1.Interacting with elements

Interacting with basic elements.

To perform actions on basic elements like textbox, buttons, radio buttons and checkboxes, we first locate these elements with the above discussed locating techniques and then click, check or enter text to them.

```
driver.findElement(By.id("Id of element")).click();
driver.findElement(By.id("Id of element")).sendKeys("Text to be entered in text box");
driver.findElement(By.id("Id of element")).clear(); /* To clear text present in a text
box*/
```

JS executors

In some cases we encounters situations when we have to interact with elements which are read only, for example a text box whose value can't be edited.
E.g.
In these cases, the webdriver API's sendKeys() method will not be able to enter text (or clear the text) from these textboxes. To handle it we can adopt any of the two solutions below :

Solution 1
Using java script executor we can change the 'readonly' attribute and set it to blank. Once the readonly attribute is changed to blank (or something other than readonly), the sendKeys() method will work as expected

```
JavascriptExecutor jse = (JavascriptExecutor)driver;
jse.executeScript("document.getElementById('ID OF ELEMENT').readOnly=''");
driver.findElement(By.id("ID OF ELEMENT")).sendKeys("Text to enter");
```

If required we can restore the readonly attribute once we are done with entering text as follows :

```
jse.executeScript("document.getElementById('ID OF ELEMENT').readOnly=''readonly");
```

Solution 2

Again with the help of java script executor, we can directly change the value in a single step :

```
JavascriptExecutor jse = (JavascriptExecutor)driver;
jse.executeScript("document.getElementById('ID OF ELEMENT').value='TEXT TO BE
ENTERED'");
```

We don't need to restore the read only attribute after doing this, as we are not changing it.

Handling popups

In webdriver to test popup windows we switch the driver to the popup window and then perform the desired accept or cancel action.

```
driver.switchTo().alert().dismiss(); /* to simulate the pressing of Cancel button on
javascript poopup.*/
driver.switchTo().alert().accept(); /* to simulate the pressing of OK button on
javascript poopup.*/
```

Handling multiple browser pages

Windows

While working with multiple browser windows, we need to switch among browser windows using the driver.switchTo() and driver.getWindowHandles methods. We can switch to a browser window using the title of browser window
or the index.

Switching browser with title

```
String mainBrowserHandle = driver.getWindowHandle();
Set<String> windowHandle_set=driver.getWindowHandles();
 for(String windowHandle : windowHandle_set){
 String windowTitle=driver.switchTo().window(windowHandle).getTitle();
 if(windowTitle.contains("Title of browser Window")){
  break;
 }
 else{
  driver.switchTo().window(mainBrowserHandle);
 }
}
```

Allerts

When dealing with alerts and need to take the alert text we need to switch our browser to the given alert.

```
Alert alert = driver.switchTo().alert();
alertText = alert.getText();
alert.accept();
```

Sometimes is possible the code to fail because the alert is not shown by the specific time and the code fails. IN this cases a wait for a given specified condition is required before the alert switching code.

```
driver.findElement(By.xpath("given xpath")).click();
wait.until(ExpectedConditions.alertIsPresent());
```

Frames

When having to switch from a frame to another, we can invoke the below code:

```
driver.switchTo().frame("main");
```

In cases when the frame in not currently present, we need to wait for it:

```
wait.until(ExpectedConditions.frameToBeAvailableAndSwitchToIt("head"));
```

Explicit and implicit waits

An implicit wait is to tell WebDriver to check the DOM for a certain amount of time when trying to find an element or elements if they are not immediately available.

```
driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
```

An explicit wait is to wait for certain condition to be true before proceeding further. For example the below code snippet will wait for 10s or the Expected Condition to be true.

WebDriverWait by default calls the ExpectedCondition every 500 milliseconds until it returns successfully or the timeout occurs.

```
new WebDriverWait(driver, 10).until(new ExpectedCondition<Boolean>() {
@Override
public Boolean apply(WebDriver d) {
List<WebElement> elements = d.findElements(By.xpath("XPATH locator of Element"));
if(elements.size()==0){
return false;
}else{
return true;
}
}
});
```

Explicit wait using 'ExpectedConditions' package

```
new
WebDriverWait(driver,100).until(ExpectedConditions.presenceOfElementLocated(By.xpath("
XPath of Element")));
```

The ExpectedConditions package contains a set of predefined conditions to use with WebDriverWait e.g. elementToBeSelected, elementToBeClickable etc.

Taking Screenshots

In order to take a screenshot with the current page, you can use

```
File scrFile = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);
```