Task

uniform

Task

CPU-bound

parallel

parallel

simultaneous

Task.Run
Task.Factory.StartNew
Parallel.Invoke
Parallel.For
Parallel.ForEach

Worker pool

Exercise

async

event-driven

Task

IO-bound

concurrent

concurrent

concurrent

concurrent

concurrent

interleaved

Continuation

function

async/await

simplicity

```
function1(function(err, res) {
  function2(function(err, res) {
    function3(function(err, res) {
      function4(function(err, res) {
        function5(function(err, res) {
          // do something useful
        })
      })
    })
  })
})
```

```
cleanLaundry.ContinueWith(t => {
  dryLaundry;
})
```

```
await cleanLaundry;
dryLaundry;
```

Exercise

SimpleAsync
FireAndForgot

Recap

best-practices

Use async Task instead of async void

dangerous

```
class VanillaHandler : IHandleMessages<AcquireVanilla>
{

  void Handle(AcquireVanilla message)
  {
    await AcquireVanillaFromGovernmentAsync();
    await DownloadRecipeFromBlobStorageAsync();
    await InsertVanillaUsageInDocumentDBAsync();
    await StoreTelemetryDataInEventHubAsync() ;
  }

}
```

dangerous

Error CS4033 The 'await' operator can only be used within an async method. Consider marking this method with the 'async' modifier and changing its return type to 'Task'.

dangerous

```
class VanillaHandler : IHandleMessages<AcquireVanilla>
{

  async void Handle(AcquireVanilla message)
  {
    await AcquireVanillaFromGovernmentAsync();
    await DownloadRecipeFromBlobStorageAsync();
    await InsertVanillaUsageInDocumentDBAsync();
    await StoreTelemetryDataInEventHubAsync() ;
  }

}
```

Recap

best-practices

Use async Task instead of async void

Library code and framework should use ConfigureAwait(false)

Recap

best-practices

Use async Task instead of async void

Library code and framework should use ConfigureAwait(false)

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

f... off

I don't care about your stupid async stuff!

cumbersome

```
class VanillaHandler : IHandleMessages<AcquireVanilla>
{

  void Handle(AcquireVanilla message)
  {
    AcquireVanillaFromGovernmentAsync().Result;
    DownloadRecipeFromBlobStorageAsync().Wait();
    InsertVanillaUsageInDocumentDBAsync().Result;
    StoreTelemetryDataInEventHubAsync().Wait();
  }

}
```

wasteful

```
class VanillaHandler : IHandleMessages<AcquireVanilla>
{

  void Handle(AcquireVanilla message)
  {
    AcquireVanillaFromGovernmentAsync().Result;
    DownloadRecipeFromBlobStorageAsync().Wait();
    InsertVanillaUsageInDocumentDBAsync().Result;
    StoreTelemetryDataInEventHubAsync().Wait();
  }

}
```

