Do’s and Don’t in Flutter

PART-2

Lets hack! 🕵️

A surprisingly important part of ***good code is good style.***

Consistent naming, ordering, and formatting helps code that ***is*** the same ***look*** the same.

It takes advantage of the powerful pattern-matching hardware most of us have in our ocular systems.

If we use a consistent style across the entire Dart ecosystem, it makes it easier for all of us to learn from and contribute to each others' code.

Built with Flutter

Save for later

Identifiers

- ***Identifiers come in three flavors in Dart.***
  - ***UpperCamelCase*** names capitalize the first letter of each word, including the first.
  - ***lowerCamelCase*** names capitalize the first letter of each word, except the first which is always lowercase, even if it’s an acronym.
  - ***lowercase_with_underscores*** names use only lowercase letters, even for acronyms, and separate words with **_**.

DO name types using UpperCamelCase.

Linter rule: ***camel_case_types***

- Classes, enum types, typedefs, and type parameters should capitalize the first letter of each word [including the first word], and use no separators.

```
class SliderMenu { ... }

class HttpRequest { ... }

typedef Predicate<T> = bool Function(T value);
```

- This even includes classes intended to be used in metadata annotations.

```
class Foo {
  const Foo([Object? arg]);
}

@Foo(anArg)
class A { ... }

@Foo()
class B { ... }
```

@bugs_fixes
Bugs and Fixes

Follow for more

- If the annotation class's constructor takes no parameters, you might want to create a separate **lowerCamelCase** constant for it.

```
const foo = Foo();

@foo
class C { ... }
```

DO name extensions using UpperCamelCase.

Linter rule: ***camel_case_extensions***

- Like types, **extensions** should capitalize the first letter of each word (including the first word), and use no separators.

```
extension MyFancyList<T> on List<T> { ... }

extension SmartIterable<T> on Iterable<T> { ... }
```

DO name packages, directories, and source files using **lowercase_with_underscores.**

Linter rule: ***file_names, package_names***

- Using underscores as the separator ensures that the name is still a valid Dart identifier, which may be helpful if the language later supports symbolic imports.

Good

```
my_package
└─ lib
   └─ file_system.dart
   └─ slider_menu.dart
```

Bad

```
mypackage
└─ lib
   └─ file-system.dart
   └─ SliderMenu.dart
```

@bugs_fixes
Bugs and Fixes

Follow for more

DO name import prefixes using **lowercase_with_underscores.**

Linter rule: ***library_prefixes***

Good

```dart
import 'dart:math' as math;
import 'package:angular_components/angular_components.dart' as angular_components;
import 'package:js/js.dart' as js;
```

Bad

```dart
import 'dart:math' as Math;
import 'package:angular_components/angular_components.dart' as angularComponents;
import 'package:js/js.dart' as JS;
```

@bugs_fixes
Bugs and Fixes

Source Code link in Bio.

PREFER using **lowerCamelCase** for constant names.

Linter rule: ***constant_identifier_names***

- In new code, use lowerCamelCase for constant variables, including enum values.

Good

```
const pi = 3.14;
const defaultTimeout = 1000;
final urlScheme = RegExp('^([a-z]+):');

class Dice {
  static final numberGenerator = Random();
}
```

Bad

```
const PI = 3.14;
const DefaultTimeout = 1000;
final URL_SCHEME = RegExp('^([a-z]+):');

class Dice {
  static final NUMBER_GENERATOR = Random();
}
```

@bugs_fixes
Bugs and Fixes

Source Code link in Bio.

DO name other identifiers using **lowerCamelCase.**

Linter rule: ***non_constant_identifier_names***

- Class members, top-level definitions, variables, parameters, and named parameters should capitalize the first letter of each word except the first word, and use no separators.

```
var count = 3;

HttpRequest httpRequest;

void align(bool clearItems) {
  // ...
}
```

PREFER using _, __, etc. for unused callback parameters.

```
futureOfVoid.then((_) {
  print('Operation complete.');
});
```

@bugs_fixes
Bugs and Fixes

Follow for more