**Java**

* **Sec 1: Introduction to Java**

Java: Object oriented programming language

Types of programming languages:

1. Structured: C, Python.
2. Object based: VB, VBScript, Python.
3. Object oriented: C++, Java, C#, Python.

OOPS:

1. Class
2. Object
3. Polymorphism
4. Inheritance
5. Abstraction
6. Encapsulation

Features:

1. Platform Independent
2. Case sensitive

3 Components:

1. JDK: Java Development Kit
2. JRE: Java Runtime Environment
3. JVM: Java Virtual machine

Environment setup:

1. JDK/java
2. IDE (Eclipse, IntelliJIDE, etc)

Versions:

1. Java 8: Sun microsystem
2. Java 9: Oracle

Java 11+ preferable

Steps:

1. Create a new java project
2. Create a new java package
3. Create a new class

Class naming conventions:

1. Class name should start with Uppercase
2. Class name should not start with number
3. Class name contains \_ (Underscore)
4. Special characters are not allowed
5. Class name contains numbers

* **Sec 2: Java Variables and Data Types**

Variables & Data types:

Variables is a container which can hold data. To represent data we need variables.

int x=100

float itemprice=10.25

int age =30

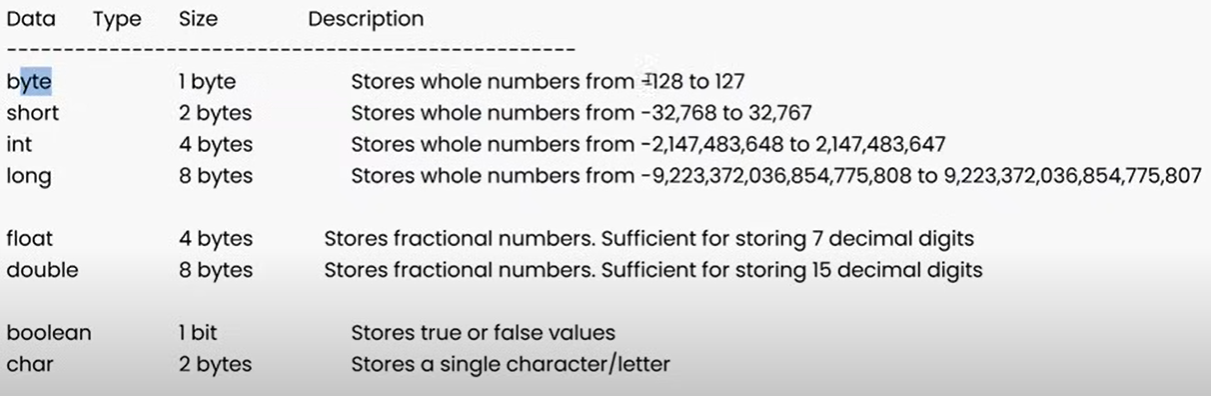
String name=”John”

char grad=’A’

Data Types:

Represents type of data

1. Premitive:
2. byte, short, int, long: number without decimal
3. float, double: decimal number
4. char: single character (single quote)
5. boolean: true/false
6. Non-premitive/derived/collections:
7. String
8. ArrayList
9. HashMap
10. HashSet



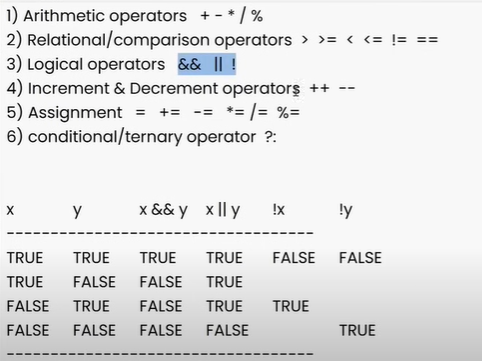
Java is a statically typed programming language

Python is a dynamically typed programming language

* **Sec 3: Java Operators and Expressions:**

Operators:

1. Arithmetic operators



**Selenium WebDriver**

**Sec 21: Selenium Introduction & Environment Setup**

Selenium WebDriver

* What is Selenium WebDriver?

1. WebDriver is one of the components in selenium
2. WebDriver is a java interface
3. WebDriver is an API (Application Programming Interface)

* WebDriver (Interface: I) – RemoteWebDriver (Compiler: C) -> ChromeDriver, FirefoxDriver, EdgeDriver, etc.
* Environment setup:

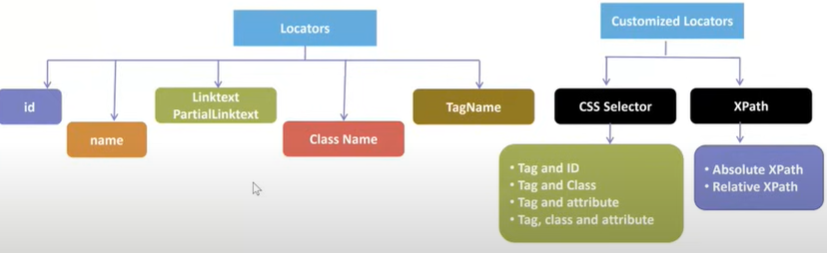
1. Download jar files and attaching them to Java project (Manually).
2. Create new java project.
3. Download webdriver jars(.zip) and extracted.
4. Attach jars to java project.
5. Create Maven project.
6. Create a new Maven project in eclipse.
7. Add webdriver dependency in pom.xml -> update.

