

.NET Conf 2022



Shaun Walker

Creator of DotNetNuke (DNN)

Microsoft MVP

ASP Insider

Creator of Oqtane

Chair of .NET Foundation Project Committee



CTO

Professional Services



cognizant

@sbwalker



Aloha .NET MAUI!

Extending Apps To Mobile and Desktop
With Blazor Hybrid



Multi-Platform Application Development...



What Kind of **App Development** to **Choose**?



WEB APP

VS



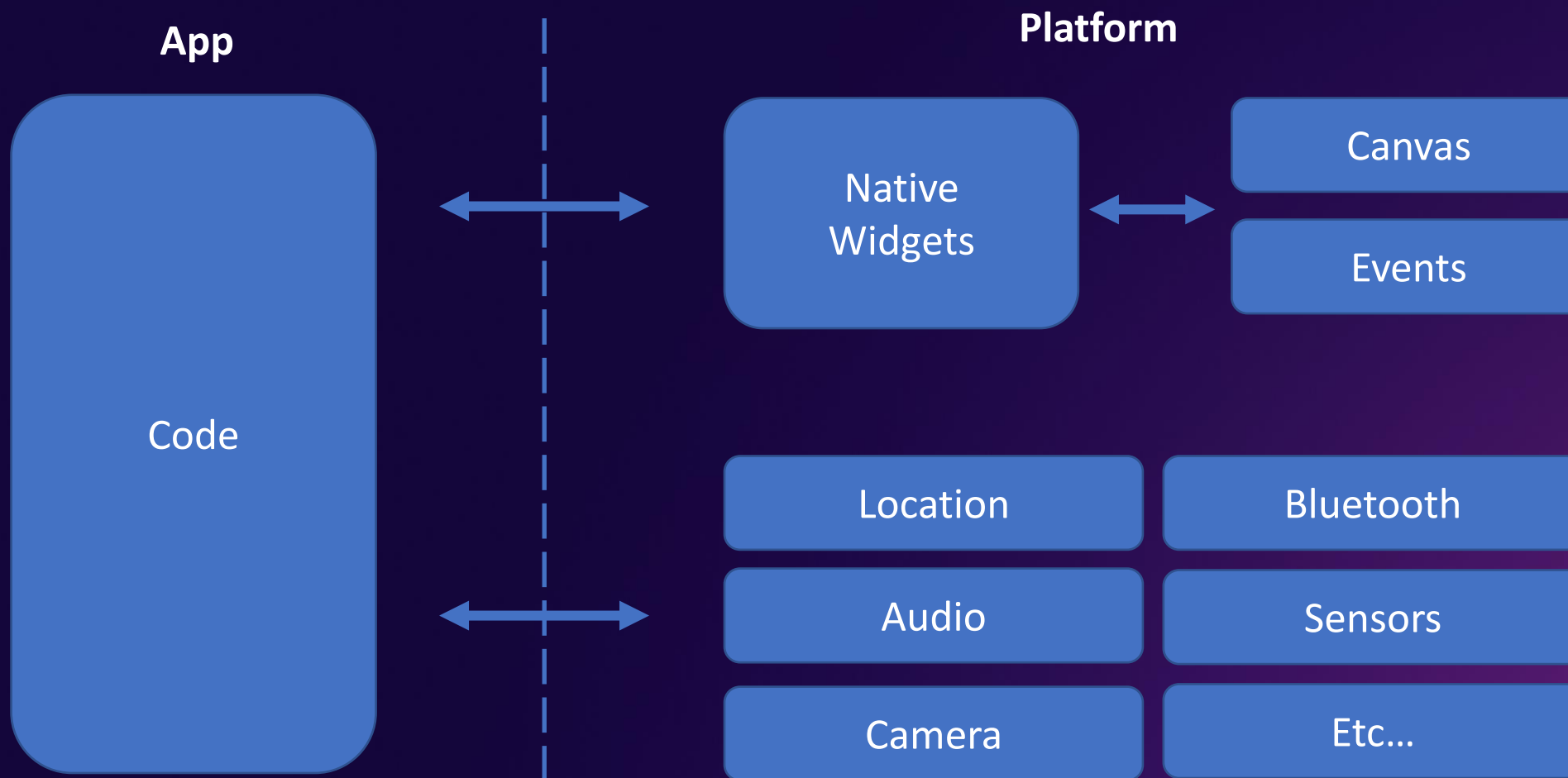
HYBRID APP

