



MOBILE PROGRAMMING

INTRODUCTION

ROBBY TAN



Prerequisites

- Basic Programming
- Object-Oriented Programming

Tools

- Browser
- Internet connection
- Android Studio

Kotlin Programming





Play Kotlin



<https://play.kotlinlang.org>

- *Code convention* pada Kotlin sama dengan Java.
 - **variable** & **method**: **camelCase**
 - showInfo()
 - birthDate
 - **Class name**: **First letter of each word >> uppercase**
 - Person
 - MainActivity
- Deklarasi *function/ method* menggunakan keyword **fun**.

The screenshot shows the Kotlin Playground interface in a web browser. The browser's address bar displays the URL: `play.kotlinlang.org/#eyJ2ZXJzaW9uIjoiaW4yLjIwIiwicGxhdGZvcml0eUJqYXZlbiYXJncyI6IiIsIm5vbmVNYXJrZXJzIjp0cnVlLCJ0aGVhZSI6Imlk`. The page header includes the Kotlin logo and navigation links: Solutions, Docs, API, Community, Teach, and Play. Below the header, there are dropdown menus for the Kotlin version (2.2.20) and the target JVM (JVM), along with a text input for 'Program arguments'. To the right of these are buttons for 'Copy link', 'Share code', and a 'Run' button. The main area is a code editor containing the following Kotlin code:

```
/**
 * You can edit, run, and share this code.
 * play.kotlinlang.org
 */
fun main() {
    println("Hello, world!!!")
}
```

Below the code editor, the output of the program is displayed: 'Hello, world!!!'. There are help and close icons on the right side of the output area.

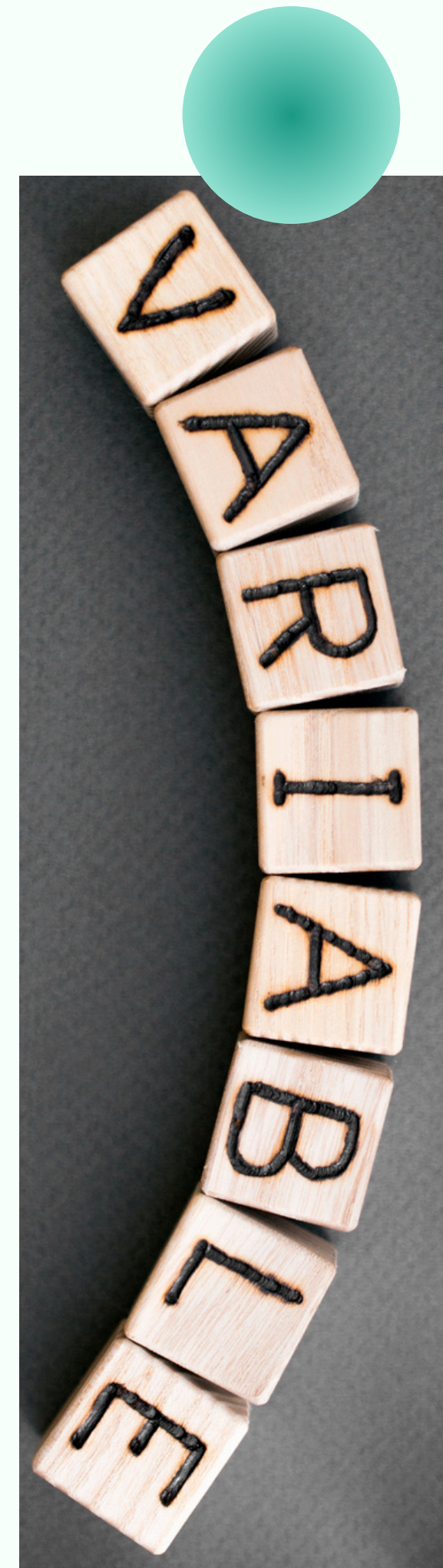


Variable and Data Type

- Kotlin data type
 - **Number**: Byte, Short, Int, Long
 - **Floating-point**: Float, Double
 - **Boolean**
 - **String**
- Deklarasi variabel menggunakan keyword **var** dan **val**.

```
var myMoney: Double = 2_000_000_000
val firstName: String = "Robby"
```

```
1. fun main() {
2.     val byteValue: Byte = Byte.MAX_VALUE
3.     val shortValue: Short = Short.MAX_VALUE
4.     val intValue: Int = Int.MAX_VALUE
5.     val longValue: Long = Long.MAX_VALUE
6.     val floatValue: Float = 3.14f
7.     val doubleValue: Double = 2096.98198752
8.     val name: String = "Robby"
9.     println("Byte val: $byteValue, short val: $shortValue")
10.    println("Int val: $intValue, long val: $longValue")
11.    println(String.format("Float val: %f; Double val: %f", floatValue, doubleValue))
12.    println("My name is $name")
13. }
```

