CS1632, LECTURE 12: SYSTEMS TESTING THE WEB WITH Selenium

Wonsun Ahn

Background

- Systems testing: testing the entire system as a whole
  - We would like to automate systems testing, just like we did for unit testing
- So far, all of our testing has been text-based java programs
  - A.k.a. Command Line Interface (CLI) programs
- Automating testing for CLI programs is easy!
  - Just create an “input script” and redirect to stdin
  - Redirect stdout to file and compare to expected output

Example Automated Systems Test for CLI Programs

```
#!/bin/bash

# Test case 1: Drink empty coffee and lose
echo -e "D\n" | java -jar coffeemaker.jar > observed.out
diff observed.out expected.lose.out

# Test case 2: Drink properly brewed coffee and win
echo -e "L\nN\nN\nL\nN\nN\nN\nL\nD\n" | java -jar
coffeemaker.jar > observed.out
diff observed.out expected.win.out
```

Automated Systems Test for GUI Programs

- Turns out not that not every program is a CLI program
  - GUIs, web pages, mobile applications, etc.
  - How do we deal with these?

- We need different tools to test more complicated interfaces

- But the basic concepts remain the same
  - Compare **observed behavior vs. expected behavior**
  - Preconditions
  - Execution steps
  - Postconditions

Testing the Web

- Web app: An example of a GUI app – other GUI apps work similarly
- Insight: GUI apps are also in the end … just text

- Web app = text
  - Text in the form of HTML (HyperText Markup Language)
  - HTML elements are displayed on the web browser
- Mobile app = text
  - Text in the form of XML (Extensible Markup Language)
  - XML elements like buttons and menus are displayed on the app

Theoretically, we could test web pages like so

```
// Any downsides to this?
@Test
public void testWeb() {
    String expectedHtml =
“<html><head></head><body>Hello world</body></html";
    String pageText = getPage("http://example.com");
    assertEquals(expectedHtml, pageText);
}
```

- Test text output of a web app just like you would for CLI programs

Downsides

1. Unreadable
   - Unclear which value in which element in the HTML page tester is trying to check.
2. Fragile tests
   - Changing any part of the page (even unrelated to test case) will break the test.
3. What about JavaScript?
   - Just check JS code letter by letter?
   - Remember JS is a program. We should check functionality not code.
4. Simplistic and low-level
   - No semantic understanding (e.g. of links, textboxes)
   - No sense of HTML or XML hierarchy
   - Tries to deal with a flat text string – almost like assembly programming

Web Testing Frameworks

- Sure, you could program everything in assembly
  - You could yourself parse HTML into a tree data structure
  - You could yourself interpret and run JS code on HTML elements
- Why not have a framework that does all of that for you?
  - That creates the HTML tree for you and provides APIs to search it
  - That handles events like clicking or typing by running JS code
- That framework is a *web testing framework*
  - Includes *web drivers* for each type of web browser
  - Framework often also includes an IDE to auto-generate a test script, by simply interacting with the website on a web browser.

What is a Web Driver?

- Web driver (according to w3.org)
  - A remote control interface to instruct the behavior of web browsers
  - Primarily intended to allow writing of automated tests
  - Also used to write scripts to automate repeated tasks (e.g. ordering delivery)

- Capabilities
  - APIs to find HTML elements and access their values
  - APIs to generate events that emulate user interactions
  - Runs web app JavaScript code in response to those events
  - Platform- and language-neutral

Selenium = Web Driver + IDE

- Selenium: An open-source web testing framework
  - Licensed under Apache License 2.0
  - Works with Windows, OS X, Linux, other OSes
  - Works with Java, Ruby, Python, other languages
  - Works with Chrome and Firefox (Edge uses the Chrome engine nowadays)
  - https://www.selenium.dev/

- Selenium IDE: Browser extension for automated test script generation
  - https://www.selenium.dev/selenium-ide/

