



Deep Dive Into Android
State Restoration

TWITTER

@cyrilmottier

WEBSITE

cyrilmottier.com

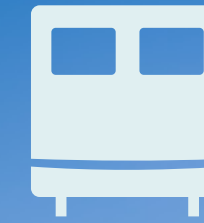


capitaine train 🚂



LC
54.67
WHR

capitaine train



capitaine train interface showing search options and results for a trip from Lyon to Paris.

Where can we take you?

→ Lyon
← Paris

Monday 22 September 2014 from 06:00
One-way

Cyril

[Add a discount code](#) [SEARCH](#)

Select a departure station

- Lyon
- Paris-Gare-de-Lyon
- Lyon Part-Dieu
- Lyon Perrache
- Lyon St-Exupéry TGV
- Lyon Jean Macé
- Lyon Vaise
- Lyon St-Paul

Your Lyon → Paris trip

Cheapest Semi-flex Flexible Fastest routes only

Monday 22 September 2nd class 1st class

Time	Price
05:16 → 10:22	€32.00
05:16 → 10:22	€48.00

Trip

05:16 Lyon Part-Dieu
10:22 Paris-Gare-de-Bercy
TER 17774 2nd

Passengers

Cyril €32.00

Billet Abonnement Fréquence: Détenir une Carte Fréquence (justificatif à présenter à bord du train). Billet échangeable sans frais

capitaine train

capitaine train interface on a laptop screen. The top navigation bar includes icons for home, search, cart, tickets, help, and user profile (Cyril).

Where can we take you?

Search input: Lyon
Dropdown: Paris

Date: Monday 22 September 2014 from 06:00
Type: One-way
User: Cyril

Buttons: Add a discount code, SEARCH

Select a departure station

- Lyon
- Paris-Gare-de-Lyon
- Lyon Part-Dieu
- Lyon Perrache
- Lyon St-Exupéry TGV
- Lyon Jean Macé
- Lyon Vaise
- Lyon St-Paul

Your Lyon → Paris trip

Fares: Cheapest (selected), Semi-flex, Flexible. Fastest routes only:

Monday 22 September

Time	Class	Price
05:16 → 10:22	2 nd class	€32.00
	1 st class	€48.00

Trip

05:16 Lyon Part-Dieu
10:22 Paris-Gare-de-Bercy
TER 17774 2nd

Passengers

Cyril €32.00
Billet Abonnement Fréquence: Détenir une Carte Fréquence (justificatif à présenter à bord du train). Billet échangeable sans frais

capitaine train interface on a smartphone. The screen displays a ticket summary for a one-way trip from Lyon to Paris on Monday, September 22, for €35.00.

Ticket Lyon → Paris

One-way on Mon, Sep 22 €35.00

Monday, September 22

06:04 Lyon Part-Dieu
2nd Carriage 8, seat 96 Upstairs, twin side-by-side, window SNCF TGV 6602

08:13 Paris Gare-de-Lyon

Your non-transferable E-Ticket RWMRTB is stored on the Carte Voyageur, which must be presented upon inspection.

Cyril Mottier - Outward
Present barcode during inspection.

Fare terms and conditions

Cyril Mottier €35.00
Carte Jeune
Exchange and refund free of charge until the eve of departure, with a deduction of €3 on the day of departure.

capitaine train

capitaine train interface on a laptop screen. The top navigation bar includes icons for a train, search, cart, tickets, help, and a user profile for Cyril with 4 points. The main content area is divided into two columns. The left column, titled 'Where can we take you?', contains a search input with 'Lyon' selected, a dropdown for 'Paris', a date selector for 'Monday 22 September 2014' starting at '06:00', a 'One-way' trip type, and the user name 'Cyril'. A green 'SEARCH' button is at the bottom. The right column, titled 'Select a departure station', lists several stations: Lyon, Paris-Gare-de-Lyon, Lyon Part-Dieu, Lyon Perrache, Lyon St-Exupéry TGV, Lyon Jean Macé, Lyon Vaise, and Lyon St-Paul. Below this, a section titled 'Your Lyon → Paris trip' features a 'Fares' slider set to 'Cheapest' and a 'Fastest routes only' checkbox. It shows a table of options for Monday 22 September, with a selected route from 05:16 to 10:22 in 2nd class for €32.00. A 'Passengers' section lists 'Cyril' for €32.00.

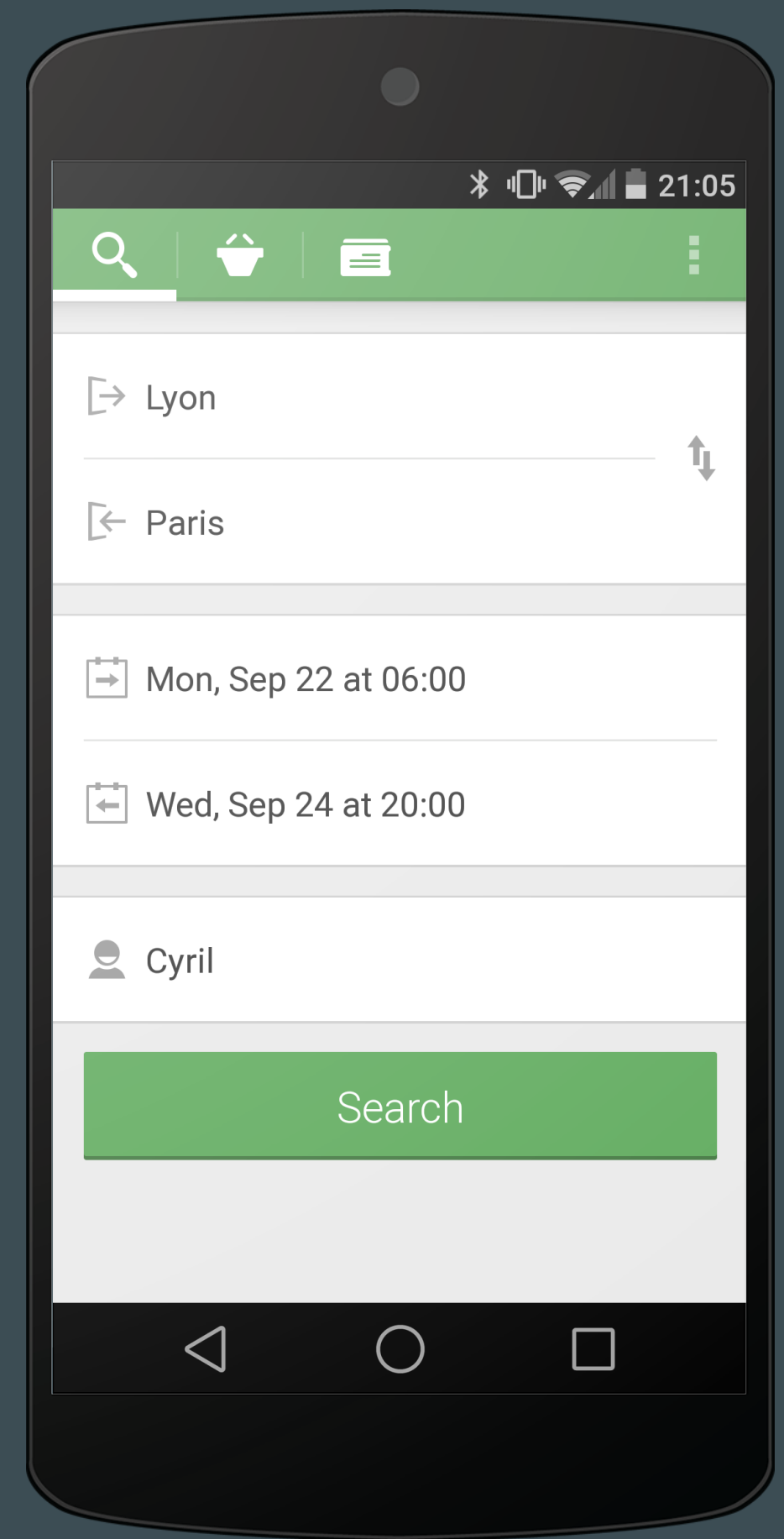
Smartphone displaying a 'Ticket Lyon → Paris' for €35.00 on Monday, September 22. The ticket details include a departure at 06:04 from Lyon Part-Dieu (2nd class, Carriage 8, seat 96) and an arrival at 08:13 at Paris Gare-de-Lyon. It notes that the non-transferable E-Ticket RWMRTB is stored on the Carte Voyageur. The passenger is identified as Cyril Mottier - Outward.

Tablet displaying a 'Paris-Lille' ticket for 10:46 departure on TGV 7033, Carriage 6, seat 56.



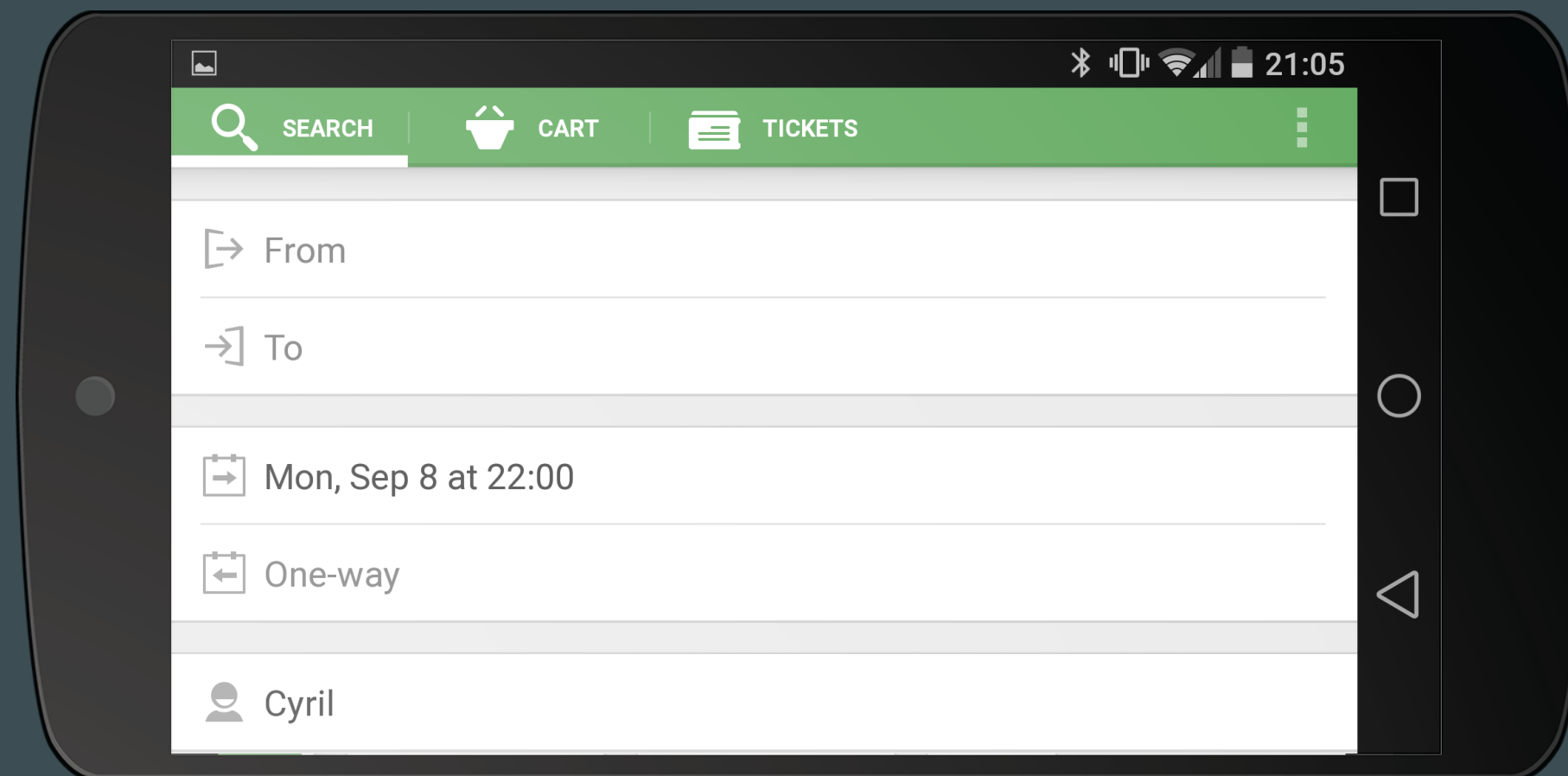
The story of a newbie
Android developer

Kevin has just developed his first
Android app





He discovers an annoying bug:
Fields are cleared on rotate



3 options



3 options

Don't care

3 options

Don't care | Block orientation

3 options

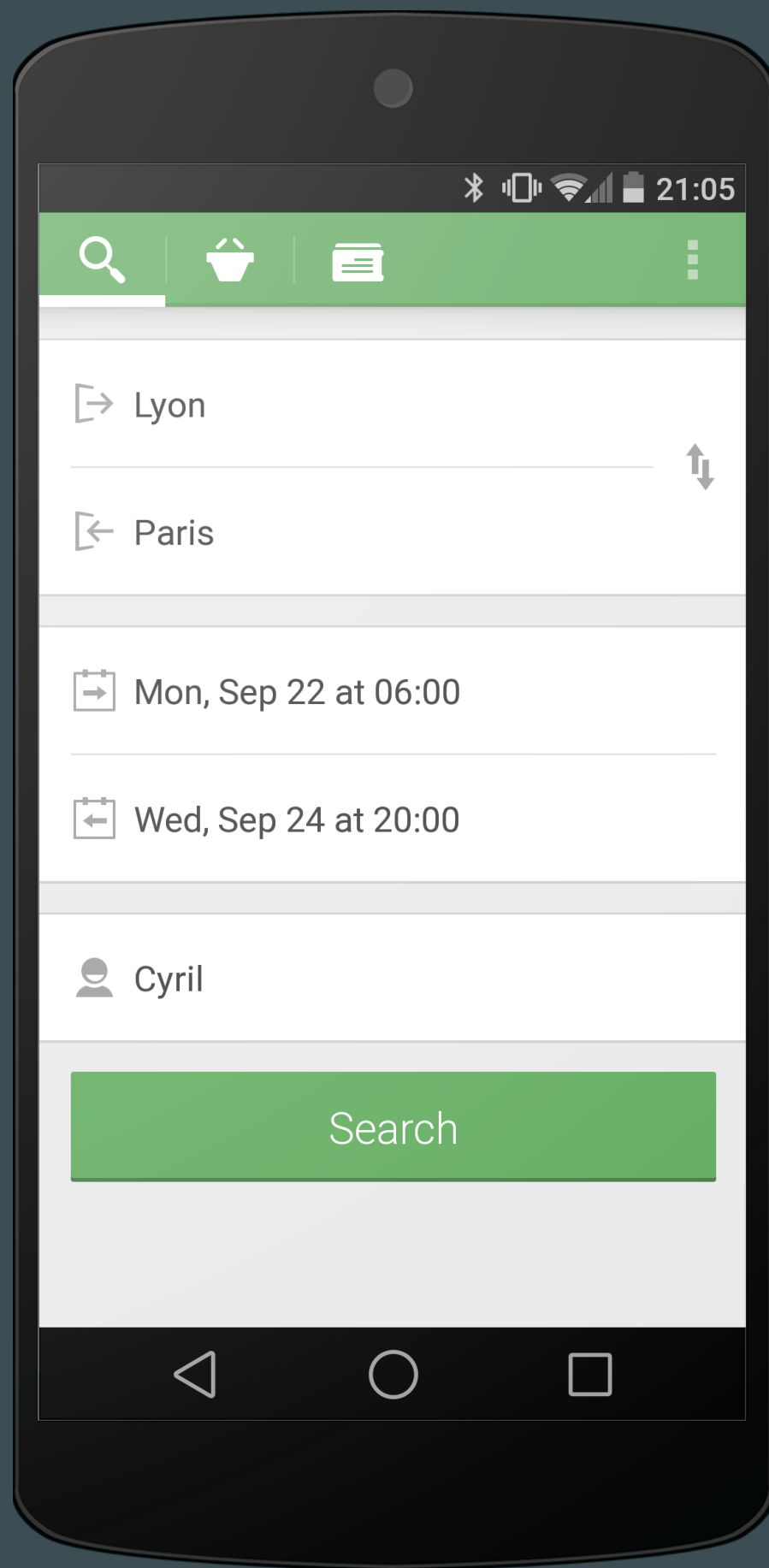
Don't care | Block orientation | Use configChanges

Hint: all options are 

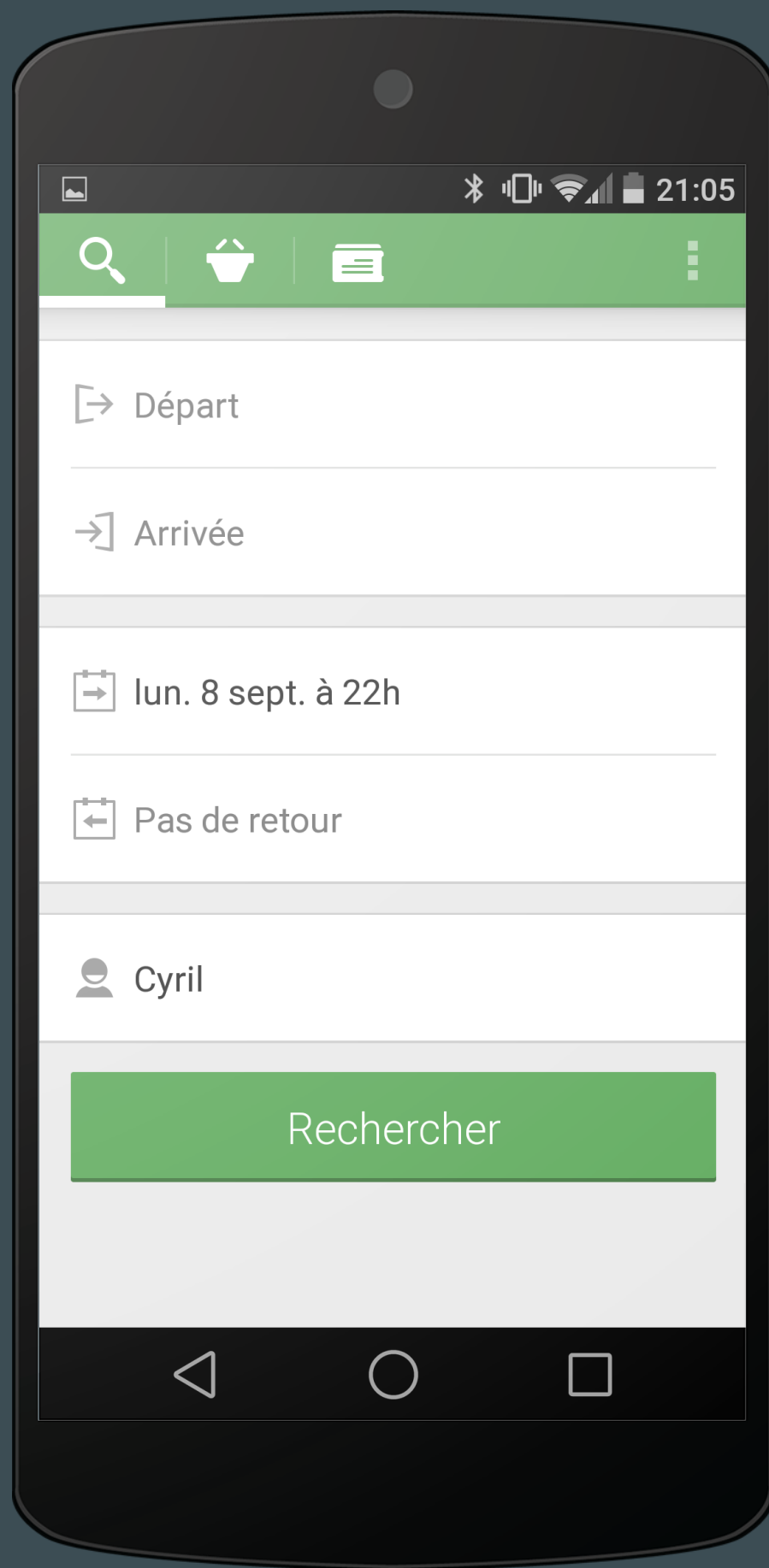
```
1 <activity
2     android:name=".HomeActivity"
3     android:configChanges="orientation">
4
5     <!-- Some sweet IntentFilters. -->
6
7 </activity>
```




Kevin's satisfied



Still having issues on...



