# Blazor

## Blazing Into the Future of Web Development


Roland Guijt

MVP | CONSULTANT | TRAINER | AUTHOR

@rolandguijt rolandguijt.com roland.guijt@gmail.com

https://github.com/rolandguijt

# The Workshop

Consists of theory with labs

Lab files including slides in GitHub repository

Labs use .NET 6

Intermediate level

Breaks

Questions

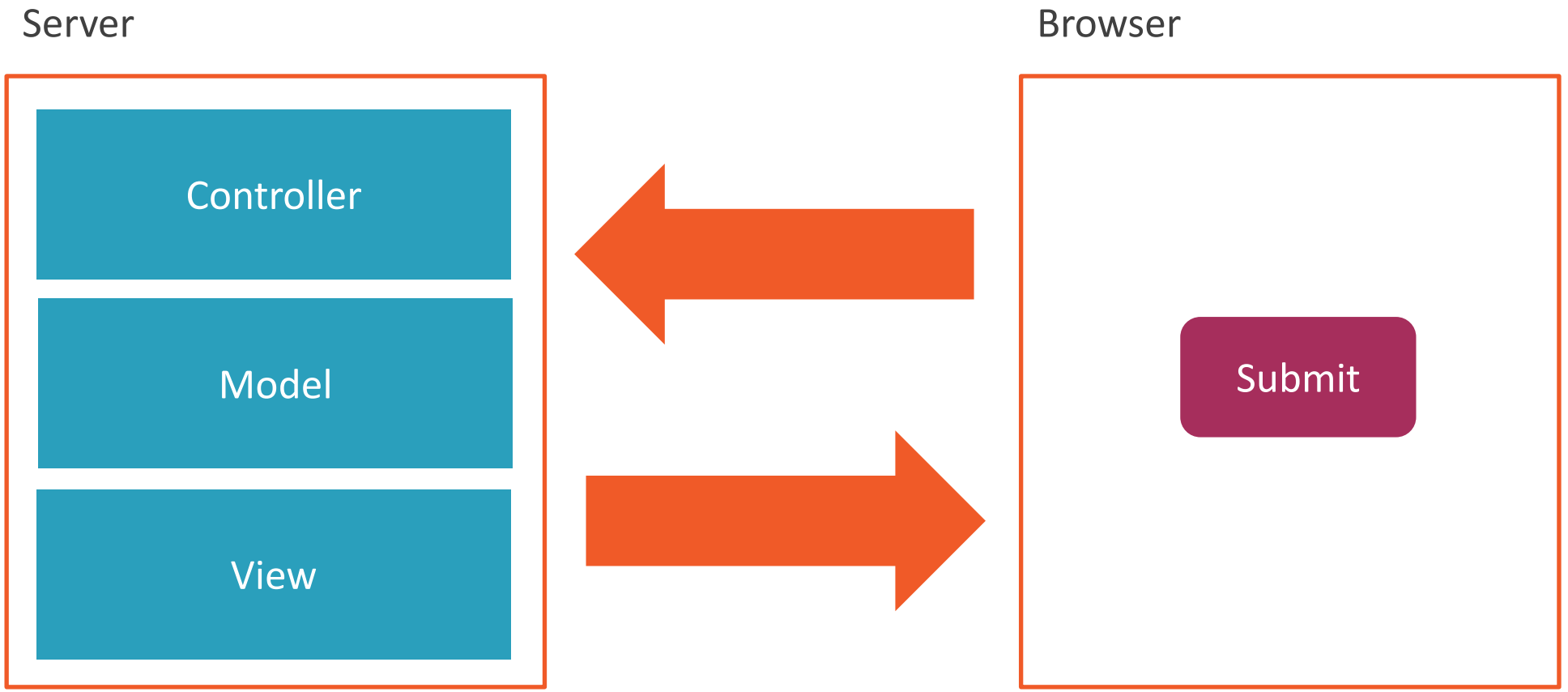
# The Plan

What is Blazor?

