**Selenium is a browser Automation tool,**

**Selenium comes with 4 versions**

**1. Selenium IDE**

**2. Selenium RC**

**3. Selenium WebDriver**

**4. Selenium GRID**

**Jason Huggins introduced Selenium when he was working in ThoughtWorks, in the year 2004 and Selenium IDE is introduced to marked in the year 2006 as a Addon with FF browser.**

**Features of Selenium :**

**------------------------------**

**1. Open source freely available software**

**2. Selenium is the only tool which supports multiple programming languages**

**java, C#, python, Ruby, JavaScript, Kthlon**

**3. Supports almost all popular Browsers**

**chrome, FF, edge, opera, safari**

**4. Supports multiple platforms - HW & SW**

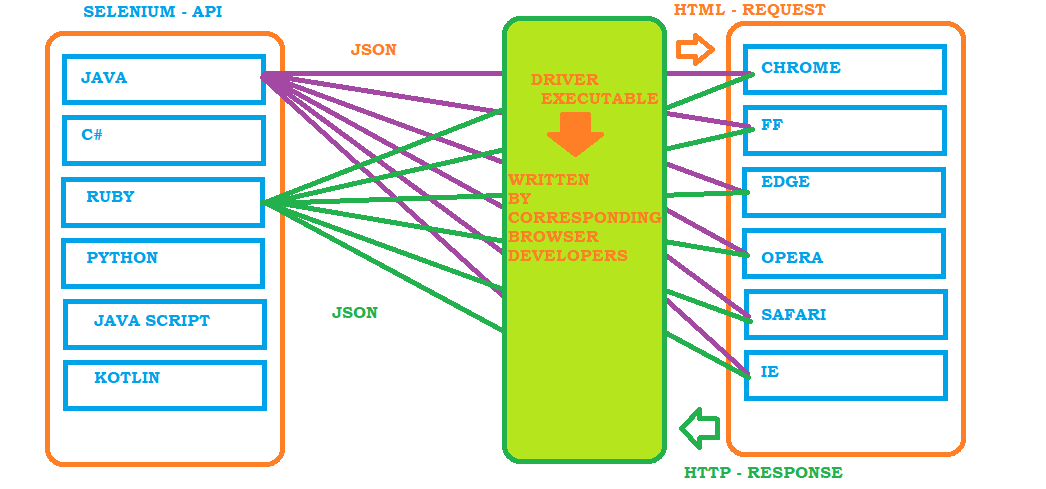
**5. Selenium Supports distributed Execution - Using Selenium GRID**

**6. Parallel Execution of Tests are supported in Selenium**

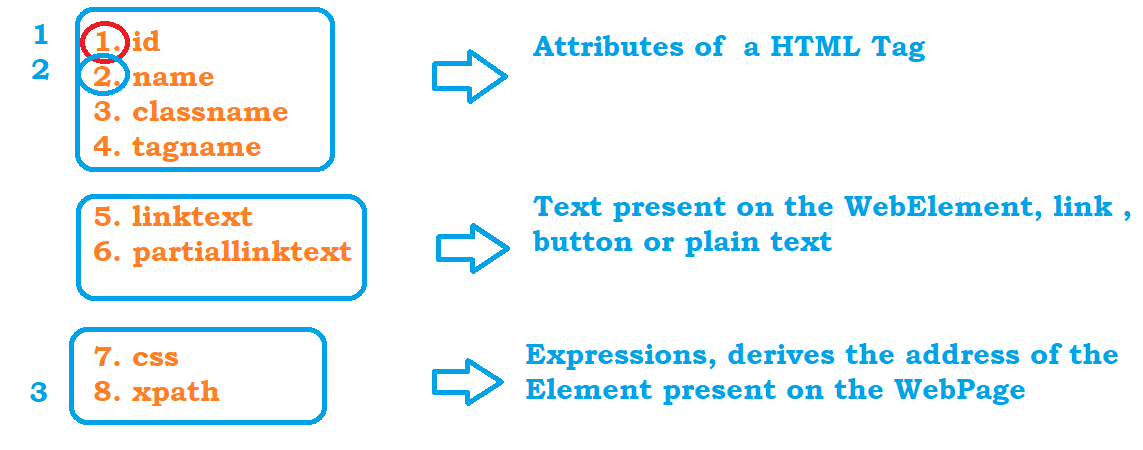
**7. Integrate any 3rd Party tools easily**

