Basic UI One screen

https://www.syncfusion.com/succinctly-free-ebooks/xamarin-community-toolkit-succinctly/creating-the-user-interface-with-c-markup

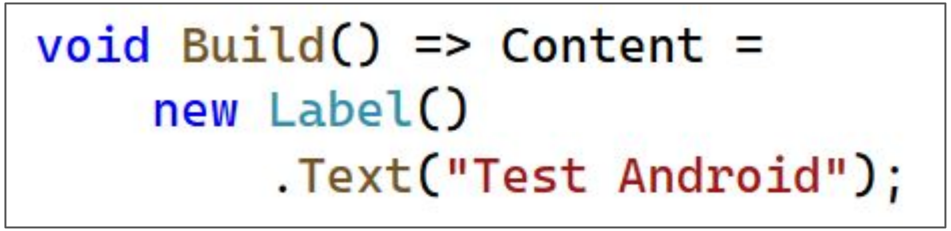
Let’s start inside a Page: **MainPage**.xaml.cs

- ❏ Remember to add C# Markup (**CommunityToolkit.Maui.Markup**)

Top common UI from the start

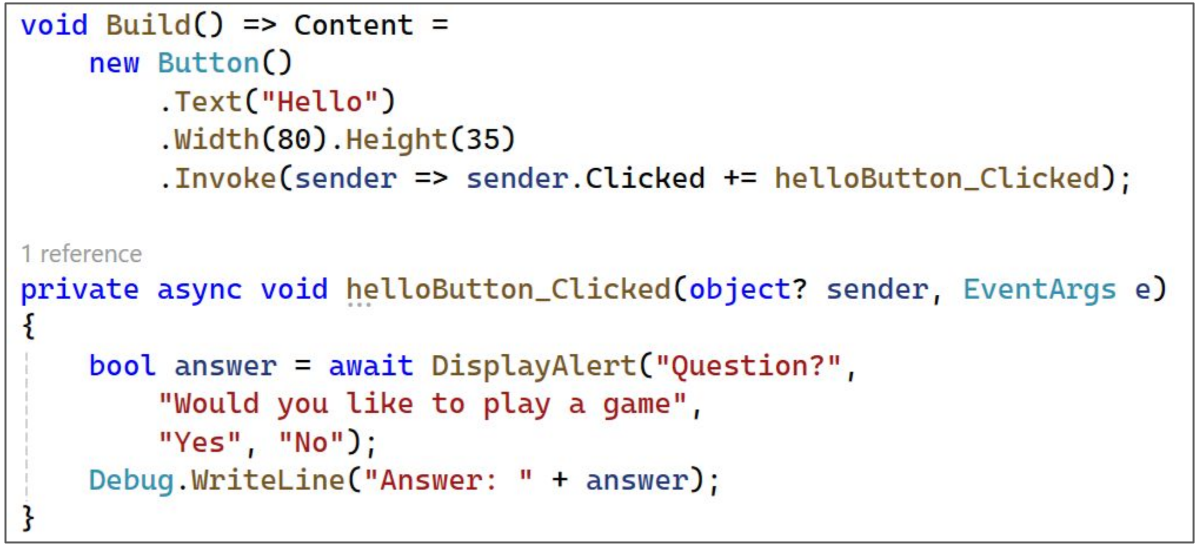
1. Label / **Text**
2. **Button / ImageButton**
3. TextBox / **Entry** / Input
4. **Image / FontIcon**
5. MessageBox / **Alert / ActionSheet**

1. Text

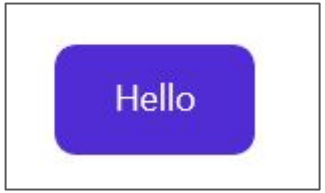


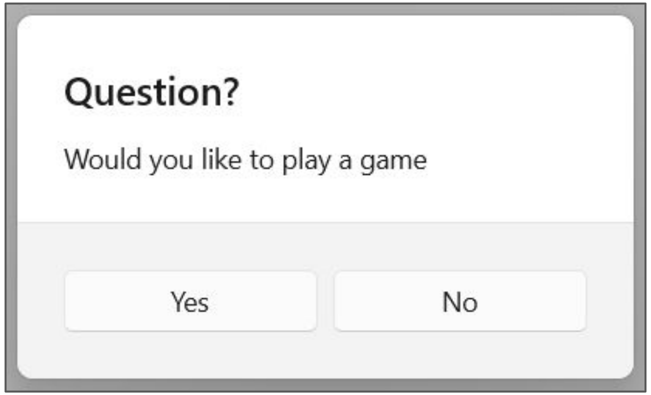
```
void Build() => Content =
    new Label()
        .Text("Test Android");
```

2. Button



```
void Build() => Content =
    new Button()
        .Text("Hello")
        .Width(80).Height(35)
        .Invoke(sender => sender.Clicked += helloButton_Clicked);

1 reference
private async void helloButton_Clicked(object? sender, EventArgs e)
{
    bool answer = await DisplayAlert("Question?",
        "Would you like to play a game",
        "Yes", "No");
    Debug.WriteLine("Answer: " + answer);
}
```



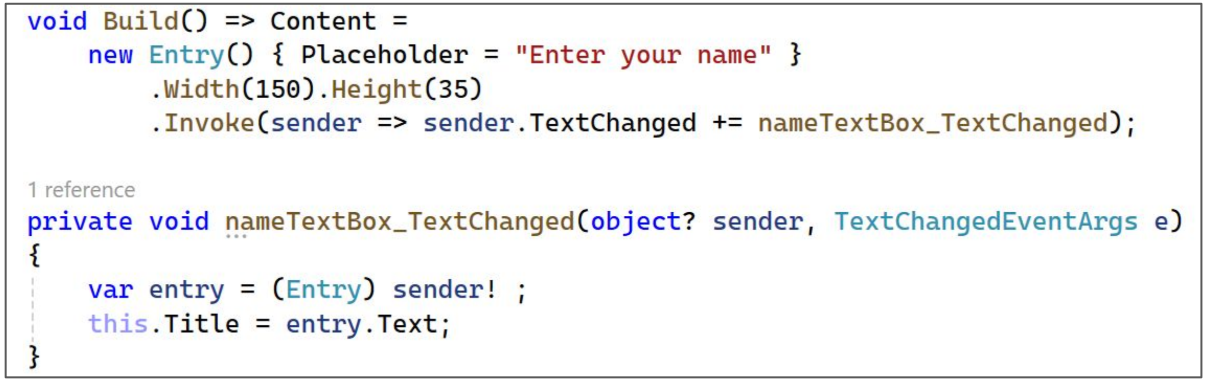


Question?

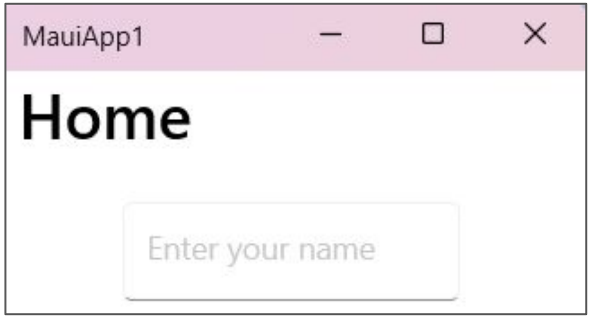
Would you like to play a game

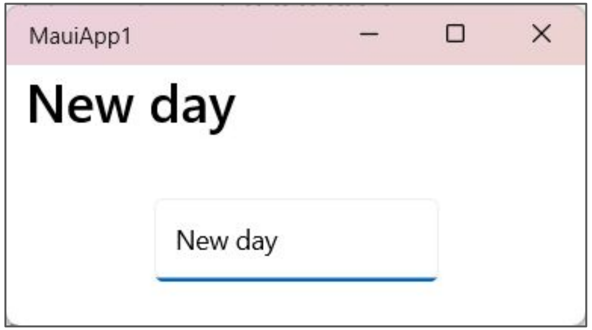
https://learn.microsoft.com/en-us/dotnet/maui/user-interface/pop-ups?view=net-maui-8.0

3. Entry



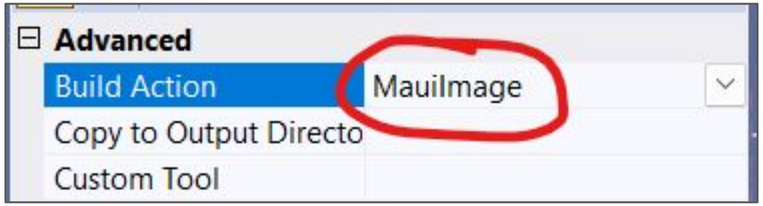
```
void Build() => Content =
    new Entry() { Placeholder = "Enter your name" }
        .Width(150).Height(35)
        .Invoke(sender => sender.TextChanged += nameTextBox_TextChanged);

1 reference
private void nameTextBox_TextChanged(object? sender, TextChangedEventArgs e)
{
    var entry = (Entry) sender! ;
    this.Title = entry.Text;
}
```





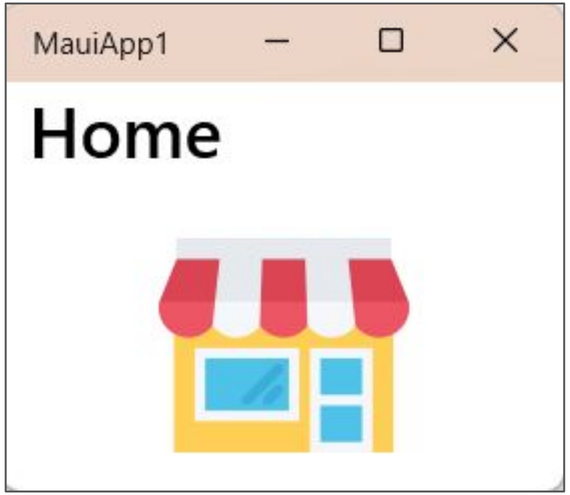
4. Image

```
void Build() => Content =
     new Image()
     .Source("shop.png")
     .Width(100).Height(100);
```