Getting Started with Selenium IDE

1. Go to https://www.selenium.dev/selenium-ide/

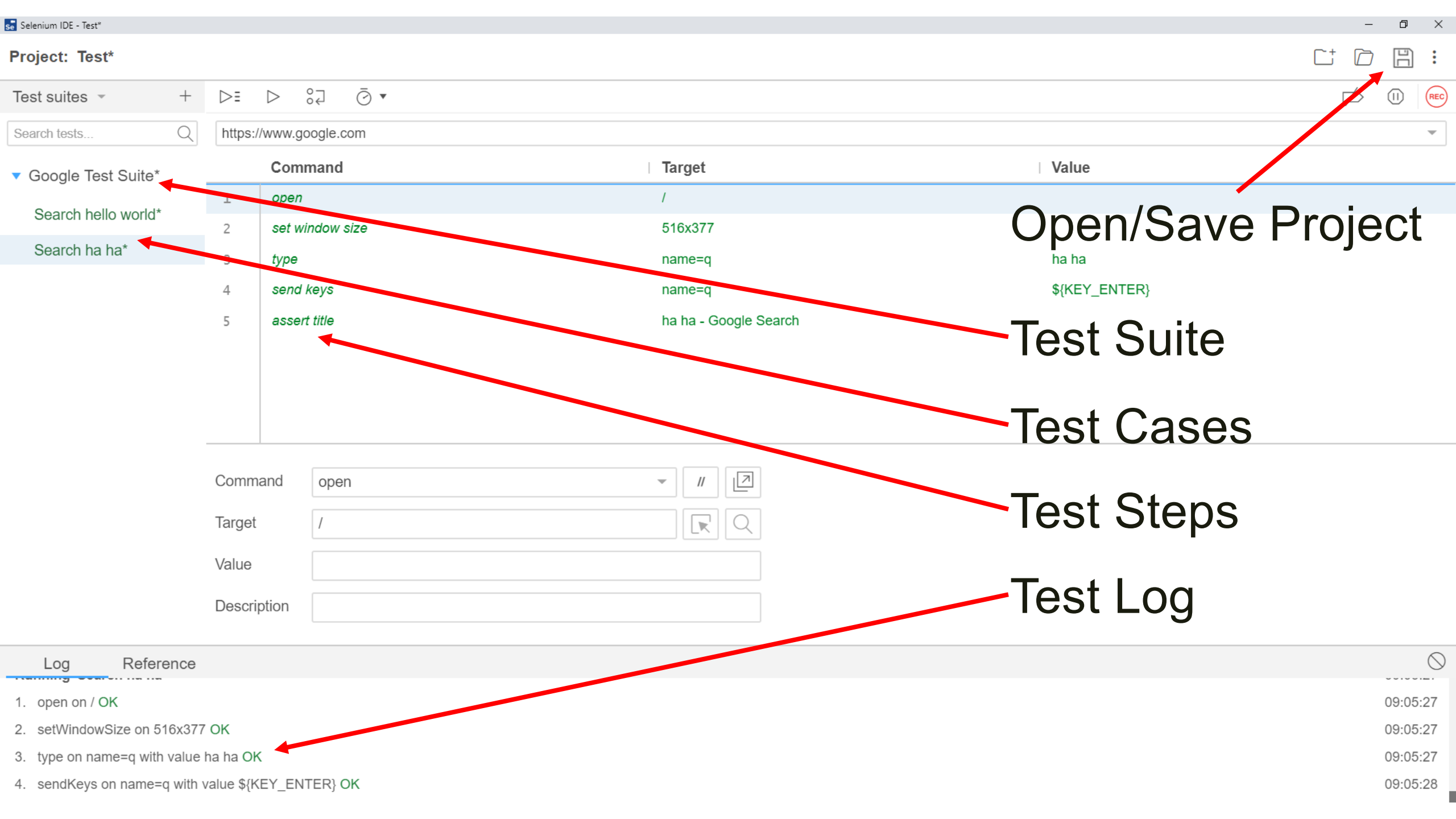
2. Add the appropriate extension for your web browser
   (By clicking on Chrome Download or Firefox download)

3. Install extension when taken to the webstore page

4. Click on the "Se" icon in the upper right-hand corner

Selenium IDE

- What we would call a “test plan”, Selenium calls a “test suite”
- Test suites contain test cases
- Test cases contain test steps


Selenium IDE - Test*
Project: Test*
Test suites
Search tests...
https://www.google.com
Google Test Suite*
Search hello world*
Search ha ha*
Command
Target
Value
1 open /
2 set window size 516x377
3 type name=q ha ha
4 send keys name=q ${KEY_ENTER}
5 assert title ha ha - Google Search
Command open
Target /
Value
Description
Log
Reference
1. open on / OK 09:05:27
2. setWindowSize on 516x377 OK 09:05:27
3. type on name=q with value ha ha OK 09:05:27
4. sendKeys on name=q with value ${KEY_ENTER} OK 09:05:28
Open/Save Project
Test Suite
Test Cases
Test Steps
Test Log

Creating a Simple Test Case

1. Create a new test case
   - Select “Tests” in upper left dropdown box
   - Click on the “+” button
2. Record an operation (Press “REC” button)
3. Do something
4. Stop recording
5. Run test case - it does what you just did

Selenium IDE - Test

Project: Test

Tests

Search tests...

Search ha ha

Search hello world

https://www.google.com

1
2
3
4
5

Command open

Target /

Value

Description

Log Reference

1. open on / OK 09:05:27
2. setWindowSize on 516x377 OK 09:05:27
3. type on name=q with value ha ha OK 09:05:27
4. sendKeys on name=q with value ${KEY_ENTER} OK 09:05:28

Add new test case

TEST CASE NAME

Search yolo

ADD CANCEL

Type here to search

9:23 AM
2/24/2020

Selenium IDE - Test*

Project: Test*

Tests

Search tests...