Stable: 4.18.1 (February 19, 2024).

pom.xml ----🡪dependencies <https://mvnrepository.com/>

**Sec 22: Selenium Locators**

Locators:



Link Text is preferred over partial link text for identifying/locating element.

Ex: 1) Tablets Table – linkText()

1. Submit Send – partialLinkText()

Locators:

1. Id
2. Name
3. linkText
4. partialLinkText
5. TagName: used for group of webelements
6. Class: used for group of webelements

* findElement() ---- single webelement
* findElements() ---- multiple webelements

Scenario 1: Locator matching with single web element

* findElement(loc) --🡪 single web element: Return Type: WebElement
* findElements(loc) -🡪 single web element: Return Type: List<WebElement>

Scenario 2: Locator matching with multiple web element

* findElement(loc) --🡪 single web element: Return Type: WebElement
* findElements(loc) -🡪 multiple web element: Return Type: List<WebElement>

Scenario 3: Locator is not matching with any element/s

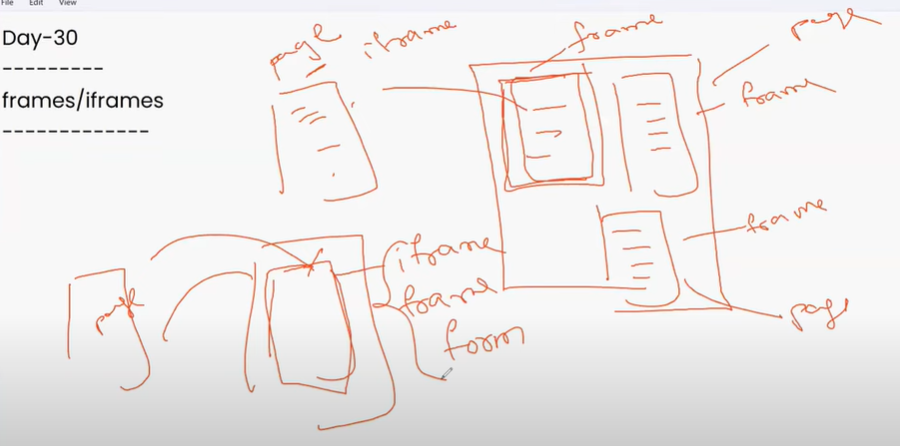
* findElement(loc) --🡪 Exception Returned: No Such Element Exception.
* findElements(loc) -🡪 Will not throw any Exception. Returns 0.

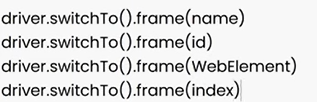
**Sec 30: Selenium Handling Frames/iFrames & Nested iFrames**

frames/iframes

id/name/webelement

driver.SwitchTo().frome()





**Sec 31: Selenium Handling Dropdown**

* Dropdown box:

1. Select dropdown
2. Bootstrap dropdown
3. Hidden dropdown

* Select Dropdown

Select:

1. selectByVisibleText() : visible text means visible value of text.
2. selectByvalue() : means attribute value
3. selectByIndex()

Ex:

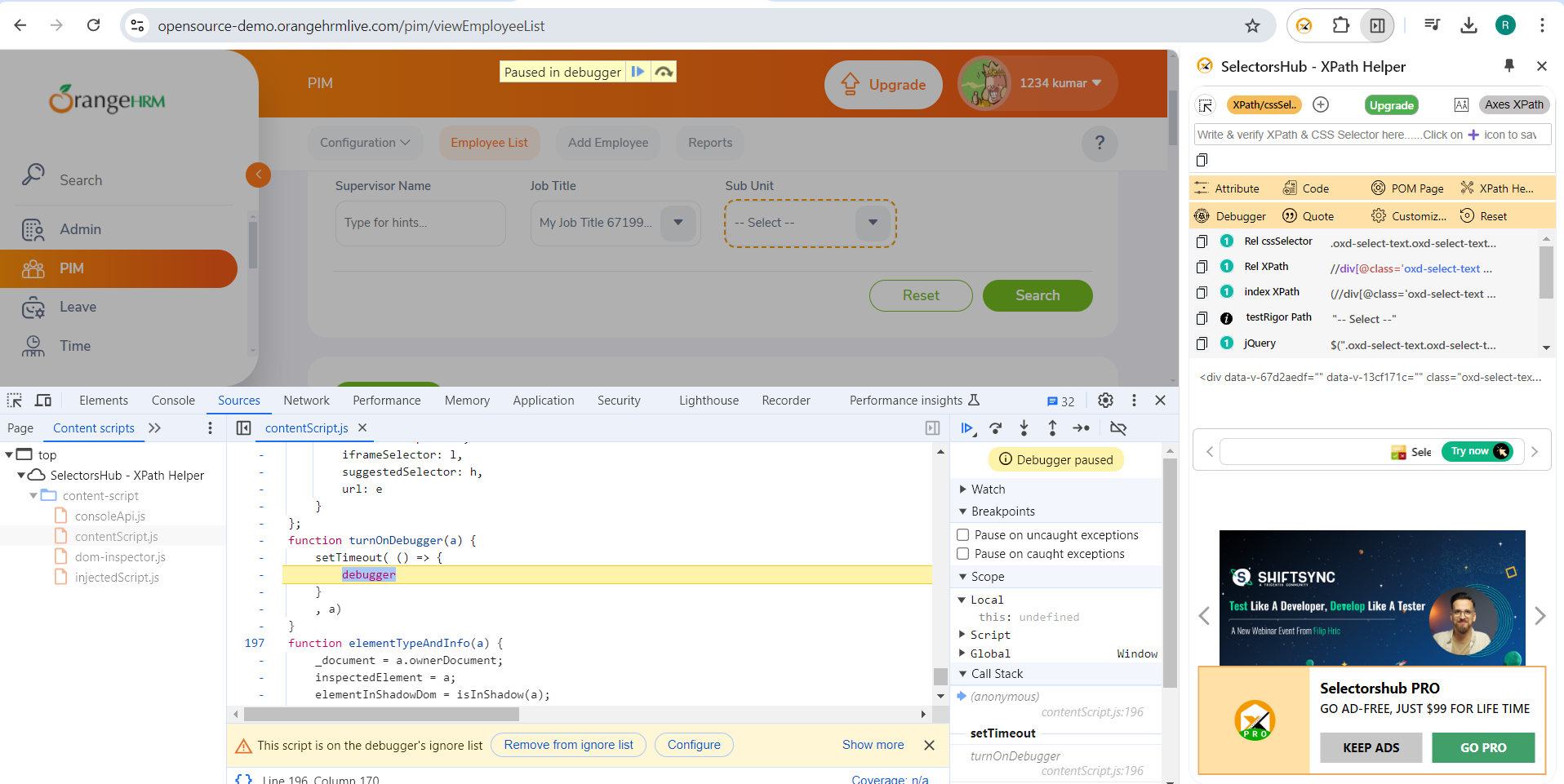
WebElement drpCountryEle = driver.findElement(By.*xpath*("//select[@id='country']"));

