Selenium WebDriver

Synchronization with Selenium.

Selenium Waits

SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

You Have Questions?

Software University

sli.do

#QA-FrontEnd

Table of Contents

Synchronization

What is it and why is it needed?

Synchronization Intro

Software University

- In **Test Automation**, there are **two components**:
  - **Application Under Test**
  - **Automation Tool** (e.g., Selenium)
- The **speed** at which the **Application processes commands** may **not align** with the speed at which the **Automation Tool sends them**
- This often leads to race conditions:
  - Sometimes the **browser** is **ready first** (tests work as intended)
  - Sometimes **Automation Tool executes commands first** (tests fail)
- This is a primary cause for **"Element Not Found" errors**

Synchronization Explained

- **Navigation commands** wait for '**readyState**' to be "**complete**" before continuing
- '**readyState**' only checks HTML asset loading
- JavaScript changes might not be ready, causing **element unavailability**
- Single-page apps **dynamically add** or **change element visibility**
- **Elements** must be **present** and **visible** for interaction

Synchronization Challenges Example

- On the page example, clicking "**Add a box!**" creates a new div element
- Clicking "**Reveal a new input**" displays a hidden text field
- Both transitions take a few seconds
- If Selenium clicks these buttons and tries to interact immediately, **it will fail**

What is Synchronization?

- Synchronization is a mechanism which involves more than one components to **work parallel with each other**
- Synchronization can be classified into two categories:
  - **Unconditional:**
    - Thread.Sleep()
  - **Conditional:**
    - Implicit waits
    - Explicit waits

