ProfElect3

Module 6

introduction to SELENIUM

Automation ↗

Start

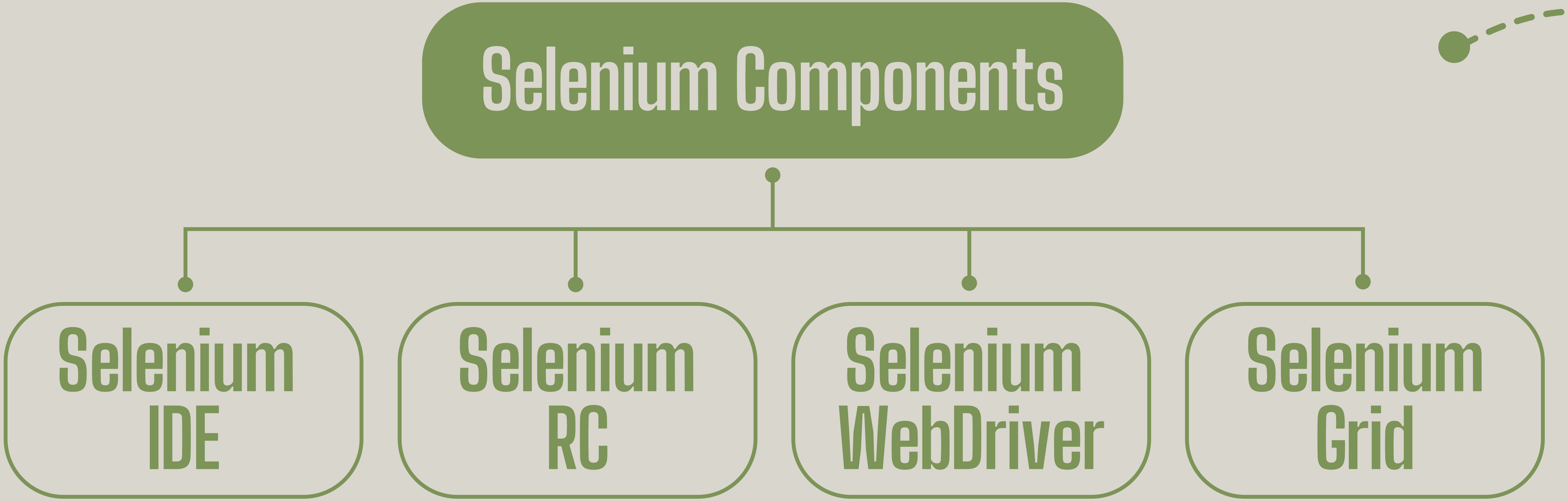
October 2025

Instructor: Florabel Banal

What is Selenium ?

Selenium is a free (open-source) automated testing framework used to validate web applications across different browsers and platforms. You can use multiple programming languages like Java, C#, Python, etc. to create Selenium Test Scripts. Testing done using the Selenium testing tool is usually referred to as Selenium Testing.

It can automate all websites and does not depend on the technology in which the application is designed. Performance and execution speed of Selenium Automation tool is much better than any other tools which are available in the market.