Still having issues on...

language changes

```
1 <activity
2     android:name=".HomeActivity"
3     android:configChanges="orientation|locale">
4
5     <!-- Some annoying IntentFilters. -->
6
7 </activity>
```


Angry Kevin is



ANGRY!

```
1 <activity
2     android:name=".HomeActivity"
3     android:configChanges="orientation|locale|
      mcc|mnc|touchscreen|keyboard|
      keyboardHidden|navigation|uiMode|
      screenLayout|fontScale|screenSize|
      smallestScreenSize">
4
5     <!-- Some fuc*** IntentFilters. Arrggh! -->
6
7 </activity>
```



The nightmare continues...

Still having issues when moving the app to the background



God save the

STATE

State restoration
key components

The container Parcel



The container

`Parcel`



The content

`Primitives types`

`Primitives arrays`

`Parcelable`

The content

```
1 parcel.writeInt(1);  
2 parcel.writeLong(2L);  
3 parcel.writeFloat(3F);  
4 parcel.writeString("Hi!");
```

Primitives types

Primitives arrays

Parcelable

The content

Primitives types

Primitives arrays

Parcelable

```
1 parcel.writeIntArray(new int[]{1, 2, 3});
2 parcel.writeLongArray(new long[]{1L, 2L, 3L});
3 parcel.writeDoubleArray(new double[]{1, 2, 3});
4 parcel.writeStringArray(new String[]{
5     "Hi", "Droidcon", "guys!"
6 });
```

```
1 public final class Suggestion implements Parcelable {
2
3     public final String id;
4     public final String name;
5     public final int type;
6
7     public Suggestion(String id, String name, int type) {
8         this.id = Objects.requireNonNull(id);
9         this.name = Objects.requireNonNull(name);
10        this.type = type;
11    }
12
13 }
```



```
1  @Override
2  public int describeContents() {
3      return 0;
4  }
5
6  @Override
7  public void writeToParcel(Parcel dest, int flags) {
8      dest.writeString(id);
9      dest.writeString(name);
10     dest.writeInt(type);
11 }
12
13 public static final Parcelable.Creator<Suggestion> CREATOR =
14     new Parcelable.Creator<Suggestion>() {
15     public Suggestion createFromParcel(Parcel in) {
16         return new Suggestion(in.readString(), //
17             in.readString(), //
18             in.readInt());
19     }
20
21     public Suggestion[] newArray(int size) {
22         return new Suggestion[size];
23     }
24 };
```

Parcelable.Creator

The base creator interface

Parcelable.ClassLoaderCreator

A creator with the ClassLoader passed on read.

ParcelableCompat &

ParcelableCompatCreatorCallbacks

Compatibility stuff

Bundle

A key-value map & type-safe Parcelable

Parcel

internally uses reflection

(required to get the CREATOR instance)

...i.e. beware Proguard

Activity level
state restoration

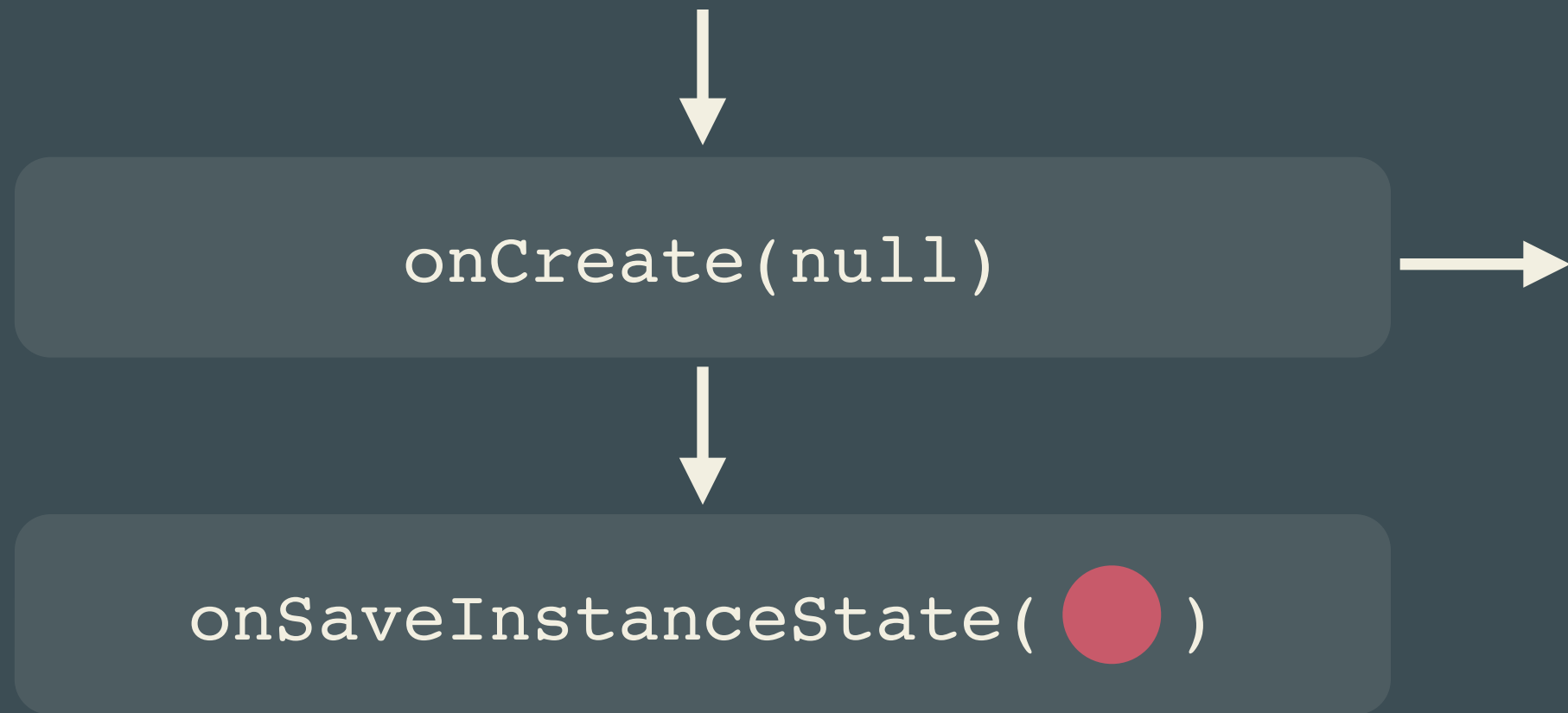


```
onCreate(null)
```

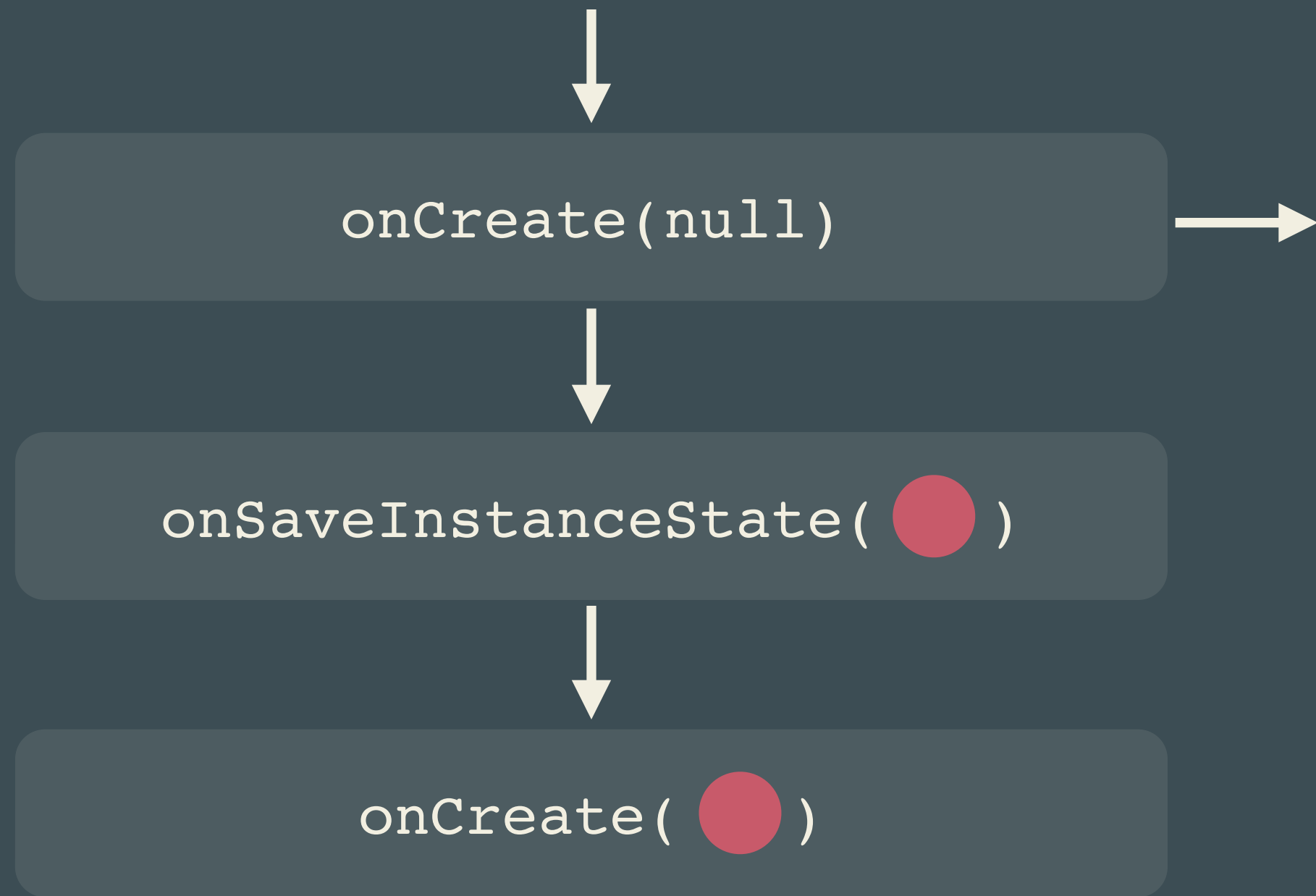


```
graph TD; A[ ] --> B[onCreate(null)]; C[ ] --> B;
```

