TDDC73 - LAYOUT, INTERACTION, AND CODE-ORGANIZATION

LiU EXPANDING REALITY

Agenda

- Questions
- Lab-Assistants
- Grouping Code (Framework/API/SDK)
- Layout
  - Android - standard and Android - compose
  - React-Native Flexbox
  - Flutter
- Manage interaction
  - Callbacks/Listeners
- Structuring your code Model-View-Whatever
    - MVC ,MVP ,MVVM
- Demo

Assistants

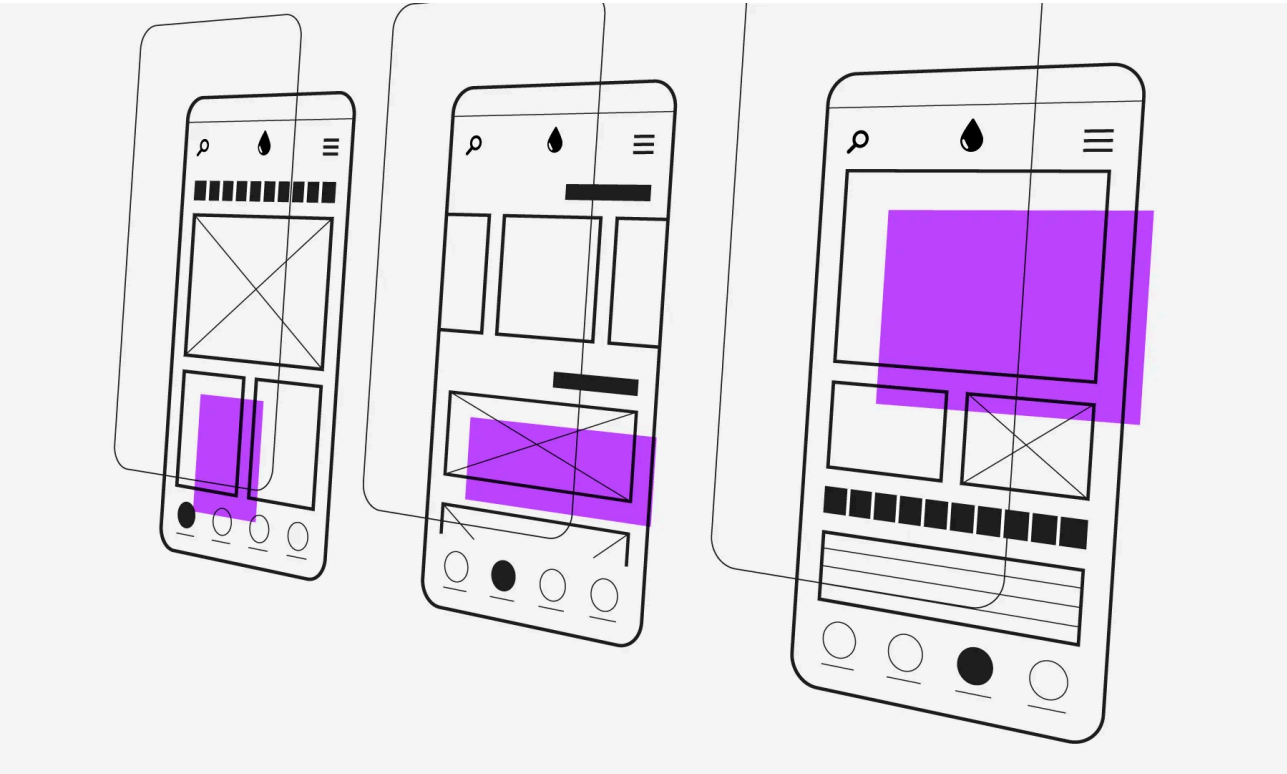
- Two lab assistants
  - Scheduled for the SU-rooms
  - Ask for meetings in Teams (email your assistant)

Framework

- Fuctions/Classes/Objects
  - Libraries
    - Framework
- SDK Software Development Kit
  - We add some tools

Layouts

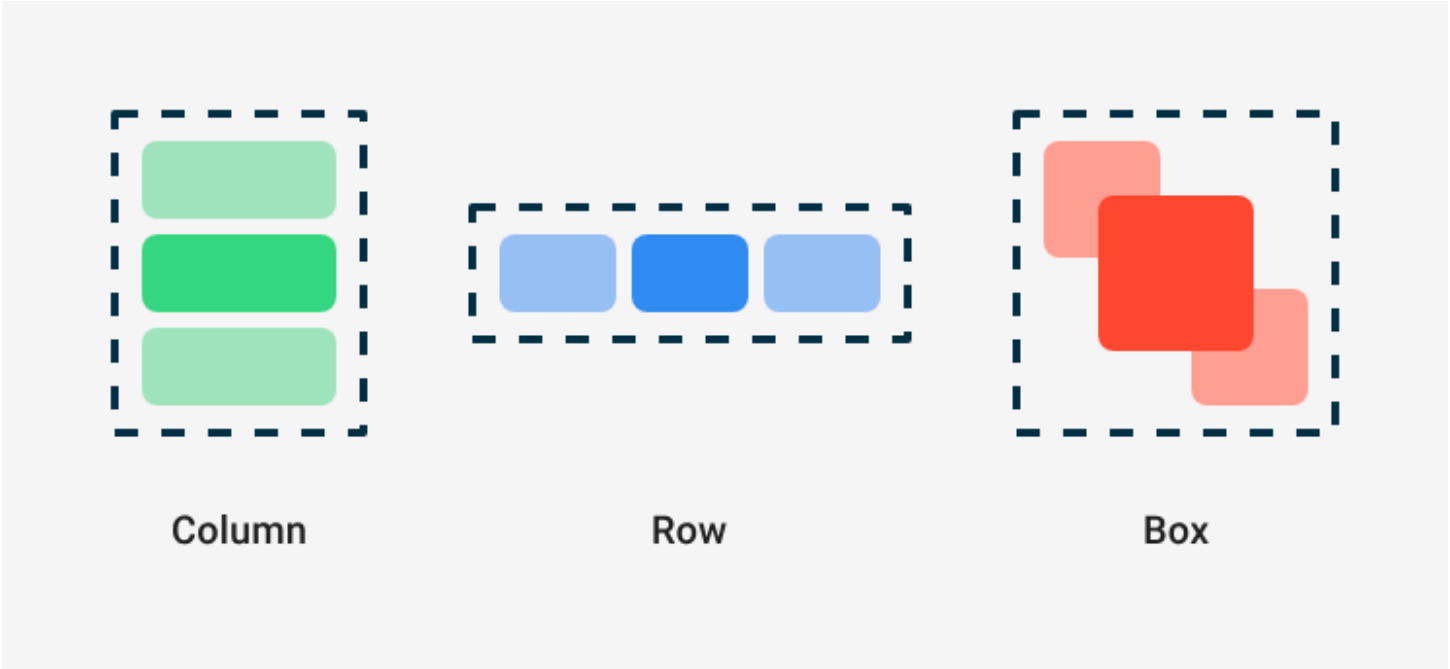
- Putting stuff in the right place
- Keeping things in their proper place
- Make shit look and hopefully work nice



Standard Android layouts

- ViewGroup - a View that supports adding child views
  - LinearLayout
    - Left-to-Right or Top-to-Bottom
  - FrameLayout
    - Designed for a single View child
  - RelativeLayout
    - Views placed in relation to each other or the parent
  - GridLayout
    - Rows and Columns
  - ConstraintLayout
    - RelativLayout on steroids
  - MotionLayout
    - ConstraintLayout on steroids

Layout in Compose



Column

```
@Composable
fun ArtistCardColumn() {
    Column {
        Text("Alfred Sisley")
        Text("3 minutes ago")
    }
}
```

