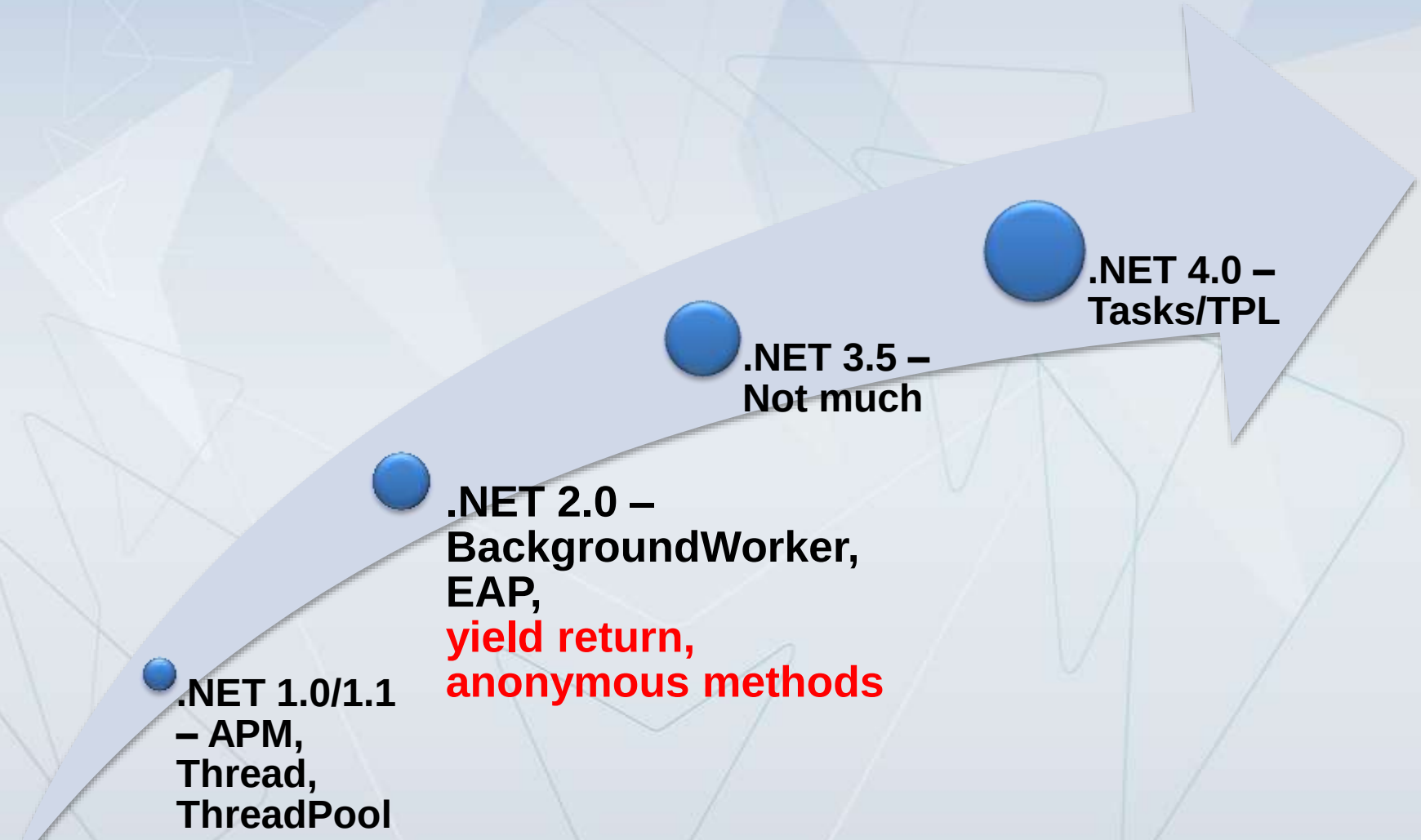
Asynchronous programming with C# 5.0 & .NET 4.5

Avner Shahar-Kashtan
Consultant, CodeValue

avnerk@codevalue.net
http://bits.strawjackal.org
blogs.microsoft.co.il/blogs/avnerk

Evolution of Async Programming



CodeValue

Asynchronous Programming Model

- BeginXXX/EndXXX, IAsyncResult
- Code is scattered
- Explicit synchronization
- Explicit Thread/Threadpool use

- Demo

CodeValue

Event-based Async Programming

- XXXAsync/XXXCompleted
- Code is scattered
- Implicit synchronization
- Explicit Thread/Threadpool use

- Demo

CodeValue

Asynchronous Methods

- **Goal: Just like synchronous programming**
- **Compare the state of Remote Execution (Remoting/WS/WCF)**
- **Syntax hides complexity**
  - “async” modifier marks method or lambda as asynchronous
  - “await” operator yields control until awaited task completes

Task<T>

- **Represents “ongoing operation”**
  - Could be async I/O, background worker, anything...
  - Single object for status, result and exceptions
- **Composable callback model**
  - var task2 = task1.ContinueWith(t => … t.Result …);
  - The “await” operator rewrites to continuations
- **Combinators**
  - WhenAll, WhenAny, Delay, etc.

Composable Tasks

```
async Task ProcessFeedAsync(string url) {
    var text = await DownloadFeedAsync(url);
    var doc = ParseFeedIntoDoc(text);
    await SaveDocAsync(doc);
    ProcessLog.WriteEntry(url);
}
```

```
Task ProcessFeedAsync(string url) {
 var task = DownloadFeedAsync(url);
 task.ContinueWith(t => ParseFeedIntoDoc(t.Result));
 task.ContinueWith(t => SaveDocAsync(t.Result));
 task.ContinueWith(t => ProcessLog.WriteEntry(url));
 task.Start();
 return task;
}
```

Async Code Locality

- **What yield return is to collections**

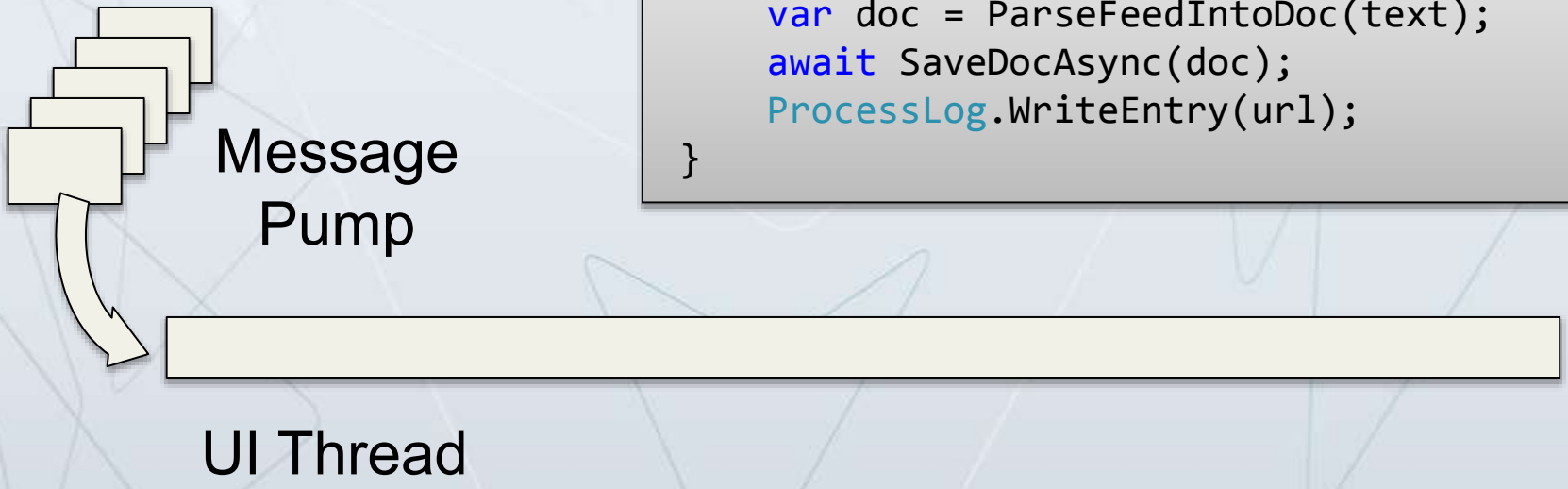
```
IEnumerable<string> GetLines() {
    yield return “Line 1”;
    yield return “Line 2”;
    yield return “Line 3”;
}
```

