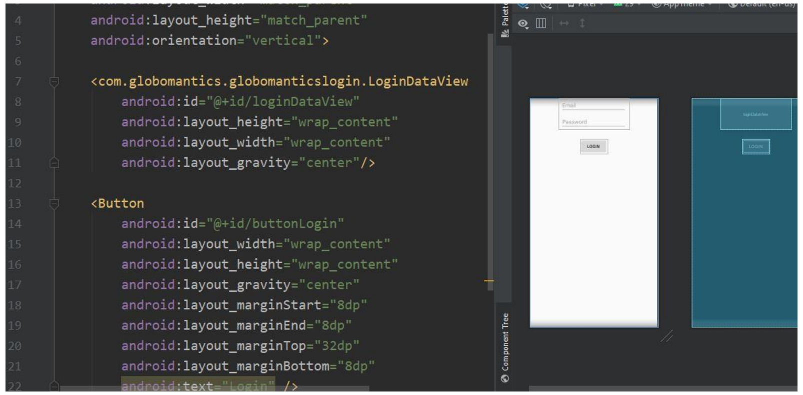
Why Do You Need View Binding?

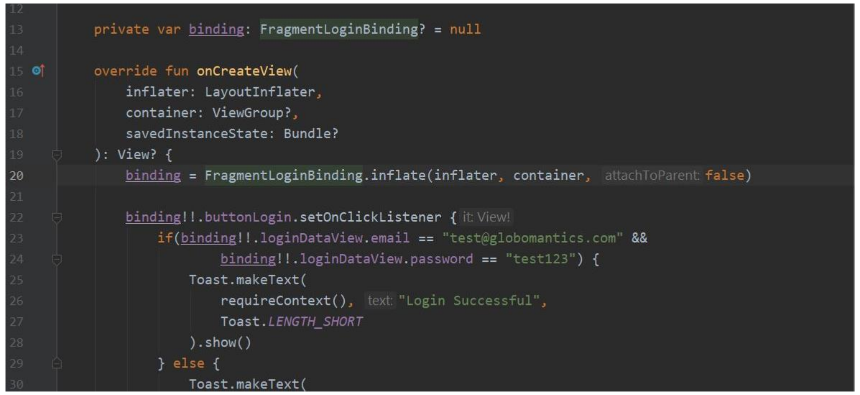


```
    android:layout_height="match_parent"
    android:orientation="vertical">

    <com.globomantics.globomanticslogin.LoginDataView
        android:id="@+id/loginDataView"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:layout_gravity="center"/>

    <Button
        android:id="@+id/buttonLogin"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginStart="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginTop="32dp"
        android:layout_marginBottom="8dp"
        android:text="Login" />
```

Binding



```
    private var binding: FragmentLoginBinding? = null

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        binding = FragmentLoginBinding.inflate(inflater, container, attachToParent: false)

        binding!!.buttonLogin.setOnClickListener { it: View!
            if(binding!!.loginDataView.email == "test@globomantics.com" &&
                    binding!!.loginDataView.password == "test123") {
                Toast.makeText(
                    requireContext(), text: "Login Successful",
                    Toast.LENGTH_SHORT
                ).show()
            } else {
                Toast.makeText(
```

