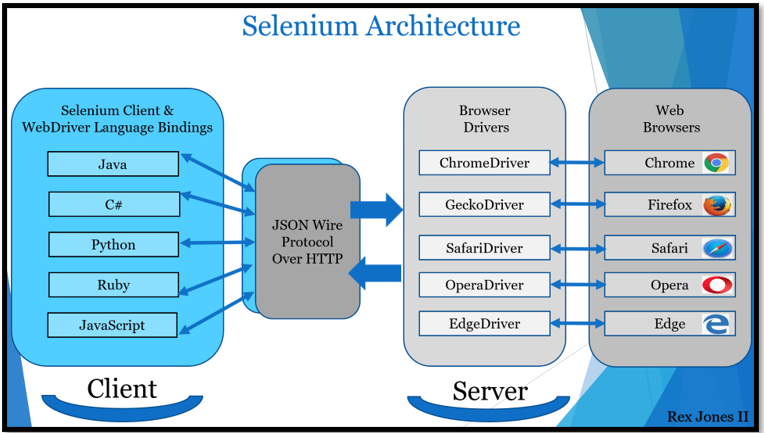
**Selenium 3 Architecture**

The architecture for Selenium 3 includes the [JSON Wire Protocol](https://github.com/SeleniumHQ/selenium/wiki/JsonWireProtocol). However, Selenium 4 does not include the JSON Wire Protocol, and that’s the contrast between Selenium 3 and Selenium 4. JSON stands for **J**ava**S**cript **O**bject **N**otation.

The JSON Wire Protocol has an assignment to transfer information from the client to the server over HTTP. HTTP is an acronym for **H**yper **T**ext **T**ransfer **P**rotocol. A Selenium request is sent from the Selenium Client and WebDriver Language Bindings component. Next, the request is received by JSON Wire Protocol Over HTTP, then secured by the Browser Driver.

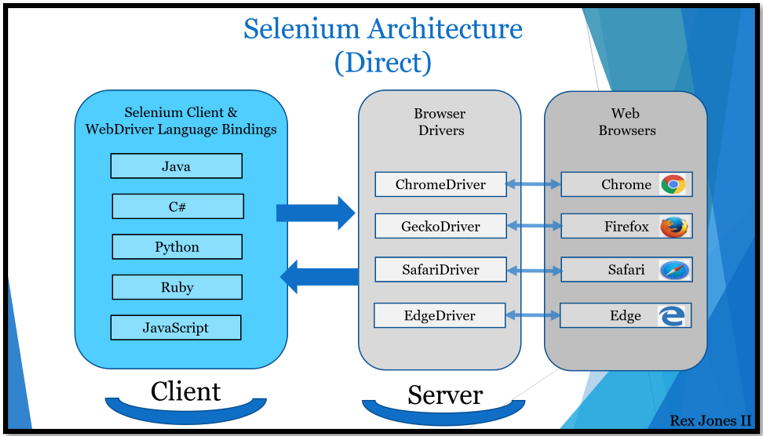
Afterwards, the request command is delivered to a Web Browser where the automation takes place. When the automation is complete, a response travels back to the Browser Driver, JSON Wire Protocol, and Selenium Client & WebDriver Language Bindings. Here is a diagram displaying Selenium 3 Architecture:



**Selenium 4 Architecture**

Unlike Selenium 3, Selenium 4 has direct communication between the client and server. The *client*still has 2 parts (Selenium Client & WebDriver Language Bindings) while Browser Drivers are the *server*.

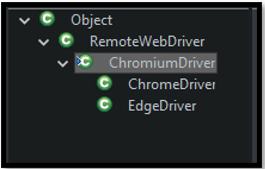
* **Selenium Client** sends out a request to perform a command.
* The **WebDriver Language Bindings** is a code library designed to drive actions.
* **Browser Drivers** receive the request and then return a response after an automation Test Script executes on the **Web Browser**:

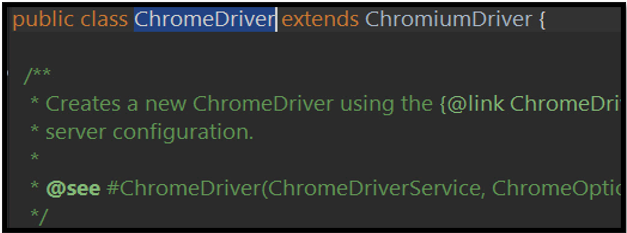


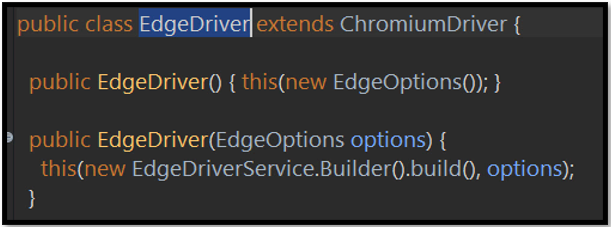
The Selenium Client & WebDriver Language Bindings is a part of the architecture where each language has their own unique bindings. Bindings mean that the same commands written for one language are also written for another language. For example, Java has a set of commands that have also been written for other languages (C#, Python JavaScript, and Ruby).

