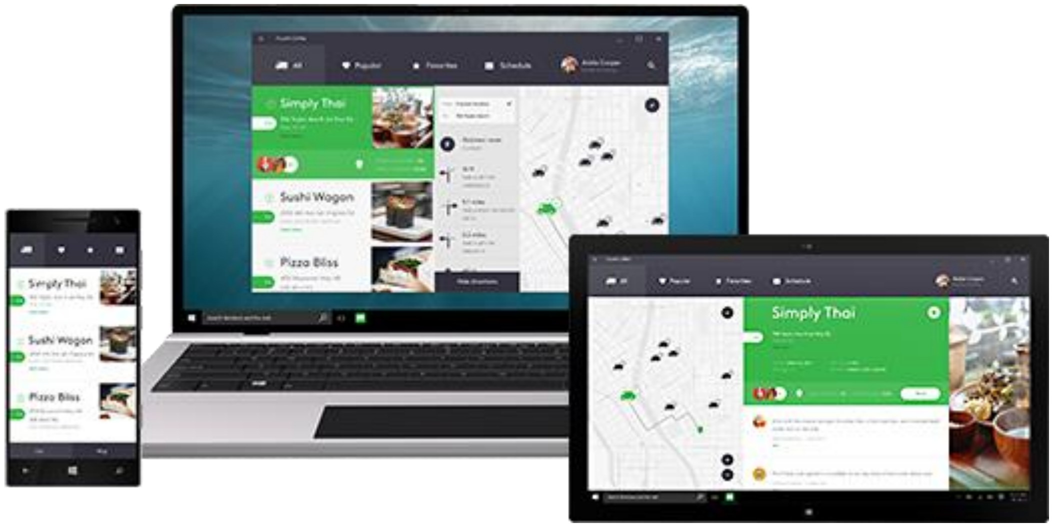
UWP Apps
Universal Windows Platform
For Windows 10



AARHUS
UNIVERSITY
SCHOOL OF ENGINEERING

One Windows Platform

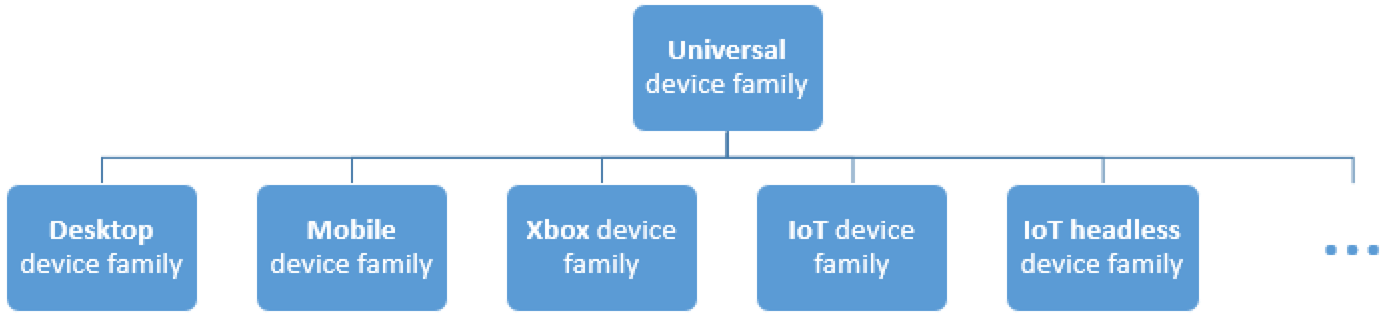
Many devices

Windows 10
Collaborative
Wearable
IoT
Mobile
Desktop
Console

The Universal Windows Platform

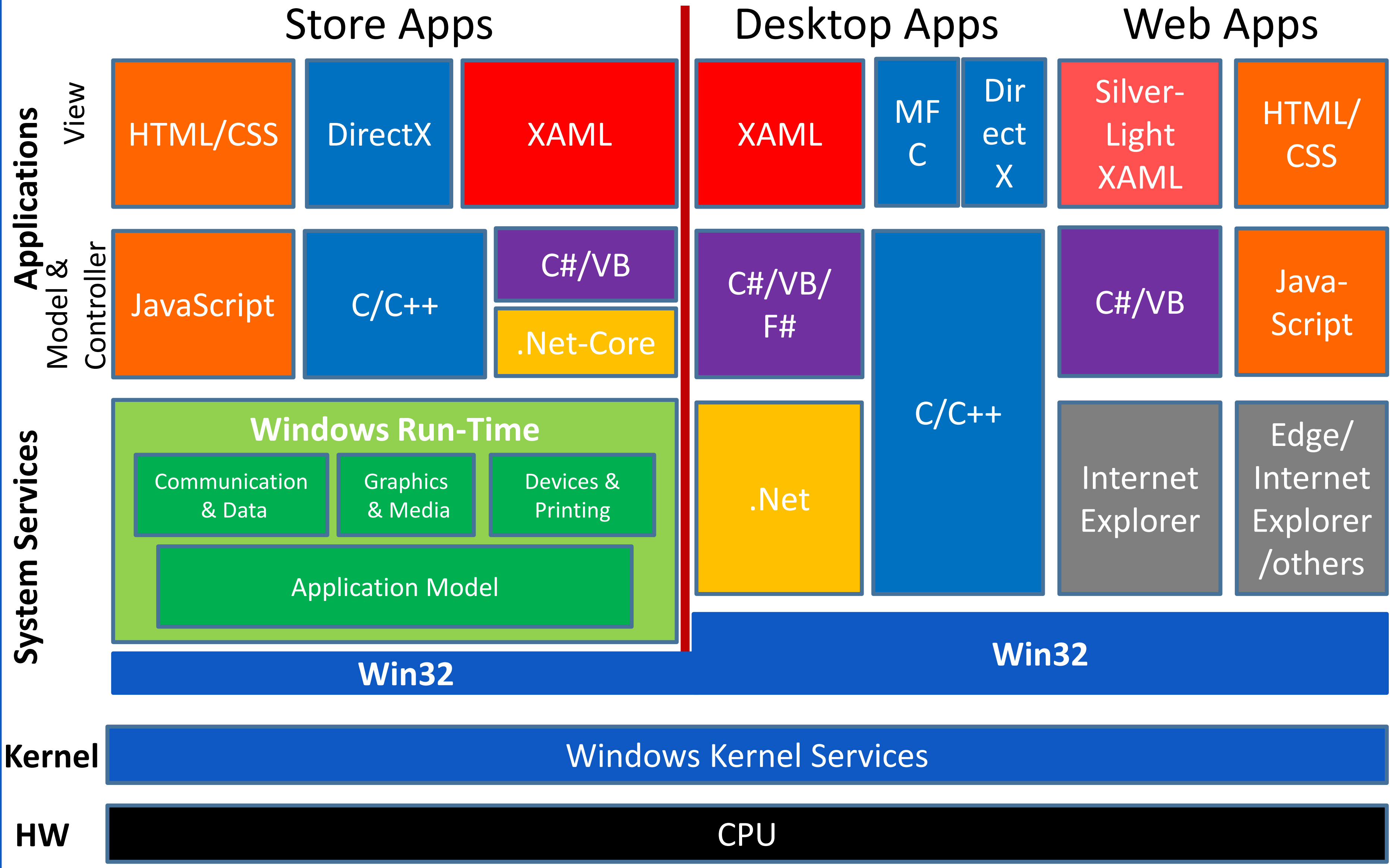
- UWP provides a common app platform available on every device that runs Windows 10
- Apps that target the UWP can call not only the WinRT APIs that are common to all devices
  - but also APIs (including Win32 and .NET APIs) that are specific to the device family the app is running on
- The UWP provides a guaranteed core API layer across devices
  - This means you can create a single app package that can be installed onto a wide range of devices
  - And the Windows Store provides a unified distribution channel to reach all the device types your app can run on
- Adaptive UI controls and new layout panels help you to tailor your UI across a broad range of screen resolutions

