Research & Development Team

Mobile Programming with Flutter

www.pnpsw.com

sommai.k@pnpsw.com

081-754-4663

Lineid : sommai.k

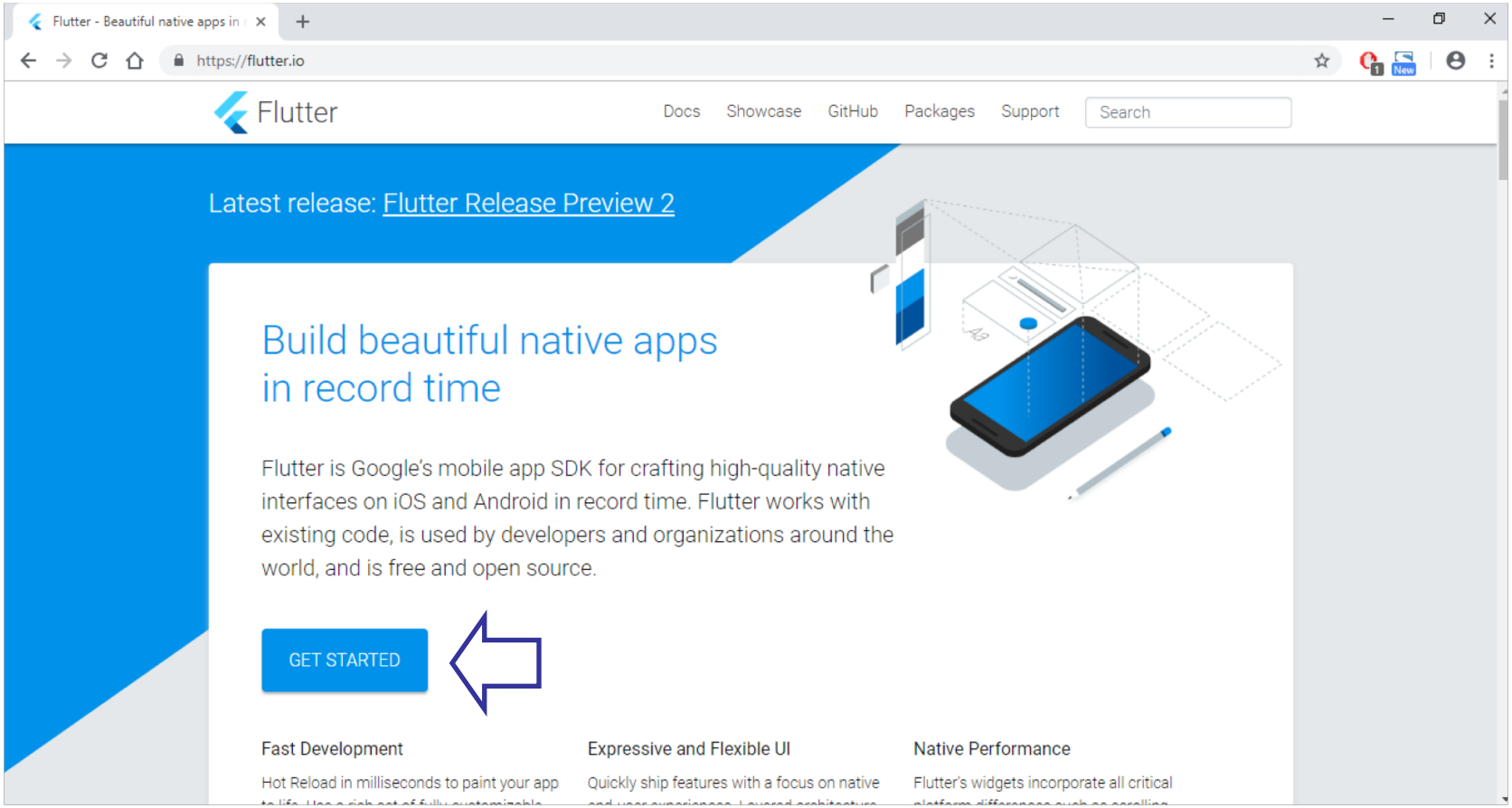
Software

INSTALLATION

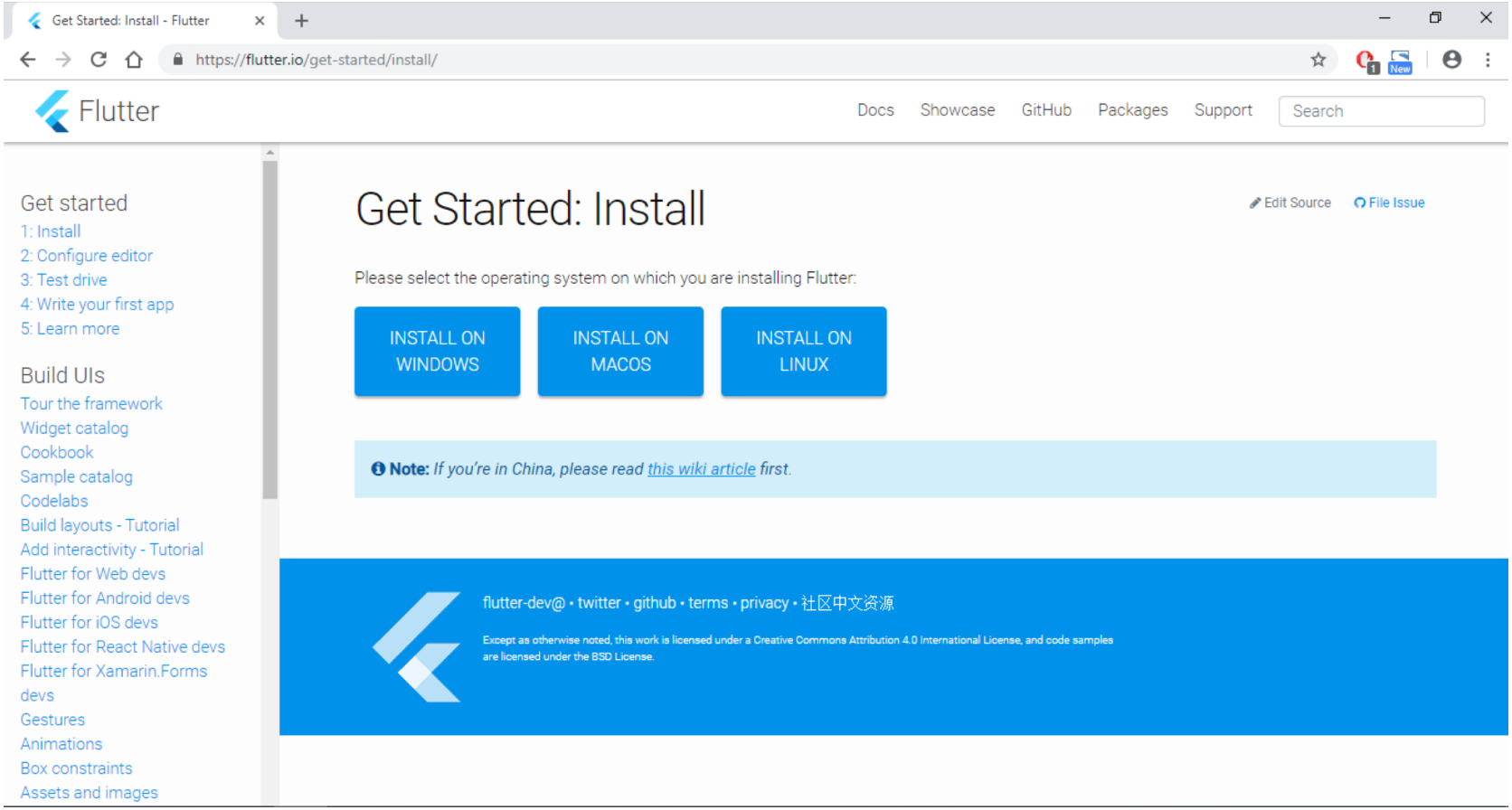
การติดตั้ง

FLUTTER

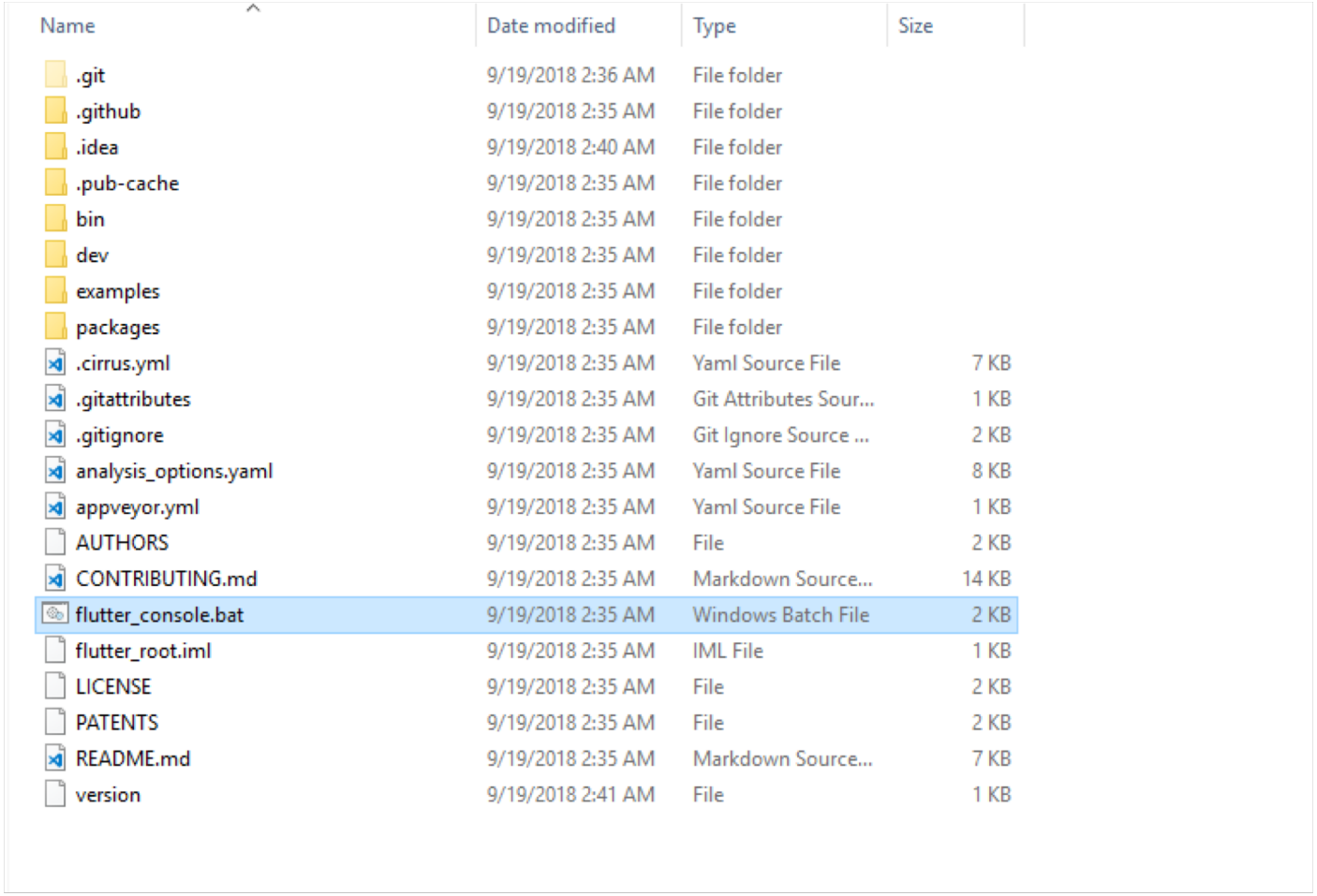
เข้า url https://flutter.io



เลือก version ที่ตรงกับ OS

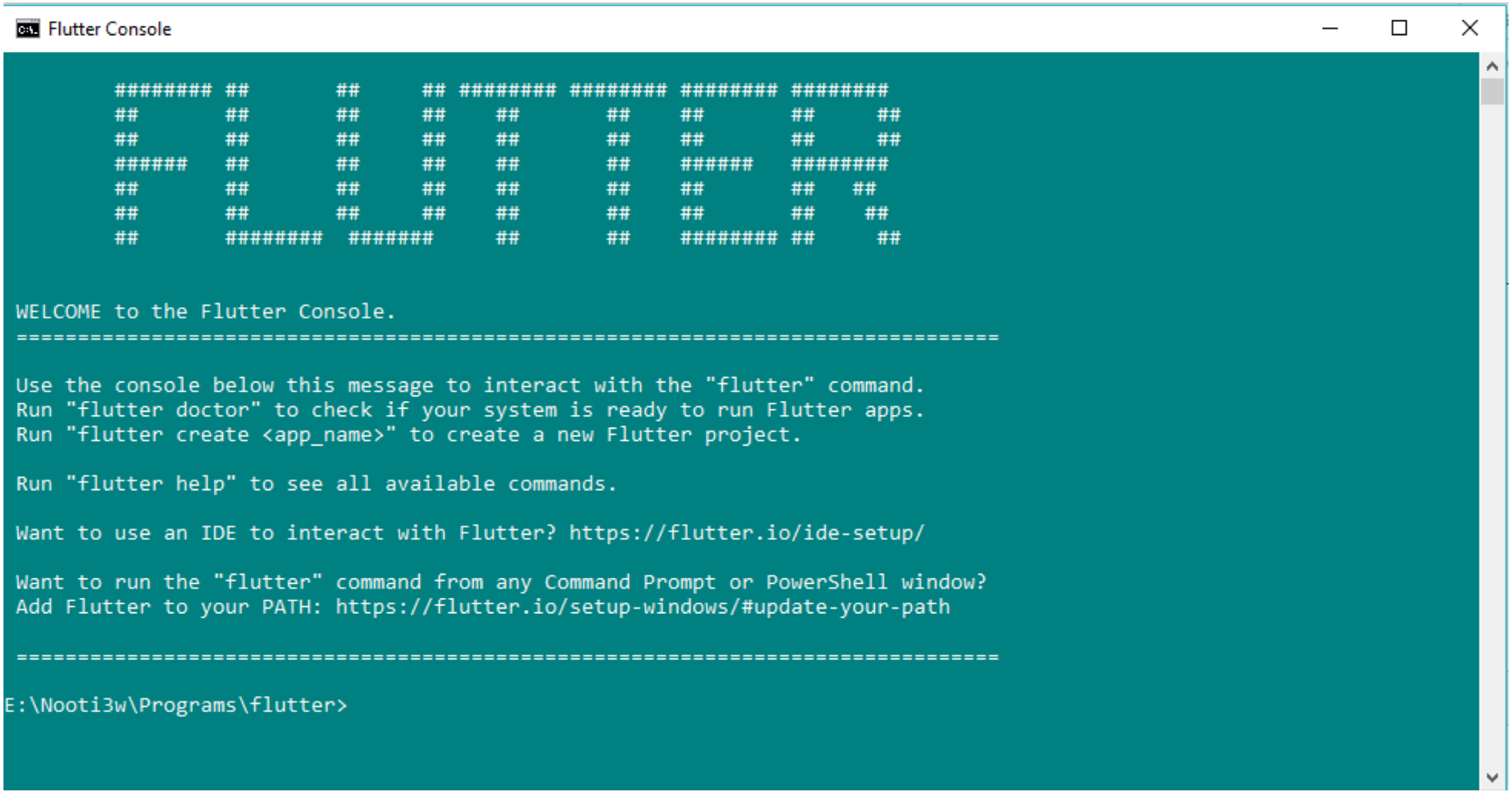


แตก file zip ที่ download มา

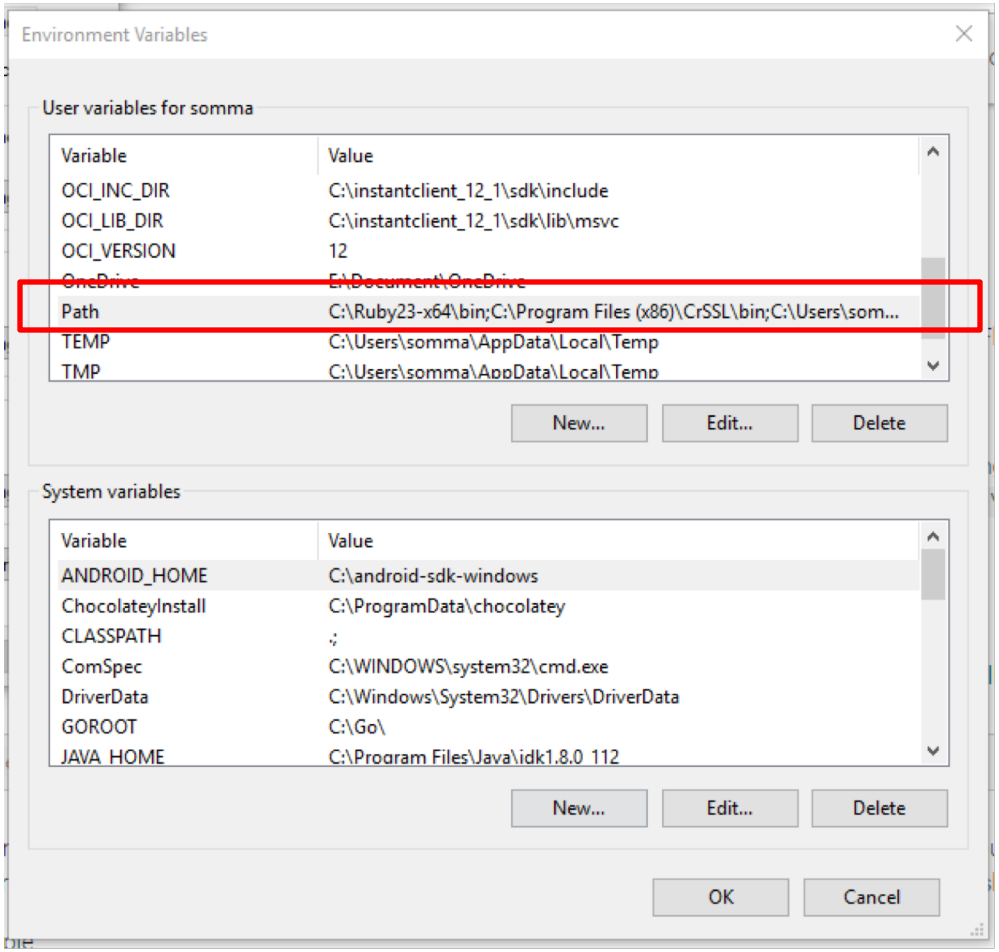


Name	Date modified	Type	Size
.git	9/19/2018 2:36 AM	File folder	
.github	9/19/2018 2:35 AM	File folder	
.idea	9/19/2018 2:40 AM	File folder	
.pub-cache	9/19/2018 2:35 AM	File folder	
bin	9/19/2018 2:35 AM	File folder	
dev	9/19/2018 2:35 AM	File folder	
examples	9/19/2018 2:35 AM	File folder	
packages	9/19/2018 2:35 AM	File folder	
.cirrus.yml	9/19/2018 2:35 AM	Yaml Source File	7 KB
.gitattributes	9/19/2018 2:35 AM	Git Attributes Sour...	1 KB
.gitignore	9/19/2018 2:35 AM	Git Ignore Source ...	2 KB
analysis_options.yaml	9/19/2018 2:35 AM	Yaml Source File	8 KB
appveyor.yml	9/19/2018 2:35 AM	Yaml Source File	1 KB
AUTHORS	9/19/2018 2:35 AM	File	2 KB
CONTRIBUTING.md	9/19/2018 2:35 AM	Markdown Source...	14 KB
flutter_console.bat	9/19/2018 2:35 AM	Windows Batch File	2 KB
flutter_root.iml	9/19/2018 2:35 AM	IML File	1 KB
LICENSE	9/19/2018 2:35 AM	File	2 KB
PATENTS	9/19/2018 2:35 AM	File	2 KB
README.md	9/19/2018 2:35 AM	Markdown Source...	7 KB
version	9/19/2018 2:41 AM	File	1 KB

