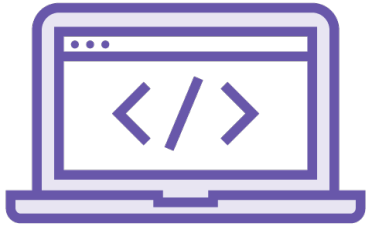
Moving Beyond the Basics with Xamarin.Forms

Matthew Soucoup
PRINCIPAL

@codemillmatt codemilltech.com

Xamarin.Forms Is

Shared
user interface



Shared
application logic

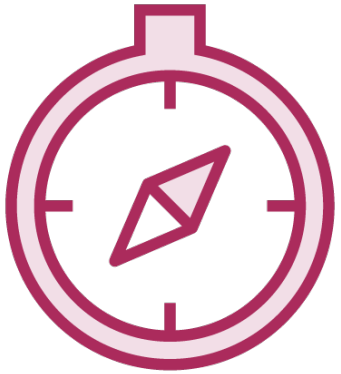
Platform
specific features

Xamarin.Forms Apps Need To

Accept and transform input

Display varying, dynamic data

Maintain a consistent look



Embed native UI elements

Xamarin.Forms Apps Need Reusable Components

Extending Input Controls

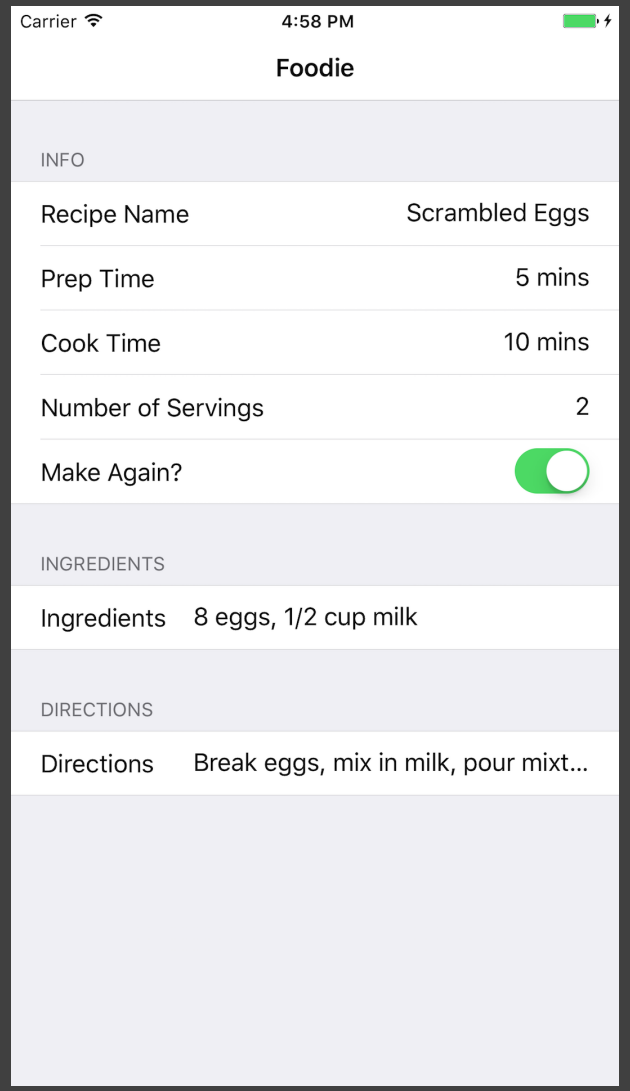
User Input Topics

Entering data

Creating custom cells

Extending functionality with Behaviors

TableView

