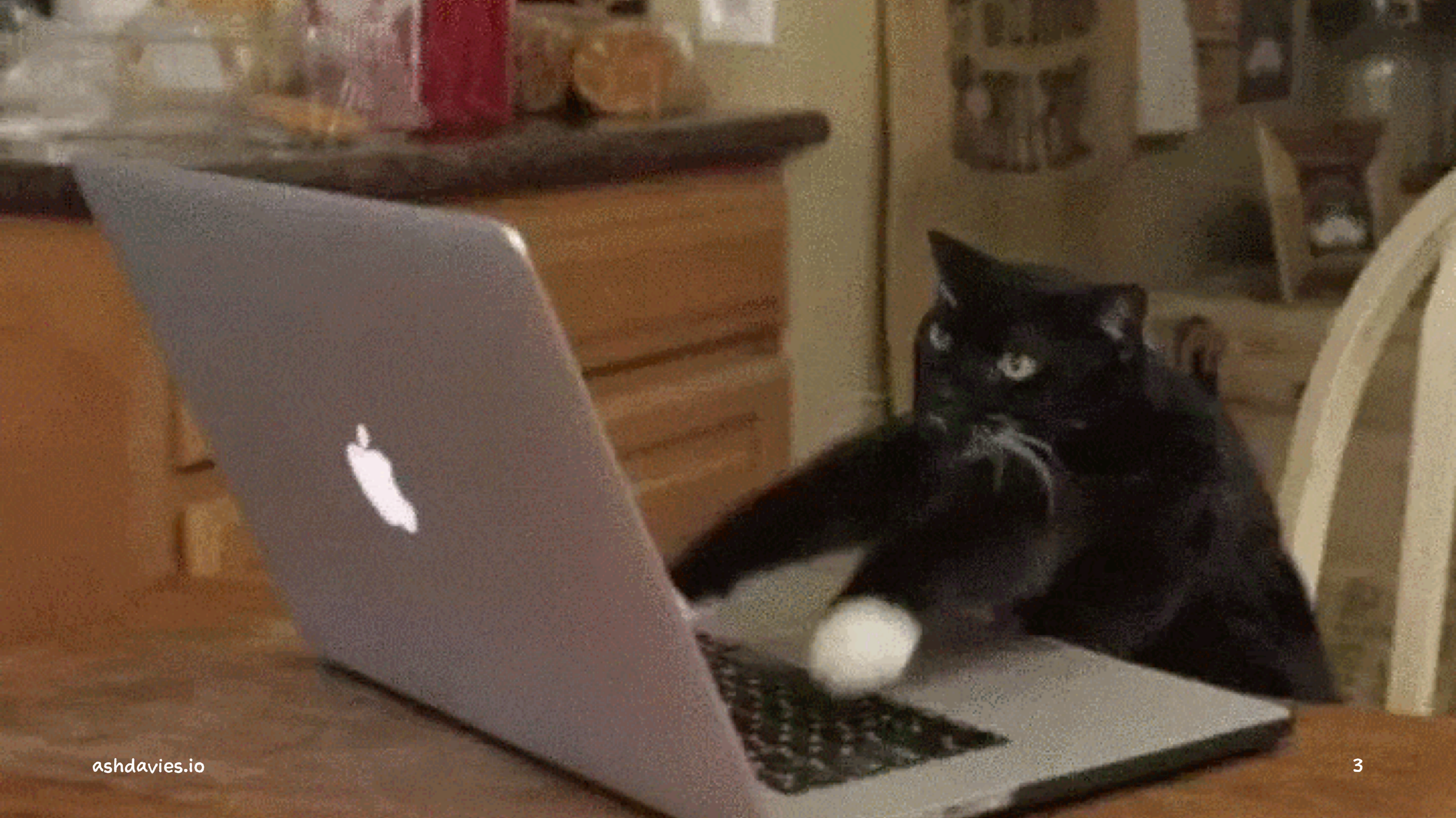
IMMOBILIEN SCOUT 24

Leveraging Android Databinding with Kotlin

2018

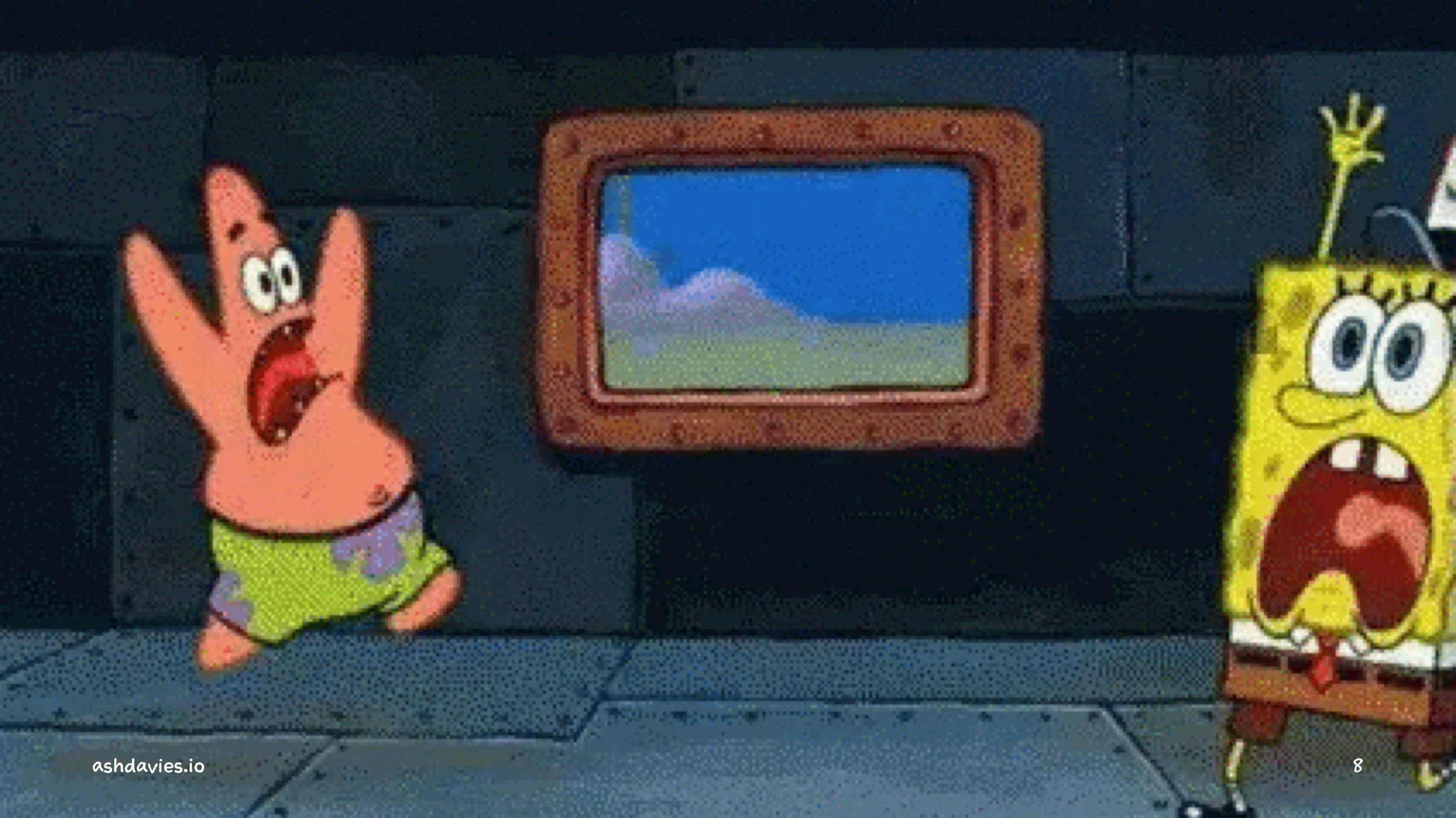


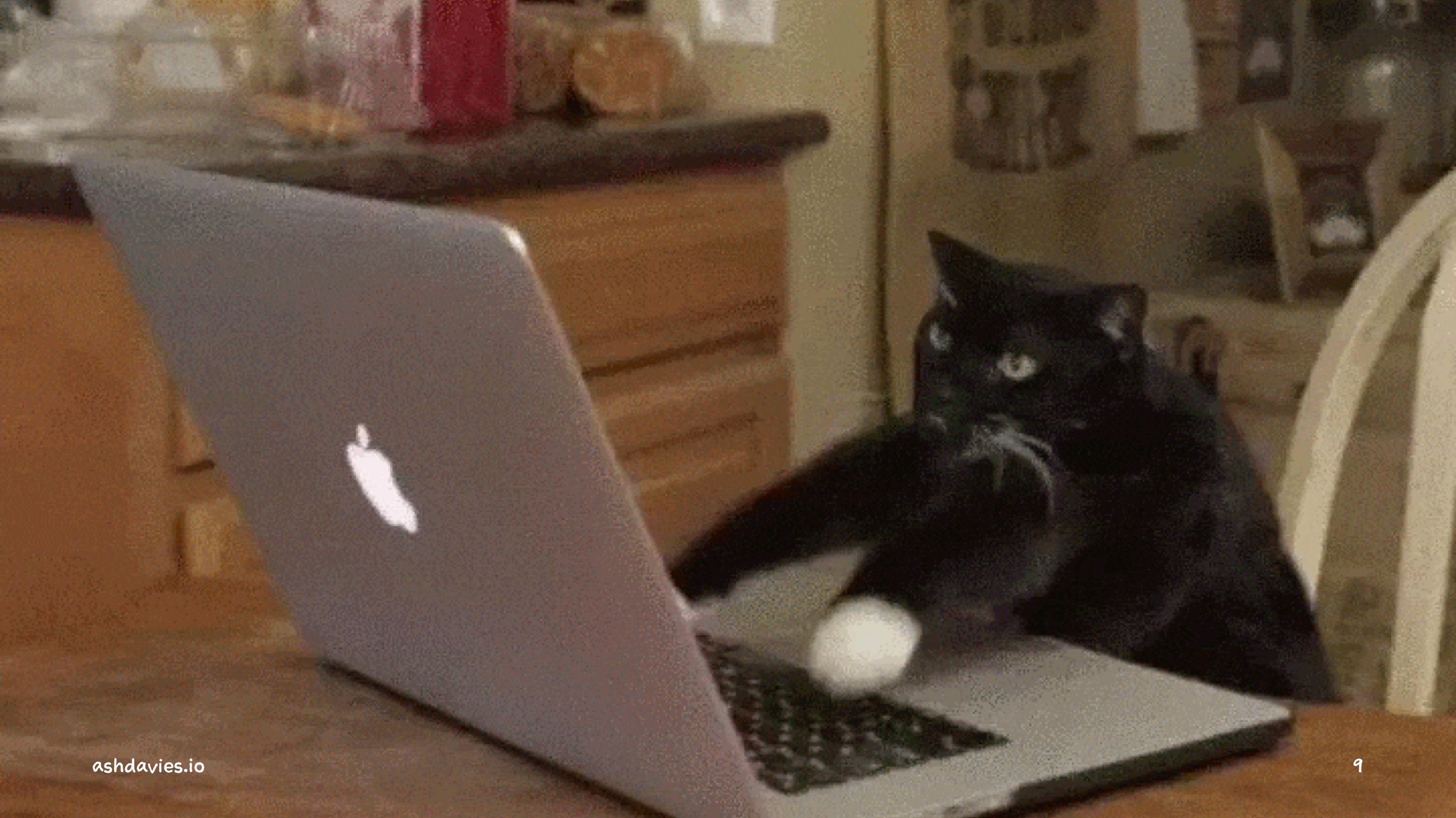
.droidconBerlin


You're in!

Check this out! Ash Davies will talk about "Leveraging