World class experience

— with React Native —

{callstack}

— Michał Chudziak —

michal_chudziak

mike866

michalchudziak

The challenge

Mobile app

written in React Native

- performant two-way infinite lists
- components shared between web and mobile
- native video players
- native driven animations
- phone & tablet support
- Chromecast integration
- many more…

{callstack}

How to make this kind of app with React Native?

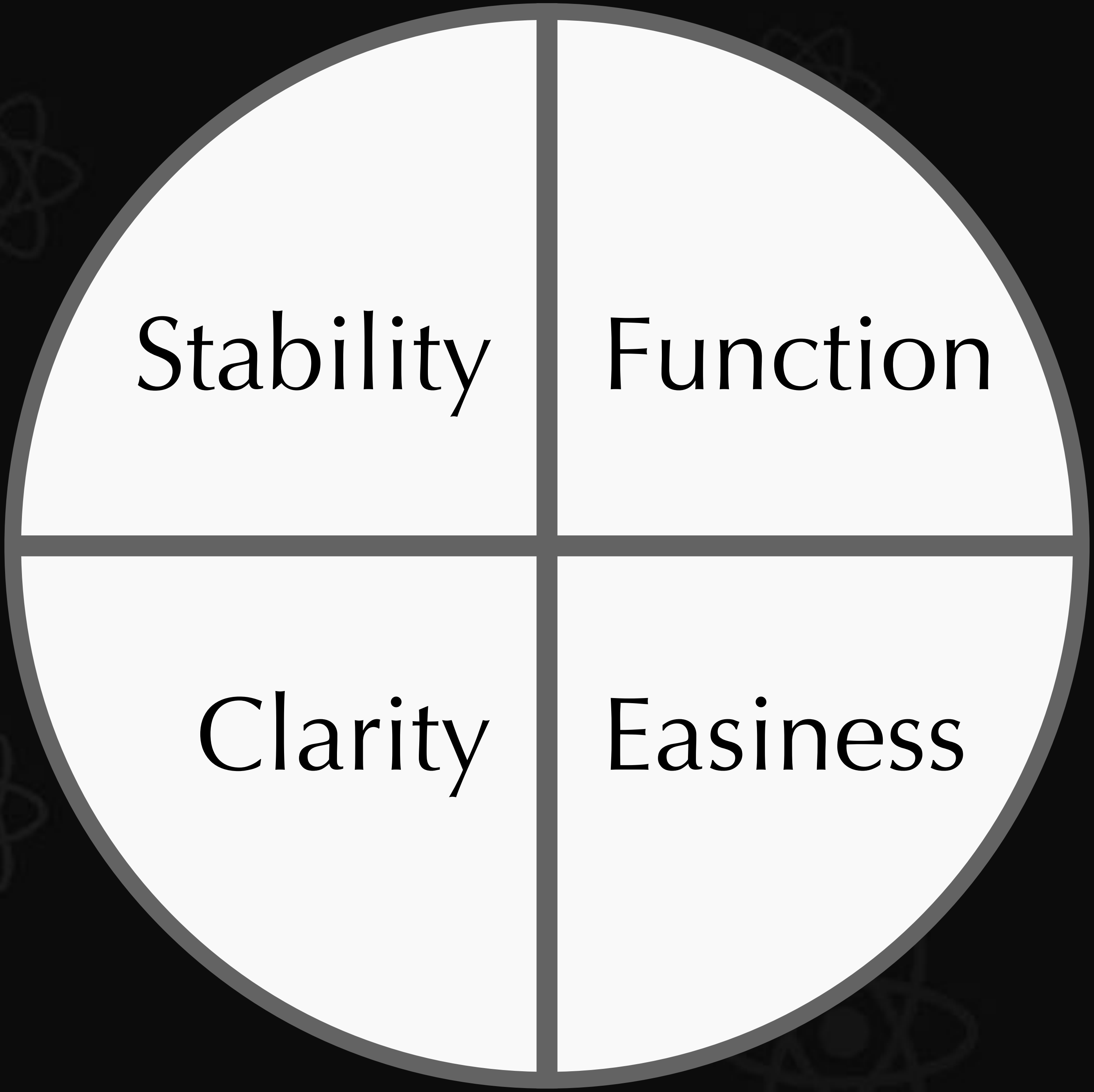
Work close to your team

"It is literally true that you can succeed best and quickest by helping others to succeed."

~ *Napoleon Hill*

Try to keep everyone satisfied

What makes a good DX?



It’s much more than this

Cooperate with QA team,

review each other’s code…

…and use nice tools!

GraphQL

— holy grail of frontend development —

Queries and mutations

```
query FavoriteClub {
  favorite_club {
    id
    background_color
    color_opacity
    crest {
      token
    }
    name {
      short
    }
  }
}
```

```
mutation SetFavClub($club: Club!) {
  set_fav_club(club: $club) {
    id
    background_color
    color_opacity
    crest {
      token
    }
    name {
      short
    }
  }
}
```

How it plays with React Native?

