Rex
Jones II

Selenium 4
Relative Locators

Table of Contents

Selenium 4 Video Playlist 2
Introduction 2
Use One Relative Locator 2
Use Multiple Relative Locators 5
Find List Of Elements 8
Contact Info 9

Selenium 4 Video Playlist

https://www.youtube.com/playlist?list=PLfp-cJ6BH8u_4AMzeLVizVfqn4SCywSTJ

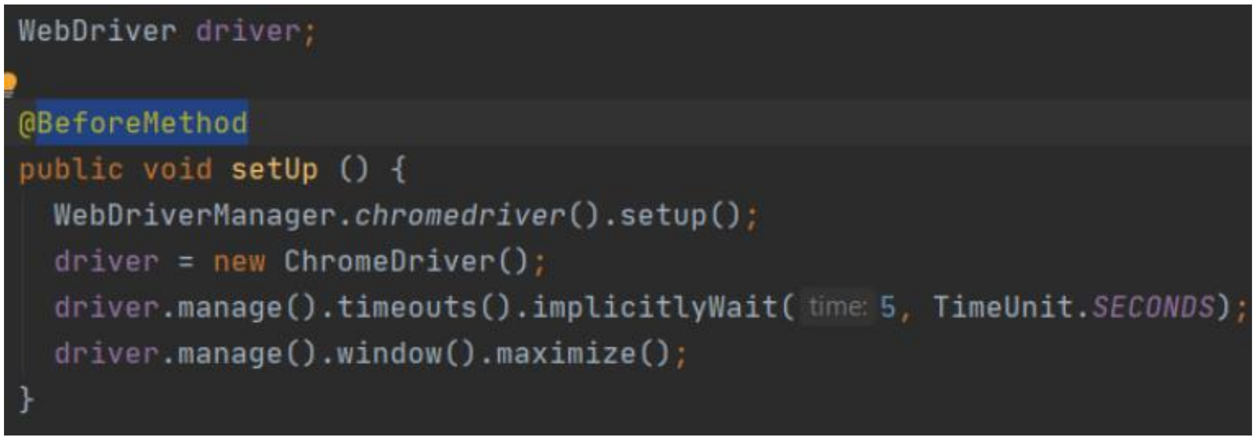
Introduction

Selenium 4 provides 5 Relative Locators to find an element or elements located above(), below(), near(), toLeftOf(), and toRightOf() another element.

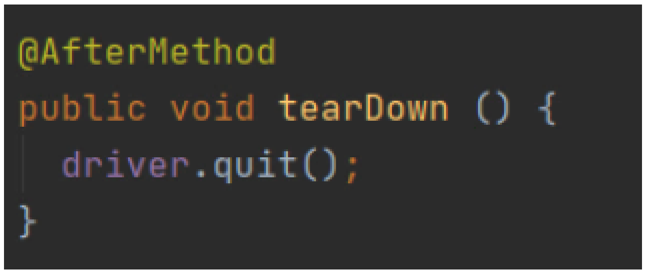
In this session, we will use TestProject's website. It's a free E2E test automation platform for web, mobile, and API. If you are interested in more videos, you can subscribe to my YouTube channel and click the bell icon. Also connect with me on LinkedIn, Facebook, and follow me on Twitter. Lately, I have been slow creating my transcripts but I will make sure to place them on GitHub.

Use One Relative Locator

Okay, let's go ahead and get straight to the point. Relative Locators were formerly called Friendly Locators. The TestNG Configuration methods BeforeMethod and AfterMethod are pre-written. BeforeMethod will set up our test and AfterMethod will tear down our test.