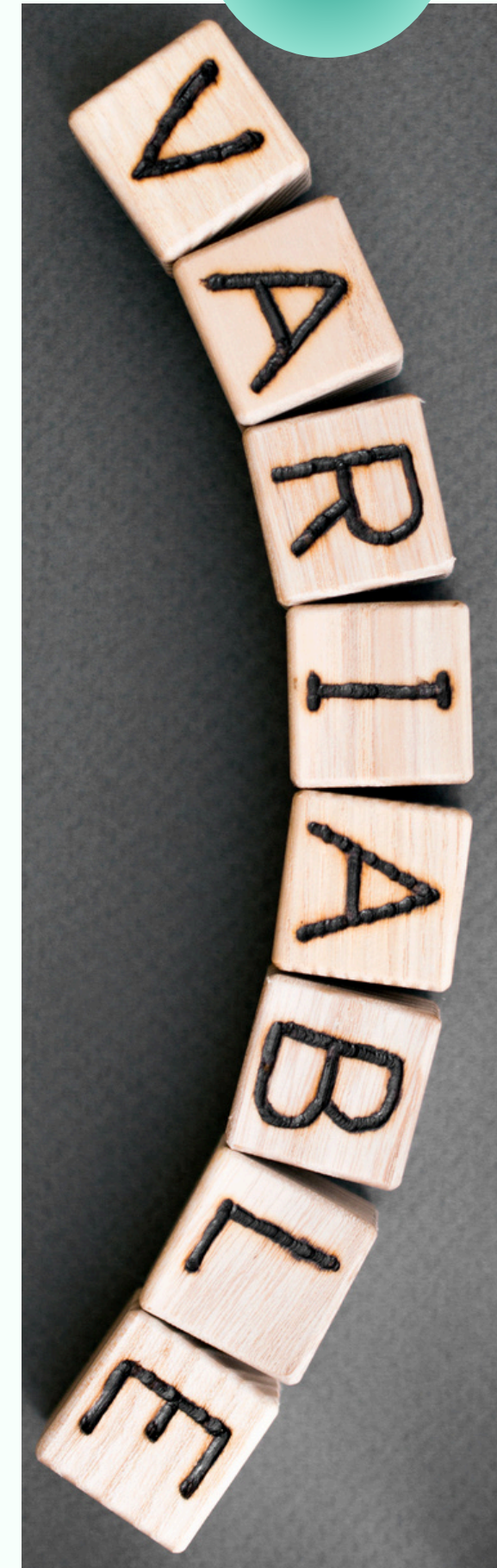




Kotlin Type Inference

- *Compiler* pada Kotlin memiliki kemampuan untuk mendefinisikan tipe data meskipun tidak didefinisikan secara eksplisit.

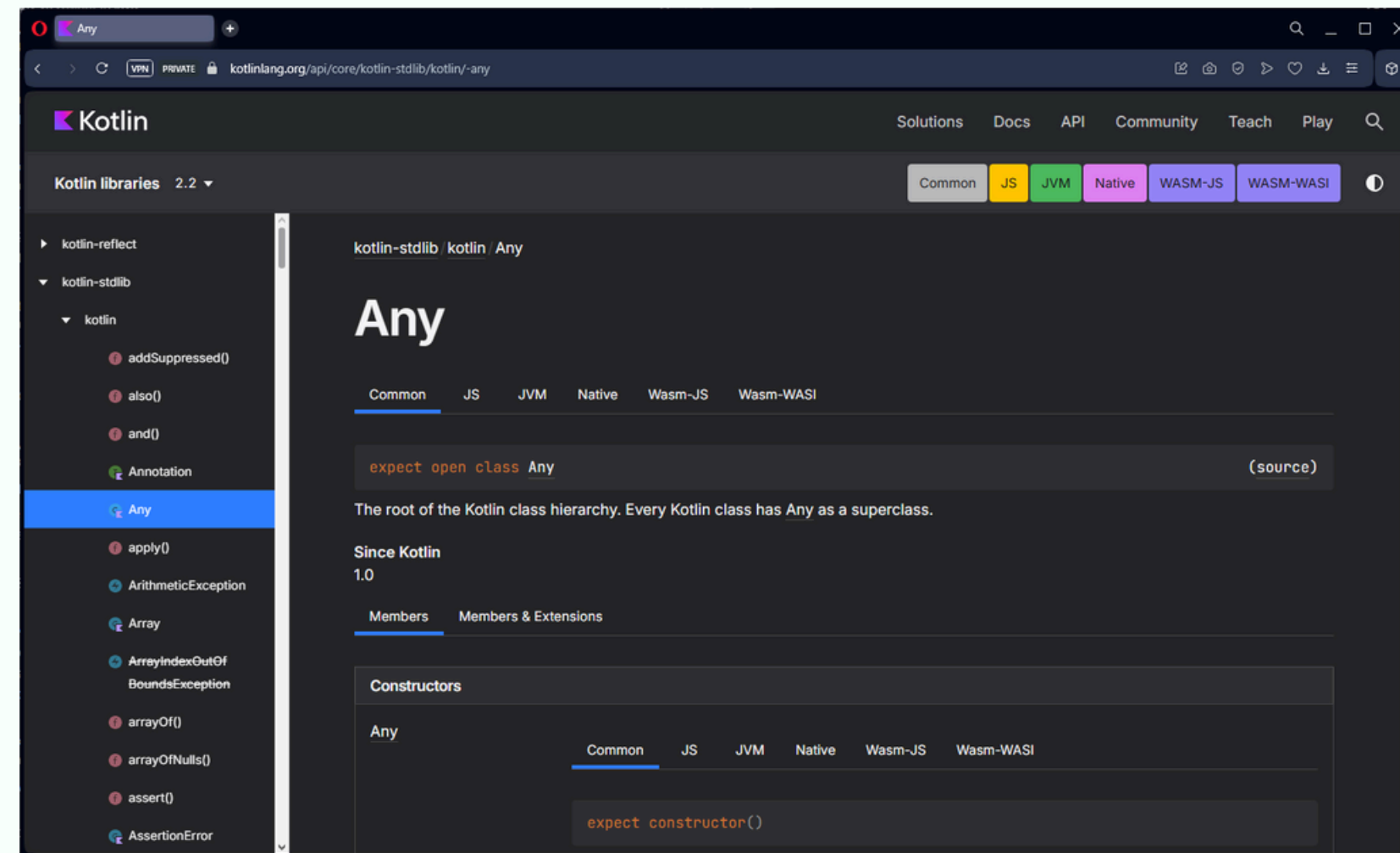
```
1. fun main() {  
2.     val number = 250  
3.     val firstName = "John"  
4.     println(String.format("Type for variable number is %s, Value = %d", number::class.simpleName, number))  
5.     println(String.format("Type for variable firstName is %s, Value = %s", firstName::class.simpleName, firstName))  
6. }
```