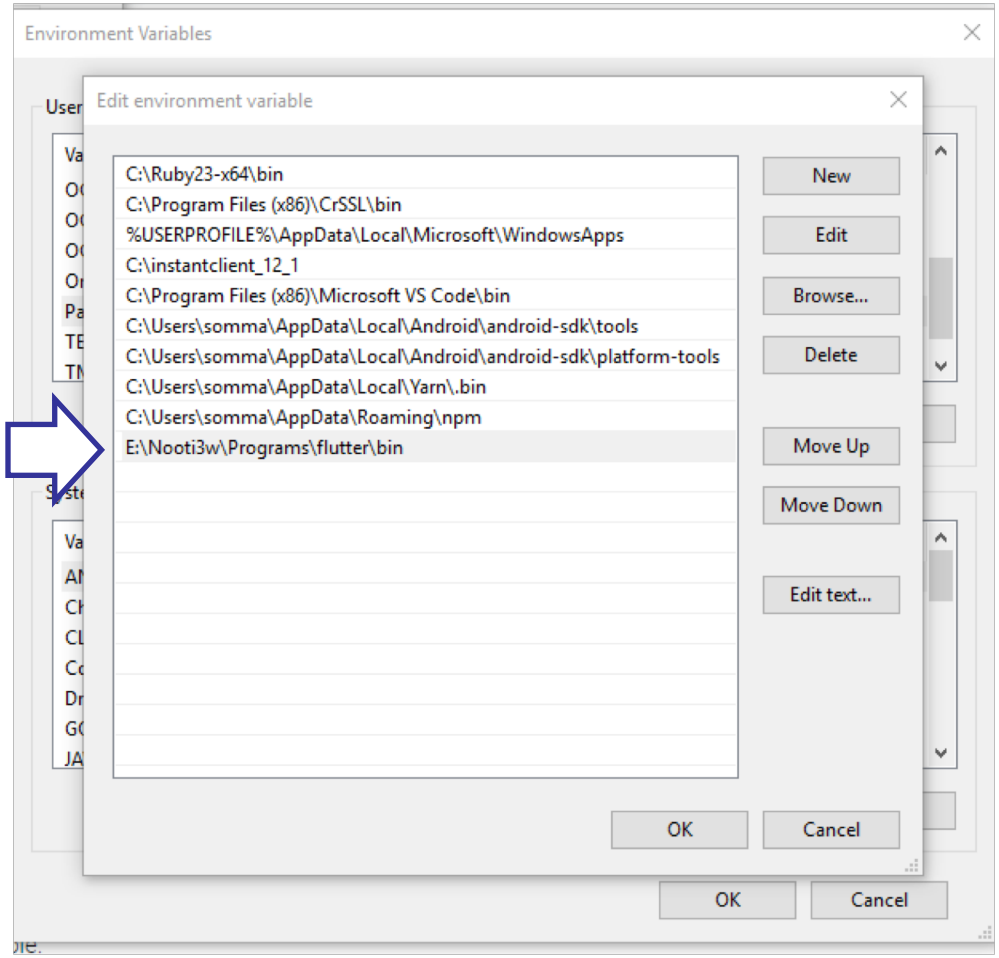
run คำสั่ง flutter_console



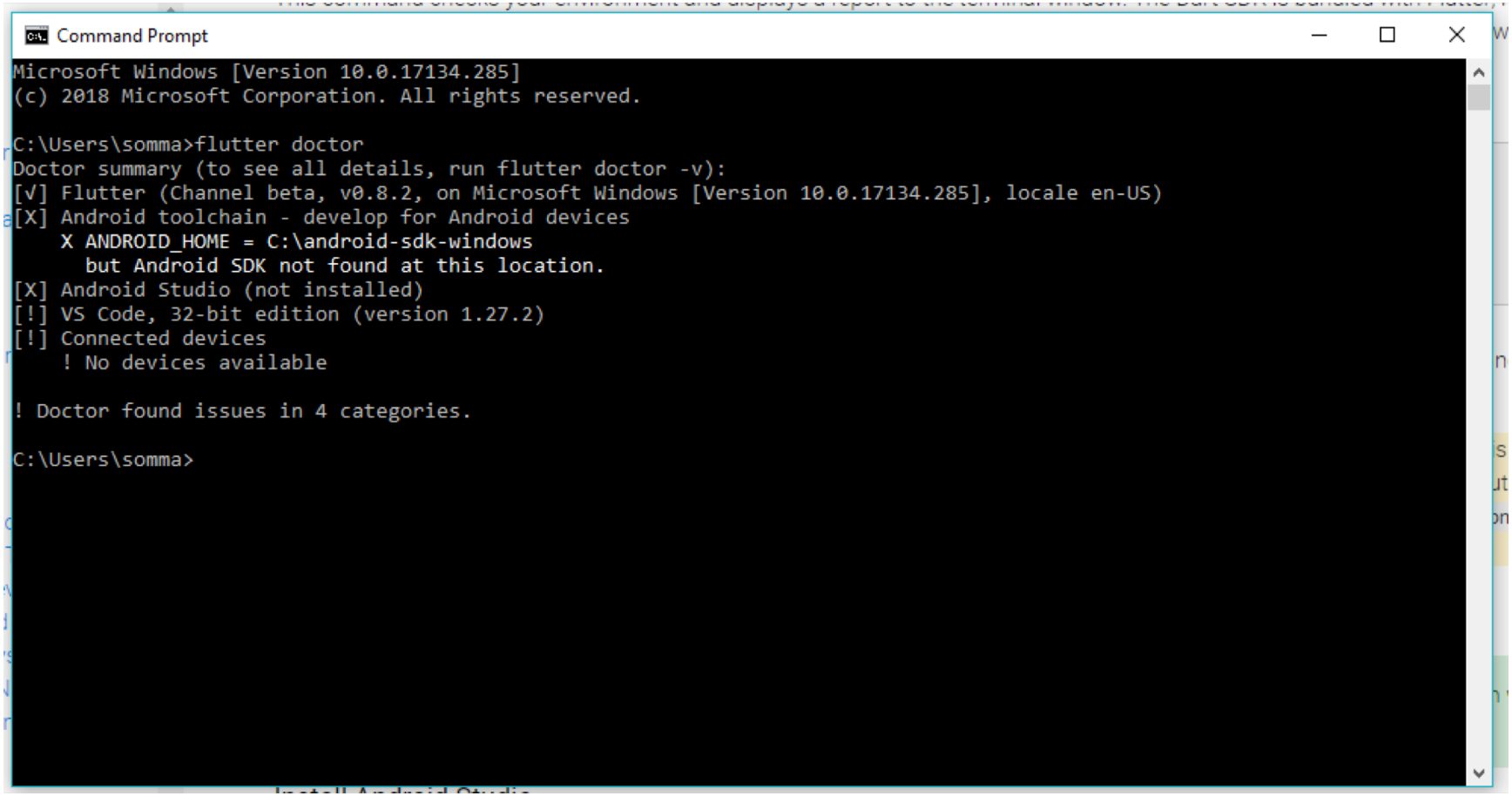
ตั้งค่า system path



ให้ชี้มาที่ flutter/bin



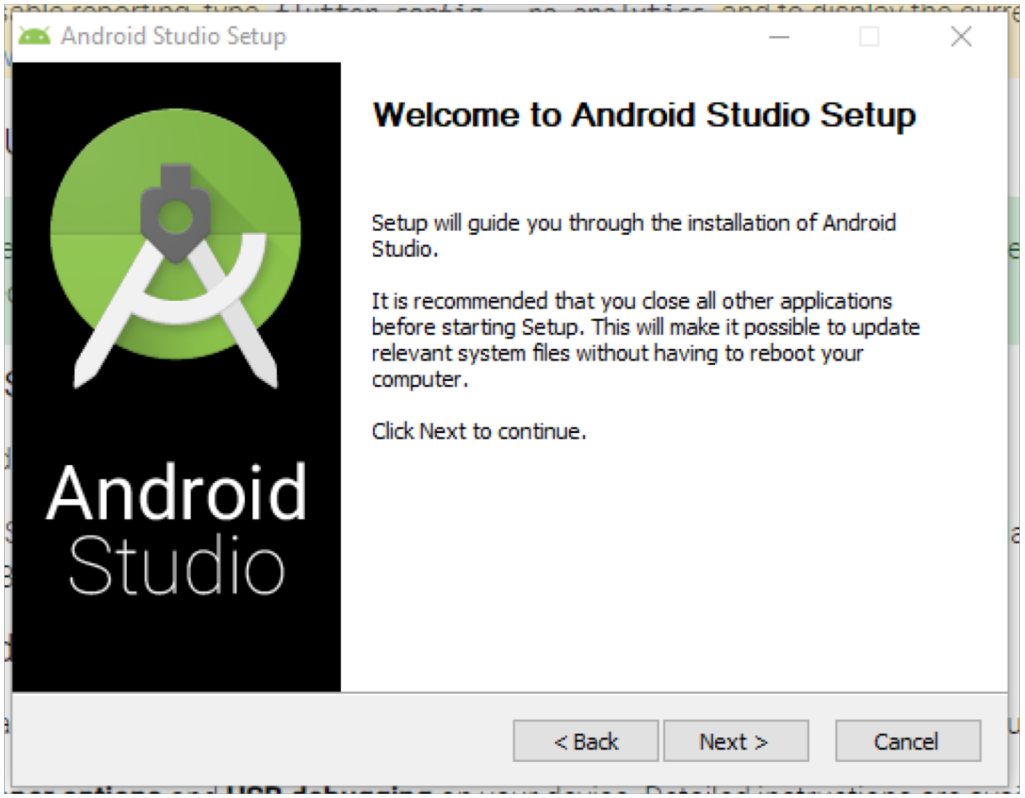
ตรวจสอบการติดตั้งด้วยคำสั่ง flutter doctor