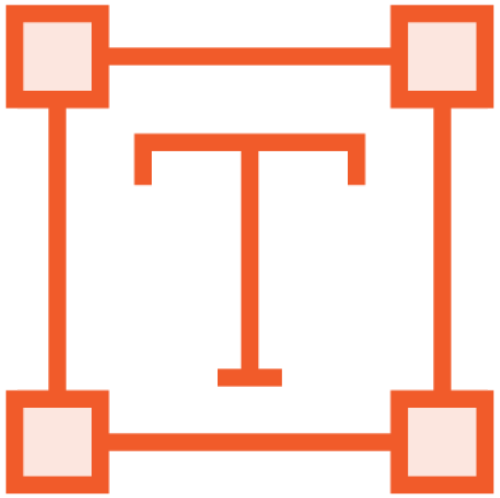


Root

Section

Cell

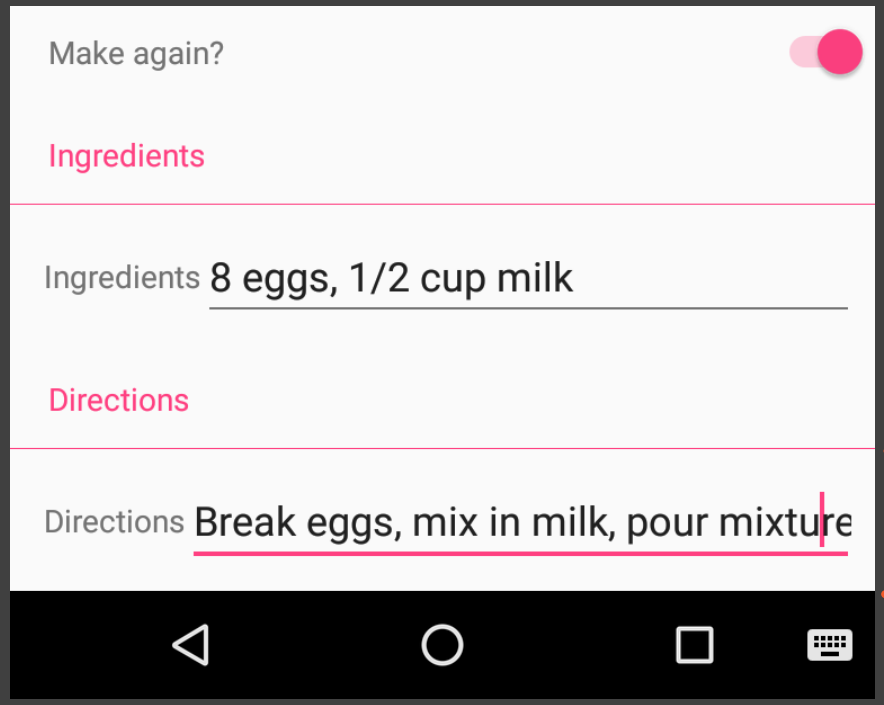
Built-In Editing Cells



Entry cell

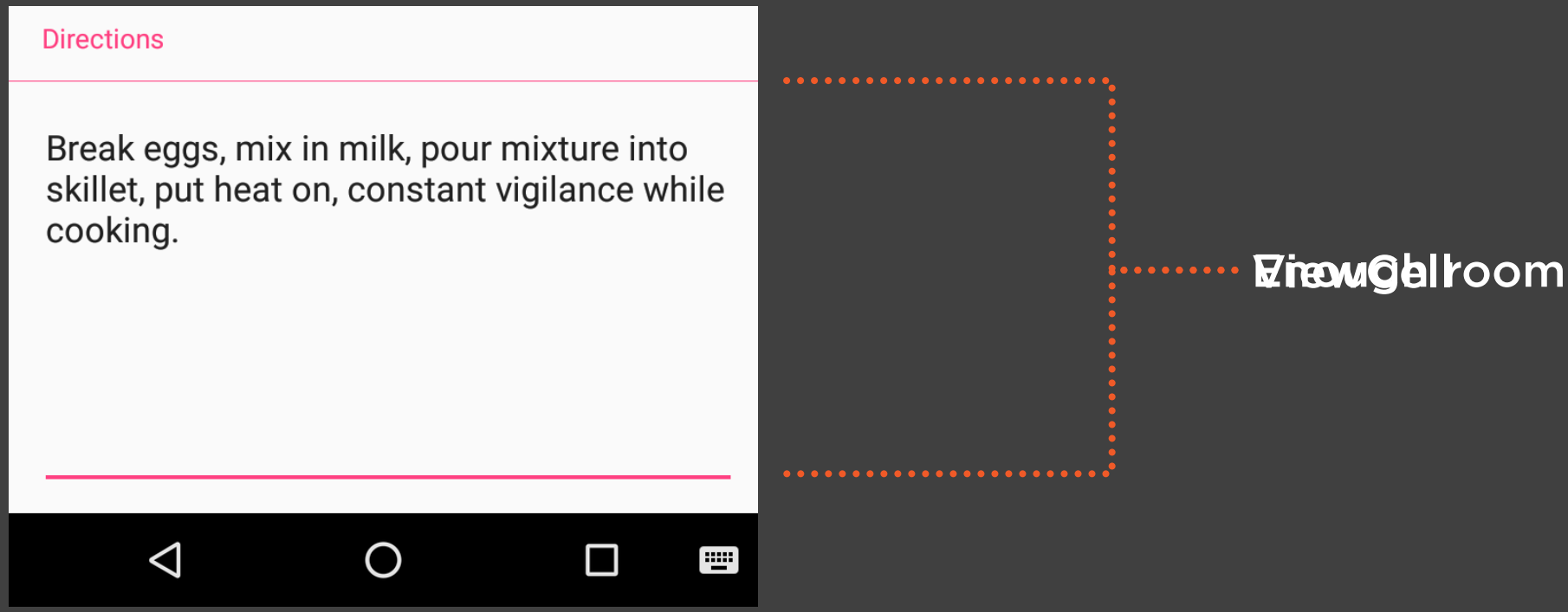
Switch cell

Custom Cells



Not enough room

Custom Cells



```
<TableView HasUnevenRows="true">

...

<ViewCell>
  <StackLayout
     Orientation="Horizontal">
    <Label
      Text="Recipe Name” />
    <Entry
      Text="Scrambled Eggs"/>
  </StackLayout>
</ViewCell>
```

- Multiple row heights
- Build custom visual layout

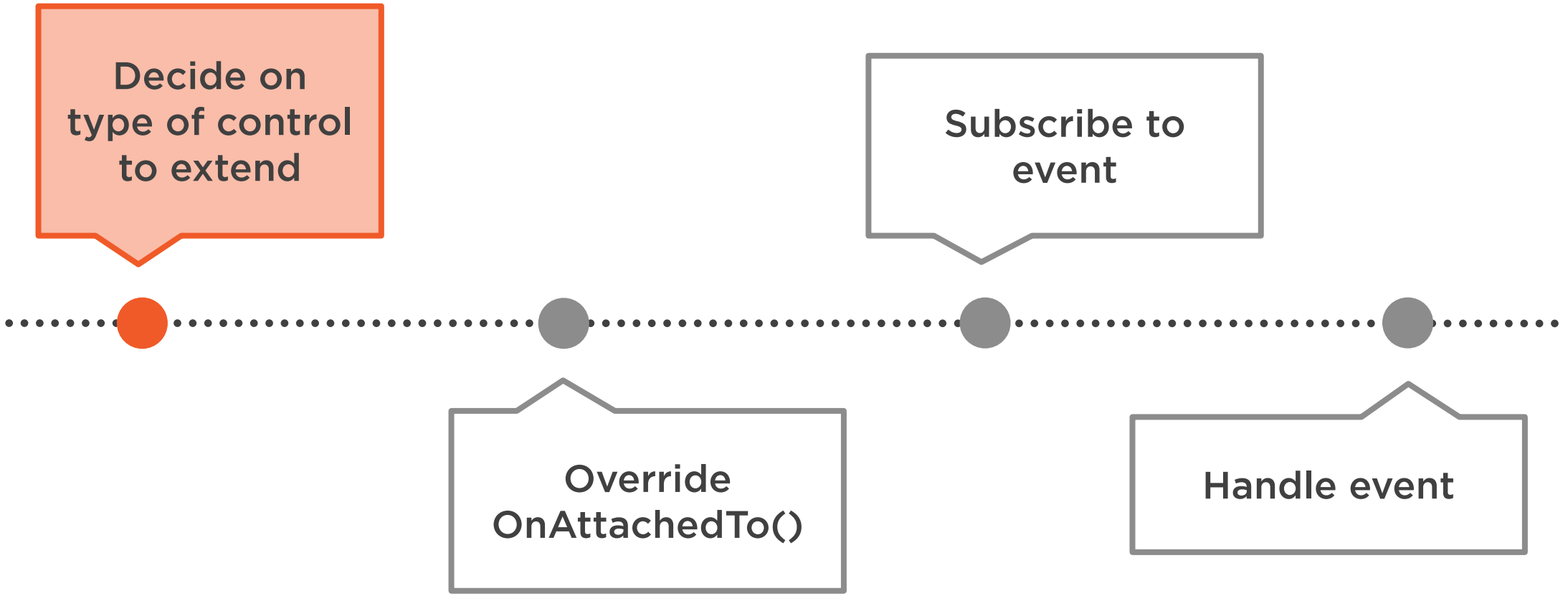
Behaviors

Add functionality to user interface controls without having to subclass them.

