Mobile Computing

Flutter
Native Channels

Platform Dependent Code

The Platform class has Properties that have different values, depending on the device platform

platform.IsAndroid (prop) → bool

platform.isIOS (prop) → bool

platform.operatingSystem (prop) → String
can return ‘android’, ‘ios’, or other operating systems (macos, linux, windows, fuchsia)

Other properties:

platform.operatingSystemVersion (prop) → String
A string representing the version number

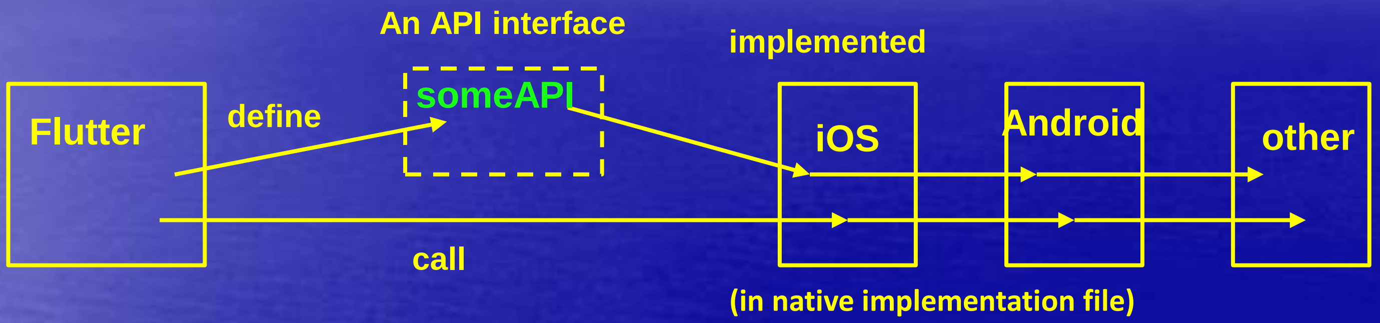
platform.localHostname (prop) → String

platform.numberOfProcessors (prop) → int

platform.localeName (prop) → String
language and region

From Flutter to Device Specific

To call code from the Common project, targeting any of the specific platform applications a MethodChannel implementation must be used.
The MethodChannel carries a method name and parameters.
Returns a value.