Row

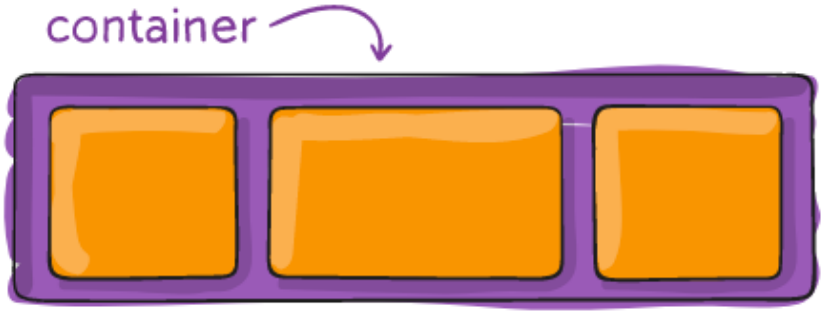
```
@Composable
fun ArtistCardRow(artist: Artist) {
    Row(verticalAlignment = Alignment.CenterVertically) {
        Image(bitmap = artist.image, contentDescription = "Artist image")
        Column {
            Text(artist.name)
            Text(artist.lastSeenOnline)
        }
    }
}
```

Alfred Sisley

3 minutes ago

Box

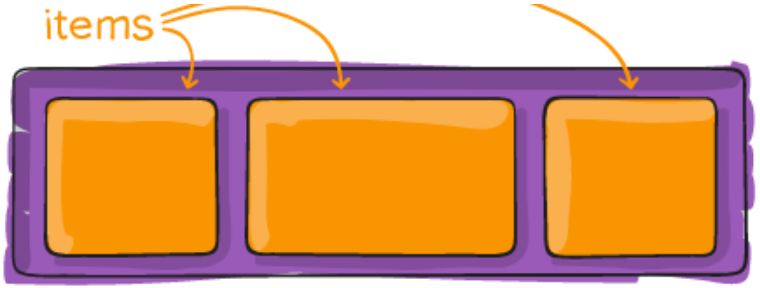
```
@Composable
fun ArtistAvatar(artist: Artist) {
    Box {
        Image(bitmap = artist.image, contentDescription = "Artist image")
        Icon(Icons.Filled.Check, contentDescription = "Check mark")
    }
}
```



Flexbox layout

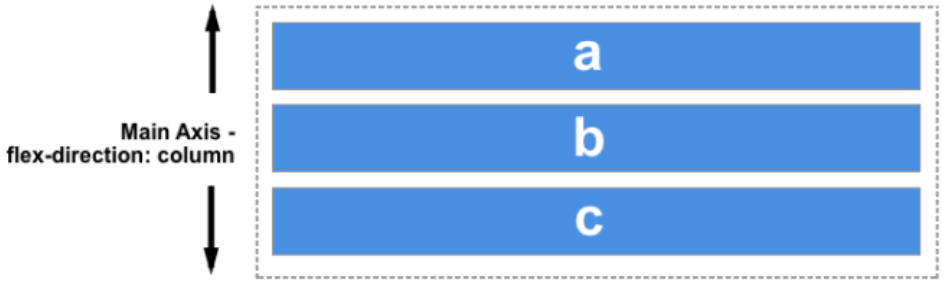
- Multiple properties are used together in order to use Flexbox layout
- Some properties are set on the container (parent, flex container)
- Some properties are set on the elements in the container (children, flex items)

- https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout
- https://css-tricks.com/snippets/css/a-guide-to-flexbox/

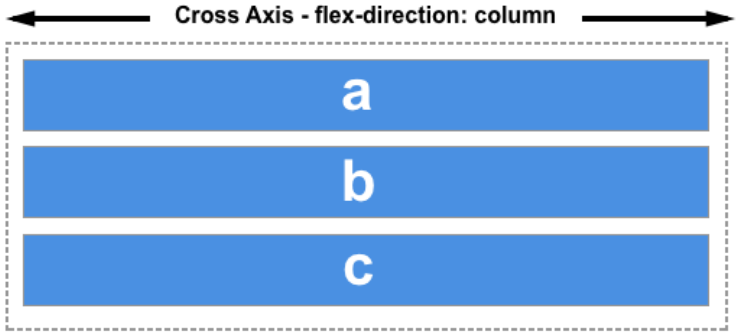


Main Axis and Cross Axis

Choose `column` or `column-reverse` and your main axis will run top to bottom of the page in the block direction.



If your main axis is `column` or `column-reverse` then the cross axis runs along the rows.

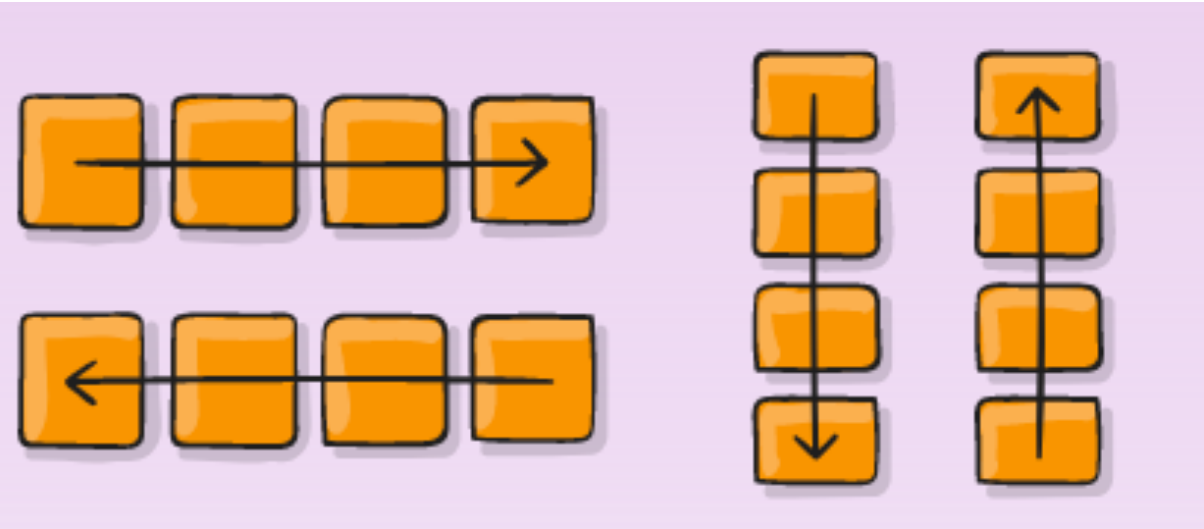


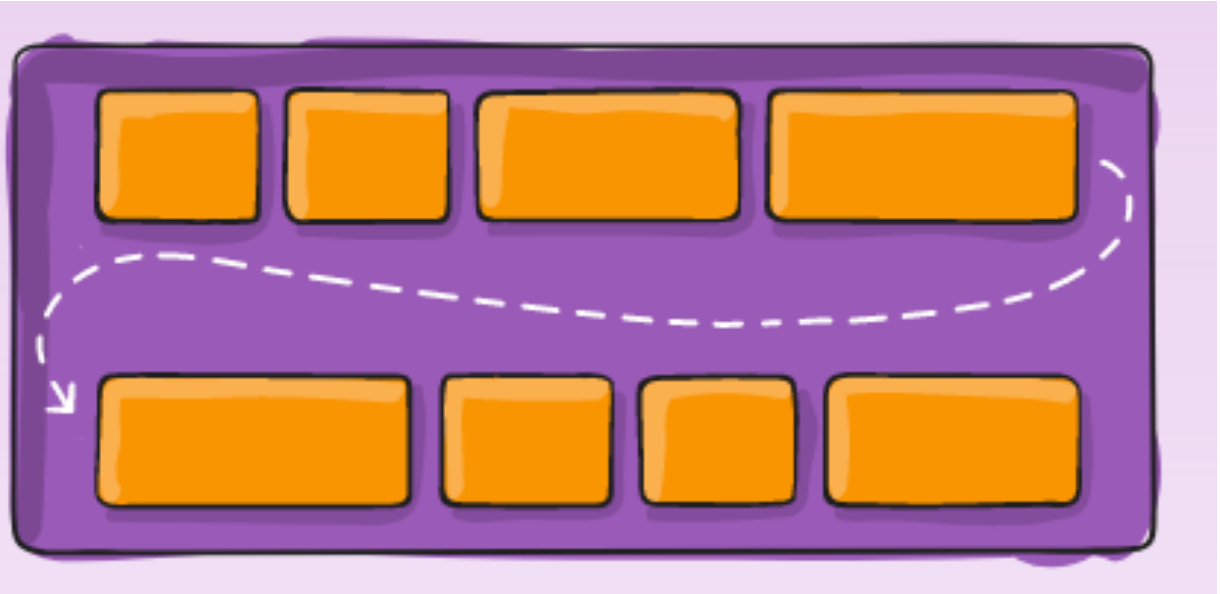
Some parent container properties

- flex-direction: row | row-reverse | column | column-reverse
- flex-wrap: nowrap | wrap | wrap-reverse
- justify-content: flex-start | flex-end | center | space-between
- align-items: stretch | flex-start | flex-end | center | baseline
- flex-flow: <flex-direction> <flex-wrap>

