FLUTTER

Mohamed Fahmy

Content

- Dart
- flutter

Dart

- Datatypes
- Var And Dynamic
- Final And Const
- Operators
- Control flow statements
- Functions
- OOP ,OOD Core Concepts

Object-Oriented Programming (OOP)

- Class
- Object
- Encapsulation
- Abstraction
- Inheritance
- Polymorphism
- Method
- Property/Attribute
- Constructor
- Instance variable
- Static variable
- Instance method
- Static method
- Interface
- Abstract class
- Final class
- Final variable
- Final method

Object-Oriented Design (OOD)

- DRY (Don't Repeat Yourself)
- KISS (Keep It Simple, Stupid)
- YAGNI (You Ain't Gonna Need It)
- SOLID

SOLID

- Single Responsibility Principle (SRP)
- Open/Closed Principle (OCP)
- Liskov Substitution Principle (LSP)
- Interface Segregation Principle (ISP)
- Dependency Inversion Principle (DIP)

Datatypes

- Numbers (num , int , double)
- Booleans
- Strings
- Lists
- Maps
- Sets

Numbers

- int: integers like 1, 2, 3

```
int age = 30;
```

- double: floating-point numbers like 3.14, 2.0

```
double height = 1.75;
```

- num is a built-in abstract class in Dart that represents a number. It is a superclass of both int and double classes

```
num value1 = 5;
num value2 = 5.5;
```

Booleans, Strings

- bool: values true or false

```
bool isStudent = true;
```

- String: a sequence of characters like "Hello, World!"

```
String name = 'dart';
print('hello ya $name ');
```

Runes

Runes: Represents a sequence of Unicode characters.

```
Runes heart = Runes('\u2665’);
print(String.fromCharCodes(heart)); // Output: ♥
```

var

```
var value; // var value = ‘’ // type > String

value = 5;
print(value.runtimeType); // Output: 5

value = 'Hello';
print(value.runtimeType); // Output: Hello

value = [1, 2, 3];
print(value.runtimeType); // Output: [1, 2,
3]
```

Dynamic

```
dynamic value;

value = 5;
print(value.runtimeType); // Output: 5

value = 'Hello';
print(value.runtimeType); // Output: Hello

value = [1, 2, 3];
print(value.runtimeType); // Output: [1, 2,
3]
```

Object

In Dart, Object is the root of the class hierarchy. Every class in Dart implicitly extends Object, meaning that every object in Dart has access to the methods defined in the Object class.

```
Object x = 0;
Object y = 'hi';
Object z = 1.9;
Object q = true;
print('$x , $y , $z , $q');
```

Lists, Maps, Sets

- List: a collection of values of the same type like [1, 2, 3]’

```
List<int> numbers = [1, 2, 3];
```

- Map: a collection of key-value pairs like {'name': 'John', 'age': 30}

```
Map<String, dynamic> person;
 person = {'name': 'John', 'age': 30, 'isStudent': true};
```

- Set: an unordered collection of unique values like {1, 2, 3}

```
Set<String> fruits = {'apple', 'banana', 'orange'};
```

final

final is a **runtime constant**. This means that it can only be assigned a value once, and that value is determined at runtime. When a final variable is assigned, it can be assigned any value, including one that is not known until runtime.

```
final x = DateTime.now(); // OK
```

const

const is a **compile-time constant**. This means that it must be assigned a value at compile-time, and that value must be known at that time. This means that a const variable can only be assigned a value that is a compile-time constant, such as a literal or a const expression.

```
const y = DateTime.now(); // Error: DateTime.now() is not a compile-time constant
const y = 11;
```

operators

- Arithmetic operators
- Comparison operators
- Logical operators
- Assignment operators
- type test operators
- type cast operators

Arithmetic operators

- + addition operator
- - subtraction operator
- * multiplication operator
- / division operator
- % modulo operator

Example

```
int x = 5;
int y = 3;

print(x + y); // Output: 8
print(x - y); // Output: 2
print(x * y); // Output: 15
print(x / y); // Output: 1.6666666666666667
print(x % y); // Output: 2
print(x++);   //Output: ?
print(x--);   //Output: ?
print("a" * 2); // “aa”
```

Comparison operators

- == equality operator
- != inequality operator
- > greater than operator
- < less than operator
- >= greater than or equal to operator
- <= less than or equal to operator

Example

```
int x = 5;
int y = 3;

print(x == y); // Output: false
print(x != y); // Output: true
print(x > y); // Output: true
print(x < y); // Output: false
print(x >= y); // Output: true
print(x <= y); // Output: false
```

Logical operators:

- && logical AND operator
- || logical OR operator
- ! logical NOT operator

Example

```
bool x = true;
bool y = false;

print(x && y); // Output: false
print(x || y); // Output: true
print(!x); // Output: false
print(!y); // Output: true
```

Assignment operators

- = simple assignment operator
- += addition assignment operator
- -= subtraction assignment operator
- *= multiplication assignment operator
- /= division assignment operator
- %= modulo assignment operator

Example

```
x += y; // Equivalent to x = x + y;
print(x); // Output: 13

x -= y; // Equivalent to x = x - y;
print(x); // Output: 10

x *= y; // Equivalent to x = x * y;
print(x); // Output: 30

x /= y; // Equivalent to x = x / y;
print(x); // Output: 10

x %= y; // Equivalent to x = x % y;
print(x); // Output: 1
```

Type test operators

- Is

```
dynamic message = 'Hello, World!';
if (message is String) {
  print(‘message is a String');
} else {
  print(‘message is not a String');
}
```

- Is!

```
dynamic message = 'Hello, World!';
  if (message is! int) {
    print('message is not a Int');
  }
```

type cast operators

- Type cast operators (as) allow you to convert an object from one type to another. If the object is not an instance of the specified type, a TypeError is thrown.

- as

```
num message = 1;
int number = message as int;
print(number);

Object? obj = "hello";
String str = obj as String;
print(str);
```

Control flow statements

- If-else statement
- Switch statement
- For loop
- While loop
- Do-while loop
- Break statement
- Continue statement

If-else statement

- The if-else statement allows developers to execute a block of code if a certain condition is true, and another block of code if the condition is false

- Example :

```
if (x > 10) {
    print('x is greater than 10');
  } else {
    print('x is less than or equal to 10');
  }
```

If-else shorthand syntax

- Example2 : (inline If)

condition ? expression1 : expression2

```
int number = 2 > 3 ? 5 : 6;
```

- Example2 : (If null)

expression1 ?? expression2

```
String? name;
String displayName = name ?? 'Guest';

print(displayName); // Output: Guest

bool isMale = true;
 if (!isMale) {
   print("Female");
 } else {
   print("male");
 }
 String gen = isMale ? "MALE" : "FEMALE";
```

Switch statement

- The switch statement allows developers to execute different blocks of code based on the value of a variable.
- Example :

```
var day = 'Monday';
  switch (day) {
    case 'Monday':
      print('Today is Monday');
      break;
    case 'Tuesday':
      print('Today is Tuesday');
      break;
    default:
      print('Today is not Monday or Tuesday');
  }
```

For loop

- The for loop is used to execute a block of code repeatedly for a specified number of times.
- Example :

```
for (var i = 0; i < 10; i++) {
  print('The value of i is $i');
}
```

```
var i = 0;
  var x = true;
  for (; x;) {
    x = i < 10;
    print(i);
    i++;
  }
```

Foreach vs map

```
List<int> myList = [1, 2, 3];

  myList.forEach((element) {
    print(element);
  });

  myList.map((element) {
    print(element); // not work , need return
  });


var x = myList.firstWhere((element) => element == 2);
  print(x);
```

While loop

- The while loop is used to execute a block of code repeatedly as long as a certain condition is true.

- Example :

```
var i = 0;
  while (i < 10) {
    print('The value of i is $i');
    i++;
  }
```

Do-while loop

- The do-while loop is used to execute a block of code repeatedly at least once, and then as long as a certain condition is true.

- Example :

```
var i = 0;
  do {
    print('The value of i is $i');
    i++;
  } while (i < 10);
```

Break statement

- The break statement is used to exit a loop or a switch statement.

- Example :

```
for (var i = 0; i < 10; i++) {
   if (i == 5) {
     break;
   }
   print('The value of i is $i');
 }
```

Continue statement

- The continue statement is used to skip the current iteration of a loop and move on to the next iteration.

- Example :

```
for (var i = 0; i < 10; i++) {
  if (i == 5) {
    continue;
  }
  print('The value of i is $i');
}
```

Functions

- Defining a Function
- Calling a Function
- Positional Optional Parameters
- Named Optional Parameters
- Higher-order functions

Defining a Function And Calling a Function

- Example

```
int sum(int a, int b) {
  return a + b;
}

int result = sum(3, 4);
print(result); // 7
```

Positional Optional Parameters

- Positional Optional Parameters:

```
int sum(int a, [int? b, int? c]) {
  if (b != null) {
    if (c != null) {
      return a + b + c;
    }
    return a + b;
  } else {
    return a;
  }
}

// sum(5)
print(sum(5));
print(sum(5, 5));
print(sum(5, 5, 5));
```

Named Optional Parameters

- Named Parameters:

```
int sum({int a = 0, int b = 0}) {
   return a + b;
 }

 int result1 = sum(); // 0
 int result2 = sum(a: 3); // 3
 int result3 = sum(b: 4); // 4
 int result4 = sum(a: 2, b: 5); // 7
 print(result1);
 print(result2);
 print(result3);
 print(result4);
```

Named Optional Parameters

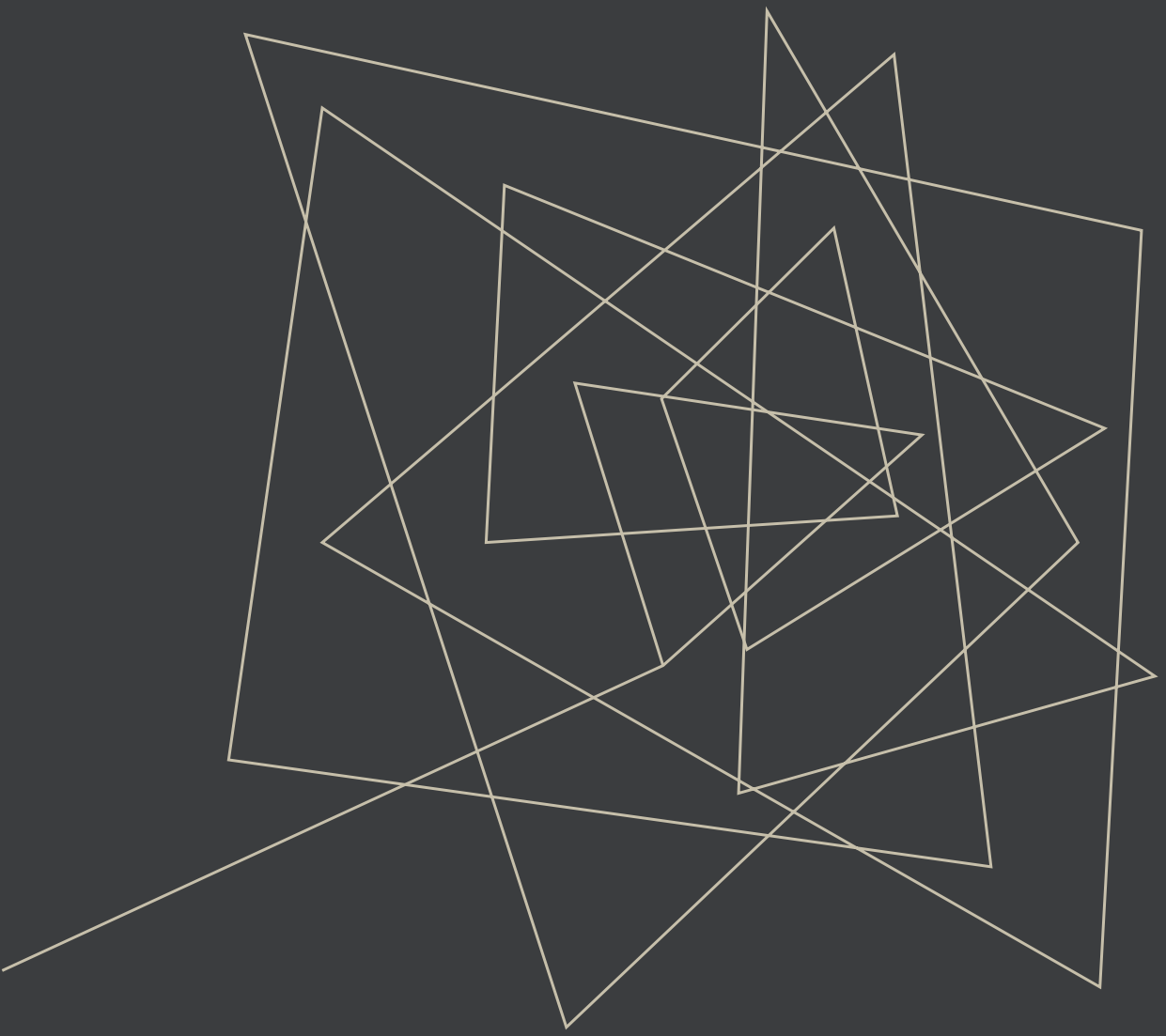
- Named Parameters (with required):

```
int sum(int c, {int a = 0, int b = 0}) {
  print(c);
  return a + b;
}

int result1 = sum(1); // 0
int result2 = sum(1,a: 3); // 3
int result3 = sum(2,b: 4); // 4
int result4 = sum(3,a: 2, b: 5); // 7
print(result1);
print(result2);
print(result3);
print(result4);
```

Higher-order functions

- a higher-order function is a function that takes one or more functions as parameters, or returns a function as its result. This is possible because functions in Dart are first-class objects, meaning they can be treated like any other value or object.

```
void looping(int times, action) {
  for (int i = 0; i < times; i++) {
    action();
  }
}

looping(5, () => {print('hello ya dart')});
```



OOP,OOD

Object-Oriented Programming (OOP)

- Class
- Object
- Encapsulation
- Abstraction
- Inheritance
- Polymorphism
- Method
- Property/Attribute
- Constructor
- Instance variable
- Static variable
- Instance method
- Static method
- Interface
- Abstract class
- Final class
- Final variable
- Final method

definitions of main concepts of Object-Oriented Programming (OOP)

1. **Class**: A blueprint or template for creating objects that encapsulate data and behavior.
2. **Object**: An instance of a class that has its own set of data and behavior.
3. **Encapsulation**: The practice of hiding the implementation details of an object and exposing only the necessary information through well-defined interfaces.
4. **Abstraction**: The process of simplifying complex systems by focusing on the essential features and ignoring the non-essential details.
5. **Inheritance**: The process of creating a new class from an existing class, where the new class inherits the properties and methods of the existing class.
6. **Polymorphism**: The ability of an object to take on multiple forms or types.
7. **Method**: A function that is defined inside a class and can be called on an object of that class.
8. **Property/Attribute**: A variable that is defined inside a class and belongs to each instance of that class.
9. **Constructor**: A special method that is called when an object is created to initialize its properties.
10. **Instance variable**: A variable that belongs to each instance of a class and can have different values for each instance.
11. **Static variable**: A variable that belongs to the class itself rather than to any instance of the class.
12. **Instance method**: A method that operates on the properties of an individual object.
13. **Static method**: A method that operates on the class itself rather than on any individual object.
14. **Interface**: A contract that specifies a set of methods and properties that a class must implement.
15. **Abstract class**: A class that cannot be instantiated and must be subclassed to be used.
16. **Final class**: A class that cannot be subclassed.
17. **Final variable**: A variable that can be assigned a value only once and cannot be changed thereafter.
18. **Final method**: A method that cannot be overridden by a subclass.

Object-Oriented Design (OOD)

- DRY (Don't Repeat Yourself)
- KISS (Keep It Simple, Stupid)
- YAGNI (You Ain't Gonna Need It)
- SOLID

SOLID

- Single Responsibility Principle (SRP)
- Open/Closed Principle (OCP)
- Liskov Substitution Principle (LSP)
- Interface Segregation Principle (ISP)
- Dependency Inversion Principle (DIP)

Don't Repeat Yourself (DRY)

- Example:

```
//Without DRY

  void displayGreetingInEnglish() {
    print("Hello!");
  }

  void displayGreetingInSpanish() {
    print("Hola!");
  }

//With DRY
  void displayGreeting(String greeting) {
    print(greeting);
  }

  displayGreeting("Hello!"); //prints Hello!
  displayGreeting("Hola!"); //prints Hola!
```

Keep It Simple, Stupid (KISS)

- Example

```
//Without KISS
  void multiplyAndPrint(List<int> numbers) {
    var result = 1;
    for (var i = 0; i < numbers.length; i++) {
      result *= numbers[i];
    }
    print("The result is: $result");
  }

//With KISS
  void multiplyAndPrint(List<int> numbers) {
    var result = numbers.reduce((a, b) => a * b);
    print("The result is: $result");
  }

  multiplyAndPrint([2, 3, 4]);  //result is: 24
```

You Ain't Gonna Need It (YAGNI)

- Example

```
//Without YAGNI
class Person {
  String? name;
  String? address;
  String? phone;
  String? email;
  String? occupation;
  int? age;
  //...and so on
}
```

```
//With YAGNI
class Person {
  String? name;
  String? email;
  int? age;
}
```

```
var person = Person();
person.name = “Ahmed";
person.email = “aa@aa.com";
person.age = 30;
```

Class

- a class is a blueprint or template for creating objects that encapsulate data and behavior

- Example

```
class Person {
  String? name;
  int? age;

  void sayHello() {
    print('Hello, my name is $name and I am $age years old.');
  }
}


Final class => Person._();
```

constructor
constructor chaining
Super keyword

- Content

```
class Animal {
  String name;
  Animal(this.name) {
    print("name from Animal Constructor is
$name");
  }
}

class Dog extends Animal {
  int? age;
  Dog(String name, this.age) : super(name) {
    print('dog constructor');
  }
}

Dog rex = Dog('rex', 2);
```

Object

- an object is an instance of a class. It represents a specific thing or entity in your program, such as a person, a car, or a bank account. An object has state, which is the set of values that define its properties or attributes, and behavior, which is the set of actions that it can perform.

- Example

```
Person person1 = Person();
person1.name = ‘mohamed';
person1.age = 25;
person1.sayHello();
// Output: "Hello, my name is mohamed and I am 25 years old."

Person person2 = Person();
person2.name = ‘ahmed';
person2.age = 30;
person2.sayHello();
// Output: "Hello, my name is ahmed and I am 30 years old."
```

Encapsulation

- encapsulation is the practice of hiding the implementation details of an object
- Example

```
class Person {
  int _age = 0; // private variable
  void _validateAge() {
    if (_age < 0 || _age > 120) {
      throw Exception('Invalid age');
    }
  }

  void setAge(int age) {
    _age = age;
    _validateAge();
  }

  int getAge() {
    return _age;
  }

  void sayHello() {
    print('I am $_age years old.');
  }
}
```

Encapsulation

- Using

```
Person person1 = new Person();
person1.setAge(25);
person1.sayHello();

person1._name = 'mohamed’;
person1.getAge();
```

Static

- In Dart, the static keyword is used to create class-level variables and methods

```
class MyClass {
  static int myStaticVariable = 0;

  static void myStaticMethod() {
    print('This is a static method');
  }
}


MyClass.myStaticMethod(); // This is a static method
print(MyClass.myStaticVariable);
```

Abstraction, Interface

- abstraction is the practice of focusing on the essential features of an object or system and hiding unnecessary details. Abstraction can be used to simplify complex systems and make them more manageable by breaking them down
into smaller, more manageable pieces.

```
abstract class Animal {
  void speak();
  void move();
}
```

Extends VS implements

```
class Dog implements Animal {
  @override
  void speak() {
    print('Woof!');
  }

  @override
  void move() {
    print('Running');
  }
}
```

Inheritance

- allows one class to inherit properties and methods from another class

```
class Person {
  String? name;
  int? age;

  void greet() {
    print('Hello, my name is $name and I am $age
years old.');
  }
}

class Student extends Person {
  int? grade;

  void study() {
    print('$name is studying hard for grade
$grade.');
  }
}
```

Inheritance

- allows one class to inherit properties and methods from another class

- Example

```
Student student = Student();
student.name = 'Ahmed';
student.age = 18;
student.grade = 12;

student.greet();
// Output: Hello, my name is Ahmed and I am 18 years old.
student.study();
// Output: Alice is studying hard for grade 12.
```

Polymorphism

- allows objects of different classes to be treated as if they were objects of the same class

- Overriding

```
class Animal {
  void makeSound() {
    print('Animal makes a sound');
  }
}

class Dog extends Animal {
  @override
  void makeSound() {
    print('Dog barks');
  }
}

class Cat extends Animal {
  @override
  void makeSound() {
    print('Cat meows');
  }
}
```

Polymorphism

- allows objects of different classes to be treated as if they were objects of the same class

- Overriding

```
Animal animal = Animal();
Dog dog = Dog();
Cat cat = Cat();

animal.makeSound(); // Output: Animal makes a sound
dog.makeSound(); // Output: Dog barks
cat.makeSound(); // Output: Cat meows
```

Polymorphism

- Overloading

In Dart, **overloading is not supported directly**, but you can achieve similar behavior by using named optional parameters.

