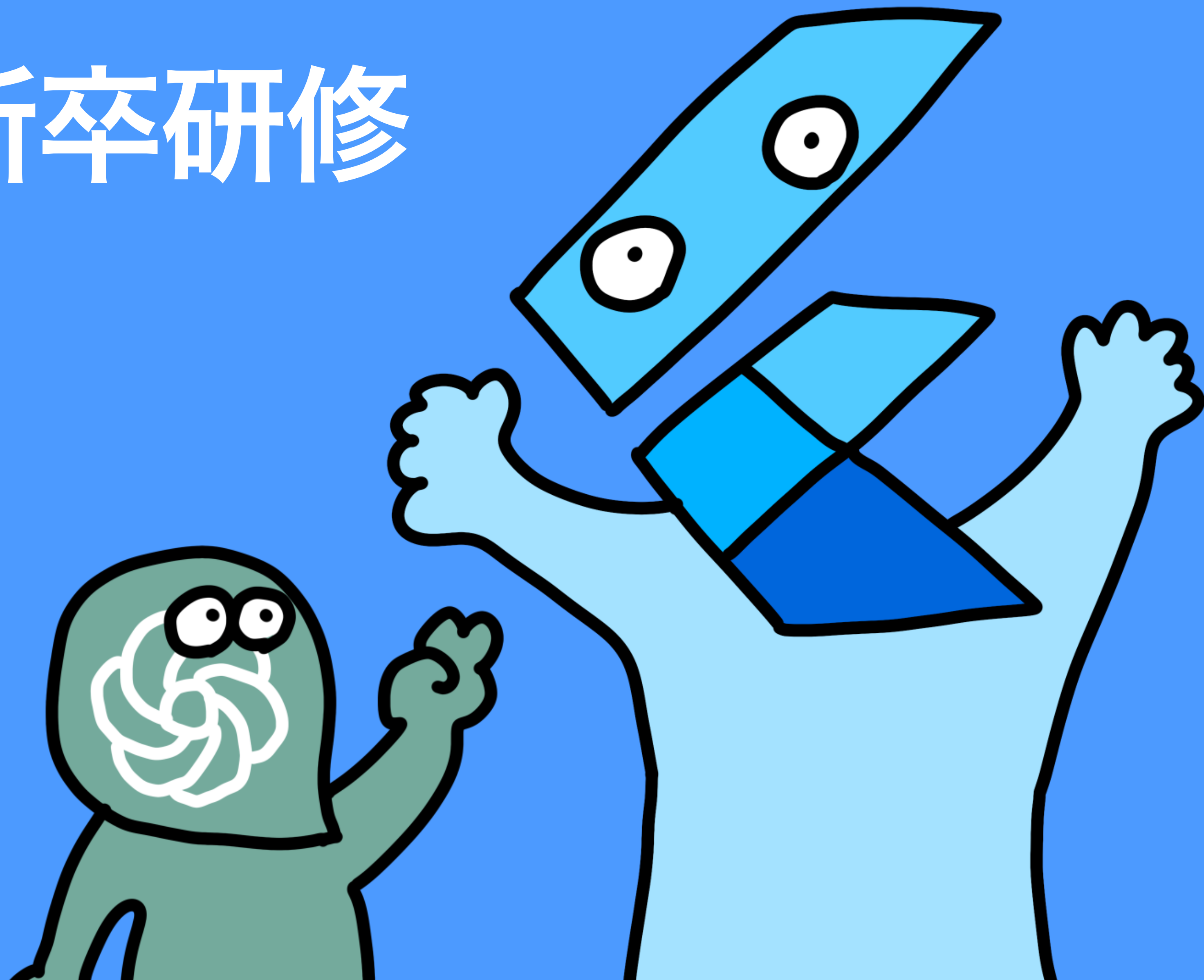


Flutter 23新卒研修

2023/4/28

Kuno Ayana



講師・チューター



久野 文菜

@ayana.kuno

21新卒

miatto / mercury

社食はキーマカレー推し



沼田 幸大

@yukihiro.numata

21年中途入社

Fansta

社食は大体いつも¥600~700

事前準備は OK？

iOS 動かしたい人

- Xcode 14.1 のインストールおよび解凍
- インストールは App Store ではなく developer サイトから落としてくることをお勧め
 - <https://developer.apple.com/download/more/>
- 事前に Apple ID が必要
 - 会社のメールアドレスでアカウントを作成

Android 動かしたい人

- AndroidStudioのインストールおよび解凍
 - 一旦起動しないと flutter doctor したときに Unable to locate Android SDK. のエラーになる

本日のお品書き

10:30 ~ Flutter とは

11:00 ~ flutter create コマンドでアプリを作ってみよう！

11:30 ~ 休憩

11:40 ~ create されたアプリを見ていこう！

#1 Introduction

#2 State

#3 TextField

#4 画面遷移 (Navigator 1.0)

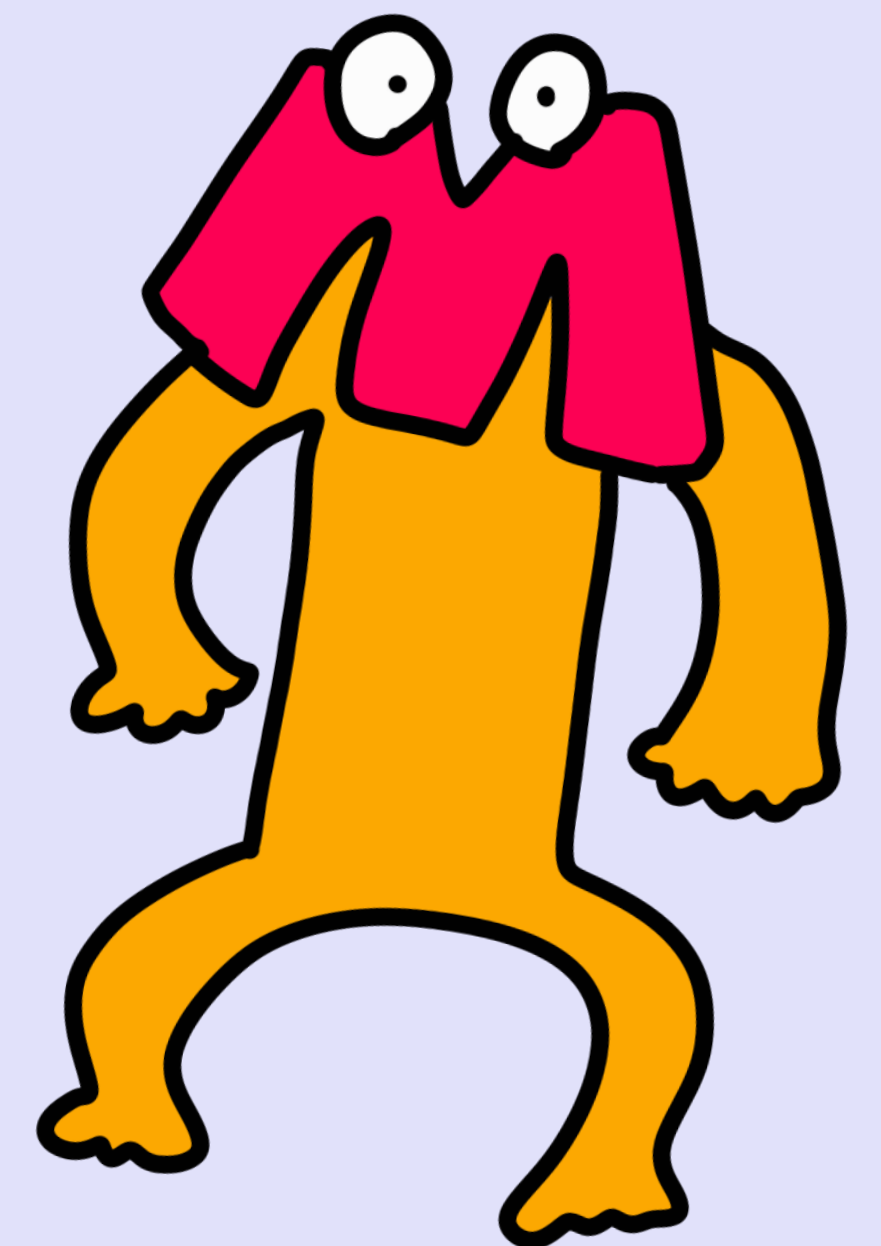
13:00 ~ ランチ

14:00 ~ #5 ネットワーク通信 (http 通信)

