ASP.NET Core 6: A Workshop

Roland Guijt
Freelance .NET consultant and trainer

@rolandguijt roland.guijt@gmail.com

A Big Picture

Outcome

What ASP.NET Core is

What you can do with it

Theoretical and technical

Choose what to learn next

Starting Point

Web basics

HTML and CSS knowledge

.NET and C#

https://github.com/RolandGuijt/aspnetworkshop

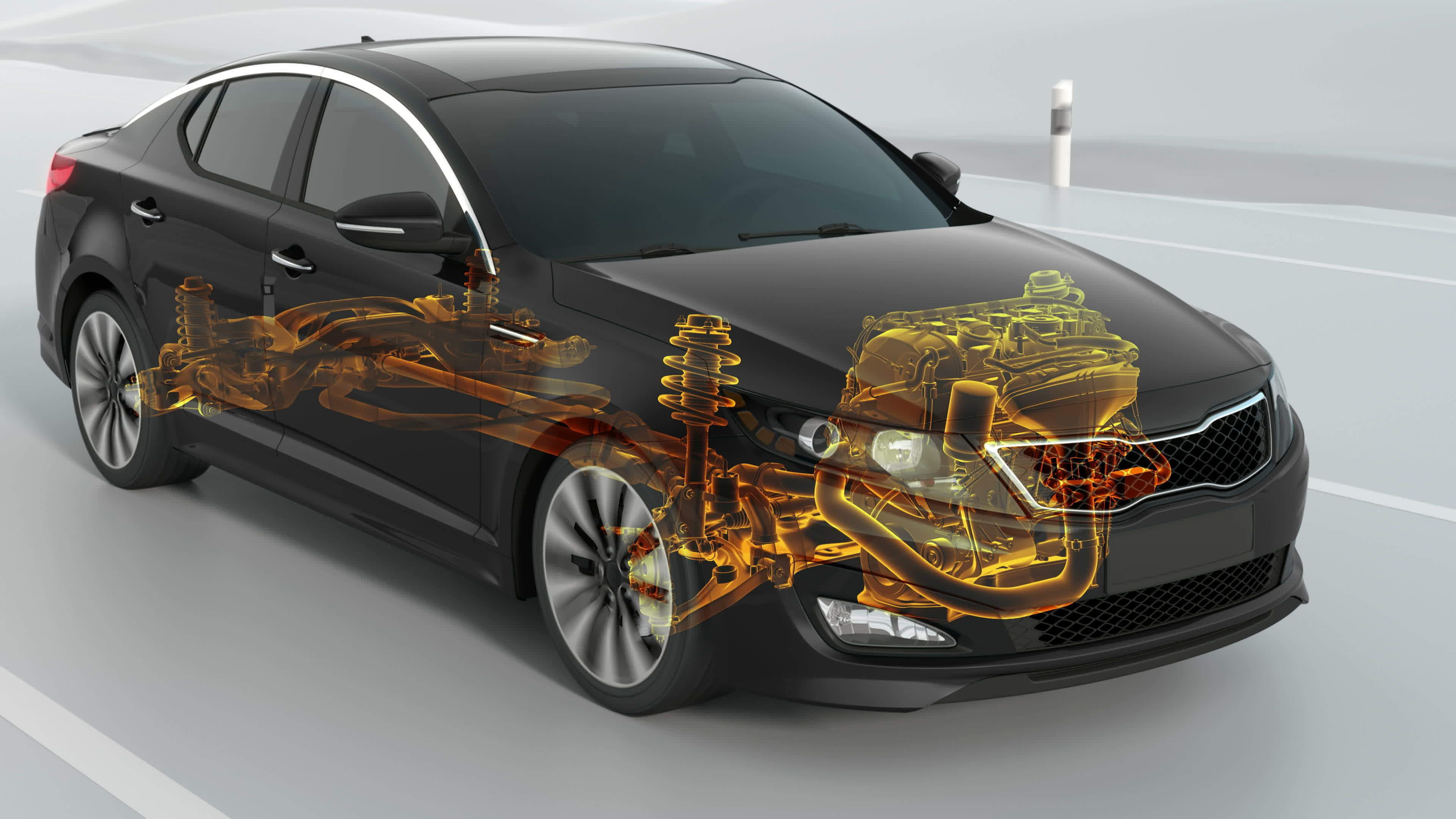
Web site

Web application

Web Application Framework

Building blocks

Structure



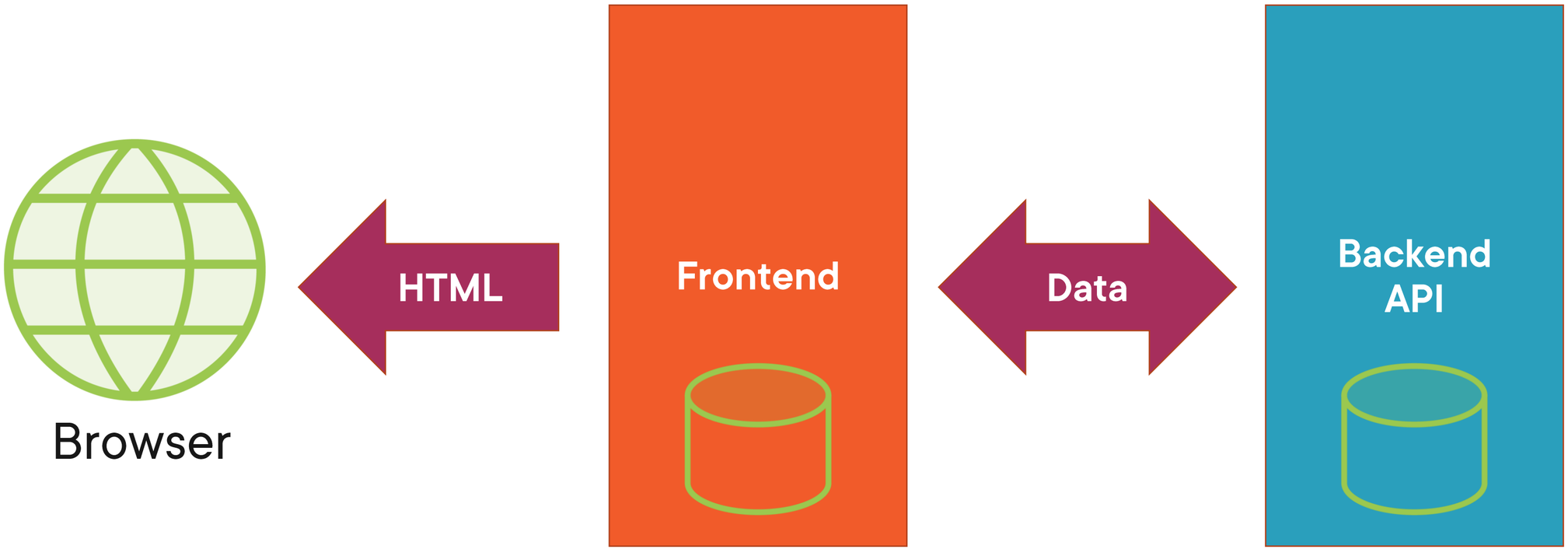
ASP.NET Core is a
web application framework

ASP.NET Core: Application Types

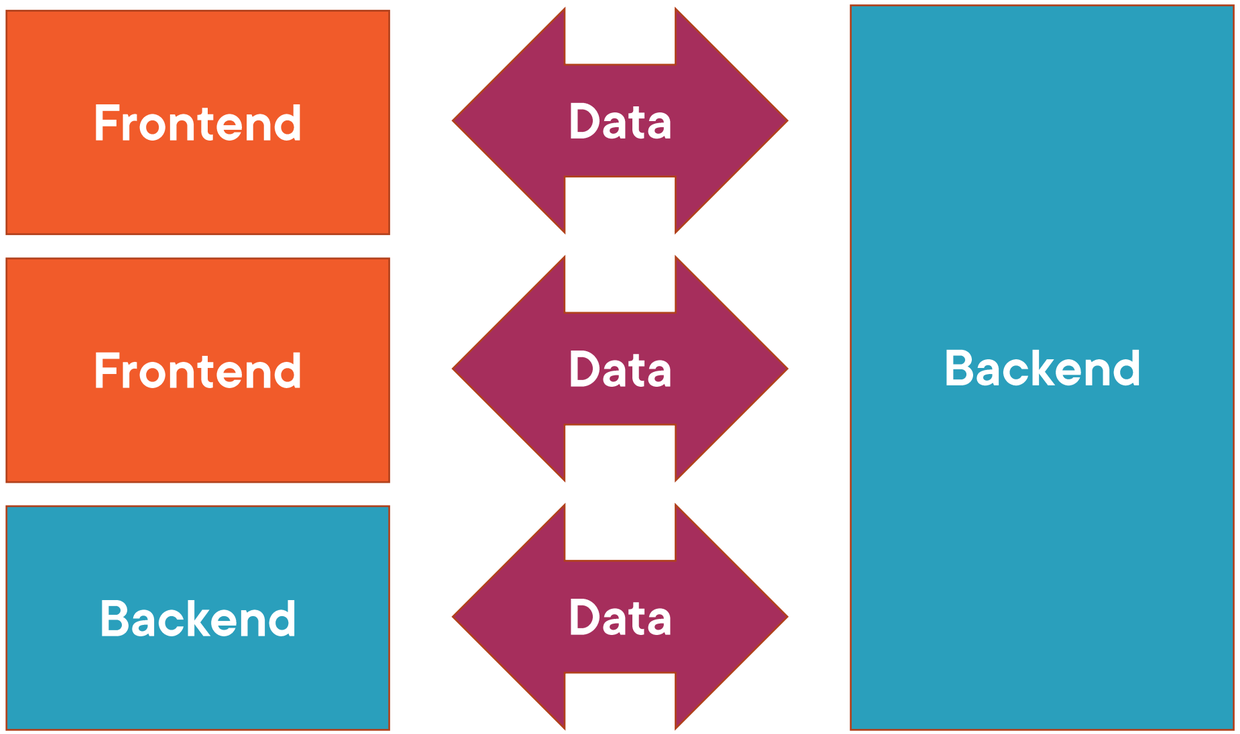
Frontend

Backend

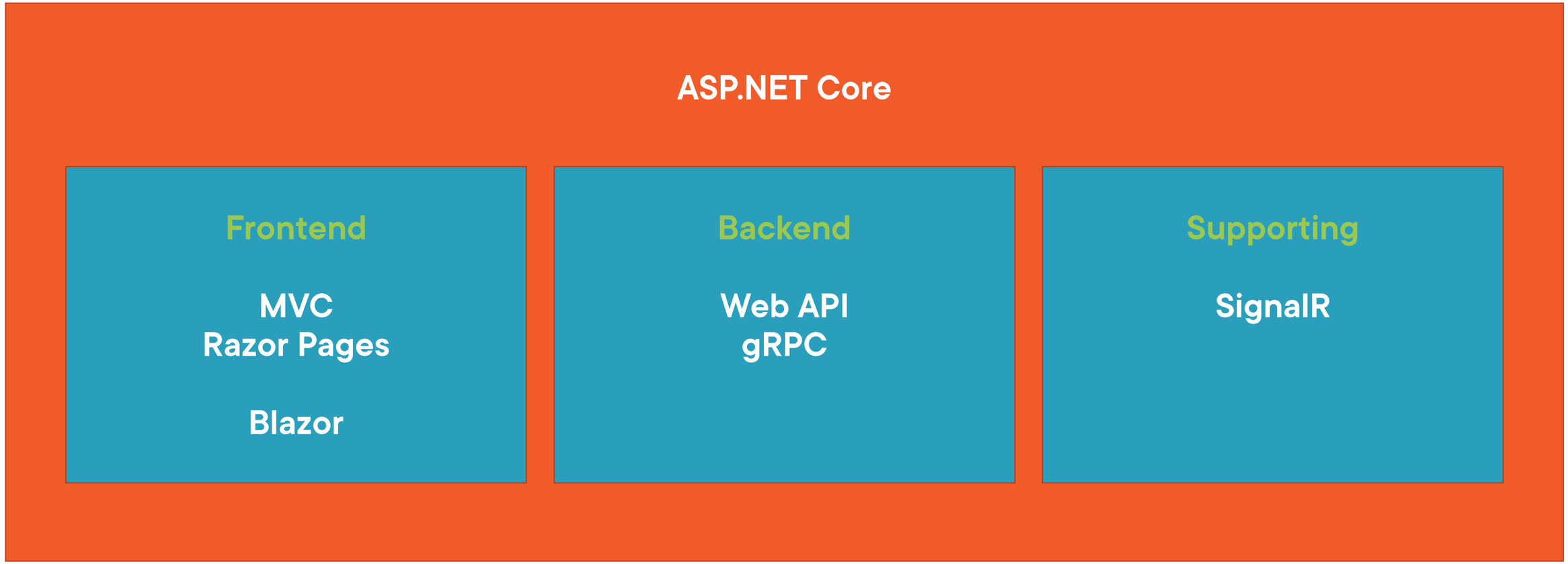
Frontend and Backend



The Centralized Backend



What To Build With ASP.NET Core?