Alternatively, you can achieve similar behavior
by using named optional parameters.
Here's an example:

```
class Calculator {
  int add(int a, int b, {int? c, int? d}) {
    if (c != null && d != null) {
      return a + b + c + d;
    } else if (c != null) {
      return a + b + c;
    } else {
      return a + b;
    }
  }
}
```

Polymorphism

- Overloading

In Dart, overloading is not supported directly, but you can achieve similar behavior by using named optional parameters.

Alternatively, you can achieve similar behavior
by using named optional parameters.
Here's an example:

```
Calculator calculator = Calculator();

int sum1 = calculator.add(2, 3);
int sum2 = calculator.add(2, 3, c: 4);
int sum3 = calculator.add(2, 3, c: 4, d: 5);

print('Sum 1: $sum1'); // Output: Sum 1: 5
print('Sum 2: $sum2'); // Output: Sum 2: 9
print('Sum 3: $sum3'); // Output: Sum 3: 14
```

Abstraction, Interface

- abstraction is the practice of focusing on the essential features of an object or system and hiding unnecessary details. Abstraction can be used to simplify complex systems and make them more manageable by breaking them down into smaller, more manageable pieces.

```
abstract class Animal {
  void speak();
  void move();
}
```

Extends VS implements

```
class Dog implements Animal {
  @override
  void speak() {
    print('Woof!');
  }

  @override
  void move() {
    print('Running');
  }
}
```

mixin

```
mixin LoggerMixin {
  void log(String message) {
    print('[LoggerMixin] $message');
  }
}

class MyClass with LoggerMixin {
  void doSomething() {
    log('Doing something...');
    // Rest of the code
  }
}

void main() {
  var myObj = MyClass();
  myObj.doSomething();
}
```

mixin

```
mixin Breathing {
  void swim() => print("Breathing");
}

mixin Walking {
  void walk() => print("Walking");
}

mixin Coding {
  void code() => print("print('Hello world!')");
}

/// This class now has the `walk()` method
class Human with Walking {}

/// This class now has the `walk()` and `code()`
methods
class Developer with Walking, Coding {}
```

Async, await, Future

Async*

```
Future<void> main(List<String> args) async {
  final results = await Client()
      .get(Uri.parse('link'))
      .then((value) => print(value.body))
      .catchError((e) => prints(e));
```

Search for **Isolates and Event Loops in dart**

Stream

```
final myPeriodicStream = Stream.periodic(const
Duration(seconds: 1));
  final subscription = myPeriodicStream.listen((event) {
    print("asd");
  });
  await Future.delayed(const Duration(seconds: 4));
  subscription.cancel();
```

Search for **Isolates and Event Loops in dart**

https://medium.flutterdevs.com/event-loop-in-dart-226a7487b4aa

Imperative programming and declarative programming

- **Imperative** programming is a programming paradigm that focuses on describing **how** the program should achieve a particular task

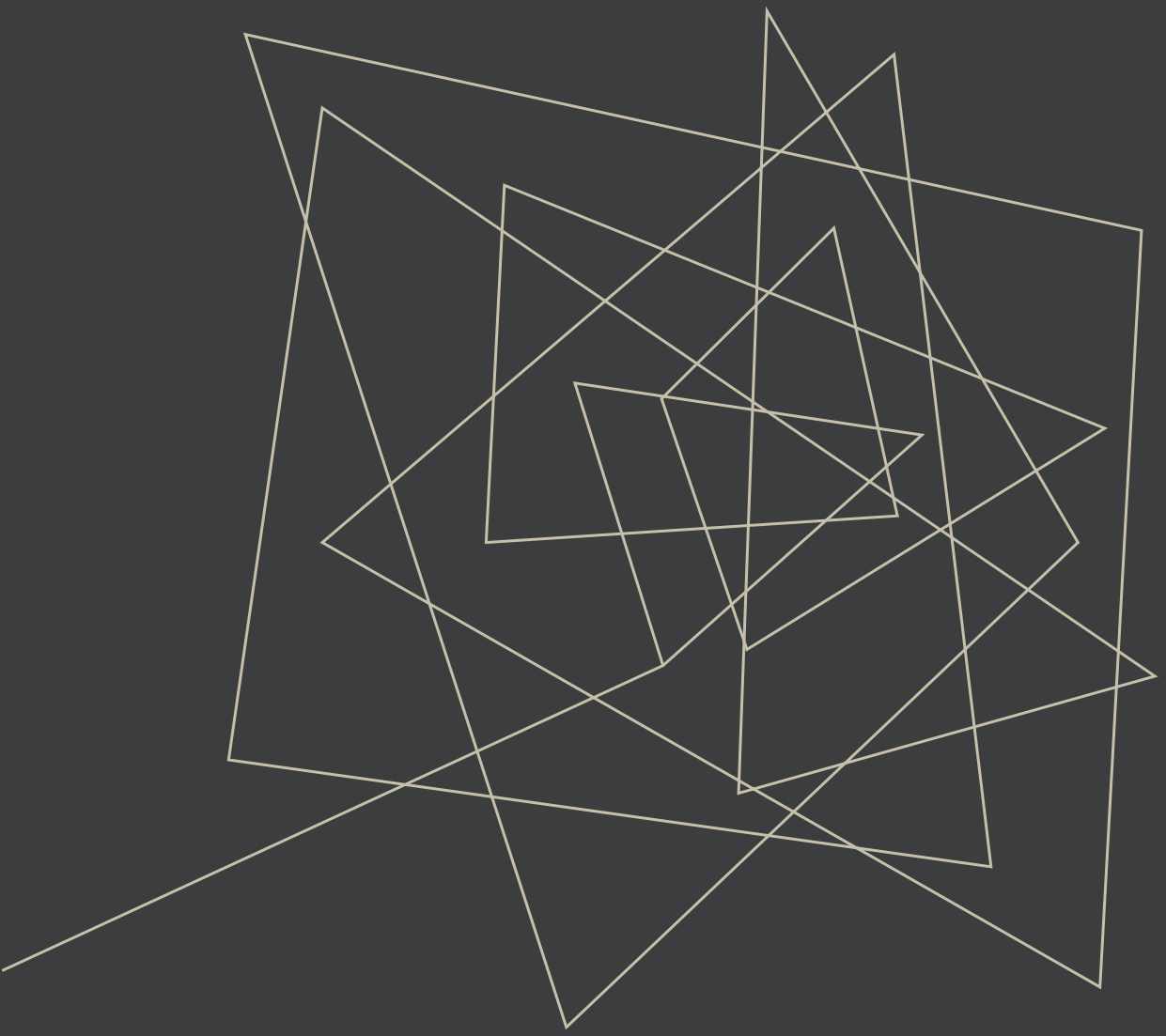
```
void printEvenNumbers(int n) {
  for (int i = 0; i < n; i++) {
    if (i % 2 == 0) {
      print(i);
    }
  }
}
```

- declarative programming is a programming paradigm that focuses on **describing what the program should achieve**, without specifying how it should achieve

```
void printEvenNumbers(int n) {
   List<int> numbers = List.generate(n, (index) => index);
   numbers.where((number) => number % 2 == 0).forEach(print);
 }
```

Flutter introduction

- Introduction to mobile development
- Install and setup Environment
- Make first app using flutter
- File and folder structure
- Basic widgets in flutter
- Statefull vs Stateless Widgets
- Counter app



INSTALL AND SETUP ENVIRONMENT

https://developer.android.com/studio#command-tools

developer.android.com/studio#command-tools

Gmail YouTube Maps

developers Platform Android Studio Google Play Jetpack Kotlin More Search Language Sign in

ANDROID STUDIO

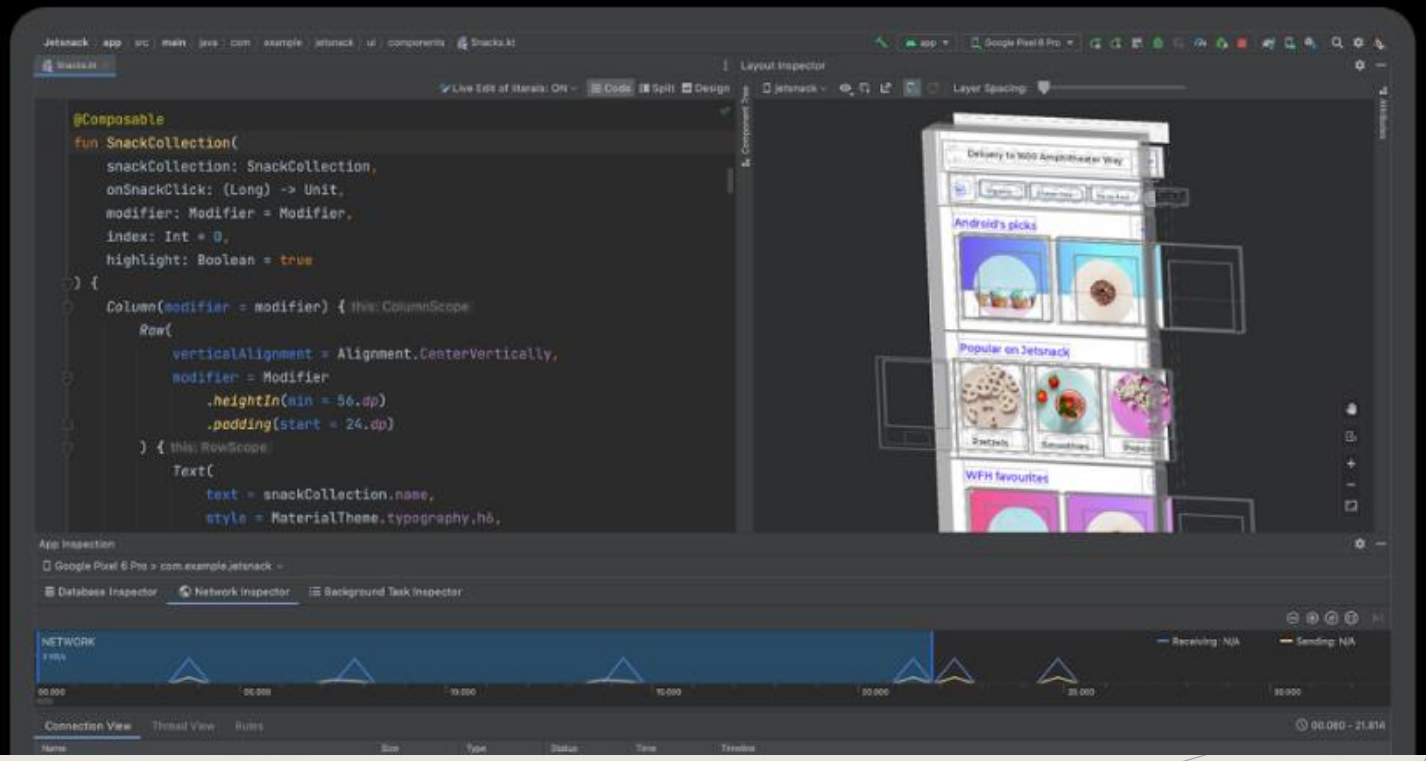
Download What's new User guide Preview

Android Studio

Get the official Integrated Development Environment (IDE) for Android app development.

Download Android Studio Electric Eel

Read release notes



developer.android.com/studio#command-tools

Gmail YouTube Maps

developers Platform Android Studio Google Play Jetpack Compose Kotlin More Search Language Sign in

ANDROID STU

Download

14.1 The License Agreement constitutes the whole legal agreement between you and Google and governs your use of the SDK (excluding any services which Google may provide to you under a separate written agreement), and completely replaces any prior agreements between you and Google in relation to the SDK. 14.2 You agree that if Google does not exercise or enforce any legal right or remedy which is contained in the License Agreement (or which Google has the benefit of under any applicable law), this will not be taken to be a formal waiver of Google's rights and that those rights or remedies will still be available to Google. 14.3 If any court of law, having the jurisdiction to decide on this matter, rules that any provision of the License Agreement is invalid, then that provision will be removed from the License Agreement without affecting the rest of the License Agreement. The remaining provisions of the License Agreement will continue to be valid and enforceable. 14.4 You acknowledge and agree that each member of the group of companies of which Google is the parent shall be third party beneficiaries to the License Agreement and that such other companies shall be entitled to directly enforce, and rely upon, any provision of the License Agreement that confers a benefit on (or rights in favor of) them. Other than this, no other person or company shall be third party beneficiaries to the License Agreement. 14.5 EXPORT RESTRICTIONS. THE SDK IS SUBJECT TO UNITED STATES EXPORT LAWS AND REGULATIONS. YOU MUST COMPLY WITH ALL DOMESTIC AND INTERNATIONAL EXPORT LAWS AND REGULATIONS THAT APPLY TO THE SDK. THESE LAWS INCLUDE RESTRICTIONS ON DESTINATIONS, END USERS AND END USE. 14.6 The rights granted in the License Agreement may not be assigned or transferred by either you or Google without the prior written approval of the other party. Neither you nor Google shall be permitted to delegate their responsibilities or obligations under the License Agreement without the prior written approval of the other party. 14.7 The License Agreement, and your relationship with Google under the License Agreement, shall be governed by the laws of the State of California without regard to its conflict of laws provisions. You and Google agree to submit to the exclusive jurisdiction of the courts located within the county of Santa Clara, California to resolve any legal matter arising from the License Agreement. Notwithstanding this, you agree that Google shall still be allowed to apply for injunctive remedies (or an equivalent type of urgent legal relief) in any jurisdiction. *July 27, 2021*