**8. No Dedicated Machine is needed to execute the tests**

ARCHITECTURE



Locators in Selenium



**CSS Selector :**

**---------------------**

**1. htmltag[attribute='value']**

**input[name='username']**

**2. htmltag#idValue OR #idValue**

**a#loginButton OR #loginButton**

**3. htmltag.classAttributeValue OR .classAttibute Value**

**div.productNameContainer OR .productNameContainer**

***Traversing from parent to Child***

***--------------------------------------------------------------***

**parent\_css\_expression > child tag or expression**

**td#loginButtonContainer > a**

Xpath

# Basic xpath

//htmlTag[@attibute=’value’]

# Xpath using functions

* 1. text() 🡪 **to find the element which has text, we have to pass the exact text**

***Syntax :***

//htmltag[text()=’value’]

***Example*** :

**//div[text()='Login ']**

**//label[text()='Keep me logged in']**

* 1. contains(arg1, arg2) 🡪 can be used to find any type of element present on the WebPage
     1. arg1 – attribute or text() function
     2. arg2 – partial value of first argument

***Syntax :***

//htmltag[contains(arg1, ’value’)]

***Example*** :

**//div[contains(text(),'Login')] – Actitime login**

**//label[contains(text(),'Keep')] – Actitime login**

**//td[contains(@class,'secondLine')] – Actitime Home page**

* 1. starts-with(arg1,arg2) 🡪 can be used to find any type of element present on the WebPage
     1. arg1 – attribute or text() function
     2. arg2 – partial value of first argument

***Syntax :***

//htmltag[starts-with(arg1, ’value’)]

***Example*** :

**Google Search for IBM :**

**//h3[starts-with(text(),'IBM')]**

# Xpath Using Logical operators – When one property is not sufficient to find the elements then we can use more than one property to find the elements using ,

1. **and :** element will be located when both the properties are present in the HTML Tag

***Syntax :***

//htmltag[@attribute1=’value1’ and @attribute2=’value2’]

***Example*** :

//input[@class='textField' and @id='username']

1. **or:** element will be located when any one property is present in the HTML Tag

***Syntax :***

//htmltag[@attribute1=’value1’ or @attribute2=’value2’]

***Example*** :

**//input[@class='textField pwdfield' or @class='textField']**

1. **not –** negation operation, if we want to ignore any property then we can go with not

***Syntax :***

//htmltag[@attribute1=’value1’ or not ( @attribute2=’value2’)]

***Example*** :

//input[@class='textField pwdfield' and not (@class='textField')]

**//input[@class='textField pwdfield' and not (@class='textField')]**

# Traversing from parent to Child

1. Write an expression from parent HTML Tag

2. If child is an immediate child then use single slash /

Ex : Actitime login Page : //a[@id='loginButton']/div

3. If child is not an immediate child then use double slash //

Ex : Actitime login Page : //td[@id='loginButtonContainer']//div[text()='Login ']

# Traversing from Child to Parent

Whenever we have dependent and independent elements in our application then we have to traverse from child to parent

Step1 : Write and xpath for independent Element

**//th[text()='Directed by']**

Step2 : Travers to its parent until both dependent and independent elements highlighted

//**tr[**th[text()='Directed by']**]**

Step3 : Travers to dependent element

//tr[th[text()='Directed by']]**//a**

NOTE : to traverse from child to parent, put the entire expression inside the square bracket and specify the parent html tag

//div[div[div[div[div[span[contains(text(),'Goa')]]]]]]//span[@id='price\_detail']

Xpath using Axes Functions

update image

Selenium 3.0 Setup

1. Created a Maven Project
2. Update compiler from 1.5 to what is installed on the machine
3. Update JRE to use JRE Present in JDK location
4. Update pom.xml with dependencies
5. Write your first Test