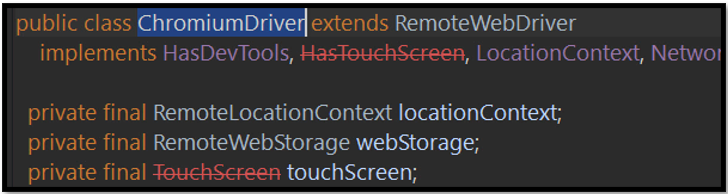
When it comes to the Browser Drivers and Web Browsers, WebDriver drives each browser using the browser’s built-in automation support. A Browser Driver such as ChromeDriver controls the Chrome browser.

In Selenium 4, a change was made to the drivers. ChromeDriver and EdgeDriver extend ChromiumDriver while RemoteWebDriver is the parent to ChromiumDriver. Here are some screenshots showing the relationship between each driver:









**Selenium 4 Advantages**

With the upgrade, Selenium 4 successively gained at least 3 advantages. The advantages are:

* **Standards**
* **Stability**
* **Updated Actions API**

The standards are an advantage because our Test Scripts run more consistently on each browser. All browser vendors have a standard. Since Selenium 4 is compliant with W3C WebDriver, there is no more required encoding and decoding of the request.

Stability is another advantage because of backwards compatibility. The Java Bindings and the Selenium Server provide a mechanism to use the old JSON Wire Protocol. There have been updates to the Actions API for keyboard and mouse events. It supports a way to carry out more than one action at the same time, like pressing 2 keys.

**Selenium 4 New Features**

The new features in Selenium 4 involve [Selenium IDE](https://blog.testproject.io/2020/12/21/selenium-ide-vs-selenium-ai-powered-test-recorder/), [Selenium WebDriver](https://blog.testproject.io/2020/11/11/selenium-webdriver-from-a-to-z/), and [Selenium Grid](https://www.selenium.dev/documentation/en/grid/)

* **Selenium IDE**allows us to record, playback the recording, edit, and debug our test.
* **Selenium WebDriver**is an API that executes our test by driving a browser for automating an Application Under Test (AUT).
* **Selenium Grid**executes our test across multiple browsers, operating systems, and machines.

Selenium IDE acquired Backup Element Selectors and Control Flows. The Backup Element Selectors record multiple locators for each element. Control Flows assist with executing statements in Selenium IDE. The Control Flows are Conditionals and Loops.

**Conditionals**

* if
* else-if
* else
* end

**Loops**

* do-repeat if
* while
* times
* forEach

In addition to the W3C WebDriver Protocol, Selenium WebDriver has new features that include Chrome DevTools, Relative Locators, multiple ways to capture a screenshot, and manage a window. Selenium Grid provides a new approach to setting up an infrastructure of browsers and operating systems. The initial grid was released in 2011, however now it’s more modern. Three chapters were written to dive deeper into the [Selenium 4 New Features](https://blog.testproject.io/2020/06/08/selenium-4-new-features/) for Selenium IDE, Selenium WebDriver, and Selenium Grid.

### Difference Between Selenium 3 And Selenium 4

| **Selenium 3** | **Selenium 4** |
| --- | --- |
| The Browser interaction is based on JSON wire protocol due to which it needs encoding and decoding of the API | It follows the W3C standard Protocol due to which driver and the browser communication follows the standard procedure, does not require encoding and decoding of the API, so communication happens directly. |

### Important Architecture Related Changes

**#1) Supports removed for some browsers:**Now, they have removed the supports for the browsers Opera and Phantom JS. Opera users can use the Chrome browser, whereas Phantom JS users can use Chrome or Firefox in headless mode.

**#2) Optimized Selenium Grid:**Selenium Grid was developed long back in 2011.

Selenium 4 has come up with new architecture to remove the issues which occurred earlier during installation and configuration, also during the connection between the hub and node machine.

**Let’s understand in brief about the Selenium Grid, which contains two major components:**

* **Node:** It is used to execute tests on individual computer systems. There can be multiple nodes in a grid.
* **Hub:** It is the central point from where it controls all the machines present in the network, and it contains only one hub which helps in allocating test execution to different nodes.

But in Selenium 4, Grid is very flexible. It allows us to test the cases against multiple browsers, browsers of different versions, and also on different Operating systems. Even now, there is no need for a setup to start hub and nodes individually once the user starts the server, the Grid automatically works as both nodes and hub.