Device Family



- A device family identifies:
  - the APIs
  - system characteristics
  - behaviors
- It also determines the set of devices on which your app can be installed from the Store
- Each child device family adds its own APIs to the ones it inherits
- The decision about which device family (or families) your app will target is yours

Windows Application Types



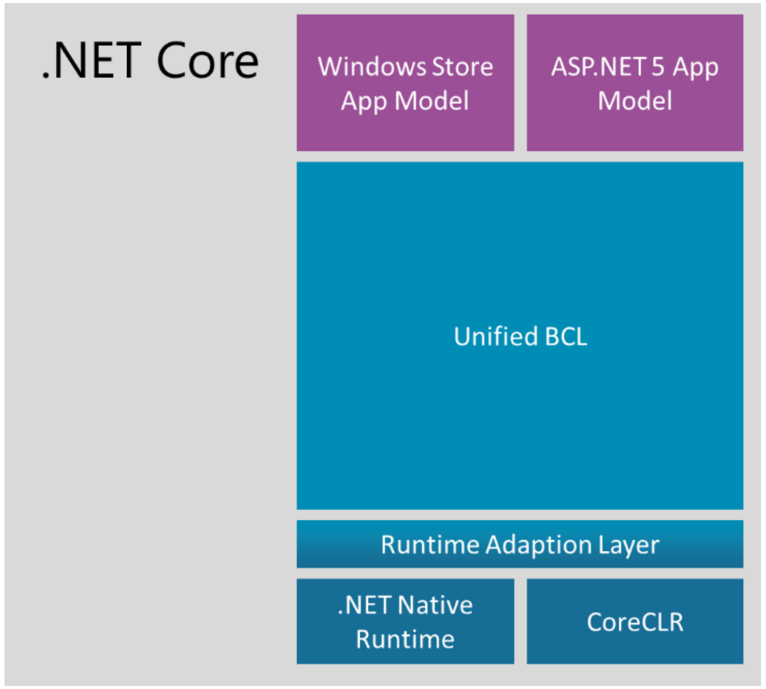
AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

.NET Core

- Is Microsofts open-source cross-platform implementation of .Net
- Is a modular runtime and library implementation that includes a subset of the .NET Framework
  - Is not a full port of .Net!
- Consists of a set of libraries, called “CoreFX”, and a small, optimized runtime, called “CoreCLR”
- The CoreCLR runtime and CoreFX libraries are distributed via NuGet
  - The CoreFX libraries are factored as individual NuGet packages according to functionality
- .Net Core can run console and ASP.Net applications on Windows, Linux, and Mac OSX
- UWP apps runs on .NET Core but only on Windows
  - Unless you use Xamarin

Unified implementation for .NET Native and ASP.NET

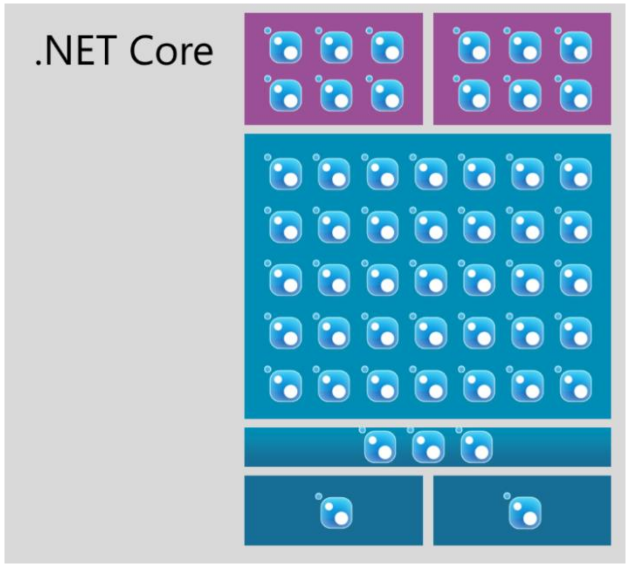
- On top of the BCL, there are app-model specific APIs. For instance, the .NET Native side provides APIs that are specific to Windows client development, such as WinRT interop. ASP.NET 5 adds APIs such as MVC that are specific to server-side web development



- At the bottom of the BCL we have a very thin layer that is specific to the .NET runtime. We've currently two implementations: one is specific to the .NET Native runtime and one that is specific to CoreCLR, which is used by ASP.NET Core

