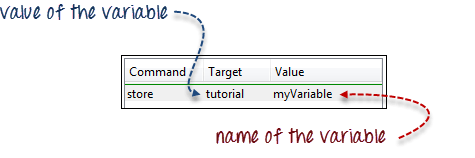
## Store variables, Echo, Alert, PopUp handling

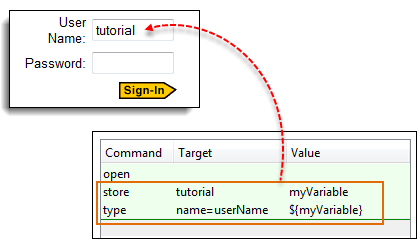
## in Selenium IDE

### Store

To store variables in Selenium IDE, we use the "store" command. The illustration below stores the value "tutorial" to a variable named "myVariable."

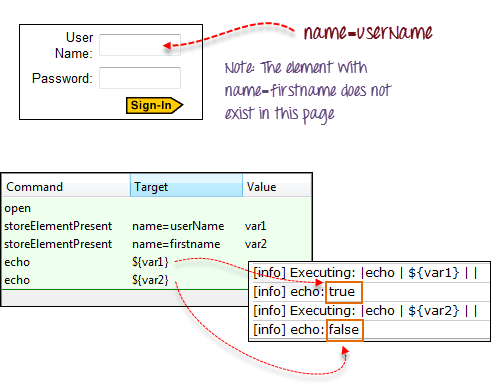
[](https://www.guru99.com/images/image010.png)

To access the variable, simply enclose it in a ${ ... } symbol. For example, to enter the value of "myVariable" onto the "userName" textbox of Mercury Tours, enter ${myVariable} in the Value field.

[](https://www.guru99.com/images/image011.png)

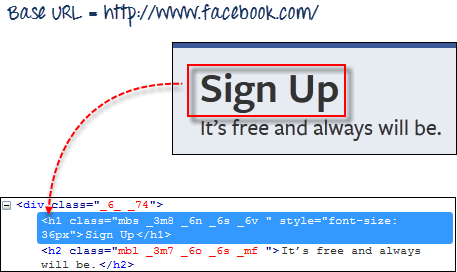
## StoreElementPresent

This command stores either "true" or "false" depending on the presence of the specified element. The script stores the Boolean value "true" to "var1" and "false" to "var2". To verify, we will use the "echo" command to display the values of var1 and var2. The Base URL for the illustration below was set to Mercury Tours homepage.

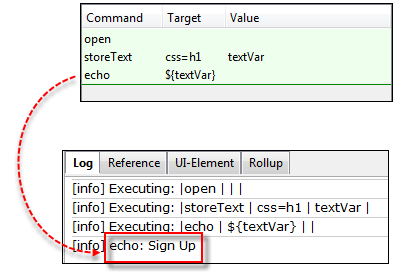


**StoreText**

This command is used to store the inner text of an element onto a variable. The illustration below stores the inner text of the tag in Facebook onto a variable named 'textVar.'

[](https://www.guru99.com/images/image013.png)

Since it is the only element in the page, it is safe to use 'css=h1' as our target. The image below shows that Selenium IDE was able to save the string "Sign Up" in the 'textVar' variable by printing its value correctly.

[](https://www.guru99.com/images/image014.png)

* **Alerts, PopUp and Multiple Windows**

Alerts are probably the simplest form of pop-up windows. The most common Selenium IDE commands used in handling alerts are the following:

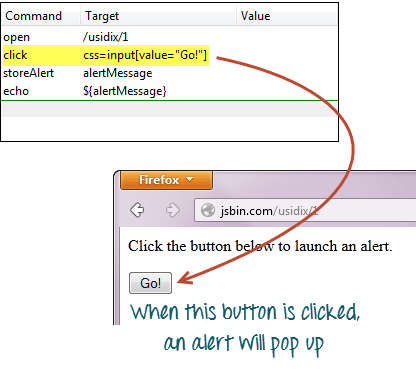
|  |  |
| --- | --- |
| assertAlert assertNotAlert | retrieves the message of the alert and asserts it to a string value that you specified |
| assertAlertPresentassertAlertNotPresent | asserts if an Alert is present or not |
| storeAlert | retrieves the alert message and stores it in a variable that you will specify |
| storeAlertPresent | returns TRUE if an alert is present; FALSE if otherwise |
| verifyAlert verifyNotAlert | retrieves the message of the alert and verifies if it is equal to the string value that you specified |
| verifyAlertPresentverifyAlertNotPresent | verifies if an Alert is present or not |

Remember these two things when working with alerts:

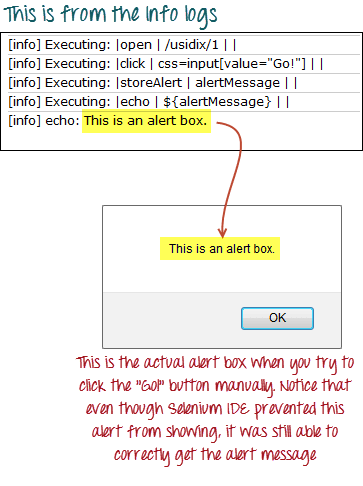
* Selenium IDE will automatically click on the OK button of the alert window, and so you will not be able to see the actual alert.
* Selenium IDE will not be able to handle alerts that are within the page's onload() function. It will only be able to handle alerts that are generated after the page has completely loaded.

