Яндекс

Event Loop.
Platform Channels

Илья Вирник, Flutter Team Lead

Зачем нам это всё

〉 Выполнение асинхронного кода, а также многопоточное программирование в Dart’е радикально отличается от Swift/Kotlin

〉 Для разработки полноценных программ необходимо уметь работать с асинхронным и многопоточным кодом

Зачем нам это всё

〉 Flutter сам по себе — UI-фреймворк. Многие возможности операционных систем и смартфонов ему недоступны

〉 Для разработки сложных приложений почти всегда необходимо взаимодействовать с платформой, для этого во Flutter есть специальные механизмы

О чём будем говорить

00 | Future и что он из себя представляет

01 | Как устроен Dart Event Loop

02 | Что такое PlatformChannel

03 | Какими бывают PlatformChannel’ы

00

Future и что он из себя представляет

А также что же там внутри особенного

Что такое Future

Класс Future — generic обёртка над результатом выполнения асинхронной операции

У Future есть 3 состояния:

〉 Uncompleted — незавершённое, операция ещё не запущена или в процессе выполнения

〉 Completed with result — операция завершена успешно

〉 Completed with error — операция завершена с ошибкой

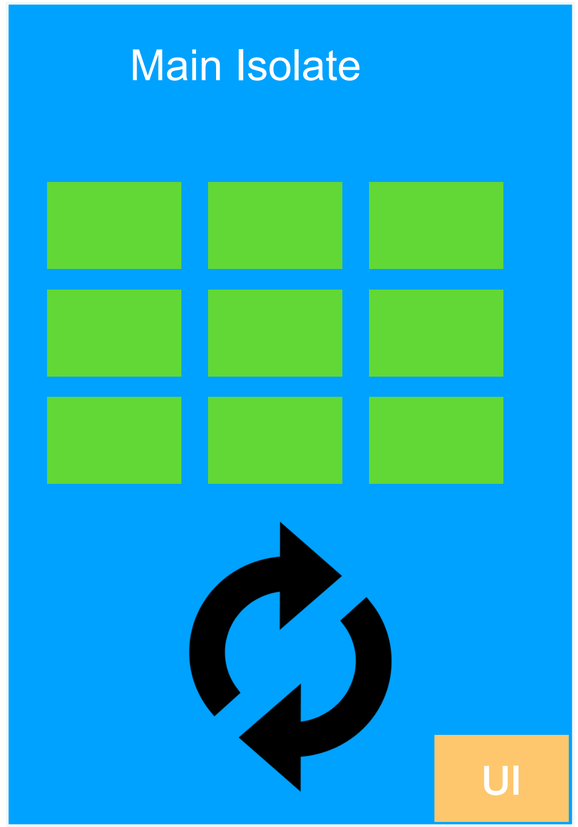
01

Как устроен Dart Event Loop

И что это вообще за зверь такой

Как устроен Dart Event Loop

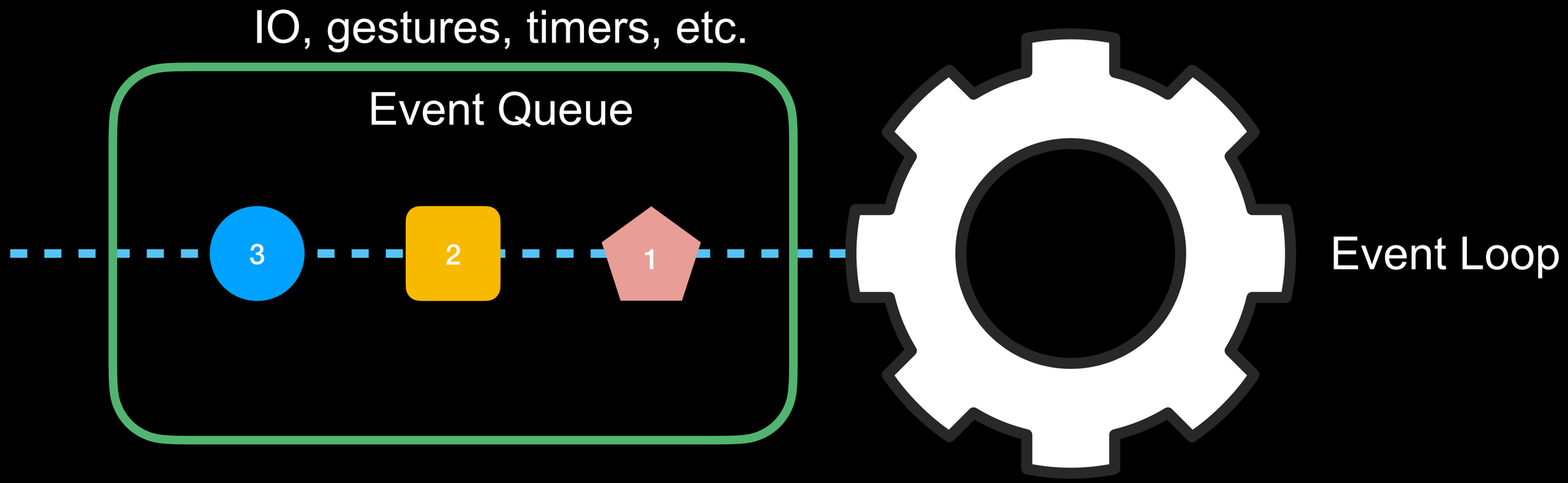
- Dart — однопоточный язык
- Главный поток в Dart’е — он же main isolate — поток, на котором выполняются все задачи, синхронные и асинхронные



