

React Native ことはじめ

2022-04-08 社内勉強会 in airCloset

Satoshi Nitawaki

これはなに？

React Native に詳しくない人をターゲットに、React Native のノウハウを話してみようの会！

少しでも学ぶことがあったら幸いだよ！

- 対象読者
 - 1、2回触ったことがあるひと
- 情報粒度：
 - 各項目は知らない事があるかも、各単位は概要レベル。
 - 詳しく知りたいがあればQ&Aで補いたいよ！

話すこと

1. React Native に関するスキルインプット
2. React Native の基礎知識
3. ライブドアの話
4. かゆいところに手が届く知識
5. よもやま話

👉 「React Native 触ったことアル」から「React Native チョットデキル」に

話すこと

- airCloset、airCloset Fitting のドメイン知識
- リポジトリのドメイン知識

これらは別途、「業務上でキャッチアップしてもらう」「また別人から教わる事ができる」ので話さないよ！

タイムライン

約45分予定

15 分

React Native の基礎知識

5 分

Q&A タイム

10 分

ライブラリの話

5 分

Q&A タイム / トイレ休憩

5 分

かゆいところに手が届く知識

5 分

Q&A タイム / トイレ休憩

終了

TL;DR

今の React Native の公式ドキュメントは豊富な内容で、これを読んでおくだけでOK！

まだ読んだことのない方は、読んでみることをオススメします！

React Native · Learn once, write anywhere

はじめに

React Native を使うには、JavaScript (JS) の知識が必要になります！

- JS のチュートリアルサイトはいくつもありますが、ここでは「現代の JavaScript チュートリアル」を紹介します！
- 現代の JavaScript チュートリアル
- JavaScript の基礎 までは抑えると、基本的に困ることはないはず！
- あと、estaが上げてた jsprimer もおすすめ（これが見つからなくて👉を挙げた感じある）

好きな構文

NULL 合体演算子(Nullish coalescing operator) `??`

- `a ?? b` の結果は:
 - `a` が `null` あるいは `undefined` でなければ `a` ,
 - それ以外の場合は `b` .

```
let height = 0;

alert(height || 100); // 100
alert(height ?? 100); // 0
```

React Native とは

「Learn Once, Write Anywhere」

これは「ワンソースでクロスプラットフォーム対応ができるもの」ではないので注意してください。

作り込んでいくと必ずワンソースでは済まなくなります。体感 8割～9割程度。

残りの 1～2 割は、OS 専用のコードや考慮が発生します。これは OS によって設計思想や挙動が異なることが原因で発生することが多いです。（いくつか後述します）

腐らず丁寧に対応してあげてください。 😎

React Native の Component

React Native が提供するコンポーネント

- iOS (Swift / Objective-C)
- Android (Kotlin / Java) のコンポーネント

に接続されています。

これにより、React Native を 1 度構築するだけで、iOS アプリと Android アプリをビルドすることができます。

React Native が提供するコアコンポーネントについてはこちらをみるとよいでしょう 

[Core Components and Native Components · React Native](#)

React Native の Component

よもやま

react-native-web では、RN をビルドすることで dom を生成することができます。つまり web も開発できちゃうのです。ただ、WEB サイト全体を RN で構築することはおすすめしません。なぜならナビゲーションの問題や逆に考慮が増えることでコスト増につながることがあります。

stand.fm や twitter lite 、旧 wantedy（現在は RN をやめている）のパーツとして react-native-web を使用するケースが多いようです。

- https://note.com/moriyuu_/n/n1c5ac722dc26

React Native の Component

よもやま

air-closet では Production コードでは使用していませんが、 ac-native-components という atomic design の共通パーツライブラリでの storybook を簡単に確認する方法として react-native-web を使用しています。

- ac-native-components

emulator 上で storybook を確認したい場合は、 expo を利用することで簡単に確認できるようにしています。

- ac-native-components-expo

Expo と 非 Expo

- Expo
 - React Native の Native 層を隠蔽し、XCode・Android Studio のビルドをせずに、JavaScript だけで動作できる仕組み
 - 公式ドキュメントのサンプルコードは、Expo Snack という playground を提供する仕組みで記載されている

