**fePengenalan Effective Dart**

Akhirnya kita telah sampai di modul terakhir di kelas Memulai Pemrograman dengan Dart. Pada dasarnya sebagai seorang programmer, khususnya yang bekerja dalam sebuah tim, mayoritas waktu kita akan digunakan untuk membaca kode daripada menulisnya. Untuk itulah hadir *coding convention*. *Coding convention* adalah panduan untuk bahasa pemrograman tertentu yang merekomendasikan gaya pemrograman, praktik, dan metode untuk setiap aspek program yang ditulis dalam bahasa tersebut. Konvensi ini biasanya meliputi indentasi, komentar, deklarasi, penamaan, arsitektur, dll.

*Code convention* sangat penting bagi programmer karena beberapa alasan berikut:

* 40% - 80% biaya dari sebuah perangkat lunak digunakan untuk pemeliharaan (*maintenance*).
* Sangat jarang suatu perangkat lunak dipelihara seterusnya oleh penulis aslinya.
* *Code convention* meningkatkan keterbacaan kode, memungkinkan programmer untuk memahami kode baru dengan lebih cepat dan menyeluruh.
* Source code lebih tertata rapi dan bersih sebagai sebuah produk.

Dart juga telah mengembangkan *coding convention* yang bertujuan supaya kita bisa menulis kode yang konsisten, kuat, dan cepat. Convention Dart ini dikenal dengan Effective Dart. Intinya Effective Dart dibuat untuk mewujudkan dua hal:

1. *Be consistent* (Konsisten). Ketika membahas tentang *formatting* akan banyak argumen subjektif tentang mana yang lebih baik. Namun konsistensi tentunya akan sangat membantu secara objektif.
2. *Be brief* (Ringkas). Dart dirancang supaya familiar dengan bahasa pemrograman lain seperti C, Java, JavaScript, dll. Namun Dart ditambah dengan fitur-fitur lain yang belum ditawarkan bahasa lain. Jika ada banyak cara untuk mengungkapkan sesuatu, tentu Anda akan menggunakan cara yang paling ringkas bukan?

## Panduan Effective Dart

Guidelines dari Effective Dart dibagi menjadi empat bagian sesuai dengan fungsinya.

* Style guide. Mendefinisikan aturan untuk meletakkan dan mengatur kode. Panduan ini juga menentukan bagaimana format penamaan sebuah identifier, apakah menggunakan camelCase, \_underscore, dll.
* Documentation guide. Panduan ini terkait tentang apa yang boleh dan tidak ada di dalam komentar. Baik itu komentar dokumentasi atau komentar biasa.
* Usage guide. Panduan ini mengajarkan bagaimana memanfaatkan fitur bahasa secara terbaik untuk menerapkan perilaku. Penggunaan statement atau expression akan dibahas di sini.
* Design guide. Ini adalah panduan dengan cakupan terluas namun paling tidak mengikat. Panduan ini mencakup bagaimana mendesain API library yang konsisten dan bisa digunakan.

Effective Dart memiliki banyak rules. Masing-masing aturan akan diawali dengan kata kunci untuk menentukan bagaimana sifat dari aturan tersebut. Lima kata kunci tersebut, antara lain:

* ***DO***. Ketika aturan diawali dengan DO maka praktik tersebut harus selalu diikuti.
* ***DON’T***. Sebaliknya, aturan yang diawali dengan DON’T bukan merupakan hal yang baik untuk diterapkan.
* ***PREFER***. Ini adalah praktik yang harus diikuti. Namun, mungkin ada keadaan di mana lebih masuk akal untuk melakukan sebaliknya. Pastikan Anda memahami konsekuensi ketika Anda mengabaikan aturan ini.
* ***AVOID***. Ini adalah kebalikan dari PREFER. Panduan ini menjelaskan hal-hal yang tidak boleh dilakukan, namun kemungkinan ada alasan bagus untuk melakukannya pada beberapa kejadian.
* ***CONSIDER***. Panduan ini adalah praktik yang bisa Anda ikuti atau tidak Anda ikuti, tergantung pada keadaan dan preferensi Anda sendiri.

