

Welcome

Async / Await

Chain of Responsibility



```
graph LR; A[Pattern] --> B[Build it]; B --> C[WrapUp]
```

Pattern

Build it

WrapUp



Pattern

Build It

WrapUp

OWIN

```
appBuilder.Use(async (ctx, next) =>
{
    // do some things here
    await next();
    // or here
});
```

ASP.NET Core Middleware

```
public class Startup {  
    public void Configure(IApplicationBuilder app) {  
        app.Use(async (context, next) =>  
        {  
            // do some things here  
            await next.Invoke();  
            // or here  
        });  
    }  
}
```

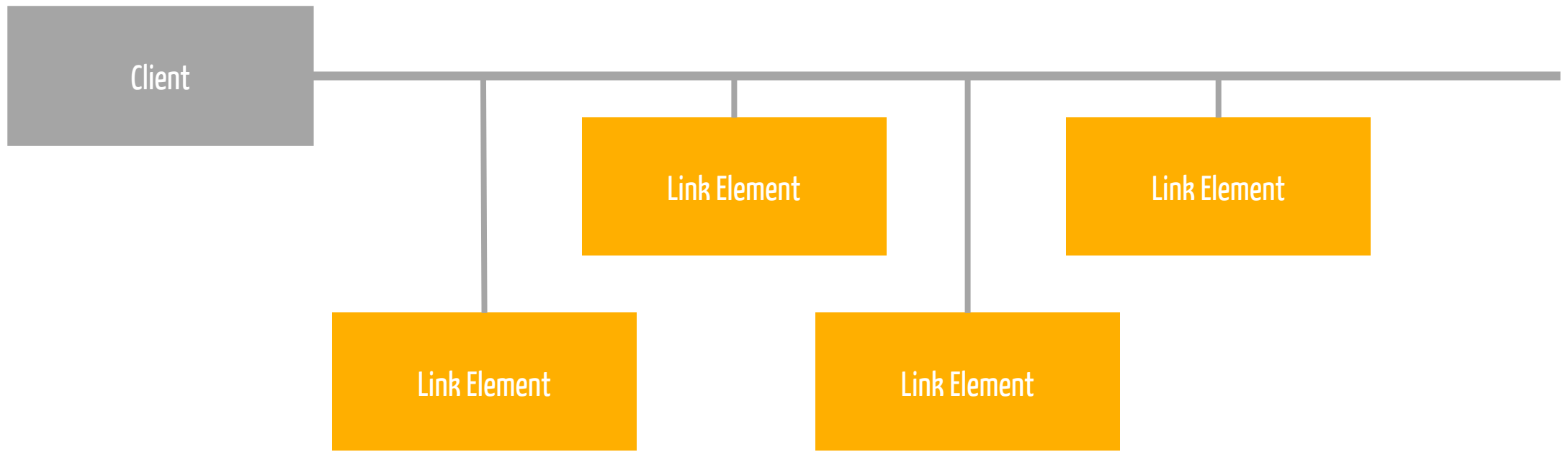
ASP.NET Core Filters

```
class ActionFilter : IAsyncActionFilter {  
    public async Task OnActionExecutionAsync(  
        ActionExecutionContext context, ActionExecutionDelegate next)  
    {  
        // do some things here  
        var resultContext = await next();  
        // or here  
    }  
}
```

