

Flutter & Dart Complete Guide

Prepared by Eng. Yousef Salah Kassem

DAFL 1
Dart Type System & Control Flow

Lecture 2



Chapter

02

Table Of Content

- Dynamic Language
- Final and Const
- Control Flow
- Switch Statement
- Enumerations



Dynamic Language

Dynamic Type

- ◆ Dart is an optionally typed language.
- ◆ If the variable type is not specified explicitly, then the variable type is dynamic. The dynamic keyword is used for type annotation explicitly.



Variable Default Value

- ❖ While declaring the variable without initializing the value then the Dart compiler provides default value (Null) to the variable.
- ❖ Even the numeric type variables are initially assigned with the null value.
- ❖ Let's consider the following example. `int count;`



Live Coding

Final & Const

Final and Const

- ❖ When we do not want to change a variable in the future then we use final and const. It can be used in place of var or in addition to a type.
- ❖ A final variable can be set only one time where the variable is a compile-time constant. The example of creating a final variable is given below.
- ❖ Example -
final name = 'Rashmi'; // final variable without type annotation.
final String msg = 'Hi?'; // final variable with type annotation.



Chapter
02

Final and Const

- ❖ The `const` is used to create compile-time constants. We can declare a value to compile-time constant such as number, string literal, a `const` variable, etc.

```
const a = 1000;
```

- ❖ The `const` keyword is also used to create a constant value that cannot be changed after its creation.

```
var f = const[];
```

- ❖ If we try to change it, then it will throw an error.

```
f = [12]; //Error, The const variable cannot be change
```



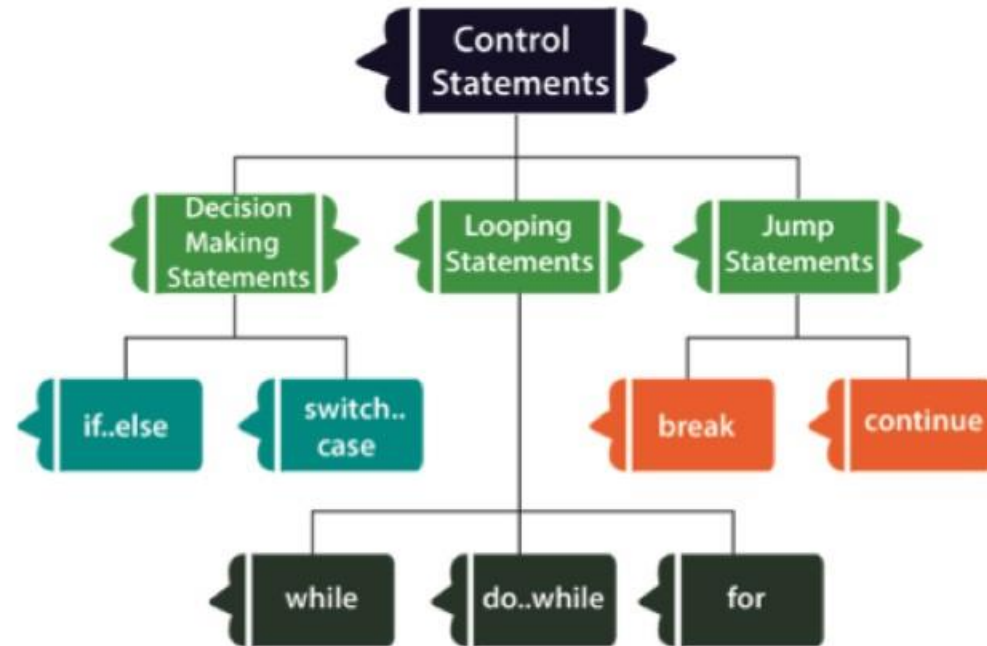
Live Coding

Control Flow

Control flow statement

- ❖ Control flow statement can be categorized mainly in three following ways.
 - Decision-making statements
 - Looping statements
 - Jump statements





If Statement

```
if (condition) {  
    //statement(s)  
}
```



If-else Statement

```
if(condition) {  
    // statement(s);  
}  
else{  
    // statement(s);  
}
```



Chapter
02**If-else if -else Statement**

```
if (condition1) {  
    // statement(s)  
}  
else if(condition2) {  
    // statement(s)  
}  
else if (conditionN) {  
    // statement(s)  
}  
else {  
    // statement(s)  
}
```



Chapter
02**Nested If**

It is possible to write an if in another if statement also.



