



Selenium

Class 5

Agenda

WebElement Commands

Working with Radio Buttons, CheckBoxes, DropDowns

WebElement Commands

WebDriver either returns something or return void(means return nothing). The same way **findElement** command of **WebDriver** returns **WebElement**. So, to get the WebElement object write the below statement:

```
WebElement element = driver.findElement(By.id("UserName"));
```

- And now if you type **element dot**, Eclipse will populate the complete list of actions
- One more thing to notice that WebElement can be of any type, like it can be a **Link, Radio Button, Drop Down, Web Table** or any HTML element.

WebElement Commands

- **sendKeys(CharSequence... keysToSend) : void** - This simulate typing into an element, which may set its value. This method accepts CharSequence as a parameter and returns nothing.
- **Command** - **element.sendKeys("text");**
- This method works fine with text entry elements like **INPUT** and **TEXTAREA** elements.

```
WebElement element = driver.findElement(By.id("UserName"));  
element.sendKeys("SyntaxTech");  
//Or can be written as  
driver.findElement(By.id("UserName")).sendKeys("SyntaxTech");
```

WebElement Commands

- **clear() : void** - If this element is a text entry element, this will clear the value. This method accepts nothing as a parameter and returns nothing.
- **Command - `element.clear();`**
- This method has no effect on other elements. Text entry elements are **INPUT** and **TEXTAREA** elements.

```
WebElement element =  
driver.findElement(By.name("UserName"));  
element.clear();  
//Or can be written as  
driver.findElement(By.name("UserName")).clear();
```

WebElement Commands

- **click() : void** - This simulates the clicking of any element. Accepts nothing as a parameter and returns nothing.
- **Command** - **element.click();**
- Clicking is perhaps the most common way of interacting with web elements like text elements, links, radio boxes and many more.

```
WebElement element = driver.findElement(By.linkText("Pricing"));  
element.click();  
//Or can be written as  
driver.findElement(By.linkText("Pricing")).click();
```

WebElement Commands

- **submit() : void**- This method works well/better than the click() if the current element is a form, or an element within a form. This accepts nothing as a parameter and returns nothing.
- **Command - `element.submit();`**
- If this causes the current page to change, then this method will wait until the new page is loaded.

```
WebElement element =  
driver.findElement(By.xpath("//*[@value='Login']"));  
element.submit();
```

//Or can be written as

```
driver.findElement(By.xpath("//*[@value='Login']")).submit();
```

WebElement Commands

- **isDisplayed() : boolean** - This method determines if an element is currently being displayed or not. This accepts nothing as a parameter but returns boolean value(true/false).

- **Command - `element.isDisplayed();`**

- **isEnabled() : boolean** - This determines if the element currently is **Enabled or not?** This accepts nothing as a parameter but returns boolean value(true/false).

- **Command - `element.isEnabled();`**

- **isSelected() : boolean** - Determine whether or not this element is selected or not. This accepts nothing as a parameter but returns boolean value(true/false).

- **Command - `element.isSelected();`**

- This operation only applies to input elements such as **Checkboxes, Select Options** and **Radio Buttons**. Th

WebElement Commands

- **getText() : String**- This method will fetch the visible (i.e. not hidden by CSS) innerText of the element. This accepts nothing as a parameter but returns a String value.
- **Command - `element.getText();`**
- This returns an innerText of the element, including sub-elements, without any leading or trailing whitespace.

```
WebElement element =  
driver.findElement(By.xpath("//*[@value='Login']"));  
String elementString = element.getText();
```

//Or can be written as

```
String elementString =  
driver.findElement(By.xpath("//*[@value='Login']")).getText();
```

WebElement Commands

- **getAttribute(String Name) : String**- This method gets the value of the given attribute of the element. This accepts the String as a parameter and returns a String value.
- **Command - `element.getAttribute()`;**
- Attributes are Ids, Name, Class extra and using this method you can get the value of the attributes of any given element.

```
WebElement element = driver.findElement(By.id("SubmitButton"));
String attValue = element.getAttribute("id");
//This will return "SubmitButton"
```