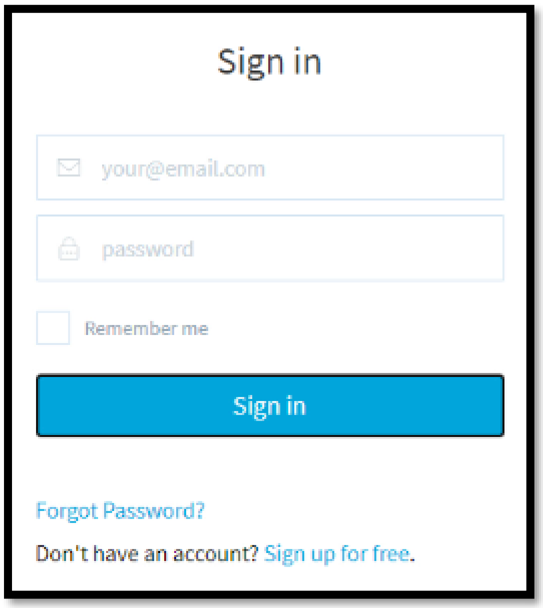


```java
WebDriver driver;

@BeforeMethod
public void setUp () {
  WebDriverManager.chromedriver().setup();
  driver = new ChromeDriver();
  driver.manage().timeouts().implicitlyWait( time: 5, TimeUnit.SECONDS);
  driver.manage().window().maximize();
}
```



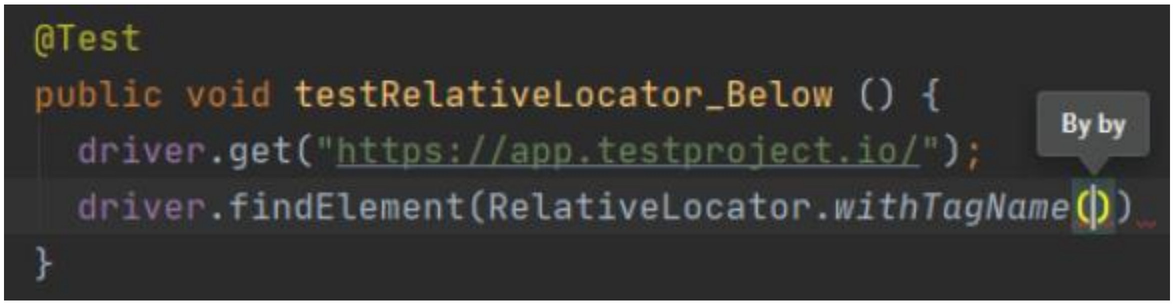
```java
@AfterMethod
public void tearDown () {
  driver.quit();
}
```

Our Test Script is testRelativeLocator_Below. It will find the Sign In button then click the Forgot Password link below the Sign In button.

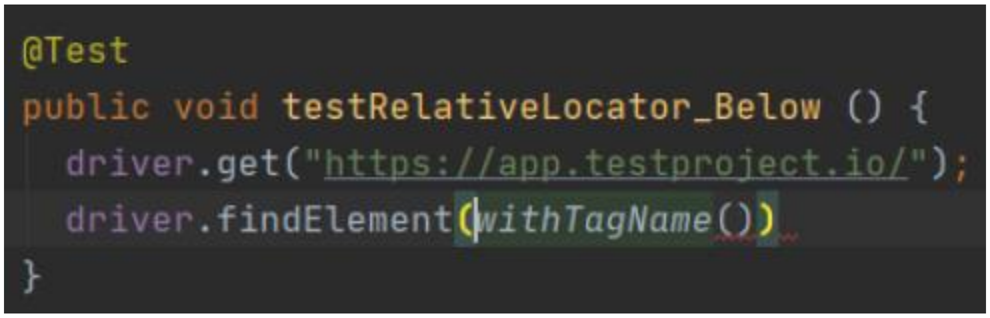


Let's inspect the Sign In button and we see the value for id is tp-sign-in. Inspect the Forgot Password link and we see the value for the Tag Name is 'a' is the tagName. 'a' is for hyperlink. That's all we need to click the Forgot Password link.

There are 2 ways to write a Relative Locator. We start by writing driver.findElement(RelativeLocator.withTagName) and the other way is to bypass writing RelativeLocator and to start by using withTagName. Import.



