**1. Enhanced Selenium Grid**

The new Selenium Grid comes with Docker support. This will enable developers or testers to spin up the containers rather than setting up heavy virtual machines. Moreover, it is redesigned in a way that will allow QAs to deploy the grid on Kubernetes for better scaling.

Managing Selenium Grid is now smooth and easy as there will no longer be any need to set up and start hubs and nodes separately.

Teams or individual testers can now deploy the grid in three modes:

1. Standalone mode
2. Hub and Node
3. Fully distributed

Unlike in earlier versions, the Grid will now support IPv6 addresses and one can communicate with the Grid using the HTTPS protocol. In Grid 4, the configuration files used for spinning up the grid instances can be written in TOML ([Tom’s Obvious, Minimal Language](https://github.com/toml-lang/toml)) which will make it easier for humans to understand.

The Grid in Selenium 4 also comes with an enhanced user-friendly GUI. Overall, the revamped Selenium Grid will enhance the DevOps process as it provides compatibility with tools like Azure, AWS, and more.

**2. Upgraded Selenium IDE**

Most QA engineers using the Selenium tool suite are familiar with the popular record and playback tool – [Selenium IDE](https://www.browserstack.com/guide/what-is-selenium-ide). This Selenium IDE was earlier available just as a Firefox add-on. Later, it got deprecated with the introduction of the latest Firefox versions. This is because the add-ons in the latest Firefox (ver. 55) were standardized under the **Web Extension mechanism.**

With Selenium 4, the IDE is revived and now its add-on is available for major web-browsers like Firefox and Chrome. The add-on for Selenium IDE is now also [available on the MS store](https://microsoftedge.microsoft.com/addons/detail/selenium-ide/ajdpfmkffanmkhejnopjppegokpogffp).

The new Selenium 4 IDE provides some notable features like:

1. Improved GUI for intuitive user experience.
2. The new IDE also comes bundled with a SIDE tool aka Selenium IDE runner. It allows QAs to run .side projects on a node.js platform. This SIDE runner also enables individual QAs to run cross browser tests on local or [Cloud Selenium Grid](https://www.browserstack.com/cloud-selenium-grid).
3. Improved control flow mechanism that enables testers to write better “while” and “if” conditions.
4. The new IDE comes with an enhanced element locator strategy (Like a backup strategy) which helps locate an element in case the web element couldn’t be located. It will result in the creation of stable test cases.
5. The code for test cases recorded using Selenium IDE can be exported in the desired language binding like Java, C#, Python, .NET, and JavaScript.

**3. Relative Locators in Selenium 4**

Selenium 4 brings an easy way of locating elements with the inclusion of relative locators. This means testers can now locate specific web elements using intuitive terms that are often used by users like:

1. To left of
2. To right of
3. Above
4. Below

The introduction of this new method in Selenium 4 helps locate web elements based on the visual location relative to other DOM elements. One can refer to this [source](https://angiejones.tech/selenium-4-relative-locators/) to learn more about the implementation of relative locators in detail.

#### ****. Improved Documentation****

The documentation section has been revamped significantly with a neat UI for navigating to the desired section or page. This will help testers and developers find relevant information they need for a specific tool, language binding, etc.

The exhaustive documentation covers information about all tools and APIs under the Selenium umbrella. This will help individual testers (particularly the beginners) get acquainted with all the features and prerequisites to get started with [automation testing](https://www.browserstack.com/automate).

#### ****5. Support for Chrome Debugging Protocol****

Selenium 4 comes with native support for Chrome DevTools Protocol. This means QAs can now use Chrome development properties like Fetch, Network, Profiler, Performance, Application cache, and more. QAs can also leverage the APIs offered by Chrome DevTools to [simulate poor network conditions](https://www.browserstack.com/guide/how-to-perform-network-throttling-in-chrome) and perform [geolocation testing](https://www.browserstack.com/geolocation-testing).

Using this API will also help developers or QAs to test and resolve critical bugs for specific web-pages faster and on the fly.