Как устроен Dart Event Loop

- Каждый изолят содержит Event Loop — это вечный цикл, выполняющий все поступающие в изолят задачи
- Есть 2 FIFO очереди задач: обычные и microtasks
- Очередь microtasks имеет приоритет над обычными и опустошается первой
- Обычная очередь задач начинает выполнять задачи последовательно после выполнения microtasks

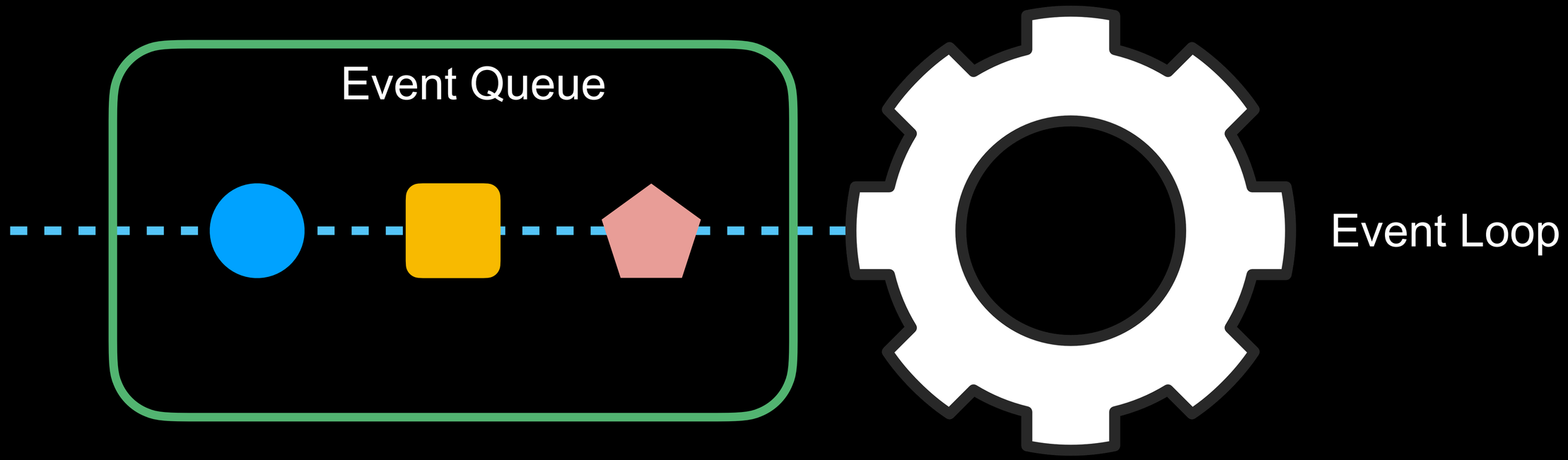
Как устроен Dart Event Loop



1.Жест на экране
