Madhan K Follow

Apr 17, 2020 · 6 min read · Listen

Save

Selenium cheat sheet — a comprehensive list of selenium commands

A curated list of selenium commands in Java

1. Browser property setup

- **Chrome:**

```
System.setProperty(“webdriver.chrome.driver”,
“/path/to/chromedriver”);
```

- **Firefox:**

```
System.setProperty(“webdriver.gecko.driver”,-
“/path/to/geckodriver”);
```

- **Edge:**

```
System.setProperty(“webdriver.edge.driver”,
“P/path/to/MicrosoftWebDriver”);
```

2. Browser Initialization

- **Firefox**

```
WebDriver driver = new FirefoxDriver();
```

- **Chrome**

```
WebDriver driver = new ChromeDriver();
```

- **Internet Explorer**

```
WebDriver driver = new InternetExplorerDriver();
```

- **Safari Driver**

```
WebDriver driver = new SafariDriver();
```

3. Desired capabilities

> *(Doc link)*

- **Chrome:**

```
DesiredCapabilities caps = new DesiredCapabilities();
caps.setCapability(“browserName”, “chrome”);
caps.setCapability(“browserVersion”, “80.0””);
caps.setCapability(“platformName”, “win10”);

WebDriver driver = new ChromeDriver(caps); // Pass the capabilities
as an argument to the driver object
```

- **Firefox:**

```
DesiredCapabilities caps = new DesiredCapabilities();
caps.setCapability(“browserName”, “firefox”);
caps.setCapability(“browserVersion”, “81.0””);
caps.setCapability(“platformName”, “win10”);

WebDriver driver = new FirefoxDriver(caps); // Pass the capabilities
as an argument to the driver object
```

4. Browser options

- **Chrome: (Doc link)**

```
ChromeOptions chromeOptions = new ChromeOptions();
chromeOptions.setBinary("C:Program Files
(x86)GoogleChromeApplicationchrome.exe"); // if chrome is not in
default location
chromeOptions.addArguments("--headless"); // Passing single option
chromeOptions.addArguments("--start-maximized", "--incognito","--
disable-notifications" ); // Passing multiple options

WebDriver driver = new ChromeDriver(chromeOptions); // Pass the
capabilities as an argument to the driver object
```

- **Firefox: (Doc link)**

```
FirefoxOptions firefoxOptions = new FirefoxOptions();
firefoxOptions.setBinary(new FirefoxBinary(new File("C:Program
FilesMozilla Firefoxirefox.exe")));
firefoxOptions.setHeadless(true);

WebDriver driver = new FirefoxDriver(caps); // Pass the capabilities
as an argument to the driver object
```

> ***Options VS Desired capabilities:***
> *There are two ways to specify capabilities.*
>
> *1. **ChromeOptions/FirefoxOptions** class — Recommended*

> *2. Or you can specify the capabilities directly as part of the **DesiredCapabilities** — its usage in Java is deprecated*

5. Navigation

- **Navigate to URL — (doc link1 link2)**

```
driver.get("http://google.com")
driver.navigate().to("http://google.com")
```

> ***Myth —** get() method waits till the page is loaded while navigate() does not.*

Referring to the selenium official doc, get() method is a synonym for to() method. Both do the same thing.

> ***Myth —** get() does not store history while navigate() does.*

All the URLs loaded in the browser will be stored in history and the navigate method allows us to access it. Try executing the below code

```
driver.get("http://madhank93.github.io/");
driver.get("https://www.google.com/");
driver.navigate().back();
```

- **Refresh page**

```
driver.navigate().refresh()
```

- **Navigate forwards in the browser history**

```
driver.navigate().forward()
```

- **Navigate backward in the browser history**

```
driver.navigate().back()
```

6. Find element VS Find elements

> *(doc link)*

- **driver.findElement()**

```
When no match has found(0) throws NoSuchElementException
when 1 match found returns a WebElement instance
when 2+ matches found returns only the first matching web element
```

