Android, iOS and Hybrid Applications

Mobile-App Development

LESSON 5: AGENDA

- Grading, evaluation, and schedule
- Hybrid Apps: hands-on
  - Communication native/webView
    - Examine sample
    - Design interface

Your final grade will be based on the following:

- "Projektwoche": teamwork and implementation
- Individual project
  - Implementation (05.07.2021)
  - Presentation (12.07.2021)
- Written exam: Zoom/40 Minutes (05.07.2021)
- Class participation

Grading the Implementation

- Has the required features (more is better 😎)
- Compiles and runs
- Architecture
- Implementation
- UI and UX
- Documentation (readme, developer docs, etc.)
- Tests
- See "/Bewertung Einzelarbeit.xlsx" in the *APE2021_App* repository

Grading the Presentation

- Introduction
- Logical structure
- Demonstrates the app
- Time limit (10 Minutes)

Lesson 8: Timeline and Deadlines

- Written exam
- Work on project
  - Final questions
  - Final tweaks and changes
  - Projects must be pushed by the end of the lesson
  - I will pull code at 21:30
  - I will send grades by mail

Presentation schedule (Proposed)

- Tiago: 17:50-18:10
- Ralph: 18:15-18:35
- Christian: 18:40-19:00
- Felix: 19:05-19:25
- **Break**
- Nathan: 19:45-20:05
- Sven: 20:10-20:30
- Raffaele: 20:30-20:50

Questions?

OVERVIEW

- Hybrid Applications
- Interoperability with the native part
  - Design a possible interface
  - Present your approach
- Create a small working sample

Hybrid Applications

- Native Part which provides a JS-Interface
- More than 50% market share (hard to prove)
- Browsers support modern HTML/CSS/JS
  - Check https://caniuse.com/

Hybrid Applications: Pros

- Share or reuse (UI)-Code (from website etc.)
- It’s easier to find Web-Devs then native Devs
- Possible to update App without releasing through the store
- Fallbacks/Combination with native possible

HYBRID APPLICATIONS: CONS

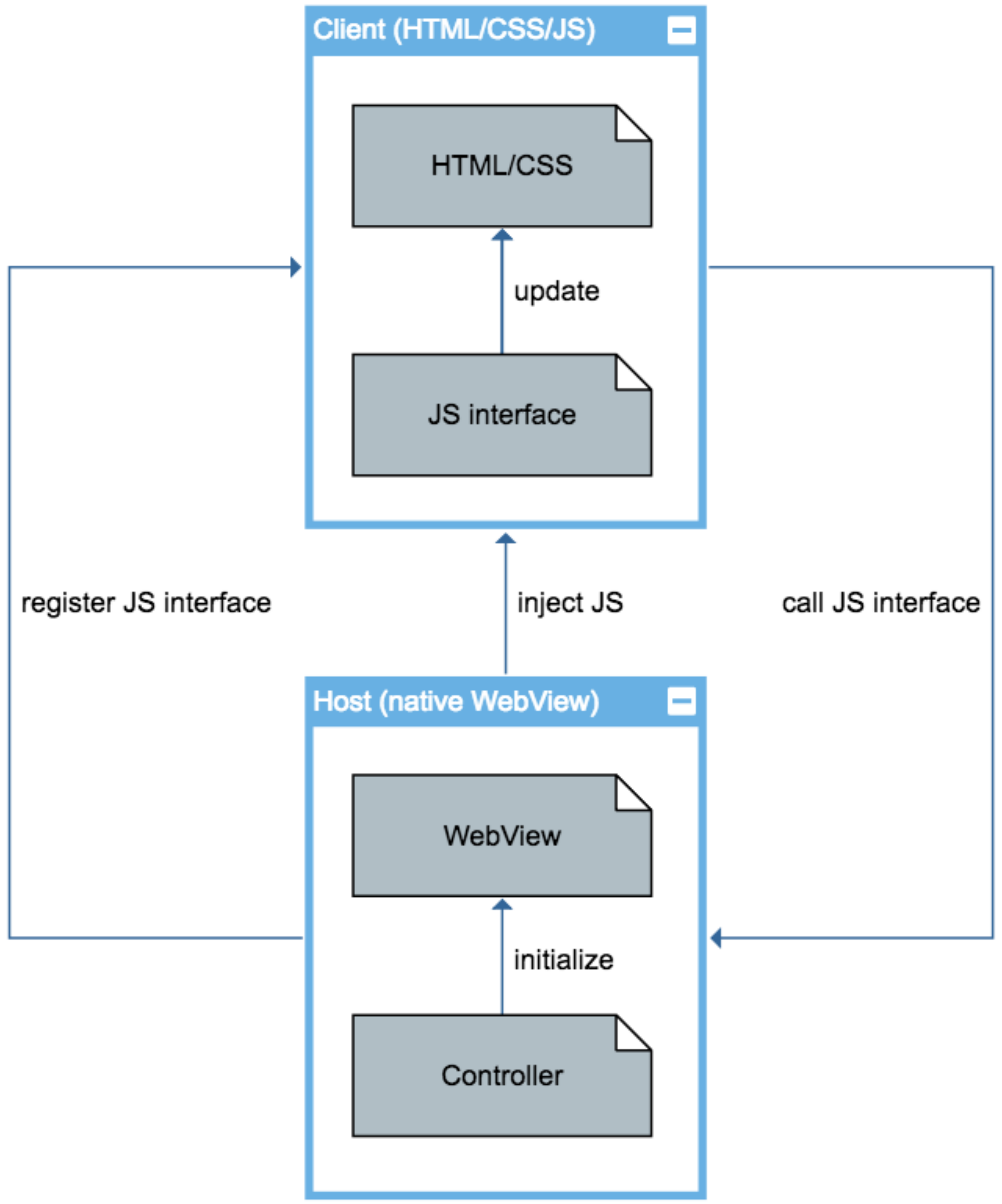
- Sometimes don’t feel that “responsive”
  - Getting better with later releases/engines
- You need to understand both worlds (native & web)

