YALOVA UNIVERSITY

HUAWEI

Android Programming with Huawei Mobile Services

Yunus OZEN
Muhammed Salih KARAKASLI
Cenk Faruk CAVGA
Sezer Yavuzer BOZKIR

Android

Introduction

- What is Android?
- Milestones of Android
- Basic Architecture of Android

Fundamentals

- App Components
  - Activities,
  - Services
  - Broadcast Receivers
  - Content Providers
- Manifest File
  - Permisions
- App Resources

First App

- Lets make our first app for android.

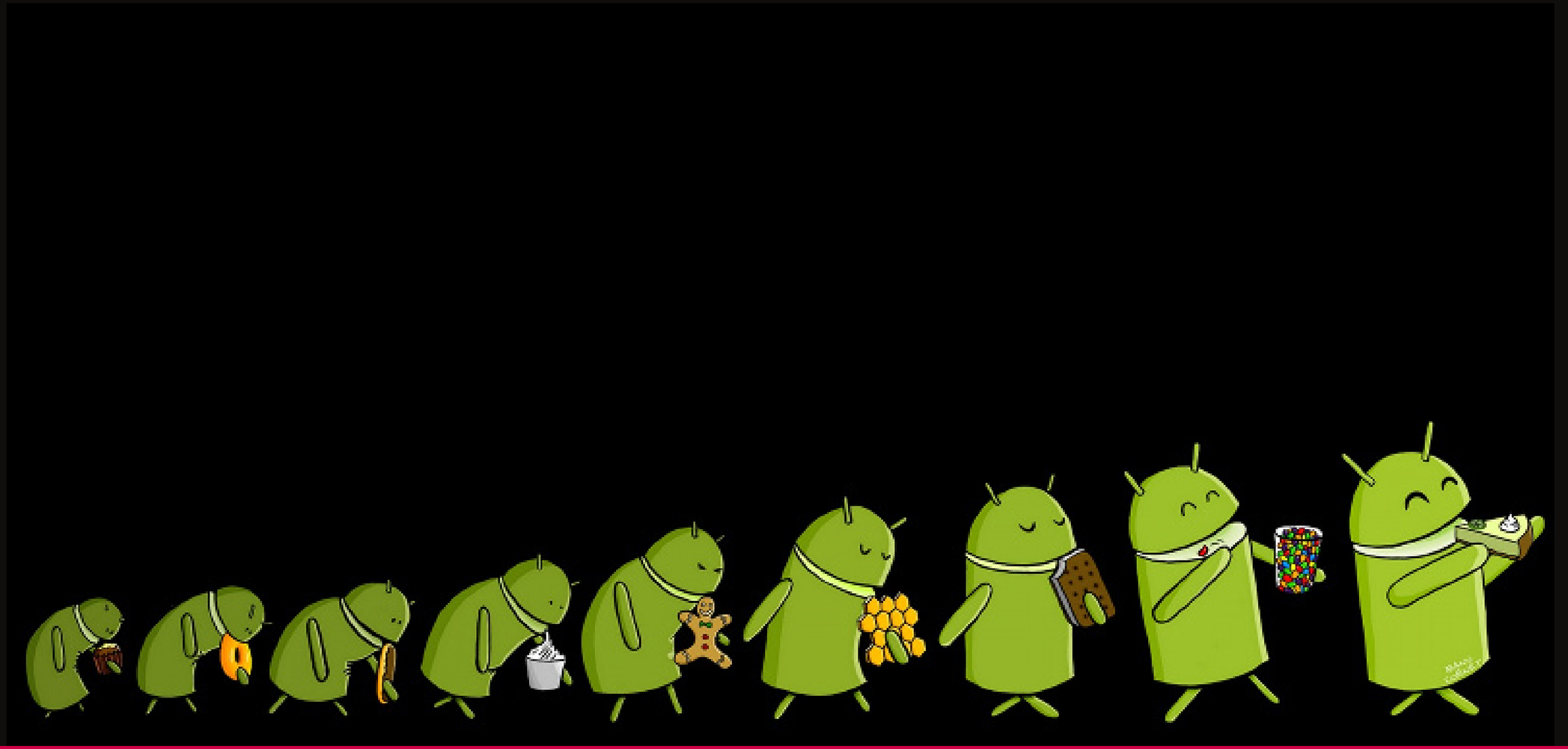
Introduction

android

linux based
open source
mobile operating system

It was originally developed by Open Handset Alliance and released in 2008 as Android Open Source Project (AOSP).

