* What is automation testing ?

→ Converting manual test cases into test script with the help of some tool is called as automation testing.

→ The tools are called as automation testing tools.

* Which type of application we can automate ?

→ Web application as well as mobile application (Appium).

* When we will go for automation testing ?

→ If the application is stable we can go for automation testing (second release).

* Which test cases we can automate ?

→ Regression test cases as well as functional test cases.

* Which test cases we can't automate ?

→ if manual test cases is incorrect we can't automate.
→ Human's dependency testcases we can't automate like OTP, captcha, Barcode, fingerprint, facelock.
→ Gaming application we can't automate.
→ MP3, MP4 related testcases we can't automate.

Date: / /

* Why we go for Automation testing?

→ To save time

→ To perform more operation in less resource.

→ for better accuracy and efficiency.

→ To sustain in market competion.

* What are the major automation testing tool we have?

→ Selenium (open source tool)
→ QTP (license based tool)

* What are the major 3rd party tool's we have?

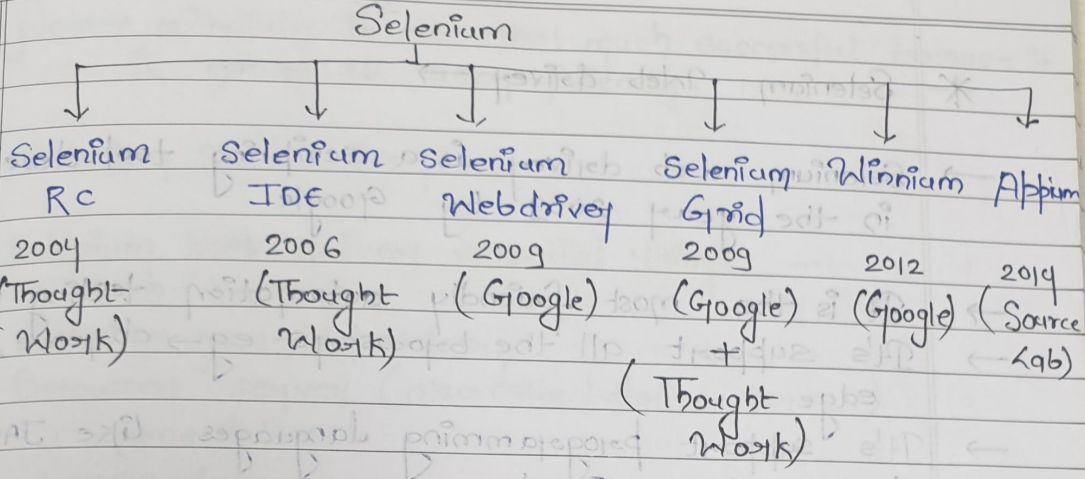
→ Auto IT
→ Sikuli

* father of Seleniam → JSON NUGGINS

* History of Selenium →

→ Selenium is the community name in which automation testing tools are present.

→



* Selenium RC →

→ Selenium RC stands for remote Control.

→ It is a scripting tool developed in the year 2004 by thought Work (company name)

→ The drawback of the tool is it cannot automate secured applications like https.

* Selenium IDE → (Integrated Development Environment)

→ It is a UI related automation testing tool developed in the year 2006 by TW.

→ The drawback of this tool is it cannot automated AJAX applications (Asynchronised Java Script XML Applications).

* Selenium Web driver →

→ Selenium web driver is a scripting tool developed in the year 2009 by Google.

→ It is the most friendly automation testing tool.
→ It's support all the browsers. eg → chrome, firefox, edge.
→ It's support programming languages like JAVA, python, RUBY, Pearl.
→ The only drawback of this application is it cannot automate stand alone applications directly.
→ With the help of 3rd party tools it can automate stand alone applications.

* Selenium Grid →

→ It is a scripting tool develop in a year 2009 by Google and Two.
→ Selenium grid features are same as selenium web driver but it is much more preferable choice to perform compatibility testing.

* Winnium →

→ Winnium is a scripting tool developed in the year 2012 by google.
→ Winnium can automate stand alone application directly.

→ All the browsers are stand alone application.
→ www.facebook.com → it is a web application.

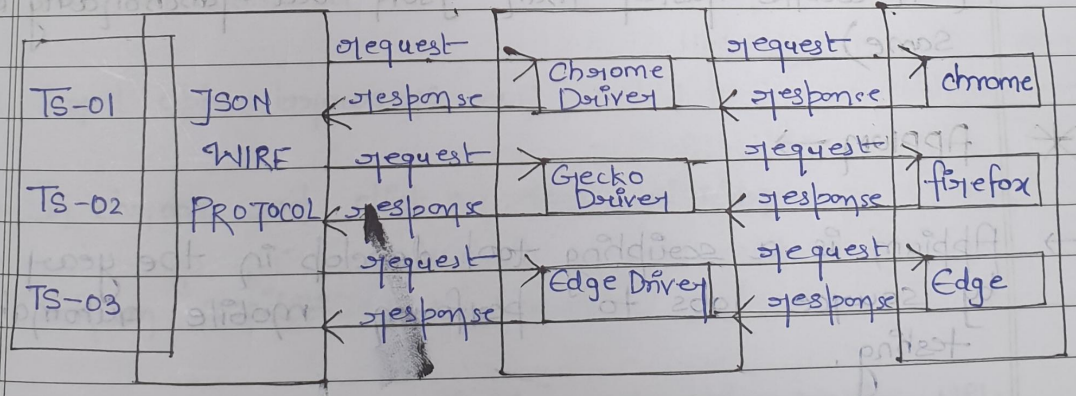
papergrid
Date: / /

Note → Winnium is not that much successful because it is not at all user friendly.

* Selenium Web driver →

⇒ Selenium web driver essential things →
a) JDK 1.8 (version)
b) Selenium jar / dependency
c) Browser server (also called driver server)
i) chrome driver
ii) Gecko driver
iii) edge driver
d) Actual browser
i) chrome
ii) firefox
iii) Edge

⇒ Selenium Web driver communication level architecture →



IDE tool → Browser Server → Actual Browser

↳ We have to write Test scripts in 2 different tool →
Eclipse & Intelligent

Date: / /

→ To establish the connection in between client IDE tool and actual browsers internally few operations will be performed.

a) First we have to write test scripts based on JAVA and selenium.

b) In the client IDE tool one protocol is there called as JSON wire protocol (which will convert JAVA to JSON & JSON to JAVA).

c) Once it will be converted to JSON the drivers will carry the request to actual browsers to perform the operation.

d) Once the operation is performed it will carry the response to client IDE tool by converting JSON to JAVA language.

NOTE → In case of selenium 4 we don't use JSON wire protocol instead of that we use W3C protocol for better performance. (It works faster than JSON but functionality is same)

* Appium →

→ Appium is a scripting tool develop in the year 2014 by source labs to perform mobile automation testing.

*
1)
2)
3)
4)
Note →
5)
*
1)
2)
3)
4)
5)
6)
7)
Note →
8)

* Steps to download selenium jar file →

1) Go to selenium.dev website
2) Click on download.
3) Scroll upto previous releases & Click on releases
4) Select the version and download jar file.

Note → don't download zip file

5) Once it will be downloaded copy and paste it to the selenium folder.

* Steps to download chrome drivers →

1) Go to selenium.dev website
2) click on download
3) Scroll upto browsers
4) In chrome and click on documentation.
5) Click on "availibility dashboard"
6) Select chromeDriver win64/win32 and copy the URL.
7) Open new tab and paste the URL

Note → Automatically it will be downloaded

8) Once it will be downloaded Extract the .zip file to selenium folder.

* Steps to download Geckodriver →

→ Go to selenium.dev website
→ click on download
→ Scroll down upto previous release and click on browsers
→ In firefox and click on documentation
→ Click on "geckodriver release"
→ In 0.34 scroll down upto assets
→ Click on "show all 13 assets"
→ click on win64.zip file it will download
→ Once it will be downloaded extract it to the selenium file folder.

* Steps to add jar file in project →

→ Right click on the project.
→ Go to build path → configure build path
→ Click on libraries
→ click on classpath
→ Click on "Add External Jar".
→ Select selenium jar
→ click on "apply and close".

Note → Selenium jar will be visible in "referenced libraries".

If the return type is void we can't use in print statement.

* Web driver and Web element methods →

1) get("URL") → This method will help us to access to a web application.

syntax → driver.get("URL");

2) getTitle() → With the help of this method we can fetch the title of a current web page.

syntax → driver.getTitle();

3) getCurrentUrl() → This method will help us to fetch the current web page URL.

syntax → driver.getCurrentUrl();

4) getPageSource() → With the help of this method we can fetch the front end source code of a current web page.

syntax → driver.getPageSource();

5) manage() → With the help of this method we can perform different type of operations in browser window. with the help of method chaining concept.

eg:- Maximization, minimisation, customize the window size, and relocate the window.

6) Navigate() → With the help of this method we can perform navigation related operations in a browser.

eg:- back, forward, refresh, accessing to a web applicat

Note → This method is acting as a helper method to perform all this operation.

7) close() → With the help of this method we can close the parent window.

syntax → driver.close();

8) findElement() → With the help of this method we can find an unique web element present in the web page.

→ The return type of this method is web element.

Syntax → driver.findElement(By.locator());

9) findElements() → With the help of this method we can find multiple web elements present in the web page.

→ The return type of this method is ~~list of web element~~. List<webElement>. {collection interface}

Syntax → driver.findElements(By locator());

10) sendkeys() → This method will help us to pass the string value inside a text box present in a web page.

Syntax → element.sendkeys();

11) Click() → With the help of this method we can perform click operation in a webpage.

Syntax → element.click();

→ The methods which you r calling by driver reference is called as web driver method for the whole operation.

→ The methods which we call by element reference is called as web element method and only for specific element we use it.

12) Submit() → This method will help us to perform any type of submit operation in a web page.

Note → This method will work only if "type = submit" is present inside the tag.

Syntax → element.submit();

13) getText() → This method will help us to fetch text value from a webpage in string format.

Syntax → element.getText();

14) isDisplayed() → With the help of this method we can verify that element is displayed or not.

Syntax → element.isDisplayed();

15) isEnabled() → With the help of this method we can verify an element is enabled or disabled.

Syntax → element.isEnabled();

16) isSelected() → This method helps us to verify an element is selected or not.

Syntax → element.isSelected();

17.) getSize() → With the help of this method we can fetch the size of a particular web element.
Syntax → element.getSize();

18.) getLocation() → This method helps us to fetch the location of a particular element.
Syntax → element.getLocation();

19) getAttribute() → With the help of this method we can fetch the attribute value of a particular web element.
Syntax → element.getAttribute();

20) quit() → With the help of this method we can close the parent window as well as child windows.
Syntax → driver.quit();

Jar will be adding in path and drivers

papergrid
Date: / /

* LAUNCHING CHROME BROWSER

→ We need to set the property of chrome driver path.
→ We have to create an object of chromeDriver.
→ To access any web application we need to use a method called as get("String URL").

Program →

```
class LaunchChromeBrowser
{
public static void main(String[] args)
  { (setting the path of chrome driver)
   System.setProperty("webdriver.chrome.driver",
       "C:\\Selenium folder\\chromedriver-win64\\chromedriver.exe");
(creating an object for ChromeDriver)
 ChromeDriver driver = new ChromeDriver();
    (Access web application)
  driver.get("https://www.flipkart.com/");

  }
}
```

o/p → flipkart applicatn is opened in chrome Browser

→ path of chromedriver.

* System.setProperty( )

⇒ Here, System is the class which is present in java.lang.package. Here we are calling the method by with the help of system class i.e. the static method.

→ If we call anything by class with the help of classname - that means this is static.

* Scenario 1 → Launch chrome browser
2 → Access demoappqspiders.com
3 → fetch the title, URL, Source code of the current web page.

```
Class Program1 {
public static void main(String[] args) {
 System.setProperty("webdriver.chrome.driver",
    "C:\\seleniumfolder\\chromedriver\\");

ChromeDriver driver = new ChromeDriver();

driver.get("https://demoapps.qspiders.com/?scenario=1");

String title = driver.getTitle();
String Url = driver.getCurrentUrl();
String SrcCode = driver.getPageSource();

 sop(title);
 sop(Url);
 sop(SrcCode);
```

Note

* Maximization and Minimization of Window →

→ To perform maximizatn & minimizatn operations we have to use method chaining concept.