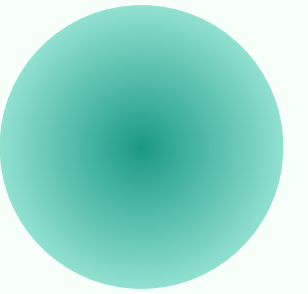
Kotlin Superclass: **Any**

- **Any** adalah induk dari seluruh *class* pada Kotlin.
- Padanan pada Java: **Object** *class*.



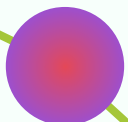


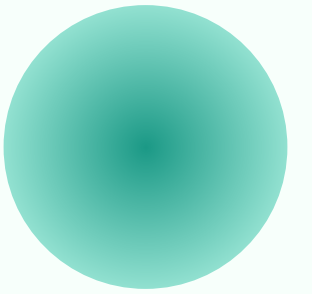
Kotlin Function



- Function pada Kotlin dideklarasikan dengan menggunakan *keyword* **fun**.
- Function pada Kotlin secara *default* me-return **Unit** meskipun tidak ditulis secara eksplisit (**void** pada Java).

```
1. fun main() {  
2.     println(hello("Robby Tan"))  
3.     sum(25, 111)  
4. }  
5.  
6. fun hello(name: String): String {  
7.     return "Hello $name. Welcome to Kotlin Programming"  
8. }  
9.  
10. fun sum(a: Int, b: Int) {  
11.     print("Sum of $a and $b is ${a + b}.")  
12. }
```





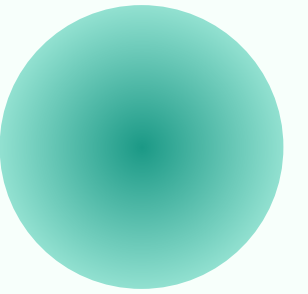
Kotlin Named Arguments & Default Value

- Kita dapat mengubah urutan input *arguments* dengan menggunakan **named arguments** (baris ke-3).
- *Named arguments* disesuaikan dengan nama variabel yang dibuat pada fungsi.

```
1. fun main() {  
2.     introduce("Richard Smith", "Soccer")  
3.     introduce(hobby = "Fishing", name = "John Doe")  
4.     drive("Sport Center")  
5.     drive()  
6. }  
7.  
8. fun introduce(name: String, hobby: String): Unit {  
9.     println("Hello. My name is $name. My hobby is $hobby")  
10. }  
11.  
12. fun drive(destination: String = "Home") {  
13.     println("Command accepted. We are going to $destination")  
14. }
```

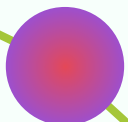
Android Studio





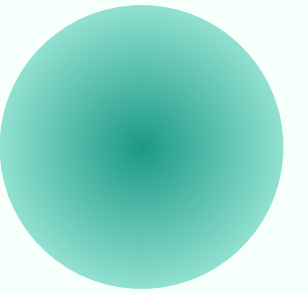
Android Studio System Requirements

- 64-bit Microsoft® Windows® 8/10/11
- x86_64 CPU architecture; 2nd generation Intel Core or newer, or AMD CPU with support for a Windows Hypervisor
- 8 GB RAM or more
- 8 GB of available disk space minimum (IDE + Android SDK + Android Emulator)
- Download link: <https://developer.android.com/studio#get-android-studio>