Selenium Waits

Ensuring Synchronization in Automated Testing

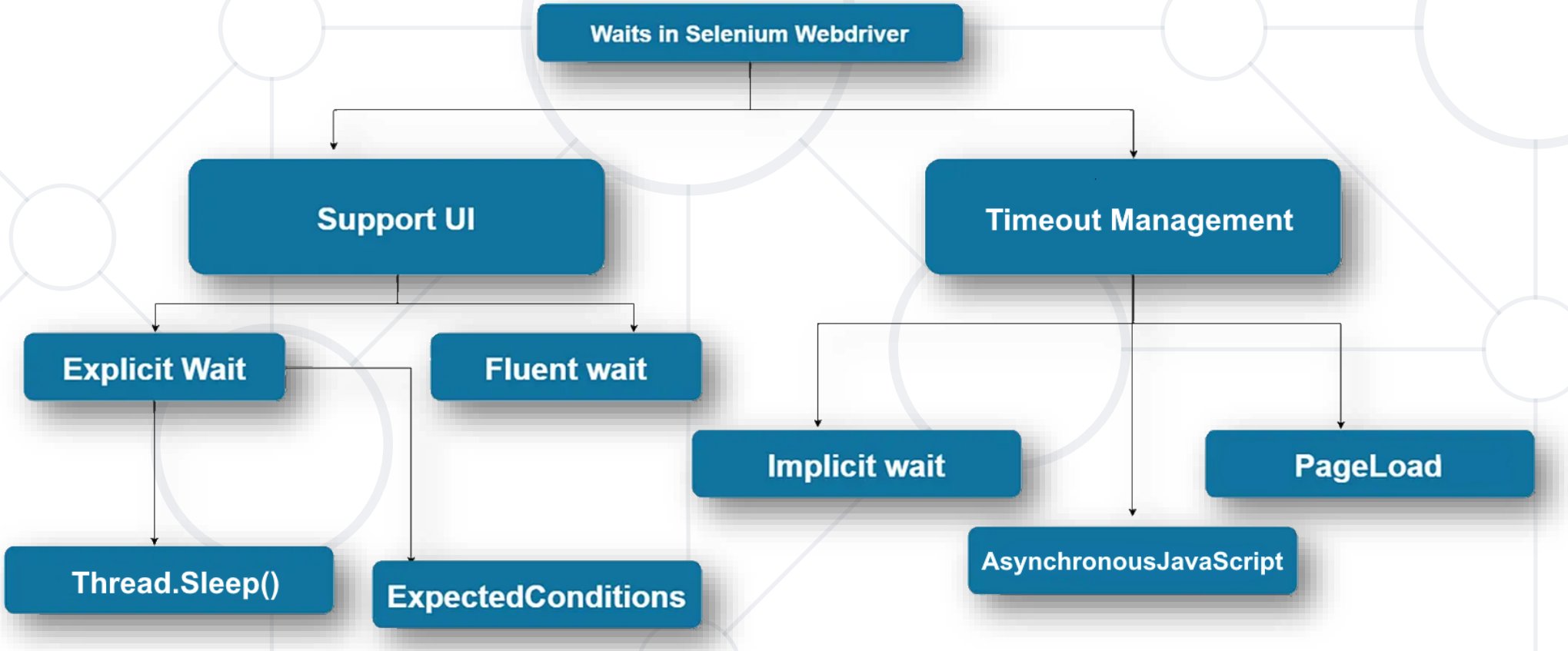
Selenium Waits

Software University

- To handle synchronization issues, the concept of "**waits**" is used
- Selenium provides different types of waits to handle synchronization in automated tests
- Selenium "**wait**" commands pause the test execution for a certain amount of time until the web elements are in the desired state before performing any actions on them

Structure of Selenium Waits

- The following schema illustrates the different types of waits in Selenium, categorized

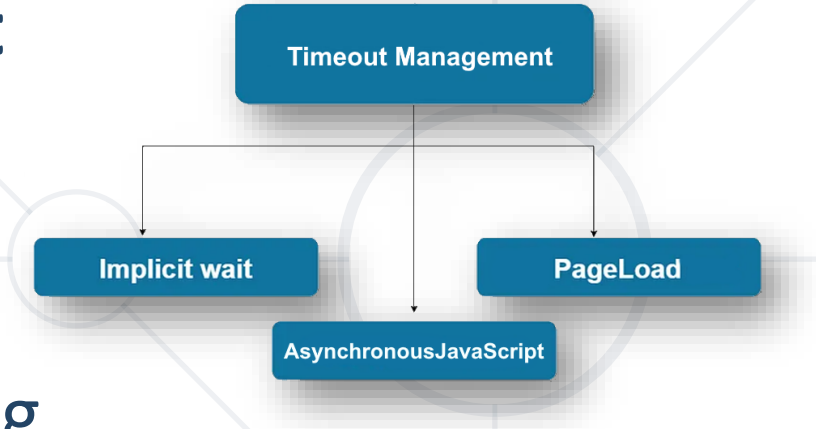


Timeout Management

Implicit Waits, Page Load, Asynchronous JavaScript

Implicit Wait

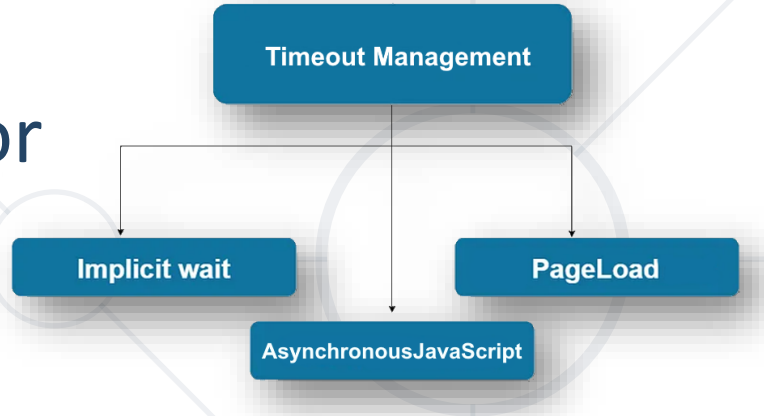
- You are already familiar with the **Implicit Wait**
- A global setting that applies to every element location call for the entire session
- By default, the implicit wait value is 0, meaning if an element is not found, it will immediately return an error
- When an implicit wait is set, the driver will wait for the specified duration before returning an error




```
driver.Manage().Timeouts().ImplicitWait =
  TimeSpan.FromSeconds(5);
```

