Intro: Android development

Applikationsudvikling

NIFR 2024 - https://github.com/nicklasdean/ITA-intro-android

Introduction: Android Development

Applikationsudvikling

- Mobile Development
- From XML views to Jetpack compose
- Getting started in the Android Studio IDE
  - Software Development Kit (SDK) & build configurations
  - Project overview
- Composable Functions


2007
cingular
9:41 AM
Text
Calendar
Photos
Camera
Calculator
Stocks
Maps
Weather
Notes
Clock
Settings

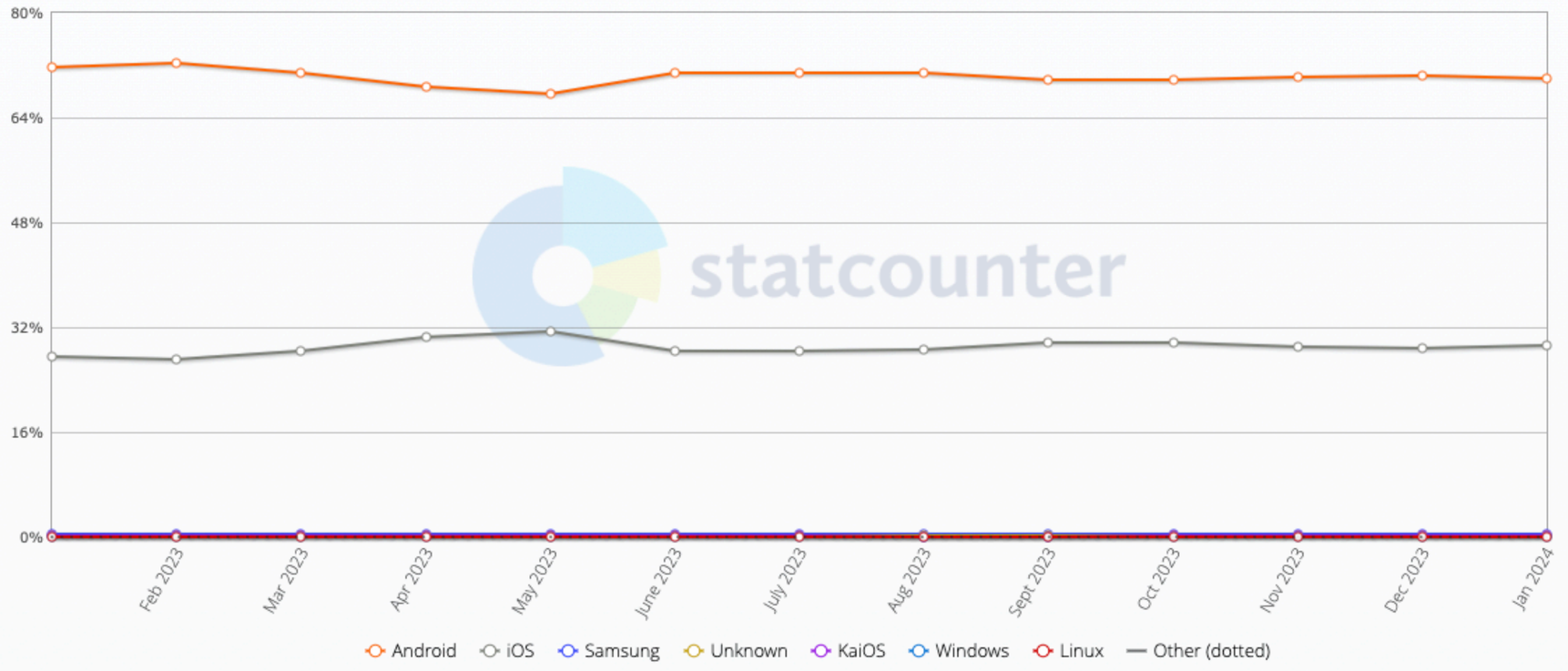
Android	iOS	Samsung	Unknown	KaiOS	Windows
69.94%	29.32%	0.36%	0.16%	0.16%	0.02%

