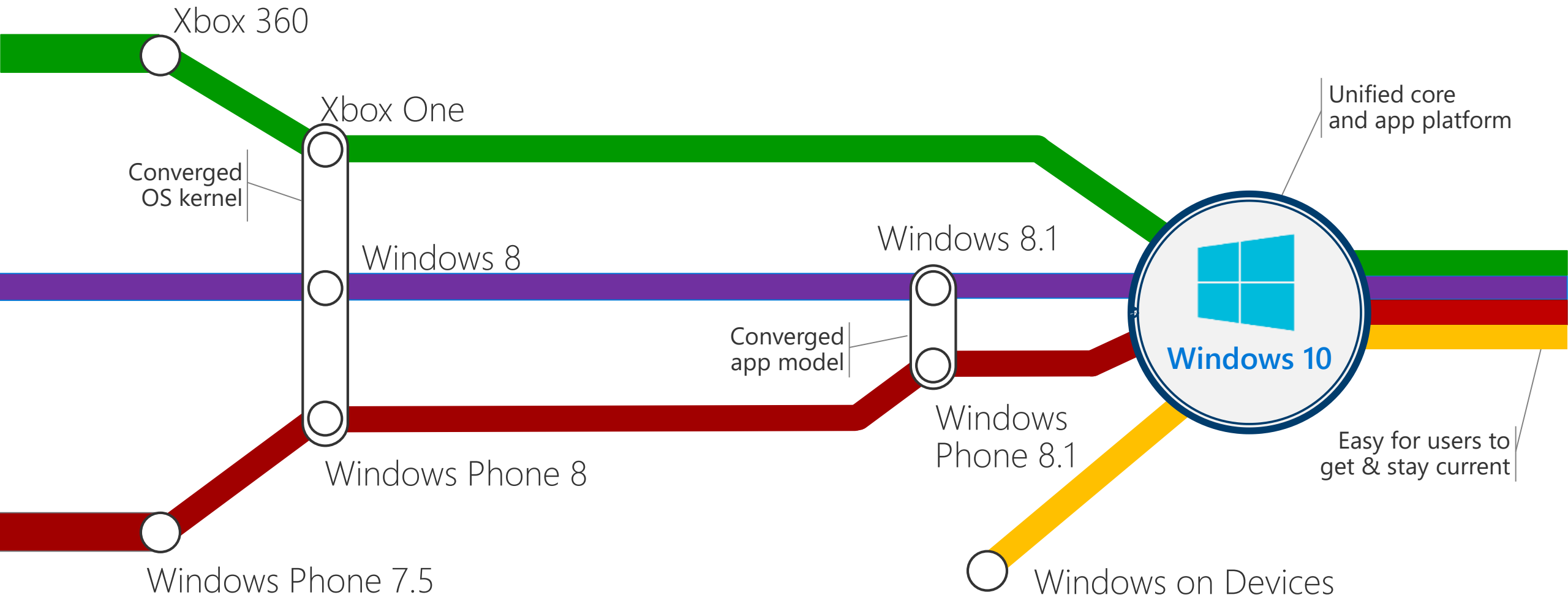
Developing Apps for the Universal Windows Platform

An Introduction

Microsoft

The convergence journey



Windows 10 since release

29 July 2015

Version 10.0
Build 10240

Version 1511: Bringing our device family together

29 July 2015

Version 10.0
Build 10240

Holiday 2015

Version 1511
Build 10586

2016: Bringing our device family together

29 July 2015

Holiday 2015

2016

A single, unified platform

Phone

Phablet

Small Tablet

Large Tablet

2-in-1s
(Tablet or Laptop)

