// interknowlogy /

http://www.interknowlogy.com/

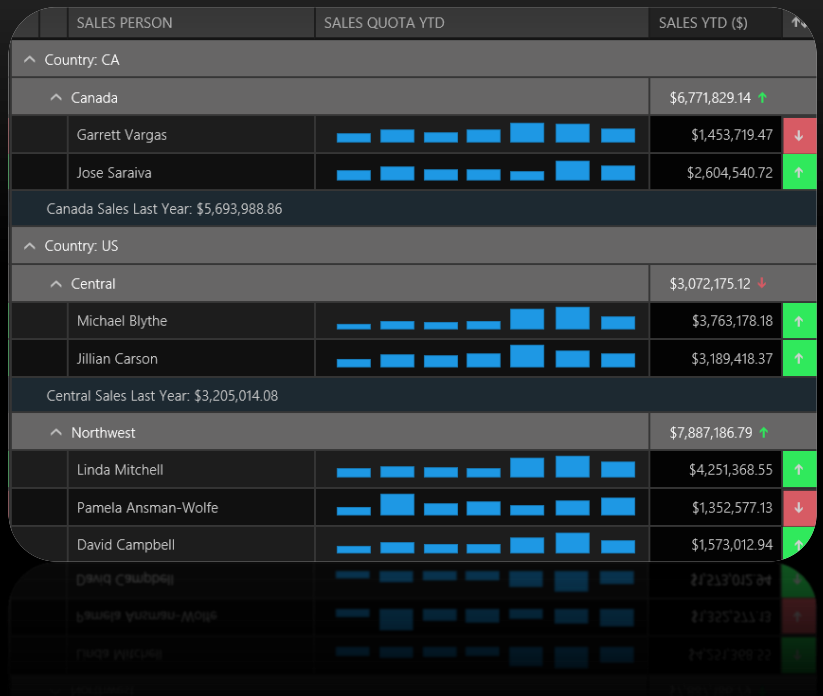
EXTENDING WINDOWS 8.1 WITH CUSTOM CONTROLS

Philip Japikse (@skimedic)
skimedic@outlook.com
www.skimedic.com/blog
Microsoft MVP, ASPInsider, MCSD, MCDBA, CSM, CSP
Principal Architect, InterKnowlogy

PHIL.TOSTRING()

- Principal Architect, InterKnowlogy, Inc.
  - http://www.interknowlogy.com
- Microsoft MVP, ASPInsider, MCSD, MCDBA, CSM, CSP
- Founder, Agile Conferences, Inc.
- President, Cincinnati .NET User's Group
- Co-host, Hallway Conversations
  - www.hallwayconversations.com