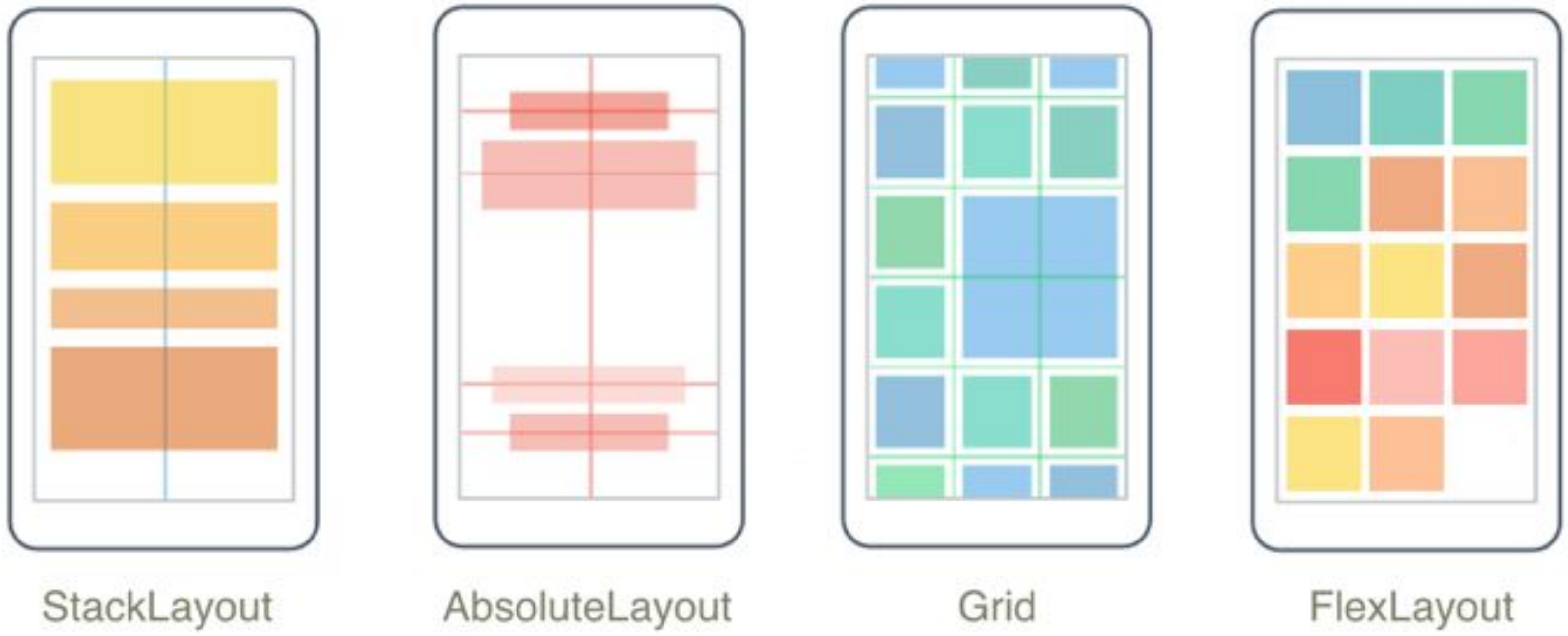
Advanced

Build Action	MauiImage
Copy to Output Directo	
Custom Tool	



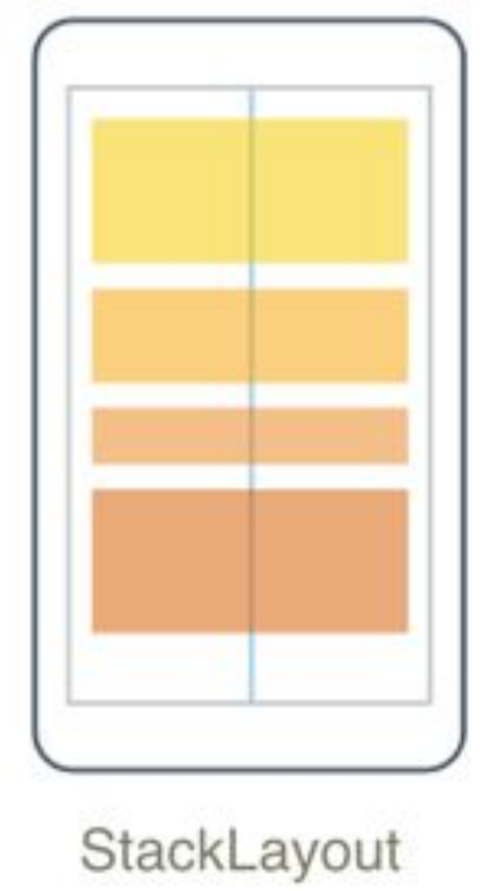
Basic layout

4 main types of layout



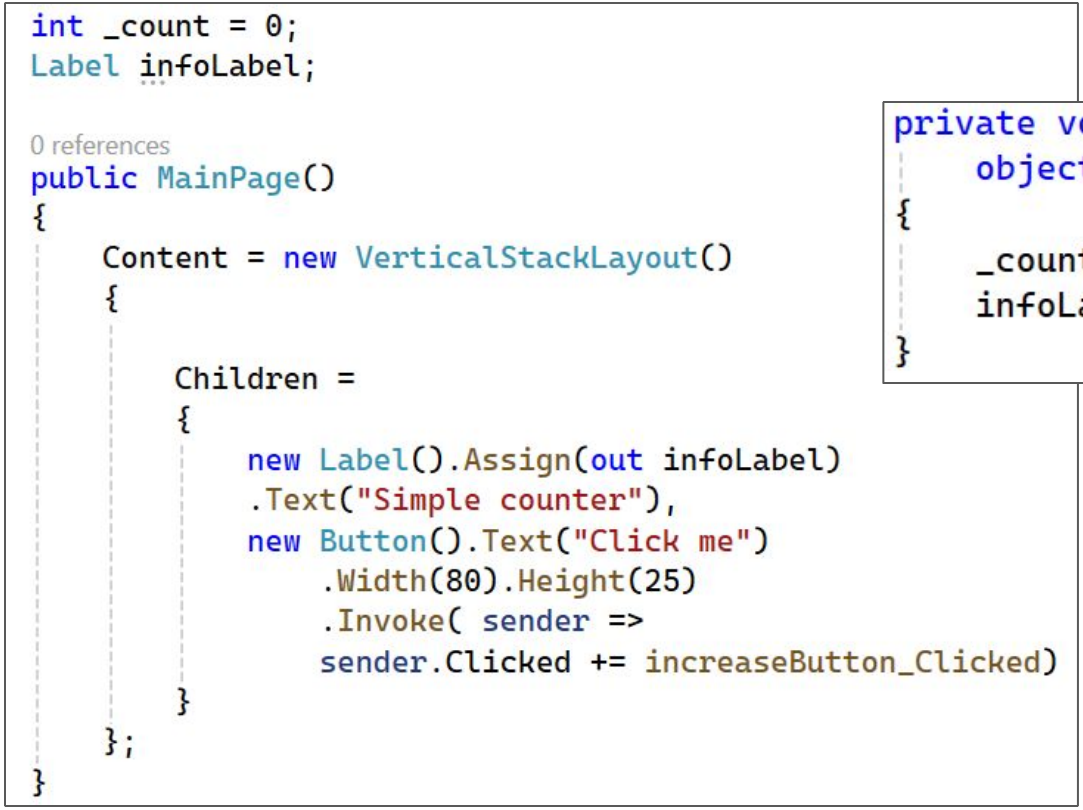
[Layouts - .NET MAUI | Microsoft Learn]

StackLayout

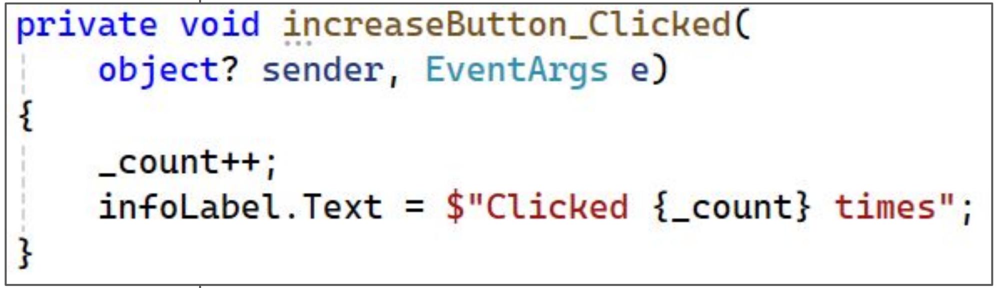


StackLayout - .NET MAUI | Microsoft Learn

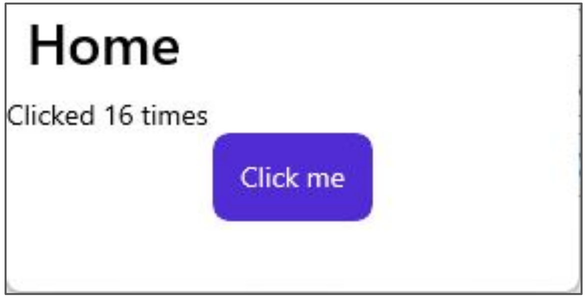
VerticalStackLayout



```
int _count = 0;
Label infoLabel;

0 references
public MainPage()
{
    Content = new VerticalStackLayout()
    {

        Children =
        {
            new Label().Assign(out infoLabel)
            .Text("Simple counter"),
            new Button().Text("Click me")
                .Width(80).Height(25)
                .Invoke( sender =>
                sender.Clicked += increaseButton_Clicked)
        }
    };
}
```



```
private void increaseButton_Clicked(
    object? sender, EventArgs e)
{
    _count++;
    infoLabel.Text = $"Clicked {_count} times";
}
```

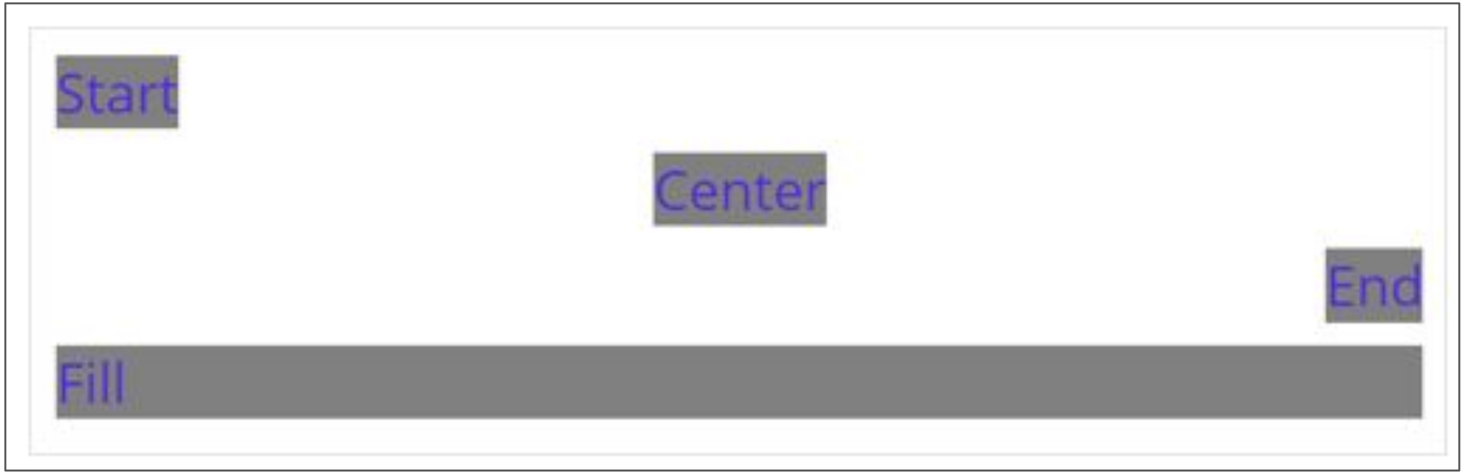


Home

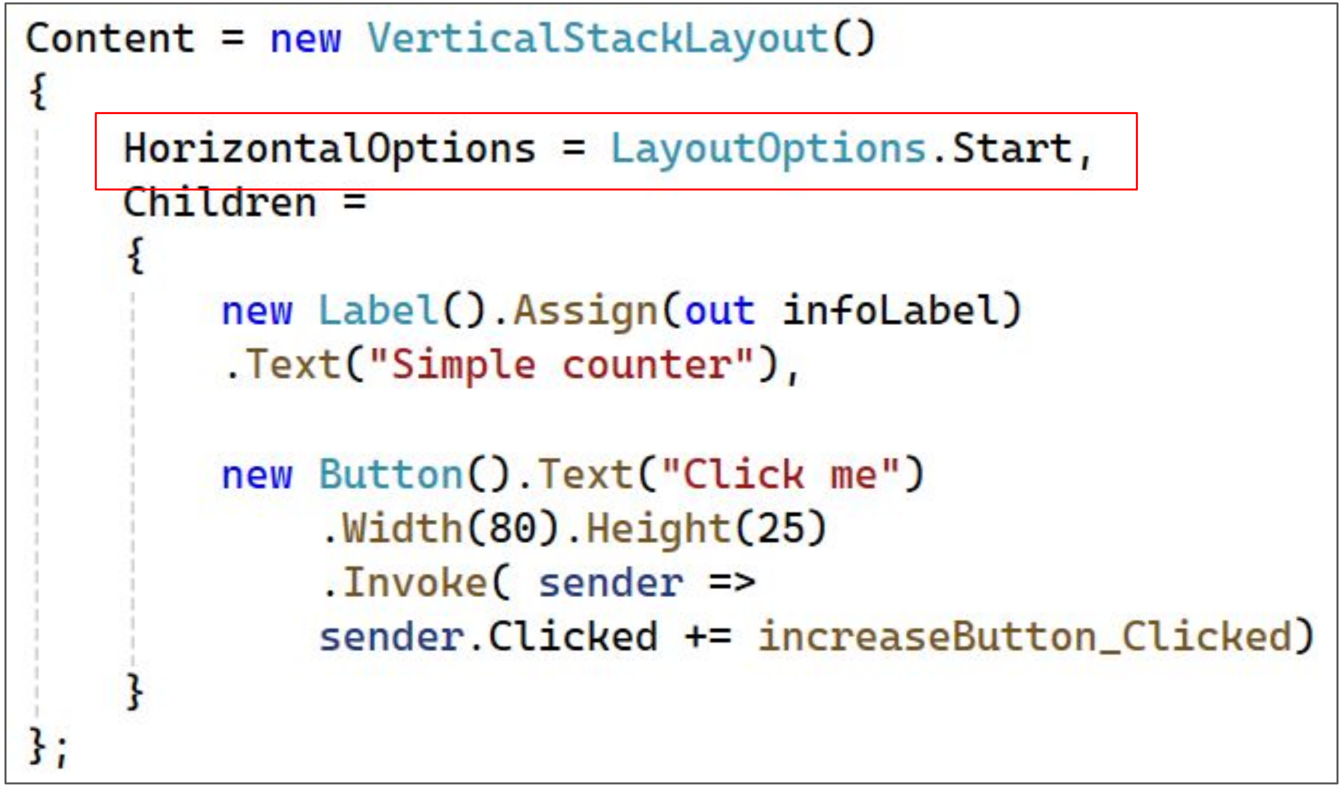
Clicked 16 times

[VerticalStackLayout - .NET MAUI | Microsoft Learn]