Syntax for maximizatn →

```
driver.manage().window().maximize();
```

In this above case driver is managing the window for maximization.

Syntax for minimization →

```
driver.manage().window().minimize();
```

In this above case driver is managing the window for minimization.

Note → In selenium 3 minimization operation cannot be performed, from selenium 4 we can perform minimizatn operatn.

papergrid

Date: / /

* Scenario 2 → 1) Launch chrome Browser
2) Access qspiders.com
3) Maximize and minimize the window

Note → In b/w maximise & minimise provide 2secs of interval

```
class MaximizationAndMinimazation
{
 public static void main (String [] args) throws Exception
 {
  System.set Property ("webdriver.chrome.driver",
                       "path of chrome driver");

  ChromeDriver driver = new ChromeDriver();
  driver.get ("https\\demoappqspiders.com\\");
  driver.manage().window().maximize();
  Thread.sleep (2000);
  driver.manage().window().minimize();
  Thread.sleep (2000);
  driver.manage().window().minimized();
 }
}
```

* Resize & Relocate of Window →

→ To perform resize operation we have to use a class called as Dimension, and we ve to use 1 method called as setSize().

Syntax → driver.manage().window().setSize(new Dimension(300,400));
relocate

→ To perform ~~relocation~~ operatn in a window we ve to use a class called as Point and we ve to use 1 method called as setPosition().

Syntax → driver.manage().window().setPosition(new Point (200,300));

* Scenario – 3 → Launch chrome Browser
Access flipkart.com
Maximise & minimize the window
Again maximise the window
resize & relocate the window.

Note → In b/w every operation provide 2000 ms or 2sec time interval.

```
class ResizeAndRelocateWindow
{
 psvm(String[]args) throws Exception
{
 System.setProperty("webdriver.chrome.driver",
                    "path of chrome driver");

 ChromeDriver driver = new ChromeDriver();
```

```
driver.get("https://www.flipkart.com/");
Thread.sleep(2000);
driver.manage().window().maximize();
Thread.sleep(2000);
driver.manage().window().minimize();
Thread.sleep(2000);
driver.manage().window().maximize();
Thread.sleep(2000);
//custome size 1st way
driver.manage().window().setSize(new Dimension(300,400));
//custome size 2nd way
Dimension d = new Dimension(300,400);
driver.manage().window().setsize(d);
Thread.sleep(500);
// custome window 1st way
driver.manage().window().setPosition(new Point(200,300));
//custome window 2nd way
Point p = new Point(200,300);
driver.manage().window().setPosition(p);
}
}
```

Date: / /

* Navigation Interface and its methods →

Inside navigation interface 4 methods are present.

1) to("url")
2) back()
3) forward()
4) refresh()

1) to("url") → With the help of this method we can access to a web application.

2) back() → This method will help us to go back to the previous web page.

3) forward() → With the help of this method we can navigate to the next web page.

4) refresh() → This method will help us to refresh the current web page.
(⊖ we cannot refresh the previous web page)

* Scenario 4 → 1) Launch chrome browser
2) access flipkart.com
3) access amazon.com
4) Go back to flipkart
5) Navigate forward to amazon
6) Refresh the current web page.
7) Close the window

Note → Pls b/w all the operations provide 2sec time interval.

```
PSVM (String[] args) throws Exception {
System.setProperty("webdriver.chrome.driver",
                   "path of chrome driver");
Chrome driver driver = new ChromeDriver();
// Navigation nav = driver.navigate(); → one way
// nav.to("https://www.flipkart.com/");
driver.navigate().to("https://www.flipkart.com/"); → 2nd wa
driver.manage().window().maximize();
Thread.sleep(3000);
// nav.to("https://www.amazon.in/");
driver.navigate().to("https://www.amazon.in/");
Thread.sleep(3000);
// nav.back();
driver.navigate().back();
```

```
Thread.sleep(2000);
// nav.forward();

driver.navigate().forward();

Thread.sleep(2000);
// nav.refresh();

driver.navigate().refresh();

Thread.sleep(2000);

driver.close();
}
```

Date: / /

* Scenario 5 → 1) Launch chrome browser
2) Access myntra.com
3) Maximise the window
4) Custome the size of the window
5) Relocate the window
6) Again maximise the window
7) Access one more applicatn. Selenium.dev
8) & Verify that selenium.dev website is accessed (Get title name)
9) Go back to previous web page
10) Refreshit
11) Go bak forword to next web page
12) Close the window.

Note: In b/w all operations give time interval of 2sec

```
psvm(String[] args) throws Exception.
{
System.setProperty("webdriver.chrome.driver", "path of chrome driver");

ChromeDriver driver = new ChromeDriver();

Navigation nav= driver.navigate();

nav.to("myntra.com");
Thread.sleep(2000);
driver.manage().window().maximize();

Thread.sleep(2000);
driver.manage().window().setSize(new Dimension(200,300));
```

```
Thread.sleep(2000);
driver.manage().window().setPosition(new Point(400,600));

Thread.sleep(2000);
driver.manage().window().maximise();

Thread.sleep(2000);
nav.to("selenium.dev");

Sring title = driver.getTitle();
SOP("Current webpage is" + title);

Thread.sleep(2000);
nav.back();

Thread.sleep(2000);
nav.refresh();

Thread.sleep(2000);
nav.forward();

Thread.sleep(2000);
nav.driver.close();

}
```

HTML

→ HTML stands for hypertext markup language.
→ HTML is not a case sensitive programming language.
→ HTML depends upon 3 major components.

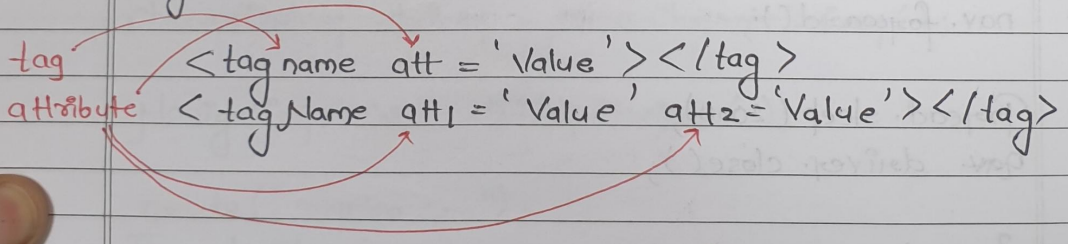
a) tag
b) Attribute
c) Visible text / link test

a) tag → The text value which we are providing just after the angular bracket is consider as tag.
eg → tag < >

b) Attribute → The information which we are providing inside the tag is considered as attribute.

eg
→ All attributes must contain some value.
→ Inside the tag multiples attributes can be present.

eg →



c) Visible / link text → The text value which we are providing in between same open and close tag is considered as visible text.

eg→ Visible text / link text → <tag> Visible text </tag>

<tag att = 'value'> Visible text </tag>

link-text → The text value which is present in between open and close "a" tag is considered as link text.

eg→ <a> text
↳ link text

* Rules of html →

1) All the html pages start with html tag.

2) Inside html tag 2 siblings will be there
   a) head
   b) body

3) Once the head tag will be closed, body tag will be started.

eg→ <html>
Parent and child ← <head> - </head>
Sibling ← <body>
</body>
</html>

Date: / /

* Creating a Web page —

First name
Mid name
Last name
Password
o Male
o Female
o Others
☐ Accept all the terms and conditions
submit

```
<html>
    <head>
        <title>Happy birthday</title>
    </head>
    <body>
        <center>
            <h1>Registration</h1>
            <b>
            <lable>First name</lable>
            <input type='text'></input><br></br>
            <lable>Middle name</lable>
                <input type='text'></input><br></br>
            <lable>Last name='text'></lable>
                <input type='text'></input><br></br>
            <lable>Password</lable>
                <input type='password'></input><br></br>
            <input type='radio' name='a'></input>
                <lable>Male</lable><br></br>
            <input type='radio' name='a'></input>
                <lable>female</lable><br></br>
            <input type='radio' name='a'></input>
                <lable>others</lable><br></br>
```

```
<input type = 'checkbox' name='a'></input>
    <lable> Accept all terms & conditions </lable><br></br>
<input type = 'button' value='submit'></input>
</b>
</center>
</body>
</html>
```

* Creation of dropdown →

To create a dropdown you have to use select tag and inside select tag you have to use option tag.

State [KA ▽]

Bank Name [SBI ▽]

```
<html>
    <head>
        <title>DropdownCreation</title>
    </head>
    <body>
        <lable>State</lable>
        <select>
            <option> KA </option>
            <option> AP </option>
            <option> OR </option>
            <option> TS </option>
            MH
            KL
        </select>
```

```
<BR></BR>.          <label>Bank name</label>
<select>
    <option>RBI</option>
    <option>SBI</option>
    <option>ICICI</option>
    <option>HDFC</option>
</select>
</body>
</html>
```

* Loan details —

Name [ ]
Age [ ]
Account number [ ]
Password [ ]
Gender o Male
o Female
o Others
Bank [v]
Account type [v]
loan type [v]
☐ terms & Condition
[Register]

```
<html>
  <head>
    <title>Loan Details</title>
  </head>
  <body>
    <h1>Loan details</h1>
    <lable>Name</lable>
      <input type='text'></input><br></br>
    <lable>Age</lable>
      <input type='text'></input><br></br>
```

```
<lable>Account Number</lable>
    <input type='text'></input><br><br>
<lable> User Name </lable>
    <input type = 'password'></input><br><br>
    <input type = 'radio' name='a'></input>
    < lable> Male </lable><br><br>
    <input type = 'radio' name='a'></input>
     < lable> female </lable><br><br>
    <input type='radio' name='a'></input>
      <lable> others </lable> <br><br>
    <lable> BankName </lable>
       <select>
            <option> HDFC </option>
            <option> SBI </option>
        </select> <br><br>
    <lable>Account type </lable>
       <select>
            <option> Saving Account </option>
            <option> Current Account </option>
        </select><br><br>
     <lable> Loan Type </lable>
       <select>
          <option> Home loan </option>
          <option> Commercial Loan </option>
        </select><br><br>
     <input type ='checkbox' name ='a' ></input>
       <lable>Accept all terms & Conditions </lable><br><br>
     <input type = 'button' value = 'Register'></input>
</body>
</html>
```

* Web Table → To create a web table we've to use table tag.

→ Inside table tag, table body should come.
→ Inside table body, table rows should come.
→ Inside table row, table data should come.

eg→

```
<table border = 1>
    <table body> or <tbody>
        <table row> <tr>
            <table data> ——— </table data></td>
            <table data> ——— </table data>
            <table data> ——— </table data>
        </table row>
    </table body>
</table>
```

S No	Book Name	Book Price	Qty
1)	Harry Potter	2000	12
2)	All splendid sun	1000	40
3)	Kite runner up	750	30
4)	Two states	1500	20

```
<html>
    <head>
        <title> Table Creation </title>
    </head>
    <body>
        <h1> Library Table </h1>
        <table body>
            <tr>
                <td> SNo </td>
```

```
<td > BookName < /td>
<td> Book Price < /td>
<td> Quantity < /td>
</tr>
<tr>
<td> Sno< /td>
<td> BookName < /td>
<td> Book Price
<tr>
<td> 1 </td>
<td> Harry Potter</td>
<td> 2000 < /td>
<td> 12 < /td>
</tr>
<tr>
<td> €2 </td>
<td> All splendid sun < /td>
<td> 1000 </td>
<td> 40 < /td>
</tr>
<tr>
<td> 3 </td>
<td> kite runner up </td>
<td> 750 < /td>
<td> 20 </td>
</tr>
</tbody>
</table>
</body>
</html>
```

papergrid
Date: / /

* LOCATORS →

→ The elements which are present in a web page is called as web elements.

→ Locators are nothing but static methods present in 'By' class.

→ Locators are used to locate the web elements present in web page.

→ We are having 8 types of locators.

a) id()
b) name()
c) className()
d) linkText()
e) partialLinkText()
f) cssSelector()
g) xPath()
h) tagName()

a) id() → id locator will perform operations only with id attribute.

→ So find the unique web element based on id we have to check it is showing 1 of 1.

