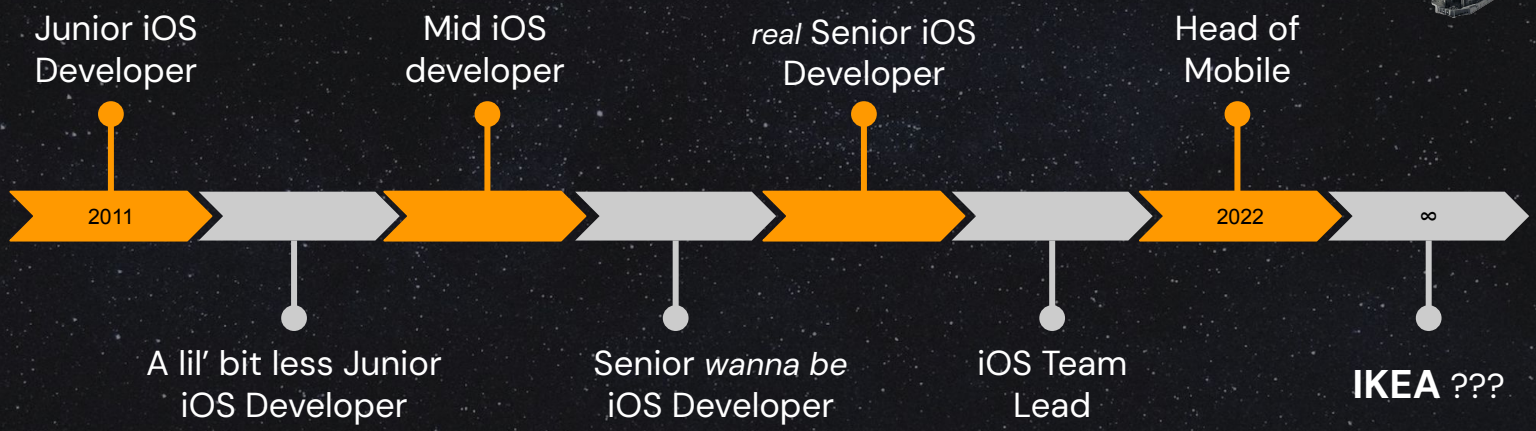
MOBILE WARS

Revenge of the Benchmark

Who am I?



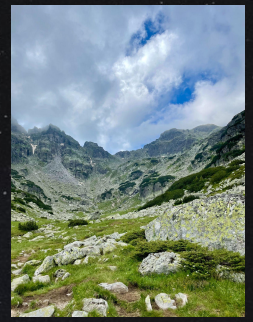
Don't over engineer

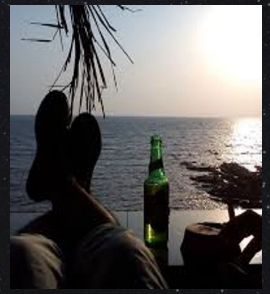
iOS rulz!

People centric

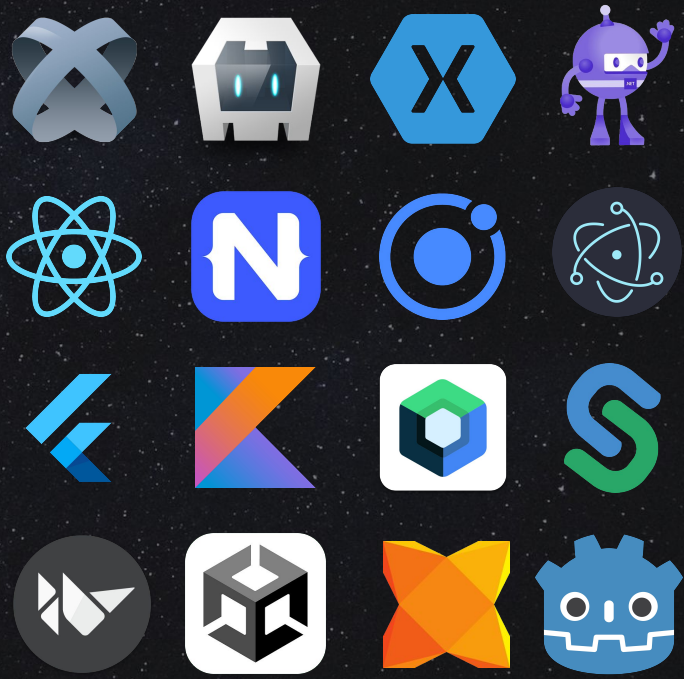
Who am I?

- Sci-Fi/Physics/TV Shows
- Travelling/Hiking
- Eating/Beers
- Hip Hop
- Basketball
- Chillin’





The topic/battle



The topic/battle

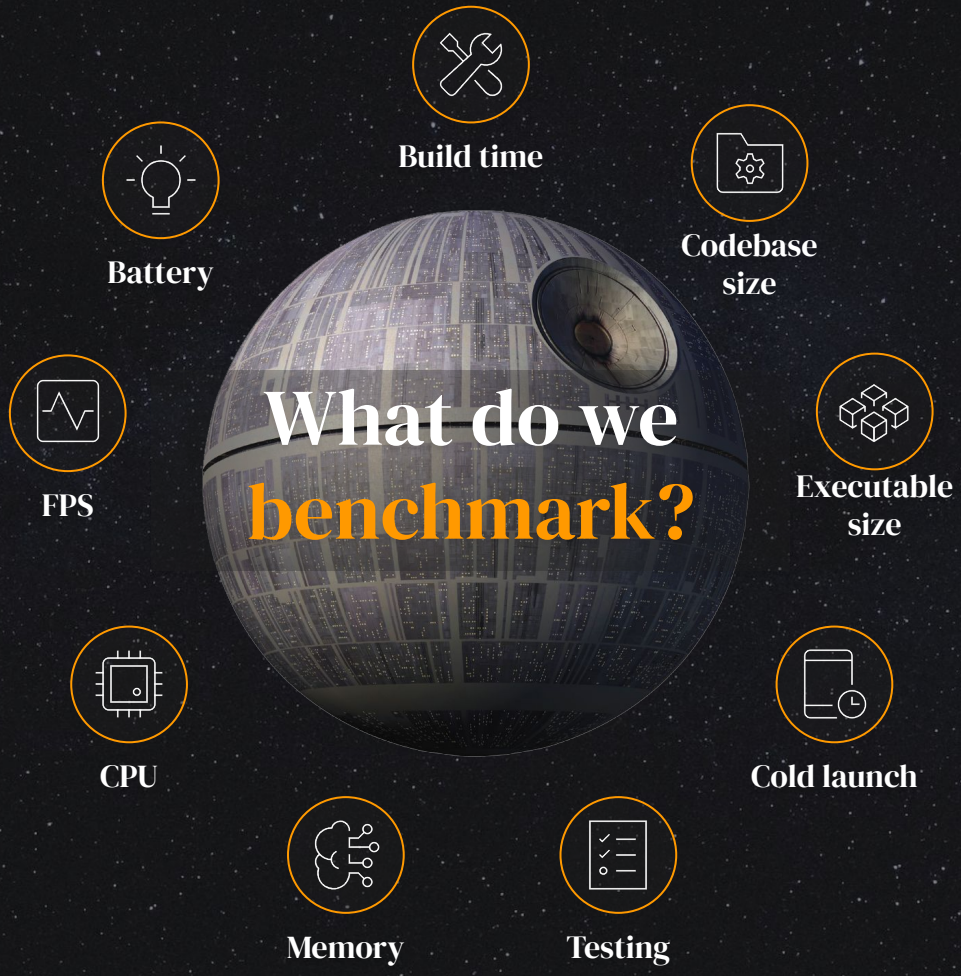
Games

Web-Based

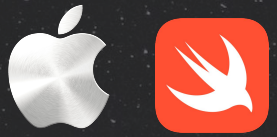
Native-like

Create the same application in all major mobile technologies and benchmark various aspects of them.

- Simple
- Latest
- Feature parity
- No third parties
- Same hardware
- By experts


Build time
Codebase size
Executable size
Cold launch
Testing
Memory
CPU
FPS
Battery
What do we benchmark?

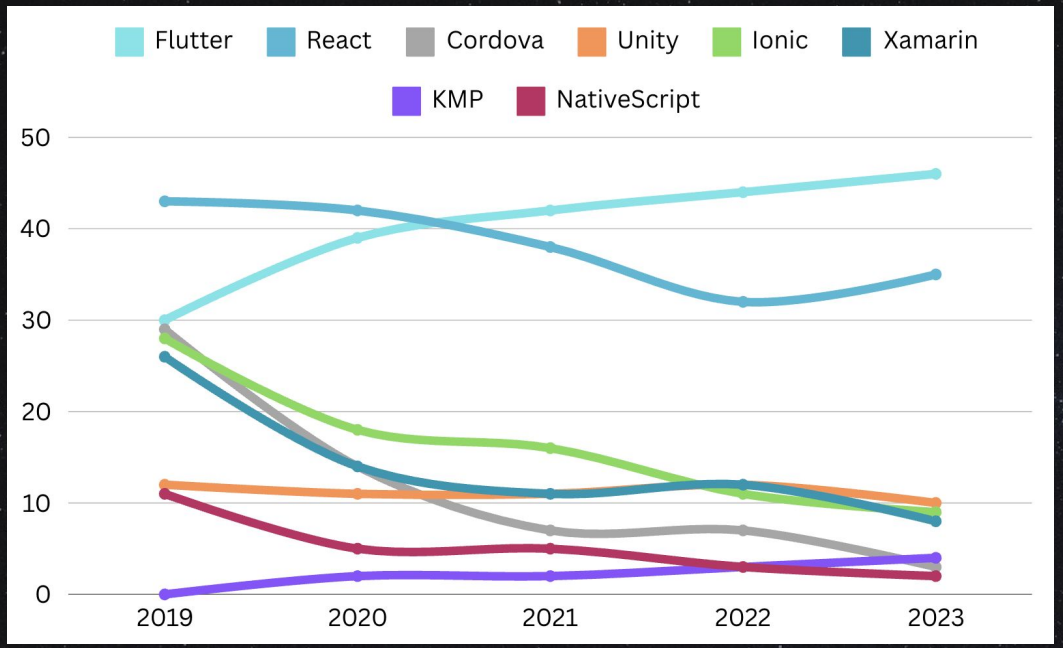
iOS

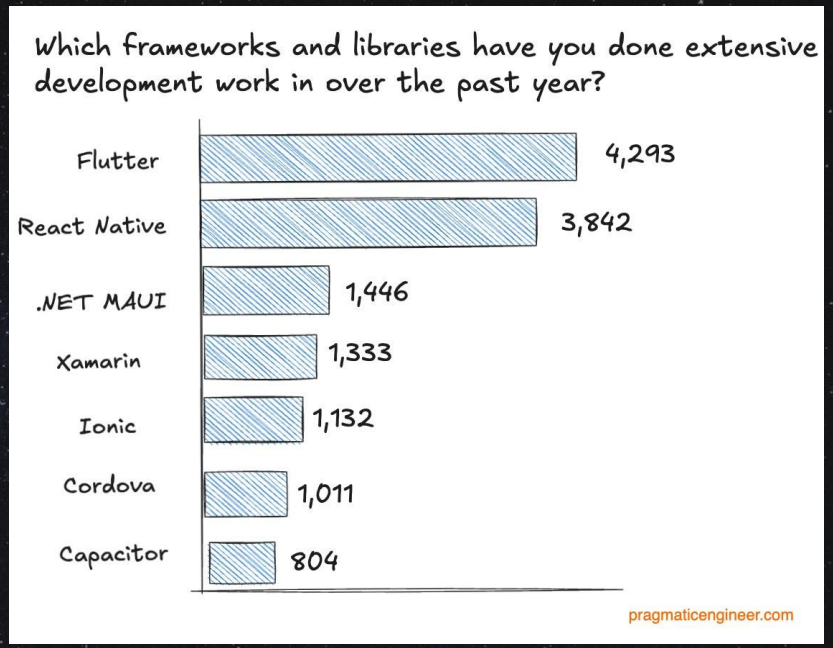


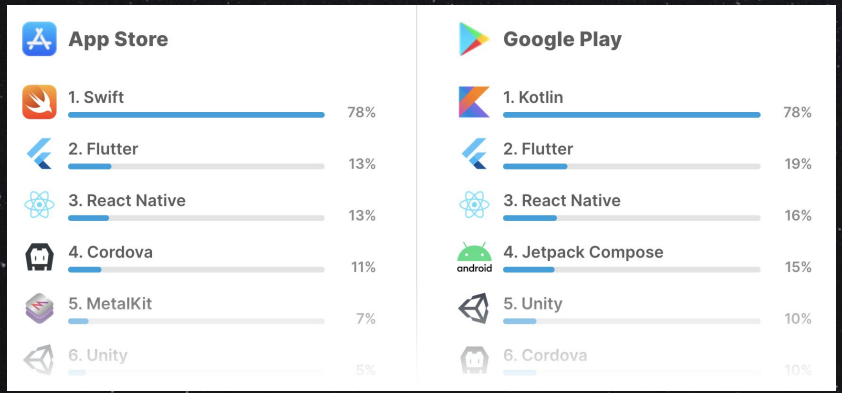
Old iOS

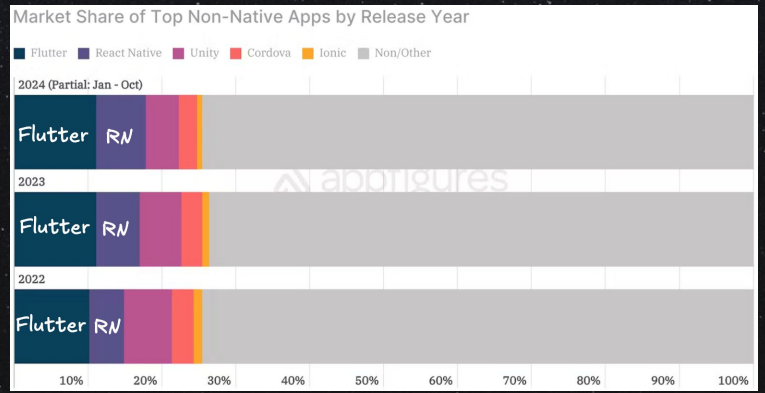
Android

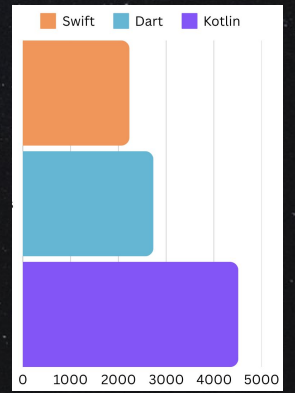
Old Adroid


Flutter
React
Cordova
Unity
Ionic
Xamarin
KMP
NativeScript
50
40
30
20
10
0
2019
2020
2021
2022
2023


Which frameworks and libraries have you done extensive development work in over the past year?
Flutter
4,293
React Native
3,842
.NET MAUI
1,446
Xamarin
1,333
Ionic
1,132
Cordova
1,011
Capacitor
804
pragmaticengineer.com


App Store
1. Swift 78%
2. Flutter 13%
3. React Native 13%
4. Cordova 11%
5. MetalKit 7%
6. Unity 5%
Google Play
1. Kotlin 78%
2. Flutter 19%
3. React Native 16%
4. Jetpack Compose 15%
5. Unity 10%
6. Cordova 10%

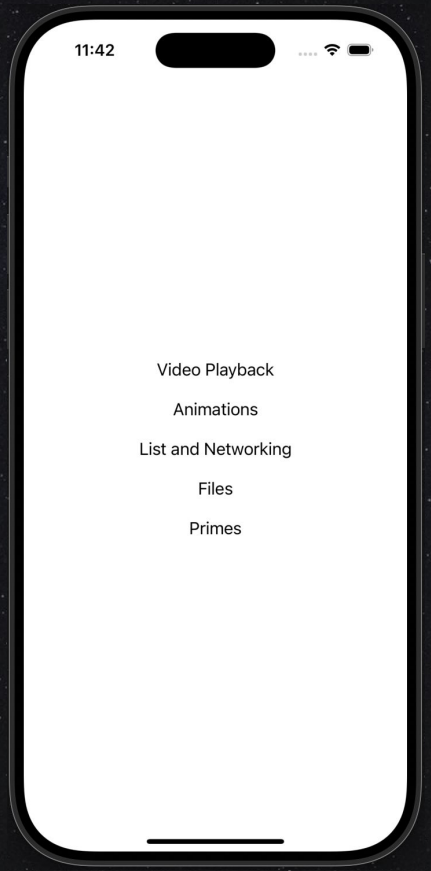

Market Share of Top Non-Native Apps by Release Year
Flutter
React Native
Unity
Cordova
Ionic
Non/Other
2024 (Partial: Jan - Oct)
2023
2022
Flutter
RN
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%


Swift
Dart
Kotlin
0
1000
2000
3000
4000
5000


iOS
Old iOS
[ObjC]
KMP
Flutter
Android
Old Android
Java
CMP
ReactNative
TS

The App

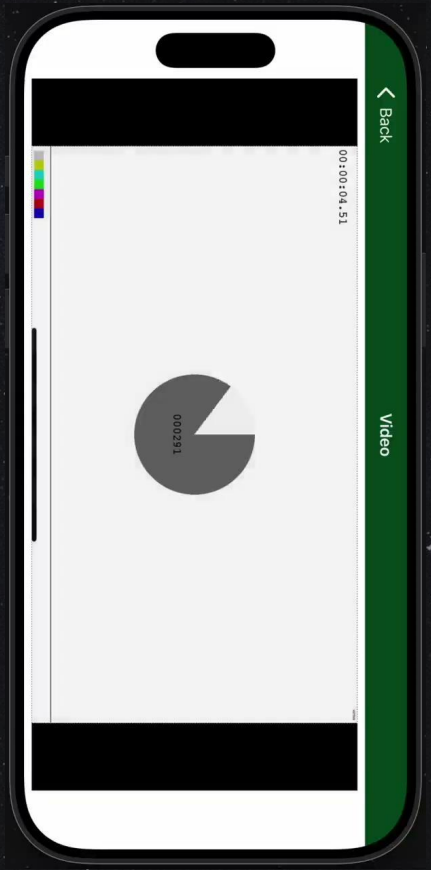
Landing screen



The App

- Landing screen
- Video playback

- 60fps video
- Full HD
- Landscape only
- .m3u8
- Custom video layer



The App

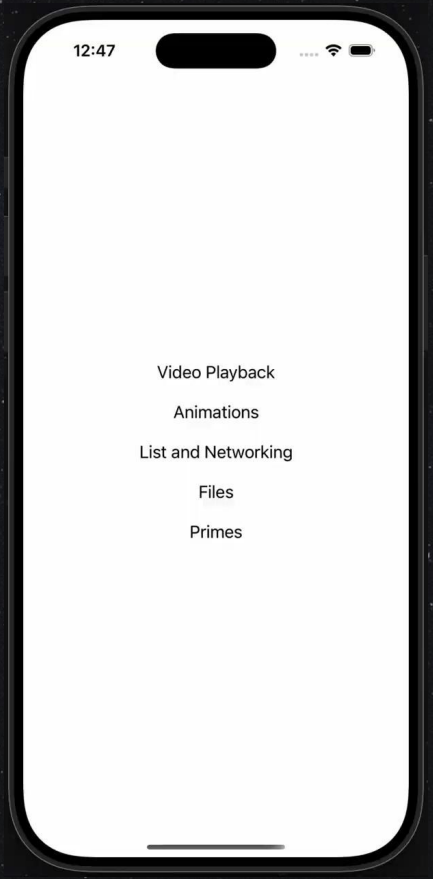
- Landing screen
- Video playback
- Animations

- 3 images (png, jpg, svg)
- 3 infinite animations
  - fade
  - scale
  - rotate
- x100 per type

The App

- Landing screen
- Video playback
- Animations
- List and Networking

- GET request to a 26 MB JSON
- Serialize in an array (12000)
- Perform sort and filter (6000)
- List with remote image and title



The App

- Landing screen
- Video playback
- Animations
- List and Networking
- Files

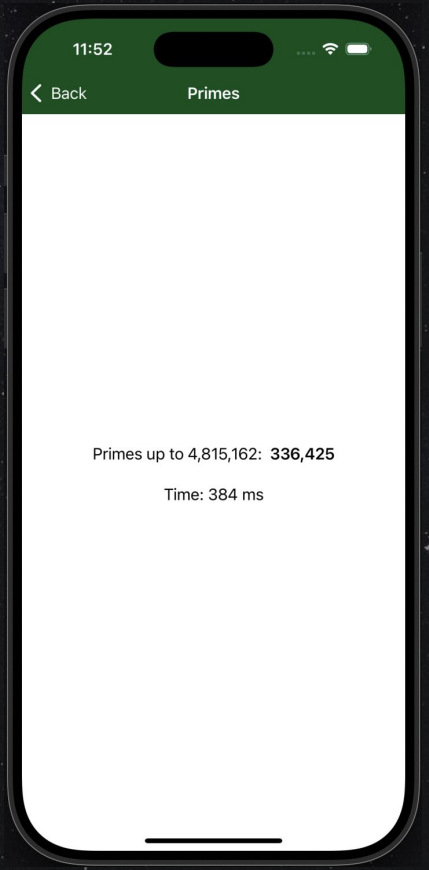
- GET request to a 26 MB JSON
- Write it 20 times on the file system
- Delete all files