VS

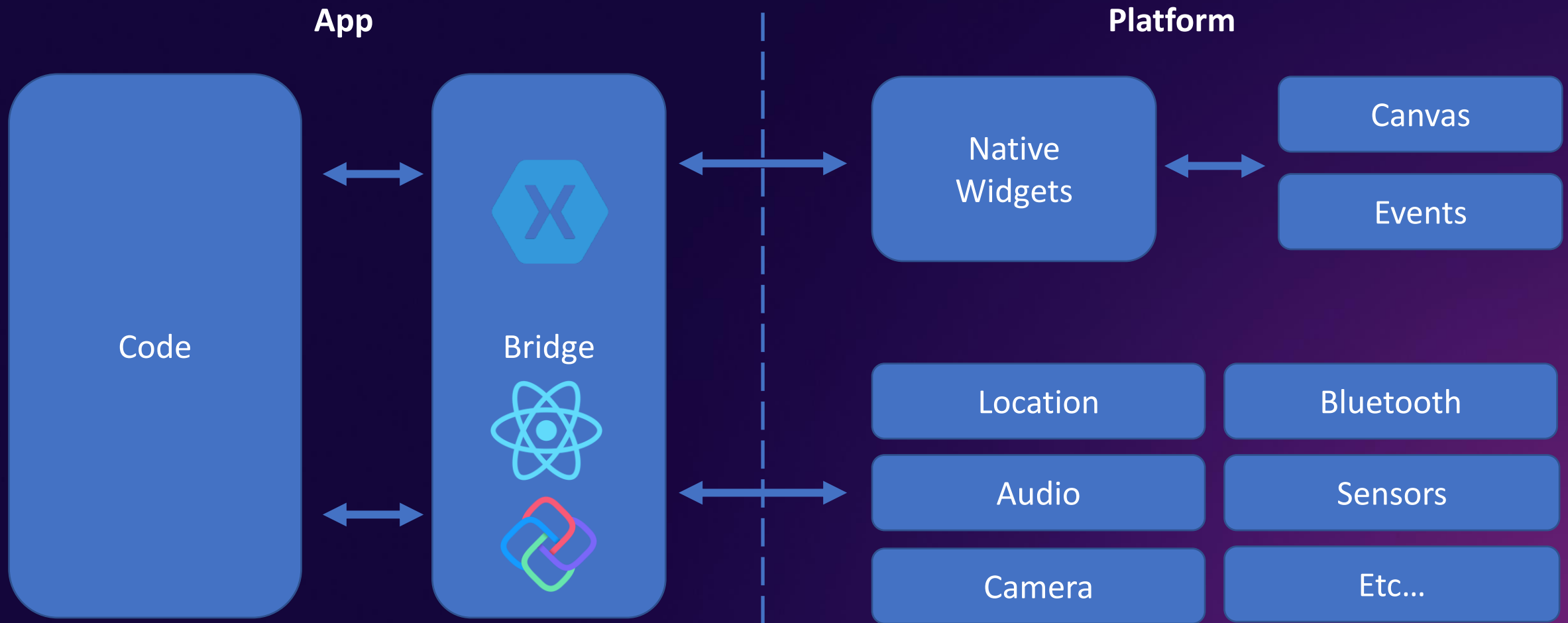


NATIVE APP

Pure Native App

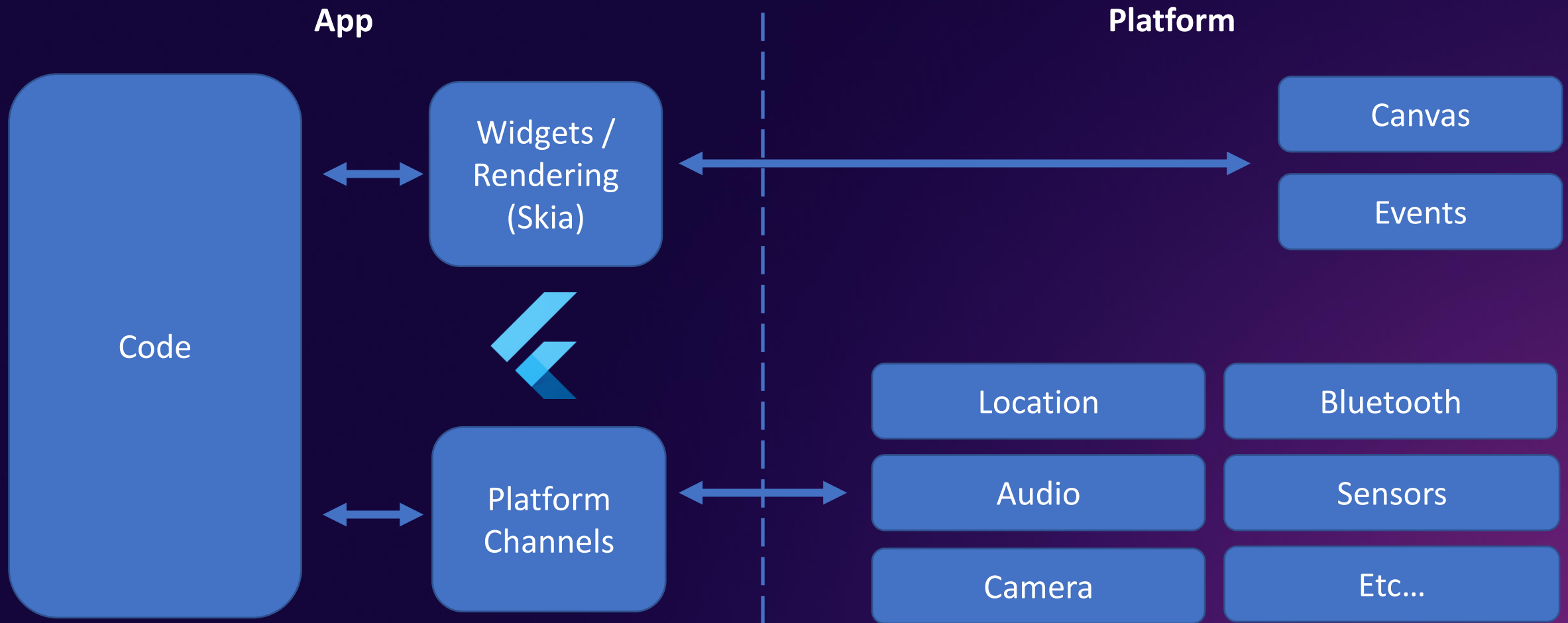


Xamarin, Uno Platform & React Native Apps

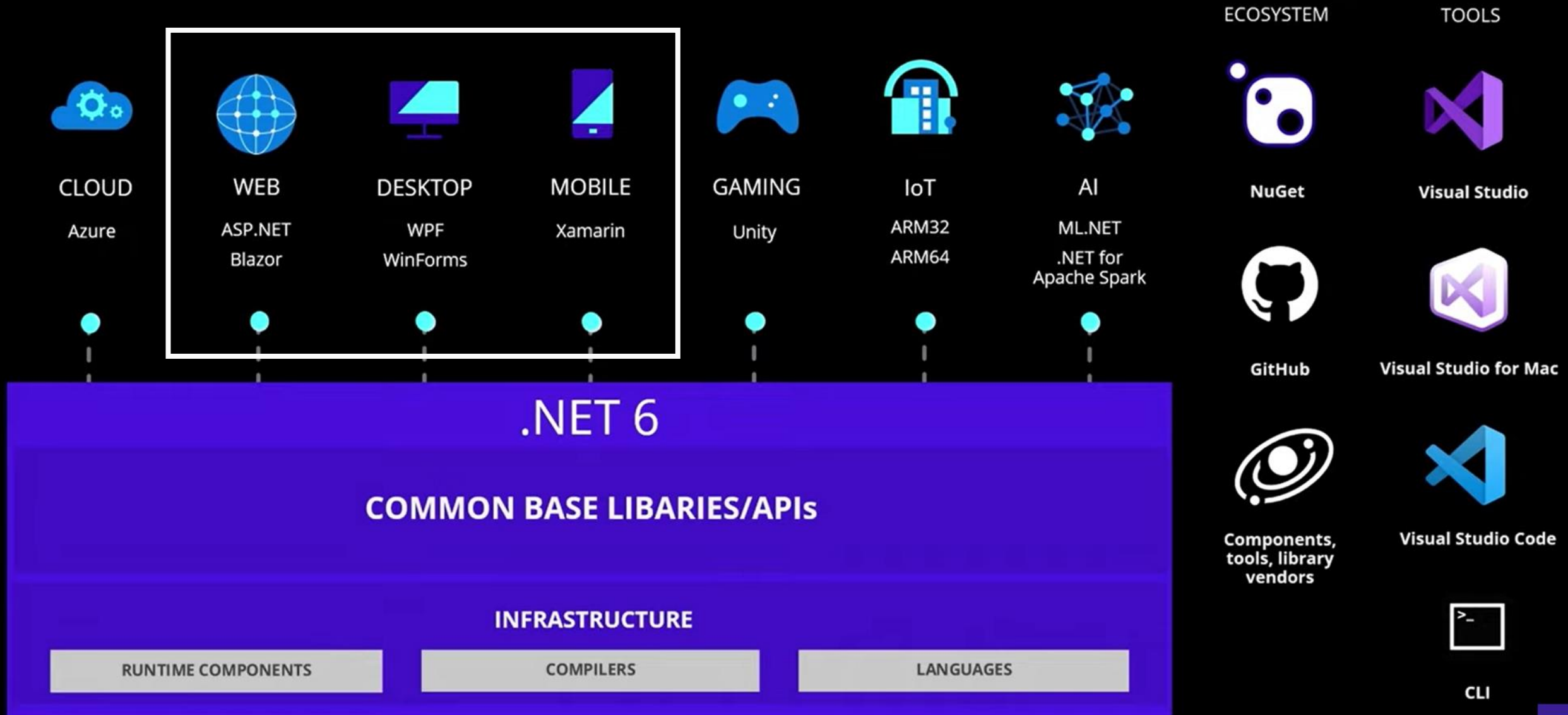


Flutter App

PROGRAMMERS'
WEEK 2022



.NET development platform



MOLOKA'I

MAUI MAP



LANAI



MOLOKINI CRATER



“.NET MAUI” **Multi-platform** **App UI**



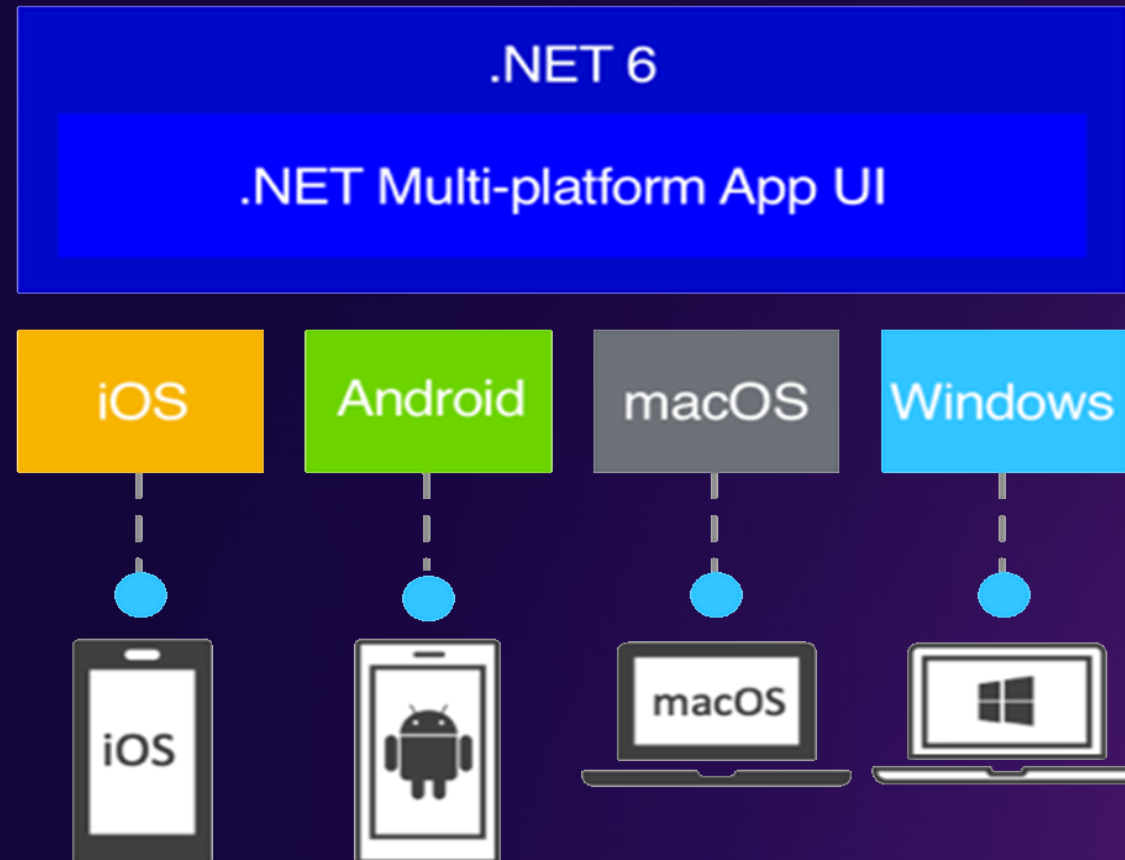
The Evolution of Xamarin...