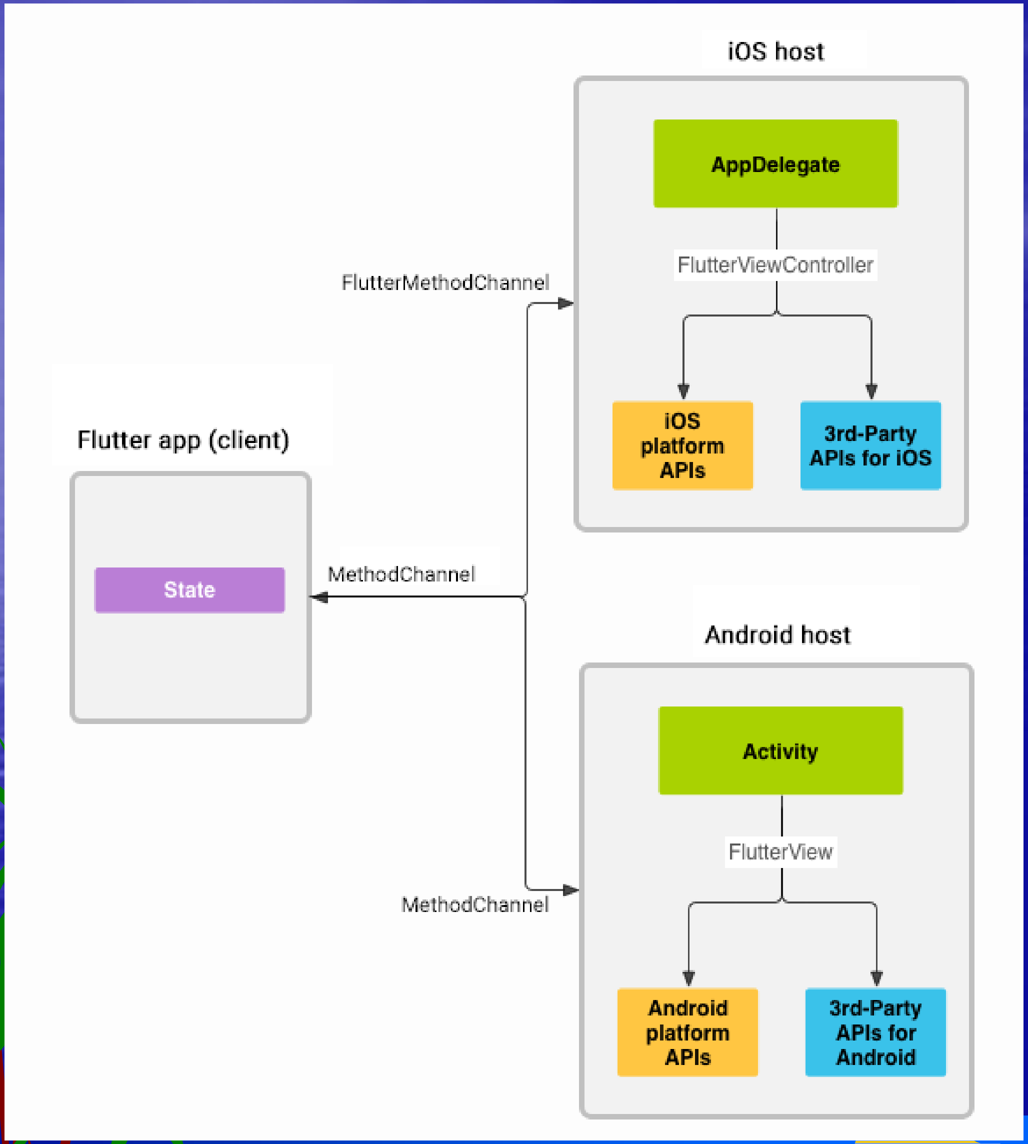


(accessing in Flutter code)

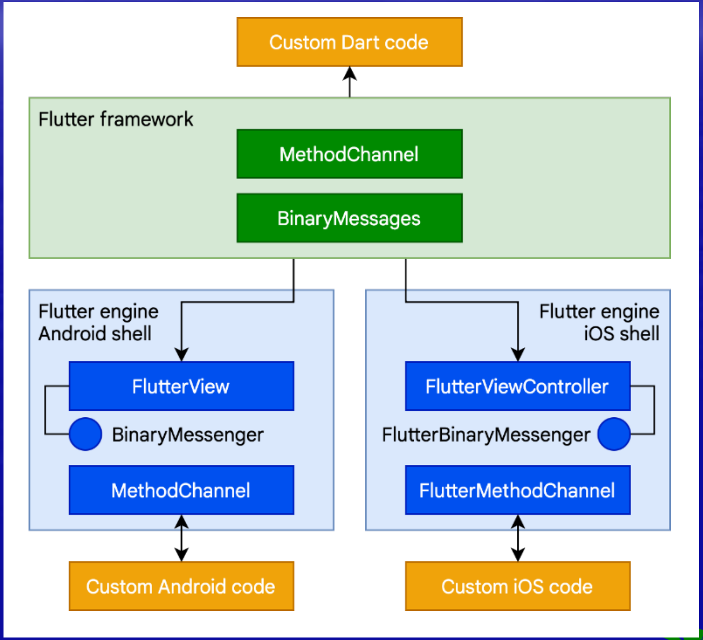
```
…
const channel = MethodChannel('API_name');
…
final ResultType response = await channel.invokeMethod('method_name', parameters);
…
```

The channel can also define Callback methods to be called from native code.