15:00 ~ チームでチャレンジ！

ChatGPT とのチャットアプリを作ってみよう！

18:00 ~ クロージング



研修のゴール

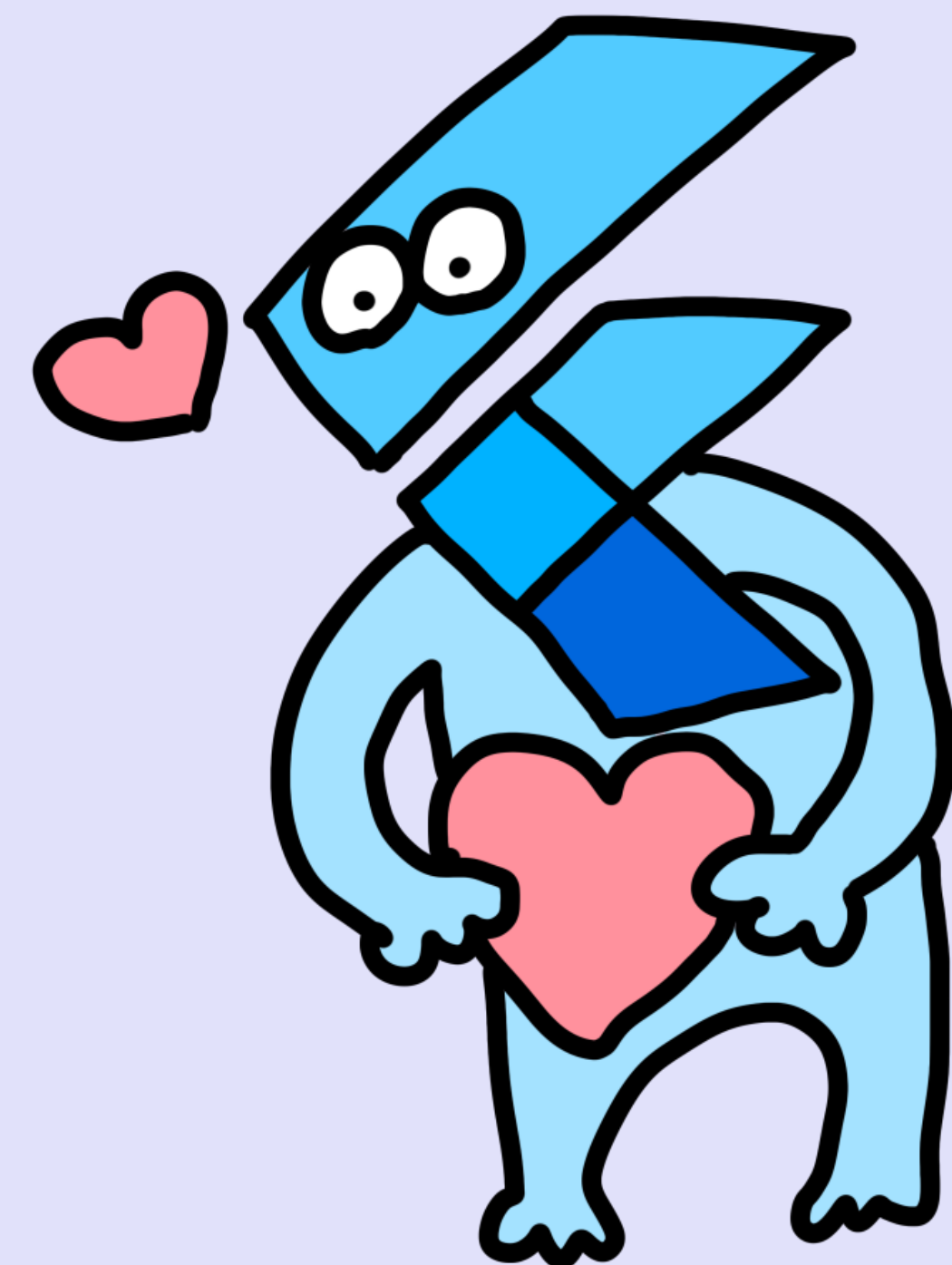
研修のゴール

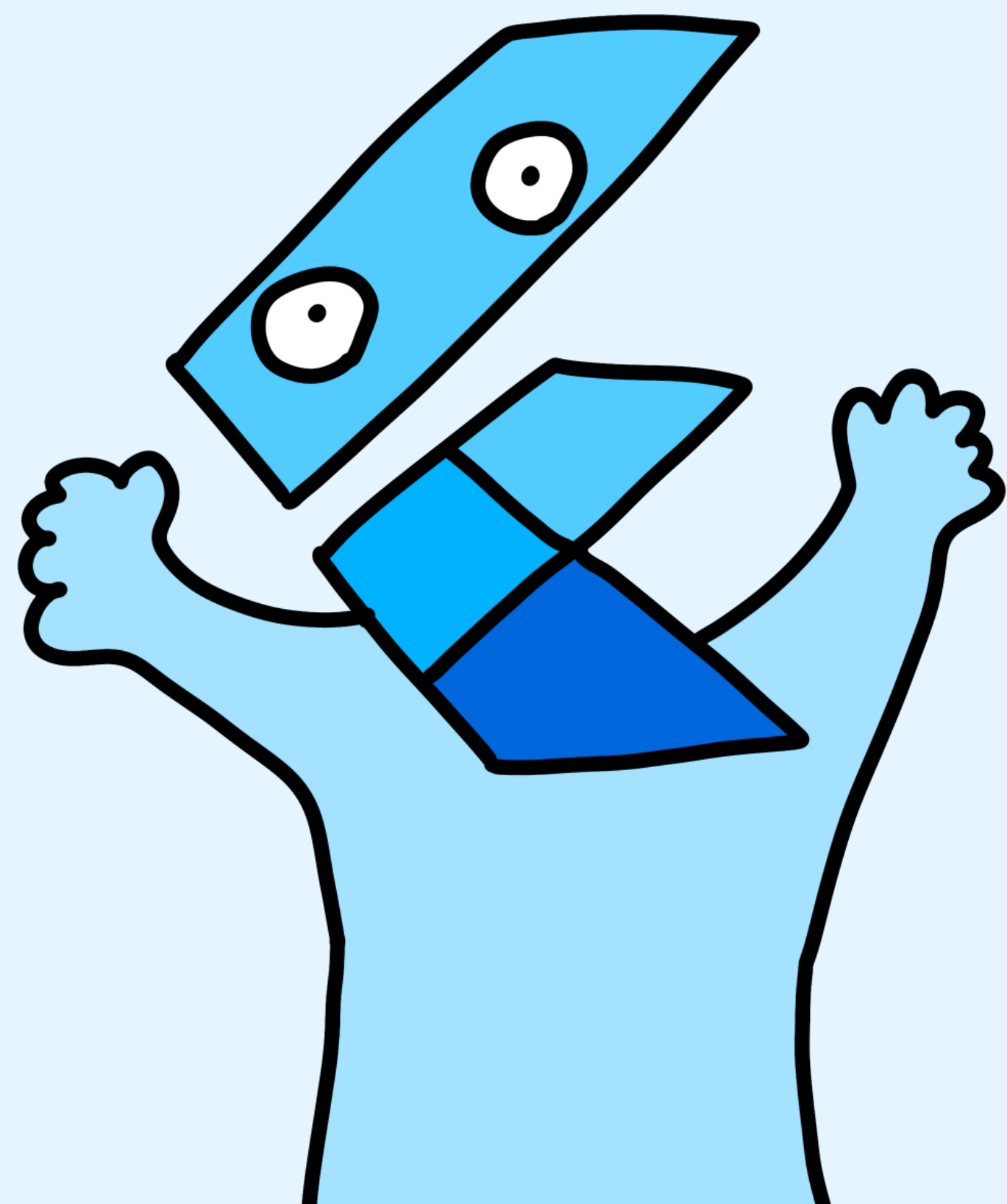
アプリ開発初めての人

Flutter って楽しい～！の気持ちを持ち帰って！

Flutter チョットできる人

**人に教えることで理解度を上げよう！
応用問題まで頑張れ！**

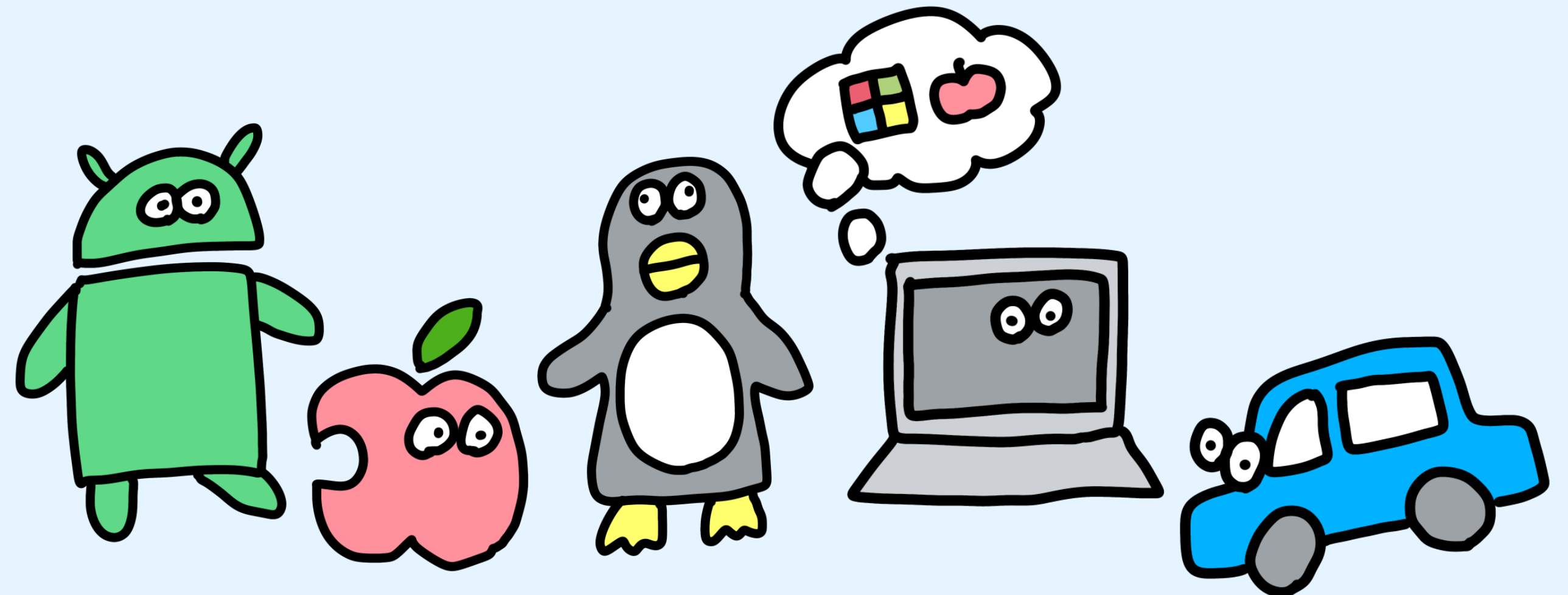




Flutter とは

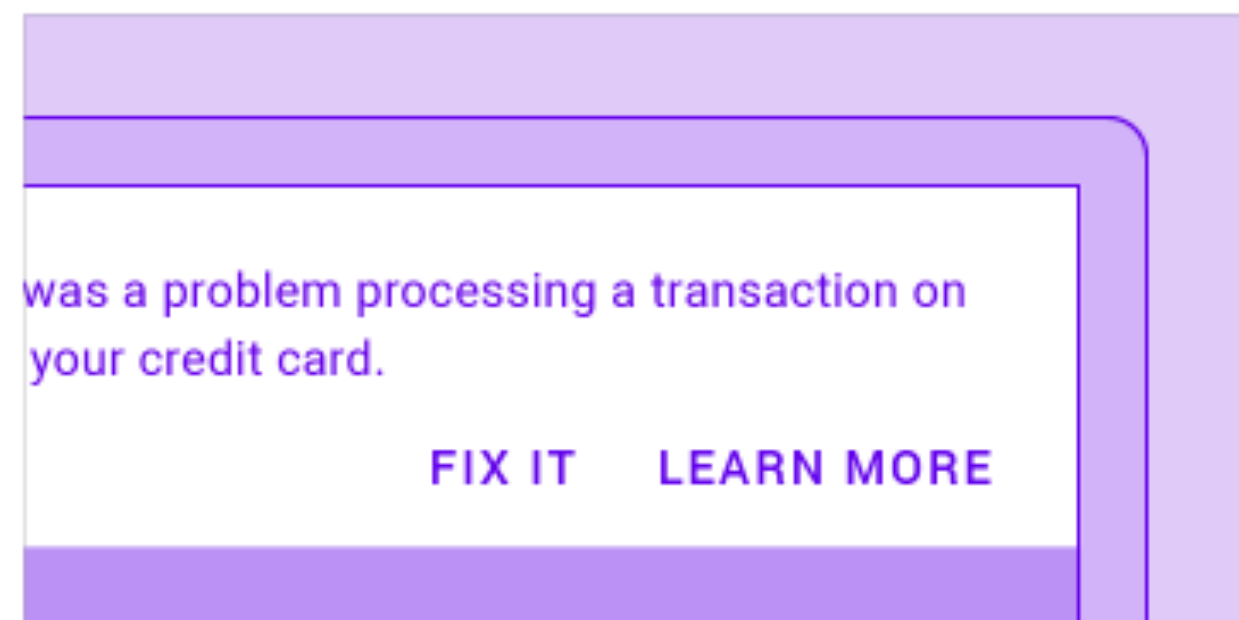
Flutter とは

- **マルチプラットフォーム開発**が可能なフレームワーク
 - 7つのプラットフォーム上で動くアプリを作れる！
- 世界には 500 万人以上の開発者と 70 万以上のアプリがある
- GitHub のコントリビューター数は VSCode, home assistant に次いで **3 位**！
- 3D モデルの表示もできるように！



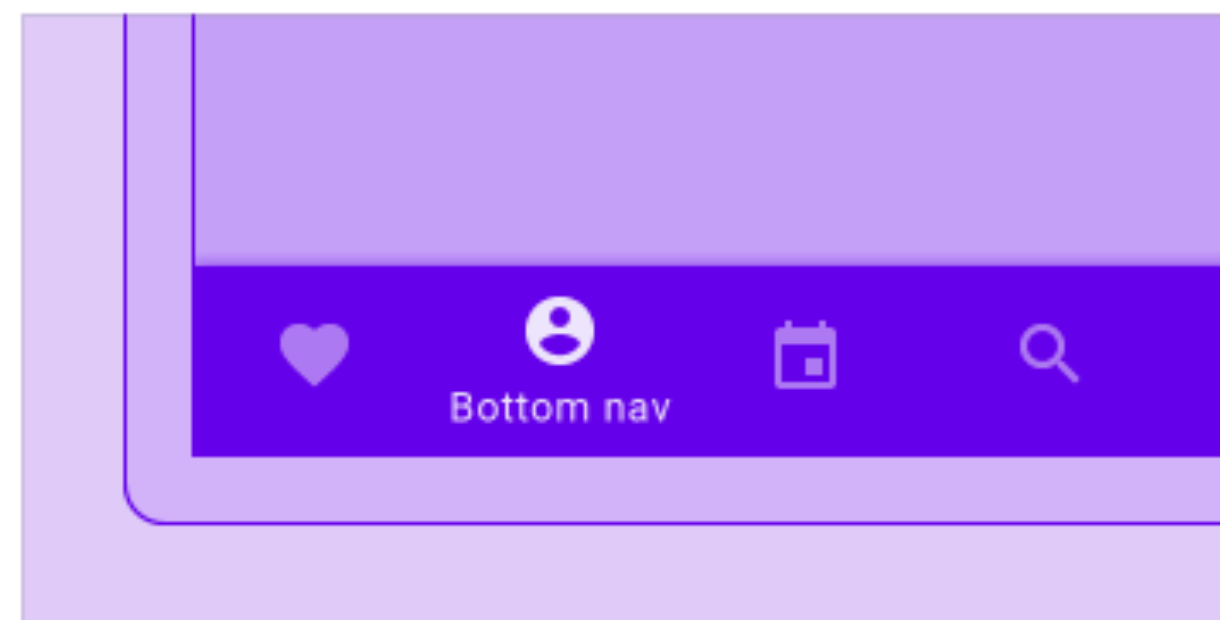
Flutter とは

- Google が本気を出して作ってる
 - Google のアプリ（GoogleAds や Google Assistant など）でも利用されてる
 - **マテリアルデザイン**のコンポーネントを Android より先に実装してたりする
 - **コンポーネントを標準で用意してくれてる**から初心者でも良い感じに作れる！



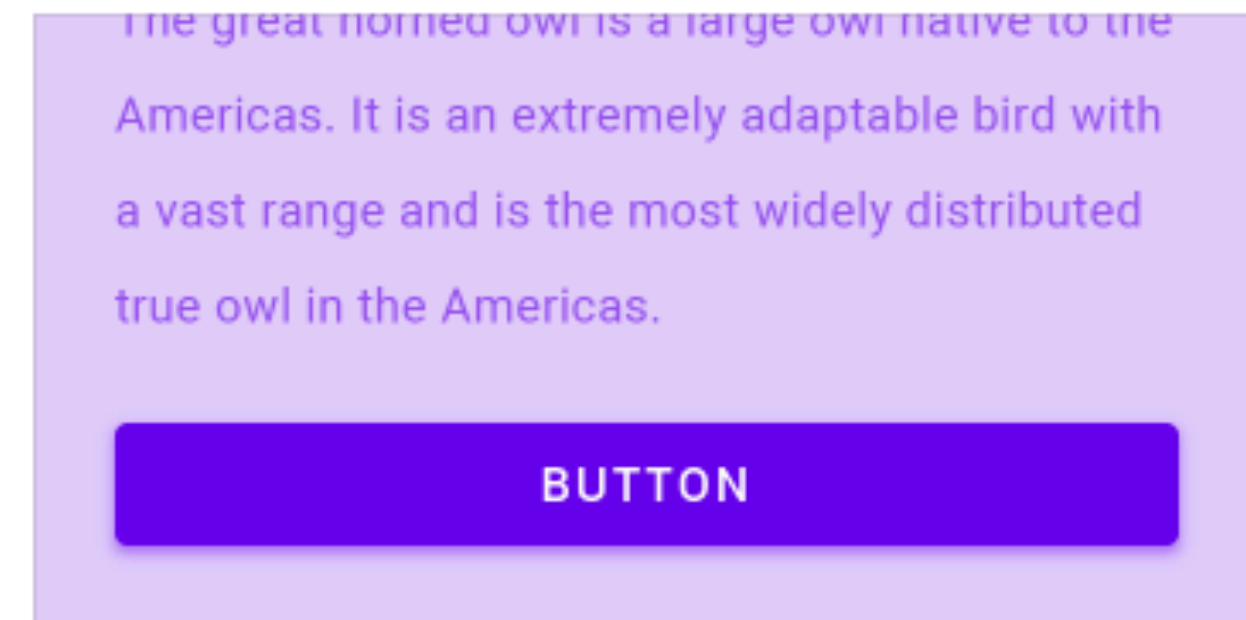
Banners

A banner displays a prominent message and related optional actions



Bottom navigation

Bottom navigation bars allow movement between primary destinations in an app

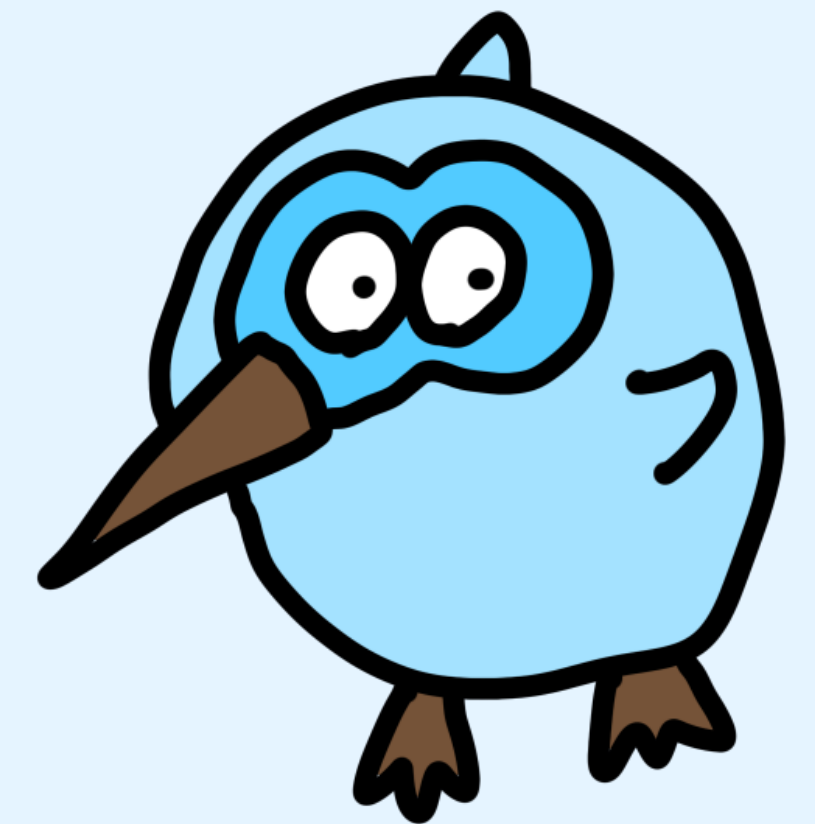


Buttons

Buttons allow users to take actions, and make choices, with a single tap

Flutter とは

- **Dart** という言語で実装する
 - Google が JavaScript の問題点を改善した代替言語として 2011 年に開発されたもの
 - Dart 2.0 で **Null Safety** になった (別ページで説明)
- レンダリングエンジンも Skia を使って自前で作られているので**速い!**
 - Flutter Forward 2023 で新レンダリングエンジンの Impeller が公開された
- **宣言的 UI** で書くことが可能 (別ページで説明)
 - Storyboard を使った Swift のコードより React の方が似てるかも