Meskipun ada banyak sekali aturan dan pedoman yang perlu diikuti dalam Effective Dart, kita tidak perlu khawatir dengan peraturan yang sangat ketat. Karena sebagian besar panduan yang ada bisa dibilang merupakan common sense dalam membuat program, bahkan jika tidak tertulis sekalipun. Selain itu konvensi bertujuan supaya kode kita menjadi lebih bagus, mudah dibaca, dan tentunya dipelihara.

## Ringkasan Aturan Effective Dart

Setelah mempelajari code convention dan Effective Dart, pada modul ini kita akan membahas tentang beberapa contoh aturan yang ada di dalam Effective Dart.

### DO name types using UpperCamelCase.

Class, enum, typedef, dan type parameter harus menggunakan huruf kapital pada huruf pertama dari setiap kata termasuk kata pertama.

1. abstract class Animal {}
2. abstract class Mammal extends Animal {}
3. mixin Flyable {}
4. class Cat extends Mammal with Walkable {}

### DON’T use prefix letters.

Karena Dart dapat memberitahu Anda tipe, cakupan, dan properti lain pada sebuah deklarasi, maka tidak ada alasan untuk menambahkan prefix pada sebuah identifier.

1. var instance;  // good
2. var mInstance;  // bad

### PREFER starting function or method comments with third-person verbs.

Sebuah komentar dokumentasi harus fokus menjelaskan apa yang dilakukan kode tersebut. Menambahkan kata kerja dari sudut pandang orang ketiga di awal komentar adalah salah satu cara melakukannya.

1. // Returns `true` if [username] and [password] inputs are valid.
2. bool isValid(String username, String password) { }

### PREFER a noun phrase for a non-boolean property or variable.

Seorang developer yang membaca kode kita akan fokus pada apa yang ada pada property. Jika mereka lebih peduli tentang bagaimana suatu property ditentukan, maka lebih baik dibuat menjadi method dengan nama menggunakan kata kerja.

1. // Good
2. cat.furColor;
3. calculator.firstNumber;
4. list.length;
6. // Bad
7. list.deleteItems;

Sementara untuk variabel atau property booleans PREFER gunakan kata kerja non-imperative, seperti:

1. list.isEmpty
2. dialog.isOpen

### DO use ?? to convert null to a boolean value.

Aturan ini berlaku ketika sebuah expression dapat mengevaluasi nilai true, false, atau null dan Anda perlu meneruskan hasilnya ke sesuatu yang tidak menerima nilai null.

1. if(stock?.isEnough) {
2. print('Making you a cup of coffee...');
3. }

Kode di atas akan menghasilkan exception ketika stock-nya null. Untuk mengatasinya kita perlu memberikan nilai default ketika nilai awalnya null. Sehingga kodenya akan menjadi seperti berikut:

1. stock?.isEnough ?? false;

### AVOID using curly braces in interpolation when not needed.

Seperti yang kita tahu, Dart dilengkapi dengan fitur string interpolation untuk menggabungkan nilai string dengan nilai lain secara lebih mudah.

1. print('Hi, ${name}, You are ${thisYear - birthYear} years old.');

Namun jika Anda menginterpolasi identifier sederhana, maka curly braces ({}) tidak perlu ditulis.

1. print('Hi, $name, You are ${thisYear - birthYear} years old.');

### PREFER async/await over using raw futures.

Kode asynchronous bisa jadi sangat sulit untuk dibaca dan di-debug, bahkan ketika menggunakan abstraksi yang bagus seperti Future. Sintaks async-await memungkinkan Anda menuliskan kode asynchronous dengan gaya synchronous sehingga memudahkan membaca kode.

1. // raw future
2. void main() {
3. getOrder().then((value) {
4. print('You order: $value');
5. })
6. .catchError((error) {
7. print('Sorry. $error');
8. });
9. print('Getting your order...');
10. }

13. // async-await
14. void main() async {
15. print('Getting your order...');
16. try {
17. var order = await getOrder();
18. print('You order: $order');
19. } catch (error) {
20. print('Sorry. $error');
21. }
22. }

### CONSIDER making the code read like a sentence.

Penamaan dalam kode baik itu nama variabel, fungsi, maupun lainnya adalah hal yang sangat penting namun juga tidak mudah. Sebagai solusi kita bisa membayangkannya seolah sedang membuat kalimat.

