WebDriver Initialization

Chrome

```
System.setProperty("webdriver.chrome.driver", "path/to/chromedriver");
WebDriver driver = new ChromeDriver();
```

Firefox

```
System.setProperty("webdriver.gecko.driver", "path/to/geckodriver");
WebDriver driver = new FirefoxDriver();
```

Internet Explorer

```
System.setProperty("webdriver.ie.driver", "path/to/IEDriverServer.exe");
WebDriver driver = new InternetExplorerDriver();
```

WebDriver:

This is an interface in Selenium that defines a set of methods for browser automation, such as get(), findElement(), click(), etc

driver:

This is the reference variable of type WebDriver. It will hold the instance of the browser driver you are using.

new ChromeDriver():

This creates a new instance of the ChromeDriver class,

🔍 2. Locating Elements

- **By ID**:

```
driver.findElement(By.id("elementId"));
```

- **By Name**:

```
driver.findElement(By.name("elementName"));
```

- **By Class Name**:

```
driver.findElement(By.className("className"));
```

- **By CSS Selector**:

```
driver.findElement(By.cssSelector("cssSelector"));
```

- **By Link Text**

```
driver.findElement(By.linkText("Link Text"));
```

- **By Partial Link Text**:

```
driver.findElement(By.partialLinkText("Partial Text"));
```

- **By Tag Name**:

```
driver.findElement(By.tagName("tagName"));
```

- **By XPath**:

```
driver.findElement(By.xpath("//tag[@attribute='value']"));
```

□ 3. Browser Navigation

```
driver.get("https://example.com"); // Open URL
driver.navigate().to("https://example.com"); // Navigate to URL
driver.navigate().back(); // Navigate back
driver.navigate().forward(); // Navigate forward
driver.navigate().refresh(); // Refresh page
```

🖱️ 4. Interacting with Web Elements

```
WebElement element = driver.findElement(By.id("elementId"));
element.click(); // Click element
element.sendKeys("text"); // Enter text
element.clear(); // Clear text
String text = element.getText(); // Get text
String attribute = element.getAttribute("attributeName"); // Get attribute value
```

🖼️ 5. Handling Alerts

```
Alert alert = driver.switchTo().alert();
alert.accept(); // Click OK
alert.dismiss(); // Click Cancel
alert.sendKeys("text"); // Enter text
String alertText = alert.getText(); // Get alert text
```

📸 6. Taking Screenshots

```
TakesScreenshot screenshot = (TakesScreenshot) driver;

File srcFile = screenshot.getScreenshotAs(OutputType.FILE);
File destFile = new File("path/to/save/screenshot.png");
FileUtils.copyFile(srcFile, destFile);
```

7. Working with Dropdowns

```
WebElement dropdown = driver.findElement(By.id("dropdownId"));
Select select = new Select(dropdown);
select.selectByVisibleText("Option Text");
select.selectByIndex(1);
select.selectByValue("optionValue");
```

8. Handling Frames

```
driver.switchTo().frame("frameName"); // By name or ID
driver.switchTo().frame(0); // By index
driver.switchTo().frame(driver.findElement(By.id("frameId"))); // By WebElement

// Switch back to the main page
driver.switchTo().defaultContent();
```

9. Handling Multiple Windows

```
// Store the current window handle (main window)
String mainWindow = driver.getWindowHandle();

// Get all open window handles (main + any child windows)
Set<String> allWindows = driver.getWindowHandles();

// Loop through all window handles
for (String window : allWindows) {

    // Check if the window is not the main window
    if (!window.equals(mainWindow)) {

        // Switch focus to the child window
        driver.switchTo().window(window);

        // Perform any actions needed on the child window
        // Example: driver.findElement(By.id("someElement")).click();

        // Close the child window
        driver.close();
    }
}
```

```
// Switch focus back to the main window
driver.switchTo().window(mainWindow);
```

⌛ 10. Waits

Implicit Wait

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

Explicit Wait

```
WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));
wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("elementId")));

// Wait until an element with ID "submitButton" is clickable WebElement
submitButton =
wait.until(ExpectedConditions.elementToBeClickable(By.id("submitButton")));

//Wait until an element with class "result" is visible
WebElement result =
wait.until(ExpectedConditions.visibilityOfElementLocated(By.className("result")));
```

🖱 11. Mouse and Keyboard Actions

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

// 🔹 Hover over an element
actions.moveToElement(element).perform();

// 🔹 Click on an element
actions.click(element).perform();  // Or actions.click().perform();

// 🔹 Double click
actions.doubleClick(element).perform();

// 🔹 Right click (context click)
actions.contextClick(element).perform();

// 🔹 Drag and drop
actions.dragAndDrop(sourceElement, targetElement).perform();

// 🔹 Click and hold (for custom drag)
actions.clickAndHold(Element).perform();
```

```
// ◈ Send keyboard keys
actions.sendKeys(Keys.ENTER).perform(); // Press Enter key

// ◈ Move by offset (used for sliders or precise movements)
actions.moveByOffset(100, 50).click().perform();

// ◈ Key combinations (e.g., Ctrl + A)
actions.keyDown(Keys.CONTROL).sendKeys("a").keyUp(Keys.CONTROL).perform();

◈ Typing into an element (keyboard input)
actions.moveToElement(inputElement).click().sendKeys("Hello").perform();

// ◈ Release all held keys/buttons
actions.release().perform();
```

▯ 12. Handling Stale Element Reference Exception

StaleElementReferenceException occurs in Selenium when a previously located WebElement is no longer valid, typically because the DOM has changed or the element has been refreshed or removed. It means the reference to the element is "stale" and must be re-acquired.

```
WebElement element = driver.findElement(By.id("elementId"));
try {
   element.click();
} catch (StaleElementReferenceException e) {
   element = driver.findElement(By.id("elementId"));
   element.click();
}
```