～モバイルアプリを作ろう～
夏のインターンシップ

2018年9月27日, 28日

（株）アドヴァンスト・ソフト・エンジニアリング

目次（前半）

- 講師自己紹介
- スマホアプリ開発は難しい
- クロスプラットフォーム開発
- 休憩（15分程度）
- javascriptプログラミングの基礎
- Reactの基礎

10：15 ～ 12:00を予定

講師自己紹介

講師自己紹介

- 名前：　柳沢　俊彰
- 出身：　北海道札幌市
- 年齢：　26歳（3年目）
- 趣味：　酒・ゲーム・ボルダリング・一人旅など
- 座右の銘：　明日にゃ明日の風が吹く
- 2016年4月にASE入社、防災・医療・警察関係のシステム開発に従事、業務での使用言語は主にC#とJavaScript(React.js)

講師自己紹介

- 名前：　目黒　将志
- 出身：　北海道札幌市
- 年齢：　38歳（17年目）
- 趣味：　サッカー、ゲーム、音楽
- さくせん：　たのしくやろうぜ
- 2002年4月にASE入社、以来15年間C言語でレガシーなサーバアプリの開発・保守を担当していたが、去年Webエンジニアへのクラスチェンジを果たす。今はRuby on RailsとJavaScript(React.js)を主に扱う。

スマホアプリ開発は難しい

スマホアプリ開発は難しい
何故難しいのか

スマホアプリ開発の障壁となる要素たち

- 対応が必要な画面サイズが多い
- タップ、スワイプなど、旧来のシステム開発ではあまり考慮されてこなかった制御が多い
- デバッグが難しい（簡単に試せない）
- リリースにお金がかかる

Apple Developer Program：年間参加費 $99

Googleディベロッパー：登録料 $25

スマホアプリ開発は難しい
何故難しいのか

たぶん一番の障壁はこれ

- 各OSごとの対応
  - iOS
  - Android
  - ~~WindowsPhone~~
  - ~~FireOS~~

iOSとAndroidで開発に必要な環境や言語が違う

異なる二つのシステムを開発すると考えた方が良いくらい違う

→純粋に労力が2倍

スマホアプリ開発は難しい
Androidの開発

開発環境

- 必要環境：
  - PC（OSは問わない）
  - JAVA環境
  - Android Studio（公式IDE）
  - デバッグ用の実機、またはシミュレータ

- 使用言語：Java（Kotlin）

→もはや説明不要な20年選手、オブジェクト指向の考えを取り入れ大流行した。スマホアプリ開発でも言語仕様は大きく変わらない。

近年、KotlinというJavaの後継言語も出てきており、それを利用することもできる。

スマホアプリ開発は難しい
iOSの開発

開発環境

- 必要環境：
  - mac OS系のPC
  - Xcode(公式SDK)
  - デバッグ用の実機、またはシミュレータ

- 使用言語：Swift（Objective-C）

→Appleが開発した比較的新しい（2014～）プログラミング言語

静的言語で「型が強い」「null安全」などの特徴があり、堅牢な仕様。

昔のアプリはObjective-Cという言語で開発されていた（今もできる）

スマホアプリ開発は難しい
難しさは回避可能か？

ここで紹介した障壁には回避できないものもあります。
（Apple Developer Programに支払う1万円とか。）

しかし、「各OSごとの対応」などについては完全に避ける事はできなくても、別の方法を選ぶ事も可能です。

クロスプラットフォーム開発

クロスプラットフォーム開発

クロスプラットフォームとは

異なるOSで同じ仕様のものが実行できるプログラムのこと。
上記のプログラムを開発すること = クロスプラットフォーム開発

- メリット
  - 一つの言語（JavaScript、C#など）でiOS、Android両方の開発が可能
  - 開発期間や学習コスト等を下げる効果が期待できる
  - テストも共通化できる。

- デメリット
  - 動作が重くなる場合が多い
  - 完全には共通化できない場合が多い（各デバイスごとにバグ対応などを行う必要があるケースも）

クロスプラットフォーム開発
代表的な開発手法

① Apache Cordova

- 特徴
  - WebViewエンジンを使用することでクロスプラットフォームを実現
  - カメラなどのネイティブ機能がプラグインを追加する事で利用できる
    （標準では組み込まれない）
  - Web標準の技術を使用するため、パフォーマンスは出にくい
  - 利用者の多い手法のため、情報が多い

- 使用言語：HTML, CSS, JavaScript

→Web開発の基本となる技術セット、ネイティブ機能の利用にはAPIを利用する。

クロスプラットフォーム開発
代表的な開発手法

② Xamarin（ザマリン）

- 特徴
  - iOS、Androidの両環境で動作する.NET環境
  - Visual Studio内に開発環境が含まれている
  - ネイティブ機能のAPI移植率100%
  - コンパイルする事でネイティブアプリ化する事が可能

- 使用言語：C#

→Microsoftが2000年に公開したプログラミング言語。現在も積極的に機能追加がなされており、現在でもJava、Python、Javasctriptと並んで世界トップクラスのシェアを持つプログラミング言語となっている。

クロスプラットフォーム開発
代表的な開発手法

③React Native

- 特徴
  - Facebookが開発したReactという言語をモバイル開発に使用する。
  - Reactのデザインパターン等をほぼそのまま用いる事が可能
  - コンパイルする事でネイティブアプリ化する事が可能
  - 周辺環境が整備されており、実機デバッグが容易
  - ユーザー数は多いが新しい技術のため日本語資料が少ない

- 使用言語：React.js(JavaScript)

→Facebookが2013年に公開した新しい言語、JavaScriptのライブラリという位置づけだが、異なる部分が多いため、JavaScriptの経験があっても初めて挑む場合は学習コストがかかる。

クロスプラットフォーム開発
代表的な開発手法

③React Native

- 特徴
  - Facebookが開発したReactという言語をモバイル開発に使用する
  - Reactのデザインパターン等をほぼそのまま用いる事が可能
  - コンパイルする事でネイティブアプリ化する事が可能
  - 周辺環境が整備されており、実機デバッグが容易
  - ユーザー数は多いが新しい技術のため日本語資料が少ない

- 使用言語：React.js(JavaScript)

→Facebookが2013年に公開した新しい言語、JavaScriptのライブラリという位置づけだが、異なる部分が多いため、JavaScriptの経験があっても初めて挑む場合は学習コストがかかる。

