Ngôn ngữ Dart căn bản (Phần 1)

...

Giới thiệu

- Dart là ngôn ngữ lập trình đa mục đích được phát triển bởi Google
- Dart được sử dụng để xây dựng các ứng dụng Web, Server, máy tính để bàn và thiết bị di động
- Dart là ngôn ngữ hướng đối tượng, được xác định theo lớp, cú pháp kiểu C có thể tuỳ ý dịch mã sang Javascript
- Dart là ngôn ngữ mã nguồn mở và miễn phí
- Release: Dart version 2.5.1

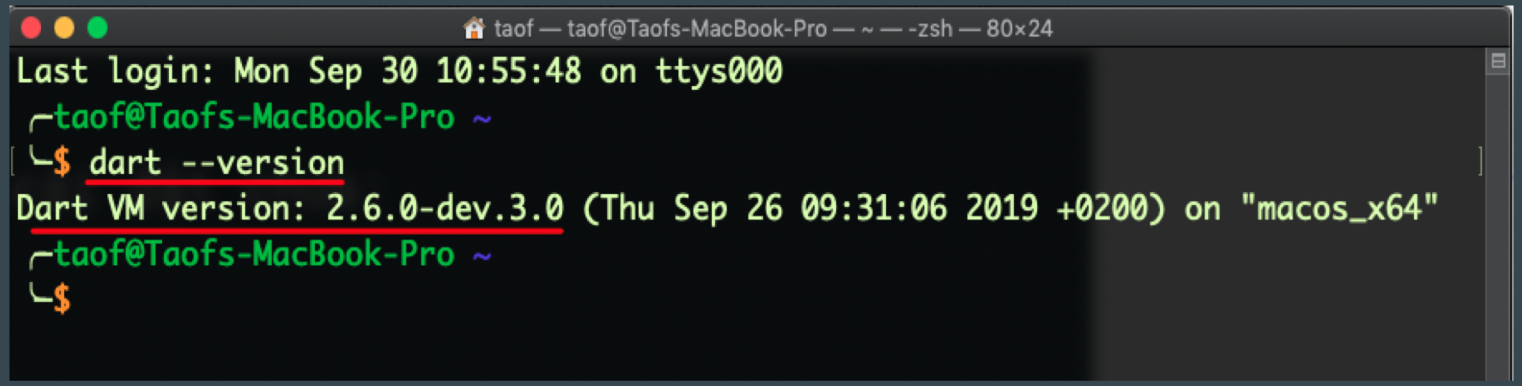
Ưu điểm của Dart

Năng suất	Nhanh	Di động	Dễ gần	Reactive
Cú pháp rõ ràng công cụ đơn giản nhưng mạnh mẽ, Dart có thư viện cốt lõi và hệ sinh thái gồm hàng ngàn package.	Dart cung cấp tối ưu hoá việc biên dịch trước để dự đoán hiệu suất và khởi động nhanh trên các thiết bị di động và web.	Dart biên dịch thành mã ARM và x86, để các mobile app có thể chạy tự nhiên trên Android và iOS. Với web, Dart biên dịch mã sang Javascript	Dart có cú pháp gần giống C++, C#, Java nên quen thuộc với nhiều nhà phát triển hiện có.	Dart phù hợp lập trình Reactive, hỗ trợ quản lý các đối tượng tồn tại trong thời gian ngắn Ví dụ như các widget UI.