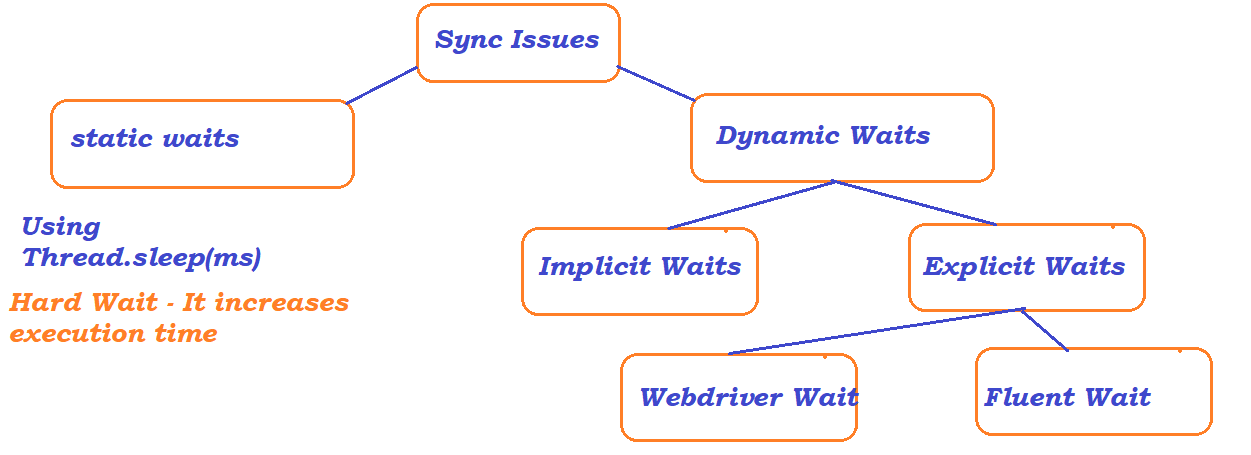
Basic Operations

<https://formy-project.herokuapp.com/form>

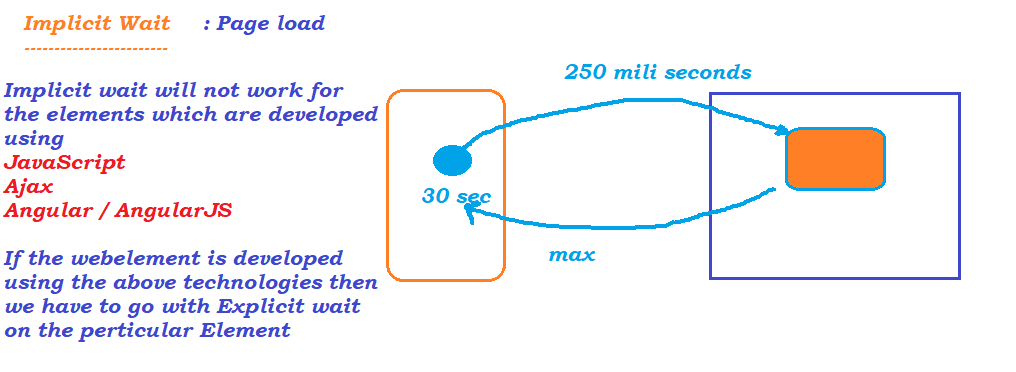
# Killing any Process from cmd prompt or Java program

taskkill /F /IM chromedriver.exe

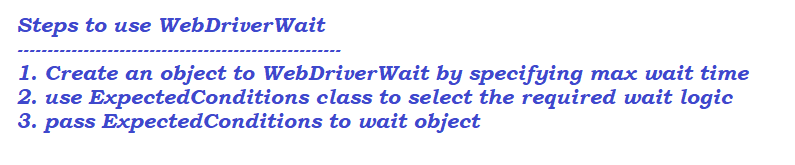
Handling Sync issues



Implicit wait

****

Explicit Wait :

****

Fluent Wait :

1. When we want to have our own wait logic
2. if we want to change the polling time
3. if we want to ignore any exceptions during wait
4. to log the wait messages

Steps :

1. Create an object to fluent wait by passing
   1. what is the web element
   2. what is the polling time
   3. what is the max wait
   4. what are the exceptions to ignore
2. override apply method present inside function interface
3. wait.until(obj to function interface)

Browser Operations :

* click on back button
* click on forward button
* click on refresh button
* click on maximize button
* print current url
* full screen