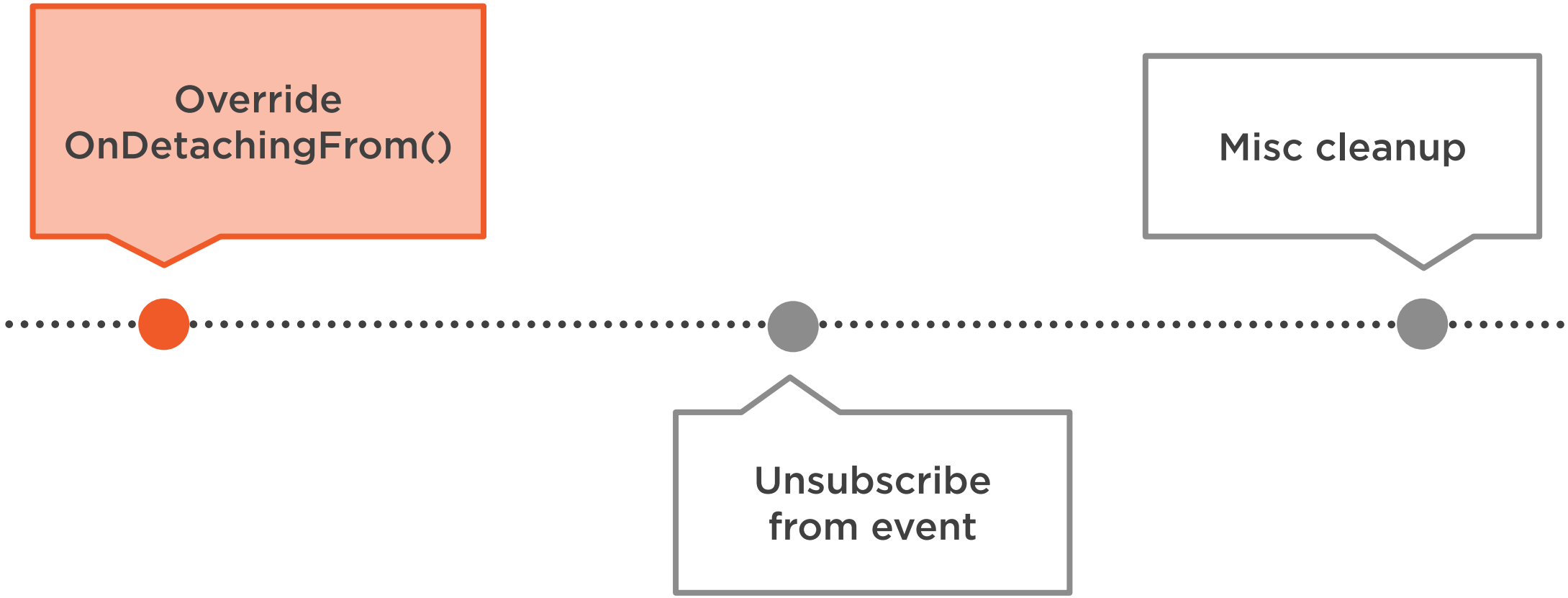
Behavior


Behavior
=
Extend control functionality
+
Add many to single control
+
Reusable

Implementing a Behavior - Setup



Implementing a Behavior – Tear Down



Demo

Adding a custom cell

Creating a Behavior

Consuming a Behavior

Table views organized way to accept data

Custom cells extend built-in functionality

Behaviors extend control functionality

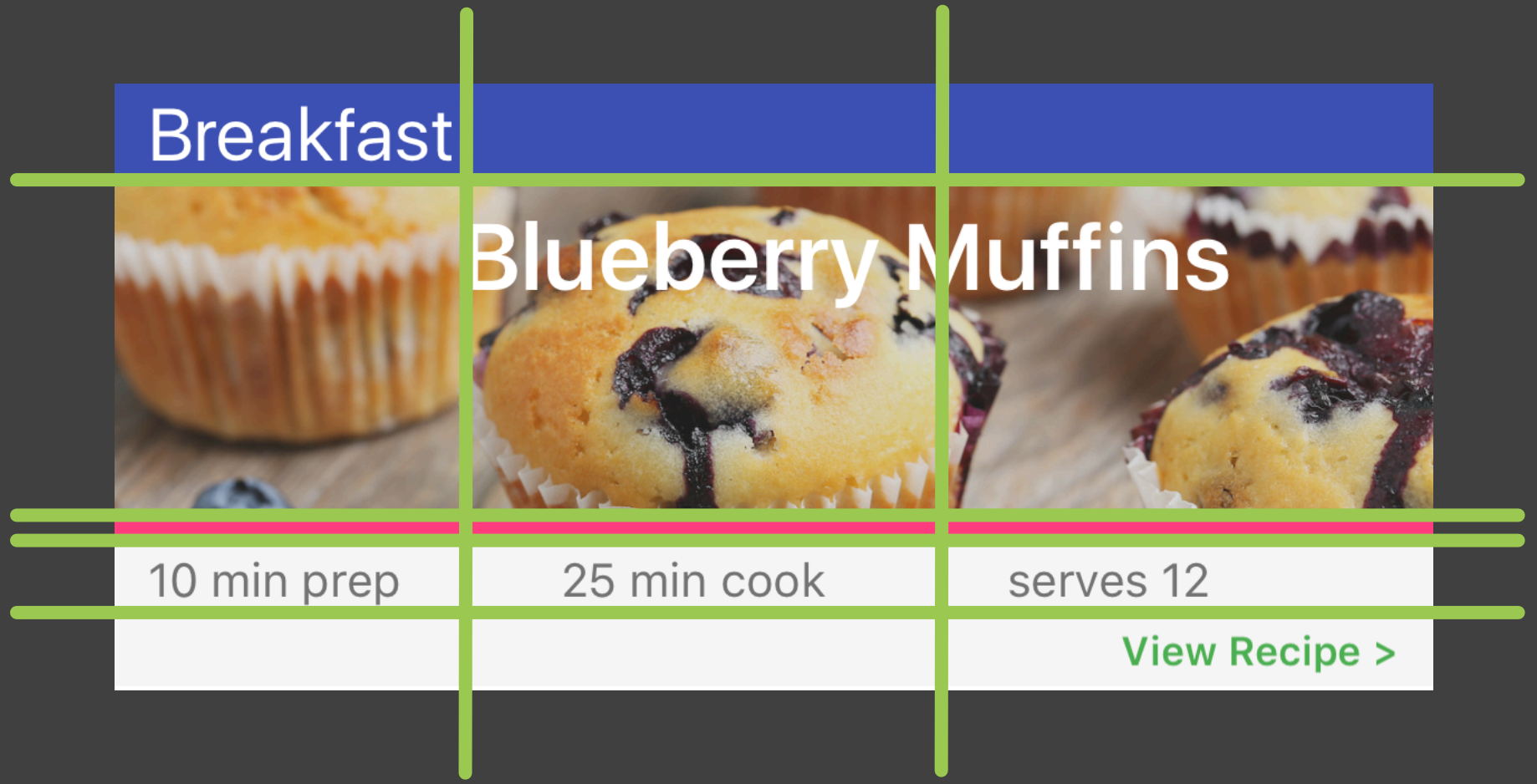
Displaying Data Meaningfully

Displaying Data Topics

Laying out controls with the Grid

Employing DataTemplateSelectors

Grid Layout


Breakfast
Blueberry Muffins
10 min prep
25 min cook
serves 12
View Recipe >

Grid

Easy to build with

Explicitly for 2D

No visible manifestation

```
<Grid.RowDefinitions>
    <RowDefinition Height="30" />
    <RowDefinition Height="Auto" />
    <RowDefinition Height="*" />
    <RowDefinition Height="2*" />
</Grid.RowDefinitions>
```

Declaring Rows

RowDefinitions collection

One RowDefinition object per row

Height property

```
<Grid.ColumnDefinitions>
    <ColumnDefinition Width="Auto" />
    <ColumnDefinition Width="45" />
    <ColumnDefinition Width="*" />
</Grid.ColumnDefinitions>
```

