Windows 8 – Let me Introduce

Razorfish Internal Presentation

October 25, 2012

Microsoft

razorfish

Agenda

Introduction

Quick Tour of Windows8

UX considerations

Presentation Layer Technologies

Back End Technologies

App Demo

Q&A

Introduction

Windows8 started with few releases starting last year – consumer preview, release preview, RTM release and the final Windows8 the New OS is getting launched tomorrow Oct 26 2012.

Razorfish Seattle office been engaged to develop apps for the MSFT Retail business group from October 2010 (last year).

We formed a windows8 team to work on these initiatives from Seattle and Portland offices for design, UX, Creative, Architecture, Development and QA.

During this design and build process of the app, we been working closely with windows engineering Team, UX and Store team for approval process which is a critical piece to get your app ready for windows store.

Having this opportunity of working closely with various internal MSFT teams, helped the RF teams to understand the best practices, design considerations, process involved and also access to MSFT windows 8 group Internal documents and resources.

The App itself is a challenging piece which includes a mixed technology, supporting more than 34+ localization and flexible enough to support any variation of content and design (to some extent) for more than 100+ retailers, 5000+ stores world wide, running 100,000+ PC to start with and all of this are part of customization without touching the code base.

The entire app is build on total flexible architecture to adapt changing business content and design requirement. The app is totally re-usable with some level of customization for any other businesses/clients to provide a windows store app experiences to their customer base.

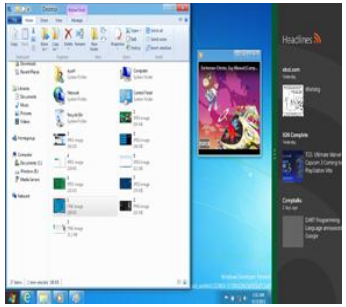
Introduction

Single touch Navigate

Picture password

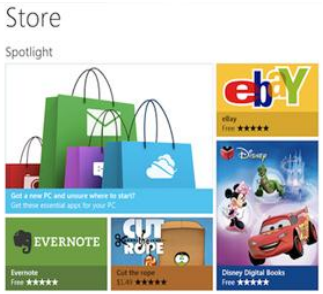
Semantic Zoom

Smooth transition between Interfaces



Charms

Windows Store



Building Windows 8 Apps

UX Considerations

Chad Hessoun

Microsoft

razorfish

Modern or Desktop?

Two types of apps, both are equally important

Modern apps are good at…

- touch first, unified, simple experiences
- consuming content
- information and entertainment apps

Desktop apps are good at…

- precision, control, complexity
- creating content
- line of business, transactional and deep-featured apps

Core Principles for Modern (metro)

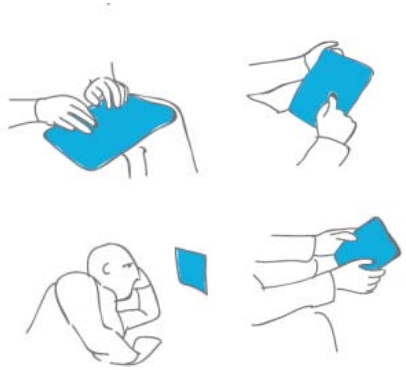
Do more with less	Pride in craftsmanship	Be fast and fluid	Authentically Digital	Win as one
Content before chrome Visual clarity	Typography Grid	Continuity through motion Ergonomics and touch Simplified flow	Beyond physical metaphor Infographics and glyphs	Consistent UI model Contracts: 1+1=3
Avoid distractions and focus on what the user is doing right now	The details matter when there's no chrome, so get them right	The interaction should be as fresh and clean as the visual presentation	Leave the physical world behind and take advantage of the available flexibility	Working together lightens the load and make the sum greater than the whole of the parts

Touch First

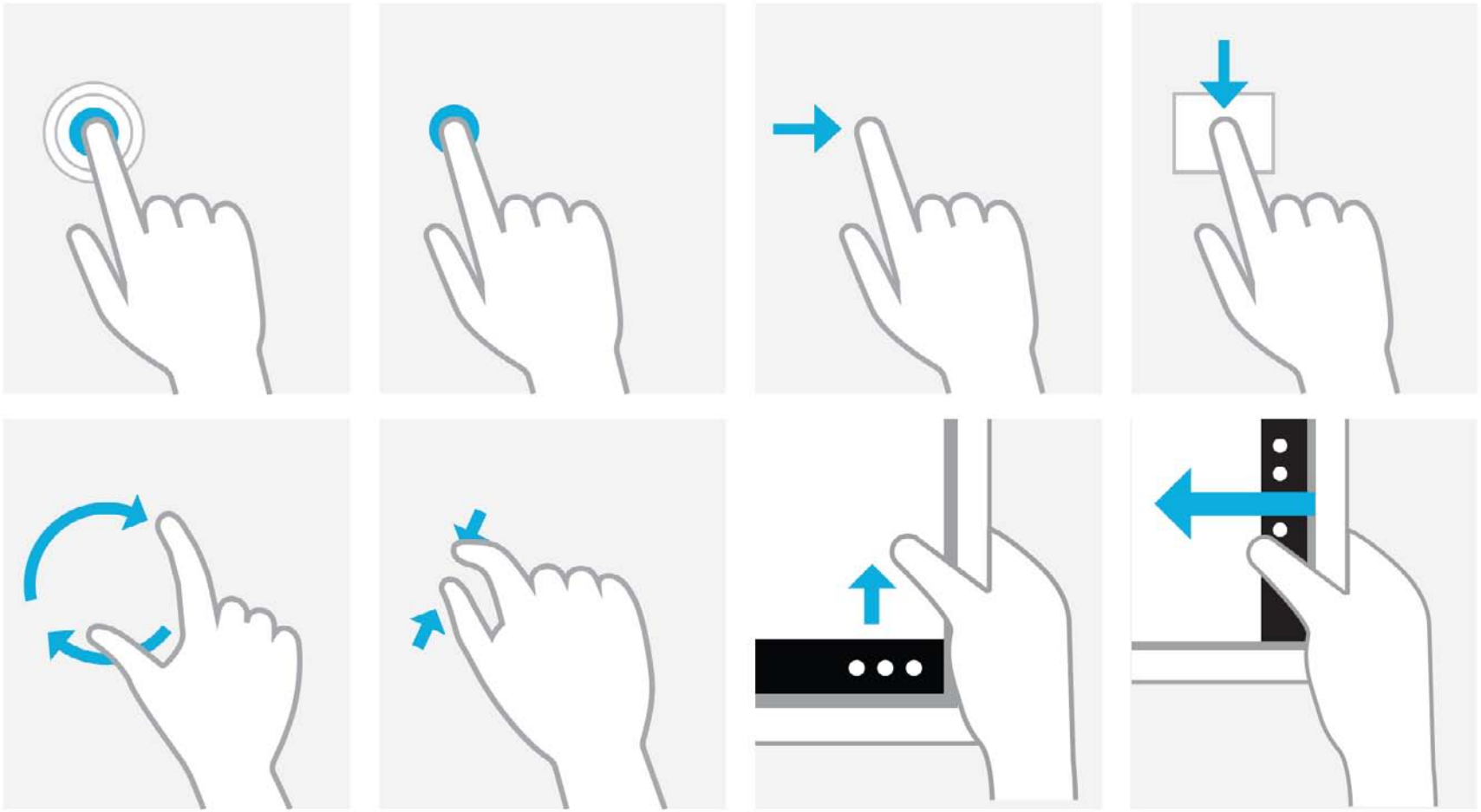
Touch input viscerally connects the user to the interaction and content

Think beyond tap and take advantage of all the available gestures (ex. sliding and flicking)