Select drpCountry = **new** Select(drpCountryEle);

* getOptions() : returns all the options from the dropdown as a WebElement.
* Bootstrap dropdown

Hidden Dropdowns:

1. Extension: Selector Hub Chrome Extension <https://chromewebstore.google.com/detail/selectorshub-xpath-helper/ndgimibanhlabgdgjcpbbndiehljcpfh?hl=en&pli=1>



**Sec 32: Selenium Handling Auto-suggest Dropdown & Static Web Table**

* Auto – Suggest Dropdown:

Ex:

**package** day33SeleniumHandlingAutoSuggestDropDownStaticWebTable;

**import** java.time.Duration;

**import** java.util.List;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.WebElement;

**import** org.openqa.selenium.chrome.ChromeDriver;

**public** **class** GoogleSearch\_AutoSuggestDropDown {

**public** **static** **void** main(String[] args) **throws** InterruptedException {

WebDriver driver = **new** ChromeDriver();

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

driver.get("https://www.google.com/");

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

driver.findElement(By.*name*("q")).sendKeys("selenium"); //Serch Box

Thread.*sleep*(5000);

List<WebElement> list = driver.findElements(By.*xpath*("//ul[@role='listbox']//li//div[@role='option']"));

System.***out***.println(list.size());

**for**(**int** i=0; i<list.size(); i++) {

System.***out***.println(list.get(i).getText());

**if**(list.get(i).getText().equals("selenium")) {

list.get(i).click();

**break**;

}

}

}

}

* Web Tables:

1. **Static Web Table:** Fixed no. of rows/columns and data is also static i.e. it does not change any time.

* Use xpath: Multiple tables in a webpage.

Ex: **int** rows = driver.findElements(By.*xpath*("//table[@name='BookTable']//tr")).size();

* Use tagName: Single table in a webpage. (Not recommended for Multiple tables)

Ex: **int** rows = driver.findElements(By.*tagName*("tr")).size();

* Syntax of xpath for dyanamic value(To pass variable as parameters): Keep variable in double quotes and pass add sign and pass the value. Ex: //table[@name='BookTable']//tr["+r+"]//td["+c+"]

1. **Dynamic Web table**: Data keeps changing, no. of rows keep adding/reduced. No. of columns will be fixed but no. of rows will change. Ex: <https://opensource-demo.orangehrmlive.com/admin/viewSystemUsers>.
2. **Table with Pagination:** Ex: table contains 100 records but first page contains 10 records. Ex: <https://demo.opencart.com/admin/index.php?route=customer/customer&user_token=c26ea660109e97bcefc2836432dc67e3>**.**

**Sec 33: Selenium Handling Dynamic Pagination Web Table**

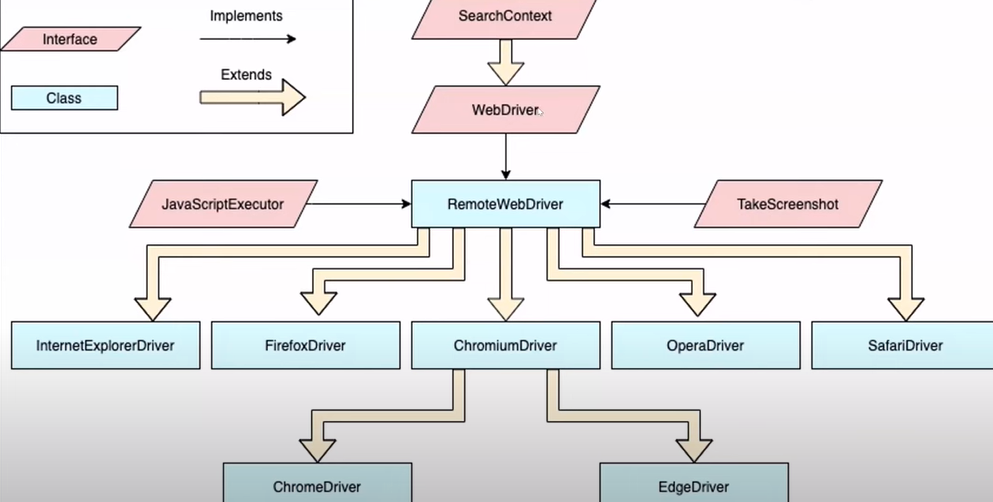
Dynamic table with pagination

Ex: <https://demo.opencart.com/admin/index.php?route=customer/customer&user_token=f0effd7f9d97b23793096bc1c23523df>.