Classic
Laptop

Desktops
& All-in-Ones

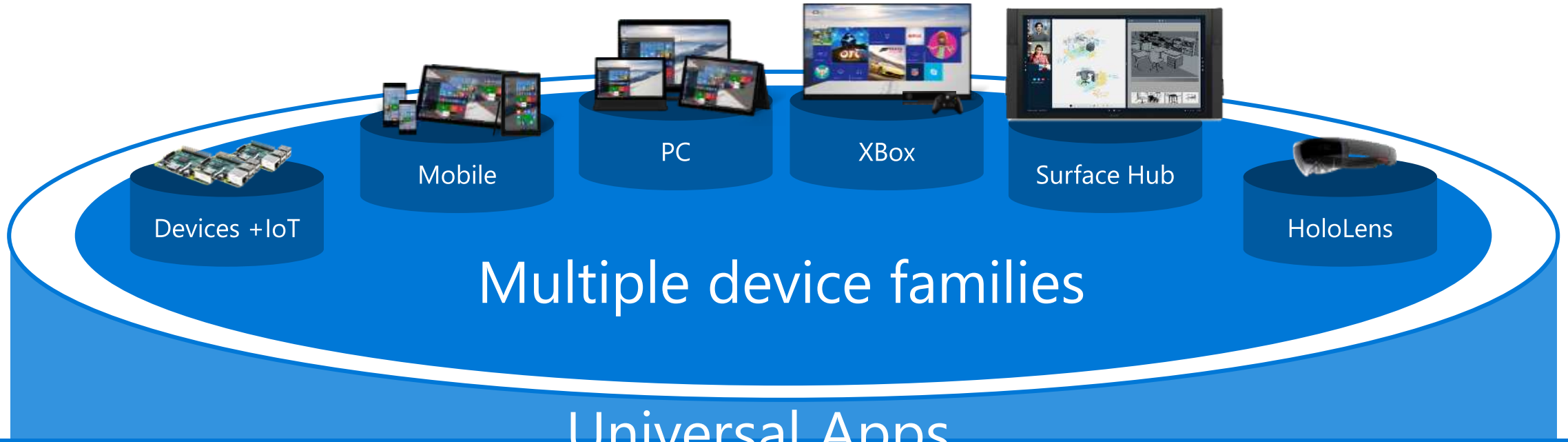
Windows 10

Surface Hub

Xbox

Holographic

IoT



Universal Apps

Adaptive User Interface

Natural User Inputs

One SDK + Tooling

One Store + One Dev Center

Reuse Existing Code

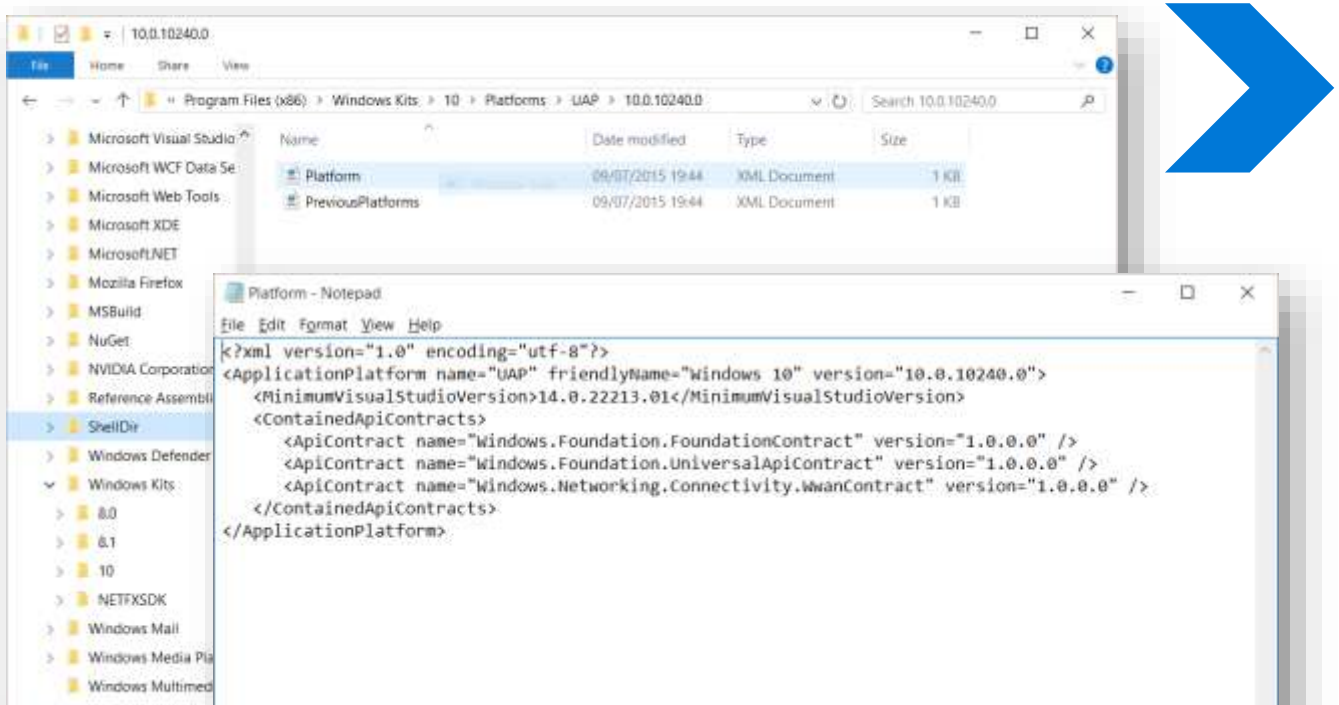
One Universal Windows Platform

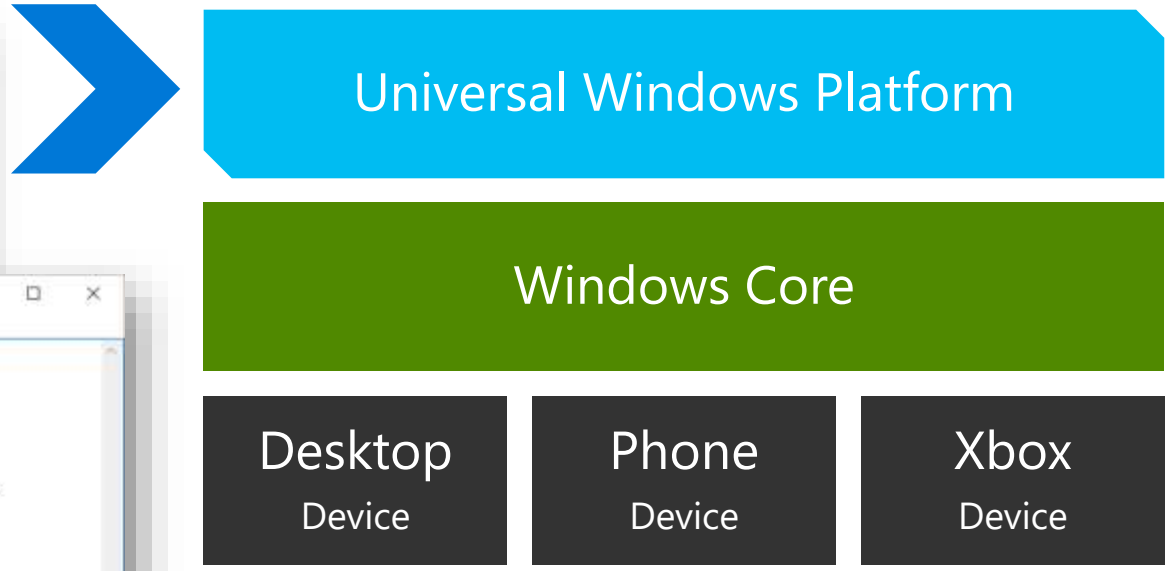
Universal Windows Platform

A single API surface

A guaranteed API surface

The same on all devices





A whole lot of APIs…

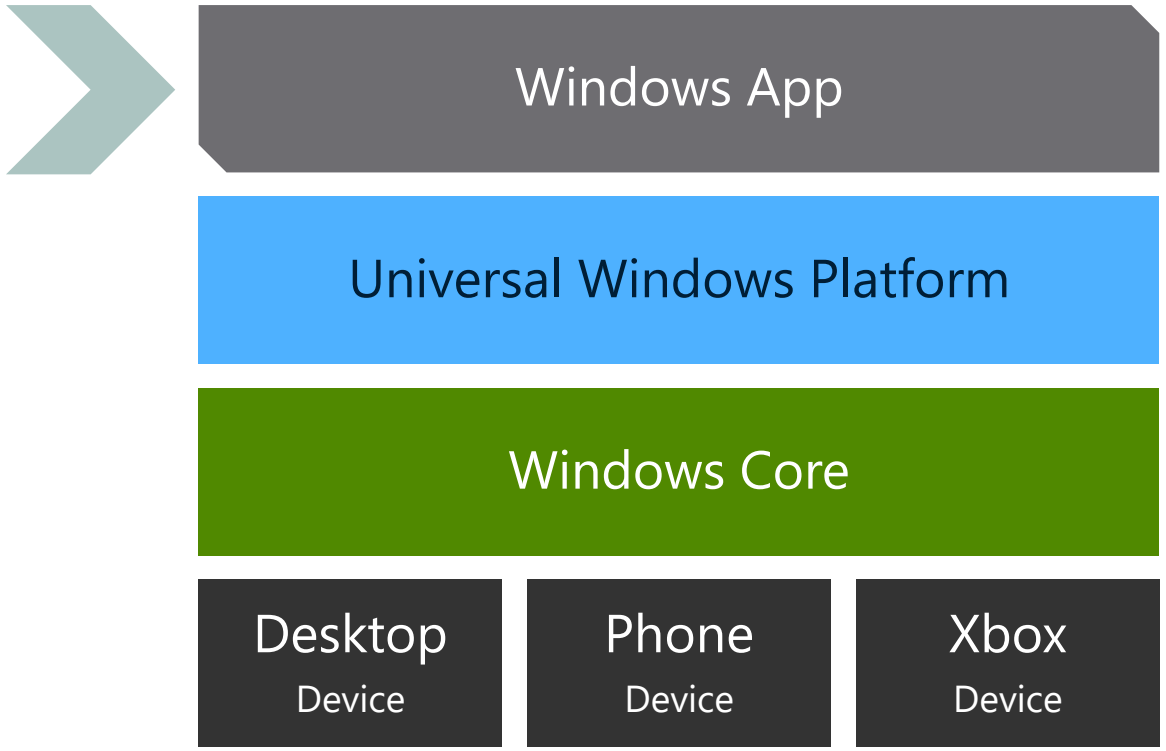
- Storage
- DirectX 12
- Speech and Cortana
- Networking
- NFC and Bluetooth
- Holographic
- Audio and Video
- Appointments/ Calendar
- Authentication Broker
- Background Transfer
- Maps and Location
- Sensors: Accelerometer, light, magnet …
- Tiles and Notifications
- App to App and App Services
- Inking
- XAML
- Background Tasks
- Data Roaming
- Data.XML
- Media Casting

Many, many more….

Windows app

A single binary

Running on any device
Testing for capabilities
Adjusting to devices



Universal Windows Platform

One Operating System

One Windows core for all devices

One App Platform

Apps run across every family

One Dev Center

Single submission flow and dashboard

One Store

Global reach, local monetization
Consumers, Business & Education

Some questions that may be forming

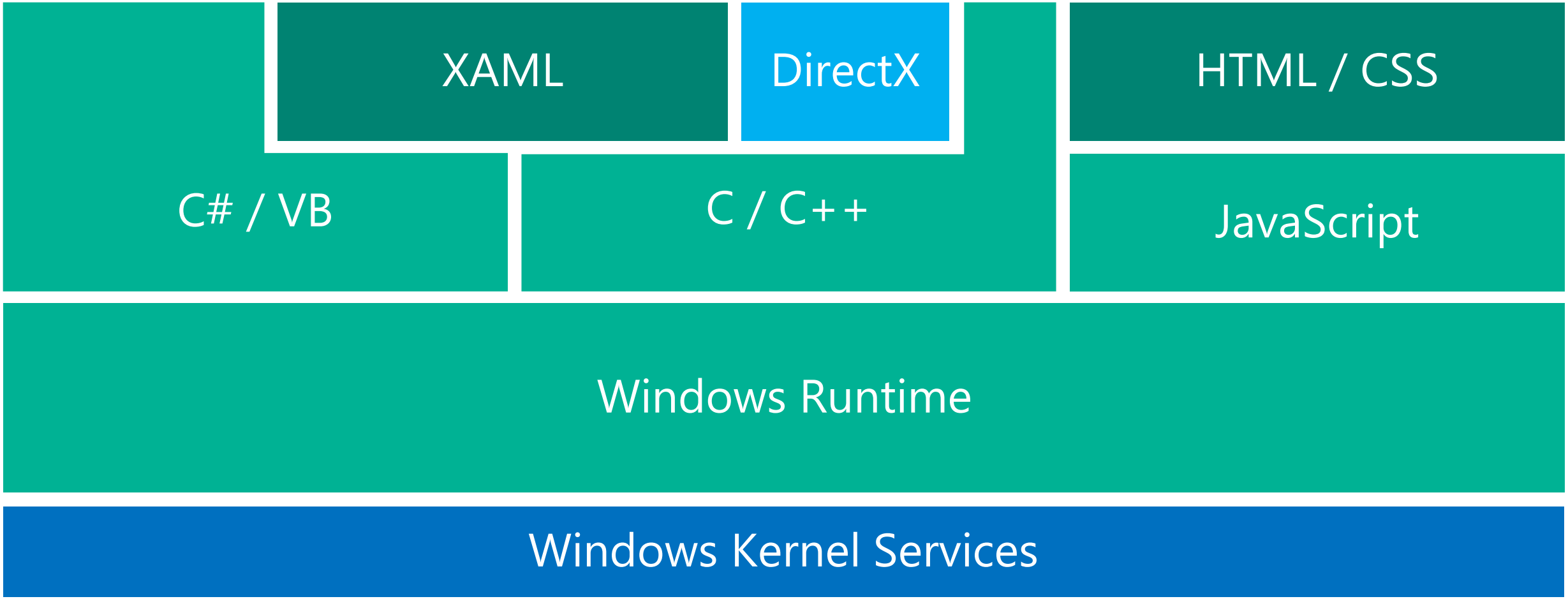
How do I target specific device types?

How can I access unique device features?

How do I design for different form factors, screen sizes and pixel densities

Development languages, tools

Development Options



Visual Studio 2015 Editions

Enterprise

Architecture Modelling, Testing Tools, VSO/ALM & Release Management, Lab Management

Professional

Architecture Validation, VSO/ALM & Feedback / Backlog Management

Community Editions

World Class IDE, Debugging, Code Metrics, Static Analysis etc

Where can I develop?