Goals

target

Chain of Responsibility





son wife husband

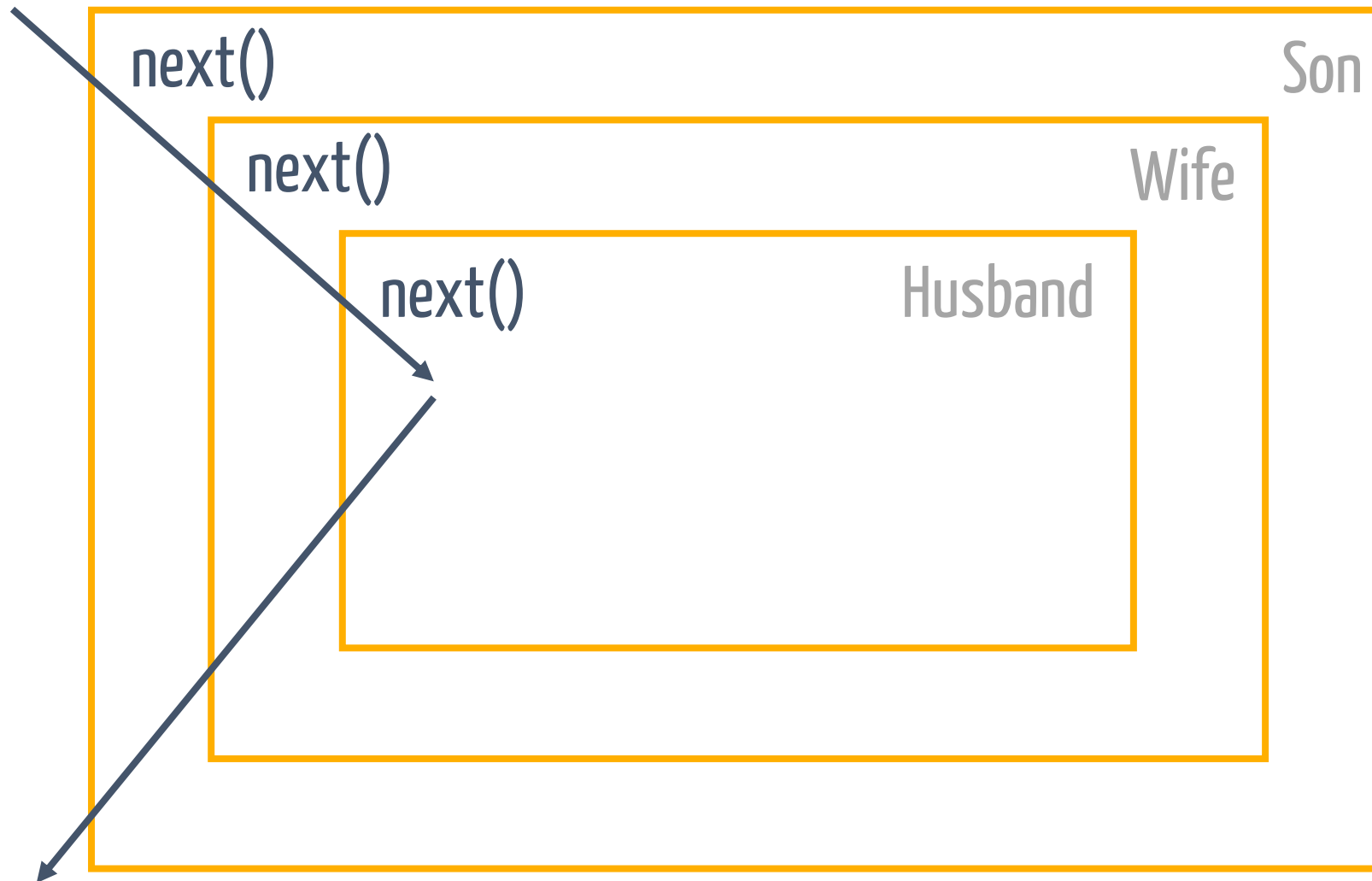


son wife husband

```
static void Person(Action next)
{
    // Implementation
    next();
}
```

```
public void ManualDishwasherUnloading()  
{  
    Son(()) => Wife(()) => Husband(()) => Done());  
}
```

