

СКАЧАНО С WWW.SW.HELP - ПРИСОЕДИНЯЙСЯ!

Проверить, идет ли запись!





Меня хорошо видно && слышно?

Ставьте + , если все хорошо

Напишите в чат, если есть проблемы

Platform Channels.
Работа с нативным кодом.
Flutter Plugins



Вирник Илья

Flutter Team Lead

Яндекс.Такси

Telegram: @virnik

Вирник Илья



Telegram: @virnik

Вирник Илья

- Руковожу группой Flutter-разработки приложения Яндекс.Про в Яндекс.Такси
- Один из первопроходцев во Flutter-разработке в Яндексе
- Бэкграунд — нативная iOS-разработка, выпускник Apple Developer Academy

Маршрут вебинара

Flutter Plugin

↓

PlatformChannel

↓

Типы каналов, кодеки

↓

Pigeon

Маршрут вебинара

FFI

Платформенные view

Рефлексия

Домашнее задание

Цели вебинара | После занятия вы сможете

1. Работать с платформенным кодом в кроссплатформенной среде
2. Писать собственные плагины и переиспользовать нативный код во Флаттер приложениях
3. Глубже понимать устройство фреймворка и возможности Флаттер приложений

Смысл | Зачем вам это уметь

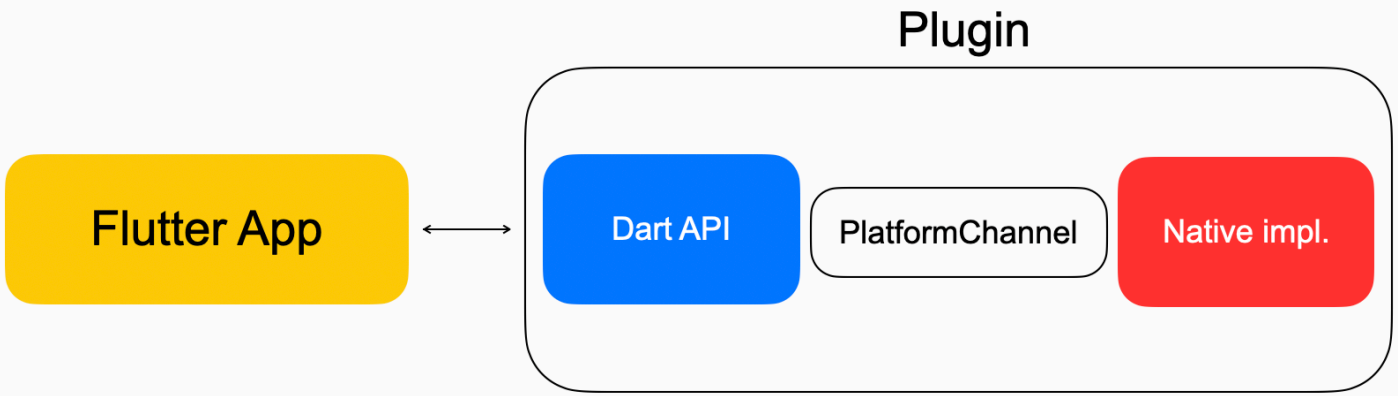
1. Реализовывать функциональность, которая недоступна фреймворку

2. Использовать нативные SDK, в том числе сторонние, Флаттер версий которых пока нет