The basics of component building

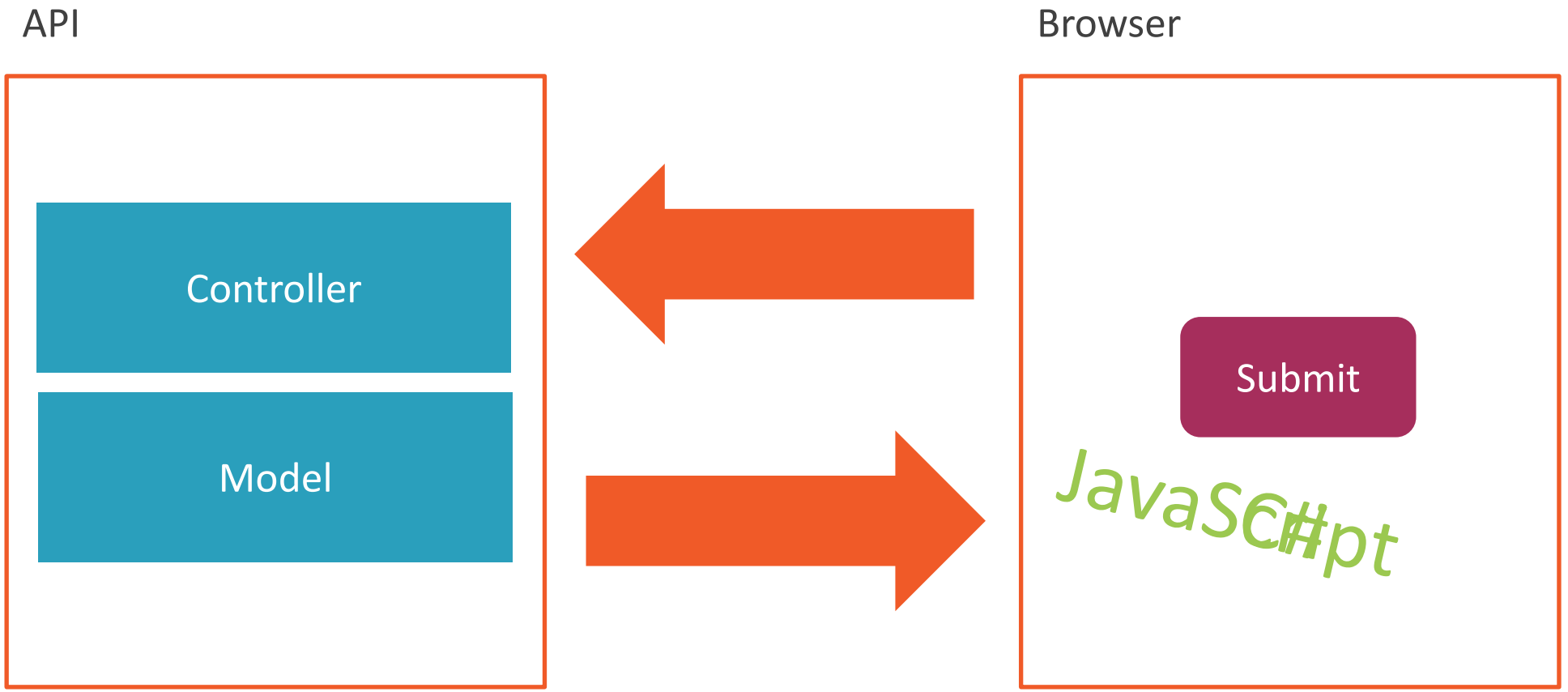
More advanced component capabilities

# MVC



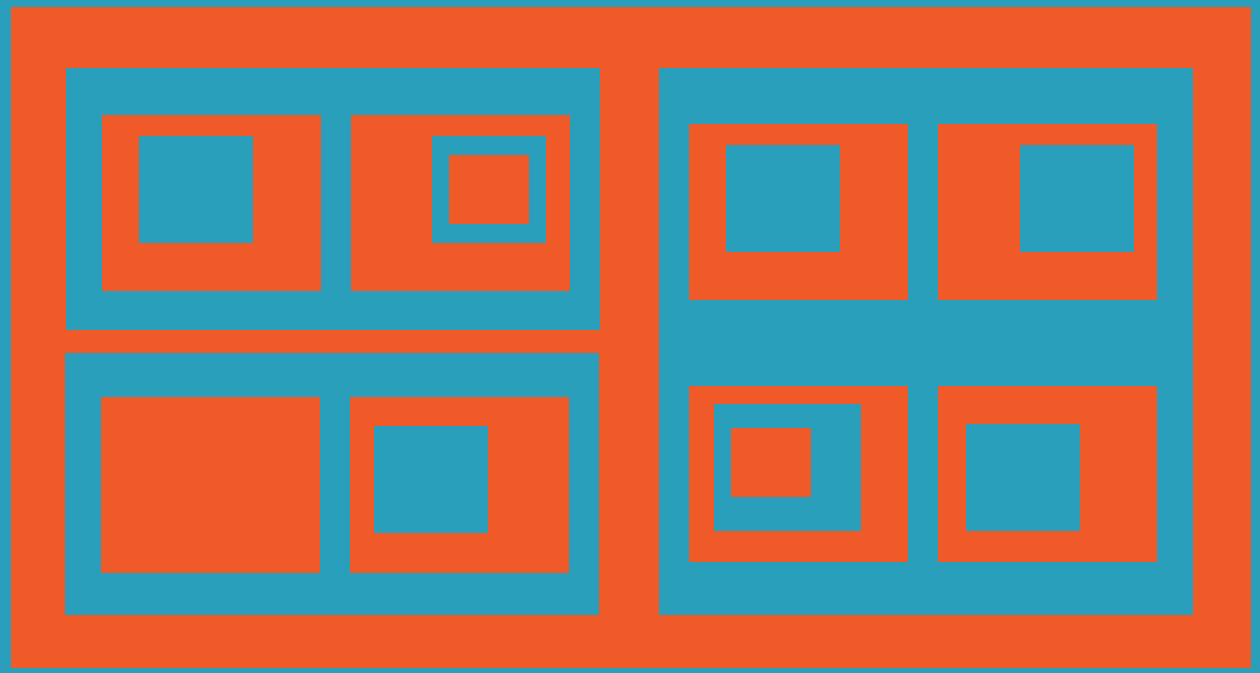

@rolandguijt

# Single Page Application

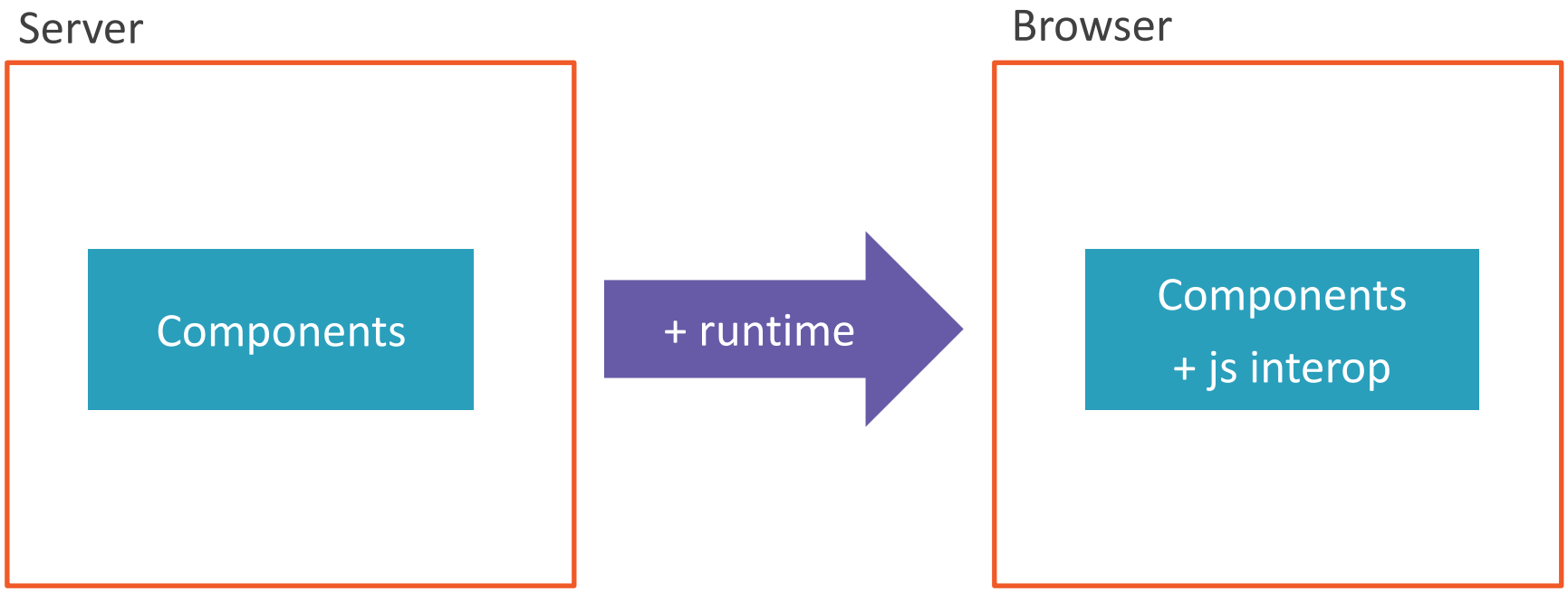


Blazor is a single-page application framework that lets you write C# on the browser side.

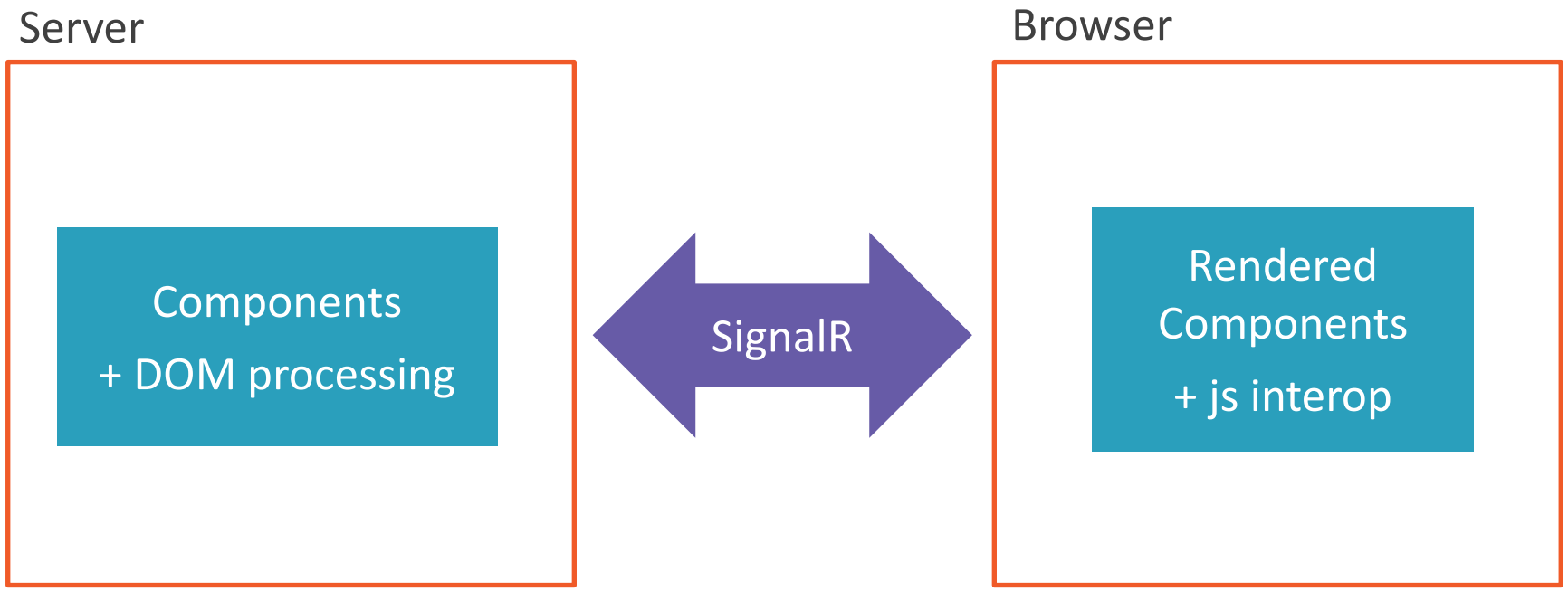
# Creating Blazor App ==
# Writing components.



# Hosting Model #1: Blazor Web Assembly



# Hosting Model #2: Blazor Server



Component structure is identical for both hosting models.

# Demo

Server- and Client-side Blazor: Comparison

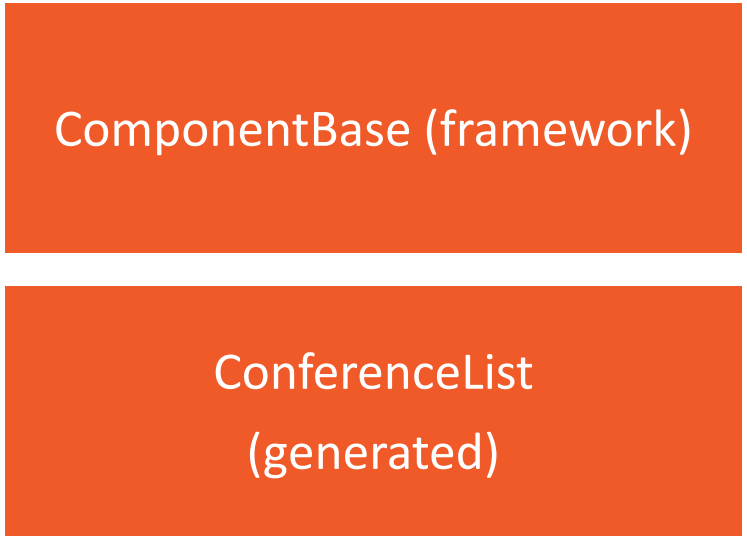
# Demo

## Exploring a demo app

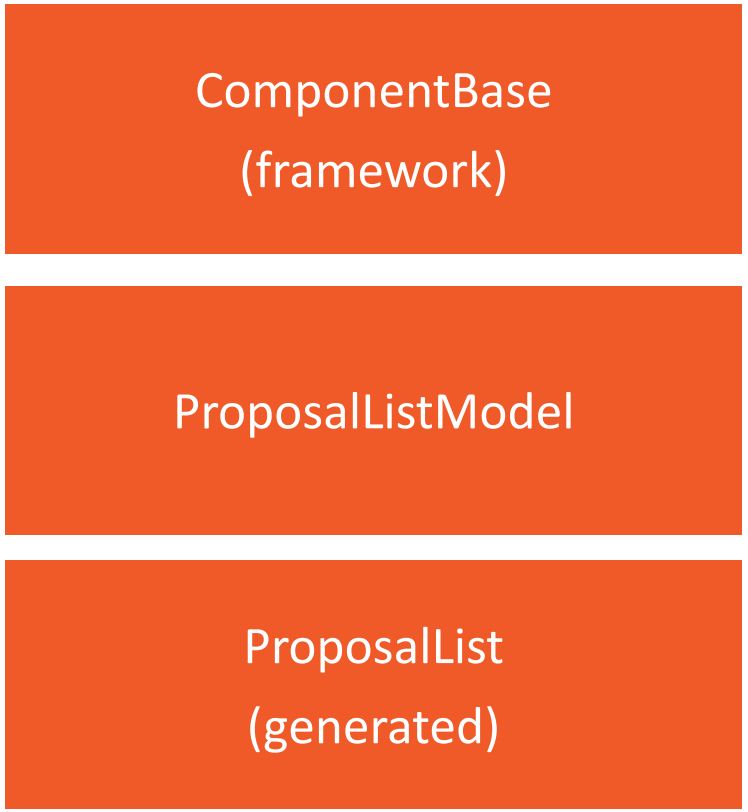
https://github.com/RolandGuijt/BlazorExample

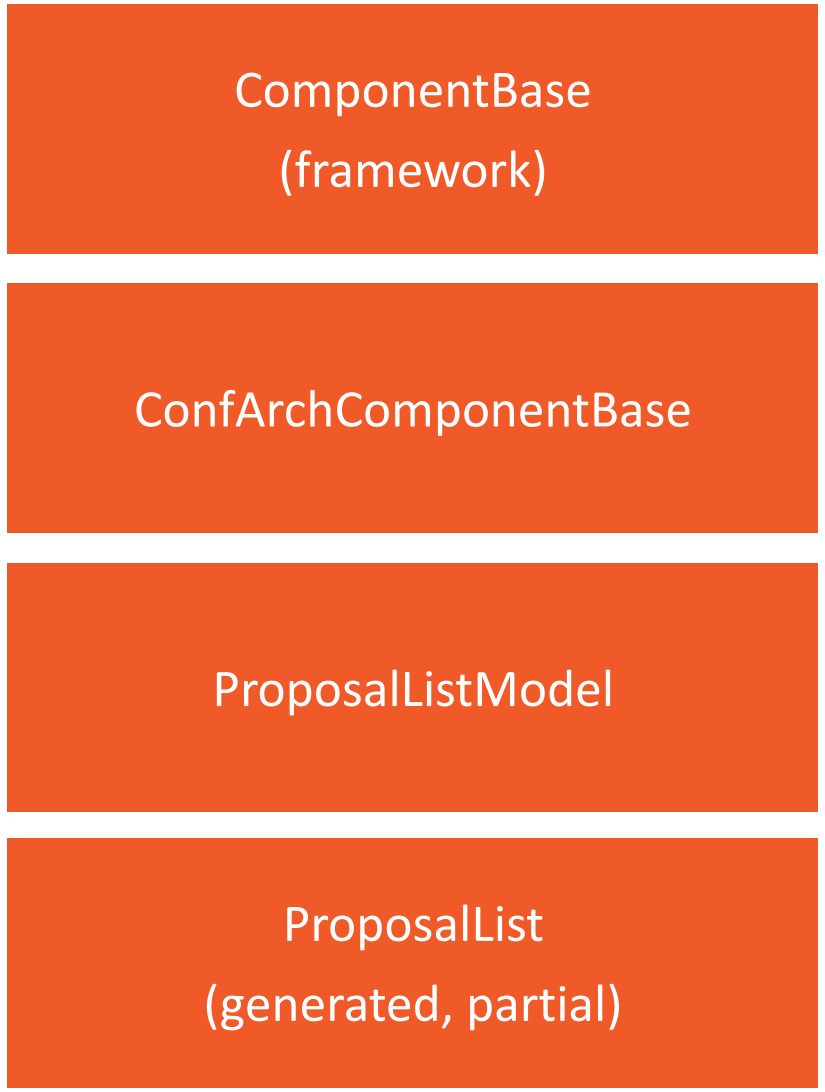
- Understanding the diff mechanism
- Getting to know the code structure