Selenium Components

Selenium IDE

Selenium RC

Selenium WebDriver

Selenium Grid

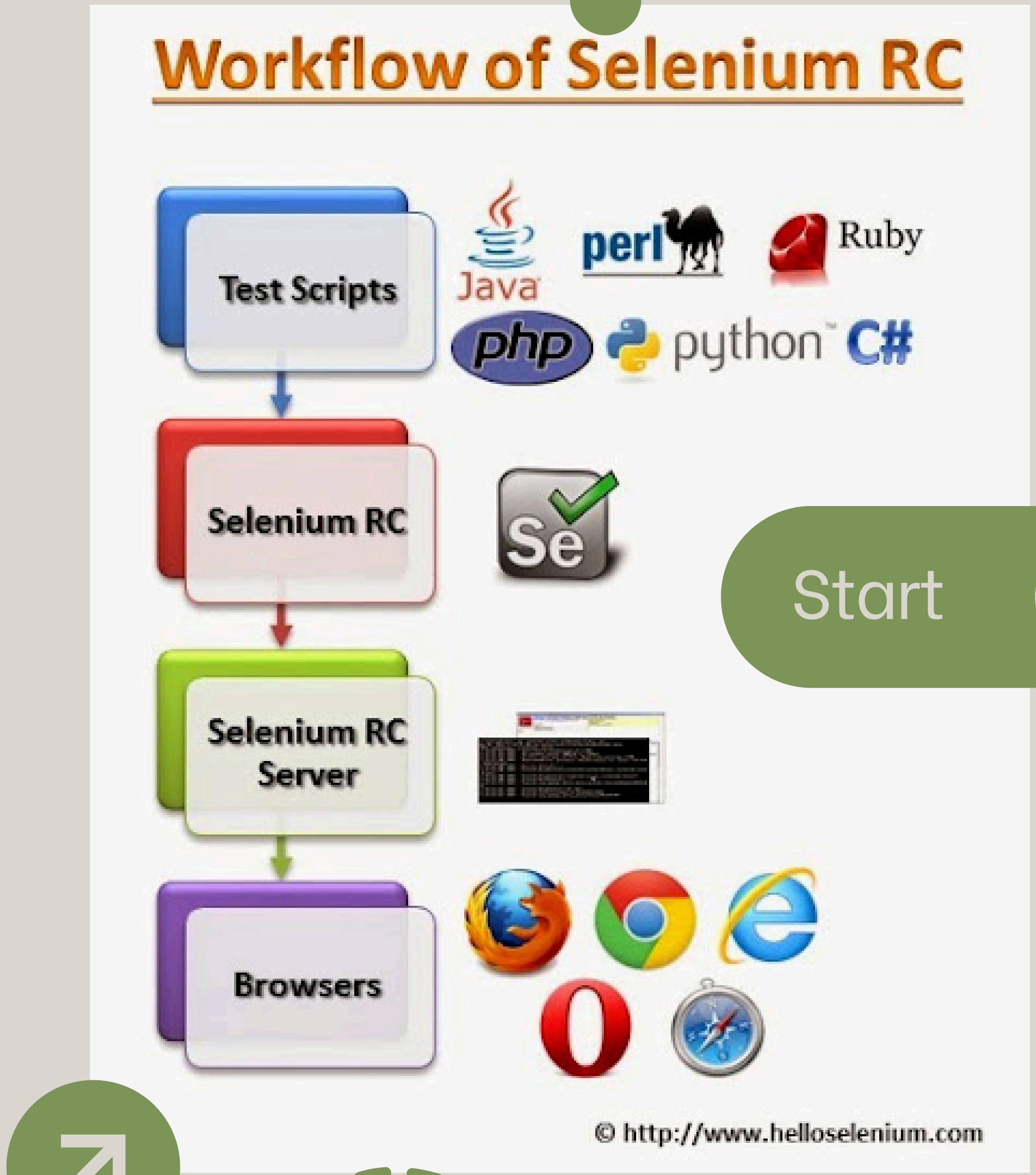
Selenium IDE

Selenium IDE is a simple record and playback kind of tool which comes as an add-on for Mozilla Firefox only. It is used for prototype testing. Test cases written in IDE can be exported in many programming languages like Ruby, Java, C#, etc. Edit and Debug options along with record are also available. It is an excellent tool for beginners to understand the syntax of Selenium WebDriver.

Selenium RC ?

Selenium RC (Remote Control) was the first tool of Selenium Suite. Earlier it was known as JavaScript Executor. RC was the tool which made Selenium famous in the market.

It was the first tool which provided the support for multiple programming languages (JAVA, Ruby, Perl, PHP, Python, and C#).

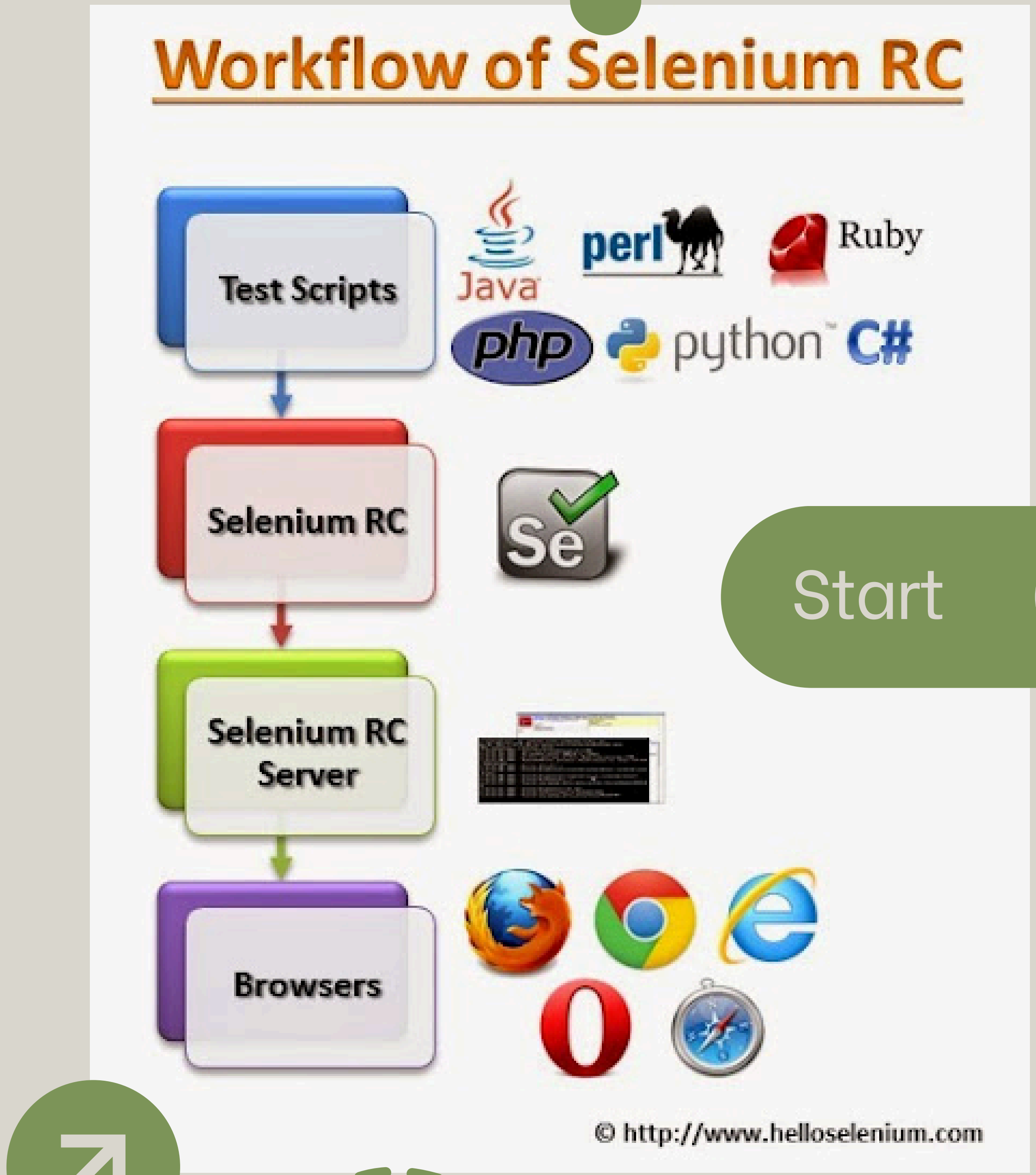


Start

Selenium RC ?

