

Better Parallel Code with C# Channels

Jeremy Clark

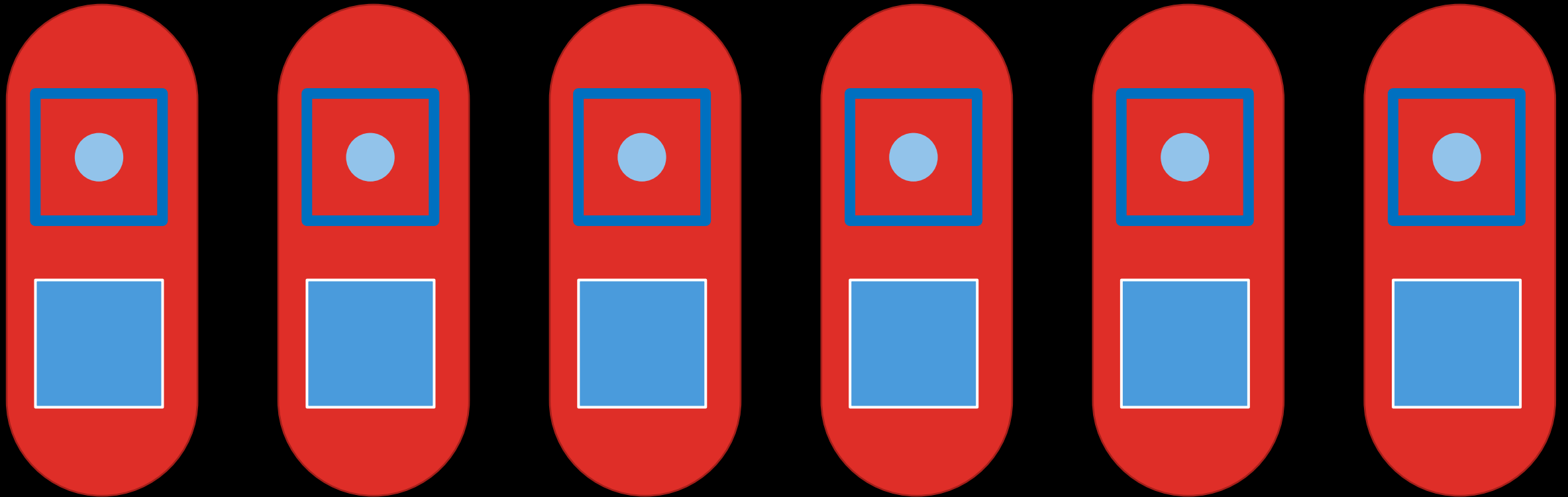
www.jeremybytes.com

@jeremybytes

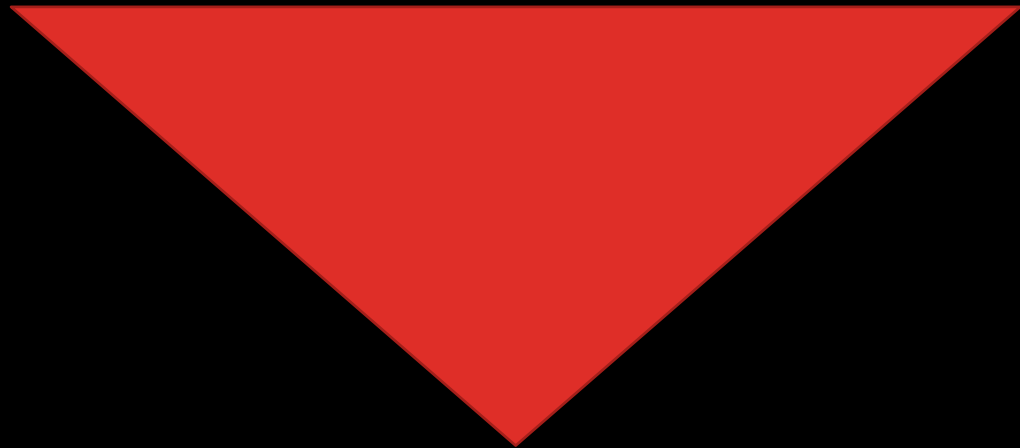
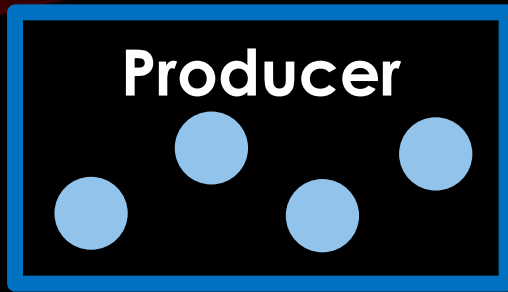
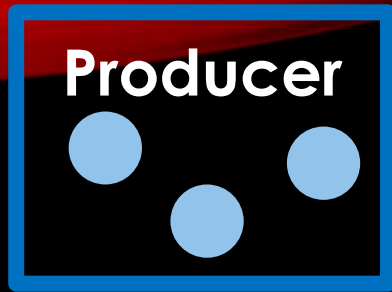
Channel<T>

- Rolled in to .NET Core 3.0
Prior to that, there was a separate NuGet package
- Communicate between async methods/functions
- Similar to a concurrent queue, but with reader/writer optimizations and signaling built-in
- Not for communicating between distributed applications

Get Data / Use Data



Producer / Consumer



Consumer

Producers and consumers can communicate asynchronously through a channel.



Major Operations

- Channel<T>
 - Create a channel
 - Write to a channel
 - Read from a channel
 - Mark the channel “complete”

Creating a Channel

- `CreateBounded<T>`
 - Creates a channel of a specific size
 - If the channel is full, writers are blocked until space is available

```
var channel = Channel.CreateBounded<Person>(10);
```

Channel Reader / Writer

Reader property

- ChannelReader<T>
 - ReadAllAsync()

Writer property

- ChannelWriter<T>
 - WriteAsync()
 - Complete()

Writing to a Channel

- `writer.WriteAsync()`
 - Writes an item to the channel

```
await writer.WriteAsync(item);
```

Marking a Channel “Complete”

- `writer.Complete()`
 - Indicates that no further items will be written
 - Writing to a “complete” channel throws an exception
 - Reading from a “complete” channel will continue normally until the channel is empty

Reading from a Channel

- `reader.ReadAllAsync()`
 - Returns an `IEnumerable<T>`

```
await foreach (var item in reader.ReadAllAsync())  
{  
    // use item here  
}
```

- If the channel is empty, the loop will pause until an item is available.
- If the channel is “complete”, the loop will exit.

Other Stuff

- `ChannelReader<T>`
 - `WaitToReadAsync()`
 - `ReadAsync()`
 - `TryRead()`
- `ChannelWriter<T>`
 - `WaitToWriteAsync()`
 - `TryWrite()`
 - `TryComplete()`

Other Stuff

- `Channel.CreateUnbounded<T>`
- `ChannelCreationOptions`
 - `SingleReader`
 - `SingleWriter`
 - `AllowSynchronousContinuations`These allow for compiler and runtime optimizations



Resources

Code Samples & Resources

[https://github.com/jeremybytes/
sdd-2022](https://github.com/jeremybytes/sdd-2022)

An abstract graphic at the top of the slide featuring a series of overlapping, wavy bands in shades of orange, red, yellow, and green, set against a black background.

Thank You!

Jeremy Clark

- www.jeremybytes.com
- github.com/jeremybytes
- @jeremybytes