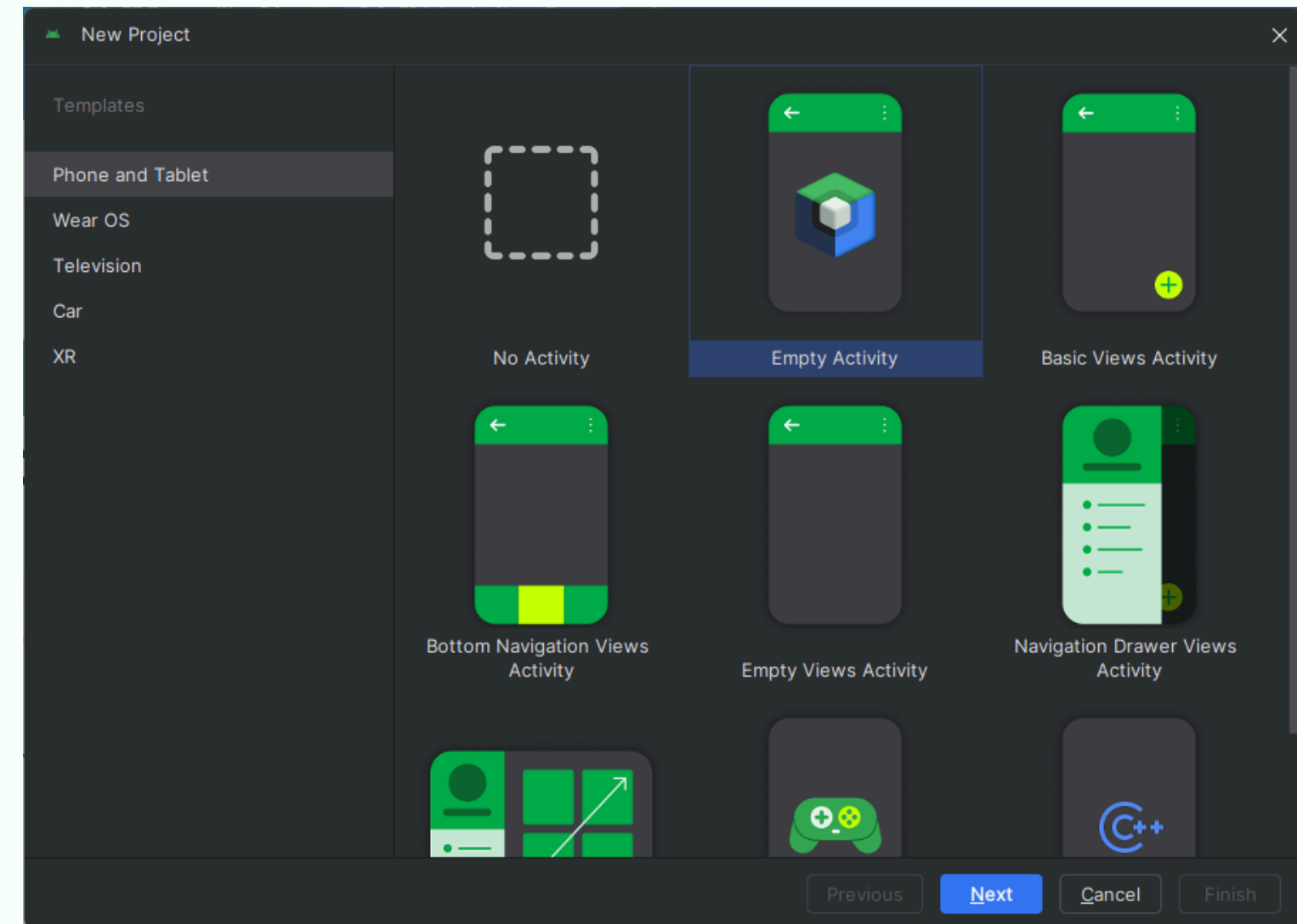