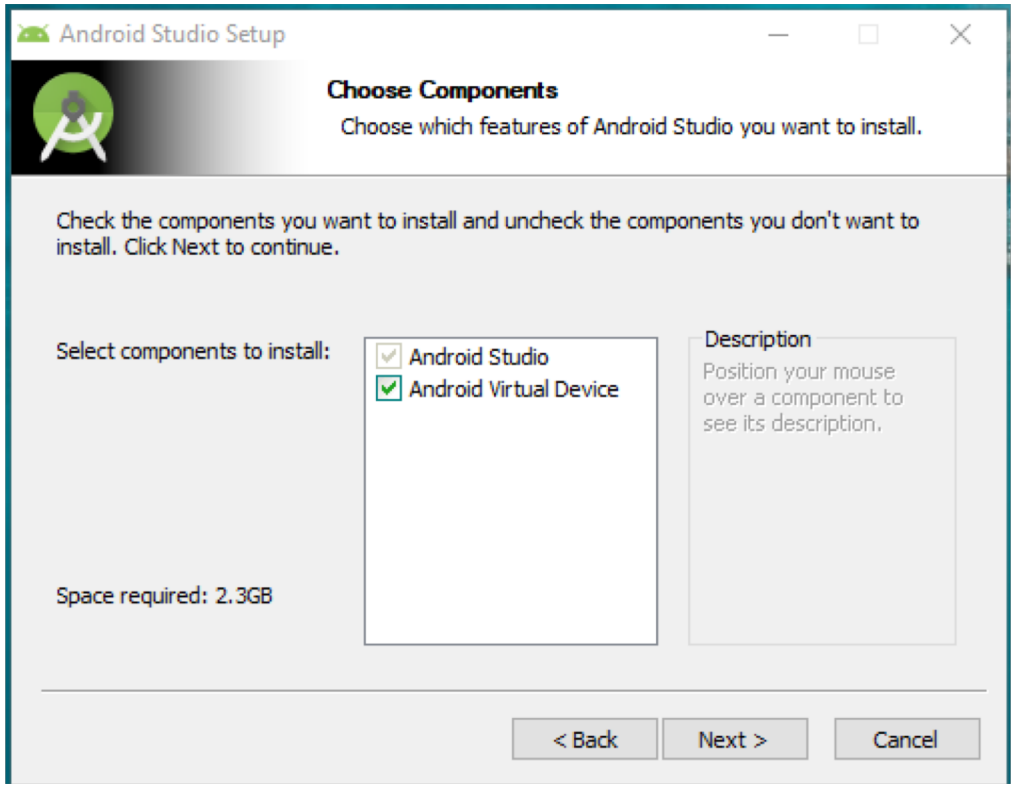
การติดตั้ง

ANDROID STUDIO

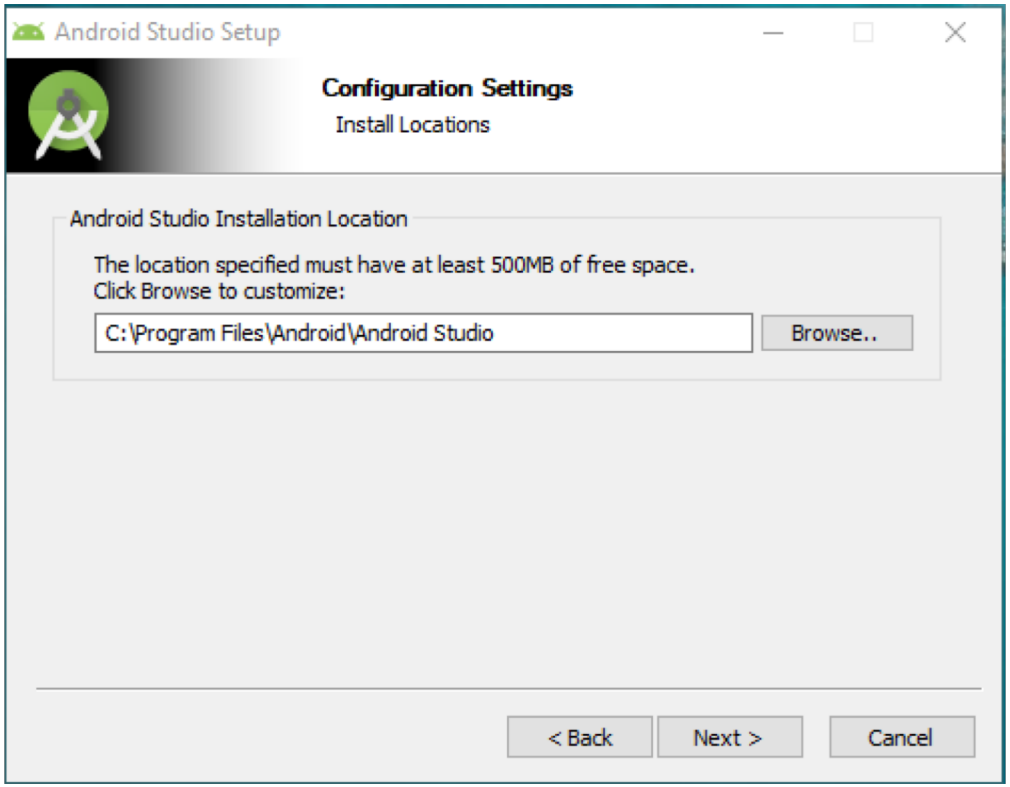
ติดตั้ง Android Studio #1



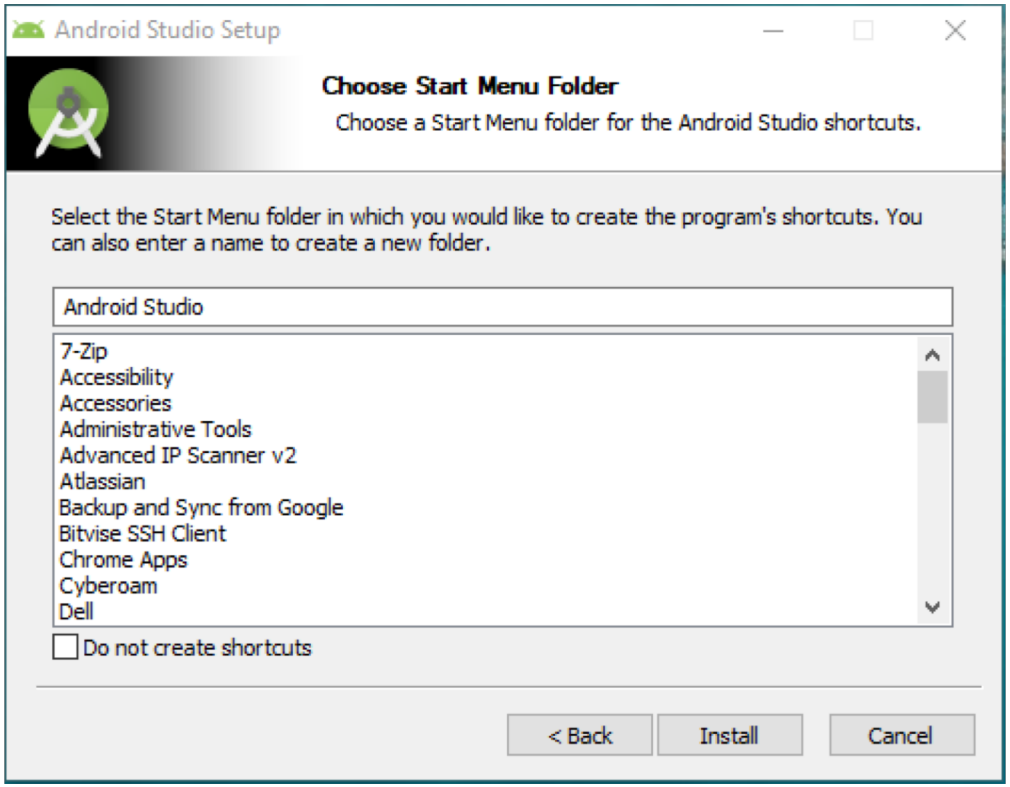
ติดตั้ง Android Studio #2



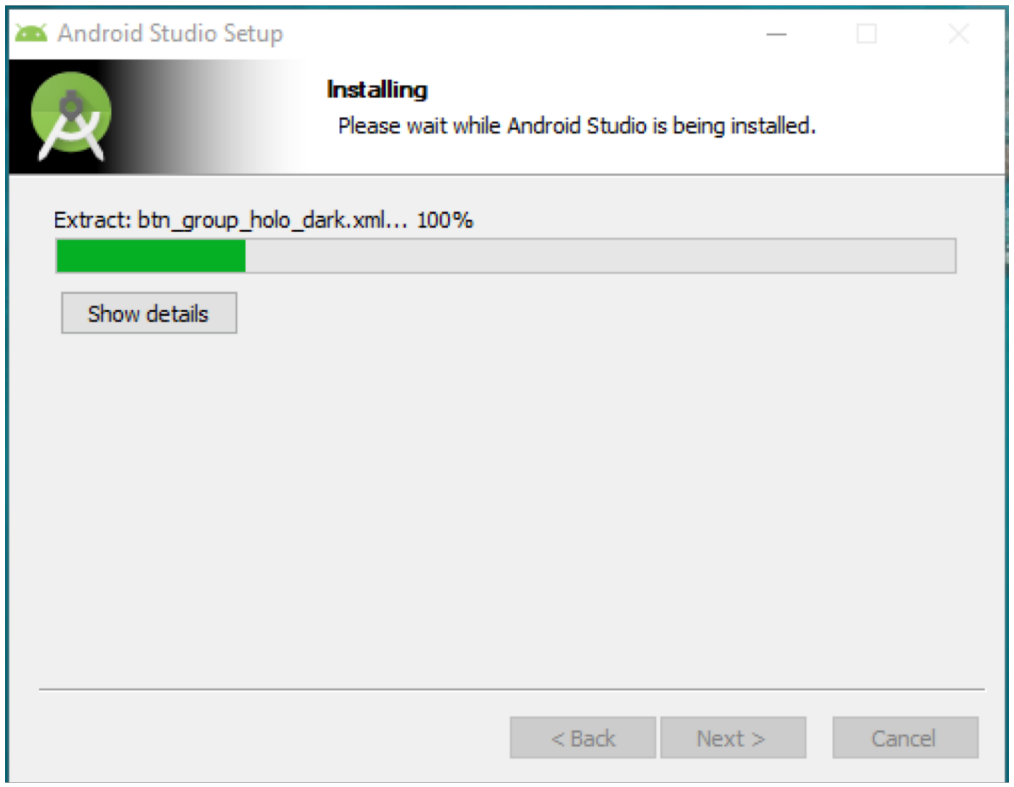
ติดตั้ง Android Studio #3



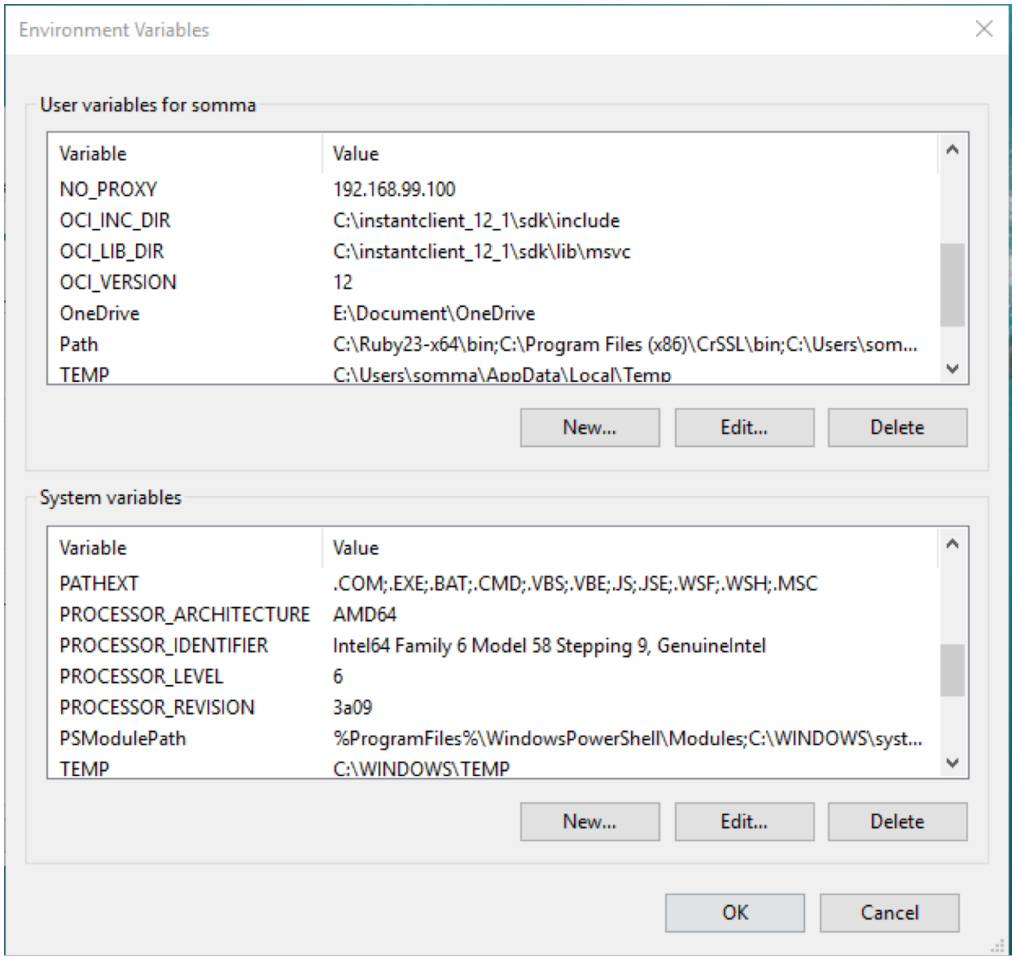
ติดตั้ง Android Studio #4



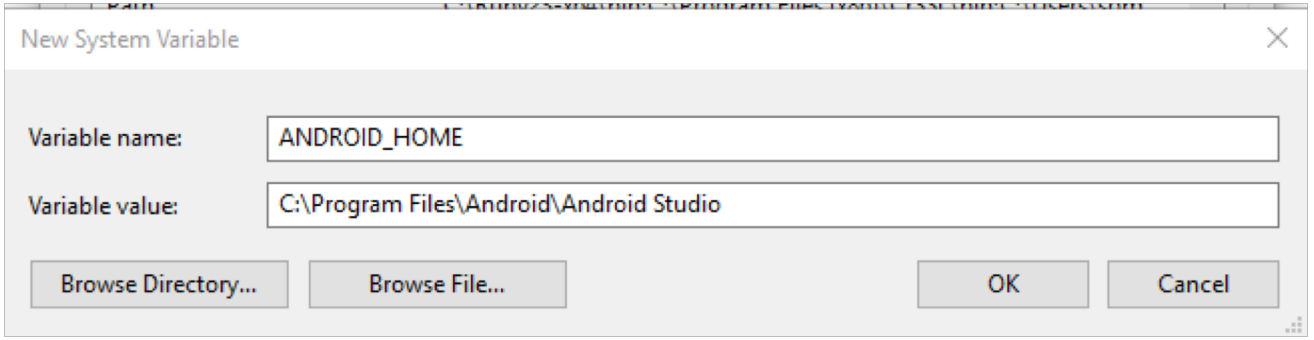
