SELENIUM

Quick Learning Guide

🗓️ Selenium Self-Learning Guide – Day 1

Topic: Selenium Basics (WebDriver, Locators, Handling WebElements)

📌 Day 1 Agenda

---

1 Introduction to Selenium & WebDriver

Selenium is an open-source **automation testing framework** used for web applications. It supports multiple browsers (Chrome, Firefox, Edge) and multiple programming languages (Java, Python, C#).

What is WebDriver?

Selenium WebDriver is an automation tool that interacts with web elements. It acts as a bridge between the **test script and the web browser**.

✓ Why WebDriver?

✅ Faster execution than Selenium RC.
✅ Works with multiple browsers.
✅ Supports dynamic web applications.

---

2 Setting Up Selenium WebDriver

Step 1: Install Required Components

- **Java** (JDK 11 or later)
- **Maven** (Optional but recommended)
- **Eclipse/IntelliJ IDE**
- **Selenium WebDriver JARs or Maven Dependency**

Step 2: Add Selenium Maven Dependency

If using Maven, add the following dependency in pom.xml:

```
<dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>4.12.1</version>
</dependency>
```

Step 3: Download and Setup WebDriver

📌 Download **ChromeDriver**: https://chromedriver.chromium.org/downloads
📌 Add WebDriver path in code:

```
System.setProperty("webdriver.chrome.driver", "path_to_chromedriver");
```

---

3 Basic WebDriver Commands

📌 **Launching a Browser & Navigating to URL**

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class OpenBrowser {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver", "path_to_chromedriver");
        WebDriver driver = new ChromeDriver();
        driver.get("https://www.google.com");
        driver.manage().window().maximize();
        System.out.println("Page Title: " + driver.getTitle());
        driver.quit();
    }
}
```

✓ **get()** – Opens URL
✓ **getTitle()** – Fetches page title
✓ **manage().window().maximize()** – Maximizes browser
✓ **quit()** – Closes browser

---

4 Locators in Selenium

Locators help find elements on a web page.

📌 **Types of Locators & Examples**

Locator Type	Example
ID	driver.findElement(By.id("username"))
Name	driver.findElement(By.name("password"))
Class Name	driver.findElement(By.className("login-btn"))
Tag Name	driver.findElement(By.tagName("input"))
CSS Selector	driver.findElement(By.cssSelector("#submit"))

Locator Type	Example
XPath	driver.findElement(By.xpath("//button[text()='Login']"))

📌 Finding Elements using Locators

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class LocatorsExample {
    public static void main(String[] args) {
        WebDriver driver = new ChromeDriver();
        driver.get("https://example.com/login");

        // Locating Elements
        WebElement username = driver.findElement(By.id("username"));
        WebElement password = driver.findElement(By.name("password"));
        WebElement loginButton = driver.findElement(By.className("login-btn"));

        // Performing Actions
        username.sendKeys("testuser");
        password.sendKeys("password123");
        loginButton.click();

        driver.quit();
    }
}
```

---

5 Handling WebElements

📌 Performing Actions on WebElements

✓ Clicking a button

```
driver.findElement(By.id("loginBtn")).click();
```

✓ Typing into an input field

```
driver.findElement(By.name("username")).sendKeys("testuser");
```

✓ Getting text from an element

```
String text = driver.findElement(By.tagName("h1")).getText();
System.out.println("Header Text: " + text);
```

✓ **Verifying if an element is displayed**

```
boolean isVisible = driver.findElement(By.id("welcomeMessage")).isDisplayed();
System.out.println("Is Welcome Message Visible? " + isVisible);
```

---

6 Day 1 Exercises

1 Write a script to:

- ✅ Open Chrome browser
- ✅ Navigate to "https://www.google.com"
- ✅ Print the page title
- ✅ Close the browser

2 Create a login automation script:

- ✅ Visit a demo login page
- ✅ Enter username & password
- ✅ Click on login button
- ✅ Print success/failure message

3 Find and interact with elements using locators

- ✅ Locate elements using **ID, Name, Class, XPath, CSS Selector**

---

7 Best Practices for Selenium Basics

✓ **Use unique locators** – Prefer **ID, Name** over XPath for stability.
✓ **Maximize browser window** before interactions.
✓ **Use quit() instead of close()** to release WebDriver.
✓ **Use proper waits (Implicit & Explicit Waits)** instead of Thread.sleep().
✓ **Maintain code structure** – Create **separate classes** for test cases and reusable functions.