2.Ввели символ
3.Запустили таймер

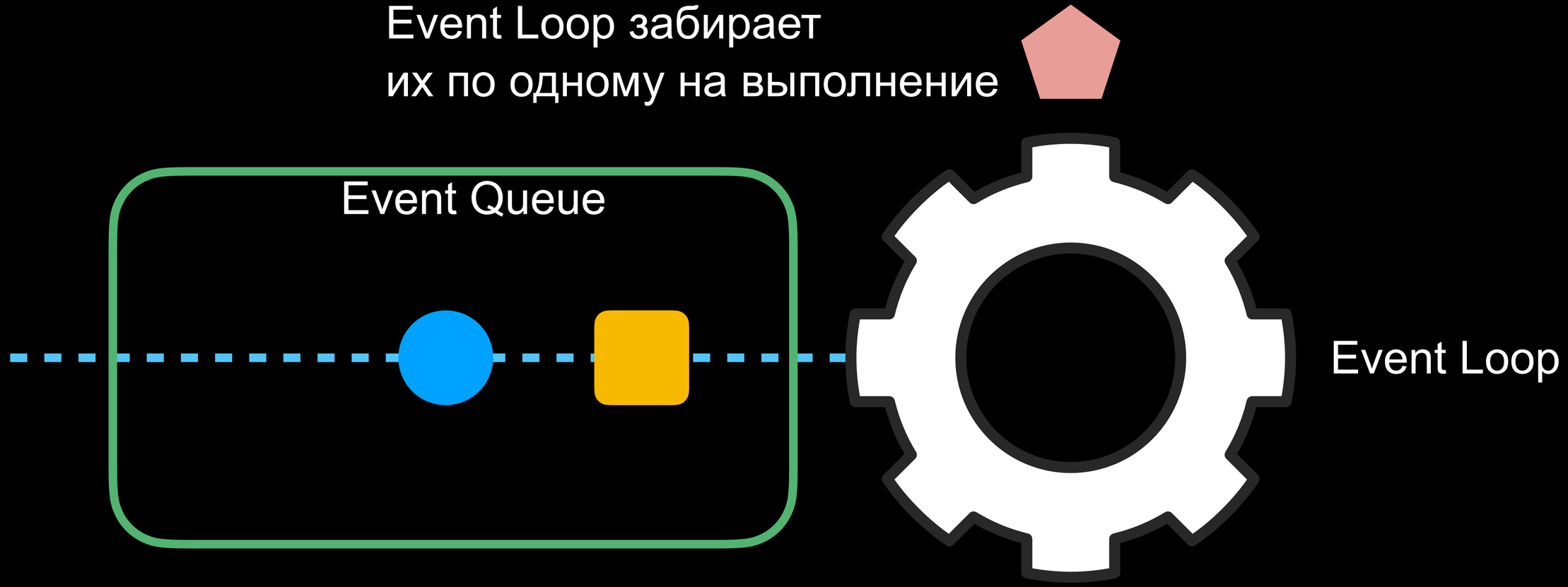
Как устроен Dart Event Loop

В очереди событий есть события



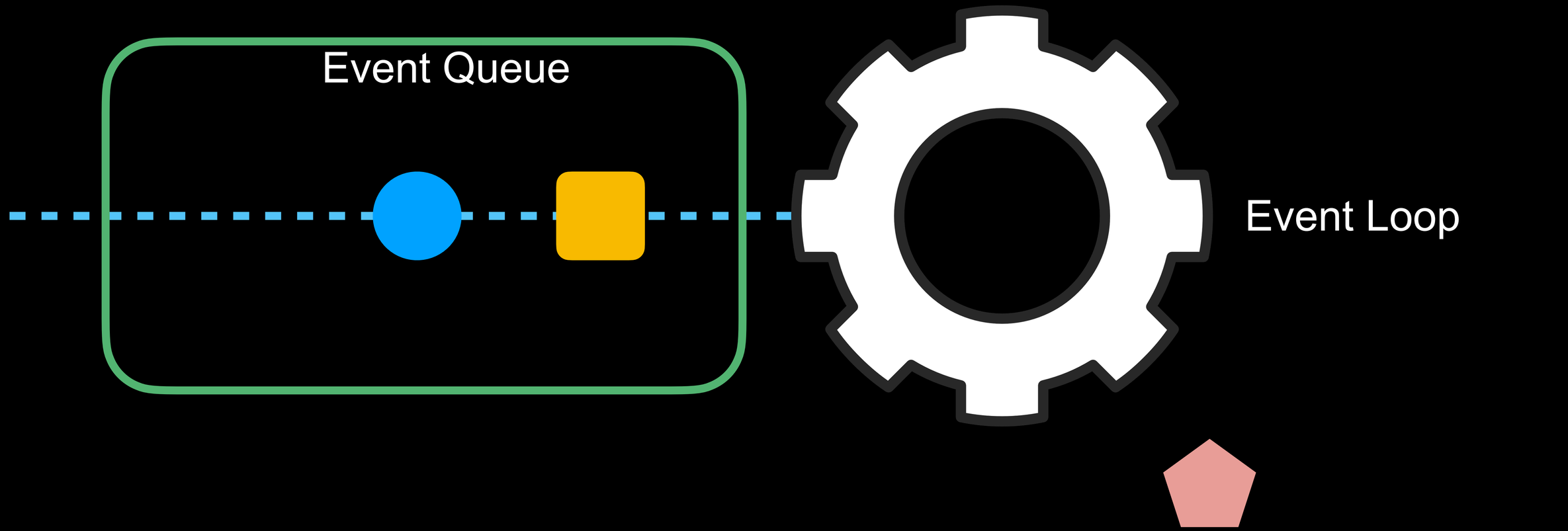
Как устроен Dart Event Loop