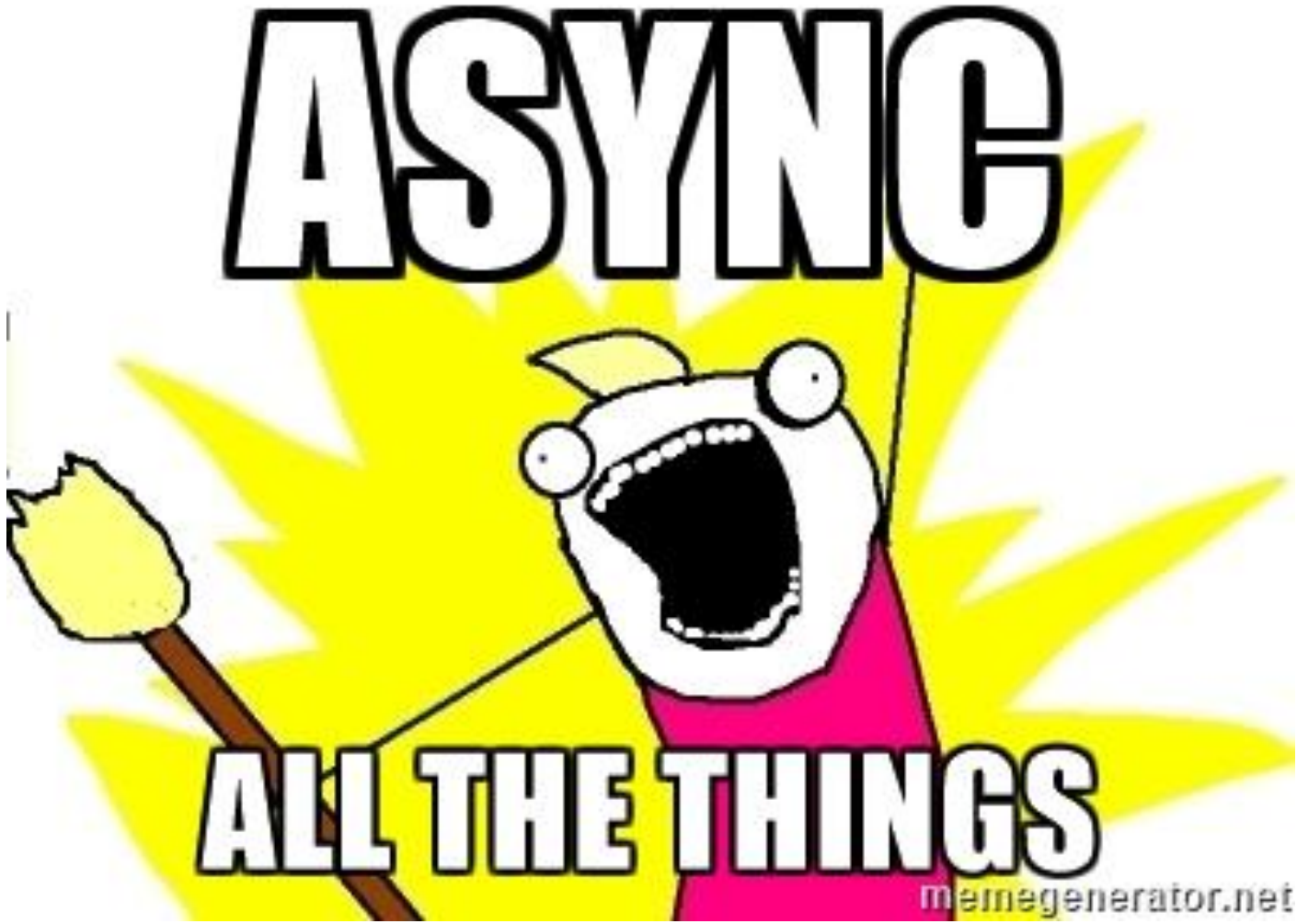
NServiceBus

Azure Service Bus	26 times
Azure Storage Queues	6 times
RabbitMQ	5 times
MSMQ	3 times

more message throughput

https://particular.net/blog/rabbitmq-updates-in-nservicebus-6

https://github.com/Particular/EndToEnd/tree/master/src/PerformanceTests


ASYNC
ALL THE THINGS
memegenerator.net

Async / await

is viral

but

It kicks your

servers

butt

Task.Run
Task.Factory.StartNew
Parallel.For
Parallel.ForEach

Worker pool

IO pool

await iobound
iobound.FireForget()

Task.Run
Task.Factory.StartNew
Parallel.For
Parallel.ForEach

Worker pool

IO pool

await iobound
iobound.FireForget()

Exercise

AsyncVoid
ConfigureAwait
AsyncAllTheWay
SequentialExecution

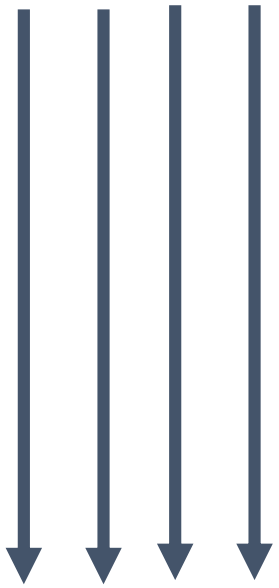
Implicit execution concurrency
Code flow is sequential

```
await Method1();
try {
  await Method2();
} catch() { }
```

If you stick to that almost nothing can go wrong

Dangerzone

Explicit execution concurrency



```
Method1Async();
Method2Async();
WhenAll / WhenAny
ContinueWith
```

