

Common Lisp Selenium Webdriver

A.U. Thor <bug-sample@gnu.org>

Table of Contents

1	Introduction	1
2	Usage.....	2
2.1	Actions.....	2
3	Installation.....	4
4	Utils.....	5
4.1	Interactive session.....	5
4.2	Utils API conventions	5
4.3	Waiting for the reaction	6
4.4	Running tests	6
5	API.....	7
5.1	CL-SELENIUM package.....	7
5.2	CL-SELENIUM-UTILS package	15
6	Index.....	17

1 Introduction

CL Selenium WebDriver is a binding library to the Selenium 2.0

This software is in development. The APIs will be likely to change.

2 Usage

```
;; see examples/*.lisp and t/*.lisp
(in-package :cl-user)

(eval-when (:compile-toplevel :load-toplevel :execute)
  (ql:quickload :cl-selenium))

(defpackage go-test
  (:use :cl :cl-selenium))

(in-package :go-test)

(defparameter *code* "
package main
import \"fmt\"

func main() {
    fmt.Print(\"Hello WebDriver!\")
}")

(with-session ()
  (setf (url) "http://play.golang.org/?simple=1")
  (let ((elem (find-element "#code" :by :css-selector)))
    (element-clear elem)
    (element-send-keys elem *code*))
  (let ((btn (find-element "#run")))
    (element-click btn))

  (loop
    with div = (find-element "#output")
    for ouput = (element-text div)
    while (equal ouput "Waiting for remote server...")
    do (sleep 0.1)
    finally (print ouput)))
```

2.1 Actions

cl-selenium implements a little language for performing actions.

The Actions API provides a low-level interface for providing virtualised device input to the web browser. Conceptually, the Actions commands divide time into a series of ticks. The local end sends a series of actions which correspond to the change in state, if any, of each input device during each tick. For example, pressing a key is represented by an action sequence consisting of a single key input device and two ticks, the first containing a `keyDown` action, and the second a `keyUp` action, whereas a pinch-zoom input is represented by an action sequence consisting of three ticks and two pointer input devices of type touch,

each performing a sequence of actions `pointerDown`, followed by `pointerMove`, and then `pointerUp`.

See <https://www.w3.org/TR/webdriver/#actions> for the whole explanation.

To perform actions in *cl-selenium* use `[PERFORM-ACTIONS]`, page 11. That function implements a little language, with the following syntax:

Syntax:

```
actions ::= ({actions-input-source}*)
actions-input-source ::= ({input-source-type} {action}*)
input-source-type ::= :none | :pointer | :mouse | :pen | :touch | :key
action ::= {pause} | {pointer-move} | {pointer-down} | {pointer-up} | {key-down} | {key-up}
{pause} ::= (:pause duration)
{pointer-move} ::= (:pointer-move x y)
{pointer-down} ::= (:pointer-down button-number)
{pointer-up} ::= (:pointer-up button-number)
{key-down} ::= (:key-down key)
{key-up} ::= (:key-up key)
```

Arguments and values:

- *actions*—a list of actions-input-sources. One list for each type of input source that wants to be used.
- *actions-input-source*—a list. Specifies the list of actions to perform for a particular input source.
- *duration*—an integer. The time to pause in milliseconds.
- *key*—a string. A string with the character (e.g. “a”). Use `[KEY]`, page 8 for entering special characters.
- *button*—an integer greater than or equal to 0.

Examples:

```
(perform-actions '(:pen
  (:pointer-move 22 33)
  (:pause 2000)
  (:pointer-move 23 54))))
```

3 Installation

```
git clone https://github.com/TatriX/cl-selenium-webdriver ~/quicklisp/local-projects/
(ql:quickload :cl-selenium)
```

You need a running instance of selenium-server-standalone.

