Particular Software

Welcome

Rearchitect your code

towards async/await

Particular Software

Solution Architect
Enthusiastic Software Engineer
Microsoft MVP for systems integration

@danielmarbach
particular.net/blog
planetgeek.ch

Goals

target

Why async is the future

How to gradually move your code towards async / await

The toolbelt for an async ninja

Premise


Intro
Phases
WrapUp

The die

is cast

javascript

ES2015

```
async function chainAnimationsPromise(elem, animations)
{
    let ret = null;
    try {
    for(const anim of animations) {
        ret = await anim(elem);
    }
    catch(e) { /* ignore and keep going */ }
    return ret;
}
```

```
$ npm install babel-plugin-syntax-async-functions
$ npm install babel-plugin-transform-async-to-generator
```

httpclient

```
using (var client = new HttpClient()) {
  var response = await
      client.GetAsync("api/products/1");
  if (response.IsSuccessStatusCode)
  {
     var product = await
        response.Content.ReadAsAsync<Product>();
  }
}
```

Azure SDK

```
var queryable =
client.CreateDocumentQuery<Entity>(...)
    .AsDocumentQuery();

while (queryable.HasMoreResults)
{
  foreach(var e in await
queryable.ExecuteNextAsync<Entity>())
  {
    // Iterate through entities
  }
}
```

