Selenium – Java Cheat Sheet

Driver Initialization

Chrome	`WebDriver driver = new ChromeDriver();`
Firefox	`WebDriver driver = new FirefoxDriver();`
Edge	`WebDriver driver = new EdgeDriver();`
Safari	`WebDriver driver = new SafariDriver();`

Locating Elements - By:

id:	`driver.findElement(By.id("idValue"));`
name:	`driver.findElement(By.name("nameValue"));`
className:	`driver.findElement(By.className ("classValue"));`
tagName:	`driver.findElement(By.tagName ("html tagName"));`
cssSelector:	`driver.findElement(By.cssSelector("input[type='submit']"));`
xPath:	`driver.findElement(By.xpath("//input[@type='submit']"));`
linkText:	`driver.findElement(By.linkText ("Sale"));`
partialLinkText:	`driver.findElement(By.partialLinkText ("link text"));`

Dynamic xPath:

`//*[@type='submit']`	→ any tag with type submit
`//h2[contains(@id, 'ageCont')]`	→ selects id that contains ageCont value
`(//h2[starts-with(@id, 'u_')])[1]`	→ the first input whose id starts with u_
`//input[ends-with(@id, 'P7')]`	→ selects id that ends with p7
`//h2[@id='page-ent' or @class='nav-flex']`	→ one or the other statement
`//h2[@id='page-ent' and @class='nav-flex']`	→ both statements
`//*[.= 'Sign in']`	→ any tag & attribute just give me the text
`//*[(text() = 'Welcome')]`	→ selects only text
`//*[contains(text(), 'Welcome to')]`	→ selects only text that contains
	→ Use index when there are multiple matches

CSS Selector:

.classValue	→ `By.cssSelector(".form-control")`
#idValue	→ `By.cssSelector("#ageCont")`

Selenium Operations

Launch a Webpage:

```
driver.get("https://www.google.com");
OR driver.navigate().to("https://www.google.com");
```

Click a button:

```
WebElement searchBtn = driver.findElement(By.name("btnK")).click();
OR searchButton.click();
```

Accept an alert pop-up: `driver.switchTo( ).alert( ).accept();`

Print the page title:

```
String title = driver.getTitle(); System.out.println(title);
```

Clear the input field text:

```
WebElement searchInput = driver.findElement(By.name("q"));
searchInput.sendKeys("selenium"); searchInput.clear();
```

Disable a field (set the 'disabled' attribute):

```
JavascriptExecutor javascript = (JavascriptExecutor) driver;
String toDisable = "document.getElementsByName('fname')[0]
    .setAttribute('disabled', '');";
    javascript.executeScript(toDisable);
```

Enable a field (remove the 'disabled' attribute):

```
JavascriptExecutor javascript = (JavascriptExecutor) driver;
String toEnable = "document.getElementsByName('fname')[0]
    .setAttribute(enabled, '');";
    javascript.executeScript(toEnable);
```

Wait Operations

Selenium Dynamic Wait

Implicit wait – global wait:

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

Explicit wait – local wait:

1. Create WebDriverWait object

```
WebDriverWait wait = new WebDriverWait(driver,Duration.ofSeconds(10));
```

2. Use the object to add expected conditions

```
WebElement classABC = wait.until(ExpectedConditions
    .visibilityOfElementLocated(By.cssSelector(".classlocator")));
```

→ better than implicit wait when element is not visible / clickable / displayed

FluentWait – local wait. Is like Explicit wait with more options:

```
Wait<WebDriver> fluentWait = new FluentWait<WebDriver>(driver)
    .withTimeout(Duration.ofSeconds(30))
    .pollingEvery(Duration.ofSeconds(5))//will check every 5 sec
    .ignoring(NoSuchElementException.class); //ignores exception
```

Same as Explicit Wait:

```
WebElement classABC = wait.until(ExpectedConditions
    .visibilityOfElementLocated(By.cssSelector(".classlocator")));
```

ScriptTimeout & PageLoad Timeout:

```
driver.manage().timeouts().scriptTimeout(Duration.ofMinutes(2));
driver.manage().timeouts().pageLoadTimeout(Duration.ofSeconds(10));
```

