Automation Testing

Selenium Architecture

A Day in the Life of a Automation Test Engineer

Joel has decided to use an open-source tool that automates browsers for his projects.

As an Automation Test Engineer, when he automates a web application, he will need a tool that provides options to automate browsers and provides easy navigation and interactions with many internal web element components like textbox, radio buttons, dropdowns and many more.

He will acquire a few ideas in this session that can assist him in solving the problem so that he can accomplish the objectives.

Learning Objectives

By the end of this lesson, you will be able to:

- Explain WebDriver architecture
- Analyze WebDriver commands
- Analyze Navigation commands
- Explain WebElement commands

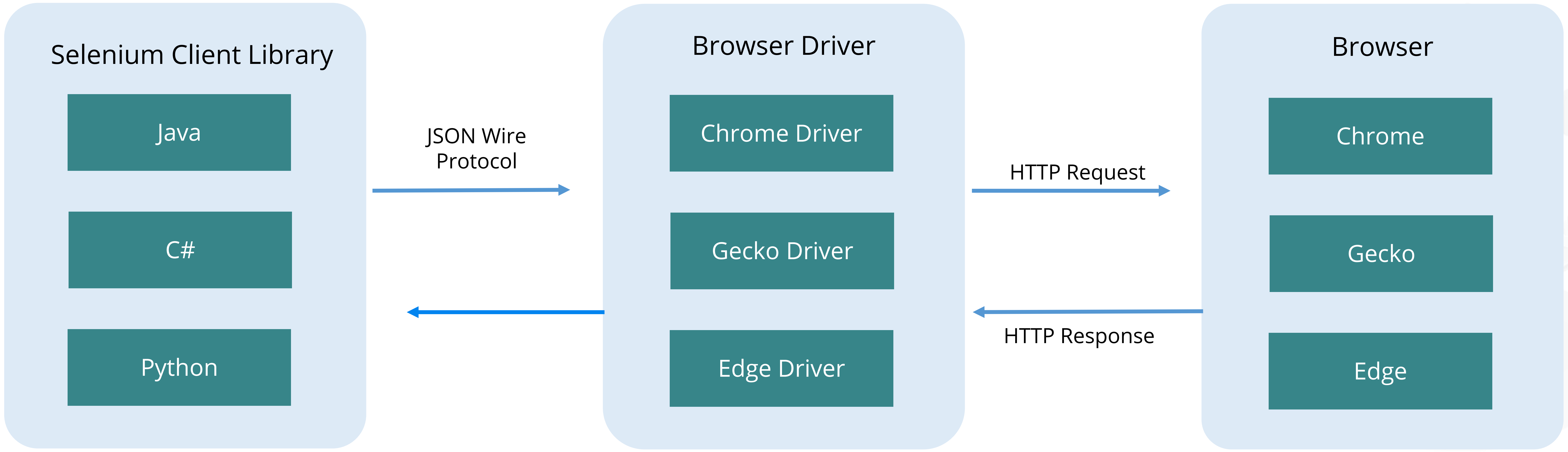
Selenium Architecture

Selenium Architecture

- Browser drivers and browsers may communicate with one another thanks to the Selenium WebDriver API.
- The **Selenium Client Library, JSON Wire Protocol, Browser Drivers, and Browsers** make up the four levels of this architecture.
- Languages including Java, Ruby, Python, C#, and others are available in the Selenium Client Library.

Selenium Architecture

Selenium architecture till v3:



Layers

Layers of Selenium architecture are:

Language bindings

To support multiple languages, selenium has language bindings. Supported language bindings are Java, C#, JavaScript, Python, etc.

Selenium WebDriver

- It is a set of APIs that makes the communication between programming languages and browsers possible.
- It has specific commands to perform actions on browsers like launching a URL and many more.