CodeValue

Asynchronous Control Flow

```
async void DoWorkAsync() {
    var t1 = ProcessFeedAsync("www.acme.com/rss");
    var t2 = ProcessFeedAsync("www.xyznews.com/rss");
    await Task.WhenAll(t1, t2);
    DisplayMessage("Done");
}
```

```
async Task ProcessFeedAsync(string url) {
    var text = await DownloadFeedAsync(url);
    var doc = ParseFeedIntoDoc(text);
    await SaveDocAsync(doc);
    ProcessLog.WriteEntry(url);
}
```

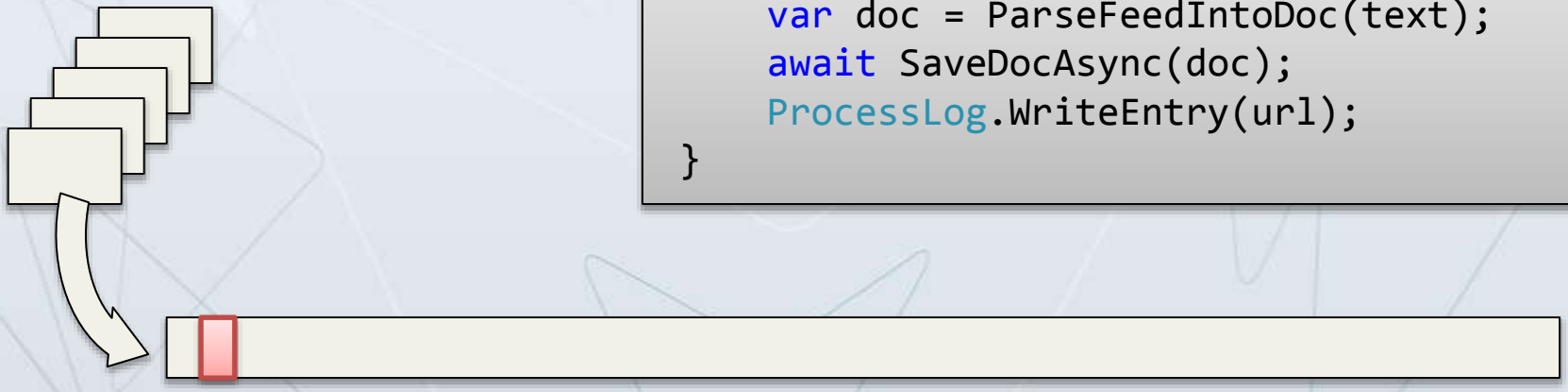


CodeValue

Asynchronous Control Flow

```
async void DoWorkAsync() {
    var t1 = ProcessFeedAsync("www.acme.com/rss");
    var t2 = ProcessFeedAsync("www.xyznews.com/rss");
    await Task.WhenAll(t1, t2);
    DisplayMessage("Done");
}
```

```
async Task ProcessFeedAsync(string url) {
    var text = await DownloadFeedAsync(url);
    var doc = ParseFeedIntoDoc(text);
    await SaveDocAsync(doc);
    ProcessLog.WriteEntry(url);
}
```

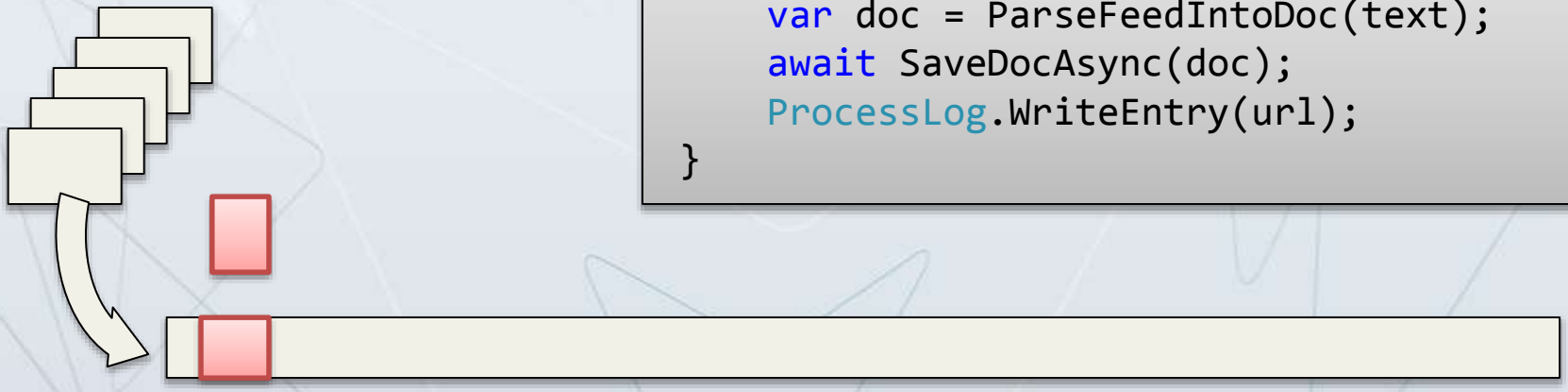


CodeValue

Asynchronous Control Flow

```
async void DoWorkAsync() {
    var t1 = ProcessFeedAsync("www.acme.com/rss");
    var t2 = ProcessFeedAsync("www.xyznews.com/rss");
    await Task.WhenAll(t1, t2);
    DisplayMessage("Done");
}
```

```
async Task ProcessFeedAsync(string url) {
    var text = await DownloadFeedAsync(url);
    var doc = ParseFeedIntoDoc(text);
    await SaveDocAsync(doc);
    ProcessLog.WriteEntry(url);
}
```

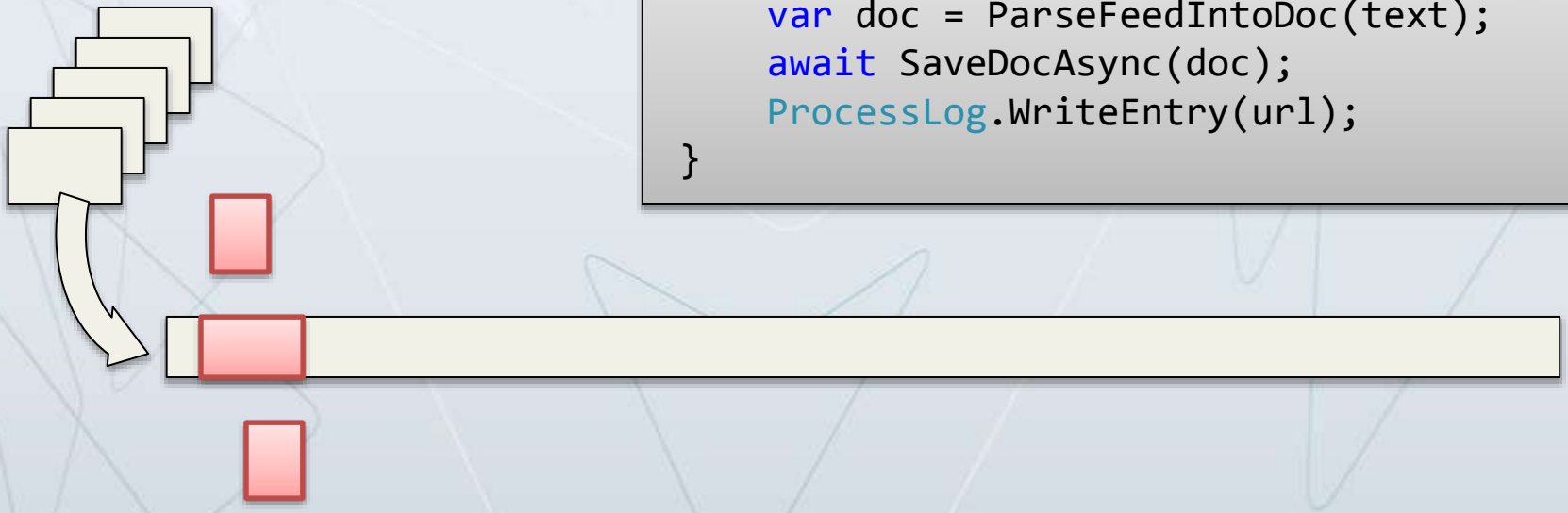


CodeValue

Asynchronous Control Flow

```
async void DoWorkAsync() {
    var t1 = ProcessFeedAsync("www.acme.com/rss");
    var t2 = ProcessFeedAsync("www.xyznews.com/rss");
    await Task.WhenAll(t1, t2);
    DisplayMessage("Done");
}
```

```
async Task ProcessFeedAsync(string url) {
    var text = await DownloadFeedAsync(url);
    var doc = ParseFeedIntoDoc(text);
    await SaveDocAsync(doc);
    ProcessLog.WriteEntry(url);
}
```

