**1.What are the limitations of selenium web driver?**

* It does not support and non web-based applications, it only supports web based applications.
* Its an open source tool so in case of any technical issues you need to rely on the selenium community forums to get your issue resolved.
* You need to know at least one of the supported language very well in order to automate your application successfully.
* No inbuilt reporting capability so you need plugins like JUnit and TestNG for test reports.
* Lot of challenges with IE browser.

**2.installing/configure selenium?**

* step1:You can download the Selenium Java Client Driver from this location:<http://docs.seleniumhq.org/download/>
* step2: Click on Download link which is associated with java release. Download the jar files. You will find client drivers for other languages there, but only choose the one for Java as shown above in the screenshot.  
  After downloading you will get a folder which should contain two jar files related to selenium, a lib folder which also contains jar files inside it. Beside this, there is a change log file as well. Refer to the screenshot below.
* step3:Navigate to C drive and open eclipse folder where we have extacted all the application files related to eclipse. In Eclipse folder you would find a 'eclipse.exe' file. Now click on the .exe file This would ask you to select a workspace, you may accept the default location or simply create a new workspace inside C drive and confirm OK.

- Now in Eclipse create a new project and for creating click on File menu –> New –> Java Project -> Name the project as “testproject” and click finish.

- Create a new Java class, for doing this again click on File menu –> New -> Class then name it as “testclass” and select the checkbox for 'public static void main (String[] args) and click finish.

On clicking 'Finish' in 'Java Class' wizard, your Eclipse IDE should look like the image below.

- Now Right-click on testproject and select Properties.  
 - On the Properties window, click on “Java Build Path”.

- Click on the Libraries tab, and then click “Add External JARs..”

- Navigate to C:\selenium-2.37.0\ (or any other location where you saved the extracted contents of “selenium-2.37.0.zip” in step 3).  
 - Add all the JAR files inside and outside the “libs” folder. Your Properties window after adding all the jar file should now look similar to the image below.

- Finally, click OK and we have completed importing Selenium libraries into our project.

step4:write code

**3.what are different ways of locating elements in selenium?**

There are some browser tools that you can use in order to identify web elements in the DOM easier. These are:

* Firebug for Firefox
* Google Developer Tools for Chrome
* Web Inspector for Safari

There are various strategies to locate elements in a page. You can use the most appropriate one for your case. Selenium provides the following methods to locate elements in a page:

* find\_element\_by\_id
* find\_element\_by\_name
* find\_element\_by\_xpath
* find\_element\_by\_link\_text
* find\_element\_by\_partial\_link\_text
* find\_element\_by\_tag\_name
* find\_element\_by\_class\_name
* find\_element\_by\_css\_selector

[Finding web elements with Selenium WebDriver](https://loadfocus.com/blog/2013/09/05/how-to-locate-web-elements-with-selenium-webdriver/) by ID is usually the fastest option, but here is the list of the best and fastest selectors Selenium WebDriver Tests run faster:

**4. which is fastest way to identify elements in web page?**

1. ID selectors (By.ID – Matches by @id attribute)
   * IDs are the safest, fastest locator option and should always be your first choice
   * IDs should be unique in every page according to W3C website
   * even if the DOM changes, if the ID is still there, then WebDriver can still locate it
   * always try and get extra IDs added into the code, this makes testers life easier
   * fastest locator as it uses the document.getElementById() javascript command which is optimised by many browsers
2. CSS and Name selectors (Matches by CSS selector or @name attribute)
   * faster than XPath
   * whenever IDs are not available/ usable, usually @name can be used to identify elements
   * CSS and XPath locators are very similar
   * not very flexible in identifying elements comparing to XPath
3. XPath locators (Matches with arbitrary XPath expression)
   * most flexible in order to build reliable web element locators
   * very slow locator (particularly in IE) since in order to locate the element it needs to traverse the whole DOM of the page which is a time consuming operation
   * check here [how to identify and validate XPath locators inside Chrome Developer Tool](http://loadfocus.com/blog/tech/2015/07/how-to-find-the-xpath-of-your-selenium-webdriver-test-using-chrome-browser/), also you can install a Firebug extension to be able to identify and validate XPaths easily with FirePath from FireBug (adds a development tool to edit, inspect and generate XPath 1.0 expressions, CSS 3 selectors and JQuery selectors).

**5.what is absolute path and relative path in xpath?**

In this example, we've created a sample XML document students.xml and its stylesheet document students.xsl which uses the XPath expressions.