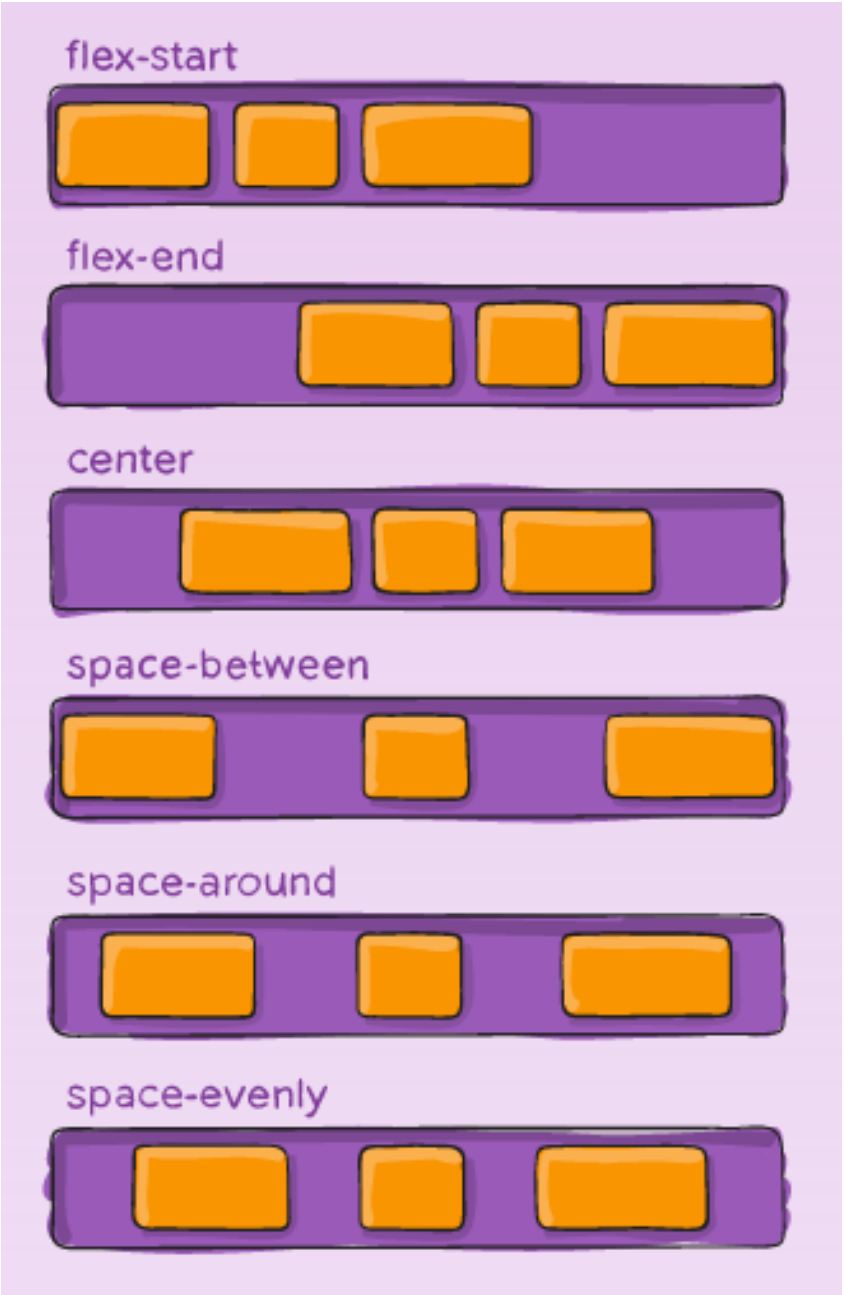
flex-direction and flex-wrap

-

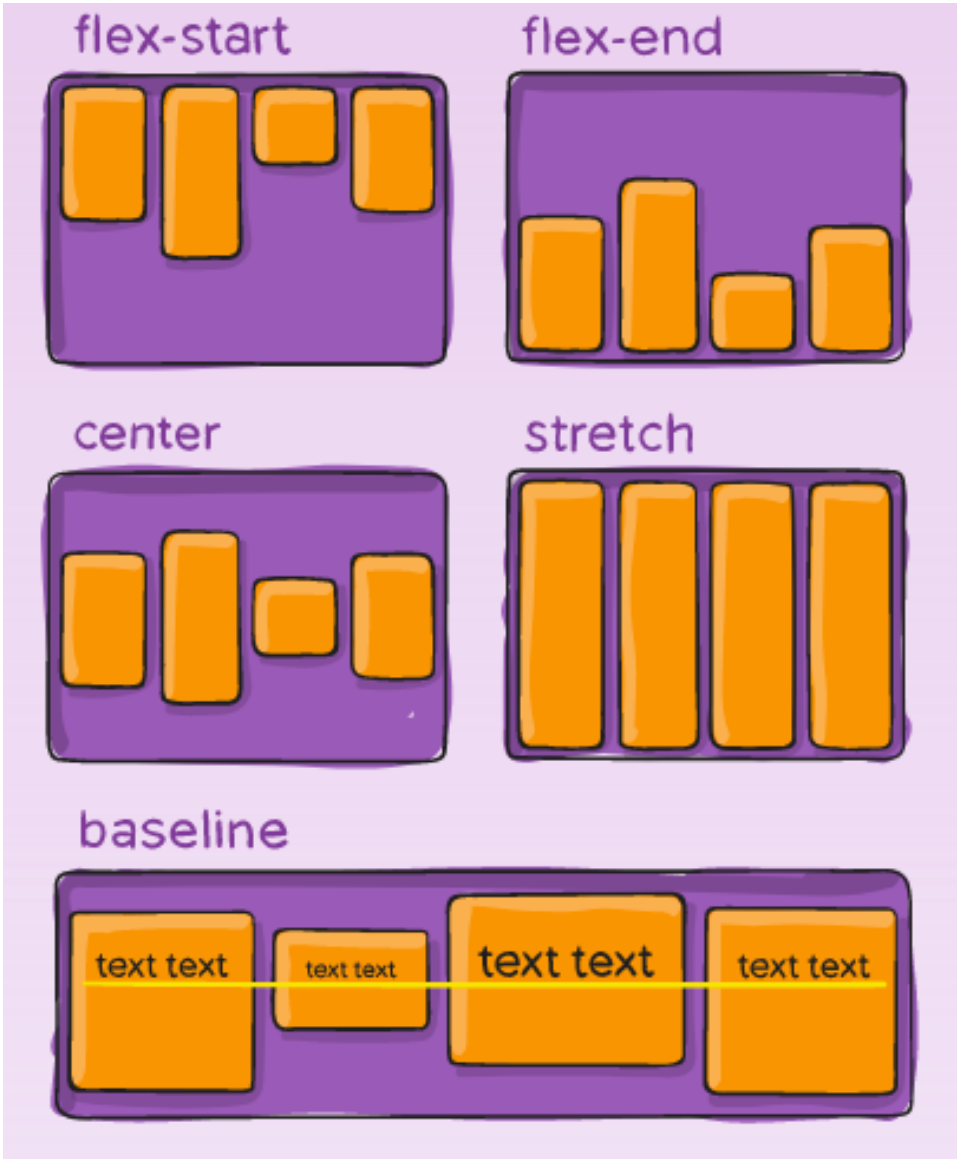


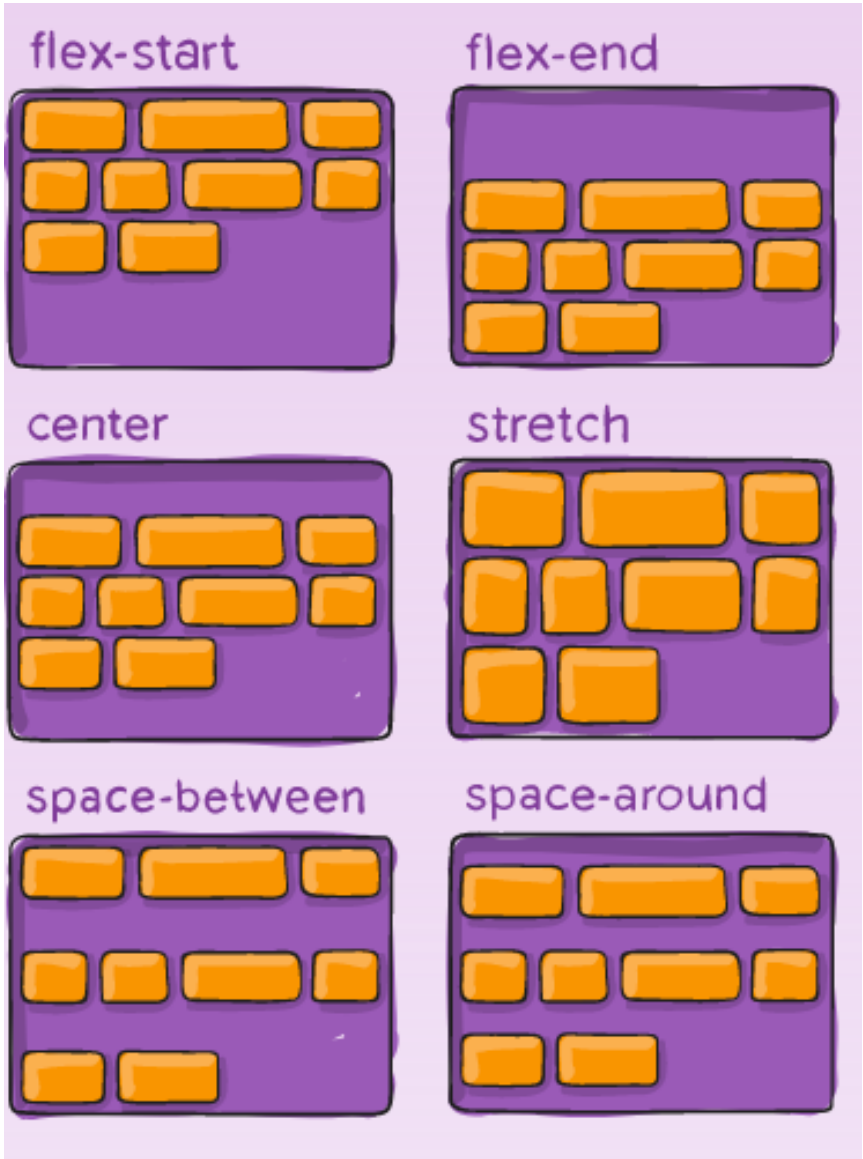


justify-content



align-items and align-content

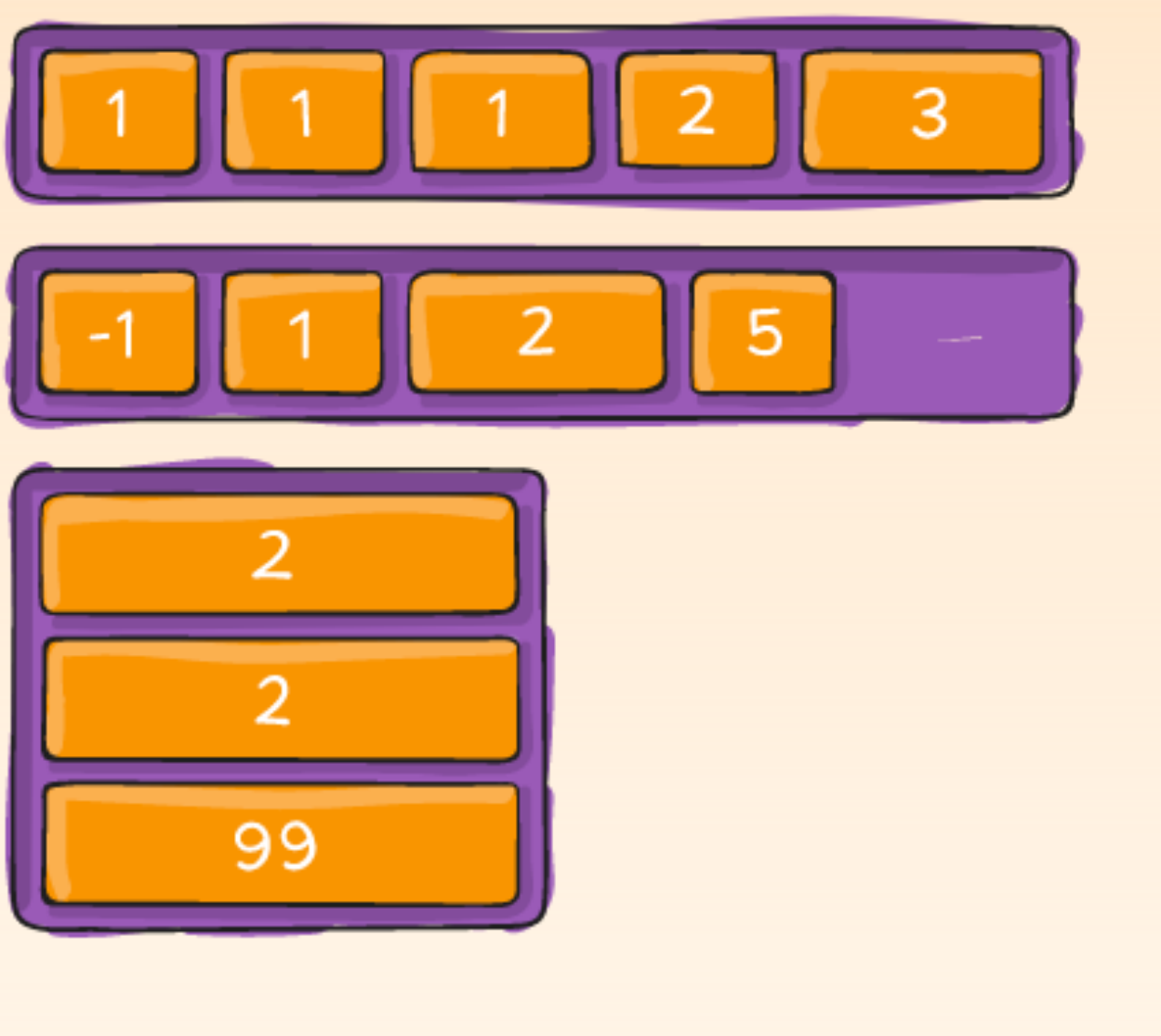