Remember: users are probably holding the device



Gesture Library



Content Before Chrome

Views should be about where you are, not where you might go

Be immersive

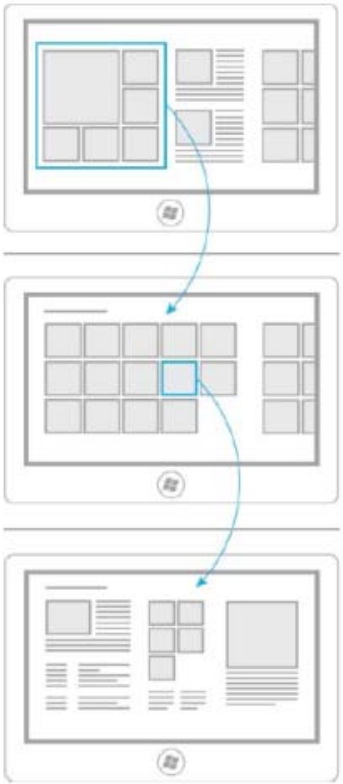
- flow content edge to edge
- remove lines and boxes
- create structure with spacing to give content breathing room

Avoid persistent wayfinding chrome (ex. tabs)

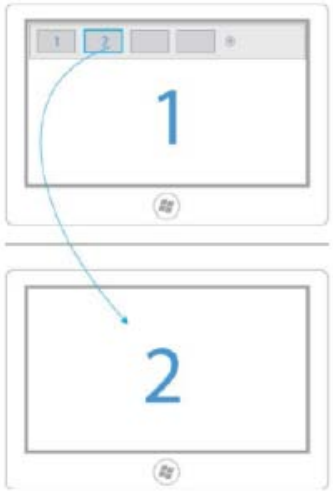
Integrate navigation into the content area

Two Possible Structures

Hierarchical



Flat



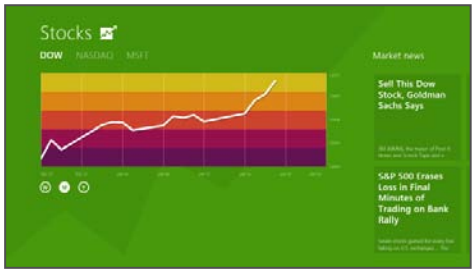
Design For Every Screen

Large or Small Screens

High Pixel Density

Design for Every View

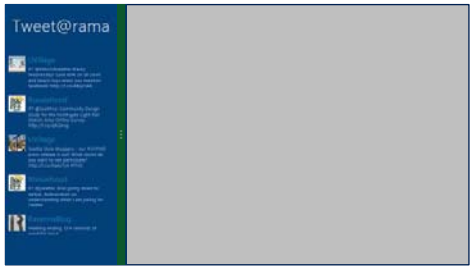
Full, Portrait, Fill, Snap



Full

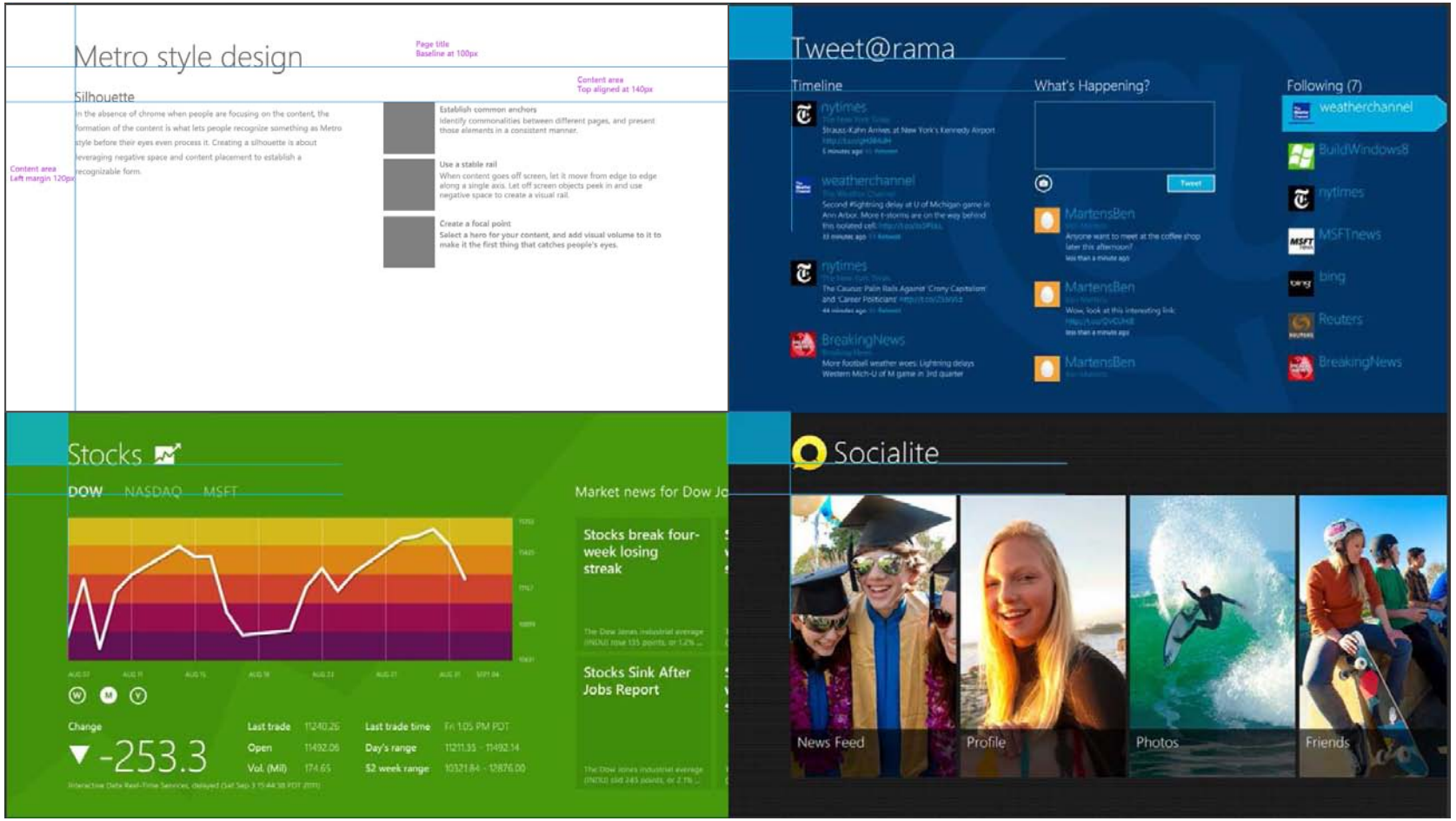
Full (portrait)

Fill



Snap

A Unified Grid



Win as One

Our app is the best at _____

- focus on core content or features
- work together to enable users

Leverage the capabilities of other apps through contracts and charms

Make your app's features available to others

Consistency reinforces the Windows 8 experience

Building Windows 8 Apps

Presentation Layer Technologies

Luke Veach

Microsoft

razorfish

Building the Modern UI

Similarities between Modern Apps and web applications:

- Internet Explorer 10 rendering engine
- CSS2/3 + MSFT extensions
- JavaScript
- Html

Working with the HTML/JS controls will feel very familiar to people with experience in ASP.Net and Visual Studio. Visual styles are controlled via CSS which provides quite a bit of flexibility.

