

Flutter 2

Dart Language

Sound Null Safety



Oum Saokosal

Master of Engineering in
Information Systems, South Korea
012 252 752 (Telegram)



Dart Language with Sound Null Safety

ភាសា Dart ជាមួយ Null Safety

- ភាសាសរសរកម្មវិធី Dart ត្រូវបានបង្កើតឡើងដោយ Google នាទីថ្ងៃទី ២០១១ ដែលជាការការពារកម្មវិធីទូទៅ ដែលដំបូងឡើងគោលដោបង្កើតសំរាប់សរស់របស់ Web, Server, Desktop ដែលក្រាយមកគេបានយកប្រើជាមួយនឹង Flutter SDK។
- នៅដើមឆ្នាំ ២០២១នេះដែរស្ថិតិថាលើ Flutter 2, ភាសា Dart កើតាន upgrade ដួងដែរដែលវាសង្គត់ផ្តល់លើជានប់មិនអាយមាន NULL ។ ព្រមទាំង NULL នេះគឺជានធ្វើអាយមាន Bug ជាប្រើននៅក្នុង Runtime របស់ App ។
- ទំនើបក្នុងភាគគ្រឿនរបស់ Dart គឺជូចគ្នាដាមួយនឹង Java គ្រាន់តែ Dart បានកាត់បន្ថយ ចំនួចមិនចាំបាច់មួយចំនួន ហើយបញ្ចូលនូវចំនួចលូផ្សេងៗនៃភាសា C, C++, C#, Javascript ជាដើម។
- បែបទន្លេការសរស់រក្នុង Flutter & Dart គឺមានច្បាស់ OOP, Functional Programming និង Reactive Programming។

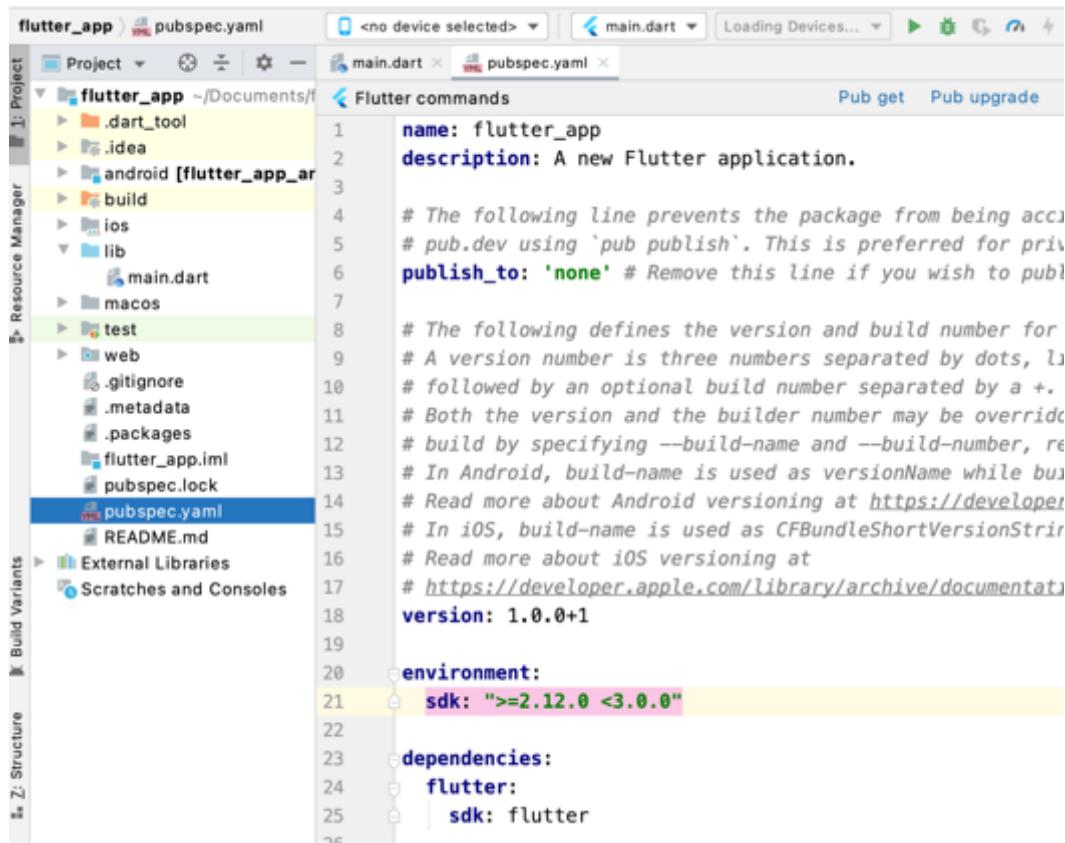


Dart

កំណត់ Version ដើម្បីប្រើ Dart Null Safety

- ចូលកើន pubspec.yaml ហើយកែរត្រង់ sdk ដែលបានជាដោះ

sdk = ">=2.12.0 <3.0.0"



