

Jacob Alzén

# Exploring the standard layouts

A guide to the standard layouts and their common usage.



# About me

- Open source philanthropist, privacy advocate and amateur photographer.
- First year Computer Science student.
- Gophers Slack: @jacalz
- GitHub: <https://github.com/jacalz>






# Layouts and containers

- The layout package handles position and size of objects.
  - Works from the window down; allocating space as it is needed.
- The container groups together objects and a layout (optionally without a layout).
  - More than just a layout shorthand.

```
type Layout interface {  
    // Layout will manipulate the listed CanvasObjects Size and Position  
    // to fit within the specified size.  
    Layout([]CanvasObject, Size)  
    // MinSize calculates the smallest size that will fit the listed  
    // CanvasObjects using this Layout algorithm.  
    MinSize(objects []CanvasObject) Size  
}
```



```
package main  
  
import (  
    "fyne.io/fyne/v2/layout"  
    "fyne.io/fyne/v2/container"  
)  
  
var box1    = container.New(layout.NewVBoxLayout(), ...)  
var box2    = container.NewVBox(...)   
var manual  = container.NewWithoutLayout(...)
```



# Horizontal Box (HBox)

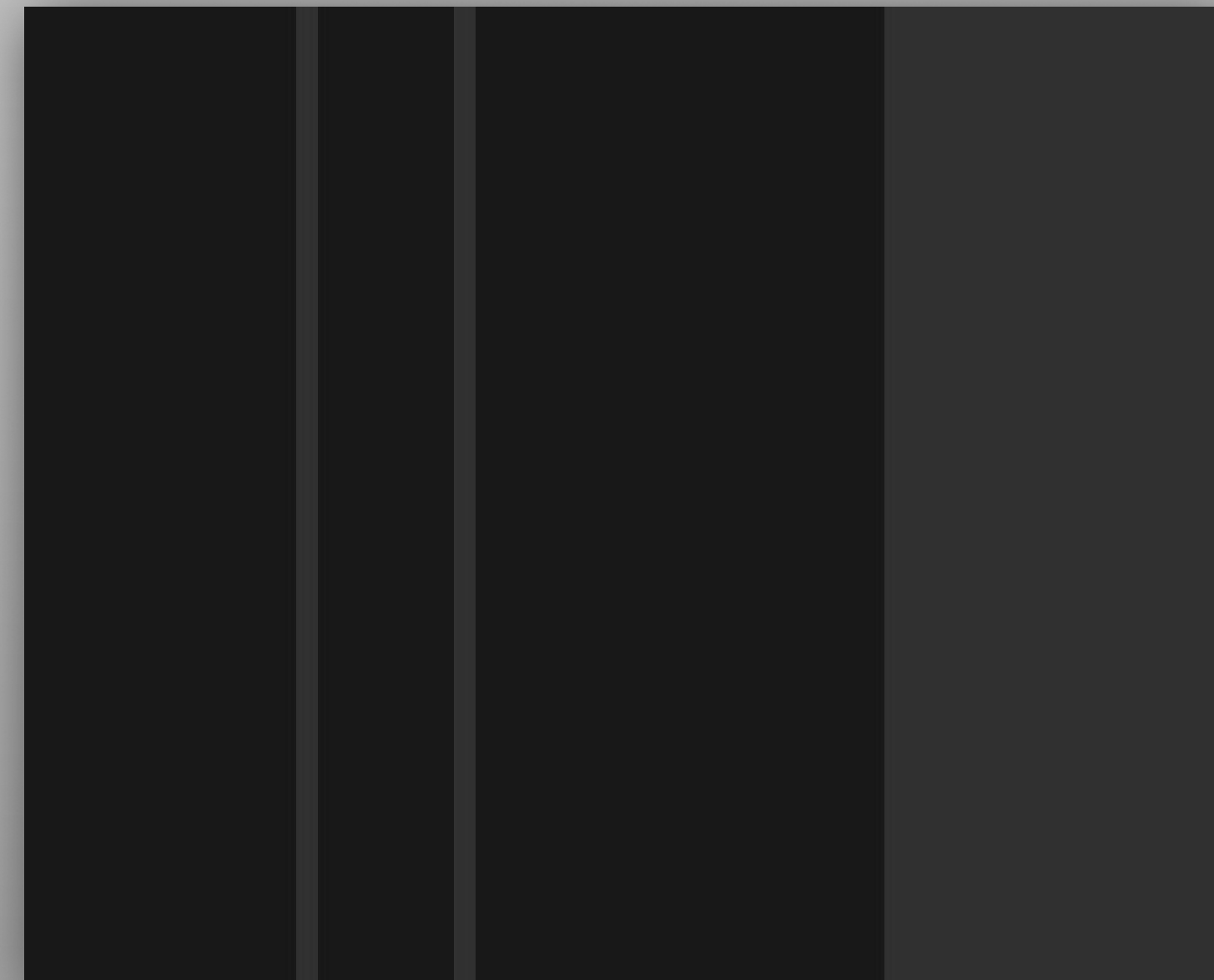
```
layout.NewHBoxLayout( )
```

- Add objects to the right of each other.
- Objects are rendered at their minimum width (MinSize).