Mobile Operating System Market Share Worldwide - January 2024

Mobile Operating System Market Share Worldwide

Jan 2023 - Jan 2024

Edit Chart Data



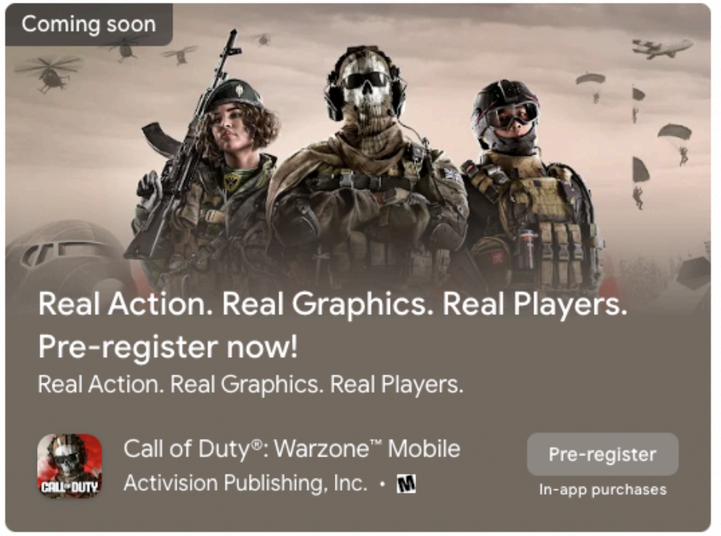
https://gs.statcounter.com/os-market-share/mobile/worldwide