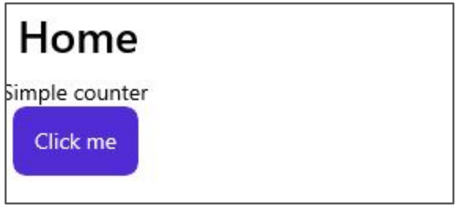
HorizontalOptions



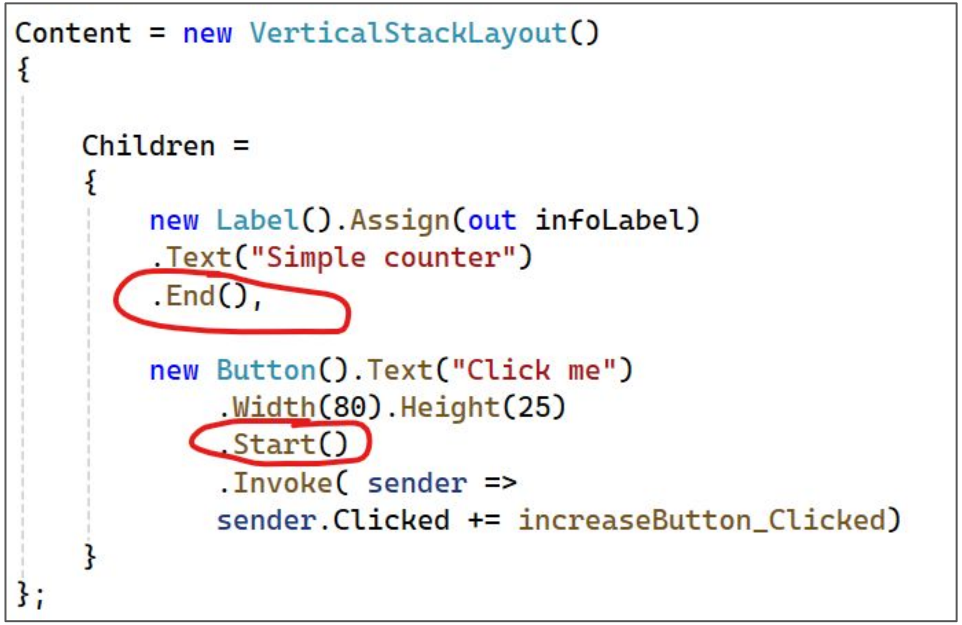
Forcing all left



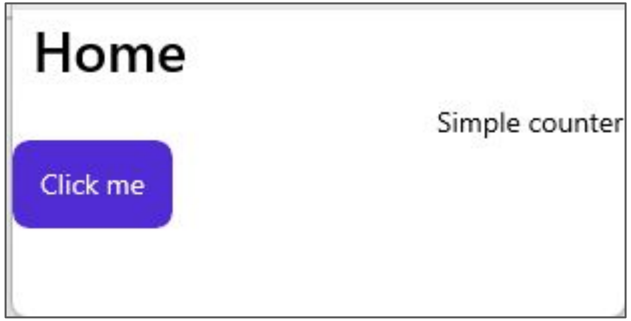
```
Content = new VerticalStackLayout()
{
    HorizontalOptions = LayoutOptions.Start,
    Children =
    {
        new Label().Assign(out infoLabel)
        .Text("Simple counter"),

        new Button().Text("Click me")
            .Width(80).Height(25)
            .Invoke( sender =>
            sender.Clicked += increaseButton_Clicked)
    }
};
```



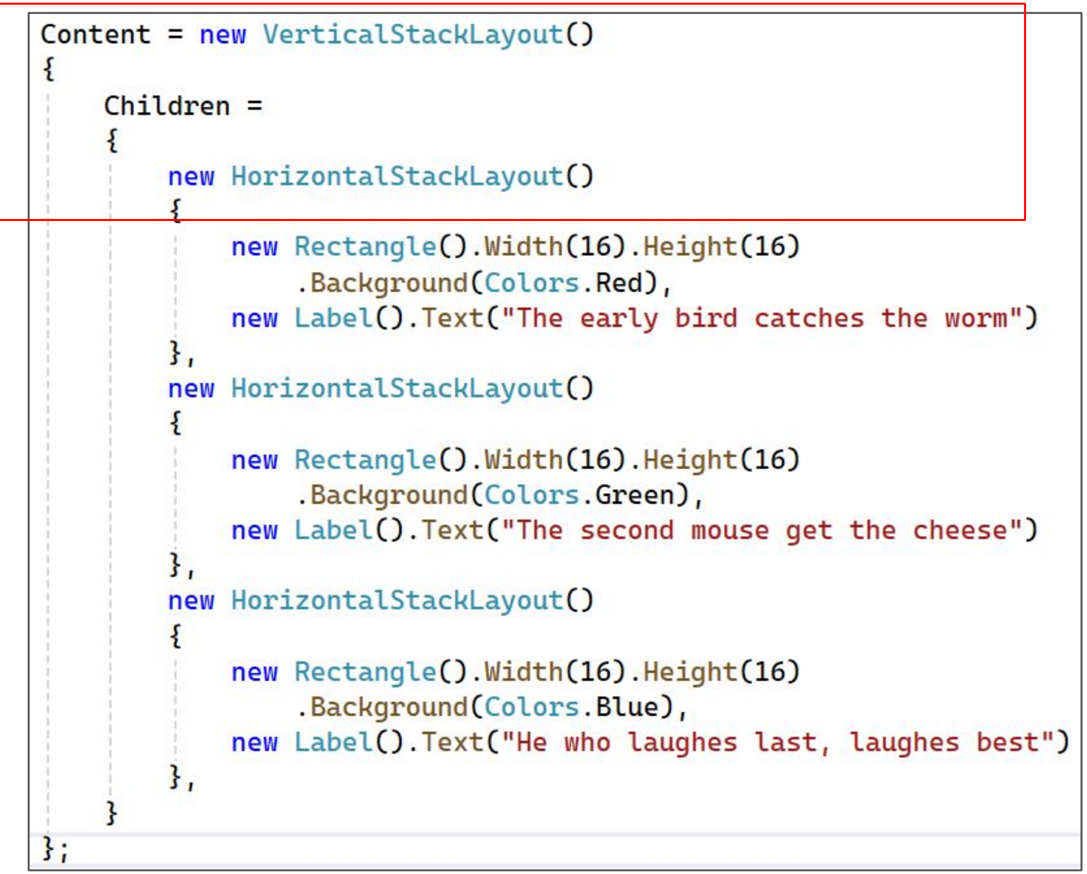
End & Start



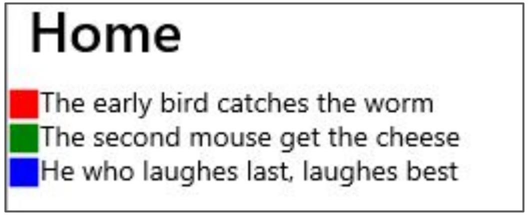
```
Content = new VerticalStackLayout()
{

    Children =
    {
        new Label().Assign(out infoLabel)
        .Text("Simple counter")
        .End(),

        new Button().Text("Click me")
            .Width(80).Height(25)
            .Start()
            .Invoke( sender =>
            sender.Clicked += increaseButton_Clicked)
    }
};
```



Vertical & Horizontal combination



```
Content = new VerticalStackLayout()
{
    Children =
    {
        new HorizontalStackLayout()
        {
            new Rectangle().Width(16).Height(16)
                .Background(Colors.Red),
            new Label().Text("The early bird catches the worm")
        },
        new HorizontalStackLayout()
        {
            new Rectangle().Width(16).Height(16)
                .Background(Colors.Green),
            new Label().Text("The second mouse get the cheese")
        },
        new HorizontalStackLayout()
        {
            new Rectangle().Width(16).Height(16)
                .Background(Colors.Blue),
            new Label().Text("He who laughes last, laughes best")
        },
    }
};
```

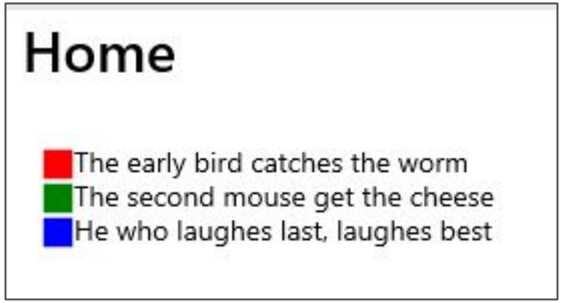


Home

🟥The early bird catches the worm
🟩The second mouse get the cheese
🟦He who laughes last, laughes best

Margin & Spacing

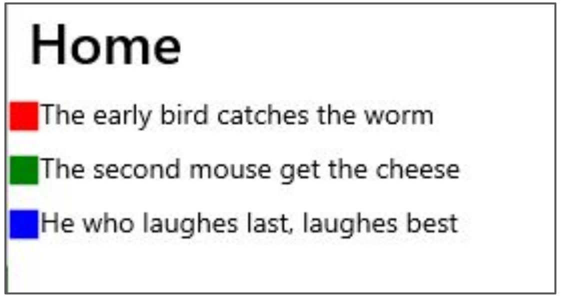
Margin = 20



Home

🟥The early bird catches the worm
🟩The second mouse get the cheese
🟦He who laughes last, laughes best

