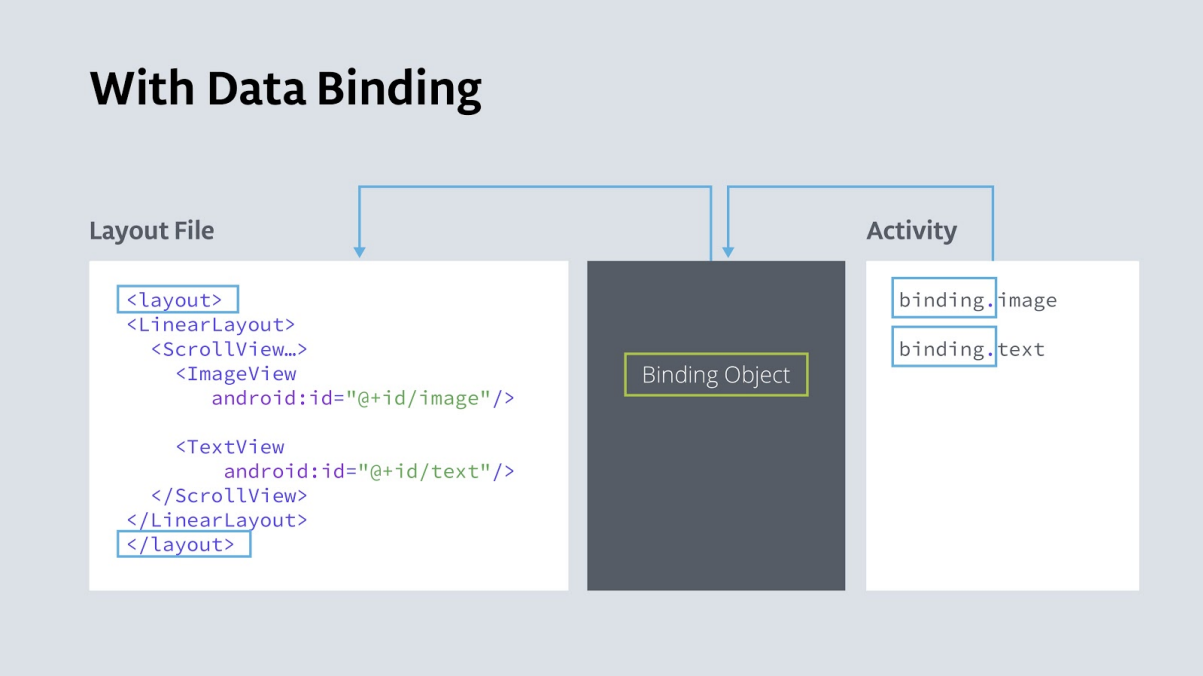
Data binding

`findViewById()` → may slow down the app

Solution: create an object that contains a reference to each view. This object, called `Biding` object, can be used by your whole app. → Data biding.



Data biding benefits

- Code is shorter, easier to read, and easier to maintain than code that uses `findViewById()`.
- Data and views are clearly separated.
- The Android system only traverses the view hierarchy once to get each view, and it happens during app startup, not at runtime when the user is interacting with the app.
- You get type safety for accessing views.

1. Enable data bidings

▼ Open the `build.gradle (Module: app)` file.

```
buildFeatures {
    dataBinding true
}
```

2. Change layout file to be usable with data binding

▼ Add `<layout></layout>` as the outermost tag around the layout XML file.

```
<layout>
   <LinearLayout ... >
   ...
   </LinearLayout>
</layout>
```

▼ Cut the namespace declarations from the `<LinearLayout>` and paste them into the `<layout>` tag.

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

3. Create a binding object in the main activity

▼ Before `onCreate()`, at the top level, create a variable for the binding object. This variable is customarily called `binding`.

The type of `binding`, the `ActivityMainBinding` class, is created by the compiler specifically for this main activity. The name is derived from the name of the layout file, that is, `activity_main + Binding`.

```
private lateinit var binding: ActivityMainBinding
```

▼ Inside `onCreate()`, replace the current `setContentView()` function with an instruction that does the following:

- Creates the binding object.
- Uses the `setContentView()` function from the `DataBindingUtil` class to associate the `activity_main` layout with the `MainActivity`.
  This `setContentView()` function also takes care of some data binding setup for the views.

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

4. Use the binding object to replace all calls to `findViewById()`

```
binding.apply {
   nicknameText.text = nicknameEdit.text.toString()
   nicknameEdit.visibility = View.GONE
   doneButton.visibility = View.GONE
   nicknameText.visibility = View.VISIBLE
}
```

Use data binding to display data

- For example, we have a data class

```
data class MyName(var name: String = "", var nickname: String = "")
```

- Add data to the layout
  - Inside `<layout></layout>` , declair variable

```
<data>
        <variable
            name="myName"
            type="com.example.android.aboutme.MyName" />
    </data>
```

Now, instead of using the string resource for the name, you can reference the `myName` variable.

For example, replace `android:text="@string/name"` to `android:text="@={myName.name}"`

Summary

Steps to use data binding to replace calls to `findViewById()`:

1. Enable data binding in the android section of the `build.gradle` file.

2. Use `<layout>` as the root view in your XML layout.
3. Define a binding variable: `private lateinit var binding: ActivityMainBinding`
4. Create a binding object in `MainActivity`, replacing `setContentView`: `binding = DataBindingUtil.setContentView(this, R.layout.activity_main)`
5. Replace calls to `findViewById()` with references to the view in the binding object. For example: `findViewById<Button>(R.id.done_button)` ⇒ `binding.doneButton` (In the example, the name of the view is generated camel case from the view's `id` in the XML.)

Steps for binding views to data:

1. Create a data class for your data.
2. Add a `<data>` block inside the `<layout>` tag.
3. Define a `<variable>` with a name, and a type that is the data class.
4. In `MainActivity`, create a variable with an instance of the data class. For example: `private val myName: MyName = MyName("Aleks Haecky")`
5. In the binding object, set the variable to the variable you just created: `binding.myName = myName`
6. In the XML, set the content of the view to the variable that you defined in the `<data>` block. Use dot notation to access the data inside the data class. `android:text="@={myName.name}"`