WEBVIEWS

- Xamarin
  - Customizing a WebView
  - Xamarin.Forms WebView
- iOS
  - WKWebView
  - **Don't** use UIWebView
- Android
  - WebView
  - Updates independent of OS (since v4.4.4/API19)

WebViews

- The control is a wrapper - they run in their own process
- You're not limited to only use one WebView
- Load local HTML pages or remote ones
- Think about CORS when using a mix


Client (HTML/CSS/JS)
HTML/CSS
update
JS interface
register JS interface
inject JS
call JS interface
Host (native WebView)
WebView
initialize
Controller

SETUP THE APP

- Sample implementation in APE2021_App
  - Branch: feature/day5_hybrid_apps
  - Folder: /Hybrid
  - Implementation: Xamarin w/Android-only

Register a JS Interface

```
// Set the content layout which contains a simple web view.
SetContentView(Resource.Layout.activity_main);

// Extract the web view from the layout.
var webView = (WebView)FindViewById(Resource.Id.webView);

// Configure WebView to allow JS and inject our custom interface.
webView.Settings.JavaScriptEnabled = true;
webView.AddJavascriptInterface(new JavaScriptInject(this), "Native");

// Load a local HTML file.
webView.LoadUrl("file:///android_asset/index.html");
```

REGISTER A JS INTERFACE

```
public class JavaScriptInject : Object
{
  /// <summary>
  /// Annotate methods with the <see cref="JavascriptInterfaceAttribute"/> and
  /// the <see cref="ExportAttribute"/> to call them from JS.
  /// </summary>
  [JavascriptInterface]
  [Export("doSomething")]
  public void FromJavaScript()
  {
  }

  /// <summary>
  /// Annotated methods can also accept parameters.
  /// </summary>
  [JavascriptInterface]
  [Export("doSomething")]
  public void FromJavaScript(string message)
  {
  }
}
```

Invoke Native from JS (WebView -> Native)

- Invoke it with Native.yourMethod()

```
<input type="button" onclick="Native.doSomething()" value="Invoke native" />
```

- Also possible with parameters

```
<input type="button" onclick="Native.doSomething('Another message...')" value="Invoke native
with param" />
```

INJECT JS (NATIVE -> WEBVIEW)

```
webView.EvaluateJavascript("do some JS magic...", null);
```

```
webView.EvaluateJavascript("do some JS magic...", new Callback());

public class Callback : Java.Lang.Object, IValueCallback
{
  public void OnReceiveValue(Object value)
  {
    // Do something with the value...
  }
}
```

https://docs.microsoft.com/en-gb/dotnet/api/android.webkit.webview.evaluatejavascript