- Search ha ha
- Search hello world
- Search yolo

https://www.google.com

Command	Target	Value

Record

Command

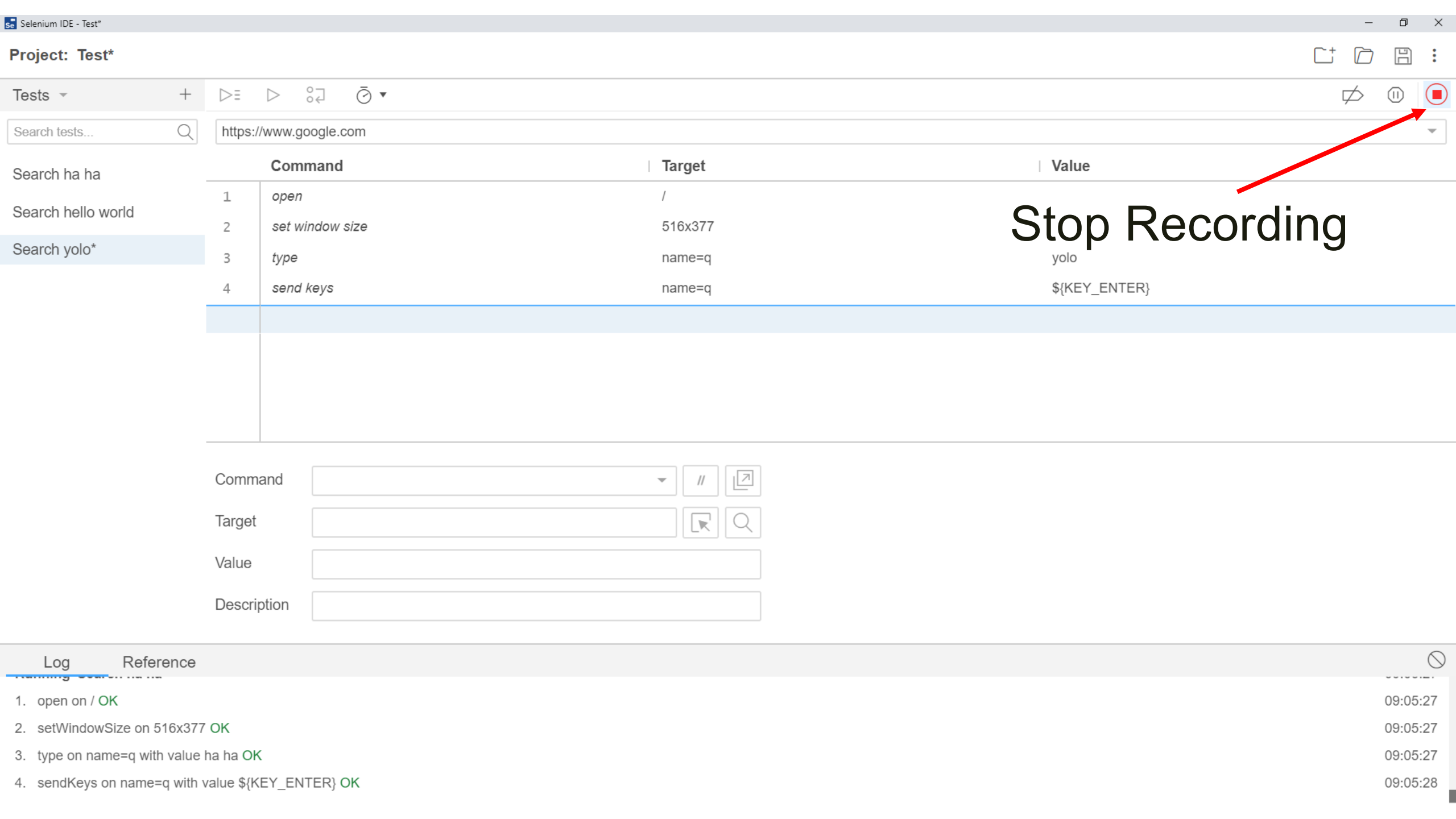
Target

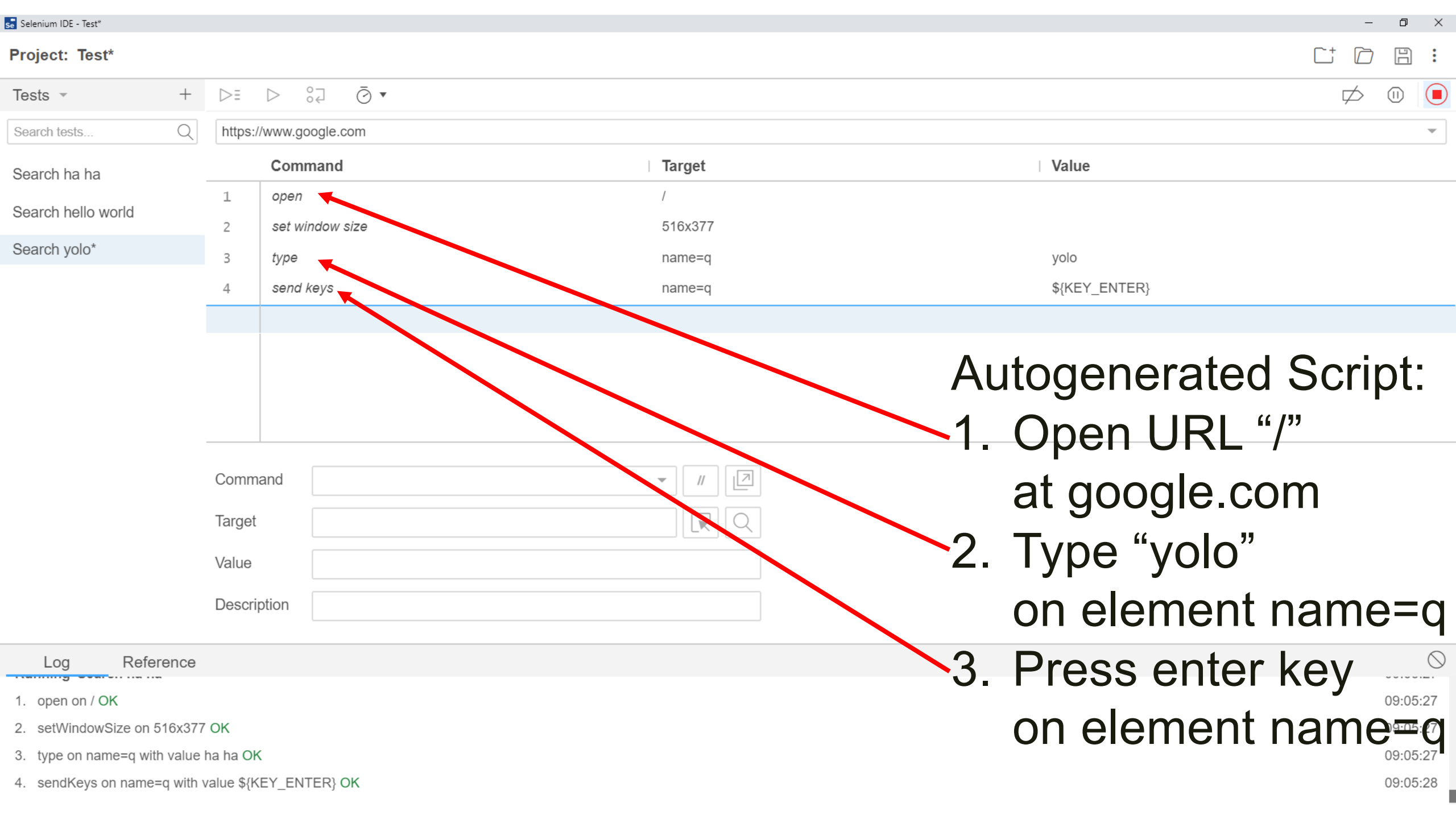
Value

Description

Log Reference

1. open on / OK 09:05:27
2. setWindowSize on 516x377 OK 09:05:27
3. type on name=q with value ha ha OK 09:05:27
4. sendKeys on name=q with value ${KEY_ENTER} OK 09:05:28


Selenium IDE - Test*
Project: Test*
Tests
Search tests...
Search ha ha
Search hello world
Search yolo*
https://www.google.com
Command
Target
Value
1 open / 516x377
2 set window size 516x377
3 type name=q yolo
4 send keys name=q ${KEY_ENTER}
Stop Recording
Command
Target
Value
Description
Log
Reference
1. open on / OK 09:05:27
2. setWindowSize on 516x377 OK 09:05:27
3. type on name=q with value ha ha OK 09:05:27
4. sendKeys on name=q with value ${KEY_ENTER} OK 09:05:28


Selenium IDE - Test*
Project: Test*
Tests
Search tests...
Search ha ha
Search hello world
Search yolo*
https://www.google.com
Command
Target
Value
1 open / 516x377
2 set window size 516x377
3 type name=q yolo
4 send keys name=q ${KEY_ENTER}
Command
Target
Value
Description
Log
Reference
1. open on / OK 09:05:27
2. setWindowSize on 516x377 OK
3. type on name=q with value ha ha OK 09:05:27
4. sendKeys on name=q with value ${KEY_ENTER} OK 09:05:28
Autogenerated Script:
1. Open URL "/" at google.com
2. Type "yolo" on element name=q
3. Press enter key on element name=q

You can add your own commands

- Modify a recorded script or create from scratch
- Click on the row below the last command to add a test step
- Note that it is NOT a textbox, so it is a little awkward to use

The test step is divided into three parts

- Command - What to do
  - E.g. open a page, click on something, type
- Target - To what?
  - E.g. A URL or an element on the page
- Value - How?
  - E.g. Type what?

Common Commands

- open - open a URL
- click - click on a web element
- type - type something in a web element
- assert - assert that something is true
  - The raison d'être of a testing script

Selenium IDE - Test*

Project: Test*

Tests

Search tests...