String s = “Showing 1 to 10 of 6804 (681 Pages)”

s.substring(s.indexOf(“(“)+1, s.indexOf(“Pages”)-1) -🡪 681

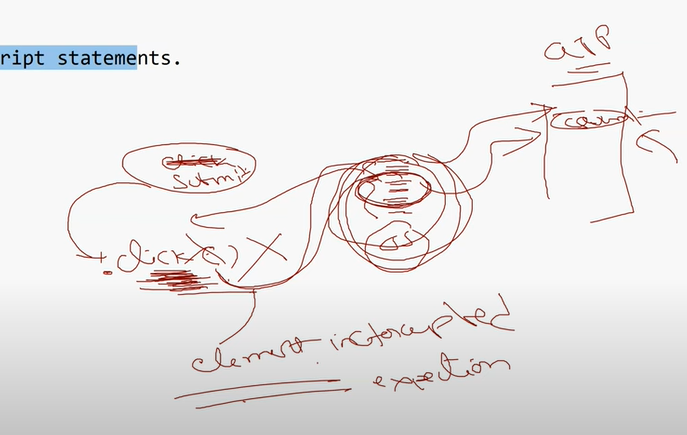
**Sec 37: Selenium JavascriptExecutor Scrolling Pages Upload Files**



* JavascriptExecutor:

Methods: executeScript(),

1. executeScript() – we can execute javascript statements.
2. Element intercepted exception – when we call any method from webdriver internally execute javascript statements which talks with browsers. But sometimes this javascript statements are not able to talk internally so this will throw an exception i.e. ‘**element intercepted exception’**.



1. exexuteScript() method is used in place of sendKeys(), click() methods.

1. WebDriver driver = new ChromeDriver();  //upcasting

JavascriptExecutor js = (JavascriptExecutor)driver;

2. ChromeDriver driver = new ChromeDriver();

JavascriptExecutor js = driver; // no need of typecasting we can directly assign it to driver

1. Alternate of sendkeys method:

Using get & setAttribute

Ex:

        js.executeScript("arguments[0].setAttribute('value','John')", inputbox);

1. Scrolling Bar:

Ex:

**package** day37SeleniumJavascriptExecutorScrollingPagesUploadFiles;

**import** java.time.Duration;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.JavascriptExecutor;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.WebElement;

**import** org.openqa.selenium.chrome.ChromeDriver;

**public** **class** ScrollingPage {

**public** **static** **void** main(String[] args) **throws** InterruptedException {

WebDriver driver = **new** ChromeDriver();

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

driver.get("https://demo.nopcommerce.com/");//driver.get("https://www.countries-ofthe-world.com/flags-of-the-world.html");

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

JavascriptExecutor js = (JavascriptExecutor)driver;

//1) scroll down page by pixel number

/\* js.executeScript("window.scrollBy(0,1500)", ""); //js.executeScript("window.scrollBy(0,3000)", "");

System.out.println(js.executeScript("return window.pageYOffset;")); //1500

//For horizontal off set use : pageXOffset

\*/

//2) scroll the page till element is visible

/\*

WebElement ele = driver.findElement(By.xpath("//strong[normalize-space()='Community poll']"));

js.executeScript("arguments[0].scrollIntoView();", ele);

System.out.println(js.executeScript("return window.pageYOffset;")); //2103.428466796875

//For horizontal off set use : pageXOffset

\*/

//3) scroll page till end of the page

js.executeScript("window.scrollBy(0,document.body.scrollHeight)");

//For horizontal offset use : scrollWidth in place of scrollHeight

System.***out***.println(js.executeScript("return window.pageYOffset;")); //2103.428466796875

//For horizontal off set use : pageXOffset

Thread.*sleep*(5000);

//scrolling up to initial position

js.executeScript("window.scrollBy(0, -document.body.scrollHeight)");

//For horizontal offset use : scrollWidth in place of scrollHeight

}

}

1. Zoom In Zoom Out

JavascriptExecutor js = (JavascriptExecutor)driver;

js.executeScript("document.body.style.zoom='50%'"); // here document refers to webpage; //set zoom level 50%

1. Upload Files:

Single File:

driver.findElement(By.xpath("//input[@id='filesToUpload']")).sendKeys("D:\\Automation\\Selenium BDD\\Selenium-Java\\Test1.txt");

        if(driver.findElement(By.xpath("//ul[@id='fileList']//li")).getText().equals("Test1.txt")) {

            System.out.println("File is successfully uploaded");

        }

Multiple Files:

//multiple file upload

        String file1 = "D:\\Automation\\Selenium BDD\\Selenium-Java\\Test1.txt";

        String file2 = "D:\\Automation\\Selenium BDD\\Selenium-Java\\Test2.txt";

        driver.findElement(By.xpath("//input[@id='filesToUpload']")).sendKeys(file1+"\n"+file2);

**int** noOfFilesUploaded = driver.findElements(By.*xpath*("//ul[@id='fileList']//li")).size();

**if**(noOfFilesUploaded==2) {

System.***out***.println("All Files are uploaded");