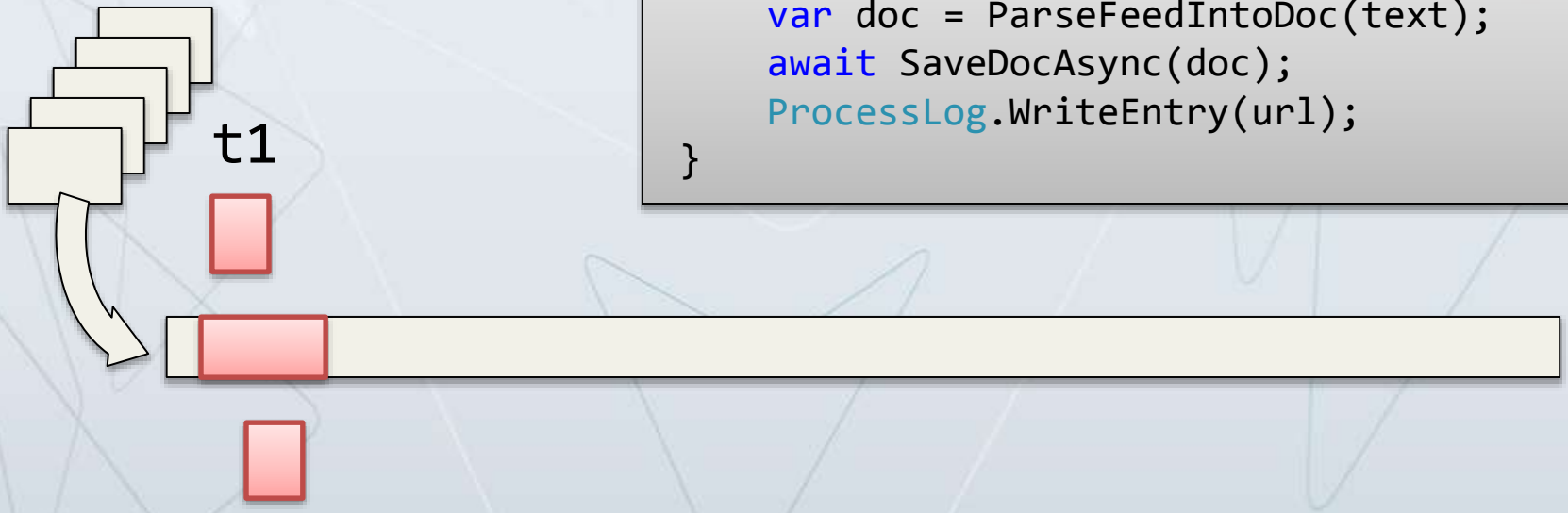


CodeValue

Asynchronous Control Flow

```
async void DoWorkAsync() {
    var t1 = ProcessFeedAsync("www.acme.com/rss");
    var t2 = ProcessFeedAsync("www.xyznews.com/rss");
    await Task.WhenAll(t1, t2);
    DisplayMessage("Done");
}
```

```
async Task ProcessFeedAsync(string url) {
    var text = await DownloadFeedAsync(url);
    var doc = ParseFeedIntoDoc(text);
    await SaveDocAsync(doc);
    ProcessLog.WriteEntry(url);
}
```

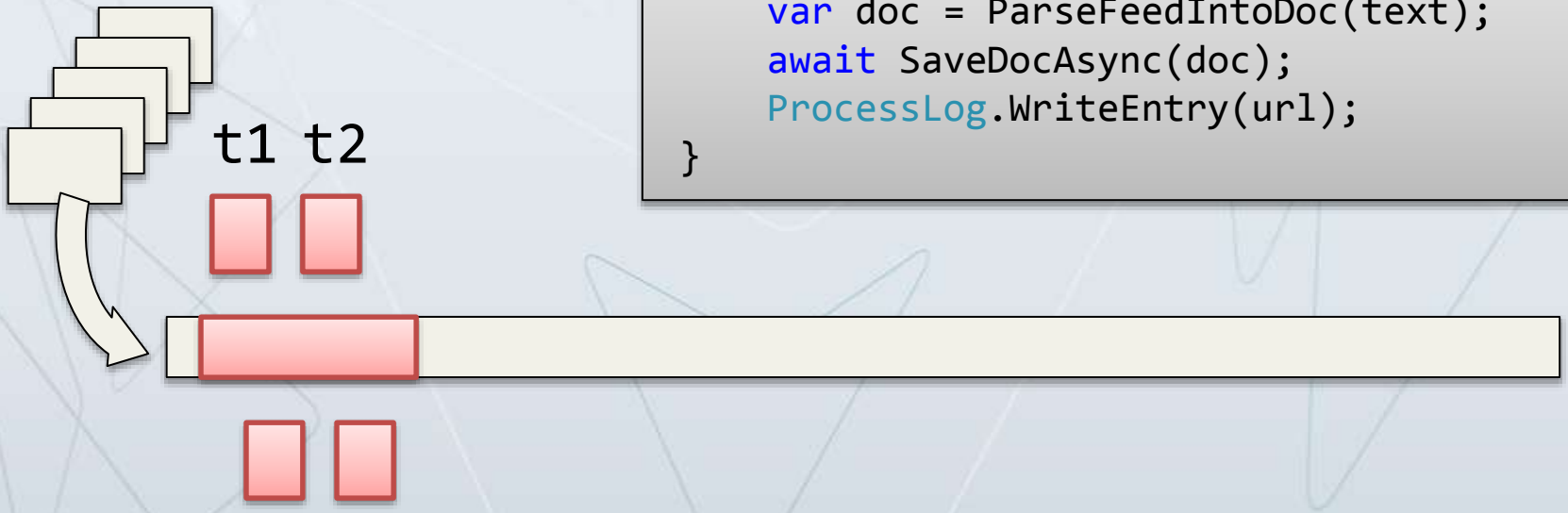


CodeValue

Asynchronous Control Flow

```
async void DoWorkAsync() {
    var t1 = ProcessFeedAsync("www.acme.com/rss");
    var t2 = ProcessFeedAsync("www.xyznews.com/rss");
    await Task.WhenAll(t1, t2);
    DisplayMessage("Done");
}
```

```
async Task ProcessFeedAsync(string url) {
    var text = await DownloadFeedAsync(url);
    var doc = ParseFeedIntoDoc(text);
    await SaveDocAsync(doc);
    ProcessLog.WriteEntry(url);
}
```

