Determining Lifecycle State

Jim Wilson
MOBILE SOLUTIONS DEVELOPER & ARCHITECT

@hedgehogjim blog.jwhh.com

What to Expect from This Module

- Events are only part of the lifecycle story
- Lifecycle-driven asynchronous operations
- Lifecycle state
- Relationship between lifecycle state and events

Events Are Only Part of the Story

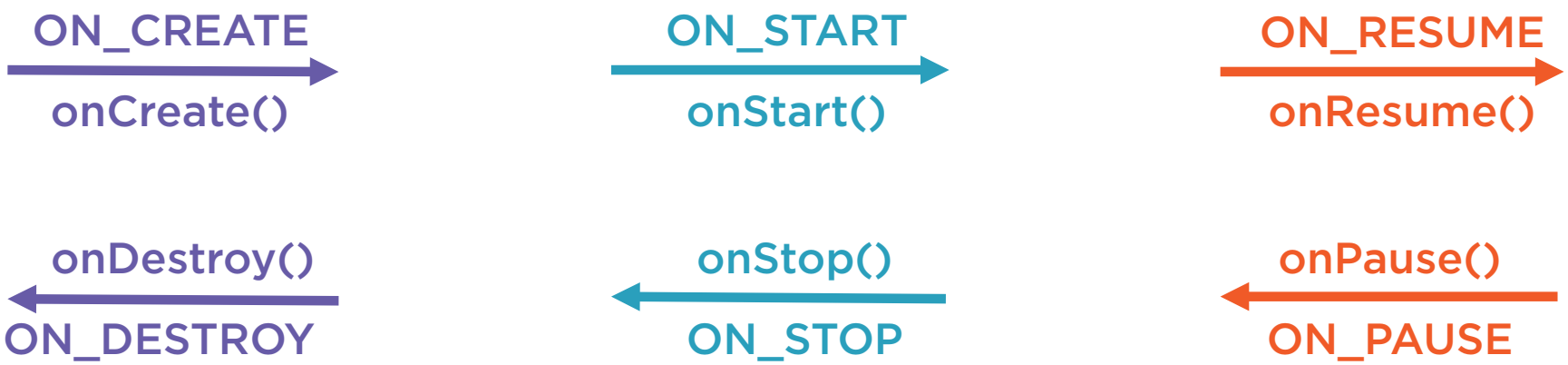
Lifecycle Events

- Indicate a lifecycle transition
- Great for triggering work

Lifecycle is more than transitions

- Activity is always in some stage of life
- Need a way to determine that stage
- Especially important during asynchronous work

Lifecycle Transitions



Lifecycle State

Lifecycle State

- A point in the lifecycle

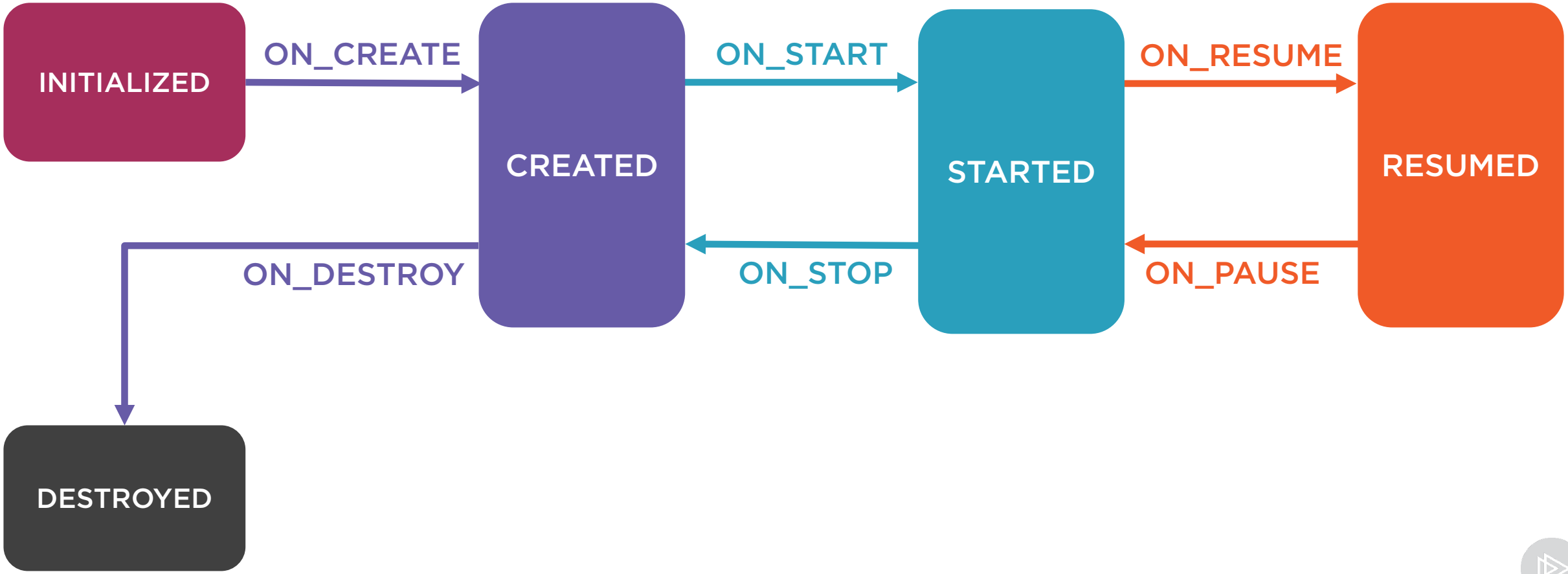
State and events work together

- Events fire as part of the transition from one state to another

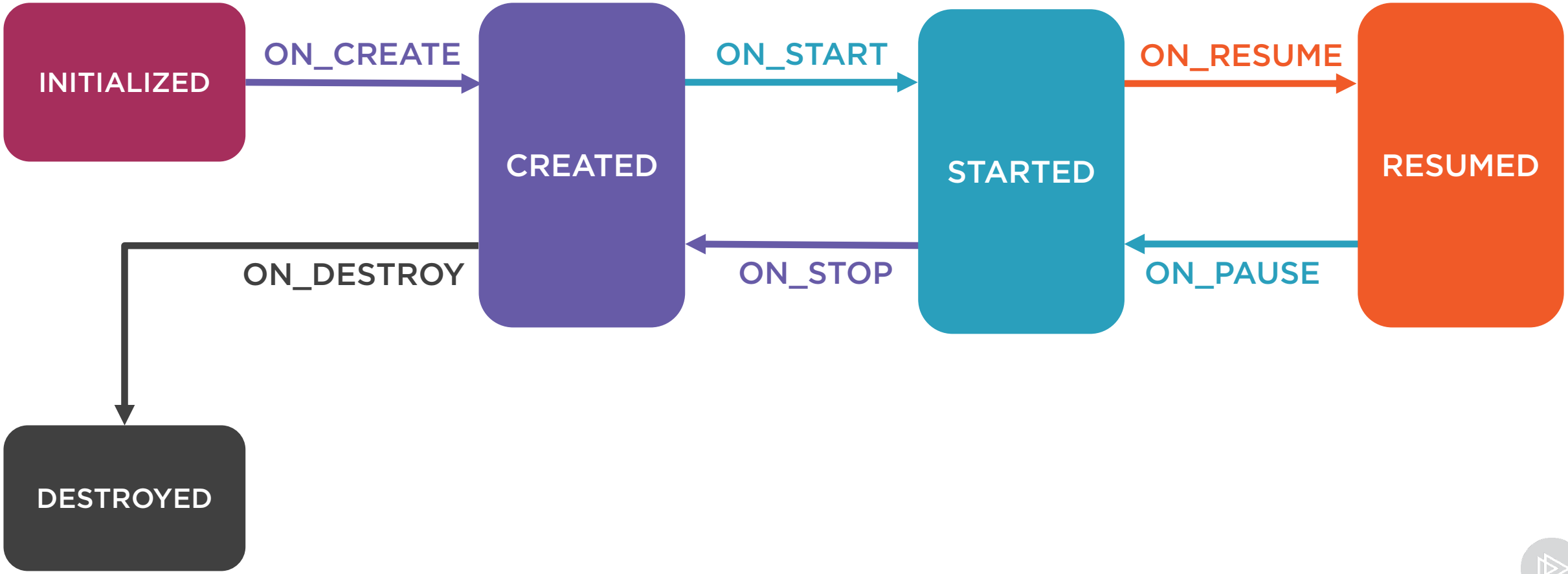
State can be accessed at any time

- Use Lifecycle.currentState
- Can check for specific state
- Can check that is at least at some state

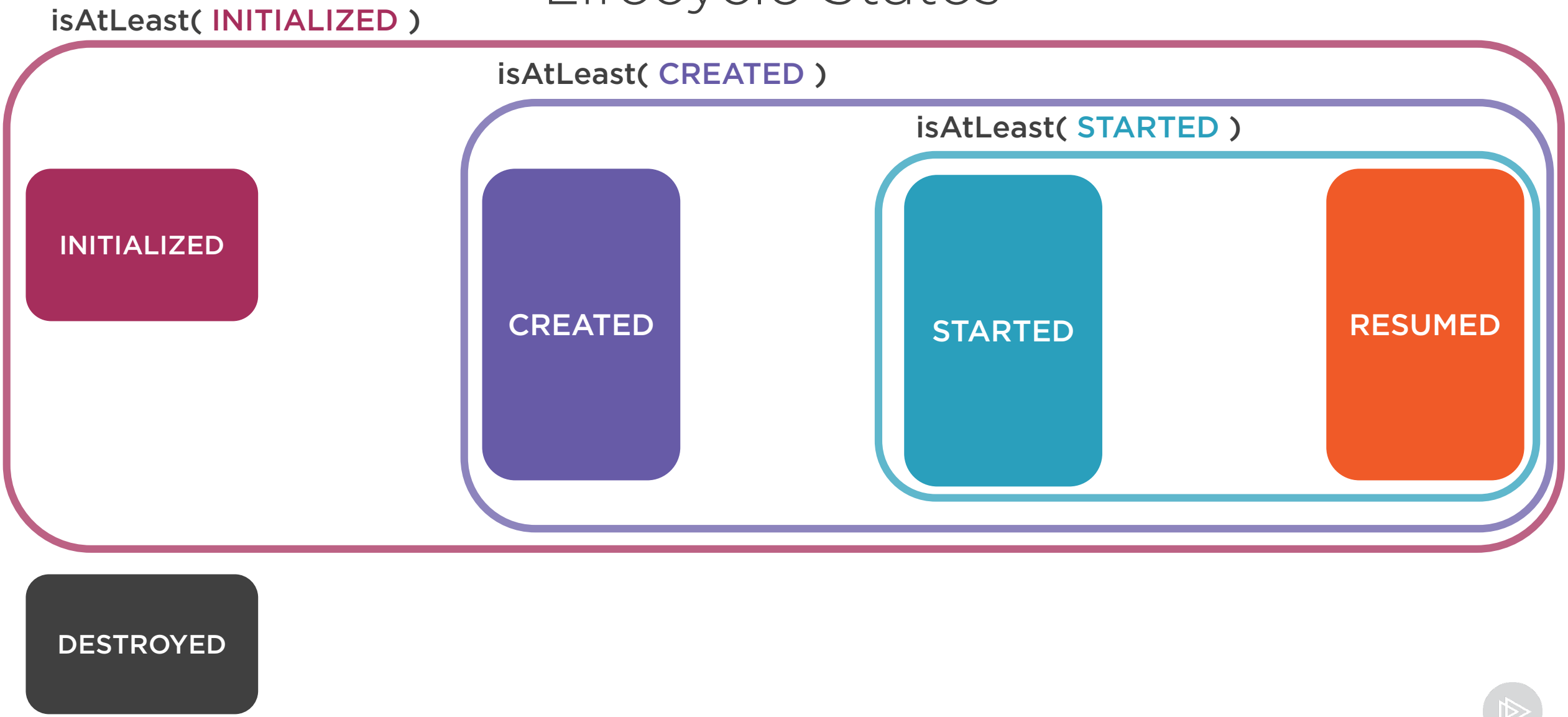
Lifecycle States



Lifecycle States



Lifecycle States



Summary

Lifecycle State

- A point in the lifecycle

Checking the lifecycle state

- Accessed with Lifecycle.currentState
- Use isAtLeast to determine if at or beyond a point in the lifecycle

Summary

Events are tied to state

- Fire as part of a state transition
- Makes it easy to initiate work based on state transition

Android Apps with Kotlin

Keep building your skills...

- This course is part of a large series on Android and Kotlin

The best way to find us...

- Search for *Android Apps with Kotlin* in the Pluralsight course list

Android Apps with Kotlin

Make sure you have the fundamentals down...

- Android Apps with Kotlin: Build Your First App

Android Apps with Kotlin

Make sure you have the fundamentals down...

- Android Apps with Kotlin: Build Your First App
- Android Apps with Kotlin: Tools and Testing

Android Apps with Kotlin

Make sure you have the fundamentals down...

- Android Apps with Kotlin: Build Your First App
- Android Apps with Kotlin: Tools and Testing
- Android Apps with Kotlin: Resources, Styles, and Themes

Android Apps with Kotlin

Make sure you have the fundamentals down...

- Android Apps with Kotlin: Build Your First App
- Android Apps with Kotlin: Tools and Testing
- Android Apps with Kotlin: Resources, Styles, and Themes
- Android Apps with Kotlin: RecyclerView and Navigation Drawer

Android Apps with Kotlin

Make sure you have the fundamentals down...

- Android Apps with Kotlin: Build Your First App
- Android Apps with Kotlin: Tools and Testing
- Android Apps with Kotlin: Resources, Styles, and Themes

Keep building your skills...

- Search for *Android Apps with Kotlin* in the Pluralsight course list

Android Apps with Kotlin

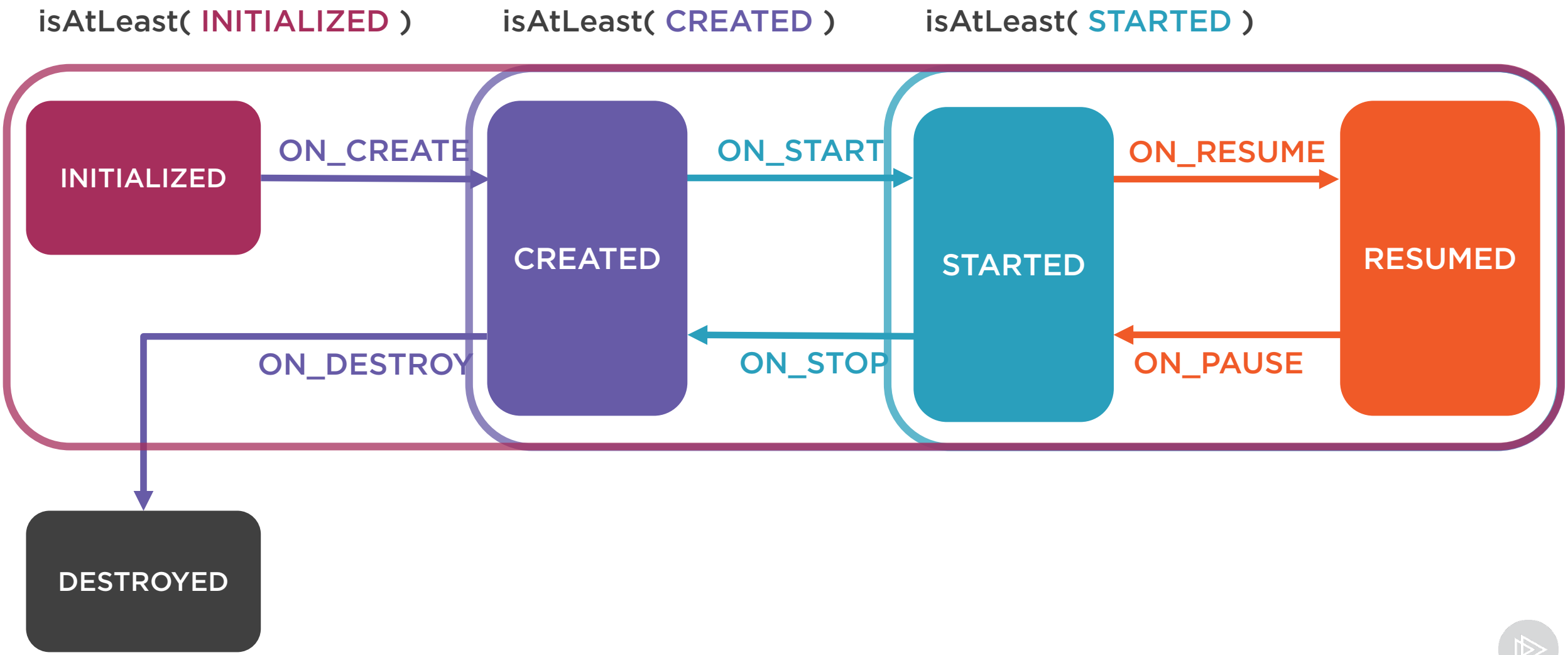
Keep building your skills...