# Partial Component Class Hierarchy Without Code-behind



# Partial Component Class Hierarchy with Code-behind




ComponentBase
(framework)
ConfArchComponentBase
ProposalListModel
ProposalList
(generated, partial)

# Lab time!
# Part 1

# Github repository:

https://4sh.nl/blazorwork

# Surprise lab: Building a new application

# Steps

- Get the HousesAPI running
- Create a Blazor application that lists the houses
- Add a details page for a house
- Add a page to add a house
- Display the bids for a house on the details page
- Add functionality to add a bid on the details page

# .NET 8+ Features

New rendering mode: static

Other rendering modes:

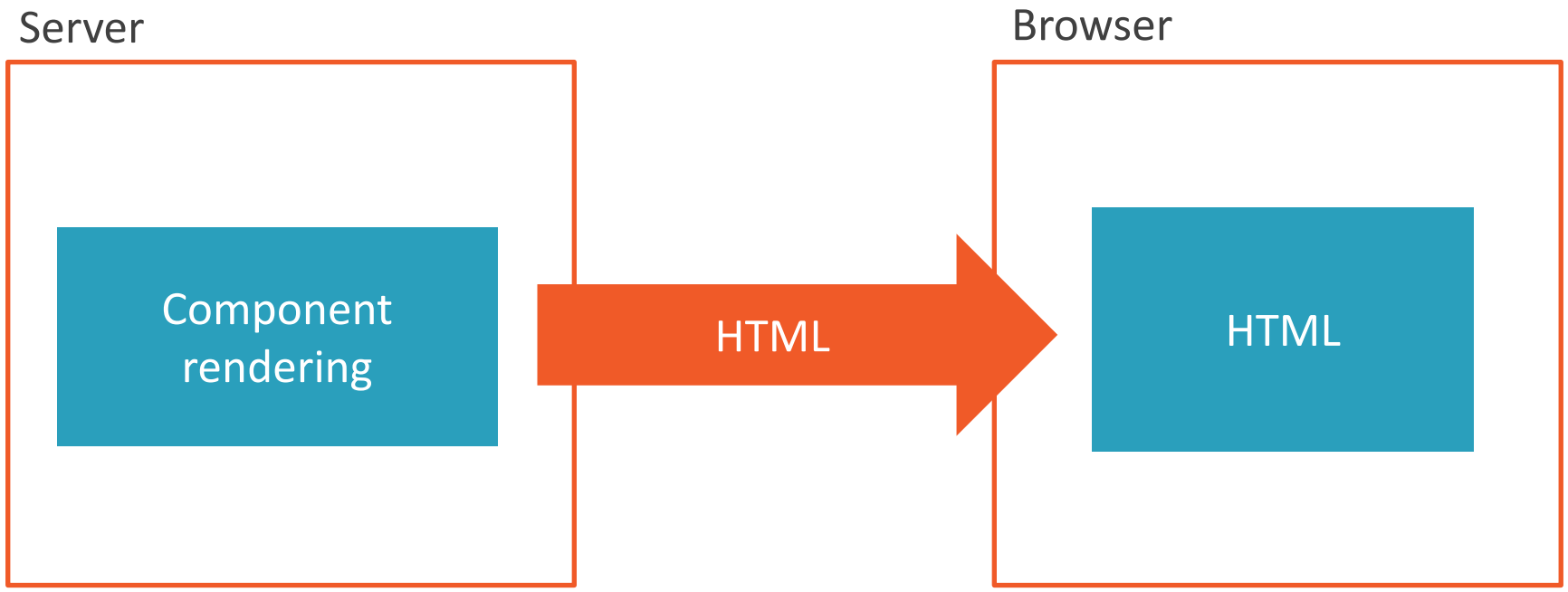
- Interactive Server
- Interactive Client
- Auto

