Mobile Application Development

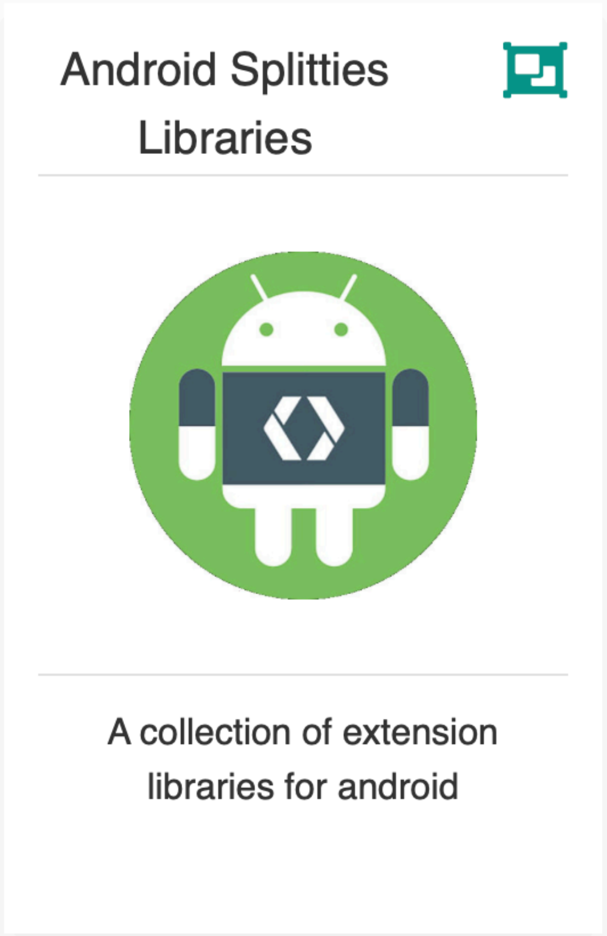
Produced by

David Drohan (ddrohan@wit.ie)

Department of Computing & Mathematics
Waterford Institute of Technology
http://www.wit.ie

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

Introducing Android *Splitties*



Android Splitties Libraries

A collection of extension libraries for android

Agenda

- ❑ Background
- ❑ Extension Functions
- ❑ The Android Splitties Packages
- ❑ KTX in our Case Study (Donation)

