Microsoft

winhec

Workshop Taipei 2016

Any questions please contact winhectpe@microsoft.com

Microsoft

Moving to Universal Windows Platform:

Porting an App from Windows 8.1 XAML or Windows Phone Silverlight to Windows 10

Yu-Ping Huang / Helen Lo
Software Engineer
Partner Enablement Team

Agenda

- Why move to Universal Windows Platform (UWP)?
- Migration Path
- Porting Strategy
- General Porting Process
- Demo:
  - Windows Runtime 8.x to UWP
  - WP Silverlight to UWP

Why move your apps to UWP?

Goal: One Billion Devices.

Now Over 200 Million Devices!

The Number Is Still Growing!

- New features (Inking, Cortana, AllJoyn, …)
- Performance Improvements
- Single code base

Extend your app to multiple device families and use new capabilities by targeting the UWP

Migration Paths to Windows 10

Porting from…

- Windows Store 8.1 apps
- Windows Phone 8.1 apps
- Universal Windows 8.1 app (dual head)
- Windows Phone Silverlight apps

Porting from…

Windows Store 8.1 apps

Windows 8.0

Minimal code update required

Responsive UX Design/Implementation

Windows 10

Porting from…

Windows Phone 8.1 apps

Windows Phone 8.1

Minor code updates for UWP APIs

Design UX for multiple form factors

Windows 10

Porting from…

Universal Windows 8.1 apps (dual head)

Windows 8.1

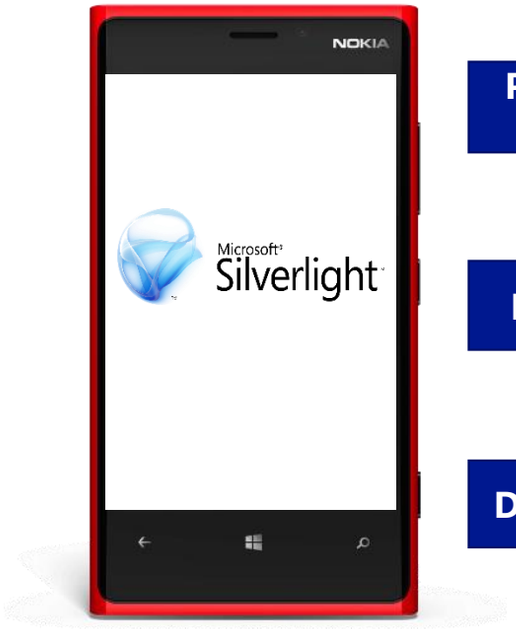
Windows Phone 8.1

Merge UX

Refactor to single code-base & to target UWP APIs

Windows 10

Porting a Windows Silverlight app



Port the UI Silverlight -> Windows XAML

Rewrite code to target UWP APIs*

Design UX for multiple form factors

Windows Phone 7.5/7.8

Windows Phone 8.0

Windows Phone 8.1*

Windows 10

Porting – What You Can Expect

8.1 WinRT app code needs few changes

- App lifecycle, background execution, Tiles and toasts – all the same
- UWP APIs (Universal API set + Extension SDK API set) are a superset of the Windows 8.1 WinRT APIs
- Review/change logic that relied on compiler conditionals (#if…) to handle platform differences
- A few APIs are deprecated (example, Phone 8.1 …AndContinue APIs)
- Charms bar gone, so app must now incorporate UI to launch Settings, Share or Search

8.1 WinRT XAML UI ports across fairly easily

- Build adaptive UI working across multiple device families
- Care needed with deprecated and altered styles and with font size changes