Test Case

TC 1: Swag Labs Positive login:

1. Open chrome browser
2. Go to "https://www.saucedemo.com/"
3. Enter valid username and valid password and click login
4. Verify robot icon is displayed
5. Verify "Products" text is available next to the robot icon

TC 2: Swag Labs Negative login:

1. Open chrome browser
2. Go to "https://www.saucedemo.com/"
3. Enter invalid username and password and click login
4. Verify error message contains: "Username and password do not match any user in this service"

Difference between FindElement & FindElements Commands

The difference between **findElement()** and **findElements()** method is the first returns a WebElement object otherwise it throws an exception and the **findElements()** returns a List of WebElements, it can return an empty list if no DOM elements match the query.

findElement()	findElements()
<ul style="list-style-type: none">• On Zero Match : throws NoSuchElementException• On One Match : returns WebElement• On One+ Match : returns the first appearance in DOM	<ul style="list-style-type: none">• On Zero Match : return an empty list• On One Match : returns list of one WebElement only• On One+ Match : returns list with all matching instance

Test Case

TC 1: Amazon link count:

1. Open chrome browser
2. Go to “https://www.amazon.com/”
3. Using Iterator get text of each link
4. Get number of links that has text

CheckBox & Radio Button Operations

- **CheckBox & Radio Button Operations** are easy to perform and most of the times the simple **ID attributes** work fine for both of these.
- But selection and d-selection is not the only thing we want with Checkboxes and Radio Buttons. We might like to check that if the Checkbox is already checked or if the Radio Button is selected by default or anything.
- Checkboxes and Radio Button deals exactly the same way and you can perform below mentioned operations on either of them.

CheckBox & Radio Button Operations

1. Various methods to select checkbox and radio button.

1.1- Use ID for Selecting Checkbox/Radio button.

You can use the ID attribute to select a Radio Button or a CheckBox. We've provided the Webdriver command to click which you can apply to both types of elements.

1.2- Use IsSelected Method to Check the State of Checkbox/Radio button.

If you've selected/deselected a Checkbox/Radio Button and you just want to check its final state. Then, you can use the <IsSelected> command to know that the correct status of the element.

Working with Radio Button/CheckBoxes

// Store all the elements of the same category in the list of WebElements.

```
List<WebElement> list = driver.findElements(By.name("radioButton"));
```

// Create a boolean variable to store true/false.

```
Boolean is_selected = list.get(0).isSelected();
```

// If 'is_selected' is true that means the first radio button is selected.

```
if (is_selected == true) {
```

```
    // If the first radio button is selected by default then, select the second radio button.
```

```
    list.get(1).click();
```

```
} else {
```

```
    // If the first radio button is not selected then, click the first radio button.
```

```
    list.get(0).click();
```

```
}
```


CheckBox & Radio Button Operations

Note: When there is a group of Radio Buttons/Check Boxes on the page then, it is possible that their names are same, but values are different. That's why we've used the Webdriver findElements command to get the list of web elements.

1.3- Use Element Value for Selecting Checkbox/Radio button.

One of the intuitive ways to select the Radio Buttons/Check Boxes is by toggling their Values. Please follow the below code example for more clarity.