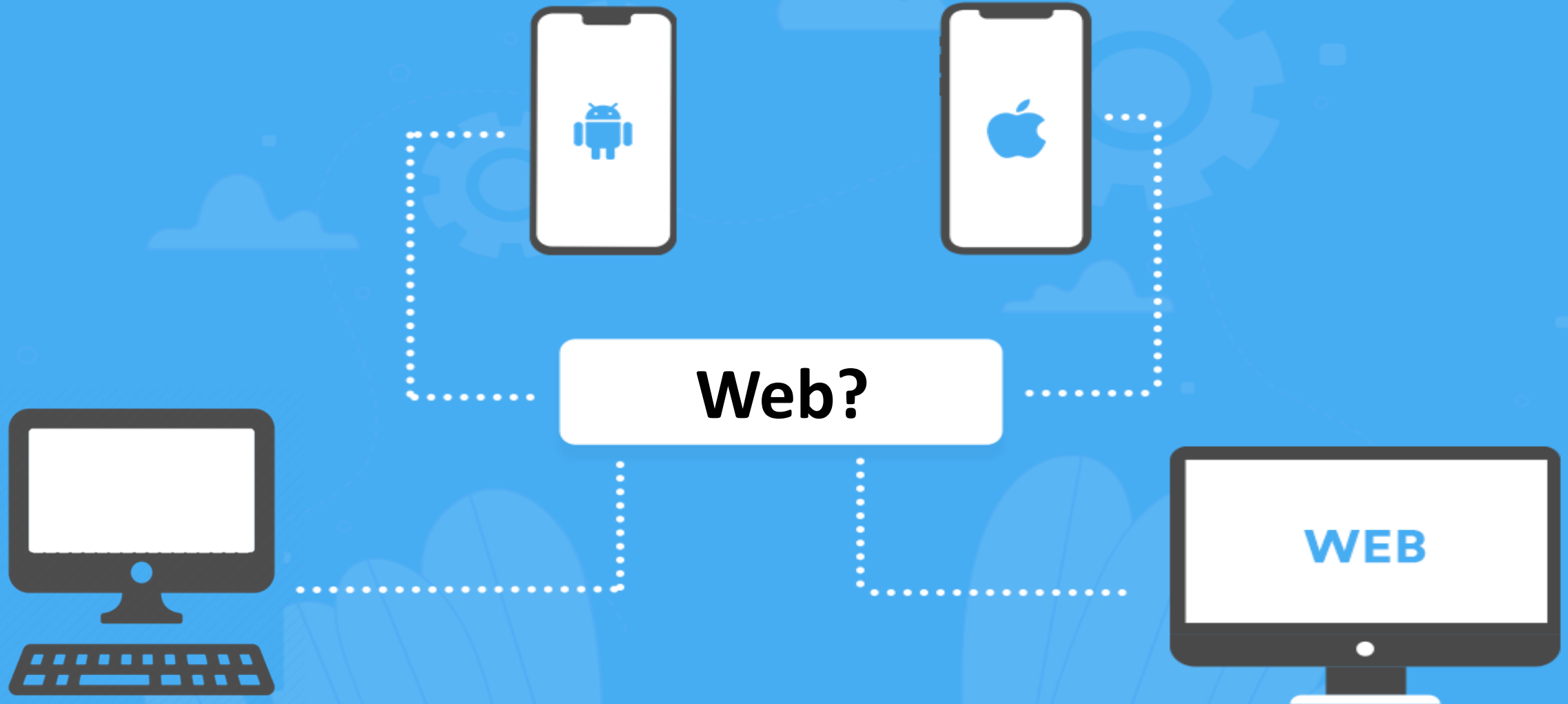
Released May 24, 2022 for General Availability (GA)
Fully supported for production workloads

.NET MAUI is a cross-platform framework for creating native mobile and desktop apps with C# and XAML

PROGRAMMERS'
WEEK 2022



But What About Web?



And How?

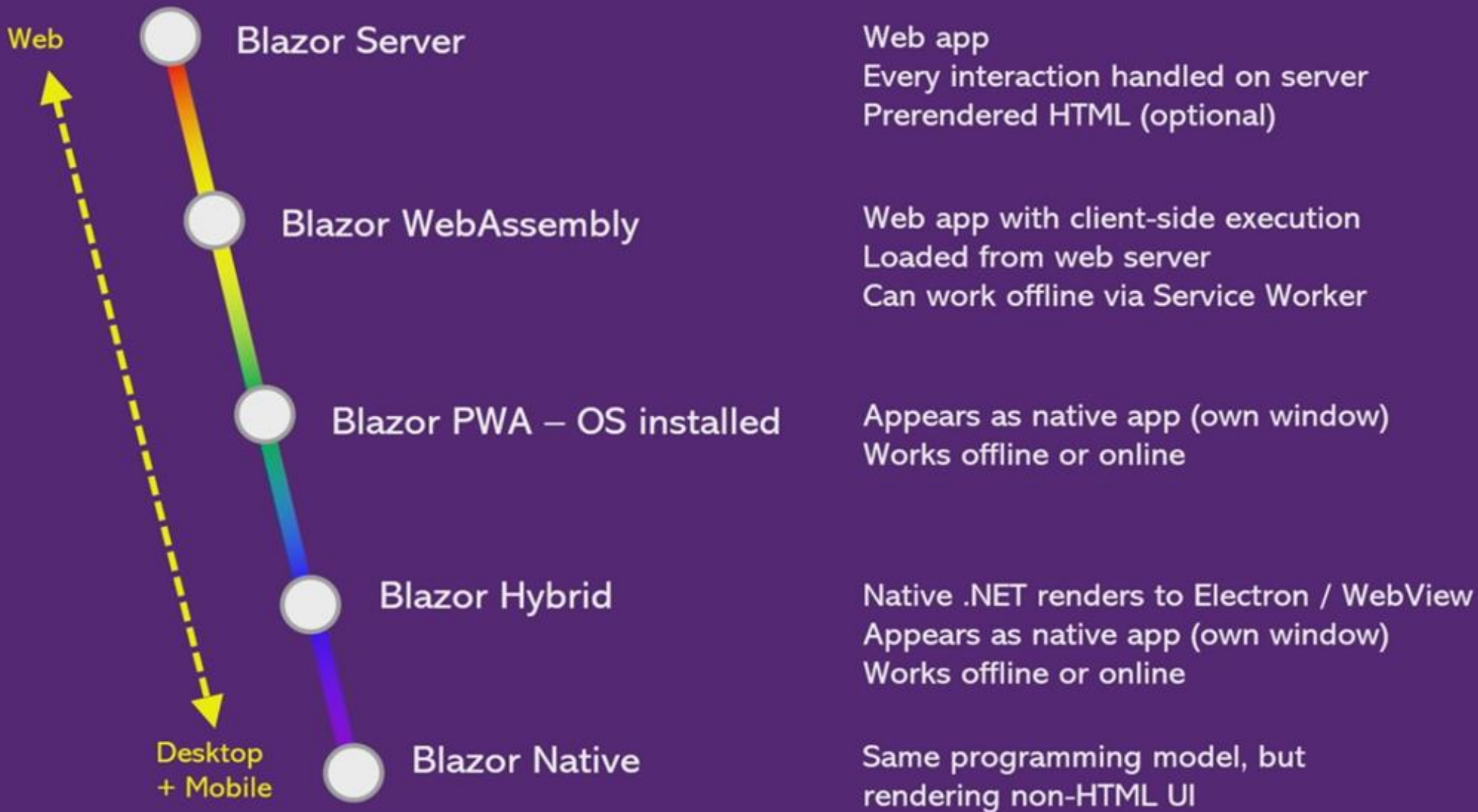


Blazor To The Rescue...