☑ I have read and agree with the above terms and conditions

Download Android Studio Electric Eel | 2022.1.1 Patch 1 for Windows

android-studio-2022.1.1.20-windows.exe

An

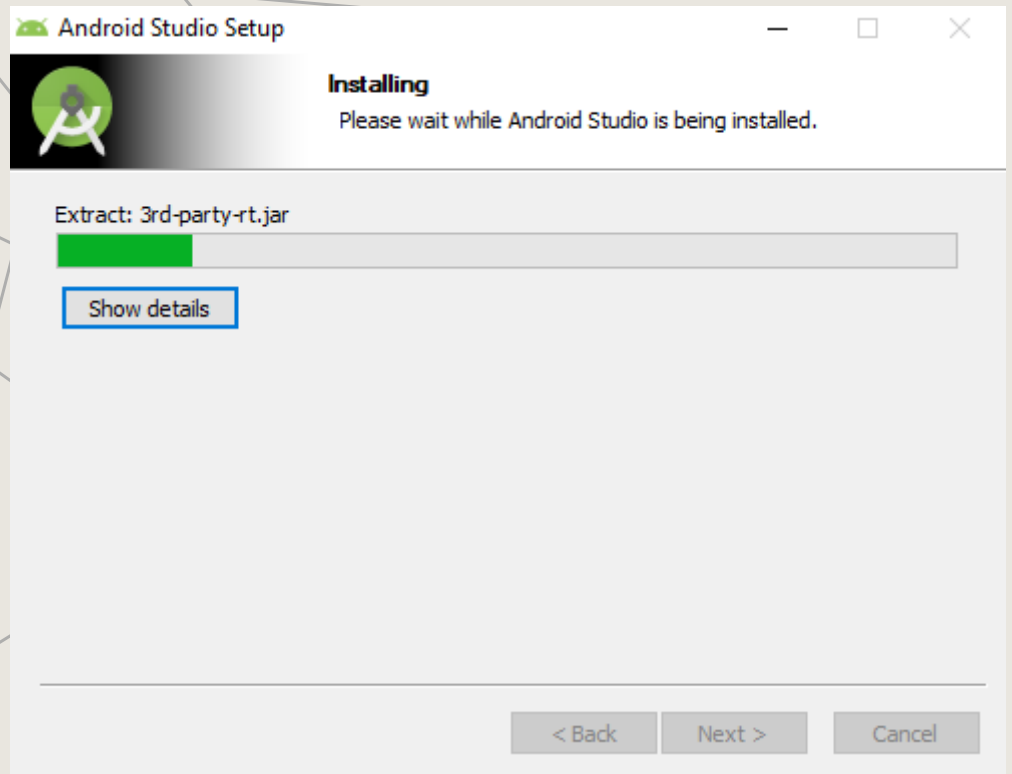
Get the

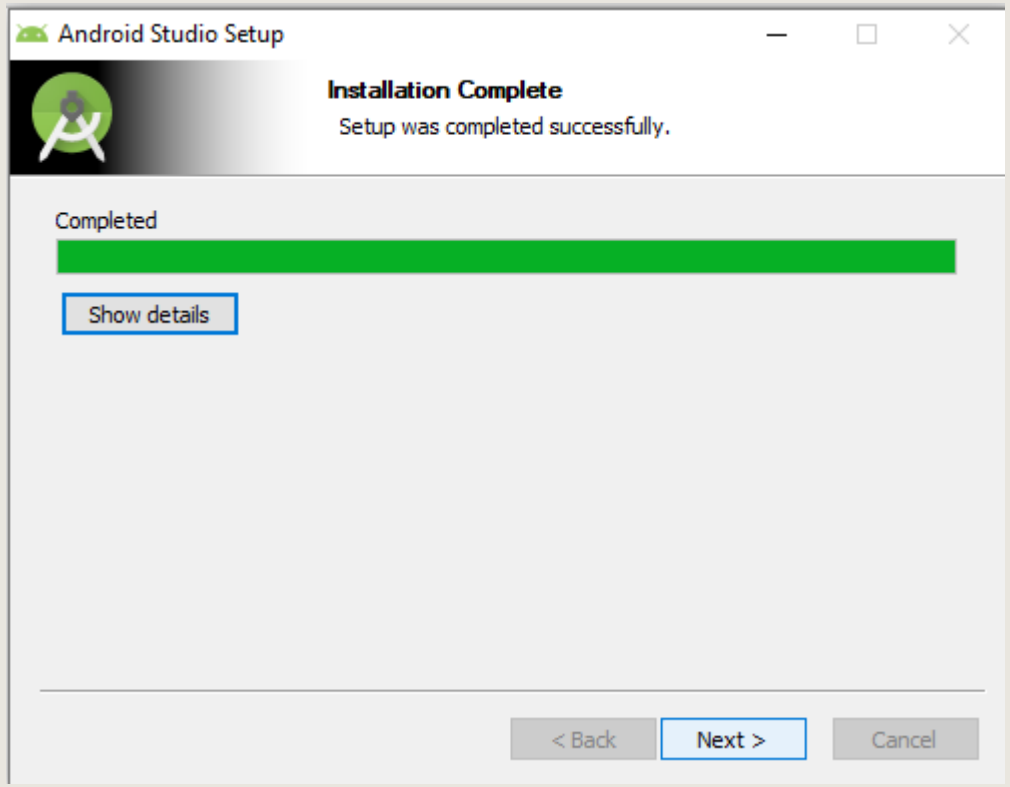
Environ

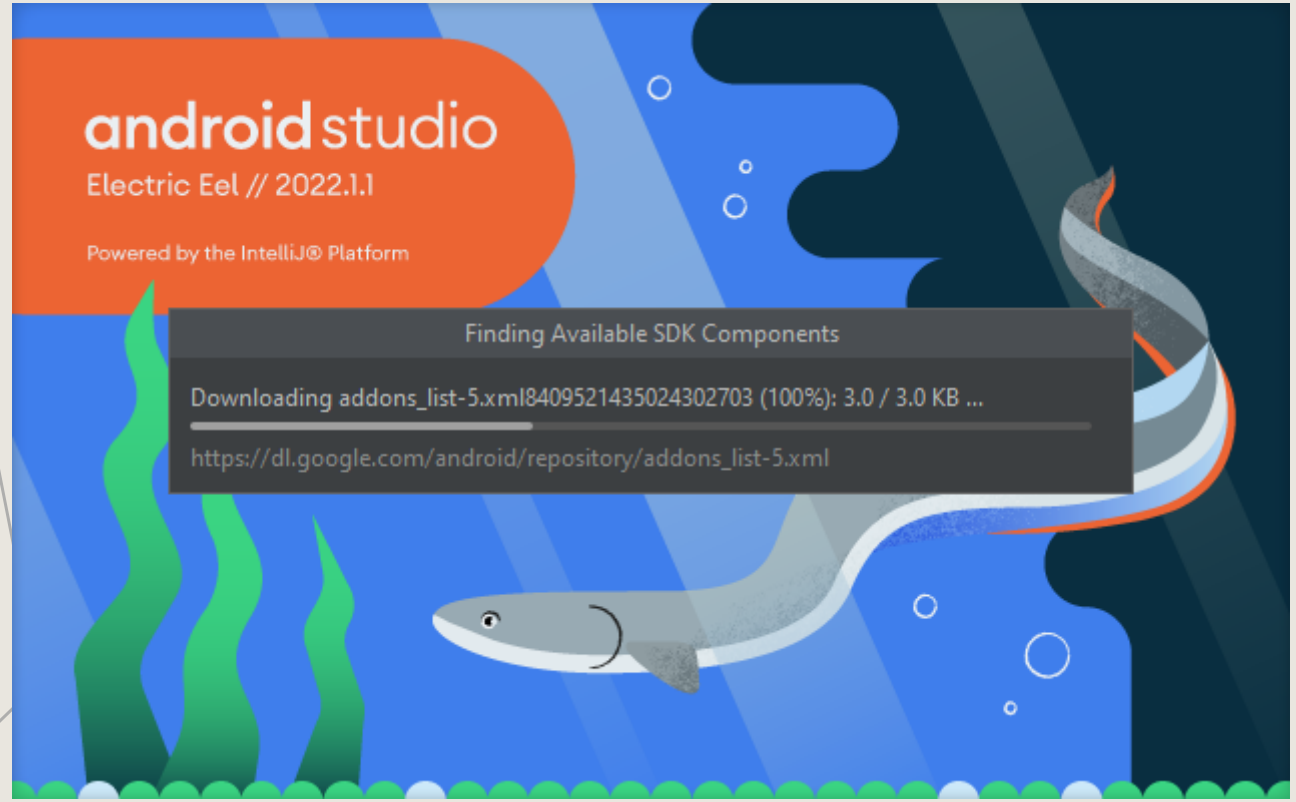
develop

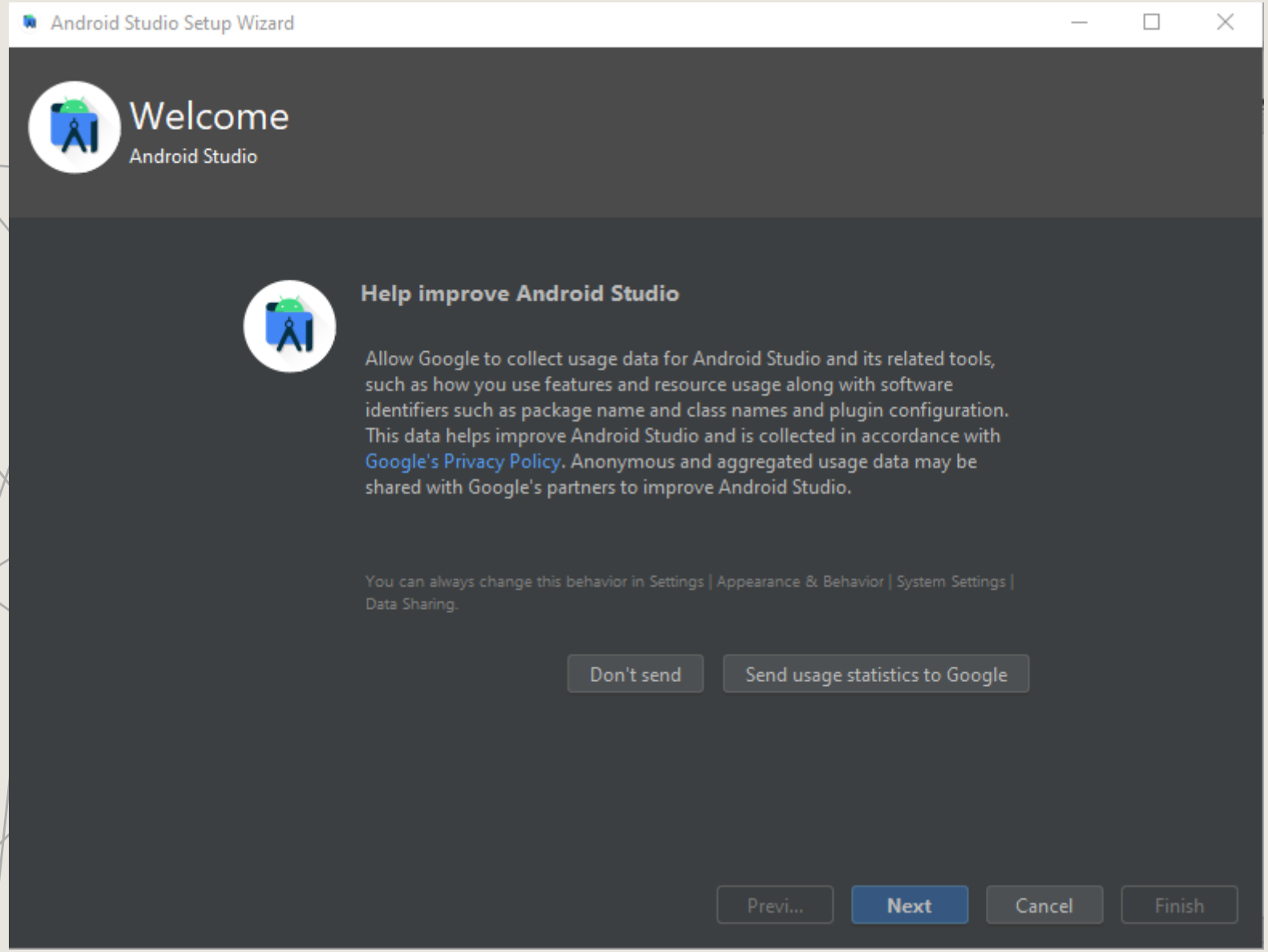
Downl

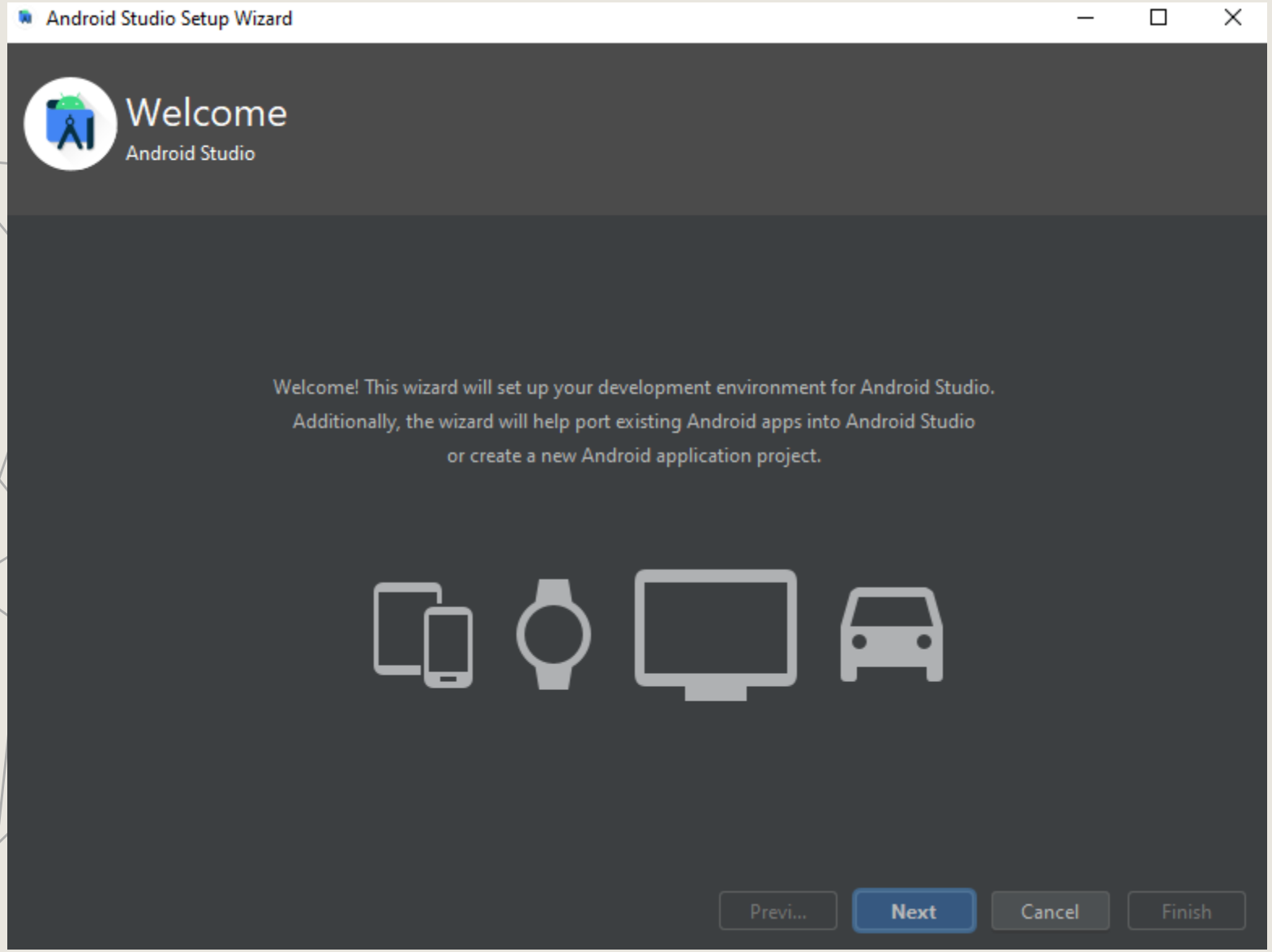
Read rel

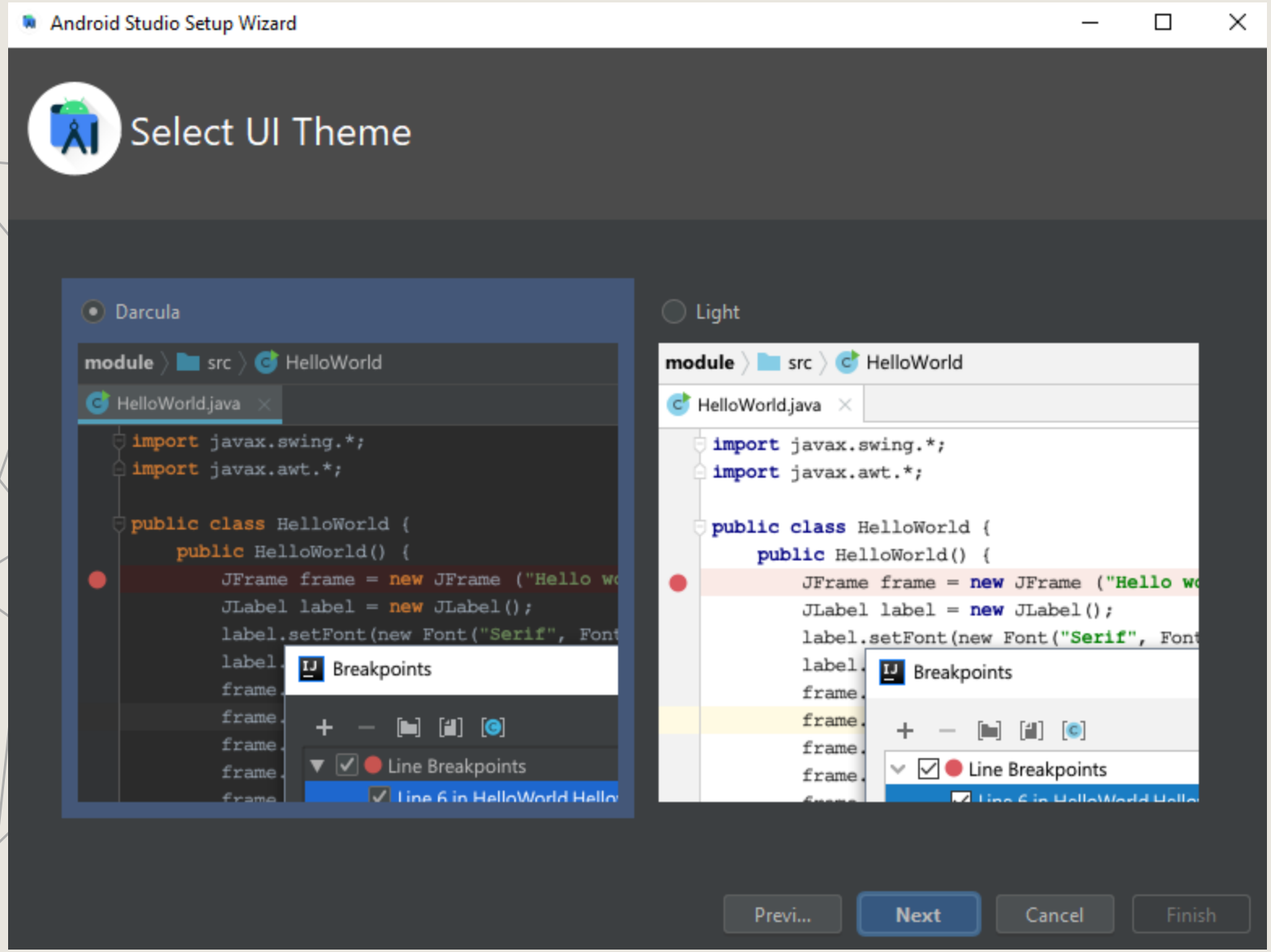

Android Studio Setup
Installing
Please wait while Android Studio is being installed.
Extract: 3rd-party-rt.jar
Show details
< Back
Next >
Cancel

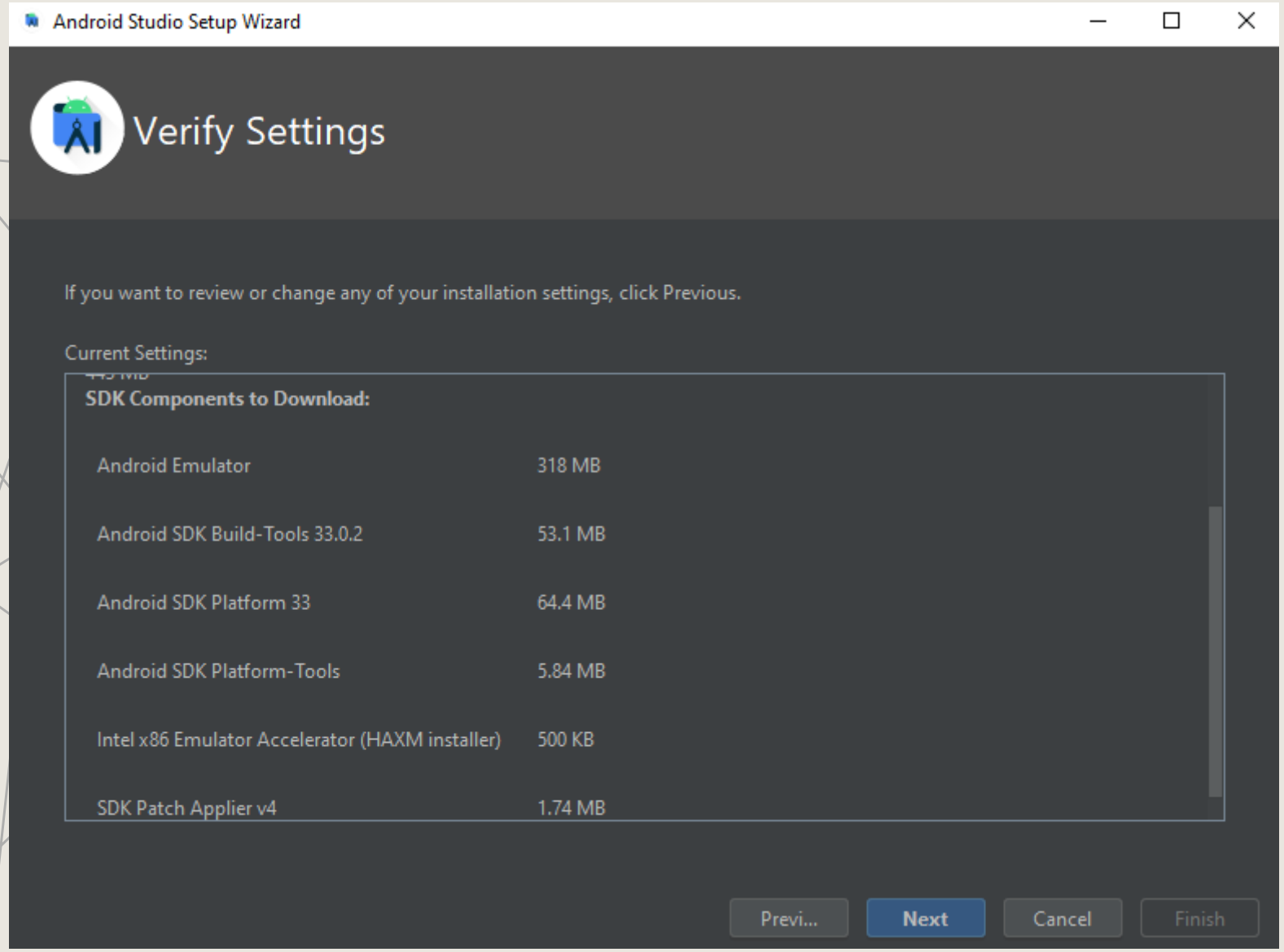

Android Studio Setup
Installation Complete
Setup was completed successfully.
Completed
Show details
< Back
Next >
Cancel

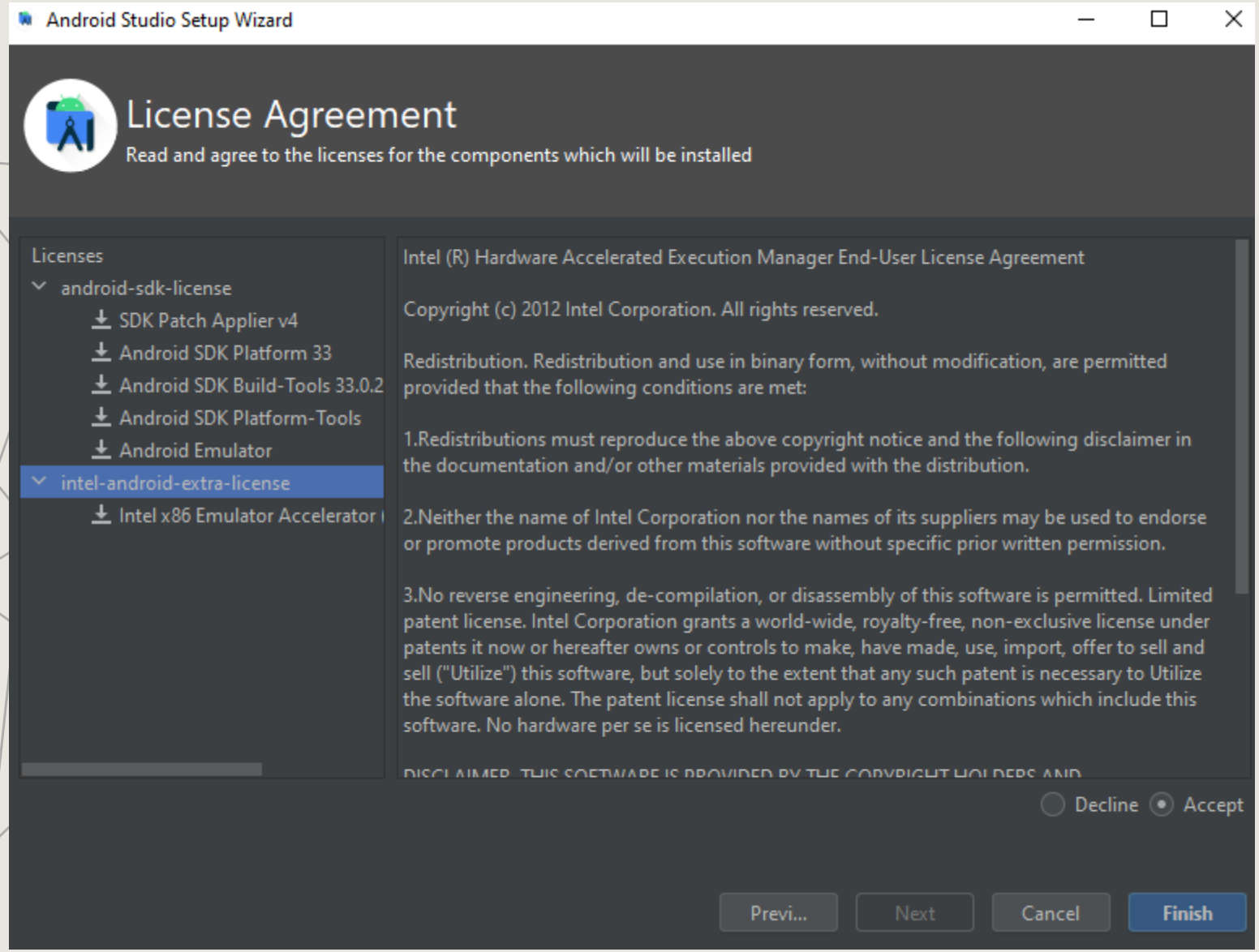

android studio
Electric Eel // 2022.1.1
Powered by the IntelliJ® Platform
Finding Available SDK Components
Downloading addons_list-5.xml8409521435024302703 (100%): 3.0 / 3.0 KB ...
https://dl.google.com/android/repository/addons_list-5.xml