Event Loop забирает
их по одному на выполнение

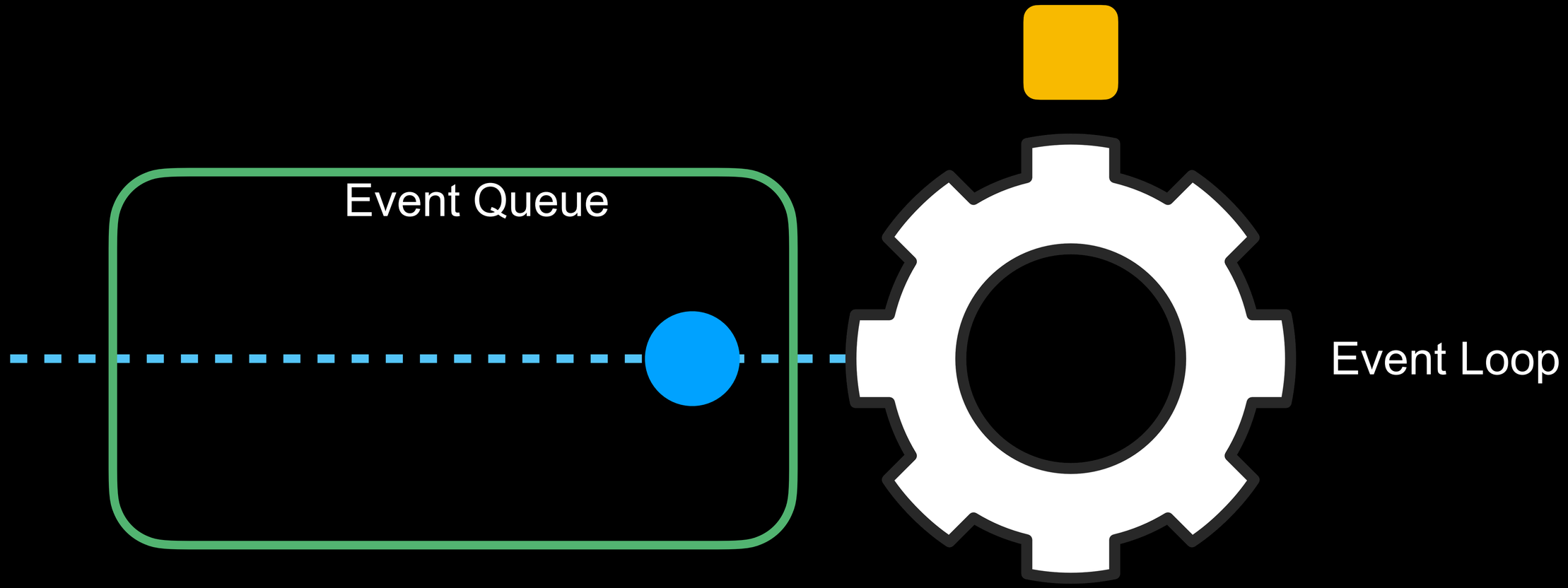


Как устроен Dart Event Loop

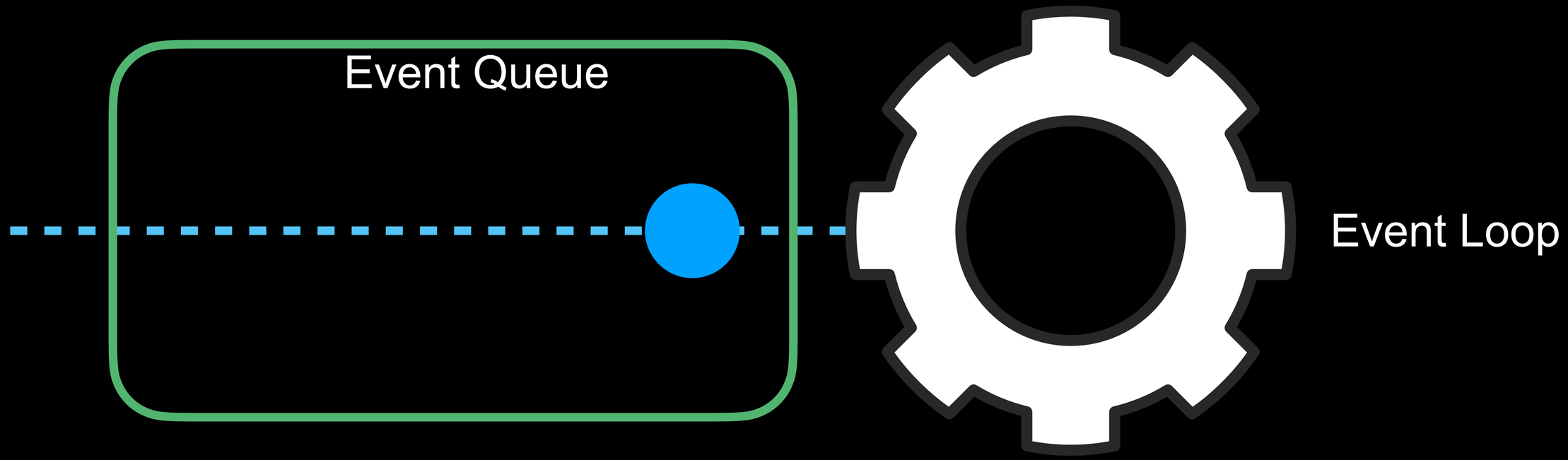
После завершения переходит к следующему