PROGRAMMERS'
WEEK 2022



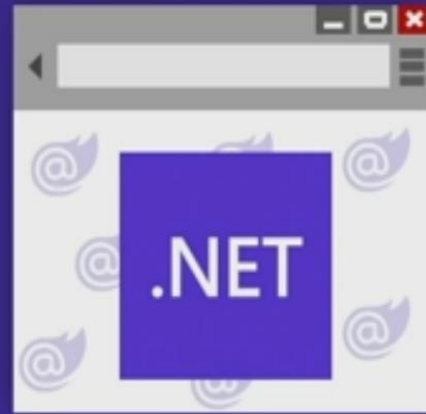
.NET BLAZOR



Modern web UI with .NET & Blazor



Server

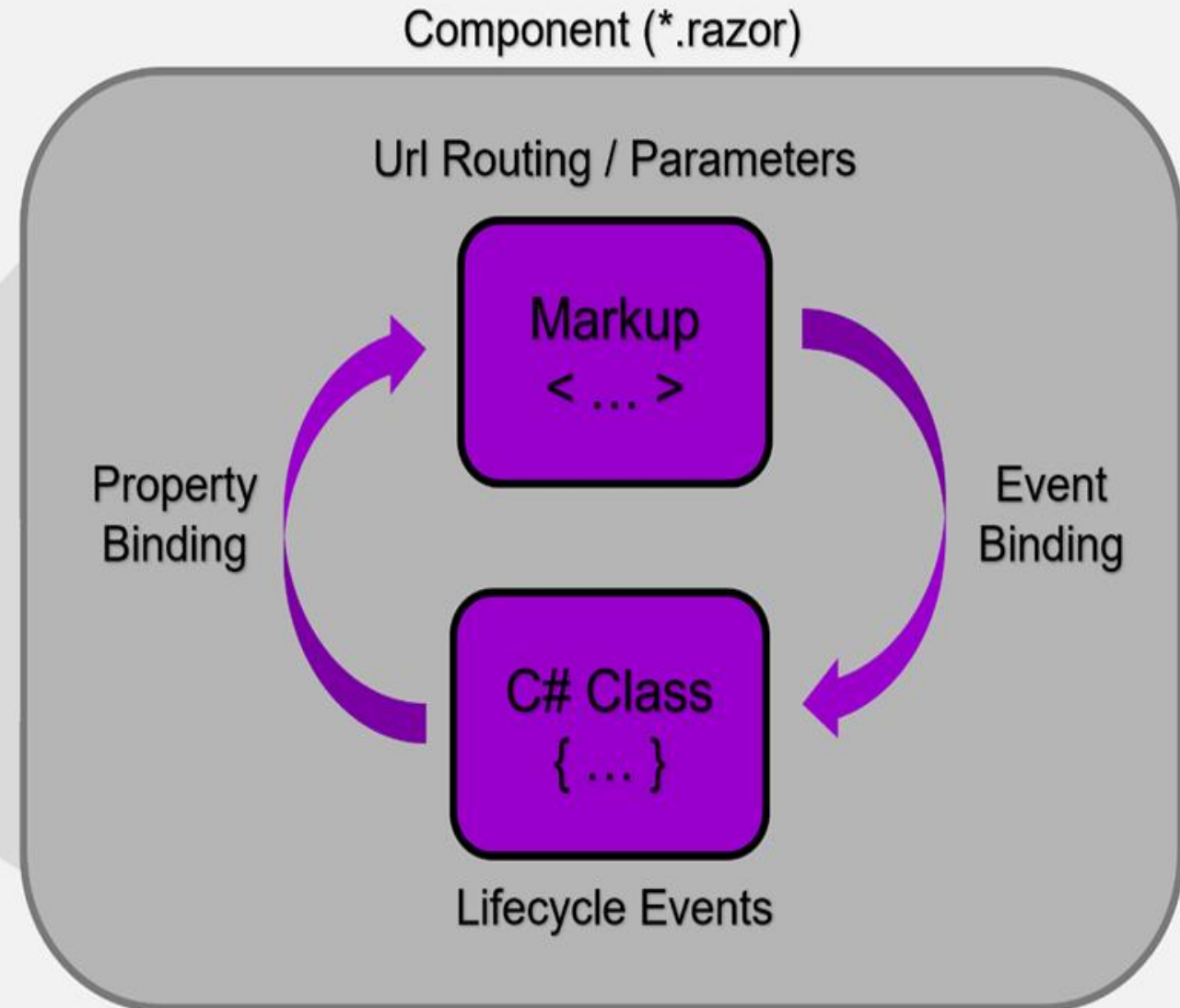
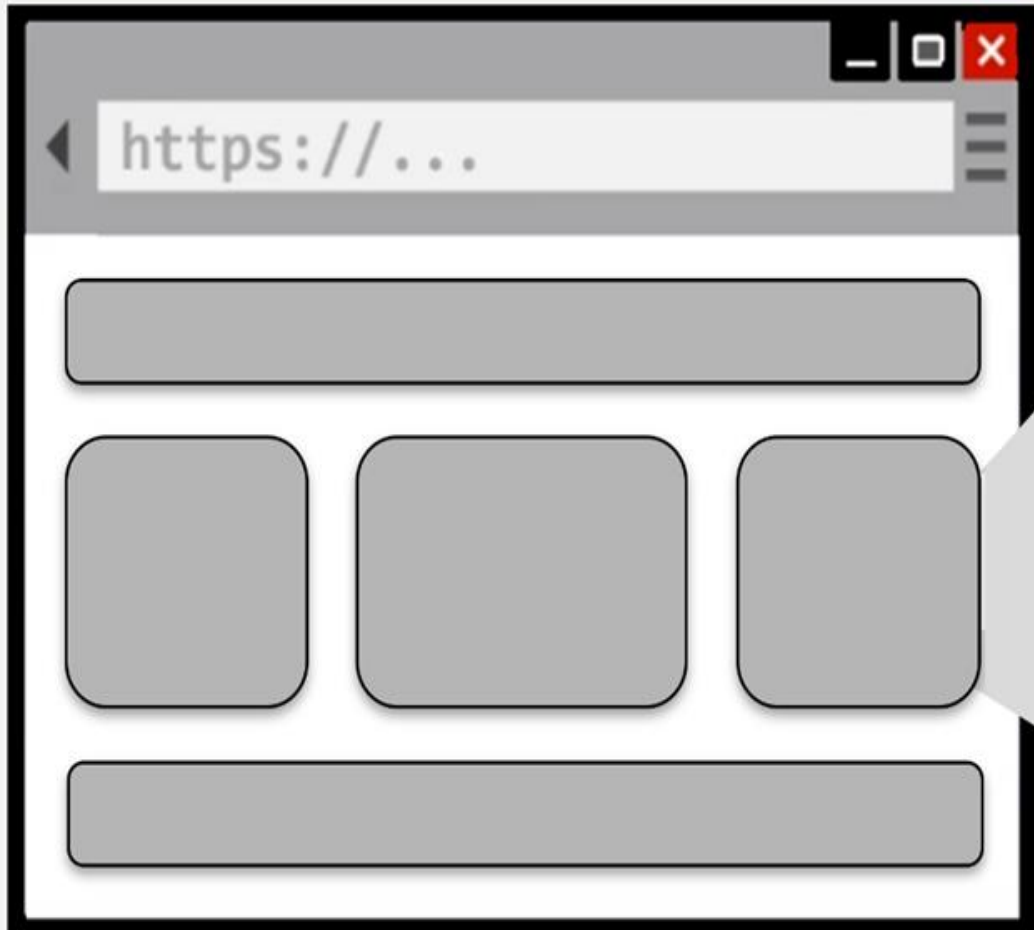