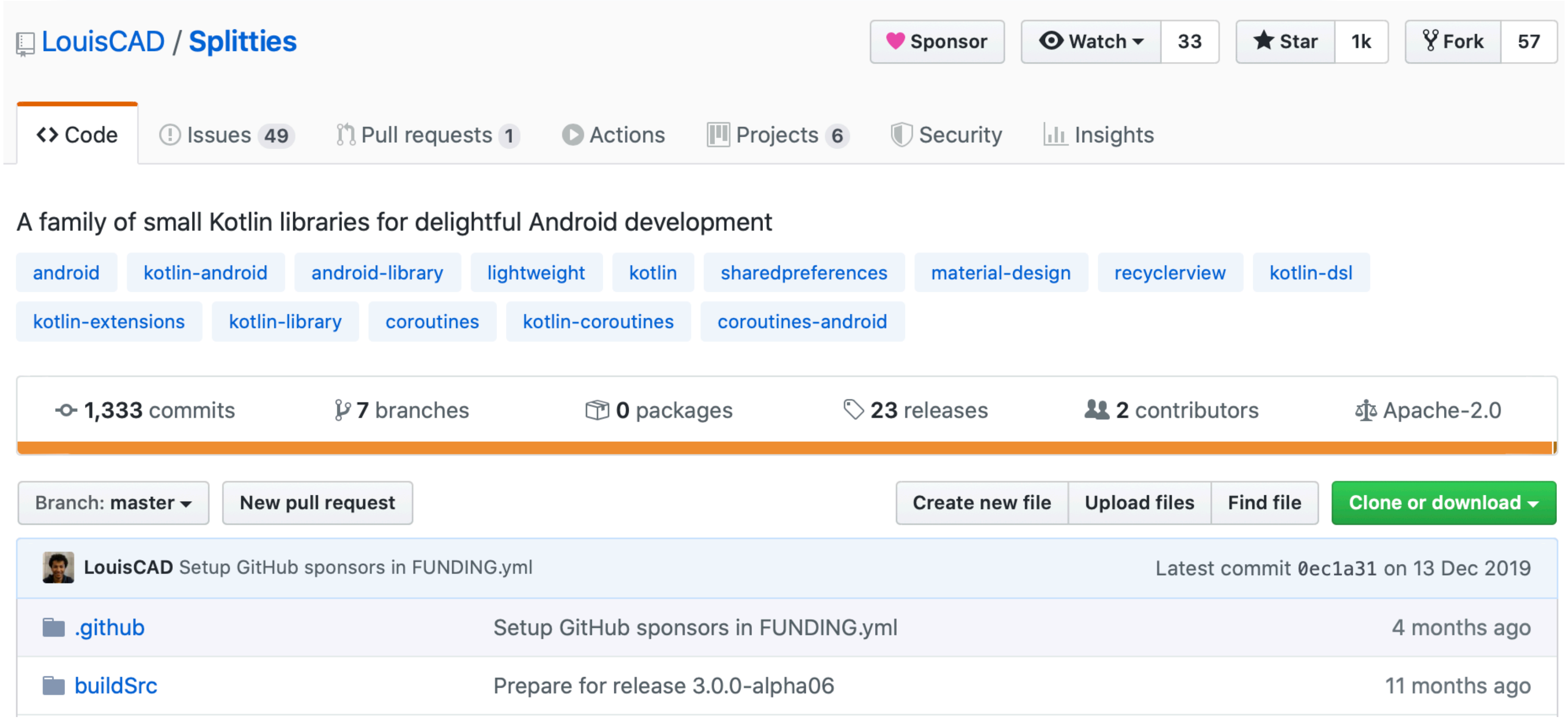
Agenda

- ❑ Background
- ❑ Extension Functions
- ❑ The Android Splitties Packages
- ❑ KTX in our Case Study (Donation)

What is it?

- Splitties a collection of small Kotlin multiplatform libraries (with Android as first target)
- intended to reduce the amount of code you have to write, so you can focus more on what you want to build for your users
- named as such because it is split in small modules, distributed as independent libraries - add only the ones you need - helping reduce the size of the final binary download

Official Docs (https://github.com/LouisCAD/Splitties)


LouisCAD / Splitties

Sponsor | Watch 33 | Star 1k | Fork 57

Code | Issues 49 | Pull requests 1 | Actions | Projects 6 | Security | Insights

A family of small Kotlin libraries for delightful Android development

android kotlin-android android-library lightweight kotlin sharedpreferences material-design recyclerview kotlin-dsl kotlin-extensions kotlin-library coroutines kotlin-coroutines coroutines-android

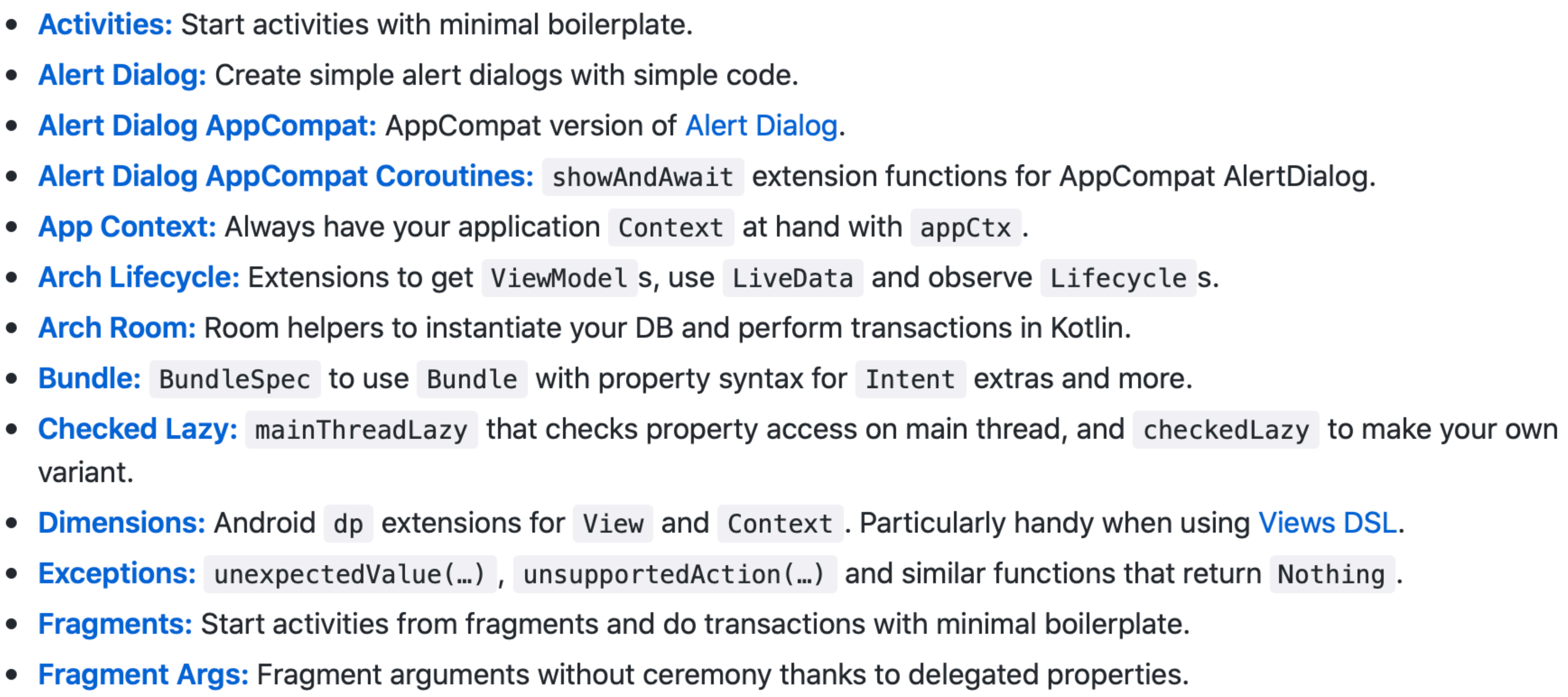
1,333 commits | 7 branches | 0 packages | 23 releases | 2 contributors | Apache-2.0