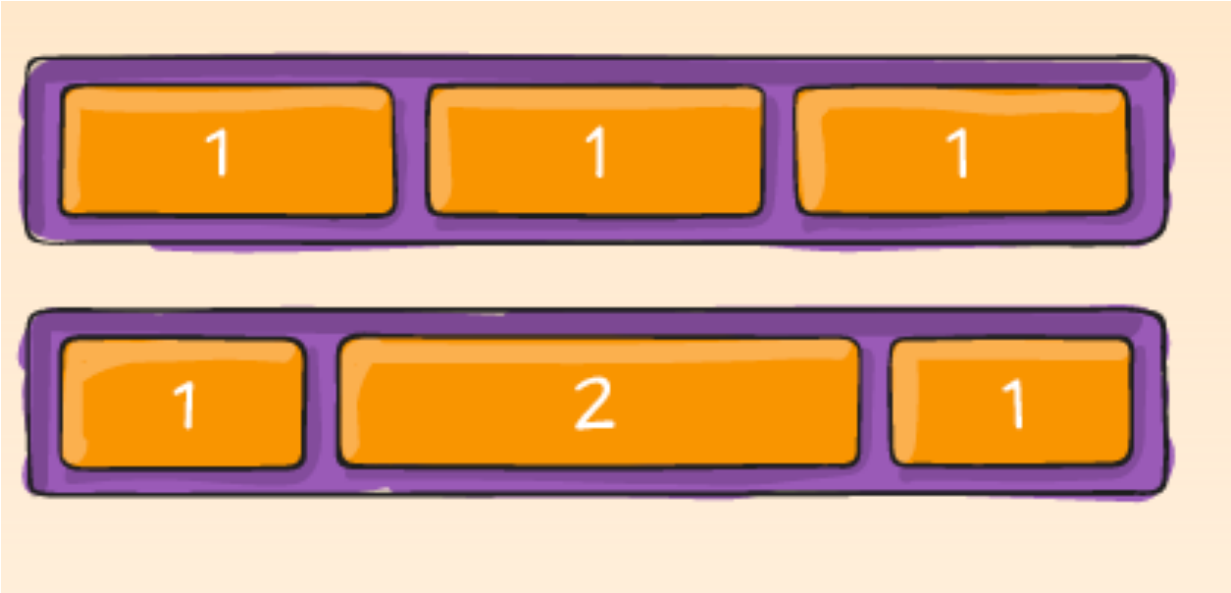
Properties set on children

- order: <integer>
- flex-grow: <positive number>
- flex-shrink: <positive number>
- flex-basis: <width/height>
- align-self: auto | flex-start | flex-end | center | baseline | stretch;
- flex (see reference documentation)

