



Las Vegas

Desktop Apps Your Way

Sam Basu | [@samidip](https://twitter.com/samidip)
Developer Advocate | Progress

Your Code Powers the World.
Our Training Powers You.

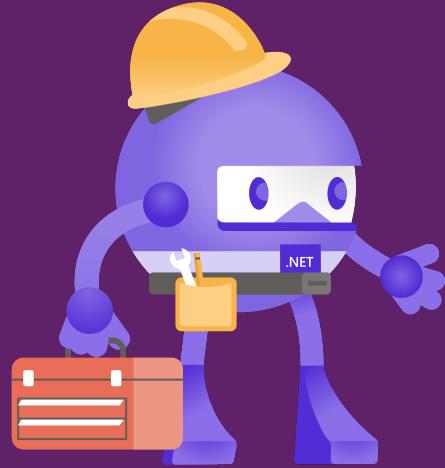
#VSLIVE



Perhaps not the coolest kid ...

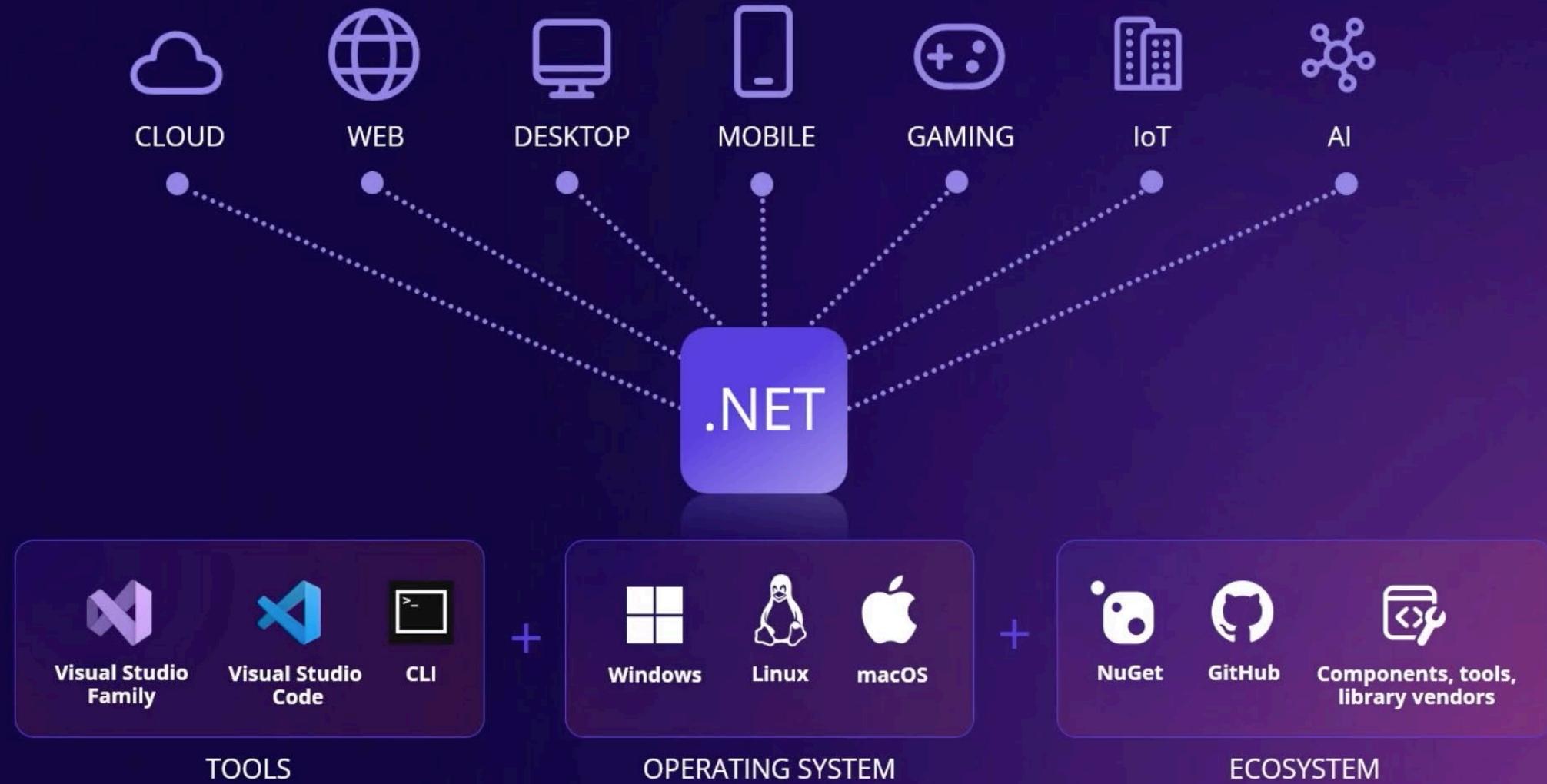
But Desktop runs businesses



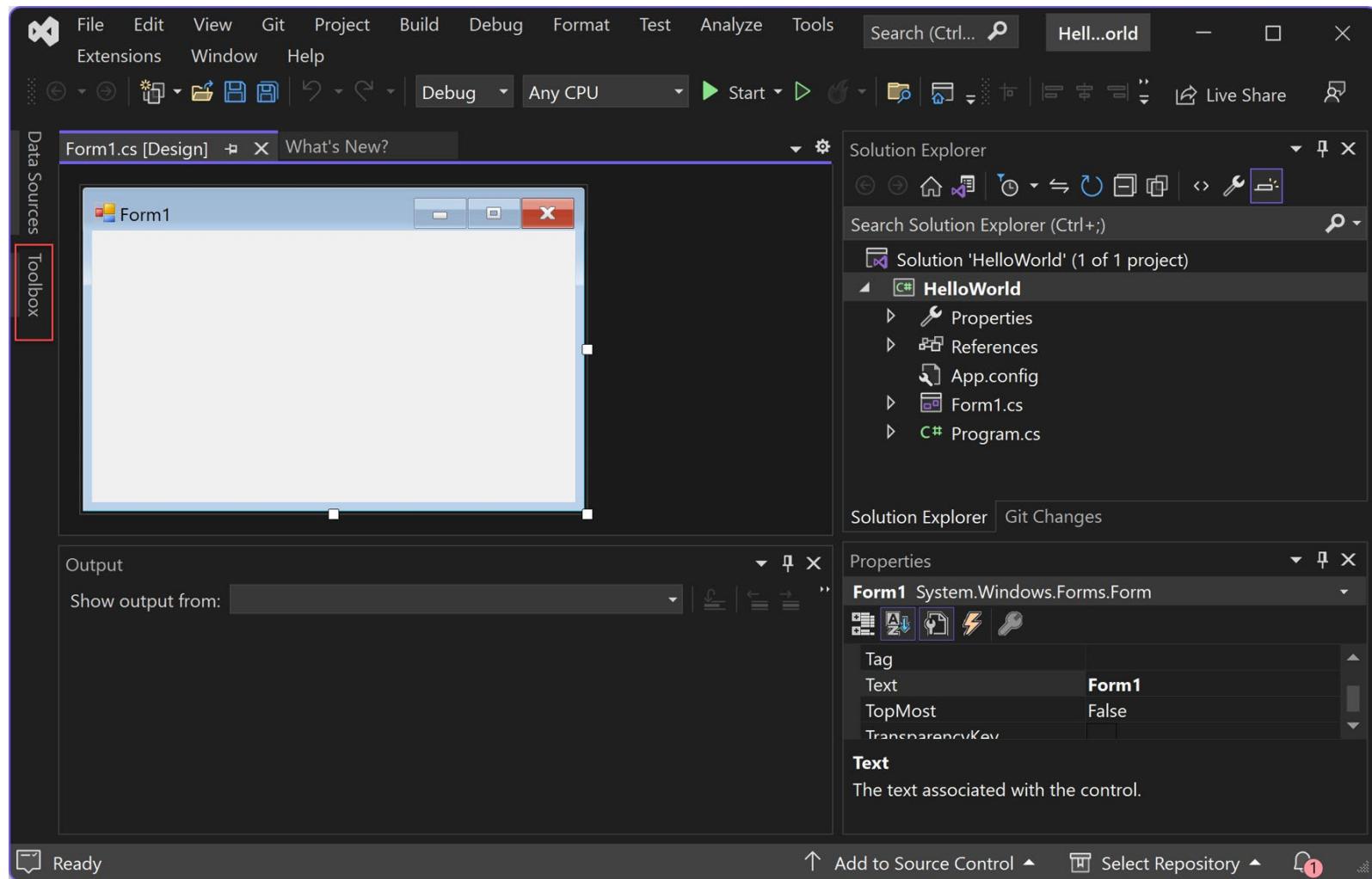


Building for Windows

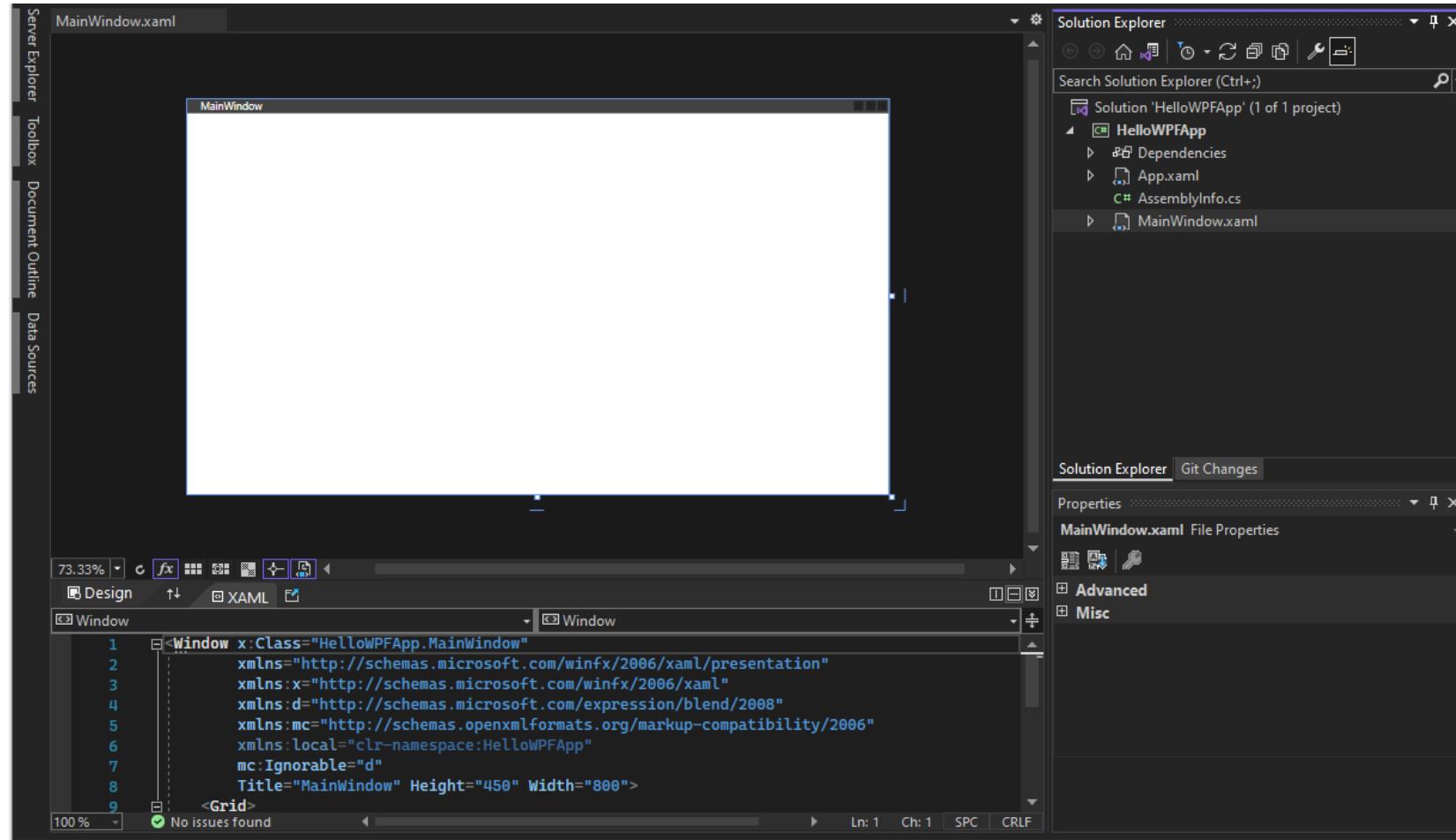
Build anything with a unified platform



Windows Forms (WinForms)



Windows Presentation Foundation (WPF)



Universal Windows Platform (UWP)



Windows UI 3 (WinUI)

The screenshot shows the 'What's New' section of the WinUI 3 documentation. On the left is a sidebar with a search bar and a list of categories: What's New (selected), All controls, Basic Input, Collections, Date and Time, Dialogs and Flyouts, Layout, Media, Menus and Toolbars, Motion, Navigation, Scrolling, Status and Info, Styles, Text, and Settings. The main content area features a large banner with a mountain background, the text 'What's