Particular Software

Welcome

Async / Await

The die is cast

Particular Software

Solution Architect
Enthusiastic Software Engineer
Microsoft Azure MVP

@danielmarbach
particular.net/blog
planetgeek.ch

Goals

target

CPU-bound vs IO-bound

Threads and Tasks

Async best-practices

Why async is the future


Terminology
Why
WrapUp


Terminology
Why
WrapUp

async

event-driven

Task

uniform

Task

IO-bound

Task

CPU-bound

concurrent

concurrent

concurrent

concurrent

concurrent

interleaved

parallel

parallel

simultaneous

Continuation

function


Terminology
Why
WrapUp

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;
```

await Demo

javascript

ES2015

```
function chainAnimationsPromise(elem, animations) {
    let ret = null;
    let p = currentPromise;
    for(const anim of animations) {
      p = p.then(function(val) {
        ret = val;
        return anim(elem);
      })
    }
    return p.catch(function(e) {
      /* ignore and keep going */
    }).then(function() {
      return ret;
    });
  }
```

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

Try it out babeljs.io

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

```
"use strict";

var chainAnimationsAsync = function () {
    var ref = _asyncToGenerator(regeneratorRuntime.mark(function _call
        var ret, _iteratorNormalCompletion, _didIteratorError, _iterat

        return regeneratorRuntime.wrap(function _callee$(_context) {
            while (1) {
                switch (_context.prev = _context.next) {
                    case 0:
                        ret = null;
                        _context.prev = 1;
                        _iteratorNormalCompletion = true;
                        _didIteratorError = false;
                        _iteratorError = undefined;
                        _context.prev = 5;
                        _iterator = animations[Symbol.iterator]();

                    case 7:
                        if (_iteratorNormalCompletion = (_step = _iter
                            _context.next = 15;
                            break;
                        }

                        anim = _step.value;
                        _context.next = 11;
                        return anim(elem);

                    case 11:
                        ret = _context.sent;
```

dart

release 1.9

```
runUsingFuture() {
  //...
  findEntrypoint().then((entrypoint) {
    return runExecutable(entrypoint, args);
  }).then(flushThenExit);
}
```

dart

release 1.9

```
runUsingAsyncAwait() async {
  //...
  var entrypoint = await findEntrypoint();
  var exitCode = await
      runExecutable(entrypoint, args);
  await flushThenExit(exitCode);
}
```

python

release 3.5

```
import asyncio

async def http_get(domain):
    reader, writer =
        await asyncio.open_connection(domain, 80)

async for line in reader:
    print('>>>', line)
```

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

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();
  }

}
```