Demo



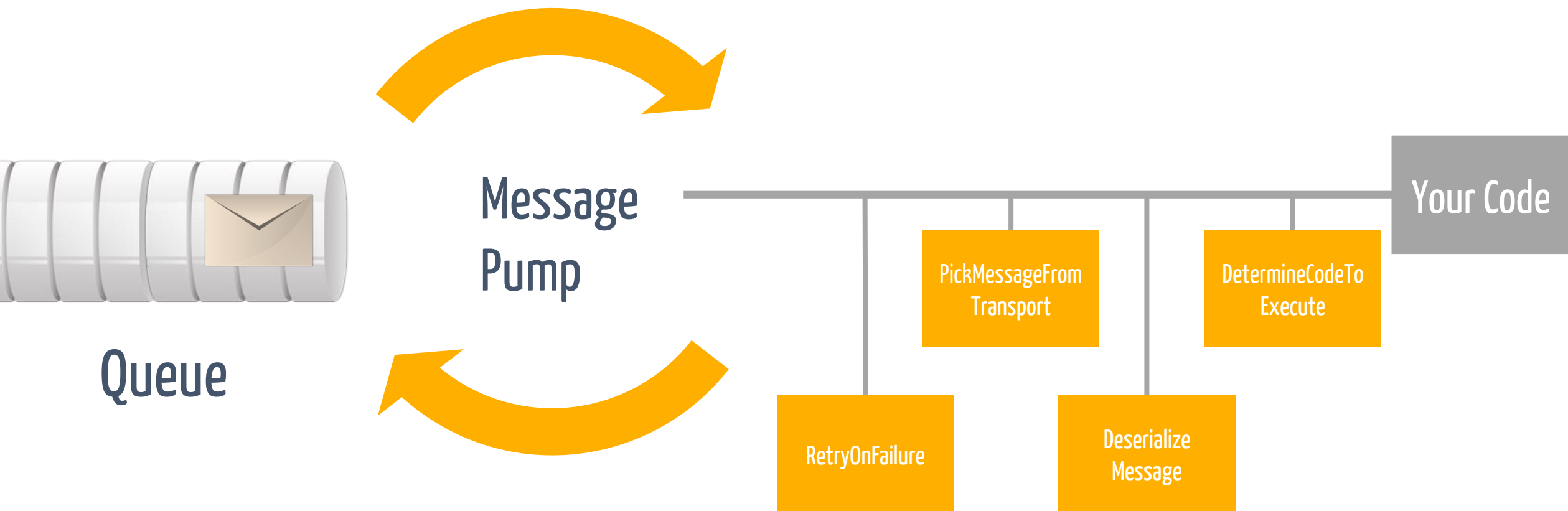
cumbersome

await Demo

missing ConfigureAwait(false) ;)


```
static void IgnoreDishStillWetException(Action next)
{
    try {
