Rex Jones II

Selenium How To Locate An Element With findElement

Shadow DOM – How To Locate An Element With Find Element

Locating an element using Selenium's findElement method requires us to walkthrough the DOM step-by-step. For an illustration using our diagram, we are going to find the Shadow Host then locate each element in the Shadow Tree. From the previous session, our setup method controlled Chrome and loaded the book application.

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

In this session, we will use the same setup and the Test Script will be @Test public void findShadowDOMWithSeleniumFindElement () {}. We are going to group all of the statements into 3 statements starting with // #1 Find Shadow Host // #2 Execute JavaScript To Return The Shadow Root // #3 Find The WebElement Then Perform An Action On The WebElement

```
@Test
public void findShadowDOMWithSeleniumFindElement () {
  // #1 Find Shadow Host

  // #2 Execute JavaScript To Return The Shadow Root

  // #3 Find The WebElement Then Perform An Action On The WebElement
}
```

The Shadow Host is a WebElement so we write WebElement shadowHost; Now we have to find the element. Recall from the diagram, the shadow host comes before the Shadow Tree and a Shadow Tree

is a tree in the Shadow DOM. All Shadow Trees begin with a shadow-root. Go to our AUT. Inspect the search field. We see book-app comes before shadow-root.

Therefore, our code will be driver.findElement(By.tagName("book-app"));

```
@Test
public void findShadowDOMWithSeleniumFindElement () {
  // #1 Find Shadow Host
  WebElement shadowHost = driver.findElement(By.tagName("book-app"));
```

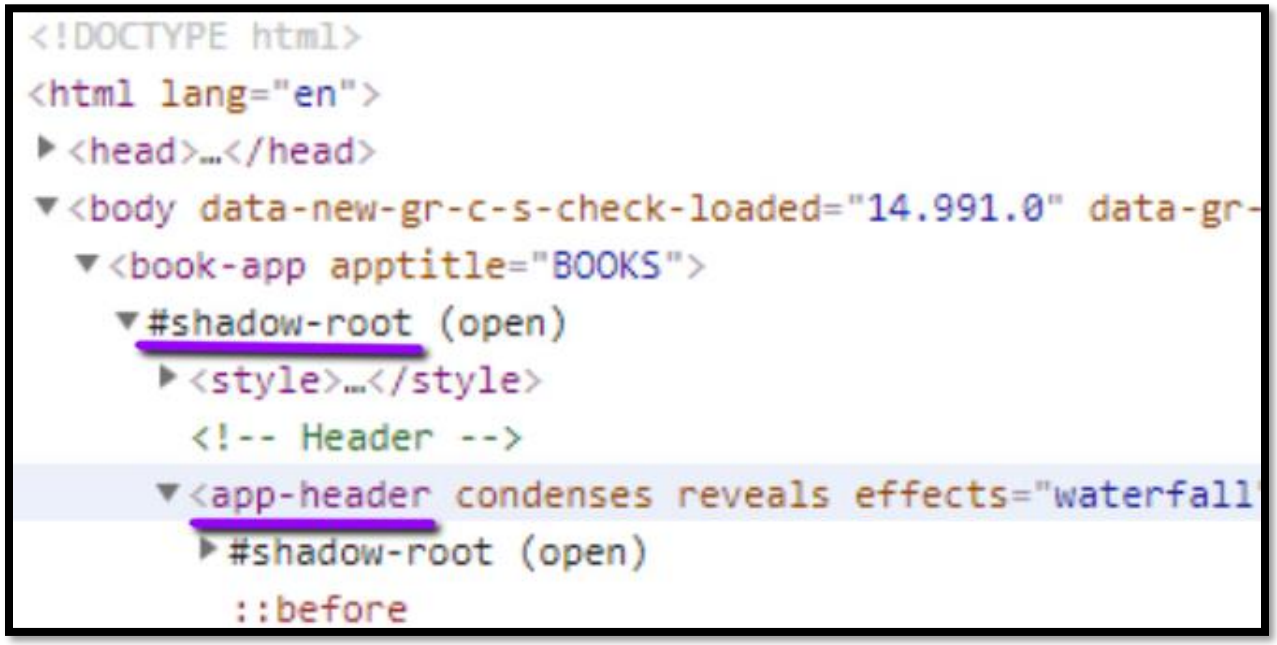
We have 2 statements for JavaScript to return the Shadow Root. The first statement is to cast (JavascriptExecutor) from the driver then assign it to JavascriptExecutor with any reference name like jsExecutor =. The next statement will cast (WebElement) and use jsExecutor. to execute Java Script(). We execute JavaScript with 2 parameters. ("return arguments[0].shadowRoot", and shadowHost).

The first parameter returns the 1st argument representing the Shadow Root element. Notice the 2nd parameter is shadowHost. We need the shadowHost to access the WebElement for shadowRoot =. The shadowRoot is located inside of the Shadow Host.