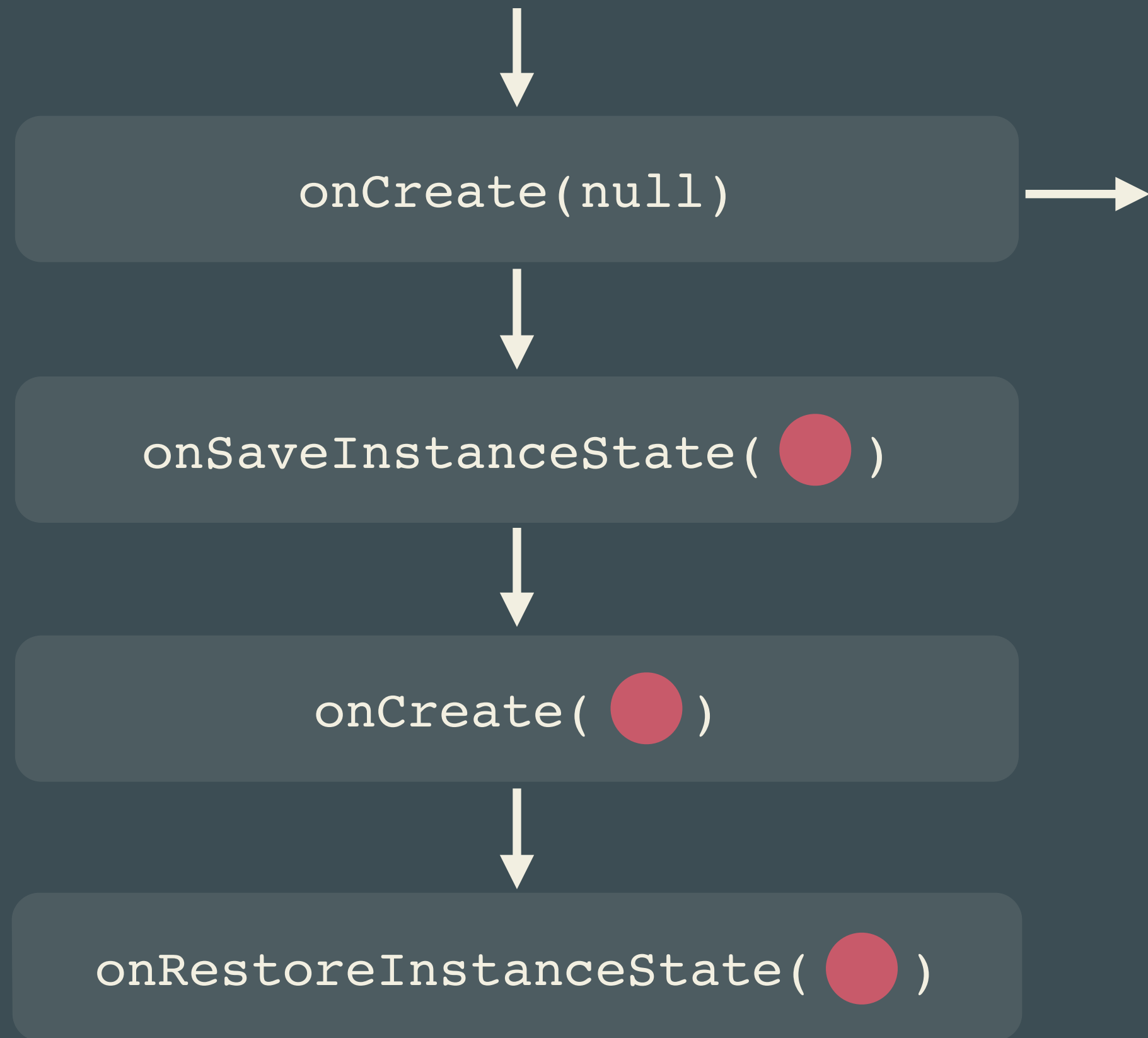
`onCreate(null)`



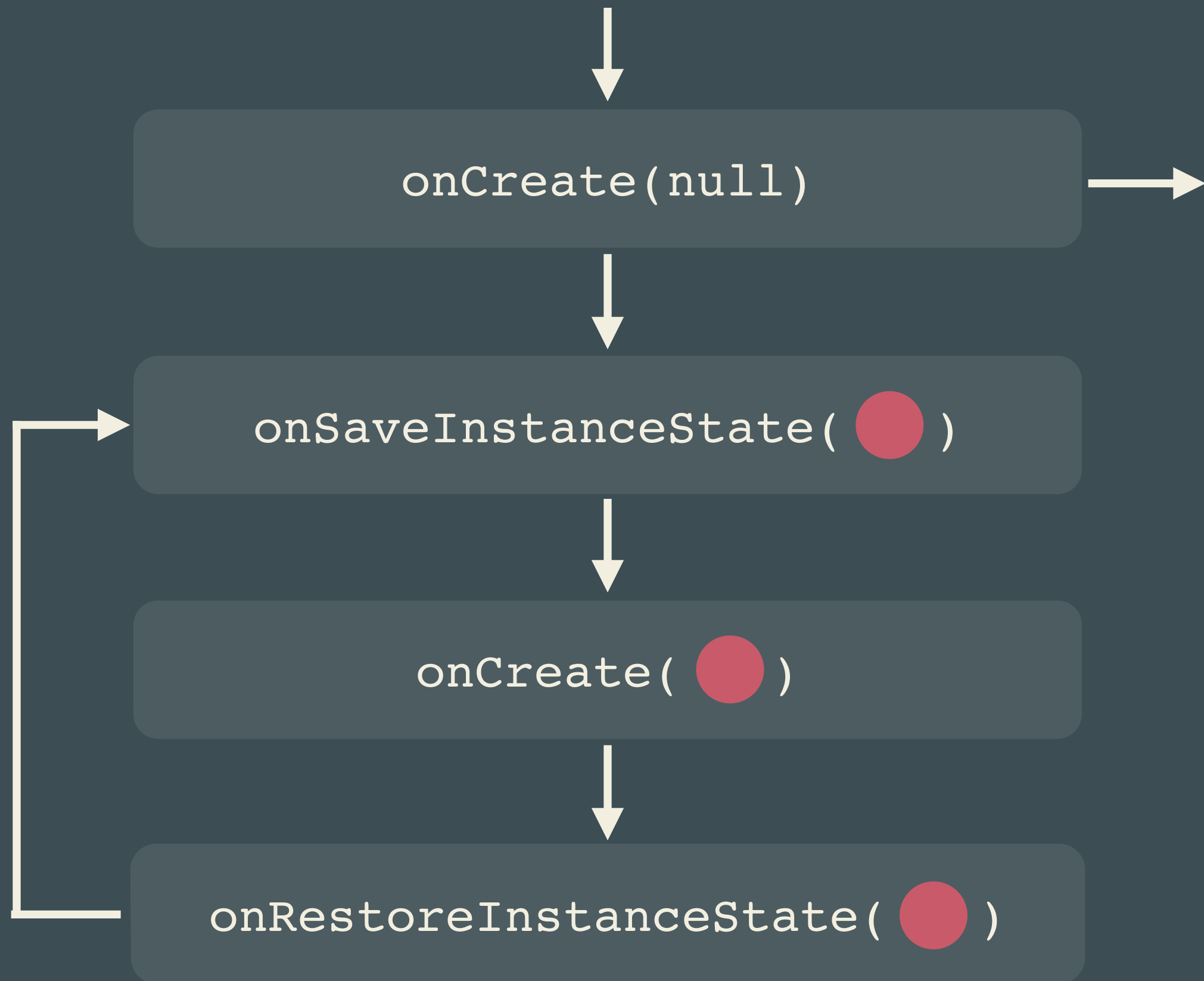
: *non-null Bundle*



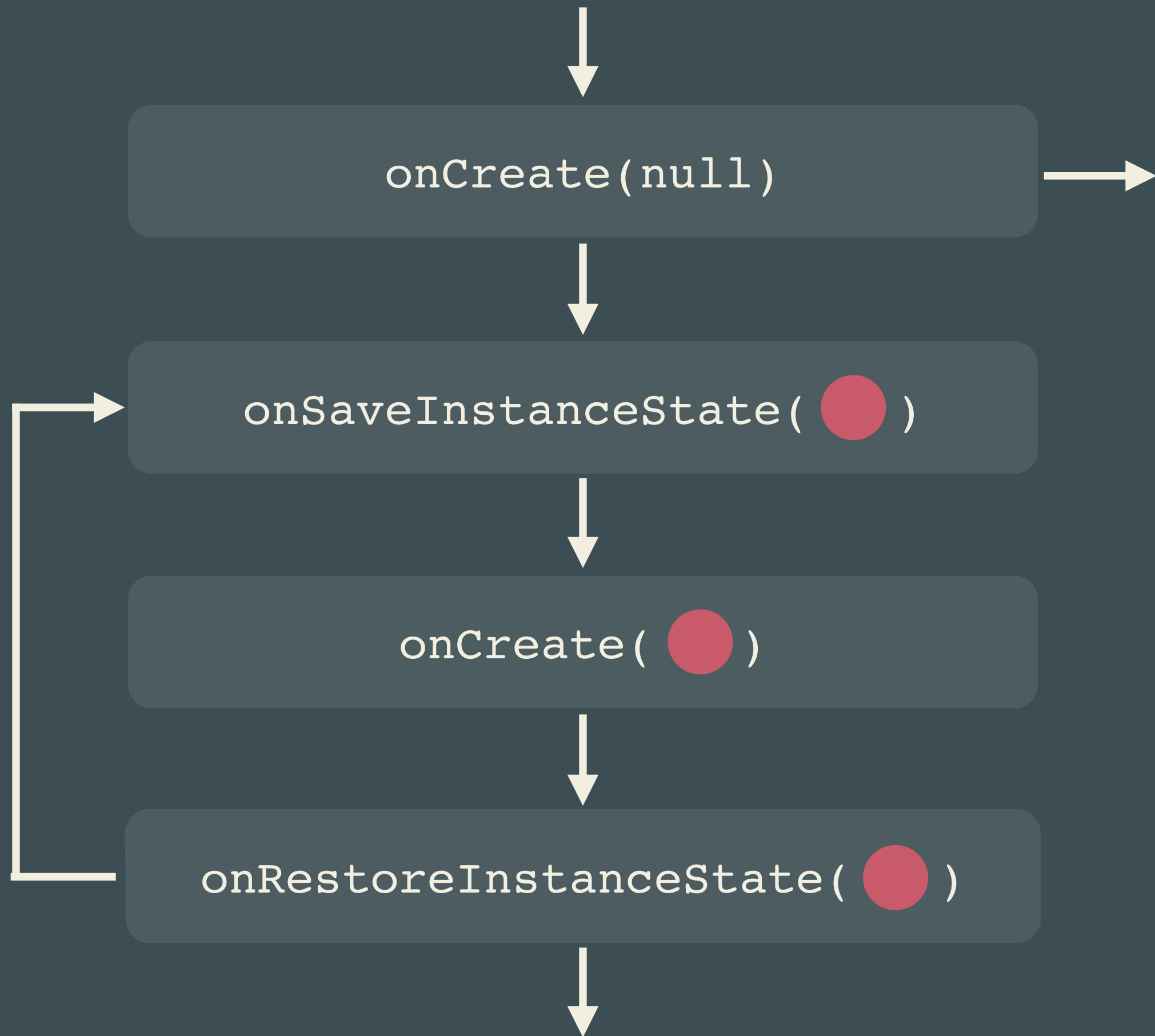
●: *non-null Bundle*




●: *non-null Bundle*



●: *non-null Bundle*



: *non-null Bundle*

What to save?

Non persistent or non reconstructible info

```
1 public class SearchActivity extends Activity {
2
3     private static final String STATE_OUTWARD = "state:outward";
4
5     private DateComponents mOutward;
6
7     @Override
8     protected void onCreate(Bundle savedInstanceState) {
9         super.onCreate(savedInstanceState);
10
11         if (savedInstanceState != null) {
12             DateComponents components = savedInstanceState.getParcelable(STATE_OUTWARD);
13             if (components != null) {
14                 setOutward(components);
15             }
16         }
17     }
18
19     @Override
20     protected void onSaveInstanceState(Bundle savedInstanceState) {
21         super.onSaveInstanceState(savedInstanceState);
22         savedInstanceState.putParcelable(STATE_OUTWARD, mOutward);
23     }
24 }
```


onSaveInstanceState saves

Window

onSaveInstanceState saves

Window | Fragments

onSaveInstanceState saves

Window | Fragments | Dialogs

Always call the

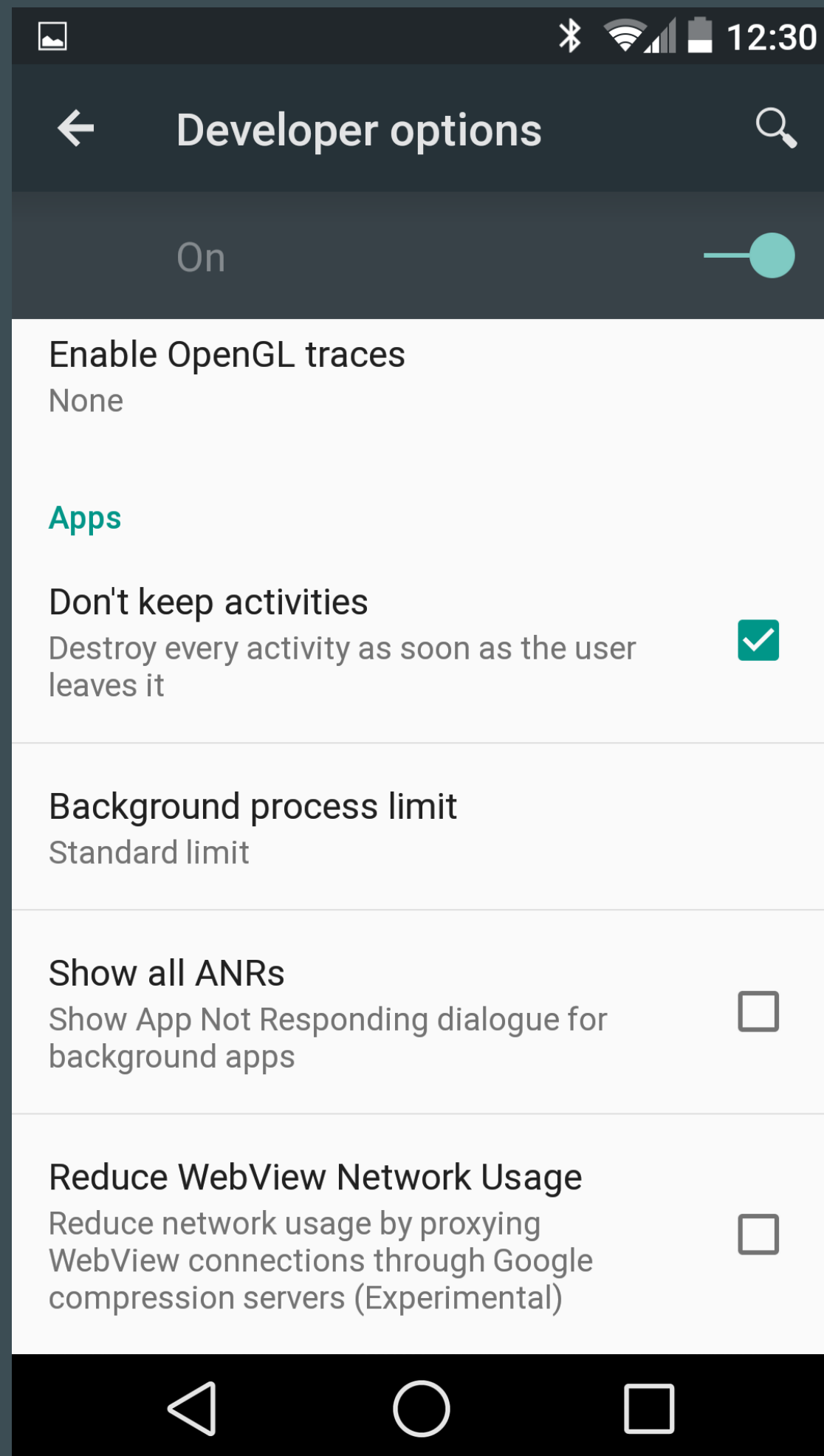
SUPER METHODS

Android has no guards on save-related methods

android:stateNotNeeded

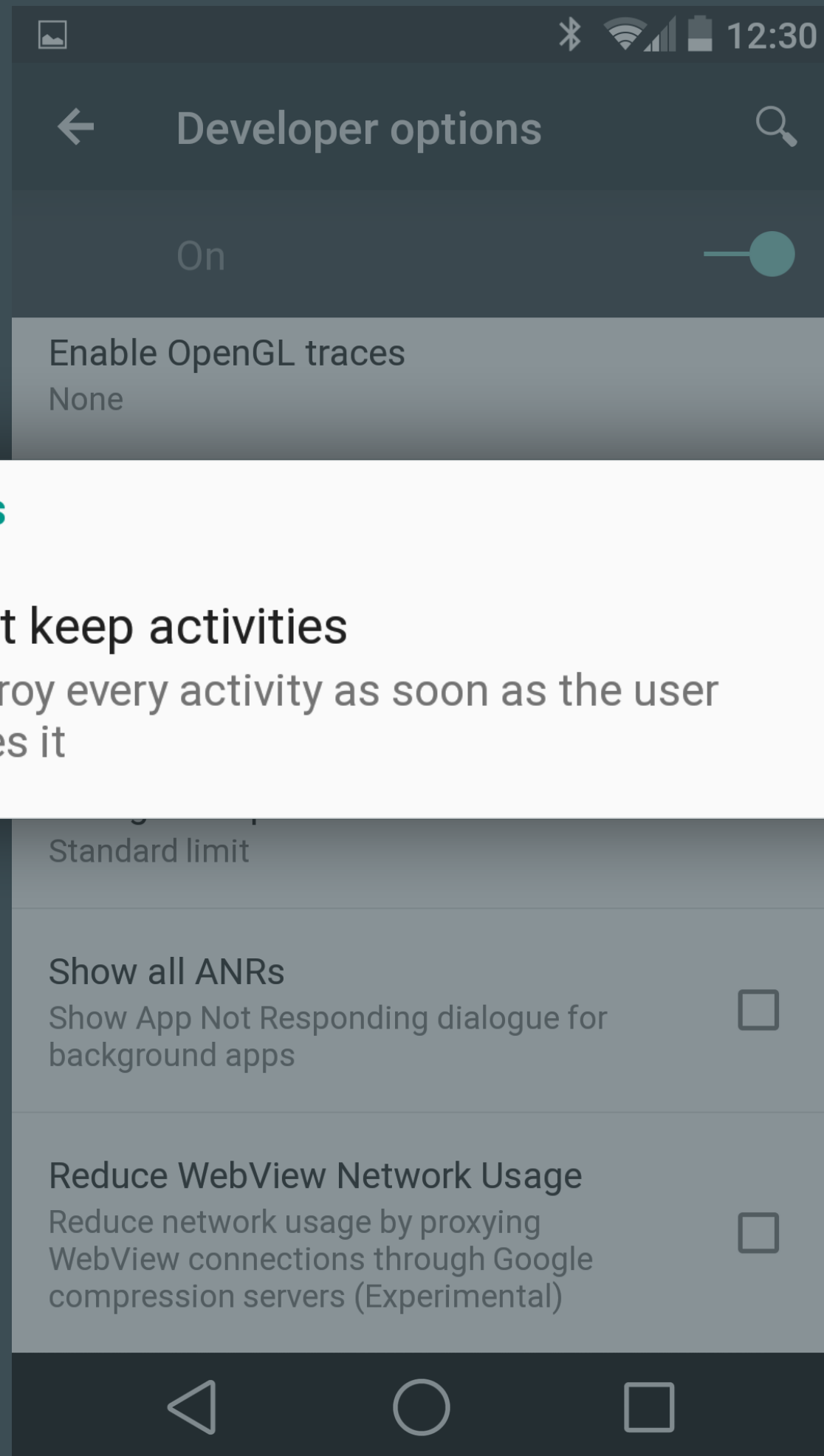
For restart-on-crash apps only

(i.e. launcher app)



Developer options

Don't keep activities



Apps

Don't keep activities

Destroy every activity as soon as the user leaves it



Developer options

Don't keep activities

View level

state restoration

Android saves UI state

AUTOMAGICALLY

Android saves UI state

AUTOMAGICALLY

(aka “It just works!™”)

Android saves UI state

AUTOMAGICALLY

(aka “It just works!TM”)