ASP.NET Core

Application styles

Foundation

Open source

High performance

Cross platform

Modular

.NET and C#, F#, VB

Tools

Visual Studio

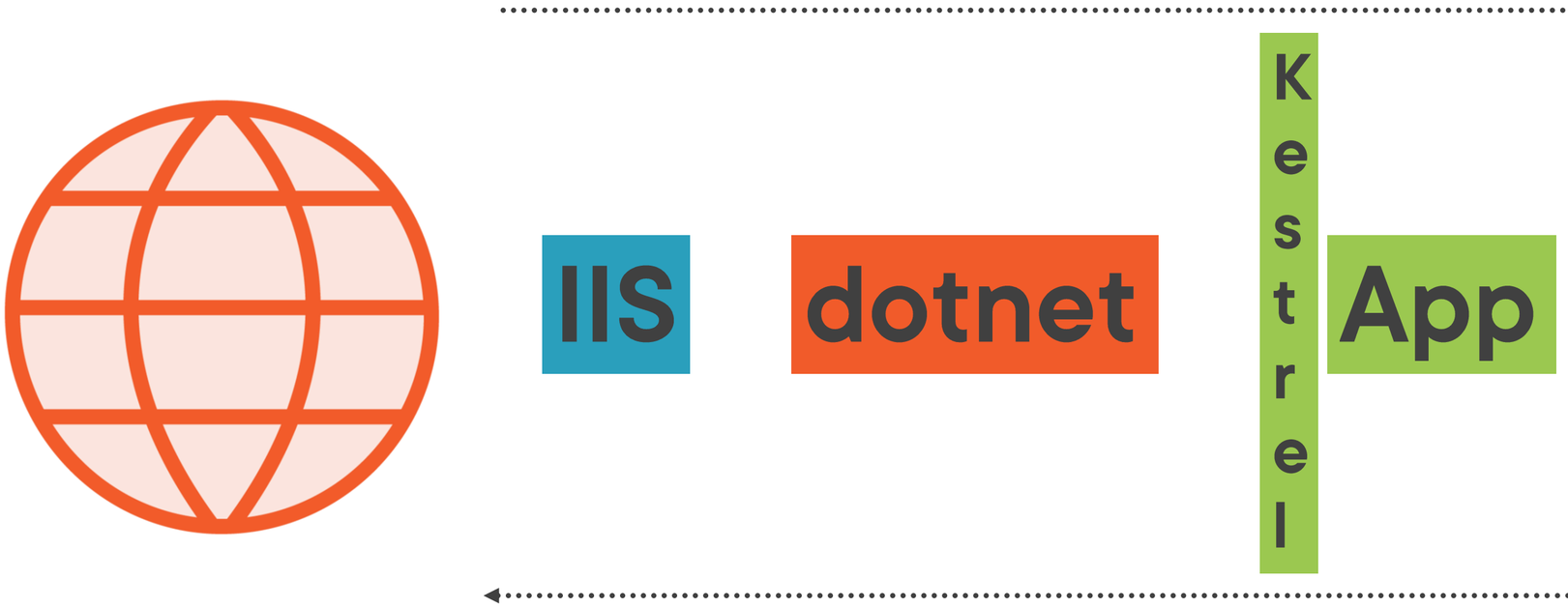
Rider

Visual Studio Code

CLI

The Built-In Kestrel Web Server

IIS In Front Of Kestrel



Dependency Injection

Manages the lifetime of objects

Register

Instance

Object Lifetimes

Singleton

Transient

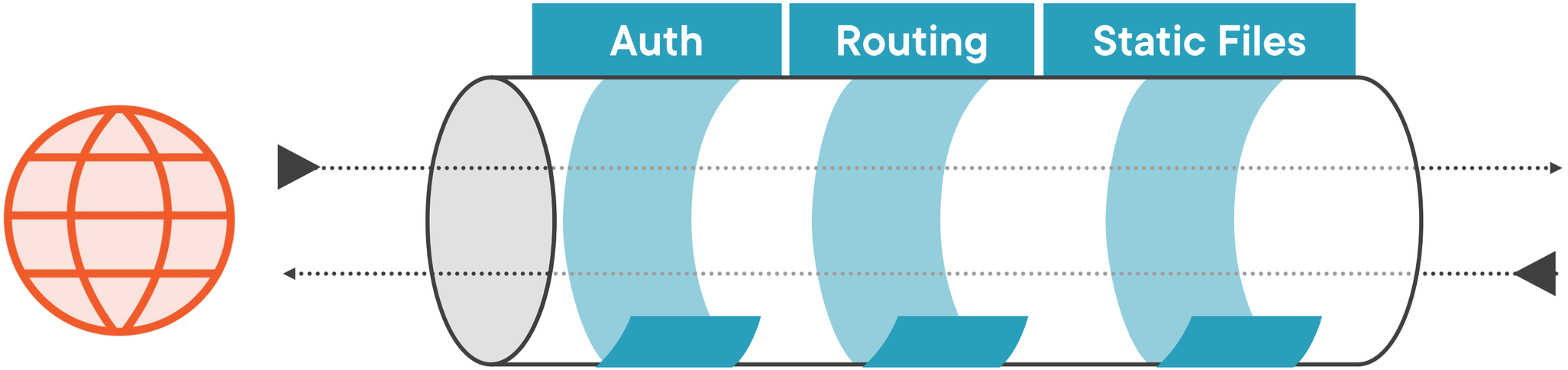
Scoped

Dependency Injection: Advantages

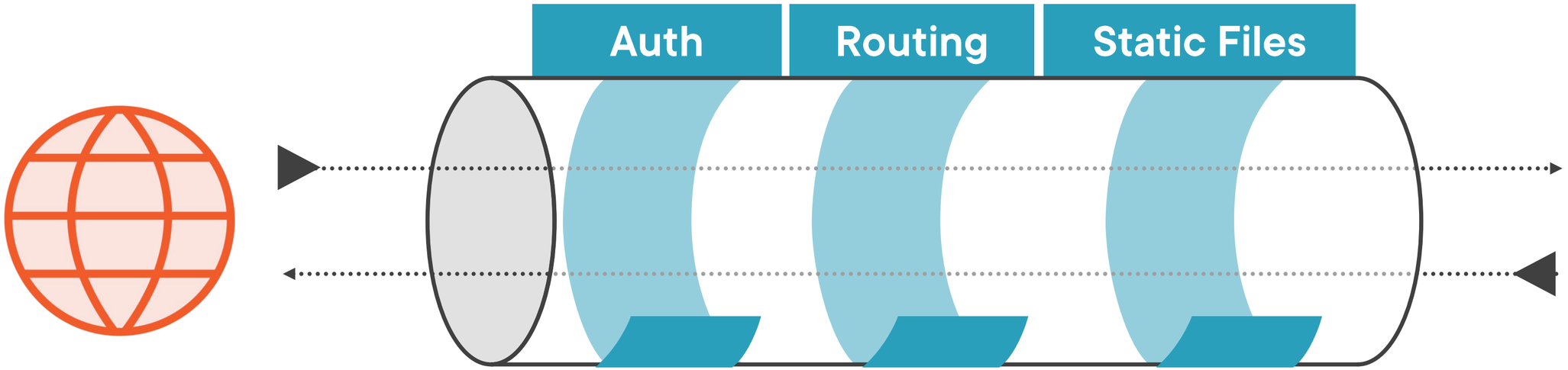
No complexities around the lifetime of objects

No "hard" dependencies between types

The Request Pipeline

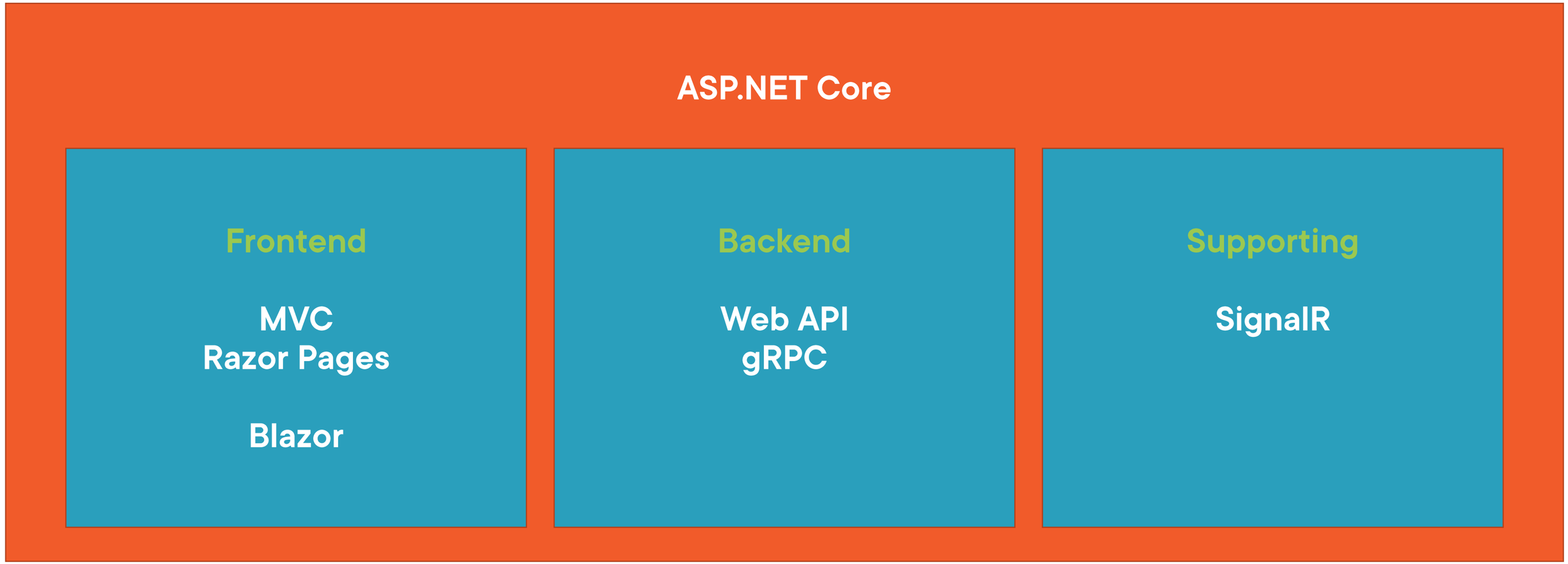


The Request Pipeline

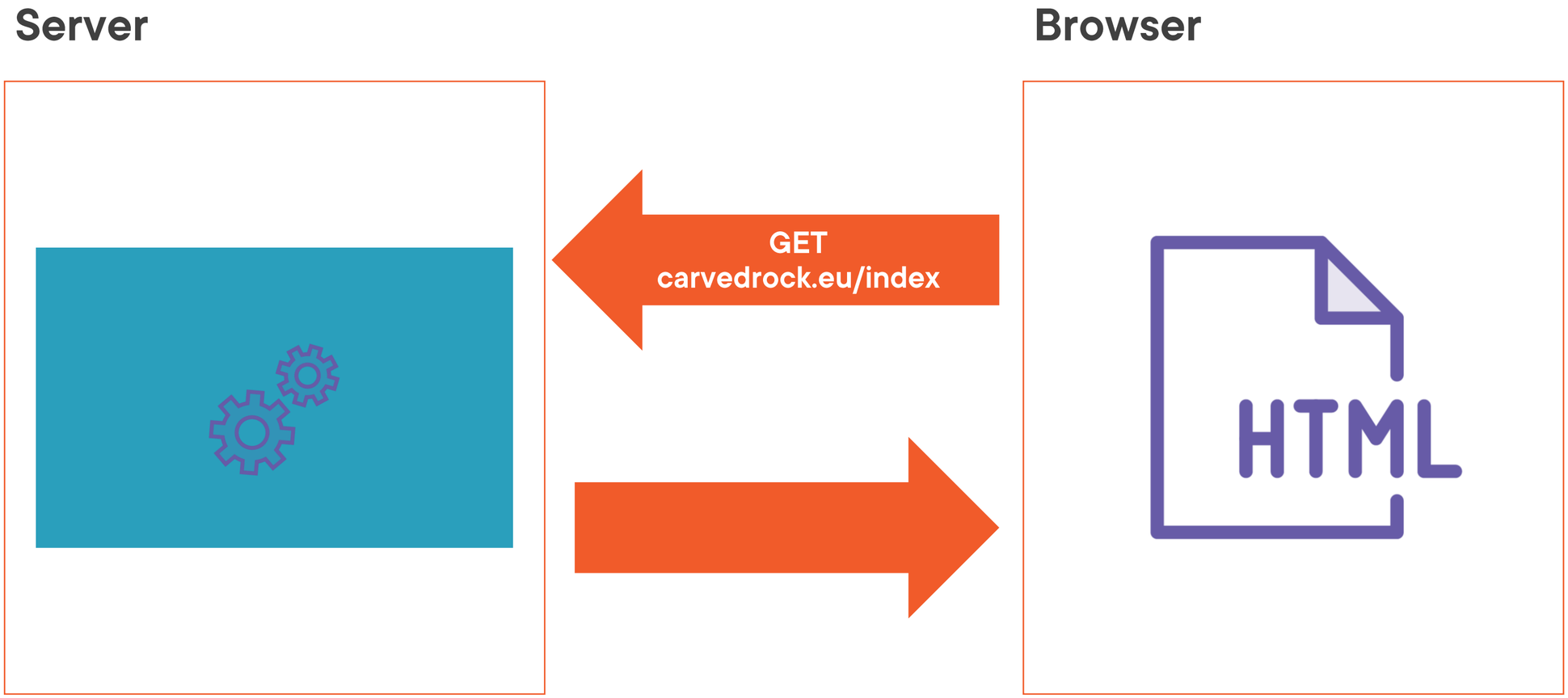


Razor is HTML with C#

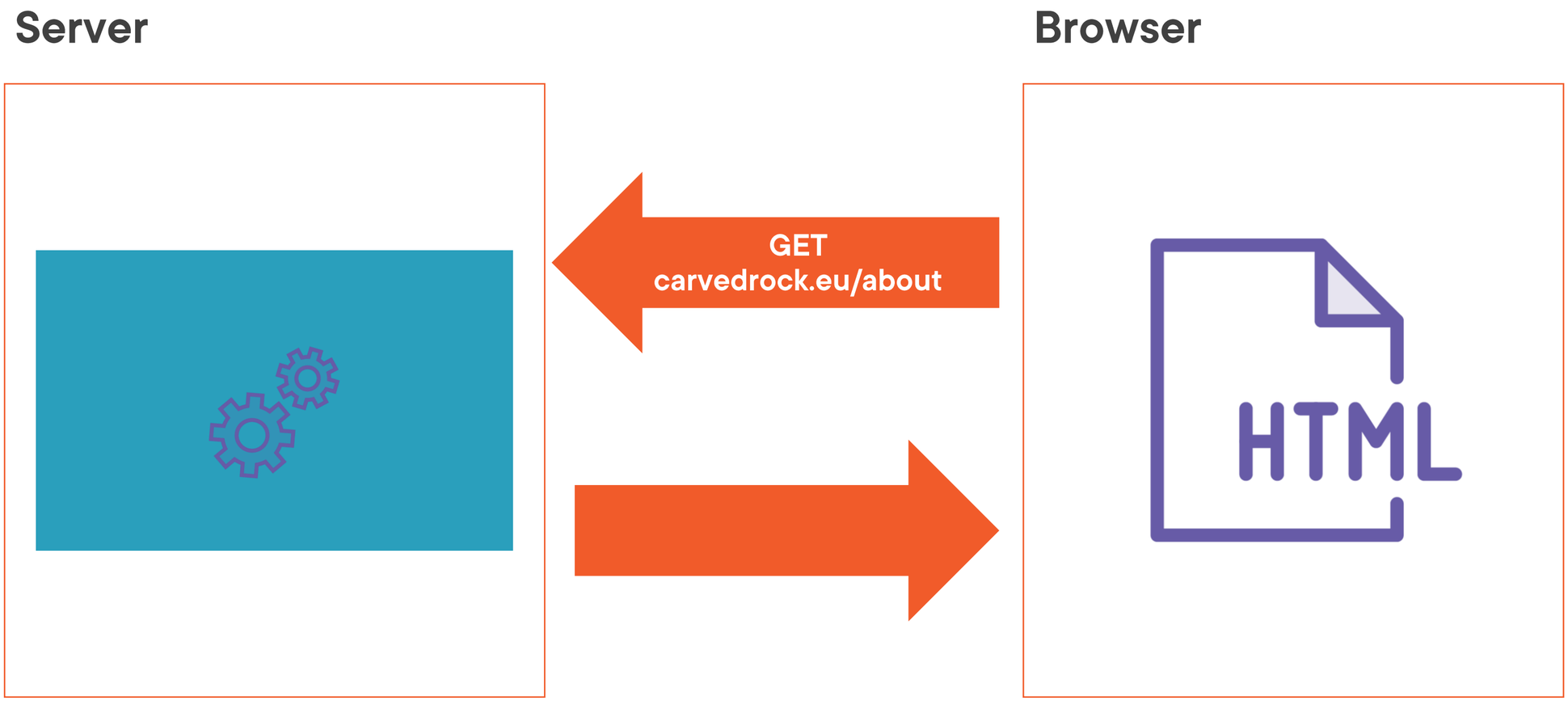
What to Build with ASP.NET Core?



