**Selenium Interview Q&A**

**Q #1) Why should Selenium be selected as a test tool?**

Selenium: is a free and open source

have a large user base and helping communities

have cross Browser compatibility (Firefox, Chrome, Internet Explorer, Safari etc.)

have great platform compatibility (Windows, Mac OS, Linux etc.)

supports multiple programming languages (Java, C#, Ruby, Python, Pearl etc.)

has fresh and regular repository developments

supports distributed testing

**Q #2) What are the different types of locators in Selenium?**

The locator can be termed as an address that identifies a web element uniquely within the webpage.

**ID**: **import** org.openqa.selenium.WebDriver;

driver = webdriver.Chrome('./chromedriver')

driver.get("https://ssf.deva.ss.signanthealth.dev/SSP-150-DEVA/" )

Login ID = driver.findElement(By.id("id-search-field"));

**ClassName =** driver.findElemnt(By.className(“name of the class”));

**Name=** driver.findElemnt(By.Name(“name of the locator”));

**TagName**

**LinkText=** driver.findElemnt(By.linkText(“complete text of the locator”));

**PartialLinkText** = driver.findElemnt(By.partialLinkText(“complete text of the locator”));

**Xpath**

**CSS** **Selector**

**DOM**

**Q #3) What is the difference between assert and verify commands?**

**Assert (Hard Assert):** Assert command checks whether the given condition is true or false. Let’s say we assert whether the given element is present on the web page or not.

If the condition is true then the program control will execute the next test step but if the condition is false, the execution would stop, and no further test would be executed.

**Verify (Soft Assert):** Verify command also checks whether the given condition is true or false. Irrespective of the condition being true or false, the program execution doesn’t halt i.e. any failure during verification would not stop the execution and all the test steps would be executed.

**Q #4) What is the difference between “/” and “//” in Xpath?**

Single Slash “/” – Single slash is used to create Xpath with absolute path i.e. the xpath would be created to start selection from the document node/start node.

Double Slash “//” – Double slash is used to create Xpath with relative path i.e. the xpath would be created to start selection from anywhere within the document.

**Q #5) What is an XPath?**

XPath is used to locate a web element based on its XML path. XPath can be used to locate HTML elements. Enabling a user to find an element with the reference of another element.

**Q #6)** **When should I use Selenium Grid?**

Selenium Grid can be used to execute same or different test scripts on multiple platforms and browsers concurrently so as to achieve distributed test execution, testing under different environments and saving execution time.

**Q #7) How do I launch the browser using WebDriver?**

The following syntax can be used to launch Browser:

Webdriver driver;

System.setProperty(Webdriver.chrome.driver, "location of chrome driver from local system");

driver = new chromedriver ();

driver.mangae().window().maximize();

driver.get("url which you want to launch");  
*WebDriver driver = new FirefoxDriver();*

*WebDriver driver = new InternetExplorerDriver();*

**Q #8) What are the different types of waits available in WebDriver?**

There are two [types of waits available in WebDriver](https://www.softwaretestinghelp.com/selenium-webdriver-waits-selenium-tutorial-15/):

1. Implicit Wait
2. Explicit Wait

**Implicit Wait:**Implicit waits are used to provide a default waiting time (say 30 seconds) between each consecutive test step/command across the entire test script.

temp = driver.Manage().Timeouts().ImplicitWait(10, Timeunit.Seconds);

**Explicit Wait:** Explicit waits are used to halt the execution till the time a particular condition is met or the maximum time has elapsed

WebElement messageElement = wait.until( ExpectedConditions.presenceOfElementLocated(By.id("loginResponse")) );

* alertIsPresent()
* elementSelectionStateToBe()
* elementToBeClickable()
* elementToBeSelected()
* frameToBeAvaliableAndSwitchToIt()
* invisibilityOfTheElementLocated()
* invisibilityOfElementWithText()
* presenceOfAllElementsLocatedBy()
* presenceOfElementLocated()
* textToBePresentInElement()
* textToBePresentInElementLocated()
* textToBePresentInElementValue()
* titleIs()
* titleContains()
* visibilityOf()
* visibilityOfAllElements()
* visibilityOfAllElementsLocatedBy()
* visibilityOfElementLocated()

**Q #9) When do we use findElement() and findElements()?**

**findElement():**findElement() is used to find the first element in the current web page matching to the specified locator value.