...except in some cases

Works out-of-the-box if Views

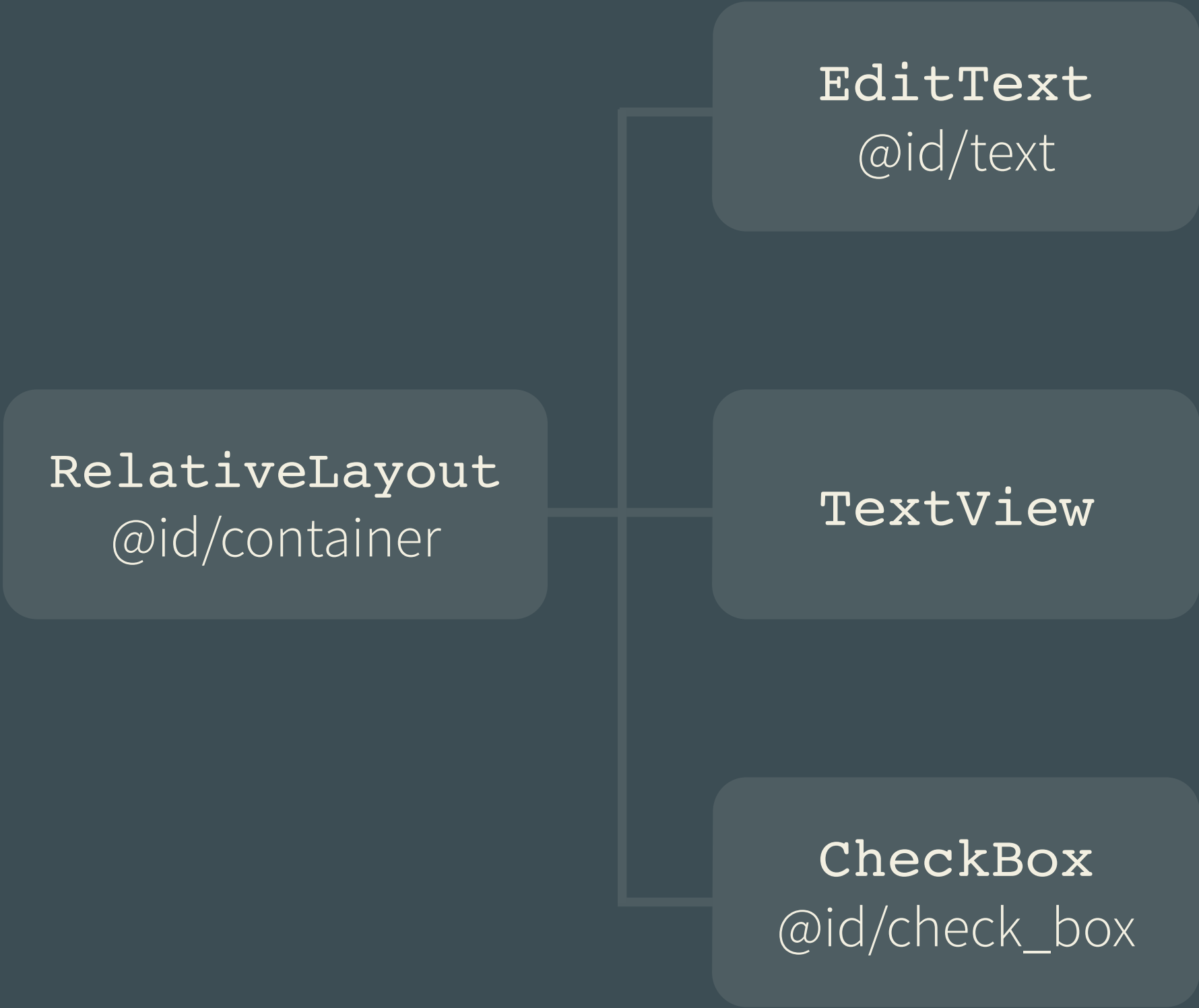
1. Have an ID

2. Are “save” enabled

3. Come from the framework

It always begins with a call to

```
saveHierarchyState()
```



RelativeLayout
@id/container

EditText
@id/text

TextView

CheckBox
@id/check_box

`SparseArray<Parcelable>`

RelativeLayout
@id/container

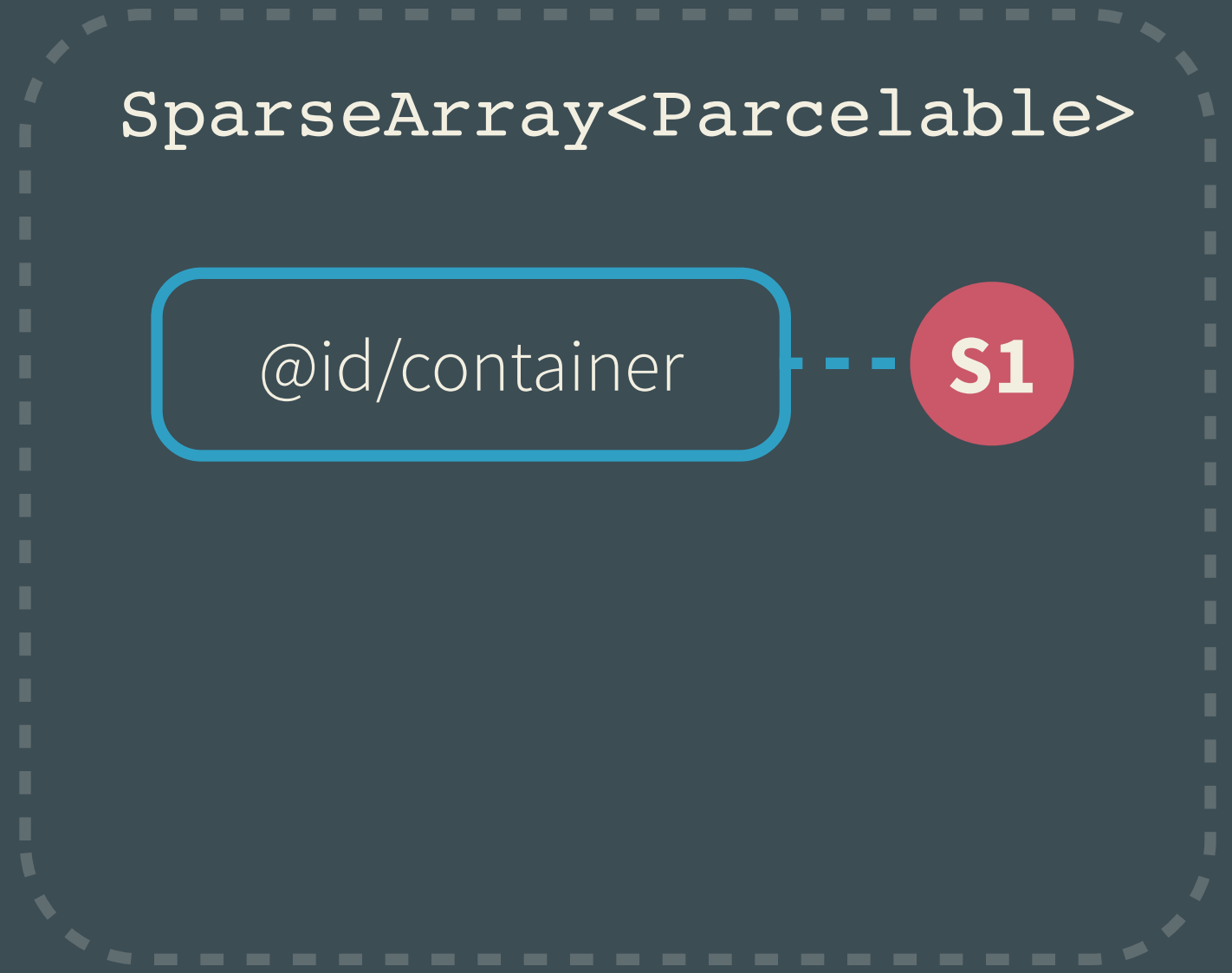
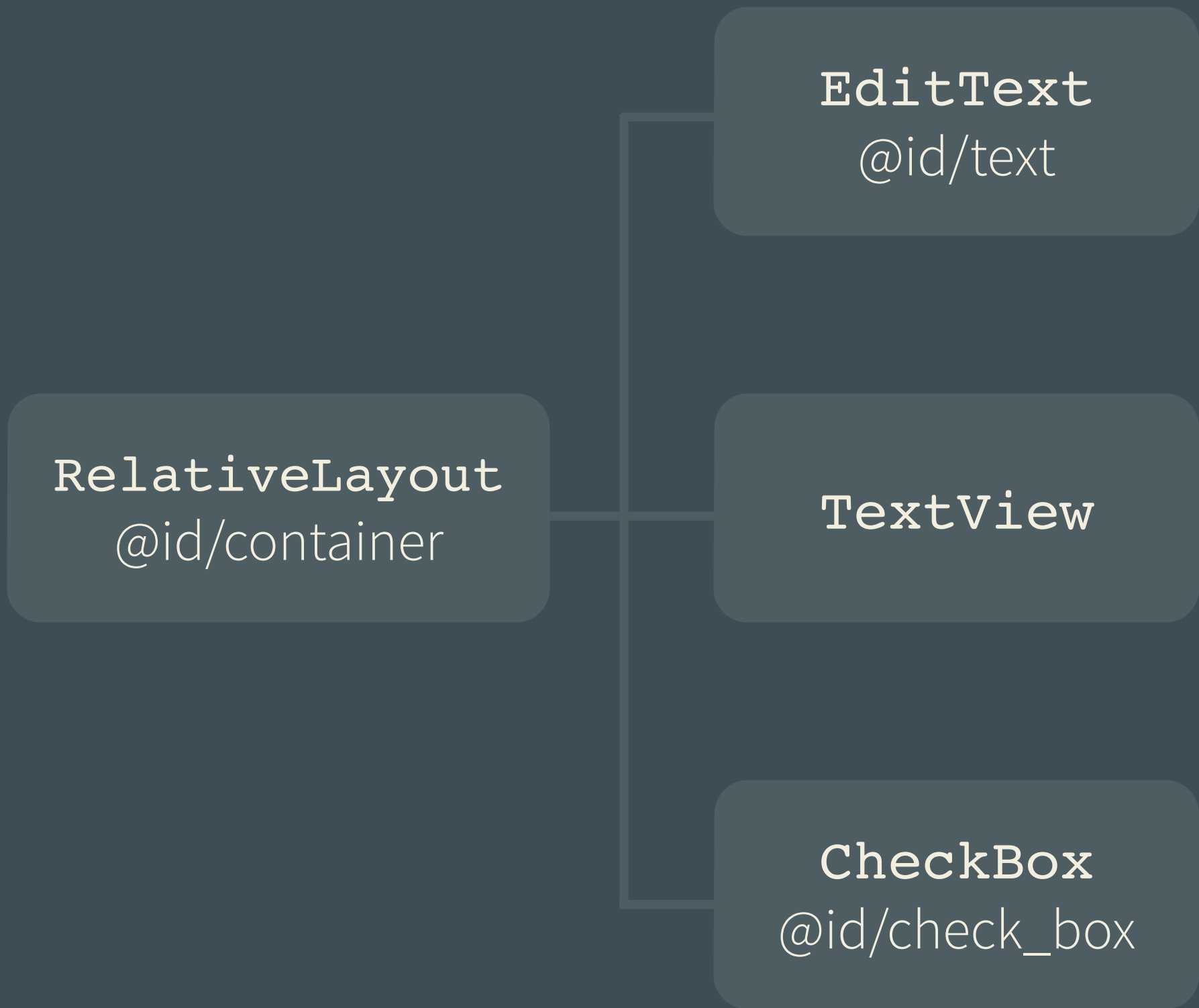
S1

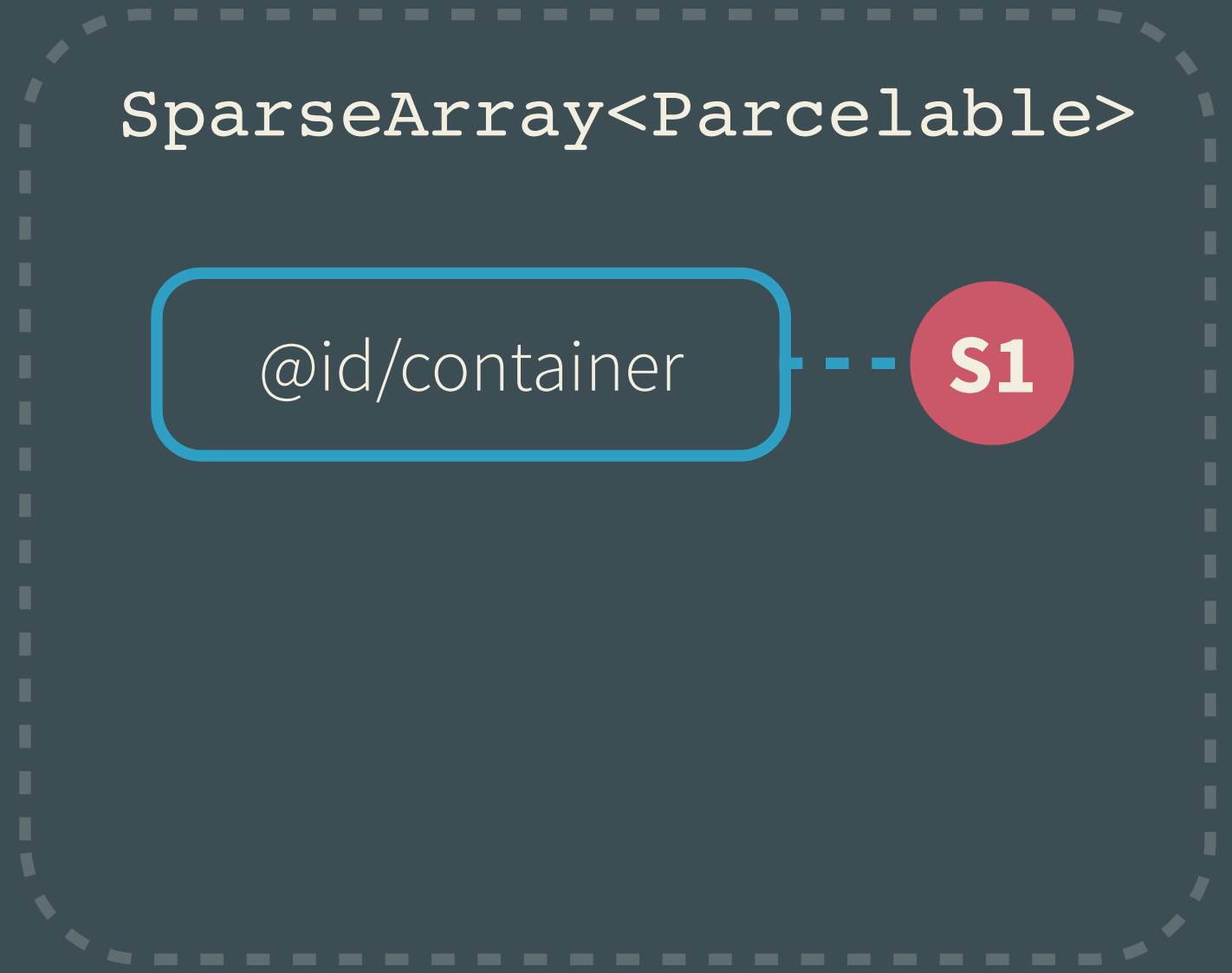
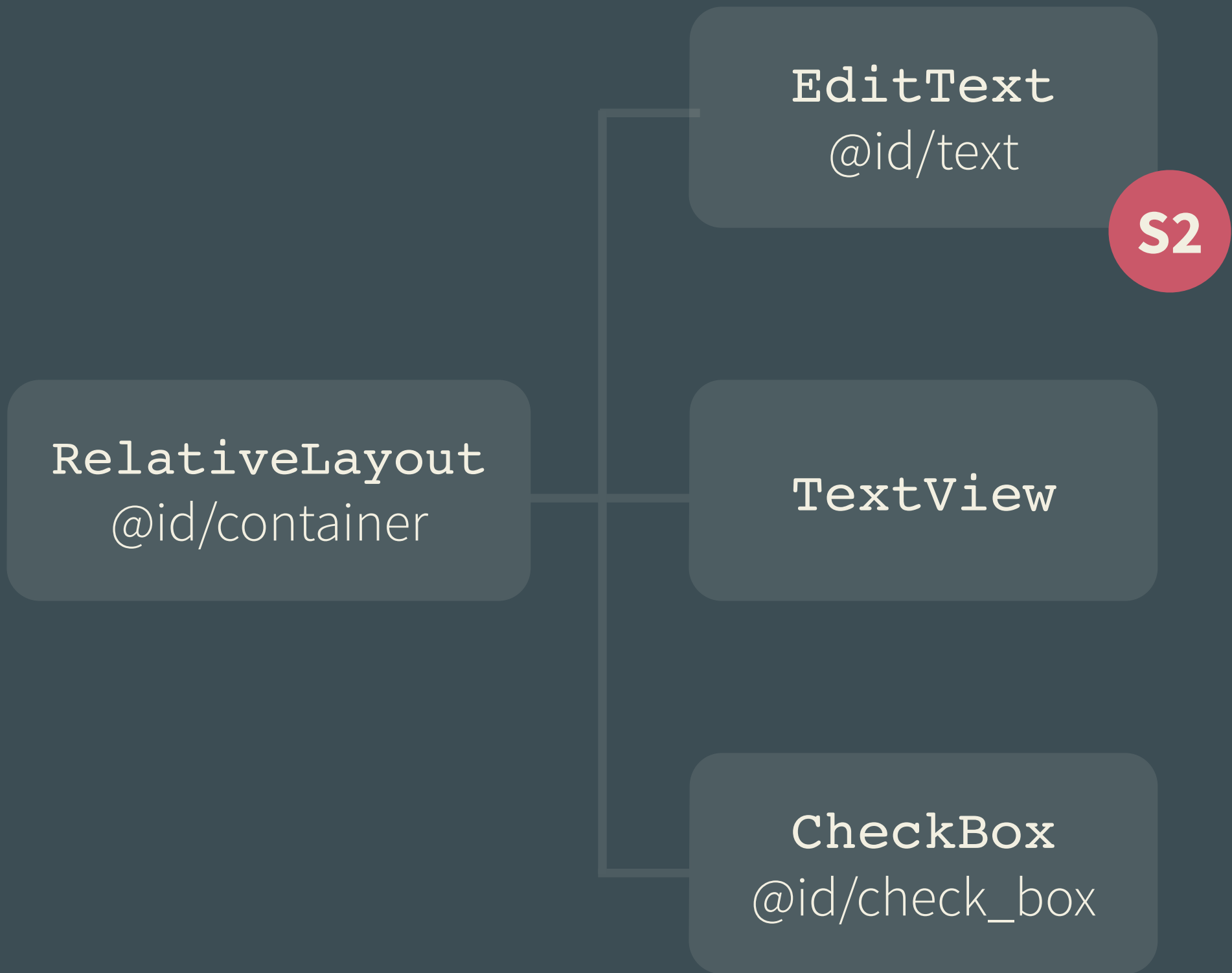
EditText
@id/text

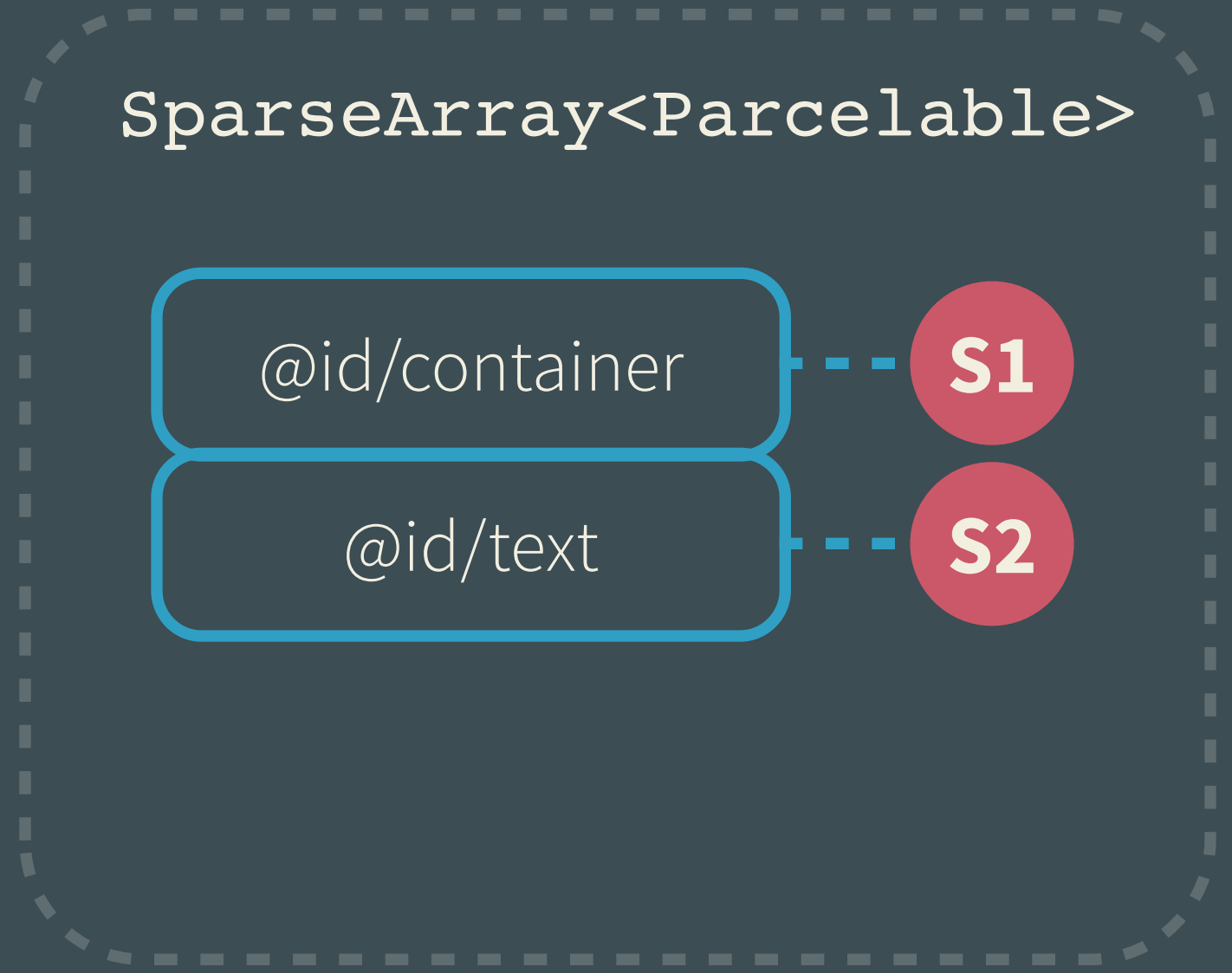
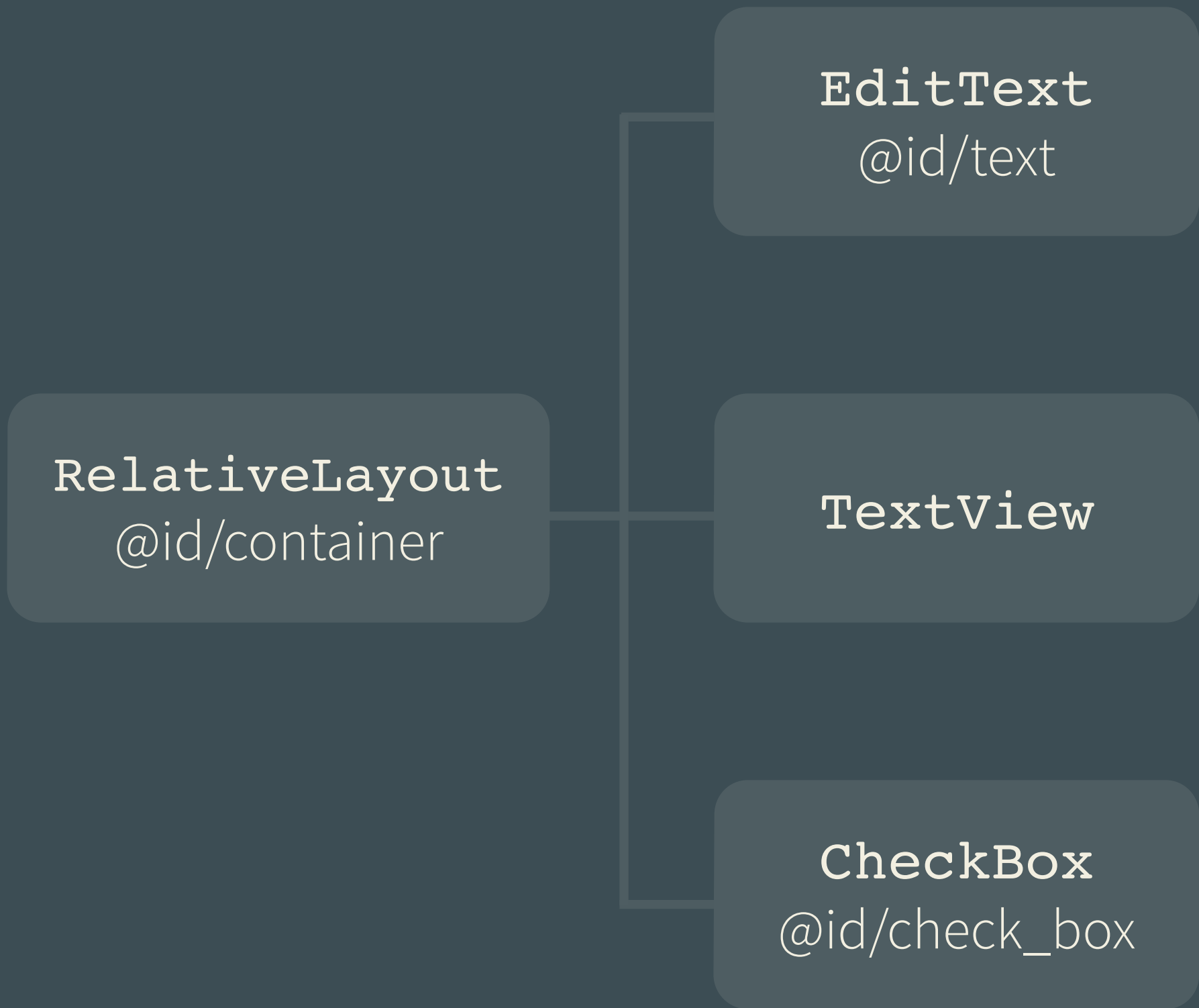
TextView