No need to choose between rendering modes

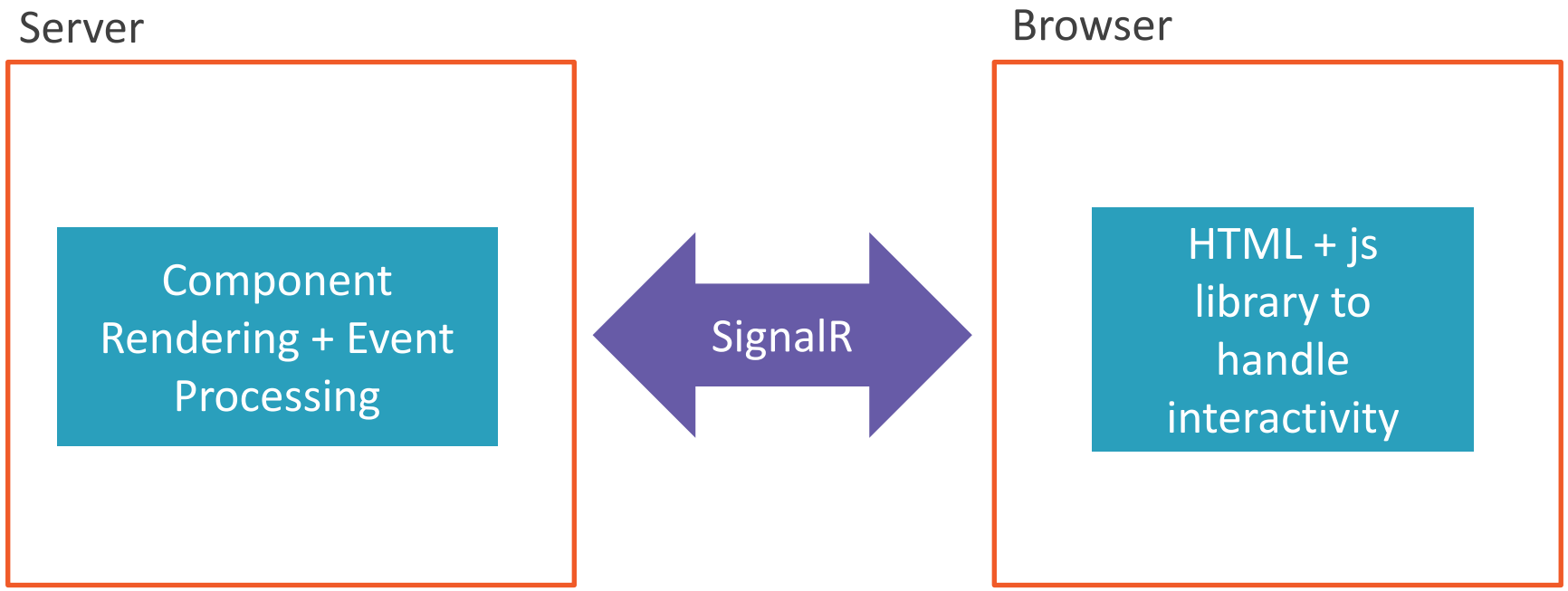
Also new: Stream rendering

Let me show you

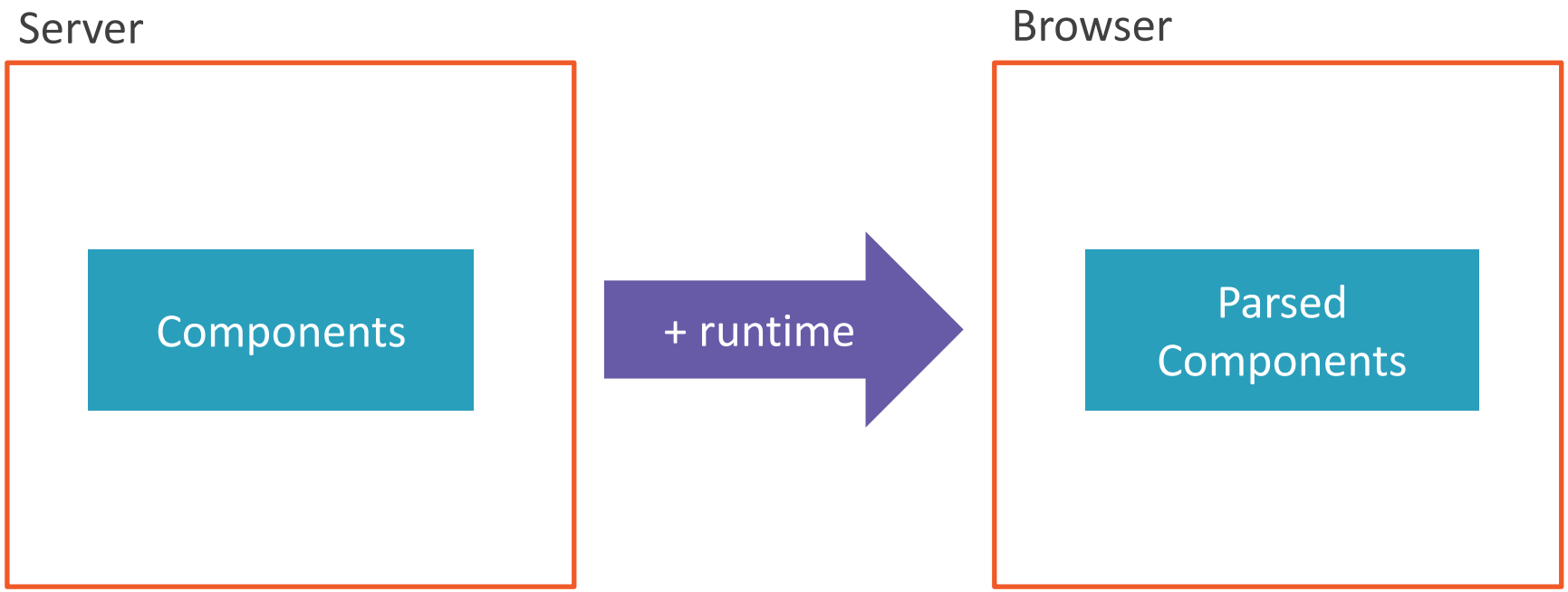
# Static Components: No Interaction



# Interactive Server Components

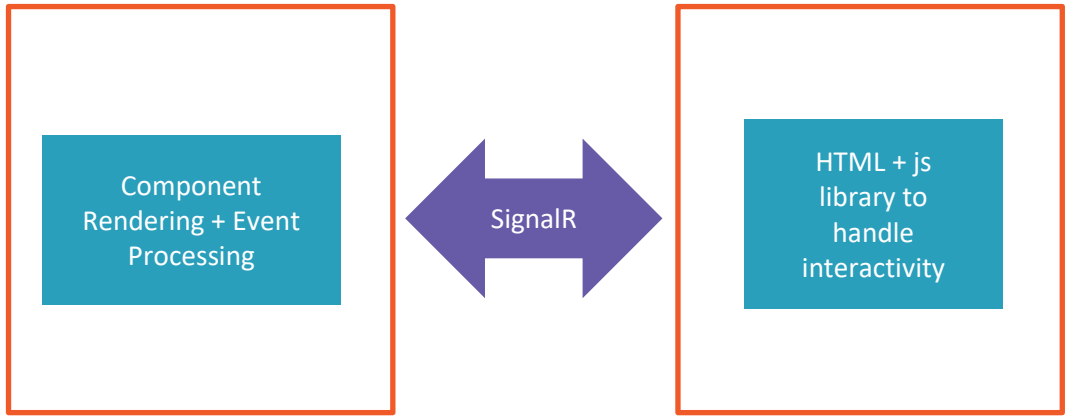


# Interactive WebAssembly Components

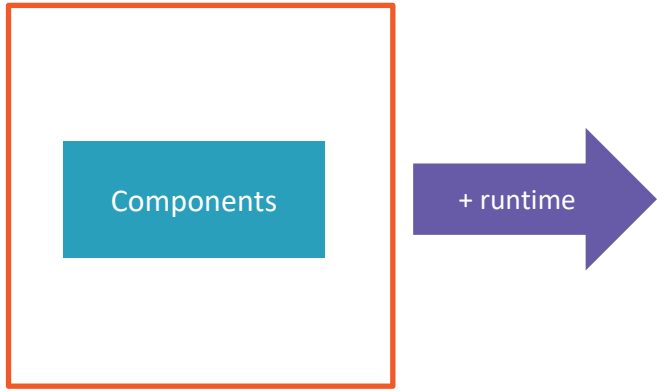


# Auto Interactive Components

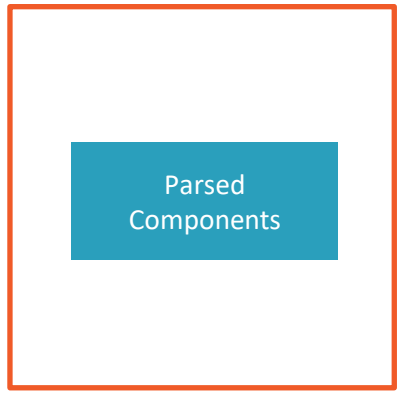
1. Start as an interactive server component



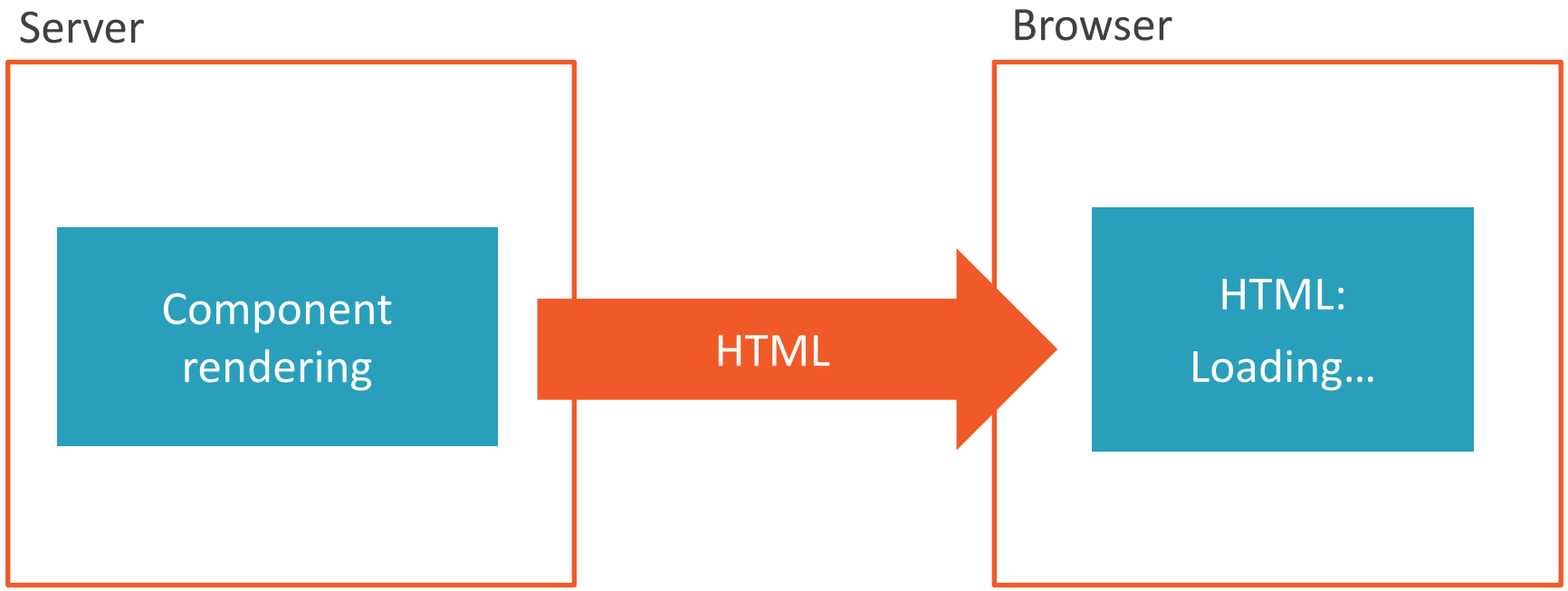

2. Load WebAssembly runtime + assemblies



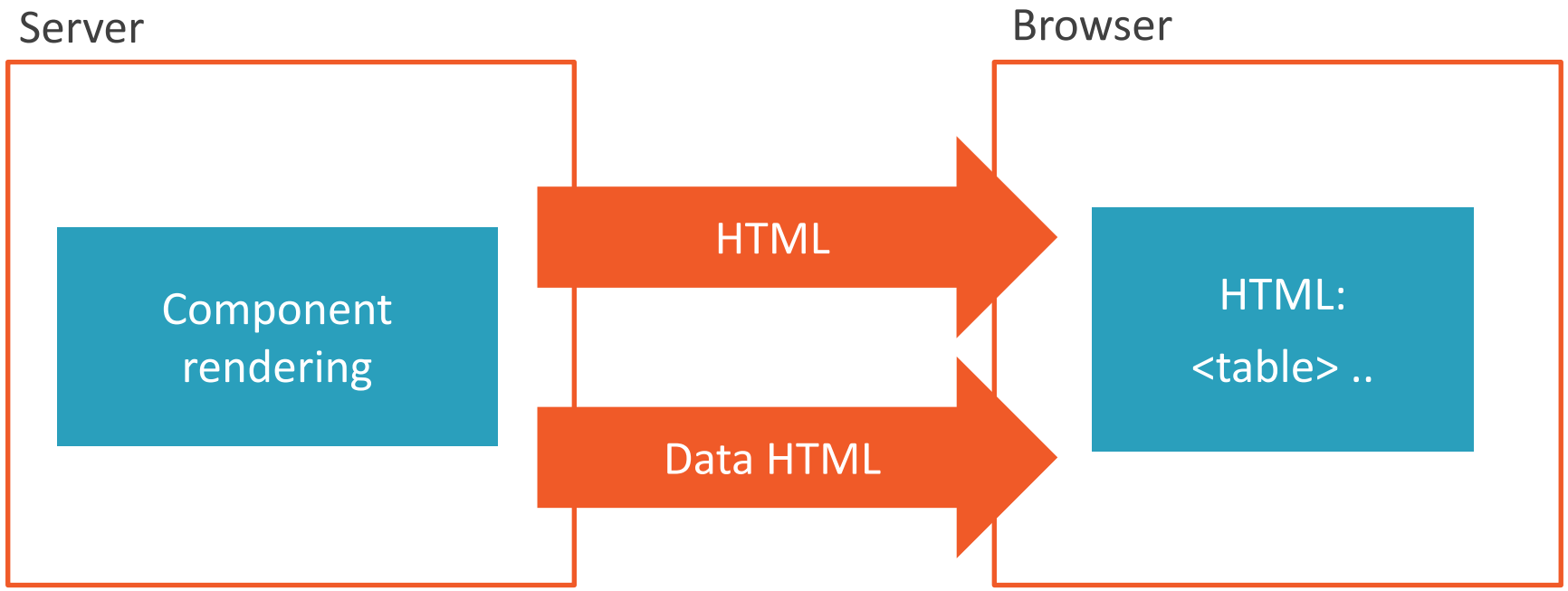

2. Continue life as an interactive client component



# Static Components: Stream Rendering



# Static Components: Stream Rendering



Any code that initializes the component fires twice!

# Demo

Component writing with:

Child Content

@bind

Chained binds

More

# Component Hierarchy

EmployeeModel

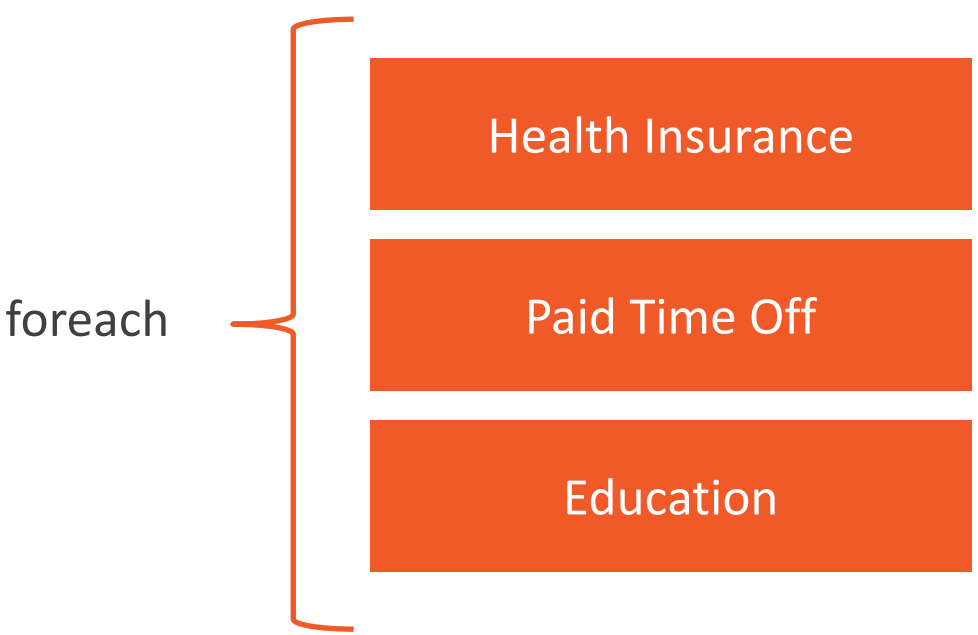
EmployeeOverview

EmployeeRow

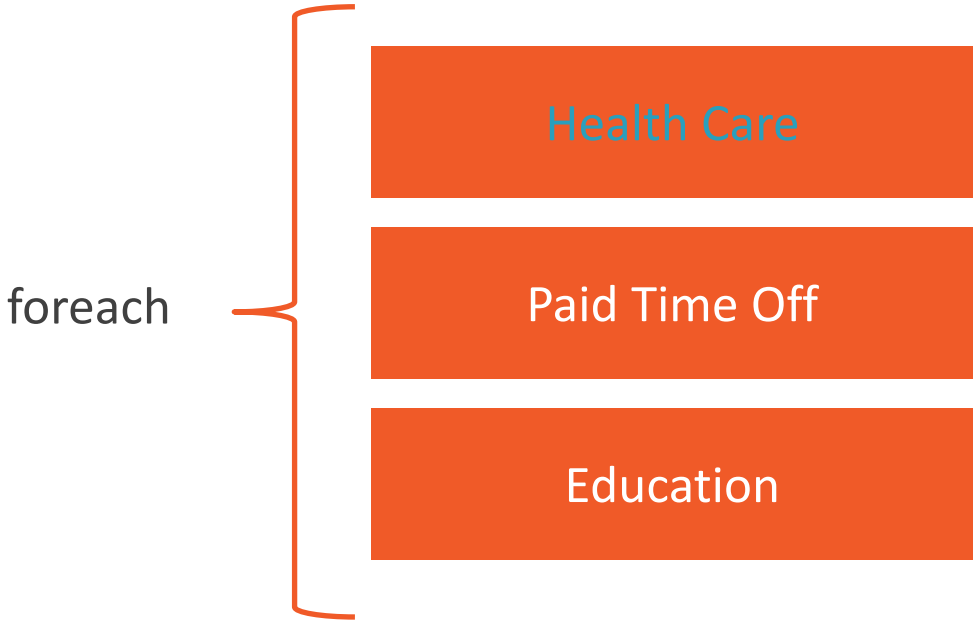
BenefitSelector

# @key

Collection:
Benefit "Health Insurance"
Benefit "Paid Time Off"
Benefit "Education"

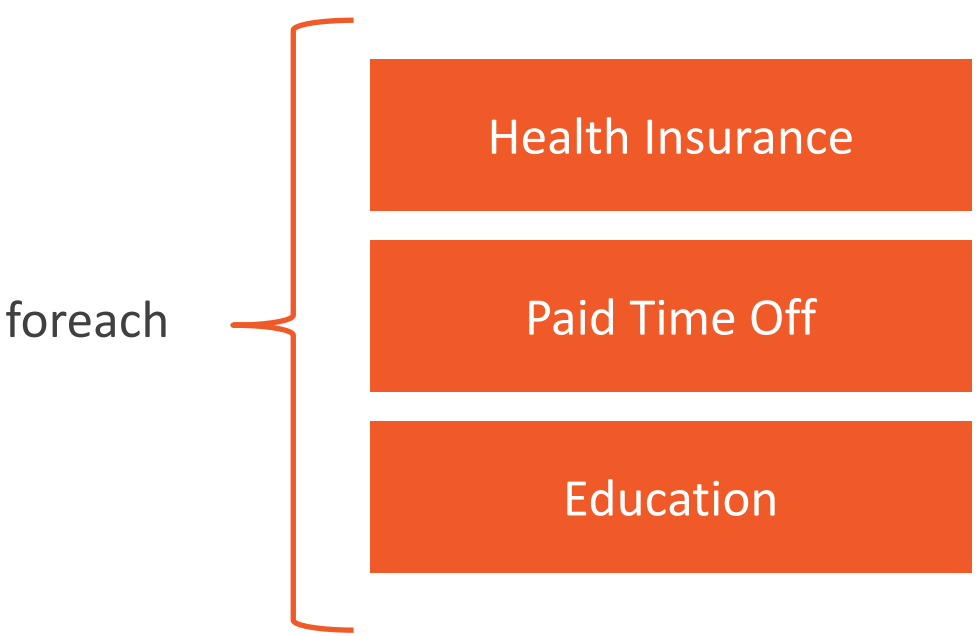



Collection:
Benefit "Health Care"
Benefit "Paid Time Off"
Benefit "Education"

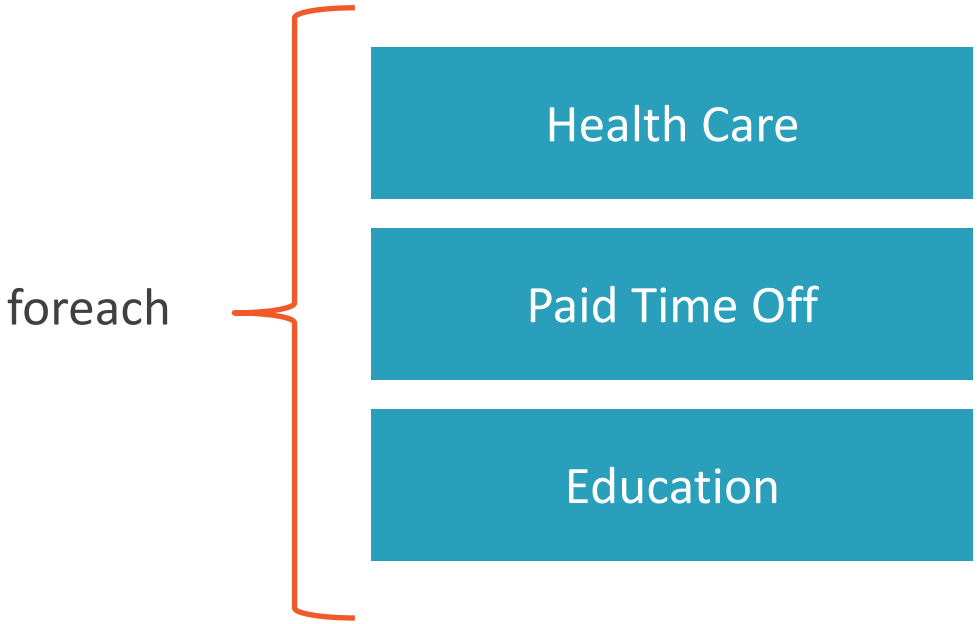


# @key

Collection:
Benefit "Health Insurance"
Benefit "Paid Time Off"
Benefit "Education"

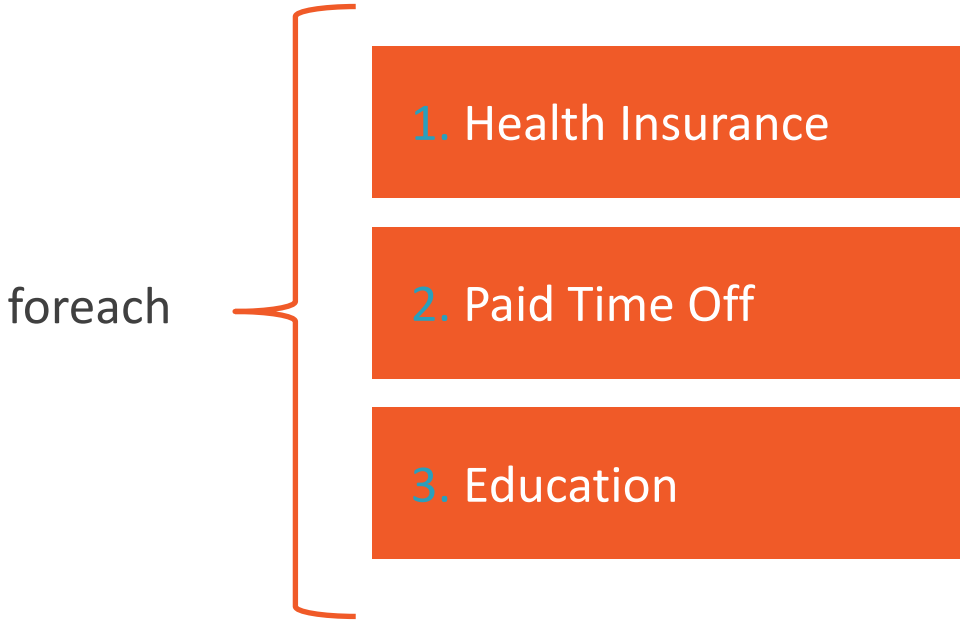



Collection:
Benefit "Health Care"
Benefit "Paid Time Off"
Benefit "Education"