---

8 Interview Questions & Evaluation Topics

📌 **General Selenium Questions**

- 🔹 What is Selenium WebDriver?
- 🔹 How do you launch a browser using Selenium?
- 🔹 What are the different types of locators in Selenium?
- 🔹 What is the difference between findElement() and findElements()?
- 🔹 How do you handle a scenario where an element is not found?

📌 **Coding Questions**

- 🔹 Write a Selenium script to open a website, search for a keyword, and print search results.
- 🔹 Automate a login form and verify successful login.

---

9 Summary of Selenium Day 1

✓ **Introduction to Selenium WebDriver**
✓ **Setting up WebDriver & Maven dependencies**
✓ **Locating elements using ID, Name, Class, XPath, CSS**
✓ **Performing actions like click, sendKeys, getText**
✓ **Best practices for Selenium automation**
✓ **Interview questions and coding exercises**

📅 Selenium Self-Learning Guide – Day 2

Topic: Waits (Implicit, Explicit, Fluent)

📌 Day 2 Agenda:

1 Why do we need waits in Selenium?
2 Types of Waits in Selenium - Implicit Wait, Explicit Wait, Fluent Wait
3 Handling Synchronization Issues
4 Hands-on Examples
5 Day 2 Exercises
6 Best Practices
7 Possible Interview Questions

---

1 Why Do We Need Waits in Selenium?

When automating web applications, elements may not be **immediately available** due to:

✅ Network latency
✅ JavaScript execution delays
✅ Dynamic content loading (e.g., AJAX calls)
✅ Animations or pop-ups

If we don't use waits, our script might fail with errors like:

❌ NoSuchElementException – Element not found
❌ ElementNotInteractableException – Element not ready for interaction
❌ StaleElementReferenceException – Element no longer present

🛠 **Solution?** Selenium provides **Waits** to handle such scenarios.

---

2 Types of Waits in Selenium

📌 1. Implicit Wait

📍 Implicit Wait **instructs WebDriver to wait for a specified time** before throwing NoSuchElementException.

📍 It applies **globally** to all elements in a session.

Syntax:

```
driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));
```

- WebDriver will **wait up to 10 seconds** for an element to be found before throwing an error.

Example: Implicit Wait

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.By;
import org.openqa.selenium.chrome.ChromeDriver;
import java.time.Duration;

public class ImplicitWaitExample {
    public static void main(String[] args) {
        WebDriver driver = new ChromeDriver();
        driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10)); // Implicit Wait
        driver.get("https://example.com/login");

        WebElement username = driver.findElement(By.id("username"));
        WebElement password = driver.findElement(By.id("password"));
        WebElement loginButton = driver.findElement(By.id("loginBtn"));

        username.sendKeys("testuser");
        password.sendKeys("password123");
        loginButton.click();

        driver.quit();
    }
}
```

✓ **Advantages**: Simple to implement
❌ **Disadvantage**: Cannot handle dynamic conditions like visibility, clickability

---

📌 **2. Explicit Wait**

- **Explicit Wait** waits for a **specific condition** before proceeding.
- Used when we need to **wait for a particular element** instead of waiting globally.
- Uses WebDriverWait with ExpectedConditions.

Common Conditions in Explicit Wait:

Expected Condition	Description
visibilityOfElementLocated(By)	Waits until element is visible

Expected Condition	Description
elementToBeClickable(By)	Waits until element is clickable
presenceOfElementLocated(By)	Waits until element is present in DOM
textToBePresentInElement(By, text)	Waits until text is present in element
alertIsPresent()	Waits for alert to appear

Example: Explicit Wait

