1. **IsElementPresent/Text Present  function in Selenium WebDriver**
2. Finding elements by using function that take argument of By classprivate boolean isElementPresent(WebDriver driver, By by)  
   try{  
   driver.findElement(by);  
   return true;  
   }  
   catch(Exception e)  
   {  
   return false;  
   }  
   }

B. Using the size to decide whether element is there or not  
if(driver.findElements(Locator).size()>0  
{  
return true  
}else  
{  
return false  
}  
}

1. Finding the text using the **PageSource**driver.PageSource.Contains("TEXT that you want to see on the page");
2. **Finding WebElement  by using various locators in WebDriver**
   1. **Using ID**  WebElement welement = driver.findElement(By.id("Id from webpage"));
   2. **Using Name**  WebElement welement = driver.findElement(By.name("Name of WebElement"));
   3. **Using Tag Name**  WebElement welement = driver.findElement(By.tagName("tag name"));
   4. **Using Xpath**  WebElement welement = driver.findElement(By.xpath("xpath of  webElement"));
   5. **Using CSS**  WebElement welement = driver.findElement(By.CSS("CSS locator path"));
   6. **Using LinkText**  WebElement welement = driver.findElement(By.LinkText("LinkText"));
3. **Fetching pop-up message in Selenium-WebDriver**  
   this is the function that would help you in fetching the message  
     
   public static String getPopupMessage(final WebDriver driver) {  
   String message = null;  
   try {  
   Alert alert = driver.switchTo().alert();  
   message = alert.getText();  
   alert.accept();  
   } catch (Exception e) {  
   message = null;  
   }  
   System.out.println("message"+message);  
   return message;  
   }
4. **Canceling pop-up in Selenium-WebDriver**  
   public static String cancelPopupMessageBox(final WebDriver driver) {  
   String message = null;  
   try {  
   Alert alert = driver.switchTo().alert();  
   message = alert.getText();  
   alert.dismiss();  
   } catch (Exception e) {  
   message = null;  
   }  
   return message;  
   }
5. **Inserting string in Text Field in Selenium-WebDriver**public static void insertText(WebDriver driver, By locator, String value) {  
   WebElement field = driver.findElement(locator);  
   field.clear();  
   field.sendKeys(value);  
   }
6. **Reading ToolTip text in in Selenium-WebDriver**  
   public static String tooltipText(WebDriver driver, By locator){  
   String tooltip = driver.findElement(locator).getAttribute("title");  
   return tooltip;  
   }
7. **Selecting Radio Button in Selenium-WebDriver**  
   public static void selectRadioButton(WebDriver driver, By locator, String value){ List select = driver.findElements(locator);  
   for (WebElement element : select)  
   {  
   if (element.getAttribute("value").equalsIgnoreCase(value)){  
   element.click();  
   }  
   }
8. **Selecting CheckBox in Selenium-WebDriver**  
     