        next();
    }
    catch(DishStillWetException) { }
}
```

await Demo





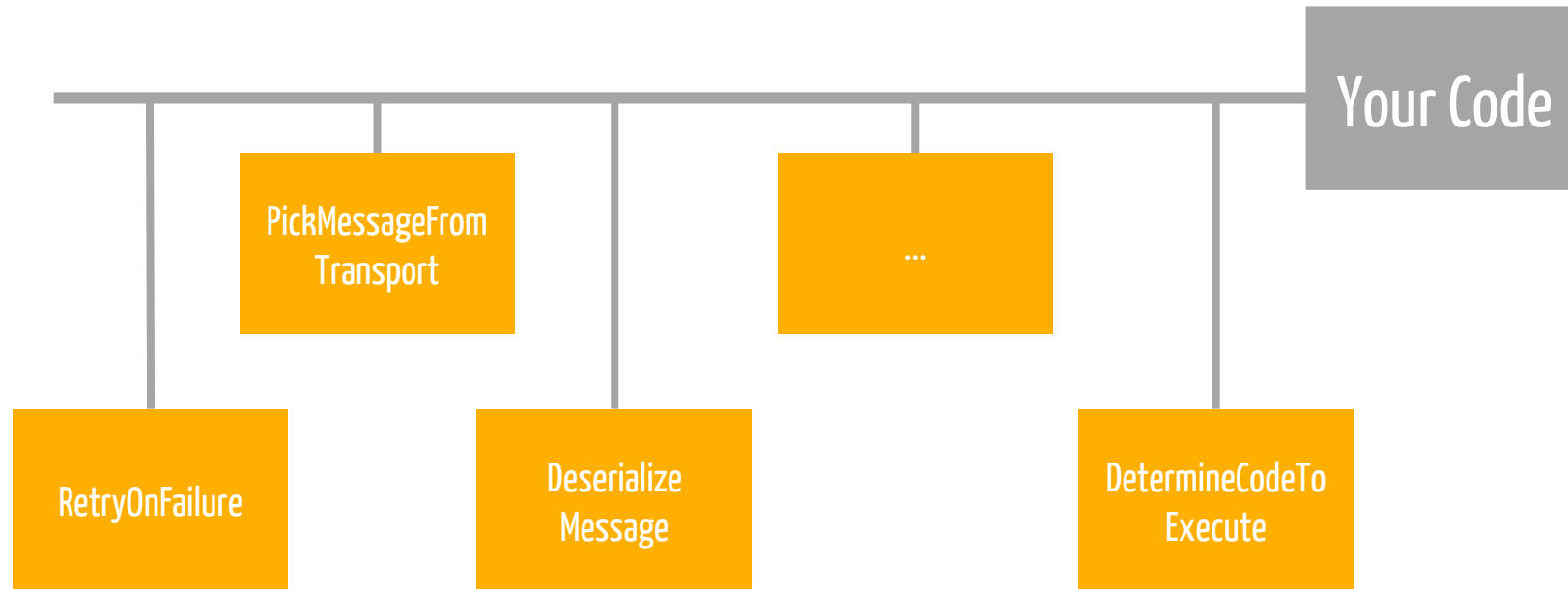
```
graph LR; A[Pattern] --> B[Build It]; B --> C[WrapUp];
```