Perfect

thanks to *react-apollo*

react-apollo

```
const withGraphql = graphql(QUERY, {
  skip: ({ localData }) => localData.loading || localData.error,
  props: ({ data }) => ({
    error: data.error,
    loading: data.loading,
    userConfig: data.userConfig,
  }),
  options: ({ shouldPoll, userId }) => ({
    pollInterval: shouldPoll ? 60000 : 0,
    variables: {
      userId,
    },
  }),
});

export default compose(withLocalData, withGraphql)(FollowStar);
```

Why react-apollo?

- simplicity
- composability
- universality
- understandability
- extensibility
- amazing plugin system

```
"apollo-cache-inmemory": “1.1.7",
"apollo-cache-persist": "0.1.1",
"apollo-client": "2.2.3",
"apollo-link": "1.2.1",
"apollo-link-batch-http": "1.0.5",
"apollo-link-dedup": "1.0.6",
"apollo-link-error": "1.0.5",
"apollo-link-http": "1.5.3",
"apollo-link-persisted-queries": "0.2.0",
"apollo-link-state": "0.3.1",
```

Speaking of plugins

You can use them to control things like network requests, caching and…

… the state management!

apollo-link-state

```
import { withClientState } from 'apollo-link-state';

ApolloLink.from([
  withClientState({
    // Same cache as ApolloClient
    cache,
    resolvers: {
      Mutation: {
        update_local_state: async (_: any, args: Object) => {
          await setLocalData(args);

          return null;
        },
      },
      Query: {
        local_state: async () => await getLocalData(),
      },
    },
  }),
]);
```

Now you can use this simple mutation and query in your components

```
{
  update_local_state(userId: 1) @client {
    local_state
  }
}
```

```
{
  local_state @client
}
```