[Download](<http://www.seleniumhq.org/download/>) it and run:

```
curl -LO https://goo.gl/SP94ZB -o selenium-server-standalone.jar
java -jar selenium-server-standalone.jar
```

4 Utils

There is a `:cl-selenium-utils` package which should reduce boilerplate. For example:

```
(defpackage my-test
  (:use :cl :cl-selenium)
  (:import-from :cl-selenium-utils
    :send-keys
    :click
    :wait-for
    :classlist))

(in-package :my-test)

(with-session ()
  (setf (url) "http://google.com")
  (send-keys "cl-selenium-webdriver")
  (click "[name=btnK]")
  (wait-for "#resultStats"))
```

4.1 Interactive session

You can just start the session and control it from your repl:

```
(in-package :my-test)

(start-interactive-session)

(setf (url) "http://google.com")
(send-keys "cl-selenium-webdriver")
(send-keys (key :enter))
(classlist "#slim_appbar") ; prints ("ab_tnav_wrp")

(stop-interactive-session)
```

4.2 Utils API conventions

If utility function needs an element to work on it defaults to `'(active-element)'`.

```
(click) ; click on the current active element.
```

You can also pass a css selector as a last parameter.

```
(print (id "#submit")) ; print id the of matched element

(assert (= (first (classlist "div")) "first-div-ever"))
```

To change default element you can:

```
(setf cl-selenium-utils:*default-element-func* (lambda () (find-element "input[type=su
```


4.3 Waiting for the reaction

Often you need to wait for some action to be done. For example if you do a `(click)` on the button to load search results, you need to wait them to load.

```
(wait-for ".search-result" :timeout 10) ; wait 10 seconds
```

Timeout defaults to 30 seconds. You can globally change it:

```
(setf cl-selenium-utils:*timeout* 3)
```

4.4 Running tests

REPL

```
(ql:quickload '(:cl-selenium :prove))  
(setf prove:*enable-colors* nil)  
(prove:run :cl-selenium-test)
```

Shell

```
sh  
./test.sh
```

5 API

5.1 CL-SELENIUM package

CL-SELENIUM [PACKAGE]

This package exports functions for working with Selenium WebDriver.

For documentation see:

- <https://github.com/SeleniumHQ/selenium/wiki/JsonWireProtocol>
- <https://www.w3.org/TR/webdriver1>

External definitions

Session

USE-SESSION (*session*) [CL-SELENIUM]

Make *SESSION* the current *session*.

Category: *Session*

MAKE-SESSION (**&key** (*browser-name* :chrome) *browser-version* [CL-SELENIUM]
platform-name platform-version accept-ssl-certs additional-capabilities)

Creates a new WebDriver session with the endpoint node. If the creation fails, a session not created error is returned.

Category: *Session*

See: <https://www.w3.org/TR/webdriver1/#new-session> .

See: <https://www.w3.org/TR/webdriver1/#capabilities> .

DELETE-SESSION (*session*) [CL-SELENIUM]

Delete the WebDriver *SESSION*.

Category: *Session*

START-INTERACTIVE-SESSION (**&rest** *capabilities*) [CL-SELENIUM]

Start an interactive session. Use this to interact with Selenium driver from a REPL.

Category: *Session*

See: [MAKE-SESSION], page 7

STOP-INTERACTIVE-SESSION *nil* [CL-SELENIUM]

Stop an interactive session.

Category: *Session*

WITH-SESSION (**((&rest** *capabilities*) **&body** *body*) [CL-SELENIUM]

Execute *BODY* inside a Selenium session.

Category: Session

See: [MAKE-SESSION], page 7

Uncategorized

MOUSE-CLICK (*button* &**key** (*session* **session**)) [CL-SELENIUM]

Click any mouse *button* (at the coordinates set by the last *moveto* command). Note that calling this command after calling *button* down and before calling *button* up (or any out-of-order interactions sequence) will yield undefined behaviour).

See: <https://github.com/SeleniumHQ/selenium/wiki/JsonWireProtocol#sessionsessionidclick>

SELENIUM-STATUS *nil* [CL-SELENIUM]

Get Selenium Webdriver status information

MOUSE-MOVE-TO (*x y* &**key** *element* (*session* **session**)) [CL-SELENIUM]

Move the mouse by an offset of the specified *element*. If no *element* is specified, the move is relative to the current mouse cursor. If an *element* is provided but no offset, the mouse will be moved to the center of the *element*. If the *element* is not visible, it will be scrolled into view.