Windows 10

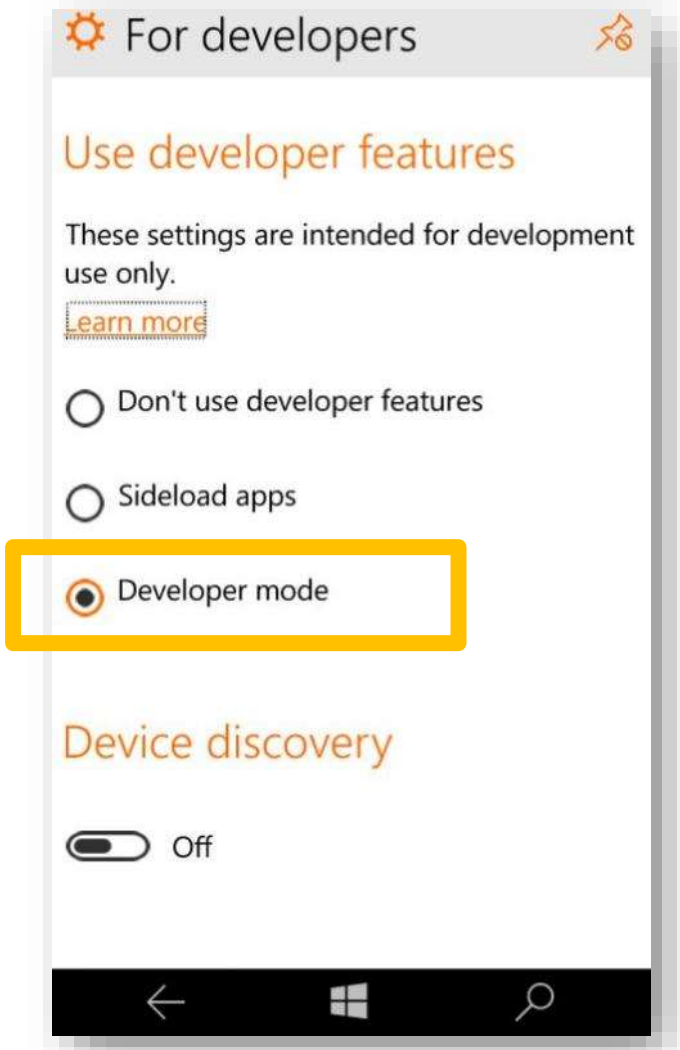
Requires Visual Studio 2015

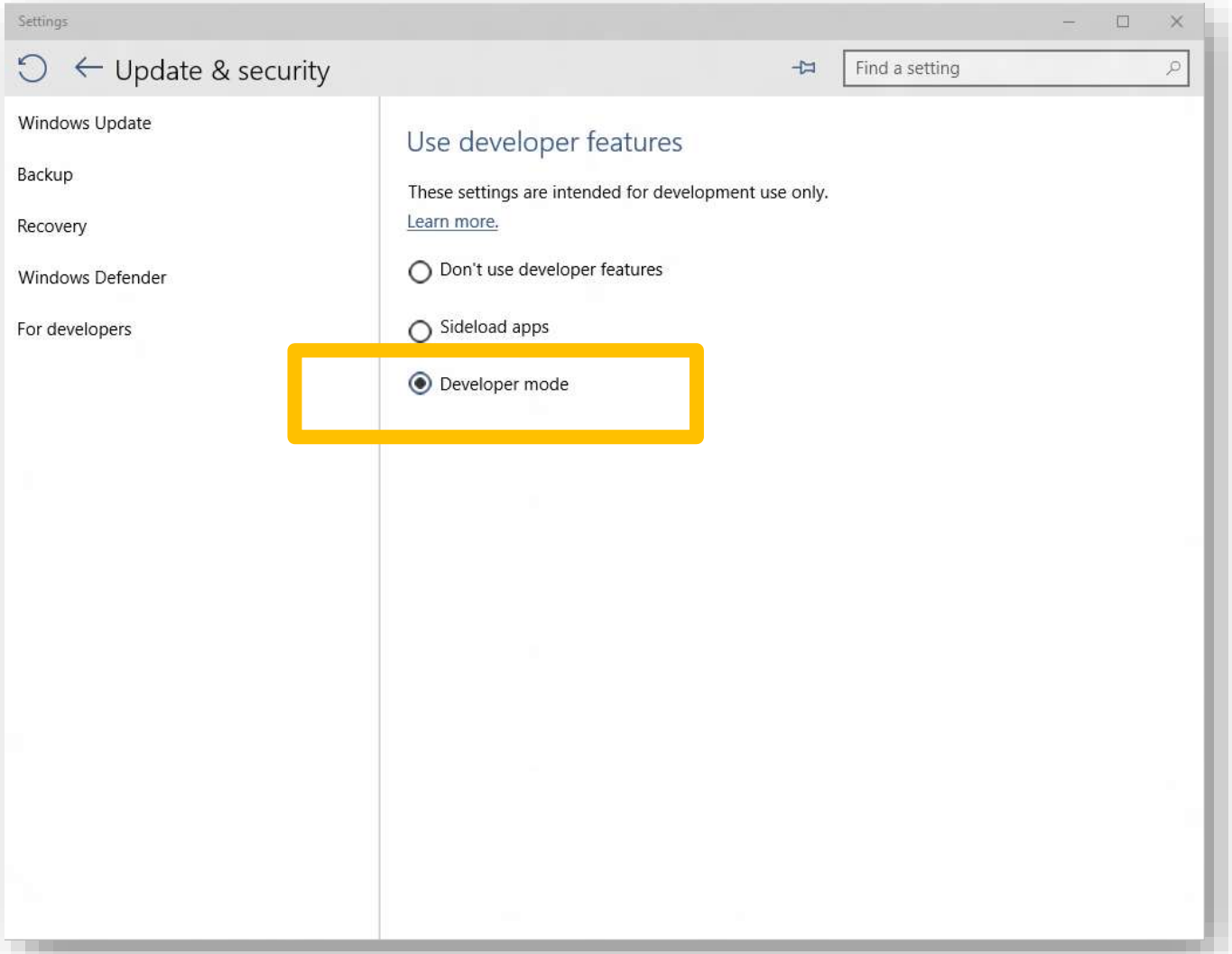
Windows 8.1 & Windows Server 2012 R2

The Visual Studio designer does not function

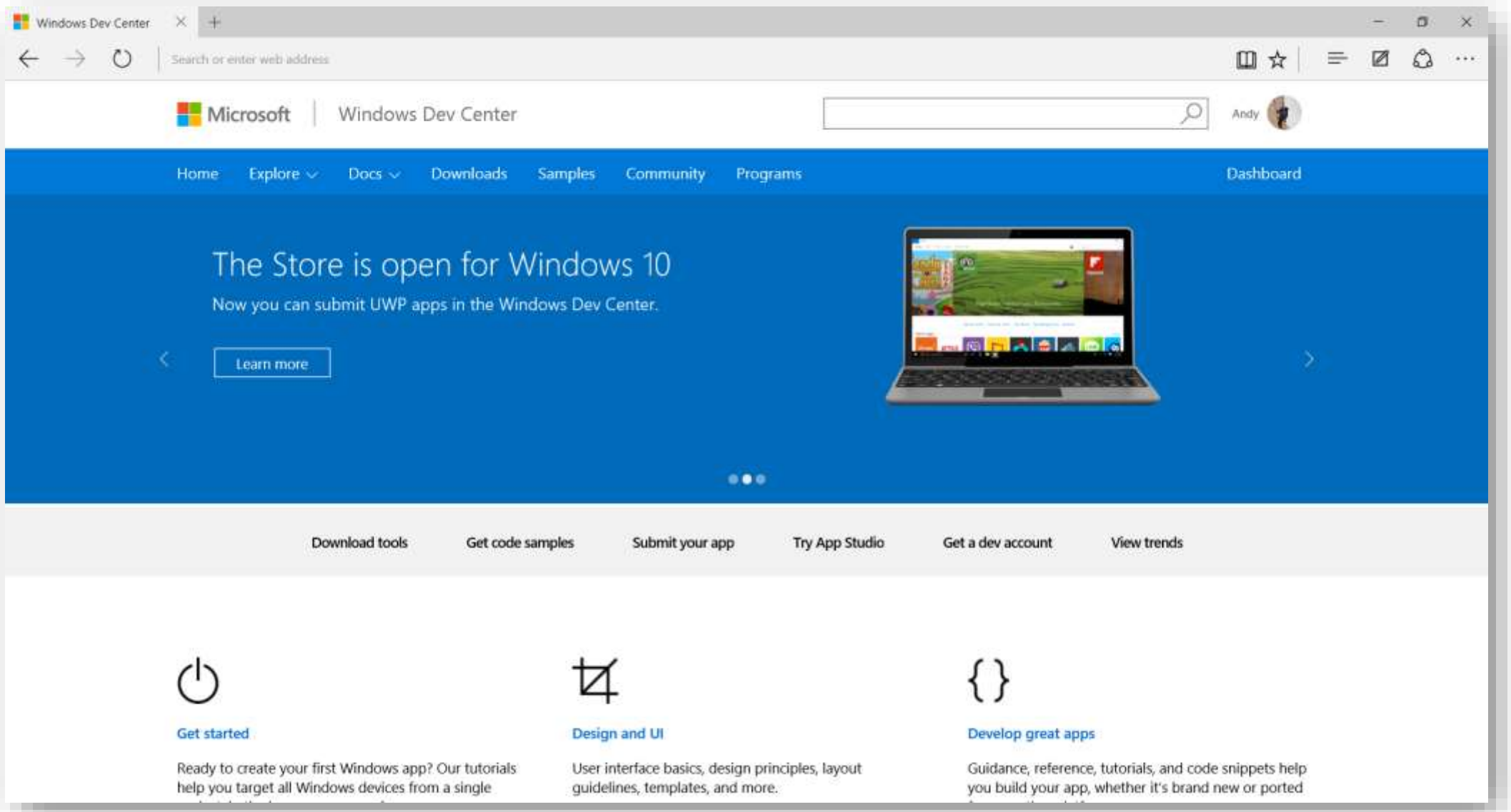
Debugging requires a Windows 10 device or Remote Debugging Tools

Developer unlock





http://dev.windows.com



Adaptive Code

Windows apps adapt to different versions of the platform

Windows apps adapt to different types of devices

Windows apps adapt to different screen sizes

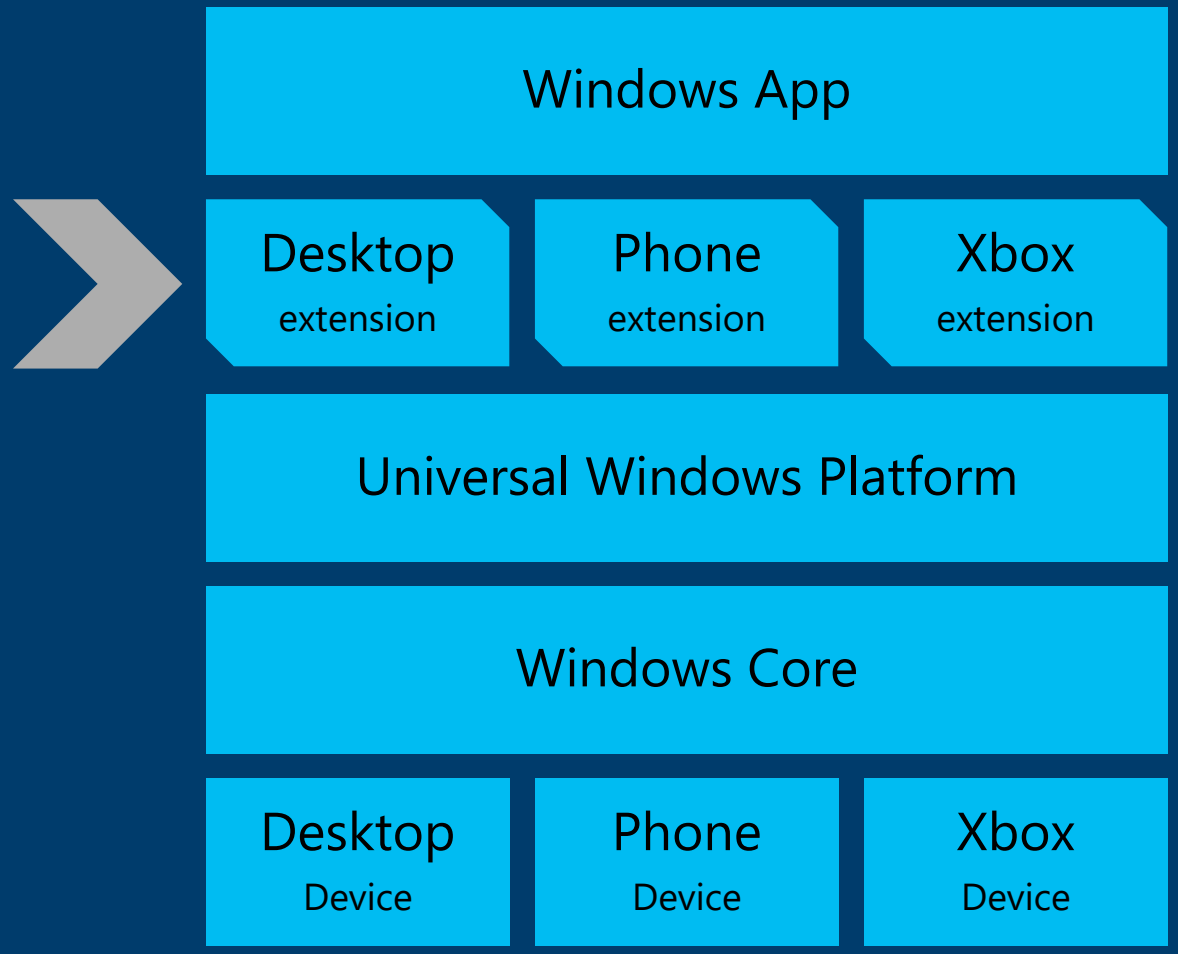
Adaptive UI handles different screens

Adaptive Code:
execute code only when running on specific device families and/or particular versions of platform/extension APIs

Introducing
Platform Extension SDKs

Platform extensions

- Device-specific API
  - Family-specific capabilities
  - Compatible across devices
  - Unique update cadence



The device families you choose determines which APIs you can call

The ApiInformation API tests for capabilities at runtime.

