**Selenium:**

* Selenium automates browsers. That’s it.
* Which is developed by Jason Huggins in 2004. He developed a JavaScript program called the JavaScriptTestRunner that automated web application testing. This program was renamed Selenium in 2004.
* Previously, there was a product called Mercury. Which is also an automation tool developed by HP Group. But it was expensive.
* The name 'Selenium' came from an email in which Huggins mocked a competitor named 'Mercury', saying that one can cure mercury poisoning by taking selenium supplements.

**Launching a browser:**

* To launch the browser, we should need two lines of code.
* setProperty method enables QAs to set the properties for the desired browser for test automation.
* The system setProperty method has two attributes – “property Name” and “value.” The property Name represents the name of the browser-specific driver, and the value points to the path of that browser driver.

System.setProperty(“webdriver.chrome.driver”,”Path”);

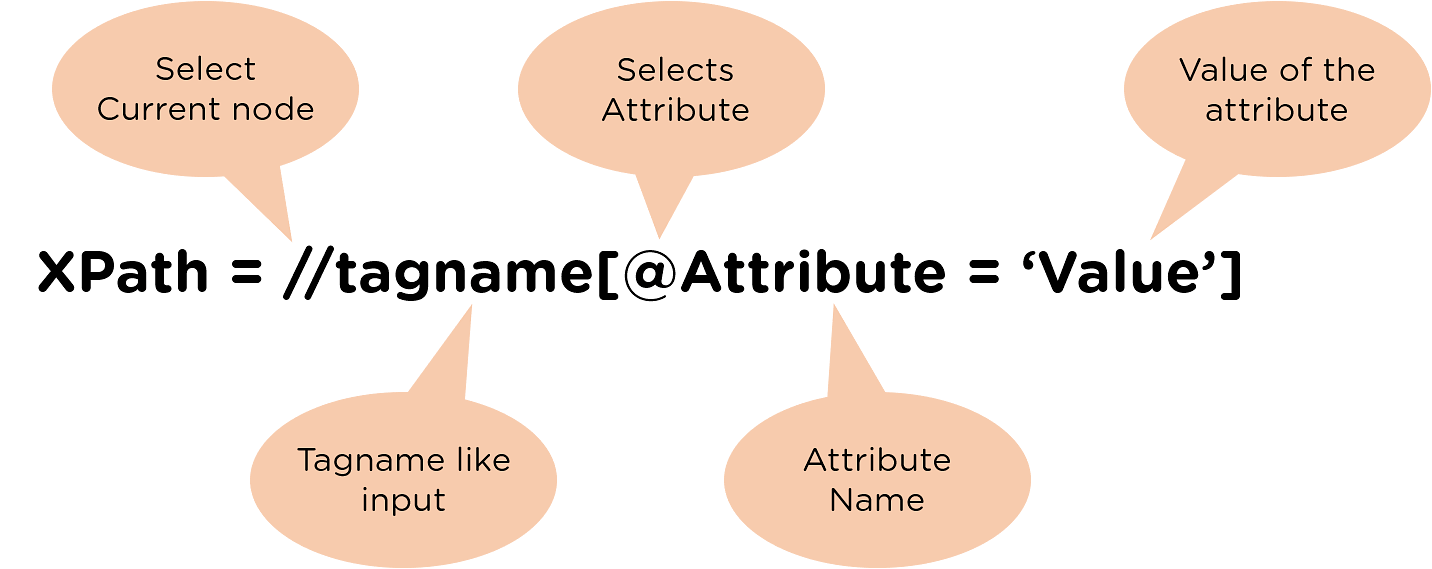
ChromeDriver driver = new ChromeDriver();

**Locators:**

* Locators is a way to identify the element on a web page. This element we called as Web Element.
* Generally, it is passed as an argument to the find element method.
* There are eight types of locators.

