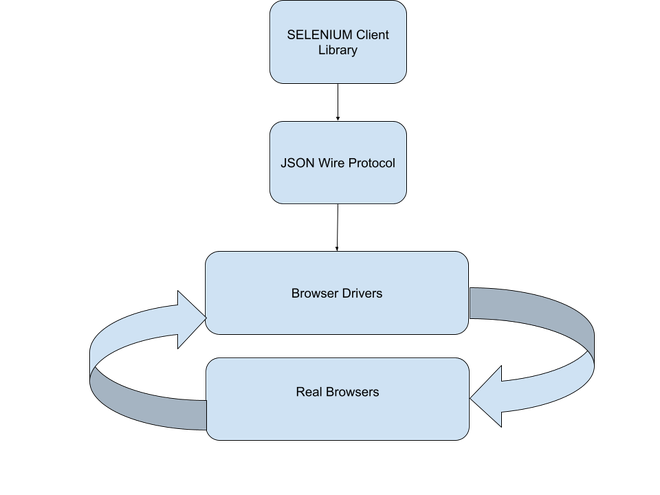


**Working of Selenium 3 WebDriver**

1. Write a test script using the Selenium client library in your preferred language.
2. The test script sends commands through the client library to interact with the browser.
3. The client library converts commands into JSON format and sends them via HTTP request.
4. The browser driver decodes JSON commands and interacts with the real web browser.
5. Browser performs actions (e.g., clicking buttons, entering text) on the web page based on the received commands.

Architecture of Selenium 3 WebDriver

**Architecture of Selenium 4 WebDriver**

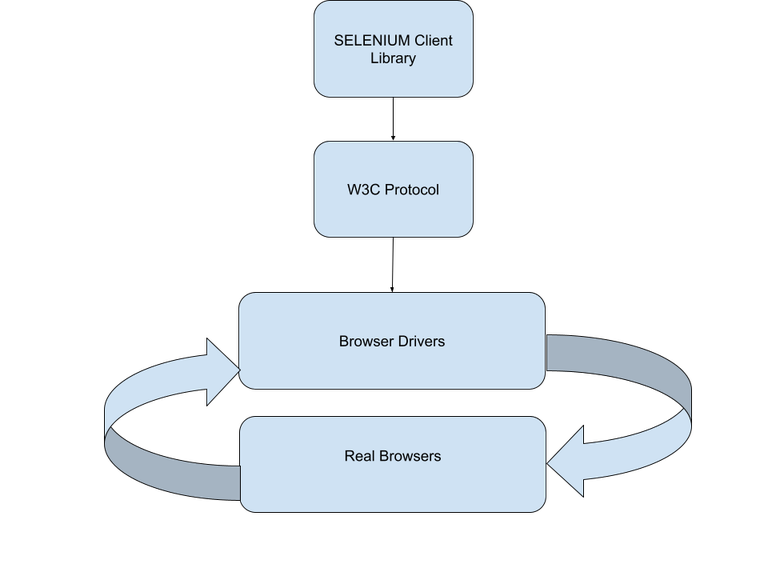
Selenium four brings significant improvements to the architecture, often with the introduction of the [W3C WebDriver Protocol](https://www.geeksforgeeks.org/w3c-full-form/). This protocol standardizes interactions between the purchaser and server, selling higher compatibility and consistency across one-of-a-kind implementations. Moreover, Selenium 4 affords a better guide for present-day net technology and progressed overall performance. The architecture of Selenium 4 WebDriver has made a key change compared to Selenium 3 which is a communication protocol. Like Selenium 3, Selenium 4 offers client libraries for various programming languages, which help WebDriver interact with the browser. WebDriver W3C protocol is the major change in Selenium 4 as it completely replaces JSON Protocol which was in Selenium 3. The WebDriver W3C Protocol is defined by the World Wide Web Consortium (W3C) that ensure better compatibility and stability on different browsers and client libraries.

**Components of Selenium 4 WebDriver**

1. **Selenium Client Library:**This component provides language-specific bindings or APIs (e.g., Java, Python, Ruby) that allow users to write test scripts and interact with the WebDriver.
2. **WebDriver W3C Protocol:** WebDriver is a protocol that provides a standard way for web browsers to communicate with an automation script. In Selenium 4, it focuses on W3C WebDriver Protocol, for better consistency and compatibility across different browsers.
3. **Browser Drivers:** These are executable files that establish a communication channel between the WebDriver and the actual web browsers such as Chrome, Firefox, Safari, etc. Each browser requires its specific driver (e.g., ChromeDriver, GeckoDriver, etc.) to enable WebDriver to control and automate browser actions.
4. **Real Browsers:** These are web browsers like Google Chrome, Mozilla Firefox, Microsoft Edge, etc., where the actual testing and automation take place. The WebDriver interacts with these browsers through their respective browser drivers to perform actions like clicking elements, filling forms, navigating pages, and validating content.

**Working of Selenium 3 WebDriver**

1. Write your test script using the Selenium client library in your comfortable language.
2. The test script sends commands through the client library to interact with the browser.
3. Client library converts commands into WebDriver W3C Protocol format.
4. The browser driver receives commands via WebDriver W3C Protocol.
5. Browser drivers understand commands and interact with real web browsers.

Architecture of Selenium 4 WebDriver

**Difference between Architecture of Selenium 3 & Selenium 4**

Selenium 4 has some big differences from Selenium 3. the changes make the Selenium framework better and more powerful. They will improve how it will be working and make it easier to use. Upgrading to Selenium 4 can be helping developers and testers to build the better web applications with minimum efforts.

| **Aspect** | **Selenium 3** | **Selenium 4** |
| --- | --- | --- |
| **WebDriver Protocol** | JSON Wire Protocol | W3C WebDriver Protocol |
| **Communication** | Client-server model | Client-server model |
| **Protocol Standardization** | Not fully standardized | Fully standardized (W3C specification) |
| **Browser Compatibility** | Limited | Improved compatibility |
| **Performance** | Moderate | Improved performance |
| **Support for Modern Web Tech** | Limited | Better support for modern web technologies |
| **Native Events** | Relies on the browser's native automation engine | Enhanced native event support |
| **Interactions with Browser** | Through browser-specific drivers (e.g., GeckoDriver, ChromeDriver) | Through browser-specific drivers (e.g.,[GeckoDriver](https://www.geeksforgeeks.org/how-to-run-gecko-driver-in-selenium-using-java/), ChromeDriver) |
| **Future-Proofing** | Limited future-proofing with reliance on browser-specific implementations | Improved future-proofing with standardized protocol |