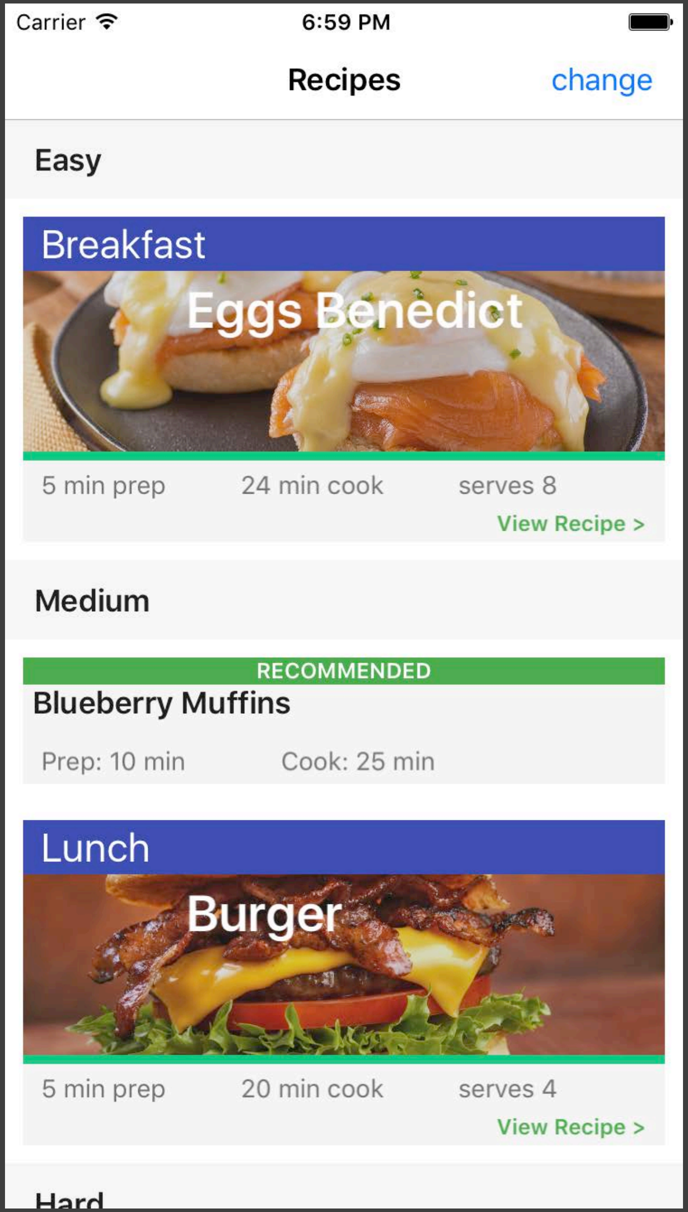
Styling Xamarin.Forms Apps

Matthew Soucoup
PRINCIPAL

@codemillmatt   codemilltech.com



Carrier 6:59 PM

Recipes change

Easy

5 min prep 24 min cook serves 8

View Recipe >

Medium

RECOMMENDED

Blueberry Muffins

Prep: 10 min Cook: 25 min

5 min prep 20 min cook serves 4

View Recipe >

Hard

Styling Apps

Understanding styles

Style hierarchy

Creating implicit styles

Swapping at runtime

Understanding Styles

Understanding Styles

Label A

FontSize = Medium
TextColor = Blue
Margin = 5

Label B

FontSize = Medium
TextColor = Blue
Margin = 5

Label C

FontSize = Medium
TextColor = Blue
Margin = 5

Label D

FontSize = Medium
TextColor = Blue
Margin = 5

Label E

FontSize = Medium
TextColor = Blue
Margin = 5

Label F

FontSize = Medium
TextColor = Blue
Margin = 5

Understanding Styles

MyStyle: FontSize = Medium, TextColor = Blue, Margin = 5

Label A

Style = MyStyle

Label B

Style = MyStyle

Label C

Style = MyStyle

Label D

Style = MyStyle

Label E

Style = MyStyle

Label F

Style = MyStyle

Xamarin.Forms Styles

Maintain a consistent and customized UI

Collection of property setters

Defined for specific types of controls

Defined in resource dictionary

```xml
<ResourceDictionary>
    <Style x:Key="prepInfoStyle" TargetType="Label">

    </Style>
</ResourceDictionary>
```