See: <https://github.com/SeleniumHQ/selenium/wiki/JsonWireProtocol#sessionsessionidmoveto>

LOGS (*type* &**key** (*session* **session**)) [CL-SELENIUM]

Return the logs of a particular *TYPE*.

See: [LOG-TYPES], page 8.

LOG-TYPES (&**key** (*session* **session**)) [CL-SELENIUM]

Return the types of logs supported by the WebDriver.

- browser: Javascript console logs from the browser.
- client: Logs from the client side implementation of the WebDriver protocol (e.g. the Java bindings).
- driver: Logs from the internals of the driver (e.g. FirefoxDriver internals).
- performance: Logs relating to the performance characteristics of the page under test (e.g. resource load timings).
- server: Logs from within the selenium server.

See: <https://github.com/SeleniumHQ/selenium/wiki/Logging>.

ELEMENT-ID (*sb-pcl::object*) [CL-SELENIUM]

NO-SUCH-ELEMENT-ERROR [CL-SELENIUM]

Error signaled when no such element is found.

Class precedence list: `no-such-element-error`, `find-error`, `error`, `serious-condition`, `condition`, `t`

KEY (*key*) [CL-SELENIUM]

Returns a string with *KEY*'s codepoint.

Elements

FIND-ELEMENT (*value* **&key** (*by* *:css-selector*) (*session* **session**)) [CL-SELENIUM]

The Find Element command is used to find an element in the current browsing context that can be used as the web element context for future element-centric commands.

For example, consider this pseudo code which retrieves an element with the #toremove ID and uses this as the argument for a script it injects to remove it from the HTML document:

```
let body <undefined> [=], page <undefined> session.find.css("#toremove");
session.execute("arguments[0].remove()", [body]);
```

The *BY* parameter represents the element location strategy.

It can be one of:

- *:id* : Finds element *by* id.
- *:class-name* : Finds element *by* class name.
- *:css-selector* : Returns element that matches css selector.
- *:link-text* : Returns element that matches <a> element text.
- *:partial-link-text*: Returns element that matches <a> element text partially.
- *:tag-name*: Returns element that matches tag name.
- *:xpath*: Returns element that matches the XPath expression.

If result is empty, a <undefined> [HANDLE-FIND-ERROR], page <undefined> is signaled.

Category: Elements

See: <https://www.w3.org/TR/webdriver1/#dfn-find-element> .

ELEMENT-TEXT (*element* **&key** (*session* **session**)) [CL-SELENIUM]

The Get *Element* Text command intends to return an *element*'s text "as rendered". An *element*'s rendered text is also used for locating a elements by their link text and partial link text.

Category: Elements

See: <https://www.w3.org/TR/webdriver1/#get-element-text> .

ELEMENT-TAGNAME (*element* **&key** (*session* **session**)) [CL-SELENIUM]

Return the *ELEMENT*'s tag name.

Category: Elements

ELEMENT-ATTRIBUTE (*element name* **&key** (*session* **session**)) [CL-SELENIUM]

Return the *ELEMENT*'s attribute named *NAME*.

Category: Elements

ELEMENT-RECT (*element* &**key** (*session* **session**)) [CL-SELENIUM]

The Get *Element* Rect command returns the dimensions and coordinates of the given web *element*. The returned value is a dictionary with the following members:

x

X axis position of the top-left corner of the web *element* relative to the current browsing context's document *element* in CSS pixels.

y

Y axis position of the top-left corner of the web *element* relative to the current browsing context's document *element* in CSS pixels.

height

Height of the web *element*'s bounding rectangle in CSS pixels.

width

Width of the web *element*'s bounding rectangle in CSS pixels.

Category: Elements

ELEMENT-DISPLAYED (*element* &**key** (*session* **session**)) [CL-SELENIUM]

Returns if *ELEMENT* is visible.

Category: Elements

See: <https://www.w3.org/TR/webdriver1/#element-displayedness> .

FIND-ELEMENTS (*value* &**key** (*by* :*css-selector*) (*session* **session**)) [CL-SELENIUM]

Find elements that match *VALUE* using location strategy in *BY*.

Category: Elements

See [FIND-ELEMENT], page 9.

See <https://www.w3.org/TR/webdriver1/#find-elements> .