Turns out you don’t need Redux or MobX at all

Haul

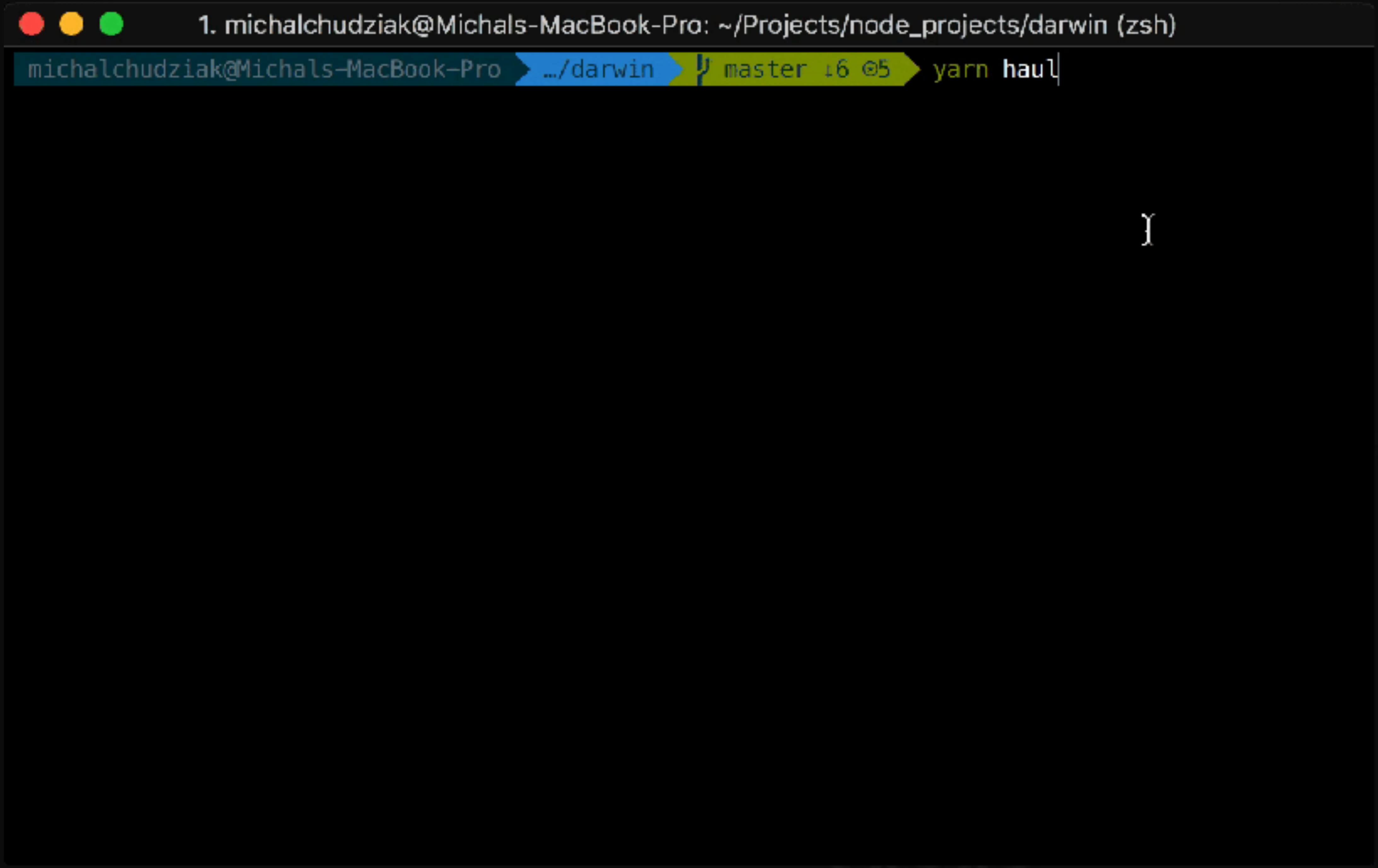
— feel the power of a webpack —

What’s haul?

Haul is a drop-in replacement for react-native CLI built on open tools like Webpack. It can act as a development server or bundle your React Native app for production.

- metro bundler replacement
- easy access to webpack ecosystem
- no watchman needed
- HMR

Haul



```
1. michalchudziak@Michals-MacBook-Pro: ~/Projects/node_projects/darwin (zsh)
michalchudziak@Michals-MacBook-Pro  …/darwin  master ↓6 ⊙5  yarn haul
```

Picking a good tools can save you a lot of time

It’s also worth to automate some processes

Continuous Integration & Continuous Deployment

— from GitHub to your testers with one click —

CircleCI 2.0

How to set it up for react-native?

Create different environments

JavaScript

```
js_env: &js_env
  working_directory: ~/darwin
  docker:
    - image: circleci/node:latest
```

Create different environments

iOS

```
ios_env: &ios_env
  working_directory: ~/darwin/ios
  macos:
    -  xcode: 9.3.0
```

Create different environments

Android

```yaml
android_env: &android_env
  working_directory: ~/darwin/android
  docker:
    - image: farwayer/react-native
```

You can use CircleCI
(or any other CI system) to perform
some code quality checks

What to check?

ESLint

You can run the linter to make sure a code in the PR matches your styleguide

```
lint:
  <<: *js_env
  steps:
    - attach_workspace:
       at: ~/darwin
    - run: yarn eslint
```

Flow

Static type checking helps you fetch a lot of bugs in early stage

```
flow:
  <<: *js_env
  steps:
    - attach_workspace:
        at: ~/darwin
    - run: yarn flow
```

Jest

Unit test are very important,
especially while working in a bigger team

```
unit-tests:
  <<: *js_env
  steps:
    - attach_workspace:
        at: ~/darwin
    - run: yarn test
```

Danger

Danger helps you codify your teams norms, leaving humans to think only about harder problems

```
danger:
  <<: *js_env
  steps:
    - attach_workspace:
        at: ~/darwin
    - run: yarn danger
```

