**✅ Selenium & TestNG Notes**

**🔹 1) Difference Between findElement() and findElements()**

| **Method** | **Description** |
| --- | --- |
| findElement() | Returns a **single** WebElement.  If multiple elements match the locator, only the **first one** is returned.  🔴 **Throws NoSuchElementException** if not found. |
| findElements() | Returns a **List<WebElement>** of **all matching elements**. If none are found, it returns an **empty list**.  ✅ No exception is thrown. |

**🔹 2) Difference Between getWindowHandle() and getWindowHandles()**

| **Method** | **Description** |
| --- | --- |
| ✅ getWindowHandle() | Returns the **unique handle (ID)** of the **current active window**.  🔁 **Return type**: String  📌 Example: String currentWindow = driver.getWindowHandle(); |
| ✅ getWindowHandles() | Returns a **set of all open window handles** opened by WebDriver.  🔁 **Return type**: Set<String>  📌 Example: Set<String> allWindows = driver.getWindowHandles(); |

**🔹 3) What is TestNG?**

**TestNG** (Test Next Generation) is a powerful testing framework inspired by JUnit and NUnit, designed for **Java**.

**🔧 Supports:**

* Unit Testing
* Integration Testing
* Functional / End-to-End Testing
* Automation Testing (e.g., with Selenium)

**✨ Key Features:**

* Annotations
* Grouping
* Parallel execution
* Data-driven testing (@DataProvider)
* HTML & XML Reporting

**🔹 4) TestNG Annotations Summary**

| **Annotation** | **Description** |
| --- | --- |
| @BeforeSuite | Runs once **before all tests** in the suite.  🔸 (E.g., Load browser, connect DB) |
| @AfterSuite | Runs once **after all tests** in the suite.  🔸 (E.g., Close browser, disconnect DB) |
| @BeforeTest | Runs before any test method inside <test> tag (from testng.xml).  🔸 (E.g., Initialize browser, load env config) |
| @AfterTest | Runs after all test methods in <test> tag.  🔸 (E.g., Stop services, close DB/browser) |
| @BeforeClass | Runs once **before** the first method in the class.  🔸 (E.g., Setup DB, start browser) |
| @AfterClass | Runs once **after** all methods in the class.  🔸 (E.g., Clean up resources, log result) |
| @BeforeMethod | Runs **before each test method**.  🔸 (E.g., Login, init data, open page) |
| @AfterMethod | Runs **after each test method**.  🔸 (E.g., Logout, reset state, take screenshot) |
| @Test | Marks a method as a **test case**. |
| @DataProvider | Provides **test data** to test methods. |
| @Parameters | Injects parameters from testng.xml.  🔸 (e.g., browser type, environment) |
| @BeforeGroups | Runs **before** test methods in a specific group.  🔸 (E.g., setup for UI group only) |
| @AfterGroups | Runs **after** test methods in a specific group.  🔸 (E.g., cleanup after group tests) |
| @Listeners | Used to attach custom listeners to test classes.  🔸 (E.g., Retry logic, report generation) |
| @Factory | Used to create **multiple test instances dynamically** at runtime.  🔸 (E.g., Run same test class with different input data) |

**📌 Summary Usage Map**

| **Annotation Type** | **Used For** | **Scope** |
| --- | --- | --- |
| @BeforeSuite / @AfterSuite | Setup/teardown entire test suite | Global |
| @BeforeTest / @AfterTest | Setup/cleanup per <test> tag | XML-based |
| @BeforeClass / @AfterClass | Run once before/after class | Class-level |
| @BeforeMethod / @AfterMethod | Run before/after each test method | Method-level |
| @BeforeGroups / @AfterGroups | Group-specific setup/cleanup | Group-level |

5)what is locator in selenium

✅ What is a Locator in Selenium?

A locator in Selenium is a way to identify elements on a web page (such as buttons, links, input fields, etc.) so that Selenium can interact with them (like clicking, typing, or verifying content).

Locators tell Selenium “which element on the page you want to work with.”

🔹 Common Types of Locators in Selenium

Id--Locates an element by its id attribute (must be unique).

Ex--driver.findElement(By.id("username"))

Name--Locates by the name attribute.

Ex--driver.findElement(By.name("email"))

className--Locates elements by their class attribute.

Ex--driver.findElement(By.className("btn-primary"))

tagName--Locates elements by their HTML tag.

Ex--driver.findElement(By.tagName("input"))

linkText --Finds a link (<a>) using its exact text.

Ex--driver.findElement(By.linkText("Login"))

partialLinkText--Finds a link using partial match of link text.

