
Flutter

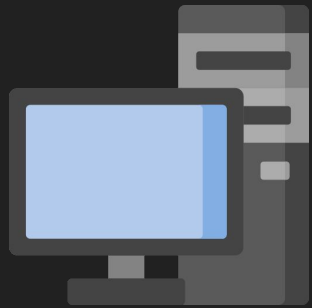
ปูพื้นฐาน Flutter & Dart (เรียนฟรี!!!)

1.Flutter & Dart เบื้องต้น (100+ ตอน)

เข้าเรียนได้ที่ =>https://bit.ly/355m3xF

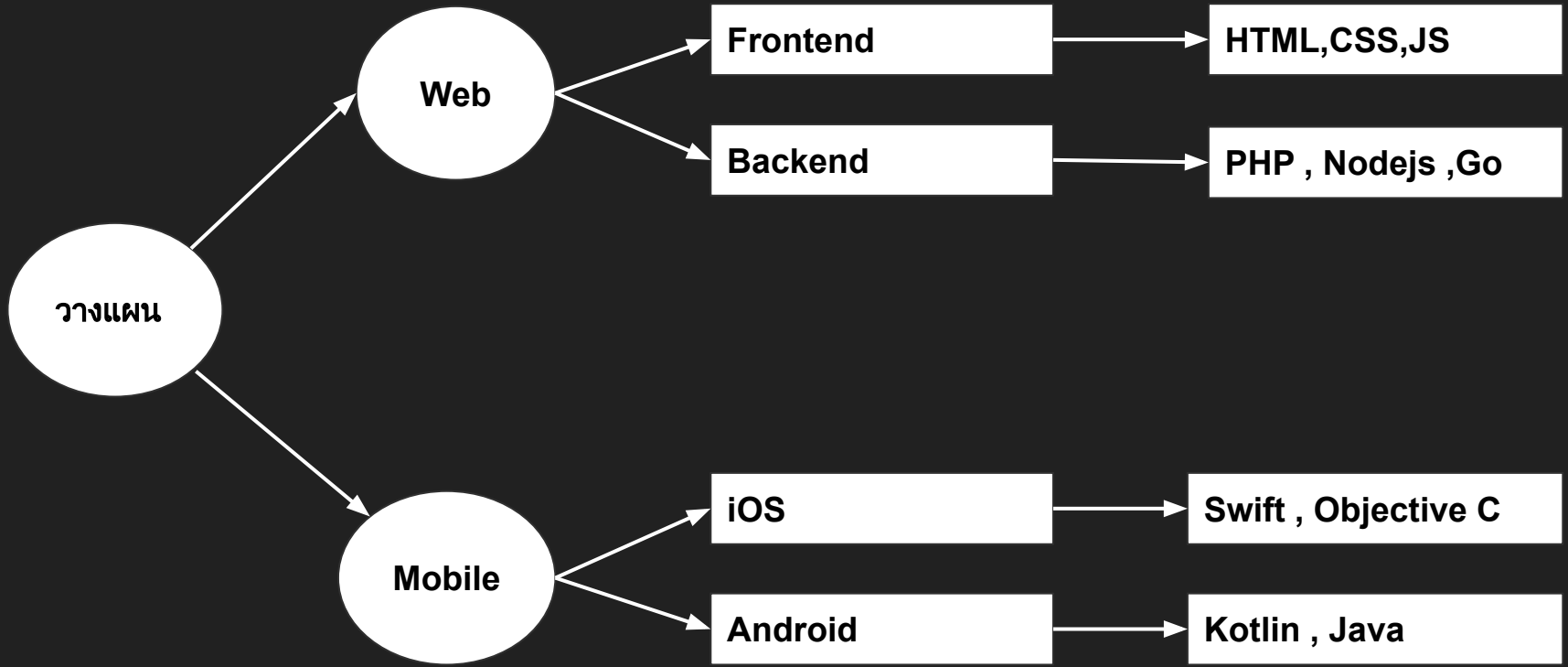
2.Flutter & ฐานข้อมูล

เข้าเรียนได้ที่ => รออัพเดตที่ช่องยูทูป



https://www.youtube.com/channel/UCQ1r_4x-P-fETLIU4pqf98w https://www.facebook.com/KongRuksiamTutorial/

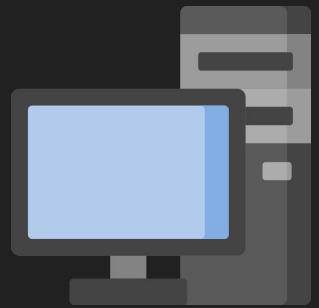
การพัฒนาระบบต่างๆในอดีต



การพัฒนาแอพแบบ Cross Platform

การที่แอพพลิเคชั่นหรือซอร์ฟแวร์สามารถทำงานได้หลายแพลตฟอร์มและรองรับการทำงานได้หลายระบบโดยเขียนโปรแกรมแค่ครั้งเดียว ซึ่งมีอยู่หลายรูปแบบ เช่น

- ทำงานบน PC ได้หลายระบบปฏิบัติการ (Windows , Mac , Linux)
- ทำงานบนเว็บได้ผ่าน Web Browser (Chrome , Firefox , Opera , Safari)
- ทำงานในสมาร์ทโฟนได้หลายระบบปฏิบัติการ (iOS , Android)



ทำความรู้จักกับ Flutter

Flutter คือเครื่องมือที่ถูกพัฒนาขึ้นโดย Google สำหรับนำมาพัฒนา mobile application ที่ทำงานในอุปกรณ์พกพา เช่น smart phone , tablet

