Selenium

Locators :

1. What are locators in the context of software testing?

Locators are mechanisms used in software testing to identify and locate specific elements on a web page or mobile application. They allow testers to interact with these elements for validation, manipulation, or automation purposes.

**What are the characteristics of a good Locator?**

Some characteristics of a good locator are:

1. **Unique** - Uniquely identifies an element in the web page
2. **Robust and Reliable** - Any changes in the surrounding elements must not impact the element. Element should be consistently identified everytime without any glitches
3. **Short and Descriptive** - Easier to read and including only the important attributes in the locator is essential
4. **Fast** - Time taken to locate an element differs across each locator. This is evidently noticeable when automating large web pages. Choosing a fast locator makes your automation script faster

#### What are the Types of Locators?

* **Attribute Based Locators** identify the elements based on their tags and attributes
* **Structure Based Locators** identify elements based on the DOM structure. Structure based Locators are dependent on the other elements and their order in the DOM

#### Order of Preference in Locator selection

ID (Attribute Based)

NAME(Attribute Based)

CLASS NAME(Attribute Based)

TAG TYPE(Attribute Based)

CSS SELECOR(Structure Based)

XPATH(Structured Based)

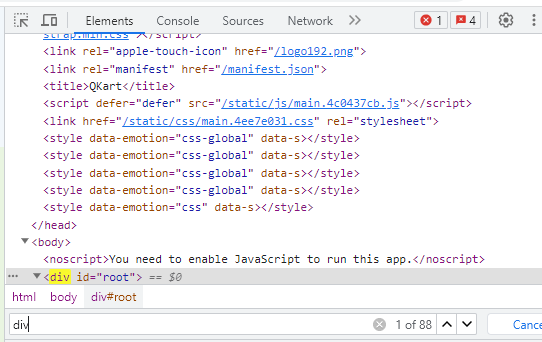
The following options are available exclusive to <a> elements

LINK TEXT(Attribute Based)

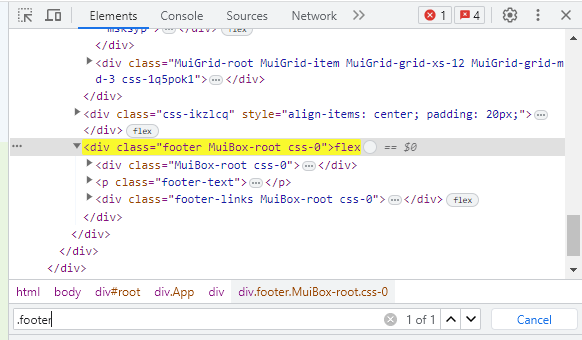
PARTIAL LINK TEXT(Attribute Based)

CSS SELECTORS Examples:

1 Select all **<div>** elements on the page:



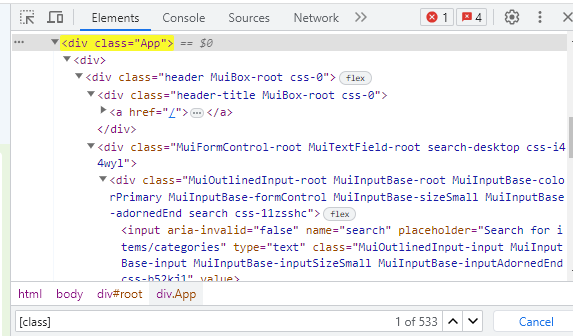
2. Select elements with the class "footer":



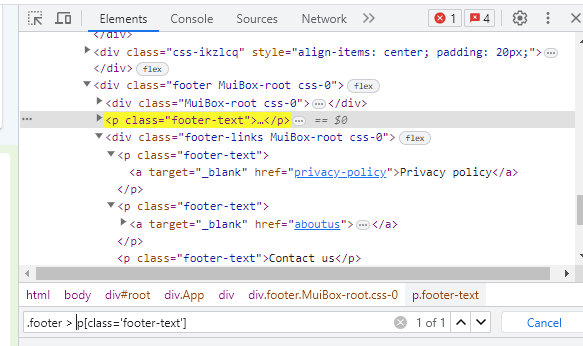
3. Select the element with the ID "my-id":

