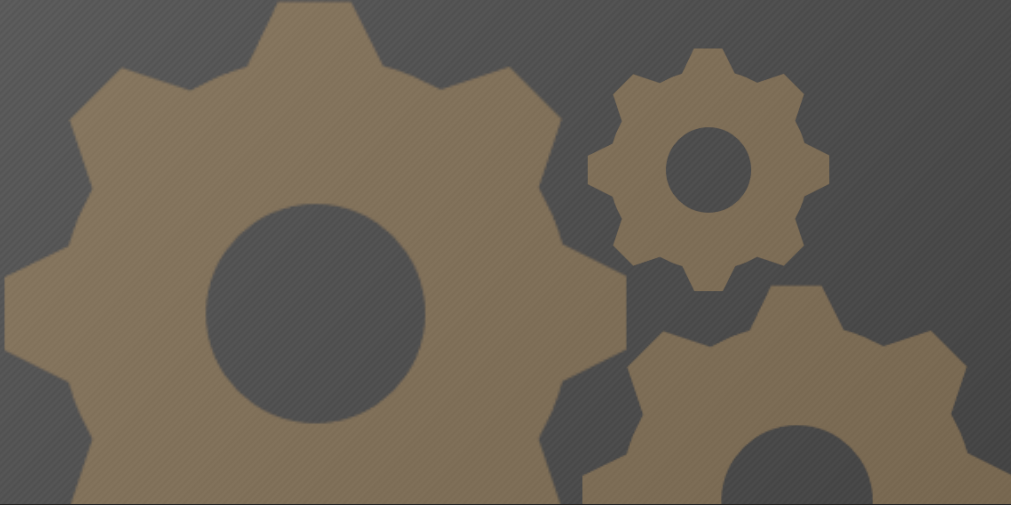




Dart programming language

Application and Game Development for Mobile Device



DART : Part 2

เตรียมความพร้อมสำหรับการสร้าง Mobile Application

<https://dart.dev>

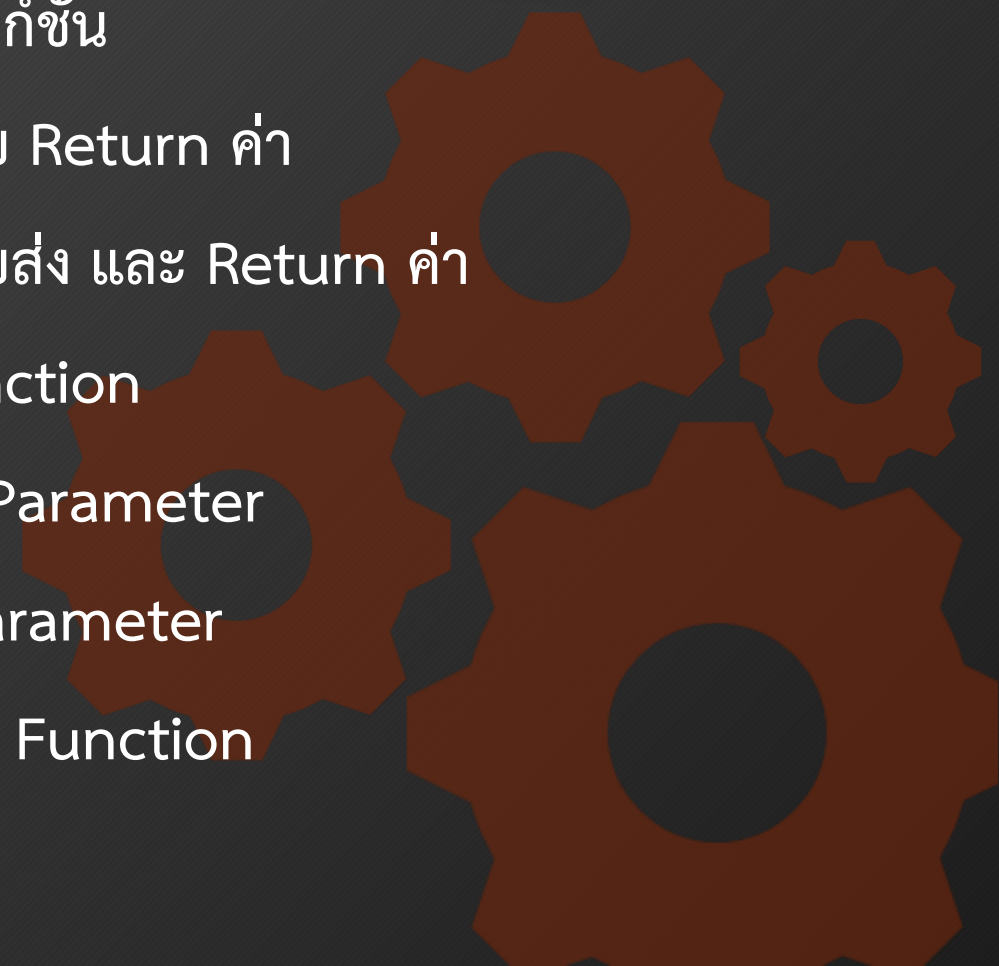


เนื้อหา (Part 2)



- ตัวดำเนินการทางตรรกศาสตร์
- Ternary Operator
- Switch..Case
- While Loop
- For Loop
- Do..While
- Break และ Continue
- การใช้งาน Loop แต่ละแบบ

- การสร้างฟังก์ชัน
- ฟังก์ชันแบบ Return ค่า
- ฟังก์ชันแบบส่ง และ Return ค่า
- Arrow Function
- Optional Parameter
- Named Parameter
- First-Class Function



ตัวดำเนินการทางตรรกศาสตร์

การกำหนดให้เงื่อนไขมีความสัมพันธ์กัน โดยมี 2 เงื่อนไขขึ้นไป

Operator	คำอธิบาย
&&	AND
	OR
!	NOT

ตัวดำเนินการทางตรรกศาสตร์

ตัวอย่าง

```
if(condition1 && condition2){  
    // เป็นจริงทั้ง 2 เงื่อนไข  
    // do something  
}
```

```
if(condition1 || condition2){  
    // เป็นจริงเงื่อนไขใดเงื่อนไขหนึ่ง  
    // do something  
}
```

```
if(!condition){  
    // ไม่ตรงกับเงื่อนไขที่กำหนด  
    // do something  
}
```

Ternary Operator คือ If..else แบบลดรูป

