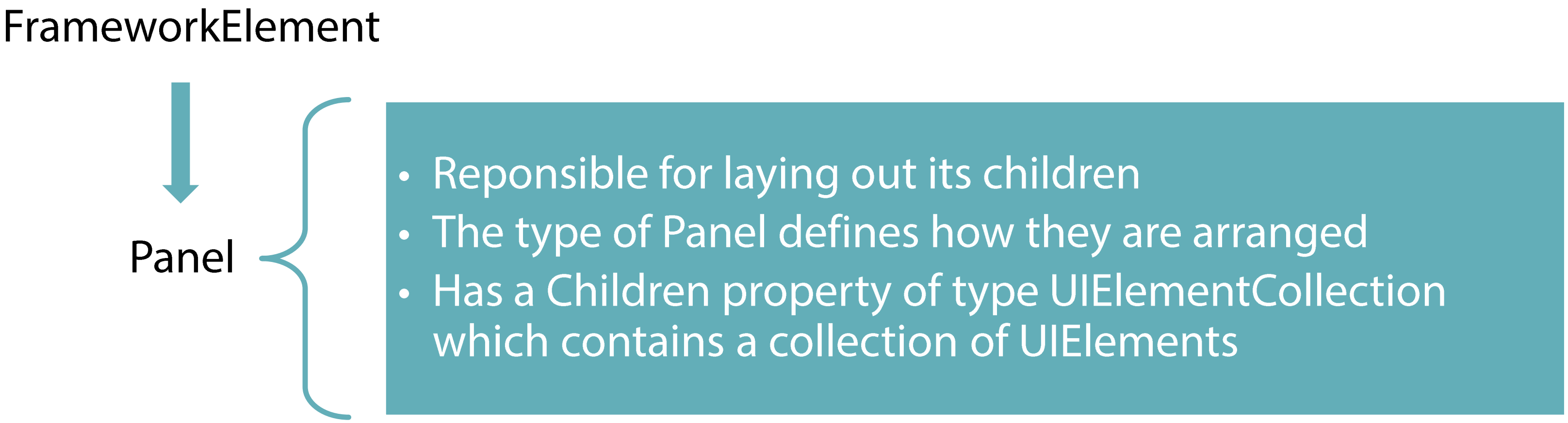
Panels and the Layout System

Kevin Dockx
http://blog.kevindockx.com/
@KevinDockx

pluralsight
hardcore dev and IT training

Panels

Panels allow us to arrange a group of child elements in simple to very complex layouts.



FrameworkElement

Panel

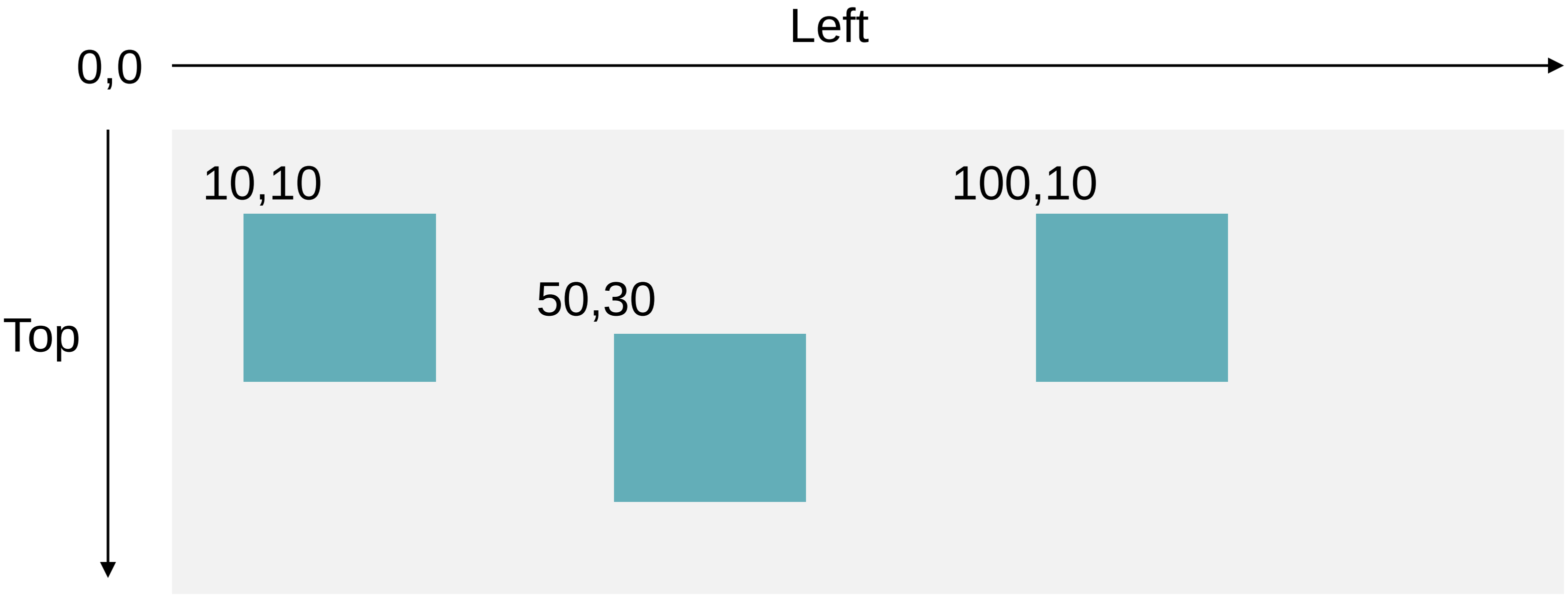
- Reponsible for laying out its children
- The type of Panel defines how they are arranged
- Has a Children property of type UIElementCollection which contains a collection of UIElements