Flutter Plugin

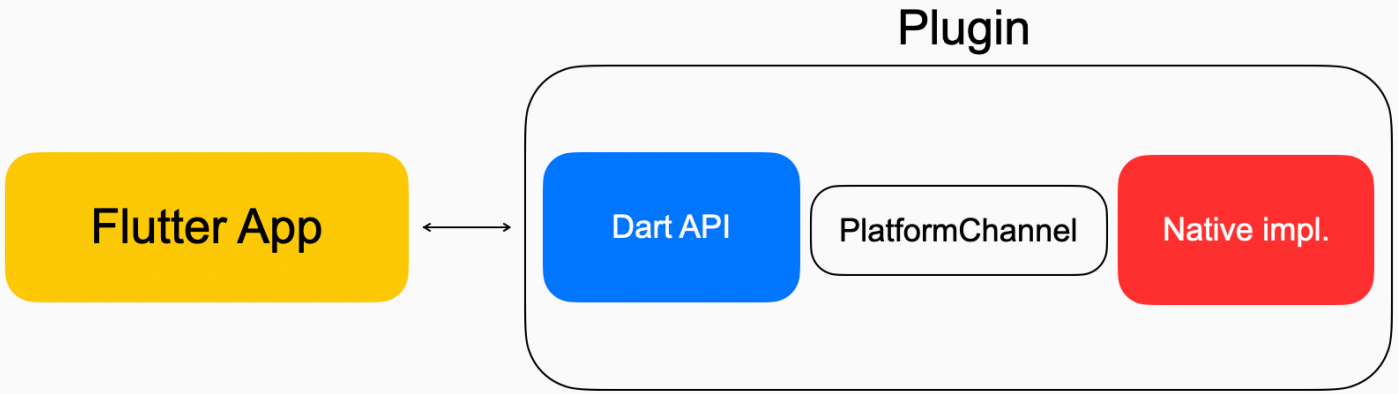
Flutter Plugin

- Описанный на Dart’е **API**



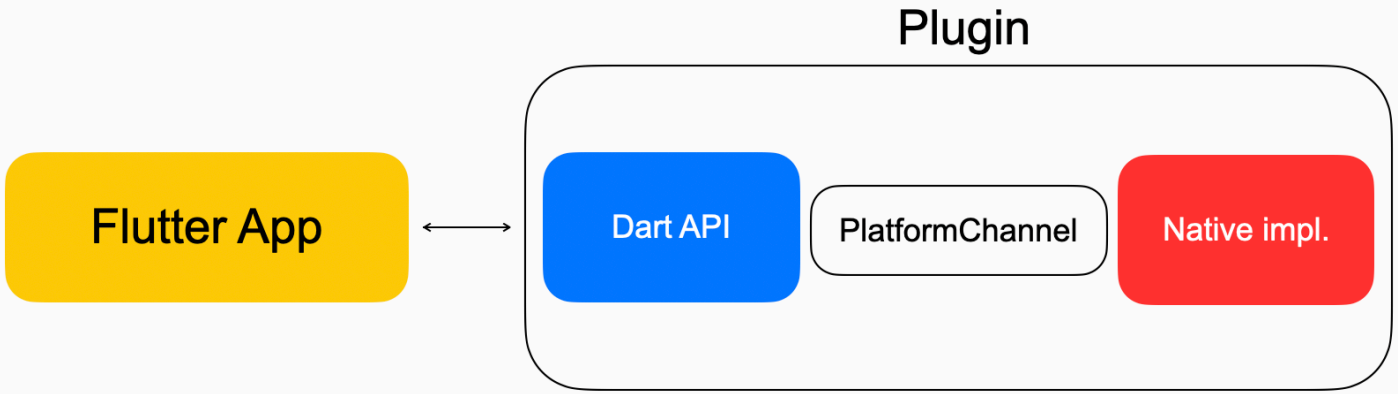
Flutter Plugin

- Описанный на Dart’е **API**
- Платформоспецифичная реализация на Objective-C/Swift, Java/Kotlin



Flutter Plugin

- Описанный на Dart’е **API**
- Платформоспецифичная реализация на Objective-C/Swift, Java/Kotlin
- PlatformChannel между ними




PlatformChannel

Определение

Механизм взаимодействия платформоспецифичного кода и Dart кода

PlatformChannel

- Двусторонний канал связи Dart <=> Native

PlatformChannel

- Двусторонний канал связи Dart <=> Native
- Вызовы асинхронны

PlatformChannel

- Двусторонний канал связи Dart <=> Native
- Вызовы асинхронны
- Каждый канал должен иметь уникальный идентификатор