Button 8

Button 9

Button 10

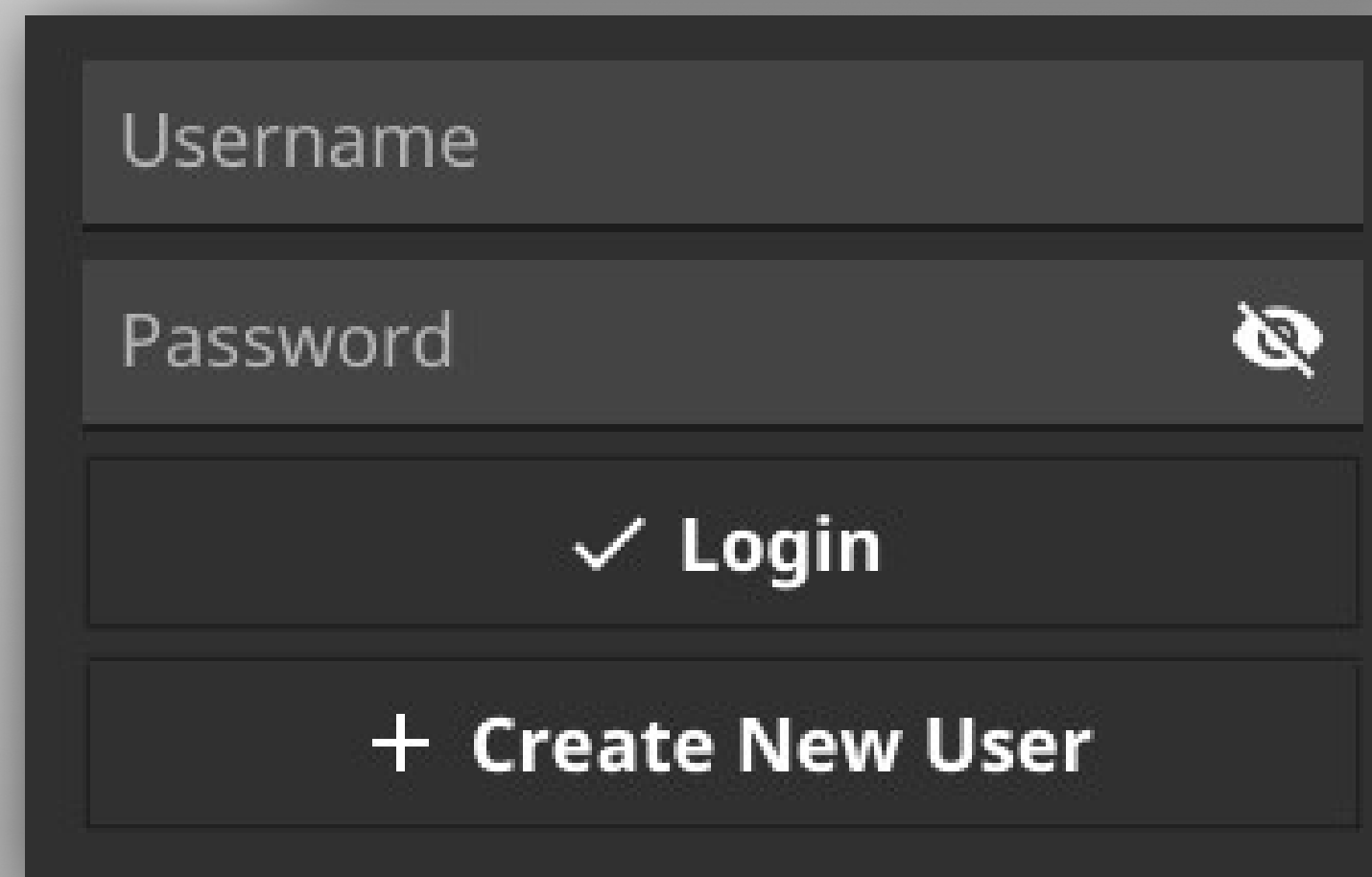




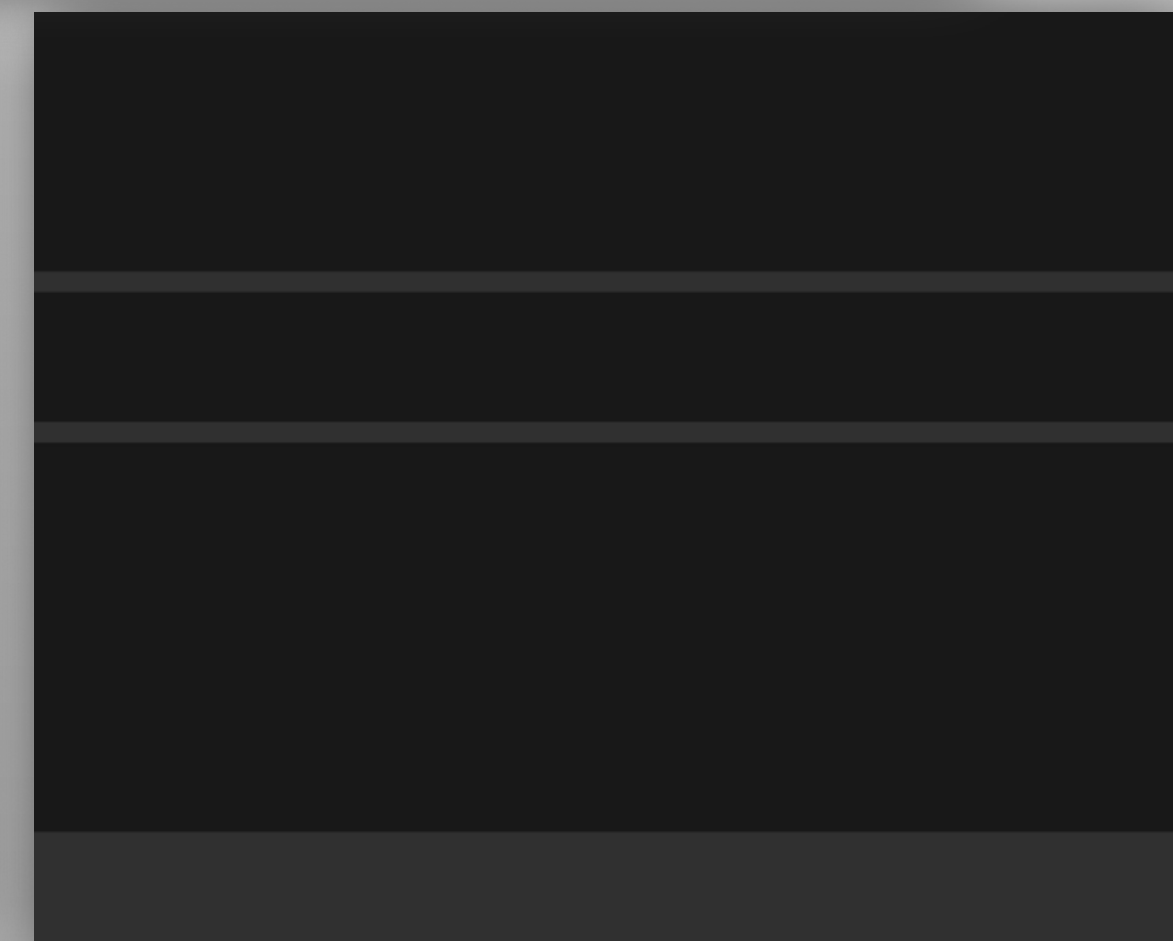
# Vertical Box (VBox)

```
layout.NewVBoxLayout( )
```

- Add objects below each other.
- Objects are rendered at their minimum height (MinSize).



A vertical box layout containing a login form. The form consists of four stacked elements: a text input field labeled "Username", a text input field labeled "Password" with a toggle icon on the right, a button labeled "✓ Login", and a button labeled "+ Create New User".



A vertical box layout containing a list of five empty rectangular items, stacked vertically.



# Grid

- Positions the objects in a grid pattern with a set amount of rows or columns.
- Elements will get scaled so that all objects are the same size while keeping the amount of rows and columns the same.

```
layout.NewGridLayout(cols int)  
layout.NewGridLayoutWithColumns(cols int)  
layout.NewGridLayoutWithRows(rows int)
```