WebAssembly



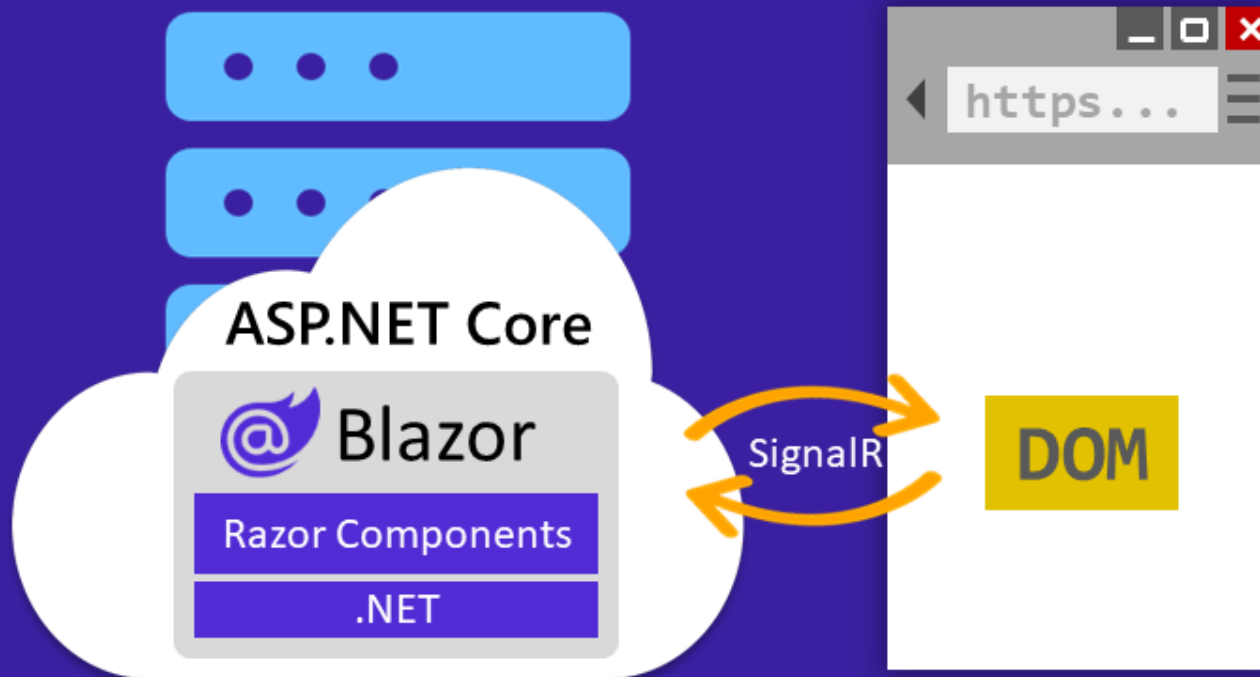
Hybrid

Razor Components

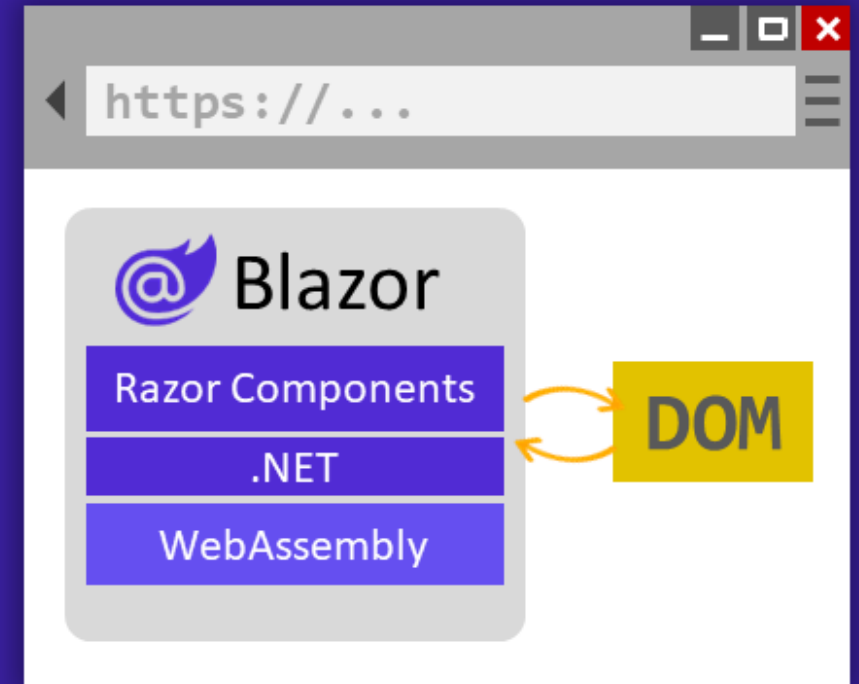


Blazor Server & Blazor WebAssembly

Blazor Server



Blazor WebAssembly



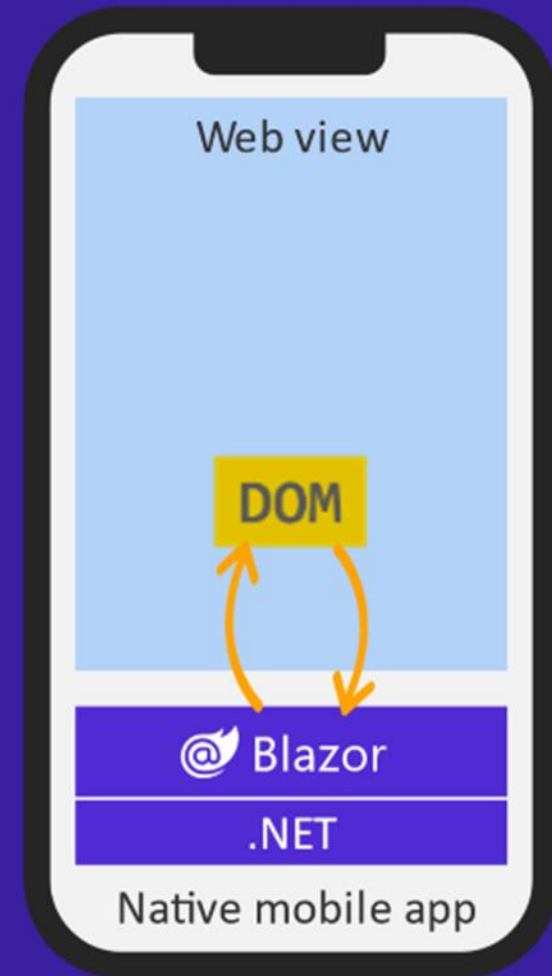
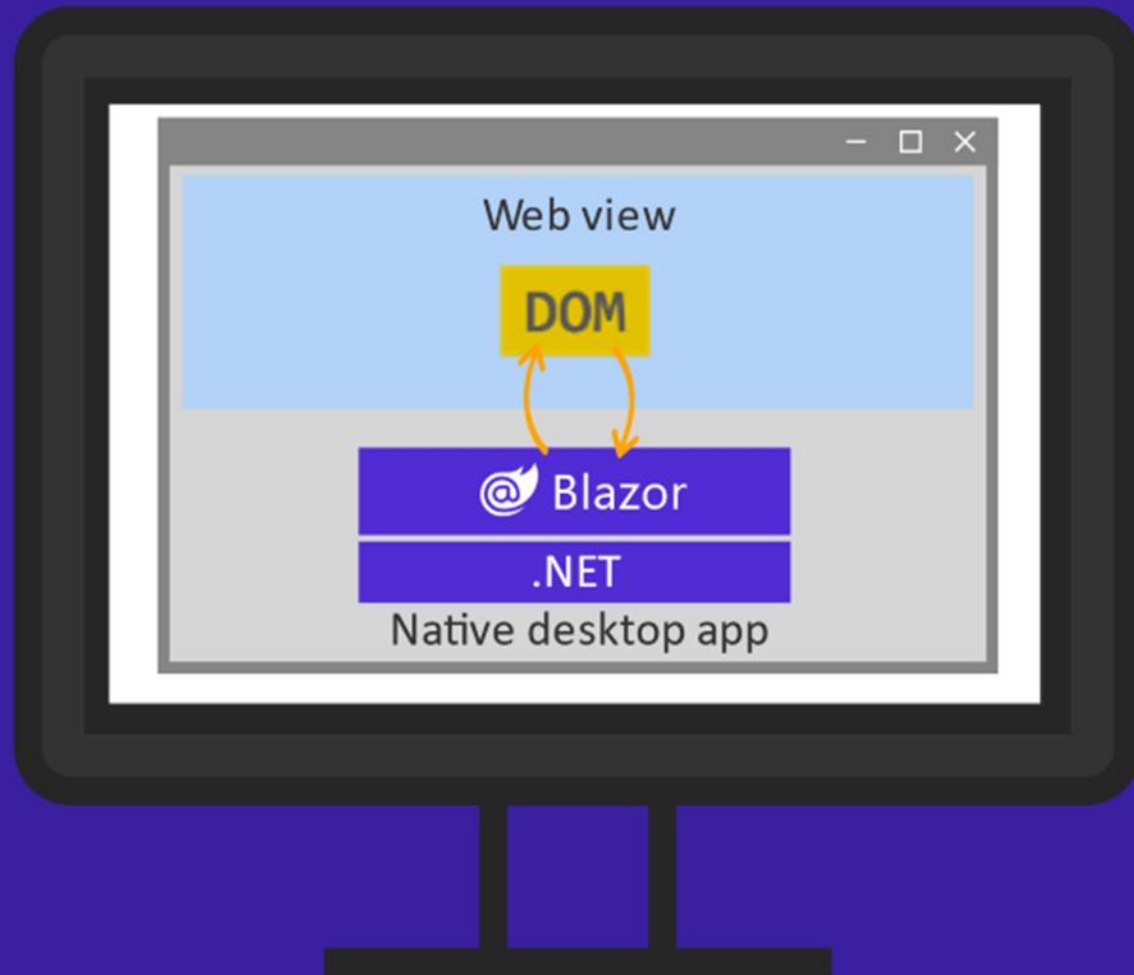
Mobile Blazor Bindings

Build **native** and **hybrid** cross-platform apps using Blazor.

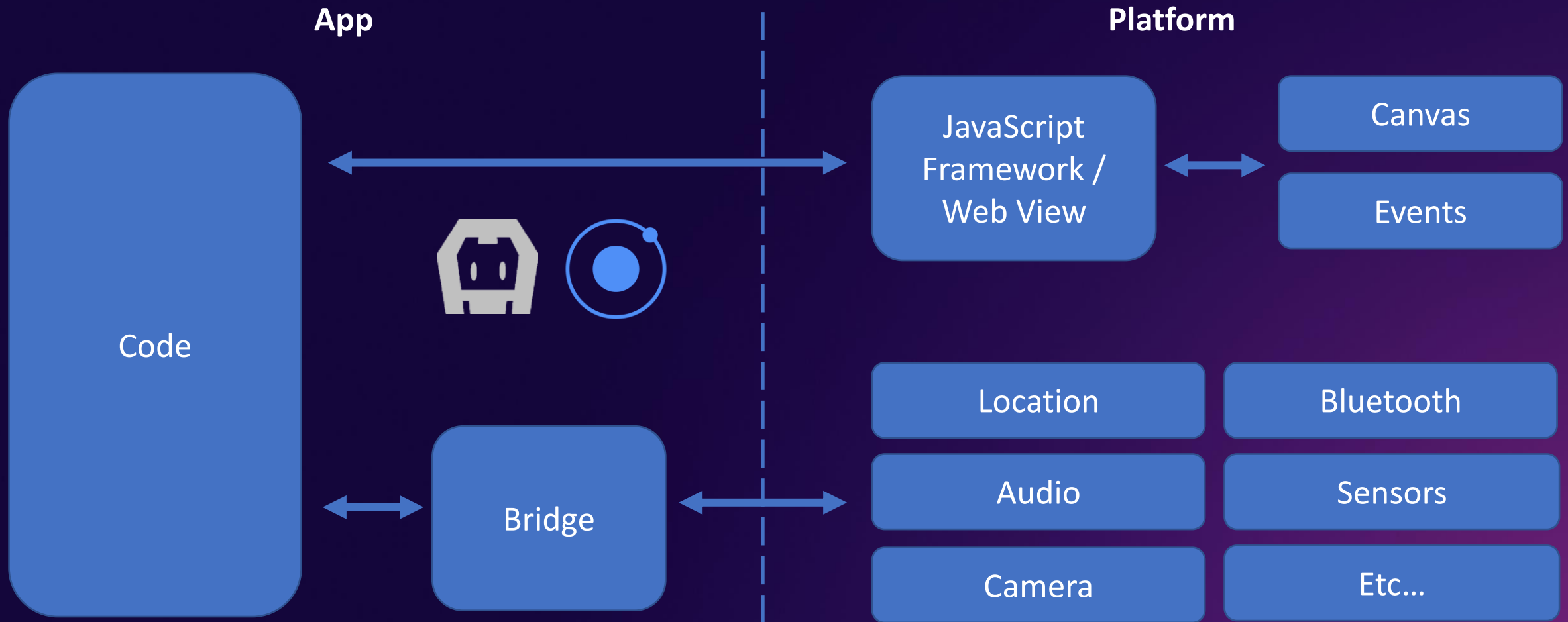
Run on iOS, Android, Windows, and macOS.



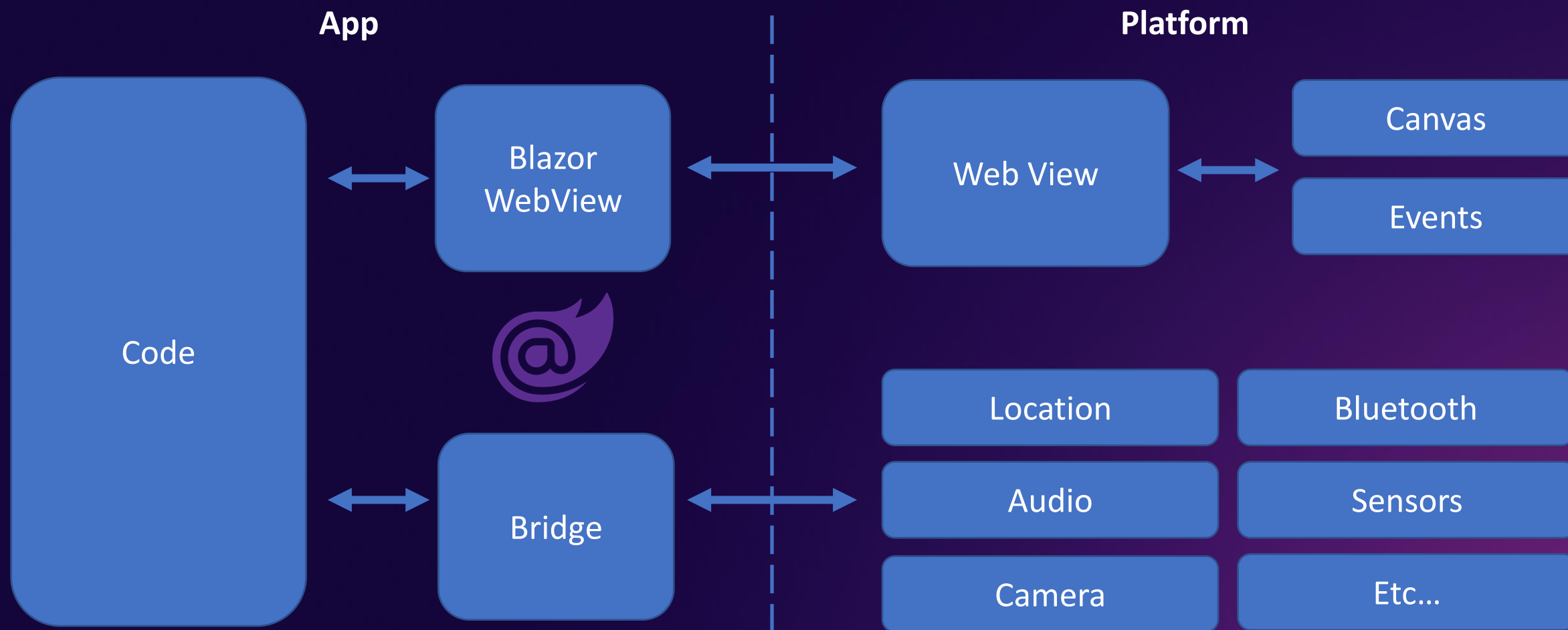
Blazor Hybrid for Desktop & Mobile!