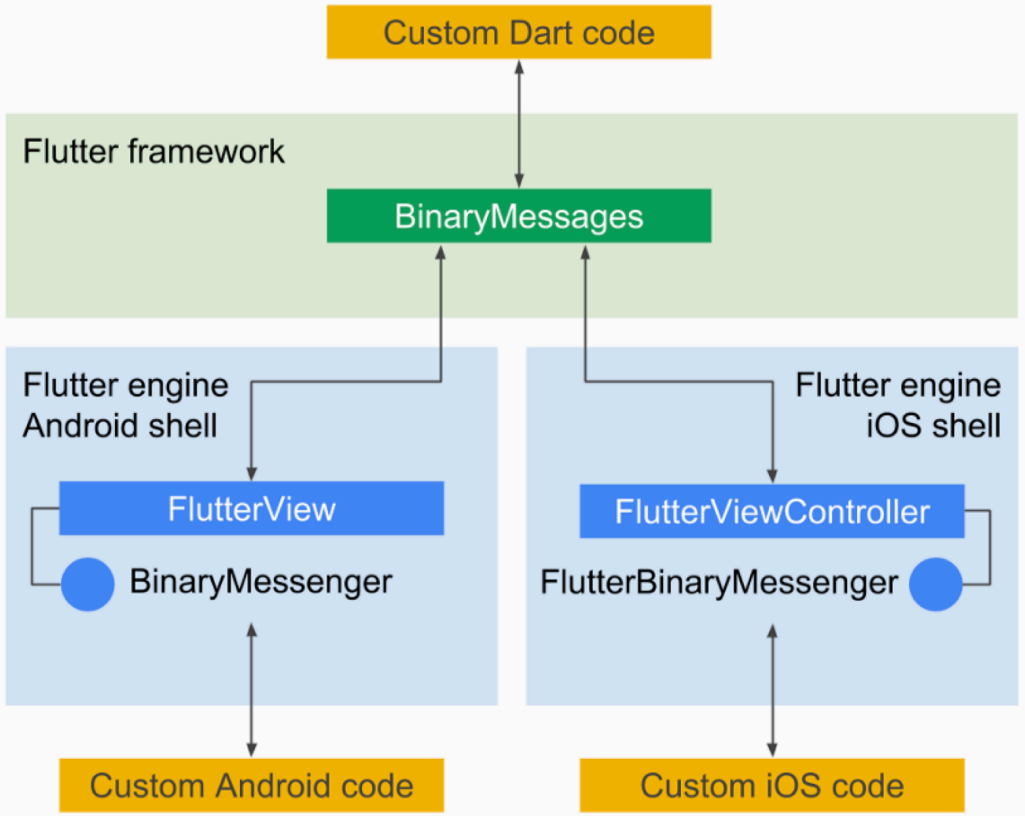
PlatformChannel





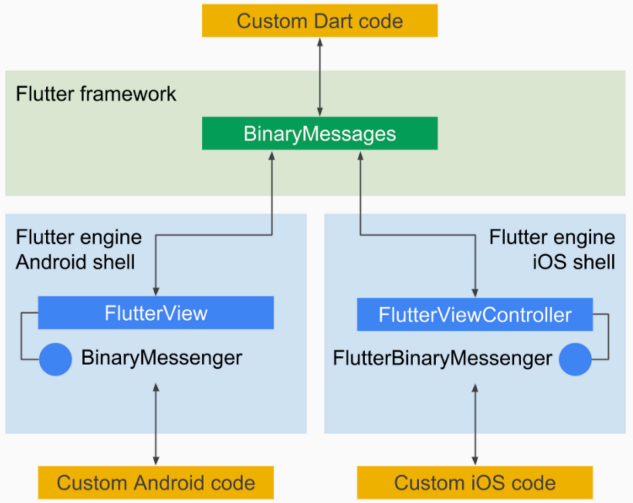
Типы PlatformChannel

BinaryMessenger



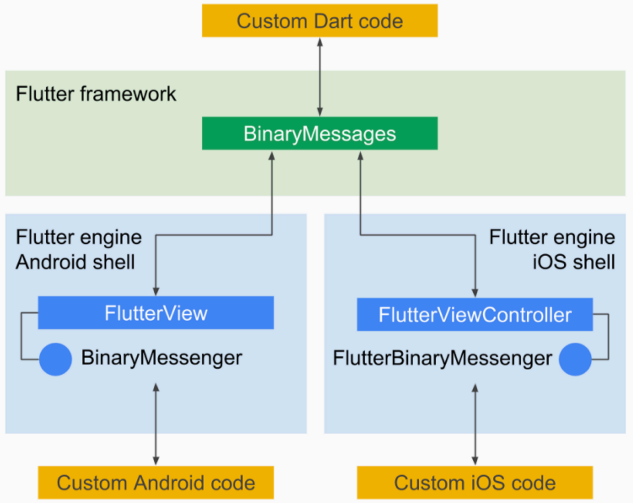
BinaryMessenger

- Сердце любого канала



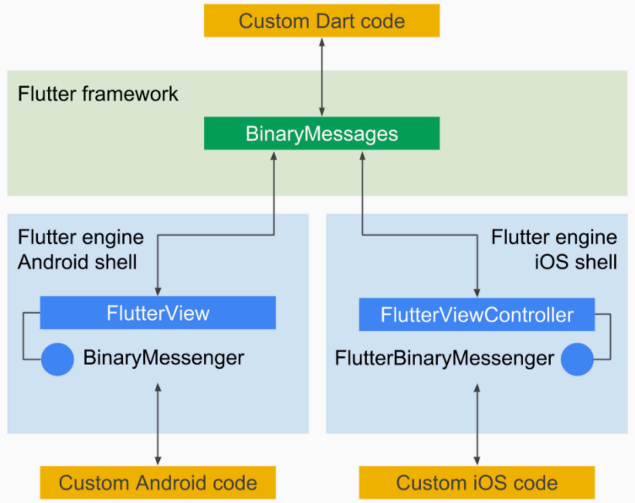
BinaryMessenger

- Сердце любого канала
- Не используется сам по себе



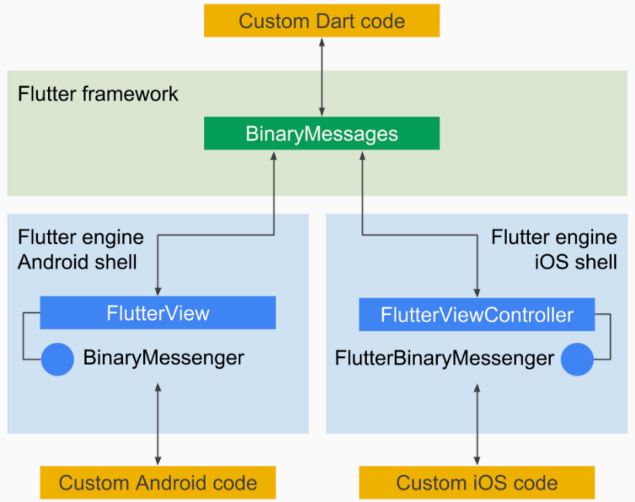
BinaryMessenger

- Сердце любого канала
- Не используется сам по себе
- Byte Buffer payload



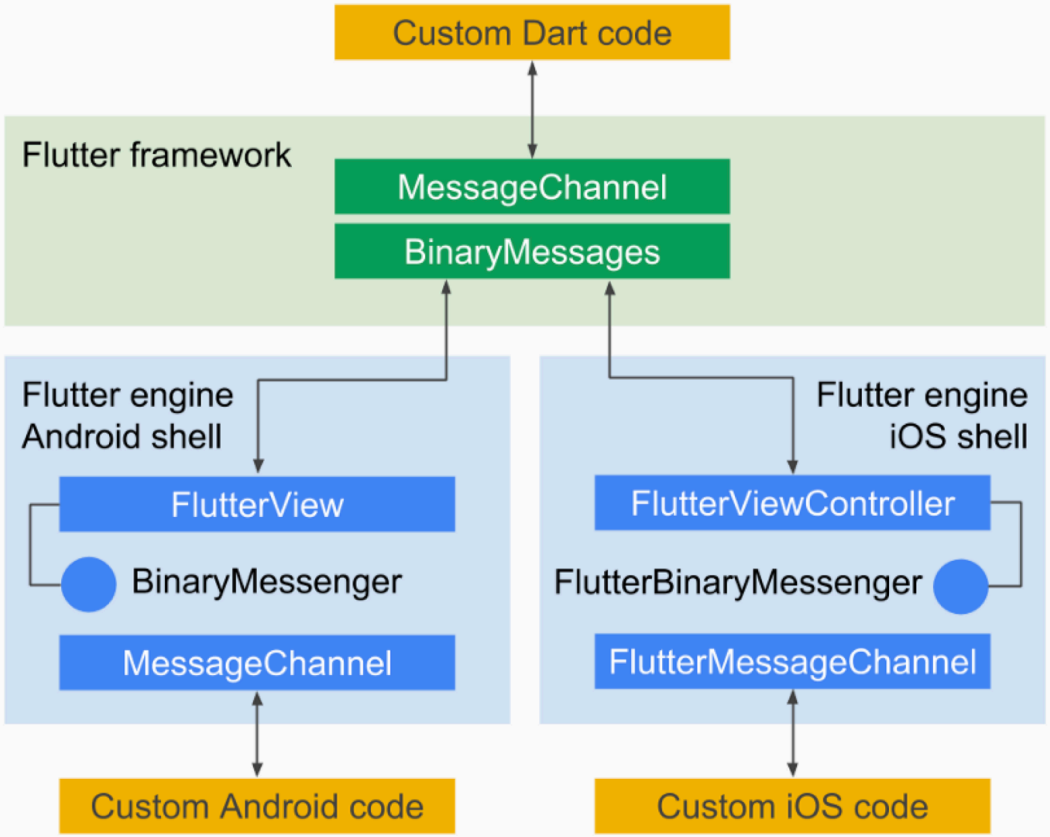
BinaryMessenger

- Сердце любого канала
- Не используется сам по себе
- Byte Buffer payload
- Обязательный ответ (хотя бы null)