クロスプラットフォーム開発
代表的な開発手法

ちなみに、世間的なトレンドは以下のような感じ

検索ワード「Cordova」「 Xamarin」「 React Native」の比較

https://trends.google.co.jp/trends/explore?q=Cordova,%2Fm%2F0gtv959,React%20Native

世界的にはReact NativeとCordovaの2強
日本を見るとXamarinとReact Nativeが強い

これまで述べた3つの手法以外にも、様々な手法がある
（ TitanumMobileやゲーム開発用のUnity、Cocos2d-xなど）

休憩

Javascriptプログラミングの基礎

Javascriptプログラミングの基礎

JavaScriptとは

Wikipediaより引用

https://ja.wikipedia.org/wiki/JavaScript

JavaScript（ジャバスクリプト）とは、プログラミング言語のひとつである。Javaと名前が似ているが、全く異なるプログラミング言語である。

ウェブブラウザ上で動作し動的なウェブサイト構築やリッチインターネットアプリケーションの開発に用いられる。また、2010年以降はnode.jsなどのサーバサイドJavaScript実行環境や各種ライブラリの充実により、MEANに代表されるように、Web開発の全ての領域で活用されるようになってきている。

※MEAN・・・MongoDB,Express,AngularJS,Node.jsによる開発。
DB・サーバからフロントエンドまですべてがJavaScriptのみで開発できるという特徴がある。

Javascriptプログラミングの基礎

ECMAScriptとは

またしてもWikipedia
https://ja.wikipedia.org/wiki/ECMAScript

ECMAScriptとはJavaScriptの標準規格である。
ECMAScriptのEdition5（ES5）が2009年策定
ECMAScriptのEdition6（ES6）が2015年策定

古いJavaScriptの本だとES5ベースで書かれている場合が多いが、
今回はES6を使用。

Javascriptプログラミングの基礎

WebIDEで練習

WebIDE

https://js.do/

JavaScript リファレンス

https://developer.mozilla.org/ja/docs/Web/JavaScript/Reference

以下に記載する二つのコードの意味が分かればこの章は終わり

```
const sum = (first, second) => first + second;
document.write(sum(4, 2));
```

```
const items = ['リンゴ', 'ゴリラ', 'ラッパ'];
items.forEach(item => document.write(`${item}, `));
```

Javascriptプログラミングの基礎
変数宣言

以下の例はいずれも「userId」が「1」であると宣言しているものだが、少し違いがある。

- let userId = 1

→ソースの他の場所でuserIdに2などを入れる事ができる。

- const userId = 1

→ソースの他の場所でuserIdに2などを入れようとするとエラーになる。

基本的にはconstを使用、再代入が必要な限られた場面でだけletを使用するのが理想（constの方が安全）

Javascriptプログラミングの基礎

変数宣言

IDEで以下の二つのコードを実行してみよう。

```
let userId = 1
userId = 2
document.write(userId);
```

```
const userId = 1
userId = 2
document.write(userId);
```

左では右側「Result」の空間に「2」と現れ

右の場合は画面上部にエラーが表示される。

他にも「var」という宣言方法(letやconstの仲間)もあるが、今回は使いません。

Javascriptプログラミングの基礎
型

稀にJavaScriptはCやJavaと違って型が無いという人がいますが嘘です。

JavaScriptにもCやJavaと同じように「number」,「string」などの型があります。スクリプト実行時に必要に応じてデータ型が自動的に変換されるだけ。

→動的型付け言語(他にはRuby, PHP, Pythonなど)

参考：

https://developer.mozilla.org/ja/docs/Web/JavaScript/Guide/Grammar_and_types

https://qiita.com/mod_poppo/items/a4bbed44ccfa59740f32

Javascriptプログラミングの基礎
変数宣言

IDEで以下のコードを実行してみよう。

typeof 演算子

https://developer.mozilla.org/ja/docs/Web/JavaScript/Reference/Operators/typeof

```
const userId = 1
document.write(typeof userId);
// numberが出力される
```

```
const userId = '1'
document.write(typeof userId);
// stringが出力される。
```

```
const userId = 1
const message = `userIDは${a}`
document.write(typeof message);
// stringが出力される。
```

```
const isAdmin = true
document.write(typeof isAdmin);
// booleanが出力される。
```

Javascriptプログラミングの基礎
関数

プログラミング経験者なら苦しまずに理解できるはず。

https://developer.mozilla.org/ja/docs/Web/JavaScript/Guide/Functions

```
function square(number) {
  return number * number;
}
document.write(square(2));
// 4が出力される。
```

Javascriptプログラミングの基礎

アロー関数

本当は通常の関数と様々な違いがありますが、ひとまず「関数を短縮して書ける」と覚えておいてください。

ちゃんと知りたい方はこちら

https://developer.mozilla.org/ja/docs/Web/JavaScript/Reference/arrow_functions

```
const square = number => number * number;
document.write(square(2));
// 4が出力される。
```

Javascriptプログラミングの基礎
確認①

今ならこれ、説明できそうじゃない？

```
const sum = (first, second) => first + second;
document.write(sum(4, 2));
```

Javascriptプログラミングの基礎

配列

凄くまとまった良い記事があるので、これをベースに練習

JavaScriptの配列の使い方まとめ。要素の追加,結合,取得,削除。

https://qiita.com/takeharu/items/d75f96f81ff83680013f

Javascriptプログラミングの基礎

繰り返し

まずは教科書的なfor文

```
const numbers = [1, 3, 5];
for(let i = 0; i < numbers.length; i++) {
  document.write(`${numbers[i]},`);
}
```

Javascriptプログラミングの基礎
繰り返し

for-in文

```
const numbers = [1, 3, 5];
for(let number in numbers) {
  document.write(`${number},`);
}
```

Javascriptプログラミングの基礎
繰り返し

ちょっとカッコつけてイテレータとforEach

```
const numbers = [1, 3, 5];
numbers.forEach(number => document.write(`${number},`));
```

イテレータは深いので極めたい方は色々調べてみてください。

参考：

https://qiita.com/kura07/items/cf168a7ea20e8c2554c6

Javascriptプログラミングの基礎
確認②

勝ったな（確信）

```
const items = ['リンゴ', 'ゴリラ', 'ラッパ'];
items.forEach(item => document.write(`${item},`));
```

Reactの基礎

Reactの概要

Reactとは？

Facebookが2013年に公開した新しいJavaScriptライブラリ

Reatc Nativeに挑戦する上で避けては通れない要素。

特徴

- コンポネント指向を実現しやすくする
  →テストしやすい、ソースコード全体としての見通しが良くなる
- データが更新されると自動的に画面を再描画
- Virtual DOMという仕組みで差分だけの再描画を実現
- jsxという独自シンタックスを使う => babel等で変換が必要

