Selenium notes:

* basic java code to run test cases in selenium:

```
//Code to a program
    import org.openqa.selenium.webdriver;
    import org.openqa.selenium.chrome.chromeDriver;

    public class Demo {
        public static void main(String[] args) {

                System.property("webdriver.chromeDriver", "C:\\..\chromeDriver.exe");
                WebDriver driver = new ChromeDriver();
                driver.get("http://google.com");
                String msg = driver.getTittle();
                System.out.println("Tittle: " + msg);
                }
        }
```

***Not important Points on Selenium Webdriver:**

1. Every object may have ID, className and name – XPath and CSS are preferred locators used worldwide.

2. Alpha numeric id may vary on every refresh – check before using it.
   For e.g. – id = "u_901" is not a valid reliable source to fetch the element for automation.

3. Confirm the link object with anchor tag-
   for e.g. - <a href = "http://www.awful-valentine.com"> link Text </a>.

4. Concept of ***compound classes*** is not permitted while automating through selenium webdriver classes. i.e. for <Tagname class= "*class1 class2 class3*"> </TagName> we cannot ***findElement***(***By.className***("class1 class2 class3")) is not allowed.

5. With some layers of abstraction, Selenium scans from the Top-Left and the picks the first ones that best suits the case. For e.g. if one class name is same for both the element and client code is automated based on class locator then the controller is shifted to first one and the second one will be left unseen by the controller of selenium.

6. Idk about this for sure but seems like double quotes inside double quotes are not valid as URI, URL and File Paths in Java as we cannot provide \" into paths and locations.

7. XPath / CSS can be defined in ***n*** number of ways it might have same syntax but can have variations as the attribute changes.

8. ***Firepath*** plugin from ***firefox v55*** or below is deprecated because it's thought to be way ahead of its time. Even chrome was feeling the shame.

9. XPath can be of any form but if its something like, html/body/div[@class = "className"] then its not reliable. The thing is that anything XPath staring from

Selenium notes:

10. To validate the XPath and CSS in browser we need to open console and from there type ***$x("xpath")*** to validate the XPath and type ***$("css") works in firefox and IE but use $$("") works for chrome and others as*** to validate the CSS for a particular.

11. ***//<Tag-name>[@<Attribute> = '<value>'] and //*[@<Attribute> = '<value>']*** is the syntax to generate the XPath for a website.

12. ***<Tag-name>[<Attribute> = '<value>']*** or ***[<Attribute> = '<value>']*** or ***<Tag-name>#<id-value>*** or ***<Tag-name>.<className>*** is the syntax to generate the CSS for a website.

Selenium notes:

Date: Sunday, June 14, 2020.

Regular Expression:
Sometimes values the values keeps on changing and sometimes it's the only value that keeps on changing. For e.g. - <input type="text" name = "username1234"/> and the assume the name keeps changing.
For that how will we automate now?

We will use Regular Expression with ***Regular expression*** –
//TagName[contains(@attribute, 'value')] – syntax of XPath for Regular Expressions.

It works as the driver is sent a command by selenium to check for a given tag-name is there any given attribute that contains the given if not wholly then partially(as a subtext) if something matches then it clicks that value otherwise returns an error.

For e.g. if we put regular expression. input[contains(@name, 'username')]. According to this example it will check the input tag with attribute name and check if the subtext matches that of the text in html. If yes, then it performs the command on the given element.

//TagName[attribute *= 'value'] – syntax of CSS for Regular expressions.
For e.g. Input[type *= 'username']

Quiz: which locator uses the contains keyword in the regular expression?
Ans: XPath.

Code.

```
Public class Elements {
public static void main(String[] args) {
    System.SetProperty("webdriver.chrome.driver","");
    WebDriver driver = new ChromeDriver();
    driver.get("http://www.rediff.com");
    driver.findElements(By.cssSelector("a[tittle *= 'Sign in']")).click();
    driver.findElements(By.xpath("//input[@id = 'login1']")).sendKeys("thisisit");
    diver.findElements(By.cssSelector("input#password")).sendKeys("thisisit");
    driver.findElements(By.xpath("//input[contains(@name, 'proceed')]")).click();
    driver.close();
}
    }
```

Selenium notes:

Date: Tuesday, June 16, 2020.