The App

- Landing screen
- Video playback
- Animations
- List and Networking
- Files
- Computation

- Simple but heavy algorithm
- Find first n prime number lower that 4 815 162

```
static func countPrimes(upTo n: Int) async -> Int {
    if n <= 1 { return 0 }
    var count = 0

    for number in 2...n {
        var isPrime = true
        let maxDivisor = Int(Double(number).squareRoot())
        if maxDivisor >= 2 {
            for divisor in 2...maxDivisor {
                if number % divisor == 0 {
                    isPrime = false
                    break
                }
            }
        }
        if isPrime {
            count += 1
        }
    }
    return count
}
```

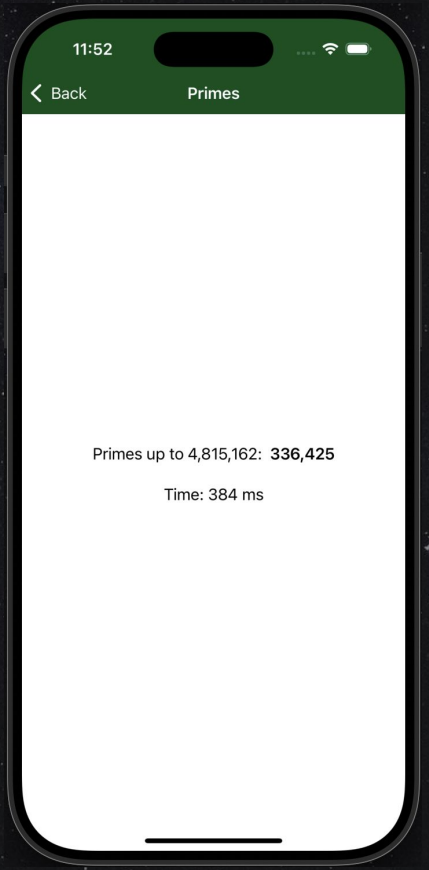


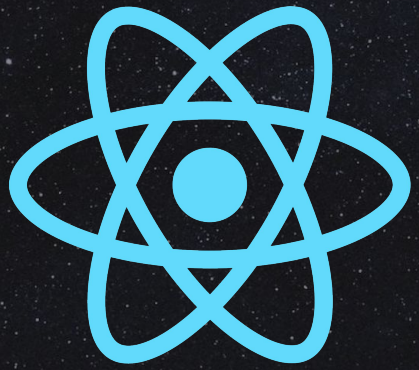
The App

- Landing screen
- Video playback
- Animations
- List and Networking
- Files
- Computation
- Unit testing

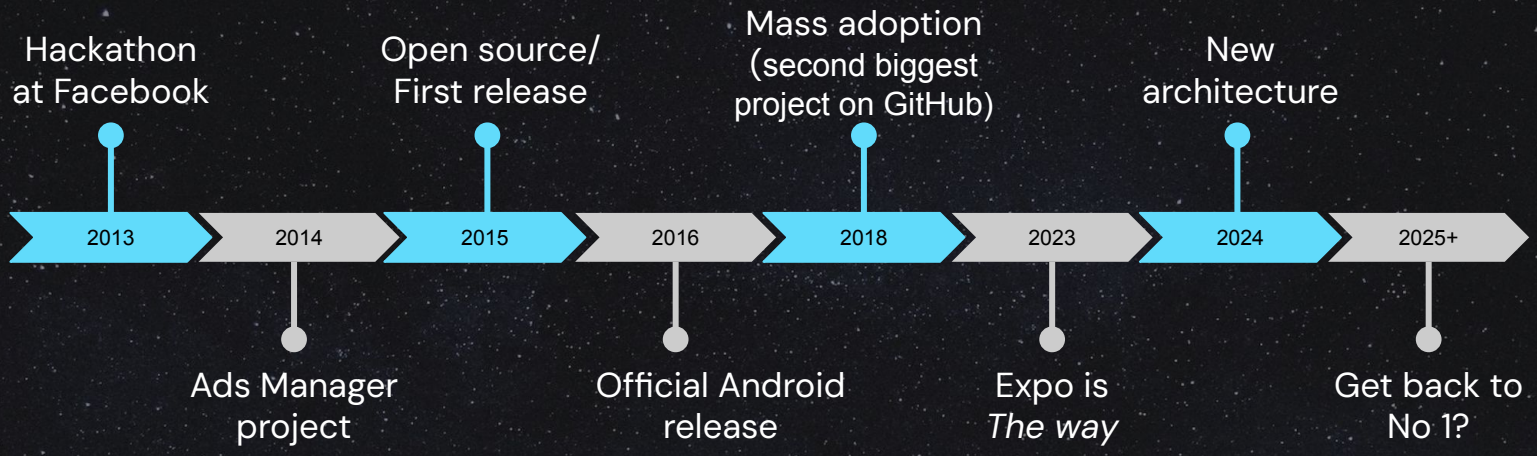
- Simple but heavy algorithm
- Find first n prime number lower that 4 815 162

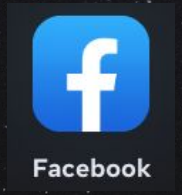
```
static func countPrimes(upTo n: Int) async -> Int {
    if n <= 1 { return 0 }
    var count = 0

    for number in 2...n {
        var isPrime = true
        let maxDivisor = Int(Double(number).squareRoot())
        if maxDivisor >= 2 {
            for divisor in 2...maxDivisor {
                if number % divisor == 0 {
                    isPrime = false
                    break
                }
            }
        }
        if isPrime {
            count += 1
        }
    }
    return count
}
```

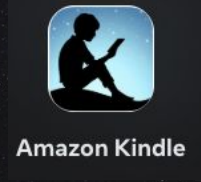




React Native


Hackathon at Facebook
Open source/ First release
Mass adoption (second biggest project on GitHub)
New architecture
2013
2014
2015
2016
2018
2023
2024
2025+
Ads Manager project
Official Android release
Expo is *The way*
Get back to No 1?


Facebook


Amazon Kindle
Microsoft Office

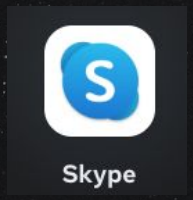

Microsoft Teams


OpenVPN Connect


Amazon Shopping

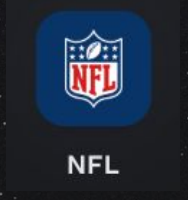

Shopify

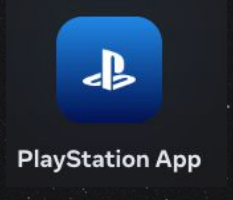

Instagram

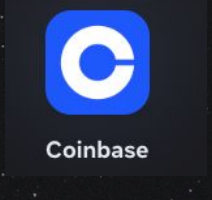

Skype


Tesla


Discord


NFL


PlayStation App


Coinbase

Language: TypeScript

```typescript
export const countPrimes = (n: number): number => {
 if (n <= 1) return 0;
 let count = 0;

 for (let number = 2; number <= n; number++) {
   let isPrime = true;
   const maxDivisor = Math.floor(Math.sqrt(number));
   if (maxDivisor >= 2) {
     for (let divisor = 2; divisor <= maxDivisor; divisor++) {
       if (number % divisor === 0) {
         isPrime = false;
         break;
       }
     }
   }
   if (isPrime) count++;
 }
 return count;
};
```

```swift
static func countPrimes(upTo n: Int) async -> Int {
    if n <= 1 { return 0 }
    var count = 0

    for number in 2...n {
        var isPrime = true
        let maxDivisor = Int(Double(number).squareRoot())
        if maxDivisor >= 2 {
            for divisor in 2...maxDivisor {
                if number % divisor == 0 {
                    isPrime = false
                    break
                }
            }
        }
        if isPrime {
            count += 1
        }
    }
    return count
}
```

UI: JSX

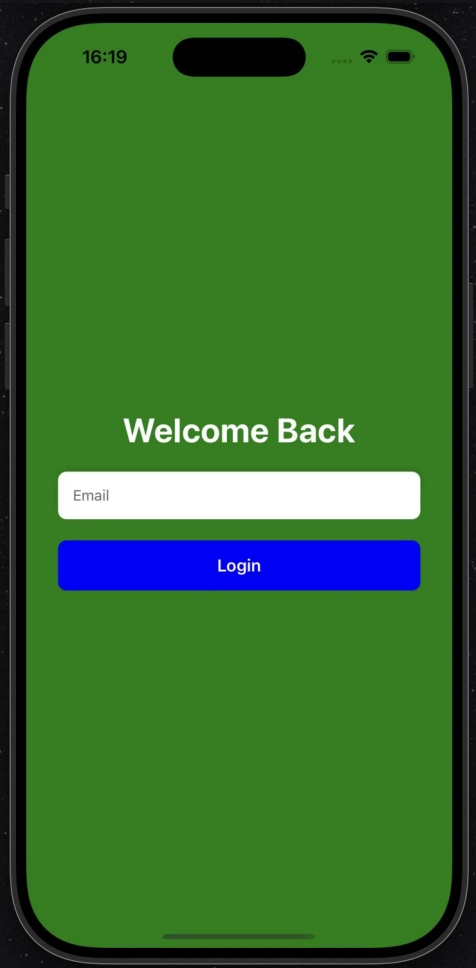
```
struct LoginView: View {
    @State private var email: String = ""

    var body: some View {
        VStack(spacing: 20) {
            Spacer()

            Text("Welcome Back")
                .font(.largeTitle)
                .foregroundColor(.white)

            TextField("Email", text: $email)
                .padding()
                .background(Color.white)
                .padding(.horizontal, 30)

            Button(action: {
                // ToDo: Login
            }) {
                Text("Login")
                    .frame(maxWidth: .infinity)
                    .padding()
                    .foregroundColor(.white)
                    .background(Color.blue)
            }
            .padding(.horizontal, 30)
            .padding(.top, 10)

            Spacer()
        }
        .background(.green)
    }
}
```

```
import React, { useState } from 'react';…

const LoginScreen: React.FC = () => {
  const [email, setEmail] = useState('');

  const handleLogin = () => {
    // ToDo: Login
  };

  return (
    <SafeAreaView style={styles.container}>
      <KeyboardAvoidingView
        style={styles.inner}
      >
        <View style={styles.spacer} />

        <Text style={styles.title}>Welcome Back</Text>

        <TextInput
          style={styles.input}
          placeholder="Email"
          value={email}
          onChangeText={setEmail}
        />

        <TouchableOpacity style={styles.button} onPress={handleLogin}>
          <Text style={styles.buttonText}>Login</Text>
        </TouchableOpacity>

        <View style={styles.spacer} />
      </KeyboardAvoidingView>
    </SafeAreaView>
  );
};

const styles = StyleSheet.create({…
});
```

UI: JSX

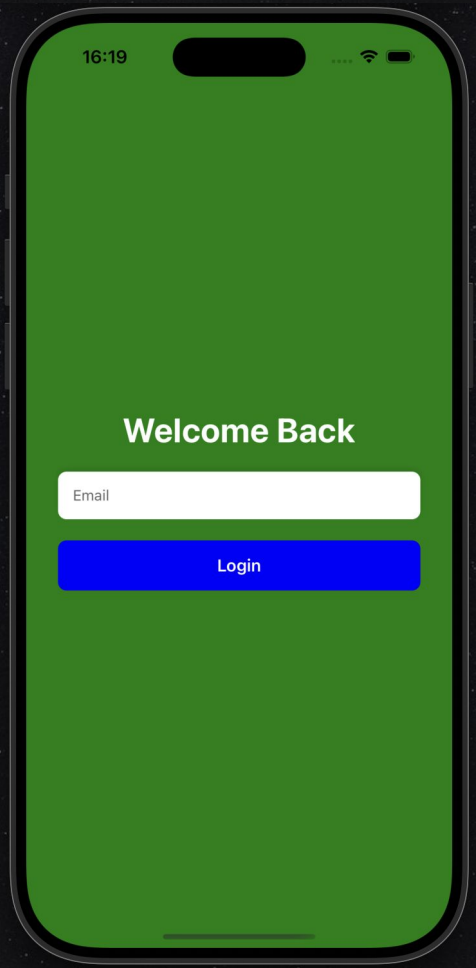
```tsx
import React, { useState } from 'react';

const LoginScreen: React.FC = () => {
  const [email, setEmail] = useState('');

  const handleLogin = () => {
    // ToDo: Login
  };

  return (
    <SafeAreaView style={styles.container}>
      <KeyboardAvoidingView
        style={styles.inner}
      >
        <View style={styles.spacer} />

        <Text style={styles.title}>Welcome Back</Text>

        <TextInput
          style={styles.input}
          placeholder="Email"
          value={email}
          onChangeText={setEmail}
        />

        <TouchableOpacity style={styles.button} onPress={handleLogin}>
          <Text style={styles.buttonText}>Login</Text>
        </TouchableOpacity>

        <View style={styles.spacer} />
      </KeyboardAvoidingView>
    </SafeAreaView>
  );
};

const styles = StyleSheet.create({
});
```

```tsx
const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: 'green',
    justifyContent: 'center',
    paddingHorizontal: 30,
  },
  title: {
    fontSize: 32,
    fontWeight: 'bold',
    color: 'white',
    textAlign: 'center',
    marginBottom: 20,
  },
  input: {
    backgroundColor: 'white',
    padding: 14,
    borderRadius: 8,
    elevation: 2,
    marginBottom: 10,
  },
  button: {
    backgroundColor: 'blue',
    paddingVertical: 14,
    borderRadius: 8,
    marginTop: 10,
  },
  buttonText: {
    color: 'white',
    fontWeight: '600',
    fontSize: 16,
    textAlign: 'center',
  },
});
```

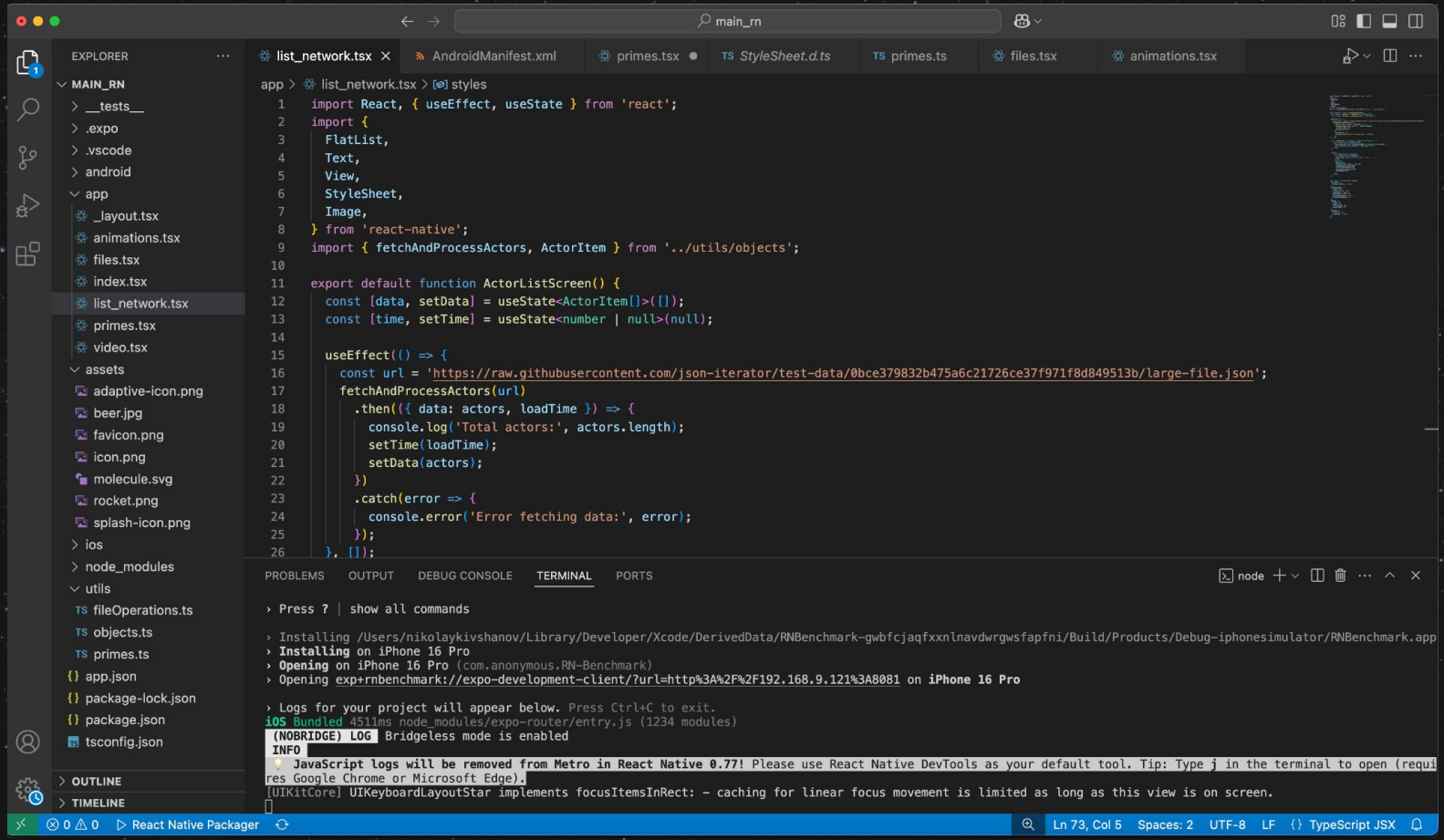


UI: JSX

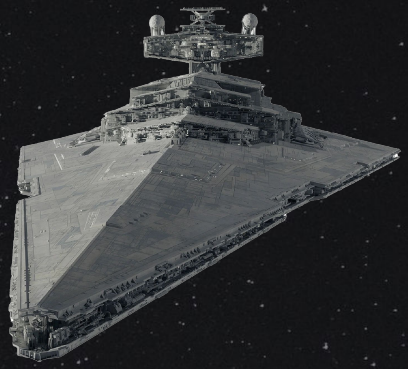
```jsx
<Text style={styles.title}>Welcome Back</Text>

<TextInput
  style={styles.input}
  placeholder="Email"
  value={email}
  onChangeText={setEmail}
/>

<TouchableOpacity style={styles.button} onPress={handleLogin}>
  <Text style={styles.buttonText}>Login</Text>
</TouchableOpacity>
```



IDE: Visual Studio Code



How does it work?