Expo と 非 Expo

Pros / Cons

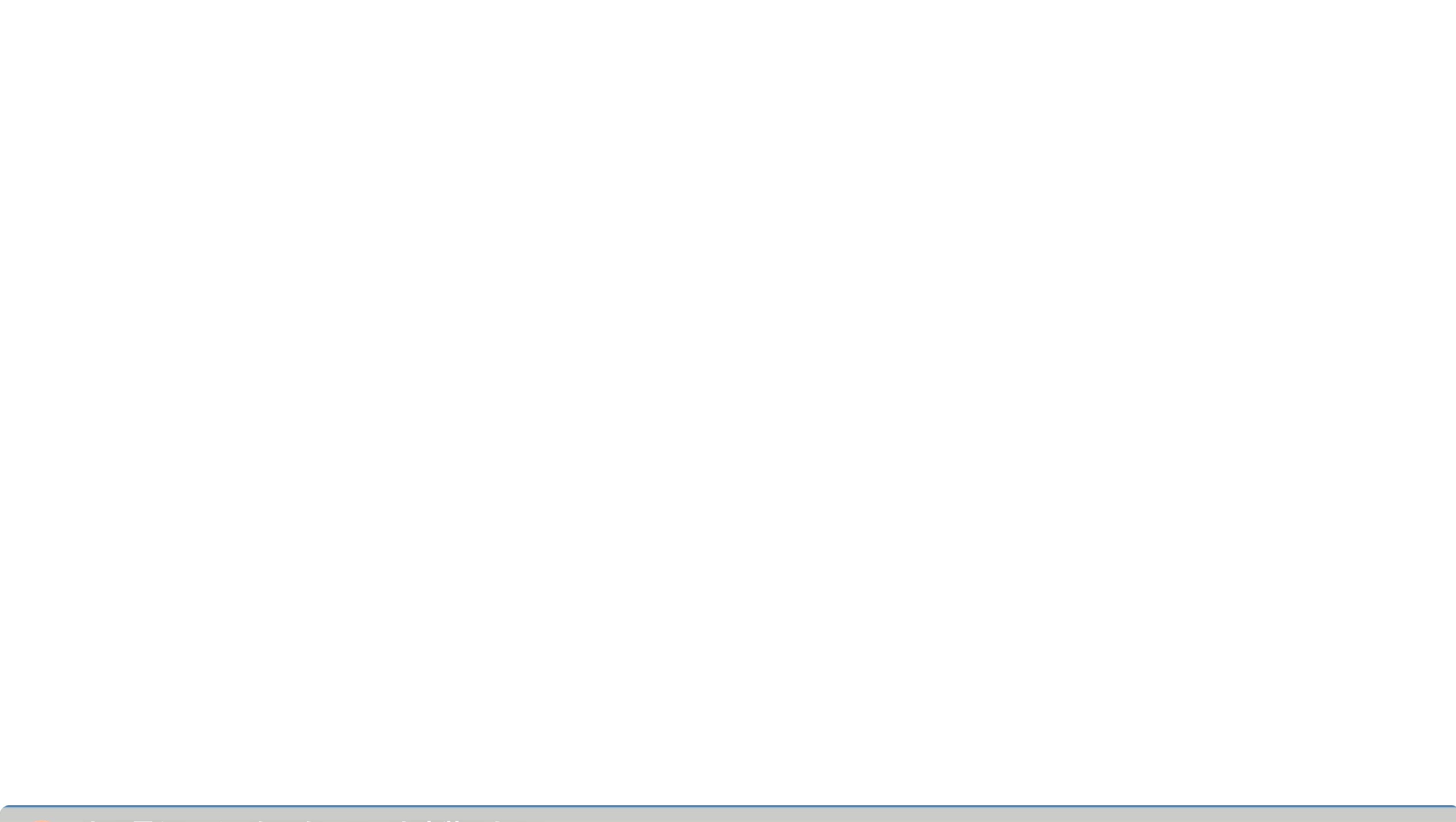
- Pros
 - Native の環境構築にハマらない
 - 既に Expo が整備してくれている way に乗れるので、エラーが発生しない（しにくい）
 - 環境構築コストがゼロ
- Cons
 - Native のコードを触ることができない。
 - Expo が提供していない機能にはアクセスできない
 - Expo の守備範囲を超えるものを自作することができない。
 - 自作するには Expo reject しなければいけない。

- 公式も始めは Expo が簡単だと言ってる

Expo と 非 Expo

Expo API

- expoはめちゃくちゃ豊富にある



ライブラリの違い

<https://reactnative.dev/docs/libraries>

JS ライブラリ

- npm
 - package.json で管理されるもの

ライブラリ

<https://reactnative.dev/docs/libraries>

Native ライブラリ

- ios
 - Cocoapod
 - podfile
- android
 - gradle? `app/build.gradle` で管理されている

よもやま

React Native Directory という便利なサイトができていたので、RN のライブラリはここで探すとよいかもしれない。

React Native Directoryは、ReactNative 専用に構築されたライブラリの検索可能なデータベースです。これは、ReactNative アプリのライブラリを探す最初の場所です。