Few website: practice/ RahulShettyacademy.com
All the front end web automaton tools like ***Selenium***, ***protractor*** and ***Cypress***.

Misconception about **XPath** that you can write **XPath** based on ***Parent*** **-** ***Child*** relationship XPath. If your attribute for the said element keeps on changing, then you need to use Parent – child XPath for that element. First you find a parent of the element which is static (have unique attribute) and then write an XPath for the said element and then navigate to the element in the path.

```
<div class = “ist-cc”>
        <div>
                <input/>
        </div>
        <div>
                <div><input type = “text”/></div>
        </div>
</div>
```

Now to write an XPath for the 2nd input in the given in above html code we need to use the Parent – Child relationship XPath syntax.

//div[@class = ‘ist-cc’]/div[2]/div/input.

Tag[index] is the syntax for multiple tags under one parent.

Idea: use google and search for anything and visit any random website and then click on any random button. Is it possible using selenium?

Note: **Chropath** for *chrome* is what **FirePath** used to be for *firefox*.
Use Chropath to validate the XPath for elements.

Interview Questions:

QAClickAcademy.com

Q. Difference between Relative and Absolute XPath?
A. **Relative XPath (Preferred)**: The element of desired choice is directly traversed using XPath syntax. This will always work if the attributes are selected carefully.

Absolute XPath: Absolute XPath is when the element of desired choice is traversed from the parent nodes using slashes. As the child node depends on the parent nodes; it can cause issues in tests if directory structure changes.

Or

Absolute XPath: Absolute XPath starts from the root node of your choice – it doesn’t need to start from the root node.
It starts with a forward slash (/) E.g. - /html/head/body/table/tbody/tr/th

Selenium notes:

Relative XPath (Preferred): A relative XPath is one where the path starts from the node of your choice – it doesn't need to start from the root node. It starts with Double forward slashes (//).
Syntax: //table/tbody/tr/th

Other example :- .//*[@id = 'username'] The '.' At the start indicates that the processing will start from the current node with the @id-attribute-value equal to 'username'. Relative XPath takes more time in identifying the element as we specify the partial path not (exact path).
If there are multiple elements for the same path, it will select the first element.

Problem here: problem here is that if the current node is the already id = 'username' then what is the point of saying processing will start form the current node, when we already reached the desired node. And what kind of process they are talking about.

Q. How to traverse to sibling element using XPath?
A. Use syntax ***<tagname>[<index>]*** in which the index shows the which child node of the parent node is being located. For index equals N the nth- child is traversed.

For e.g.

```
    <ul id = "xyz">
        <li id = "this">First Element </li>
        <li id = "that">Second Element </li>
    </ul>
```

Syntax for traversing B/W siblings *(XPath to the element).following-sibling::<tagname>[<index>]*
To Traverse from first element to second element. //*[@id='this'].following-sibling::li[2]

Q. How to traverse back to Parent element from Child element XPath?
A. Syntax for traversing from parent to child
(XPath to the element) / parent::<parent-tag>
To traverse back to the ul parent from second element. //li[@id= 'that']/parent:ul

NOTE: In **CSS** you will **NOT** be able to **traverse back** from parent to child.

Q. How to identify element with Text based?

```
    For e.g. <div>
                <ul>
                    <li> selenium </li>
                    <li>Appium</li>
                </ul>
            </div>
```

Selenium notes:

A. ***//*[text() = ‘ selenium ‘].*** It is prescribed not to use these codes as they are considered as a hot coded one which basically means that texts can pretty much change and then it will be problematic if that happened that is why we need to stick to the general syntax mostly.

//Rahulonlinetutor@gmail.com.

//driver.findElement(By.xpath(“//*[@id=’tablist1-tab2’]/parent::ul”)).getAttribute(“role”);

CssSelector has an edge over XPath as it is way faster than XPath in terms of execution speed and its ability to go class under a class.