- Android Apps with Kotlin: ViewModel and Lifecycle

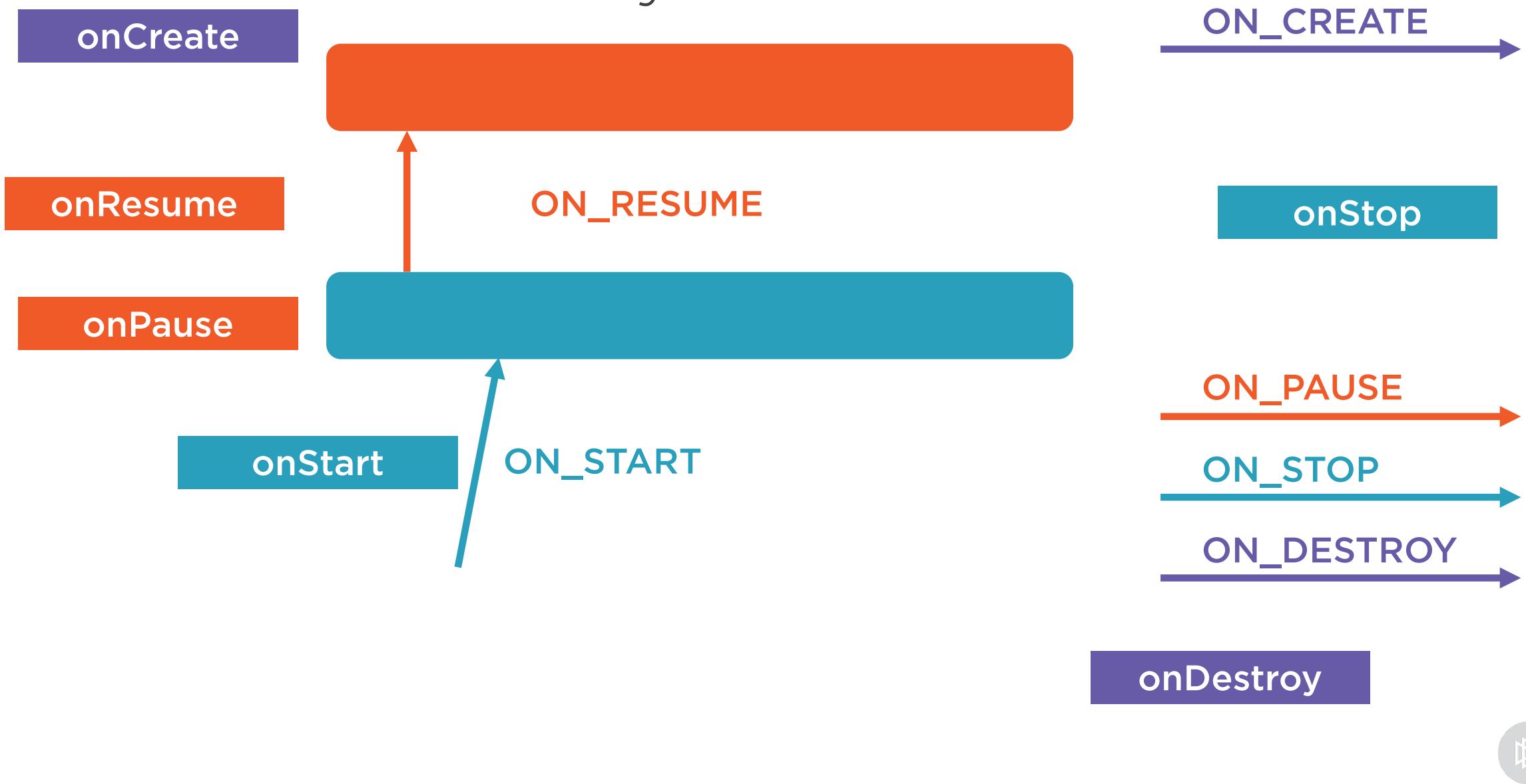
Summary

Summary

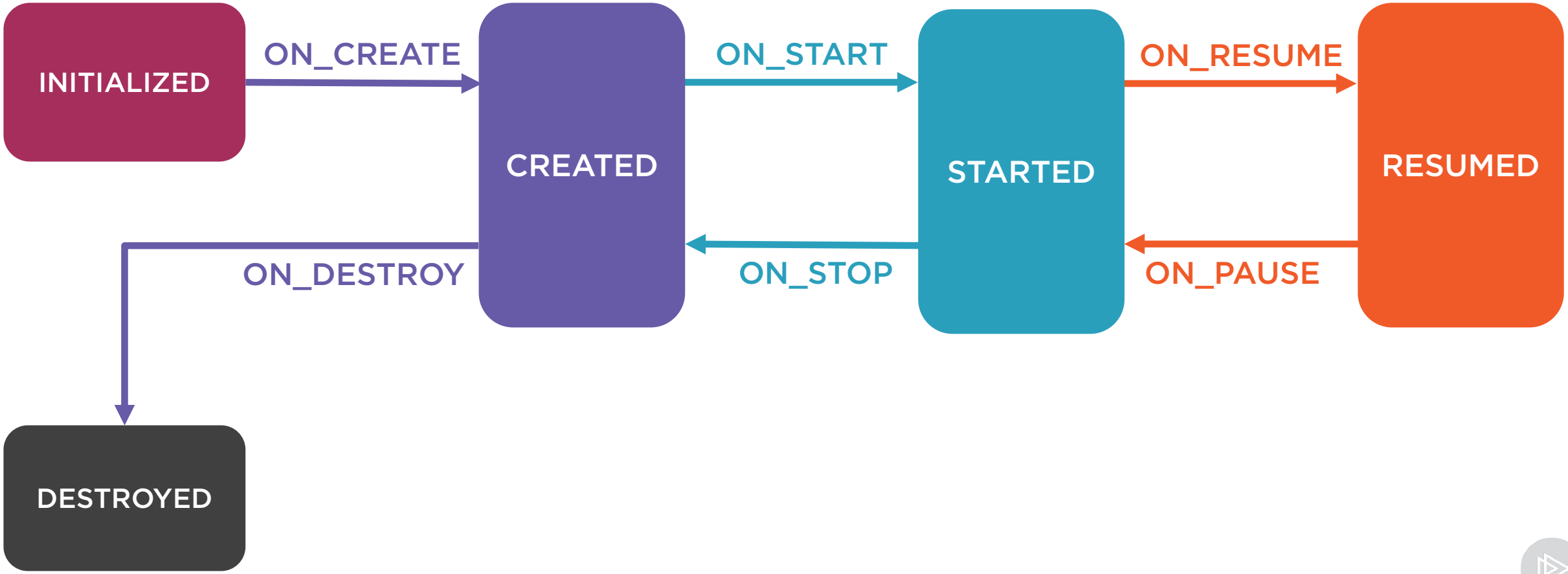
Lifecycle States



Lifecycle Events



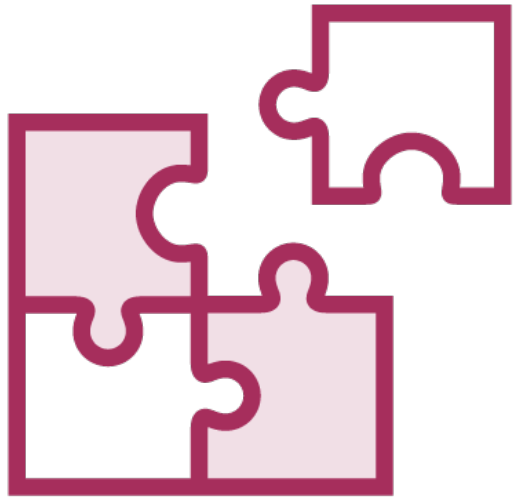
Lifecycle Events



Cooperating with Activity Lifecycle

Activities have a lifecycle

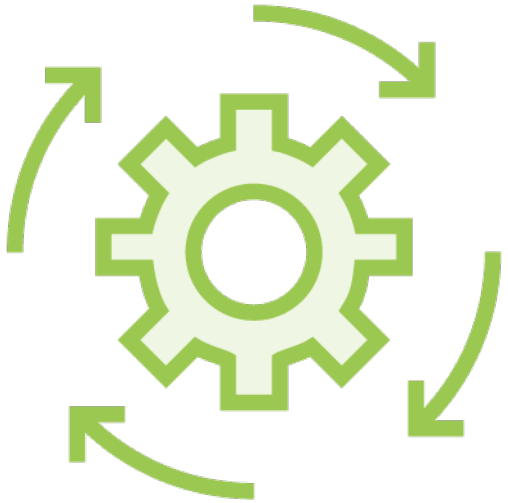
Our code needs to cooperate with that lifecycle



App's leverage other components

Some components need to cooperate with that lifecycle

Components and Activity Lifecycle

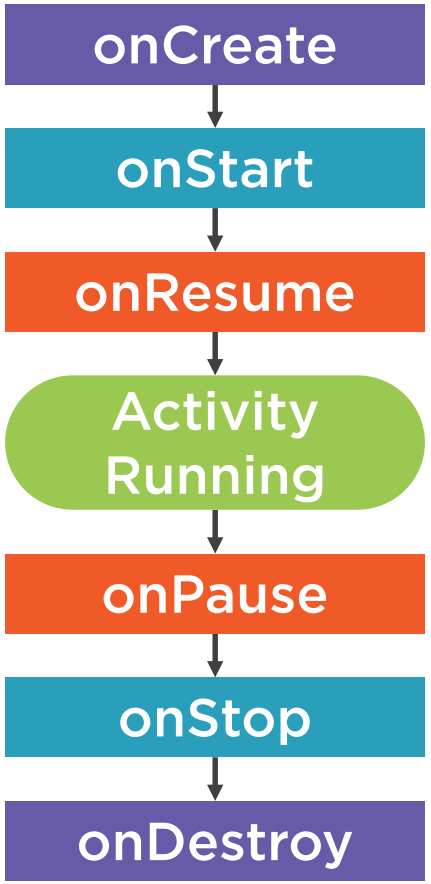


Start or connect to a local service

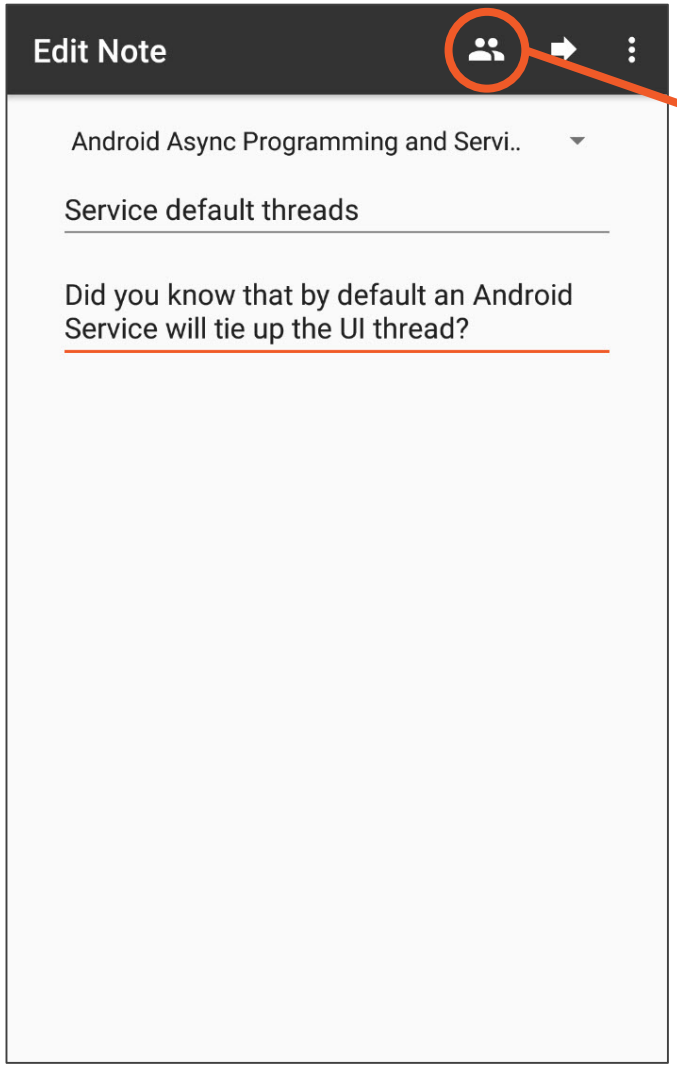
Connect to a remote backend

May leak resources if not cleaned up

Activity Lifecycle Methods



Our App's Get-together Feature



Invite a pre-defined list of people to meet you at your current location to discuss the info in your note

Location Component

Messaging Component

Get-together Components

PseudoLocationManager

- Simulates location updates
- Start when activity onStart called
- Stop when activity onStop called
- Provide callback for location updates

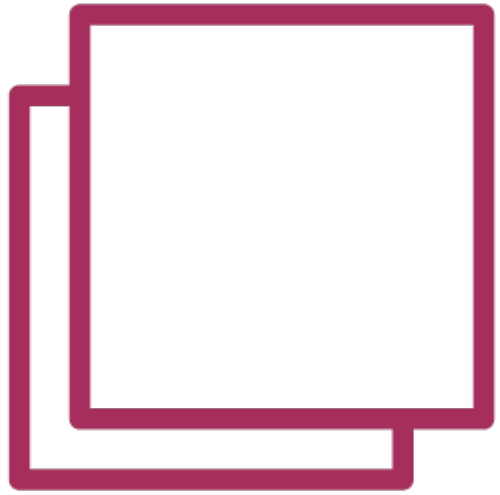
PseudoMessagingManager

- Simulates messaging behavior
- Connect when activity onStart called
- Disconnect when activity onStop called
- Use to send location and note info

Improving Lifecycle Handling

Cooperate with activity lifecycle

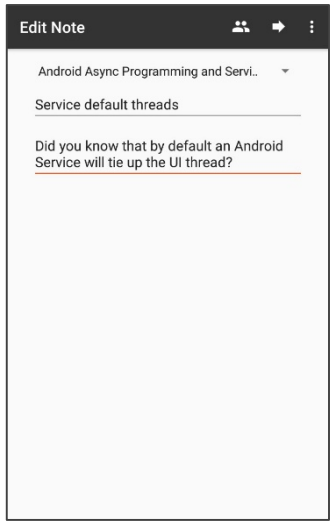
But want to avoid bloating the activity class