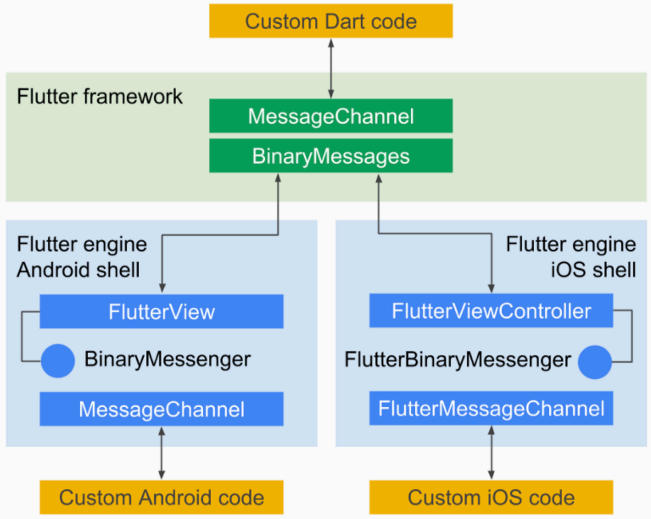

```dart
final WriteBuffer buffer = WriteBuffer()
  ..putFloat64(3.1415);
final ByteData message = buffer.done();
await BinaryMessages.send('channel_name', message);
print('Message sent, reply ignored');
```