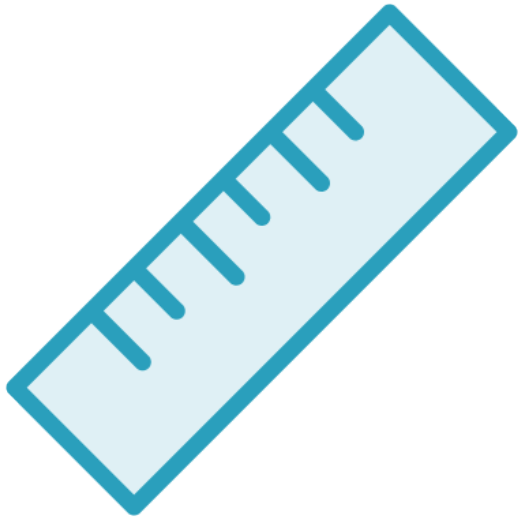
Declaring Columns

ColumnDefinitions collection

One ColumnDefinition object per column

Width property

Row and Column Sizing



GridUnitType Enum

Absolute

- Specify device-specific units

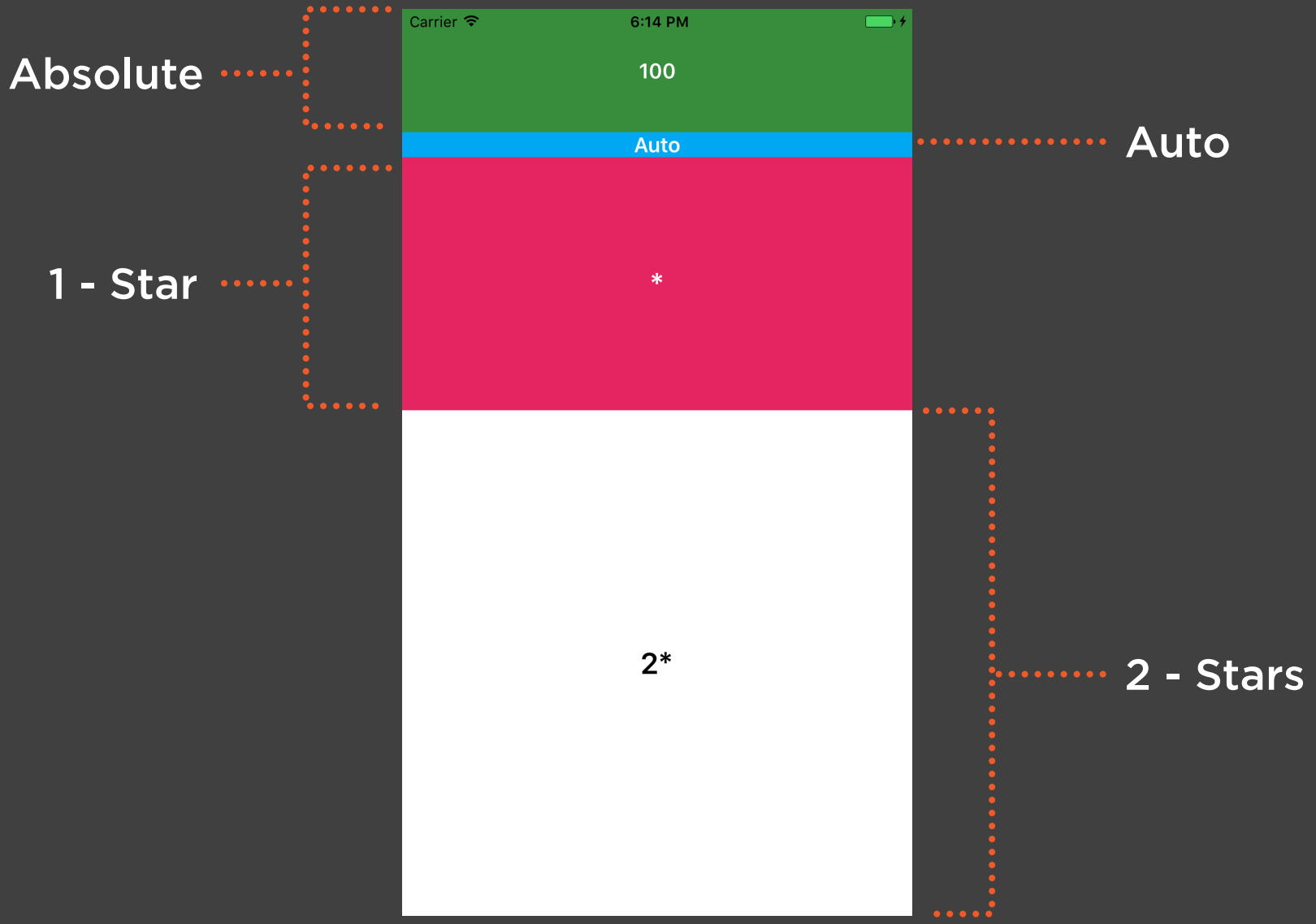
Auto

- Grid chooses size to fit children

Star

- Proportionally allocates area

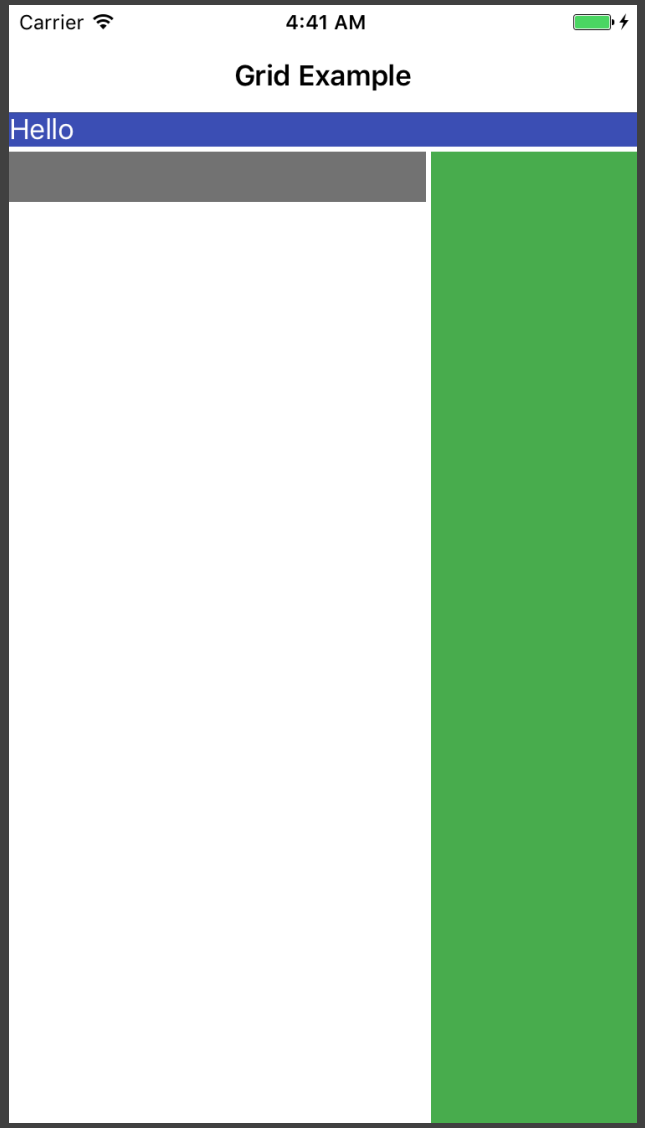
GridUnitType



Populating a Grid

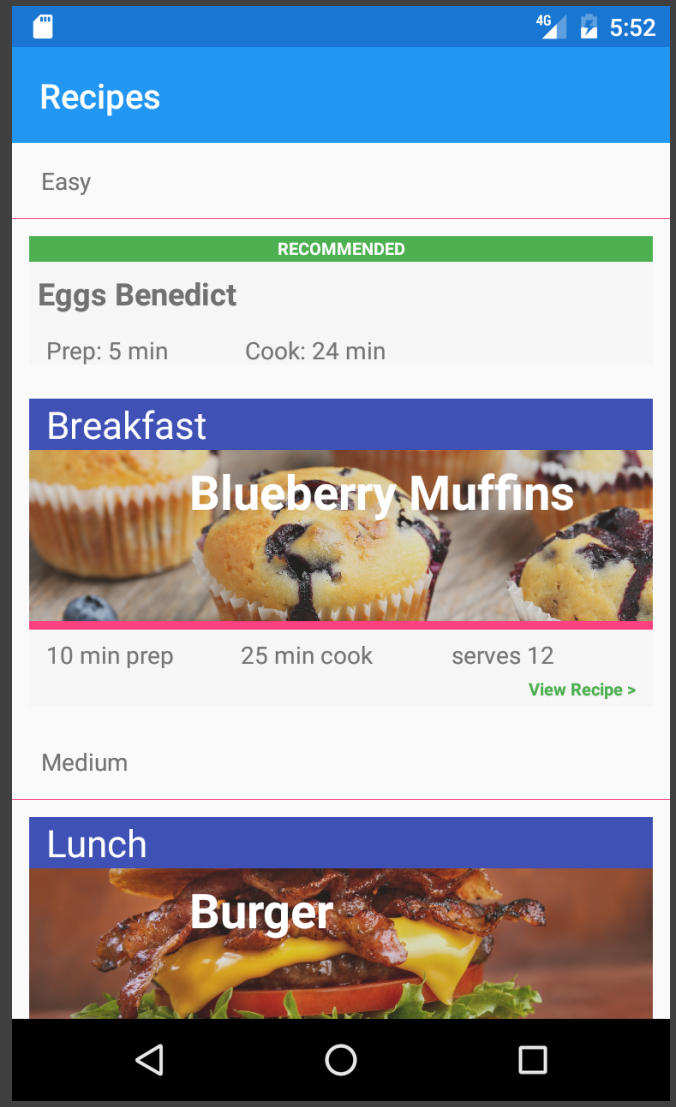
```
<Grid RowSpacing="3" ColumnSpacing="3">
    <Grid.RowDefinitions>
        <RowDefinition Height="Auto" />
        <RowDefinition Height="30" />
        <RowDefinition Height="*" />
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="*" />
        <ColumnDefinition Width="*" />
        <ColumnDefinition Width="*" />
    </Grid.ColumnDefinitions>
    <Label Text="Hello" BackgroundColor="#3F51B5" TextColor="White"
        Grid.Row="0" Grid.Column="0" Grid.ColumnSpan="3" />
    <BoxView Color="#757575"
        Grid.Row="1" Grid.Column="0" Grid.ColumnSpan="2" />
    <BoxView Color="#4CAF50"
        Grid.Row="1" Grid.Column="2" Grid.RowSpan="2" />
</Grid>
```

Grid Spacing



DataTemplateSelector

Recommended Recipe



Recipe

```
public class RecipeDataTemplateSelector : DataTemplateSelector
{
    protected override DataTemplate OnSelectTemplate(
            object item, BindableObject container)
    {

    }
}
```

DataTemplateSelector

OnSelectTemplate

- DataTemplate return contains ViewCell
- item parameter is data being bound

```
<ResourceDictionary>
    <local:RecipeDataTemplateSelector x:Key="recipeTempSel" />
</ResourceDictionary>

...

<ListView
    ItemsSource="{x:Static local:RecipeData.AllRecipesGrouped}"
    ItemTemplate="{StaticResource recipeTempSel}" />
```

