Hari-2-Dart intro, type data & keyword

1. Dart adalah sebuah bahasa pemrograman yang dikembangkan oleh **Google** dan digunakan untuk membangun aplikasi *mobile*, *desktop*, *backend* dan *web*, bahasa pemrograman ini bertipe ***Object Oriented*** dimana struktur kode kita berada di dalam *Class*, *Class* berisi data dan *method*. Dart menggunakan *C-style syntax* yang mirip dengan C,Java. Javascript. Swift.

Menjalankan Dart

Ada Bermacam cara untuk menjalankan kode dart yang kita buat

1. Menjalankan dart di console browser
2. Menjalankan dart di tools online seperti https://dartpad.dev/
3. menjalankan dart dengan dart dev yang tersedia di website resmi dart

Menjalankan dart dengan dart dev

Pada materi ini kita akan mencoba untuk menjalankan script dengan dart.

Dart Sdk adalah software berbasis bahasa pemerograman dart yang dijalankan di sisi server. Jika biasanya dart kita kenal erat kaitannya dengan client/browser tapi dengan Dart ini kita bisa membangun server menggunakan bahasa Dart.

Rangkuman Materi Video

Install Dart

Pertama-tama tentu kita harus menginstall terlebih dahulu dart sdk di komputer kita. Berikut ini link untuk download https://gekorm.com/dart-windows/ (disarankan memilih versi STABLE). Untuk OS Windows dan macOs tinggal diikuti saja instalasinya sampai selesai, sedangkan untuk Ubuntu 18.04 kamu bisa install mengikuti https://dart.dev/tools/sdk/archive.

Untuk mengecek apakah instalasi dart sudah berhasil kita bisa jalankan script di terminal kita:

$ dart --version

Dart SDK version: 2.10.5 (stable) (Tue Jan 19 13:05:37 2021 +0100) on "windows_x64"

dengan perintah tersebut, diketahui saat ini terinstall dart dengan versi 2.10.5

Hello World

Untuk menjalankan dart mari kita coba buat file nya terlebih dahulu dengan menuliskan.

main.dart

```
void main(){
   print("Hello World")
}
```

Demikian cara untuk menjalankan dart dengan dart sdk.

Dart I/O (input dan ouput)

di dalam pemerogrman dart kita juga dapat menggunakn input dan output, penggunaannya juga tergolong mudah dan cepat

dengan menggunakan import 'dart.io'

contoh :

```
import 'dart:io';

void main(List<String> args) {

  print("masukan password: ");

  String inputText = stdin.readLineSync();

  print("password: "+ inputText);

}
```

jika code diatas error dapat di ganti untuk print nya menjadi seperti ini

```
import 'dart:io';

void main(List<String> args){

  print("masukan password");

  String inputText = stdin.readLineSync();

  print("password:  ${inputText}");

}
```

Data Type

Data Type atau dalam bahasa indonesia Tipe Data adalah sekumpulan informasi yang memiliki nilai dan karakteristik tertentu. Beberapa contoh tipe data pada dart di antaranya:

1. **Number** : tipe data angka
2. **String** : tipe data berupa text atau kumpulan karakter, biasanya string dibungkus dalam tanda petik ganda (double quote) atau tanda petik tunggal (single quote).
3. **Boolean:** tipe data dengan nilai `true` atau `false`
4. **List & maps :** daftar tipe data untuk merepresentasikan sekumpulan object

Variable

Variable adalah suatu blok data untuk menampung sekumpulan data dengan berbagai tipe data apapun. Dengan variable kita bisa menyimpan suatu nilai untuk kemudian kita olah kembali pada program kita. Untuk deklarasi variable dalam dart kita bisa gunakan sintaks `var` lalu diikuti nama variablenya.

```
void main() {

  var name = "John" ;// Tipe
  var angka = 12;
  var todayIsFriday = false ;

  print(name); // "John"
  print(angka) ;// 12
  print(todayIsFriday); // false

}
```

Waspadai Pendeklarasian variabel yang tidak bernilai !

```
var items;

print(items); //null
```

Operator adalah karakter khusus yang merepresentasikan sebuah tindakan. Operator terbagi ke dalam beberapa jenis:

1. Operator Aritmatika Operator yang melibatkan operasi matematika seperti tambah, kurang, kali, bagi.
    - Tambah **(+)**
    - Kurang **(-)**
    - Kali **(*)**
    - Bagi **(/)**
    - Modulus **(%)**
      Modulus adalah sisa bagi. Contohnya 5%3 hasilnya adalah 2, 100%5 hasilnya 0.
2. Operator Assignment (`=`), Operator untuk mendaftarkan atau meng-assign suatu nilai ke dalam suatu variable
    ```
    var angka;
    angka = 10; // Contoh assignment variable angka dengan nilai 10
    ```
3. Operator Perbandingan, Operator yang membandingkan suatu nilai dengan nilai yang lain. Hasil dari perbandingan ini akan dikembalikan dalam tipe data boolean `true` atau `false`.
    - Equal Operator (`==`)
      ```
      var angka = 100;
      print(angka == 100); // true
      print(angka == 20); // false
      ```
    - Not Equal (`!=`)
      ```
      var sifat = "rajin";
      print(sifat != "malas"); // true
      print(sifat != "bandel"); //true
      ```
    - Kurang dari & Lebih Dari (`<`, `>`, `<=`, `>=`)
      ```
      var number = 17;
      print( number < 20 ); // true
      print( number > 17 ); // false
      print( number >= 17 ); // true, karena terdapat sama dengan
      print( number <= 20 ); // true
      ```
4. Operator Kondisional, Operator yang mengkombinasikan dua nilai kebenaran . Terdapat operator AND (`&&`) dan OR (`||`)
    - OR (`||`)
      ```
      print(true || true); // true
      print(true || false); // true
      print(true || false || false); // true
      print(false || false); // false
      ```
    - AND (`&&`)
      ```
      print(true && true); // true
      print(true && false); // false
      print(false && false); // false
      print(false && true && true); // false
      print(true && true && true); // true
      ```

2. Type Data

Mengenal lebih dalam tentang tipe data string pada dart

String

String adalah tipe data yang berisi karakter-karakter dibungkus dalam tanda petik (`""` atau `''`). Karakter-karakter pada suatu string dapat diakses dengan menggunakan indeks atau posisi karakter berada. Indeks pada string selalu mulai dari 0.

```
void main() {
  var sentences = "dart";
  print(sentences[0]); // "d"
  print(sentences[2]);// "r"
}
```

String pada dart juga memiliki property dan methods tertentu. Property dan methods tersebut bisa kita gunakan dalam memanipulasi data agar sesuai dengan program yang kita inginkan.

Numbers

int : adalah tipe data berdasarkan angka yang tidak membutuhkan koma di belakang

double : adalah tipe data yang spesifik sehingga membutuhkan koma

contoh :

```
   // declare a double value
   double num2 = 10.50;

   // print the values
   print(num1); //10
   print(num2); //10.5
}
```

mengubah string menjadi integer, kita dapat mengubah tipe data di dart dengan menggunakan method num.parse()

contoh :

```
void main() {
   print(num.parse('12')); //12
   print(num.parse('10.91')); //10.91
}

dan jangan salah memasukan angka atau akan error
```

```
void main() {
   print(num.parse('12A'));
   print(num.parse('AAAA'));
}

kode ini akan menghasilkan error :
```

```
Unhandled exception:
FormatException: 12A
#0 num.parse (dart:core/num.dart:446)
#1 main (file:///D:/Demos/numbers.dart:4:13)
#2 _startIsolate.<anonymous closure> (dart:isolatepatch/isolate_patch.dart:261)
#3 _RawReceivePortImpl._handleMessage (dart:isolatepatch/isolate_patch.dart:148)
```

Mengubah dari int ke string

```
void main() {
   int j = 45;
   String t = "$j";
   print("hello"+ t);
}
```

Referensi Tambahan

https://www.tutorialspoint.com/dart_programming/dart_programming_data_types.htm

https://www.tutorialspoint.com/dart_programming/dart_programming_string.htm

https://www.tutorialspoint.com/dart_programming/dart_programming_boolean.htm

3. Keyword Pada Dart

Dart: Perbedaan Final dan Const

Terkadang kita membuat variabel dan memberi nilainya yang sangat spesifik dengan tujuan agar **tidak pernah** mengubah nilainya. Agar program dapat berjalan dengan sesuai yang di harapkan, yang terpenting adalah nilai variabel tetap sama selama masa pemakaiannya. Untuk membuat variabel yang dimaksud menggunakan **const** dan **final** yang seharusnya digunakan. Sebelum kita membahas tentang **const** dan **final**, kita perlu mengetahui perbedaan **compile-time** dan **runtime**.

Compile-Time dan *Run-Time*

Compile Time dan *Run Time* adalah istilah dari pemrograman yang mengarahkan pada tahapan di dalam ruang lingkup pemrograman. Berfungsi untuk membuat sebuah program yang pertama kali kita buat. sehingga *source code* yang kita buat akan menentukan bagaimana program akan berfungsi yang sesuai diharapkan.

Compilation dan *Compile* Time (Kompilasi dan Waktu Kompilasi)

Komputer tidak dapat memahami kode yang kita tulis dalam bahasa pemrograman yang berbeda, komputer hanya mengetahui satu bahasa yaitu bahasa mesin 1 dan 0. *source code* yang kita tulis harus dikompilasi menjadi kode mesin agar dapat dieksekusi. Proses ini yang disebut *compilation* (Kompilasi).

Sekarang kita telah memahami apa itu kompilasi (*compilation*) yang bisa kita jadikan sebagai dasar pemahaman tentang kompilasi (*compilation*) dan sekarang kita akan mempelajari tentang *compile time* (waktu kompilasi) melalui studi kasus.