โดย Flutter เป็นชุดโปรแกรมที่เน้นในส่วนของหน้าตาของแอพ หรือ User Interface (UI) ให้นักพัฒนาสามารถสร้างและปรับปรุงแก้ไขหน้าตาของแอพได้สะดวกตามที่ต้องการ

ทำความรู้จักกับ Flutter

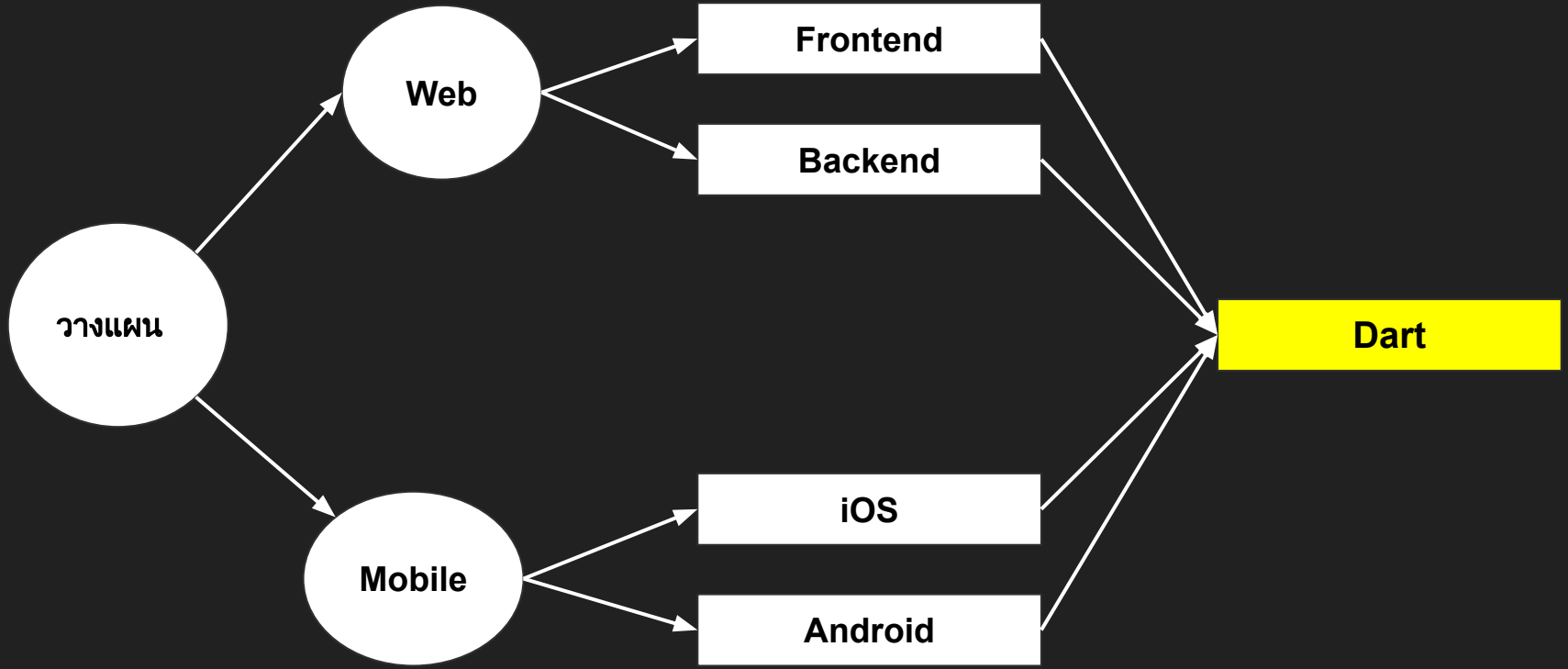
Flutter มีภาษาโปรแกรมก็คือ**ภาษา Dart** ซึ่งภาษา Dart จะถูกใช้ในการสร้าง User Interface หรือการเขียนโปรแกรมส่วนอื่นๆ เช่น การติดต่อการฐานข้อมูล การเข้าถึงและจัดการเกี่ยวกับอุปกรณ์พกพา เช่น กล้องถ่ายรูป และ ตัวสแกนบาร์โค้ด เป็นต้น

ภาษา Dart สามารถนำไปใช้ในการพัฒนา Web Application ได้ด้วยโดยการแปลงภาษา Dart เป็น JavaScript และรันใน Web Browser สามารถเข้าดูรายละเอียดเพิ่มเติมได้ที่

https://flutter.dev/docs/get-started/web

ทำความรู้จักกับ Flutter

Flutter นั้นจะทำให้เราเขียนและพัฒนาแอพได้ง่ายมากยิ่งขึ้น โดยการเขียนแอพแค่ครั้งเดียวแต่สามารถ**ใช้งานแอพนั้นได้ในระบบหลักๆถึง 2 ระบบ คือ iOS และ Android** หรือเรียกว่า **Cross Platform**


วางแผน
Web
Mobile
Frontend
Backend
iOS
Android
Dart

สรุป Flutter

1. ชุดโปรแกรมด้าน User Interface ถูกสร้างโดย Google
2. จัดการส่วนต่างๆได้ง่ายและสะดวก
3. เป็น Open Source ใช้งานได้ฟรี
4. พัฒนา Mobile Application ให้สามารถใช้งานได้ทั้ง iOS และ Android

โครงสร้างโปรเจค

- **lib** เก็บไฟล์ที่แยกหน้าแอพต่างๆมีนามสกุล .dart เช่น main.dart ถูกประกาศให้เป็นจุดเริ่มต้นทำงานของแอพเราโดยรันผ่านเมธอด main หรือทำเป็นรูปแบบ mvc ก็ได้
- **pubspec.yaml** สำหรับการตั้งค่าโปรเจคหรือใช้งานไลบราลี่ต่างๆ
- **android , ios** จัดเก็บโปรเจคของ application แต่ละระบบเอาไว้