ติดตั้ง Android Studio #5



ตั้งค่า System Path



เพิ่ม ANDROID_HOME ให้ชี้ไปที่ path ที่ติดตั้ง Android studio



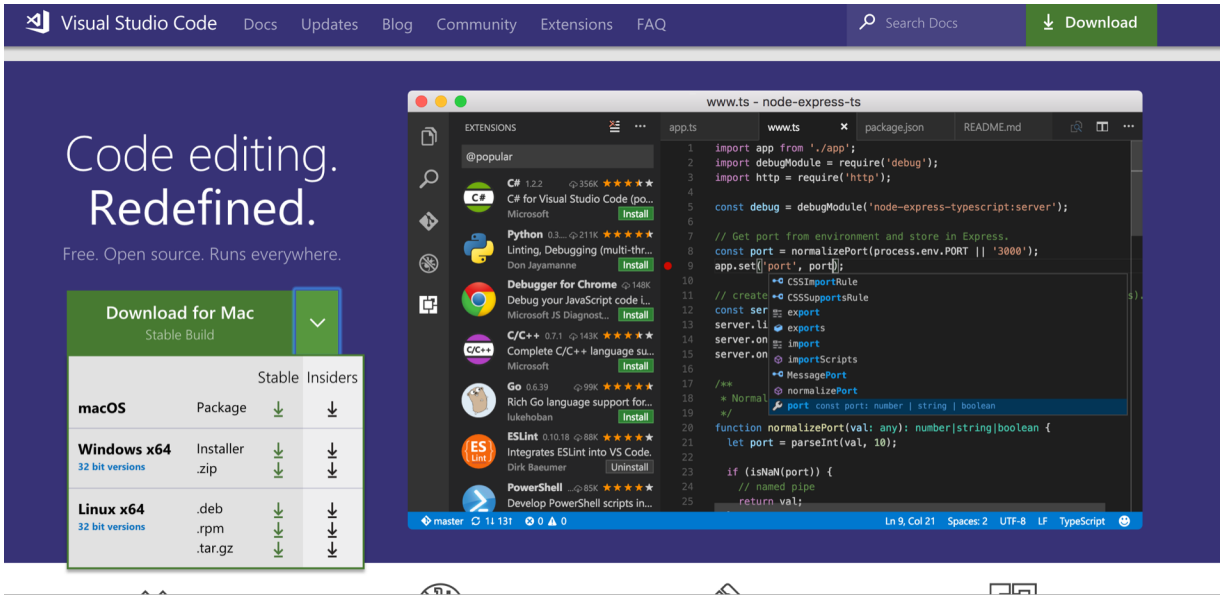
การติดตั้ง

VISUAL STUDIO CODE

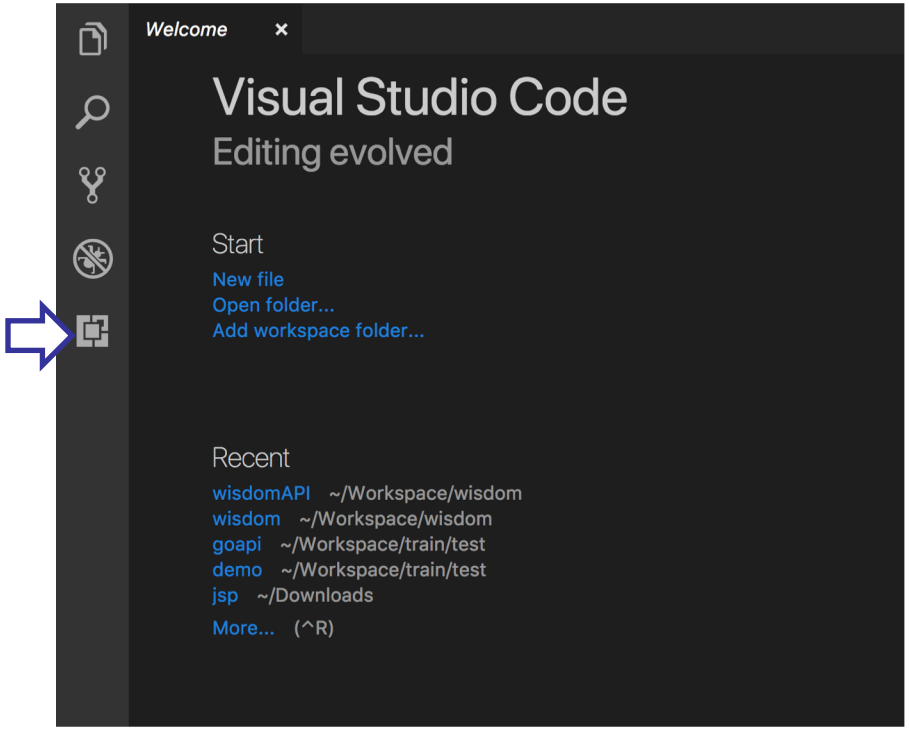
การติดตั้ง VSC

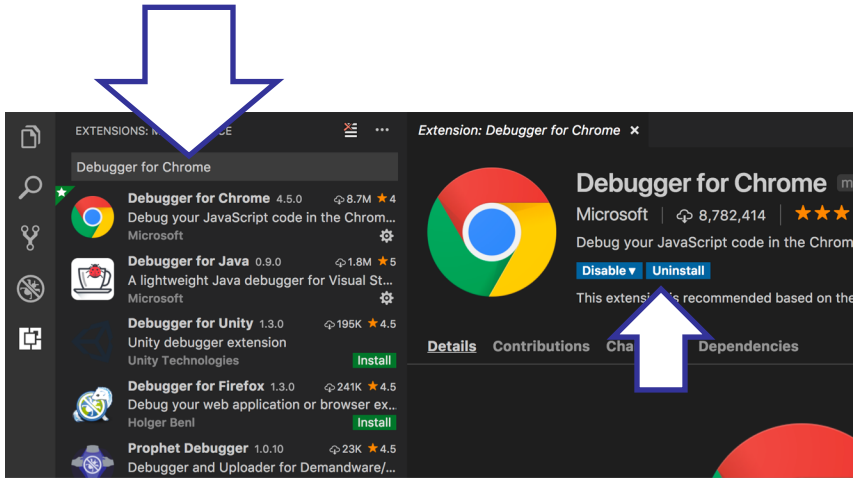
เข้าไปที่ website https://code.visualstudio.com/

เลือก download สำหรับ windows (stable)