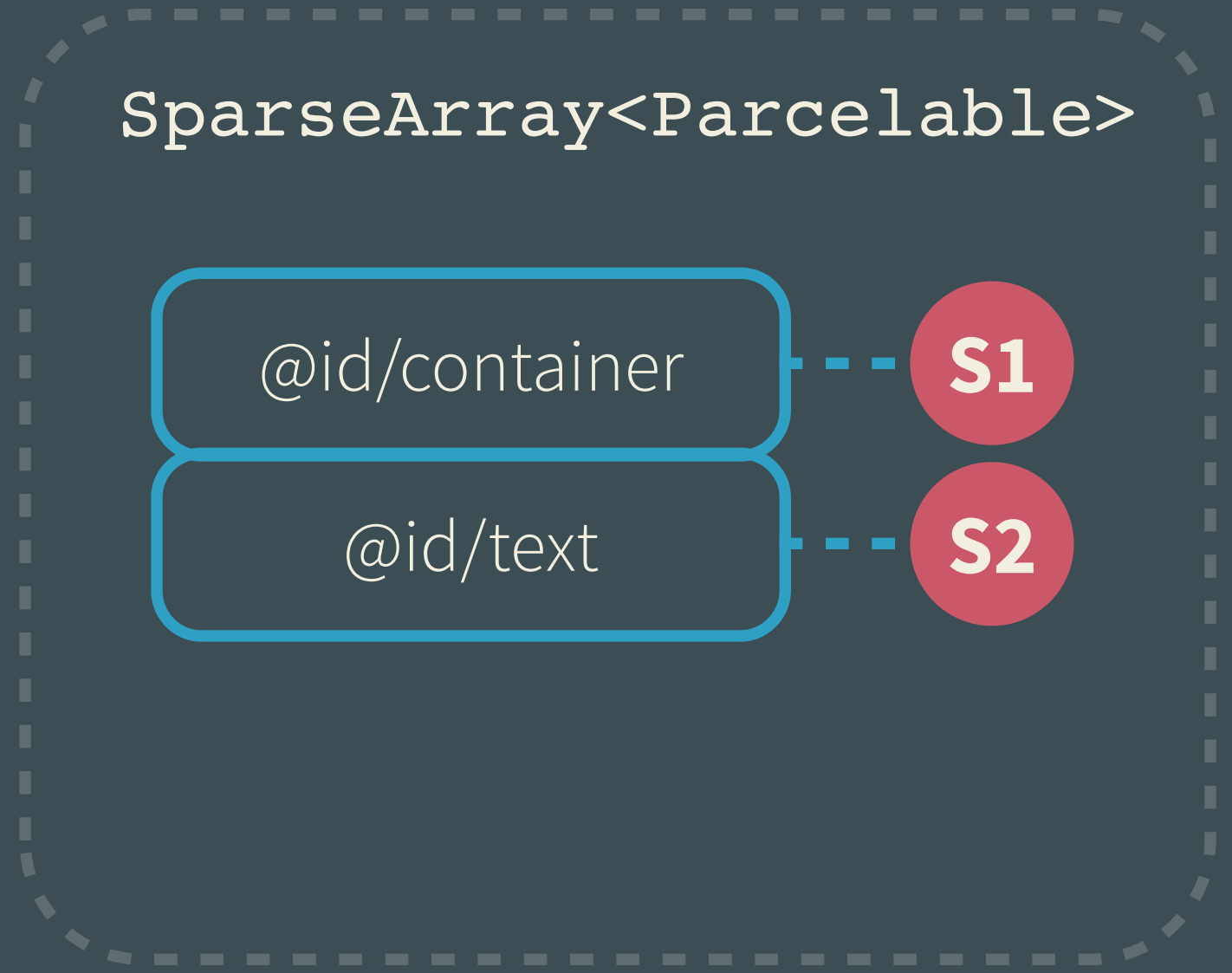
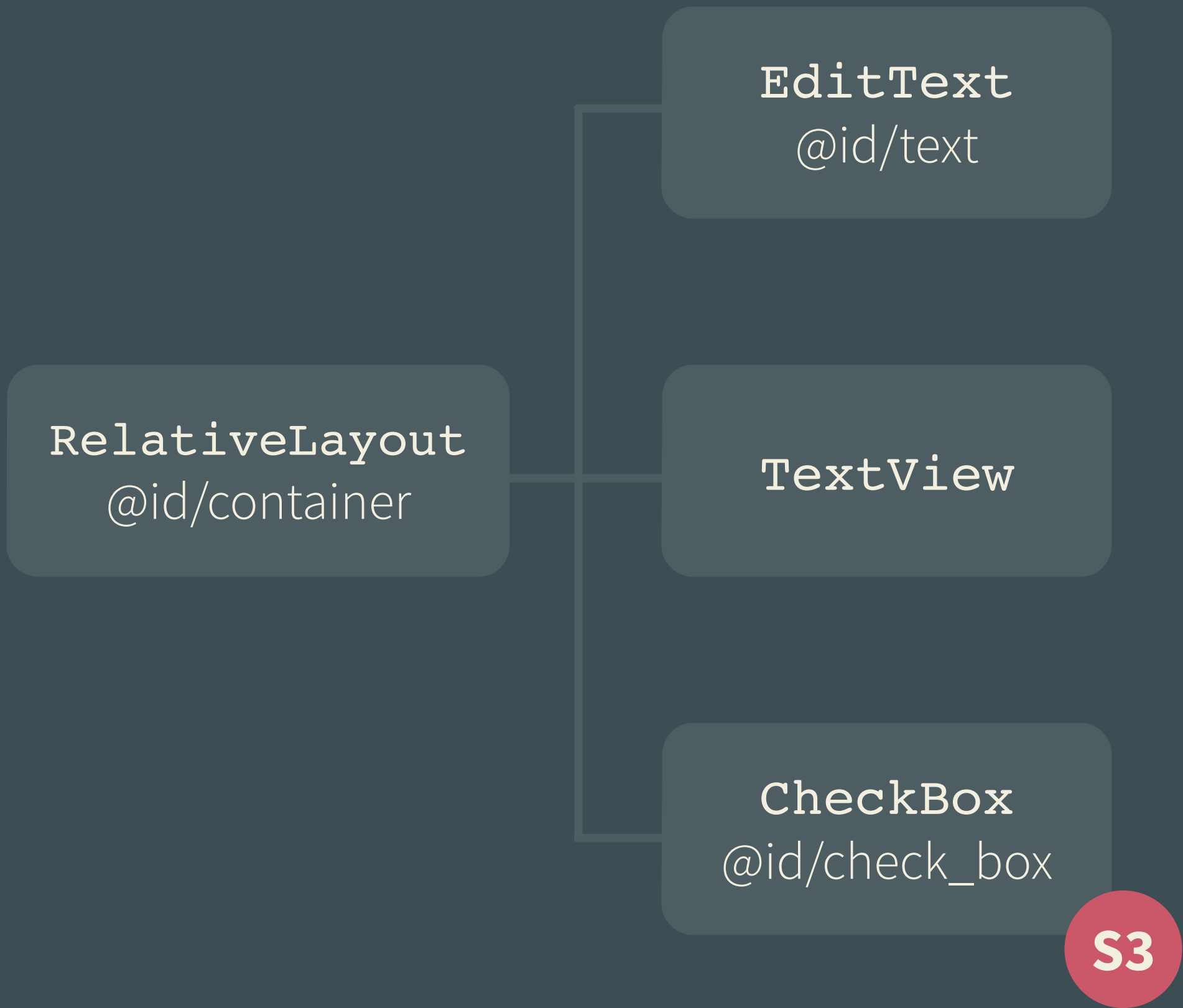
CheckBox
@id/check_box

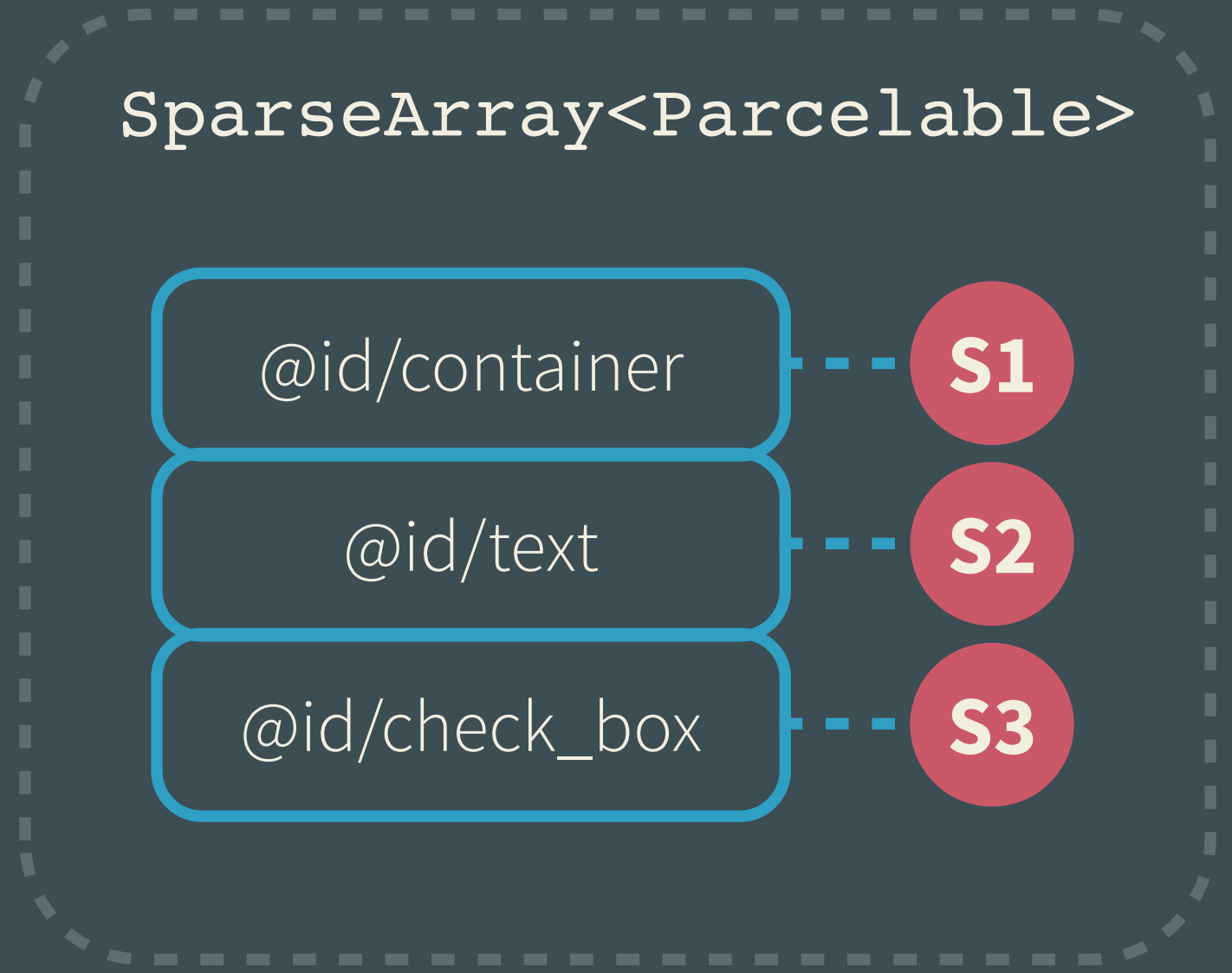
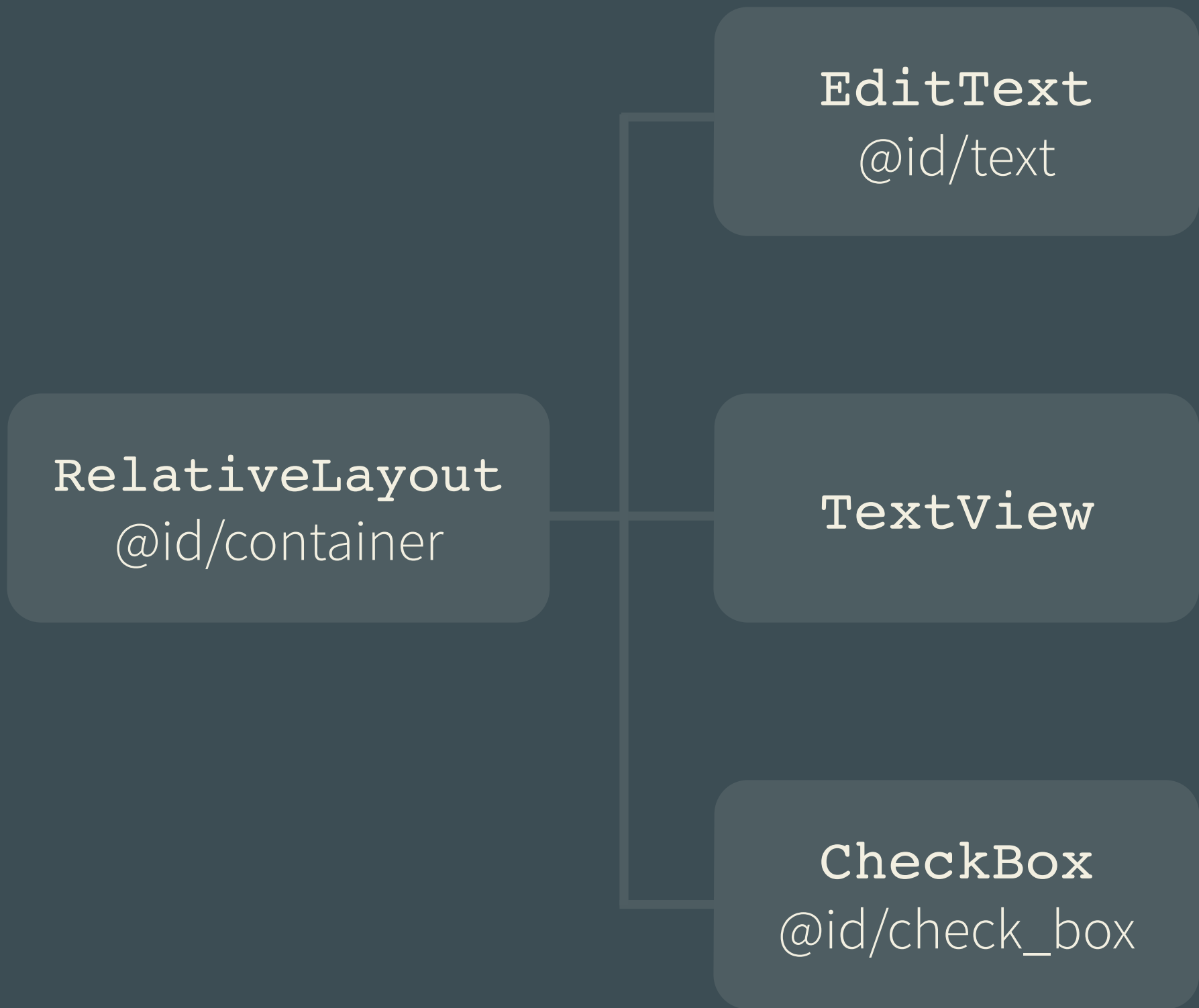
`SparseArray<Parcelable>`











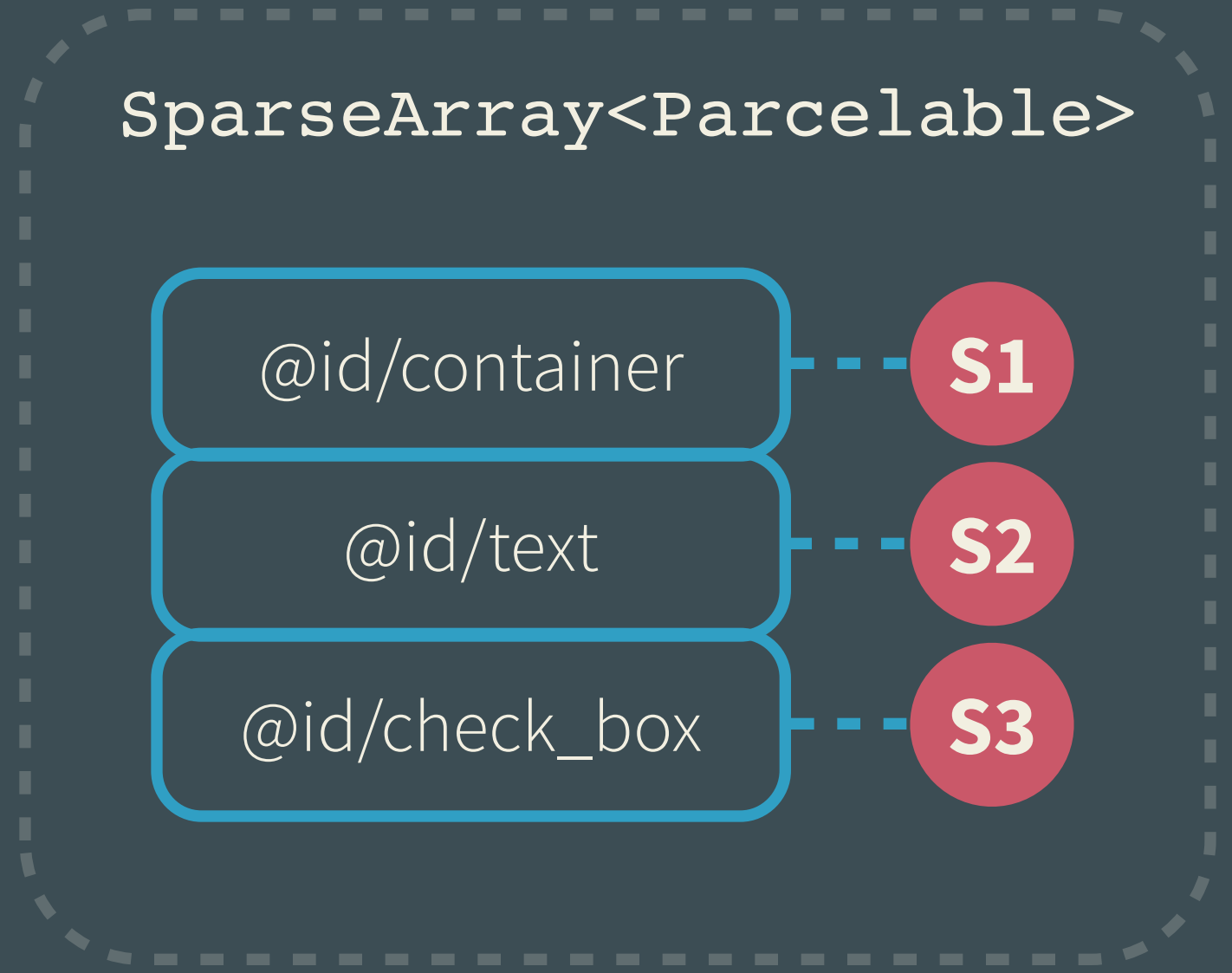
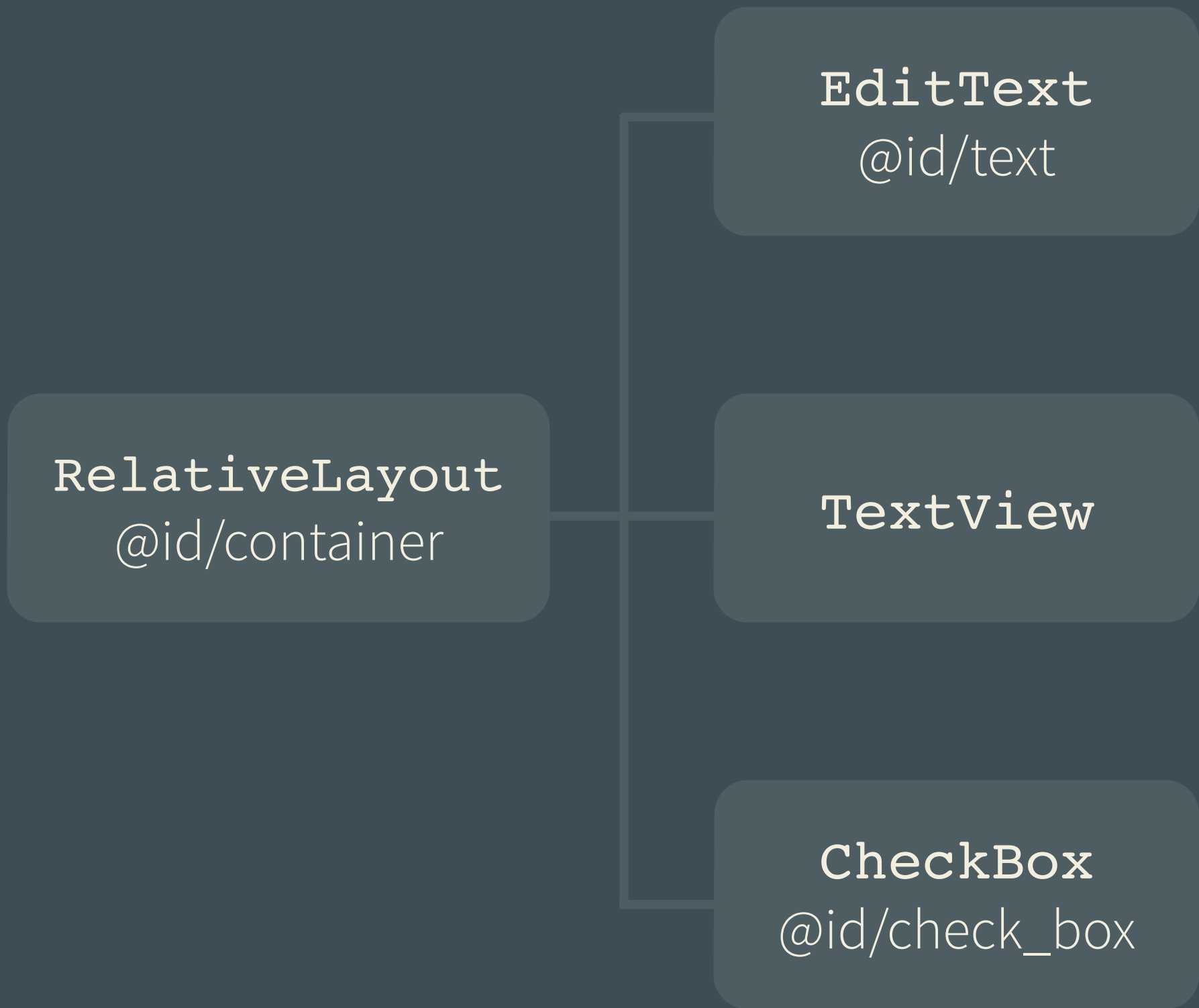
Controlling save