```java
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.By;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.openqa.selenium.support.ui.ExpectedConditions;
import java.time.Duration;

public class ExplicitWaitExample {
    public static void main(String[] args) {
        WebDriver driver = new ChromeDriver();
        driver.get("https://example.com/login");

        WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10)); // Explicit Wait

        WebElement loginButton = wait.until(ExpectedConditions.elementToBeClickable(By.id("loginBtn")));
        loginButton.click();

        driver.quit();
    }
}

```

✓ **Advantages**: More efficient, waits only when needed
❌ **Disadvantage**: Needs to be applied for each element separately

---

📌 3. Fluent Wait

📍 Similar to **Explicit Wait**, but checks the condition **at regular intervals** instead of waiting blindly.
📍 Useful for highly **dynamic elements** that may appear/disappear unpredictably.

Syntax:

```java
FluentWait<WebDriver> wait = new FluentWait<>(driver)
        .withTimeout(Duration.ofSeconds(15))  // Maximum wait time
        .pollingEvery(Duration.ofSeconds(2))  // Check every 2 seconds
        .ignoring(NoSuchElementException.class);
```

Example: Fluent Wait

```java
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.By;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.FluentWait;
import org.openqa.selenium.support.ui.Wait;
import java.time.Duration;
import java.util.function.Function;

public class FluentWaitExample {
    public static void main(String[] args) {
        WebDriver driver = new ChromeDriver();
        driver.get("https://example.com/dynamic");

        Wait<WebDriver> wait = new FluentWait<>(driver)
                .withTimeout(Duration.ofSeconds(15))
                .pollingEvery(Duration.ofSeconds(2))
                .ignoring(Exception.class);

        WebElement dynamicElement = wait.until(new Function<WebDriver, WebElement>() {
            public WebElement apply(WebDriver driver) {
                return driver.findElement(By.id("dynamicElement"));
            }
        });

        dynamicElement.click();
        driver.quit();
    }
}
```

✓ **Advantages**: Highly customizable, polls frequently
❌ **Disadvantage**: Complex to implement

---

3 Handling Synchronization Issues

🚀 **Which wait to use?**
✅ Use **Implicit Wait** for simple page load waits.
✅ Use **Explicit Wait** for elements that appear after some action.
✅ Use **Fluent Wait** for elements that load unpredictably.

---

4 Day 2 Exercises

1 Write a script using Implicit Wait to:
✅ Open browser & navigate to "https://www.google.com"

✅ Search for "Selenium WebDriver"
✅ Click on the first search result

2 Write a script using Explicit Wait to:
✅ Click a button only when it is clickable
✅ Print the text of an element once it is visible

3 Write a script using Fluent Wait to:
✅ Handle a dynamically loading element

---

5 Best Practices for Using Waits

✓ **Avoid Thread.sleep()** – It slows execution.
✓ **Prefer Explicit Wait over Implicit Wait** – More control over waiting conditions.
✓ **Use Fluent Wait for unpredictable elements.**
✓ **Set reasonable timeout values** – Avoid waiting too long.
✓ **Use ExpectedConditions wisely** – Choose the right condition.

---

6 Possible Interview Questions & Evaluation Topics

📌 **Conceptual Questions**
🔹 What is the difference between Implicit and Explicit Wait?
🔹 When should you use Fluent Wait?
🔹 Why is using Thread.sleep() a bad practice?
🔹 Explain ExpectedConditions in Selenium.
🔹 What happens if the wait condition is not met?

📌 **Coding Questions**
🔹 Write a script that waits for a dynamic dropdown to load and then selects an option.
🔹 Automate a login scenario where the "Login" button becomes visible after an API response.

---

7 Summary of Selenium Day 2

✓ **Understanding the need for waits**
✓ **Using Implicit, Explicit, and Fluent Waits**
✓ **Handling synchronization issues effectively**
✓ **Best practices and real-world examples**
✓ **Interview questions and hands-on exercises**

🗓️ Selenium Self-Learning Guide – Day 3

Topic: Actions, Alerts, Frames

📌 Day 3 Agenda:

1 Understanding Actions Class in Selenium

- Mouse Hover
- Double Click
- Right Click (Context Click)
- Drag and Drop
- Keyboard Actions

2 Handling Alerts in Selenium

- Simple Alert
- Confirmation Alert
- Prompt Alert

3 Handling Frames in Selenium

- Switching to a Frame
- Switching Back to Main Document
- Nested Frames

4 Hands-on Examples
5 Day 3 Exercises
6 Best Practices
7 Possible Interview Questions

---

1 Understanding Actions Class in Selenium

Why do we need Actions?
Selenium's Actions class is used to handle **mouse and keyboard interactions**, such as:
✅ Hovering over elements
✅ Drag and drop
✅ Right-click, Double-click
✅ Keyboard shortcuts

📌 Syntax:

```
Actions actions = new Actions(driver);
```

📌 1.1 Mouse Hover

Used when a dropdown or submenu appears only on hovering over an element.

Example: Mouse Hover

```java
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.interactions.Actions;

public class MouseHoverExample {
    public static void main(String[] args) {
        WebDriver driver = new ChromeDriver();
        driver.get("https://example.com");

        WebElement menu = driver.findElement(By.id("menu"));
        Actions actions = new Actions(driver);
        actions.moveToElement(menu).perform(); // Mouse hover

        WebElement submenu = driver.findElement(By.id("submenu"));
        submenu.click(); // Click on submenu

        driver.quit();
    }
}
```

📌 1.2 Double Click

Used when an element requires a **double click** action.

Example: Double Click

```java
actions.doubleClick(element).perform();
```

📌 1.3 Right Click (Context Click)

Used to **right-click** on an element and select an option.

Example: Right Click

```java
actions.contextClick(element).perform();
```

📌 1.4 Drag and Drop

Used to **move an element from one location to another**.

Example: Drag and Drop

```
WebElement source = driver.findElement(By.id("source"));
WebElement target = driver.findElement(By.id("target"));
actions.dragAndDrop(source, target).perform();
```

📌 1.5 Keyboard Actions

Used for **sending keyboard keys** like ENTER, TAB, etc.

Example: Sending Keys

```
actions.sendKeys(Keys.ENTER).perform();
```

---

2 Handling Alerts in Selenium

📌 Types of Alerts:

Type	Description
Simple Alert	Just shows an alert message with an OK button
Confirmation Alert	Asks for confirmation (OK & Cancel buttons)
Prompt Alert	Takes user input via a text box

📌 Handling Alerts:

```
Alert alert = driver.switchTo().alert();
```

📌 2.1 Handling a Simple Alert

```
alert.accept(); // Click OK
```

📌 2.2 Handling a Confirmation Alert

```
alert.dismiss(); // Click Cancel
```

📌 2.3 Handling a Prompt Alert

```
alert.sendKeys("Automation Testing");
alert.accept();
```

---

3 Handling Frames in Selenium

📌 **What are Frames?**

Frames are **HTML documents embedded within another HTML document**.

📌 **Switching to a Frame**

```
driver.switchTo().frame("frameName");
```

📌 **Switching Back to Main Document**