Separate handling from the activity

Need a way to be notified when each lifecycle event occurs

Improving Lifecycle Handling

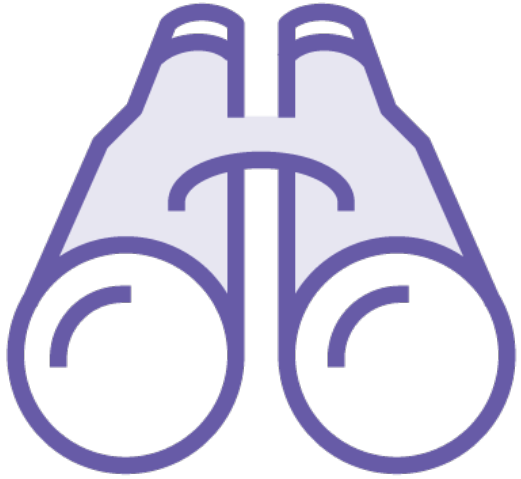


LifecycleOwner

Implemented by types that have a lifecycle

Lifecycle

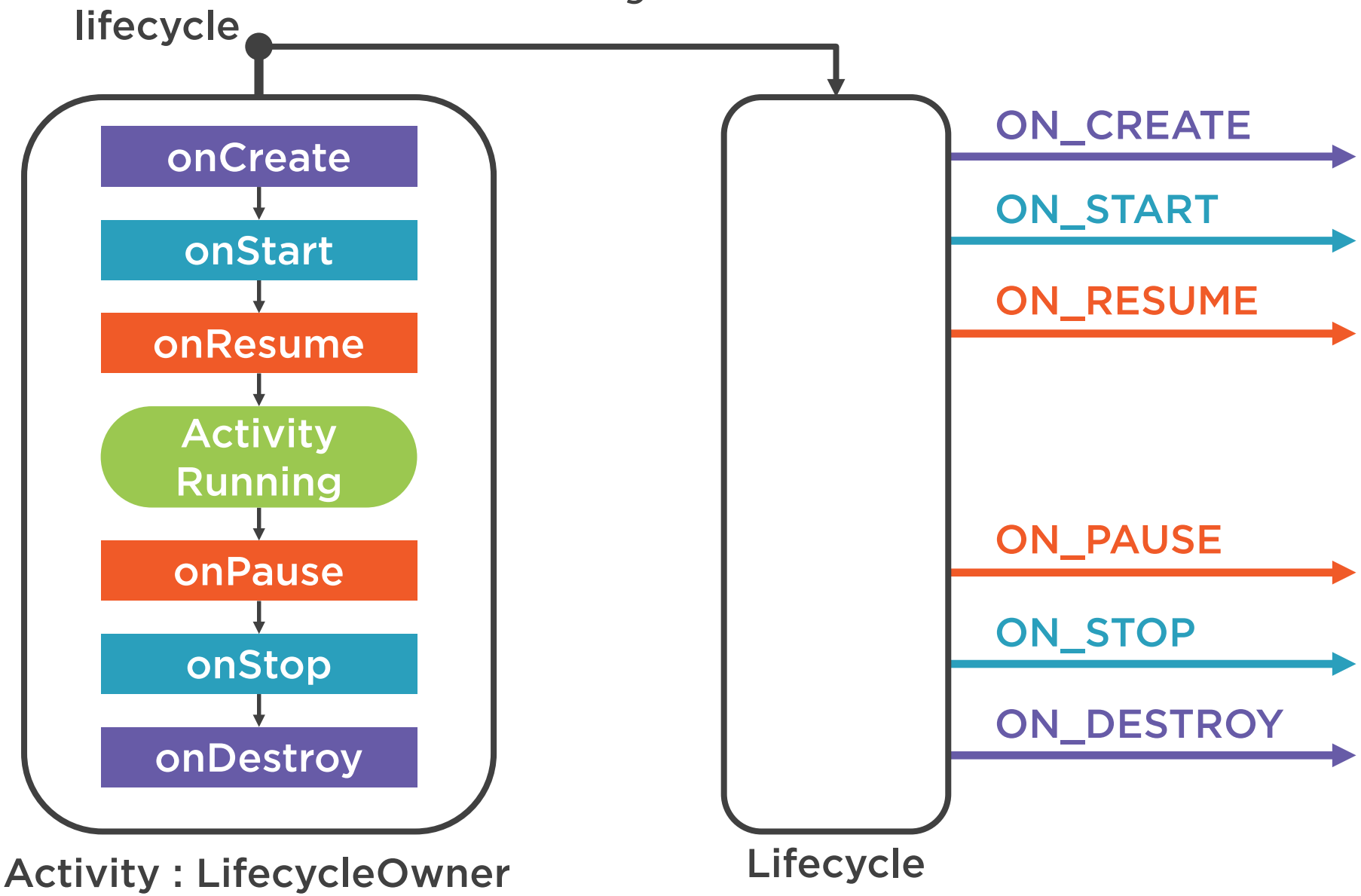
Represents an object's lifecycle

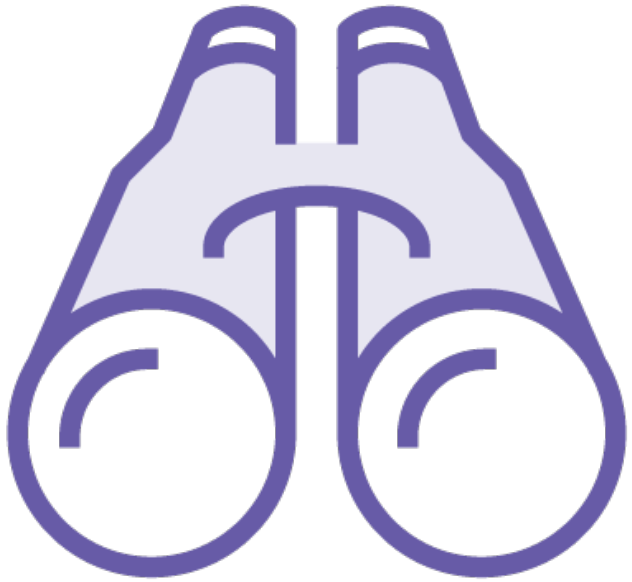


LifecycleObserver

Implement to indicate support for receiving lifecycle events

Lifecycle Events





Observing Lifecycle Events

Mark class as an observer

- Implement LifecycleObserver

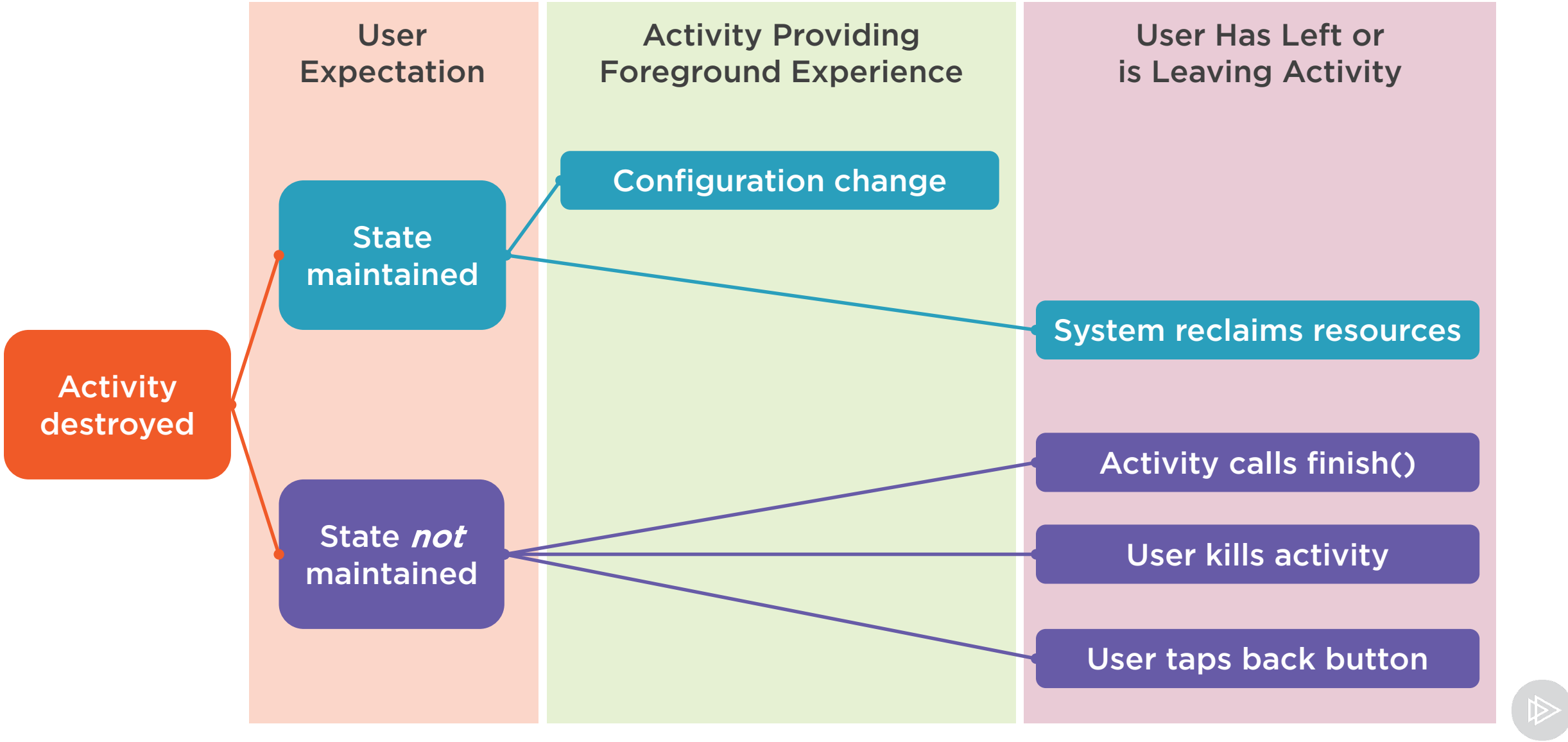
Mark event handling functions within class

- Use OnLifecycleEvent annotation
- Specify event of interest with Lifecycle.Event.ON_xxxx constants

