Microsoft

# Correcting Common Async/Await Mistakes in .NET

Brandon Minnick
Developer Advocate

@TheCodeTraveler

https://www.codetraveler.io/NDCOslo-AsyncAwait/

## Multi Threading

# Thread Pool

## Background Threads Managed by .NET

- Managed by Synchronization Context
- Maximum # of Threads
  - Depends on size of virtual memory
  - When all threads are busy, tasks are queued

# ReadDataFromUrl

```
async Task ReadDataFromUrl(string url)
{
    WebClient wc = new WebClient();
    byte[] result = await wc.DownloadDataTaskAsync(url);
    string data = Encoding.ASCII.GetString(result);
    LoadData(data);
}
```

## ReadDataFromUrl

```
async Task ReadDataFromUrl(string url)
{
    WebClient wc = new WebClient();
    byte[] result = await wc.DownloadDataTaskAsync(url);
    string data = Encoding.ASCII.GetString(result);
    LoadData(data);
}
```

Thread 1

## ReadDataFromUrl

```
async Task ReadDataFromUrl(string url)
{
    WebClient wc = new WebClient();
    byte[] result = await wc.DownloadDataTaskAsync(url);
    string data = Encoding.ASCII.GetString(result);
    LoadData(data);
}
```

Thread 2*

*Can be any thread other than Thread 1, e.g. Thread 32

## ReadDataFromUrl

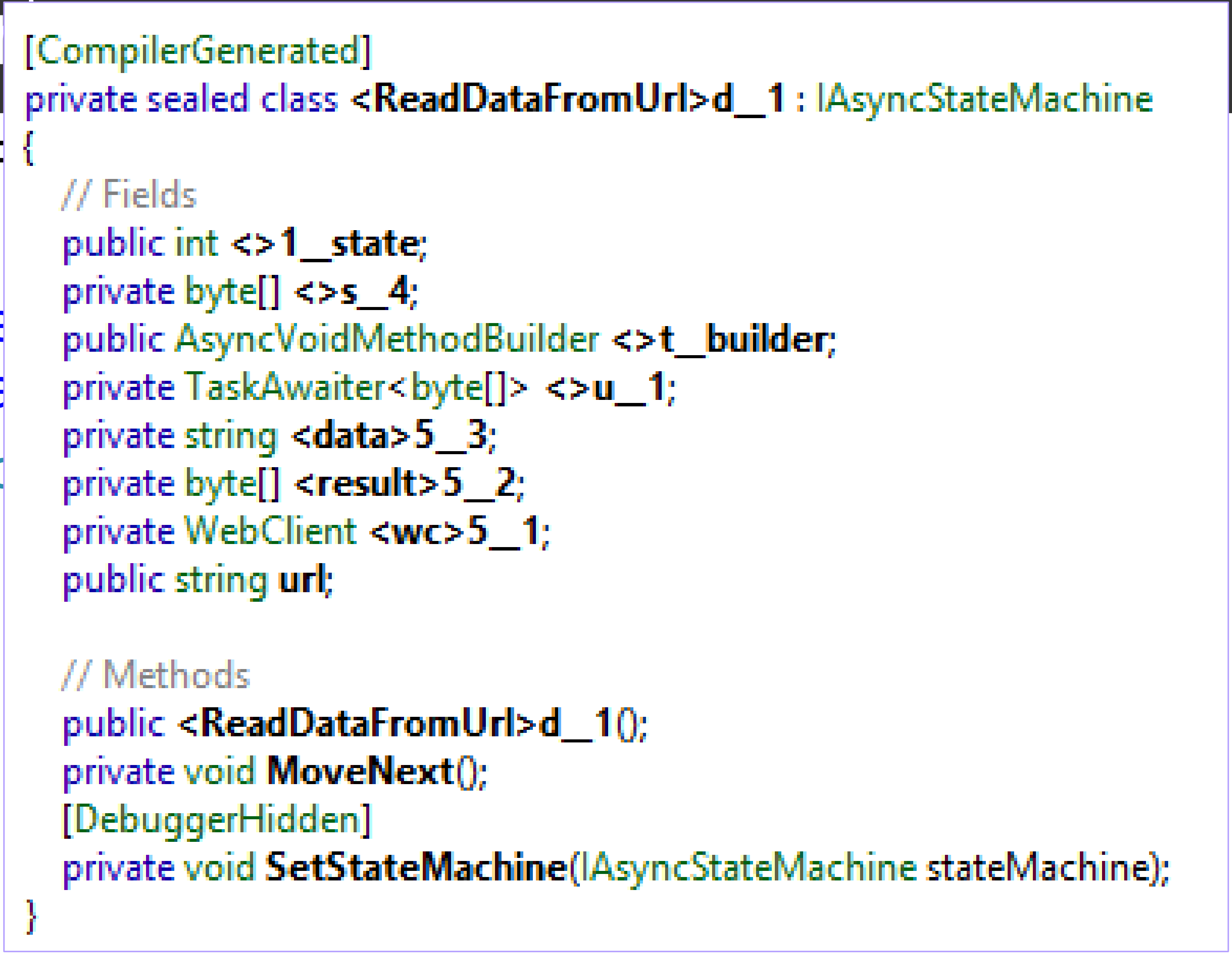
```
async Task ReadDataFromUrl(string url)
{
    WebClient wc = new WebClient();
    byte[] result = await wc.DownloadDataTaskAsync(url);
    string data = Encoding.ASCII.GetString(result);
    LoadData(data);
}
```