Silverlight 7.x/8.x apps need reimplementation

- Though these apps still run on Windows 10 Mobile devices!
- Porting Tooling

Develop a Porting Strategy

Upgrading Options

1. Use a single UWP to target all devices
2. Create separate UWPs targeting device families (Desktop/Mobile/Xbox/IoT/…)

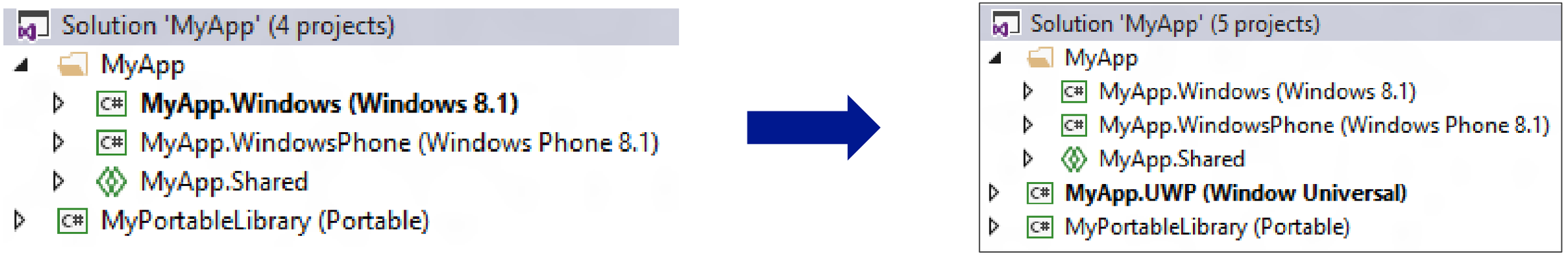
Example
Single UWP App to target Mobile and Single UWP to target all others

Example Manifest entry

```
<Dependencies>
    <TargetDeviceFamily Name="Windows.Mobile"
                        MinVersion="10.0.1.0" MaxVersionTested="10.0.1.0" />
```

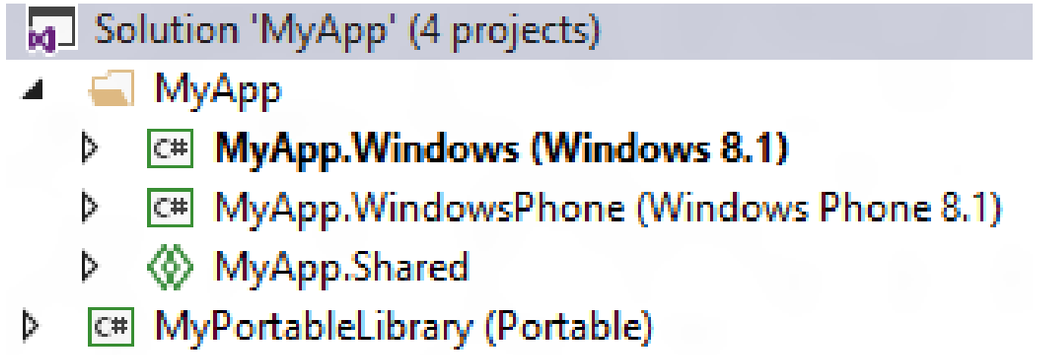
Strategy Option 1

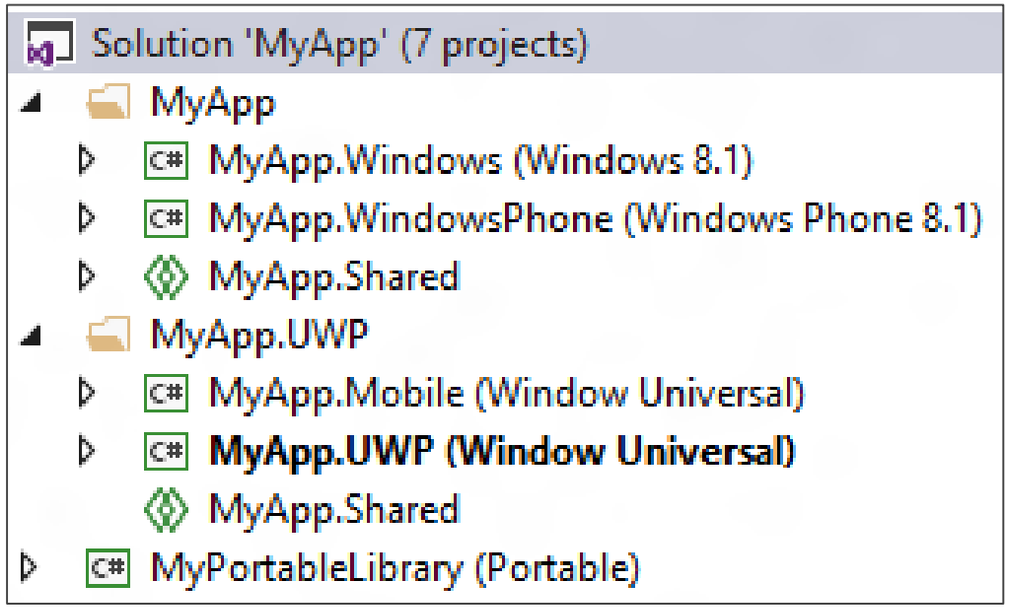
- Add a single UWP Project
- Merge all code into single solution, create single adaptable views
- Self contained not leveraging shared code