Как устроен Dart Event Loop

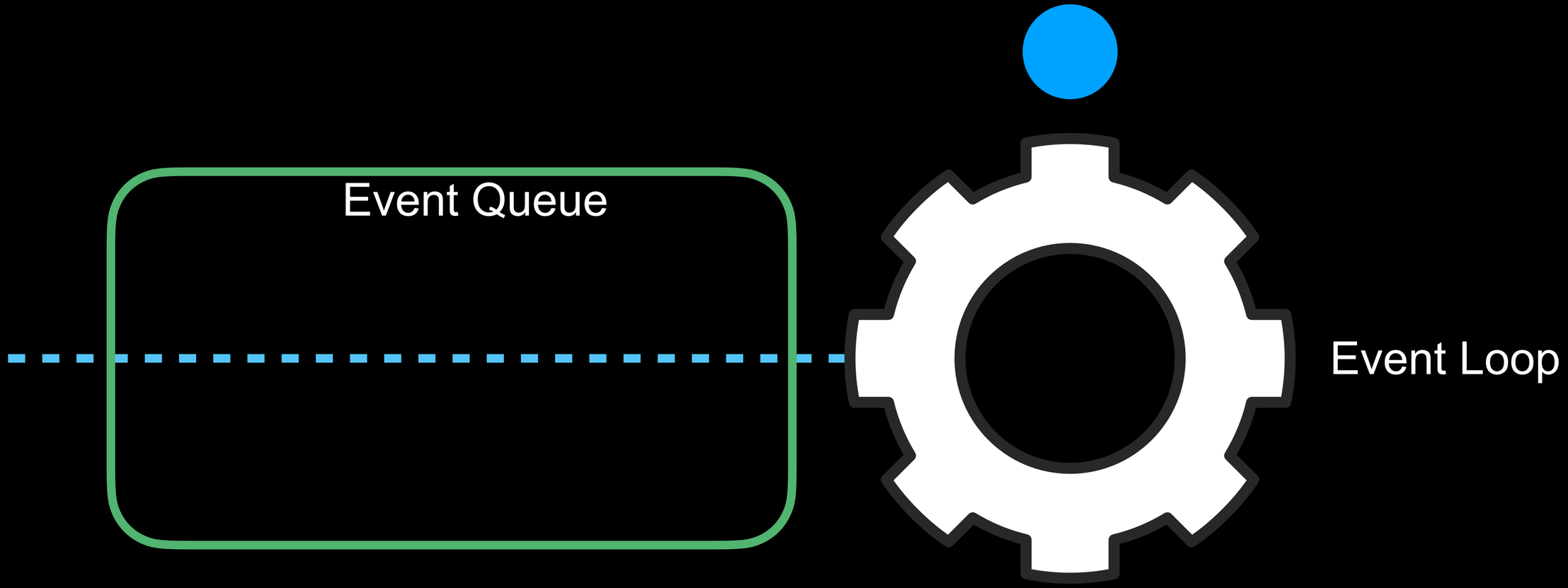


Как устроен Dart Event Loop

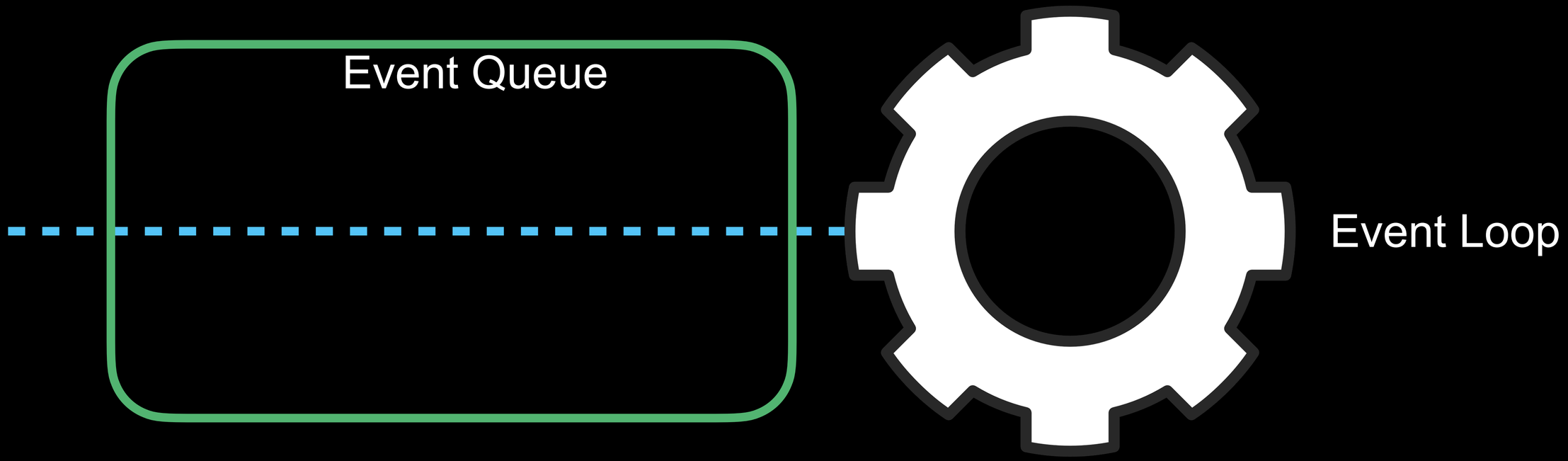


Задание выполнено

Как устроен Dart Event Loop



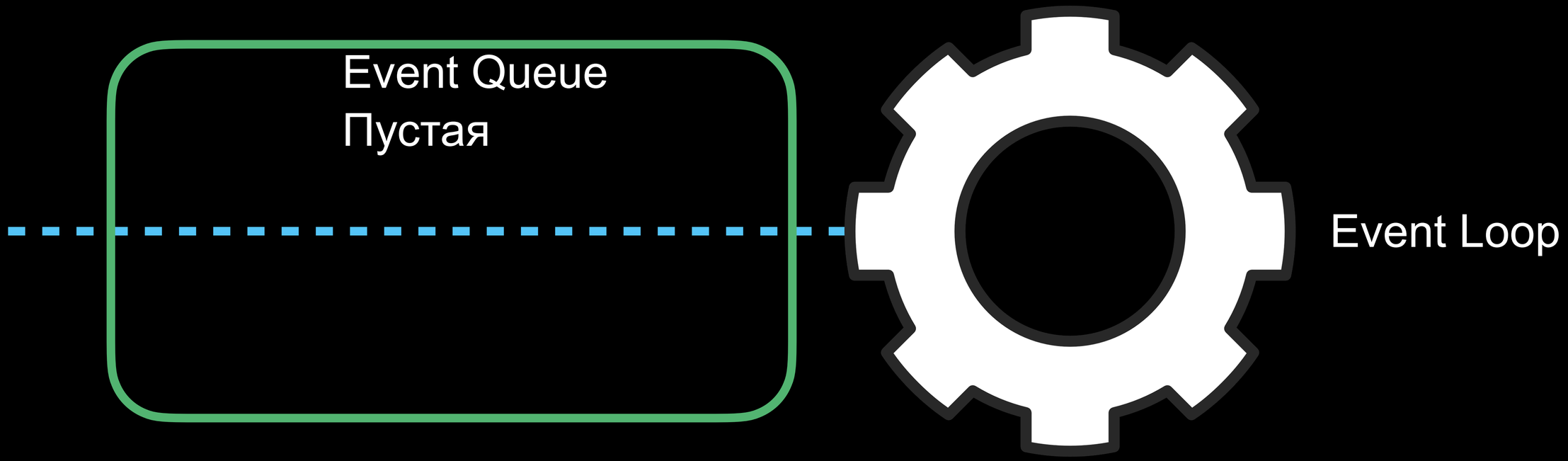