Search Syntax → [id = 'value']

Script Syntax → driver.findElement(By.id("id attribute value"));

(b) name() → name locator will perform operations only with name attribute and it should show 1 of 1.

Search Syntax → ['name = 'value']

Script Syntax → driver.findElement(By.name("namevalue"));

(c) className() → class name locator will perform operation only with class attribute and it should be 1 of 1.

Search Syntax → [class = 'value']

Script Syntax → driver.findElement(By.className("value"));

* Scenario 5 → Launch chrome Browser
→ Access demoapps.qspiders
→ In the web element section text box option is present, in that pass name, email id and password.

Note → To pass name use id locator, to pass email use name locator and for password use id locator.

```
ChromeDriver driver = new ChromeDriver();
driver.get("https://demoapps.qspiders.com/ui?scenario=1");
driver.manage().window().maximize();
Thread.sleep(3000)
driver.findElement(By.id("name")).sendKeys("aniruddha");
```

```
Thread.sleep(3000);
driver.findElement(By.name("email")).sendKeys
        ("mamta c@2022@gmail.com");
Thread.sleep(3000);
driver.findElement(By.id("password")).sendKeys
                        ("welcome");
    }
}
```

* Scenario 1 → Launch chrome browser
→ Access Orangehrm application
→ In the web element section in that access username and password.
→ Time interval 3000.

```
ChromeDriver driver = new ChromeDriver();
driver.get("https://opensource-demo.orangehrmlive.
                com/web/index.php/auth/login");
driver.manage().window().maximize();
  Thread.sleep(3000);
    driver.findElement(By.name("username")).sendKeys
                        ("Admin");
Thread.sleep(3000);
    driver.findElement(By.name("password")).
        sendKeys("admin123");
    driver.close();
```

* CSS Sel
as
Search Synta
Script
Syntax
* Scen

* cssSelector() ⟶ cssSelector performs operation with all the attributes as well as css expressions also.

Search Syntax → tagname[attribute = "attribute value"]
(attribute name)

Script ⟶ driver.findElement(By.cssSelector("input[placeholder='username']"));
Syntax (tagname) (attribute name) (attribute value)

* Scenario 8 ⟶ Launch Chrome Browser
→ Access demoapps qspiders
→ Maximize the window
→ Provide name, mail id, password
→ click on register button and do the registration operation by using cssSelector()

```
ChromeDriver driver = new ChromeDriver();
driver.get("demoapps.qspiders.com");
driver.manage().window().maximize();
Thread.sleep(3000);
WebElement element = driver.findElement
(By.cssSelector("input[placeholder='Enter your
name']"));
element.sendkeys("mamta");
Thread.sleep(3000);
driver.findElement(By.cssSelector("input[type=
'email']")).sendkeys("mamta2022
@gmail.com");
```

papergrid
Date: / /

```
Thread.sleep(3000);
driver.findElement(By.id("password")).
                    sendKeys("welcome");
Thread.sleep(3000);
WebElement element1 = driver.findElement
    (By.cssSelector("button[type='submit']"));
element1.click();
```

Scenario 9 → Launch chrome Browser
→ Access Orangehrm application
→ In the web element section provide user name, password and click on login button by using cssSelector().
→ Time interval of 3sec.

```
ChromeDriver driver = new ChromeDriver();
driver.get("Orangehrm url");
driver.manage().window().maximize();
Thread.sleep(3000);
WebElement element = driver.findElement(By.cssSelector
    ("input[placeholder='Username']"));
element.sendKeys("Admin");
driver.findElement(By.cssSelector("input[type='password']
    )).sendKeys("admin123");
WebElement element1 = driver.findElement(By.cssSelector
    ("button[type='submit']"));
element1.click();
driver.close();
```

9/2/24

X-Path

→ X-Path is
→ X path is
→ X path
play visi
with
as well

⇒ Types of

1) Absolute
2) Relative

1) Absolute

→ Absolute
the pa
→ Absolute

Tree

papergrid
Date: / /

9/2/24

X-Path() →

→ X-path is the last option to locate an unique Webelement.
→ X path is the most papular locator present in selenium.
→ X path can perform it's operations with all the attributes, by visible text as well as link text.
→ With the help of X-path we can locate static elements as well as dynamic elements also.

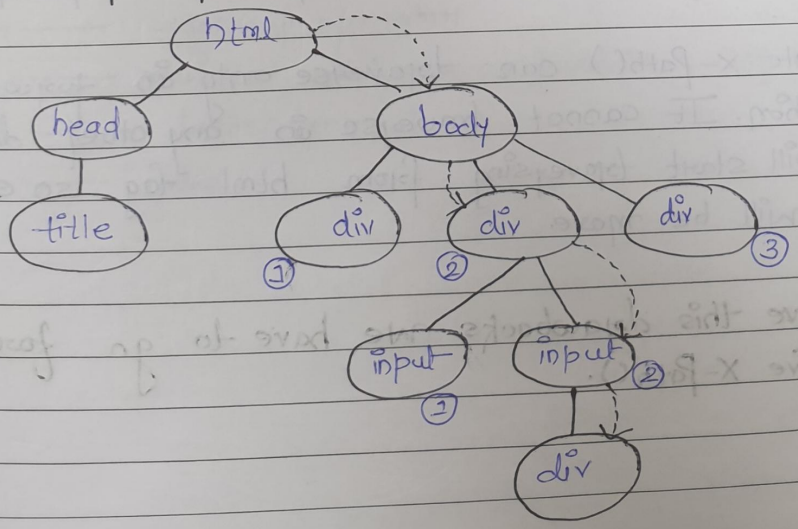
⇒ Types of X-Path() →

1) Absolute X-path()
2) Relative X-path()

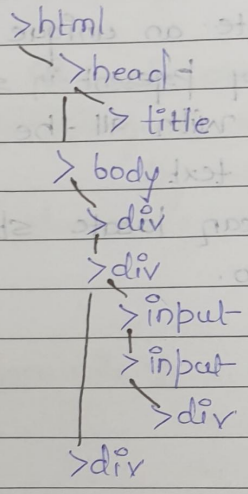
1) Absolute x-Path() →

→ Absolute X-path() will start journey from html tag to the particular unique Web element.
→ Absolute X-path() signature is " / "

Tree Structure →



Dom Structure →



Absolute X path →

/html/body/div[2]/input[2]/div

⇒ Drawback of Absolute X path () →

→ In real time we don't use absolute X path because

1) It is completely depends on index and index can be changed any time. So chance of test script failure is more.
2) Absolute X-Path() can traverse only in forward direction. It cannot traverse in any other directions.
3) It will start traversing from html tag so execution time will be more.

Note→ To solve this drawbacks we have to go for relative X-Path().

2) Relative X-Path () →

Relative X-Path we can divide in two types –

1) Basic
2) Advance

1) Basic →

a) X-Path by attribute value.
b) X-path by contains attribute value.
c) X-Path by visible text –
d) X-Path by contains visible text

2) Advance →

a) X-Path by forward traversing
b) —"— backward traversing.
c) —"— following siblings.
d) —"— preceding siblings.
e) —"— surrounding elements
f) —"— starts with
g) —"— regional operator
h) —"— multiple attributes and visible text
i) —"— descendent
j) —"— index

Date: / /

⇒ Basic –

a) X-path by attribute value →

→ This type of x-Path will always perform its operations with attribute value.

b) X-path by contains attribute value →

→ This type of x-Path will perform its operations with partial attribute value and we have to use contains method.

Syntax for attribute value →

Search Syntax → //tagname[@attribute name = 'attribute value'];

Script Syntax → driver.findElement(By.xpath("//tagname[@attribute name = 'attribute value']"));

Syntax for contains attribute value →

Search Syntax → //tagName[contains(text'@ attname, partial att value)];

Script Syntax → driver.findElement(By.xPath(tagname[contains(@ attname, partial att value)]));

→ text is a method. It is applicable for visible text.

→ contains is a method.

papergrid

Date: / /

c) x path by visible text →

→ This x-path will perform its operation with visible text as well as link text.

Search syntax → //tagName[text()='visible text'];

Script syntax → driver.findElement(By.xPath(//tagname[text()='visible text']));

d) x path by contains visible text →

This x path will perform its operation with partial link text as well as partial visible text by using contains method.

Search syntax → //tagname[contains(text(), 'partial visible text')];

Script Syntax → driver.findElement(By.xPath(//tagname[contains(text(), 'partial visible text')]));

arg 1 → text()

arg 2 → 'partial visible text'

Scenario 10 → Launch Chrome Browser
→ Access flipkart
→ Search for any product
→ fetch the result of searching operation

```
{
{
ChromeDriver driver = new ChromeDriver();
driver.get("https://www.flipkart.com/");
driver.manage().window().maximize();
Thread.sleep(3000);
WebElement element = driver.findElement(By.xPath
("//input[@name='q']"));
element.sendKeys("iphone 11");
element.submit();
Thread.sleep(3000);
WebElement result = driver.findElement(By.xPath
("//span[contains(text(),'showing')]"));
String result value = result.getText();
Sop(result Value);
Thread.sleep(3000);
driver.close();
}
}
```

for searching WebElement

for searching item

{Here, in inspect it is compulsory to have "type=submit" then only we can give submit button here}

{See we have to fetch the result "showing for iphone 11" and this will not same for every time. It keeps changing a dynamic element. So that's why we go with partial visible text because it is a text so we have to edit till where we get the unique value which we will not change ex}

{coz it is

{See text is a string so we ve to store in a string container}

{We want to print the string result so we use "sop".}

Here getText() → This method will provide you string text and its return value is string.

{search ching}

{searching item}

Scenario 11 → Launch ChromeDriver
→ Access to flipkart
→ Search for iphone 11 or any product
→ Click on Price-Low to high
→ fetch the first product name
→ close the window

```
ChromeDriver driver = new ChromeDriver();
driver.get("flipkart URL");
driver.manage().window().maximize();
Thread.sleep(3000);
{searching} WebElement element = driver.findElement(By.xpath
                ("//input[@name='q']"));
element.sendKeys("iphone 11");
{searching...} element.submit();
Thread.sleep(2000);
driver.findElement(By.xpath("//div[text()='Price --
                Low to high']")).click();
Thread.sleep(3000);
String ProductName = driver.findElement(By.xpath
        ("//div[text()='Apple iphone 5s (Silver,
                16 GB)']")).getText();
Thread.sleep(2000);
sop(ProductName);
Thread.sleep(2000);
driver.close();
```

coz it is

{→ ["Price - Low to high" is a text so we go with X-path for visible text]}

Again we have to fetch a product and store in a string container and print that product so use "sop".

Scenario 10 → Launch Chrome Browser
→ Access flipkart
→ Search for any product
→ fetch the result of searching operation

```
{
{
ChromeDriver driver = new ChromeDriver();
driver.get("https://www.flipkart.com/");
driver.manage().window().maximize();
Thread.sleep(3000);
WebElement element = driver.findElement(By.xPath
("//input[@name='q']"));
element.sendKeys("iphone 11");
element.submit();
Thread.sleep(3000);
WebElement result = driver.findElement(By.xPath
("//span[contains(text(),'showing')]"));
String resultvalue = result.getText();
sop(resultValue);
Thread.sleep(3000);
driver.close();
}
}
```

for searching item

{Here, in inspect it is compulsory to have type=submit then only we can give submit button here}

for searching item

{See we have to fetch the result "showing for iphone 11" and this will not same for every time. It keeps changing coz it is a dynamic element. so that's why we go with partial visible text because if it is a text so we have to edit a till where we get the unique value which we will not change ever}

{see text is a string so we ve to store in a string container}

{We want to print the string result so we use "sop".}

Here getText() → This method will provide you string text and its return value is string.

Scenario 11 → Launch Chrome Driver
→ Access to flipkart
→ Search for iphone 11 or any product
→ Click on Price-low to high
→ fetch the first product name
→ close the window

```
ChromeDriver driver = new ChromeDriver();
driver.get("flipkart URL");
driver.manage().window().maximize();
Thread.sleep(3000);
WebElement element = driver.findElement(By.xPath
("//input['@name='q']"));
element.sendKeys("iphone 11");
element.submit();
Thread.sleep(2000);
driver.findElement(By.xpath("//div[text()='Price--
Low to high']")).click();
Thread.sleep(3000);
String ProductName = driver.findElement(By.xpath
("//div[text()='Apple iphone 5s (Silver,
16 GB)']")).getText();
Thread.sleep(2000);
sop(ProductName);
Thread.sleep(2000);
driver.close();
```

searching item

{searching item}

{"Price-Low to high" is a text so we go with X-path for visible text}

Again we have to fetch a product and store in a string container and print that product so use "sop".