Flutter のアーキテクチャ: <https://docs.flutter.dev/resources/architectural-overview>

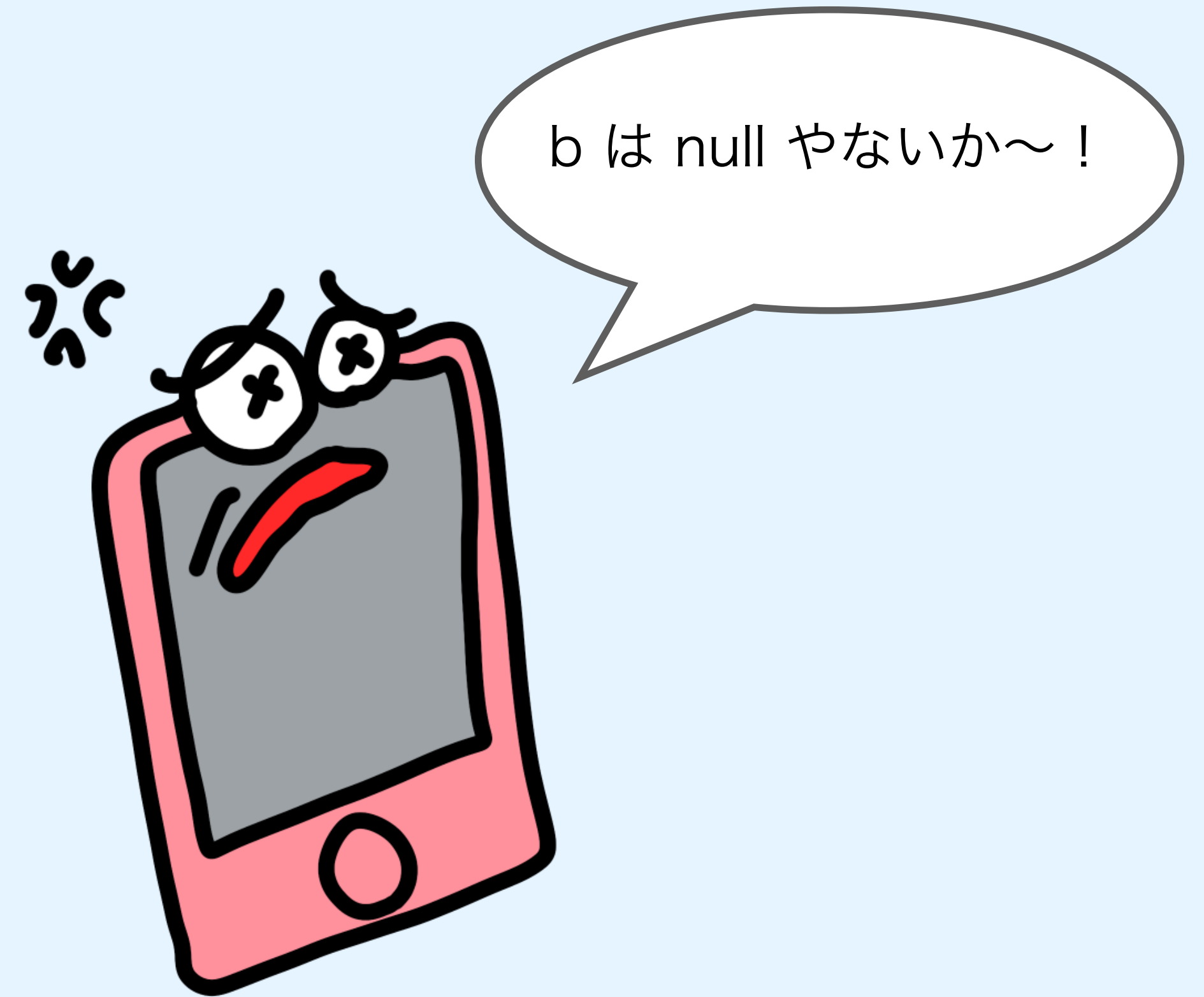
Impeller について: <https://docs.flutter.dev/perf/impeller>

dash 君について: <https://docs.flutter.dev/dash>

Null Safety って？

```
int a = 100;  
int b = null;  
  
a = 100;  
  
print(a + b);
```

どの型でも null を使って良い



実行時に null だった場合
エラーを吐く・アプリが落ちる

Null Safety って？

```
int a = 100;
int? b = null;

a = 100;

print(a + b); // エラー
if (b != null) {
  print(a + b);
}
print(a + b!);
```

基本的に null 使っちゃダメ
使いたい時は **?** をつける



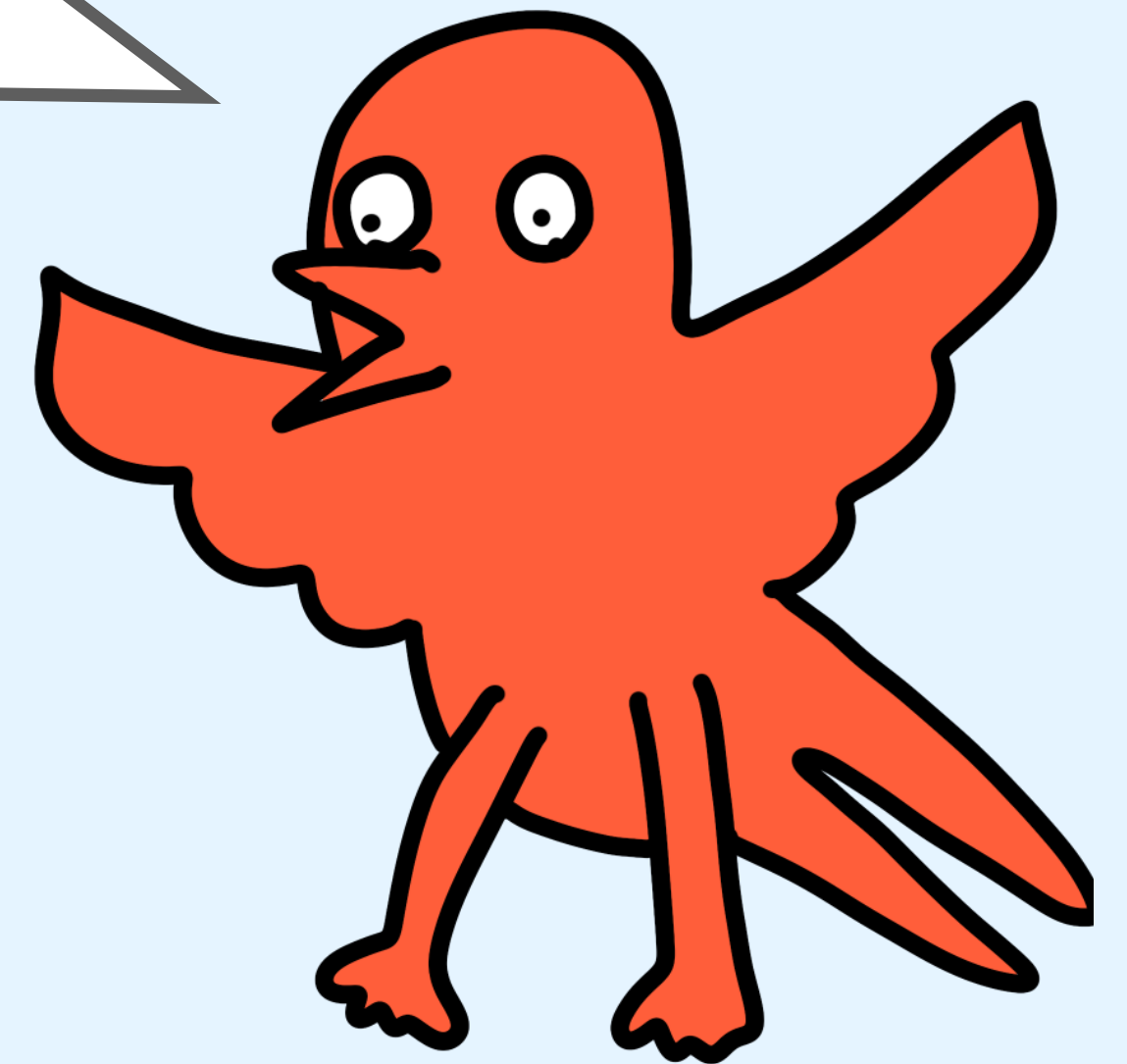
実行時に null のエラーで
落ちることはなくなる

宣言的 UI って？

命令的 (Swift)

```
label = UILabel()  
label.text = "Babu~"  
label.textColor = .black  
label.textAlignment = .center  
view.addSubview(label)
```

ラベルを作れ！
ラベルのテキストはバブ~であれ！
色は黒であれ！真ん中寄せであれ！
そのラベルをViewに追加せえ！



宣言的 (Dart)

```
Text(  
  "Babu~",  
  style: TextStyle(color: Colors.black),  
  textAlign: TextAlign.center,  
),
```


宣言的 UI って？

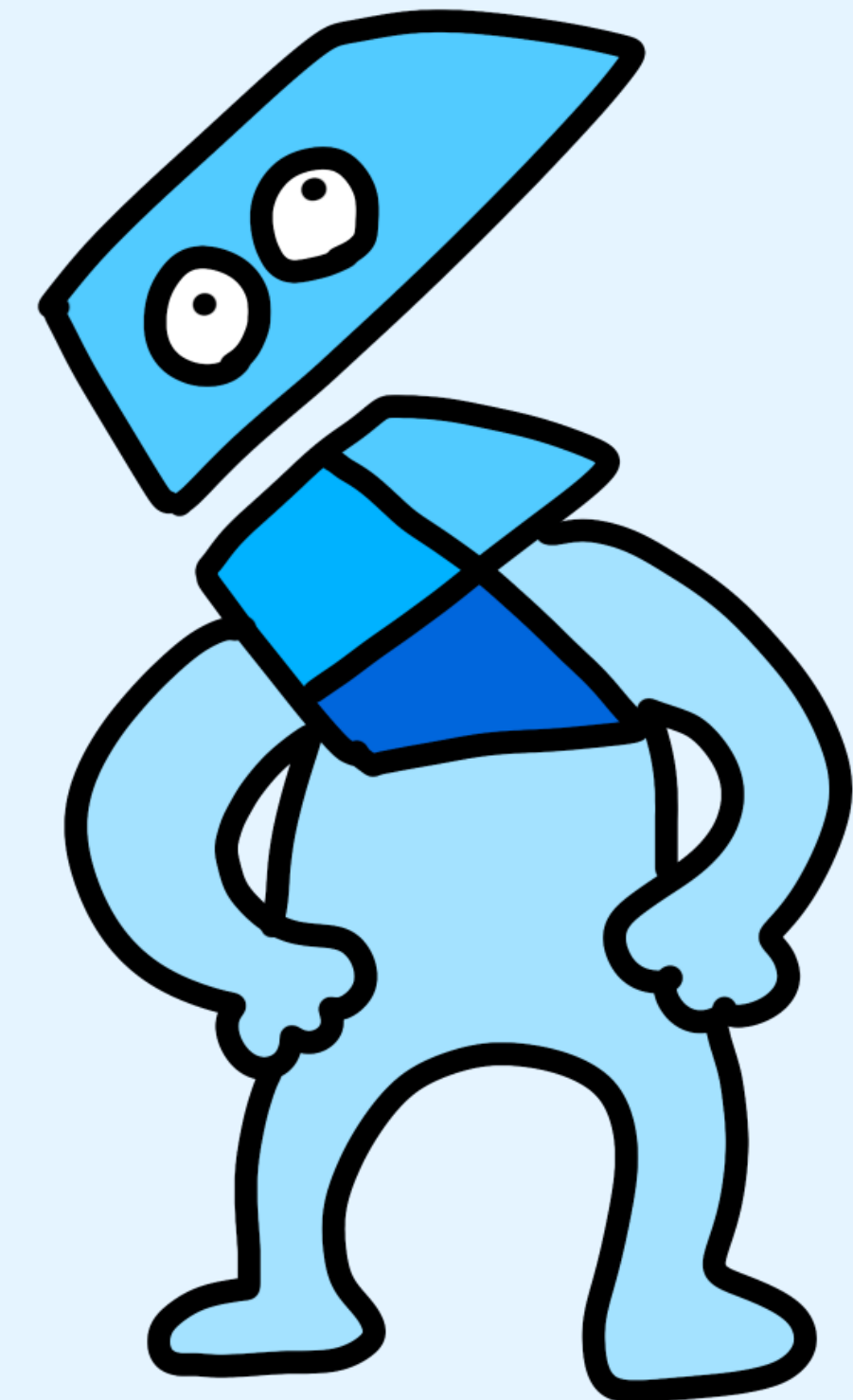
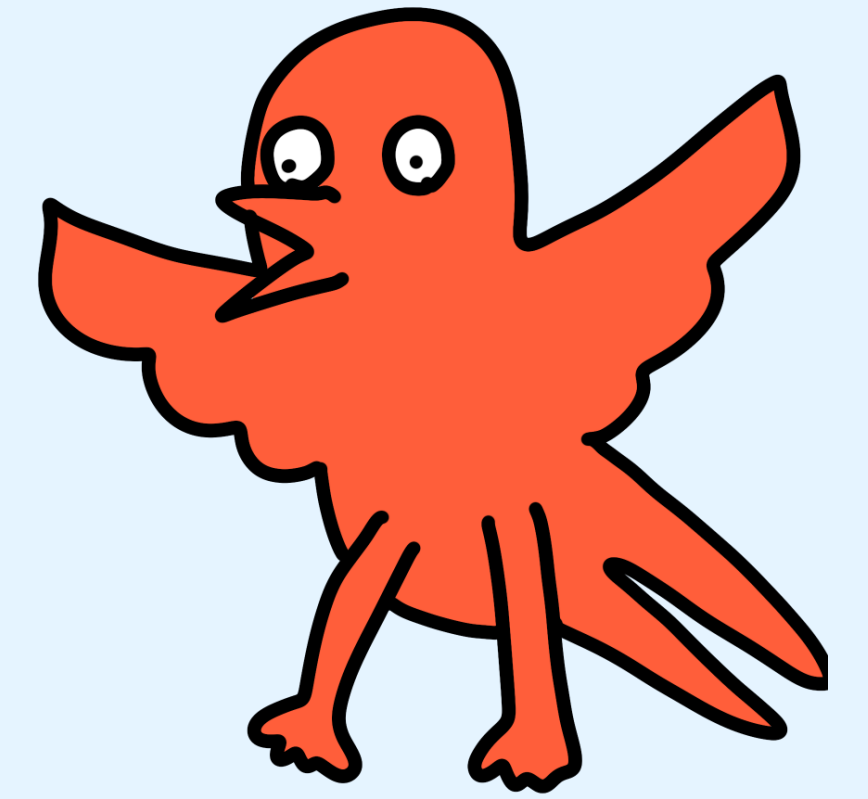
命令的 (Swift)

```
label = UILabel()  
label.text = "Babu~"  
label.textColor = .black  
label.textAlignment = .center  
view.addSubview(label)
```

