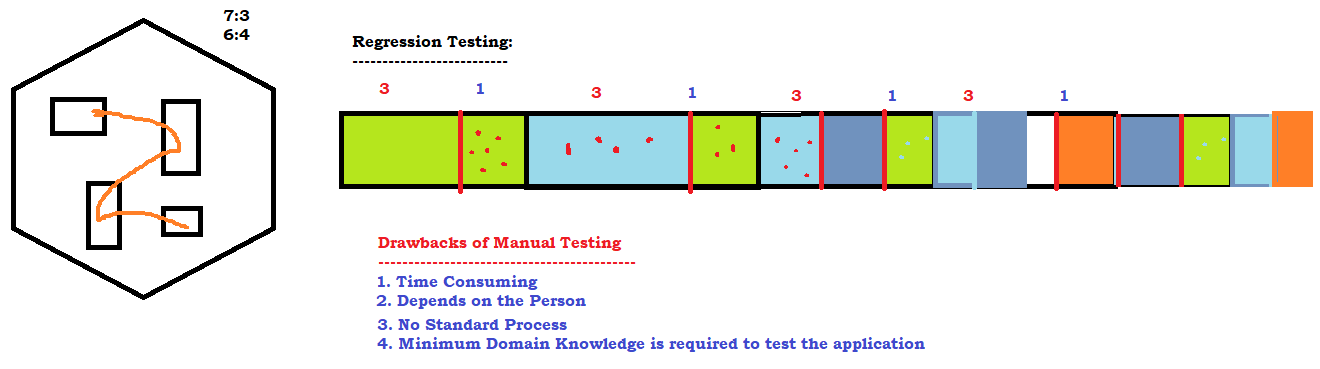
**Selenium Automation**

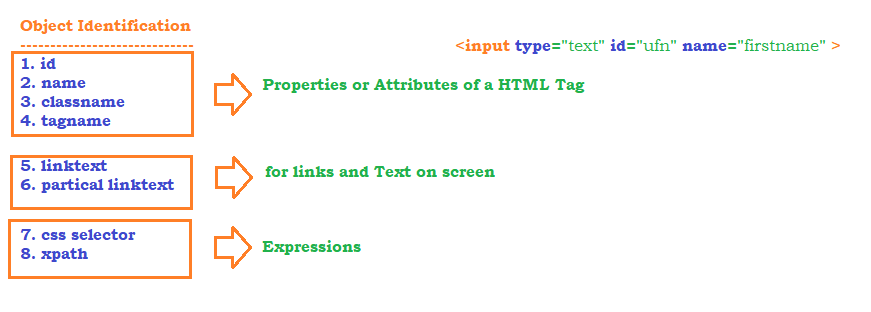
1. **Some intro to Testing**

* **What is Software Testing**
* **What are the different type of testing we do ???**
* **What are the drawbacks of manual testing**



* **Why Automation is needed ?**
* **What are the Tools Available in Market ?**
  + **UFT**
  + **RFT**
  + **Selenium**
  + **Sahi**
  + **Eggplant**
  + **Auto IT**
  + **siquli .. .. .. .. .. ..**

1. **Why Selenium / Features Of Selenium**
   * **Open source freely available software**
   * **Only tool to support Multiple programming languages – java, c#, ruby, perl, python, js**
   * **OT – almost all the popular browsers**
   * **OT – Parallel Execution**
   * **No Dedicated machine is required**
   * **Headless browser execution**
   * **Distributed Execution**
2. **History**
3. **Versions**

* **Selenium IDE**
  + **Beginners – NOT much time**
  + ****
  + **Object Identification mechanism**
* **CSS**

|  |  |  |
| --- | --- | --- |
| **Element** | **Formula** | **Expression** |
|  | htmltag[attribute=’value’] |  |
|  | Htmltag#’idvalue’ OR #idvalue |  |
|  | Htmltag.’classvalue’ OR .classvalue |  |
| Parent to child | Htmltag[attribute = ‘value’] > child\_tag |  |

* **Xpath**

1. **Basic Xpath :**

Syntax: //htmltag[@attribute=’Value’]

Application: Actitime

Example: //input[@id='username']

//input[@placeholder='Username']

1. **Xpath Using Functions:**
   1. text()

Syntax: //htmltag[text()=’exactValue’]

Application: www.actitime.com

Example: //li[text()='About actiTIME']

* 1. contains(arg1, arg2)

arg1- can be attribute or function call

arg2 – can be corresponding partial/complete value

Syntax - //htmltag[contains(arg1,arg2)]

Example- //li[contains(text(),'About')]

//img[contains(@src,'timer')]

* 1. starts-with(arg1,arg2)