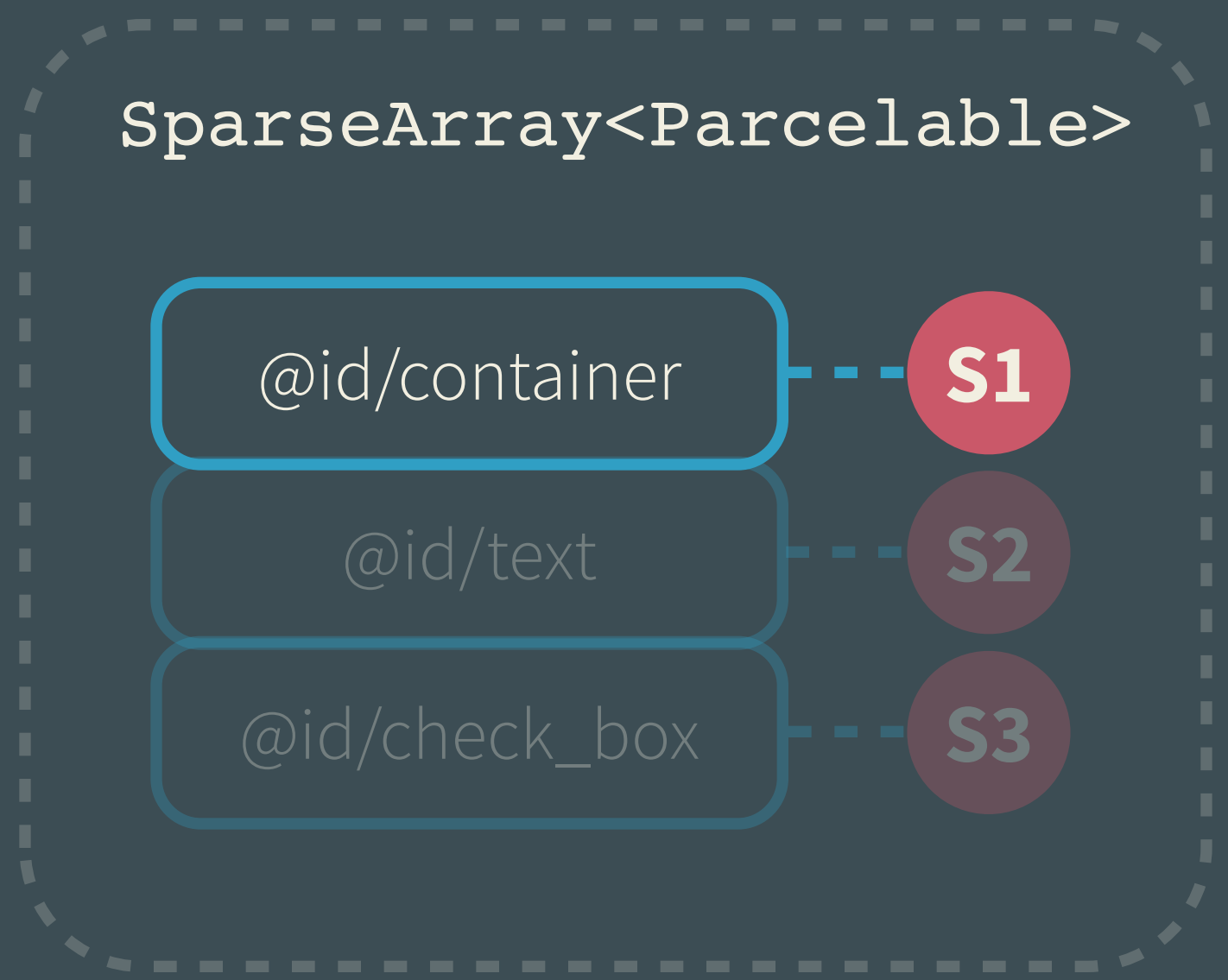
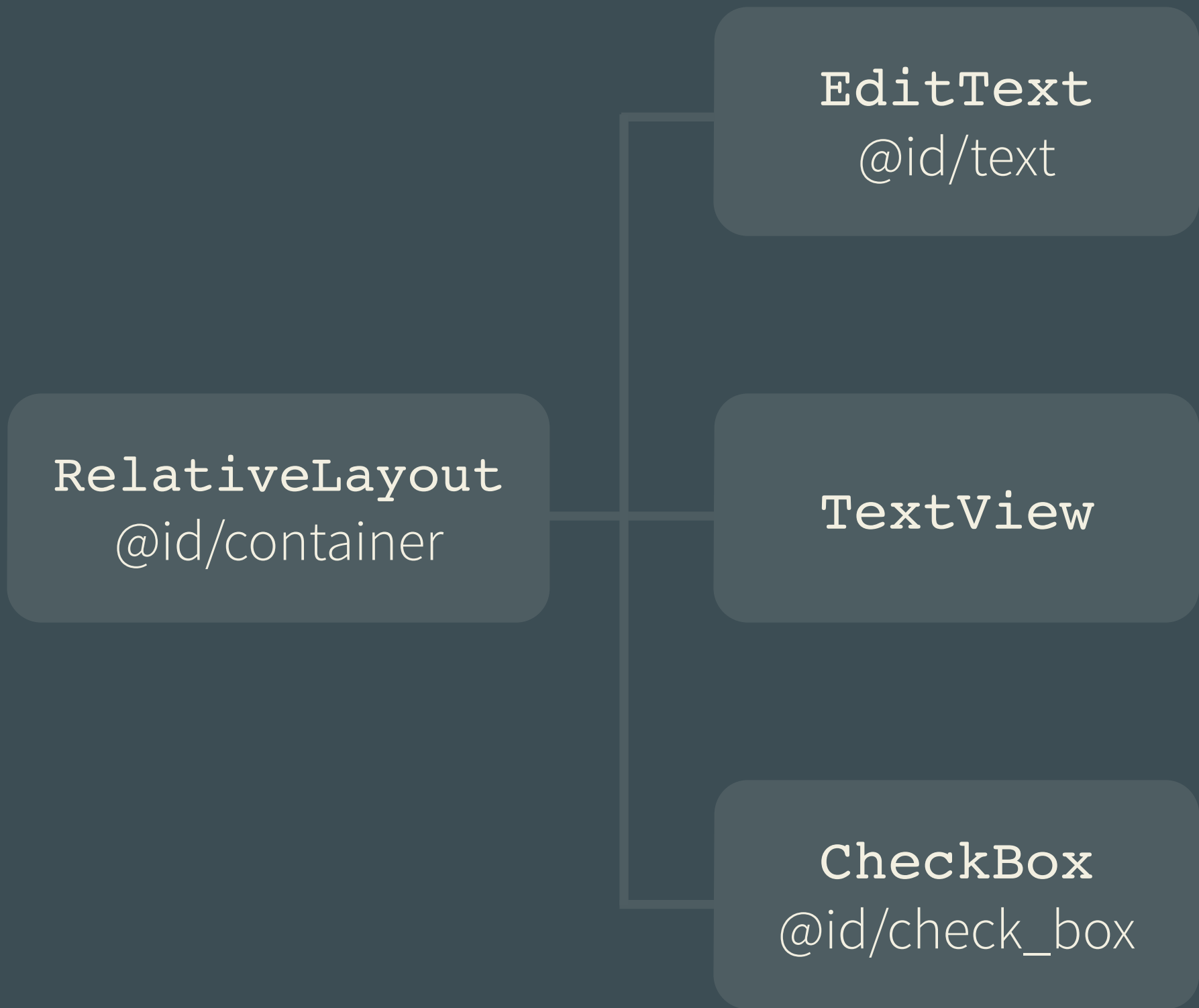
```
setSaveEnabled(boolean)
```

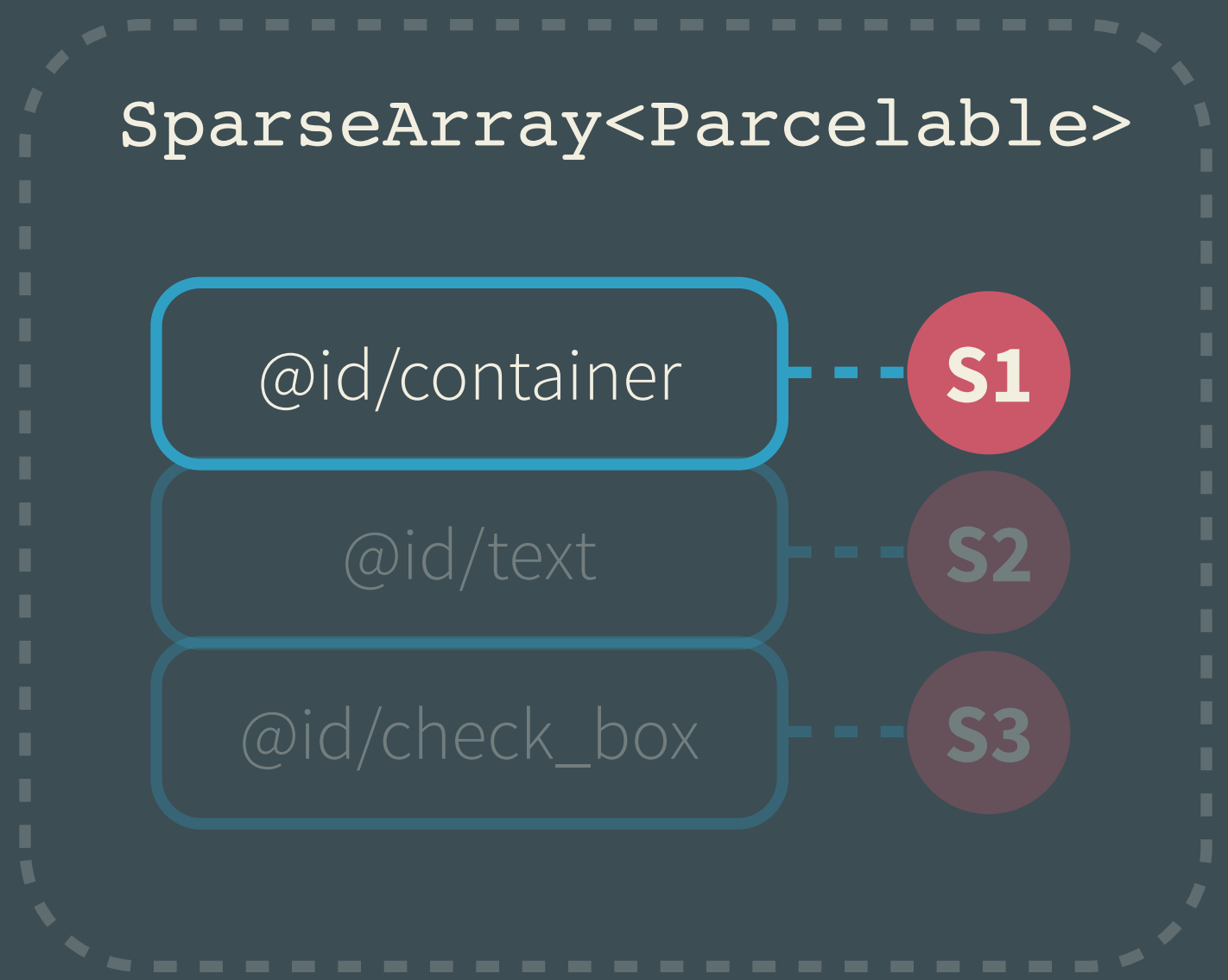
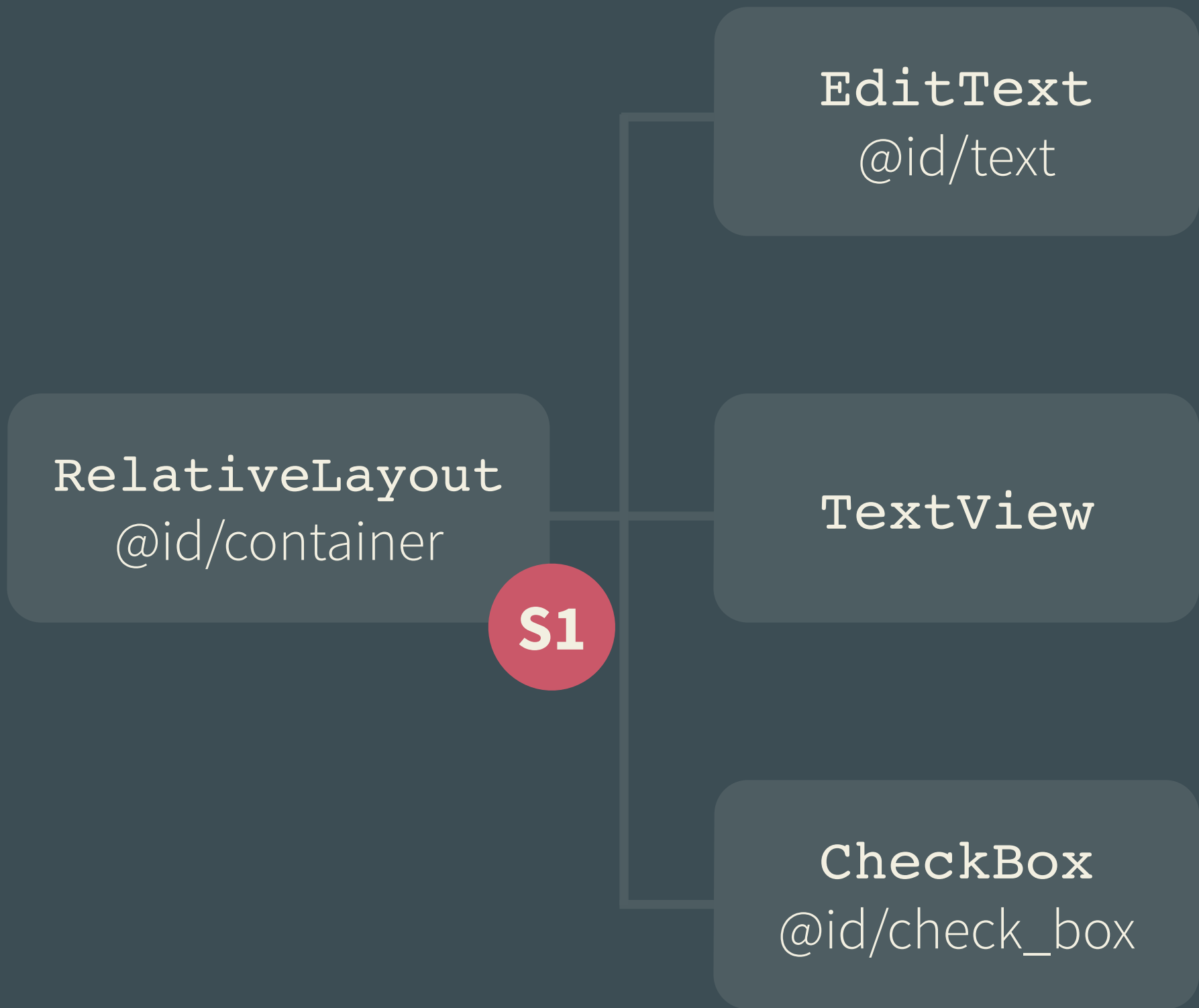
```
setSaveFromParentEnabled(boolean)
```

