LỚP FLUTTER 2024
SYNTAX BASICS

Leader: Nguyễn Tiến Duy

BIẾN (VARIABLES)

BIẾN

Biến

Khai báo biến

```
var name = ‘Duy’;
```

Biến tên `name` chứa tham chiếu đến đối tượng `String` với giá trị `‘Duy’`

Kiểu dữ liệu của name tự được nhận là String.

Nếu không giới hạn kiểu dữ liệu của biến ta có thể thay đổi giá trị mang kiểu dữ liệu khác bằng cách dùng Object, dynamic

```
Object name = 'Bob’;

name = 4;
```

BIẾN

Null-safety

Null-safety ngăn chặn việc truy cập vô ý vào một đối tượng null. Khi cố tình truy cập hệ thống sẽ thông báo là lỗi ở thời điểm biên dịch

`Ngoại lệ có toString() và hashCode.`

BIẾN

Null-safety

1. Biến có thể nhận null phải được thông báo rõ rang khi khai báo.

```
String? name;  // Khai báo đối tượng nullable, nó có thể nhận giá trị null hoặc một String, giá trị mặc định là null

String name;   // Khai báo đối tượng non-nullable, chỉ nhận giá trị là một String, cần khởi tạo trước khi sử dụng
```

2. Chỉ có thể sử dụng biến khi đã khởi tạo, tránh các lỗi liên quan khi gọi một thuộc tính hay phương thức mà đối tượng `null` không hỗ trợ

3. Không thể truy cập, gọi một thuộc tính hay phương thức với kiểu nullable (trừ ngoại lệ)

BIẾN

Late variables

Sử dụng trong 2 trường hợp:

- Khai báo một biến không rỗng được khởi tạo sau khai báo
- Khởi tạo lười (Lazily initializing)

Thường thì luồng điều khiển của Dart phát hiện ra việc sử dụng biến non-nullable được cho giá trị non-null trước khi sử dụng, nhưng trong một số trường hợp thì không (sử dụng biến top-level hoặc biến instance).

Khi ta chắc chắn một biến đã được khởi tạo trước khi dùng nhưng Dart không cho, ta có thể fix bằng cách sử dụng khai báo late

```
late String description;

void main() {
  description = ‘cute cute';
  print(description);
}
```

BIẾN

Final & Const

Không muốn thay đổi giá trị của một biến? Hãy sử dụng final hoặc const

Biến final chỉ có thể đặt giá trị 1 lần, biến const cần khởi tạo ngay khi khai báo

Ngoài ra, const còn dùng cho các giá trị không đổi, các hàm tạo ra các giá trị không đổi.

```
var foo = const [];

final bar = const [];

const baz = []; // Equivalent to `const []`
```

Final là đối tượng không thể thay đổi nhưng các trường của nó có thể thay đổi, còn const thì không

TOÁN TỬ (OPERATORS)

TOÁN TỬ (OPERATORS)

Toán tử số học

+ - * / %

~/ : Chia lấy nguyên

++ --

Toán tử quan hệ

== != > < >= <=

Toán tử thử kiểu dữ liệu

as is is!

Sử dụng as để chuyển một Object cụ thể sang kiểu dữ liệu nào đó khi biết chắc Object là kiểu đó.

TOÁN TỬ (OPERATORS)

Toán tử gán

```
=   +=  -=
*=  /=  ~/=
>>>=    >>=     <<=
&=  |=  ^=
A ??= B (Gán cho A giá trị B nếu A null, ngược lại giữ nguyên)
```

Toán tử logic

```
!   ||  &&
```

Toán tử bitwise và dịch bit

```
&  |  ^  ~  <<  >>  >>>
```

TOÁN TỬ (OPERATORS)

Toán tử ba ngôi

```
condition ? exp1 : exp2
```

Tìm hiểu thêm: Toán tử thác nước

```
..
```

Một số toán tử khác

```
()      Gọi hàm
[]      Truy cập
?[]     Truy cập có điều kiện
.       Truy cập thành viên
?.      Truy cập thành viên có điều kiện
!       Xác nhận không null (VD: exp!)
```

GHI CHÚ

GHI CHÚ

Hàng đơn

```
//hehe
```

Nhiều hàng

```
/*
Hehe
hehe
*/
```

Viết doc

```
///
//* hehe *//
```

TÌM HIỂU THÊM: METADATA

TỪ KHÓA (KEYWORDS)

KIỂU DỮ LIỆU (TYPES)

KIỂU DỮ LIỆU (TYPES)

Number

```
int : 64 bit
Double : 64 bit
```

String: sử dụng bộ ký tự UTF-16

```
var s1 = 'Single quotes work well for string literals.';
var s2 = "Double quotes work just as well.";
var s3 = 'It\'s easy to escape the string delimiter.';
var s4 = "It's even easier to use the other delimiter.";
```

Boolean

KIỂU DỮ LIỆU (TYPES)

List

```
var list = [1, 2, 3];
var list = [ 'Car',  'Boat',  'Plane’,];
```

Set

```
var class = {'Anh', 'Hoa', 'Bac', 'Son', 'Thi’};
```

Map

```
var gifts = {
  // Key:    Value
  'first': 'partridge',
  'second': 'turtledoves',
  'fifth': 'golden rings'
};

var nobleGases = {
  2: 'helium',
  10: 'neon',
  18: 'argon',
};
```

KIỂU DỮ LIỆU (TYPES)

Typedef

```
typedef IntList = List<int>;

IntList il = [1, 2, 3];
```

TÌM HIỂU THÊM: PARTTERNS

HÀM (FUNCTION)

HÀM (FUNCTION)

Dart là ngôn ngữ hướng đối tượng nên hàm cũng là một đối tượng, có một kiểu là kiểu function. Nghĩa là hàm cũng có thể dùng như một biến hay làm đối số.

VÒNG LẶP & RẼ NHÁNH

LỚP