Scenario 12 → Launch Chrome Browser
→ Access flipkart
→ Search for any product
→ fetch all the product name
→ close the window.
→ and validate also from the actual value.

```
ChromeDriver driver = new ChromeDriver();
driver.get("flipkart url");
driver.manage().window().maximize();
Thread.sleep(3000);
WebElement element = driver.findElement(By.xpath
    ("//input[@type='text']")); element.sendKeys("iphone11pro");
element.submit();
Thread.sleep(2000);
List<WebElement> elements = driver.findElements
    (By.xpath("//div[@class='_4rR01T']"));
Thread.sleep(2000);
/* for(WebElement all : elements)
{
    System.out.println(all.getText());
}
*/

Iterator<WebElement> itr = elements.iterator();
while(itr.hasNext())
{
    String allValues = itr.next().getText();
    System.out.println(allValues);
}
```

for searching

we've to fetch all elements so that's why we go "for" loop to fetch

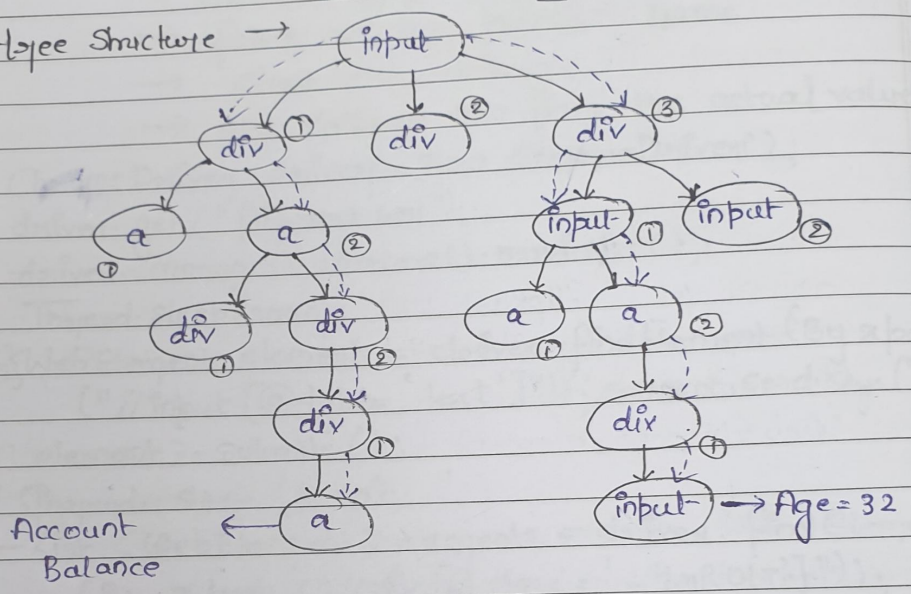
In interview we have to use iterator not for loop

We want all iphone 11 pro so we want multiple elements so that's y we use "list" here and for multiple elements we can't take visible text we have to use unique value which is common for every element.

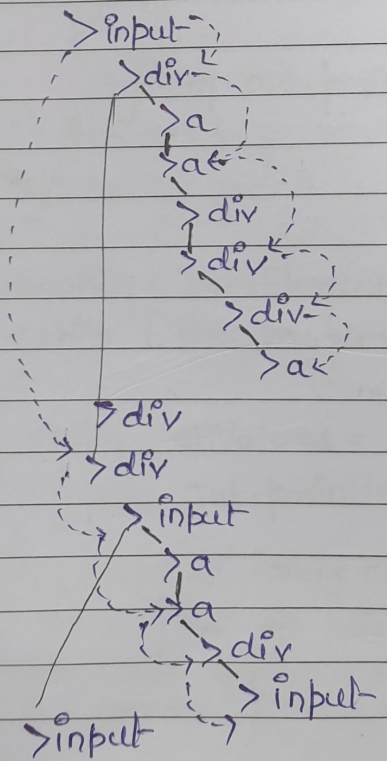
* X path by forward traversing and backward traversing.

[type = 'text']

⇒ tree Structure →



⇒ dome Structure →



papergrid
Date: / /

⇒ Forward traversing ⇒

for age ⇒ //input[@type='text']/div[3]/input[1]/a[2]/div/input

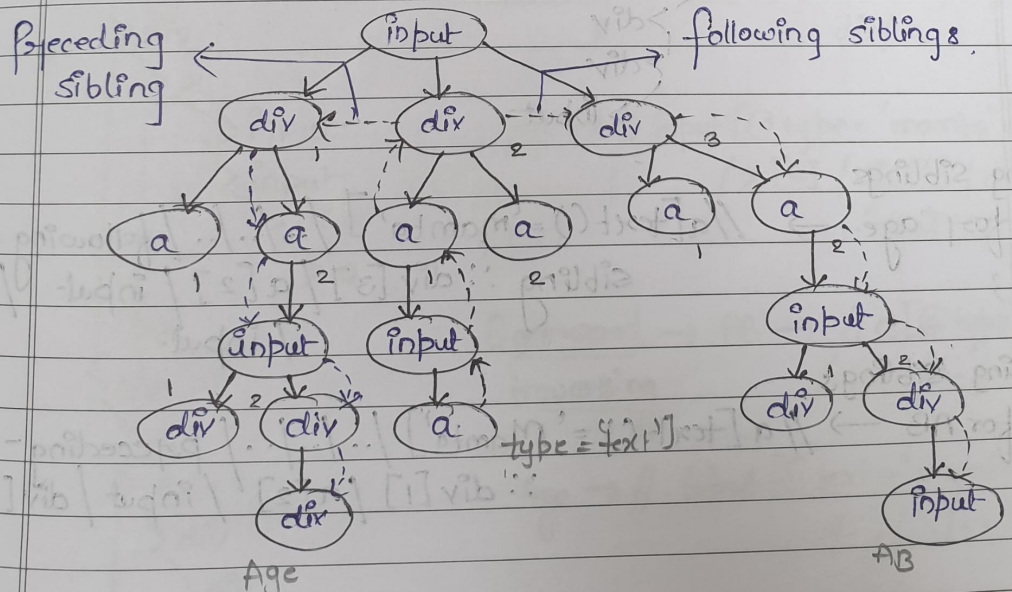
for Account balance ⟶ //input[@type='text']/div[1]/a[2]/div[2]/div/a

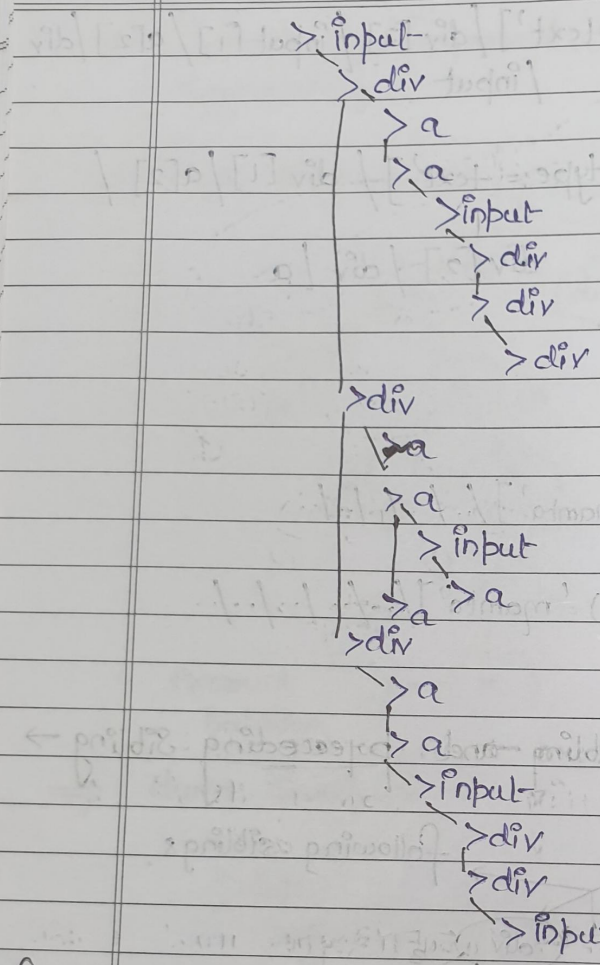
⇒ Backward traversing ⇒

forage ⟶ //a[text()='mamta']/../../../../..

for AB ⟶ //input[text()='mamta']/../../../../..

✳ X path by following sibling and preceding sibling →





following siblings

for age → //a[text()='mamta']/../../../following-sibling::div[3]/a[2]/input-0/div[3]/input-

Preceding siblings

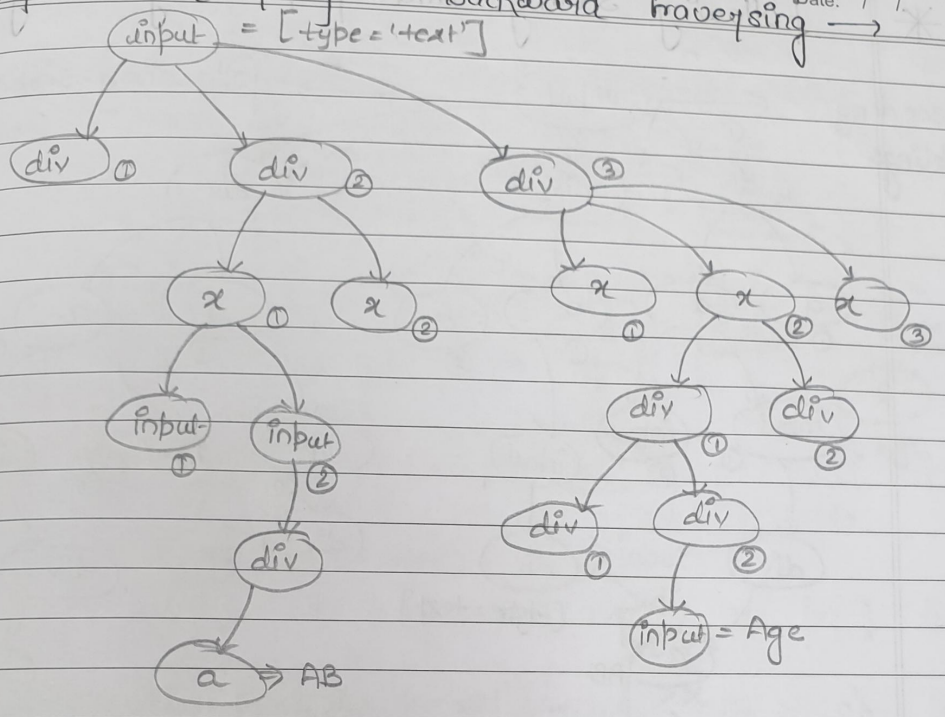
for AB → //a[text()='Mamta']/../../../preceding-sibling::div[1]/a[2]/input/div[2]/div