- รูปแบบปกติ

```
if(เงื่อนไข){  
    คำสั่งเมื่อเงื่อนไขเป็นจริง ; }  
  
else{  
    คำสั่งเมื่อเงื่อนไขเป็นเท็จ; }
```

- รูปแบบลดรูป

ตัวแปร = (เงื่อนไข) ? คำสั่งเมื่อเงื่อนไขเป็นจริง : คำสั่งเมื่อเงื่อนไขเป็นเท็จ

Ternary Operator

ตัวอย่าง

=> แบบปกติ

```
void main() {  
    int number1=100, number2=50;  
    String result;  
    if(number1>=number2){  
        result = "มากกว่าเท่ากับ";  
    } else {  
        result = "น้อยกว่า";  
    }  
    print(result);  
}
```

=> แบบลัดรูป

```
void main() {  
    int number1=100, number2=50;  
    String result;  
    result = (number1>=number2) ? "มากกว่าเท่ากับ" : "น้อยกว่า";  
    print(result);  
}
```


switch..case

- switch เป็นคำสั่งที่ใช้กำหนดเงื่อนไขคล้ายๆ กับ if
- แต่จะเลือกเพียงหนึ่งทางเลือกออกมาทำงาน
- โดยนำค่าในตัวแปรมากำหนดเป็นทางเลือกผ่านคำสั่ง case
- คำสั่ง break จะทำให้โปรแกรมกระโดดออกไปทำงานนอกคำสั่ง switch

switch..case

- รูปแบบ

```
switch(){  
    case ค่าที่ 1 : คำสั่งที่ 1; break;  
    case ค่าที่ 2 : คำสั่งที่ 2; break;  
  
    ...  
  
    default : คำสั่งที่ N; break;  
}
```