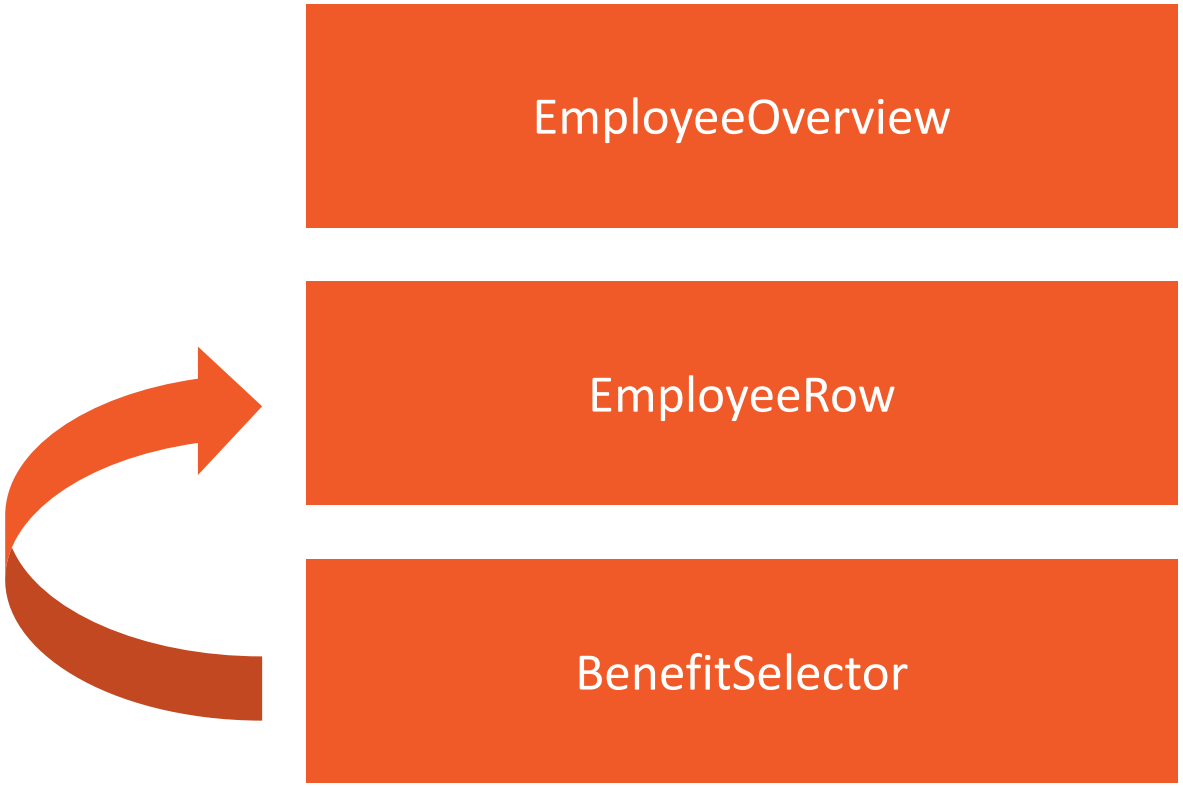


# @key

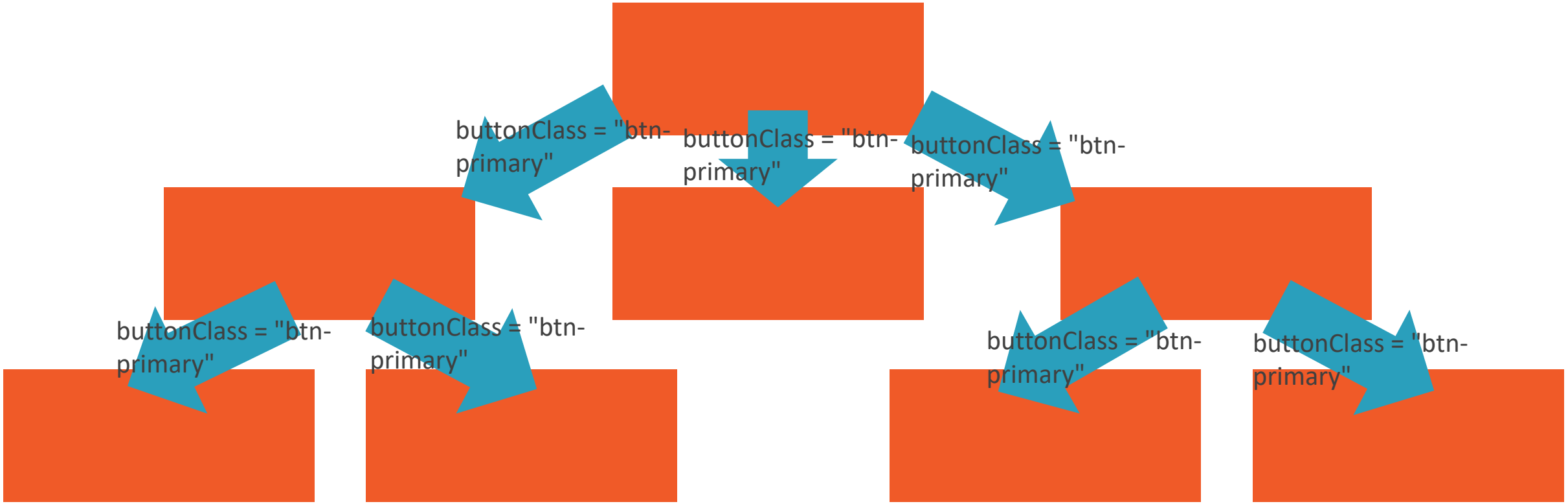
Collection:
Benefit 1 "Health Insurance"
Benefit 2 "Paid Time Off"
Benefit 3 "Education"



# Component Hierarchy



# Cascading Values



# Cascading Values

```
<CascadingValue Value="@buttonClass">
        <CascadingValue Value="@inputClass">
            <EmployeeOverView>
                <AddEmployeeDialog>
                [CascadingParameter]
                public string CascadingValue { get; set ; }
```

# Cascading Values

```
<CascadingValue Value="@buttonClass" Name="button">
        <CascadingValue Value="@inputClass" Name="input">
            <EmployeeOverView>
                <AddEmployeeDialog>
                [CascadingParameter(Name = "button")]
                public string CascadingValue { get; set ; }
```