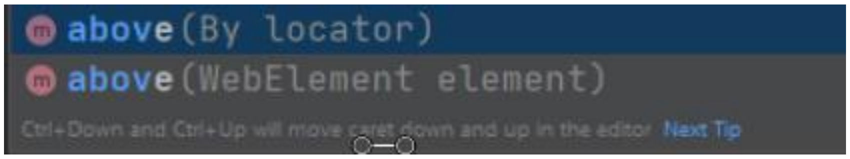
```
@Test
public void testRelativeLocator_Below () {
  driver.get("https://app.testproject.io/");
  driver.findElement(RelativeLocator.withTagName())
}
```



```
@Test
public void testRelativeLocator_Below () {
  driver.get("https://app.testproject.io/");
  driver.findElement(withTagName())
}
```

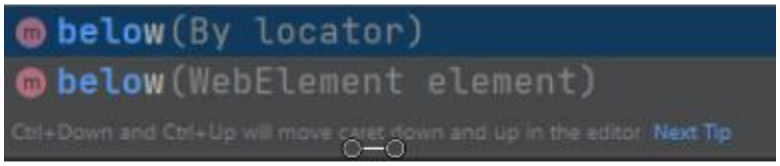
However, there's a difference with the import statements. To use withTagName without writing Relative Locator, we import the package with static. If you want to use the other way with Relative Locator.withTagName. It's the other package without static. By default, Eclipse does not import the static package.

The tagName for Forgot Password is "a" so we enter "a" dot. Here's a list of our Relative Locators. I want you to notice something. Each Relative Locator is overloaded with more than 1 option: above has a By locator parameter and a WebElement element parameter, the same with below: By locator, WebElement element, near is the only locator with 4 options (By locator, WebElement element, By locator, int atMostDistanceInPixels, and WebElement element), int atMostDistanceInPixels), toLeftOf has a (By locator and WebElement element_, the same with toRightOf (By locator and WebElement element).



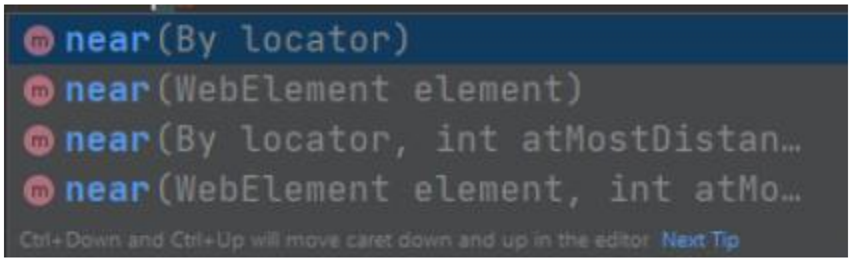
```
above(By locator)
above(WebElement element)
```

Ctrl+Down and Ctrl+Up will move caret down and up in the editor Next Tip



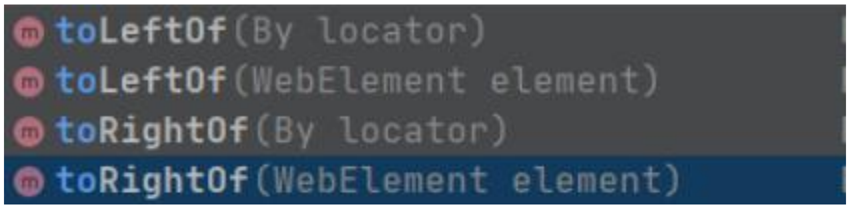
```
below(By locator)
below(WebElement element)
```

Ctrl+Down and Ctrl+Up will move caret down and up in the editor Next Tip



```
near(By locator)
near(WebElement element)
near(By locator, int atMostDistan…
near(WebElement element, int atMo…
```