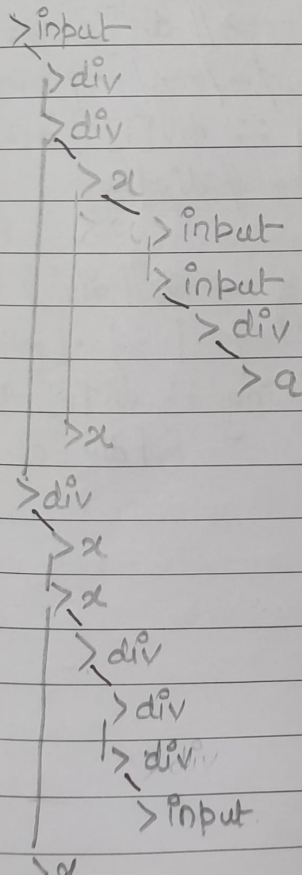
papergrid

Date: / /

Example →

* forward traversing & Backward traversing →





forward traversing → AB → //input [@type = 'mamta']
/div[2]/x[1]/input[2]/
div/a

Age → //input [@type = 'mamta']/div[3]
/x[2]/div[1]/div[2]/
input

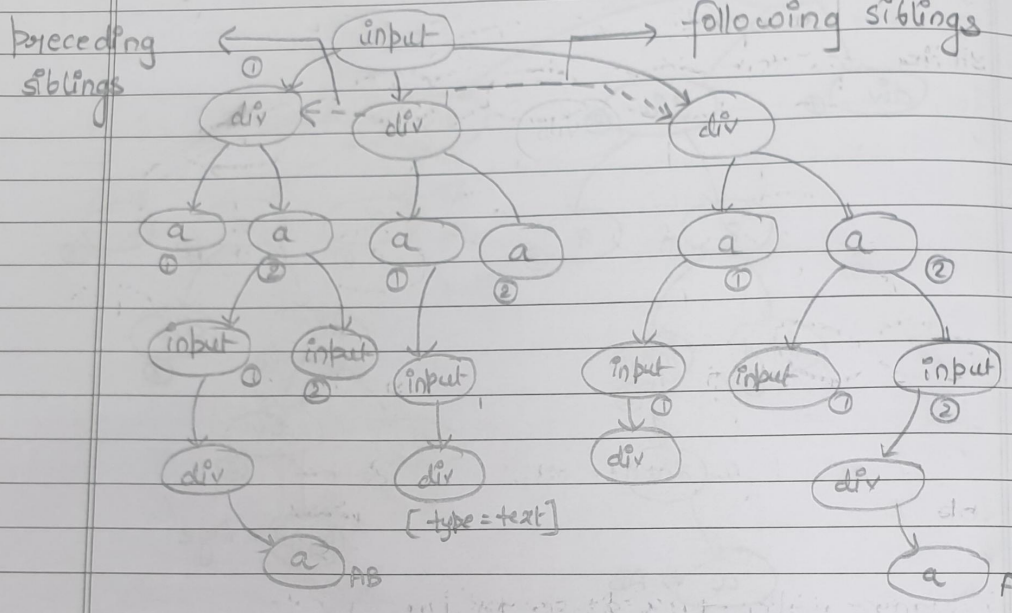
Backward traversing → AB → //a[@type = 'mamta']
/../../../../..

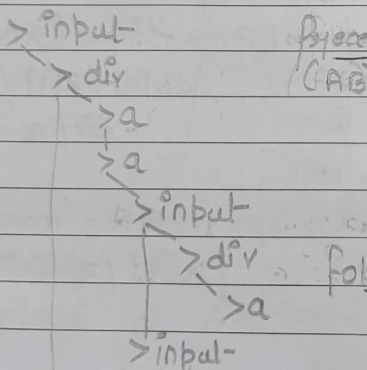
Age → //input [@text = 'mamta']/../..
/../../..

papergrid

Date: / /

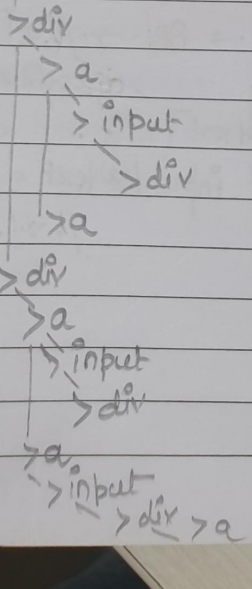
* Following Sibling and preceding sibling →





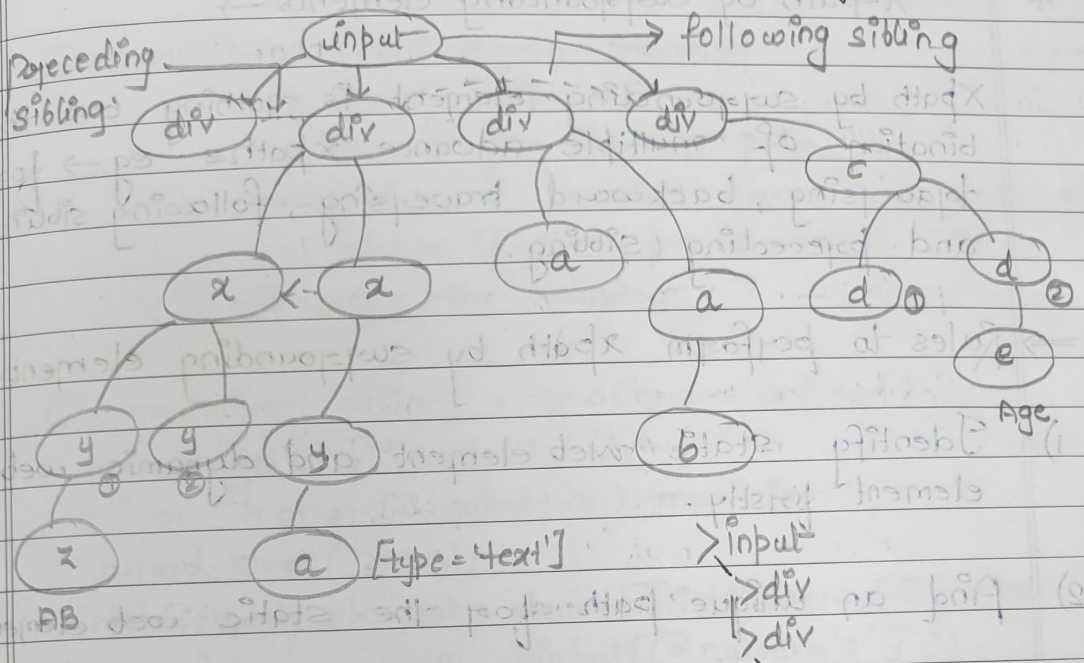
preceding sibling (AB) → // div[@type = 'mamta'] /../../../preceding-sibling :: div[1] | a[2] | input[1] | div | a

following sibling (Age) → // div[@type = 'mamta'] /../../../following-sibling :: div[1] | a[2] | input[2] | div | a



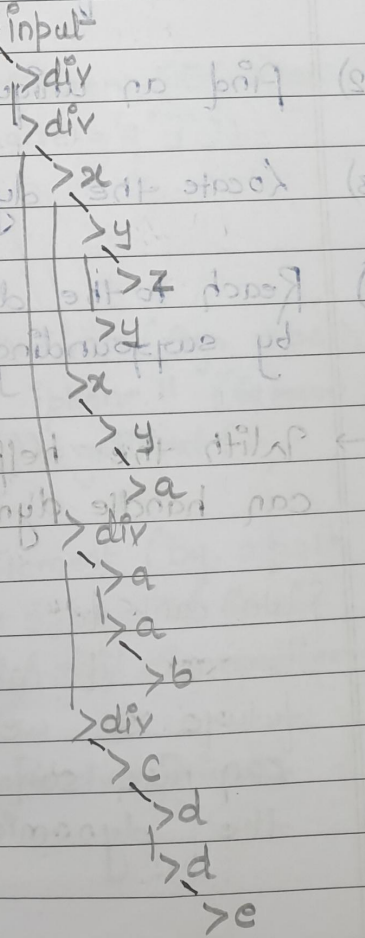
Date: / /

* Following Sibling and Preceding Sibling →



Preceding sibling → //a[@type='text']/..
(AB) /../preceding-sibling
::x[1]/y[1] | z

Following sibling → //a[@type='text']
(Age) /../../../following-
sibling::div[2] | c | d[2] | e



Date: / /

* X-path by Surrounding Elements →

Xpath by surrounding element is nothing but the combination of multiple advance xpaths. eg → forward traversing, backward traversing, following sibling and preceding sibling.

⇒ Rules to perform xpath by surrounding elements →

1) Identify static web element and dynamic web element firstly.
2) Find an unique path for the static web element.
3) Locate the dynamic web element.
4) Reach to the dynamic web element by using xpath by surrounding element.

Note → With the help of Xpath by surrounding elements we can handle dynamic web elements.

{ We can't take directly xpath for dynamic elements because it might be frequently changed in future so we go with the locator. and we can use surrounding (body) and we can fetch the dynamic element by getText(). }

* Scenario13 → Lauch chrome Browser
→ Access flipkart
→ Search for iphone 11 pro
→ fetch the product name and price of apple iphone 11 pro (Green, 64 GB)
→ close the window

```
ChromeDriver driver = new ChromeDriver();
driver.get("flipkart url");
driver.manage().window().maximize();
Thread.sleep(2000);
WebElement element = driver.findElement(By.xpath("//
                    input[@name='q']"));
element.SendKeys("iphone 11");
element.submit();
Thread.sleep(2000);
String productName = driver.findElement(By.xpath("//
                div[text()='Apple iphone 11 (Green, 64GB)'
                ]")).getText();
Thread.sleep(2000);
String productPrice = driver.findElement(By.xpath("//
                div[text()='Apple iphone 11 (Green, 64GB)']
    /../following-sibling::div[1]/div/div.getText();
Thread.sleep(2000);
SOP(ProductName + "----->" + ProductPrice);
Thread.sleep(2000);
driver.close();
```

* Classes are more changeable so for dynamic its good to go with locator.

→ length method is only applicable for string class.

→ In array we go for size method.

papergrid

Date: / /

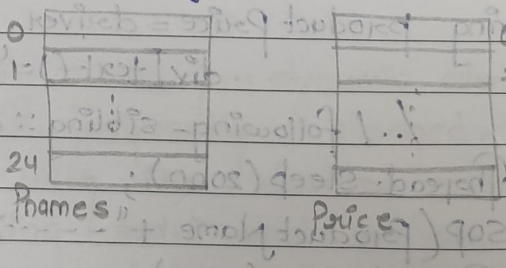
Scenario 14 ⟶ Access flipkart
Search for iphone 11
fetch all product name and its price

Here size means the total size of the list starting from 0 and ending on size

```
ChromeDriver driver = new ChromeDriver();
driver.get("https://www.flipkart.com/");
driver.manage().window().maximize();
Thread.sleep(2000);
WebElement element = driver.findElement(By.xpath
        ("//input[@name='q']"));
element.sendKeys("iphone 11");
element.submit();
List<WebElement> pNames = driver.findElements(By.xpath
        ("//div[@class='_4ROiT']"));
List<WebElement> pPrice = driver.findElements(By.xpath
        ("//div[@class='_4ROiT']/../following-sibling::div/div[1]
        /div/div[1]"));
for(int i=0; i<=pNames.size()-1; i++)
{
    Sop(pNames.get(i).getText()+"----->"+pPrices.get(i).
        getText());
}
Thread.sleep(2000)
driver.close();
```

Here we write the code in such (generic way) a way in future the count of product name & price will change, it will not affect our code.

starting point, ending point



⇒ i<=pNames.size()-1 ⟶ Here we fetch price based on product name

→ Here we store the all data into the list format so it will be store in the form of object we fetch the value from index like for '0' index → 1 value is there, for '1' index → 1 value is there (this things stored will be stored based on collection)

→ How to fetch invisible value from my ~~cursor~~ list → by using getO method.
→ And I want all values so" i "~~to~~ value will be changing so papergrid
i=0 then i will reinitialize to 1 by i++. & we want text Date: / /
value which is present in the object → by using getText(). (store in a string).

* X path by Starts-with →
→ xpath by starts-with can perform operation with attribute as well as visible text.

Note → The partial value which we are using it should be start from the beginning.