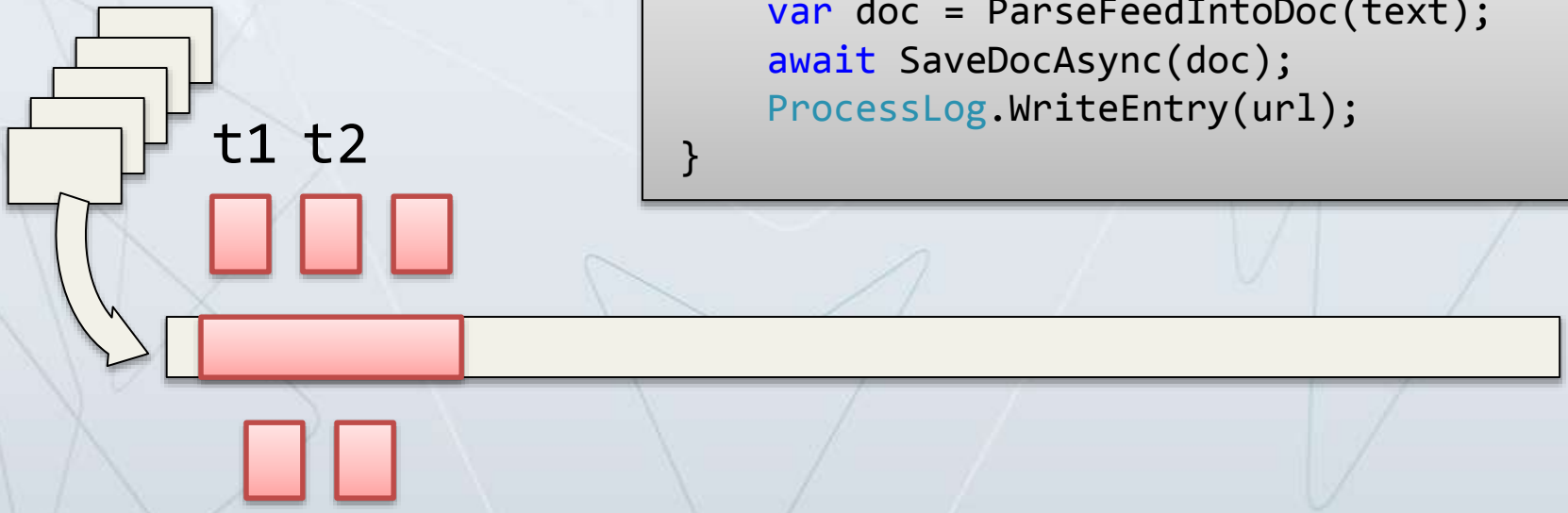


CodeValue

Asynchronous Control Flow

```
async void DoWorkAsync() {
    var t1 = ProcessFeedAsync("www.acme.com/rss");
    var t2 = ProcessFeedAsync("www.xyznews.com/rss");
    await Task.WhenAll(t1, t2);
    DisplayMessage("Done");
}
```

```
async Task ProcessFeedAsync(string url) {
    var text = await DownloadFeedAsync(url);
    var doc = ParseFeedIntoDoc(text);
    await SaveDocAsync(doc);
    ProcessLog.WriteEntry(url);
}
```

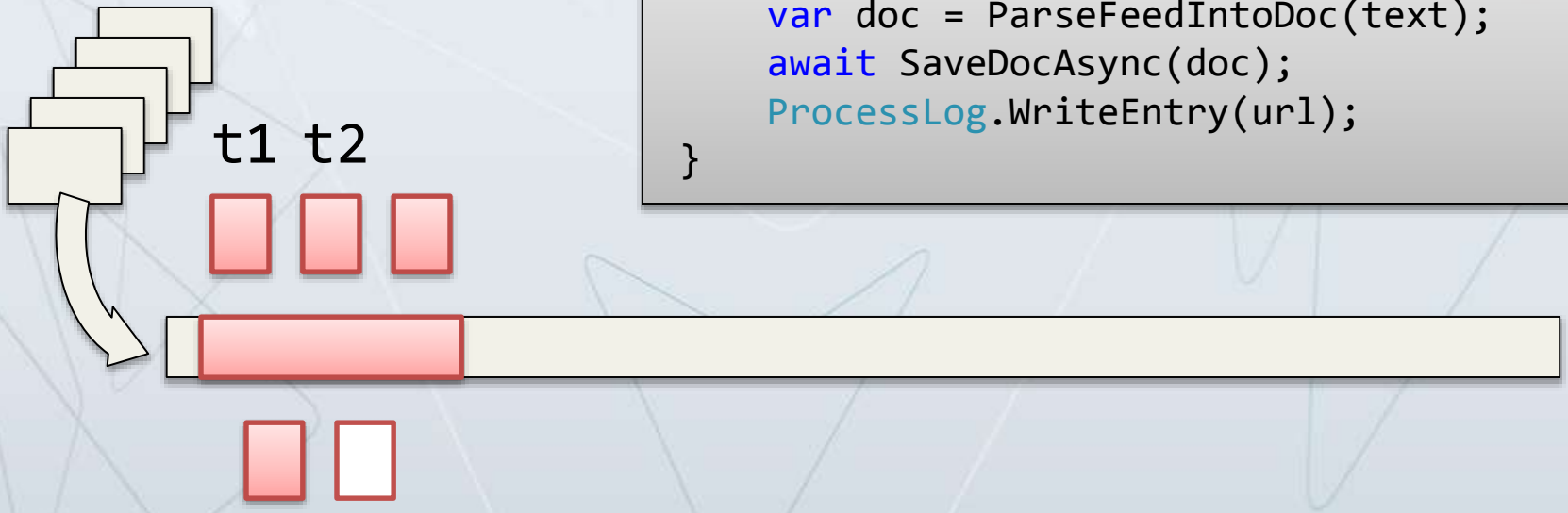


CodeValue

Asynchronous Control Flow

```
async void DoWorkAsync() {
    var t1 = ProcessFeedAsync("www.acme.com/rss");
    var t2 = ProcessFeedAsync("www.xyznews.com/rss");
    await Task.WhenAll(t1, t2);
    DisplayMessage("Done");
}
```

```
async Task ProcessFeedAsync(string url) {
    var text = await DownloadFeedAsync(url);
    var doc = ParseFeedIntoDoc(text);
    await SaveDocAsync(doc);
    ProcessLog.WriteEntry(url);
}
```

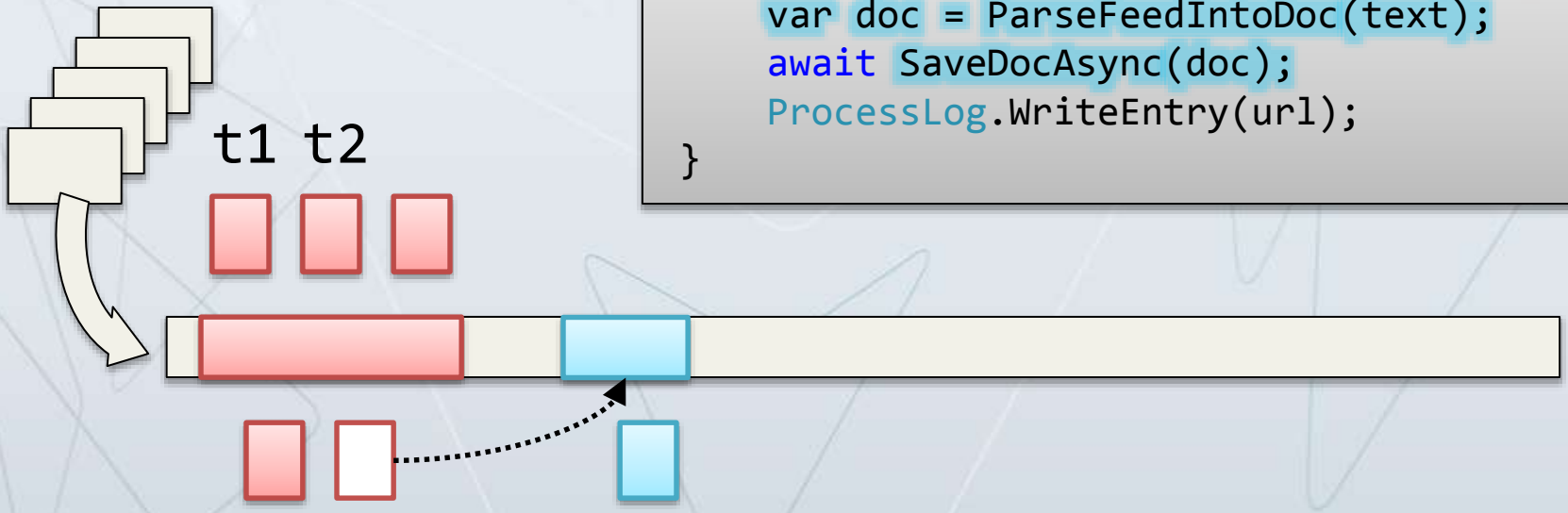


CodeValue

Asynchronous Control Flow

```
async void DoWorkAsync() {
    var t1 = ProcessFeedAsync("www.acme.com/rss");
    var t2 = ProcessFeedAsync("www.xyznews.com/rss");
    await Task.WhenAll(t1, t2);
    DisplayMessage("Done");
}
```

```
async Task ProcessFeedAsync(string url) {
    var text = await DownloadFeedAsync(url);
    var doc = ParseFeedIntoDoc(text);
    await SaveDocAsync(doc);
    ProcessLog.WriteEntry(url);
}
```