switch..case

- ตัวอย่าง

```
void main() {  
    int dayNum=1;  
    String dayName;  
    switch(dayNum){  
        case 1: dayName="วันอาทิตย์"; break;  
        case 2: dayName="วันจันทร์"; break;  
        case 3: dayName="วันอังคาร"; break;  
        case 4: dayName="วันพุธ"; break;  
        case 5: dayName="วันพฤหัสบดี"; break;  
        case 6: dayName="วันศุกร์"; break;  
        case 7: dayName="วันเสาร์"; break;  
        default: dayName="ไม่รู้จัก"; break;  
    }  
    print(dayName);  
}
```

Console

วันอาทิตย์

โครงสร้างคำสั่งแบบทำซ้ำ (Loop)

กลุ่มคำสั่งที่ใช้ในการวนรอบ (loop)

โปรแกรมจะทำงานไปเรื่อยๆ จนกว่า เงื่อนไขที่กำหนดไว้จะเป็นเท็จ จึงจะหยุดทำงาน

while

for

do...while

While Loop

การทำงานจะดำเนินการภายใน while ไปเรื่อยๆ เมื่อเงื่อนไขที่กำหนดเป็นจริง

รูปแบบ

```
while (เงื่อนไข) {  
    ชุดคำสั่งที่จะให้ทำงานเมื่อเงื่อนไขเป็นจริง ;  
}
```

While Loop (ต่อ)

ตัวอย่าง

```
void main() {  
    int count = 5;  
    while (count>0) {  
        print('(True)count = $count');  
        count--;  
    }  
    print('(False)count = $count');  
}
```

Console

```
(True)count = 5  
(True)count = 4  
(True)count = 3  
(True)count = 2  
(True)count = 1  
(False)count = 0
```

For Loop

การใช้งานการตรวจสอบเงื่อนไข เพื่อทำซ้ำ โดยมีการกำหนดค่าเริ่มต้น และเปลี่ยนแปลงค่าไปพร้อมๆ กัน เมื่อเงื่อนไขในคำสั่ง for เป็นจริงโปรแกรมจะทำงานชุดคำสั่งที่อยู่ภายใน for ไปเรื่อยๆ

รูปแบบ

```
for(ค่าเริ่มต้นตัวแปร; เงื่อนไข; เปลี่ยนค่าตัวแปร) {  
    ชุดคำสั่งที่จะให้ทำงานเมื่อเงื่อนไขเป็นจริง ;  
}
```

For Loop (ต่อ)

ตัวอย่าง

```
void main() {  
    int count = 5;  
    for (count; count>0; count--) {  
        print('(True)count = $count');  
    }  
    print('(False)count = $count');  
}
```

Console

```
(True)count = 5  
(True)count = 4  
(True)count = 3  
(True)count = 2  
(True)count = 1  
(False)count = 0
```


Do..While

โปรแกรมจะทำงานตามคำสั่ง `do` อย่างน้อย 1 รอบ เมื่อทำงานเสร็จจึงจะมาตรวจสอบเงื่อนไขที่คำสั่ง `while` ถ้าเงื่อนไขเป็นจริงจะวนกลับขึ้นไปทำงานตามคำสั่งอีกรอบ แต่ถ้าเงื่อนไขเป็นเท็จจะหลุดออกจากการทำซ้ำ

รูปแบบ

```
do {
```

```
    ชุดคำสั่งที่จะให้ทำงานเมื่อเงื่อนไขเป็นจริง ;
```

```
} while (เงื่อนไข) ;
```

Do..While (ต่อ)

ตัวอย่าง

```
void main()
{
    int number = 2, count = 1;
    print( 'โปรแกรมสูตรคูณ' );
    do{
        print( '$number * $count = ${number*count}' );
        count++;
    }while(count<=12);
    print( 'จบโปรแกรม' );
}
```

Console

โปรแกรมสูตรคูณ

```
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
2 * 10 = 20
2 * 11 = 22
2 * 12 = 24
```