1. Id - Locates elements whose ID attribute matches the search value
2. Name - Locates elements whose NAME attribute matches the search value
3. class name - Locates elements whose class name contains the search value (compound class names are not permitted)
4. Link Text - Locates anchor elements whose visible text matches the search value
5. Partial Link Text - Locates anchor elements whose visible text contains the search value. If multiple elements are matching, only the first one will be selected.
6. XPath - Locates elements matching an XPath expression
7. CSS selector - Locates elements matching a CSS selector
8. tag Name - Locates elements whose tag name matches the search value.

* **XPath:** 
  + XPath is nothing but XML path expression. Here we consider HTML doc as XML Doc.
  + There are two types of XPath. They are: Absolute and Relative XPath.
  + Absolute XPath: The XPath starts with single slash (/) and the element is selected from the root node.
  + Relative XPath: The XPath starts with double slash (//) and the element is selected from anywhere from the web page.
  + **Syntax of XPath**: //\*[text(),’Cricket’]



**WebElement:**

* Web element is nothing but a HTML element present in a web page.
* We can fetch web element using method findlelement.
* driver. findElement(By.id(“FirstName”)); 🡪 This piece of code will return a webElement.

**Handling Inputs:**

* There are several commands that can be executed on web element based on its state.

1. .sendKeys() : This command will help us to pass the value to the editable text box in a web page.

This command also used to append the text.

1. Keyboard TAB : Example - .sendKeys(“person”,Keys.TAB). This argument will press TAB key automatically after entering the text ‘person’.
2. .getAttribute(“name”) : This command will return the value of the attribute ‘name’ from the desired HTML tag.
3. Clear() : This method will clear the text.
4. isEnabled() : This method helps us to confirm whether the field is editable.
5. .getAttribute(“readonly”) : This method will return true if the element contains readonly attribute.
6. driver.quit() : This method will close all the tabs.
7. driver.close() : This method will close the current tab.

**Driver Navigation method:**

* driver.navigate().to(“Paste the URL here”) – This method will navigate to particular URL without waiting for page load. Particularly it maintains history. Due to that reason, it can additionally do further operation on this method. Those methods are refresh(), back(), forward().
* driver.get(“Paste the URL here”) -- This method will navigate directly to the URL with waiting for page load.

**Note: findElement method will return webElement interface. To capture this webElement we create reference for it. E.g., WebElement element = driver.findElement(By.id(“fname”)); . Here, ‘element’ variable is the reference.**

**Handling Button:**

* We can handle button in several ways. We are going to discuss three main ways.
* First thing is to locate the coordinates of the button. Refer below written code to locate the coordinates.

//get the coordinates of x and y

WebElement ele = driver.findElement(By.id("position"));

Point point = ele.getLocation();

int x = point.getX(); // return int

int y = point.getY(); // return int

System.out.println("Value of x-coordinates => "+ x);

System.out.println("Value of y-coordinates => "+ y);

System.out.println("point object also return the x and y coordinates : " + point);

**NOTE:** getLocation method will return Point class. We can also retrieve the coordinates using getPoint method. To know about this method, just scroll down and go through the last line of code in height and width section.

* Second is to identify the color of the button. Refer below code to identify the color.

//get color of the button

String backgroundColor = driver.findElement(By.id("color")).getCssValue("background-color");

String color = driver.findElement(By.id("color")).getCssValue("color"); System.out.println("Background color : "+ backgroundColor);

System.out.println("Color of the text : "+ color);

**NOTE:** getCssValue will return string value of the CSS property mentioned inside the double quotes. In the above case, it will return the color value in the form of rgb(12,23,54,1).

* Finally, we will find the height and width of the button.

//get width and height of the button

Rectangle rect = driver.findElement(By.id("property")).getRect();

Dimension dim = rect.getDimension();

int width = dim.getWidth();

int height = dim.getHeight();

System.out.println("Width : "+ width + " Height : "+ height + " " + rect.getHeight());

System.out.println(dim); // This line also return same width and height

System.out.println("getPoint funtion will fetch the position of the button : "+ rect.getPoint());

**NOTE:** getRect method will return Rectangle class. getDimension will return Dimension class. Another important point to note that we can straight away get the height and width in the form of string. Without using dimension class, we can directly fetch the width and height of the button with the help of rect.getWidth and rect.getHeight method.

