Driver Initilization		
WebDriver driver = new ChromeDriver();		
WebDriver driver = new FirefoxDriver();		
WebDriver driver = new InternetExplorerDriver();		
WebDriver driver = new EdgeDriver();		
WebDriver driver = new SafariDriver();		
WebDriver driver = new HtmlUnitDriver();		
Advanced Browser Cnfiguration		
ChromeOptions chromeOptions = new ChromeOptions();		
chromeOptions.addArguments("--ignore-certificate-errors");		
chromeOptions.addArguments("user-data-dir=Path");		
chromeOptions.addArguments("--headless");		
chromeOptions.addArguments("--start-maximized", "--incognito","--disable-notifications");		
driver = new ChromeDriver(chromeOptions);		
//Desired Capabilities		
DesiredCapabilities dc = new DesiredCapabilities();		
dc.setCapability(“browserName”, “chrome”);		
dc.setCapability(“browserVersion”, “96.0””);		
dc.setCapability(“platformName”, “win10”);		
WebDriver driver = new ChromeDriver(dc);		
Element Locators		
driver.findElement(By.id("Id Value"));		
driver.findElement(By.name("Name Value"));		
driver.findElement(By.className("Class Name Value"));		
driver.findElement(By.linkText("Link text Value"));		
driver.findElement(By.partialLinkText("Partial Text Constant Value"));		
driver.findElement(By.tagName("Tag Name Value"));		
driver.findElement(By.cssSelector("CSS Value"));		
driver.findElement(By.xpath("Xpath Value"));		
driver.findElement(with(By.tagName("input")) .above(passwordField));		
driver.findElement(with(By.tagName("input")) .below(emailAddressField));		
driver.findElement(with(By.tagName("button")) .toLeftOf(submitButton));		
driver.findElement(with(By.tagName("button")).toRightOf(submitButton));		
driver.findElement(with(By.tagName("input")).near(emailAddressLabel));		
driver.findElement(new ByAll(By.className("ElementClass Name"), By.id("Element Id"), By.name("Element Name")))		
Elements Operations		
WebElement element = driver.FindElement(By.ElementLocator("Value of Element Locator"));		
element.click();		
element.sendKeys("Input Text");		

element.clear();		
element.submit();		
element.getAttribute(“type”);		
String innerText = element.getText();		
boolean enabledstatus = element.isEnabled();		
boolean displayedstatus = element.isDisplayed();		
boolean selectedstatus = element.isSelected();		
//Operation on drop down		
Select select = new Select(element);		
select.selectByIndex(Integer Index);		
select.selectByVisibleText("Text");		
select.SelectByValue("Value");		
select.deselectAll();		
select.deselectByIndex(Integer Index);		
select.deselectByVisibleText("Text");		
select.deselectByValue("Value");		
WebElement selectedOptions = select.getOptions();		
Browser Operations		
String pageTitle = driver.getTitle();		
String currentURL = getCurrentUrl();		
String currentPageSource = driver.getPageSource();		
// Navigation history		
driver.get("https://www.facebook.com/");		
driver.manage().window().maximize();		
driver.manage().window().fullscreen();		
driver.navigate().to("https://www.google.com/");		
driver.navigate().back();		
driver.navigate().forward();		
driver.navigate().refresh();		
driver.close();		
driver.quit();		
// Handle Alert		
Alert alert = driver.switchTo().alert();		
alert.accept();		
alert.dismiss();		
alert.getText();		
alert.sendKeys(“Input Data");		
//Handle Cookies		
Cookie cookie = new Cookie(“cookieName”, “cookieValue”);		
driver.manage().addCookie(cookie);		
driver.manage().getCookies();		
driver.manage().getCookieNamed(arg0);		
driver.manage().deleteAllCookies();		
driver.manage().deleteCookieNamed(arg0);		
// Handle frames		
driver.switchTo().frame(int Frame Index);		
driver.switchTo().frame("frameName");		
WebElement element = driver.FindElement(By.ElementLocator("Value of Element Locator"));		
driver.switchTo().frame(element);		

driver.SwitchTo().defaultContent();		
Screenshots Capture		
TakesScreenshot screenshot =((TakesScreenshot)driver);		
File srcFile= screenshot.getScreenshotAs(OutputType.FILE);		
FileHandler.copy(srcFile, destFile);		
Actions Class/Mouse Event		
Actions action = new Actions(driver);		
action.keyDown(Keys.CONTROL);		
action.keyUp(Keys.CONTROL);		
action.clickAndHold(webElement).build().perform();		
action.doubleClick(webElement).build().perform();		
action.moveToElement(webElement).build().perform();		
action.moveByOffset(xOffset,yOffset).build().perform();		
action.dragAndDrop(sourceEle,targetEle).build().perform();		
action.release().build().perform();		
Manage Timeouts		
driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);		
welement = wait.until(Syntax: WebDriverWait wait = new WebDriverWait(driver, timeout); ExpectedConditions.elementToBeClickable(locator)); welement.click();		
driver.manage().timeouts().pageLoadTimeout(30, TimeUnit.SECONDS);		
Upload a File		
WebElement element = driver.FindElement(By.ElementLocator("Value of Element Locator"));		
element.sendKeys(filePath);		
Window Handle		
String handle=driver.getWindowHandle();		
Set<String> handles = getWindowHandles();		
driver.switchTo().window(handle);		
Capture Width and Height of an Element		
Point point = element.getLocation();		
int elementWidth = element.getSize().getWidth();		
int elementHeight = element.getSize().getHeight();		
Window Handle		
String handle=driver.getWindowHandle();		
Set<String> handles = getWindowHandles();		
driver.switchTo().window(handle);		
Scroll Down or Up Web Page		
JavascriptExecutor js = (JavascriptExecutor)driver;		
js.executeScript("window.scrollBy(0,100)");		
js.executeScript("window.scrollTo(0, document.body.scrollHeight)");		
WebElement element = driver.FindElement(By.ElementLocator("Value of Element Locator"));		