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

LouisCAD Setup GitHub sponsors in FUNDING.yml — Latest commit `0ec1a31` on 13 Dec 2019

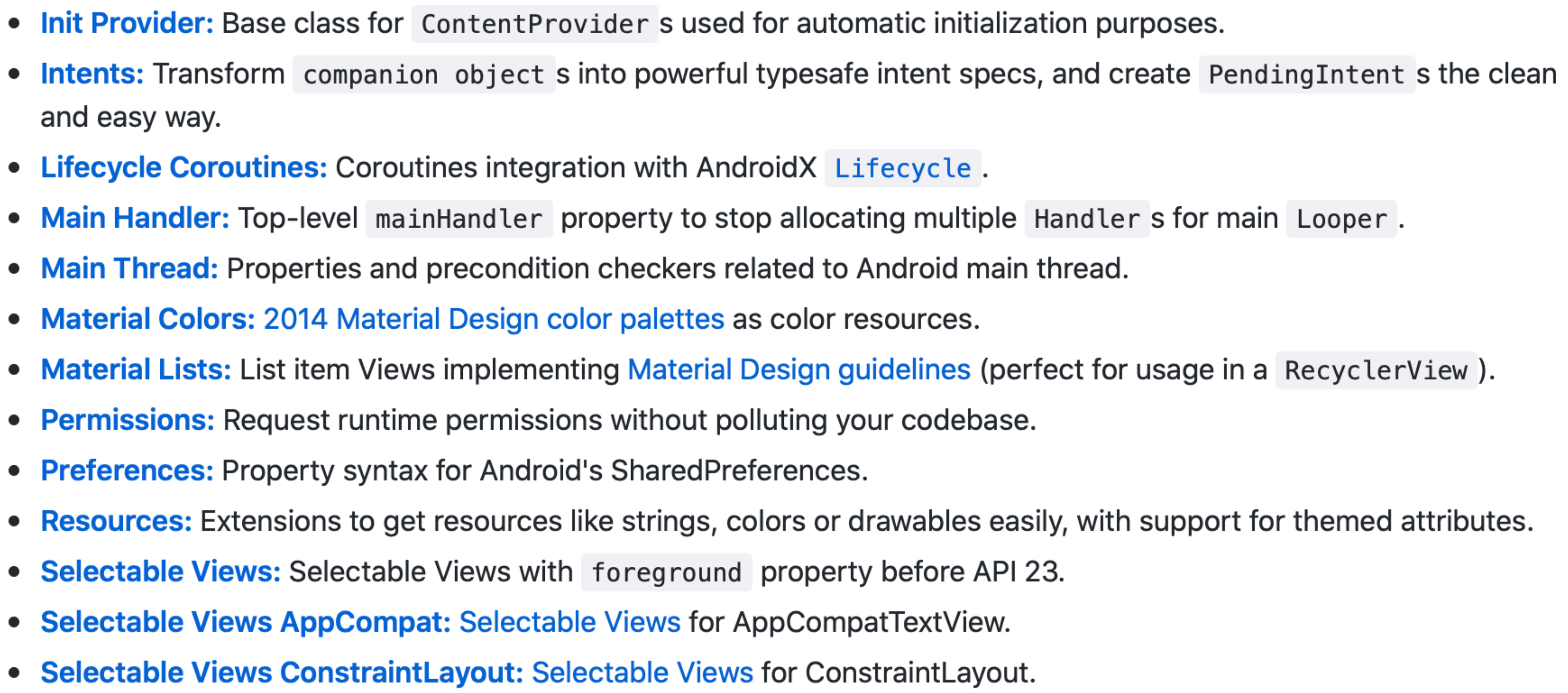
.github	Setup GitHub sponsors in FUNDING.yml	4 months ago
buildSrc	Prepare for release 3.0.0-alpha06	11 months ago