Syntax → for visible text →

//tagname[Starts-with(text(), 'Partial Visible text')];
method arg1 arg2

for attribute Value →

//tagname [Starts-with (@attname, 'Partial Att Value')]

* x path by multiple attributes and visible text →
→ In this case two combinations are there -
a) attributes and attributes
b) attributes and visibleText

a) Ex → attributes and attributes
<input type="text" placeholder='username']

xpath → //input[@type='text' and @placeholder='username']

b) Ex → attributes and visibletext
Apple iphone 11

xpath → //div[@class='_24Abc' and text()='Apple iphone']

* X-path by index →

(//div[@class='_24Abc'])[24] 1 of 24

Syntax → [//tagname[@attname='attValue']])[index]

* Scenario15 → Launch Chrome Browser
→ Access flipkart application
→ Select any product (iphone 11)
→ Click on Add to compare of 1, 3, 5 and last product
→ Close the window.

```
ChromeDriver driver = new ChromeDriver();
driver.get("flipkart URL");
driver.manage().window().maximize();
Thread.sleep(2000);
WebElement element = driver.findElement(By.xpath
        ("//input[@name='q']"));
element.submit();
Thread.sleep(2000);
driver.findElement(By.xpath("(//span[text()=
        'Add to compare'])[1]")).click();
Thread.sleep(2000);
```

⟹ Validate means to compare b/w the expected value and the actual value.

papergrid

Date: / /

```
driver.findElement(By.xpath("(//span[text()='Add to compare'])[3]")).click();
Thread.sleep(2000);
driver.findElement(By.xpath("(//span[text()='Add to compare'])[5]")).click();
Thread.sleep(2000);
driver.findElement(By.xpath("//span[text()='Add to compare'])[24]")).click();
Thread.sleep(2000);
driver.close();
```

* Scenario 16 → Launch Chrome Browser
→ Access demoapps.qspiders.com
→ Under web elements section click on button
→ Fetch the header of the page and validate it
→ In option 1 click on yes
→ In option 2 click on No
→ In option 3 select 4

```
String expectedValue = "Feed Back on Shopping Registration";
ChromeDriver driver = new ChromeDriver();
driver.get("Demoapps.qspiders.com");
driver.manage().window().maximize();
Thread.sleep(2000);
driver.findElement(By.xpath("//section[text()='Button']")).click();
Thread.sleep(3000);
String actualValue = driver.findElement(By.xpath("//h3[text()='Feed Back on shopping Registration']")).getText();
```

Here we use equals() ← by default overriding is done because string ↑ is a inbuilt class.

```
if (actualValue.equals(expectedValue))
{
  sop("Validation pass");
}
else
{
  sop("validation failed");
}
Thread.sleep(3000);
driver.findElement(By.xpath("//p[text()='1. Are you satisfied with the registration process?']/button[text()='yes']")).click();
```

Here we fetch the xpath with que for future use ... may be this que will changed

```
Thread.sleep(3000);
driver.findElement(By.xpath("//p[text()='2. Do you have any concerns about the registration process?']/../button[text()='No']")).click();
Thread.sleep(3000);
driver.findElement(By.xpath("//p[text()='3. Rate on the scale of 1 to 5']/../button[text()='4']")).click();
Thread.sleep(3000);
driver.close();
```

WEB TABLE -

papergrid
Date: / /

Assignment

* Launch chrome Browser
→ Access demoapps.qspiders.com
→ click on web table under webelement section
→ fetch the quantity and product name of the 3rd product
o/p → [Pname ---> Qty]

* Launch chrome Browser
→ Access: demoapps.qspiders.com
→ click on web table and under webelement section
→ fetch the product names and the discount
o/p [pname ---> discount]

* Scenario 17 → Launch chromeBrowser
→ Access demoapps.qspider.com
→ Click on webtable under webelements
→ fetch the quantity name and product name of the 3rd product.
→ o/p is shown as → Pname ---> quantity.

```
ChromeDriver driver = new ChromeDriver();
driver.get("demoapps.qspider.com");
driver.manage().window().maximize();
Thread.sleep(2000);
driver.findElement(By.xpath("//section[text()='Web Table']")).click();
Thread.sleep(2000);
String element = driver.findElement(By.xpath("//table/tbody/tr[3]/th[text()='Applelphone']")).getText();
Thread.sleep(2000);
String element1 = driver.findElement(By.xpath("//table/tbody/tr[3]/td[contains(text(),'')][2]")).getText();
Thread.sleep(2000);
System.out.println(element+"---->"+element1);

Thread.sleep(2000);
driver.close();
```

visible text

→ Why we use contains here?
→ Bcoz quantity will be changed so we want a particular data which is more unique.
→ contains we are using means partial and partial means nothing but something is there. By contains we will get more accurate data.

* Scenario 17 → Launch chromeBrowser
→ Access demoapps.qspider.com
→ Click on web table under web elements
→ fetch the quantity name and product name of the 3rd product.
→ o/p is shown as → Pname ---> quantity.

```
ChromeDriver driver = new ChromeDriver();
driver.get("demoapps.qspider.com");
driver.manage().window().maximize();
Thread.sleep(2000);
driver.findElement(By.xpath("//section[text()='Web Table']")).click();
Thread.sleep(2000);
String element = driver.findElement(By.xpath("//tbody/tr[3]/th[text()='Applelphone']")).getText();
Thread.sleep(2000);
String element1 = driver.findElement(By.xpath("//tbody/tr[3]/td[contains(text(),'[2]')]")).getText();
Thread.sleep(2000);
System.out.println(element+"---->"+element1);

Thread.sleep(2000);
driver.close();
```

→ Why we use contains here?
→ Bcoz quantity will be changed so we want a particular data which is more unique.
→ contains we are using means partial and partial means nothing but something is there. By contains we will get more accurate data.

* Scenario 18 → Launch chrome Browser
→ Access demoapps.qspider.com
→ click on webtable under web element section
→ fetch the all product names and the discount.
→ o/p → is shown as → Pr.name ----> discount (for all).

```
ChromeDriver driver = new ChromeDriver();
driver.get("demoapps.qspider.com");
driver.manage().window().maximize();
Thread.sleep(2000);
driver.findElement(By.xpath("//section[text()='WebTable']")).click();
Thread.sleep(2000);
List<WebElement> pNames = driver.findElements(By.xpath
        ("//th[@scope='row']"));

List<WebElement> discount = driver.findElements(By.
        xpath("//th[@scope='row']/../td
        contains(text(),'')][3]"));

for(int i=0; i<=pNames.size()-1; i++)
{
  sop(pNames.get(i).getText()+"---->"+discount.
        get(i).getText());

}
Thread.sleep(2000);
driver.close();
```

Scenario 19 → All the steps are same as previous assi
→ fetch the quantity of all the products
in ascending order.

```
ChromeDriver driver = new ChromeDriver();
Thread.sleep(3000);
driver.get("URL");
driver.manage().window().maximize();
Thread.sleep(3000);
List<WebElement> qty = driver.findElements
        (By.xpath("//table//tbody//tr[*]//td[contai
                text(),'1')][2]"));
ArrayList<String> alist = new ArrayList<String>();
Iterator<WebElement> itr = qty.iterator();
Thread.sleep(3000);
while(itr.hasNext())
{
String allValues = itr.next().getText();
alist.add(allvalues);
}
Collections.sort(alist);
Sop(alist);
```

Questions

papergrid
Date: / /

1) Diff. between findElement & findElements.

findElement	findElements
→ With the help of findElement we can find a single webElement.	With the help of findElements we can find multiple web elements.
→ The return type of findElement is WebElement.	The return type of findElements is List<WebElement>.

2) Write down the diff b/w close() and quit().

Close() → With the help of close() we can close only the parent window.

Quit() → with the help of quit() we can close the parent window as well as child window.

3) Diff b/w get() and navigate().

get()	navigate()
→ With the help of get() we can perform only 1 operation.	With the help of navigate() we can perform 4 operatn. in a web applicat
→ get() helps us to access in a web application.	navigate() helps us.— a) to access d) to refresh b) to go back c) to go forward

get()

→ If we use get method until the web page is fully loaded the next operation cannot be performed.
(Internal execution process to save time)

navigate()

navigate() will not wait for the web page to be fully loaded that only it will perform the next operation.

* Absolute xpath

→ Absolute xpath denotes by single '/'.

→ Absolute xpath is completely depend on Index. So the chance of failure is more.

→ Absolute xpath can traverse only in forward direction.

→ Absolute xpath works slower than relative xpath because it will start its journey from html tag.

Relative xpath

Relative xpath denotes by '//'.

Relative xpath is not only depends on index it has basic & advance types to get an unique element.

Relative xpath can traverse in multiple directions.

Relative xpath works faster than absolute xpath

papergrid

Date: / /

Scenario 20 → Launch Chrome Browser
→ Access demoapps.qspiders.com
→ Click on checkbox section under webElement
→ for the 1st particular question select Whatsapp
→ for the 2nd question sleet shoes
→ for the 3rd particular quest^n select regarding similar products.

Note → Also validate the heading of this page "Check out page".

```
String expectedElement = "checkout Page";
WebDriver driver = new ChromeDriver();
driver.get("demoapps.qspiders.com");
driver.manage().window().maximize();
Thread.sleep(2000);
String actualElement = driver.findElement(By.xpath
    ("h1[text()='checkout page']")).getText();
if(actualElement.equals(expectedElement))
{
 sop("validation passesd");
}
else
{
 sop("validation failed");
}
Thread.sleep(2000);
driver.findElement(By.xpath("//p[text()='1. On
 which platform would you like to receive
 notifications ?']/following-sibling::main/div[2]/
  input")).click();
```

```
Thread.sleep(2000);
driver.findElement(By.xpath("//p[text()='2. would you like to receive recommendations for similar products?']//following-sibling::main//div//input")).click();
Thread.sleep(2000);

driver.findElement(By.xpath("//p[text()='3. would you like to receive any customer assistance for this product?']//following-sibling::main//div[2]//input")).click();

Thread.sleep(2000);
driver.close();
```

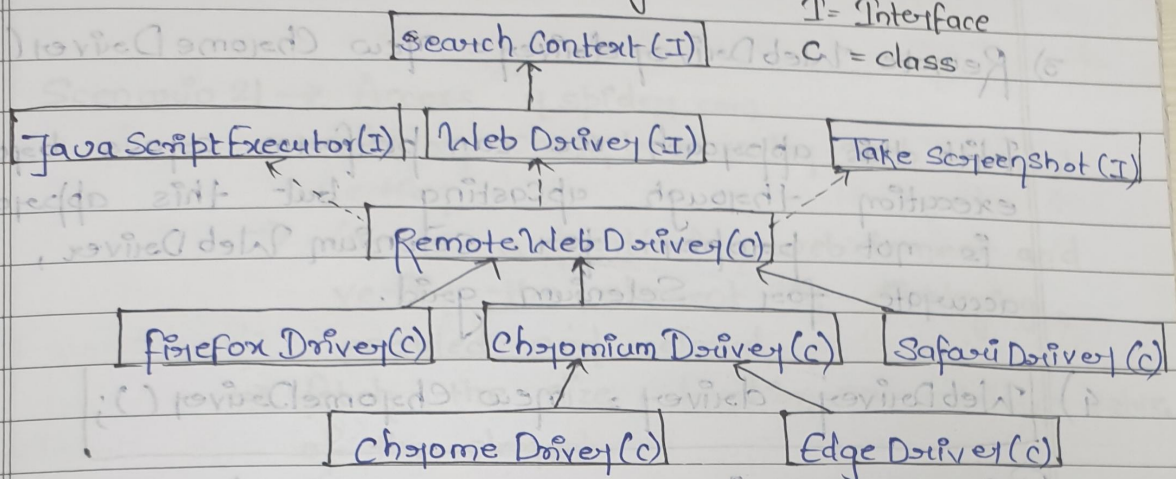
```
Thread.sleep(2000);
driver.findElement(By.xpath("//p[text()='2. would you like to receive recommendations for similar products?']/following-sibling::main/div[2]/input")).click();
Thread.sleep(2000);

driver.findElement(By.xpath("//p[text()='3. would you like to receive any customer assistance for this product?']/following-sibling::main/div[2]/input")).click();

Thread.sleep(2000);
driver.close();
```

Date: / /

* Selenium Web driver Coding level Architecture →



⇒ Accurate approach to perform cross browser execution in Selenium Web driver →

1) ChromeDriver driver = new ChromeDriver();

With this approach we can access only chrome Browser, other browsers we cannot access. That'y it is not the proper approach.

2) ChromiumDriver driver = new ChromeDriver();

This approach is only applicable for chrome and Edge by performing upcasting but for the other browsers this approach is not proper, that'y for cross browser execution this approach is not accurate.

Date: / /

3) RemoteWebDriver driver = new ChromeDriver();

With this approach we can perform cross browser execution through upcasting but this approach is not proper for selenium WebDriver, it is accurate for Selenium grid.

4) WebDriver driver = new ChromeDriver();

This approach is the best approach to perform cross browser execution incase of selenium WebDriver tool through upcasting.

5) SearchContext driver = new ChromeDriver();

With the help of this approach we can launch multiple browsers but we cannot access any web application. That's this approach is not accurate.

✳ Class is non primitive data type. That's why it's showing type name is discouraged while creating class

papergrid

✳ Why main method is static?

