Rex Jones II

Selenium How To Find An Element In Shadow DOM

In this session, we are going to start by locating an element in the Shadow DOM without using Selenium findElement method. I am going to use the same book application from our Introduction and find this search field.

Inspect. We are not able to locate this search field from the Elements tab because the search field is in a Shadow DOM. Do you see how the id attribute has a value of input? If this were the original DOM, we could use xpath or cssSelector to find this element. For cssSelector, start with the tag name input then id # and the value input. Did not find it. Remove the input tag name and still no success. For xpath, we write //input[@id='input'] and once again we are not able to find the search field.

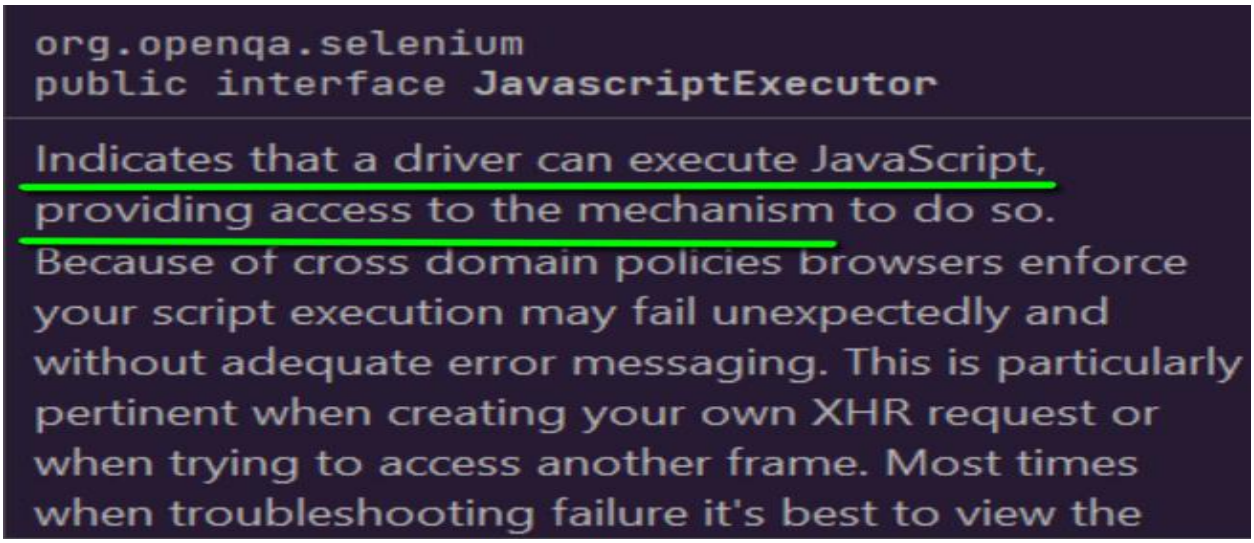