Example

- Walkthrough

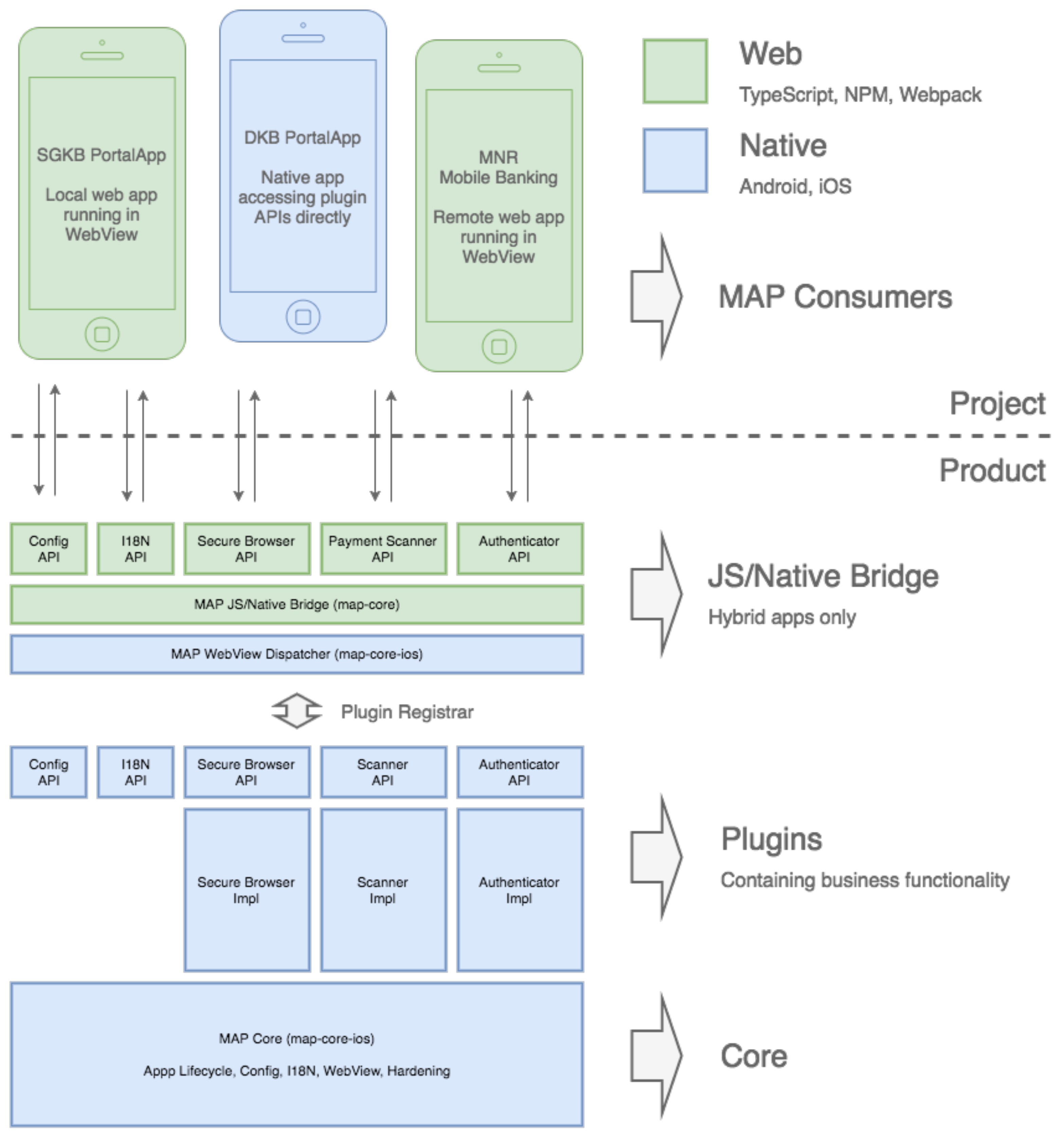
EXERCISE

- Break into groups (breakout rooms)
- Set up the basic Android project
- Clone the repo for your group
- Think about an approach on how to create a messaging bus between Native and Web
- Present your solution/idea