NuGet as Delivery Vehicle

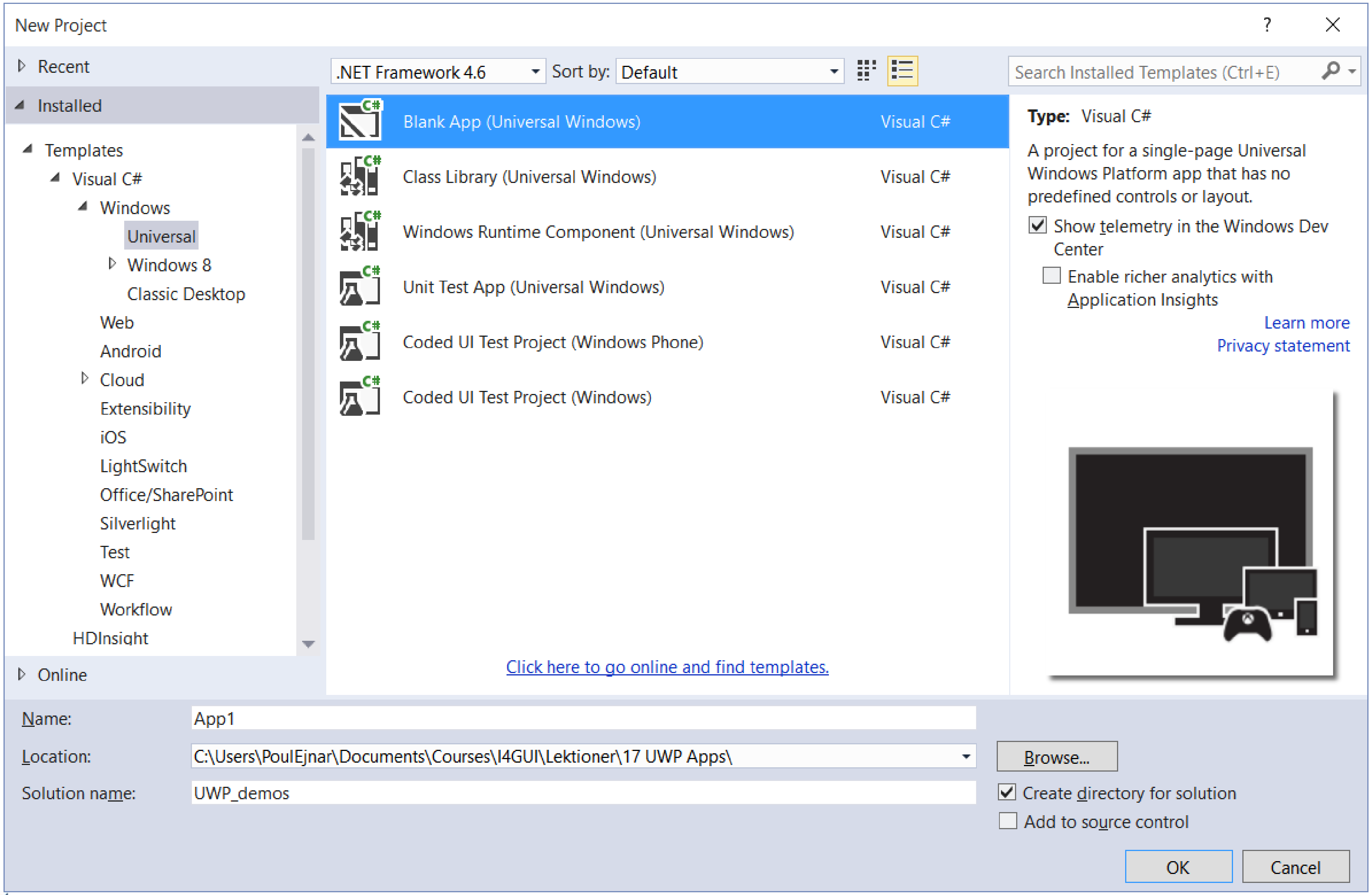
- In contrast to the .NET Framework, the .NET Core platform will be delivered as a set of NuGet packages
- The entire .NET Core platform is delivered as a set of fine grained NuGet packages
- NuGet allows MS to deliver .NET Core in an agile fashion
  - So if MS provide an upgrade to any of the NuGet packages, you can simply upgrade the corresponding NuGet reference



Project.json

```
{
  "dependencies": {
    "Microsoft.NETCore.UniversalWindowsPlatform": "5.0.
  },
  "frameworks": {
    "uap10.0": {}
  },
  "runtimes": {
    "win10-arm": {},
    "win10-arm-aot": {},
    "win10-x86": {},
    "win10-x86-aot": {},
    "win10-x64": {},
```

The Project Templates in VS



New Project

- Recent
- Installed
  - Templates
    - Visual C#
      - Windows
        - Universal
        - Windows 8
        - Classic Desktop
      - Web
      - Android
      - Cloud
      - Extensibility
      - iOS
      - LightSwitch
      - Office/SharePoint
      - Silverlight
      - Test
      - WCF
      - Workflow
    - HDInsight
- Online

.NET Framework 4.6 | Sort by: Default | Search Installed Templates (Ctrl+E)

Blank App (Universal Windows)	Visual C#
Class Library (Universal Windows)	Visual C#
Windows Runtime Component (Universal Windows)	Visual C#
Unit Test App (Universal Windows)	Visual C#
Coded UI Test Project (Windows Phone)	Visual C#
Coded UI Test Project (Windows)	Visual C#

Click here to go online and find templates.

Type: Visual C#

A project for a single-page Universal Windows Platform app that has no predefined controls or layout.

☑ Show telemetry in the Windows Dev Center

☐ Enable richer analytics with Application Insights

Learn more

Privacy statement

Name: App1

Location: C:\Users\PoulEjnar\Documents\Courses\I4GUI\Lektioner\17 UWP Apps\

Solution name: UWP_demos

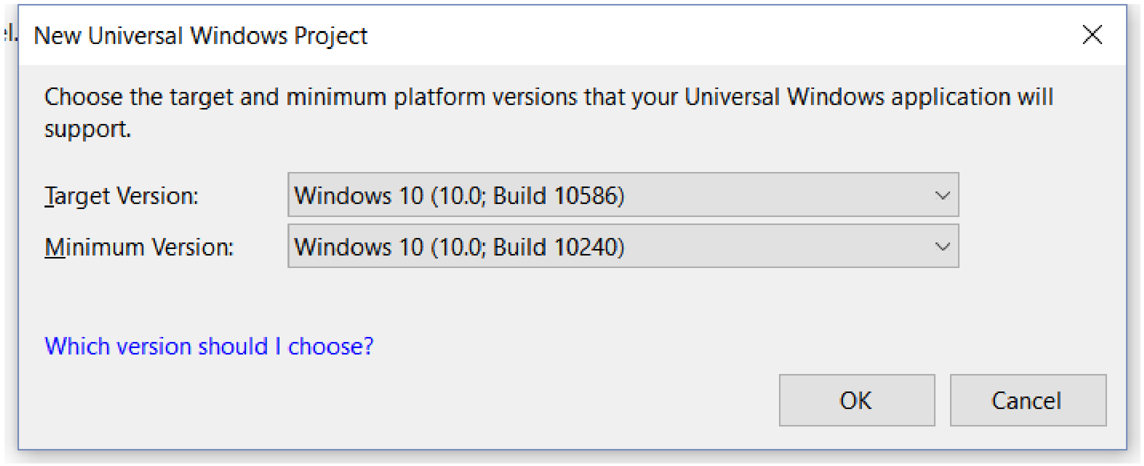
Browse...

☑ Create directory for solution

☐ Add to source control

OK | Cancel

Versions Supported


New Universal Windows Project

Choose the target and minimum platform versions that your Universal Windows application will support.

Target Version: Windows 10 (10.0; Build 10586)