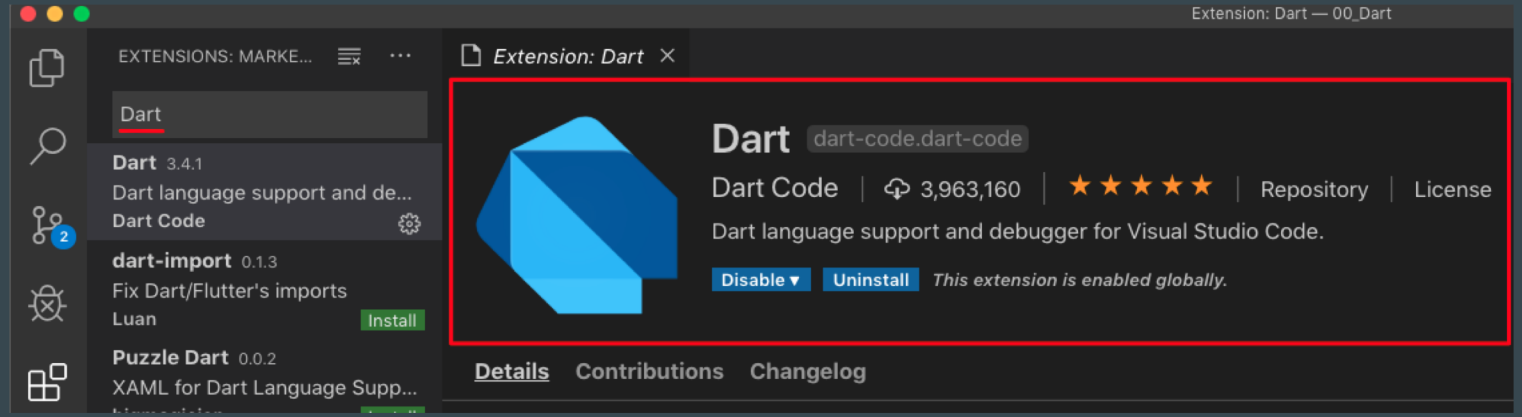
Bắt đầu

...

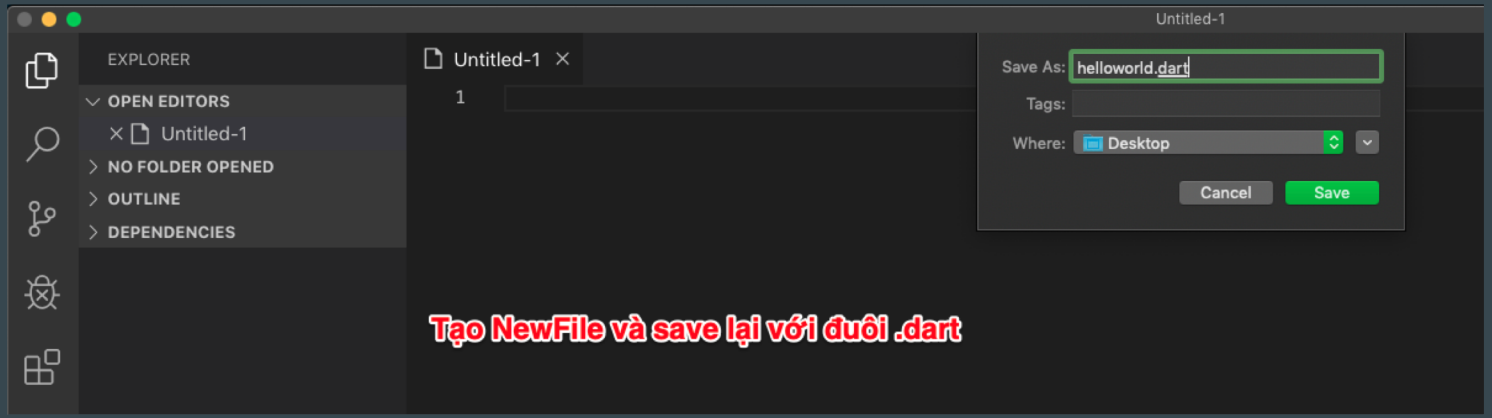
- Kiểm tra xem Dart được cài đặt chưa:

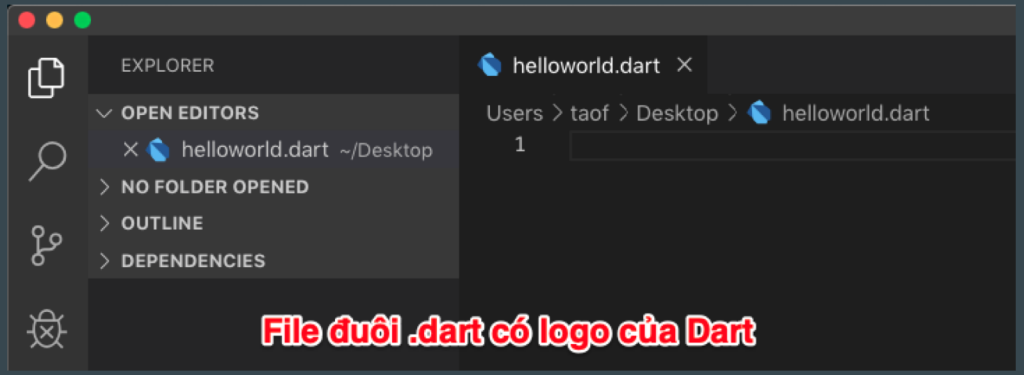


- Sử dụng IDE Visual Studi Code, cài extension Dart:



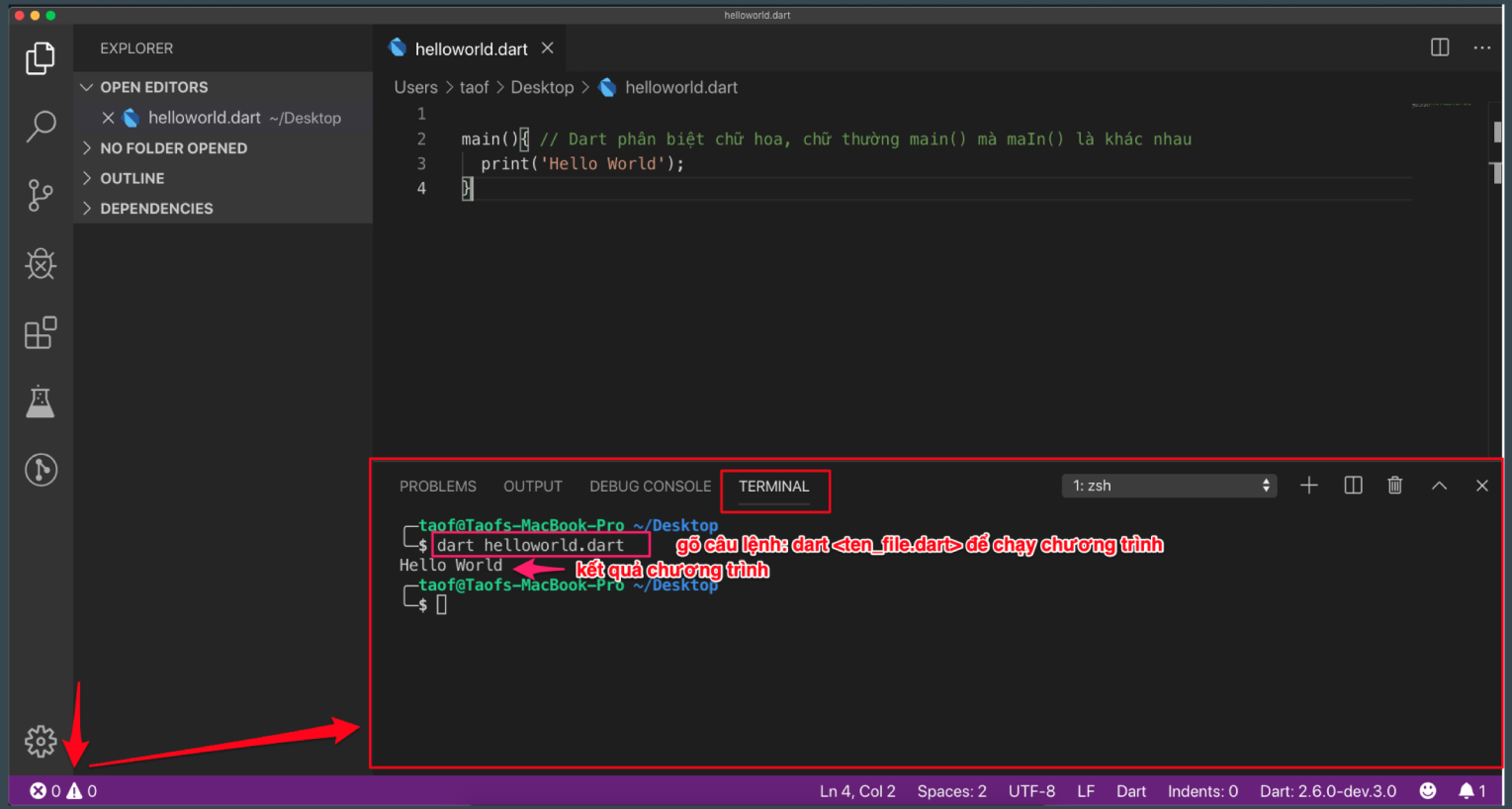
Tạo file nguồn .dart





Chương trình đầu tiên

- Một ứng dụng Dart bắt đầu chạy từ hàm main() (bắt buộc):