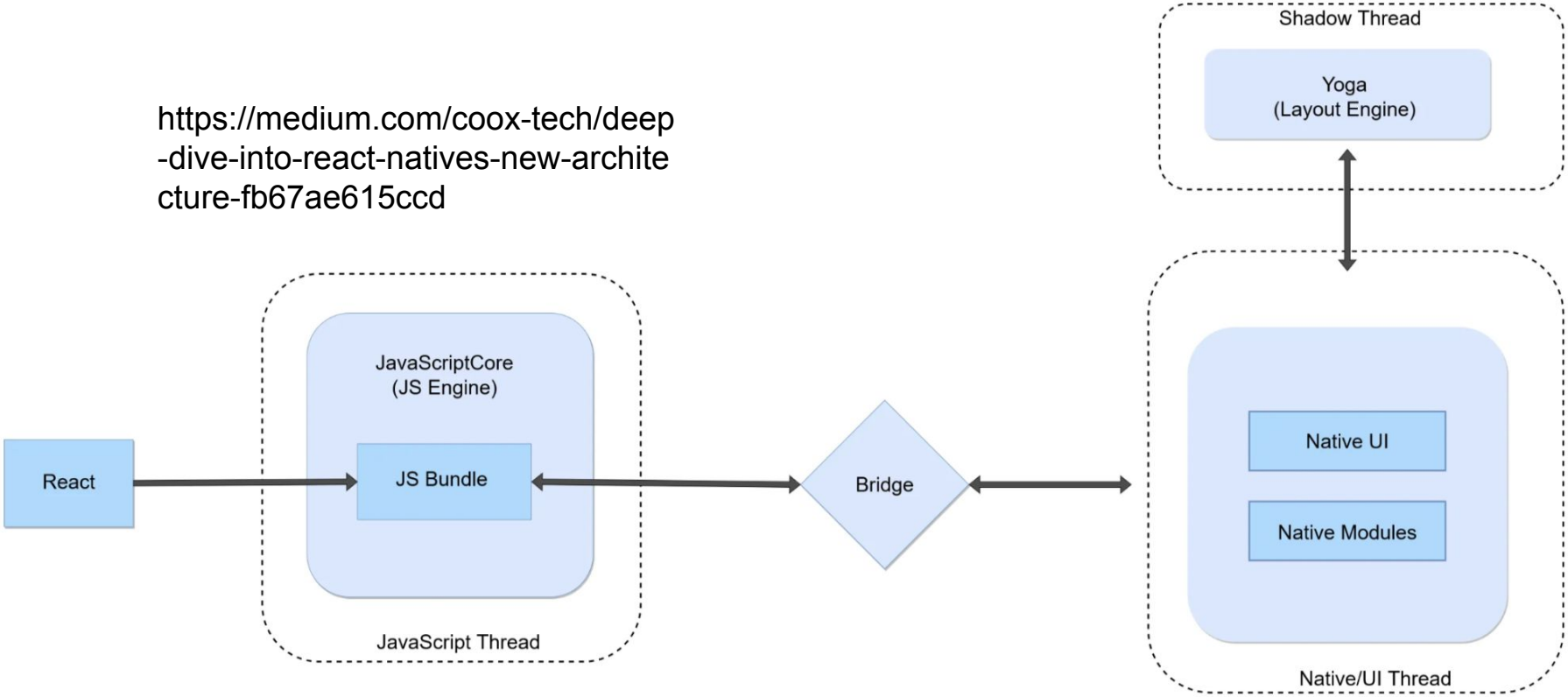


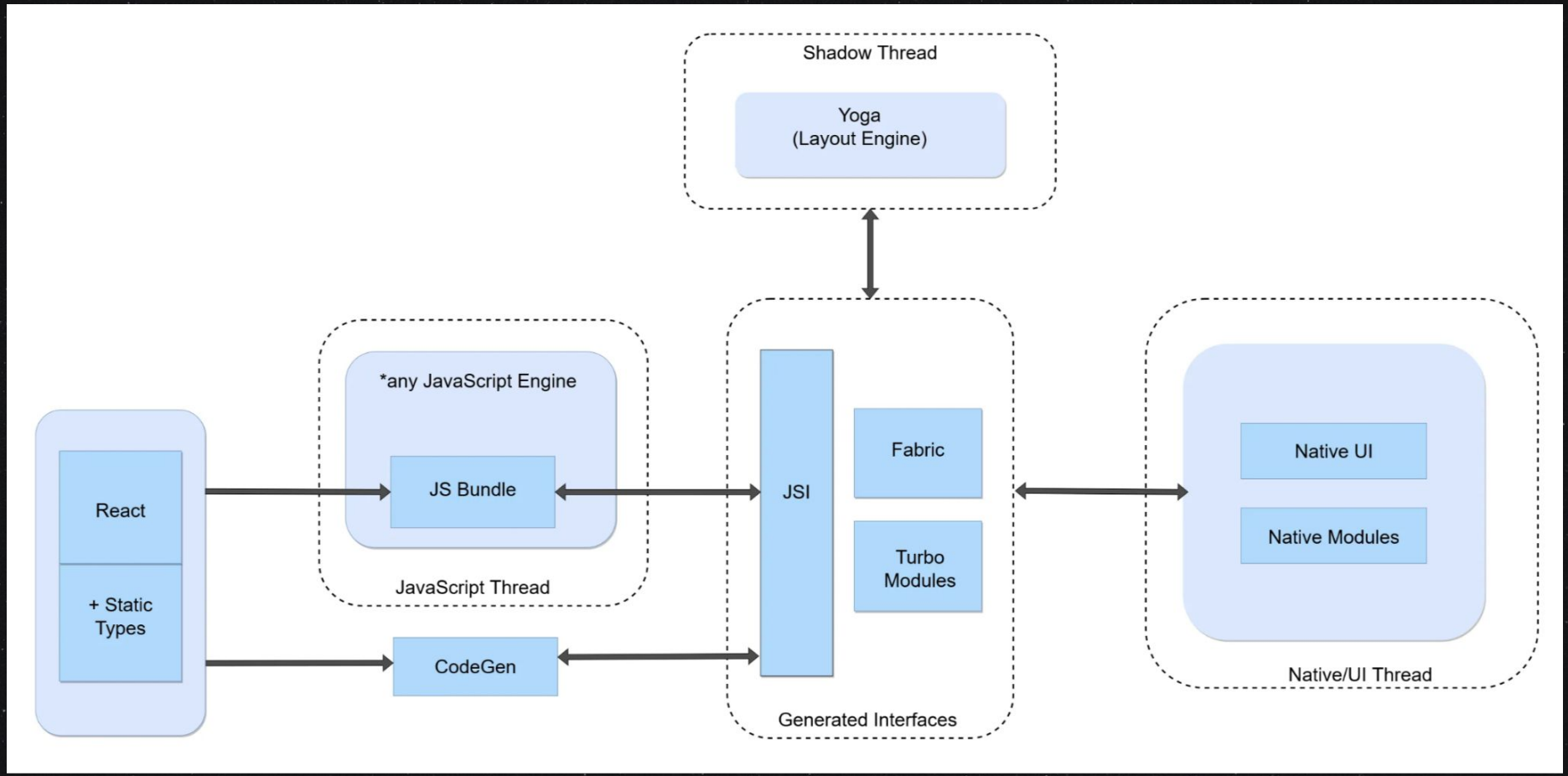
Using JavaScript and React, you can make apps for both iOS and Android

You write in TypeScript or JavaScript using JSX and React, which runs on a JavaScript engine, communicates asynchronously with native APIs via a bridge, and ultimately renders into native UI components like UIViews.

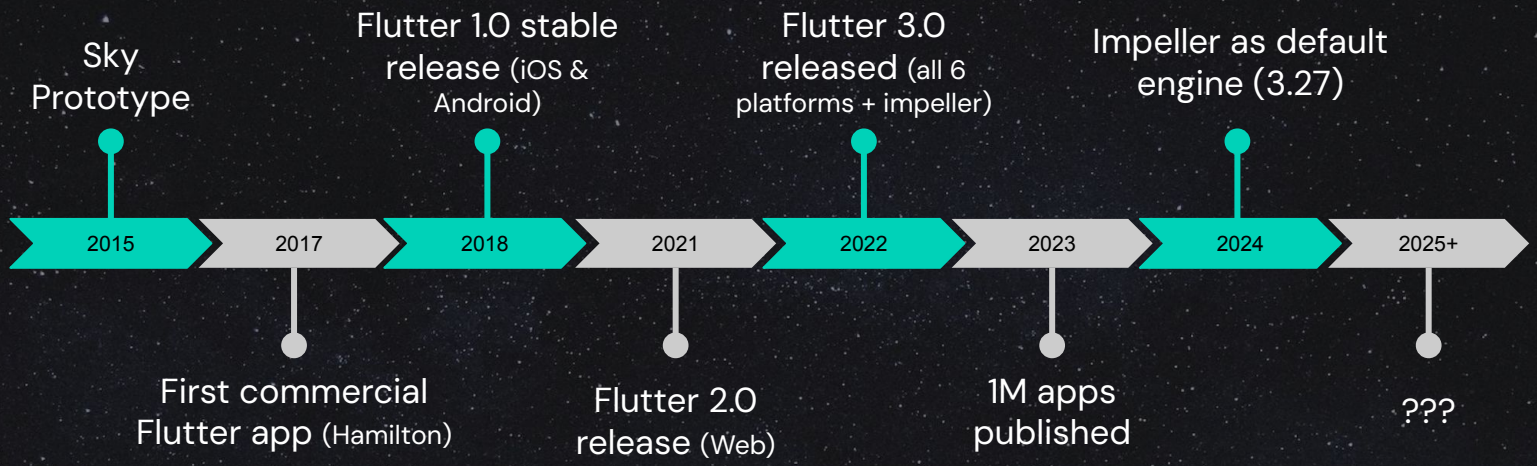
Three main threads: the JS thread, the Native UI thread, and the Shadow thread. Data sent across the bridge must be batched and serialized as JSON.
The Shadow thread is responsible for computing layout using the Yoga engine.

https://medium.com/coox-tech/deep-dive-into-react-natives-new-architecture-fb67ae615ccd




Shadow Thread
Yoga
(Layout Engine)
*any JavaScript Engine
JS Bundle
JavaScript Thread
React
+ Static
Types
CodeGen
JSI
Fabric
Turbo
Modules
Generated Interfaces
Native UI
Native Modules
Native/UI Thread


Flutter



Google Play / Earth / Classroom / Cloud / Analytics LG Electronics Xiaomi

Whirlpool Universal Studios Alibaba eBay BMW iRobot Aboutyou

Language: Dart

```dart
int countPrimes(int n) {
 if (n <= 1) return 0;
 int count = 0;

 for (int number = 2; number <= n; number++) {
   bool isPrime = true;
   int maxDivisor = sqrt(number.toDouble()).toInt();
   if (maxDivisor >= 2) {
     for (int divisor = 2; divisor <= maxDivisor; divisor++) {
       if (number % divisor == 0) {
         isPrime = false;
         break;
       }
     }
   }
   if (isPrime) {
     count++;
   }
 }
 return count;
}
```

```swift
static func countPrimes(upTo n: Int) async -> Int {
    if n <= 1 { return 0 }
    var count = 0

    for number in 2...n {
        var isPrime = true
        let maxDivisor = Int(Double(number).squareRoot())
        if maxDivisor >= 2 {
            for divisor in 2...maxDivisor {
                if number % divisor == 0 {
                    isPrime = false
                    break
                }
            }
        }
        if isPrime {
            count += 1
        }
    }
    return count
}
```

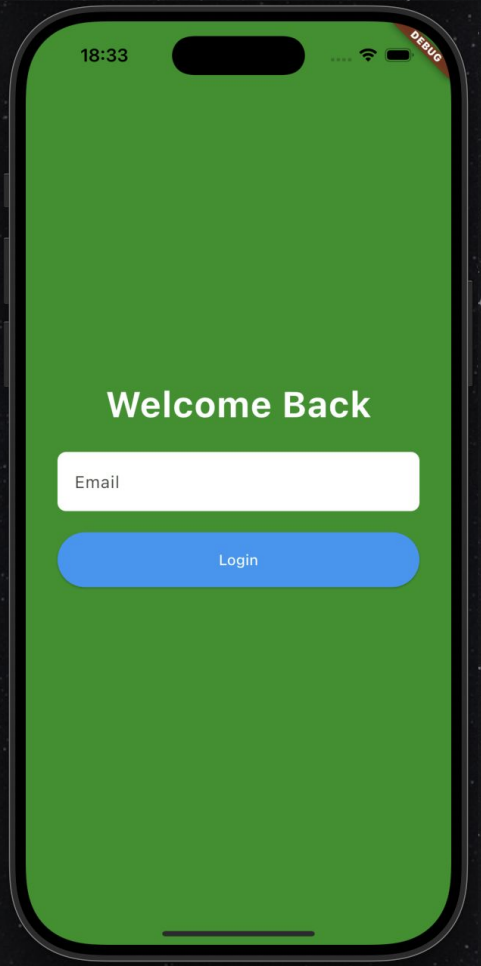
UI: Widgets

```dart
import 'package:flutter/material.dart';

class LoginView extends StatefulWidget {
  const LoginView({Key? key}) : super(key: key);

  @override
  State<LoginView> createState() => _LoginViewState();
}

class _LoginViewState extends State<LoginView> {
  String email = '';

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.green,
      body: Padding(
        padding: const EdgeInsets.symmetric(horizontal: 30),
        child: Center(
          child: Column(
            mainAxisSize: MainAxisSize.min,
            crossAxisAlignment: CrossAxisAlignment.stretch,
            children: [
              const Text(
                "Welcome Back",
                textAlign: TextAlign.center,
                style: TextStyle(
                  fontSize: 34,
                  color: Colors.white,
                  fontWeight: FontWeight.bold,
                ), // TextStyle
              ), // Text
              const SizedBox(height: 20),
              TextField(
                onChanged: (value) => setState(() => email = value),
                decoration: const InputDecoration(
                  hintText: "Email",
                  filled: true,
                  fillColor: Colors.white,
                  contentPadding: EdgeInsets.all(16),
                  border: OutlineInputBorder(
                    borderSide: BorderSide.none,
                    borderRadius: BorderRadius.all(Radius.circular(8)),
                  ), // OutlineInputBorder
                ), // InputDecoration
              ), // TextField
              const SizedBox(height: 20),
              ElevatedButton(
                onPressed: () {
                  // TODO: Login action
                },
                style: ElevatedButton.styleFrom(
                  backgroundColor: Colors.blue,
                  padding: const EdgeInsets.all(16),
                ),
                child: const Text("Login", style: TextStyle(color: Colors.white)),
              ), // ElevatedButton
            ],
          ), // Column
        ), // Center
      ), // Padding
    ); // Scaffold
  }
}
```

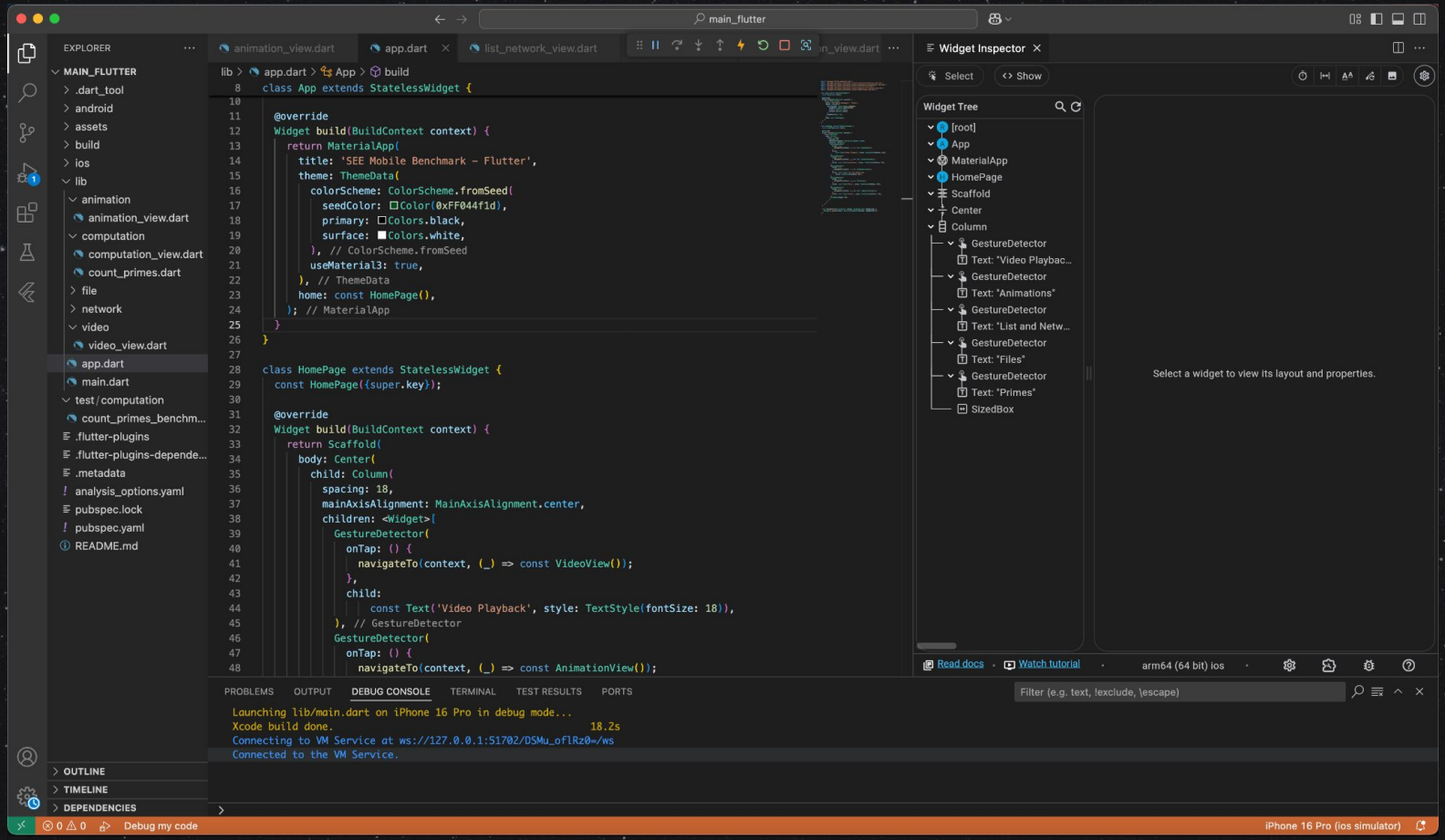
```swift
struct LoginView: View {
    @State private var email: String = ""

    var body: some View {
        VStack(spacing: 20) {
            Spacer()

            Text("Welcome Back")
                .font(.largeTitle)
                .foregroundColor(.white)

            TextField("Email", text: $email)
                .padding()
                .background(Color.white)
                .padding(.horizontal, 30)

            Button(action: {
                // ToDo: Login
            }) {
                Text("Login")
                    .frame(maxWidth: .infinity)
                    .padding()
                    .foregroundColor(.white)
                    .background(Color.blue)
            }
            .padding(.horizontal, 30)
            .padding(.top, 10)

            Spacer()
        }
        .background(.green)
    }
}
```

UI: Widgets

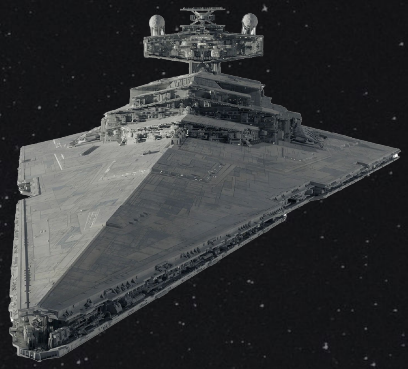
```
children: [
  const Text(
    "Welcome Back",
    textAlign: TextAlign.center,
    style: TextStyle(
      fontSize: 34,
      color: Colors.white,
      fontWeight: FontWeight.bold,
    ), // TextStyle
  ), // Text
  const SizedBox(height: 20),
  TextField(
    onChanged: (value) => setState(() => email = value),
    decoration: const InputDecoration(
      hintText: "Email",
      filled: true,
      fillColor: Colors.white,
      contentPadding: EdgeInsets.all(16),
      border: OutlineInputBorder(
        borderSide: BorderSide.none,
        borderRadius: BorderRadius.all(Radius.circular(8)),
      ), // OutlineInputBorder
    ), // InputDecoration
  ), // TextField
  const SizedBox(height: 20),
  ElevatedButton(
    onPressed: () {
      // TODO: Login action
    },
    style: ElevatedButton.styleFrom(
      backgroundColor: Colors.blue,
      padding: const EdgeInsets.all(16),
    ),
    child: const Text("Login", style: TextStyle(color: Colors.white)),
  ), // ElevatedButton
```



IDE: Visual Studio Code



How does it work?



Dart gets compiled JIT during development (Hot Reload).

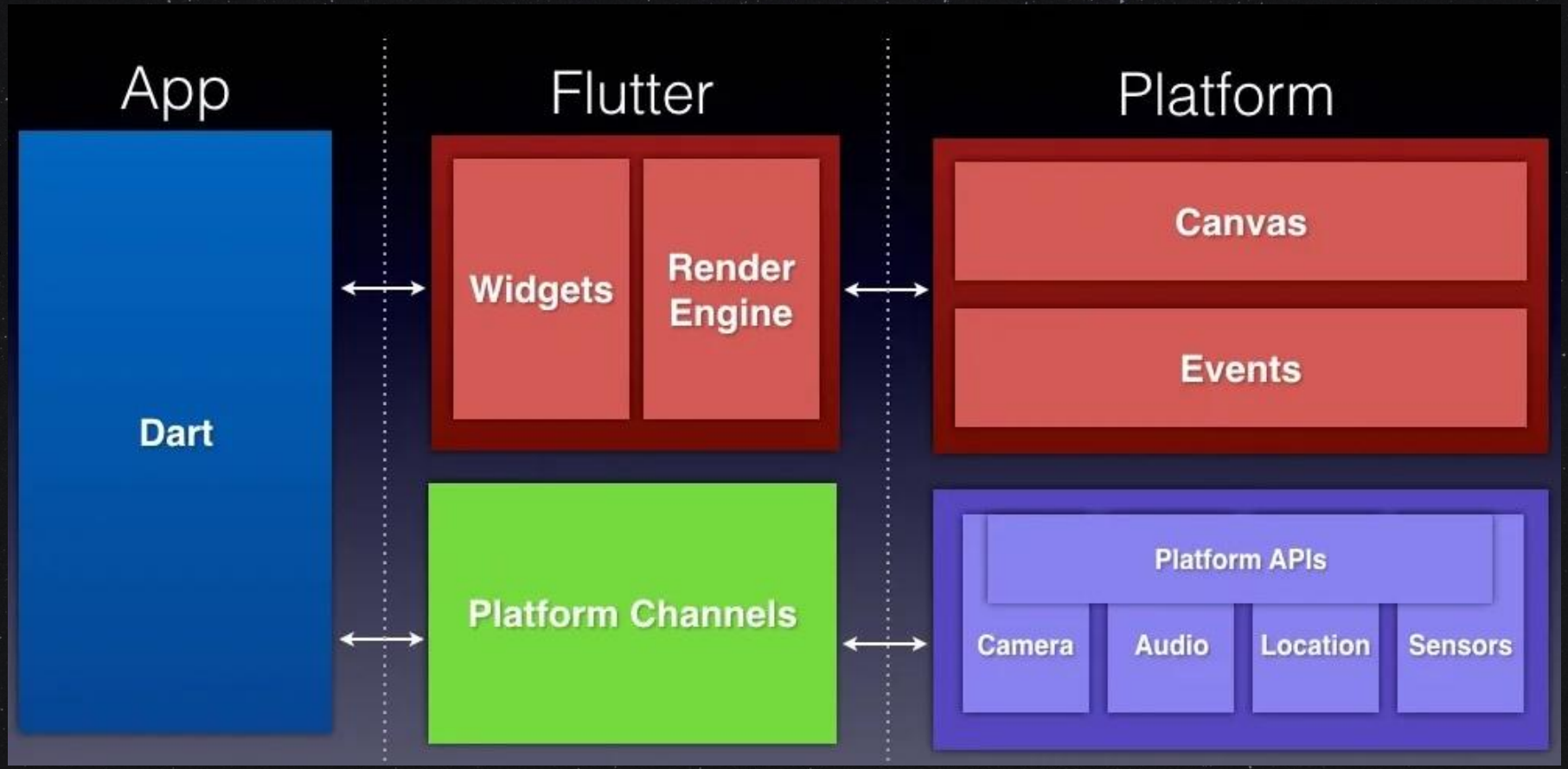
Dart gets compiled AOT for production into native machine code (ARM).