Ex--driver.findElement(By.partialLinkText("Sign"))

cssSelector--Finds elements using CSS syntax. Very flexible and powerful.

Ex--driver.findElement(By.cssSelector("input[type='text']"))

xpath--Uses XML path to locate elements. Extremely powerful and supports complex conditions.

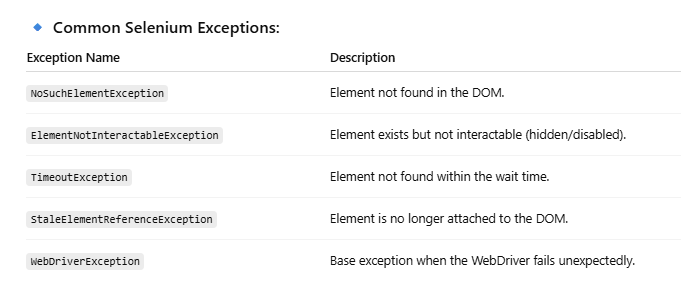
Ex--driver.findElement(By.xpath("//input[@id='password']"))

A screenshot of a computer program

AI-generated content may be incorrect.

**✅ What is a Selenium Exception?**

In Selenium, an **exception** is an error that occurs during the execution of a Selenium script due to unexpected events — like missing elements, timing issues, or browser-related errors.



**❓ What is StaleElementReferenceException?**

StaleElementReferenceException occurs when a **WebElement is no longer valid**, usually due to:

* Page reload
* DOM structure change
* Element is removed and recreated dynamically

🧠 **Key reason**: The previously located element is no longer attached to the DOM.

A screenshot of a computer program

AI-generated content may be incorrect.

**A screen shot of a computer code

AI-generated content may be incorrect.✅ Best Practices to Avoid StaleElementReferenceException:**

* Avoid storing WebElement in variables too early.
* Always interact with elements just after locating them.
* Use **explicit waits** (WebDriverWait) for dynamic elements.
* Re-locate the element if page refresh or DOM update happens.

**✅ Waits in Selenium**

In Selenium, **waits** are used to pause the test execution **until a certain condition is met** (like element is visible, clickable, or present in the DOM). This helps handle **dynamic content** and **synchronization issues**.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**ChatGPT said:**

**✅ What is WebDriver in Selenium?**

**Selenium WebDriver** is a **core component** of the Selenium framework.  
It provides a **programming interface to interact with web browsers** like Chrome, Firefox, Edge, Safari, etc.

With WebDriver, you can:

* Open and close browsers
* Navigate to URLs
* Locate and interact with web elements (click, type, etc.)
* Handle alerts, frames, and windows
* Take screenshots, manage cookies, etc.

**🔹 WebDriver Interface**

WebDriver is an **interface** in the org.openqa.selenium package.  
It is implemented by browser-specific classes like:

* ChromeDriver
* FirefoxDriver
* EdgeDriver
* SafariDriver

A screenshot of a computer

AI-generated content may be incorrect.

**✅ How to Find Broken Links on a Webpage Using Selenium**

Selenium itself **cannot check HTTP response codes** directly — it can only interact with elements on the page.  
To find **broken links**, you need to:

1. **Collect all the <a> tag links (hrefs)** using Selenium.
2. **Use Java's HttpURLConnection** (or similar library) to check if the link returns a **200 OK** or **error like 404/500**, etc.

**🔸 Step-by-Step Approach:**

1. Launch browser using Selenium.
2. Use findElements(By.tagName("a")) to get all link elements.
3. Extract href attributes.
4. Pass each URL to HttpURLConnection.
5. Check the response code:
   * 200 → Valid link
   * 404, 500, etc. → Broken link

A screenshot of a computer program

AI-generated content may be incorrect.

**✅ How to Handle Dynamic Objects in Selenium**

In real-world applications, web elements (like IDs, XPaths, or class names) often change dynamically — either partially or completely — every time the page loads. These are called **dynamic elements** or **dynamic locators**.

**🔹 Common Problems with Dynamic Elements:**

* IDs like ctl00\_MainContent\_user\_1234 change on every page load.
* Attributes like class="btn-xyz-5678" contain random or session-specific parts.
* Text or position may vary due to layout changes.

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

**✅ How to Handle Dropdown in Selenium**

To interact with dropdowns (<select> HTML tag) in Selenium, you use the \*\*Select\*\* class provided by Selenium WebDriver.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

**✅ How to Handle Alerts and Popups in Selenium**

In Selenium, JavaScript alerts and popups are handled using the **Alert interface**, which is part of the WebDriver API.