async

event-driven

Task

uniform

Task

IO-bound

Task

CPU-bound

Recap

best-practices

Use async Task instead of async void

Async all the way, don't mix blocking and asynchronous code

Async / await

is viral

but

It kicks your

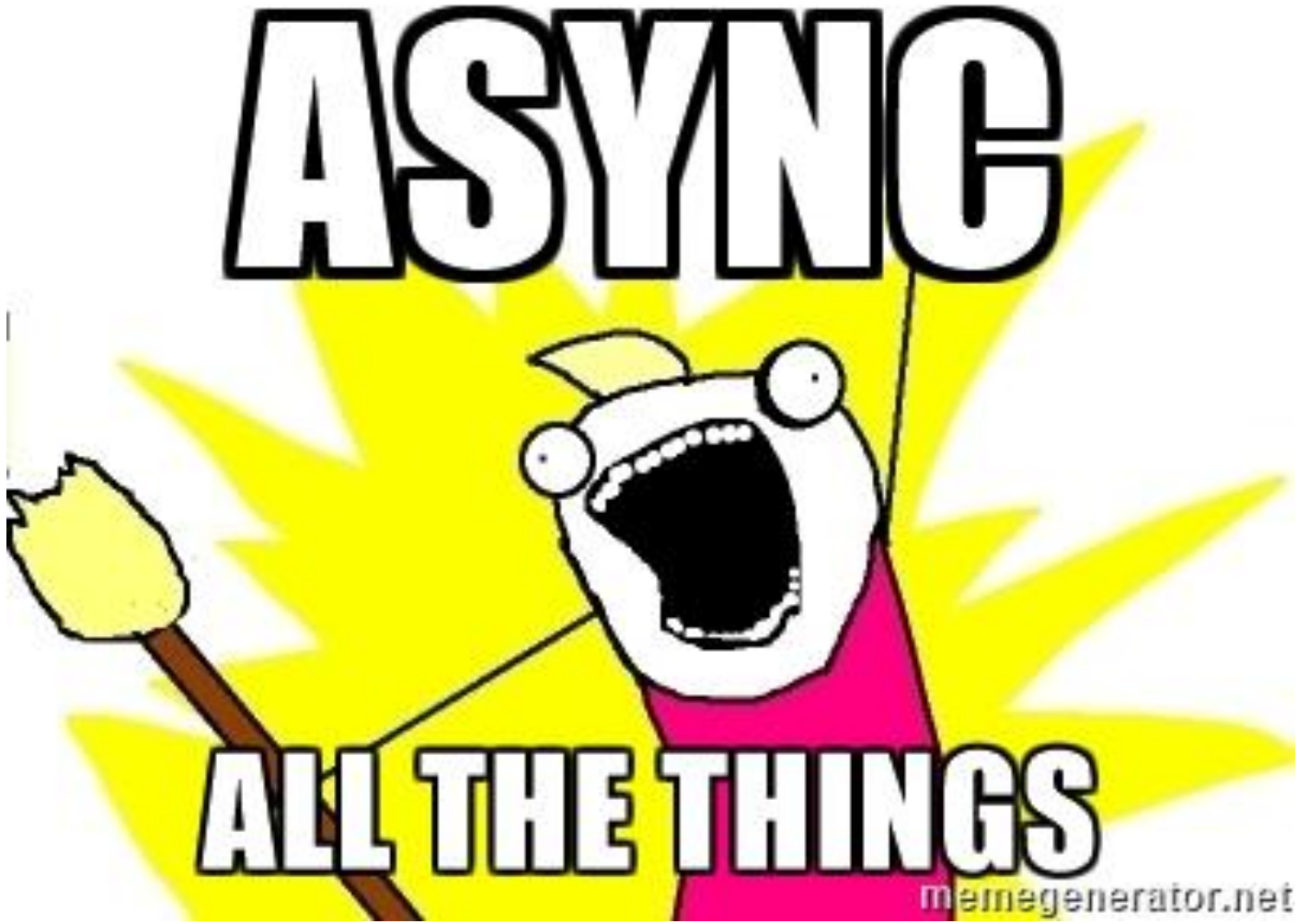
servers

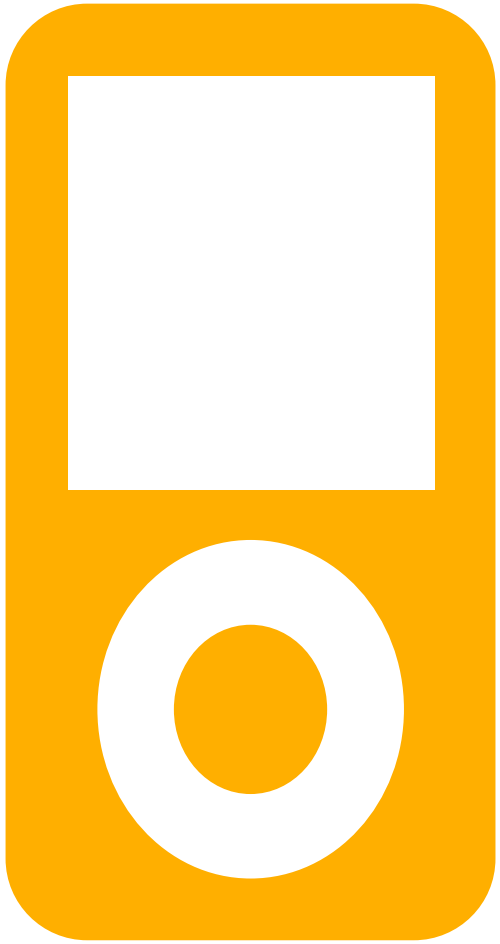
butt

NServiceBus

Azure Service Bus	26 times
Azure Storage Queues	6 times
MSMQ	3 times

more message throughput


ASYNC
ALL THE THINGS
memegenerator.net

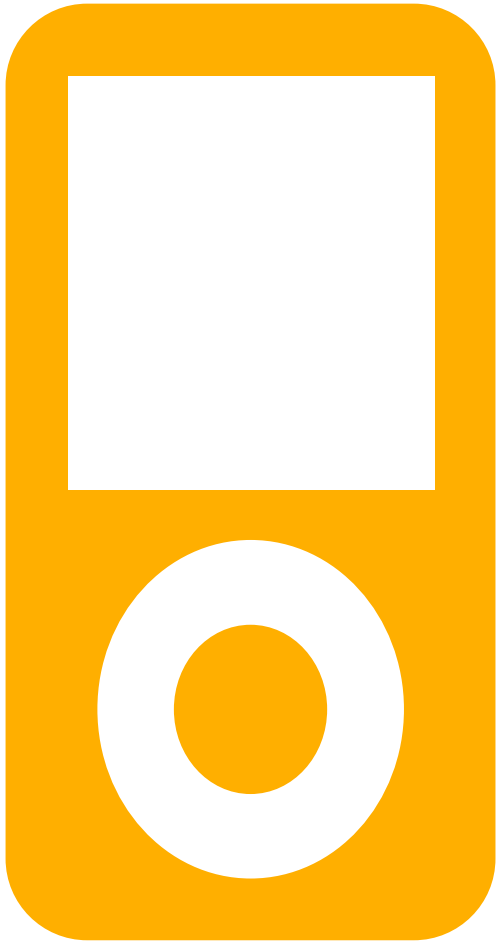


identify

ExPlore

Overcome

bring together



identify

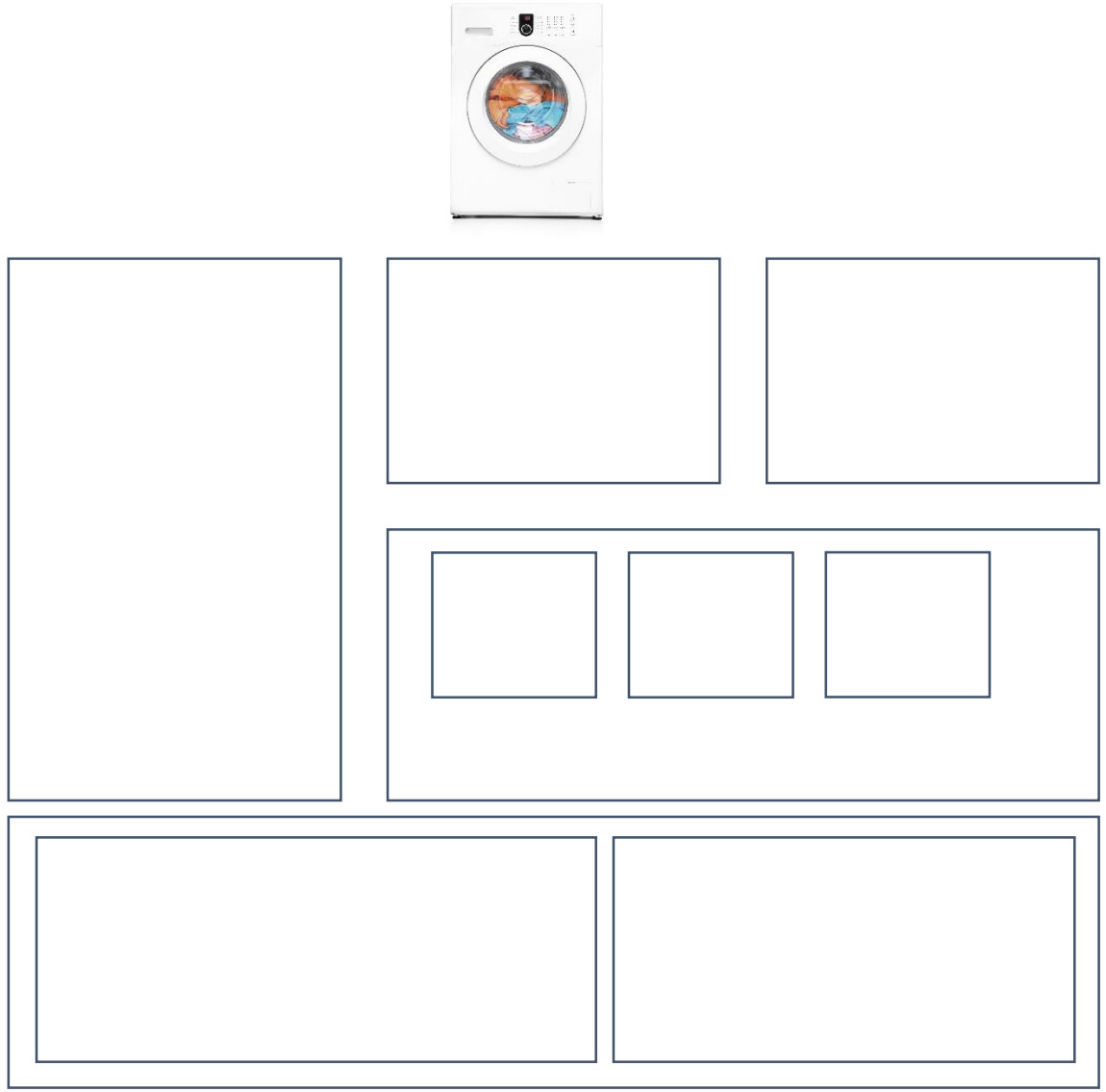
ExPlore

Overcome

bring together

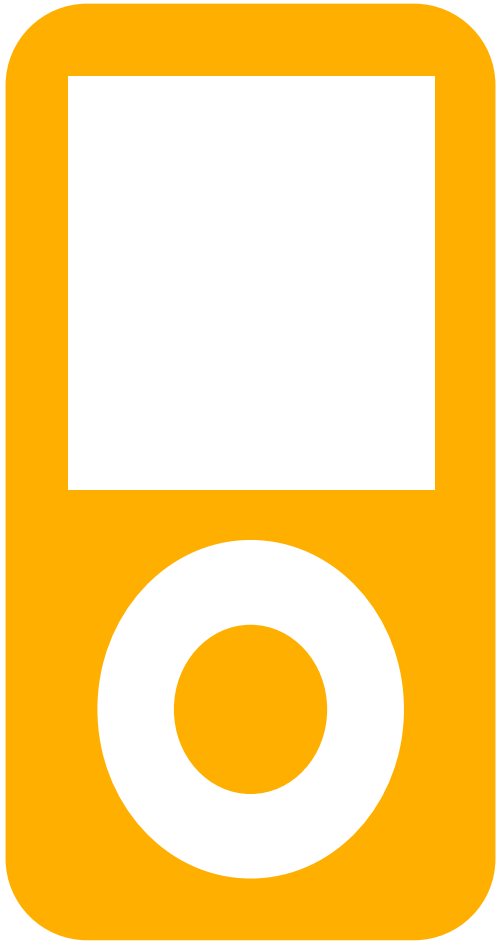