Page Load Timeout

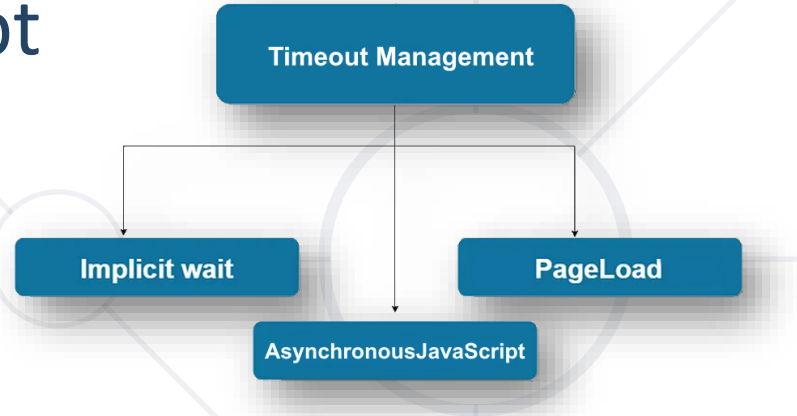
- Sets the maximum time the WebDriver will wait for a page to load before throwing an error
- Helps ensure that the page has fully loaded before any further actions are taken
- Essential for pages with heavy content or complex scripts that may take longer to load




```
driver.Manage().Timeouts().PageLoad =
  TimeSpan.FromSeconds(20);
```

Asynchronous JavaScript Timeout

- Sets the time to wait for an asynchronous script to finish execution before throwing an error
- Particularly useful for managing scripts that may take longer to run
- Useful for handling scenarios where JavaScript code execution takes time, such as AJAX calls or dynamically loaded content



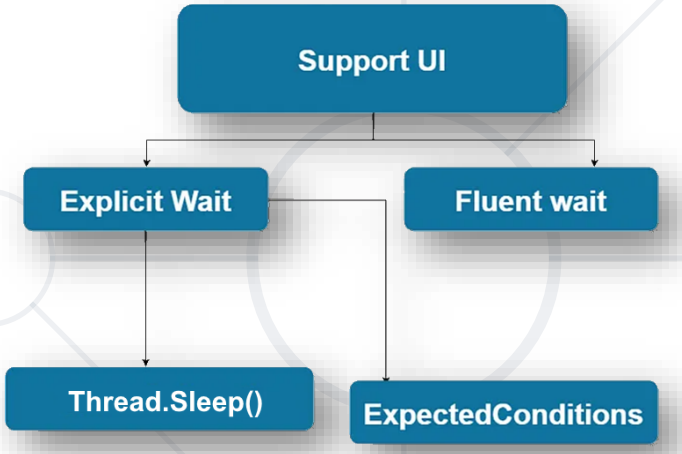

```
driver.Manage().Timeouts().AsynchronousJavaScript =
  TimeSpan.FromSeconds(15);
```

WaitHelpers

Explicit Waits, Fluent Waits

Explicit Wait

- Used to **halt the execution until a certain condition is met** or for a **maximum amount of time**
- **Flexible** and ideal where specific conditions need to be met before proceeding, e.g. waiting for an element to be clickable or visible
  - A condition is defined and a maximum time to wait for that condition to be met
  - Selenium checks the condition and proceed with the script as soon as the condition is satisfied