New', the WinUI 3 logo, and 'Reunion 0.5'. Below the banner are sections for 'Recently Added Samples' and 'Recently Updated Samples'. The 'Recently Added Samples' section includes a card for 'WebView2' (New). The 'Recently Updated Samples' section includes cards for 'AppBarButton' (Updated), 'ComboBox' (Updated), 'ListView' (Updated), and 'MenuBar' (Updated).

What's New

WinUI 3

Reunion 0.5

Recently Added Samples

WebView2 New

A Microsoft Edge (Chromium) based control that hosts HTML content in an app.

Recently Updated Samples

AppBarButton Updated

App bar buttons differ from standard buttons in several ways:
- Their default appearance is a transparent background with a smaller...

ComboBox Updated

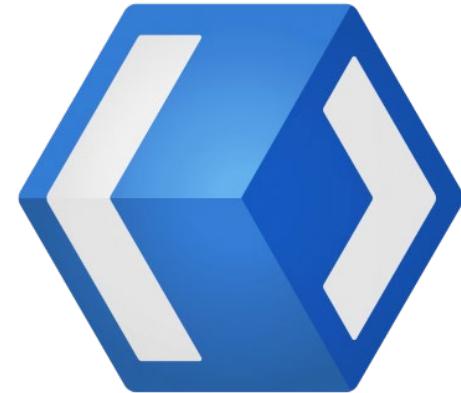
Use a ComboBox when you need to conserve on-screen space and when users select only one option at a time. A ComboBox shows only the currently...

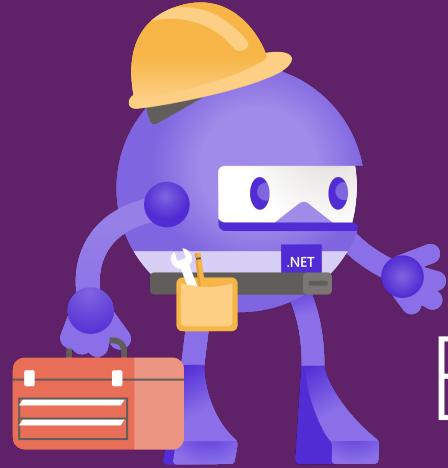
ListView Updated

The ListView lets you show a collection of items in a list that scrolls vertically.