Milestones of Android



2008: First Android phone released

Google Maps.
Camera.
Gmail, Contacts, and Google Synchronization.
Web Browser.
Wireless supports – Wi-Fi and Bluetooth.

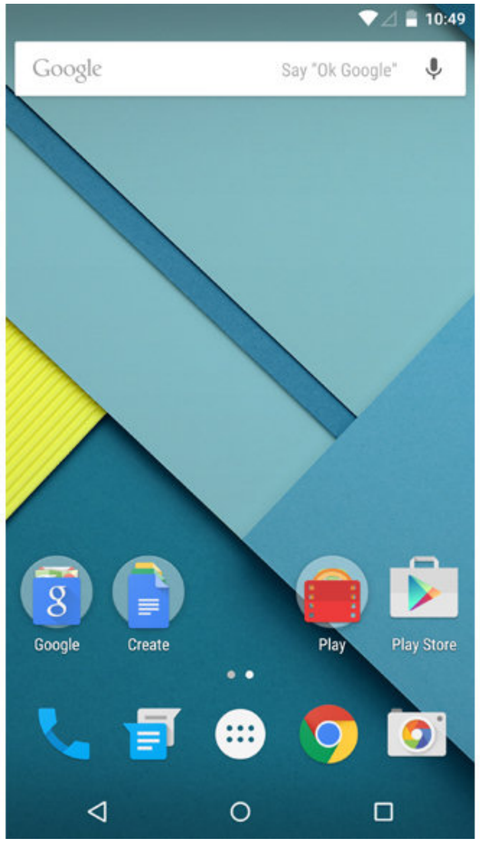
2012: 500 million Androids activated!

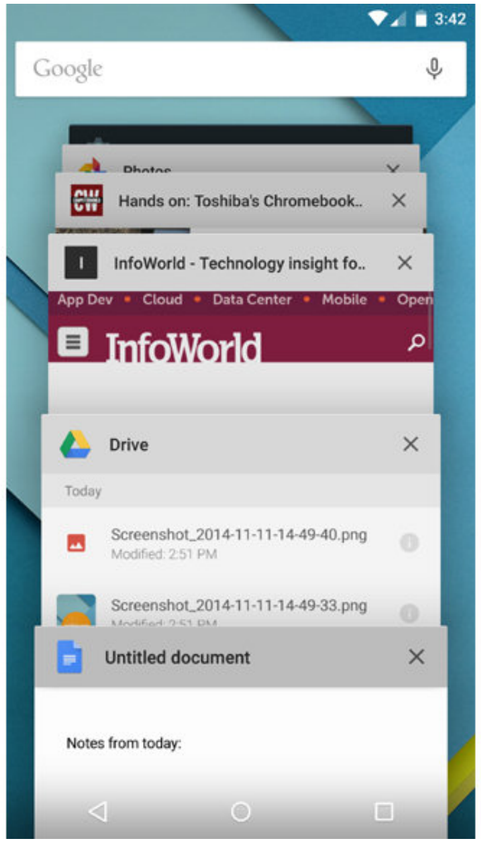
Android is the most popular operating system in the world, with 1.5 million new devices activated daily.

2013: 1 billion Android devices activated total

Android continues to grow and doesn't seem to be slowing down. It's still the most popular mobile operating system in the world and is starting to make its way into other devices like laptops, TVs, and watches.

