Best Practice: Use Explicit Waits

There are many situations in which your test script may run ahead of the website or application you're testing, resulting in timeouts and a failing test. For example, you may have a dynamic content element that, after a user clicks on it, a loading appears for five seconds. If your script isn't written in such a way as to account for that five second load time, it may fail because the next interactive element isn't available yet.

The general advice from the Selenium community on how to handle this is to use explicit waits. While you could also use implicit waits, an implicit wait only waits for the appearance of certain elements on the page, while an explicit wait can be set to wait for broader conditions. Selenium guru Dave Haeffner provides an excellent example of why you should use explicit waits on his Elemental Selenium blog. Whether you use explicit or implicit waits, you should not mix the two types in the same test.

These code samples, from the SeleniumHQ documentation on explicit and implicit waits, shows how you would use an explicit wait. In their words, this sample shows how you would use an explicit wait that "waits up to 10 seconds before throwing a `TimeoutException`, or, if it finds the element, will return it in 0 - 10 seconds. `WebDriverWait` by default calls the `ExpectedCondition` every 500 milliseconds until it returns successfully. A successful return for `ExpectedCondition` type is Boolean return `true`, or a not null return value for all other `ExpectedCondition` types."

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait # available since 2.4.0
from selenium.webdriver.support import expected_conditions as EC # available since 2.26.0

ff = webdriver.Firefox()
ff.get("http://somedomain/url_that_delays_loading")
try:
    element = WebDriverWait(ff, 10).until(EC.presence_of_element_located((By.ID, "myDynamicElement")))
finally:
    ff.quit()
```

```
WebDriver driver = new FirefoxDriver();
driver.get("http://somedomain/url_that_delays_loading");
WebElement myDynamicElement = (new WebDriverWait(driver, 10))
  .until(ExpectedConditions.presenceOfElementLocated(By.id("myDynamicElement")));
```

```
IWebDriver driver = new FirefoxDriver();
driver.Url = "http://somedomain/url_that_delays_loading";
WebDriverWait wait = new WebDriverWait(driver, TimeSpan.FromSeconds(10));
IWebElement myDynamicElement = wait.Until<IWebElement>((d) =>
    {
        return d.FindElement(By.Id("someDynamicElement"));
    });
```

```
require 'selenium-webdriver'

driver = Selenium::WebDriver.for :firefox
driver.get "http://somedomain/url_that_delays_loading"

wait = Selenium::WebDriver::Wait.new(:timeout => 10) # seconds
begin
  element = wait.until { driver.find_element(:id => "some-dynamic-element") }
ensure
  driver.quit
end
```