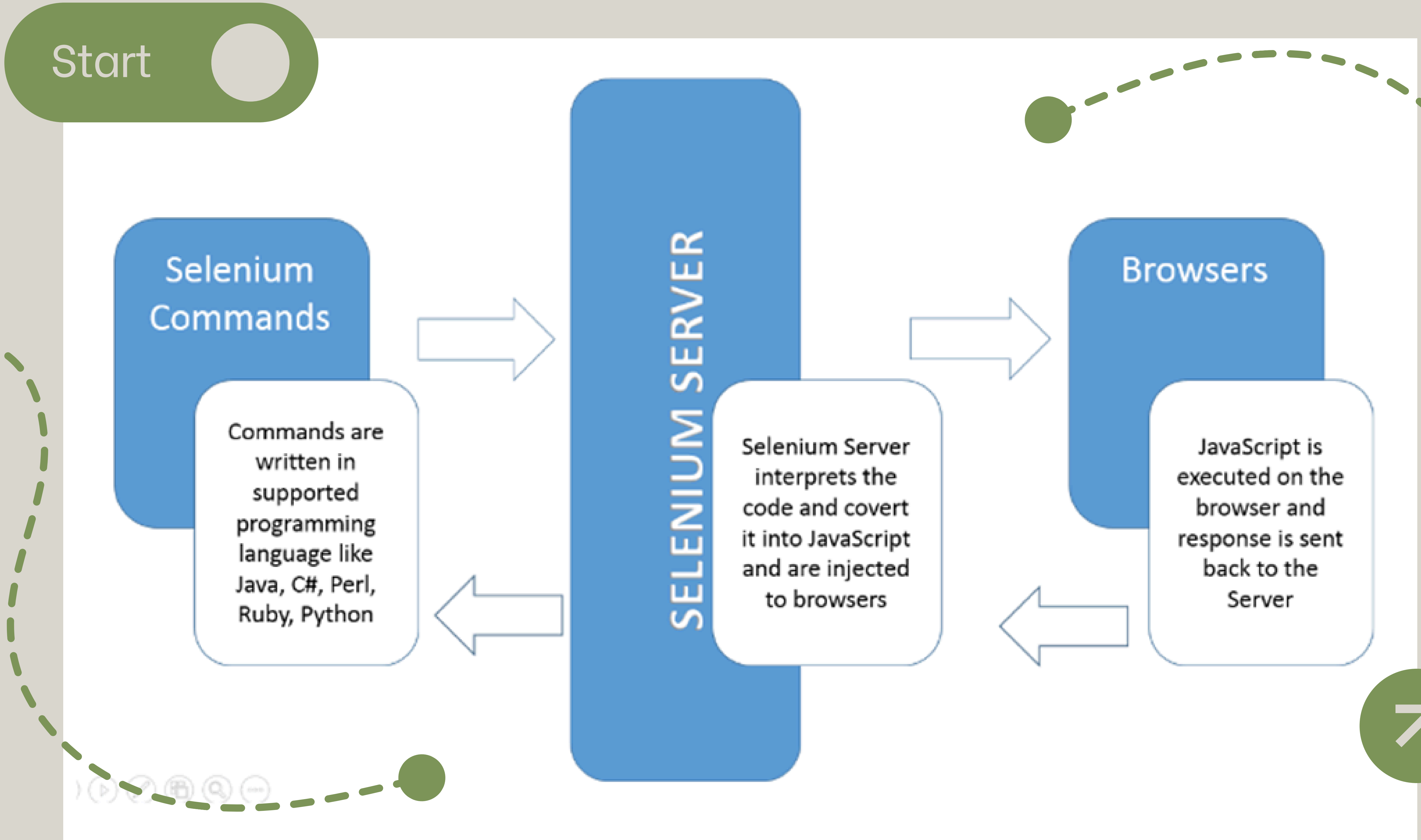
It also supported almost all the major vendors of Browsers like Mozilla Firefox, Google Chrome, Internet Explorer. All the browsers which support JavaScript can be automated using this tool.

Selenium RC is also known as Selenium 1.



Start

Start



Selenium Commands

Commands are written in supported programming language like Java, C#, Perl, Ruby, Python

SELENIUM SERVER

Selenium Server interprets the code and covert it into JavaScript and are injected to browsers