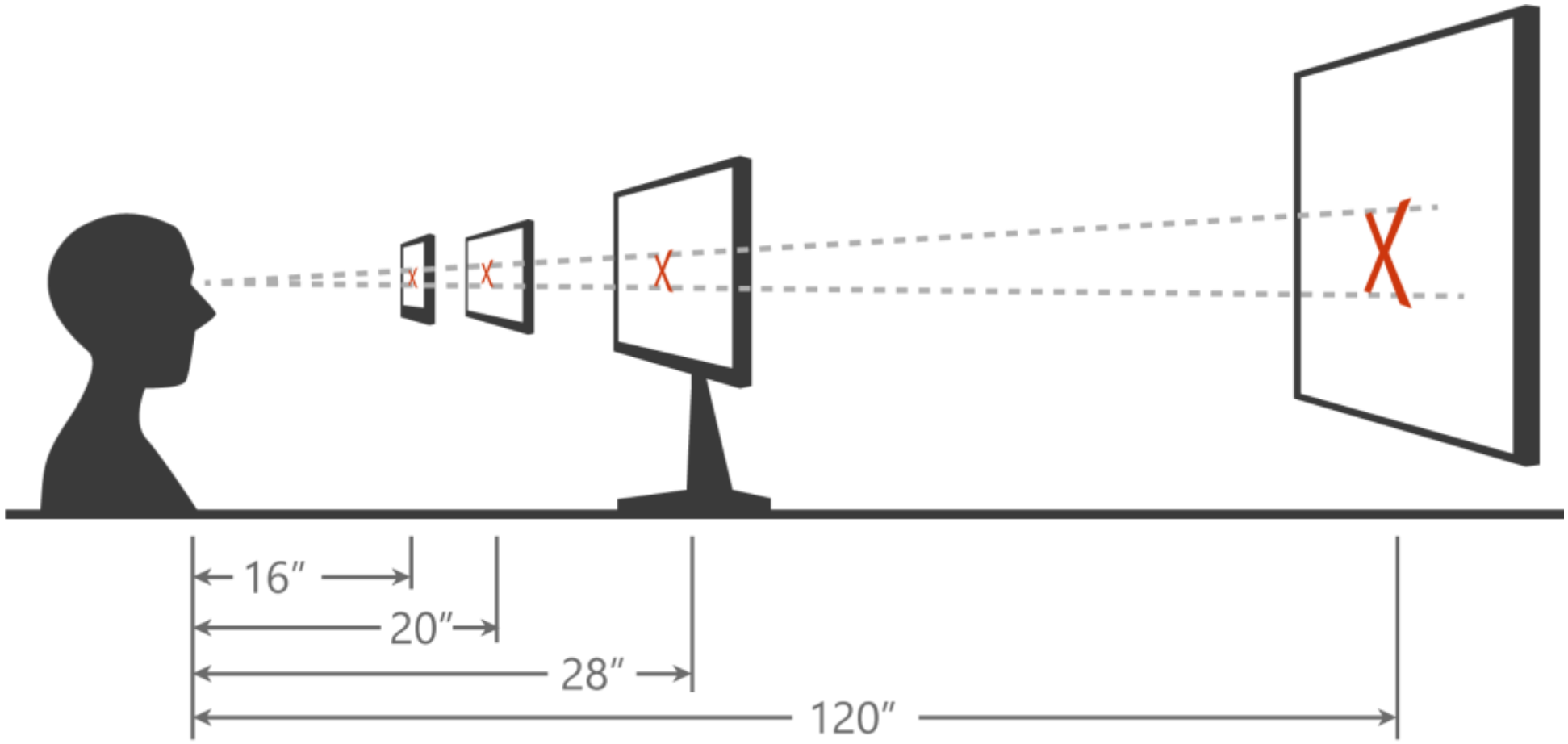
Many Apps need no Extension SDKs at all

The Windows Universal Core APIs cover nearly all common app needs

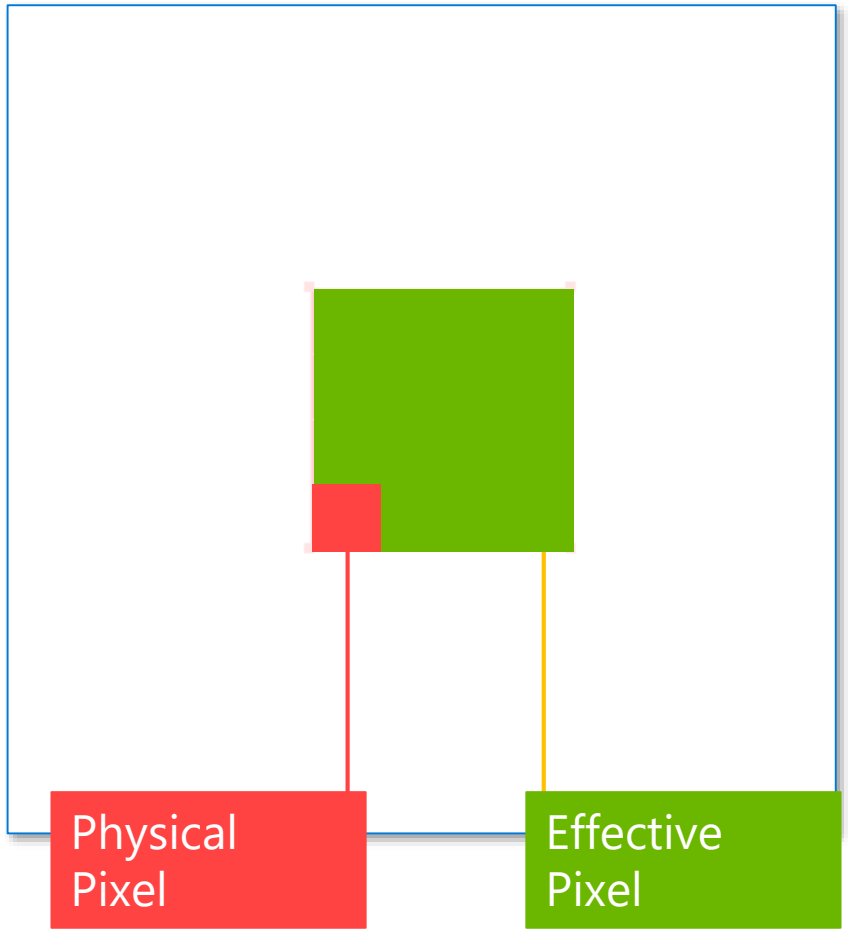
Use APIs in Extension SDKs to ‘light up’ your app when running on a specific device family

Adapting the UI

Scaling algorithm



Effective pixel

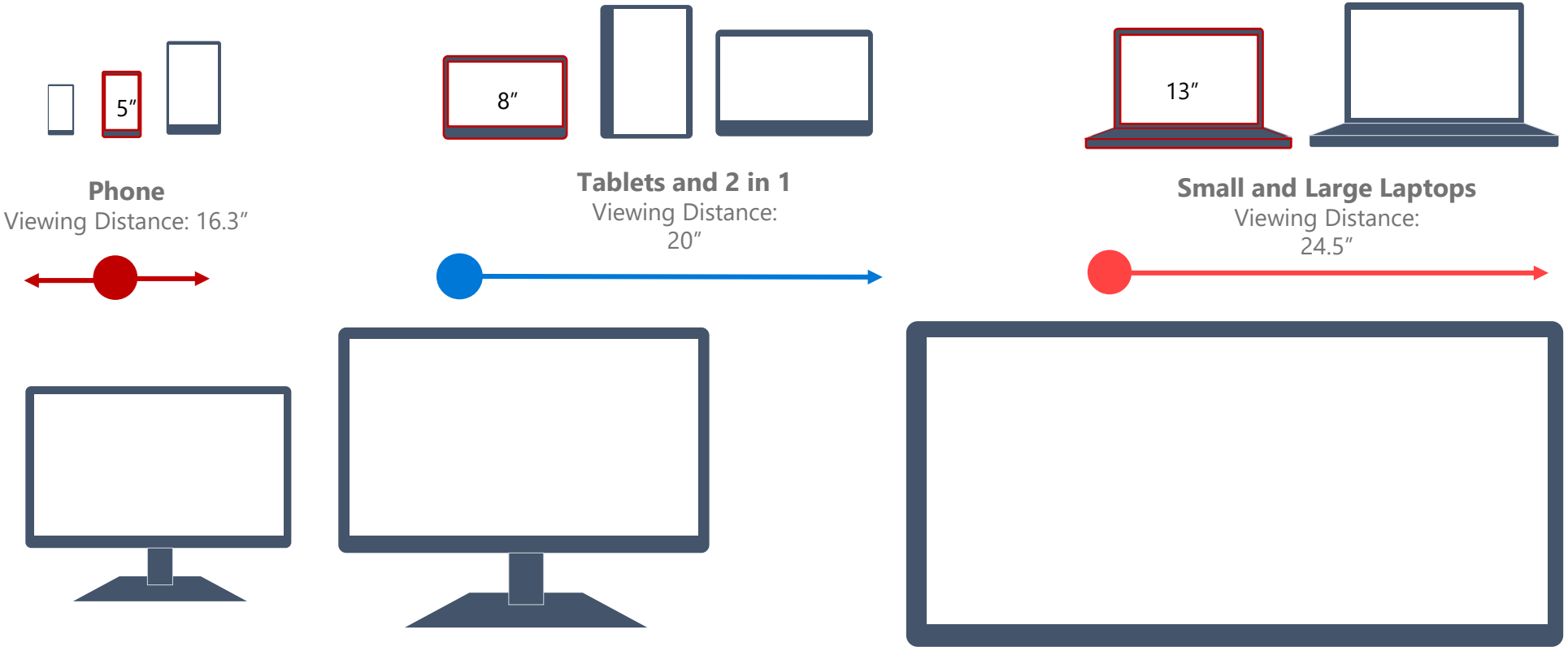


Ignore scale, resolution, & dpi.
Design in Effective Pixels

XAML is already in Effective Pixels

What am I designing?