```
driver.switchTo().defaultContent();
```

📌 **Switching to a Frame by Index**

```
driver.switchTo().frame(0);
```

📌 **Handling Nested Frames**

```
driver.switchTo().frame("outerFrame");
driver.switchTo().frame("innerFrame");
```

---

4 Day 3 Exercises

1 Mouse Hover & Submenu Click

- ✅ Open https://example.com
- ✅ Hover over **Menu** and click **Submenu**

2 Right Click & Select Option

- ✅ Right-click an element
- ✅ Select **"Delete"**

3 Double Click & Verify Alert Message

- ✅ Double click an element
- ✅ Handle the alert

4 Handle Alerts

- ✅ Open https://example.com/alerts
- ✅ Handle all types of alerts (Simple, Confirmation, Prompt)

5 Handle Frames

- ✅ Switch between frames and extract text from an element inside a frame

---

5 Best Practices

✓ **Use Actions class for complex user interactions**
✓ **Always switch back to the main document after handling frames**
✓ **Use Explicit Waits before interacting with elements inside frames**
✓ **Always handle alerts gracefully to avoid script failures**

---

6 Possible Interview Questions & Evaluation Topics

📌 **Conceptual Questions**

- What is the Actions class in Selenium?
- How do you handle an alert in Selenium?
- What are the different types of alerts?
- How do you switch between frames?
- How do you perform drag and drop in Selenium?

📌 **Coding Questions**

- Write a script to **perform mouse hover and select a submenu option**.
- Write a script to **handle a confirmation alert and print the alert text**.
- Write a script to **switch to a nested frame and extract text**.

📌 Summary of Selenium Day 3

✓ **Understanding Actions class for mouse and keyboard interactions**
✓ **Handling different types of Alerts**
✓ **Switching between Frames in Selenium**
✓ **Best practices and real-world scenarios**
✓ **Interview questions and hands-on exercises**

🗓️ Selenium Self-Learning Guide – Day 4

Topic: Assertions + TestNG + BDD (Cucumber, Gherkin)

📌 Day 4 Agenda:

1 Assertions in Selenium

- Hard Assertions (JUnit & TestNG)
- Soft Assertions (TestNG)

2 TestNG Framework

- Installing TestNG in Eclipse
- Writing a TestNG Test Case

- TestNG Annotations
- Running TestNG XML

3 Introduction to BDD & Cucumber

- What is BDD?
- Understanding Cucumber & Gherkin Syntax

4 Implementing BDD using Cucumber

- Writing Feature Files
- Step Definitions
- Running Cucumber Tests

5 Day 4 Exercises
6 Best Practices
7 Possible Interview Questions

---

1 Assertions in Selenium

📌 **What is an Assertion?**
Assertions help **validate expected vs. actual results** in test scripts. If an assertion fails, the test case stops execution.

📌 **Types of Assertions:**

Type	Description
Hard Assertion	If assertion fails, the test stops
Soft Assertion	If assertion fails, the test continues execution

📌 1.1 Hard Assertions (TestNG)

Hard Assertions fail immediately and **stop execution**.

Example: Using Assert in TestNG

```
import org.testng.Assert;
import org.testng.annotations.Test;

public class HardAssertionExample {
    @Test
    public void testTitle() {
        String expectedTitle = "Google";
        String actualTitle = "Google";

        Assert.assertEquals(actualTitle, expectedTitle, "Title does not match!");
        System.out.println("This will execute only if assertion passes");
    }
}
```

📌 1.2 Soft Assertions (TestNG)

Soft Assertions allow the test **to continue execution** even after failure.

Example: Using SoftAssert

```
import org.testng.annotations.Test;
import org.testng.asserts.SoftAssert;

public class SoftAssertionExample {
    @Test
    public void testValues() {
        SoftAssert softAssert = new SoftAssert();

        softAssert.assertEquals(5, 10, "Values do not match!");
        softAssert.assertTrue(false, "This condition is false!");

        System.out.println("Execution continues even after failures");

        softAssert.assertAll(); // Marks the test as failed if any assertion failed
    }
}
```

---

2 TestNG Framework

📌 **Why TestNG?**

✓ Helps organize test cases

✓ Provides **annotations (@Test, @BeforeTest, @AfterTest)**

✓ Supports parallel execution

✓ Generates **HTML reports**

📌 **Installing TestNG in Eclipse:**

1 Open **Eclipse** → Go to **Help** → Select **Eclipse Marketplace**

2 Search for **TestNG** → Install

📌 2.1 Writing a Basic TestNG Test Case