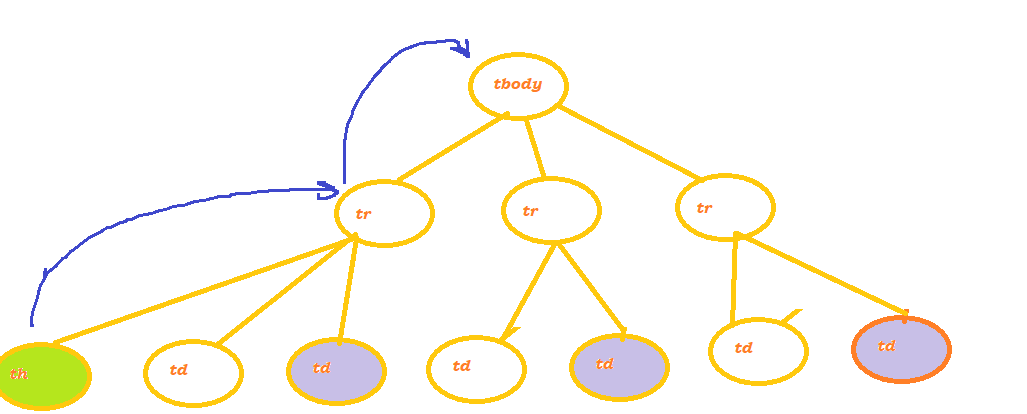
arg1- can be attribute or function call

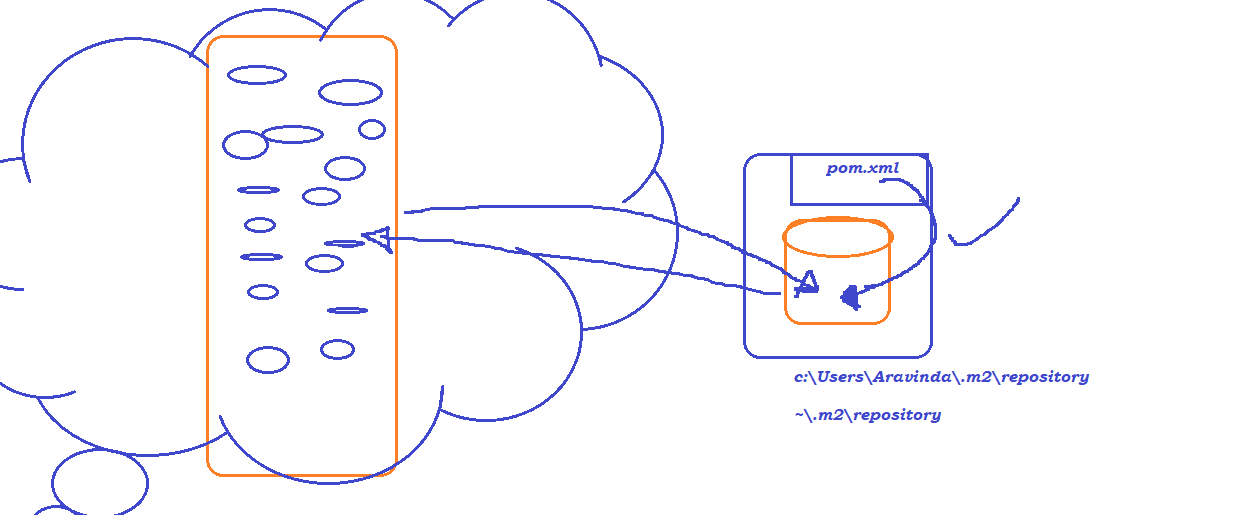
arg2 – can be corresponding starting value

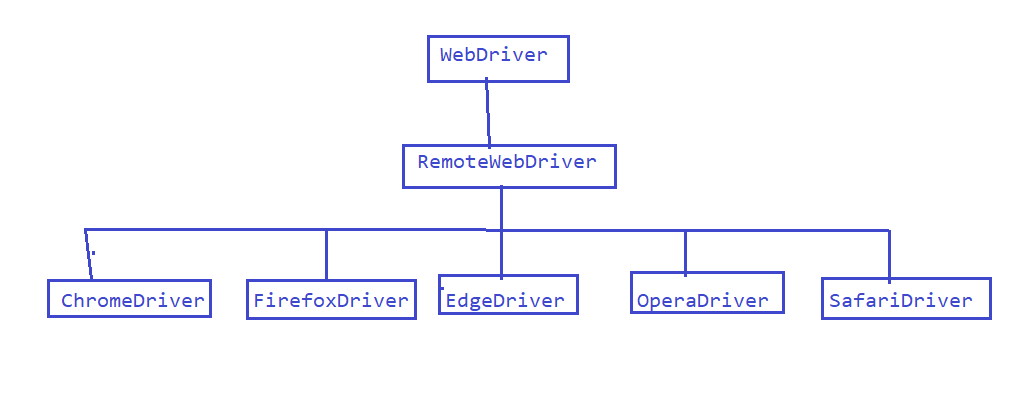
Syntax: //htmltag[starts-with(arg1,arg2)