2014: Android 5.0 is released



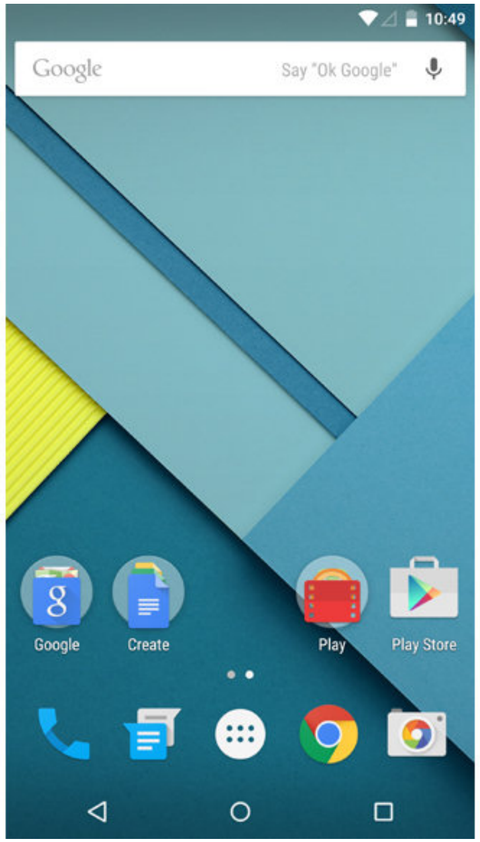


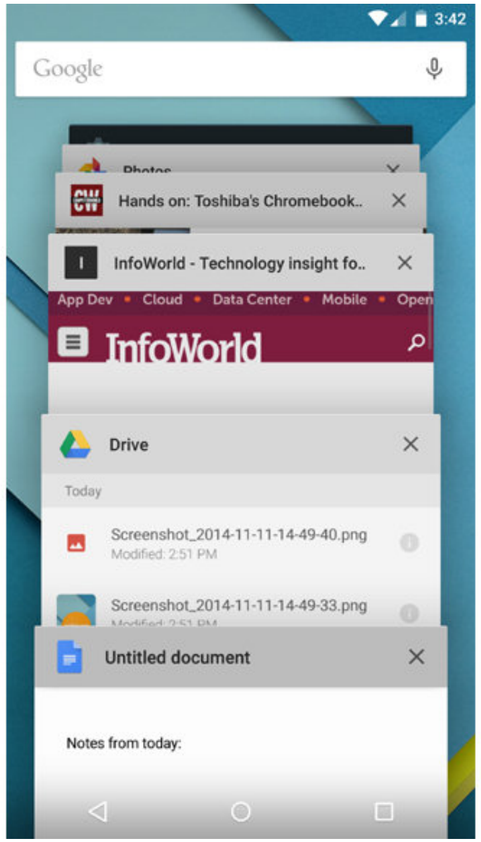
Lollipop launched the still-present-today Material Design standard, which brought a whole new look that extended across all of Android

2016: Securing the source!

Most importantly, 2016 saw Android start taking security and privacy more seriously. Marshmallow introduced platform-wide support for fingerprint readers, for example, and forced app developers to seek permission before accessing user data such as calls, location, and messages. These security-focused updates were a huge win for consumers.

2018: Reaching maturity





it would be the platform's new level of maturity. Android 9 Pie was a fantastic update to the operating system, with both user- and developer-centric features.

