



*ESRI DEVELOPER SUMMIT 2023*

# Design Apps with Style and Layout

Kevin Gonzago

# Agenda

- Page Types
- Layouts Types
- Size and Position
- Mobile Optimization

# Page

# Scrolling vs Fullscreen

```

final Layer = view.findViewById<Layer>()
view.findViewById<Layer>()
.then { layerView: LayerView? } {
    // If there were problems with
    // attaching this widget
}

```

# Scrolling Page

- One long single page
- View one block at a time
- Scroll to navigate
- Content arrangement
  - Top down – block by block
  - Pre-configured templates
    - Block
    - Screen Group

# Fullscreen Page

- Single block page
- No scrolling navigation
- Scroll to navigate
- Content arrangement
  - Place widgets on a page just like a sticky note on a whiteboard
  - You determine the size and position of a widget
  - Widgets can overlap each other



```
let view = new ScrollView({
  content: "view",
  width: 100,
  height: 100,
  scroll: true,
  style: {
    backgroundColor: "red"
  }
});
```

[@ {

</STYLE>

# Scrolling page vs Fullscreen page Demo

```
const layerView = new LayerView({
  content: "layerView",
  width: 100,
  height: 100,
  scroll: true,
  style: {
    backgroundColor: "red"
  }
});
```

# Layouts

# Simple vs Compound

```

final Layer = view.findViewById<Layer>()
view.findViewById<Layer>()
.then { layerView: LayerView? -> {
    // If there were problems with
    // attaching this widget