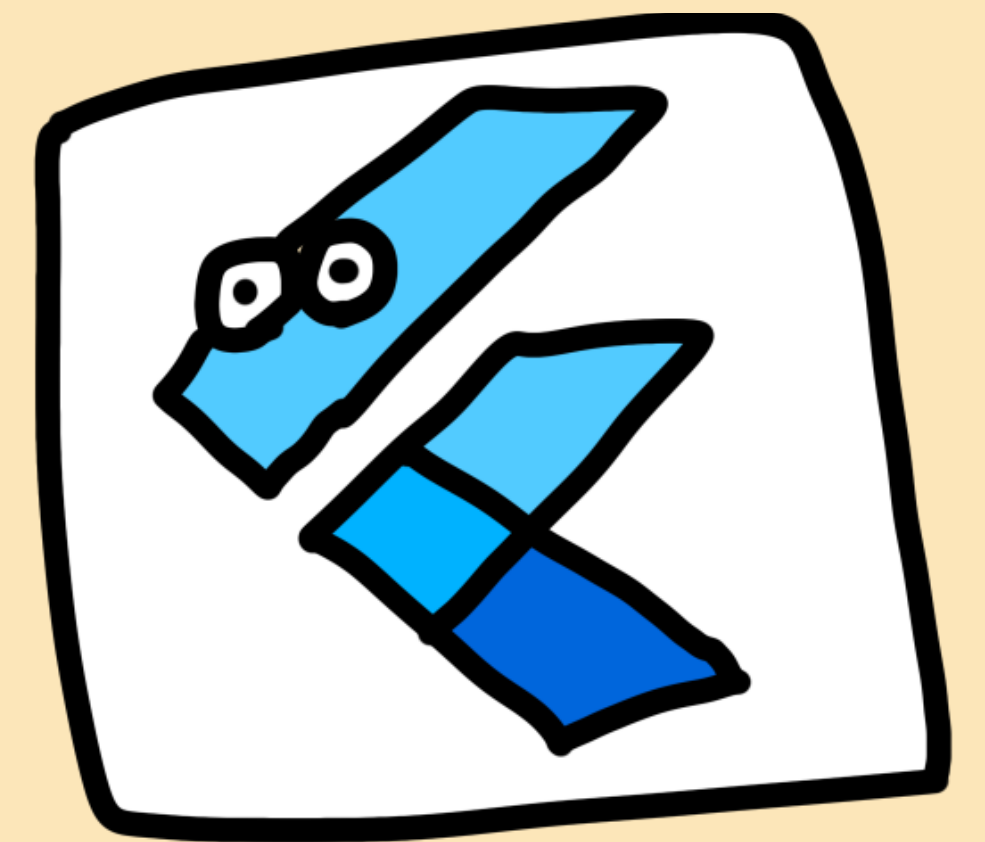
私はバブ~という黒色の
真ん中寄せテキストである！

宣言的 (Dart)

```
Text(  
  "Babu~",  
  style: TextStyle(color: Colors.black),  
  textAlign: TextAlign.center,  
),
```



flutter create コマンドで
アプリを作ってみよう！



アプリを作ってみよう

プロジェクト名はアンダバーで繋がないと
『"プロジェクト名" is not a valid Dart package name.』
ってエラー吐いて怒られる

Android の人は実機選択もしくは
Android スタジオ立ち上げてエミュ選択

```
$ flutter create <<プロジェクトの名前>>  
$ cd <<プロジェクトの名前>>  
$ open iOS/Runner.xcworkspace // ios の人のみ  
$ flutter run
```

iOS 派の人

The screenshot shows the Xcode interface for a project named "Runner" by "team/use-hive-generator". The top status bar indicates "Build Succeeded | 2023/04/12 at 15:24". The left sidebar shows the project structure with folders for Flutter, Runner, and Products, and files like Main, Assets, LaunchScreen, Info, GeneratedPluginRegistrant, AppDelegate, and Runner-Bridging-Header. The main pane is set to the "Signing & Capabilities" tab for the "Runner" target. The "Signing" section is expanded, showing "Automatically manage signing" checked, with a note that Xcode will create and update certificates, app IDs, and provisioning profiles. The "Team" is set to "MIXI, Inc." and the "Bundle Identifier" is "com.example.chatSample". The "iOS" section shows the "Provisioning Profile" as "Xcode Managed Profile" and the "Signing Certificate" as "Apple Development: ayana kuno (9RNH4HTC92)". A speech bubble with the text "事前準備で作ったアカウントになっているか?" (Is it using an account prepared in advance?) points to the "Team" field. At the bottom, a note says "Add capabilities by clicking the '+' button above."

Runner
team/use-hive-generator

Runner > KNO3

Build Succeeded | 2023/04/12 at 15:24

Runner

General Signing & Capabilities Resource Tags Info

+ Capability All Debug Release Profile

PROJECT
Runner

TARGETS
Runner

Automatically manage signing
Xcode will create and update certificates, app IDs, and provisioning profiles.

Team MIXI, Inc.

Bundle Identifier com.example.chatSample

iOS

Provisioning Profile Xcode Managed Profile ⓘ

Signing Certificate Apple Development: ayana kuno (9RNH4HTC92)

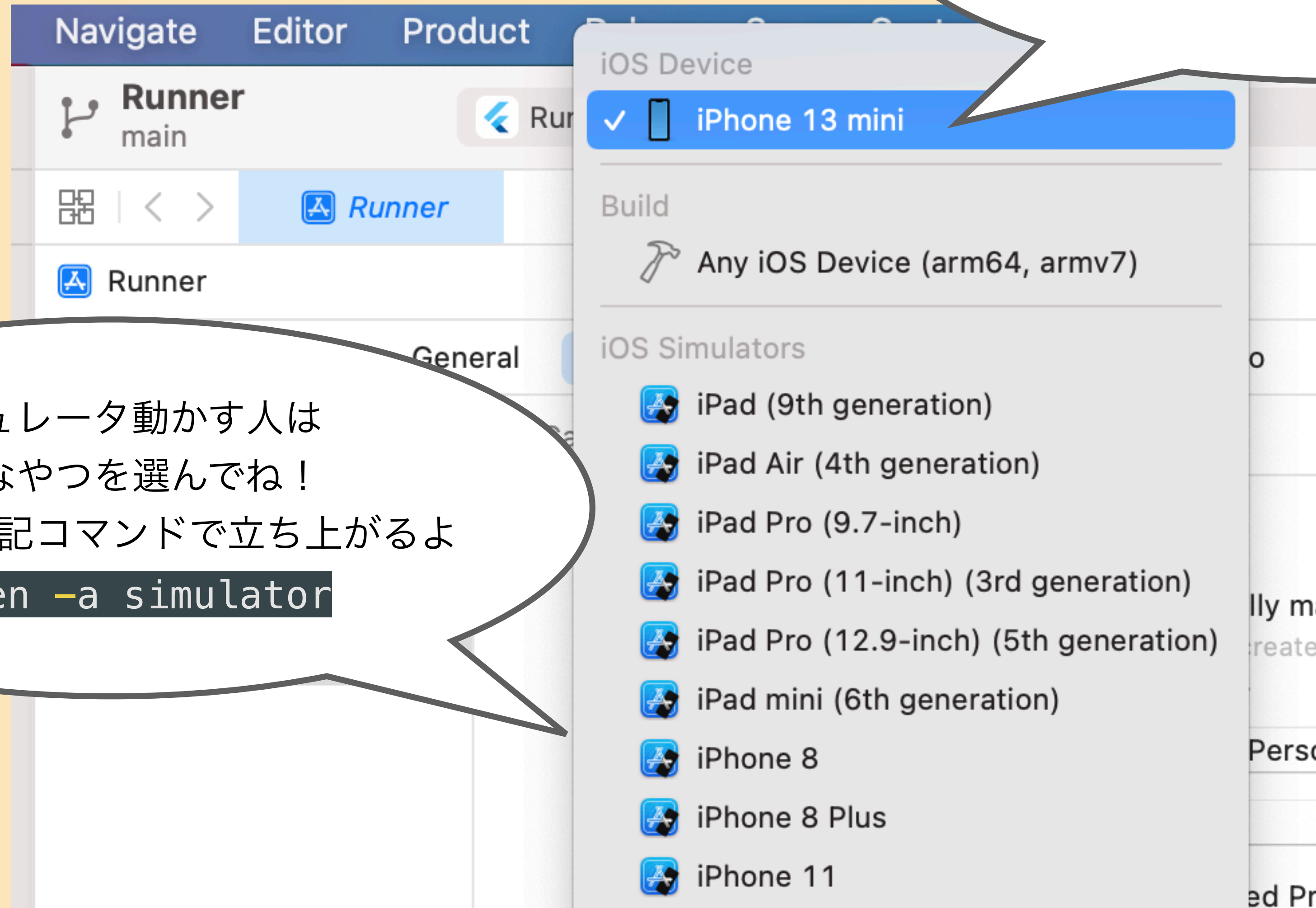
Add capabilities by clicking the "+" button above.

事前準備で作った
アカウントになっているか？

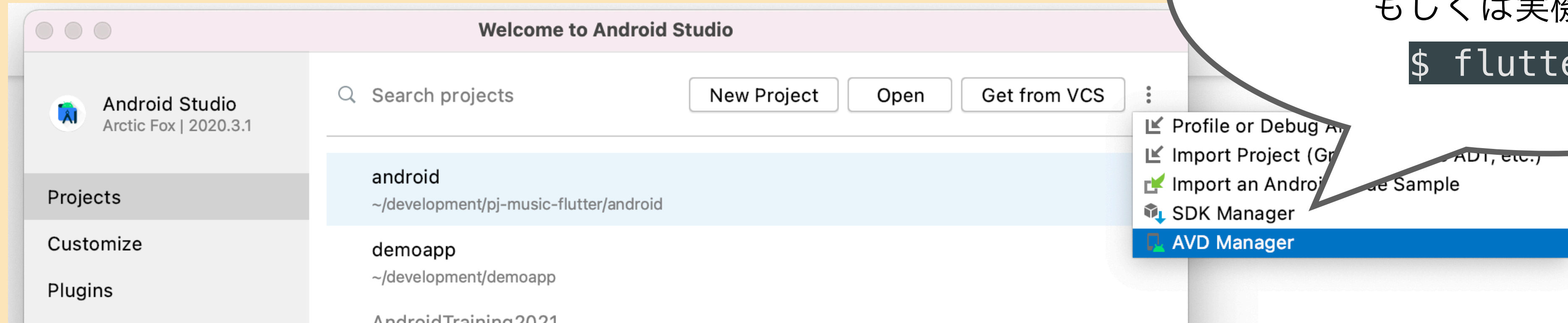
iOS 派の人

実機接続してたら iOS Device のところに
こんな風に出てくるはず

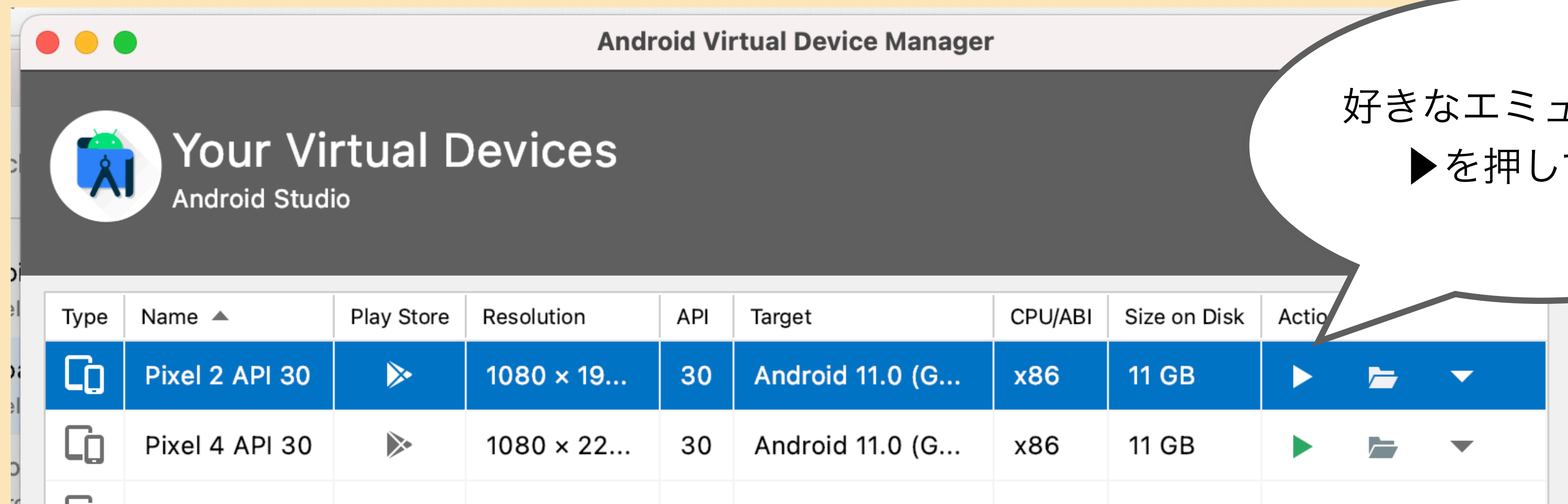
シミュレータ動かす人は
好きなやつを選んでね！
次回からは下記コマンドで立ち上がるよ
`$ open -a simulator`