AGENDA

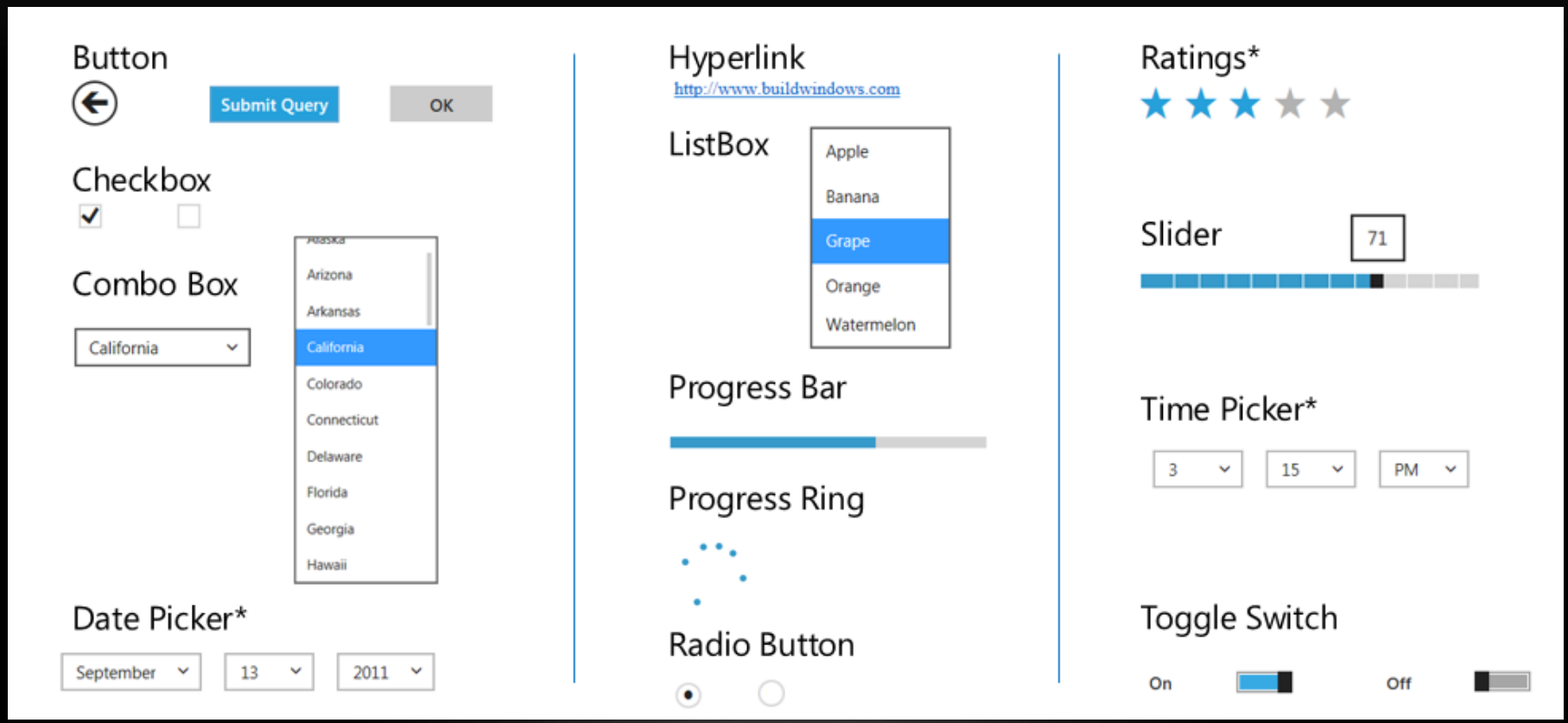


- Why should I care?
- Properties/events
- Make it toolable
- Distribution
- Resources

THE PERSONALITY OF WINDOWS CONTROLS

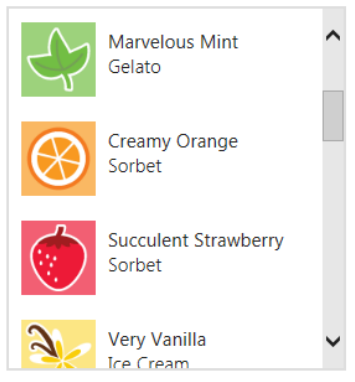
- Optimized for touch first
- Designed for keyboard and mouse
- Independently animated