**🔹 Types of JavaScript Alerts:**

1. **Simple Alert** – Just an OK button
2. **Confirmation Alert** – OK and Cancel buttons
3. **Prompt Alert** – Accepts user input

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

**✅ Difference between driver.get() and driver.navigate().to() in Selenium**

Both driver.get() and driver.navigate().to() are used to **open a URL** in Selenium, but they differ slightly in **functionality and internal implementation**.

**🔹 1) driver.get(String url)**

* Loads a new web page in the current browser window.
* Waits until the page is fully loaded.
* **Simpler and more commonly used** for opening a web page.
* No navigation history is maintained.

**2) driver.navigate().to(String url)**

* Does the **same thing** as driver.get(), but with more **control**.
* Part of the **Navigation interface**, which also provides:
  + back()
  + forward()
  + refresh()
* Maintains **browser history**, enabling back/forward navigation.

**✅ How to Switch Between Multiple Tabs and Windows in Selenium**

In Selenium WebDriver, **each tab or window** has a unique **window handle** (a String), which you can use to switch between them.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

**✅ How to Handle Frames and iFrames in Selenium**

In Selenium, **frames** and **iframes** are HTML elements used to embed another HTML page inside a current page. To interact with elements inside a frame, you **must switch the WebDriver's context** to that frame first.

**🔹 Why Switch to Frame?**

You **cannot access** elements inside a frame or iframe **directly** without switching into it first.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**✅ How to Run Selenium Tests in Parallel Using TestNG**

Running Selenium tests in parallel helps reduce execution time by **executing multiple tests or classes at the same time**. This is commonly done using the **TestNG framework**.

**🔹 Step-by-Step Guide to Run Tests in Parallel:**

**✅ 1. Add TestNG to Your Project**

Make sure testng.xml and testng dependency are set up in your Java project (Maven/Gradle or manually).

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

**✅ How to Perform Cross-Browser Testing in Selenium**

Cross-browser testing ensures that your application works correctly across **multiple browsers** like **Chrome, Firefox, Edge**, etc.

**🔹 Steps to Perform Cross-Browser Testing in Selenium (with TestNG)**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A white background with black border

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

To **fetch data from an Excel file in Selenium (Java)**, you typically use **Apache POI** – a powerful Java library for reading and writing Microsoft Office documents.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer code

AI-generated content may be incorrect.

**✅ What is POM (Page Object Model) in Selenium?**

**Page Object Model (POM)** is a **design pattern** used in Selenium test automation to create an object repository for web elements.

In POM, **each web page is represented as a separate class**, and the elements on the page are defined as variables inside that class. All interactions (methods) with those elements are defined within the same class.

A screenshot of a computer

AI-generated content may be incorrect.

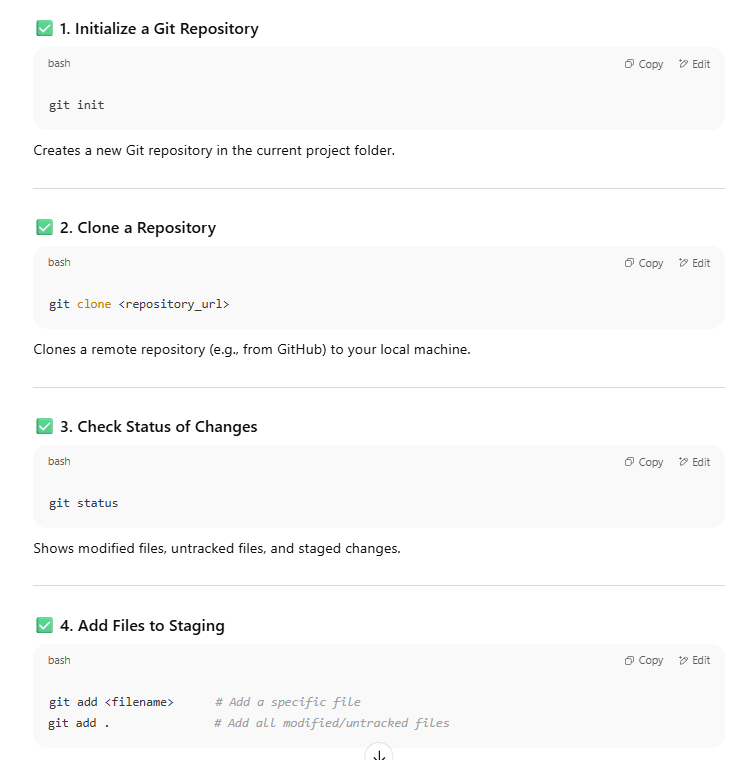
A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

GIT command



A screenshot of a chat

AI-generated content may be incorrect.

A screenshot of a chat

AI-generated content may be incorrect.

A white background with black text

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.