BasicMessageChannel



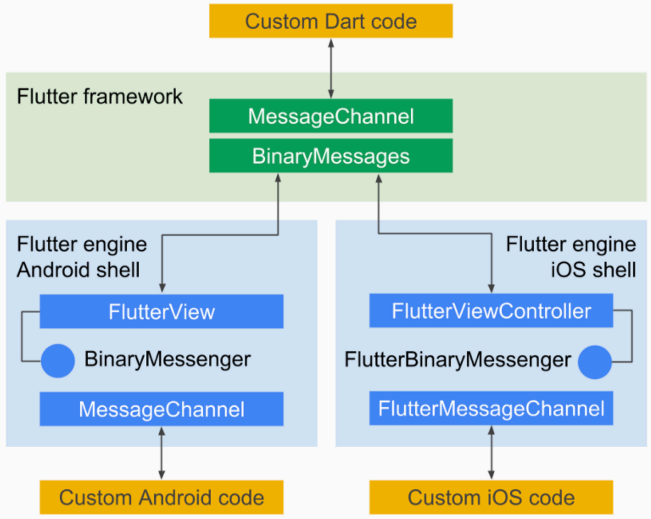
BasicMessageChannel

- Под капотом BinaryMessenger



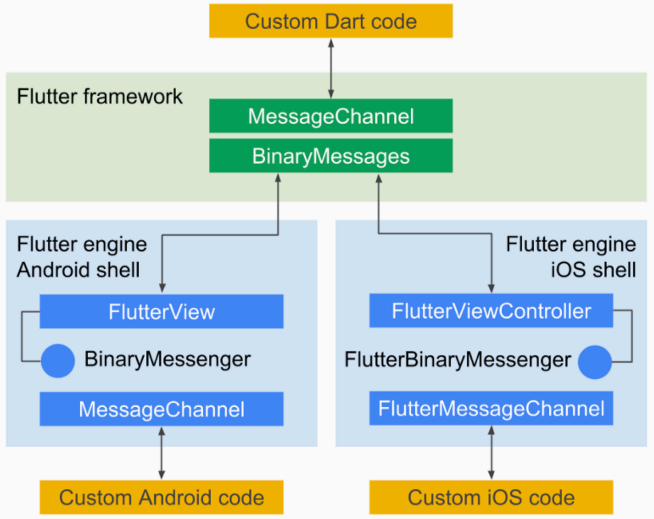
BasicMessageChannel

- Под капотом BinaryMessenger
- Умеют в базовые типы при помощи codec’ов



BasicMessageChannel

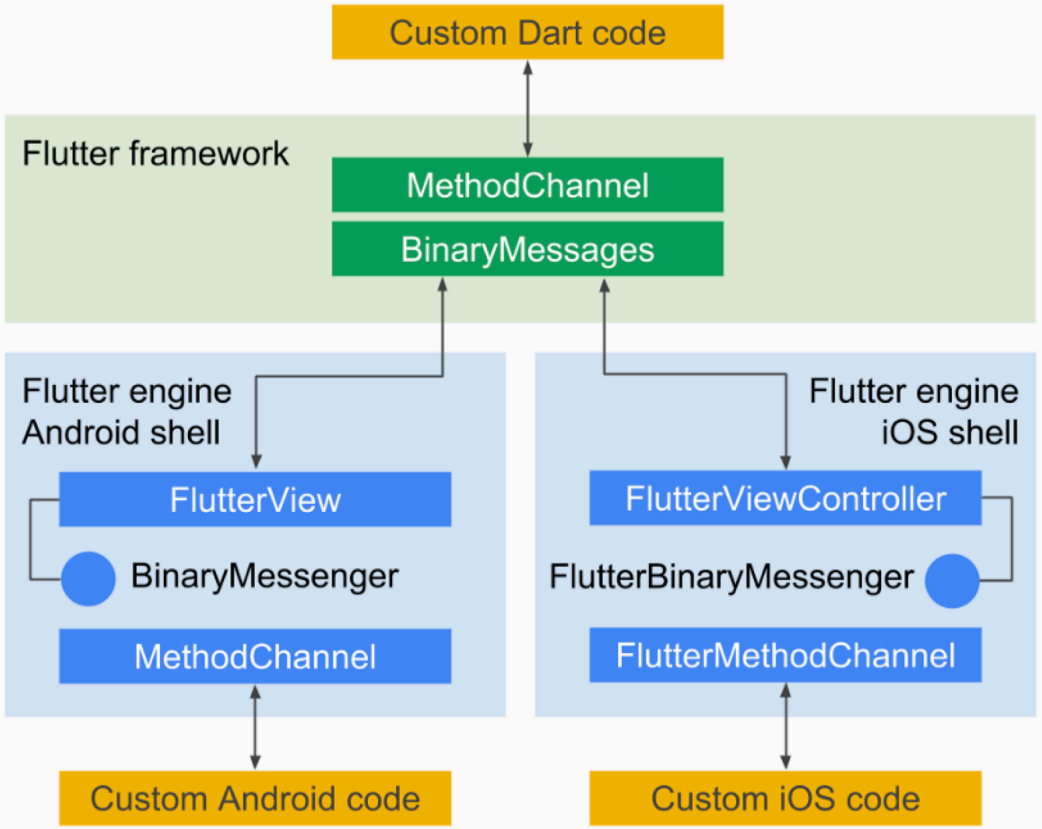
- Под капотом BinaryMessenger
- Умеют в базовые типы при помощи codec’ов




```dart
const channel = BasicMessageChannel<String>('channel_name', StringCodec());

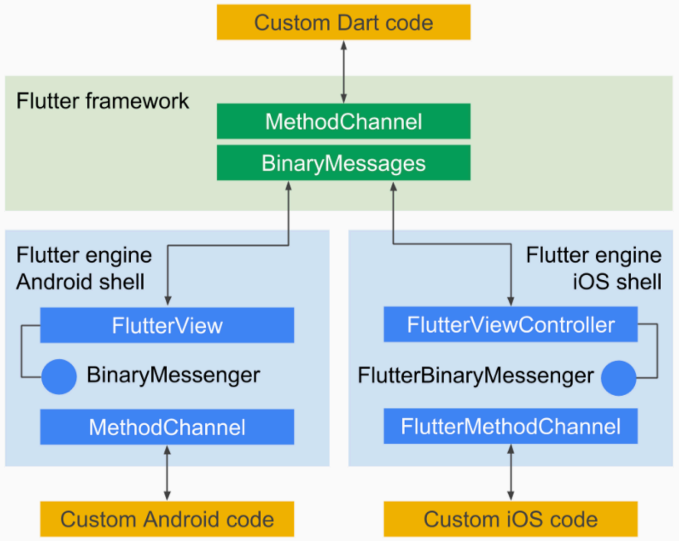
final String reply = await channel.send('Hello, world');
```

MethodChannel



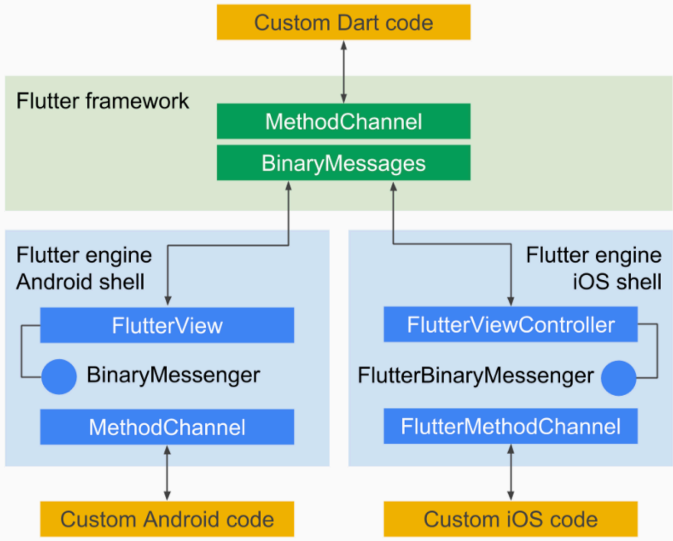
MethodChannel

- Всё так же BinaryMessenger



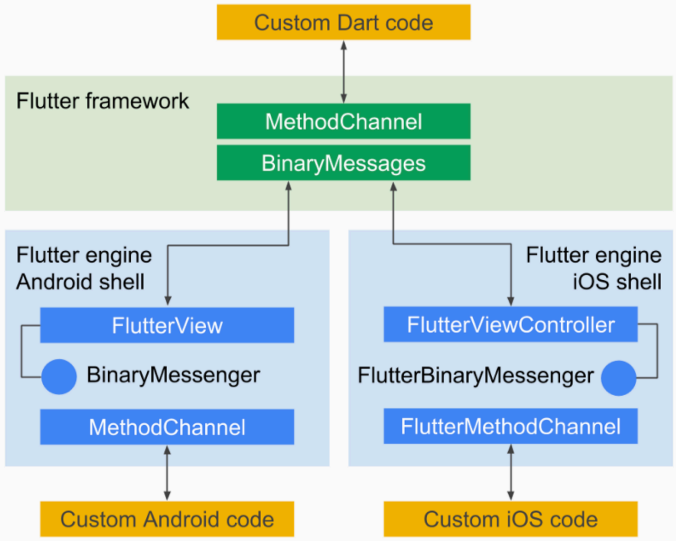
MethodChannel

- Всё так же BinaryMessenger
- Использует codec StandardMethodCodec — binary encoding