The screenshot shows the Android Studio interface with the project 'flutter_app' open. The 'pubspec.yaml' file is selected in the center editor window. The 'sdk' constraint line, which is highlighted with a pink background, is visible in the code:

```
name: flutter_app
description: A new Flutter application.

# The following line prevents the package from being accidentally published to
# pub.dev using 'pub publish'. This is preferred for private packages.
publish_to: 'none' # Remove this line if you wish to publish

# The following defines the version and build number for
# A version number is three numbers separated by dots, like 1.2.3
# followed by an optional build number separated by a +.
# Both the version and the builder number may be overridden by
# build by specifying --build-name and --build-number, respectively.
# In Android, build-name is used as versionName while build-number is used as
# Read more about Android versioning at https://developer.android.com/studio/publish/versioning
# In iOS, build-name is used as CFBundleShortVersionString and build-number is used as
# Read more about iOS versioning at https://developer.apple.com/library/archive/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/TestingYouriOSApp/TestingYouriOSApp.html
version: 1.0.0+1

environment:
  sdk: ">=2.12.0 <3.0.0"

dependencies:
  flutter:
    sdk: flutter
```

ស្ថិតិមាលក្នុងការសរស់សេរក្នុងរបស់ Dart

- **UpperCamelCase:** សំរាប់ classes, enums, typedefs, និង type parameters

- class SliderMenu { ... }
- class HttpRequest { ... }
- typedef Predicate<T> = bool Function(T value);



- **lowerCamelCase:** សំរាប់ Class members, top-level definitions, variables, និង parameters.

- const pi = 3.14;
- const defaultTimeout = 1000;
- final urlScheme = RegExp('^([a-zA-Z]+):');



- **lowercase_with_underscores:** សំរាប់ libraries និង source files
 - library peg_parser.source_scanner;
 - import 'file_system.dart';
 - import 'slider_menu.dart';
- **ការ Import តាមលំដាប់:** ត្រូវ import Dart មុន lib ធ្វើនេះ
 - import 'dart:async';
 - import 'dart:html';
 - import 'package:bar/bar.dart';
 - import 'package:foo/foo.dart';

ការប្រកាសអញ្ជូនិធាមួយនឹង Null Safety

- ការប្រកាសអញ្ជូនិធាមួយនឹង Dart អាចប្រើ datatype កំណាន មិនដាក់កំណាន ប៉ុន្តែដាច់ខាត ត្រូវពេតផ្តល់តំលៃលំបើងទ្វាមញ្ជូនិធាន

 - បើមិនចង់កំណាន datatype ទេ គឺអាចប្រើ var ឬ dynamic
 - បុរាណកំណាន datatype ដោយ int, double, num, bool, String

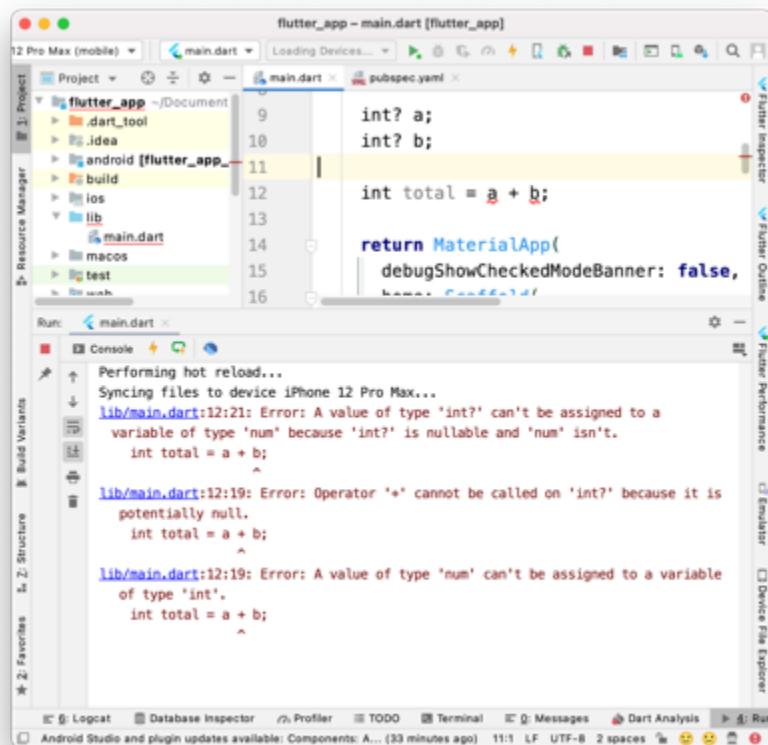
```
main() {  
    int a = 10;  
    var b = 7.5;  
    double c = 1.5;  
    bool d = false;  
    String title = "Cambodia";  
    dynamic subtitle = 'Kingdom of Wonder';  
    print("$title, $subtitle");  
}
```

Cambodia, Kingdom of Wonder

ប្រកាសអញ្ជូតិដែលអាច Null ជាមួយនឹងសញ្ញាស្តី ?

- នេះជាដំឡើងសំរាប់ Dart ដំនាន់ថ្មីនេះ គឺវាទមទារទ្វាយឱ្យយើងកំណត់តែតែលើដំបូងអាយក ព្រមទាំងបានបញ្ជាក់ថាបានផ្តល់នូវការបញ្ចូន Null
- តើទេ ជាយើងនេះកី វាអនុញ្ញាតឱ្យប្រកាសអញ្ជូតិដែលអាច Null បានដោយ ដើម្បី
ប្រកាសអញ្ជូតិដែលអាច Null តើត្រូវជាក់សញ្ញាស្តី ? ។ កីបុន្ណែនបើយើងមិនបានជាក់
តែតែទ្រូវរាយការណីងចេញ Error !

```
int? a;  
  
int? b;  
  
int total = a + b;
```