Date: / /

✳ Scenario 21 → Access 'qspider.com'
→ click on check box under web element section
→ Locate a particular web element and verify the element is displaying, enabled and selected.
→ fetch the size, location and attribute value of the element.

```
WebDriver driver = new ChromeDriver();
driver.get("Demoapps.qspider.com");
Thread.sleep(3000);
driver.findElement(By.xpath("//section[text()='checkBox']")).click();
Thread.sleep(3000);
WebElement element = driver.findElement(By.xpath("//input[@id='domain1']"));
Thread.sleep(3000);
sop("element is visible:"+element.isDisplayed());
sop("element is enabled:"+element.isEnabled());
sop("element is selected:"+element.isSelected());
sop("element size is:"+element.getSize());
sop("element location is:"+element.getLocation());
sop("attribute value is:"+element.getAttribute("id"));
```

papergrid

Date: / /

SECTION-2

19/2/24

* KeyBoard Stroke Handling →

→ KeyBoard Stroke Handling can be done in 2 ways-

a) Keys class
b) Robot class

a) Keys class → It is an inbuilt class present in selenium package.

→ Keys class you have to declare inside sendkeys().
→ Keys class cannot perform operations with "Alphabets"
→ Keys class cannot access desktop application.

Note → To solve the drawbacks we can go for Robot class.

b) Robot class → Robot class is present in "JAVA.AWT" Package.

→ To use robot class we need to create an object of the robot class.
→ Robot class can perform operations with "Alphabets."
→ Robot class can access desktop application also.

* Scenario 22 → Lauch ChromeBrowser
→ Access orangehrmlive.com
→ Maximise the window
→ perform the login operation with the help of keys class.

```
import org.openqa.selenium.Keys;
WebDriver driver = new ChromeDriver();
driver.get("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login");
drive.manage().window().maximize();
Thread.sleep(3000);
driver.findElement(By.xpath("//input[@placeholder='Username']")).
SendKeys("Admin", Keys.TAB, "admin123",
Keys.ENTER).Keys.CONTROL);
```

* Scenario 23 → Launch ChromeBrowser
→ Access demoappqspiders
→ choose the WebElement and click on keyboard action.
→ Click on virtual keyboard
→ provide email and copy the same email and provide into password
→ perform submit operation (using Robot class).

```
import java.awt.Robot;
import java.awt.event.KeyEvent;
WebDriver driver = new ChromeDriver();
driver.get("demoappqspiders.com");
driver.manage().window().maximize();
Thread.sleep(2000);
driver.findElement(By.xpath("//section[text()='Web Elements']")).click();
Thread.sleep(2000);
driver.findElement(By.xpath("//section[text()='Keyboard Actions']")).click();
Thread.sleep(2000);
driver.findElement(By.xpath("//section[text()='Virtual keyboard']")).click();
Thread.sleep(2000)
driver.findElement(By.xpath("//input[@id='email']")).sendKeys("Abc@gmail.com");
Robot r = new Robot();
```

```
r.keyPress(KeyEvent.VK_CONTROL);
r.keyPress(KeyEvent.VK_A);

r.keyRelease(KeyEvent.VK_CONTROL);
r.keyRelease(KeyEvent.VK_A);

r.keyPress(KeyEvent.VK_CONTROL);
r.keyPress(KeyEvent.VK_C);

r.keyRelease(KeyEvent.VK_CONTROL);
r.keyRelease(KeyEvent.VK_C);
        Thread.sleep(2000);
r.keyPress(KeyEvent.VK_TAB);
r.keyRelease(KeyEvent.VK_TAB);
        Thread.sleep(2000);
r.keyPress(KeyEvent.VK_CONTROL);
r.keyPress(KeyEvent.VK_V);

r.keyRelease(KeyEvent.VK_CONTROL);
r.keyRelease(KeyEvent.VK_V);

r.keyPress(KeyEvent.VK_ENTER);
r.keyRelease(KeyEvent.VK_ENTER);
```

```
r.keyPress(KeyEvent.VK_CONTROL);
r.keyPress(KeyEvent.VK_A);

r.keyRelease(KeyEvent.VK_CONTROL);
r.keyRelease(KeyEvent.VK_A);

r.keyPress(KeyEvent.VK_CONTROL);
r.keyPress(KeyEvent.VK_C);

r.keyRelease(KeyEvent.VK_CONTROL);
r.keyRelease(KeyEvent.VK_C);
Thread.sleep(2000);
r.keyPress
r.keyRelease(KeyEvent.VK_TAB);
r.keyRelease(KeyEvent.VK_TAB);
Thread.sleep(2000);
r.keyPress(KeyEvent.VK_CONTROL);
r.keyPress(KeyEvent.VK_V);

r.keyRelease(KeyEvent.VK_CONTROL);
r.keyRelease(KeyEvent.VK_V);

r.keyPress(KeyEvent.VK_ENTER);
r.keyRelease(KeyEvent.VK_ENTER);
```

20/2/24

ACTION CLASS METHODS →

1) moveToElement (target WebElement) → With the help of this method we can move the mouse Hover to a particular web element.

Arg1

Syntax → act.moveToElement (element);

2) ContextClick() and contextClick (target WebElement) → With the help of this method we can perform right click operation on a particular web element present in the webpage.

Syntax → act.contextClick();

3) build() → build acts as a bridge in between two different mouse actions.

Arg1

Syntax → act.moveToElement (element).build().
contextClick();

4) clickAndHold() and clickAndHold (target WebElement) → With the help of this method we can perform clickAndHold operation on a particular element present in a web page.

Syntax → act.clickAndHold (target WebElement);

5) dragAndDrop (Src, target) → With the help of this method we can drag a particular web element to the destination

Syntax → act.dragAndDrop (Source, target);

Date: / /

20/2/24

ACTION CLASS METHODS →

1) moveToElement (target WebElement) → With the help of this method we can move the mouse Hover to a particular web element.

Arg↓

Syntax → act.moveToElement (element);

2) ContextClick() and contextClick (target WebElement) → With the help of this method we can perform right click operation on a particular web element present in the webpage.

Syntax → act.contextClick();

3) build() → build acts as a bridge in between two different mouse actions.

Arg↓

Syntax → act.moveToElement (element).build().contextClick();

4) clickAndHold() and clickAndHold (target WebElement) → With the help of this method we can perform clickAndHold operation on a particular element present in a web page.

Syntax → act.clickAndHold (target WebElement);

5) dragAndDrop (Src, target) → With the help of this method we can drag a particular web element to the destination.

Syntax → act.dragAndDrop (Source, target);

papergrid

Date: / /

6) dragAndDropBy() → With the help of this method we can perform slider handling operations in a web page.

Syntax → act.dragAndDropBy(source, x, y);

7) perform() → ~~With~~ Perform() is the most important method present in actions class without this method we cannot perform any mouse actions.

Syntax → act.perform();

8) scrollToElement() → (target WebElement) → With the help of this method we can perform scroll down operation to a particular web element.

Syntax → act.scrollToElement(target WebElement);

papergrid

Date: / /

* ACTIONS CLASS

→ Actions is an inbuilt class present in selenium package.
→ With the help of actions class multiple we can perform different types of mouse actions in a web page.

Note → To perform this mouse actions inbuild functions are present inside actions class.

* Scenario 24 → Launch Chrome Browser
→ Access demoapps.qspider.com
→ Click on web elements to collapse
→ click on mouse actions and select mouseHover
→ Enter password
→ click on the eye button to display the password
→ Right click on the information symbol

```
import org.openqa.selenium.interactions.Actions;

WebDriver driver = new ChromeDriver();
driver.get("demoapp.qspider");
driver.manage().window().maximize();
Thread.sleep(3000);

WebElement element = driver.
        findElement(By.xpath("//section[text()=
            'Web Elements']"));
Thread.sleep(3000);
Actions act = new Actions(driver);
```

```
Thread.sleep(3000);
act.moveToElement(element).click().perform();

Thread.sleep(3000);
WebElement mouseActions = driver.findElement(By.xpath
    ("findElement //section[text()='Mouse Actions']
    "));
Thread.sleep(3000);
act.moveToElement(mouseActions).click().perform();

Thread.sleep(3000);

driver.findElement(By.xpath("//section[text()='Mouse
                Hover']")).click();

Thread.sleep(3000);

driver.findElement(By.xpath("//input[@placeholder=
            'Enter password']")).sendKeys("hvhom");

Thread.sleep(3000);

WebElement eyeBtn = driver.findElement(By.xpath
    ("//div[@class='flex']/img[1]"));

Thread.sleep(3000);
act.click(eyeBtn).perform();
Thread.sleep(3000);

WebElement infoIcon=driver.findElement(By.xpath("//div[@
                class='flex']/img[2]"));
```

papergrid

Date: / /

```
Thread.sleep(3000);
act.contextClick(infoIcon).perform();
```

* Example for of flipkart by using 2 methods at a time -

```
WebDriver driver = new ChromeDriver();
driver.get("https://www.flipkart.com/");
driver.manaage().window().maximize();

Thread.sleep(2000);
Actions act = new Actions(driver);
Thread.sleep(2000);
act.moveToElement(element).contextClick().perform
                                          ();
```

* Scenario 25 → Launch chrome Browser
→ Access demoapps.qspider
→ Collapse this webElement section
→ click on mouse actions
→ select click and hold, and expand the circle by using mouse Actions.

```
WebDriver driver = new ChromeDriver();
driver.get("demoapps.qspiders.com");
driver.manage().window().maximize();
Thread.sleep(2000);
driver.findElement(By.xpath("//section[text()='Web
                    Elements']")).click();
```

```
Thread.sleep(2000);
driver.findElement(By.xpath("//section[text()='Mouse Actions']")).click();
Thread.sleep(2000);
driver.findElement(By.xpath("//section[text()='click and Hold']")).click();
Thread.sleep(2000);
Actions act = new Actions(driver);
WebElement circle = driver.findElement(By.xpath("//div[@id='circle']"));
Thread.sleep(2000);
act.moveToElement(circle).clickAndHold().perform();
```

* Scenario 26 → Launch Chrome Browser
→ Access demoapp.qspider.com
→ Collapse Web Elements
→ click on mouse Actions
→ click on drag and drop
→ click on drag position
→ Drag the mobile items to mobile accessories and drag the laptop items to laptop accessories

```
WebDriver driver = new ChromeDriver();
driver.get("demoapps.qspider.com");
driver.manage().window().maximize();
Thread.sleep(2000);
driver.findElement(By.xpath("//section[text()='Web Element']")).click();
```

papergrid

Date: / /

```
Thread.sleep(2000);
driver.findElement(By.xpath("//section[text()='Mouse
                    Actions']")).click();
Thread.sleep(2000);
driver.findElement(By.xpath("//section[text()='Click and
                    Hold']")).click();
Thread.sleep(2000);
Actions act = new Actions(driver);
WebElement circle = driver.findElement(By.xpath("
                    //div[@id='circle']"));
Thread.sleep(2000);
act.moveToElement(circle).clickAndHold().perform();
```

* Scenario 26 → Launch Chrome Browser
→ Access demoapp.qspider.com
→ Collapse Web Elements
→ click on mouse Actions
→ click on drag and drop
→ click on drag position
→ Drag the mobile items to mobile accessories and drag the laptop items to laptop accessories

```
WebDriver driver = new ChromeDriver();
driver.get("demoapps.qspider.com");
driver.manage().window().maximize();
Thread.sleep(2000);
driver.findElement(By.xpath("//section[text()=
                    'Web Element']")).click();
```

```
Thread.sleep(2000);
driver.findElement(By.xpath("//section[text()='Mouse Actions']")).click();
Thread.sleep(2000);

driver.findElement(By.xpath("//section[text()='Drag and drop']")).click();

Thread.sleep(2000);
driver.findElement(By.xpath("//a[text()='Drag position']")).click();

Thread.sleep(2000);
WebElement mobCharger = driver.findElement(By.xpath
("//div[text()='Mobile Charger']"
));

Thread.sleep(2000);
WebElement mobDestinatn = driver.findElement(By.xpath
("//div[contains(@class,'drop-column m')]"));
Thread.sleep(2000);

Actions act = new Actions(driver);
act.dragAndDrop(mobCharger, mobDestination).perform();

Thread.sleep(2000);

WebElement laptopCharger = driver.findElement(By.
xpath("//div[text()='Laptop Charger']"));

Thread.sleep(2000);

WebElement laptopDestination = driver.findElement
(By.xpath("//div[text()='Laptop Accessories']
/.."));
```

Thread.sleep(2000);
act.dragAndDrop(laptopcharger, laptopDestination).perform();

* Scenario 27 → Launch Chrome Browser
→ Access to flipkart
→ Search for Iphone 11
→ Move the price pointer from left to right and right to left.

```
WebDriver driver = new ChromeDriver();
driver.get("flipkart.com");
driver.manage().window().maximize();
Thread.sleep(3000);
WebElement element = driver.findElement(By.xpath
                     ("//input[@name='q']"));
element.sendKeys("iphone 11");
element.submit();

Actions act = new Actions(driver);
Thread.sleep(3000);

WebElement leftPointer = driver.findElement(By.xpath
                         ("//div[@class='_3FdLqY'])[1]"));
act.dragAndDropBy(leftPointer, 50, 0).perform();

Thread.sleep(3000);
```

Thread.sleep(2000);
act.dragAndDrop(laptop charger, laptop Destination).perform();

* Scenario 27 → Launch Chrome Browser
→ Access to flipkart
→ Search for iphone 11
→ Move the price pointer from left to right and right to left.

