Lecture 3 – Xamarin.Forms Layouts

Mr. Yousif Garabet Arshak
Computer Science Department
University of Zakho
yousif.arshak@uoz.edu.krd

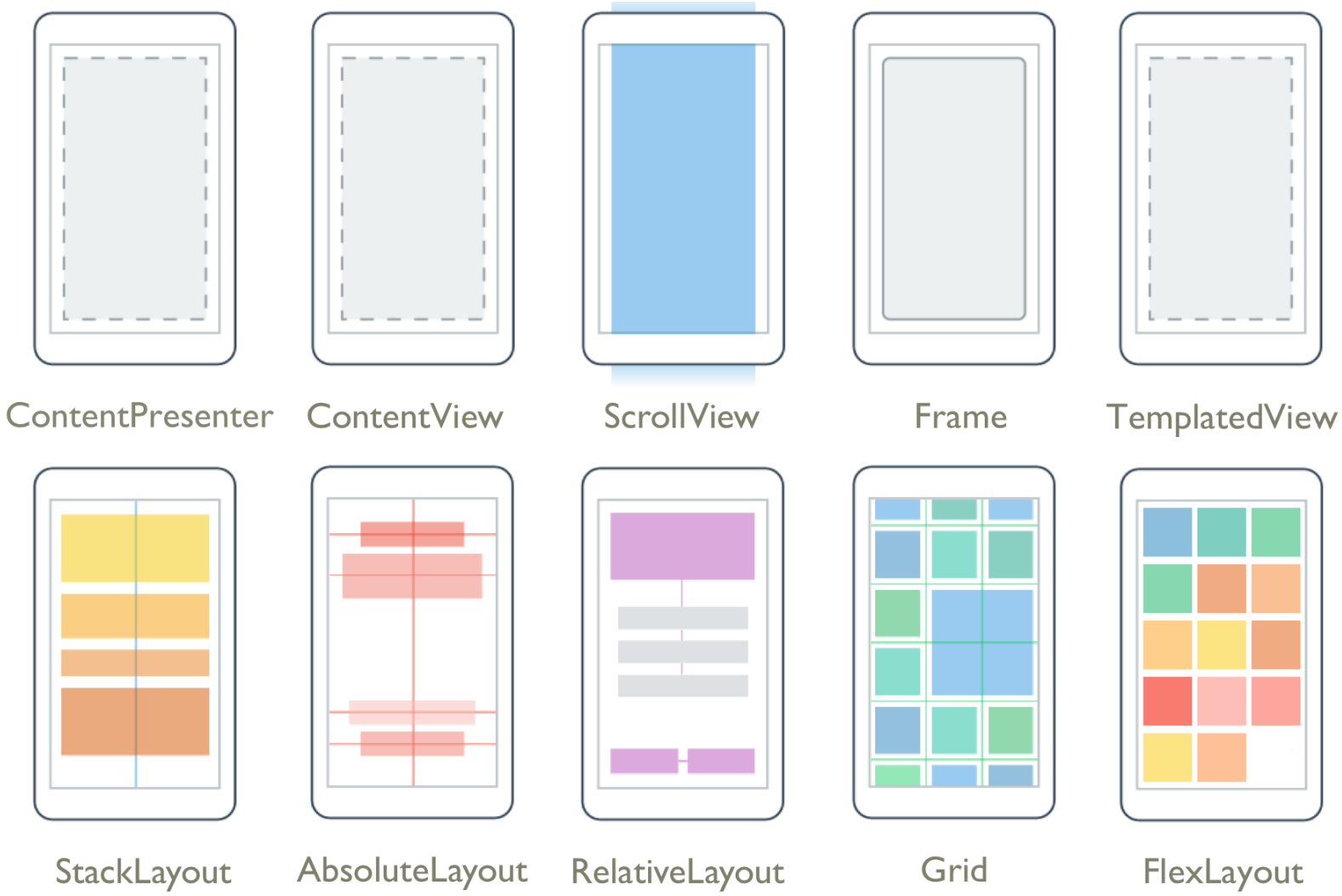
Outlines

- Xamarin.Forms Layouts
- Layouts with Single Content
    1. ContentView
    2. Frame
    3. ScrollView
    4. TemplatedView
    5. ContentPresenter
- Layouts with Multiple Children
    1. StackLayout
    2. Grid
    3. AbsoluteLayout
    4. RelativeLayout
    5. FlexLayout

Xamarin.Forms Layouts

- *Xamarin.Forms Layouts are used to compose user-interface controls into visual structures.*
- The Layout and Layout<T> classes in Xamarin.Forms are specialized subtypes of views that act as containers for views and other layouts.
- The Layout class itself derives from View.
- A Layout derivative typically contains logic to set the position and size of child elements in Xamarin.Forms applications.

