Selenium WebDriver Interview Questions & Answers

Q 1: What is Selenium?

Ans: Selenium is a suite of software tools to automate web browsers across many platforms (Different Operation Systems like MS Windows, Linux Macintosh etc.). It was launched in 2004, and it is open source Test Tool suite.

Q 2: What is Selenium 3.0 ?

Ans: Web testing tools Selenium RC and WebDriver are consolidated in single tool in Selenium 3.0

Selenium 1.0 + WebDriver = Selenium 3.0

Q 3: What is Selenium WebDriver?

Ans: Selenium WebDriver is a tool for writing automated tests of websites. It is an API name and aims to mimic the behavior of a real user, and as such interacts with the HTML of the application. Selenium WebDriver is the successor of Selenium Remote Control which has been officially deprecated.

Q 4: What is cost of WebDriver, is this commercial or open source?

Ans: Selenium is an open source and free of cost.

Q 5: How you specify browser configurations with Selenium 3.0?

Ans: Following driver classes are used for browser configuration

AndroidDriver,

ChromeDriver,

EventFiringWebDriver,

FirefoxDriver,

HtmlUnitDriver,

InternetExplorerDriver,

IPhoneDriver,

IPhoneSimulatorDriver,

RemoteWebDriver

Q 6: How is Selenium 3.0 configuration different than Selenium 1.0?

Ans: In case of Selenium 1.0 you need Selenium jar file pertaining to one library for example in case of java you need java client driver and also Selenium server jar file. While with Selenium 3.0 you need language binding (i.e. java, C# etc) and Selenium server jar if you are using Remote Control or Remote WebDriver.

Q 7: Can you show me one code example of setting Selenium 3.0?

Ans: Below is example

```
WebDriver driver = new FirefoxDriver();

driver.get("https://www.google.co.in/");

driver.findElement(By.cssSelector("#gb_2 > span.gbts")).click();
```

Q 8: Which web driver implementation is fastest?

Ans: HTMLUnitDriver. Simple reason is HTMLUnitDriver does not execute tests on browser but plain http request – response which is far quick than launching a browser and executing tests. But then you may like to execute tests on a real browser than something running behind the scenes

Q 9: What all different element locators are available with Selenium 3.0?

Ans: Selenium 3.0 uses same set of locators which are used by Selenium 1.0 – id, name, css, XPath but how Selenium 3.0 accesses them is different. In case of Selenium 1.0 you don't have to specify a different method for each locator while in case of Selenium 3.0 there is a different method available to use a different element locator. Selenium 3.0 uses following method to access elements with id, name, css and XPath locator – driver.findElement(By.id("HTMLid"));

```
driver.findElement(By.name("HTMLname"));

driver.findElement(By.cssSelector("cssLocator"));

driver.findElement(By. className ("CalssName"));

driver.findElement(By. linkText ("LinkeText"));

driver.findElement(By. partialLinkText ("PartialLink"));
```

driver.findElement(By. tagName ("TanName”));

driver.findElement(By.xpath("XPathLocator));

Q 10: How do I submit a form using Selenium?

Ans: WebElement el = driver.findElement(By.id("ElementID"));

el.submit();

Q 11: How to capture screen shot in Webdriver?

Ans:

File file= ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);

FileUtils.copyFile(file, new File("c:\\name.png"));

Q 12: How do I clear content of a text box in Selenium 3.0?

Ans:

WebElement el = driver.findElement(By.id("ElementID"));

el.clear();

Q 13: How to execute java scripts function?

Ans:

JavascriptExecutor js = (JavascriptExecutor) driver;

String title = (String) js.executeScript("pass your java scripts");

Q 14: How to select a drop down value using Selenium3.0?

Ans: I am going to show you how to capture clip of page element using WebDriver.