Explicit Wait

- Two classes for the implementation of Explicit Wait
  - WebDriverWait

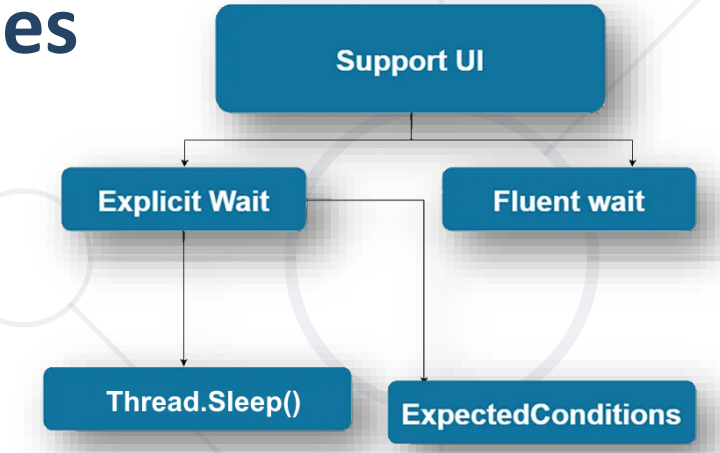
```
// Set up WebDriverWait with a timeout of 10 seconds
WebDriverWait wait = new WebDriverWait(driver,
  TimeSpan.FromSeconds(10));
```

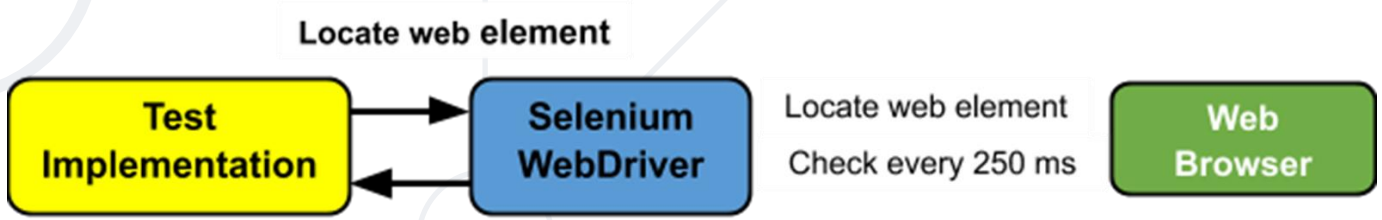
  - ExpectedConditions

```
// Wait until the element is visible
IWebElement element =
  wait.Until(ExpectedConditions.ElementIsVisible(By
  .Id("dynamicElement")));
```

Fluent Wait

- A type of explicit wait with **additional capabilities**
- Allows defining the frequency with which to check the condition
- Can ignore specific types of exceptions during the wait period
- Default value is 500ms
- Has timeout (e. g. 5 seconds) and polling interval (e. g. 250 ms)





Fluent Wait Example

- The attribute .pollingfrquency tells the driver to check every 250 ms, until 5 seconds are reached, before throwing an exception

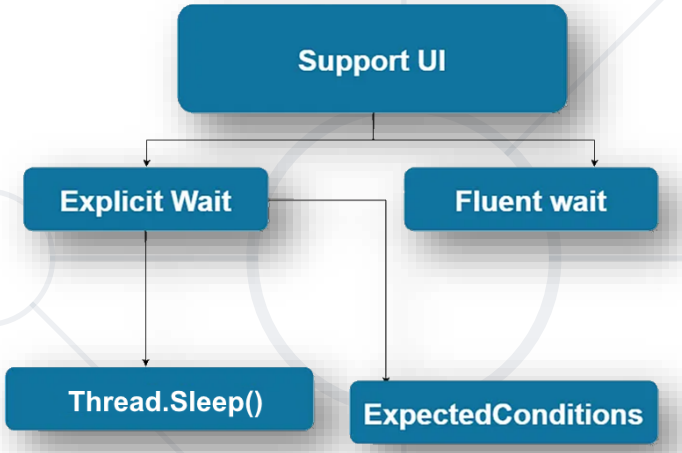
```
DefaultWait<IWebDriver> fluentWait = new
  DefaultWait<IWebDriver>(driver);


fluentWait.Timeout = TimeSpan.FromSeconds(5);


fluentWait.PollingInterval =
  TimeSpan.FromMilliseconds(250);
```

Thread.Sleep()