# Lab time!
# Part 2
# Until stage 8

# Github repository:

https://4sh.nl/blazorwork

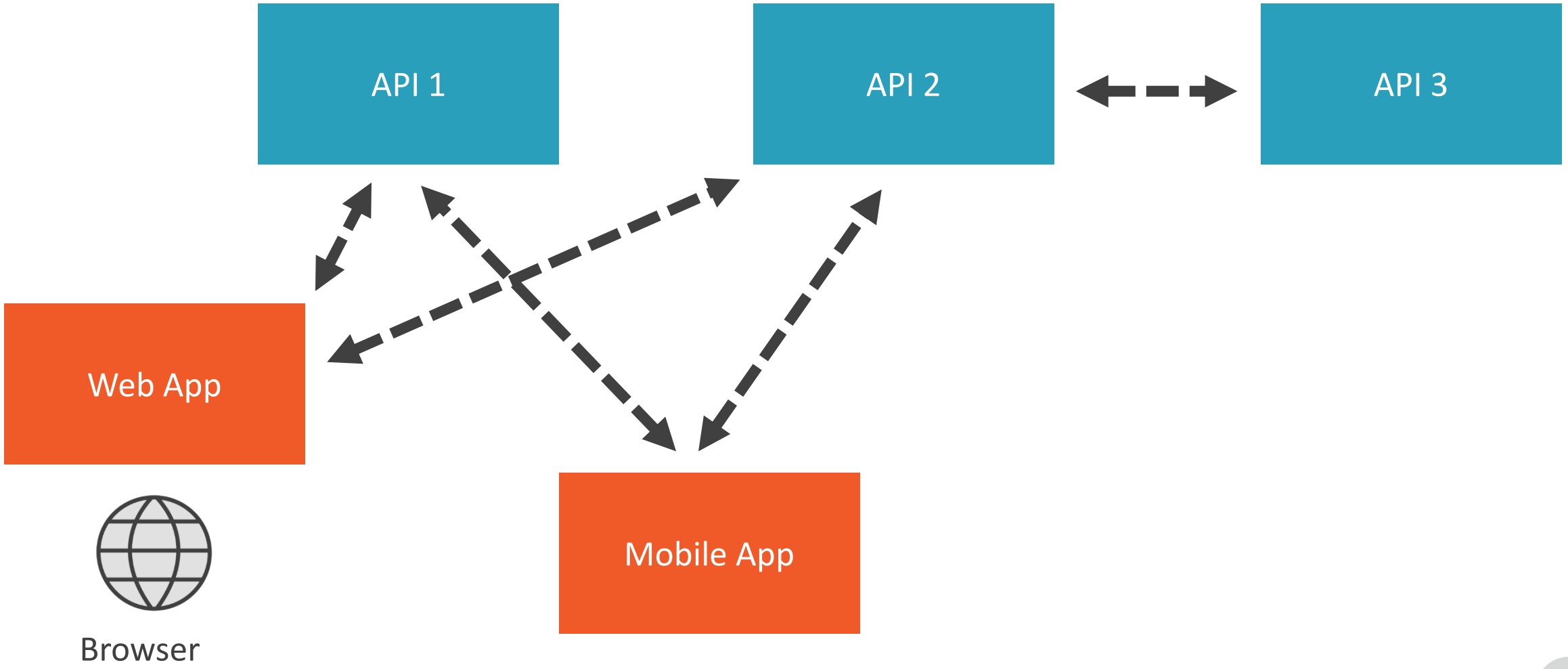
# Demo

Templated Components

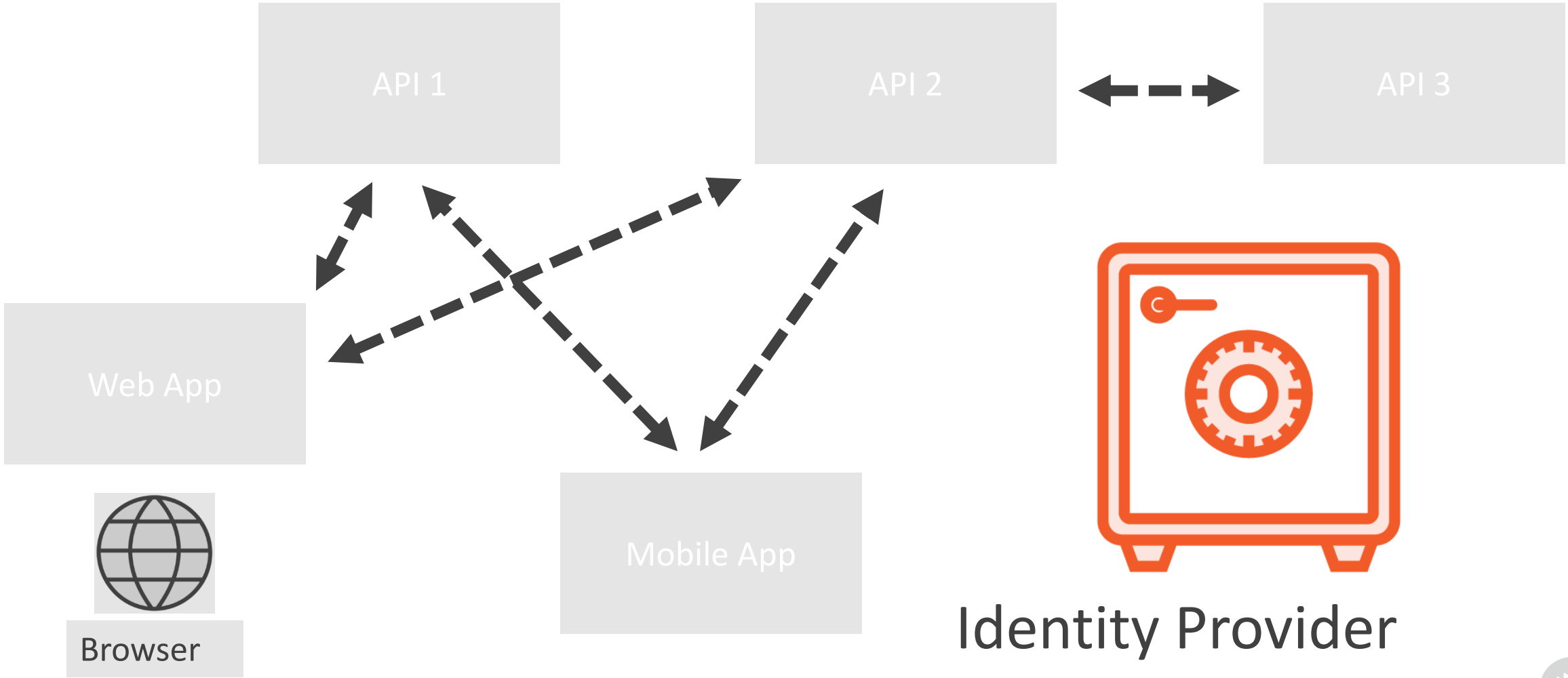
# Auth

Uses standard ASP.NET Core authentication and authorization support

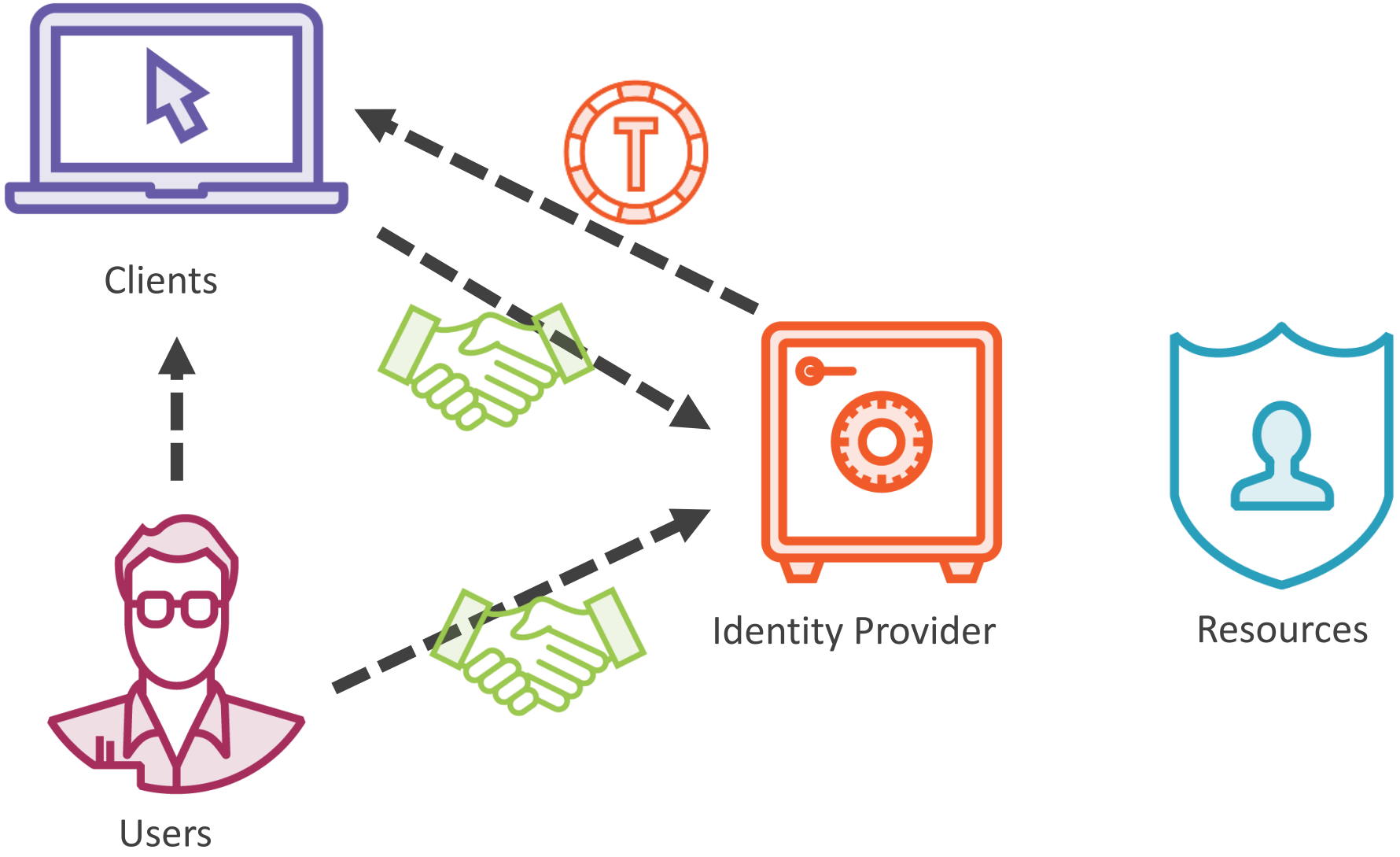
# A Typical Modern Application



# The Identity Provider



# Concepts

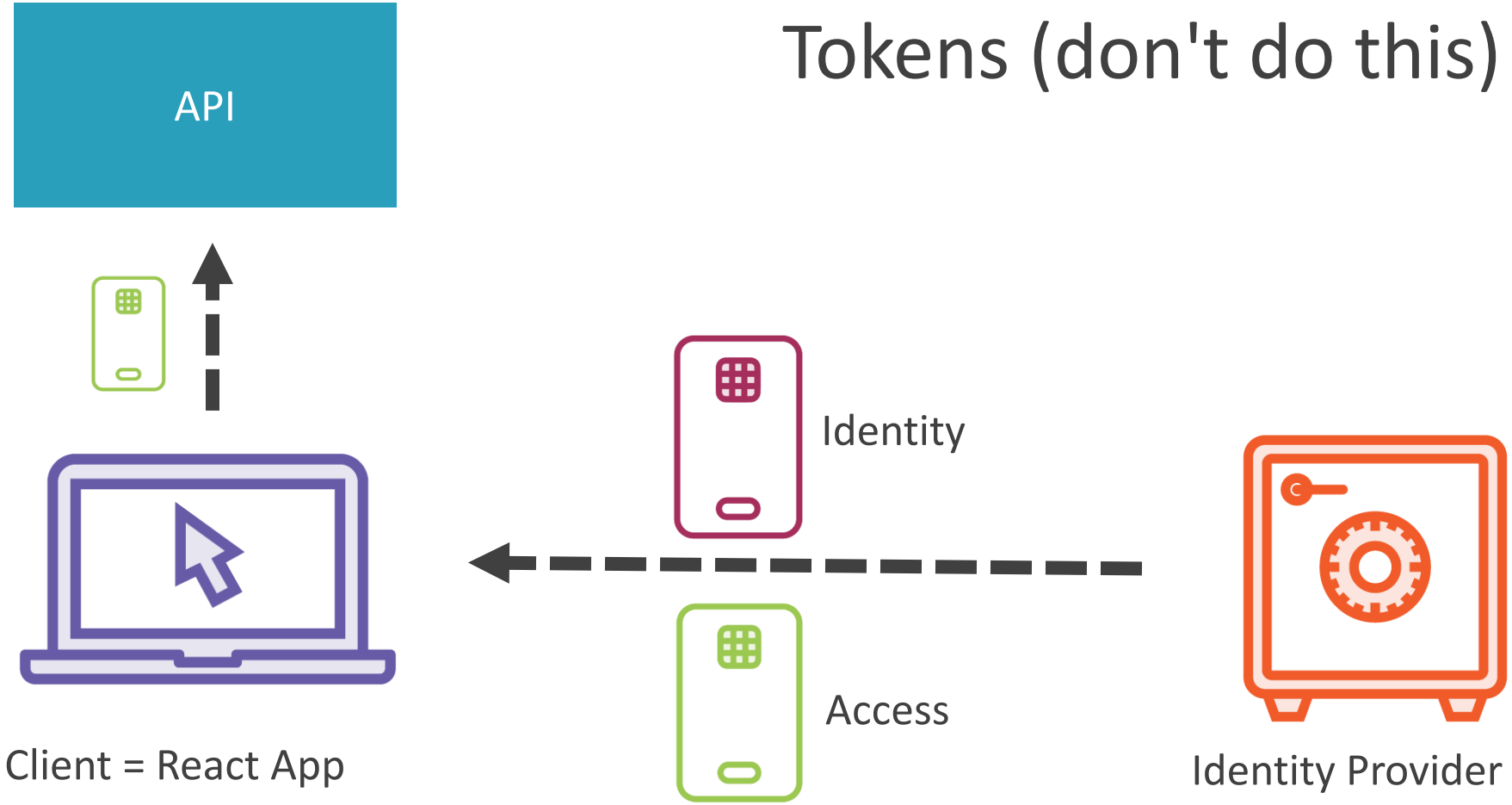


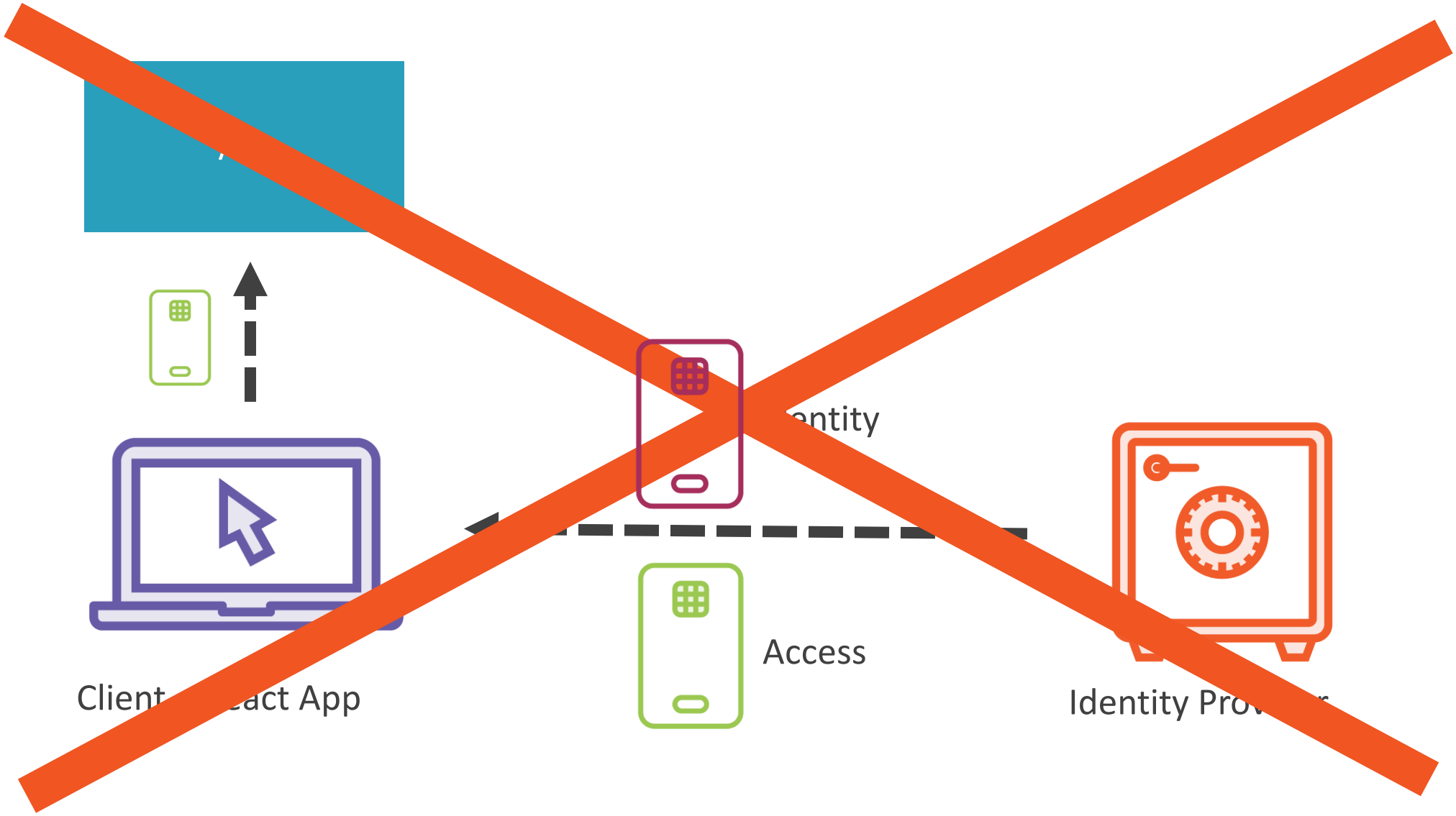
# Demo

Auth