Planning your design



Phone
Viewing Distance: 16.3″

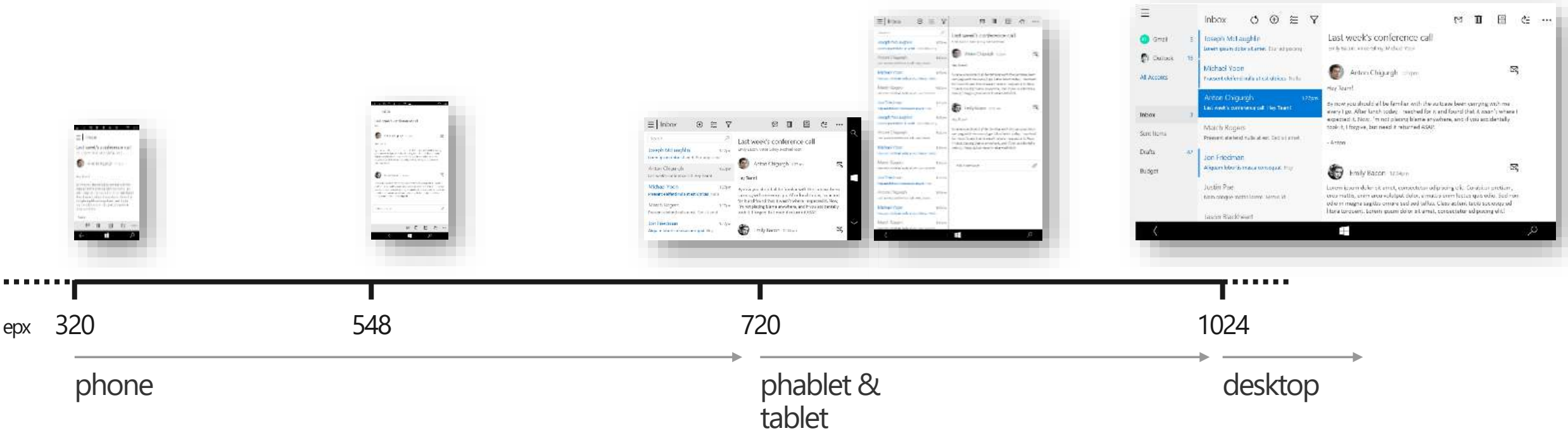
Tablets and 2 in 1
Viewing Distance:
20″

Small and Large Laptops
Viewing Distance:
24.5″

Small and Large Desktop Monitors
Viewing Distance:
28″

TV
Viewing Distance:
84″

Snap points



Design Techniques for Adaptive UI

Use standard responsive/adaptive design techniques

1 Reposition

2 Resize

3 Reflow

4 Reveal

5 Replace

6 Re-architect

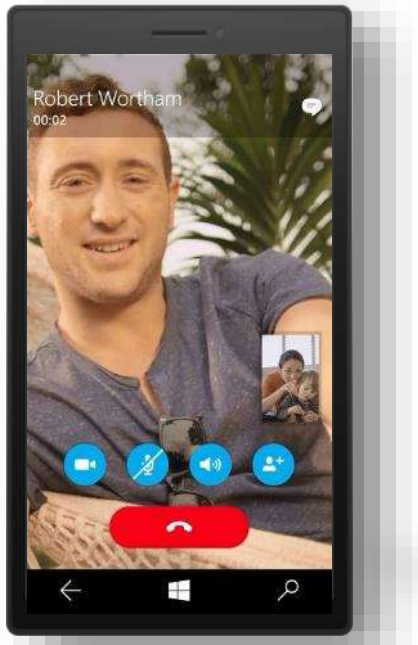
Adaptive design

Build a page that adapts to different screen sizes and orientations

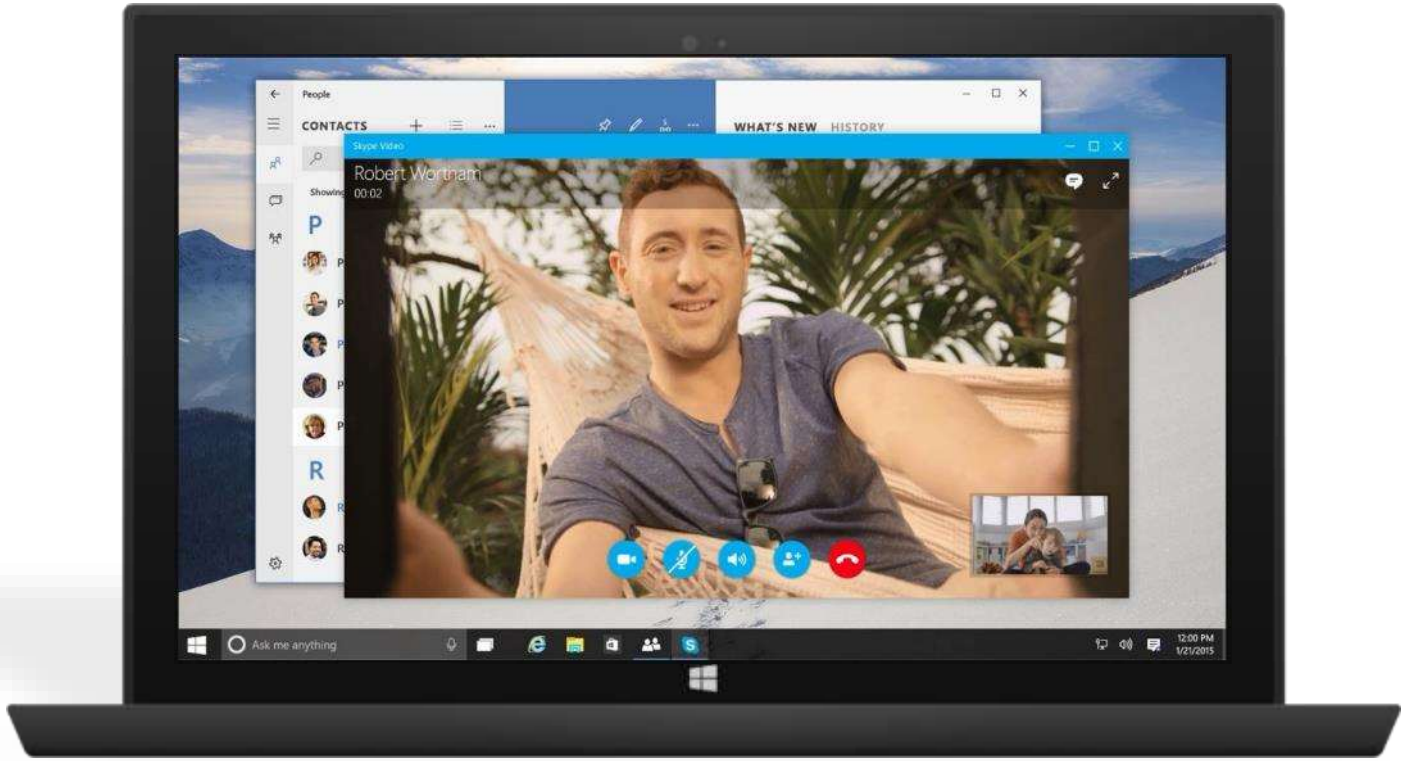
Use Visual States and Adaptive Triggers to change layout

Use RelativePanel to position blocks of content relative to peers, re-positioning in different visual states

Phone (portrait)