ACTIVE-ELEMENT (&**key** (*session* **session**)) [CL-SELENIUM]

Return the active element of the current browsing context's document.

The active element is the Element within the DOM that currently has focus.

If there's no active element, an error is signaled.

Category: Elements

See: <https://www.w3.org/TR/webdriver2/#get-active-element> .

See: <https://developer.mozilla.org/en-US/docs/Web/API/Document/activeElement> .

ELEMENT-ENABLED (*element* &**key** (*session* **session**)) [CL-SELENIUM]

Returns if *ELEMENT* is enabled.

Category: Elements

See: <https://www.w3.org/TR/webdriver1/#is-element-enabled> .

Element interaction

ELEMENT-SEND-KEYS (*element* **&key** (*session* **session**)) [CL-SELENIUM]

The *Element* Send Keys command scrolls into view the form control *element* and then sends the provided *keys* to the *element*. In case the *element* is not keyboard-interactable, an *element* not interactable error is returned.

KEYS should be a string.

Category: *Element* interaction

See: <https://www.w3.org/TR/webdriver1/#element-send-keys>.

ELEMENT-CLEAR (*element* **&key** (*session* **session**)) [CL-SELENIUM]

Clear the contents of *ELEMENT* (for example, a form field *element*).

Category: *Element* interaction

See: <https://www.w3.org/TR/webdriver1/#dfn-element-clear>.

ELEMENT-CLICK (*element* **&key** (*session* **session**)) [CL-SELENIUM]

The *Element* Click command scrolls into view the *element* if it is not already pointer-interactable, and clicks its in-view center point.

If the *element*'s center point is obscured by another *element*, an *element* click intercepted error is returned. If the *element* is outside the viewport, an *element* not interactable error is returned.

Category: *Element* interaction

See: <https://www.w3.org/TR/webdriver1/#element-click>.

Windows

CLOSE-CURRENT-WINDOW (**&key** (*session* **session**)) [CL-SELENIUM]

Close the current window.

Category: Windows

Actions

PERFORM-ACTIONS (*actions* **&optional** (*session* **session**)) [CL-SELENIUM]

The *Actions* API provides a low-level interface for providing virtualised device input to the web browser.

Conceptually, the *Actions* commands divide time into a series of ticks. The local end sends a series of *actions* which correspond to the change in state, if any, of each input device during each tick. For example, pressing a key is represented by an action sequence consisting of a single key input device and two ticks, the first containing a keyDown action, and the second a keyUp action, whereas a pinch-zoom input is represented by an action sequence consisting of three ticks and two pointer input devices of type touch, each performing a sequence of *actions* pointerDown, followed by pointerMove, and then pointerUp.

Category: *Actions*

See: <https://www.w3.org/TR/webdriver/#actions>

Navigation

URL (&key (session *session*)) [CL-SELENIUM]
Get the current url in *session*.

Category: Navigation

See: <https://www.w3.org/TR/webdriver1/#dfn-get-current-url> .

SWITCH-TO-FRAME (id &key (session *session*)) [CL-SELENIUM]
Change focus to another frame on the page. If the frame *id* is null, the server should switch to the page's default content.

In the context of a web browser, a frame is a part of a web page or browser window which displays content independent of its container, with the ability to load content independently.

Category: Navigation

See: <https://github.com/SeleniumHQ/selenium/wiki/JsonWireProtocol#sessionsessionidframe> .

See: [https://en.wikipedia.org/wiki/Frame_\(World_Wide_Web\)](https://en.wikipedia.org/wiki/Frame_(World_Wide_Web)) .

PAGE-TITLE (&key (session *session*)) [CL-SELENIUM]
This command returns the document title of the current top-level browsing context, equivalent to calling `document.title`.

Category: Navigation

See: <https://www.w3.org/TR/webdriver2/#get-title> .

REFRESH (&key (session *session*)) [CL-SELENIUM]
Refresh the current page.

Category: Navigation

BACK (&key (session *session*)) [CL-SELENIUM]
This command causes the browser to traverse one step backward in the joint *session* history of the current top-level browsing context. This is equivalent to pressing the back button in the browser chrome or invoking `window.history.back`.

Category: Navigation

See: <https://www.w3.org/TR/webdriver1/#dfn-back> .