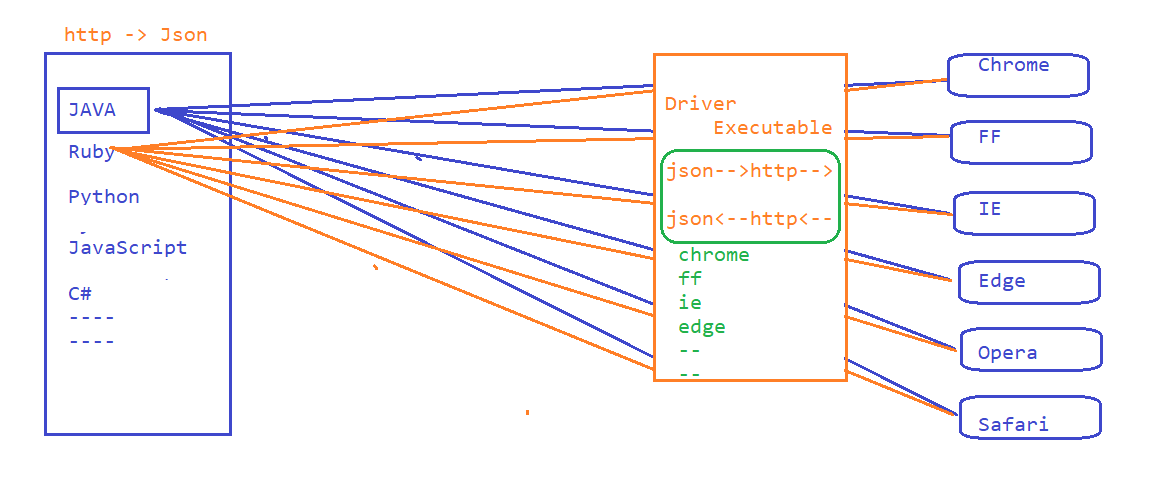
Example: //h3[span[starts-with(text(),'IBM')]]

Applicaion : Google Search

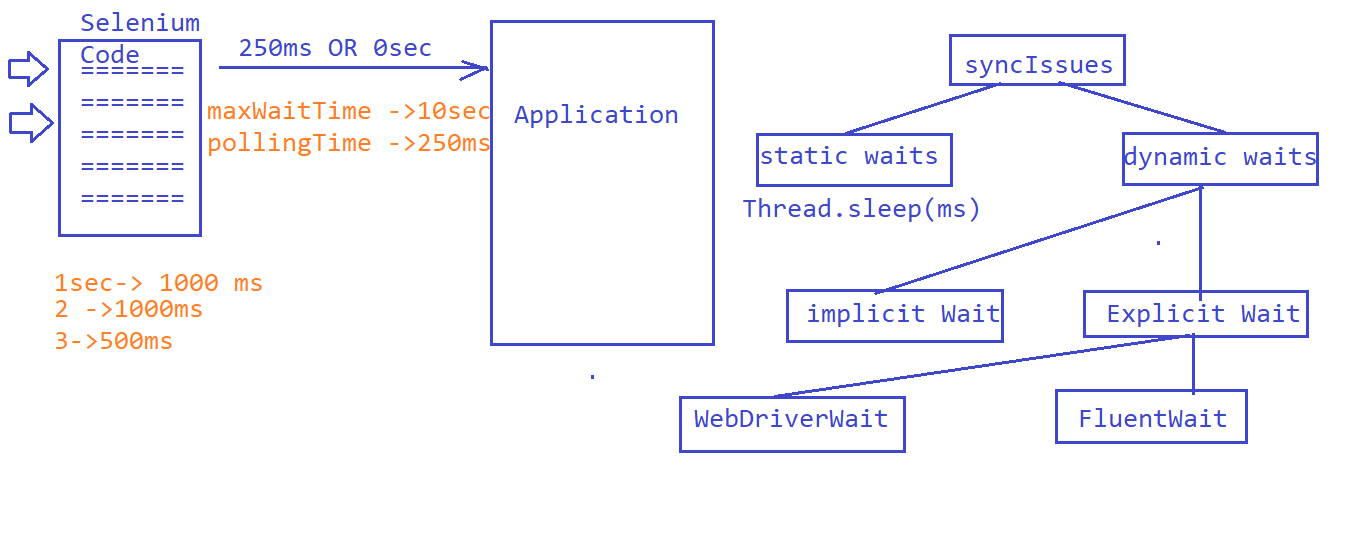
1. Logical Operators
   1. AND
      1. Syntax - //htmlTag[@attribute1=’value1’ and @attribute2=’value2’]
      2. //span[text()='IBM | LinkedIn' and not (text()='IBM - Wikipedia')]
   2. OR
      1. Syntax - //htmlTag[@attribute1=’value1’ or @attribute2=’value2’]
      2. Ex - //span[text()='IBM | LinkedIn' or text()='IBM - Wikipedia']
   3. NOT
      1. //span[text()='IBM | LinkedIn' and not (text()='IBM - Wikipedia')]
      2. //span[text()='IBM | LinkedIn']
   4. Examples
      1. //td[(@class='current day' or @class='wd day' or @class='we day') and text()='9'] – Redbus.in
      2. //td[text()='10' and not (@class='past day')]
2. Traversing from parent to child
   1. Syntax - //parent\_xpath/immediate\_child
   2. Syntax - //parent\_xpath//anywhere\_in\_parent
   3. Ex - //a[@id='loginButton']/div
   4. Ex - //div[@id='review-body']//span[contains(text(),'Galaxy A')]
   5. //div[@id='review-body']//span[(contains(text(),'iPhone')]
   6. //div[@id='review-body']//span[(contains(text(),'iPhone') or contains(text(),'iPad'))]
3. Traversing from child to parent
   * 1. 
     2. //tr[th[text()='Directed by']]//a
     3. //tbody[tr[th[text()='Main Camera']]]//td[@class='nfo']
     4. 
4. Axes Functions
   1. Traversing to siblings
      1. Following Sibling - Syntax - //element\_xpath/following-sibling::sibling\_tag
      2. //th[text()='Directed by']/following-sibling::td