จบโปรแกรม

Break และ Continue (คำสั่งที่เกี่ยวข้องกับ Loop)

- **break**

ถ้าพบคำสั่งนี้ โปรแกรมจะหลุดออกจากการทำงานใน Loop ทันที เพื่อไปทำงานคำสั่งอื่นๆ ที่อยู่นอก Loop

- **continue**

ถ้าพบคำสั่งนี้ โปรแกรมจะหยุดการทำงาน แล้วย้อนกลับกลับไปเริ่มต้นการทำงานที่ต้น Loop ใหม่

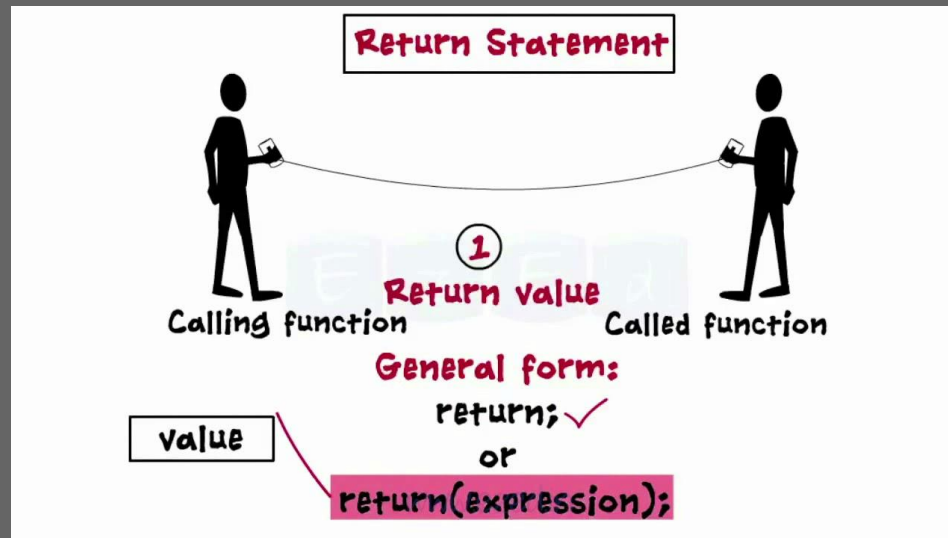
การใช้งาน Loop แต่ละแบบ

- ข้อแตกต่างของการใช้งาน Loop แต่ละแบบ
- For ใช้ในกรณี รู้จำนวนรอบ ที่ชัดเจน
- While ใช้ในกรณีที่ ไม่รู้จำนวนรอบ
- Do...while ใช้ในกรณีที่ต้องการให้ ลองทำก่อน 1 รอบ แล้วทำซ้ำไปเรื่อยๆ เมื่อเงื่อนไขยังเป็นจริง



การสร้างฟังก์ชัน

- ฟังก์ชัน คือ โปรแกรมย่อยที่นำเข้ามาเป็นส่วนหนึ่งของโปรแกรมหลัก เพื่อให้สามารถเรียกใช้งานได้ โดยไม่จำเป็นต้องเขียนโค้ดคำสั่งใหม่ทั้งหมด



การสร้างฟังก์ชัน (ต่อ)

รูปแบบของฟังก์ชัน

- ฟังก์ชันที่ 1 ไม่มีการรับส่งค่า
- ฟังก์ชันที่ 2 มีการรับค่าเข้ามาทำงาน
- ฟังก์ชันที่ 3 มีการส่งค่าออกมา
- ฟังก์ชันที่ 4 มีการรับค่าเข้ามาทำงาน และส่งค่าออกมา

ฟังก์ชัน 1 (ไม่มีการรับส่งค่า)

- รูปแบบของฟังก์ชัน ไม่มีการรับส่งค่า

```
void ชื่อฟังก์ชัน() {  
    ชุดคำสั่งต่างภายในฟังก์ชัน ;  
}
```

การเรียกใช้งาน

ชื่อฟังก์ชัน();

```
void showNumber(){  
    for (int i = 0; i < 3; i++) {  
        print('> ${i + 1}');  
    }  
}
```

```
void main() {  
    showNumber();  
}
```