- Search ha ha
- Search hello world
- Search yolo*

https://www.google.com

	Command	Target	Value
1	*open*	/	
2	*set window size*	516x377	
3	*type*	name=q	yolo
4	*send keys*	name=q	${KEY_ENTER}
5	//		

Command

Target

Value

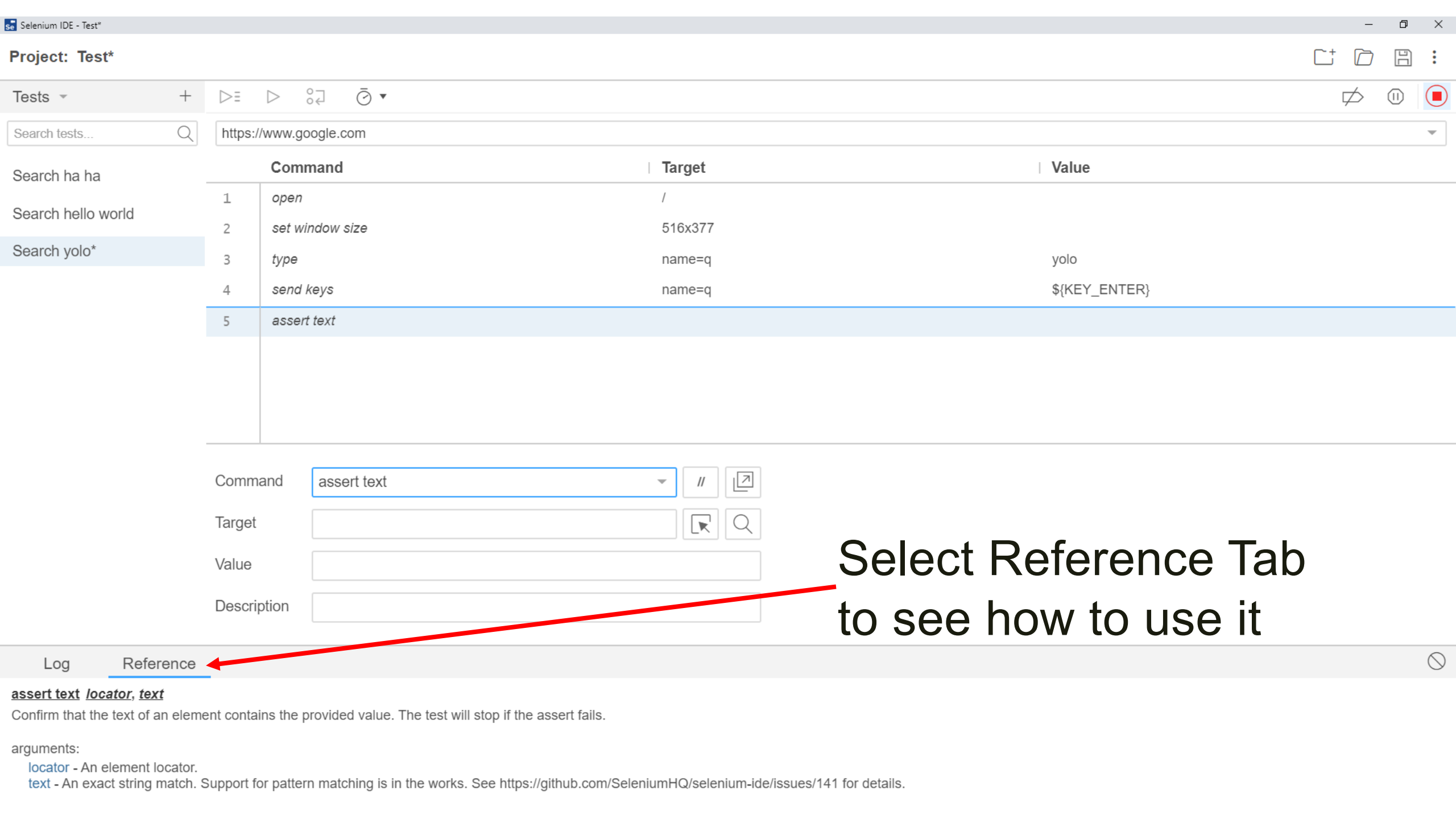
Description

- add selection
- answer on next prompt
- assert
- assert alert
- assert checked
- assert confirmation
- assert editable

Pulldown to peruse;
Type to search command

Log Reference

1. open on / OK 09:05:27
2. setWindowSize on 516x377 OK 09:05:27
3. type on name=q with value ha ha OK 09:05:27
4. sendKeys on name=q with value ${KEY_ENTER} OK 09:05:28


Selenium IDE - Test*
Project: Test*
Tests
Search tests...
Search ha ha
Search hello world
Search yolo*
https://www.google.com
Command
Target
Value
1 open /
2 set window size 516x377
3 type name=q yolo
4 send keys name=q ${KEY_ENTER}
5 assert text
Command assert text
Target
Value
Description
Select Reference Tab
to see how to use it
Log
Reference
assert text locator, text
Confirm that the text of an element contains the provided value. The test will stop if the assert fails.
arguments:
locator - An element locator.
text - An exact string match. Support for pattern matching is in the works. See https://github.com/SeleniumHQ/selenium-ide/issues/141 for details.