Flutter framework consists of various dart libraries (animations, gestures, rendering etc).

UI is done with declarative widgets, structured later in a widget tree.

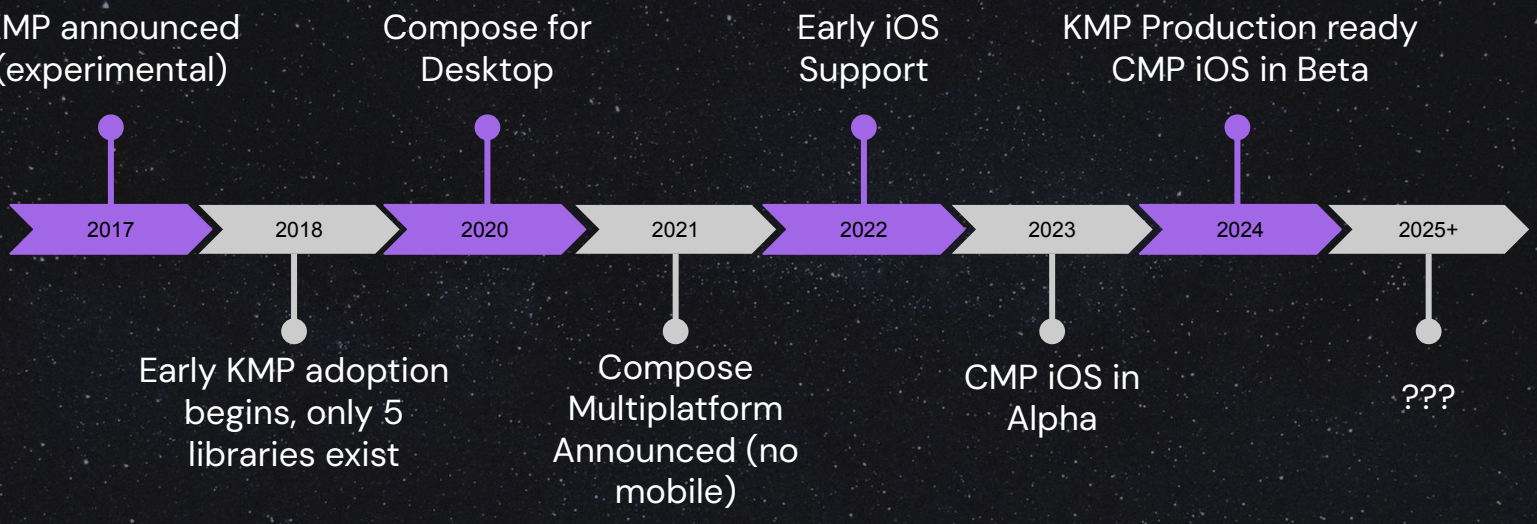
The Flutter engine (written in C++) handles UI rendering using the Impeller graphics engine (directly on the GPU) and provides a runtime environment for Dart and Flutter frameworks.

Platform channels are used as a very fast bidirectional messaging system between Dart and native code.


App
Flutter
Platform
Dart
Widgets
Render Engine
Canvas
Events
Platform Channels
Platform APIs
Camera
Audio
Location
Sensors

Business logic in Kotlin, Native UI

Business logic in Kotlin, UI in JetpackCompose (smells like Flutter)



Forbes

NETFLIX

McDonald's

9GAG

Cash App

Meetup

Compose Multiplatform

wrike

Compose Multiplatform

UI: Jetpack Compose

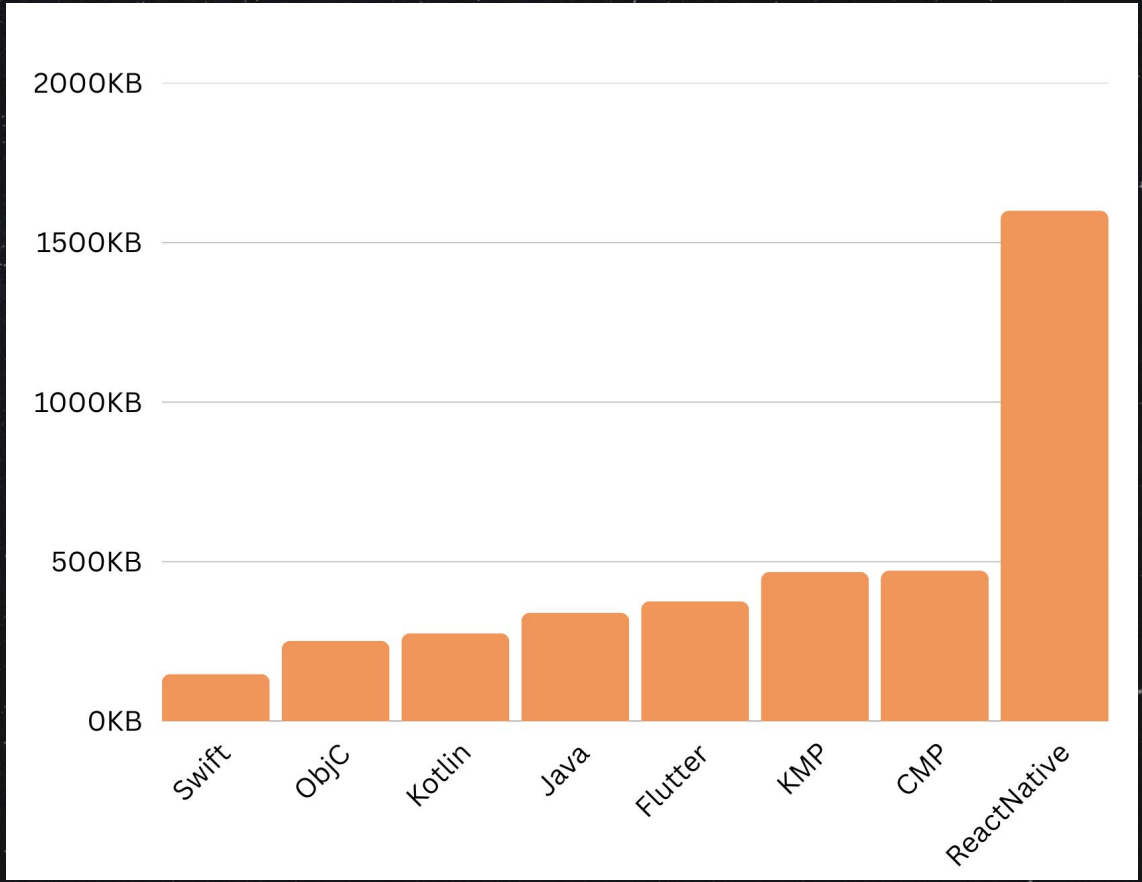
```
Column(
    modifier = Modifier
        .fillMaxWidth()
        .padding(horizontal = 32.dp),
    verticalArrangement = Arrangement.spacedBy(16.dp),
    horizontalAlignment = Alignment.CenterHorizontally
) {
    Text(
        text = "Welcome Back",
        style = MaterialTheme.typography.h4,
        color = Color.White
    )

    TextField(
        value = email,
        onValueChange = { email = it },
        placeholder = { Text( text: "Email") },
        colors = TextFieldDefaults.textFieldColors(
            backgroundColor = Color.White,
            focusedIndicatorColor = Color.Transparent,
            unfocusedIndicatorColor = Color.Transparent
        ),
        modifier = Modifier.fillMaxWidth()
    )

    Button(
        onClick = { /* TODO: Handle Login */ },
        modifier = Modifier.fillMaxWidth(),
        colors = ButtonDefaults.buttonColors(backgroundColor = Color.Blue)
    ) {
        Text(text = "Login", color = Color.White)
    }
}
```

```
struct LoginView: View {
    @State private var email: String = ""

    var body: some View {
        VStack(spacing: 20) {
            Spacer()

            Text("Welcome Back")
                .font(.largeTitle)
                .foregroundColor(.white)

            TextField("Email", text: $email)
                .padding()
                .background(Color.white)
                .padding(.horizontal, 30)

            Button(action: {
                // ToDo: Login
            }) {
                Text("Login")
                    .frame(maxWidth: .infinity)
                    .padding()
                    .foregroundColor(.white)
                    .background(Color.blue)
            }
            .padding(.horizontal, 30)
            .padding(.top, 10)

            Spacer()
        }
        .background(.green)
    }
}
```

RESULTS

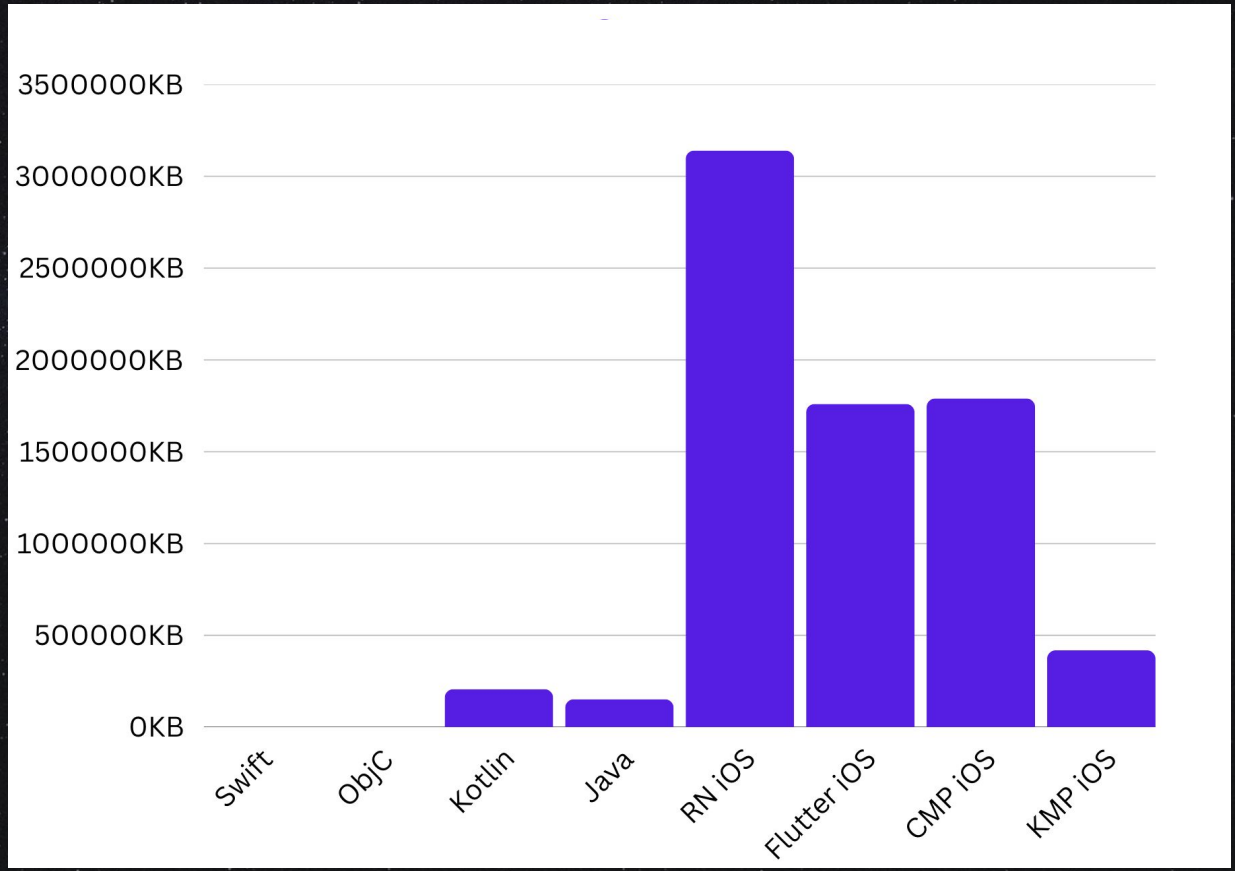
GitHub repo size

Swift	148KB
ObjC	252KB
Kotlin	276KB
Java	340KB
ReactNative	1600KB
Flutter	376KB
CMP	472KB
KMP	468KB



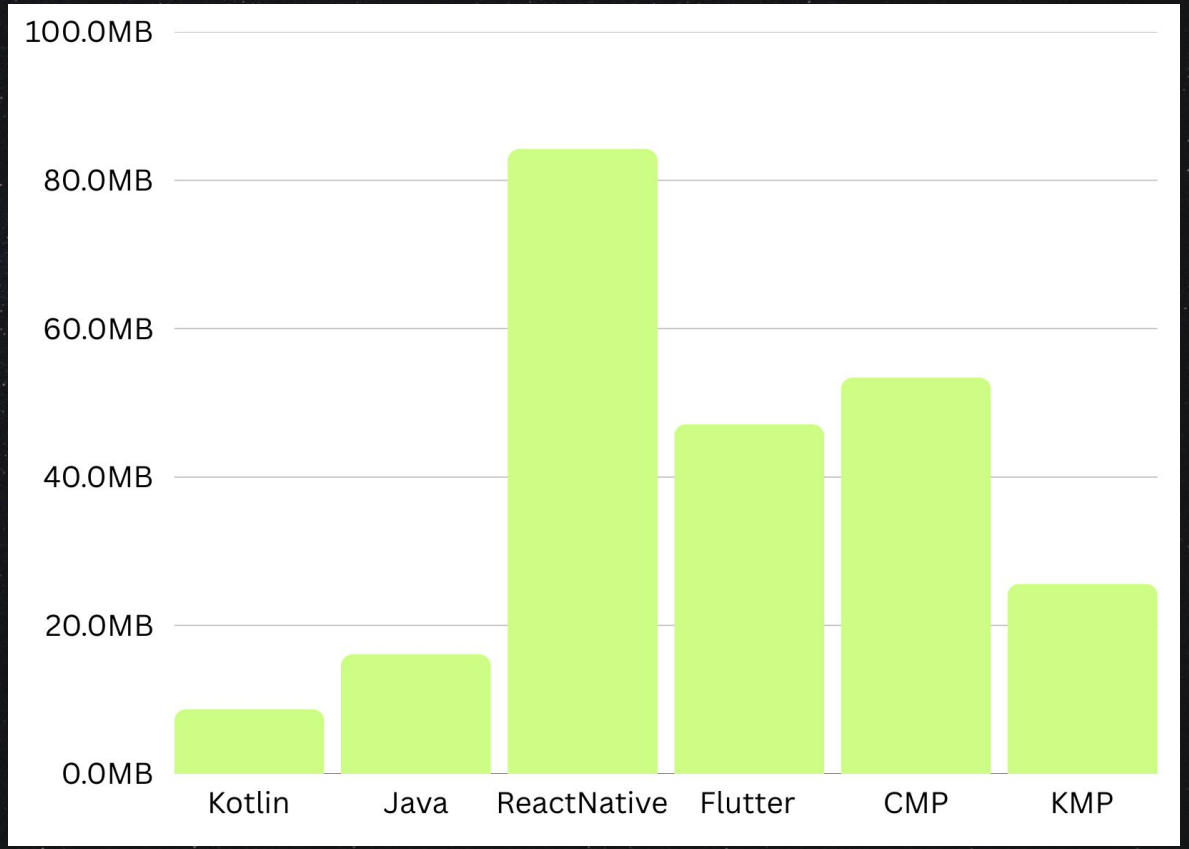
Codebase size

Swift	151KB
ObjC	256KB
Kotlin	206.7MB
Java	151.8MB
RN	3.14GB
Flutter	1.76GB
CMP	1.79GB
KMP	418.5MB