It also supports advanced tools like Docker, AWS, Azure, and much more, useful in the DevOps process. Now Grid has a more user-friendly UI and contains relevant information related to the session, running, capacity, etc.

***>> Click***[***here***](https://github.com/SeleniumHQ/selenium/wiki/Selenium-Grid-4)***for more information.***

**#3) Standardized Documentation:**Documents play an essential role for any user, and Selenium documents were not updated since Selenium 2.0.

With Selenium 4, they have updated all the official documents related to Selenium that include web driver, Grid, and IDE to explain the latest changes.

**#4) Improvement in Selenium 4 IDE:**Selenium IDE is a record and playback tool that is used for user interaction with the browser, and the new version of IDE is available with more features

**These features include:**

* It improves the browser support. Now with a new version, any browser vendor can easily plug into the latest Selenium IDE.
* CLI Runner will be based on NodeJS, not the HTML-based runner, and it supports the parallel execution from which it provides the report with the total number of test cases passed/failed along with execution time taken.

**#5) Better Monitoring:** Logging and request tracing process are now improved to make a better grip on the debugging process for automation engineers as it is not limited to DevOps only.

### Features Of Selenium 4

**The features are as follows:**

**#1) Capture screenshot of specific web element:**

Earlier, users can take a screenshot of the entire page as there was no provision to take the screenshot of the specific web element. But with Selenium 4, users can take the screenshot of a specific web element.

**Please find the below code:**

|  |
| --- |
| WebElement logo=driver.Findelement (By.xpath(“//div[@id=’divLogo’]//img”));  File file=logo.getScreenshotAs(OutputType.FILE);  File destFile =new File(“logo.png”);  FileUtils.copyFile(file,destfile); |

**#2) Open the new tab on the browser:**

Now, in Selenium 4, the user can open a URL in a new tab along with the existing tab.

**For Example:** If the user wants to open 2 URLs in two different tabs at the same time, the user can do that with the Selenium 4.

**Please find the below code for reference:**

|  |
| --- |
| driver.get(https://www.google.com/);  driver.switchTo().newWindow(WindowType.TAB);  driver.navigate().to(https://www.crmpro.com/); |

**#3) Open a new window on the browser:**

Using Selenium 4, the users can also open the new window on the same browser.

**For Example,** if the user wants to access two applications in the same browser, the user can now do this.

**Please find the below code for reference:**

|  |
| --- |
| driver.get(https://www.google.com/);  driver.switchTo().newWindow(WindowType.WINDOW);  driver.navigate().to(https://www.crmpro.com/); |

**#4) Object Location:**

Now with Selenium 4, users can achieve the coordinates, dimension, height, width, etc. as the location of the web elements or object.

**Please find the below code for your reference:**

|  |
| --- |
| WebElement logo1=driver.Findelement(By.xpath(“//div[@id=’divLogo’]//img”));  System.out.println(“Height:” +logo.getRect().getDimension().getHeight());  System.out.println(“Height:” +logo.getRect().getDimension().getWidth());  System.out.println(“X Location: “ +Logo.getRect().getX());  System.out.println(“Y Location: “ +Logo.getRect().getY()); |

**#5) Relative Locators:**

These are also known as Friendly Locators, and this functionality is being added to find out the element which is present nearby to other web element or, we can say that it can find the web elements based on GUI location.

**There are five locators added in Selenium 4:**

* **below():** Web element located below for the specified element.
* **toLeftOf() :** Target web element which is present to the left of specified element.
* **toRightOf():** Target web element which is presented to the right of a specified element.
* **above():** Web element located above for the specified element.
* **near() :** Target web element which is away(approx. 50 pixels) from the specified element.

**Note:** All the above relative locators method support “withTagName” method.

**The below example is for the toLeftof() and below() locators:**

|  |
| --- |
| WebElement book;  book = driver.Findelement(RelativeLocators.withTagName(“li”).toLeftOf(By.id(“pid1”))  .below(By.id(“pid2”)));  String id1=book.getAttribute (“id1”); |

**The below example is for the toRightOf() and above() locators:**

|  |
| --- |
| WebElement book1;  book1 = driver.Findelement(RelativeLocators.withTagName(“li”).toRightOf(By.id(“pid1”))  .above(By.id(“pid2”)));  String id2=book1.getAttribute (“id2”); |

**#6) Chrome Dev tools:**

In the new version of Selenium, they have made some internal changes in the API. Earlier in Selenium 3, the Chrome driver extends directly to the Remote Web Driver class. But now in Selenium 4, Chrome driver class extends to Chromium Driver. Chromium Driver class has some predefined methods to access the dev tool.

**Note:** Chromium Driver extends the Remote Web driver class.

**By using the API, we can perform the following operations:**

* Enable Network Offline
* Enable Network Online
* Get Console Logs
* Load Insure Web Site