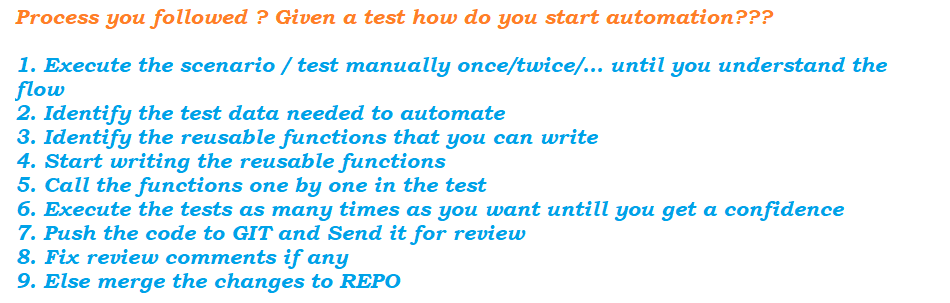
# AutoSuggestions

# WebDriverManager

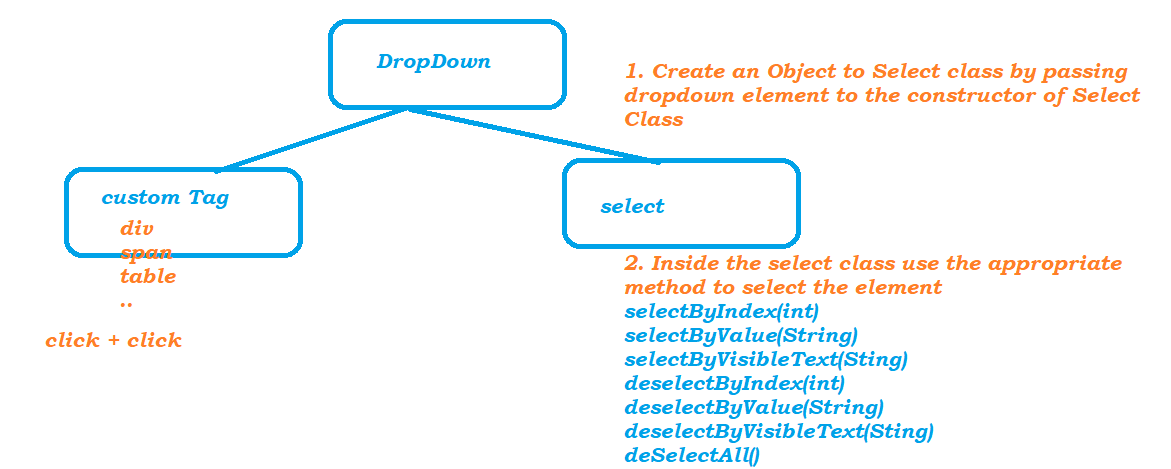
Third party class/ library which will be used to download the driver executable