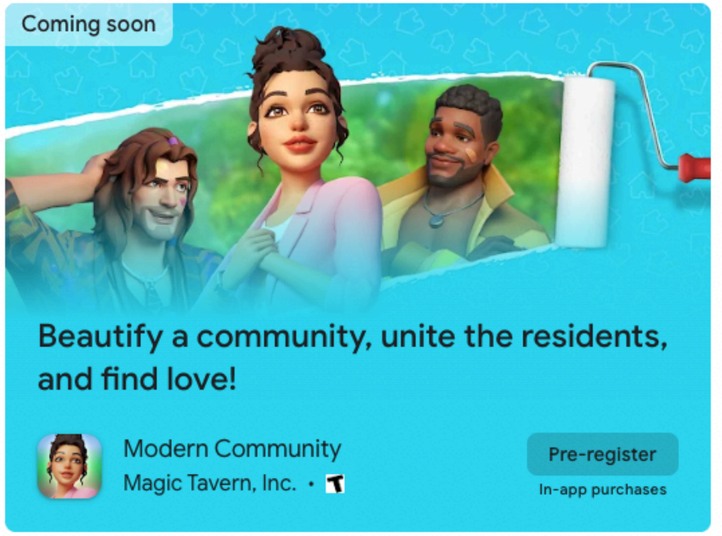
IOS & Android

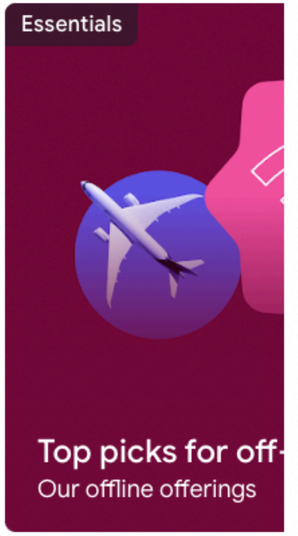
Google Play

Games | Apps | Movies & TV | Books | Kids

Windows | Phone | Tablet | TV | Chromebook | Watch







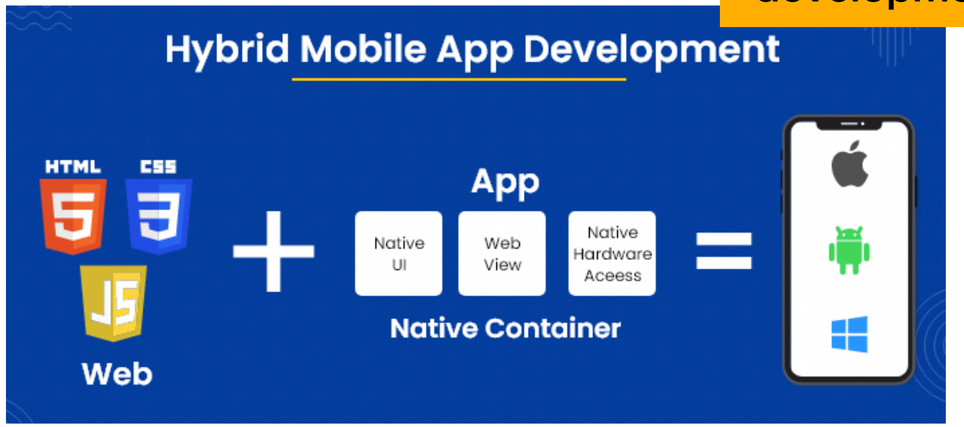
Top charts

Top free | Top grossing | Top paid

1	MONOPOLY GO! Board 4.7 ★	4	Block Blast! Puzzle 4.3 ★	7	Traffic Escape! Puzzle 4.6 ★
2	Royal Match Puzzle 4.6 ★	5	Ragnarok Origin: ROO New · Role Playing 4.1 ★	8	Hexa Sort Puzzle 4.5 ★
3	Roblox Adventure 4.4 ★	6	Wood Nuts & Bolts Puzzle Puzzle 4.5 ★	9	Subway Surfers Arcade 4.6 ★

Responsive web design

Hybrid app development



Native app development

Progressive web apps

Native applications

Android: Kotlin / IOS: Swift

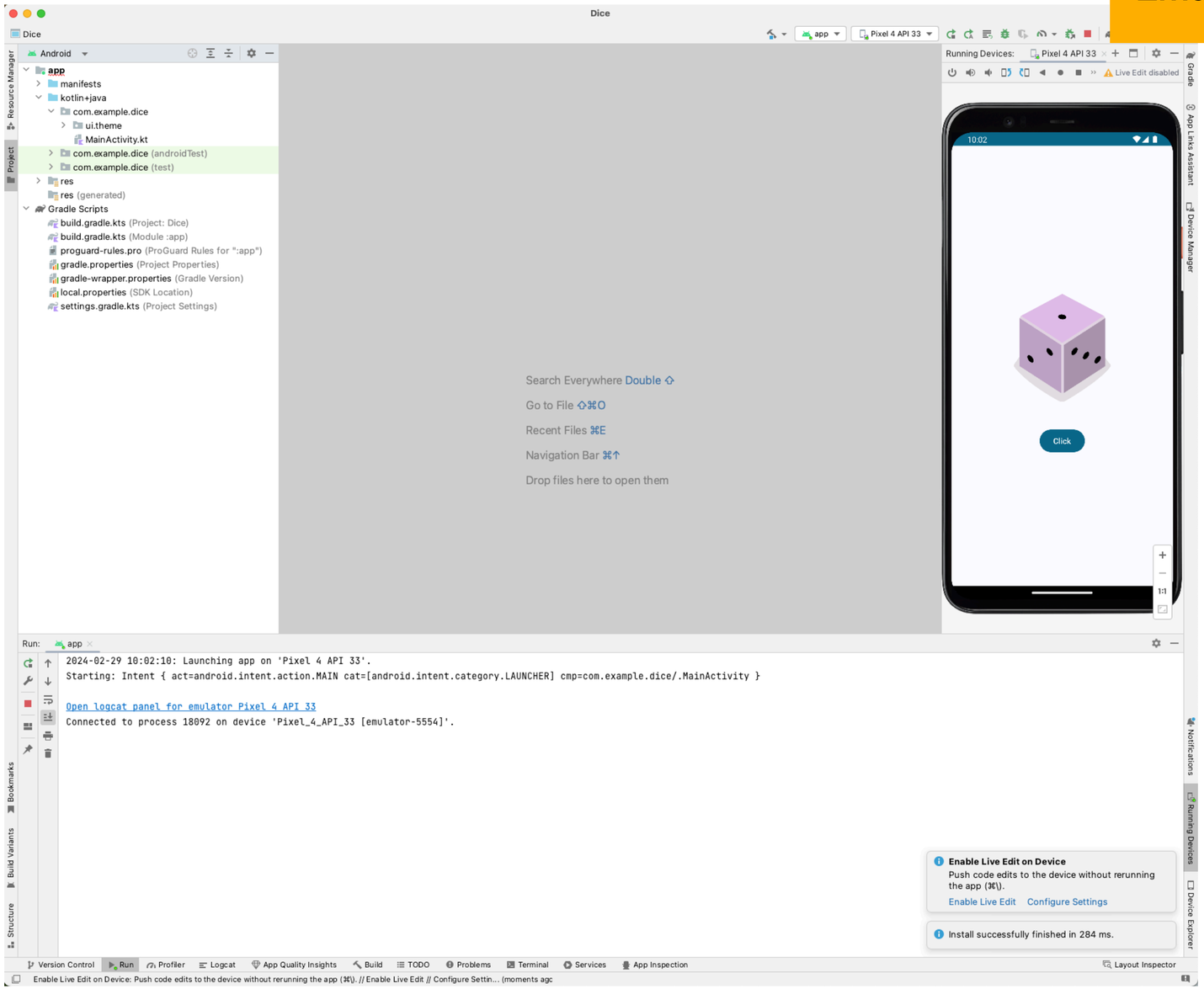
- Single code base = single app
- App needs to be installed on device
- Apps are located at a store (Play store, app store)
- Better performance
- Slower cross-platform time-to-market