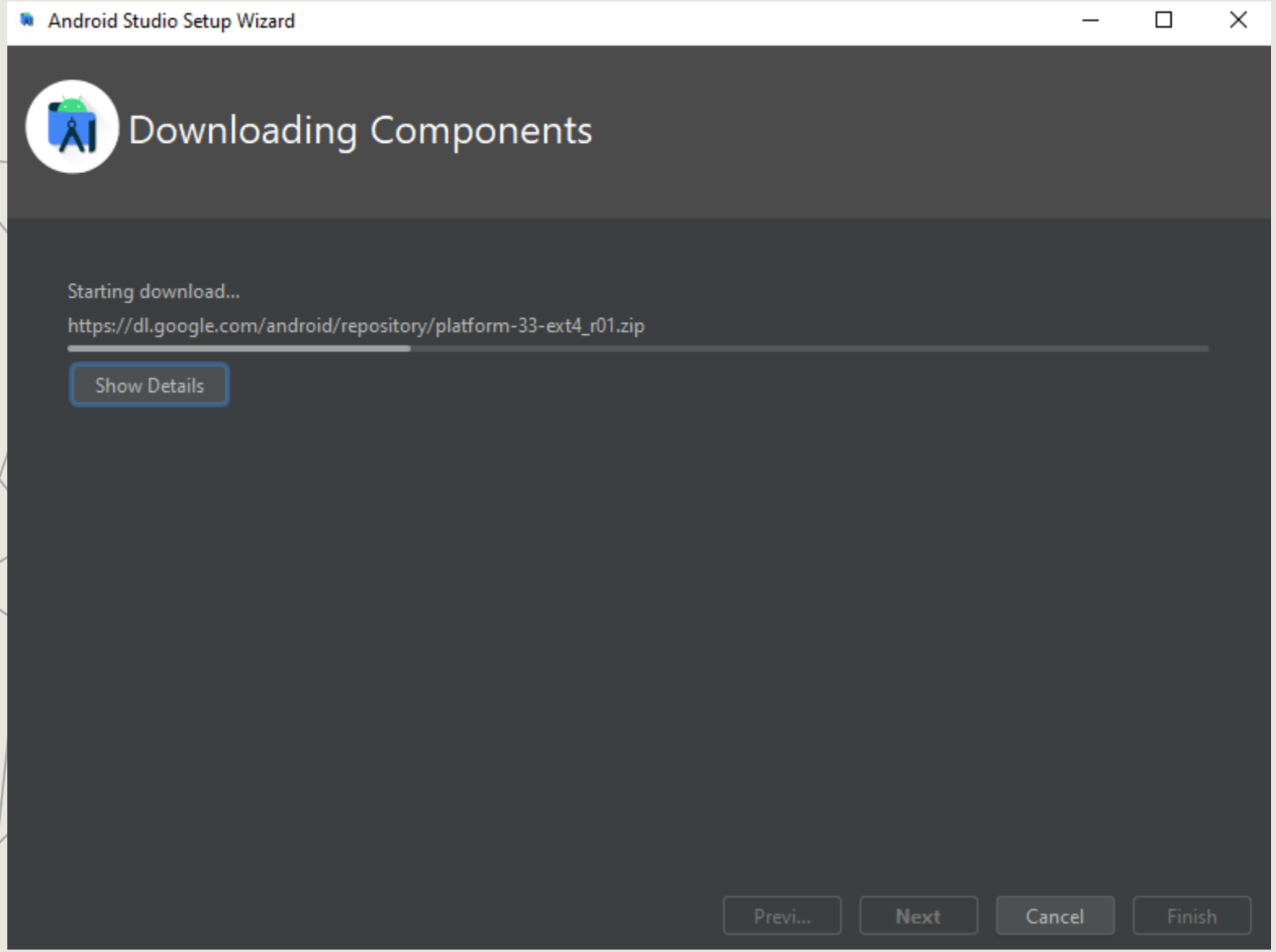

Android Studio Setup Wizard
Welcome
Android Studio
Help improve Android Studio
Allow Google to collect usage data for Android Studio and its related tools, such as how you use features and resource usage along with software identifiers such as package name and class names and plugin configuration. This data helps improve Android Studio and is collected in accordance with Google's Privacy Policy. Anonymous and aggregated usage data may be shared with Google's partners to improve Android Studio.
You can always change this behavior in Settings | Appearance & Behavior | System Settings | Data Sharing.
Don't send
Send usage statistics to Google
Previ...
Next
Cancel
Finish


Android Studio Setup Wizard
Welcome
Android Studio
Welcome! This wizard will set up your development environment for Android Studio.
Additionally, the wizard will help port existing Android apps into Android Studio
or create a new Android application project.
Previ...
Next
Cancel
Finish


Android Studio Setup Wizard
Select UI Theme
Darcula
module
src
HelloWorld
HelloWorld.java
import javax.swing.*;
import javax.awt.*;
public class HelloWorld {
public HelloWorld() {
JFrame frame = new JFrame ("Hello wo
JLabel label = new JLabel();
label.setFont(new Font("Serif", Font
Breakpoints
Line Breakpoints
Line 6 in HelloWorld.Hello
Light
module
src
HelloWorld
HelloWorld.java
import javax.swing.*;
import javax.awt.*;
public class HelloWorld {
public HelloWorld() {
JFrame frame = new JFrame ("Hello wo
JLabel label = new JLabel();
label.setFont(new Font("Serif", Font
Breakpoints
Line Breakpoints
Line 6 in HelloWorld.Hello
Previ...
Next
Cancel
Finish


Android Studio Setup Wizard
Verify Settings
If you want to review or change any of your installation settings, click Previous.
Current Settings:
SDK Components to Download:
Android Emulator 318 MB
Android SDK Build-Tools 33.0.2 53.1 MB
Android SDK Platform 33 64.4 MB
Android SDK Platform-Tools 5.84 MB
Intel x86 Emulator Accelerator (HAXM installer) 500 KB
SDK Patch Applier v4 1.74 MB
Previ...
Next
Cancel
Finish


Android Studio Setup Wizard
License Agreement
Read and agree to the licenses for the components which will be installed
Licenses
android-sdk-license
SDK Patch Applier v4
Android SDK Platform 33
Android SDK Build-Tools 33.0.2
Android SDK Platform-Tools
Android Emulator
intel-android-extra-license
Intel x86 Emulator Accelerator
Intel (R) Hardware Accelerated Execution Manager End-User License Agreement
Copyright (c) 2012 Intel Corporation. All rights reserved.
Redistribution. Redistribution and use in binary form, without modification, are permitted provided that the following conditions are met:
1.Redistributions must reproduce the above copyright notice and the following disclaimer in the documentation and/or other materials provided with the distribution.
2.Neither the name of Intel Corporation nor the names of its suppliers may be used to endorse or promote products derived from this software without specific prior written permission.
3.No reverse engineering, de-compilation, or disassembly of this software is permitted. Limited patent license. Intel Corporation grants a world-wide, royalty-free, non-exclusive license under patents it now or hereafter owns or controls to make, have made, use, import, offer to sell and sell ("Utilize") this software, but solely to the extent that any such patent is necessary to Utilize the software alone. The patent license shall not apply to any combinations which include this software. No hardware per se is licensed hereunder.
Decline
Accept
Previ...
Next
Cancel
Finish

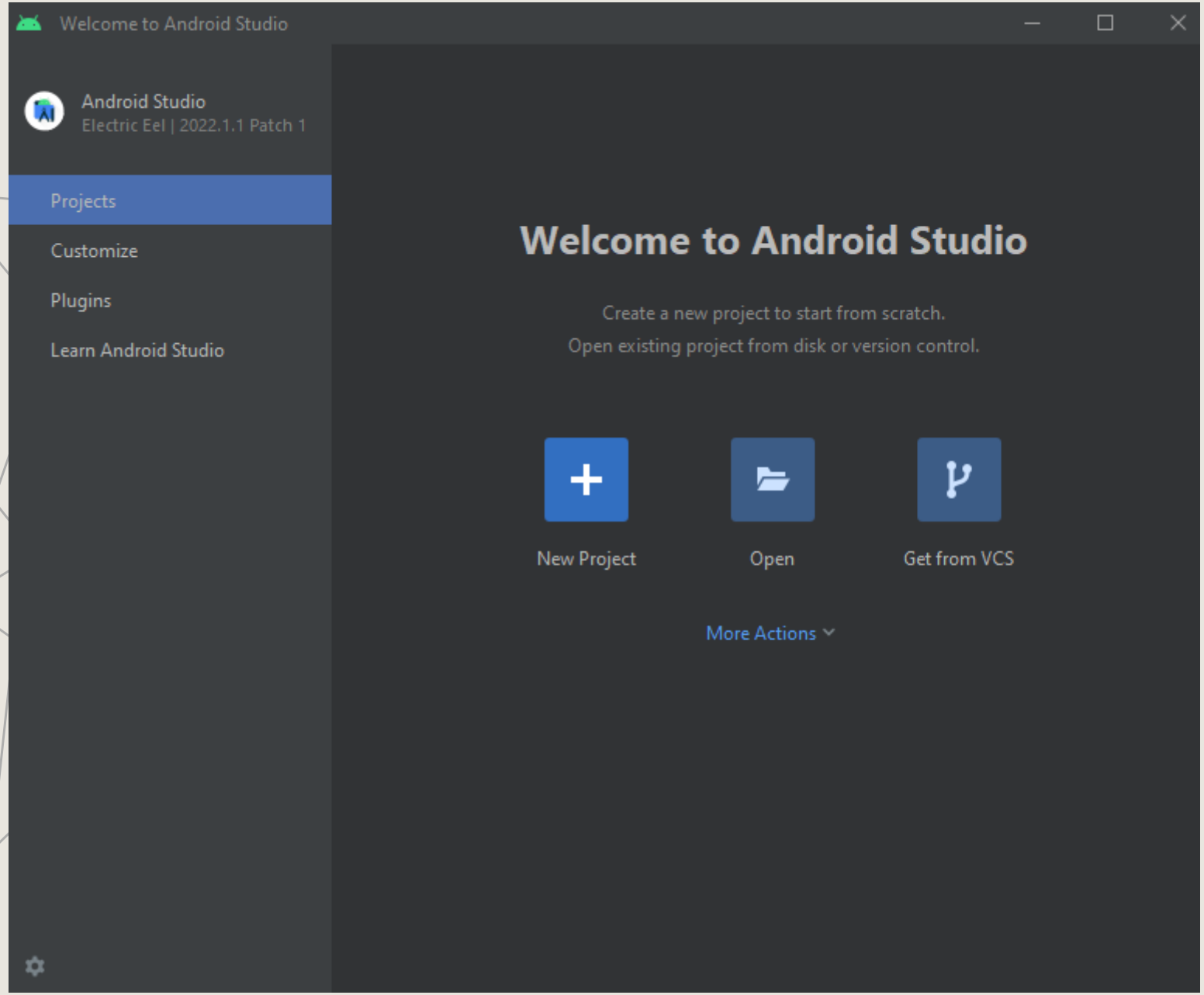

Android Studio Setup Wizard
Downloading Components
Starting download...
https://dl.google.com/android/repository/platform-33-ext4_r01.zip
Show Details
Previ...
Next
Cancel
Finish

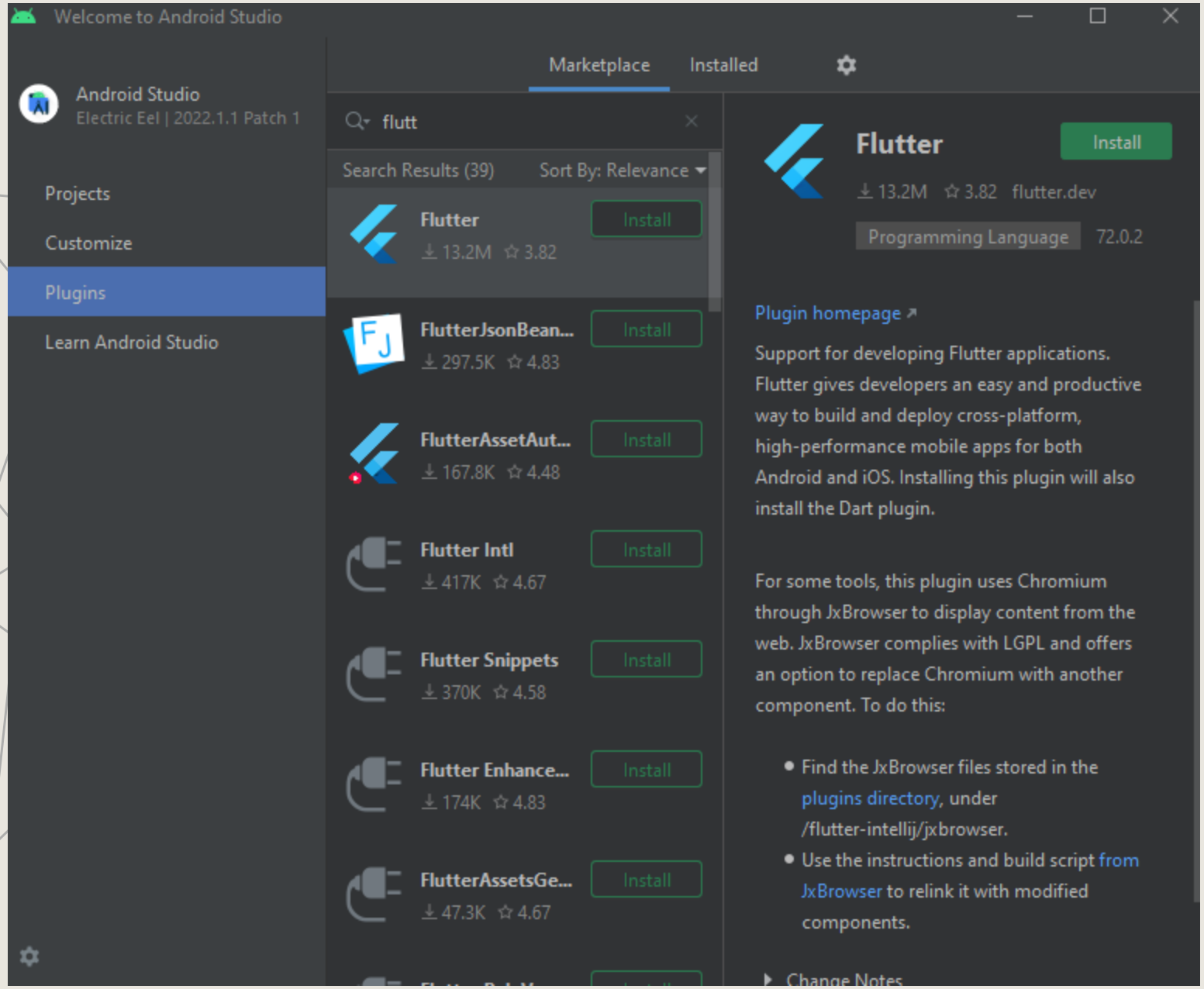
Android Studio Setup Wizard

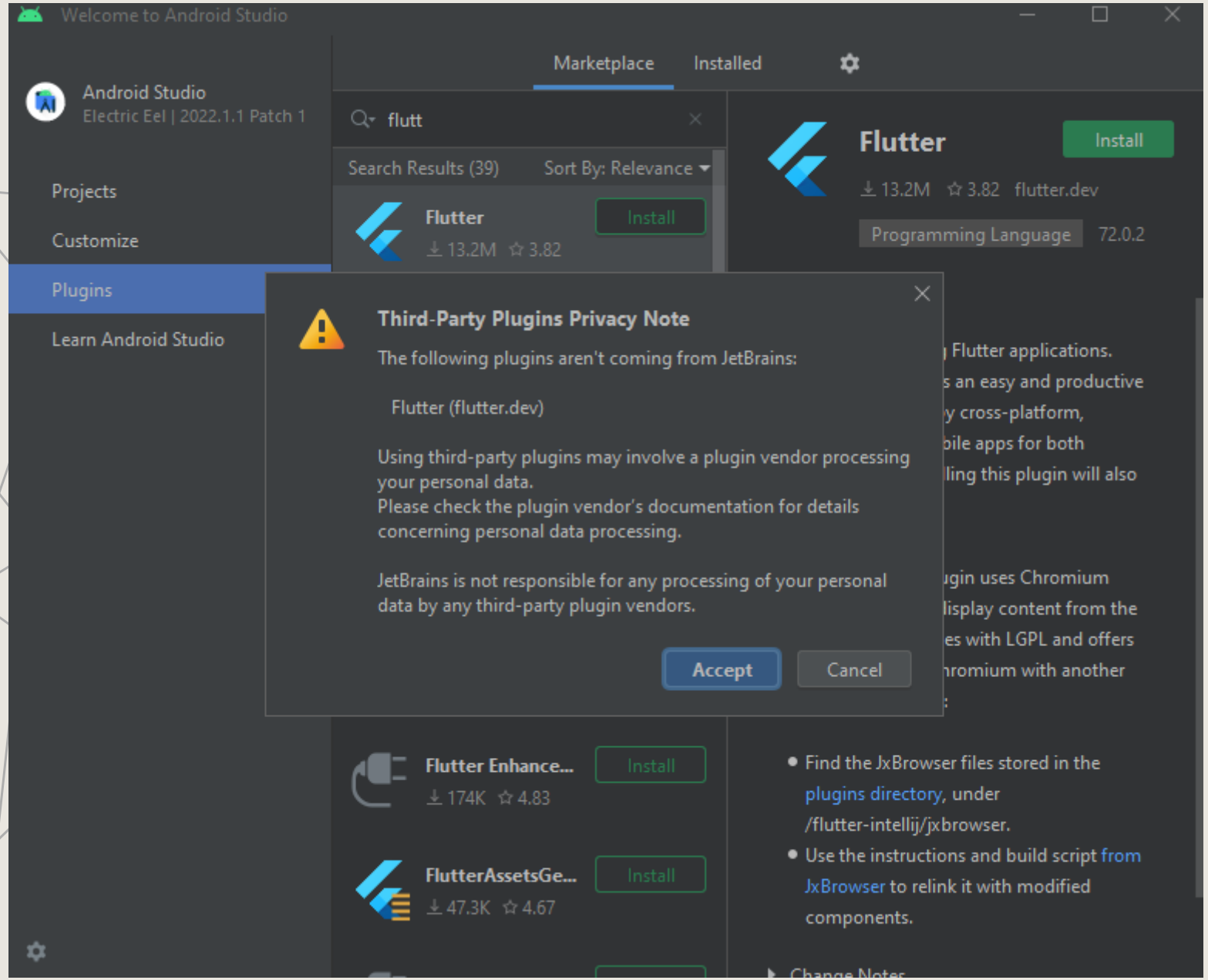
Downloading Components

```
Parsing C:\Users\sd-mfahmy\AppData\Local\Android\Sdk\build-tools\33.0.2\package.xml
Parsing C:\Users\sd-mfahmy\AppData\Local\Android\Sdk\extras\intel
 \Hardware_Accelerated_Execution_Manager\package.xml
Parsing C:\Users\sd-mfahmy\AppData\Local\Android\Sdk\patcher\v4\package.xml
Parsing C:\Users\sd-mfahmy\AppData\Local\Android\Sdk\platform-tools\package.xml
Parsing C:\Users\sd-mfahmy\AppData\Local\Android\Sdk\platforms\android-33-ext4
 \package.xml
java.net.SocketTimeoutException: Read timed out
Warning: An error occurred while preparing SDK package Android Emulator: Read timed
 out.

Preparing "Install Android Emulator (revision: 32.1.11)".
Downloading https://dl.google.com/android/repository/emulator-windows_x64-9536276.zip
"Install Android Emulator (revision: 32.1.11)" ready.
```