#my-id

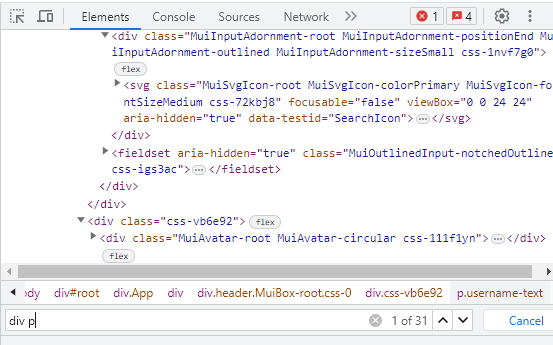
4.Select all elements with the attribute "data-type":



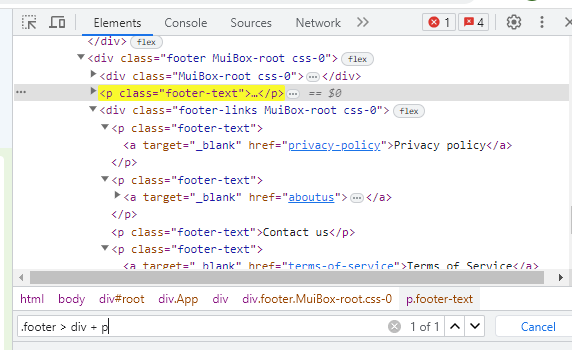
5. Select elements with the attribute "data-type" and value "example":



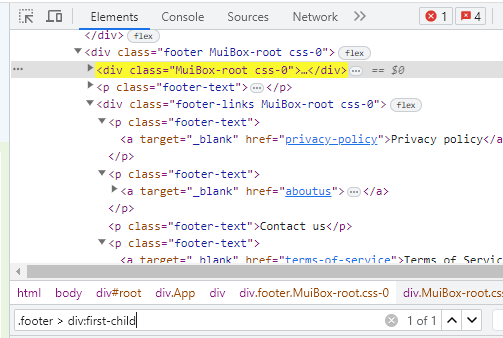
6.Select all **<p>** elements inside a **<div>**:



7. Select the **<p>** element that directly follows an **<div>**:



8. Select the first child element of a parent element:



Select even-numbered elements inside a parent element:



XPATH:

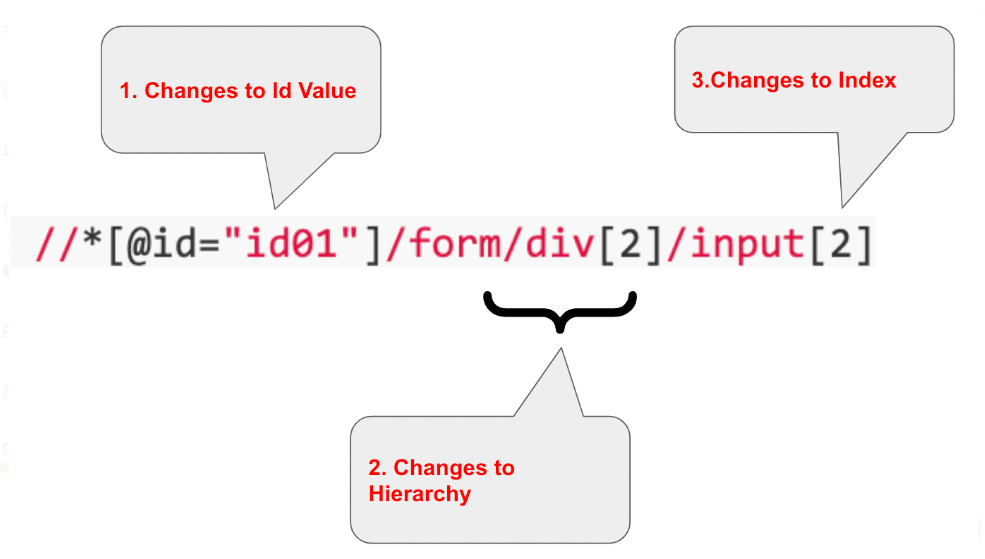
**Absolute XPath**: Starts with a single forward slash (**/**) and specifies the complete path from the root element to the target element. For example:

/html/body/div[1]/form/input[3]

**Relative XPath**: Does not begin with a forward slash and provides a more flexible and concise way to locate elements relative to their context. For example:

//input[@name='username']

**What could possibly go wrong in the above locator?**



#### How do I make my script resilient against these changes?

Use an attribute based locator instead of structure based locators. In the above case, name = psw seems to be one of the best ways

#### What are Objects that cannot be located by Selenium?

### 1. Multi Factor Authentication

### 2.Browser Menu

### 3.Virtual Keyboards

### 4.Capcha

#### What are chained Locators?



//We find the outer <div> first

WebElement outer\_div\_user\_name = driver.findElement(By.id("Username"));

//We find the input element under the previously identified <div>

WebElement username\_input = outer\_div\_user\_name.findElement(By.tagName("input"));

**In Another way**

//Chained Locator: find Outer Div and Input Tag inside it

WebElement username\_txtbox = driver.findElement(new ByChained( By.id("Username"), By.tagName("input")));

//Chained Locator: find Outer Div and Input Tag inside it

WebElement dob\_txtbox = driver.findElement(new ByChained( By.id("dob"), By.tagName("input")));

//Chained Locator: find Outer Div and Input Tag inside it

WebElement password\_txtbox = driver.findElement(new ByChained( By.id("pwd"), By.tagName("input")));

What is the difference between XPath and CSS locators? When would you choose one over the other?

1. **Syntax**: XPath uses a path-like syntax to traverse the XML/HTML document structure, while CSS locators use a selector syntax similar to how CSS styles are applied to elements.
2. **Targeting Capabilities**: XPath has more powerful targeting capabilities as it allows you to navigate the entire document structure, select elements based on attributes, text content, position, and complex relationships. CSS locators are primarily focused on targeting elements based on classes, IDs, tag names, and simple relationships.
3. **Readability**: CSS locators are generally more concise and easier to read, especially when selecting elements based on classes or IDs. XPath expressions can become complex and lengthy, especially when targeting specific attributes or performing advanced traversals.
4. **Browser Support**: CSS locators are widely supported by browsers and testing frameworks, making them a reliable choice for element identification. XPath is also well-supported but may have minor differences in implementation across different browsers.
5. **Performance**: CSS locators are generally faster than XPath in terms of performance due to their simpler syntax and optimized browser implementations. However, the performance difference may vary depending on the complexity of the selector and the specific browser.

When to choose XPath over CSS locators:

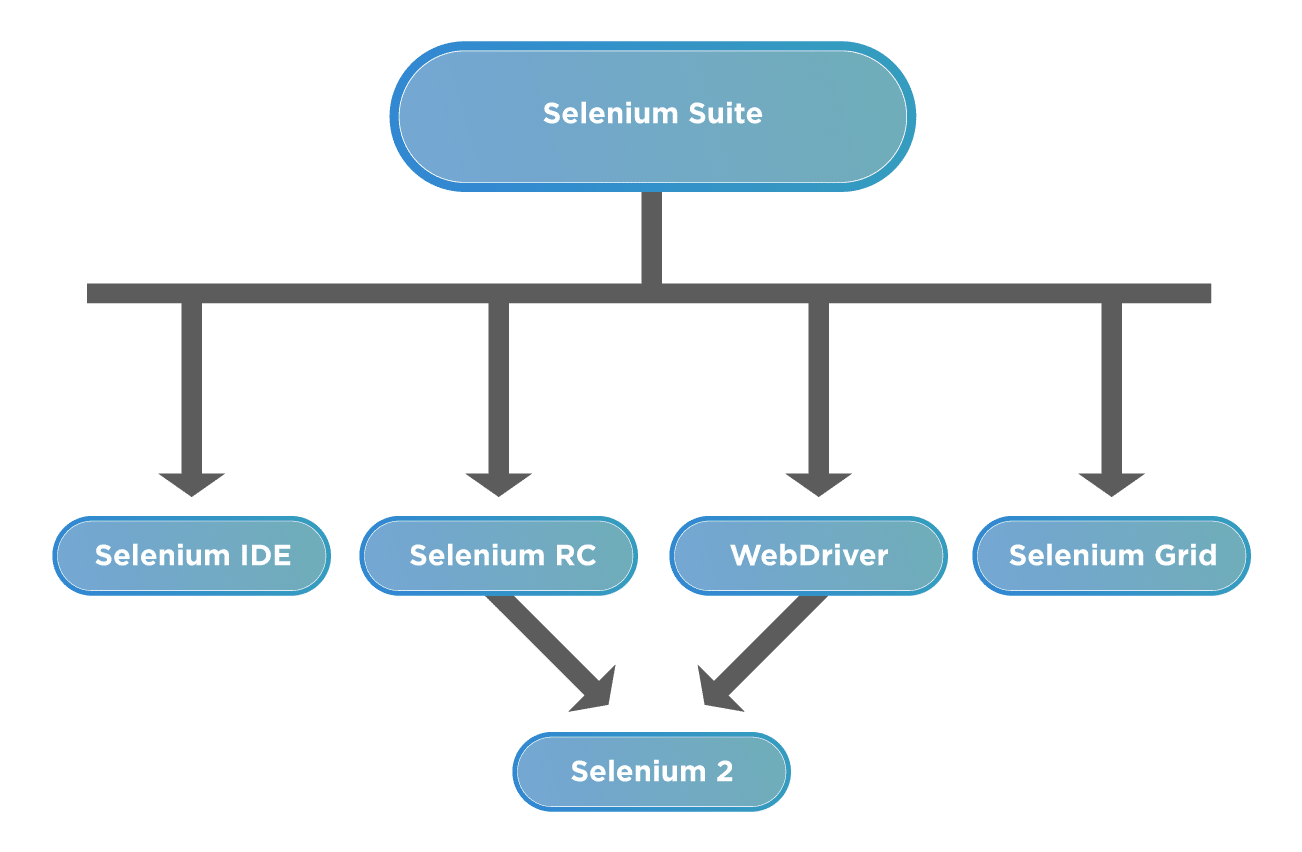
* When targeting elements based on complex relationships or traversing the document structure.
* When there is no suitable attribute, class, or ID available for targeting.
* When targeting elements based on text content, such as exact matches or partial matches.
* When working with XML documents or HTML documents with inconsistent or non-standard structures.

When to choose CSS locators over XPath:

* When targeting elements based on classes, IDs, or tag names, which are efficiently handled by CSS locators.
* When readability and conciseness are important, especially for simpler element targeting.
* When working with HTML documents with a well-defined structure that can be easily navigated using CSS selectors.

## What is Selenium WebDriver?

***Selenium WebDriver*** is a set of open-source [***APIs,***](https://en.wikipedia.org/wiki/Application_programming_interface) which provided to interact with any of the web-browsers and then, in-turn to automate the user actions with that browser. Selenium is not an independent tool; rather, it is a collection of tools that make the Selenium suite, which was created when two projects ***Selenium RC and WebDriver were merged.***



WebDriver was integrated with Selenium RC to overcome a few of the limitations of Selenium RC and has now become the de-facto for Web automation. You must be aware that after Selenium 2, there are new versions have been released. By now in Jan 2021 they have reached ***Selenium 4.***