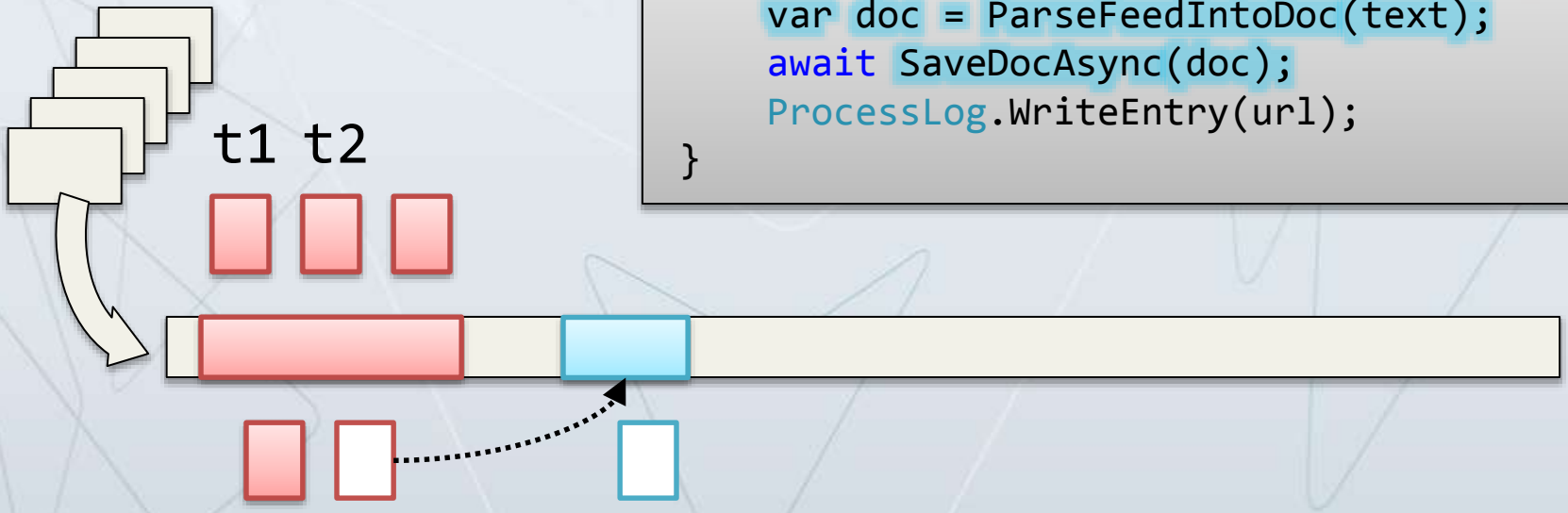


CodeValue

Asynchronous Control Flow

```
async void DoWorkAsync() {
    var t1 = ProcessFeedAsync("www.acme.com/rss");
    var t2 = ProcessFeedAsync("www.xyznews.com/rss");
    await Task.WhenAll(t1, t2);
    DisplayMessage("Done");
}
```

```
async Task ProcessFeedAsync(string url) {
    var text = await DownloadFeedAsync(url);
    var doc = ParseFeedIntoDoc(text);
    await SaveDocAsync(doc);
    ProcessLog.WriteEntry(url);
}
```

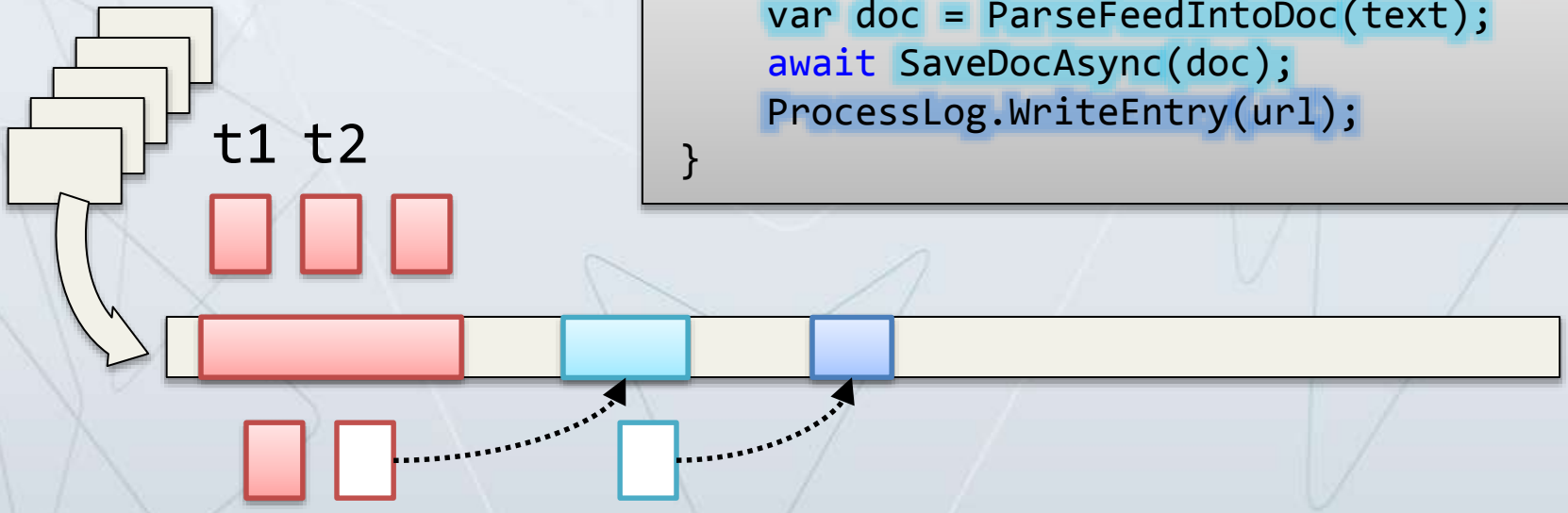


CodeValue

Asynchronous Control Flow

```
async void DoWorkAsync() {
    var t1 = ProcessFeedAsync("www.acme.com/rss");
    var t2 = ProcessFeedAsync("www.xyznews.com/rss");
    await Task.WhenAll(t1, t2);
    DisplayMessage("Done");
}
```

```
async Task ProcessFeedAsync(string url) {
    var text = await DownloadFeedAsync(url);
    var doc = ParseFeedIntoDoc(text);
    await SaveDocAsync(doc);
    ProcessLog.WriteEntry(url);
}
```