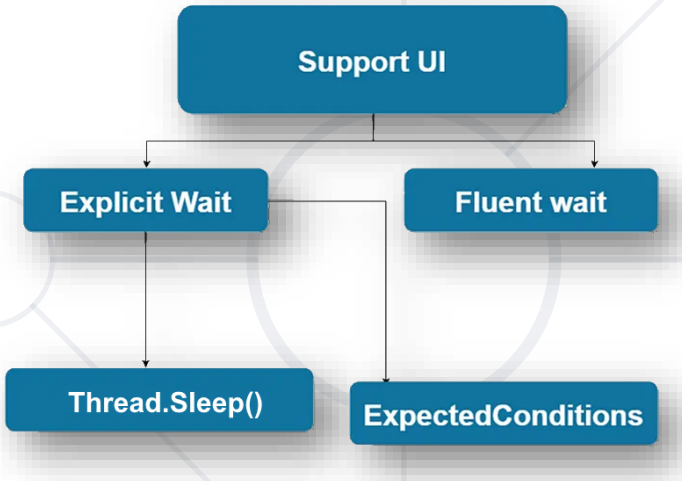
- Although it is not a Selenium feature, **Thread.Sleep()** is actually an example of explicit wait
- Common in most programming languages
- **Pauses** the **execution of the thread**
- Instructs the program to **do absolutely nothing** for a period of time, just sleep
- Doesn't matter what the application under test is up to, we don't care, our checks are having a nap time!



Thread.Sleep()

- The value provided is in milliseconds, so this code would sleep the test for 5 seconds

```
Thread.Sleep(5000);
```



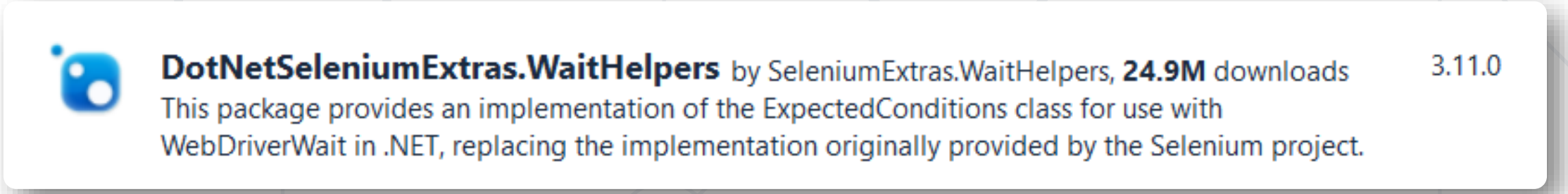
- It's important you are aware that "sleep" exists
- **Don't use it! It's a bad practice**
  - **Inefficiency**: Pauses the entire thread regardless of whether the application is ready, wasting valuable time
  - **Unreliable**: Fixed sleep periods may be too short (leading to failures) or too long (wasting time), as the exact wait time is often unpredictable
  - **Lack of Flexibility**: It does not adapt to the state of the application

Expected Conditions

Selenium Extras

ExpectedConditions (Selenium Extras)

- **ExpectedConditions** is a set of predefined conditions provided by Selenium that can be used with explicit waits
- These conditions help wait for specific states, such as element visibility, element clickability, etc.
- ExpectedConditions class simplifies explicit waiting in Selenium
- Install **DotNetSeleniumExtras.WaitHelpers** from NuGet



Common Conditions

- Some of the **common methods** used are:
  - **ElementIsVisible()** - Checks if the element is present in the DOM and is visible;
  - **ElementExists()** - The Element is present in the DOM;
  - **ElementToBeClickable(By Locator); ElementToBeClickable(IWebElement)** - The element is visible and enabled, such that you can click it;
  - **ElementToBeSelected()** - Checks if the element is selected

Common Conditions

- **TitleContains(string title)** - Title of the page contains the specified string
- **UrlMatches(string regex)** - Current URL is valid using specified regular expression
- **TextToBePresentInElementValue(IWebElement, string text)** - Specified text is present in the element's value attribute
- **TextToBePresentInElement()** - Specified text is present in the element