### *****Why Selenium WebDriver is popular?*****

***Multi-Browser Compatibility***

***Multi-Language Support***

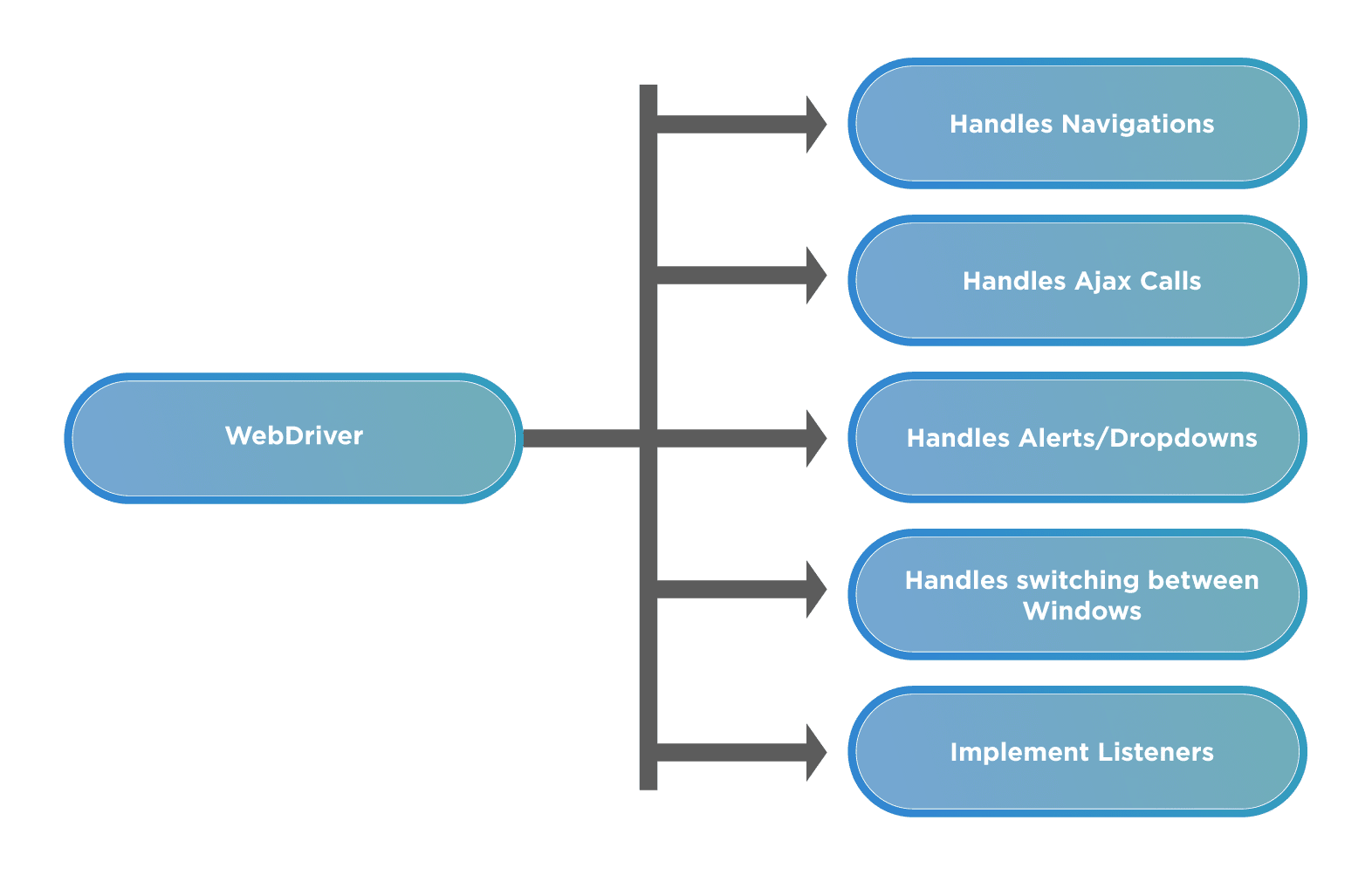
***Faster Execution*** : WebDriver directs communications with browsers using a defined protocol (JSON Wire),