Layout Design

XML files

App Logic

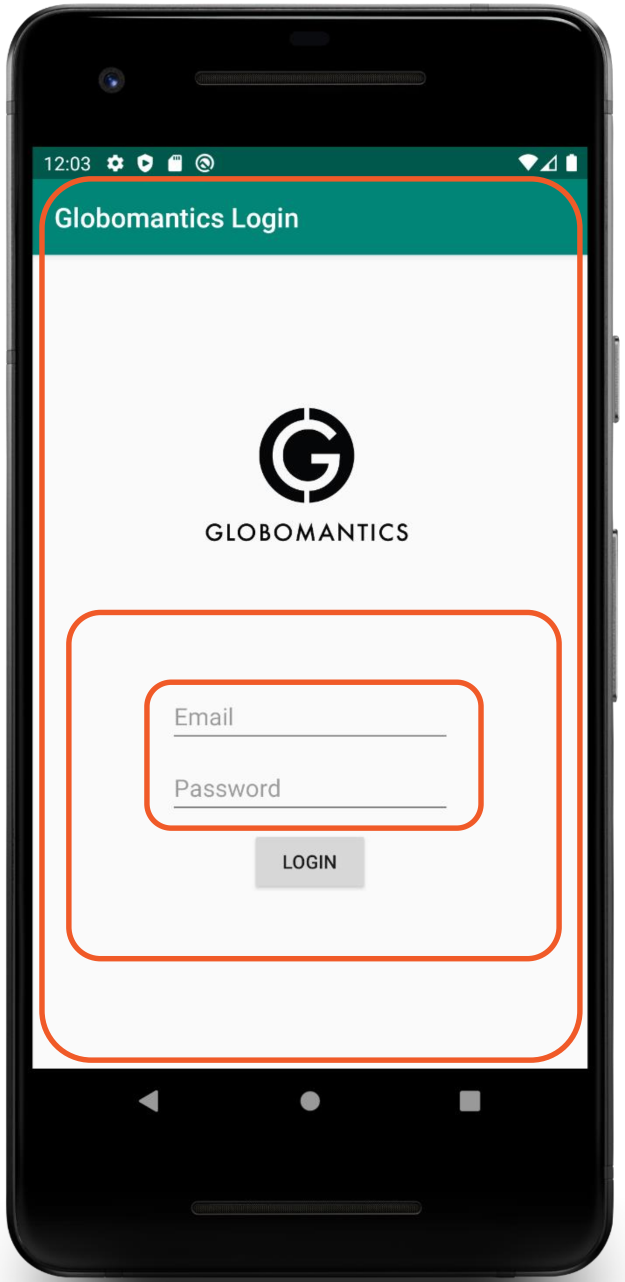
Kotlin and Java files

findViewById

- Forces you to explicitly create a reference to every view
- Forces you to write a lot of overhead code

View Binding library

- Replacement for `findViewById`
- It generates all overhead code for you
- Takes care of nullability and is type-safe



Working on demo project

Demo project should not be used as an example of how an Android app's architecture should look like