Why native in the first place?

Development & context

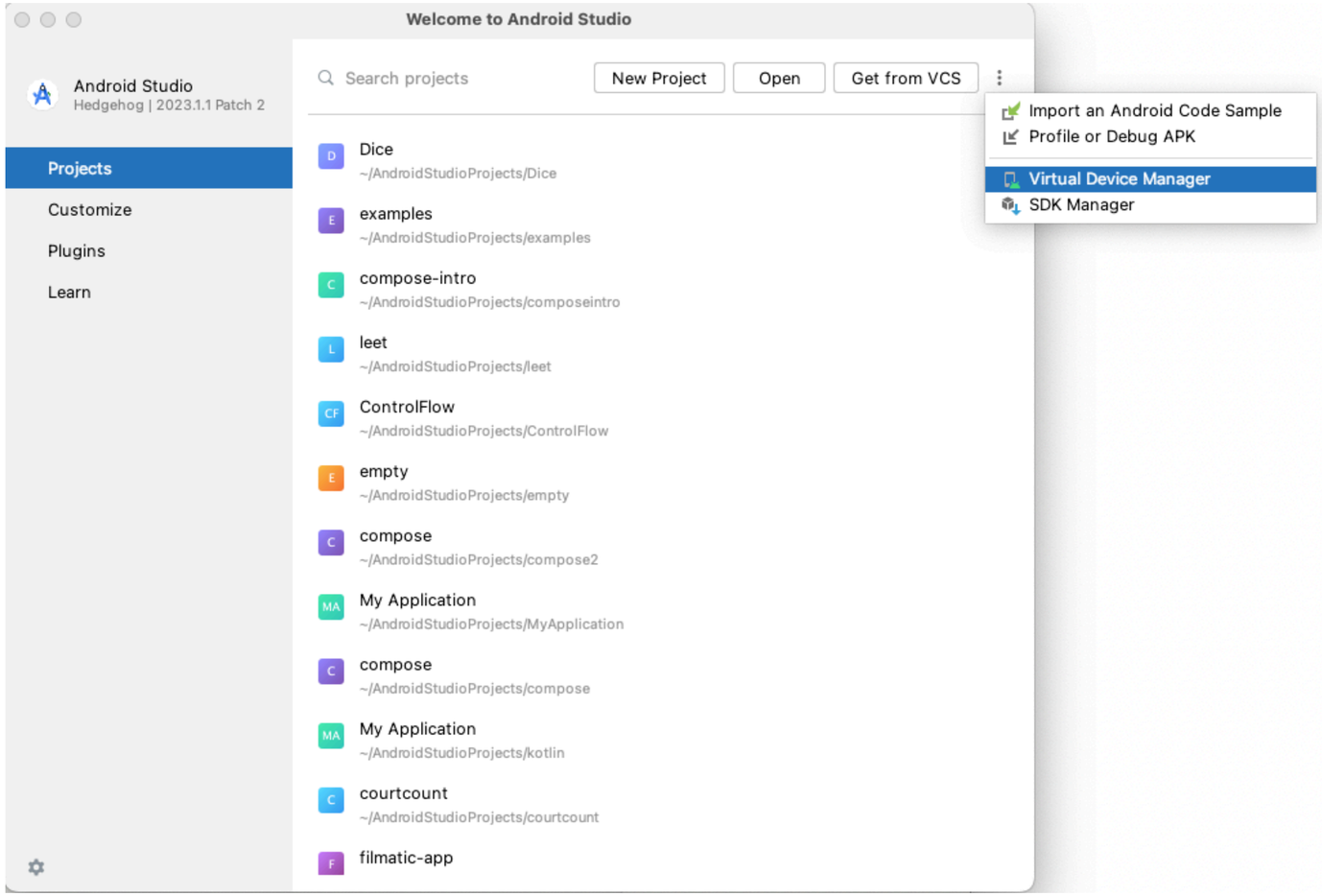
- API features were not accessible by the browser
  - Camera, offline mode, push notifications camera, geo-location
  - Background sync, bluetooth, device motion, contacts, file access, touch gestures
- But! Some of these features are now accessible by web (PWA)
  - Making PWA development more appealing
  - https://2048game.com/