Strategy Option 2

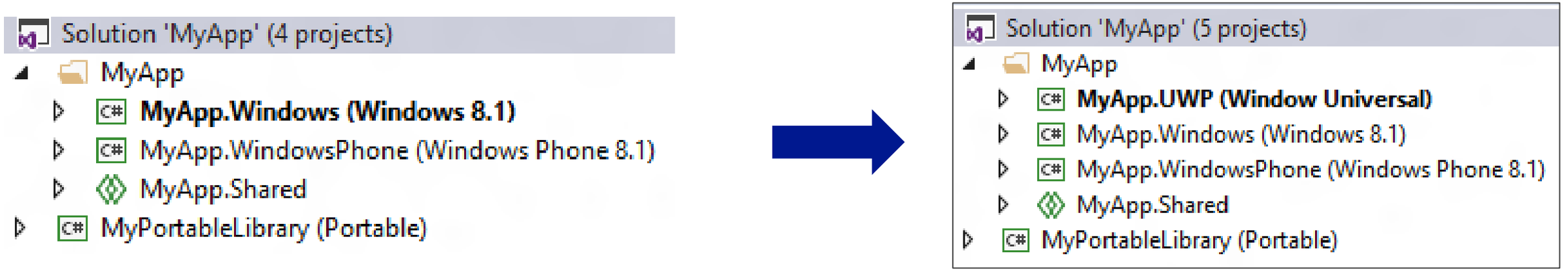
- Create separate UWPs targeting device families
  - One for Mobile
  - One for all others
- Self contained leveraging own set of shared code





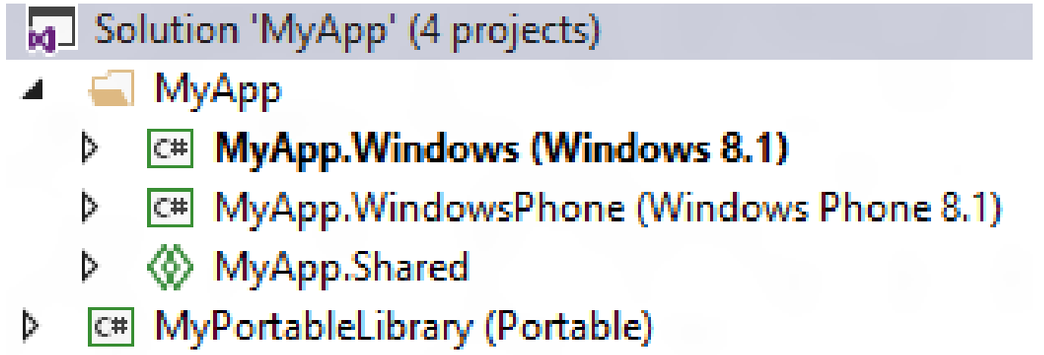
Strategy Option 3

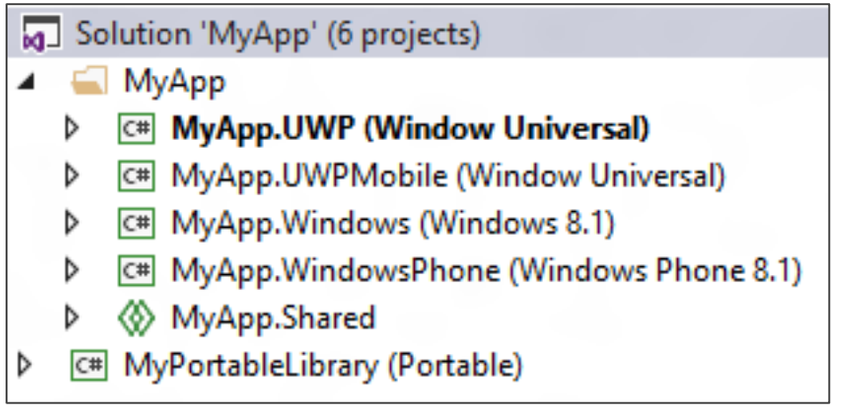
- Add a new UWP “Head”
- Shares same code as 8.1 projects
  - Use #if’s for API differences in shared code if needed