Hello, assertions, my old friend…
I've come to assert you again..

- We are going to use assertions to specify expected behavior

- Same as traditional Junit assertions, just at different level of abstraction

A subset of Selenium assertions…

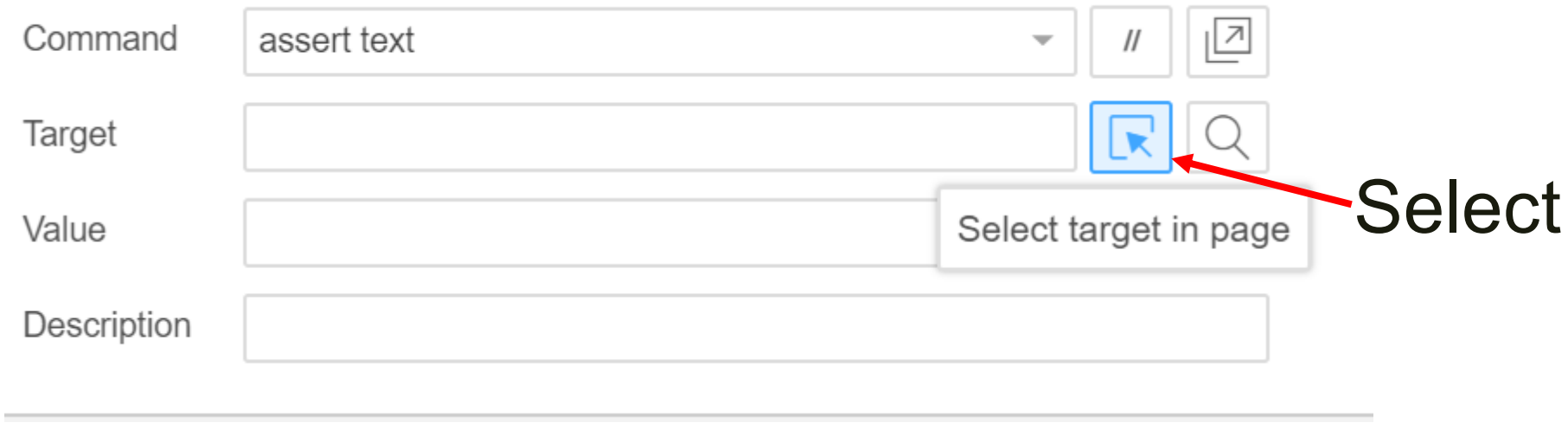
- assertText - Assert that text for element equals a regex
- assertTextPresent - Assert that regex exists somewhere on the page
- assertElementPresent - Assert that element exists somewhere on page
- assertCookie - Assert that a cookie exists
- assertAlert - Assert that an alert took place
- assertEditable - Assert that an element is editable
- assertEval - Evaluate some JavaScript and assert the result

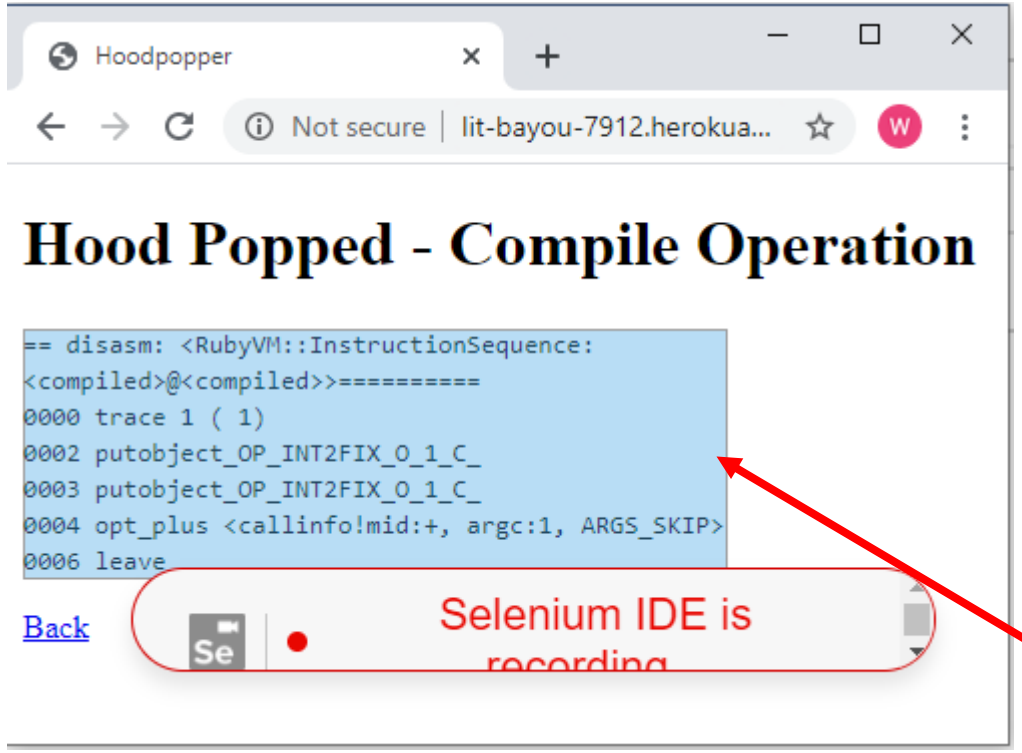
I want to assert something about this particular text section…
but how?