MenuBar Updated

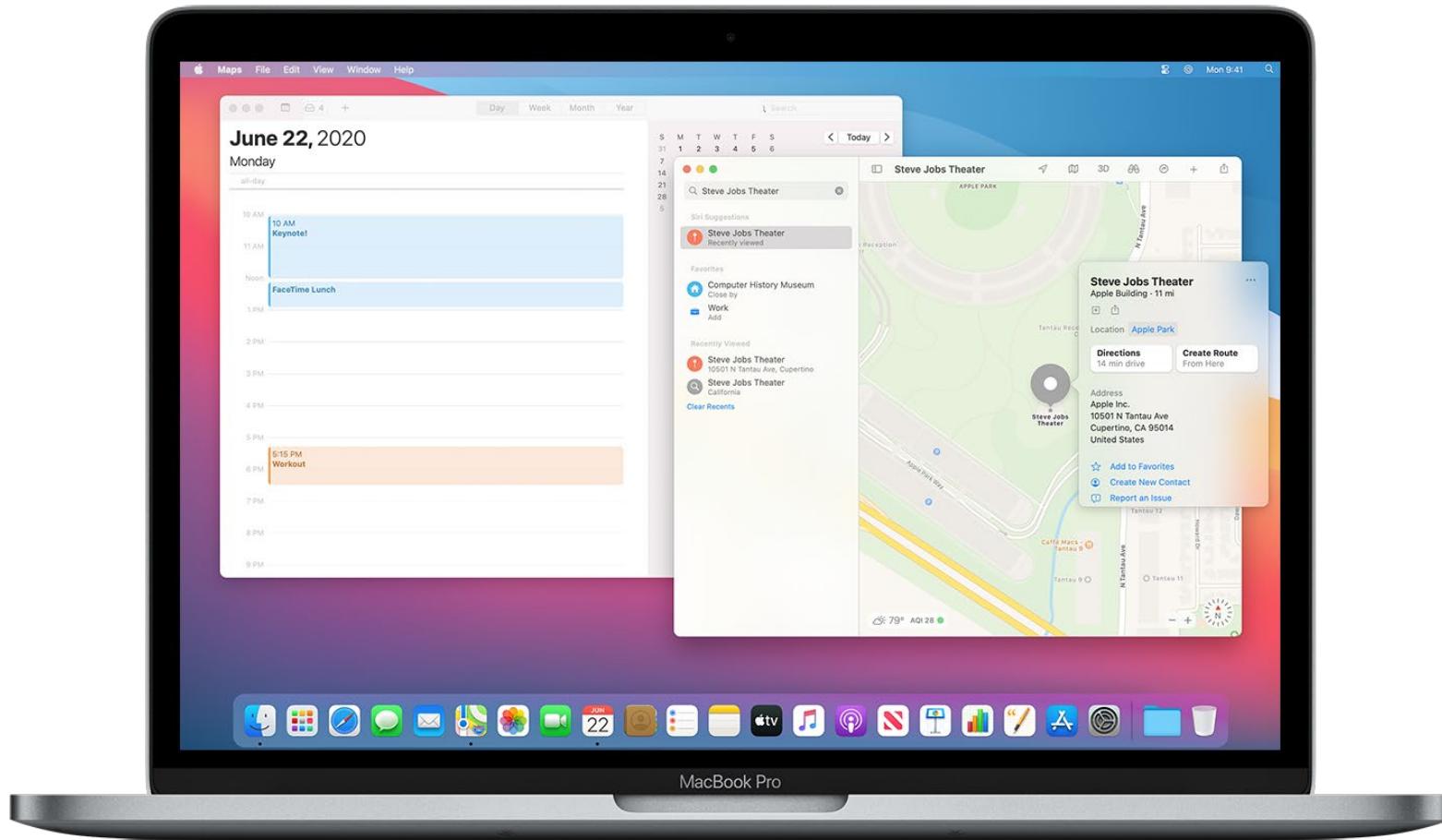
The Menubar simplifies the creation of basic applications by providing a set of menus at the top of the app or window





Building for macOS/Linux

AppKit + Xcode + SwiftUI





Mac Catalyst

Mac apps built with Mac Catalyst share code with your iPad apps, and you can add more features just for Mac. The latest desktop-class features in iPadOS 16 translate beautifully onto macOS 13. Your Mac app's toolbars are automatically optimized and document-based apps gain key features, such as menu items and the ability to rename the document in the toolbar. And you can use new Mac Catalyst APIs to enhance multiwindow behaviors, add custom views to your toolbars, and more.



Bring your iPad app to Mac



Get a head start on your Mac app

Your iPad app can transform into an amazing Mac app using the latest version of Xcode. Begin by adding a Mac Catalyst run destination in the project settings of your existing iPad app to create a Mac app that you can enhance further. Your Mac and iPad apps share the same project and source code, so it's easy to make changes in one place.



The Outliers

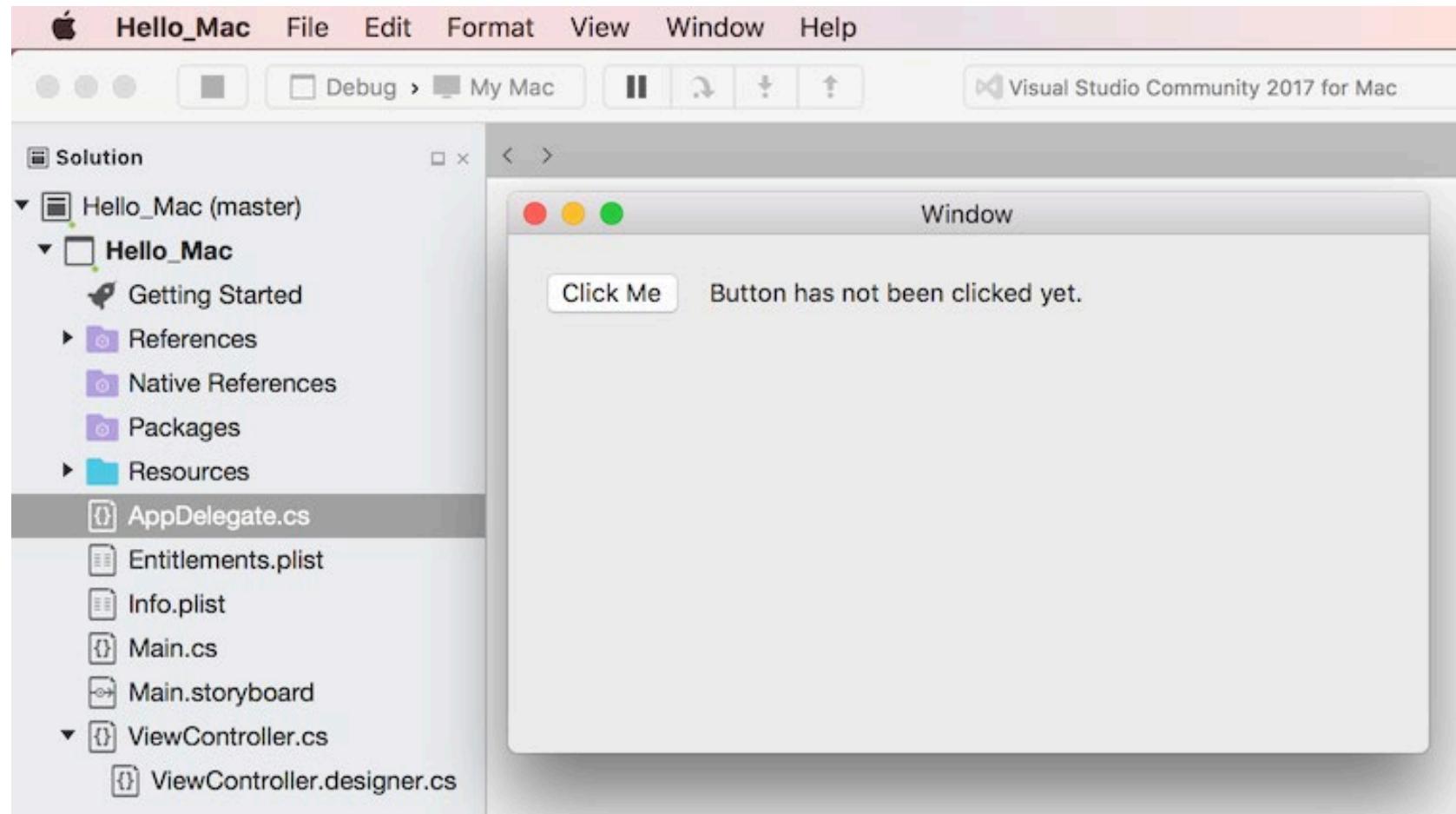
Traditional UI approach



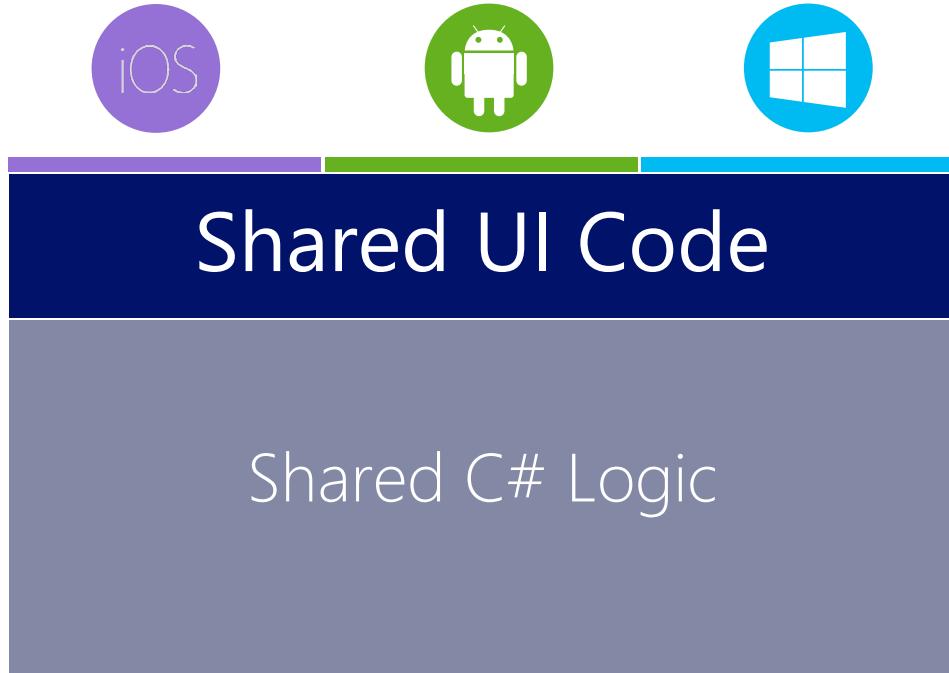
3 Native User Interfaces
Shared App Logic



