# Automation Testing

## Web Automation

1. **What are the different types of locators in Selenium?**

In Selenium, locators are used to identify and interact with elements on a web page. There are several types of locators available in Selenium:

1. **ID Locator:**
   * Syntax: **By.id("idValue")**
   * Example: **By.id("username")**
   * Usage: Locates an element by its unique HTML **id** attribute.
2. **Name Locator:**
   * Syntax: **By.name("nameValue")**
   * Example: **By.name("password")**
   * Usage: Locates elements by the **name** attribute.
3. **ClassName Locator:**
   * Syntax: **By.className("classValue")**
   * Example: **By.className("login-btn")**
   * Usage: Locates elements by the value of their **class** attribute.
4. **TagName Locator:**
   * Syntax: **By.tagName("tagname")**
   * Example: **By.tagName("a")**
   * Usage: Locates elements by HTML tag name.
5. **XPath Locator:**
   * Syntax: **By.xpath("xpathExpression")**
   * Example: **By.xpath("//input[@id='username']")**
   * Usage: Locates elements using XPath expressions. Relative XPath provides a powerful way to locate an element while an Absolute XPath is not recommended to use.
6. **CSS Selector Locator:**
   * Syntax: **By.cssSelector("cssSelector")**
   * Example: **By.cssSelector("input#username")**
   * Usage: Locates elements using CSS selectors. CSS selectors are a concise way to identify elements based on class, ID, attributes, etc.
7. **LinkText Locator:**
   * Syntax: **By.linkText("linkText")**
   * Example: **By.linkText("Login")**
   * Usage: Locates hyperlinks by the exact text of the link.
8. **PartialLinkText Locator:**
   * Syntax: **By.partialLinkText("partialLinkText")**
   * Example: **By.partialLinkText("Forgot")**
   * Usage: Locates hyperlinks by partial text matching.
9. **What are the different types of Drivers available in WebDriver?**
10. **FirefoxDriver:**
    * Used for interacting with the Mozilla Firefox browser.
    * Example:

WebDriver driver = new FirefoxDriver();

1. **ChromeDriver:**
   * Used for interacting with the Google Chrome browser.
   * Example:

WebDriver driver = new ChromeDriver();

1. **InternetExplorerDriver:**
   * Used for interacting with the Internet Explorer browser.
   * Example:

WebDriver driver = new InternetExplorerDriver();

1. **EdgeDriver:**
   * Used for interacting with the Microsoft Edge browser.
   * Example:

WebDriver driver = new EdgeDriver();

1. **SafariDriver:**
   * Used for interacting with the Safari browser on macOS.
   * Example:

WebDriver driver = new SafariDriver();

1. **OperaDriver:**
   * Used for interacting with the Opera browser.
   * Example:

WebDriver driver = new OperaDriver();

1. **RemoteWebDriver:**
   * Allows you to run tests on a remote machine (on a different node of the Selenium Grid).
   * Example:

WebDriver driver = new RemoteWebDriver(new URL("http://gridhuburl:4444/wd/hub"), capabilities);

1. **What are the different types of waits available in WebDriver?**
2. **Implicit Wait:**

* Implicit waits are used to set a default waiting time (in seconds) for the entire script. The WebDriver will wait for the specified time before throwing an exception if an element is not immediately present.
* Example:

WebDriver driver = new ChromeDriver();

driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(2));

1. **Explicit Wait:**

* Explicit waits allow you to wait for a specific condition to occur before proceeding with the execution. You can specify the maximum amount of time to wait for a certain condition and the frequency with which to check the condition.
* Example using **ExpectedConditions**:

WebDriver driver = new ChromeDriver();

WebDriverWait wait = new WebDriverWait(driver, 10);

WebElement element= wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("elementId")));

1. **Fluent Wait:**

* Fluent waits are a more flexible way of setting up waits in Selenium. With fluent waits, you can specify the maximum amount of time to wait, the frequency of checking the condition, and also ignore specific exceptions during the wait.
* Example:

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

.withTimeout(Duration.ofSeconds(30))

.pollingEvery(Duration.ofSeconds(5))

.ignoring(NoSuchElementException.class);

WebElement element = wait.until(driver -> driver.findElement(By.id("elementId")));

1. **PageLoadTimeout:**

* PageLoadTimeout sets the amount of time to wait for a page to load completely. If the page doesn't load within the specified time, a TimeoutException is thrown.
* Example:

WebDriver driver = new ChromeDriver();

driver.manage().timeouts().pageLoadTimeout(30, TimeUnit.SECONDS);

1. **Static Wait:**

* Type of wait not recommended to use because it is a static wait, the WebDriver will wait for a specific amount of time even it locates the elements or not.
* Example: Thread.sleep(1000);

1. **What is the difference between driver.quite() and driver.close()?**

* Use **driver.quit()** when you want to close all browser windows and terminate the WebDriver session.
* Use **driver.close()** when you want to close only the currently active browser window or tab, leaving the WebDriver session open for further interactions with other windows or tabs.

It's a good practice to call driver.quit() at the end of your test script to ensure that all browser instances are closed, releasing system resources.