***Locating Web Elements***

***Handling dynamic web elements***

***Handling Waiting for Elements***



### *****What are the drawbacks of Selenium WebDriver?*****

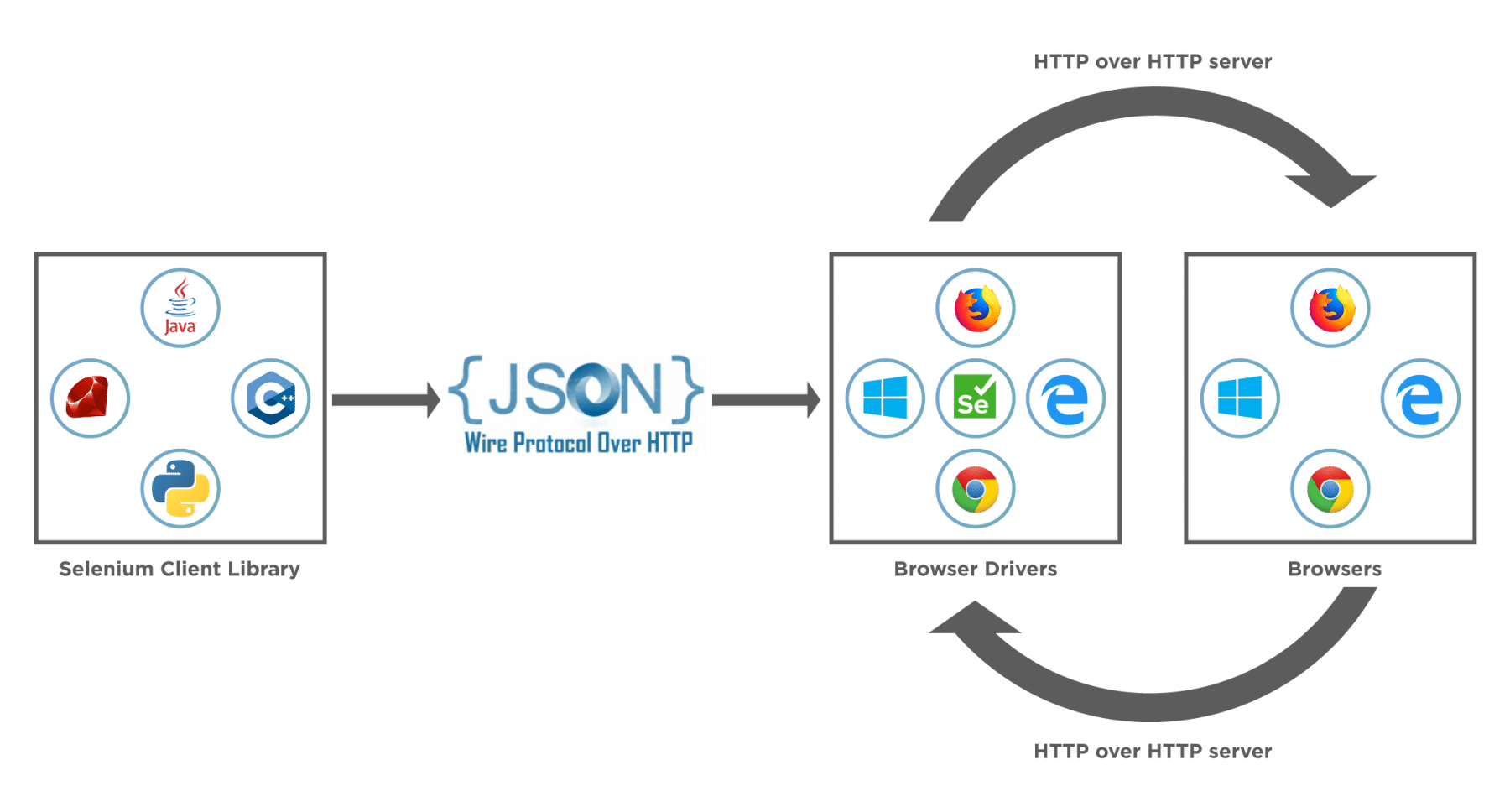
***Requires Programming Knowledge and Expertise***

***No Support for Desktop Applications.***

***No Customer Support***

***Managing Browser-Selenium Dependencies***

## Selenium WebDriver Architecture



***Is Selenium WebDriver a framework?***

***Is not a framework it is a tool and it selenium webdriver is a part of selenium framework – set of components like selenium IDE,Selenium RC,Selenium webdriver,Selenium grid***

* *Selenium WebDriver is a set of APIs, which makes the interaction and actions on the browser very easy and quick.*
* *Selenium WebDriver provides quite a few unique features, such as it can automate dynamic web pages. Additionally, it can automate all the web applications, no matter in which programming language we use to develop them.*