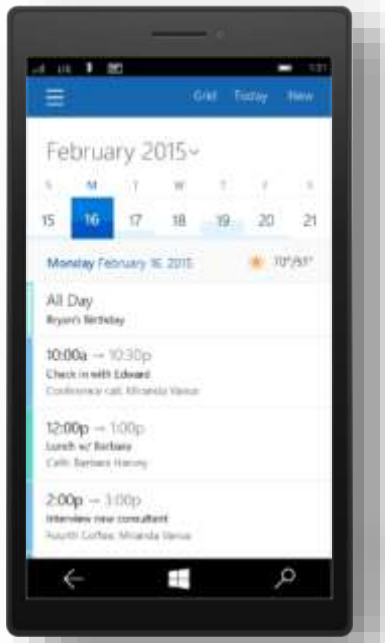
Tablet (landscape) / Desktop



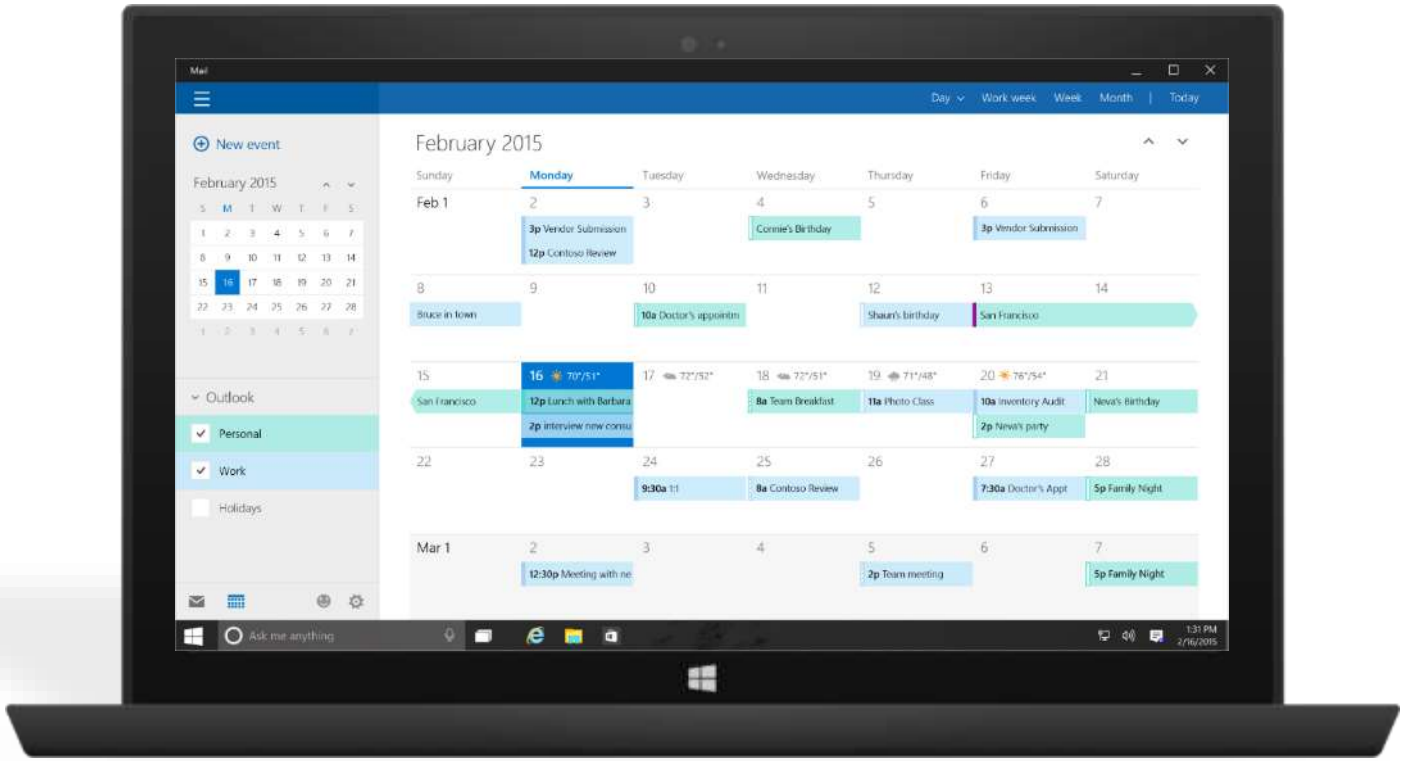
Tailored design

Build unique experiences on different devices

Phone (portrait)



Tablet (landscape) / Desktop

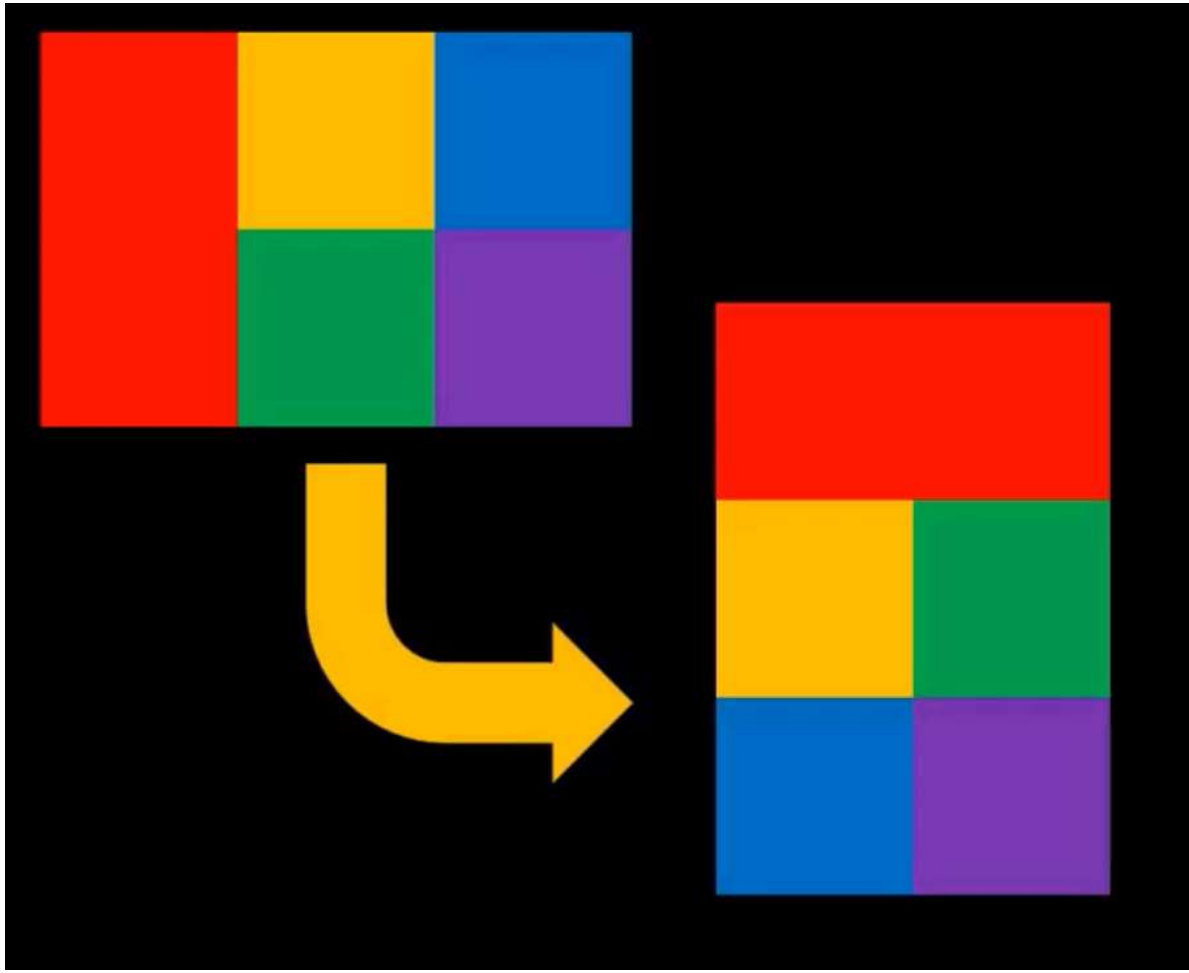


RelativePanel

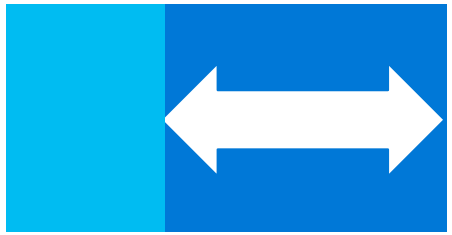
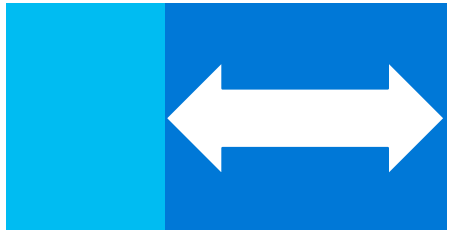
Some child elements act as anchors

Most child elements relate to others

It's a layout technique friendly with States



SplitView

	IsPaneOpen="True"	IsPaneOpen="False"
DisplayMode= "Inline"		
DisplayMode= "Overlay"		
DisplayMode= "CompactInline"		
DisplayMode= "CompactOverlay"		