**Syntax:**

*WebElement element = driver.findElements(By.xpath(“//div[@id=’example’]//ul//li”));*

**findElements():**findElements() is used to find all the elements in the current web page matching to the specified locator value. Return type is “List”

**Syntax:**  
*List <WebElement> elementList = driver.findElements(By.xpath(“//div[@id=’example’]//ul//li”));*

**Q #10) What is the difference between driver.close() and driver.quit command?**

**close()**: Closes the web browser window that the user is currently working on

**quit()**: Closes down all the windows that the program has opened.

**Q #11) How can we handle Alert pop-up?**

* void dismiss() – The dismiss() method clicks on the “Cancel” button as soon as the pop-up window appears.
* void accept() – The accept() method clicks on the “Ok” button as soon as the pop-up window appears.
* String getText() – The getText() method returns the text displayed on the alert box.
* void sendKeys(String stringToSend) – The sendKeys() method enters the specified string pattern into the alert box.

**Syntax:**  
*Alert abc = driver.switchTo().alert();*  
 *abc.accept();*

**Q #12) How to assert the title of the web page?**

*assertTrue(“The title of the window is incorrect.”,driver.getTitle().equals(“Title of the page”));*

**Q #13) How to mouse hover on a web element using WebDriver?**

we have used Action Interface to mouse hover on a drop down which then opens a list of options.

**Sample Code:**

|  |  |
| --- | --- |
| // Instantiating Action Interface  Actions actions=**new** Actions(driver);  // howering on the dropdown  actions.moveToElement(driver.findElement(By.id("id of the dropdown"))).perform();  // Clicking on one of the items in the list options  WebElement subLinkOption=driver.findElement(By.id("id of the sub link"));  subLinkOption.click();  **Q #14) How to capture screenshot in WebDriver?**   |  | | --- | | // Code to capture the screenshot  File scrFile = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);  // Code to copy the screenshot in the desired location  FileUtils.copyFile(scrFile, **new** File("C:\\CaptureScreenshot\\google.jpg")) | |

**Q #15)** **How to type in a textbox using Selenium?**

The user can use sendKeys(“String to be entered”) to enter the string in the textbox.

**Syntax:**  
*WebElement username = drv.findElement(By.id(“Email”));*  
*username.sendKeys(“sth”);*

**Q #16)** **How can you find if an element in displayed on the screen?**

To check the visibility of the web elements can be buttons, drop boxes, checkboxes, radio buttons, labels etc.

1. isDisplayed()
2. isSelected()
3. isEnabled()

**Syntax:**

**isDisplayed():**  
***boolean****buttonPresence = driver.findElement(By.id(“gbqfba”)).isDisplayed();*

**isSelected():**  
***boolean****buttonSelected = driver.findElement(By.id(“gbqfba”)).isSelected();*

**isEnabled():**  
***boolean****searchIconEnabled = driver.findElement(By.id(“gbqfb”)).isEnabled();*

**Q #17)** **How can we get a text of a web element?**

Get command is used to retrieve the inner text of the specified web element, it returns a string value.

**Syntax:**  
*String Text = driver.findElement(By.id(“Text”)).getText();*

**Q #18) How to select value in a dropdown?**

The value in the dropdown can be selected using WebDriver’s Select class.

**Syntax:**

**selectByValue:**  
*Select selectByValue =****new****Select(driver.findElement(By.id(“SelectID\_One”)));*  
*selectByValue.selectByValue(“greenvalue”);*

**selectByVisibleText:**  
*Select selectByVisibleText =****new****Select (driver.findElement(By.id(“SelectID\_Two”)));*  
*selectByVisibleText.selectByVisibleText(“Lime”);*

**selectByIndex:**  
*Select selectByIndex =****new****Select(driver.findElement(By.id(“SelectID\_Three”)));*  
*selectByIndex.selectByIndex(2);*

**Q #19) What are the different types of navigation commands?**

**navigate().back()** – The above command requires no parameters and takes back the user to the previous webpage in the web browser’s history.

**Sample code:**  
*driver.navigate().back();*

**navigate().forward()** – This command lets the user to navigate to the next web page with reference to the browser’s history.

**Sample code:**  
*driver.navigate().forward();*

**navigate().refresh()** – This command lets the user to refresh the current web page there by reloading all the web elements.

**Sample code:**  
*driver.navigate().refresh();*

**navigate().to()** – This command lets the user to launch a new web browser window and navigate to the specified URL.

**Sample code:**  
*driver.navigate().to(“https://google.com”);*

**Q #20) How to click on a hyper link using linkText?**

We have 2 methods to click on link text

1. linktext
2. partiallinktext

*driver.findElement(By.linkText(“Google”)).click();*

The command finds the element using link text and then click on that element and thus the user would be re-directed to the corresponding page.