js. executeScript("arguments[0].scrollIntoView()", element);		
Selenium Grid		
//Start hub		
java-jar selenium-server- standalone-x.xx.x.jar-role hub		
//Start node		
java-jar selenium-server- standalone-x.xx.x.jar-role node-hub		
http://localhost:4444/grid/register		
//Server		
http://localhost:4444/grid/console		
TestNG Annotations		
@Test		
@BeforeMethod		
@AfterMethod		
@BeforeTest		
@AfterTest		
@BeforeClass		
@AfterClass		
@Test(enabled = false)		
@Test(enabled = true)		
@Test(priority=2)		
@Test(priority=5,dependsOnMethods={"method1","method2"})		
@Test(dependsOnMethods = {"method1"}, alwaysRun=true)		
@Test(groups = { "Group1", "Group2" })		
@Parameters({"testparameter1", "testparameter2"})		
@Listeners(packagename.ListenerClassName.class)		
@Test (dataProvider = "getUserIDandPassword")		
@Test (description = "Open Facebook Login Page", timeOut=35000)		
@Test (invocationCount = 3, invocationTimeOut = 20000)		
@Test (invocationCount = 3, skipFailedInvocations = true)		
@Test (invocationCount = 3)		
@Test (invocationCount = 7, threadPoolSize = 2)		
TestNG Assertions		
SoftAssert softassert= new SoftAssert();		
softassert.assertEquals(1, 1);		
softassert.assertAll();		
Assert.assertEquals(11, 11);		
Assert.assertEquals(true, true, "Not Matching");		
TestNG XML File		
<suite name="TestNGSuite" parallel="methods" thread-count="4">		
<listeners>		
<listener class-name="packagename.classname"></listener>		
</listeners>		
<parameter name="Data1" value="10" />		

<parameter name="Data2" value="3" />		
<test name="TestNGTest">		
<groups>		
<run>		
<exclude name="Test.*" />		
<include name="SmokeTest" />		
</run>		
</groups>		
<classes>		
<class name="packagename.classname"/>		
</classes>		
</test> <!-- TestNGTest -->		
</suite> <!-- TestNGSuite -->		
Different approach to create XPath and CSS Selector		
Description	**XPath**	**CSS Selector**
Whole WebPage	/html	html
Whole WebPage body	/html/body	body
image element	//img	img
Link	//a[@href = 'url']	a[href = 'url']
Direct Child	//div/a	div > a
Id	//tagName[@id='idValue']	tagName#idValue
Class	//tagName[@class='classValue']	tagName.Value of Class attribute
Attribute	//tagname[@attribute-name='value1']	tagName[attribute=Value of attribute]
Multiple Attributes	//input[@type='submit' and @name='btnLogin']	tagname[attribute1='value1'][attribute2='value2']
Contains	//*[contains(@type,'sub')]	<HTML tag><[attribute*=sub String]>
Starts with	//tagname[starts-with(@attribute, 'Start value')]	<HTML tag><[attribute^=prefix of the String]>
Ends with	//tagname[ends-with(@attribute, 'End value')]	<HTML tag><[attribute$=suffix of the String]>
Matches	//*[matches(@type,'value')]	N/A
First Child	//ul[@id='list']/li[1]	ul#list li:first-child
Last Child	//ul[@id='list']/li[last()]	ul#list li:last-child
nth Child	//ul[@id='list']/li[3]	ul#list li:nth-child(3)
Text Value	//td[text()='textname']	N/A
Element preceding some sibling	//E2/preceding-sibling::E1	N/A
Sibling element immediately preceding	//E/preceding-sibling::*[1]	N/A
User interface element that is disabled	//E[@disabled]	E:disabled
Checkbox (or radio button) that is checked	//*[@checked]	*:checked
Text Value	//td[text()='textname']	N/A
Important Tagname and their Description		
Tag Name	**Specification**	
	Defines an unordered list of items.	

<tr>	Defines a row of cells in a table.	
<title>	Represents title to an HTML document.	
<th>	Used for creates header of a group of cell in HTML table.	
<table>	Used to defines a table in an HTML document.	
<tbody>	Used for grouping table rows.	
<td>	Used for creates standard data cell in HTML table.	
	Used to grouping and applying styles to inline elements.	
<section>	Used to divide a document into number of different generic section.	
<select>	Used to create a drop-down list.	
<menu>	Used to display a unordered list of items/menu of commands.	
	Define a list item either ordered list or unordered list.	
<iframe>	Defines a inline frame that embedded external content into current web document.	
<img>	Used to insert image into a web document.	
<input>	Define a get information in selected input	
	Define a division part	
<body>	Defines a main section(body) part in HTML document	
 	Specific a single line break	
<button>	Specifies a press/push button	
<a>	Specific a anchor (Hyperlink)	