Deep dive on native AOT

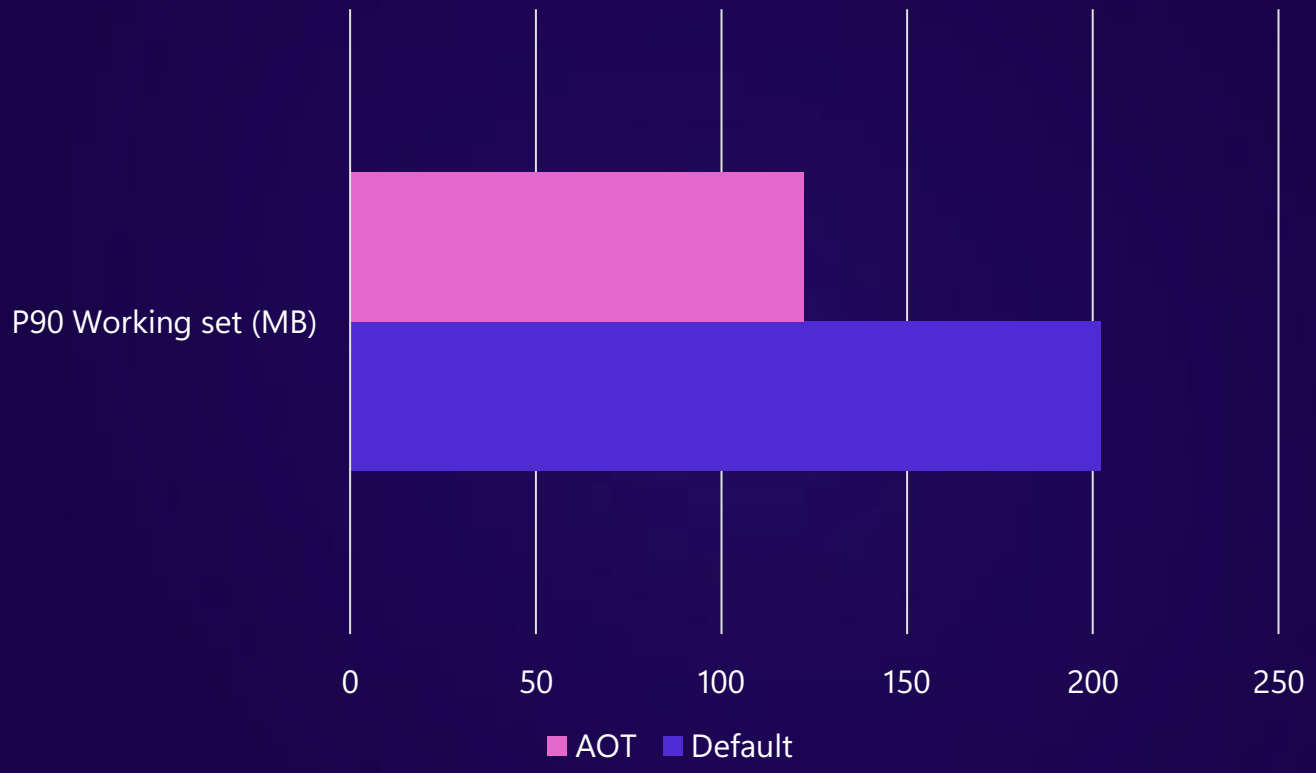
Michal Strehovský

Why native AOT?

- Less memory use
- Faster startup
- Smaller apps

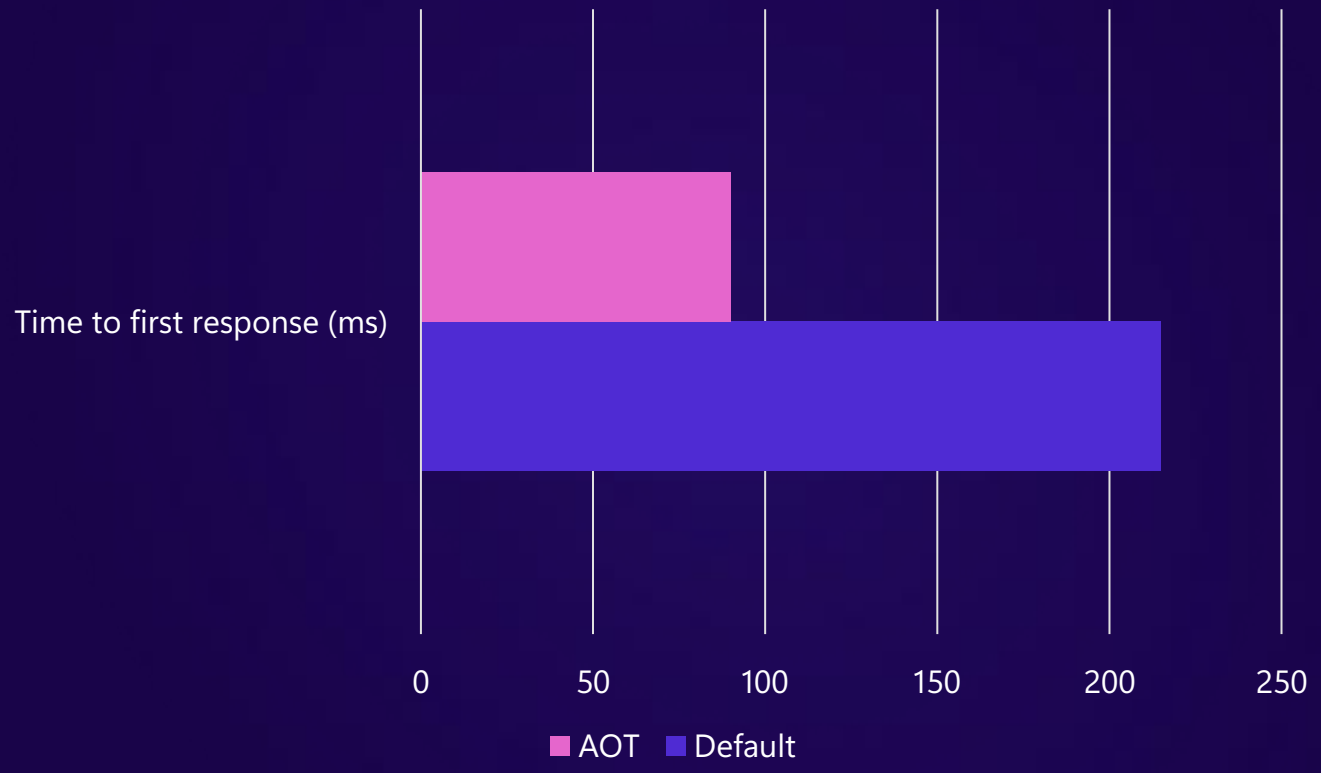
Less memory use

Private memory in active use while under load serving HTTP requests.