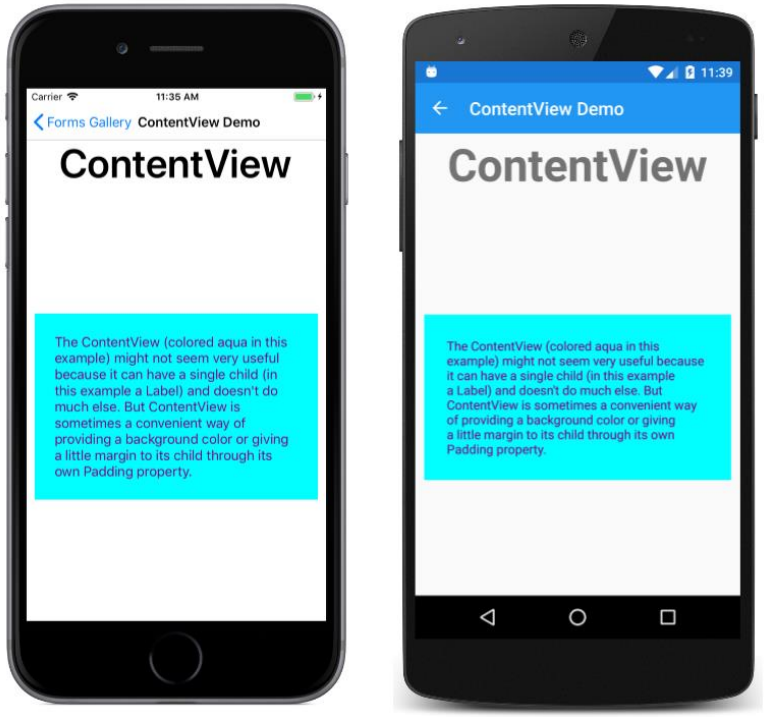
Xamarin.Forms Layouts



Layouts with Single Content

These classes derive from Layout, which defines Padding and IsClippedToBounds properties:

1. ContentView:
   ContentView contains a single child that is set with the Content property.
   The Content property can be set to any View derivative, including other Layout derivatives.
   ContentView is mostly used as a structural element and serves as a base class to Frame.



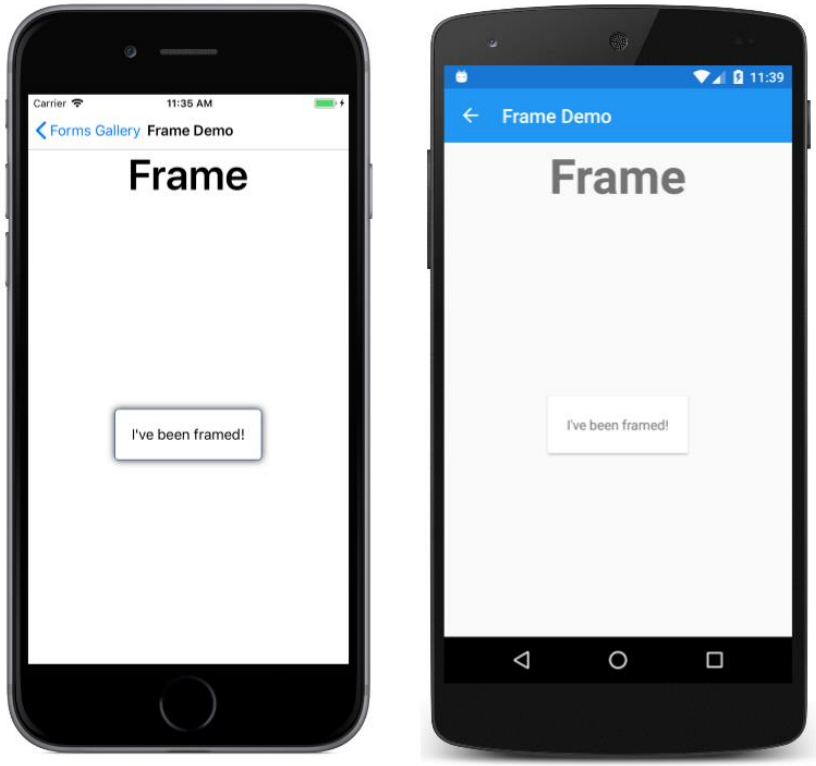
Xaml Code

Note: You can download all the XAML code in the this link:

[Xamarin.Forms Layouts - Xamarin | Microsoft Docs]

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
        xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
        x:Class="FormsGallery.XamlExamples.ContentViewDemoPage"
        Title="ContentView Demo">
  <StackLayout>
    <Label Text="ContentView"
          FontSize="50"
          FontAttributes="Bold"
          HorizontalOptions="Center" />
    <ContentView BackgroundColor="Aqua"
              Margin="10"
              Padding="25"
              HorizontalOptions="Fill"
              VerticalOptions="CenterAndExpand">
      <Label Text= “some text here”
TextColor="Purple" />
    </ContentView>
  </StackLayout>
</ContentPage>
```

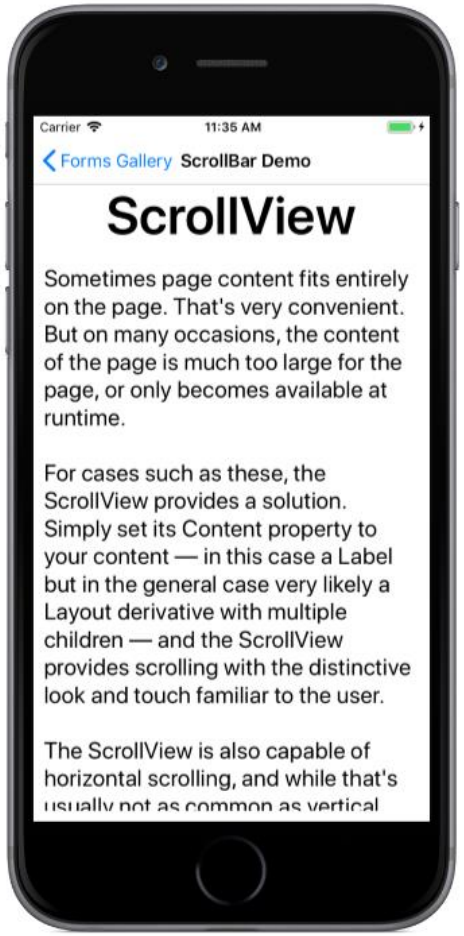
2. Frame: The Frame class derives from ContentView and displays a border, or frame, around its child. The Frame class has a default Padding value of 20, and also defines BorderColor, CornerRadius, and HasShadow properties.



Xaml Code

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
        xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
        x:Class="FormsGallery.XamlExamples.FrameDemoPage"
        Title="Frame Demo">
   <StackLayout>
      <Label Text="Frame"
           FontSize="50"
           FontAttributes="Bold"
           HorizontalOptions="Center" />
      <Frame OutlineColor="Accent"
           HasShadow="True"
           HorizontalOptions="Center"
           VerticalOptions="CenterAndExpand">
        <Label Text="I've been framed!" />
      </Frame>
   </StackLayout>
</ContentPage>
```

3. ScrollView: is capable of scrolling its contents. Set the Content property to a view or layout too large to fit on the screen. (The content of a ScrollView is very often a StackLayout.) Set the Orientation property to indicate if scrolling should be vertical, horizontal, or both.