WINDOWS “OUT OF THE BOX” COMMON CONTROLS

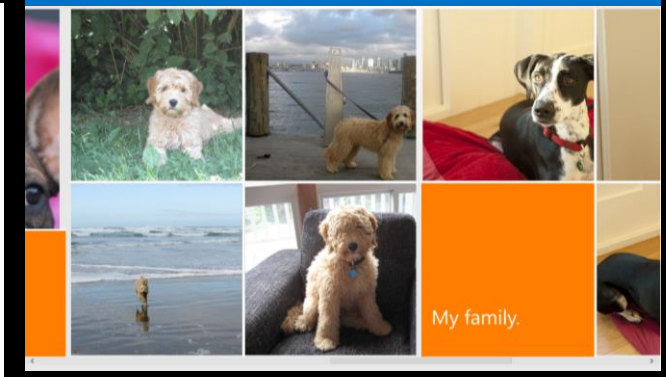


DATA VIEWS

ListView



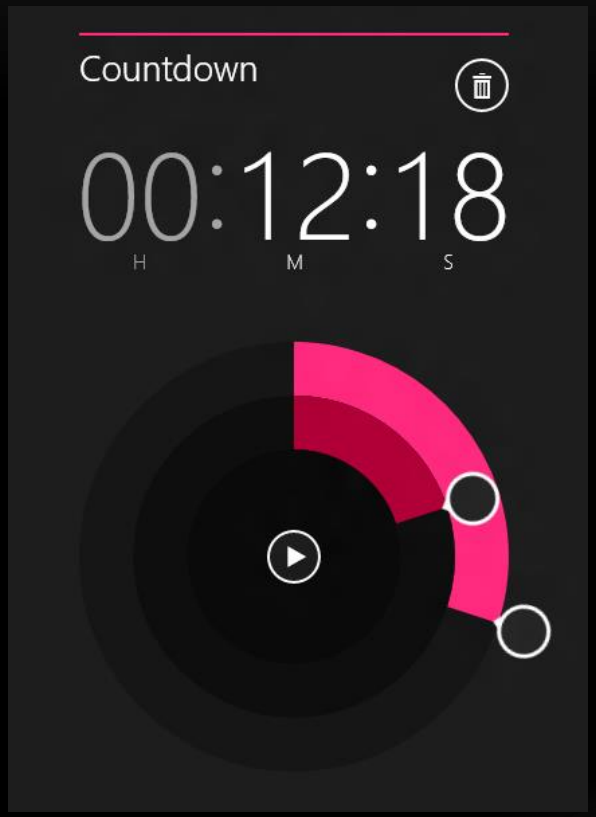
GridView

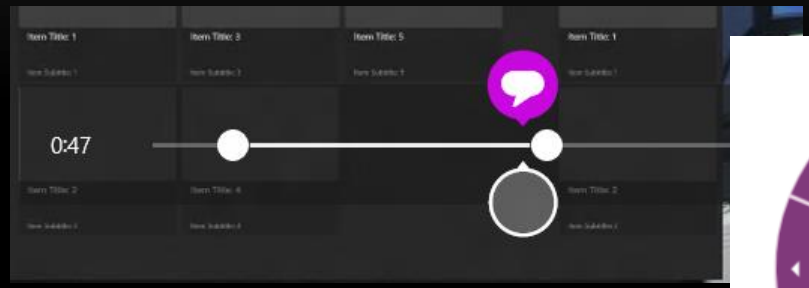


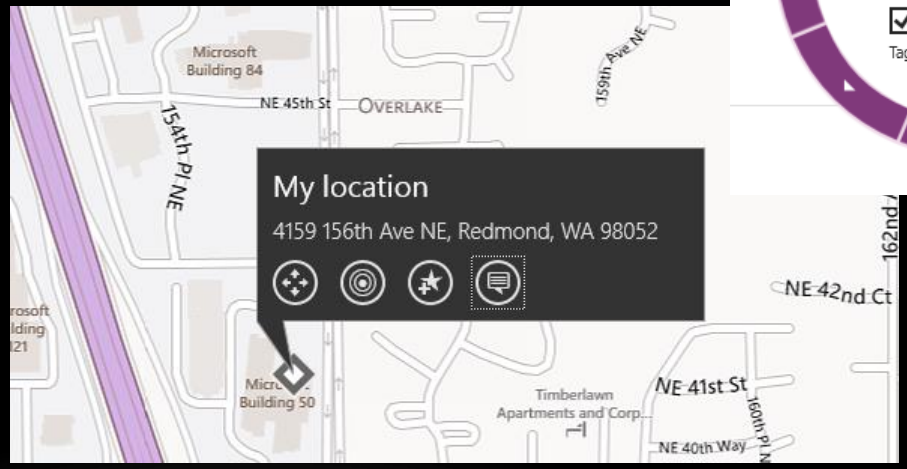
FlipView

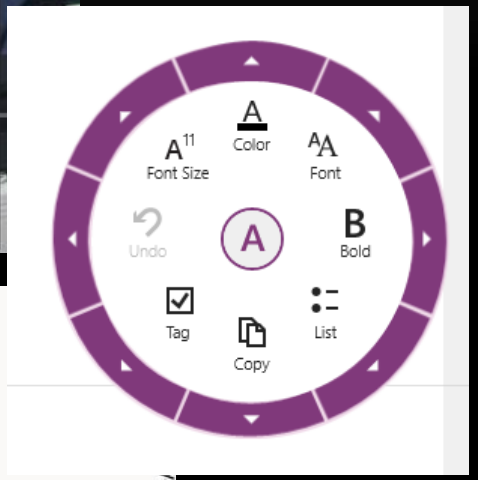
WHY EXTEND THE UI PLATFORM?

