**Functions**

*Functions* pada Dart digunakan untuk menghasilkan output berdasarkan input tertentu yang diberikan, selain itu juga sebagai blok kode atau prosedur yang dapat digunakan kembali. Sadar atau tidak, sebenarnya kita telah mengimplementasikan beberapa *functions* pada kode kita. Semua program Dart dimulai dari fungsi main(). main() adalah contoh fungsi utama yang selalu kita gunakan. Selain itu, print() juga termasuk fungsi.

1. print('Hello Dart!');

Fungsi print() akan mengambil nilai *String* atau objek lainnya dan menampilkannya ke konsol. Untuk mencetak sesuatu ke konsol sebenarnya dibutuhkan beberapa instruksi yang lebih *low-level*, namun kita menjadi sangat terbantu dengan adanya fungsi print() ini dan dapat menggunakannya secara berulang.

Untuk mendeklarasikan fungsi, caranya sama dengan penulisan fungsi main() yaitu dengan menentukan tipe nilai balik atau *return value* lalu nama fungsi dan *parameter* inputnya.

1. returnType functionName(type param1, type param2, ...) {
2. return result;
3. }

Setiap fungsi Dart selalu mengembalikan nilai. Namun ada satu tipe data khusus yang bisa kita lihat pada fungsi main yaitu *return type****void***. *Keyword* void berarti fungsi tersebut tidak menghasilkan output atau nilai kembali. Biasanya fungsi seperti ini digunakan untuk kumpulan instruksi atau prosedur yang berulang dan sering digunakan.

Setelah fungsi dibuat, selanjutnya kita bisa memanggilnya pada fungsi main() atau pada bagian program lain yang Anda inginkan.

1. void main() {
2. greet(); // output : Hello!
3. }
5. void greet() {
6. print('Hello!');
7. }

Pada contoh sederhana di atas fungsi greet() memang belum menghemat banyak kode yang Anda tulis. Namun, apabila Anda memiliki 30 instruksi *greet* dan ternyata versi terbaru aplikasi Anda memerlukan perubahan teks yang ditampilkan, Anda cukup ubah satu baris kode saja, tak perlu 30 baris kode yang berbeda. Selain itu, jika Anda memiliki kode yang cukup panjang akan lebih baik jika kode tersebut dimasukkan ke dalam fungsi supaya lebih mudah dibaca.

**Function parameters**

Pada beberapa kasus fungsi bisa memerlukan input data untuk diproses. Input data ini kita kenal sebagai ***parameter***. Untuk menambahkan *parameter* ke dalam fungsi, kita bisa memasukkannya ke dalam tanda kurung. Sebuah fungsi bisa menerima nol, satu, atau beberapa parameter.

Contoh penggunaan parameter pada fungsi yang pernah kita lihat adalah pada fungsi print().

1. print('Hello Dart!');

Berikut ini adalah contoh fungsi dengan dua parameter:

1. void main() {
2. greet('Dicoding', 2015);  // output : Halo Dicoding! Tahun ini Anda berusia 5 tahun
3. }
5. void greet(String name, bornYear) {
6. var age = 2020 - bornYear;
7. print('Halo $name! Tahun ini Anda berusia $age tahun');
8. }

Sebuah fungsi juga bisa menghasilkan output atau mengembalikan nilai. Fungsi yang mengembalikan nilai ditandai dengan definisi *return type* selain *void* dan memiliki *keyword* return. Contohnya seperti berikut:

1. void main() {
2. var firstNumber = 7;
3. var secondNumber = 10;
4. print('Rata-rata dari $firstNumber & $secondNumber adalah ${average(firstNumber, secondNumber)}');
5. }
7. double average(num num1, num num2) {
8. return (num1 + num2) / 2;
9. }

Jika fungsi hanya memiliki satu baris kode atau instruksi di dalamnya, maka bisa disingkat dengan anotasi =>. Ini juga dikenal dengan nama arrow syntax.

1. double average(num num1, num num2) => (num1 + num2) / 2;
2. void greeting() => print('Hello');

**Optional parameters**

Anda memiliki fungsi seperti berikut:

1. void greetNewUser(String name, int age, bool isVerified)

Satu-satunya cara untuk bisa memanggil fungsi di atas adalah dengan cara berikut:

1. greetNewUser('Widy', 20, true);

Namun, Dart mendukung ***optional parameter***, di mana kita tidak wajib mengisi parameter yang diminta oleh fungsi. Untuk bisa membuat parameter menjadi opsional, kita perlu memasukkannya ke dalam kurung siku seperti contoh berikut:

1. void greetNewUser([String name, int age, bool isVerified])

Cara ini disebut dengan ***positional optional parameters***. Dengan *optional parameter* seperti di atas kita bisa memanggil fungsi seperti berikut:

1. greetNewUser('Widy', 20, true);
2. greetNewUser('Widy', 20);
3. greetNewUser('Widy');
4. greetNewUser();

Setiap parameter yang tidak dimasukkan akan memiliki nilai ***null***. Dengan cara ini, urutan parameter masih perlu diperhatikan sehingga jika kita hanya ingin mengisi parameter terakhir, kita perlu mengisi parameter sebelumnya dengan *null*.

1. greetNewUser(null, null, true);

Untuk mengatasi masalah di atas kita bisa memanfaatkan ***named optional parameters***. Pada opsi ini kita menggunakan kurung kurawal pada parameter.

1. void greetNewUser({String name, int age, bool isVerified})

Dengan cara ini Anda bisa memasukkan parameter tanpa mempedulikan urutan parameter dengan menyebutkan nama parameternya.

1. greetNewUser(name: 'Widy', age: 20, isVerified: true);
2. greetNewUser(name: 'Widy', age: 20);
3. greetNewUser(age: 20);
4. greetNewUser(isVerified: true);

Terkadang *null* bukanlah pilihan yang kita inginkan ketika menggunakan *optional parameter*. Sebagai solusi, kita bisa menggunakan ***default value parameter***. Kita akan memberikan nilai *default* pada parameter lalu nilai ini akan digunakan jika kita tidak memasukkan parameternya.

1. void greetNewUser({String name = "Dicoding", int age = 5, bool isVerified = false})