Following is the sample XML used.

students.xml

<?xml version = "1.0"?>

<?xml-stylesheet type = "text/xsl" href = "students.xsl"?>

<class>

<student rollno = "393">

<firstname>Dinkar</firstname>

<lastname>Kad</lastname>

<nickname>Dinkar</nickname>

<marks>85</marks>

</student>

<student rollno = "493">

<firstname>Vaneet</firstname>

<lastname>Gupta</lastname>

<nickname>Vinni</nickname>

<marks>95</marks>

</student>

<student rollno = "593">

<firstname>Jasvir</firstname>

<lastname>Singh</lastname>

<nickname>Jazz</nickname>

<marks>90</marks>

</student>

</class>

Stylesheet:

<?xml version = "1.0" encoding = "UTF-8"?>

<xsl:stylesheet version = "1.0"

xmlns:xsl = "http://www.w3.org/1999/XSL/Transform">

<xsl:template match = "/" >

<html>

<body>

<h3>Details of each Students. </h3>

<table border = "1">

<tr bgcolor = "#9acd32">

<th>Roll No</th>

<th>First Name</th>

<th>Last Name</th>

<th>Nick Name</th>

<th>Marks</th>

</tr>

<tr>

<td><xsl:value-of select = "/class/student[1]/@rollno"/></td>

<td><xsl:value-of select = "/class/student[1]/firstname"/></td>

<td><xsl:value-of select = "/class/student[1]/lastname"/></td>

<td><xsl:value-of select = "/class/student[1]/nickname"/></td>

<td><xsl:value-of select = "/class/student[1]/marks"/></td>

</tr>

<tr>

<td>

<xsl:value-of select = "/class/student/@rollno"/>

</td>

<td><xsl:value-of select = "/class/student[2]/firstname"/></td>

<td><xsl:value-of select = "/class/student[2]/lastname"/></td>

<td><xsl:value-of select = "/class/student[2]/nickname"/></td>

<td><xsl:value-of select = "/class/student[2]/marks"/></td>

</tr>

<tr>

<td>

<xsl:value-of select = "/class/student[3]/@rollno"/>

</td>

<td><xsl:value-of select = "/class/student[3]/firstname"/></td>

<td><xsl:value-of select = "/class/student[3]/lastname"/></td>

<td><xsl:value-of select = "/class/student[3]/nickname"/></td>

<td><xsl:value-of select = "/class/student[3]/marks"/></td>

</tr>

</table>

</body>

</html>

</xsl:template>

</xsl:stylesheet>

xpath example:

<?xml version = "1.0" encoding = "UTF-8"?>

<xsl:stylesheet version = "1.0"

xmlns:xsl = "http://www.w3.org/1999/XSL/Transform">

<xsl:template match = "/" >

<html>

<body>

<h3>Details of each Students. </h3>

<table border = "1">

<tr bgcolor = "#9acd32">

<th>Roll No</th>

<th>First Name</th>

<th>Last Name</th>

<th>Nick Name</th>

<th>Marks</th>

</tr>

<xsl:for-each select = "/class/student">

<tr>

<td><xsl:value-of select = "@rollno"/></td>

<td><xsl:value-of select = "firstname"/></td>

<td><xsl:value-of select = "lastname"/></td>

<td><xsl:value-of select = "nickname"/></td>

<td><xsl:value-of select = "marks"/></td>

</tr>

</xsl:for-each>

</table>

</body>

</html>

</xsl:template>

</xsl:stylesheet>

**6.different types of waits or synchronization in selenium webdriver?**

Synchronization can be classified into two categories:

**1. Unconditional**  
**2. Conditional Synchronization**

**Unconditional :**  
In this we just specify timeout value only. We will make the tool to wait until certain amount of time and then proceed further.

Examples: Wait() and [Thread.Sleep();](https://docs.oracle.com/javase/tutorial/essential/concurrency/sleep.html" \t "_blank)

The main disadvantage for the above statements are, there is a chance of unnecessary waiting time even though the application is ready.

The advantages are like in a situation where we interact for third party systems like interfaces, it is not possible to write a condition or check for a condition. Here in this situations, we have to make the application to wait for certain amount of time by specifying the timeout value.

**Conditional Synchronization:**

We specify a condition along with timeout value, so that tool waits to check for the condition and then come out if nothing happens.

It is very important to set the timeout value in conditional synchronization, because the tool should proceed further instead of making the tool to wait for a particular condition to satisfy.

In Selenium we have implicit Wait and Explicit Wait conditional statements. Check here for [Examples on how to use Webdriver Waits](http://seleniumeasy.com/selenium-tutorials/webdriver-wait-examples)

**1. Implicit Wait.**

An implicit wait is to tell WebDriver to poll the DOM for a certain amount of time when trying to find an element or elements if they are not immediately available.

The default setting is 0. Once when we define the implicit wait, it will set for the life of the WebDriver object instance.

It is a mechanism which will be written once and applied for entire session automatically. It should be applied immediately once we initiate the Webdriver.

Implicit wait will not work all the commands/statements in the application. It will work only for "FindElement" and "FindElements" statements.

If we set implicit wait, find element will not throw an exception if the element is not found in first instance, instead it will poll for the element until the timeout and then proceeds further. We should always remember to add the below syntax immediately below the Webdriver statement.

**Example using implicit timeout**

WebDriver driver = **new** FirefoxDriver();

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

driver.**get**("[http://www.google.com"](http://www.google.com/));

**Explicit Wait:**

We need to define a wait statement for certain condition to be satisfied until the specified timeout period. If the Webdriver finds the element within the timeout period the code will get executed.