Browsers

JavaScript is executed on the browser and response is sent back to the Server

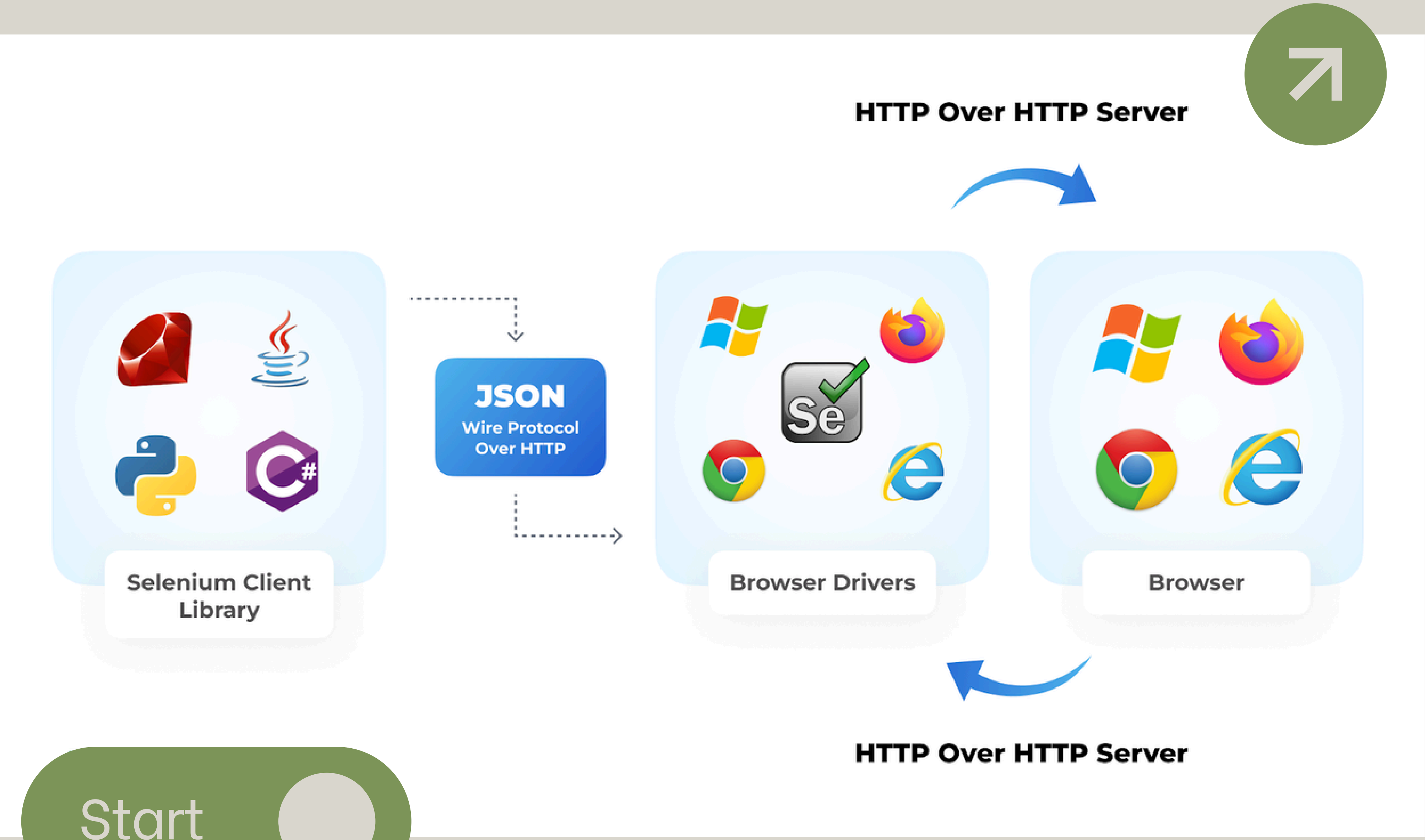
Selenium RC ?

The architecture of Selenium RC:

In Selenium RC, there is a manual process called Selenium Server is mandatory to start before execution, which acts as a middleman between the code and the browser. The commands (API's) are sent to Server. It interprets the command and converts it into JavaScript and then JavaScript is injected to the browser. Now the browser executes the javascript and responds to a server, which again interprets the command and returns to code in the respective language.

Selenium WebDriver ?

