Dependencies

This lesson introduces commonly used Android libraries.

WE'LL COVER THE FOLLOWING

The Android library ecosystem is huge, and most projects use dozens of libraries. Unlike Android SDK, libraries can be independently updated. That's why even Google released around 30 libraries which are part of Android Jetpack family.

Most of the Android libraries are available through maven. To add the library, declare the group id, artifact id, and version in the *app/build.gradle* file in the `dependencies` section.

```
dependencies {
    // ui
    implementation 'androidx.appcompat:appcompat:1.1.0'
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
    implementation 'com.google.android.material:material:1.1.0-alpha10'
}
```

build.gradle

Here is the list of most commonly used libraries:

- *appcompat* - makes the apps developed with newer versions work with older versions
- *constraintlayout* - allows creating large and complex layouts with a flat view hierarchy
- *material* - brings material design components to Android

- *retrofit* - a type-safe HTTP client library

- *moshi* - a JSON parser library
- *glide* - an image loading library
- *room* - an official Android ORM database
- *dagger* - a static, compile-time dependency injection framework

Through this course, we will use most of the libraries listed above, but for now, let's focus on the first three.

Appcompat

The *appcompat* library makes the apps developed with newer versions work with older versions.

```
implementation 'androidx.appcompat:appcompat:1.1.0'
```

The main component of this library is `AppCompatActivity`, which is a base class for activities. This class provides some backward compatibility with old Android versions. All of our activities are going to extend `AppCompatActivity`.

```
package com.travelblog;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {
      ...
}
```

MainActivity.java

Constraint layout

The *constraintlayout* library allows creating large and complex layouts with a flat view hierarchy. It came as a replacement for old layout components, which are still useful if view hierarchy is simple.

```
implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
```

In most cases, we will use `ConstraintLayout` as a root layout in all our layout files. We will discuss `ConstraintLayout` in more detail in the next lessons.

```
<androidx.constraintlayout.widget.ConstraintLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        android:layout_width="match_parent"
        android:layout_height="match_parent">
    ...
</androidx.constraintlayout.widget.ConstraintLayout>
```

