain, measurable achievements, team size, and role title. Do not include a tool or responsibility unless you can explain a real example in follow-up questions.

11. Tell me about your Selenium project experience

Use this structure: application/domain → your role → framework architecture → test execution → reporting/CI → challenges and results.

Sample answer: “In my project, we use Selenium WebDriver with Java to automate important end-to-end regression flows. The framework follows a hybrid approach using Cucumber for readable feature scenarios and the Page Object Model to separate page locators/actions from test logic. Common utilities handle waits, configuration, test data, screenshots, and reusable browser actions. Tests are organized by feature and tags so we can run smoke tests for quick feedback and broader regression suites before releases. We review failed tests to distinguish application defects from environment, data, timing, or locator issues, and publish execution results for the team. I contribute to scenario design, framework maintenance, code reviews, debugging, execution analysis, and coordination with developers and business analysts.”

Be ready to describe one real scenario end to end, how you handle waits, how test data is managed, how reports are generated, how tests run in CI, and a difficult defect you found. Add only tools actually used in your project (for example, Maven, Git, Jenkins, TestNG, Extent Reports, or Allure).

12. How are OOP concepts related to your Selenium project?

Object-oriented programming helps make automation code reusable, organized, and maintainable.

- Encapsulation: a page class keeps locators and page operations together and exposes methods such as login(username, password).
- Abstraction: test steps call methods such as loginPage.login(...) without needing to know the low-level click/sendKeys details.
- Inheritance: common setup or shared page behavior may be placed in a base class, but avoid deep inheritance chains.
- Polymorphism: a WebDriver reference can point to different browser driver implementations; overridden methods can provide specialized behavior.

```
public class LoginPage {
    private final WebDriver driver;
    private final By username = By.id("username");
    private final By password = By.id("password");
    private final By loginButton = By.id("login");
```

```
    public LoginPage(WebDriver driver) {
        this.driver = driver;
    }

    public void login(String user, String pass) {
        driver.findElement(username).sendKeys(user);
        driver.findElement(password).sendKeys(pass);
        driver.findElement(loginButton).click();
    }
}

// Test code uses the abstraction:
LoginPage loginPage = new LoginPage(driver);
loginPage.login("testuser", "secret");
```

Explain that private locators demonstrate encapsulation, the login() method is an abstraction, and WebDriver is an interface implemented by ChromeDriver, FirefoxDriver, and other drivers. Do not claim every use of a base class is automatically good design.

13. What is the difference between a web server and an application server?

A web server primarily handles HTTP requests and responses and commonly serves static content such as HTML, CSS, JavaScript, and images. It can also reverse-proxy requests to application services. An application server runs application/business logic and may manage transactions, security, connection pools, messaging, and other enterprise services.

Examples: Nginx and Apache HTTP Server are commonly used as web servers/reverse proxies. Apache Tomcat is a Java servlet container often called an application server in everyday conversation, though it is more precisely a servlet container and does not provide the full set of services of a Jakarta EE application server. WildFly and Open Liberty are examples of fuller Jakarta EE application servers.

Typical request flow: Browser → web server/reverse proxy → application runtime → database or downstream service → response back to browser. These roles can overlap; modern deployments often use embedded application servers or cloud gateways rather than a separate traditional web-server tier.

Interview summary: “The web server focuses on HTTP handling and static content or proxying; the application server/runtime executes dynamic application logic. The exact boundary depends on the technology and deployment architecture.”

Quick revision checklist

- Know your real Selenium, Java, Chrome, and Firefox versions and where to verify them.
- Use explicit waits and robust locators; avoid fixed sleeps as the first solution.
- For sorting checks, compare the original UI order with a separately sorted copy.
- For repeated Edit links, locate the matching row first, then find Edit inside that row.
- Keep your self-introduction and project explanation truthful, concise, and supported by examples.
- Connect OOP concepts to concrete classes and methods in your actual framework.