Official Docs (https://developer.android.com/kotlin/ktx)



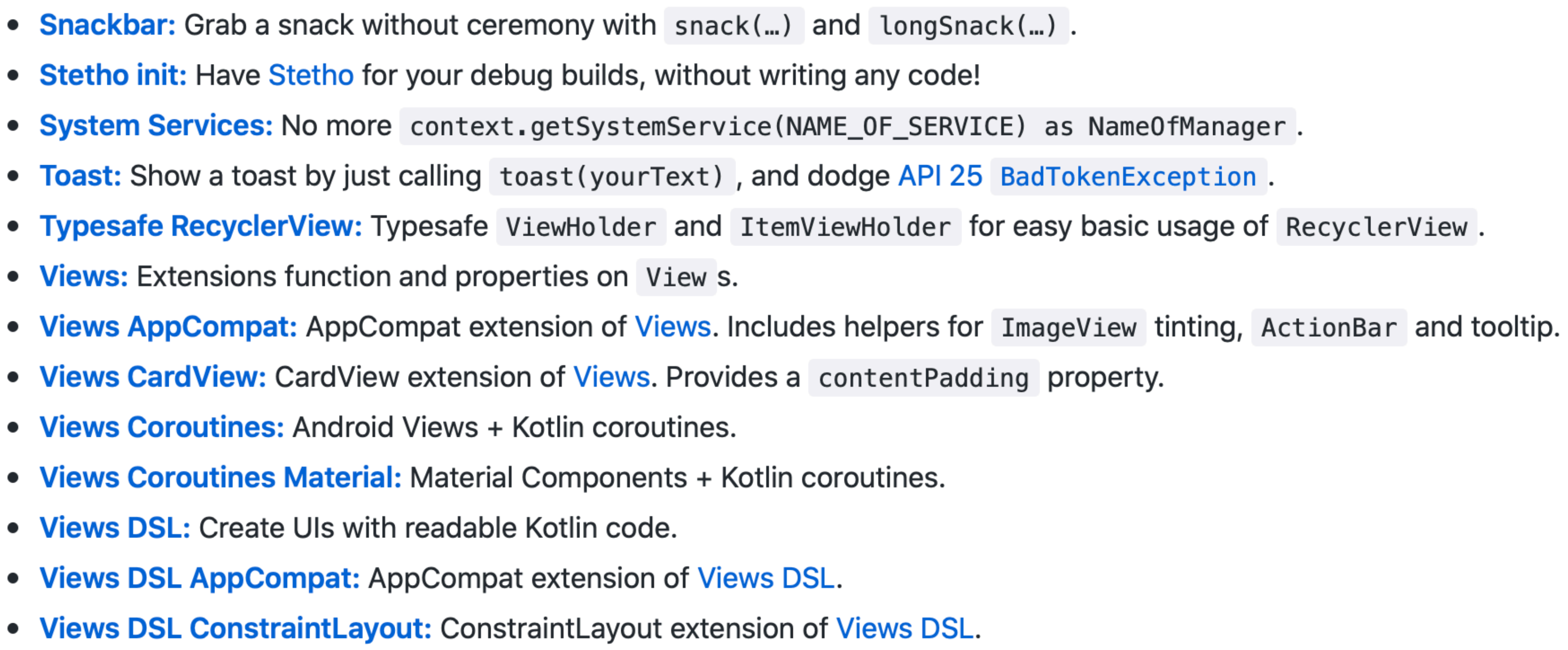
- **Activities:** Start activities with minimal boilerplate.
- **Alert Dialog:** Create simple alert dialogs with simple code.
- **Alert Dialog AppCompat:** AppCompat version of Alert Dialog.
- **Alert Dialog AppCompat Coroutines:** `showAndAwait` extension functions for AppCompat AlertDialog.
- **App Context:** Always have your application `Context` at hand with `appCtx`.
- **Arch Lifecycle:** Extensions to get `ViewModel`s, use `LiveData` and observe `Lifecycle`s.
- **Arch Room:** Room helpers to instantiate your DB and perform transactions in Kotlin.
- **Bundle:** `BundleSpec` to use `Bundle` with property syntax for `Intent` extras and more.
- **Checked Lazy:** `mainThreadLazy` that checks property access on main thread, and `checkedLazy` to make your own variant.
- **Dimensions:** Android `dp` extensions for `View` and `Context`. Particularly handy when using Views DSL.
- **Exceptions:** `unexpectedValue(…)`, `unsupportedAction(…)` and similar functions that return `Nothing`.
- **Fragments:** Start activities from fragments and do transactions with minimal boilerplate.
- **Fragment Args:** Fragment arguments without ceremony thanks to delegated properties.