```

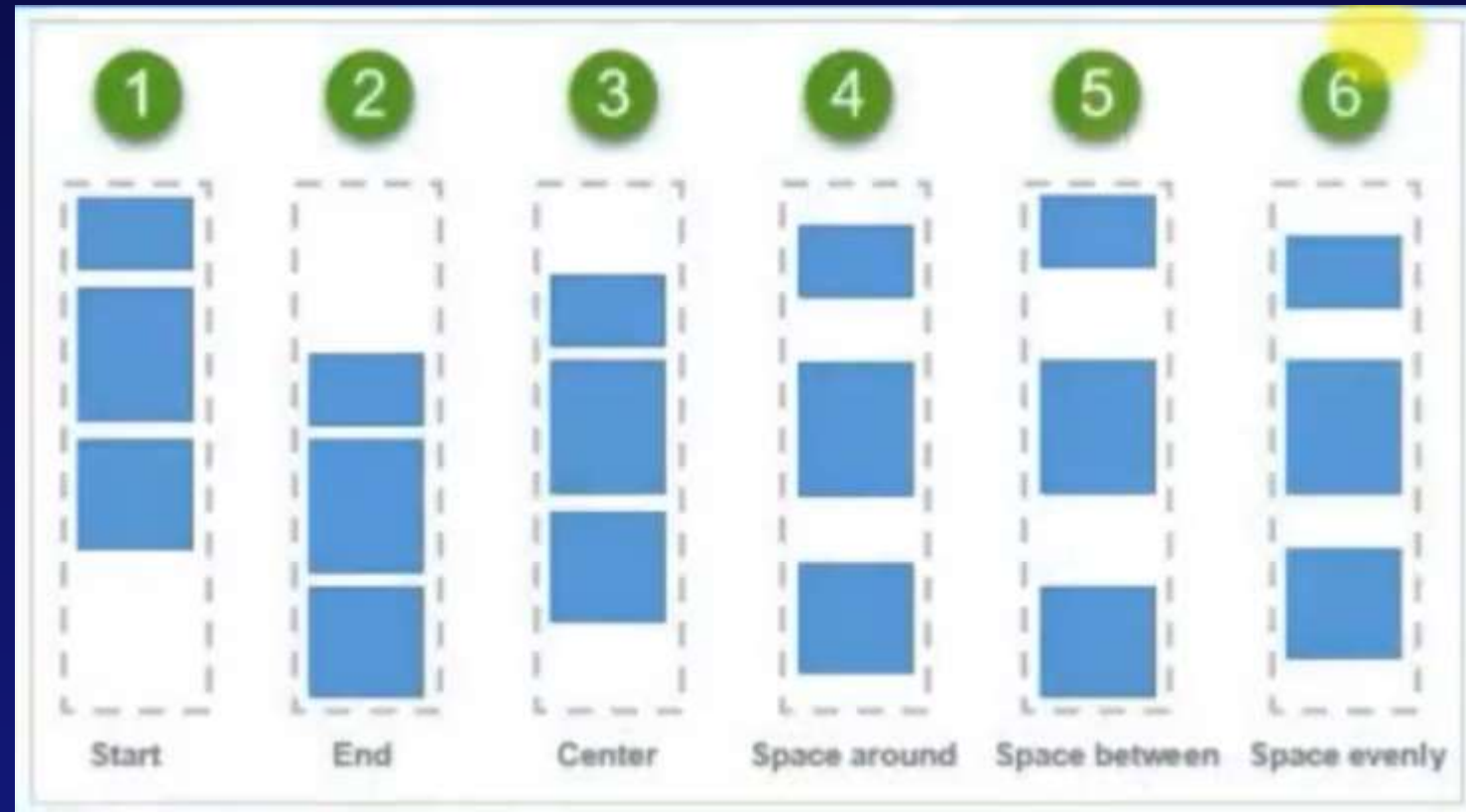
# Layout - Simple

- Ways to organize elements
  - Vertically
  - Horizontally
  - Randomly



# Layout - Simple

- Vertical organization – Column
  - Organizes content vertically as a single column
  - Vertical expands as more content is added to it
  - Each grid can manually be resized
  - Gap, padding and vertical alignment is adjustable