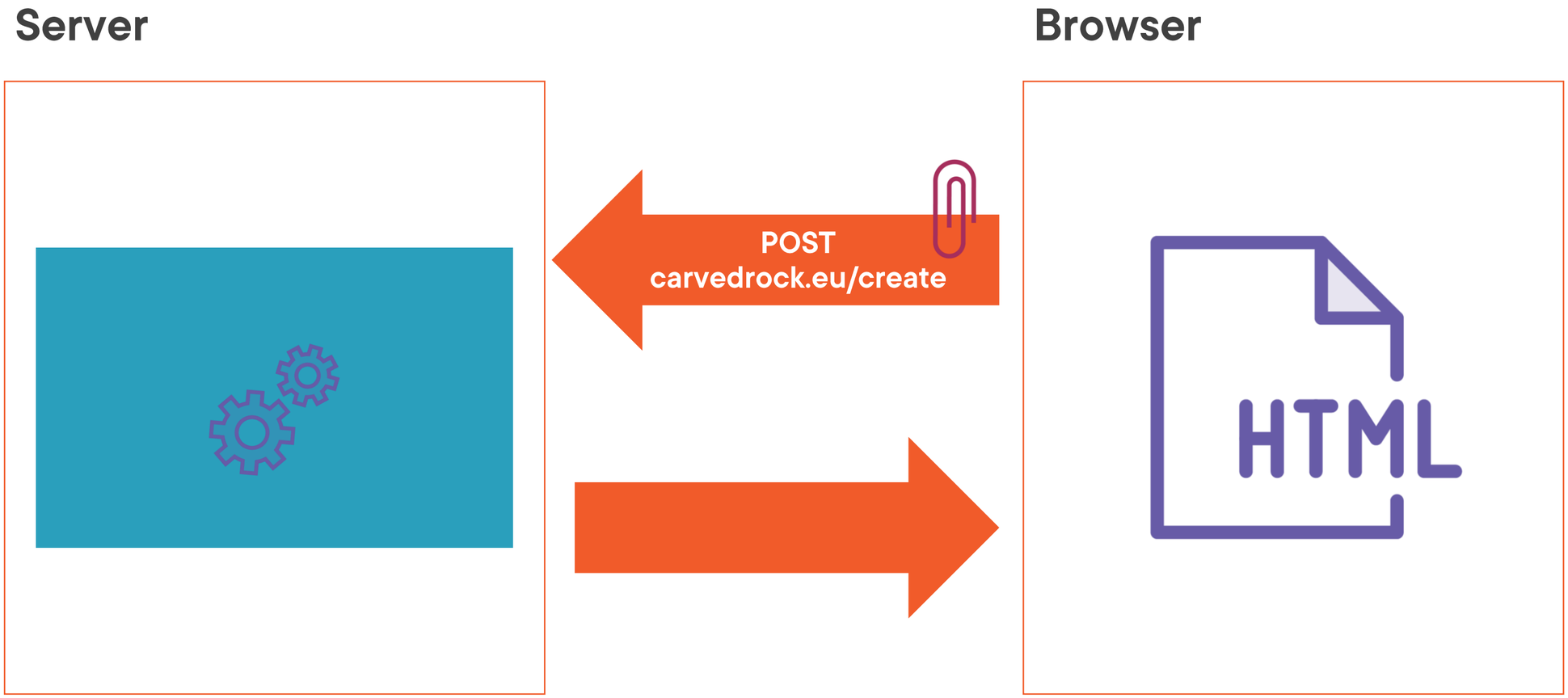
Server-rendered Applications



Server-rendered Applications



Server-rendered Applications

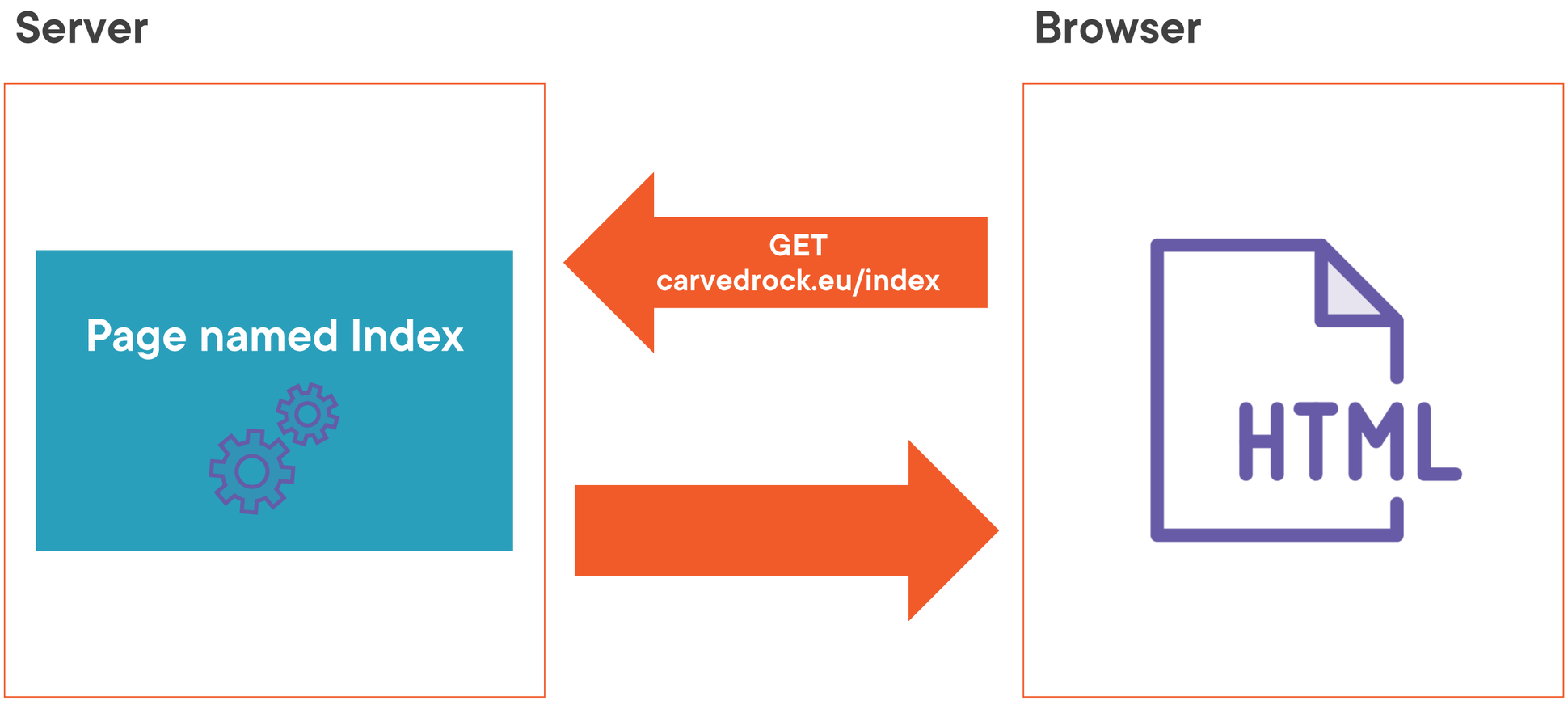


Server-rendered Application Patterns

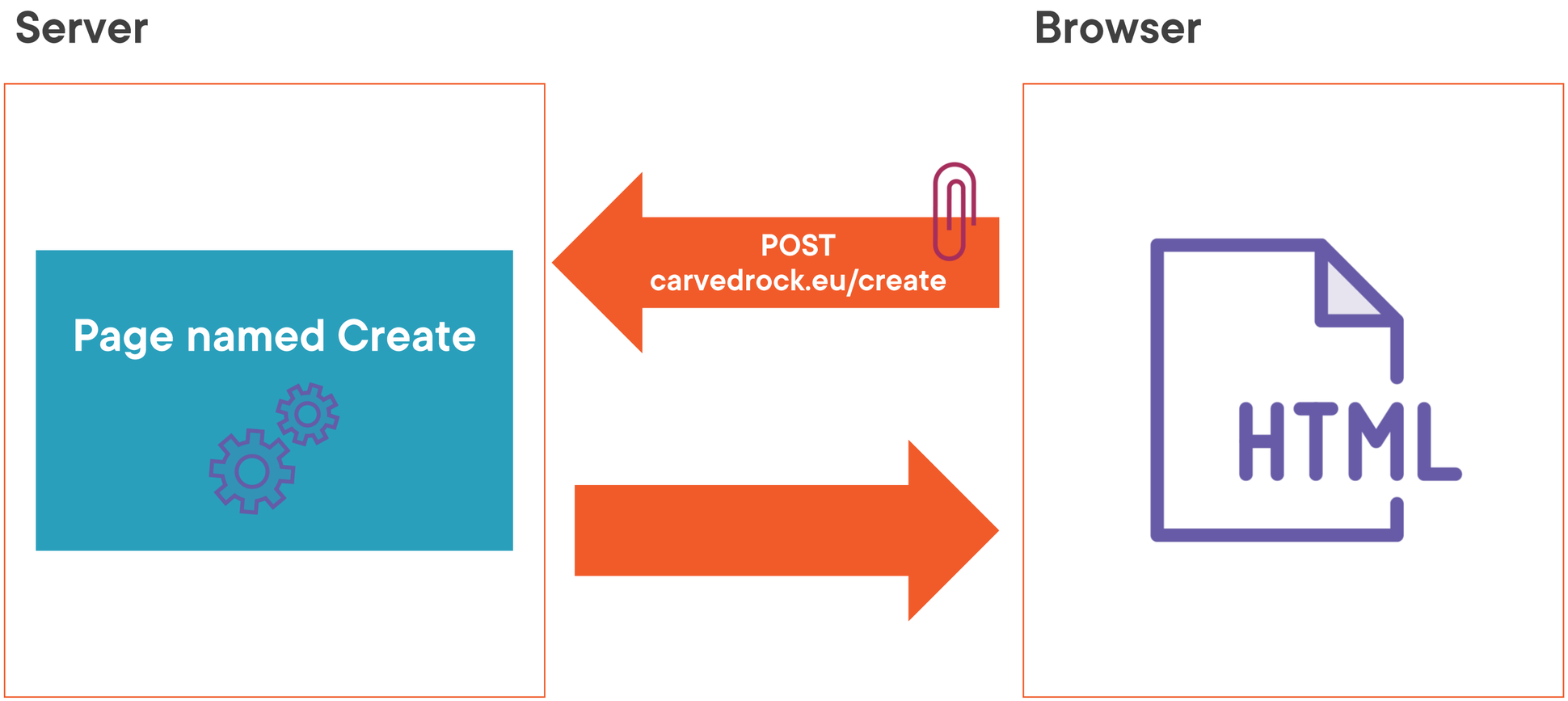
Razor Pages

MVC

Razor Pages


Server
Browser
Page named Index
GET
carvedrock.eu/index
HTML

Razor Pages


Server
Browser
Page named Create
POST
carvedrock.eu/create
HTML

Data Binding

Server

```
[BindProperty]
Product NewProduct { get; set; }
```

POST
carvedrock.eu/create

Browser

```
<input
name="NewProduct.Name"
value="Backpack" />

<input
name="NewProduct.Price"
value="100" />
```

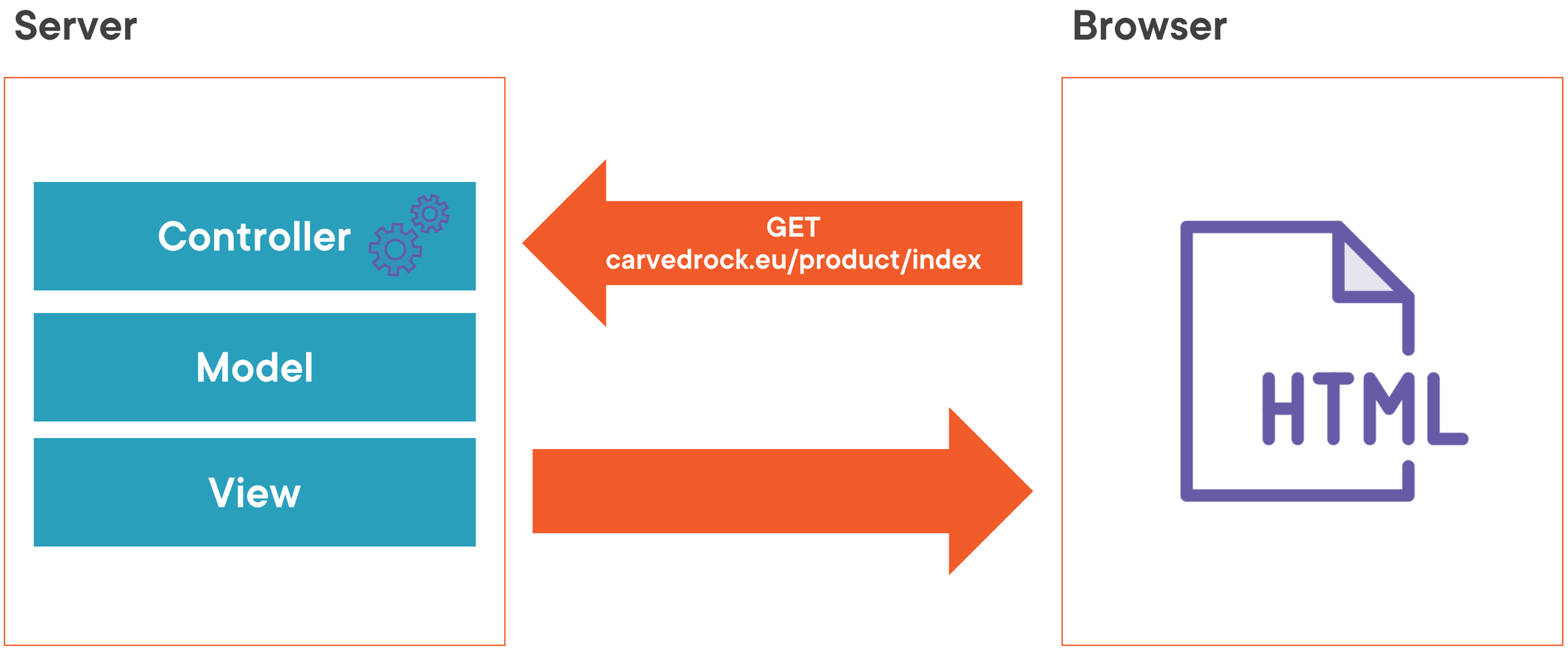
MVC

Model

View

Controller

MVC


Server
Controller
Model
View
GET
carvedrock.eu/product/index
Browser
HTML

MVC vs. Razor Pages

Complexity

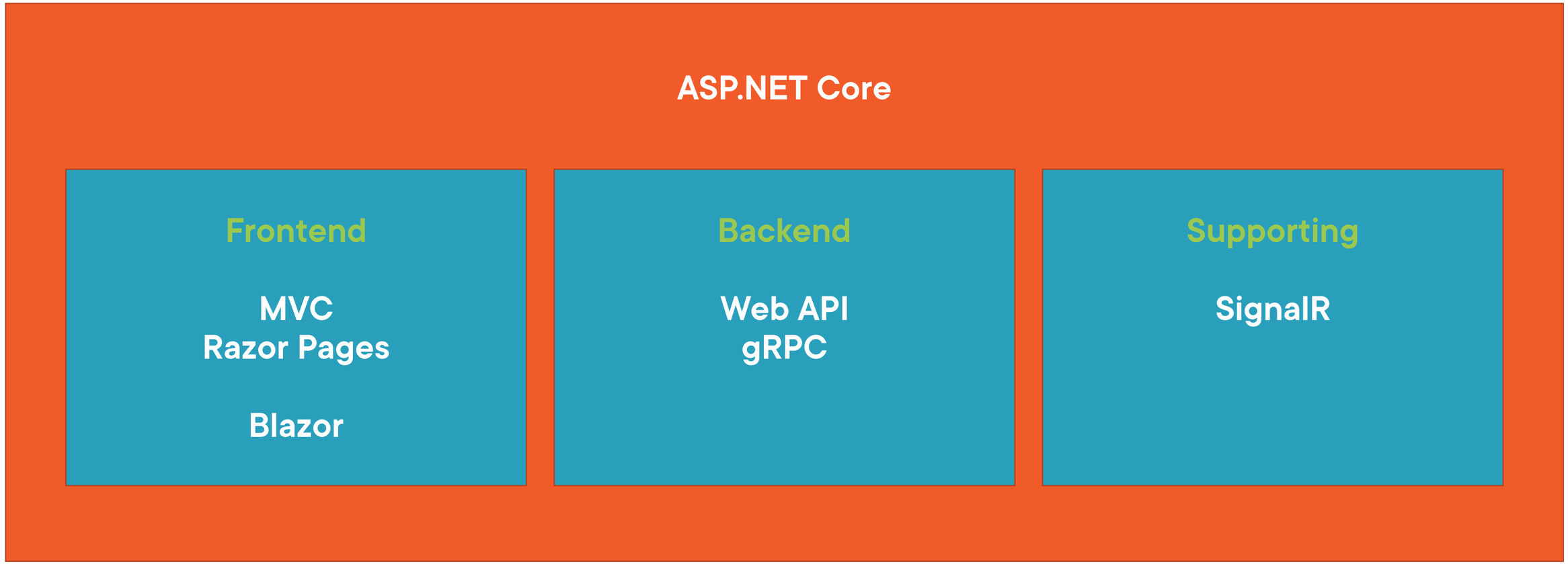
Routing

Separation

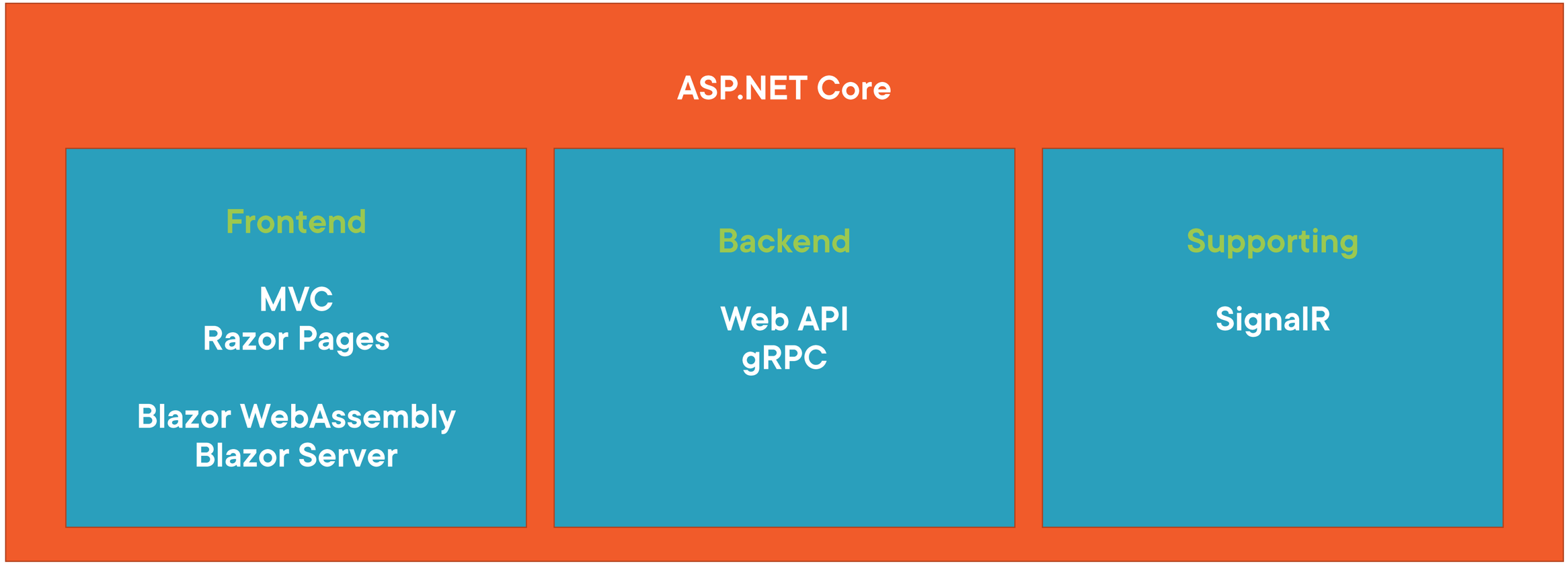
Changeability

Reusability

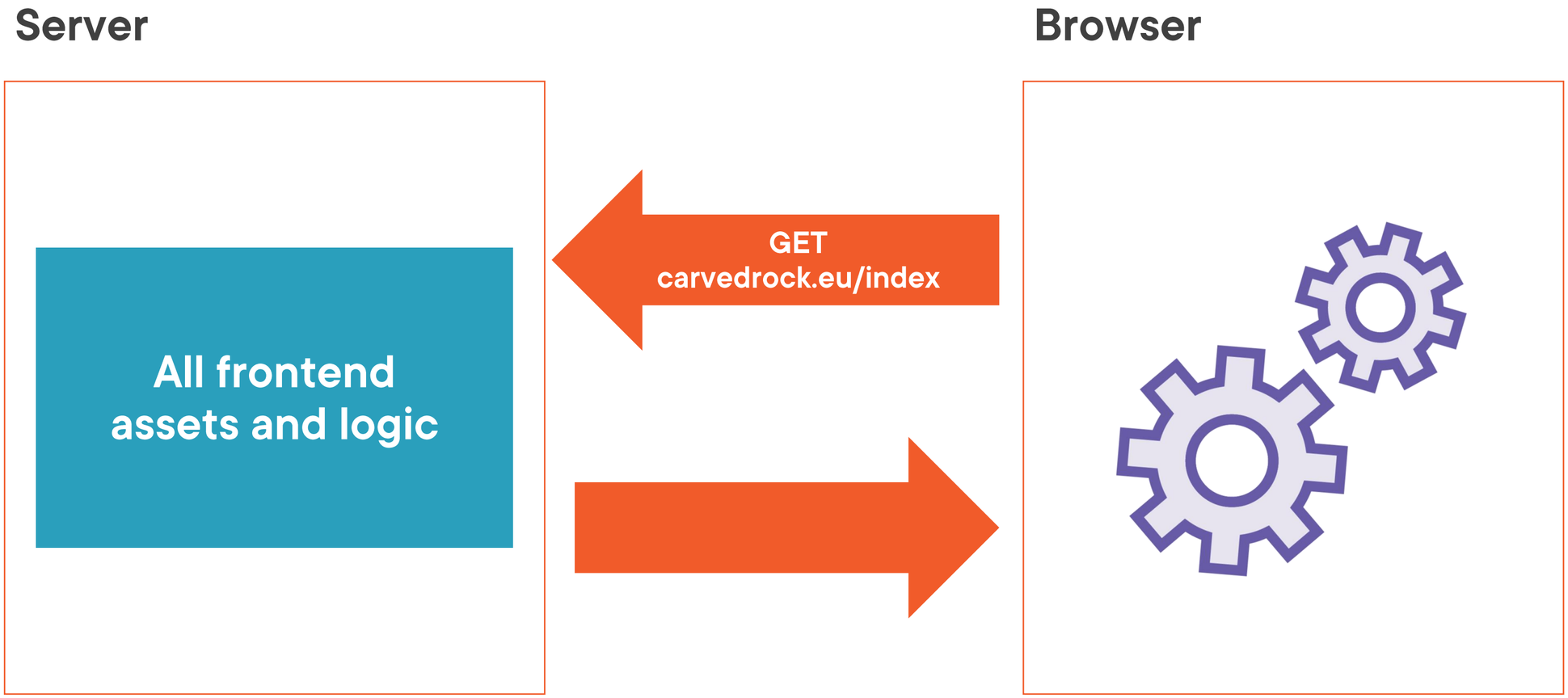
What To Build With ASP.NET Core?