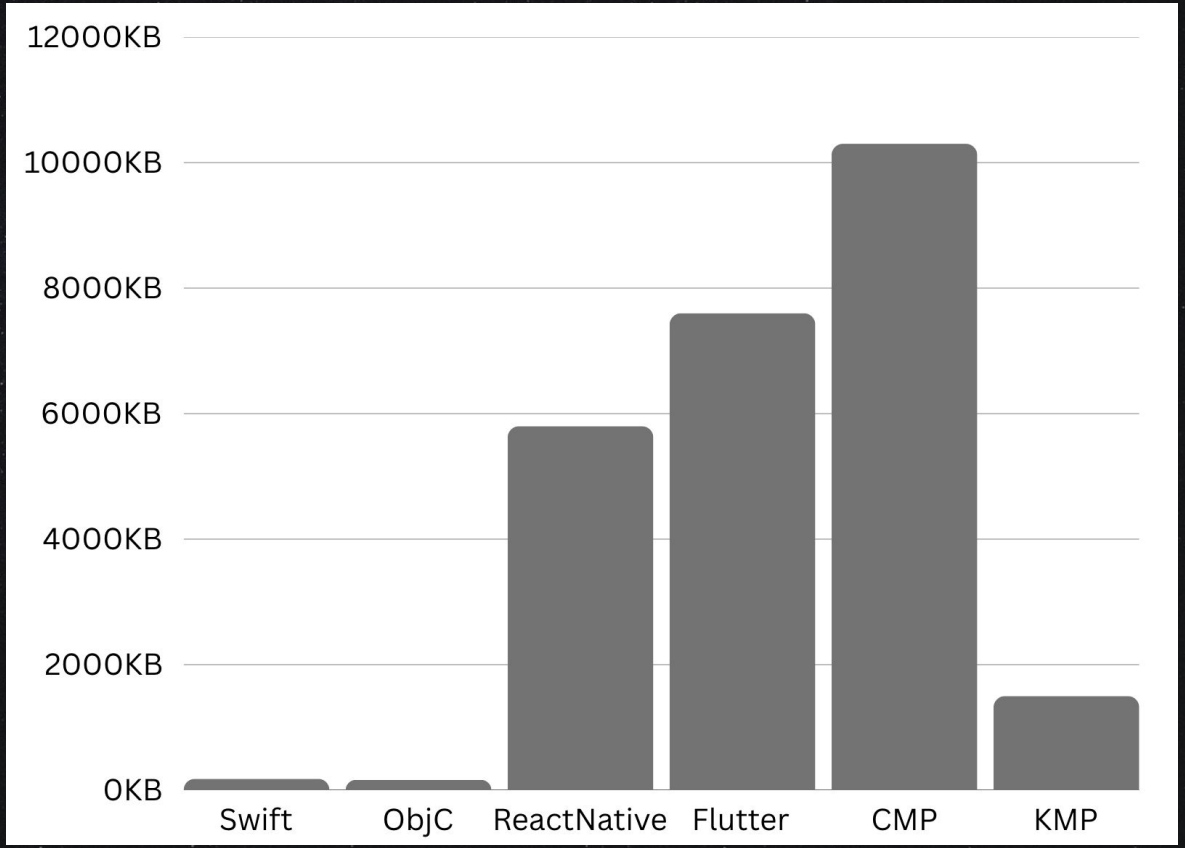
apk size

Kotlin	8.7MB
Java	16.1MB
ReactNative	84.2MB
Flutter	47.1MB
CMP	53.4MB
KMP	25.6MB



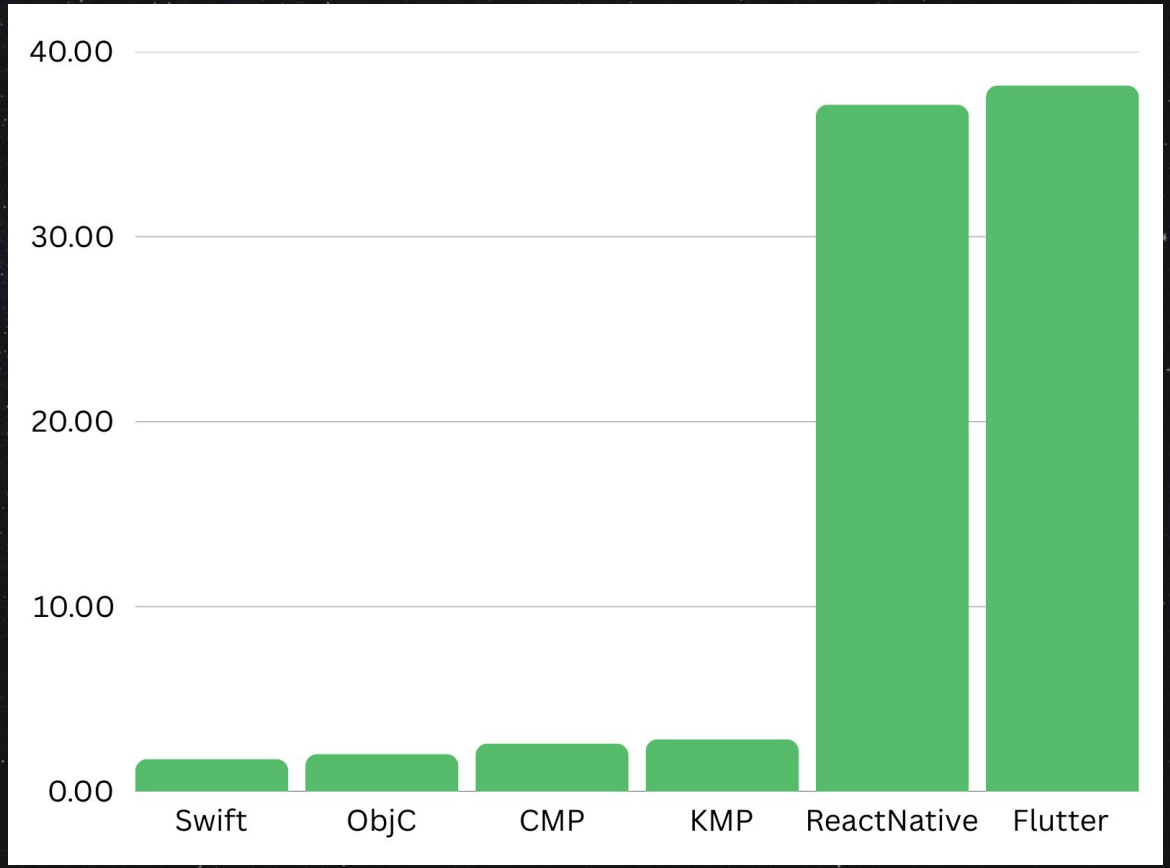
ipa size

Swift	181KB
ObjC	164KB
ReactNative	5800KB
Flutter	7600KB
CMP	10300KB
KMP	1500KB



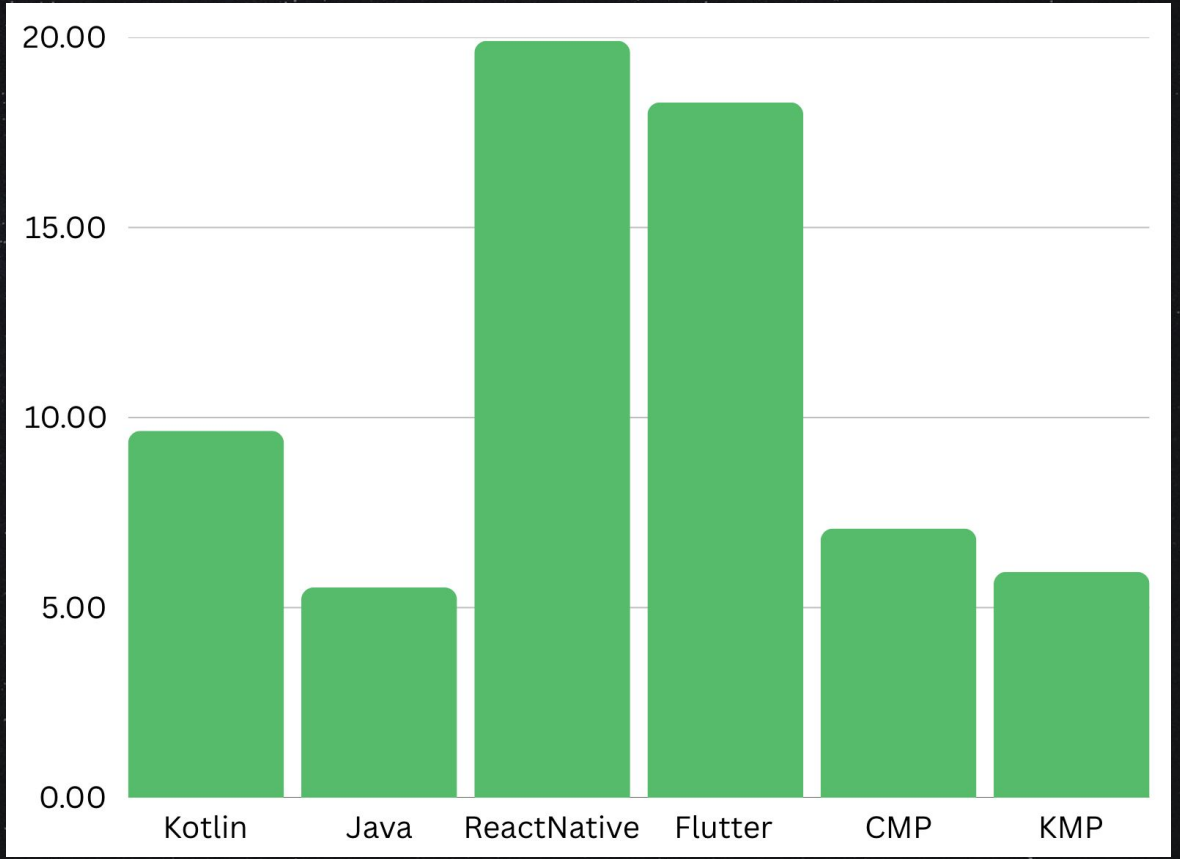
Build and run time

Swift	1.76
ObjC	2.03
ReactNative	37.14
Flutter	38.18
CMP	2.6
KMP	2.83



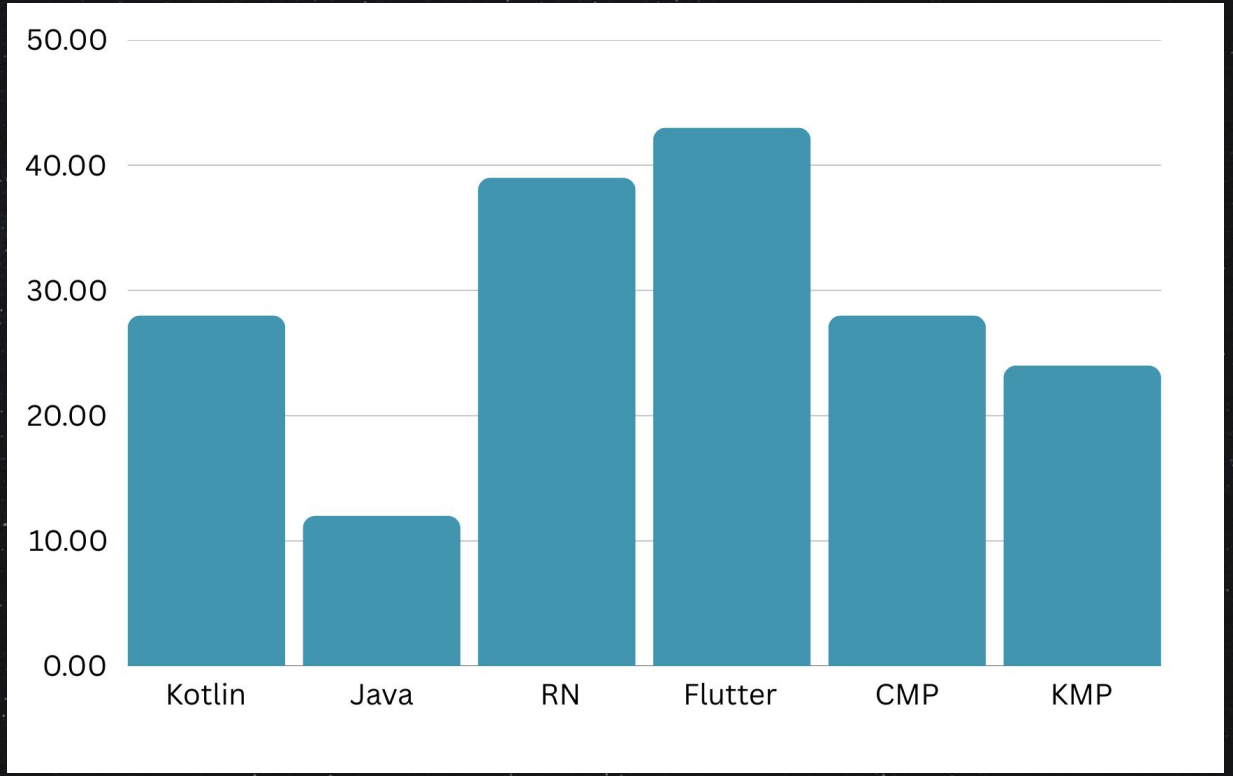
Build and run time

Kotlin	9.65
Java	5.53
ReactNative	19.91
Flutter	18.29
CMP	7.08
KMP	5.94



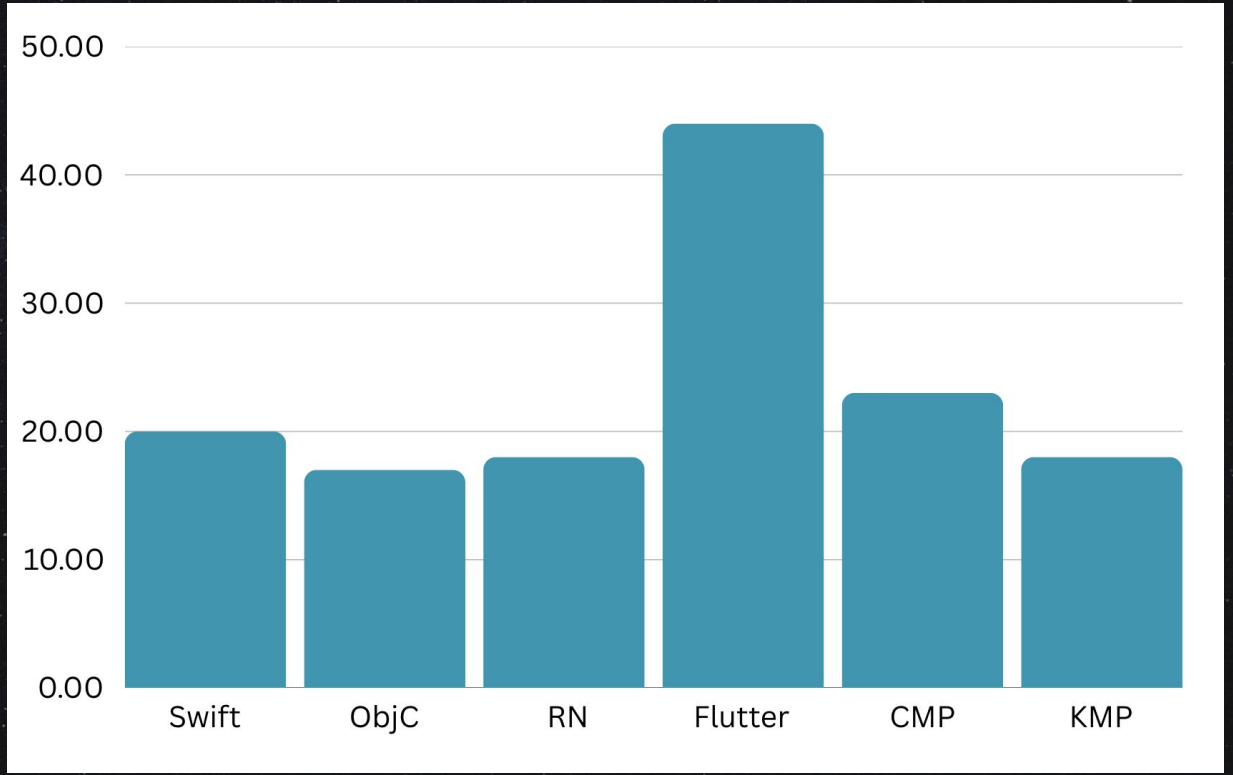
Android Launch time

Kotlin	28
Java	12
ReactNative	39
Flutter	43
CMP	28
KMP	24

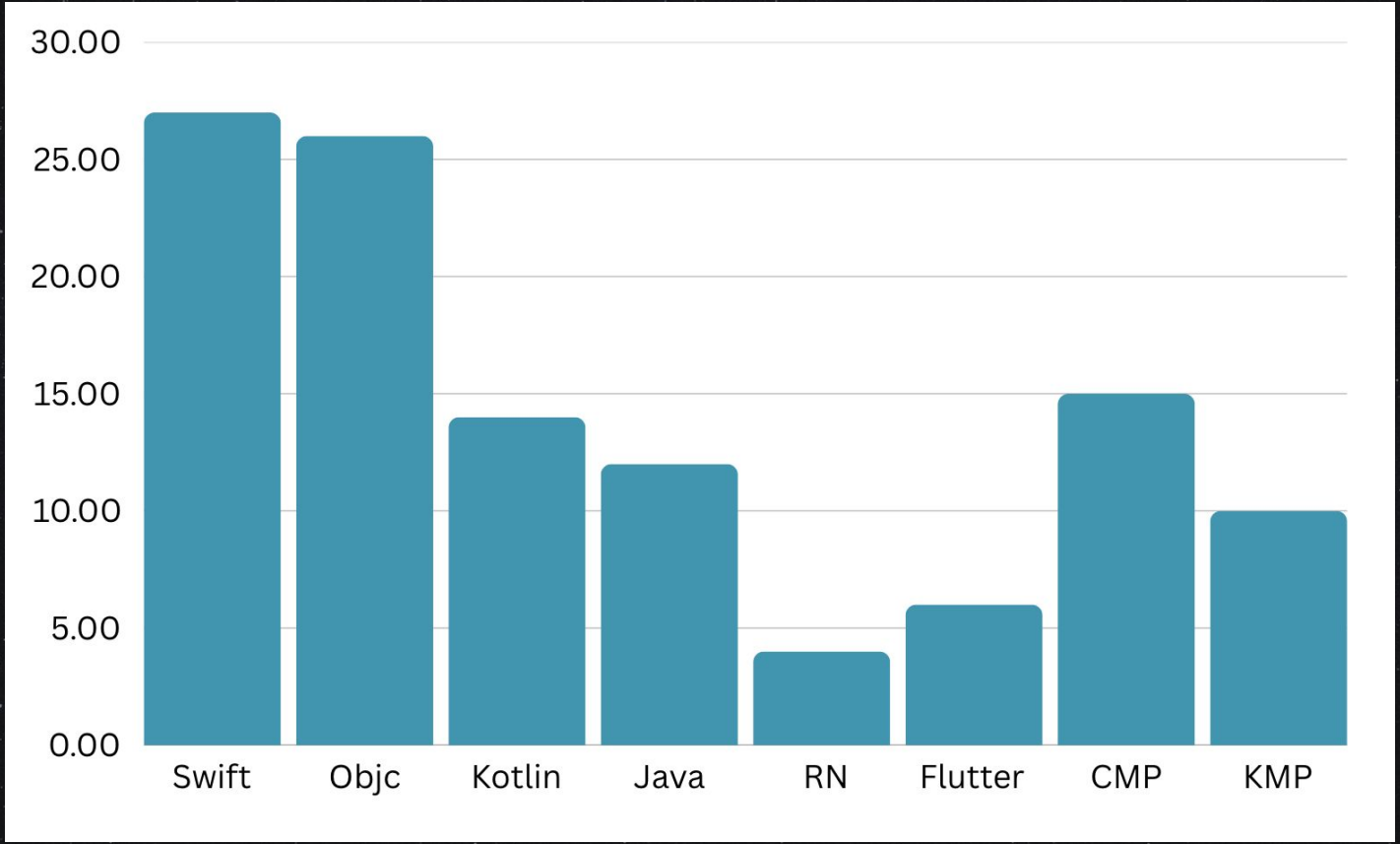


iOS Launch time

Swift	20
ObjC	17
ReactNative	18
Flutter	44
CMP	23
KMP	18

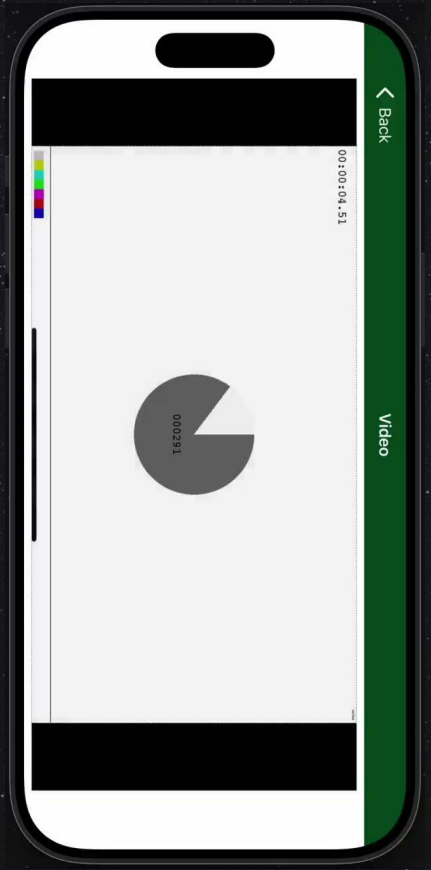


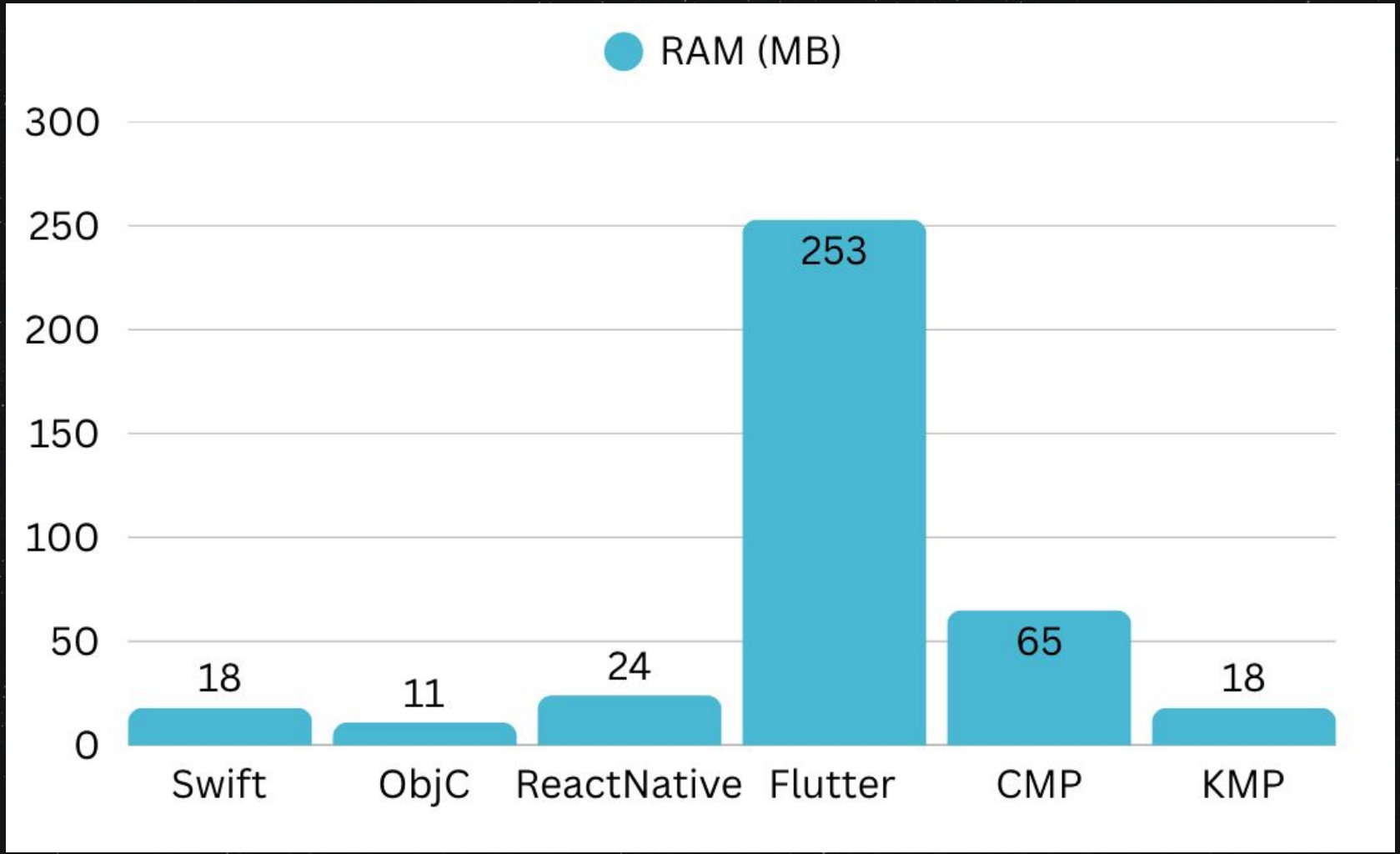
Tests time

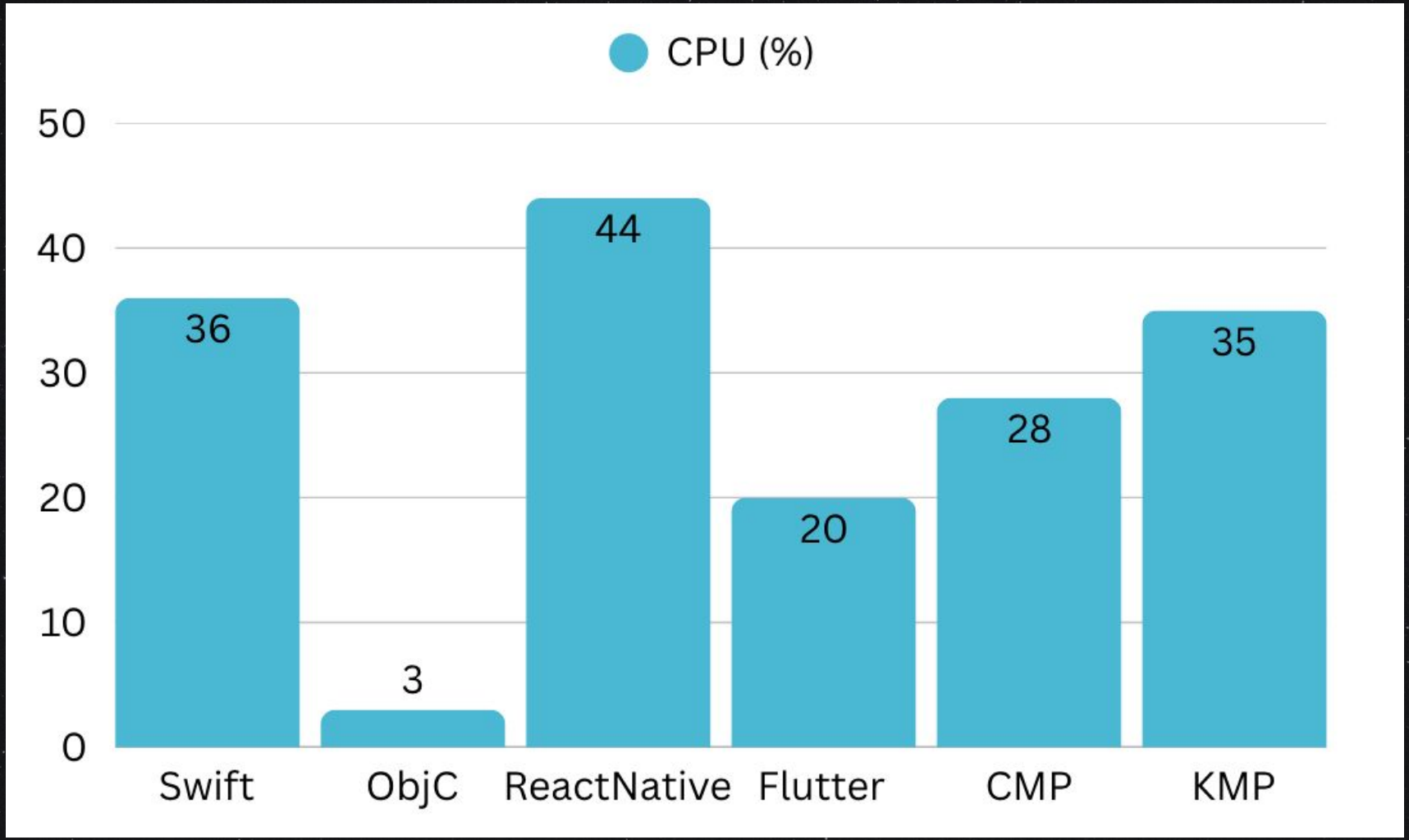

30.00
25.00
20.00
15.00
10.00
5.00
0.00
Swift
Objc
Kotlin
Java
RN
Flutter
CMP
KMP