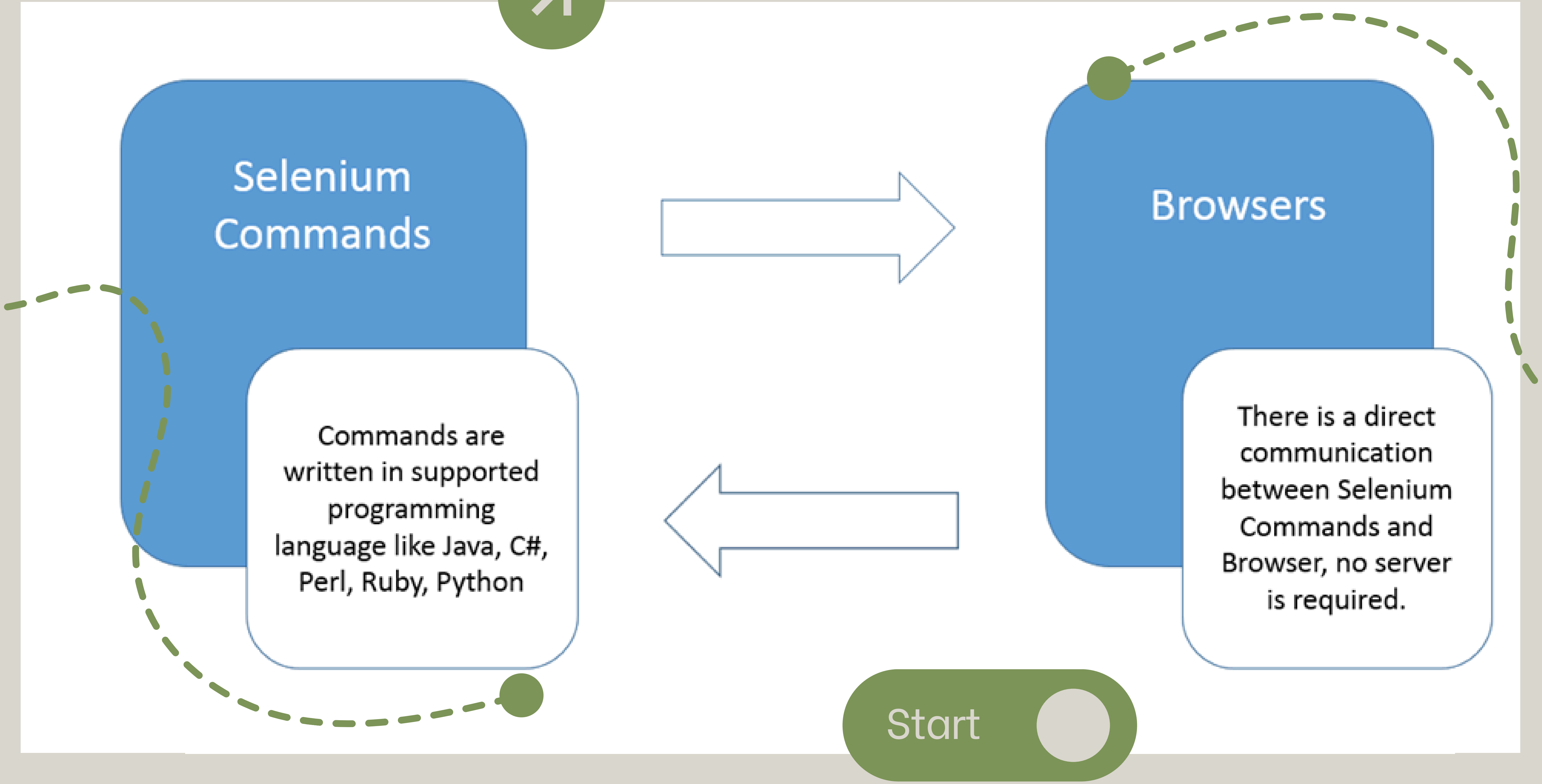
Selenium WebDriver is the most important tool of the Selenium suite. Because of many limitations with RC, WebDriver was developed. It does not require any manual process like Selenium Server. There is a direct communication between code and browser.



Start

SELENIUM WEBDRIVER FEATURES

- Open source
- Supports all the key vendors of the browser like Mozilla Firefox, Internet Explorer, Google Chrome, Safari, etc.
- Support Multiple languages like C#, JAVA, Ruby, Perl, Python, and PHP.
- Supports multiple platforms like Linux, Windows, MAC, etc.
- No middleman like Selenium RC server is required.
- Easy to remember API's.
- Easy to integrate with Testing frameworks.
- Framework Development.
- Parallel Testing capabilities.



Selenium Commands

Commands are written in supported programming language like Java, C#, Perl, Ruby, Python

Browsers

There is a direct communication between Selenium Commands and Browser, no server is required.

Start

Selenium WebDriver ?

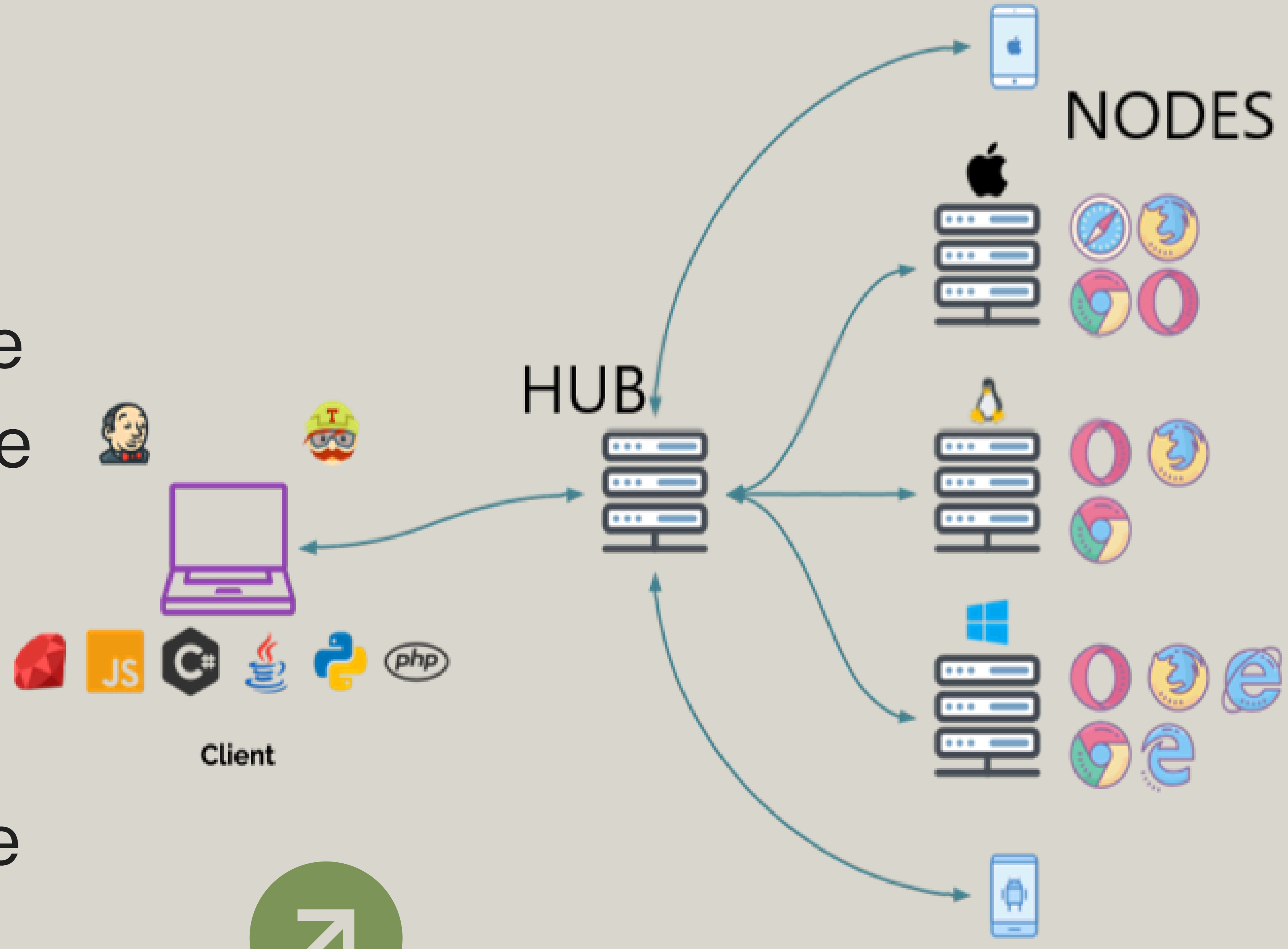
The architecture of Selenium WebDriver:

As discussed earlier, Selenium WebDriver does not require Selenium Server as a middleman. The API's written in WebDriver can directly interact with browsers.

Selenium Grid ?

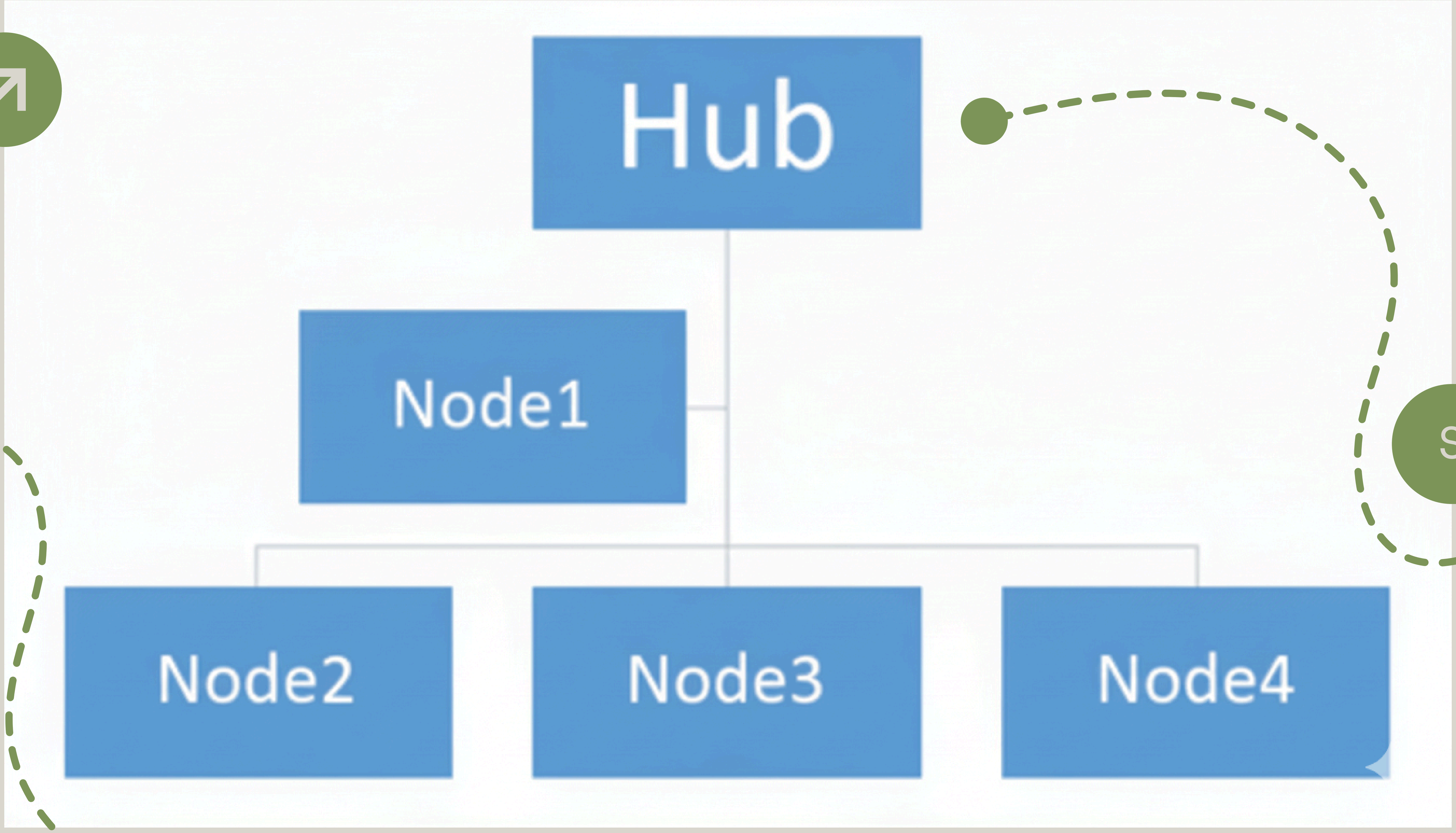
It is the last component of the selenium suite and is used for parallel testing or distributive testing. It allows us to execute test scripts on different machines at same time.