Working with Radio Button/ Checkboxes

```
// Get radio group
```

```
List<WebElement> radioBtn=driver.findElements(By.name("sex"));
```

```
// Now loop from first checkbox to last checkbox.
```

```
for(WebElement radio: radioBtn) {
```

```
//Store the checkbox name to the string variable, using 'Value' attribute
```

```
String value=radio.getAttribute("value");
```

```
//check if values is same as you want and element is enabled
```

```
    if(radio.isEnabled() && value.equals("Female")) {
```

```
        // click on that element
```

```
        radio.click();
```

```
        // This statement will get you out of the for loop.
```

```
        break;
```

```
    }
```

Test Case

TC 1: Tools QA check all elements

1. Open chrome browser
2. Go to
“<https://www.toolsqa.com/automation-practice-form/>”
3. Fill out:
 - First Name
 - Last Name
 - Check that sex radio buttons are enabled and select “Male”
 - Check all Years of Experience radio buttons are clickable
 - Date
 - Select both checkboxes for profession
 - Check all Automation Tools checkboxes are clickable and keep “Selenium WebDriver” option as selected
4. Quit browser

DropDown & Multiple Select Operation

- DropDown & Multiple Select Operations also works together and almost the same way.
- To perform any action, the first task is to identify the element group. It is a group, as DropDown /Multiple Select is not a single element.
- They always have a single name but they contains one or more than one elements in them/more than one option in DropDown and Multiple Select.
- The only difference between these two is deselecting statement & multiple selections are not allowed on DropDown

DropDown & Multiple Select Operation

To work with drop down we can choose it by ID, Name, Css & Xpath etc.

To perform any action on drop down element it is required to import '**import org.openqa.selenium.support.ui.Select**' package and to use it we need to create a new Select Object of class Select.

Selenium webdriver provides Select class, with help of select class we can handle dropdowns on the webpage

```
Select s = new Select()
```