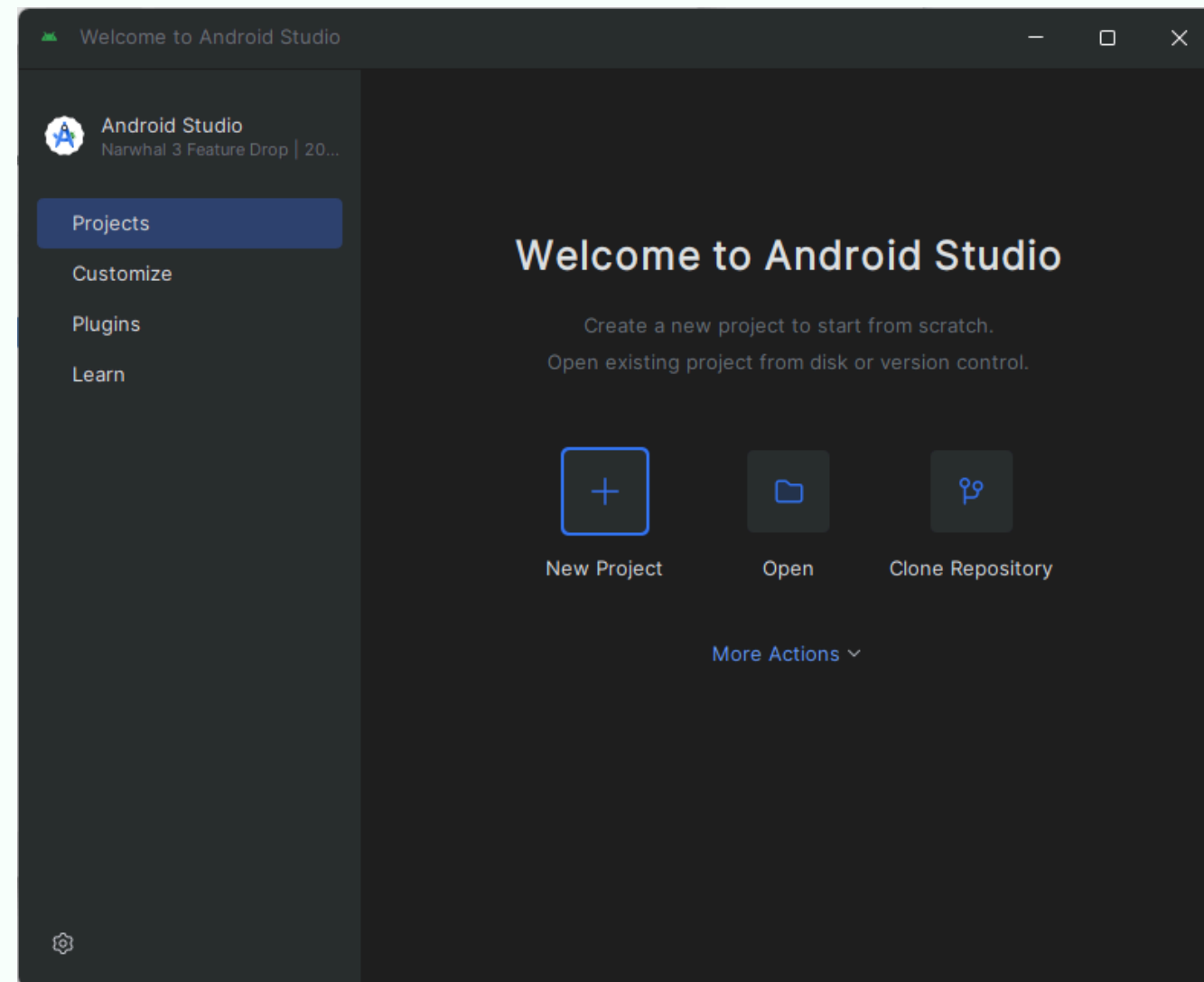