Layers

Browser Drivers

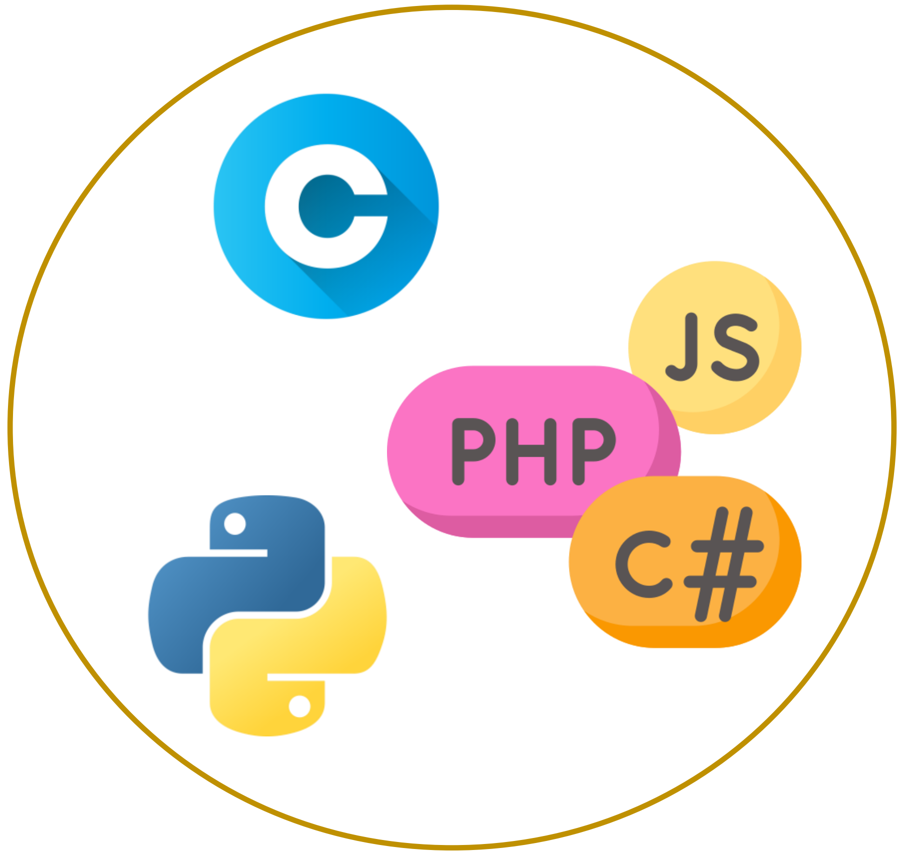
- A browser drivers helps Selenium WebDriver APIs to communicate with browser without revealing the internal logic of browser's functionality.
- The browser driver remains the same regardless of the language used for automation.

Browser

- Is where actions are performed.

WebDriver API

Selenium supports several programming language bindings like JAVA, C#, Python, JavaScript, etc., and supports all major vendors of browsers like Chrome, Firefox, Edge, Safari, etc.