กำหนดชื่อแอพ Android ใหม่

1. android -> app ->src -> main->AndroidManifest.xml
2. กำหนดชื่อแอพใหม่ใน android:label
3. Build
4. เสร็จสมบูรณ์

ความละเอียดหน้าจอ

1. hdpi (high-density) - หน้าจอความละเอียดสูง
2. mdpi (medium-density) หน้าจอความละเอียดปานกลาง
3. xhdpi
4. xxhdpi
5. xxxhdpi

กำหนดไอคอนให้แอพ

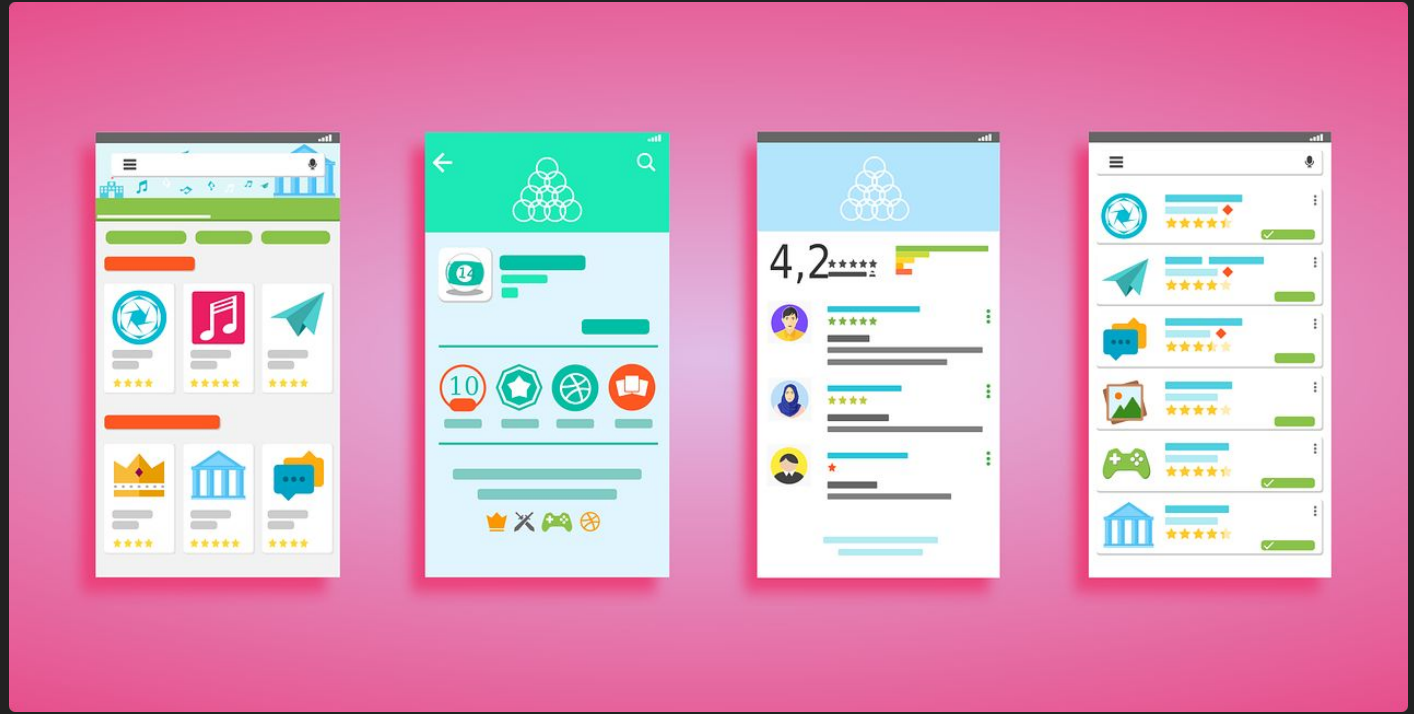
1. pub.dev
2. ติดตั้ง package flutter_launcher_icons
3. android -> app->src->main->res (resource) -> รูปแบบความละเอียดหน้าจอ
4. โฟลเดอร์ที่แยกเก็บภาพไอคอนสำหรับใช้แสดงผลในหน้าจอที่มีความละเอียดแตกต่างกัน

Widget คืออะไร

แนวคิดในการแบ่ง User Interface ออกเป็นชิ้นส่วนต่างๆ แล้วทำมาแสดงผลบนหน้าจอ เราจะเรียกชิ้นส่วนนี้ว่า Widget เช่น ปุ่ม ข้อความ เป็นต้น

Widget แต่ละตัวจะมีส่วนที่เรียกว่า **Properties** สำหรับกำหนดคุณสมบัติให้กับ Widget นั้นๆ ซึ่ง Widget แต่ละตัวก็มี Properties ที่หลากหลายให้เราใช้งานแตกต่างกันออกไป เช่น กำหนดสี เส้นขอบ ส่วนโค้ง เป็นต้น

Widget คืออะไร



Widget พื้นฐานที่อยู่ใน Dart

- Text Widget สำหรับแสดงข้อความ
- RaisedButton สำหรับจัดการปุ่ม
- Row , Column สำหรับสร้าง Layout (เค้าโครงหน้าแอพ) แบบแนวตั้งและแนวนอน
- Stack สร้าง Layout แบบซ้อนทับหรือเรียงลำดับ
- Container กล่องที่รวบรวม Widget ต่างๆ