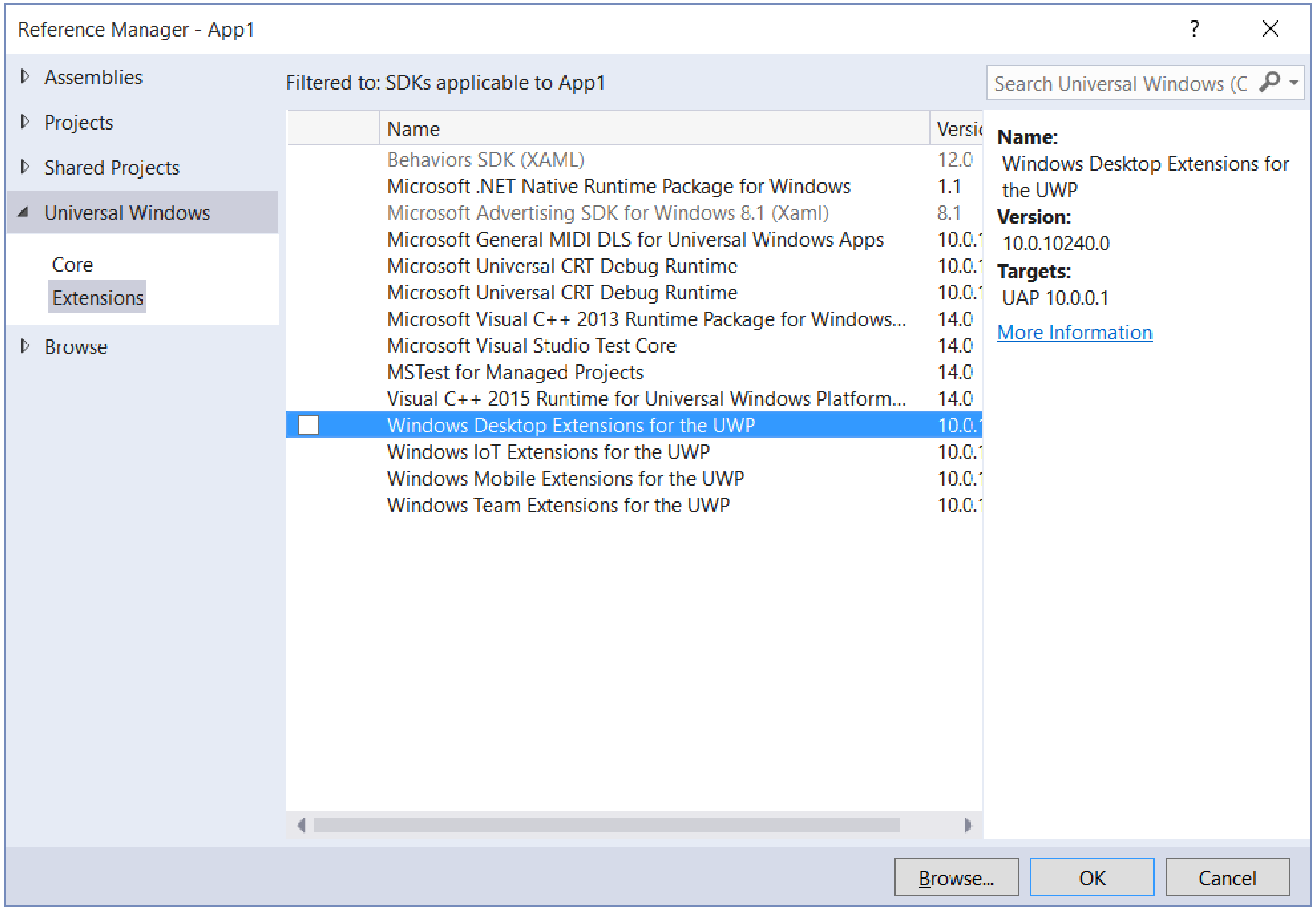
Minimum Version: Windows 10 (10.0; Build 10240)

Which version should I choose?

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Extensions

Visual Studio's IntelliSense will not recognize APIs unless they are implemented by your app's target device family or any extension SDKs that you have referenced



Filtered to: SDKs applicable to App1

Name	Versi
Behaviors SDK (XAML)	12.0
Microsoft .NET Native Runtime Package for Windows	1.1
Microsoft Advertising SDK for Windows 8.1 (Xaml)	8.1
Microsoft General MIDI DLS for Universal Windows Apps	10.0.
Microsoft Universal CRT Debug Runtime	10.0.
Microsoft Universal CRT Debug Runtime	10.0.
Microsoft Visual C++ 2013 Runtime Package for Windows...	14.0
Microsoft Visual Studio Test Core	14.0
MSTest for Managed Projects	14.0
Visual C++ 2015 Runtime for Universal Windows Platform...	14.0
Windows Desktop Extensions for the UWP	10.0.
Windows IoT Extensions for the UWP	10.0.
Windows Mobile Extensions for the UWP	10.0.
Windows Team Extensions for the UWP	10.0.

Name: Windows Desktop Extensions for the UWP
Version: 10.0.10240.0
Targets: UAP 10.0.0.1

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Using Extensions

- Use `IsTypePresent`
  - If you want to call just a small number of APIs

```
bool isHardwareButtonsAPIPresent =
     Windows.Foundation.Metadata.ApiInformation.IsTypePresent(
                     "Windows.Phone.UI.Input.HardwareButtons");
if (isHardwareButtonsAPIPresent) {
    Windows.Phone.UI.Input.HardwareButtons.CameraPressed +=
                                 HardwareButtons_CameraPressed;
}
```

- For individual members use:

  `IsEventPresent, IsMethodPresent or IsPropertyPresent`

- For API contracts use:

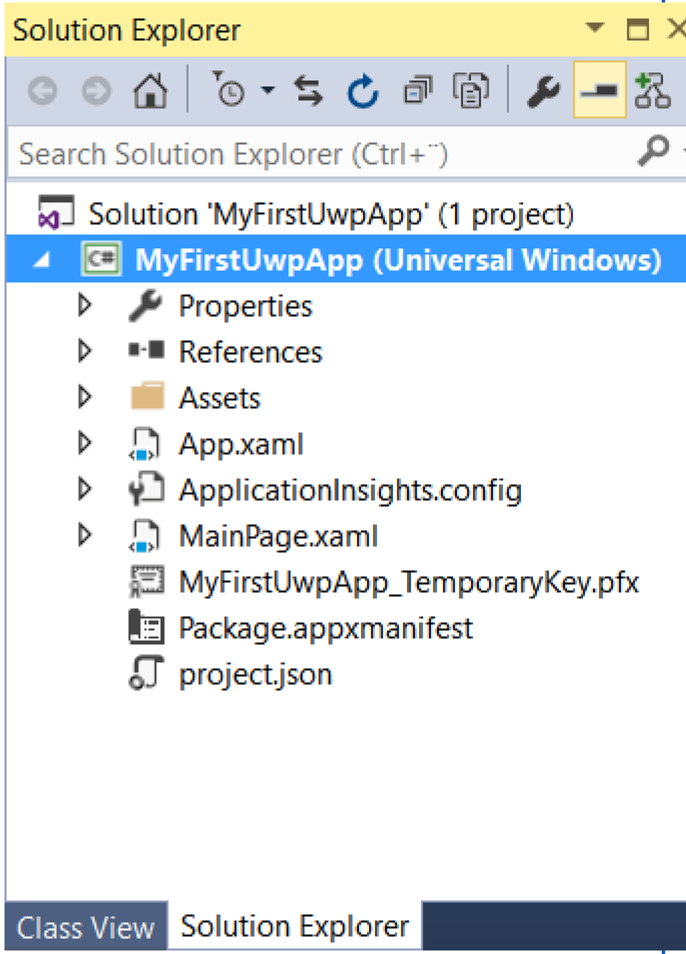
  `ApiInformation.IsApiContractPresent`

Project Structure

- **App.xaml.cs**
  Takes care of initializing everything needed by the application and has all the entry points to manage the application's lifecycle

- **The Assets folder**
  Contains all the visual assets (images, logos, icons, etc.)

- **Package.appxmanifest**
  Define all the main features of the application, like the standard visual assets (logos, tiles), the metadata (name, description), the capabilities, the integration with operating system, etc.