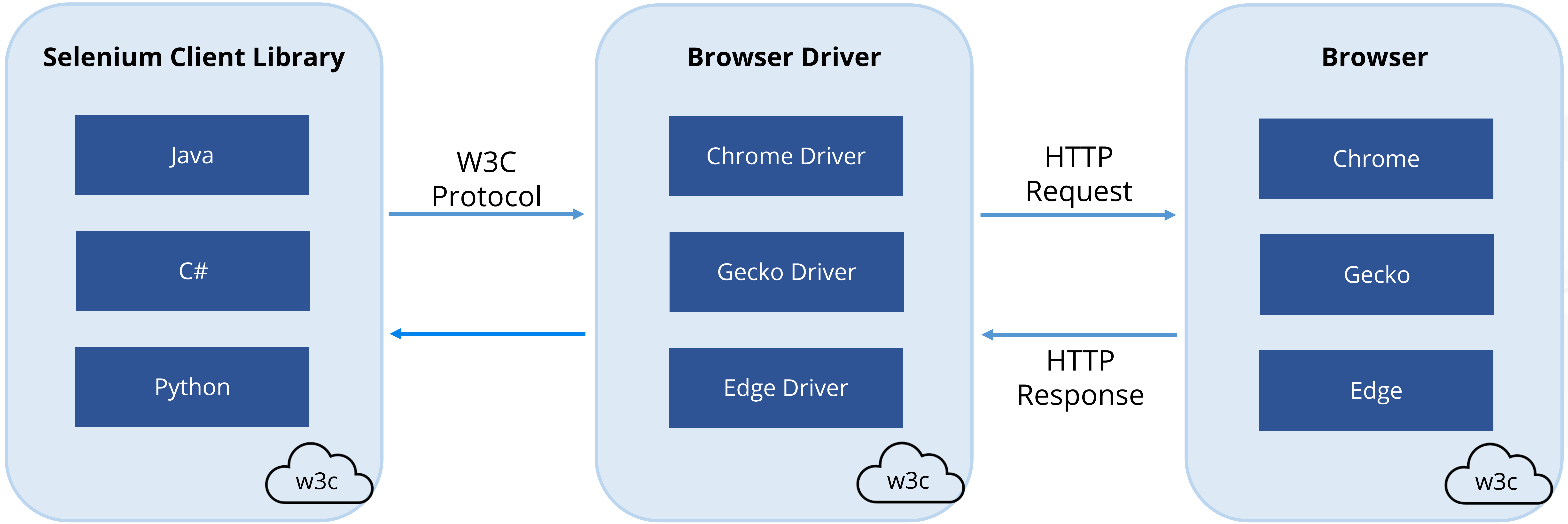
WebDriver API

Selenium WebDriver

- Selenium WebDriver is a set of well-designed object-oriented APIs that communicates with the browse.
- Selenium WebDriver launches a browser-specific server first and then sends instructions provided by programming statements to launch the server, such as loading a URL.
- Till v3, the WebDriver APIs are used to communicate with the browser using a JSON wire protocol. This wire protocol defines a RESTful web service using JSON over HTTP.

Selenium Architecture After v4

In v4, JSON wire protocol was replaced with W3C protocol.



Selenium Architecture After v4

- Most modern browsers, including Internet Explorer (Chrome, Firefox, Safari), are regarded as W3C compatible.
- Selenium 4 eliminates the need to apply **tweaks** to the test script to make it functional across various browsers.
- With WebDriver becoming fully W3C standardized, there are no compatibility concerns when using it with other frameworks.

Source link: https://www.oxfordwebstudio.com/en/did-you-know/what-is-w3c

WebDriver Commands

WebDriver Commands

The following categories is used to categorize the commands provided by Selenium WebDriver:

01

Browser Commands

02

Navigation Commands

03

Web Element Commands

Browser Commands

Selenium commands or APIs used to interact directly with browsers:

1. get()

get() method navigates to the URL. It accepts a single string argument, the URL of the page.

Syntax:

```
$ driver.get(URL_PATH);
```

Browser Commands

2. getTitle()

- getTitle() returns the title of the webpage from the active window.
- If the webpage does not have a title, a null string is returned. No argument is required for this command.

Syntax:

```
$ String page_title= driver.getTitle();
```

Browser Commands

3. getCurrentUrl()

getCurrentUrl() returns the URL of the webpage of the currently active window. No arguments are required, and it returns a string value.

Syntax:

```
$ String currentURL = driver.getCurrentUrl();
```

Browser Commands

4. getPageSource()

getPageSource() returns the source code of the current web page, takes no arguments and returns a **String** value.

Syntax:

```
$ String PageSource = driver.getPageSource()
```

Browser Commands

5. close()

The close() method closes the currently active window. No argument is required, and it returns a null.