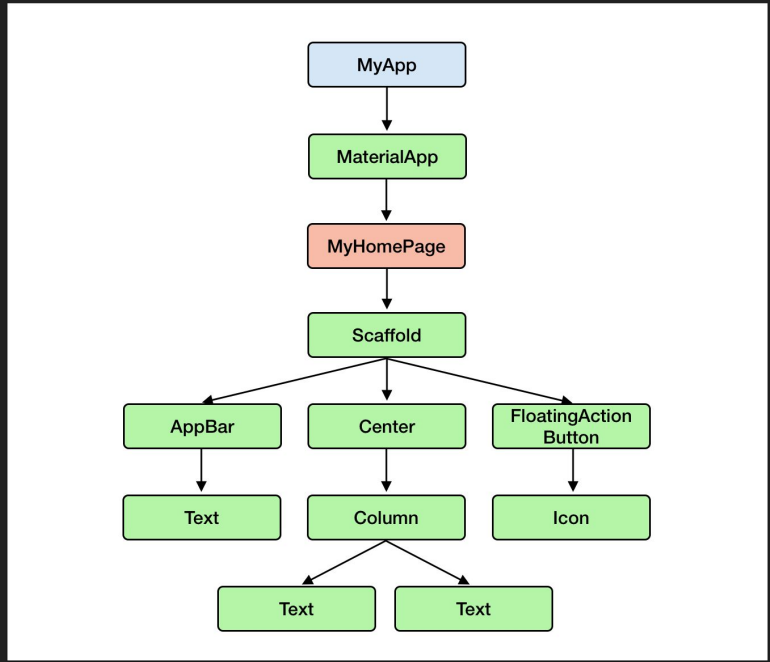
ใน Flutter จะมี Widget ที่ชื่อว่า Material Design

(มาตรฐานหรือโครงสร้างในการออกแบบบนสมาร์ทโฟนและบนเว็บ)

มากำหนดเป็น Layout ง่ายๆให้เรากำหนดชื่อ Title บน Appbar (เมนูด้านบนแอพ)

Widget Tree

คือการนำ Widget มาซ้อนทับกันแบบแผนผังต้นไม้ Widget ที่บรรจุ Widget อื่นให้อยู่ภายในเรียกว่า Parent Widget ส่วน Widget ที่ด้านในจะเรียกว่า Children Widget



เริ่มต้นใช้งาน Widget

```
// ต้องการแสดงหน้าจอสวยๆแทน console

import ‘package:flutter/material.dart’;
```

```
void main(){

    runApp (app); // app คือการระบุ widget ต้องการทำงาน

    runApp(Text(“Hello Dart”));

}
```

เริ่มต้นใช้งาน MaterialApp Widget

MaterialApp จะใช้การควบคุมการแสดงผลในหน้าแอพของเรา

```
void main(){

var app = MaterialApp(title:”Hello”,home:Text(‘My App’)); // สร้าง widget ขึ้นมา

runApp(app);

}
```

จัด Layout หน้าแอพด้วย Scaffold Widget (Template)

Scaffold คือ Widget หน้าต่างสำเร็จรูปสำหรับจัดการ Layout หรือโครงสร้างของหน้าแอพมีการคำนวณระยะห่างของแอพกับหน้าจอ Emulator ให้อัตโนมัติ

```
runApp(MaterialApp(
home:Scaffold( // home จะให้แสดงผลอะไรในหน้านี้
        title:”Hello”,
        appBar:AppBar(title:Text:(“KongRuksiam”))//เมนูด้านบนแอพ
        body:Text(‘Hello Flutter’) // ข้อความที่แสดงในพื้นที่
)
theme:ThemeData(primarySwatch:Colors.green);//เปลี่ยนสี
));
```

การสร้าง Widget

ใน Flutter ถ้าต้องการสร้าง Widget ขึ้นมาใช้งานเอง สามารถสร้างได้ 2 แบบ คือ

- **Stateless Widget** เป็น Widget ที่ไม่สามารถเปลี่ยนแปลงค่าได้ใช้สำหรับสร้าง Widget แบบคงที่ เช่น ข้อความ ไอคอน เป็นต้น
- **Stateful Widget** เป็น Widget ที่สามารถเปลี่ยนแปลงค่าได้หรือทำงานได้หลาย State เช่น Checkbox , Slider , Textfield เป็นต้น

(สามารถเปลี่ยน Widget จาก ful เป็น less หรือจาก less เป็น ful ก็ได้)

Center Widget

เป็น Widget ที่ครอบ Widget อื่นๆให้อยู่ตรงกลางหน้าจอ

```
Scaffold(
    body:Center(
        child:Text('KongRuksiam');
    )
)
```

กำหนดขนาดและสีข้อความ

```
Text('ข้อความ',
      style:TextStyle(
      fontSize:30,
      color:Color.blue
))
```

การนำภาพเข้ามาทำงาน

```
Scaffold (
    body:Center(
    child:Image(image:NetWorkImage('http://www.....png'))

    )
)
```

Column Widget

เป็น Widget ที่รับเอา Widget อื่นๆมาจัดเรียงในแนวตั้งหรือในแนวดิ่ง

```
body:Center(
    child:Column(
        children:<Widget>[
            Text('เพิ่มข้อมูล')
        ]
    )
)
```