What To Build With ASP.NET Core?



Client-rendered Applications



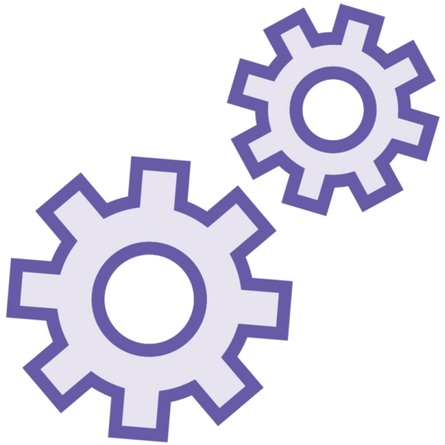
Client-rendered Applications

Server

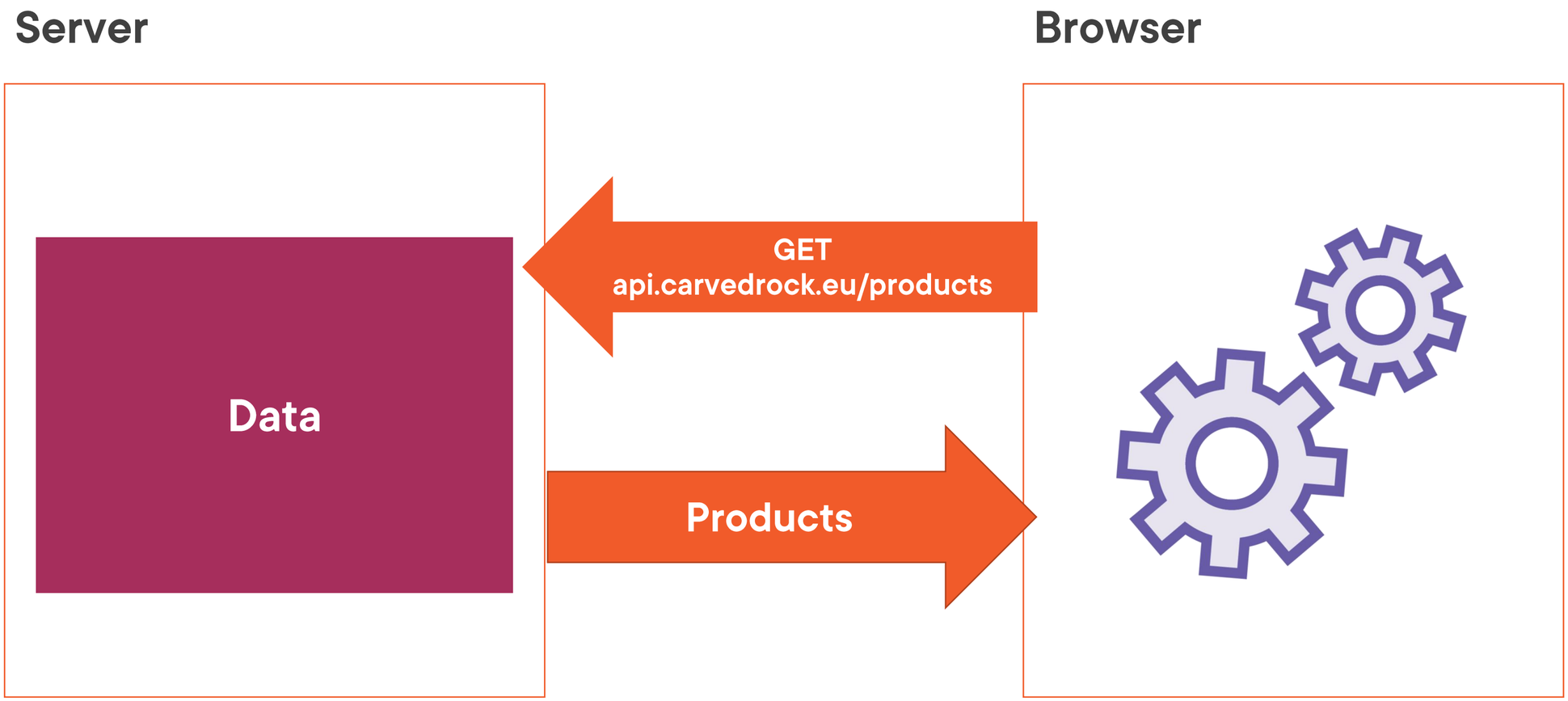
All frontend assets and logic

GET
carvedrock.eu/about

Browser

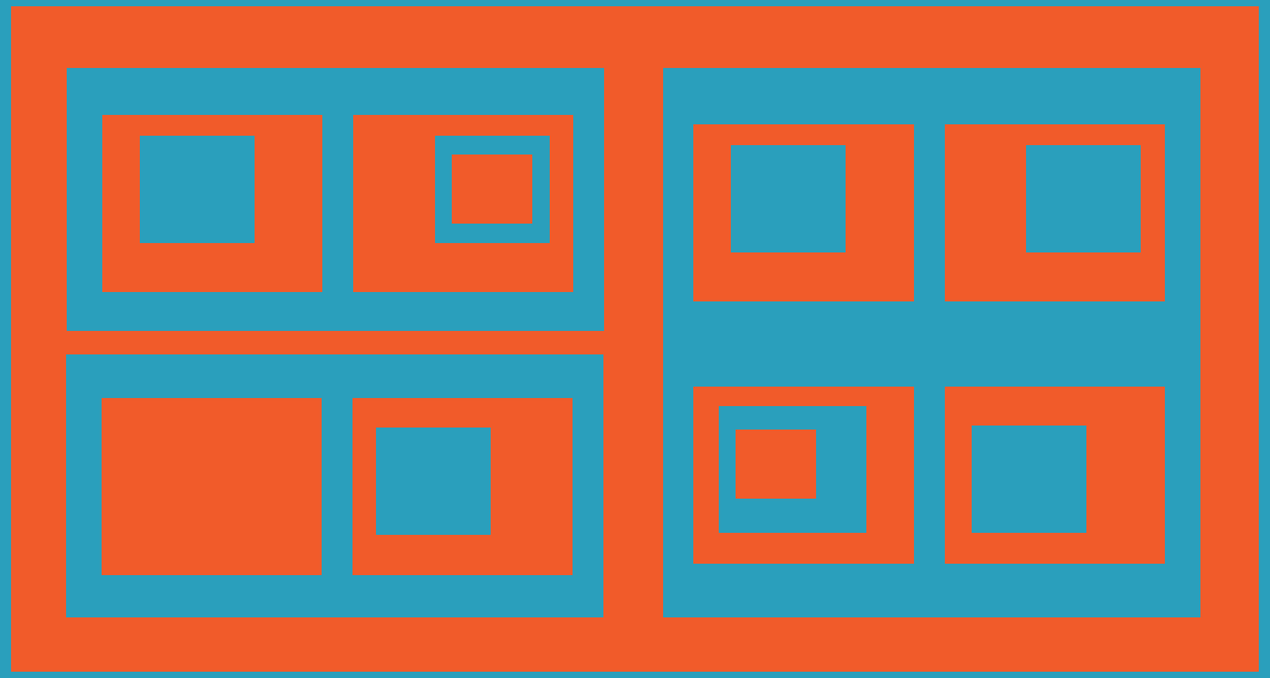


Client-rendered Applications



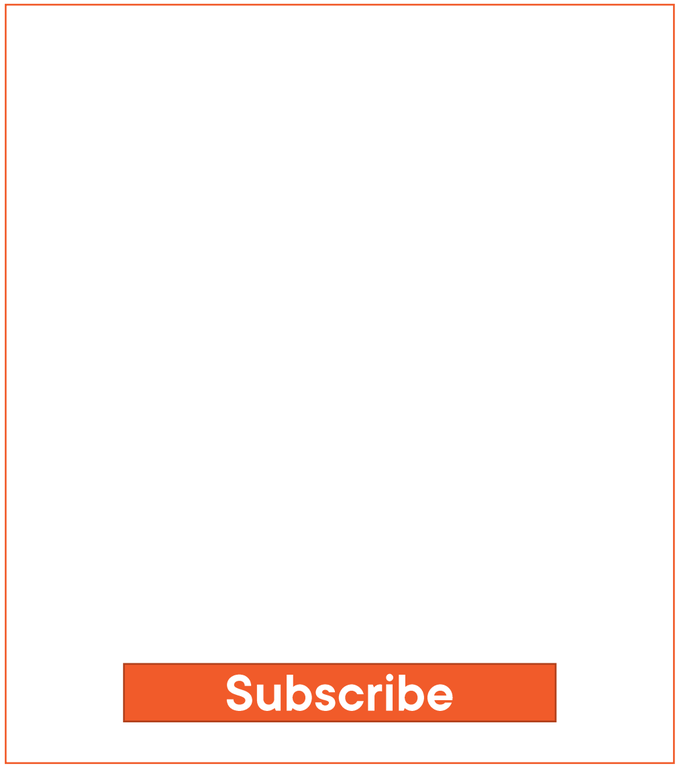
Blazor
A component-based
single-page application
framework using .NET and C#