Syntax:

```
$ driver.close()
```

6. quit()

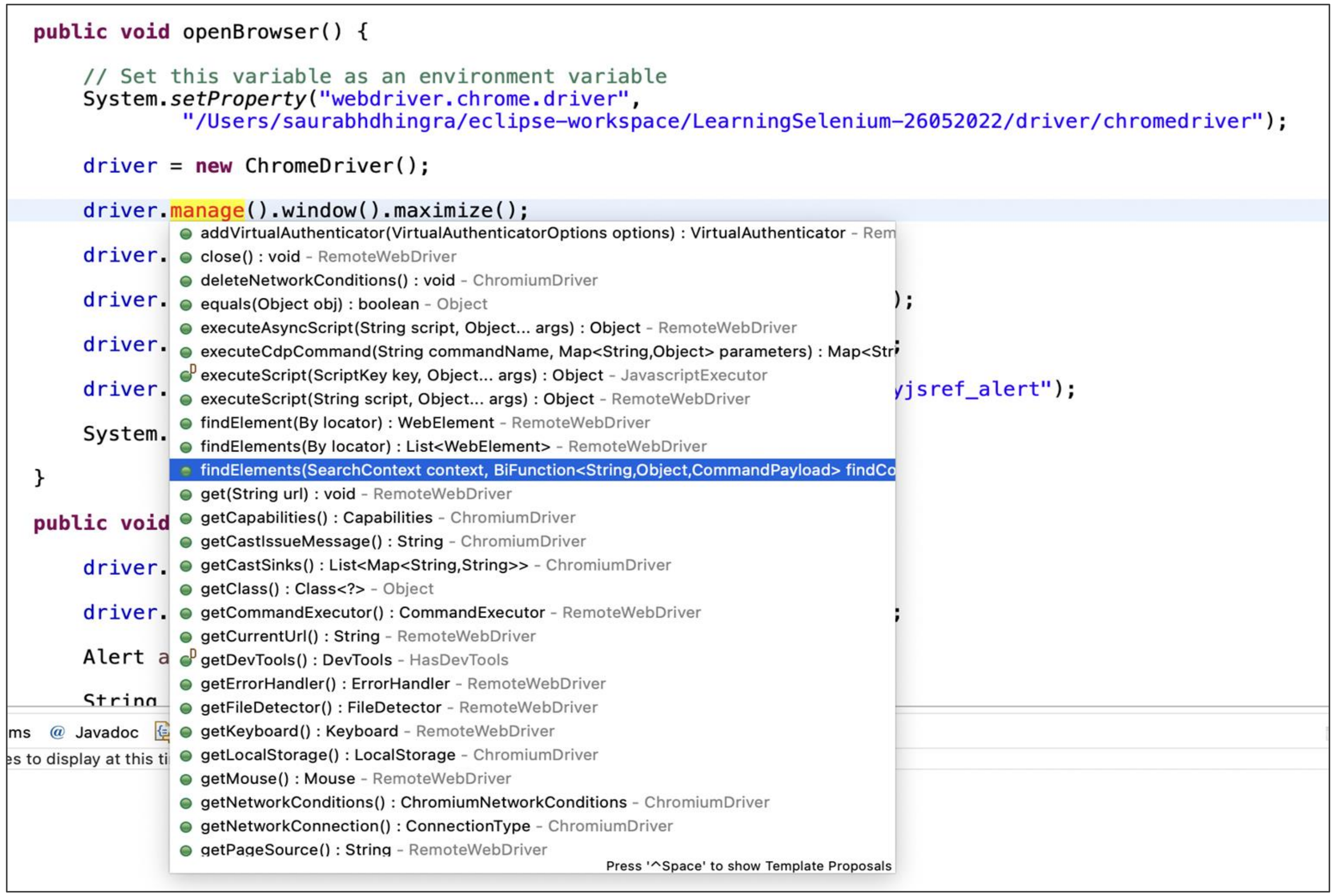
The quit() method closes all the windows open during the current Selenium session. No argument required and it return null.