Identify

IO-bound



NServiceBus

IO-bound



identify

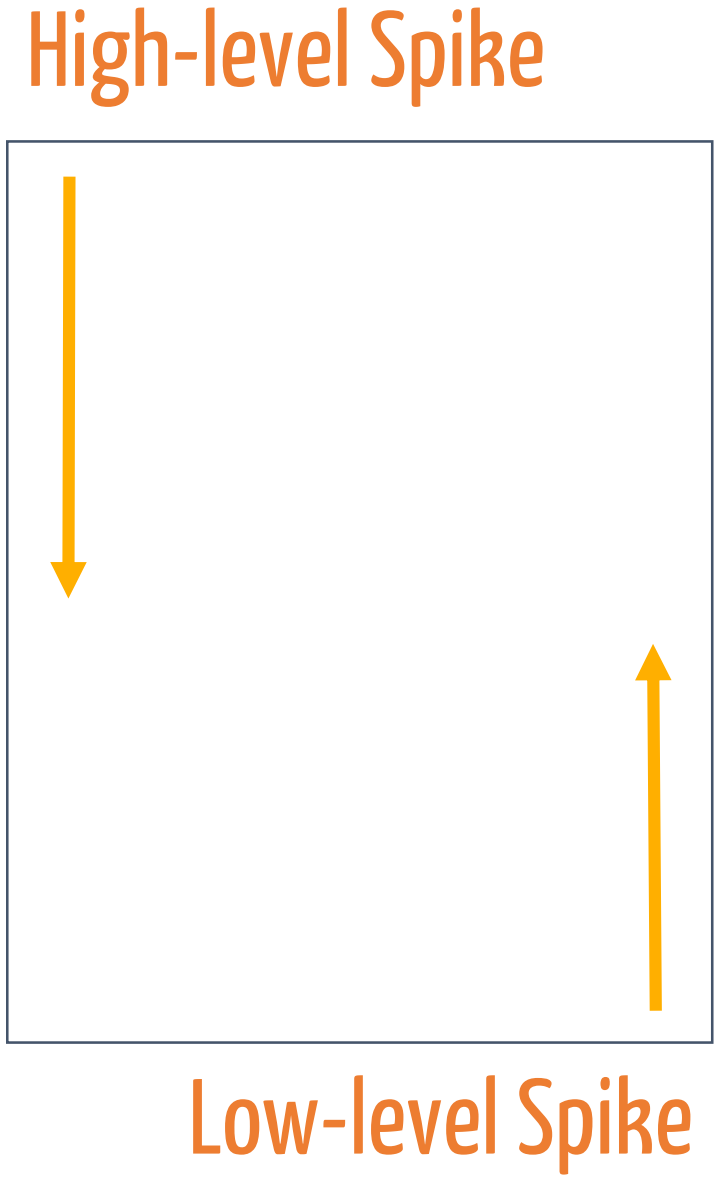
Ex**P**lore

Overcome

bring together

Explore

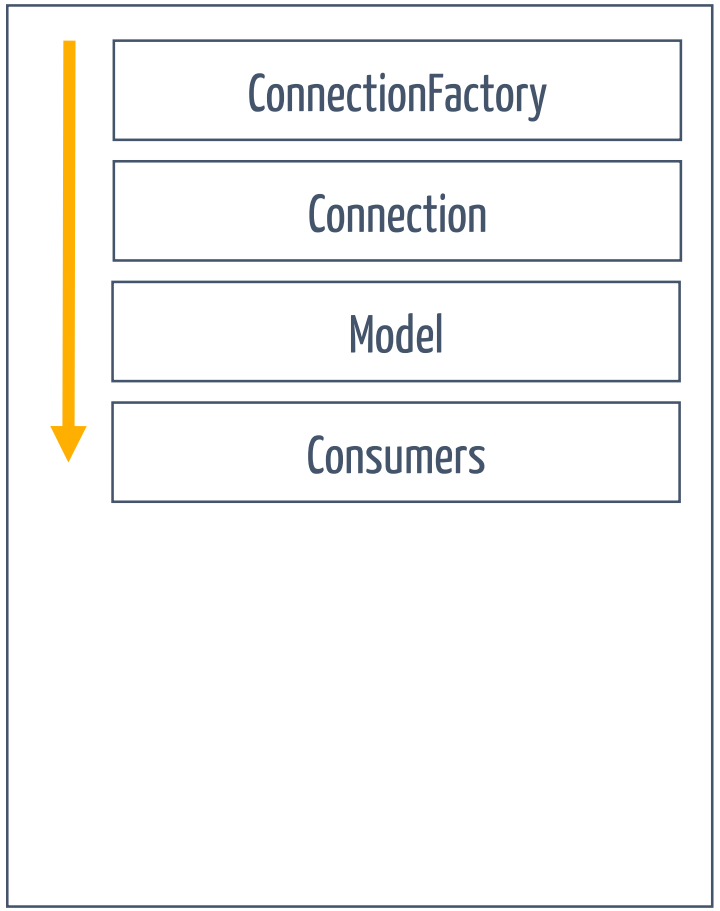
IO-bound



RabbitMQ Client

IO-bound

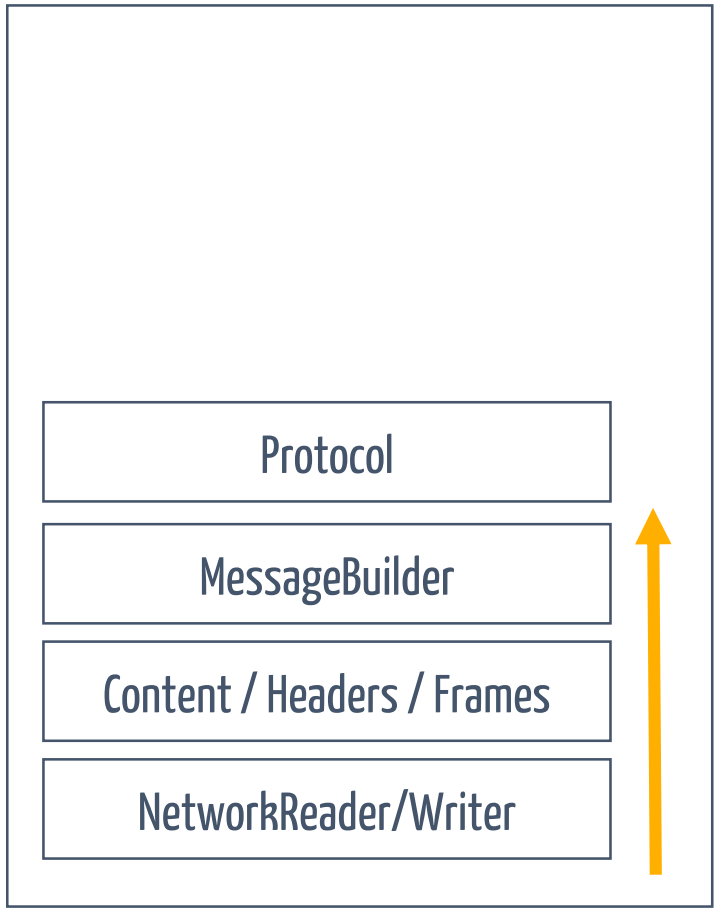
High-level Spike



github.com/rabbitmq/rabbitmq-dotnet-client/pull/151

RabbitMQ Client

IO-bound



Low-level Spike

github.com/rabbitmq/rabbitmq-dotnet-client/pull/149

Event handlers

Locks

Monitor

Semaphore / Mutex

Auto / ManualResetEvent

Ref/Out parameters

Thread

Ambient state

IO-bound calls in 3rd Party libs

Remote Procedure Calls

Event handlers


```
public delegate void EventHandler(object sender, EventArgs e);