- Các lệnh trong Dart được viết mà các thành phần không bị ảnh hưởng bởi các khoảng trắng, kết thúc câu lệnh bằng dấu ;

```
  //Câu lệnh
  return a + b;


  //Viết lại như sau là tương đương
  return
          a
              +
      b;
```

Comment - Chú thích code

- Chú thích 1 dòng sử dụng //
- Chú thích nhiều dòng sử dụng /* */

```
// Đây là chú thích trên 1 dòng

/*
  Đây
  là chú thích
  nhiều dòng
*/
```

Khái niệm cơ bản

- Mọi thứ lưu trong biến đều là đối tượng (kể cả số, null)
- Mọi đối tường đều sinh ra từ định nghĩa lớp, mọi lớp đều kế thừa từ lớp cơ sở có tên là Object
- Khi biến chấp nhận mọi kiểu thì sử dụng từ khoá dynamic
- Dart hỗ trợ định nghĩa kiểu generic
- Dart cho phép định nghĩa hàm lồng nhau
- Dart không có từ khoá public, private, protected. Nếu thuộc tính, tên lớp bắt đầu bằng _ thì hiểu đó là private
- Các định danh bắt đầu bằng chữ (a-z, A-Z) hoặc _, theo sau là chuỗi chữ có thể kết hợp với số

Quy tắc trình bày code

Những quy tắc này không bắt buộc tuân theo, nhưng nếu có code sẽ sáng sủa, dễ đọc và thống nhất:

- Nên đặt tên kiểu UpperCamelCase cho lớp, enums, tham số
- Tên file, thư viện viết chữ thường, các từ nối với nhau bởi _
- Biến, hằng, object đặt tên kiểu lowerCamelCase, chữ đầu các từ viết thường, trừ từ đầu tiên

Biến trong Dart

…

Khai báo biến

- Biến để lưu các đối tượng khi ứng dụng hoạt động, sử dụng từ khoá `var`. Cú pháp:

```
var <ten_bien>;
// Ví dụ:
// khai báo biến không có giá trị
var a;
// Khai báo biến có giá trị
var b = 5;
```

So sánh khai báo biến có giá trị khởi tạo và không có giá trị khởi tạo:

```
var a;
// khi này giá trị của a là null
// kiểu dữ liệu của a là kiểu dữ liệu của giá trị gán vào nó
a = 'Today'; // khi này a là một string
a = 5; // khi này a là một số
print(a); // kết quả: 5


// khởi tạo giá trị ban đầu cho biến
var b = 5; // khi này b có kiểu int
b = 'a'; // lỗi: không thể gán cho b kiểu dữ liệu khác
print(b); // kết quả: 5
```

Khai báo và chỉ định kiểu dữ liệu cụ thể cho biến:

```
  // khai báo và chỉ định kiểu dữ liệu cụ thể cho biến,
  // mỗi khi gán dữ liệu vào biến thì giá trị phải cùng kiểu
  String  s = 'Đây là một chuỗi';
  int     number = 4;
  bool    isCheck = true;
  double  so = 3.2323;
  print('$s, $number, $isCheck, $so'); // sử dụng $ để truyền tham
số vào chuỗi
```

Khai báo và không quan tâm đến kiểu dữ liệu:

```
  // Trong trường hợp sử dụng biến mà biến đó không xét đến kiểu (chấp nhận gán vào nó nhiều loại kiểu) thì dùng từ khóa dynamic
  dynamic dev = 'Dev';
  dev = 3;
  dev = true;
  print(dev); // Kết quả: true
```

Hằng số trong Dart

...

Hằng số const

- Hằng số lưu giá trị mà không thể thay đổi, sử dụng từ khoá const. Cú pháp:

```
  const ten_hang_s0 = biểu_thức_giá_trị;
// Cách khai báo này gọi là hằng số lúc biên dịch, giá trị của nó phải là cụ thể ngay lúc viết code.
// Ví dụ:
  const dow_0      = 'Sunday';
  const dow_1      = 'Monday';
  const minutes    = 24 * 60;
```