Creating Blazor Applications == Writing components

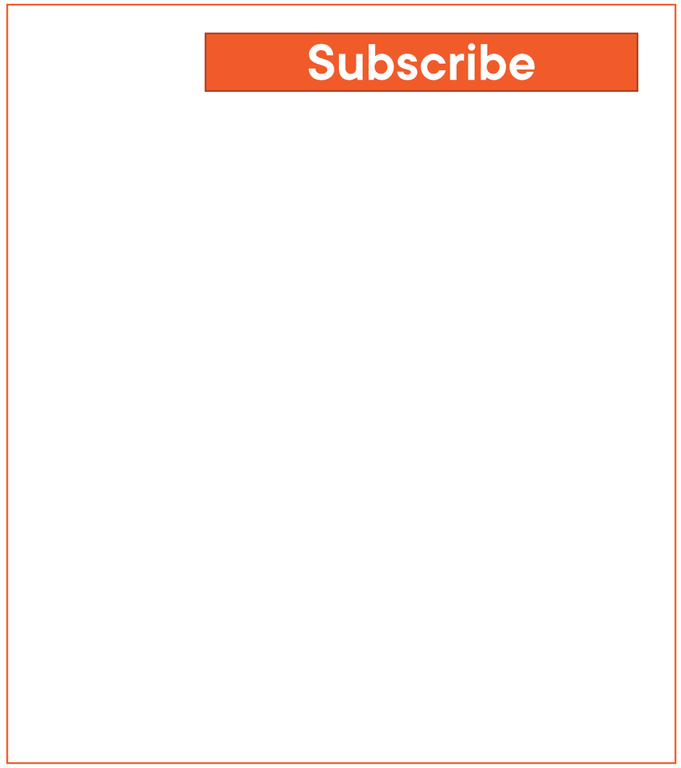


Components

Page: Index

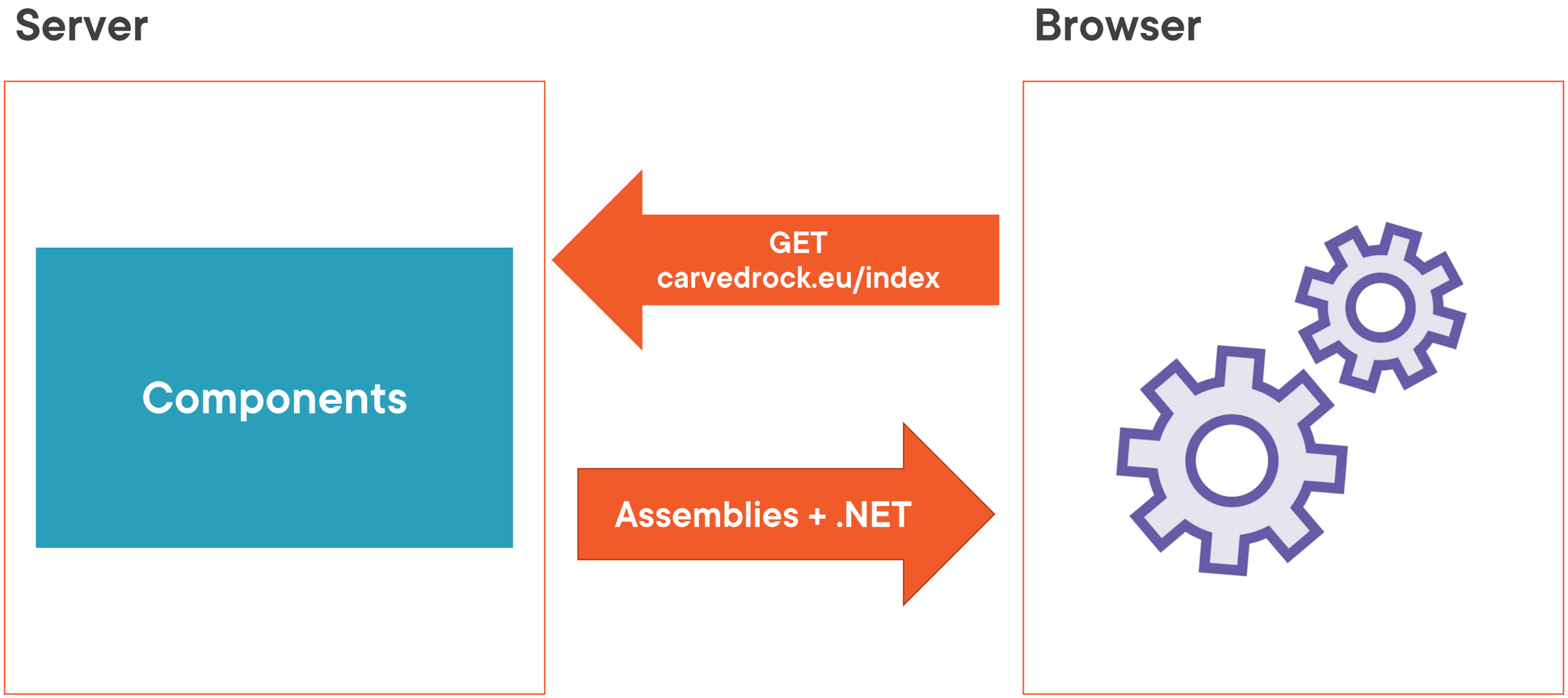


Page: About



Components can be reused

Blazor WebAssembly



WebAssembly

Language browsers can interpret

Lower level, binary

Code is compiled to WebAssembly

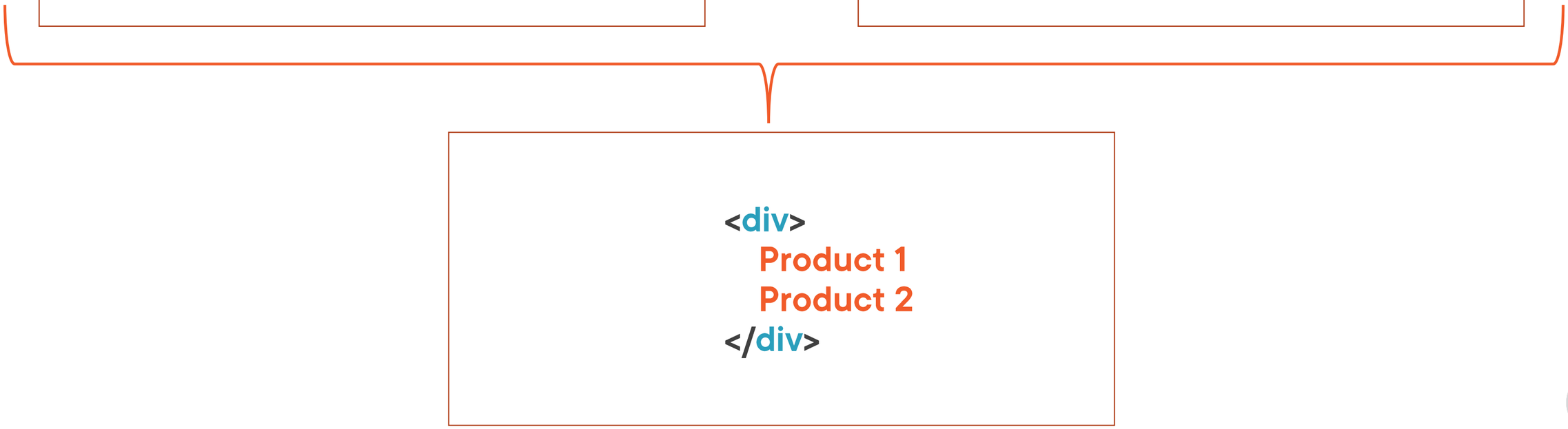
Updating the UI

Old

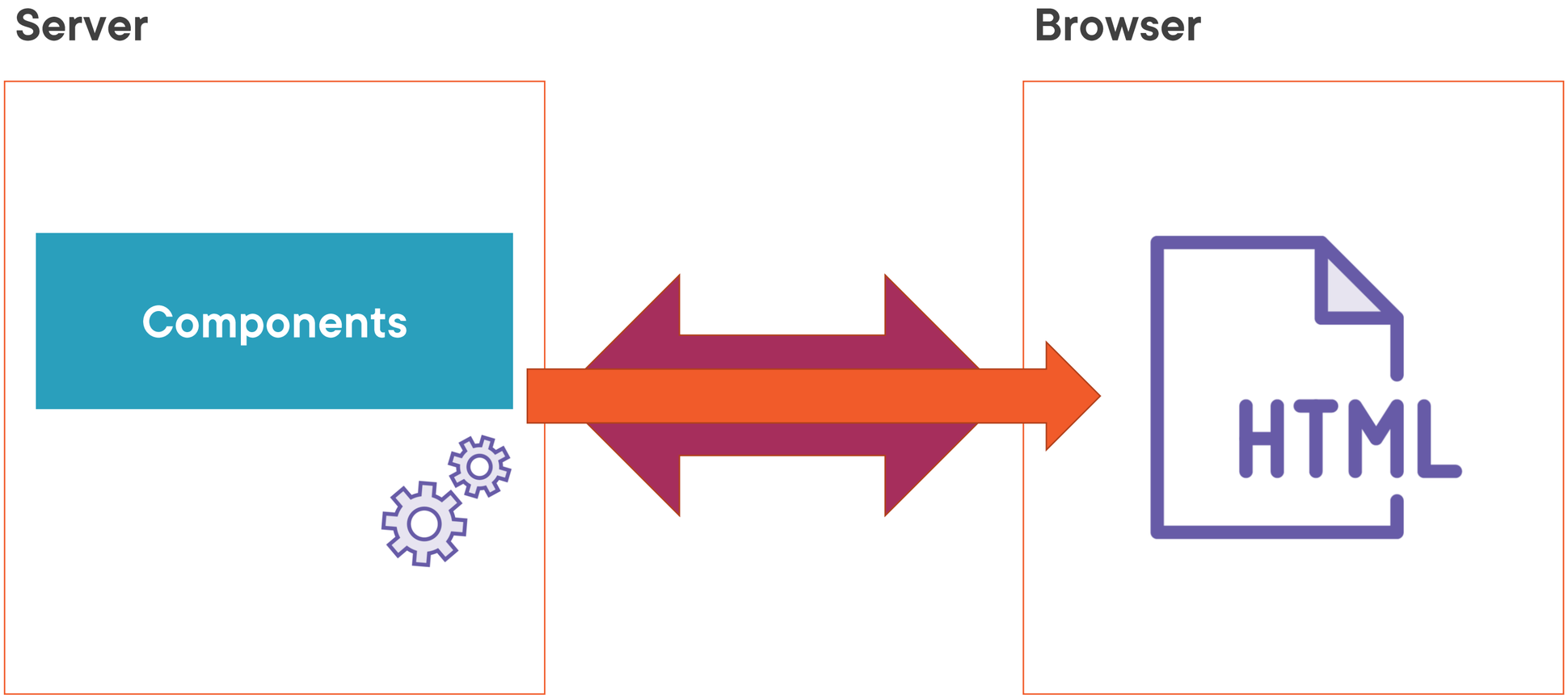
```
<div>
</div>
```

New

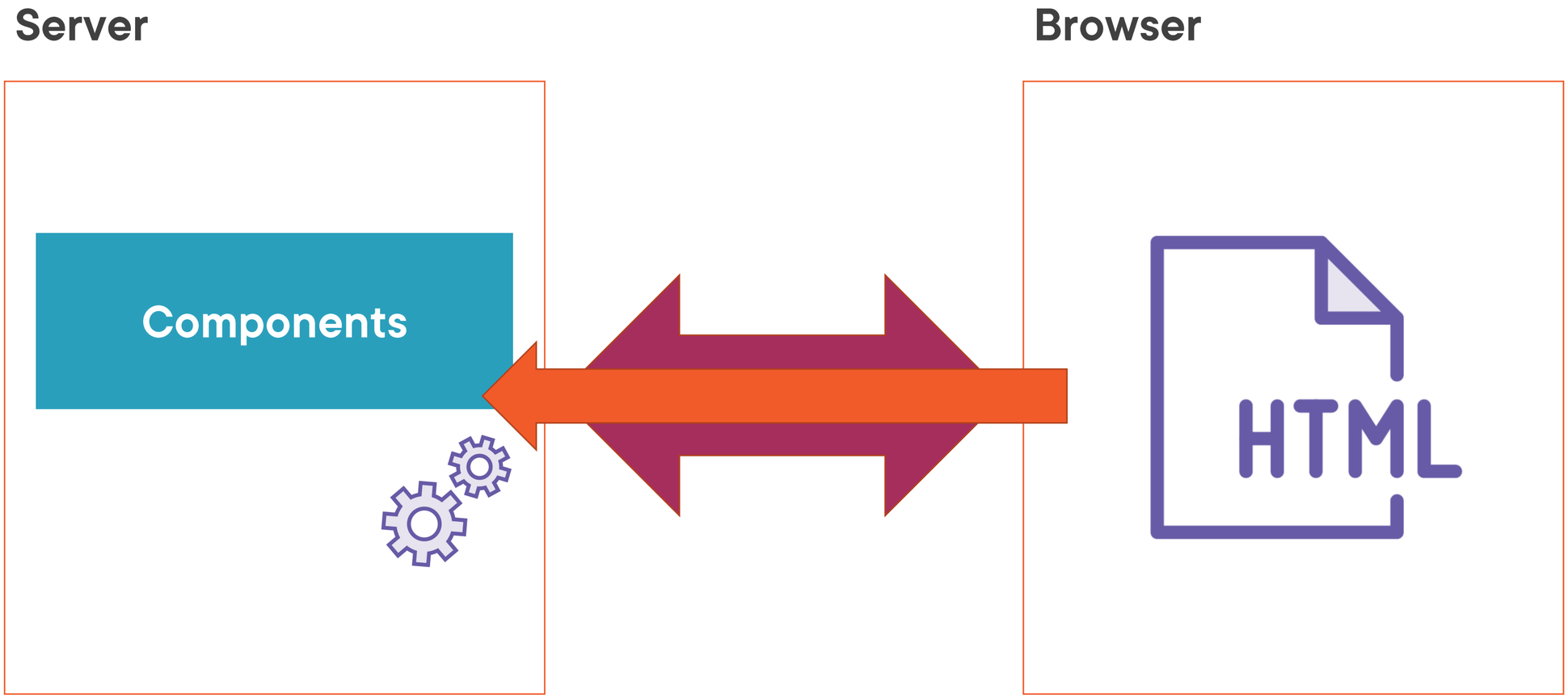
```
<div>
   Product 1
   Product 2
</div>
```