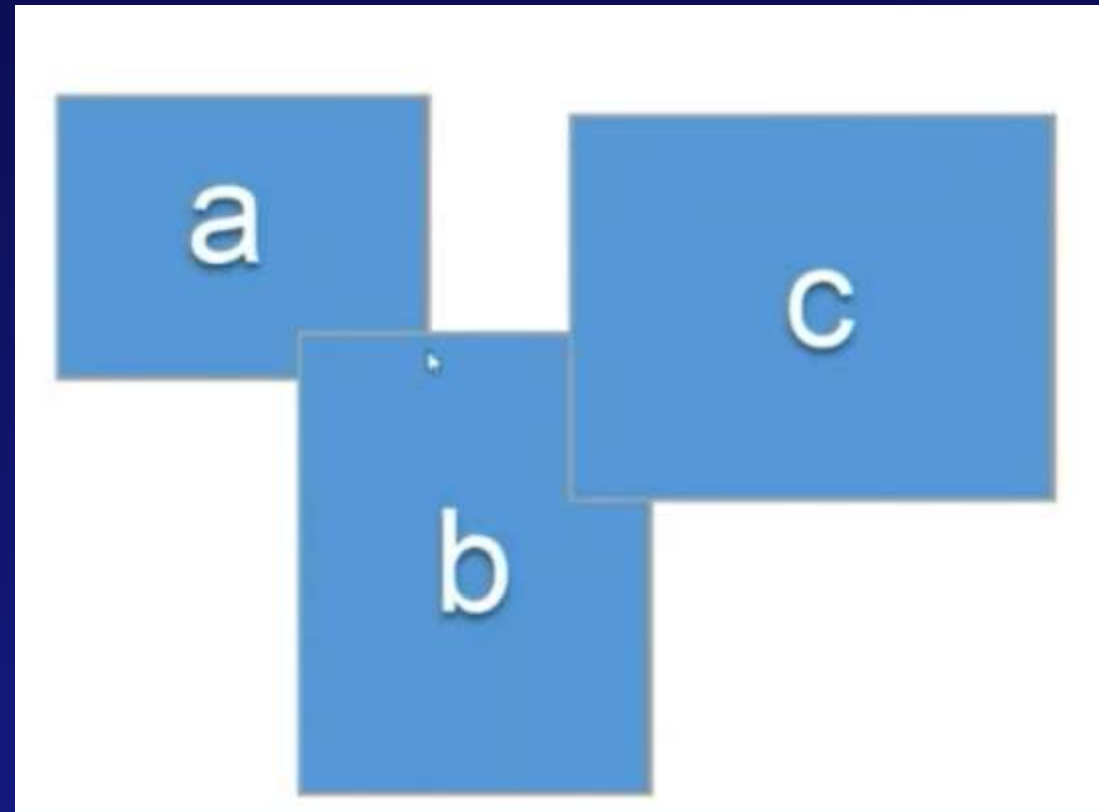
# Layout - Simple

- Horizontal organization – Row
  - Organizes content side-by-side in a row
  - Contains 12-column grids
  - All grids are equal in size
  - Grid size adjusts based on the gap and padding and settings
  - Widget snaps to the grid and can span across one or more grids



# Layout - Simple

- Random organization – Fixed panel
  - Randomly place widgets inside a Fixed panel
  - Like sticky notes on a whiteboard
  - Resize nested widgets with absolute or relative positions
  - Fullscreen page is like a large fixed panel



```
let view = new ScrollView({
  container: "viewport",
  map: map,
  style: {
    width: 100,
    height: 100,
    border: 1px solid black,
    border-radius: 10px,
    background-color: "white",
    padding: 10px,
    margin: 10px,
    box-shadow: 5px 5px 0px #ccc,
    transition: "transform 0.3s ease-in-out"
  }
});
```

[@ {

</STYLE>

# Simple Layout Demo

```
const map = new Map({
  layers: [
    {
      name: "Map",
      type: "Image",
      url: "https://www.mapbox.com/mapbox-gl-static/styles/streets-v8.json",
      style: {
        width: 100,
        height: 100,
        border: 1px solid black,
        border-radius: 10px,
        background-color: "white",
        padding: 10px,
        margin: 10px,
        box-shadow: 5px 5px 0px #ccc,
        transition: "transform 0.3s ease-in-out"
      }
    }
  ]
});

const view = new MapView({
  container: "viewport",
  map: map,
  style: {
    width: 100,
    height: 100,
    border: 1px solid black,
    border-radius: 10px,
    background-color: "white",
    padding: 10px,
    margin: 10px,
    box-shadow: 5px 5px 0px #ccc,
    transition: "transform 0.3s ease-in-out"
  }
});

// If you're using the layer view, you'll get an error here
const layerView = new LayerView({
  container: "viewport",
  map: map,
  style: {
    width: 100,
    height: 100,
    border: 1px solid black,
    border-radius: 10px,
    background-color: "white",
    padding: 10px,
    margin: 10px,
    box-shadow: 5px 5px 0px #ccc,
    transition: "transform 0.3s ease-in-out"
  }
});
```