Ctrl+Down and Ctrl+Up will move caret down and up in the editor Next Tip

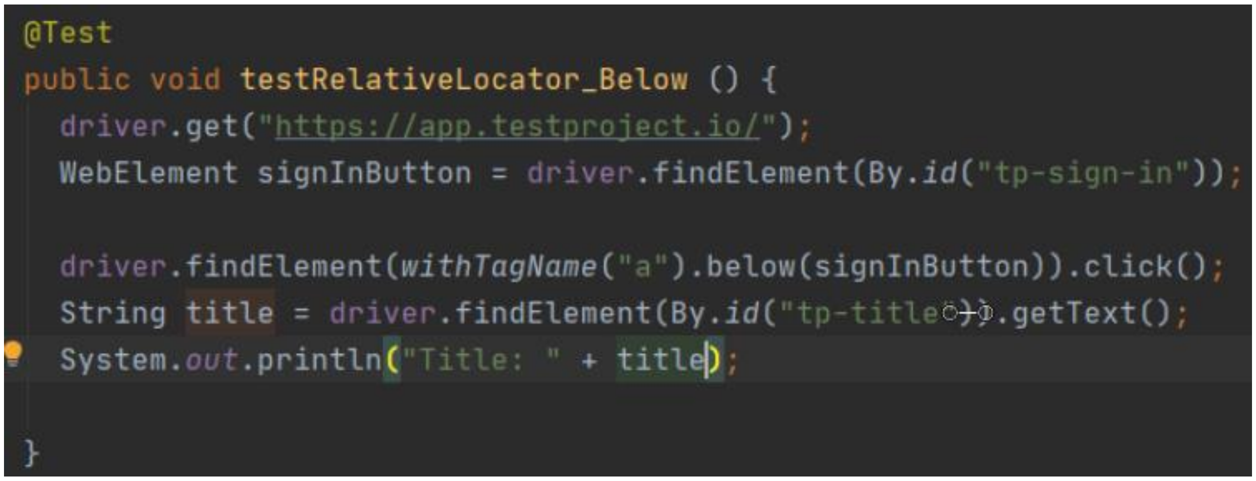


```
toLeftOf(By locator)
toLeftOf(WebElement element)
toRightOf(By locator)
toRightOf(WebElement element)
```

Right now, we are trying to find the Forgot Password link which has a tagName of "a" and located below the Sign In button so we select below with a By locator parameter. We write the locator for the Sign In button: By.id("tp-sign-in")). Now, we can also do 1 more thing after entering the value for id. When it comes to a link Forgot Password, next is to click the Password link. If you wanted to, you can also use WebElement by writing WebElement signInButton = driver.findElement(By.id("tp-sign-in")). Next, we pass in signInButton. Either way is okay. Now we have the Forgot Password link and we are clicking the Password link. Next, let's go ahead and see if we actually land on the next page. Go back to the AUT and click the Forgot Password link and to verify we landed on the Forgot Password. We are going to inspect

Reset Password and it has an id value of tp-title. If we wanted to, we can also use an assert statement but I want to make sure a statement is printed to the Console. Relative Locators can be good for dynamic elements. Dynamic elements are elements like tp-title that changes every time but in this case the value is not dynamic. If the value changes but the tagName remains the same then that's a good reason to use Relative Locator.

Write String title = driver.findElement(By.id("tp-title")).getText());
sysout("Title: " + title)