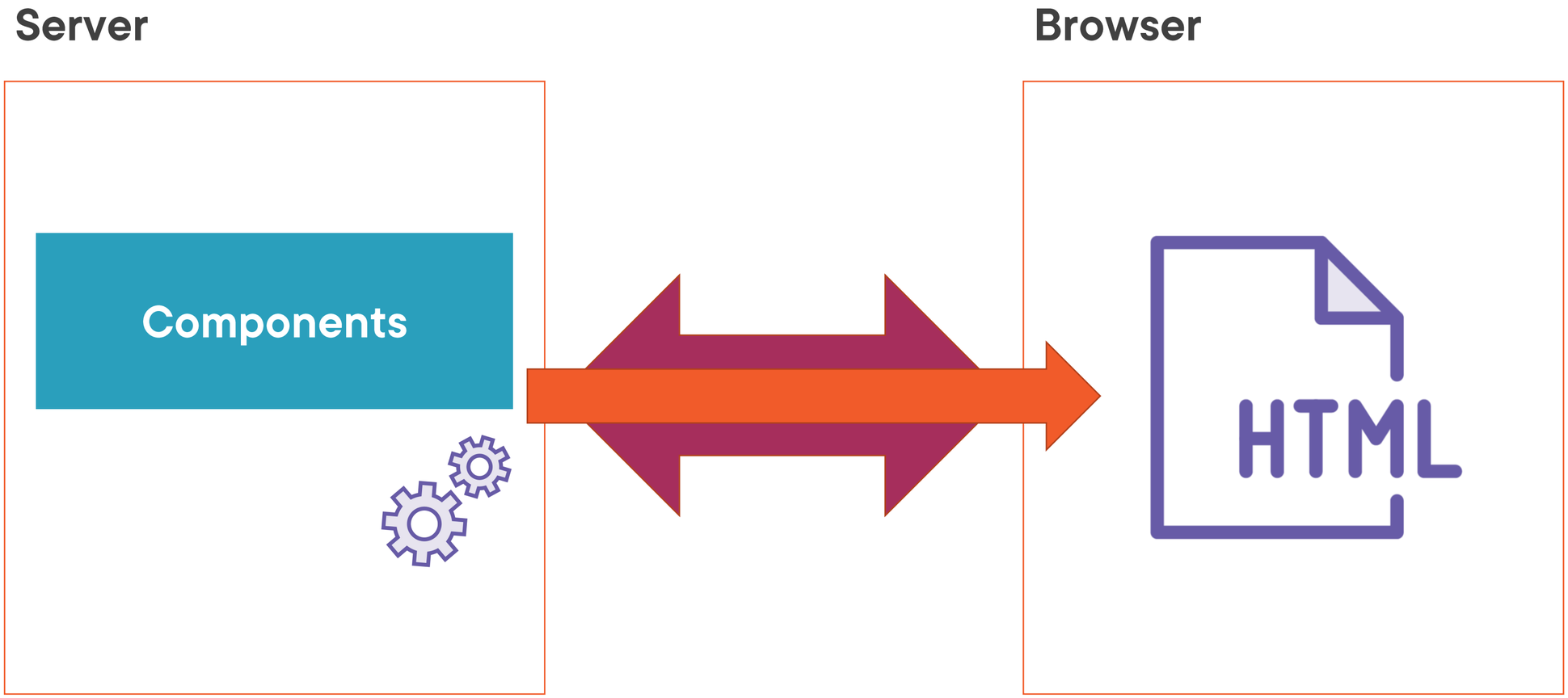
Blazor Server



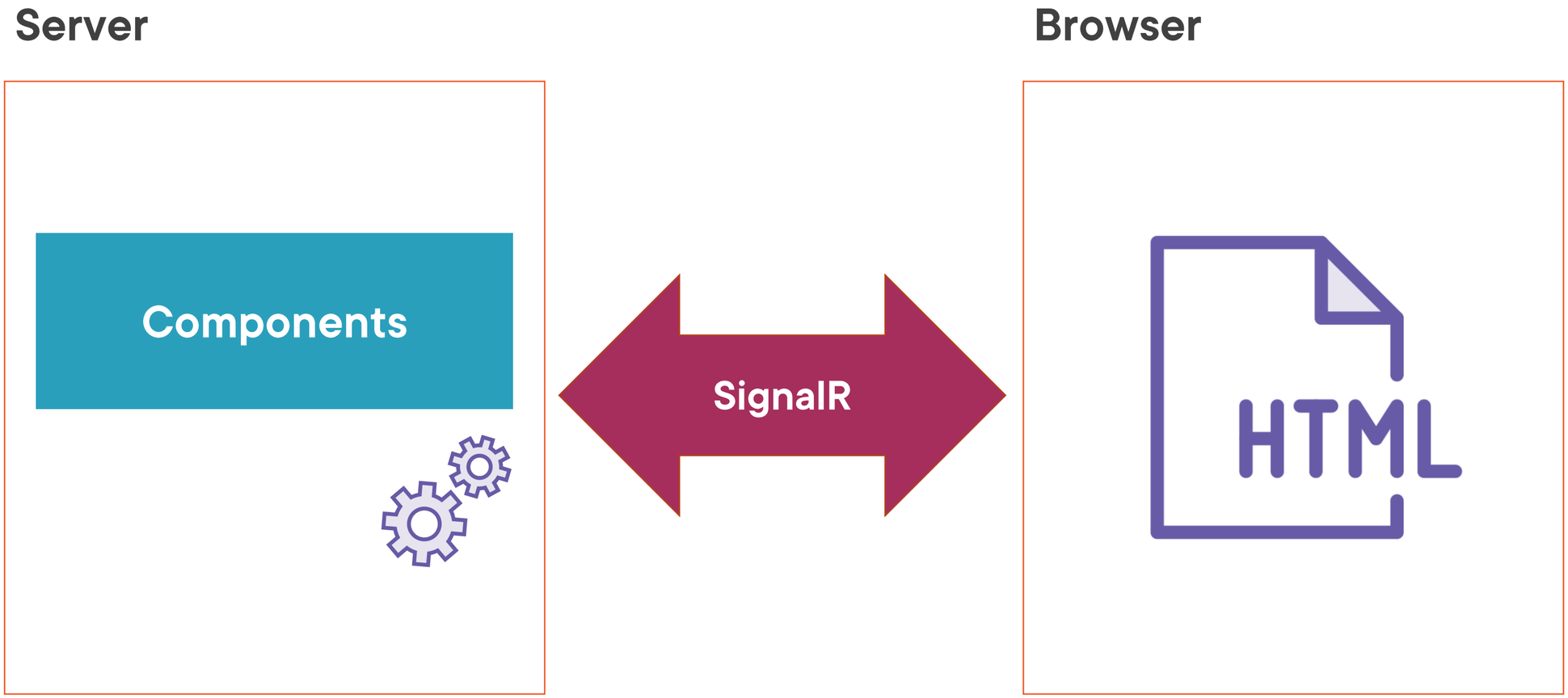
Blazor Server



Blazor Server



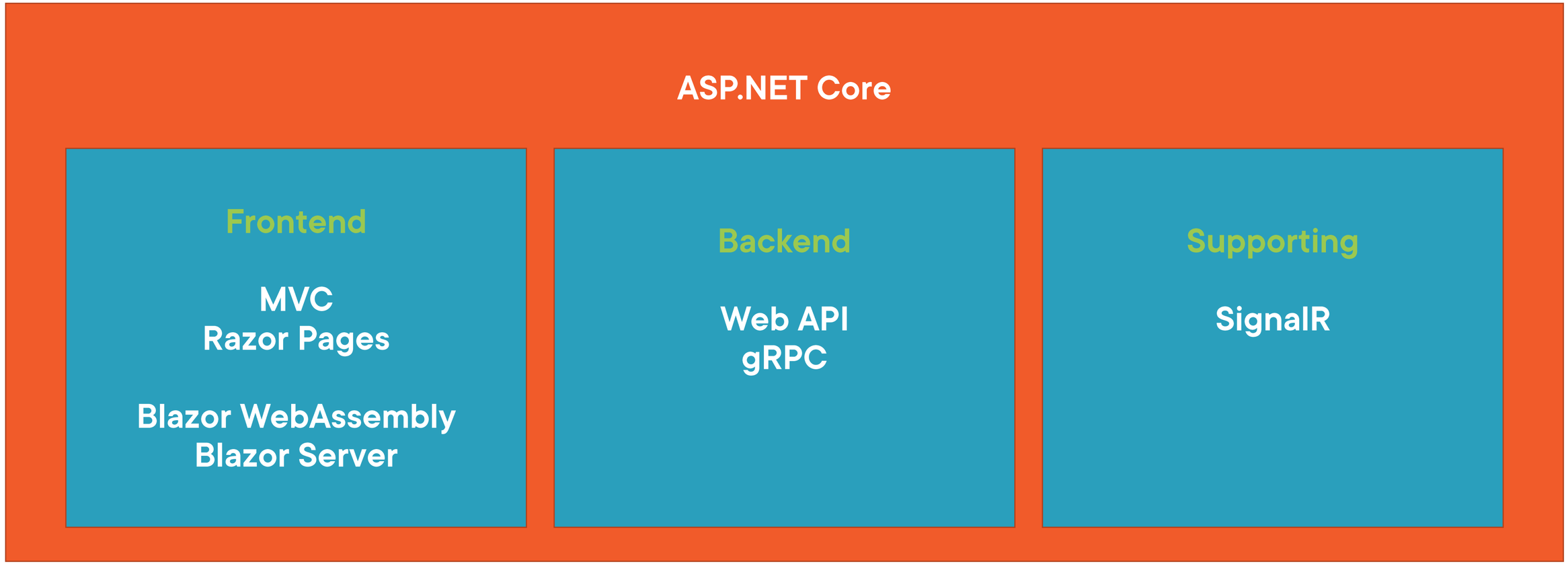
Blazor Server



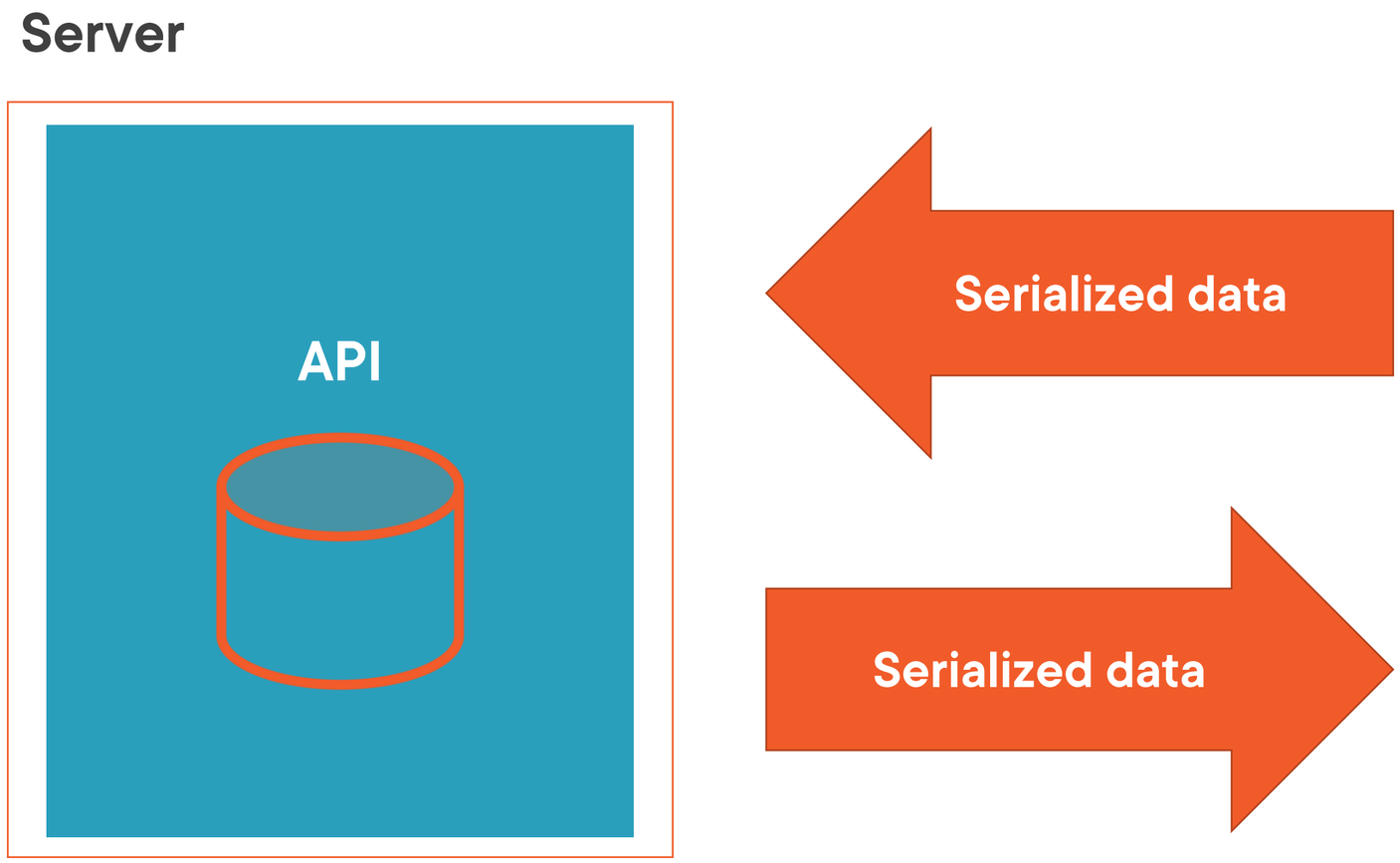
Blazor WebAssembly vs. Blazor Server

Blazor WebAssembly	Blazor Server
Minimizes server load	Server is work horse
Scaling is cost effective	Potentially more scaling costs
Longer initial loading times	Normal initial loading times
Option to work offline	Requires connection
Restricted to browser capabilities	Can use most .NET APIs

What to Build with ASP.NET Core?



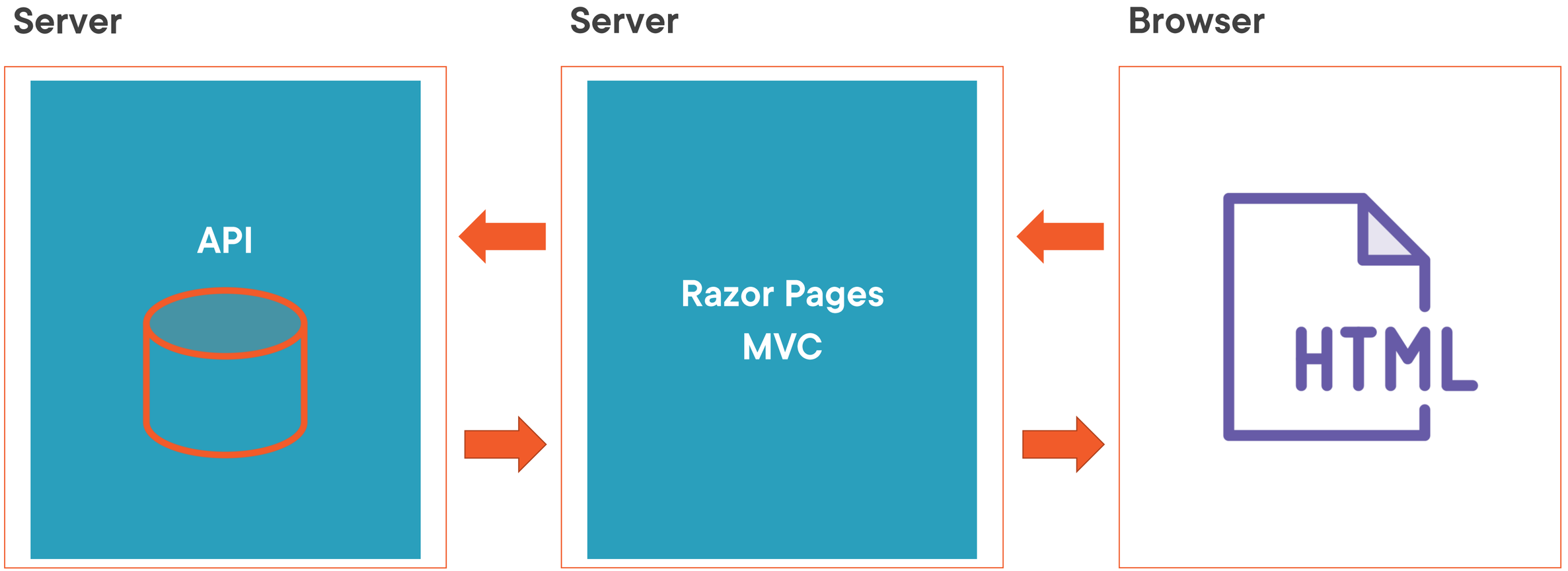
Backend API Applications



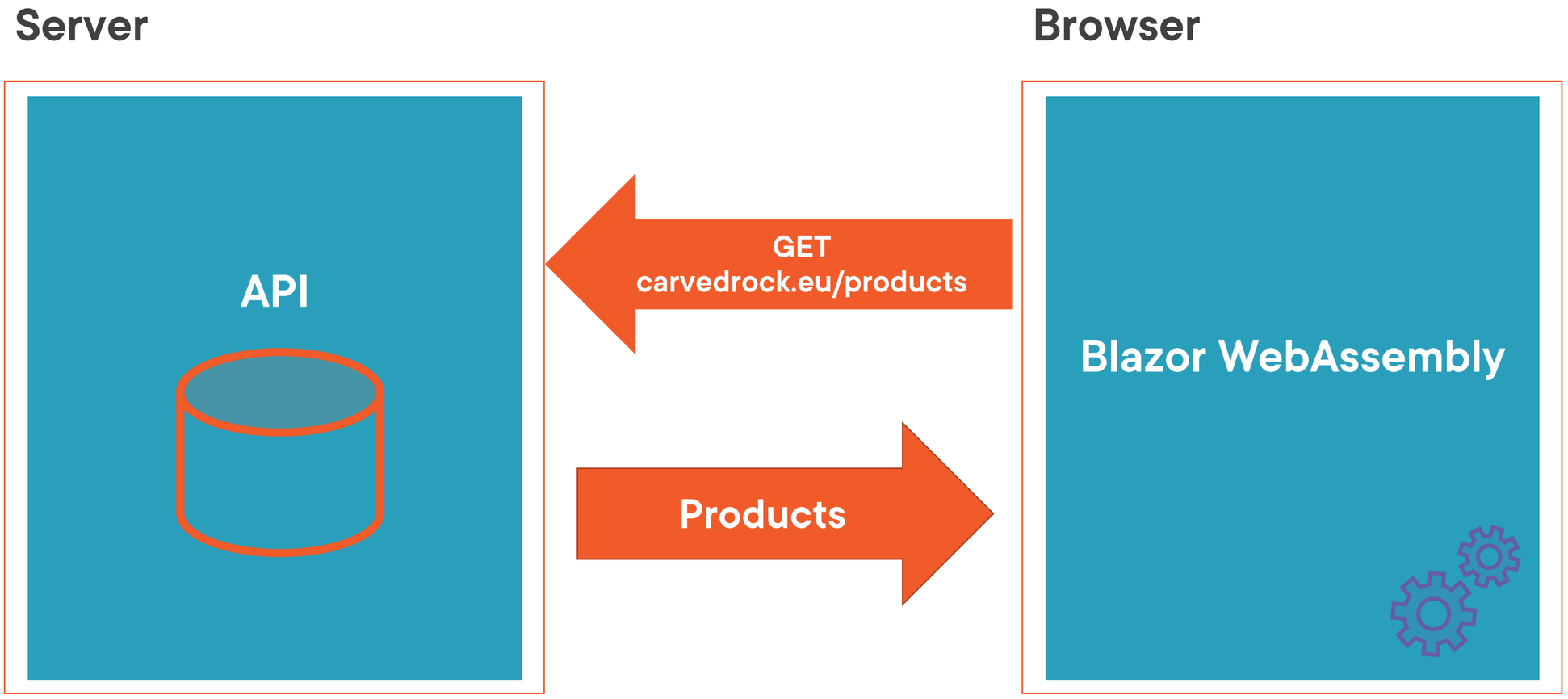
JSON Example


```
{
    "id": 1,
    "name": "Mountain Walkers",
    "price": 219.5,
    "stock": 12
}
```

APIs With Server-Rendered Frontends



APIs With Client-Rendered Frontends



REST APIs With Web API

Leverage HTTP protocol

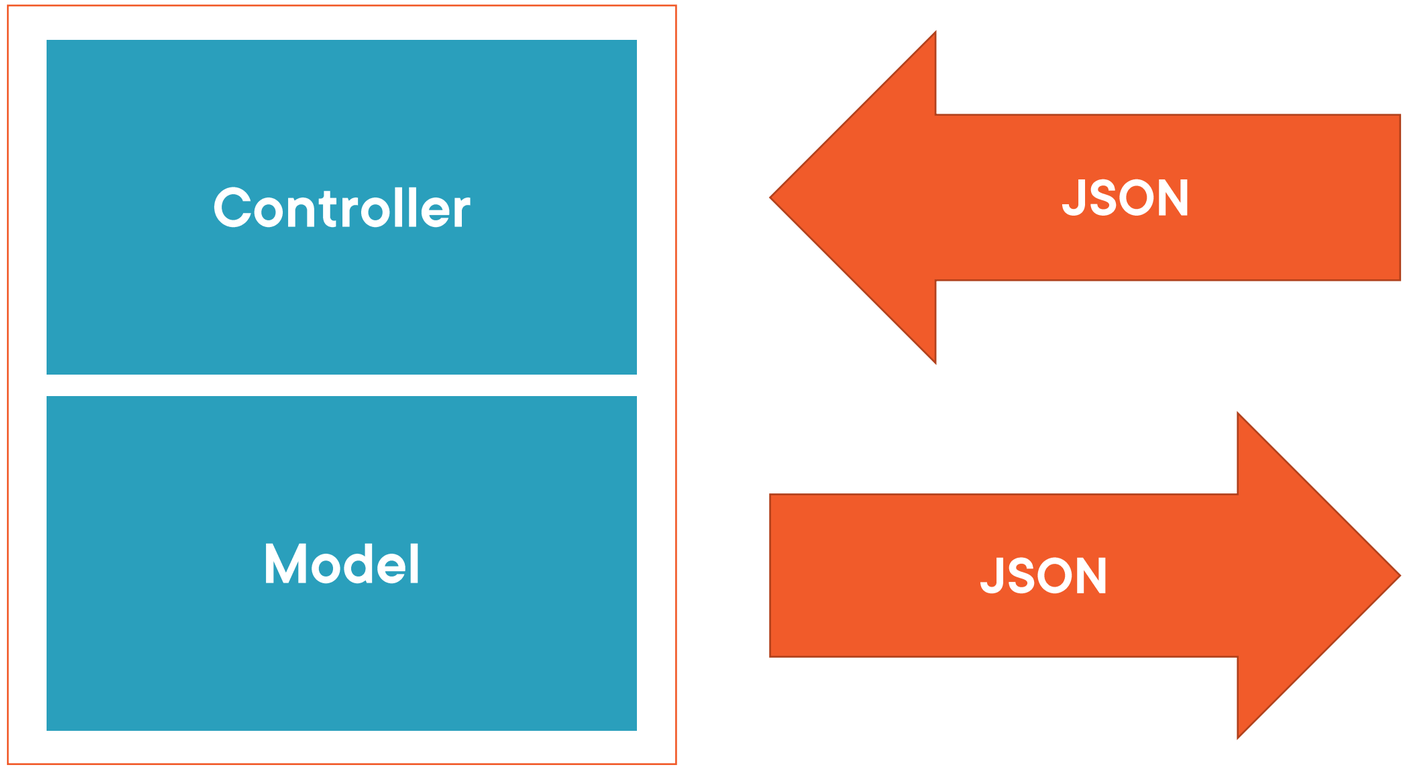
Each piece of data is available at a unique location