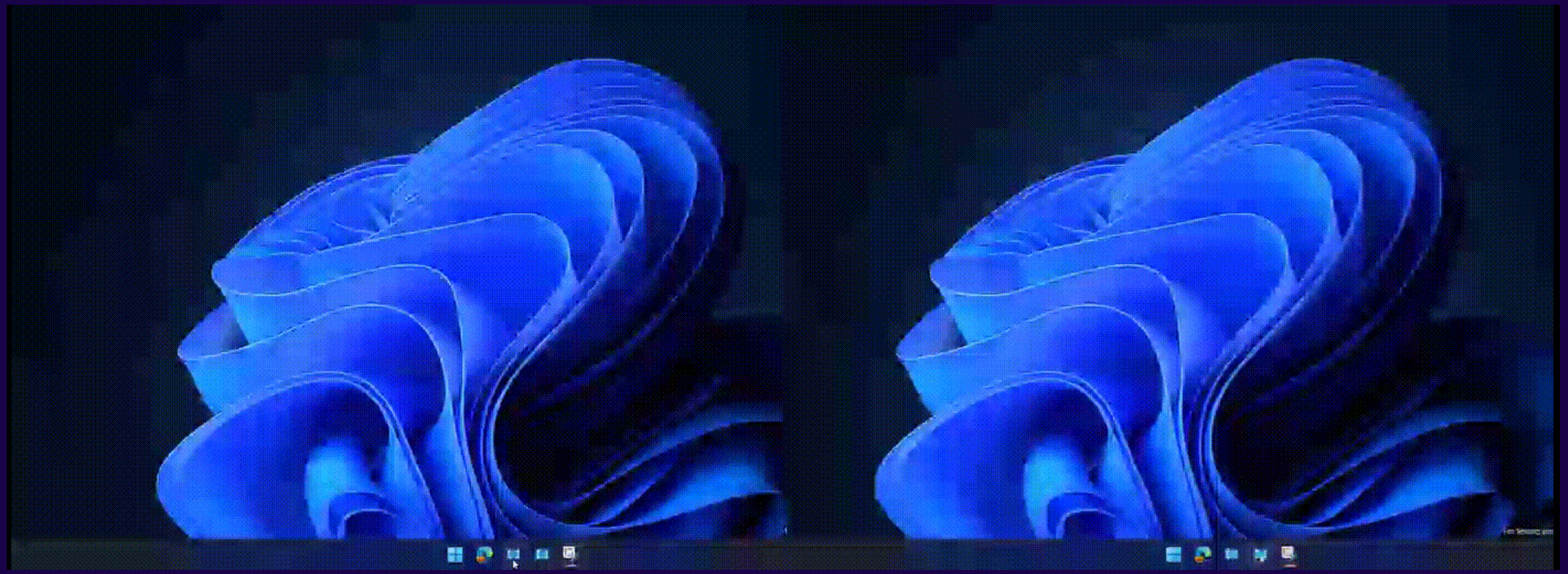


Faster startup

Time it takes until an ASP.NET web API responds to the fist HTTP request



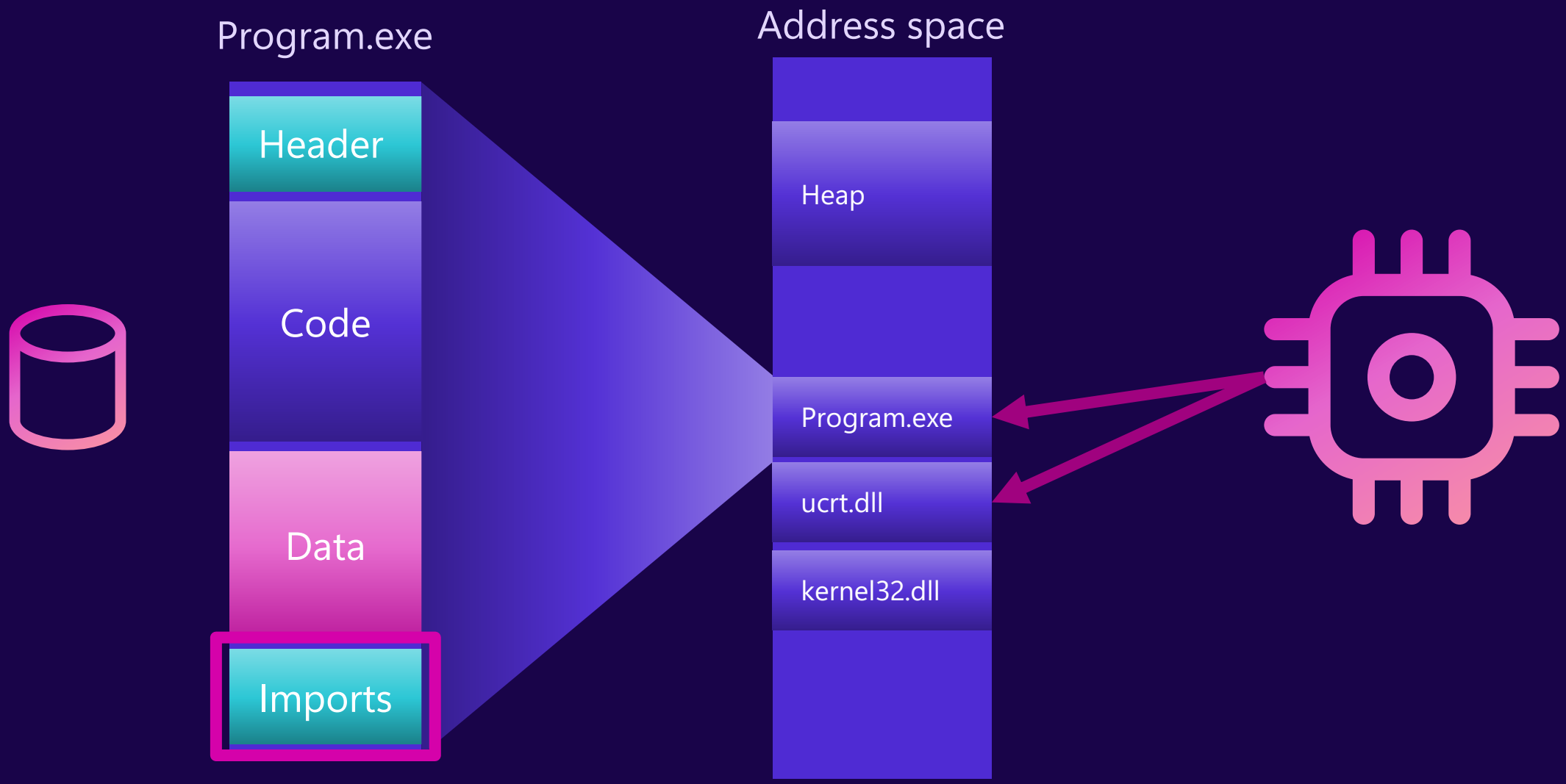
Faster startup



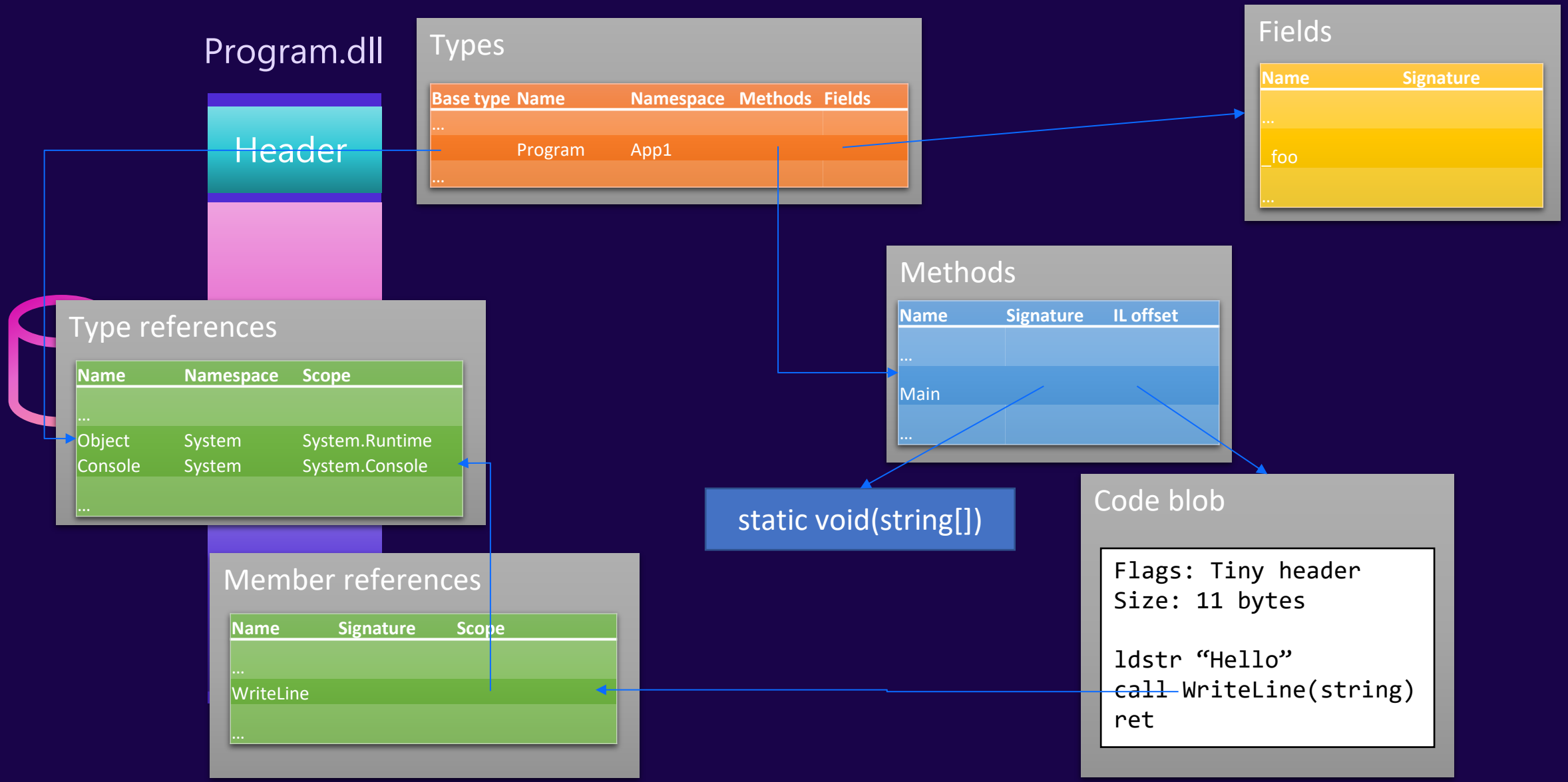