}

**else** {

System.***out***.println("Files are not uploaded or incorrect files uploaded");

}

//validate file names

**if**(driver.findElement(By.*xpath*("//ul[@id='fileList']//li[1]")).getText().equals("Test1.txt")

&& driver.findElement(By.*xpath*("//ul[@id='fileList']//li[2]")).getText().equals("Test2.txt")) {

System.***out***.println("File names are matching...");

}

**else** {

System.***out***.println("Files name are not matching");

}

Ex:

**package** day37SeleniumJavascriptExecutorScrollingPagesUploadFiles;

**import** java.time.Duration;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.chrome.ChromeDriver;

**import** io.opentelemetry.exporter.logging.SystemOutLogRecordExporter;

**public** **class** FileUpload {

**public** **static** **void** main(String[] args) {

WebDriver driver = **new** ChromeDriver();

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

driver.get("https://davidwalsh.name/demo/multiple-file-upload.php");

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

//single file upload - Test1.txt

/\* driver.findElement(By.xpath("//input[@id='filesToUpload']")).sendKeys("D:\\Automation\\Selenium BDD\\Selenium-Java\\Test1.txt");

if(driver.findElement(By.xpath("//ul[@id='fileList']//li")).getText().equals("Test1.txt")) {

System.out.println("File is successfully uploaded");

}

else {

System.out.println("Upload failed");

}

\*/

//multiple file upload

String file1 = "D:\\Automation\\Selenium BDD\\Selenium-Java\\Test1.txt";

String file2 = "D:\\Automation\\Selenium BDD\\Selenium-Java\\Test2.txt";

driver.findElement(By.*xpath*("//input[@id='filesToUpload']")).sendKeys(file1+"\n"+file2);

**int** noOfFilesUploaded = driver.findElements(By.*xpath*("//ul[@id='fileList']//li")).size();

**if**(noOfFilesUploaded==2) {

System.***out***.println("All Files are uploaded");

}

**else** {

System.***out***.println("Files are not uploaded or incorrect files uploaded");

}

//validate file names

**if**(driver.findElement(By.*xpath*("//ul[@id='fileList']//li[1]")).getText().equals("Test1.txt")

&& driver.findElement(By.*xpath*("//ul[@id='fileList']//li[2]")).getText().equals("Test2.txt")) {

System.***out***.println("File names are matching...");

}

**else** {

System.***out***.println("Files name are not matching");

}

}

}

**Sec 38: Selenium Screenshots Headless SSL Ad Block Extensions**

* How to capture screenshots:

1. Full page
2. Specific area of the page
3. Web Element

* ChromeOptions, EdgeOptions, FirefoxOptions.
* Headless mode (without UI): ChomeOptions class

**ChromeOptions:**

**1) Headless testing**

1) Headless Testing:

ChromeOptions options = new ChromeOptions();

        options.addArguments("--headless=new");  // setting for headless mode of execution

        WebDriver driver = new ChromeDriver(options);

Advantages:

1. We can perform multiple tasks (since execution happed backed).
2. Faster execution.

Disadvantages:

1. User can not see the actions on the page. So, he can not understand flow/functionality of the test.
2. **SSL Handling:**

ChromeOptions options = new ChromeOptions();

        options.setAcceptInsecureCerts(true); // accepts SSL certificates

        WebDriver driver = new ChromeDriver(options);

**Ex:**

**public** **class** HandleSSL {

**public** **static** **void** main(String[] args) {

ChromeOptions options = **new** ChromeOptions();

options.setAcceptInsecureCerts(**true**); // accepts SSL certificates

WebDriver driver = **new** ChromeDriver(options);

driver.get("https://expired.badssl.com/");

System.***out***.println("title of the page: "+driver.getTitle()); //Privacy error

}

}

1. Remove "Chrome" is controlled by automated test software:

3) Remove "Chrome" is controlled by automated test software:

ChromeOptions options = new ChromeOptions();

        options.setExperimentalOption("excludeSwitches", new String[] {"enable-automation"});

        WebDriver driver = new ChromeDriver(options);

1. To run test in incognito mode

To run test in incognito mode

ChromeOptions options = new ChromeOptions();

        options.addArguments("--incognito");

1. Enable extensions in browser

Seteps:

1. Add CRX Extractor/Downloader to chrome browser (manually). (<https://chromewebstore.google.com/detail/crx-extractordownloader/ajkhmmldknmfjnmeedkbkkojgobmljda> )
2. Add SelectorHub plugin to chrome browser (manually).
3. Capture crx file for selectorshub extension.
4. Pass crx file path in automaton script in chrome options.
5. Enable browser extensions in run time:

ChromeOptions options = **new** ChromeOptions();

File file = **new** File("D:\\Automation\\Selenium BDD\\Selenium-Java\\seleniumWebDriver\\crx files\\uBlock-Origin.crx");

options.addExtensions(file);

1. To block all ads/advertisement

**‘Ublock Origin’** Extension: <https://chromewebstore.google.com/detail/ublock-origin/cjpalhdlnbpafiamejdnhcphjbkeiagm>

1. **UBlock-Origin**
2. **AdBlocker**

**Sec 39: Selenium Handling Broken Links SVG Elements Shadow DOM**

**Broken Links:**

* If the status code is less than 400 then it is not broken.
* If the status code is above 400 then it is broken.