      3. Preceding Sibling - Syntax - //element\_xpath/preceding-sibling::sibling\_tag
      4. //td[a[div[text()='REPORTS']]]/preceding-sibling::td
   2. Traversing to child
      1. //a[@id='loginButton']/child::div
   3. Traversing to parent
      1. //div[text()='REPORTS']/parent::a
   4. Traversing to ancestor
      1. //th[text()='Main Camera']/ancestor::tbody//td[@class='nfo']
      2. //h3[contains(text(),'Retreat')]/ancestor::div[contains(@class,'boxShadow')]//p[contains(@class,'font26')]
   5. Traversing till beginning of the page
      1. //div[@id='toc']/following::a
   6. Traversing till end of the page
      1. //div[@id='toc']/preceding::a
5. CREATING FIRST SELENIUM PROJECT
   1. MAVEN - >Everything is a plugin in maven
      1. Add the libraries
      2. Write code
      3. Delete the previously generated class files
      4. compile the program
      5. perform unit test
      6. create a jar / war file
      7. configure server
      8. copy the jar/war file to the server
   2. Installation
      1. Installing Maven on Windows
         1. Download maven from <https://maven.apache.org/download.cgi>
         2. unzip and keep it in any directory
         3. Set MAVEN\_HOME -D:\maven\apache-maven-3.6.1
         4. Set M2 -D:\maven\apache-maven-3.6.1\bin
         5. update PATH -> D:\maven\apache-maven-3.6.1\bin
      2. Installing Maven on Eclipse – Builtin
   3. Creating Selenium Project
      1. Change compiler version to the latest
      2. Change maven project JRE to the JRE Present inside JDK
      3. Add Selenium Dependencies
   4. MAVEN Architecture