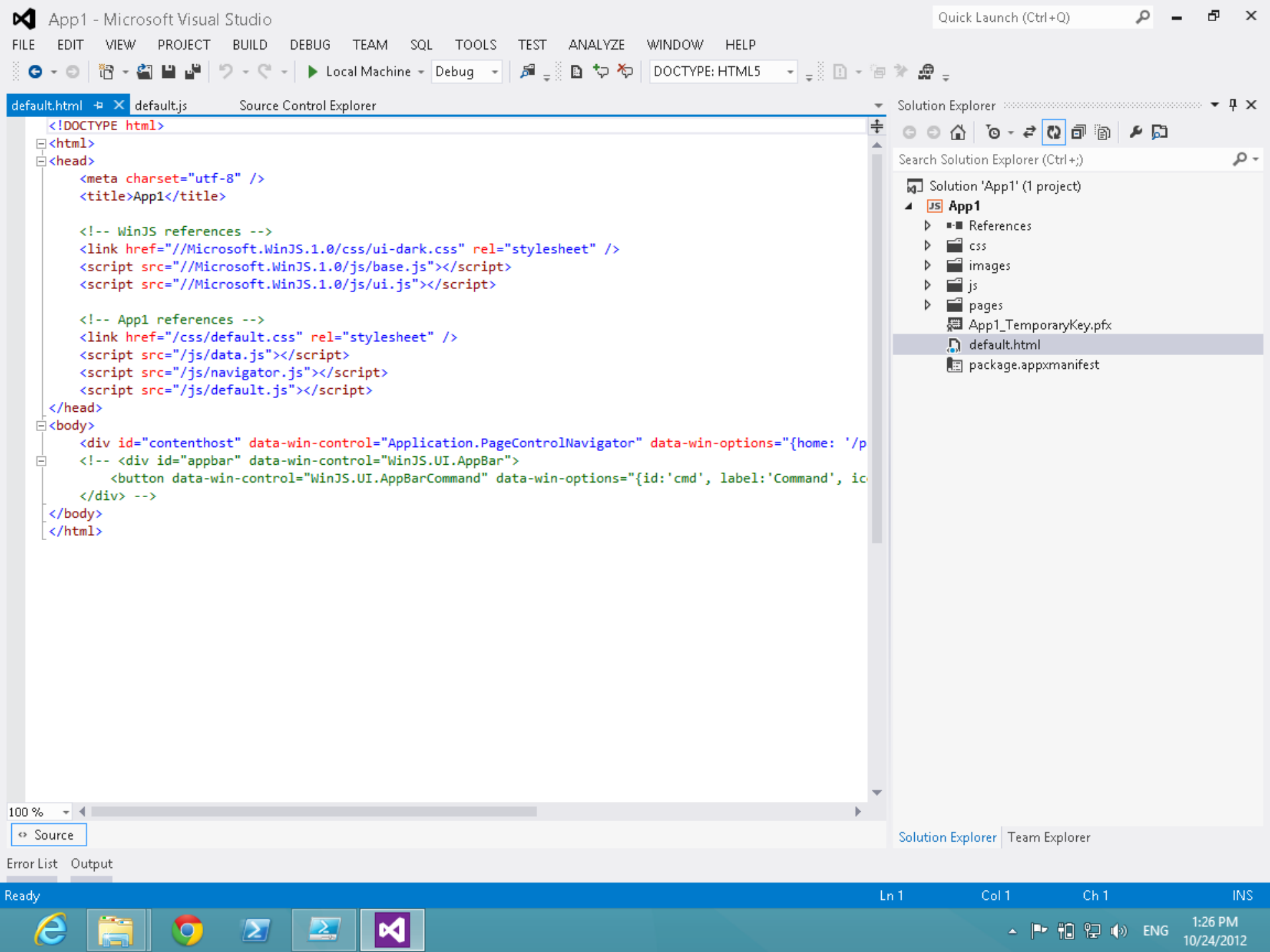
Building the Modern UI

There are a few notable differences, specifically around JavaScript processing:

- window methods such as *alert*, *prompt*, *open*, *moveBy*, *moveTo*, *resizeBy*, and *resizeTo* don't work in Windows Store apps using JavaScript.
- Security restrictions when adding dynamic html by default
  - innerHTML and outerHTML
  - insertAdjacentHTML
  - pasteHTML
  - document.write and document.writeln
  - DOMParser.parseFromString

JavaScript

```
var someElement = document.getElementById('someElementID');
MSApp.execUnsafeLocalFunction(
    function() { someElement.innerHTML = '<div onclick="console.log(\"hi\");">hi</div>' }
);
```


App1 - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL TOOLS TEST ANALYZE WINDOW HELP
Local Machine
Debug
DOCTYPE: HTML5
default.html
default.js
Source Control Explorer
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<title>App1</title>
<!-- WinJS references -->
<link href="//Microsoft.WinJS.1.0/css/ui-dark.css" rel="stylesheet" />
<script src="//Microsoft.WinJS.1.0/js/base.js"></script>
<script src="//Microsoft.WinJS.1.0/js/ui.js"></script>
<!-- App1 references -->
<link href="/css/default.css" rel="stylesheet" />
<script src="/js/data.js"></script>
<script src="/js/navigator.js"></script>
<script src="/js/default.js"></script>
</head>
<body>
<div id="contenthost" data-win-control="Application.PageControlNavigator" data-win-options="{home: '/p
<!--
<button data-win-control="WinJS.UI.AppBarCommand" data-win-options="{id:'cmd', label:'Command', ic
 -->
</body>
</html>
100 %
Source
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'App1' (1 project)
App1
References
css
images
js
pages
App1_TemporaryKey.pfx
default.html
package.appxmanifest
Solution Explorer
Team Explorer
Error List
Output
Ready
Ln 1
Col 1
Ch 1
INS
ENG
1:26 PM
10/24/2012

JavaScript and WinJS

The JS code to support application data and UI interactions is equally familiar. MSFT has provided a number of utility and helper functions along with the core API.

```
WinJS.Binding.optimizeBindingReferences = true;

var app = WinJS.Application;
var activation = Windows.ApplicationModel.Activation;
var nav = WinJS.Navigation;
```

*note: developers are welcome to use 3rd party JS libraries in addition to the core files.

Fluid layout

Level of Complexity:

Current target resolutions

Screen size

Pixel density (“retina display”)

Orientation

This impacts:

- Creative and UX
- Dev/Technology
- QA

Adapting the layout via CSS

The main layout is generally defined in CSS. Leveraging media queries we are able to define style rules for each view-port mode and orientation:

```
@media screen and (-ms-view-state: fullscreen-portrait),
       screen and (-ms-view-state: fullscreen-landscape),
       screen and (-ms-view-state: filled)...
@media screen and (-ms-view-state: snapped)...
@media screen and (-ms-view-state: fullscreen-portrait) { .groupeditemspage .win-surface { margin-le

/* Global High Contrast Changes */
@media (-ms-high-contrast)...
@media (-ms-high-contrast) and (-ms-view-state: fullscreen-landscape),
       (-ms-high-contrast) and (-ms-view-state: fullscreen-portrait),
       (-ms-high-contrast) and (-ms-view-state: filled)...
/* Global high contrast snapped */
@media (-ms-high-contrast) and (-ms-view-state: snapped)...

/* 1920 at 96dpi*/
@media screen and (min-resolution: 96dpi) and (max-resolution: 96dpi) and (min-device-width: 1920px)

/* 2560x1440 at 96dpi*/
@media screen and (min-resolution: 96dpi) and (max-resolution: 96dpi) and (min-device-width: 2560px)
```

Adapting the layout via JS

WinJS classes give us the ability to determine display data such as:

- Device dimensions and resolution
- Available-screen dimensions
- logical-DPI

Which in turn allows us to scale elements, swap in hi-rez images, and choose what content is visible.

Additionally we can test for High-Contrast and other accessibility modes.

process / tools:

CSS/JS tools:

- Visual Studio
- SASS / LESS
- CoffeeScript

Dev Environment:

- Visual Studio (Win8 + VirtualBox + OSX)
- PowerShell

Getting into the App Store

Once the app passes internal QA and the automated store validation you can submit it for inclusion in the store.

The review cycle can take some time, and once the app is accepted, any additional updates must be resubmitted and take the form of an app update.

This has potential ramifications for maintenance and content-refresh.

Lessons Learned

Don't:

- Use sizes that aren't multiples of 5px
- Try to embed HTML5 video in a listview
- Don't re-create Windows 8 gestures or animations and use them differently.