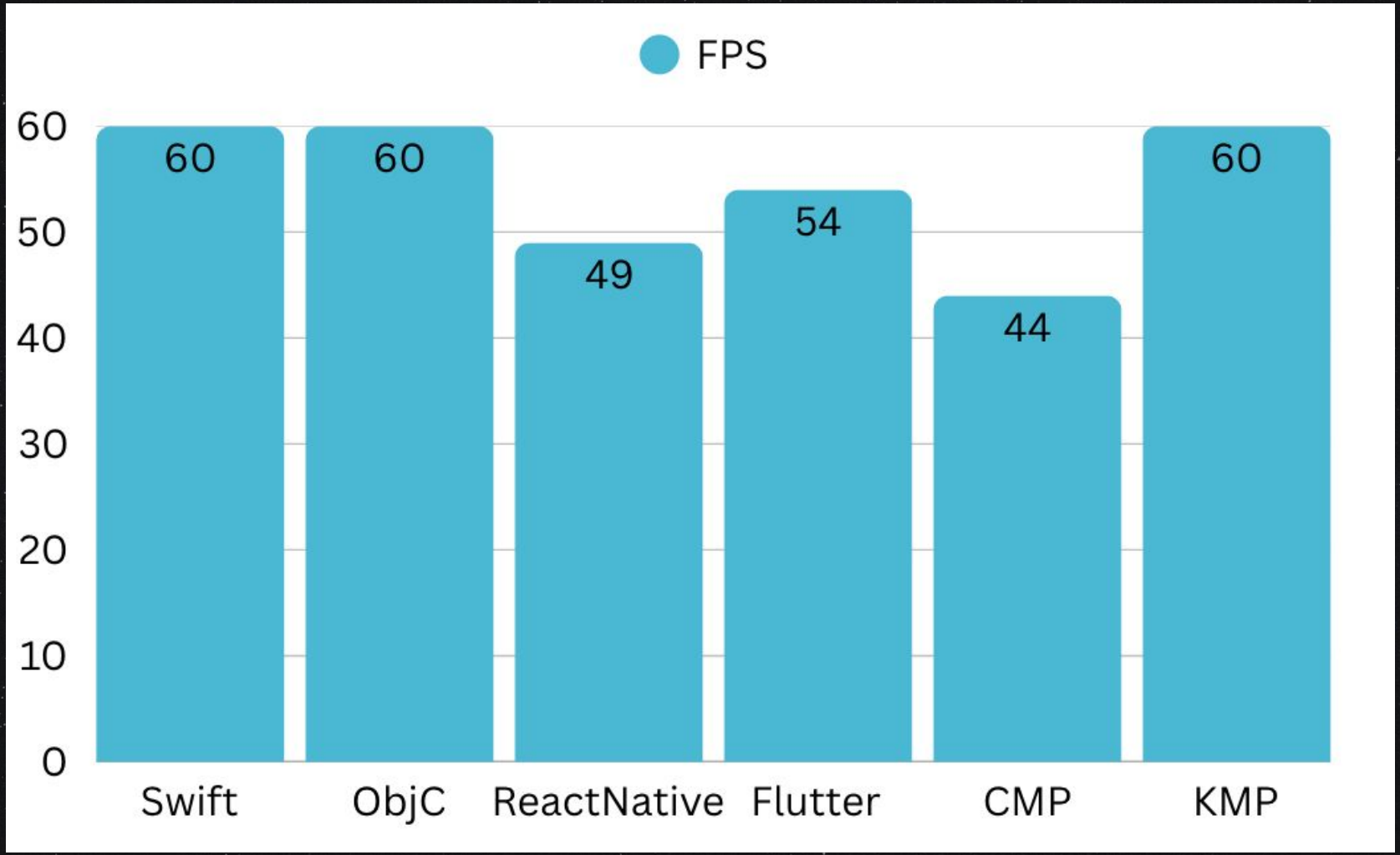
Video

- 60fps video
- Full HD
- Landscape only
- .m3u8
- Custom video layer



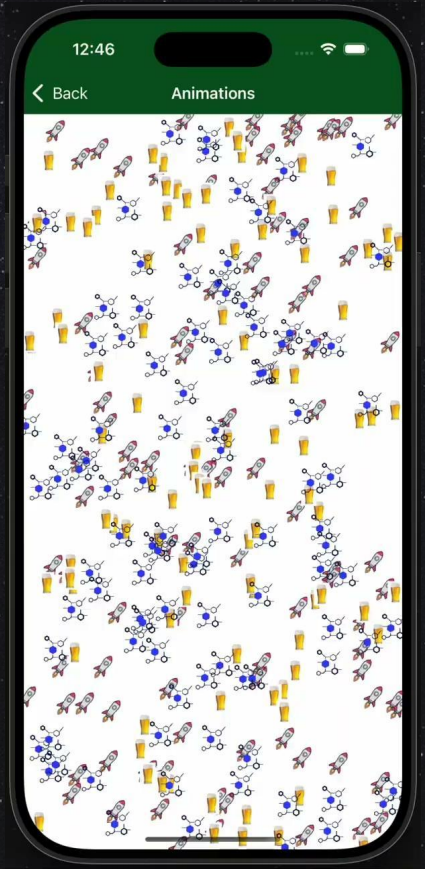

RAM (MB)
300
250
200
150
100
50
0
18
11
24
253
65
18
Swift
ObjC
ReactNative
Flutter
CMP
KMP

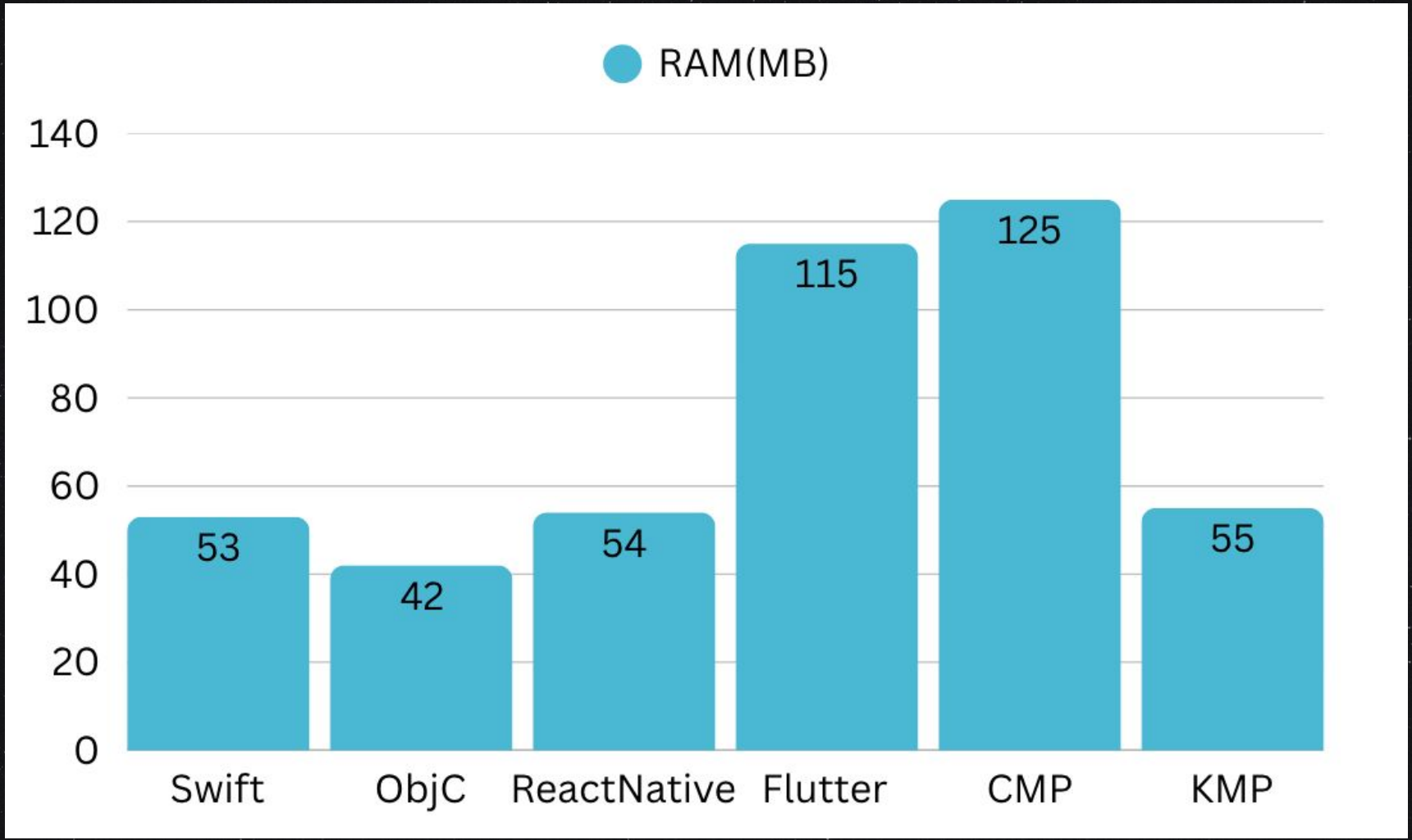

CPU (%)
50
40
30
20
10
0
36
3
44
20
28
35
Swift
ObjC
ReactNative
Flutter
CMP
KMP

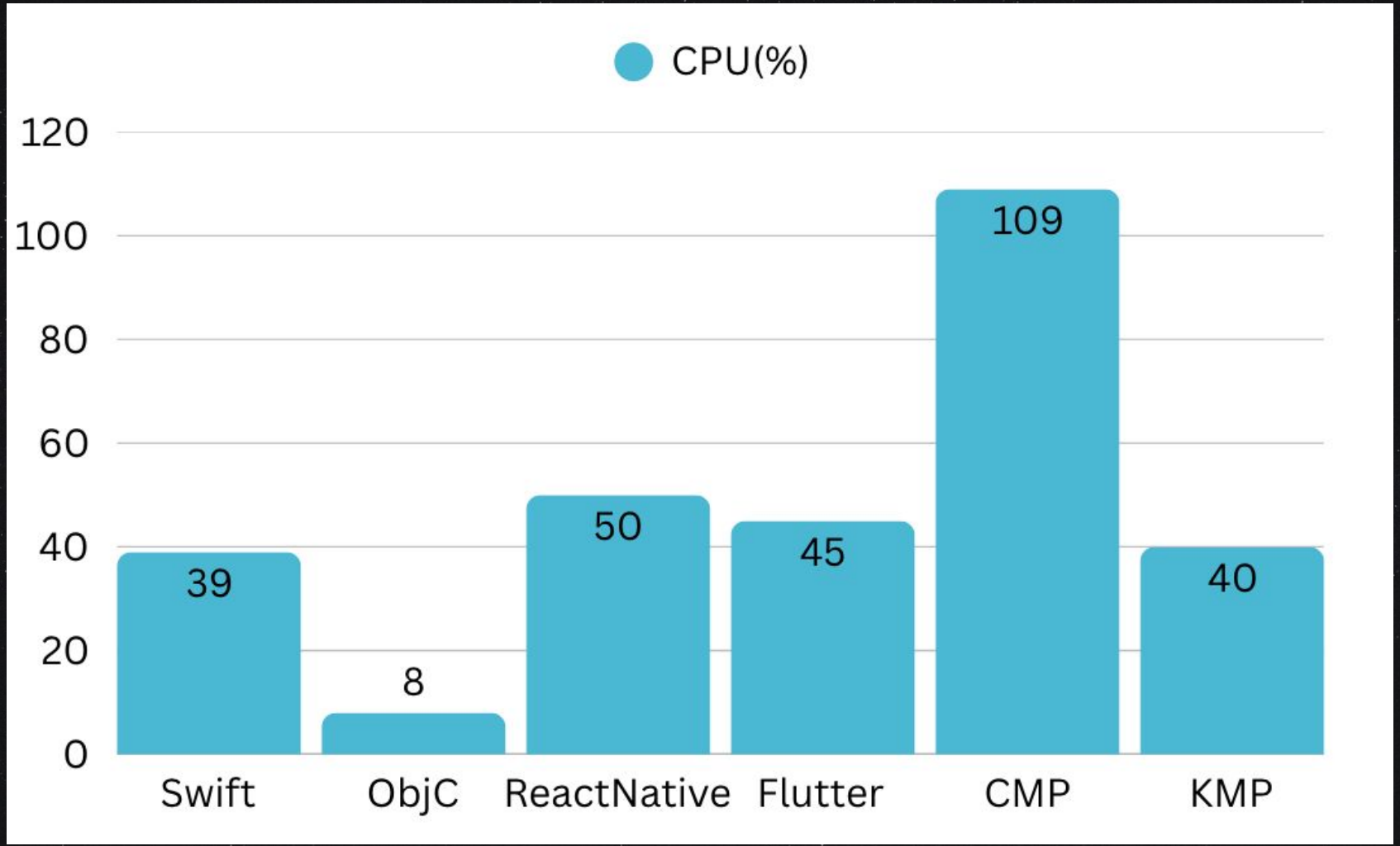

FPS
60
50
40
30
20
10
0
60
60
49
54
44
60
Swift
ObjC
ReactNative
Flutter
CMP
KMP

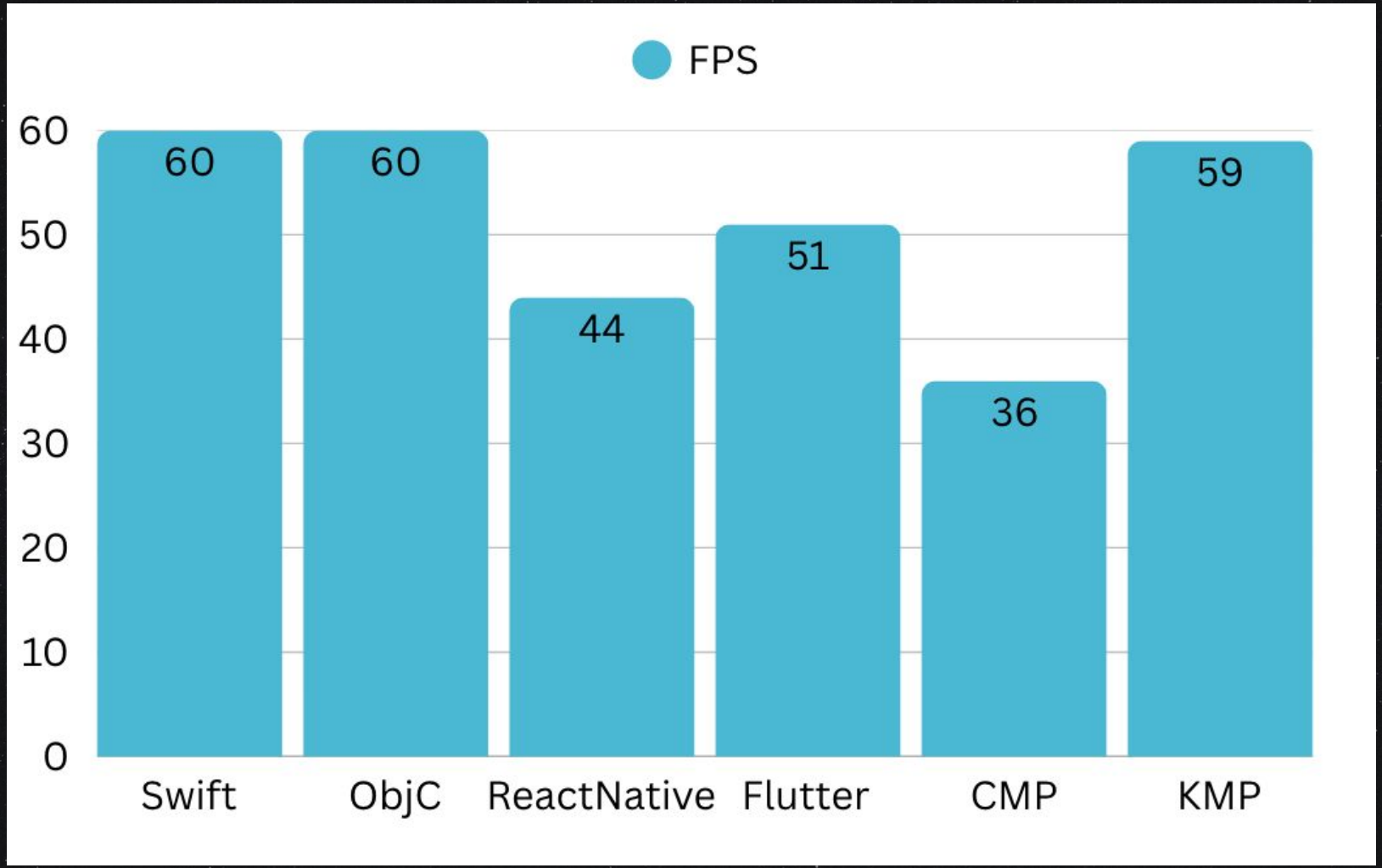
Animations

- 3 images (png, jpg, svg)
- 3 infinite animations
  - fade
  - scale
  - rotate
- x100 per type



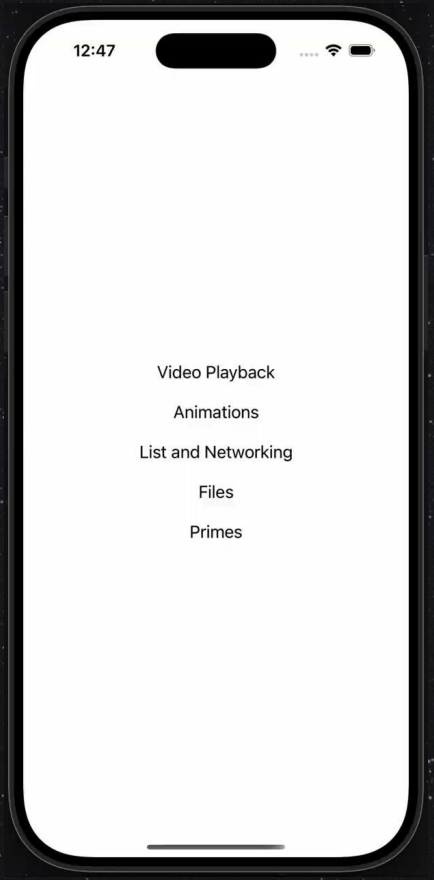

RAM(MB)
140
120
100
80
60
40
20
0
53
42
54
115
125
55
Swift
ObjC
ReactNative
Flutter
CMP
KMP

