Selenium

Short notes for beginners · Java · WebDriver · TestNG · QA freshers

Uncodemy Handwritten Notes · free for students

I. Testing and Selenium Basics

- **Manual testing** = a person runs test cases step by step. **Automation testing** = scripts run the test cases. Automate tests that are repeated often, stable, data heavy or needed for regression. Do not automate one-time or fast-changing tests.
- **Selenium** is a free, open source tool to automate web browsers. Supports Java, Python, C#, JavaScript, Ruby, Kotlin and browsers Chrome, Firefox, Edge, Safari. Works on Windows, Linux and macOS.

➤ Selenium suite

Component	Purpose
Selenium IDE	Record and playback browser extension, good for quick tests
Selenium WebDriver	Main API: controls the browser through code
Selenium Grid	Run tests on many machines and browsers in parallel
Selenium Manager	Built in from 4.6, downloads the correct browser driver itself

- Selenium RC was the old tool and is removed in Selenium 4. WebDriver follows the **W3C WebDriver standard**. It cannot automate desktop apps, captcha or barcode readers. Use Appium for mobile and Postman or REST Assured for APIs.
- Architecture: Test script → Selenium client library → (JSON over HTTP) → Browser driver (chromedriver, geckodriver, msedgedriver) → Browser.

2. Setup and First Script

- Install JDK, an IDE (IntelliJ or Eclipse), Maven, and add the selenium-java and testng dependencies in pom.xml. With Selenium Manager there is no need to download the driver manually.

EXAMPLE `WebDriver driver = new ChromeDriver(); driver.get("https://example.com"); System.out.println(driver.getTitle()); driver.quit();`

- WebDriver is an interface. ChromeDriver, FirefoxDriver, EdgeDriver implement it. Using the interface type lets you change the browser by changing only one line.

➤ Browser commands

Command	What it does
`get(url)` / `navigate().to(url)`	Open a URL
`getTitle()`, `getCurrentUrl()`	Read page title and current URL
`getPageSource()`	Return HTML of the page
`navigate().back()` / `forward()` / `refresh()`	Browser history and reload
`manage().window().maximize()`	Maximise the window
`close()`	Close the current window or tab

Command	What it does
quit()	Close all windows and end the session

REMEMBER close() closes only the window in focus. quit() closes every window and ends the driver session. Always call quit() at the end to free memory.

3. Locators and WebElements

- A **locator** tells Selenium how to find an element: driver.findElement(By.id("email")). findElement returns one WebElement (throws NoSuchElementException if absent). findElements returns a list (empty if absent).

Locator	Example	Note
id	By.id("email")	Fastest and best if unique
name	By.name("q")	Common for form fields
className	By.className("btn")	Single class only
tagName	By.tagName("a")	Good for lists of links
linkText	By.linkText("Login")	Full visible text of a link
partialLinkText	By.partialLinkText("Log")	Part of the link text
cssSelector	By.cssSelector("input#email")	Fast and readable
xpath	By.xpath("//input[@id='email']")	Most powerful and flexible

- **XPath**: absolute starts with / (breaks easily), relative starts with // (preferred). Functions: contains(), starts-with(), text(), and / or. Axes: parent, ancestor, following-sibling, preceding-sibling.

EXAMPLE //a[contains(text(),'Login')] //div[@class='card']//button[1] //label[text()='Email']/following-sibling::input

- **CSS selector** patterns: #id, .class, tag.class, parent > child, [attr='v'], [attr^='start'], [attr$='end'], [attr*='part']. Prefer id, then name, then CSS, then XPath.

- WebElement commands: click(), sendKeys(), clear(), getText(), getAttribute("value"), getCssValue(), isDisplayed(), isEnabled(), isSelected(), submit().

4. Waits and Synchronisation

- Pages load at different speeds, so scripts must wait. Without waits you get NoSuchElementException or ElementNotInteractableException. Thread.sleep() is a fixed delay and should be avoided.

Wait	How it works
Implicit	Global timeout for finding elements: implicitlyWait(Duration.ofSeconds(10))
Explicit	Waits for one condition on one element: WebDriverWait + ExpectedConditions
Fluent	Explicit wait with polling interval and ignored exceptions

EXAMPLE WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));
WebElement e = wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("msg")));

- Common conditions: elementToBeClickable, visibilityOfElementLocated, presenceOfElementLocated, titleContains, alertIsPresent, frameToBeAvailableAndSwitchToIt, invisibilityOfElementLocated.
- Do not mix implicit and explicit waits, as the total wait time becomes unpredictable. Best practice: use explicit waits.
- Page load timeout: manage().timeouts().pageLoadTimeout(Duration.ofSeconds(30)). Exceptions to know: StaleElementReferenceException (element changed in DOM), TimeoutException, NoSuchElementException.