Как устроен Dart Event Loop



Задание выполнено

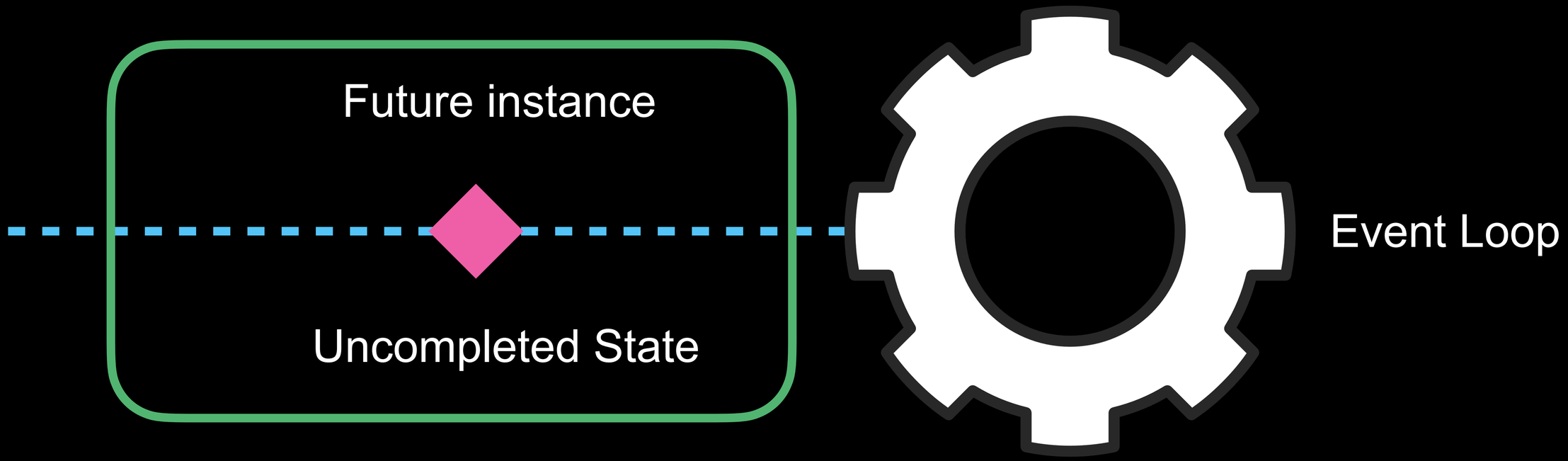
Как устроен Dart Event Loop

Ожидаем следующие события

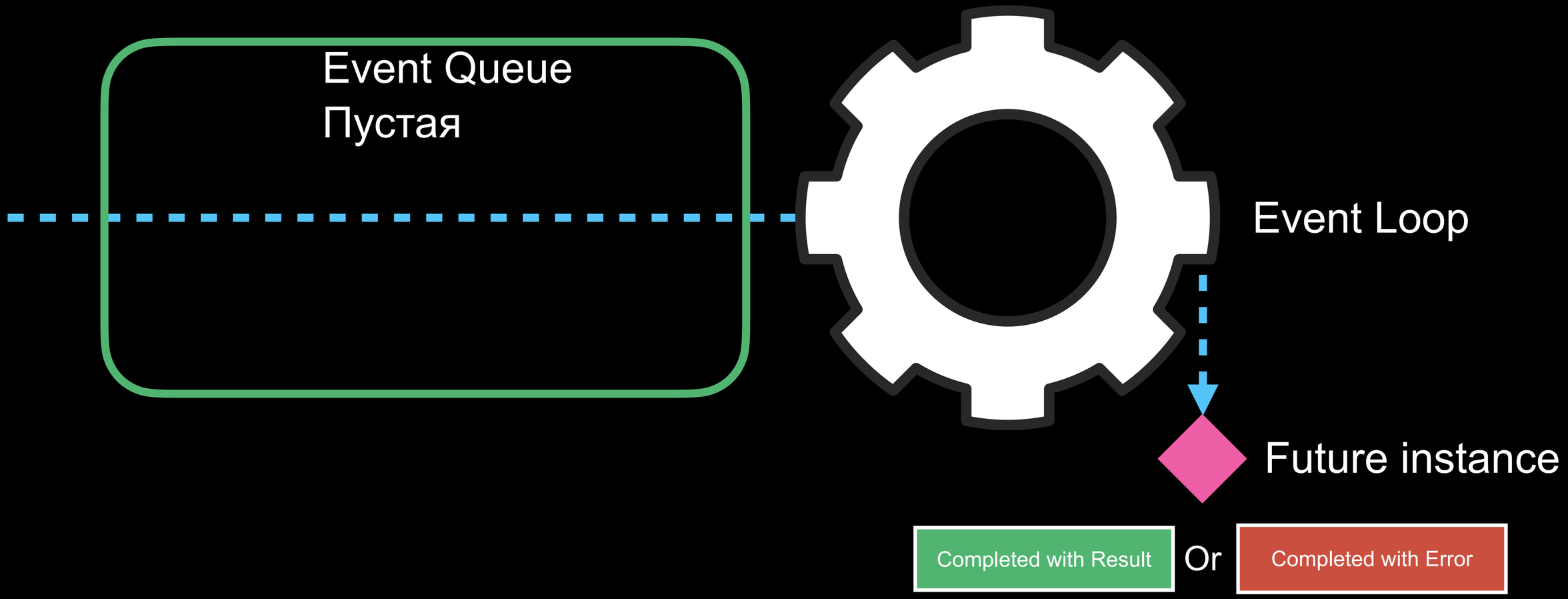


А что с Future?