public delegate void EventHandler<TEventArgs>(object sender, TEventArgs e);
```

```
async void MyEventHandler(object sender, EventArgs e)
{
    await Task.Yield();
    throw new InvalidOperationException();
}
```

ManualResetEvent


```
var syncEvent = new ManualResetEvent(false);

var t1 = Task.Run(() => {
   syncEvent.WaitOne();
});

var t2 = Task.Run(() => {
  Thread.Sleep(2000);
  syncEvent.Set();
});

await Task.WhenAll(t1, t2);
```

void stinks wait smells

Remember

Async all the way means avoid blocking code

locks


```
 var locker = new object();
lock (locker)
{
   await Task.Yield();
}
```

Error CS1996

Cannot await in the body of a lock statement

http://stackoverflow.com/questions/7612602/why-cant-i-use-the-await-operator-within-the-body-of-a-lock-statement

Ref/Out


```
static async Task Out(string content, out string parameter)
{
  var randomFileName = Path.GetTempFileName();
  using (var writer = new StreamWriter(randomFileName))
  {
    await writer.WriteLineAsync(content);
  }
  parameter = randomFileName;
}
```

Error CS1988

Async methods cannot have ref or out parameters

Ambient state

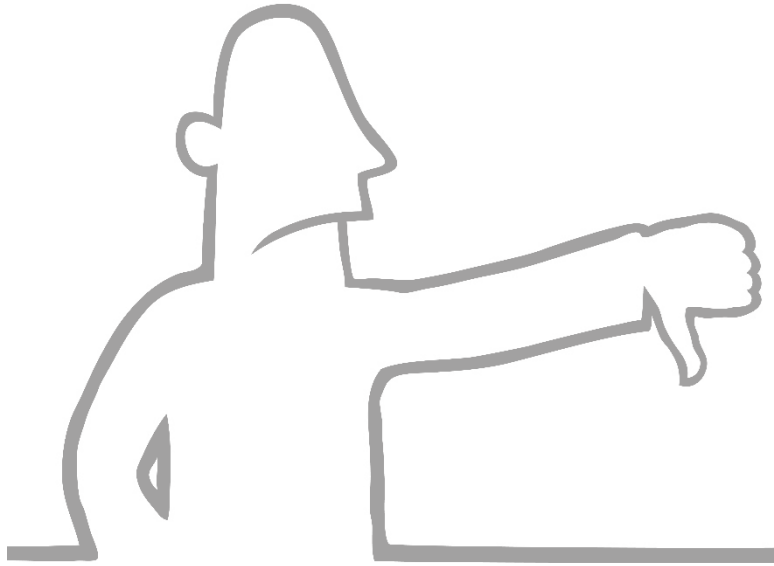

```
class ClassWithAmbientState
{
  static ThreadLocal<int> ambientState =
      new ThreadLocal<int>(() => 1);

  public void Do()
  {
    ambientState.Value++;
  }
}
```