**Step1.** In Selenium IDE, set the Base URL to http://jsbin.com. & the full url is: http://jsbin.com/usidix

**Step2.** Create the script as shown below.

[](https://www.guru99.com/images/image015.png)

**Step3.** Execute the script and do not expect that you will be able to see the actual alert.

**[](https://www.guru99.com/images/image016.png)**

**Confirmations:**

Confirmations are popups that give you an OK and a CANCEL button, as opposed to alerts which give you only the OK button. The commands you can use in handling confirmations are similar to those in handling alerts.

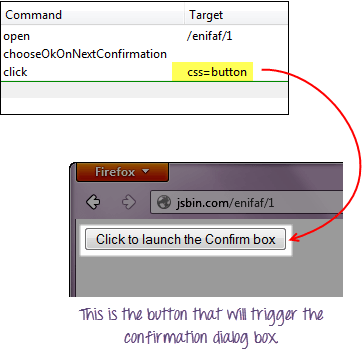
* assertConfirmation/assertNotConfirmation
* assertConfirmationPresent/assertConfirmationNotPresent
* storeConfirmation
* storeConfirmationPresent
* verifyConfirmation/verifyNotConfirmation
* verifyConfirmationPresent/verifyConfirmationNotPresent
* chooseOkOnNextConfirmation/chooseOkOnNextConfirmationAndWait
* chooseCancelOnNextConfirmation

**You should use these commands before a command that triggers the confirmation box so that Selenium IDE will know beforehand which option to choose.**

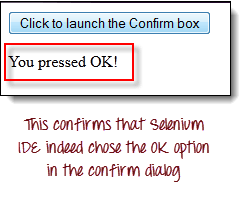
Let us test a webpage that has a button that was coded to show whether the user had pressed the OK or the CANCEL button.

**Step1.** In Selenium IDE, set the Base URL to http://jsbin.com & the full url is: http://jsbin.com/enifaf.

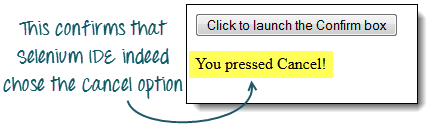
**Step2.** Create the script as shown below. This time, we will press the OK button first.

[](https://www.guru99.com/images/image017.png)

**Step3.**Execute the script and notice that you do not see the actual confirmation, but the webpage was able to indicate which button Selenium IDE had pressed.

[](https://www.guru99.com/images/image018.png)

**Step4.** Replace the "chooseOkOnNextConfirmation" command with "chooseCancelOnNextConfirmation" and execute the script again.

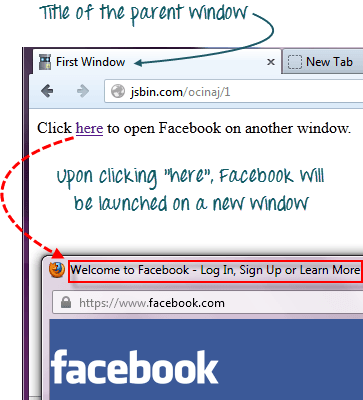
[](https://www.guru99.com/images/image019.png)

* **Multiple Windows**

If you happen to click on a link that launches a separate window, **you must first instruct Selenium IDE to select that window first before you could access the elements within it**. To do this, you will **use the window's title as its locator**. **We use the selectWindow command in switching between windows.**

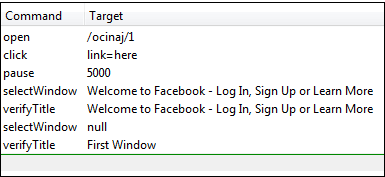
We will use a link http://jsbin.com/ocinaj/1 whose title is "First Window." The "here” hyperlink found on that page will open Facebook in a new window, after which we shall instruct Selenium IDE to do the following:

* Transfer control from the parent window to the newly launched Facebook window using the "selectWindow" command and its title as the locator
* Verify the title of the new window
* Select back the original window using the "selectWindow" command and "null" as its target.
* Verify the title of the currently selected window.

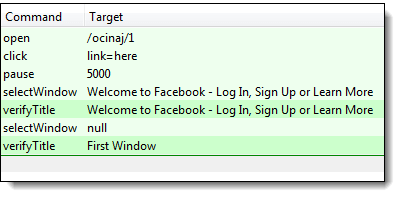
[](https://www.guru99.com/images/image020.png)

**Step1.** Set the Base URL to http://jsbin.com.

**Step2.** Create the script as shown below.

[](https://www.guru99.com/images/image021.png)

**Step3.** Execute the script. Notice that the[Test Case](https://www.guru99.com/test-case.html)passed, meaning that we were able to switch between windows and verify their titles successfully.



Always remember that setting selectWindow's target to "null" will automatically select the parent window (in this case, the window where the element "link=here" is found).