MethodChannel

- Всё так же BinaryMessenger
- Использует codec StandardMethodCodec — binary encoding




```
const channel = MethodChannel('channel_name');
final String greeting = await channel.invokeMethod('getHelloWorld', 'Hello,
World');
```


LIVE

MethodCodec — типы

- StandardMethodCodec — стандартный механизм, использует binary encoding (двоичное кодирование)

MethodCodec — типы

- StandardMethodCodec — стандартный механизм, использует binary encoding (двоичное кодирование)
  - Пишет в буфер отдельно имя вызываемого метода и аргументы, используя StandardMessageCodec

MethodCodec — типы

- StandardMethodCodec — стандартный механизм, использует binary encoding (двоичное кодирование)
  - Пишет в буфер отдельно имя вызываемого метода и аргументы, используя StandardMessageCodec
  - Ответ содержит в самом начале один байт: 0 или 1, определяя успешный вызов или ошибку
    - При успехе ответ декодируется также через StandardMessageCodec
    - При ошибке декодируется ошибка: код, сообщение и детали, всё — строки

StandardMessageCodec

Dart	Java	Kotlin	Obj-C	Swift
null	null	null	nil (NSNull when nested)	nil
bool	java.lang.Boolean	Boolean	NSNumber numberWithBool:	NSNumber(value: Bool)
int	java.lang.Integer	Int	NSNumber numberWithInt:	NSNumber(value: Int32)
int, if 32 bits not enough	java.lang.Long	Long	NSNumber numberWithLong:	NSNumber(value: Int)
double	java.lang.Double	Double	NSNumber numberWithDouble:	NSNumber(value: Double)
String	java.lang.String	String	NSString	String
Uint8List	byte[]	ByteArray	FlutterStandardTypedData typedDataWithBytes:	FlutterStandardTypedData(bytes: Data)
Int32List	int[]	IntArray	FlutterStandardTypedData typedDataWithInt32:	FlutterStandardTypedData(int32: Data)
Int64List	long[]	LongArray	FlutterStandardTypedData typedDataWithInt64:	FlutterStandardTypedData(int64: Data)
Float64List	double[]	DoubleArray	FlutterStandardTypedData typedDataWithFloat64:	FlutterStandardTypedData(float64: Data)
List	java.util.ArrayList	List	NSArray	Array
Map	java.util.HashMap	HashMap	NSDictionary	Dictionary

MethodCodec — типы

- JSONMethodCodec — UTF-8 encoded JSON.

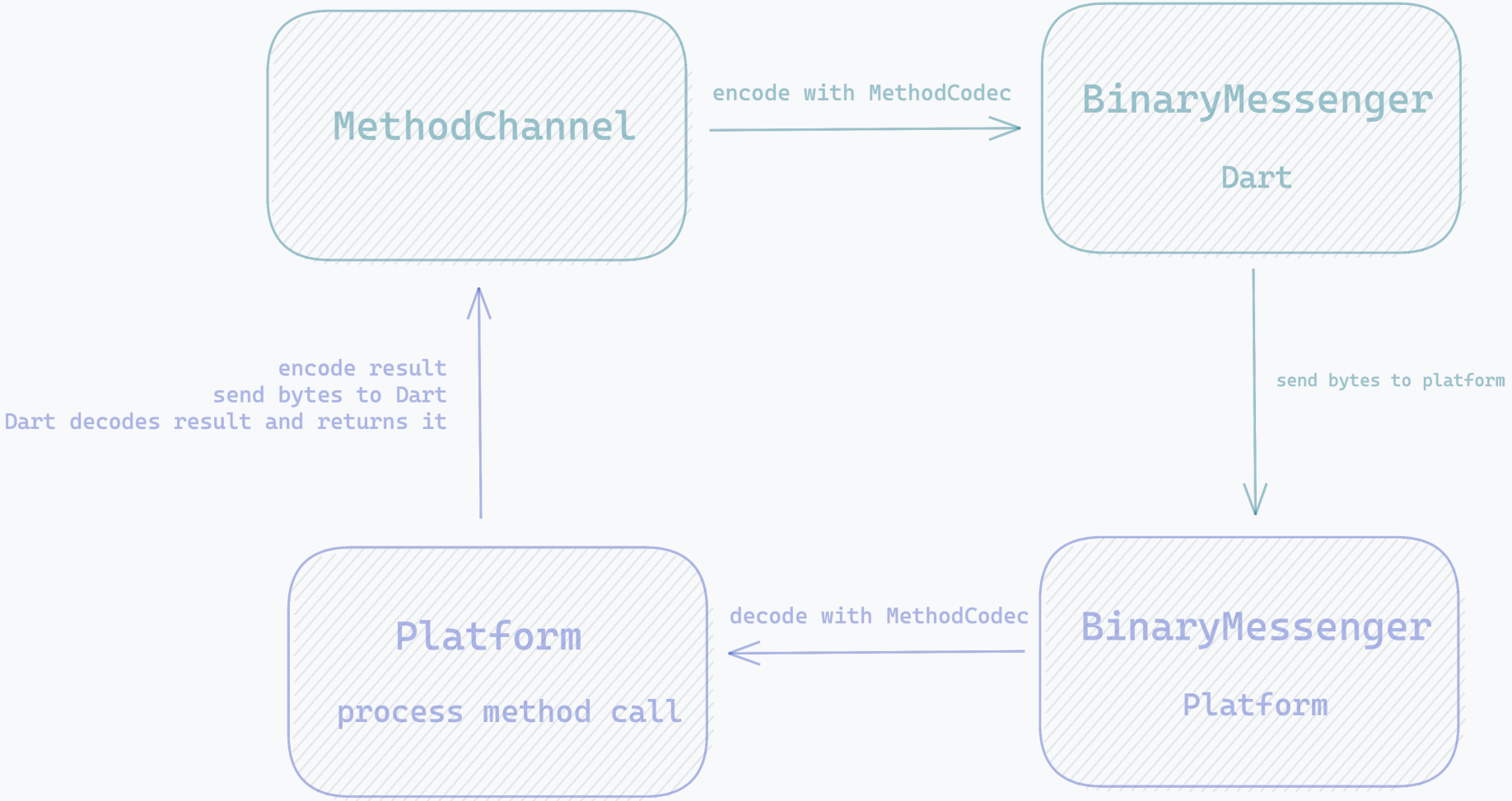
MethodCodec — типы

- JSONMethodCodec — UTF-8 encoded JSON.
  - Можем кодировать всё, что поддерживает JSON