Spacing = 10



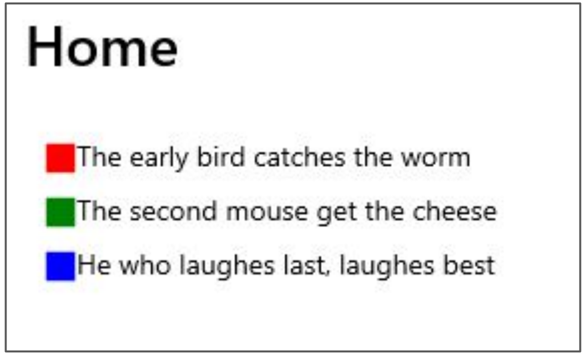
Home

🟥The early bird catches the worm

🟩The second mouse get the cheese

🟦He who laughes last, laughes best

Margin = 20
Spacing = 10



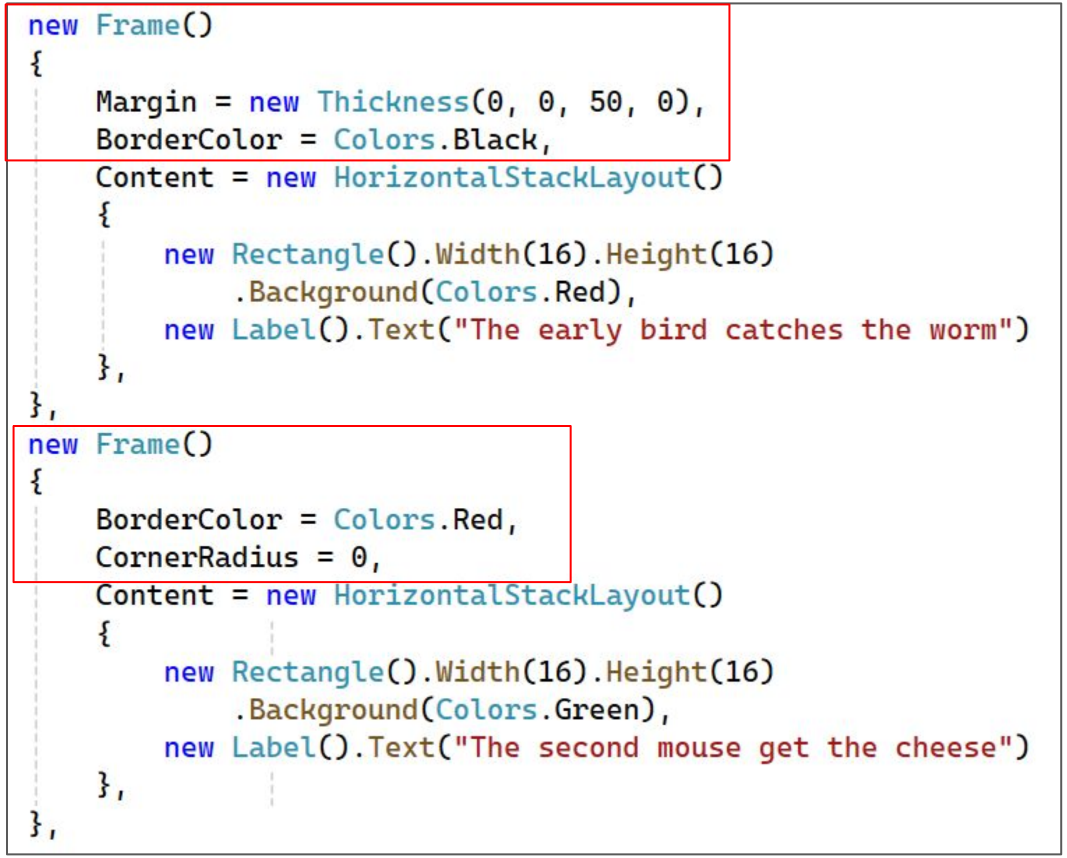
Home

🟥The early bird catches the worm

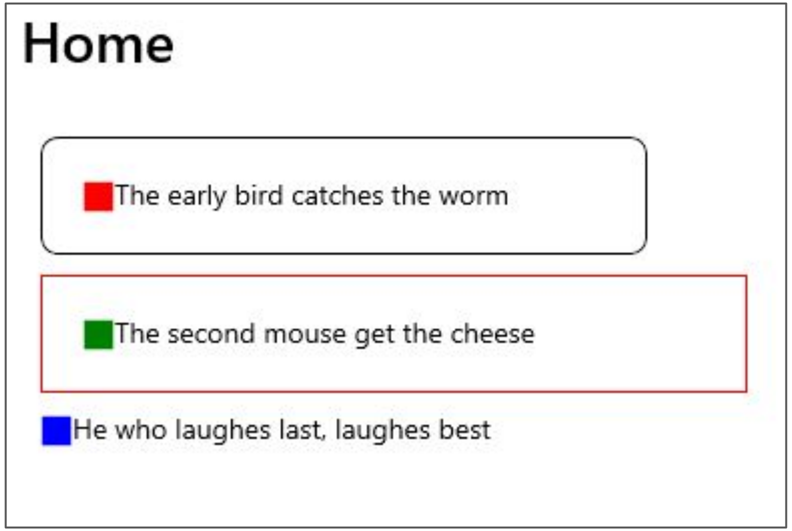
🟩The second mouse get the cheese

🟦He who laughes last, laughes best

Adding a Frame (deprecated - Border instead)



```
new Frame()
{
    Margin = new Thickness(0, 0, 50, 0),
    BorderColor = Colors.Black,
    Content = new HorizontalStackLayout()
    {
        new Rectangle().Width(16).Height(16)
            .Background(Colors.Red),
        new Label().Text("The early bird catches the worm")
    },
},
new Frame()
{
    BorderColor = Colors.Red,
    CornerRadius = 0,
    Content = new HorizontalStackLayout()
    {
        new Rectangle().Width(16).Height(16)
            .Background(Colors.Green),
        new Label().Text("The second mouse get the cheese")
    },
},
```



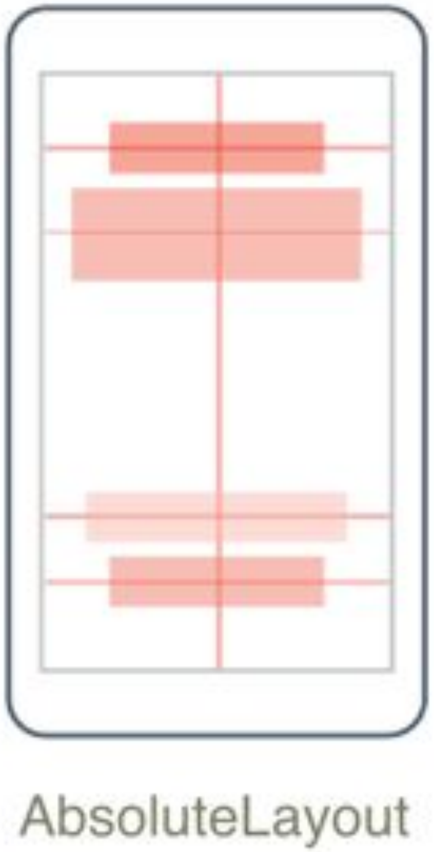
Home

🟥The early bird catches the worm

🟩The second mouse get the cheese

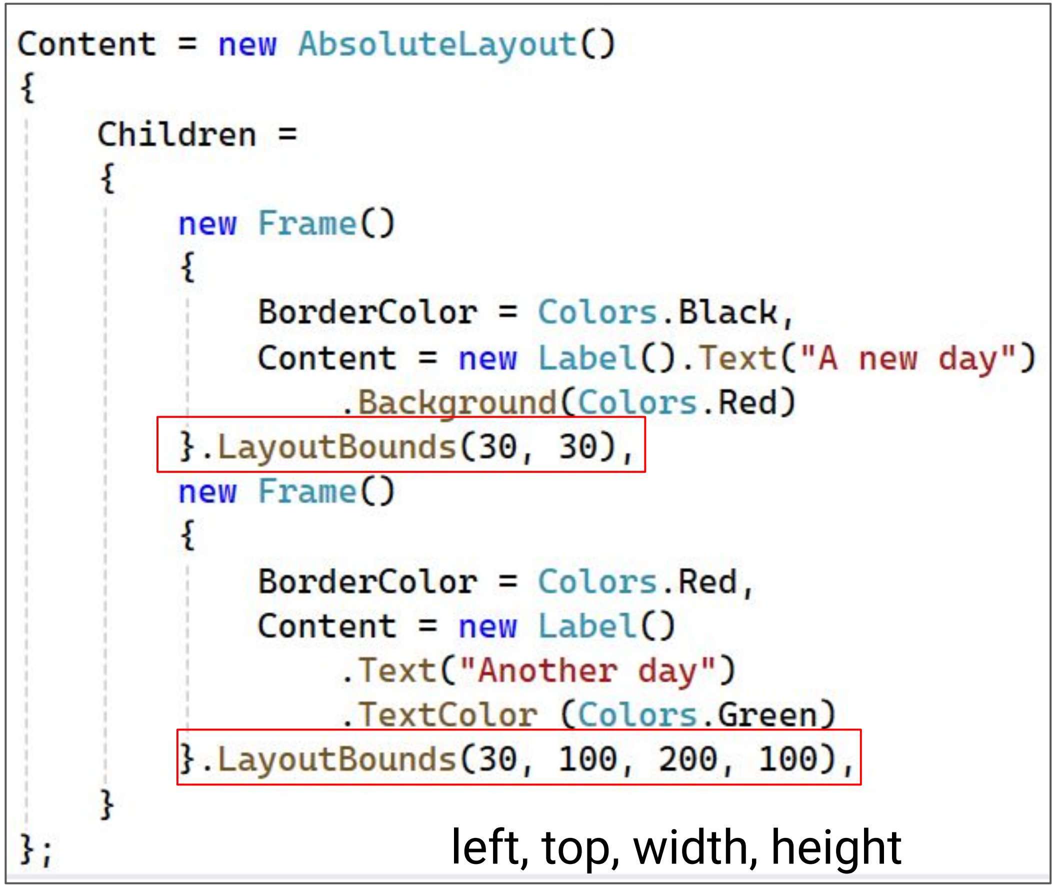
🟦He who laughes last, laughes best

AbsoluteLayout



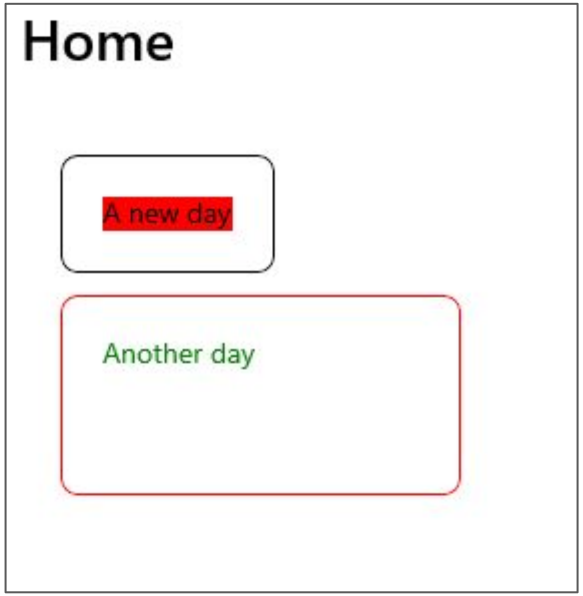
[AbsoluteLayout - .NET MAUI | Microsoft Learn]

Absolute position

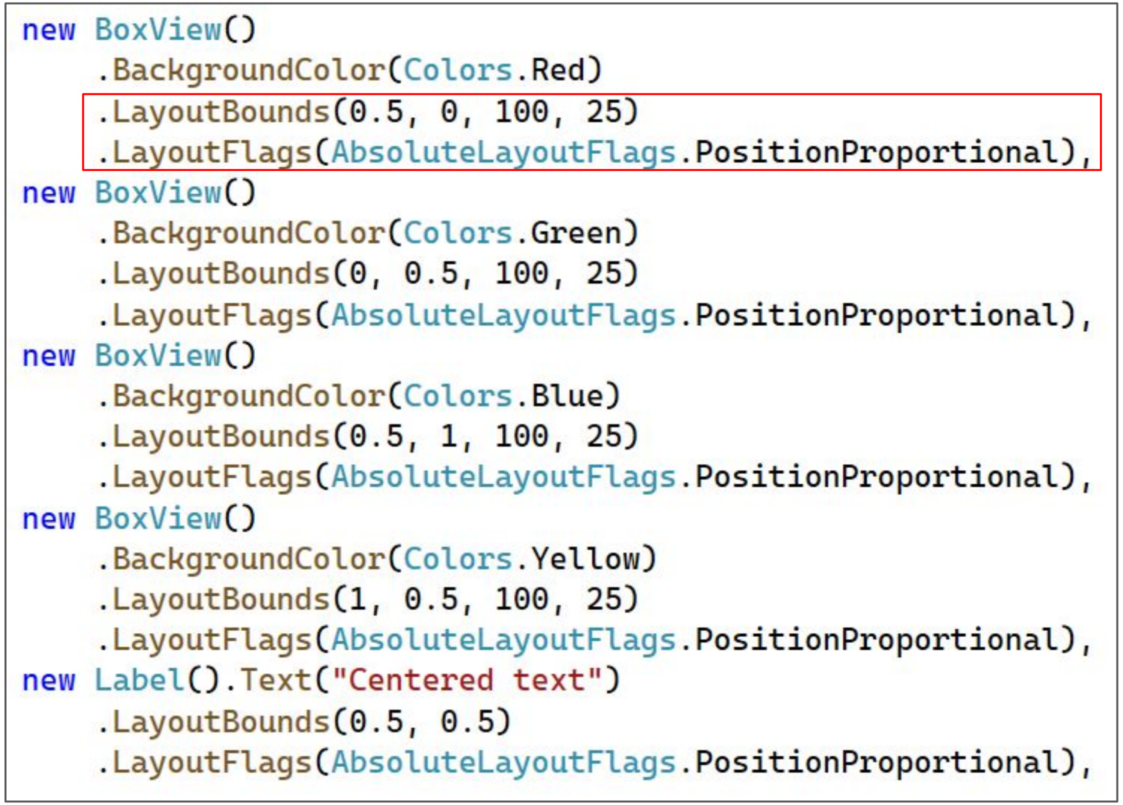


```
Content = new AbsoluteLayout()
{
    Children =
    {
        new Frame()
        {
            BorderColor = Colors.Black,
            Content = new Label().Text("A new day")
                .Background(Colors.Red)
        }.LayoutBounds(30, 30),
        new Frame()
        {
            BorderColor = Colors.Red,
            Content = new Label()
                .Text("Another day")
                .TextColor (Colors.Green)
        }.LayoutBounds(30, 100, 200, 100),
    }
};
```