Android 派の人



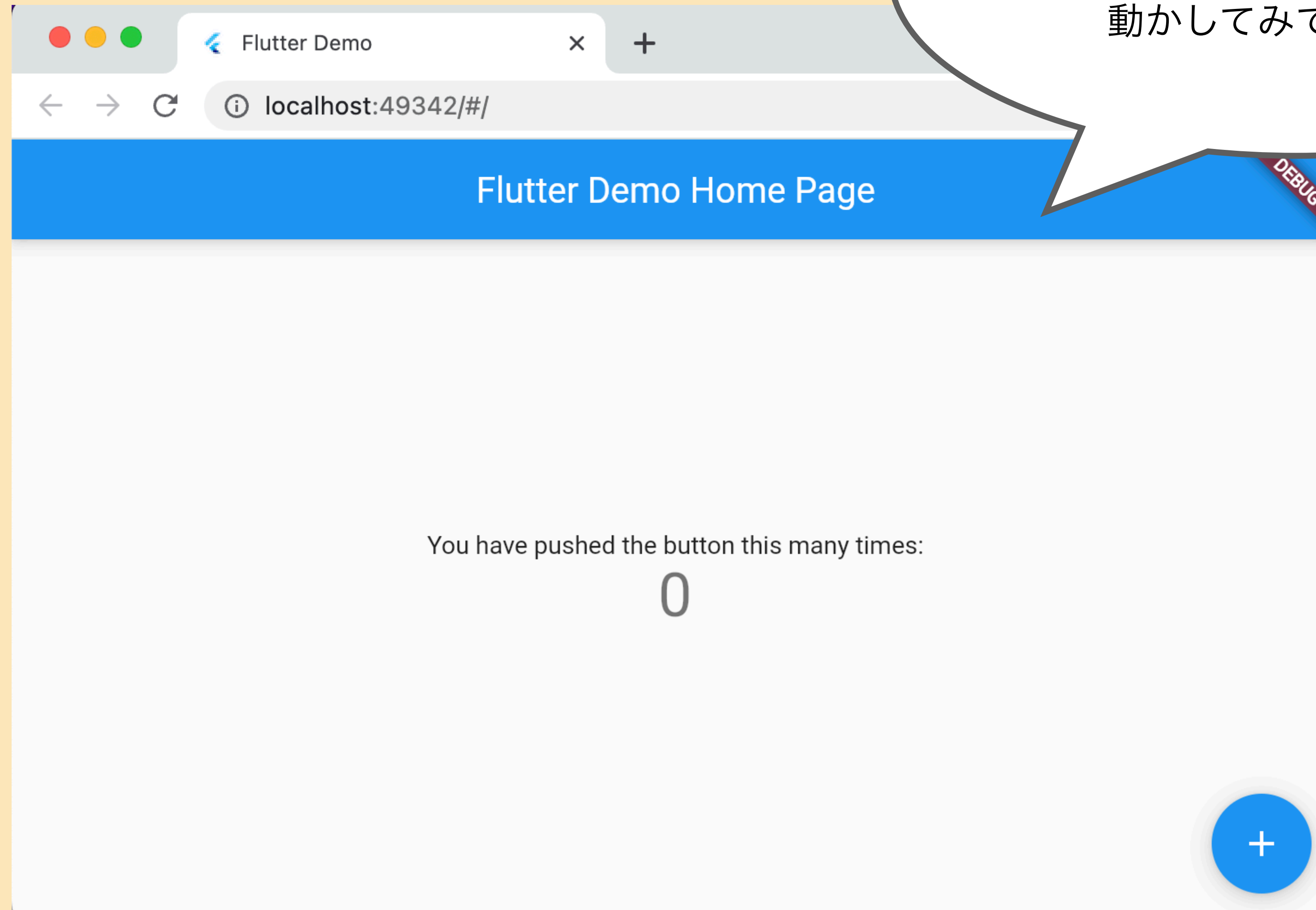
Android Studio を立ち上げて
AVD Manager を選択
もしくは実機接続して
`$ flutter run`



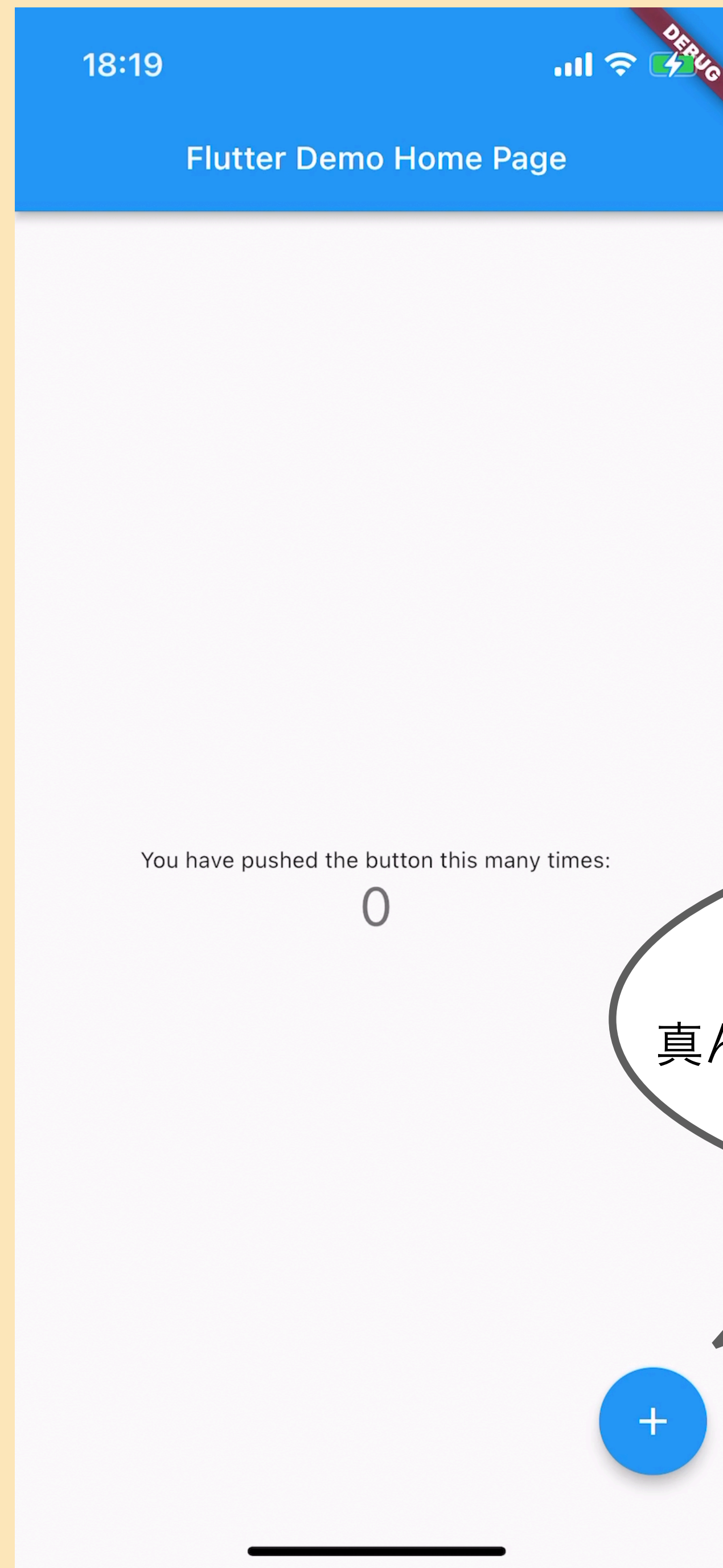
好きなエミュレータを
▶を押して起動

Web でも動く！

Chrome でも動くけど
せっかくならスマホかシミュレーターで
動かしてみしてほしい！



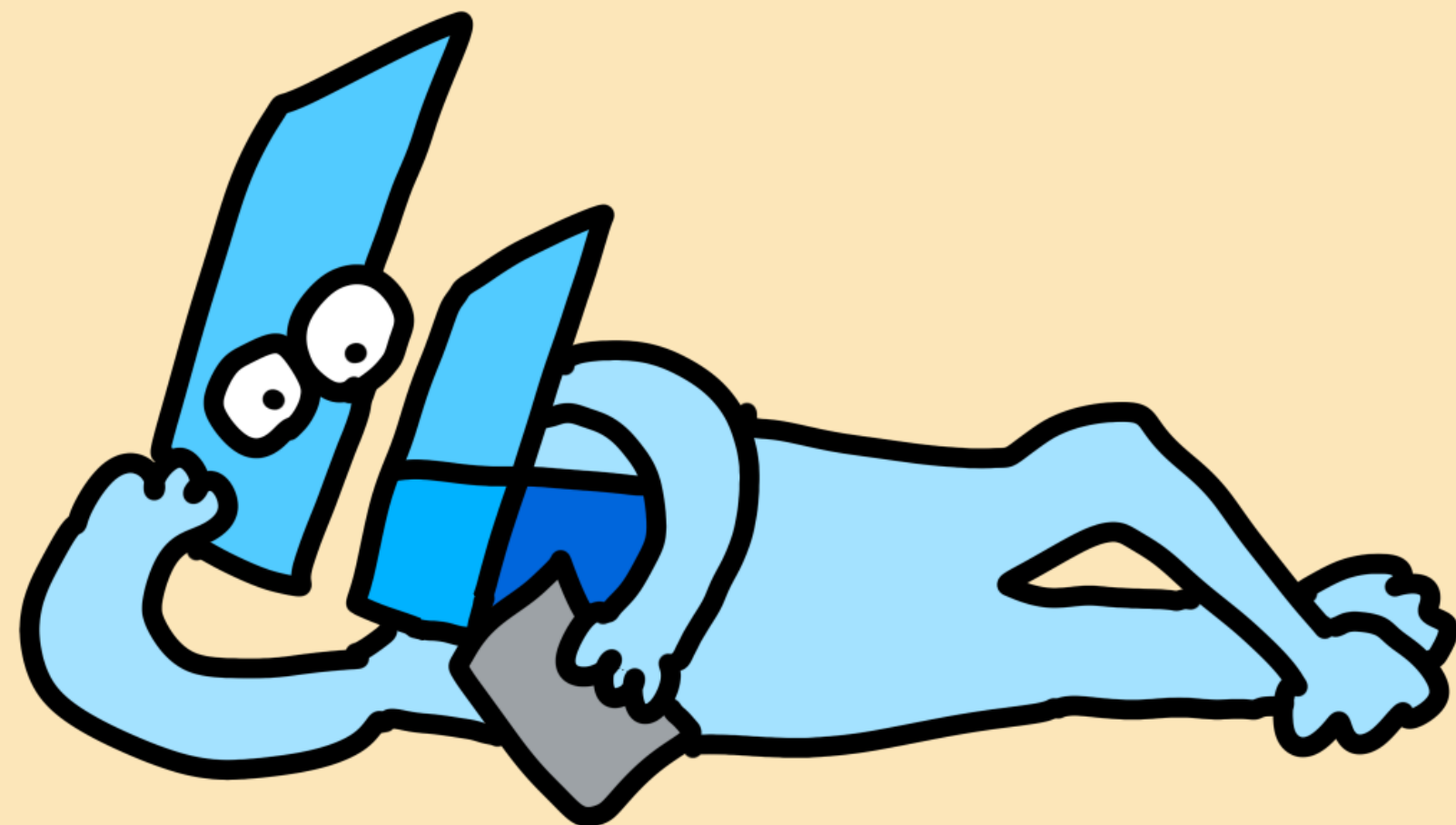
動いたかな???