Below I have written a “CaptureElementClip.java“java webdriver test script of a google application where I capture google menu clip and save into project.

package com.webdriver.test;

```
import java.awt.image.BufferedImage;
import java.io.File; import java.io.IOException;
import java.util.concurrent.TimeUnit; import
javax.imageio.ImageIO; import
org.apache.commons.io.FileUtils; import
org.openqa.selenium.By; import
org.openqa.selenium.OutputType; import
org.openqa.selenium.Point; import
org.openqa.selenium.TakesScreenshot; import
org.openqa.selenium.WebDriver; import
org.openqa.selenium.WebElement; import
org.openqa.selenium.firefox.FirefoxDriver; import
org.testng.annotations.AfterSuite; import
org.testng.annotations.BeforeSuite; import
org.testng.annotations.Test;

public class CaptureElementClip {

private WebDriver driver;
private String baseUrl;

@BeforeSuite
public void setUp() throws Exception {
 driver = new FirefoxDriver();
```

```
baseUrl = "http://google.com";
driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
        }

      @Test
public void testGoogle() throws IOException {
 //open application url
driver.get(baseUrl);
 //take screen shot
File screen = ((TakesScreenshot) driver).getScreenshotAs(OutputType.FILE);
//get webelement object of google menu locator
WebElement googleMenu = driver.findElement(By.id("gbz"));
Point point = googleMenu.getLocation();
//get element dimension
int width = googleMenu.getSize().getWidth();
int height = googleMenu.getSize().getHeight();
BufferedImage img = ImageIO.read(screen);
BufferedImage dest = img.getSubimage(point.getX(), point.getY(), width, height);
ImageIO.write(dest, "png", screen);
File file = new File("Menu.png");
FileUtils.copyFile(screen, file);
      }

@AfterSuite
```

```
public void tearDown() throws Exception {
                driver.quit();
 }
}
```

Q 15: How to automate radio button in Selenium 3.0?

Ans:

```
WebElement el = driver.findElement(By.id("Radio button id"));
 //to perform check operation
el.click() ;

//verfiy to radio button is check it return true if selected else false el.isSelected();
```

Q 16: How to count total number of rows of a table using Selenium 3.0?

Ans:

```
List {WebElement} rows = driver.findElements(By.className("//table[@id='tableID']/tr")); int
totalRow = rows.size();
```

Q 17: How to capture page title using Selenium 3.0?

Ans:

```
String title =  driver.getTitle();
```

Q 18: How to store page source using Selenium 3.0?

Ans:

```
String pagesource = driver.getPageSource();
```

Q 20: How to store current url using Selenium3.0?

Ans:

```
String currentURL  = driver.getCurrentUrl();
```

Q 21: How to assert text assert text of webpage using Selenium 3.0?

Ans:

```
WebElement el  =  driver.findElement(By.id("ElementID"));
//get test from element and stored in text variable
String text = el.getText();
 //assert text from expected
Assert.assertEquals("Element Text", text);
```

Q 22: How to get element attribute using Selenium 3.0?

Ans:

```
WebElement el  =  driver.findElement(By.id("ElementID"));
//get test from element and stored in text variable
String attributeValue = el. getAttribute("AttributeName") ; Q
```

23: How to double click on element using Selenium 3.0?

Ans:

```
WebElement el  =  driver.findElement(By.id("ElementID"));
Actions builder = new Actions(driver);
builder.doubleClick(el).build().perform();
```

Q 24: How to perform drag and drop in Selenium 3.0?

Ans:

```
WebElement source  =  driver.findElement(By.id("Source ElementID"));
WebElement destination  =  driver.findElement(By.id("Taget ElementID"));
Actions builder = new Actions(driver); builder.dragAndDrop(source,
destination ).perform();
```

Q 25: How to maximize window using Selenium 3.0?

Ans:

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

Q 26: How to verify PDF content using Selenium 3.0?

Ans: I will explain the procedure to verify PDF file content using java WebDriver. As some time we need to verify content of web application PDF file, opened in browser.

Use below code in your test scripts to get PDF file content.

```
//get current urlpdf file url

URL url = new URL(driver.getCurrentUrl());

//create buffer reader object

BufferedInputStream fileToParse = new BufferedInputStream(url.openStream());

PDFParserPDFParser = newPDFParser(fileToParse);

PDFParser.parse();

//savePDF text into strong variable

String pdftxt = newPDFTextStripper().getText(pdfParser.getPDDocument());

//closePDFParser object

PDFParser.getPDDocument().close();
```

After applying above code, you can store all PDF file content into “pdftxt” string variable. Now you can verify string by giving input. As if you want to verify “Selenium or WebDiver” text. Use below code.

```
Assert.assertTrue(pdftxt.contains(“Selenium or WebDiver”))
```

Q 27: How to verify response 200 code using Selenium 3.0?