Ambient state


```
var instance = new ClassWithAmbientState();
var tasks = new Task[3];
for (int i = 0; i < 3; i++) {
  tasks[i] = Task.Run(() => {
    instance.Do();
    Thread.Sleep(200);
    instance.Do();
  });
}

await Task.WhenAll(tasks);
```

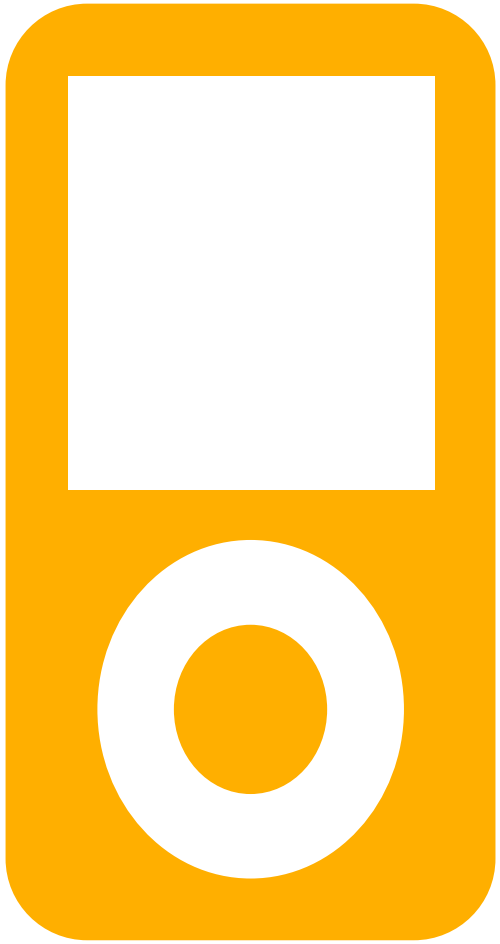


Older constructs bound to threads
fall apart in the async/await world

Remember

Forget thread!

think Task



identify

ExPlore

Overcome

bring together

Event handlers


```
public delegate Task AsyncEventHandler(object sender, EventArgs e);

async Task MyAsyncEventHandler(object sender, EventArgs e) { }

protected virtual Task OnMyAsyncEvent() {
  var invocations = handler.GetInvocationList();
  var handlerTasks = new Task[invocationList.Length];

  for (int i = 0; i < invocations.Length; i++) {
   handlerTasks[i] = ((AsyncEventHandler)invocations[i])(...);
  }
  return Task.WhenAll(handlerTasks);
}
```

ManualResetEvent


```
var tcs = new TaskCompletionSource<object>();

var t1 = ((Func<Task>)(async () => {
  await tcs.Task;
}));

var t2 = ((Func<Task>)(async () => {
  await Task.Delay(2000);
  tcs.TrySetResult(null);
}));

await Task.WhenAll(t1, t2);
```

ManualResetEvent

Works for set once events only.
For reset events an approach is available on my github account

locks

Can we change the code so that we don't have to await inside the lock?

locks


```
int sharedRessource = 0;
var semaphore = new SemaphoreSlim(1);

var tasks = new Task[3];
for (int i = 0; i < 3; i++) {
  tasks[i] = ((Func<Task>) (async () => {
    await semaphore.WaitAsync();
    sharedRessource++;
    semaphore.Release();
  }))();
}
await Task.WhenAll(tasks);
```

Ref/Out


```
static async Task<string> Out(string content)
{
  var randomFileName = Path.GetTempFileName();
  using (var writer = new StreamWriter(randomFileName))
  {
    await writer.WriteLineAsync(content);
  }
  return randomFileName;
}
```

Ambient state


```
class ClassWithAmbientState {
  static AsyncLocal<int> ambientState =
      new AsyncLocal<int>();

  static ClassWithAmbientState() {
    ambientState.Value = 1;
  }

  public void Do() {
    ambientState.Value++;
  }
}
```