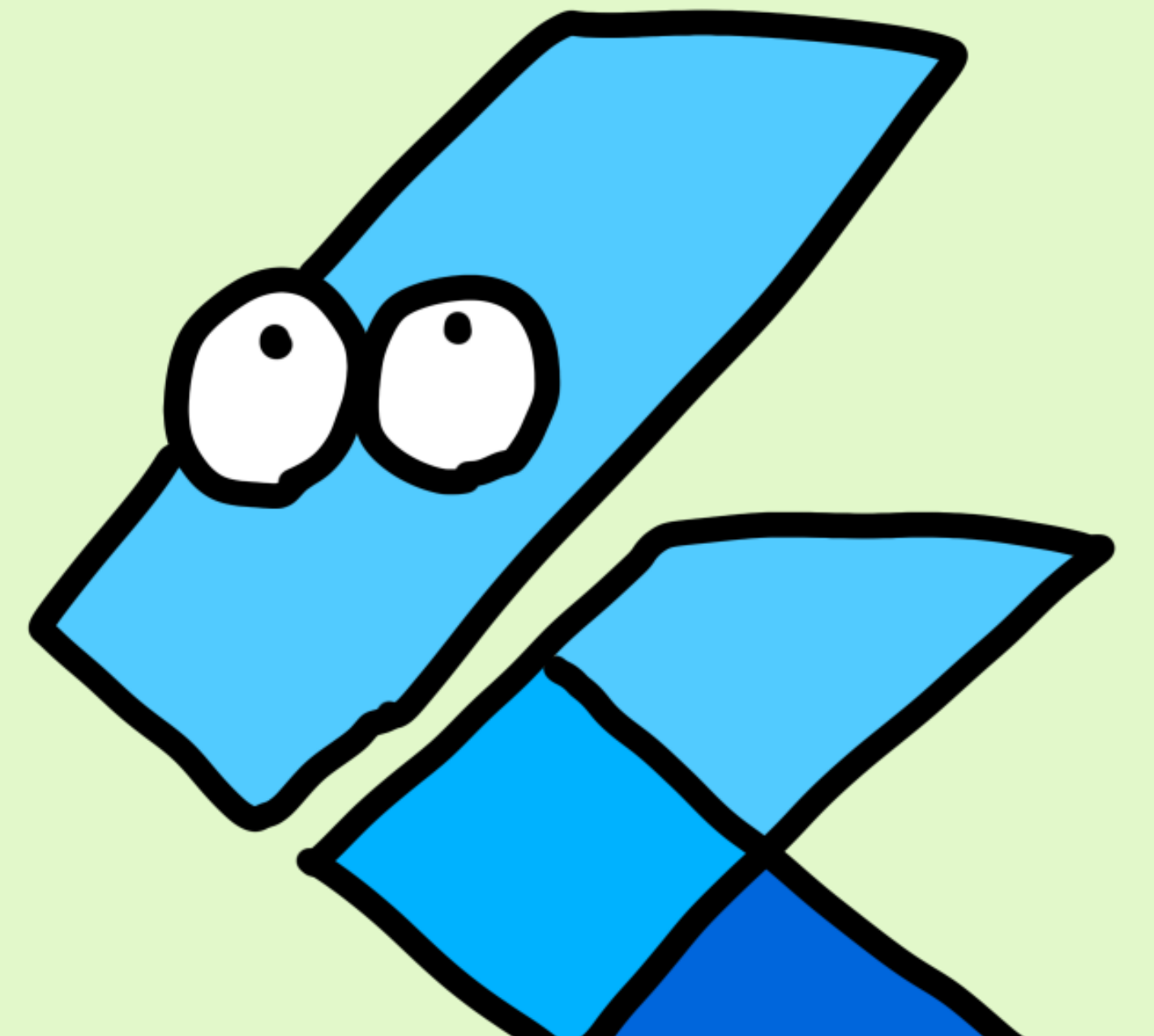
ボタンを押すと
真ん中の数字がカウントアップされていくアプリ

10分間休憩

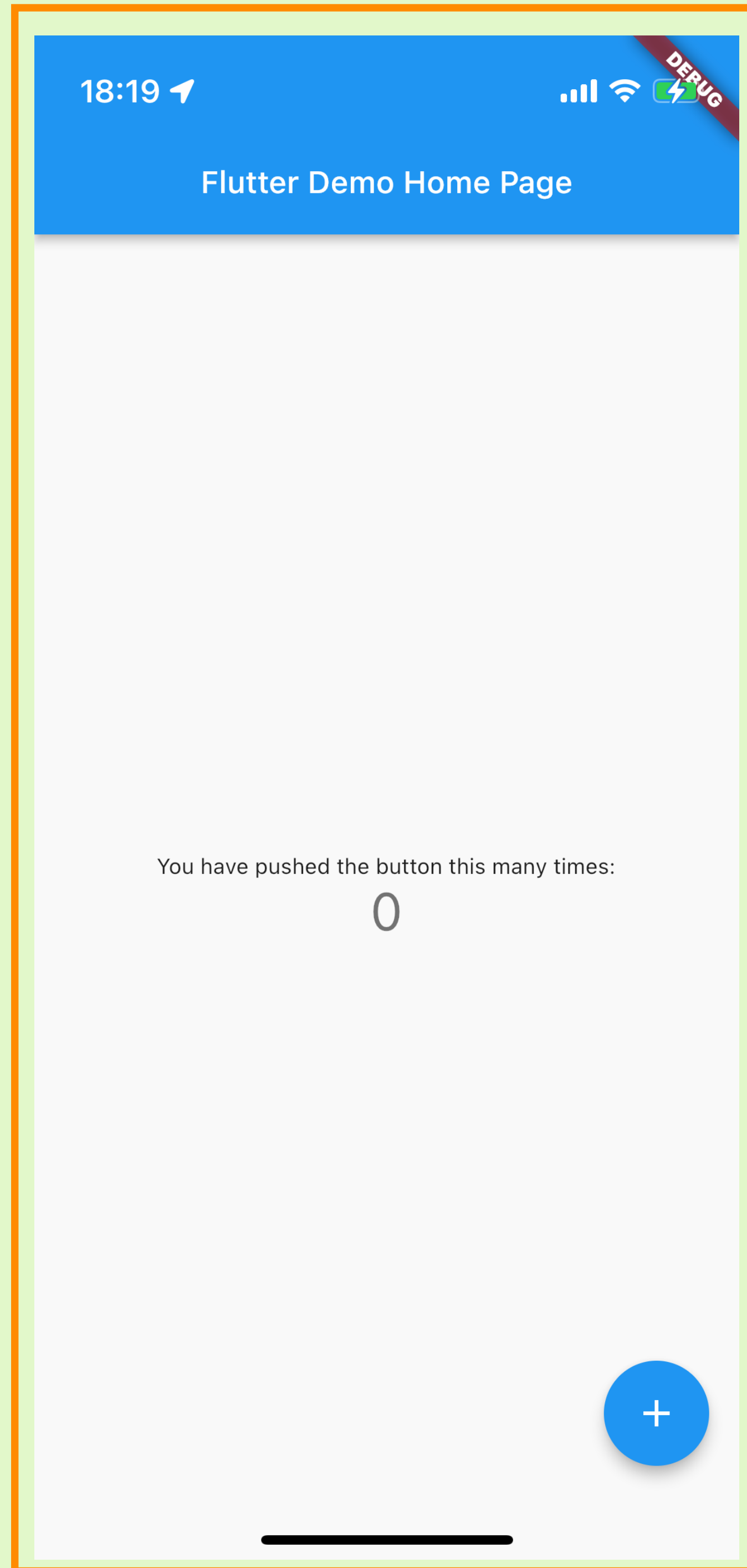
もっと休憩したかったら言ってネ



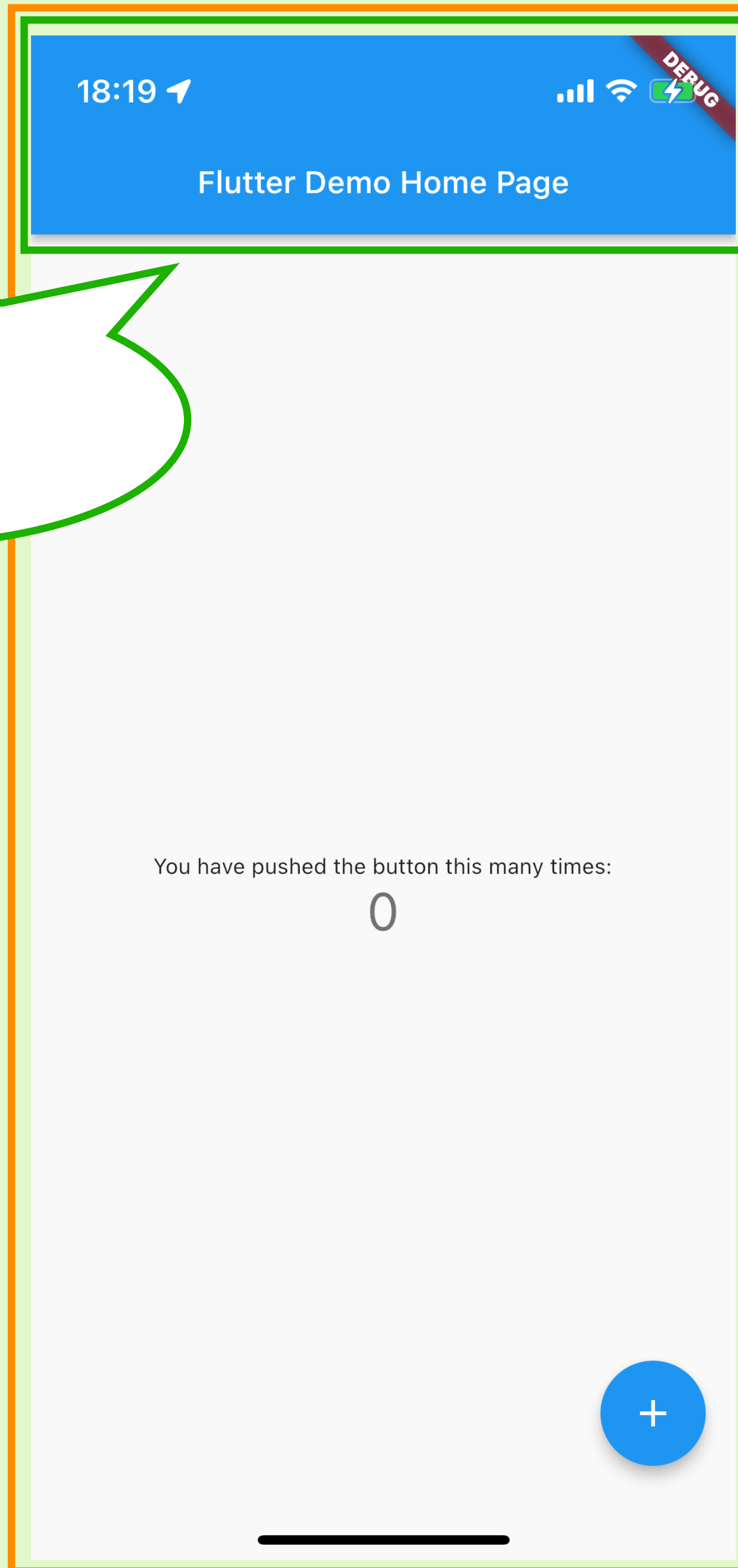
create されたアプリを
見ていこう！



ウィジェット



ウィジェット



AppBar

ページ全体

ウィジェット



AppBar

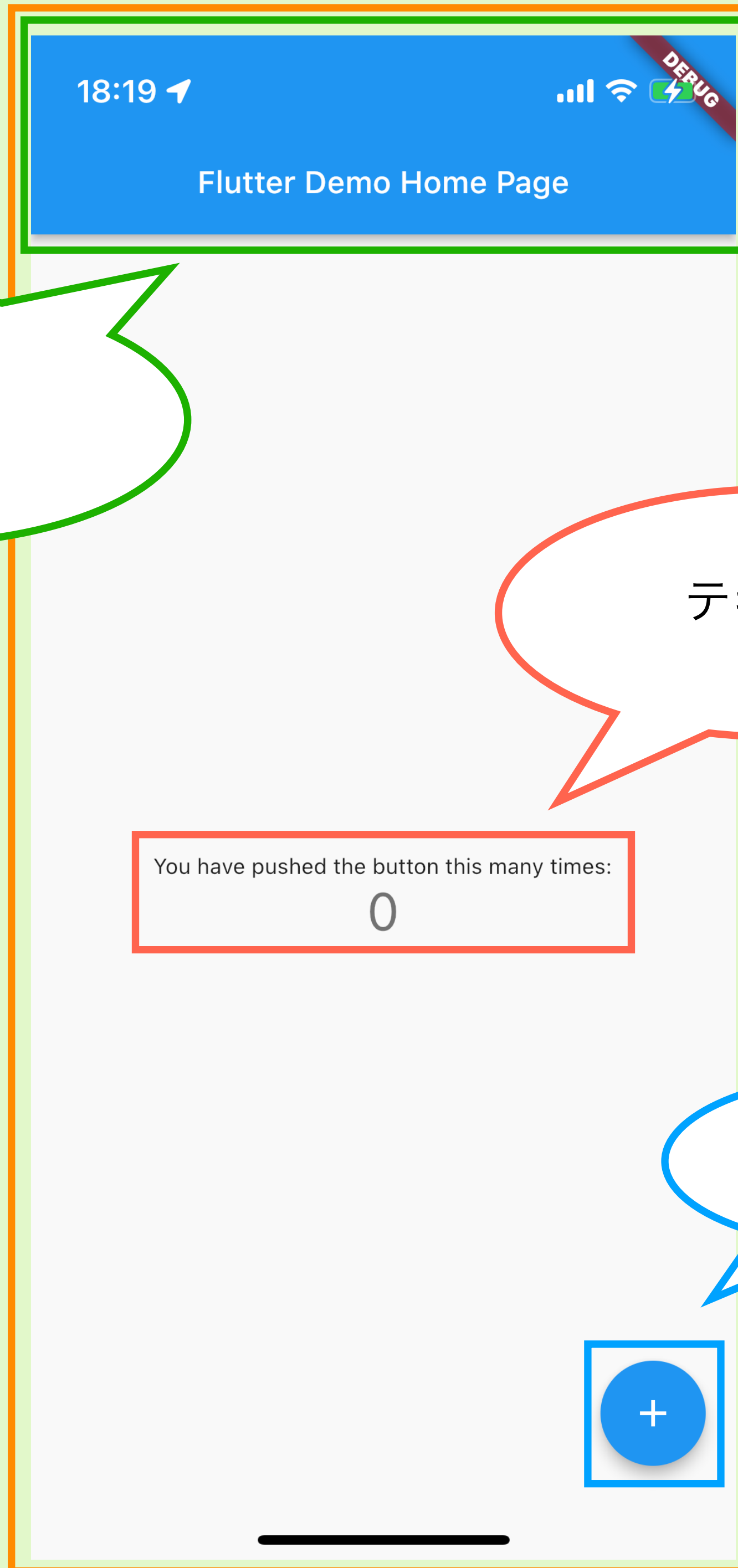
テキスト

ページ全体

You have pushed the button this many times:
0



ウィジェット



AppBar

ページ全体

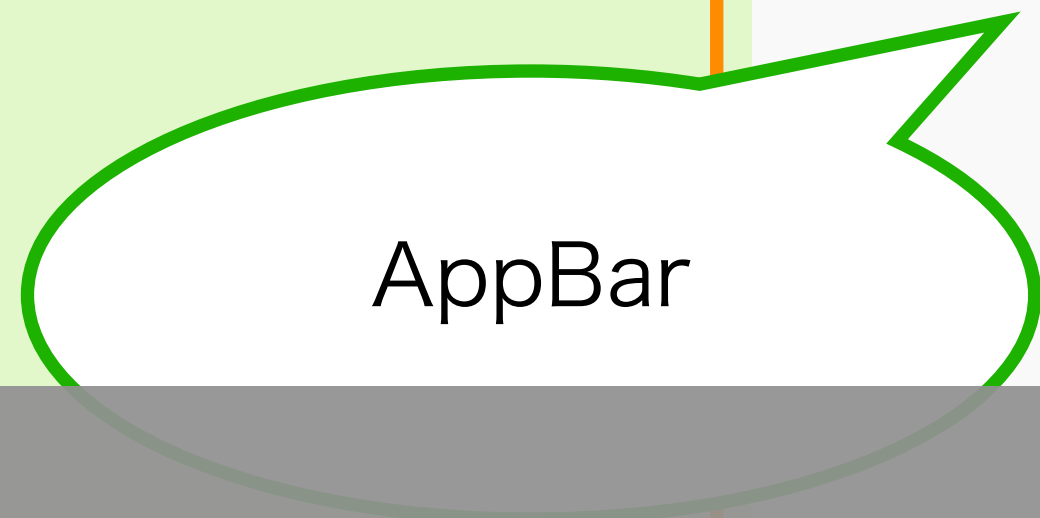
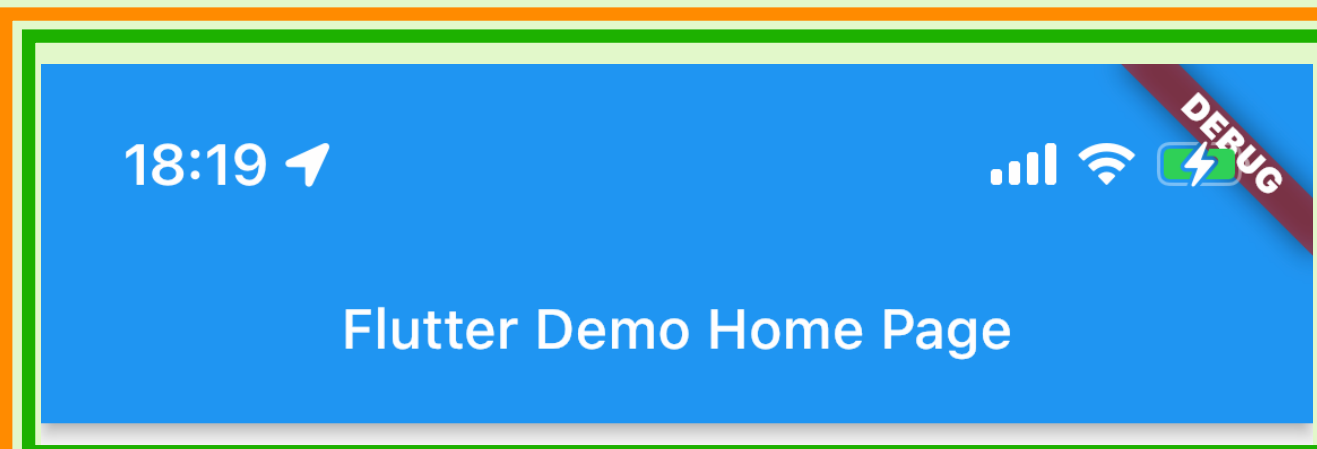
テキスト