From now on more than a code monkey brain is required

Ambient state

Ambient state


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

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

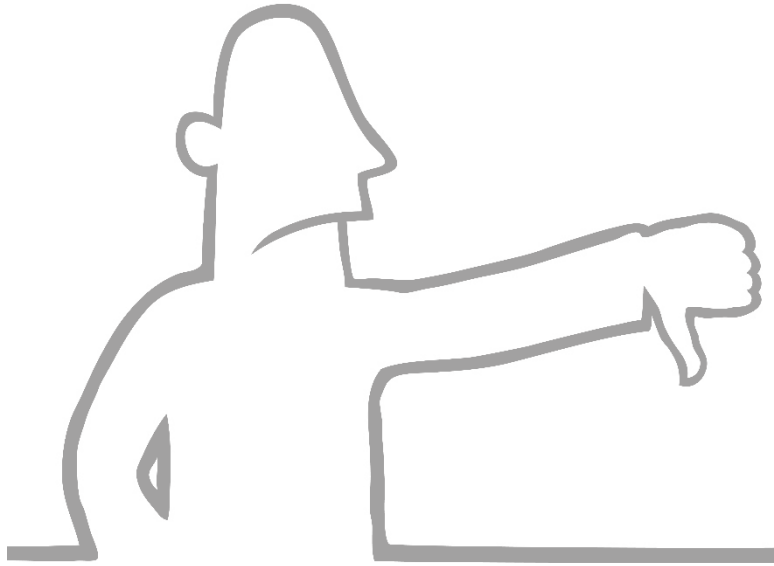
Ambient state


```
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);
```

AmbientState passed

```
05:50:09:187: Thread: 4, Value: 2
05:50:09:187: Thread: 8, Value: 2
05:50:09:187: Thread: 9, Value: 2
05:50:09:390: Thread: 4, Value: 3
05:50:09:391: Thread: 9, Value: 3
05:50:09:391: Thread: 8, Value: 3
```



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

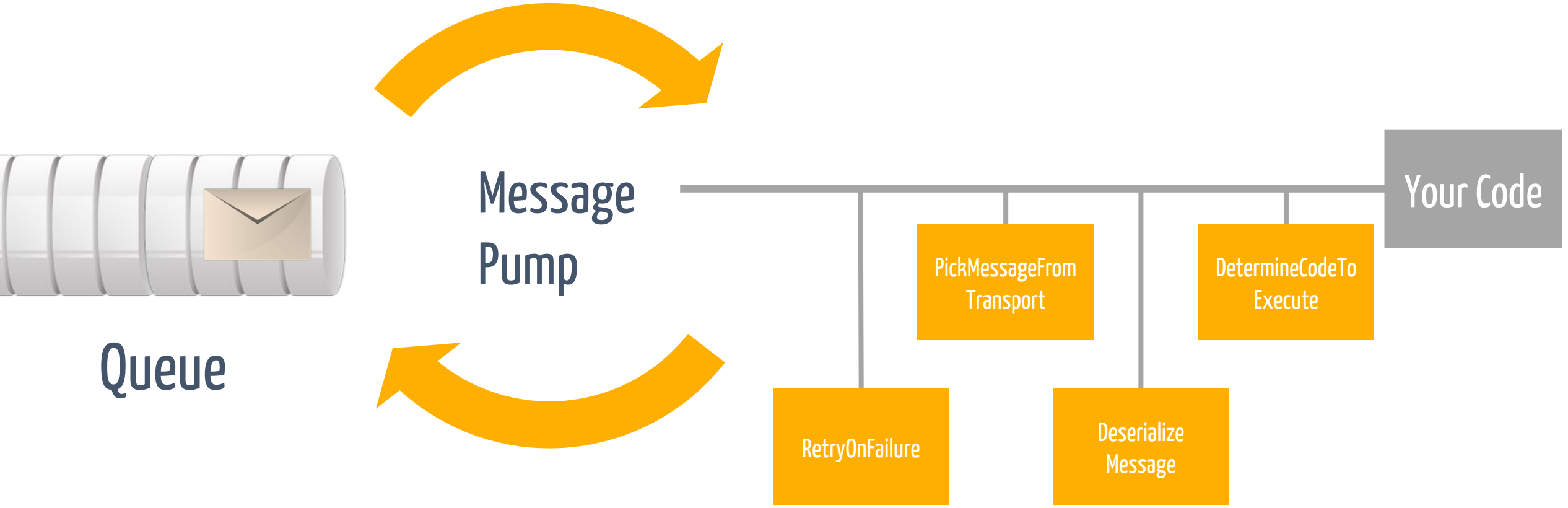
Remember

Forget thread!

think Task

Exercise

DangerZone
All the other exercises
until Concurrency Limit ;)


Message
Pump
Queue
Your Code
PickMessageFrom
Transport
DetermineCodeTo
Execute
RetryOnFailure
Deserialize
Message

Exercise

Your Pump

Exercise

Multi Producer Concurrent Consumer

Testing

Debugging