Syntax:

```
$ driver.quit()
```

Other Browser Commands

Other methods like findElement() and findElements():

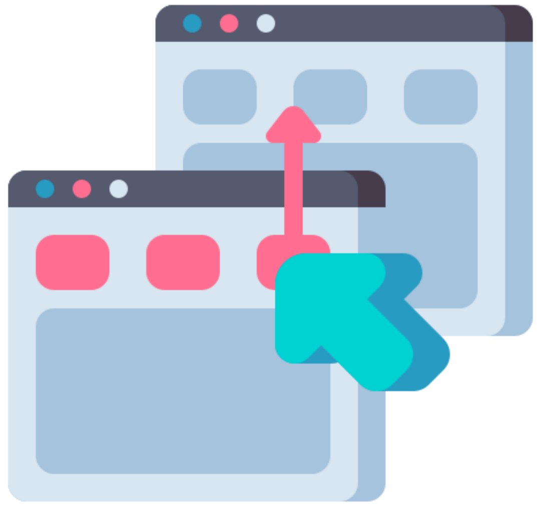


```
public void openBrowser() {

    // Set this variable as an environment variable
    System.setProperty("webdriver.chrome.driver",
            "/Users/saurabhdhingra/eclipse-workspace/LearningSelenium-26052022/driver/chromedriver");

    driver = new ChromeDriver();

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

    driver.

    driver.                                                                        );

    driver.                                                                        ;

    driver.                                                                        yjsref_alert");

    System.

}

public void

    driver.

    driver.                                                                        ;

    Alert a

    String
```

```
addVirtualAuthenticator(VirtualAuthenticatorOptions options) : VirtualAuthenticator - Rem
close() : void - RemoteWebDriver
deleteNetworkConditions() : void - ChromiumDriver
equals(Object obj) : boolean - Object
executeAsyncScript(String script, Object... args) : Object - RemoteWebDriver
executeCdpCommand(String commandName, Map<String,Object> parameters) : Map<Str
executeScript(ScriptKey key, Object... args) : Object - JavascriptExecutor
executeScript(String script, Object... args) : Object - RemoteWebDriver
findElement(By locator) : WebElement - RemoteWebDriver
findElements(By locator) : List<WebElement> - RemoteWebDriver
findElements(SearchContext context, BiFunction<String,Object,CommandPayload> findCo
get(String url) : void - RemoteWebDriver
getCapabilities() : Capabilities - ChromiumDriver
getCastIssueMessage() : String - ChromiumDriver
getCastSinks() : List<Map<String,String>> - ChromiumDriver
getClass() : Class<?> - Object
getCommandExecutor() : CommandExecutor - RemoteWebDriver
getCurrentUrl() : String - RemoteWebDriver
getDevTools() : DevTools - HasDevTools
getErrorHandler() : ErrorHandler - RemoteWebDriver
getFileDetector() : FileDetector - RemoteWebDriver
getKeyboard() : Keyboard - RemoteWebDriver
getLocalStorage() : LocalStorage - ChromiumDriver
getMouse() : Mouse - RemoteWebDriver
getNetworkConditions() : ChromiumNetworkConditions - ChromiumDriver
getNetworkConnection() : ConnectionType - ChromiumDriver
getPageSource() : String - RemoteWebDriver
                                        Press '^Space' to show Template Proposals
```

Navigation Command

Navigation commands are used for navigating between multiple pages.



For example, opening a web page URL, going to another web page by clicking any element, and using the browser's history to back, forward, or refresh the web page.

Navigation Command

Following are the Navigation commands:

driver.navigate().to(“<url>”)

This method navigates to the URL of the page passed as an argument.

driver.navigate().forward()

This method navigates you back to the browsing history.

driver.navigate().back()

This method navigates you forward in the browsing history.

driver.navigate().refresh()

This method reloads the webpage.

WebElement Command

A WebElement represents an HTML element on a web page.

Selenium WebDriver has WebElement interface to enable well-organized web page interactions, such as identifying elements, getting attribute properties, asserting text in WebElement, and more.