Syntax For xpath = //tagname[@attribute=’value’]
For cssSelector = tagname[attribute=’value’]
For Regular Expression xpath = //*[@attribute = ‘value’]
And for cssSelector = [attribute = ‘value’]

Code:

```
        public static void main(String[] args) {
        ChromeDriver driver = new WebDriver();

        driver.get("http://w3c.abcd.com");
        driver.findElement(By.cssSelector("[class = 'username']")).sendKeys(“this”);
        driver.findElement(By.cssSelector("input#password")).sendKeys("*****");
        driver.findElement(By.xpath("//button[@id='abcd']")).click();
        dirver.close();
}
```

Date: Wednesday, June 17, 2020.

Note: CSS does provide a way to use the compound classes in Selenium using a dot (.) operator which means <input class = “Support Every” > to identify the element we need to use either ***findElement(By .cssSelector(“input.Support.Every”));*** or ***findElement(By.cssSelector(“.Support.Every”));***

Whenever you are trying to find an element using locators use a logical way so that the class you gave is unique to the website you are automating otherwise test will fail. Hence in most applications use ID as ID will be unique and sometimes so unique that it even changes with every refresh of your webpage. So, you have to find a fine line in between to know what could and couldn’t be used as a unique attribute.

Selenium notes:

Date: Thursday, June 18, 2020.

Selenium Automation: Handling Static Dropdowns, Handling Dynamic Dropdowns, Handling Checkboxes, Handling Radio Buttons, Handling Text Buttons, Handling Alerts – Java Popups, Selenium Webdriver Form methods.

you can use any travel or hotel or movie/show booking sites for this as they have got these elements on their websites for e.g. www.SpiceJet.com , www.makemytrip.com, www.bookmyshow.com or www.goaibbo.com.

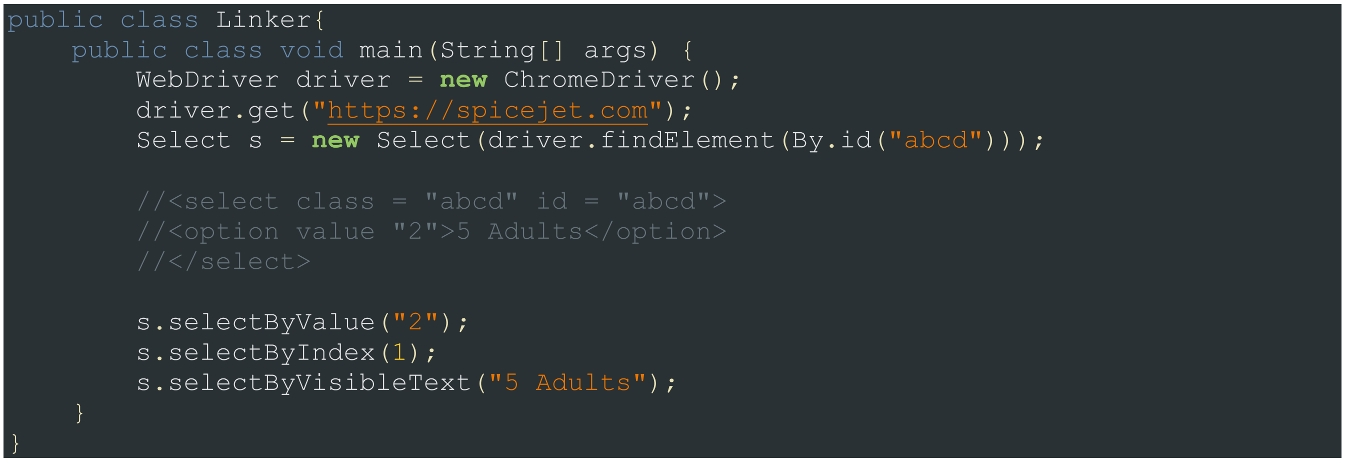
Static dropdown are the drop downs which have all the option preloaded. Which means that the options are already visible and we have to select from the list. It uses <Select> tag.

e.g. <Select class = “eofeds select valid”>

selenium have a class called select class.

Code:

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



```
public class Linker{
    public class void main(String[] args) {
        WebDriver driver = new ChromeDriver();
        driver.get("https://spicejet.com");
        Select s = new Select(driver.findElement(By.id("abcd")));

        //<select class = "abcd" id = "abcd">
        //<option value "2">5 Adults</option>
        //</select>

        s.selectByValue("2");
        s.selectByIndex(1);
        s.selectByVisibleText("5 Adults");
    }
}
```

Note: id should be preferred over the XPath and CSS as Id is more stable than any of these.