Building Windows 8 Apps

Back End Technologies

Paul McKee

Microsoft

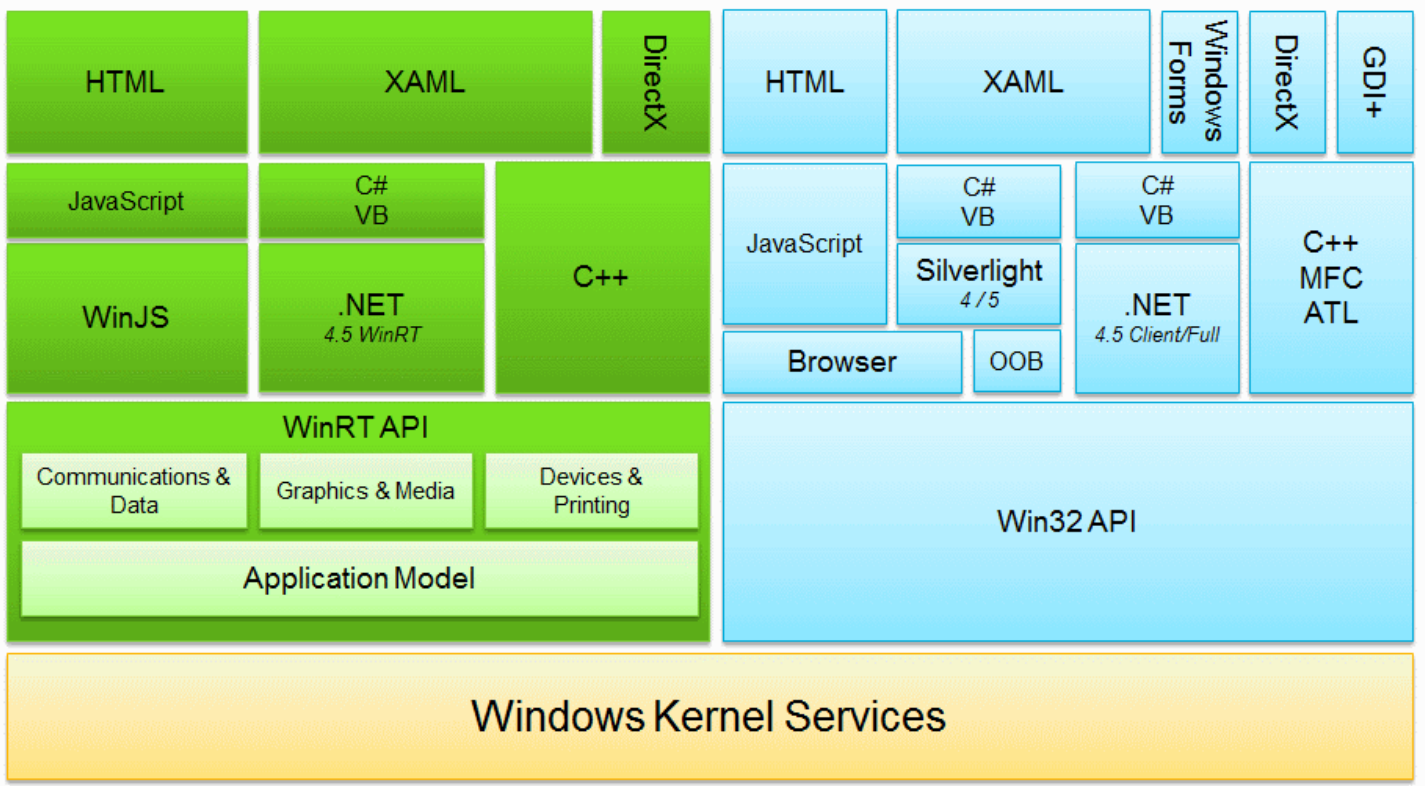
razorfish

Windows 8 Technology Stacks

It's your choice

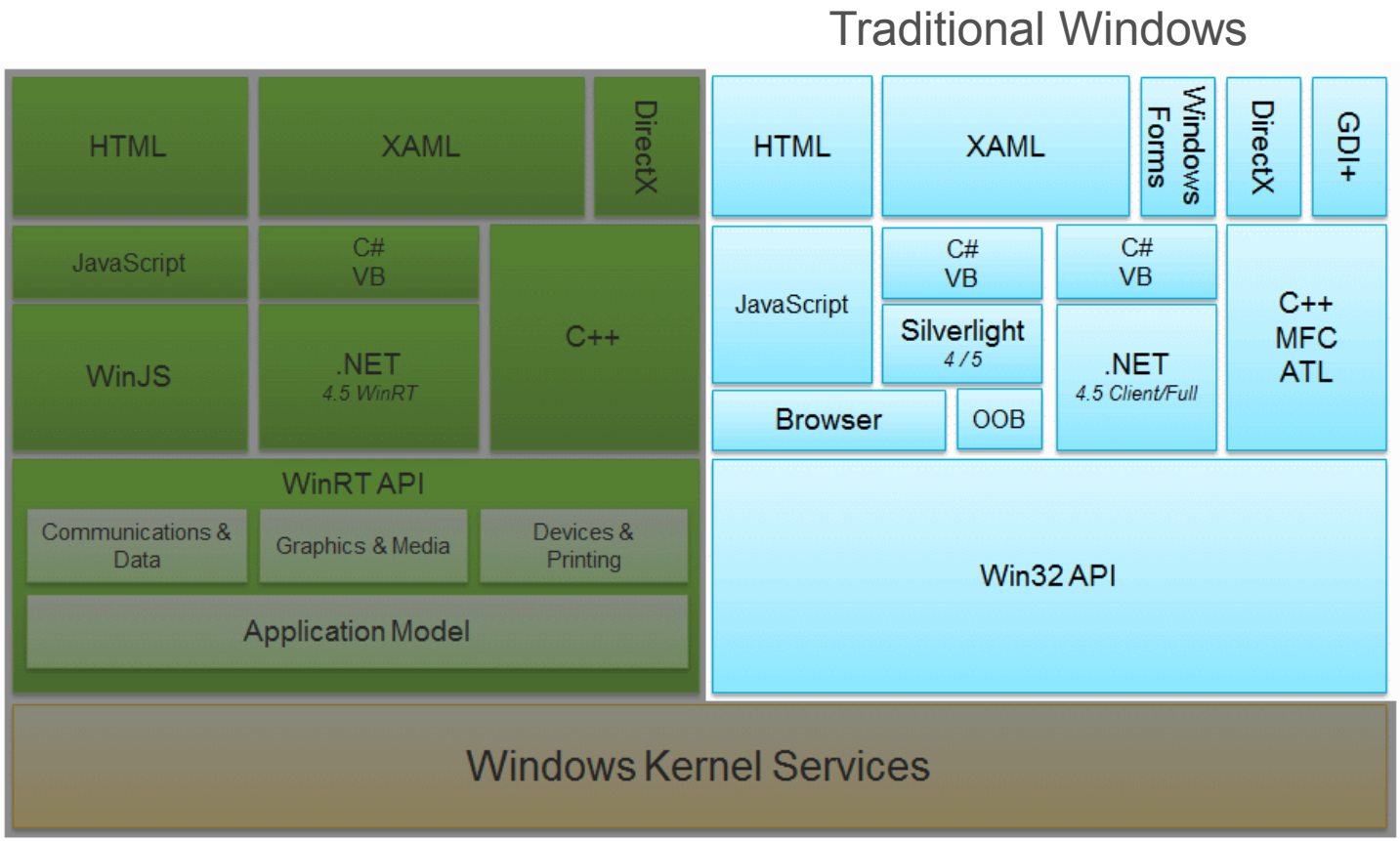
- XAML or DirectX with C++
- XAML with C# (or VB)
- HTML/CSS with Javascript