Solution Explorer
Search Solution Explorer (Ctrl+¨)
Solution 'MyFirstUwpApp' (1 project)
MyFirstUwpApp (Universal Windows)
Properties
References
Assets
App.xaml
ApplicationInsights.config
MainPage.xaml
MyFirstUwpApp_TemporaryKey.pfx
Package.appxmanifest
project.json
Class View
Solution Explorer

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

UI

- A UWP app can run on many different kinds of devices that have different forms of input, screen resolutions, DPI density, and other unique characteristics
- Windows 10 provides new universal controls, layout panels, and tooling to help you adapt your UI to the devices your app may run on
- Controls such as buttons and sliders automatically adapt across device families and input modes
- But your app's user-experience design may need to adapt to the device the app is running on

Frame and Pages

- Windows Store apps are based on pages show in a Frame
- The Frame is the container of all the app pages
- Every page displays some content and the user can navigate from one page to another to explore the app

```
// Extracted from App.xaml.cs and simplified
protected override void OnLaunched(LaunchActivatedEventArgs e)
{
   Frame rootFrame = new Frame();
   Window.Current.Content = rootFrame;
   rootFrame.Navigate(typeof(MainPage), e.Arguments);
}
```

```
<Page x:Class="MyFirstUwpApp.MainPage"
   <Grid>
        <TextBlock Text="Hello World" />
   </Grid>
</Page>
```

The Page's Life Cycle

- **OnNavigatedTo()**
  - Is triggered when the user navigates to the page
  - Use it to initialize the data that needs to be displayed in the page
- **OnNavigatedFrom()**
  - Is triggered when the user navigates away from the current page to another
  - Use it to save the data

Navigating across Pages

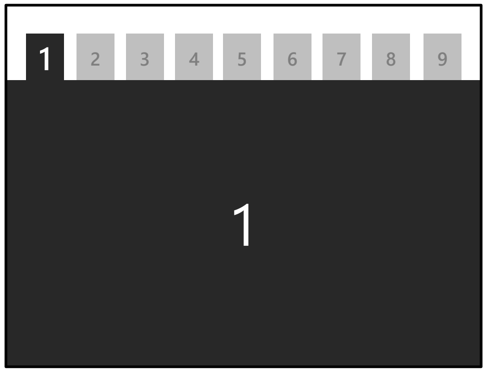
- The **Frame** class offers the basic methods with which to perform navigation from one page to another
- The basic one is called **Navigate()** and it accepts, as parameter, the type that identifies the page to where you want to redirect the user

```
void OnGoToMainPageClicked(object sender, RoutedEventArgs e){
    Person person = People.SelectedItem as Person;
    this.Frame.Navigate(typeof(MainPage), person);
}
```

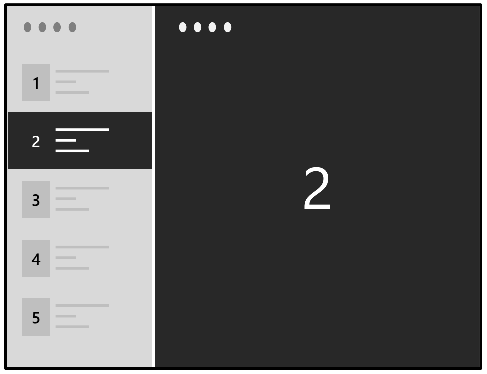
```
protected override async void OnNavigatedTo (NavigationEventArgs e)
{
    Person person = e.Parameter as Person;
    MessageDialog dialog = new MessageDialog(person.Name);
    await dialog.ShowAsync();
}
```

Navigation patterns

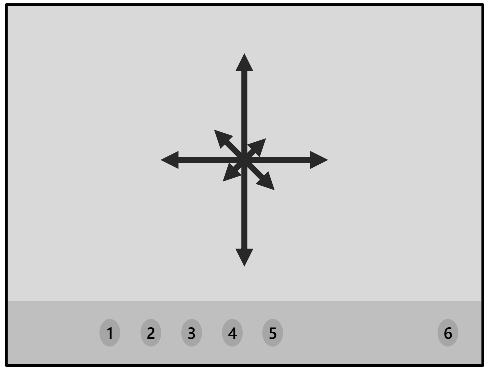
TABS (Pivot)