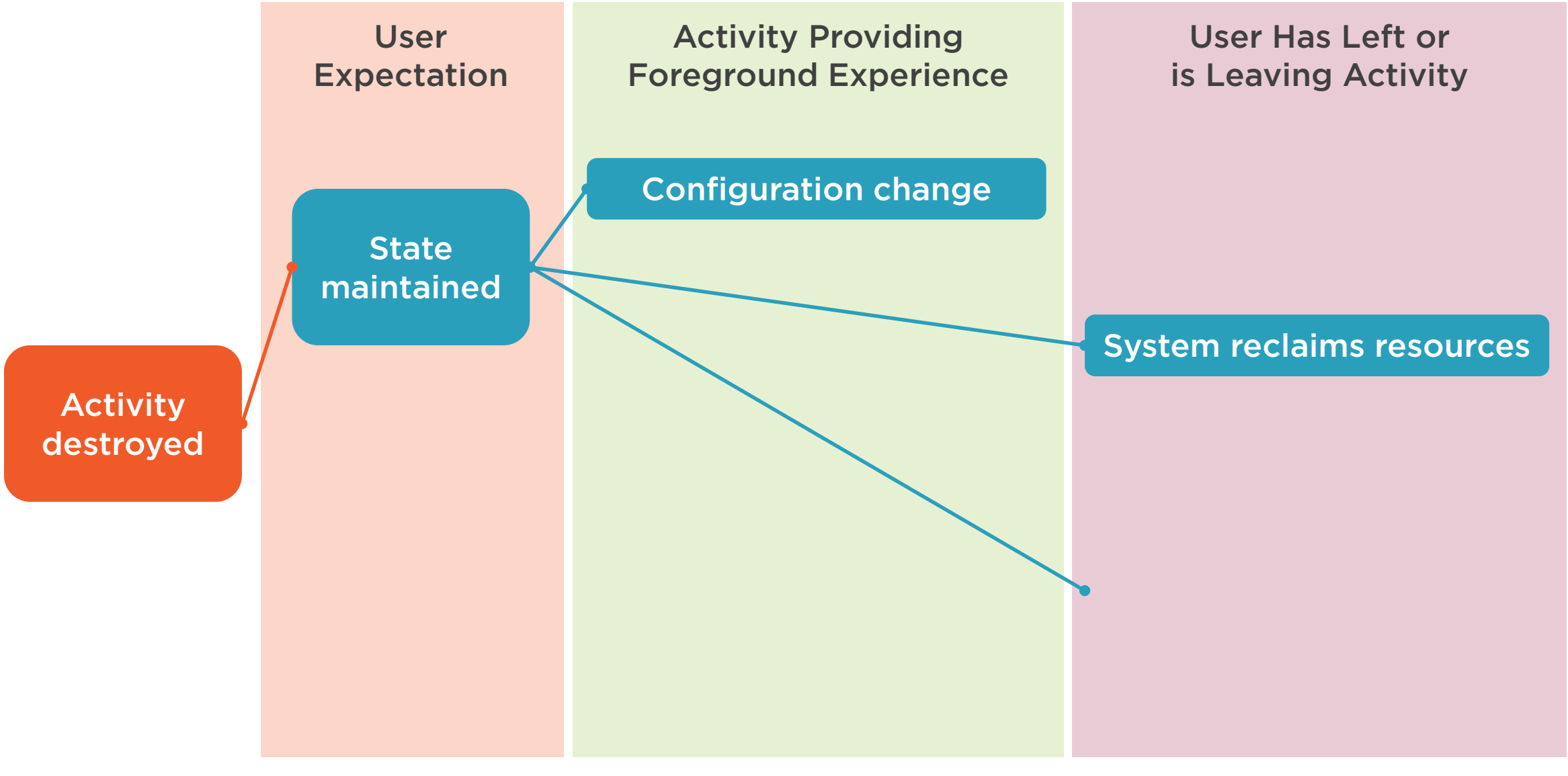
Register as an observer

- Use Lifecycle.addObserver

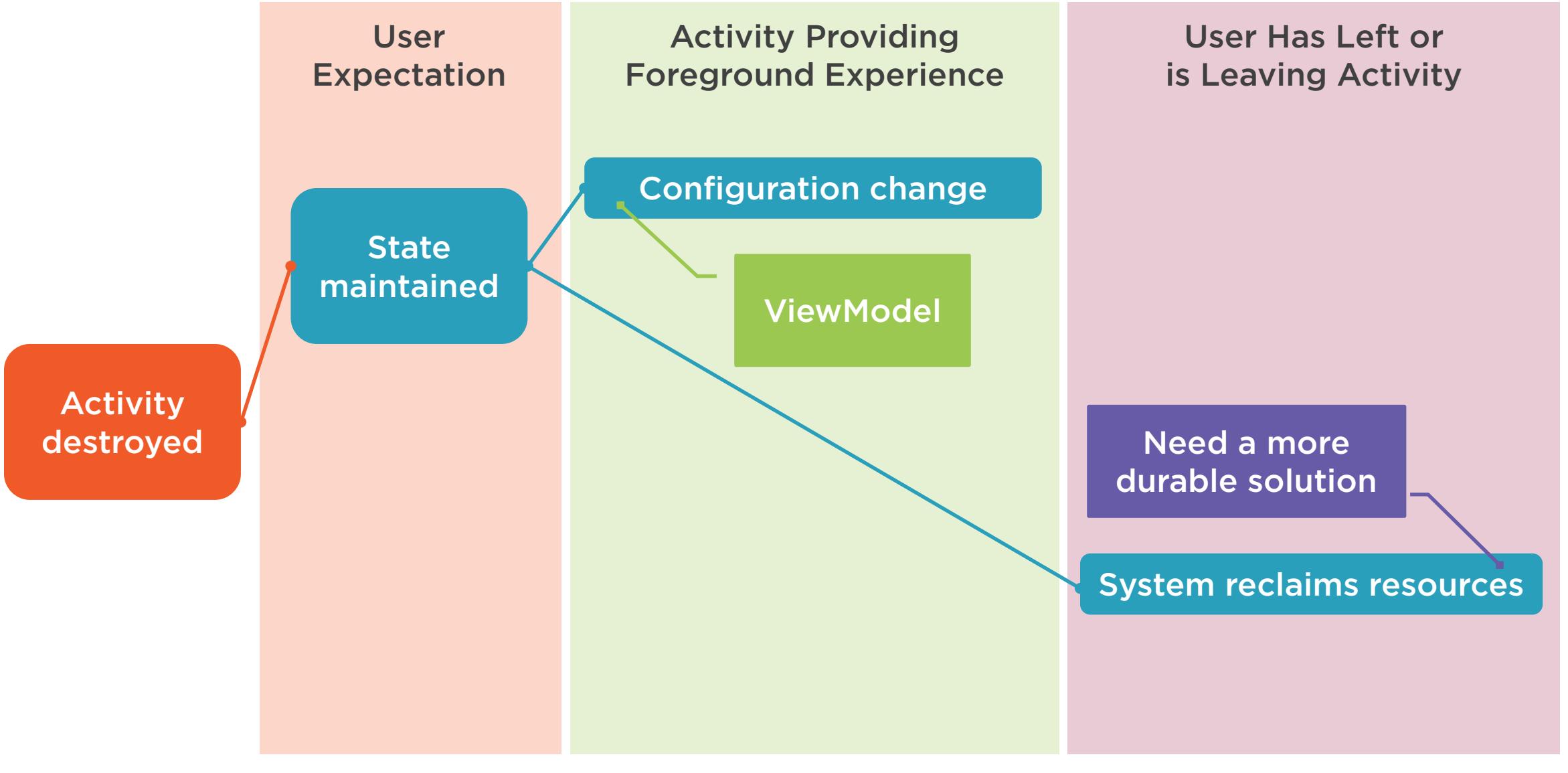
Activity Destruction Effects on State



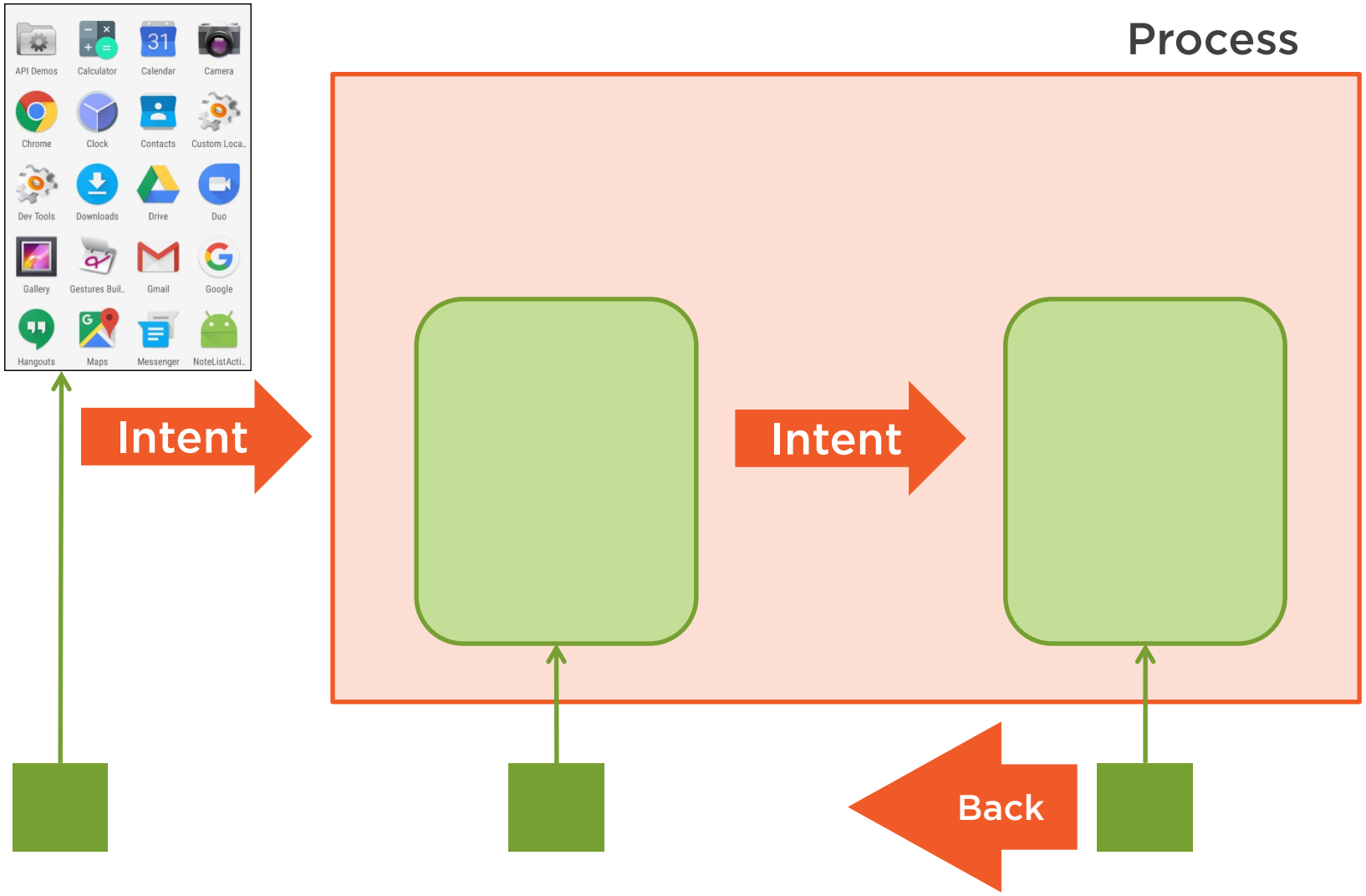
Activity Destruction Effects on State



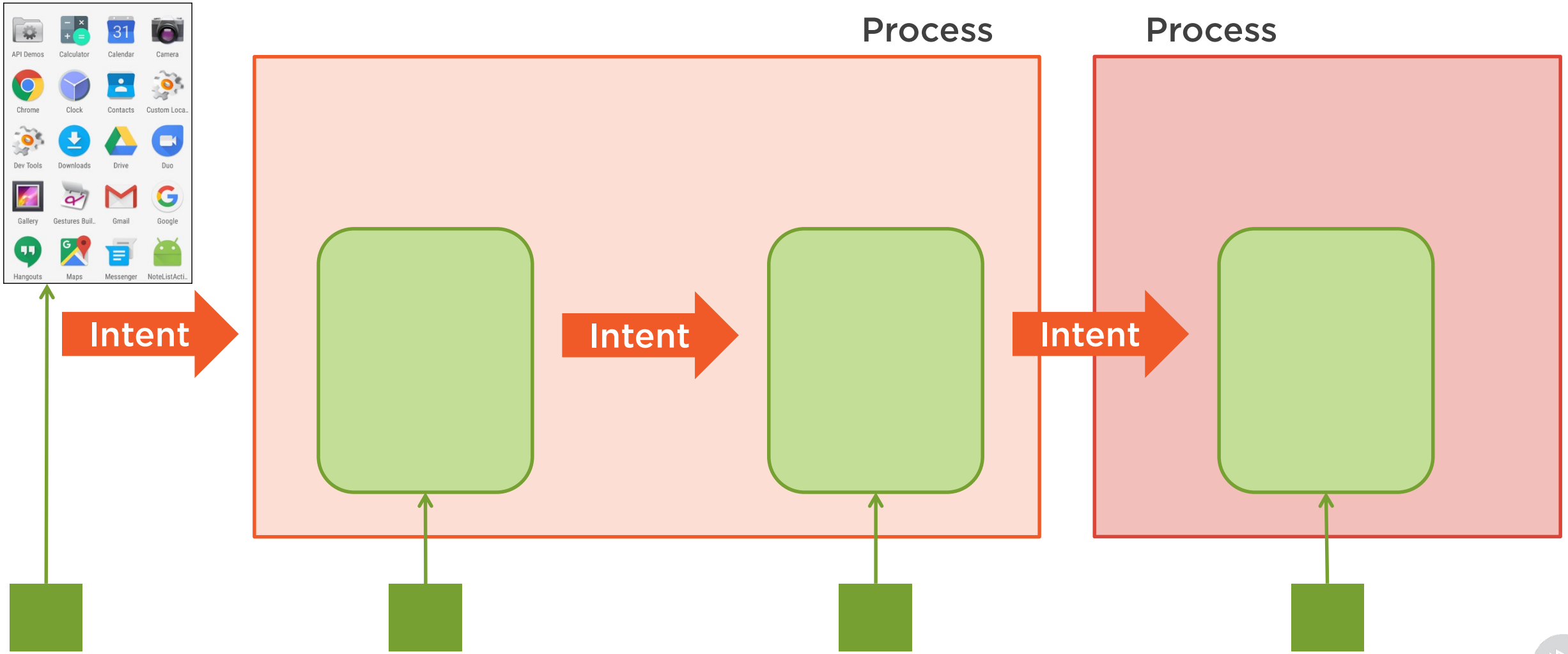
Activity Destruction Effects on State



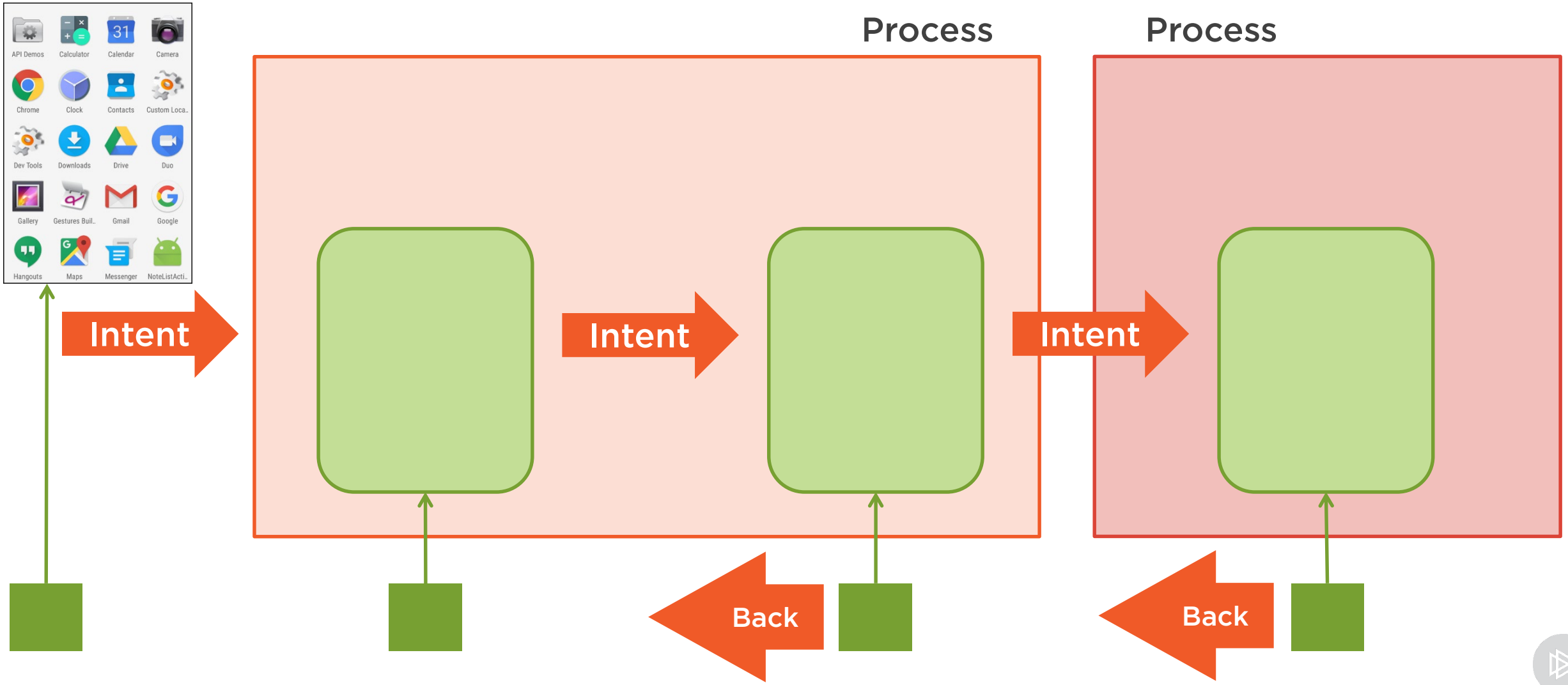
The Need for Durable Activity State



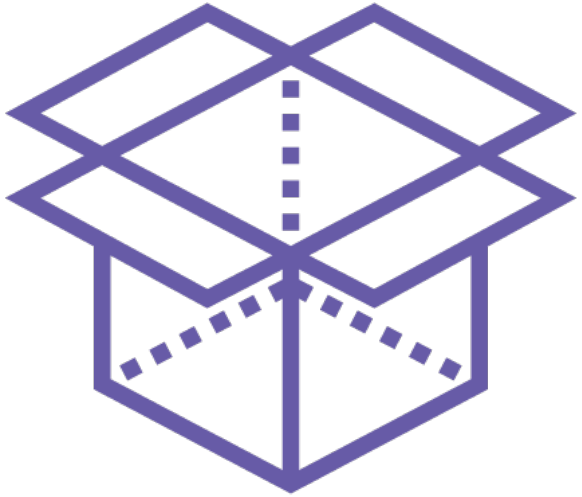
The Need for Durable Activity State



The Need for Durable Activity State



Durable Activity State



onSaveInstanceState

Save state into a bundle

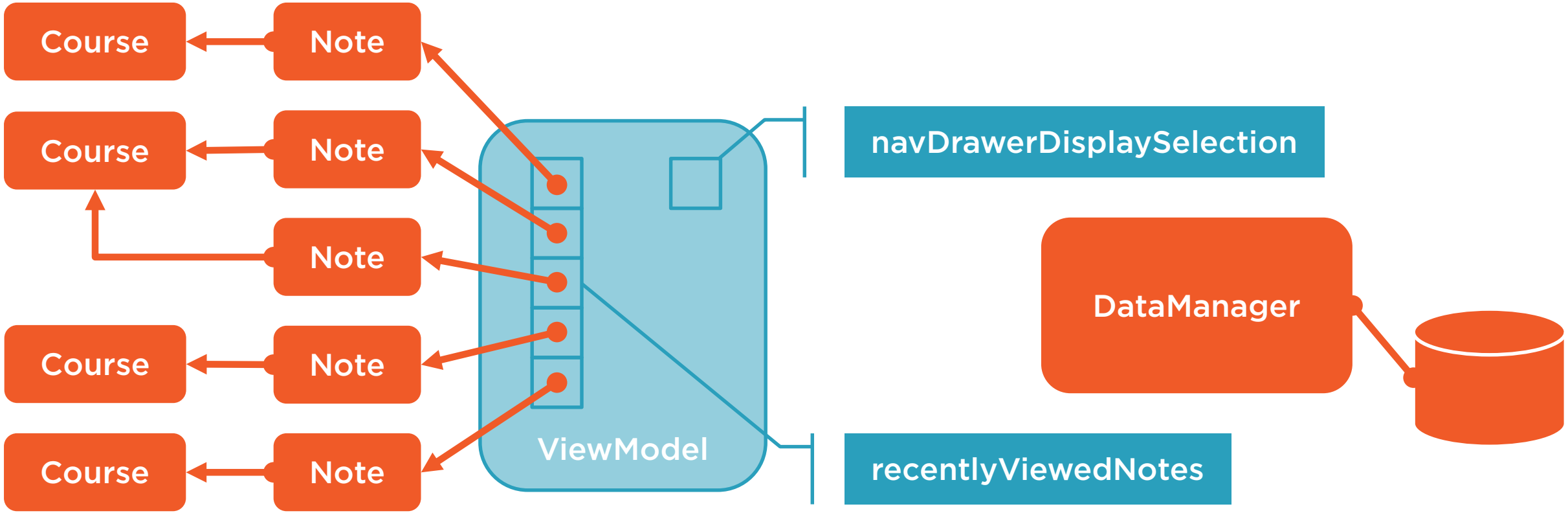
System persists bundle

onCreate

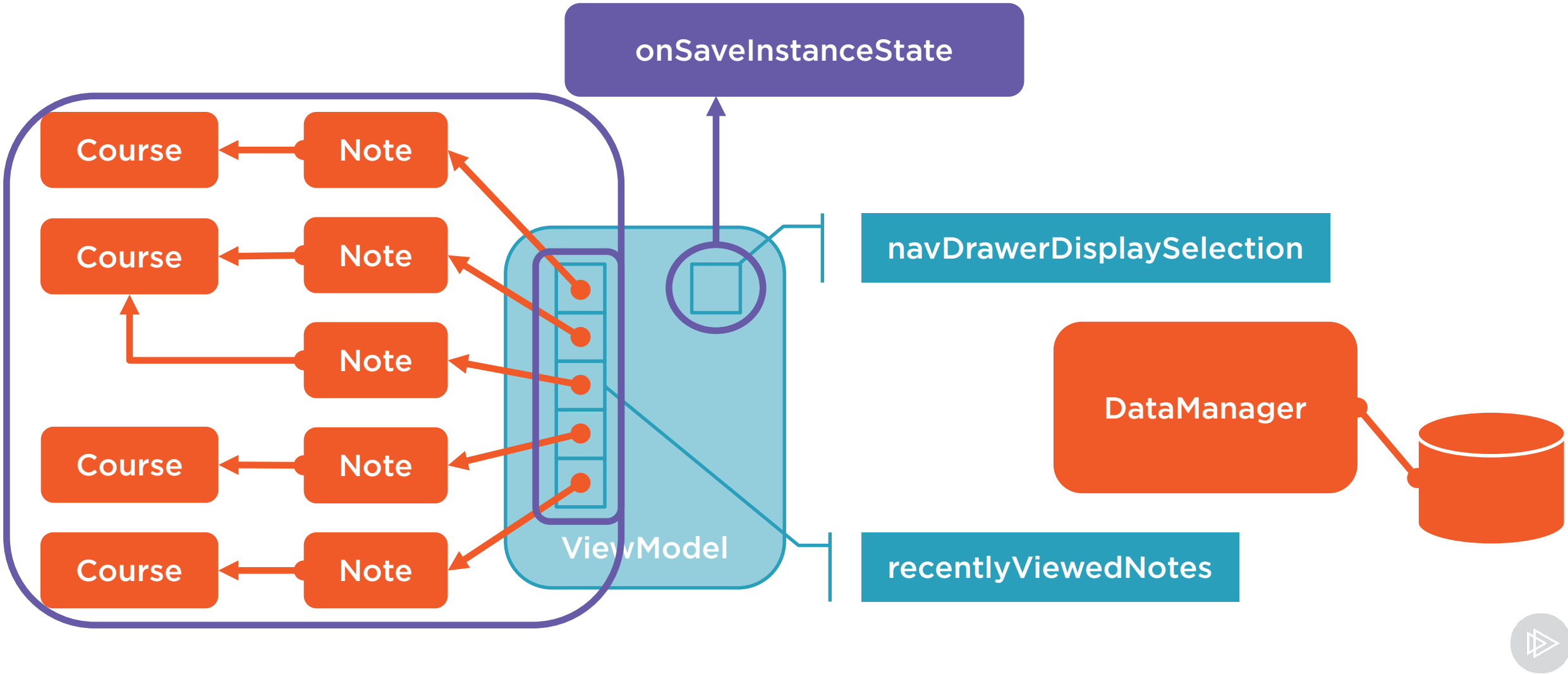
Receives bundle with saved state values

Bundle is null on initial create

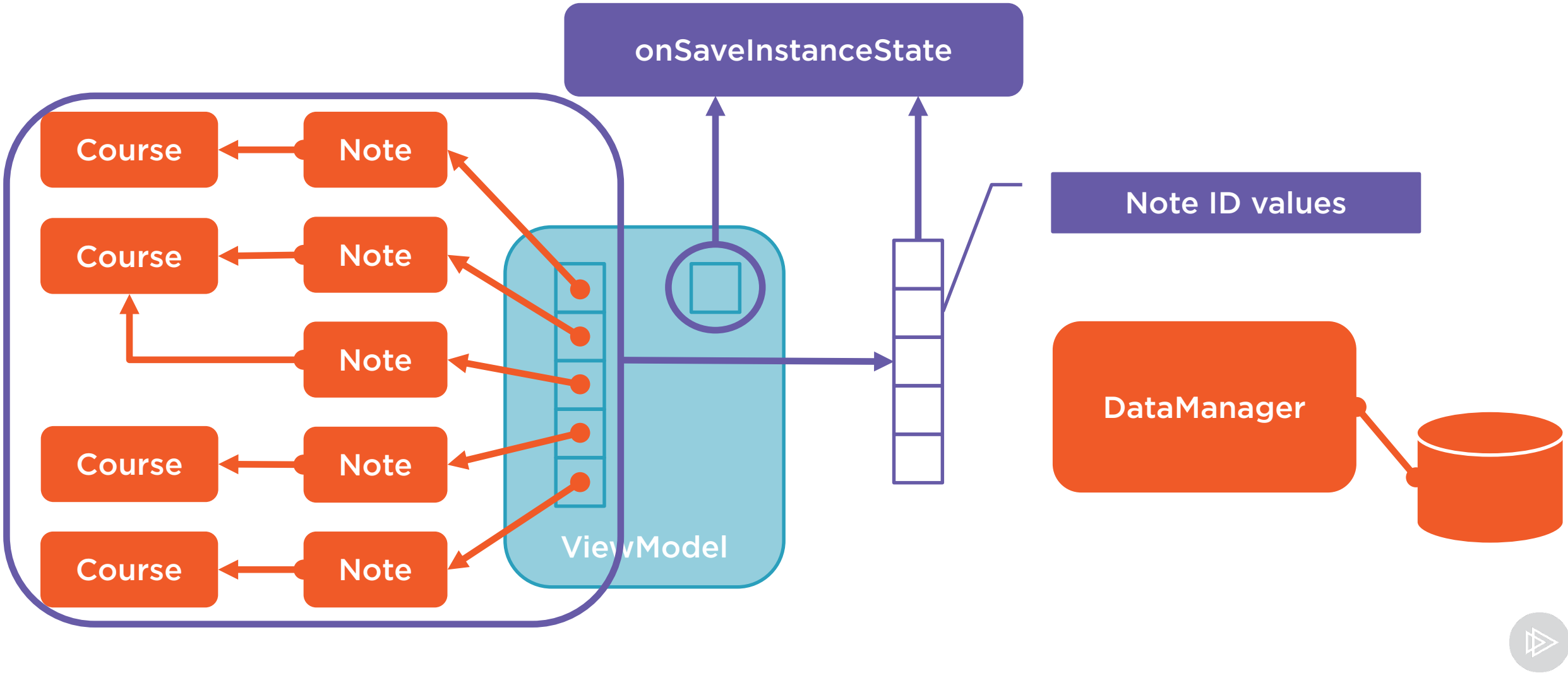
Our App's Data and State



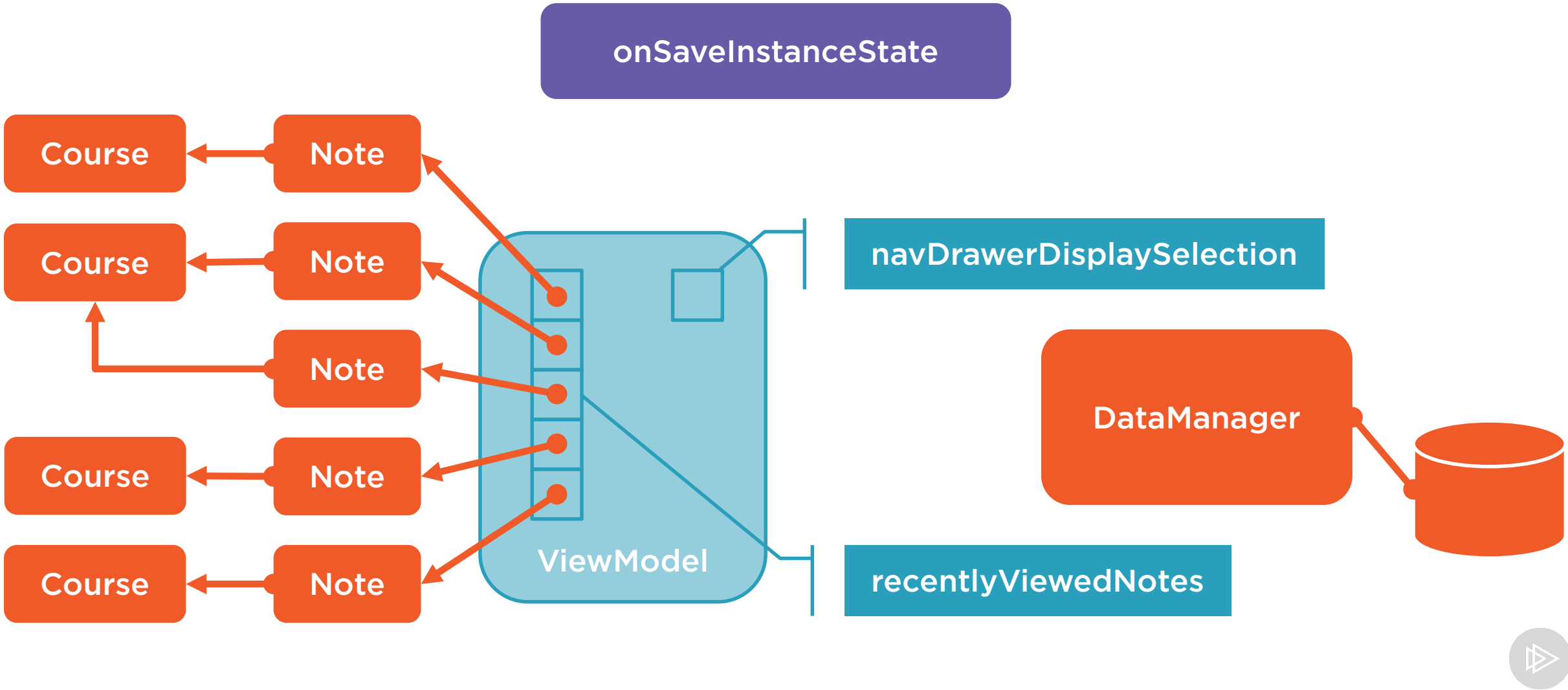
Our App's Data and State



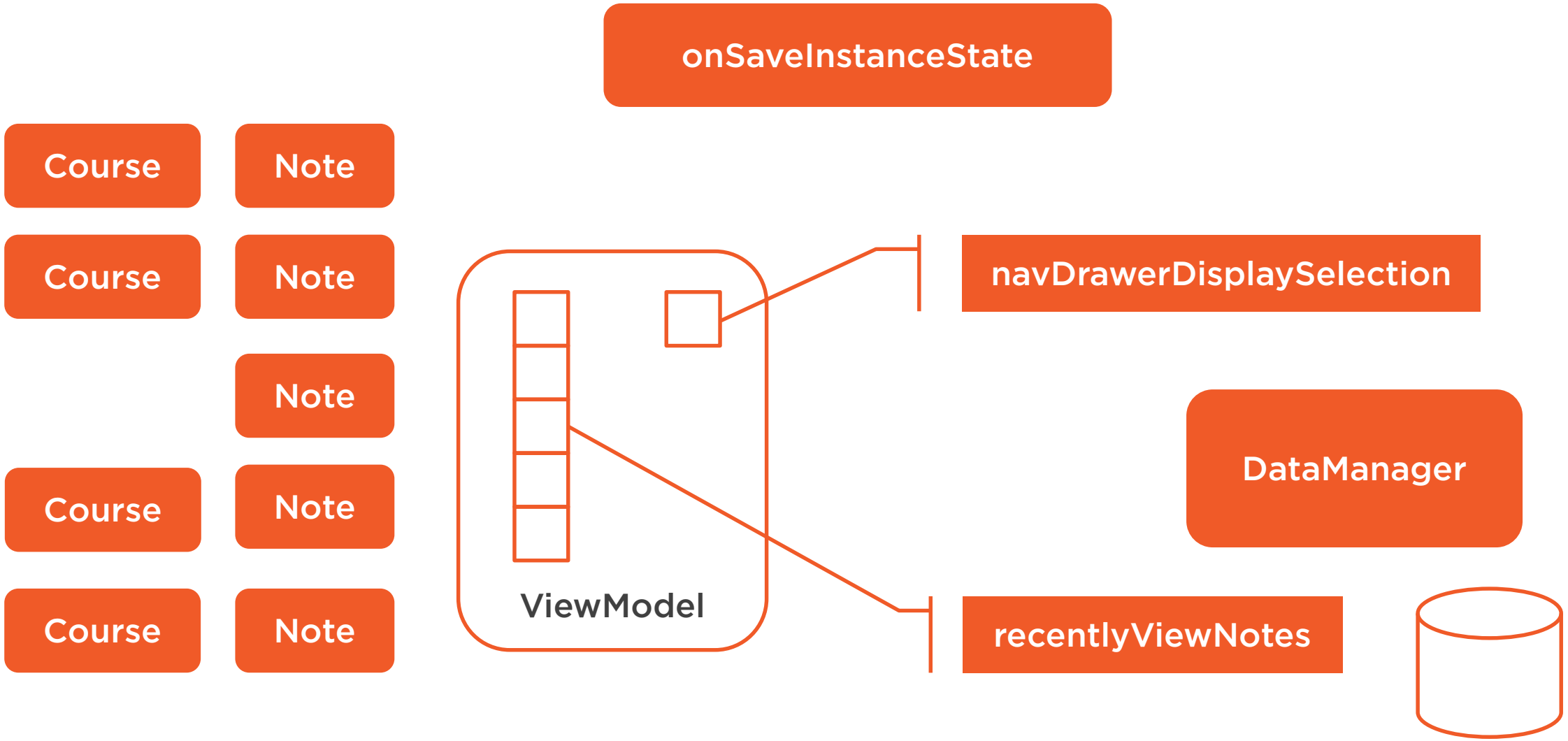
Our App's Data and State



Our App's Data and State



Our App's Data and State



onSaveInstance State Do's and Don'ts

Do save this

Simple state values

Id's of

Avoid saving this

Receives bundle with saved state values

Bundle is null on initial create

STUFF TO DO/NOT DO

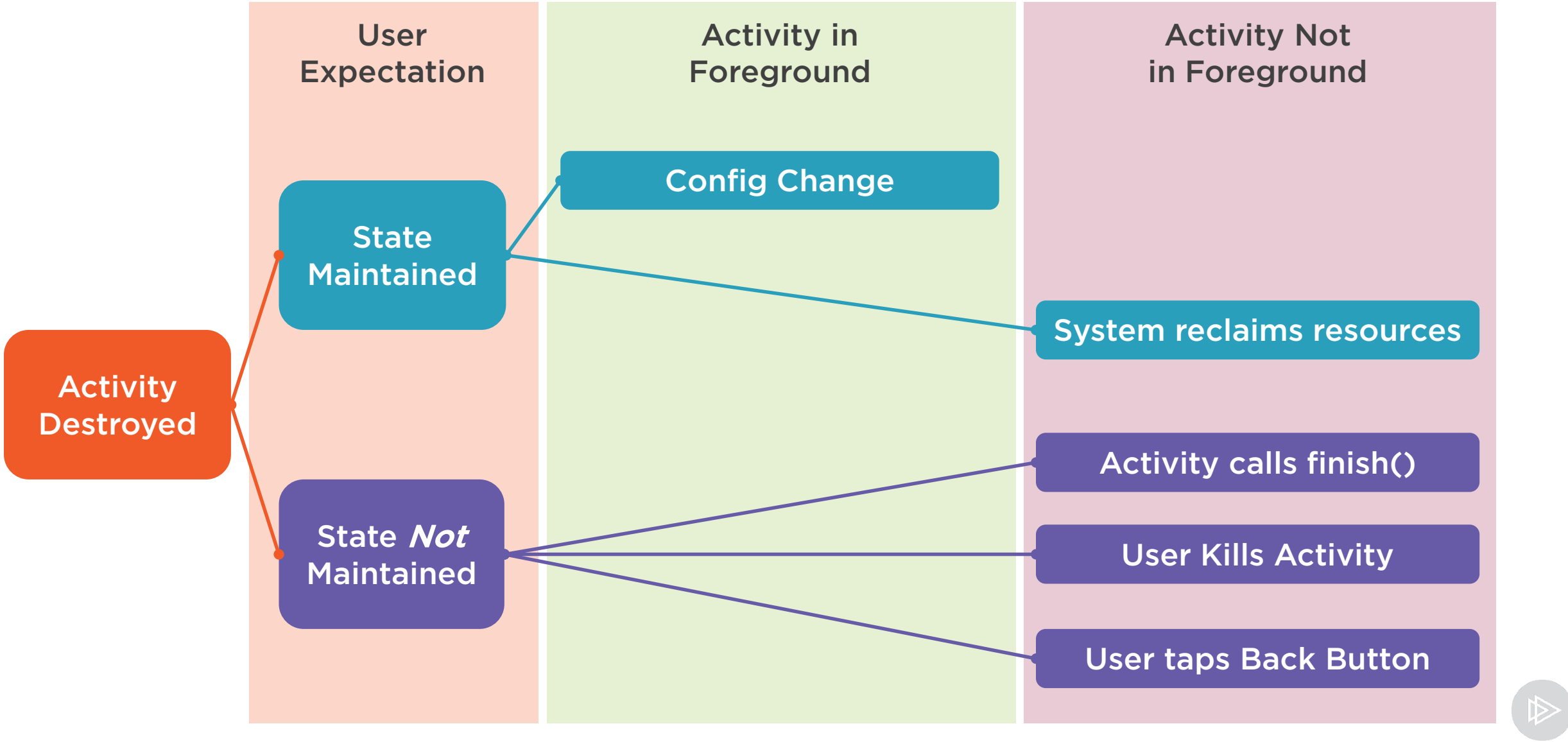
Don’t store references to views

Don’t store references to activities

Don’t store references to Lifecycles, etc.

Most views store their state

Activity Destruction and Activity State



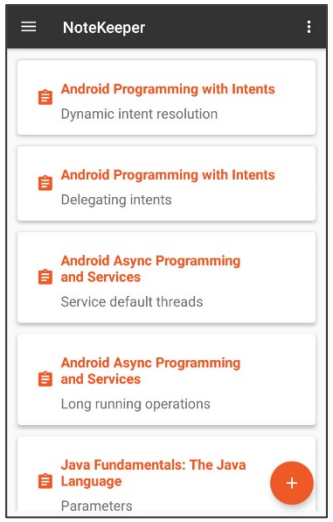
Activity Destruction and Activity State