Android databinding with Kotlin" at #DCBERLIN18 bit.ly/2lkVGKE

droidcon Berlin @droidconBerlin
4:21pm - 16 May 2018





ContentOverflowException!


Google Developers
#DCBERLIN18
AI

Android Data Binding

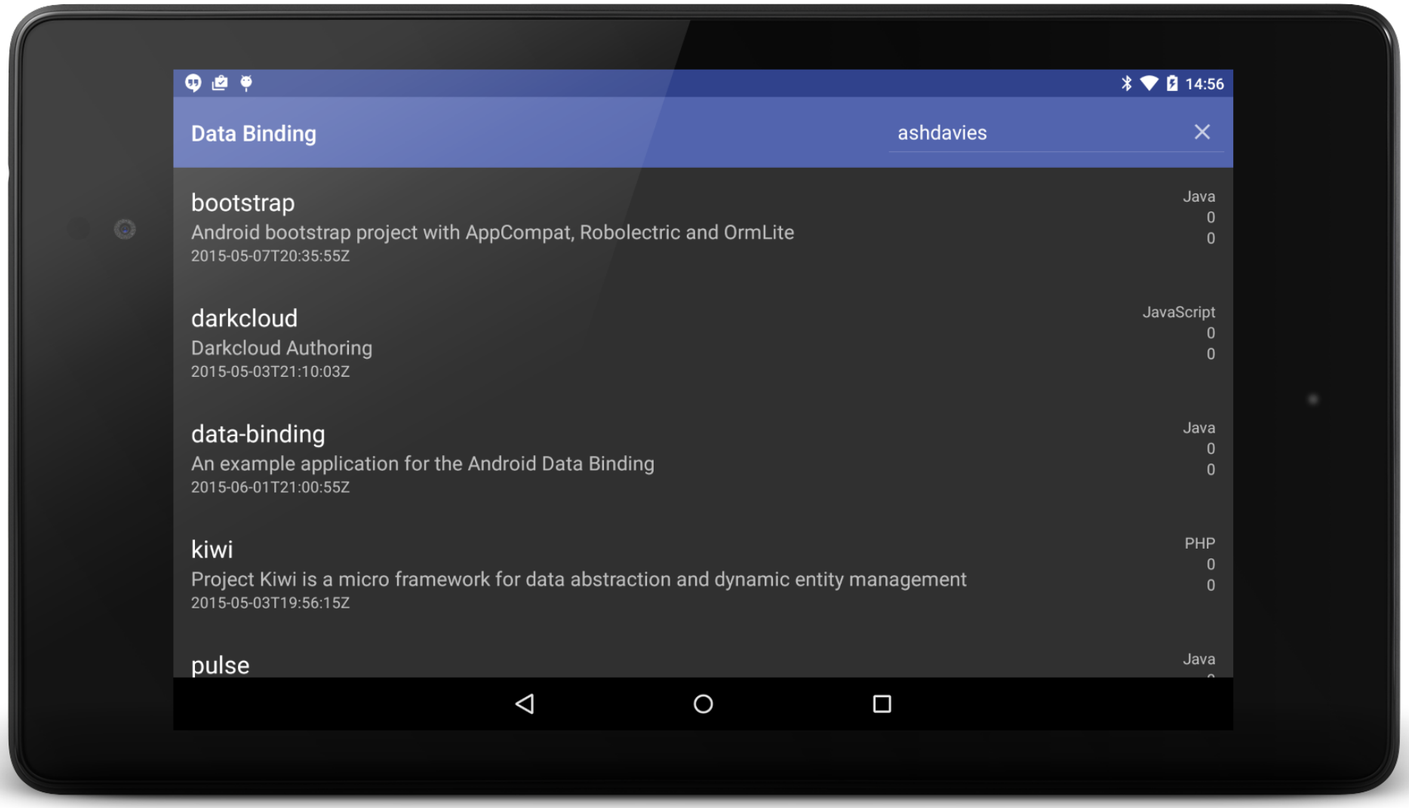
What is Data Binding?