Default

With native AOT

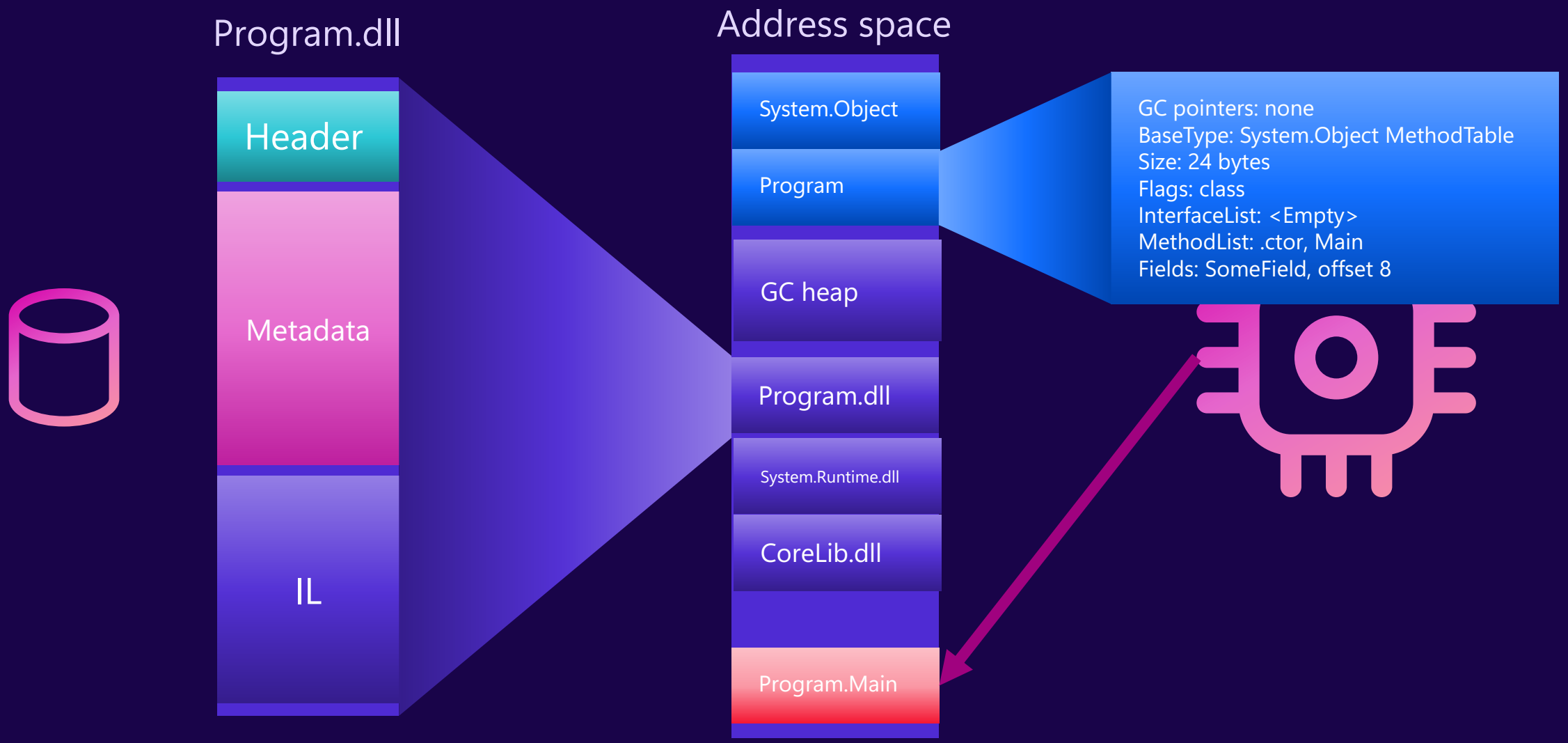
Starting a native process



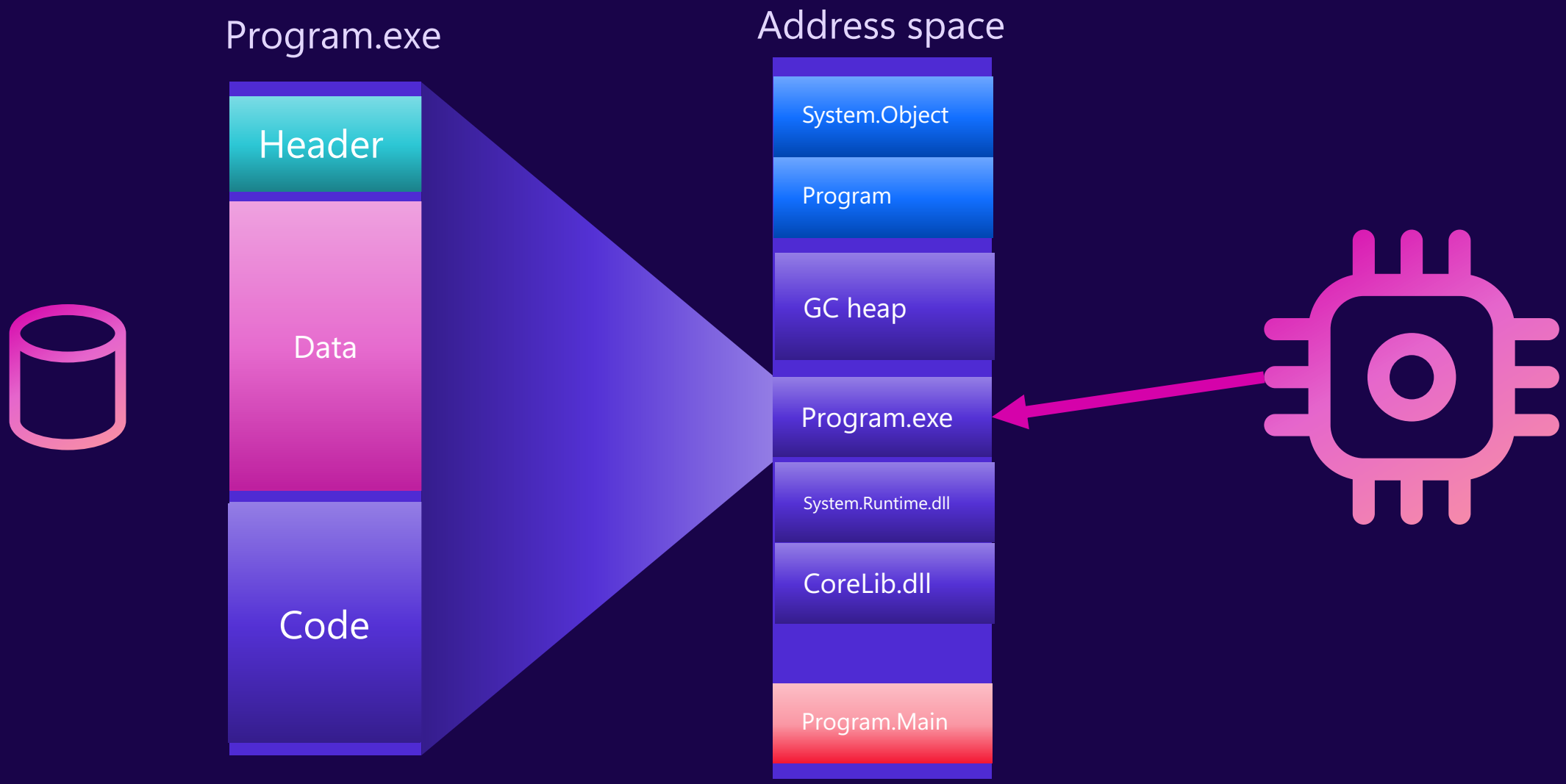
Starting a VM-based process



Starting a VM-based process



Starting a native AOT process