ฟังก์ชัน 2 (มีการรับค่าเข้ามาทำงาน)

- รูปแบบของฟังก์ชัน มีการรับค่าเข้ามาทำงาน

```
void ชื่อฟังก์ชัน( parameter1, parameter1, ...) {  
    ชุดคำสั่งต่างภายในฟังก์ชัน ;  
}
```

การเรียกใช้งาน

```
ชื่อฟังก์ชัน(argument1, argument2, ...);
```

```
void showName(String name){  
    print("Hello $name");  
}
```

```
void main() {  
    showName("Piyaphol");  
}
```

ฟังก์ชัน 3 (มีการส่งค่าออกมา)

- รูปแบบของฟังก์ชัน มีการส่งค่าออกมา

```
type ชื่อฟังก์ชัน( ) {  
    ชุดคำสั่งต่างภายในฟังก์ชัน ;  
    return ค่าที่ต้องการส่งออก;  
}
```

การเรียกใช้งาน

ตัวแปรรับค่า = ชื่อฟังก์ชัน();

```
String returnHello(){  
    return "Hello";  
}
```

```
void main() {  
    String helloText = returnHello();  
    print("Text return $helloText");  
}
```

ฟังก์ชัน 4 (มีการรับค่าเข้าไปทำงาน และส่งค่าออกมา)

- รูปแบบของฟังก์ชัน มีการส่งค่าออกมา

```
ชนิดข้อมูลที่ต้องการส่งค่าออก ชื่อฟังก์ชัน( parameter1, parameter1, ...) {  
    ชุดคำสั่งต่างภายในฟังก์ชัน ;  
    return ค่าที่ต้องการส่งออก;  
}
```

การเรียกใช้งาน

ตัวแปรรับค่า = ชื่อฟังก์ชัน(argument1, argument2, ...);

Arrow Function

- การเขียน Arrow ฟังก์ชันจะคล้ายกับ JavaScript
- เป็นวิธีการลดรูปการเขียนฟังก์ชันแบบเดิม เหมาะสำหรับใช้งานในหนึ่งคำสั่ง
- โดยใช้ => (Arrow) ให้มีความสั้นกระชับมากขึ้น

แบบเดิม

```
String returnHello(){  
    return "Hello";  
}
```

```
int addNumber(int num1, int num2){  
    int answer = 0;  
    answer = num1 + num2;  
    return answer;  
}
```

แบบ Arrow Function

```
String returnHello() => "Hello";
```

```
int addNumber(int num1, int num2) => num1 + num2;
```

Arrow Function (ต่อ)

- `int add(int x, int y) {`
- `return x + y;`
- `}`
- `// สามารถเขียนย่อได้ว่า`
- `add(x, y) => x + y;`

Optional Parameter

- ฟังก์ชันแบบกำหนดค่าเริ่มต้น (Optional Parameter)
- เป็นการกำหนดค่าเพื่อใช้สำหรับการส่งค่าเข้าฟังก์ชันไม่ครบจำนวน Parameter ที่ฟังก์ชันนั้นๆ ต้องการในการทำงาน
- ทำให้เมื่อมีการเรียกใช้งานฟังก์ชันแล้วยังไม่ทราบค่าของ Parameter ฟังก์ชัน จะนำค่าที่กำหนดเริ่มต้นไว้มาทำงานแทน
- รูปแบบ

type ชื่อฟังก์ชัน(String name, [String customer="Normal"]){ }

Optional Parameter (ต่อ)

- ตัวอย่าง
- `int add(int x, [int y = 1]) {`
- `return x + y;`
- `}`

- `add(10, 20);` `// result: 30`
- `add(10);` `// ไม่ใส่ค่า y, ดังนั้น y = 1 result: 11`