Windows 8 Technology Stacks

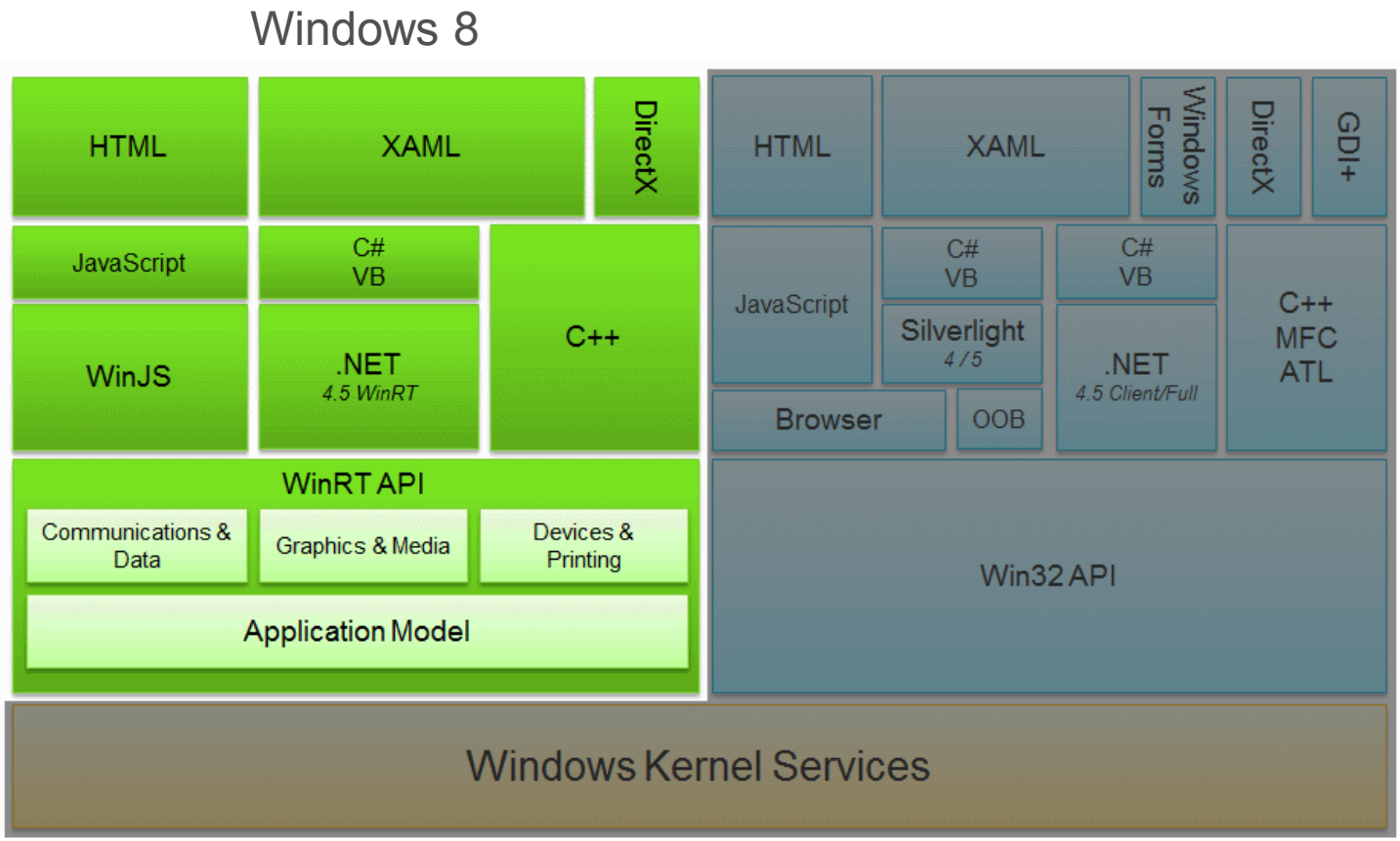


Source: http://www.lhotka.net/weblog/Windows8DevelopmentPlatformClarified.aspx

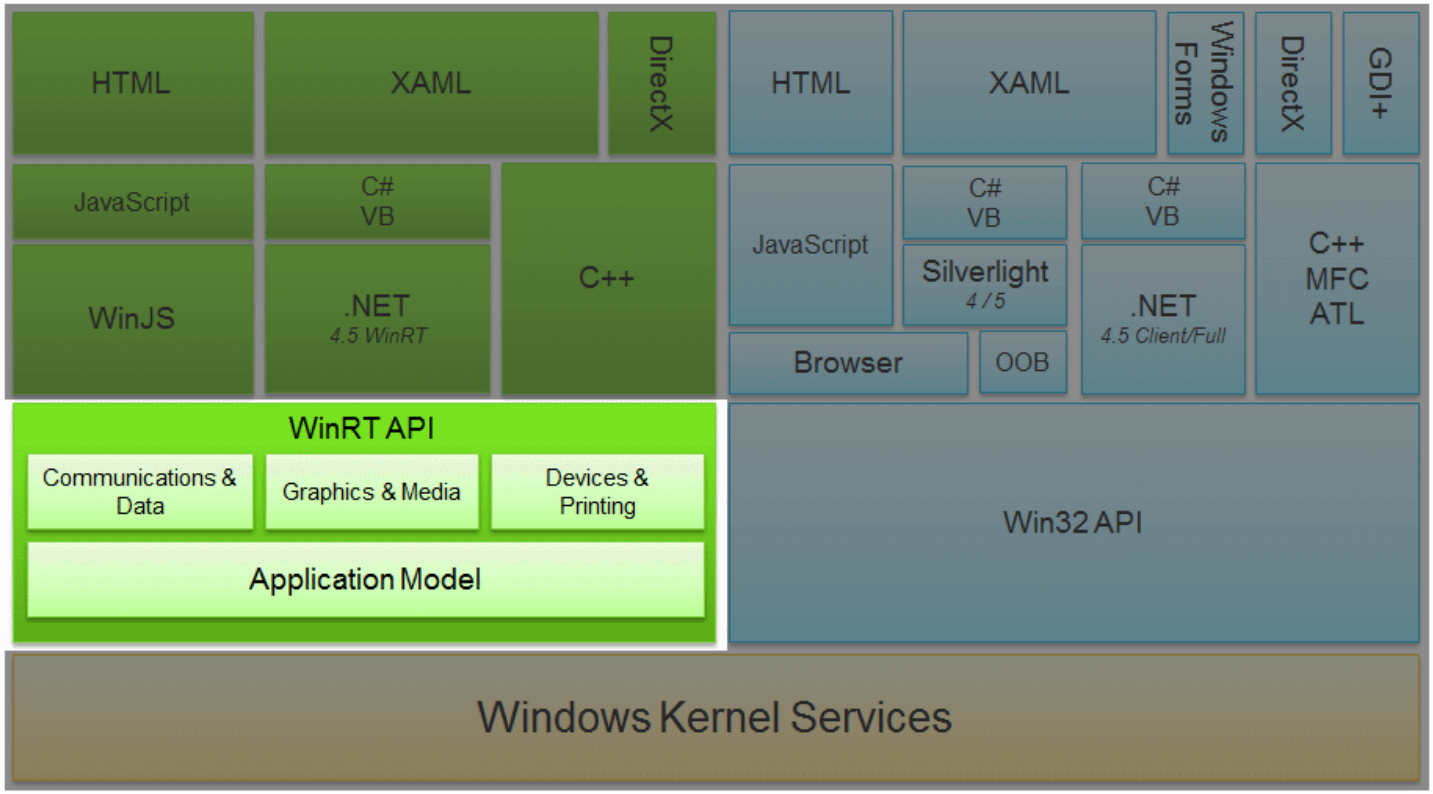
Windows 8 Technology Stacks



Windows 8 Technology Stacks



WinRT API

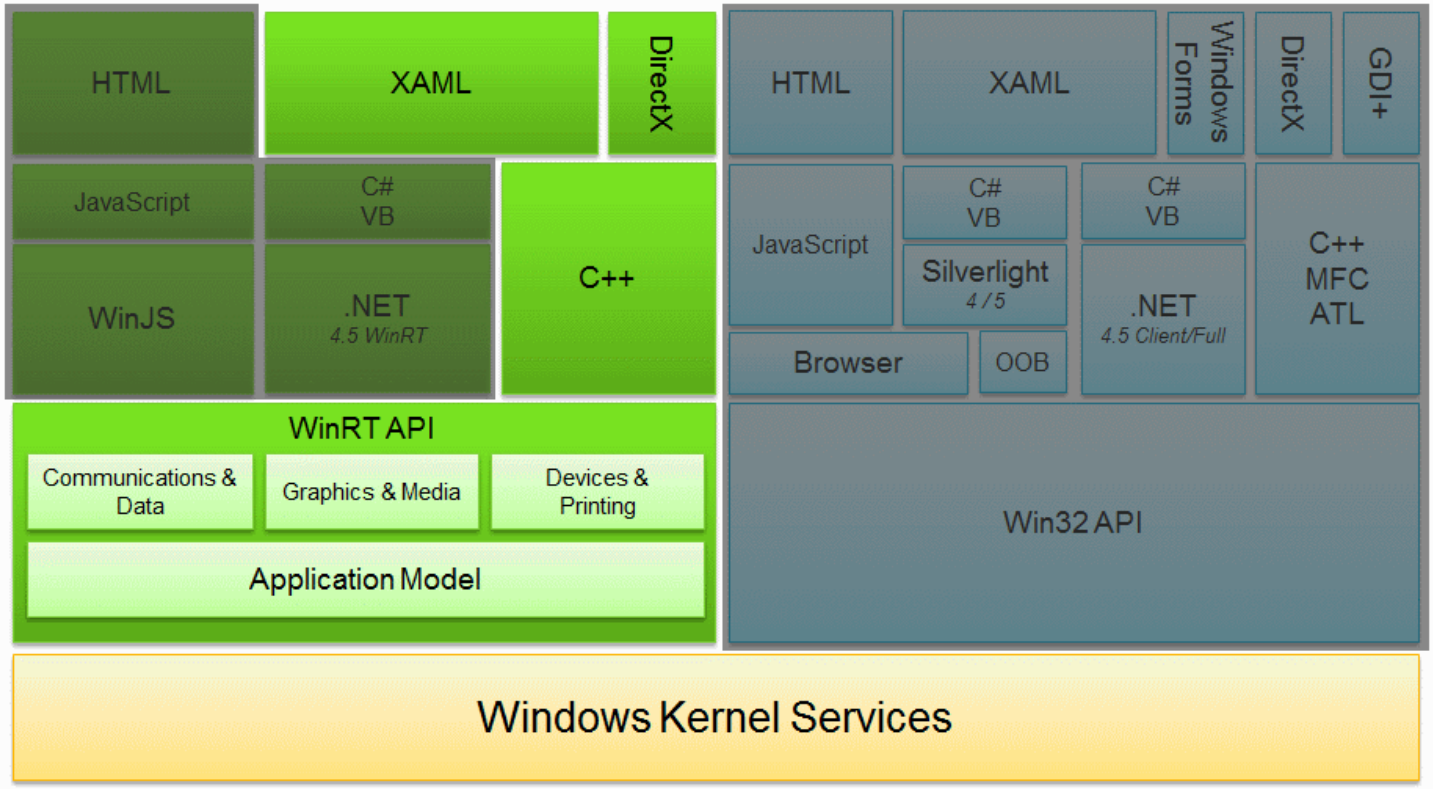


WinRT API