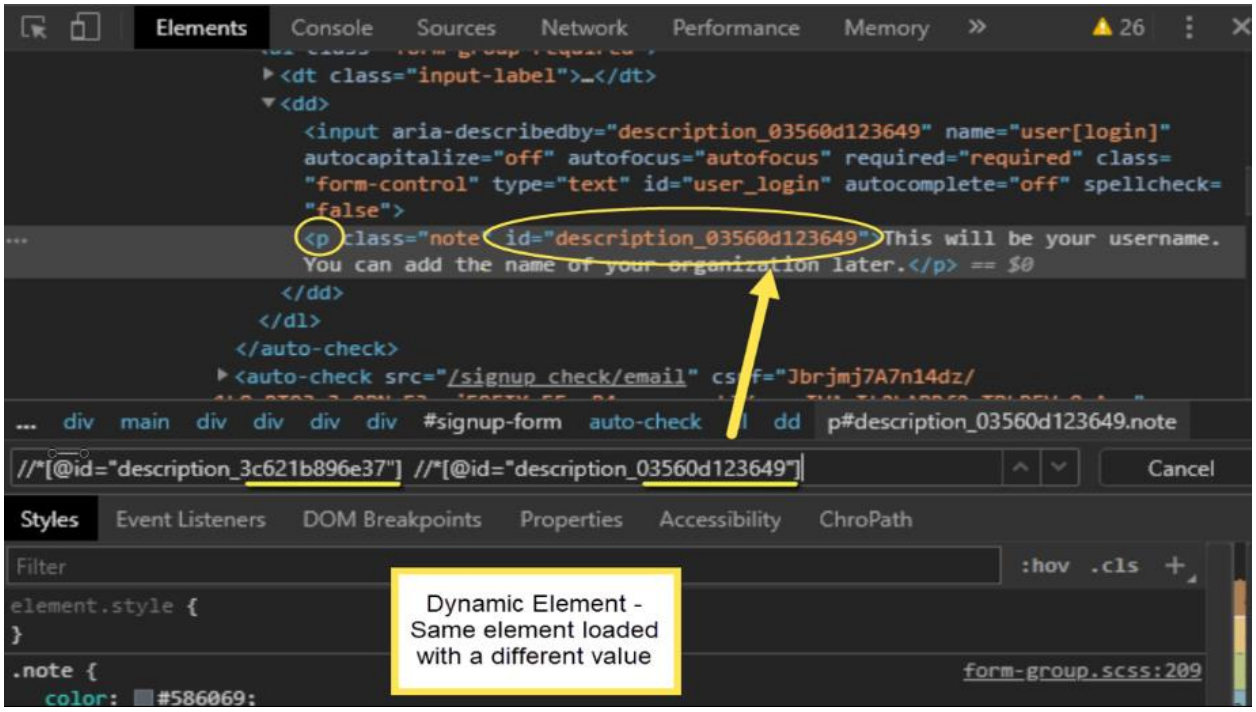


```
@Test
public void testRelativeLocator_Below () {
  driver.get("https://app.testproject.io/");
  WebElement signInButton = driver.findElement(By.id("tp-sign-in"));

  driver.findElement(withTagName("a").below(signInButton)).click();
  String title = driver.findElement(By.id("tp-title")).getText();
  System.out.println("Title: " + title);

}
```

Let's Run. Bingo Title = Reset Password.

Use Multiple Relative Locators

In this session, we will look at the benefit of using more than 1 Relative Locator. I mentioned in the previous session that Relative Locators are good for Dynamic Elements. Here's an example of a dynamic element that I covered in my xpath video 87 talking about "Why We Should Never Copy XPath Values". There are 2 id values. On the left, the value ends with 37 and on the right, the value ends with 49. In this screenshot, the arrow is pointing to the most recent id value that ends with 49. The value changed but the tagName remained at p for paragraph.



```
▶<dt class="input-label">…</dt>
▼<dd>
   <input aria-describedby="description_03560d123649" name="user[login]"
   autocapitalize="off" autofocus="autofocus" required="required" class=
   "form-control" type="text" id="user_login" autocomplete="off" spellcheck=
   "false">
   <p class="note" id="description_03560d123649">This will be your username.
   You can add the name of your organization later.</p> == $0
 </dd>
</dl>
</auto-check>
▶<auto-check src="/signup_check/email" csrf="Jbrjmj7A7n14dz/
```

