In Playwright, **fixtures** are an essential part of how Playwright Test (@playwright/test) manages setup and teardown in tests. Fixtures are pre-configured, reusable setups that can be used in different test cases to provide a consistent environment. Playwright provides **built-in fixtures** as well as the capability to create **custom fixtures**. Let’s explore the types of fixtures, their syntax, and their differences.

**Types of Fixtures in Playwright**

1. **Built-in Fixtures**: Provided by Playwright Test, these include essential elements like browser instances, contexts, and pages.
2. **Custom Fixtures**: You can create your own fixtures to handle specific setups that are reused across tests, such as database connections, mock servers, or pre-authenticated sessions.

**1. Built-in Fixtures**

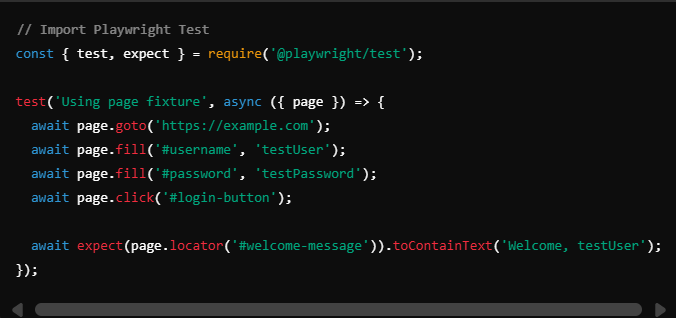
Playwright Test comes with several built-in fixtures that simplify the browser automation process. These fixtures are automatically managed by Playwright, meaning they are created before the test runs and cleaned up after it finishes. Below are the commonly used built-in fixtures:

**Built-in Fixtures Provided by Playwright Test**

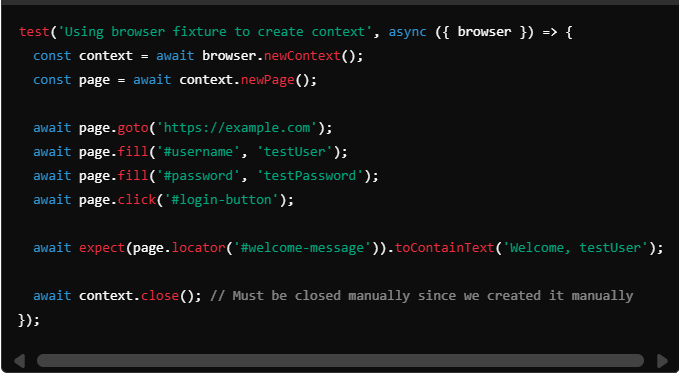
* **browser**: Represents the browser instance (e.g., Chromium, Firefox, WebKit).
  + Syntax: async ({ **browser** }) => { ... }
* **context**: Represents a new browser context, which provides isolation between different tests.
  + Syntax: async ({ **context** }) => { ... }
* **page**: Represents a new page (i.e., tab) within a browser context. It is commonly used to perform actions like navigating to URLs, filling forms, etc.
  + Syntax: async ({ **page** }) => { ... }

**Examples of Using Built-in Fixtures**

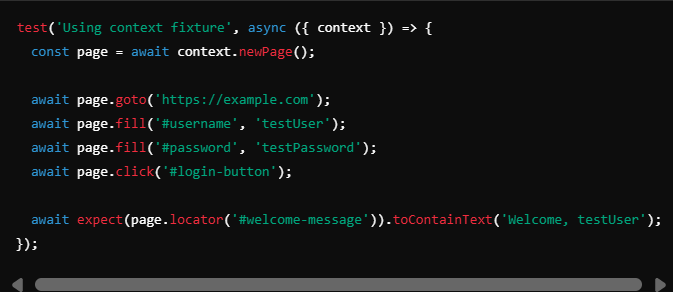
1. **Using page Fixture**:
   * The page fixture is the most commonly used and is created automatically before each test.
   * You don’t need to manage the lifecycle manually; Playwright Test will handle it for you.



1. **Using browser Fixture**:
   * The browser fixture gives you more control, such as creating multiple contexts manually.
   * You need to close the resources manually when using browser or context.



1. **Using context Fixture**:
   * The context fixture allows each test to have an independent browser context, automatically managed by Playwright.

****

**2. Custom Fixtures**

Playwright Test allows you to extend the existing functionality by creating **custom fixtures**. Custom fixtures help you set up more complex scenarios, such as logged-in states, mock servers, or shared database connections.

**Creating Custom Fixtures**

To create custom fixtures, you extend the Playwright’s test fixture by using test.extend(). Below is the syntax and an example:

* **Syntax**: const test = base.extend({ fixtureName: async ({}, use) => { ... } });