Commonly used Panels: Canvas, StackPanel, Grid

Canvas

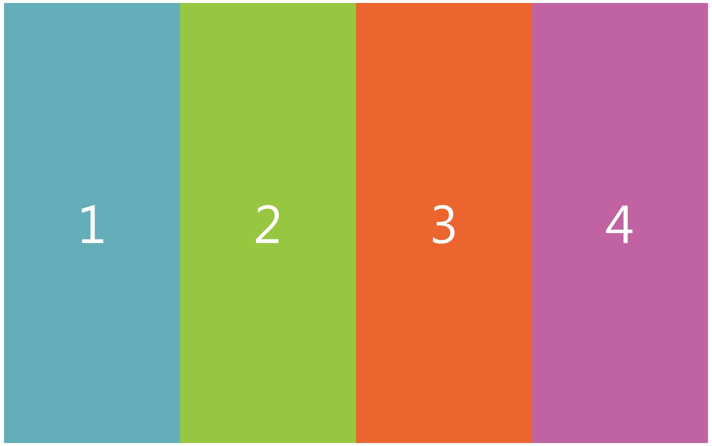
Used to explicitly position children relative to the Canvas area.

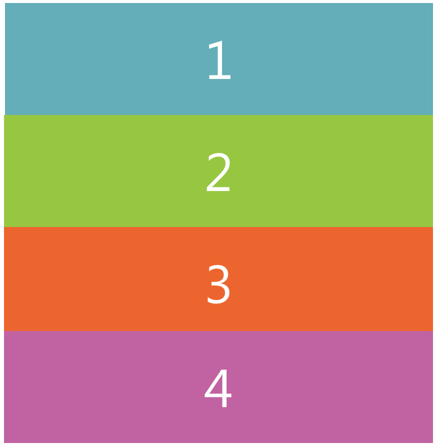
… but the absolute positioning means it’s not so good for scaling



StackPanel

Used to arrange children into a single line, horizontally or vertically.

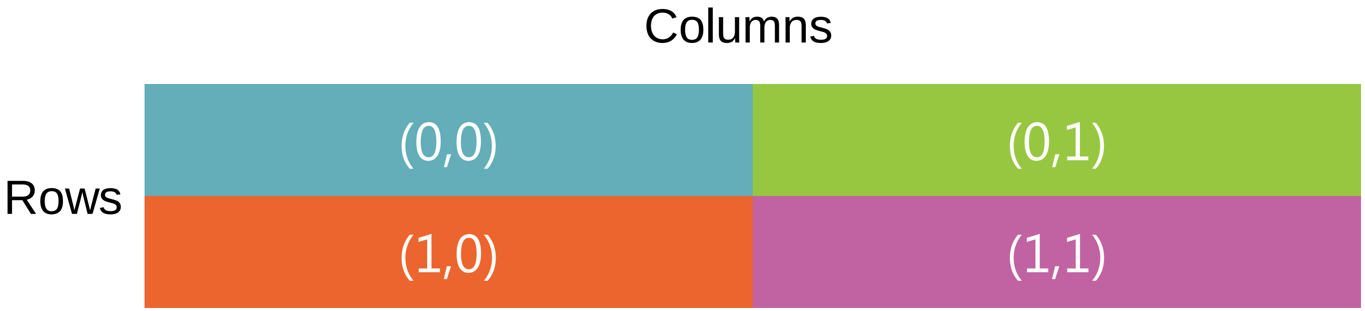




Grid

Defines a flexible grid area consisting of rows and columns.