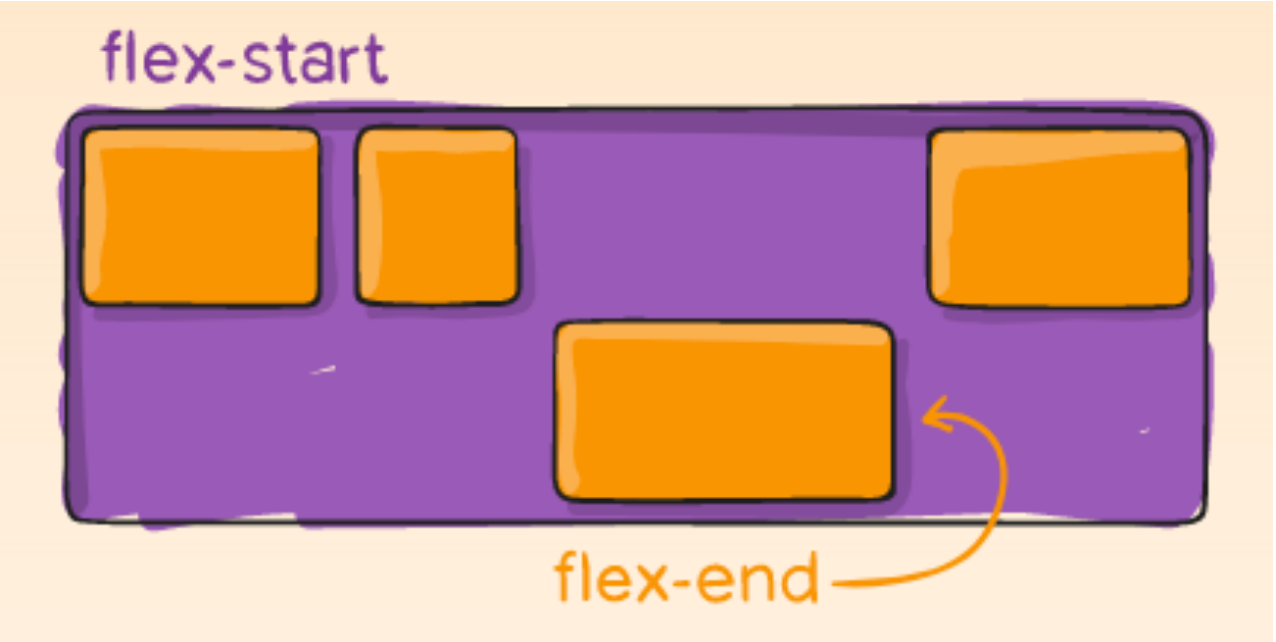
order



flex-grow and flex-shrink

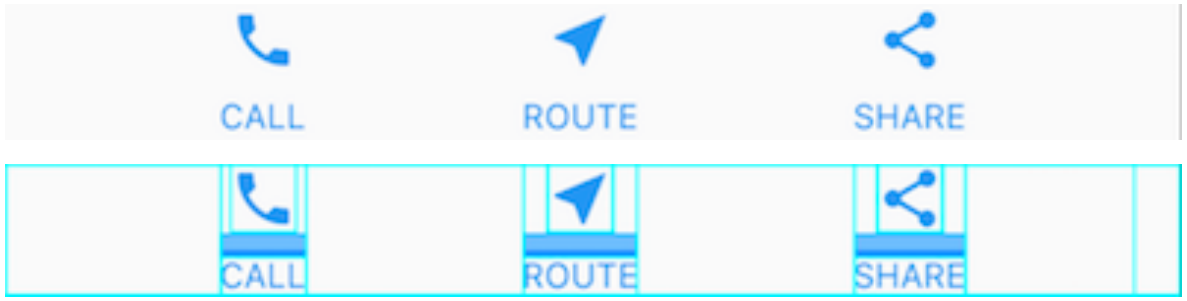


align-self

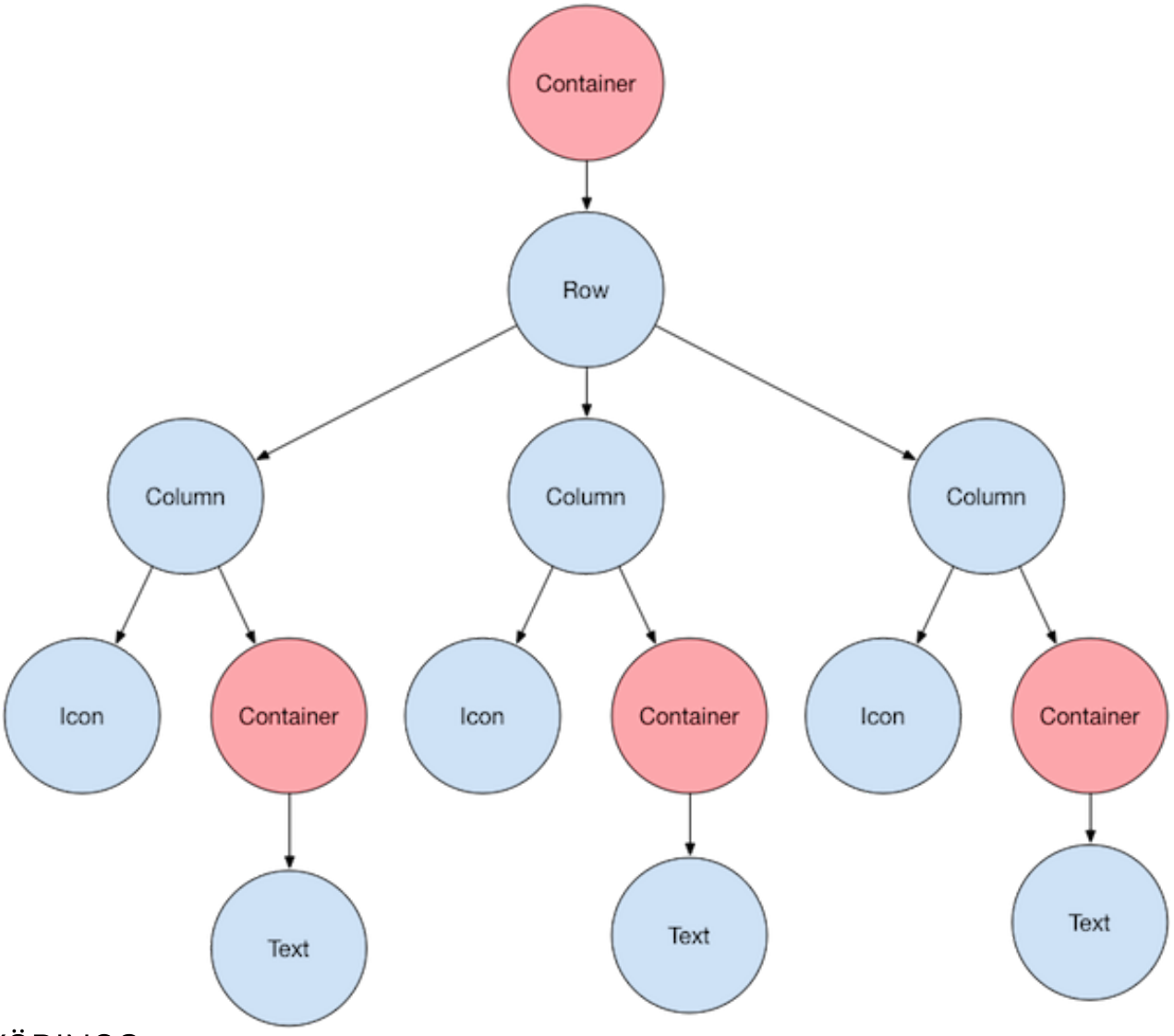


Layout in Flutter

- Based on Widgets in rows and columns
- Everything is a widget (almost)




CALL
ROUTE
SHARE


Container
Row
Column
Column
Column
Icon
Container
Icon
Container
Icon
Container
Text
Text
Text

Layout widgets

- Containers are layout widgets
    - https://flutter.dev/docs/development/ui/widgets/layout

Maganging Interaction

When the users does something in our application

LiU EXPANDING REALITY

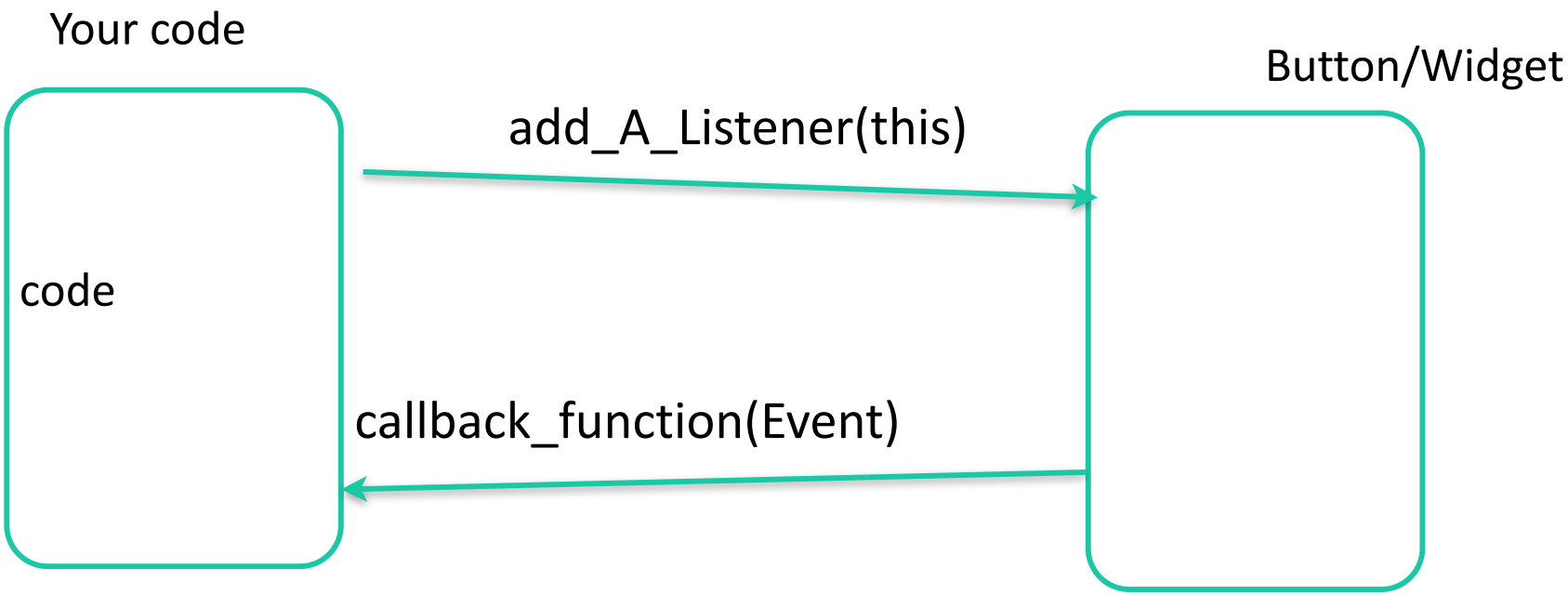
Widget Messages

- Messages from widgets to your code
  - Information about a change in state
  - Messages of user generated actions
    - Mouse movement, keyboard action
    - Touch , Sensors
- Widgets has a set of Messages (zero or more) you choose which you care about

Widget Messages

- Messages passed though callback functions
- You registers yourself with component using listeners
  - android : setOnClickListener
- Receive notification
  - onClick(View v)

Listeners


Your code
Button/Widget
code
add_A_Listener(this)
callback_function(Event)

Button-click kotlin

```
button.setOnClickListener{
            counter++
            textView.text = "Click counter : $counter"
        }
```

```
button.setOnClickListener(object: View.OnClickListener {
            override fun onClick(v: View?) {
                counter++
                textView.text = "Click counter : $counter"
                v?.setBackgroundColor(Color.MAGENTA)
            }
        })
```

Button-click flutter

```
FlatButton(
          child: Text('I am FlatButton'),
          onPressed: (){
            print('You tapped on FlatButton');
          },
        ),
```