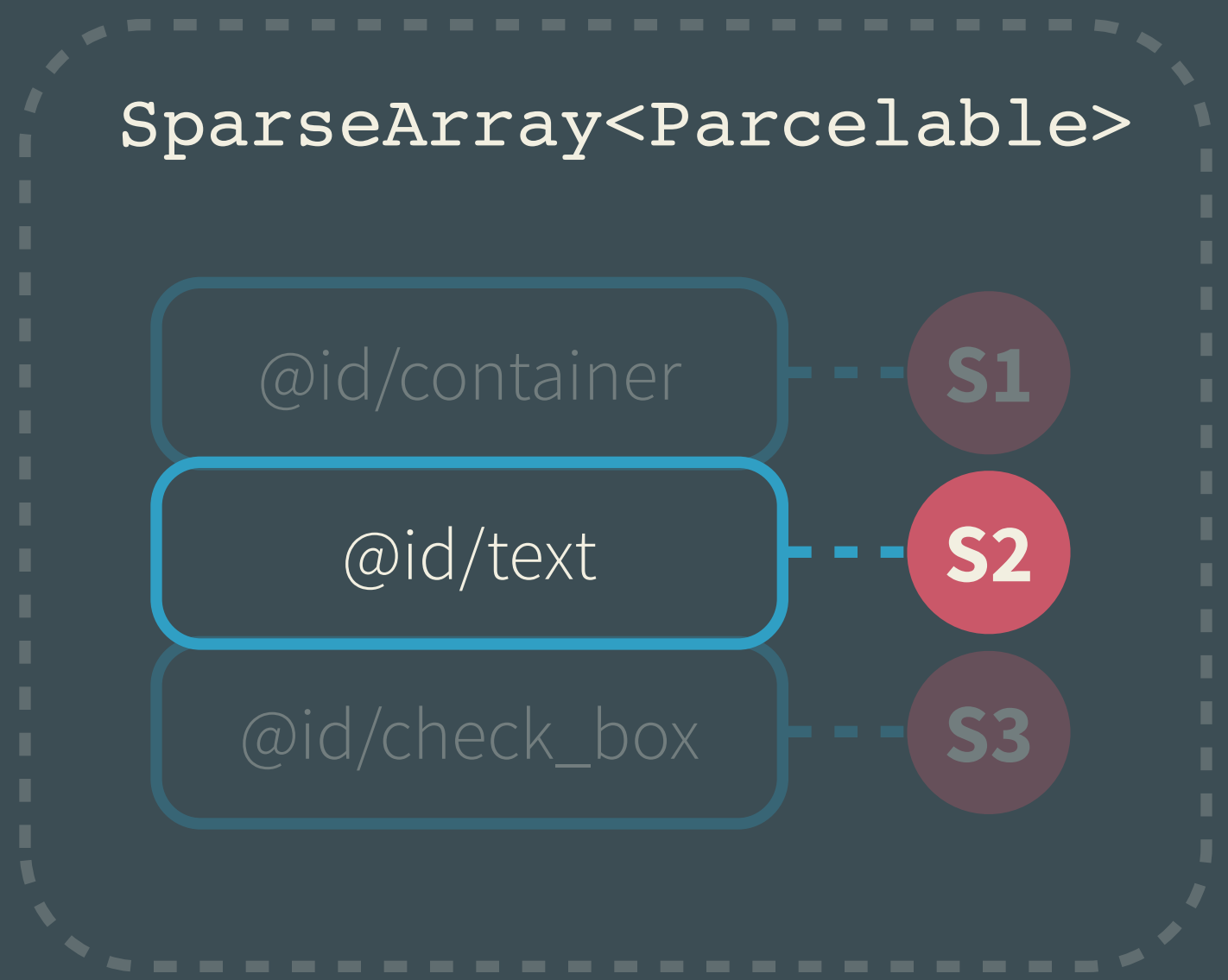
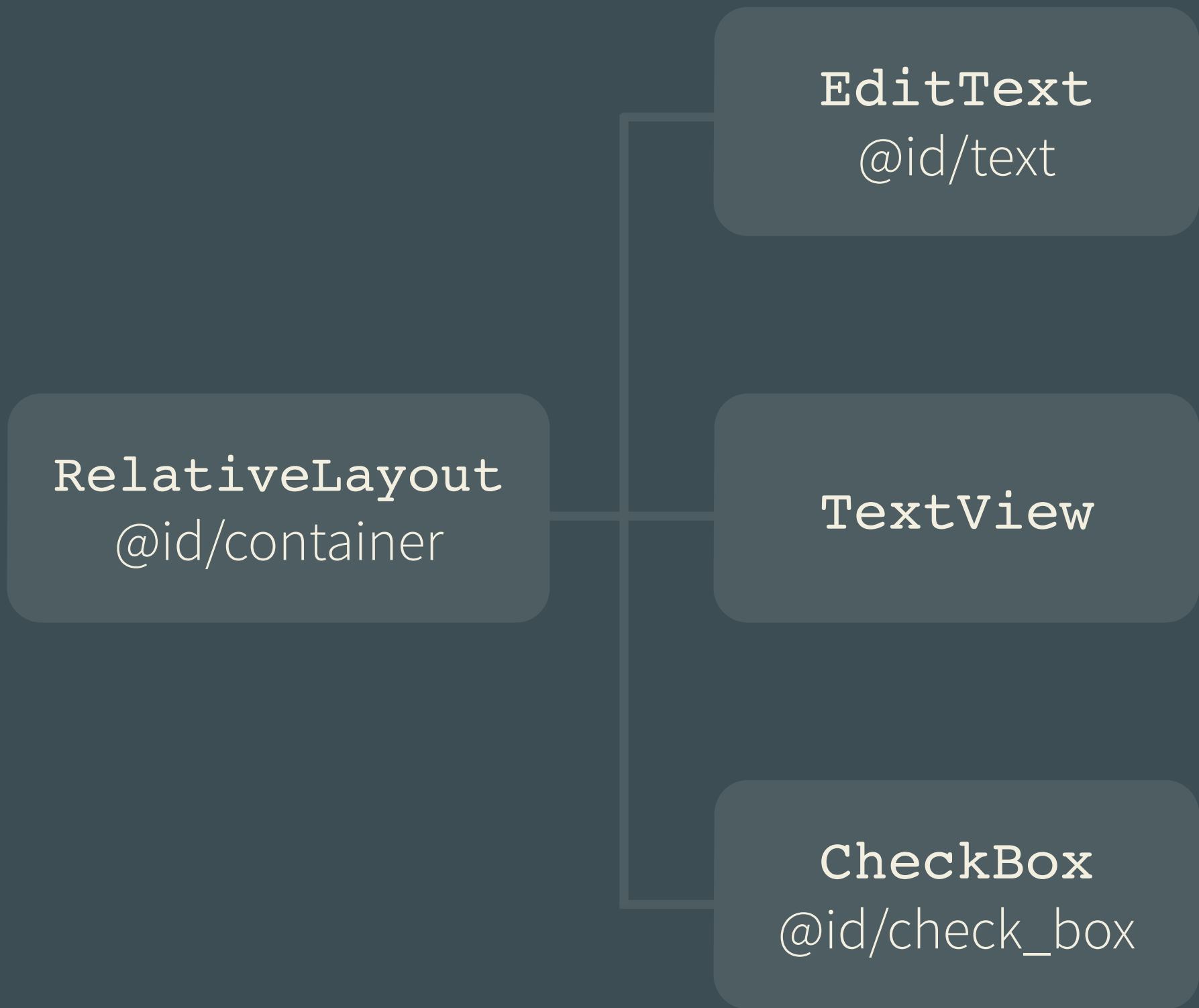
It always ends with a call to

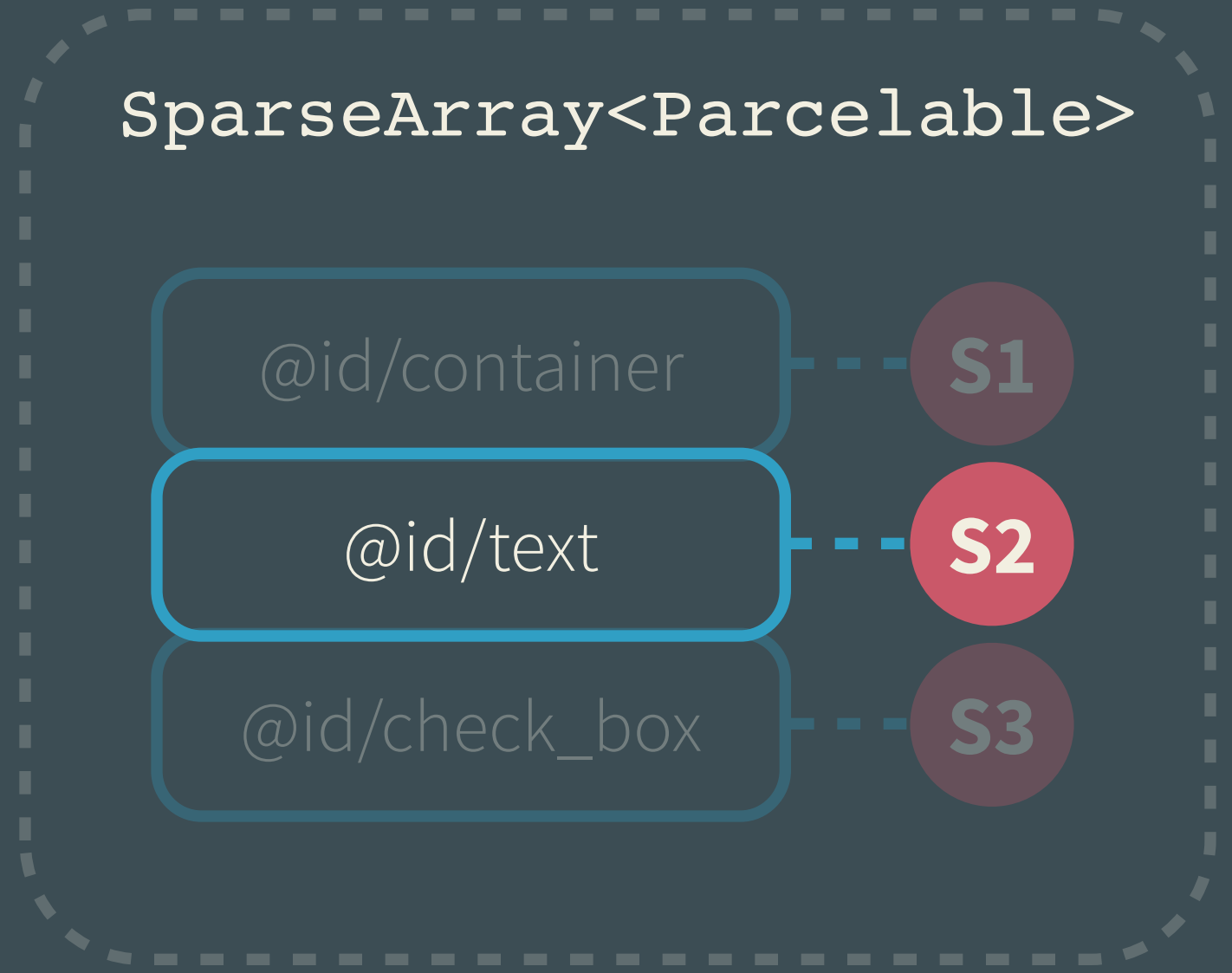
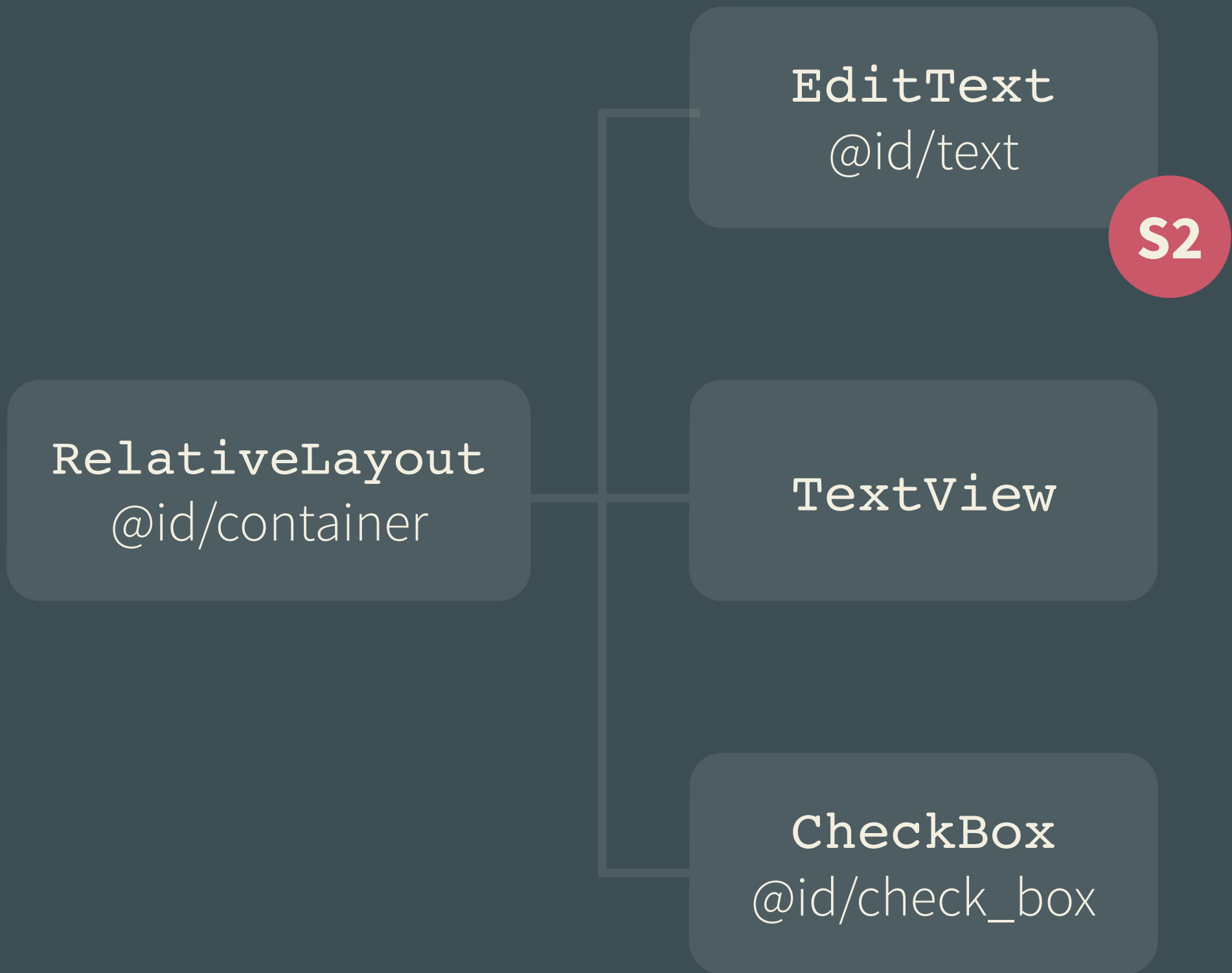
```
restoreHierarchyState()
```

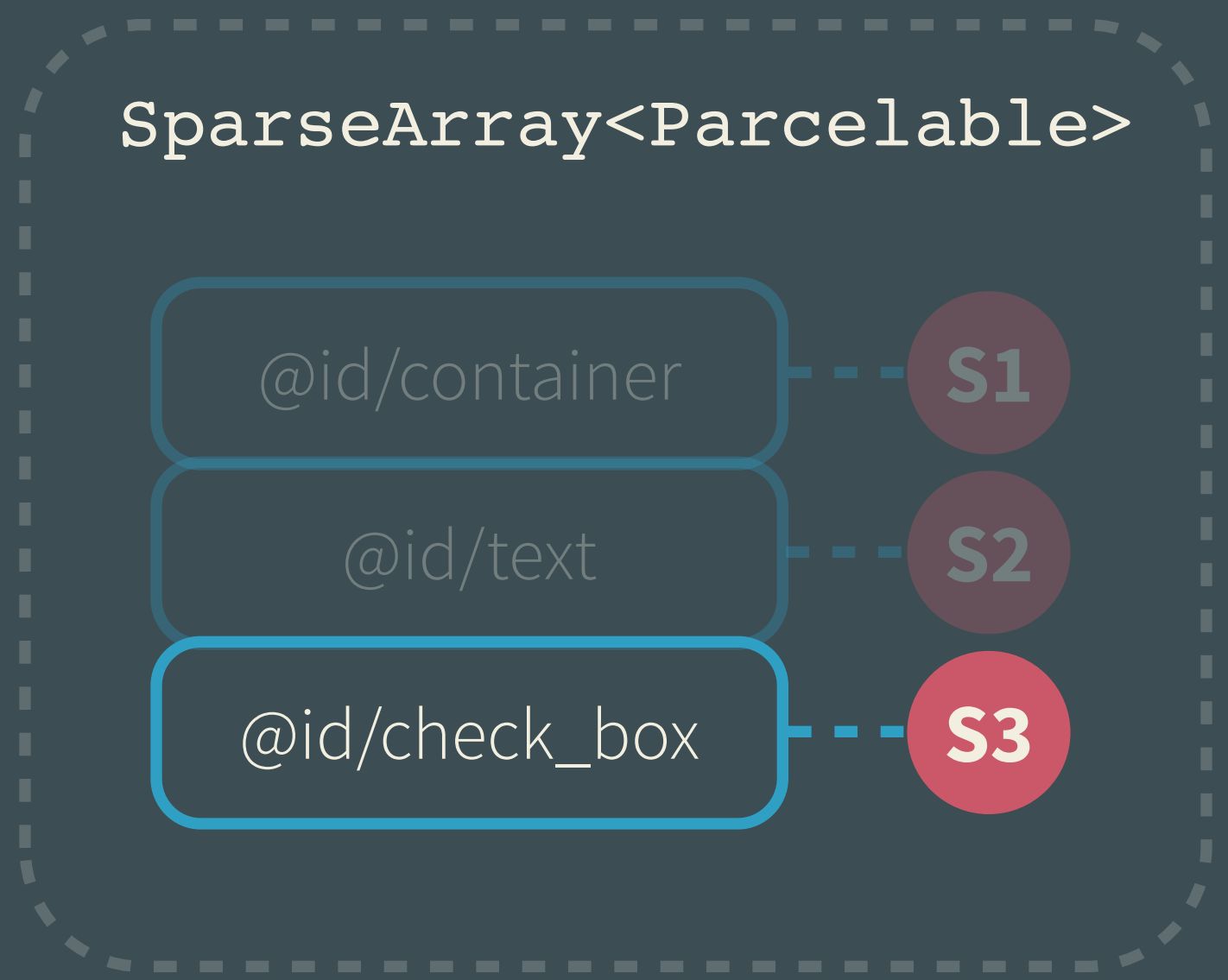
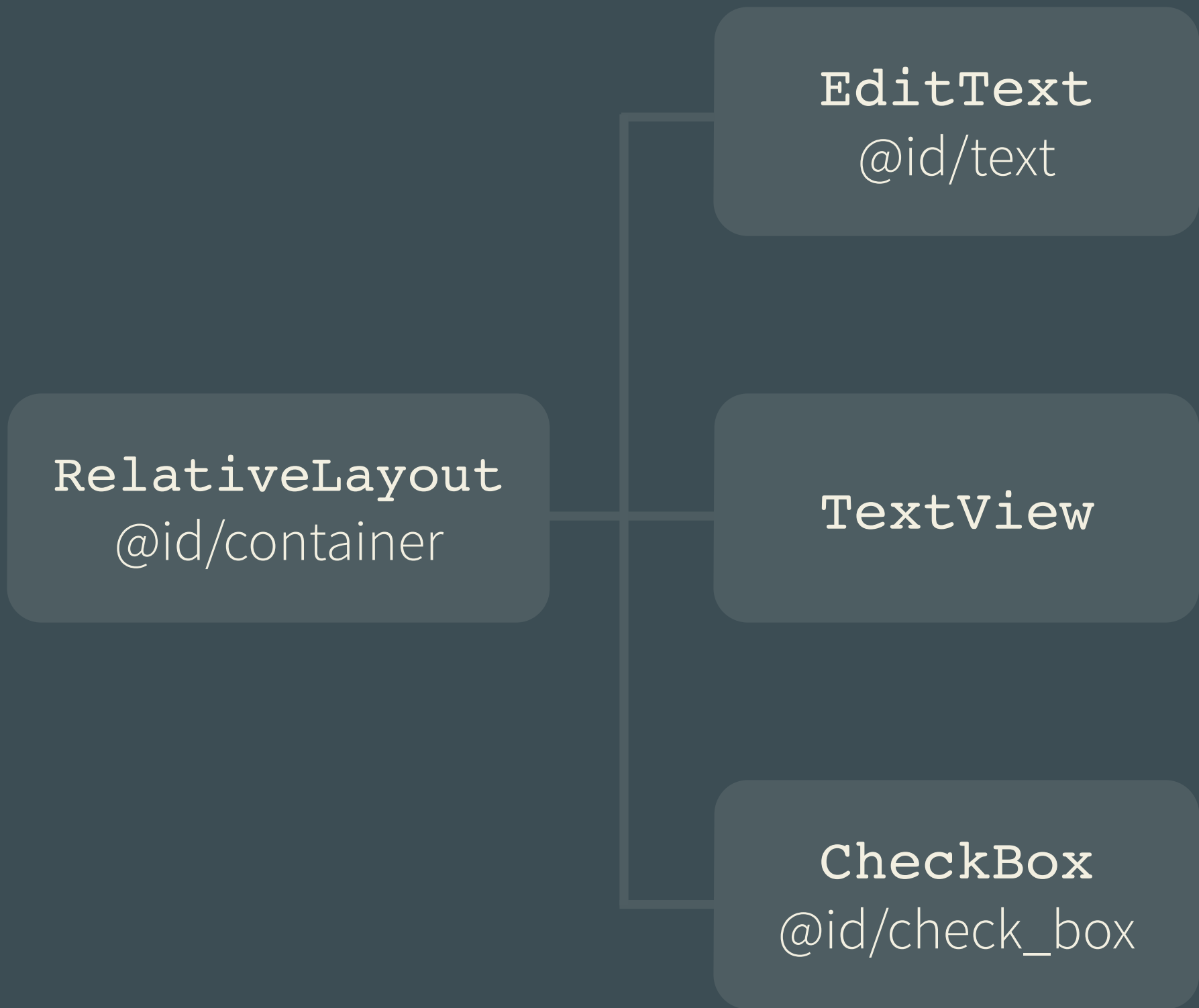