Xaml Code

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
        xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
        x:Class="FormsGallery.XamlExamples.ScrollViewDemoPage"
        Title="ScrollView Demo">
   <StackLayout>
      <Label Text="ScrollView"
          FontSize="50"
          FontAttributes="Bold"
          HorizontalOptions="Center" />
      <ScrollView VerticalOptions="FillAndExpand"
             Margin="10">
      <Label Text=
"Sometimes page content.... “
          FontSize="Large" />
      </ScrollView>
   </StackLayout>
</ContentPage>
```

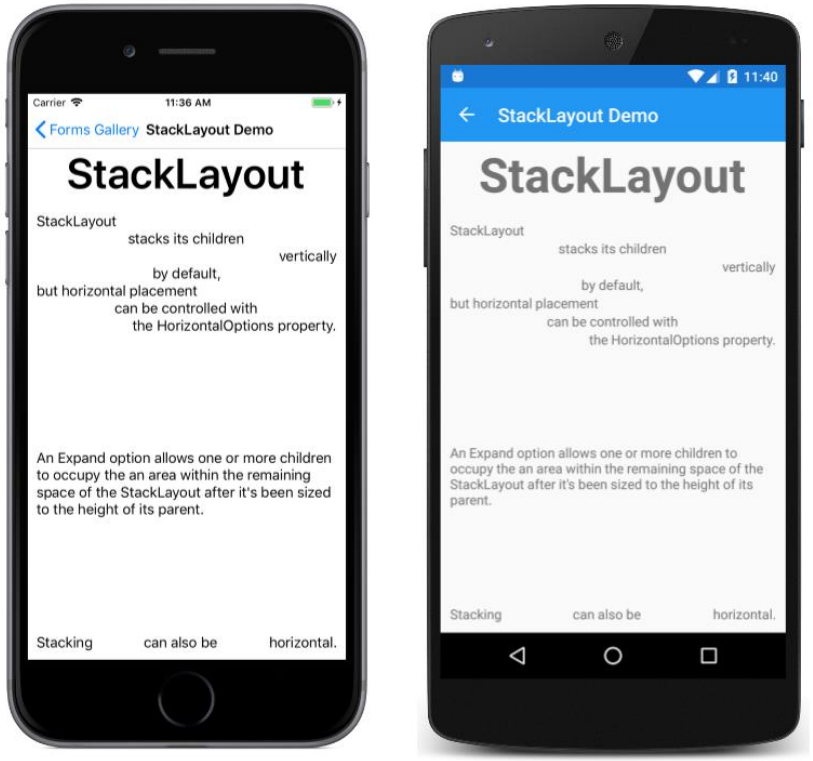
4. TemplatedView: TemplatedView displays content with a control template, and is the base class for ContentView.

5. ContentPresenter: is a layout manager for templated views, used within a ControlTemplate to mark where the content that is to be presented appears.

Layouts with Multiple Children

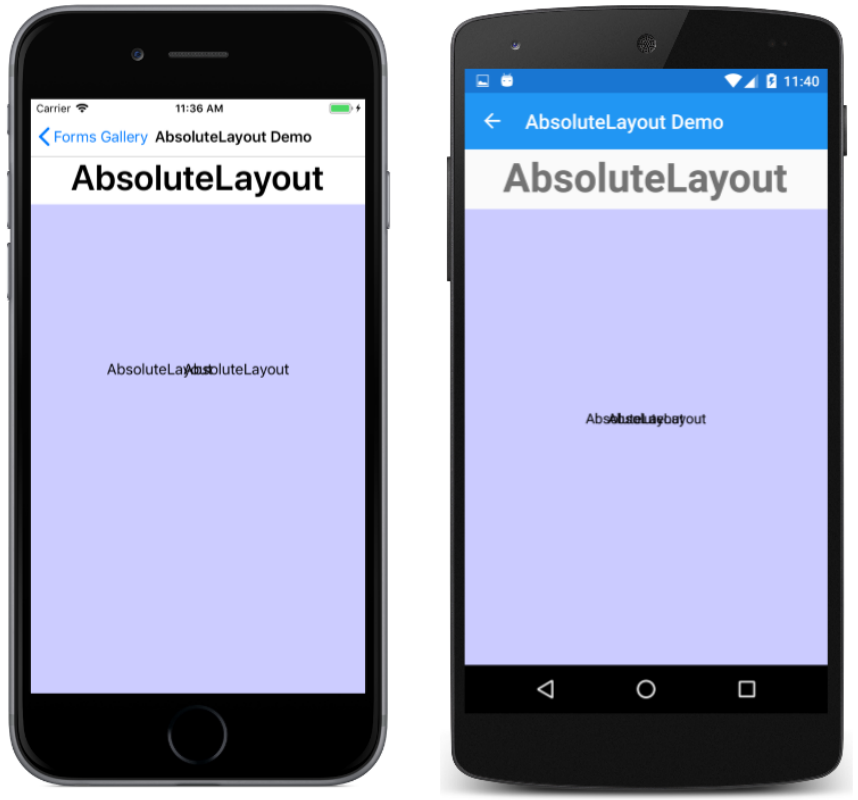
These classes derive from Layout<View>:

1. StackLayout: StackLayout positions child elements in a stack either horizontally or vertically based on the Orientation property. The Spacing property governs the spacing between the children, and has a default value of 6.