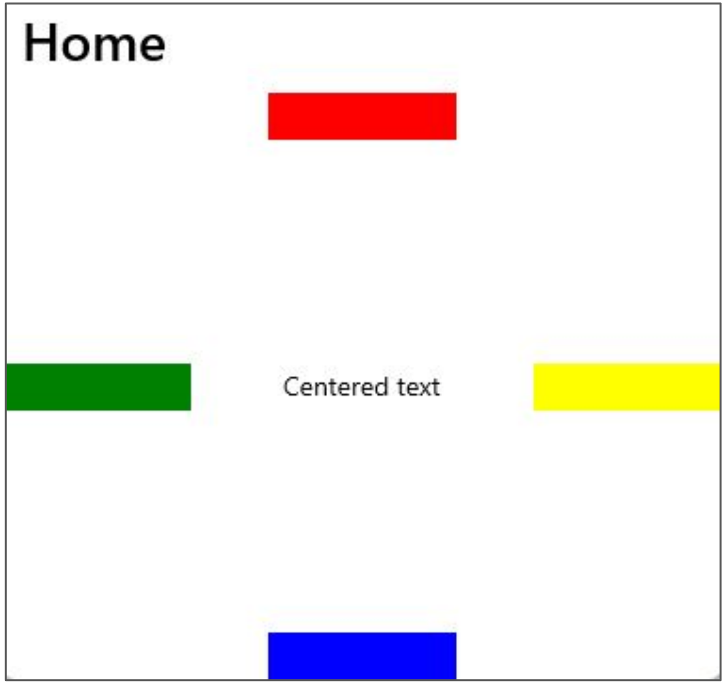
left, top, width, height



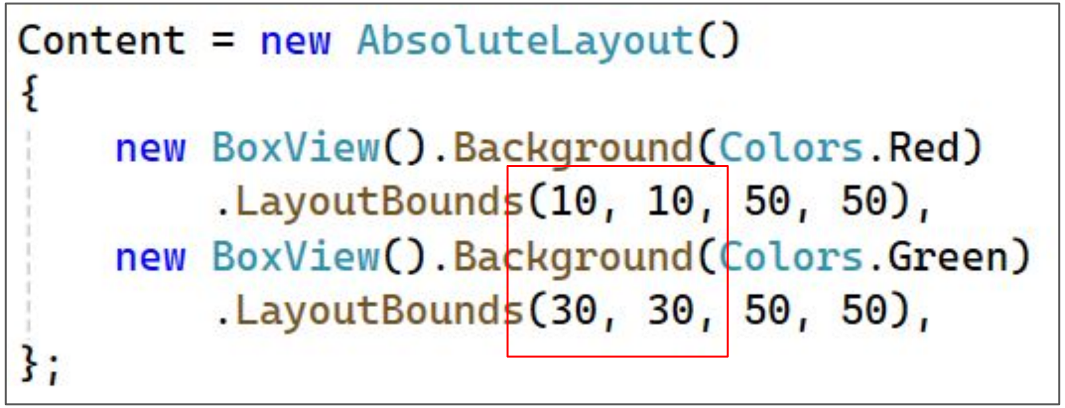
Proportional position



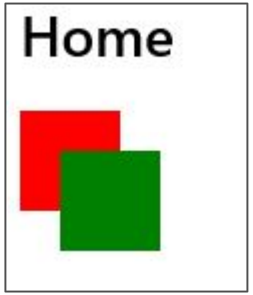
```
new BoxView()
    .BackgroundColor(Colors.Red)
    .LayoutBounds(0.5, 0, 100, 25)
    .LayoutFlags(AbsoluteLayoutFlags.PositionProportional),
new BoxView()
    .BackgroundColor(Colors.Green)
    .LayoutBounds(0, 0.5, 100, 25)
    .LayoutFlags(AbsoluteLayoutFlags.PositionProportional),
new BoxView()
    .BackgroundColor(Colors.Blue)
    .LayoutBounds(0.5, 1, 100, 25)
    .LayoutFlags(AbsoluteLayoutFlags.PositionProportional),
new BoxView()
    .BackgroundColor(Colors.Yellow)
    .LayoutBounds(1, 0.5, 100, 25)
    .LayoutFlags(AbsoluteLayoutFlags.PositionProportional),
new Label().Text("Centered text")
    .LayoutBounds(0.5, 0.5)
    .LayoutFlags(AbsoluteLayoutFlags.PositionProportional),
```



Overlapping



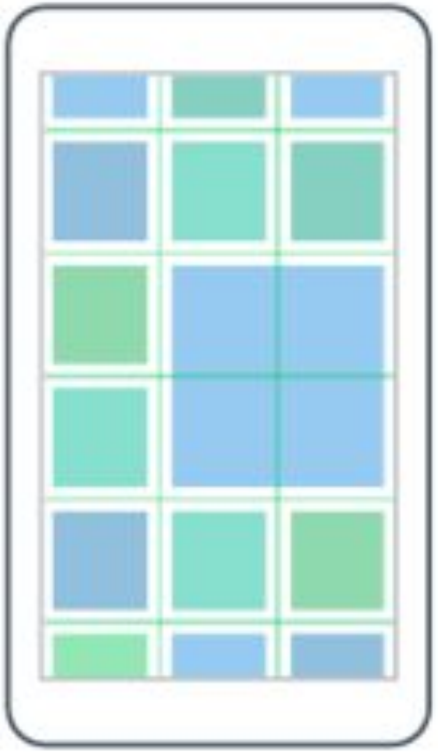
```
Content = new AbsoluteLayout()
{
    new BoxView().Background(Colors.Red)
        .LayoutBounds(10, 10, 50, 50),
    new BoxView().Background(Colors.Green)
        .LayoutBounds(30, 30, 50, 50),
};
```



Other options

- SizeProportional
- All

Grid



Grid

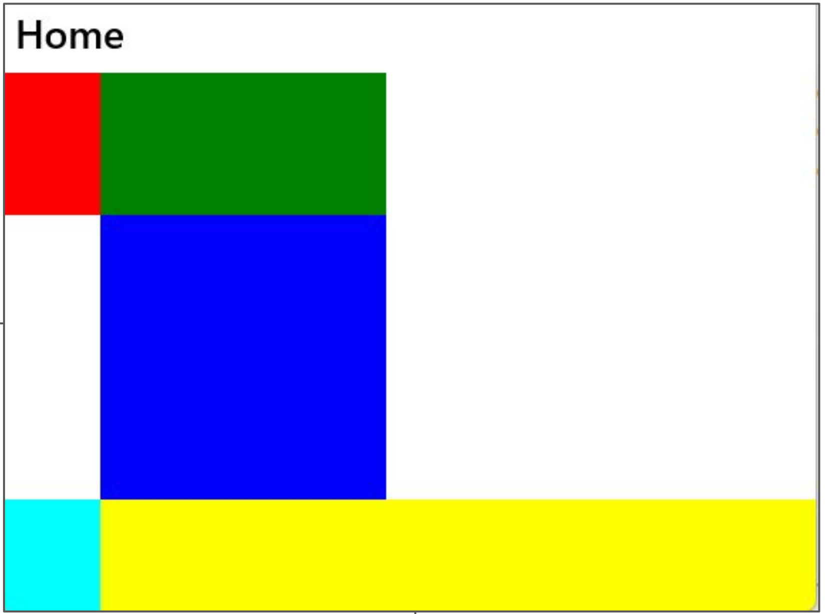
Grid - .NET MAUI | Microsoft Learn

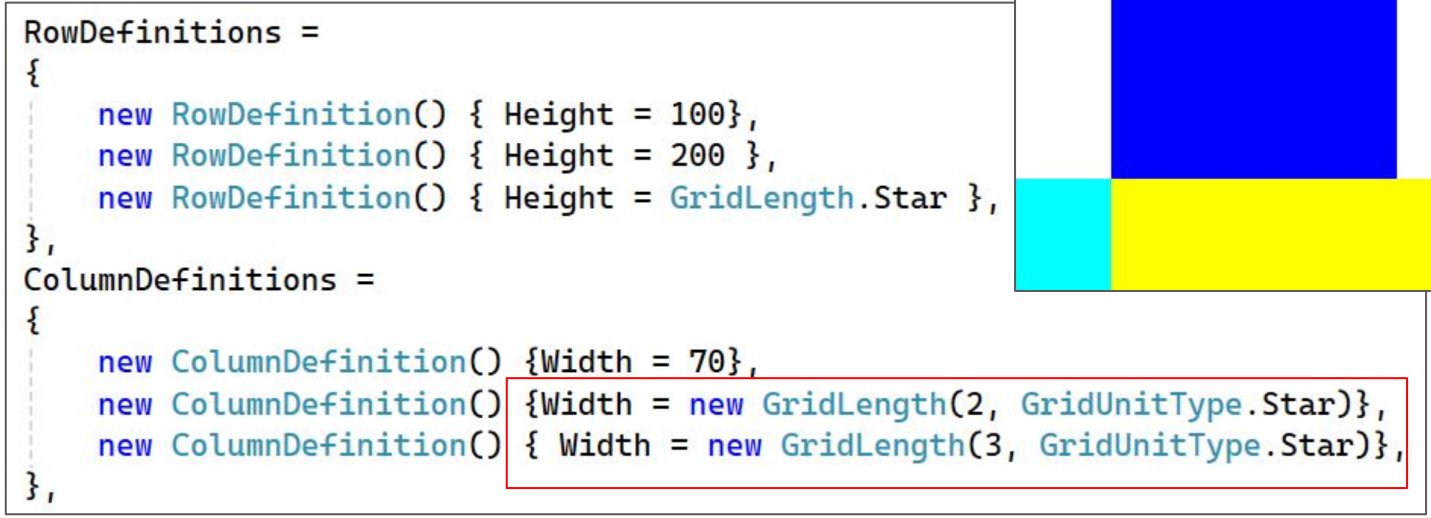
Grid

- Organizes its children into rows and columns
- Can have proportional or absolute sizes
- Used for **layout**

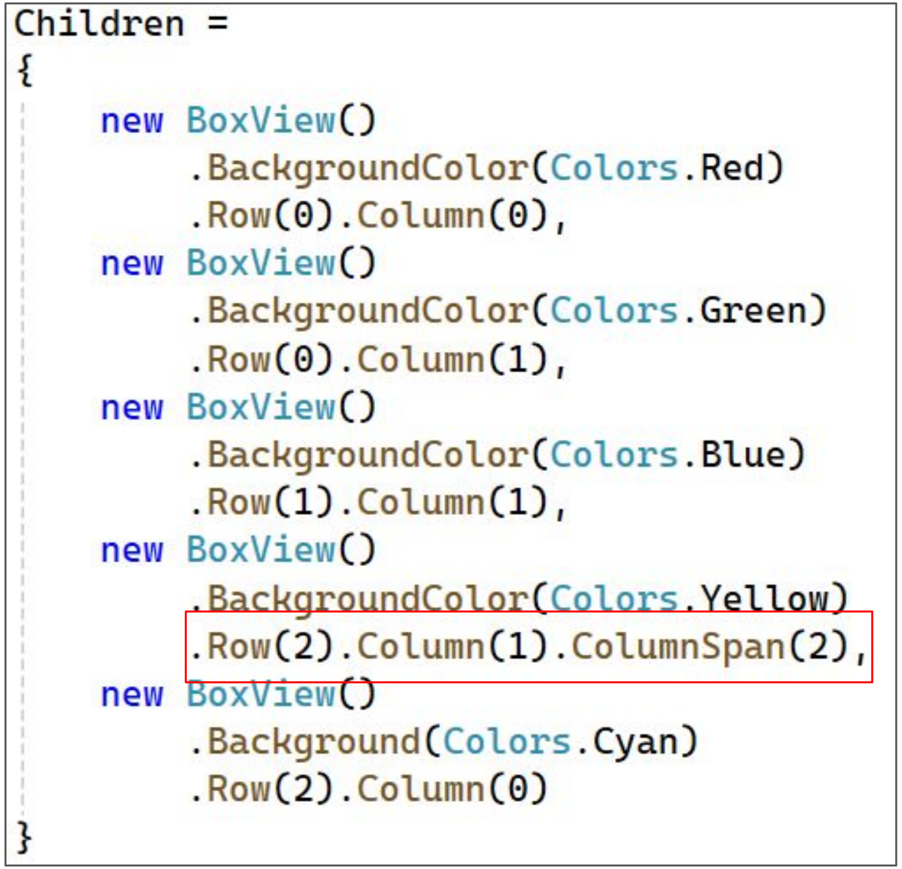
- For displaying **tabular data,** use
  - ListView
  - CollectionView