Select can help specify a target

- Lots of ways to specify an element on a webpage
  - CSS
  - xpath
  - id
  - Other tag
- Select can help you find one that works
  - It will find a value which uniquely identifies that element

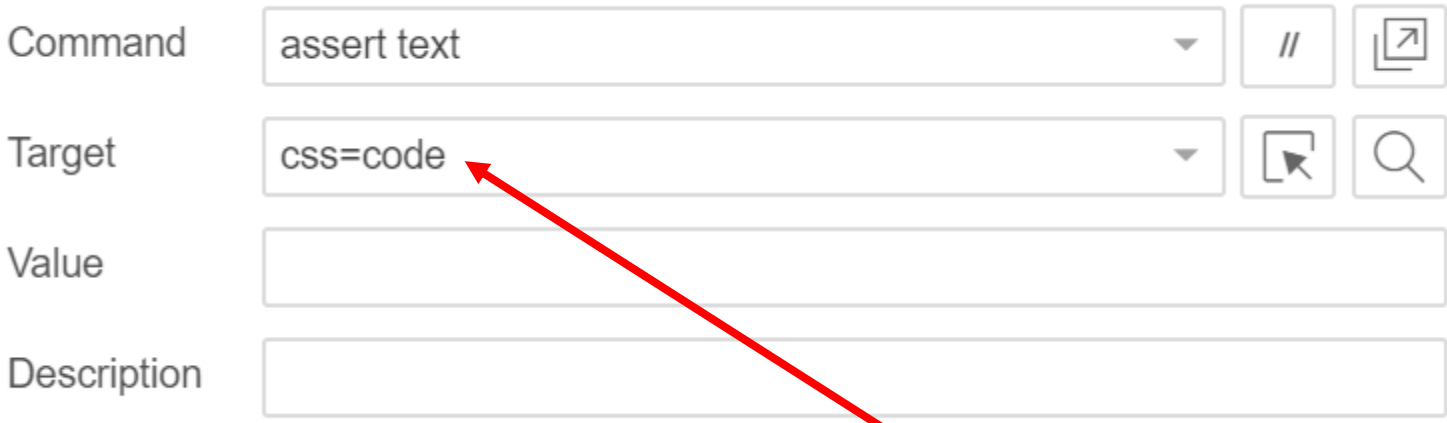
Selecting an HTML element





Click on HTML element you want to select

Autogenerated target!

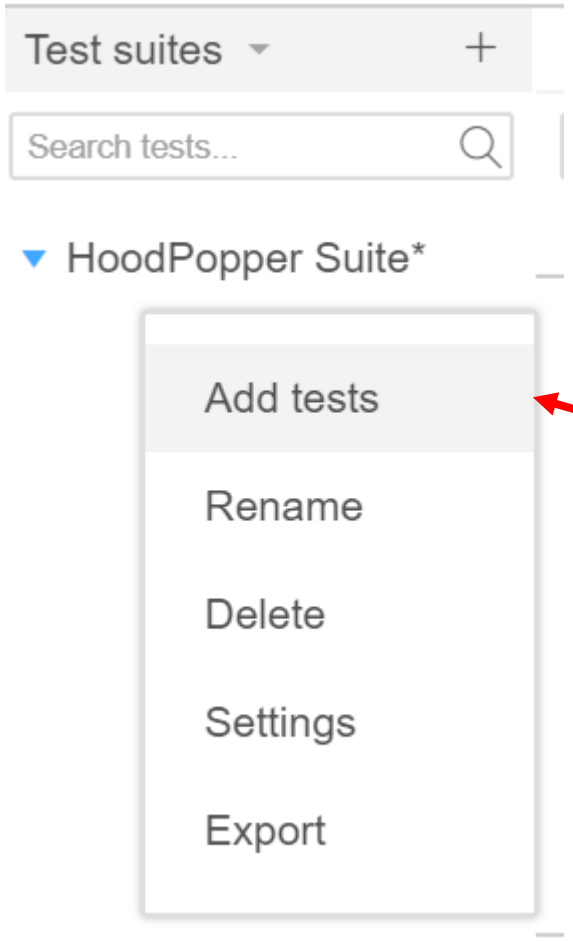


Finds a way to uniquely specify that element!