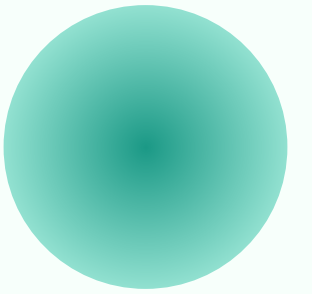
First Android App





Create Android Project





Android Project Name and Settings

New Project

Empty Activity

Create a new empty activity with Jetpack Compose

Name: Demo MP 01

Package name: com.robby.demomp01

Save location: E:\Projects\Android\DemoMP01

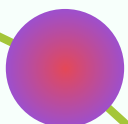
Minimum SDK: API 24 ("Nougat"; Android 7.0)

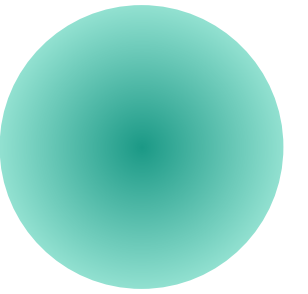
Information: Your app will run on approximately 98.6% of devices. [Help me choose](#)

Build configuration language: Kotlin DSL (build.gradle.kts) [Recommended]

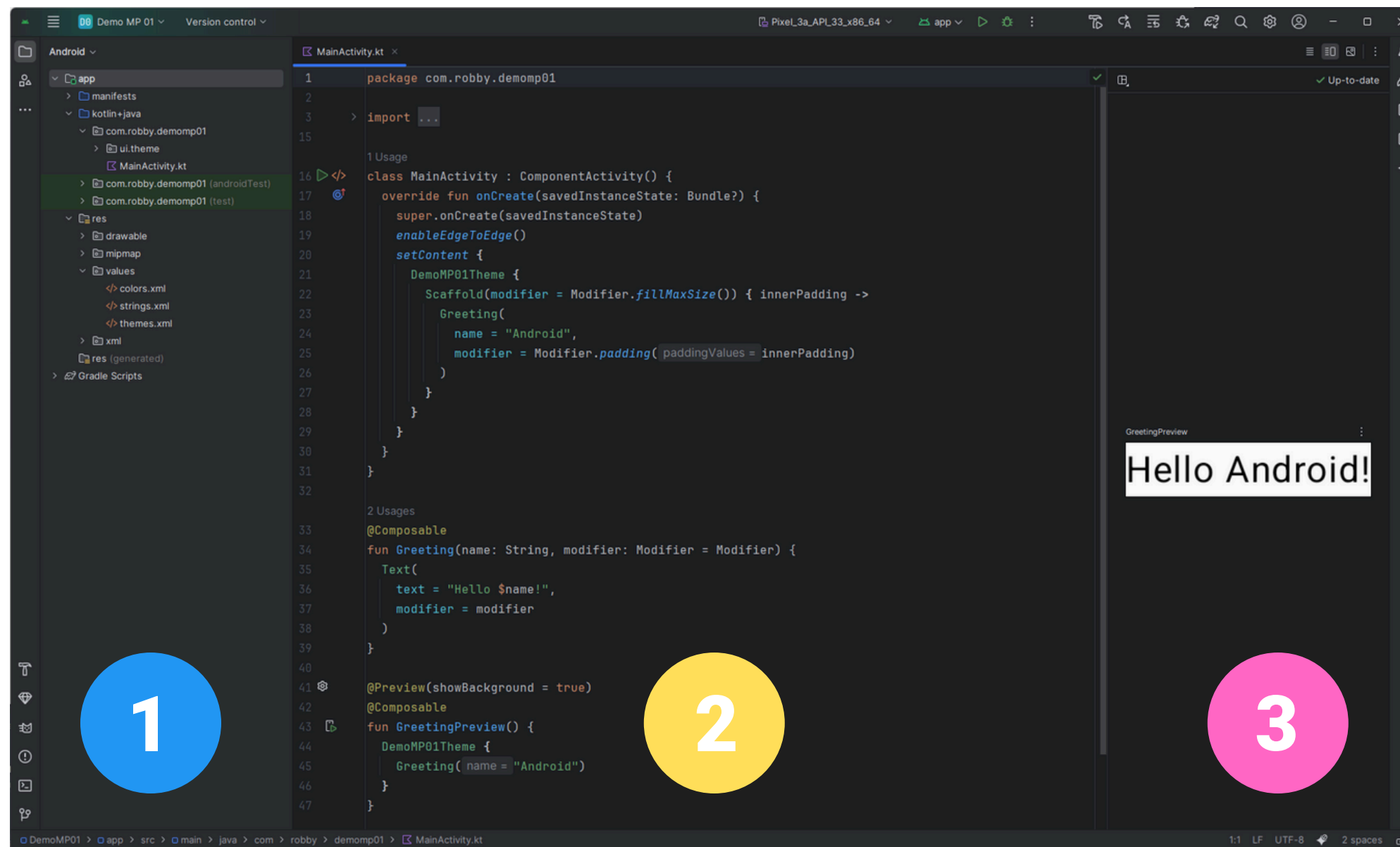
Previous Next Cancel Finish

- Penamaan *package name* >> **reverse domain**.
- *Package name* bersifat **unik** untuk setiap aplikasi.
- SDK yang di-*install* sebaiknya adalah SDK yang paling umum digunakan oleh pengguna.
- Proses *build gradle* membutuhkan koneksi internet dan **relatif memakan waktu**.





Android Project Name and Settings



1

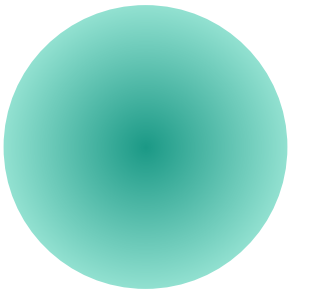
Project view >> berisi struktur folder dan file dari proyek Android

2

Android source code

3

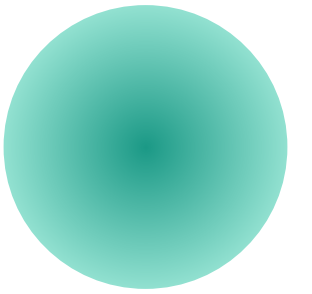
Hasil preview berdasarkan source code (didapat setelah proses build & refresh)



Source Code Explanation (1/2)

```
1. class MainActivity : ComponentActivity() {  
2.     override fun onCreate(savedInstanceState: Bundle?) {  
3.         super.onCreate(savedInstanceState)  
4.         enableEdgeToEdge()  
5.         setContent {  
6.             DemoMP01Theme {  
7.                 Scaffold(modifier = Modifier.fillMaxSize()) { innerPadding ->  
8.                     Greeting(  
9.                         name = "Android",  
10.                        modifier = Modifier.padding(innerPadding)  
11.                    )  
12.                }  
13.            }  
14.        }  
15.    }  
16. }
```