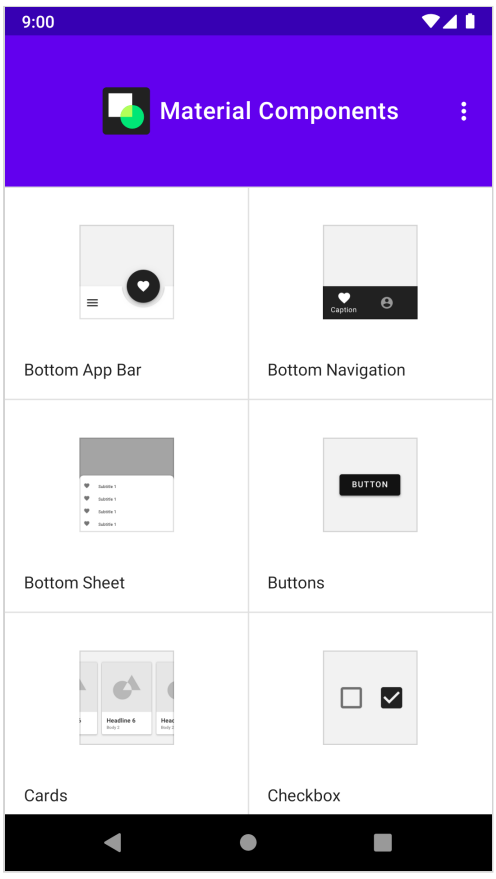
Material design

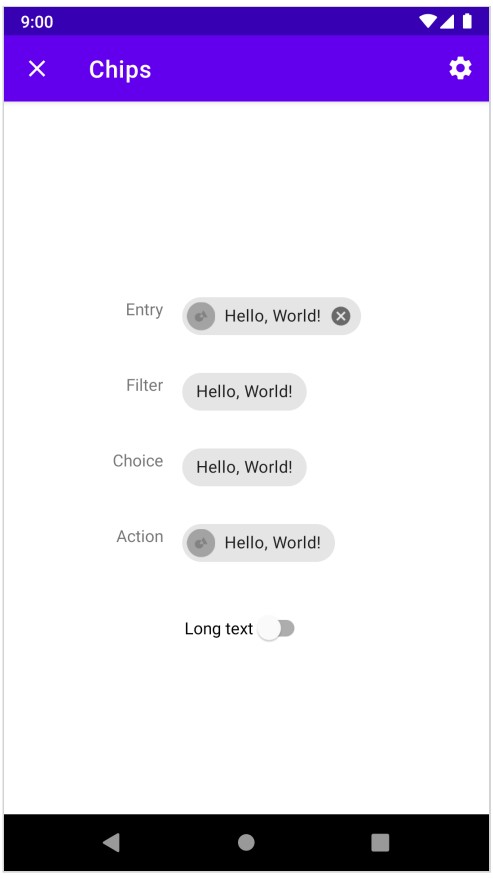
Material design is a design language created by Google, which is used to create user-friendly components. The *material* library brings Material Design components to Android.

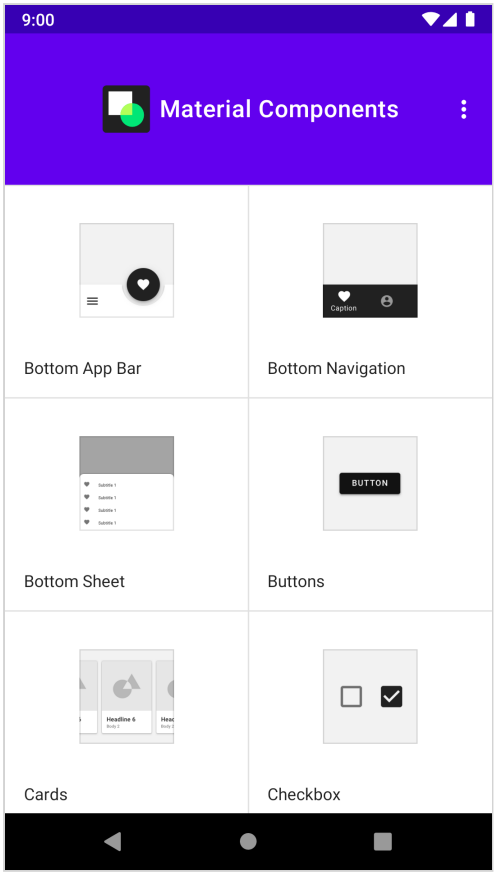
```
implementation 'com.google.android.material:material:1.1.0-alpha10'
```

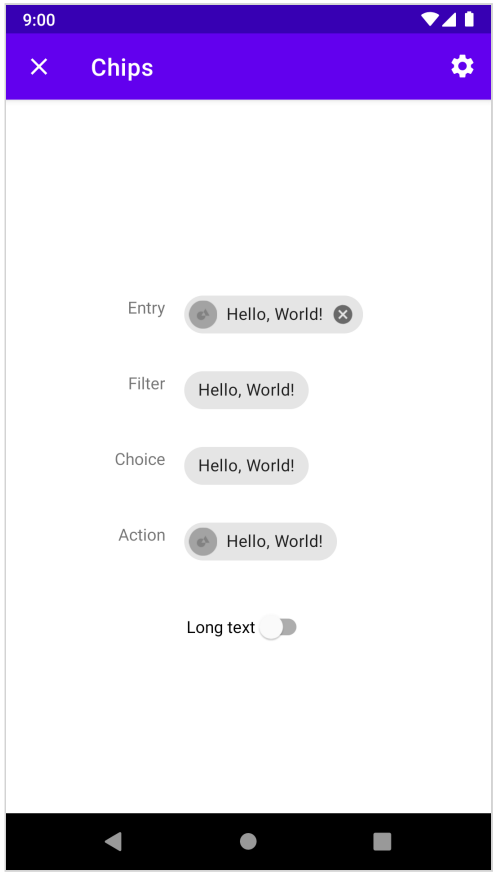
The official Material Design site provides a list of available components and tutorials of how to use them.

In addition to the official material design site, there is a Catalog App which demonstrates how Material Design components and principles behave on real devices.









Through this course, we are going to use some of the material components and learn how to customize them.

In the next lesson, we will cover what is Android activity.