Thread 1

## Compiled Code

# ReadDataFromUrl

```
async Task ReadDataF
{
    WebClient wc = ne
    byte[] result = a
    string data = Enc
    LoadData(data);
}
```



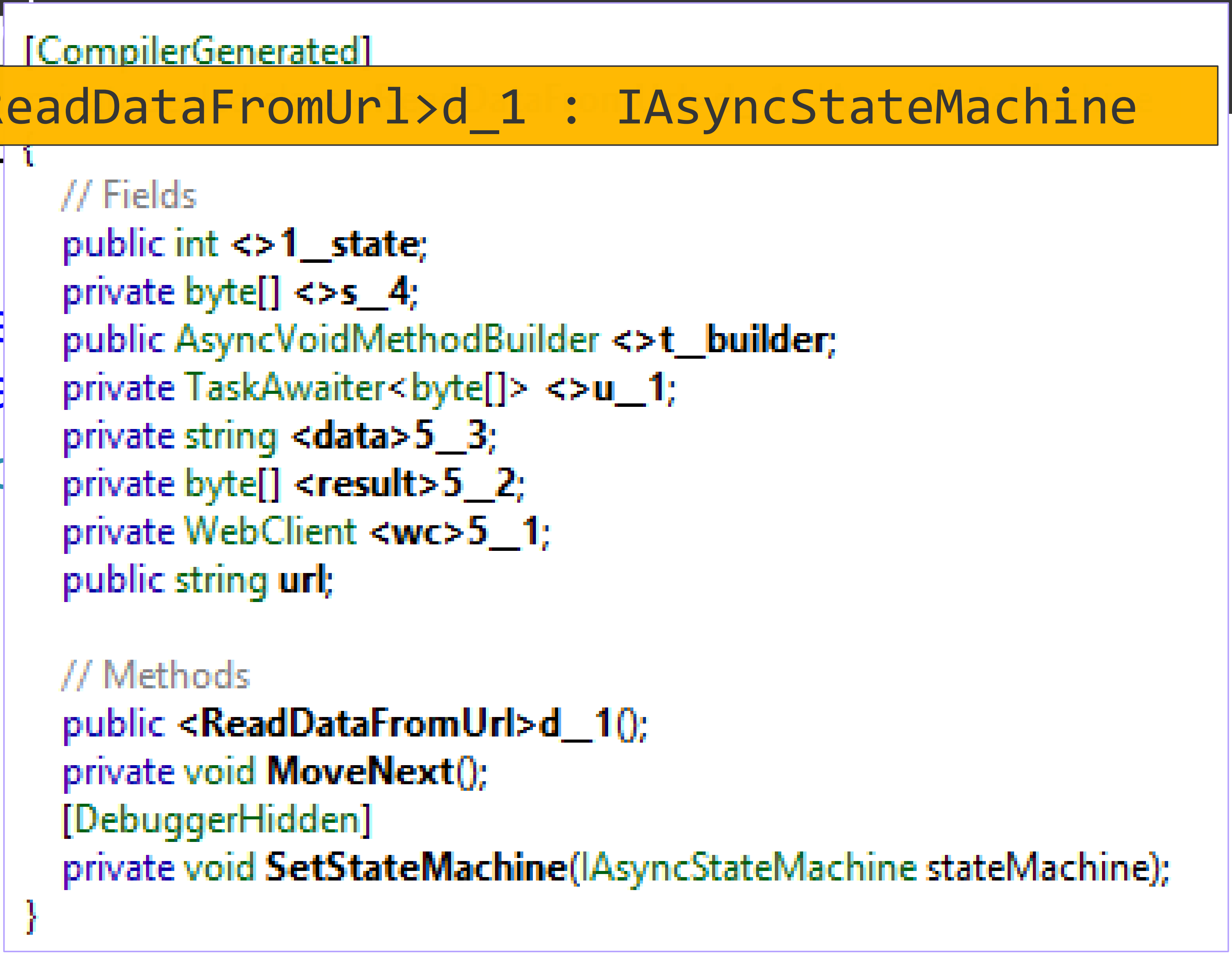
```
[CompilerGenerated]
private sealed class <ReadDataFromUrl>d__1 : IAsyncStateMachine
{
    // Fields
    public int <>1__state;
    private byte[] <>s__4;
    public AsyncVoidMethodBuilder <>t__builder;
    private TaskAwaiter<byte[]> <>u__1;
    private string <data>5__3;
    private byte[] <result>5__2;
    private WebClient <wc>5__1;
    public string url;

    // Methods
    public <ReadDataFromUrl>d__1();
    private void MoveNext();
    [DebuggerHidden]
    private void SetStateMachine(IAsyncStateMachine stateMachine);
}
```