Drop Down with Select Class

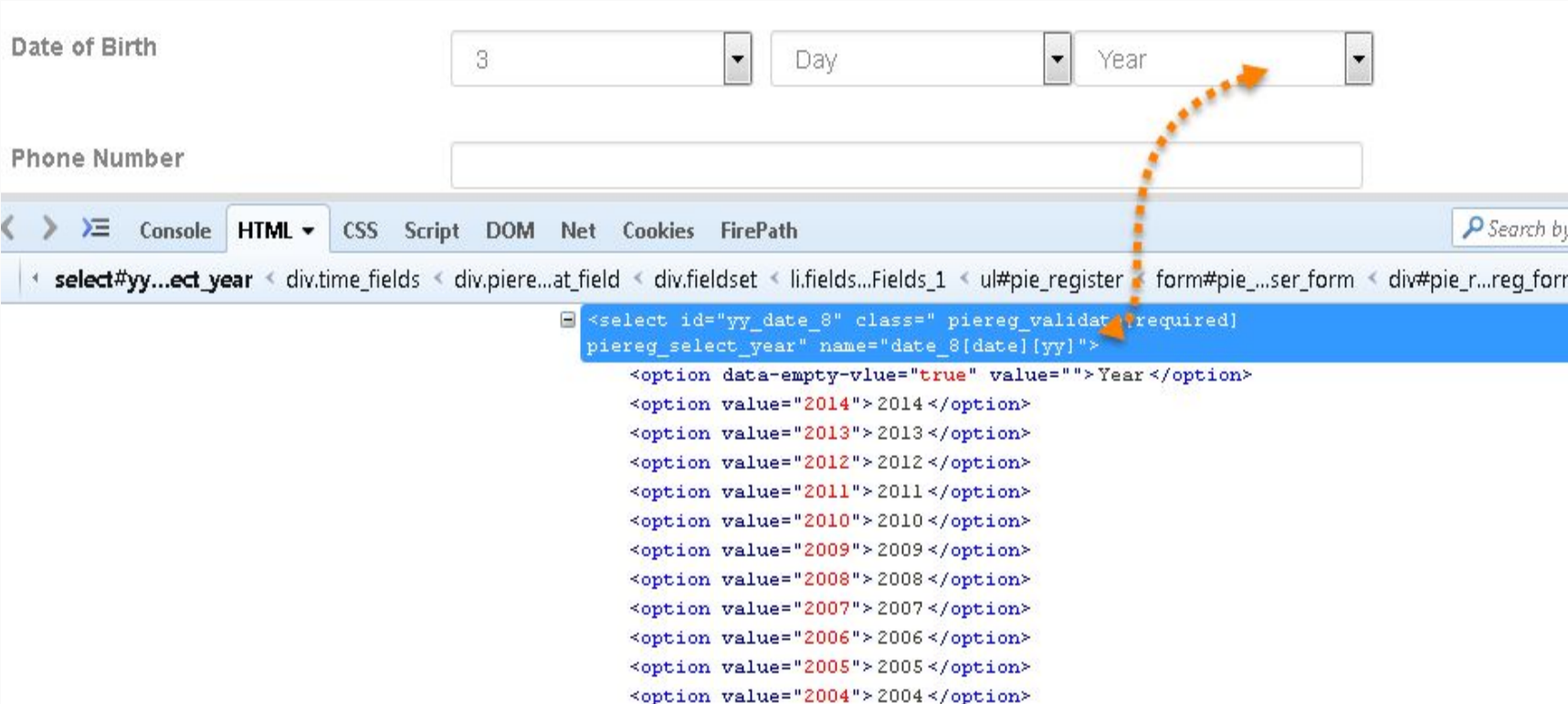
Date of Birth

3

Day

Year

Phone Number



```
<select id="yy_date_8" class=" pierereg_validation required"
pierereg_select_year" name="date_8[date][yy]">
  <option data-empty-value="true" value="">Year </option>
  <option value="2014">2014 </option>
  <option value="2013">2013 </option>
  <option value="2012">2012 </option>
  <option value="2011">2011 </option>
  <option value="2010">2010 </option>
  <option value="2009">2009 </option>
  <option value="2008">2008 </option>
  <option value="2007">2007 </option>
  <option value="2006">2006 </option>
  <option value="2005">2005 </option>
  <option value="2004">2004 </option>
```