Android Data Binding

Data Binding Beta

2015

Data Binding Sample



Data Binding Sample

Reza Goli 11 Nov 2015 at 00:46 1 0

★☆☆☆☆

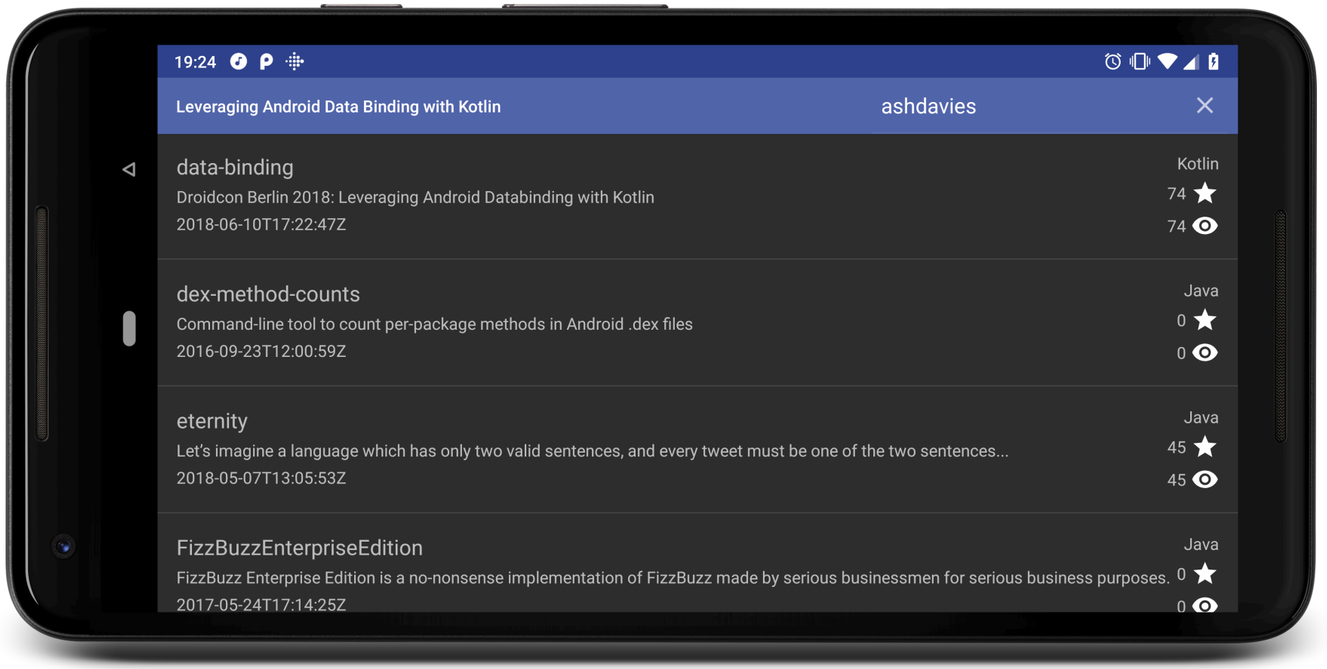
What's that.

Reply to this review

Device: L Fino (l70pds) App version: – OS: Android 4.4 MORE

Data Binding Re-Sample

github.com/ashdavies/data-binding



Data Binding Re-Sample

- Kotlin Coroutines

Data Binding Re-Sample

- Kotlin Coroutines
- Constraint Layout

Data Binding Re-Sample

- Kotlin Coroutines
- Constraint Layout
- Architecture Components

Data Binding Re-Sample

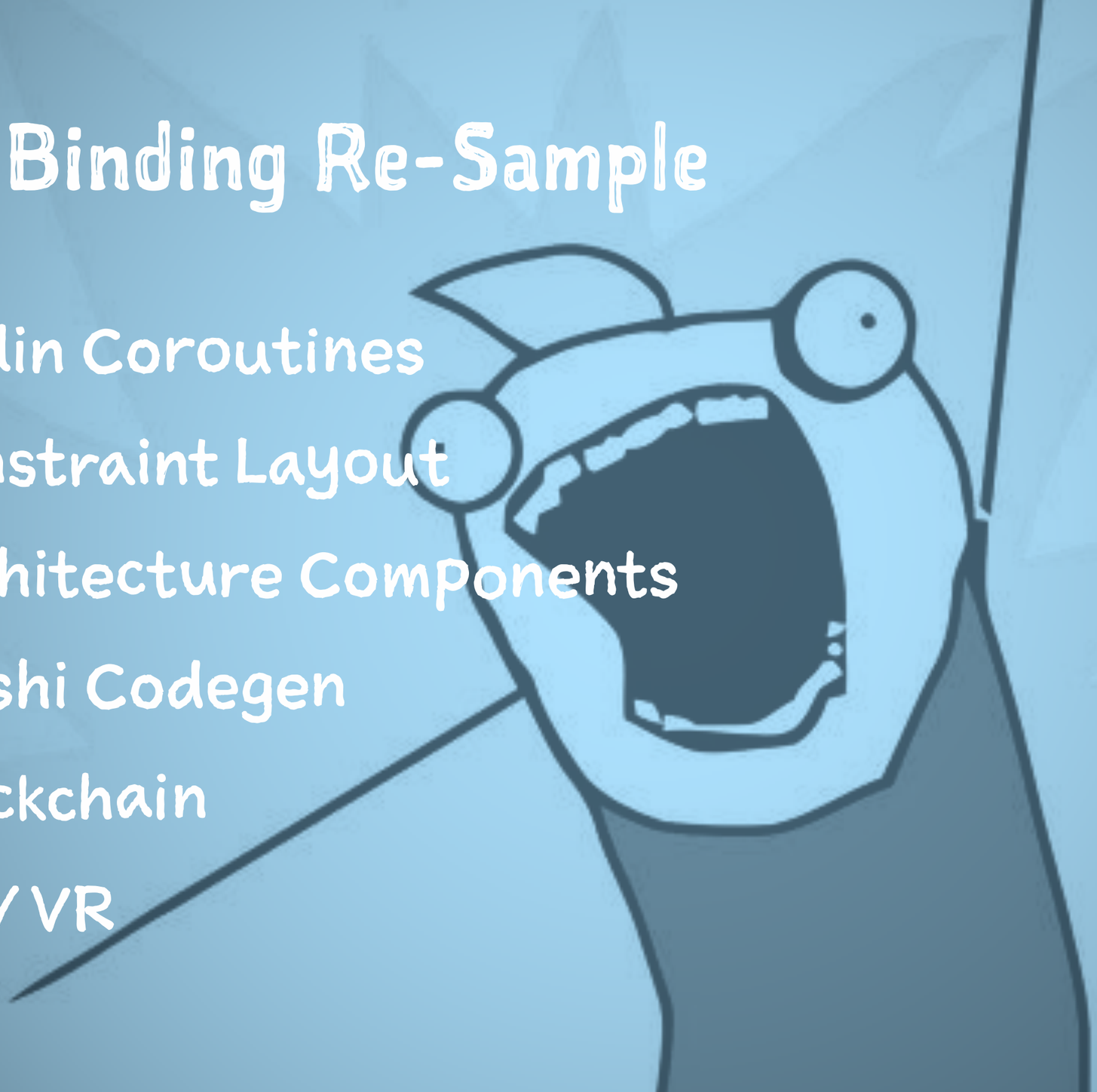
- Kotlin Coroutines
- Constraint Layout
- Architecture Components
- Moshi Codegen