Previ... Next Cancel **Finish**


Welcome to Android Studio
Android Studio
Electric Eel | 2022.1.1 Patch 1
Projects
Customize
Plugins
Learn Android Studio
Welcome to Android Studio
Create a new project to start from scratch.
Open existing project from disk or version control.
New Project
Open
Get from VCS
More Actions


Welcome to Android Studio
Android Studio
Electric Eel | 2022.1.1 Patch 1
Projects
Customize
Plugins
Learn Android Studio
Marketplace
Installed
flutt
Search Results (39)
Sort By: Relevance
Flutter
13.2M 3.82
Install
FlutterJsonBean...
297.5K 4.83
Install
FlutterAssetAut...
167.8K 4.48
Install
Flutter Intl
417K 4.67
Install
Flutter Snippets
370K 4.58
Install
Flutter Enhance...
174K 4.83
Install
FlutterAssetsGe...
47.3K 4.67
Install
Flutter
Install
13.2M 3.82 flutter.dev
Programming Language 72.0.2
Plugin homepage
Support for developing Flutter applications. Flutter gives developers an easy and productive way to build and deploy cross-platform, high-performance mobile apps for both Android and iOS. Installing this plugin will also install the Dart plugin.
For some tools, this plugin uses Chromium through JxBrowser to display content from the web. JxBrowser complies with LGPL and offers an option to replace Chromium with another component. To do this:
Find the JxBrowser files stored in the plugins directory, under /flutter-intellij/jxbrowser.
Use the instructions and build script from JxBrowser to relink it with modified components.
Change Notes


Welcome to Android Studio
Android Studio
Electric Eel | 2022.1.1 Patch 1
Projects
Customize
Plugins
Learn Android Studio
Marketplace
Installed
flutt
Search Results (39)
Sort By: Relevance
Flutter
Install
13.2M 3.82
Flutter
Install
13.2M 3.82 flutter.dev
Programming Language
72.0.2
Third-Party Plugins Privacy Note
The following plugins aren't coming from JetBrains:
Flutter (flutter.dev)
Using third-party plugins may involve a plugin vendor processing your personal data.
Please check the plugin vendor's documentation for details concerning personal data processing.
JetBrains is not responsible for any processing of your personal data by any third-party plugin vendors.
Accept
Cancel
Flutter Enhance...
Install
174K 4.83
FlutterAssetsGe...
Install
47.3K 4.67
Find the JxBrowser files stored in the plugins directory, under /flutter-intellij/jxbrowser.
Use the instructions and build script from JxBrowser to relink it with modified components.
Change Notes

Google

flutter

All | Images | Videos | News | Books | More | Tools

About 196,000,000 results (0.37 seconds)

https://flutter.dev

Flutter - Build apps for any screen

Flutter is an open source framework by Google for building beautiful, natively compiled, multi-platform applications from a single codebase. · Fast · Productive.

Install

Install Flutter and get started. Downloads available for ...

Documentation

Install - Widget catalog - Flutter Gallery - Cookbook - Flutter - ...

Write your first Flutter app

You are now ready to start the "First Flutter app" codelab. In ...

Development

Flutter transforms the app development process so you ...

More results from flutter.dev »

Flutter

Software

Flutter is an open-source UI software development kit created by Google. It is used to develop cross-platform applications for Android, iOS, Linux, macOS, Windows, Google Fuchsia, and the web from a single codebase. First described in 2015, Flutter was released in May 2017. Wikipedia

Flutter

Multi-Platform Development Ecosystem Showcase Docs Get started

Get started

1. Install
2. Set up an editor
3. Test drive
4. Write your first app
5. Learn more

From another platform?

Flutter for Android devs

Flutter for SwiftUI devs

Flutter for UIKit devs

Set up an editor

Install

Get started > Install

Select the operating system on which you are installing Flutter:

Windows

macOS

Linux

Chrome OS

Get the Flutter SDK

1. Download the following installation bundle to get the latest stable release of the Flutter SDK:

   flutter_windows_3.7.0-stable.zip

   For other release channels, and older builds, see the SDK releases page.

2. Extract the zip file and place the contained `flutter` in the desired installation location for the Flutter SDK (for example, `C:\src\flutter`).

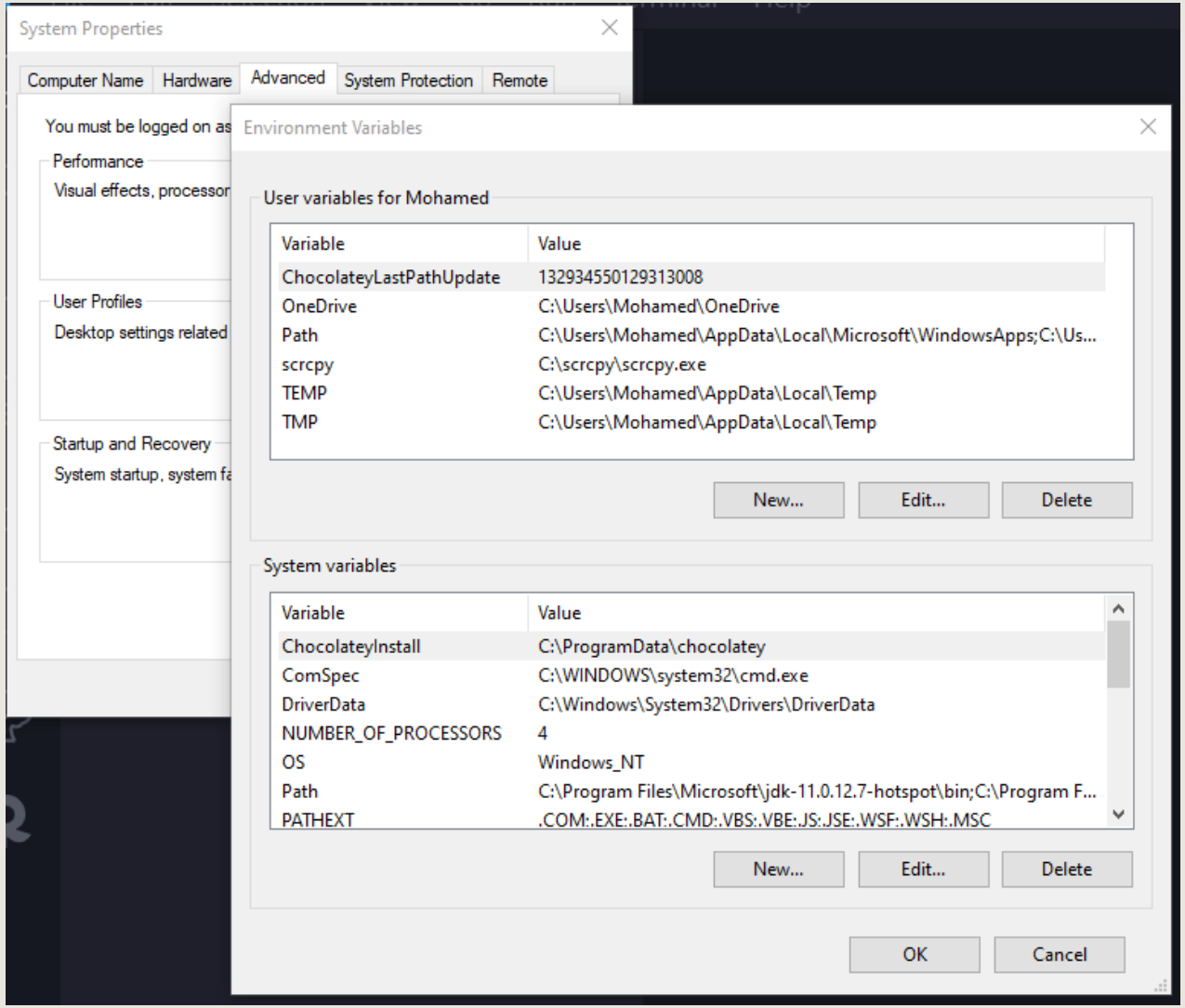
> ⚠ **Warning:** Do not install Flutter to a path that contains special characters or spaces.

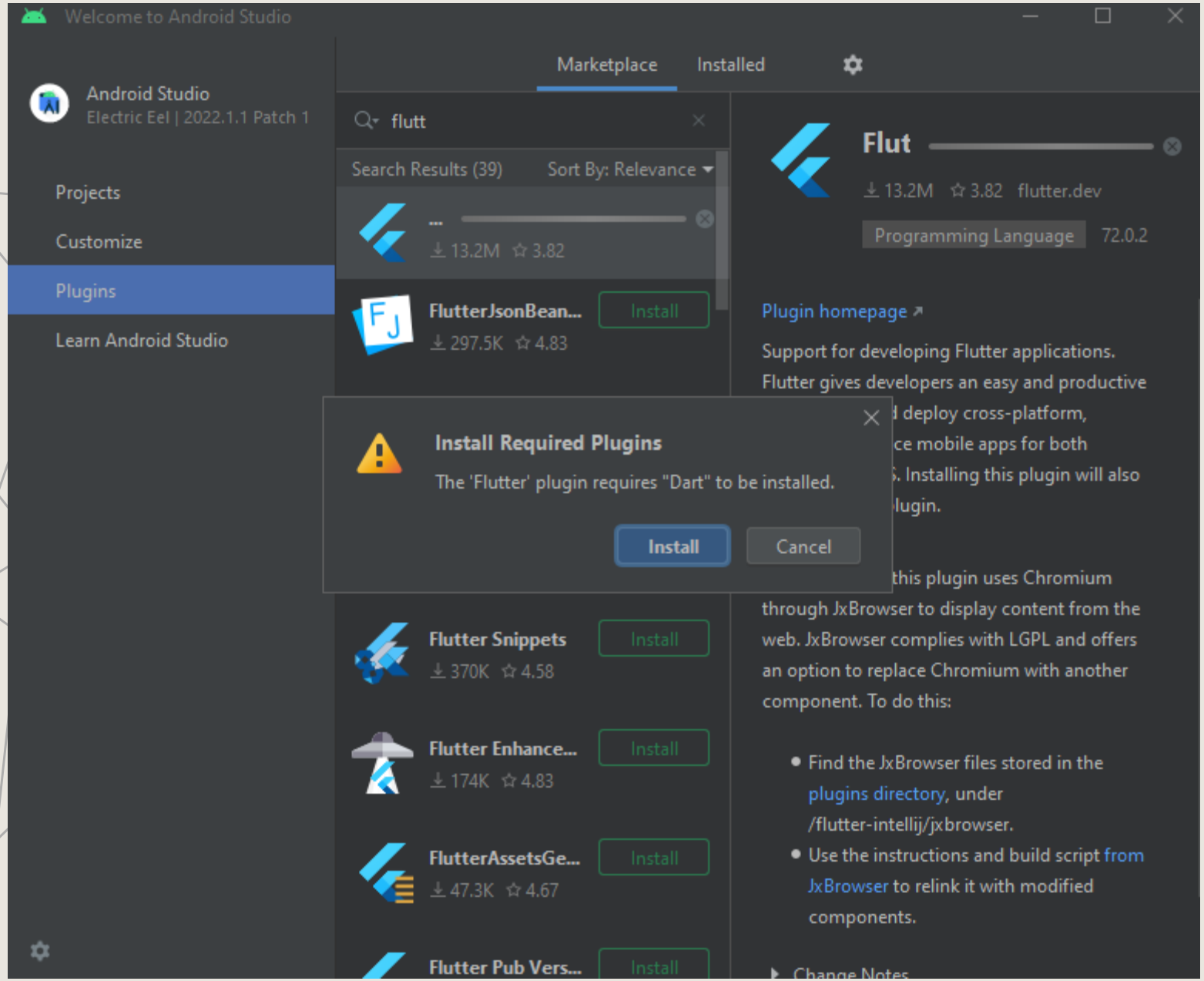
> ⚠ **Warning:** Do not install Flutter in a directory like `C:\Program Files\` that requires elevated privileges.

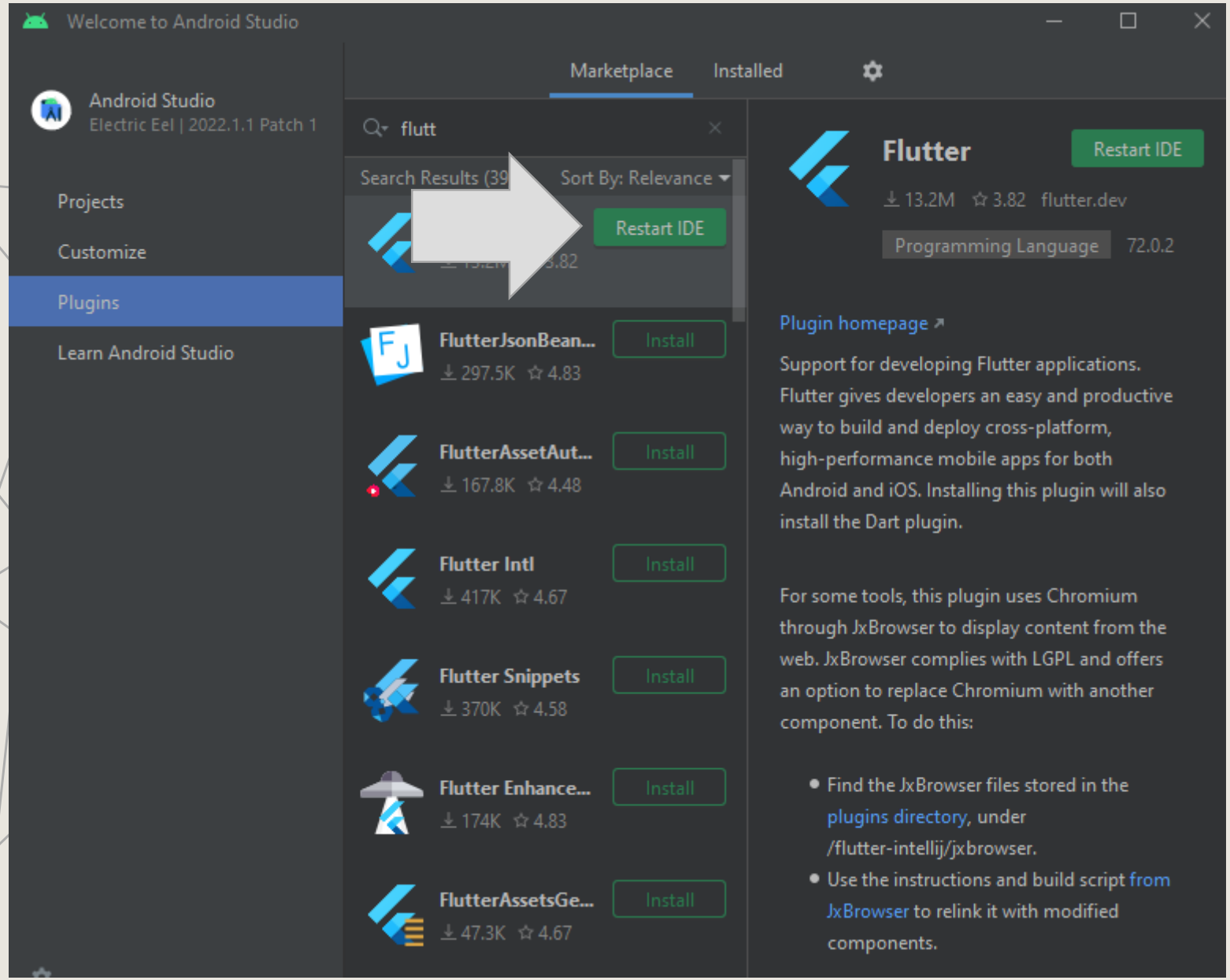
If you don't want to install a fixed version of the installation bundle, you can skip steps 1 and 2. Instead, get the source code from the Flutter repo on GitHub, and change branches or tags as needed. For example:

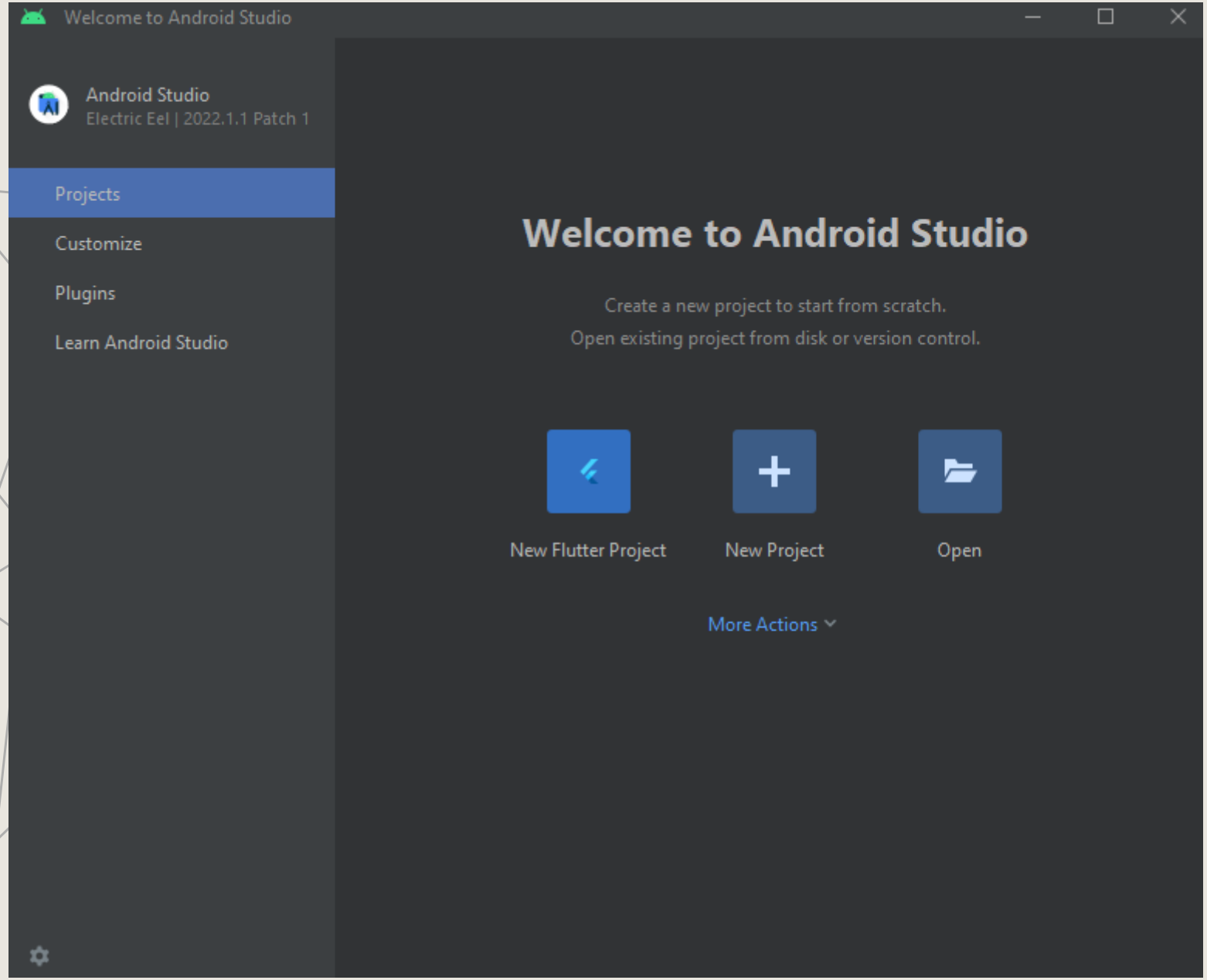
Extract the zip file and place the contained flutter in the desired installation location for the Flutter SDK

