|  |  |
| --- | --- |
| WebElement user = driver.findElement(By.xpath(""));  Actions actions = new Actions(driver);  actions.moveToElement(user).click().build().perform(); | **MOUSEHOVER** |
| actions.dragAndDrop(source\_el, target\_el).build().perform(); | DRAG & DROP |
| actions.doubleClick(button).build().perform(); | DOUBLE CLICK |
| actions.contextClick(button).build().perform(); | RIGHT CLICK |
| actions.moveToElement(slider).dragAndDropBy(SLIDER, 400, 0).build().perform(); | |
| actions.moveToElement(element).dragAndDropBy(element, 200, 150).build().perform(); | |
| actions.sendKeys(Keys.ENTER). build().perform(); | ENTER KEY |
| actions.sendKeys(Keys.CONTROL+"a").build().perform(); | MULTIPLE KEY |
| actions.keyDown(element, Keys.SHIFT);  actions.sendKeys("TextToBeConvertAndSendInUpperCase");  actions.keyUp(Keys.SHIFT); | PRESS & RELEASE SHIFT KEY |
| WebElement username = wait.until(ExpectedConditions. visibilityOfElementLocated(By.name("username")));  username.sendKeys("mercury"); 🡺🡺 elementToBeClickable; elementToBeSelected; invisibilityOfAllElements | EXPLICIT  WAIT  Is an Interface |
| WebElement drop = driver.findElement(By. xpath(" ");  Select dropdown = new Select(drop); //Handling Drop Down  List<WebElement> options = dropdown.getOptions(); //get all values  for (WebElement e : options) {SOUT(e.getText()):}  dropdown.selectByVisibleText(); selectByIndex(); selectByValue(); | DROP DOWN |
| List<WebElement> links = driver.findElements(By.tagName("a"));  System.out.println(links.size());  for (WebElement e: links) {System.out.println(e. getText());} | COUNT ALL LINK IN PAGE |
| JavascriptExecutor js = (JavascriptExecutor) driver; //INTERFACE  js.executeScript("window.scrollBy(0,4000)", ""); //BY PIXEL  WebElement flag = driver.findElement(By.xpath("")); //till we found the elem  js.executeScript("arguments[0].scrollIntoView()", flag); //at the end of the  js.executeScript("window.scrollBy(0,document.body.scrollHeight)"); | SCROLL DOWN  JS gives 02 methods  JSE |
| driver.findElement(By.id("hide-textbox")).click(); // Hide button is clicked  go for xpath and click console and write document.getElementById  js.executeScript("document.getElementById('displayed-text').value='Nurul Mahmud';"); //ElementNotVisible or Interactable | HIDDEN ELEMENT NOT HANDLE |
| String title= (String) js.executeScript("return document.title"); SOUT (title) | READ VAR |
| Converting the driver to the screenshot object and Call getScreenshotAs method to create image file  File screenshotFile = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE); Interface  FileUtils.copyFile(screenshotFile, new File(".//screenshot/"+screenshotfilename+".png")); | |
| 1. String defaultWindow = driver.getWindowHandle(); 4. itr.next();//first window handle  2. Set<String> allWindows= driver.getWindowHandles(); 5. String childWindow = itr.next(); //2nd window  3. Iterator<String> itr = allWindows.iterator(); 6.driver.switchTo().window(childWindow); | |
| ChromeOptions options = new ChromeOptions(); //CLASS  options.setHeadless(true); OR options.addArguments("--headless");  WebDriver driver = new ChromeDriver(options); HtmlUnitDriver driver = new HtmlUnitDriver ();  driver.get("http//tutorialsninja.com/demo/"); System.out.println(driver.getTitle()); | |
| String projectPath= System.getProperty("user.dir");  driver.findElement(By.id("uploadfile")).sendKeys(projectPath+"Files\\s1.png"); | FILE UPLOAD |
| driver.switchTo().alert().accept().dismiss().getText().sendKeys("Text"); ALERT | |
| driver.manage().timeouts().pageLoadTimeout(40, TimeUnit.SECONDS); TimeOutExceptions  driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS); NoSuchElementException | |
| driver.switchTo().frame("packageListFrame"); driver.switchTo().defaultContent(); | |
| NoSuchElementException, ElementNotVisibleExeception, ElementNotSelectable, TimeOutExc,  StaleElementReference, NoSuchWindow, NoAlertPresent, InvalidSelector, NoSuchFrame | |

|  |  |
| --- | --- |
| If we get 4xx or 5xx HTTP Status codes in the response from the Server, when we hit any URL, then they are BROKEN links.  List<WebElement> links = driver.findElements(By.tagName("a"));  for (WebElement link : links) { String url = link.getAttribute("href");  if (url==null || url.isEmpty()) {continue;}  HttpURLConnection huc= (HttpURLConnection)(new URL(url).openConnection());  huc.connect(); if (huc.getResponseCode()>=400) { | |
| Stale element means an old element or no longer available element. If the DOM changes then the Web Element goes stale. If we try to interact which is staled then the StaleElementReferenceException thrown  WebElement link = driver.findElement(By.xpath("")); We need to refresh our  link.click(); driver.navigate().back(); Thread.sleep(4000); reference/element  try {link.click(); } catch (StaleElementReferenceException e) {  link = driver.findElement(By.xpath("")); link.click(); } | |
|  | |
| String actual\_text = driver.findElement(By.xpath("")).getText();  String expected\_text = "American Express® Gold Card";  Assert.assertEquals(actual\_text, expected\_text, "Test Failed"); | Assert.assertTrue()  Assert.assertFalse()  Assert.fail().NotEquals |
| @DataProvider(name = "users")  String [] [] loginData(){  String [] [] data = {{"ABC","ABC"}, {"XYZ","XYZ"}, {"MON","MON"}}; return data;}  @Test(dataProvider = "users") //dataProvider is a parameter of @Test Annotation  void loginTest(String username, String password) throws InterruptedException {  driver.findElement(By.id("txtUsername")).sendKeys("Admin");  driver.findElement(By.id("txtPassword")).sendKeys("admin123");} | |
| CROSS BROWSER TESTNG | |
| @Test(priority = 2, dependsOnMethods = {"openUrl"}, enabled = false) | |
| Extent Report with cucumber: cucumber-extents report; com.relevantodes; com.avenstack | |
| Pass parameter in TestNG through XML File and through method-@DataProvider. Parallel testing is achieved through parameterization. Execute test case in multiple browsers. | |
| **ITestListener-interface; IReporter; ISuiteListener; IConfigurationListener; IConfigurableListener** | |
| **Have priorities/sequence (which should be execute first or which should be execute last). Have dependency means one test case is depend on another test case. Grouping (suppose you have 100 test case; you can group the test cases). Most important feature is data provider feature @DataProvider. @Test(invocationCount=10). 10 times** | |