Data Binding Re-Sample

- Kotlin Coroutines
- Constraint Layout
- Architecture Components
- Moshi Codegen
- Blockchain

Data Binding Re-Sample

- Kotlin Coroutines
- Constraint Layout
- Architecture Components
- Moshi Codegen
- Blockchain
- AR / VR



Data Binding Re-Sample

- Kotlin Coroutines
- Constraint Layout
- Architecture Components
- Moshi Codegen
- ~~Blockchain~~
- ~~AR / VR~~

Setup

Setup

build.gradle

```
android {
  dataBinding {
    enabled = true
  }
}
```

Data Binding v2

Data Binding v2

Incremental class generation ⚡

Data Binding v2

Incremental class generation ⚡

Pre-compilation class generation 💪

Getting Started

Data Binding

```
class RepoActivity : AppCompatActivity() {

  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_repo)
  }
}
```

Data Binding

```
class RepoActivity : AppCompatActivity() {

  private lateinit var binding: ActivityRepoBinding

  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)

    binding = DataBindingUtil.setContentView(this, R.layout.activity_repo)
  }
}
```

Data Binding

```
internal class RepoActivity : AppCompatActivity() {

  private lateinit var binding: ActivityRepoBinding

  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)

    binding = DataBindingUtil.setContentView(this, R.layout.activity_repo)
    binding.setLifecycleOwner(this)
  }
}
```

Data Binding

```
internal class Repoactivity : AppCompatActivity() {

  private lateinit var binding: ActivityRepoBinding
  private lateinit var service: RepoService

  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)

    binding = DataBindingUtil.setContentView(this, R.layout.activity_repo)
    binding.setLifecycleOwner(this)

    service = RepoServiceFactory().get()
  }

  override fun onResume() {
    super.onResume()
    service.fetch { binding.items = it }
  }
}
```

Layout (XML 🤦‍♀️)

```
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">

  <data>

    <variable
        name="items"
        type="java.util.List&lt;io.ashdavies.databinding.Repo&gt;"/>

  </data>

  <androidx.coordinatorlayout.widget.CoordinatorLayout... />
</layout>
```

Data Binding

```
class RepoActivity : AppCompatActivity() {

  private lateinit var binding: ActivityRepoBinding
  private lateinit var service: RepoService

  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)

    binding = DataBindingUtil.setContentView(this, R.layout.activity_home)
    binding.setLifecycleOwner(this)

    service = RepoServiceFactory().get()
  }

  override fun onResume() {
    super.onResume()
    service.fetch { binding.items = it }
  }
}
```

Delegated Properties

Delegated Properties

```
val value: String by lazy {
  println("computed!")
  "Hello"
}
```

Delegated Properties

```
interface ReadOnlyProperty<in R, out T> {
  operator fun getValue(thisRef: R, property: KProperty<*>): T
}

interface ReadWriteProperty<in R, T> {
  operator fun getValue(thisRef: R, property: KProperty<*>): T
  operator fun setValue(thisRef: R, property: KProperty<*>, value: T)
}
```

Delegated Properties

```
class ActivityBindingProperty<out T : ViewDataBinding> : ReadOnlyProperty<Activity, T> {

  override operator fun getValue(thisRef: Activity, property: KProperty<*>): T {
    TODO("not implemented")
  }
}
```

Delegated Properties

```
class ActivityBindingProperty<out T : ViewDataBinding>(
  @LayoutRes private val resId: Int
) : ReadOnlyProperty<Activity, T> {

  override operator fun getValue(thisRef: Activity, property: KProperty<*>): T {
    TODO("not implemented")
  }

  private fun createBinding(activity: Activity): T {
    return DataBindingUtil.setContentView(activity, resId)
  }
}
```

Delegated Properties

```
class ActivityBindingProperty<out T : ViewDataBinding>(
  @LayoutRes private val resId: Int
) : ReadOnlyProperty<Activity, T> {

  private var binding: T? = null

  override operator fun getValue(thisRef: Activity, property: KProperty<*>): T {
    return binding ?: createBinding(thisRef).also { binding = it }
  }

  private fun createBinding(activity: Activity): T {
    return DataBindingUtil.setContentView(activity, resId)
  }
}
```

Delegated Properties

```
class RepoActivity : AppCompatActivity() {

  private val binding by ActivityBindingProperty<ActivityRepoBinding>(R.layout.activity_repo)

  private lateinit var service: RepoService

  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    binding.setLifecycleOwner(this)

    service = RepoServiceFactory().get()
  }

  override fun onResume() {
    super.onResume()
    service.fetch { binding.items = it }
  }
}
```

ProTip: Extension Function! 👍

```
class RepoActivity : AppCompatActivity() {

  private val binding by activityBinding<ActivityRepoBinding>(R.layout.activity_repo)

  private lateinit var service: RepoService

  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    binding.setLifecycleOwner(this)

    service = RepoServiceFactory().get()
  }

  override fun onResume() {
    super.onResume()
    service.fetch { binding.items = it }
  }
}

fun <T : ViewDataBinding> FragmentActivity.activityBinding(
  @LayoutRes resId: Int
) = ActivityBindingProperty(resId)
```

Data Binding

```
class RepoActivity : AppCompatActivity() {

  private val binding by activityBinding<ActivityRepoBinding>(R.layout.activity_repo)

  private lateinit var service: RepoService

  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    binding.setLifecycleOwner(this)

    service = RepoServiceFactory().get()
  }

  override fun onResume() {
    super.onResume()
    service.fetch { binding.items = it }
  }
}

fun <T : ViewDataBinding> FragmentActivity.activityBinding(
  @LayoutRes resId: Int
) = ActivityBindingProperty(resId)
```

Introducing

Android Architecture Components

Model-View-ViewModel


View
ViewModel
Model

Model-View-ViewModel

```
class RepoActivity : AppCompatActivity() {

  private val binding by activityBinding<ActivityRepoBinding>(R.layout.activity_repo)

  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    binding.setLifecycleOwner(this)
  }
}
```