DropDown & Multiple Select Operation

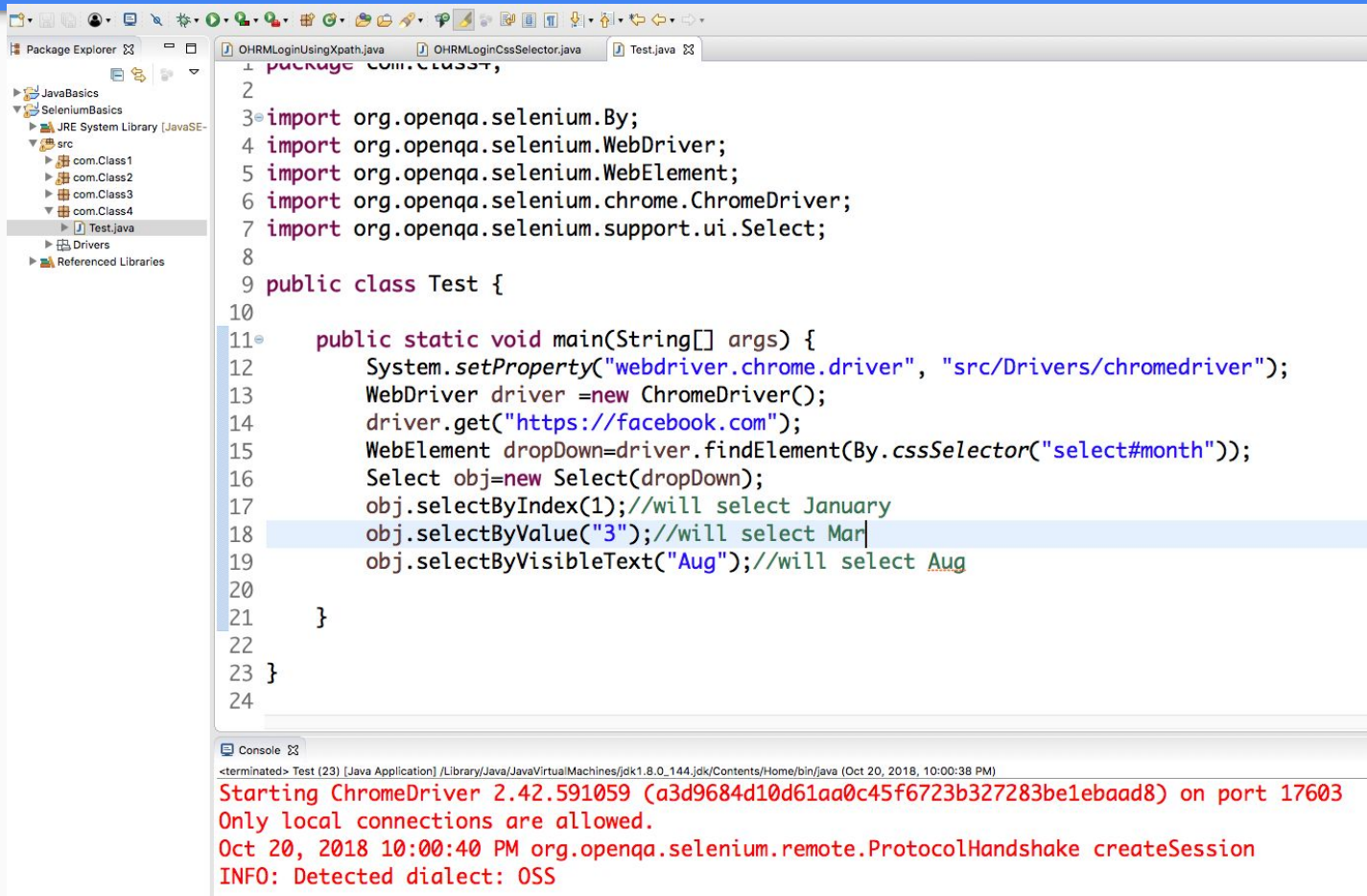
Dropdown are formed using select tag in html, if dropdown **is not** formed with select tag you **cannot use this Select** Class methods in selenium.

To Handle Drop Down And Multi Select List in Selenium we use the following types of Select Methods:

- selectByVisibleText Method
- selectByIndex Method
- selectByValue Method

Note: to verify if drop down allows multiple selection use method `isMultiple()`;

DropDown demo



The screenshot shows an IDE with a Package Explorer on the left and a code editor on the right. The Package Explorer shows a project structure with a 'Test.java' file selected. The code editor displays the following Java code:

```
1 package com.class2;  
2  
3 import org.openqa.selenium.By;  
4 import org.openqa.selenium.WebDriver;  
5 import org.openqa.selenium.WebElement;  
6 import org.openqa.selenium.chrome.ChromeDriver;  
7 import org.openqa.selenium.support.ui.Select;  
8  
9 public class Test {  
10  
11     public static void main(String[] args) {  
12         System.setProperty("webdriver.chrome.driver", "src/Drivers/chromedriver");  
13         WebDriver driver =new ChromeDriver();  
14         driver.get("https://facebook.com");  
15         WebElement dropDown=driver.findElement(By.cssSelector("select#month"));  
16         Select obj=new Select(dropDown);  
17         obj.selectByIndex(1);//will select January  
18         obj.selectByValue("3");//will select Mar|  
19         obj.selectByVisibleText("Aug");//will select Aug  
20  
21     }  
22  
23 }  
24
```

The console output at the bottom shows the following messages:

```
<terminated> Test (23) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_144.jdk/Contents/Home/bin/java (Oct 20, 2018, 10:00:38 PM)  
Starting ChromeDriver 2.42.591059 (a3d9684d10d61aa0c45f6723b327283be1ebaad8) on port 17603  
Only local connections are allowed.  
Oct 20, 2018 10:00:40 PM org.openqa.selenium.remote.ProtocolHandshake createSession  
INFO: Detected dialect: OSS
```