ディレクトリにあるライブラリの多くは、ReactNativeCommunity または Expo からのものです。

Linking

<https://reactnative.dev/docs/linking>

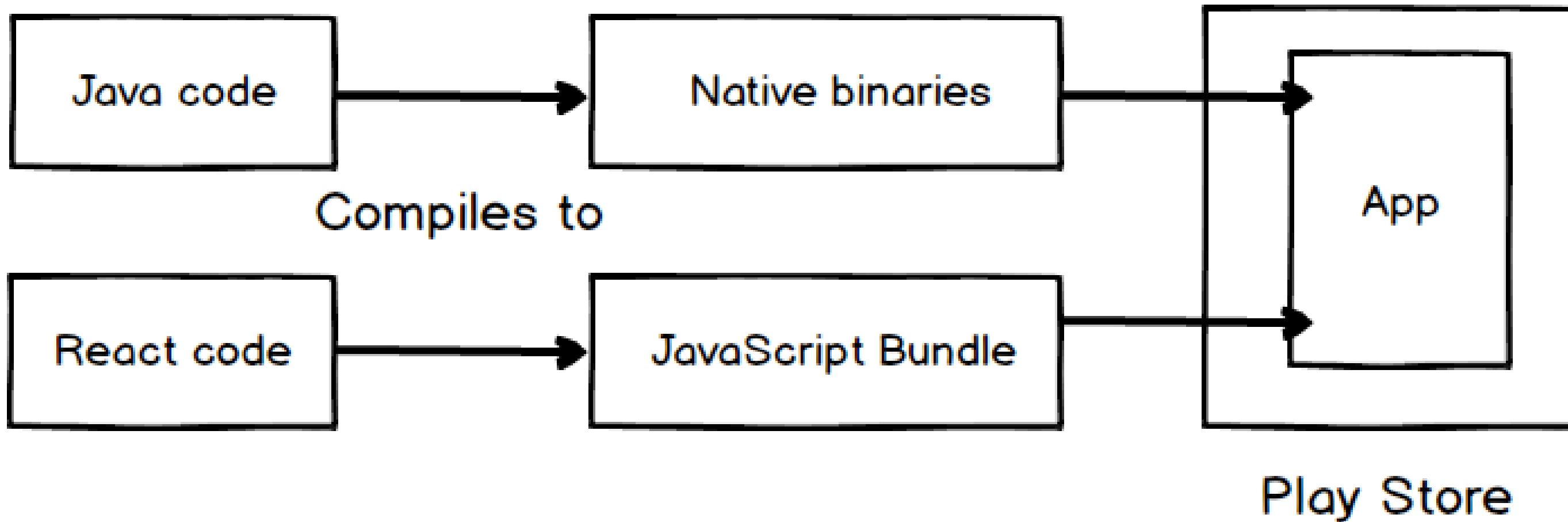
Linkingとは、`node_modules`に存在する native code をRNプロジェクトに接続すること。

- Auto Linking
 - `npm install`するだけで、接続完了（0.6Xで追加された）
- Manual Linking
 - ネイティブコードを直接編集して接続する
 - e.g. KARTE

ライブラリ

code-push

通常のアプリデプロイフロー



出典: <https://vijayt.com/post/integrate-codepush-to-a-react-native-android-app/>

code-push

code-pushデプロイフロー

firebase

react-native-firebaseはドキュメントが💩💩💩

firebase自体のドキュメントは優秀

各ライブラリに触れたかったが、時間がないので割愛。

- firebase/dynamic-link
- firebase/push
- firebase/log-event

react-navigation

最もポピュラーなライブラリから、公式ドキュメントに記載されるライブラリへ (react-navigation, 公式Doc)

dark-modeだったり、safe-areaだったりをこのライブラリ単体でも仕組みとして持っている。ややこしい。

- v1
 - redux前提
- v2
 - 脱reduxの流れを組み、脱redux
- v3
 - v2 の未成熟な部分の強化のイメージ
- v4
 - インタフェースが変わって、オブジェクトの構成から、componentの構成に変わった
- v5
 - (まだキャッチアップできていない)
- v6

react-navigation

よもやま

- pickss
 - v1 -> v3
 - reduxの残骸が残っている
- air-closet
 - v2 -> v4
 - 元々redux管理じゃないので、upgradeが楽
- react native upgradeのタイミングで一緒に引き上げるのがタイミング的に良いと思われる。

react-native-screens

<https://github.com/software-mansion/react-native-screens>

元々JSオンリーだった react-navigation にネイティブライブラリが登場した。

個人的に、ナビゲーションを実装する上で、スクリーン制御をJSのみで完結するのはパフォーマンスの限界があったのだと思う。

patch-package

なんとなく紹介

React Nativeは数々のライブラリに支えられているフレームワーク。

そのため、ライブラリでバグが発生することが多い。

そのバグ修正がリリースされる前にこちらでライブラリに対するパッチを当てることができる

それがこちら。

お役立ち情報

React Native Upgrade

<https://reactnative.dev/docs/upgrading>

- RN フレームワーク部分のupgrade.
- 公式の Upgrade Helper を使うとよい。

先人いわく、

「ゴールの見えない暗闇を2週間走り続けるようなものだ」

React Native Upgrade

難しいポイント

1. ネイティブコードの修正が必須
 2. 旧アプリの場合、auto-linking で install されておらず、manual install で入っていると、既存コードとフレームワークのコードの調整をしなければいけない。
- 2022年4月現在もネイティブコードの修正が必要なライブラリは存在する：e.g. KARTE
KARTEはRNアプリの上にポップアップを被せるといった、Native層に依存度の高い動作があるためと推測する。