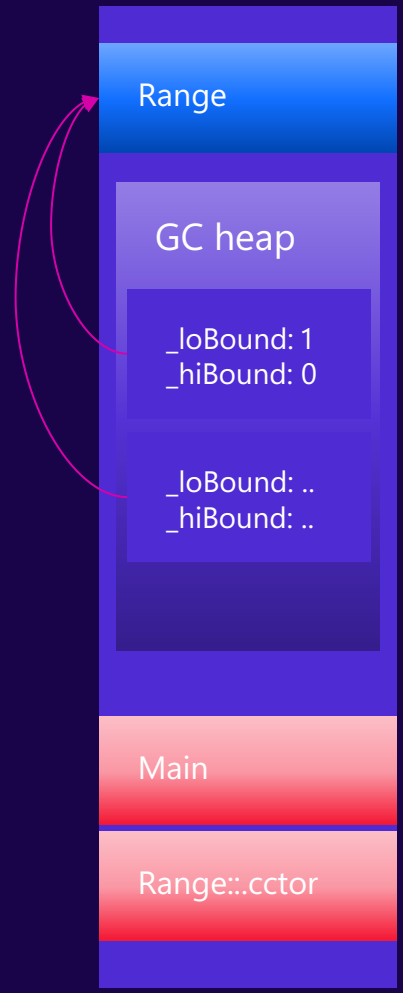


AOT everything

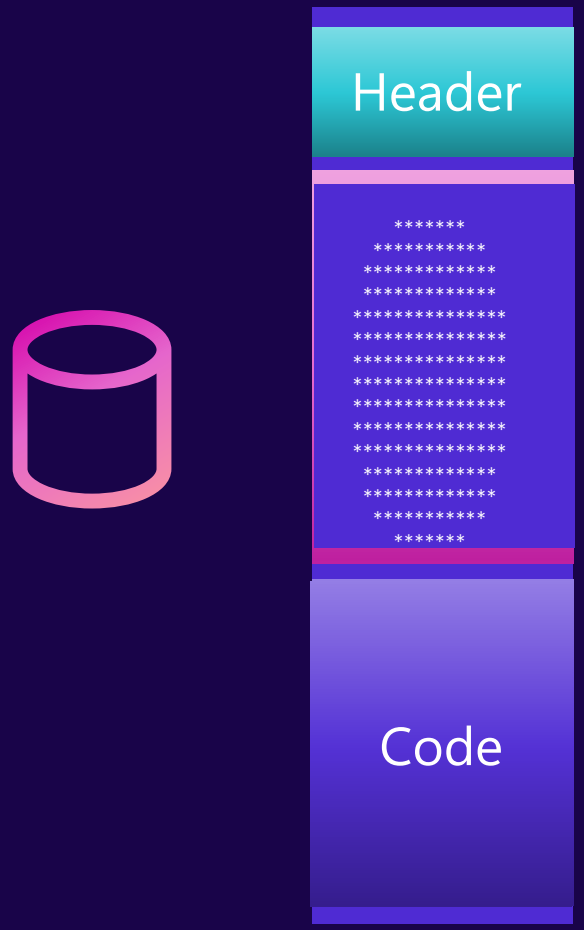
Program.exe


```
Console.WriteLine(Range.Empty.Includes(5));
Console.WriteLine(Range.Full.Includes(5));

class Range(int _loBound, int _hiBound)
{
    public bool Includes(int val) =>
        val >= _loBound && val <= _hiBound;

    public static readonly Range Empty
        = new Range(1, 0);

    public static readonly Range Full
        = new Range(int.MinValue, int.MaxValue);
}
```