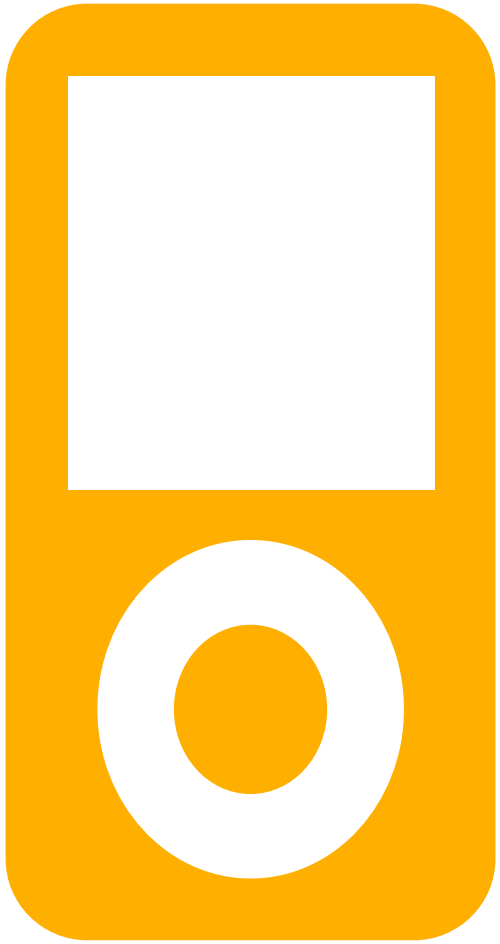
Ambient state

Even better:
Can we change the code so that we float state into methods that need it?

Ambient state


```
var instance = new ClassWithFloatingState();

var tasks = new Task[3];
for (int i = 0; i < 3; i++) {
  tasks[i] = ((Func<Task>)(async () => {
    int current = 1;
    current = instance.Do(current);
    await Task.Delay(200).ConfigureAwait(false);
    instance.Do(current);
  }))();
}
await Task.WhenAll(tasks);
```



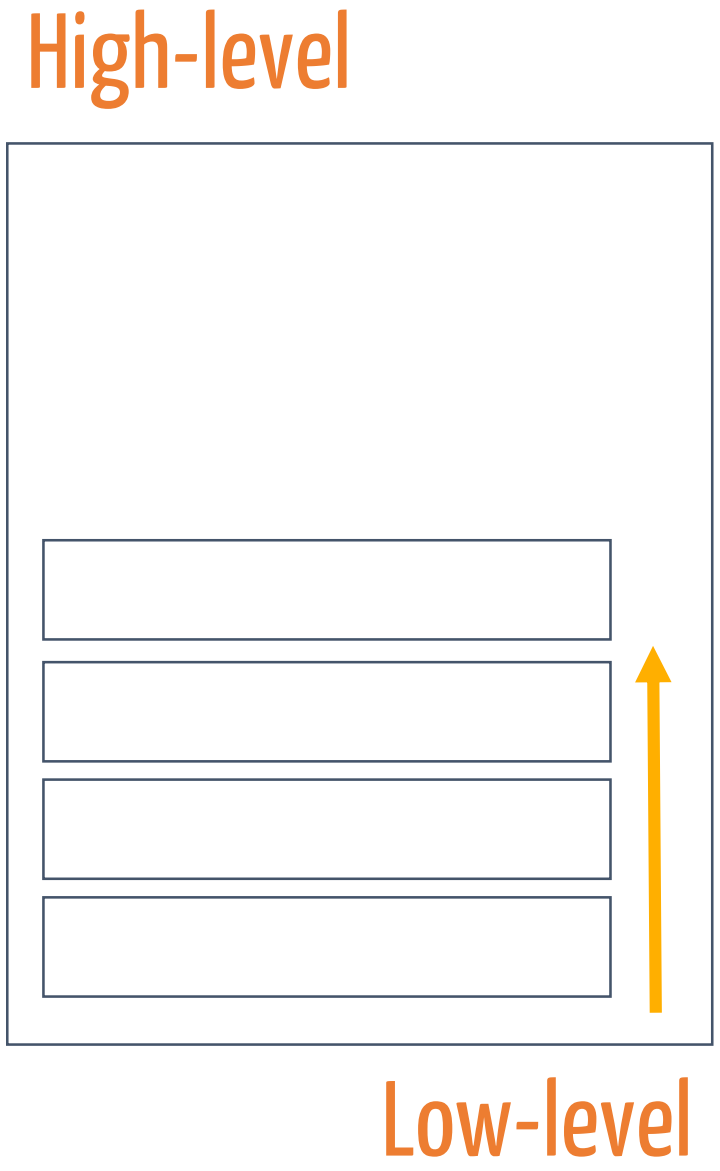
identify

Ex**P**lore

Overcome

bring together

Bring it together



Bring it together

```
void HighLevel() {
  try {
    MidLevel();
  } catch(InvalidOperationException) { }
}

void MidLevel() {
  ...
  LowLevel();
  ...
}

void LowLevel() {
}
```

Bring it together

```
void HighLevel() {
  try {
    MidLevel();
  } catch(InvalidOperationException) { }
}

void MidLevel() {
  …
  LowLevel().GetAwaiter().GetResult();
  …
}

async Task LowLevel() {
}
```

Commit. Push.

Bring it together

```
void HighLevel() {
  try {
    MidLevel().GetAwaiter().GetResult();
  } catch(InvalidOperationException) { }
}

async Task MidLevel() {
  ...
  await LowLevel().ConfigureAwait(false);
  ...
}

async Task LowLevel() {
}
```

Commit. Push.