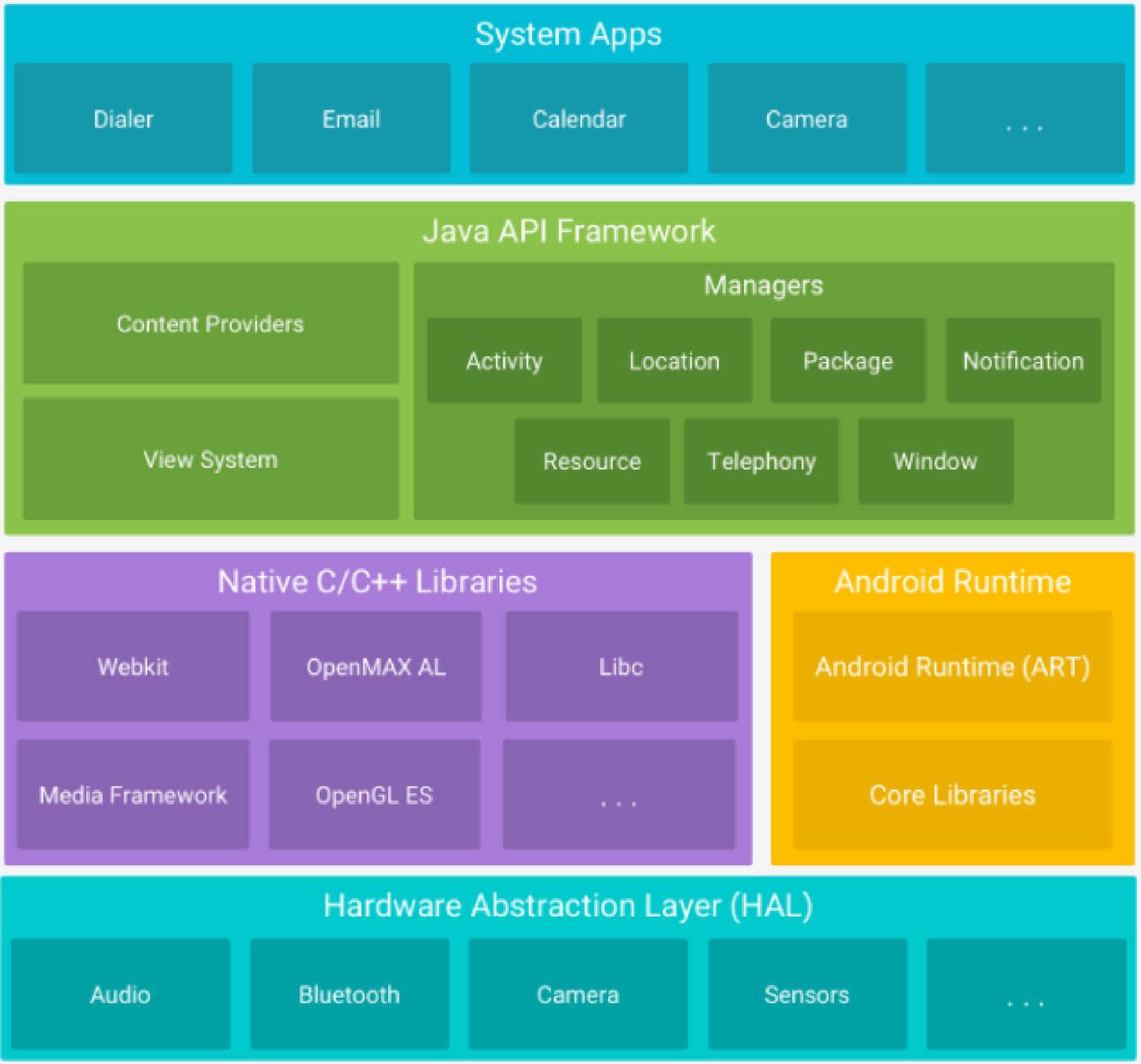
2020: Android 11

People-centric and expressive, with a new controls space and more privacy features.

Check official documents of Android;

developer.android.com/docs

Android Platform Architecture



Android comes with a set of core **apps** for email, SMS messaging, calendars, internet browsing, contacts, and more.

The entire feature-set of the Android OS is available to you through **APIs written in the Java** language. These APIs form the building blocks you need to create Android apps by simplifying the reuse of core, modular system components and services

Many core Android system components and services, such as ART and HAL, are built from native code that require native libraries written in **C and C++**. The Android platform provides Java framework APIs to expose the functionality of some of these native libraries to apps.

ART is written to run multiple virtual machines on low-memory devices by executing DEX files, a bytecode format designed specially for Android that's optimized for minimal memory footprint.

The hardware abstraction layer (HAL) provides standard interfaces that expose device hardware capabilities to the higher-level Java API framework. The HAL consists of multiple library modules, each of which implements an interface for a specific type of hardware component, such as the camera or bluetooth module. When a framework API makes a call to access device hardware, the Android system loads the library module for that hardware component.

App Components

App Components

App components are **the essential building blocks** of an Android app. Each component is an entry point through which the system or a user can enter your app. Some components depend on others.