Ans: I will explain you to verify HTTP response code 200 of web application using java webdriver. As webdriver does not support direct any function to verify page response code. But using "WebClient" of HtmlUnit API we can achieve this.

Html unit API is GUI less browser for java developer, using WebClent class we can send request to application server and verify response header status.

Below code, I used in my webdriver script to verify response 200 of web application

```
String url = "http://www.google.com/";

WebClient webClient = new WebClient();

HtmlPage htmlPage = webClient.getPage(url);

 //verify response

Assert.assertEquals(200,htmlPage.getWebResponse().getStatusCode());

Assert.assertEquals("OK",htmlPage.getWebResponse().getStatusMessage());
```

If HTTP authentication is required in web application use below code.

```
String url = "Application Url";

WebClient webClient = new WebClient();

DefaultCredentialsProvider credential = new DefaultCredentialsProvider();

//Set some example credentials

credential.addCredentials("UserName",  "Passeord");

webClient.setCredentialsProvider(credential);

HtmlPage htmlPage = webClient.getPage(url);

//verify response

Assert.assertEquals(200,htmlPage.getWebResponse().getStatusCode());

Assert.assertEquals("OK",htmlPage.getWebResponse().getStatusMessage());
```

Q 28: How to verify image using Selenium 3.0?

Ans: I have explained that how to verify images in webdriver using java. As webdriver does not provide direct any function to image verification, but we can verify images by taking two screen

shots of whole web page using “TakesScreenshot” webdriver function, one at script creation time and another at execution time,

In below example I have created a sample script in which first I captured a Google home page screen shot and saved (GoogleInput.jpg) into my project, Another screen shot “GoogleOutput.jpg” captured of same page at test executing time and saved into project. I compared both images if they are not same then test will script fail. Here is sample code for same

```
package com.test;

import java.awt.image.BufferedImage;

import java.awt.image.DataBuffer;

import java.io.File;

import java.io.IOException;

import java.util.concurrent.TimeUnit;

import javax.imageio.ImageIO;

import org.apache.commons.io.FileUtils;

import org.openqa.selenium.OutputType;

 import org.openqa.selenium.TakesScreenshot;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.testng.Assert;
import org.testng.annotations.AfterSuite;
import org.testng.annotations.BeforeSuite;
import org.testng.annotations.Test;
```

```
public class ImageComparison {

public WebDriver driver;       private
String baseUrl;

@BeforeSuite
public void setUp() throws Exception
{
            driver = new FirefoxDriver();
             baseUrl = "https://www.google.co.in/";
            driver.manage().timeouts().implicitlyWait(30,
            TimeUnit.SECONDS);
      }

@AfterSuite
public void tearDown() throws Exception {
          driver.quit();
      }

@Test
public void testImageComparison()
throws IOException, InterruptedException
      {
```

```
        driver.navigate().to(baseUrl);
        File screenshot = ((TakesScreenshot)driver).
        getScreenshotAs(OutputType.FILE);
        Thread.sleep(3000);
        FileUtils.copyFile(screenshot, new File("GoogleOutput.jpg"));
        File fileInput = new File("GoogleInput.jpg");
        File fileOutPut = new File("GoogleOutput.jpg");
        BufferedImage bufileInput = ImageIO.read(fileInput);
        DataBuffer dafileInput = bufileInput.getData().getDataBuffer();
        int sizefileInput = dafileInput.getSize();
        BufferedImage bufileOutPut = ImageIO.read(fileOutPut);
        DataBuffer dafileOutPut =bufileOutPut.getData().getDataBuffer();
        int sizefileOutPut = dafileOutPut.getSize();
        Boolean matchFlag = true;
          if(sizefileInput == sizefileOutPut) {
          for(int j=0; j<sizefileInput; j++) {
          if(dafileInput.getElem(j) != dafileOutPut.getElem(j)) {
          matchFlag = false;

          break;
         }
      }
}
else
          matchFlag = false;
```

```
        Assert.assertTrue(matchFlag, "Images are not same");
    }
}
```