Traditional Hybrid Web View App (Cordova, Ionic)



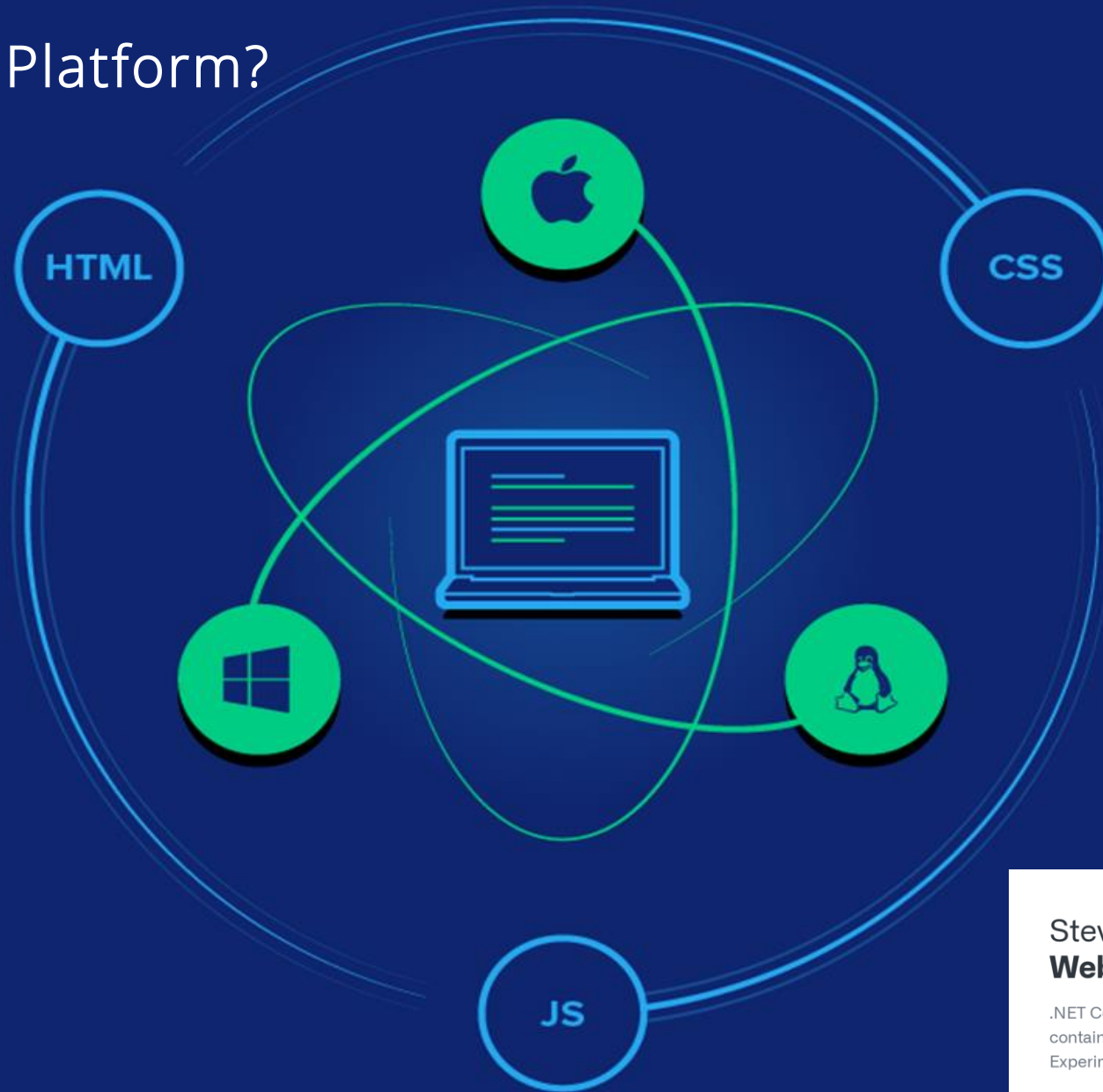
Blazor Hybrid App



Blazor Hybrid Windows Desktop (Microsoft Edge WebView2)



What about Cross Platform?



SteveSandersonMS/ **WebWindow**



.NET Core library to open native OS windows containing web UI on Windows, Mac, and Linux. Experimental.

6
Contributors

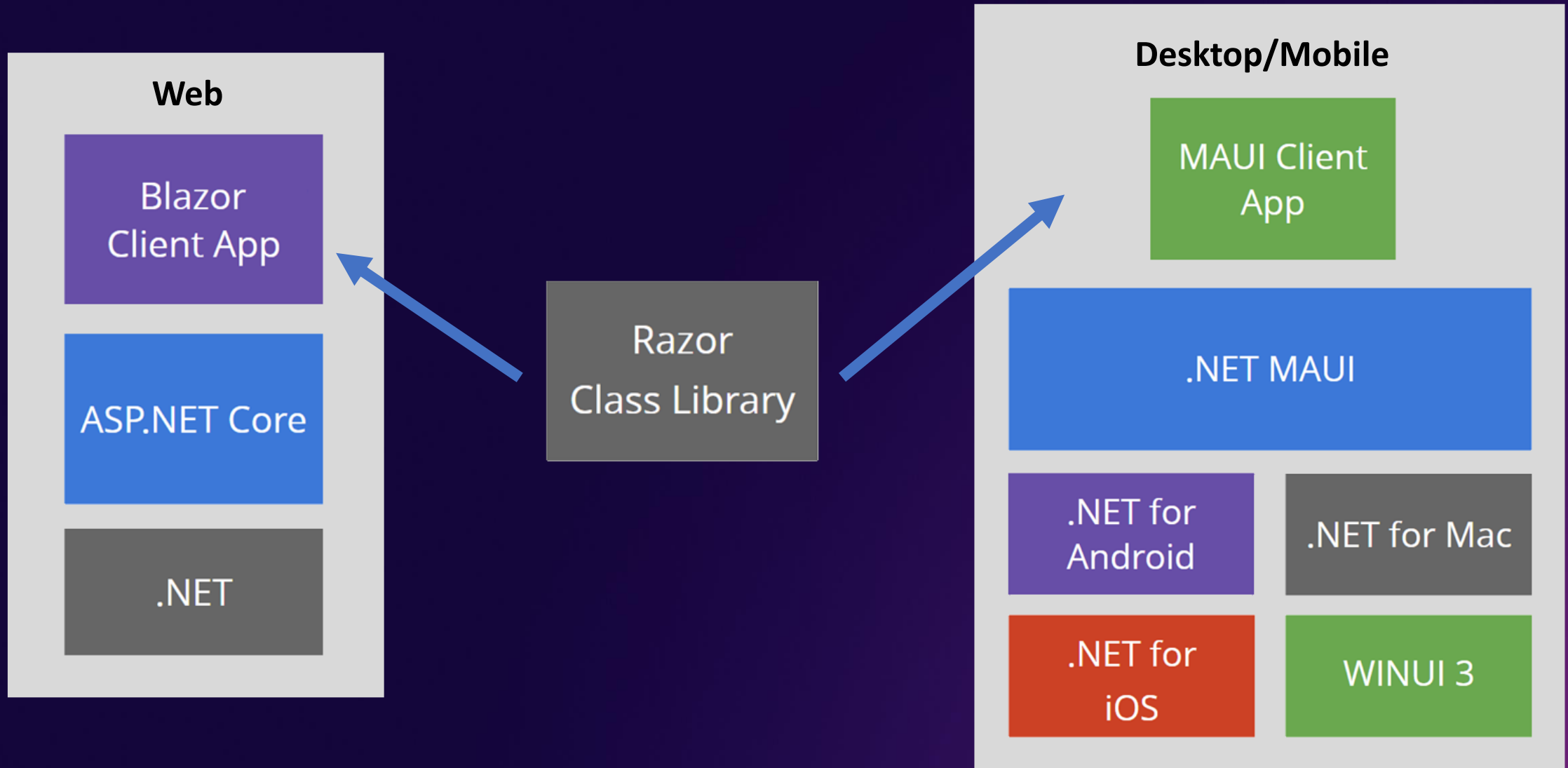
38
Used by

2k
Stars

216
Forks



Superior Code Reusability



What Are The Tradeoffs?