Every Windows app will be compiled with .Net Native

.NET Native

Next generation compiler in the cloud

Apps use the standard C++ optimizer

As optimizer performance improves, so does .Net native

Apps with .Net bootstrapper

Includes garbage collection

There is no runtime

This is machine code

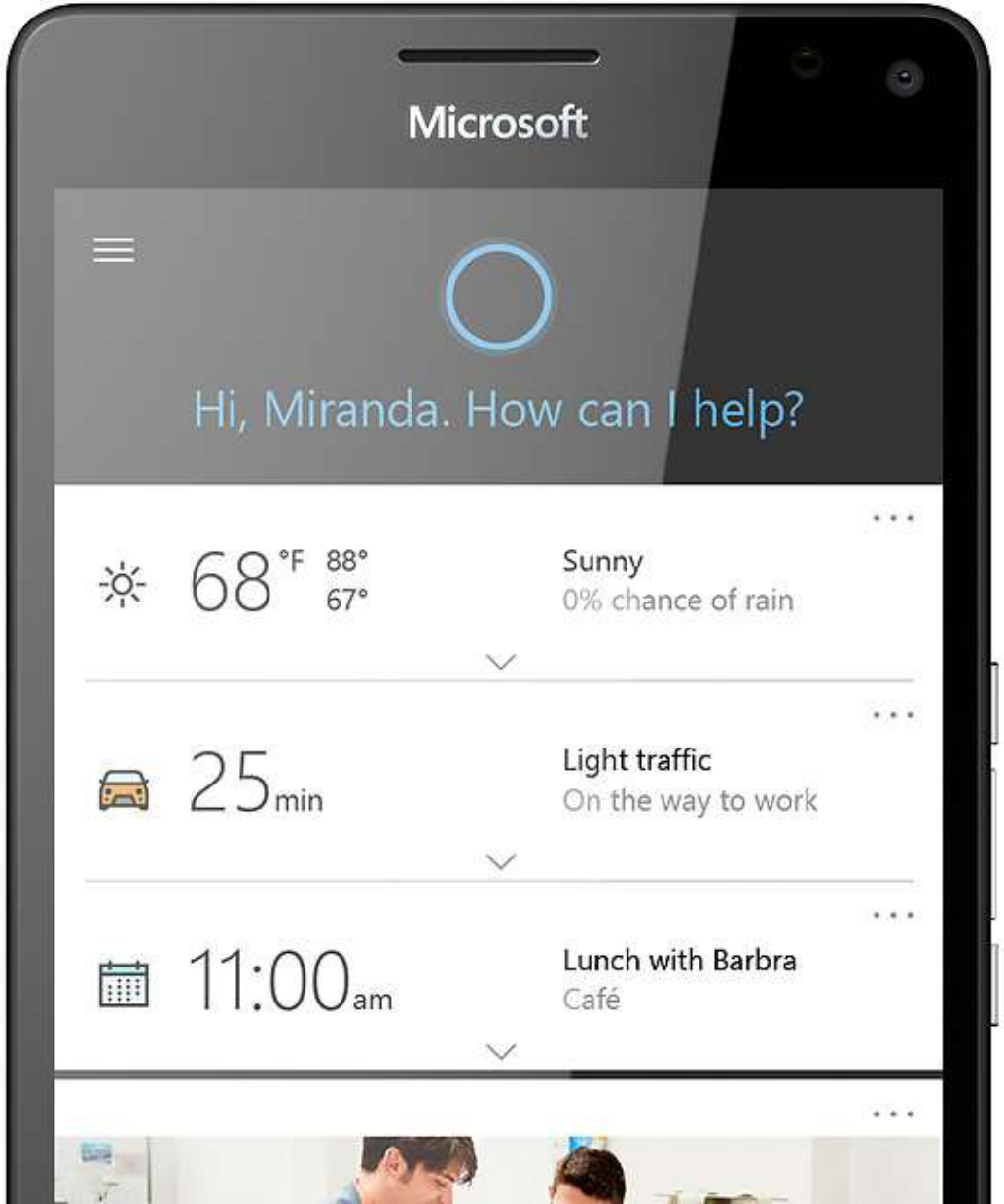
Real benefits with .Net Native

Faster average startup time

Less average memory usage

Cortana

Cortana



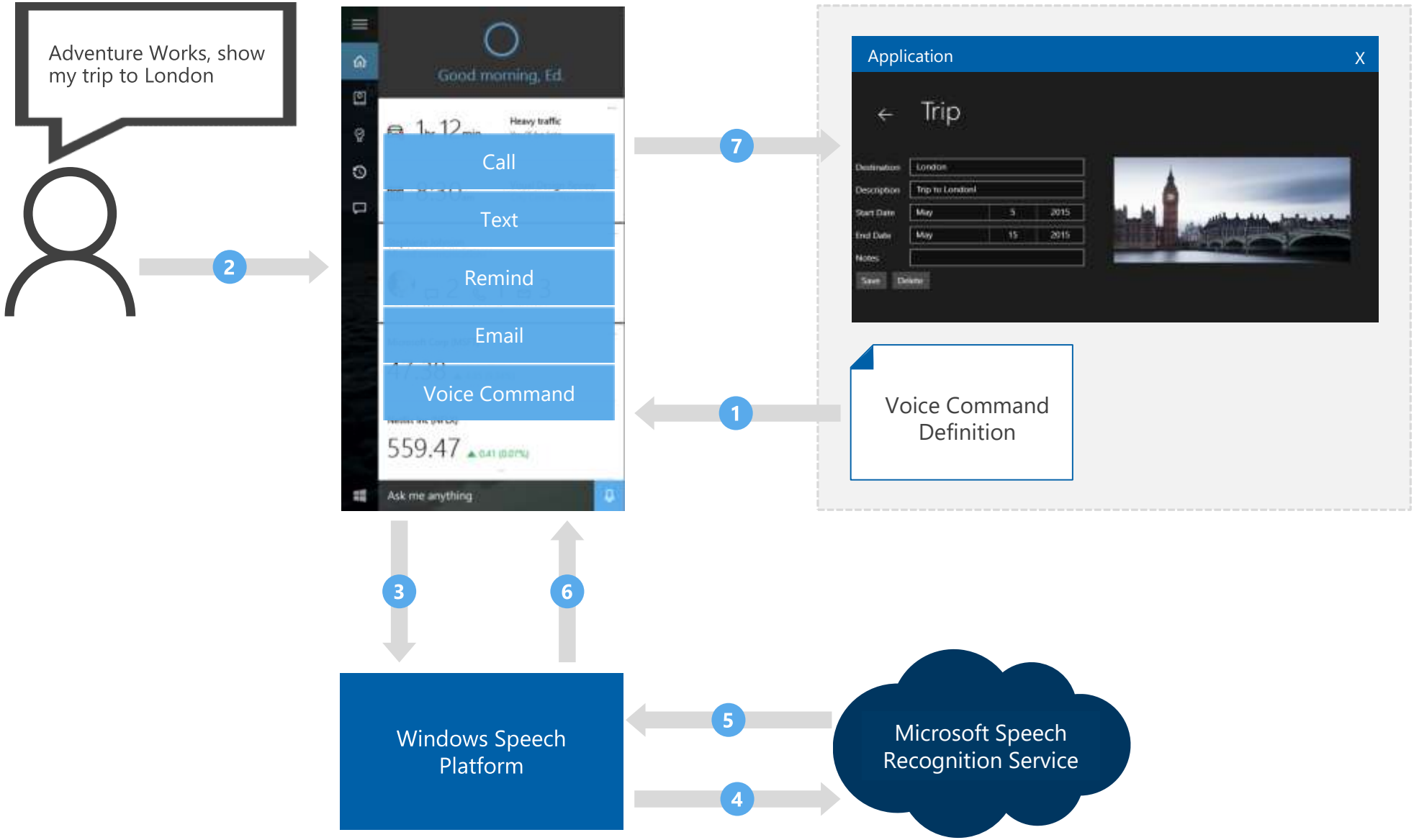
Personal assistant across all your devices

- Informative
- Reminders
- Contextual
- Voice Commands
- Extensible

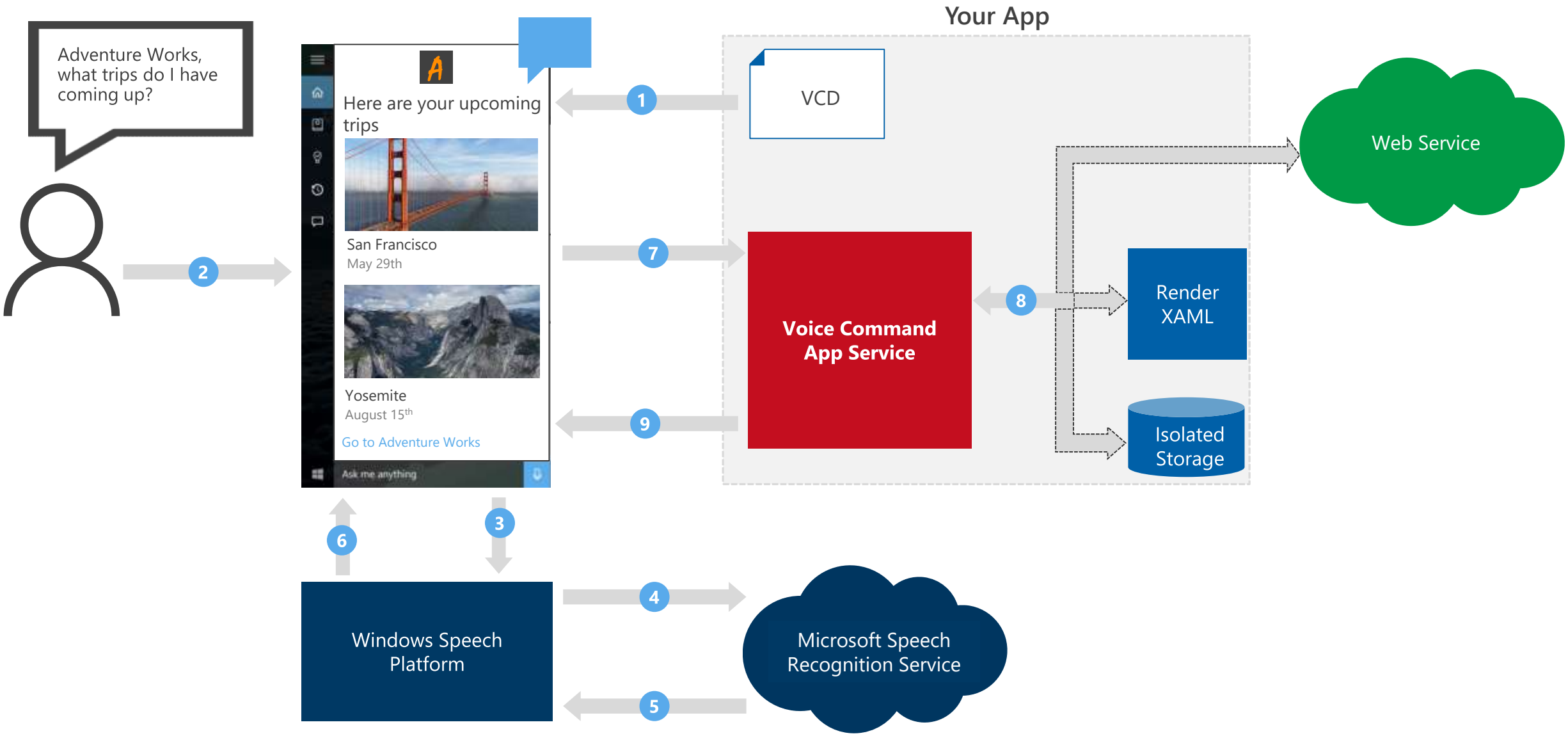
UWP Voice Commands

- Foreground
- Background

Foreground Voice Commands Architecture



Background Voice Commands Architecture



Bringing your code to Windows 10

Middleware Bridges

Wherever your code was born, you can bring it to Windows

Middleware platforms

Middleware (e.g., Xamarin)

Game engines (e.g., Unity)