React Nativeの短所

長所は一旦割愛。

1. アプリのサイズが大きい
2. Android でのメモリリーク問題（バックグラウンドでの不要プロセス、メモリリークのマニュアル考慮）
3. 依存関係によるアプリ起動遅い件

出典: <https://freexbcodes.com/page-60/react-native7/>

依存関係によるアプリ起動遅い件

👉 アプリサイズをへらす

アプリで使用するライブラリとコンポーネント数を減らす 画像を圧縮してリソースを最適化する

SectionList、FlatList、VirtualList を使用してメモリリークを解消する ListView はガーベジコレクションが働くことがあるとのこと。メモリリークされないので基本的に使わない。公式ドキュメント上からも、0.60 以降のリニューアル後では紹介すらされなくなったことからも非推奨に近いことがわかる。

■ 公式ドキュメント：ListView での FlatList 紹介

```
const FlatListBasics = () => {
  return (
    <View style={styles.container}>
      <FlatList
        data={[
          { key: "Devin" },
          { key: "Dan" },
          { key: "Dominic" },
          { key: "Jackson" },
          { key: "James" },
          { key: "Joel" },
          { key: "John" },
          { key: "Jillian" },
          { key: "Jimmy" },
          { key: "Julie" },
        ]}
        renderItem={({ item }) => <Text style={styles.item}>{item.key}</Text>}
      />
    </View>
  );
};
```

■ 公式ドキュメント：昔の LiveView

```
class MyComponent extends Component {
  constructor() {
    super();
    const ds = new ListView.DataSource({
      rowHasChanged: (r1, r2) => r1 !== r2,
    });
    this.state = {
      dataSource: ds.cloneWithRows(["row 1", "row 2"]),
    };
  }

  render() {
    return (
      <ListView
        dataSource={this.state.dataSource}
        renderRow={(rowData) => <Text>{rowData}</Text>}
      />
    );
  }
}
```

パフォーマンスチューニング

- 第一に、「Android はパフォーマンス・チューニングが必要（必須）」という認識を持つこと。
- iOS が動作していても、Android だと動作しない（遅い）ことがままある。
- 逆に、Android でセーフでも iOS がアウトのときがある。



やっておくこと

1. resizeMethod(Image Component)

- 事前にAndroid側にresizeする情報を与えておく

2. removeClippedSubview(FlatList Component)

- デフォルトは画面外の component のレンダリング結果をメモリに保持しつつける
- 開放するflagがこちら

文字列は必ず`<Text>`を使用する

- web と違って、Text で囲まないとエラーになる
 - このままリリースするとクラッシュするので気をつけて

```
const { label = "" } = props;  
  
return <View>{label}</View>;
```

react-native-tips - Snack



- 【React Native】 Cannot Add a child that doesn't have a YogaNode to a parent with out a measure function
 - [Qiita](#)

YellowBox / RedBox (LogBox)は無視しない

みんな、LogFileちゃんと消してます？（消してませんよね？）

YellowBox / RedBox (LogBox)は無視しない

- LogBox
 - `console.warn`: Yellow
 - `console.error`: Red
- なぜ無視したらダメ?
 - パフォーマンス上のリスクを警告してくれている可能性がある
 - Yellowでも無視してはいけない
 - 過去にwarnを無視して、バグを引き起こした事がある
 - e.g. `<Image>` に border を当ててしまい YellowBox が表示され、リリースビルドでクラッシュ

その他

- 【React Native】 AndroidのTextでlineHeight当てるのにセンターにならない - Qiita
- 【React Native】 画面が真っ白で起動しない - Qiita
- 【react-navigation】 カスタマイズヘッダー適用時にカックカクする問題 - Qiita
- 【React Native】 TextInputでキーボードが押し上げられる【Android】 - Qiita
- 【React Native】 Android実機デバック手順 - Qiita
- 【React Native】 FlatListのデータが変更されてもrenderされない - Qiita
- 【JavaScript】 Self Objectの値を参照してプロパティ値設定したい - Qiita
- 【React Native】 Android emulatorで 'localhost' を参照しない - Qiita
- react-native-modalboxで`<ScrollView>`のスクロールが効かないとき - Qiita
- 【React Native】 borderBottomが当たらない - Qiita