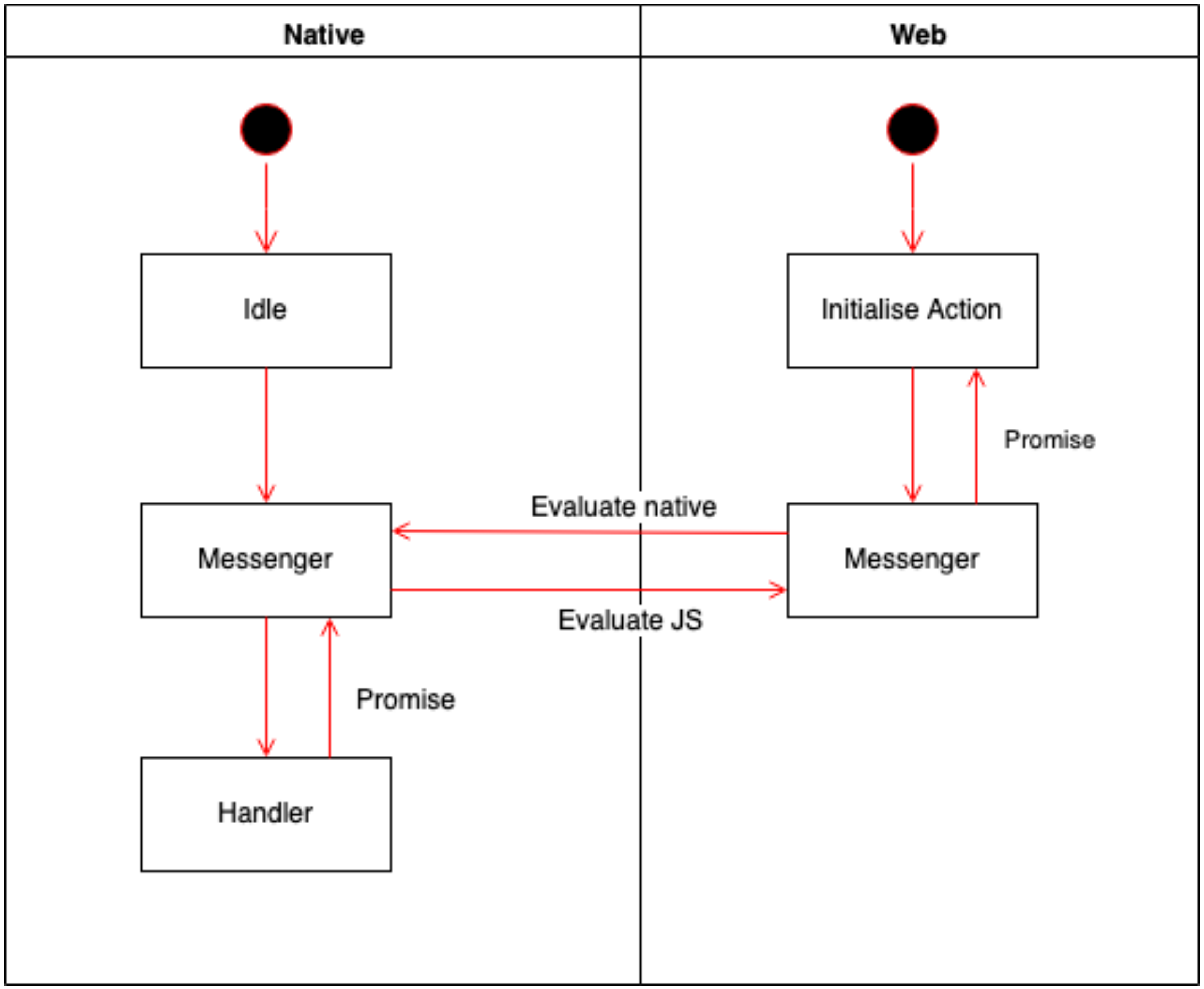
Possible Solution

- Mediator pattern across Native/JS
- Send messages and distribute them
- Web “drives” the app
- Native is used like an “API”
- Pattern is also known as “Messenger” in WPF

