08

메서드 채널, 이벤트 채널

네이티브 연동

메서드 채널 이용하기

다트에서 네이티브로 보내기

- 다트 코드
  - MethodChannel 생성자의 매개변수에 지정한 문자열은 채널 이름
  - invokeMethod() 함수로 네이티브를 실행
  - 첫 번째 매개변수는 메서드 이름
  - 두 번째 매개변수는 보낼 데이터

• 메서드 채널로 네이티브에 데이터 보내기(다트)

```
const channel = const MethodChannel('myMethodChannel');

try {
  var details = {'Username':'kkang','Password':'1234'};
  final Map result = await channel.invokeMethod("oneMethod", details);
} on PlatformException catch (e) {
  print("Failed: '${e.message}'.");
}
```

메서드 채널 이용하기

다트에서 네이티브로 보내기

- 안드로이드 코드
  - MethodChannel 객체로 매서드 채널을 만들며 생성자의 두 번째 매개변수에 채널 이름을 지정
  - setMethodCallHandler() 함수의 매개변수에 콜백 함수를 등록
  - 첫 번째 매개변수는 MethodCall 객체인데 다트에서 호출한 정보
  - 이 객체의 method 속성으로 다트에서 호출한 메서드 이름을 얻고, arguments 속성으로 다트에서 보낸 데이터를 받습니다.
  - 두 번째 매개변수는 Method Channel.Result 타입의 객체인데, 이 객체의 success() 함수를 호출하면 매개변수로 다트에 결괏값을 반환

• 다트에서 보낸 데이터 받기(코틀린)

```
val methodChannel = MethodChannel(flutterEngine.dartExecutor.binaryMessenger,
                                  "myMethodChannel")

methodChannel.setMethodCallHandler { call, result ->
  if (call.method == "oneMethod") {
    val map = call.arguments as Map<String, String>
    result.success(mapOf("one" to 10, "two" to 20))
  } else {
    result.notImplemented()
  }
}
```

메서드 채널 이용하기

다트에서 네이티브로 보내기

- iOS 코드
  - FlutterMethodChannel로 메서드 채널 객체를 만들며 name에 채널 이름을 지정
  - setMethodCallHandler() 함수의 매개변수에 콜백 함수를 지정
  - 첫 번째 매개변수는 다트에서 호출한 메서드 정보이며 메서드 이름과 보낸 데이터
  - 두 번째 매개변수로 전달된 FlutterResult 객체를 이용해 다트에 결과 데이터를 반환

• 다트에서 보낸 데이터 받기(스위프트)

```
let methodChannel = FlutterMethodChannel(
    name: "myMethodChannel",
    binaryMessenger: controller.binaryMessenger)

methodChannel.setMethodCallHandler({
  (call: FlutterMethodCall, result: @escaping FlutterResult) -> Void in
  swich(call.method) {
  case "oneMethod":
    let argument = call.arguments as? Dictionary<String, Any>
    let resultArg = ["one":30, "two":40]
    result(resultArg)
  default:
  break;
  }
})
```

메서드 채널 이용하기

네이티브에서 다트로 보내기

- 다트 코드
  - MethodChannel의 setMethodCallHandler() 함수를 이용
  - 매개변수에 네이티브에서 보낸 데이터를 받을 때 실행할 콜백 함수를 등록
  - 첫 번째 매개변수는 네이티브에서 호출한 메서드 정보
  - method 속성으로 메서드 이름을, arguments 속성으로 보낸 데이터를 얻을 수 있습니다.

• 네이티브에서 보낸 데이터 받기(다트)

```
const channel = const MethodChannel('myMethodChannel');
channel.setMethodCallHandler((call) async{
  switch(call.method){
    case 'twoMethod':

      ... (생략) ...
      return 'Reply from Dart';
  }
});
```

메서드 채널 이용하기

네이티브에서 다트로 보내기

- 안드로이드 코드
  - Method Channel로 메서드 채널을 만들고 이 객체의 invokeMethod() 함수를 호출
  - 번째 매개변수는 호출할 메서드 이름, 두 번째 매개변수는 보낼 데이터
  - 세 번째 매개변수는 다트가 반환하는 결과를 받는 콜백 객체