Hằng số final

- Trong Dart có kiểu hằng số final, giá trị của hằng số này có thể khác nhau mỗi lần chạy

```
  // hằng số final chỉ được gán một lần duy nhất, gán lần thứ 2 sẽ lỗi (trước khi sử dụng phải có 1 lần gán).
  // Nó gọi là hằng số lúc chạy, giá trị hằng số này có thể khác nhau mỗi lần chạy
  // Ví dụ:
  var so_ngau_nhien = new Random().nextInt(500);
  final a = so_ngau_nhien * 2;
  print('$so_ngau_nhien, $a');
```

Kiểu dữ liệu

…

Các kiểu dữ liệu trong Dart

- Kiểu số: int, double
- Kiểu chuỗi: String
- Kiểu logic: bool
- Kiểu tập hợp: List, Map
- Kiểu khác: symbol, Runes

Kiểu số - int

```
// biểu diễn các giá trị số nguyên
int number = 100;
// ép kiểu string thành int
int numberString = int.parse("200");
print('int: $number, $numberString');
```

Kiểu số - double

```
// biểu diễn giá trị số thực 64bit
double bien_1 = 100; //100.0
double bien_2 = 54.32;
var bien_3 = 32.23; // khai báo thiếu kiểu dữ liệu
// ép kiểu string thành double
double bien_4 = double.parse("23.98");
// ép kiểu double thành int
int b = bien_2.toInt();
// ép kiểu double thành string
String c = bien_4.toString();
```

Kiểu chuỗi

- Biểu diễn chuỗi ký tự Unicode(UTF-16). Nó được nhập vào trong cặp nháy ‘’ hoặc nháy kép “”.
- Dùng ký hiệu \’, \” để biểu diễn ký tự “, ‘ trong chuỗi

```
print("Helloworld");
print('Chuỗi sử dụng dấu \"\" nháy kép);
print("Chuỗi sử dụng dấu \'\' nháy đơn");
// Để nối các chuỗi lại với nhau dùng toán tử +
String s1 = "S1";
String s2 = "S2";
String s3 = s2 + '_' + s1;
print(s3); // S2_S1
```

Kiểu chuỗi (tiếp)

```
  // Muốn nhập chuỗi trên nhiều dòng, dùng cú pháp sau (các dòng nằm giữa cặp ''' hoặc """);
  String s5 = '''
  Các dòng
  chữ trong chuỗi s5
  ''';
  //Có thể chèn một biến hoặc một biểu thức vào chuỗi bằng cách ký hiệu $tên_biến, ${biểu thức giá trị}
  var e = 10;
  var f = 20;
  String kq = "Hai số $e, $f có tổng ${e + f}";
```

Kiểu logic

- Biểu diễn logic đúng sai với 2 giá trị là true/false

```
  bool found = true;


  if (found) {
     //Do something
  }
```

Kiểu tập hợp - List

- List là một tập hợp dữ liệu, giống khái niệm Array ở các ngôn ngữ khác, quản lý phần tử theo index.

```
  // Các cách khởi tạo List
  var list1 = ['T2', 'T3', 'T4']; // khởi tạo list có giá trị ban đầu
  var list2 = new List(5); // khởi tạo list rỗng giới hạn, mảng này sau này k thể thêm phần tử
  var list3 = new List(); // khởi tạo list rỗng không giới hạn phần tử
  var list4 = List<int>(); // khởi tạo list số nguyên
```

List - thuộc tính và phương thức

```dart
var listDemo = new List();
listDemo.add('a'); // thêm một phần tử
listDemo.addAll([3, '4']); // thêm một List phần tử
listDemo.length; // kiểm tra độ dài List
listDemo.isEmpty; // trả về true nếu List rỗng
listDemo.isNotEmpty; // trả về true nếu List không rỗng
listDemo.single; // trả về giá trị phần tử nếu nó là duy nhất, ngượcl lại sẽ lỗi
listDemo.first; // trả về phần tử đầu tiên
listDemo.last; // trả về phần tử cuối cùng
listDemo.replaceRange(0, 2, [3, 5, 2]); // thay thế một đoạn bằng một đoạn
listDemo.reversed; // đảo list
listDemo.remove('a'); // xoá phần tử theo giá trị
listDemo.removeAt(0); // xoá phần tử theo index
listDemo.removeRange(1, 2); // xoá phần tử theo đoạn index
listDemo.clear(); // xoá toàn bộ phần tử
```