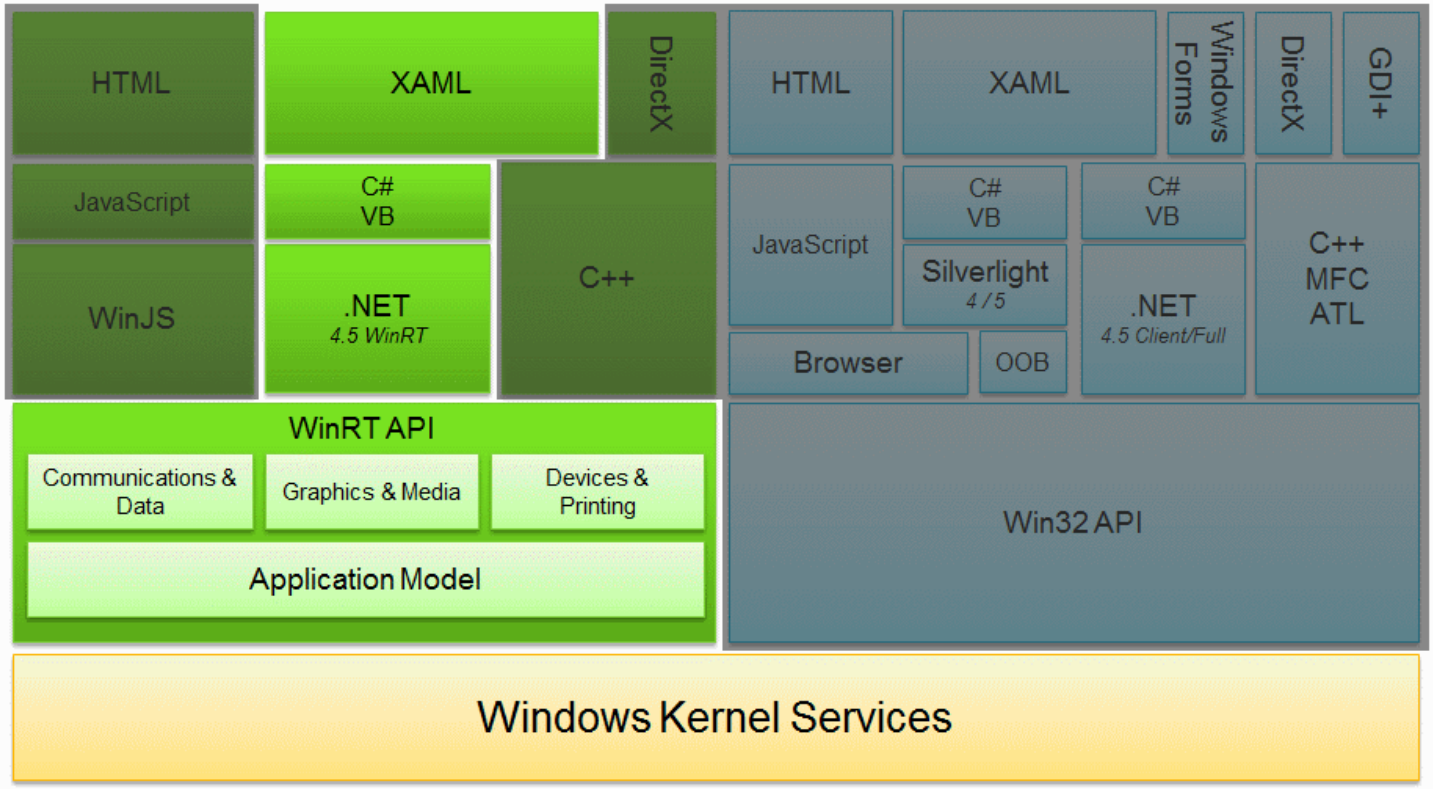
Modern, object-oriented API for Windows, covering

- Data (XML, JSON, Atom)
- Devices & Sensors
- Files
- Globalization
- Graphics
- Media
- Networking
- Printing
- Windows
- Notifications
- Security (Authentication, Credentials, Cryptography)
- Contacts
- UI Automation
- Pickers