There is a Hub which controls the execution on various machines, and there are multiple nodes on which actual implementation is done.



SELENIUM GRID FEATURES

- Parallel Execution on multiple nodes
- Platform Independent, support almost all Operating System
- Language Independent.
- Browser Independent supports almost all the main vendors of Browser.
- Fast Execution, reduces the execution time as test cases are executed parallelly.



Start

Selenium Grid ?

The architecture of Selenium Grid:

In Grid one of the systems is created as Hub. Hub works as a central point controlling all the nodes. Nodes are an actual machine on which execution is done.

Selenium IDE and Selenium RC are obsolete products now, so we won’t spend much time on them. In the upcoming tutorials, we will focus more on Selenium WebDriver and Selenium Grid.

What is WebDriver ?

The WebDriver proves to be better than Selenium IDE and Selenium RC in many aspects. It implements a more modern and stable approach in automating the browser's actions. WebDriver, unlike Selenium RC, does not rely on JavaScript for Selenium Automation Testing. It controls the browser by directly communicating with it.

The supported languages are the same as those in Selenium RC.

- Java
- C#
- PHP
- Python
- Perl
- Ruby

What is Selenium Grid? ?

Selenium Grid is a tool used together with Selenium RC to run parallel tests across different machines and different browsers all at the same time. Parallel execution means running multiple tests at once.

What is Selenium Grid? ?

Features:

- Enables simultaneous running of tests in multiple browsers and environments.
- Saves time enormously.
- Utilizes the hub-and-nodes concept. The hub acts as a central source of Selenium commands to each node connected to it.

Selenium Browser & Environment Support

	Selenium IDE	WebDriver
Browser Support	Mozilla Firefox and Mircrosoft Edge	Google Chrome 12+ Firefox Internet Explorer 7+ and Edge Safari, HtmlUnit and PhantomUnit
Operating System *Note: Opera Driver no longer works*	Windows, Mac OS X, Linux	All operating systems where the browsers above can run.

How to Choose the Right Selenium Tool for Your Need?

Start

Selenium IDE ?

- To learn about concepts on automated testing and Selenium, including:
  - Selenese commands such as type, open, clickAndWait, assert, verify, etc.
  - Locators such as id, name, xpath, css selector, etc.
  - Executing customized JavaScript code using runScript
  - Exporting test cases in various formats.
- To create tests with little or no prior knowledge in programming.
- To create simple test cases and test suites that you can export later to RC or WebDriver.
- To test a web application against Firefox and Microsoft Edge only.

Selenium RC ?

- To design a test using a more expressive language than Selenese
- To run your test against different browsers (except HtmlUnit) on different operating systems.
- To deploy your tests across multiple environments using Selenium Grid.
- To test your application against a new browser that supports JavaScript.
- To test web applications with complex AJAX-based scenarios.

WebDriver ?

- To use a certain programming language in designing your test case.
- To test applications that are rich in AJAX-based functionalities.
- To execute tests on the HtmlUnit browser.
- To create customized test results.

Selenium Grid ?

- To run your Selenium RC scripts in multiple browsers and operating systems simultaneously.
- To run a huge test suite, that needs to complete in the soonest time possible.