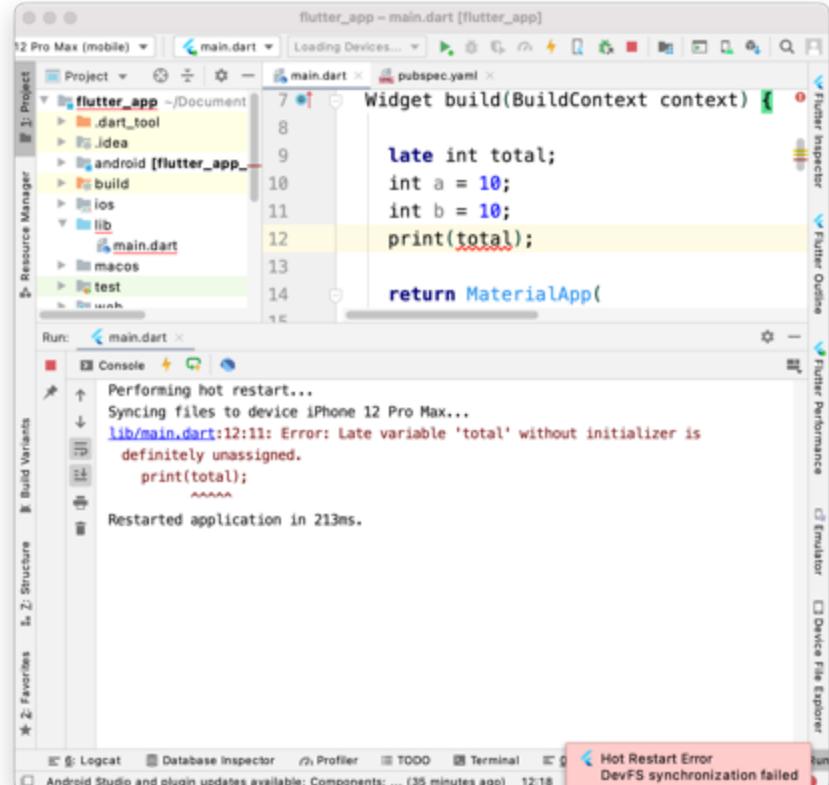
ប្រកាសអញ្ជីជាមួយ late

- ពេលខ្លះអញ្ជីមួយចំនួនមិនមានកំណែដំបូងទេ ត្រូវរានឹងទទួលកំណែពេលក្រាយ។
ក្នុងករណីបែបនេះយើងអាចប្រកាសអញ្ជីជាមួយ keyword ថ្មីណ៍ឱ្យោះ late:

```
late int total;  
int a = 10;  
int b = 20;  
total = a + b;
```

- ក្នុងករណីយើងត្រូវបានកំណែចូលតម្លៃតុល្យ total
នៅរានឹងចេញ Error បែបនេះ:

```
late int total;  
int a = 10;  
int b = 10;  
print(total);
```



ប្រមាណវិធីជាមួយ Null

- ??= គឺជាប្រមាណវិធីនៃអោយតម្លៃដំបូងទៅ variable ដែលមាន null ខាងក្រោម

```
int? a; // = null  
a ??= 3;  
print(a); // <-- Prints 3.  
a ??= 5; print(a); // <-- Still prints 3.
```

- ?? គឺជាប្រមាណវិធីដូចត្រឡប់នឹង if null ខាងក្រោម

```
int? a; // = null  
int b = a ?? 0; // b = 0
```

ក្នុងខាងលើនេះគឺដូចត្រឡប់នឹងក្នុងខាងក្រោមនេះ

```
int? a;  
int b = 0;  
if(a != null) b = a;
```

dynamic ឬ var ?

- dynamic គឺអាចប្រើសំរាប់អោយ object ទាំងអស់ក្នុងភាសាណ Dart
- var គឺជា keyword សំរាប់ប្រកាសអញ្ជីតិណាមួយក៏បាន
- var ខ្លួនពី dynamic គឺត្រូវបានដែលអោយតម្លៃដំបូង (initialize) បើយើងអោយតម្លៃដំបូងជាប្រភេទ integer វានឹងជាប់ជាន់ integer រហូត។ តែ dynamic គឺអាចប្រើប្រភេទលក្ខាយបាន។

```
void example(){
    var a = 10;
    a = 10.5; //error

    dynamic b = 10;
    b = 10.5; //no error
}
```

លេខ និងចំនួនក្នុងជាតា

```
main() {  
  
    //Number  
    num age = 23;           // 23  
    num pi = 3.14;          // 3.14  
    int year = 2000;         // 2000  
    double hafl = 0.5;       // 0.5  
  
    //Boolean  
    bool isTrue = true;      // true  
    bool isFalse = false;     // false  
}
```

final និង const

- final និង const គឺជាប្រើសំរាប់កំណត់មិនអាយអញ្ញតិធម្មយដ្ឋាស់បុរតំលែបានដូចត្រា
គ្រាន់តែមែន const គឺសំរាប់ប្រកាសអញ្ញតិនៅក្នុង class ហើយ final សំរាប់បានទាំងក្នុង
និងក្នុង class។
- ចំនួចសំខាន់មួយឡើតរបស់ const គឺការកំណត់តម្លៃអាយថែរ ក្នុងពេល compile។

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

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

const double earthGravity = 9.807;
final int million = 1000000;

class MyApp extends StatelessWidget {

    const double pi = 3.14; //error
    final double PI = 3.14; //ok

    @override
    Widget build(BuildContext context) {
        return MaterialAnn(
```

និមិត្តសញ្ញាប្រមាណវិធី

- + (បុក)
- - (ដែក)
- * (គុណា)
- / (ចែក)
- ~/ (ចែកយកតំលៃជាន់ integer)
- % (ចែករកសំណាល់ | Modulo)
- ++ (កែន)
- -- (ចូល)
- ++ និង -- អាចប្រើបានចំពោះវិធី variable ដែលមិន null ត្រូវបានដោរៈ

++ និង -- ជាមួយនឹងអញ្ជូតដែលអាច NULL

- ក្នុងខាងក្រោមនេះ Error:

```
int? _counter;  
_counter++;
```

យើងត្រូវប្រមកជាបែបនេះវិញ៖

```
int? _counter;  
_counter = (_counter ?? 0) + 1;
```

និមិត្តសញ្ញាណបែងដោយ

Operator	Description	Example
>	Greater than	(A > B) is False
<	Lesser than	(A < B) is True
>=	Greater than or equal to	(A >= B) is False
<=	Lesser than or equal to	(A <= B) is True
==	Equality	(A==B) is True
!=	Not equal	(A!=B) is True

កិច្ចការសម្រាប់ក្នុងផ្តា

Operator	Description	Example
&&	And – The operator returns true only if all the expressions specified return true	(A > 10 && B > 10) is False.
	OR – The operator returns true if at least one of the expressions specified return true	(A > 10 B > 10) is True.
!	NOT – The operator returns the inverse of the expression's result. For E.g.: !(7>5) returns false	!(A > 10) is True.

លក្ខខណ្ឌកុង Dart

Condition in Dart

- if Statement
- for Loop
- while , do while
- break & continue
- switch case

if Statement

```
if (a > b) {  
    print(a);  
} else if (a < b) {  
    print(b)  
} else if (a > 0 && b >= 0) {  
    print(b)  
} else if (a == 0 || c != 0) {  
    print(b)  
} else {  
    print("equal");  
}
```

Ternary operator (? :)

```
int a = 5;  
bool b = (a > 0 ? true : false);  
String c = (b ? "Hello" : "Hi");  
int d = a > 0 ? 1 : 2;
```

for Loop

```
String message = "Hello Dart";
for (var i = 0; i < 5; i++) {
  print("$i $message ");
}
```

```
List numbers = [0, 1, 2];
for (var x in numbers) {
  print(x); // 0 1 2
}
```

while និង do while loop

```
int order = 1;  
while(order<10) {  
    print(${order++});  
}
```

```
do {  
    print(order);  
} while (order > 1);
```

break និង continue

```
int order = 1;
while(order<10) {
    if(order == 5) break; // stop the loop
    print("${order++}");
}
```

```
int order = 1;
while(order<10) {
    if(order == 5) continue; // skip the loop
    print("${order++}");
}
```

switch case

```
var order = 1;  
switch (order){  
    case 1:  
        print(order);  
        break;  
    case 2:  
        print(order);  
        break;  
    default:  
        print("None");  
}
```

Object Oriented Programming in Dart

របៀបសរស់របៀប Function ក្នុង Dart

- Function គឺជាសំណុំនៃយុទ្ធបញ្ជា ដែលមានតាមរយៈក្នុងសំចែករាងអ្នីមួយ។

```
Type functionName (data_type param_1, data_type param_2){  
    return action;  
}
```

- Function អាចសរស់របៀបយិនចាំបាច់ប្រកាស return type ក៏បាន ហើយ datatype របស់ parameter មិនជាក់ក៏បានដ៏ឡើ ពេលដែលយើងមិនជាក់ return type គឺវានឹងជាក់ dynamic អាយុយើង៖

```
getName(name){  
    return "hello" + name;  
}
```

Function - របៀបស្រើស្រាវជ្រ័យ

- យើងអាចប្រើសញ្ញា => សំរាប់ដំឡើសមាយក្នុង function ខ្លួច
- សញ្ញា => អាចប្រើដំឡើសមាយ return ប្រសិនបើមានការបានតែម្ខោគដោយ function ។

```
Type functionName (data_type param_1, data_type param_2) => expr;
```

```
bool isPositiveNum(int a) => a >= 0;  
void showSum(int a, int b) => print("${a+b}");  
  
void main(){  
    print(isPositiveNum(-1));           // false  
    showSum(5, 7);                   //print("12")  
}
```

Function – Optional Parameter និង Default Value

- Function ក្នុងភាសា Dart មានលក្ខណៈពីសេសម្បយគឺវា អាចមាន optional parameter (បានផ្តល់តម្លៃជាក់ក់បានមិនជាក់ក់បាន)។ Optional parameter មានព្រមទាំង
- positional optional parameter: []

```
showProfile(String name, [int age = 0, String gender = "male"]){
    //do something
}
showProfile("Sok San");
showProfile("Sok San", 25);
showProfile("Sok San", 25, "male");
```

- Naming optional parameter: { }

```
showProfile(String name,{int age = 0, String gender = "female"}){
    //do something
}
showProfile("Sok San");
showProfile("Sok San", age: 25);
showProfile("Sok San", gender: "male", age: 25);
```

ណែនាំអ៊តិ Class

- class គឺជាពុម្ពសំរាប់បង្កើត object
- នៅក្នុង file dart មួយ, យើងអាចដាក់ class បានច្រើន ដោយលេខោះ class និងលេខោះ file ពី ចំណាត់ថ្នាក់មានលេខោះដូច Java នោះទេ។
- ក្នុងភាសា dart គឺត្រូវ keyword សំរាប់ public និង private ទេ។ គឺត្រូវនៅក្នុងក្រាន់ដោយក្នុង underscore ពីមុខលេខោះ class គឺត្រូវជាទិន្នន័យជាទិន្នន័យ។ បើមិនដាក់ underscore នៅវាតី ជាទិន្នន័យជាទិន្នន័យ។ សំរាប់ protected គឺមិនមានឡើក្នុង Dart។