First update the pom.xml with the dependencies

1. ToolTip



Dropdown

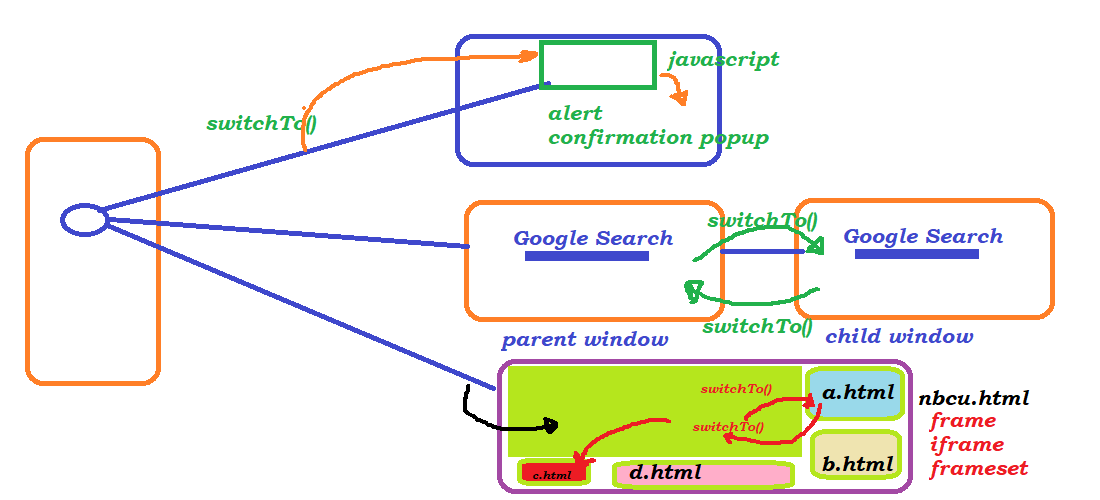


Actions in Selenium

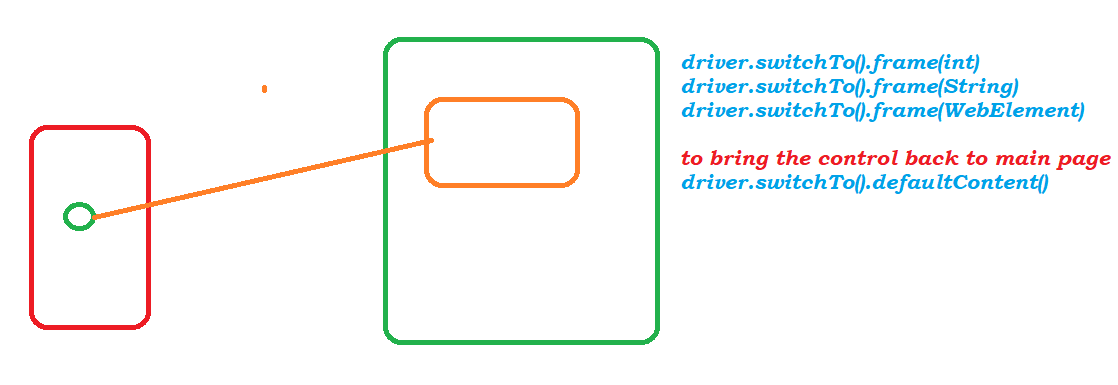
Actions in selenium are used to perform exact keyboard and mouse movement operations.

Whatever the keyboard and mouse movement operations we do is only on the browser

SwitchTo



Handling Frames



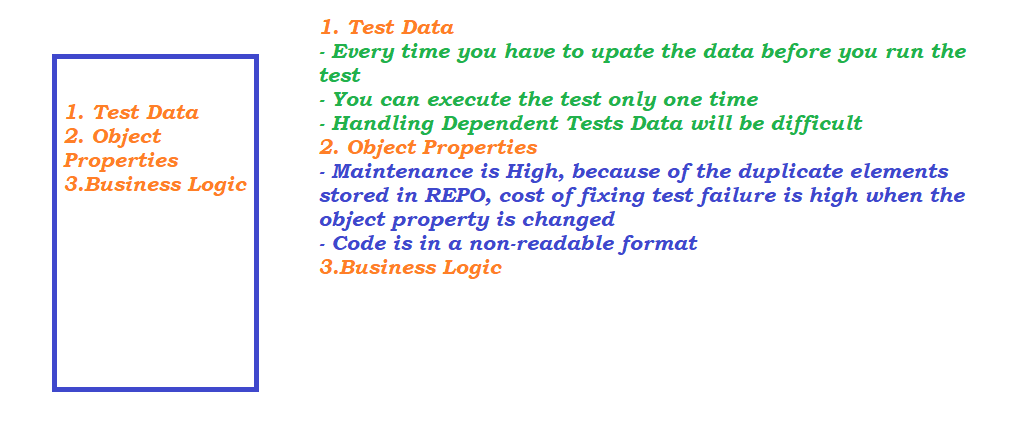
Popups

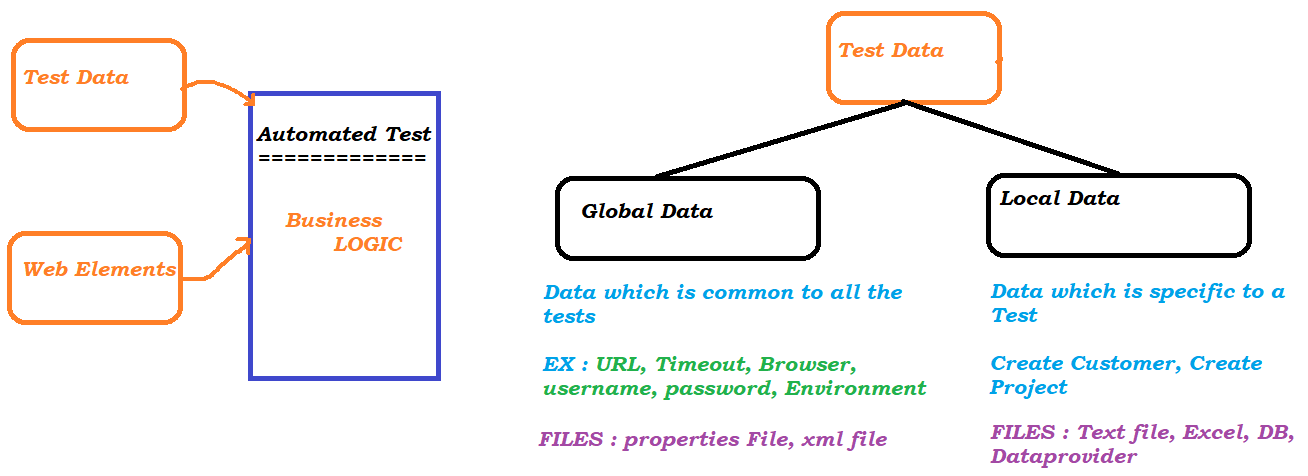
* alerts -- switchTo()
* confirmation popup -- switchTo()
* Hidden division popup Normal Selenium Code
* On Page load popup -Windows PopUPs
* File Download -Windows PopUPs
* File Upload -Windows PopUPs