- **driver.findElements()**

```
when no macth has found (0) returns an empty list
when 1 match found returns a list with one WebElement
when 2+ matches found returns a list with all matching WebElements
```

7. Locator Strategy

> *(doc link)*

- *By id*

```
<input id=”login” type=”text” />
```

`element = driver.findElement(By.id(“login”))`

- *By Class Name*

```
<input class=”Content” type=”text” />
```

`element = driver.findElement(By.className(“Content”));`

- *By Name*

```
<input name=”pswd” type=”text” />
```

```
element = driver.findElement(By.name(“pswd”));
```

- *By Tag Name*

```
<div id=”forgot-password” >…</div>
```

```
element = driver.findElement(By.tagName(“div”));
```

- *By Link Text*

```
<a href=”#”>News</a>
```

```
element = driver.findElement(By.linkText(“News”));
```

- *By XPath*

```
<form id=”login” action="/action_page.php">
      <input type="text" placeholder="Username" name="username">
      <input type="text" placeholder="Password" name="psw">
      <button type="submit">Login</button>
</form>
```

```
element = driver.findElement(By.xpath(“//input[@placeholder=’Username’]”));
```

```
List of Keywords — and, or, contains(), starts-with(), text(), last()
```

- *By CSS Selector*

```
<form id=”login” action=”submit” method=”get”>
Username: <input type=”text” />
```

```
Password: <input type="password" />
</form>
```

```
element = driver.findElement(By.cssSelector("input.username"));
```

8. Click on an element

- **click()** — method is used to click on an element

```
driver.findElement(By.className("Content")).click();
```

9. Write text inside an element — input and textarea

- **sendKeys()** — method is used to send data

```
driver.findElement(By.className("email")).sendKeys("abc@xyz.com");
```

10. Clear text from the text box

- **clear()** — method is used to clear text from the text area

```
driver.findElement(By.xpath("//input[@placeholder='Username']")).clear();
```

11. Select a drop-down

> *(doc link)*

```
// single select option
<select id="country">
<option value="US">United States</option>
<option value="CA">Canada</option>
<option value="MX">Mexico</option>
</select>

// multiple select option
<select multiple="" id="fruits">
     <option value="banana">Banana</option>
     <option value="apple">Apple</option>
     <option value="orange">Orange</option>
     <option value="grape">Grape</option>
</select>
```

- **selectByVisibleText() / selectByValue() / selectByIndex()**

- **deselectByVisibleText() / deselectByValue() / deselectByIndex()**

```
// import statements for select class
import org.openqa.selenium.support.ui.Select;

// Single selection
Select country = new Select(driver.findElement(By.id("country")));
country.selectByVisibleText("Canada"); // using
selectByVisibleText() method
country.selectByValue("MX"); //using selectByValue() method

//Selecting Items in a Multiple SELECT elements
Select fruits = new Select(driver.findElement(By.id("fruits")));
fruits.selectByVisibleText("Banana");
fruits.selectByIndex(1); // using selectByIndex() method
```

12. Get methods in Selenium

- **getTitle()** — used to retrieve the current title of the webpage
- **getCurrentUrl()** — used to retrieve the current URL of the webpage
- **getPageSource()** — used to retrieve the current page source of the webpage
- **getText()** — used to retrieve the text of the specified web element
- **getAttribute()** — used to retrieve the value specified in the attribute

13. Handle alerts: (Web-based alert pop-ups)

- **driver.switchTO().alert.getText()** — to retrieve the alert message
- **driver.switchTO().alert.accept()** — to accept the alert box
- **driver.switchTO().alert.dismiss()** — to cancel the alert box
- **driver.switchTO().alert.sendKeys("Text")** — to send data to the alert box

14. Switch frames

- **driver.switchTo.frame(int frameNumber)** — mentioning the frame index number, the Driver will switch to that specific frame
- **driver.switchTo.frame(string frameNameOrID)** — mentioning the frame element or ID, the Driver will switch to that specific frame