… this great flexibility makes it one of the most-often used layout panels

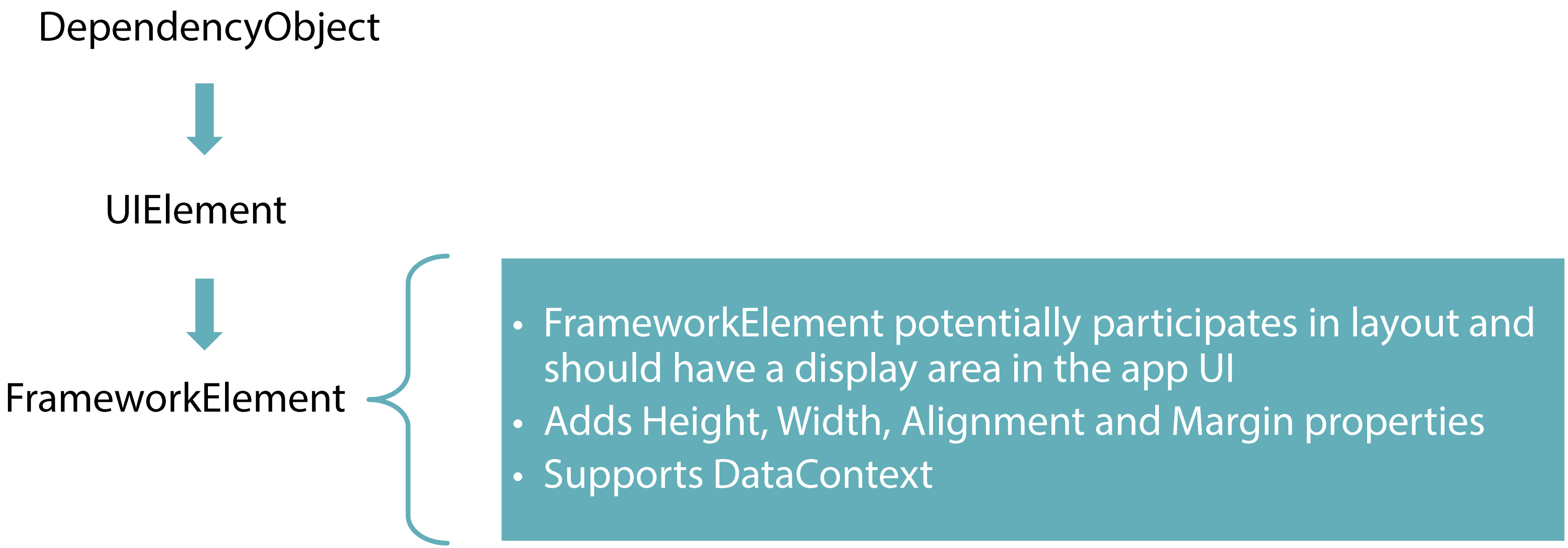


Columns

Rows

(0,0)	(0,1)
(1,0)	(1,1)

