ICT Department

ប្រើ Future, async, await សំរាប់ធ្វើការជាមួយ Asynchronous

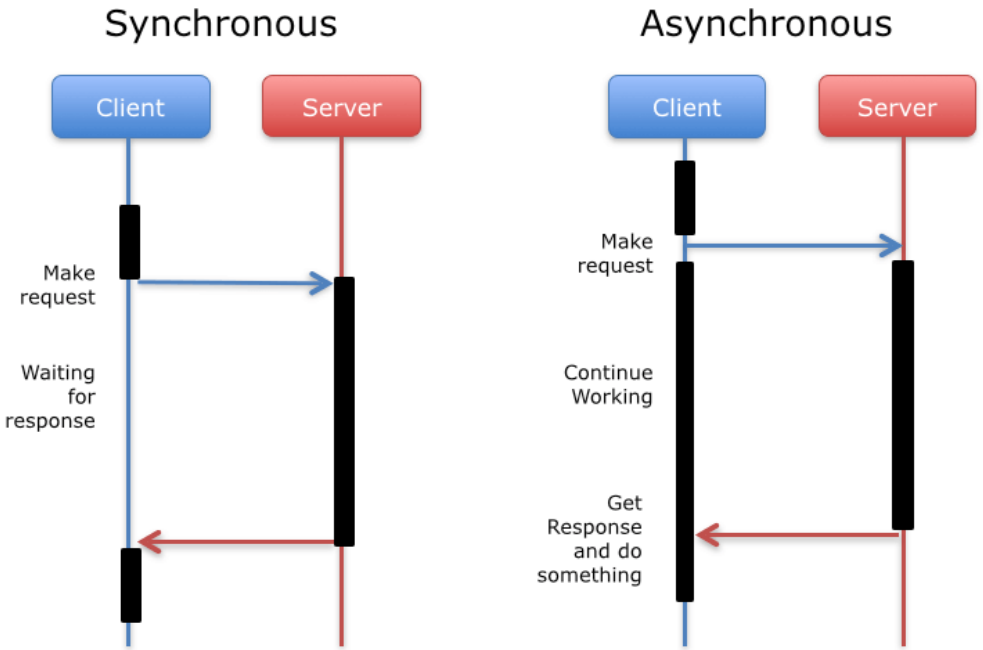
Instructor: **Oum Saokosal**, *Master of Engineering in Information Systems, South Korea '2010*

Email: oumsaokosal@gmail.com

Phone: 012 252 752 (Telegram)

Asynchronous

- ប្រតិបត្តិការដែលដើរបែប Asynchronous គឺអនុញ្ញាតិអោយកម្មវិធីរបស់យើងបញ្ចប់ដំណើរការងារទី១ ក្នុងខណៈពេលកំពុងរង់ចាំការងារទី២កំពុងបញ្ចប់ដែរ។
- ចំនុចចាំបាច់ដែលត្រូវប្រើ Asynchronous គឺនៅពេលដែលការងារនោះត្រូវប្រើរយៈយូរ។ ឧទាះ
  - ទាញយកទិន្នន័យពី Server
  - ផ្ទុកទិន្នន័យចូលក្នុង database
  - ទាញយកទិន្នន័យពី file ជាដើម



Asynchronous នៅក្នុង Flutter

- នៅក្នុង Flutter ដើម្បីប្រើ Asynchronous គឺយើងអាចប្រើ៖
  - Future
  - async
  - await
- ហើយសំរាប់ Widget វិញ គឺ Flutter មាន៖
  - FutureBuilder
  - StreamBuilder

    (សំរាប់មេរៀន StreamBuilder យើងនឹងនិយាយគ្នាក្នុងមេរៀន Firebase)

Future

Future គឺអាចប្រើជា datatype របស់ function/method ដែលវាបញ្ជាក់ថា ទិន្នន័យណាមួយវានឹងកើតឡើងនាពេលបន្តិចទៀតនោះ។

- បង្កើត Future function:

```
Future sayHello(String name){
  print("Hello $name");
}
```

- ការហៅ Future function:

```
sayHello("Sok");
```

```
Future<String> getOnlyDate(DateTime now){
  int year = now.year;
  int month = now.month;
  int day = now.day;
  return Future.value("$year/$month/$day");
}
```

```
getOnlyDate(DateTime.now()).then((date){
  String someDate = date;
  print(someDate);
});
```

delayed របស់ Future

- Future ក៏អនុញ្ញាតិអោយយើងបង្កើត object ជាលក្ខណៈ Future ផងដែរ។

```
Future<String> future = Future(() => "Latest News");
future.then((news) {
  print(news);
});
```