5. Handling Web Elements

- **Dropdown**: Select s = new Select(element); s.selectByVisibleText("India"); selectByValue(); selectByIndex(); isMultiple(); getOptions().
- **Alerts**: Alert a = driver.switchTo().alert(); a.getText(); a.accept(); a.dismiss(); a.sendKeys("text"). **Frames**: switchTo().frame(index / name / element), then defaultContent() or parentFrame() to come back.
- **Windows and tabs**: getWindowHandle() (current) and getWindowHandles() (all, a Set). Loop over handles and use switchTo().window(handle). Selenium 4: switchTo().newWindow(WindowType.TAB).
- **Actions class**: moveToElement (hover), doubleClick, contextClick (right click), dragAndDrop, keyDown / keyUp, sendKeys. Always end with .perform().
- **JavascriptExecutor**: js.executeScript("arguments[0].scrollIntoView(true);", element) for scrolling or clicking hidden elements. **Screenshots**: ((TakesScreenshot) driver).getScreenshotAs(OutputType.FILE).
- File upload: use sendKeys("C:\\path\\file.pdf") on <input type="file">. Checkbox / radio: check isSelected() before click(). Tables: loop rows and columns with XPath.

6. TestNG Framework

- **TestNG** (Test Next Generation) is a test framework for Java with annotations, grouping, priorities, parallel runs, data providers and HTML reports. Annotation @Test marks a test method.

Annotation	Runs
@BeforeSuite / @AfterSuite	Once before / after the whole suite
@BeforeTest / @AfterTest	Before / after each <test> tag in testng.xml
@BeforeClass / @AfterClass	Before / after the first / last method of a class
@BeforeMethod / @AfterMethod	Before / after every @Test (open and close browser)
@Test	The test itself

Annotation	Runs
@DataProvider	Supplies multiple data sets to a test

- **@Test attributes**: priority (lower runs first, default 0), enabled=false, dependsOnMethods, groups, invocationCount, timeOut, expectedExceptions. Without priority, methods run in alphabetical order.
- **Assertions**: Assert.assertEquals(actual, expected), assertTrue, assertFalse, assertNotNull. A failed Assert stops that test. **SoftAssert** collects all failures and reports at assertAll().
- **testng.xml** defines suite, test, classes, packages, groups, parameters and parallel="tests" / "classes" / "methods". Reports go to the test-output folder. Listeners (ITestListener) and IRetryAnalyzer re-run failed tests.

7. Frameworks and Page Object Model

- **Page Object Model (POM)**: one class per page that holds locators and actions as methods. Tests call these methods. Benefits: less duplicate code, easy maintenance (change a locator in one place), readable tests.

Framework type	Idea
Data driven	Test data from Excel (Apache POI), CSV, JSON or DB
Keyword driven	Actions written as keywords in a sheet
Hybrid	Mix of POM + data driven + utilities
BDD (Cucumber)	Gherkin steps: Given, When, Then; business readable

- Typical stack: Java + Selenium + TestNG + Maven + Git + Jenkins (CI) + Extent / Allure reports + Log4j logging. Folders: base, pages, tests, utils, resources, testdata, reports.
- Maven runs tests with the Surefire plugin: mvn clean test. Jenkins can run this on every commit and email the report.

8. Selenium 4, Grid and Best Practices

- **Selenium 4** features: relative locators (above, below, toLeftOf, toRightOf, near), new window / tab API, element screenshots, Chrome DevTools Protocol access, improved Grid, W3C standard.
- **Grid 4** modes: Standalone, Hub and Node, Fully Distributed. Use it to run the same suite on many browsers and OS versions at the same time, or use cloud services like BrowserStack and LambdaTest.
- Headless run (no UI window): options.addArguments("--headless=new"). Use ChromeOptions for incognito, disabling notifications and window size.

➤ Good practice

- Keep tests independent; use stable locators (id, data-testid) instead of long XPath; use explicit waits; one assertion goal per test; clean up data; avoid hard-coded values; keep browser setup in a base class.
- Flaky test? Check waits, unstable locators, shared test data and timing. Retry is only a patch, find the real cause.

9. Most Asked Questions (practice)

1. What is Selenium? Explain its components.
2. Difference between close() and quit(). Why is WebDriver an interface?
3. Name all locators. XPath vs CSS selector, absolute vs relative XPath.
4. Implicit vs explicit vs fluent wait.
5. How do you handle dropdowns, alerts, frames and multiple windows?
6. What is the Actions class? Give an example of mouse hover.
7. Explain TestNG annotations and their execution order.
8. What is a DataProvider? Difference between Assert and SoftAssert.
9. What is Page Object Model? What are its benefits?
10. How do you run tests in parallel and on many browsers?

TIP Interview: be ready to write a login test with POM + TestNG on a whiteboard, and explain how you handle dynamic elements, waits and flaky tests.