In our diagram, the search field would be an element in the 1st Shadow DOM. Now, let us write our code to find the search field. The setUp method is pre-written for Chrome and it loads the book application. The tearDown method is commented out to quit the driver.

```
public class ShadowDOM {

  WebDriver driver;

  @BeforeMethod
  public void setUp () {
    WebDriverManager.chromedriver().setup();
    driver = new ChromeDriver();
    driver.manage().window().maximize();
    driver.get("https://books-pwakit.appspot.com/");
  }

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

Our method is @Test public void findShadowDOMWithoutSeleniumFindElement () { } We have 3 steps to find an element in the Shadow DOM. 1st we // Provide Access So The Driver Can Execute JavaScript 2nd is to // Execute JavaScript To Return A WebElement 3rd we // Perform An Action On The WebElement

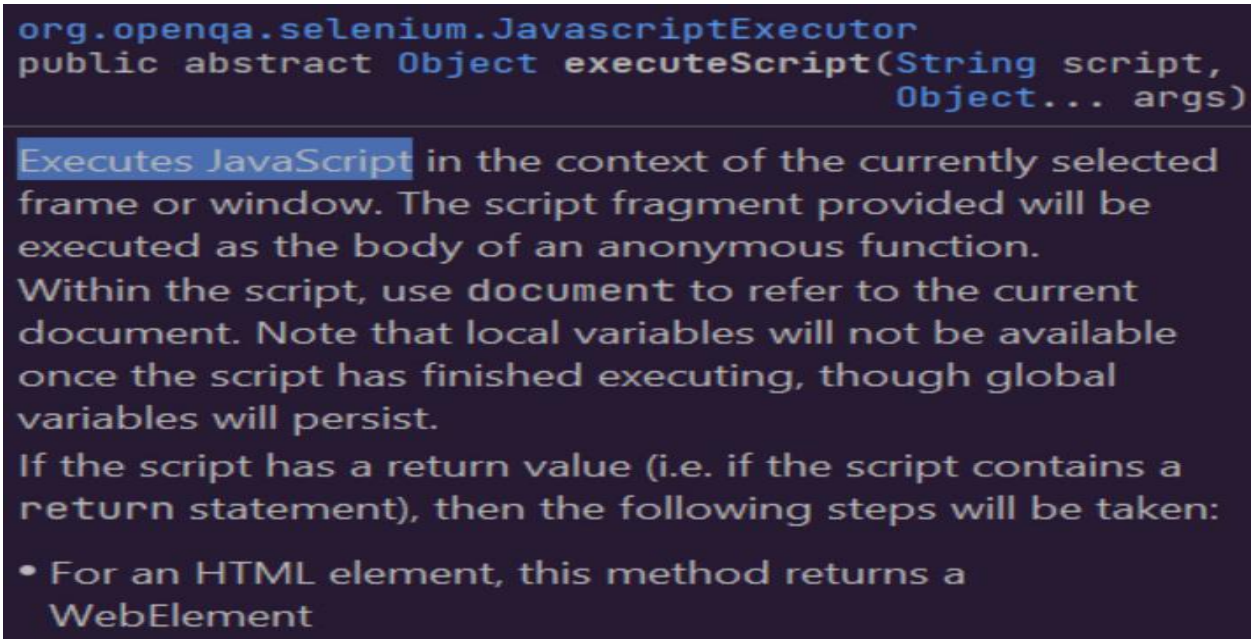
```
@Test
public void findShadowDOMWithoutSeleniumFindElement () {
  // Provide Access So The Driver Can Execute JavaScript