You have pushed the button this many times:
0

ボタン

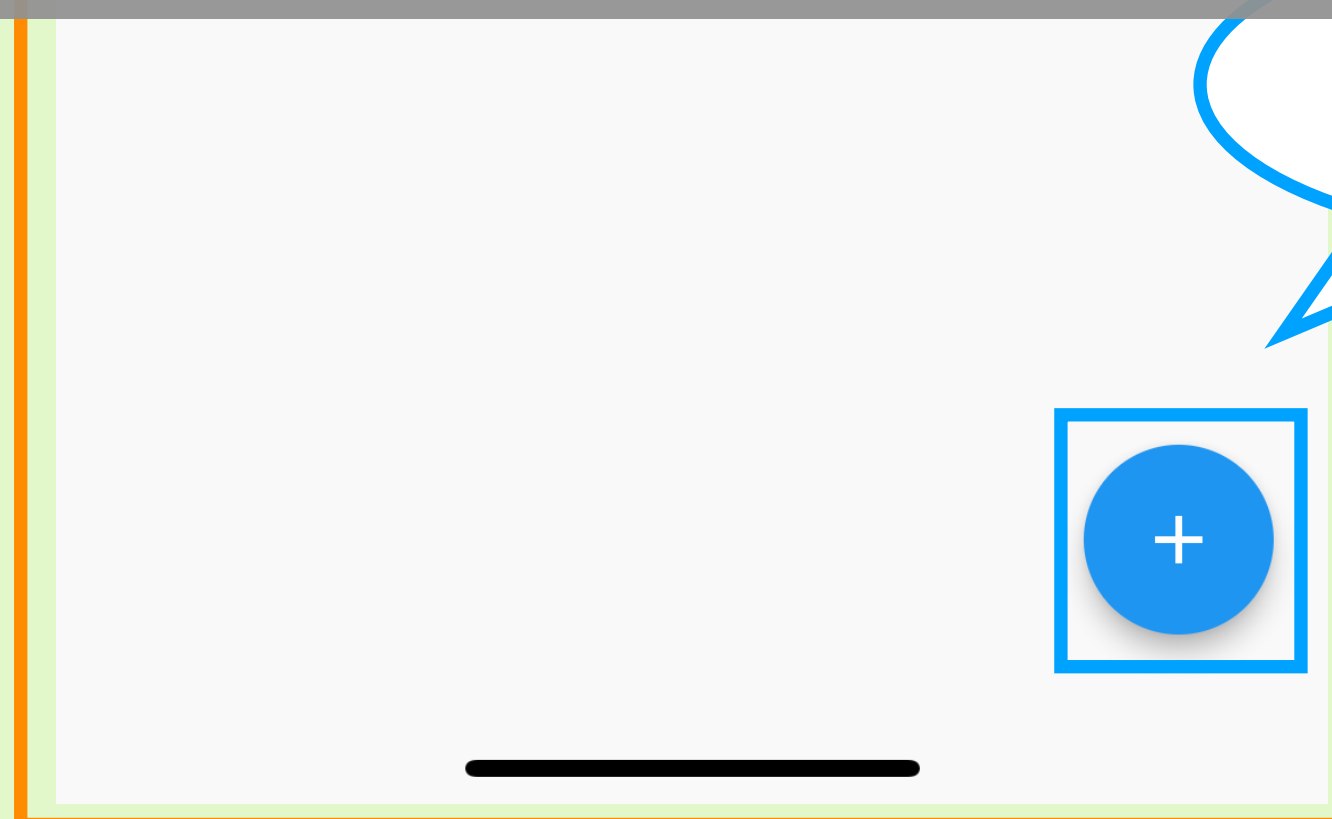
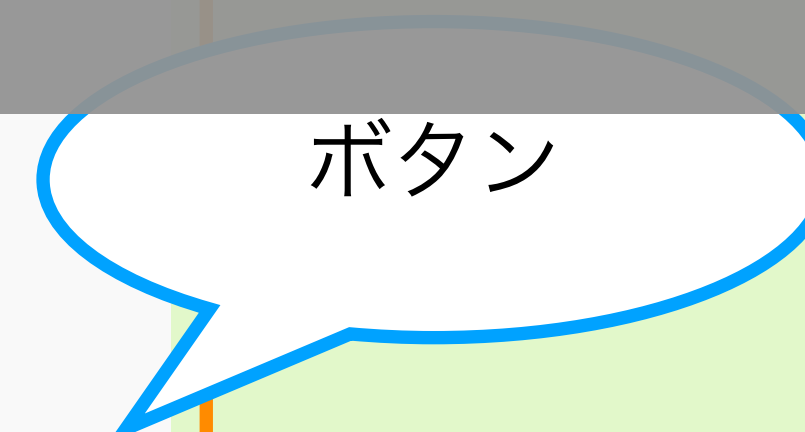


ウィジェット



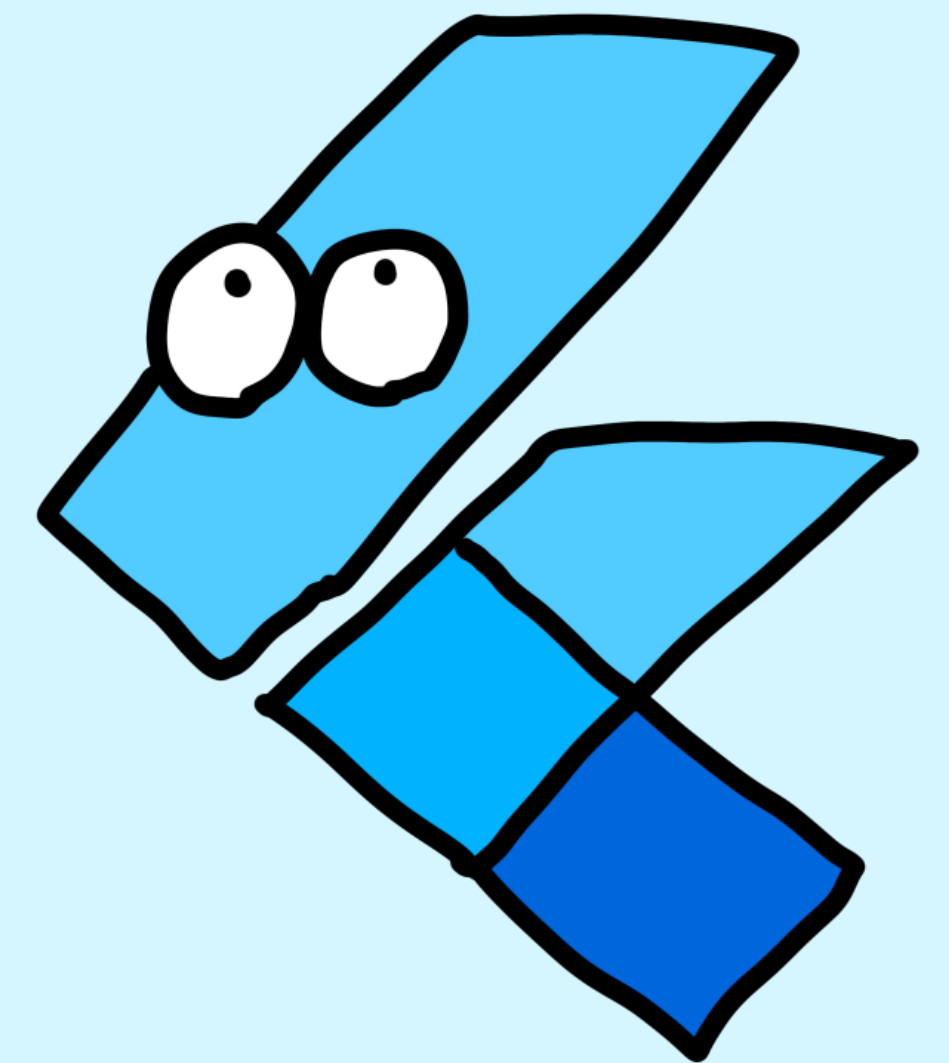
全部ウィジェット！

画面はウィジェットで成り立っている！



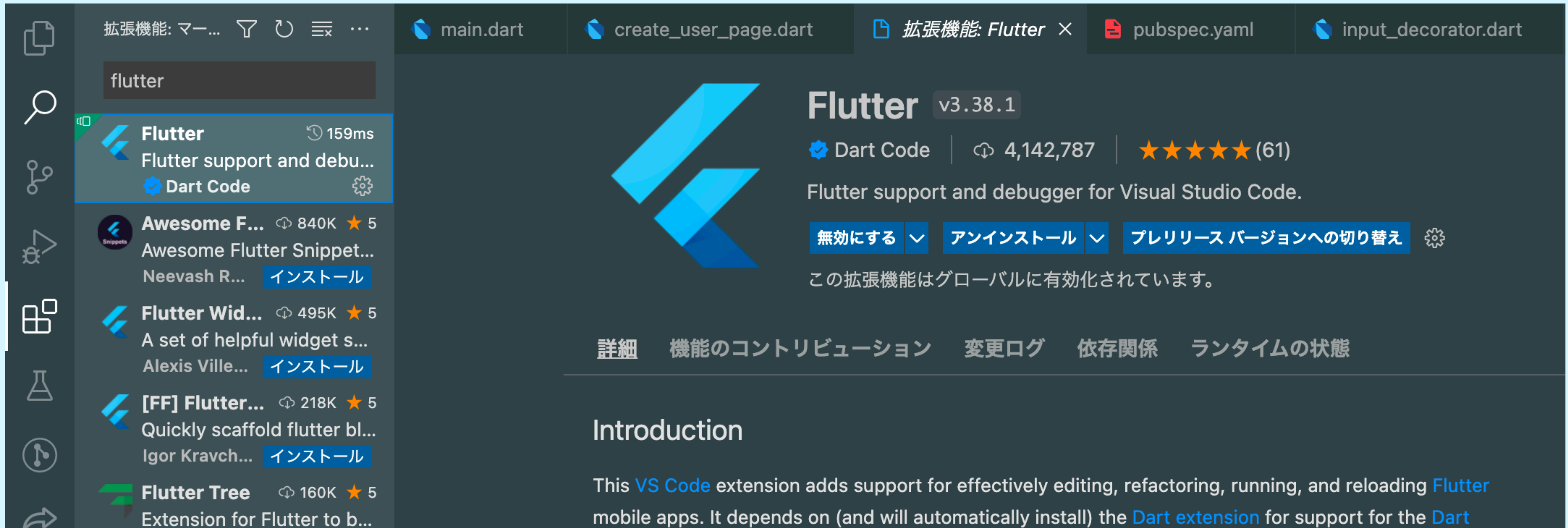
実際のコードを見ていこう！

- ライブコーディングしていくので一緒にアプリを動かしていこう！
 - たまに解説でスライドに戻ってきます
 - Try は一緒に， Challenge は1人で頑張ってみよう！
- リポジトリにコードを置いてあるので聞きそびれたら参考にしてね！
 - <https://github.com/mixigroup/2023BeginnerTrainingFlutter>
 - コミットメッセージやコード内コメントが説明になっています！



VSCode の人～！

拡張機能入れておくとめっちゃくちゃ便利です



The screenshot shows the Visual Studio Code interface with the search bar containing 'flutter'. The search results list several extensions, with the top one being 'Flutter' by Dart Code. The details for the 'Flutter' extension are displayed on the right, showing its version (v3.38.1), download count (4,142,787), and a 5-star rating (61 reviews). The extension is currently installed and enabled globally. Below the details, there are tabs for 'Introduction', '機能のコントリビューション', '変更ログ', '依存関係', and 'ランタイムの状態'. The 'Introduction' tab is active, showing a brief description of the extension's purpose.

拡張機能: マー... | main.dart | create_user_page.dart | 拡張機能: Flutter × | pubspec.yaml | input_decorator.dart

flutter

Flutter v3.38.1
Flutter support and debu...
Dart Code

Awesome F... 840K ★ 5
Awesome Flutter Snippet...
Neevash R... インストール

Flutter Wid... 495K ★ 5
A set of helpful widget s...
Alexis Ville... インストール

[FF] Flutter... 218K ★ 5
Quickly scaffold flutter bl...
Igor Kravch... インストール

Flutter Tree 160K ★ 5
Extension for Flutter to b...

Flutter v3.38.1
Dart Code | 4,142,787 | ★★★★★ (61)
Flutter support and debugger for Visual Studio Code.

無効にする | アンインストール | プレリリースバージョンへの切り替え

この拡張機能はグローバルに有効化されています。

詳細 | 機能のコントリビューション | 変更ログ | 依存関係 | ランタイムの状態


Introduction

This VS Code extension adds support for effectively editing, refactoring, running, and reloading Flutter mobile apps. It depends on (and will automatically install) the Dart extension for support for the Dart

『flutter』で検索して一番上に出るやつ

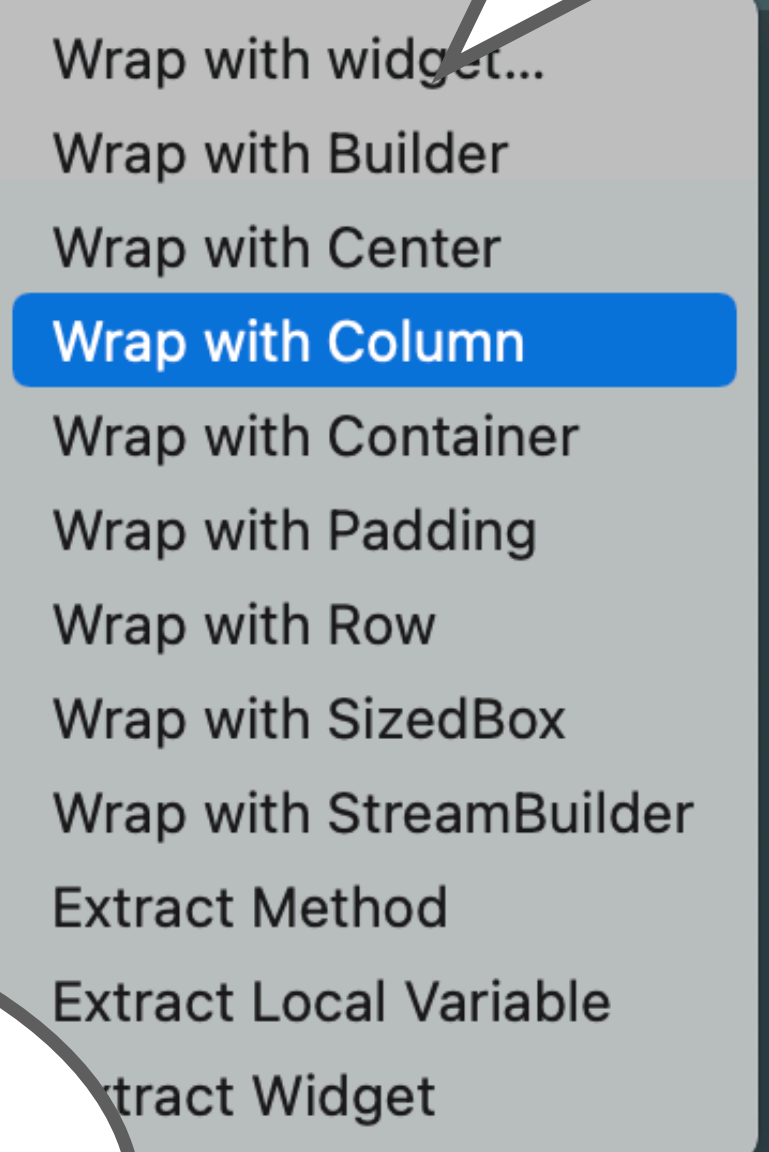
拡張機能はこう使う！

```
title: const Text("ようこそ"), // AppBar
), // AppBar
body: const Center(
  child: Text("よろろひ~"), // Center
); // Scaffold
```