Official Docs (https://developer.android.com/kotlin/ktx)



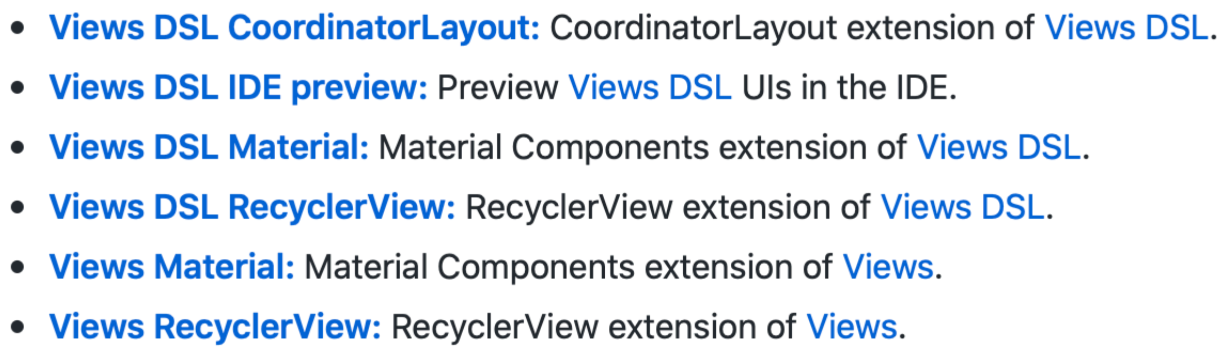
- **Init Provider:** Base class for `ContentProvider`s used for automatic initialization purposes.
- **Intents:** Transform `companion object`s into powerful typesafe intent specs, and create `PendingIntent`s the clean and easy way.
- **Lifecycle Coroutines:** Coroutines integration with AndroidX `Lifecycle`.
- **Main Handler:** Top-level `mainHandler` property to stop allocating multiple `Handler`s for main `Looper`.
- **Main Thread:** Properties and precondition checkers related to Android main thread.
- **Material Colors:** 2014 Material Design color palettes as color resources.
- **Material Lists:** List item Views implementing Material Design guidelines (perfect for usage in a `RecyclerView`).
- **Permissions:** Request runtime permissions without polluting your codebase.
- **Preferences:** Property syntax for Android's SharedPreferences.
- **Resources:** Extensions to get resources like strings, colors or drawables easily, with support for themed attributes.
- **Selectable Views:** Selectable Views with `foreground` property before API 23.
- **Selectable Views AppCompat:** Selectable Views for AppCompatTextView.
- **Selectable Views ConstraintLayout:** Selectable Views for ConstraintLayout.