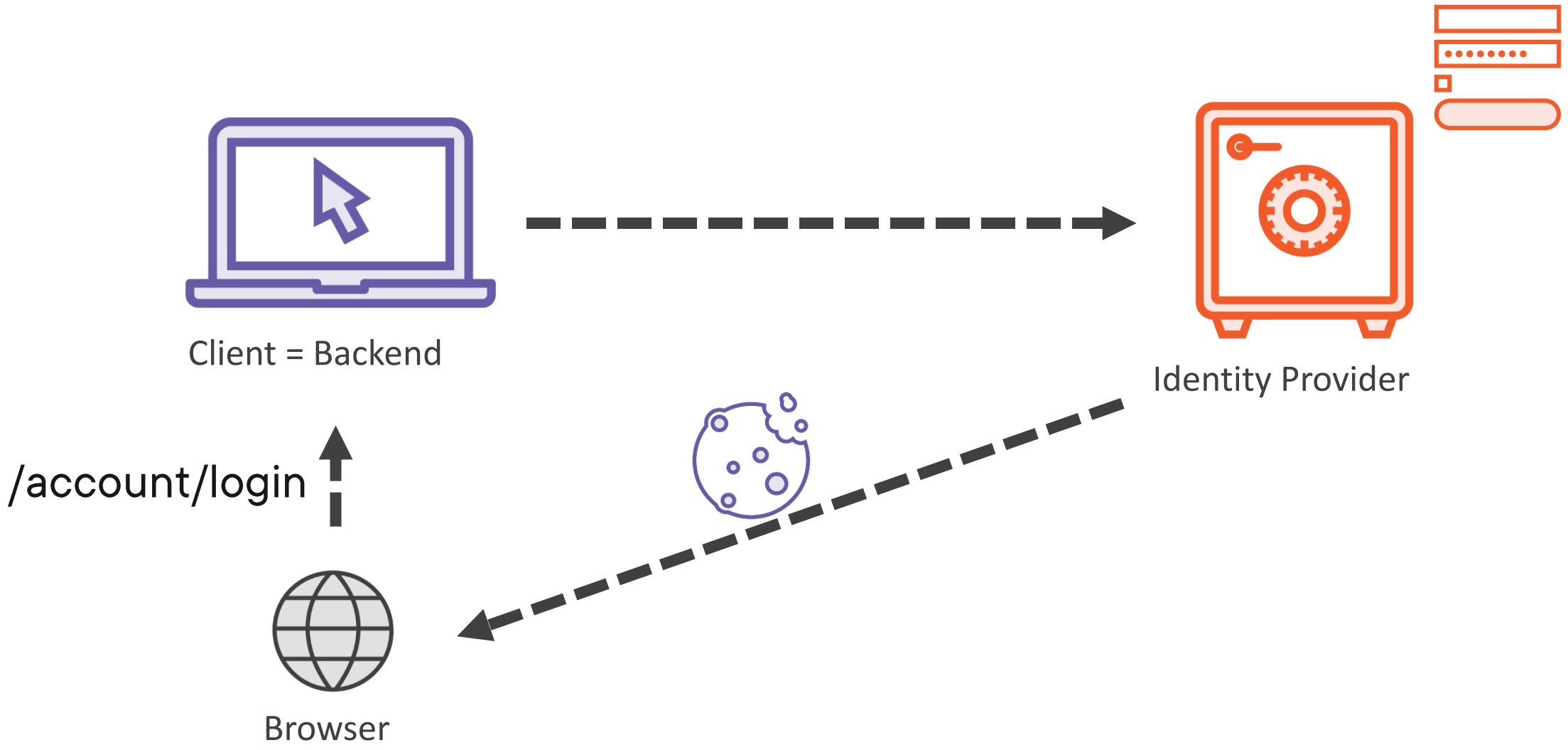
# Authentication with Blazor Wasm

# Tokens (don't do this)

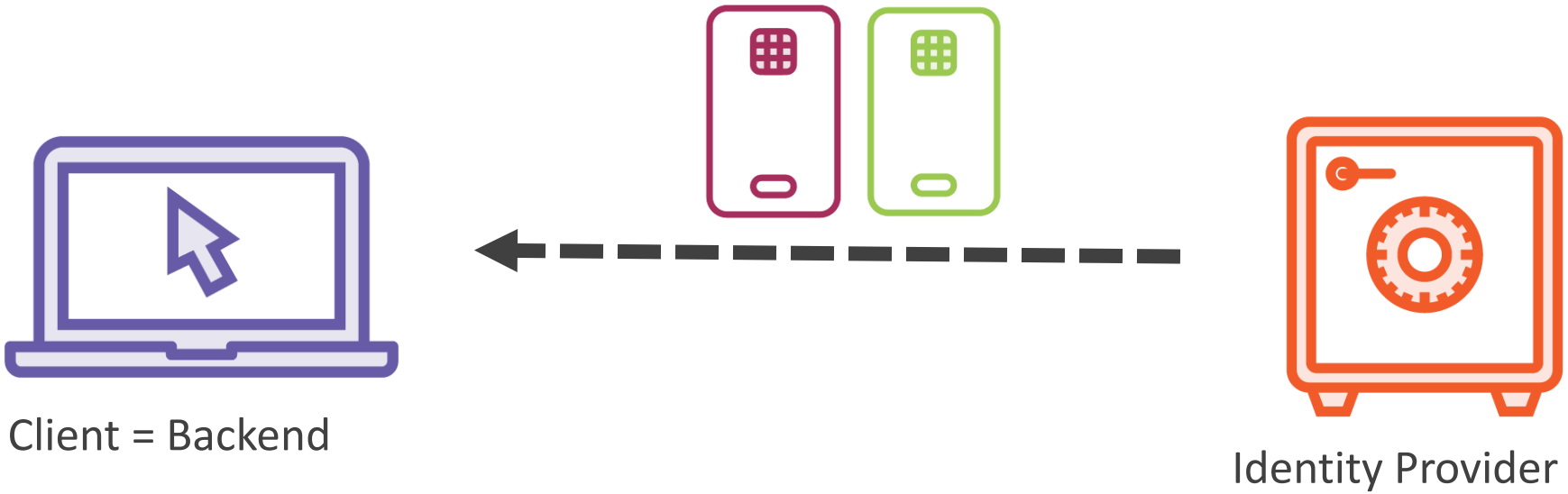



ntity
Access
Client
act App
Identity Pro

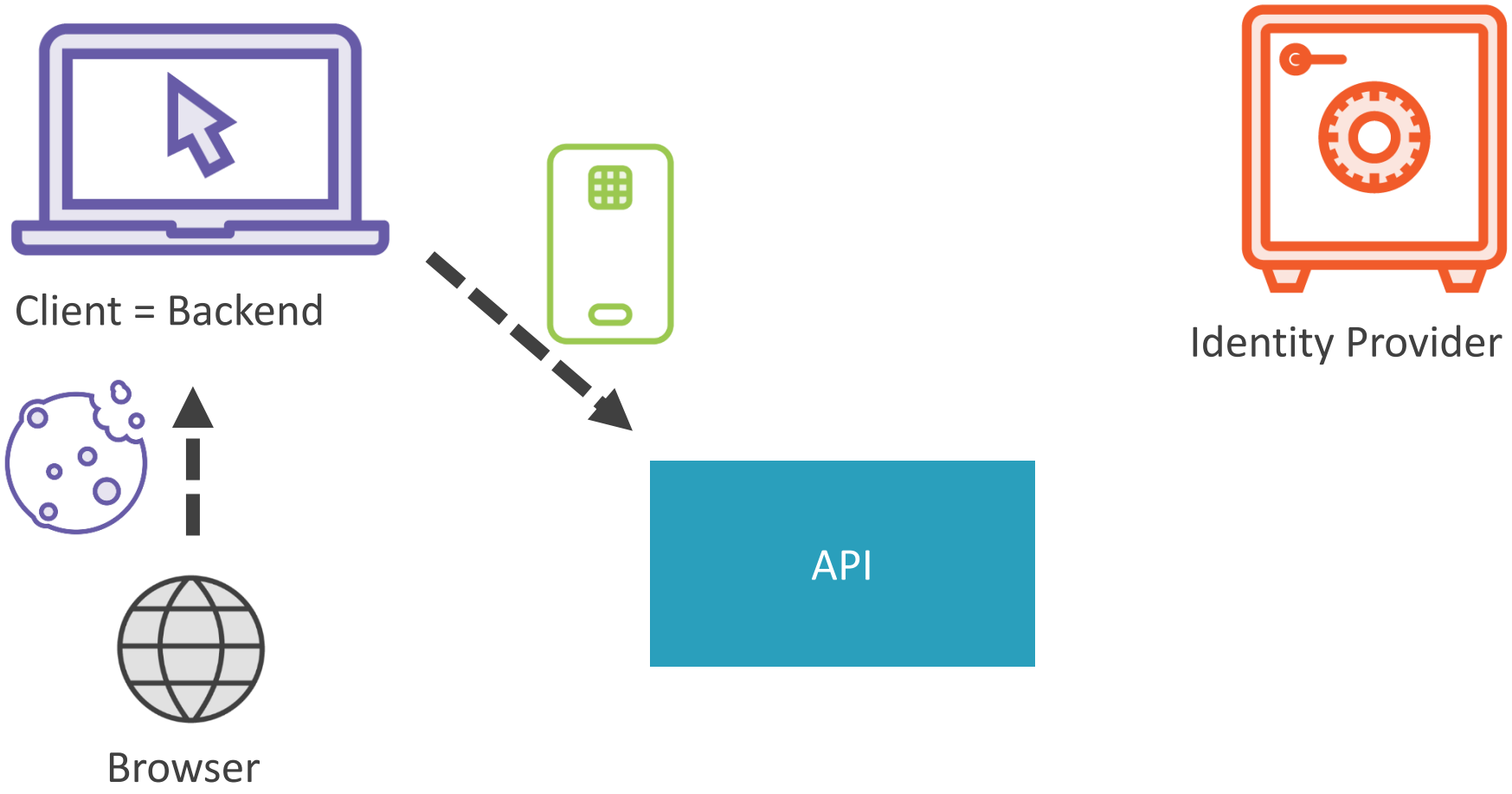
# OpenId Connect with BFF



# OpenId Connect with BFF



# OpenId Connect with BFF



# Lab time!
# Part 2

# Thanks


## Roland Guijt

MVP | CONSULTANT | TRAINER | AUTHOR

@rolandguijt rolandguijt.com roland.guijt@gmail.com

https://github.com/rolandguijt