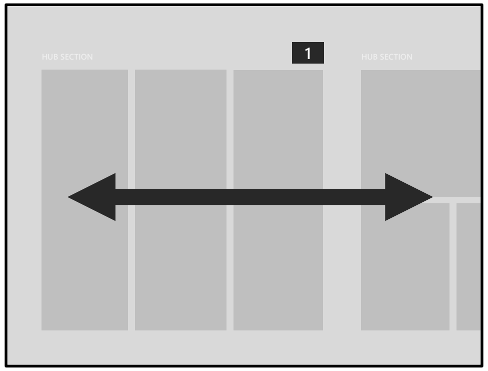
MASTER DETAIL



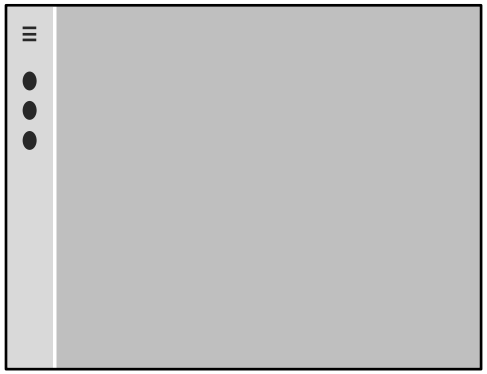
ACTIVE CANVAS



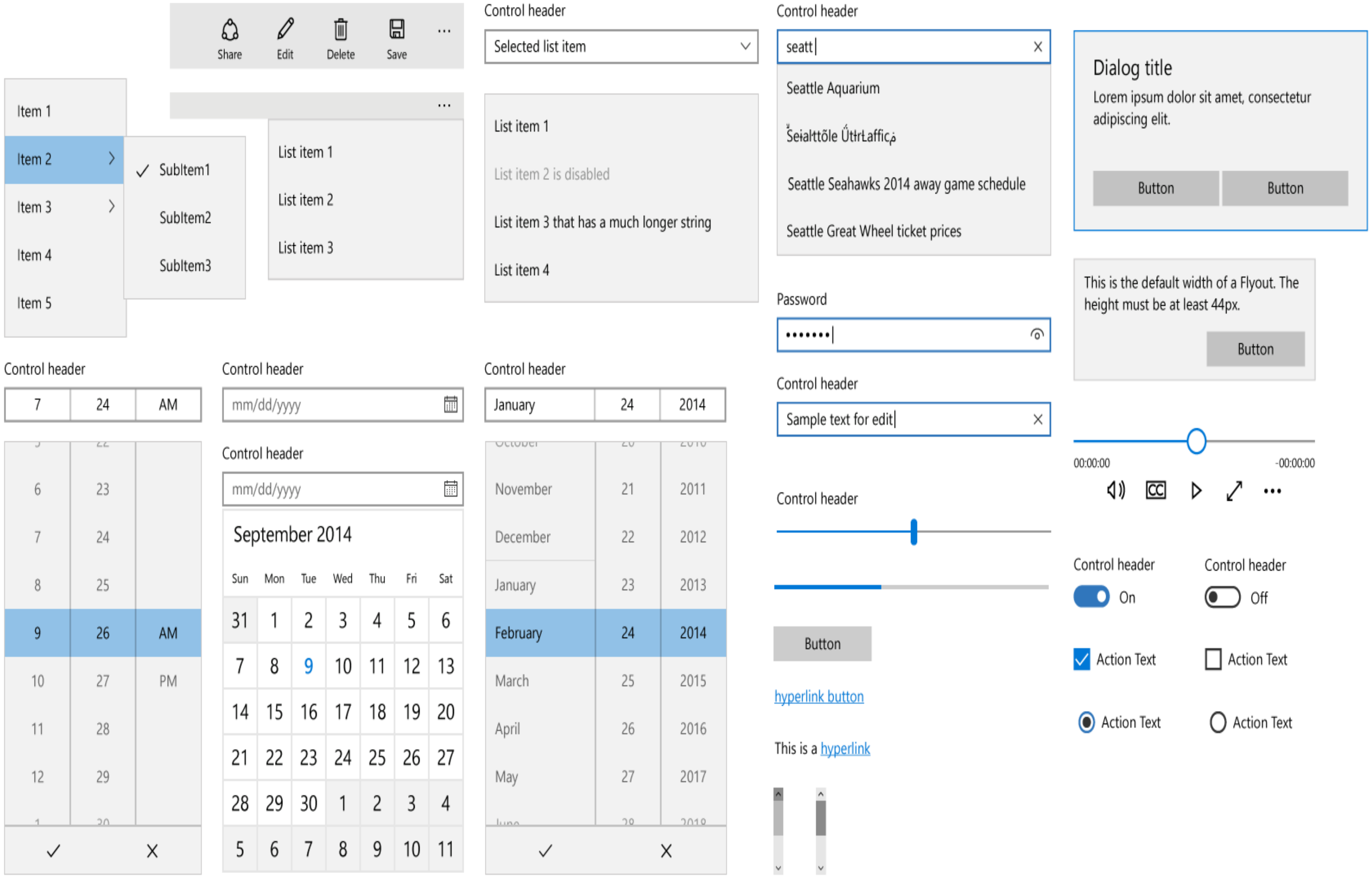
HUB



NAV PANE



Use the universal control library



Lorem ipsum dolor sit amet, consectetur adipiscing elit.

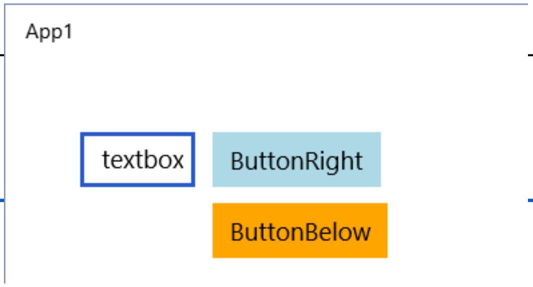
This is the default width of a Flyout. The height must be at least 44px.

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Adaptive Panels

- The RelativePanel implements a style of layout that is defined by the relationships between its child elements
- It's intended for use in creating app layouts that can adapt to changes in screen resolution
  - The RelativePanel eases the process of rearranging elements by defining relationships between elements, which allows you to build more dynamic UI without using nested layouts

```
<RelativePanel>
    <TextBox x:Name="textBox1" Text="textbox" Margin="5"/>
    <Button x:Name="blueButton" Margin="5" Background="LightBlue"
          Content="ButtonRight" RelativePanel.RightOf="textBox1"/>
    <Button x:Name="orangeButton" Margin="5" Background="Orange"
          Content="ButtonBelow" RelativePanel.RightOf="textBox1"
          RelativePanel.Below="blueButton"/>
</RelativePanel>
```



Resources Based On The Theme

- A new keyword, called **ThemeResource**, which can be used to automatically adapt a resource according to the device’s theme

```
<Application.Resources>
  <ResourceDictionary>
    <ResourceDictionary.ThemeDictionaries>
      <ResourceDictionary x:Key="Dark">
        <SolidColorBrush Color="Red" x:Key="ApplicationTitle" />
      </ResourceDictionary>
      <ResourceDictionary x:Key="Light">
        <SolidColorBrush Color="Blue" x:Key="ApplicationTitle" />
      </ResourceDictionary>
    </ResourceDictionary.ThemeDictionaries>
  </ResourceDictionary>
</Application.Resources>
```

```
<TextBlock Text="Hello"
           Foreground="{ThemeResource ApplicationTitle}"/>
```

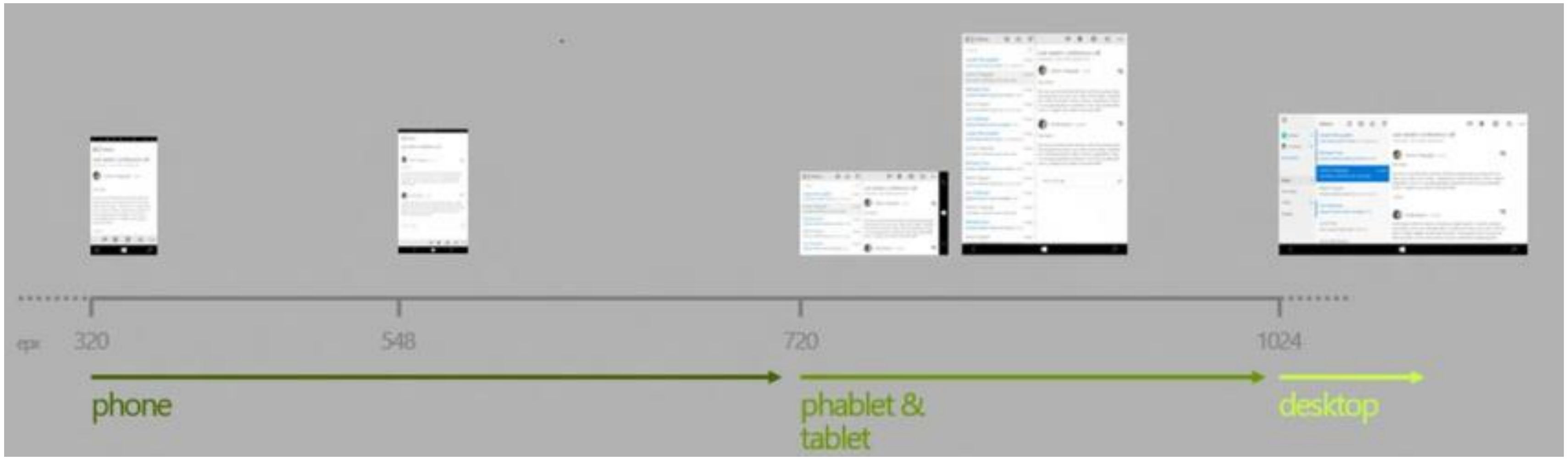
Visual State Triggers

- Use visual state triggers to build UI that can adapt to available screen space
- StateTriggers define a threshold at which a visual state is activated, which then sets layout properties as appropriate for the window size that triggered the state change
- When the window is less than 720 pixels, the **narrowView** visual state is triggered because the **wideView** trigger is no longer satisfied and so no longer in effect

```
<Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
  <VisualStateManager.VisualStateGroups>
    <VisualStateGroup>
      <VisualState x:Name="wideView">
        <VisualState.StateTriggers>
          <AdaptiveTrigger MinWindowWidth="720" />
        </VisualState.StateTriggers>
        <VisualState.Setters>
          <Setter Target="best.(RelativePanel.RightOf)" Value="free"/>
          <Setter Target="best.(RelativePanel.AlignTopWidth)" Value="free"/>
        …
```