Model-View-ViewModel

```
class RepoActivity : AppCompatActivity() {

  private val binding by activityBinding<ActivityRepoBinding>(R.layout.activity_repo)

  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)

    binding.setLifecycleOwner(this)
    binding.model = ViewModelProviders
        .of(this, RepoViewModelFactory())
        .get(RepoViewModel::class.java)
  }
}
```

ProTip: Extension Function! 👍

```
class RepoActivity : AppCompatActivity() {

  private val binding by activityBinding<ActivityRepoBinding>(R.layout.activity_repo)

  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)

    binding.setLifecycleOwner(this)
    binding.model = getViewModel(RepoViewModelFactory())
  }
}

inline fun <reified T : ViewModel> FragmentActivity.getViewModel(
  factory: ViewModelProvider.Factory = ViewModelProvider.NewInstanceFactory()
) = ViewModelProviders.of(this, factory).get(T::class.java)
```

Model-View-ViewModel

```
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">

  <data>

    <variable
        name="model"
        type="io.ashdavies.databinding.RepoViewModel"/>

  </data>

  <androidx.coordinatorlayout.widget.CoordinatorLayout... />
</layout>
```

Model-View-ViewModel

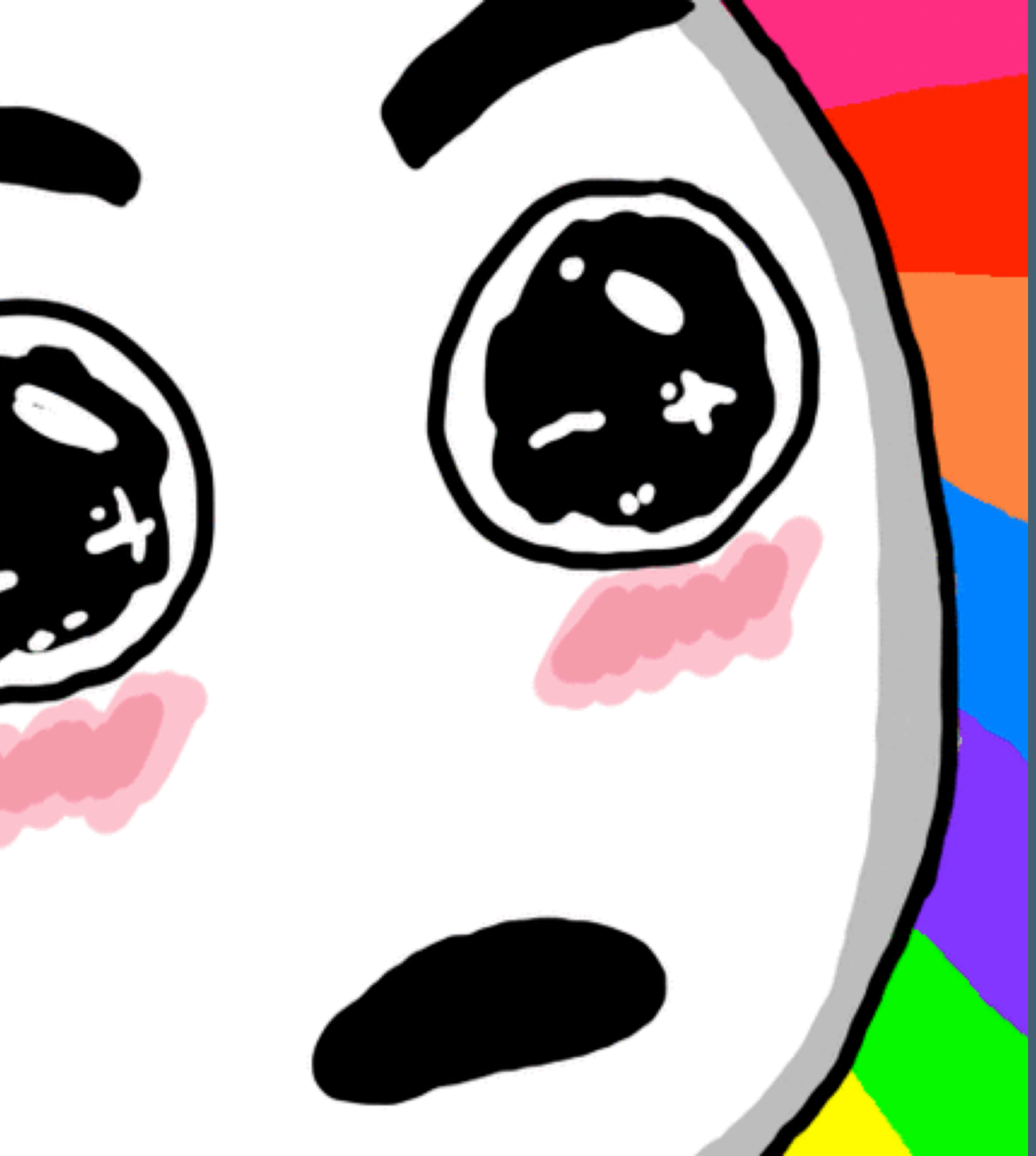
```
class RepoActivity : AppCompatActivity() {

  private val binding by activityBinding<ActivityRepoBinding>(R.layout.activity_repo)
  private val model by lazy { getViewModel<RepoViewModel>(RepoViewModelFactory()) }

  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)

    binding.setLifecycleOwner(this)
    binding.model = model
  }
}
```

ViewModel

```
class RepoViewModel(service: RepoService) : ViewModel {

  val items: ObservableField<List<String>> = ObservableField()

  fun query(value: String) {
    service.fetch(value, items::set)
  }
}
```

Observable

```
class RepoViewModel(service: RepoService) : ViewModel {

  val items: ObservableField<List<String>> = ObservableField()

  fun query(value: String) { /* ... */ }
}
```



LiveData Data Binding

LiveData Data Binding

```
class RepoViewModel(service: RepoService) : ViewModel {

    val items = MutableLiveData<List<String>>()

    fun query(value: String) { /* ... */ }
}
```

ProTip: Extension Functions! 👍

```
class RepoViewModel(service: RepoService) : ViewModel {

  val items = mutableLiveData<List<String>>()

  fun query(value: String) { /* ... */ }
}

fun <T> ViewModel.mutableLiveDataOf() = MutableLiveData<T>()
```

LiveData Data Binding

Transformations

LiveData Transformations

```
class RepoViewModel(private val service: RepoService) : ViewModel {

  val items = mutableLiveDataOf<List<String>>()
  val kotlin = map(items) {
    it.filter {
        it.contains("Kotlin")
    }
  }

  fun query(value: String) {
    service.fetch(value) {
      items.value = it
    }
  }
}
```

LiveData Data Binding

MediatorLiveData

MediatorLiveData Data Binding

```
class EmptyLiveData<in T>(items: LiveData<List<T>>, loading: LiveData<Boolean>) : MediatorLiveData<Boolean>() {

  init {
    addSource(items) { value = it?.isEmpty() ?: false && loading.value ?: false }
    addSource(loading) { value = it ?: false && items.value?.isEmpty() ?: false }
    value = true
  }
}
```