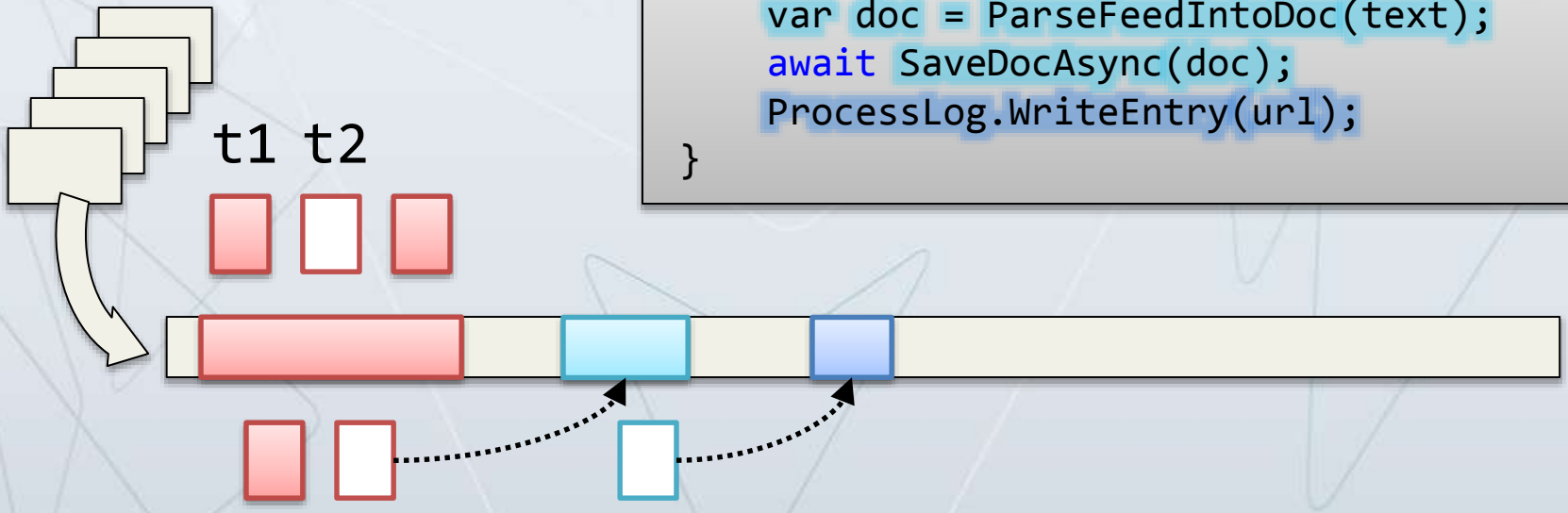


CodeValue

Asynchronous Control Flow

```
async void DoWorkAsync() {
    var t1 = ProcessFeedAsync("www.acme.com/rss");
    var t2 = ProcessFeedAsync("www.xyznews.com/rss");
    await Task.WhenAll(t1, t2);
    DisplayMessage("Done");
}
```

```
async Task ProcessFeedAsync(string url) {
    var text = await DownloadFeedAsync(url);
    var doc = ParseFeedIntoDoc(text);
    await SaveDocAsync(doc);
    ProcessLog.WriteEntry(url);
}
```

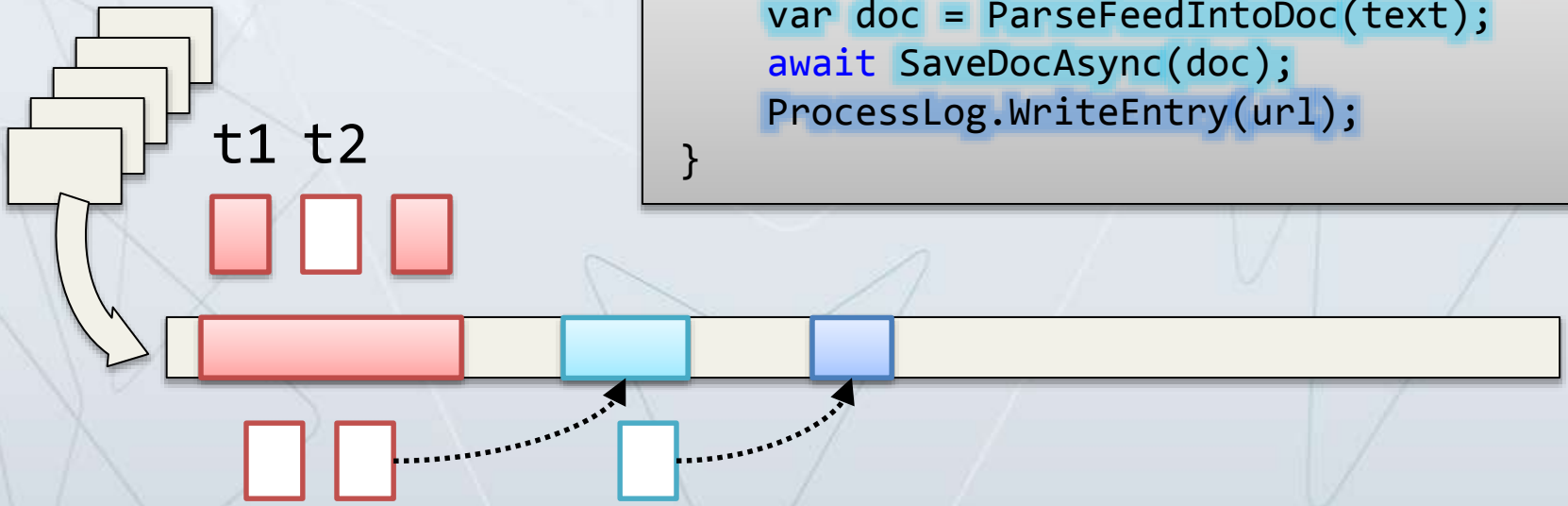


CodeValue

Asynchronous Control Flow

```
async void DoWorkAsync() {
    var t1 = ProcessFeedAsync("www.acme.com/rss");
    var t2 = ProcessFeedAsync("www.xyznews.com/rss");
    await Task.WhenAll(t1, t2);
    DisplayMessage("Done");
}
```

```
async Task ProcessFeedAsync(string url) {
    var text = await DownloadFeedAsync(url);
    var doc = ParseFeedIntoDoc(text);
    await SaveDocAsync(doc);
    ProcessLog.WriteEntry(url);
}
```

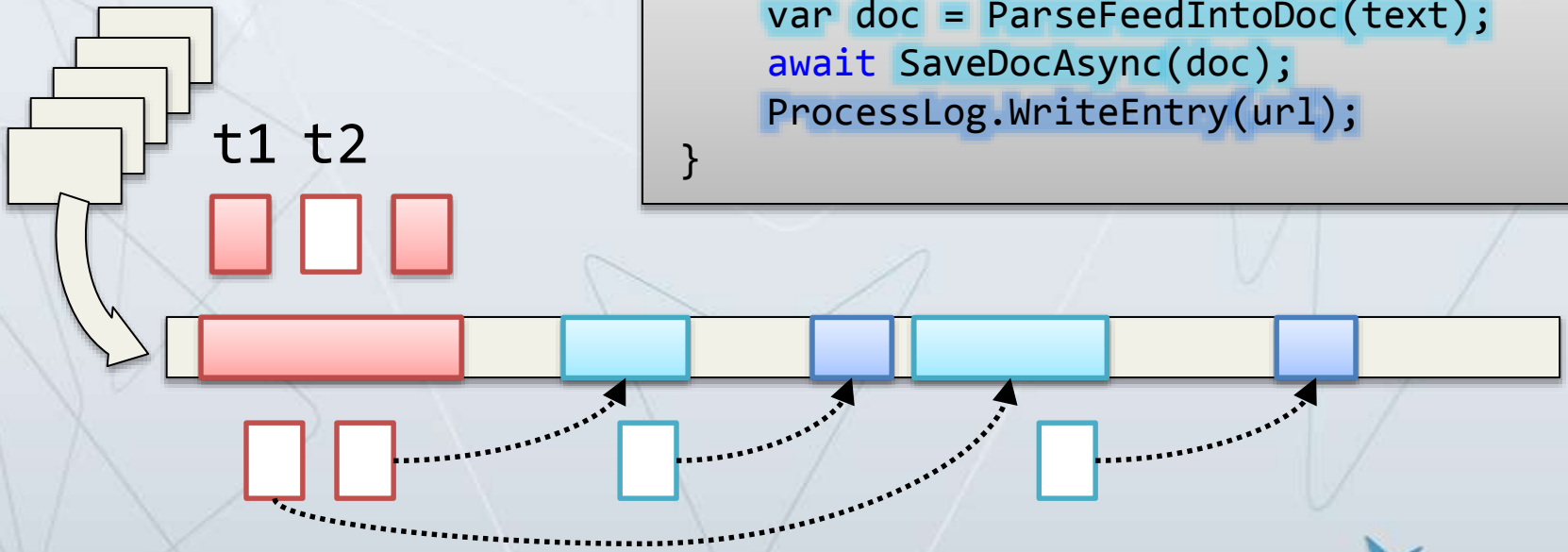


CodeValue

Asynchronous Control Flow

```
async void DoWorkAsync() {
    var t1 = ProcessFeedAsync("www.acme.com/rss");
    var t2 = ProcessFeedAsync("www.xyznews.com/rss");
    await Task.WhenAll(t1, t2);
    DisplayMessage("Done");
}
```

```
async Task ProcessFeedAsync(string url) {
    var text = await DownloadFeedAsync(url);
    var doc = ParseFeedIntoDoc(text);
    await SaveDocAsync(doc);
    ProcessLog.WriteEntry(url);
}
```