**Screenshot:**

* This method will help us to capture screenshot. We can capture snaps in three different ways.

1. Capture Entire Page

Steps to Approach:

* 1. Take the screenshot and store it in a file
  2. Create a dest file and screenshot in our local path
  3. Copy and paste in our dest file

//taking screenshot of entire page

File src = driver.getScreenshotAs(OutputType.FILE);

File dest = new File("./snaps/img1.png");

FileHandler.copy(src, dest);

1. Capture Particular Element

Steps to Approach:

* 1. Take the screenshot and store it in a file
  2. Create a dest file and screenshot in our local path
  3. Copy and paste in our dest file

//taking screenshot of particular element

WebElement ele = driver.findElement(By.id("color"));

File src1 = ele.getScreenshotAs(OutputType.FILE);

File dest1 = new File("./snaps/img2.png");

FileHandler.copy(src1, dest1);

1. Capture Particular section from HTML DOM Structure (combination of fields inside a section)

Steps to Approach:

* 1. Take the screenshot and store it in a file
  2. Create a dest file and screenshot in our local path
  3. Copy and paste in our dest file

//taking screenshot of particular section

WebElement ele1 = driver.findElement(By.xpath("//div[@class='column is-3-desktop is-4-tablet']//div[@class='card-content']"));

File src2 = ele1.getScreenshotAs(OutputType.FILE);

File dest2 = new File("./snaps/img3.png");

FileHandler.copy(src2, dest2);

**Alert:**

* Alert is nothing but a popup or a message box which blocks our screen.
* Alert is an Interface.
* Alert is not part of the HTML DOM structure. Which comes under JavaScript.
* Whenever a popup arises, our first duty is to respond to the alert then only we can proceed to automate further.
* There are different types of alerts. Those are:

1. Simple Alert: Simple Alert consist of only one button.
2. Confirm Alert: It contains two buttons. Those are ok and cancel.
3. Prompt Alert: Alert with an editable textbox field.

* There are fours ways to handle an alert.

1. Accept:

driver.switchTo().alert().accept();

Alert alert = driver.switchTo().alert();

1. Dismiss:

alert.dismiss();

1. getText():

alert.getText();

1. sendKeys():

alert.sendKeys();

**Handling Frames:**

* Frames means a HTML document embedded in parent HTML document. It is similar to the concepts of nested.
* Five ways to switch frame but it only differs by the argument we send. Those are:

1. frame(int index) : Passing index as an argument.
2. frame(String name) : Passing attribute as an argument.
3. frame(WebElement frameElement) : Passing webElement as argument.
4. parentFrame() : This method will switch to immediate parent HTML document.
5. defaultContent() : This method can return the focus to main page.

* **Interview Question:** Overloading concepts widely used in frame concept.

**Maximizing Window:**

* This method will maximize the window. (Method Chaining)

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

(OR)

Options opt = driver.manage(); --> return interface

Window win = opt.window(); --> return interface

win.maximize();

**Implicit Wait:**

* Implicit wait is to tell the driver to wait for a certain amount of time when it tries to find element or elements if they are not available immediately.
* The default timing is 0.
* Interesting point to note, findElement will wait for 500ms (milli seconds).
* Time Unit is an Enum type. To use time unit, we should import java.util.concurrent package.

//add implicit wait

driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

**Drag & Drop:**

* The title itself defines that an action performed using a mouse to move a particular web element from one place to another place.
* Should import selenium.interactions.actions package.
* Action is a class.

//Drag and Drop operation

WebElement source = driver.findElement(By.id("draggable"));

WebElement dest = driver.findElement(By.id("droppable"));

Actions builder = new Actions(driver);

builder.dragAndDrop(source, dest).perform(); //don't chain build() because inbuilt perform has it

**Drag & Drop By:**

* This is same as above concept but only differ by argument.
* Passing coordinates to the desired position instead of mentioning destination web element.\

//Drag and Drop By operation

Actions builder =new Actions(driver);

WebElement source = driver.findElement(By.id("sample-box"));

int x = source.getLocation().getX();

int y = source.getLocation().getY();

builder.dragAndDropBy(source,500,250).perform();

System.out.println("Coordinates of source element : "+ x +" "+ y);

**Right Click:**

* This is also an action performed using mouse.