Consuming DataTemplateSelector

Define in ResourceDictionary

Set in ItemTemplate

DataTemplateSelector Rules

Return same template for same data

Cannot return DataTemplateSelector classes

Same instance of template should be returned

20 templates max on Android

Demo

Create grid-based ViewCells

Create a DataTemplateSelector

Implement

Simple, meaningful, beautiful

Grid for elegant and easy layout

- 2-dimenstional layout
- GridUnitType sizing

DataTemplateSelector for multiple templates at runtime

- Multiple ListView templates at runtime
- Several rules to follow

Styling the App

Styling Apps

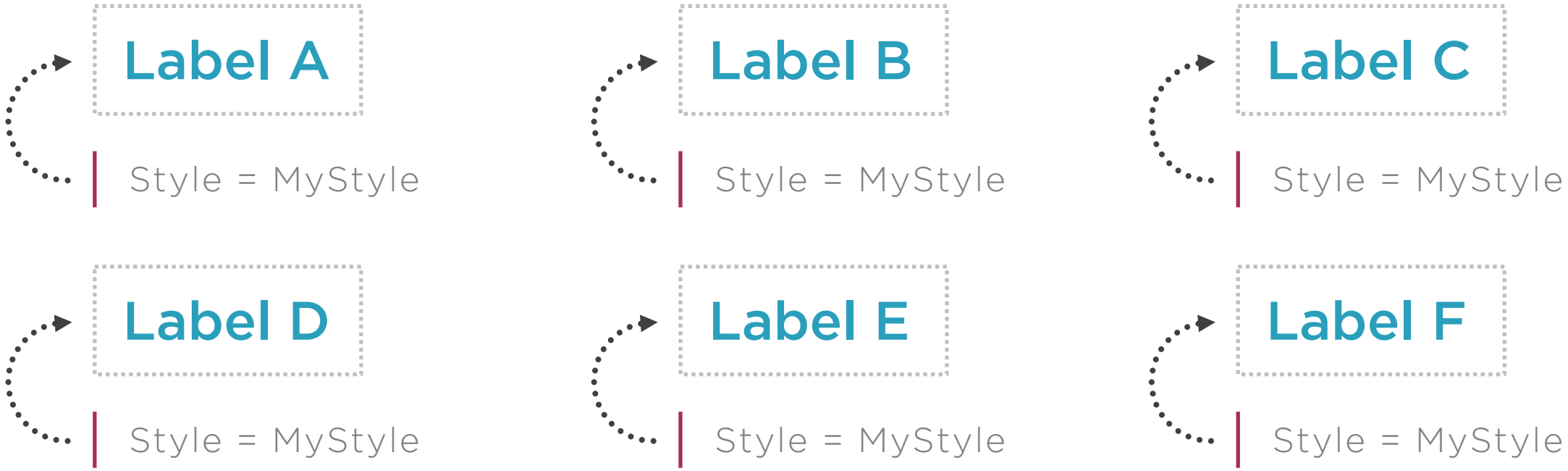
Understanding styles

Swapping at runtime

Understanding Styles

Understanding Styles

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



Xamarin.Forms Styles

Maintain a consistent and customized UI

Collection of property setters

Defined for specific types of controls

Defined in resource dictionary

```
<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}" />
```

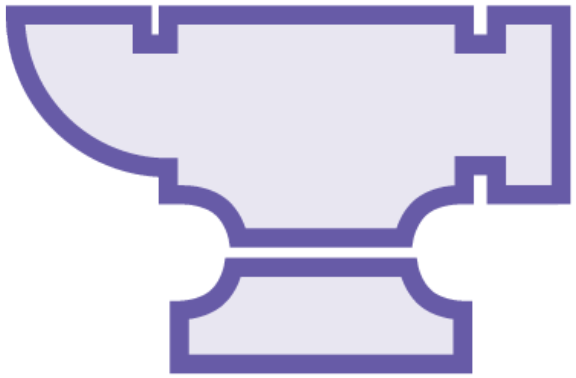
Consuming a Style

Style property

StaticResource

- Reference the key name

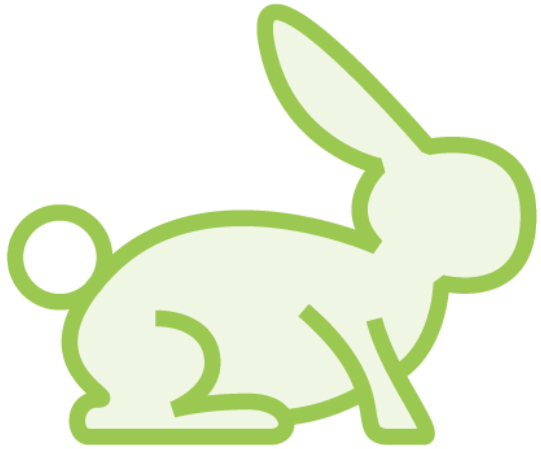
Dynamic Styles



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

Ability for app theming

Demo

Create and consume dynamic styles

Add app theming

Styles

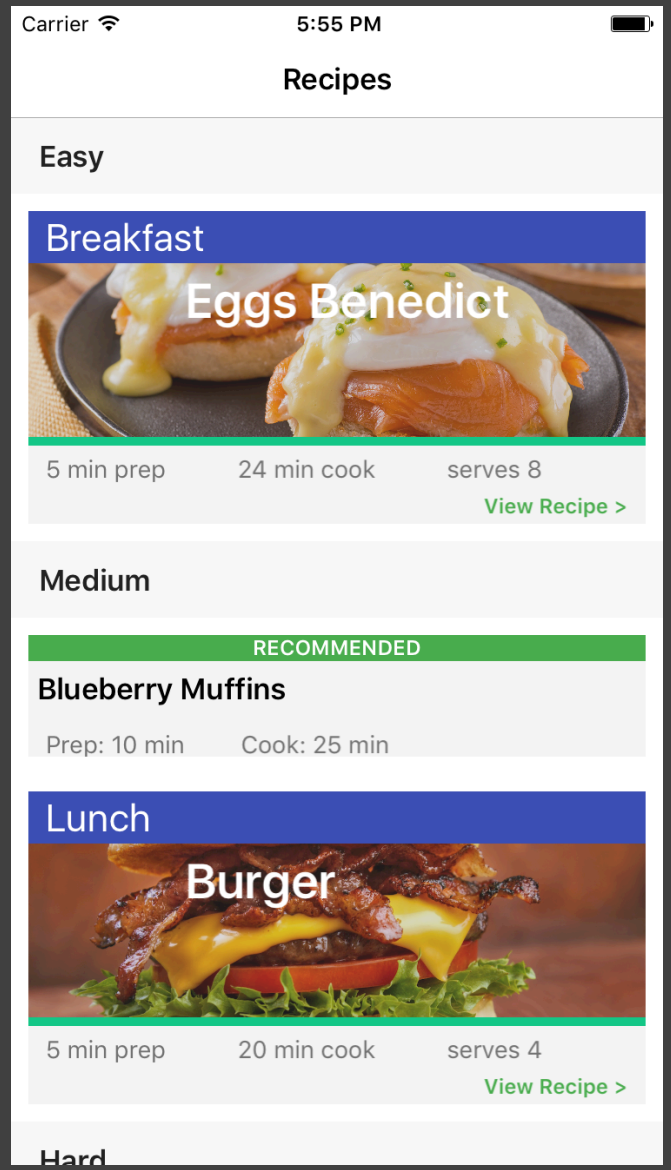
- Consistent, customizable UI
- Collection property setters
- ResourceDictionary