- **MainActivity**
 - *subclass* dari ComponentActivity (**Jetpack Compose**)
 - dahulu *subclass* dari AppCompatActivity (**Android Legacy**)
- **onCreate()**
 - *entry point* (**main**) aplikasi Android
 - enableEdgeToEdge()
 - pengaturan aplikasi sehingga menggunakan seluruh area dari layar
 - setContent()
 - Fungsi untuk mendefinisikan layout UI aplikasi
 - Fungsi yang dipanggil wajib memiliki anotasi @Composable

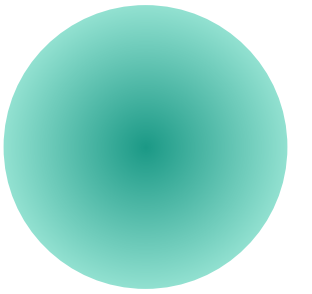


Source Code Explanation (2/2)

```
1. class MainActivity : ComponentActivity() {
2.     override fun onCreate(savedInstanceState: Bundle?) {
3.         super.onCreate(savedInstanceState)
4.         enableEdgeToEdge()
5.         setContent {
6.             DemoMP01Theme {
7.                 Scaffold(modifier = Modifier.fillMaxSize()) { innerPadding ->
8.                     Greeting(
9.                         name = "Android",
10.                        modifier = Modifier.padding(innerPadding)
11.                    )
12.                }
13.            }
14.        }
15.    }
16. }
```

```
1. @Composable
2. fun Greeting(name: String, modifier: Modifier = Modifier) {
3.     Text(
4.         text = "Hello. My name is $name!",
5.         modifier = modifier
6.     )
7. }
```

- **onCreate()** cont.
 - **DemoMP01Theme** >> tema aplikasi (Theme.kt)
 - **Scaffold**
 - *composable* yang menyediakan struktur tata letak dasar untuk layar (e.g. top bar, bottom bar, FAB)
 - **Modifier.fillMaxSize()** >> pengisi ke seluruh layar
 - **Greeting**
 - Fungsi **composable**
 - Mengisi parameter name dari fungsi Greeting

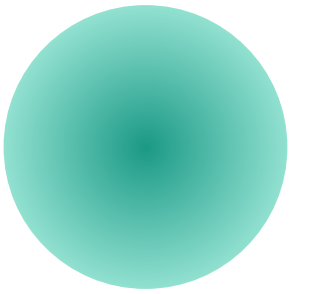


Source Code Explanation (2/2)

```
1. class MainActivity : ComponentActivity() {
2.     override fun onCreate(savedInstanceState: Bundle?) {
3.         super.onCreate(savedInstanceState)
4.         enableEdgeToEdge()
5.         setContent {
6.             DemoMP01Theme {
7.                 Scaffold(modifier = Modifier.fillMaxSize()) { innerPadding ->
8.                     Greeting(
9.                         name = "Android",
10.                        modifier = Modifier.padding(innerPadding)
11.                    )
12.                }
13.            }
14.        }
15.    }
16. }
```

```
1. @Composable
2. fun Greeting(name: String, modifier: Modifier = Modifier) {
3.     Text(
4.         text = "Hello. My name is $name!",
5.         modifier = modifier
6.     )
7. }
```

- **onCreate()** cont.
 - **DemoMP01Theme** >> tema aplikasi (Theme.kt)
 - **Scaffold**
 - *composable* yang menyediakan struktur tata letak dasar untuk layar (e.g. top bar, bottom bar, FAB)
 - **Modifier.fillMaxSize()** >> pengisi ke seluruh layar
 - **Greeting**
 - Fungsi **composable**
 - Mengisi parameter name dari fungsi Greeting



Change Background with Surface

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

@Preview
@Composable
```

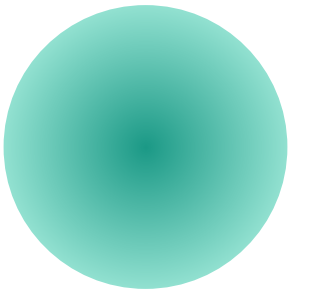
This intention surrounds selected compose code with a widget.

Surround with widget

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

Surround with Container
Surround with Row
Surround with Column

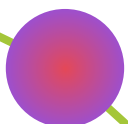
- Penambahan container untuk composable/ component Text
 - Alt + Enter pada atribut text
 - Pilih Surround with widget >> Surround with Container
- Default Container yang diberikan adalah Box. Ubah menjadi Surface.

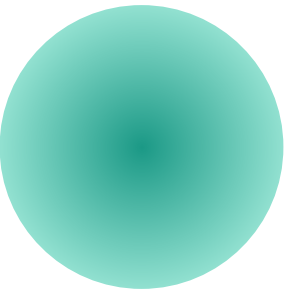


Change Background with Surface: Result

```
1. @Composable
2. fun Greeting(name: String, modifier: Modifier = Modifier) {
3.     Surface(color = Color.Yellow) {
4.         Text(
5.             text = "Hello. My name is $name!",
6.             modifier = modifier
7.         )
8.     }
9. }
```

- Note:
 - Class **Color** untuk *property* color berada pada *package* **androidx.compose.ui.graphics.Color**

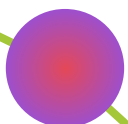




Add Padding