Steps:

1. Link href = ” <https://xyz.com>”
2. <https://xyz.com> ---🡪 Server -----🡪 Status code
3. Status code => 400 broken link

Status code < 400 not a broken link

**Broken Link is a link which do not have any resource in a server.**

Ex:

**package** day39SeleniumHandlingBrokenLinksSVGElementsShadowDOM;

**import** java.io.IOException;

**import** java.net.HttpURLConnection;

**import** java.net.URL;

**import** java.time.Duration;

**import** java.util.List;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.WebElement;

**import** org.openqa.selenium.chrome.ChromeDriver;

/\*

Steps:

1) Link href = ” https://xyz.com”

2) https://xyz.com ---◊ Server -----◊ Status code

3) Status code => 400 broken link

Status code < 400 not a broken link

\*/

**public** **class** BrokenLinks {

**public** **static** **void** main(String[] args) **throws** IOException {

WebDriver driver = **new** ChromeDriver();

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

driver.get("http://www.deadlinkcity.com/");

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

//capture all the links from website

List<WebElement> links = driver.findElements(By.*tagName*("a"));

System.***out***.println("Total number of links: "+ links.size());

**int** noOfBrokenLinks = 0;

//1) Link href = ” https://xyz.com”

**for**(WebElement linkElement:links) {

String hrefattValue = linkElement.getAttribute("href");

**if**(hrefattValue==**null** || hrefattValue.isEmpty()) {

System.***out***.println("href attribute value is null or empty. So not possible to check");

**continue**; // continue will skip the current value

}

//2) https://xyz.com ---◊ Server -----◊ Status code

//hit url to the server

**try** {

URL linkURL = **new** URL(hrefattValue); // converted href value from string to URL format

HttpURLConnection conn = (HttpURLConnection) linkURL.openConnection(); //opens the connection to the server

conn.connect(); //connect to server and send request to the server

**if**(conn.getResponseCode()>=400) {

System.***out***.println(hrefattValue + " ===> Broken link");

noOfBrokenLinks++;

}

**else** {

System.***out***.println(hrefattValue+" ===> Not a Broken link");

}

}

**catch**(Exception e) {

}

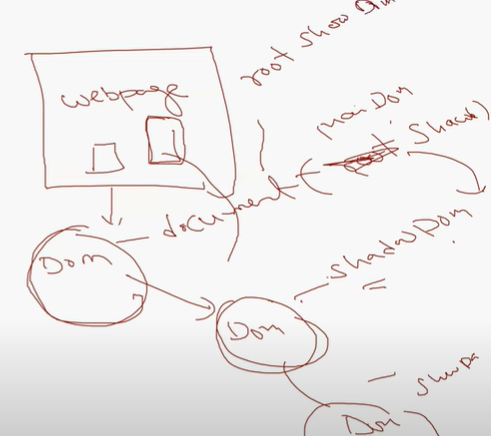
}

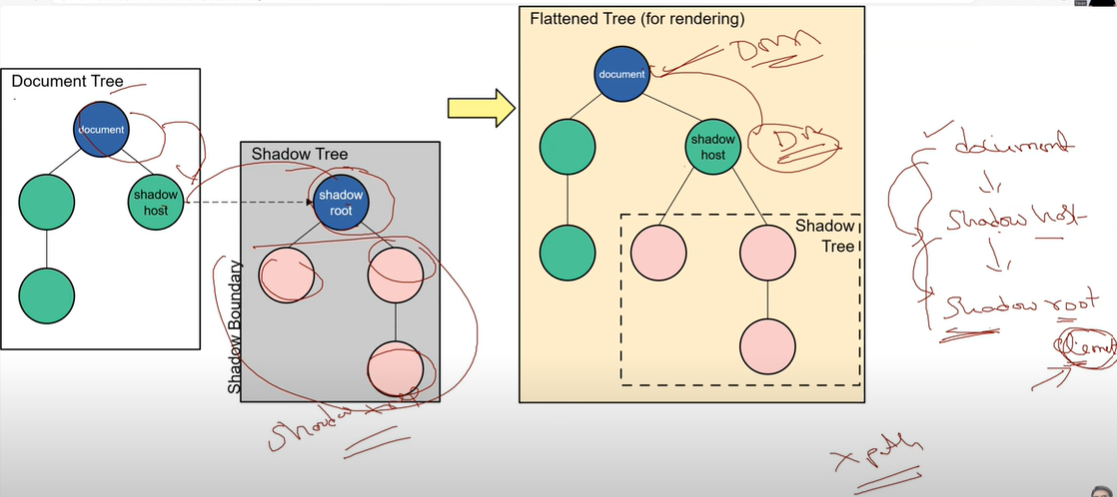
System.***out***.println("Number of broken links: "+ noOfBrokenLinks);

}

}

**Shadow DOM:**





day39SeleniumHandlingBrokenLinksSVGElementsShadowDOM

1. .