- **driver.switchTo.frame(WebElement frameElement) —** mentioning the frame web element, the Driver will switch to that specific frame
- **driver.switchTo().defaultContent() —** Switching back to the main window

15. Handle multiple windows and tabs

- **getWindowHandle() —** used to retrieve the handle of the current page (a unique identifier)
- *getWindowHandles()* — used to retrieve a set of handles of the all the pages available
- **driver.switchTo().window("windowName/handle") —** switch to a window
- **driver.close() —** closes the current browser window

16. Waits in selenium

There are 3 types of waits in selenium,

- **Implicit Wait —** used to wait for a certain amount of time before throwing an exception

```
driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
```

- **Explicit Wait —** used to wait until a certain condition occurs before executing the code.

```
WebDriverWait wait = new WebDriverWait(driver,30);
wait.until(ExpectedConditions.presenceOfElementLocated(By.name("logi
n")));
```

List of explicit wait:

```
alertIsPresent()
elementSelectionStateToBe()
elementToBeClickable()
elementToBeSelected()
frameToBeAvaliableAndSwitchToIt()
```

```
invisibilityOfTheElementLocated()
invisibilityOfElementWithText()
presenceOfAllElementsLocatedBy()
presenceOfElementLocated()
textToBePresentInElement()
textToBePresentInElementLocated()
textToBePresentInElementValue()
titleIs()
titleContains()
visibilityOf()
visibilityOfAllElements()
visibilityOfAllElementsLocatedBy()
visibilityOfElementLocated()
```

- **Fluent Wait —** defines the maximum amount of time to wait for a certain condition to appear

```
Wait wait = new FluentWait(WebDriver reference)
.withTimeout(Duration.ofSeconds(SECONDS))
.pollingEvery(Duration.ofSeconds(SECONDS))
.ignoring(Exception.class);

WebElement foo=wait.until(new Function<WebDriver, WebElement>() {
public WebElement apply(WebDriver driver) {
return driver.findElement(By.id("foo"));
}
});
```

17. Element validation

- **isEnabled()** — determines if an element is enabled or not, returns a boolean.
- **isSelected()** — determines if an element is selected or not, returns a boolean.
- **isDisplayed()** — determines if an element is displayed or not, returns a boolean.

18. Handling proxy

- **Chrome:**

```
ChromeOptions options = new ChromeOptions();

// Create object Proxy class - Approach 1
Proxy proxy = new Proxy();
proxy.setHttpProxy("username:password.myhttpproxy:3337");
```

```
// register the proxy with options class - Approach 1
options.setCapability("proxy", proxy);

// Add a ChromeDriver-specific capability.
ChromeDriver driver = new ChromeDriver(options);
```

- **Firefox:**

```
FirefoxOptions options = new FirefoxOptions();

// Create object Proxy class - Approach 2
Proxy proxy = new Proxy();
proxy.setHttpProxy("myhttpproxy:3337");
proxy.setSocksUsername("username");
proxy.setSocksPassword("password")

// register the proxy with options class - Approach 2
options.setProxy(proxy);

// create object to firefx driver
WebDriver driver = new FirefoxDriver(options);
```

19. Window management

- **Get window size:**

```
//Access each dimension individually
int width = driver.manage().window().getSize().getWidth();
int height = driver.manage().window().getSize().getHeight();

//Or store the dimensions and query them later
Dimension size = driver.manage().window().getSize();
int width1 = size.getWidth();
int height1 = size.getHeight();
```

- **Set window size:**

```
driver.manage().window().setSize(new Dimension(1024, 768));
```

- **Get window position:**

```
// Access each dimension individually
int x = driver.manage().window().getPosition().getX();
int y = driver.manage().window().getPosition().getY();

// Or store the dimensions and query them later
Point position = driver.manage().window().getPosition();
int x1 = position.getX();
int y1 = position.getY();
```

- **Set window position:**

```
// Move the window to the top left of the primary monitor
driver.manage().window().setPosition(new Point(0, 0));
```

- **Maximize window:**

```
driver.manage().window().maximize();
```

- **Fullscreen window:**

```
driver.manage().window().fullscreen();
```

20. Page loading strategy

The `document.readyState` property of a document describes the loading state of the current document. By default, WebDriver will hold off on responding to a `driver.get()` (or) `driver.navigate().to()` call until the document ready state is `complete`

By default, when Selenium WebDriver loads a page, it follows the normal pageLoadStrategy.