```
1. @Composable
2. fun Greeting(name: String, modifier: Modifier = Modifier) {
3.     Surface(color = Color.Yellow) {
4.         Text(
5.             text = "Hello. My name is $name!",
6.             modifier = modifier.padding(24.dp)
7.         )
8.     }
9. }
```

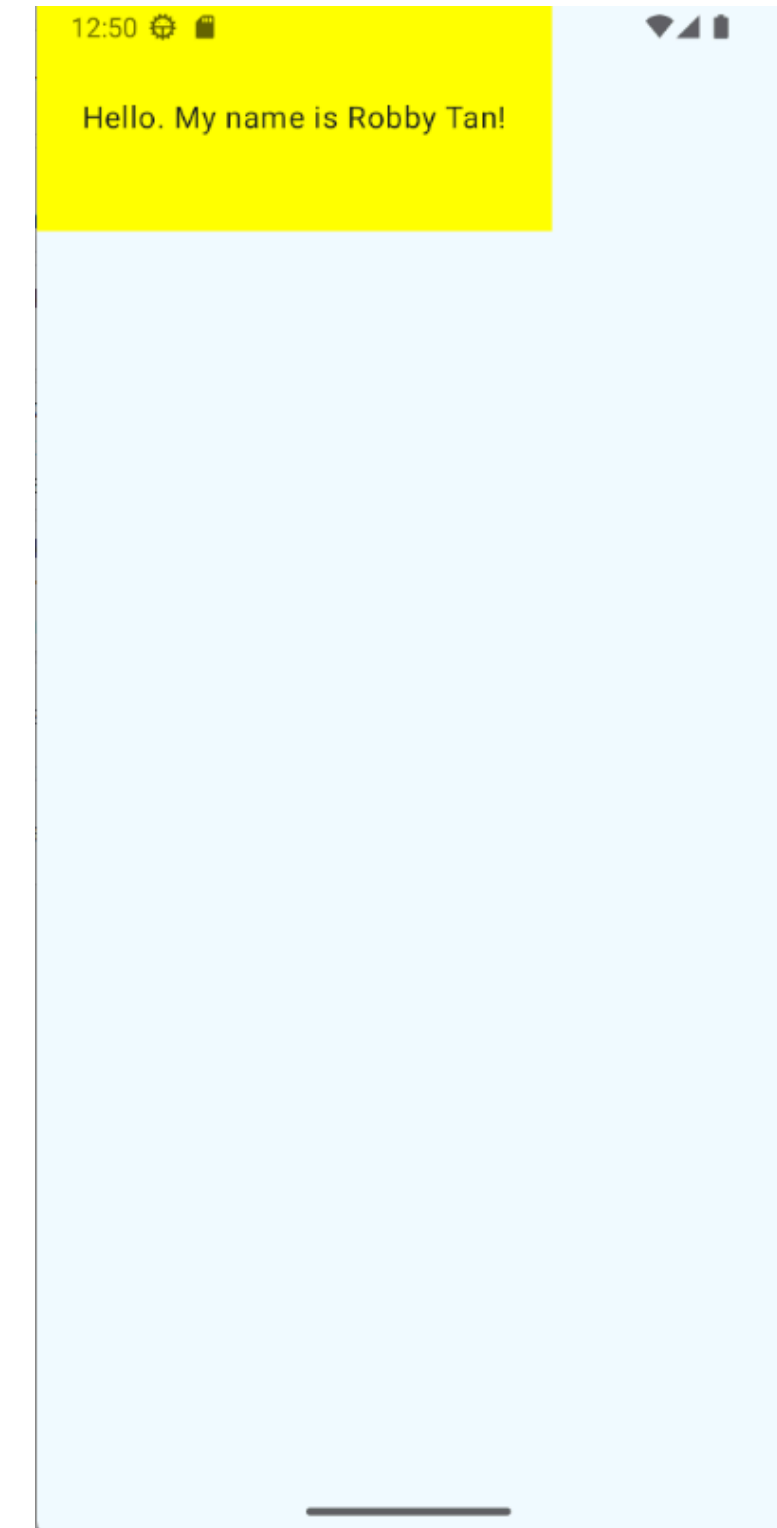
- Note:
 - dp berada pada *package* **androidx.compose.ui.unit.dp**
- Penulisan padding
 - 24.dp >> all padding (start, top, end, bottom)
 - 24.dp, 10.dp >> horizontal and vertical
 - 8.dp, 9.dp, 10.dp, 11.dp >> start, top, end, bottom





Deployment

- Emulator
- Real device using phone or WiFi
 - Cable
 - install Google USB Driver
 - Activate developer mode >> USB debugging
 - Reference: <https://developer.android.com/codelabs/basic-android-kotlin-compose-connect-device#2>
 - WiFi
 - Same network
 - Activate developer mode >> Wireless debugging
 - Scan QR with phone
 - Reference: <https://developer.android.com/codelabs/basic-android-kotlin-compose-connect-device#4>





Thank You

123 ANYWHERE ST., ANY CITY