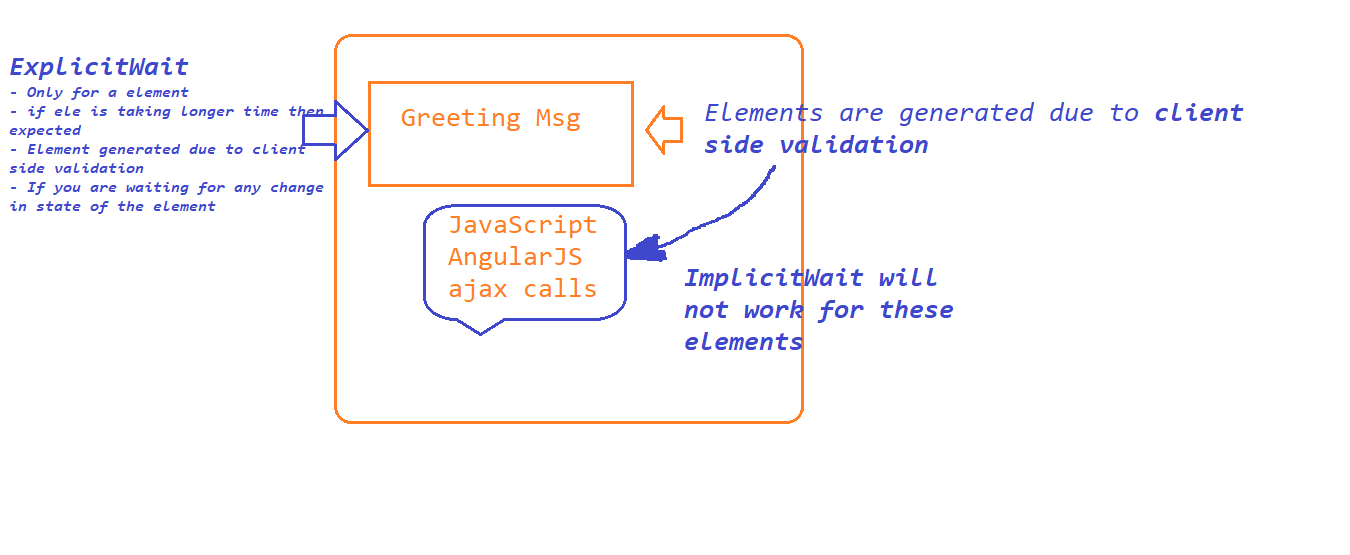


* 1. Selenium Architecture
     1. 
     2. W3C Architecture



* + 1. Sync Issues or Synchronization issue





* + 1. Fluent Wait

1. **FluentWait**

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

1. Step - Create an Object to Fluent Wait By passing input type

1.1 What is the WebElement

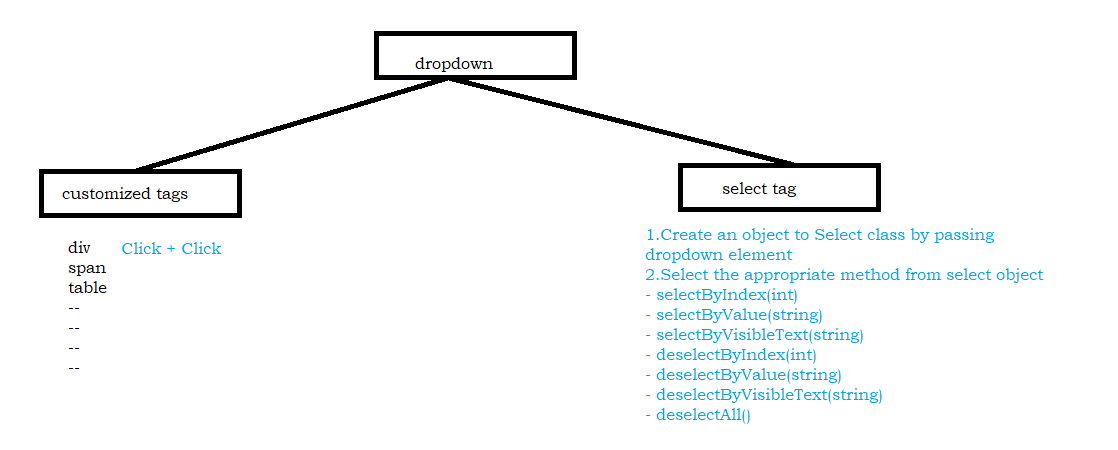
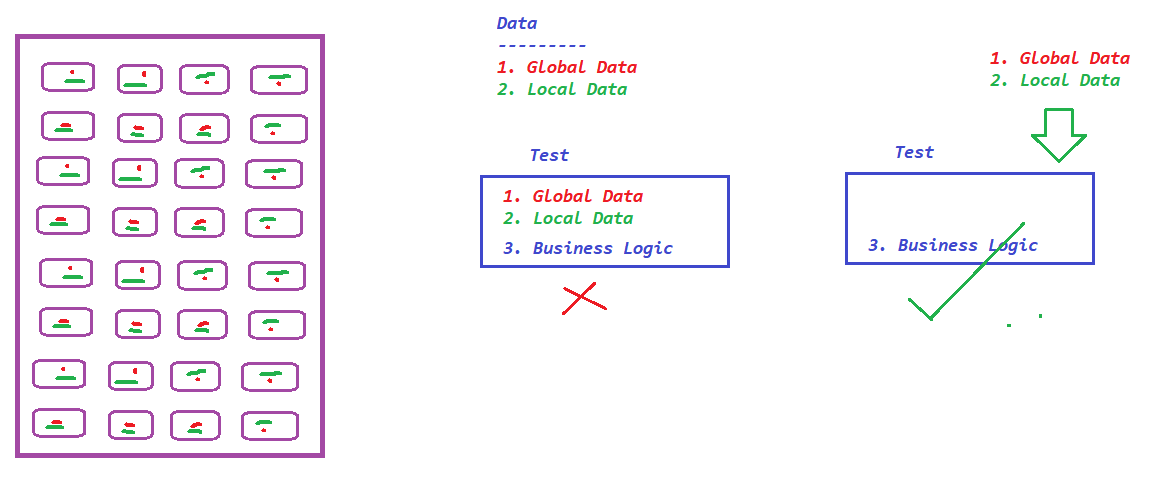
1.2 What is the max Wait time