ADAPTING TO YOUR UNIQUE PERSONALITY









EXTENDING THE UI PLATFORM THROUGH CUSTOM CONTROLS

CUSTOM CONTROL MOTIVATION

- Encapsulate logic
- Promote reuse
- Increase consistency
- Increase quality
- Release faster
- Get home on time!

AS EASY AS 1-2-3 (4-5)

- Create the user interface
- Add properties/methods/events
- Add metadata for IntelliSense
- Make it toolable in Blend/Visual Studio
- Ship it!

CREATING THE CONTROL ASSEMBLY (XAML / C#)

- Start with a class library (Windows Store apps)
- Add templated control
  - Adds themes \ Generic.xaml
- Design your control
  - Control must have default no-arg constructor

CREATING THE CONTROL UI (HTML / WINJS)

- Start with a JavaScript file
- Add WinJS namespace and WinJS class
  - Constructor needs ‘element’ as first parameter
- Draw your control

DEFINING THE WINJS CONTROL

```
(function () {
    "use strict";
    WinJS.Namespace.define("Samples.UI", {
        ClockControlHTMLUI: WinJS.Class.define(function (element) {
            // Create an element if none provided
            element = element || document.createElement("div");
            element.winControl = this;
            this.element = element;
        element.style.display = "inline-block";
        this._drawClock();
        }, {
        _drawClock: function () {
                // Draw the control
            },
        }, {
        })
    });
})();
```

DEMO

Creating the control UI

ADDING AND CONSUMING PROPERTIES

ADDING/CONSUMING PROPERTIES (XAML / C#)

- Adding
  - Dependency properties (if using styles)
  - Standard getter/setters
  - Set defaults in XAML or #
- Consuming
  - Through markup
  - Through code

ADDING PROPERTIES (XAML / C#)

```
// Creating dependency properties
public static readonly DependencyProperty ClockFillBrushProperty =
    DependencyProperty.Register("ClockFillBrush", typeof(Brush),
    typeof(ClockControlXAMLProps), new PropertyMetadata(null));
public Brush ClockFillBrush
{
    get { return (Brush)this.GetValue(ClockFillBrushProperty); }
    set { this.SetValue(ClockFillBrushProperty, value); }
}
--------------------------------------------------------------------------------
<!—Defaults in XAML-->
<Setter Property="ClockFillBrush" Value="Red"/>
<Setter Property="MinWidth" Value="200"/>
<Setter Property="MinHeight" Value="200"/>
```

CONSUMING PROPERTIES (XAML / C#)

```
<localProps:ClockControlXAMLProps Grid.Column="1" x:Name="MyClockControlProps"
        ClockFillBrush="Gray" MinWidth="150 MinHeight="150 />

-------------------------------------------------------------

var myClock = new B_CustomClockXAMLProps.ClockControlXAMLProps()
{
      ClockFillBrush = new SolidColorBrush(Colors.Blue),
      MinWidth = 150,
      MinHeight = 150
};

MyClockControlProps.ClockFillBrush = new SolidColorBrush(Colors.Black);
```

ADDING/CONSUMING PROPERTIES (HTML / WINJS)

- Adding
  - Add options parameter to class constructor
    - Call WinJS.UI.setOptions
  - Add properties as class instance methods
    - Redraw control as necessary
- Order of processing is key!
- Consuming
  - Through markup via data-win-options
  - Through code

ADDING PROPERTIES (HTML / WINJS)

```
ClockControlHTMLProps: WinJS.Class.define(function (element, options) {
    // allow a null options parameter
    options = options || {};
    // WinJS utility function to assign/create properties from anonymous options object
    WinJS.UI.setOptions(self, options);
    //Omitted for brevity
  } , {
    color: {
        get: function () {
            return this._color || "red";
        },
        set: function (value) {
            if (this._color != value) {
                this._color = value;
                //Redraw the User Interface
                this._drawClock();
            }
        }
    },
  },
})
```