Official Docs (https://developer.android.com/kotlin/ktx)



- **Snackbar:** Grab a snack without ceremony with `snack(…)` and `longSnack(…)`.
- **Stetho init:** Have Stetho for your debug builds, without writing any code!
- **System Services:** No more `context.getSystemService(NAME_OF_SERVICE) as NameOfManager`.
- **Toast:** Show a toast by just calling `toast(yourText)`, and dodge API 25 `BadTokenException`.
- **Typesafe RecyclerView:** Typesafe `ViewHolder` and `ItemViewHolder` for easy basic usage of `RecyclerView`.
- **Views:** Extensions function and properties on `View`s.
- **Views AppCompat:** AppCompat extension of Views. Includes helpers for `ImageView` tinting, `ActionBar` and tooltip.
- **Views CardView:** CardView extension of Views. Provides a `contentPadding` property.
- **Views Coroutines:** Android Views + Kotlin coroutines.
- **Views Coroutines Material:** Material Components + Kotlin coroutines.
- **Views DSL:** Create UIs with readable Kotlin code.
- **Views DSL AppCompat:** AppCompat extension of Views DSL.
- **Views DSL ConstraintLayout:** ConstraintLayout extension of Views DSL.

Official Docs (https://developer.android.com/kotlin/ktx)

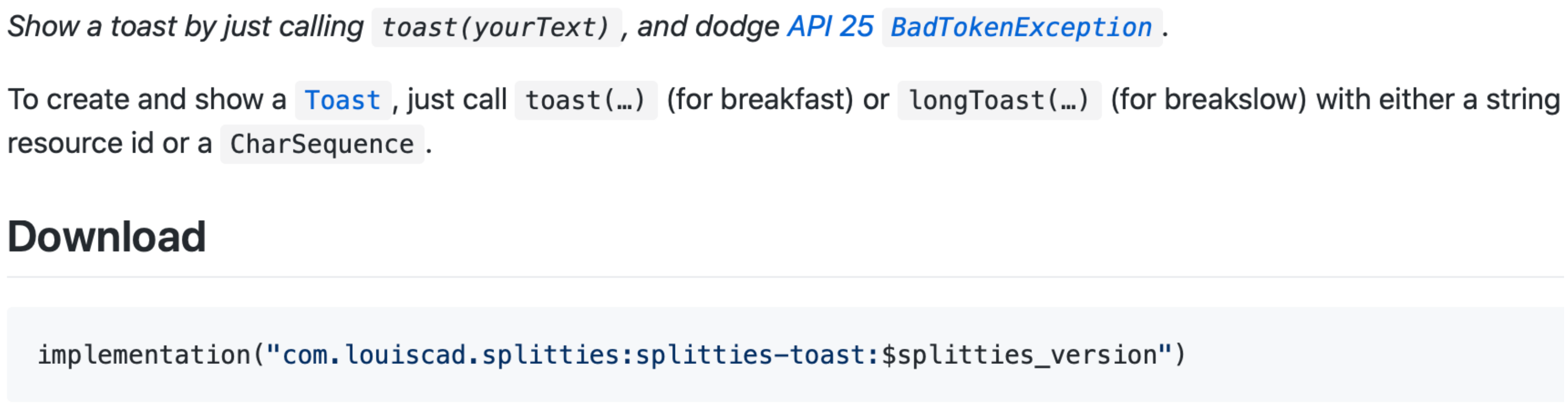


- **Views DSL CoordinatorLayout:** CoordinatorLayout extension of Views DSL.
- **Views DSL IDE preview:** Preview Views DSL UIs in the IDE.
- **Views DSL Material:** Material Components extension of Views DSL.
- **Views DSL RecyclerView:** RecyclerView extension of Views DSL.
- **Views Material:** Material Components extension of Views.
- **Views RecyclerView:** RecyclerView extension of Views.