Explicit wait is mostly used when we need to Wait for a specific content/attribute change after performing any action, like when application gives AJAX call to system and get dynamic data and render on UI.

Example: Like there are drop-downs Country and State, based on the country value selected, the values in the state drop-down will change, which will take few seconds of time to get the data based on user selection.

Example:

/\*Explicit **wait** **for** **state** dropdown field\*/

    WebDriverWait **wait** = new WebDriverWait(driver, 10);

**wait**.**until**(ExpectedConditions.visibilityOfElementLocated(By.id("statedropdown")));

The above statement waits up to 10 seconds before throwing Exception (TimeoutException - Timed out after 10 seconds waiting for visibility of element) or if it finds the element, it will return in 0 - 10 seconds.

There are different waits that can be used based on the needs which we frequently come across when automating web applications. Check here for [WebDriver Waits Example](http://seleniumeasy.com/selenium-tutorials/webdriver-wait-examples).

**Fluent Wait:**

Using FluentWait we can define the maximum amount of time to wait for a condition, as well as the frequency with which to check for the condition.

And also the user can configure to ignore specific types of exceptions such as ["NoSuchElementExceptions"](http://selenium.googlecode.com/git/docs/api/java/org/openqa/selenium/NoSuchElementException.html) when searching for an element. NoSuchElement exception is thrown by findElement(By) and findElements(By). When ever it try to find any element it returns the first matching element on the current page else it throws NoSuchElementException - when no matching elements are found.

Syntax:

Wait<WebDriver> wait = new FluentWait<WebDriver>(driver)

//Wait for the condition

.withTimeout(30, TimeUnit.SECONDS)

// which to check for the condition with interval of 5 seconds.

.pollingEvery(5, TimeUnit.SECONDS)

//Which will ignore the NoSuchElementException

.ignoring(NoSuchElementException.class);

**7.how to save screen shots using selenium webdriver?**

**import** java.io.File; **import** org.apache.commons.io.FileUtils; **import** org.openqa.selenium.By; **import** org.openqa.selenium.OutputType; **import** org.openqa.selenium.TakesScreenshot; **import** org.openqa.selenium.WebDriver; **import** org.openqa.selenium.firefox.FirefoxDriver; **import** org.testng.annotations.Test;          **public** **class** **takeScreenShotExample**{         **public** WebDriver driver;                @Test       **public** **void** **openBrowser**() **throws** Exception {       driver = **new** FirefoxDriver();       driver.manage().window().maximize();       driver.get("[http://www.google.com"](http://www.google.com/));       **try**{                 //the below statement will throw an exception as the element is not found, Catch block will get executed and takes the screenshot.       driver.findElement(By.id("testing")).sendKeys("test");                                     //if we remove the below comment, it will not return exception and screen shot method will not get executed.       //driver.findElement(By.id("gbqfq")).sendKeys("test");       }       **catch** (Exception e){       System.out.println("I'm in exception"); //calls the method to take the screenshot.       getscreenshot();        }       }             **public** **void** **getscreenshot**() **throws** Exception       {               File scrFile = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);            //The below method will save the screen shot in d drive with name "screenshot.png"               FileUtils.copyFile(scrFile, **new** File("D:\\screenshot.png"));       } }

@Test

public void test () throws InterruptedException

{

driver.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);

driver.findElement(By.xpath("//b[contains(.,'Open New Page')]")).click();

// Get and store both window handles in array

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

String window1 = (String) AllWindowHandles.toArray()[0];

System.out.print("window1 handle code = "+AllWindowHandles.toArray()[0]);

String window2 = (String) AllWindowHandles.toArray()[1];

System.out.print("\nwindow2 handle code = "+AllWindowHandles.toArray()[1]);

Switch to window2(child window) and performing actions on it.

driver.switchTo().window(window2);

driver.findElement(By.xpath("//input[@name='fname']")).sendKeys("My Name");

driver.findElement(By.xpath("//input[@value='Bike']")).click();

driver.findElement(By.xpath("//input[@value='Car']")).click();

driver.findElement(By.xpath("//input[@value='Boat']")).click();

driver.findElement(By.xpath("//input[@value='male']")).click();

Thread.sleep(5000);

//Switch to window1(parent window) and performing actions on it.

driver.switchTo().window(window1);

driver.findElement(By.xpath("//option[@id='country6']")).click();

driver.findElement(By.xpath("//input[@value='female']")).click();

driver.findElement(By.xpath("//input[@value='Show Me Alert']")).click();

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

Thread.sleep(5000);

//Once Again switch to window2(child window) and performing actions on it.

driver.switchTo().window(window2);

driver.findElement(By.xpath("//input[@name='fname']")).clear();

driver.findElement(By.xpath("//input[@name='fname']")).sendKeys("Name Changed");

Thread.sleep(5000);

driver.close();

//Once Again switch to window1(parent window) and performing actions on it.

driver.switchTo().window(window1);

driver.findElement(By.xpath("//input[@value='male']")).click();

Thread.sleep(5000);}