Screen capture

ELEMENT-SCREENSHOT (element &key (session *session*)) [CL-SELENIUM]
The *Take Element Screenshot* command takes a screenshot of the visible region encompassed by the bounding rectangle of an *element*. If given a parameter argument

scroll that evaluates to false, the *element* will not be scrolled into view.

Category: Screen capture

See: <https://www.w3.org/TR/webdriver1/#take-element-screenshot> .

SCREENSHOT (**&key** (*session* **session**)) [CL-SELENIUM]

Screenshots are a mechanism for providing additional visual diagnostic information. They work by dumping a snapshot of the initial viewport's framebuffer as a lossless PNG image. It is returned to the local end as a Base64 encoded string.

Category: Screen capture

See: <https://www.w3.org/TR/webdriver2/#screen-capture> .

Cookies

COOKIE [CL-SELENIUM]

A cookie is described in [RFC6265] by a name-value pair holding the cookie's data, followed by zero or more attribute-value pairs describing its characteristics.

Category: Cookies

Class precedence list: `cookie`, `standard-object`, `t`

Slots:

- **name** — initarg: `:name`
The name of the cookie
- **value** — initarg: `:value`
The cookie value
- **path** — initarg: `:path`
The cookie path. Defaults to `'/'` if omitted when adding a cookie.
- **domain** — initarg: `:domain`
The domain the cookie is visible to. Defaults to the current browsing context's active document's URL domain if omitted when adding a cookie.
- **secure** — initarg: `:secure`
Whether the cookie is a secure cookie. Defaults to false if omitted when adding a cookie.
- **http-only** — initarg: `:http-only`
Whether the cookie is an HTTP only cookie. Defaults to false if omitted when adding a cookie.
- **expiry** — initarg: `:expiry`
When the cookie expires, specified in seconds since Unix Epoch. Must not be set if omitted when adding a cookie.

COOKIE (**&key** (*session* **session**)) [CL-SELENIUM]

Retrieve all cookies visible to the current page.

Category: Cookies

See: <https://www.w3.org/TR/webdriver1/#get-all-cookies> .

See: <https://github.com/SeleniumHQ/selenium/wiki/JsonWireProtocol#sessionsessionidcookie> .

DELETE-ALL-COOKIES (**&key** (session *session*)) [CL-SELENIUM]
Deletes all cookies

Category: Cookies

See: <https://www.w3.org/TR/webdriver1/#delete-all-cookies>

MAKE-COOKIE (name value **&key** path domain secure expiry) [CL-SELENIUM]
Create a cookie object.

Category: Cookies

DELETE-COOKIE (cookie-name **&key** (session *session*)) [CL-SELENIUM]
Delete the cookie with name *COOKIE-NAME*.

Category: Cookies

See: <https://www.w3.org/TR/webdriver1/#delete-cookie>

FIND-COOKIE (cookie-name **&key** (session *session*)) [CL-SELENIUM]
Retrieve the cookie with name *COOKIE-NAME*.

Category: Cookies

See: <https://www.w3.org/TR/webdriver1/#get-named-cookie>

User prompts

DISMISS-ALERT (**&key** (session *session*)) [CL-SELENIUM]
The Dismiss Alert command dismisses a simple dialog if present. A request to dismiss an alert user prompt, which may not necessarily have a dismiss button, has the same effect as accepting it.

Category: User prompts

See: <https://www.w3.org/TR/webdriver1/#dismiss-alert>

ACCEPT-ALERT (**&key** (session *session*)) [CL-SELENIUM]
Accept Alert.

Category: User prompts

See: <https://www.w3.org/TR/webdriver1/#dfn-accept-alert>

ALERT-TEXT (**&key** (session *session*)) [CL-SELENIUM]
Get Alert Text.

Category: User prompts

See: <https://www.w3.org/TR/webdriver1/#get-alert-text>

Document handling

EXECUTE-SCRIPT (*script* *args* **&key** (*session* **session**)) [CL-SELENIUM]

Inject a snippet of JavaScript into the page for execution in the context of the currently selected frame. The executed *script* is assumed to be synchronous and the result of evaluating the *script* is returned to the client.