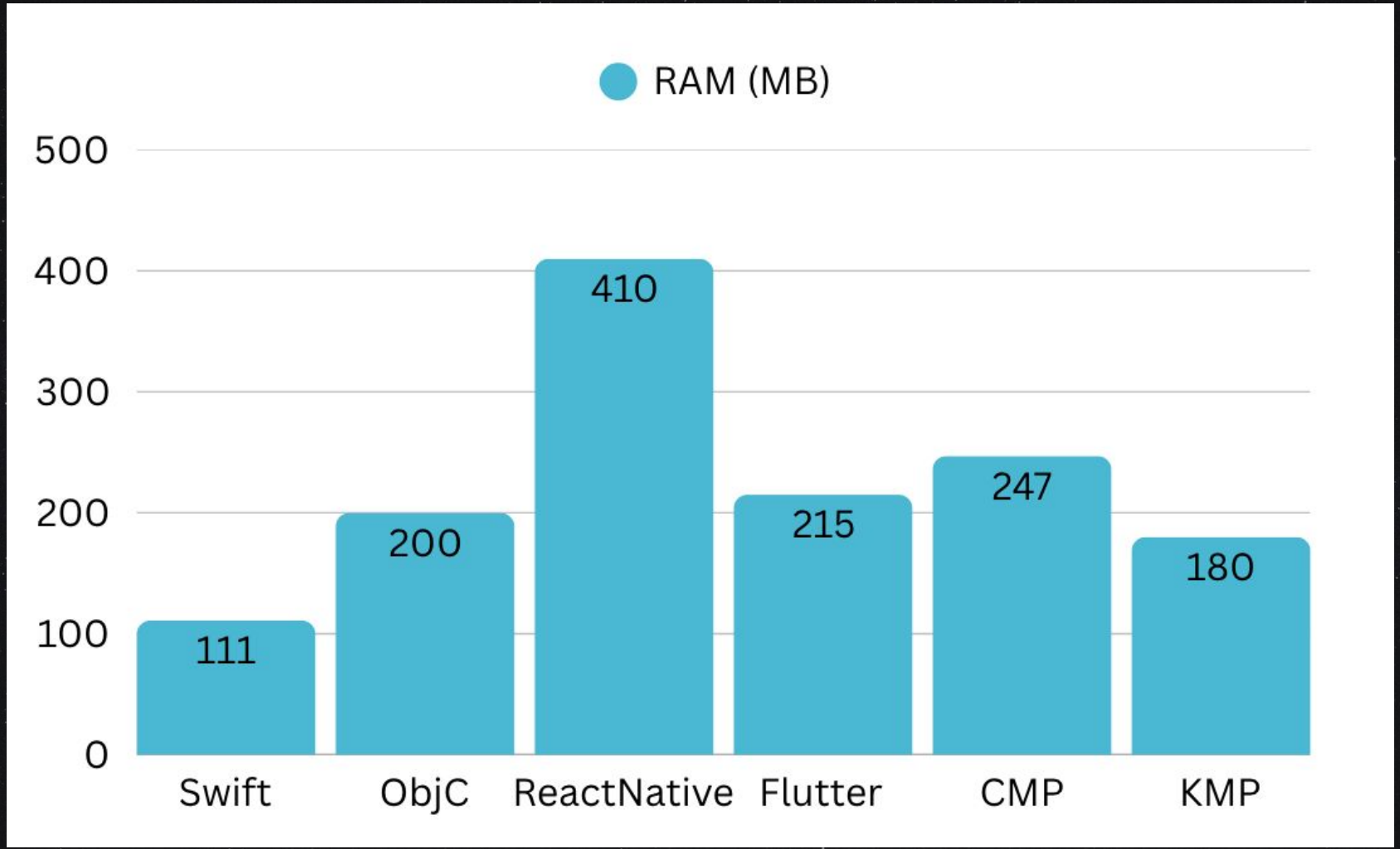

CPU(%)
120
100
80
60
40
20
0
39
8
50
45
109
40
Swift
ObjC
ReactNative
Flutter
CMP
KMP

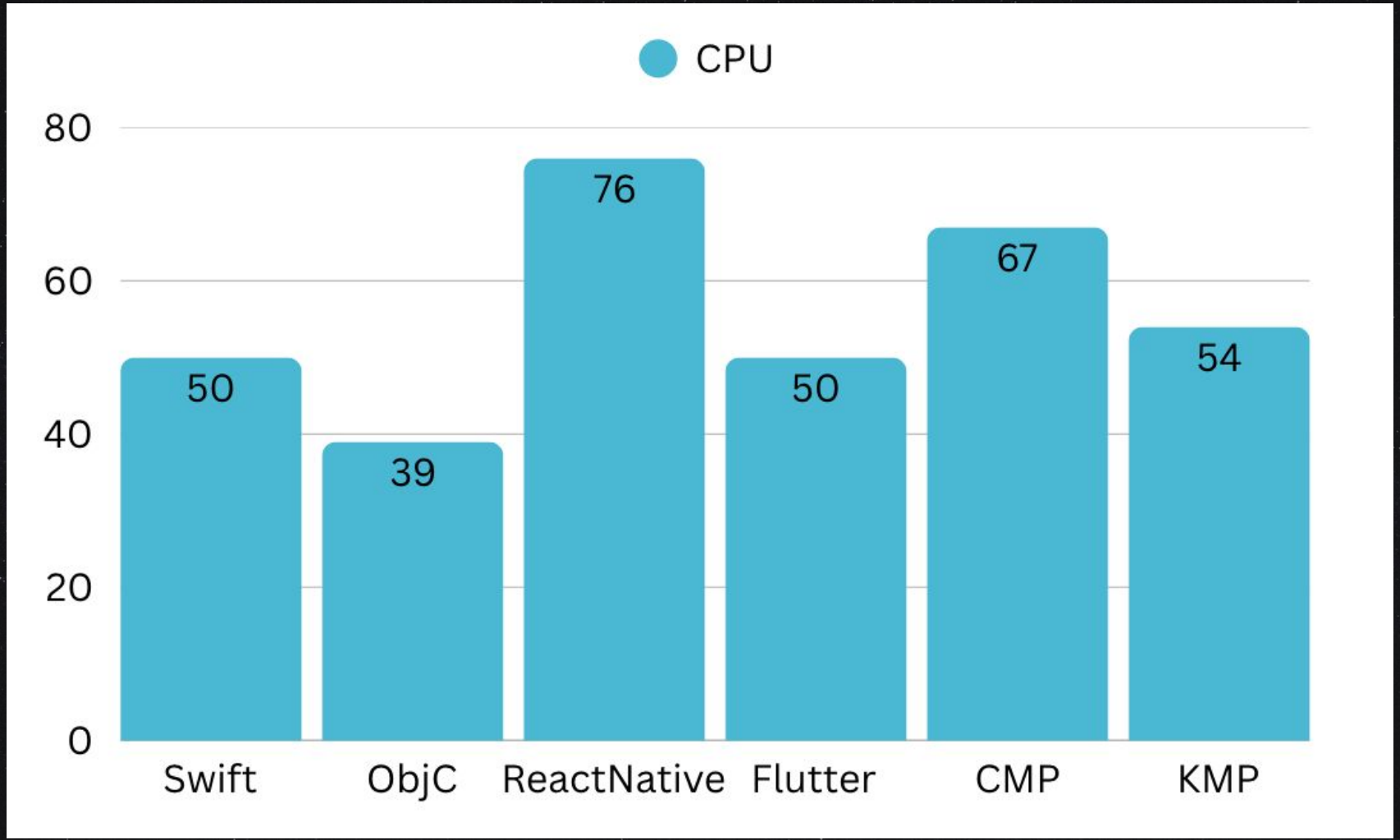

FPS
60
50
40
30
20
10
0
60
60
44
51
36
59
Swift
ObjC
ReactNative
Flutter
CMP
KMP

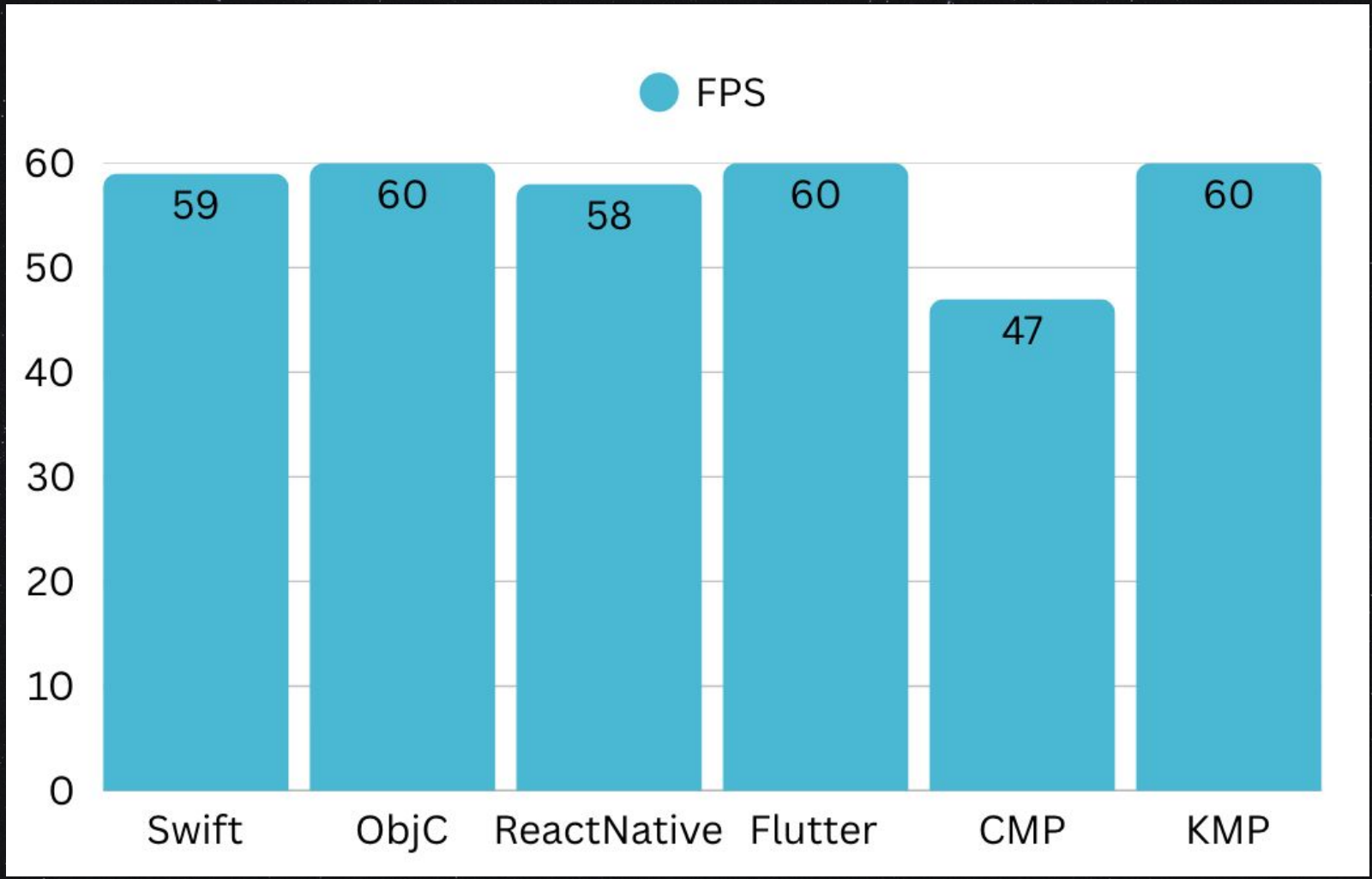
List

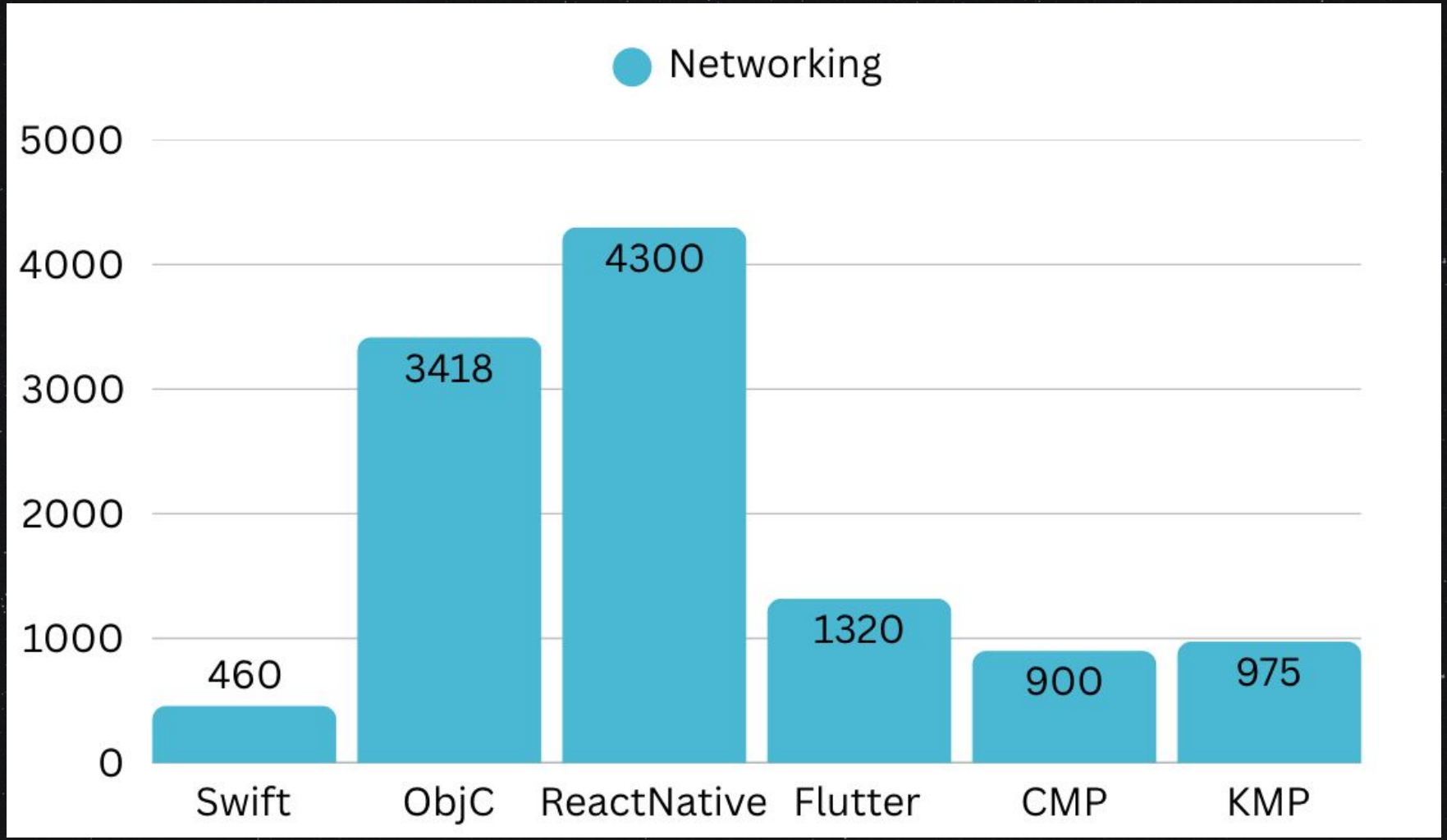
- GET request to a 26 MB JSON
- Serialize in an array (12000)
- Perform sort and filter (6000)
- List with remote image and title




RAM (MB)
500
400
300
200
100
0
111
200
410
215
247
180
Swift
ObjC
ReactNative
Flutter
CMP
KMP