Install Extensions



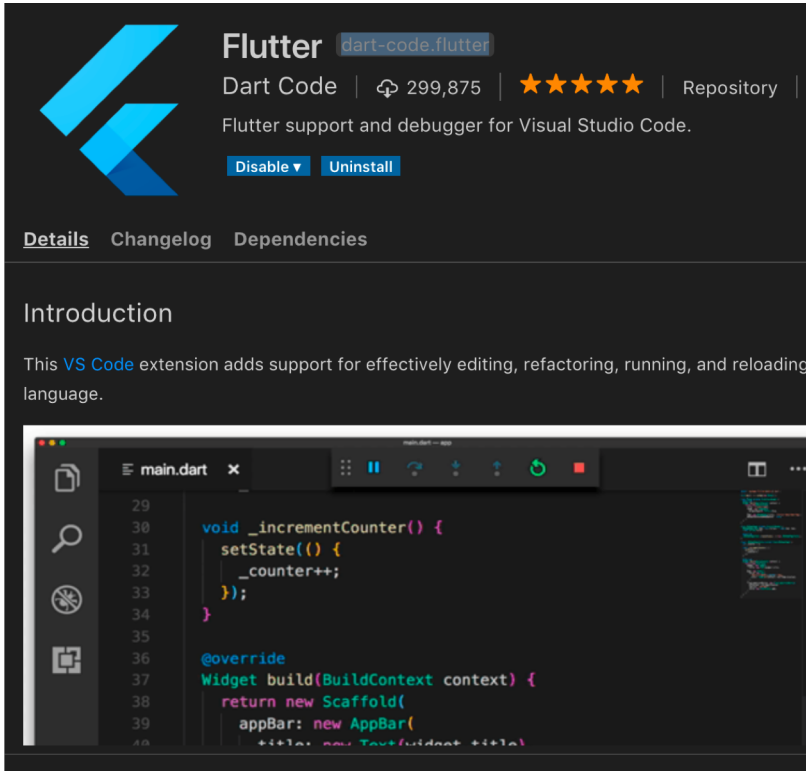


Install Extensions

Visual Studio Code Extensions

Dart [dart-code.dart-code]

Flutter [dart-code.flutter]

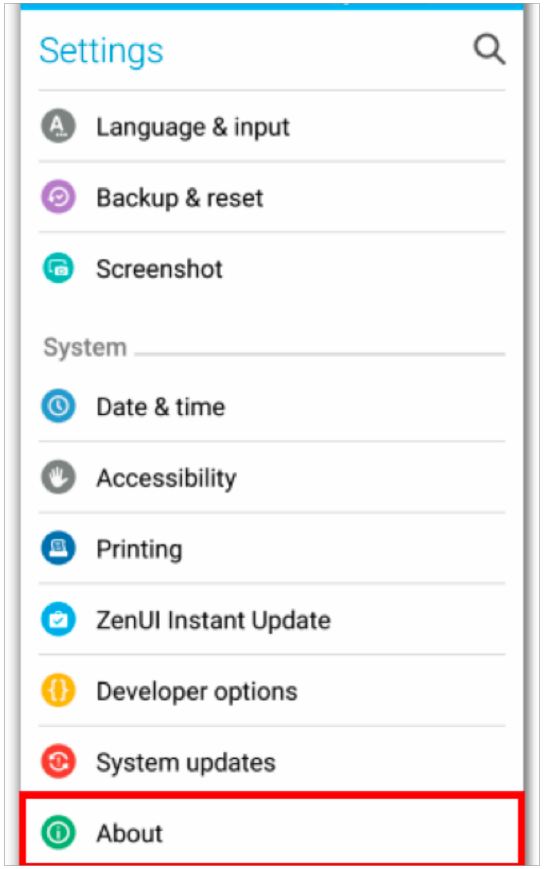


วิธีการตั้งค่า

DEVELOPER MODE ใน ANDROID

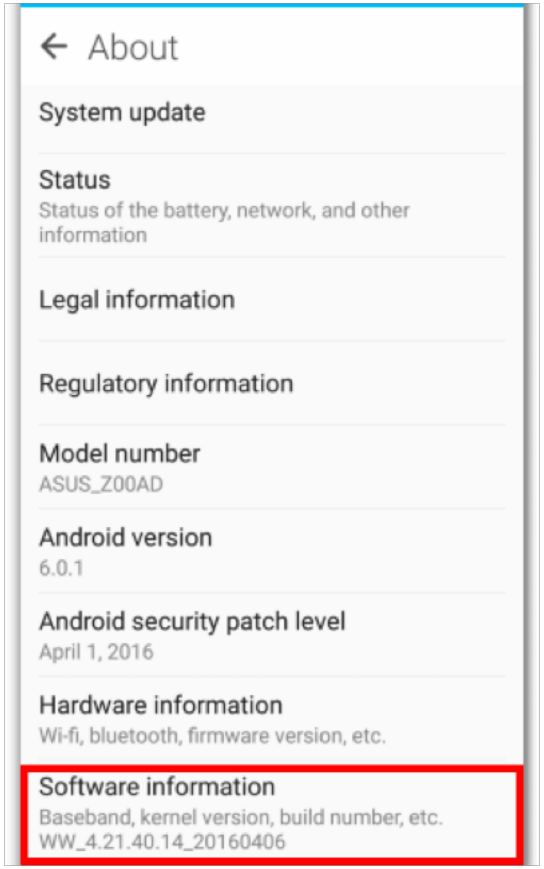
ตั้งค่า developer mode

- เลือก Settings / About



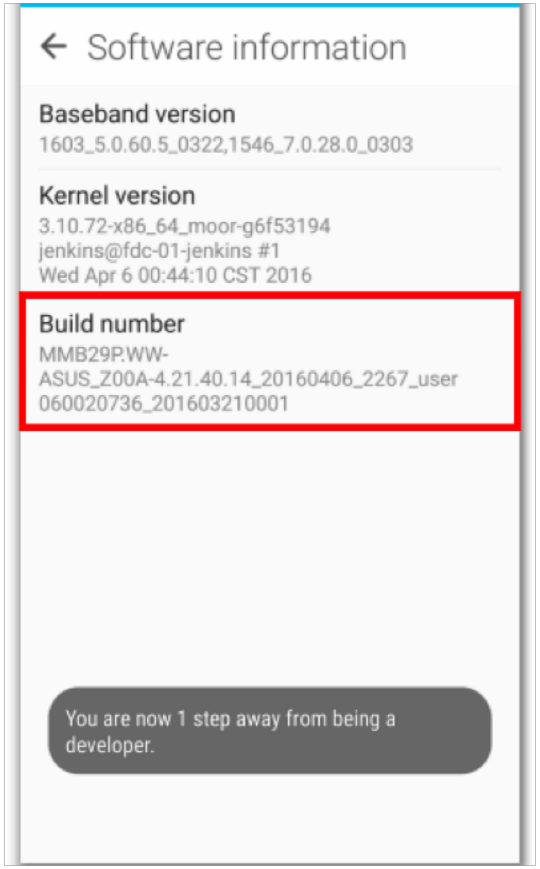
ตั้งค่า developer mode #2

- เลือก software information



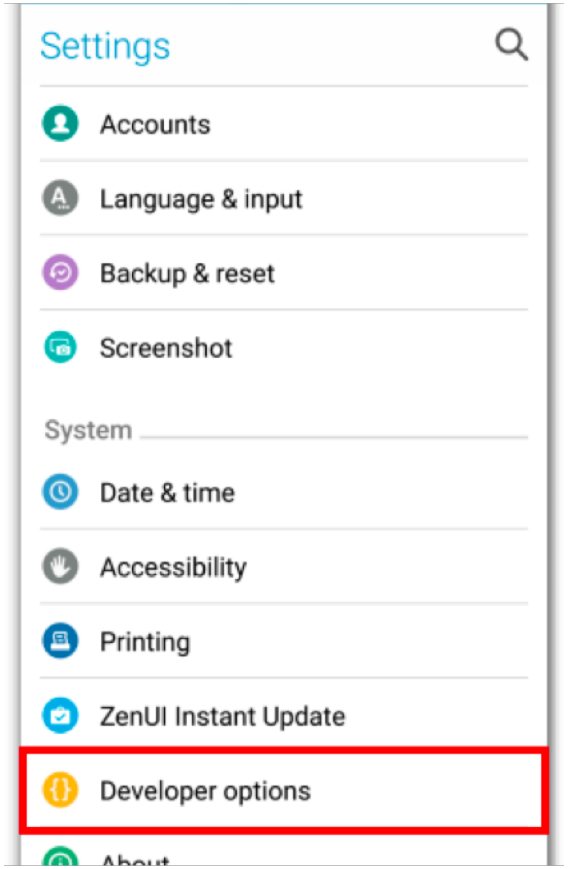
ตั้งค่า developer mode #3

- เคาะที่ Build number 7 ครั้งเพื่อเปิด mode developer



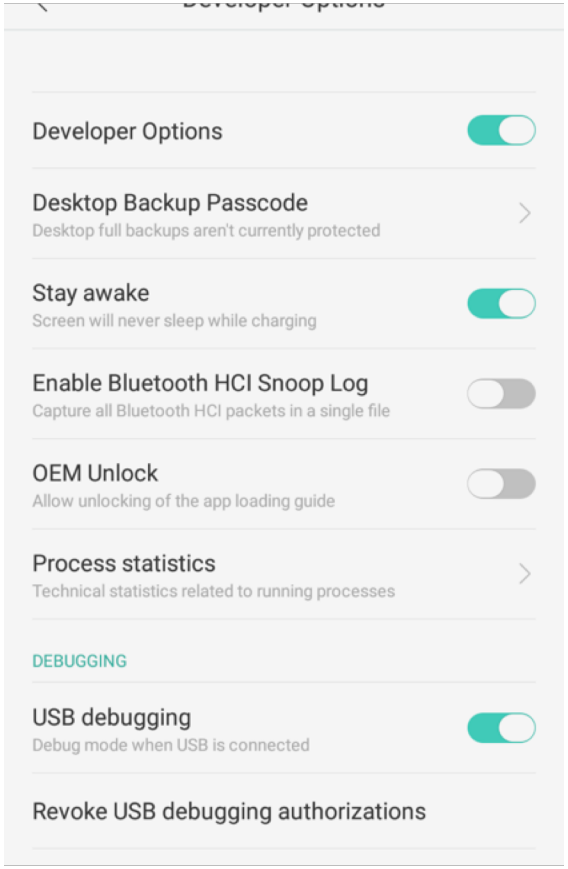
ตั้งค่า developer mode #4

- เลือก Developer options เพื่อตั้งค่าใช้งาน



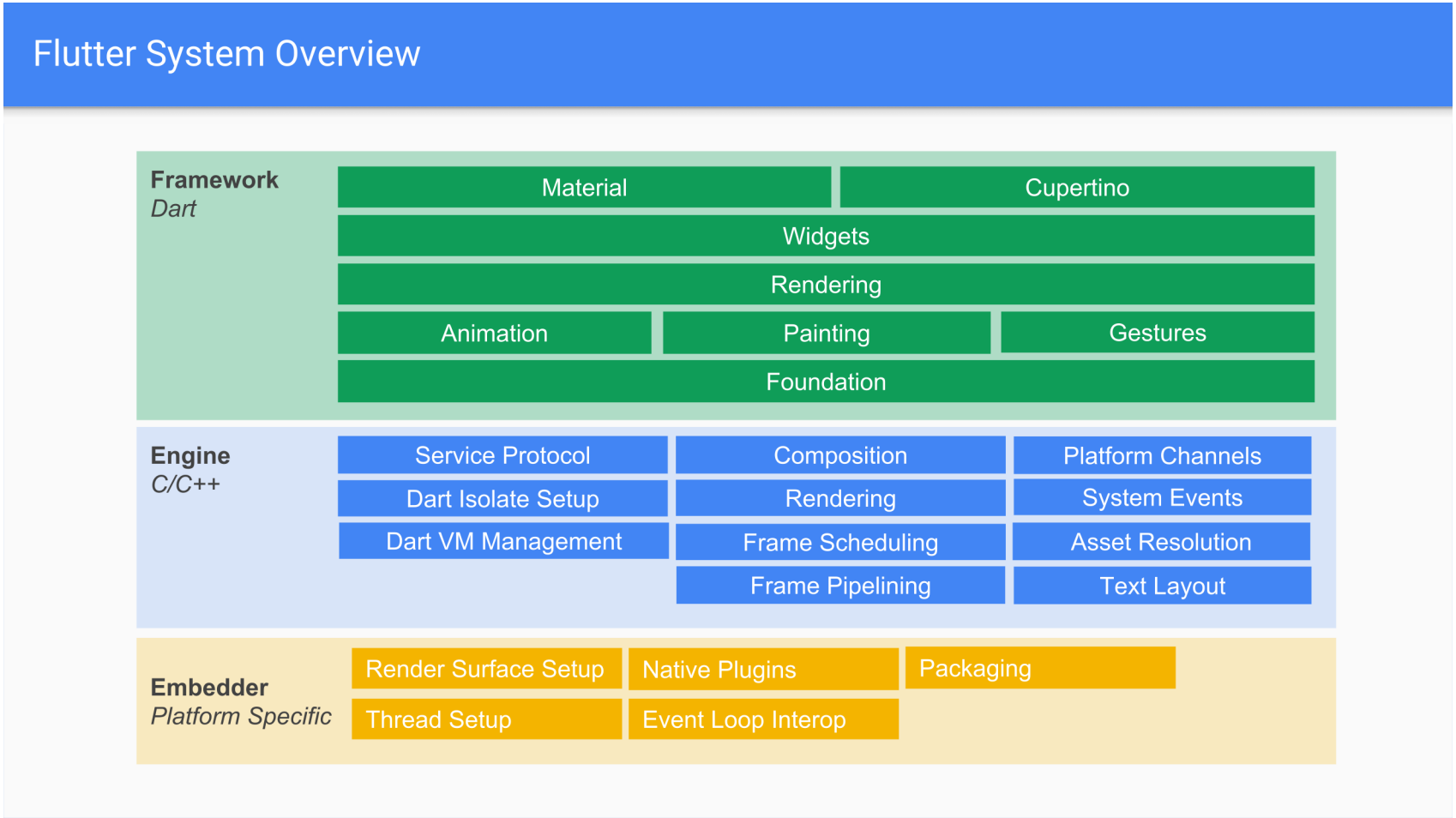
ตั้งค่า developer mode #5

- เปิดการใช้งานดังนี้
- Developer Options
- Stay awake
- USB debugging



INTRODUCTION TO FLUTTER

Flutter System Overview



Dart Programming Language

WHAT IS DART?