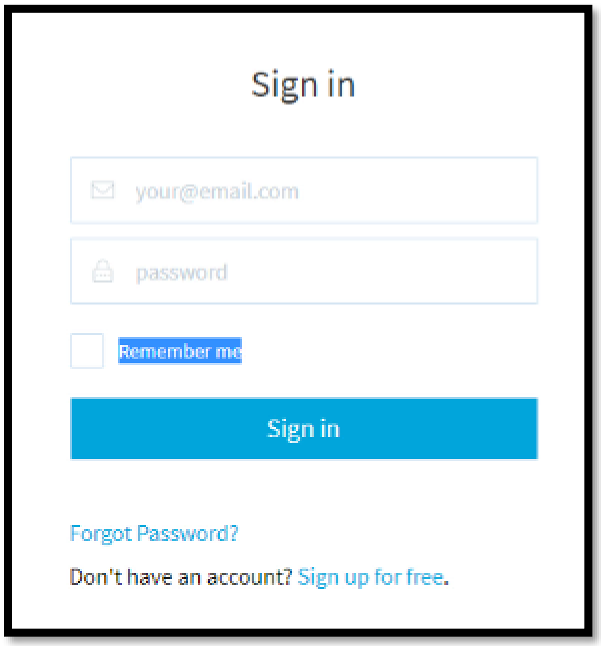
```
... div main div div div div #signup-form auto-check dl dd p#description_03560d123649.note
```

```
//*[@id="description_3c621b896e37"] //*[@id="description_03560d123649"]
```

```
element.style {
}
.note {
  color: #586069;
```

```
form-group.scss:209
```

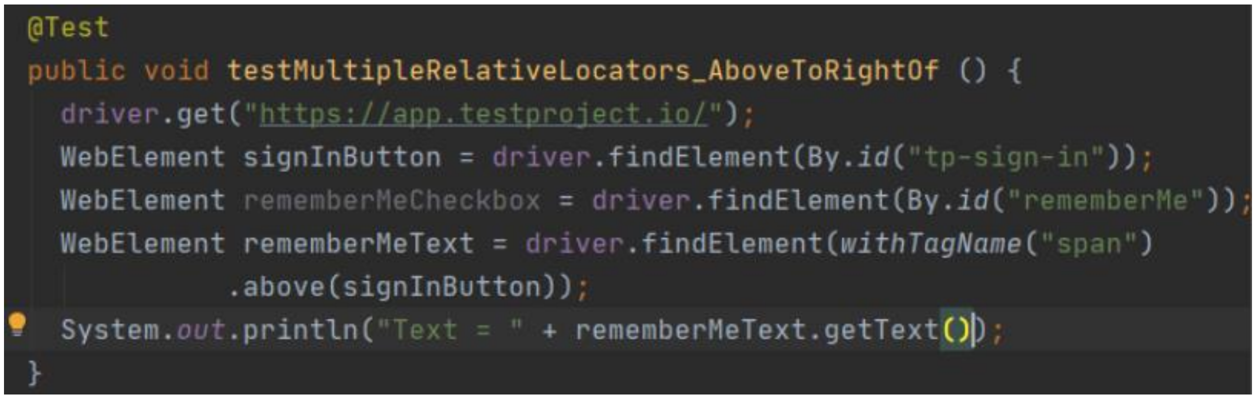
We can include Relative Locators to the other locators such as xpath and cssSelector for finding Dynamic values. For this session, let's use the same Sign In page as the previous session but this time get the text of Remember me which is above the Sign In button and to the right of the checkbox.



