Locate DOM Elements

Related pages:

- Write Your First Script
- Wait for DOM Elements
- Work with Screenshots
- Add Logs to Troubleshoot
- Verify Program Correctness


Your synthetic script simulates end-user interactions, so you'll need to be able to locate the DOM elements of your web pages. For example, your script might click on buttons, links, or enter text into text fields: this requires your script being able to locate and then select relevant HTML elements.

Methods for Locating Elements

The Selenium WebDriver library provides CSS selectors and XPath statements for selecting HTML elements. See 4. Locating Elements for the list of the library methods and usage examples.

You can also use the AppDynamics WebDriver Scripting Assistant, a Chrome extension, to help you create the selector statements.

Best Practices for Locating Elements

When locating elements, you are recommended to do the following. Think of it as a checklist.

- Understand the following about your application:
    - the DOM structure
    - which pages are dynamically and statically loaded
    - which elements of a page are loaded and visible.
- Use unique IDs for elements and selectors that are as short as possible: Selectors break all the time. Long hierarchical selectors break more easily than shorter ones, and using short selectors will reduce script maintenance over time.
- Click on the user-visible element instead, or send a **Return** key instead of submitting forms using the method `submit()`.

- Pay attention to element visibility. The specification is complex, and the results are not always what users expect. See the WebDriver specification for more information.