User Expectation	User Interacting with Activity	User Not Interacting with Activity
	Config Change	
State Maintained		System reclaims resources
Activity Destroyed		
State *Not* Maintained		Activity calls finish()
		User Kills Activity
		User taps Back Button

Activity Destruction and Activity State

User Expectation	User Interacting with Activity	User Not Interacting with Activity
State Maintained	Config Change	System reclaims resources
Activity Destroyed		Activity calls finish()
State NOT Maintained		User Kills Activity
		Back Button

Activities – More Than Just a Pretty Face



App user experience provided by activities

Appear to user as simple app screens

But there's much more going on

Activities have a lifecycle

Our code needs to cooperate with that lifecycle

Life, Death, and Life of an Activity

Created

Has app-defined initial state

User Interaction

State reflects user's action

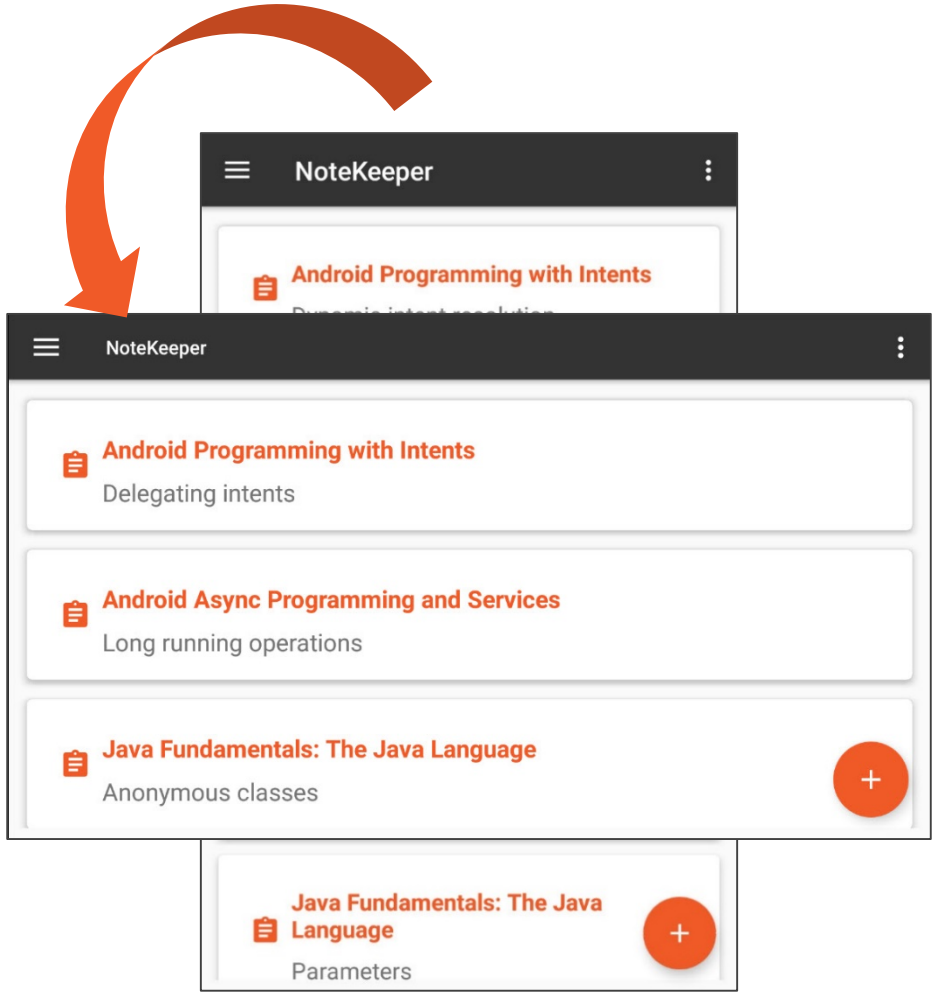
Destroyed

State stored within activity instance is lost

Recreated

Should restore previous state

Configuration Changes



Managing Activity State

Maintaining activity state

- Writing to a persistent store is expensive
- Need a better solution for maintaining state across configuration changes

ViewModel

- Stores activity state in-process
- State stored separate from the activity
- Extend ViewModel class to customize
- Add properties and methods specific to your activity's state requirements

Managing Activity State

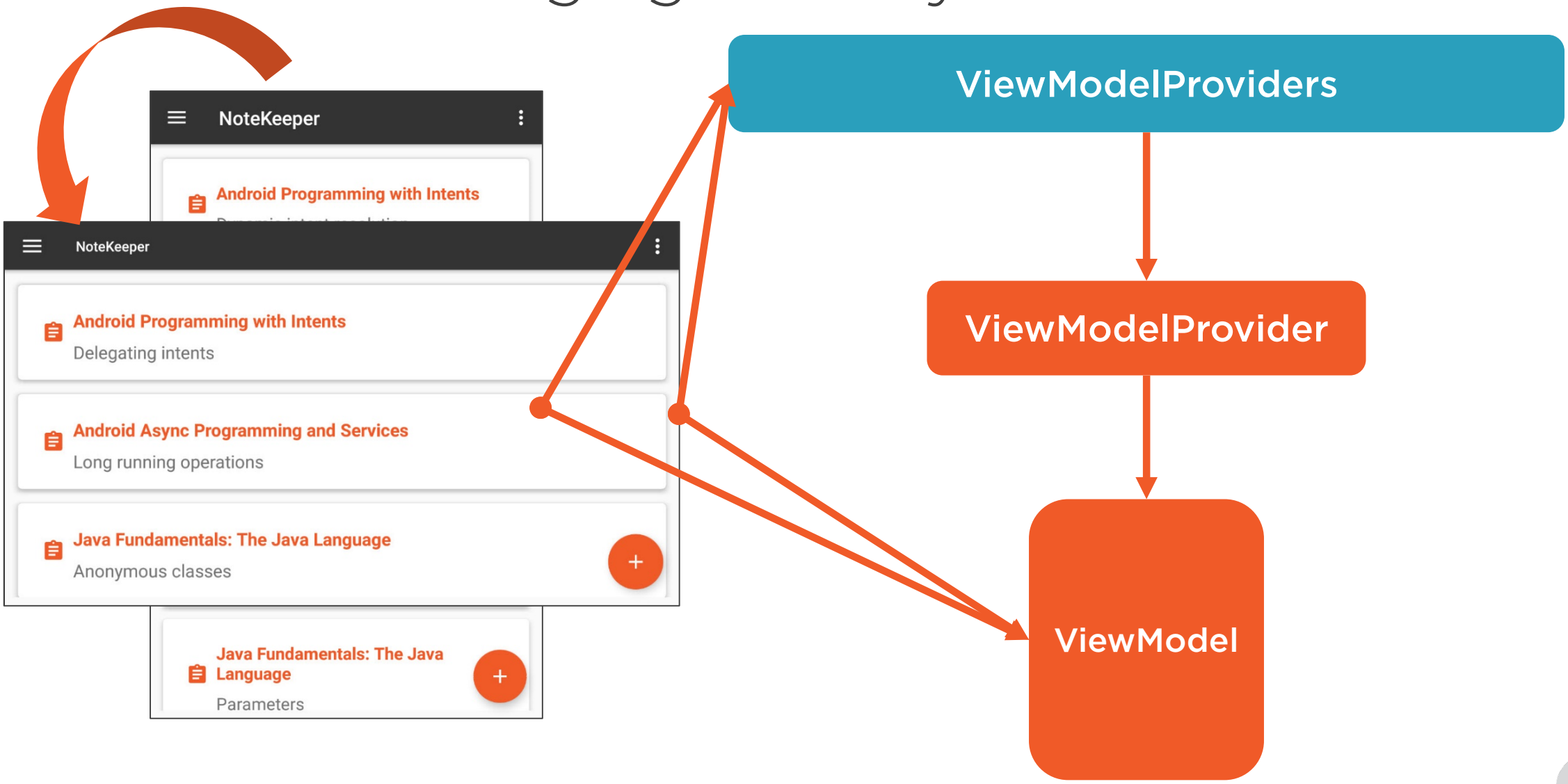
ViewModelProvider

- Manages ViewModel instances
- Creates new instance when needed
- Retrieves existing when available