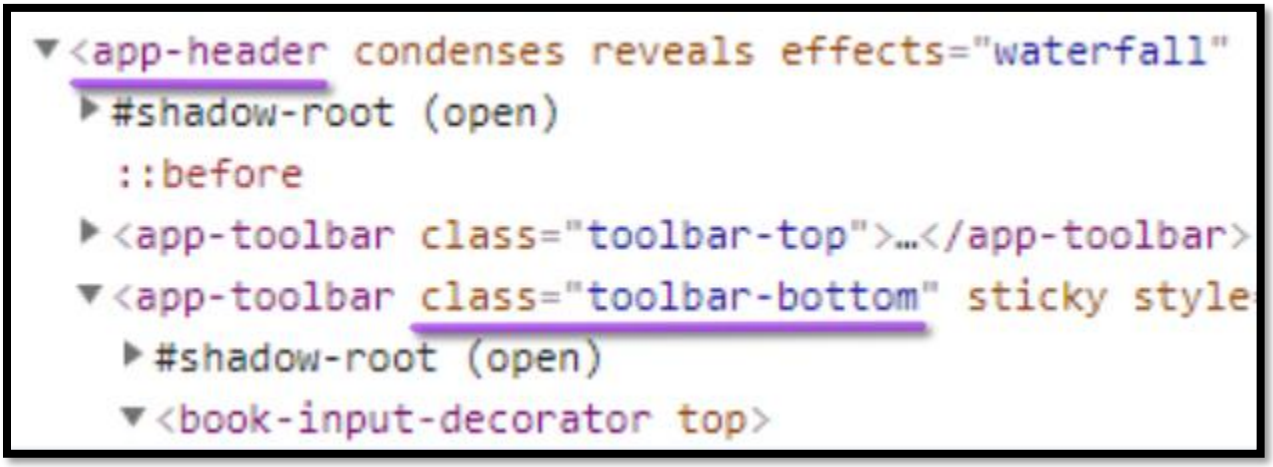
```
@Test
public void findShadowDOMWithSeleniumFindElement () {
  // #1 Find Shadow Host
  WebElement shadowHost = driver.findElement(By.tagName("book-app"));

  // #2 Execute JavaScript To Return The Shadow Root
  JavascriptExecutor jsExecutor = (JavascriptExecutor) driver;
  WebElement shadowRoot = (WebElement) jsExecutor.executeScript(
          script: "return arguments[0].shadowRoot", shadowHost);
```

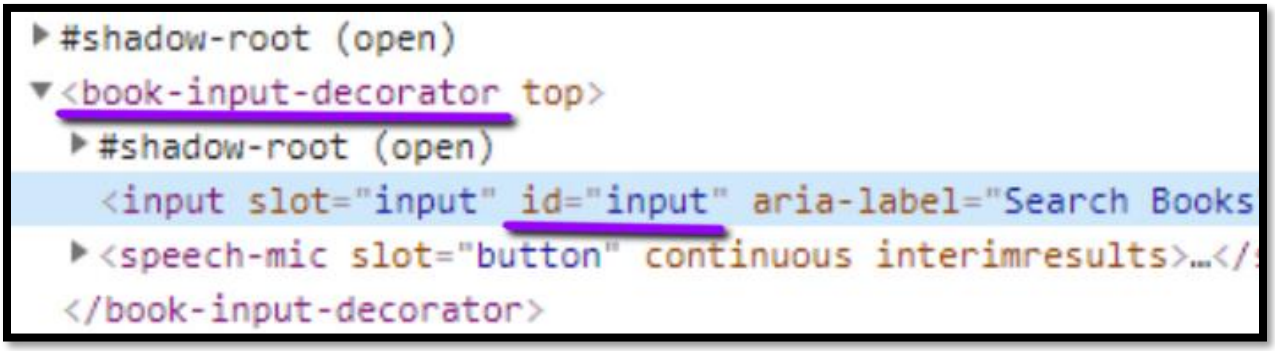
Now we can use the shadow root for the WebElement app_header; Go back to our AUT and we see app-header is below shadow-root.



For our code, each WebElement will build on top of the previous WebElements. We are going to use shadow-root and the app-header tag name. = shadowRoot. then find the element (By.tagName("app-header")); The next element is app-toolbar but there are 2 tags with the same name. So we must identify the element using the class name which is toolbar-bottom. Just like the last statement. We build on the current element using the previous element. So we use app-header and toolbar-bottom.



WebElement app_toolbar = previous WebElement app_header.findElement(By.) At this point, we can find the element by Selenium locators className or cssSelector. className would be (“toolbar-bottom”) but let’s use cssSelector(“.toolbar-bottom”)); The next tag is book-input-decortator then the book search field which has an id value of input.



We are going to build on top of app-toolbar then find book-input-decorator followed by the search field. So the WebElement will be WebElement book_input_decorator = app_toolbar.findElement(By.). If we wanted to, we can combine the book-input-decorator element with the search field using cssSelector(“”). The value would be both tag names (“book-input-decorator > input”). However, let’s break it down and use only the tagName(“book-input-decorator”). Now, we are at the last WebElement which is the WebElement for searchField = book_input_decorator.findElement(By.id(“input”)). Now that we have found our WebElement, let’s perform an action on the WebElement by writing searchField.sendKeys(“Shadow DOM With Find Element”);

```
// #3 Find The WebElement Then Perform An Action On The WebElement
WebElement app_header = shadowRoot.findElement(By.tagName("app-header"));
WebElement app_toolbar
        = app_header.findElement(By.cssSelector(".toolbar-bottom"));
WebElement book_input_decorator
        = app_toolbar.findElement(By.tagName("book-input-decorator"));
WebElement searchField = book_input_decorator.findElement(By.id("input"));
searchField.sendKeys( ...keysToSend: "Shadow DOM With Find Element");
```

Let's run. We see the search field shows Shadow DOM With Find Element. Thanks for watching and I'll see you in the next section for Nested Shadow DOM's.

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/