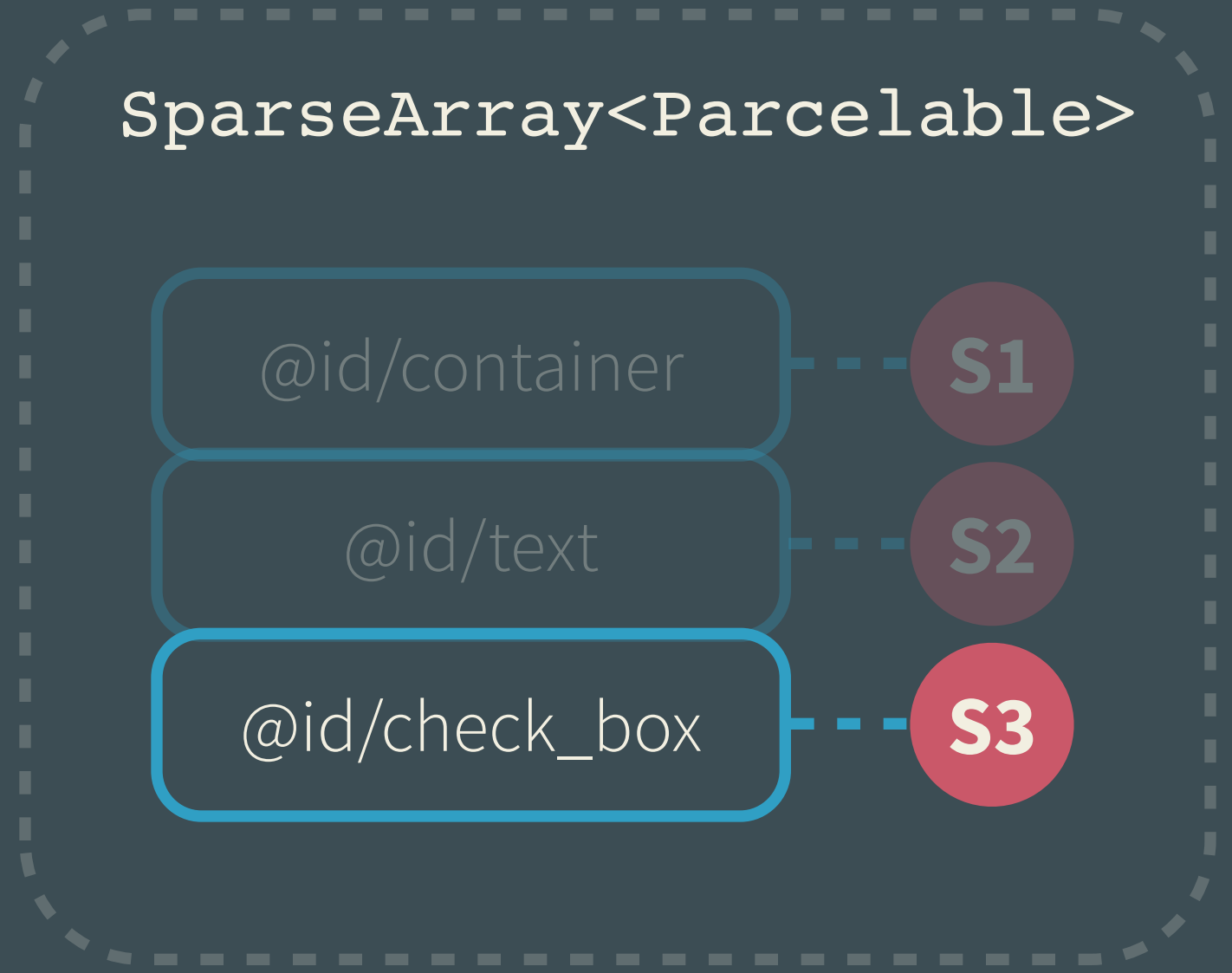
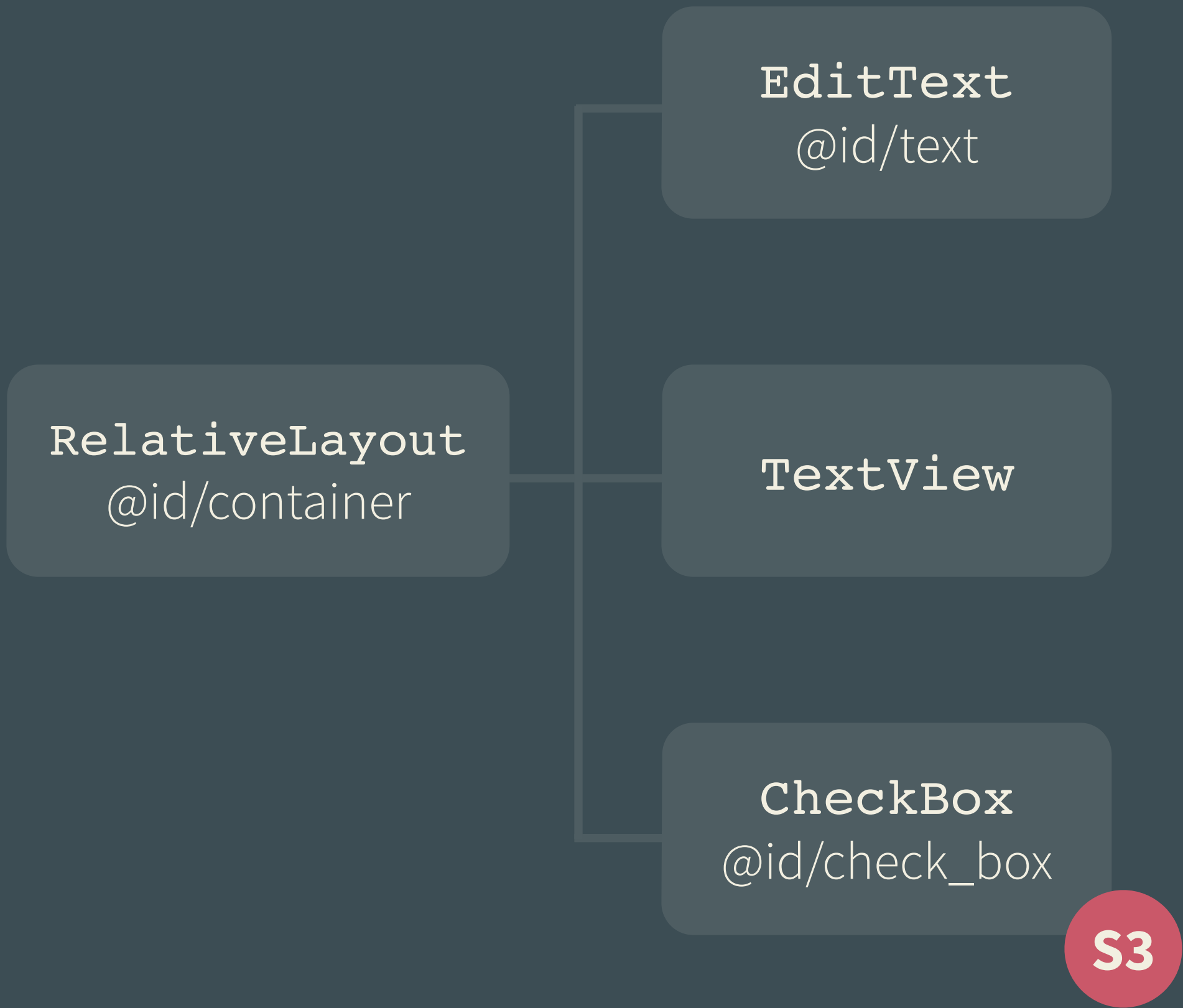








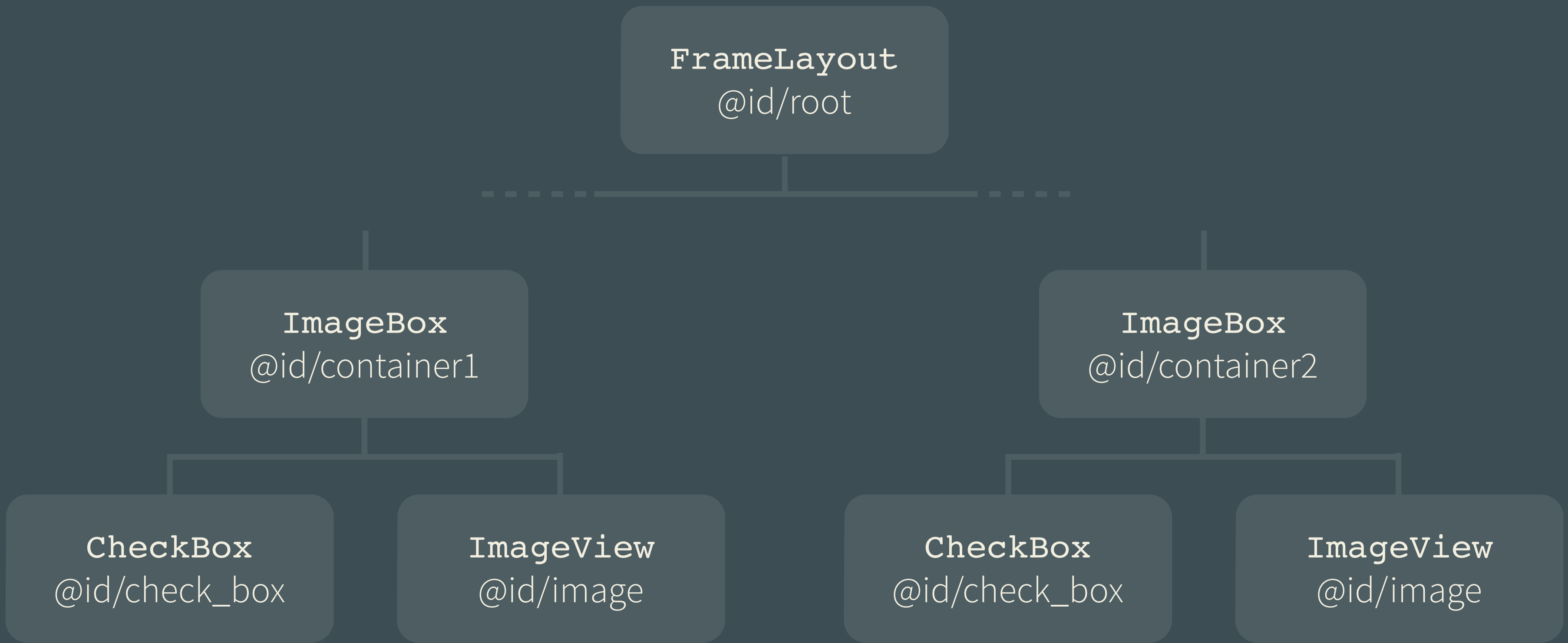


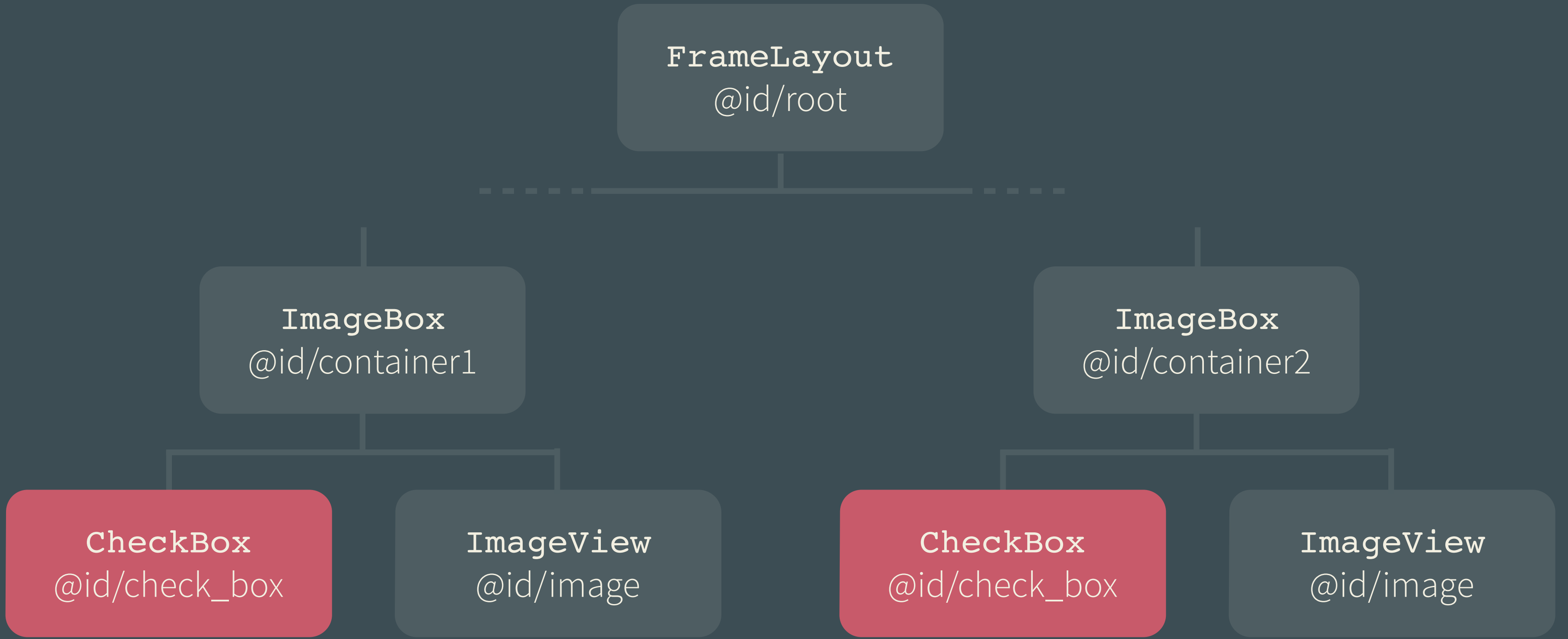


Ensure your Views' IDs are
unique & constant

```
1 static class SavedState extends BaseSavedState {
2     int checked;
3
4     SavedState(Parcelable superState) { super(superState); }
5
6     private SavedState(Parcel in) {
7         super(in);
8         checked = in.readInt();
9     }
10
11     @Override
12     public void writeToParcel(Parcel out, int flags) {
13         super.writeToParcel(out, flags);
14         out.writeInt(checked);
15     }
16
17     public static final Parcelable.Creator<SavedState> CREATOR =
18         new Parcelable.Creator<SavedState>() {
19             public SavedState createFromParcel(Parcel in) {
20                 return new SavedState(in);
21             }
22             public SavedState[] newArray(int size) {
23                 return new SavedState[size];
24             }
25         };
26 }
```

```
1 @Override
2 public Parcelable onSaveInstanceState() {
3     final Parcelable superState = super.onSaveInstanceState();
4     SavedState ss = new SavedState(superState);
5     ss.checked = isChecked() ? 1 : 0;
6     return ss;
7 }
8
9 @Override
10 public void onRestoreInstanceState(Parcelable state) {
11     SavedState ss = (SavedState) state;
12     super.onRestoreInstanceState(ss.getSuperState());
13     setChecked(ss.checked == 1);
14 }
```





Custom views with children with same IDs


```
1 static class SavedState extends BaseSavedState {
2
3     SparseArray childrenStates;
4
5     SavedState(Parcelable superState) { super(superState); }
6
7     private SavedState(Parcel in, ClassLoader loader) {
8         super(in);
9         childrenStates = in.readSparseArray(loader);
10    }
11
12    @Override
13    public void writeToParcel(Parcel out, int flags) {
14        super.writeToParcel(out, flags);
15        out.writeSparseArray(childrenStates);
16    }
17
18    public static final Creator<SavedState> CREATOR = ParcelableCompat.
19        newCreator(new ParcelableCompatCreatorCallbacks<SavedState>() {
20            @Override
21            public SavedState createFromParcel(Parcel source, ClassLoader loader) {
22                return new SavedState(source, loader);
23            }
24            @Override
25            public SavedState[] newArray(int size) {
26                return new SavedState[size];
27            }
28        });
29 }
```

```
1 @Override
2 public Parcelable onSaveInstanceState() {
3     final Parcelable superState = super.onSaveInstanceState();
4     SavedState ss = new SavedState(superState);
5     ss.childrenStates = new SparseArray<Parcelable>();
6     for (int i = 0; i < getChildCount(); i++) {
7         getChildAt(i).saveHierarchyState(ss.childrenStates);
8     }
9     return ss;
10 }
11
12 @Override
13 public void onRestoreInstanceState(Parcelable state) {
14     SavedState ss = (SavedState) state;
15     super.onRestoreInstanceState(ss.getSuperState());
16     for (int i = 0; i < getChildCount(); i++) {
17         getChildAt(i).restoreHierarchyState(ss.childrenStates);
18     }
19 }
```

That has solved nothing!

Still need to block save/restore dispatch

```
1 @Override
2 protected void dispatchSaveInstanceState(SparseArray<Parcelable> container) {
3     dispatchFreezeSelfOnly(container);
4 }
5
6 @Override
7 protected void dispatchRestoreInstanceState(SparseArray<Parcelable> container) {
8     dispatchThawSelfOnly(container);
9 }
```

Fragment level
state restoration

Very similar to Activities
state restoration lifecycle.

(Fragments are tied to Activity after all)

Fragment blocks Activity save mechanism

with framework

`setSaveFromParentEnabled(false)`

with support library

`NoSaveStateFrameLayout`

Fragment + View

common case

2 distinct states

View only

detach, addToBackStack, etc.

Leveraging save/restore

Can to be used to create smooth transitions between your Activities:

- Save the state S_A of A
- Start B with no animations passing S_A
- Apply S_A to B
- Transition between A and B was smooth

Summarizing
in three rules

Always save the state

An Android app must survive configuration changes & low memory conditions.

Only save essential info

*Only save info that is non persistent
or can not be reconstructed later.*

Use correct levels

Save instance states at the appropriate component level: Activity, Fragment or View.

Thank you!

@cyrilmottier

cyrilmottier.com

Resources

Dressed for Iceland • *Cécile Bernard*

Moelwynion, Eryri, Cymru • *Marc Poppleton*

Happy, Confused, Wink, Sad, Angry • *Megan Sheehan*

Floppy-Disk • *Alex Auda Samora*

Fonts

Source Sans Pro

Courier