Яндекс Такси

5. Асинхронность

Кроссплатформенная разработка мобильных
приложений на Flutter

Дмитрий Тараев, разработчик

Асинхронность vs. Синхронность

Dart – однопоточный язык

- (выполнение одной инструкции не прерывается другой)

- Поэтому нужно постараться, чтобы выполнение кода не мешало плавности работы приложения

Как всё происходит в Dart 1/2

При запуске приложения:

- Создаётся и запускается новый процесс (thread, isolate) – единственный процесс в приложении
- Инициализируют очереди MicroTask и Event
- Выполняется main()
- Запускается Event Loop (цикл событий)

Цикл событий (event loop)

Процесс, который управляет порядком исполнения кода.

“Бесконечный цикл”

- Если нет кода, то берёт операции сначала из MicroTask, потом из Event

Как всё происходит в Dart 2/2

MicroTask

- Очень короткие действия

Event

- События (операции ввода/вывода, жесты, таймеры, потоки…)
- Future

- При срабатывании события код ставится на обработку в очередь Event.

Future

Асинхронная задача, которая завершается (успешно или неуспешно) когда-то в будущем

Пример: https://pastebin.com/W5CGPkHA

Future. Последовательность выполнения

1. экземпляр создаётся и хранится во внутреннем массиве, управляемом Dart
2. код, который должен быть исполнен данным экземпляром Future, добавляется напрямую в очередь Event
3. возвращается экземпляр Future со статусом не завершено (incomplete)
4. если есть синхронный код для исполнения, он исполняется (но не код Future)
5. Код, связанный с экземпляром Future будет исполнен как любой другой Event, как только Event Loop возьмёт его из очереди.
6. По завершении кода (успешно или с ошибкой) будет вызван метод then() или catchError() экземпляра Future.

Асинхронные методы (async)

- **Результат выполнения – Future**
- **Код исполняется синхронно до слова await. После этого приостанавливается выполнение до тех пор, пока не будет завершён связанный с ним Future.**
- **ВАЖНО! Добавлять await перед Future, иначе последовательность выполнения может быть неожиданной.**

Изоляты (isolates). Принципы работы

- Разновидность потока (thread)
- У каждого изолята свой цикл событий (и очереди MicroTask и Event)
- У изолятов нет общей памяти – обмениваются сообщениями
- Код в изоляте исполняется независимо друг от других изолятов
- Т.о. получается многозадачность

Изоляты (isolates). Взаимодействие

SendPort – порт у изолята, в который отсылается сообщение от другого

ReceivePort – у того, кто отправляет

- Есть поле sendPort, которое передаём в другой изолят.

Пример: https://pastebin.com/TsAVPgky

Изоляты (isolates). Способы создания

Низкоуровневое

- Isolate.spawn

Однократное

- compute, эта функция

1. Создаёт изолят
2. Исполняет на нём callback функцию
3. Возвращает значение
4. Удаляет изолят, завершает выполнение callback функции

Пример. https://flutter.dev/docs/cookbook/networking/background-parsing

Рекомендации по многопоточности

Если всё последовательно

- Обычный синхронный подход

Если части кода работают независимо не влияя на плавность

- Future

Тяжёлые вычисления (могут влиять на плавность работы)

- Изоляты
- Например, декодирование большого JSON’а, обработка изображений, загрузка изображений

Примеры и применение

- Работа с сетью
- Базы данных
- Платформенный код (Platform channel communication)
- …

Яндекс Такси

Спасибо

Дмитрий Тараев

Разработчик

 dtaraev@yandex-team.ru

 @dmitryta