# ReadDataFromUr

```
private sealed class <ReadDataFromUrl>d_1 : IAsyncStateMachine
```

```
async Task ReadDataF
{
    WebClient wc = ne
    byte[] result = a
    string data = Enc
    LoadData(data);
}
```

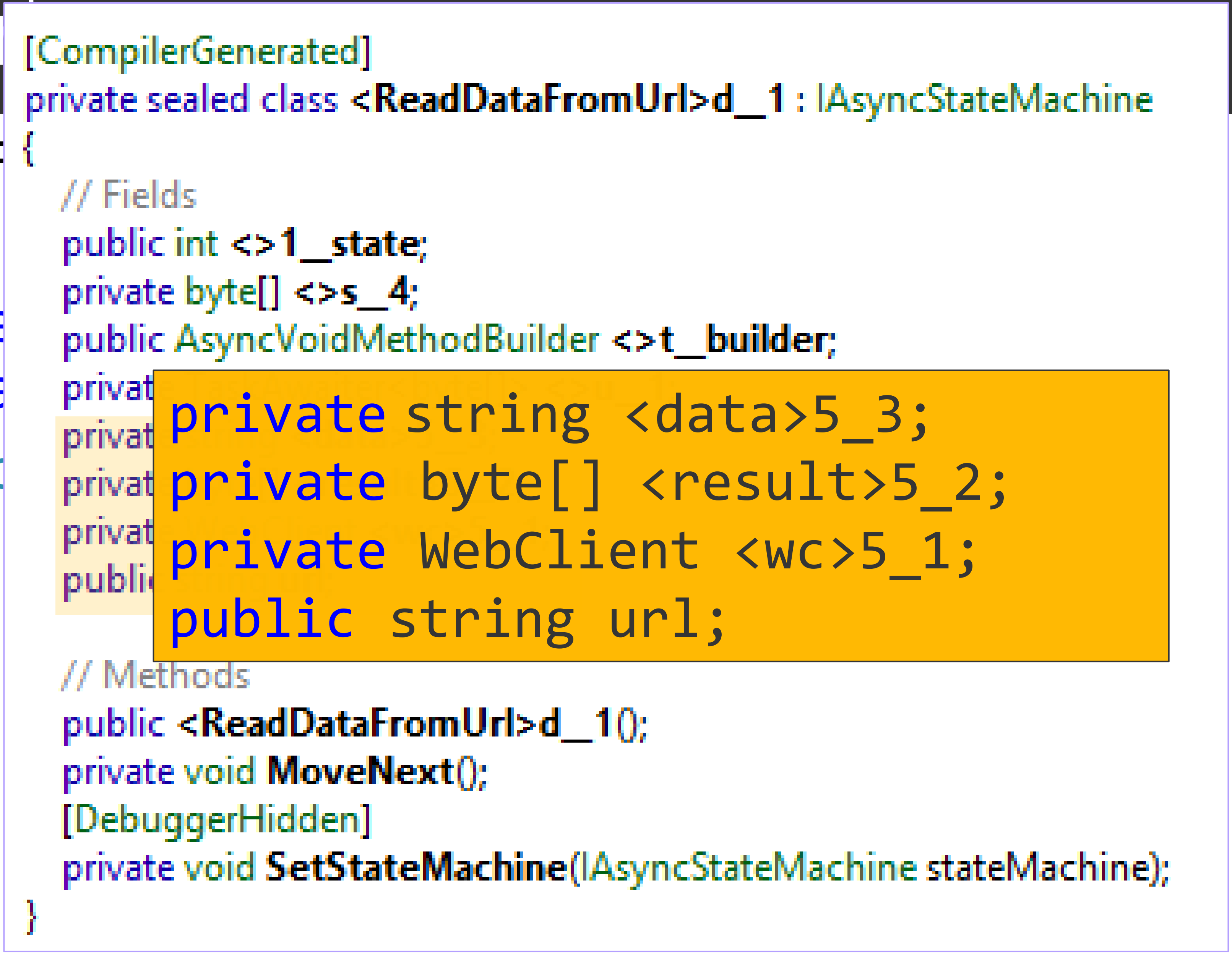


```
[CompilerGenerated]
{
    // Fields
    public int <>1__state;
    private byte[] <>s__4;
    public AsyncVoidMethodBuilder <>t__builder;
    private TaskAwaiter<byte[]> <>u__1;
    private string <data>5__3;
    private byte[] <result>5__2;
    private WebClient <wc>5__1;
    public string url;

    // Methods
    public <ReadDataFromUrl>d__1();
    private void MoveNext();
    [DebuggerHidden]
    private void SetStateMachine(IAsyncStateMachine stateMachine);
}
```

# ReadDataFromUr

```
async Task ReadDataF
{
    WebClient wc = ne
    byte[] result = a
    string data = Enc
    LoadData(data);
}
```