# Layout - Compound

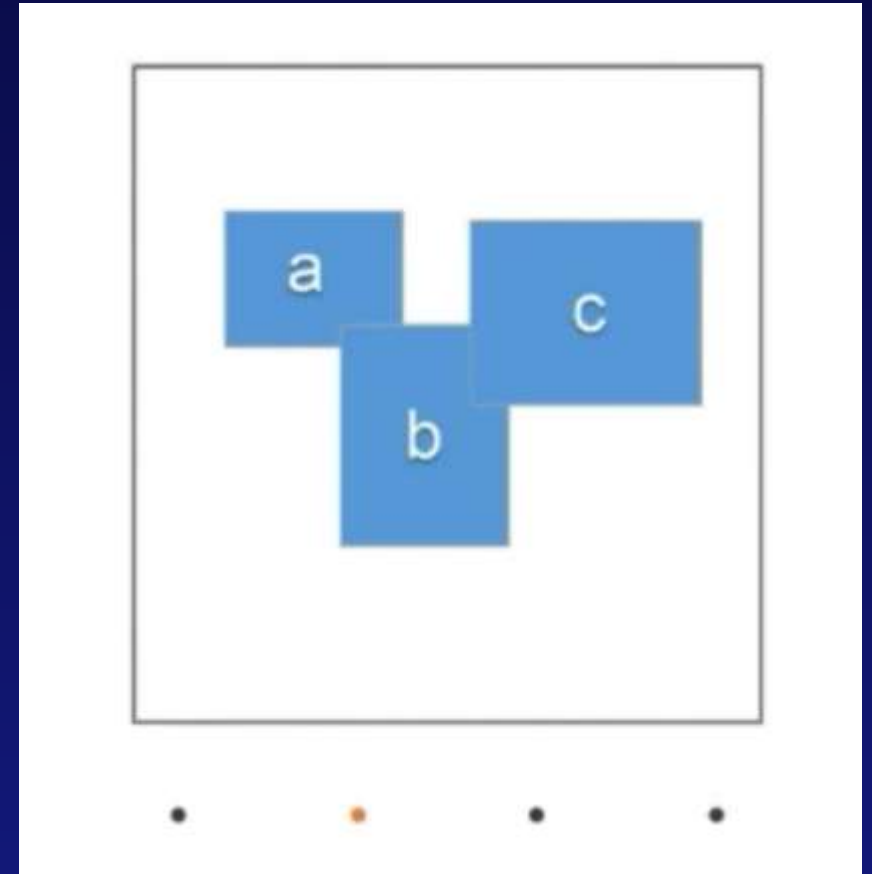
- Sidebar
  - Consists of two fixed panels
  - One panel is collapsible
  - Dock style is configurable



# Layout - Compound

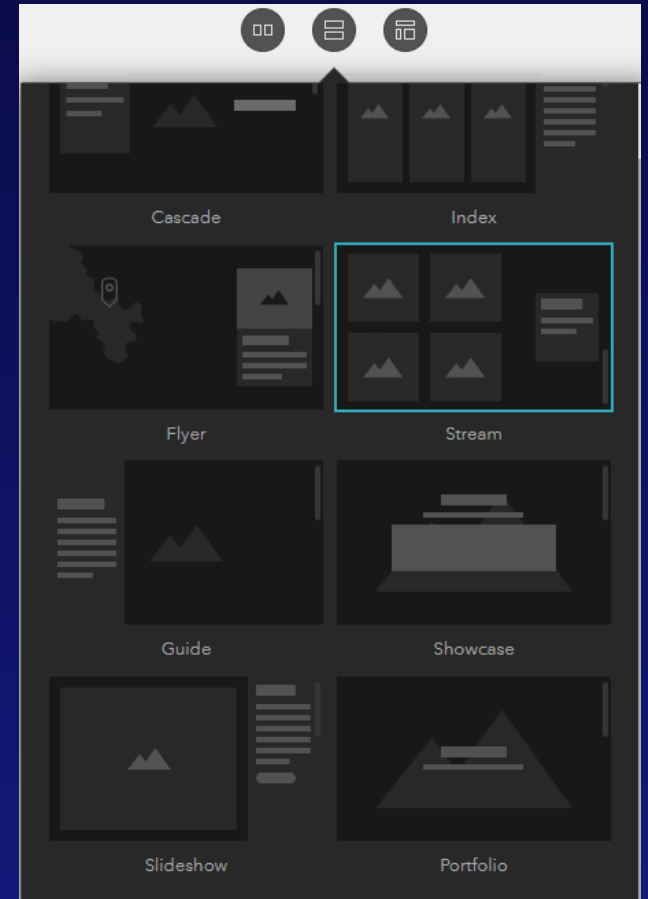
- Section

- Accommodates for multiple views
- Show one view at a time
- Navigate with Views Navigation
- Each view can include a combination of any widgets
- Random arrangement inside a view
- Section is like a stack of fixed panels



# Layout - Compound

- Screen Group
  - Multiple screens
  - Scrollable within the screen group
  - Each screen has a main stage
  - Each screen can also have a scrolling panel
  - Available in scrolling page only
  - Main stage remains in place until one full screen scrolls away
  - Scrolling panel is extendable
  - Scrolling height of a single screen based on the panel's content



```

public void show() {
    new SuccessiveLower() {
        public void show() {
            System.out.println("Successive Lower");
        }
    }.show();
}