Flutter has multiple Button widgets

Button-click React Native

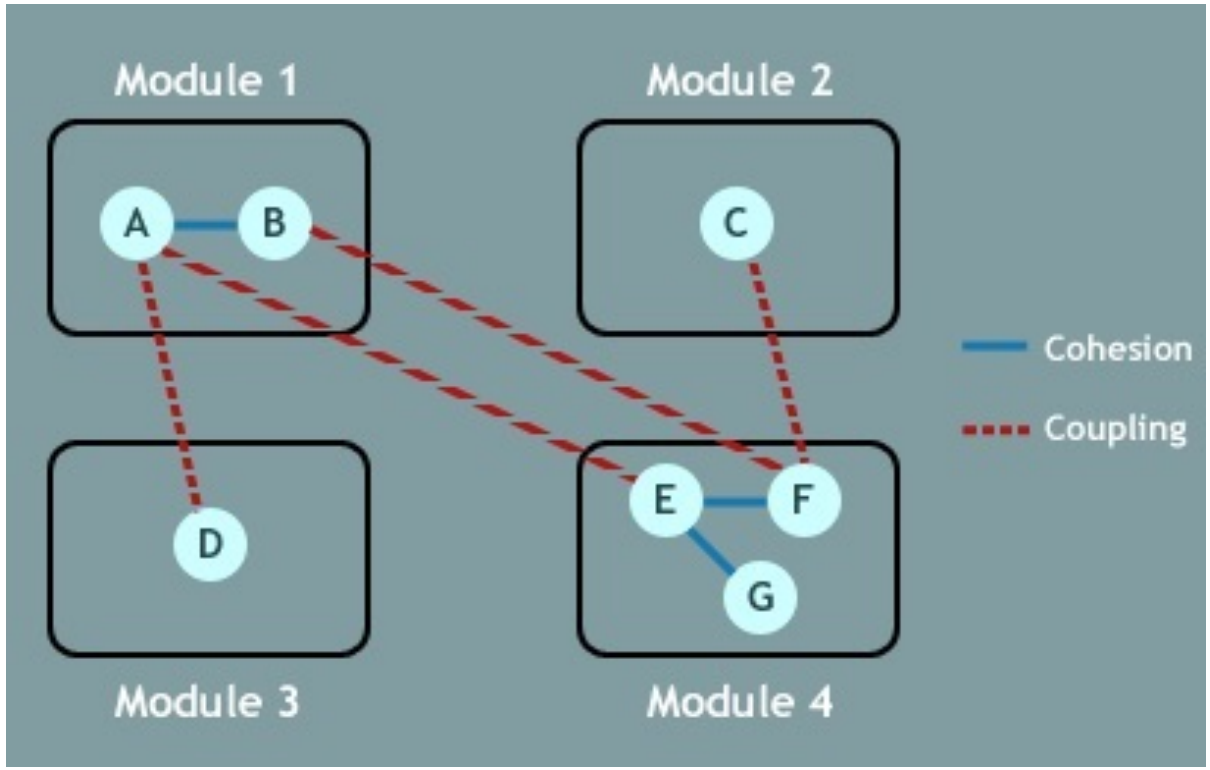
```
<Button
  onPress={() => {
    alert('You tapped the button!');
  }}
  title="Press Me"
/>
```

Organisation of code

- Biggest challenge of UI development (Ok one of)
  - Maintaining the correct and same state between the UI and the system/model/backend
- Separation of concerns
- Architecture Presentation Patterns
  - MVC
  - MVP
  - MVVM