ViewModel

```
class RepoViewModel(private val service: RepoService) : ViewModel {

  val items = mutableLiveDataOf<List<String>>()

  fun query(value: String) {
    service.fetch(value) {
      items.value = it
    }
  }
}
```

ViewModel

```
class RepoViewModel(private val service: RepoService) : ViewModel {

  val items = mutableLiveDataOf<List<String>>()
  val loading = mutableLiveDataOf<Boolean>()

  fun query(value: String) {
    loading.value = true

    service.fetch(value) {
      loading.value = false
      items.value = it
    }
  }
}
```

Layout Expressions

```
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">

  <data>

    <import type="android.view.View"/>

    <variable
        name="model"
        type="io.ashdavies.databinding.RepoViewModel"/>

  </data>

  <ProgressBar...
      android:visibility="@{model.loading ? View.VISIBLE : View.GONE}"/>

</layout>
```

Layout Expressions

```
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">

  <data>

    <import type="android.view.View"/>

    <variable
        name="model"
        type="io.ashdavies.databinding.RepoViewModel"/>

  </data>

  <TextView...
      android:text="@string/activity_repo_empty"/>

  <ProgressBar...
      android:visibility="@{model.loading ? View.VISIBLE : View.GONE}"/>

</layout>
```

Layout Expressions

```
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">

  <data>

    <import type="android.view.View"/>

    <variable
        name="model"
        type="io.ashdavies.databinding.RepoViewModel"/>

  </data>

  <TextView...
      android:text="@string/activity_repo_empty"
      android:visibility="@{model.items.count == 0 ? View.VISIBLE : View.GONE}"/>

  <ProgressBar...
      android:visibility="@{model.loading ? View.VISIBLE : View.GONE}"/>

</layout>
```

Layout Expressions

```
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">

  <data>

    <import type="android.view.View"/>

    <variable
        name="model"
        type="io.ashdavies.databinding.RepoViewModel"/>

  </data>

  <TextView...
      android:text="@string/activity_repo_empty"
      android:visibility="@{model.items.count == 0 &amp;&amp; !model.loading ? View.VISIBLE : View.GONE}"/>

  <ProgressBar...
      android:visibility="@{model.loading ? View.VISIBLE : View.GONE}"/>

</layout>
```


Too Complicated!
1
ピタゴラ

Single Property Expressions

```
class RepoViewModel(private val service: RepoService) : ViewModel {

  val items = mutableLiveDataOf<List<String>>()
  val loading = mutableLiveDataOf<Boolean>()
  val empty = mutableLiveDataOf<Boolean>()

  fun query(value: String) {
    loading.value = true
    empty.value = true

    service.fetch(value) {
      items.value = it
      empty.value = it.isEmpty()
      loading.value = false
    }
  }
}
```

```
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">

  <data>

    <import type="android.view.View"/>

    <variable
        name="model"
        type="io.ashdavies.databinding.RepoViewModel"/>

  </data>

  <TextView...
      android:text="@string/activity_repo_empty"
      android:visibility="@{model.empty ? View.VISIBLE : View.GONE}"/>

  <ProgressBar...
      android:visibility="@{model.loading ? View.VISIBLE : View.GONE}"/>

</layout>
```

@BindingAdapter

@BindingAdapter

```
@BindingAdapter("goneUnless")
fun goneUnless(view: View, visible: Boolean) {
  view.visibility = if (visible) View.VISIBLE else View.GONE
}
```

ProTip: Extension Property! 👍

```
@set:BindingAdapter("isVisible")
inline var View.isVisible: Boolean
  get() = visibility == View.VISIBLE
  set(value) {
    visibility = if (value) View.VISIBLE else View.GONE
  }
```

@BindingAdapter

```
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">

  <data>

    <import type="android.view.View"/>

    <variable
        name="model"
        type="io.ashdavies.databinding.RepoViewModel"/>

  </data>

  <TextView...
      android:text="@string/activity_repo_empty"
      app:isVisible="@{model.empty}"/>

  <ProgressBar...
      app:isVisible="@{model.loading}"/>

</layout>
```

Simple Properties

Simple Properties

- `android:text="@{model.name}"`
  - `fun TextView.setText(value: String)`
  - `fun ViewModel.getName(): String`

Simple Properties

- `android:text="@{model.name}"`
  - `fun TextView.setText(@StringRes resId: Int)`
  - `@StringRes fun ViewModel.getName(): Int`

@BindingAdapter

Event Handling

Event Handling

```
@BindingAdapter("android:onQueryTextChange")
fun setOnQueryTextListener(view: SearchView, listener: OnQueryTextChange) {
  view.setOnQueryTextListener(object : SearchView.OnQueryTextListener {

    override fun onQueryTextSubmit(it: String): Boolean = true
    override fun onQueryTextChange(it: String): Boolean = listener(it)
  })
}

interface OnQueryTextChange {

  operator fun invoke(string: String): Boolean
}
```


POLDARK
BBC one

Event Handling

```
<androidx.appcompat.widget.SearchView
    android:id="@+id/search"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="end"
    android:onQueryTextChange="@{(value) -> model.onQuery(value)}">
```

Event Handling

```
<androidx.appcompat.widget.SearchView
    android:id="@+id/search"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="end"
    android:onQueryTextChange="@{model::onQuery}">
```


SENORGIF.COM

@BindingMethods

@BindingMethods

```
@BindingMethods({
   @BindingMethod(
         type = TextView.class,
         attribute = "android:inputType",
         method = "setRawInputType"
   )
})
public class TextViewBindingAdapter { /* ... */ }
```

@Bindable

@Bindable

```
public class BR {
  public static final int _all = 0;

  public static final int item = 1;

  public static final int model = 2;

  public static final int empty = 3;
}
```

@Bindable

ObservableViewModel

@Bindable

PropertyChangeRegistry

@Bindable

```
abstract class ObservableViewModel : ViewModel(), Observable {

  private val callbacks: PropertyChangeRegistry = PropertyChangeRegistry()

  override fun addOnPropertyChangedCallback(callback: Observable.OnPropertyChangedCallback) {
    callbacks.add(callback)
  }

  override fun removeOnPropertyChangedCallback(callback: Observable.OnPropertyChangedCallback) {
    callbacks.remove(callback)
  }

 fun notifyChange() {
    callbacks.notifyCallbacks(this, 0, null)
  }

 fun notifyPropertyChanged(fieldId: Int) {
    callbacks.notifyCallbacks(this, fieldId, null)
  }
}
```

@Bindable

```
class RepoViewModel(private val service: RepoService) : ObservableViewModel {

  val items = mutableLiveDataOf<List<String>>()
  val loading = mutableLiveDataOf<Boolean>()

  fun query(value: String) {
    loading.value = true

    service.fetch(value) {
      loading.value = false
      items.value = it
    }
  }
}
```

@Bindable

```
class RepoViewModel(private val service: RepoService) : ObservableViewModel {

  @get:Bindable
  var items: List<String> = emptyList()
    private set(value) {
      notifyPropertyChanged(BR.items)
      field = value
    }

  val loading = mutableLiveDataOf<Boolean>()

  fun query(value: String) {
    loading.value = true

    service.fetch(value) {
      loading.value = false
      items = it
    }
  }
}
```

