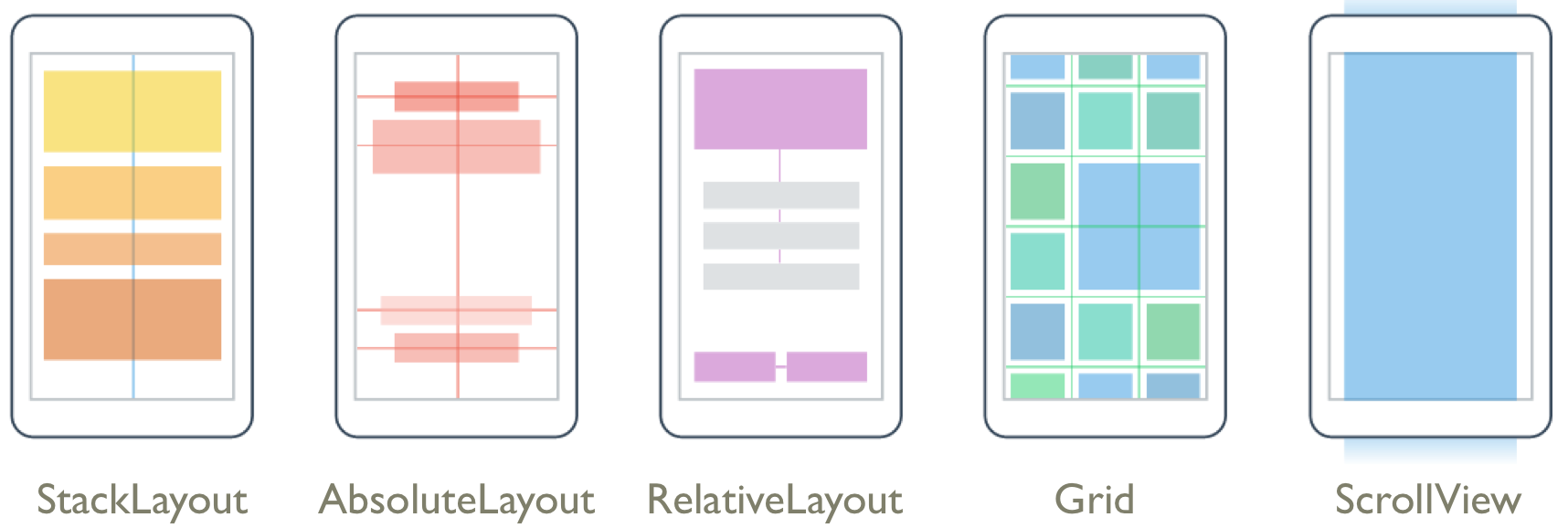
# Layouts in Xamarin

The following is mainly paraphrased from the Xamarin Developer documentation site. Xamarin forms uses layouts to organize content on the screen. Controls arranged in the layouts will be translated to native controls when compiled. For instance, a label will become an Android Text View.



* **StackLayout** – used to arrange views linearly, either horizontally or vertically. Views in a StackLayout can be aligned to the center, left or right of the layout.
* **AbsoluteLayout** – used to arrange views by setting coordinates & size in terms of absolute values or ratios. AbsoluteLayout can be used to layer views as well as anchor them to the left, right or center.
* **RelativeLayout** – used to arrange views by setting constraints relative to their parent's dimensions & position.
* **Grid** – used to arrange views in a grid. Rows and columns can be specified in terms of absolute values or ratios.
* **ScrollView** – used to provide scrolling when a view can't fit entirely within the bounds of the screen.

# LayoutOptions Structure

Every Xamarin.Forms view has HorizontalOptions and VerticalOptions properties, of type LayoutOptions. There are two layout preferences:

* Alignment – preferred alignment, which determines its position and size within its parent layout.
* Expansion – only in StackLayout, and indicates if the view should use extra space, if it's available.

Each view can have a HorizontalOptions and VerticalOptions property setting which can be one of Start, Center, End, Fill. There are also StartAndExpand, CenterAndExpand, EndAndExpand, and FillAndExpand.

* For horizontal alignment, **Start** positions the View on the left hand side of the parent layout, and for vertical alignment, it positions the View at the top of the parent layout.
* For horizontal and vertical alignment, **Center** horizontally or vertically centers the View.
* For horizontal alignment, **End** positions the View on the right hand side of the parent layout, and for vertical alignment, it positions the View at the bottom of the parent layout.
* For horizontal alignment, **Fill** ensures that the View fills the width of the parent layout, and for vertical alignment, it ensures that the View fills the height of the parent layout.

The **StartAndExpand**, **CenterAndExpand**, **EndAndExpand**, and **FillAndExpand** values are used to define the alignment preference, and whether the view will occupy more space if available within the parent StackLayout.

Example setting a label to Center in XAML:

<Label Text="Center" BackgroundColor="Gray" HorizontalOptions="Center" />

# StackLayout

Stacks are the simplest and most common kind of layout. Combining stacks together can make creating forms fairly easy both in XAML and in code. Set the orientation of the stack (vertical is default) and the layout options:

<StackLayout HorizontalOptions="CenterAndExpand" VerticalOptions="CenterAndExpand" Orientation=”Horizontal”>

…

</StackLayout>

You can also add a margin property into the header in the form Margin=”#,#,#,#” to define margins around the outside of the stack layout.

Further documentation:

<https://developer.xamarin.com/guides/xamarin-forms/user-interface/layouts/stack-layout/>

# RelativeLayout

Used to position views relative to other views or the layout. Documentation at:

<https://developer.xamarin.com/guides/xamarin-forms/user-interface/layouts/relative-layout/>

# AbsoluteLayout

Not explored in this class, documentation at:

<https://developer.xamarin.com/guides/xamarin-forms/user-interface/layouts/absolute-layout/>

# Grid

Used for a grid of objects, different from a table. TableView is probably better for laying out a form or table. Grid is used specifically if you want a grid, such as a grid of calculator buttons. To use a Grid, define the rows, the columns, and then the child views that will go into the grid. Using a \* for the size of a row/column will have it fill the available space.

Further documentation:

<https://developer.xamarin.com/guides/xamarin-forms/user-interface/layouts/grid/>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="\*" />

<RowDefinition Height="\*" />

…

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*" />

…

</Grid.ColumnDefinitions>

<Button Text="1" Grid.Row="0" Grid.Column="0" />

<Button Text="2" Grid.Row="0" Grid.Column="1" />

….

</Grid>