• 안드로이드에서 데이터 보내기(코틀린)

```
val methodChannel = MethodChannel(flutterEngine.dartExecutor.binaryMessenger,
                                  "myMethodChannel")
... (생략) ...
methodChannel.invokeMethod("twoMethod", "Hello from Android", object : MethodChannel.
Result {
  override fun success(result: Any?) {
    io.flutter.Log.d("flutter", "${result as String}")
  }

  override fun error(errorCode: String, errorMessage: String?, errorDetails: Any?) {
  }

  override fun notImplemented() {
  }
```

메서드 채널 이용하기

네이티브에서 다트로 보내기

- iOS 코드
  - FlutterMethod Channel로 채널을 만들며 이 채널 객체의 invokeMethod() 함수를 호출
  - 첫 번째 매개변수는 다트의 함수 이름이고, 두 번째 매개변수는 보낼 데이터
  - 세 번째 매개변수로 지정한 함수는 다트에서 결과를 반환할 때 호출할 함수

• iOS에서 데이터 보내기(코틀린)

```
let methodChannel = FlutterMethodChannel(
  name: "myMethodChannel",
  binaryMessenger: controller.binaryMessenger)
  ... (생략) ...

methodChannel.invokeMethod("twoMethod", arguments: "Hi from iOS") {
  (result: Any?) -> Void in
  if let error = result as? FlutterError {

  } else if FlutterMethodNotImplemented.isEqual(result) {

  } else {
    print("%@", result as! String)
  }
}
```

이벤트 채널 이용하기

- 이벤트 채널은 메시지나 메서드 채널과 다르게 네이티브에서 다트를 실행하는 방법으로만 사용

- 다트 코드
  - EventChannel로 채널 객체를 만들면서 생성자 매개변수에 이벤트 채널 이름을 지정
  - registerBroadcastStream() 함수를 호출하면 이 채널로 전달되는 데이터를 계속해서 받을 수 있는 Stream 객체가 반환
  - Stream의 listen() 함수에 데이터를 받을 때마다 호출할 콜백 함수를 매개변수로 지정

• 이벤트 핸들러 등록하기

```
const channel = EventChannel('eventChannel');
channel.receiveBroadcastStream().listen((dynamic event) {
  ... (생략) ...
}, onError: (dynamic error) {
  print('Received error: ${error.message}');
});
```

이벤트 채널 이용하기

- 안드로이드 코드
  - EventChannel 객체로 채널을 만들며 setStreamHandler() 함수의 매개변수에 EventChannel.StreamHandler 타입의 객체를 지정
  - 이벤트 채널의 데이터를 받겠다고 등록하는 순간 onListen() 함수가 실행
  - EventChannel.EventSink는 이벤트를 주입하는 객체
  - EventSink의 success() 함수로 데이터를 보냅니다.

• 이벤트 주입하기(코틀린)

```
val eventChannel = EventChannel(flutterEngine.dartExecutor, "eventChannel");
eventChannel.setStreamHandler( object : EventChannel.StreamHandler {
  override fun onListen(p0: Any?, p1: EventChannel.EventSink?) {
    io.flutter.Log.d("platform", "onListen.......")
    p1?.success("send event data..from native..")
  }
  override fun onCancel(p0: Any?) {

  }
})
```

이벤트 채널 이용하기

- iOS 코드
  - FlutterEventChannel로 이벤트 채널 객체를 만들고 setStreamhandler() 함수의 매개변수에 FlutterStreamHandler 타입의 객체를 지정
  - 이벤트를 받겠다고 등록하는 순간 FlutterStream Handler의 onListen() 함수가 자동으로 호출
  - 두 번째 매개변수로 이벤트를 주입하는 FlutterEventSink 객체가 전달

• 이벤트 주입하기(스위프트)

```
class SwiftStreamHandler: NSObject, FlutterStreamHandler {
  public func onListen(withArguments arguments: Any?,
    eventSink events: @escaping FlutterEventSink) -> FlutterError? {
    events("send event data..from ios native..")
    return nil
  }

  public func onCancel(withArguments arguments: Any?) -> FlutterError? {
    return nil
  }
}
... (생략) ...
let eventChannel = FlutterEventChannel(name: "eventChannel",
  binaryMessenger: controller.binaryMessenger)
eventChannel.setStreamHandler(SwiftStreamHandler())
```

감사합니다

단단히 마음먹고 떠난 사람은
산꼭대기에 도착할 수 있다.
산은 올라가는 사람에게만 정복된다.

윌리엄 셰익스피어
William Shakespeare