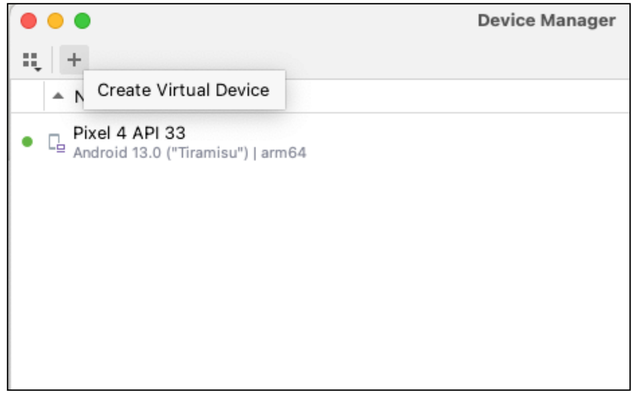
Emulator


Dice
Android
app
manifests
kotlin+java
com.example.dice
ui.theme
MainActivity.kt
com.example.dice (androidTest)
com.example.dice (test)
res
res (generated)
Gradle Scripts
build.gradle.kts (Project: Dice)
build.gradle.kts (Module :app)
proguard-rules.pro (ProGuard Rules for ":app")
gradle.properties (Project Properties)
gradle-wrapper.properties (Gradle Version)
local.properties (SDK Location)
settings.gradle.kts (Project Settings)
Search Everywhere Double ⇧
Go to File ⇧⌘O
Recent Files ⌘E
Navigation Bar ⌘↑
Drop files here to open them
Running Devices: Pixel 4 API 33
Live Edit disabled
10:02
Click
Run: app
2024-02-29 10:02:10: Launching app on 'Pixel 4 API 33'.
Starting: Intent { act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] cmp=com.example.dice/.MainActivity }
Open logcat panel for emulator Pixel 4 API 33
Connected to process 18092 on device 'Pixel_4_API_33 [emulator-5554]'.
Enable Live Edit on Device
Push code edits to the device without rerunning the app (⌘\).
Enable Live Edit
Configure Settings
Install successfully finished in 284 ms.
Version Control
Run
Profiler
Logcat
App Quality Insights
Build
TODO
Problems
Terminal
Services
App Inspection
Layout Inspector
Enable Live Edit on Device: Push code edits to the device without rerunning the app (⌘\). // Enable Live Edit // Configure Settin... (moments ago