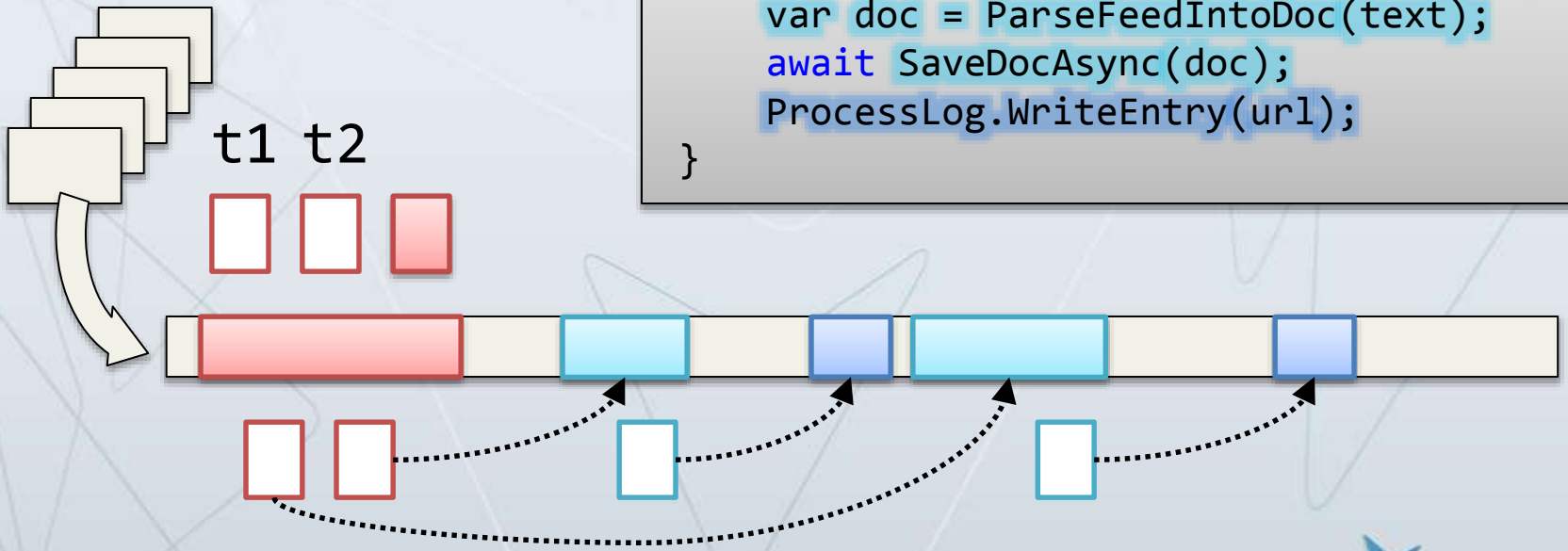


CodeValue

Asynchronous Control Flow

```
async void DoWorkAsync() {
    var t1 = ProcessFeedAsync("www.acme.com/rss");
    var t2 = ProcessFeedAsync("www.xyznews.com/rss");
    await Task.WhenAll(t1, t2);
    DisplayMessage("Done");
}
```

```
async Task ProcessFeedAsync(string url) {
    var text = await DownloadFeedAsync(url);
    var doc = ParseFeedIntoDoc(text);
    await SaveDocAsync(doc);
    ProcessLog.WriteEntry(url);
}
```

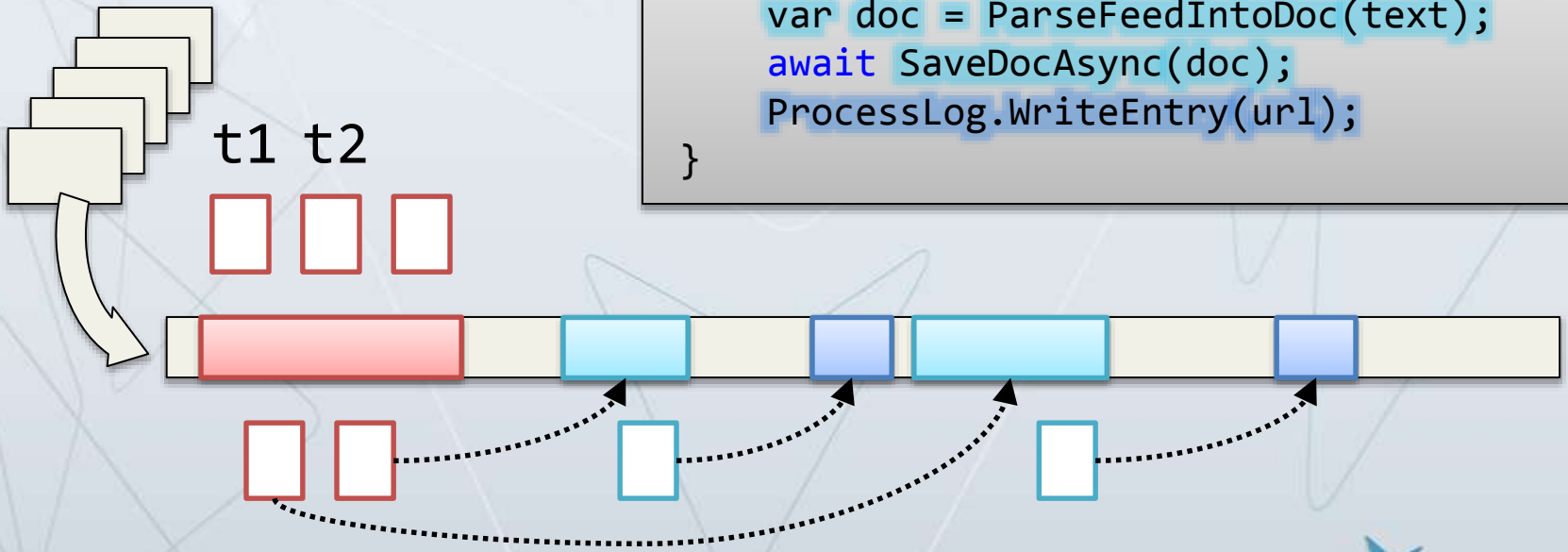


CodeValue

Asynchronous Control Flow

```
async void DoWorkAsync() {
    var t1 = ProcessFeedAsync("www.acme.com/rss");
    var t2 = ProcessFeedAsync("www.xyznews.com/rss");
    await Task.WhenAll(t1, t2);
    DisplayMessage("Done");
}
```

```
async Task ProcessFeedAsync(string url) {
    var text = await DownloadFeedAsync(url);
    var doc = ParseFeedIntoDoc(text);
    await SaveDocAsync(doc);
    ProcessLog.WriteEntry(url);
}
```