Let's inspect the checkbox and the id value is rememberMe. We also need the tagName of the Remember me verbiage. So, let's it Inspect and the tagName is span. We are going back to IntelliJ and I need to copy the element for the Sign In button and paste in our new Test Script. We also need to go ahead and get the value of the Remember Me checkbox so I'm going to write WebElement rememberMeCheckbox = driver.findElement(By.id("rememberMe"));

Now, let's find the Remember Me text by writing WebElement rememberMeText = driver.findElement(withTagName("span"))
The verbiage is located .above(signInButton));

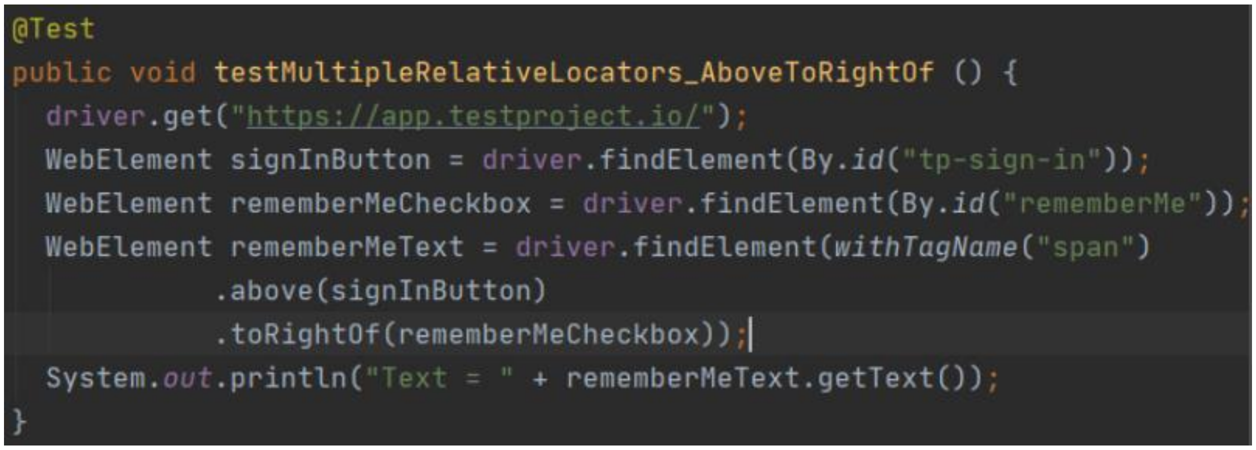
Print the verbiage sout("Text = " + rememberMeText.getText());



```
@Test
public void testMultipleRelativeLocators_AboveToRightOf () {
  driver.get("https://app.testproject.io/");
  WebElement signInButton = driver.findElement(By.id("tp-sign-in"));
  WebElement rememberMeCheckbox = driver.findElement(By.id("rememberMe"));
  WebElement rememberMeText = driver.findElement(withTagName("span")
          .above(signInButton));
  System.out.println("Text = " + rememberMeText.getText());
}
```

Let's run. We see Text is blank. Why is it blank? We see is blank. Why is it blank? It is blank because the checkbox and text are both located above the Sign In button. Our Test Script found the checkbox. Inspect the checkbox again and we see it also has a span tag just like the Remember me. The checkbox was selected for the Relative Locator because it's the first span tag.

Recall, the findElement method finds the first WebElement so that's why checkbox was selected. In this scenario, we need to use multiple Relative Locators because it's more accurate than only using 1 Relative Locator. Remove the semi-colon and add the dot. The text was located above the checkbox and .toRightOf(rememberMeCheckbox);.



```
@Test
public void testMultipleRelativeLocators_AboveToRightOf () {
  driver.get("https://app.testproject.io/");
  WebElement signInButton = driver.findElement(By.id("tp-sign-in"));
  WebElement rememberMeCheckbox = driver.findElement(By.id("rememberMe"));
  WebElement rememberMeText = driver.findElement(withTagName("span")
          .above(signInButton)
          .toRightOf(rememberMeCheckbox));
  System.out.println("Text = " + rememberMeText.getText());
}
```

Now. let’s run again and see what happens. Now, text has a value equal to Remember Me and it PASSED. Next, let’s find a list of WebElements.