```
class Student{  
    late String name;  
    late int age;  
}  
  
class _Subject{  
    late String title;  
    late double duration;  
}
```

សំគាល់ class

- Fields ឬ property គឺជាអញ្ជូនដែលបង្កើតក្នុង class
- Setters និង Getters: គឺជាលក្ខណៈកំណត់សិទ្ធិក្នុងការបញ្ចូលនិងទទួលតម្លៃរបស់ property។ 1 Setter អាចរោងគេជាក់តម្លៃចូល property បាន។ នឹង getter អាចរោងគេយកកំម្លៃរោងគេយ property បាន។
- Constructors មានត្រនាទីសំរាប់កំណត់ទីតាំង memory និង object ដួង ព្រមទាំងជាកំន្លែងសំរាប់ផ្តល់តម្លៃដំបូង (default value) និងក្នុងផែង។
- Method: ជាជម្យភាក់ដូចត្រូវនិង function បែបធម៌ដែលគ្រាន់តែ method រាជធានីក្នុង class ឬក្នុងរបៀប។ function គឺមិនស្ថិតនៅក្នុង class ឬក្នុងរបៀប។

Instance ແລະ object

```
class Student{  
    late int id;  
    late String name;  
}
```

```
void main() {  
    Student s1; //s1 is an object with a value of null  
    Student(); //create instance, call to the constructor Student()  
    Student s2 = new Student(); //create s2 object and instance Student()  
    print("$s1 and $s2");  
}
```

Setter និង Getter

- Setters និង Getters គឺជាលក្ខណៈកំណត់សិទ្ធិភាពការបញ្ចូលនិងទទួលតម្លៃរបស់ property ។ ១ Setter អាចធ្វាយគោដាក់តម្លៃចូល property បាន។ នឹង getter អាចធ្វាយគោយកំពេលយកតម្លៃ។
- ឧទាហរណ៍ ឈ្មោះម៉ាស៊ីនឡូរសព្ទដែលទាញចេញមក គឺអាចម៉ែលបាន តែមិនសរស់រច្ឆបាបីបានទេ ចំនួចនេះគឺគោលក្រោចបាន។ ចំពោះ password វិញគឺគោយបញ្ចូលបាន តែ ទាញមកវិញមិនបានទេ ចំនួចគោលក្រោចបាន។

```
void main() {  
    Student s1 = new Student();  
    s1.id = 12;           // Setter  
    s1.name = "dara";     // Setter  
    String n = s1.name;          // Getter  
    print("${s1.id} is ${s1.name}"); // Getter  
}
```

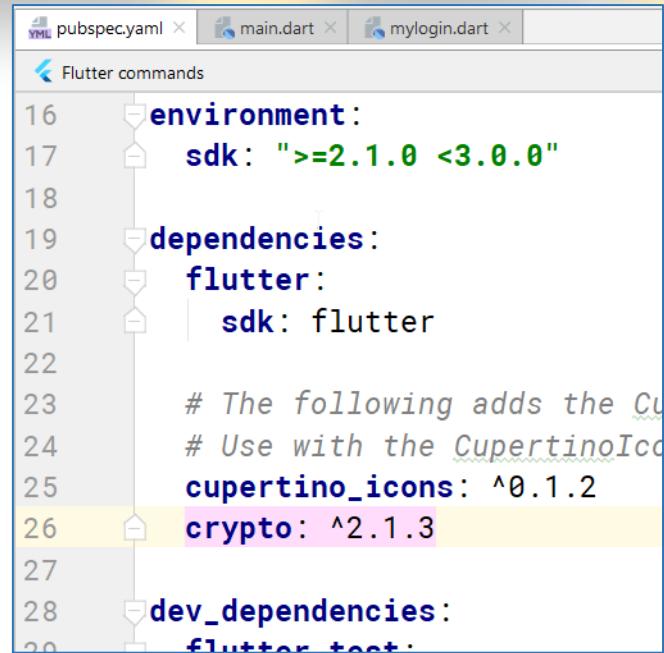
របៀបបង្កើត Setter និង Getter

```
import 'package:crypto/crypto.dart';
import 'dart:convert';

class MyLogin{
    late String _pass;

    set password(String text){
        var bytes = utf8.encode(text);
        var digest = sha1.convert(bytes);
        _pass = digest.toString();
    }

    String get hashPassword{
        return _pass;
    }
}
```



```
pubspec.yaml
environment:
  sdk: ">=2.1.0 <3.0.0"
dependencies:
  flutter:
    sdk: flutter
  # The following adds the Cupertino Icons font
  # Use with the CupertinoIcons class const
  cupertino_icons: ^0.1.2
  crypto: ^2.1.3
dev_dependencies:
  flutter_test:
```

```
MyLogin login = MyLogin();
login.password = "abc";
String p = login.password; //error on getter
login.hashPassword = "xyz"; //error on setter
String hp = login.hashPassword;
```

Constructor

- Constructor គឺជា method ពិសេសម្បយ ដែលមានត្បាងនៅក្នុងរបៀបកំណត់ទីតាំង memory អាយុយ object ដើម្បីចាត់កន្លែងសំភាប់ផ្តូលតំលៃដំបូង (default value) អាយុយក្នុងឯង។ យើងអាចបោះ constructor គឺក្នុងពេលតំណាងលក្ខាតាគារម្បយនឹងបង្កើត instance។
 - ❖ Default Constructor
 - ❖ Parameterized Constructor
 - ❖ Named Constructor
- ពេលបង្កើត constructor យើងមិនអាចជាក់ new កំបាន មិនជាក់កំបាន។