Also some times dropdowns can have absence of select tag. They are created using list in html and can be easily invoked using normal id, xpath and css.

In Java Thread.sleep(2000L) will introduce a 2 second sleep in program but its nor ideal to use it blindly.

Dynamic Dropdown are when the list elements are changed when the list is opened or dropdown is clicked and which in turn introduce new elements to html code which were absent initially. And thus we need to be careful when selecting the tag.

Selenium notes:

Sometimes it so happens that as we try to invoke an element using XPath and unfortunately it’s not unique as Selenium scans in document order: it invokes the first one it finds. But in order to fetch the second or nth element. We need to use syntax . ***(xpath)[index]*** starting from 0.
For. E.g. – (.//*[@id = ‘value’]).[2]

Which basically means that any element with id equals to the value find the k third instance where the condition hold true and processing will start from there.

Date: Friday, June 19, 2020.

Dynamic Dropdown continued….
Syntax:

```
WebDriver driver = new ChromeDriver();
dirver.findElement(By.id("ct—something"));
driver.findElement(By.xpath("//a[@value=’BLR’])")).click();
Thread.sleep(2000); //as it will take few seconds to open the next list.
driver.findElement(By.xpath(["(//a[@value=’MAA’])[2]")).click();
```

if xpath is not correct then the Error will be “**xpath not found**” but in this case as the dynamic dropdown list has already disappeared. An Error will say “**element not visible**” which means element isn’t available for clicks.

Parent – Child relationship locator to Identify the unique objects :

NOTE: So in every organization there will be a concept called pull request. That means you write your code and submit it to the repository (whether git or svn) there after people will check your code and then only your code will be merged to the actual branch. you basically raise a request and after you raise a request then team will look into your code and approve.

Note: There are some coding standard in some places and people will generally not allow using index in your code. but there are some people who just don’t like to use indices. maybe they don’t like the hard coded stuff.

This method uses relative xpath

```
<div id = "random">
    <table id =  "this is it" ></table>
</div>

<div id = "random2">
    <table id =  "this is it" ></table>
</div>
```

now xpath for the first table can be written as **//*[@id=’random’]//@*[@id=’this is it’] or //*[@id=’random’]/table.**
This will be okay if table is static and we know what the tag is but if tag is not visible then we will go for id as ID’s are very stable (generally).

Hint: whenever you find multiple instances of same xpath just go for parent-child Relationship

Handling Auto – suggestive dropdown using selenium:

Selenium notes:

The dropdown which does not give any kind of predefined list rather just provides a list based upon what you type as suggestions then that are called auto suggestive dropdown.

for e.g. on flight booking websites we have to choose source and destination. If they are programmed as auto suggestive dropdown, then we will have to first click on the source list then we need to type first 3 letters for suggestions to begin (MUM for Mumbai. etc.) the we have to click on the appropriate suggestions.

Code:

```
import org.openqa.selnium.WebElement;

public class AutoSuggest {
    public static void main(String[] args) {
    ChromeDriver driver = new WebDriver();
    driver.get("http://makemytrip.com");     //url in the browser

    driver.findElement(By.id("hp-widget_sfrom")).clear();
    driver.findElement(By.id("hp-widget_sfrom")).sendKeys("MUM");
    Thread.sleep("2000L");
    driver.findElement(By.id("hp-widget_sfrom")).sendKeys(Keys.Enter);
    //selnium provides easy Key management

    WebElement source = driver.findElement(By.id("hp-widget_sTo"));
    source.click();
    source.sendKeys("DEL");
    Thread.sleep("2000L");
    source.sendKeys(Keys.ARROW_DOWN);
    //press down arrow to select 2nd option.
    source.sendKeys(Keys.Enter);
    }
}
```

destination should show a list in which Delhi is at the top and Delaine at 2nd position. If you did not select delaine then your script failed and shows the problem with the website because that's what you are testing.

Date: Saturday, June 20, 2020.