HTTP methods are mapped to actions

HTTP status codes are used to determine outcomes

Response can also contain pointers on what to do next

Web API



gRPC

Standard

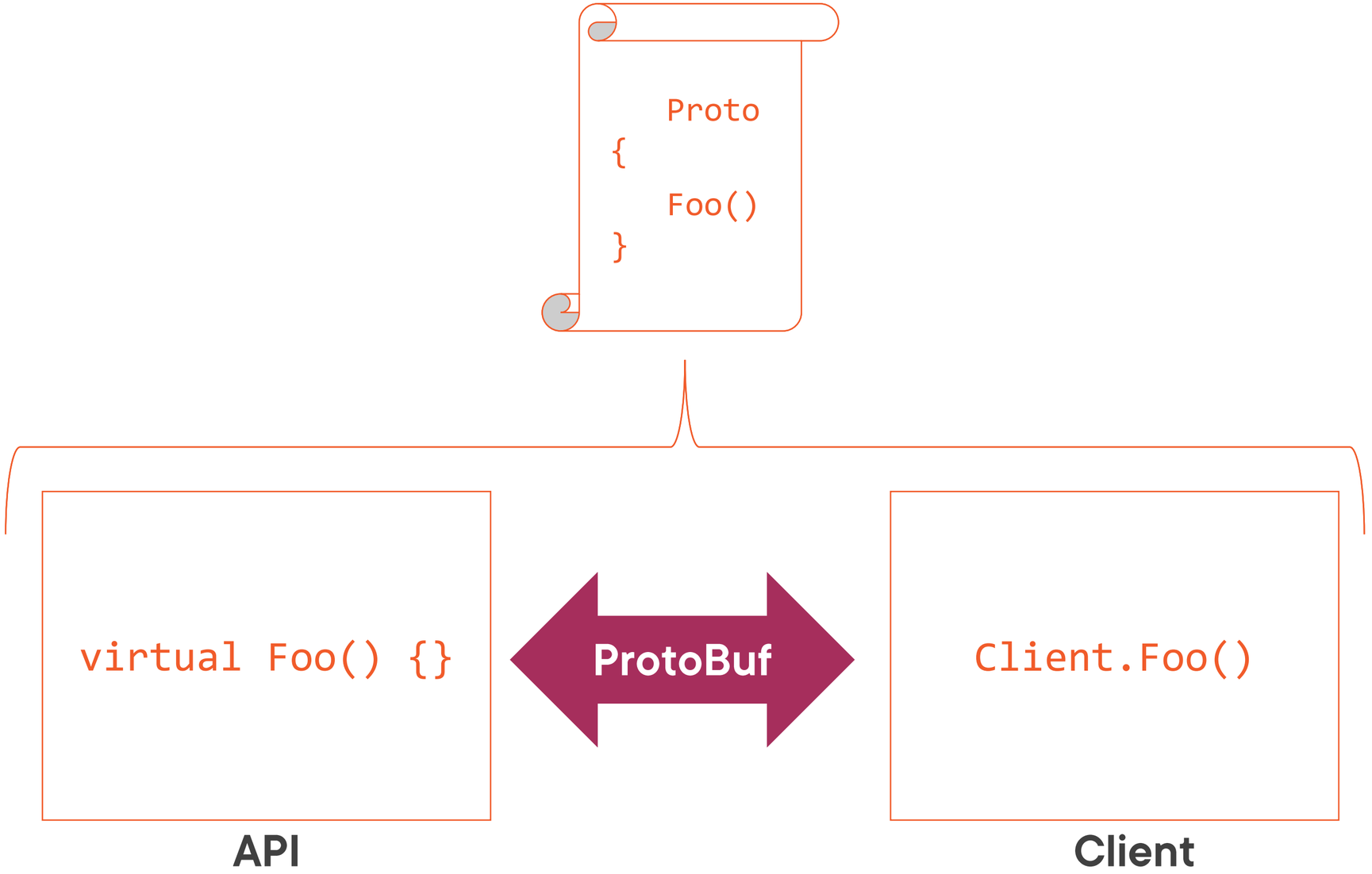
Remote Procedure Call

Focus on performance

Protocol Buffers

Contract first

gRPC with Protocol Buffers



Web API and gRPC Compared

REST with Web API	gRPC
Content first (URLs, HTTP verb, JSON)	Contract first (proto file)
Message content is human readable	Contract is human readable
Utilizes HTTP	Hides remoting complexity using RPC