There are four different types of app components:

1. Activities
2. Services
3. Broadcast receivers
4. Content providers

1. Activities

An activity is the entry point for interacting with the user. It represents a single screen with a user interface. For example, an email app might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails.

2. Services

A service is a general-purpose entry point for **keeping an app running in the background** for all kinds of reasons. It is a component that runs in the background to perform long-running operations or to perform work for remote processes.

A service **does not provide** a user interface.

3. Broadcast Receiver

A broadcast receiver is a component that enables the system to **deliver events** to the app outside of a regular user flow, allowing the app to respond to system-wide broadcast announcements. Because broadcast receivers are another well-defined entry into the app.

The system can deliver broadcasts **even to apps that aren't currently running.**

Many broadcasts originate from the system—for example, a broadcast announcing that the **screen has turned off, the battery is low, or a picture was captured**.

4. Content Provider

A content provider **manages a shared set of app data** that you can store in the file system, in a SQLite database, on the web, or on any other persistent storage location that your app can access.

Through the content provider, other apps can **query or modify the data** if the content provider allows it.

For example, the Android system provides a content provider that manages the **user's contact** information. As such, any app with the proper permissions can query the content provider, such as ContactsContract.Data, to read and write information about a particular person.

The Manifest

Android Manifest File

Before the Android system can start an app component, the **system must know** that the component exists **by reading the app's manifest file**, AndroidManifest.xml.

Your app must declare all its components in this file, which must be at the root of the app project directory.The manifest does a number of things in addition;

- Identifies any **user permissions** the app requires, such as Internet access.
- Declares the **minimum API** Level required by the app,
- Declares **hardware and software features** used or required by the app,
- Declares **API libraries** the app needs to be linked

1. Declaring Component

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
    <application android:icon="@drawable/app_icon.png" ... >
        <activity android:name="com.example.project.ExampleActivity"
                  android:label="@string/example_label" ... >
        </activity>
        ...
    </application>
</manifest>
```

- **<activity>** elements for activities.
- **<service>** elements for services.
- **<receiver>** elements for broadcast receivers.
- **<provider>** elements for content providers.

2.Declaring component capabilities

```
<manifest ... >
    ...
    <application ... >
        <activity android:name="com.example.project.ComposeEmailActivity">
            <intent-filter>
                <action android:name="android.intent.action.SEND" />
                <data android:type="*/*" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

When you declare an activity in your app's manifest, you can optionally include **intent filters** that declare the capabilities of the activity so it can respond to intents from other apps.

3. Declaring app requirements

```
<manifest ... >
    <uses-feature android:name="android.hardware.camera.any"
                  android:required="true" />
    <uses-sdk android:minSdkVersion="7" android:targetSdkVersion="19" />
    ...
</manifest>
```

For example, if your app requires a camera and uses APIs introduced in Android 2.1 (API Level 7), you must declare these as requirements in your manifest file as shown in the above example.

App Resources

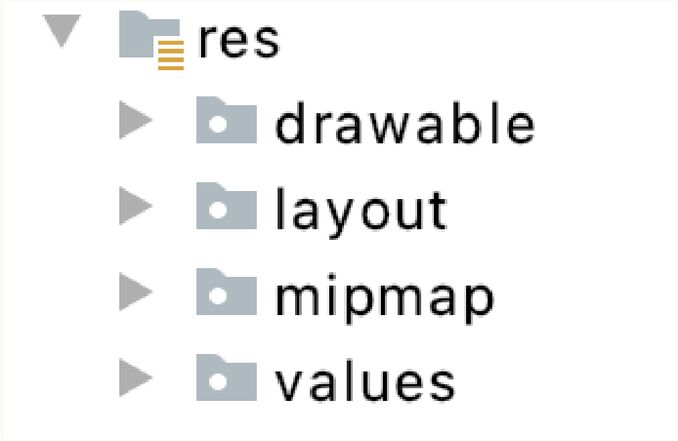
Resources

An Android app is composed of more than just code—it requires resources that are separate from the source code, such as **images, audio files, and anything relating to the visual presentation** of the app. For example, you can define animations, menus, styles, colors, and the layout of activity user interfaces with XML files.

Using app resources makes it easy to update various characteristics of your app without modifying code.