```
Student s1 = new Student();  
var s2 = Student();
```

```
class Student{  
    late int id;  
    late String name;  
    Student(){  
        id = 10;  
        name = "sok";  
    }  
}
```

```
void main() {  
    Student student1 = Student();  
    print("${student1.id} and ${student1.name}");  
}
```

ប្រកាស Constructor និងលទ្ធផល

```
class Student{  
    int id;  
    String name;  
    Student({this.id = 0, this.name = "unnamed"});  
}
```

```
void main() {  
    Student s1 = Student(); //id = 0, name = "unnamed"  
    Student s2 = Student(id: 1); //id = 1, name = "unnamed"  
    Student s3 = Student(name: "sok", id: 1);  
}
```

Named Constructor - Constructor មានឈ្មោះ

- នៅក្នុងភាសា Dart, យើងមិនអាចបង្កើត overloading method (method មានឈ្មោះដូចគ្នា) ដូចក្នុងភាសាដែលទេ។ ដូចនេះបើយើងចង់បង្កើត constructor ធ្វើដោយបង្កើតឯងត្រូវជាក់ឈ្មោះអាយី constructor នេះ (Named Constructor)។

```
class Student{  
    late int id;  
    late String name;  
  
    Student({this.id = 0, this.name = "unnamed"});
```

```
    Student.fromMap(Map<String, dynamic> map){  
        id = map['id'];  
        name = map['name'];  
    }  
}
```

```
void main() {  
    Student student1 = new Student.fromMap({'id':10,'name':'sok'});  
}
```

លំអិតកល្វ់ OOP ក្នុង Dart

Inheritance តំណែង

- Inheritance ផ្លូវជាន់ទីនៃការបោះពុម្ព ដែលអនុញ្ញាតឱ្យ subclass អាចទទួលបានមរតកដែលមានស្រាប់ពី superclass ណាមួយធ្វើឯង។
- Inheritance គឺជាការផ្តល់អនុញ្ញាតឱ្យ subclass អាចទទួលបានមរតកដែលមានស្រាប់ពី superclass ណាមួយធ្វើឯង។
- ក្រឹម extends keyword
- Subclass អាចមានតែមួយប៉ុណ្ណោះ

```
class ChildClass extends ParentClass{  
    // class member  
}
```

```
class Vehicle{          // parent class
    late String name;
    late String model;
    int year;
    Vehicle(this.name, this.model, [this.year = 1900]);
}
```

```
class Car extends Vehicle{ // sub class
    Car(String name, String model) : super(name, model);
    get getName => this.name;
}
```

Override Method (Method លើក)

- នៅក្នុង subclass យើងអាចសរសើរ override method ដោយលើបពីលើ method របស់ class មែបាន។
- ប្រើ @override ពីមុខ method

```
@override  
TypeName methodName() {  
    // TODO: implement  
}
```

```
@override  
String nameVehicle() {  
    // TODO: implement nameVehicle  
    return "this is $branch";  
}
```

Abstract Class និង Abstract Method

- Abstract class ប្រើសំរាប់ធ្វើទំនងគ្នា អាយុយ class
- ប្រើ abstract keyword ពីខាងមុខ class
- ចំណេះ abstract method គឺមិនមានប្រើ abstract keyword ដូចជាការសារ Java នៅទេ គឺ ត្រាន់តែ កំសរស់រក្សាទាងក្នុង ហើយសរសរបិទបញ្ចប់ដោយ ; ទៅជាការស្របច្បាប់
- Abstract class មិនអនុញ្ញាតឱ្យអាយុយយើង instance បានទេ
- Abstract class អាចបង្កប្រើបានត្រឹមតែ datatype តែបុណ្យ៖
- នៅក្នុង abstract class យើងអាចសរសរលាយបញ្ចប់ត្រានូវ abstract method និង method ផម្ពតាបាន។
- Abstract class គឺជាប្រយោជន៍សំរាប់ polymorphism។

```
abstract class ClassName{  
    // abstract method  
    void showName();  
}
```

```
abstract class Motorcycle{  
    String motoName = "Motorcycle";  
    void showName(); //abstract method  
    int numWheel() => 2;  
}
```

```
class Suzuki extends Motorcycle{  
    @override  
    void showName() {  
        print(super.motoName); // print parent variable  
    }  
}
```

```
void main(){  
    Motorcycle myMoto = new Suzuki();  
    myMoto.showName();  
    print(myMoto.numWheel());  
}
```

Interface

- នៅក្នុងភាសា Dart, គិតបានបង្កើតអោយ interface មានចំនួចប្រសើរជាការសារឲ្យង់ដូចជា Java និង C# ជាគើម។ Interface ក្នុង dart គឺអាចបង្កើតឡើង ដោយមិនបាត់ keyword អីទេ ពីសេសទេ គឺស្មើនឹង class សាមញ្ញដម្ពលកាតែនឹងអាចភ្លាយជារាងបានដូរ។