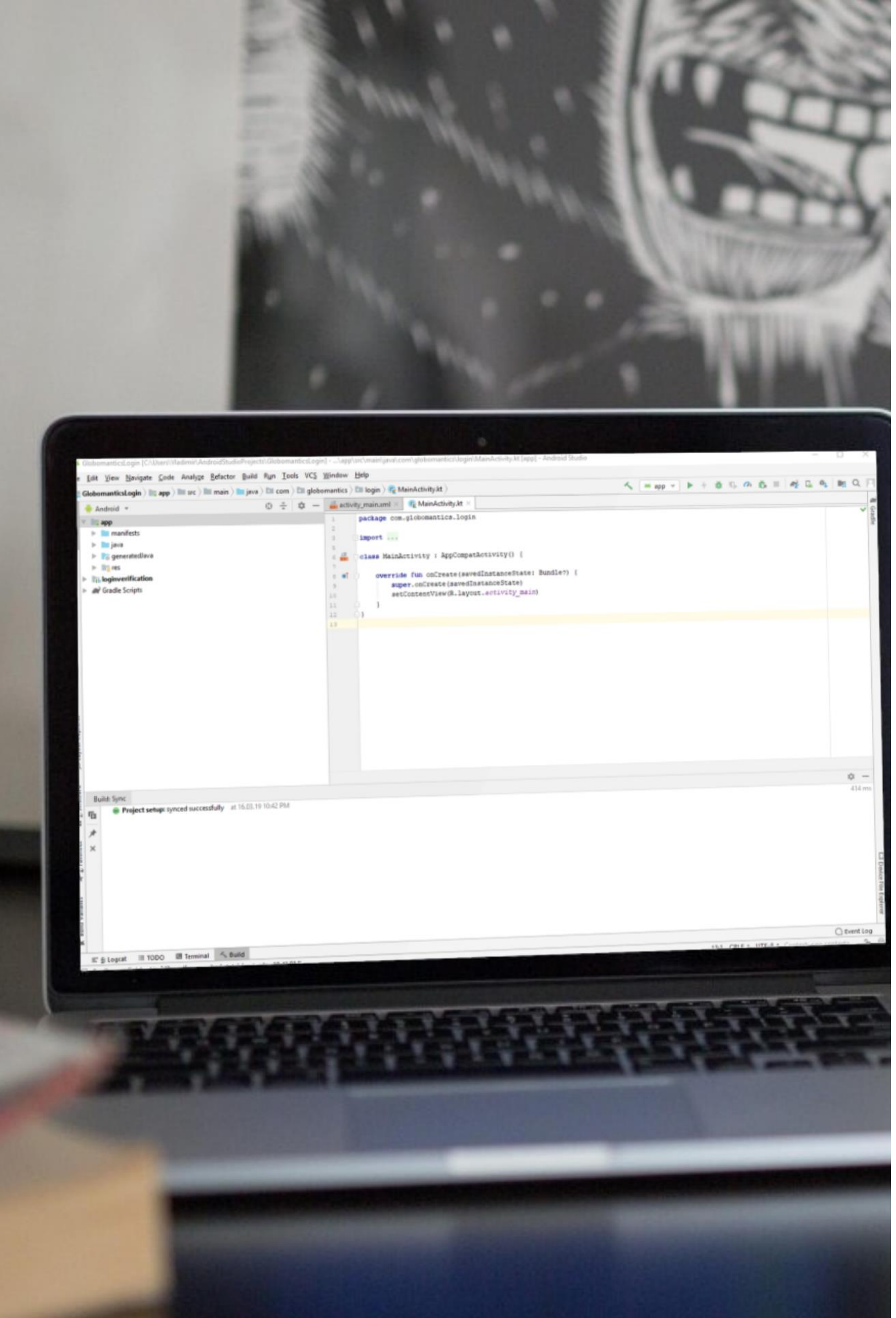
Create an Android project

Enable View Binding library

`LoginActivity`

`LoginFragment`

`LoginDataView`



```
package com.globomantics.login

import ...

class MainActivity : AppCompatActivity() {

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

Android Studio IDE

- Download and install before starting with the next clip
- “Getting Started with Android Studio” course

Good for solving some problems, but not all of them

You may consider using other view binding libraries

Limitations of View Binding

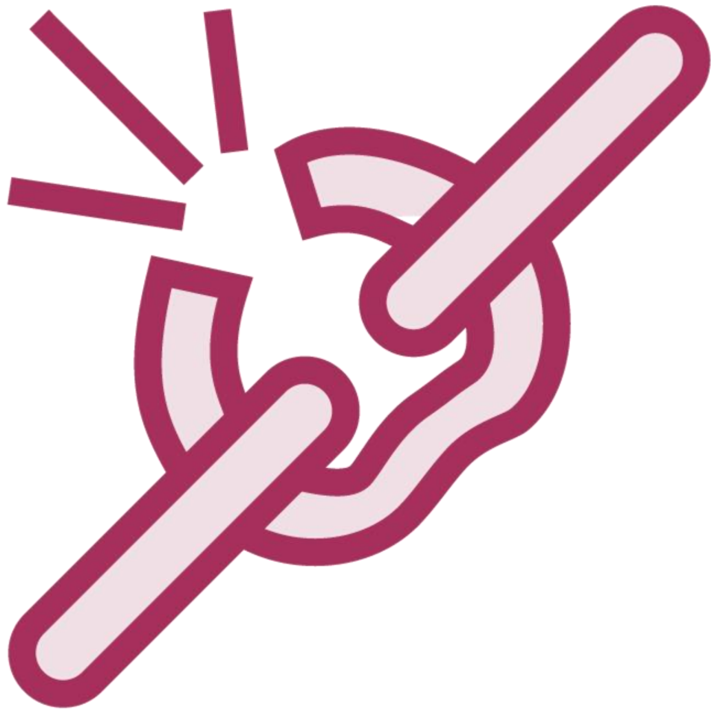
- Problems that can’t be solved using View Binding

Data Binding library

Kotlin Synthetics and Butterknife

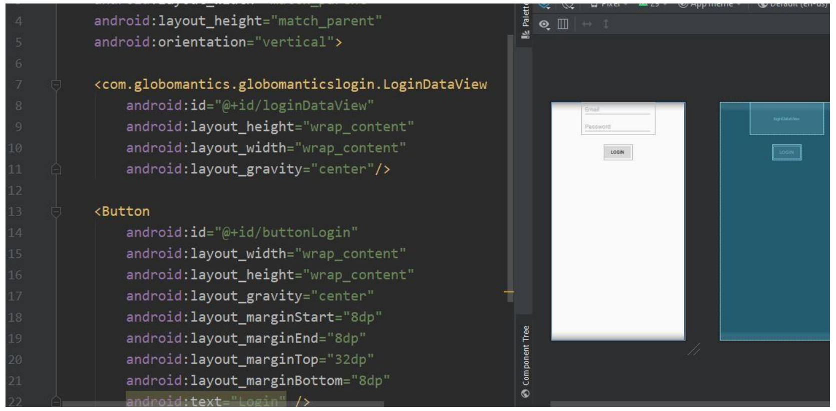
- Check how they can be replaced by View Binding

Limitations of View Binding



It's not a perfect solution that will cover all your needs

View Binding works in only one direction



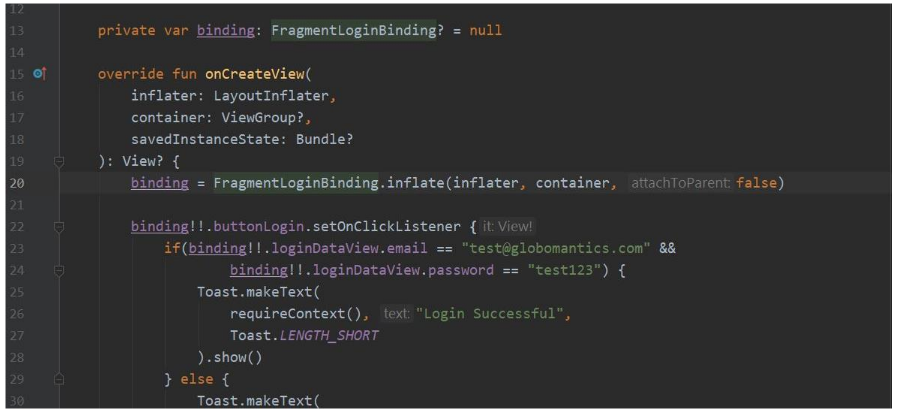
```xml
    android:layout_height="match_parent"
    android:orientation="vertical">

    <com.globomantics.globomanticslogin.LoginDataView
        android:id="@+id/loginDataView"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:layout_gravity="center"/>

    <Button
        android:id="@+id/buttonLogin"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginStart="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginTop="32dp"
        android:layout_marginBottom="8dp"
        android:text="Login" />
```

View Binding

Binding



```kotlin
    private var binding: FragmentLoginBinding? = null

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        binding = FragmentLoginBinding.inflate(inflater, container, attachToParent: false)

        binding!!.buttonLogin.setOnClickListener { it: View!
            if(binding!!.loginDataView.email == "test@globomantics.com" &&
                    binding!!.loginDataView.password == "test123") {
                Toast.makeText(
                    requireContext(), text: "Login Successful",
                    Toast.LENGTH_SHORT
                ).show()
            } else {
                Toast.makeText(
```

Layout Design

App Logic

Limitations of View Binding