```

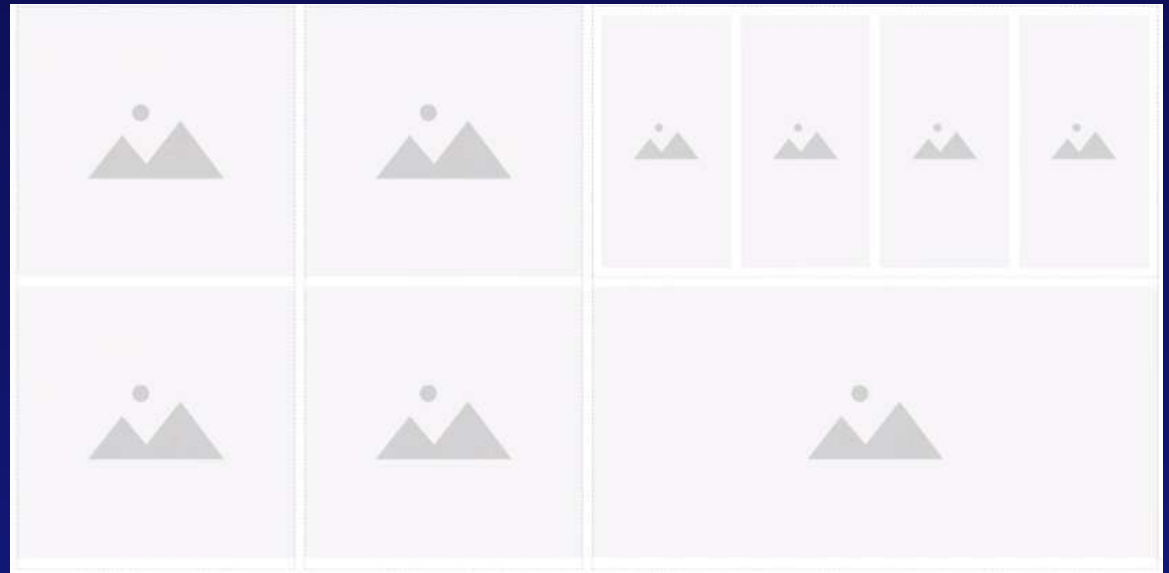
# Compound Layout Demo

```
console.log(layer.getViewAt(index));
// ...
view.getViewAt(index)
// ...
// console.log(layerView)
// If you try to access the layerView with the layerview, you'll get an error here
call
```



# Layout - Composite

- Combine different layout: put one into another
- In one layout, you may also have the power of another



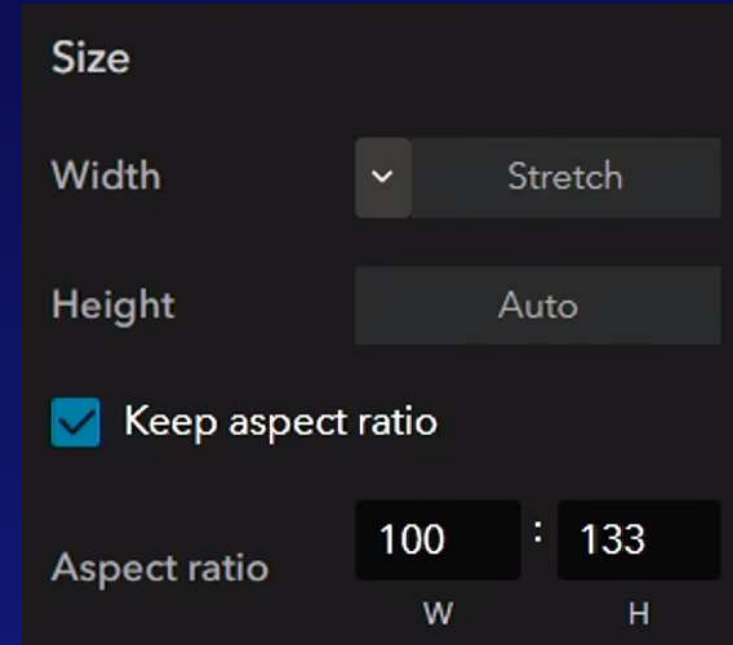
# Size and Position - Alignment

- Alignment with the parent container
- Widget's position determined by the align origin and offsets
- Pixel
  - Absolute
  - Not responsive during resize event
- Percentage
  - Relative positioning
  - Responsive during the resize event