Strategy Option 4

- Add multiple new UWP “Heads”
  - One for Mobile
  - One for all others
- Shares same code as 8.1 projects
  - Use #if’s for API differences in shared code if needed





Areas to address while porting

- Build Breaks
  - APIs moved between UWP and extension SDKs (ex: camera button)
  - Deprecated APIs (ex: no charms bar for 10.0 UWP apps)
  - UWP app project does not support AnyCPU (.NET Native)
- Runtime breaks
  - Missing XAML resource styles
    - Phone* resources gone (ex: PhoneAccentBrush)
    - See Dev Center for Design Guidelines and type ramp
- User Experience Breaks
  - Upgrade breaking changes to Controls
    - Design Refresh
    - ex: GridView now flows vertically instead of horizontal

General Porting Process

- Create a new UWP project
- Copy code to new project
- Add References if applicable
- Modify Manifest file (add capabilities, BG Tasks,…)
- Build -> Fix -> Build -> repeat…

Review #if conditional compilation

Compiler conditionals may be used in shared code:

```
            this.Page.Loaded += (sender, e) =>
            {
#if WINDOWS_PHONE_APP
                Windows.Phone.UI.Input.HardwareButtons.BackPressed += HardwareButtons_BackPressed;
#else
                // Keyboard and mouse navigation only apply when occupying the entire window
                if (this.Page.ActualHeight == Window.Current.Bounds.Height &&
                    this.Page.ActualWidth == Window.Current.Bounds.Width)
                {
                    // Listen to the window directly so focus isn't required
                    Window.Current.CoreWindow.Dispatcher.AcceleratorKeyActivated +=
                        CoreDispatcher_AcceleratorKeyActivated;
                    Window.Current.CoreWindow.PointerPressed +=
                        this.CoreWindow_PointerPressed;
                }
#endif
            };
```