Detox

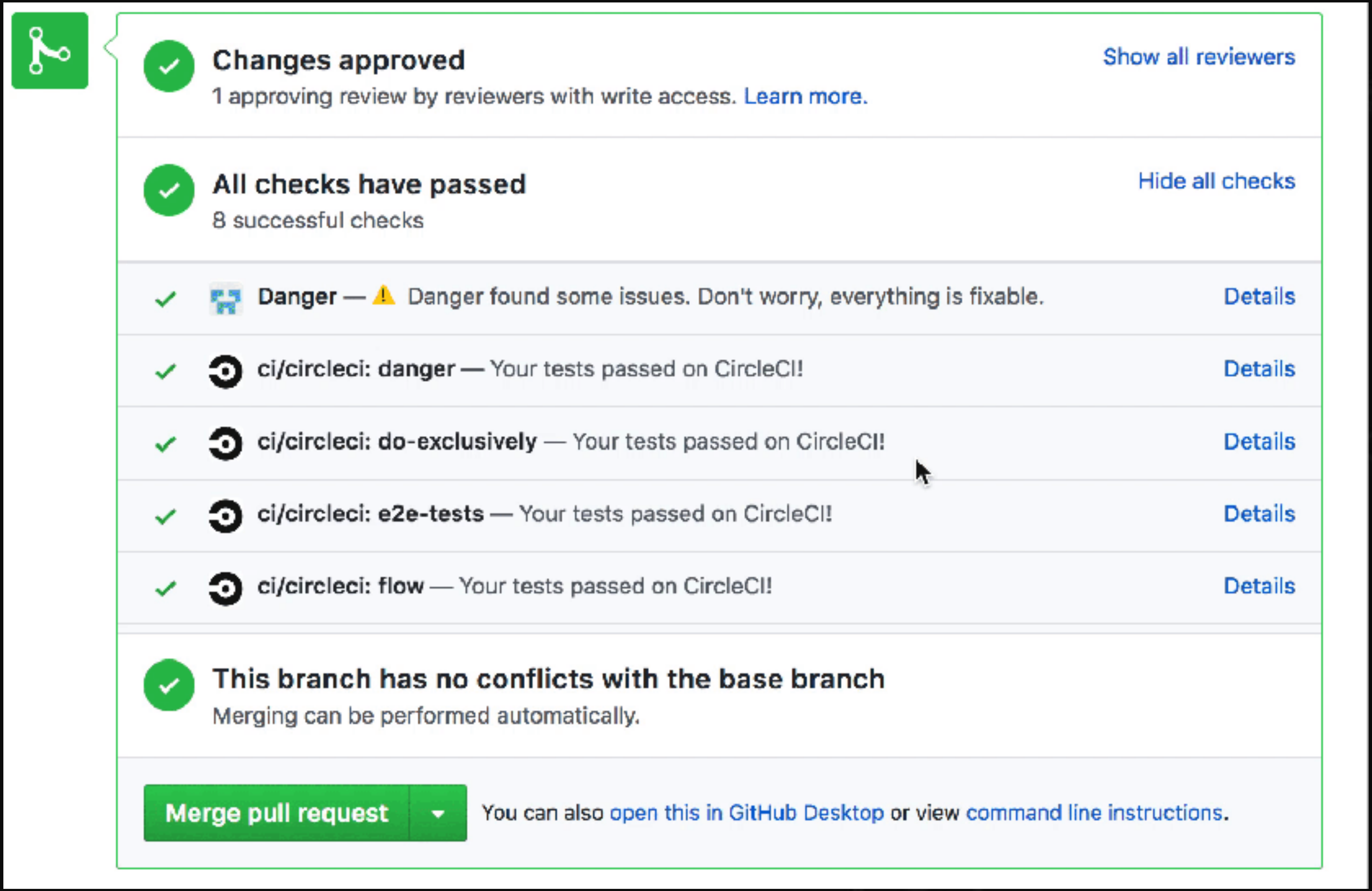
Gray box end-to-end testing and automation library

```
e2e-ios:
  <<: *ios_env
  steps:
    - attach_workspace:
        at: ~/darwin
    - run:
        name: Load Detox dependencies
        command: |
          brew update
          brew tap wix/brew
          brew install --HEAD applesimutils
          yarn global add detox-cli
    - run: yarn e2e:ios:release
```

Check blog.callstack.com for more information about setting up these tools in your project

If every test passes and PR gets accepted

We can merge it to the master



Changes approved
1 approving review by reviewers with write access. Learn more.

Show all reviewers

All checks have passed
8 successful checks

Hide all checks

✓ **Danger** — ⚠ Danger found some issues. Don't worry, everything is fixable. Details

✓ **ci/circleci: danger** — Your tests passed on CircleCI! Details

✓ **ci/circleci: do-exclusively** — Your tests passed on CircleCI! Details

✓ **ci/circleci: e2e-tests** — Your tests passed on CircleCI! Details

✓ **ci/circleci: flow** — Your tests passed on CircleCI! Details

This branch has no conflicts with the base branch
Merging can be performed automatically.

Merge pull request

You can also open this in GitHub Desktop or view command line instructions.

This is where we run beta deployment tasks

Fastlane

Fastlane is OSS platform which lets you automate every aspect of deployment flow

Code signing

```
match(
 type: "adhoc",
 force_for_new_devices: true,
 app_identifier: [
  "com.yourorg.app.NotificationService",
  "com.yourorg.app"
 ]
)
```

Building the app

iOS

```
gym(
  scheme: "darwin",
  silent: true,
  clean: true,
  sdk: “iphoneos11.3",
  export_method: "ad-hoc",
)
```

Android

```
gradle(
  task: "assembleRelease",
  properties: {
    'versionName' =>
       package["version"]
    'versionCode' =>
       ENV["BUILD_NUM"]
  }
)
```

Uploading to Appcenter

```
appcenter_upload(
  api_token: "xxxxxxxxxxxxxxxxxxxxxxxxxxx",
  owner_name: "mike866",
  group: "Alpha Testers",
  app_name: "Darwin",
  // IOS SPECIFIC FIELDS
  ipa: lane_context[SharedValues::IPA_OUTPUT_PATH],
  dsym: lane_context[SharedValues::DSYM_OUTPUT_PATH],
  // ANDROID SPECIFIC FIELDS
  apk: lane_context[SharedValues::GRADLE_APK_OUTPUT_PATH],
)
```