Native

- Native UX
- Better Performance
- Hardware Integration



Hybrid

- Reusable Code
- Faster Development
- Less Expensive

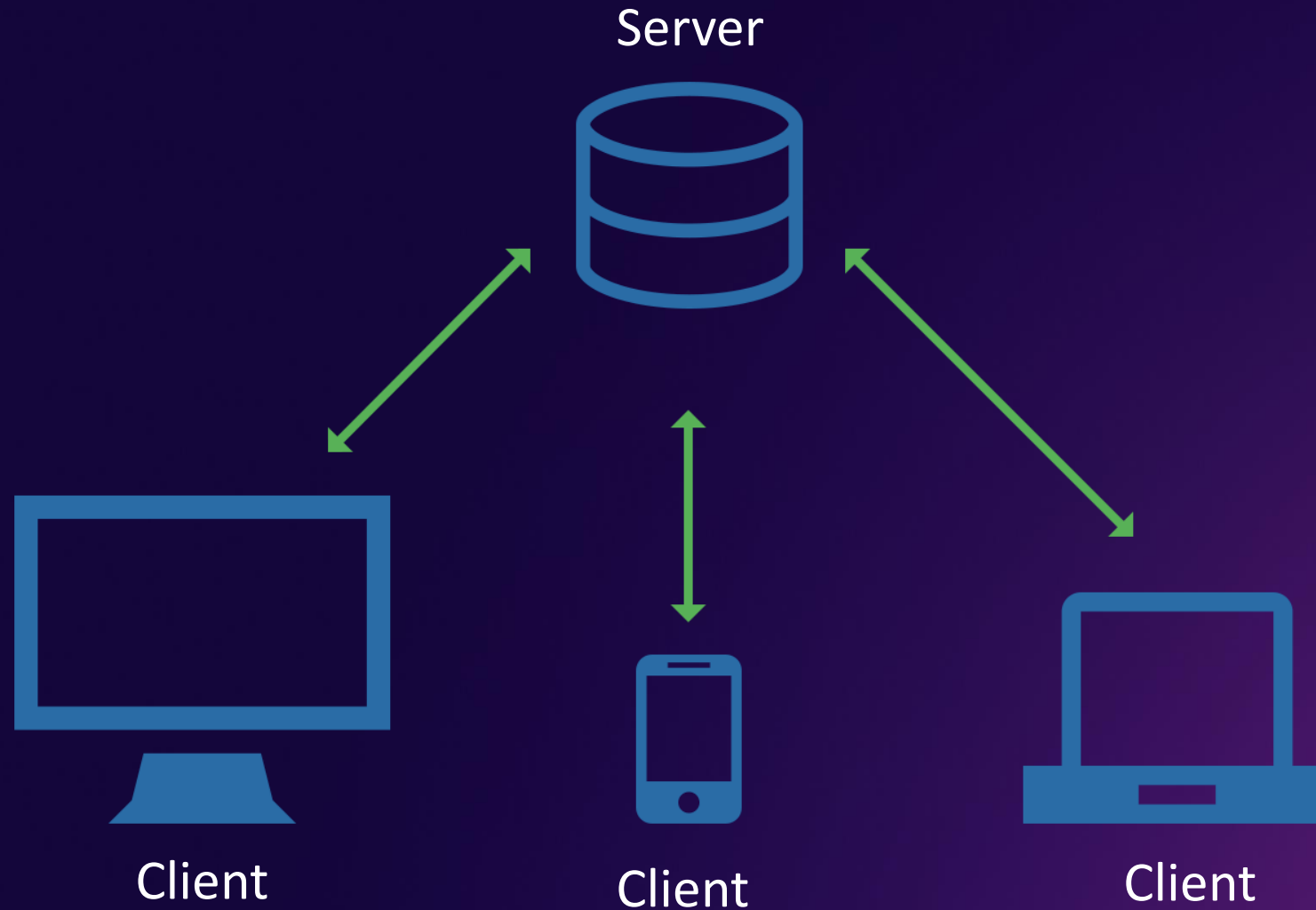
Demo

oqtane

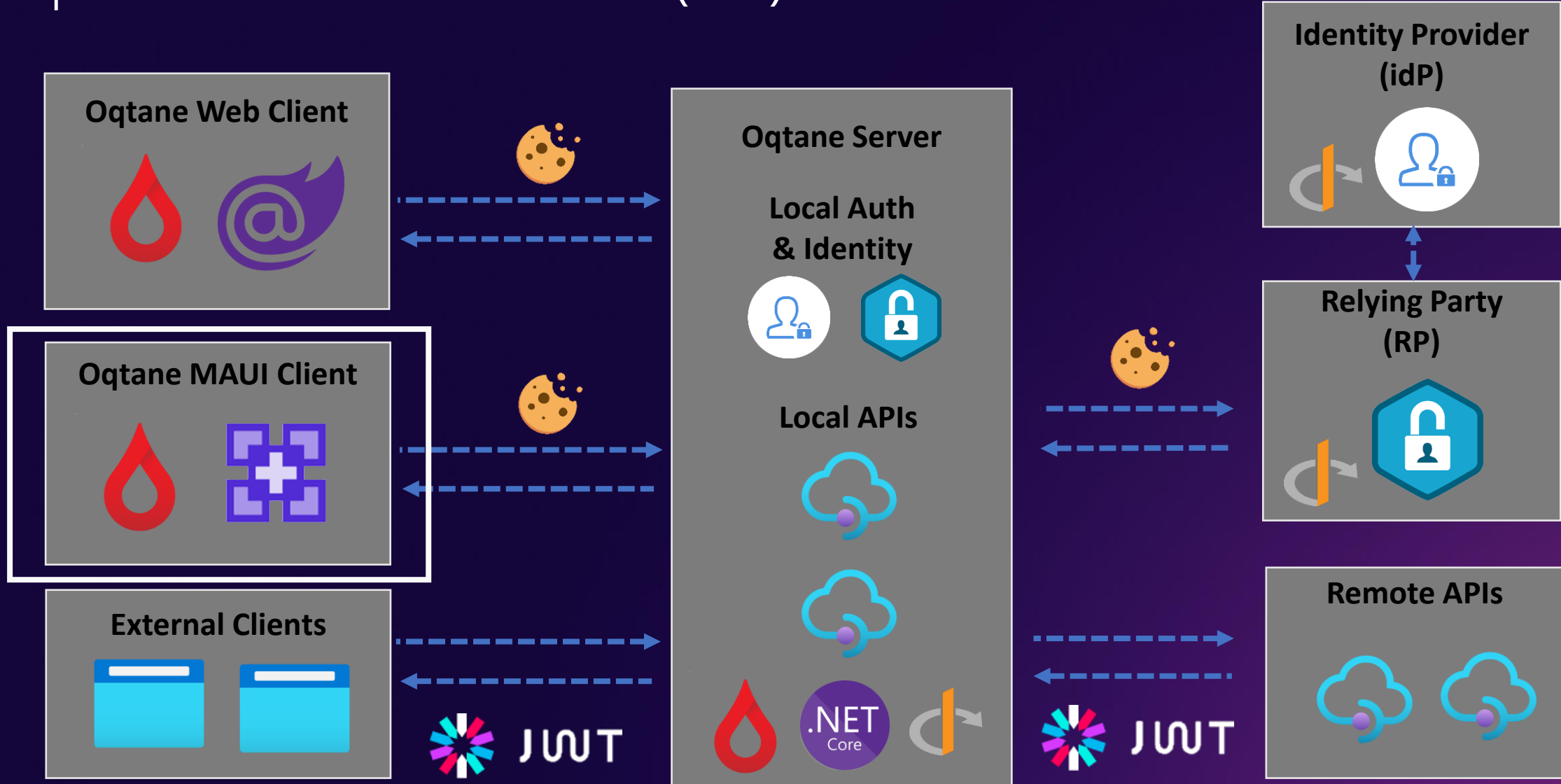
Blazor Application

- Open source modular application framework
- 260,000 lines of code
- 100 razor components
- 30 API services
- Custom router
- Plug-in modules and themes
- Localization
- Runs on Blazor Server & WebAssembly

Client/Server Architecture Is Critical!



Oqtane Backend-for-Frontend (BFF) Architecture



Blazor Hybrid Client Application

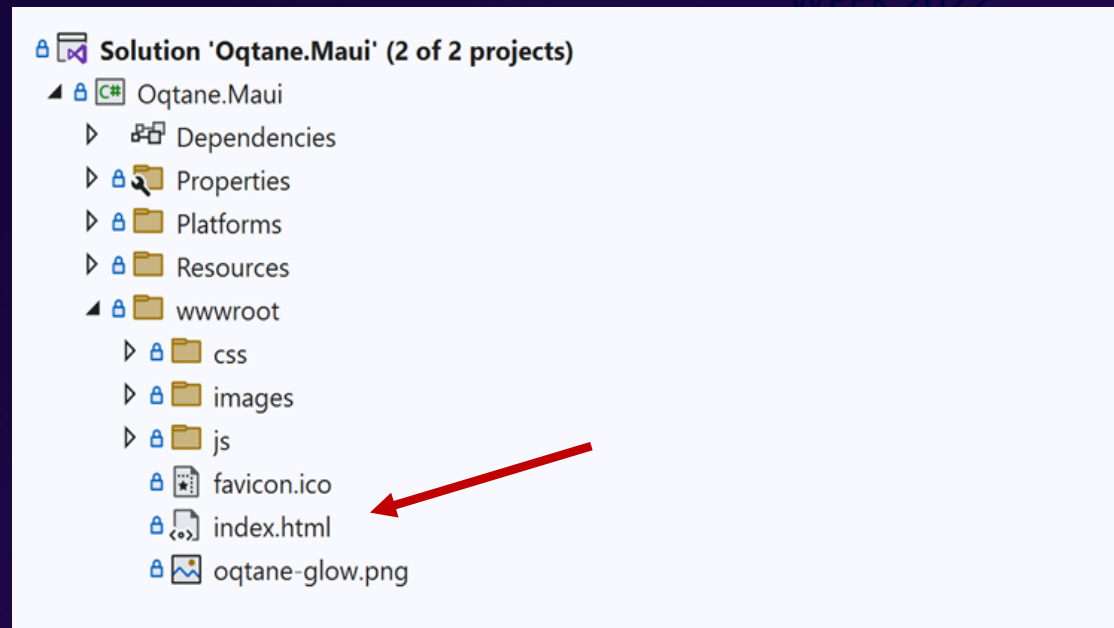
- Scaffolded Blazor Hybrid application using template
- **MainPage.xaml** contains RootComponent
- Defines the **HostPage**
- **Main.razor** can use DynamicComponent to reference a component in another assembly and pass parameters

```
<BlazorWebView HostPage="wwwroot/index.html">  
  <BlazorWebView.RootComponents>  
    <RootComponent Selector="#app" ComponentType="{x:Type local:Main}" />  
  </BlazorWebView.RootComponents>  
</BlazorWebView>
```

```
<DynamicComponent Type="@ComponentType" Parameters="@Parameters"></DynamicComponent>  
  
@code {  
    Type ComponentType = Type.GetType("Oqtane.App, Oqtane.Client");  
    private IDictionary<string, object> Parameters { get; set; }  
  
    protected override void OnInitialized()  
    {  
        Parameters = new Dictionary<string, object>();  
        Parameters.Add(new KeyValuePair<string, object>("AntiForgeryToken", ""));  
        Parameters.Add(new KeyValuePair<string, object>("Runtime", "Hybrid"));  
        Parameters.Add(new KeyValuePair<string, object>("RenderMode", "Hybrid"));  
        Parameters.Add(new KeyValuePair<string, object>("VisitorId", -1));  
        Parameters.Add(new KeyValuePair<string, object>("RemoteIPAddress", ""));  
        Parameters.Add(new KeyValuePair<string, object>("AuthorizationToken", ""));  
    }  
}
```