# Size and Position – Width and Height modes

- Four types
  - Custom – Set specific values
  - Auto – automatic based on the contained content
  - Stretch – fulfill based on the size of its container
  - Aspect ratio – keep the ratio of width and height



A screenshot of a configuration panel for size and position. The panel has a dark background with light text. It includes sections for 'Size', 'Width', 'Height', a checkbox for 'Keep aspect ratio', and an 'Aspect ratio' section with input fields for width and height.

Size

Width ▼ Stretch

Height Auto

☒ Keep aspect ratio

Aspect ratio  :   
W H

```

    Kirk view = new ScatterView(
        controller: "viewOnly",
        model: model,
        viewFormat: "j"
    );
    @Html.RenderViewAsync(
        new ViewDataDictionary {
            "ViewName" = "ScatterViewOnly"
        },
        view
    );
}

```

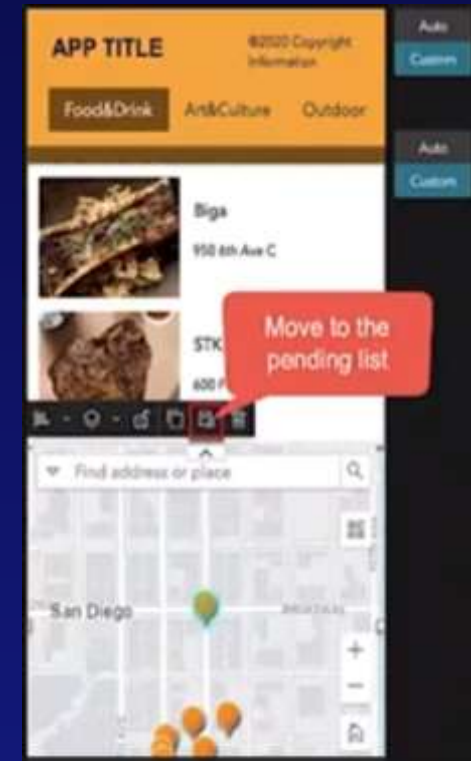
# Size and Position Demo

```
console.log(layer.getViewAt(index));
// ...
view.getViewAt(index)
// ...
// console.log(layerView)
// If you try to access the layerView with the layerView, you'll get an error here
call
```



# Mobile Optimization

- Configure your app for mobile form factor separate from the desktop experience
  - Change behavior of widgets, such as
    - Turn off tools on the map
    - Change the font color and size
  - Use less or different widgets
  - Change the layout of your app
  - Move widgets to the pending list instead of deleting them
    - These widgets will be available for the desktop version of your app



```
let view = new ScrollView({
  container: "viewport",
  map: map,
  style: {
    height: 100,
    width: 100,
    backgroundColor: "white"
  }
});
```

[@ {

</STYLE>

# Mobile Demo

```
const layerView = new LayerView({
  map: map,
  style: {
    height: 100,
    width: 100,
    backgroundColor: "white"
  }
});
```

# Mobile Optimization

## 7 tips for responsive design

1. Use structural layout widgets to organize content (Row, Column, Sidebar, etc.)
2. If fixed layout is used, be aware of sizing, positioning, alignment and offset
3. Design your app content first then customize the layout
4. Change how widgets will be displayed
  - A. Example: use smaller font size or icons for widgets
5. Move unneeded widgets to pending list
6. Preview all available screen sizes
7. **Recommendation: Use one of the ExB templates for convenience**





esri®

THE  
SCIENCE  
OF  
WHERE®

Copyright © 2023 Esri. All rights reserved.

</SCRIPT>

LIVE  
BY  
THE  
CODE }