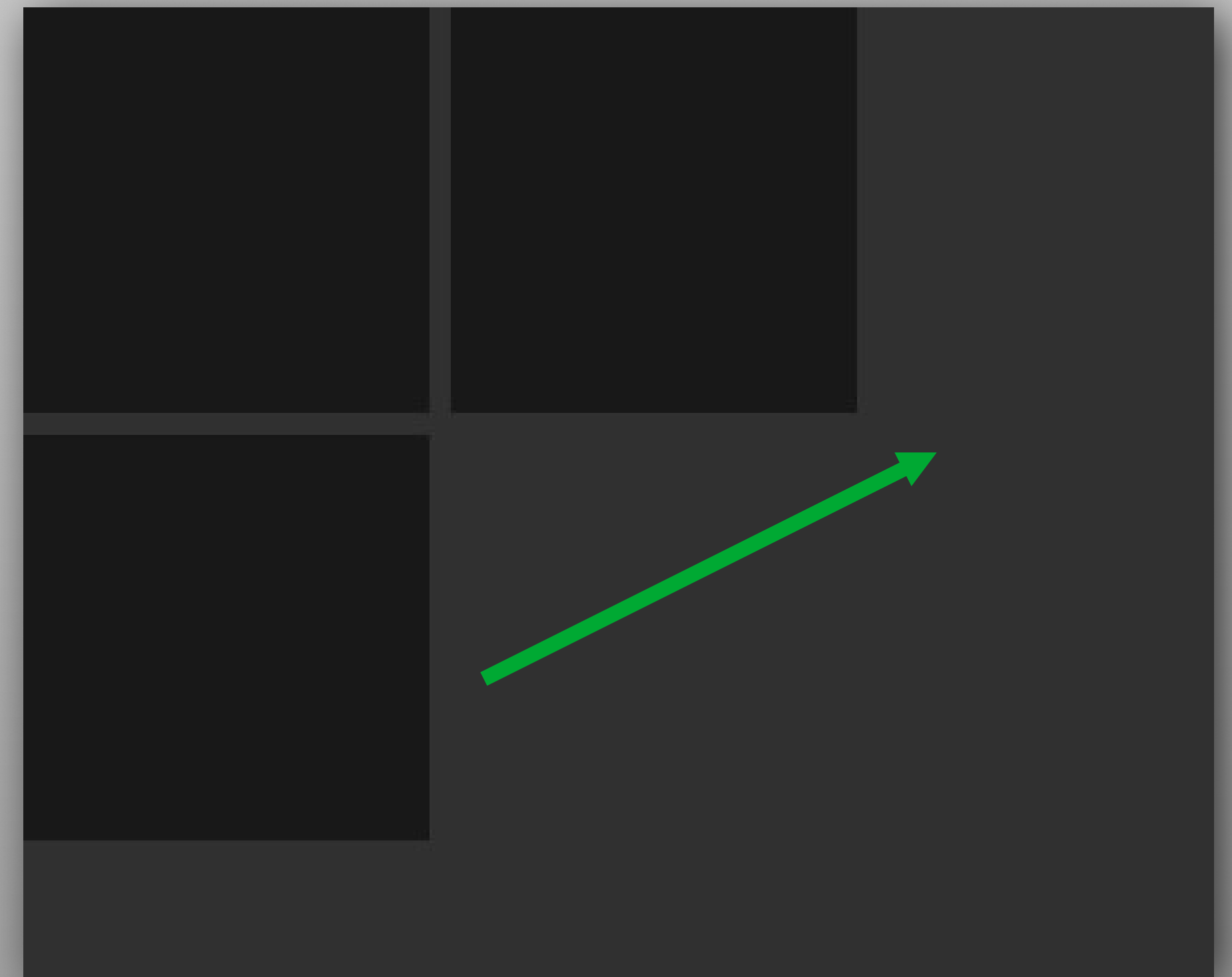




# GridWrap

```
layout.NewGridWrapLayout(size fyne.Size)
```

- Like a grid, but the grid object size is fixed. Items that don't fit on the current row will drop down to populate the next row.
- Most commonly used in file managers for displaying files.

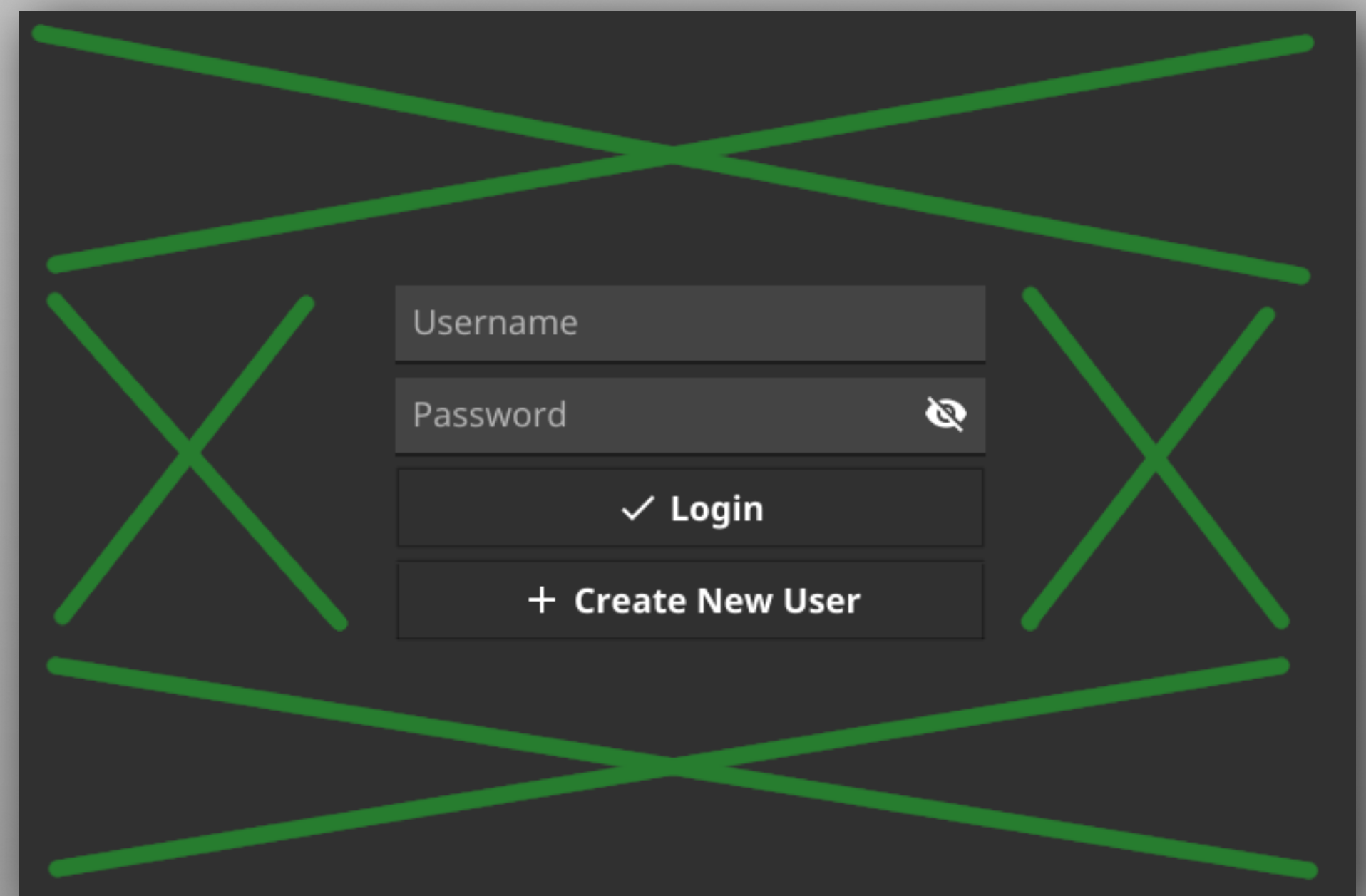




# Spacer

- Not directly a standard layout.
- Used for spacing out objects inside containers.
  - Commonly used inside Grid or VBox containers.

```
layout.NewSpacer( )
```