```
class SimpleClass{  
    void show(text) => print(text);  
}  
  
class AnotherClass implements SimpleClass{  
    @override  
    void show(text) => print("here is $t");  
}
```

- តើអ្វីដែលសំខាន់គឺ subclass គឺត្រូវ override ឡើងវិញទាំងអស់ទាំង property និង method ដែលមានក្នុង interface។ លើកលែងតែ constructor គឺមិន override ទេ។
- Class ម្នាយអាចមាន interface បានប្រចាំនៃ ហើយប្រើ implements keyword។

- ត្រូវ abstract class ឬ interface

```
// interface  
abstract class Person{  
    late String name;  
    String getName();  
}
```

```
class Student implements Person, OtherInterface{  
    @override  
    late String name;  
    @override  
    String getName() {  
        return "Sok";  
    }  
}
```

Mixin (ល្អាយ)

- នៅក្នុងភាសា Dart, Mixin វាមានវិធីសារស្ថិតិយក្នុងបន្លំមន្ទរសន្តែន functionalities ចូលក្នុងទៅ class ។ វាប្រសង់ដូចតាមនឹង Inheritance បុន្ថែវាមិនមានលក្ខណៈពីងារដូច Inheritance ពេកទៅ យើងដឹងហើយថា subclass អាច Inherit ចោរពី super class មួយតែប៉ុណ្ណោះ តែយើងអាចជាក់ Mixin ច្រើនចូលទៅក្នុង subclass ៖

```
mixin UsefulMixin1 {  
    void usefulMethod1() {  
        // ...  
    }  
}
```

```
mixin UsefulMixin2 {  
    void usefulMethod2() {  
        // ...  
    }  
}
```

```
class ImportantClass with UsefulMixin1, UsefulMixin2 {}
```

```
mixin Fly{  
    void fly() => print("Can fly");  
}  
  
mixin Swim {  
    void swim() => print("Can swim");  
}  
  
class Duck with Fly, Swim{  
    Duck(){  
        fly();  
        swim();  
    }  
}  
  
void main() {  
    Duck();  
}
```

Can fly
Can swim

Mixin ជាមួយនឹង on

- ពេលខ្លះកំណត់អាយ Mixin ជាកម្មសិទ្ធិជាកំលាំកំរបស់ superclass ណាមួយ គឺយើងអាចបង្កើត Mixin ជាមួយនឹង on ។ លក្ខណៈនេះគឺលក្ខណៈបញ្ចប់តារាង inheritance និង Mixin ។ មានន័យថាបើយើងចង់ប្រើ Mixin គឺយើងត្រូវតែ extends នូវ superclass វាដែរ៖

```
class Duck {  
    void quack() => print("Duck quacks");  
}  
  
mixin Fly on Duck{  
    void say(){  
        quack();  
    }  
  
    void fly() => print("Can fly");  
}  
  
mixin Swim on Duck {  
    void swim() => print("Can swim");  
}
```

```
class SimpleDuck extends Duck{  
}  
  
class BlackDuck with Swim{  
}  
  
class KapaDuck extends Duck with Swim, Fly{  
}
```

Callback Function

- ជាជម្រើន function មួយដែលអាចបញ្ចប់តែលកម្លាម parameter ហើយទទួលតំលៃពីវា តាមរយៈ return។ ចំណោកជា Callback function វិញ វាតី function ដែលគេប្រើវាជាទុលទៅក្នុង function ណាមួយទៀត។ Callback function មានភ្លាមាធិជាអ្នកចូលទៅយកតំលៃថែរូពី function ណាមួយដោយ ដើម្បីយកមកធ្វើការបន្ថទៀតក្នុង function របស់ខ្លួនឯង៖

```
int input(int a, int b, {required int Function(int a, int b) calc}) {
    return calc(a, b);
}

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

int multi(int a, int b){
    return a * b;
}
```

The screenshot shows the code editor with the main.dart file open. The code defines an input function that takes two integers and a callback function as parameters. It also defines two helper functions, sum and multi, which return the sum and product of their arguments respectively. The main.dart file uses these functions to calculate the sum and product of 10 and 20, and prints the results to the console.

```
int s = input(10, 20, calc: (a, b) => sum(a, b));
print("sum = $s");

int m = input(30, 10, calc: (int a, int b) => multi(a, b));
print("multi = $m");
```

main.dart

Console

Performing hot reload...

Syncing files to device iPhone 12 Pro Max...

flutter: sum = 30

flutter: multi = 300

Callback Function ដែលមានក្រុាប់ក្នុង Flutter Dart

Normal Callback:

- `typedef VoidCallback = void Function();`
- `typedef ValueGetter<T> = T Function();`
- `typedef ValueSetter<T> = void Function(T value);`
- `typedef ValueChanged<T> = void Function(T value);`

Async CallBack:

- `typedef AsyncCallback = Future<void> Function();`
- `typedef AsyncValueGetter<T> = Future<T> Function();`
- `typedef AsyncValueSetter<T> = Future<void> Function(T value);`

សំណុំធាតុ
(Collection)

List []

- ក្នុង Dart, List គឺមាន index ត្រីមត្រូវ និងមានប្រអ័ងច្បាស់លាស់ ហើយចំនួនធាតុរបស់ភាគចិត្តរឹក្សមជាន់។

```
List myList = new List();           // dynamic length  
myList.add(12);  
myList.add('hi');  
myList.add(true);  
print(myList);
```

- `isEmpty()` : បញ្ជីតម្លៃ true ប្រសិនបើ List ទទួល
- `length()` : បញ្ជីតម្លៃចំនួនសរុបនៃធាតុដែលមានក្នុង List
- `add()` : បន្ថែមធាតុ
- `remove()` : លើបធាតុ
- `contains()` : បញ្ជីតម្លៃ true ប្រសិនបើធាតុពិតជាស្ថិតក្នុង List