A Step Back to the FrameworkElement



DependencyObject

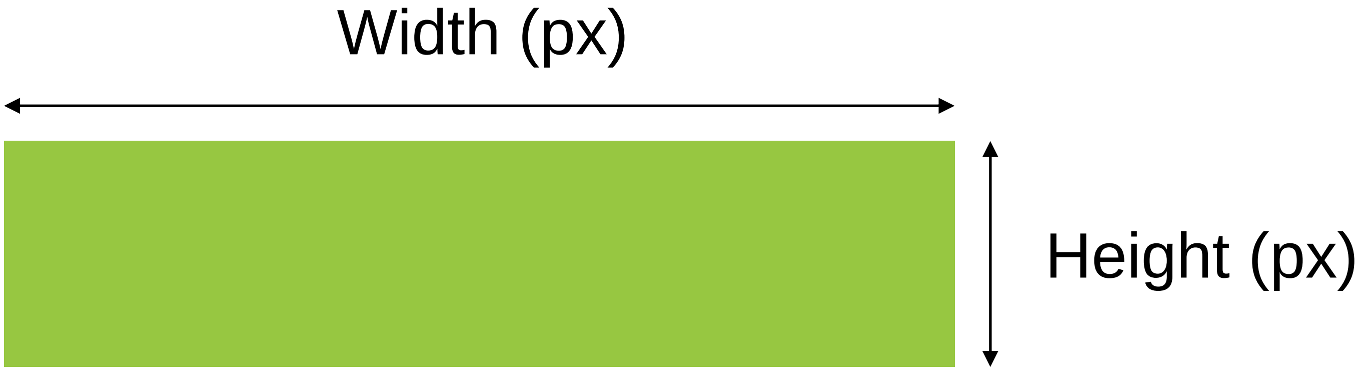
UIElement

FrameworkElement

- FrameworkElement potentially participates in layout and should have a display area in the app UI
- Adds Height, Width, Alignment and Margin properties
- Supports DataContext

Width and Height

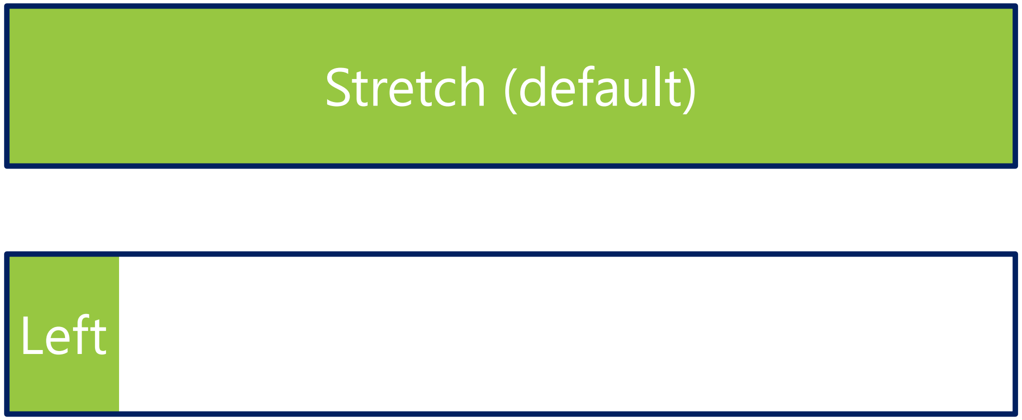
Define how wide, respectively high an element should be.



HorizontalAlignment

Defines how the element is horizontally aligned within a parent element's allocated layout space.

- Valid values are Left, Right, Center, Stretch (default)



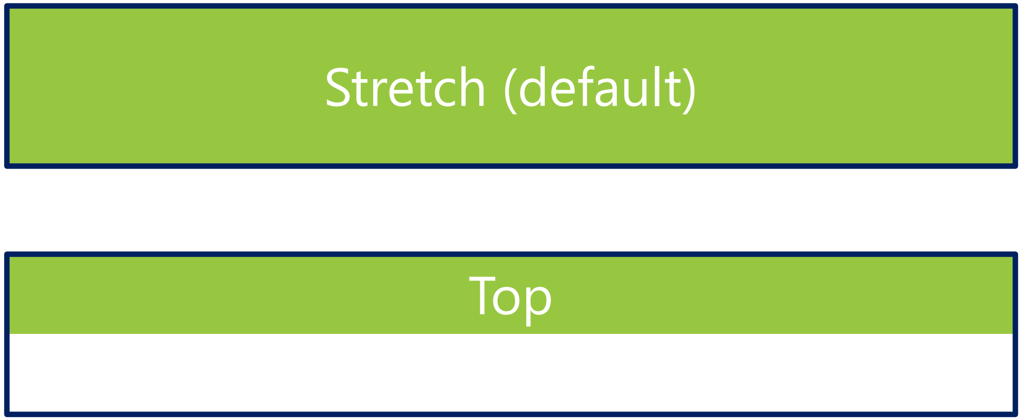
Stretch (default)

Left

VerticalAlignment

Defines how the element is vertically aligned within a parent element's allocated layout space.