Exceptions

Handling Common Errors

Ignoring Exceptions While Waiting

Software University

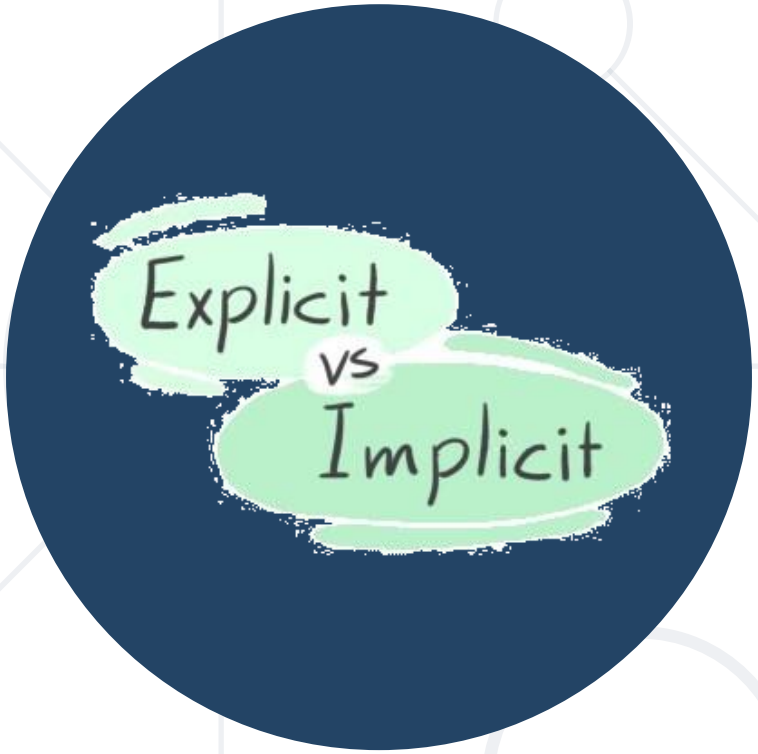
- Exceptional situations may occur while the driver searches for the web element in the DOM
- To ignore these errors while waiting for a condition, use the **IgnoreExceptionTypes()** method
- Common errors that can be ignored:
  - **NoSuchElementException**
  - **StaleElementReferenceException**
  - **ElementNotVisibleException**
  - **ElementNotSelectableException**
  - **NoSuchFrameException**

Ignoring Exceptions Example

- The syntax while using Explicit Wait is as follows:

```
WebDriverWait wait = new WebDriverWait(driver,
  TimeSpan.FromSeconds(10));


wait.IgnoreExceptionTypes(
  typeof(ElementNotSelectableException));
```



Implicit vs. Explicit

Understanding the Differences

Implicit vs. Explicit

- The implicit wait is asked to wait for a specific amount of time for the element to be available on the DOM of the page
- Applied globally to all elements on the webpage
- Does not require to meet any conditions

- Explicit wait is asked to wait till a certain condition is satisfied
- Not a global wait and is applied to a particular scenario
- Required to satisfy a particular condition

Summary

- Ensure timing and coordination in automation with Synchronization
- Selenium Waits - Mechanisms for handling delays and element availability
- Expected Conditions - Define predicates and checks to wait for specific states
- Exceptions - Handle errors and specify which to ignore
- Difference between Implicit and Explicit Waits

Questions?

SoftUni

Software University

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

СофтУни БУДИТЕЛ

Finance Academy

Diamond Partners

Software University

SmartIT

Coca-Cola HBC Bulgaria

SUPER HOSTING .BG

createX

Postbank

top EMPLOYER България Bulgaria 2024 FOR A BETTER WORLD OF WORK

DRAFT KINGS

THE CROWN IS YOURS

VIVACOM

INDEAVR

Serving the high achievers

AMBITIONED

PHAR VISION

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity

License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://about.softuni.bg/
- © Software University – https://softuni.bg