Handling Back Button

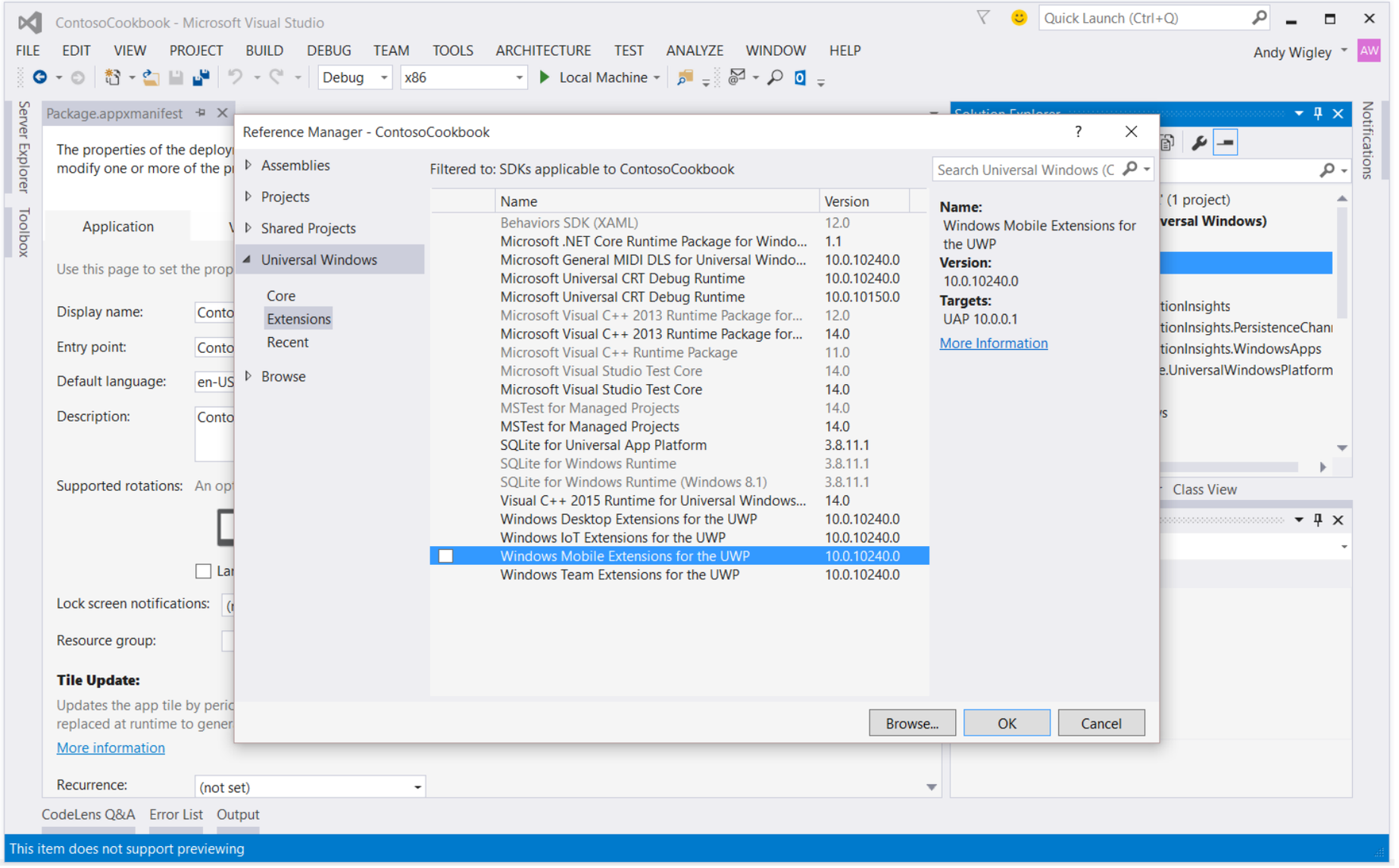
Desktop: Enable Chrome back button

Single converged back button API

Example:

```
protected override void OnNavigatedTo(NavigationEventArgs e)
{
    var frame = Window.Current.Content as Frame;
    if (frame.CanGoBack)
    {
        SystemNavigationManager.GetForCurrentView().AppViewBackButtonVisibility =
                                                    AppViewBackButtonVisibility.Visible;
    }
    else
    {
        SystemNavigationManager.GetForCurrentView().AppViewBackButtonVisibility =
                                                    AppViewBackButtonVisibility.Collapsed;
    }

    SystemNavigationManager.GetForCurrentView().BackRequested += Page_BackRequested;
}
```

Add Ref for Platform extensions



Adaptive code

```
var api = "Windows.Phone.UI.Input.HardwareButtons";
if (Windows.Foundation.Metadata.ApiInformation.IsTypePresent(api))
{
   Windows.Phone.UI.Input.HardwareButtons.CameraPressed
        += CameraButtonPressed;
}
```

Replace Deprecated APIs

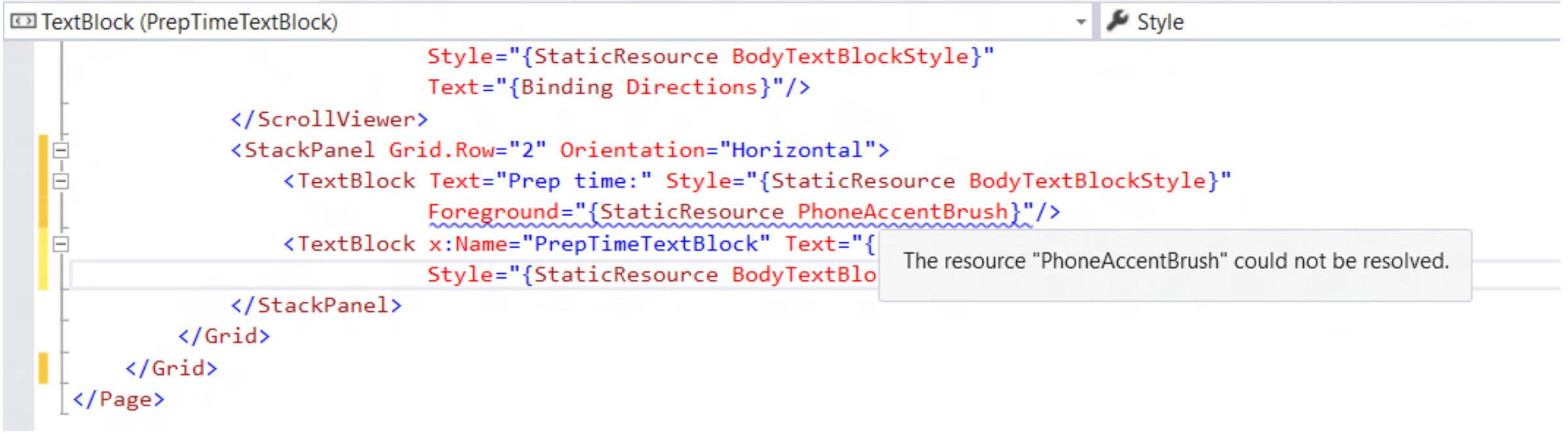
```
1 reference
private async void PhotoButton_Click(object sender, RoutedEventArgs e)
{
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.ViewMode = PickerViewMode.Thumbnail;
    openPicker.SuggestedStartLocation = PickerLocationId.PicturesLibrary;
    openPicker.FileTypeFilter.Add(".jpg");
    openPicker.FileTypeFilter.Add(".jpeg");
    openPicker.FileTypeFilter.Add(".png");

    // Call the FileOpenPicker in Windows Phone mode
    openPicker.PickSingleFileAndContinue();
}
```