Handling Checkboxes:

```
public class Checkbox {
    public static void main(String[] args) {
    ChromeDriver driver = new WebDriver();
    driver.get("http://spicejet.com");  //url in the browser

    boolean statusSelected = false;
    System.out.println(statusSelected);

driver.findElement(By.cssSelector("input[id*='SeniorCitizenDiscount']")).click();
    statusSelected =
driver.findElement(By.cssSelector("input[id*='SeniorCitizenDiscount']")).isSelected
();
    System.out.println(statusSelected);

    int count = driver.findElements(By.cssSelector("input[type='checkbox']")).size;
    System.out.println("count of checkboxes " + count);
    }
}
```

Importance of assertion in web automation:

Note: TestNG is one of the testing framework. Junit is one of the other framework.

If the testing fails the assertion will tell whatever selenium is giving as a result is correct or not. These are used to put validation in our tests. Validation like the expected value is not returned then these engines returns false.

Testng**.**Assert classs contains methods like void ***assertTrue***(condition), ***assertFalse***(condition) and ***assertEquals***(actual, expected)

```
  boolean statusSelected = false;
    statusSelected = driver.findElement(By.
    cssSelector("input[id*='SeniorCitizenDiscount']")).isSelected();

    Assert.assertFalse(statusSelected);
    //assertFalse will return an error if inside cndn is ***true***

    driver.findElement(By.
    cssSelector("input[id*='SeniorCitizenDiscount']")).click();

    Assert.assertTrue(statusSelected);
    //assertFalse will return an error if inside cndn is ***false***

    int count = driver.findElements(By.
    cssSelector("input[type='checkbox']")).size;
    Assert.assertEquals(count, "6");
```

Error: java.lang.AssertionError: expected [true] but found [false] or in general expected [value] but found [value].

Handling Calendars in any websites:

These websites handle calendars by creating a table and then putting table row for each row and providing anchor in table data for each date as shown below.

now to get the current date using cssSelector ***"a.ui-state-default.item-ui-state-active.item-state-highlight"*** an example of calender date analogy used in websites.

```
<table>
<!---each table data defines ***each date*** -->
<tr>
<!---each ***past*** date is by default set as class inactive-->
<td class = "ui-state-disabled">
<a class = "ui-state-disabled item-ui-state-inactive"></a>
</td>
<td class = "ui-state-disabled">
<a  class = "ui-state-disbled item-ui-state-inactive">
</a></td>

<!---each ***current*** date is by default
 set as class as active and higlight-->
<td><a  class = "ui-state-default item-ui-state-active
 item-state-highlight"></a></td>
```

Selenium notes:

```
<!---each ***future*** date is by default set as class as only active-->
<td><a class = "ui-state-default item-ui-state-active">
</a></td>
<td><a class = "ui-state-default item-ui-state-active">
</a></td>
</tr>
</table>
```

Code:

```
driver.findElement(By.cssSelector("a.date-today.item-ui-select-active.item-select-highlight")).click();
    //selects the current date.
```

for current date it’s easy but when comes for future date we won’t get unique class.

Validating if UI elements are disabled:

Radio buttons are checkboxes but rather buttons are used to fill circles instead of checks in boxes. E.g.

```
<input checked=“checked” type = “radio”>
```

java code to select one-way button and check if return calendar element got disable.

```
driver.findElement(By.id("ct100-mainContent-rbtn-trip-1")).click();
Assert.assertTrue(driver.findElement(By.id(“Div1”)).isDisabled());
    //This should technically work but it won't.
```

Note: Selenium’s ***boolean isEnabled***() and ***boolean isDisabled***() are two methods that are not used. As they some are not compatible with the current Html and browsers out there. One of the reasons is they use UI elements clicks on knowing whether something is enabled or not and if its clickable then it should work as well. but that’s not the case with today’s websites; the element can still be clicked even after it disabled (not all but only some elements and some websites).

one of the method these guys use is to use style attribute to make it look disabled using css style.

```
<div style = "visibility: block; opacity: 1;"></div>
```

just by varying the opacity from 1 to .5 they make look like little faded out and disabled.

```
boolean isDisabled =
driver.findElement(By.id("Div1")).getAttributes("style").contains(".5");
Assert.assertTrue(isDisabled);
if(isDisabled) {
      System.out.println("Element is disabled");
    }
```

Handling everything at once (end to end automation):