Selenium WebDriver Wait Commands

Listing out the different WebDriver ***Wait*** statements that can be useful for an effective scripting and can avoid using the ***Thread.sleep()*** commands.

To learn advance waits you can go to ***Advance Selenium Wait*** and ***Handle Ajax Waits in Selenium.***

ImplicitlyWait Command

Purpose: Selenium WebDriver has borrowed the idea of ***implicit waits*** from ***Watir.*** This means that we can tell Selenium that we would like it to wait for a certain amount of time before throwing an ***exception*** that it cannot find the element on the page. We should note that implicit waits will be in place for the entire time the browser is open. This means that any search for elements on the page could take the time the implicit wait is set for.

```
WebDriver driver => new FirefoxDriver();
 driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
 driver.get("https://url_that_delays_loading");
 WebElement myDynamicElement = driver.findElement(By.id("myDynamicElement"));
```

FluentWait Command

Purpose: Each ***FluentWait*** instance defines the maximum amount of time to wait for a condition, as well as the frequency with which to check the condition. Furthermore, the user may configure the wait to ignore specific types of exceptions whilst waiting, such as ***NoSuchElementExceptions*** when searching for an element on the page.

```
// Waiting 30 seconds for an element to be present on the page, checking
  // for its presence once every 5 seconds.
  Wait wait = new FluentWait(driver)
```

```
    .withTimeout(30, SECONDS)
    .pollingEvery(5, SECONDS)
    .ignoring(NoSuchElementException.class);
  WebElement foo = wait.until(new Function() {
    public WebElement apply(WebDriver driver) {
    return driver.findElement(By.id("foo"));
    }
   });
```

ExpectedConditions Command

Purpose: Models a condition that might reasonably be expected to eventually evaluate to something that is neither null nor false.

```
WebDriverWait wait = new WebDriverWait(driver, 10);
 WebElement element = wait.until(ExpectedConditions.elementToBeClickable(By.id(>someid>))
```

PageLoadTimeout Command

Purpose: Sets the amount of time to wait for a page-load to complete before throwing an error. If the timeout is negative, page loads can be indefinite.

```
driver.manage().timeouts().pageLoadTimeout(100, SECONDS);
```

SetScriptTimeout Command

Purpose: Sets the amount of time to wait for an asynchronous script to finish execution before throwing an error. If the timeout is negative, then the script will be allowed to run indefinitely.

```
driver.manage().timeouts().setScriptTimeout(100,SECONDS);
```

Sleep Command

Purpose: This is rarely used, as it always force the browser to wait for a specific time. ***Thread. Sleep*** is never a good idea and that's why Selenium provides wait for primitives. If you use them you can specify much higher timeout value which makes tests more reliable without slowing them down as the condition can be evaluated as often as it's required.

```
thread.sleep(1000);
```

Practice Exercise

. Launch new Browser

. Open URL "***https://toolsqa.com/automation-practice-switch-windows/***"

. Click on the Button "***Timing Alert***"

. Accept the Alert (*Alert will take 3 seconds to get displayed, Use WebDriverWait to wait for it*)

Solution

```
package practiceTestCases;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.Alert;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;
```

```
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
public class PracticeWaitCommands {
        public static WebDriver driver;
        public static void main(String[] args) {
                // Create a new instance of the Firefox driver
        driver = new FirefoxDriver();
        // Put an Implicit wait, this means that any search for elements on the page coul
        driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
        // Launch the URL
        driver.get("https://toolsqa.com/automation-practice-switch-windows/");
        // Click on the Button "Timing Alert"
        driver.findElement(By.name("Timing Alert")).click();
        System.out.println("Timer JavaScript Alert is triggered but it is not yet opened"
        // Create new WebDriver wait
        WebDriverWait wait = new WebDriverWait(driver, 10);
        // Wait for Alert to be present
        Alert myAlert = wait.until(ExpectedConditions.alertIsPresent());
        System.out.println("Either Pop Up is displayed or it is Timed Out");
        // Accept the Alert
        myAlert.accept();
        System.out.println("Alert Accepted");
        // Close the main window
        driver.close();
        }
}
```