Faster Code; More time for 

How to try to avoid writing the slowest code possible

Repo: https://github.com/mattg23/csharp_perf_talk


C'mon,
do something...

- Optimization vs. *not writing the slowest code possible*
  - When? From the beginning?
  - **Be aware** of the *context* you are in!
- Tradeoffs:
  - Dev time
  - Readability
  - [Portability] (mostly non-issue in .NET)
- Learning by doing!

What to think about when programming?

- Stack vs. Heap
- Value vs. Reference Types
- Loops vs. Linq
- Process data while its loading (Streaming vs. Non-Streaming)
- Parallel processing
- Batching / Chunking

```
var bytes = new byte[...some large byte array..];
var currentTS = BitConverter.ToDouble(
    bytes.Skip(SOME_INDEX).Take(TS_SIZE_IN_BYTES).Reverse().ToArray(), 0);
```

```
ReadOnlySpan<byte> spanBytes
    = bytes[SOME_INDEX..(SOME_INDEX + TS_SIZE_IN_BYTES)];
var currentTs = BinaryPrimitives.ReadDoubleBigEndian(spanBytes);
```

Less slide more code pls 🥱

Take aways:

- Measure - do not optimize blindly
- Use the knowledge you have!
  - what do you know about the values you work with?
  - contraints of the data format you are parsing?
- Think about bottlenecks:
  - Network, DB, SSD/HDD
  - external systems
  - cpu speed, memory throughput
- `Span<T>` is your friend in .NET
- Think before use:
  - Linq
  - Reflection

Other .NET Classes to checkout:

- `BinaryPrimitives`
- `MemoryMarshal`
- `CollectionsMarshal`
- `ArrayPool<T>`
- `MemoryPool<T>`
- `MemoryExtensions`

Other Resources:

- Paper: What every programmer should know about Memory
- Casey Muratori
- BenchmarkDotNet
- .NET Performance Tips