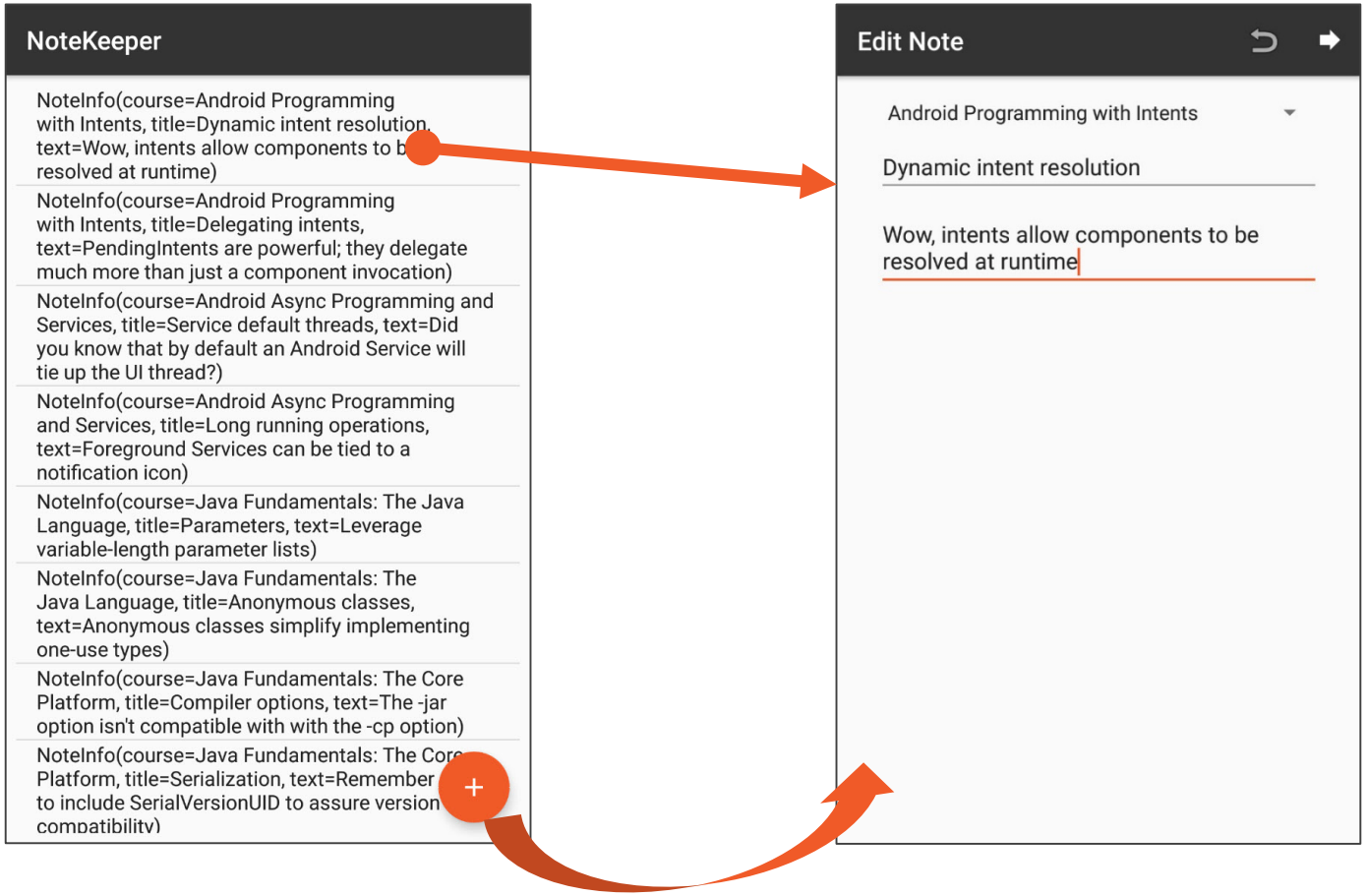
ViewModelProviders

- Manages association between activities and ViewModelProvider instances

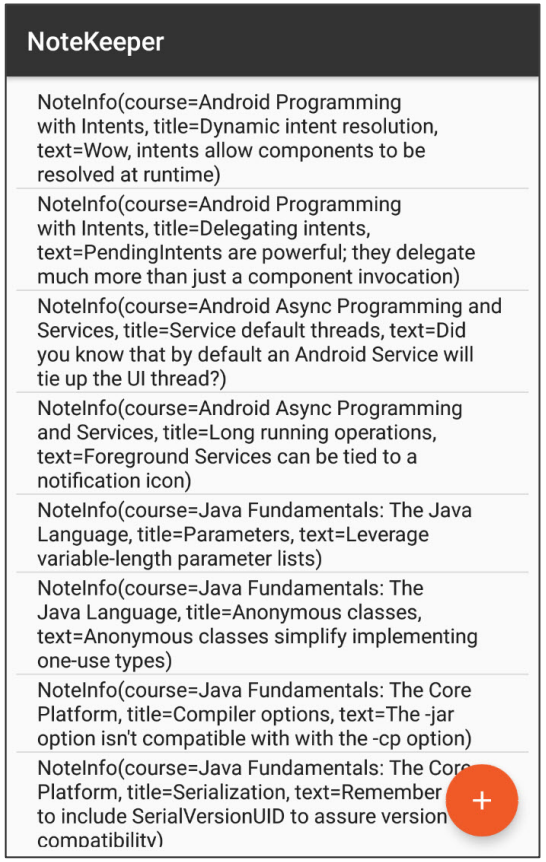
Managing Activity State



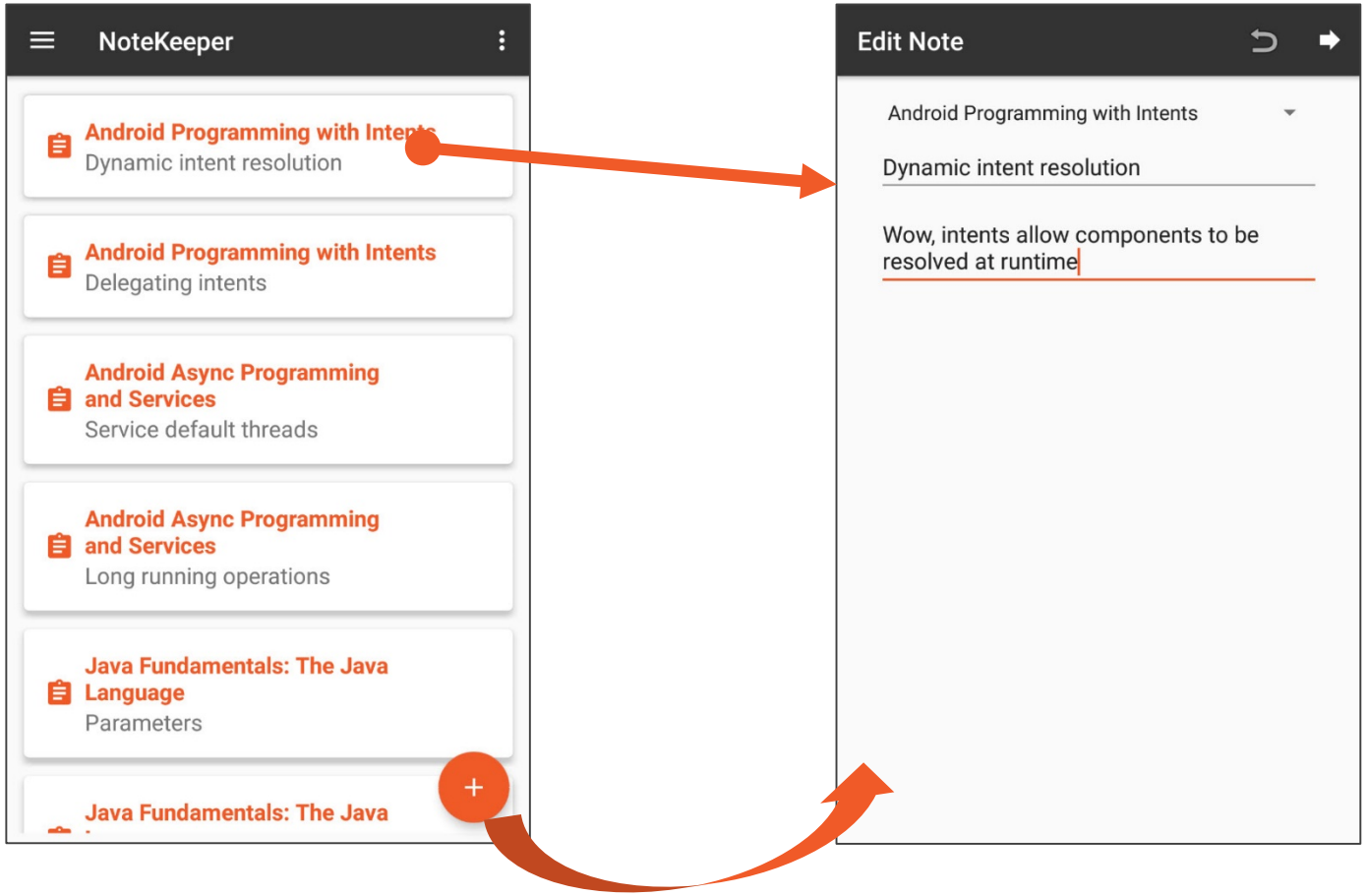
Overview of Our App

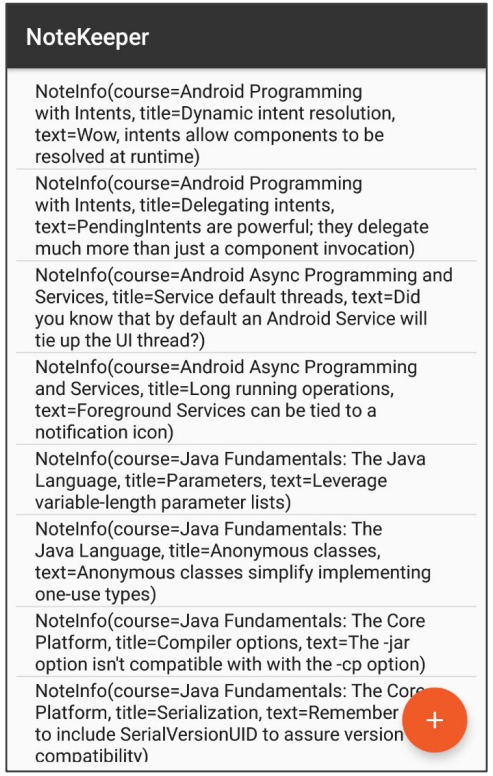


Overview of Our App



Overview of Our App





Displaying lists of data very common

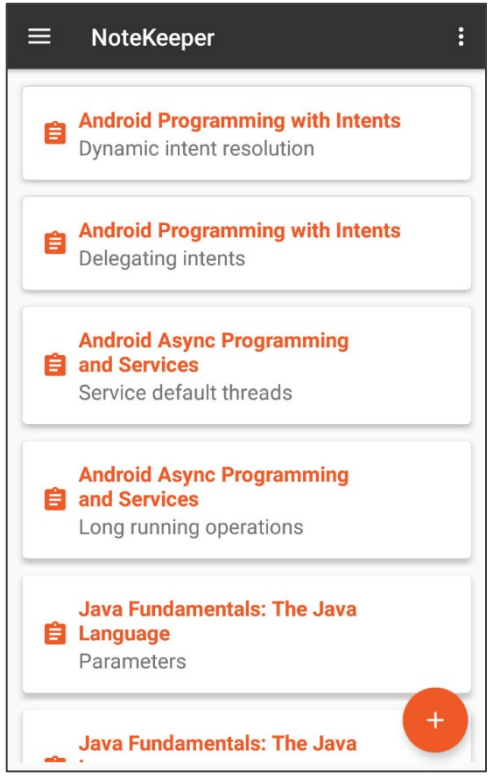
- Historically relied on ListView

ListView has limitations

- Always displays as vertical list
- Can be challenging to customize
- Performance challenges in some cases

ListView and modern app expectations

- Need a solution with more flexibility



RecyclerView is designed for modern apps

- Extremely flexible

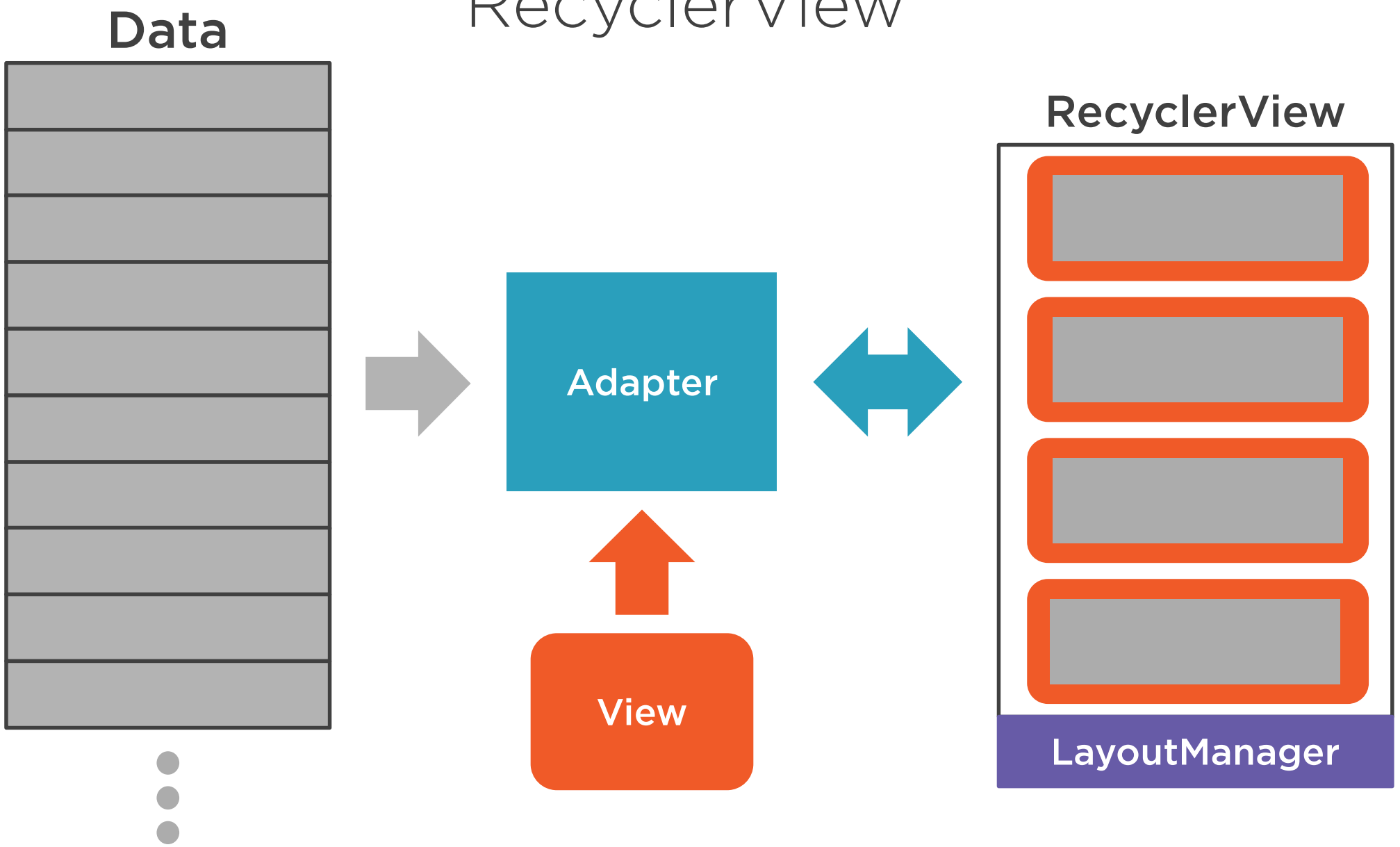
List display divided into distinct phases