  // Execute JavaScript To Return A WebElement

  // Perform An Action On The WebElement
}
```

JavaScriptExecutor is an interface with jsExecutor; as the reference; JavaScriptExecutor indicates that a driver can execute JavaScript, providing access to the mechanism.



So we convert to = (JavaScriptExecutor) from the driver; At this point, the driver has access to execute JavaScript.

Now, that we have access the jsExecutor. will execute the Script(“”);The executeScript method executes JavaScript. I want you to notice 2 things in this intellisense. #1 Within the script, we use document to refer to the current document and #2 If the script has a return value, then the following steps will be taken. Our test will return an HTML element, so this method returns a WebElement.



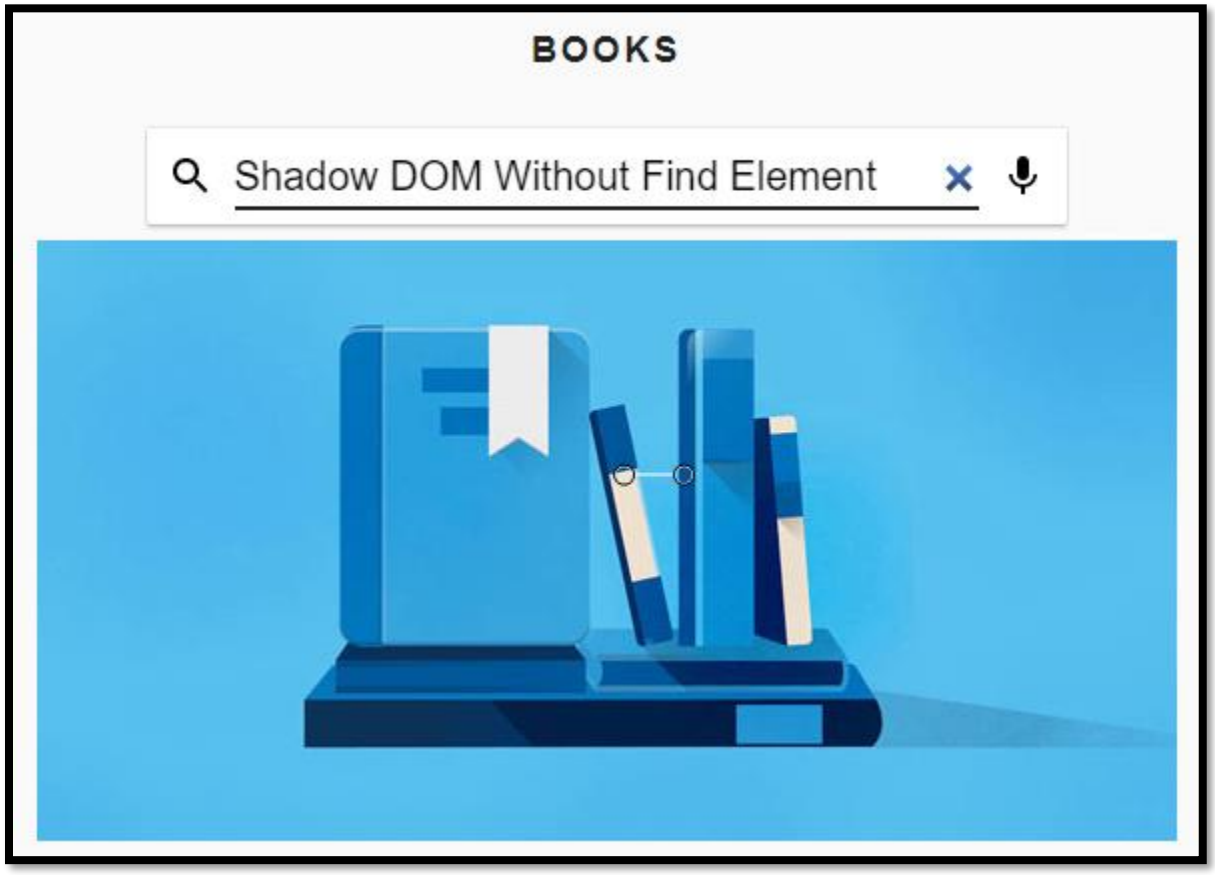
To sum up the information from intellisense, our executeScript method will “return a WebElement and use document”. Since the script returns a WebElement, we cast (WebElement) then assign it to WebElement with a name like bookSearchField =. To complete this script, we need the path from JavaScript.

Go back to our AUT. I’m going to show you 2 more ways for finding this path but for now let’s just keep it simple and we right click the element, select Copy, Copy JS path. That’s it. Go back to our code and Paste the script. This is the path for our bookSearchField and the script returns a WebElement, document represents our web page, querySelector allows us to find the book-app element using cssSelector, shadowRoot is a property representing the shadow root element. Finally, we have the querySelector to find the search field element using cssSelector hashtag #input.

Last step is to perform an action on the WebElement by writing bookSearchField.sendKeys(“Shadow DOM Without Find Element”).

```
@Test
public void findShadowDOMWithoutSeleniumFindElement () {
  // Provide Access So The Driver Can Execute JavaScript
  JavascriptExecutor jsExecutor = (JavascriptExecutor) driver;

  // Execute JavaScript To Return A WebElement
  WebElement bookSearchField = (WebElement) jsExecutor.executeScript(
          script: "return document.querySelector(\"body > book-app\")." +
                  "shadowRoot.querySelector(\"#input\")");

  // Perform An Action On The WebElement
  bookSearchField.sendKeys( ...keysToSend: "Shadow DOM Without Find Element");
}
```

Let's Run. Bingo, in the search field we see Shadow DOM Without Using Find Element.



That's it. Thanks for watch and I'll see you in the next session.

Rex Jones II

Contact Info

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