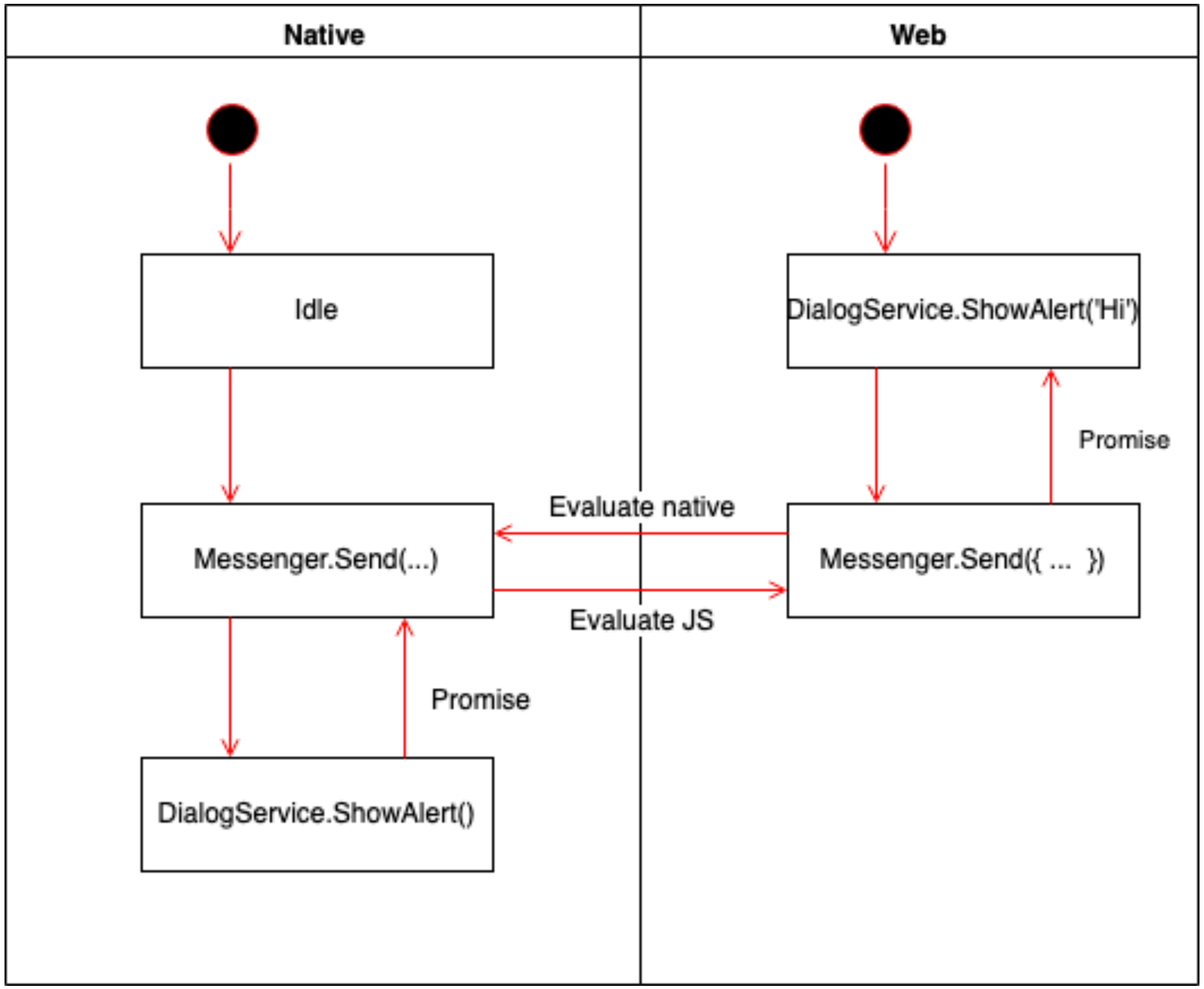
ARCHITECTURE



WORKFLOW


Native
Web
Idle
Initialise Action
Promise
Evaluate native
Messenger
Messenger
Evaluate JS
Promise
Handler

WORKFLOW


Native
Web
Idle
Messenger.Send(...)
DialogService.ShowAlert()
Promise
DialogService.ShowAlert('Hi')
Messenger.Send({ ... })
Promise
Evaluate native
Evaluate JS

EXAMPLE CODE

```
// JavaScript/WebView
return this.messenger.send(new SetBiometricValueMessage(entry, btoa(value)))
    .then((response: OperationResponse) => {
        return response.success;
    });

// Java/Android
messageRegistrar.registerHandler(
                                  SetBiometricValueMessage.TAG,
                                  SetBiometricValueMessage.class,
                                  new MessageHandler<EmptyResponse, SetBiometricValueMessage>() {
  @Override
  public MapPromise<EmptyResponse> invoke(SetBiometricValueMessage
setBiometricValueMessage) {
    return biometricStorage.setValue(
                                 setBiometricValueMessage.key,
                                 encodingUtils.fromBase64(setBiometricValueMessage.value));
  }
});
```

Rest of the Evening

- Continue working on the Forms application
- Try to finish goals from previous lessons