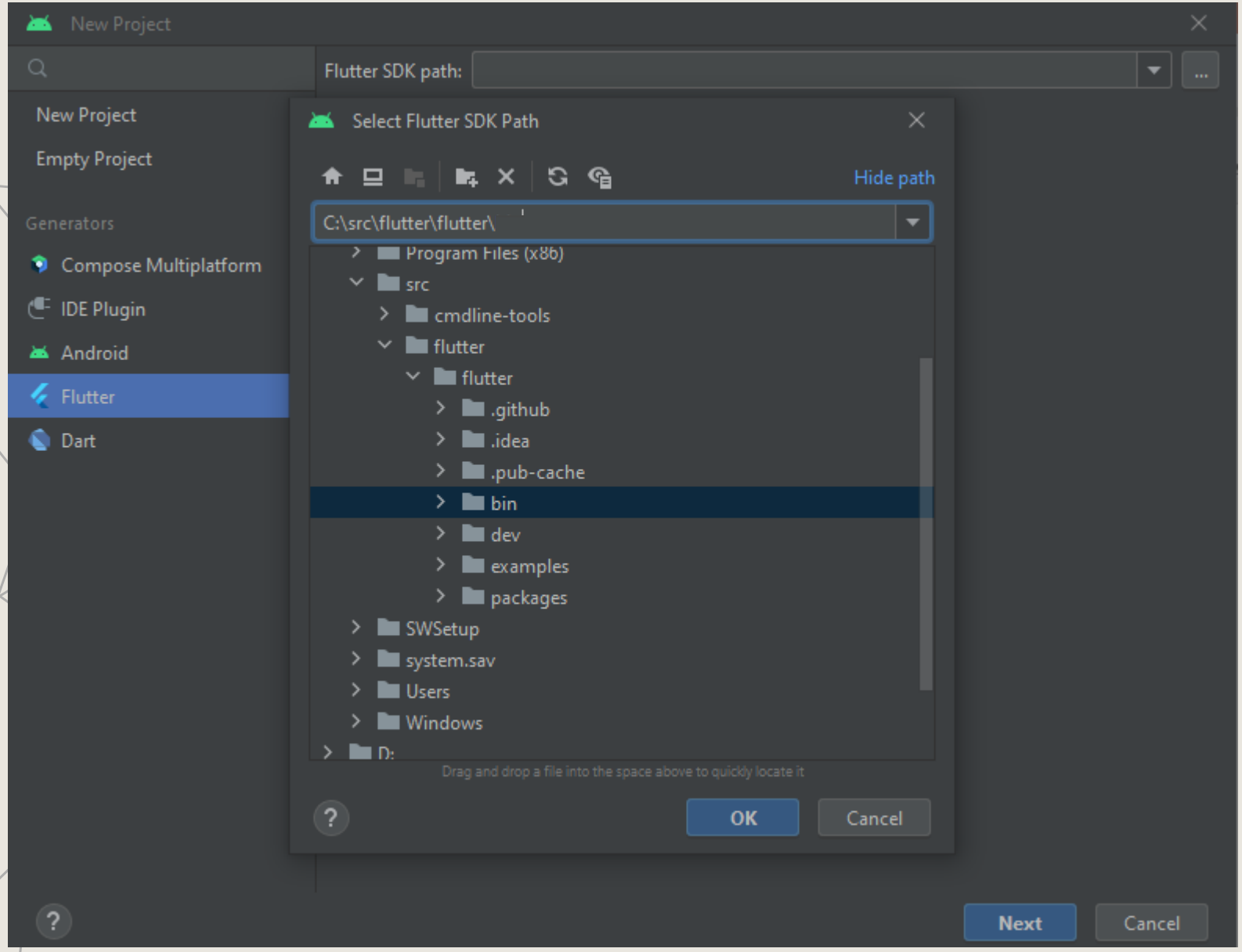
Update your path

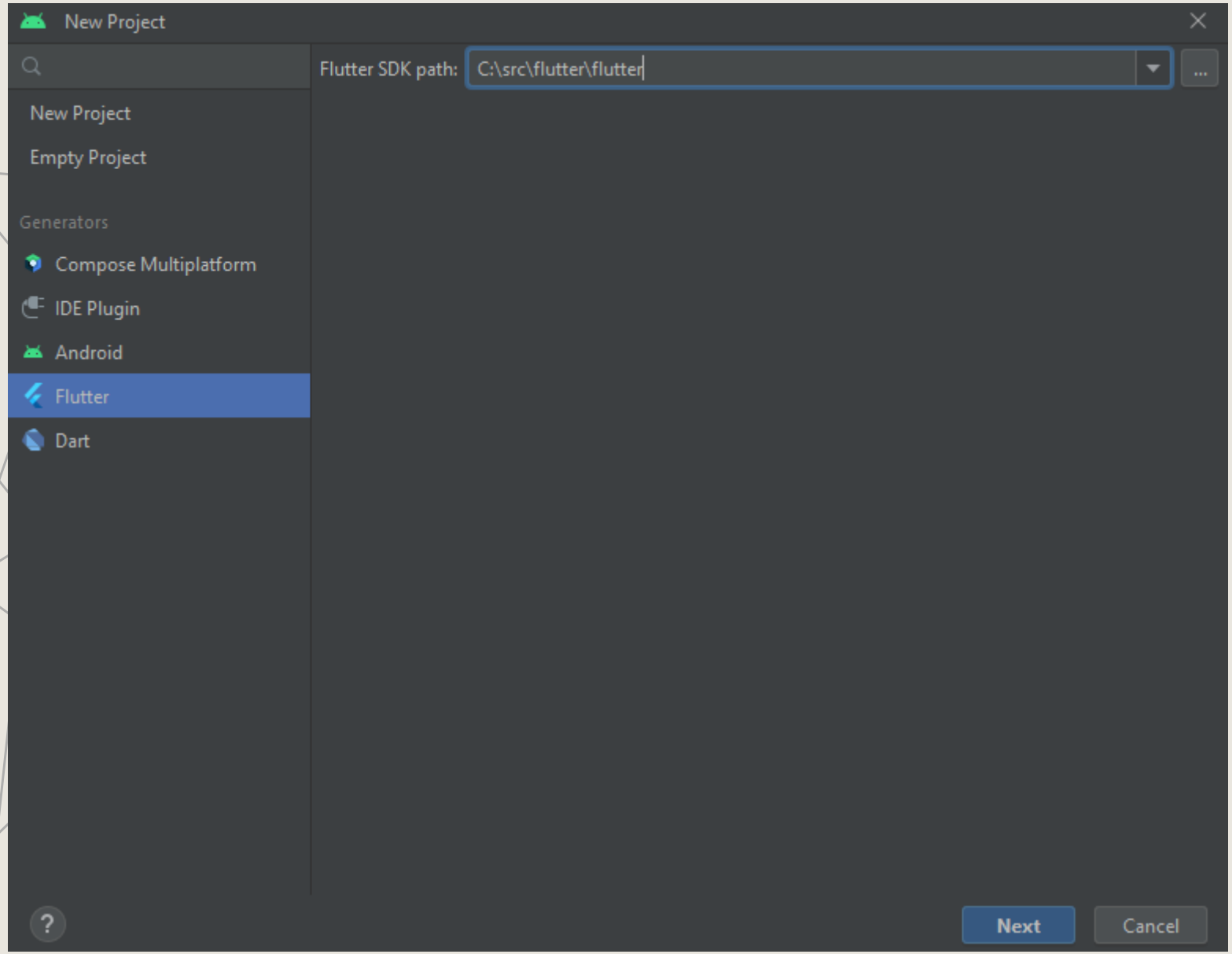


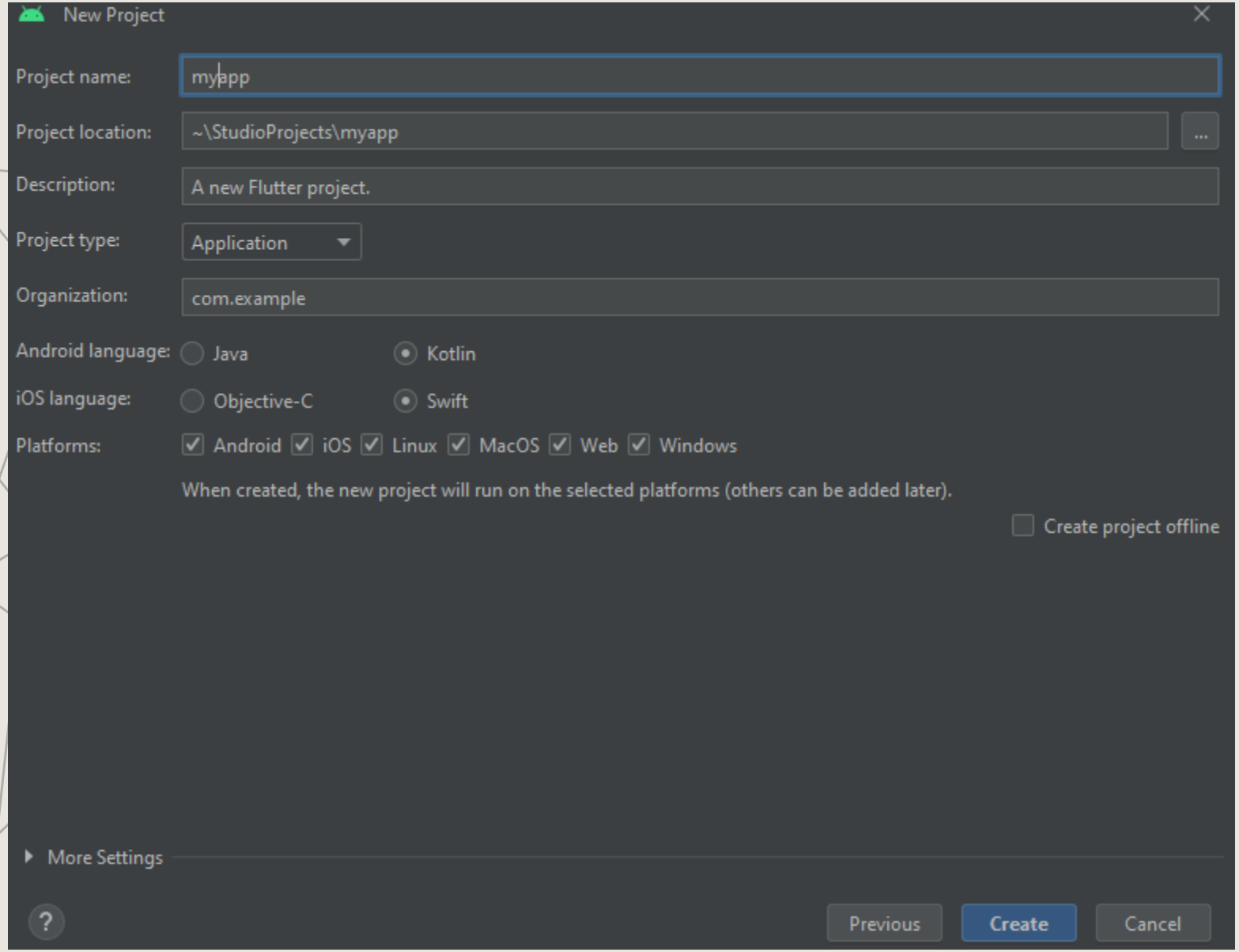

Welcome to Android Studio
Android Studio
Electric Eel | 2022.1.1 Patch 1
Projects
Customize
Plugins
Learn Android Studio
Marketplace
Installed
flutt
Search Results (39)
Sort By: Relevance
13.2M 3.82
FlutterJsonBean...
Install
297.5K 4.83
Install Required Plugins
The 'Flutter' plugin requires "Dart" to be installed.
Install
Cancel
Flutter Snippets
Install
370K 4.58
Flutter Enhance...
Install
174K 4.83
FlutterAssetsGe...
Install
47.3K 4.67
Flutter Pub Vers...
Install
Flut
13.2M 3.82 flutter.dev
Programming Language
72.0.2
Plugin homepage
Support for developing Flutter applications.
Flutter gives developers an easy and productive
deploy cross-platform,
ce mobile apps for both
. Installing this plugin will also
lugin.
this plugin uses Chromium
through JxBrowser to display content from the
web. JxBrowser complies with LGPL and offers
an option to replace Chromium with another
component. To do this:
Find the JxBrowser files stored in the
plugins directory, under
/flutter-intellij/jxbrowser.
Use the instructions and build script from
JxBrowser to relink it with modified
components.
Change Notes

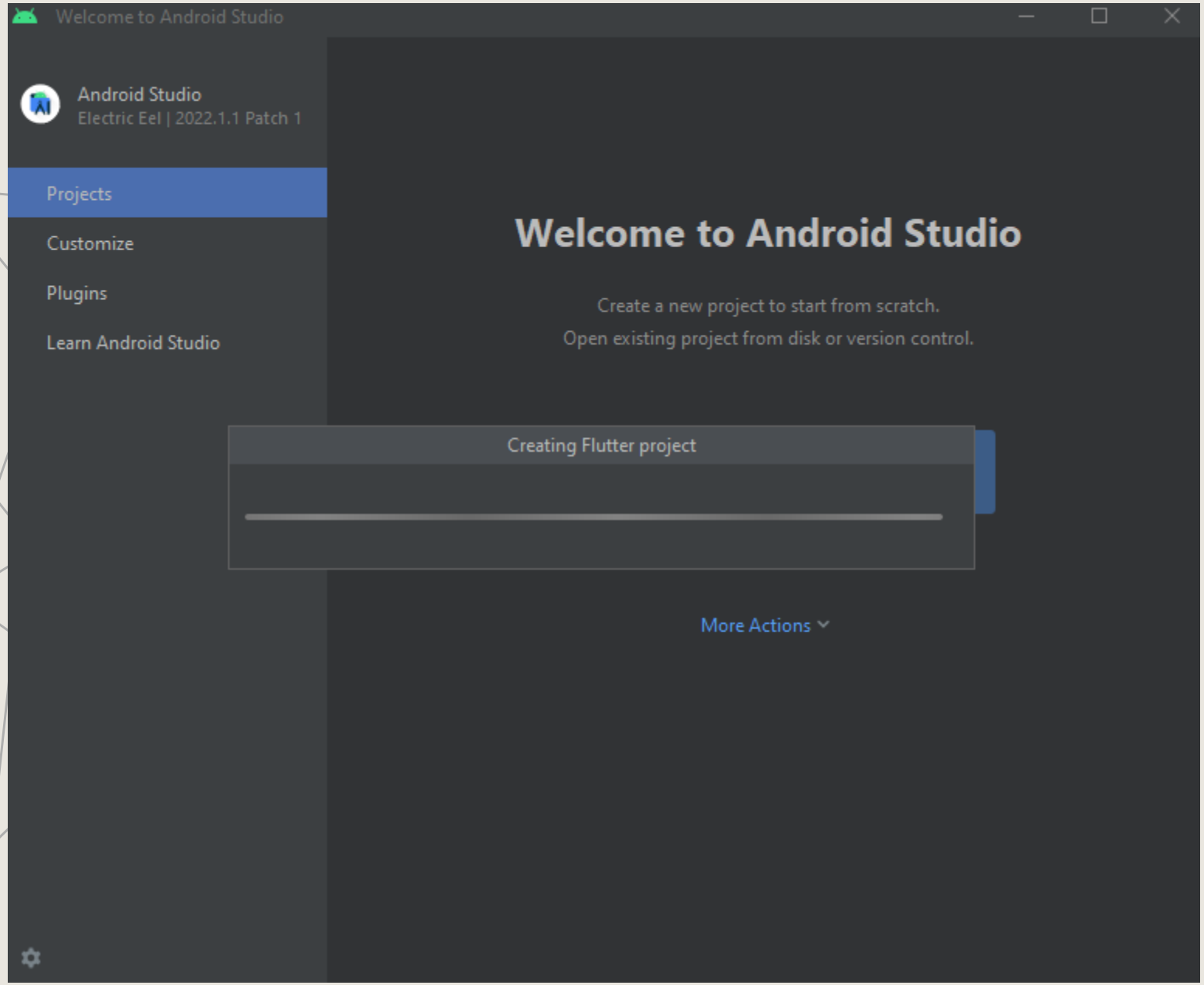

Welcome to Android Studio
Android Studio
Electric Eel | 2022.1.1 Patch 1
Projects
Customize
Plugins
Learn Android Studio
Marketplace
Installed
flutt
Search Results (39
Sort By: Relevance
Restart IDE
FlutterJsonBean...
297.5K 4.83
Install
FlutterAssetAut...
167.8K 4.48
Install
Flutter Intl
417K 4.67
Install
Flutter Snippets
370K 4.58
Install
Flutter Enhance...
174K 4.83
Install
FlutterAssetsGe...
47.3K 4.67
Install
Flutter
Restart IDE
13.2M 3.82 flutter.dev
Programming Language 72.0.2
Plugin homepage
Support for developing Flutter applications. Flutter gives developers an easy and productive way to build and deploy cross-platform, high-performance mobile apps for both Android and iOS. Installing this plugin will also install the Dart plugin.
For some tools, this plugin uses Chromium through JxBrowser to display content from the web. JxBrowser complies with LGPL and offers an option to replace Chromium with another component. To do this:
Find the JxBrowser files stored in the plugins directory, under /flutter-intellij/jxbrowser.
Use the instructions and build script from JxBrowser to relink it with modified components.


Welcome to Android Studio
Android Studio
Electric Eel | 2022.1.1 Patch 1
Projects
Customize
Plugins
Learn Android Studio
Welcome to Android Studio
Create a new project to start from scratch.
Open existing project from disk or version control.
New Flutter Project
New Project
Open
More Actions


New Project
Flutter SDK path:
New Project
Empty Project
Generators
Compose Multiplatform
IDE Plugin
Android
Flutter
Dart
Select Flutter SDK Path
Hide path
C:\src\flutter\flutter\
Program Files (x86)
src
cmdline-tools
flutter
flutter
.github
.idea
.pub-cache
bin
dev
examples
packages
SWSetup
system.sav
Users
Windows
D:
Drag and drop a file into the space above to quickly locate it
OK
Cancel
Next
Cancel


New Project
Flutter SDK path:
C:\src\flutter\flutter
New Project
Empty Project
Generators
Compose Multiplatform
IDE Plugin
Android
Flutter
Dart
Next
Cancel


New Project
Project name:
myapp
Project location:
~\StudioProjects\myapp
Description:
A new Flutter project.
Project type:
Application
Organization:
com.example
Android language:
Java
Kotlin
iOS language:
Objective-C
Swift
Platforms:
Android
iOS
Linux
MacOS
Web
Windows
When created, the new project will run on the selected platforms (others can be added later).
Create project offline
More Settings
Previous
Create
Cancel


Welcome to Android Studio
Android Studio
Electric Eel | 2022.1.1 Patch 1
Projects
Customize
Plugins
Learn Android Studio
Welcome to Android Studio
Create a new project to start from scratch.
Open existing project from disk or version control.
Creating Flutter project
More Actions

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help myapp - main.dart [myapp]

myapp | <no device selected> | main.dart | No Devices

Project

myapp C:\Users\sd-mfahmy\StudioProjects\myapp

- .dart_tool
- .idea
- android [myapp_android]
- ios
- lib
- test
- web
- windows
- .gitignore
- .metadata
- analysis_options.yaml
- myapp.iml
- pubspec.lock
- pubspec.yaml
- README.md
- External Libraries
- Scratches and Consoles

README.md | main.dart

```dart
import 'package:flutter/material.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        // This is the theme of your application.
        //
        // Try running your application with "flutter ru
        // application has a blue toolbar. Then, without
        // changing the primarySwatch below to Colors.gr
        // "hot reload" (press "r" in the console where
        // or simply save your changes to "hot reload" i
        // Notice that the counter didn't reset back to
        // is not restarted.
        primarySwatch: Colors.blue,
```

Assistant: What's New

What's New in Electric Eel

This panel describes some of the new features and behavior changes included in

To open this panel again later, select **Help > What's New in Android Studio** fror

Read in a browser

New Logcat

Logcat has bee
most significar

New formatter: Logcat now formats logs to make it easier to scan useful inform
messages, and

Create multipl
easily switch b
can click and drag to rearrange the tabs. Additionally, to help you more easily co

Kotlin code style
Do you want to update your Kotlin code style...
Apply the code style | Don't ask again

Welcome to Flutter!
The Flutter plugin reports feature usage statistics...
Sounds good! | No thanks

Version Control | TODO | Problems | Terminal | App Inspection | Logcat | App Quality Insights | Services | Profiler | Dart Analysis | Layout Inspector

Kotlin code style: Do you want to update your Kotlin code style settings to the recommended ones? // Apply the code style // Don'... (a minute ago) Downloading 1 file... 1:1 CRLF UTF-8 2 spaces

Don’t forget to create account

ORACLE Products Industries Resources Customers Partners Developers Events Company View Accounts Contact Sales

macOS x64 Compressed Archive	155.83 MB	jdk-11.0.17_macos-x64_bin.tar.gz
macOS x64 DMG Installer	155.32 MB	jdk-11.0.17_macos-x64_bin.dmg
Solaris SPARC Compressed Archive		s-sparcv9_bin.tar.gz
Windows x64 Installer		ws-x64_bin.exe
Windows x64 Compressed Archive		ws-x64_bin.zip

You must accept the Oracle Technology Network License Agreement for Oracle Java SE to download this software.