```
import org.testng.annotations.Test;

public class TestNGExample {
    @Test
    public void testMethod() {
        System.out.println("Executing TestNG Test Case");
    }
}
```

📌 2.2 TestNG Annotations

Annotation	Description
@BeforeTest	Runs before any test method
@AfterTest	Runs after all test methods
@BeforeMethod	Runs before each test method
@AfterMethod	Runs after each test method
@Test	Defines a test case

Example: Using TestNG Annotations

```
import org.testng.annotations.*;

public class TestNGAnnotationsExample {
    @BeforeTest
    public void setUp() {
        System.out.println("Setup Before Test");
    }

    @Test
    public void testOne() {
        System.out.println("Executing Test One");
    }

    @Test
    public void testTwo() {
        System.out.println("Executing Test Two");
    }

    @AfterTest
    public void tearDown() {
        System.out.println("Cleanup After Test");
    }
}
```

---

3 Introduction to BDD & Cucumber

📌 What is BDD (Behavior-Driven Development)?

✓ Helps make test cases **human-readable**

✓ Uses **Cucumber & Gherkin**

📌 What is Cucumber?

Cucumber is a BDD tool that runs **Gherkin syntax-based test cases**.

📌 What is Gherkin?

Gherkin uses simple **Given-When-Then** statements to define scenarios.

---

4 Implementing BDD using Cucumber

📌 Structure of Cucumber Framework

📂 **Project Folder**

├── 📂 src/test/java
├── 📂 features *(Contains .feature files)*
├── 📂 stepDefinitions *(Contains Java step definitions)*
├── 📂 runners *(Contains Test Runner file)*

📌 4.1 Writing a Feature File

Create a file: **Login.feature**

```
Feature: Login functionality

  Scenario: Successful Login
    Given User is on login page
    When User enters valid username and password
    Then User should be redirected to home page
```

📌 4.2 Writing Step Definitions

```java
import io.cucumber.java.en.*;

public class LoginSteps {
    @Given("User is on login page")
    public void user_is_on_login_page() {
        System.out.println("User navigates to login page");
    }

    @When("User enters valid username and password")
    public void user_enters_valid_credentials() {
        System.out.println("User enters login credentials");
    }

    @Then("User should be redirected to home page")
    public void user_should_be_redirected() {
        System.out.println("User is on home page");
    }
}
```

📌 4.3 Running Cucumber Tests with TestNG

```
import io.cucumber.testng.CucumberOptions;
import io.cucumber.testng.AbstractTestNGCucumberTests;

@CucumberOptions(
    features = "src/test/resources/features",
    glue = "stepDefinitions"
)
public class TestRunner extends AbstractTestNGCucumberTests {
}
```

---

5 Day 4 Exercises

1 Write a TestNG test case with @BeforeTest and @AfterTest
2 Implement Soft Assertions in TestNG
3 Create a Cucumber test for login functionality
4 Write a Feature File for "Add to Cart" scenario
5 Implement Step Definitions for "Add to Cart" scenario

---

6 Best Practices

✓ Use Soft Assertions for multiple validations in one test case
✓ Follow the Given-When-Then format strictly in BDD
✓ Keep Step Definitions short & reusable
✓ Use TestNG XML for better test execution control

---

7 Possible Interview Questions & Evaluation Topics

📌 Conceptual Questions

- What are Assertions in Selenium?
- What is the difference between Hard and Soft Assertions?
- Explain TestNG Annotations.
- What is BDD and why is it useful?
- How do you run a Cucumber test with TestNG?

📌 Coding Questions

- Write a TestNG test case with multiple test methods.
- Write a Cucumber test for a login scenario.
- Implement Hard and Soft Assertions in TestNG.

---

8 Summary of Selenium Day 4

✓ **Assertions (Hard & Soft) using TestNG**
✓ **TestNG Framework, Annotations, and Execution**
✓ **BDD & Cucumber Implementation with Gherkin**
✓ **Best practices, hands-on exercises, and interview questions**
