Latest Selenium Interview Questions:

1. Difference between get and navigate method in Selenium?
2. Difference between quit and close methods in Webdiver?
3. What is implicit wait?
4. Difference between implicit and explicit wait?
5. In how many ways we can handle frames in the application using webdriver methods?
6. Code to handle 3rd child window?
7. How to handle https certifications?
8. Different type of locators present in webdriver?
9. Write Syntax for xpath and css if id and tag  are given
10. How to use Contains regular expression to xpath?
11. How to use regular expression to CSS?
12. What is the class available in selenium to handle drop downs?
13. What is the method to check if checkbox is selected ?
14. How to validate if element is visible or hidden in webpages?
15. How to get the count of similar objects list in the web page?
16. Importance of desired capabilites Mechanism
17. How to enter the text in capslock?
18. How to mouse over on the web element on page?
19. Methods to handle Java Alert?
20. How to get links count in the page?
21. How to validate if we are navigated to child window successfully?
22. Difference between relative and absolute xpath ?
23. Write down the sample xpath syntax to handle parent from child object?
24. What driver is must to run tests in firefox driver?
25. What driver is must to run tests in Chrome driver?
26. How do you set driver in firefox and chrome drivers through script?
27. Difference between findElement and findELements
28. List out any 2 methods available in explicit wait
29. How to take screenshots with selenium webdriver
30. How to hit enter from webdriver commands

1. GET method will get a page to load or get page source or get text that’s all. GET will wait till the whole page gets loaded i.e. the onload event has fired before returning control to our test or script. If our pages uses lot of AJAX then we can’t know that when our pages has completely loaded. To overcome this we can use WAIT.

NAVIGATE will just redirect to our required page and will not wait. It will guide us through the history like refresh, back, forward. For example if we want to move forward and do some functionality and back to the home page then this can be achieved through navigate() only.

GET and NAVIGATE do exactly the same thing, the only difference between them is that one is easier to type than other.

2.close( ) WebDriver command closes the Browser window which is in focus.

If there are more than one Browser window opened by the Selenium Automation, then the close( ) command will only close the Browser window which is having focus at that time. It wont close the remaining Browser windows.

Where as quit( ) WebDriver command is generally used to shut down the WebDrivers instance. Hence it closes all the Browser windows that are opened by the Selenium Automation.

close( ) and quit( ) work in the similar way when Selenium Automation opens only single Browser window. They differ in their functionality when there are more than one Browser windows opened by the Selenium Automation.

3, 4

Implicit Wait: During Implicit wait if the Web Driver cannot find it immediately because of its availability, it will keep polling (around 250 milli seconds) the DOM to get the element. If the element is not available within the specified Time an NoSuchElementException will be raised. The default setting is zero. Once we set a time, the Web Driver waits for the period of the WebDriver object instance.

Explicit Wait: There can be instance when a particular element takes more than a minute to load. In that case you definitely not like to set a huge time to Implicit wait, as if you do this your browser will going to wait for the same time for every element.

To avoid that situation you can simply put a separate time on the required element only. By following this your browser implicit wait time would be short for every element and it would be large for specific element.

5. With FrameID, Frame name, Frame webElement

6.

tring parent=driver.getWindowHandle();

// This will return the number of windows opened by Webdriver and will return Set of St//rings

Set<String>s1=driver.getWindowHandles();

// Now we will iterate using Iterator

Iterator<String> I1= s1.iterator();

I1.next()// Gets child window id

I1.next() // Get 2nd child window

I1.next() // Gets 3rd child window

7.DesiredCapabilities cap=DesiredCapabilities.chrome();

// Set ACCEPT\_SSL\_CERTS variable to true

cap.setCapability(CapabilityType.ACCEPT\_SSL\_CERTS, true);

// Set the driver path

System.setProperty("webdriver.chrome.driver","Chrome driver path");

// Open browser with capability

WebDriver driver=new ChromeDriver(cap);

8.ID, Xpath, CSS, ClassName, Name, Linktext, tagName

9.Xpath = //tagName[@attribute=‘value’]

CSS = tagName[attribute=‘value’]

10.//tagname[contains(@attribute,’value’)]

11. tagName[attribute\*=‘value’]

12. Select class

13.isSelected()

14. isDisplayed()

15.driver.findElements(By.xx(“”)).size();

16.The desired capability is a series of key/value pairs that stores the browser properties like browsername, browser version, the path of the browser driver in the system, etc. to determine the behaviour of the browser at run time

17. driver.findelement("YOURELEMENTLOCATOR").sendKeys(Keys.SHIFT,"yourtexttobetyped")

18.Actions a =new Actions(driver);

a.moveToElement().build().perform()

19.driver.switchTo.Alert();

20.driver.findElements(By.tagName(“a”).size();

21. By checking the title of child window

22. Absolute Xpath: It uses Complete path from the Root Element to the desire element.

Relative Xpath: You can simply start by referencing the element you want and go from there.

Always Relative Xpaths are preferred as they are not the complete paths from the Root element. (//html//body) ..Beacuse in future any of the webelement when added/Removed then Absolute Xpath changes. So Always use Relative Xpaths in your Automation.

23.//tagName[@attribute=value]/parent::tag name

24.Geckodriver

25. ChromeDriver

26. Using System.setProperty(web driver.chrome.driver ‘path to .exe’)

27.FindElement identifies first object which matches with provided locator on the screen with top left scanning

FindElements gets all the objects which matched and takes into List

28. visibilityOfElementLocated, PresenceOfElementLocated

29.driver.get("http://www.google.com");

// Take screenshot and store as a file format

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

try {

// now copy the screenshot to desired location using copyFile //method

FileUtils.copyFile(src, new File("C:/selenium/error.png"));

}

catch (IOException e)

{

System.out.println(e.getMessage());

}

}

30. sendKeys(“Keys.Enter”)