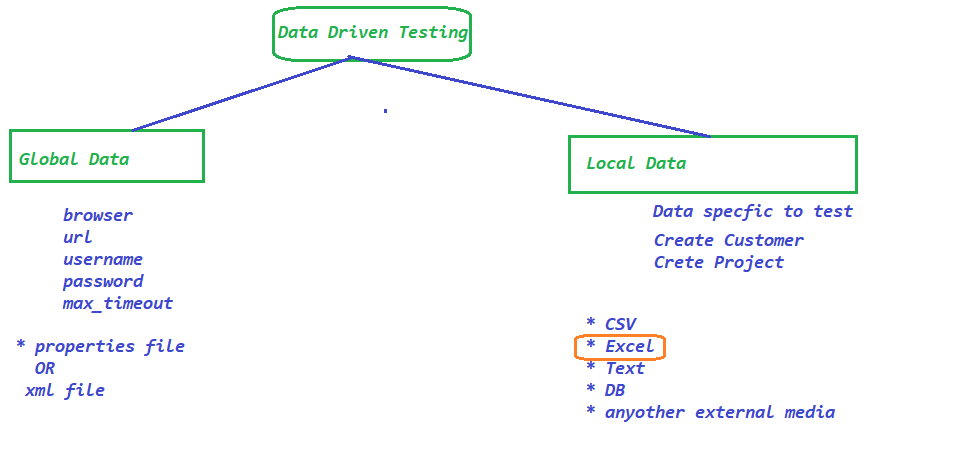
1.3 What is the polling time

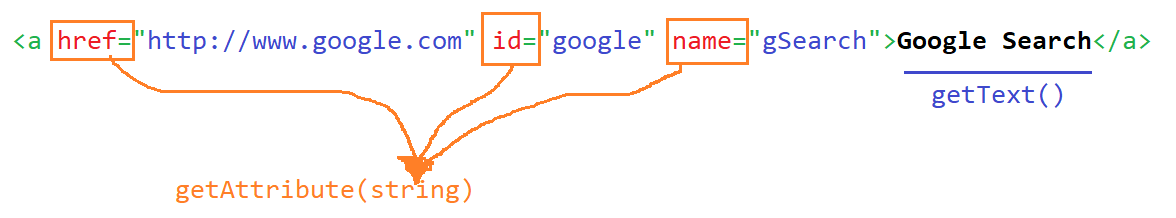
1.4 What are the exception to be ignored???

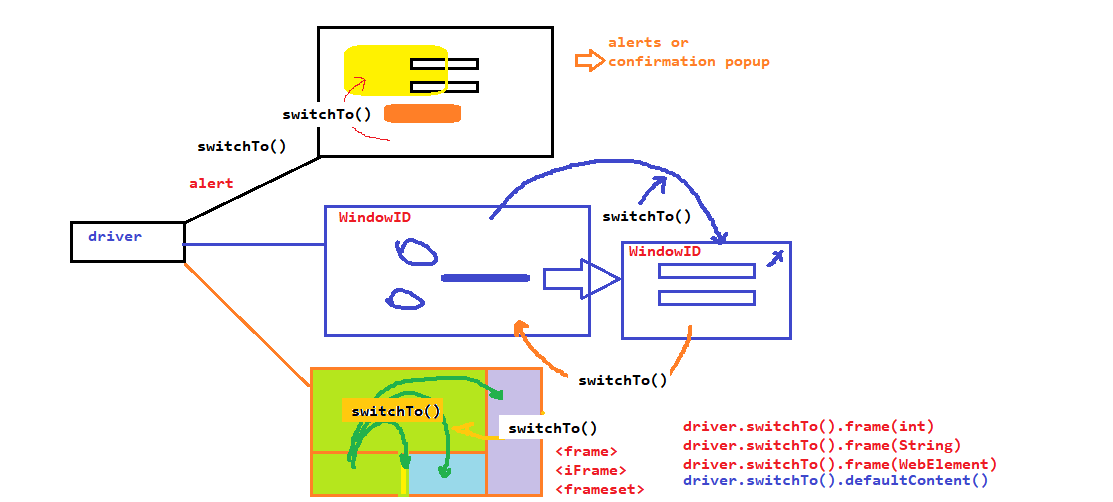
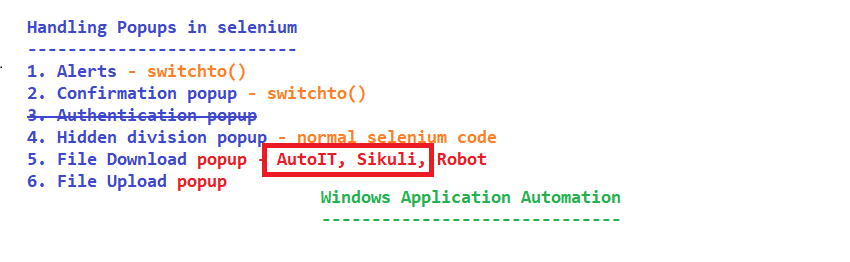
2. Write your own wait logic using Function interface. (Anonymous InnerClass).Inside function interface we have apply method. Keep your logic inside this apply method

3. use wait.until(funcion obj)

1. Handling Dropdown
   1. 
2. Writing Reusable Functions
3. Validations in selenium
   1. ele.isDisplayed()
   2. ele.isSelected()
   3. ele.isEnabled()
   4. ele.getAttribute("placeholder")
   5. ele.getText()
4. Actions in Selenium – *Exact kb and mouse movement operations.*
5. How to download Driver Executable automatically
6. Automate Some tests
   1. Execute the tests manually 1 or 2 or 3....
   2. Identify Test DATA
   3. Identify the reusable functions
   4. start implementing reusable functions
   5. Call them from the TEST
   6. Execute the newly created TEST 1.2.3.4.5.........10-15 times
   7. CI/CD to run your tests
7. HANDLING TOAST MESSAGES
   1. Talk to developers
   2. search for a keyword toast in DOM
   3. search for a message (which you see on toast message) in DOM
8. Read the Text on WebElement
9. Data Driven Testing 