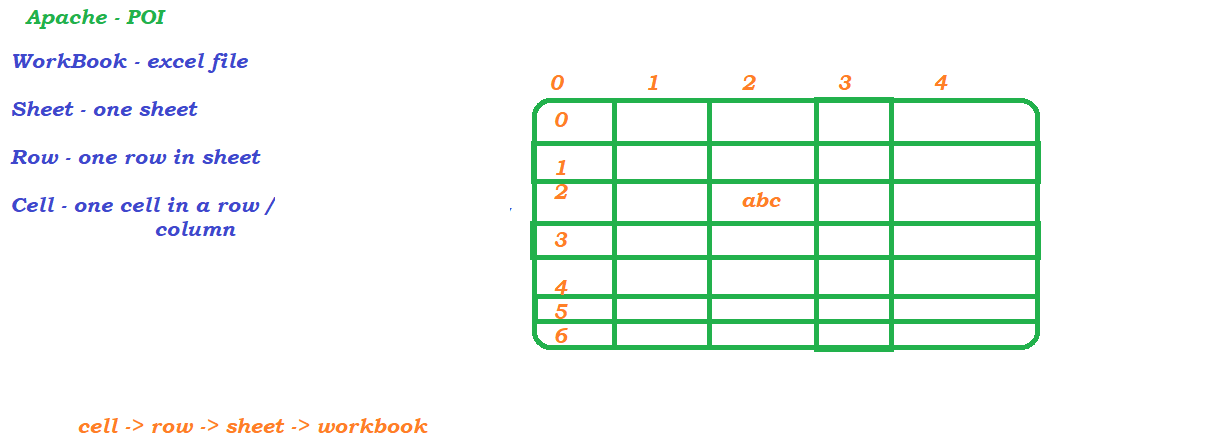
Windows PopUp Can be Automated by any of the 3rd party tools

* **AutoIT – Windows Application Automation Tool**
  + Install AutoIT
  + https://www.autoitscript.com/site/autoit/downloads/
* **Siquli – Image based Automation**
* **Roboat Class**