- Each phase offers chance to customize

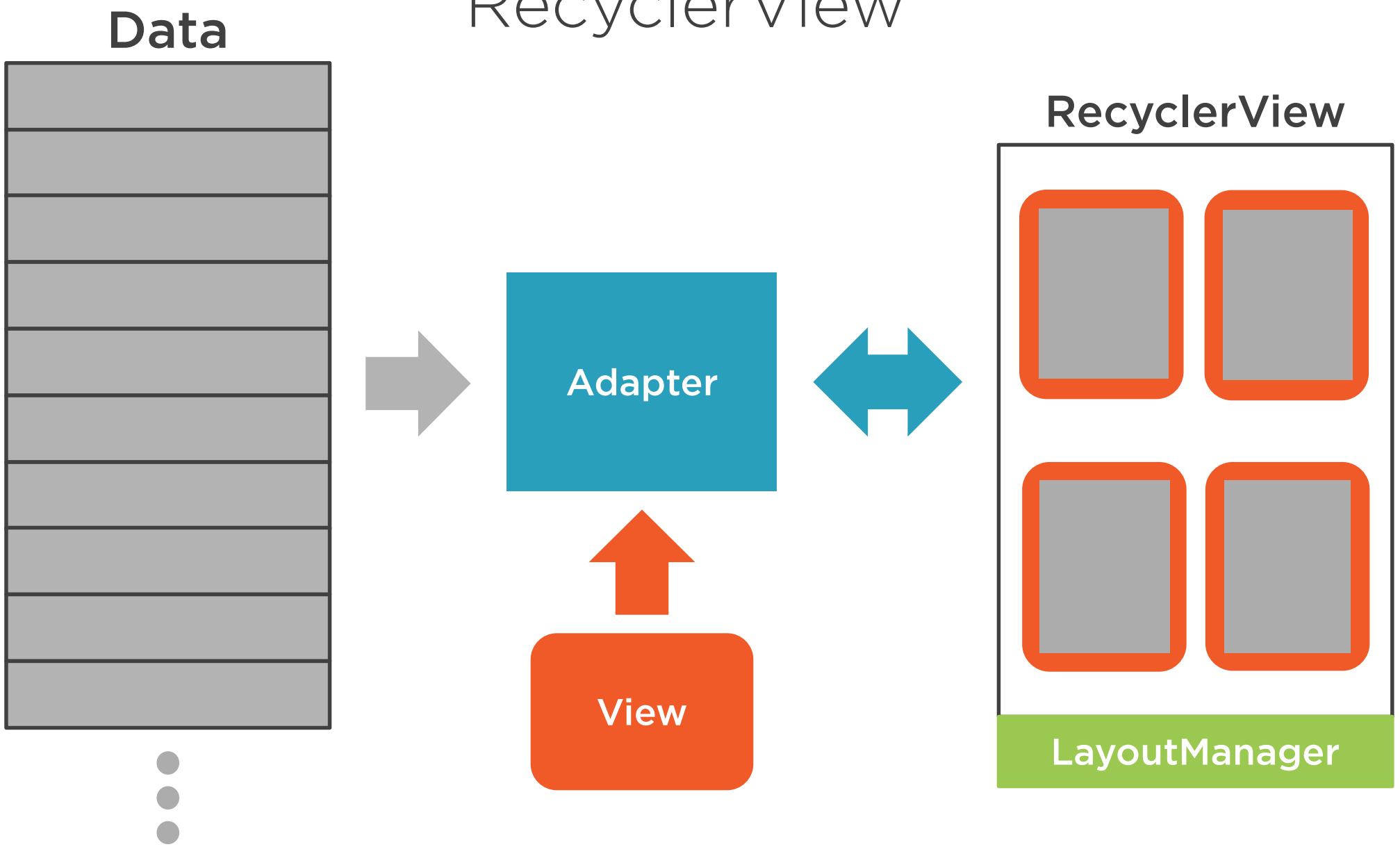
Provides efficient display management

- Separates details of data from display

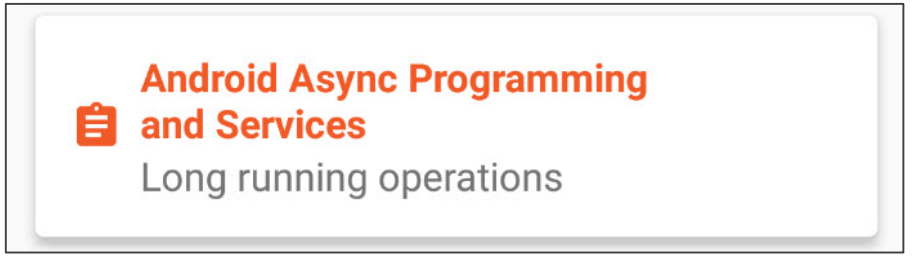
RecyclerView



RecyclerView



Developing RecyclerView Components



Design the RecyclerView

Handled much like any other view

Usually part of a layout resource

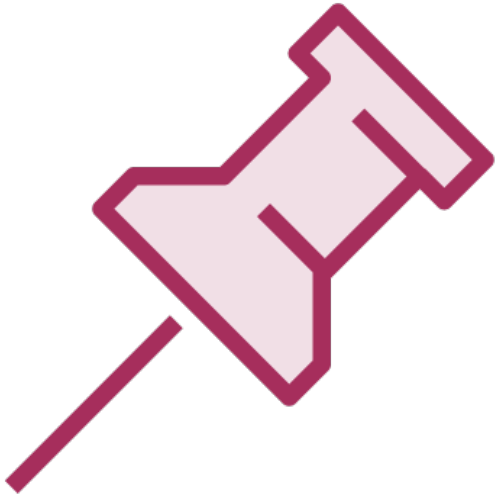
Design the item view

Controls appearance of individual item

Usually a layout resource

Separate resource from RecyclerView

Developing RecyclerView Components



Create and associate layout manager

Controls item arrangement and positioning

Create and associate adapter

Constructs item view instances

Manages data interaction

Associates data items with item views

Layout Manager

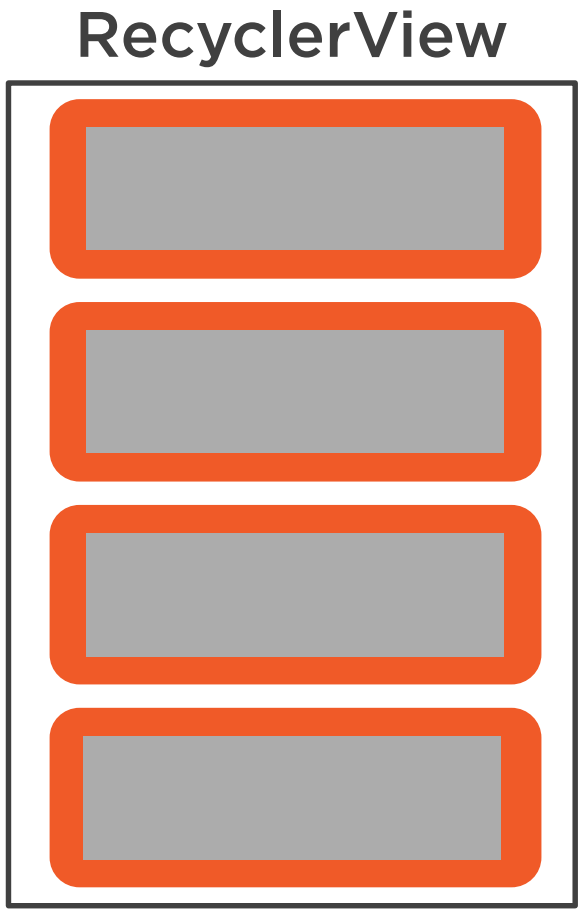
RecyclerView.LayoutManager

- Base class for layout managers
- Extend to create custom layout manager

Android provides several implementations

- Handle most common scenarios
- Support vertical & horizontal orientation

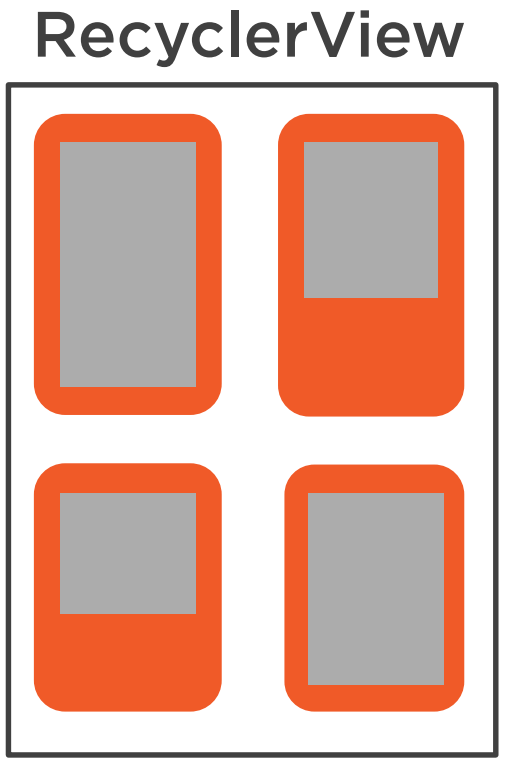
LinearLayoutManager



Items organized as linear list

- Similar to ListView

GridLayoutManager



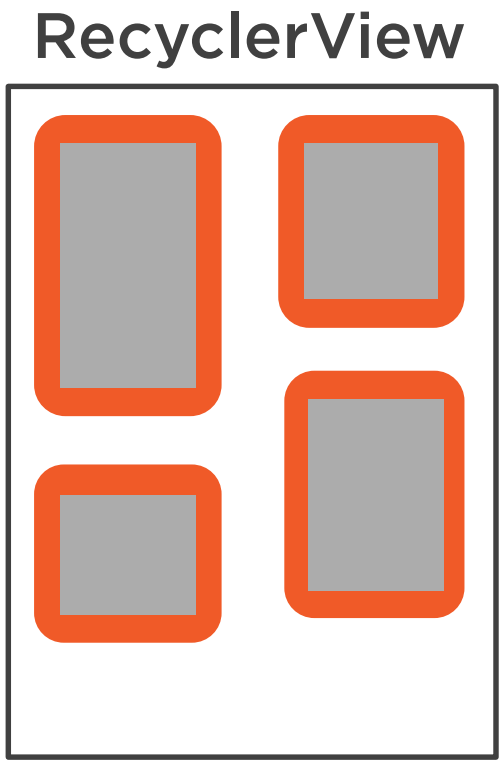
Items organized as a grid

- Adjacent items consistently sized

Can specify span

- Columns for vertical orientation
- Rows for horizontal orientation

StaggeredGridLayoutManager



Items organized as a grid

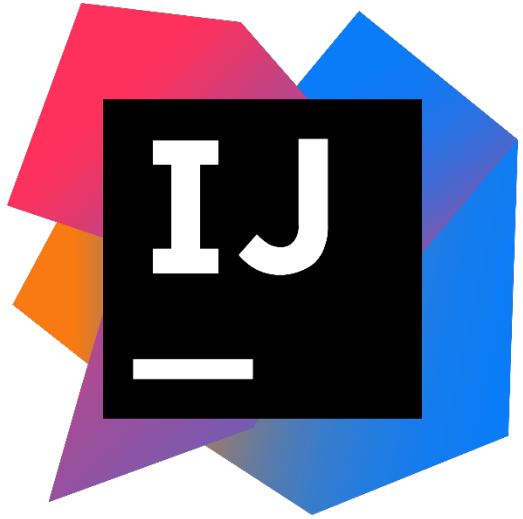
- Each item individually sized
- Can specify span

Android Studio

Android Studio

Primary development environment

Handles full dev cycle



Built on IntelliJ IDEA

Consistent developer experience

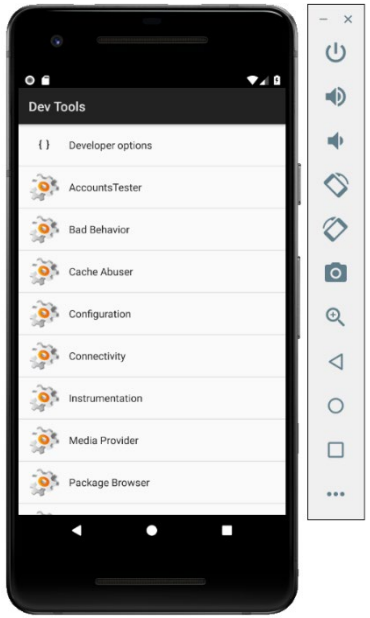
Android Studio Handles Installation Details

Java

Kotlin

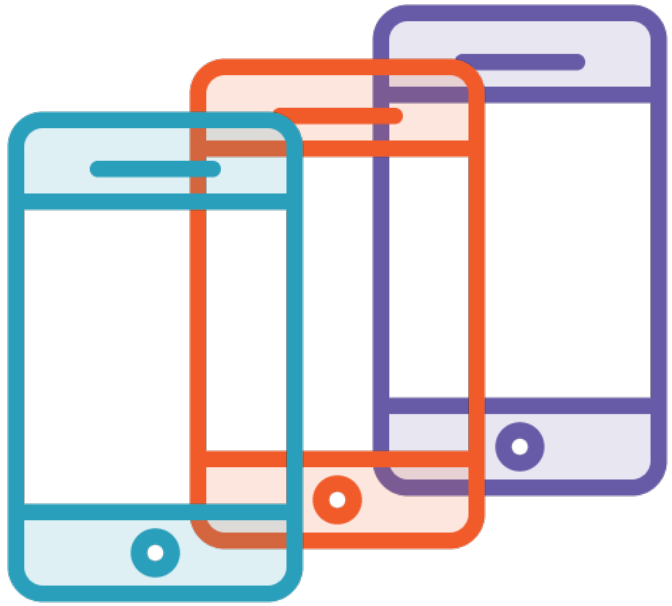
Android SDK

Running Code



Emulator

Run/debug apps directly on the desktop



Physical Devices

Run/debug apps on real device

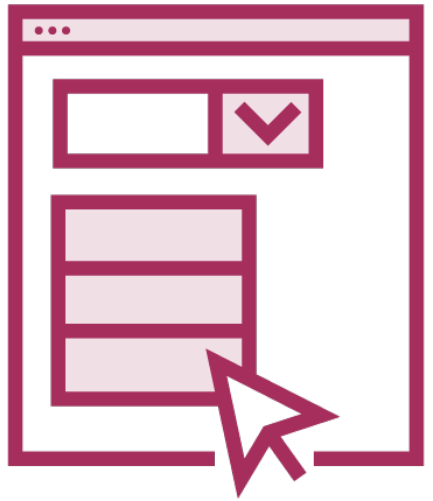
May require installation of USB drivers

Automated Testing

Unit Testing

Runs logic tests directly on desktop

Android Studio includes JUnit



Automated UI Testing

Runs interactive tests on emulator/device

Android Studio includes Espresso

Photo Circle with Text

Move the text boxes to keep the text aligned with the image

Photos works better than an icons

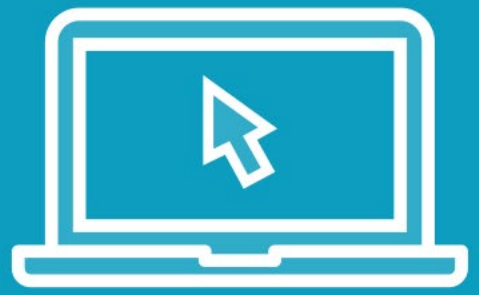
Photos permitted for commercial use

This Is the Module Title in Titlecase

Author Name
AUTHOR TITLE

@authortwitter www.authorsite.com

Demo



This bullet list is preset with animations

Use this layout to introduce your demo

How to do this one thing

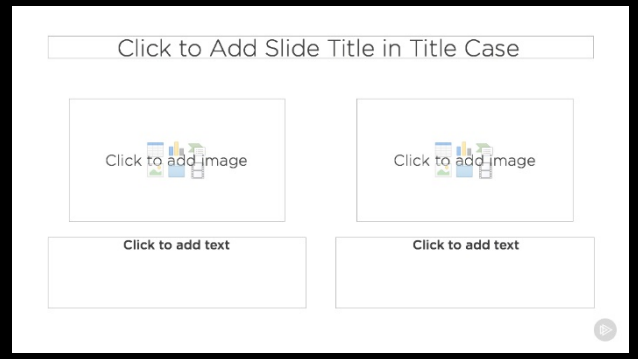
- Why we do it
- How we do it

Then there's that thing

Don't forget to do this

We'll finish it off with this thing

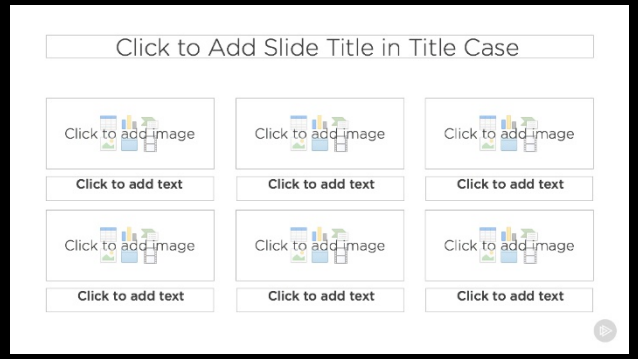
Using the **Image Chunking Slides**



Two Image Chunking

Three Image Chunking

Four Image Chunking



Six Image Chunking

These layouts can be used as an alternative to a bulleted list.

They're built specifically for **photos** or **graphics** and look especially awesome when you incorporate icons from the **Pluralsight Icon Library**.

See them in action in the next 4 slides.

Example of Image Chunking Two Items

Jill Anderson

Some information about this graphic goes here and four lines or fewer is best

John Doe

Some information about this graphic goes here and four lines or fewer is best

Example of Image Chunking Three Items

Clipboard
Some information goes here; three lines or fewer is best

Book
Some information goes here; three lines or fewer is best

Film
Some information goes here; three lines or fewer is best

Example of Image Chunking Four Items

Write

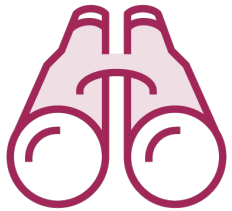
Create

Record

Learn

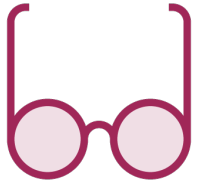
Example of Image Chunking Six Items

Address book

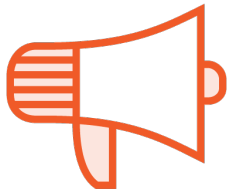


Binoculars

Camera



Eyeglasses

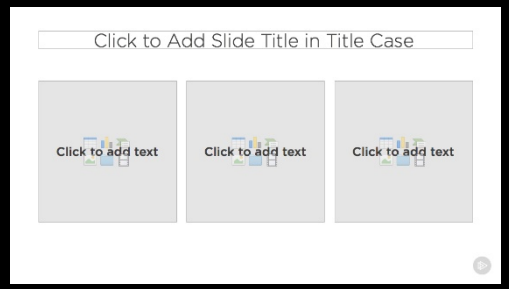


Megaphone

World

Using the **Text Chunking Slides**

Two Text Chunking



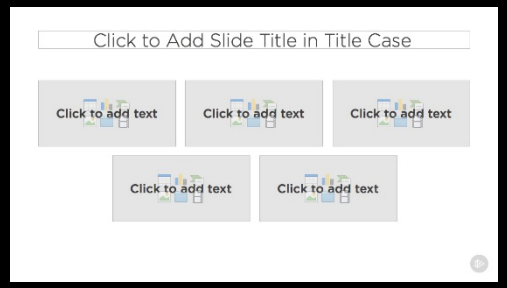
Three Text Chunking

These layouts are intended to group chunks of text. Among other uses, they can be a great alternative to a bullet list.

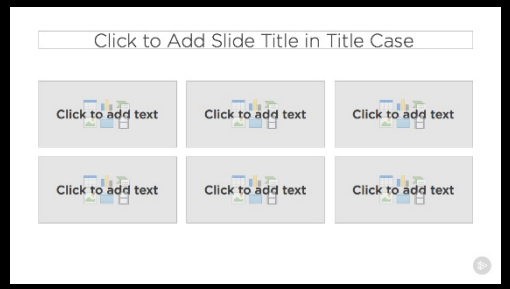
Use **animations** to bring focus to the point you're speaking on one at a time, and/or use **color** to group points together.

If you have more than six points to discuss, you may want to use a standard bullet list.

Four Text Chunking



Five Text Chunking



Six Text Chunking

We have provided some **example uses** of these layouts in the next few slides.

Text Chunking **Two Items**

Talking point one

Be concise and keep the text to four lines or fewer

Talking point two

Be concise and keep the text to four lines or fewer

Text Chunking Three Items

Talking point one

Be concise and keep the text to four lines or fewer

Talking point two

Be concise and keep the text to four lines or fewer

Talking point three

Be concise and keep the text to four lines or fewer

Text Chunking **Four Items**

This is the first talking point that should be kept to three lines or fewer

This is the second talking point that should be kept to three lines or fewer

This is the third talking point that should be kept to three lines or fewer

This is the fourth talking point that should be kept to three lines or fewer

Text Chunking Five Items

Talking point one

Keep the text to three lines or fewer

Talking point two

Keep the text to three lines or fewer

Talking point three

Keep the text to three lines or fewer

Talking point four

Keep the text to three lines or fewer

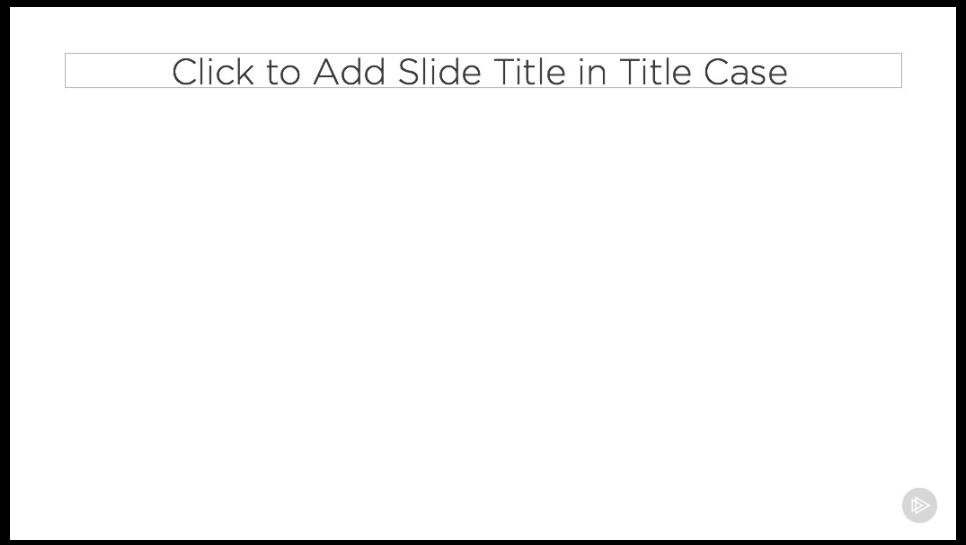
Talking point five

Keep the text to three lines or fewer

Today's Mobile World

iPhone	Nexus 5	Lumia 950 XL
iPad	Nexus 7	Surface

Using the **Title Only Slide**



Title Only

This is the slide you'll want to use when you just need a big space for a diagram, chart, or graphic.

Make sure you check out the training videos available on the **Author Kit** for design best practices.

If you need help bringing your ideas for this space to life, contact your Editor about getting help from one of our **Content Graphic Designers**. In most cases, you just need to submit a rough outline and let our designers work their magic. However, in some special cases, your Editor can get you in touch with a designer directly.

We included some possible starting points for you in the next few slides.

Remember, we are here to help!