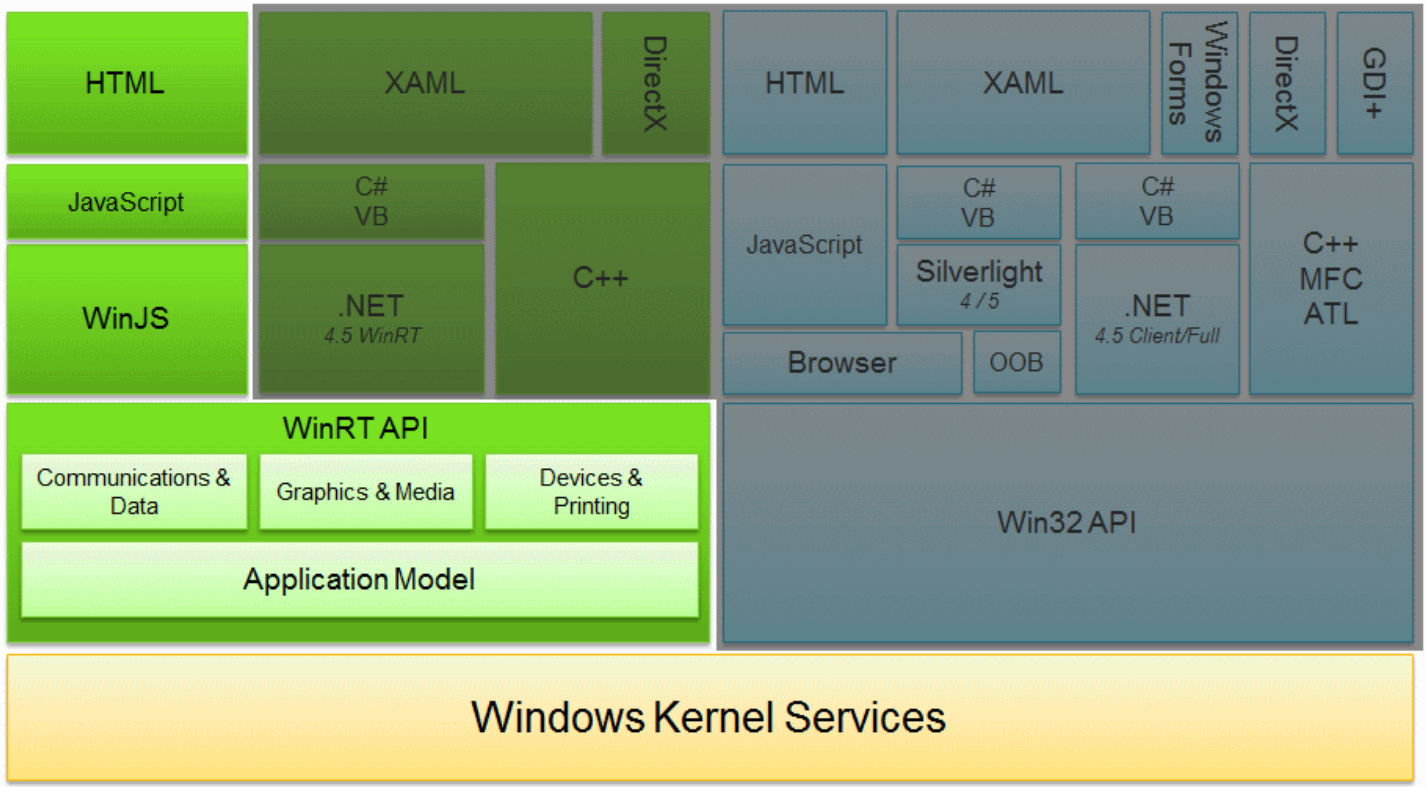
XAML or DirectX with C++



XAML with C# or VB

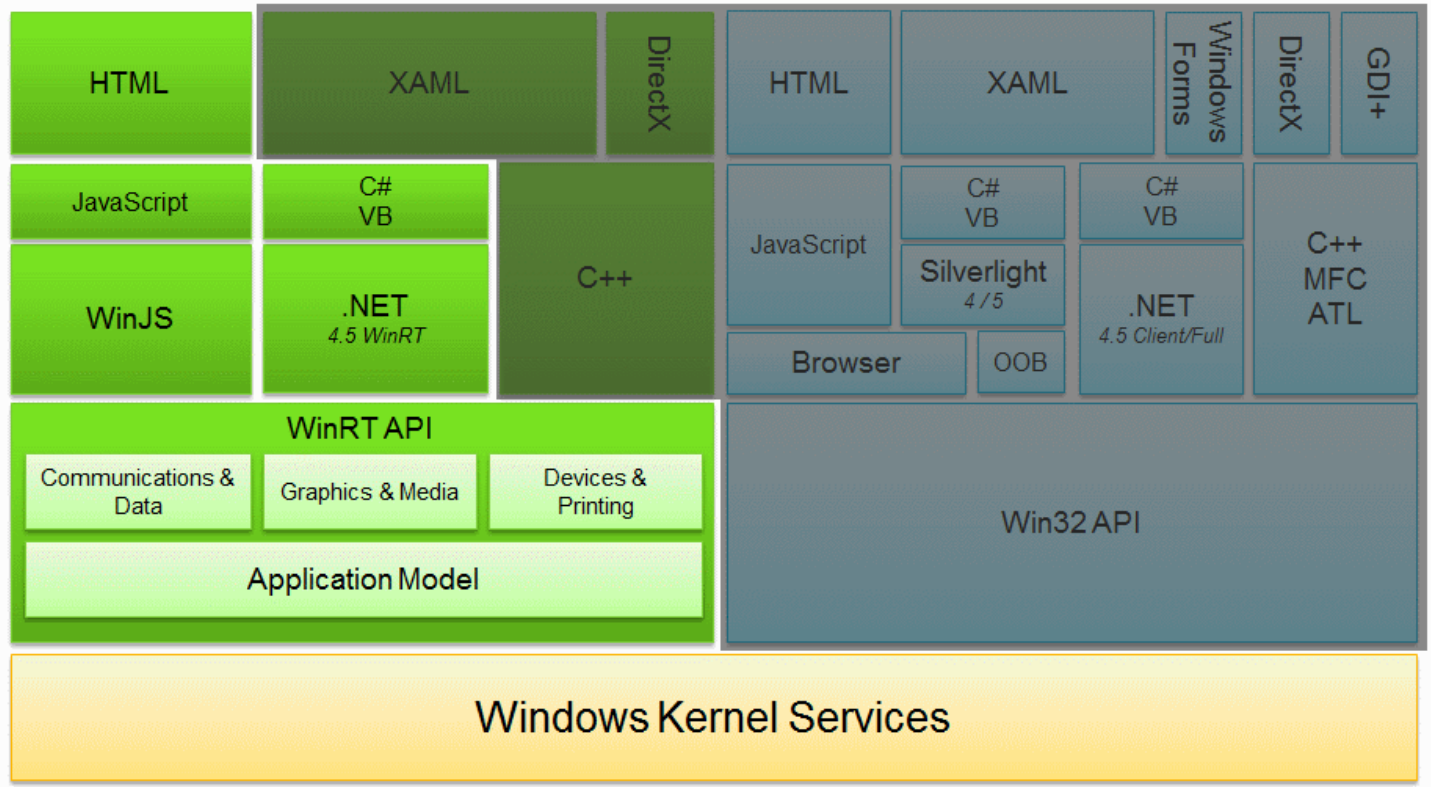


HTML/CSS with Javascript



HTML/CSS with JS, plus .NET

This is what we did for the About Me app



Windows 8 Technology Stacks

It's your choice

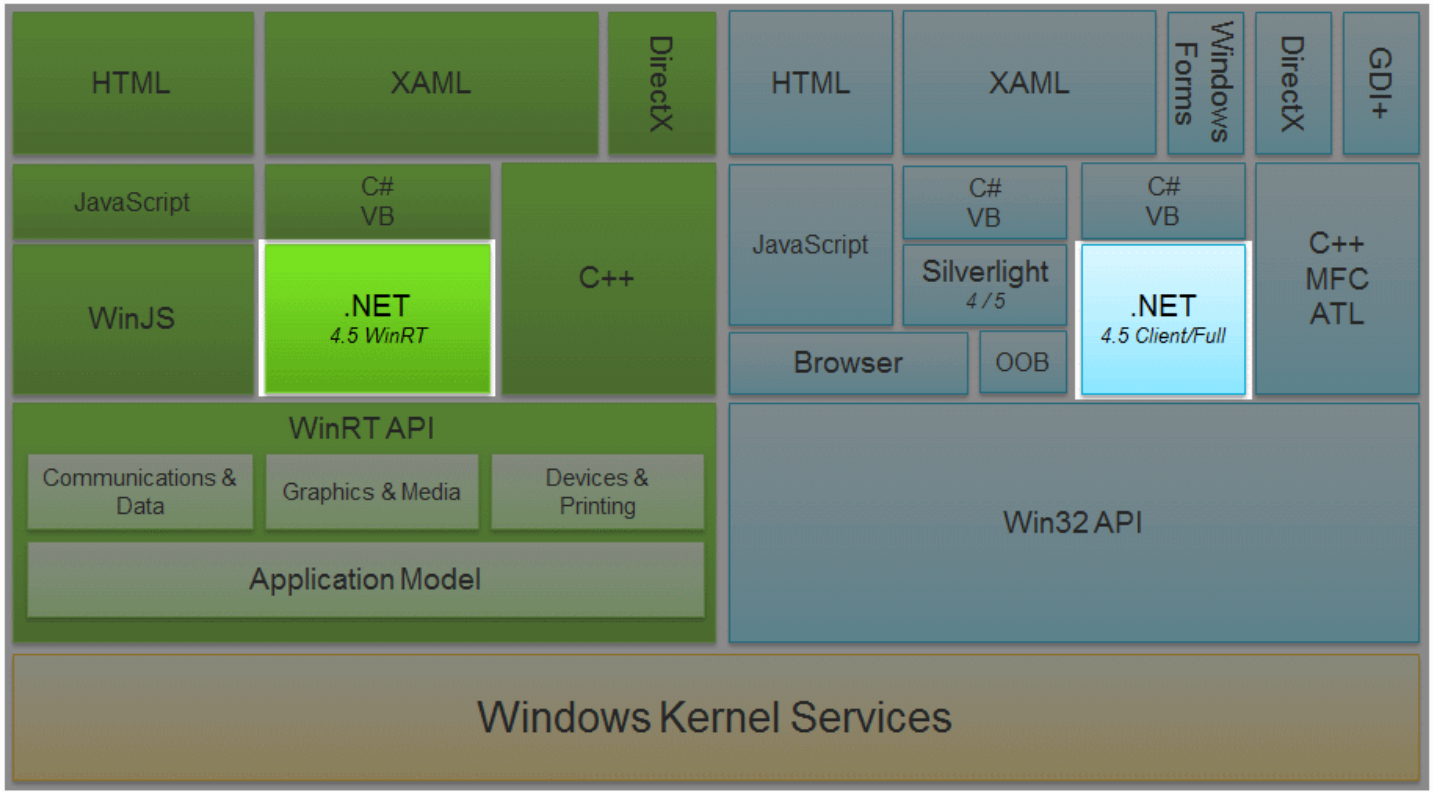
- XAML or DirectX with C++
- XAML with C# (or VB)
- HTML/CSS with Javascript
- HTML/CSS with Javascript, plus .NET libraries (C# or VB)