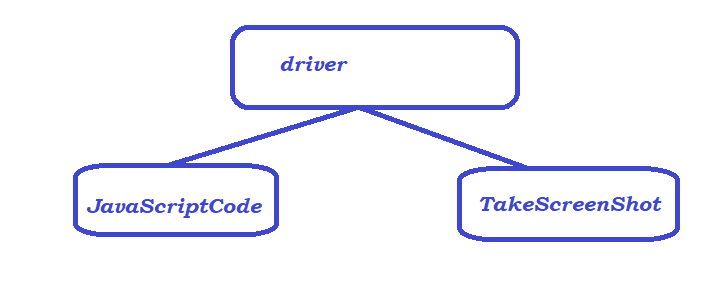
Data driven Testing







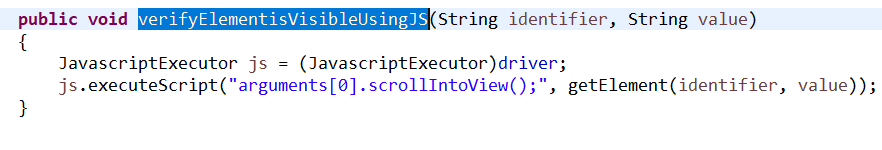
Executing JS code from Selenium

****

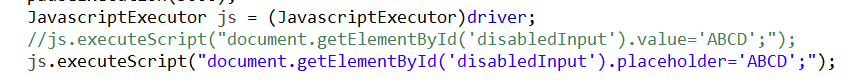
**Scroll for 1000 pixel**

****

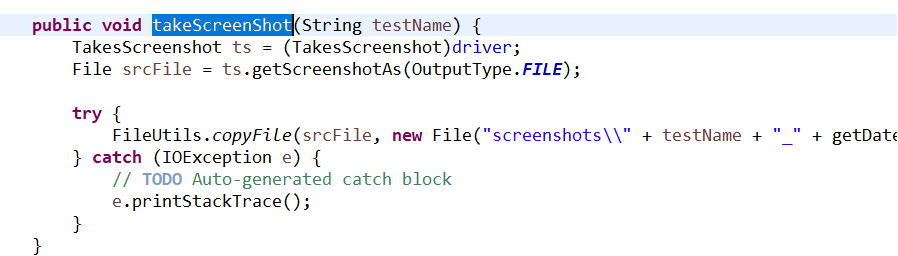
**Scroll until Element is visible**

****

**Type operation using JS :**

****

**Capturing Screen Shots on test Failure**



Framework

1. set of Rules or Guidelines or Procedure which describes

a. How to write TestCases

b. What are all the methods / functions to use while Automating the TestCases

c. Format of the Automating the TestCases

2. Contains information about what kind of Reports will be generated post Execution

3. Should we re-execute the failed tests

## Types of Framework :

1. FUNCTION DRIVEN AUTOMATION FRAMEWORK

2. DATA DRIVEN AUTOMATION FRAMEWORK

3. KEYWORD DRIVEN AUTOMATION FRAMEWORK

4. HYBRID DRIVEN AUTOMATION FRAMEWORK

5. TESTNG

6. POM

TestNG

* + testing.org
  + Annotations
  + Order of Execution
  + Changing the order of execution
  + Reading Global Data from xml file
  + Reading local data from Dataprovider
  + Parallel Execution
  + Groups
  + include and exclude
  + invocationCount
  + Listeners
    - ITestResult

POM

POM

Reports

-ExtentReports

* How to create
* How to add logs
* How to Attach SS on Failure

Selenium GRID

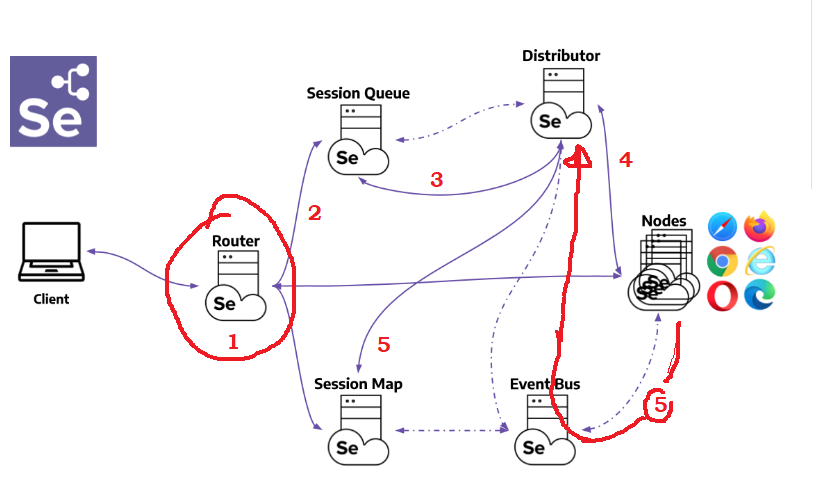
To achieve Distributed Execution we have to configure our test with Selenium GRID

In Selenium 4.0 GRID can be configured using

1. Standalone : Both Hub and Node in a same Machine
2. Hub and Node : To configure Hub and Node on a different machine we can go with Hub and Node mode.
3. Distributed : To configure Router, Session Map, Session Queue, Distributor, Event Bus and Node on a different machine

Distributed Mode is only for complex Test suites

Architecture



### Standalone :

1. Create a Test to execute test in RemoteWebDriver



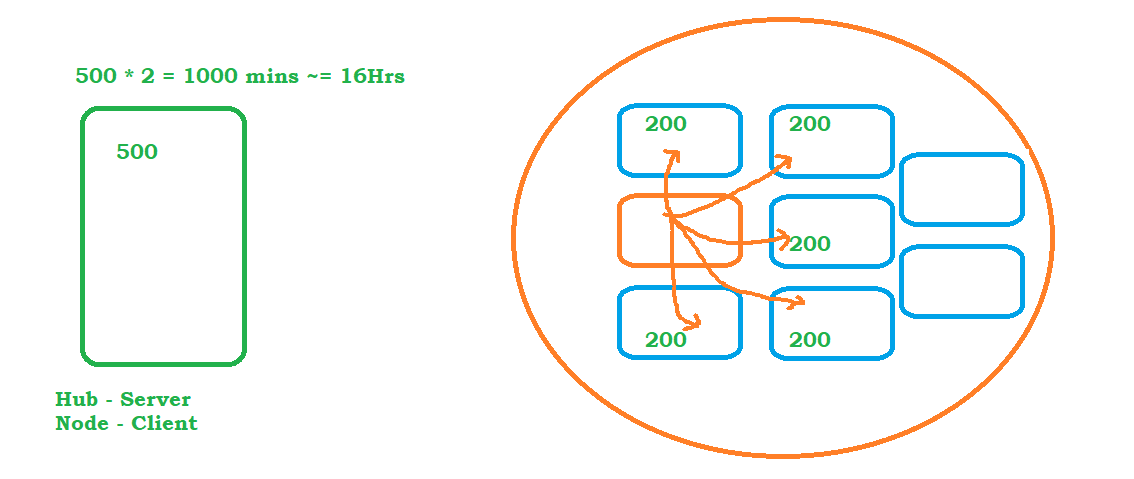
2. Start the Grid in standalone mode , to do that execute the download grid jar with standalone opetion