Chapter
02**Conditional Operators**

- The Conditional Operator is same as if-else statement and provides similar functionality as conditional statement.
- It is the second form of if-else statement. It is also identified as "Ternary Operator". The syntax is given below.
- Syntax 1 -
condition ? exp1 : exp2

If the given condition is TRUE then it returns exp1 otherwise exp2.

Conditional Operators

❖ Syntax 2 -

- `exp1 ?? expr2`
- If the `exp1` is not-null, returns its value, otherwise returns the `exp2`'s value.

Conditional Operators

```
void main() {  
    var x = null;  
    var y = 20;  
    var val = x ?? y;  
    print(val);  
}
```

Chapter

02

Conditional Operators

```
void main() {  
    var a = 30;  
    var output = a > 38 ? "value greater than 10":"value lesser than equal to 30";  
    print(output);  
}
```

Chapter
02**switch case**

```
switch( expression ) {  
  
    case value-1: { //statement(s)  
  
        Block-1;  
    }  
  
    case value-N: {  
  
        break;  
        //statement(s)  
  
        Block-N;  
    }  
  
    break;  
}  
  
default: { //statement(s);
```



For Loop

```
for (var i in expression) { //statement(s)  
}
```

```
for(Initialization; condition; incr/decr) { // loop body  
}
```



while Loop

```
while(condition) {  
    // loop body  
}
```



do while Loop

```
do{  
    // loop body  
} while(condition);
```



Live Coding

Enums

Chapter
02

Enums

Enums, or enumerated types, are a data type that allows you to define a set of named constants. Enums are useful for representing fixed sets of values, such as the days of the week, the months of the year, or the different states of a button.

Enums can be used to improve the readability, maintainability, and performance of your Flutter apps in a number of ways. For example, enums can be used to:

- Avoid using magic numbers in your code.
- Make your code more self-documenting.
- Improve the performance of your code by using type checking.
- Make your code more reusable.



Chapter
02

Creating Enums

- ❖ To create an enum in Flutter, you use the `enum` keyword. For example, the following code creates an enum called `Color` with three constants: `red`, `green`, and `blue`:

```
enum Color { red, green, blue }
```
- ❖ Once you have created an enum, you can use it to declare variables and parameters. For example, the following code declares a variable called `favoriteColor` of type `Color`:

```
final Color favoriteColor = Color.blue;
```



Chapter
02

Creating Enums

- ❖ You can also use enums to define function parameters. For example, the following code defines a function called

`printColor()` that takes a `Color` parameter:

```
void printColor(Color color) {  
    print(color);  
}
```



Chapter
02

Enums in Switch Statement

Enums are particularly useful for use in switch statements. For example, the following code uses a switch statement to print the name of the selected color:

```
final Color selectedColor = Color.red;

switch (selectedColor) {
    case Color.red:
        print('Red');
        break;
    case Color.green:
        print('Green');
        break;
    case Color.blue:
        print('Blue');
        break;
    default:
        print('Unknown color');
}
```



```
enum Fruit { apple, banana, orange }

void main() {
    // Create a variable of type Fruit.
    Fruit favoriteFruit = Fruit.apple;

    // Switch on the favoriteFruit variable.
    switch (favoriteFruit) {
        case Fruit.apple:
            print('Your favorite fruit is an apple!');
            break;
        case Fruit.banana:
            print('Your favorite fruit is a banana!');
            break;
        case Fruit.orange:
            print('Your favorite fruit is an orange!');
            break;
    }

    // Get the index of the favoriteFruit variable.
    int fruitIndex = favoriteFruit.index;

    // Print the index to the console.
    print('The index of your favorite fruit is: $fruitIndex');

    // Get the name of the Fruit enum.
    String fruitName = Fruit.values[fruitIndex].name;

    // Print the name to the console.
    print('The name of your favorite fruit is: $fruitName');
}
```

Live Coding

What is Next ?

<https://zapp.run/>



Thank You