

Asynchronous Programming in C#

By

Narasimha Rao T

Microsoft.Net FSD Trainer

Professional Development Trainer

tnrao.trainer@gmail.com

Introduction to Async Programming

1. Introduction to Async Programming

What is Asynchronous Programming?

Asynchronous programming allows your program to perform tasks (like I/O operations) without blocking the main execution thread. It improves **responsiveness**, especially in applications where **waiting for I/O** (e.g., network calls, file access) can lead to delays.







Benefits:

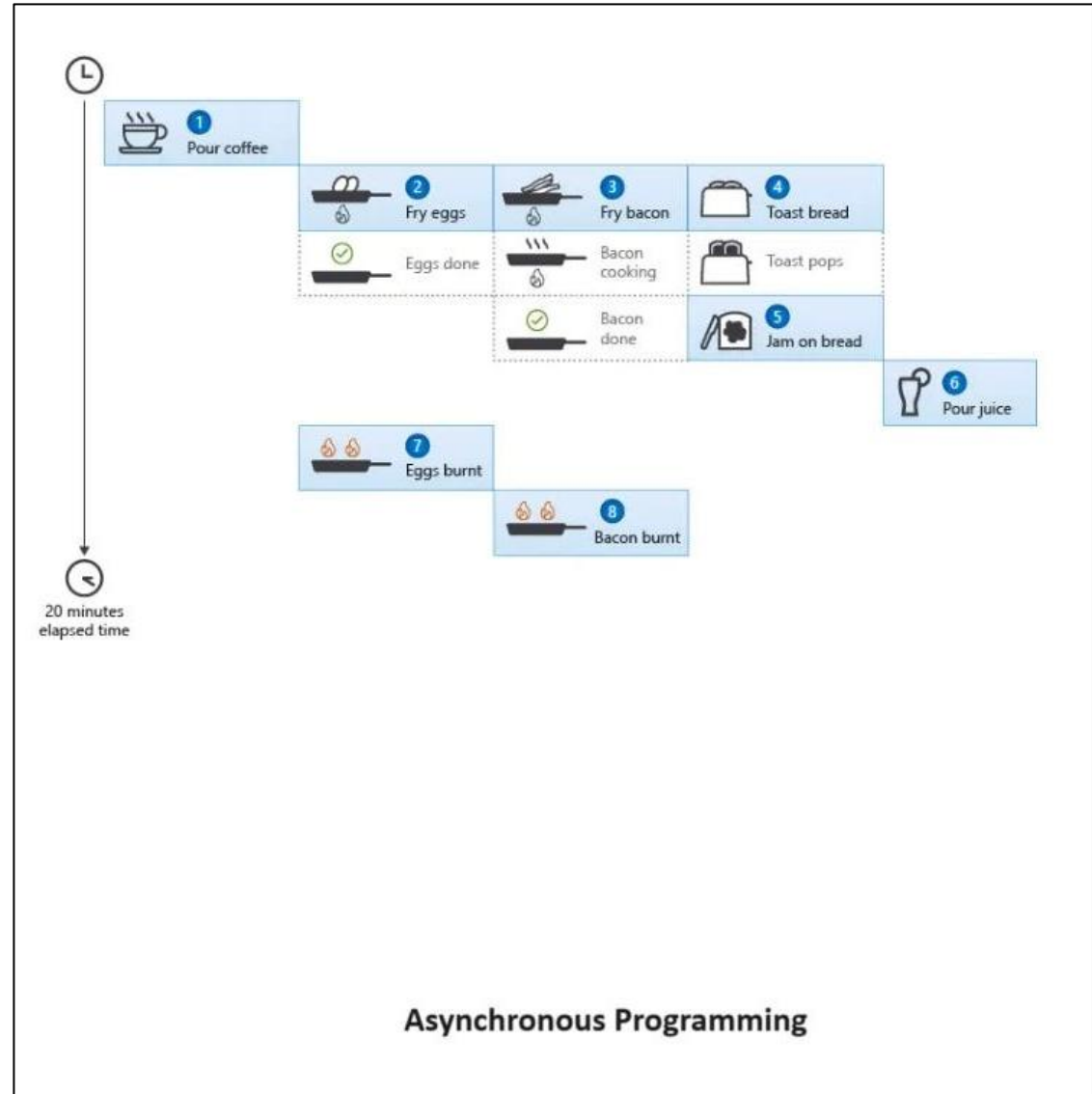
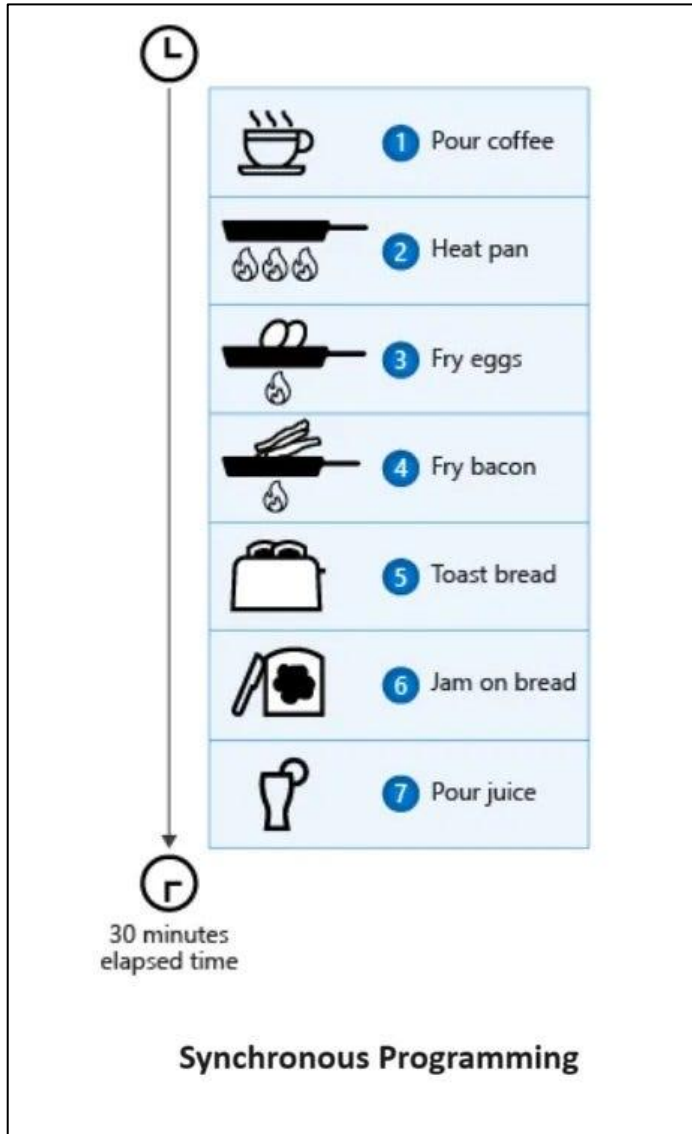
- Better resource utilization.
- Enhanced responsiveness (especially in UI applications).
- Scalability in server apps (e.g., handling many concurrent API requests).