//Right click operation

WebElement button = driver.findElementByXPath("//span[@class='context-menu-one btn btn-neutral']");

Actions builder = new Actions(driver);

builder.contextClick(button).perform();

**Dropdown:**

* Dropdown is something a list of options.
* We can able to select either multiple elements or single element according to the dropdown.
* Generally, select tag is used to create drop down list.
* Dropdown can be handled in three ways:
  1. Select by index

WebElement Fruits = driver.findElement(By.id("fruits"));

Select myFruits = new Select(Fruits);

myFruits.selectByValue("3");

* 1. Select by value (based on value of the attribute available in html page)

WebElement myCountry = driver.findElementById("country");

Select country = new Select(myCountry);

country.selectByValue("Chile");

* 1. Select by visible text

Select Heros = new Select(heros);

Heros.selectByVisibleText("Doc Savage");

* Heros.getFirstSelectedOption().getText()
* Heros.isMultiple()
* Heros.getAllSelectedOptions();
* Heros.deselectByIndex(2);
* List <WebElement> selectedOptions = Heros.getAllSelectedOptions();

for(WebElement options : selectedOptions)

{

System.out.println(options.getText());

}

**getAttribute vs getText:**

* getText will retrieve visible text.
* getAttribute will retrieve the value of the attribute from an element.

**Close vs Quit:**

* Close method closes only the current focused window but it never kills the driver memory.
* Quit method will close entire browser and ends the driver session.
* Quit method also destroy the entire object (leave memory).
* To kill the memory, first we need to open cmd prompt and type below command.

taskkill /f /im chromedriver.exe (Corresponding driver)

im refers to image name.

**Validation:**

* Validation means checking whether the UI elements are rendered on the web pages correctly or not.
* There are three validation methods.

1. isDisplayed() : Verifies whether the element is displayed or not. Check only 'disabled' attribute in tag.
2. isEnabled() : check whether the element is interactable or not. Check only 'disabled' attribute in tag.
3. isSelected() : check whether the element is already selected or not. (Preferably used for radio and checkbox element). Check only 'checked' attribute in tag

**Submit Function:**

* Submit method will work similar as when we hit enter after filling a form.
* After entering all required details in form, we proceed to hit enter key. In same way we have method to submit the form using the method submit().
* Must have a form tag to work this concept.

WebElement Submit = driver.findElementByXPath("//button[@type='submit']");

Submit.submit();

**Active Element:**

* activeElement() - Switches to the element that currently has focus within the document currently.

WebElement username = driver.switchTo().activeElement();

username.sendKeys("rkpranav",Keys.TAB,"Password",Keys.ENTER);

* Note: There was a question in interview that whether can we pass value to a textbox without using findElement method. The answer for this question is Active Element.

**Window Handling:**

* It is a unique identifier which hold the address of the window.
* get.windowhandle(): This method helps to get the window handle of the current window. It will return string.
* get.windowhandles(): This method helps to get the handles of all the windows opened. It will return set of string. Set interface is used because set will not accept the duplicate.
* To view all window handle, we usually get all window handle element in set and convert it to List interface. By that way, we can use the get() method. Because get store the data in order.
* List interface need util.list package.

//Switching from set to list

Set<String> windows = driver.getWindowHandles();

List<String> list = new ArrayList<String>(windows);

driver.switchTo().window(list.get(1));

**Mouse Hover:**

* Simple meaning of hover is moving to and fro. If we do same action with mouse then it becomes mouse hover.
* Proper definition is basically an action where a user places a mouse over a designated area like a hyperlink. It can cause some event to get triggered.
* When we mention action then obviously, we are going to use action class.
* Another particular method will do the same work. That is, moveByOffset(x,y).
* Two different ways to implement this method.

1. moveToElement(webElement,x,y)
2. moveToElement(webElement)

Actions builder = new Actions(driver);

builder.moveToElement(Fashoin).perform();