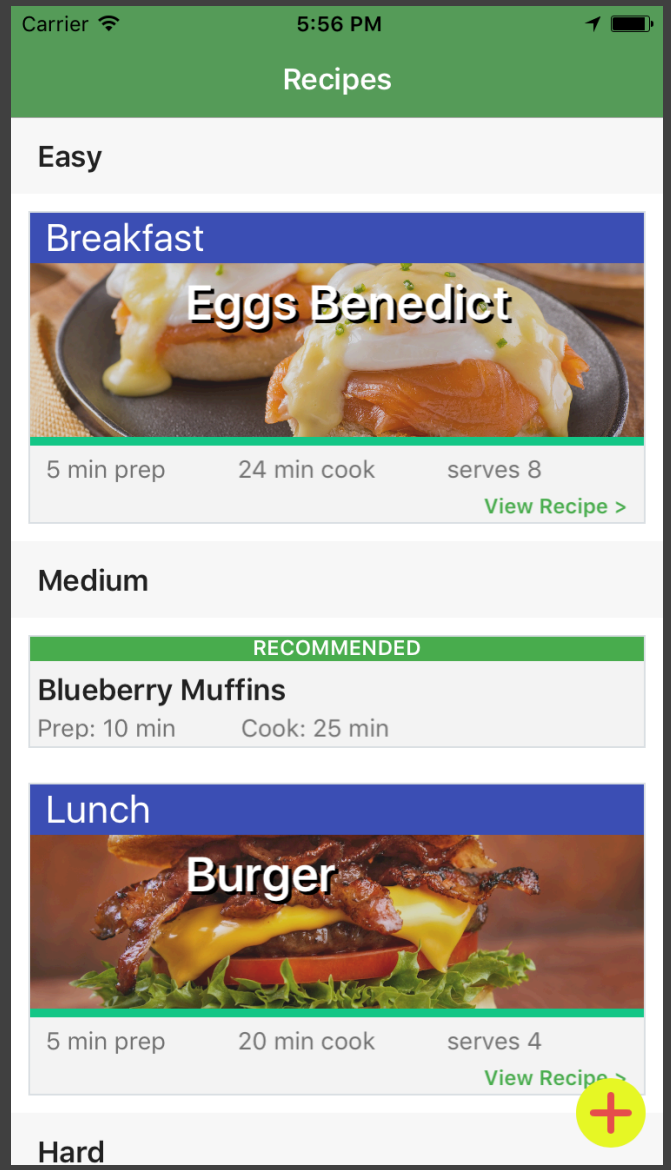
Dynamic

- Style changes at runtime
- DynamicResource
- App theming possible

Native UI

Native UI Enhancements



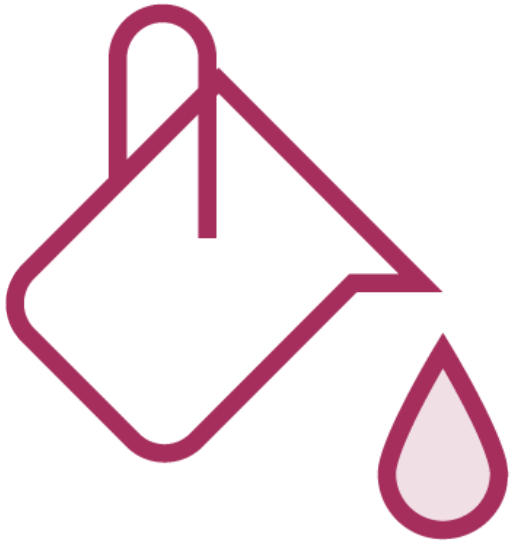


Native UI

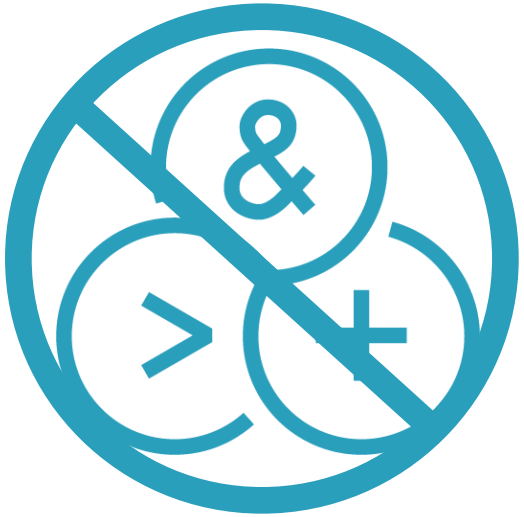
Creating effects

Embedding native views

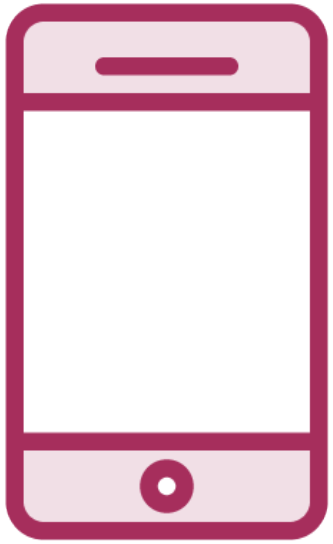
Effects



Small tweaks to the UI



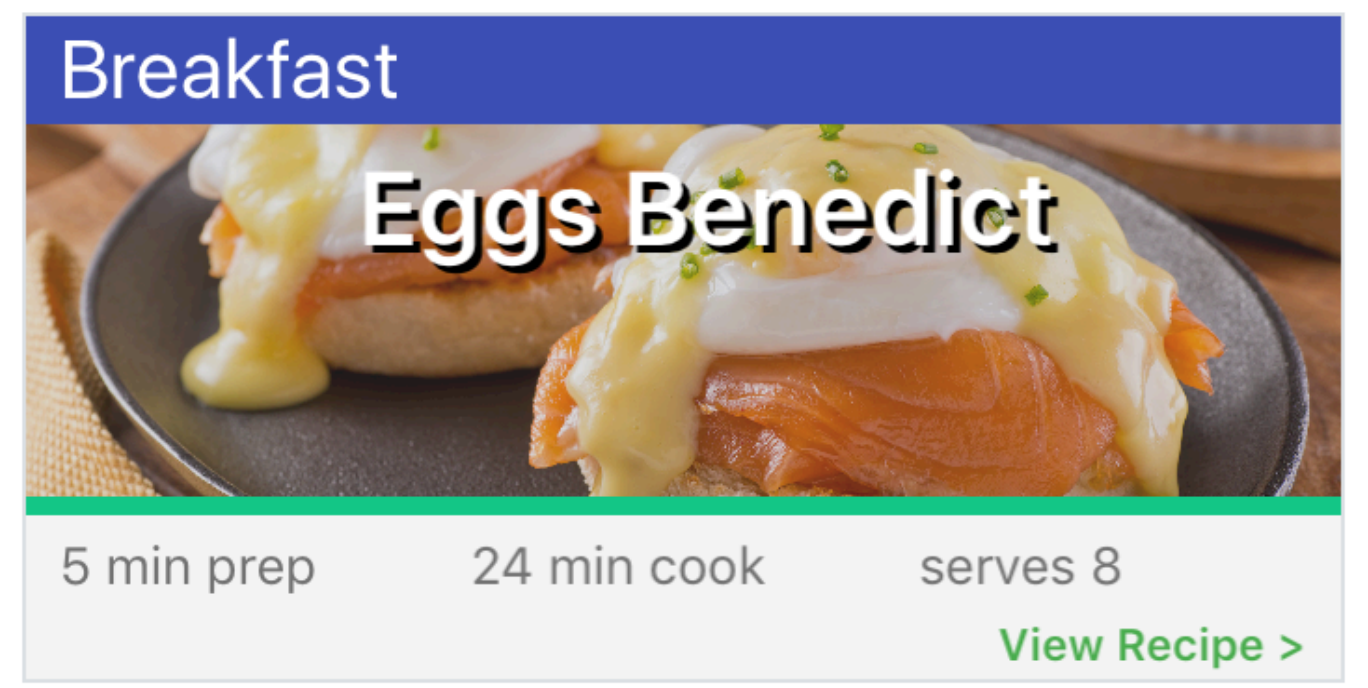
Not meant for behavior changes



Written in platform projects

Consumed in core project

Shadow Effect



Subclass PlatformEffect

Override

- OnAttached
- OnDetached
- OnElementPropertyChanged

Properties

- Element
- Control
- Container

Effect Lifecycle

OnAttached()

Perform any setup
Save pre-existing state
Added to any control

OnDetached()

Perform cleanup
Unsubscribe from events
Revert to initial state

OnElementPropertyChanged()

React to changes in Xam.Forms state
PropertyChangedEventArgs
Indicates property name

Properties of PlatformEffect

Container

Platform control that performs layout of rest

Control

Holds platform equivalent of Forms control

Element

Forms control

Register an Effect

ResolutionGroupName attribute

- One per assembly
- Namespace to avoid collisions

ExportEffect attribute

- Registers a name for the effect

```
public class ShadowEffect : RoutingEffect
{
    public ShadowEffect(): base("CodeMill.ShadowEffect")
    {
    }
}
```

Consuming effects

RoutingEffect

Constructor

- Concatenate ResolutionGroupName + effect name

Effect not needed on all platforms

```
xmlns:local="clr-namespace:Foodie; assembly=Foodie"
...
<Label Text="{Binding RecipeName}"
    Grid.Row="1" Grid.Column="1" Grid.ColumnSpan="2" >
    <Label.Effects>
        <local:ShadowEffect />
    </Label.Effects>
</Label>
```

Consuming Effects

Reference namespace

Add to Effects collection

New instance per control

Bindable Native Views

Pure native view referenced in core project's XAML

Properties in native view bindable to view model

Core project can handle native view events

Invoke constructors and factory methods

```
<ContentPage

...

xmlns:foodieIOS="clr-
namespace:Foodie.iOS;assembly=Foodie.iOS;targetPlatform=iOS"