Named Parameter

- Named Parameter เป็นการกำหนดชื่อ และลำดับของ Parameter
- เนื่องจากการใช้งานฟังก์ชันแบบปกติ จำเป็นต้องเรียงลำดับของ Parameter ให้ถูกต้องตามที่ฟังก์ชันนั้นๆ กำหนดไว้
- บ้างครั้งอาจเกิดความผิดพลาดของการเรียงลำดับข้อมูล Parameter จึงทำให้ข้อมูลที่นำไปใช้งานเกิดความผิดพลาดขึ้นได้
- Named Parameter จึงเข้ามาช่วยแก้ไขข้อผิดพลาดที่อาจจะเกิดขึ้นได้
- โดยการระบุชื่อของตัวแปรที่ต้องการจะส่งข้อมูลเข้าไปในฟังก์ชัน และไม่จำเป็นต้องเรียงตามลำดับของ Parameter ที่ฟังก์ชันนั้นประกาศไว้

Named Parameter (ต่อ)

- รูปแบบ

```
type ชื่อฟังก์ชัน ( { String name, String surname, int cardId } ) {  
    print($name $surname, $cardId);  
}
```

- การเรียกใช้งาน (parameter name : value or variable)

```
String name="piyaphol", lastname="yuenyongsathaworn";  
int CardId=745888;  
ชื่อฟังก์ชัน (name:name, cardId:CardId; lastname:lastname);
```

First-Class Function

- การทำให้ฟังก์ชันกลายเป็นตัวแปรประเภทฟังก์ชันได้
- หลักการเหมือนการสืบทอดคุณสมบัติในการเขียนโปรแกรมเชิงวัตถุ (OOP)
- รูปแบบ

```
var result = function;  
print(result());           // output : Piyaphol
```

```
function()=> "Piyaphol"
```

First-Class Function (ต่อ)

กำหนดว่าตัวแปรฟังก์ชันจะเป็น type อะไร
และมี parameter อะไรบ้าง

return-type Function (*params-type*)

```
void func1(){ ... }  
int func2(){ ... }  
String func3(int x){ ... }  
  
void main(){  
    void Function() f1 = func1;  
    int Function() f2 = func2;  
    String Function(int) f3 = func3;  
}
```

ใช้ได้กับ method

```
class People{  
    String sayHi() => "Hi!";  
}  
  
void main(){  
    People p = People();  
    String Function() f = p.sayHi;  
    print(f());    // output: Hi!  
}
```

Setup Tools

- Download Visual Studio Code
- Install Visual Studio Code
- Install extensions Dart
- Test dart and coding



Part 1 ✓

- Dart เบื้องต้น
- การแสดงผลข้อมูล
- Comment
- ตัวแปรและชนิดข้อมูล
- Dynamic Type
- Constant & Final
- กฎการตั้งชื่อตัวแปร
- จัดการอักขระและข้อความด้วย String
- ตัวดำเนินการทางคณิตศาสตร์
- ตัวดำเนินการเปรียบเทียบ
- ตัวดำเนินการเพิ่มและลดค่า
- Compound Assignment
- If Statement
- If..Else
- If แบบหลายเงื่อนไข

Part 2 ✓

- ตัวดำเนินการทางตรรกศาสตร์
- Ternary Operator
- Switch..Case
- While Loop
- For Loop
- Do..While
- Break และ Continue
- การใช้งาน Loop แต่ละแบบ
- การสร้างฟังก์ชัน
- ฟังก์ชันแบบ Return ค่า
- ฟังก์ชันแบบส่ง และ Return ค่า
- Arrow Function
- Optional Parameter
- Named Parameter
- First-Class Function

Part 3

- โครงสร้างข้อมูล List
- List Properties & Function
- เข้าถึงสมาชิกใน List ด้วย For
- เข้าถึงสมาชิกใน List ด้วย ForEach
- ฟังก์ชันเพิ่มสมาชิกใน List
- ฟังก์ชันลบสมาชิกใน List
- โครงสร้างข้อมูล Map
- แปลง List เป็น Map
- แปลง Map เป็น List
- ทฤษฎีการเขียนโปรแกรมเชิงวัตถุ
- การสร้าง Class
- การสร้าง Object
- Public / Private
- Getter / Setter
- Constructor
- การสืบทอดคุณสมบัติ
- Overriding Method