Explicit Style

Within ResourceDictionary

TargetType - must always declare

Key - makes explicit

```
<Style x:Key="prepInfoStyle" TargetType="Label">
    <Setter Property="HorizontalTextAlignment" Value="Center"/>
    <Setter Property="VerticalTextAlignment" Value="Center"/>
    <Setter Property="TextColor" Value="#CF5C36"/>
</Style>
```

Style Setters

Setter collection

Declare property - must be bindable

Specify value

```
<Label Style="{StaticResource prepInfoStyle}"
                        Text="{Binding PreparationTime}" />
<Label Style="{StaticResource prepInfoStyle}"
                        Text="{Binding CookTime}" />
<Label Style="{StaticResource prepInfoStyle}"
                        Text="{Binding NumOfServings}" />
```

Consuming a Style

Style property

StaticResource

- Reference the key name

Demo

Create a style

Consume it

Understanding Styles Summary

Organize consistent, customized UI

Style class

- Target specific control type
- Collection of property setters

Define in ResourceDictionary

Consume via Style property

Style Hierarchy and Inheritance

Where Do Styles Live?

Globally

Page Level

Control Level

Global Style Declaration

```
<Application xmlns="http://xamarin.com/schemas/2014/forms" ...
<Application.Resources>
    <ResourceDictionary>

        <Style x:Key="RecipeLabel" TargetType="Label">
            <Setter Property="Margin" Value="-30,5,0,0" />
            <Setter Property="FontSize" Value="28" />
            <Setter Property="FontAttributes" Value="Bold" />
            <Setter Property="TextColor" Value="#FFFFFF" />
        </Style>

    </ResourceDictionary>
</Application.Resources>
</Application>
```

Style Tips

```xml
<Thickness x:Key="LeftPad">-30,5,0,0</Thickness>
<Color x:Key="LightText">#FFFFFF</Color>

<Style x:Key="RecipeLabel" TargetType="Label">
    <Setter Property="Margin" Value="{StaticResource LeftPad}" />
    <Setter Property="FontSize" Value="28" />
    <Setter Property="FontAttributes" Value="Bold" />
    <Setter Property="TextColor" Value="{StaticResource LightText}" />
</Style>
```