Grid definition

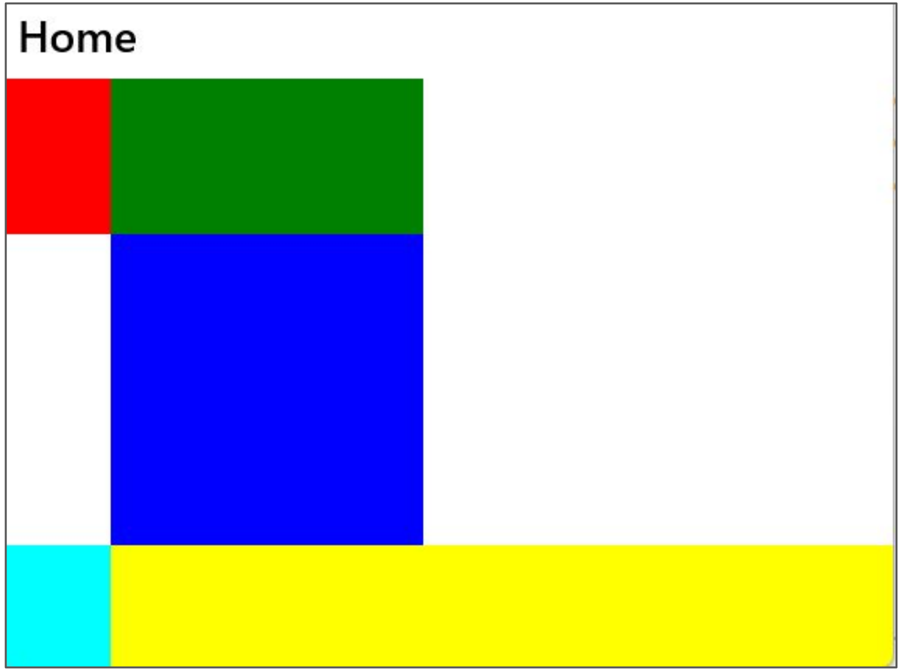



```
RowDefinitions =
{
    new RowDefinition() { Height = 100},
    new RowDefinition() { Height = 200 },
    new RowDefinition() { Height = GridLength.Star },
},
ColumnDefinitions =
{
    new ColumnDefinition() {Width = 70},
    new ColumnDefinition() {Width = new GridLength(2, GridUnitType.Star)},
    new ColumnDefinition() { Width = new GridLength(3, GridUnitType.Star)},
},
```

Setting up position



```
Children =
{
    new BoxView()
        .BackgroundColor(Colors.Red)
        .Row(0).Column(0),
    new BoxView()
        .BackgroundColor(Colors.Green)
        .Row(0).Column(1),
    new BoxView()
        .BackgroundColor(Colors.Blue)
        .Row(1).Column(1),
    new BoxView()
        .BackgroundColor(Colors.Yellow)
        .Row(2).Column(1).ColumnSpan(2),
    new BoxView()
        .Background(Colors.Cyan)
        .Row(2).Column(0)
}
```



Star vs Auto

- **Star**: use the left space available
- **Auto**: content width will be the column width

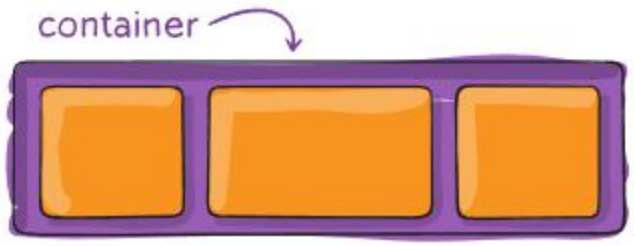
FlexLayout

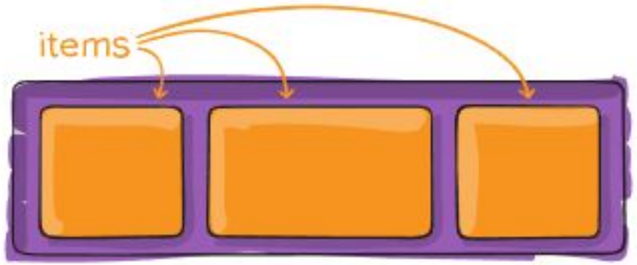
FlexLayout - .NET MAUI | Microsoft Learn

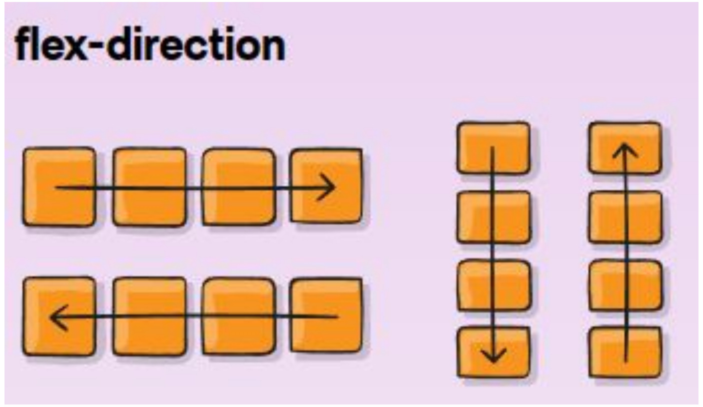
FlexLayout

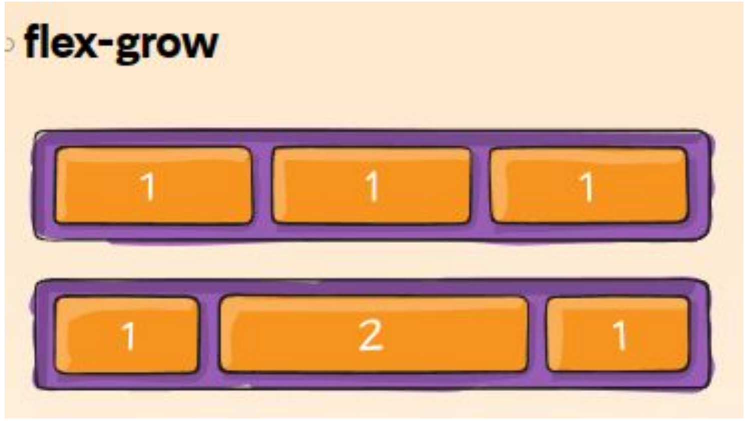
- Adapt to different screen sizes
- Based on the Cascading Style Sheets (CSS) **Flexible Box** Layout Module.

FlexBox model 01



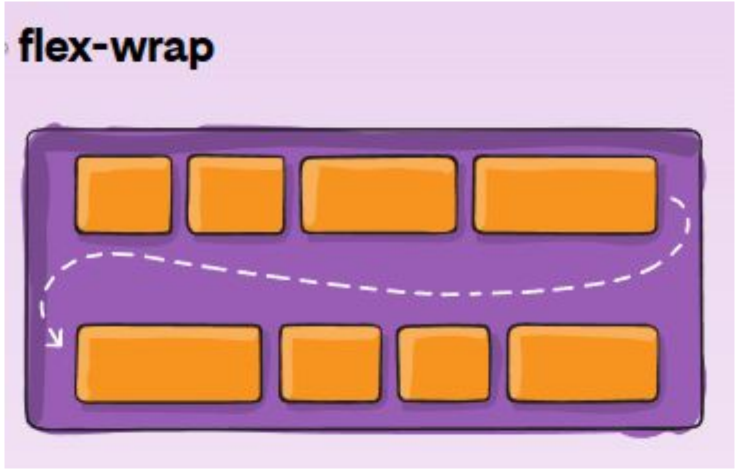


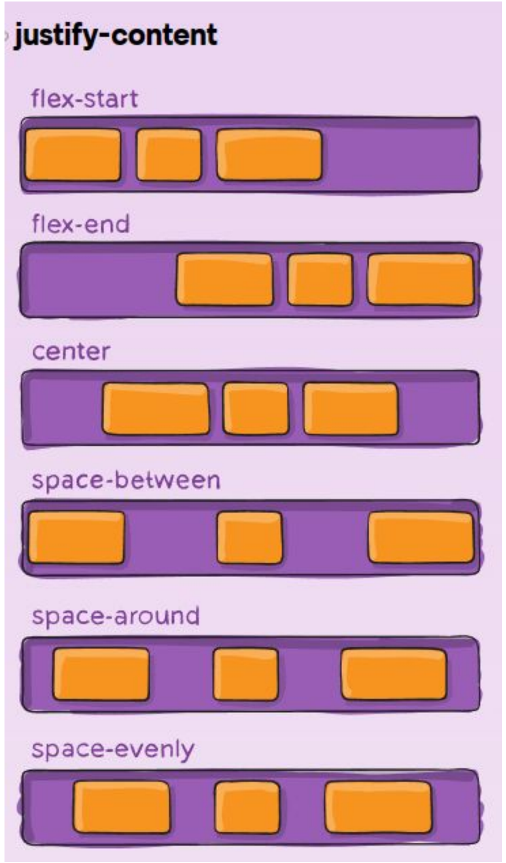


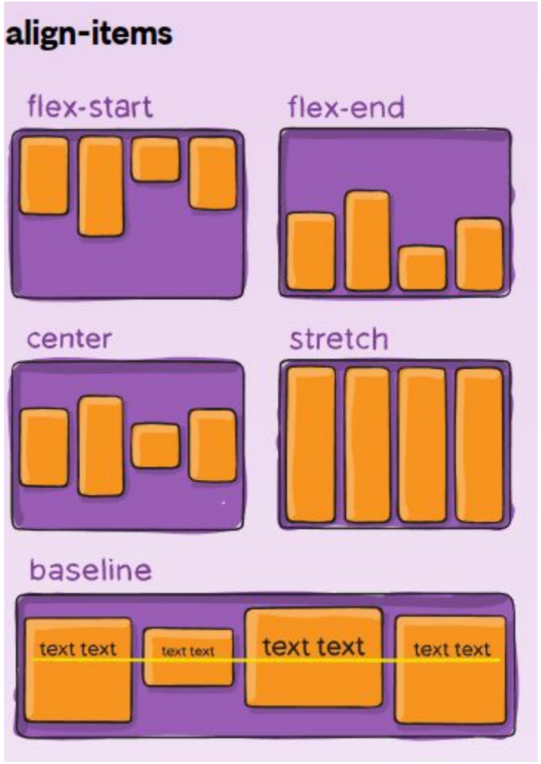


[A Complete Guide to Flexbox | CSS-Tricks - CSS-Tricks]

