Document object model

**Code Explanation**

java

CopyEdit

WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));

WebElement playButton = wait.until(ExpectedConditions.elementToBeClickable(By.className("ytp-large-play-button")));

**1. Understanding Each Component**

**1.1 WebDriverWait**

**What is WebDriverWait?**

* WebDriverWait is a Selenium class used to implement **explicit waits**.
* It waits for a certain condition to be met **before** proceeding with the next step.
* This prevents NoSuchElementException, ElementNotInteractableException, etc.

**Syntax & Explanation:**

java

CopyEdit

WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));

* **driver** → This is your active Selenium WebDriver instance.
* **Duration.ofSeconds(10)** → This sets the maximum wait time to **10 seconds**.
* If the element appears before 10 seconds, execution continues immediately.

**1.2 Using ExpectedConditions**

java

CopyEdit

WebElement playButton = wait.until(ExpectedConditions.elementToBeClickable(By.className("ytp-large-play-button")));

**How ExpectedConditions.elementToBeClickable Works?**

* It waits until the element is both:
  1. **Present in the DOM** (Document Object Model).
  2. **Visible & Enabled** so it can be clicked.

**Breaking Down the Line**

1. wait.until(...)
   * This tells Selenium to **pause execution** until the given condition is met.
   * If the condition is met before 10 seconds, it proceeds immediately.
   * If not, Selenium throws a TimeoutException.
2. ExpectedConditions.elementToBeClickable(...)
   * Checks if the element is visible **and** enabled.
3. By.className("ytp-large-play-button")
   * Locates an element by its **class name** (ytp-large-play-button).
   * This is likely the "Play" button in a YouTube video.
4. WebElement playButton = ...
   * Once the element is found and clickable, it is stored in playButton.

**2. When Should You Use WebDriverWait?**

* When dealing with **dynamic elements** that take time to appear or become clickable.
* Useful for **handling AJAX calls, animations, and page load delays**.
* Prevents **NoSuchElementException or ElementNotInteractableException**.

**3. Example Scenario**

Imagine you are automating YouTube video playback:

java

CopyEdit

WebDriver driver = new ChromeDriver();

driver.get("https://www.youtube.com/watch?v=dQw4w9WgXcQ");

WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));

// Wait for the play button to be clickable

WebElement playButton = wait.until(ExpectedConditions.elementToBeClickable(By.className("ytp-large-play-button")));

// Click the play button

playButton.click();

**How This Works:**

1. Opens a YouTube video.
2. Waits for the play button to appear & be clickable.
3. Clicks the play button.

**4. Alternative Ways to Locate Elements**

If By.className("ytp-large-play-button") does not work, try:

**Using By.cssSelector**

java

CopyEdit

By.cssSelector(".ytp-large-play-button")

**Using By.xpath**

java

CopyEdit

By.xpath("//button[contains(@class, 'ytp-large-play-button')]")

**Using By.id (if available)**

java

CopyEdit

By.id("play-button")

**5. Handling TimeoutException**

If the element does not become clickable within the given time, Selenium throws:

makefile

CopyEdit

org.openqa.selenium.TimeoutException: Expected condition failed: waiting for element to be clickable

**Solution:**

* **Increase timeout**:

java

CopyEdit

WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(20));

* **Use alternative locators** (e.g., XPath, CSS Selector).
* **Check if the element is inside an iframe** and switch to it before interacting.

**Final Thoughts**

* WebDriverWait is **essential** for handling **dynamic elements**.
* ExpectedConditions.elementToBeClickable ensures that an element is **both visible and enabled** before interaction.
* Helps prevent **synchronization issues** in Selenium test automation.