Creating a project

Installing an emulator: Virtual Device Manager





Install the virtual device

Creating a project

Project initialisation

- Minimum SDK = The minimum version of android your app should be configured to run
  - Google Play target API level = API level 33 (August 2023)
- Build configuration language = Recommended: Gradle Kotlin DSL

https://developer.android.com/google/play/requirements/target-sdk

Configuration settings

Virtual Device

- Medium Phone
- Tiramisu API level 33
- Verify configuration > Finish

App configuration

Learning objectives

- Manifests
  - Launch configuration
- Resources
  - Images, strings, colours etc.
- Gradle scripts
  - Build scripts & Dependencies

Resources: Text & Color

Why common resources?

DRY - Don’t repeat yourself

- Application name
- Titles
- Capitalisation
- Common reference
- “Source of truth”

```
<resources>
    <string name="app_name">FirstApplication</string>
    <string name="author_name">Nicklas Frederiksen</string>
</resources>
```

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            FirstApplicationTheme {
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.background
                ) {
                    Greeting(stringResource(id = R.string.author_name));
                }
            }
        }
    }
}
```

Making another activity
- for exercises

```
<activity
    android:name=".SecondaryActivity"
    android:exported="false"
    android:label="@string/title_activity_secondary"
    android:theme="@style/Theme.FirstApplication" />
<activity
    android:name=".MainActivity"
    android:exported="true"
    android:label="@string/app_name"
    android:theme="@style/Theme.FirstApplication" >
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

MainActivity Launches

Secondary Activity Launches

```
<activity
    android:name=".SecondaryActivity"
    android:exported="true"
    android:label="@string/title_activity_secondary"
    android:theme="@style/Theme.FirstApplication" >
<intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>

<activity
    android:name=".MainActivity"
    android:exported="false"
    android:label="@string/app_name"
    android:theme="@style/Theme.FirstApplication" >

</activity>
```

From XML views to jetpack compose

Android Development

```
android.support.constraint.ConstraintLayout
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/coloryellow"
    tools:context="com.example.navdeepsingh.new_test.MainActivity">

    <Button
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:onClick="click"
        android:text="@string/button"
        tools:layout_constraintTop_creator="1"
        tools:layout_constraintRight_creator="1"
        android:layout_marginStart="8dp"
        android:layout_marginEnd="8dp"
        app:layout_constraintRight_toRightOf="parent"
        android:layout_marginTop="163dp"
        tools:layout_constraintLeft_creator="1"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"/>

</android.support.constraint.ConstraintLayout>
```

```
<TextView
    android:id="@+id/textView2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:text="@string/question_one"
    android:textSize="20sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

From XML views to jetpack compose

Android Development

```
@Composable
fun SimpleButton(onClick: () -> Unit) {
    Button(
        onClick = onClick,
        modifier = Modifier.padding(16.dp)
    ) {
        Text(text = "Click Me")
    }
}
```

```
@Composable
fun SimpleTextView() {
    Text(
        text = "Hello, Jetpack Compose!",
        style = MaterialTheme.typography.body1
    )
}
```

Understanding the Jetpack compose boilerplate

Functions in Kotlin

Higher order functions

- In Kotlin functions are first class citizens (like javascript)
- This means the language supports passing functions as arguments
- A higher order function is a function that takes another function as argument

```
fun higherOrderFunction(func: () -> Unit) {
    println("Before calling the passed function")
    func()
    println("After calling the passed function")
}

fun sayHello() {
    println("Hello, world!")
}

higherOrderFunction(::sayHello)
```

Lambda Functions

Kotlin & Compose

- Functions in Kotlin with a function parameter as the last parameter can use a simplified syntax