```
myList.add("dart");           // insert String dart to list  
myList.length                 // 1 length of list  
myList.contains("dart")       // true  
myList.remove("dart");        // remove dart from list  
myList.isEmpty                // true
```

- យើងអាចកំណត់ datatype នៅយ៉ា List ដោយជាក់បញ្ចូលនូវ generic type ក្នុងចន្ទាន់សញ្ញា < >

```
List<type> myList = new List<type>();
```

```
List<String> myList = new List<String>();  
myList.add("Hello");  
myList.add("World");  
print(myList);
```

- Initialization: យើងអាចនោយតែលដំបូង ដោយប្រើសញ្ញា []

```
List<int> myList = [];  
myList.add(12);  
myList.add(7);  
print(myList);
```

- `forEach()`: សំរាប់បញ្ចូនទិន្នន័យតាមទំនើស `for each`
- `insert()`: បញ្ចូលធាតុចូលចេញនៃទីតាំង `index` ណាមួយ
- `removeAt()`: លើបធាតុចូលចេញនៃទីតាំង `index` ណាមួយ
- `reversed`: តំបន់ក្រឡាស់
- `sort`: សំរាប់តំបន់អាជីវកម្មពីតូចទៅធំ បុរីជំឡើតូច
- `where()`: អាជីវកម្មតែលក្ពខណ្ឌណាមួយសំរាប់ស្វែងរកធាតុក្នុង `List`

```
List<int> numbers = [4, 2, 4, 1, 5];
numbers.forEach((n)=>print("number: $n"));
numbers.insert(1, 6);    print(numbers);           // [4, 6, 2, 4, 1, 5]
numbers.removeAt(0);    print(numbers);           // [6, 2, 4, 1, 5]
List<int> list1 = numbers.reversed.toList();
print(list1);            // [5, 1, 4, 2, 6]
numbers.sort((a, b) => a.compareTo(b));
print(numbers);          // [1, 2, 4, 5, 6]
List<int> list2 = numbers.where((x)=> x > 2).toList();
print(list2);            // [4, 5, 6]
```

បច្ចុបេង List

- `map()`: បង្កើត list មួយថ្មីចោរពីការបំលែងធាតុនិមួយៗ
- `List.from()` ឬ `toList()`: បង្កើត list ចោរពី collection ណាមួយ

```
List<int> numInts = [5, 6, 2, 1, 3];
print(numInts); // [5, 6, 2, 1, 3]
List<double> numDoubles = numInts.map((x) => x.toDouble()).toList();
print(numDoubles); // [5.0, 6.0, 2.0, 1.0, 3.0]
List<String> numString = List.from(numInts.map((x) => "num $x"));
print(numString); // [num 5, num 6, num 2, num 1, num 3]
```

Map {key:value}

- Map គឺជាសំណើរបាយក្រុណៈជាន់ key/value
- key និង value អាចមាន Datatype ជាប្រភេទអ្វីកបាន
- Key ត្រូវកែតម្លៃមិនស្មើ (unique) ហើយមិន null

```
var myMap = new Map(); or var myMap = {};// define map
```

```
myMap[1] = "10";
myMap["name"] = "Instinct";
myMap["isBool"] = true;
print(myMap);
```

- យើងក៏អាចកំណត់ datatype នៅយោ key និង value បានដោយ គឺជាក់តាមរយៈ generic type

```
Map<String, String> words = {};
data["apple"] = "ផ្ទៃពេញ";
data["ball"] = "បាល់";
```

```
Map<String, dynamic> jsonData = {"id":1,"product":"iMac","price":1150.0};
jsonData["qty"] = 5;
jsonData["date"] = DateTime.now().toString();
```

- `addAll()`: បន្ថែមទិន្នន័យចូលក្នុង map
- `putIfAbsent()`: បន្ថែមទិន្នន័យចូលក្នុង map ប្រសិនបើមិនទាន់មាន
- `keys`: បញ្ជីតម្លៃ keys ធាតុទាំងអស់ក្នុង map
- `values`: បញ្ជីតម្លៃ value ធាតុទាំងអស់ក្នុង map
- `isEmpty`: បញ្ជីតម្លៃ true ប្រសិនបើ map ត្រូវបានធាតុ
- `remove()`: លើបានធ្វើពី map តាមលក្ខខណ្ឌ key
- `removeWhere()`: លើបានធ្វើ តាមលក្ខខណ្ឌណាមួយ
- `containsKey()`: បញ្ជីតម្លៃ true ប្រសិនរកឃើញ key ក្នុង map
- `containsValue()`: បញ្ជីតម្លៃ true ប្រសិនរកឃើញ value ក្នុង map
- `clear()`: លើបានទាំងអស់

```
Map<String, dynamic> jsonData = {"id":1,"product":"iMac","price":1150.0};

print(jsonData);           // {id: 1, product: iMac, price: 1150.0}

jsonData.addAll({"qty": 10});

jsonData.putIfAbsent("sold", () => 5);

print(jsonData.keys);      // (id, product, price, qty, sold)

print(jsonData.values);    // (1, iMac, 1150.0, 10, 5)

print(jsonData.isEmpty);   // false

jsonData.remove("qty");

jsonData.removeWhere((k,v) => k == "product" && v == "iMac");

print(jsonData);           // {id: 1, price: 1150.0, sold: 5}

print(jsonData.containsKey("id"));        // true

print(jsonData.containsValue(1150.0));    // true
```