- **Valid values are Top, Bottom, Center, Stretch (default)**



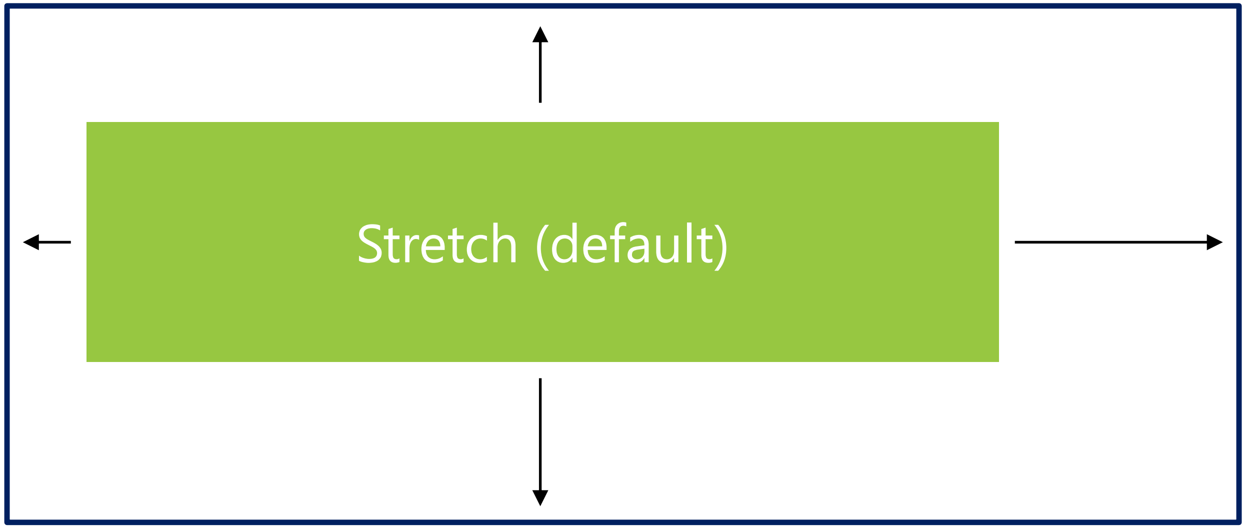
Stretch (default)

Top

Margin

Describes the distance between an element and its child or peers.

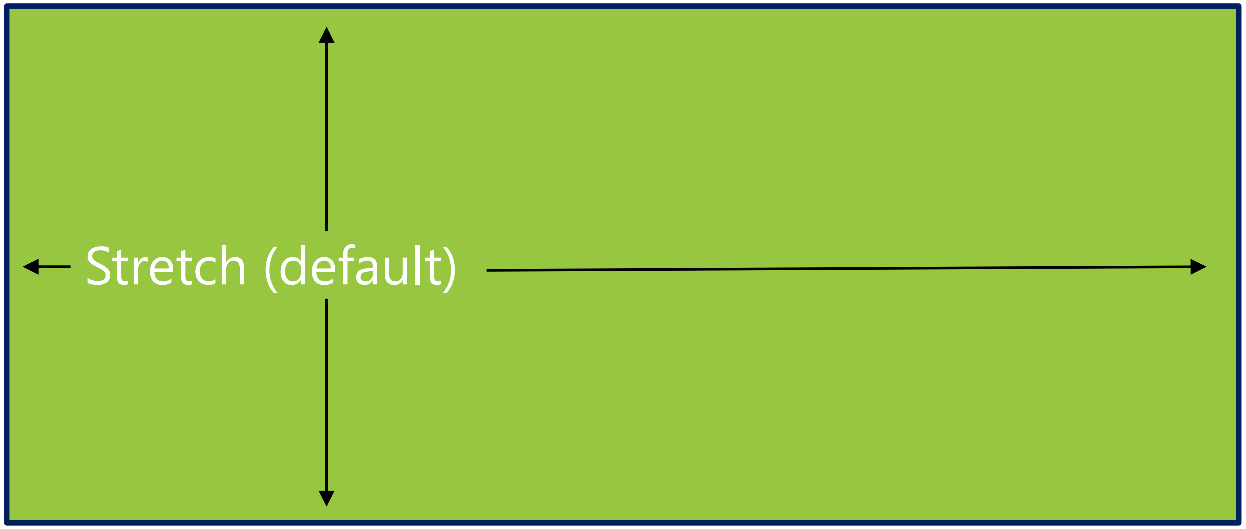
- OUTSIDE of the element
- Of type Thickness (left, top, bottom, right)