Pattern

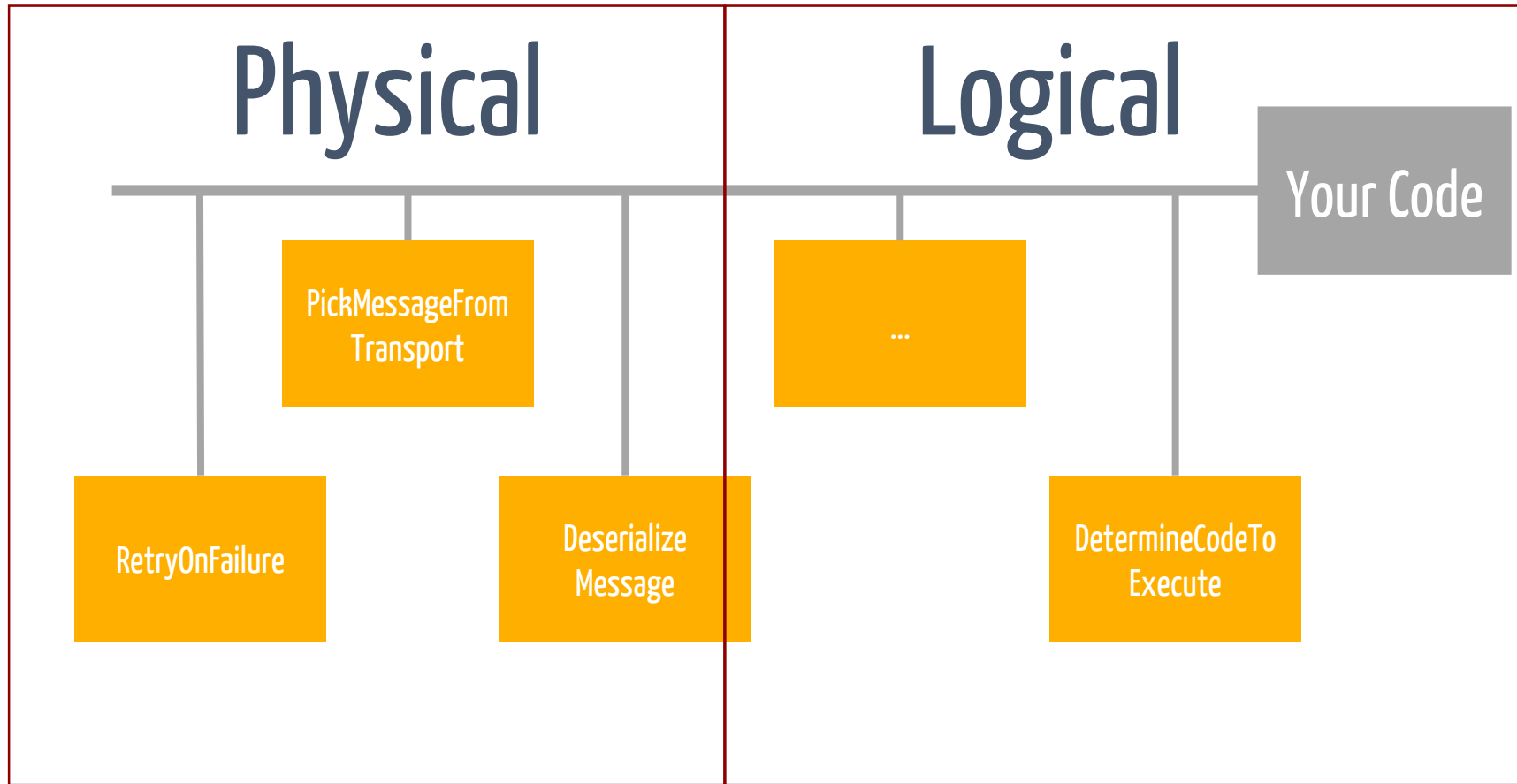
Build It

WrapUp

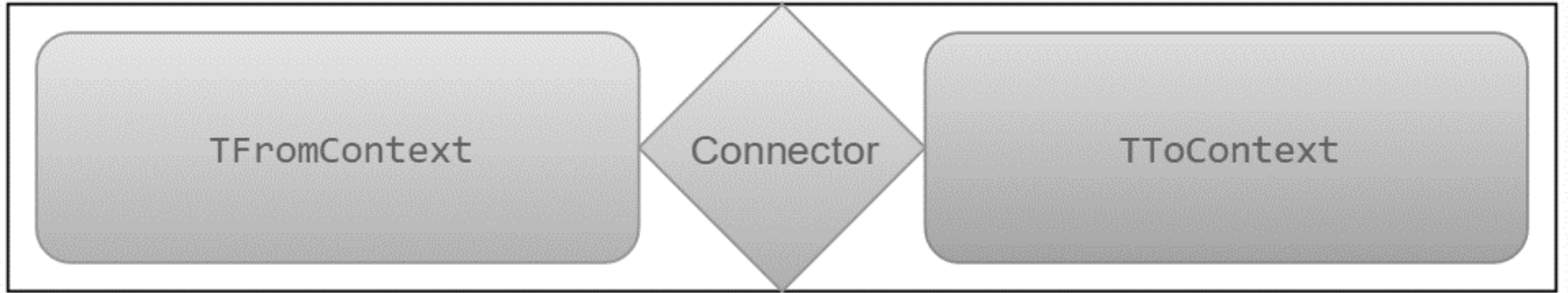
await Demo



Where to place links?