Customer Acquisition and Loyalty

Observed higher sales

42%

Observed more loyal customers

70%

Timeline of Events

1940s
McCulloch and Pitts
Neural network theory

1956
Stephen Cole Kleene
Regular events/sets

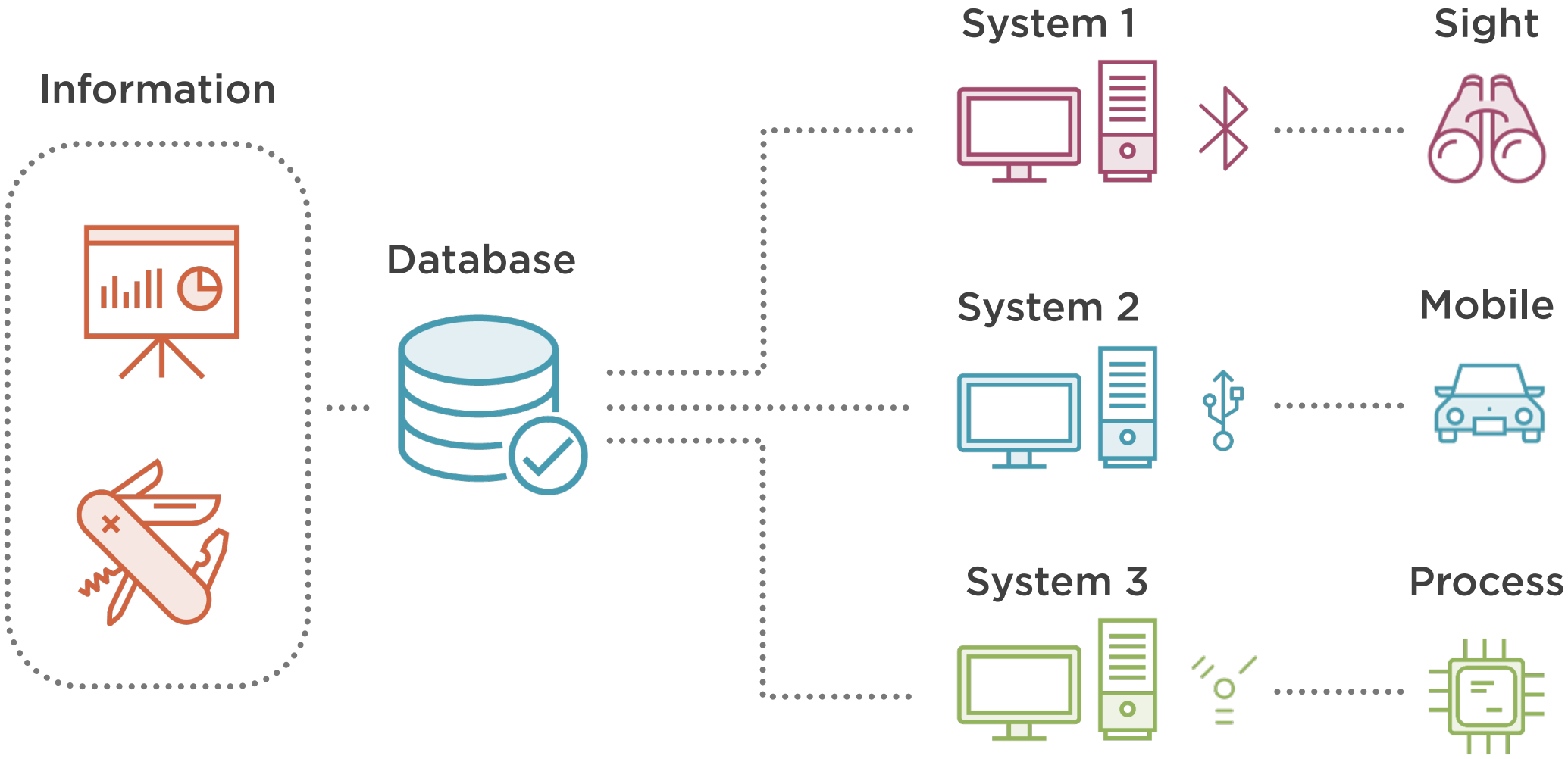
1968
Ken Thompson
First in computing

1973
Ken Thompson
First release of *grep*

1986
Henry Spencer
regex library

1987
Larry Wall
Integration into Perl

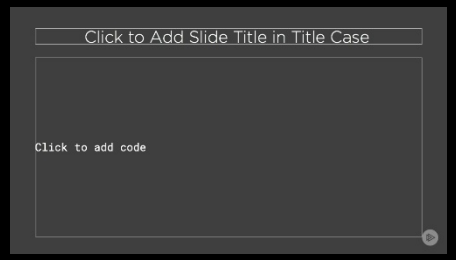
Title Only Layout Example



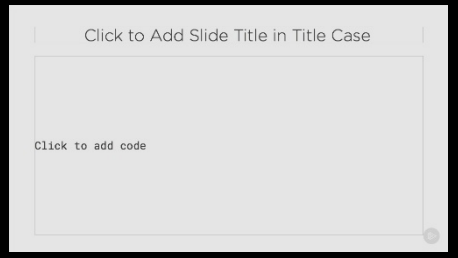
Using the **Code Slides**

Code Top (Dark)

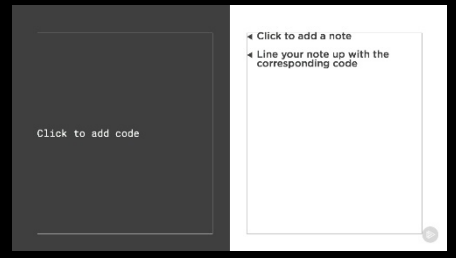
Code Top (Light)



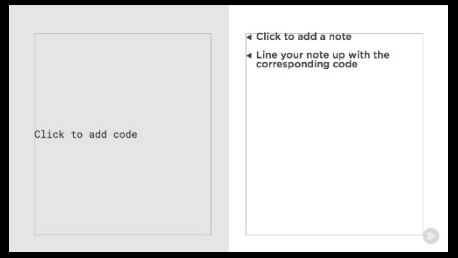
Code (Dark)



Code (Light)



Code Notes (Dark)



Code Notes (Light)

Code Top Layouts

Use when you need a slide title and info about your code

Code Layouts

Best for larger code snippets

Code Left Layouts

Great for annotating code structure

Make use of the color palette to highlight code.

We recommend using the **Roboto Mono** typeface for your code slides. However, if you use a different font for code in your demos, feel free to use that instead to reinforce a consistent look.

```
<div class="row carousel-indicators">

            <div style="background-color:red;" class="col-
md-4" data-target="#homeCarousel" data-slide-to="0"
class="active”>

<div class="row carousel-indicators”>
```

Slide Title in Titlecase

Information about the code above

```
<div class="row carousel-indicators">

            <div style="background-color:red;" class="col-
md-4" data-target="#homeCarousel" data-slide-to="0"
class="active”>

<div class="row carousel-indicators”>
```

Slide Title in Titlecase

Information about the code above

Code Snippet on Dark

```
<div class="row carousel-indicators">

            <div style="background-color:red;" class="col-
md-4" data-target="#homeCarousel" data-slide-to="0"
class="active">

            </div>

            <div style="background-color:green;"
class="col-md-4" data-target="#homeCarousel" data-slide-
to="1">

            </div>
```

Code Snippet on Light

```
<div class="row carousel-indicators">

            <div style="background-color:red;" class="col-
md-4" data-target="#homeCarousel" data-slide-to="0"
class="active">

            </div>

            <div style="background-color:green;"
class="col-md-4" data-target="#homeCarousel" data-slide-
to="1">

            </div>
```

```
Put code on this side

var proto = {
foo:’Hello World’
};


function Bar(){}
Bar.prototype = proto;


var baz = new Bar();


console/log(baz.foo);
```

- Line up with these notes
- Set up prototype object
- Constructor function and set prototype property
- Create instance
- Call inherited member

```
Put code on this side

var proto = {
foo:'Hello World'
};


function Bar(){}
Bar.prototype = proto;


var baz = new Bar();


console/log(baz.foo);
```

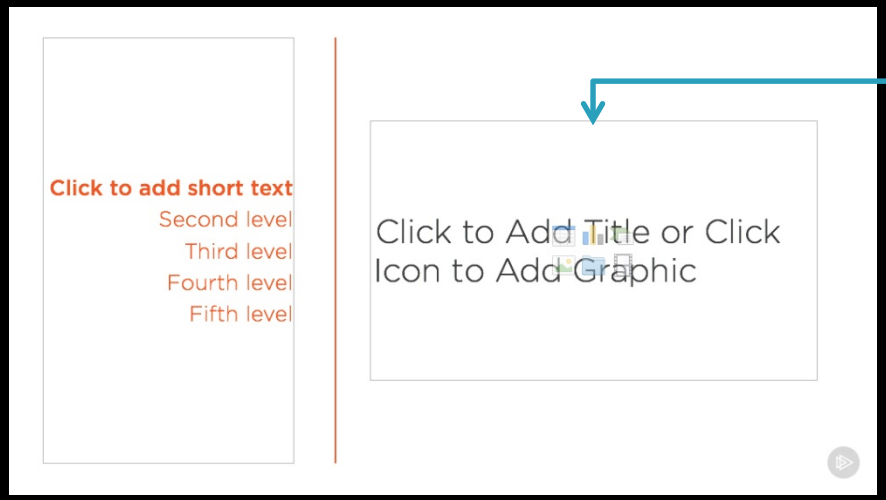
- Line up with these notes
- Set up prototype object
- Constructor function and set prototype property
- Create instance
- Call inherited member

Using Bullet List Slides

We've provided some bullet list layouts to accommodate various quantities of information.

Content left | Title/Image right

Intended for bullet text that is shorter and titles/images that are larger

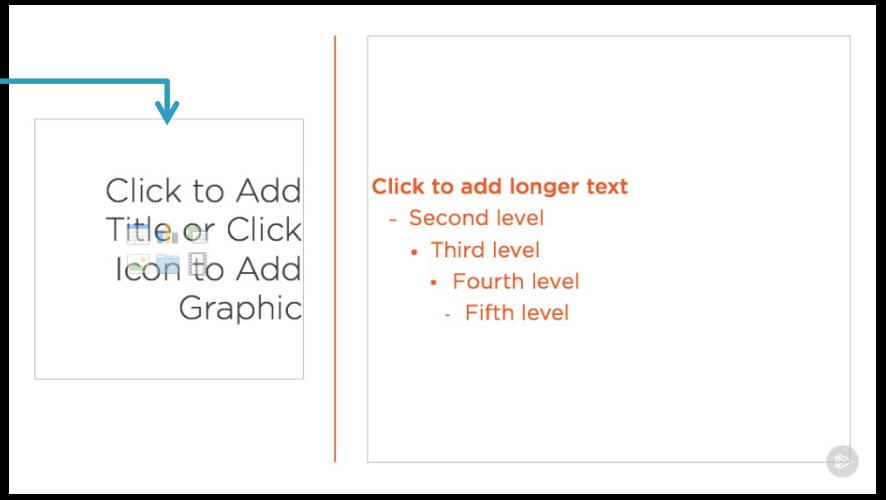


Content | Image/Title

Remember, you can use **text** or **images** in these placeholders.

Title/Image left | Content right

Intended for bullet text that is longer and titles/images that are smaller



Image/Title | Content

Animation built in

Bullet alternative

Sentence fragments

List of things

Procedure list

Talking points

Title or Relevant Graphic

Animation built in

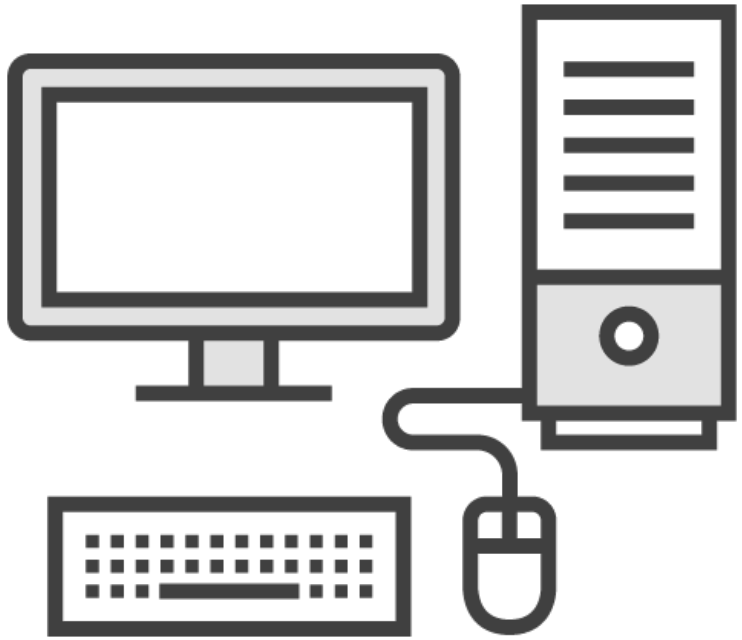
Bullet alternative

Sentence fragments

List of things

Procedure list

Talking points



Title or Relevant Graphic

Animation built in

Bullet alternative

Room for a bit more text

Use this layout for

- Longer sentence fragments
- List of things
- Procedure list
- Talking points

Animation built in

Bullet alternative

Room for a bit more text

Use this layout for

- Longer sentence fragments
- List of things
- Procedure list
- Talking points

Title Space with Image

Animation built in

Bullet alternative

Room for a bit more text

Use this layout for

- Longer sentence fragments
- List of things
- Procedure list
- Talking points

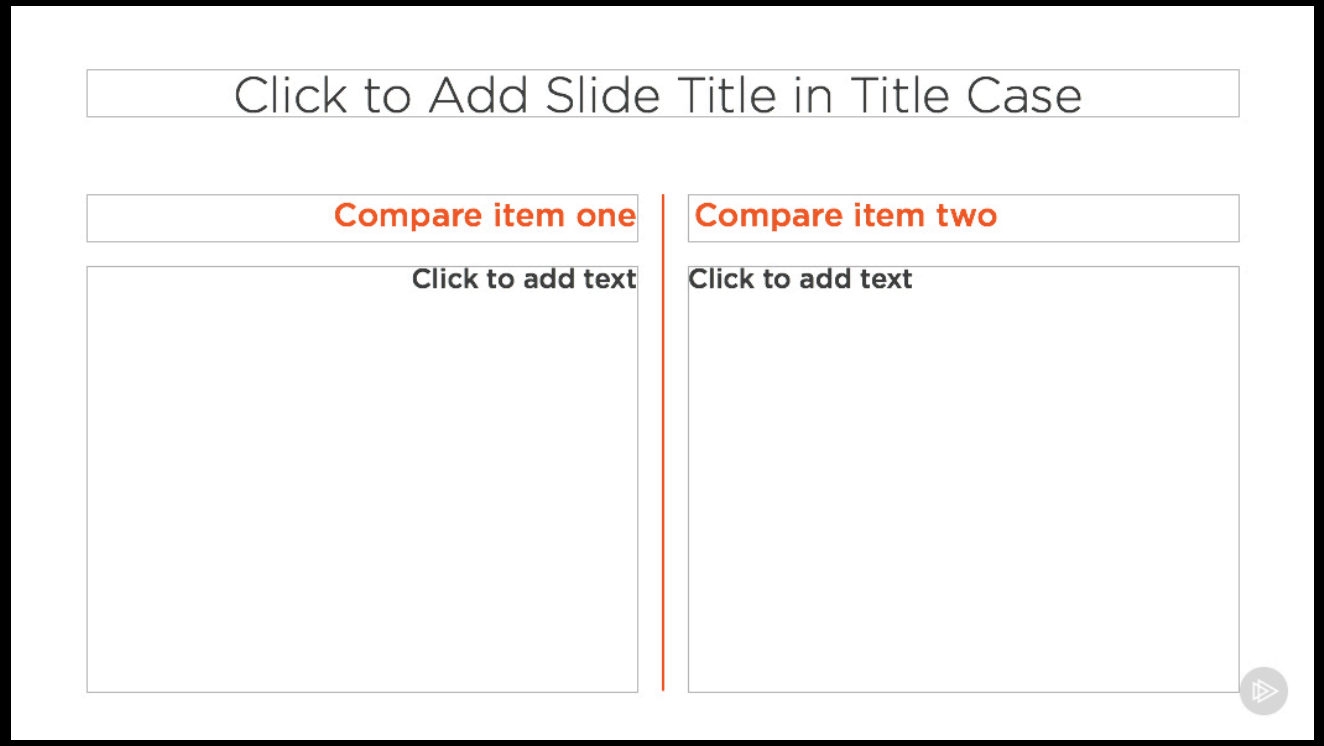
Graphic on left should fill the entire space

- Graphic must be high quality and royalty free

Graphic and text animation is built in

Comparison Slide

Use this slide if you need to compare single items or groups of items.



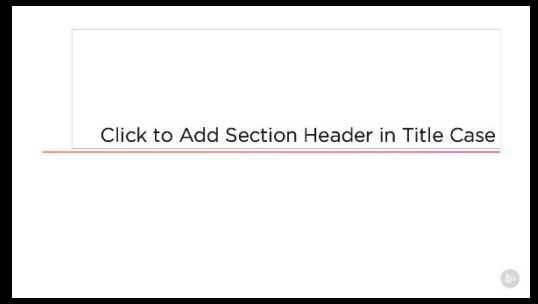
Comparison Example

Functional group	Objectives
Configure and administer security	Manage vSphere storage virtualization
Configure advanced networking	Configure software-defined storage
Configure advanced storage	Configure vSphere storage multipathing and failover
Administer and manage resources	Perform advanced VMFS and NFS configurations and upgrades
Configure availability solution	
Deploy and consolidate vSphere	

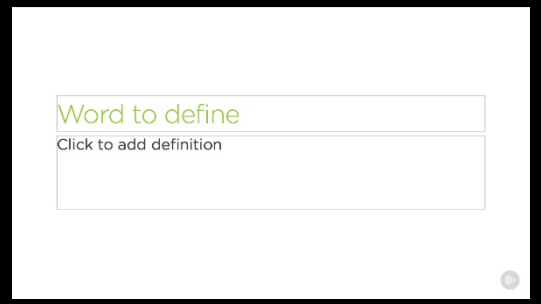
Other Slides

The following self-explanatory slides are a good way of adding diversity into the flow of your course.

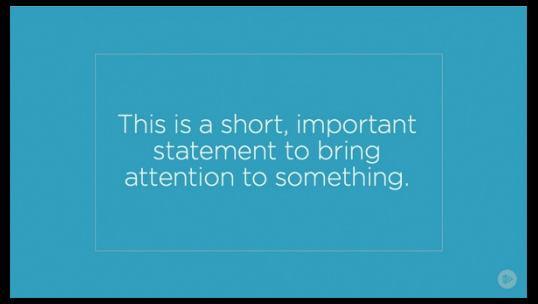
Use them purposefully.



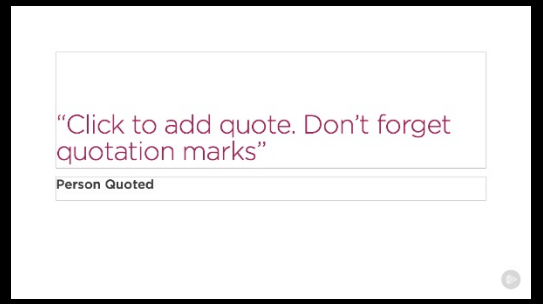
Section Header



Definition



Important Statement



Quotation

Section Heading

Word Definition

Here is where you put the definition. This is one of the few places where complete sentences are appropriate. Be sure to cite your source.

This is a short, important statement to bring attention to something.

“Using quotes in your slides can be powerful if used sparingly.”

Heather Ackmann