Padding

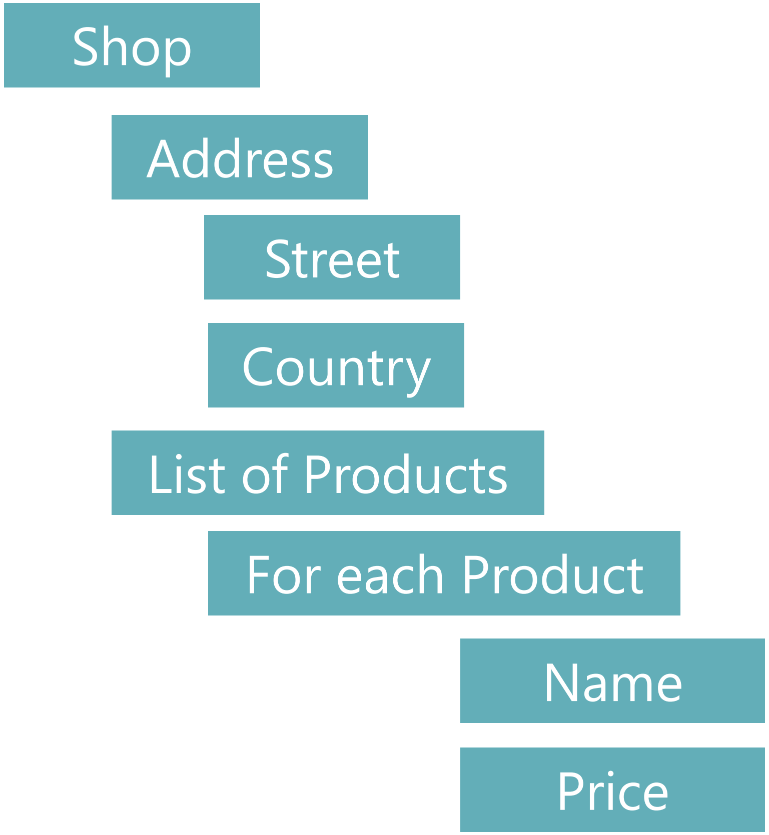
Enlarges the element by the provided Thickness.

- INSIDE of the element
- Of type Thickness (left, top, bottom, right)



An Important Concept: The Visual Tree

But first… the object tree. This defines how objects that are created and exist at run-time are related to each other.



Shop

Address

Street

Country

List of Products

For each Product

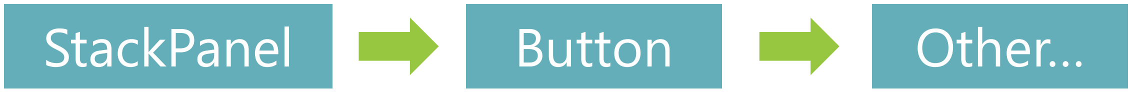
Name

Price

From Object to Visual Tree

The visual tree is a filter on the object tree, only containing objects that have a rendering implication.

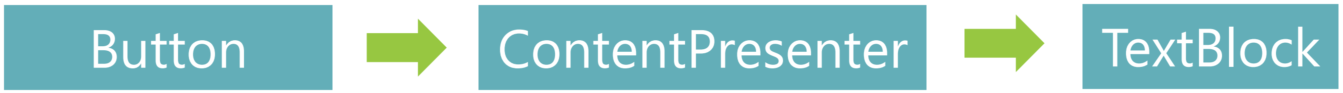
```
<StackPanel>
    <Button Content="Hi there!" />
</StackPanel>
```



StackPanel → Button → Other...

The Visual Tree

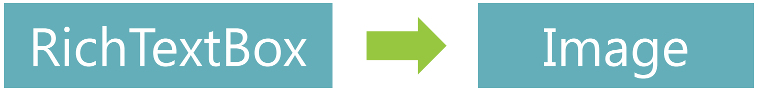
```
<Button>
    <TextBlock Text="Hi there!" />
</Button>
```



```
<Button.Template>
    <ControlTemplate TargetType='Button'>
        <ContentPresenter ... />
    </ControlTemplate>
</Button.Template>
```

The Visual Tree

```
<RichTextBox>
    <FlowDocument>
        <Paragraph>
            Hello <Image ... />
        </Paragraph>
    </FlowDocument>
</RichTextBox>
```