☑ I reviewed and accept the Oracle Technology Network License Agreement for Oracle Java SE

Required

You will be redirected to the login screen in order to download the file.

Download jdk-11.0.17_windows-x64_bin.exe

Java SE Development Kit 11

This software is licensed under the Oracle Technology Network License Agreement for Oracle Java SE

JDK 11.0.16.1 checksum

Product / File Description	File Size	Download
Linux ARM 64 RPM Package	140.76 MB	jdk-11.0.16.1_linux-aarch64_bin.rpm

Check Your Email

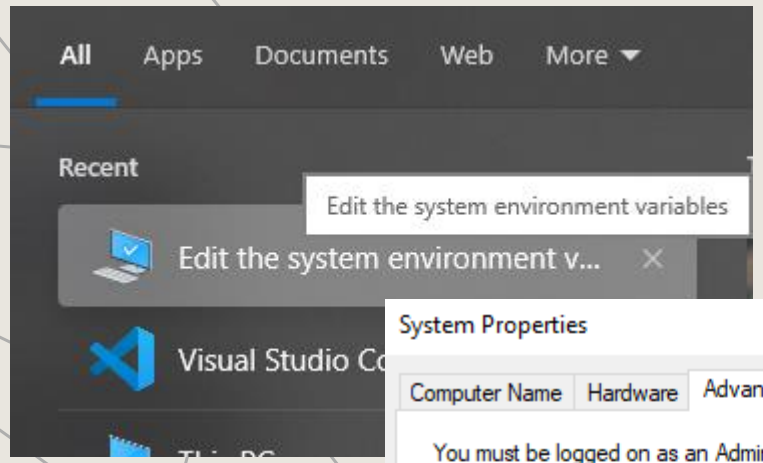
Verify your email address to use your account.

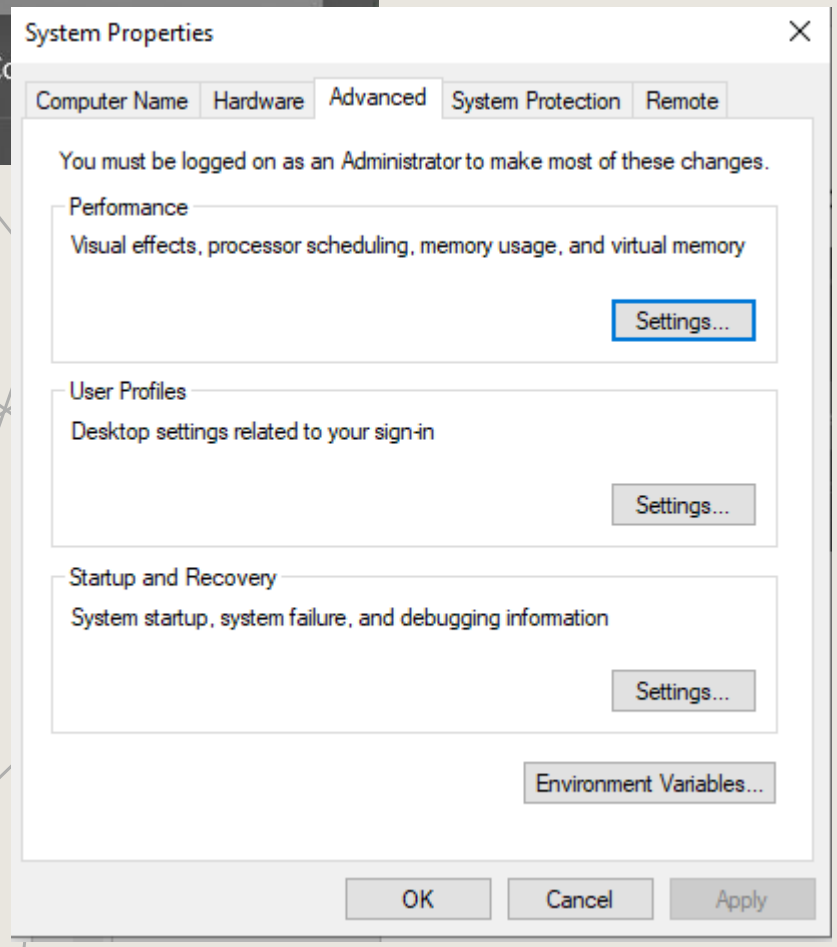
We sent an email to q[illegible]za@gmail.com with a button to verify your email address.

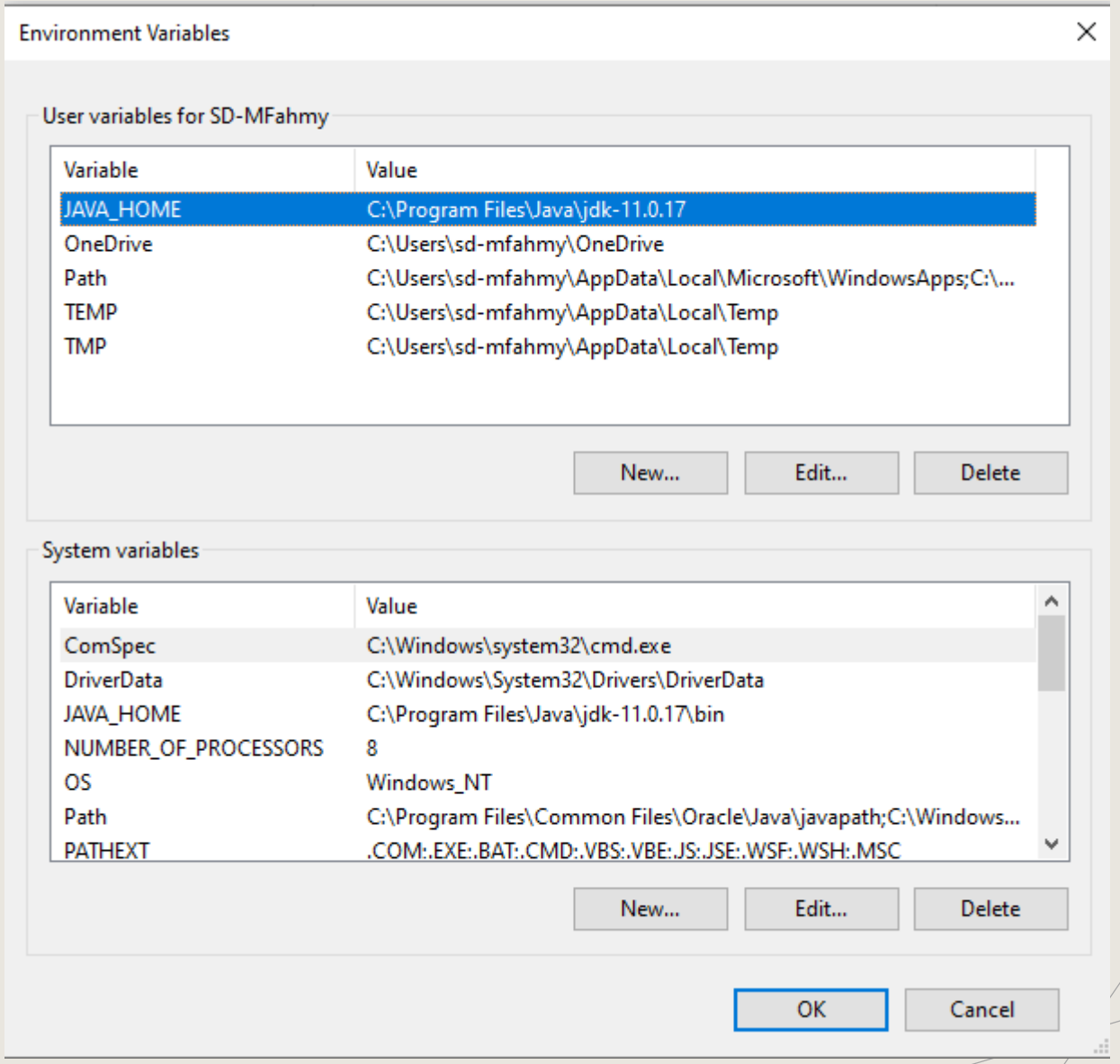
Did you receive the email? If not, check your spam folder or request a new verification email for up to 3 days. If you do not verify your email address within 3 days, you will need to create a new account. If you are having trouble, see Account Help.

Account Help | Subscriptions | Unsubscribe | Terms of Use and Privacy | Cookie Preferences

Add JAVA_HOME to your Path







File Edit View Navigate Code Refactor Build Run Tools VCS Window Help myapp - main.dart [myapp]

myapp 〉 lib 〉 main.dart

CPH2095 (mobile) | main.dart | OPPO CPH2095

Project

myapp C:\Users\sd-mfahmy\StudioProjects\myapp
- .dart_tool
- .idea
- android [myapp_android]
- build
- ios
- lib
- test
- web
- windows
- .gitignore
- .metadata
- analysis_options.yaml
- myapp.iml
- pubspec.lock
- pubspec.yaml
- README.md

External Libraries

```
import 'package:flutter/material.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        // This is the theme of your application.
```

Run: main.dart

Console

```
Launching lib\main.dart on CPH2095 in debug mode...
Running Gradle task 'assembleDebug'...
```

Kotlin code style

Do you want to update your Kotlin code style...

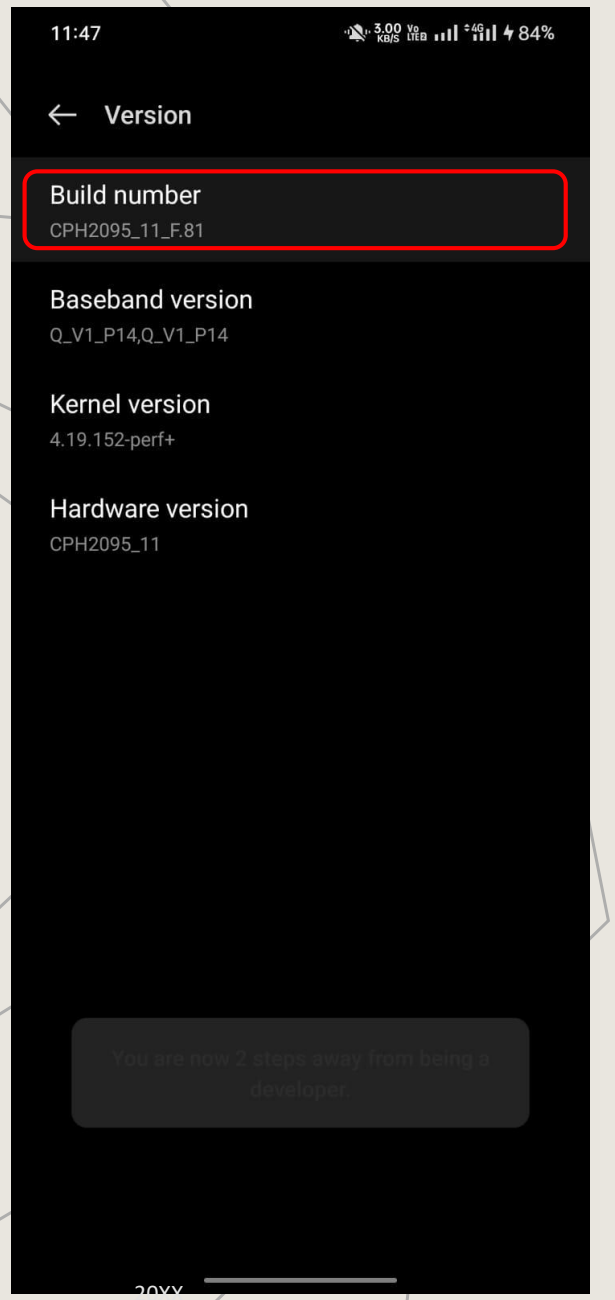
Apply the code style | Don't ask again

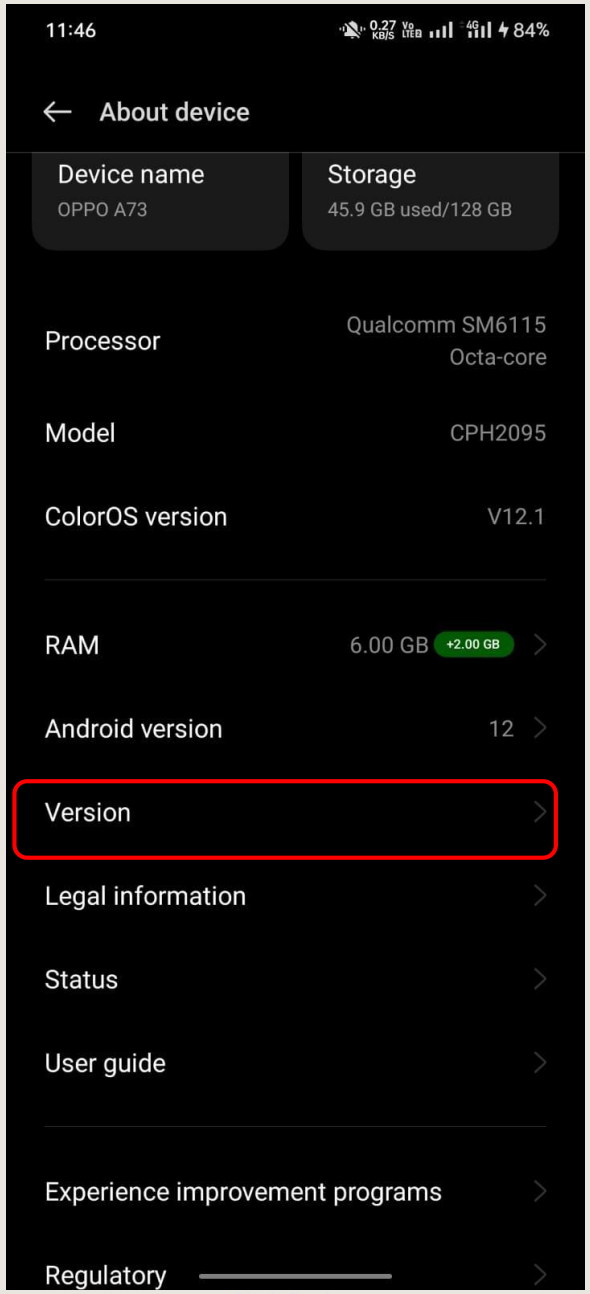
Version Control | Run | Profiler | Logcat | App Quality Insights | TODO | Problems | Terminal | Services | App Inspection | Dart Analysis | Layout Inspector

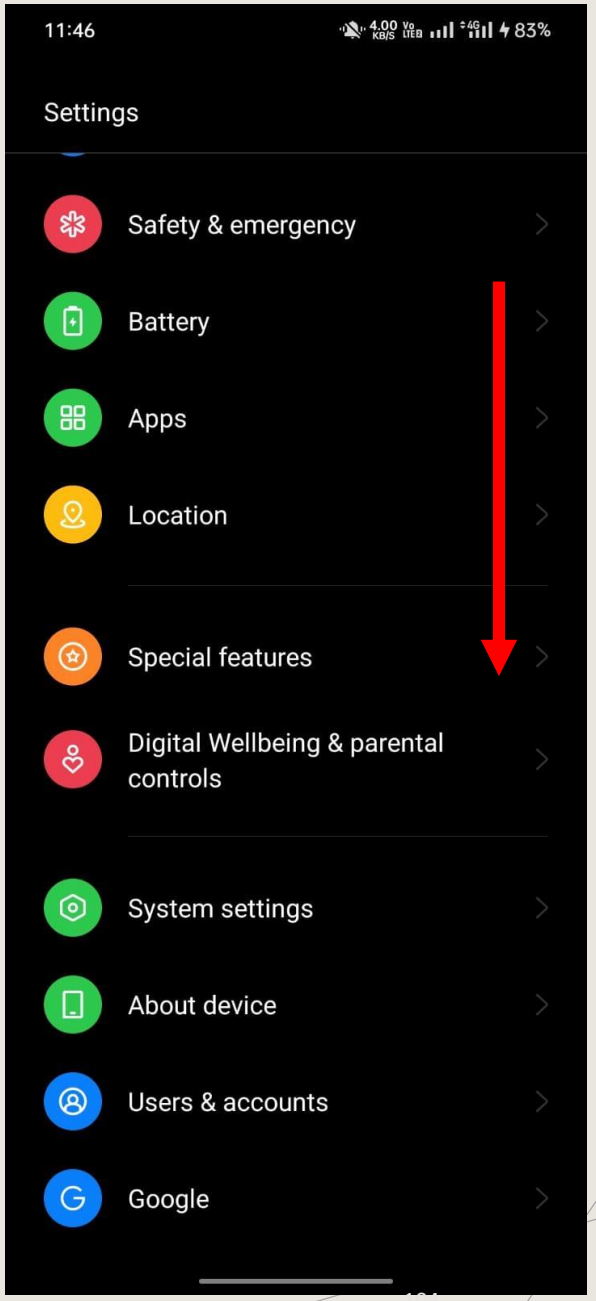
Kotlin code style: Do you want to update your Kotlin code style settings to the recommended ones? // Apply the code sty... (a minute ago) Running Gradle task 'assembleDebug'... 2:1 CRLF UTF-8 2 spaces