D:\EclipseWS\grid>java -jar selenium-server-4.18.1.jar **standalone**

Hub and Node

Refer : https://www.selenium.dev/documentation/grid/getting\_started/#hub-and-node

To register node from different machine 



Selenium 4.0

# Selenium Manager :

No need to download driver Executable from selenium 4.8.0 onwards

# Relative Locators

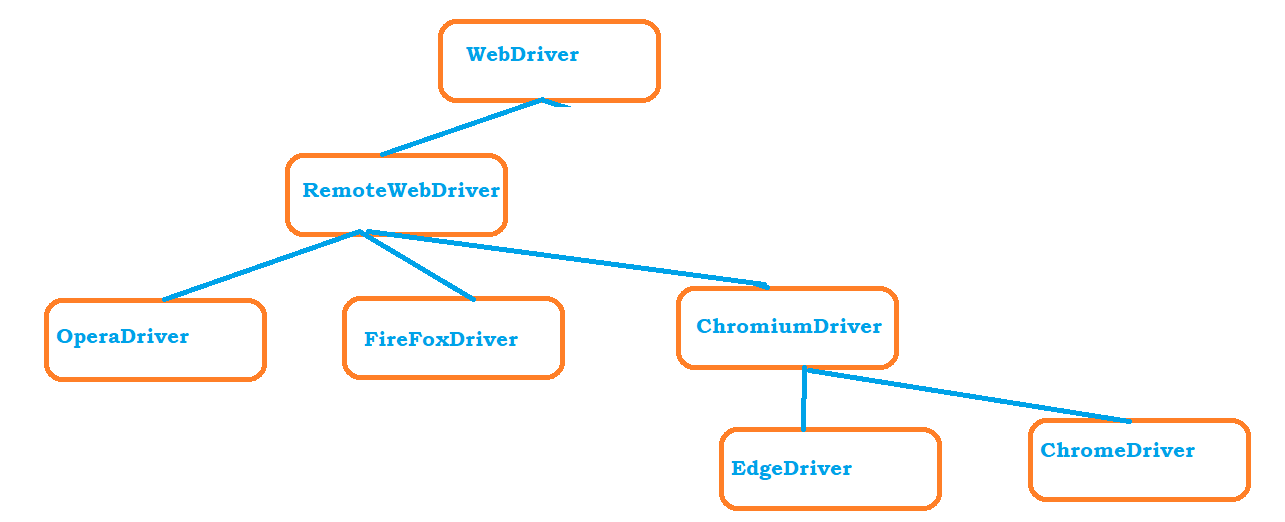
Relative Locators below –

* below
* toRightOf
* below and toLeftOf
* toRightOf and toLeftOf
* below
* below and above
* near - Search button (Click)

# W3C Standarts

* communication is using W3C Protocol
* before selenium 3.8 it was json wired protocol
* above 3.8 version of selenium Architecture is changed to W3C
* Advantages
  + stability
  + easy communication
  + extra functionality – capturing browser logs, errors etc
* No Changes to the way we write the code
* No Additional changes in Appium Software

# ChromiumDriver Introduced in Selenium 4.0

****

# FindsBy Annotation is completely Removed in Selenium 4.0

# Selenium IDE - By Applitools re-written completely

* conditional and looping statements are allowed in IDE

Reference : <https://www.selenium.dev/selenium-ide/docs/en/introduction/control-flow>

* Executing IDE tests from command line is possible from node package manager : selenium-side-runner
* steps : https://www.selenium.dev/selenium-ide/docs/en/introduction/command-line-runner

-Log4J

JAVA-8 Features

Selenium 4.0 Features

BDD

WebServices REST Automation POSTMAN and REST APIs

Jenkins

GIT