*driver.findElement(By.partialLinkText(“Goo”)).click();*

The above command finds the element based on the substring of the link provided in the parenthesis and thus partialLinkText() finds the web element with the specified substring and then clicks on it.

**Q #21)** **How to**[**handle frame in WebDriver**](https://www.softwaretestinghelp.com/selenium-tutorial-18/)**?**

An inline frame acronym as iframe is used to insert another document within the current HTML document or simply a web page into a web page by enabling nesting.

**Select iframe by id**  
*driver.switchTo().frame(“ID of the frame“);*

**Locating iframe using tagName**  
*driver.switchTo().frame(driver.findElements(By.tagName(“iframe”).get(0));*

**Locating iframe using index**

**frame(index)**  
*driver.switchTo().frame(0);*

**frame(Name of Frame)**  
*driver.switchTo().frame(“name of the frame”);*

*Navigate back to the parent window or exit the frame*

**frame(WebElement element)**  
**Select Parent Window**  
*driver.switchTo().defaultContent();*

**Q #22)** **How to find more than one web element in the list?**

**Sample Code**

|  |
| --- |
| // Storing the list  List <WebElement> elementList = driver.findElements(By.xpath("//div[@id='example']//ul//li"));  // Fetching the size of the list  **int** listSize = elementList.size();  **for** (**int** i=0; i<listSize; i++)  {  // Clicking on each service provider link  serviceProviderLinks.get(i).click();  // Navigating back to the previous page that stores link to service providers  driver.navigate().back();  }  **Q #23) How to retrieve CSS properties of an element?**  The values of the css properties can be retrieved using a get() method:  **Syntax:** *driver.findElement(By.id(“id“)).getCssValue(“name of css attribute”);* *driver.findElement(By.id(“id“)).getCssValue(“font-size”);* |

**Q#24) How to do right click of an element in Selenium?**

By using Actions class, we can perform right click

actions action = newActions(driver);

WebElement element = driver.findElement(By.id("elementId"));

action.contextClick(element).perform();

**Q #25)How can you handle multiple windows in Selenium?**

By using **getwindowhandles()** method we can switch between different windows.

String handle= driver.getWindowHandle();

for (String handle : driver.getWindowHandles())

{

driver.switchTo().window(handle);

}

#### **26)  Mention 5 different exceptions you had in Selenium web driver?**

The 5 different exceptions you had in Selenium web drivers are

* WebDriverException
* NoAlertPresentException
* NoSuchWindowException
* NoSuchElementException
* TimeoutException

**27) What is a page object model in selenium?**

Page Object Model in Selenium – is an object design pattern in Selenium, where web pages are represented as classes, and the various elements on the page are defined as variables on the class. Page object model (POM) can be used in any kind of framework such as modular, data-driven, keyword driven, hybrid framework etc.

Advantages:

1. Page Object Design Pattern says operations and flows in the UI should be separated from verification. This concept makes our code cleaner and easy to understand.
2. The Second benefit is the object repository is independent of test cases, so we can use the same object repository for a different purpose with different tools. For example, we can integrate Page Object Model in Selenium with TestNG/JUnit for functional[Testing](https://www.guru99.com/software-testing.html)and at the same time with JBehave/Cucumber for acceptance testing.
3. Code becomes less and optimized because of the reusable page methods in the POM classes.

Sample Code:

public class Guru99Login {

WebDriver driver;

By user99GuruName = By.name("uid");

By password99Guru = By.name("password");

By titleText =By.className("barone");

By login = By.name("btnLogin");

public Guru99Login(WebDriver driver){

this.driver = driver;

}

//Set user name in textbox

public void setUserName(String strUserName){

driver.findElement(user99GuruName).sendKeys(strUserName);

}

**28) What is the Page Factory Class?**

The Page Factory Class is an extension to the Page Object design pattern. It is used to initialize the elements of the Page Object or instantiate the Page Objects itself. Annotations for elements can also be created.

**29) What is the difference between Page Object Model (POM) and Page Factory?**

A) Page Object is a class that represents a web page and hold the functionality and members.

Page Factory is a way to initialize the web elements you want to interact with within the page object when you create an instance of it.

Example to initiate the web element:

@FindBy(xpath=”.//\*[@id=’Email’]”)  
publicWebElementgmailUserIDWebEdit;

**30) What is the use of JavaScriptExecutor?**

It is an interface that offers this mechanism. It gives methods like “executescript” and “executeAsyncScript” to run JavaScript in the condition of the currently chosen frame or window. An example of that is:

JavascriptExecutor js = (JavascriptExecutor) driver;

js.executeScript(Script,Arguments);