**Sec 42: Selenium in Java | TestNG Integration & Setup | @Test annotation**

TestNG – Test New Generation

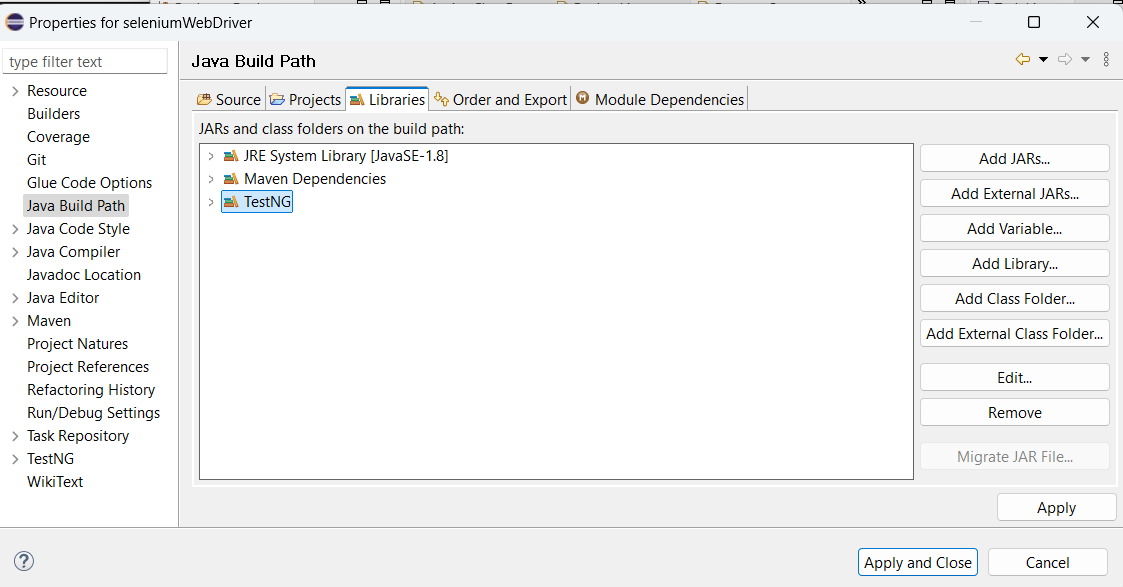
Java based unit testing tool

Advantages:

1. Test cases & test suites
2. Grouping of test cases
3. Prioritize
4. Paramerterization
5. Parallel testing
6. Reports

TestNG configuration

1. Eclipse: Window -> Preferences -> check TestNG is present
2. To install TestNG (if not present): Eclipse Marketplace -> TestNG for Eclipse
3. To add/configure TestNG library to current project: Current Project -> Properties -> Libraries -> Add TestNG -> Apply and Close



1. Other way to add TestNG is by adding dependencies in pom.xml => <https://mvnrepository.com/artifact/org.testng/testng/7.10.2>

<!-- https://mvnrepository.com/artifact/org.testng/testng -->

<dependency>

<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

<version>7.10.2</version>

<scope>test</scope>

</dependency>

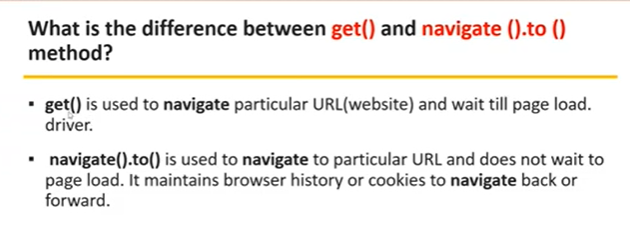
* Install testng in eclipse
* Add testng library to build path / add testng dependency in pom.xml

1. **’@Test’ annotation needs to be written before every test to implement it**.

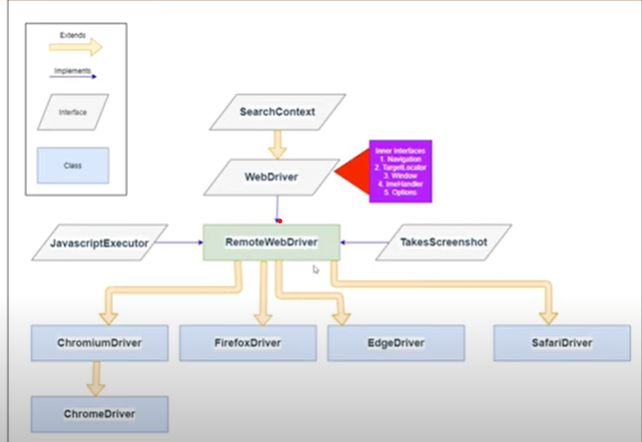
* .

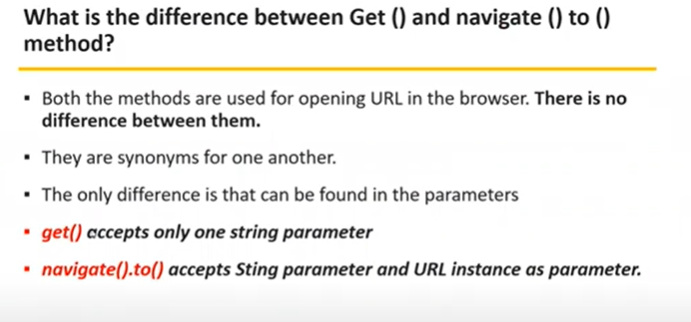
///////////////////////////////////////////////////////////////////////////////////////////////

Selenium Interview Questions

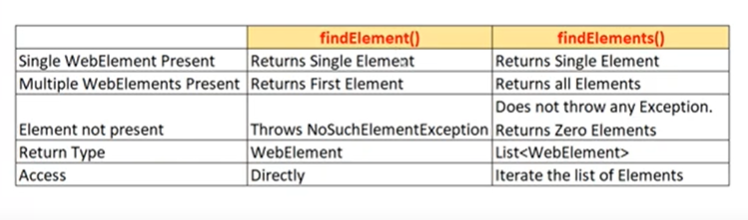


Selenium Hierarchy:



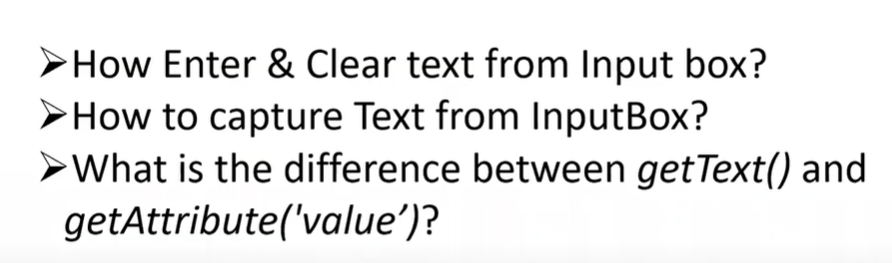


* What is difference between findElement and findElements in Selenium WebDriver?