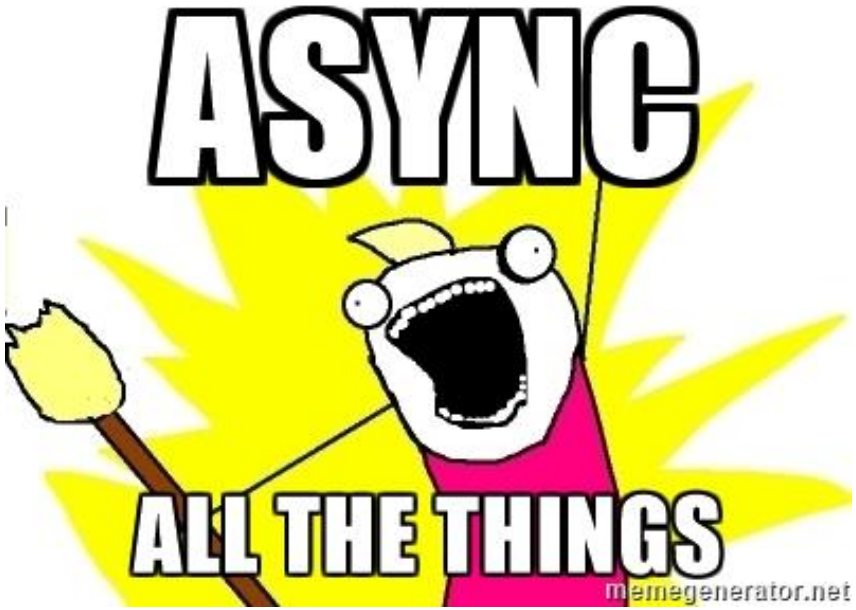
Bring it together

```
async Task HighLevel() {
  try {
    await MidLevel ().ConfigureAwait(false);;
  } catch(InvalidOperationException) { }
}

async Task MidLevel() {
  ...
  await LowLevel().ConfigureAwait(false);
  ...
}

async Task LowLevel() {
}
```

Yehaa!



Async all the way

Recap

reminder

Use iPob to move your code
step by step towards async / await

IO-bound paths benefit from async

Uniform API of Task allows to await
CPU-bound as well as IO-bound tasks

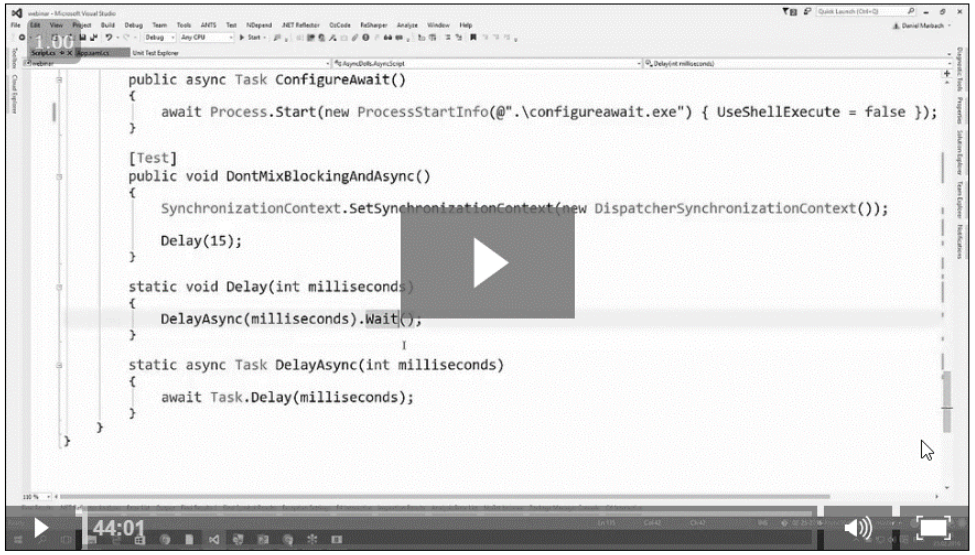
Slides, Links...

github.com/danielmarbach/RearchitectTowardsAsyncAwait

go.particular.net/ndc16.async

Async/Await Webinar Series: Best Practices

See how to avoid common pitfalls in asynchronous code bases



Share Samples Slides Comments (0)⟶

OTHER VIDEOS IN THE SERIES

▸ TPL & Message Pumps

▸ NServiceBus v6 API Update

Summary

Daniel Marbach shows how to avoid common pitfalls in asynchronous code bases.

Learn how to:

- Differentiate between IO-bound vs CPU-bound work and how this relates to Threads and Tasks
- Avoid serious production bugs as a result of asynchronous methods returning void
- Opt-out from context capturing when necessary
- Deal with synchronous code in the context of asynchronous code

Particular Software

await Q & A

Particular
Software

Thanks