Sending a build to the slack

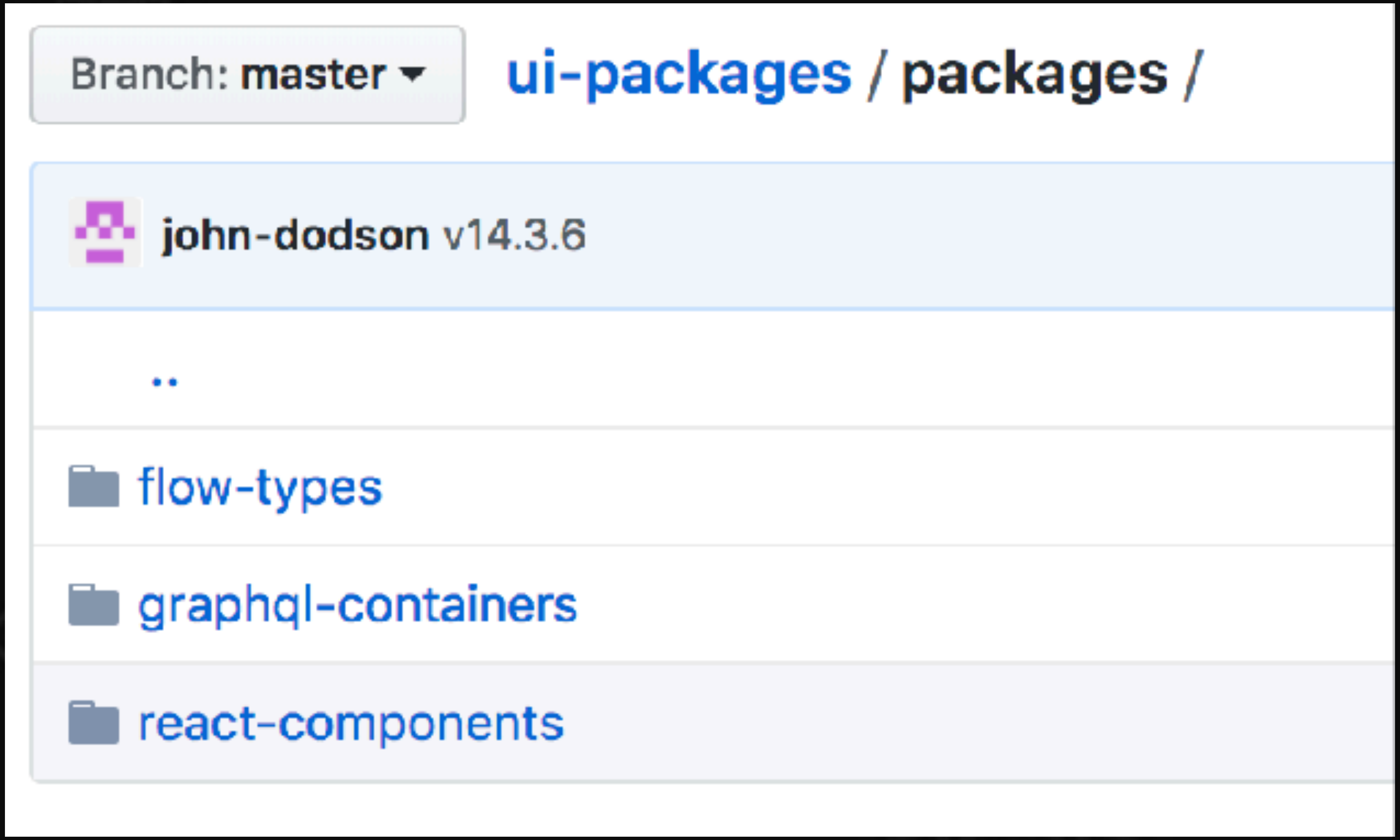
```
slack(
  channel: "#releases",
  username: "Release Bot",
  icon_url: “https://avatars.github.com/u/1111?s=48&v=4",
  message: "<https://install.appcenter.ms/users/mike866/apps/
             Darwin/releases/#{id}/|Android Releases>
             \n🚀 Release number: #{id}",
)
```

The power of React Native

— take advantage of mixed JS and Native worlds —

Share things between the platforms

We use npm packages to share between web and mobile


Branch: master

ui-packages / packages /

john-dodson v14.3.6

..

flow-types

graphql-containers

react-components

You can reuse a lot of stuff between the platforms

- UI components
- Flow typings
- Graphql queries and mutations
- Unit tests
- Utilities
- Helpers