Column Widget

จัดตำแหน่งของ Widget ในคอลัมน์

mainAxisAlignment : MainAxisAlignment :start

mainAxisAlignment : MainAxisAlignment :center

mainAxisAlignment : MainAxisAlignment :end

การสร้าง Stateful Widget

หมายถึง Widget ที่สามารถเปลี่ยนแปลงค่าได้หรือทำงานหลาย State โดย State คือตัวแปรหรือข้อมูลที่ใช้ควบคุมการทำงานของแอพให้โต้ตอบกับผู้ใช้ได้ มีการใช้งานคำสั่ง setState() เพื่อบอกว่าในแอพมีบางอย่างเปลี่ยนแปลงเกิดขึ้นที่ State สามารถทำให้ข้อมูลในหน้าแอพเปลี่ยนแปลงและทำงานต่อเนื่องได้ โดยที่เราไม่ต้อง build และรีแอพใหม่

สรุป

State ควบคุมการทำงานของแอพตามที่ต้องการ เช่น อัปเดตการทำงานของหน้าจอ หรือ Widget ผ่าน State นั่นเอง

Widget หน้าตาของแอพ

ปุ่มและไอคอน

```
floatingActionButton:FloatingActionButton(

      onPressed:(){

      },
      child:Icon(Icons.add)  หรือ
      child:Text(‘ข้อความ’)
)
```

Listview Widget

คือ Widget ที่มีการแสดงข้อมูลแบบรายการ เมื่อมีรายการมากๆเกินพื้นที่ของหน้าจอสามารถเลื่อนหน้าจอหรือ Scroll Down เพื่อดูรายการเพิ่มเติมและ Scroll Up เพื่อดูรายการก่อนหน้าได้

- **List คือ โครงสร้างข้อมูลที่จัดการเกี่ยวกับกลุ่มข้อมูล**
- **Listview** จะใช้ในกรณีที่แสดงรายการขนาดเล็ก (4-10 รายการ)
- **Listview.builder** จะใช้ในกรณีที่มีการแสดงรายการจำนวนมาก โดยระบุจำนวนรายการผ่าน Properties ชื่อว่า itemcount
- **ListTile** กำหนดรายละเอียดต่างๆของ List แต่ละรายการ เช่น หัวข้อ (title) และ หัวข้อย่อย (subtitle) เป็นต้น

Container Widget (กล่องใส่ของ)

คือ Widget ที่ใช้กำหนดพื้นที่ สามารถกำหนด ขนาด รูปร่าง หรือจัดวางตำแหน่ง รวมไปถึงการกำหนดลวดลายและตกแต่งด้วยสีให้สวยงามได้ ปกติตัว Container จะไม่แสดงผลขึ้นมาให้เห็น เราจะต้องกำหนดความสูงลงไปและขนาดของ Container ก็จะขยายเต็มพื้นที่ ใช้ในการจัดวาง UI ให้แสดงผลเหมาะสมกับหน้าจอ

BoxDecoration

สำหรับกำหนดรูปร่างของ Container เช่น สี ลักษณะรูปร่างของ Container เป็นต้น

Padding

คือ การกำหนดระยะห่างของ Widget ออกจากขอบของ Layout

Columns Widget

คือ Widget ที่ใช้แสดง Widget ย่อย (Child Widget) **ในแนวตั้ง**

Row Widget

คือ Widget ที่ใช้แสดง Widget ย่อย (Child Widget) **ในแนวนอน** มีการกำหนดคุณสมบัติคล้ายๆกับ Columns Widget แต่ต่างกันที่รูปแบบจัดเรียง

Expanded Widget

Widget ที่ใช้ขยายความสูงของ Widget ย่อย (Child Widget) ที่ต้องการ โดยทำการจัดสรรพื้นที่ของ Widget ย่อยให้เต็มพื้นที่โดยดูจากพื้นที่ว่างที่เหลือในหน้าจอ (คล้ายๆกับ Flexbox ใน css) และมีการกำหนดพื้นที่ผ่าน Flex Properties

วงจรการทำงานของ Widget (Lifecycle)

Stateless Widget	Stateful Widget
State ไม่สามารถเปลี่ยนแปลงค่าได้ตลอดเวลา	State สามารถเปลี่ยนแปลงค่าได้ตลอดเวลา
ฟังก์ชั่นที่สร้างขึ้นจะทำงานครั้งเดียว	ใช้ setState ในการสั่งให้ฟังก์ชั่นที่สร้างขึ้นทำงาน

Stateful Widget (Lifecycle)

initState() คือ เมธอดที่ถูกเรียกให้ทำงานครั้งเดียวหลังจากที่ State ถูกสร้างเรียบร้อยแล้ว (หรือ Widget ถูกสร้างขึ้นมาแล้ว) สามารถที่จะเปลี่ยนแปลงข้อมูลของ Widget ได้

build() คือ เมธอดที่ใช้ในการสร้าง Widget หรือ Widget Tree ของแอพพลิเคชั่นถ้าต้องการให้ Widget ของ แอพมีหน้าตาแบบใดก็ให้เขียนในเมธอด build ซึ่งเมธอด build จะถูกเรียกใช้งานเมื่อมีการเปลี่ยนแปลงข้อมูลใน State โดยทำงานผ่านเมธอด setState() นั่นเอง

Stateful Widget (Lifecycle)

dispose() คือ เมธอดที่จะถูกเรียกใช้เมื่อ Widget หรือ State หายไปจากหน้าแอพหรือให้หยุดทำงานและนำออกจากระบบ