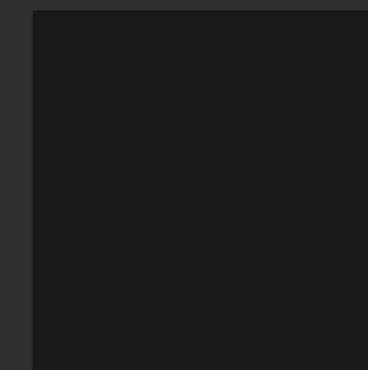
# Center

- Positions the item in the center of the available space.
- Objects are rendered at their minimum size (MinSize).

```
layout.NewCenterLayout( )
```

A diagram illustrating the CenterLayout. It consists of a large dark gray rectangle representing the container. Inside this rectangle, centered both horizontally and vertically, is a smaller, lighter gray rectangle. The text "CenterLayout" is written in white inside the lighter gray rectangle.

CenterLayout






# Form

```
layout.NewFormLayout( )
```

- Serves as the base for the Form widget.
  - Every other item will be positioned in the left column at the minimum size (MinSize).
- Most commonly used with labels in the left fields and entries in the right fields.

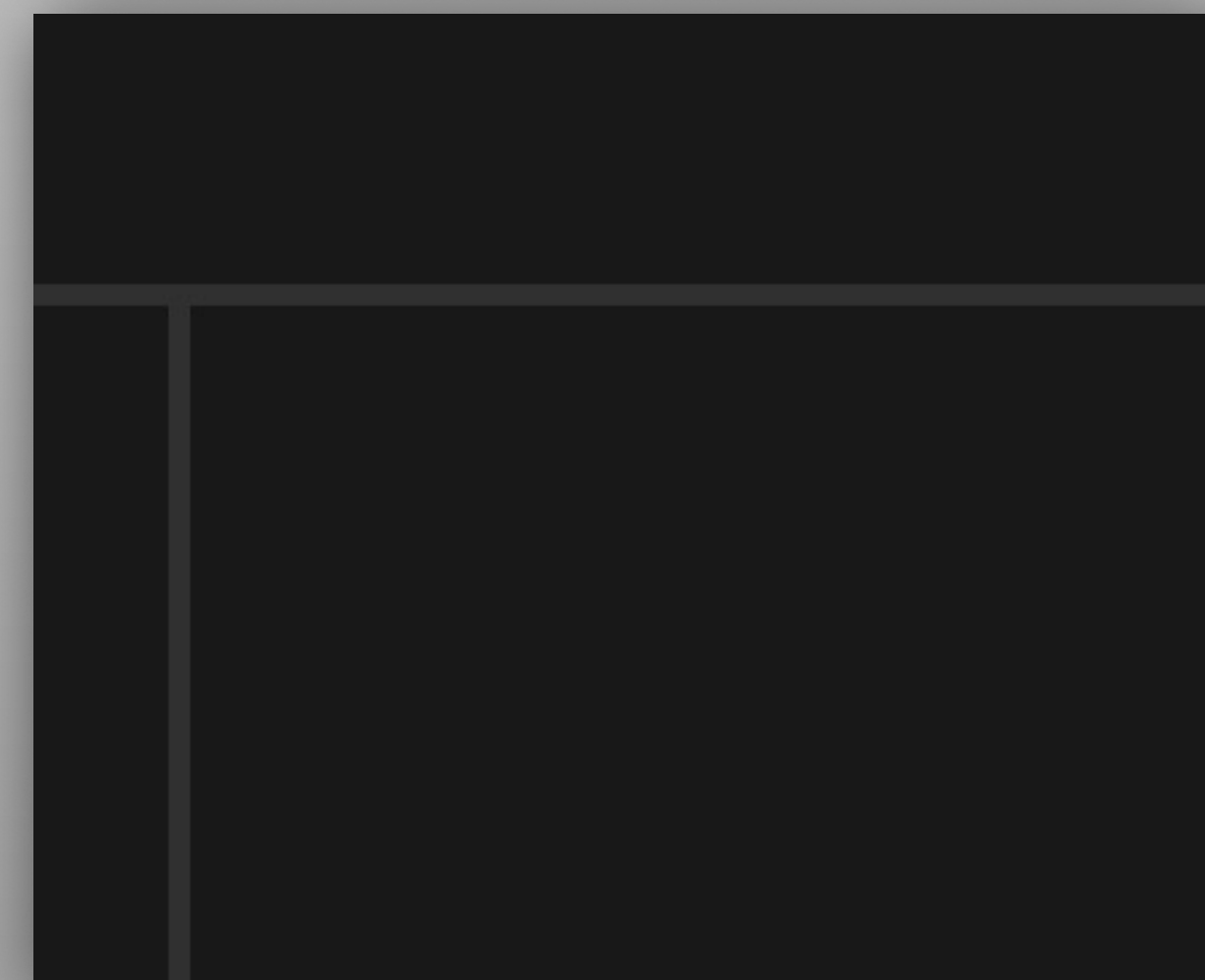
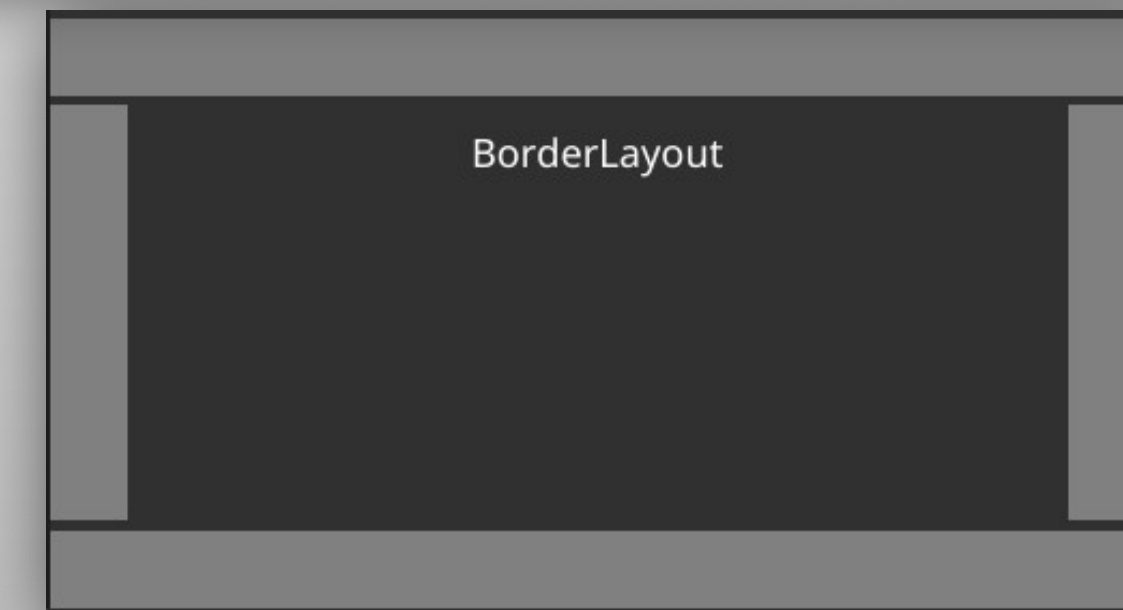
<b>Name</b>	<input type="text" value="John Smith"/> <small>Your full name</small>
<b>Email</b>	<input type="text" value="test@example.com"/> <small>A valid email address</small>
<b>Password</b>	<input type="password" value="Password"/> 