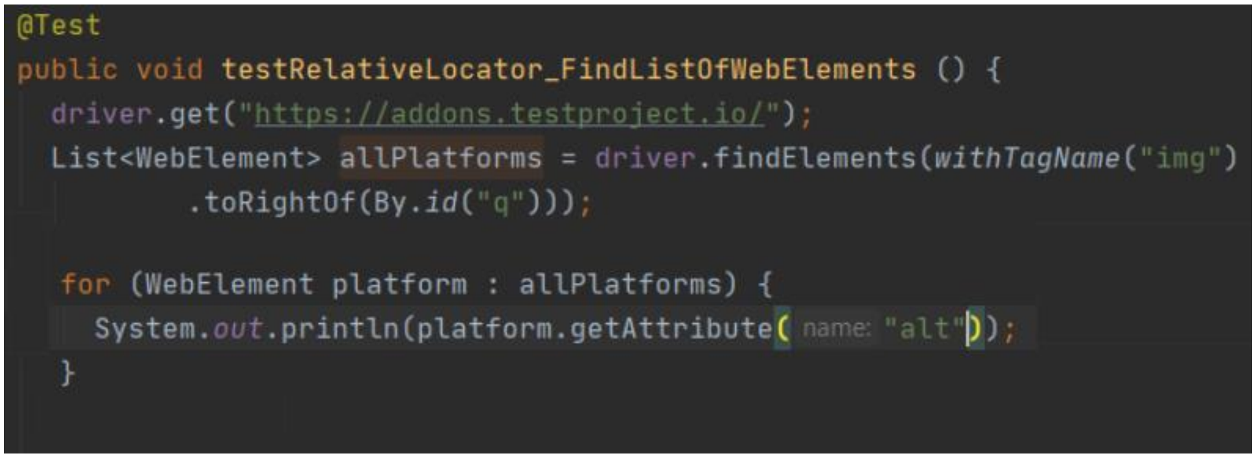
Find List Of Elements

Let’s get the URL for our next Test Script. I’m going to copy https://addons.testproject.io and paste it to take us to the Addons page. We are going to get the names of each platform. If I inspect each icon, the tagName is img and the names are located in attribute alt. We see Platform Web, Platform Android, and the third icon is Platform iOS. They are located to the right of this textbox. Inspect the textbox and the id value is q.

To find a list of elements using a Relative Locator, we start by writing List then <WebElement> and the name will be allPlatforms = driver.findElements(). Find elements with an ‘s’ find all of the elements (withTagName(“img”))
to the right of text box .toRightOf(By.id(“q”)));

Here’s the logic, we must loop through each image then print the name.

for (WebElement platform : allPlatforms) {)
  sout(platform.getAttribute(“alt”));



```java
@Test
public void testRelativeLocator_FindListOfWebElements () {
  driver.get("https://addons.testproject.io/");
  List<WebElement> allPlatforms = driver.findElements(withTagName("img")
          .toRightOf(By.id("q")));

   for (WebElement platform : allPlatforms) {
     System.out.println(platform.getAttribute( name: "alt"));
   }
```

Remember the names were located in the alt attribute so that’s why I wrote getAttribute and not getText. I created a video 12 that explains the difference between getAttribute and getText if you want to check it out. Let’s Run. Bingo, we see all of the platform names: Platform Web, Platform Android, Platform iOS and they PASSED. That’s it for using more than 1 Relative Locator to find a WebElement and how to use a Relative Locator to find a list of WebElements. I have more videos on the way. You can like, subscribe and click the bell icon. Plus connect with me on LinkedIn, Twitter and Facebook.

Contact Info

- ✓ Email Rex.Jones@Test4Success.org
- ✓ YouTube https://www.youtube.com/c/RexJonesII/videos
- ✓ Facebook http://facebook.com/JonesRexII
- ✓ Twitter https://twitter.com/RexJonesII
- ✓ GitHub https://github.com/RexJonesII/Free-Videos
- ✓ LinkedIn https://www.linkedin.com/in/rexjones34/