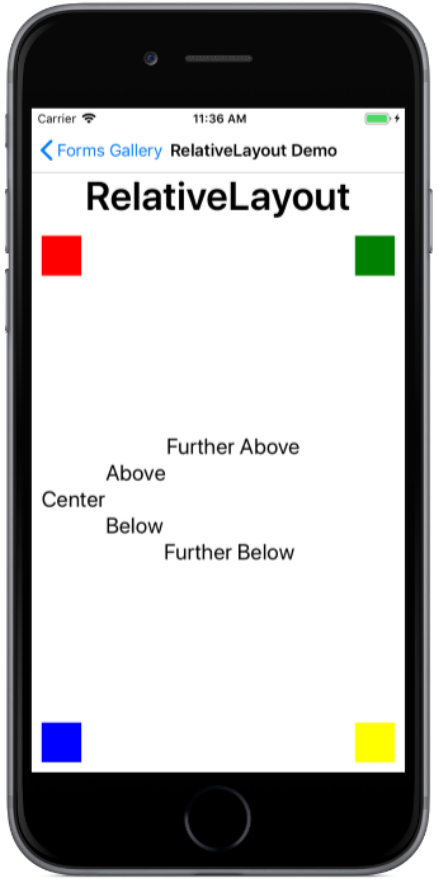


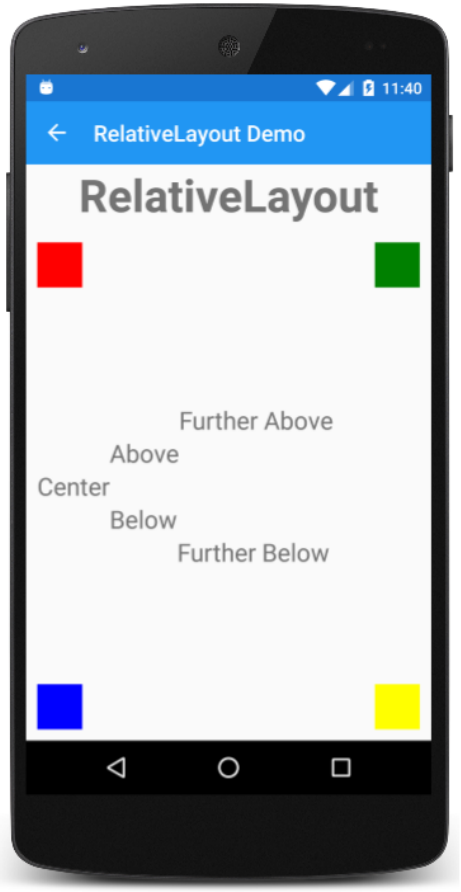
2. Grid: Grid positions its child elements in a grid of rows and columns. A child's position is indicated using the attached properties Row, Column, RowSpan, and ColumnSpan.

3. AbsoluteLayout:
   AbsoluteLayout positions child elements at specific locations relative to its parent. A child's position is indicated using the attached properties LayoutBounds and LayoutFlags. An AbsoluteLayout is useful for animating the positions of views.

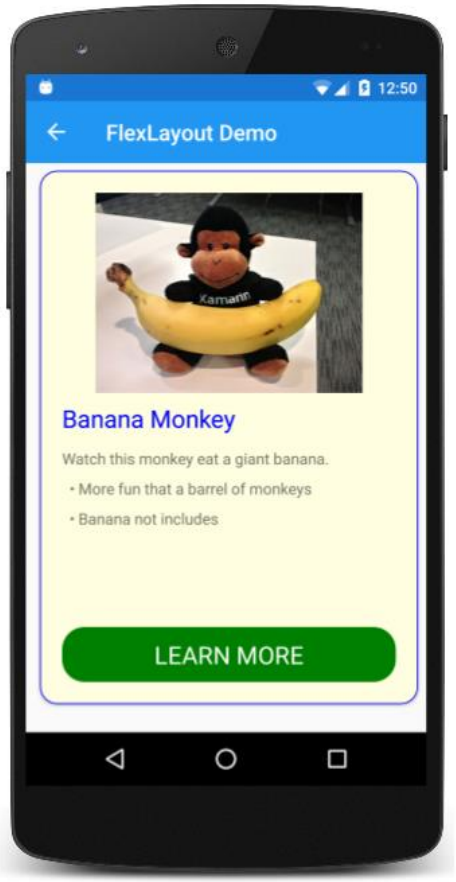


4. RelativeLayout: RelativeLayout positions child elements relative to the RelativeLayout itself or to their siblings. A child's position is indicated using the attached properties that are set to objects of type Constraint and BoundsConstraint.





5. FlexLayout: FlexLayout is based on the CSS Flexible Box Layout Module, commonly known as flex layout or flex-box. FlexLayout defines six bindable properties and five attached bindable properties that allow children to be stacked or wrapped with many alignment and orientation options.



Any Questions?