```
<CheckBox
   android:id="@+id/agreeToTerms"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content" />
```

Limitations of View Binding

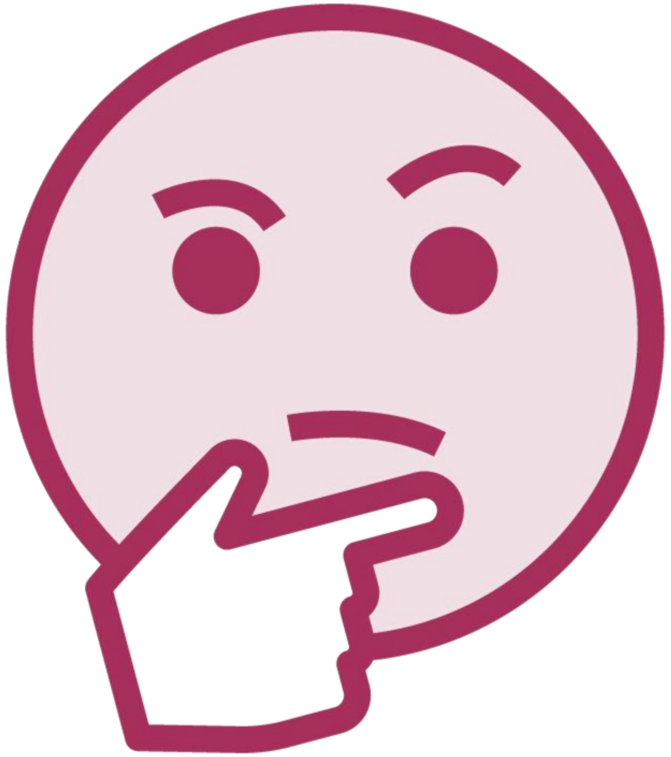
```
<CheckBox
    android:id="@+id/agreeToTerms"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="viewModel.doSomething()" />
```

Limitations of View Binding

```
<CheckBox
    android:id="@+id/agreeToTerms"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="viewModel.doSomething()"
    android:text="viewModel.terms" />
```

Limitations of View Binding

```xml
<CheckBox
    android:id="@+id/agreeToTerms"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="viewModel.doSomething()"
    android:text="viewModel.terms"
    android:checked="viewModel.isChecked" />
```



All those features that View Binding lacks sound really cool.

Is there anything that could make up for that?

Data Binding

Data Binding

```xml
<CheckBox
    android:id="@+id/agreeToTerms"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

Data Binding

```
<layout>
   <CheckBox

      android:id="@+id/agreeToTerms"

      android:layout_width="wrap_content"

      android:layout_height="wrap_content" />

</layout>
```

Data Binding

```
<layout>
    <data>
        <variable name="viewModel" type="MyViewModel"/>
    <data>
    <CheckBox
        android:id="@+id/agreeToTerms"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</layout>
```

Data Binding

```
<layout>

    <data>

        <variable name="viewModel" type="MyViewModel"/>

    <data>
    <CheckBox

        android:id="@+id/agreeToTerms"

        android:layout_width="wrap_content"

        android:layout_height="wrap_content" />

</layout>
```

Data Binding

```
<layout>

    <data>

        <variable name="viewModel" type="MyViewModel"/>

    <data>
    <CheckBox

        android:id="@+id/agreeToTerms"

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:onClick="@{() -> viewModel.doSomething()}" />

</layout>
```

Data Binding

```
<layout>

    <data>

        <variable name="viewModel" type="MyViewModel"/>

    <data>
    <CheckBox

        android:id="@+id/agreeToTerms"

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:text="@{viewModel.terms}" />

</layout>
```

Data Binding

```
<layout>

    <data>

        <variable name="viewModel" type="MyViewModel"/>

    <data>
    <CheckBox

        android:id="@+id/agreeToTerms"

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:checked="@={viewModel.isChecked}" />

</layout>
```

Why Do I Even Need View Binding?