Reactの概要

Reactとは？

従来のWeb開発とReactでの開発の違い
（勝手なイメージ）

↑従来のWeb開発(jQueryなど)
基礎(HTML)と足場(セレクタ)をベース開発

←React(どちらかと言えばMaterial Design)
部品を組み合わせて開発

Reactの概要

従来のWeb開発のアプローチ

index.html

```
<body>
  <div id="content">
    <p id="hello"></p>
    <p id="access-counter"></p>
  </div>
</body>
```

出力結果

Hello Tom

あなたは12番目の訪問者です。

index.js

```
$(function(){
  // APIアクセスとかをしているものとする
  const getUserName = () => 'Tom';
  const getVisitorCount = () => 12;
  $("#hello").text(`Hello ${getUserName()}`);
  $("#access-counter").text(`あなたは${getVisitorCount()}番目の訪問者です`);
});
```

Reactのアプローチ

index.html

```
<body id=“root">
</body>
```

index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import TopPage from './TopPage';

ReactDOM.render(
  <TopPage />,
  document.getElementById('root')
)
```

TopPage.jsx

```
import React from 'react';

export default class TopPage extends React.Component {
  getUserName = () => ‘Tom‘;
  getVisitorCount = () => 12;
  render() {
    return (
      <div>
        <p>{`Hello ${this.getUserName()}`}</p>
        <p>{`あなたは${this.getVisitorCount()}番目の訪問