Duyệt phần tử của List

- Để duyệt phần tử List, chúng ta sử dụng vòng lặp:

```
var cities = ['Hà Nội', 'Nam Định', 'Hải Phòng', 'Quảng Ning'];
for (int i = 0; i < cities.length; i++){
  print(i);
}
```

Kiểu tập hợp - Map

- Giống List, Map quản lý một tập hợp các phần tử, tuy nhiên Map quản lý phần tử theo cặp giá trị: {key: value}

```
var colors = {1: 'Green', 2: 'Red'}; // Khởi tạo map có giá trị ban đầu
var fruits = new Map(); // Khởi tạo Map rỗng
// thêm phần tử vào Map
fruits['Orange'] = 'Orange';
fruits['Mango'] = 'Yellow';
print(fruits); // {Orange: Orange, Mango: Yellow}
```

Map - thuộc tính và phương thức

- Map cũng có một số thuộc tính: length, isEmpty, isNotEmpty

```
// duyệt Map
fruits.forEach((k, v) => (print('${k}: ${v}')));
// thêm nhiều phần tử
fruits.addAll({'Tomato':'Red', 'Lemon':'Green'});
// xoá phần tử theo key
fruits.remove('Tomato');
// xoá toàn bộ phần tử
fruits.clear();
```

Runes

- String trong Dart là một chuỗi các đơn vị mã UTF-16,
- Runes là một số nguyên biểu thị một điểm mã Unicode.
- Lớp String trong thư viện dart: core cung cấp các cơ chế để truy cập rune.

Có 3 cách để truy cập:

- String.codeUnitAt() function
- String.codeUnits property
- String.runes property

Ví dụ Runes

```
import 'dart:core';
void main(){
   f1();
}
f1() {
   String x = 'Runes';
   print(x.codeUnitAt(0)); // 82
}
```

Toán tử

...

Toán tử số học

Toán tử	Ý nghĩa	Toán tử	Ý nghĩa
+	Phép cộng	%	Phép chia module (lấy phần dư)
-	Phép trừ	++a	a = a + 1, thêm 1 vào a - trong biểu thức việc tăng được thực hiện sau
*	Phép nhân	a++	a = a + 1, thêm 1 vào a - trong biểu thức việc tăng được thực hiện trước
/	Phép chia	--a	a = a - 1, trừ a đi 1 - việc trừ được thực hiện trước trong biểu thức
~/	Phép chia lấy phần nguyên	a--	a = a - 1, trừ a đi 1 - việc trừ được thực hiện sau trong biểu thức

Phép gán trong Dart

- Phép gán là = , để thực hiện gán giá trị biểu thức bên phải vào biến ở phía bên trái toán tử.

```
bien = bieu_thuc;
```

```
var a = 1 + 2 + 3;
```

- Phép gán kết hợp gán và toán tử:

a += 5	Tương đương a = a + 5
a *= 5	Tương đương a = a * 5
a / 5	Tương đương a = a / 5

Toán tử so sánh

Toán tử	Ý nghĩa
==	So sánh bằng
!=	So sánh khác
<	So sánh bé hơn
>	So sánh lớn hơn
<=	So sánh bé hơn hoặc bằng
>=	So sánh lớn hơn hoặc bằng

Toán tử logic

Toán tử	Ý nghĩa
||	Phép logic hoặc, a || b bằng true nếu a hoặc b là true
&&	Phép logic và, a && b bằng true cả a và b là true
!bieu_thuc	Phép phủ định !a, nếu a là true thì kết quả phép toán bằng false

Biểu thức điều kiện

- Nếu biểu thức điều kiện đúng thì lấy giá trị biểu_thức_1, ngược lại lấy giá trị biểu_thức_2

Cú pháp:

```
biểu_thức_điều_kiện ? biểu_thức_1 : biểu_thức_2
```

```
var a = 4;
var b = 10;
var d = (a > b) ? a : b;
print(d); // kết quả: 10
```

Một số toán tử trên lớp, đối tượng

Toán tử	Ý nghĩa
[]	Truy cập phần tử mảng
.	Truy cập phương thức, thuộc tính của đối tượng
?.	Truy cập phương thức, thuộc tính với object khác null object?.method()
as	Ép kiểu
is	Kiểm tra kiểu (trả về true)
is!	Kiểm tra kiểu (trả về false)