Как устроен Dart Event Loop



Как устроен Dart Event Loop



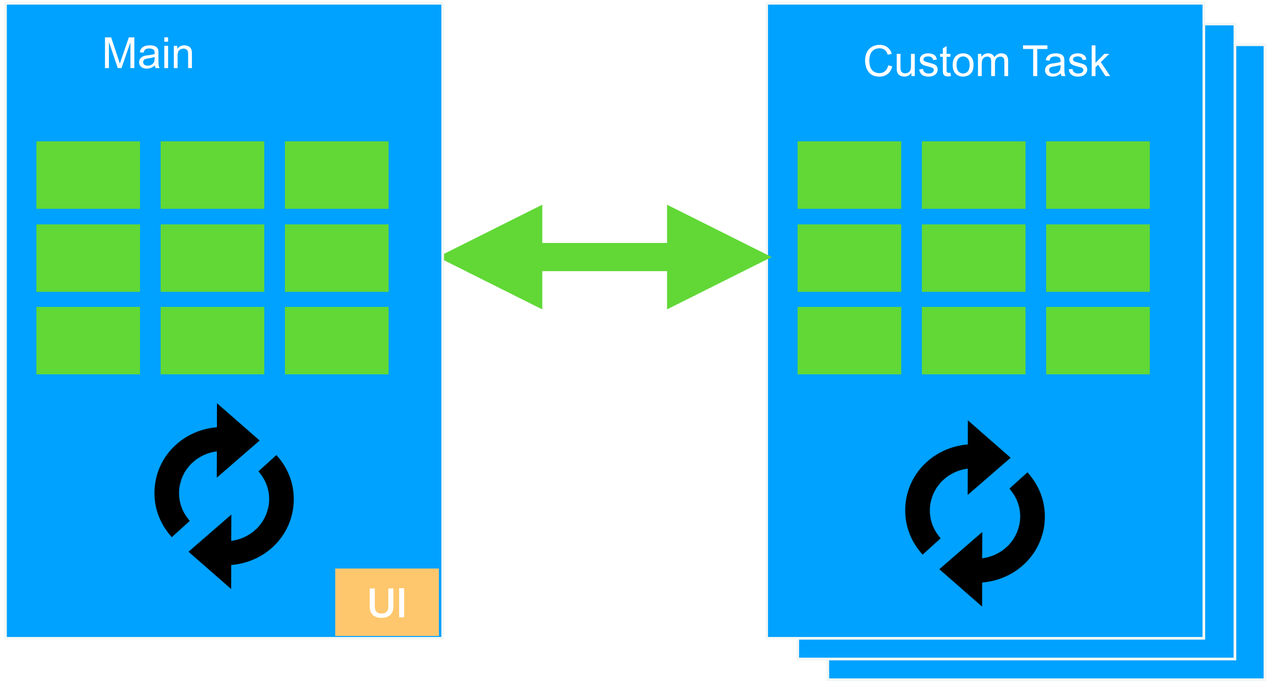
Что такое Future

- Пример синхронного и асинхронного выполнения операций

〉Источник примера и просто хорошая статья

Как устроен Dart Event Loop

- Dart — многопоточный язык
- В общем случае в приложении на Dart может быть несколько изолятов



Как устроен Dart Event Loop

- Isolate/изолят — поток в Dart’е
- Каждый изолят имеет выделенную память, которая не делится с другими изолятами
- Каждый изолят имеет собственный Event Loop
- Общение между изолятами организуется через сообщения. Данные каждый раз глубоко копируются, поскольку память у каждого своя
- Создание изолята — достаточно дорогая операция, не стоит этим злоупотреблять

Как там дела с изолятами

Как устроен Dart Event Loop

- Почитать дома

03

Что такое PlatformChannel

И с чем его едят

Механизм взаимодействия платформоспецифичного кода и Dart кода

Platform Channel

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

04

Какими бывают PlatformChannel’ы

И как с ними работать

BinaryMessenger

- 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

- Под капотом BinaryMessenger

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

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

BasicMessageChannel

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

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

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

MethodChannel

> Всё так же BinaryMessenger

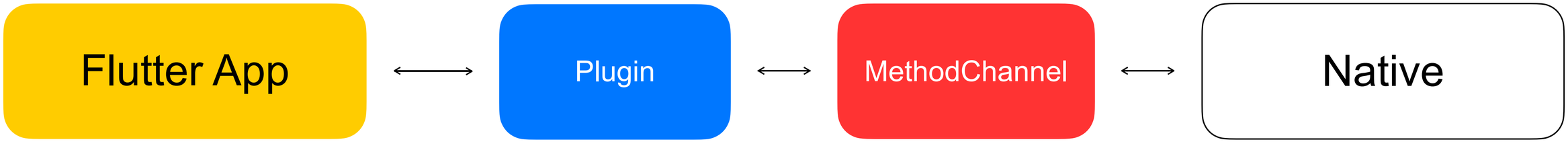
```
const channel = MethodChannel('channel_name');
final String greeting = await channel.invokeMethod('getHelloWorld', 'Hello,
World');
```

MethodChannel

- Всё так же BinaryMessenger
- Тоже использует codec MethodCall

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

MethodChannel



Вызываем через MethodChannel

Event Channel

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

Стримим в EventChannel

А если хочу встроить вьюху?

Platform 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

Встраиваем вьюхи

Вместо итога

Что почитать

- Про Dart Event Loop и Isolates
- 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

Яндекс

Q&A

Илья Вирник

i-virnik@yandex-team.ru