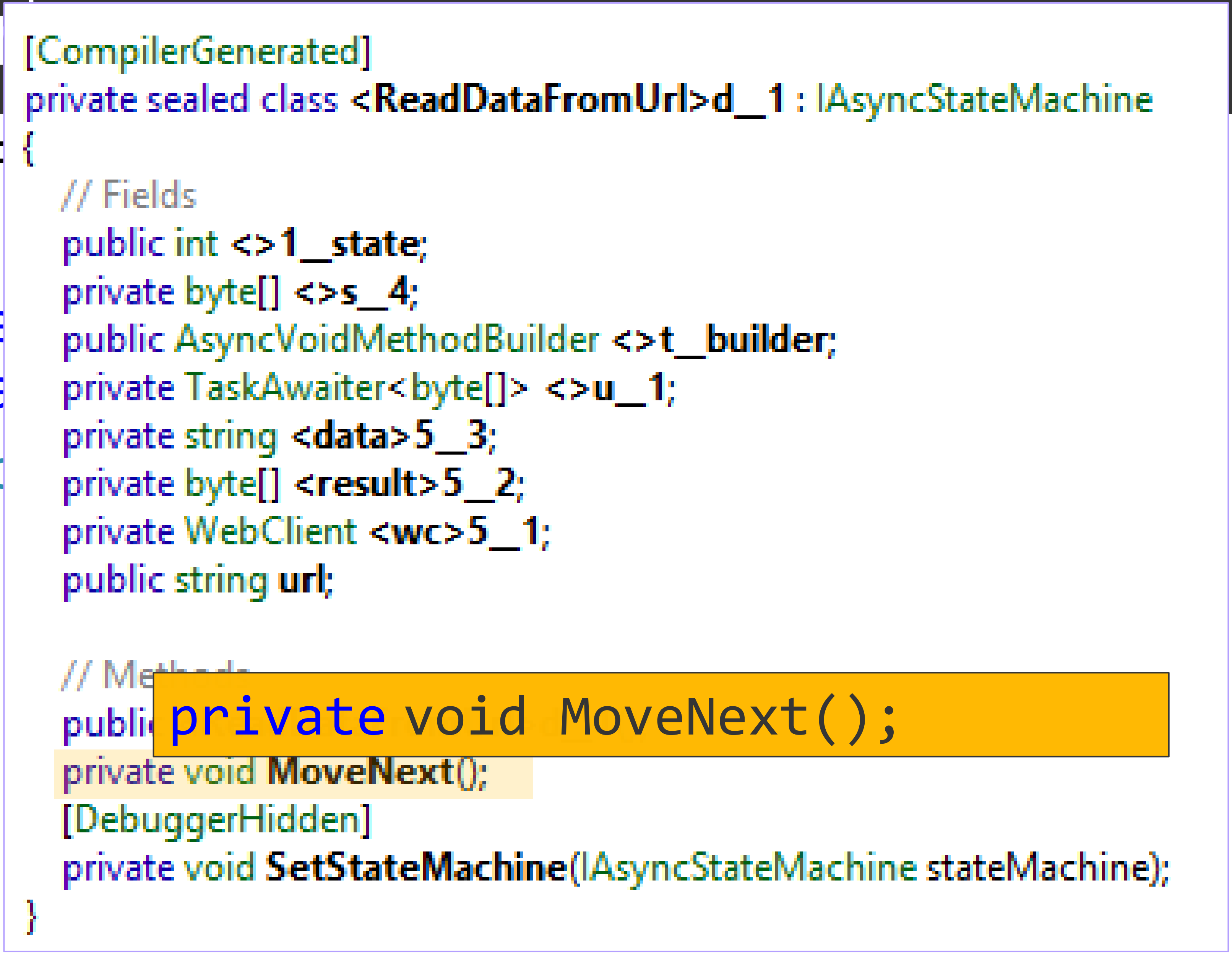
```
[CompilerGenerated]
private sealed class <ReadDataFromUrl>d__1 : IAsyncStateMachine
{
    // Fields
    public int <>1__state;
    private byte[] <>s__4;
    public AsyncVoidMethodBuilder <>t__builder;
    privat
    privat
    privat
    privat
    publi

    // Methods
    public <ReadDataFromUrl>d__1();
    private void MoveNext();
    [DebuggerHidden]
    private void SetStateMachine(IAsyncStateMachine stateMachine);
}
```

```
private string <data>5_3;
private byte[] <result>5_2;
private WebClient <wc>5_1;
public string url;
```

# ReadDataFromUr

```
async Task ReadDataF
{
    WebClient wc = ne
    byte[] result = a
    string data = Enc
    LoadData(data);
}
```



```
[CompilerGenerated]
private sealed class <ReadDataFromUrl>d__1 : IAsyncStateMachine
{
    // Fields
    public int <>1__state;
    private byte[] <>s__4;
    public AsyncVoidMethodBuilder <>t__builder;
    private TaskAwaiter<byte[]> <>u__1;
    private string <data>5__3;
    private byte[] <result>5__2;
    private WebClient <wc>5__1;
    public string url;

    // Me
    publi
    private void MoveNext();
    [DebuggerHidden]
    private void SetStateMachine(IAsyncStateMachine stateMachine);
}
```

```
private void MoveNext();
```