```
public fun ComponentActivity.setContent(
    parent: CompositionContext? = null,
    content: @Composable () -> Unit
)
```

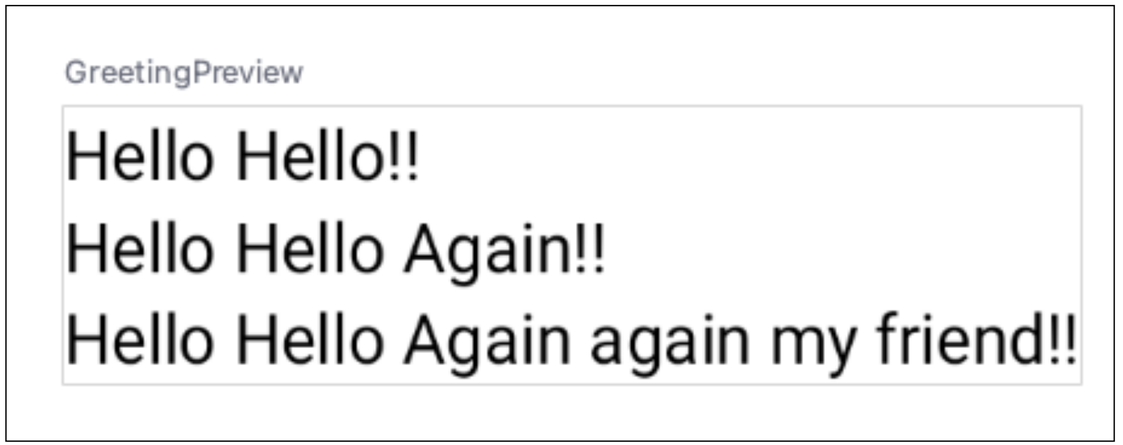
Last parameter is a function

Composable Functions

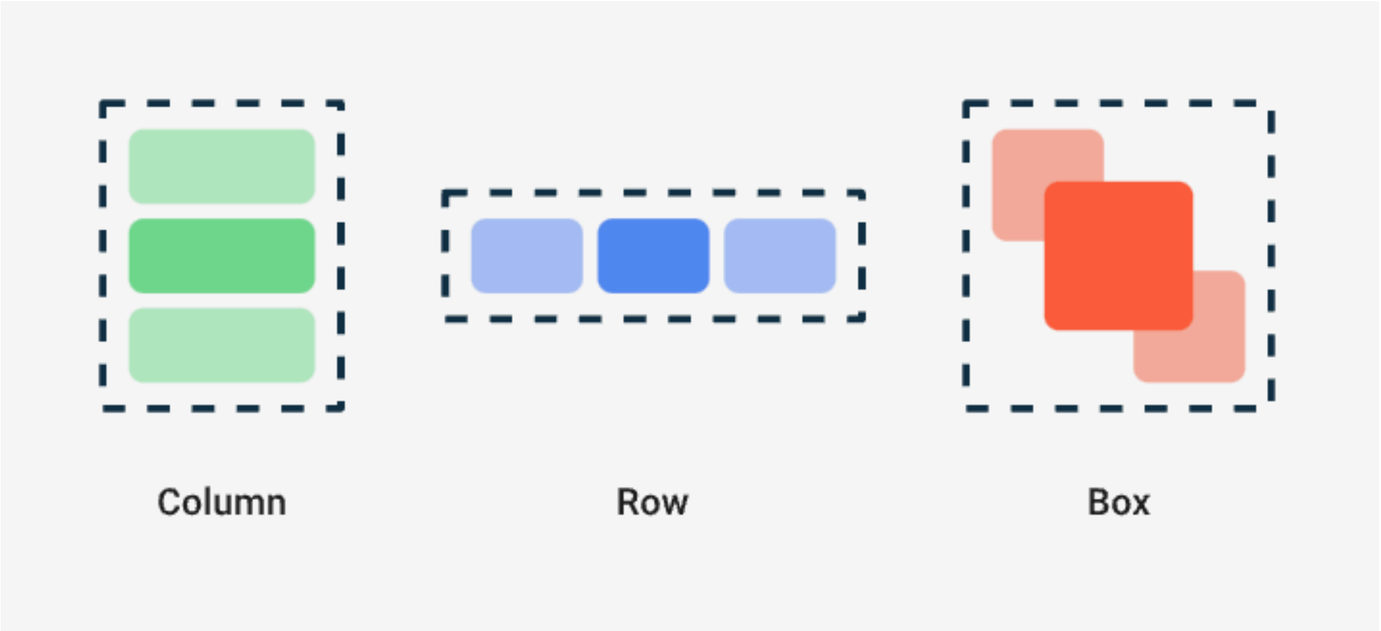
Introduction

- Composable functions are functions that represents a visual element or collections of visual elements
- Composable functions can contain other composable function
- Composables can be viewed with the @Preview annotation
- The Android API comes with composables such at Column, Row, Text
- Composable functions are written with CapitalFirstLetter