Address space



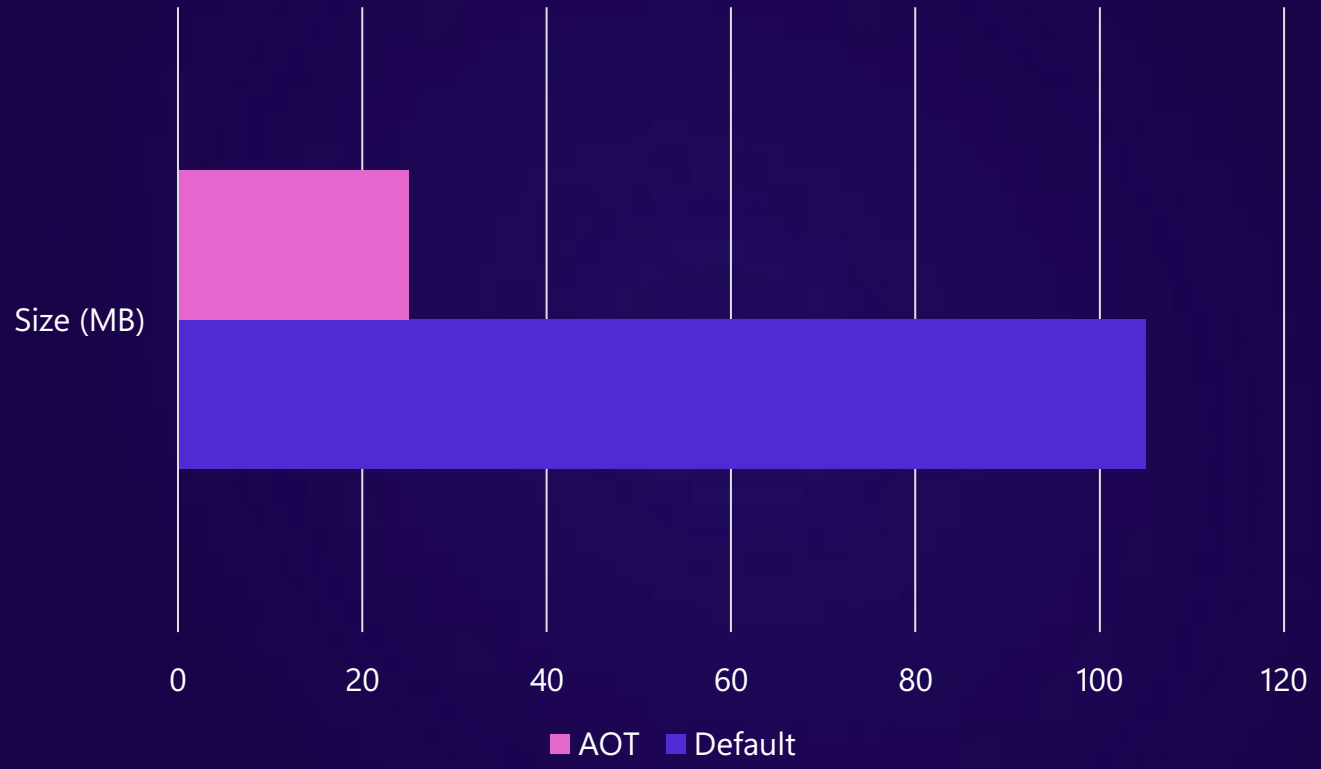
AOT everything



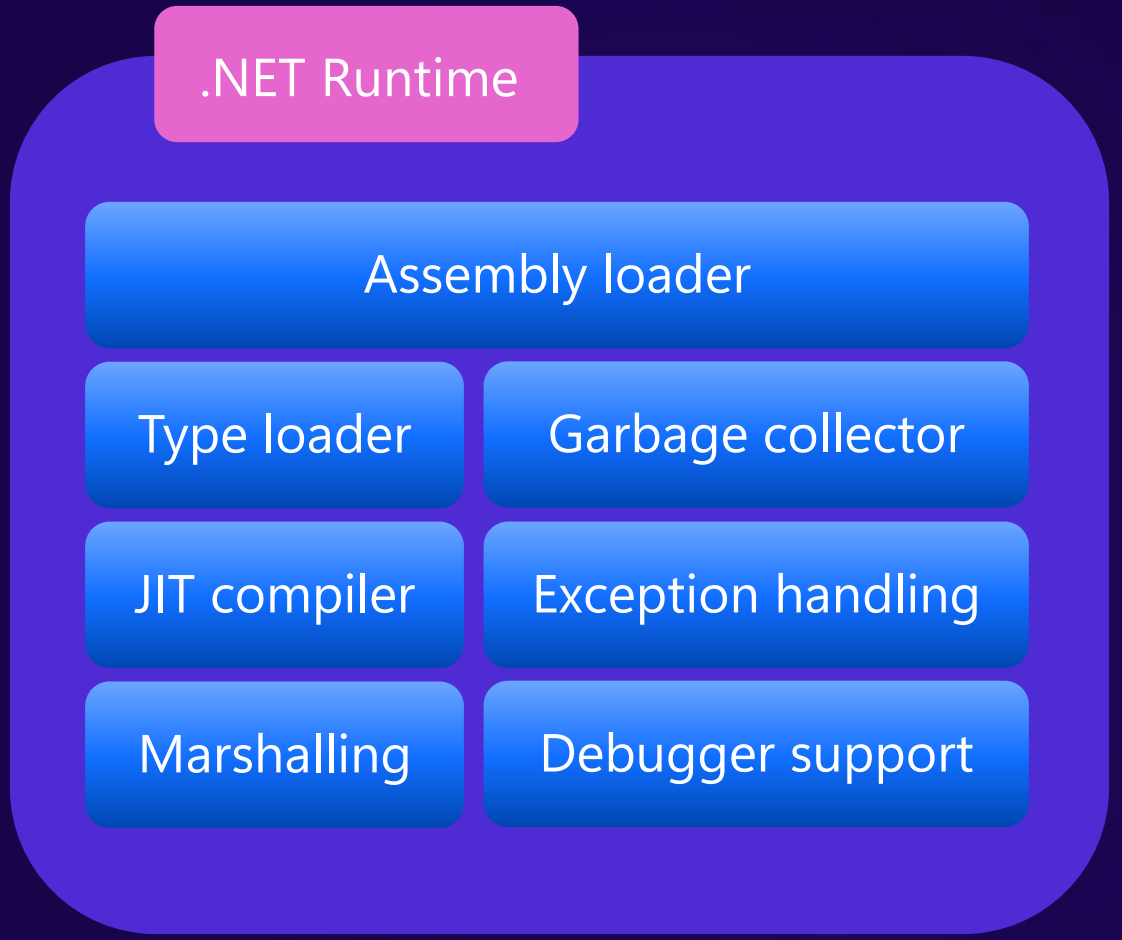

```
using System.Text;

Console.WriteLine(Encoding.ASCII.GetString(Circle.Bytes));

static class Circle
{
    public static readonly byte[] Bytes = ComputeCircleBytes();

    private static byte[] ComputeCircleBytes()
    {
        const int Width = 16;

        byte[] bytes = new byte[Width * Width];
        for (int i = 0; i < bytes.Length; i++)
        {
            int x = i % Width;
            int y = i / Width;

            if (x == 0)
            {
                bytes[i] = (byte)'\n';
                continue;
            }

            x -= Width / 2;
            y -= Width / 2;

            if (x * x + y * y < (Width / 2) * (Width / 2))
                bytes[i] = (byte)'*';
            else
                bytes[i] = (byte)' ';
        }

        return bytes;
    }
}
```