CONSUMING PROPERTIES (HTML / WINJS)

```
<div id="clockControlPropsDiv"
    data-win-control="Samples.UI.ClockControlHTMLProps"
    data-win-options="{color:'yellow',width:400, height:400}">
</div>
-------------------------------------------------------------
var clock = new Samples.UI.ClockControlHTMLProps(MyClock,
            { color: 'blue' });
clock.color='green';

clockControlPropsDiv.winControl.color='green‘;
```

Demo: Adding/consuming properties

ADDING/CONSUMING EVENTS

ADDING/CONSUMING EVENTS (XAML / C#)

- Adding
  - PropertyChangedCallBack to dependency properties
  - Standard .NET events
- Consuming
  - Through markup / code-behind
  - Through standard event handlers in C#

ADDING CALLBACK EVENTS (XAML / C#)

```
public static readonly DependencyProperty IsRunningProperty =
   DependencyProperty.Register("IsRunning", typeof(bool),
   typeof(ClockControlXAMLEvents),
   new PropertyMetadata(true,
      PropertyChangedCallback(OnIsRunningChanged)));


private static void OnIsRunningChanged(
   DependencyObject d, DependencyPropertyChangedEventArgs e)
{
     var clock = d as ClockControlXAMLEvents;
     clock.UpdateTimerState();
}
```

ADDING STANDARD EVENTS (XAML / C#)

```
public event EventHandler<ClockEventArgs> Ticked;

public class ClockEventArgs : System.EventArgs
{
    public DateTime When { get; set; }
    public ClockEventArgs(DateTime when)  {this.When = when; }
}
private void RaiseEvents()
{
    var currentTime = DateTime.Now;
    var eh = this.Ticked;
    if (eh != null)
    {
        eh(this, new ClockEventArgs(currentTime));
    }
}
```

CONSUMING EVENTS (XAML / C#)

```
<localEvents:ClockControlXAMLEvents
  x:Name="MyClockControlEvents"
  FiveSecondsElapsed="MyClock_FiveSecondsElapsed“ />
-----------------------------------------------------------------
MyClockControlEvents.Ticked += MyClock_Ticked;
private void MyClock_Ticked(object sender, ClockEventArgs e)
{
    ticker.Text = e.When.ToString();
}
private void MyClock_FiveSecondsElapsed(
      object sender, ClockEventArgs e)
{
    ticker5.Text = e.When.ToString();
}
```

ADDING/CONSUMING EVENTS (HTML / WINJS)

- Adding
  - Use WinJS.Class.mix to add the events
    - WinJS.UI.DOMEventMixin
    - WinJS.Utilities.createEventProperties
  - Use this.dispatchEvent to raise the event
- Consuming
  - Use addEventListener to bind to the event

ADDING EVENTS (HTML / WINJS)

```
WinJS.Namespace.define("Samples.UI", {
  ClockControlHTMLEvents: WinJS.Class.define(function (element, options) {
    //omitted for brevity
  }, {
    //omitted for brevity
    _drawHands: function () {
      //Omitted for brevity
      this.dispatchEvent("tick", { when: now });
      if (sec % 5 == 0) {
          this.dispatchEvent("fiveseconds", {when:now});
      };
    },
  }, {
  })
});
WinJS.Class.mix(Samples.UI.ClockControlHTMLEvents, WinJS.UI.DOMEventMixin,
   WinJS.Utilities.createEventProperties("tick","fiveseconds"));
```

CONSUMING EVENTS (HTML / WINJS)

```
clockCtDiv.winControl.addEventListener("tick", function (e) {
    tick.innerText = e.detail.when;
});
clockCtlDiv.winControl.addEventListener("fiveSeconds", function (e) {
    fiveSeconds.innerText = e.detail.when;
});
```

Demo: Adding/consuming events

SUPPORTING INTELLISENSE

SUPPORTING INTELLISENSE (XAML / HTML)