Platform Specific Code

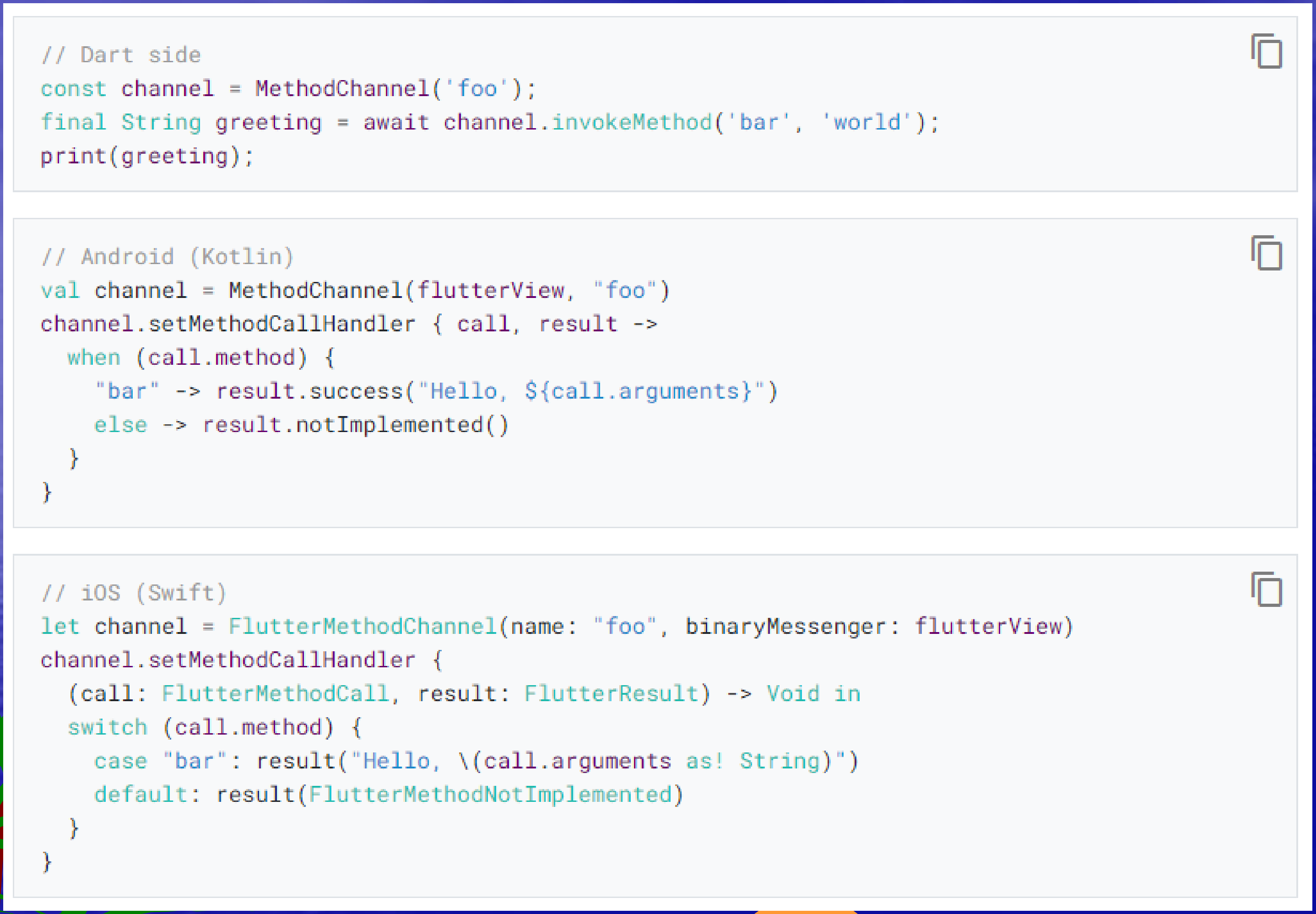


It is possible to call any native functionality using a MethodChannel



The concrete MethodChannel must be implemented both in the Flutter side, and the native side in a FlutterActivity or FlutterView (overriding configureFlutterEngine() in Android)