@Bindable

Delegated Properties

Property Delegate

ObservableProperty

@Bindable

```
class BindableProperty<T>(
  initial: T, private val observable: ObservableViewModel, private val id: Int
) : ObservableProperty<T>(initial) {

  override fun beforeChange(
    property: KProperty<*>, oldValue: T, newValue: T
  ): Boolean = { /* ... */ }

  override fun afterChange(
    property: KProperty<*>, oldValue: T, newValue: T
  ) { /* ... */ }
}
```

@Bindable

```
class BindableProperty<T>(
  initial: T, private val observable: ObservableViewModel, private val id: Int
) : ObservableProperty<T>(initial) {

  override fun beforeChange(
    property: KProperty<*>, oldValue: T, newValue: T
  ): Boolean = oldValue != newValue

  override fun afterChange(
    property: KProperty<*>, oldValue: T, newValue: T
  ) { /* ... */ }
}
```

@Bindable

```
class BindableProperty<T>(
  initial: T, private val observable: ObservableViewModel, private val id: Int
) : ObservableProperty<T>(initial) {

  override fun beforeChange(
    property: KProperty<*>, oldValue: T, newValue: T
  ): Boolean = oldValue != newValue

  override fun afterChange(property: KProperty<*>, oldValue: T, newValue: T) {
    observable.notifyPropertyChanged(id)
  }
}
```

ProTip: Extension Function! 👍

@Bindable

```
class RepoViewModel(private val service: RepoService) : ObservableViewModel() {

  @get:Bindable
  var items by bindable<List<String>>(emptyList(), BR.items)
    private set

  val items = mutableLifeDataOf<List<String>>()

  init {
    service.fetch(items::setValue)
  }
}

fun <T> ObservableViewModel.bindable(initial: T, id: Int): BindableProperty<T> {
  return BindableProperty(initial, this, id)
}
```