#### ****6. Better Window/Tab Management in Selenium 4****

There are several instances in test automation wherein one might need to open a particular link in a new tab or window to perform certain actions. To achieve this in Selenium 3, QAs had to create a new driver object and then perform the **switch operation** using the **WindowHandle method** to perform subsequent steps.

This is set to change in Selenium 4 as it comes with a new API – **newWindow** that allows users to create and switch to a new window/tab without creating a new WebDriver object.

**Sample code snippet to open a new window**

driver.get("https://www.google.com/");

// Opens a new window and switches to new window

driver.switchTo().newWindow(WindowType.WINDOW);

// Opens BrowserStack homepage in the newly opened window

driver.navigate().to("https://www.browserstack.com/");

driver.get("https://www.google.com/");

// Opens a new tab in existing window

driver.switchTo().newWindow(WindowType.TAB);

// Opens Browserstack homepage in the newly opened tab

driver.navigate().to("https://www.browserstack.com/");

**7. Deprecation of Desired Capabilities**

[Desired Capabilities](https://www.browserstack.com/guide/desired-capabilities-in-selenium-webdriver) were primarily used in the test scripts to define the test environment (browser name, version, operating system) for execution on the Selenium Grid.

In Selenium 4, capabilities objects are replaced with Options. This means testers now need to create an Options object, set test requirements, and pass the object to the Driver constructor.

Listed below are the Options objects to be used going forward for defining browser-specific capabilities:

1. Firefox – FirefoxOptions
2. Chrome – ChromeOptions
3. Internet Explorer (IE) – InternetExplorerOptions
4. Microsoft Edge – EdgeOptions
5. Safari – SafariOptions

**8. Modifications in the Actions Class**

Actions class in Selenium is primarily used to simulate input actions from mouse and keyboard on specific web elements (For eg: Left click, Right click, Double click, etc)

In Selenium 4, several new methods have been added to the Actions class:

* **click(WebElement)**

This method is added to Actions class to replace the **moveToElement(onElement).click()**. It is used to click on a certain web element.

* **clickAndHold(WebElement)**

This method will replace the **moveToElement(onElement).clickAndHold()**. It is used to click on an element without releasing the click.

* **contextClick(WebElement)**

This method will replace moveToElement(onElement).contextClick(). It will perform the right click operation.

* **doubleClick(WebElement)**

This method is added to replace **moveToElement(element).doubleClick()**. It will perform a double click on an element.

* **release()**

This method (user for releasing the pressed mouse button) was initially a part of org.openqa.selenium.interactions.ButtonReleaseAction class. Now with the updated version of Selenium, it has been moved to Actions class.

Considering all the points above, one can conclude that the features in Selenium 4 appear quite promising in terms of making the test process easier. However, one must also bear in mind that, regardless of the Selenium version in use, the tests must be performed on real devices for accurate test results.

For that instance, teams can leverage BrowserStack’s [Cloud Selenium Grid](https://www.browserstack.com/cloud-selenium-grid/) for running parallel tests across 2000+ real devices and browsers for testing at scale. It is compatible with Selenium 4 and enables QAs to run tests in [real user conditions](https://www.browserstack.com/real-user-conditions-testing-on-browserstack).

[BrowserStack Automate](https://www.browserstack.com/automate) offers complete support for Selenium 4, and its BiDi APIs. The following are a few of the Selenium 4 features Automate now supports out of the box:

* **Relative Locators:** Write simpler and more stable tests. Select page elements on the basis of their relative location to other elements. You can use the locators **above, below, toLeftOf, toRightOf,** and **near**.
* **Network Interception:** Intercept, validate and modify network requests and responses. Add additional headers, fail requests, modify the request/response body, and more.
* **Listen to JS Exceptions and Console Logs:** Monitor browser console logs and keep an eyes out for JS exceptions. Register callbacks to process exceptions and events.

Automate also supports other Selenium 4 features like Mocking Geolocation, Basic Auth, Device Mode simulation, and Network Simulation. Additionally, Selenium 4 is fully backward compatible with existing Selenium versions. That means you can migrate without compromising on the stability of your existing test cases.