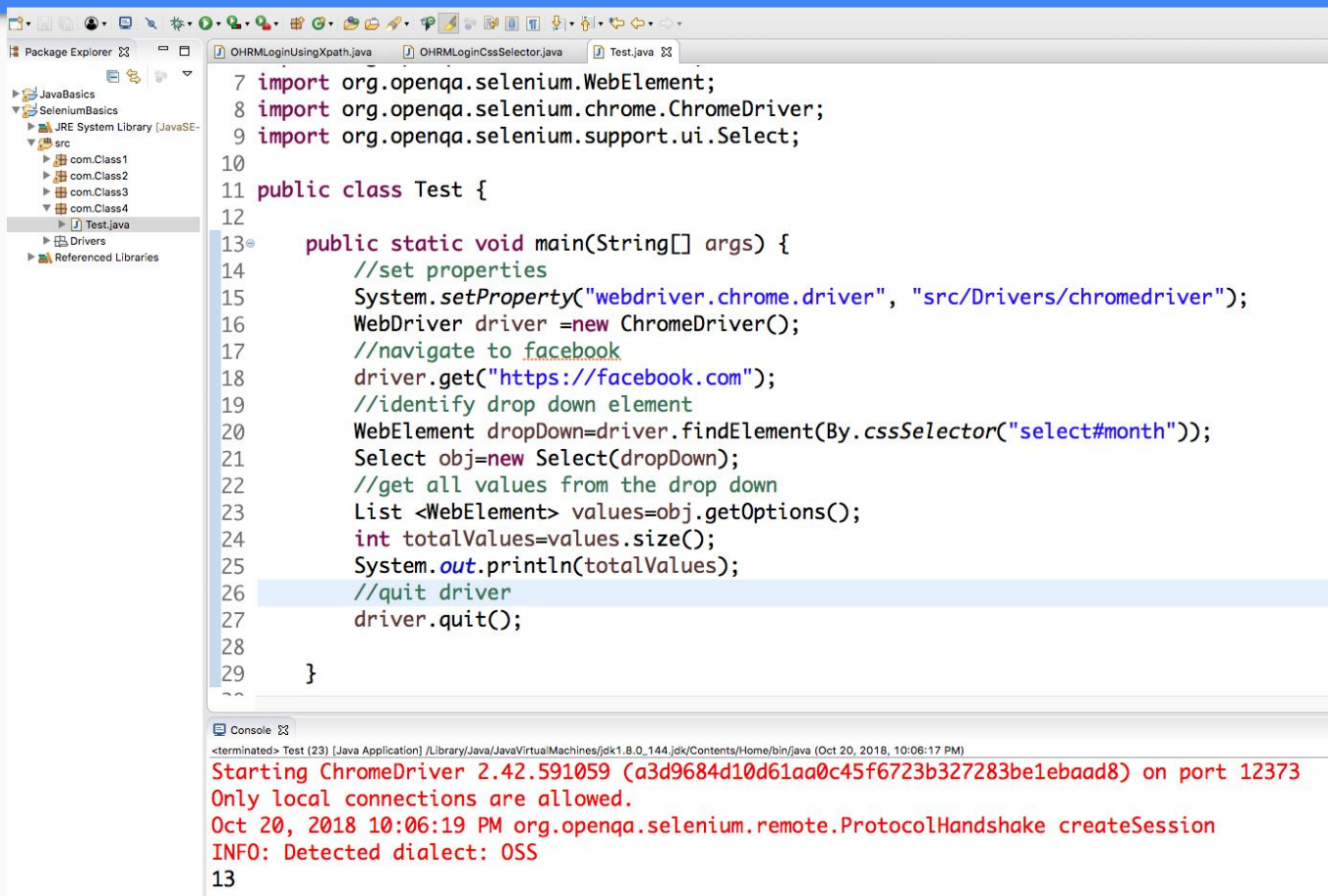

DropDown & Multiple Select Operation

`getOptions()` : `List<WebElement>` –This gets the all options belonging to the Select tag. It takes no parameter and returns `List<WebElements>`.