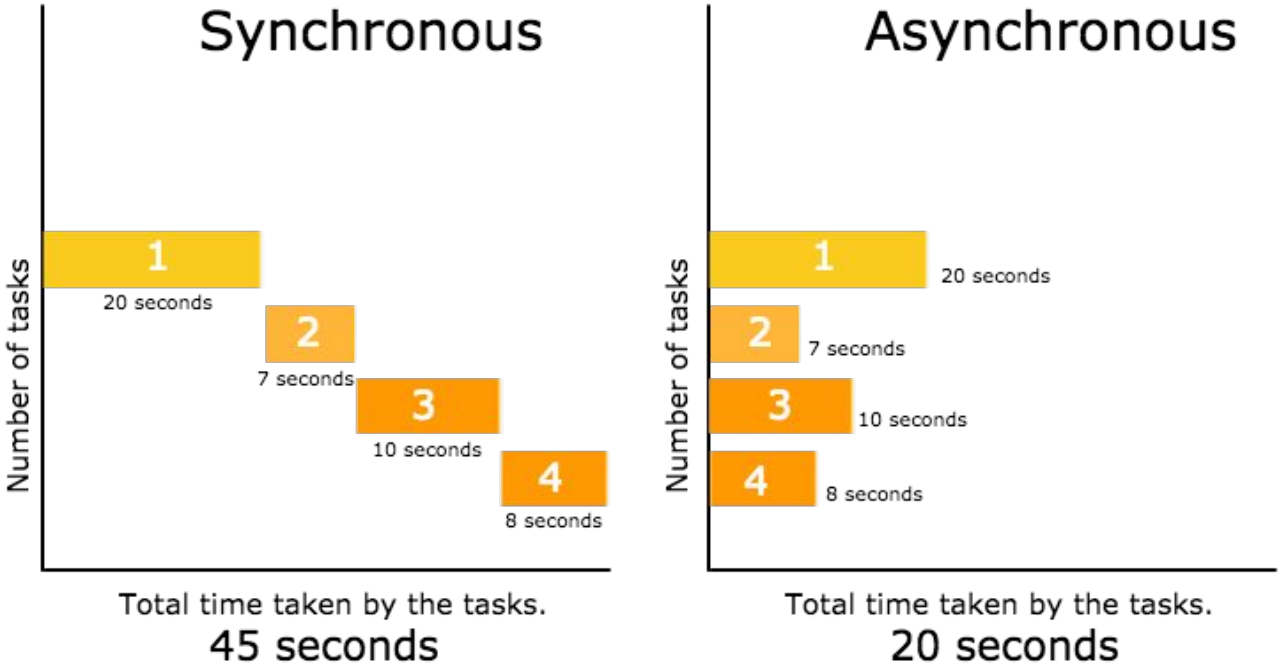
การทำงานแบบ Asynchronous

ใน Flutter จะมีการทำงานไปเรื่อยๆโดยที่ไม่ต้องรอให้ฟังก์ชั่นก่อนหน้าทำเสร็จก่อน หมายถึง สามารถทำงานฟังก์ชั่นถัดไปและกลับมาทำฟังก์ชั่นก่อนหน้าอีกครั้งได้นั่นเอง

Asynchronous

Synchronous คือ การทำงานแบบตามลำดับ สมมุติมีงาน A และงาน B โดยจะให้เริ่มทำงานใน A ให้เสร็จก่อนจนกระทั่งงาน A นั่นเสร็จสิ้นถึงจะทำงาน B หรืองานอื่นต่อได้ สรุปคือ ต้องรอทำงานคำสั่งก่อนหน้าให้เสร็จแล้วค่อยทำคำสั่งถัดๆไป

Asynchronous หรือ Non-blocking การทำงานที่สามารถจะสลับไปทำงานอื่นได้ เช่น สามารถทำงาน A และ B ไปพร้อมๆกันโดยที่ไม่ต้องรอให้อีกงานเสร็จ



http://www.phpmind.com/blog/2017/05/synchronous-and-asynchronous/

ตัวอย่าง Asynchronous ในชีวิตประจำวัน

- **กินข้าว** – เราสามารถที่จะกินข้าวไปพร้อมๆกับดูคลิปในยูทูปได้ในเวลาเดียวกัน

- **สั่งอาหาร** – เราสามารถสั่งให้คนมาส่งอาหารให้เราได้พร้อมๆกับนั่งเขียนโค้ดไปด้วย ในเวลาเดียวกัน เรารอให้คนส่งของมาถึงงานเราก็ใกล้จะเสร็จพอดีในเวลาเดียวกัน

- **โปรแกรมแชท** – เราสามารถที่จะอัพโหลดและส่งภาพในช่องแชทได้ ระหว่างที่รออัพโหลดและส่งภาพไปเราก็ยังสามารถพิมพ์แชทได้ในเวลาเดียวกัน โดยที่ไม่ต้องรอให้อัพโหลดภาพและส่งภาพเสร็จค่อยแชทได้

รู้จักกับ Future

หมายถึง Object หรือข้อมูลที่ได้จากการทำงานในรูปแบบ Asynchonous หรือ ความหมายอีกนัยหนึ่งของ Future คือ การนำค่าที่จะเกิดขึ้นในอนาคตมาใช้งาน ซึ่งจะทำการเช็คข้อมูลที่เกิดขึ้นผ่าน State เช่น สมบูรณ์ (complete) , ไม่สมบูรณ์ (incomplete) เป็นต้น

การนิยาม Future

```
Future <return type> ชื่อFuture async{
    await.....);
}
```

เมื่อมีการทำงานกับฟังก์ชั่นแบบ Asynchonous จะมีการใช้งานร่วมกับคำสั่ง async และ await สำหรับรอให้ทำงานจนเสร็จ

ตัวอย่าง Future (การรอทำงานบางอย่าง)