Xamarin.Mac

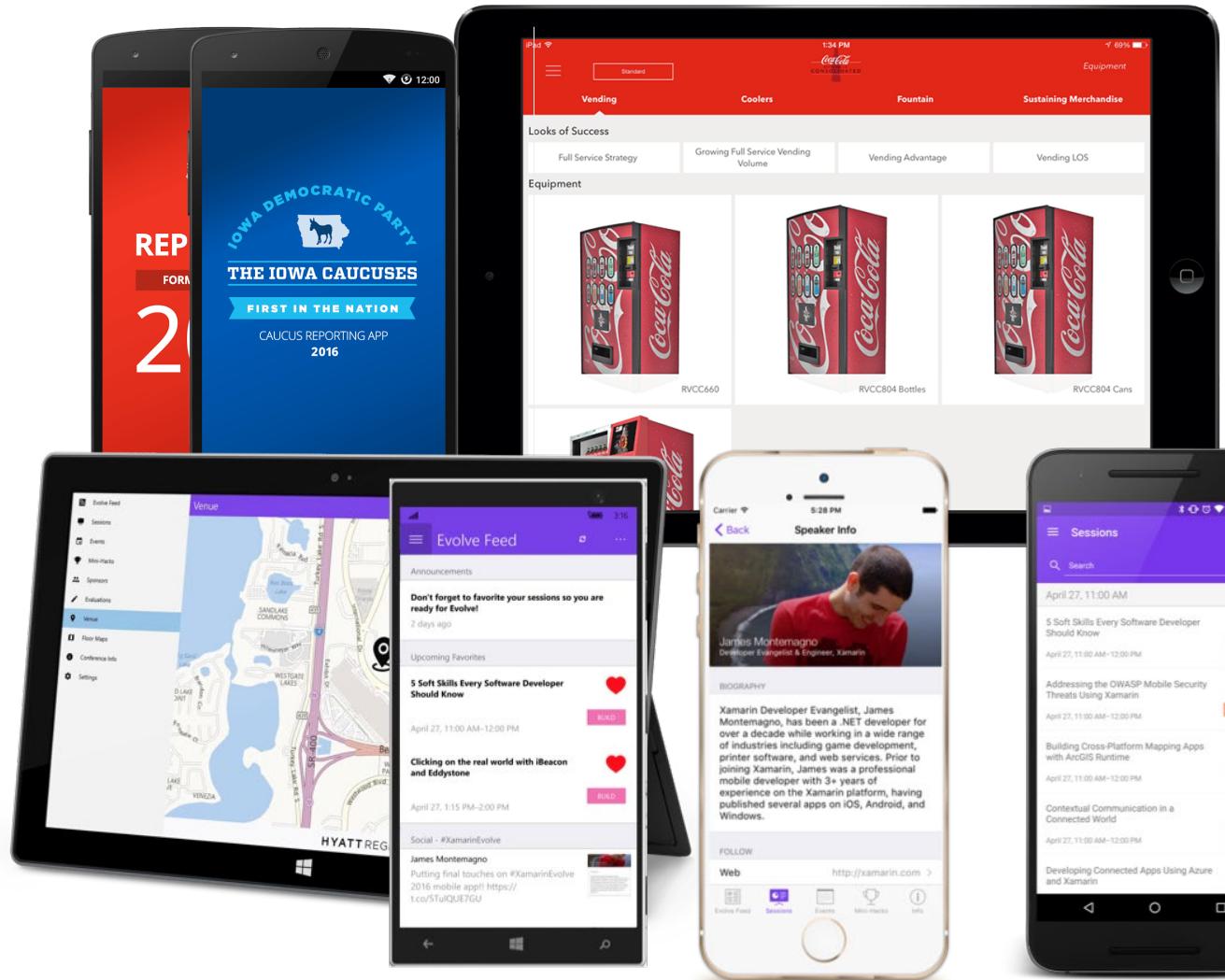


Xamarin.Forms approach

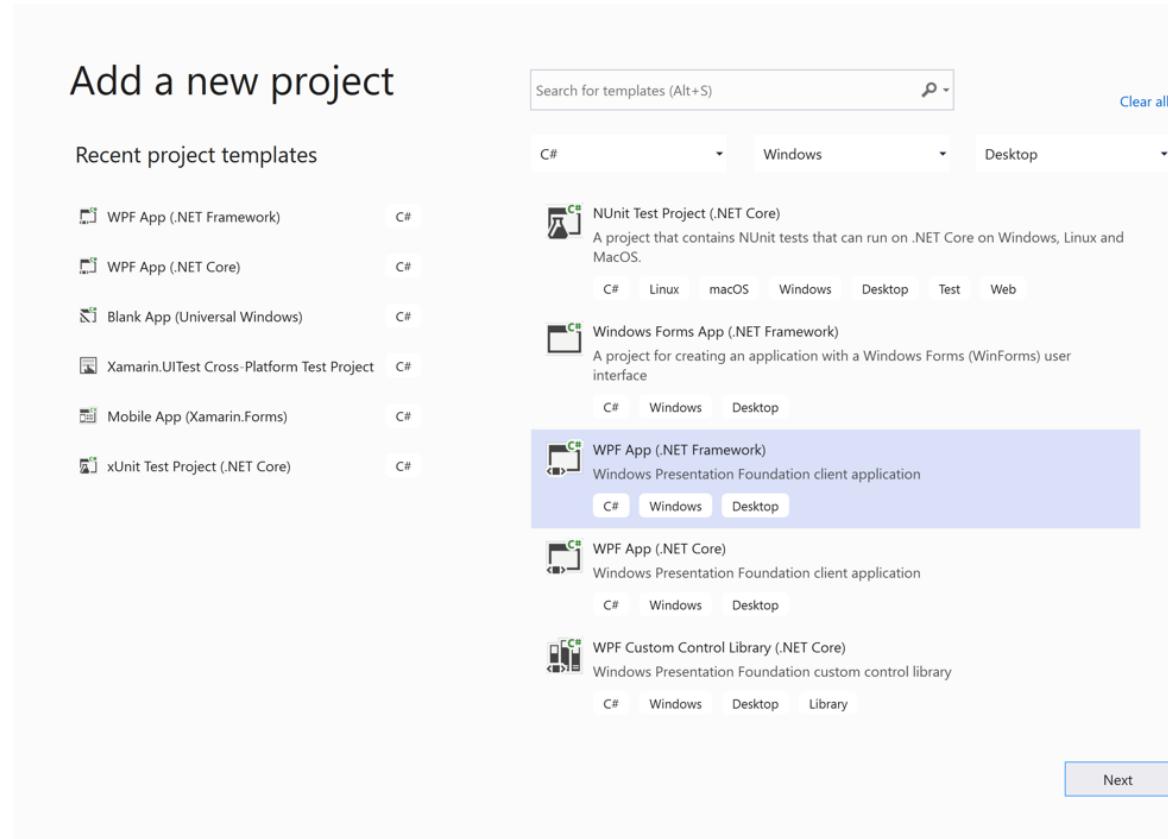


Shared User Interface

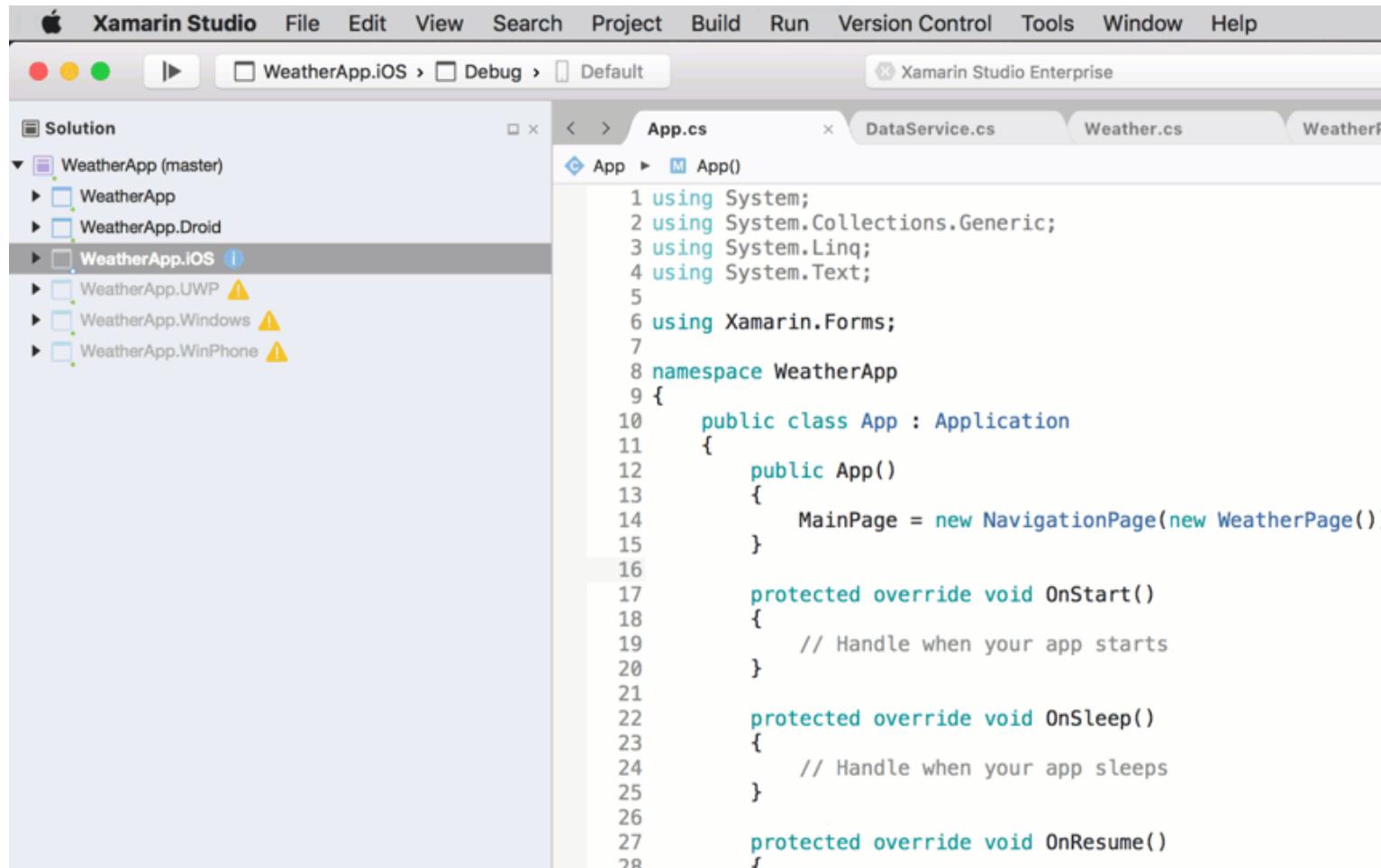
Shared App Logic



Xamarin.Forms Renderers for WPF



Xamarin.Forms Renderers for macOS/GTK



The screenshot shows the Xamarin Studio interface. The top menu bar includes File, Edit, View, Search, Project, Build, Run, Version Control, Tools, Window, and Help. The title bar indicates the project is "WeatherApp.iOS" in "Debug" mode, with "Default" selected. The "Xamarin Studio Enterprise" watermark is visible. The left sidebar shows the "Solution" tree with "WeatherApp (master)" expanded, containing "WeatherApp", "WeatherApp.Droid", "WeatherApp.iOS" (selected), "WeatherApp.UWP" (with a warning icon), "WeatherApp.Windows" (with a warning icon), and "WeatherApp.WinPhone" (with a warning icon). The main editor area displays the "App.cs" file content:

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5
6 using Xamarin.Forms;
7
8 namespace WeatherApp
9 {
10     public class App : Application
11     {
12         public App()
13         {
14             MainPage = new NavigationPage(new WeatherPage())
15         }
16
17         protected override void OnStart()
18         {
19             // Handle when your app starts
20         }
21
22         protected override void OnSleep()
23         {
24             // Handle when your app sleeps
25         }
26
27         protected override void OnResume()
28     }
}
```



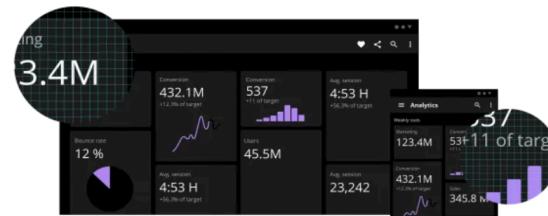
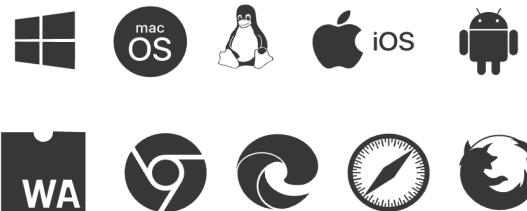
Pixel-Perfect Multi-Platform Applications with C# and WinUI

Open-source UI Platform for building single-codebase applications for Windows, iOS, Android, WebAssembly, macOS, and Linux

[Get Started](#)

Target All Operating Systems & Browsers

The first C# & XAML, **free** and **open-source** platform for creating true single-source, multi-platform applications. Reuse 99% of the business logic and UI layer across native mobile, web, and desktop.

[Read More](#)

Pixel Perfect Everywhere

Maintain **pixel-perfect control** over where pixels go while having a **choice** of developing platform-specific or custom look and feel of your application.

[Read More](#)

Introducing Avalonia XPF preview.

Cross-platform WPF, powered by Avalonia UI.

[Learn More →](#)

Create Multi-Platform Apps with .NET

Avalonia UI is open-source [|.](#) 

SUPPORTED PLATFORMS

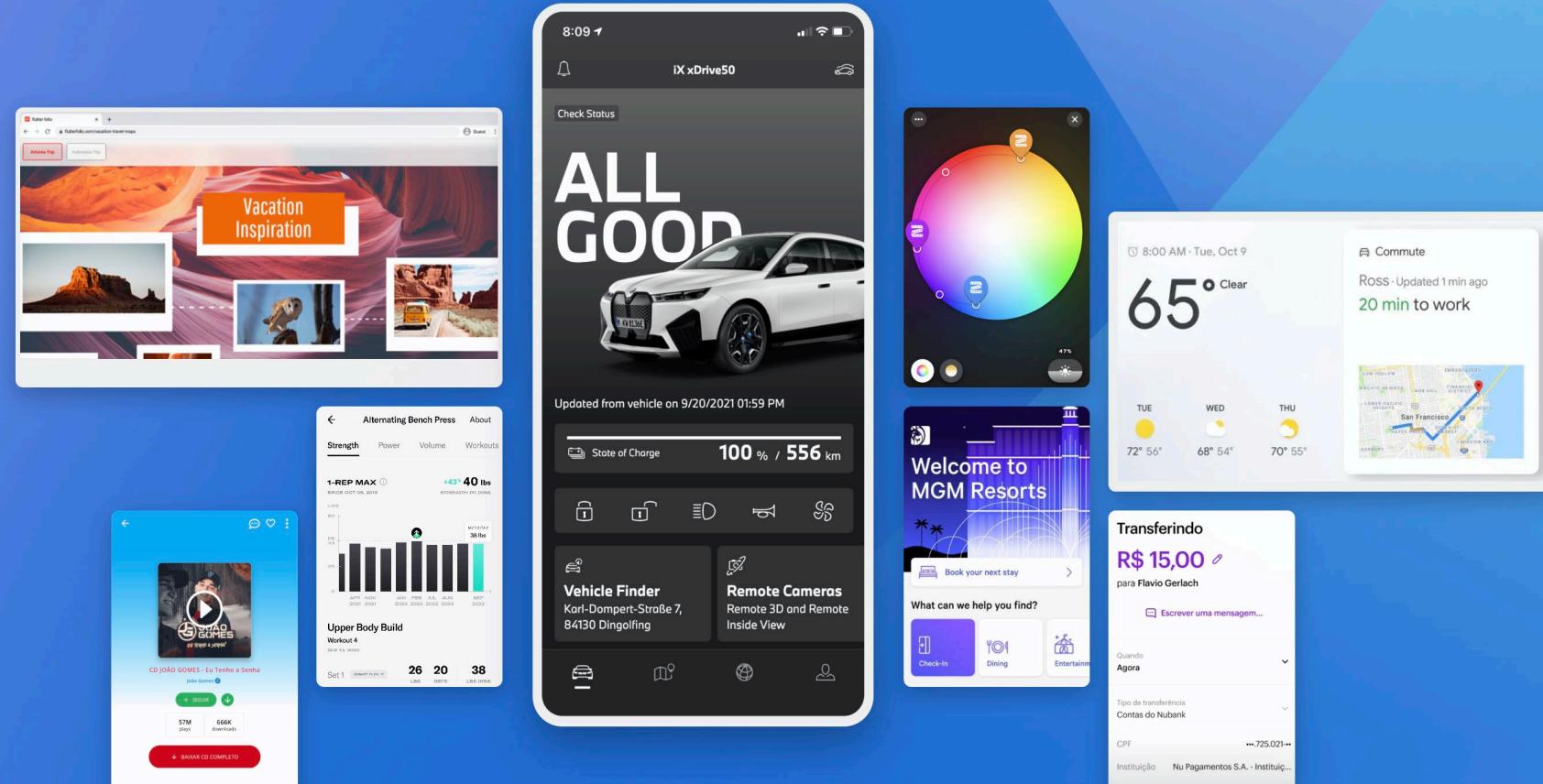
The screenshot shows a window titled "Manager for Speckle". The main interface displays a list of available connectors under the "AVAILABLE" section. The connectors listed are:

- Blender
- Archicad (Alpha)
- Grasshopper
- Rhino

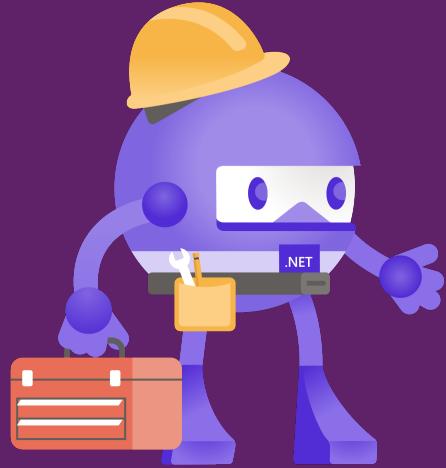
Below this, there are sections for "IN PROGRESS" (Blender) and "OTHERS" (ArcGIS, AutoCAD, Civil 3D, CSBBridge).

At the bottom of the window, there are navigation arrows and a set of small circular progress indicators.

Build apps for any screen



Flutter transforms the app development process. Build, test, and deploy beautiful mobile, web, desktop, and embedded apps from a single codebase.

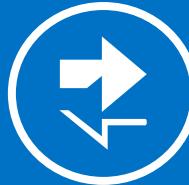


Welcoming the Web

ASP.NET Core & the Modern Web



Totally Modular



Faster Development Cycle



Seamless transition
from on-premises to cloud



Choose your Editors
and Tools



Open Source
with Contributions



Cross-Platform



Fast

Modern Web Apps

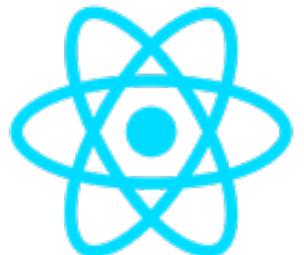
HTML



CSS



JS



Filter by title

Microsoft Edge Developer documentation

Develop for the web with Microsoft Edge

> Microsoft Edge DevTools

> Microsoft Edge extensions

› Progressive Web Apps

Overview of Progressive Web Apps (PWAs)

Get started with PWAs

Best practices for PWAs

> How to

> Reference

What's new

Progressive Web App demos

> WebView2

> Test and automation

> Development tips for Microsoft Edge

> Microsoft Edge IDE integration

> Accessibility in Microsoft Edge

Videos about web development with Microsoft Edge

Privacy whitepaper

The Web We Want initiative

Learn / Microsoft Edge /

+ | Edit | ...

Overview of Progressive Web Apps (PWAs)

Article • 01/24/2023 • 4 minutes to read • 7 contributors