Targeting Different Devices

Adaptive trigger	What is targeted
`<AdaptiveTrigger MinWindowWidth="320"/>`	Phone
`<AdaptiveTrigger MinWindowWidth="720"/>`	Tablet
`<AdaptiveTrigger MinWindowWidth="1024"/>`	Desktop
`<AdaptiveTrigger MinWindowWidth="1280"/>`	XBox
`<AdaptiveTrigger MinWindowWidth="1920"/>`	Surface Hub

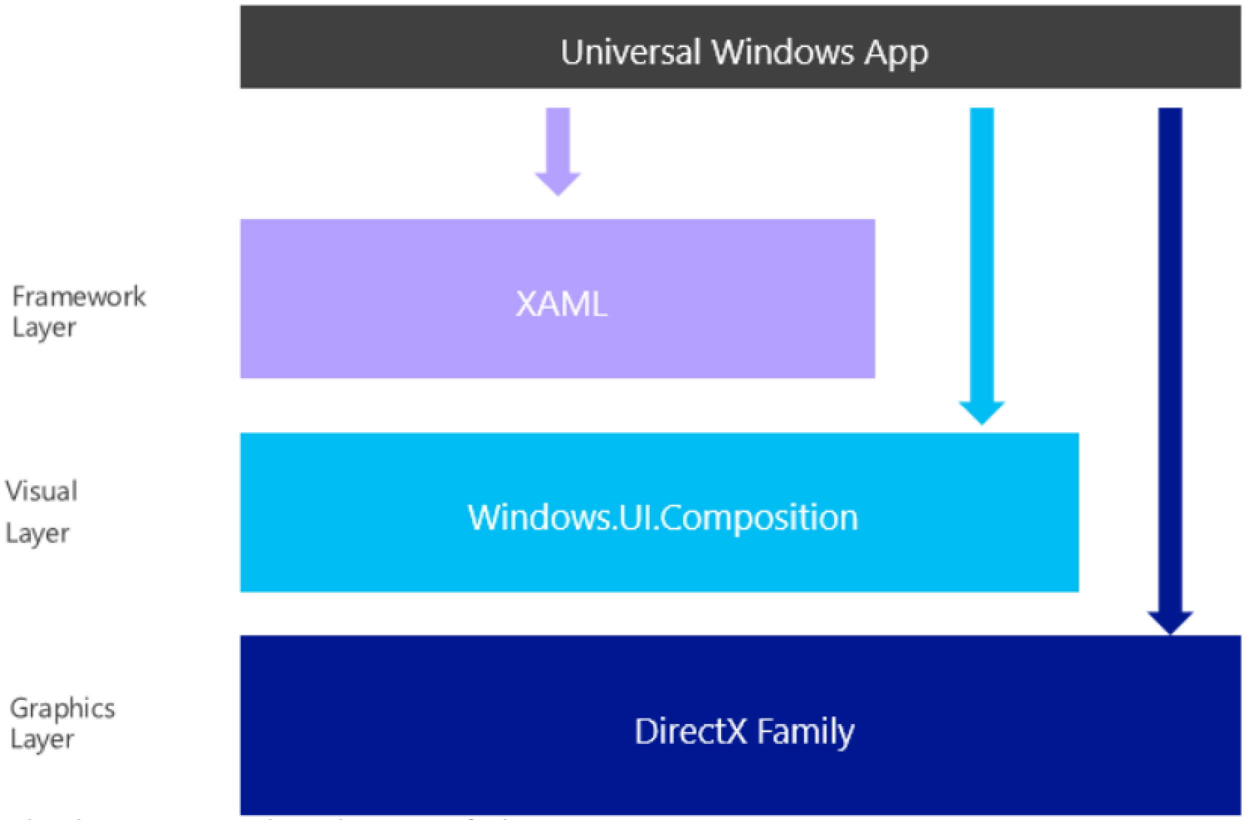


Universal Input

- Common input handling allows you to receive input through touch, a pen, a mouse, or a keyboard, or a controller such as a Microsoft Xbox controller
- **`CoreIndependentInputSource`** is a new API that allows you to consume raw input on the main thread or a background thread
- **`PointerPoint`** unifies raw touch, mouse, and pen data into a single, consistent set of interfaces and events that can be consumed on the main thread or background thread by using CoreInput
- **`PointerDevice`** is a device API that supports querying device capabilities so that you can determine what input modalities are available on the device
- The new **`InkCanvas`** XAML control and **`InkPresenter`** Windows Runtime APIs allow you to access ink stroke data

What is Windows.UI.Composition?

The Windows.UI.Compositon WinRT APIs can be called from any Universal Windows Platform app to create lightweight visuals and apply powerful animations, effects, and manipulations on those objects in your application



How does this help you with a beautiful engaging UI? By giving you access to this new layer of APIs, Windows.UI.Composition lets you create composition objects directly in the compositor layer, which improves performance and scale using exciting new features

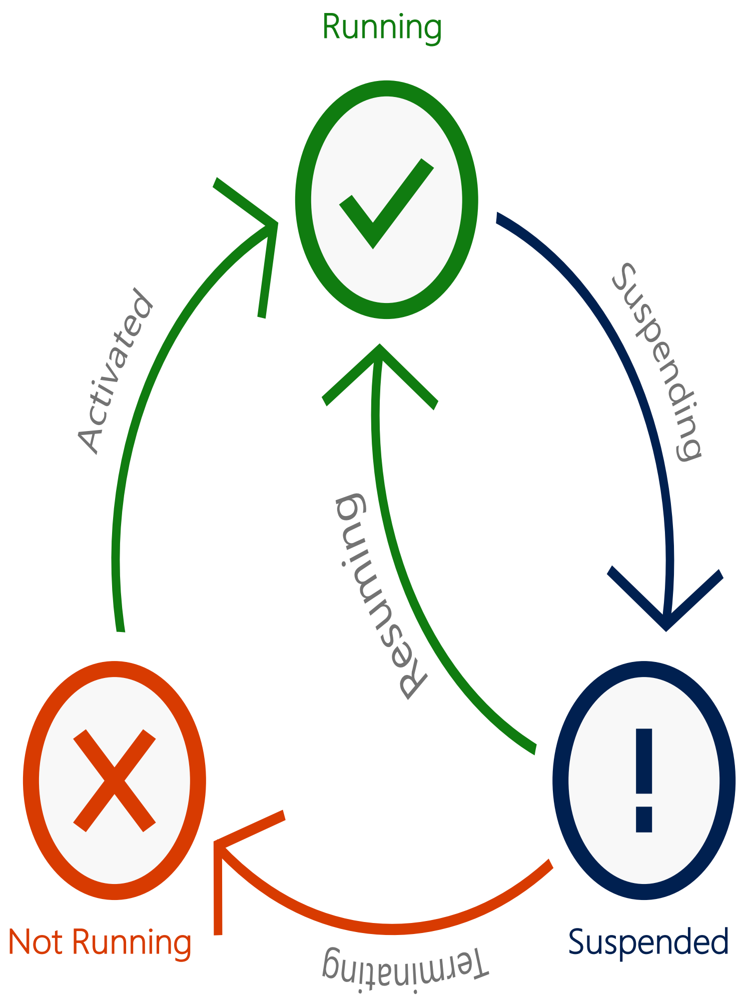
AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