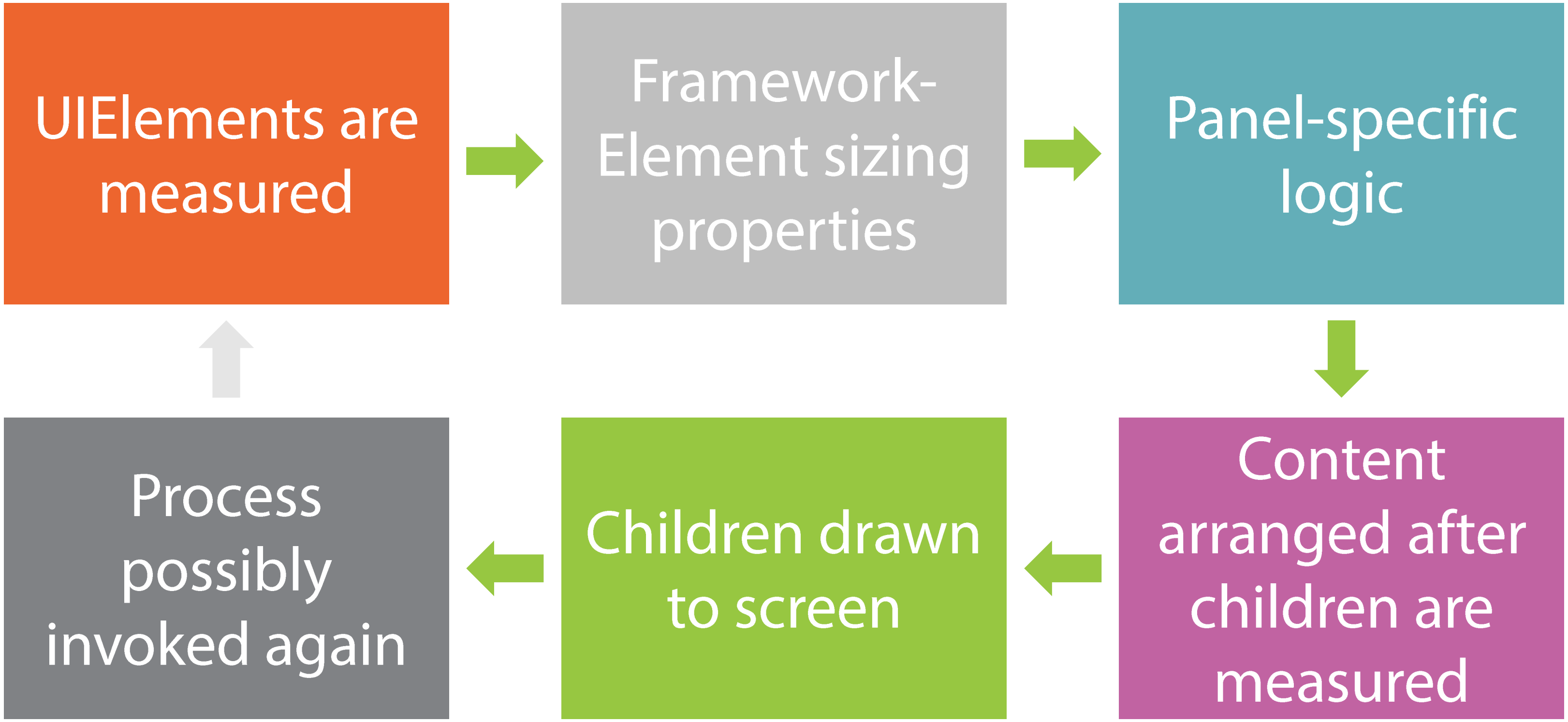
The visual tree can be seen as a scene graph, containing all the rendering information needed to compose the output to the display device.

It determines the rendering order.

The Layout System

Ensures elements are sized, positioned and drawn, recursively.

It's the process of measuring and arranging a panel's children



UIElements are measured

Framework-Element sizing properties

Panel-specific logic

Content arranged after children are measured

Children drawn to screen

Process possibly invoked again

Summary

- **Panels arrange child elements**
    - Canvas, StackPanel, Grid

- **FrameworkElement properties**
    - Width, Height, Alignment, Margin
    - Padding

- **Visual Tree**
- **Layout System**