者です`}</p>
      </div>
    );
  }
}
```

Reactの概要

Reactのアプローチ

Reactのアプローチによるメリット・デメリット

- メリット
  - トップページのことはTopPage.jsxに聞けば良い、他のファイルを漁る必要がない（勿論最終的には表示に関するロジック以外は分離する→Fluxパターン）
  - React.Componentを継承することで、stateと呼ばれる状態の変化に応じて自動でViewが再描画されるようになる。

    →コンポーネントのライフサイクル

- デメリット
  - jQueryなどと比較して導入も学習も難易度が高い&面倒くさい
  - stateが変わるとrender()が自動で走るので、何も知らずに作ると性能問題が起こりやすい

    → https://qiita.com/muraikenta/items/21eb5b2e0d1c7c95e3b8

休憩

時間が余ったらアドリブでReactの練習もするかも
https://stackblitz.com/edit/react-qjpfms

目次（後半）

- 開発環境の紹介
- 事前準備
- 動作確認
- TODO管理用の付箋アプリを作る
- ビルド方法の紹介
- 一日目のまとめと次回予告

13：00 ～ 17:00を予定

開発環境の紹介

開発環境の紹介

expo.io とは

https://expo.io/

React Nativeでの開発を支援するサービス

- ネットワークを通して実機の動作確認ができる(ケーブルでの接続が不要)
- コードの変更がリアルタイムに反映される
- 開発版アプリの配布がすぐにできる

開発環境の紹介

expo snack とは

https://snack.expo.io/

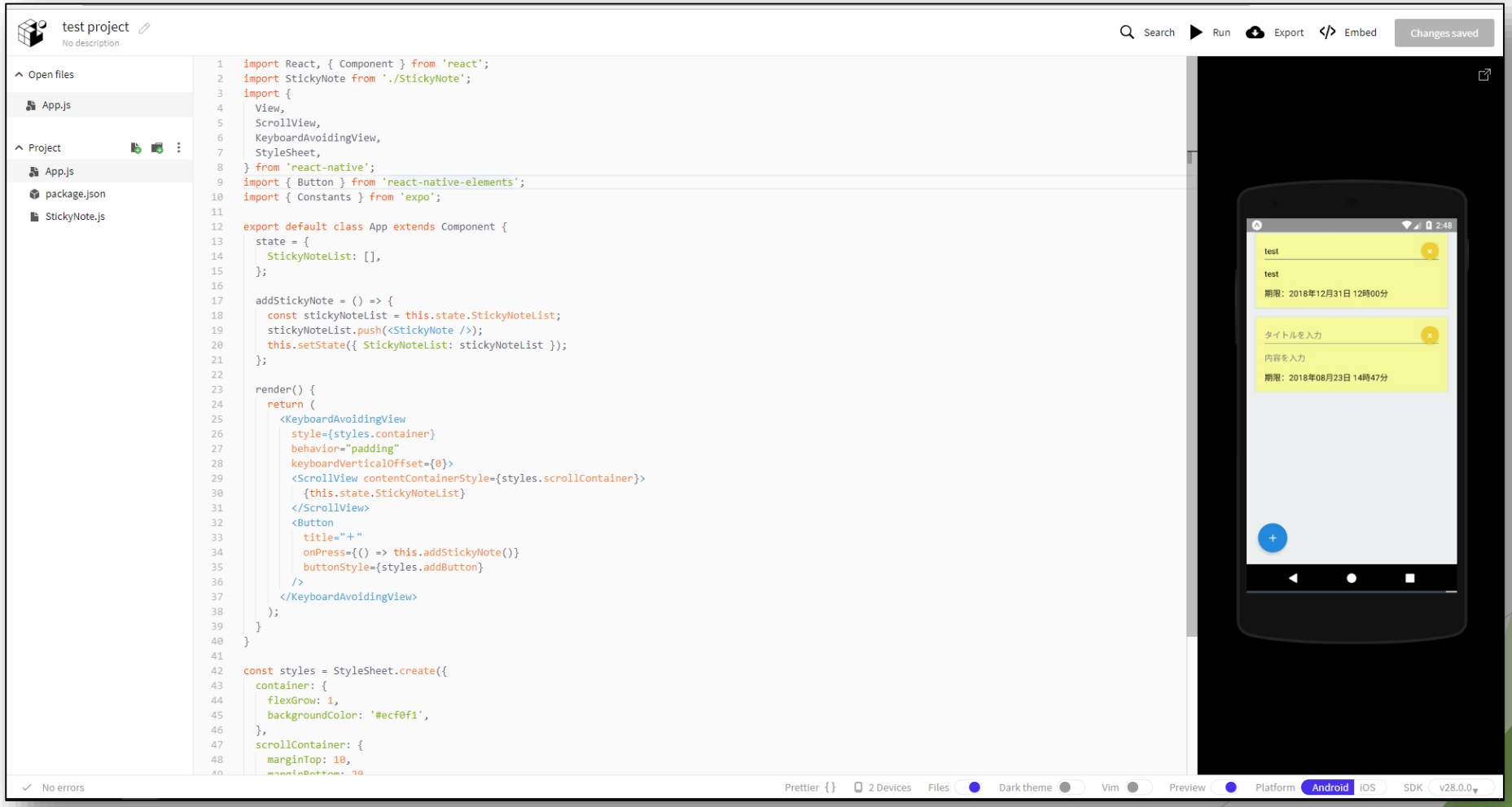
React NativeのwebIDE

- ビルド以外のすべてがWeb上で完結しているので環境構築が不要
- 動作確認用のサーバーも用意されているので、開発マシンと動作確認用のスマホが同じネットワークにある必要がない
- モック作りやハンズオンなどに最適

開発環境の紹介

expo snack（画面）

画面上にはソースコードとエミュレータが表示される。



開発環境の紹介

expo snack（画面）

自分のスマホやタブレットで動作確認をしながら開発可能！

↑ スマホ

← タブレット