The New Effects System

- Real-time effects that can be animated, customized and chained. Effects include 2D affine transforms, arithmetic composites, blends, color source, composite, contrast, exposure, grayscale, gamma transfer, hue rotate, invert, saturate, sepia, temperature and tint

- Se: https://msdn.microsoft.com/en-us/library/windows/apps/mt592878.aspx

Application Lifecycle



Apps can be in 1 of 3 states

- Not Running
- Running
- Suspended

Applications receive callbacks or events on transitions

- OnLaunched or OnActivated
- Suspending
- Resuming
- Except: Suspended->NotRunning

PACKAGING FOR WINDOWS STORE APPS

Store vs. L-O-B

- Windows apps can be very broadly classified into consumer applications and enterprise line-of-business applications
  - But there are lots of gray areas in between
- Consumer apps can only be distribute through the Windows Store
  - It provides developers with a channel through which their Windows Store apps can be discovered by users
- Enterprise L-O-B apps can be distribute through side-loading or an enterprise app store
- The Windows 10 Store accepts only UWP apps
- Because each UWP app declares its prerequisites, the Windows Store can inform users about system and software requirements so that they do not purchase apps that they won't be able to run

App Package

- A UWP app is deployed in an app package
  - which is a compressed file that contains the files that make up the app
  - along with a manifest that describes the app to Windows
- An app package has an identity that includes:
  - name
  - publisher
  - version
  - other information
- The app package is the unit of deployment and servicing
- You write the app, and Visual Studio and Windows take care of packaging and deploying it
  - All files that your app needs ship with your app

App Manifest

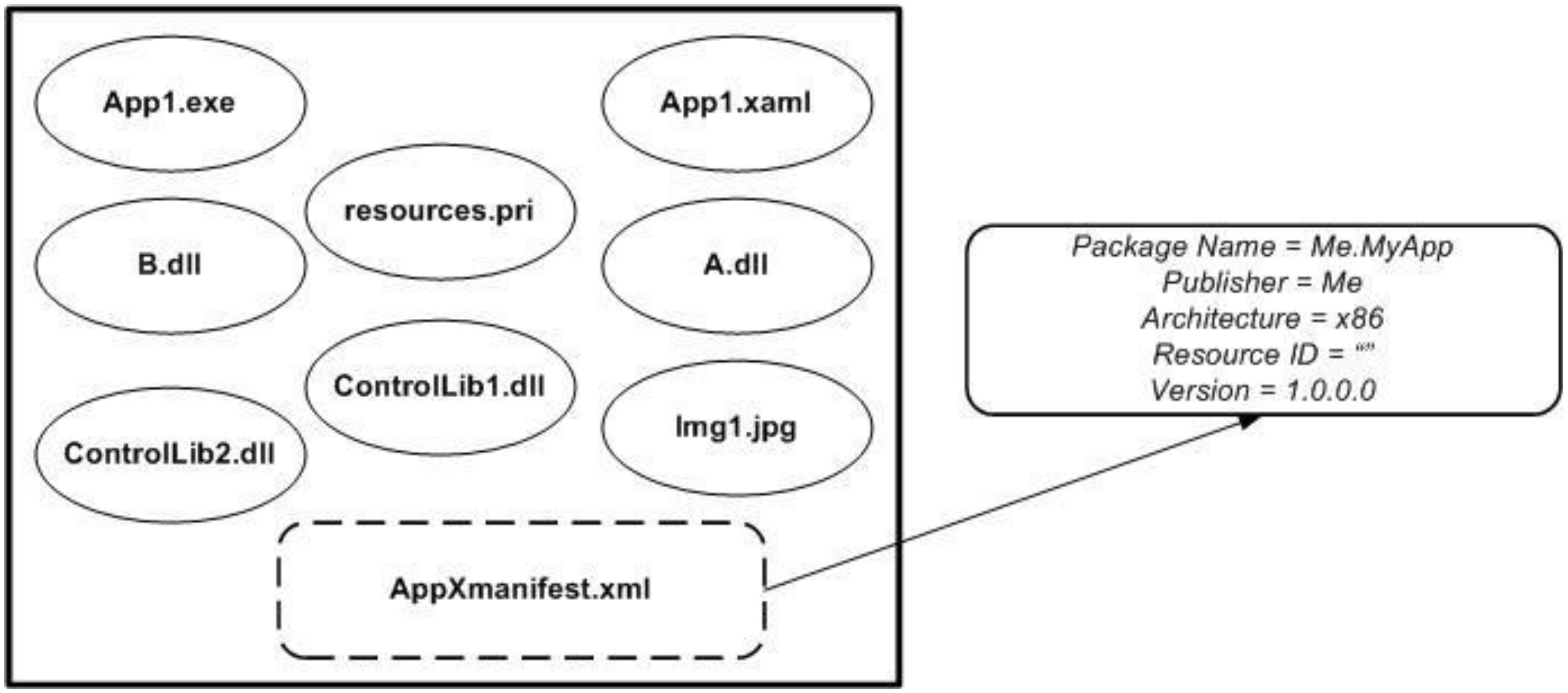
- The manifest explicitly declares the package's:
  - TargetDeviceFamily
  - Dependencies on other app packages

- The manifest enables Windows to understand the software's footprint on the system and to cleanly remove the software on behalf of the developer

- Frictionless deployment and the ability to uninstall software easily brings Windows client apps closer to the cleaner, low-impact world of web apps

```
<Dependencies>
   <TargetDeviceFamily Name="Windows.Universal" MinVersion="10.0.x.0"
MaxVersionTested="10.0.y.0"/>
</Dependencies>
```

App Package Example



References & Links

- Guide to Universal Windows Platform (UWP) apps
  https://msdn.microsoft.com/en-us/library/windows/apps/dn894631.aspx

- Guidelines for Universal Windows Platform (UWP) apps
  https://msdn.microsoft.com/en-US/library/windows/apps/hh465424
- How-to guides for Windows 10 apps
  https://msdn.microsoft.com/library/windows/apps/xaml/mt244352.aspx
- Design UWP apps
  https://developer.microsoft.com/en-us/windows/apps/design
- Windows 10 Universal Windows Platform – Tutorials
  https://comentsys.wordpress.com/2015/05/31/windows-10-universal-windows-platform-tutorials/
- Entity Framework - Getting Started on Universal Windows Platform
  http://ef.readthedocs.org/en/latest/getting-started/uwp.html

References & Links

- **Windows apps concept mapping for Android and iOS developers**
  https://msdn.microsoft.com/en-us/windows/uwp/porting/android-ios-uwp-map

- Build 2016
  https://channel9.msdn.com/Events/Build/2016/

- **Windows 10 Courses For Developers from MVA**
  http://bit.do/developers-guide-to-windows-10
  Template-10 ★★★
  https://mva.microsoft.com/en-us/training-courses/getting-started-with-template-10-16336?l=R3bBb1LRC_4305918554

- OR for beginners
  https://borntolearn.mslearn.net/b/mva/archive/2016/01/08/9-new-windows-10-courses-for-developers-8-short-ones-and-1-long-one