...

<foodieIOS:FoodieFab AbsoluteLayout.LayoutBounds="1,1,5,5"
    AbsoluteLayout.LayoutFlags="PositionProportional" />
```

Consuming on iOS

Native view already created

Reference native view namespace and assembly in xmlns

New targetPlatform attribute

```
<ContentPage
...

xmlns:foodieDroid="clr-
namespace:Foodie.Droid;assembly=Foodie.Droid;targetPlatform=Android"
xmlns:formsDroid="clr-
namespace:Android.Widget;assembly=Mono.Android;targetPlatform=Android”

...

<foodieDroid:FoodieFab x:Arguments="{x:Static formsDroid:Forms.Context}"
   UseCompatPadding="true" AbsoluteLayout.LayoutBounds="1,1,AutoSize,AutoSize"
   AbsoluteLayout.LayoutFlags="PositionProportional" />
```

Consuming on Android

Reference native view namespace and assembly in xmlns

Reference Mono.Android assembly in xmlns

Constructor requires Android Context class

```
<ios:UITextField Text="{Binding RecipeName}" />
```

Binding a Native View

Reference native property name

Same binding syntax as normal

```
<ios:UITextField Text="{Binding RecipeName, Mode=TwoWay,
    UpdateSourceEventName=Ended}" />
```

Two-way Binding with Events

Specify Mode=TwoWay

May need to specify UpdateSourceEventName

- Native event to update binding

Emedding Native View Gotchas

No XAML compiling

Cannot invoke functions on the native view

Cannot access in code behind

Cannot apply styles

Demo

Create and register an effect

Consume the effect

Consume native view

Effects change small UI properties

- Not meant for behavior changes
- Implement in platform project
- Can have multiple per control

Add native views into XAML

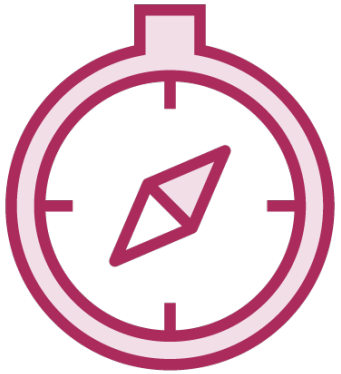
- Properties bindable
- Handle events
- Remember gotchas

Xamarin.Forms Apps

Custom cells and behaviors

Grid layout and data template selectors

Styles and theming with dynamic styles



Platform tweaks and effects

Learn More!

Pluralsight: Moving Beyond the Basics with Xamarin.Forms

https://www.pluralsight.com/courses/xamarin-forms-moving-beyond-basics

Moving Beyond the Basics with Xamarin.Forms

Matthew Soucoup
PRINCIPAL

@codemillmatt codemilltech.com