Asynchronous vs. synchronous programming

Understanding Async Programming with a Breakfast Example

Prepare breakfast that includes:

1. Making coffee 
2. Frying eggs 
3. Toasting bread 
4. Applying butter and jam  
5. Pouring orange juice 



2. Synchronous vs Asynchronous Execution

Aspect	Synchronous	Asynchronous
Execution	Blocks the thread	Frees thread for other work
Performance	Slower with I/O-bound work	Faster when awaiting I/O
Responsiveness	UI may freeze	UI remains responsive
Code Simplicity	Easier to write/read	Slightly more complex

Example:

```
// Synchronous
string content = File.ReadAllText("data.txt");

// Asynchronous
string content = await File.ReadAllTextAsync("data.txt");
```

3. Task and Task<T>

Task

Represents an asynchronous operation that can return **void**.

```
public Task DoWorkAsync()  
{  
    return Task.Run(() => {  
        // Simulated work  
        Thread.Sleep(2000);  
    });  
}
```


Task<T>

Represents an asynchronous operation that **returns a value**.

```
public Task<int> CalculateAsync()  
{  
    return Task.Run(() => {  
        return 42;  
    });  
}
```

4. async and await Keywords

`async` keyword:

- Marks a method as asynchronous.
- Must return `Task`, `Task<T>`, or `void` (event handlers only).

`await` keyword:

- Pauses method execution until the awaited task completes.
- Allows the method to return control to the caller (non-blocking).

Example:

```
public async Task<string> GetDataAsync()  
{  
    HttpClient client = new HttpClient();  
    string result = await client.GetStringAsync("https://api.example.com/data");  
    return result;  
}
```

5. Real-World Use Cases

UI Applications (WinForms, WPF)

- Keep UI responsive while performing background tasks.

```
private async void btnDownload_Click(object sender, EventArgs e)
{
    btnDownload.Enabled = false;
    string data = await GetDataAsync();
    txtResult.Text = data;
    btnDownload.Enabled = true;
}
```

Web APIs

- Handle multiple requests concurrently without blocking threads.

```
[HttpGet]
public async Task<IActionResult> GetData()
{
    var data = await _dataService.GetDataAsync();
    return Ok(data);
}
```

File Access

- Use async file methods to prevent UI freezing or blocking threads.

```
public async Task SaveToFileAsync(string path, string content)
{
    await File.WriteAllTextAsync(path, content);
}
```

6. Exception Handling in Async Methods

Use try-catch inside the async method:

```
public async Task<string> SafeGetDataAsync()
{
    try
    {
        HttpClient client = new HttpClient();
        return await client.GetStringAsync("https://invalid.url");
    }
    catch (HttpRequestException ex)
    {
        // Handle network error
        return $"Error: {ex.Message}";
    }
}
```

Exceptions thrown in async methods must be awaited to be caught!

7. Real-Time Application Scenarios

Application Type	Use Case
Desktop App	Loading data without freezing UI
Mobile App	Accessing GPS, Camera, or Internet
Web API	Calling databases or external services
Games	Asynchronous loading of assets
Cloud Apps	Concurrent file processing or job queues

8. Quiz Questions

1. What does the `async` keyword indicate in a method?

- a) The method runs on a new thread
- b) The method returns immediately
- c) The method contains `await` and runs asynchronously

2. Which return types are allowed for an async method?

- a) Task<T>
- b) void
- c) Task
- d) All of the above

3. Which of the following statements is true about `await` ?

- a) It blocks the thread
- b) It suspends execution and allows continuation
- c) It can only be used inside a constructor

4. What happens if you do not `await` an async method?

- a) It still runs to completion
- b) Exceptions can be missed
- c) Both a and b

5. Which of these scenarios benefits most from async programming?

- a) CPU-intensive calculations
- b) Waiting for file download

Q & A

Narasimha Rao T

tnrao.trainer@gmail.com