WebDriver driver = new ChromeDriver();
driver.get("flipkart.url");
driver.manage().window().maximize();
Thread.sleep(3000);

WebElement element = driver.findElement(By.xpath("//input[@name='q']"));
element.sendKeys("iphone 11");
element.submit();

Actions act = new Actions(driver);
Thread.sleep(3000);

WebElement leftPointer = driver.findElement(By.xpath("(//div[@class='_3FdLqY'])[1]"));
act.dragAndDropBy(leftPointer, 50, 0).perform();

Thread.sleep(3000);

```
WebElement sightPointer = driver.findElement
    (By.xpath("(//div[@class='_3FdLq4'])[2]"));

act.dragAndDropBy(sightPointer, -30, 0).perform();
```

* Scroll Down Handling →

→ Scroll down handling can be done in 3 ways –
a) with the help of actions class
b) with the help of keyboard shoke
c) with the help of Java Script Executor

⇒ With the help of mouse actions →

In this case we need to create an object of actions class and with the help of that we need to call a method named as scrollToElement().

```
Actions act = new Actions(driver);
act.scrollToElement(target);
```

* Scenario 28 → Lauch chrome Browser
→ Access flipkart
→ Search for any product
→ In the result page, scroll down to the last product and click on that product.
→ close the window

```
WebDriver driver = new ChromeDriver();
driver.get("https://www.flipkart.com/");
driver.manage().window().maximize();
Thread.sleep(3000);
WebElement element = driver.findElement(By.x
                    path("//input[@name='q']"));
element.sendKeys("i phone 11");
element.submit();
Thread.sleep(2000);
WebElement target = driver.findElement(By.
x path("//div[@class='_4rR01T"]
                                [24]"));
Actions act = new Actions(driver);
act.scrollToElement(target).perform();
Thread.sleep(2000);
act.moveToElement(target).click().perform();
Thread.sleep(2000);
driver.quit();
```

⇒ With the help of keyboard stroke →

In this case we have to go for robot class and we have to select a function called as "page down".

Syntax → Robot r = new Robot();
r.keyPress(KeyEvent.VK_PAGE_DOWN);
r.keyRelease(KeyEvent.VK_PAGE_DOWN);

```
Robot r = new Robot();
for(int i=1; i<=12; i++)
```

```
{
    a. keyPress(KEYEVENT.VK-PAGE-DOWN);
    a. keyRelease(KeyEvent.VK-PAGE-DOWN);
       Thread.sleep(2000);
}
```

⇒ With the help of Java Script Executor →

→ It is an interface present in selenium package.
→ With the help of this Java script executor interface we can perform scroll down operation in a webpage.

⇒ Steps to use Java script Executor →

a) First we have to downcast Java script executor (I) in our script.
Syntax → JavaScript Executor jse = (JavaScriptExecutor) driver;

b) We have to use a method called as execute Script().
Syntax → execute Script("JS syntax");

Note → This Java Script syntax we need to copy from front end console. Steps →

1) Right click on the web page
2) Click on inspect
3) Select console (beside elements)
4) Provide Java Script syntax and press enter.

papergrid
Date: / /

```
    {
    ac.keyPress(KeyEvent.VK_PAGE_DOWN);
    ac.keyRelease(KeyEvent.VK_PAGE_DOWN);
    Thread.sleep(2000);
    }
```

⇒ With the help of Java Script Executor →

→ It is an interface present in selenium package.
→ With the help of this Java script executor interface we can perform scroll down operation in a webpage.

⇒ Steps to use Java script Executor →

a) First we have to downcast Java script executor (I) in our script.
Syntax → JavascriptExecutor jse = (JavascriptExecutor) driver;

b) We have to use a method called as execute Script().
Syntax → executeScript("js syntax");

Note → This Java Script syntax we need to copy from front end console. Steps →

1) Right click on the web page
2) Click on inspect
3) Select console (beside elements)
4) Provide Java Script syntax and press enter.

Date: / /

```
WebDriver driver = new ChromeDriver();
driver.get("flipkart url");
driver.manage().window().maximize();
Thread.sleep(2000);
WebElement element = driver.findElement(By.xpath
    ("//input[@name='q.']"));
element.sendKeys("iphone 11");
element.submit();

JavascriptExecutor jse = (JavascriptExecutor) driver;
jse.executeScript("window.scrollBy(0,4000);");
```

* Scenario 29 → Launch chrome Browser
→ Access demoappsqspider.com
→ provide name, email id & password with the help of javascript and click on register button.

```
WebDriver driver = new ChromeDriver();
driver.get("demoapps.qspider.com");
driver.manage().window().maximize();
Thread.sleep(2000);
JavascriptExecutor jse = (JavascriptExecutor) driver;

jse.executeScript("document.querySelector(\"#name
    \").value = \"ani\";");

Thread.sleep(2000);
```

```
jse.executeScript("document.querySelector(\"#email\").value=\"abc@gmail.com\";");
Thread.sleep(2000);
jse.executeScript("document.querySelector(\"#password\").value=\"welcome\";");
Thread.sleep(2000);
jse.executeScript("document.querySelector(.....).click();");
```

* Screenshot Handling

→ To handle the screenshot or to capture a screenshot we have to use an interface called as "TakesScreenshot"

⇒ steps to capture screenshot →

→ Downcast takes screenshot interface in our script.
Syntax → TakesScreenshot ts=(TakesScreenshot) driver;

→ We have to capture screenshot in file format with the help of a method called as getScreenshotAs().

syntax → File src=ts.getScreenshotAs(OutputType.F

papergrid

Date: / /

→ We need to create an object of "file" class and we will provide the path of the screenshot folder.

Syntax → File dest = new File(".\\screenshotfolder\\" + "ss1.png");

".XX ss Folder Name XX" + "ss name.png"

→ We need to copy the screenshots to the destination folder with the help of a method named as copy(). which is present in "files" class.

* Scenario 30 → Access flipkart
→ Try to search for iphone with wrong xpath.
→ Capture the screenshot of the failed script.

```
import java.io.File;
import org.openqa.selenium.By;
import org.openqa.selenium.OutputType;
import org.openqa.selenium.TakesScreenshot;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import com.google.common.io.Files;

WebDriver driver = new ChromeDriver();
driver.get("https://www.flipkart.com/");
```

papergrid
Date: / /

```
driver.manage().window().maximize();
TakesScreenshot ts = (TakesScreenshot) driver;
File src = ts.getScreenshotAs (OutputType.FILE);
File dest = new File (".\\ScreenShotFolder\\" +
                      "ss1.png");
Files.copy (src, dest);
driver.findElement (By.xpath ("hqjhvj")).click;
```

* DropDown Handling →

→ To handle a dropdown we have three ways →

a) With the help of select class.
b) With the help of keyboard stroke
c) Normal findElement approach or normal inspect approach [applicable for static dropdown].

a) Select class approach →

→ It is an inbuilt class present in selenium package.
→ This class is applicable only if select tag is present
→ In this class different types of functions or methods are present to perform selection and deselection operations.

⇒ Selection Operation functions →

a) SelectByIndex (Index no);
b) selectByValue ("Value Attribute data");
c) select By Visible Text ("Visible Text");

papergrid

Date: / /

```
driver.manage().window().maximize();
TakesScreenshot ts = (TakesScreenshot) driver;
File src = ts.getScreenshotAs (OutputType.FILE);
File dest = new File (".\\screenshot folder\\" +
                      "ss1.png");
files.copy (src, dest);
driver.findElement (By.xpath ("hqjhvj")).click;
```

⇒ a) b) c)

NOTE

* DropDown Handling →

→ To handle a dropdown we have three ways →

a) With the help of select class.
b) With the help of keyboard stroke
c) Normal findElement approach or normal inspecting approach [applicable for static dropdown].

a) Select class approach →

→ It is an inbuilt class present in selenium package.
→ This class is applicable only if select tag is present
→ In this class different types of functions or methods are present to perform selection and deselection operations.

⇒ Selection Operatn functions →

a) SelectByIndex (Index no) ;
b) select By Value (" Value Attribute data");
c) select By Visible Text (" Visible Text");

papergrid

Date: / /

⇒ Deselection functions →

a) deselectByValue("Value Attribute data");
b) deselectByIndex(index no);
c) deselectByVisibleText("Visible Text");

NOTE → To identify a dropdown is multiselected or not we have to use a method is called as "isMultiple().

* Scenario 3 → Launch chromeBrowser.
→ Access qspiders.com
→ click on dropdown
→ In the gender dropdown select any one option.
→ In the country dropdown select India.
→ close the window.

```
import org.openqa.selenium.support.ui.Select;

WebDriver driver = new ChromeDriver();
driver.get("qspiders.com");
driver.manage().window().maximize();
Thread.sleep(3000);
driver.findElement(By.xpath("//section[text()='Drop down']")).click();

Thread.sleep(3000);
WebElement gender = driver.findElement(By.xpath("//select[@id='select 2']"));
Thread.sleep(3000);
Select s= new Select(gender);
```

```
s.selectByVisibleText("female");
SOP(s.isMultiple());
Thread.sleep(3000);

WebElement country = driver.findElement(By.xpath("
                    //select[@id='select3']"));

Select s1 = new Select(country);
s1.selectByVisibleText("India");
driver.close();
```

* Scenario 32 → Launch Chrome Browser
→ Access qspiders.com
→ Click on dropdown
→ click on multiselect
→ select first 3 options from country dropdown
→ check the dropdown is multiselected or not.
→ Deselect the 3rd country
→ Deselect all the countries.

```
WebDriver driver = new ChromeDriver();
driver.get("qspiders.com");
driver.manage().window().maximize();
Thread.sleep(3000);
driver.findElement(By.xpath("//section[text()='Drop
                    down']")).click();
Thread.sleep(3000);
```

Date: / /

```
driver.findElement(By.xpath("//select a[text()='Multi select']")).click();
Thread.sleep(3000);

WebElement country = driver.findElement(By.xpath("// select [[@id='select-multiple-native']][1]"));
Thread.sleep(3000);

Select s= new Select(country);
s.selectByIndex(0);
s.selectByIndex(1);
s.selectByIndex(2);

sop("dropdown is multiselected :"+ s.isMultiple());
Thread.sleep(3000);
s.deselectByIndex(2);

Thread.sleep(3000);
s.deselectAll();
```

papergrid

Date: / /

⇒ Normal find Element approach with dynamic dropdown →

→ If select tag is not there in the UI in that case we can go with this approach.

→ If select tag is not there and dropdown is dynamic in that case we have to go for screen debugger concept.

→ To make a screen pause you have to follow few steps :-

a) Right click on the web page.
b) click on sources and press function(fn + f8) or
c) only f8.

* Scenario 33 → Launch chrome Browser
→ Access flipkart
→ In the home page move to electronics and select electronics GST store.

```
WebDriver driver = new ChromeDriver();
driver.get("flipkart url");
driver.manage().window().maximize();
WebElement category = driver.findElement(By.xpath("//
        span[text() = 'Electronics']"));

Actions act = new Actions(driver);
act.moveToElement(category).perform();
Thread.sleep(2000);
driver.findElement(By.xpath("//a[text()='Electronics
        GST store']")).click();
```