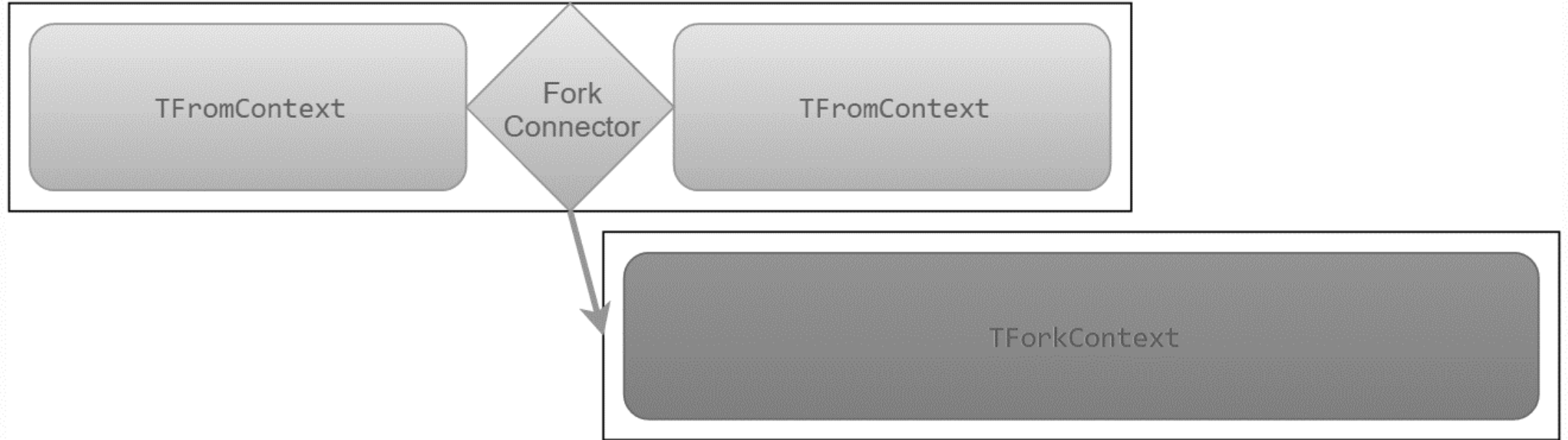


Stages

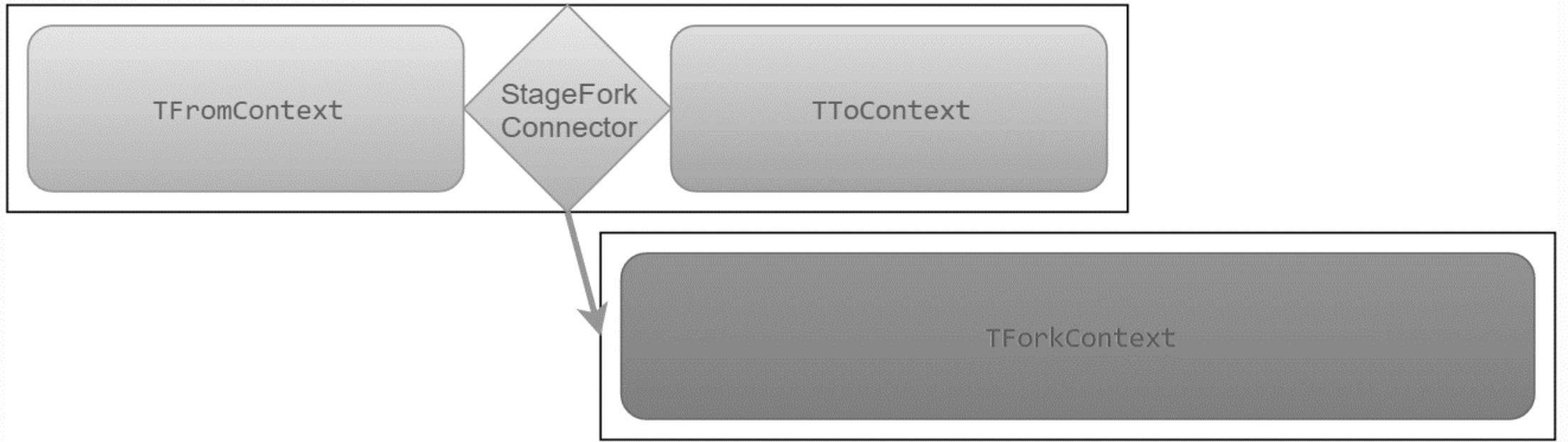


Stage Connector

await Demo



Fork Connector



Stage Fork Connector

Tree of Responsibility

Keep calm and let your head explode



Pattern

Build It

WrapUp

NSB v6

Will be **Async** all the way

Uses the **chain of responsibility** pattern heavily

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

docs.particular.net/nservicebus/pipeline/customizing-v6

Recap

reminder

Chain of Responsibility or Russian Dolls
is a **flexensible** pattern ideally suited
to build **robust IO bound pipelines**

The pattern is used in **many OSS**
projects

Know it, learn it, love it *

Thanks

Slides, Links...

github.com/danielmarbach/async-dolls

await Q & A



Software Engineer
Enthusiastic Software Engineer
Microsoft MVP

@danielmarbach
particular.net/blog
planetgeek.ch

Thanks