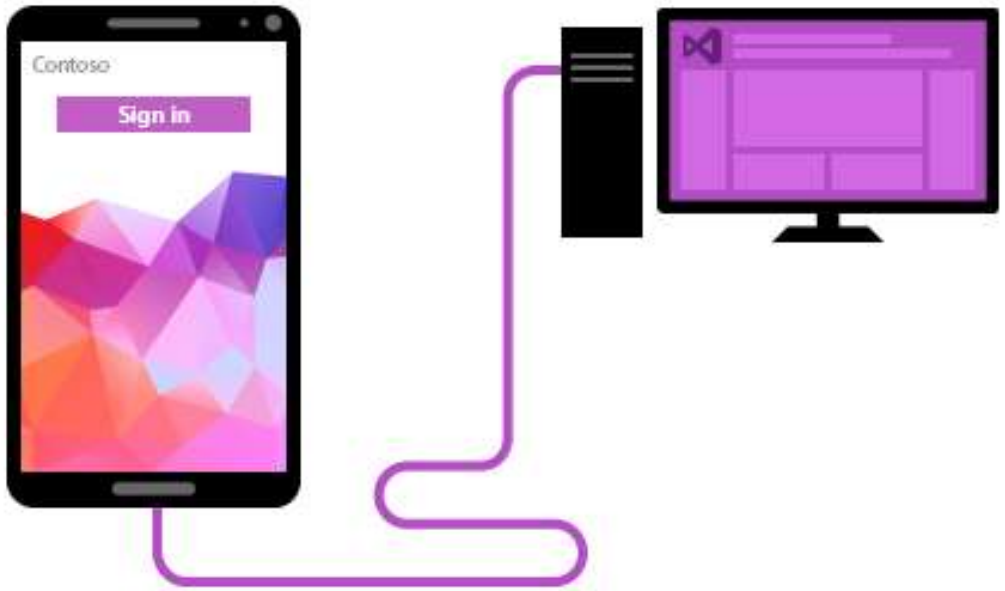
App middleware

Web Platform Bridges

Wherever your code was born, you can bring it to Windows

Visual Studio Tools for Apache Cordova

Build apps for Windows, iOS and Android using web technologies and Visual Studio



Visual Studio Tools for Apache Cordova

Single install

Code creation in Visual Studio

Preview and test on iOS, Android and Windows

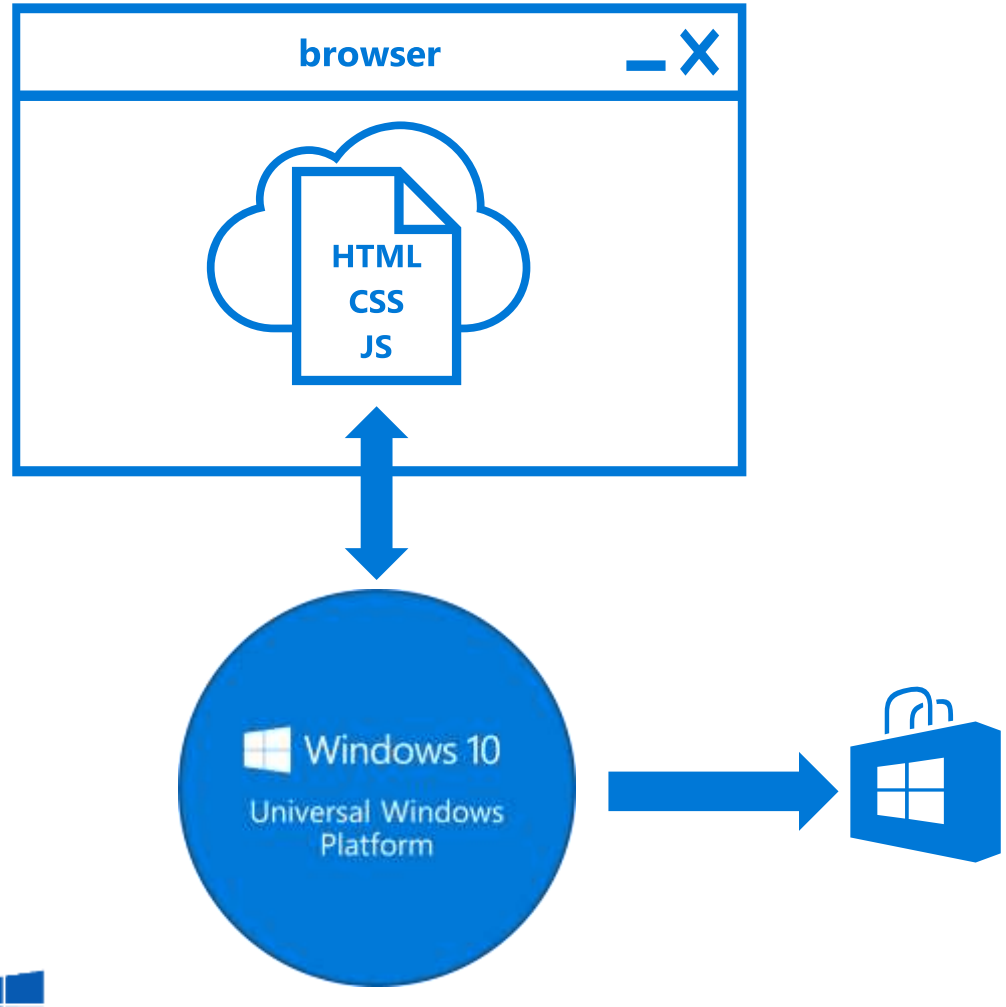
Debugging support in Visual Studio

Access native device capabilities

Use Cordova plugins within your solution to access native device capabilities

Hosted Web Apps

Extending your web site to the Universal Windows Platform on Windows 10



Hosted Web Apps

Support for websites hosted on a webserver

Packaging adds an app manifest

Microsoft Edge WebView rendering

Easier monetization + engagement

Windows Store aids discovery, install, update

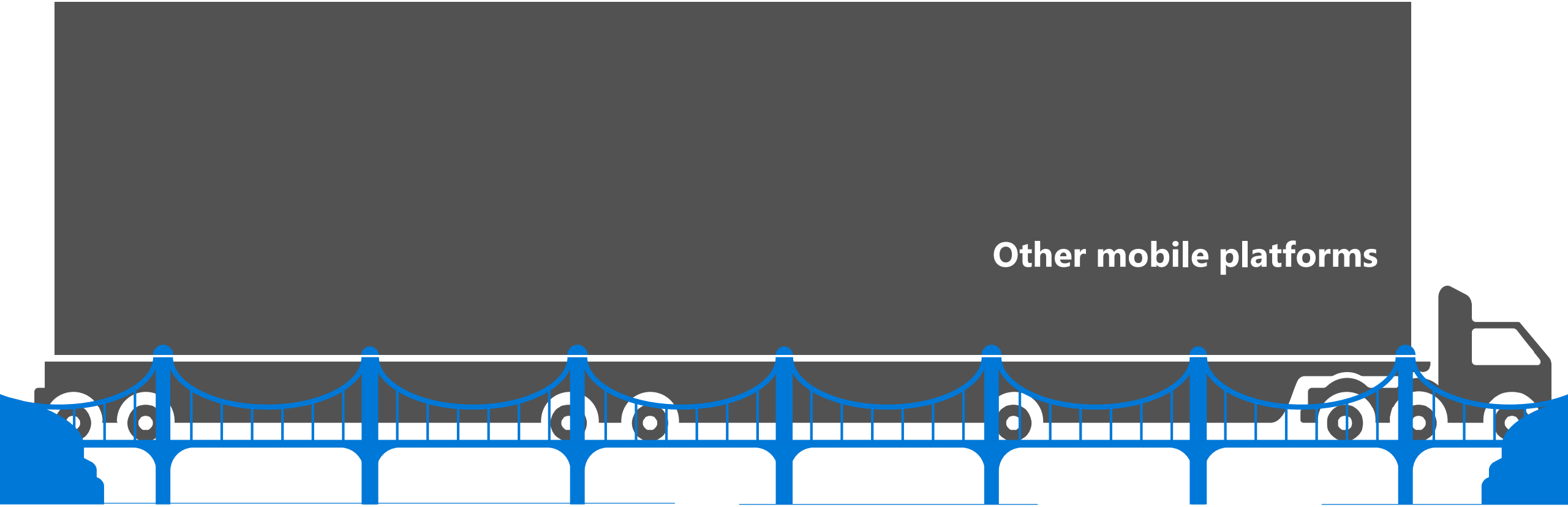
Updates happen via your webserver code/logic

Monetize using our payment instruments and APIs (subscription, IAP, ads, trials, etc.)

Use UWP APIs within your web code to use Live tiles, WNS, Geolocation, Cortana, and more

Mobile Platform Bridges

Wherever your code was born, you can bring it to Windows



What is the Windows Bridge for iOS

Objective-C language support

Compiler and Runtime

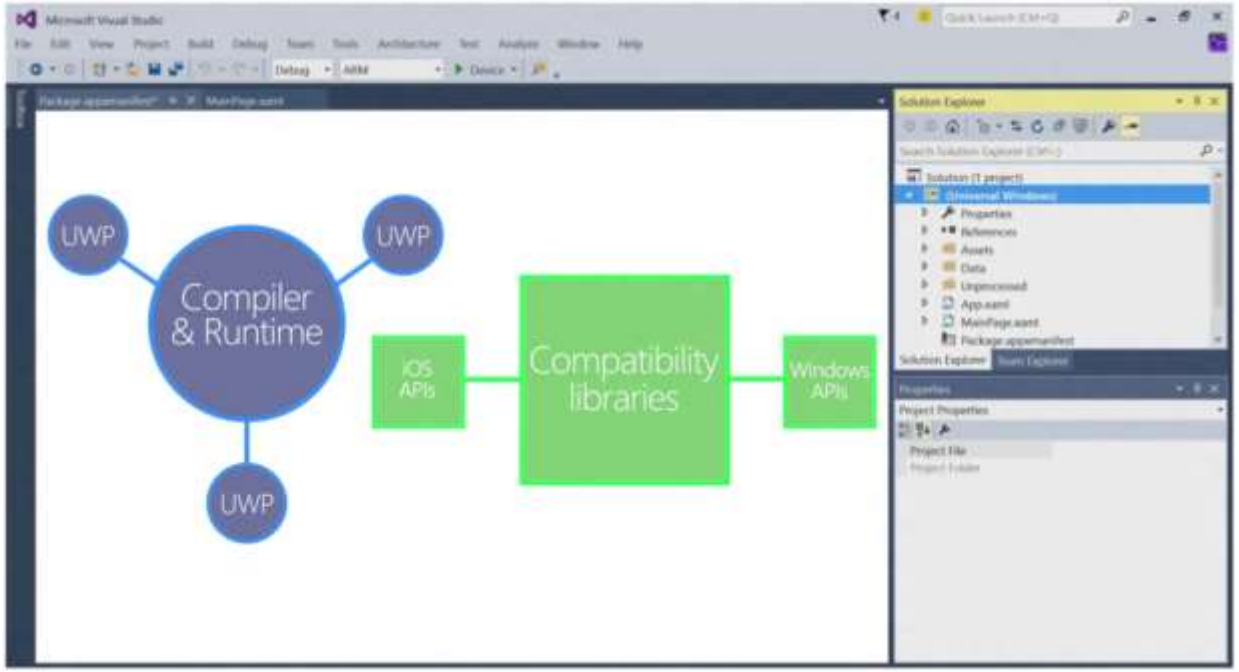
Useful and usable APIs

Compatibility libraries

Tooling

Visual Studio Editor / Workflow

Project import



API compatibility libraries

Supports a subset of iOS APIs

Does not track a particular version of iOS

Most used APIs are implemented first

Additions since open sourcing in August 2015:

- GLKit
- Xibs, AutoLayout, Storyboard
- KVO/KVC
- New sample projects

Open-source at https://github.com/Microsoft/WinObjC

Works across Win10 devices

Write a native UWP app

Across multiple form-factors

One app across IoT, phones, tablets, PCs, Xbox One, Surface Hub and HoloLens

Running ARM* and x86/64 CPUs

Using your existing Objective-C code

* ARM compiler support coming soon

Xamarin:
https://channel9.msdn.com/Events/Build/2016/B836

HockeyApp:
https://channel9.msdn.com/Events/Build/2016/P463