void FileOpenPicker.PickSingleFileAndContinue()

'FileOpenPicker.PickSingleFileAndContinue()' is obsolete: 'PickSingleFileAndContinue will be unavailable for Windows Phone 10.

Review all use of Styles and Sizes

Replace undefined Styles in XAML with alternatives:



Scaling gotcha's

Old scale factors may be larger now

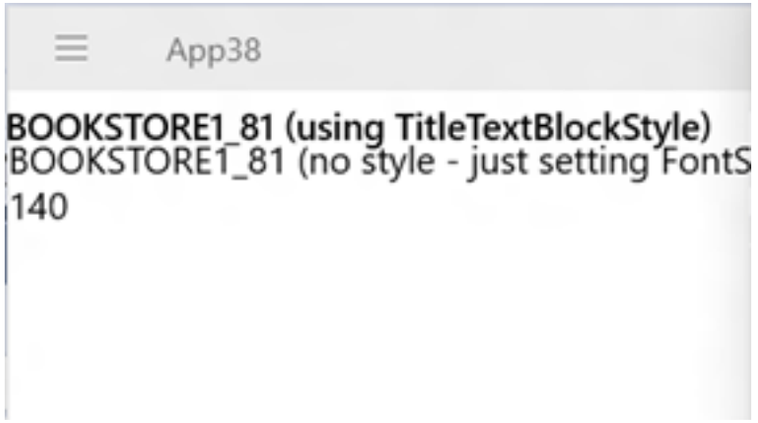
140% -> 150% , 180% - > 200%

Visual refresh has changed font sizes

Ex: TitleTextBlockStyle

(8.1) FontSize=14.667 (10.0) FontSize=24

Example (left 8.1 App, Right 10.0)