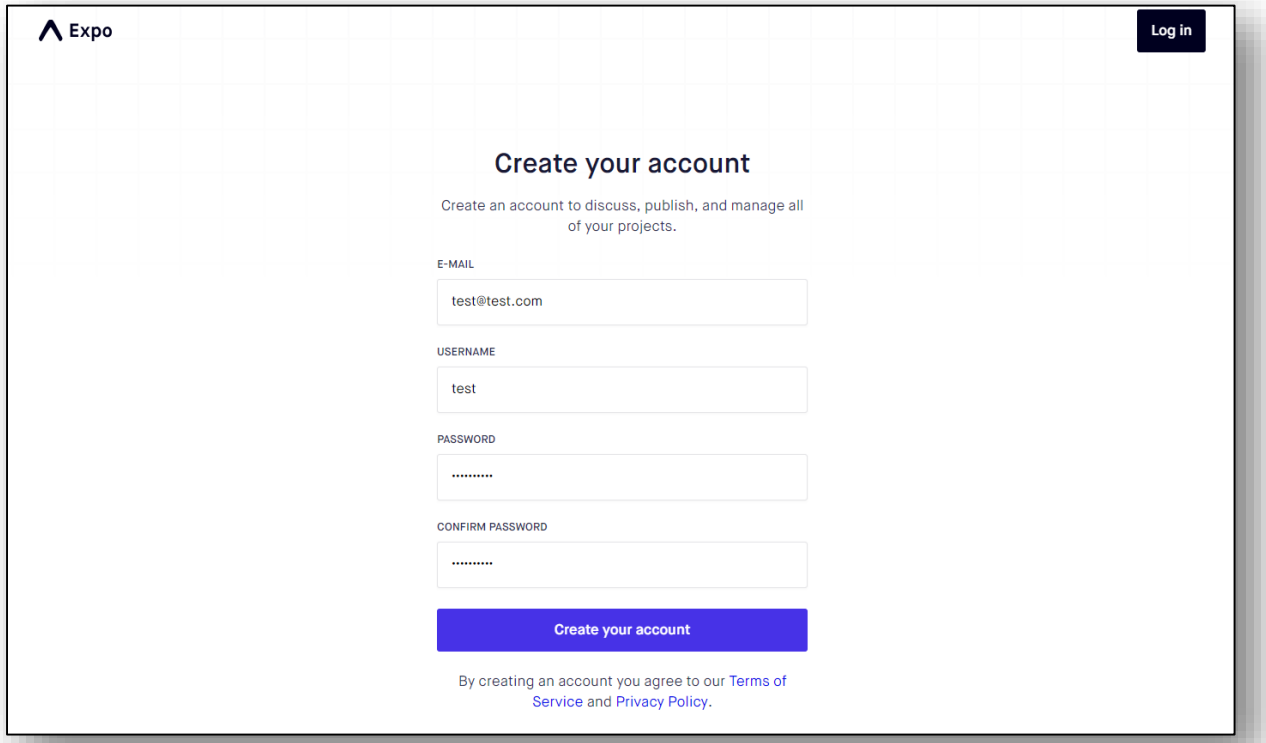
事前準備

事前準備

expo.ioにユーザー登録

https://expo.io/signup

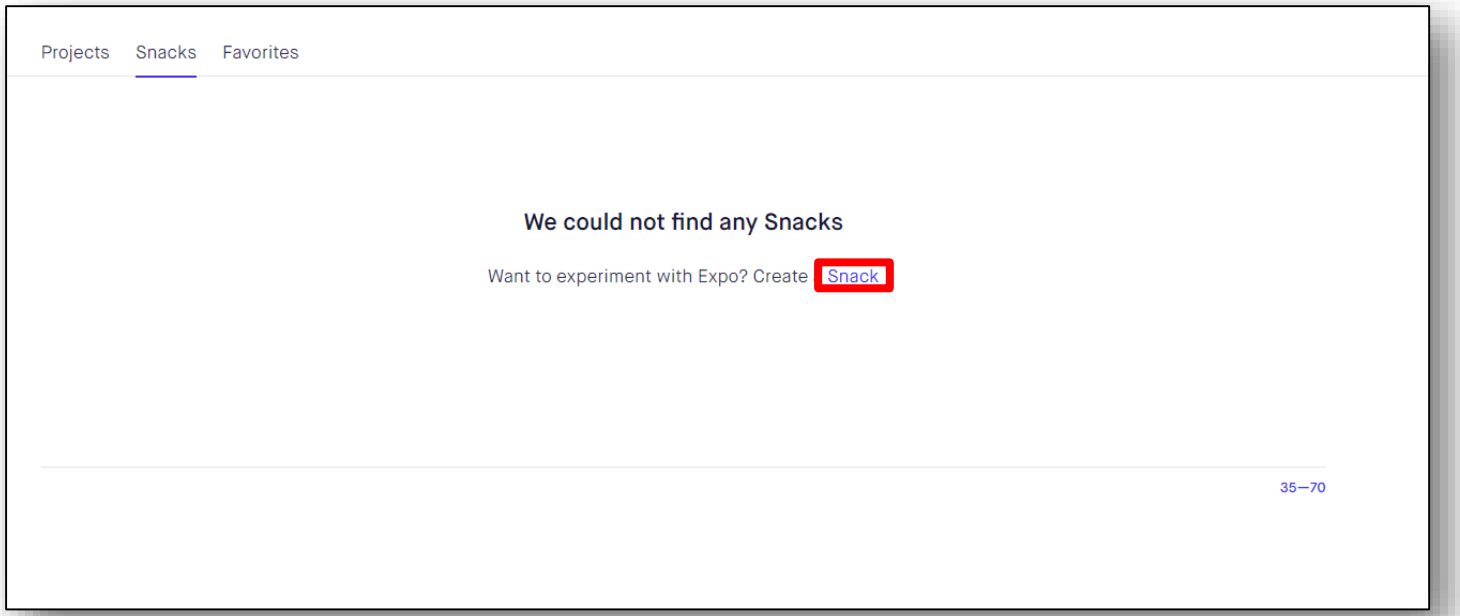
expo.ioのユーザー作成ページにアクセスし、メールアドレス、使用したいユーザー名、パスワードを入力



事前準備

Snackのプロジェクトを作成

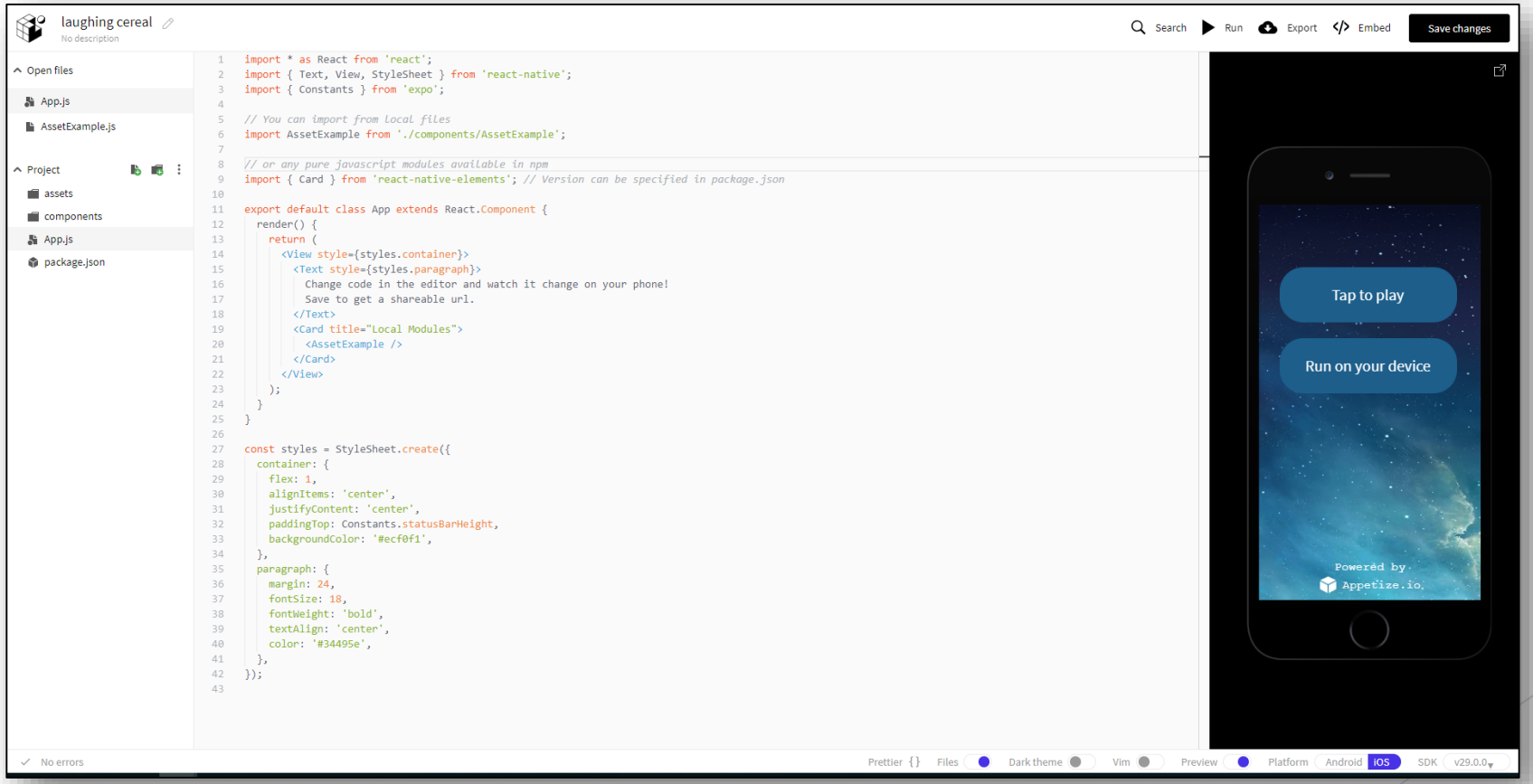
Snacksタブを選択し、画面上にあるSnackへのリンクをクリックする
（以下のキャプチャの枠線部分）



事前準備

Snackのプロジェクトを作成

以下の画像と同じ画面が表示されれば事前準備は完了



事前準備

クライアントアプリの準備

Android版

https://play.google.com/store/apps/details?id=host.exp.exponent&referrer=www

iOS版

https://itunes.apple.com/app/apple-store/id982107779

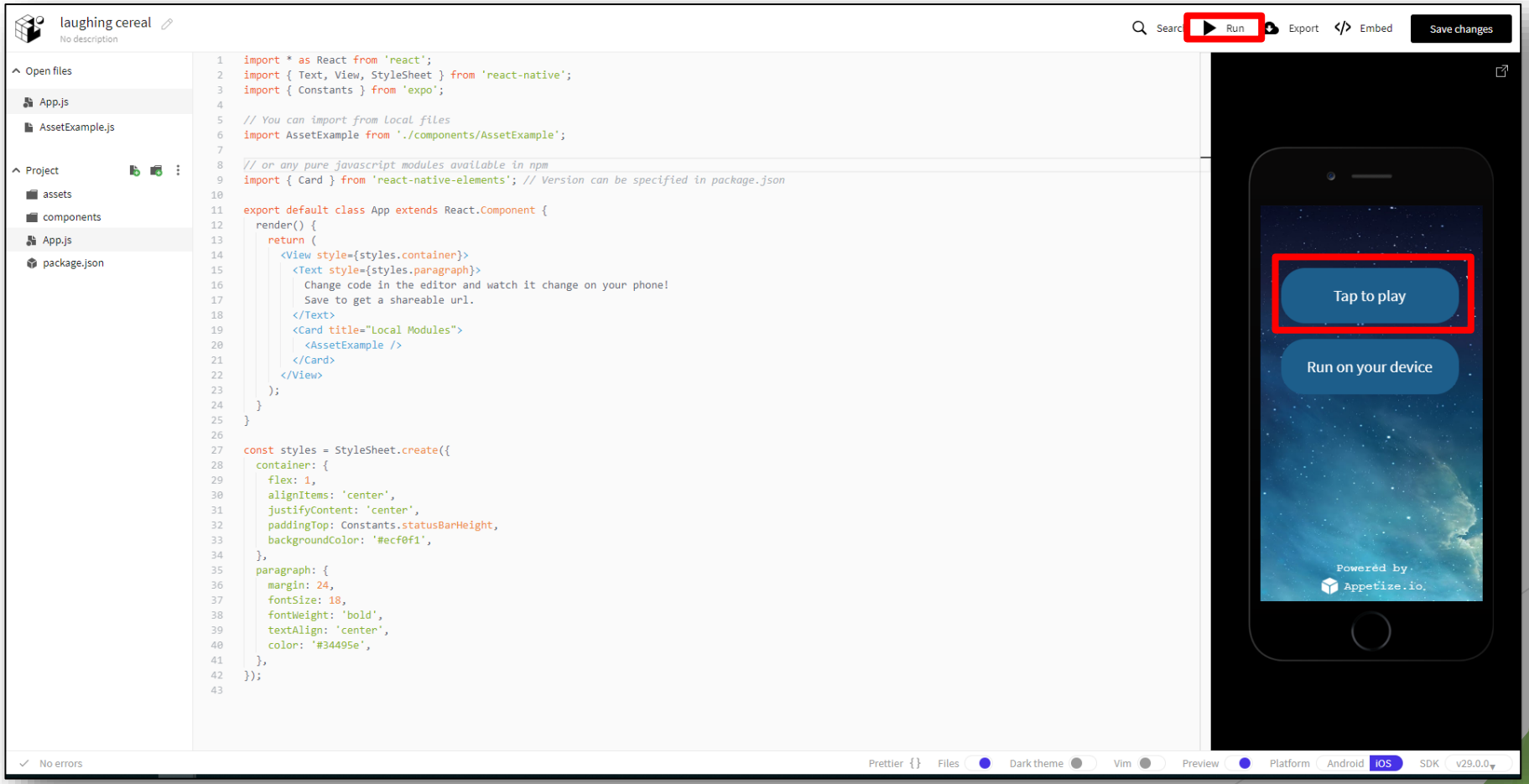
Google PlayまたはApple Storeで「expo」と検索すれば上位に表示される筈なので、インストールする。

動作確認

動作確認

実機でデバッグする

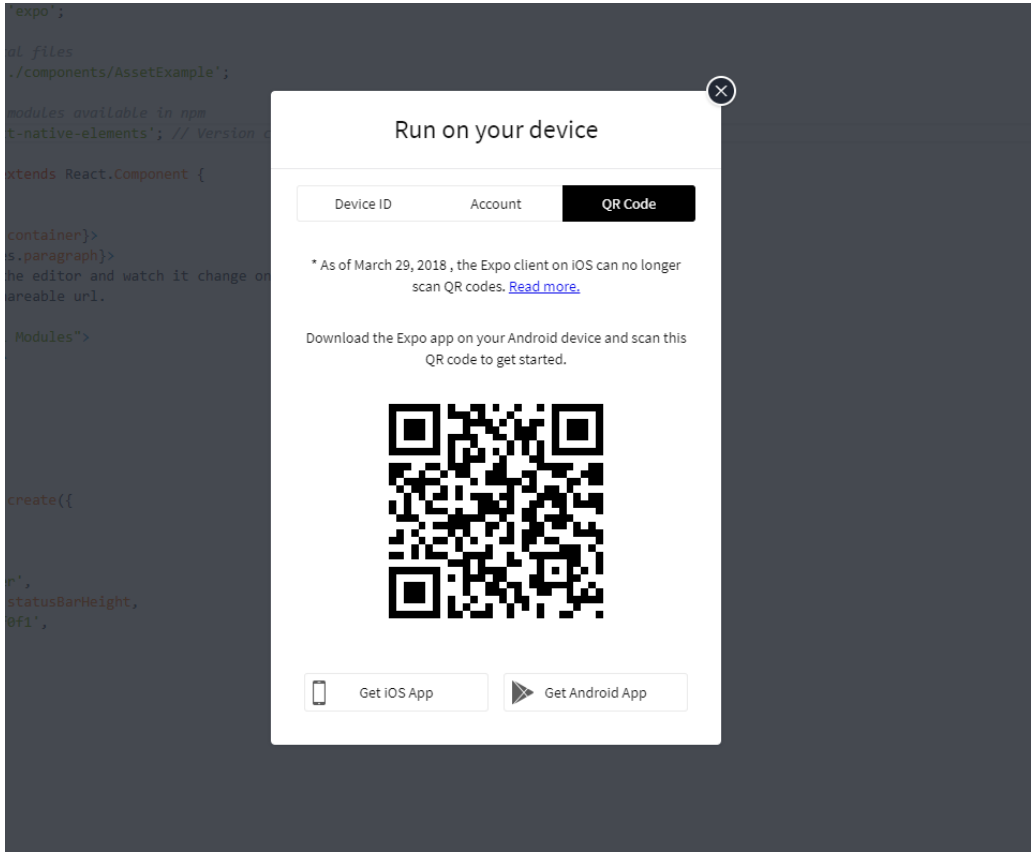
画面上の「Run」ボタン、またはエミュレータの「Tap to play」ボタンをクリックする。

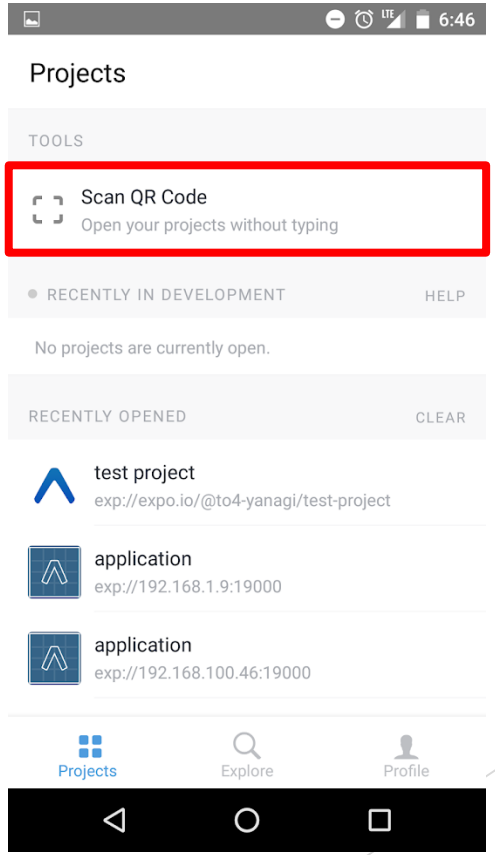


動作確認

実機でデバッグする

画面上にデバッグ用のQRコードが表示されるので、事前準備でインストールしたexpo.ioのクライアントアプリで読み込む

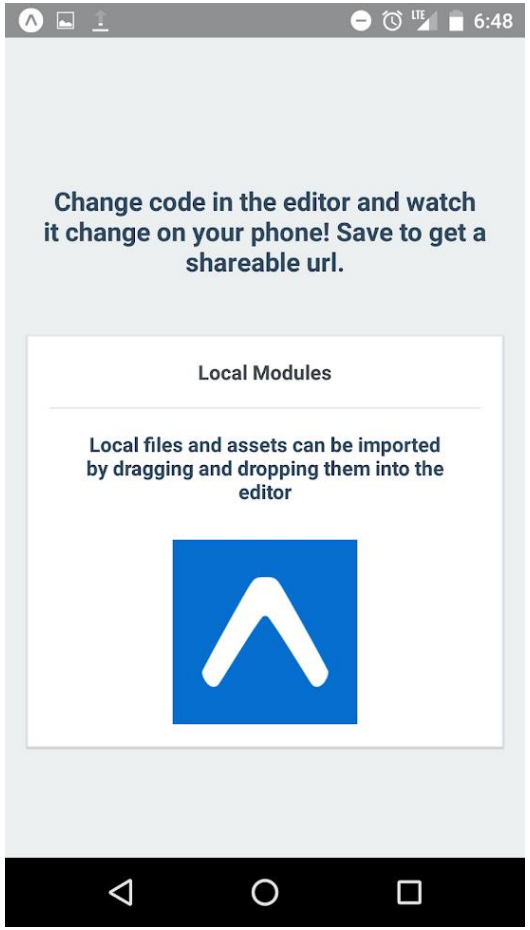




動作確認

実機でデバッグする

以下のキャプチャと同じ画面が表示されるはず。

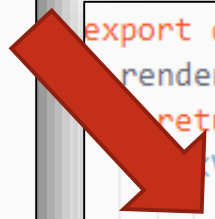


動作確認

リアルタイムで反映される事を確認

ソースコードの変更がリアルタイムで自身のスマホに反映される事を確認するためApp.jsのrender()内を以下のように変更する。