types of

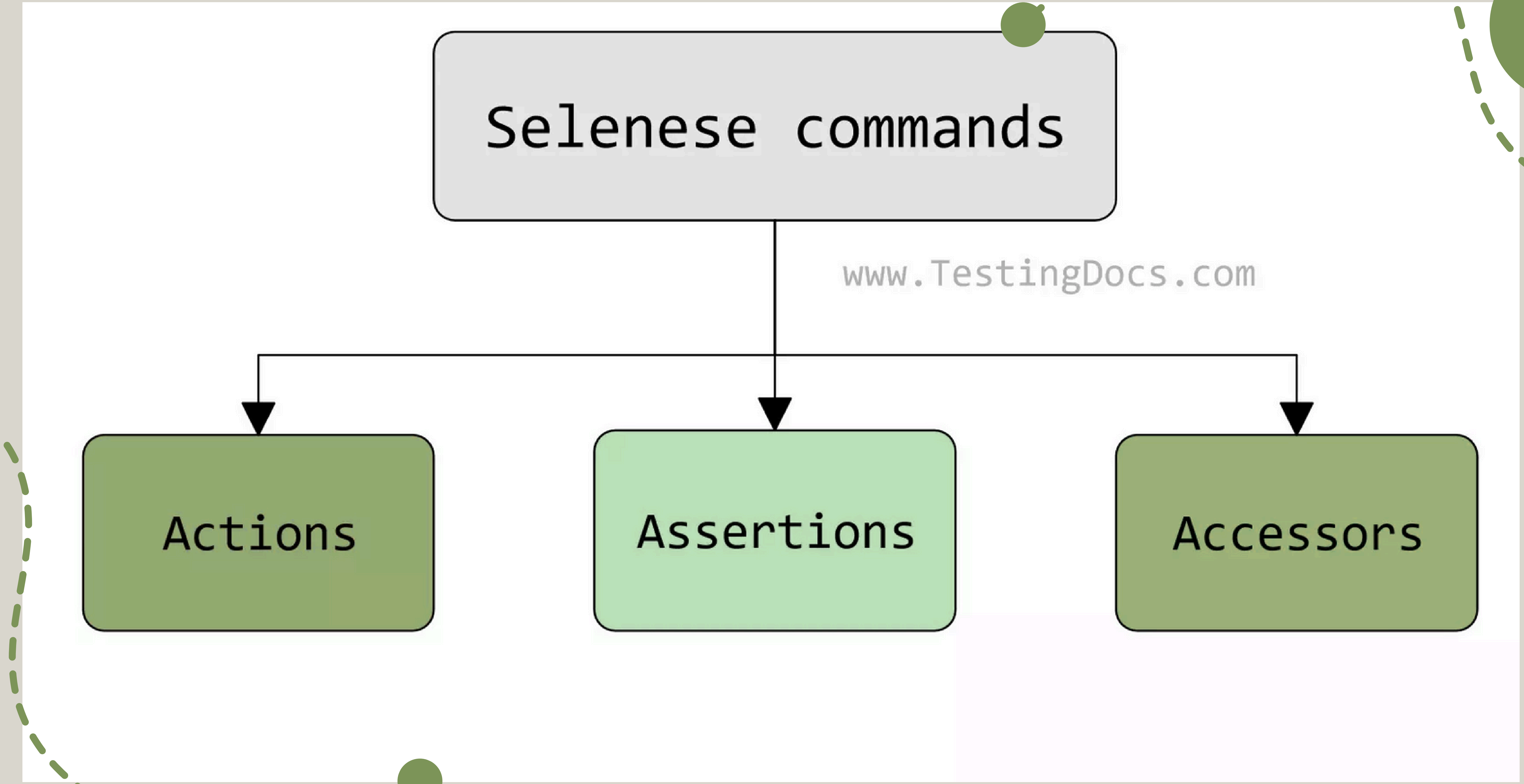
SELENIUM COMMANDS

Selenium IDE- Commands (Selenese) ?

Selenium commands, also known as “Selenese“ are the set of commands used in Selenium IDE that run your tests. Using selenese, one can perform activities like:

- Testing the existence of UI elements based on their HTML tags.
- Test for specific content.
- Test for broken links.
- Testing input fields, selection list options, submitting forms and table data among other things.
- Testing of window size, mouse options, alerts, Ajax functionality, pop-up windows, event handling and many other web application features.

A sequence of Selenium commands (Selenese) together is known as test script.



Selenese commands

Actions

Assertions

Accessors

Start

Actions ?

Actions are the selenium commands that generally manipulate the state of the application. Execution of Actions generates events like click this link, select that option, type this box, etc. If an Action fails, or has a bug, the execution of current test is stopped.

Some of the most commonly used Actions commands include:

Command/Syntax	Description
open (url)	It launches the desired URL in the specified browser and it accepts both relative and absolute URLs.
type (locator,value)	It sets the value of an input field, similar to user typing action.
typeKeys (locator,value)	This command simulates keystroke events on the specified element.
click (locator)	This command enables clicks on a link, button, checkbox or radio button.
clickAt (locator,coordString)	This command enables clicks on an element with the help of locator and co-ordinates

Command/Syntax	Description
doubleClick (locator)	This command enables double clicks on a webelement based on the specified element.
focus (locator)	It moves the focus to the specified element
highlight (locator)	It changes the background color of the specified element to yellow to highlight is useful for debugging purposes.
close()	This command simulates the user clicking the "close" button in the title bar of a popup window or tab.
store (expression,variableName)	This command specifies the name of a variable in which the result is to be stored and expression is the value to store
waitForCondition (script,timeout)	This command executes the specified JavaScript snippet repeatedly until it evaluates to "true".

Accessors ?

Accessors are the selenium commands that examine the state of the application and store the results in variables. They are also used to automatically generate Assertions.

Some of the most commonly used Accessors commands include:

Command/Syntax	Description
storeElementIndex (locator, variableName)	This command gets the relative index of an element to its parent (starting from 0).
storeBodyText (variableName)	This command gets the entire text of the page.
storeAllButtons (variableName)	It returns the IDs of all buttons on the page.
storeAllFields (variableName)	It returns the IDs of all input fields on the page.
storeAllLinks (variableName)	It returns the IDs of all links on the page.

Command/Syntax	Description
storeTitle (variableName)	This command gets the title of the current page.
storeText (locator, variableName)	This command gets the text of an element..
storeValue (locator,variableName)	This command gets the (whitespace-trimmed) value of an input field.
storeTable (tableCellAddress, variableName)	This command gets the text from a cell of a table.
storeLocation (variableName)	This command gets the absolute URL of the current page.

Assertions

Assertions are the commands that enable testers to verify the state of the application. Assertions are generally used in three modes assert, verify and waitfor.

Some of the most commonly used Assertions commands are:

Command/Syntax	Description
verifySelected(selectLocator, optionLocator)	This command verifies that the selected option of a drop-down satisfies the optionSpecifier.
verifyAlert (pattern)	This command verifies the alert text; used with accessorstoreAlert.
verifyAllButtons (pattern)	This command verifies the button which is used withaccessorstoreAllButtons.
verifyAllLinks (pattern)	This command verifies all links; used with the accessorstoreAllLinks.
verifyBodyText(pattern)	This command verifies the body text; used with the accessorstoreBodyText.

Command/Syntax	Description
verifyAttribute(attributeLocator, pattern)	This command verifies an attribute of an element; used with the accessorstoreAttribute.
waitForErrorOnNext (message)	This command enables Waits for error; used with the accessorassertErrorOnNext.
waitForAlert (pattern)	This command enables waits for the alert; used with the accessorstoreAlert.
verifyAllWindowIds (pattern)	This command verifies the window id; used with the accessorstoreAllWindowIds.