**前端编程规范（使用Dart语言）**

**1. 样式规范**

1.1 **类、枚举、类型定义和泛型，都需要大写开头的驼峰命名法，如下所示：**

class Foo() {

const Foo([args])

}

1.2 **命名库、包、目录、dart文件都应该是小写加下划线，如下所示：**

library peg\_parser.source\_scanner;

import 'file\_system.dart';

1.3 **将引用使用as转换的名字也需要小写加下划线，如下所示：**

import 'dart:math' as math;

import 'package:js/js.dart' as js;

1.4 **变量名、方法、参数名都应该是小驼峰命名法，如下所示：**

const pi = 3.14;

const defaultTimeout = 1000;

final urlScheme = RegExp('^([a-z]+):');

**2. 文档规范**

**2.1 注释使用///**

**2.2 使用一句简明扼要的话作为开头 空一格 将参数和返回值使用[]加持，如下所示：**

/// Throws an [IOError] if the file could not be found.Throws a

/// [PermissionError] if the file is present but could not be deleted.

**3. 使用规范**

**3.1使用相对路径导入依赖，如下所示：**

import 'src/utils.dart';

**4. 字符串**

**4.1 dart中，不需要使用+连接字符串，使用 ${} 连接字符串和变量，如下所示：**

'Hello, $name! You are ${year - birthday} years old. '

**5. 集合**

**5.1尽可能使用简单的字面量创建集合，如下所示：**

var points = [];

var addressses = {};

**5.2使用isEmpty或isNotEmpty判断集合是否为空，如下所示：**

if(lunchBox.isEmpty) return 'so hungry...'

if(words.isNotEmpty) return words.join(' ');

**5.3使用高阶方法转换序列，如下所示：**

var aquaticNames = animals

.where((animal) => animal.isAquatic)

.map((animal) => animal.name);

**5.4不要使用List.from() 除非打算更改结果的类型，有两种方法可以获取Iterable，分别是List.from() 和 Iterable.toList()，如下所示：**

var iterable = [1, 2, 3];

/// 输出`List<int>`

/// 如果使用List.from的话，会输出`List<dynamic>`

print(iterable.toList().runtimeType);

**5.6使用 whereType() 去用类型过滤一个集合，如下所示：**

var objects = [1, 'a', 'b2', 2];

var ints = objects.whereType<int>();

**6. 参数**

**6.1给参数设置默认值，不要将参数的默认值设置为null，如下所示：**

void insert(Object item, { int at = 0 }) { ... }

**7. 变量**

**7.1 不存储可以计算的值，如下所示：**

class Circle {

num radius;

Circle (this.radius);

num get area = pi \* radius \* radius;

num get circumference = pi \* 2.0 \* radius;

}

**8. 成员**

**8.1 不需要些没必要的 getter 和 setter，如下所示：**

class Box {

void contents;

}

**9. 构造函数**

**9.1 尽可能使用简单的初始化形式，如下所示：**

class Point {

num x, y;

Point(this.x, this.y);

}

**9.2 不要使用new创建对象，如下所示：**

Widget build(BuildContext context) {

return Row(

children: [

RaisedButton(

child: Text('Increment'),

),

Text('Click!'),

],

);

}

**10. 异常处理**

**10.1 使用rethrow重新抛出异常，如下所示：**

try {

somethingRisky();

} catch (e) {

if (!canHandle(e)) rethrow;

handle(e);

}

**11. 设计**

**11.1 避免为了实现流式调用而让方法返回this，如下所示：**

var buffer = StringBuffer()

..write('one')

..write('two')

..write('three');

**后端编程规范（使用Java语言）**

**后端使用Java开发，选定遵循google Java编程规范**