# Border

- Allows objects to be rendered at the borders (top, bottom, left, right) and in the middle.
- The border objects will be rendered at their minimum size (MinSize) and the middle item will take up the remaining space.
  - Border objects need to be passed to the layout and the container.

```
layout.NewBorderLayout(  
    top, bottom, left, right fyne.CanvasObject)
```

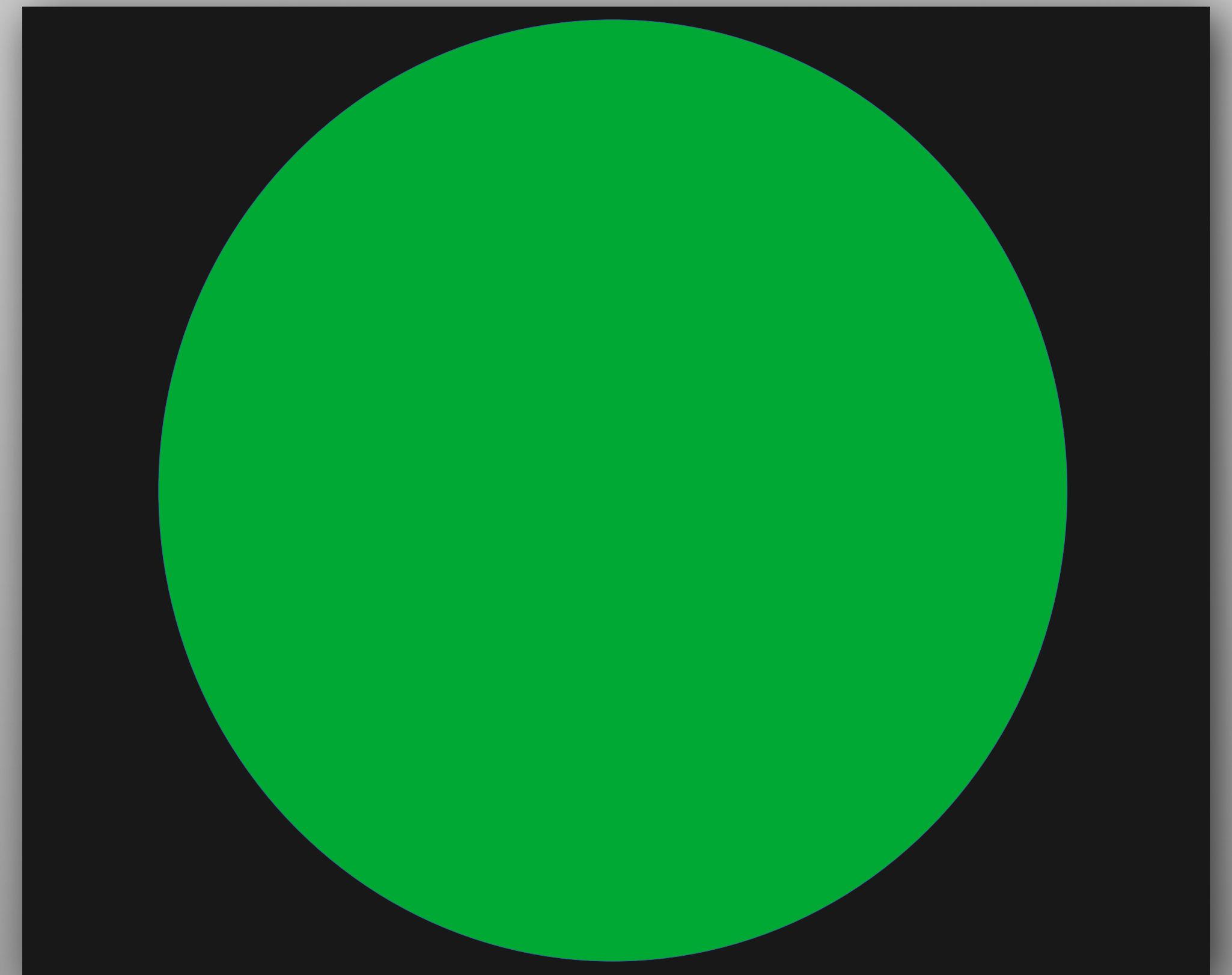




# Max

- Positions and sizes objects to fill the entire available space.
  - The default for a window unless another layout is used.
- Tip: With more than one object, it can be used to show objects above each other.

```
layout.NewMaxLayout( )
```