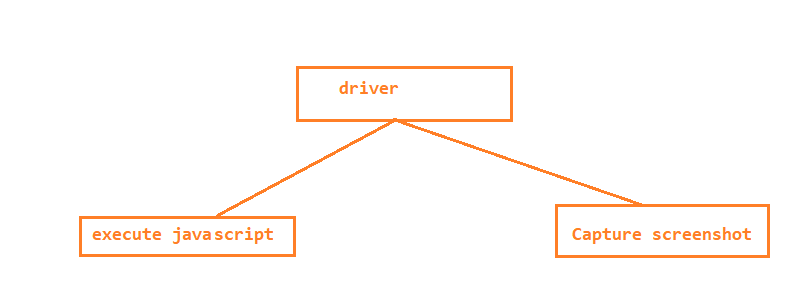


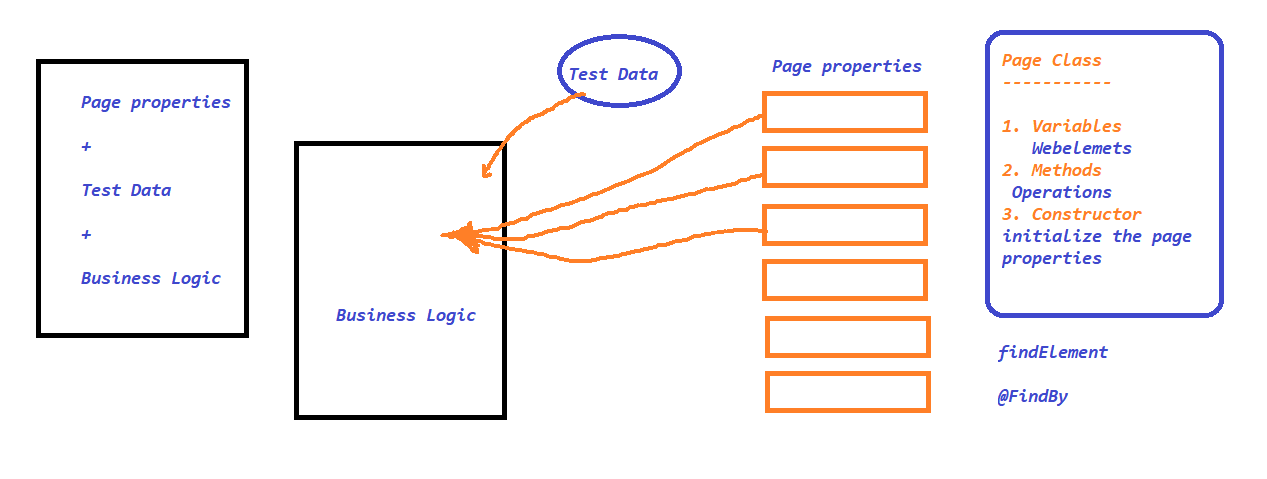
1. switchTo
   1. 
   2. 
2. FRAMEWORKS

-----------------------

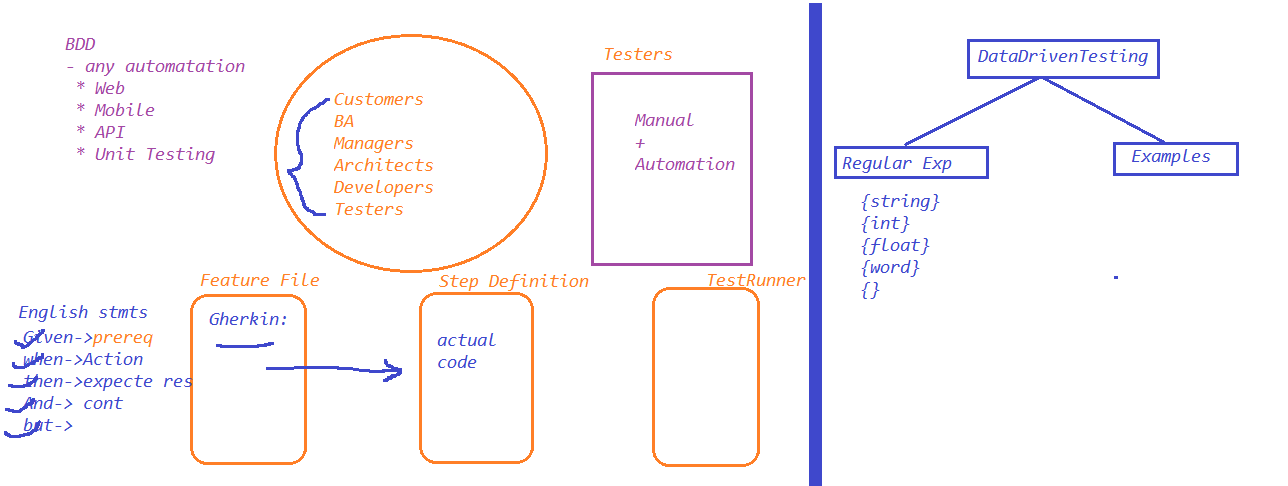
1. TESTNG



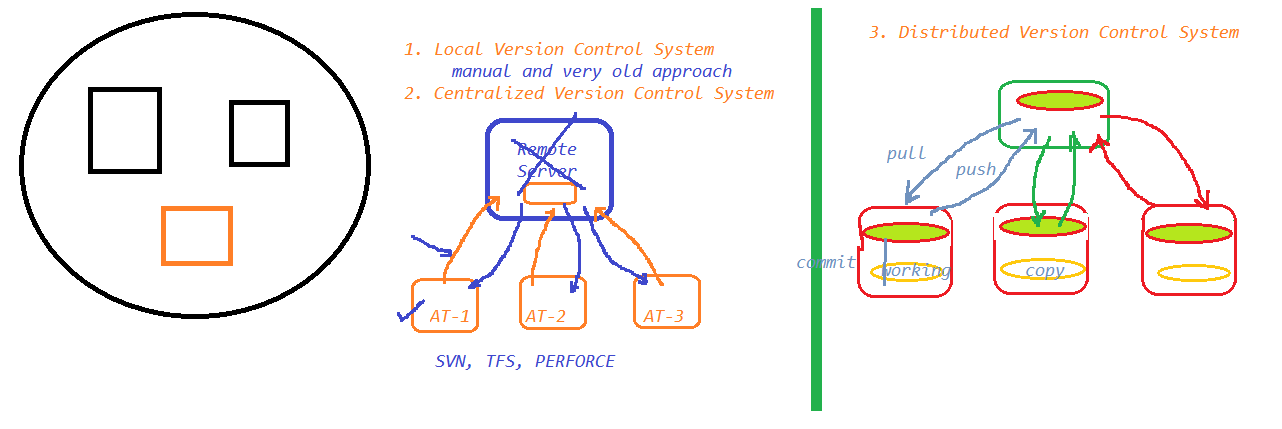




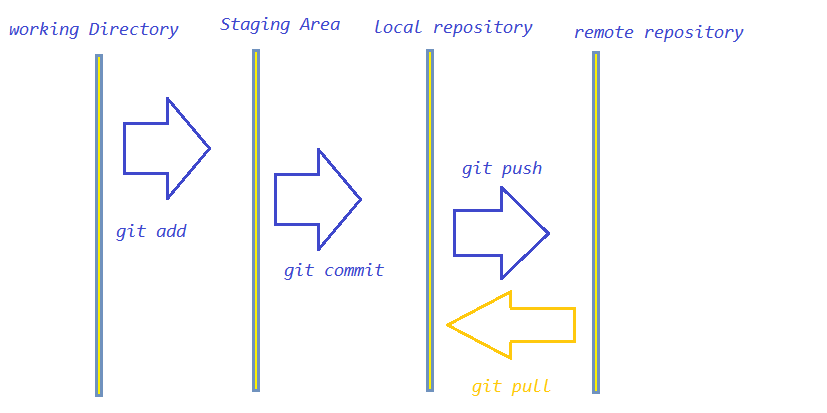
**BDD**

****

GIT



Architecture of GIT



* **~~Selenium RC~~**
* **Selenium WebDriver - 3.141.59**
  + **Operations**
    - **click – button, dropdown, image, hyperlink ...**
    - **type – textbox, text area..**
    - **dropdown elements**
    - **getting text from application**
    - **getting attribute value of a element**
  + **KB and Mouse operations**
  + **Automate –** 
    - **tooltip**
    - **suggestions**
    - **mouse hover**
  + **Browser operation**
  + **End to end tests**
  + **Popup**
    - **alert**
    - **confirmation popup**
    - **hidden division popup**
    - **file download**
    - **file upload**
    - **pageload popup**
  + **Sikuli / Auto IT**
  + **Data driven testing**
    - **Global Data – Common to all the tests**
    - **Local Data- specific to tests**
* **Selenium Grid**
  + **Distributed Execution**

**Frameworks**

* **TestNG**
  + **annotations**
  + **group**
  + **data driven testing**
  + **priority**
  + **Asserts**
  + **parallel execution**
  + **HTML Reports**
  + **execute the failed tests**
  + **customized reports**
  + **screenshots**
  + **execute java script**
* **POM – Page Object Model**

**Maven – Build Automation Tool**

1. **Installation**
2. **Maven build life cycle**
3. **phases of maven**
4. **Create a Maven Project**
5. **batch run – without opening eclipse**

**BDD – Behavioural Driven Development**

1. **Developer, tester, manager, Business analyst, Customer, Architect**
2. **Gherkin language**
   1. **Feature File**
   2. **Step Definition**
   3. **Test Runner**

**GIT – Distributed Version Control Software**

1. **Installation**
2. **Adding roles to the users etc**
3. **Difference between local repository / Global Repository**
4. **Basic GIT Commands**
5. **stash / unstash**
6. **revert the commit**
7. **resolve conflicts**

**Jenkins – CI/CD/CT**

1. **Installation**
2. **Configuration**
3. **Plugin management**
4. **How to create Builds / Jobs**
5. **How to execute the tests**
6. **How to execute the tests Automatically**
7. **How to analyze logs / Reports**