Providing sets of alternative resources enables you to **optimize your app for a variety of device configurations**, such as different languages and screen sizes.

Resources

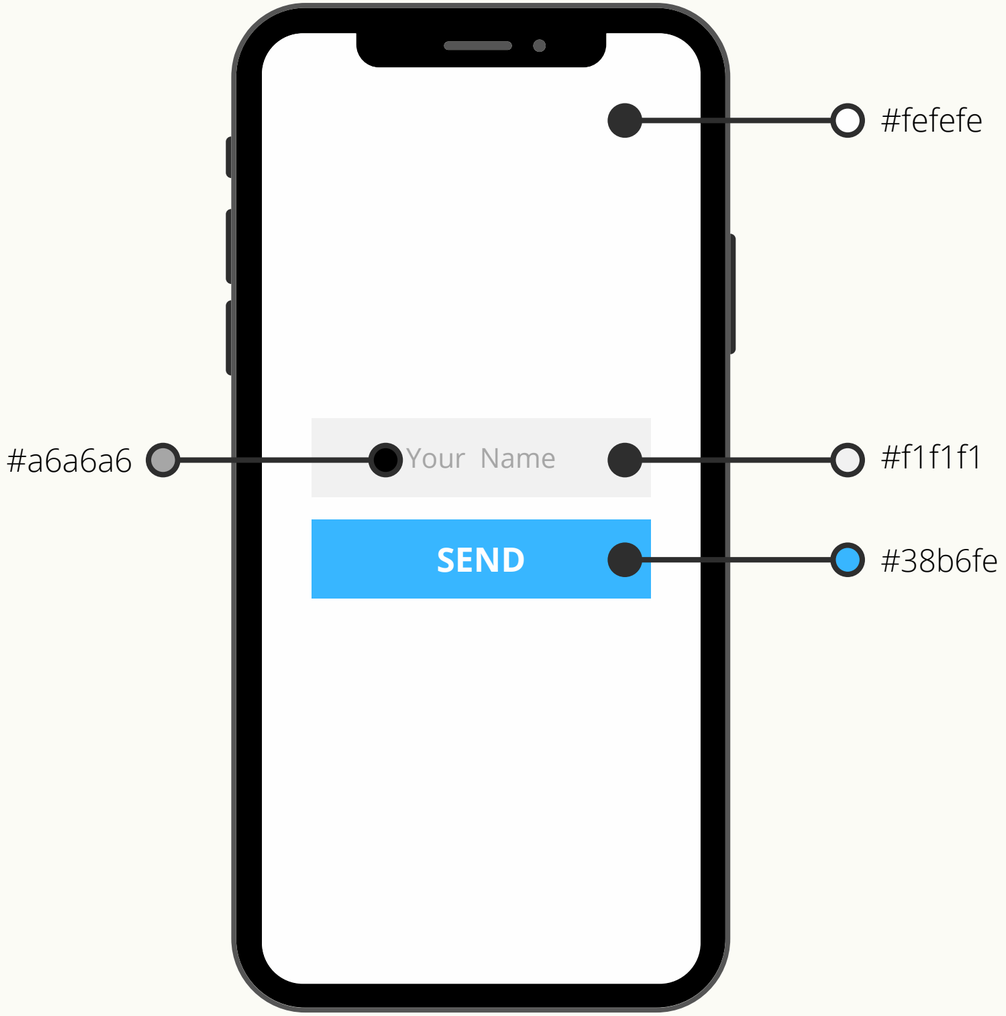


For every resource that you include in your Android project, the SDK build **tools define a unique integer ID**, which you can use to reference the resource from your app code or from other resources defined in XML.

For example, if your app contains an image file named logo.png (saved in the res/drawable/ directory), the SDK tools generate a resource ID named **R.drawable.logo**.

First App

First Activity



Design

Please follow design which shows at left side.