Smaller size

Size of a fully self-contained ASP.NET app

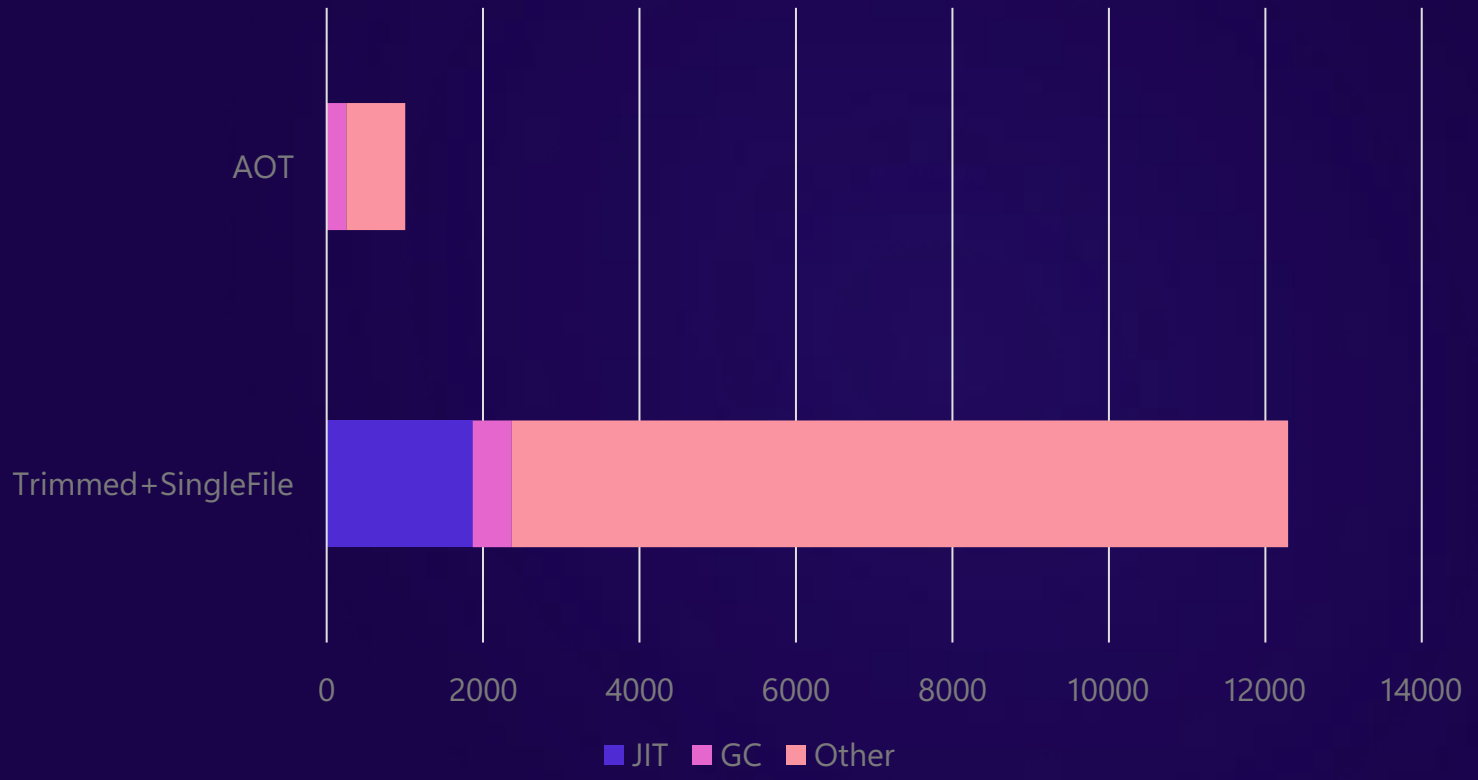


Lightweight runtime



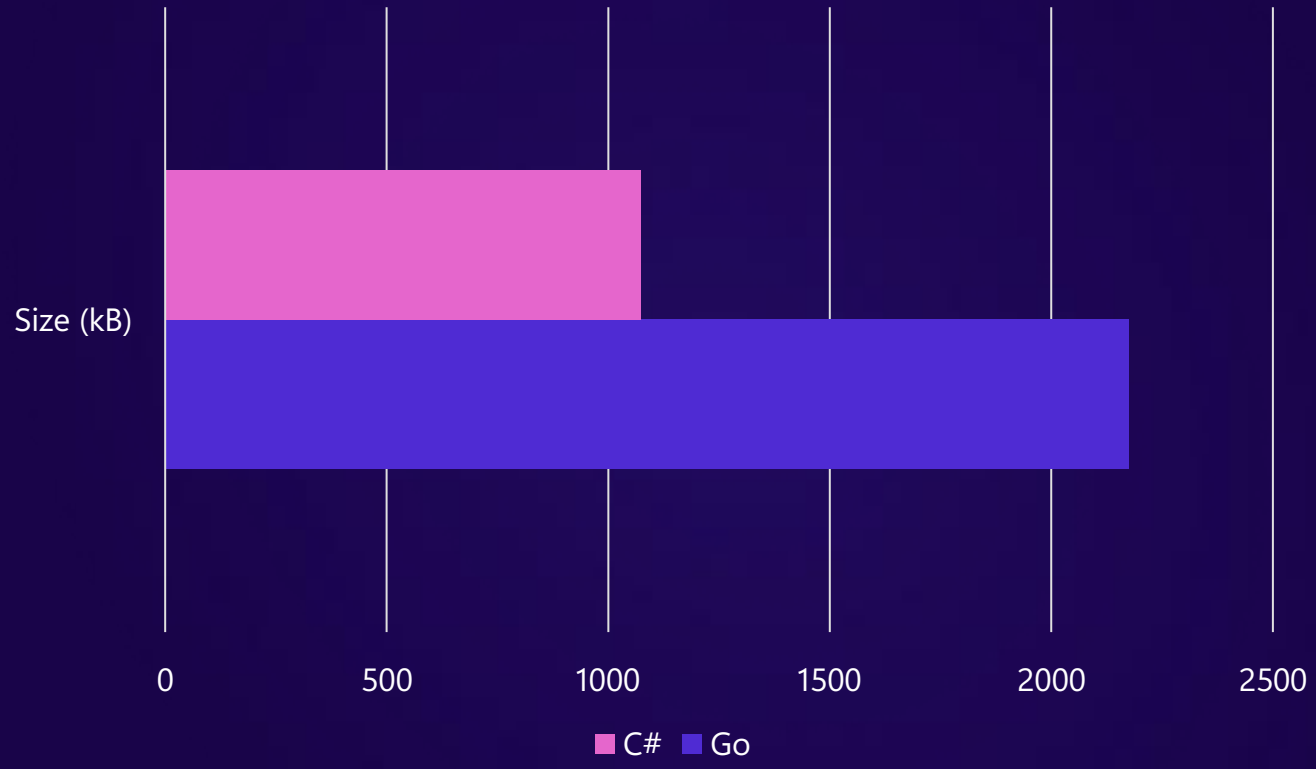
Smaller size

Size of a hello world app



Size in context

Size of a hello world in Go vs C# AOT



Dead code elimination

- Smallest code is code that wasn’t generated
- Starting with Main, look for all methods reachable from it
- Continue expanding until there’s nothing left to add

Whole program optimizations

- Optimizations that are possible because the AOT compiler can see the entire program and no new code can be created at runtime

Eliminating dead branches

Making static fields read-only

- Fields that are not written and not visible targets of reflection can be made read only
- Their value can be inlined by the code generator and provide an opportunity for dead code elimination

```
Worker.Shutdown();

class Worker
{
    static int s_workerCount;

    public static void AddWorker(Worker w)
    {
        Interlocked.Increment(ref s_workerCount);

        // ...
    }

    [MethodImpl(MethodImplOptions.NoInlining)]
    public static void Shutdown()
    {
        while (s_workerCount > 0)
        {
            Console.WriteLine("Waiting for workers to finish");
            Thread.Sleep(100);