```
export default class App extends React.Component {
  render() {
    return (
      <View style={styles.container}>
        <Text style={styles.paragraph}>
          Change code in the editor and watch it change on your phone!
          Save to get a shareable url.
        </Text>
        <Card title="Local Modules">
          <AssetExample />
        </Card>
      </View>
    );
  }
}
```

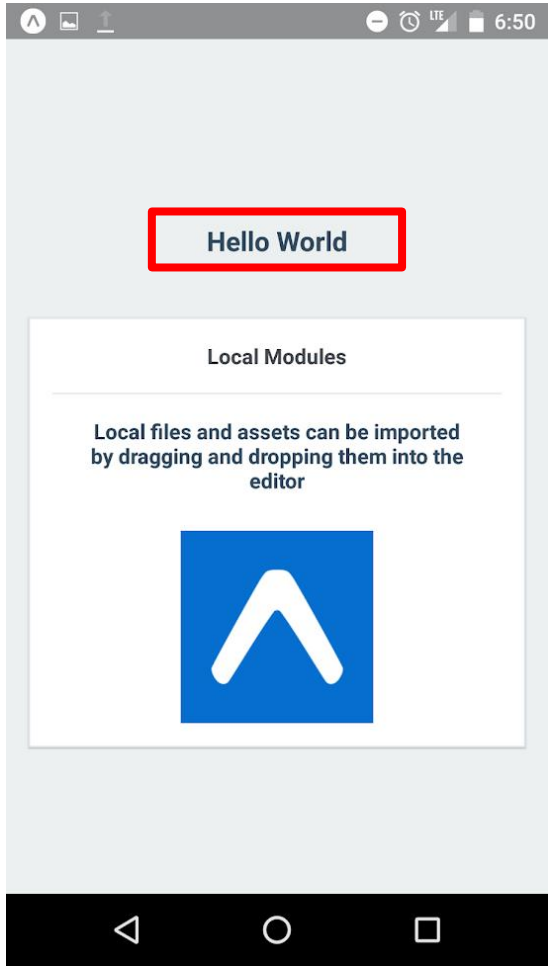