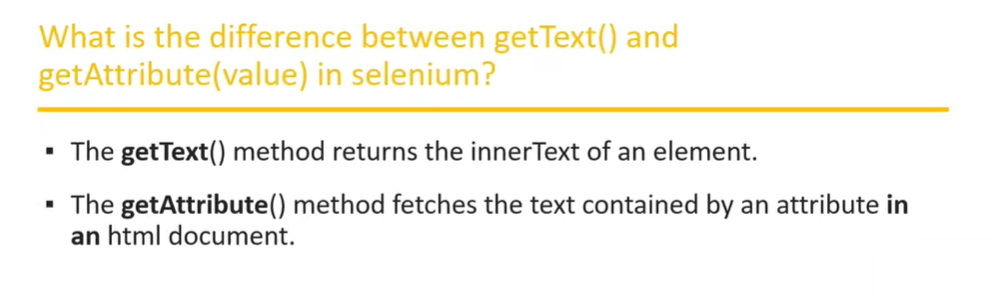




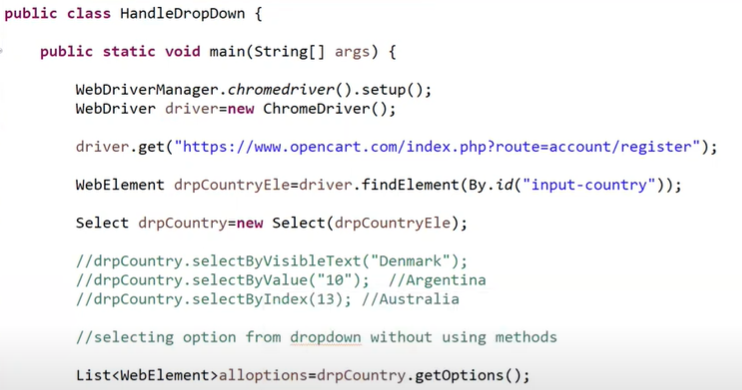
* GetText() and getAttribute() difference?

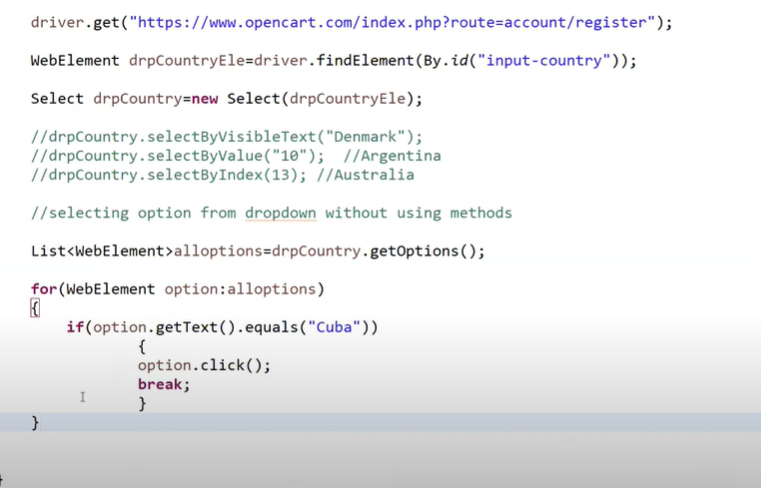






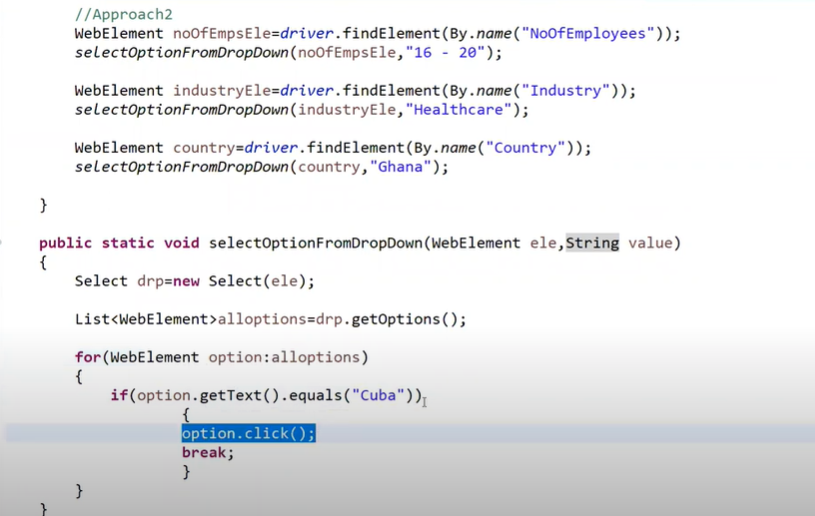
* How to handle Dropdown in Selenium?







Using user define methods:



* .