1. // “If store is open ...”
2. If (store.isOpen)
4. // “hey garfield, eat!”
5. garfield.eat();
7. // Bad example
8. // Ambigu apakah memerintahkan toko untuk tutup atau mendapatkan status dari toko
9. If (store.close)

### CONSIDER using function type syntax for parameters.

Dart memiliki sintaks khusus untuk mendefinisikan parameter yang tipenya adalah fungsi. Caranya yaitu dengan menuliskan tipe kembalian sebelum nama parameter kemudian parameter dari fungsi setelahnya.

1. List filter(bool predicate(item)) { }

Sejak Dart versi 2, terdapat notasi umum untuk tipe fungsi sehingga bisa digunakan untuk parameter berupa fungsi.

1. List filter(Function predicate) { }  // function type syntax

Sebenarnya beberapa aturan di atas hanyalah sebagian dari seluruh aturan yang ada dalam Effective Dart. Selengkapnya Anda dapat mempelajari panduan dan aturan Effective Dart ini pada tautan berikut: <https://dart.dev/guides/language/effective-dart>.

Terakhir, maksimalkan panduan dan convention yang telah dibuat ini supaya kode Anda menjadi lebih berkualitas ya. Do your best!

## Rangkuman Materi

Selamat! Kita telah tiba di modul terakhir. Di sini kita mempelajari tentang code convention dari Dart. Tujuannya, supaya kita tidak hanya bisa menuliskan kode Dart, tetapi juga menuliskan kode yang berkualitas sesuai dengan standar yang ada. Mari kita uraikan materi yang sudah Anda pelajari untuk mempertajam pemahaman.

* Coding Convention merupakan panduan yang merekomendasikan gaya pemrograman, praktik, dan metode untuk setiap aspek program yang ditulis dalam bahasa tertentu.
* Alasan code convention sangat penting bagi programmer adalah:
  + Sebagian besar waktu dalam mengembangkan aplikasi dihabiskan untuk maintenance.
  + Sangat jarang suatu software dipelihara seterusnya oleh penulis aslinya.
  + Meningkatkan keterbacaan kode dan memahami kode baru dengan lebih cepat.
  + Supaya source code lebih tertata rapi dan bersih.
* Panduan Effective Dart dibagi menjadi empat bagian sesuai dengan fungsinya, antara lain:
  + Style guide, mendefinisikan aturan untuk meletakkan dan mengatur kode.
  + Documentation guide, panduan terkait dengan bagaimana memberikan komentar atau dokumentasi pada kode.
  + Usage guide, panduan terkait bagaimana memanfaatkan fitur bahasa secara terbaik untuk menerapkan perilaku.
  + Design guide, panduan mencakup bagaimana mendesain API library yang konsisten dan bisa digunakan.
* Setiap panduan akan diawali dengan lima kata kunci ini, antara lain:
  + DO, aturan yang harus selalu diikuti.
  + DON’T, aturan yang tidak baik untuk diterapkan.
  + PREFER, aturan yang harus diikuti, namun ada keadaan di mana lebih masuk akal untuk melakukan sebaliknya.
  + AVOID, aturan yang sebaiknya dihindari atau tidak boleh dilakukan.
  + CONSIDER, aturan yang bisa diikuti atau tidak diikuti, tergantung pada keadaan dan preferensi sendiri.
* Banyak sekali aturan dan pedoman terkait Effective Dart, Anda dapat temukan seluruh aturan pada tautan berikut: <https://dart.dev/guides/language/effective-dart>.

Sampai modul terakhir ini, semoga Anda telah memahami konsep dasar Dart dengan baik dan siap untuk mengembangkan aplikasi dengan Flutter. Jika belum, jangan ragu untuk membaca dan mengulas kembali modul. Manfaatkan juga forum diskusi untuk menjawab pertanyaan Anda.

Masih ada ujian akhir yang perlu Anda selesaikan agar dinyatakan lulus dan mendapat sertifikat di kelas ini. Good luck!