Java hard wait ->

Sleep: `Thread.sleep(Time in MilliSeconds);`

TestNG Annotations

@Test	the main part of the automation script where we write the business logic we want to automate
@BeforeSuite	runs before executing all test methods in the suite
@BeforeTest	executes before executing all test methods of available classes belonging to that folder
@BeforeClass	executes before the first method of the current class is invoked
@BeforeMethod	executes before each test method runs
@AfterSuite	executes after executing all test methods in the suite
@AfterMethod	executes after executing each test method
@AfterTest	executes after executing all test methods of available classes belonging to that folder
@AfterClass	executes after executing all test methods of the current class

JUnit Annotations

@Test	Represents the method or class as a test block, also accepts parameters.
@Before	The method with this annotation gets executed before all other tests.
@BeforeClass	The method with this annotation gets executed once before class.
@After	The method with this annotation gets executed after all other tests are executed.
@AfterClass	The method with this annotation gets executed once after class.
@Ignore	It is used to ignore certain test statements during execution.
@Disabled	Used to disable the tests from execution, but the corresponding reports of the tests are still generated.

Alerts

Accept an alert: Same as clicking OK of an alert.

```
driver.switchTo().alert().accept();
```

Dissmiss an alert: Same as clicking Cancel of an alert.

```
driver.switchTo().alert().dismiss();
```

Enter text in an alert box:

```
driver.switchTo().alert().sendKeys("Selenium")
```

Retrieve alert text: To get the alert message of the alert.

```
driver.switchTo().alert().getText();
```

Java Faker

Copy Faker dependency into pom.xml file

1. Create a Faker object

```
Faker faker = new Faker();
```

2. generate fake data

```
driver
.findElement(By.name("firstname"))
.sendKeys(faker.name().firstName());
```

OR

String fName = faker.name().firstName();

fake data = mock data →fake ssn, fake name, fake address

iFrame

A page within a page → we must first **switch()** to the iframe. 3 ways:

1. by index: → index start from 0

driver.switchTo().frame(0) will switch the first iframe

2. id/name:

```
driver.switchTo().frame("id or name of the iframe");
```

3. web element (locators):

```
WebElement middleFrame =
driver.findElement(By.xpath("//frame[@name='left']"));
driver.switchTo().frame(middleFrame);
```

→ Switching back to parent / default frame:

To parent frame goes only 1 level up:

- driver.switchTo().parentFrame();

To get back to the main fraim:

- driver.switchTo().defaultContent();

Returns the total number of iframe on a page

- driver.findElements(By.tagName("iframe"));

Actions

Step 1: Create the action object:

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

Step 2: Locate the WebElement you want to work on:

```
WebElement element = driver.findElement(By.id("ID"));
```

Step 3: Perform the action on the WebElement

Right click: actions.contextClick(element).perform();

Hover over: actions.moveToElement(element).perform();

```
actions .sendKeys(Keys.ARROW_DOWN)
        .sendKeys(Keys.ARROW_UP)
        .sendKeys(Keys.PAGE_DOWN)
        .sendKeys(Keys.PAGE_UP)
        .build() //OPTIONAL : recommended with method chains
        .perform(); //MANDATORY
```

keysDown(); → to press and hold a key. Keys mean Shift,Ctrl, Alt keys.

keysUp(); → to release a pressed key after keysDown(), otherwise we may get IllegalArgumentException.

sendKeys(element,"text"); → to type into text box / text area

Selenium Navigators

Navigate to a URL	driver.get("URL")
	OR driver.navigate().to("URL");
Refresh the page	driver.navigate().refresh();
Navigate forward in browser	driver.navigate().forward();
Navigate back in browser	driver.navigate().back();

Drop Down List

Step 1: Locate the dropdown element:

```
WebElement month=driver.findElement(By.id("dropdown"));
```

Step 2: Create Select object and pass the variable to that object:

```
Select selectMonth=new Select(month);
```

Step 3: Select from a dropdown using select object with 3 different ways:

```
selectMonth.selectByIndex(0);
selectMonth.selectByValue("1");
selectMonth.selectByVisibleText("Jan");
```

We can put all dropdown elements in a List<WebElement> using **getOptions();**

```
Select selectOptions = new Select(states);
List<WebElement> options = selectOptions.getOptions();
```