- Dynamic scripting language (runs in a VM)
- Intended as successor to Javascript
  - Better performance, better tooling, better for larger projects
  - Built by people who built V8 and Java HotSpot
- Oct 2011: Announced by Google
- Nov 2013: Stable v1.0 release
- Dec 2013: Setup ECMA TC52 for standardization
- www.dartlang.org
- dartpad.dartlang.org

A basic dart program

- รูปแบบการเขียนเหมือนภาษา JavaScript เช่น if, for, comment
- โปรแกรมจะเริ่มทำงานภายใต้ method void main(){...}
- method print(...) สำหรับแสดงผลที่ console
- dart ไม่มี public, private, protected ถ้าอยากให้ตัวแปร private ให้เติม _ ด้านหน้าตัวแปร และ class ต้องอยู่ใน library
- dart รองรับการเขียนแบบ OOP
- dart สามารถติดตั้ง lib เพิ่มเติมผ่าน file pubspec.yaml

Hello Dart World

```
void main() {
  String greeting = 'Hello Dart World';
  print(greeting);
}
```

Variables

- ตัวแปรใน dart จะมี type หรือไม่มีก็ได้ เช่น
  - var name = 'John';
  - String name = 'John';
  - dynamic name = 'John';

final and const

- ใช้ final หรือ const สำหรับตัวแปรที่ไม่ต้องมีการเปลี่ยนแปลงค่า
- ทั้ง final และ const ต้องมีการใส่ค่าเริ่มต้น
- final ต้องใส่ค่าก่อนที่ constructor จะเริ่ม
- const ต้องใส่ค่าเลย โดย const จะมีการตั้งค่าที่ระดับ compile time

```
void main() {
  const prefix = 'Hello';
  Data d = Data('value');
  print('${prefix} ${d.value}');
}

class Data {
  final value;
  Data(this.value);
}
```

Built-in types

- int
- double
- String
- bool
- List (array)
- Map (json)

int

```
void main() {
  int a = 10;
  int b = int.parse('10');
  print('$a + $b = ${a+b}');
  String c = a.toString();
  print(c);
}
```

double

```
void main() {
  double a = 10.5;
  double b = double.parse('10.5');
  print('$a + $b = ${a+b}');
  String c = a.toString();
  print(c);
}
```

string

- string ของ dart เก็บข้อมูลแบบ UTF-16
- string สามารถอ้างถึงตัวแปรภายนอกได้ผ่าน ${expression}, $expression

```
void main() {
  String a = 'Hello';
  String b = 'World';
  String c = '$a $b';
  print(c);
}
```

string

- เราสามารถ concat string ด้วย + หรือ เขียนต่อกันไปก่อน ;
- เราสามารถทำ multiline string ได้ด้วย '''

```
void main() {
  String a = 'Hello'+'World';
  String b = 'Hello' 'World';
  String c = '''
  Hello
  World
  ''';
}
```

bool

- ตัวแปร bool คือตัวแปรสำหรับเก็บค่า Boolean true/false

```
void main() {
  bool a = true;
  print(a);
}
```

List (array)

- dart เก็บ array ในรูปแบบ list
- สามารถอ้างถึงข้อมูลใน list ได้ทั้งแบบ [index] หรือผ่าน method elementAt(index)

```
void main() {
  List data = [1,2,3,4];
  List list = ['One', 'Two', 'Three'];
  print(list[0]);
  print(data.elementAt(0));
}
```

Map

- สามารถประกาศ map ได้โดยใช้ { 'key' : 'value' } หรือ Map()
- วิธีการอ้างถึงใช้การอ้างผ่าน [key]

```
void main() {
  Map data = {
    'code' : '007'
  };
  print(data['code']);

  Map key = Map();
  key['code'] = '008';
  print(key['code']);
}
```

Control flow and function in Dart

Control flow

You can control the flow of your Dart code using any of the following:

- if and else
- for loops
- while and do-while loops
- break and continue
- switch and case
- assert
- conditional expressions

if and else

syntax

```
if (true) {
// true statement
} else {
// false statement
}
```

example

```
void main() {
  bool isActive = true;
  if(isActive){
    print('active');
  }else {
    print('not active');
  }
}
```

for loops

syntax

```
for (initial_count_value; termination-condition; step) {
//statements
}
```

example

```
void main() {
  List data = [1,2,3,4];
  for(int i=0; i<data.length; i++){
    print(data[i]);
  }
}
```

while and do-while loops

syntax

```
while (expression) {
        Statement(s) to be executed if expression is true
}
do {
        Statement(s) to be executed;
} while (expression);
```

example

```
void main() {
  int n = 10;
  while (n >= 0) {
    print('from while $n');
    n--;
  }
  do {
    print('from do-while $n');
    n--;
  } while (n >= 0);
}
```

break

- break เอาไว้สำหรับออกจำ flow การทำงาน

```
void main() {
   int i = 1;
   while(i<=10) {
      if (i % 5 == 0) {
         print("The first multiple of 5  between 1 and 10 is : ${i}");
         break ;
         //exit the loop if the first multiple is found
      }
      i++;
   }
}
```

continue

- continue จะเป็นการสั่งให้ไปเริ่มทำงานใน loop ใหม่ ไม่ต้องทำ statement ต่อจากนี้แล้ว

```
void main() {
  var num = 0;
  var count = 0;

  for (num = 0; num <= 20; num++) {
    if (num % 2 == 0) {
      continue;
    }
    print('odd $num');
    count++;
  }
  print(" The count of odd values between 0 and 20 is: ${count}");
}
```

switch and case

syntax

```
switch(variable_expression) {
        case constant_expr1: {  // statements; } break;
        default: { //statements; }
}
```

example

```
void main() {
  int status = 4;

  switch (status) {
    case 0: print('Normal'); break;
    case 1: print('Close'); break;
    default: print('Error');
  }
}
```

assert

- assert ใช้สำหรับตรวจสอบค่าถ้าเป็น false จะ throw exception ออกมาและหยุดการทำงาน
- assert ทำงานใน mode development เท่านั้น ใน production จะไม่ทำงาน

```
void main() {
  int status = 4;
  assert(status == 5, 'Status must be 5');
}
```

conditional expressions

- conditional expressions. คือการเขียน if-else ในรูปแบบสั้นๆ
- dart มีการใช้ conditional expressions อยู่ 2 แบบ คือ
- condition ? expr1 : expr2 ถ้า condition เป็น true จะได้ค่า expr1, false จะได้ค่า expr2
- expr1 ?? expr2 ถ้า expr1 เป็น null จะได้ค่า expr2

```
void main() {
  bool active = false;
  String a = 'Simple String';
  String b = 'Hello World';
  print( active? 'Active': 'Inactive');
  print( a??b);
}
```

Function in Dart [shorthand]

- For functions that contain just one expression, you can use a shorthand syntax:
- The => expr syntax is a shorthand for { return expr; }.
- The => notation is sometimes referred to as arrow syntax.

```
void main() {
      var loudify = (msg) => '${msg.toUpperCase()}';
      print(loudify('hello'));

      greet(String msg) => 'Hello Mr. $msg';
      print(greet('Sommai'));
}
```

Optional named parameters

- When defining a function, use {*param1*, *param2*, ...} to specify named parameters:

```
greet({String title, String msg}) => 'Hello $title $msg';
```

- When calling a function, you can specify named parameters using *paramName*: *value*. For example:

```
greet(title: 'Mr.',  msg: 'Sommai');
```

Optional positional parameters

- Wrapping a set of function parameters in [] marks them as optional positional parameters:

```
greet(String title, [String msg, String name]) => '$msg $title $name';
```

Default parameter values

- Your function can use = to define default values for both named and positional parameters.
- The default values must be compile-time constants. If no default value is provided, the default value is null.

```
greet(String title, [
        String msg = 'Hello',
        String name = 'John'
]) => '$msg $title $name';
```

The try / on / catch Blocks

syntax

```
try {
// ...
} on Exception catch (e) {
        print('Exception details:\n $e');
} catch (e, s) {
        print('Exception details:\n $e'); print('Stack trace:\n $s');
}
```

on ใช้สำหรับ เลือกจับ exception ที่ระบุ catch ใช้สำหรับจับ object ของ exception

```
main() {
  List a = [1, 2];
  try {
    a[2];
  } on RangeError catch (e, s) {
    print('Index not found');
    print(e);
  }
}
```

Dart OOP Programming

Declaring a Class

syntax

```
class class_name {
        <fields>  ตัวแปรที่อยู่ใน class
        <getters/setters>
        <constructors>
        <functions>
}
```

example

```
class Car {
  String engine;

        ค่า set ไปที่ field โดยตรง
  Car(this.engine);
  String display() => 'Engine is $engine' ;
}
```

Creating Instance of the class

syntax

```
var object_name = new class_name([ arguments ]);
class_name object_name = new class_name([ arguments ]);
```

จะมี new หรือไม่มีก็ได้

example

```
void main() {
  Car c = new Car('V8');
  print(c.display());

  Car b = Car('Twin cam 16 V');
  print(b.display());
}
```

Constructors

syntax

```
Class_name(parameter_list) {
        //constructor body
}
```

example

```
void main() {
        Car c = new Car('E1001');
}

class Car {
        Car(String engine) {
                print(engine);
        }
}
```

Named Constructors

syntax

Class_name.constructor_name(param_list)

example

```
void main() {
        Car c1 = new Car.namedConst('E1001');
        Car c2 = new Car();
}
class Car {
         Car() {
           print("Non-parameterized constructor invoked");
        }
        Car.namedConst(String engine) {
           print("The engine is : ${engine}");
        }
}
```

Named Parameter constructors

syntax

```
Class_name({param_name_list}) {
    //constructor body
}
```

example

```
class Car {
  String engine;

  Car({this.engine});
  String display() => 'Engine is $engine';
}
```

factory constructors

- factory constructor เอาไว้สำหรับให้ class สร้าง instance ของ class ตัวเอง

```
void main() {
  Data d = Data.create("007", "Jame bond");
  print(d.code);
}

class Data {
  String code;
  String name;

  Data({this.code, this.name});

  factory Data.create(String code, String name) {
    return Data(code: code, name: name);
  }
}
```

Getters and Setters

syntax getter

Return_type get identifier { }

syntax setter

set identifier { }

example

```
class Car {
  String engine;
  Car({this.engine});
  String get display => 'Engine is $engine';
  set color(String color) => print('set $color');
}

void main() {
  Car c = new Car(engine: 'V8');
  c.color = 'Red';
  print(c.display);
}
```

Class Inheritance

syntax

```
class child_class_name extends parent_class_name
```

example

```
class Shape {
  void cal_area() {
    print("calling calc area defined in the Shape class");
  }
}

class Circle extends Shape {}

void main() {
  var obj = new Circle();
  obj.cal_area();
}
```

Class Inheritance and Method Overriding

example

```
void main() {
  Child c = new Child();
  c.m1(12);
}

class Parent {
  void m1(int a) {
    print("value of a ${a}");
  }
}

class Child extends Parent {
  @override
  void m1(int b) {
    print("value of b ${b}");
  }
}
```

The static Keyword

example

```
class StaticMem {
  static int num;
  static disp() {
    print("The value of num is ${StaticMem.num}");
  }
}

void main() {
  StaticMem.num = 12;
  // initialize the static variable }
  StaticMem.disp();
  // invoke the static method
}
```

The super, this Keyword

super keyword เอาไว้สำหรับให้ child class อ้างถึงตัวแปรของ parrent
this keyword เอาไว้สำหรับอ้างถึงตัวแปรของ class ตัวเอง

```
void main() {
  Child c = new Child();
  c.m1(12);
}

class Parent {
  String msg = "message variable from the parent class";
  void m1(int a) {
    print("value of a ${a}");
  }
}

class Child extends Parent {
  String title = 'Child title';
  @override
  void m1(int b) {
    print("${this.title} value of b ${b}");
    super.m1(13);
    print("${super.msg}");
  }
}
```

abstract class

```
void main() {
  Data d = Data("007", "Jame bond");
  print(d.display());
}

class Data extends More {
  String code;
  String name;

  Data(this.code, this.name);
  String display() {
    return this.code;
  }
}

abstract class More {
  String display();
}
```

interface class

```
void main() {
  Calculator c = new Calculator();
  print("The gross total : ${c.ret_tot()}");
  print("Discount :${c.ret_dis()}");
}

class Calculate_Total {
  int ret_tot() => 0;
}

class Calculate_Discount {
  int ret_dis() => 0;
}

class Calculator implements Calculate_Total, Calculate_Discount {
  int ret_tot() {
    return 1000;
  }

  int ret_dis() {
    return 50;
  }
}
```

Generics

เพื่อกำหนดให้ collection เก็บค่าตัวแปรที่อยู่ในรูปแบบเดียวกัน

syntax

Collection_name <data_type> identifier= new Collection_name<data_type>

example

```
main() {
  List<int> a = [1, 2, "xxx"]; // จะเกิด error
  for (int i = 0; i < a.length; i++) {
    print(a[i]);
  }
}
```

The Cascade operator (..)

