Syntax and Usage of async and await

Introduction

Asynchronous programming in C# helps improve performance by allowing applications to handle tasks in the background without freezing the main thread. This guide shows you how to create and use asynchronous methods with async and await for efficient task management.

Step-by-Step Instructions

1. Define an Asynchronous Method

a. Use the **`async`** keyword to mark a method as asynchronous.

b. How to do it: Add **`async`** before the method's return type.

```
public async Task GetDataAsync()
{
  // Method logic here
}
```

2. Use the Await Keyword

a. The **`await`** keyword pauses execution until a background task finishes.

b. How to do it: Place **`await`** before methods returning a **`Task`**.

```
var data = await GetDataFromApi();
```

3. Set the Correct Return Type:

a. For methods returning values, use **`Task<T>`**; for void methods, use **`Task`**.

b. How to do it:

```
public async Task<string> GetDataAsync()
{
  var data = await GetDataFromApi();
  return data;
}
```

4. Handle Errors:

a. Manage exceptions in async methods using try-catch blocks.

b. How to do it:

```
try
{
  var data = await GetDataFromApi();
}
catch (Exception ex)
{
  Console.WriteLine(ex.Message);
}
```

Conclusion

Using **`async`** and **`await`** allows you to run tasks efficiently in the background, improving application responsiveness without interrupting the main program flow.