Cấu trúc rẽ nhánh
Vòng lặp
…

Cấu trúc if … else (dạng 1)

```
/*
   if (biểu_thức) {
    do somthing
   }
*/
  // ví dụ:
  if (20 % 5 == 0){
    print('20 chia hết cho 5');
  }
```

Cấu trúc if … else (dạng 2)

```
/* - Cú pháp:
if (biểu_thức) {
   // do something
} else {
    //do something
}
*/
```

```
// ví dụ:
var x = 20;
if (20 % 2 == 0){
   print('$x là số chẵn');
} else {
   print('$x là số lẻ');
}
```

Cấu trúc if … else (dạng 3)

```
/* Cú pháp:
// CÓ THỂ VIẾT NHIỀU LỆNH IF
if (biểu_thức_1) {
    //..Các câu lệnh
} else if (biểu_thức_2) {
    //Các câu lệnh
} else {
    //Các câu lệnh
}
*/
```

```
// ví dụ:
var a = 12;
if (a == 0) {
   print('a bằng 0');
} else if (a < 0) {
   print('a là số dương');
} else {
   print('a là số âm');
}
```

Switch … case

```
// Cú pháp
switch (biểu_thức) {
    case : giá_trị_1
      // Khối lệnh
      break;
    case : giá_trị_2
      //Khối lệnh
    break;

    default:
      //Khối lệnh mặc định
}
```

```
  var t = 0;
  switch (t) {
    case 0:
      print('Trời nắng');
      break;
    case 1:
      print('Trời mưa');
      break;
    default:
      print('Không có giá trị
nào');
  }
```

Vòng lặp for

```
// đầy đủ 3 statement
  for (var i = 1; i <= 5; i++) {
    print(i);
  }
```

```
// bỏ qua statement1
  var i = 1;
  for (; i <= 5; i++) {
    print(i);
  }
```

```
// bỏ qua statement 2
  var k = 0;
  for (;; k += 2) {
    if (k > 10) break;
    print(k);
  }
```

```
// bỏ qua statement3
  var k = 0;
  for (;;) {
    if (k > 10){ break; }
    k++;    print(k);
  }
```

Vòng lặp while

```
// Cú pháp:
while (điều-kiện) {
   //Khối lệnh
}
```

```
// ví dụ:
 var n = 0;
 while (n <= 5) {
    print(n);
    n++;
 }
```

Vòng lặp do … while

```
// Cú pháp:
do {
   //Khối lệnh
}
while (condition);
```

```
// ví dụ:
var m = 20;
 do {
   print(m);
   m++;
 } while (m <= 25);
 // thực thi ít nhất 1 lần
```

Câu lệnh break, continue

Trong vòng lặp,

- Khi gặp lệnh `continue;` nó sẽ bỏ qua các lệnh còn lại và khởi tạo vòng lặp mới
- Nếu gặp lệnh `break;` thì nó bỏ qua các lệnh còn lại và thoát vòng lặp

```
for (i = 0; i <= 70; i++) {
  if (i == 5) {
    continue; //Khởi tạo vòng lặp mới luôn
  }
  if (i >= 7) {
    break; //Thoát lặp nếu i >=7
  }
}
```

Bài tập về nhà:

1, Cho bán kính hình cầu, tính và in ra diện tích, thể tích của hình cầu đó.

Hướng dẫn: $S = 4\pi R^2$ và $V = (4/3)\pi R^3$

2, Viết chương trình nhập vào 3 số nguyên a, b, c. Tìm, in ra số lớn nhất, số bé nhất.

3, Nhập một năm công lịch bất kỳ , kiểm tra xem năm đó có phải năm nhuận hay không?

4, Viết chương trình vẽ một tam giác cân bằng các dấu * với chiều cao cho trước (chiều cao lớn hơn 1):

```
    *
  * * *
* * * * *
```

5, Viết chương trình nhập vào một số nguyên dương không lớn hơn 10000, in ra màn hình chữ số lớn thứ nhì có trong số đó (ví dụ nhập n = 1356 in ra 5). Trong trường hợp không có số lớn thứ nhì, thì in ra số lớn nhất.

Input từ bàn phím

- import thư viện dart:io
- Sử dụng stdin để nhập dữ liệu từ bàn phím

```
import 'dart:io';
void main(){
 print('Nhap ten: ');
 String str = stdin.readLineSync();
 print(str);
 print('Nhap tuoi');
 String age = stdin.readLineSync();
 print(int.parse(age));
}
```