Update Charms Bar Integration Code

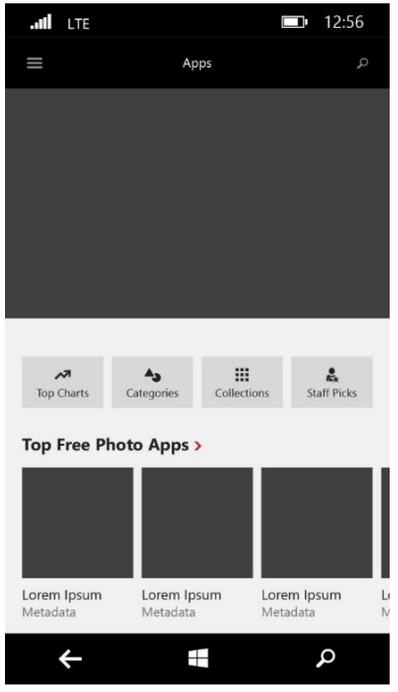
Charms bar not on Windows 10 devices

Replace with in-app UI for

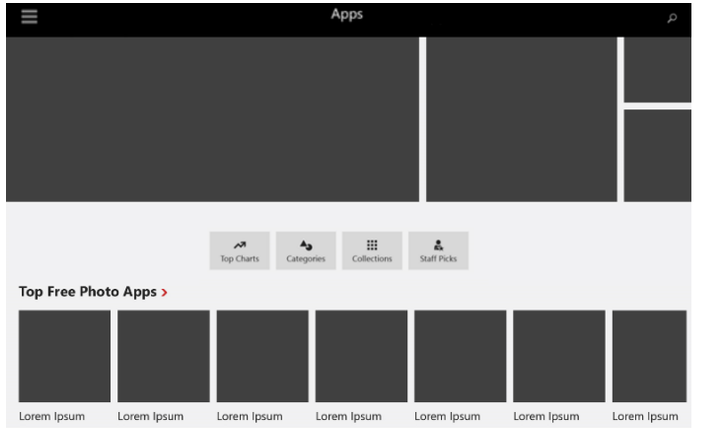
- Search
- App Settings
- Sharing

Underlying code does not change, just the users' way of accessing them

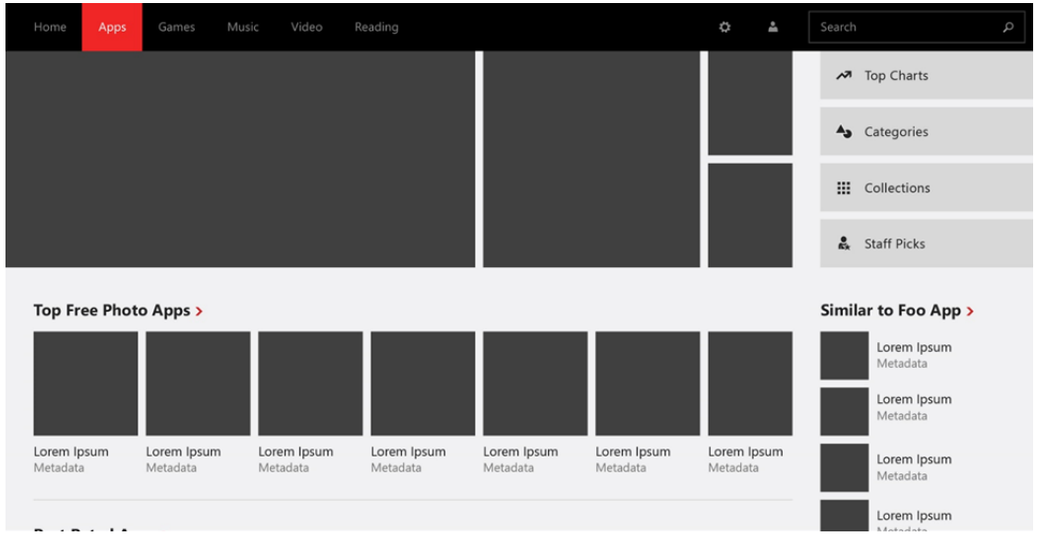
Create an awesome adaptive UI!



Phone/Narrow view



Small landscape view



Large landscape view

Demo 1 -
Migrating a real Windows 8.1 Universal app to UWP

Demo 2 -
Migrating a real Silverlight Phone app to UWP

References:

- Move from Windows Runtime 8.x to UWP
  https://msdn.microsoft.com/en-us/library/windows/apps/mt238322.aspx
- Move from Windows Phone Silverlight to UWP
  https://msdn.microsoft.com/en-us/library/windows/apps/mt238323.aspx
- Developer's Guide to Windows 10 > Porting 8.1 apps
  https://channel9.msdn.com/Events/Windows/Developers-Guide-to-Windows-10-RTM/Porting-81-apps
- Build 2015 > Moving to the Universal Windows Platform: Porting an App from Windows 8.1 XAML or Windows Phone Silverlight to Windows 10
  https://channel9.msdn.com/events/Build/2015/3-741
- Reference for Universal Windows apps
  https://msdn.microsoft.com/en-us/library/windows/apps/bg124285.aspx
- Device Families
  https://msdn.microsoft.com/en-us/library/windows/apps/dn706137.aspx
- .NET Native – What it means for Universal Windows Platform (UWP) developers
  https://blogs.windows.com/buildingapps/2015/08/20/net-native-what-it-means-for-universal-windows-platform-uwp-developers/
- Demo:
  Demo 1. Windows Phone Silverlight to UWP case study: Bookstore1
  https://msdn.microsoft.com/en-us/library/windows/apps/mt188207.aspx
  Demo 2. Windows Runtime 8.x to UWP case study: Bookstore2
  https://msdn.microsoft.com/en-us/library/windows/apps/mt188200.aspx

Calls to Action

Join WinHEC LINE Community
@winhec

We want to hear from you!

Please Complete the Evaluation Form and return it to our reception.

Your input is highly important to us!
Thank you!! ☺

Microsoft