The *script* argument defines the *script* to execute in the form of a function body. The value returned by that function will be returned to the client. The function will be invoked with the provided *args* array and the values may be accessed via the arguments object in the order specified.

Arguments may be any JSON-primitive, array, or JSON object. JSON objects that define a WebElement reference will be converted to the corresponding DOM element. Likewise, any WebElements in the *script* result will be returned to the client as WebElement JSON objects.

Category: Document handling

See: <https://github.com/SeleniumHQ/selenium/wiki/JsonWireProtocol#sessionsessionidexecute> .

5.2 CL-SELENIUM-UTILS package

CL-SELENIUM-UTILS [PACKAGE]

Package with the purpose of reducing boilerplate.

External definitions

Variables

TIMEOUT [CL-SELENIUM-UTILS]

Default timeout value to use in selenium-utils functions.

DEFAULT-ELEMENT-FUNC [CL-SELENIUM-UTILS]

Function used to get the 'default element' by selenium-utils functions.

It is [ACTIVE-ELEMENT], page 10 function by default.

Functions

ID (**&optional** *selector*) [CL-SELENIUM-UTILS]

Get active element id.

GET-COOKIE (*cookie name*) [CL-SELENIUM-UTILS]

Get value of *COOKIE* at *NAME*.

FIND-ELEM (*selector* **&key** (*by* *:css-selector*)) [CL-SELENIUM-UTILS]

Find element by *SELECTOR*. Returns NIL if the element is not found.

WAIT-FOR (<i>selector</i> &key (<i>timeout</i> <i>*timeout*</i>))	[CL-SELENIUM-UTILS]
Wait for an element that matches <i>SELECTOR</i> to appear on the screen. <i>TIMEOUT</i> indicates how much time to wait (default is <i>*TIMEOUT*</i>).	
CLASSNAME (&optional <i>selector</i>)	[CL-SELENIUM-UTILS]
Get active element classname.	
TEXT (&optional <i>selector</i>)	[CL-SELENIUM-UTILS]
Get active element's text.	
SEND-KEY (<i>key</i> &optional <i>selector</i>)	[CL-SELENIUM-UTILS]
Send a key to active element.	
CLASSLIST (&optional <i>selector</i>)	[CL-SELENIUM-UTILS]
Get active element class list.	
ATTR (<i>name</i> &optional <i>selector</i>)	[CL-SELENIUM-UTILS]
Get active element attribute.	
SEND-KEYS (<i>keys</i> &optional <i>selector</i>)	[CL-SELENIUM-UTILS]
Send <i>keys</i> to active element.	
ELEM (&optional <i>selector</i>)	[CL-SELENIUM-UTILS]
If <i>SELECTOR</i> is given, wait for an element that matches the <i>selector</i> to appear. Otherwise, call [*DEFAULT-ELEMENT-FUNC*], page 15 (the active element is returned by default).	
CLICK (&optional <i>selector</i>)	[CL-SELENIUM-UTILS]
Click on active element.	

6 Index

(Index is nonexistent)