@TheCodeTraveler

https://www.codetraveler.io/NDCOslo-AsyncAwait/

# ReadDataFromUrl_MoveNext

```
public void MoveNext()
{
   uint num = (uint)this.$PC;
   this.$PC = -1;
   try {
      switch (num) {
         case 0:
            this.<wc>__0 = new WebClient();
            this.$awaiter0 = this.<wc>__0.DownloadDataTaskAsync(this.url).GetAwaiter();
            this.$PC = 1;
            ...
            return;
            break;
         case 1:
            this.<result>__1 = this.$awaiter0.GetResult();
            this.<data>__2 = Encoding.ASCII.GetString(this.<result>__1);
            this.$this.LoadData(this.<data>__2);
            break;
         default:
            return;
      }
   }
   catch (Exception exception) { ... }
   this.$PC = -1;
   this.$builder.SetResult();
}
```

# ReadDataFromUrl_MoveNext

```
public void MoveNext()
{
   uint num = (uint)this.$PC;
   this.$PC = -1;
   try {
      switch (num) {
         case 0:
```

```
case 0:
    this.<wc>__0 = new WebClient();
    this.$awaiter0 = this.<wc>__0.DownloadDataTaskAsync(this.url).GetAwaiter();
    this.$PC = 1;
    ...
    return;
```

```
            break;
         default:
            return;
      }
   }
   catch (Exception exception) { ... }
   this.$PC = -1;
   this.$builder.SetResult();
}
```

# ReadDataFromUrl_MoveNext

```
public void MoveNext()
{
   uint num = (uint)this.$PC;
   this.$PC = -1;
   try {
      switch (num) {
         case 0:
```

```
case 1:
   this.<result>__1 = this.$awaiter0.GetResult();
   this.<data>__2 = Encoding.ASCII.GetString(this.<result>__1);
   this.$this.LoadData(this.<data>__2);
   break;
```

```
            break;
         default:
            return;
      }
   }
   catch (Exception exception) { ... }
   this.$PC = -1;
   this.$builder.SetResult();
}
```

## ReadDataFromUrl_MoveNext

```
public void MoveNext()
{
   uint num = (uint)this.$PC;
   this.$PC = -1;
   try {
try {
      switch (num) {
         case 0:
            this.<wc>__0 = new WebClient();
            this.$awaiter0 = this.<wc>__0.DownloadDataTaskAsync(this.url).GetAwaiter();
            this.$PC = 1;
            ...
            return;
            break;
         case 1:
            this.<result>__1 = this.$awaiter0.GetResult();
            this.<data>__2 = Encoding.ASCII.GetString(this.<result>__1);
            this.$this.LoadData(this.<data>__2);
            break;
         default:
            return;
catch (Exception exception) { . . . }
   catch (Exception exception) { ... }
   this.$PC = -1;
   this.$builder.SetResult();
}
```