```
export default class App extends React.Component {
  render() {
    return (
      <View style={styles.container}>
        <Text style={styles.paragraph}>
          Hello World
        </Text>
        <Card title="Local Modules">
          <AssetExample />
        </Card>
      </View>
    );
  }
}
```

動作確認

リアルタイムで反映される事を確認

画面を表示した時のメッセージが「Hello World」に変化している事が確認できる。

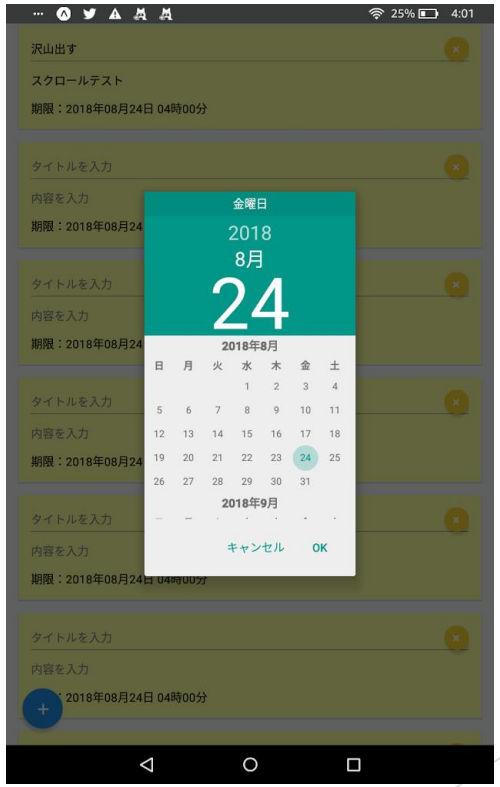


TODO管理用の付箋アプリを作る

TODO管理用の付箋アプリを作る

要件（どんなものを作るか）

- スマホの画面上にやるべき事を書いた付箋を追加・削除できるアプリ
- 付箋上に表示する項目は「タイトル」「内容」「期限」のラベルと入力フォーム
- 「＋」ボタンを押すと画面上に付箋が現れる
- 付箋上の「×」ボタンを押すと付箋が消える
- 編集は各入力フォームをタップで行う
- 「タイトル」「内容」はユーザーによる入力
- 「期限」はdatetimepickerによる選択
- データの保存機能はなし



TODO管理用の付箋アプリを作る

基本的な考え方

コンポネントは以下の二つだけで実現

（ちゃんと作るならばもうちょっと分割すべきだけど難易度上がるので…）

①画面全体（EntryPoint）

②付箋部分

TODO管理用の付箋アプリを作る

実際に作る

今回のインターンシップで書くコードを纏めたリポジトリ

https://github.com/to4-yanagi/study_work/commits/master

コミットログ

https://github.com/aselab/internship-201809/commits/master

以降はコミットログにある順番で「App.js」と「StickyNote.js」を実装する。

実装のために必要になる知識の参考情報は逐次URLを記載してある。

TODO管理用の付箋アプリを作る

最初の土台作り

コミット「initial commit」

ReactのComponentの説明

https://reactjs.org/docs/react-component.html

react-nativeのコンポネントの説明や取りうるプロパティ等は公式のdocs参照

https://facebook.github.io/react-native/docs/text

CSS in JS

https://facebook.github.io/react-native/docs/stylesheet

TODO管理用の付箋アプリを作る

付箋のひな型作り

コミット「付箋の土台を作成」

es6のexportとimportについて

https://qiita.com/bakira/items/1e41f1530c49e36a9197

Cardの説明

https://react-native-training.github.io/react-native-elements/docs/card.html

TODO管理用の付箋アプリを作る

付箋作り

コミット「メモ内容の入力フォームを実装」「タイトルの入力フォームを実装」

package.jsonのreact-native-elementsの部分を以下のように書き換える

"react-native-elements": "1.0.0-beta5"

このようにする理由はCardのコンポネントの「title」にReactComponentを使用出来るのが1.0.0-beta3以降であるため。

https://github.com/react-native-training/react-native-elements/issues/918

TODO管理用の付箋アプリを作る

付箋作り

コミット「メモ内容の入力フォームを実装」「タイトルの入力フォームを実装」

今何気に書き換えたpackage.jsonとはなにか？

https://qiita.com/dondoko-susumu/items/cf252bd6494412ed7847

パッケージ管理ツールnpmについて

https://techacademy.jp/magazine/16105

TODO管理用の付箋アプリを作る

付箋作り

コミット「日時選択ダイアログを実装」

ReactのStateについて

https://qiita.com/sekikawa_a/items/8ab70f457ef73871419f

今回使用するDateTimePicker（別の物を使用しても良いです。）

https://github.com/mmazzarolo/react-native-modal-datetime-picker

Moment.jsについて

https://qiita.com/osakanafish/items/5ef636bbcb2c3ef94953

TODO管理用の付箋アプリを作る

追加処理

コミット「追加処理を実装」

UUIDv4について

https://qiita.com/ta_ta_ta_miya/items/1f8f71db3c1bf2dfb7ea

ビルド方法の紹介

ビルド方法の紹介

ソースコードをダウンロード

Snackの「export」ボタンからこれまでSnack上で実装してきたソースコードをローカルにダウンロードする。（zip形式）

ビルド方法の紹介

npm install

ダウンロードしたzipを展開し、以下のコマンドを実行する。