Implicit Styles

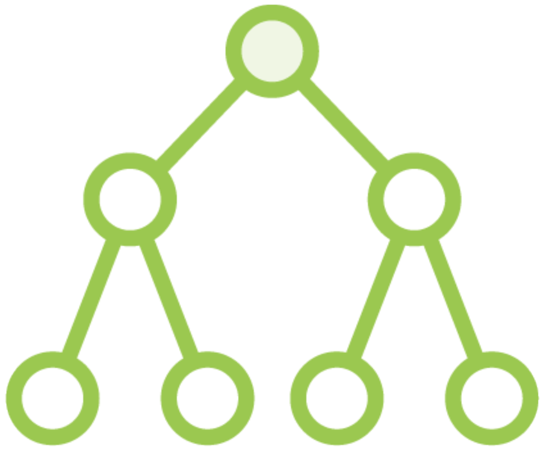
All controls of TargetType use

TargetType must match exactly to control type

No x:Key in style declaration

Control’s Style not set

Style Inheritance



Promotes style reuse

Override or define new properties

Implicit can be derived from explicit

- Not other way

Inherit from styles at same level or above

Use BasedOn keyword

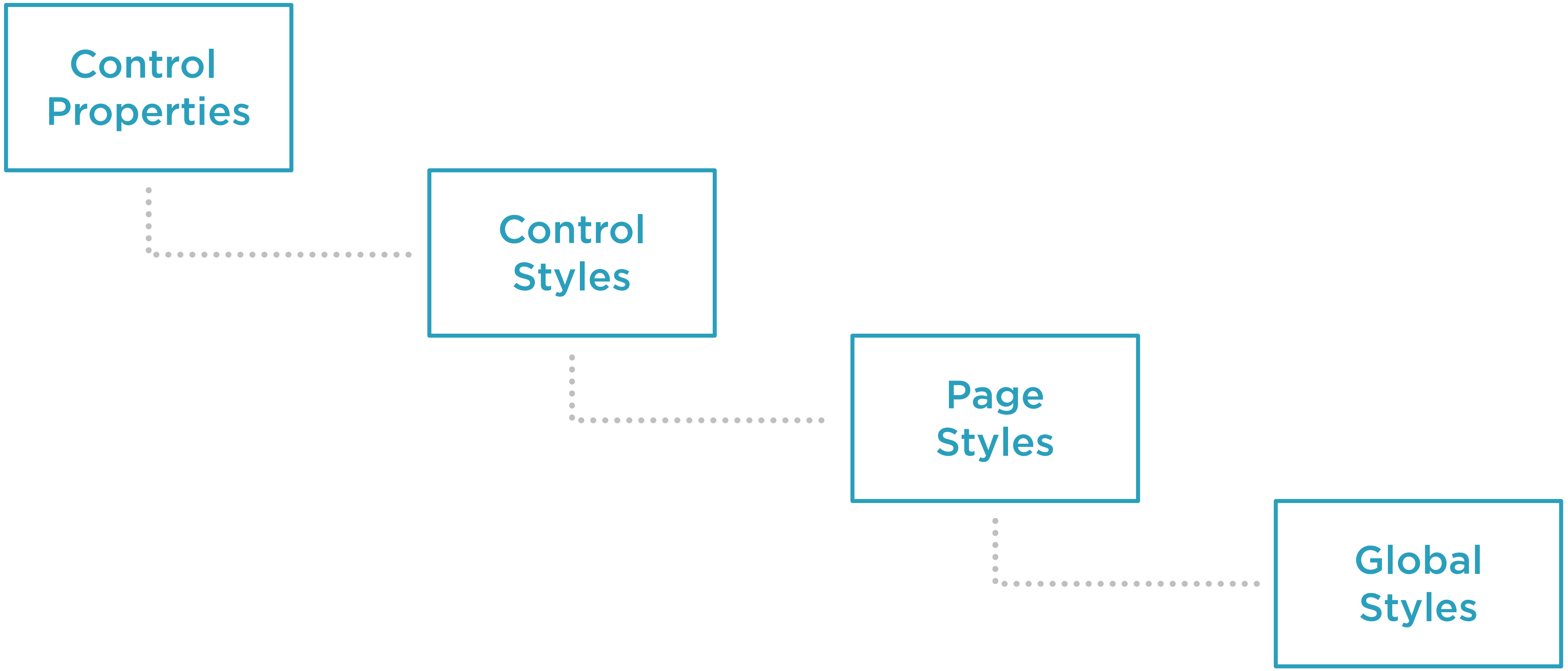
Style Inheritance

```
<Style x:Key=”baseStyle" TargetType="View">
    <Setter Property="BackgroundColor" Value="Red" />
    <Setter Property="Margin" Value="10,0,0,0" />
</Style>

<Style BasedOn="{StaticResource baseStyle}"
            x:Key="lblStyle" TargetType="Label">
    <Setter Property="TextColor" Value="Black" />
</Style>

<Style BasedOn="{StaticResource baseStyle}"
            x:Key="btnStyle" TargetType="Button" >
    <Setter Property="TextColor" Value="Gray" />
    <Setter Property="BackgroundColor" Value="#4286f4" />
</Style>
```

Style Precedence Rules



Demo

Global style

Implicit style

Inheritance and precedence

Tips and pointers

Style Hierarchy and Inheritance Summary

Defined at three levels

- Global
- Page
- Control

Precedence rules

Implicit applies to all target controls

Styles can derive from others

Dynamic Styles

Dynamic Styles

Styles

Cannot change internal structure

Property definitions stay the same

Dynamic Styles

Enables style changes at runtime

Declaring the style stays the same

Consume with DynamicResource

Inherit with BaseResourceKey

Ability for app theming

Demo

Create and consume dynamic styles

Add app theming

Dynamic Styles Summary

Enables style changes at runtime

Style declaration the same

DynamicResource

App themes possible

Styles

- Consistent, customizable UI
- Collection property setters
- ResourceDictionary

Defined at three levels

Precedence rules

Inheritance

Dynamic