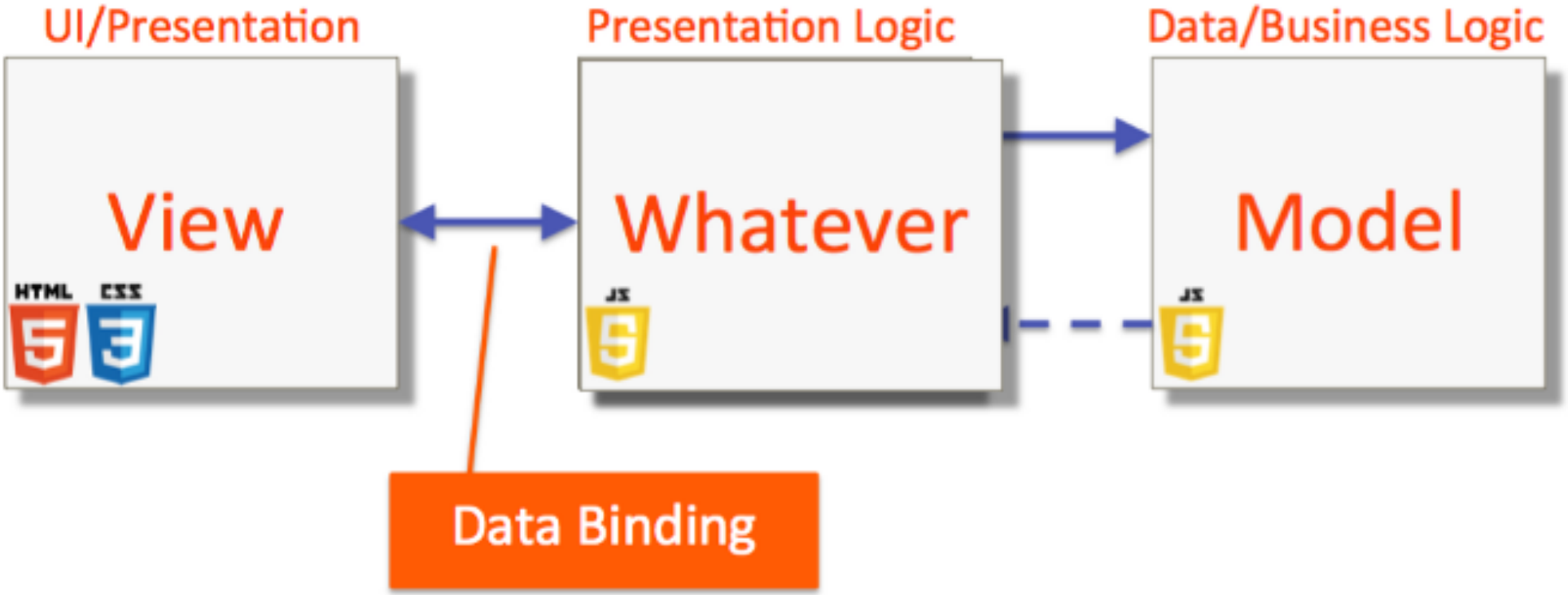
LiU EXPANDING REALITY

Cohesion and Coupling

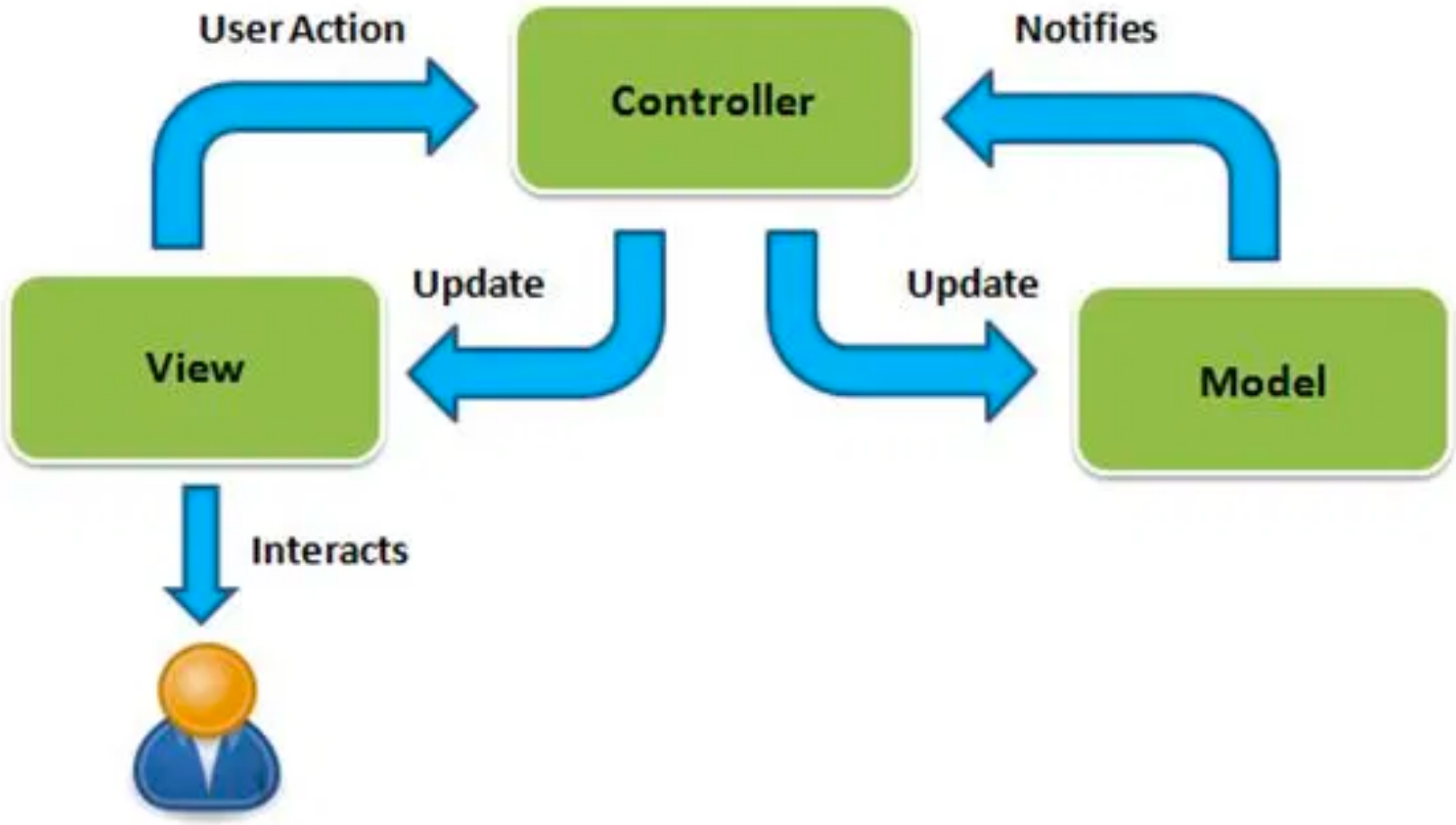


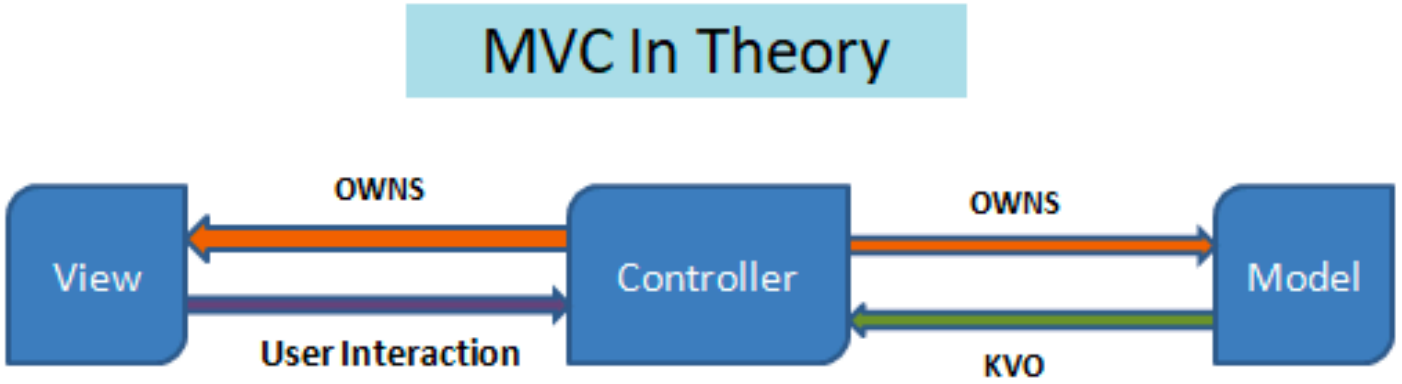
ORGANIZING YOUR CODE

MVC/MVP/MVVM

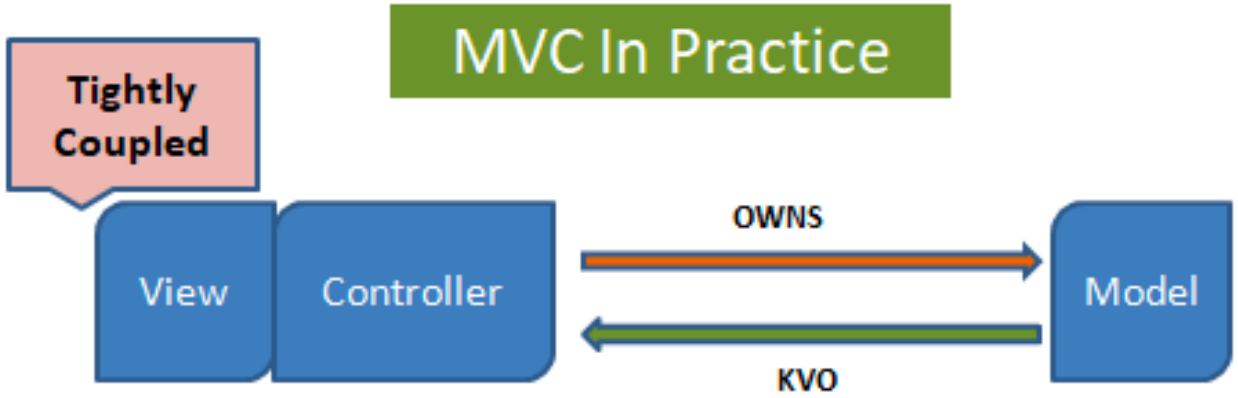


Model-View-Controller - MVC

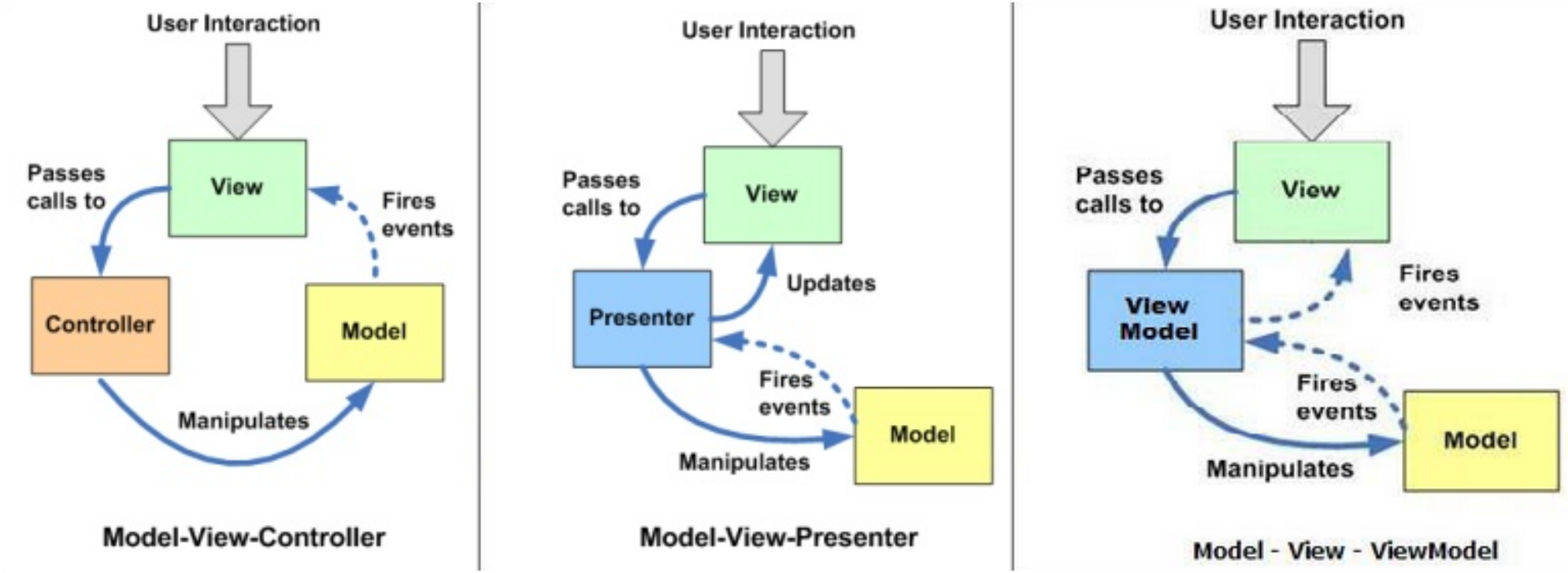




KVO = Key-Value Observing



Overview


User Interaction
Passes calls to
View
Fires events
Controller
Model
Manipulates
Model-View-Controller
User Interaction
Passes calls to
View
Updates
Presenter
Fires events
Model
Manipulates
Model-View-Presenter
User Interaction
Passes calls to
View
View Model
Fires events
Fires events
Model
Manipulates
Model - View - ViewModel

Demo