โจทย์ จำลองดึงข้อมูลผู้ใช้จากฐานข้อมูลมาแสดงผล

- เริ่มต้นเรียกข้อมูล
- ดึงข้อมูลผู้ใช้ (ใช้เวลา 10 วินาที)
- แสดงผล
- ทำงานอย่างอื่นต่อไป

รู้จักกับ API

API คือ วิธีการเรียกใช้งานโปรแกรม เป็นรูปแบบโปรแกรมกับโปรแกรม กล่าวคือสามารถทำให้โปรแกรมกับโปรแกรมคุยกันได้ เพื่อใช้สำหรับแลกเปลี่ยนข้อมูลกัน

โปรแกรมต้นทางจะเรียกว่า Server คอยเปิดและให้บริการข้อมูล

โปรแกรมปลายทาง จะเรียกว่า Client สำหรับเรียกใช้บริการจาก Server

ต้องเชื่อมต่อ Internet จึงจะรับส่งข้อมูลกันได้

การทำงานของ API

- ผู้ให้บริการจะกำหนดกฎในการเรียกใช้งานข้อมูล ก่อนใช้งาน API เราต้องรู้ว่าจะใช้คำสั่งอะไร (1 คำสั่ง = 1 API) เช่น ส่งพารามิเตอร์อะไรเข้าไป

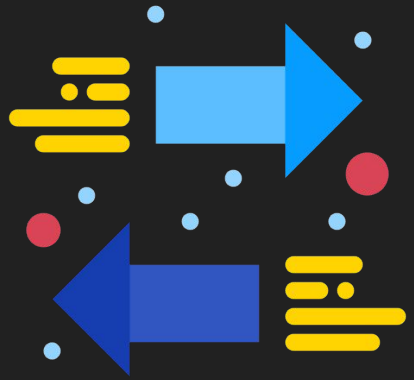
- ไม่ใช่ทุกที่จะให้บริการข้อมูลของตน อาจจะต้องมีการสมัครเพื่อรับบริการดังกล่าว ถ้าเปิดให้คนนอกสามารถเข้าถึงข้อมูลได้จะเรียกว่า OpenAPI

คำศัพท์ที่ควรรู้

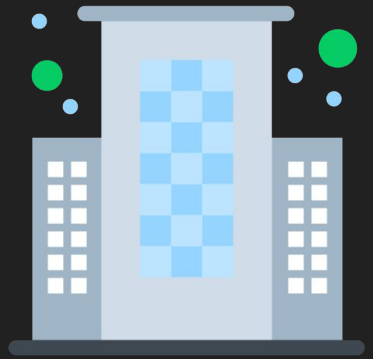
- **Request** การส่งคำขอเพื่อใช้บริการ API
- **Response** การตอบกลับส่งข้อมูลกลับมาตามคำขอที่ส่งไป (ได้หรือไม่ได้)
- **Server** ผู้ให้บริการข้อมูล (API)
- **Client** ผู้ใช้บริการ

แผนภาพ

Request (ส่งคำขอ)



Response (ตอบกลับ)



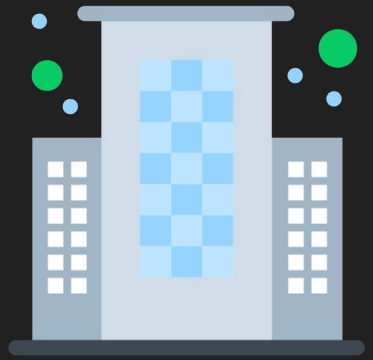
ผู้ใช้บริการ (Client)

ผู้ให้บริการ (Server)

ตัวอย่างการเรียกใช้งาน API

ชื่อจังหวัด : ชลบุรี

ข้อมูลสภาพอากาศ

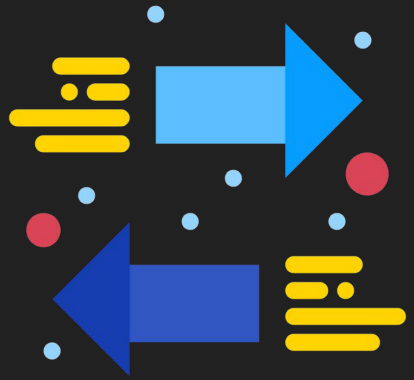


ผู้ใช้บริการ (Client)

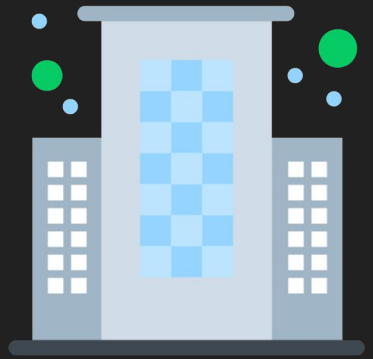
กรมอุตุนิยมวิทยา(Server)