Working with Windows

1. Get the current window handle:

```
String window1Handle = driver.getWindowHandle();
```

2. Get all window handles:

```
Set<String> allWindowHandles = driver.getWindowHandles();
```

3. Switch to a specific window:

```
for (String eachHandle : allWindowHandles){
  if (!eachHandle.equals(window1Handle)){
   driver.switchTo().window(eachHandle);
  }
}
```

OR

```
String windowHandle = driver.getWindowHandle();
driver.switchTo().window(windowHandle);
```

Switch to newly created window:

```
driver.switchTo().newWindow(WindowType.TAB);
driver.switchTo().newWindow(WindowType.WINDOW);
```

Close the current window:

```
driver.close();
```

Set window position:

```
driver.manage().window().setPosition(new Point(0, 0));
```

Maximize window:

```
driver.manage().window().maximize();
```

Minimize window:

```
driver.manage().window().minimize();
```

Fullscreen window:

```
driver.manage().window().fullscreen();
```

Take a Screenshot:

```
import org.apache.commons.io.FileUtils;
File scrFile = ((TakesScreenshot)driver)
     .getScreenshotAs(OutputType.FILE);
FileUtils.copyFile(scrFile, new File("./image.png"));
```

Working with Files

Upload a file:

```
driver.findElement(By.id("upload")).sendKeys("path/to/the/file.txt");
driver.findElement(By.id("file-submit")).submit();
```

Read data from an Excel file:

<Apache dependancy>

→ workbook > worksheet > row > cell

→ Index starts with 0 → e.g. row 1 cell 1 has the index of row 0 cell 0

1. Store file path in a string
```
String path = "resources/Capitals.xlsx";
```
OR File file = new File("resources/Capitals.xlsx");

2. Open the file
```
FileInputStream fileInputStream = new FileInputStream(path);
```
3. Open the workbook using fileInputstream
```
Workbook workbook = WorkbookFactory.create(fileInputStream);
```
4. Open the first worksheet
```
Sheet sheet1 = workbook.getSheet("Sheet1");
```
OR workbook.getSheetAt(0); //ALTERNATIVE

5. Go to first row
```
Row row1 = sheet1.getRow(0);
```
6. Go to first cell on that first row and print
```
Cell cell1 = row1.getCell(0);
```

Read data from a text file using BufferedReader:

```
FileReader reader = new FileReader("MyFile.txt");
BufferedReader bufferedReader = new BufferedReader(reader);
        String line;
        while ((line = bufferedReader.readLine()) != null)
        { System.out.println(line); }
reader.close();
```

Read data from a text file Using InputStream:

```
FileInputStream inputStream = new FileInputStream("MyFile.txt");
InputStreamReader reader = new InputStreamReader(inputStream,
"UTF-16");
int character;
        while ((character = reader.read()) != -1)
        { System.out.print((char) character); }
reader.close();
```

Read data from a text file Using FileReader:

```
FileReader reader = new FileReader("MyFile.txt");
int character;
        while ((character = reader.read()) != -1)
        { System.out.print((char) character); }
reader.close();
```

Read data from a CSV file:

```
import au.com.bytecode.opencsv.CSVReader;
String path = "C:\\Users\\Myuser\\Desktop\\csvtest.csv";
Reader reader = new FileReader(path);
CSVReader csvreader = new CSVReader(reader);
List<String[]> data = csvreader.readAll();
        for(String[] d : data){
        for(String c : d ){
        System.out.println(c); } }
```

Working with Files

We can't test desktop applications with Selenium. But we can use JAVA

```
System.getProperty("user.dir"); =>gives the path of the current folder
System.getProperty("user.home"); =>gives you the user folder
Files.exists(Paths.get("path of the file"));
=>Checks if a file path exists on your computer or not
```

Javascript Executor

1. Creating a reference
```
JavascriptExecutor js = (JavascriptExecutor) driver;
```
2. Calling the method
```
js.exectueScript(Script, Arguments);
js.executeScript(return something);
```
Example: Clicking on a button
```
    WebElement button =driver.findElement(By.name("btnLogin"));
//Perform Click on LOGIN button using JavascriptExecutor
    js.executeScript("arguments[0].click();", button);
//arguments[0] -> the first argument in executeScript method
```

Selenium Grid

Start the hub:
```
java -jar selenium-server-standalone-x.y.z.jar -role hub
```
Start a node:
```
java -jar selenium-server-standalone-x.y.z.jar -role node -hub
```
Server
```
http://localhost:4444/ui/index.html
```