gRPC: Behavioral coupling

gRPC and Web API Considerations

gRPC for internal use

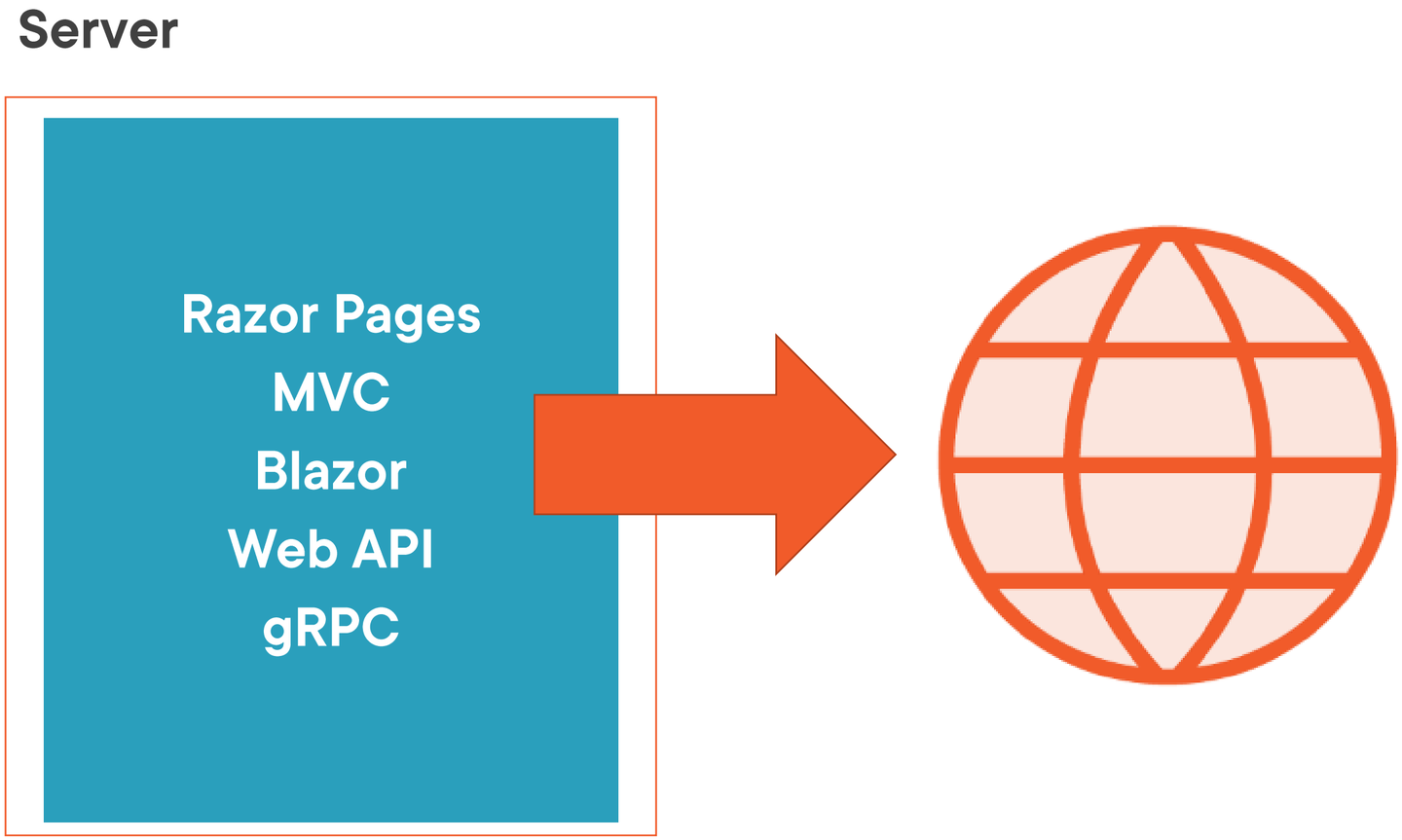
Web API also for external use

Mixing and Matching

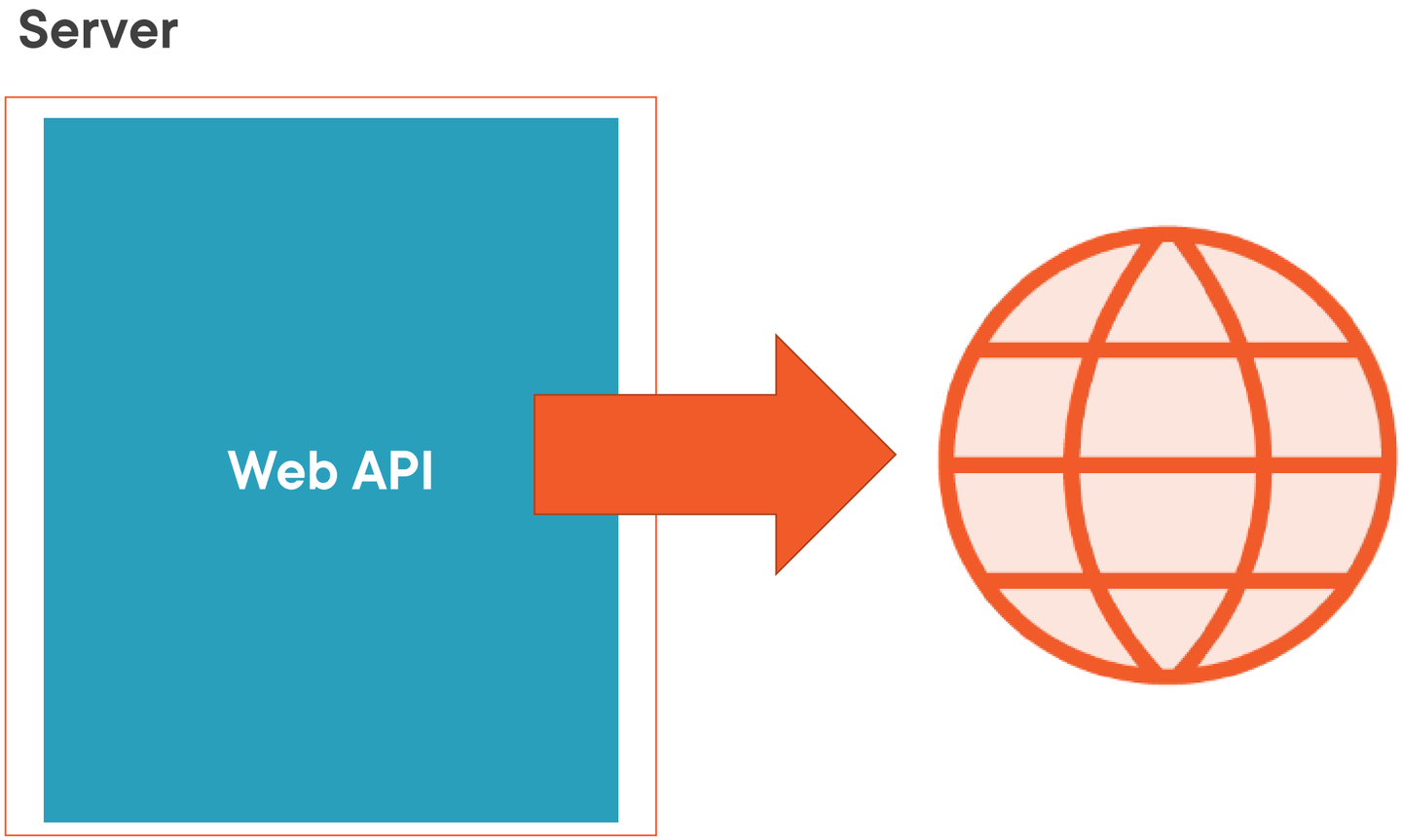
Web API and gRPC together

Frontend and backend together

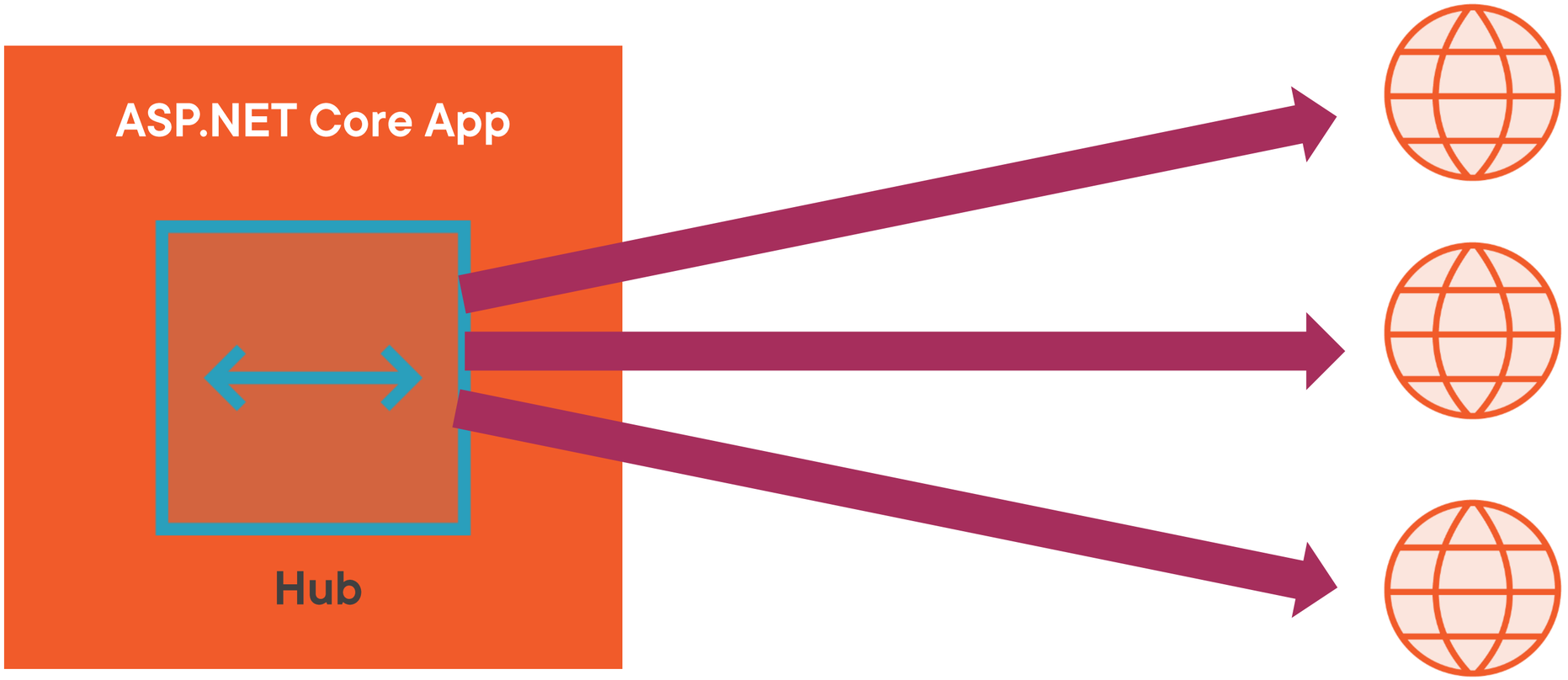
Server to Browser Client



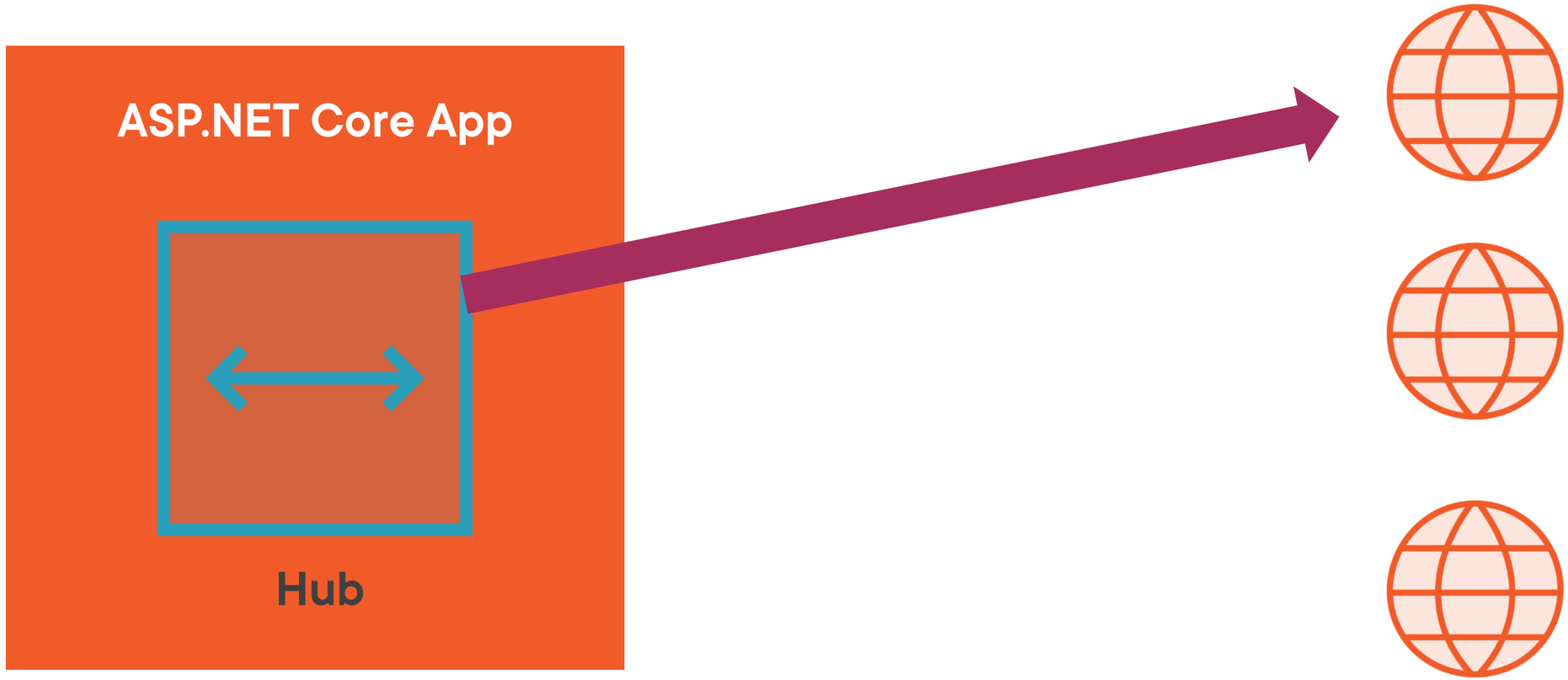
Web API to Blazor WebAssembly



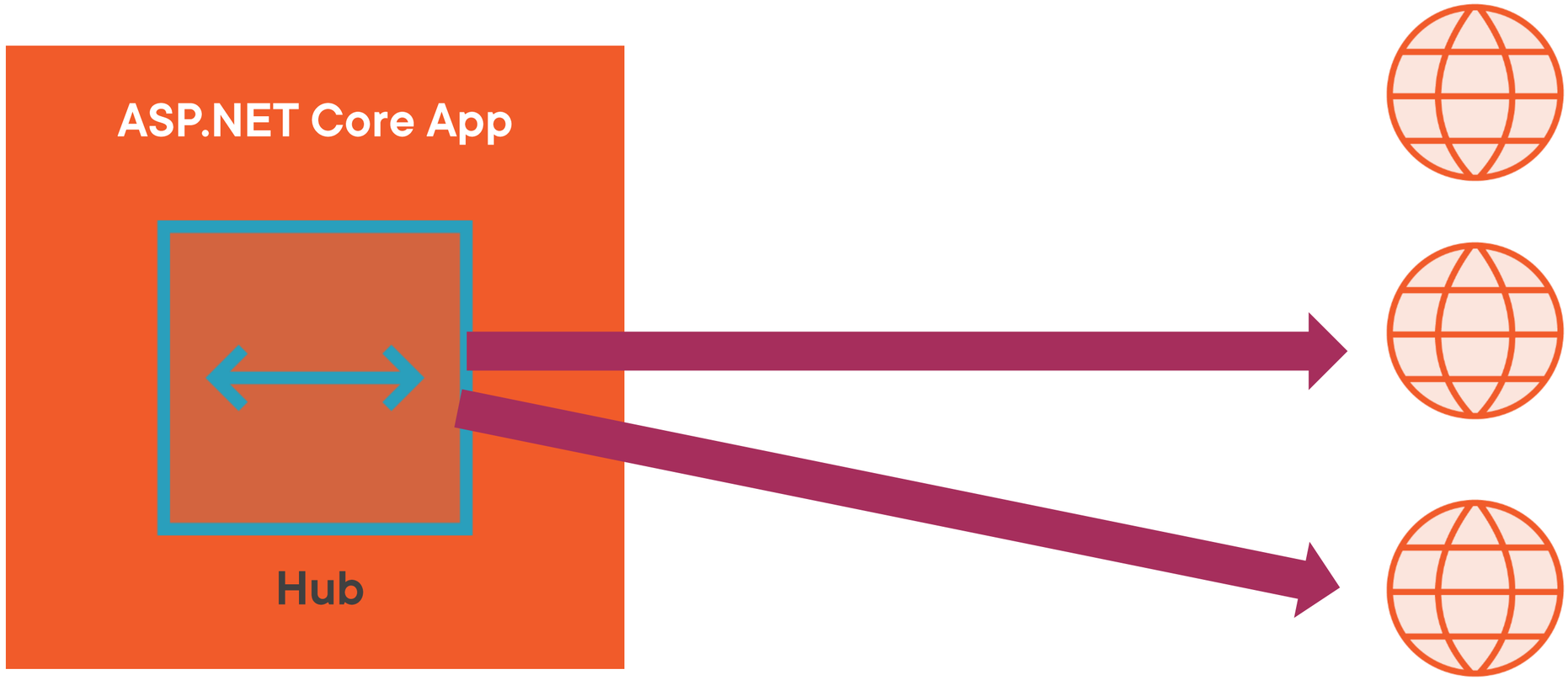
SignalR



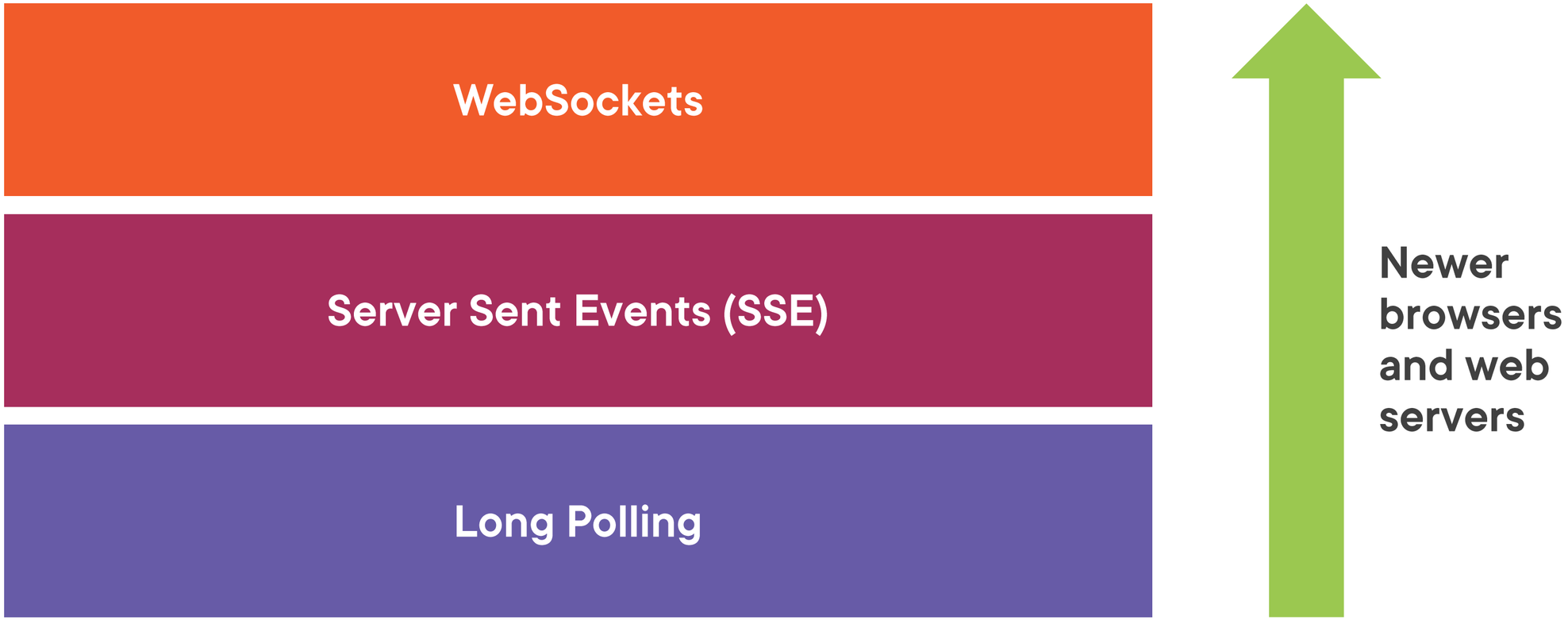
SignalR



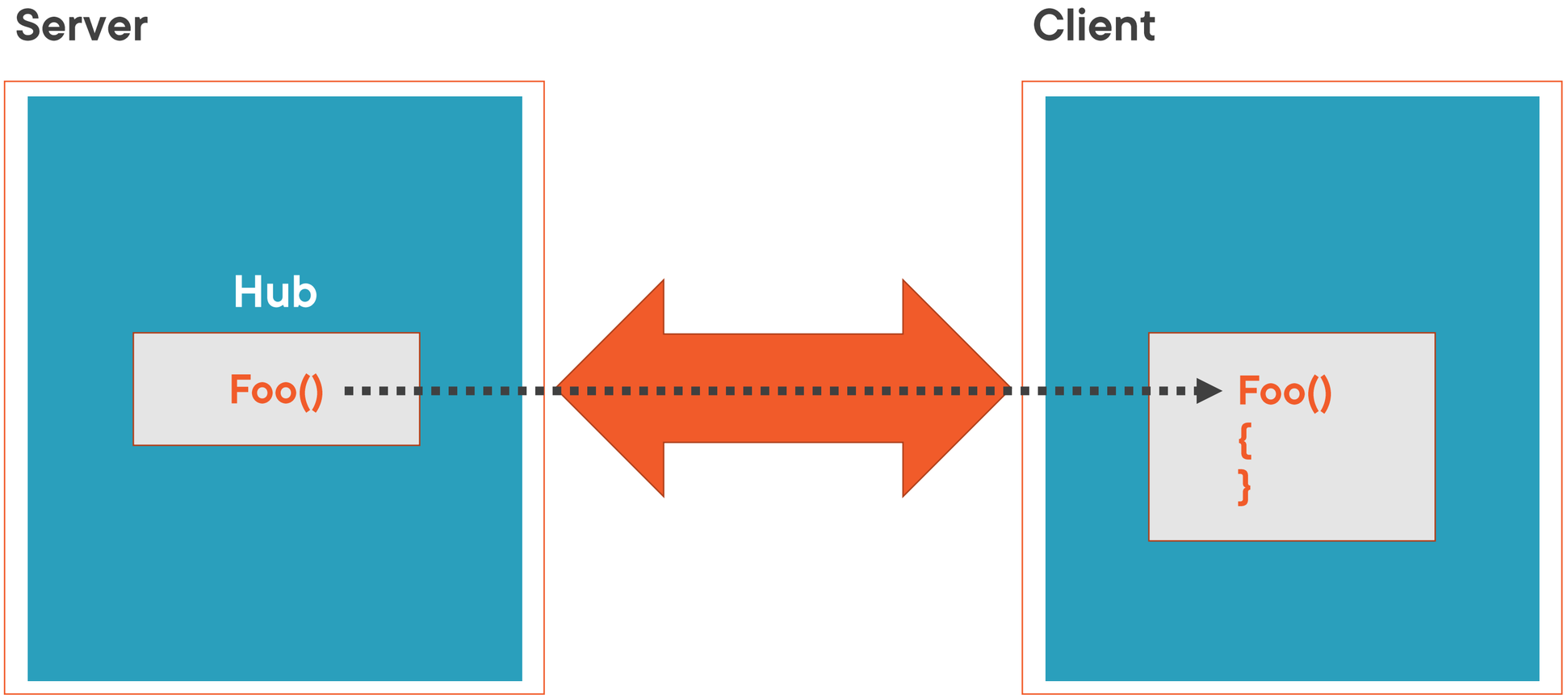
SignalR



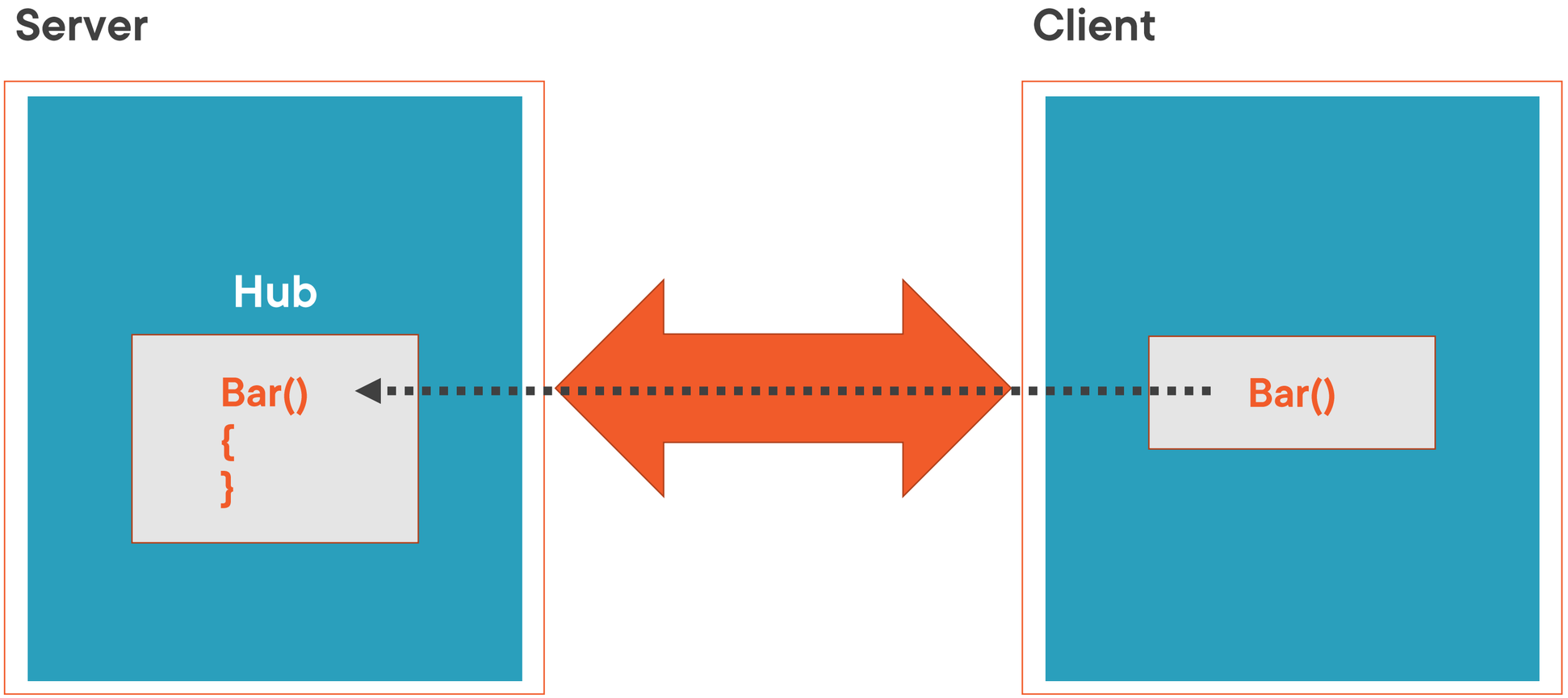
Transports and SignalR



SignalR: Remote Procedure Call



SignalR Server to Client



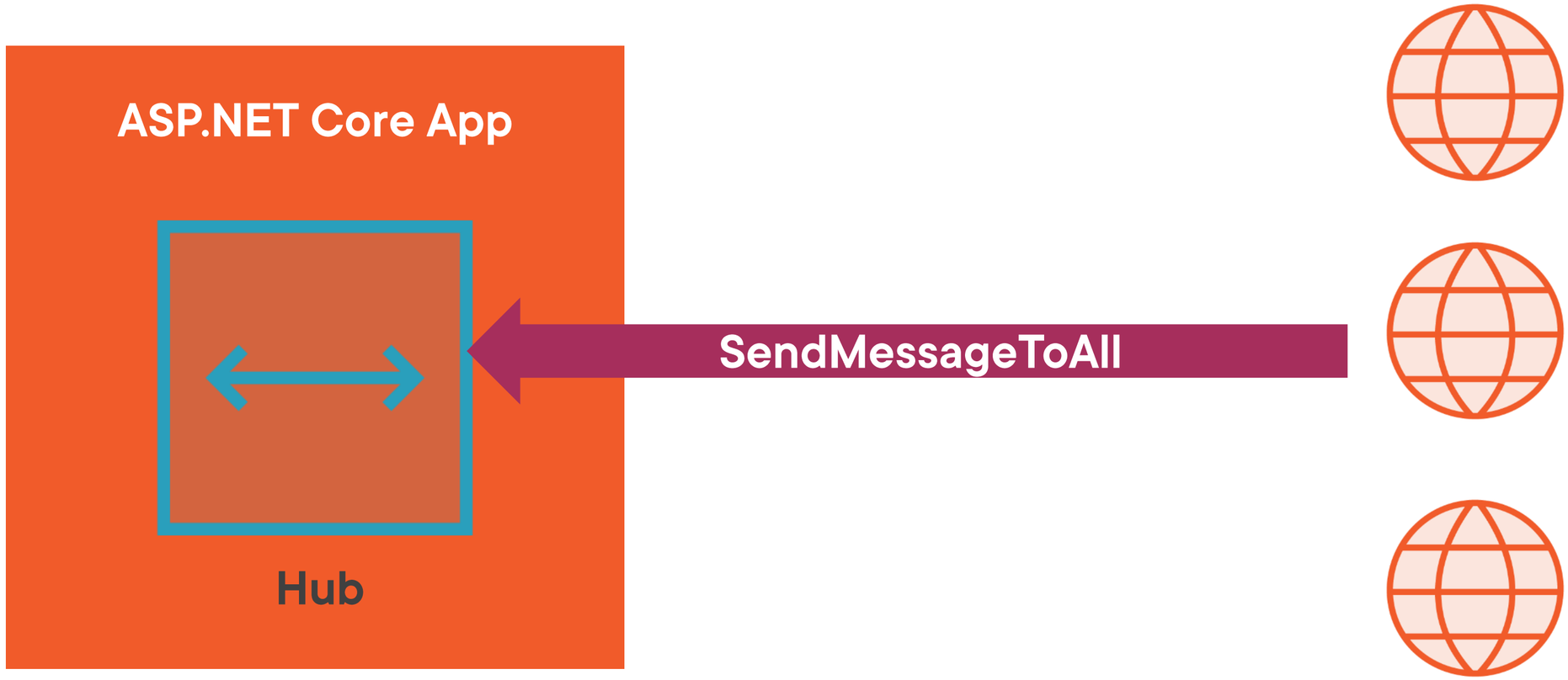
SignalR Limitations

No contract

Stability connection

Security

SignalR: Hub Method Call



SignalR: Client Methods Call