ตัวอย่างการเรียกใช้งาน API

ผู้ป่วยโควิดวันนี้



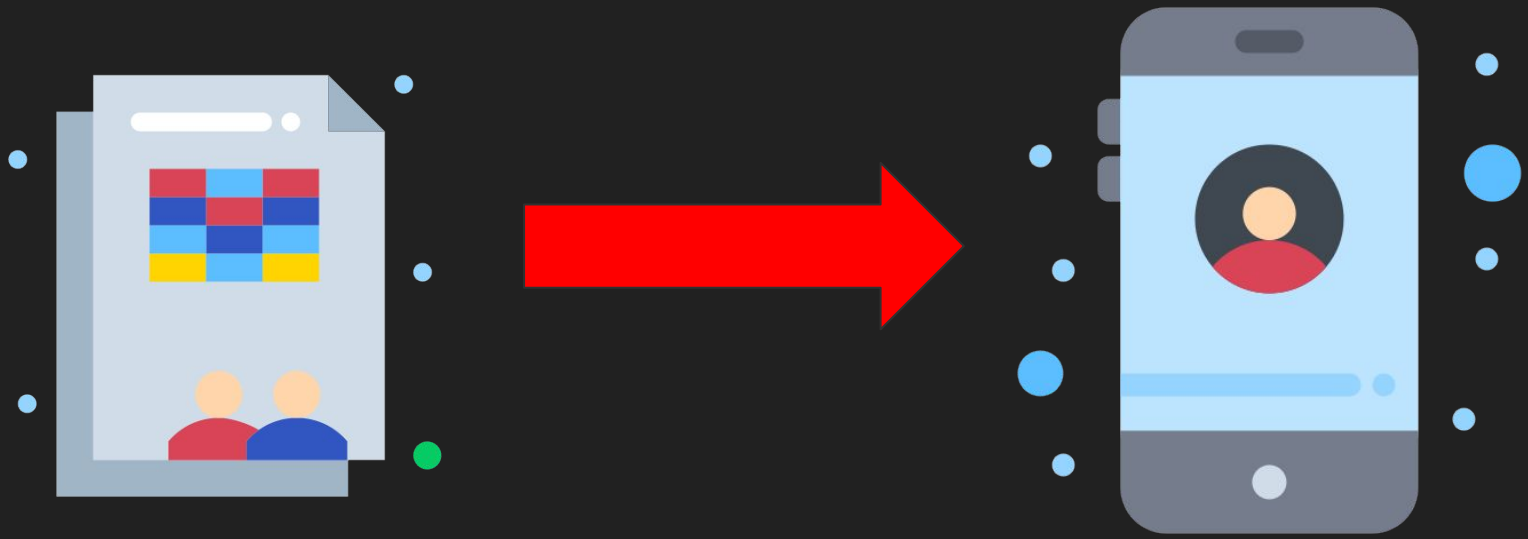
ข้อมูลผู้ป่วยโควิด



ผู้ใช้บริการ (Client)

กรมควบคุมโรค(Server)

นำข้อมูลที่ส่งกลับมาทำงานในแอพของเรา

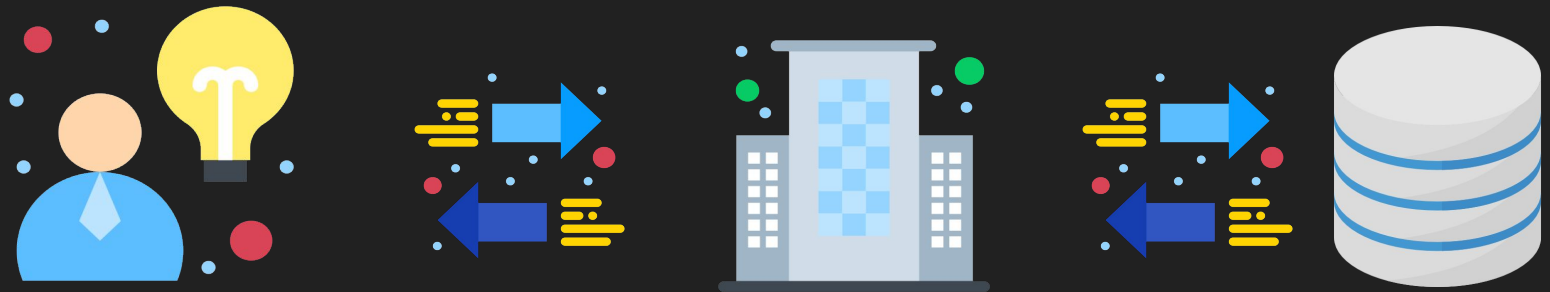


ข้อมูล

รายงานผลในแอพ Android / iOS

Web API

คือ การให้บริการข้อมูลผ่าน Web โดยรับส่งข้อมูลผ่าน HTTP ซึ่งจะอยู่ในรูปแบบของ XML และ JSON โดยข้อมูล (ภาพ,เสียง,ข้อมูลทางธุรกิจ) จะเรียกว่า Resource



HTTP Method

คือ ตัวที่บ่งบอกการกระทำกับข้อมูล (Resource)

- GET Method สำหรับร้องขอข้อมูล
- POST Method สำหรับสร้างข้อมูลใหม่
- PUT Method สำหรับอัพเดตข้อมูล
- DELETE Method สำหรับลบข้อมูล

จัดการค่า NULL

?? Null Coalescing - ใช้เช็คตัวแปรถ้ามีค่า Null จะให้ใช้ค่าเริ่มต้น (dafault) แทน ยกตัวอย่าง เช่น

```
String name;

print(name??”kongruksiam”);
```

จัดการค่า NULL

??= Null Coalescing Assignment ใช้เช็คตัวแปรถ้ามีค่า Null จริงๆหรือไม่ สามารถกำหนดค่าลงไปได้เลย

```
String name;

name??=”kongruksiam”;

print(name)
```

จัดการค่า NULL

?. Null Conditional ใช้เช็ค object ว่าเป็นค่า Null หรือไม่ สามารถเช็คค่าได้ถ้าเป็นค่า Null จะให้ข้ามไปไม่มีอะไรเกิดขึ้น

```
class Employee{
    void showData(){
        print(“Hello Flutter”)
    }
}
```

```
Employee emp;
if(emp!=null){
    emp.showData();
}
```

แบบสั้นๆ

```
emp?.showData();
```

แอพอัตราการแลกเปลี่ยนเงิน

ดึงข้อมูลมาแสดงผล

Source Code

https://github.com/kongruksiamza/Flutter-Basic