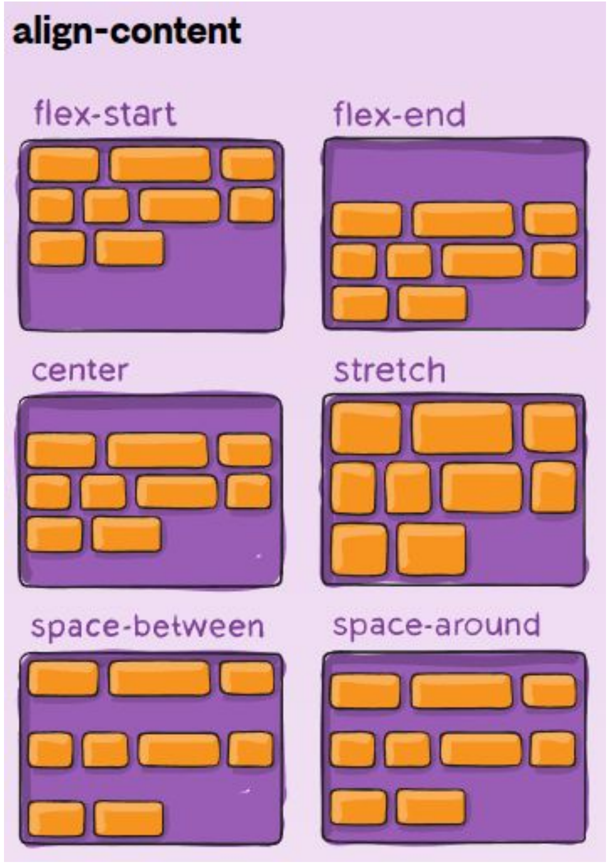
FlexBox model 02

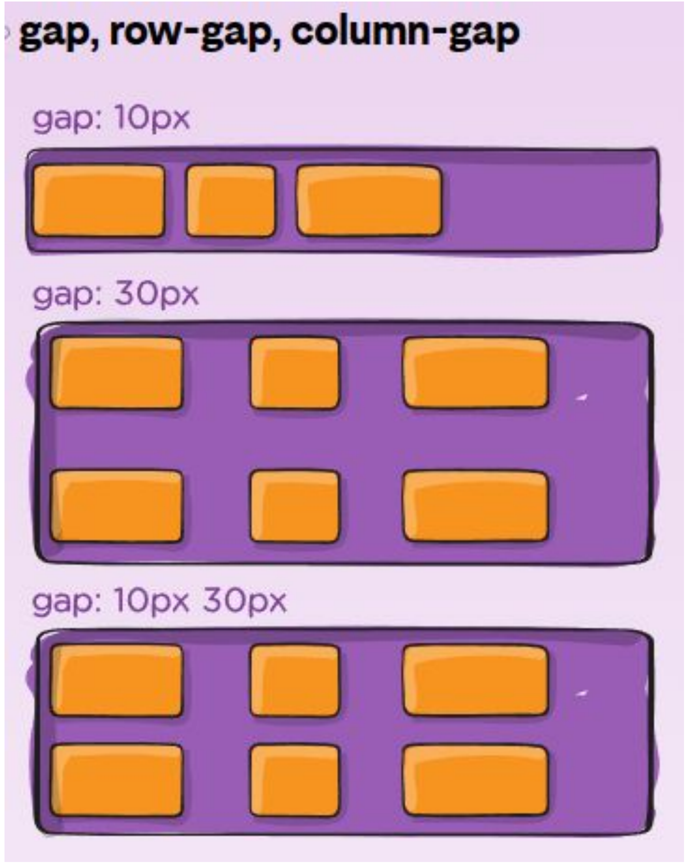






FlexBox model 03

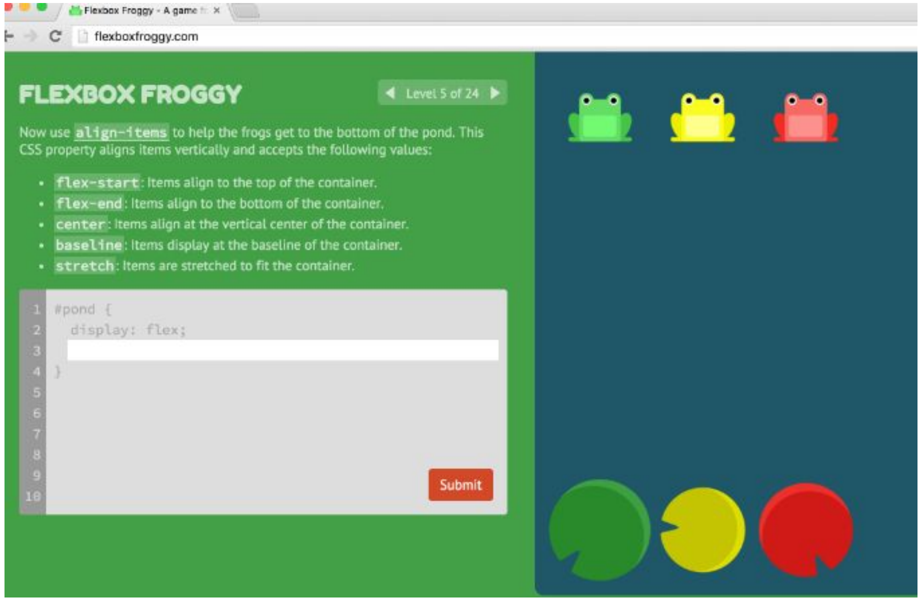




FlexBox Froggy

To understand more about FlexBox, play this game

https://flexboxfroggy.com

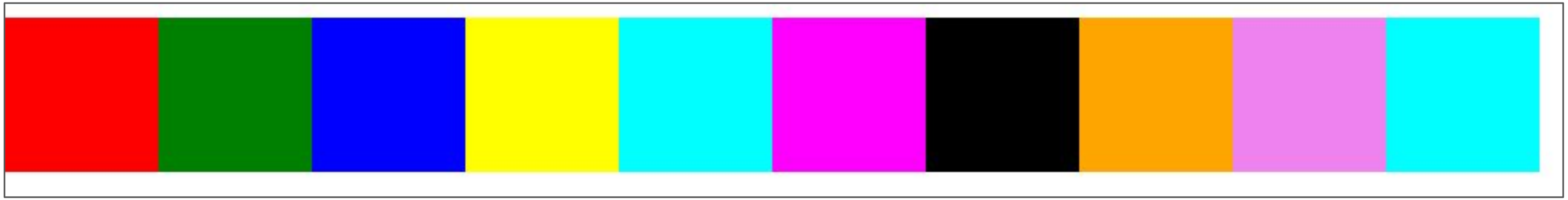


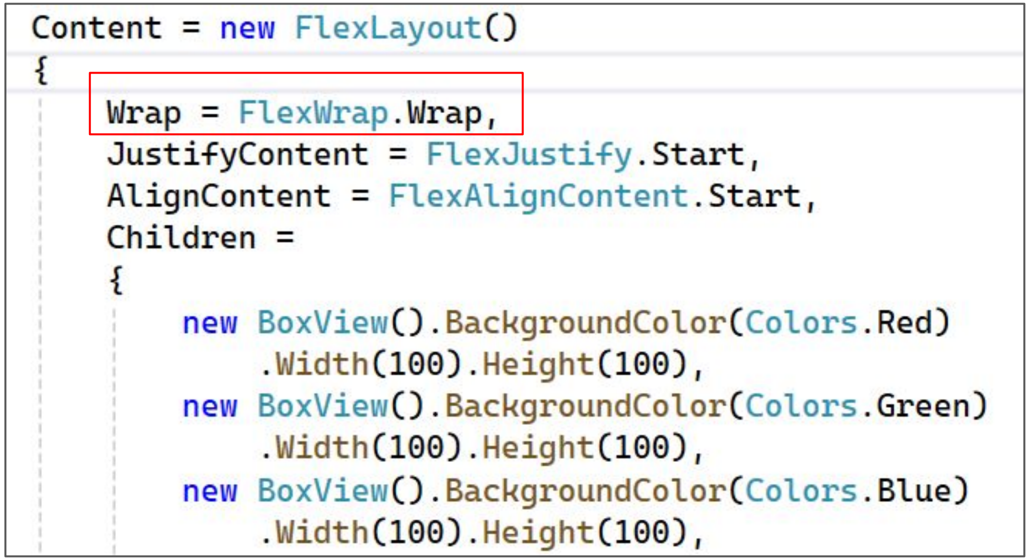
Now use `align-items` to help the frogs get to the bottom of the pond. This CSS property aligns items vertically and accepts the following values:

- `flex-start`: Items align to the top of the container.
- `flex-end`: Items align to the bottom of the container.
- `center`: Items align at the vertical center of the container.
- `baseline`: Items display at the baseline of the container.
- `stretch`: Items are stretched to fit the container.

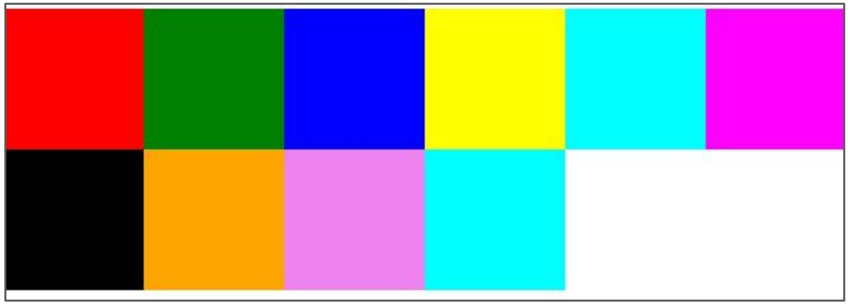
```
#pond {
  display: flex;

}
```

Usage 01 - Wrap Item



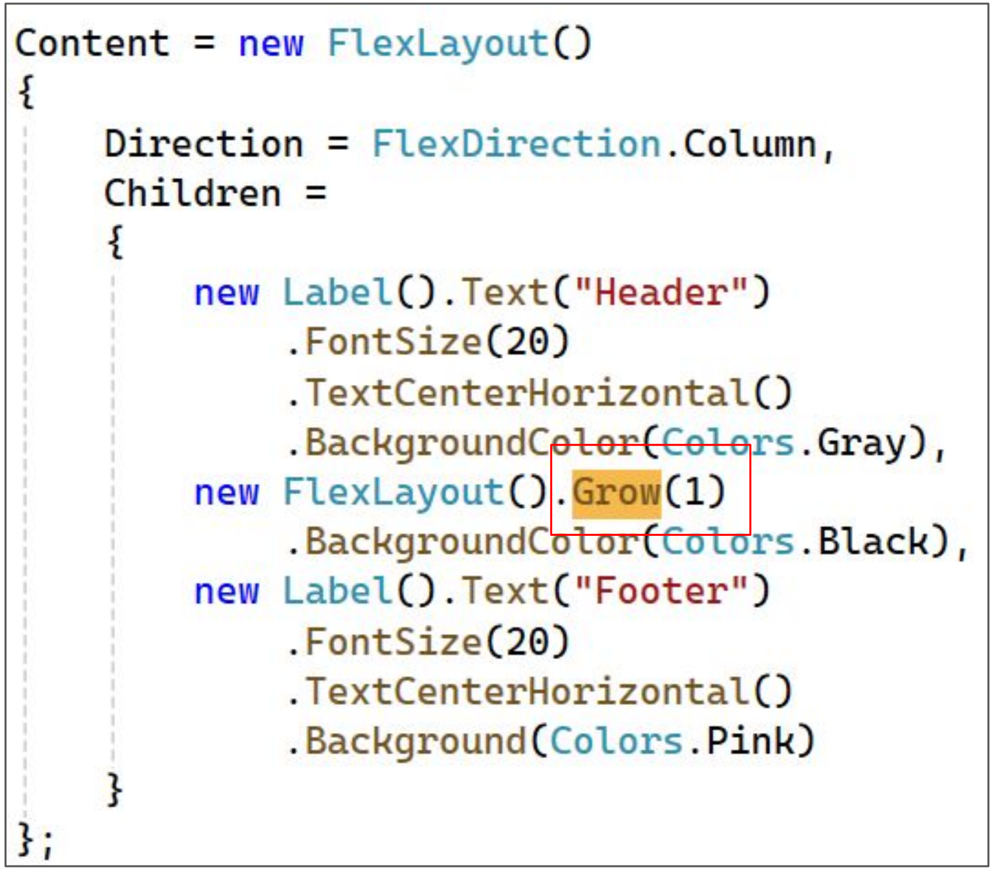


```
Content = new FlexLayout()
{
    Wrap = FlexWrap.Wrap,
    JustifyContent = FlexJustify.Start,
    AlignContent = FlexAlignContent.Start,
    Children =
    {
        new BoxView().BackgroundColor(Colors.Red)
            .Width(100).Height(100),
        new BoxView().BackgroundColor(Colors.Green)
            .Width(100).Height(100),
        new BoxView().BackgroundColor(Colors.Blue)
            .Width(100).Height(100),
```