Data Binding requires annotation processing

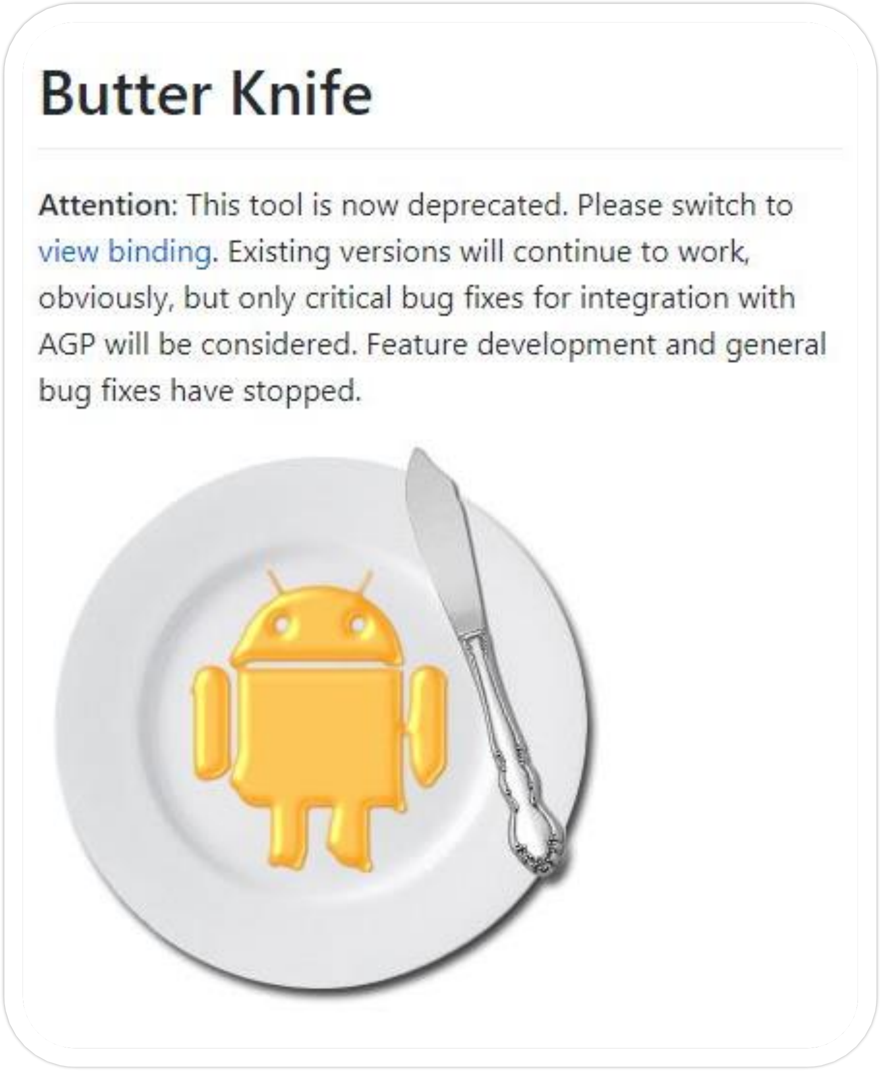
Highly complex library

- It requires specially-tagged XML layout files
- Layout files can become cluttered with logic

It's best to use View Binding and Data Binding together

If a layout file uses Data Binding, the View Binding library won't generate bindings

Butterknife



Butter Knife

Attention: This tool is now deprecated. Please switch to view binding. Existing versions will continue to work, obviously, but only critical bug fixes for integration with AGP will be considered. Feature development and general bug fixes have stopped.

Developed by Jake Wharton and released in 2013

Deprecated in 2020

If your app is using Butterknife, it's time to move to a better solution

Migration from Butterknife to View Binding

Simple Case

ActivityWithButterknife.kt

```
@BindView(R.id.view)
lateinit var view1 : View

@Nullable @BindView(R.id.view)
lateinit var view2 : View?

override fun onCreate(…) {
    …
    setContentView(R.layout.activity)
    Butterknife.bind(this)
    view1.doSomething()
    view2?.doSomethingElse()

}
```

ActivityWithViewBinding.kt

```
private lateinit var binding:
ActivityBinding

override fun onCreate(…) {
    …
    binding =
    ActivityBinding.inflate(inflater)
    setContentView(binding.root)
    binding.view1.doSomething()
    binding.view2?.doSomethingElse()

}
```

Migration from Butterknife to View Binding

Unsupported Resource Binding

ActivityWithButterknife.kt

```
@BindString(R.string.text)
lateinit var text : String

@BindDrawable(R.drawable.image)
lateinit var image : Drawable

@BindColor(R.color.color)
lateinit var color : Int

@BindDimen(R.dimen.padding)
lateinit var padding : Float
```

ActivityWithViewBinding.kt

```
val title =
getString(R.string.text)

val image =
getDrawable(R.string.drawable)

val color =
getColor(R.color.color)

val padding =
resources.getDimension(R.dimen.padding)
```