- delayed គឺជា option មួយដែលគេឧស្សាហ៍ប្រើជាមួយនឹង Future សំរាប់ពន្យាពេល៖

```
Future<String> future = Future.delayed(
  Duration(seconds: 2),
      () => "Latest News",
);

future.then((news) {
  print(news);
});
```

await និង async

- ក្នុងឧទាហរណ៍មុននេះ យើងអាចសង្កេតឃើញការហៅ Future function, គឺយើងត្រូវទាមទារប្រើ .then។ ព្រោះថា Future function ត្រូវការពេលវេលាមួយបញ្ចប់សិន ទើបវាបញ្ជូនទិន្នន័យអោយយើងតាមរយៈ method then។
- តែយើងអាចប្រើ await សំរាប់ប្រកាសយកទិន្នន័យផ្ទាល់ចេញពី Future function តែម្តង៖

```
String someDate = await getOnlyDate(DateTime.now());
```

ប៉ុន្តែពេលយើងចង់ប្រើ await គឺដាច់ខាត function វានឹងទាមទារ async៖

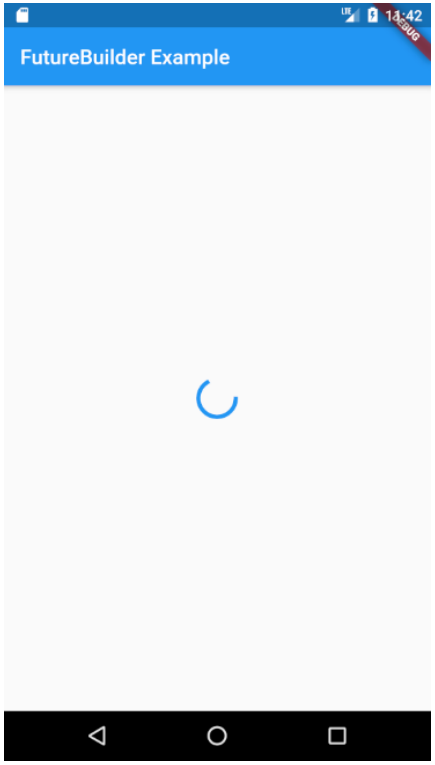
```
void someFunctions() async{
  String someDate = await getOnlyDate(DateTime.now());
  print(someDate);
}
```

FutureBuilder widget

- យើងបានដឹងហើយថា គោលដៅនៃ Asynchronous គឺជាការអនុញ្ញាតិអោយការងារមួយធ្វើការ ក្នុងខណៈពេលដែលការងារមួយទៀតកំពុងដំណើរការ ក្នុងពេលដំណាលគ្នាដែរ។ Flutter បានបង្កើតជា Widget មួយសំរាប់ប្រើជាមួយនឹងការងារ Future នេះតែម្តង គឺឈ្មោះថា FutureBuilder។ FutureBuilder គឺជា Widget ធម្មតាដូចជា Widget ដទៃផ្សេងទៀតដែរ។

```
Future<String> someFutureFunction() {
  Future<String> data = Future.delayed(
    Duration(seconds: 2),() => "Latest News",
  );
  return data;
}
```

```
return FutureBuilder<String>(
  future: someFutureFunction(),
  builder: (context, snapshot){
    if(snapshot.connectionState == ConnectionState.done){
      return Container(
        child: Text(snapshot.data),
      );
    }
    else{
      return Center(child: CircularProgressIndicator(),);
    }
  },
);
```



FutureBuilder widget

- នៅក្នុង FutureBuilder, វាត្រូវការ option សំខាន់២៖
  - future: ជាកន្លែងដែលយើងត្រូវដាក់ Future method
  - builder: ជាកន្លែងសំរាប់អោយយើងបង្ហាញលទ្ធផលដែលយើងទទួលបានពី Future method ខាងលើ។
- ចំនុចពិសេសនៅក្នុង builder គឺវាមាន connectionState ដែលបានពី snapshot៖
  - connectionState មាន none, active, waiting និង done។ connectionState done មានន័យថា ការងារបានបញ្ចប់ ដែលជាទូទៅយើងប្រើ done មួយនេះក៏បាន ហើយក្រៅពីនោះយើងអាចដាក់ក្នុង else
  - ពេលដែល connectionState ខុសពី done, គឺយើងអាចបង្ហាញ loading icon (Progress Indicator) សំរាប់បង្ហាញថា វាកំពុងតែធ្វើការអ្វីមួយ។
- snapshot.data គឺជាលទ្ធផលដែលយើងទទួលបានពី Future method ខាងលើ។