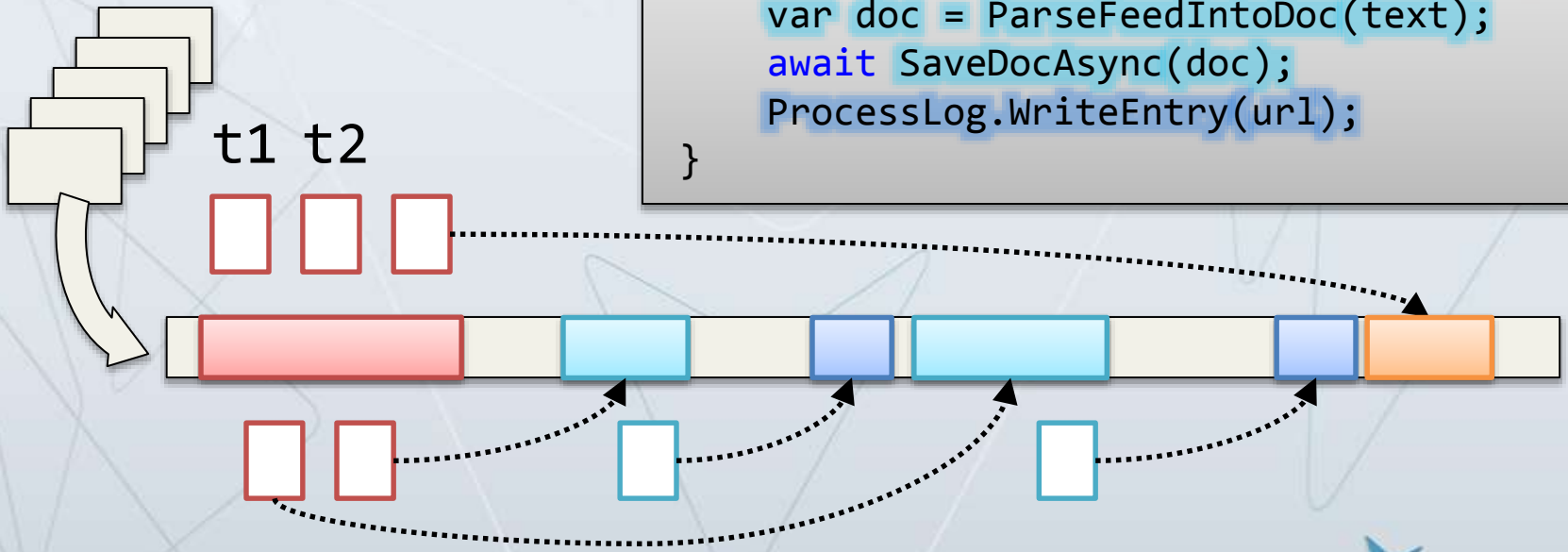


CodeValue

Asynchronous Control Flow

```
async void DoWorkAsync() {
    var t1 = ProcessFeedAsync("www.acme.com/rss");
    var t2 = ProcessFeedAsync("www.xyznews.com/rss");
    await Task.WhenAll(t1, t2);
    DisplayMessage("Done");
}
```

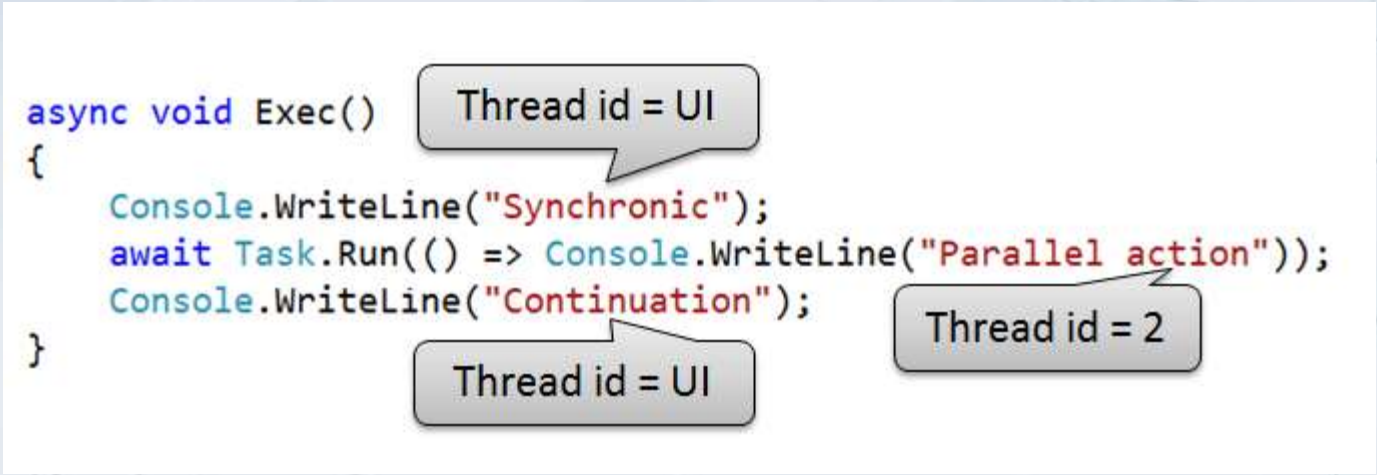
```
async Task ProcessFeedAsync(string url) {
    var text = await DownloadFeedAsync(url);
    var doc = ParseFeedIntoDoc(text);
    await SaveDocAsync(doc);
    ProcessLog.WriteEntry(url);
}
```



CodeValue

What about Threads?

- **Why do we care?**



- **Either we marshal back… or not.**

CodeValue

Await != Task.Run

```
async Task AsyncDelay() {
      await Task.Delay(1000);
      await Task.Delay(1000);
      Console.WriteLine("Done!");
}
```

```
async Task AsyncDelay() {
     for (int i = 0; i < 10; i++)
          await Task.Delay(1000);
     Console.WriteLine("Done!");
}
```

CodeValue

Await != Task.Run

```
async Task AsyncDelay() {
      await Task.Delay(1000);
      await Task.Delay(1000);
      Console.WriteLine(“Done!”);
}
```

```
Task AsyncDelay() {
 return Task.StartNew(()=> Thread.Sleep(1000))
   .ContinueWith(()=> Thread.Sleep(1000))
   .ContinueWith(()=> Console.WriteLine(“Done!”);
}
```

CodeValue

Await != Task.Run

```
async Task AsyncDelay() {
      Task t1 = Task.Delay(1000);
      Task t2 = Task.Delay(1000);
      await Task.WhenAll(t1, t2);
            // or Task.WhenAny
      Console.WriteLine(“Done!”);
}
```

CodeValue

What About Exceptions?

- **Should be as simple as the synchronous case**

```
var wc = new WebClient();
try {
   ValidateUrl(url);
   string txt = await wc.DownloadStringTaskAsync(url);
   string parsedText = DoRiskyParsing(txt);
   dataTextBox.Text = parsedText;
}
catch(ArgumentException ex) {
   // validation phase
}
catch(WebException ex) {
   // communication phase
}
catch(ParseException ex) {
   // continuation phase
}
```

What Happened to AggregateException?

- TPL allows us to handle multiple exceptions from child tasks.
- Async/await unwraps the first
- Workaround: rethrow manually:

```
await Task.WhenAll(t1, t2)
    .ContinueWith(t =>
        { if (t.Exception != null)
                throw t.Exception;
        });
```

CodeValue

Cancellation and Progress

- **Asynchronous methods may support cancellation**
  - With the `CancellationToken` type
- **Can also report progress**
  - An `IProgress<T>` interface is defined
  - And one stock implementation, `Progress<T>`
- **Can use both**

Cancellation & Progress

CodeValue

`await` Limitations

- **Cannot use await inside a catch/finally block**
- **Cannot use await inside a lock block**

- **Async/await methods are *composed*, not *atomic***

CodeValue

Custom Awaiters

- **The `await` keyword looks for a pattern**
- **Can await anything that**
  - Has a `GetAwaiter` method returning an object that has
    - `IsCompleted` property
    - `OnCompleted` method
    - `GetResult` method
    - Must implement the `INotifyCompletion` interface

CodeValue

Custom Awaiter

CodeValue

Points to Remember

- For any async block it is important to have at least one await.
- Coding convention: add Async to method name
- Can return void for root.
- Everything is managed by a State Machine Workflow by the compiler

Takeaways

- **Async/await are more than syntactic sugar**
  - Different paradigm
  - Methods are composed, not atomic
- **Let's not talk about threads**
- **Not about performance**
  - Though that doesn't hurt either

CodeValue

Questions?

- Asynchronous Programming with Async and Await: **http://msdn.microsoft.com/en-us/library/vstudio/hh191443.aspx**
- Parallel Processing Team Blog: **http://blogs.msdn.com/b/pfxteam**
- Await anything: **http://blogs.msdn.com/b/pfxteam/archive/2011/01/13/10115642.aspx**
- Async/Await Benchmarks: **http://blogs.msdn.com/b/pfxteam/archive/2011/11/10/10235962.aspx**

CodeValue