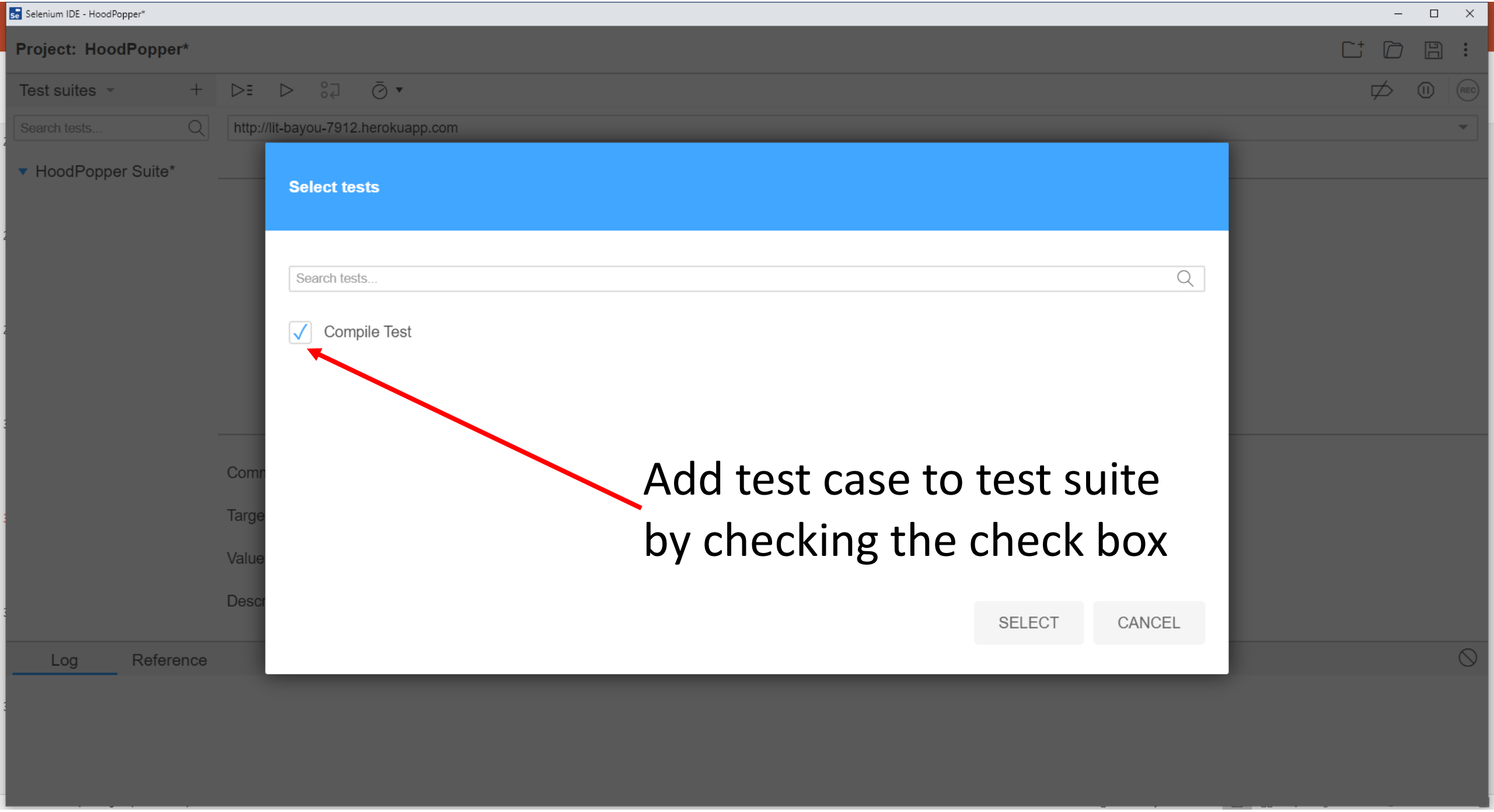
Not all targets are created the same …

- You often have multiple options which include:
  - CSS
  - xpath
  - id
  - Other tag
- Choose the target you think will be the most *stable*
  - Will not change depending on ever-changing website content
  - Will not change in future versions of the website

Adding a test case to a test suite



Select “Add tests” after right-clicking on Test Suite


Selenium IDE - HoodPopper*
Project: HoodPopper*
Test suites
Search tests...
http://lit-bayou-7912.herokuapp.com
HoodPopper Suite*
Select tests
Search tests...
Compile Test
Add test case to test suite
by checking the check box
SELECT
CANCEL
Log
Reference

Let’s Walk Through A Test Case

No Textbook Reading for This Chapter!

- Yay.

- Instead, please skim over: https://www.selenium.dev/selenium-ide/docs/en/api/commands
  - It shows all the assertions you can do and more!

- If you are interested, Appium is Selenium for Mobile Apps: http://appium.io/