Android Splitties

some examples…

Toasts



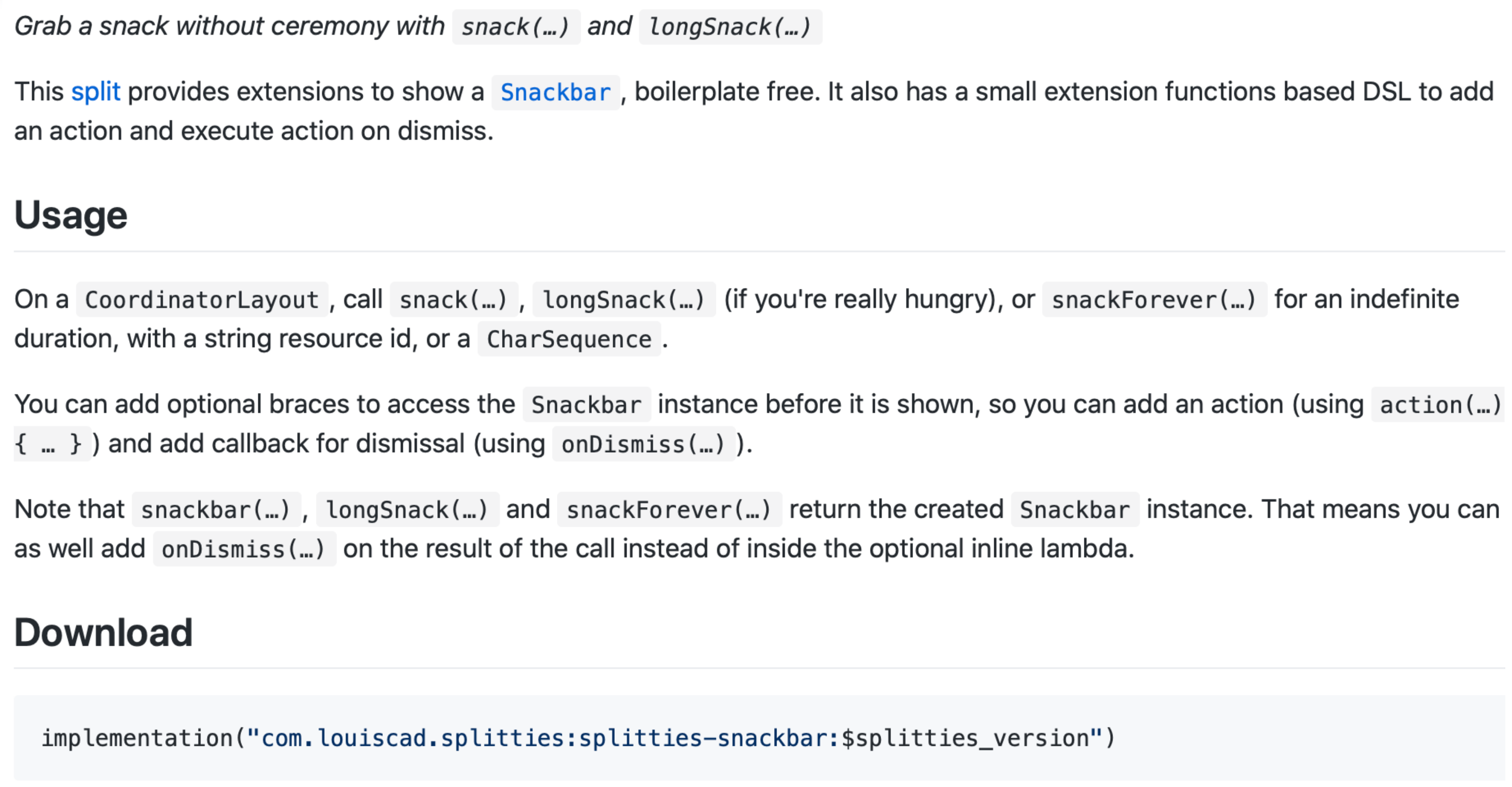
Show a toast by just calling `toast(yourText)`, and dodge API 25 `BadTokenException`.

To create and show a `Toast`, just call `toast(…)` (for breakfast) or `longToast(…)` (for breakslow) with either a string resource id or a `CharSequence`.

Download

```
implementation("com.louiscad.splitties:splitties-toast:$splitties_version")
```

Snackbars



Grab a snack without ceremony with `snack(…)` and `longSnack(…)`

This split provides extensions to show a `Snackbar`, boilerplate free. It also has a small extension functions based DSL to add an action and execute action on dismiss.

Usage

On a `CoordinatorLayout`, call `snack(…)`, `longSnack(…)` (if you're really hungry), or `snackForever(…)` for an indefinite duration, with a string resource id, or a `CharSequence`.

You can add optional braces to access the `Snackbar` instance before it is shown, so you can add an action (using `action(…) { … }`) and add callback for dismissal (using `onDismiss(…)`).

Note that `snackbar(…)`, `longSnack(…)` and `snackForever(…)` return the created `Snackbar` instance. That means you can as well add `onDismiss(…)` on the result of the call instead of inside the optional inline lambda.