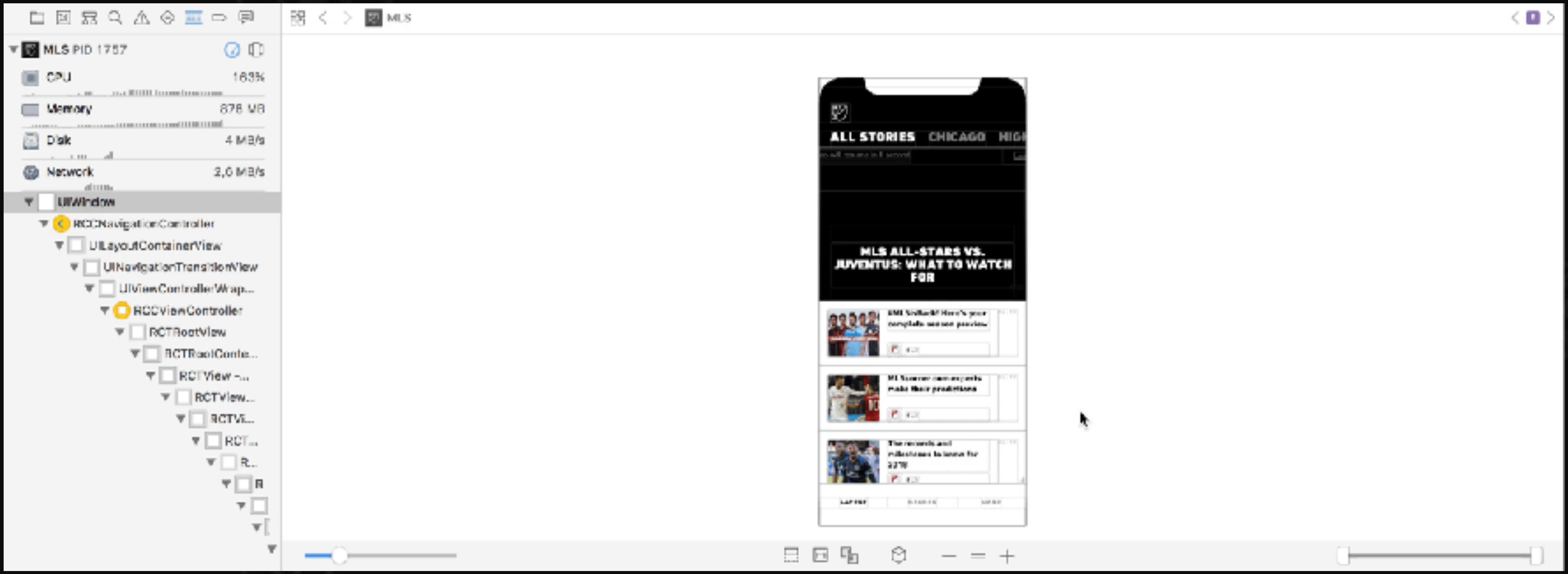
Do it wisely, don’t force universality

In the end we are working with native apps

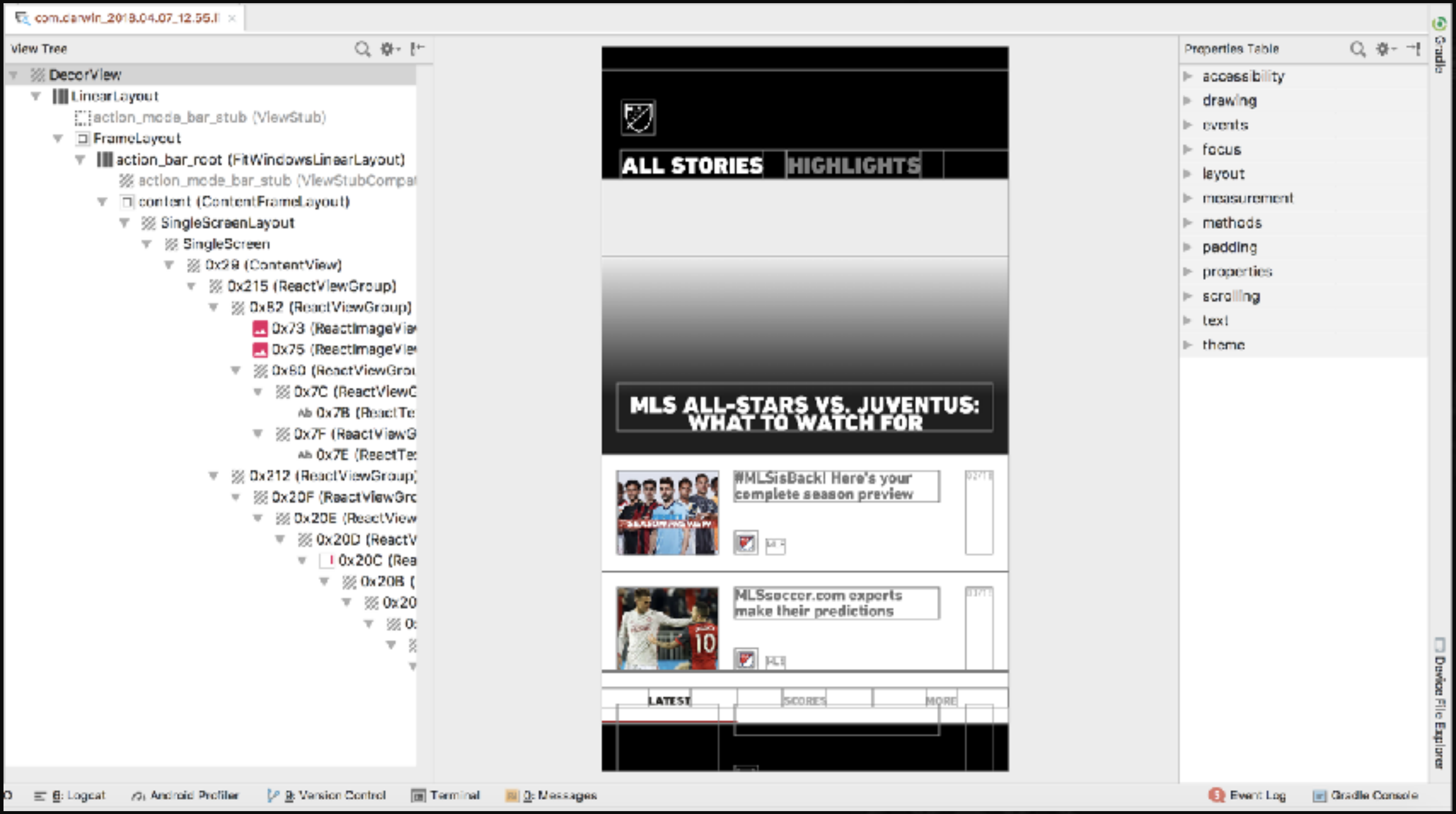
Use the native IDEs

They provide a lot of nice tools

You can debug your UI



Even in Android 😉



com.carwin_2018.04.07_12.55.li

View Tree

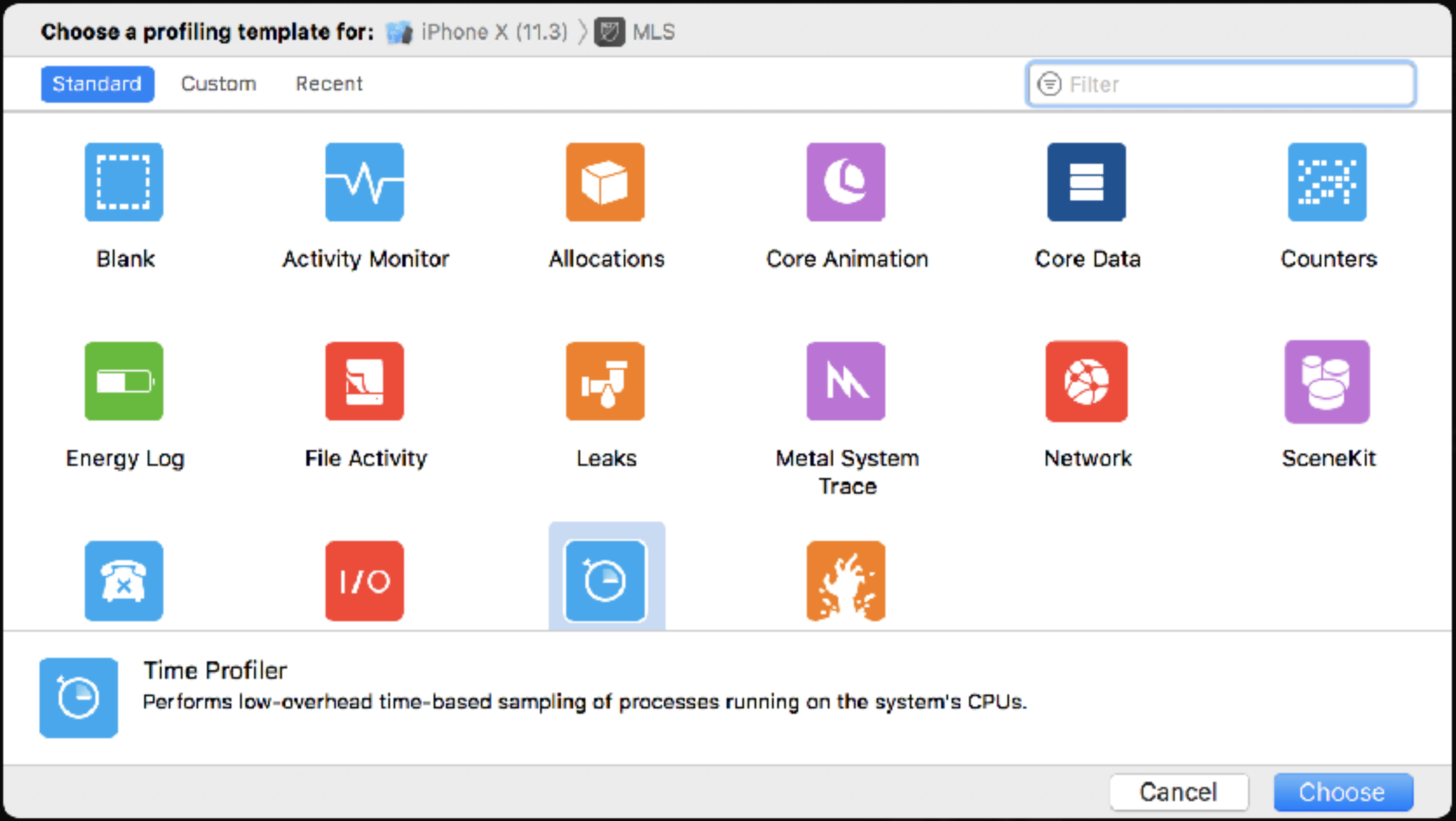
```
DecorView
  LinearLayout
    action_mode_bar_stub (ViewStub)
    FrameLayout
      action_bar_root (FitWindowsLinearLayout)
        action_mode_bar_stub (ViewStubCompat
        content (ContentFrameLayout)
          SingleScreenLayout
            SingleScreen
              0x28 (ContentView)
                0x215 (ReactViewGroup)
                  0x82 (ReactViewGroup)
                    0x73 (ReactImageVie
                    0x75 (ReactImageVie
                    0x80 (ReactViewGrou
                      0x7C (ReactViewC
                        0x7B (ReactTe
                      0x7F (ReactViewG
                        0x7E (ReactTe
                  0x212 (ReactViewGroup)
                    0x20F (ReactViewGro
                      0x20E (ReactView
                        0x20D (ReactV
                          0x20C (Rea
                            0x20B (
                              0x20
                                0:
```