[Feedback](#)

Progressive Web Apps (PWAs) are applications that you build by using web technologies, and that can be installed and can run on all devices, from one codebase.

PWAs provide native-like experiences to your users on supporting devices. They adapt to the capabilities supported by each device and they can also run in web browsers, like websites.

To start building a PWA, see [Get started with Progressive Web Apps](#).

PWA benefits

Native-like experiences

When installed on a device, PWAs function just like other apps. For example:

- PWAs have their own application icons that can be added to a device's home screen or task bar.
- PWAs can be launched automatically when an associated file type is opened.
- PWAs can run when the user signs in.
- PWAs can be submitted to application stores, such as the Microsoft Store.

Advanced capabilities

PWAs also have access to advanced capabilities. For example:

- PWAs can continue working when the device is offline.
- PWAs support push notifications.
- PWAs can perform periodic updates even when the application is not running.
- PWAs can access hardware features.

Web-related advantages

Finally, PWAs can run in web browsers, just like websites. Running like websites gives PWAs with advantages, such as:

- PWAs can be indexed by search engines.
- PWAs can be shared and launched from a standard web link.
- PWAs are safe for users because they use secure HTTPS endpoints and other user safeguards.
- PWAs adapt to the user's screen size or orientation, and input method.
- PWAs can use advanced web APIs such as WebBluetooth, WebUSB, WebPayment, WebAuthn, or WebAssembly.

Lower development cost

PWAs have a **much lower cross-platform development cost** than compiled apps that require a specific, separate codebase for each platform, such as Android, iOS, and each desktop operating system.

With a PWA, you can use a single codebase that's shared between your website, mobile app, and desktop app (across operating systems).

≡ In this article

[PWA benefits](#)[Cross-device compatibility](#)[Bridging the gap between web and native](#)[The Microsoft Store](#)[Show more ▾](#)



Build cross-platform desktop apps with
JavaScript, HTML, and CSS

Docs



Web Technologies

Electron embeds Chromium and Node.js to enable web developers to create desktop applications.



Cross Platform

Compatible with macOS, Windows, and Linux, Electron apps run on three platforms across all supported architectures.



Open Source

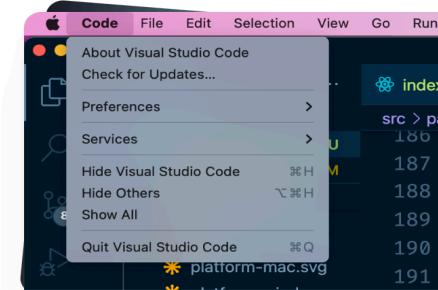
Electron is an open source project maintained by the [OpenJS Foundation](#) and an active community of contributors.

Desktop development made easy

Electron takes care of the hard parts so you can focus on the core of your application.

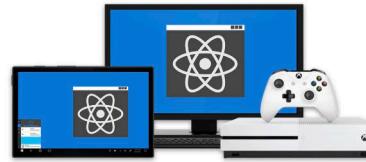
Native graphical user interfaces

Interact with your operating system's interfaces with Electron's main process APIs. Customize your [application window](#) appearance, control application [menus](#), or alert users through [dialogs](#) or [notifications](#).

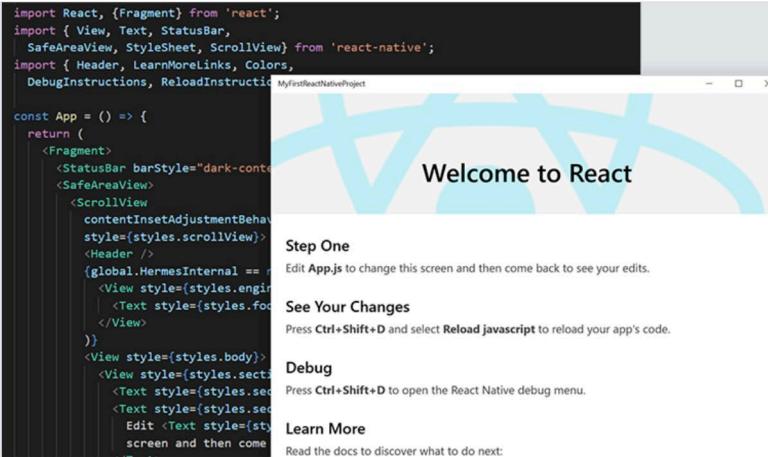


[Follow @ReactNativeMSFT](#)

React Native for Windows + macOS



Bring your React Native apps to some of the most powerful devices out there