```
npm i
```

package.jsonとはなにか？
https://qiita.com/dondoko-susumu/items/cf252bd6494412ed7847

パッケージ管理ツールnpmについて
https://techacademy.jp/magazine/16105

ビルド方法の紹介

CLI（expコマンド）からビルド

expというexpoが提供するモジュールがある

↓ドキュメント

https://docs.expo.io/versions/latest/workflow/exp-cli

以下のコマンドでインストール

```
npm i -g exp
```

ビルド方法の紹介

CLI（expコマンド）からビルド

APKファイルなどを作成するにはapp.jsonというファイルを作り、アイコンやbundleIdentifierなどの設定を行う必要がある。

↓ドキュメント

https://docs.expo.io/versions/latest/distribution/building-standalone-apps

expo.ioのプロジェクトとして公開：	exp publish
IPAファイルを作成：	exp build:ios
APKファイルを作成：	exp build:android

ビルド方法の紹介

GUI（expo XDE）からビルド

Expo XDEなるツールがあるので利用する。

↓ドキュメント

https://docs.expo.io/versions/v29.0.0/introduction/xde-tour

一日目のまとめと次回予告

一日目のまとめと次回予告

一日目のまとめ

スマホアプリは意外と作れる

そんな風に思ってもらえるとそれはとっても嬉しいなって

一日目のまとめと次回予告

次回予告

モブプログラミングやります。

https://www.slideshare.net/couger2010/mob-programming-76337184

ASE社員が実施した時の記録。

http://blog.ase.co.jp/2018/05/10/東京支店社内勉強会-モブプログラミングやってみ/

お疲れ様でした。

時間が余ったら二日目に何を作りたいかを決める。

参考

- 【徹底解説】初心者向けスマホアプリ開発に必要な言語と環境まとめ

  https://www.sejuku.net/blog/5293

- Nintendo Switchの中ではReactが動いてる！Nintendo eShop開発秘話を聞いてきた

  https://html5experts.jp/shumpei-shiraishi/24538/

- React.jsとは｜3つの特徴とインストール方法

  https://furien.jp/columns/260/

- 必見！JavaScriptで開発するネイティブアプリ「React Native」とは

  https://furien.jp/columns/259/

- Expo.ioを使ってReact Nativeの開発の準備をする

  https://qiita.com/somebody_gp/items/e018cd4a35a72334b9bb

二日目のタイムスケジュール

https://www.slideshare.net/couger2010/mob-programming-76337184

9:45～12:00	モブプロの簡単な説明 モブプロでスマホアプリを作る
12:00～13:00	昼休み
13:00～16:00	モブプロでスマホアプリを作る