Waits in Selenium : Waits are necessary to capture the web-application reaction.
Waits help user to create the Stable Automation.
Types of Wait in Selenium :
Sleep - java
PageLoadTimeOut
setScriptTimeOut
ImplicitlyWait
ExplicitWait
FluentWait

Sleep : Sleep() is a method present in thread class. It makes java code / selenium code to sleep for the specified time. Sleep() method will not worry about whether the element is present or not, it just sleeps till mentioned time. Sleep() is also called as static wait or blind wait, dead wait.

Timeout Interface in Selenium : Timeout is Interface in WebDriver Interface, which manage all waits of WebDriver Instances.
Timeouts interface has three abstract methods:
implicitlyWait
setScriptTimeout
pageLoadTimeout
There is no implementation present for these methods in Timeouts interface, the browser classes(FirefoxDriver, ChromeDriver..) provides the implementations for these methods because browser classes implements WebDriver Interface.

PageLoadTimeOut() : Page load timeout in selenium requests/set the time limit for a page to load. If page is not loaded within the given time frame selenium throws TimeOutException exception . Page load timeout is useful when we perform performance test. Setting Negative time limit makes the selenium to wait for the page load infinitely.
Program:
```
package WaitsinSelenium;
import java.time.Duration;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class PageLoadTimeout {

	public static void main(String[] args) {
		System.setProperty("webdriver.chrome.driver", "chromedriver");
		WebDriver driver = new ChromeDriver();
		driver.manage().window().maximize();
//		driver.manage().timeouts().pageLoadTimeout(5, TimeUnit.SECONDS);
		driver.manage().timeouts().pageLoadTimeout(Duration.ofSeconds(10));
		driver.get("https://edition.cnn.com/");
		driver.close();
	}
}
```

setScriptTimeOut() : setScriptTimeout sets the time limit for asynchronous script to finish execution. If process is not Completed within the given time frame selenium throws TimeOutException exception. The setScriptTimeout method affects only JavaScript code.
The default timeout for setScriptTimeout method is 0 (zero), if we don't set any time our executeAsyncScript method may fail because the javascript code may take more than zero seconds. So to avoid unnecessary failures we have to set the setScriptTimeout.
Code:-
driver.manage().timeouts().setScriptTimeout(11, TimeUnit.SECONDS);

ImplicitlyWait

The Implicit Wait in Selenium is used to tell the web driver to wait for a certain amount of time before it throws a "No Such Element Exception". The default setting is 0. Once we set the time, the web driver will wait for the element for that time before throwing an exception.

Implicit waits are used to provide a default waiting time (say 30 seconds) between each consecutive test step/command across the entire test script. Thus, the subsequent test step would only execute when the 30 seconds have elapsed after executing the previous test step/command.

Implicit wait will accept 2 parameters, the first parameter will accept the time as an integer value and the second parameter will accept the time measurement in terms of SECONDS, MINUTES, MILISECOND, MICROSECONDS, NANOSECONDS, DAYS, HOURS, etc.

Program. 10 sec = 2 sec

```
package WaitsinSelenium;

import java.time.Duration;
import java.util.List;
import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class ImplicitWaitProg {

	public static void main(String[] args) throws Exception {
		System.setProperty("webdriver.chrome.driver", "chromedriver");
		WebDriver driver = new ChromeDriver();
		driver.manage().window().maximize();
//		driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);
		driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));
		driver.get("https://www.google.co.in/");

		driver.findElement(By.xpath("//input[@name='q']")).sendKeys("seleni");

//		Thread.sleep(3000);
		List<WebElement>                    autoSugEle                    =
driver.findElements(By.xpath("//ul[@role='listbox']//li"));
```

```
                    for(int i=0;i<autoSugEle.size();i++)
                              System.out.println(autoSugEle.get(i).getText());

                    for(int i=0;i<autoSugEle.size();i++)
                    {
                              if(autoSugEle.get(i).getText().equals("selenium webdriver"))
                              {         //selenium.equals("selenium webdriver")
                                        //other than string comparison ==
//                                      String str = "ABC";
//                                      String str1 = new String("XYZ");
                                        //String compare .equals()
                                        autoSugEle.get(i).click();
                                        break;
                              }
                    }

                    Thread.sleep(5000);
                    driver.close();
          }

}
```

Explicit wait in Selenium : The explicit wait is used to tell the Web Driver to wait for certain conditions (Expected Conditions) or the maximum time exceeded before throwing an "ElementNotVisibleException" exception.
Explicit wait will be applicable for only one line (one condition), we have to use it with **ExpectedConditions** class.
WebDriverWait by default calls the Expected Condition every 500 milliseconds until it returns successfully.
Expected Condition will not impact findelement() and findelements() in Selenium.

Expected Conditions present in Explicit Wait :
alertIsPresent()
elementSelectionStateToBe(locator, selected)
elementToBeClickable(locator)
elementToBeSelected(locator)
frameToBeAvailableAndSwitchToIt(locator)
presenceOfElementLocated(locator)
textToBePresentInElement(locator, text) title()
titleIs(title)

Program :

```
package WaitsinSelenium;

import java.time.Duration;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.ExpectedCondition;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;

public class ExplicitWaitProg {

	public static void main(String[] args) throws Exception {
		System.setProperty("webdriver.chrome.driver", "chromedriver");
		WebDriver driver = new ChromeDriver();
		driver.manage().window().maximize();
		driver.get("https://chercher.tech/practice/explicit-wait");

		//Scenario-1
//		WebElement button1 =
driver.findElement(By.xpath("//button[@id='alert']"));
//		button1.click();
//		WebDriverWait w = new WebDriverWait(driver,
Duration.ofSeconds(30));
//		w.until(ExpectedConditions.alertIsPresent());
//		driver.switchTo().alert().accept();

		//Scenario-2
		WebElement button2 =
driver.findElement(By.xpath("//button[@id='populate-text']"));
		WebElement text = driver.findElement(By.xpath("//h2[@id='h2']"));
		button2.click();
		WebDriverWait w = new WebDriverWait(driver,
Duration.ofSeconds(30));
		w.until(ExpectedConditions.textToBePresentInElement(text, "Selenium
Webdriver"));

		driver.close();
	}
}
```