Selenium Waits

In **Selenium with Java**, **Waits** are essential for handling synchronization issues between your script and the web application under test. Web elements might take time to load, and interacting with them prematurely may cause your tests to fail. Selenium provides different types of waits to handle this situation.

🔹 Types of Waits in Selenium

Type of Wait	Description
Implicit Wait	Waits for a defined amount of time before throwing **`NoSuchElementException`**. Applies globally.
Explicit Wait	Waits for a certain condition to be met before proceeding. Used for specific elements.
Fluent Wait	Similar to explicit wait but allows polling frequency and ignoring specific exceptions.

✅ 1. Implicit Wait

🔸 Description:

- Set once and applies to all elements.
- Waits for the element to appear before throwing an exception.

🔸 Code:

```
java


import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import java.util.concurrent.TimeUnit;

public class ImplicitWaitExample {
    public static void main(String[] args) {
        WebDriver driver = new ChromeDriver();

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

        driver.get("https://example.com");
        // Selenium will wait up to 10 seconds for the element to be found
        driver.findElement(By.id("username")).sendKeys("admin");

        driver.quit();
    }
}
```

✅ 2. Explicit Wait

🔸 Description:

- Waits for a specific condition like visibility, presence, or clickability of an element.

🔸 Code:

```
java

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.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;

public class ExplicitWaitExample {
    public static void main(String[] args) {
        WebDriver driver = new ChromeDriver();
        driver.get("https://example.com");

        // Explicit Wait
        WebDriverWait wait = new WebDriverWait(driver, 10);
        WebElement loginButton = wait.until(
            ExpectedConditions.elementToBeClickable(By.id("login"))
        );

        loginButton.click();
        driver.quit();
    }
}
```

🔸 Common `ExpectedConditions`:

- **`visibilityOfElementLocated(By locator)`**
- **`elementToBeClickable(By locator)`**
- **`presenceOfElementLocated(By locator)`**
- **`textToBePresentInElement(By locator, "text")`**

✅ 3. Fluent Wait

🔸 Description:

- You can specify the polling interval and exceptions to ignore.

🔸 Code:

```
java

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.FluentWait;
import org.openqa.selenium.support.ui.Wait;

import java.time.Duration;
import java.util.NoSuchElementException;
import java.util.function.Function;

public class FluentWaitExample {
    public static void main(String[] args) {
        WebDriver driver = new ChromeDriver();
        driver.get("https://example.com");
```

```
        // Fluent Wait
        Wait<WebDriver> wait = new FluentWait<>(driver)
            .withTimeout(Duration.ofSeconds(15))
            .pollingEvery(Duration.ofSeconds(2))
            .ignoring(NoSuchElementException.class);

        WebElement element = wait.until(new Function<WebDriver, WebElement>() {
            public WebElement apply(WebDriver driver) {
                return driver.findElement(By.id("dynamicElement"));
            }
        });

        element.click();
        driver.quit();
    }
}
```

🚀 Best Practices

- Use **Explicit Wait** for dynamic elements instead of **`Thread.sleep()`**.
- Combine **Implicit** and **Explicit** waits carefully; using both together can lead to unpredictable wait times.
- Prefer **FluentWait** for custom polling or retry logic.

Would you like a **practice project** or **mini-assignment** to apply these waits in real scenarios?