- **normal:**

```
ChromeOptions chromeOptions = new ChromeOptions();
chromeOptions.setPageLoadStrategy(PageLoadStrategy.NORMAL);
```

```
WebDriver driver = new ChromeDriver(chromeOptions);
```

- **eager:** When setting to eager, Selenium WebDriver waits until DOMContentLoaded event fire is returned.

```
ChromeOptions chromeOptions = new ChromeOptions();
chromeOptions.setPageLoadStrategy(PageLoadStrategy.EAGER);
WebDriver driver = new ChromeDriver(chromeOptions);
```

- **none:** When set to none Selenium WebDriver only waits until the initial page is downloaded.

```
ChromeOptions chromeOptions = new ChromeOptions();
chromeOptions.setPageLoadStrategy(PageLoadStrategy.NONE);
WebDriver driver = new ChromeDriver(chromeOptions);
```

21. Keyboard and Mouse events

Action class is used to handle keyboard and mouse events

keyboard events:

- keyDown()
- keyUp()
- sendKeys()

Mouse events:

- doubleClick()
- dragAndDrop(source,target)
- dragAndDropBy(source,xOffset,yOffset)
- moveByOffset(xOffset,yOffset)

- moveByElement()
- release()

```
Actions builder = new Actions(driver);

Action actions = builder
 .moveToElement("login-textbox")
 .click()
 .keyDown("login-textbox", Keys.SHIFT)
 .sendKeys("login-textbox", "hello")
 .keyUp("login-textbox", Keys.SHIFT)
 .doubleClick("login-textbox")
 .contextClick()
 .build();

actions.perform() ;
```

22. Cookies

- **addCookie(arg)**

```
driver.manage().addCookie(new Cookie("foo", "bar"));
```

- **getCookies()**

```
driver.manage().getCookies(); // to get all cookies
```

- **getCookieNamed()**

439 | 4

```
driver.manage().getCookieNamed("foo");
```

- **deleteCookieNamed()**

```
driver.manage().deleteCookieNamed("foo");
```

- **deleteCookie()**

```
Cookie cookie1 = new Cookie("test2", "cookie2");
driver.manage().addCookie(cookie1);
driver.manage().deleteCookie(cookie1); // deleting cookie object
```

- **deleteAllCookies()**

```
driver.manage().deleteAllCookies(); // deletes all cookies
```

23. Take screenshot:

> *(doc link)*

- **getScreenshotAs —** used to Capture the screenshot and store it in the specified location. This method throws WebDriverException. copy() method from the File Handler class is used to store the screenshot in a destination folder

```
TakesScreenshot screenShot =(TakesScreenshot)driver;
FileHandler.copy(screenShot.getScreenshotAs(OutputType.FILE), new
File("path/to/destination/folder/screenshot.png"));
```

24. Execute Javascript:

> *(doc link)*

- **executeAsyncScript() —** executes an asynchronous piece of JavaScript
- **executeScript() —** executes JavaScript

```
if (driver instanceof JavascriptExecutor) {
    ((JavascriptExecutor)driver).executeScript("alert('hello
world');");
}
```

Last updated on — Apr 18, 2020

madhank93/selenium-cheatsheet-java

A comprehensive list of selenium commands in Java. Contribute to madhank93/selenium-cheatsheet-java development by...

github.com

References:

[1] https://www.selenium.dev/selenium/docs/api/java/overview-summary.html

[2] https://www.selenium.dev/documentation/en/

Selenium Cheatsheet Automation Software Testing Testing