Blazor Hybrid Client Application

- Client application bootstrapped by local **index.html**
- Includes script **blazor.webview.js**
- Local static assets must be packaged with client application when deployed



```
<body>

  <div class="status-bar-safe-area"></div>

  <div id="app">Loading...</div>

  <div id="blazor-error-ui">
    An unhandled error has occurred.
    <a href="" class="reload">Reload</a>
    <a class="dismiss">X</a>
  </div>

  <script src="js/interop.js"></script>

  <script src="_framework/blazor.webview.js" autostart="false"></script>

</body>
```


Blazor Hybrid Client Application

- **MauiProgram.cs** needs to configure HttpClient with a BaseAddress
- **MauiProgram.cs** need to register all API services


```
public static MauiApp CreateMauiApp()
{
    var builder = MauiApp.CreateBuilder();
    builder
        .UseMauiApp<App>()
        .ConfigureFonts(fonts =>
        {
            fonts.AddFont("OpenSans-Regular.ttf", "OpenSansRegular");
        });

    builder.Services.AddMauiBlazorWebView();
    #if DEBUG
    builder.Services.AddBlazorWebViewDeveloperTools();
    #endif

    var httpClient = new HttpClient { BaseAddress = new Uri(url) };
    httpClient.DefaultRequestHeaders.UserAgent.ParseAdd(Shared.Constants.MauiUserAgent);
    builder.Services.AddSingleton(httpClient);
```



```
// register scoped core services
builder.Services.AddOqtaneScopedServices();
```



Blazor Hybrid Client Application

- **MauiProgram.cs** does not support async Main()
- Oqtane Client needs to download assemblies from server and load them into App Domain
- Had to use a synchronous Get method

```
private static void LoadClientAssemblies(HttpClient http)
{
    try
    {
        // get list of loaded assemblies on the client
        var assemblies = AppDomain.CurrentDomain.GetAssemblies()
            .Select(a => a.GetName().Name).ToList();

        // get assemblies from server and load into client app domain
        var zip = http.GetByteArrayAsync("/api/Installation/load").Result;
    }
}
```


Shared Component Library

- Blazor Hybrid client runs as IP **0.0.0.0**
- Remote static asset references in component markup (ie. images, CSS, JavaScript) must use **absolute paths**
- Oqtane uses the **User agent** used to distinguish hosting models

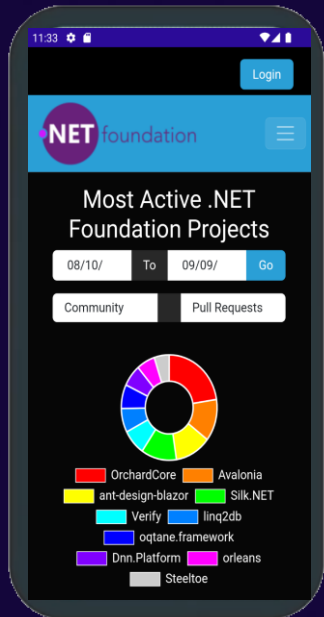
```
if (alias != null)
{
    alias.Protocol = (httpcontext.Request.IsHttps) ? "https://" : "http://";
    alias.BaseUrl = "";
    if (httpcontext.Request.Headers.ContainsKey("User-Agent") &&
        httpcontext.Request.Headers["User-Agent"] == Shared.Constants.MauiUserAgent)
    {
        alias.BaseUrl = alias.Protocol + alias.Name;
    }
    _siteState.Alias = alias;
}
```

Shared Component Library

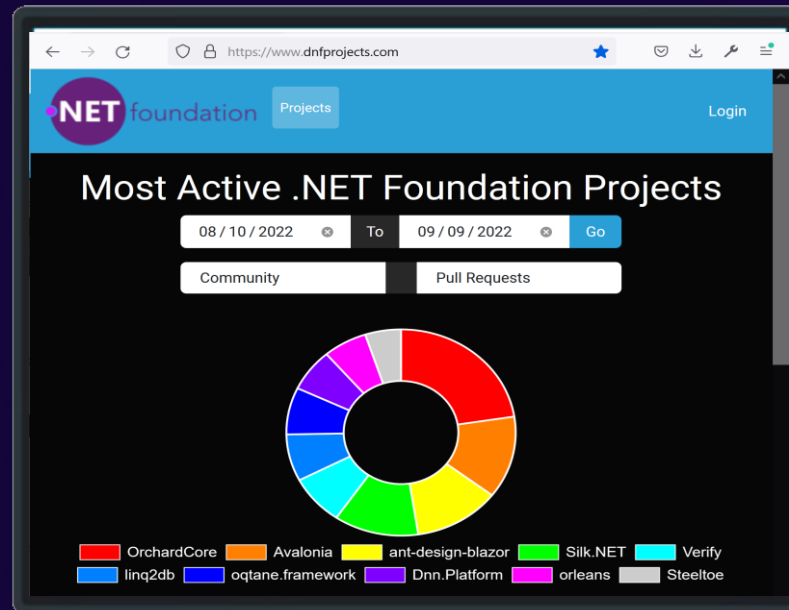
- **Authentication** needs to function similarly to Blazor WebAssembly (ie. interactive login)
- This is because **Browser Redirect Flows** from server to client are not possible on Blazor Hybrid

```
if (hybrid)
{
    // hybrid apps utilize an interactive login
    var authstateprovider = (IdentityAuthenticationStateProvider)ServiceProvider
        .GetService(typeof(IdentityAuthenticationStateProvider));
    authstateprovider.NotifyAuthenticationChanged();
    NavigationManager.NavigateTo(NavigateUrl(_returnUrl, true));
}
```

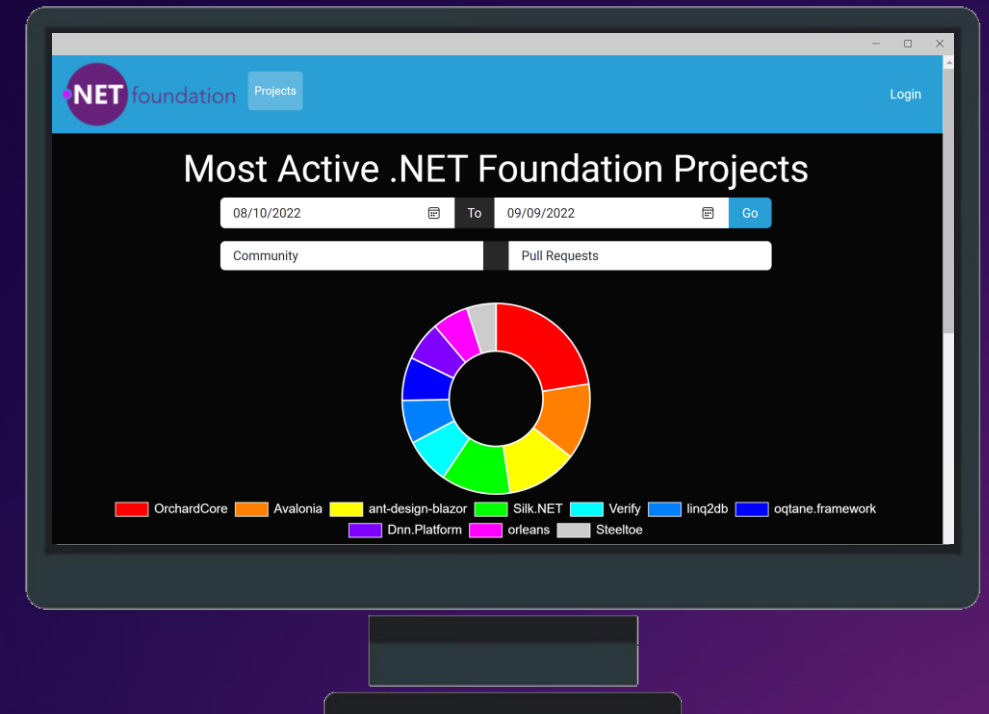
Voila! A Multi-Platform Application...



Mobile



Web



Desktop

Thank You!

Oqtane – Modular Application Framework

<https://www.oqtane.org>

<https://github.com/oqtane/oqtane.framework>