.. เอาไว้อ้างถึง method ของ object ที่ถูกสร้างขึ้นมา

```
void main() {
  Car(model: 'Colora', engine: 'V8')
    ..build()
    ..display() ;
}

class Car {
  String model;
  String engine;
  String fullSpec;

  Car({this.model, this.engine});

  void build() {
    this.fullSpec = '$model $engine';
  }

  void display() => print('full spec = $fullSpec');
}
```

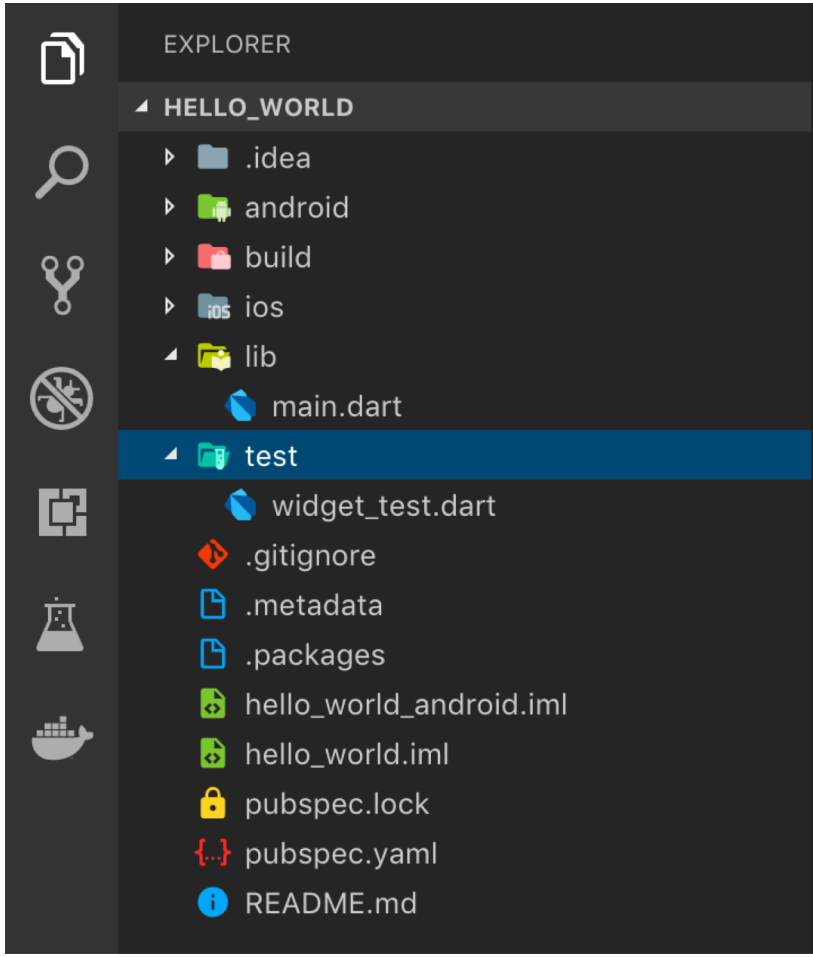
Flutter

new project

- command line
  - flutter create project_name
  - flutter create --org com.example project_name

- new project from visual studio code
  - กด ctrl + shift + p หรือ command + shift + p
  - พิมพ์ Flutter: New Project
  - พิมพ์ ชื่อ project ที่ต้องการสร้าง
  - เลือก path ที่ต้องการสร้าง project

Dart package, library and project structure



- folder lib เอาไว้เก็บ source code
- folder test เอาไว้เก็บ test code
- file pubspec.yaml เอาไว้ตั้งค่า lib package ต่างๆ

Basic Layouts and Container Widgets

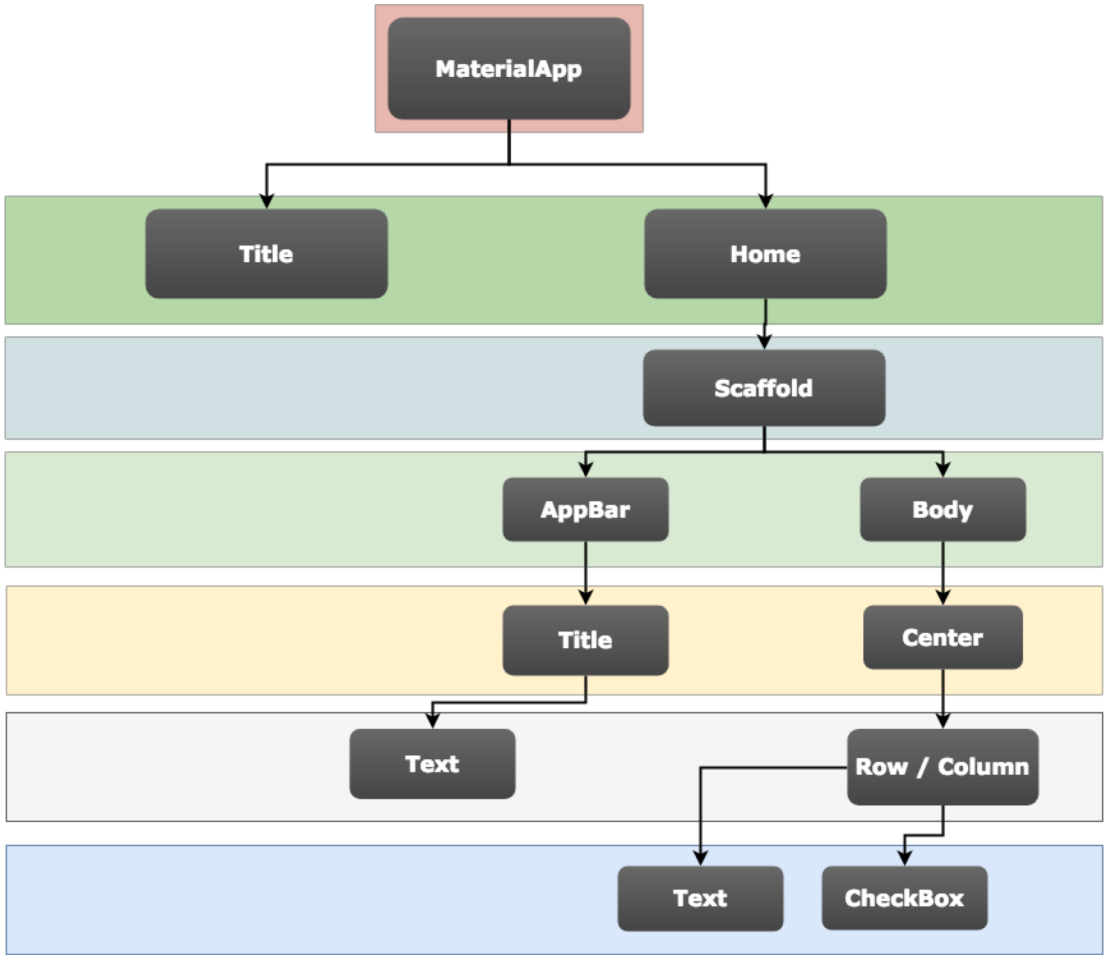
```dart
import 'package:flutter/material.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        primarySwatch: Colors.teal,
      ), // ThemeData
      home: MyHomePage(),
    ); // MaterialApp
  }
}
```

```dart
class MyHomePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Hello World'),
      ), // AppBar
      body: Text('Body'),
    ); // Scaffold
  }
}
```

Basic Layouts and Container Widgets #2

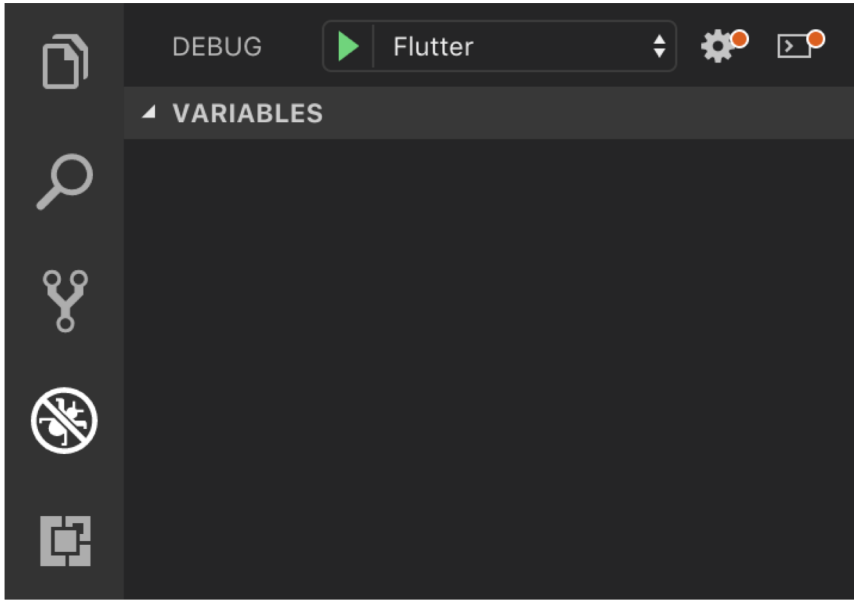


Run flutter

command line

- flutter run
- กด r เพื่อ hot reload
- กด R เพื่อ restart application
- กด q เพื่อออกจากการทำงาน

debug from VSC



Flutter Stateless and Stateful Widgets

- Stateless is a widget that does not require mutable state.
- Stateful is a widget that has mutable state.
- Stateful promptly notified when such state changes, using State.setState.

StatelessWidget

```
class GreenFrog extends StatelessWidget {
        const GreenFrog({ Key key }) : super(key: key);
                @override Widget build(BuildContext context) {
                return Container(color: Colors.blue);
        }
}
```

Stateful

```
class YellowBird extends StatefulWidget {
        const YellowBird({ Key key }) : super(key: key);
        @override _YellowBirdState createState() => _YellowBirdState();
}

class _YellowBirdState extends State<YellowBird> {
        @override Widget build(BuildContext context) {
                return Container(color: const Colors.yellow);
        }
}
```

Flutter and Material Design Widgets

- Scaffold
- AppBar
- Center
- Text
- Icon
- Container
- Row
- Column
- RaisedButton
- Tab
- Navigation
- Form with validate
- Text Form field
- Inputdecoration
- Radio
- Checkbox
- Dropdown
- Date picker
- List view
- Image
- Image picker

Scaffold, AppBar, Center, Text, Icon

```dart
class MyHomePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Hello World'),
      ), // AppBar
      body: Center(
        child: Text('Body'),
      ), // Center
      floatingActionButton: FloatingActionButton(
        child: Icon(Icons.add),
        onPressed: () {},
      ), // FloatingActionButton
    ); // Scaffold
  }
}
```

Container, Row, Column, RaisedButton

```
Container(
        child: Row(
                mainAxisAlignment: MainAxisAlignment.center,
                children: <Widget>[
                        Column(
                                mainAxisAlignment: MainAxisAlignment.center,
                                children: <Widget>[
                                 RaisedButton(
                                    child: Text('One'),
                                    onPressed: () => print('One Click'),
                                ),
                                  Text("Two"),
                                ],
                        )
                ],
        ),
)
```

Tab

กล่องใหญ่รวมทั้งหมด

ใต้แอปบา เป็นปุ่ม

การวาดโครง

คือเอาตรงที่ tap ถูกกดมาแสดง

```
class TabBarDemo extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: DefaultTabController(
        length: 3,
        child: Scaffold(
          appBar: AppBar(
            bottom: TabBar(
              tabs: [
                Tab(icon: Icon(Icons.directions_car)),
                Tab(icon: Icon(Icons.directions_transit)),
                Tab(icon: Icon(Icons.directions_bike)),
              ],
            ),
            title: Text('Tabs Demo'),
          ),
          body: TabBarView(
            children: [
              Icon(Icons.directions_car),
              Icon(Icons.directions_transit),
              Icon(Icons.directions_bike),
            ],
          ),
        ),
      ),
    );
  }
}
```