Windows 8 Technology Stacks

Which one to choose?

- Choose DirectX with C++ for games
- Otherwise, use what you know
  - XAML with C# if you have .NET expertise (especially WPF, Silverlight, or Windows Phone)
  - HTML/CSS with Javascript if you have web expertise
  - Combine HTML/JS with .NET libraries if mixed expertise

Fragmentation of .NET

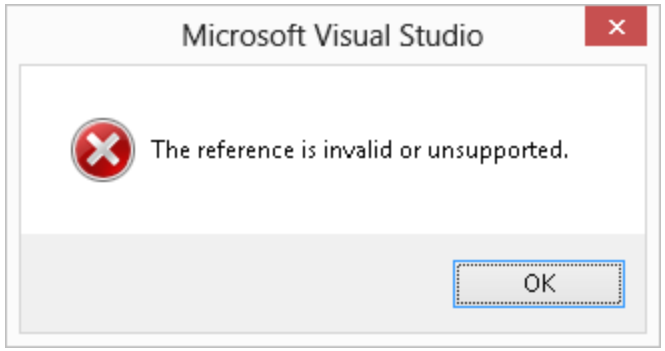


What's left out

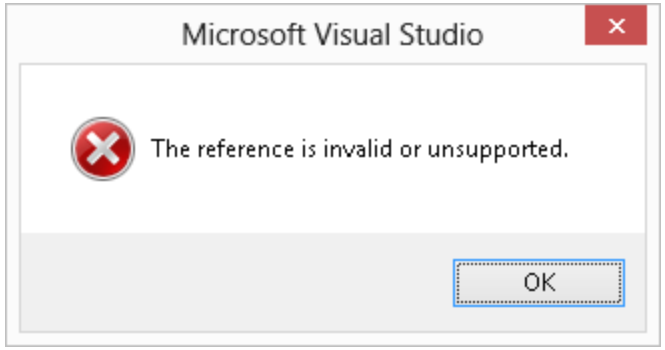
- Types and members that are not applicable to developing Windows Store apps (such as console and ASP.NET types)
- Obsolete and legacy types
- Types that overlap with Windows Runtime types
- Types and members that wrap operating system functionality (such as the event log and performance counters)
- Members that cause confusion (such as the Close method on I/O types)

Source: http://msdn.microsoft.com/en-us/library/windows/apps/br230302.aspx

Trying to use a .NET library



Trying to use a .NET library



Supported API test

FAILED Supported APIs

- **Error Found:** The supported APIs test detected the following errors:

```
API FormatMessageA in kernel32.dll is not supported for this application type. log4net.dll calls this API.
API GetConsoleOutputCP in kernel32.dll is not supported for this application type. log4net.dll calls this A
API GetConsoleScreenBufferInfo in kernel32.dll is not supported for this application type. log4net.dll call
API GetStdHandle in kernel32.dll is not supported for this application type. log4net.dll calls this API.
API OutputDebugStringA in kernel32.dll is not supported for this application type. log4net.dll calls this A
API SetConsoleTextAttribute in kernel32.dll is not supported for this application type. log4net.dll calls t
API System.ICloneable in MSCORLIB, PUBLICKEYTOKEN=B77A5C561934E089 is not supported for this application ty
API System.ApplicationException in MSCORLIB, PUBLICKEYTOKEN=B77A5C561934E089 is not supported for this appl
API System.Configuration.IConfigurationSectionHandler in SYSTEM, PUBLICKEYTOKEN=B77A5C561934E089 is not sup
API System.Runtime.Serialization.ISerializable in MSCORLIB, PUBLICKEYTOKEN=B77A5C561934E089 is not supporte
```

Over 400 unsupported APIs referenced in log4net!

Asynchrony

- Windows 8 apps have to be “fast and fluid”
- Every WinRT API that might take a long time to execute (>50ms) is asynchronous
- Very slick language/library support:
  - `async` keyword
  - `await` keyword
  - `Task<TResult>` class

Asynchrony

```
private async Task<XElement> LoadPageListAsync()
{
    await EnsureDataEnvironmentInitializedAsync();

    using (var stream = await this.GetFileStreamAsync("PageList.xml"))
    {
        return XDocument.Load(stream).Element("pageList");
    }
}
```

Asynchrony

Asynchrony will permeate your app!

- Code that calls async methods must be async itself

Don't fight it – go with it.

Payoff: your app will be responsive.

Windows Store Certification

Ways to distribute your app:

- Enterprise scenarios
- Windows Store

➔ You have to follow the Store's rules.

- Administrative rules
- Content rules
- Technical rules

Windows Store Certification Tool

The SDK provides an easy-to-use tool that scans apps for compliance with the Store's technical requirements.

You should run this periodically during development to avoid surprises at the end.

Razorfish.Win8.Utility

Utility library currently covering

- Logging
- Settings (XML-based)
- File access
- Tile notifications

Callable from C# or Javascript

Contact me for more details

Dos and Don'ts

Do…

- Consider the experience of your team when choosing which tech stack to use
- Make sure you're designing for full/fill/snap/portrait views from the beginning
- Run certification tool periodically during development
- Learn to love asynchronous code

Dos and Don'ts

Don't…

- Think that developing Windows Store apps is like developing for earlier versions of Windows
- Assume that web developers will be immediately productive on Windows 8 projects
- Expect your app will sail through the Windows Store acceptance process the first time
- Expect to reuse existing .NET libraries

Demo

Is this time for upgrade to windows8?

PROS	CONS
Consumer may easily adapt to windows8, but the change is a big decision for businesses — both small and large.	From an IT perspective, the changes made to Windows 8 could end up costing businesses a lot of money depend on the existing hardware currently in use.
Take advantage of new User interface & experience building apps on windows8.	Required additional Training and learning curve for internal Business groups and organization.
Latest advancements in hardware and processing tools, load programs quicker and manage tasks more efficiently.	If you organization still trying to work out the kinks of the migration to Windows 7 this is not the right time to upgrade to windows8.
Windows 7 will be officially supported (regularly receiving security updates) until 2015, organization will have more time to upgrade.	Finally As an added complication, Microsoft doesn't expect to release any service packs to Windows 8 until sometime in 2013. There is a high risk factor, if you run into issues after rolling out the upgrade at corporate level.

Q&A