Run dan Run Time

Setelah proses kompilasi, kita dapat menjalankan program kita, ketika kita memberikan var di suatu program yang kita buat menunjukkan bahwa nilai (*value*) yang ditampilkan oleh pernyataan *print,* dapat berubah setiap kali kita menjalankan program, tergantung pada jenis *user.* Hal-hal yang tidak dapat ditentukan hingga program benar-benar saat dijalankan telah *fixed* pada **waktu proses (*run time*)**

Immutable pada Dart

Sama halnya seperti bahasa pemrograman lain, Dart juga memiliki dua sifat variable yaitu mutable dan immutable. Secara sederhana I*mmutable* artinya **tidak bisa berubah** sedangkan *Mutable* artinya **bisa berubah.**

- variabel yang bersifat immutable variabel yang tidak bisa dirubah setelah diinialisasi, contoh program di bawah ini akan menghasilkan pesan error, karena kita akan mencoba mengubah nilai awal kedua kalinya.

```
void main() {
  final umur = 21;
  umur = 22;
  // error: 'umur', a final variable, can only be set once
  const age = 21;
  // age = 22;
  print(umur);
  //error: Constant variabels can't be assigned a value
}
```

- Ketika kita mendeklarasikan variabel immutable kita harus langsung melakukan inisiasi memberi nilai) pada variabel tersebut, program akan error ketika kita tidak menginisasi nilainya, seperti contoh dibawah ini

```
void main() {
  final umur;
  //error: The final variable 'umur' must be initialized

  const age;
  //error: Constant variabels 'age' must be initialized
}
```

yang benar seperti ini :

```
void main() {
  final umur = 21;


  const age = 22;

}
```

Untuk membuat immutable variabel pada Dart, kita dapat menggunakan keyword *final* dan *const*.

Final

Final (variabel yang menggunakan *keyword* **final**) diinialisasi pada saat pertama kali digunakan dan hanya disetel sekali. Dengan kata lain nilai *final* akan diketahui pada saat ***run-time***.

final dapat digunakan untuk deklarasi variabel *immutable* yang nilainya sudah ataupun belum diketahui pada saat waktu kompilasi berjalan.

```
void main() {
  final umur = 21;
  print(umur);
}
//output
21
```

contoh di atas kita telah menetepkan nilai dari variabel umur = 21, jadi pada saat di *compile,* nilai variabel sudah diketahui nilainya karena diinialisasi secara langsung dengan sebuah *value*. Sehingga ketika di kompilasi nilainya sudah ada.

ada contoh lain penggunaan ***final***, pada saat kompilasi nilai variabel belum diketahui secara langsung, variabelnya sudah di inialisiasi namun nilainya akan didapatkan saat kompilasi dijalankan.

```
void main() {
  final waktu = new DateTime.now();
  print(waktu);


}

//Output
2020-10-02 13:22:19.909275
```

DateTime.now() akan didapatkan setelah program dijalankan/kompilasi dengan nilai kembalinya berupa waktu sekarang. Oleh karena itu tiap dijalankan hasilnya berbeda. Sehingga dapat kita ketahui nilai dari variabel waktu pada kode final waktu = new DateTime.now() sesungguhnya baru akan didapatkan saat kompilasi dijalankan dan artinya sebelum kompilasi nilainya belum diketahui secara eksplisit.

> Pointnya adalah kata kunci final dapat digunakan untuk inisiasi variabel immutable yang mana nilai variabelnya sudah atau belum diketahui pada saat kompilasi berjalan

Const

Const dapat digunakan untuk deklarasi variabel **immutable** yang nilainya bersifat **konstan** dan harus sudah diketahui pada saat waktu kompilasi (***Compile time***) berjalan, artinya adalah nilai dari variabel tersebut harus sudah di berikan *value* secara langsung.

```
void main() {
  const umur = 21;
  print(umur);
}

//Output
21
```

contoh diatas saat *compile* nilai variabel sudah diketahui, variabelnya sudah diberi nilai secara langsung.

contoh lainnya saat kompilasi nilai variabelnya belum diketahui secara langsung, variabelnya sudah di inialisasi namun nilainya baru didapat saat kompilasi di jalankan.

```
void main() {
  const waktu = new DateTime.now();
  print(waktu);
}

//Ouput :
bin/main.dart:2:17: Error: New expression is not a constant expression.
  const waktu = new DateTime.now(); ^^^
```

contoh diatas akan menghasilkan pesan error karena nilai dari variabel *waktu* tidak diinialisasi dengan nilai yang bersifat **konstan** namun diinialisasi dengan new DateTime.now() yang mana nilainya akan didapatkan setelah program dijalankan. Padahal **const** sendiri memerlukan nilai secara langsung dan nilainya harus sudah diketahui sebelum program dikompilasi.

Berbeda **final** yang mana nilai dari variabelnya tidak mengapa (tidak error) kalau didapatkan pada saat kompilasi.

Rating - Feedback

Berikan Rating pada posting ini:

★★★★★ 0

Berikan kritik dan saran..

Submit