WebElement Command

Some commonly used methods are:

element.clear()

- This method clears the text of a text field.
- Syntax:

element.**clear**();

element.sendKeys()

- This method is used for passing “text” as a String (Char Sequence) to a web element like text field or text area.
- Syntax:

element.**sendKeys**("text");

WebElement Command

element.click()	• This method is used for clicking a web element like a link or a button on a web page. No argument is required. • Syntax: element.**click**();

element.isEnabled()	• This method checks whether the element is currently enabled or not. It takes no parameters and returns a boolean value (true/false).
	• Syntax:
	WebElement element = driver.**findElement**(By.**id**("UserID"));
	boolean status = element.**isEnabled**();

WebElement Command

element.isDisplayed()

- This method checks whether the element is visible on the screen or not. It takes no arguments and returns a boolean value (true/false).
- Syntax:

WebElement element = driver.**findElement**(By.**id**("CityName"));
boolean status = element.**isDisplayed**();

element.isSelected()

- This method checks whether the element is selected or not. It takes no input and returns a boolean value (true or false).
- Syntax:

WebElement element = driver.**findElement**(By.**id**("married-Unmarried"));
boolean status = element.**isSelected**();

WebElement Command

element.submit():

- If the current element is a form or an element within a form, this method performs a submit action. It takes no parameters and returns nothing.
- Syntax:

WebElement element = driver.**findElement**(By.**id**("Button"));
element.**submit**();

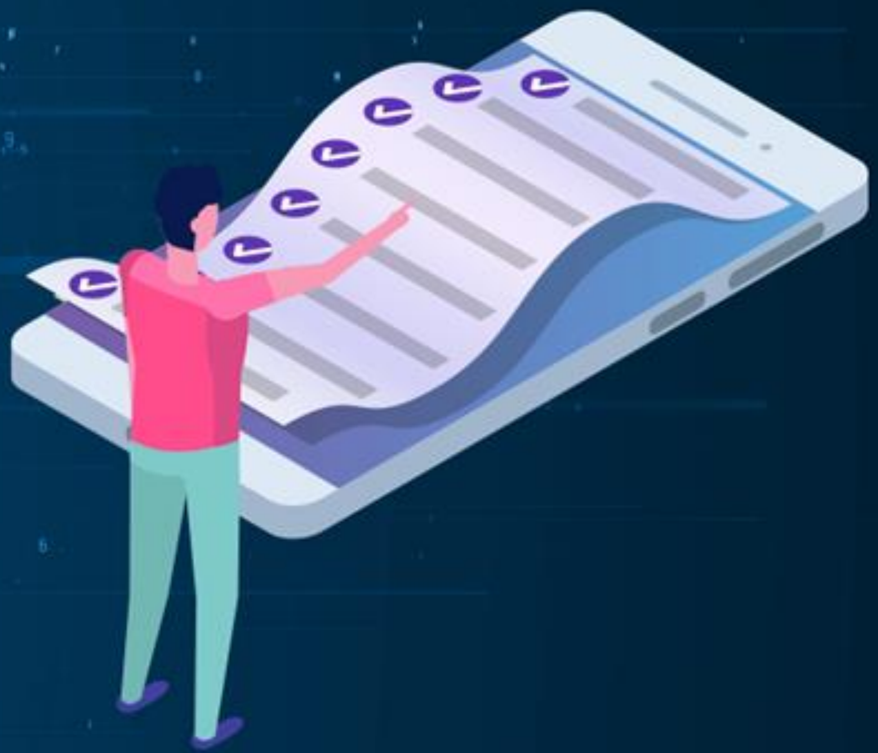
element.getAttribute()

- This method returns the value of the element's specified attribute. It takes a String as input and outputs a String value.
- Syntax:

WebElement element1 = driver.**findElement**(By.**id**("SubmitButton"));
String attVal = element1.**getAttribute**("id");

Key Takeaways

- Selenium was not W3C compliant till version 3, and the protocol used for the communication was JSON wire protocol between the client library and browser via browser driver.
- The layers of architecture are Language bindings, Selenium WebDriver, Browser Drivers, and Browser.
- In version 4, Selenium WebDriver was made W3C (World Wide Web Consortium) compliant, and JSON wire protocol was replaced with W3C.
- The categories used to categorize the commands provided by Selenium WebDriver are Browser Commands, Navigation Commands, and WebElement Commands.



#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
cout << "Hello, world!" << endl;
return 0;
}

Thank You