*

DEFAULT-ELEMENT-FUNC	15
TIMEOUT	15

A

ACCEPT-ALERT	14
ACTIVE-ELEMENT	10
ALERT-TEXT	14
ATTR	16

B

BACK	12
------	----

C

CL-SELENIUM-UTILS:ATTR	16
CL-SELENIUM-UTILS:CLASSLIST	16
CL-SELENIUM-UTILS:CLASSNAME	16
CL-SELENIUM-UTILS:CLICK	16
CL-SELENIUM-UTILS:ELEM	16
CL-SELENIUM-UTILS:FIND-ELEM	15
CL-SELENIUM-UTILS:GET-COOKIE	15
CL-SELENIUM-UTILS:ID	15
CL-SELENIUM-UTILS:SEND-KEY	16
CL-SELENIUM-UTILS:SEND-KEYS	16
CL-SELENIUM-UTILS:TEXT	16
CL-SELENIUM-UTILS:WAIT-FOR	16
CL-SELENIUM:ACCEPT-ALERT	14
CL-SELENIUM:ACTIVE-ELEMENT	10
CL-SELENIUM:ALERT-TEXT	14
CL-SELENIUM:BACK	12
CL-SELENIUM:CLOSE-CURRENT-WINDOW	11
CL-SELENIUM:COOKIE	13
CL-SELENIUM:DELETE-ALL-COOKIES	14
CL-SELENIUM:DELETE-COOKIE	14
CL-SELENIUM:DELETE-SESSION	7
CL-SELENIUM:DISMISS-ALERT	14
CL-SELENIUM:ELEMENT-ATTRIBUTE	9
CL-SELENIUM:ELEMENT-CLEAR	11
CL-SELENIUM:ELEMENT-CLICK	11
CL-SELENIUM:ELEMENT-DISPLAYED	10
CL-SELENIUM:ELEMENT-ENABLED	10
CL-SELENIUM:ELEMENT-ID	8
CL-SELENIUM:ELEMENT-RECT	10
CL-SELENIUM:ELEMENT-SCREENSHOT	12
CL-SELENIUM:ELEMENT-SEND-KEYS	11
CL-SELENIUM:ELEMENT-TAGNAME	9
CL-SELENIUM:ELEMENT-TEXT	9

C

CL-SELENIUM-UTILS:*DEFAULT-ELEMENT-FUNC*	15
CL-SELENIUM-UTILS:*TIMEOUT*	15
CL-SELENIUM:EXECUTE-SCRIPT	15
CL-SELENIUM:FIND-COOKIE	14
CL-SELENIUM:FIND-ELEMENT	9
CL-SELENIUM:FIND-ELEMENTS	10
CL-SELENIUM:KEY	8
CL-SELENIUM:LOG-TYPES	8
CL-SELENIUM:LOGS	8
CL-SELENIUM:MAKE-COOKIE	14
CL-SELENIUM:MAKE-SESSION	7
CL-SELENIUM:MOUSE-CLICK	8
CL-SELENIUM:MOUSE-MOVE-TO	8
CL-SELENIUM:PAGE-TITLE	12
CL-SELENIUM:PERFORM-ACTIONS	11
CL-SELENIUM:REFRESH	12
CL-SELENIUM:SCREENSHOT	13
CL-SELENIUM:SELENIUM-STATUS	8
CL-SELENIUM:START-INTERACTIVE-SESSION	7
CL-SELENIUM:STOP-INTERACTIVE-SESSION	7
CL-SELENIUM:SWITCH-TO-FRAME	12
CL-SELENIUM:URL	12
CL-SELENIUM:USE-SESSION	7
CL-SELENIUM:WITH-SESSION	7
CLASSLIST	16
CLASSNAME	16
CLICK	16
CLOSE-CURRENT-WINDOW	11
COOKIE	13

D

DELETE-ALL-COOKIES	14
DELETE-COOKIE	14
DELETE-SESSION	7
DISMISS-ALERT	14

E

ELEM	16
ELEMENT-ATTRIBUTE	9
ELEMENT-CLEAR	11
ELEMENT-CLICK	11
ELEMENT-DISPLAYED	10
ELEMENT-ENABLED	10
ELEMENT-ID	8
ELEMENT-RECT	10
ELEMENT-SCREENSHOT	12
ELEMENT-SEND-KEYS	11
ELEMENT-TAGNAME	9
ELEMENT-TEXT	9
EXECUTE-SCRIPT	15

F

FIND-COOKIE	14
FIND-ELEM	15
FIND-ELEMENT	9
FIND-ELEMENTS	10

G

GET-COOKIE	15
------------	----

I

ID	15
----	----

K

KEY	8
-----	---

L

LOG-TYPES	8
LOGS	8

M

MAKE-COOKIE	14
MAKE-SESSION	7
MOUSE-CLICK	8
MOUSE-MOVE-TO	8

P

PAGE-TITLE	12
PERFORM-ACTIONS	11

R

REFRESH	12
---------	----

S

SCREENSHOT	13
SELENIUM-STATUS	8
SEND-KEY	16
SEND-KEYS	16
START-INTERACTIVE-SESSION	7
STOP-INTERACTIVE-SESSION	7
SWITCH-TO-FRAME	12

T

TEXT	16
------	----

U

URL	12
USE-SESSION	7

W

WAIT-FOR	16
WITH-SESSION	7