Example



```
// Dart side
const channel = MethodChannel('foo');
final String greeting = await channel.invokeMethod('bar', 'world');
print(greeting);
```

```
// Android (Kotlin)
val channel = MethodChannel(flutterView, "foo")
channel.setMethodCallHandler { call, result ->
  when (call.method) {
    "bar" -> result.success("Hello, ${call.arguments}")
    else -> result.notImplemented()
  }
}
```

```
// iOS (Swift)
let channel = FlutterMethodChannel(name: "foo", binaryMessenger: flutterView)
channel.setMethodCallHandler {
  (call: FlutterMethodCall, result: FlutterResult) -> Void in
  switch (call.method) {
    case "bar": result("Hello, \(call.arguments as! String)")
    default: result(FlutterMethodNotImplemented)
  }
}
```

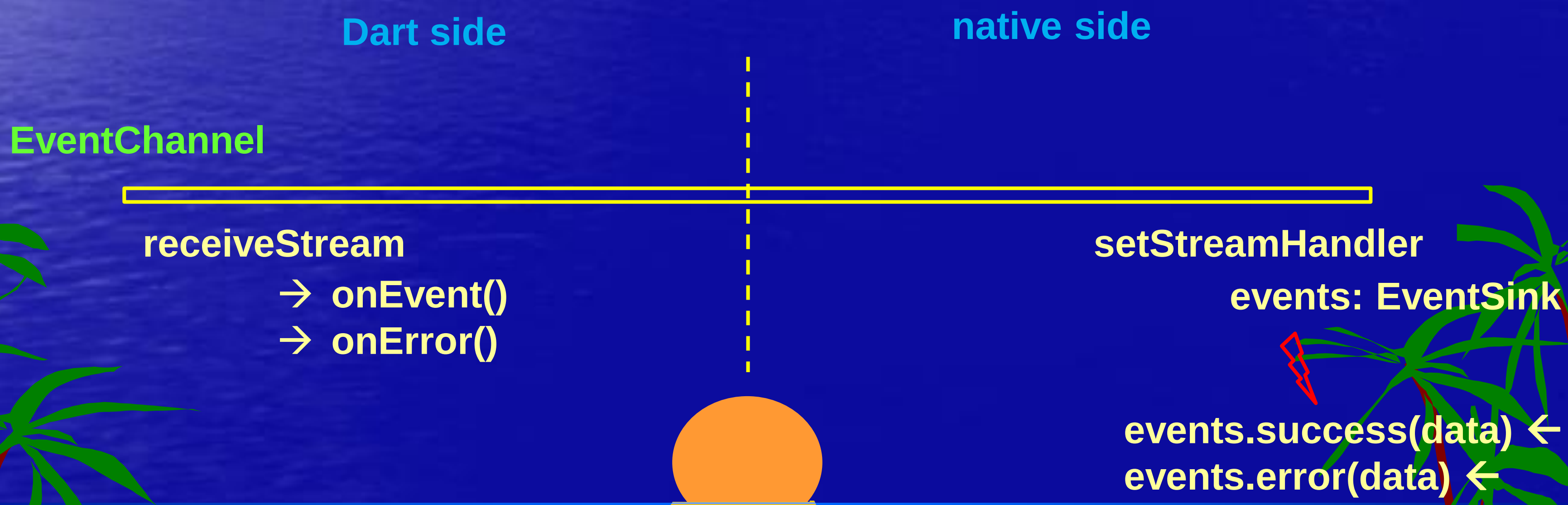
Catching Native Events

Can be done using an EventChannel between the Dart side and the native side

On the Dart side a receiver should be established defining callbacks called when an event is generated in the native side

The native side establishes a StreamHandler attached to the channel, together with an EventSink, capable of generating events that can carry data. It’s possible to also generate error events.

The appropriate Dart side callback is called whenever an event is generated.


Dart side
native side
EventChannel
receiveStream
→ onEvent()
→ onError()
setStreamHandler
events: EventSink
events.success(data) ←
events.error(data) ←