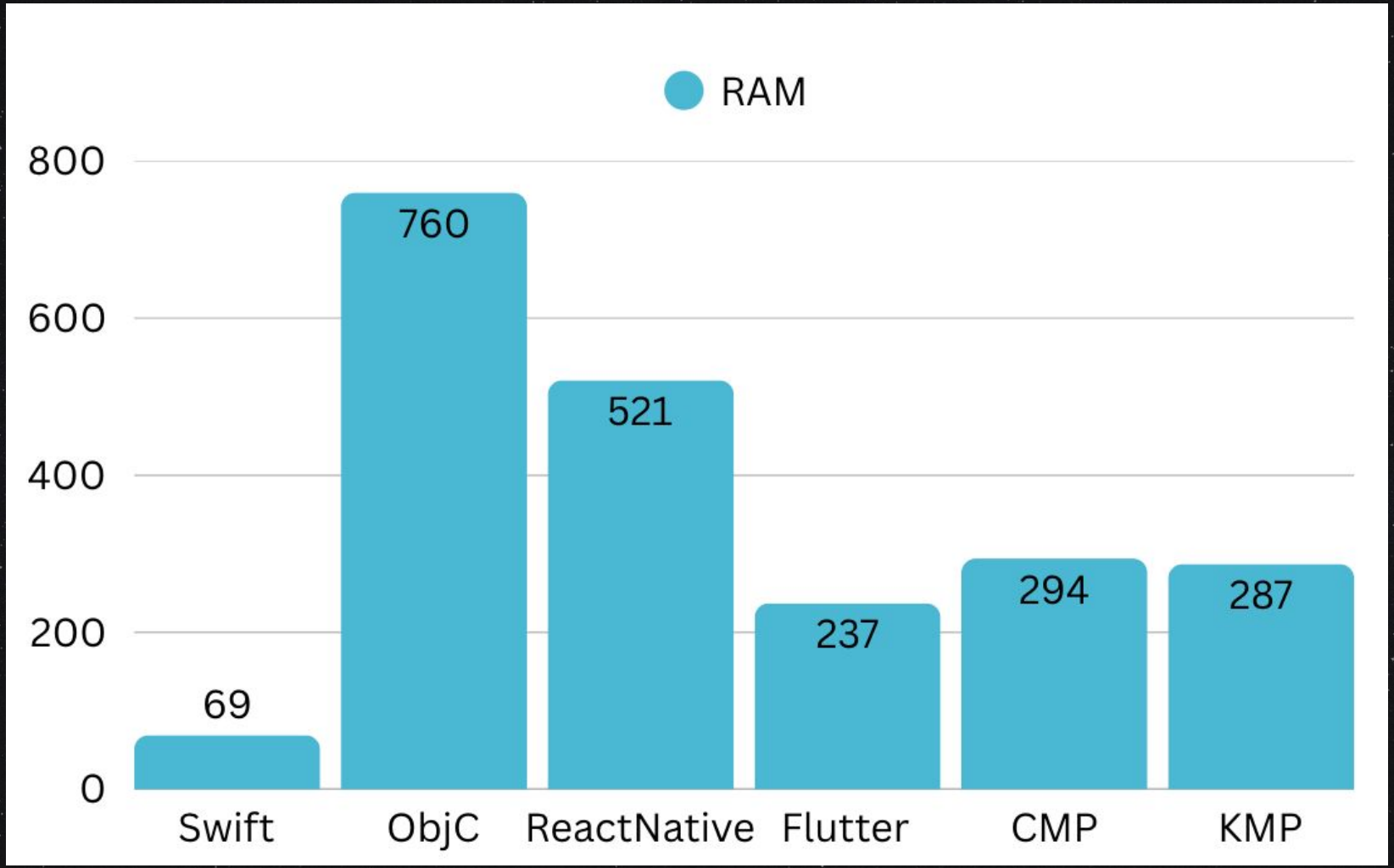

CPU
80
60
40
20
0
50
39
76
50
67
54
Swift
ObjC
ReactNative
Flutter
CMP
KMP

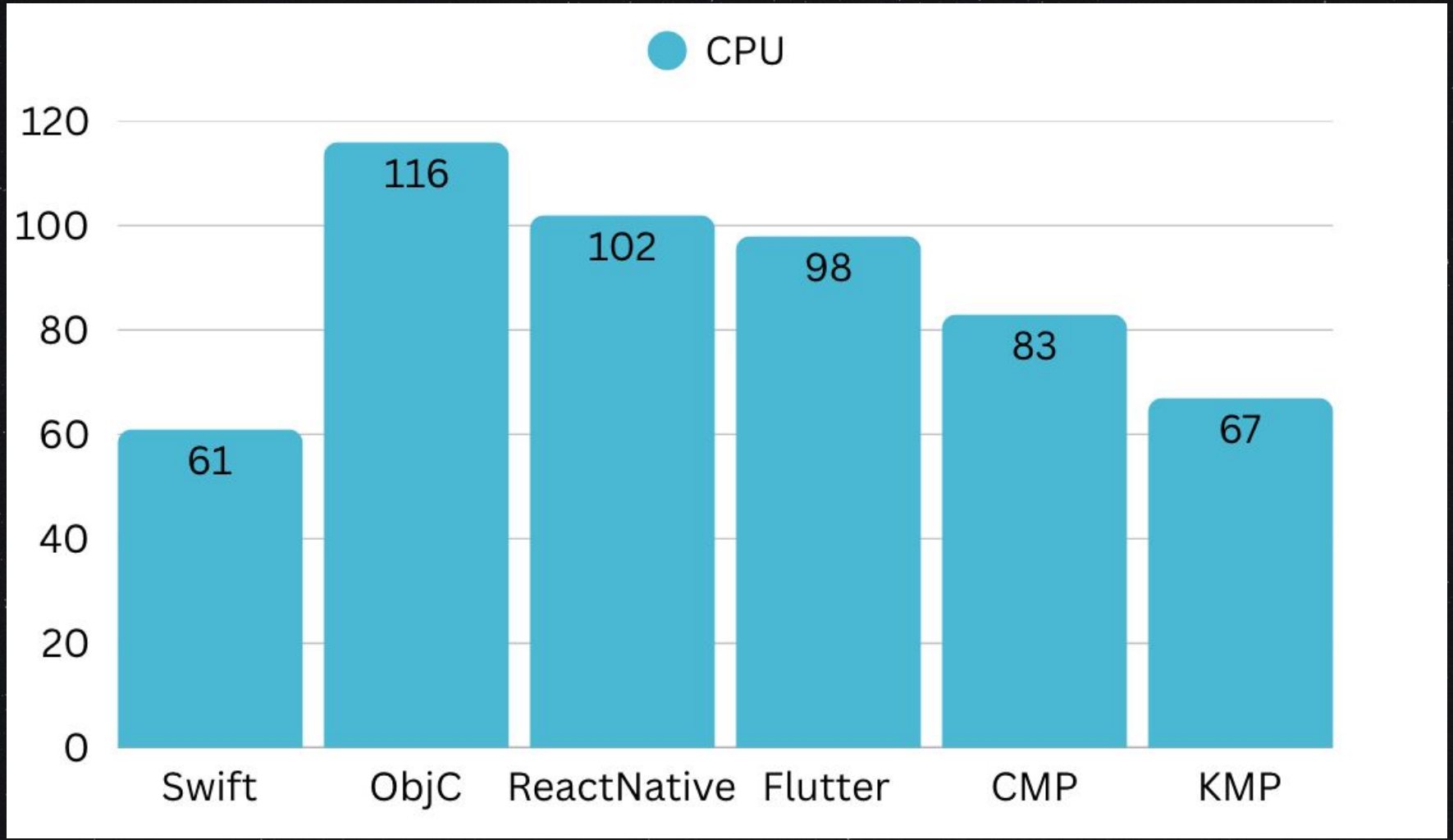

FPS
60
50
40
30
20
10
0
59
60
58
60
47
60
Swift
ObjC
ReactNative
Flutter
CMP
KMP

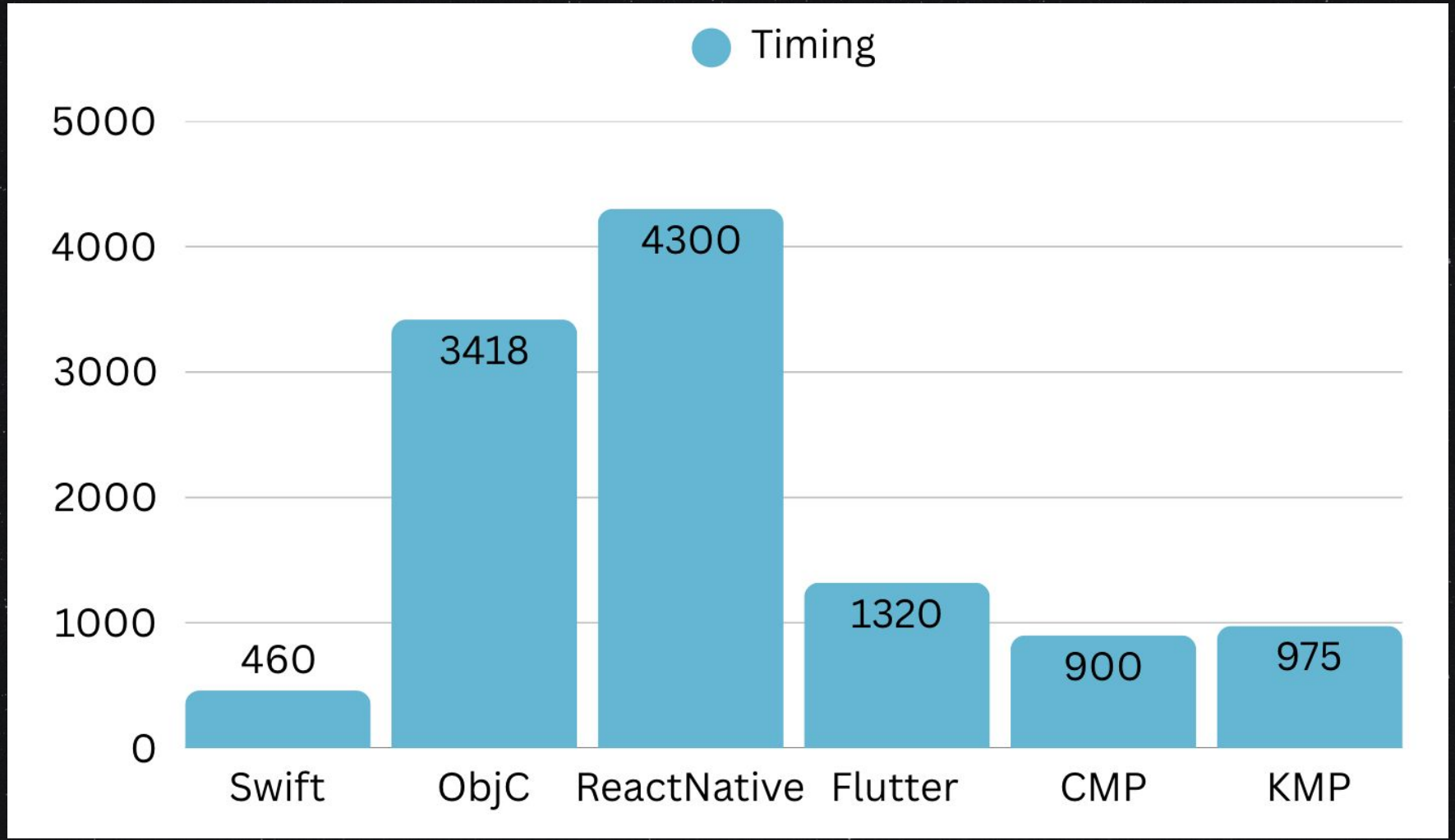

Networking
5000
4000
3000
2000
1000
0
460
3418
4300
1320
900
975
Swift
ObjC
ReactNative
Flutter
CMP
KMP

Files

- GET request to a 26 MB JSON
- Write it 20 times on the file system
- Delete all files


RAM
800
600
400
200
0
69
760
521
237
294
287
Swift
ObjC
ReactNative
Flutter
CMP
KMP

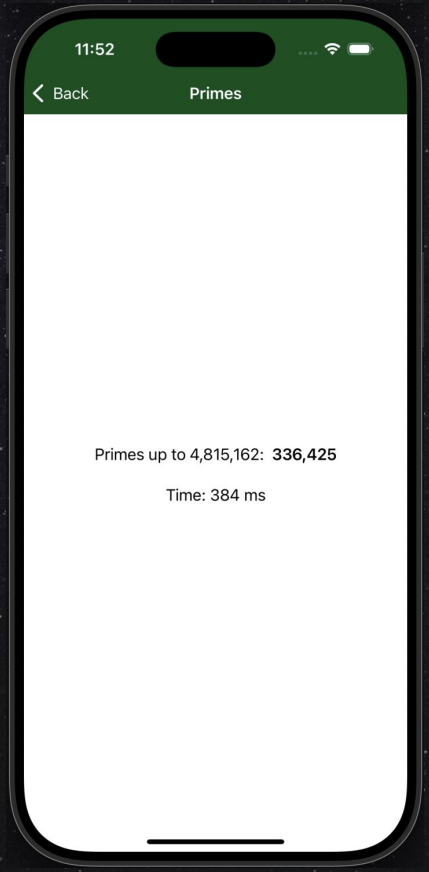

CPU
120
100
80
60
40
20
0
61
116
102
98
83
67
Swift
ObjC
ReactNative
Flutter
CMP
KMP

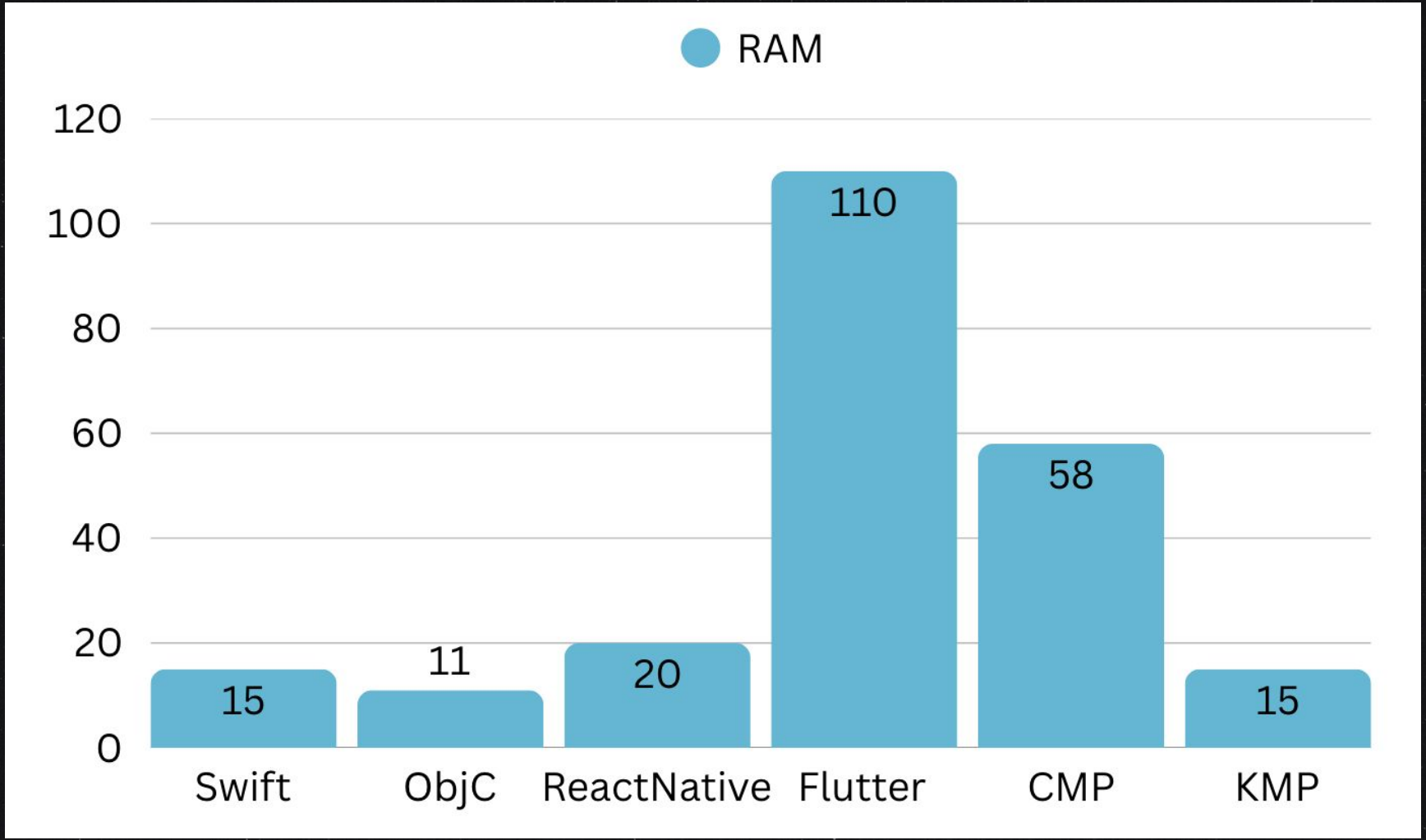

Timing
5000
4000
3000
2000
1000
0
460
3418
4300
1320
900
975
Swift
ObjC
ReactNative
Flutter
CMP
KMP

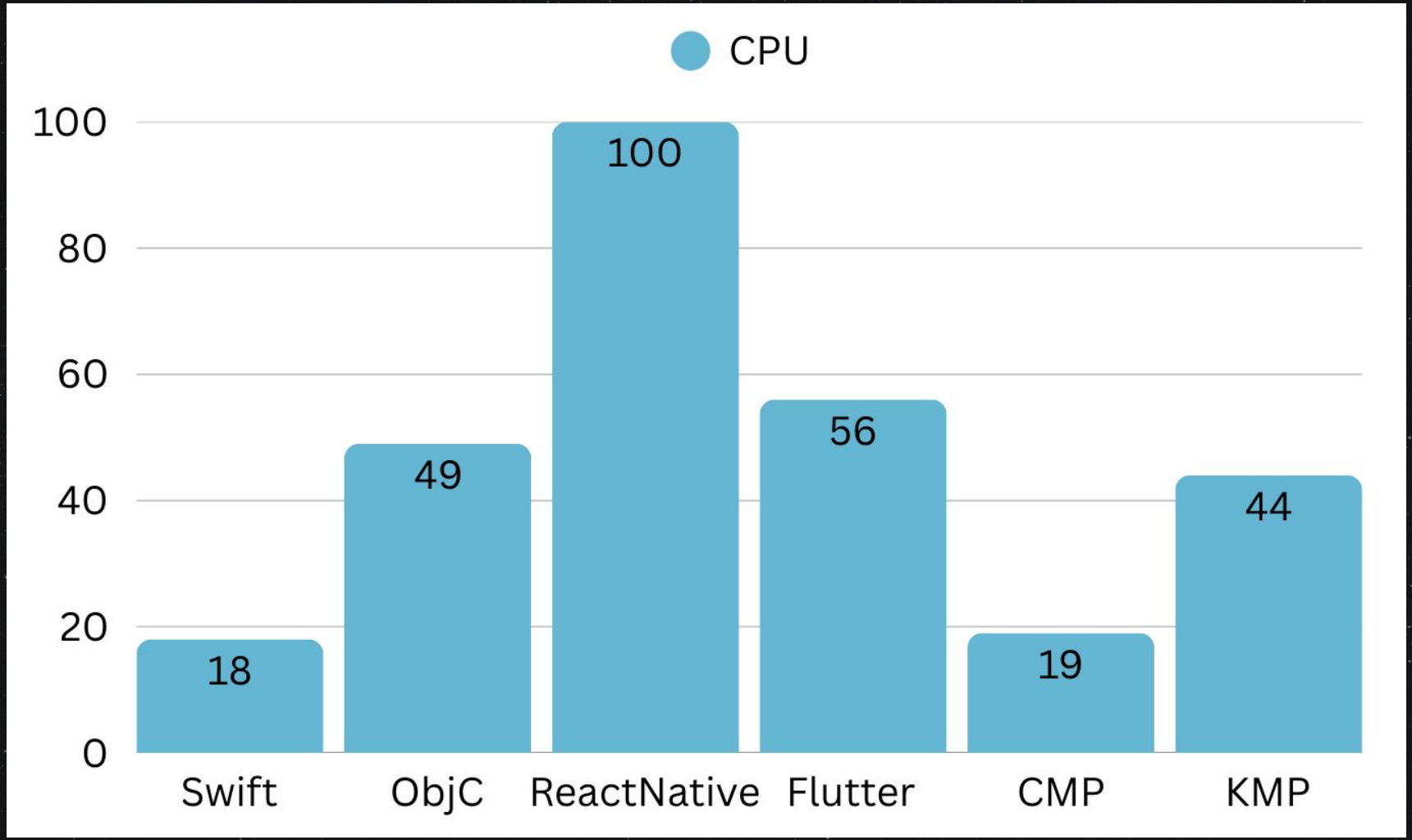
Primes

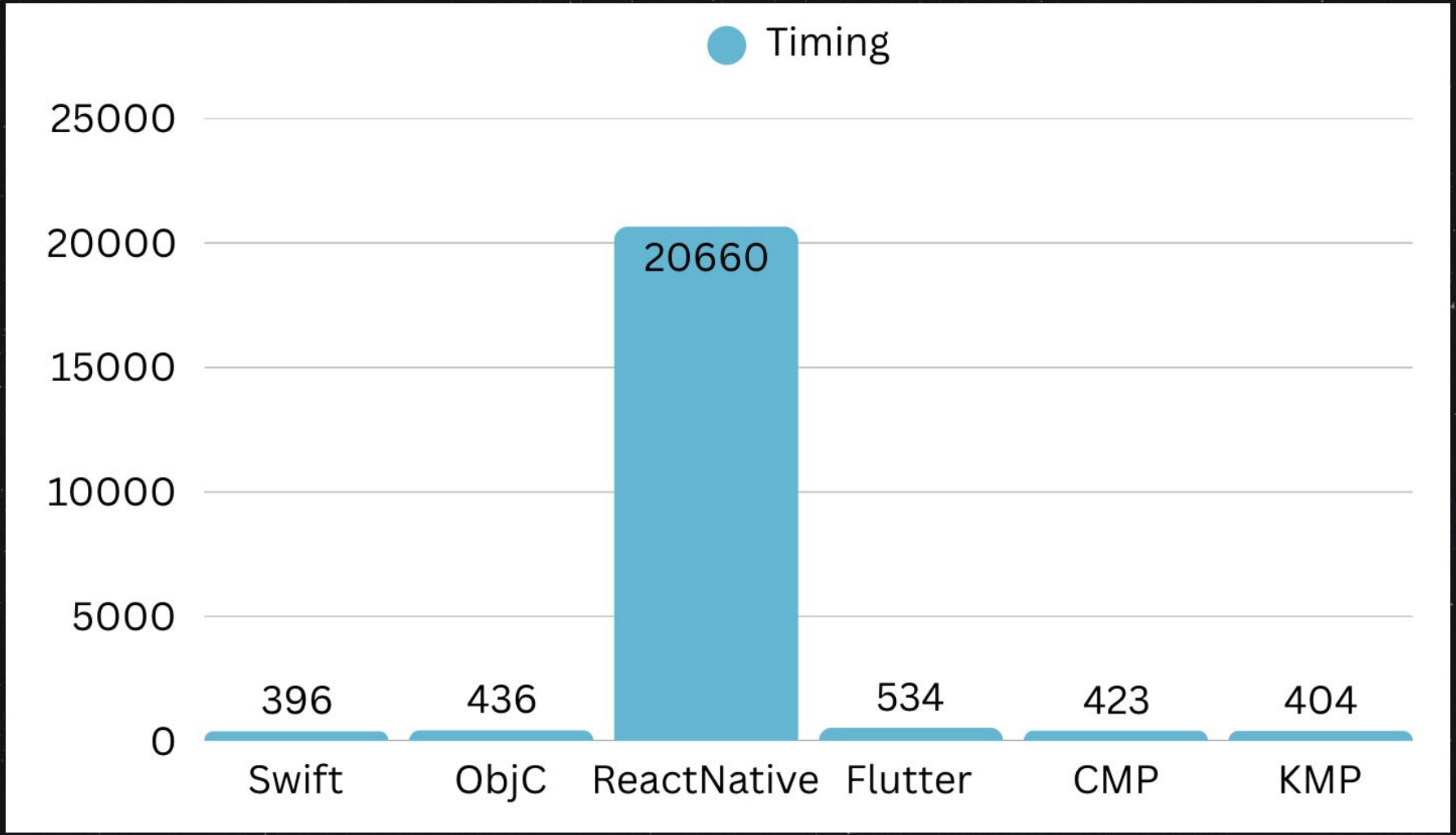
- Simple but heavy algorithm
- Find first n prime number lower that 4 815 162

```swift
static func countPrimes(upTo n: Int) async -> Int {
    if n <= 1 { return 0 }
    var count = 0

    for number in 2...n {
        var isPrime = true
        let maxDivisor = Int(Double(number).squareRoot())
        if maxDivisor >= 2 {
            for divisor in 2...maxDivisor {
                if number % divisor == 0 {
                    isPrime = false
                    break
                }
            }
        }
        if isPrime {
            count += 1
        }
    }
    return count
}
```




RAM
120
100
80
60
40
20
0
15
11
20
110
58
15
Swift
ObjC
ReactNative
Flutter
CMP
KMP


CPU
100
80
60
40
20
0
18
49
100
56
19
44
Swift
ObjC
ReactNative
Flutter
CMP
KMP


Timing
25000
20000
15000
10000
5000
0
396
436
20660
534
423
404
Swift
ObjC
ReactNative
Flutter
CMP
KMP

Best Build Time	 Swift
Smallest ipa size	 ObjC
Best cold launch time	 ObjC
Best git codebase size	 Swift
Best local codebase size	 Swift
Fastest tests	

Worst Build Time

Worst ipa size

Worst cold launch time

Worst git codebase size

Worst local codebase size

Slowest tests

Swift

Best RAM

ObjC

Best CPU

ObjC/Swift

Best FPS

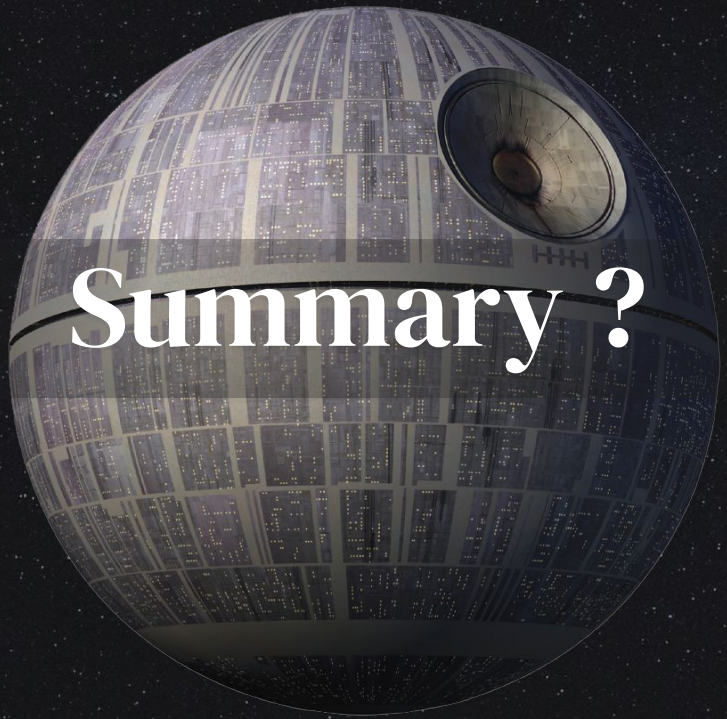
ObjC

Worst RAM

Worst CPU

Worst FPS

- LLMs / Community
- Laptop Battery
- My 50 cents



Summary ?


Thank you !