await Demo

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() ;
  }

}
```

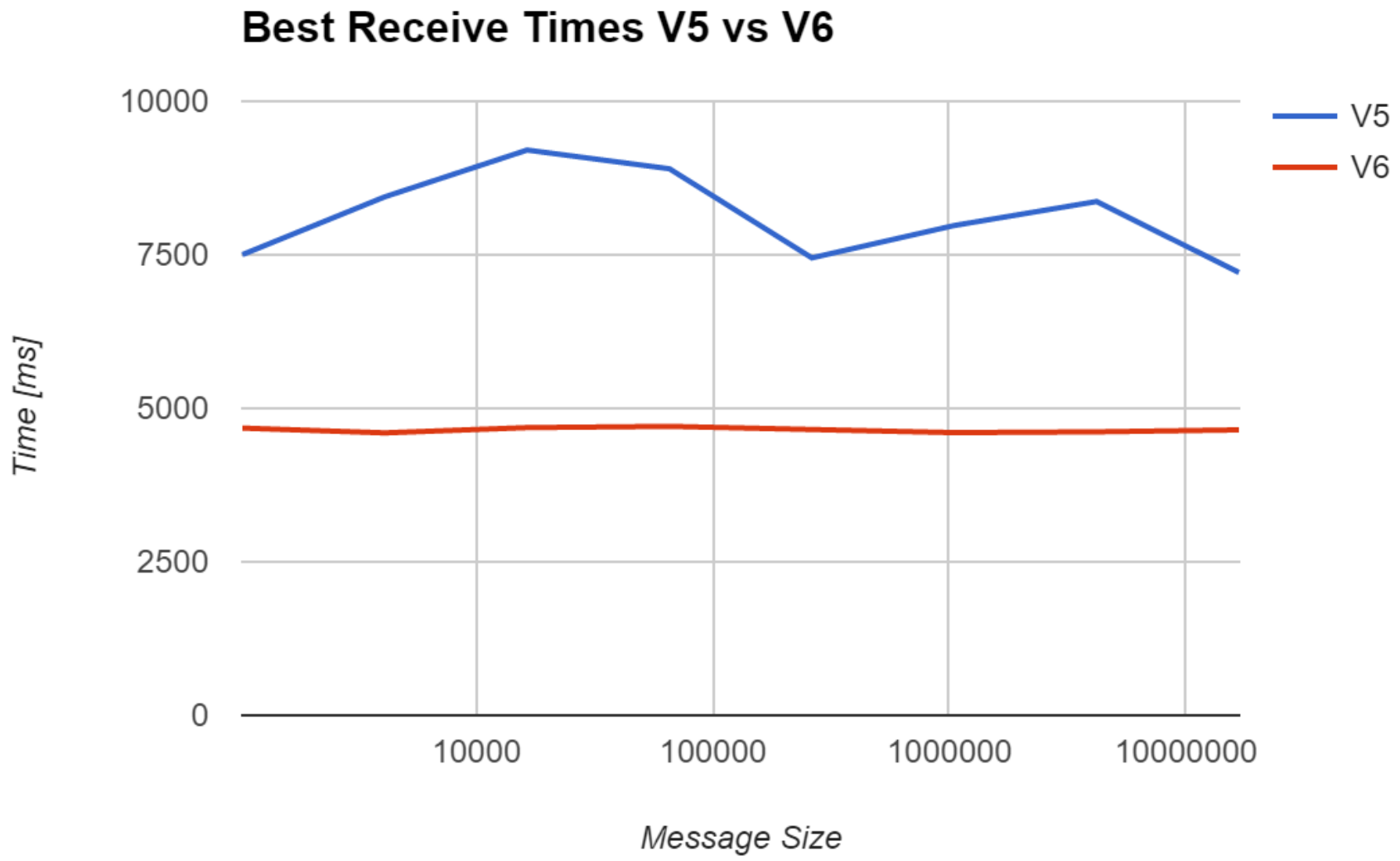
await Demo

NServiceBus

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

more message throughput

NServiceBus.SqlServer




Terminology
Why
WrapUp

Forget thread!

think Task

The die

is cast

NSB v6

Will be Async all the way

Applies the shown best-practies like ConfigureAwait(false) consequently, checked with Roslyn analyzer

particular.net/blog/async-await-its-time

Recap

reminder

Use Task.Run, Factory.StartNew for CPU-bound work

Use Task directly for IO-bound work

Use async Task instead of async void

Recap

reminder

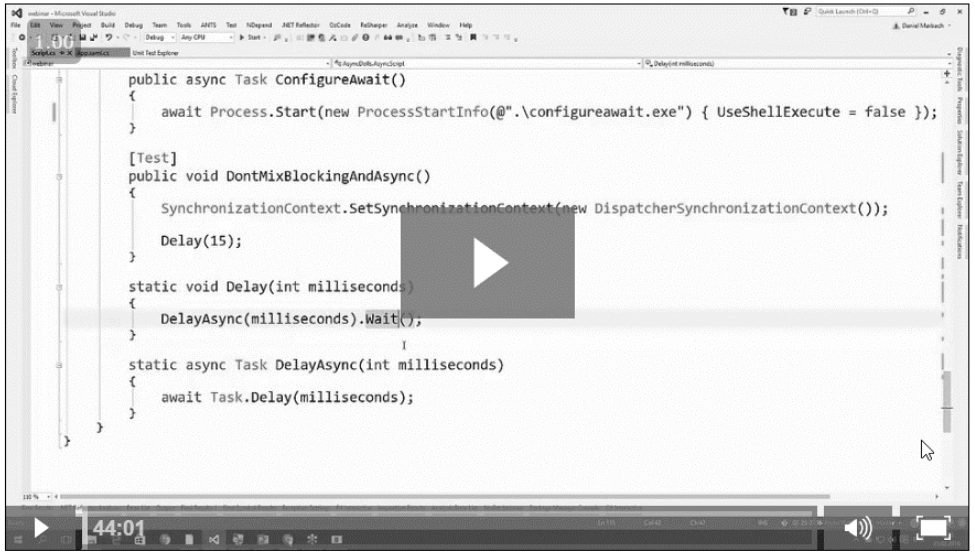
Libraries and frameworks should use ConfigureAwait(false)

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

go.particular.net/DNC016

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

Slides, Links...

github.com/danielmarbach/Async.DielsCast

Particular Software

await Q & A

Particular
Software

Thanks