## Quick Review

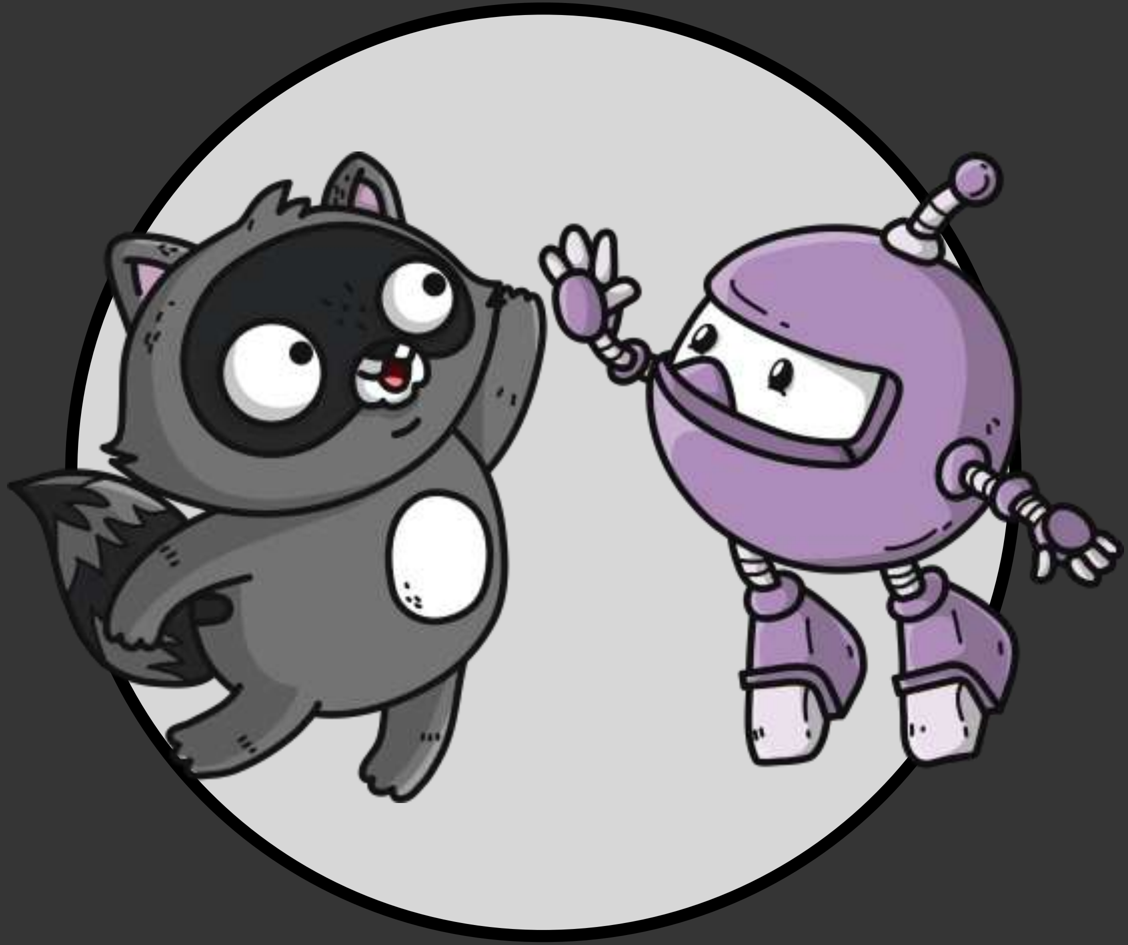
## Async Keyword Adds 100 Bytes

Every Async Method Becomes a Class

## Await Every Task

Non-awaited Tasks Hide Exceptions

## Let’s Fix Some Code



## Async/Await Best Practices

# Async/Await Best Practices

## Never Use `.Wait()` or `.Result`

- Always use `await`
- If synchronous, use `.GetAwaiter().GetResult()`

# Async/Await Best Practices

## Fire and Forget Tasks

- Use `SafeFireAndForget`
- NuGet: AsyncAwaitBestPractices

# Async/Await Best Practices

## Async Commands

- Use `IAsyncCommand`
- NuGet: AsyncAwaitBestPractices.MVVM

# Async/Await Best Practices

## Avoid `return await`

- Remove `async` keyword
    - Except: In `try/catch` blocks
    - Except: In `using( … )` blocks

## Async/Await Best Practices

### Utilize `.ConfigureAwait(false)`

- Except: When needing to return to calling thread

# Async/Await Best Practices

## Utilize `ValueTask`

- Anytime a method might not hit `await`
- NuGet: System.Threading.Tasks.Extensions

## Resources

https://www.codetraveler.io/NDCOslo-AsyncAwait/

## Thank You

https://www.codetraveler.io/NDCOslo-AsyncAwait/