```
import React, {Fragment} from 'react';
import { View, Text, StatusBar,
  SafeAreaView, StyleSheet, ScrollView } from 'react-native';
import { Header, LearnMoreLinks, Colors,
  DebugInstructions, ReloadInstructions } from 'react-native/Libraries/NewAppScreen';

const App = () => {
  return (
    <Fragment>
      <StatusBar barStyle="dark-content" />
      <SafeAreaView>
        <ScrollView
          contentInsetAdjustmentBehavior="automatic"
          style={styles.scrollView}>
          <Header />
          {global.HermesInternal == null ? null : <View style={styles.engine}>
            <Text style={styles.text}>Hermes</Text>
          </View>}
          <View style={styles.body}>
            <View style={styles.sectionHeader}>
              <Text style={styles.title}>Welcome to React</Text>
            </View>
            <Text style={styles.paragraph}>
              Edit <Text style={styles.link}>App.js</Text> to change this screen and then come back to see your edits.
            </Text>
            <Text style={styles.paragraph}>
              Secondary click in this window and choose Reload to reload your app's code.
            </Text>
            <Text style={styles.paragraph}>
              Press Ctrl+Shift+D to open the React Native debug menu.
            </Text>
            <Text style={styles.paragraph}>
              Read the docs to discover what to do next:
            </Text>
          </View>
        </ScrollView>
      </SafeAreaView>
    </Fragment>
  );
}

const styles = StyleSheet.create({
  scrollView: {
    backgroundColor: Colors.lighter,
  },
  engine: {
    position: 'absolute',
    top: -30,
    left: -30,
  },
  body: {
    padding: 10,
  },
  sectionHeader: {
    marginTop: 30,
    marginBottom: 10,
  },
  title: {
    fontSize: 24,
    fontWeight: 'bold',
  },
  paragraph: {
    fontSize: 18,
  },
  link: {
    color: Colors.primary,
  },
});

export default App;
```

Build for macOS



Extend your desktop experience to more than just Windows!
Try out our fully supported macOS extension to React Native.

[Get started with macOS](#)

Build for Windows



Take your apps across PC, Xbox, Surface Tablets, and dual-screens with our robust Windows extension to React Native.

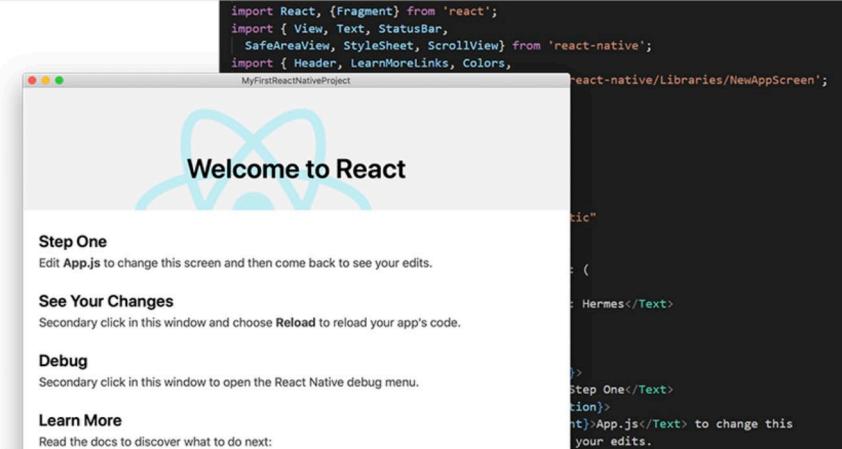
[Get started with Windows](#)

```
import React, {Fragment} from 'react';
import { View, Text, StatusBar,
  SafeAreaView, StyleSheet, ScrollView } from 'react-native';
import { Header, LearnMoreLinks, Colors,
  DebugInstructions, ReloadInstructions } from 'react-native/Libraries/NewAppScreen';

const App = () => {
  return (
    <Fragment>
      <StatusBar barStyle="dark-content" />
      <SafeAreaView>
        <ScrollView
          contentInsetAdjustmentBehavior="automatic"
          style={styles.scrollView}>
          <Header />
          {global.HermesInternal == null ? null : <View style={styles.engine}>
            <Text style={styles.text}>Hermes</Text>
          </View>}
          <View style={styles.body}>
            <View style={styles.sectionHeader}>
              <Text style={styles.title}>Welcome to React</Text>
            </View>
            <Text style={styles.paragraph}>
              Edit <Text style={styles.link}>App.js</Text> to change this screen and then come back to see your edits.
            </Text>
            <Text style={styles.paragraph}>
              Secondary click in this window and choose Reload to reload your app's code.
            </Text>
            <Text style={styles.paragraph}>
              Press Ctrl+Shift+D to open the React Native debug menu.
            </Text>
            <Text style={styles.paragraph}>
              Read the docs to discover what to do next:
            </Text>
          </View>
        </ScrollView>
      </SafeAreaView>
    </Fragment>
  );
}

const styles = StyleSheet.create({
  scrollView: {
    backgroundColor: Colors.lighter,
  },
  engine: {
    position: 'absolute',
    top: -30,
    left: -30,
  },
  body: {
    padding: 10,
  },
  sectionHeader: {
    marginTop: 30,
    marginBottom: 10,
  },
  title: {
    fontSize: 24,
    fontWeight: 'bold',
  },
  paragraph: {
    fontSize: 18,
  },
  link: {
    color: Colors.primary,
  },
});

export default App;
```



```
import React, {Fragment} from 'react';
import { View, Text, StatusBar,
  SafeAreaView, StyleSheet, ScrollView } from 'react-native';
import { Header, LearnMoreLinks, Colors,
  DebugInstructions, ReloadInstructions } from 'react-native/Libraries/NewAppScreen';

const App = () => {
  return (
    <Fragment>
      <StatusBar barStyle="dark-content" />
      <SafeAreaView>
        <ScrollView
          contentInsetAdjustmentBehavior="automatic"
          style={styles.scrollView}>
          <Header />
          {global.HermesInternal == null ? null : <View style={styles.engine}>
            <Text style={styles.text}>Hermes</Text>
          </View>}
          <View style={styles.body}>
            <View style={styles.sectionHeader}>
              <Text style={styles.title}>Welcome to React</Text>
            </View>
            <Text style={styles.paragraph}>
              Edit <Text style={styles.link}>App.js</Text> to change this screen and then come back to see your edits.
            </Text>
            <Text style={styles.paragraph}>
              Secondary click in this window and choose Reload to reload your app's code.
            </Text>
            <Text style={styles.paragraph}>
              Press Ctrl+Shift+D to open the React Native debug menu.
            </Text>
            <Text style={styles.paragraph}>
              Read the docs to discover what to do next:
            </Text>
          </View>
        </ScrollView>
      </SafeAreaView>
    </Fragment>
  );
}

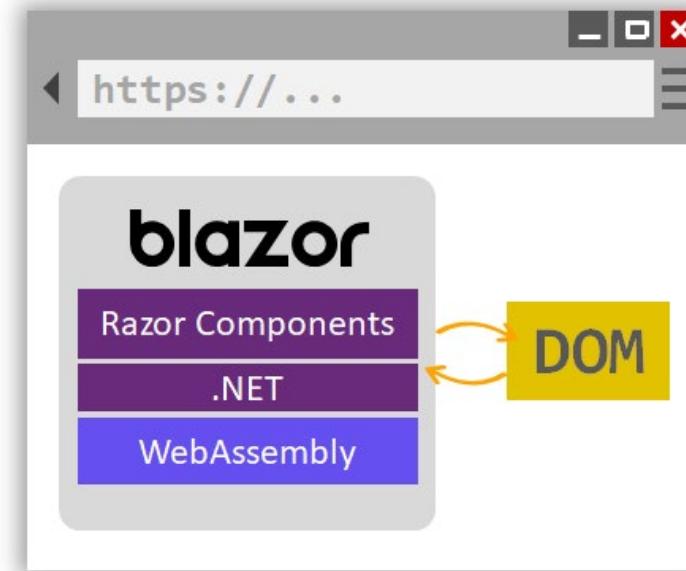
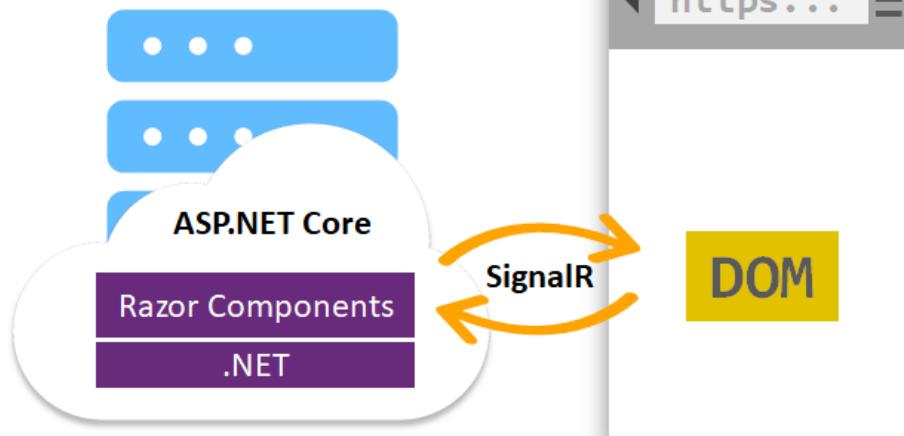
const styles = StyleSheet.create({
  scrollView: {
    backgroundColor: Colors.lighter,
  },
  engine: {
    position: 'absolute',
    top: -30,
    left: -30,
  },
  body: {
    padding: 10,
  },
  sectionHeader: {
    marginTop: 30,
    marginBottom: 10,
  },
  title: {
    fontSize: 24,
    fontWeight: 'bold',
  },
  paragraph: {
    fontSize: 18,
  },
  link: {
    color: Colors.primary,
  },
});