Properties Table

- accessibility
- drawing
- events
- focus
- layout
- measurement
- methods
- padding
- properties
- scrolling
- text
- theme

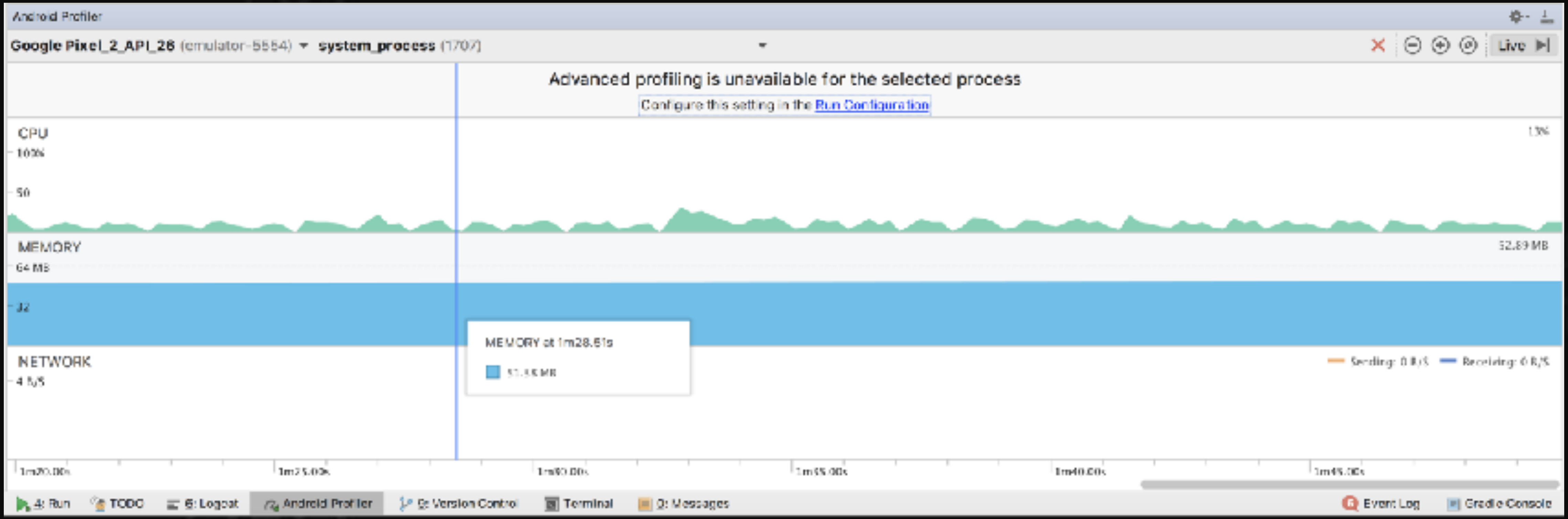
Also, there are some useful performance measuring tools such as

Instruments in Xcode



Time Profiler
Performs low-overhead time-based sampling of processes running on the system's CPUs.

and the Android Profiler



Advanced profiling is unavailable for the selected process

Configure this setting in the Run Configuration

Combining all of these can help you achieve the success

Thanks for listening!

https://blog.callstack.com/develop-react-native-apps-like-a-pro-aff7833879f0

https://blog.callstack.com/continuous-delivery-integration-for-everyone-82d596cec680

{callstack}