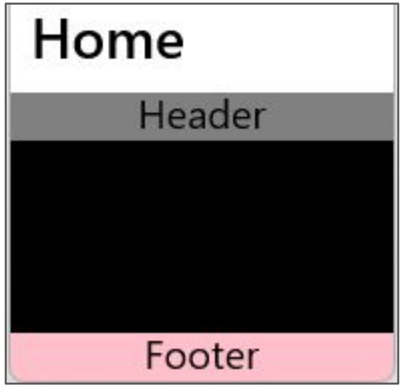


Tip: Add **ScrollView** for scrolling effect

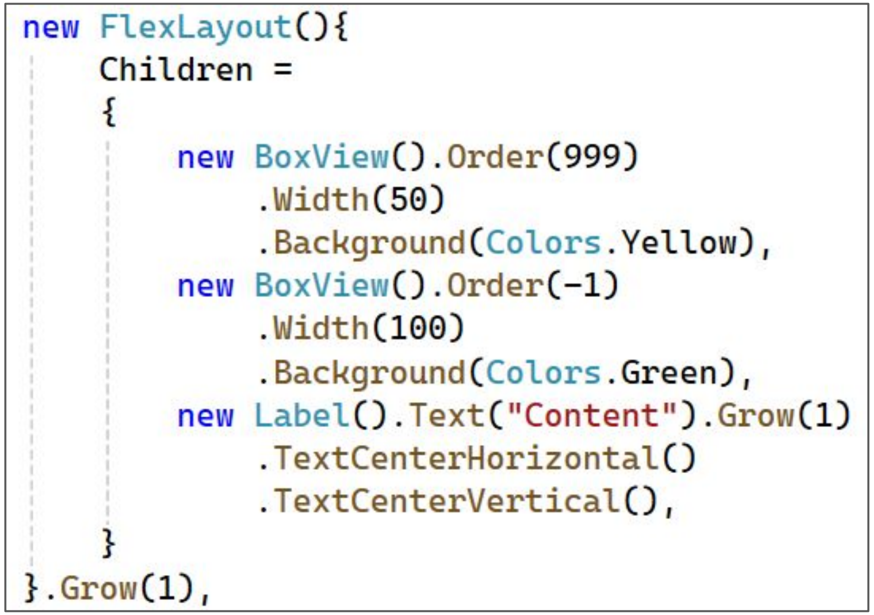
Usage 02 - Page layout: Holy grail - Step 01



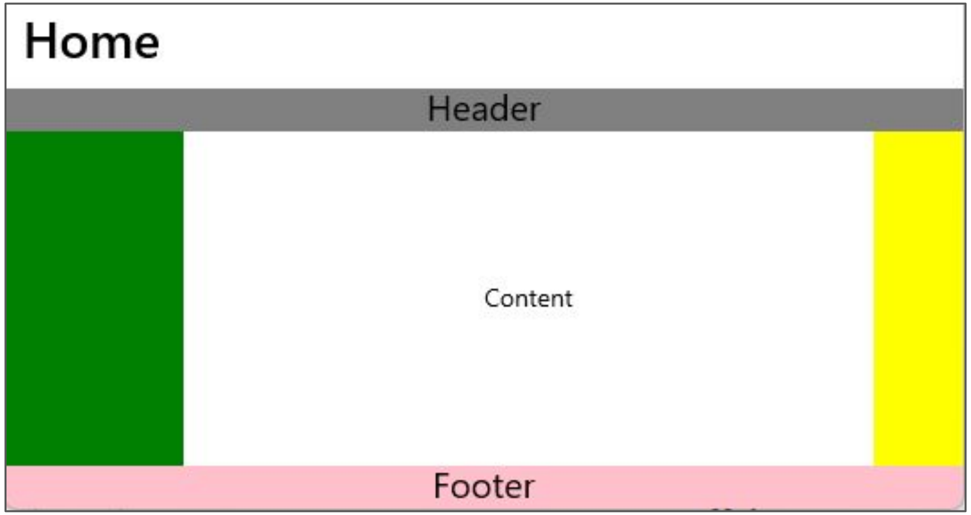
```
Content = new FlexLayout()
{
    Direction = FlexDirection.Column,
    Children =
    {
        new Label().Text("Header")
            .FontSize(20)
            .TextCenterHorizontal()
            .BackgroundColor(Colors.Gray),
        new FlexLayout().Grow(1)
            .BackgroundColor(Colors.Black),
        new Label().Text("Footer")
            .FontSize(20)
            .TextCenterHorizontal()
            .Background(Colors.Pink)
    }
};
```



Step 02 - Full page layout

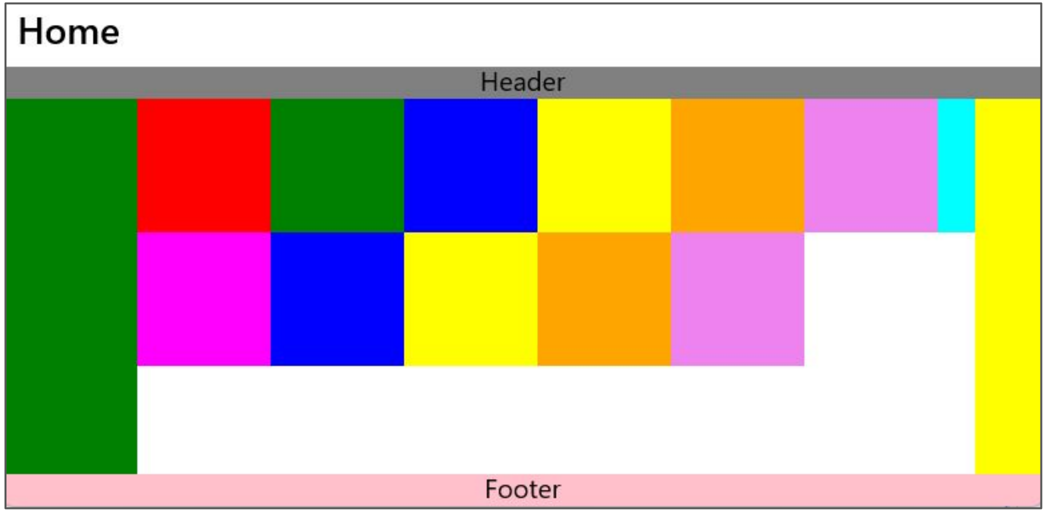


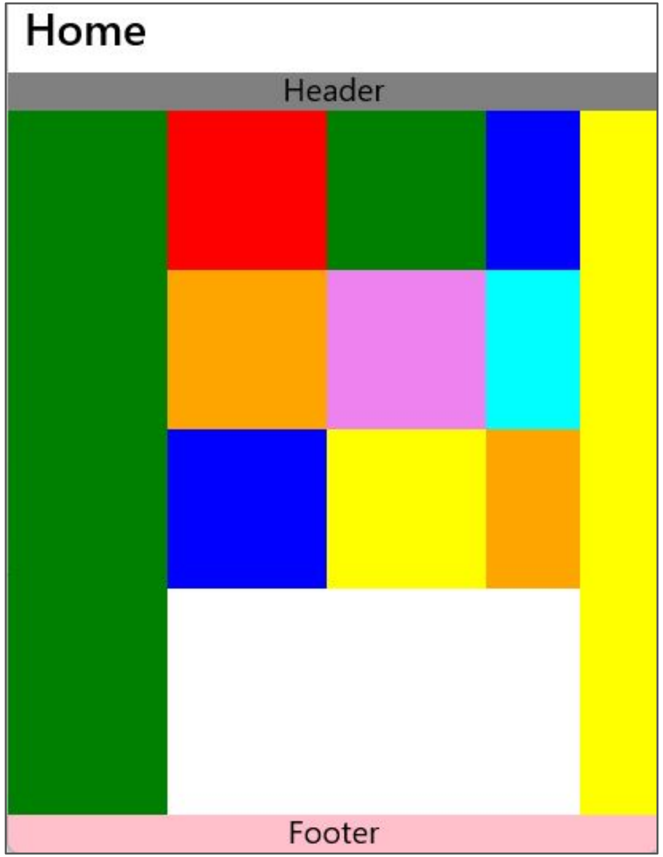
```
new FlexLayout(){
    Children =
    {
        new BoxView().Order(999)
            .Width(50)
            .Background(Colors.Yellow),
        new BoxView().Order(-1)
            .Width(100)
            .Background(Colors.Green),
        new Label().Text("Content").Grow(1)
            .TextCenterHorizontal()
            .TextCenterVertical(),
    }
}.Grow(1),
```



Basis: Initial size before dynamic resizing

Step 03 - Dynamic content





Useful UI control

Some buttons

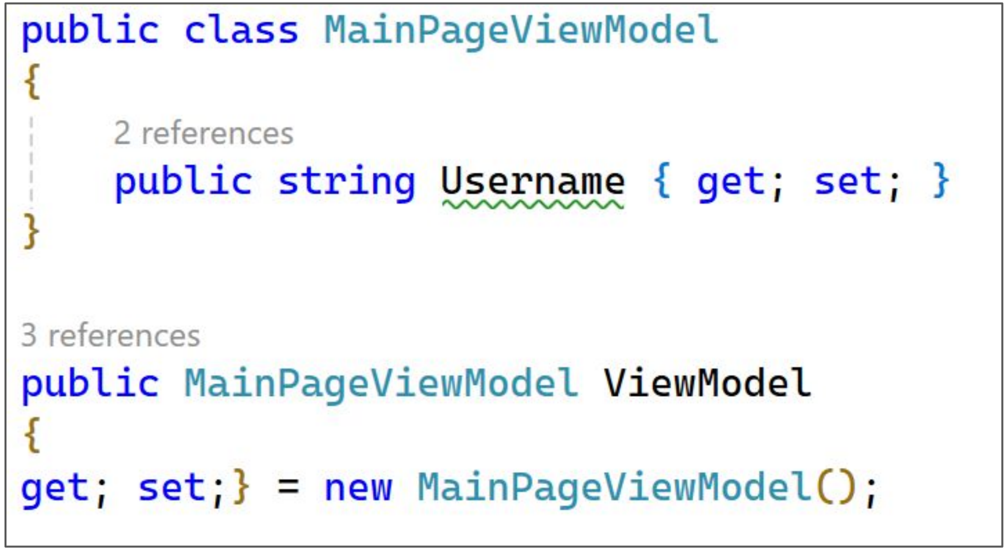
- ❑ Switch
- ❑ Slider
- ❑ Progress
- ❑ Date Time picker
- ❑ CheckBox / RadioButton / GroupBox

List of items

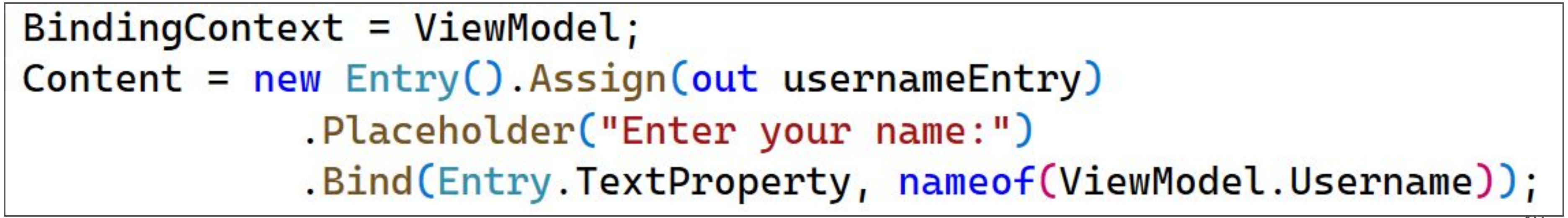
- ComboBox
- ListBox
- ListView
- TreeView
- DataGridView

Data binding

Basic data binding



```
public class MainPageViewModel
{
    2 references
    public string Username { get; set; }
}

3 references
public MainPageViewModel ViewModel
{
get; set;} = new MainPageViewModel();
```



```
BindingContext = ViewModel;
Content = new Entry().Assign(out usernameEntry)
            .Placeholder("Enter your name:")
            .Bind(Entry.TextProperty, nameof(ViewModel.Username));
```