Command - `oSelect.getOptions();`

Sometimes you may like to count the element in the dropdown and multiple select box, so that you can use the loop on Select element.

DropDown getOptions()



The screenshot shows an IDE with a Package Explorer on the left and a code editor on the right. The Package Explorer shows a project structure with a 'Test.java' file. The code editor displays the following Java code:

```
7 import org.openqa.selenium.WebElement;
8 import org.openqa.selenium.chrome.ChromeDriver;
9 import org.openqa.selenium.support.ui.Select;
10
11 public class Test {
12
13     public static void main(String[] args) {
14         //set properties
15         System.setProperty("webdriver.chrome.driver", "src/Drivers/chromedriver");
16         WebDriver driver =new ChromeDriver();
17         //navigate to facebook
18         driver.get("https://facebook.com");
19         //identify drop down element
20         WebElement dropDown=driver.findElement(By.cssSelector("select#month"));
21         Select obj=new Select(dropDown);
22         //get all values from the drop down
23         List <WebElement> values=obj.getOptions();
24         int totalValues=values.size();
25         System.out.println(totalValues);
26         //quit driver
27         driver.quit();
28
29     }
30 }
```

The console output at the bottom shows the following messages:

```
<terminated> Test (23) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_144.jdk/Contents/Home/bin/java (Oct 20, 2018, 10:06:17 PM)
Starting ChromeDriver 2.42.591059 (a3d9684d10d61aa0c45f6723b327283be1ebaad8) on port 12373
Only local connections are allowed.
Oct 20, 2018 10:06:19 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: OSS
```

The line number 13 is highlighted in the code editor.

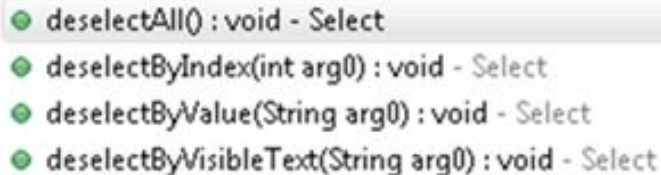
Test Case

TC 1: Facebook dropdown verification

1. Open chrome browser
2. Go to “https://www.facebook.com”
3. Verify:
 - month dd has 12 month options
 - day dd has 31 day options
 - year dd has 115 year options
4. Select your date of birth
5. Quit browser

DeSelect Methods

- The way we select different values of DropDown & Multi Select, the same way we can also **deselect** the values.
- But the only challenge is these methods do not work for DropDown and only work for Multi Select elements.
- In case you want to deselect any pre-selected option, that can be done with either `deselectAll()`, `deselectByIndex`, `deselectByValue` and `deselectByVisibletext`.



```
deselectAll() : void - Select  
deselectByIndex(int arg0) : void - Select  
deselectByValue(String arg0) : void - Select  
deselectByVisibleText(String arg0) : void - Select
```

1. deselectAll() : void – Clear all selected entries. Valid when the SELECT supports multiple selections.

Command - `oSelect.deselectAll;`

2. deselectByIndex(int arg0) : void –Deselect the option at the given index.

Command - `oSelect.deselectByIndex;`

3. deselectByValue(String arg0) : void –Deselect all options that have a value matching the argument.

Command - `oSelect.deselectByValue;`

4. deselectByVisibleText(String arg0) : void – Deselect all options that display text matching the argument.

Command - `oSelect.deselectByVisibleText`

5. isMultiple() : boolean – Tells whether the SELECT element support multiple selecting options at the same time or not. It accepts nothing by returns boolean value(true/false).

Command - `oSelect.isMultiple();`

This is done by checking the value of the “multiple” attribute.