Navigation

- Define the routes

```
MaterialApp(
  // Start the app with the "/" named route. In our case, the app will start
  // on the FirstScreen Widget
  initialRoute: '/',
  routes: {
    // When we navigate to the "/" route, build the FirstScreen Widget
    '/': (context) => FirstScreen(),
    // When we navigate to the "/second" route, build the SecondScreen Widget
    '/second': (context) => SecondScreen(),
  },
);
```

Navigation #2

- Navigate to the screen

```
// Within the `FirstScreen` Widget
onPressed: () {
  // Navigate to the second screen using a named route
  Navigator.pushNamed(context, '/second');
}
```

```
onPressed: () {
  Navigator.pushReplacementNamed(context, '/login');
},
```

Navigation #3

- Navigate with passing data

```
Navigator.push(
  context,
  MaterialPageRoute(
    builder: (context) => DetailScreen(todo: todos[index]),
  ),
);
```

Navigation #4

- Return to the previous screen

```
// Within the SecondScreen Widget
onPressed: () {
  // Navigate back to the first screen by popping the current route
  // off the stack
  Navigator.pop(context);
}
```

Form with validate #1

- Create a Form with a GlobalKey

```dart
class MyCustomForm extends StatefulWidget {
  @override
  MyCustomFormState createState() {
    return MyCustomFormState();
  }
}

class MyCustomFormState extends State<MyCustomForm> {
  final _formKey = GlobalKey<FormState>();

  @override
  Widget build(BuildContext context) {
    return Form(
      key: _formKey,
      child: ListView(),
    ); // Form
  }
}
```

Form with validate #2

- Add a TextFormField with validation logic

```
TextFormField(
  // The validator receives the text the user has typed in
  validator: (value) {
    if (value.isEmpty) {
      return 'Please enter some text';
    }
  },
);
```

Form with validate #3

- Create a button to validate and submit the form

```
RaisedButton(
  onPressed: () {
    if (_formKey.currentState.validate()) {
      Scaffold.of(context)
          .showSnackBar(SnackBar(content: Text('Processing Data')));
    }
  },
  child: Text('Submit'),
) // RaisedButton
```

TextFormField, Inputdecoration

```
TextFormField(
  decoration: InputDecoration(
    icon: Icon(Icons.email),
    labelText: 'User Id',
    hintText: 'Please input your email'
  ), // InputDecoration
  keyboardType: TextInputType.emailAddress,
  onSaved: (value) => print(value),
), // TextFormField
```

Password Field

```
TextFormField(
  decoration: InputDecoration(
    icon: Icon(Icons.lock),
    labelText: 'Password',
    hintText: 'Please input your password',
  ), // InputDecoration
  obscureText: true,
  keyboardType: TextInputType.text,
  onSaved: (value) => print(value),
), // TextFormField
```

Radio

```
Radio(
  value: 1,
  groupValue: route,
  onChanged: (value) {
    setState(() {
      route = value;
    });
  },
), // Radio
Text("Round Trip"),
Radio(
  value: 0,
  groupValue: route,
  onChanged: (value) {
    setState(() {
      route = value;
    });
  },
), // Radio
Text("One Way"),
```

Checkbox

```
Checkbox(
  value: checkboxValueA,
  onChanged: (bool value) {
    setState(() {
      checkboxValueA = value;
    });
  },
),
```

Dropdown

- Create list value

```
List<String> _passengers = <String>['', '1', '2', '3', '4', '5'];
```

Dropdown #2

```
InputDecorator(
  decoration: const InputDecoration(
    icon: const Icon(Icons.person),
    labelText: 'Passenger',
  ), // InputDecoration
  isEmpty: _passenger == '',
  child: DropdownButtonHideUnderline(
    child: DropdownButton<String>(
      value: _passenger,
      isDense: true,
      onChanged: (String value) {
        setState(() {
          _passenger = value;
        });
      },
      items: _passengers.map((String value) {
        return DropdownMenuItem<String>(
          value: value,
          child: Text(value),
        ); // DropdownMenuItem
      }).toList(),
    ), // DropdownButton
  ), // DropdownButtonHideUnderline
), // InputDecorator
```

ListView

```
class MyList extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    // TODO: implement build
    return ListView(
      children: <Widget>[
        Text('One'),
        Text('Two'),
        Text('Three'),
      ], // <Widget>[]
    ); // ListView
  }
}
```

Date picker

- install dependencies

dependencies:

datetime_picker_formfield: ^0.1.4

- run command

flutter packages get

- import package in code

import 'package:datetime_picker_formfield/datetime_picker_formfield.dart';

Date picker #2

```
DateTimePickerFormField(
  format: dateFormat,
  decoration: InputDecoration(labelText: 'Date'),
  onChanged: (dt) => setState(() => date = dt),
),
```

Image picker

- install dependencies

dependencies:

image_picker: ^0.4.10

- run command

flutter packages get

- import package in code

import 'package:image_picker/image_picker.dart';

Image picker #2

- IOS
  - Add the following keys to your *Info.plist* file, located in <project root>/ios/Runner/Info.plist:
    - NSPhotoLibraryUsageDescription - describe why your app needs permission for the photo library. This is called *Privacy - Photo Library Usage Description* in the visual editor.
    - NSCameraUsageDescription - describe why your app needs access to the camera. This is called *Privacy - Camera Usage Description* in the visual editor.
    - NSMicrophoneUsageDescription - describe why your app needs access to the microphone, if you intend to record videos. This is called *Privacy - Microphone Usage Description* in the visual editor.

Image picker #3

```dart
import 'package:image_picker/image_picker.dart';

class MyHomePage extends StatefulWidget {
  @override
  _MyHomePageState createState() => new _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  File _image;

  Future getImage() async {
    var image = await ImagePicker.pickImage(source: ImageSource.camera);

    setState(() {
      _image = image;
    });
  }

  @override
  Widget build(BuildContext context) {
    return new Scaffold(
      appBar: new AppBar(
        title: new Text('Image Picker Example'),
      ),
      body: new Center(
        child: _image == null
            ? new Text('No image selected.')
            : new Image.file(_image),
      ),
      floatingActionButton: new FloatingActionButton(
        onPressed: getImage,
        tooltip: 'Pick Image',
        child: new Icon(Icons.add_a_photo),
      ),
    );
  }
}
```

Introduction to HTTP and JSON

- install dependencies

```
dependencies:
    http: ^0.11.3
```

- run command

```
flutter packages get
```

- import package in code

```
import 'package:http/http.dart' as http;
import 'dart:convert';
```

Introduction to HTTP and JSON #2

- get method

```
Future loadData() async {

    http.Response resp = await http.get("your server url");
    Map<String, dynamic> data = json.decode(resp.body);

}
```

Introduction to HTTP and JSON #3

- post method

```
Map<String, dynamic> postBody = {
  'key': 'value'
};

Map<String, dynamic> postHeader = {
  'Content-Type' : 'application/json',
  'Authorization' : 'token'
};

http.Response postResp = await http.post('url', body: postBody, headers: postHeader);

print(postResp.body);
```

Introduction to HTTP and JSON #4

- delete method

```
Future loadData() async {

    http.Response resp = await http.delete("your server url");
    Map<String, dynamic> data = json.decode(resp.body);

}
```

Introduction to HTTP and JSON #5

- put method

```
Map<String, dynamic> postBody = {
  'key': 'value'
};

Map<String, dynamic> postHeader = {
  'Content-Type' : 'application/json',
  'Authorization' : 'token'
};

http.Response postResp = await http.put('url', body: postBody, headers: postHeader);

print(postResp.body);
```

Data parsing and ListView Widgets

- create model
- parse JSON into a List of Objects
- fetch data
- create ListView widget
- Display data

Create Model

```
class Post {
  final int userId;
  final int id;
  final String title;
  final String body;

  Post({this.userId, this.id, this.title, this.body});

  factory Post.fromJson(Map<String, dynamic> json) {
    return Post(
      userId: json['userId'] as int,
      id: json['id'] as int,
      title: json['title'] as String,
      body: json['body'] as String,
    );
  }
}
```

Parse JSON into a List of Objects

```
List<Post> parsePosts(String responseBody) {
  final parsed = json.decode(responseBody).cast<Map<String, dynamic>>();

  return parsed.map<Post>((json) => Post.fromJson(json)).toList();
}
```

fetch data

```
Future<List<Post>> fetchPosts(http.Client client) async {
  final response = await client.get('https://jsonplaceholder.typicode.com/posts');

  // compute function to run parsePosts in a separate isolate
  return compute(parsePosts, response.body);
}
```

create ListView widget

```
class ListViewPosts extends StatelessWidget {
  final List<Post> posts;

  ListViewPosts({Key key, this.posts}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Container(
      child: ListView.builder(
          itemCount: posts.length,
          padding: const EdgeInsets.all(15.0),
          itemBuilder: (context, position) {
            return Column(
              children: <Widget>[
                Divider(height: 5.0),
                ListTile(
                  title: Text('${posts[position].title}'),
                  subtitle: Text('${posts[position].body}'),
                  leading: ...,
                  onTap: () => _onTapItem(context, posts[position]),
                ),
              ],
            );
          }),
    );
  }

  void _onTapItem(BuildContext context, Post post) { ... }
}
```

Display data

```
FutureBuilder<List<Post>>(
  future: fetchPosts(http.Client()),
  builder: (context, snapshot) {
    if (snapshot.hasError) print(snapshot.error);

    return snapshot.hasData
        ? ListViewPosts(posts: snapshot.data) // return the ListView widget
        : Center(child: CircularProgressIndicator());
  },
),
```

Flutter I/O Read Write to device

- install dependencies

dependencies:

path_provider: ^0.4.1

- run command

flutter packages get

- import package in code

import 'package:path_provider/path_provider.dart';

Flutter I/O Read Write to device #2

- หา application path

```
Future<String> get _localPath async {
final directory = await getApplicationDocumentsDirectory();
return directory.path;
}
```

- สร้าง file

```
Future<File> get _localFile async {
final path = await _localPath;
return File('$path/counter.txt');
}
```

Flutter I/O Read Write to device #3

- write สง file

```
Future<File> writeCounter(int counter) async {
    final file = await _localFile;
    return file.writeAsString('$counter');
}
```

Flutter I/O Read Write to device #3

- read จาก file

```
Future<int> readCounter() async {
    try {
        final file = await _localFile;
        String contents = await file.readAsString();
        return int.parse(contents);
    } catch (e) {
        return 0;
    }
}
```

Data Persistence in Flutter

- install dependencies

dependencies:

shared_preferences: ^0.4.3

- run command

flutter packages get

- import package in code

import 'package:shared_preferences/shared_preferences.dart';

Data Persistence in Flutter

- Get Instance

```
final prefs = await SharedPreferences.getInstance();
```

- Save data

```
prefs.setInt('counter', counter);
```

- Read data

```
final counter = prefs.getInt('counter') ?? 0;
```

- Remove data

```
prefs.remove('counter');
```

Flutter Database Using SQFLITE

- install dependencies

dependencies:
    sqflite: ^0.12.2+1

- run command

flutter packages get

- import package in code

import 'package:sqflite/sqflite.dart';

CRUD Operation with SQFLITE #1

```
final String tableTodo = "todo";
final String columnId = "_id";
final String columnTitle = "title";
final String columnDone = "done";

class Todo {
  int id;
  String title;
  bool done;

  Map<String, dynamic> toMap() {
    var map = <String, dynamic>{
      columnTitle: title,
      columnDone: done == true ? 1 : 0
    };
    if (id != null) {
      map[columnId] = id;
    }
    return map;
  }

  Todo();

  Todo.fromMap(Map<String, dynamic> map) {
    id = map[columnId];
    title = map[columnTitle];
    done = map[columnDone] == 1;
  }
}
```

CRUD Operation with SQFLITE #2

```
class TodoProvider {
  Database db;

  Future open(String path) async {
    db = await openDatabase(path, version: 1,
        onCreate: (Database db, int version) async {
      await db.execute('''
create table $tableTodo (
  $columnId integer primary key autoincrement,
  $columnTitle text not null,
  $columnDone integer not null)
''');
    });
  }

  Future<Todo> insert(Todo todo) async {
    todo.id = await db.insert(tableTodo, todo.toMap());
    return todo;
  }
```

CRUD Operation with SQFLITE #3

```
  Future<Todo> getTodo(int id) async {
    List<Map> maps = await db.query(tableTodo,
        columns: [columnId, columnDone, columnTitle],
        where: "$columnId = ?",
        whereArgs: [id]);
    if (maps.length > 0) {
      return new Todo.fromMap(maps.first);
    }
    return null;
  }

  Future<int> delete(int id) async {
    return await db.delete(tableTodo, where: "$columnId = ?", whereArgs: [id]);
  }

  Future<int> update(Todo todo) async {
    return await db.update(tableTodo, todo.toMap(),
        where: "$columnId = ?", whereArgs: [todo.id]);
  }

  Future close() async => db.close();
}
```

Any questions?