export default App;
```

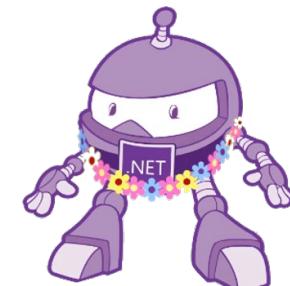
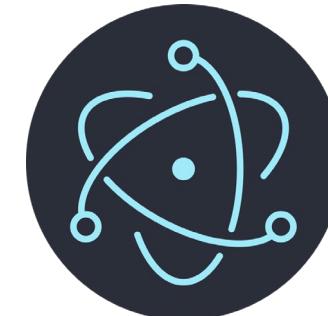
Blazing a Trail ..

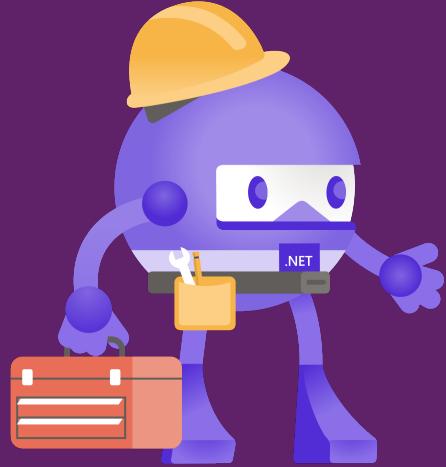


Blazing a trail ...



PWA



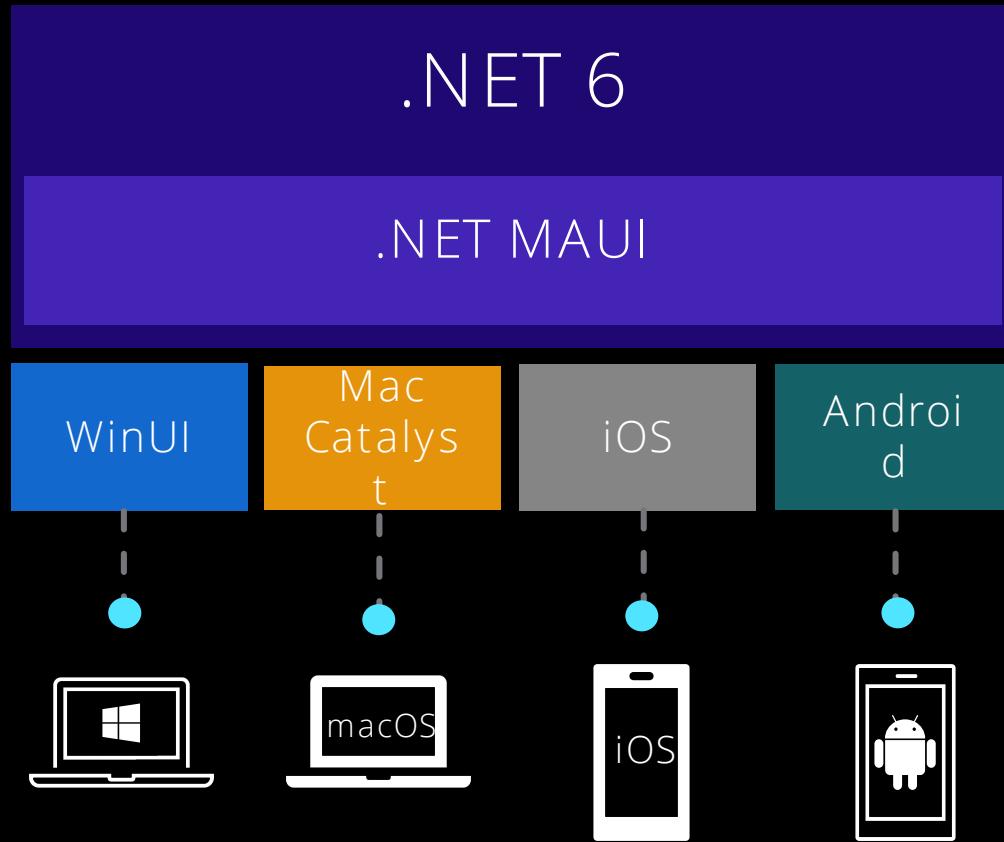


Embracing the New

.NET

MAUI

.NET MAUI Benefits



True Cross-platform | For Mobile & Desktop

Native UI & Performance | .NET Builder

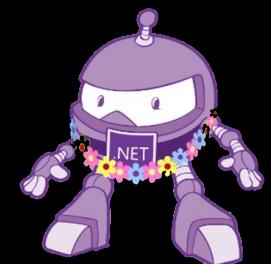
Single project system | Shared Resources

Easy Platform API access

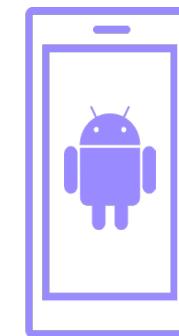
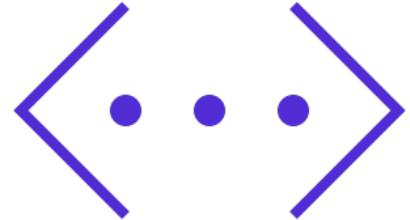
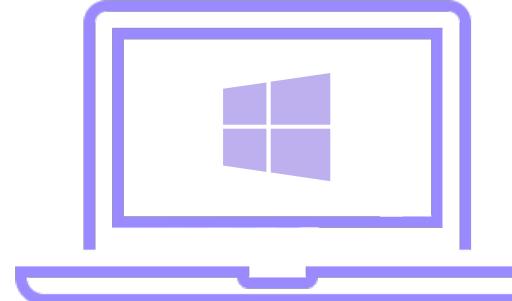
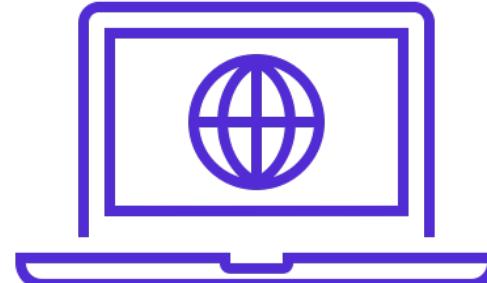
Handler Architecture | Easy Customization

Blazor Integration | Expansive ecosystem

github.com/dotnet/maui



Blazor Hybrid:@ + .NET MAUI



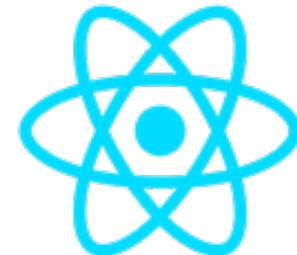
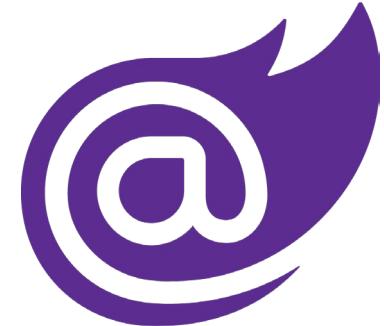


Reusability is key ...

Upgrade Assistants can Help



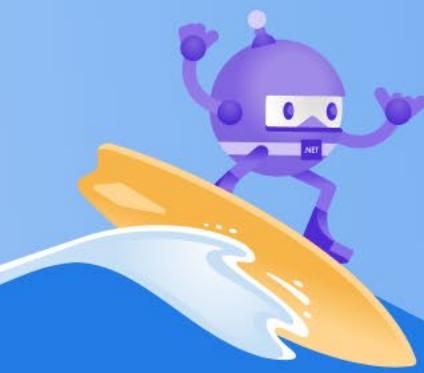
Migration & Modernization



Seeing is Believing ...



You have choices for Desktop



The world runs on software



Thank You!
Desktop Apps Your Way

Let's keep chatting | [@samidip](https://twitter.com/samidip)

Your Code Powers the World.
Our Training Powers You.