MethodCodec — типы

- JSONMethodCodec — UTF-8 encoded JSON.
  - Можем кодировать всё, что поддерживает JSON
  - Отдельно кладётся поле “method”, отдельно “args”

Путь сообщения



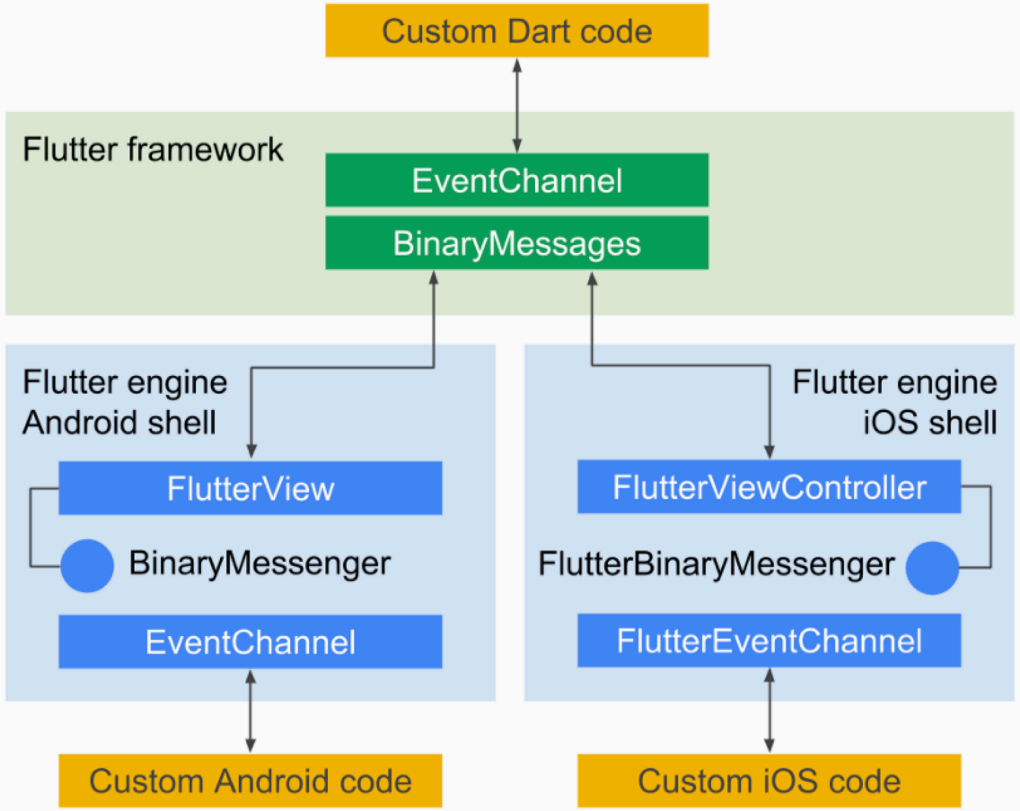
Практика

1. **Откройте** lesson_20_example **плагин**
2. **Реализуйте метод возведения в квадрат на стороне Android**



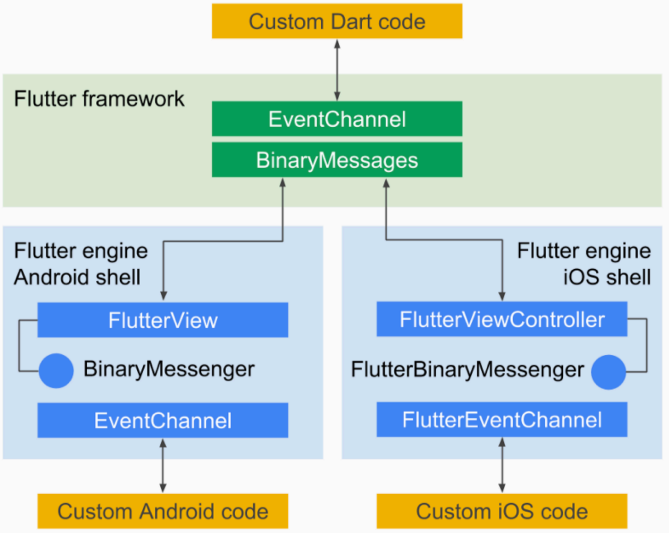
Тайминг: 10 min

EventChannel



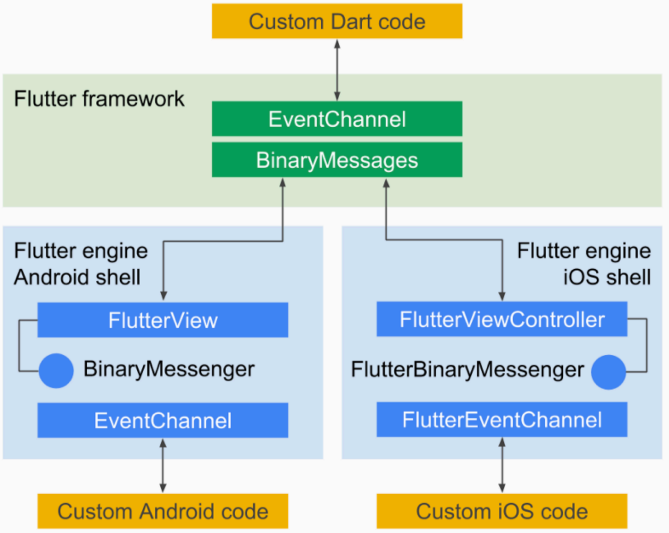
EventChannel

- Канал передачи потока событий из натива во Flutter



EventChannel

- Канал передачи потока событий из натива во Flutter
- Работает как обычный Dart Stream




LIVE

Quiz

1. Что является сердцем всех каналов? Как кодируются данные?
2. Какие типы данных можно передавать через MethodChannel?
3. Какой тип канала подходит для передачи потока данных?