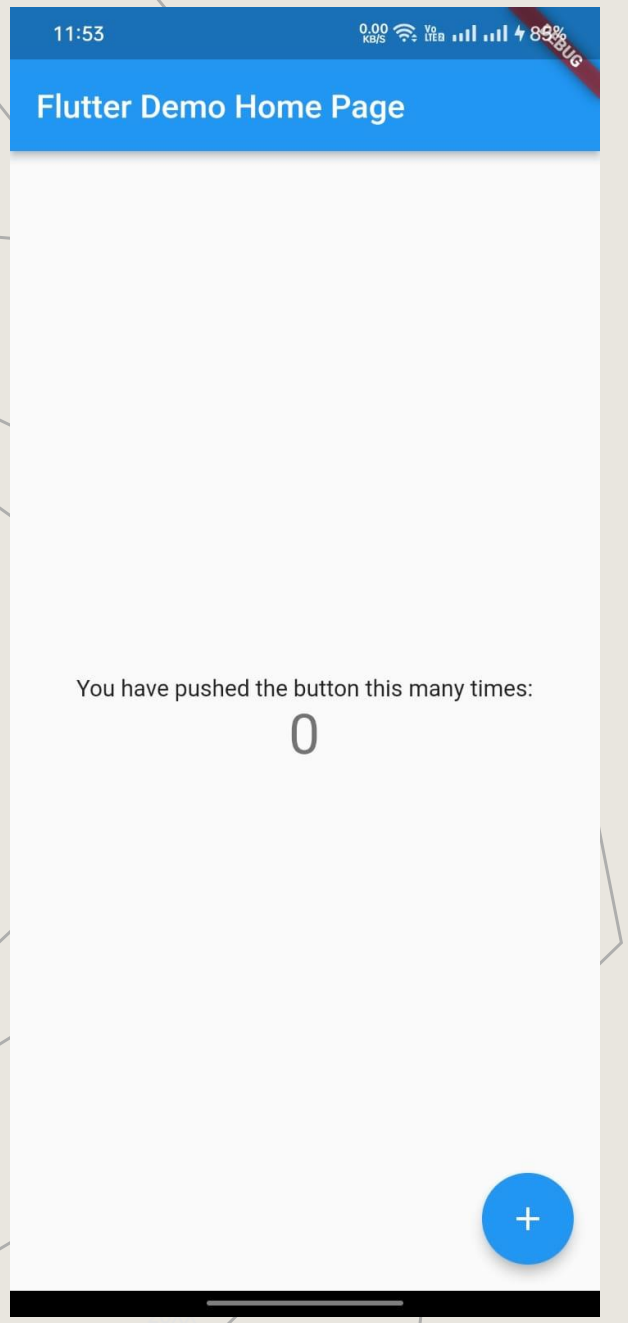
Setup mobile device

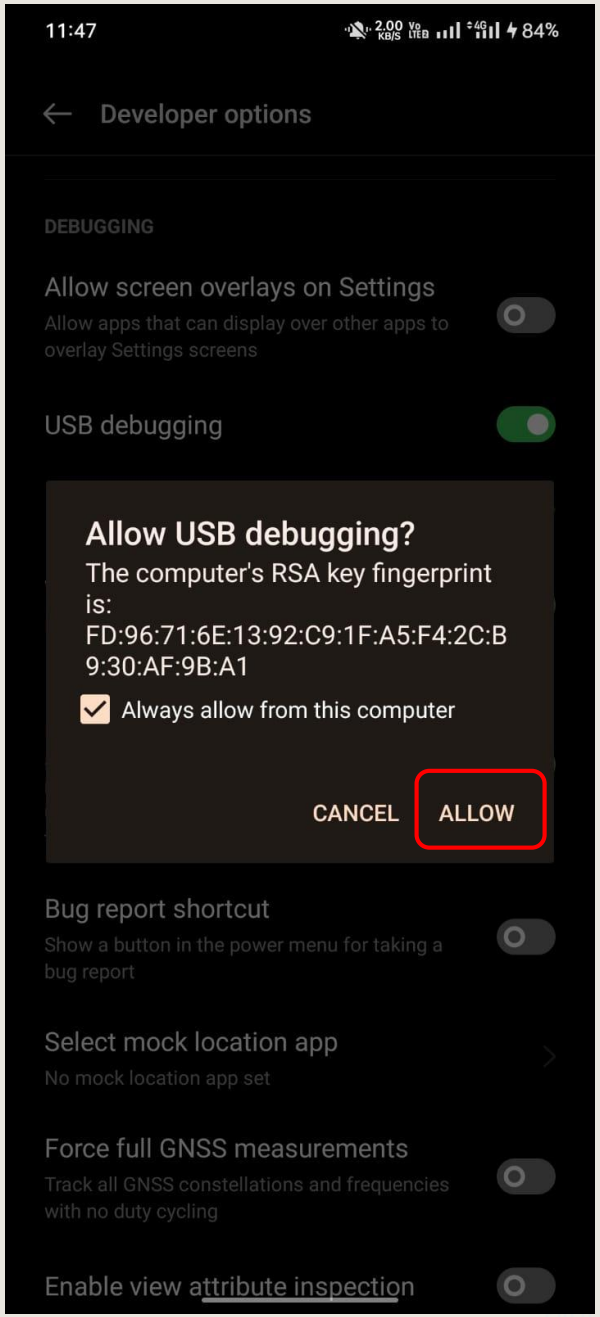
On the device,
go to :
Settings > About <device>.
Tap the Build number seven times to make Settings > Developer options available.
Then enable the USB Debugging option.
Tip: You might also want to enable the Stay awake option, to prevent your Android device from sleeping while plugged into
the USB port

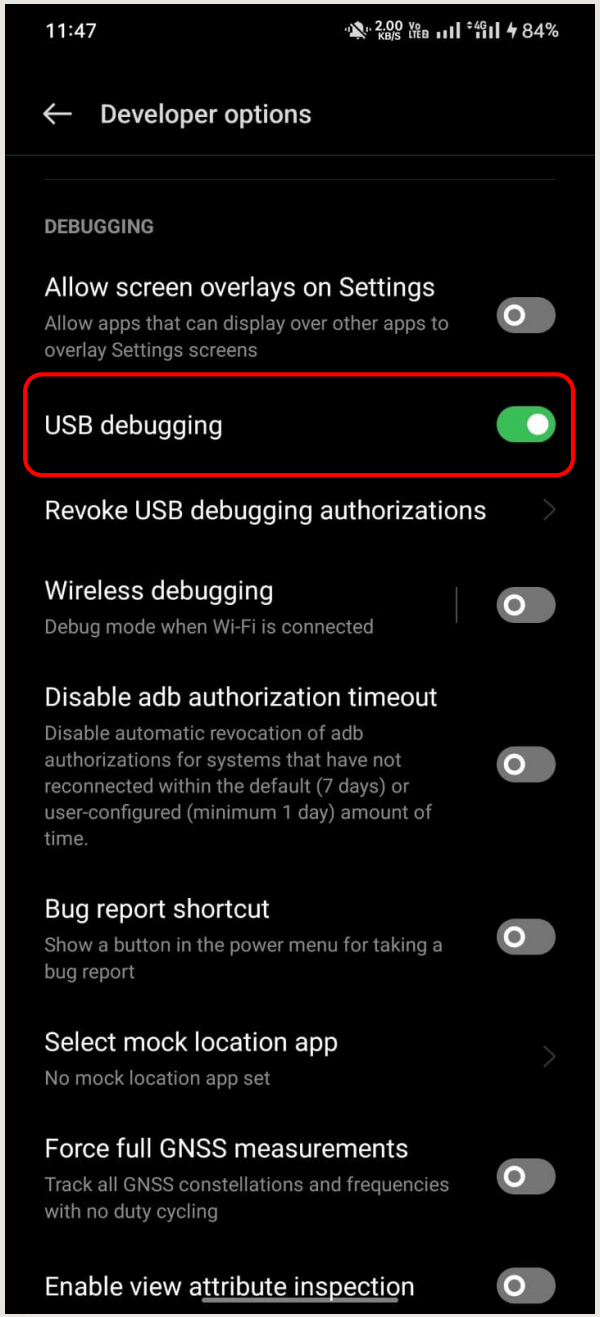

11:47
84%
Version
Build number
CPH2095_11_F.81
Baseband version
Q_V1_P14,Q_V1_P14
Kernel version
4.19.152-perf+
Hardware version
CPH2095_11
You are now 2 steps away from being a developer.

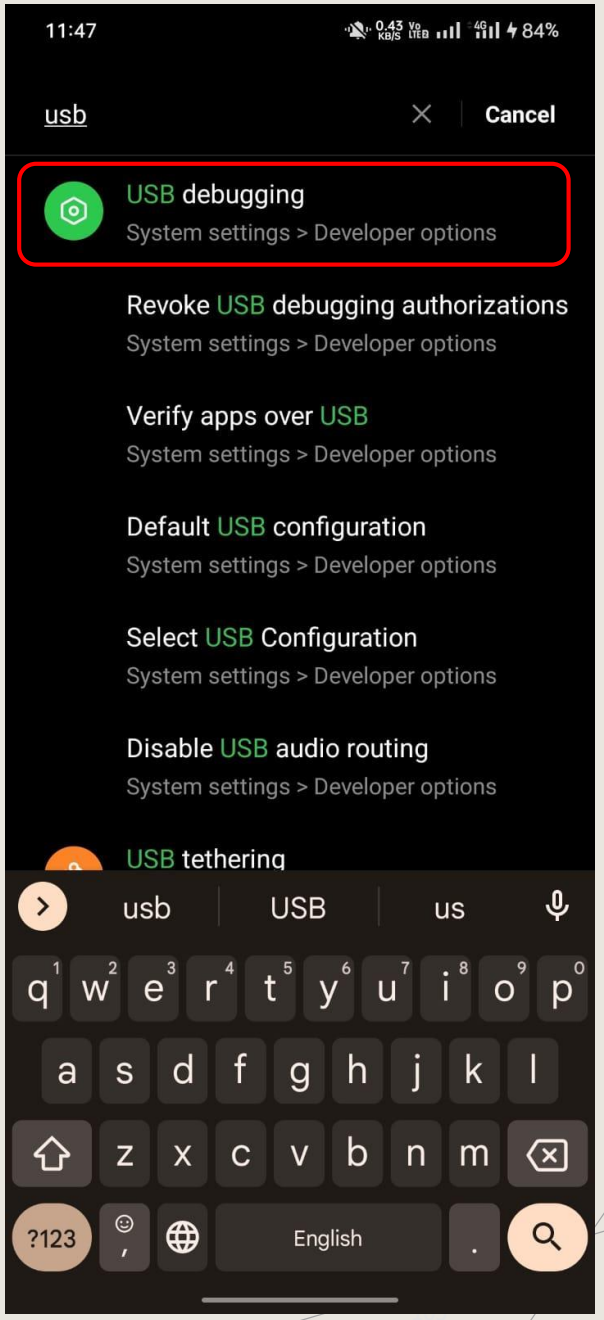

11:46
84%
About device
Device name
OPPO A73
Storage
45.9 GB used/128 GB
Processor
Qualcomm SM6115 Octa-core
Model
CPH2095
ColorOS version
V12.1
RAM
6.00 GB
+2.00 GB
Android version
12
Version
Legal information
Status
User guide
Experience improvement programs
Regulatory


11:46
83%
Settings
Safety & emergency
Battery
Apps
Location
Special features
Digital Wellbeing & parental controls
System settings
About device
Users & accounts
Google

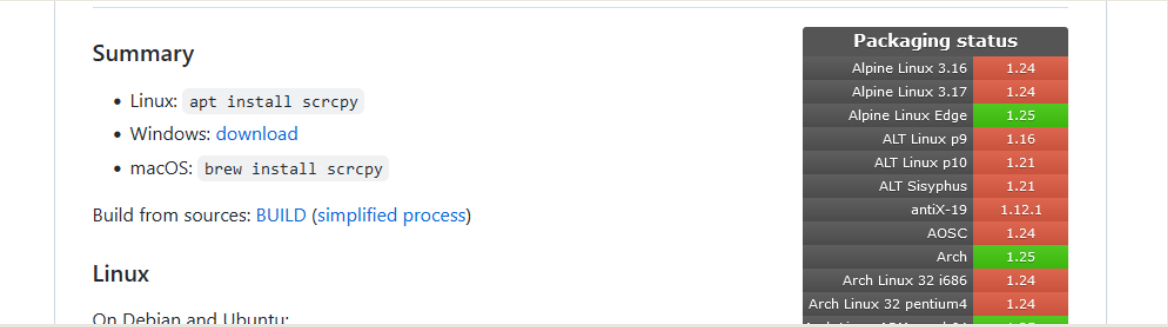

11:53
DEBUG
Flutter Demo Home Page
You have pushed the button this many times:
0
+


11:47
84%
Developer options
DEBUGGING
Allow screen overlays on Settings
Allow apps that can display over other apps to overlay Settings screens
USB debugging
Allow USB debugging?
The computer's RSA key fingerprint is:
FD:96:71:6E:13:92:C9:1F:A5:F4:2C:B9:30:AF:9B:A1
Always allow from this computer
CANCEL
ALLOW
Bug report shortcut
Show a button in the power menu for taking a bug report
Select mock location app
No mock location app set
Force full GNSS measurements
Track all GNSS constellations and frequencies with no duty cycling
Enable view attribute inspection


11:47
84%
Developer options
DEBUGGING
Allow screen overlays on Settings
Allow apps that can display over other apps to overlay Settings screens
USB debugging
Revoke USB debugging authorizations
Wireless debugging
Debug mode when Wi-Fi is connected
Disable adb authorization timeout
Disable automatic revocation of adb authorizations for systems that have not reconnected within the default (7 days) or user-configured (minimum 1 day) amount of time.
Bug report shortcut
Show a button in the power menu for taking a bug report
Select mock location app
No mock location app set
Force full GNSS measurements
Track all GNSS constellations and frequencies with no duty cycling
Enable view attribute inspection


11:47
84%
usb
Cancel
USB debugging
System settings > Developer options
Revoke USB debugging authorizations
System settings > Developer options
Verify apps over USB
System settings > Developer options
Default USB configuration
System settings > Developer options
Select USB Configuration
System settings > Developer options
Disable USB audio routing
System settings > Developer options
USB tethering
usb
USB
us
English
?123

https://github.com/Genymobile/scrcpy



Summary

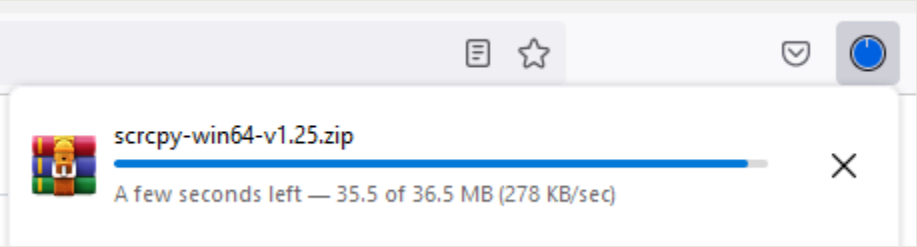
- Linux: `apt install scrcpy`
- Windows: download
- macOS: `brew install scrcpy`

Build from sources: BUILD (simplified process)

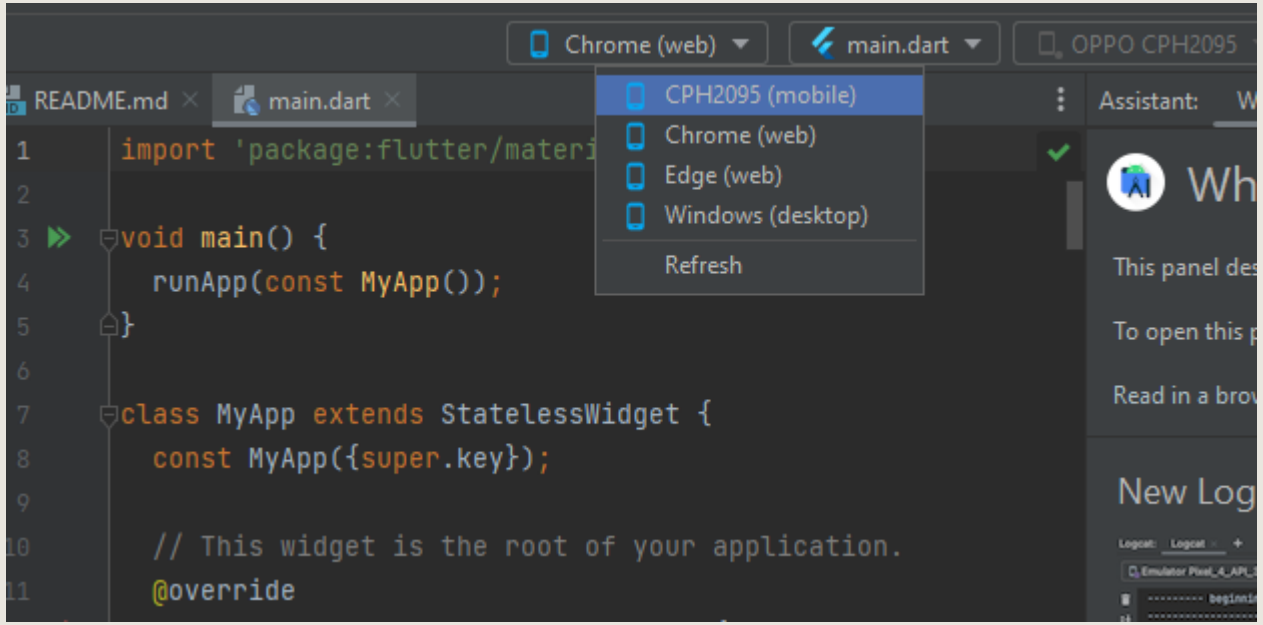
Linux

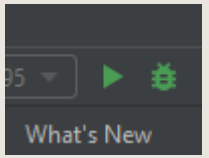
On Debian and Ubuntu:

Packaging status	
Alpine Linux 3.16	1.24
Alpine Linux 3.17	1.24
Alpine Linux Edge	1.25
ALT Linux p9	1.16
ALT Linux p10	1.21
ALT Sisyphus	1.21
antiX-19	1.12.1
AOSC	1.24
Arch	1.25
Arch Linux 32 i686	1.24
Arch Linux 32 pentium4	1.24

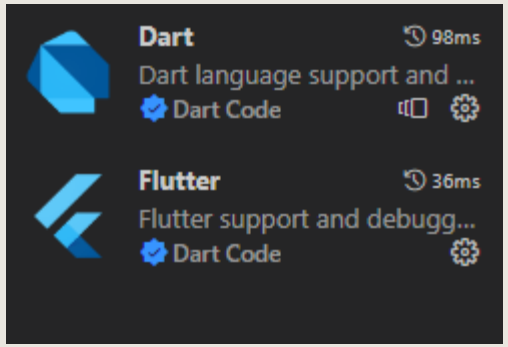


Make sure that you enabled USB Debugging

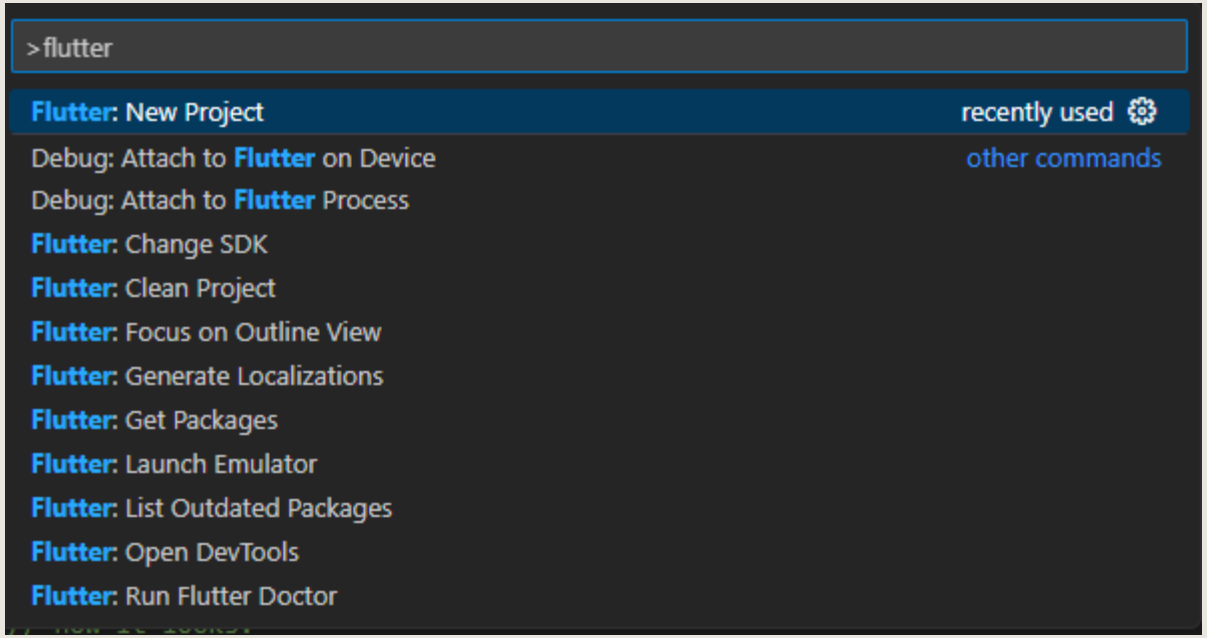




Now run VScode and add the extensions to it



Then CTRL + Shift + P to create new app



where flutter dart

```
C:\Users\Mohamed>where flutter dart
C:\flutterSdk\flutter\bin\flutter
C:\flutterSdk\flutter\bin\flutter.bat
C:\flutterSdk\flutter\bin\dart
C:\flutterSdk\flutter\bin\dart.bat
```

Ctrl + shift + p

And chose application