```
@Composable
fun Greetings(modifier: Modifier = Modifier){
    Column{
        Greeting("Hello!");
        Greeting("Hello Again!");
        Greeting("Hello Again again my friend!");
    }
}
```



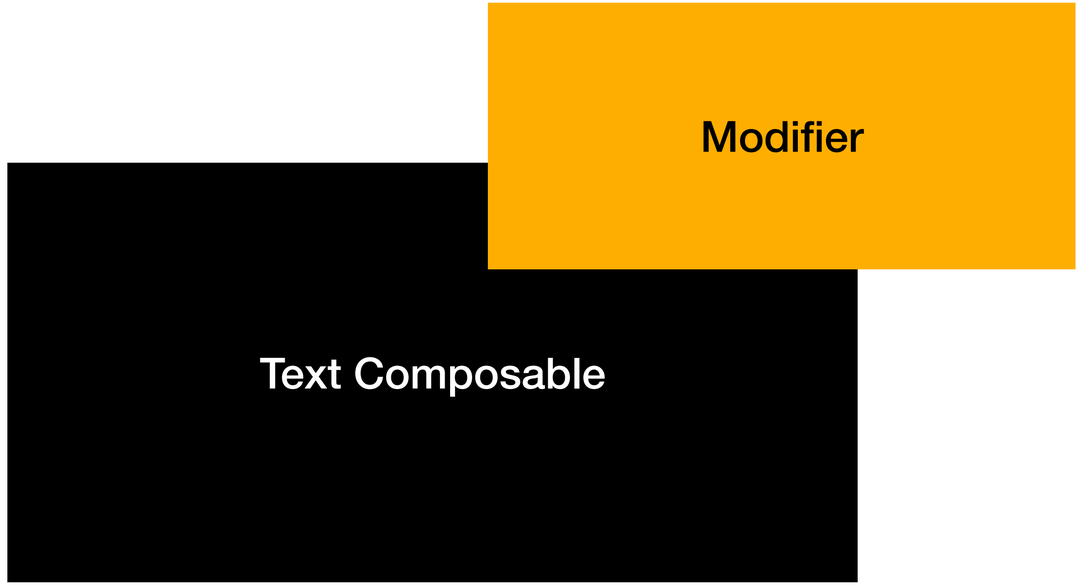
Example: From greeting to greetings in a column using composables



Modifiers

Jetpack Compose

- Companion object to a composable
- Used to modify the appearance of the composable
- Should always be passed as a companion object



https://developer.android.com/jetpack/compose/modifiers-list

Example Modifiers

```
@Composable
fun Greeting(name: String, modifier: Modifier = Modifier) {
    Text(
        text = "Hello $name!",
        modifier = Modifier
    )
}
```

GreetingPreview

Hello ITA23A!!

```
@Composable
fun Greeting(name: String, modifier: Modifier = Modifier) {
    Text(
        text = "Hello $name!",
        modifier = Modifier
            .padding(16.dp)
            .background(Color.Blue)
            .padding(8.dp)
            .size(200.dp, 100.dp)
            .padding(4.dp)
            .border(2.dp, Color.Red, shape = RoundedCornerShape(8.dp))
            .wrapContentHeight(align = Alignment.CenterVertically),
        textAlign = TextAlign.Center
    )
}
```

Reading documentation in the editor

Example Layout