🎉 Awesome! 🎉

Data Binding

Generated Code

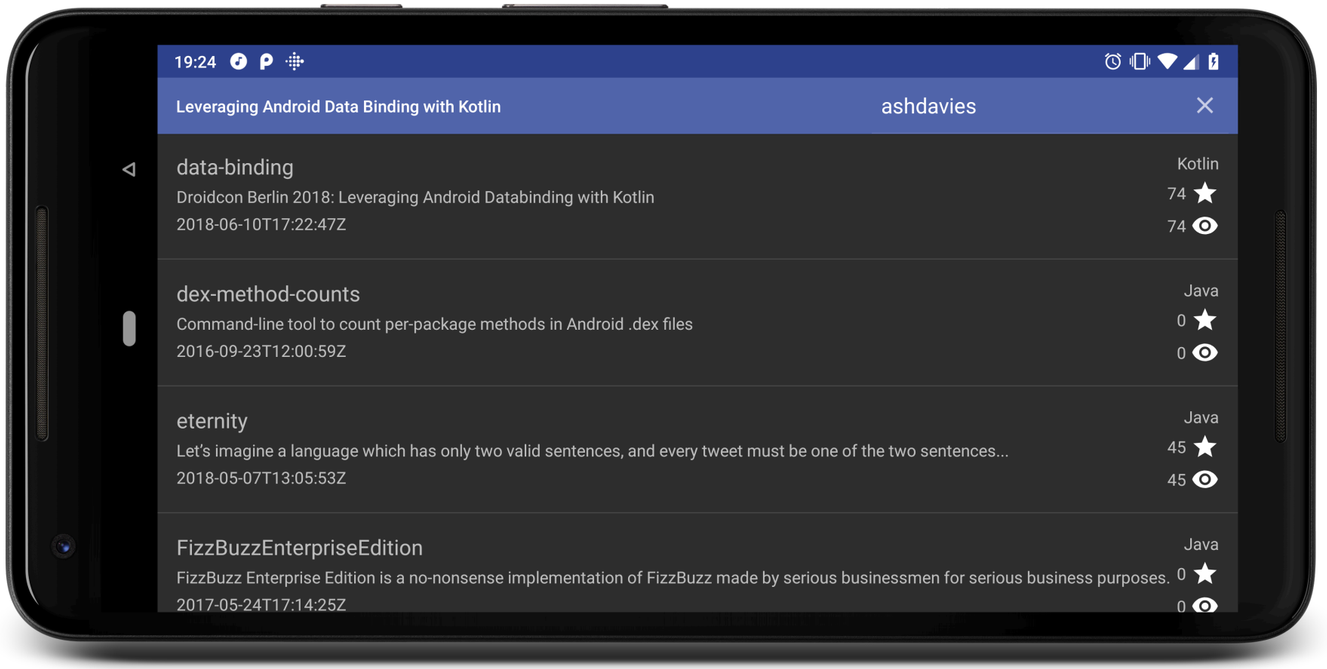
```
public class ActivityRepoBindingImpl extends ActivityRepoBinding  {

    @Override protected void executeBindings() {
        long dirtyFlags = 0;
        synchronized(this) {
            dirtyFlags = mDirtyFlags;
            mDirtyFlags = 0;
        }
        java.lang.Integer modelCountGetValue = null;
        io.ashdavies.databinding.databinding.OnQueryTextChange modelOnQueryIoAshdaviesDatabindingDatabindingOnQueryTextChange = null;
        boolean modelLoading = false;
        boolean modelEmpty = false;
        io.ashdavies.databinding.repos.RepoViewModel model = mModel;
        java.lang.String countAndroidPluralsRepoItemsCountModelCountModelCount = null;
        io.ashdavies.databinding.extensions.Visibility modelVisibilityGetValue = null;
        androidx.lifecycle.LiveData<java.lang.Integer> modelCount = null;
        androidx.lifecycle.MutableLiveData<java.lang.Boolean> ModelLoading1 = null;
        java.lang.Boolean modelLoadingGetValue = null;
        androidx.lifecycle.LiveData<io.ashdavies.databinding.extensions.Visibility> modelVisibility = null;
        boolean androidxDatabindingViewDataBindingSafeUnboxModelLoadingGetValue = false;
        boolean androidxDatabindingViewDataBindingSafeUnboxModelLoading = false;

        if ((dirtyFlags & 0x3fL) != 0) {
            if ((dirtyFlags & 0x21L) != 0) {
                    if (model != null) {
                        modelOnQueryIoAshdaviesDatabindingDatabindingOnQueryTextChange = (((mModelOnQueryIoAshdaviesDatabindingDatabindingOnQueryTextChange == null) ? (mModelOnQueryIoAshdaviesDatabindingDatabindingOnQueryTextChange = new OnQueryTextChangeImpl()) : mModelOnQueryIoAshdaviesDatabindingDatabindingOnQueryTextChange).setValue(model));
                    }
            }
            if ((dirtyFlags & 0x31L) != 0) {
                    if (model != null) {
                        modelEmpty = model.getEmpty();
                    }
            }
            if ((dirtyFlags & 0x23L) != 0) {
                    if (model != null) {
                        modelCount = model.getCount();
                    }
                    updateLiveDataRegistration(1, modelCount);
                    if (modelCount != null) {
                        modelCountGetValue = modelCount.getValue();
                    }
                    countAndroidPluralsRepoItemsCountModelCountModelCount = count.getResources().getQuantityString(R.plurals.repo_items_count, modelCountGetValue, modelCountGetValue);
                    countAndroidPluralsRepoItemsCountModelCountModelCount = count.getResources().getQuantityString(R.plurals.repo_items_count, modelCountGetValue, modelCountGetValue);
            }
            if ((dirtyFlags & 0x25L) != 0) {
                    if (model != null) {
                        ModelLoading1 = model.getLoading();
                    }
                    updateLiveDataRegistration(2, ModelLoading1);
                    if (ModelLoading1 != null) {
                        modelLoadingGetValue = ModelLoading1.getValue();
                    }
                    androidxDatabindingViewDataBindingSafeUnboxModelLoadingGetValue = androidx.databinding.ViewDataBinding.safeUnbox(modelLoadingGetValue);
                    modelLoading = !androidxDatabindingViewDataBindingSafeUnboxModelLoadingGetValue;
                    androidxDatabindingViewDataBindingSafeUnboxModelLoading = androidx.databinding.ViewDataBinding.safeUnbox(modelLoading);
            }
            if ((dirtyFlags & 0x29L) != 0) {
                    if (model != null) {
                        modelVisibility = model.getVisibility();
                    }
                    updateLiveDataRegistration(3, modelVisibility);
                    if (modelVisibility != null) {
                        modelVisibilityGetValue = modelVisibility.getValue();
                    }
            }
        }
        if ((dirtyFlags & 0x23L) != 0) {
            androidx.databinding.adapters.TextViewBindingAdapter.setText(this.count, countAndroidPluralsRepoItemsCountModelCountModelCount);
        }
        if ((dirtyFlags & 0x31L) != 0) {
            io.ashdavies.databinding.extensions.ViewKt.goneUnless(this.empty, modelEmpty);
        }
        if ((dirtyFlags & 0x25L) != 0) {
            io.ashdavies.databinding.extensions.ViewKt.goneUnless(this.mboundView4, androidxDatabindingViewDataBindingSafeUnboxModelLoadingGetValue);
            io.ashdavies.databinding.extensions.ViewKt.goneUnless(this.recycler, androidxDatabindingViewDataBindingSafeUnboxModelLoading);
        }
        if ((dirtyFlags & 0x29L) != 0) {
            io.ashdavies.databinding.extensions.ViewKt.setVisibility(this.recycler, modelVisibilityGetValue);
        }
        if ((dirtyFlags & 0x21L) != 0) {
            io.ashdavies.databinding.databinding.SearchViewBindingsKt.setOnQueryTextListener(this.search, modelOnQueryIoAshdaviesDatabindingDatabindingOnQueryTextChange);
        }
    }
}
```

```java
public class ActivityRepoBindingImpl extends ActivityRepoBinding  {

    @Override protected void executeBindings() {
        if ((dirtyFlags & 0x3fL) != 0) {
            if ((dirtyFlags & 0x21L) != 0) {
                    if (model != null) {
                        modelOnQueryIoAshdaviesDatabindingDatabindingOnQueryTextChange = (((mModelOnQueryIoAshdaviesDatabindingDatabindingOnQueryTextChange == null) ?
                            (mModelOnQueryIoAshdaviesDatabindingDatabindingOnQueryTextChange = new OnQueryTextChangeImpl())
                            : mModelOnQueryIoAshdaviesDatabindingDatabindingOnQueryTextChange).setValue(model));
                    }
            }
            if ((dirtyFlags & 0x31L) != 0) {
                    if (model != null) {
                        modelEmpty = model.getEmpty();
                    }
            }
            if ((dirtyFlags & 0x25L) != 0) {
                    if (model != null) {
                        ModelLoading1 = model.getLoading();
                    }
                    updateLiveDataRegistration(2, ModelLoading1);
                    if (ModelLoading1 != null) {
                        modelLoadingGetValue = ModelLoading1.getValue();
                    }
                    androidxDatabindingViewDataBindingSafeUnboxModelLoadingGetValue = androidx.databinding.ViewDataBinding.safeUnbox(modelLoadingGetValue);
                    modelLoading = !androidxDatabindingViewDataBindingSafeUnboxModelLoadingGetValue;
                    androidxDatabindingViewDataBindingSafeUnboxModelLoading = androidx.databinding.ViewDataBinding.safeUnbox(modelLoading);
            }
        if ((dirtyFlags & 0x31L) != 0) {
            io.ashdavies.databinding.extensions.ViewKt.goneUnless(this.empty, modelEmpty);
        }
        if ((dirtyFlags & 0x25L) != 0) {
            io.ashdavies.databinding.extensions.ViewKt.goneUnless(this.mboundView4, androidxDatabindingViewDataBindingSafeUnboxModelLoadingGetValue);
            io.ashdavies.databinding.extensions.ViewKt.goneUnless(this.recycler, androidxDatabindingViewDataBindingSafeUnboxModelLoading);
        }
        if ((dirtyFlags & 0x21L) != 0) {
            io.ashdavies.databinding.databinding.SearchViewBindingsKt.setOnQueryTextListener(this.search, modelOnQueryIoAshdaviesDatabindingDatabindingOnQueryTextChange);
        }
    }
}
```

```java
public abstract class ActivityRepoBinding extends ViewDataBinding {

  @NonNull public final CoordinatorLayout coordinator;
  @NonNull public final TextView count;
  @NonNull public final TextView empty;
  @NonNull public final RecyclerView recycler;
  @NonNull public final SearchView search;
  @NonNull public final Toolbar toolbar;
  @Bindable protected RepoViewModel mModel;

  protected ActivityRepoBinding(DataBindingComponent _bindingComponent, View _root, int _localFieldCount, CoordinatorLayout coordinator,
        TextView count, TextView empty, RecyclerView recycler, SearchView search, Toolbar toolbar) {
    super(_bindingComponent, _root, _localFieldCount);
    /* ... */
  }

  public abstract void setModel(@Nullable RepoViewModel model);

  @Nullable public RepoViewModel getModel() { /* ... */ }

  @NonNull public static ActivityRepoBinding inflate(@NonNull LayoutInflater inflater, @Nullable ViewGroup root, boolean attachToRoot) { /* ... */ }

  @NonNull public static ActivityRepoBinding inflate(@NonNull LayoutInflater inflater, @Nullable ViewGroup root, boolean attachToRoot,
      @Nullable DataBindingComponent component) { /* ... */ }

  @NonNull public static ActivityRepoBinding inflate(@NonNull LayoutInflater inflater) { /* ... */ }

  @NonNull public static ActivityRepoBinding inflate(@NonNull LayoutInflater inflater, @Nullable DataBindingComponent component) { /* ... */ }

  public static ActivityRepoBinding bind(@NonNull View view) { /* ... */ }

  public static ActivityRepoBinding bind(@NonNull View view, @Nullable DataBindingComponent component) { /* ... */ }
}
```

Data Binding Sample

github.com/ashdavies/data-binding



Android Data Binding with Kotlin

bit.ly/2yU8qUz

Additional Resources

- **Android Data Binding Library samples**
  Google Samples - [bit.ly/2MK5GMb]

- **Make your view-model properties great again**
  Aiden McWilliams - [bit.ly/2llVVHz]

- **MVVM, Viewmodel and architecture components**
  Danny Preussler - [bit.ly/2yxvGaY]

- **Android Data Binding: RecyclerView**
  George Mount - [https://bit.ly/2IgwY9w]

Cheers! 

_ashdavies