        }
    }
}
```

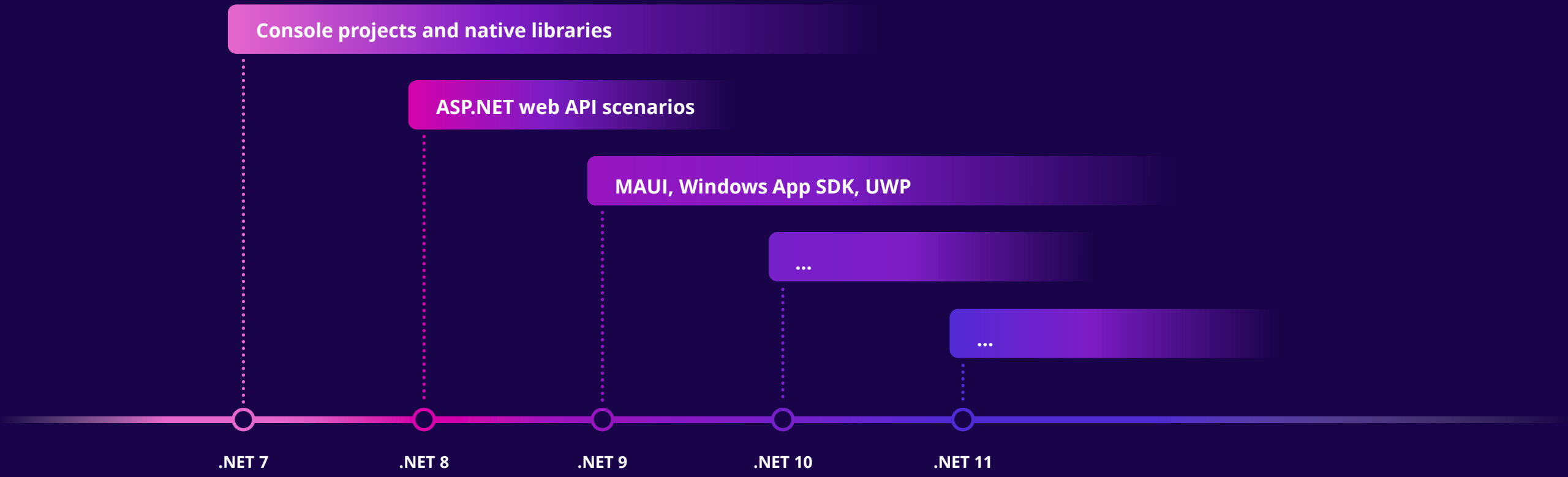
Eliminating dead branches

Removing dead type checks

- Types that are never allocated will never satisfy a type check

```
class UnusedType { }

class Program
{
    public static void HandleObject(object o)
    {
        if (o is UnusedType)
        {
            Console.WriteLine("Handling UnusedType!");
        }
    }

    static void Main()
    {
        HandleObject(new object());
    }
}
```

Third .NET release with native AOT



Get .NET 9

Download .NET 9
aka.ms/get-dotnet-9

Resources

https://learn.microsoft.com/dotnet/core/deploying/native-aot/

Native AOT overview

https://learn.microsoft.com/dotnet/core/deploying/trimming/trim-self-contained

Overview of trimming

https://www.youtube.com/watch?v=N-MrQeZ1enY

Deep .NET talk on native AOT

https://www.nuget.org/packages/sizoscope

Tool to investigate binary size of native AOT apps

https://devblogs.microsoft.com/ifdef-windows/preview-uwp-support-for-dotnet-9-native-aot/

.NET 9 support for UWP

https://devblogs.microsoft.com/dotnet/testing-your-native-aot-dotnet-apps/

Testing native AOT configuration with MSTest

https://devblogs.microsoft.com/dotnet/creating-aot-compatible-libraries/

Creating AOT compatible libraries

Thank you