View Components

EditText
Button

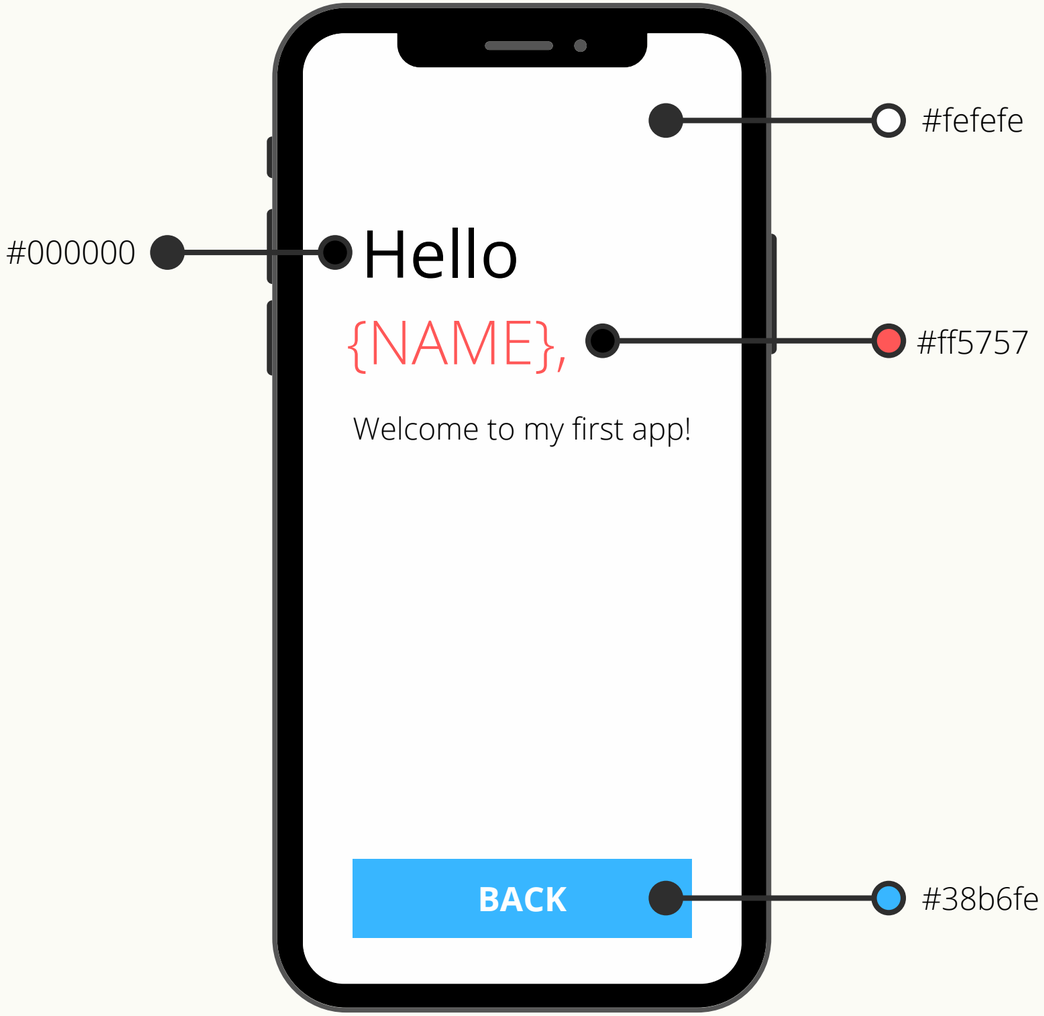
Validation Rules

Name cannot be less than 3 characters.
Name cannot be more than 20 characters.

Actions

When user click to send button;
Check validation rules, if name is not valid, show toast message, if name is valid, open second activity and send name to second activity.

Second Activity



Design

Please follow design which shows at left side.

View Components

EditText

Button

Validation Rules

There is not any validation rules.

Actions

When user click to back button;

Second activity should be closed and first activity should be present as initial.

LET'S TALK
CONTACT INFORMATİON

Muhammed Salih KARAKASLI
muhammed.salih.karakasli@huawei.com

Cenk Faruk CAVGA
cenk.faruk.cavga@huawei.com

Sezer Yavuzer BOZKIR
sezer.bozkir@huawei.com

Telegram Channel
will be created

Official Website
https://developer.huawei.com