Тайминг: 3 min

Маппинги руками

- Легко ошибиться в названиях полей

Маппинги руками

- Легко ошибиться в названиях полей
- Приходится дублировать много кода: Dart, iOS, Android

Маппинги руками

- Легко ошибиться в названиях полей
- Приходится дублировать много кода: Dart, iOS, Android
- На больших библиотеках с простой логикой написание кода превращается в monkey job

Pigeon

Pigeon

- Ещё один кодген в вашем арсенале

Pigeon

- Ещё один кодген в вашем арсенале
- Позволяет описать модели и API для работы с платформой один раз на Dart’е и сгенерить платформенный код

Pigeon

- Ещё один кодген в вашем арсенале
- Позволяет описать модели и API для работы с платформой один раз на Dart’е и сгенерить платформенный код
- Ускоряет написание платформенных каналов; исключает ошибки в нейминге


LIVE

Dart FFI x Flutter

- FFI — Foreign Function Interface

Dart FFI x Flutter

- FFI — Foreign Function Interface
- Позволяет вызывать нативный код (C/C++) прямо из Dart

Dart FFI x Flutter

- FFI — Foreign Function Interface
- Позволяет вызывать нативный код (C/C++) прямо из Dart
- Синхронные вызовы, интеграция с Flutter-приложениями


LIVE

Интеграция платформенных View

Интеграция платформенных View

- Платформенная View оборачивается во Flutter Widget и встраивается как обычный виджет в дерев

Интеграция платформенных View

- Платформенная View оборачивается во Flutter Widget и встраивается как обычный виджет в дерево
- Весь жизненный цикл View и её перерисовки при этом происходят на стороне платформы

Интеграция платформенных View

- Платформенная View оборачивается во Flutter Widget и встраивается как обычный виджет в дерево
- Весь жизненный цикл View и её перерисовки при этом происходят на стороне платформы
- View может занимать любую часть экрана, одновременно их может быть несколько


LIVE



Переиспользование нативного коа

Переиспользование нативного кода

- Функциональность может пригодиться в разных приложениях

Переиспользование нативного кода

- Функциональность может пригодиться в разных приложениях
- Отделение логики в нативе от логики во Flutter’е

Переиспользование нативного кода

- Функциональность может пригодиться в разных приложениях
- Отделение логики в нативе от логики во Flutter’е
- Вы придумали что-то классное и требующее платформенного кода и хотите поделиться этим с коммьюнити


LIVE

Хороший плагин

Хороший плагин

- Сначала функциональность, потом API

```dart
class StoragePlugin {

  /// Читаем строку
  Future<String> getString(String key) async {}

  /// Записываем строку
  Future<void> setString(String key, String value) async {}
}
```

Хороший плагин

- Вызовы платформоспецифичных методов должны обрабатываться универсальными методами в API плагина

```
if (Platform.isIos) {
  myPlugin.doIOSThing();
} else if (Platform.isAndroid) {
  myPlugin.doAndroidThing();
}
```

```
myPlugin.doThing();
```

Хороший плагин

- Избегайте специфичной для одной платформы функциональности

Цели вебинара | После занятия вы сможете

1. Работать с платформенным кодом в кроссплатформенной среде
2. Писать собственные плагины и переиспользовать нативный код во Флаттер приложениях
3. Глубже понимать устройство фреймворка и возможности Флаттер приложений

Что почитать

- Про Platform Channels, документация
- Дока про создание плагина
- Большая статья про Platform Channels и их устройство под капотом
- Создание платформенной вью во Флаттере раз,
  Создание платформенной вью два
- Pigeon
- Документация по работе с FFI

Домашнее задание

- Артефакт: подключённый к проекту плагин для вызова platform's share dialog и выполнения sharing для экрана описания коктейля.

Домашнее задание

- Склонировать соответствующий github репозиторий с заготовкой проекта для этого урока
- Подключить соответствующий плагин для вызова platform's share dialog в pubspec.yaml
- Открыть класс экрана CocktailDetailsScreen
- Внести изменения в код обработчика событий на нажатие кнопки Share (Поделиться)
- В обработчике события на Sharing сформировать текст с ссылкой на ImageThumb для описания коктейля
- В обработчике события на Sharing выполнить обращение к классу плагина и пошарить описание коктейля
- Убедиться, что сформированный текст уходит в канал #random в нашем канале Slack

Домашнее задание

- На ваше усмотрение:
  - Можно использовать любые проверенные плагины из pub.dev
  - Можно выполнить любую декомпозицию методов экрана CocktailDetailsScreen или кнопки Share для внесения изменений, требуемых для sharing
- Форма сдачи:
  - ДЗ Сдается в виде ссылки на github репозиторий с проектом
  - Приложить ссылку из канала #random в slack с пошаренным текстом
  - На ваше усмотрение : приложить к readme.md скриншот состояния sharing из экрана CocktailDetailsScreen

Спасибо за внимание!



Вирник Илья

Flutter Team Lead

Яндекс.Такси

Telegram: @virnik