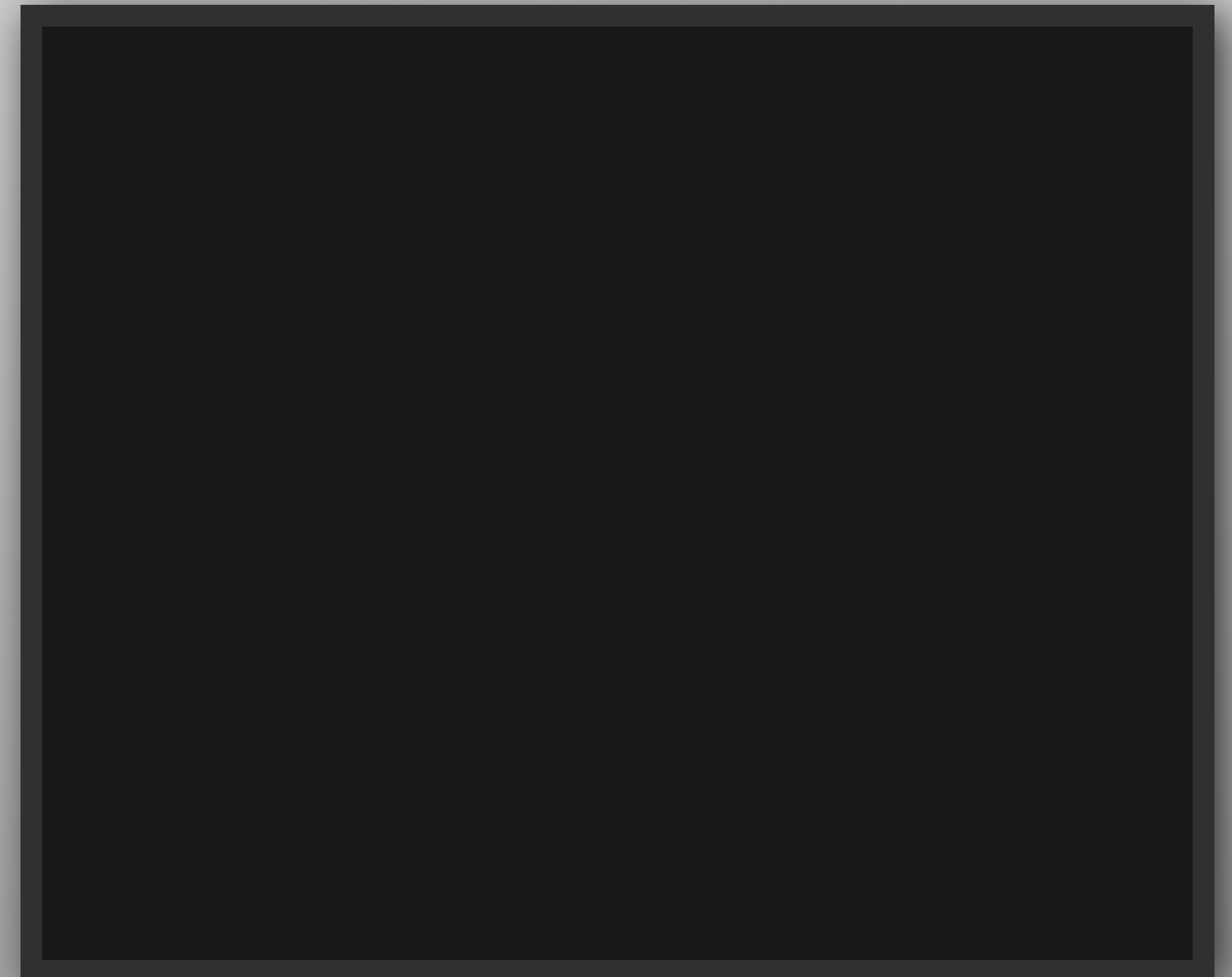




# Padded

```
layout.NewPaddedLayout( )
```

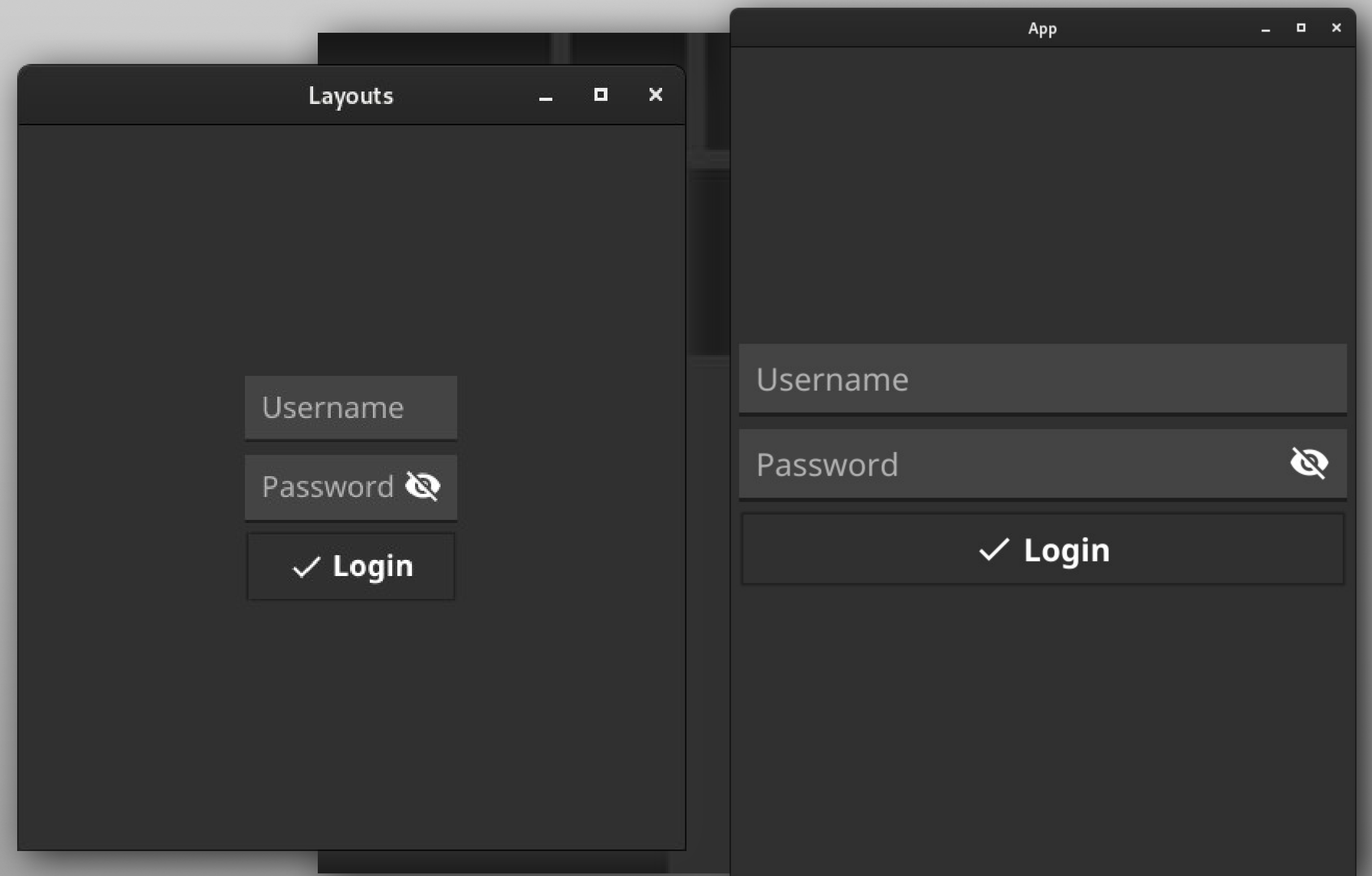
- Like a Max layout but adds padding around the object.
  - Useful for adding small spacing around an item.
- The padding size is controlled by the theme.





# Combining layouts

- Multiple simple layouts can be combined to create the desired layout.
- Next up: An example combining layouts to create a login screen that works good on mobile and desktop.





# Putting it into good use

- API reference:  
<https://developer.fyne.io/api/v2.0/layout/>
- FAQ about layouts and widget size:  
<https://developer.fyne.io/faq/layout>