   public static void selectCheckboxes(WebDriver driver, By locator,String value)  
   {  
   List abc = driver.findElements(locator);  
   List list = new ArrayListArrays.asList(value.split(",")));  
   for (String check : list){  
   for (WebElement chk : abc){  
   if(chk.getAttribute("value").equalsIgnoreCase(check)){  
   chk.click();  
   }}}}
9. **Selecting Dropdown in Selenium-WebDriver**public static void selectDropdown(WebDriver driver, By locator, String value){  
   new Select (driver.findElement(locator)).selectByVisibleText(value); }
10. **Selecting searched dropdown in Selenium-WebDriver**public static void selectSearchDropdown(WebDriver driver, By locator, String value){  
    driver.findElement(locator).click();  
    driver.findElement(locator).sendKeys(value);  
    driver.findElement(locator).sendKeys(Keys.TAB);  
    }
11. **Uploading file using  Selenium-WebDriver**public static void uploadFile(WebDriver driver, By locator, String path){  
    driver.findElement(locator).sendKeys(path);  
    }
12. **Downloading file in Selenium-WebDriver**Here we will click on a link and will download the file with a predefined name at some specified location.  
    public static void downloadFile(String href, String fileName) throws Exception{  
    URL url = null;  
    URLConnection con = null;  
    int i;  
    url = new URL(href);  
    con = url.openConnection();  
    // Here we are specifying the location where we really want to save the file.  
    File file = new File(".//OutputData//" + fileName);  
    BufferedInputStream bis = new BufferedInputStream(con.getInputStream());  
    BufferedOutputStream bos = new BufferedOutputStream(  
    new FileOutputStream(file));  
    while ((i = bis.read()) != -1) {  
    bos.write(i);  
    }  
    bos.flush();  
    bis.close();  
    }
13. **Wait() in Selenium-WebDriver**
    1. Implicit Wait :  
       driver.manage.timeouts().implicitlyWait(10,TimeUnit.SECONDS);
    2. Explicit Wait:WebDriverWait wait = new WebDriverWait(driver,10);  
       wait.until(ExpectedConditons.elementToBeClickable(By.id/xpath/name("locator"));
    3. Using Sleep method of java  
       Thread.sleep(time in milisecond)
14. **Navigation method of WebDriver Interface**
    1. **to()** method (its a alternative of **get() method**)  
       driver.navigate().to(Url);  
       This will open the URL that you have inserted as argument
    2. **back()** – use to navigate one step back from current position in recent history syntax ==driver.navigate().back();
    3. **forward()** – use to navigate one step forward in browser history driver.navigate().forward();
    4. **refresh()** – This will refresh you current open url driver.navigate().refresh();
15. **Deleting all Cookies before doing any kind of action**driver.manage().deleteAllCookies();  
    This will delete all cookies
16. **Pressing any Keyboard key using Action builder class of WebDriver**WebDriver has rewarded us with one class Action to handle all keyboard and Mouse action. While creating a action builder its constructor takes WebDriver as argument. Here I am taking example of pressing Control key  
    Actions builder = new Actions(driver);  
    builder.keyDown(Keys.CONTROL).click(someElement).click(someOtherElement).keyUp(Keys.CONTROL).build().perform();When we press multiple keys or action together then we need to bind all in a single command by using build() method and perform() method intend us to perform the action.  
    In the same way you can handle other key actions.
17. **Drag and Drop action in Webdriver**  
    In this we need to specify both WebElement  like Source and target and for draganddrop Action class has a method with two argument so let see how it normally look like  
    WebElement element = driver.findElement(By.name("source"));  
    WebElement target = driver.findElement(By.name("target"));  
    (new Actions(driver)).dragAndDrop(element, target).perform();
18. **Handling Multiple Windows in WebDriver**

Before commencing, let us first consider a few situations when we are likely to deal with multiple windows.

* Filling forms may require to select the date from a separately opened window.
* Clicking on some link/button can kick-off yet another window.
* Handling Advertisement windows

Hence, we can come up with various scenarios depending upon the application.

Now let us motion ourselves towards the challenge we face under above situations. The most particular of all is switching the focus from one window to another. Let us understand the same in the following way:

[Handling Multiple Windows in WebDriver](http://www.abodeqa.com/wp-content/uploads/2013/09/Handling-Multiple-Windows-in-WebDriver.png)

Comprehending from the above figure, the entire process can be fundamentally segregated into following steps:

**Step 1 :** Clicking on Link1 on Window A

A new Window B is opened.

**Step 2 :** Move Focus from Window A to Window B

Window B is active now

**Step 3 :** Perform Actions on Window B

Complete the entire set of Actions

**Step 4 :** Move Focus from Window B to Window A

Window A is active now

**These are the steps** which we can easily interpret out of the diagram, but there are a few more steps to add to complete this process and making our script execute. These steps don’t have visibility but plays a very vital role. Let us now re-consider the same scenario.

**Step 1 :** Clicking on Link1 on Window A

**A new Window B is opened.**

Step 2 :**Save reference for Window A**

Step 3 :**Create reference for Window B**

**Step 3 :** Move Focus from Window A to Window B

**Window B is active now**

**Step 3 :** Perform Actions on Window B

Complete the entire set of Actions

**Step 4 :** Move Focus from Window B to Window A

**Window A is active now**

Let us understand the same with a small coding example.

public class MultiWindowHandle {

WebDriver driver;

@Before

public void setup() throws Exception {

driver=new FirefoxDriver();

String URL="https://www.abc.co.in/";

driver.get(URL);

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

}

@Test

public void test() throws Exception {

// Opening Calender

driver.findElement(By.xpath("//img[@alt='Calender']")).click();

// Storing parent window reference into a String Variable

String Parent\_Window = driver.getWindowHandle();

// Switching from parent window to child window

for (String Child\_Window : driver.getWindowHandles())

{

driver.switchTo().window(Child\_Window);

// Performing actions on child window

driver.findElement(By.id("calendar\_month\_txt")).click();

List Months=driver.findElements(By.xpath("//div[@id='monthDropDown']//div"));

int Months\_Size=Months.size();

System.out.println("Month size is:"+Months\_Size);

Months.get(1).click();

driver.findElement(By.xpath("//\*[@id='calendarDiv']/div/table/tbody/tr/td[contains(text(),'16')]")).click();

}

//Switching back to Parent Window

driver.switchTo().window(Parent\_Window);

//Performing some actions on Parent Window

driver.findElement(By.className("btn\_style")).click();

}

@After

public void close() {

driver.quit();

}

}