Download

```
implementation("com.louiscad.splitties:splitties-snackbar:$splitties_version")
```

Alert Dialogs

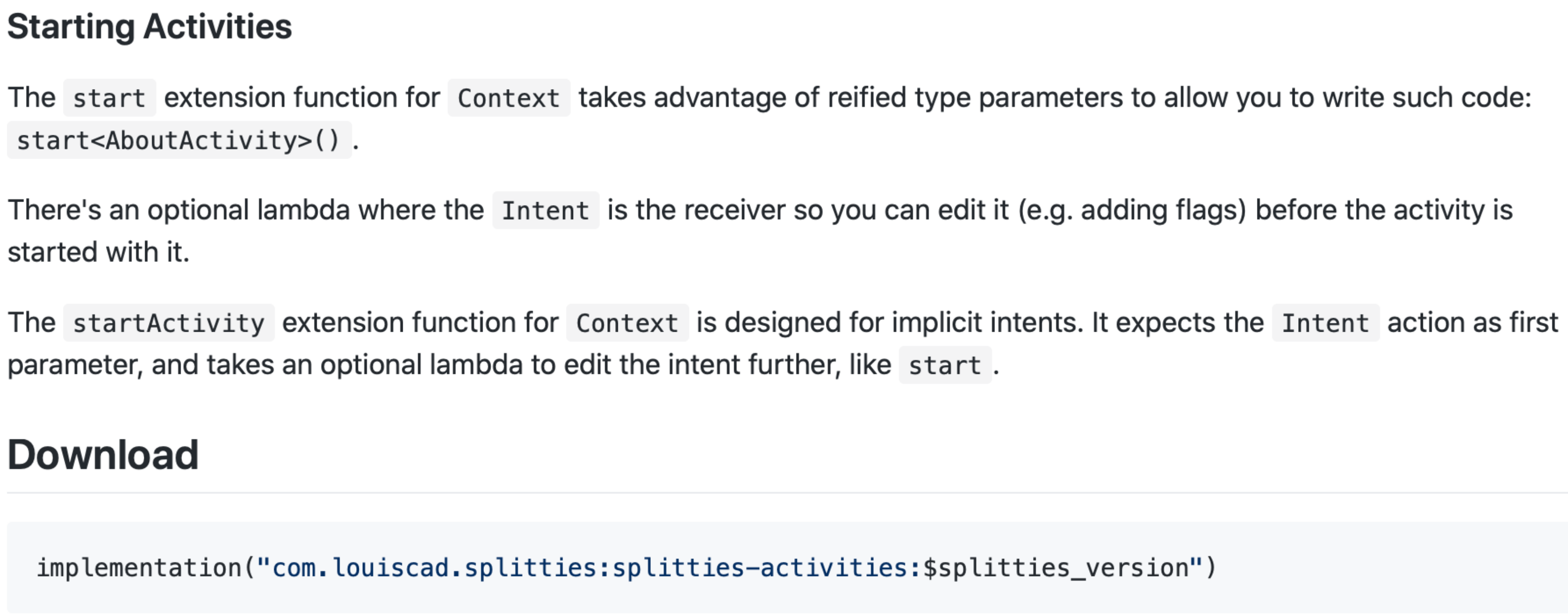
Create simple alert dialogs with simple code

```
private fun doIrreversibleStuffOrCancel() {
    alertDialog {
        messageResource = R.string.dialog_msg_confirm_irreversible_stuff
        okButton { irreversibleStuff() }
        cancelButton()
    }.onShow {
        positiveButton.textColorResource = R.color.red_500
    }.show()
}
```

Download

```
implementation("com.louiscad.splitties:splitties-alertdialog:$splitties_version")
```

Activities



Starting Activities

The `start` extension function for `Context` takes advantage of reified type parameters to allow you to write such code: `start<AboutActivity>()` .

There's an optional lambda where the `Intent` is the receiver so you can edit it (e.g. adding flags) before the activity is started with it.

The `startActivity` extension function for `Context` is designed for implicit intents. It expects the `Intent` action as first parameter, and takes an optional lambda to edit the intent further, like `start` .

Download

```
implementation("com.louiscad.splitties:splitties-activities:$splitties_version")
```

Agenda

We’ll have a look at some examples in practice in Donation in the next section

References

Sources: https://github.com/LouisCAD/Splitties

Thanks.