- XAML/C#
  - Add /// comments to:
    - Dependency properties: Add to CLR property
    - CLR properties, events, methods
- HTML/WinJS
  - Add /// comments to
    - In class constructor
    - Above class fields, properties, methods
  - http://msdn.microsoft.com/en-us/library/hh524453.aspx

INTELLISENSE SUPPORT (XAML / C#)

```
public static readonly DependencyProperty IsRunningProperty = /*Omitted for Brevity*/;

/// <summary>
/// Is it running?
/// </summary>
public bool IsRunning
{
   get { return (bool)this.GetValue(IsRunningProperty); }
   set { this.SetValue(IsRunningProperty, value);  }
}
/// <summary>
/// Event that fires every five seconds
/// </summary>
public event EventHandler<ClockEventArgs> FiveSecondsElapsed;
```

INTELLISENSE SUPPORT (HTML / WINJS)

```
/// <summary>Creates a Clock Control</summary>
/// <param name="element" type="HTMLElement">The root element for the control</param>
/// <param name="options" type="Object" optional="true">The JSON options for the
control</param>

/// <field name='color' type='string'> Sets the color of the hands and the outline of the clock
face</field>
/// <field name='height' type='int'> Sets the height of the clock face</field>
```

VISUAL STUDIO / BLEND SUPPORT

MAKE YOUR CONTROL TOOLABLE (XAML / C#)

- Visual Studio / Blend:
  - Create files / metadata package
    - SDKManifest.xml
    - Add DLL, PRI, and themes
- Add to the Windows 8 extensions folder
  - %program files (x86)\Microsoft SDKs\Windows\v8.0\ExtensionSDKs

MAKE YOUR CONTROL TOOLABLE (HTML / JS)

- Visual Studio / Blend:
  - Create files / metadata package
    - SDKManifest.xml
    - samples.ui_api_oam.xml
    - ui_<controlname>_oam.xml
  - Add to the Windows 8 extensions folder
    - %program files (x86)\Microsoft SDKs\Windows\v8.0\ExtensionSDKs
- Blend specific:
  - Properties must use getter/setter paradigm

DISTRIBUTION

CREATE A VSIX PACKAGE (XAML / C#)

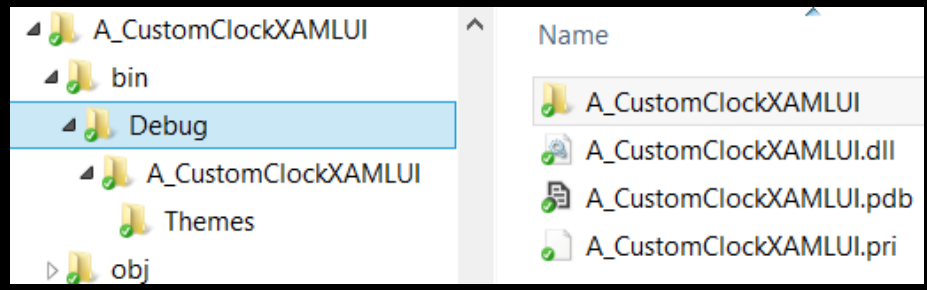
- Set property for GenerateLibraryLayout
- Install the Visual Studio 2013 SDK
- Create a VSIX project
  - Create proper folder structure
  - Add .pri, themes\xaml, .dll as assets
- Build project in release mode

CREATING THE CONTROL ASSEMBLY

Project File Update

```
<Project …>
  <Import …/>
  <PropertyGroup>
   …
  <GenerateLibraryLayout>
   true
  </GenerateLibraryLayout>
  </PropertyGroup>
  …
</Project>
```

Build Result



RESOURCES

- Visual Studio 2013 SDK: http://bit.ly/1n5eFG0
- Creating VSIX packages: http://bit.ly/11sQN5p
- XAML XML documentation: http://bit.ly/14nGf67
- WinJS specific
  - OpenAjax API Metadata Spec: http://bit.ly/108fHpY
  - OpenAjax Widget Metadata Spec: http://bit.ly/11sPy6a
  - XML Documentation: http://bit.ly/15reEzM

CONTACT ME

- skimedic@gmail.com
- www.skimedic.com/blog
- www.twitter.com/skimedic
- www.hallwayconversations.com

- www.about.me/skimedic