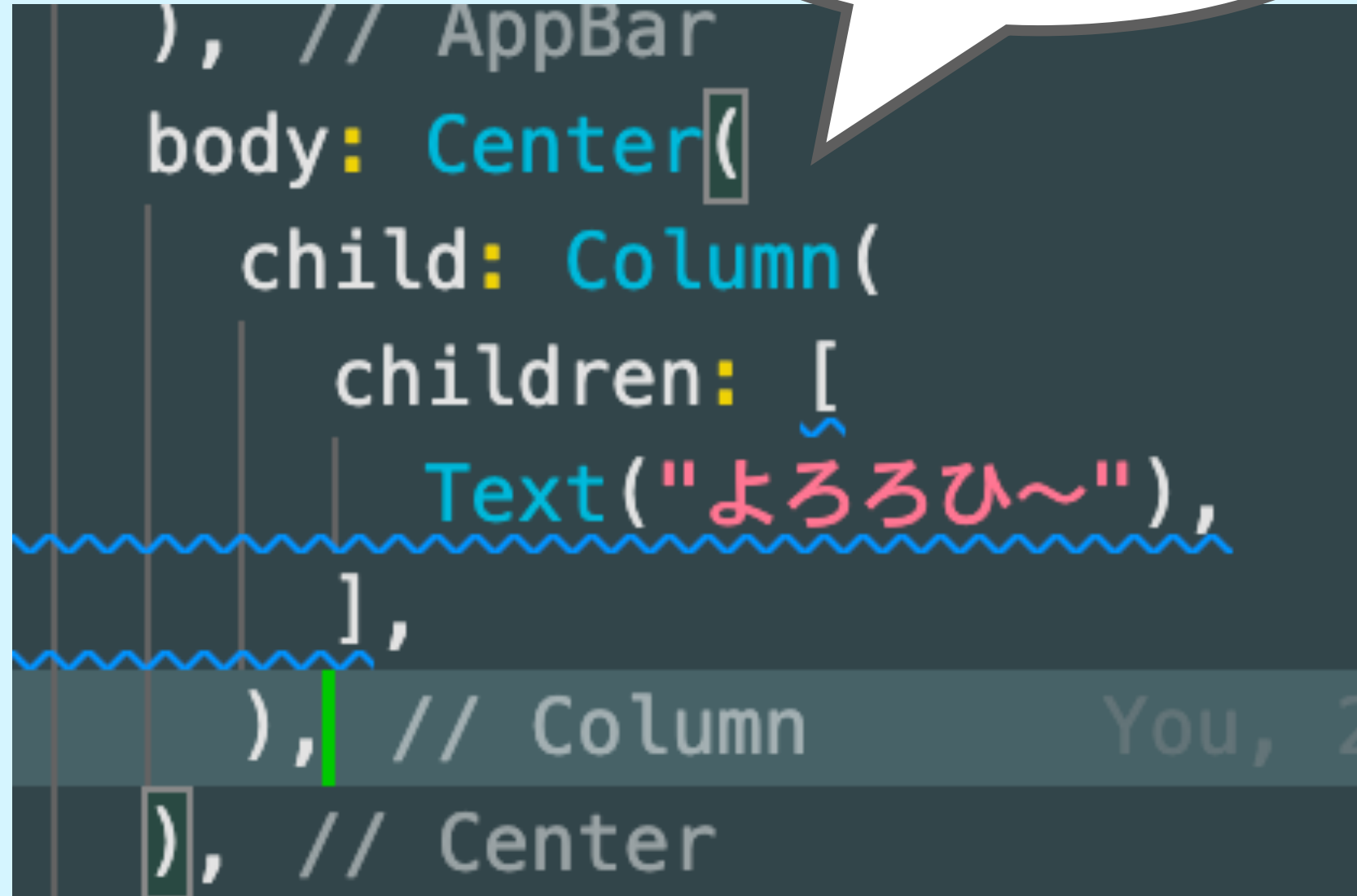
何かで囲いたい
ウィジェットを右クリック
リファクタを選択

```
), // AppBar
body: const Center(
  child: Text("よろろひ~"), // Center
); // Scaffold
```



囲いたいウィジェットを選択

```
), // AppBar
body: Center(
  child: Column(
    children: [
      Text("よろろひ~"),
    ],
  ), // Column
), // Center
```



デデン！ 囲われた！

#1 Introduction

- ライブコーディングします！
- プロジェクト（ディレクトリ）構成について
- main() の流れや class の説明
- MyHomePage のファイル分けてみる
 - 他のファイルを参照するときは import する
- Text の中身を変えてみたりする
 - **HotReload/Restart** とは？（別ページで説明）
- MyHomePage見てく
 - Scaffold とか column など**標準ウィジェット**がたくさん！（別ページで説明）
 - 拡張機能はこう使う！（別ページで説明）

Hot reload/restart とは？

いちいち ctrl-c からの flutter run しなくて大丈夫！！！！

Hot reload

ターミナルで r を押す
Text や色などウィジェットを
変更したら反映される

State はそのまま
(State に関しては後で説明)

Hot restart

ターミナルで R を押す
ほぼリセットされる

flutter run

ネイティブコードから
実行し直す

※ Hot reload ができるのはデバッグビルド時 (JITコンパイル時) のみ

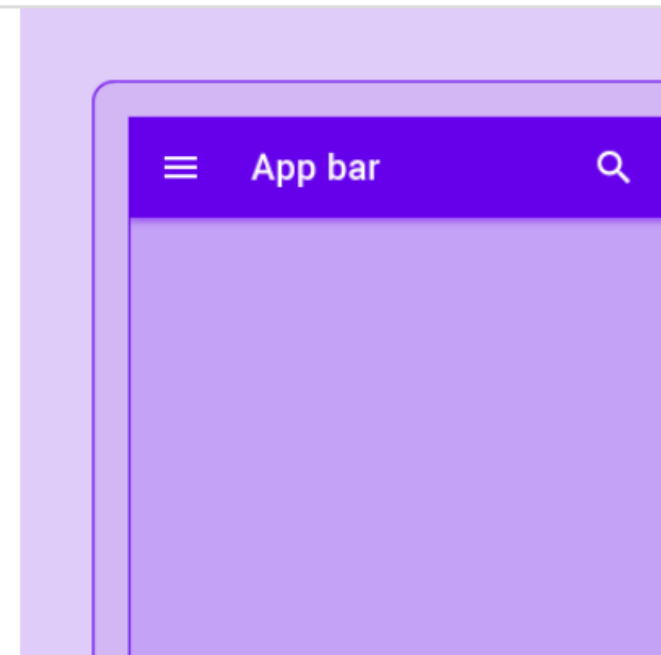
標準ウィジェット

Basic widgets

UI > Widgets > Basics

Widgets you absolutely need to know before building your first Flutter app.

See more widgets in the [widget catalog](#).



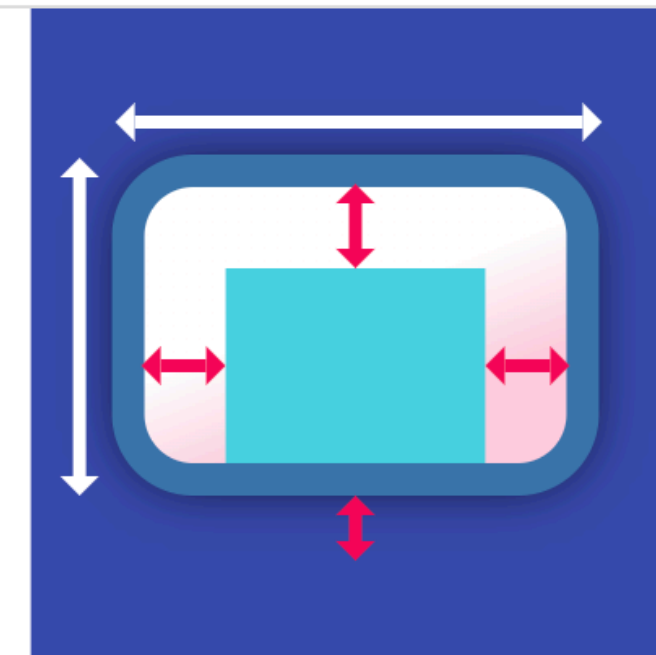
AppBar

A Material Design app bar. An app bar consists of a toolbar and potentially other widgets, such as a TabBar and a FlexibleSpaceBar.



Column

Layout a list of child widgets in the vertical direction.



Container

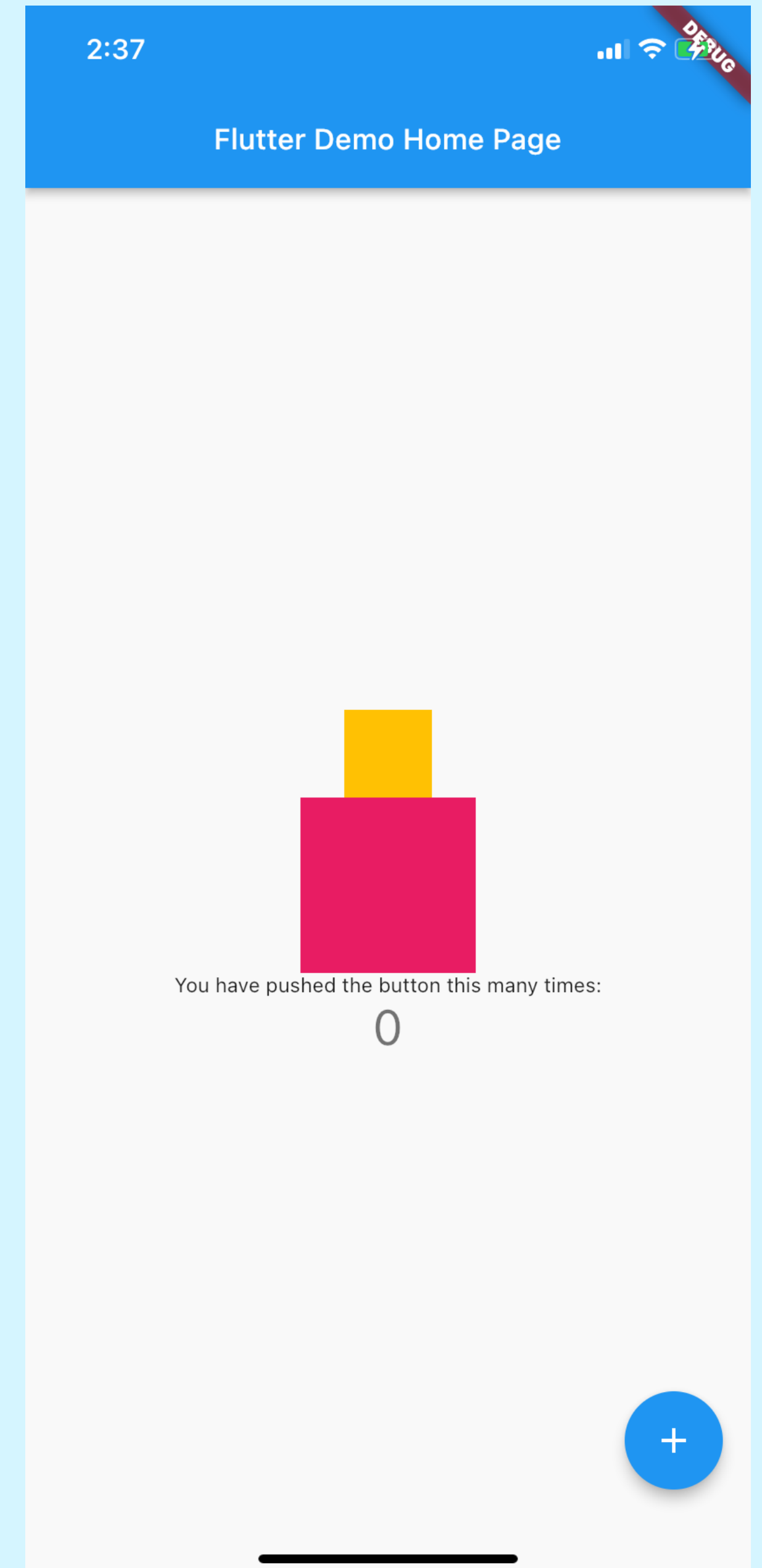
A convenience widget that combines common painting, positioning, and sizing widgets.

よく使うウィジェットが用意されてる！

#1 Try! 標準ウィジェットを使ってみよう

<https://github.com/mixigroup/2023BeginnerTrainingFlutter/pull/1>

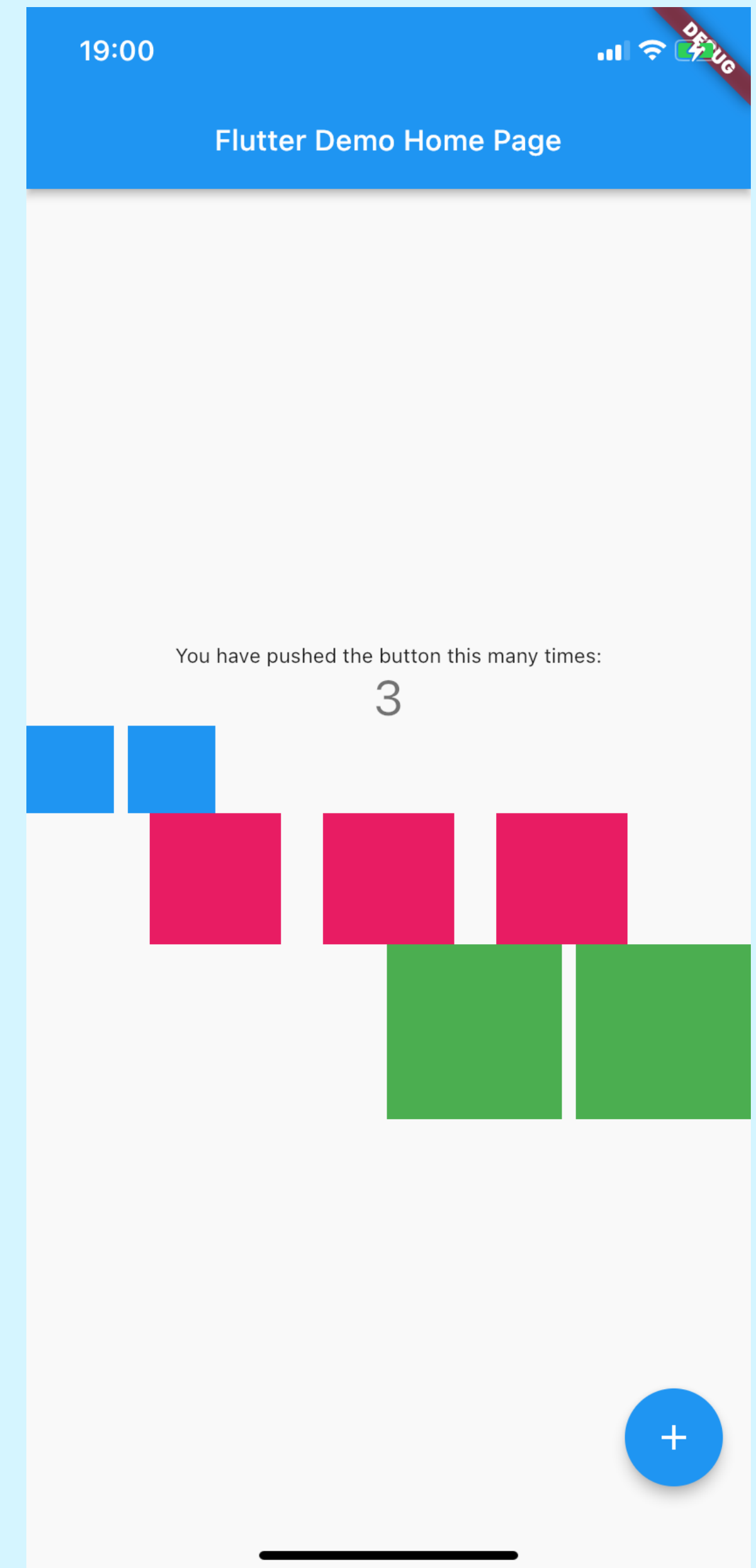
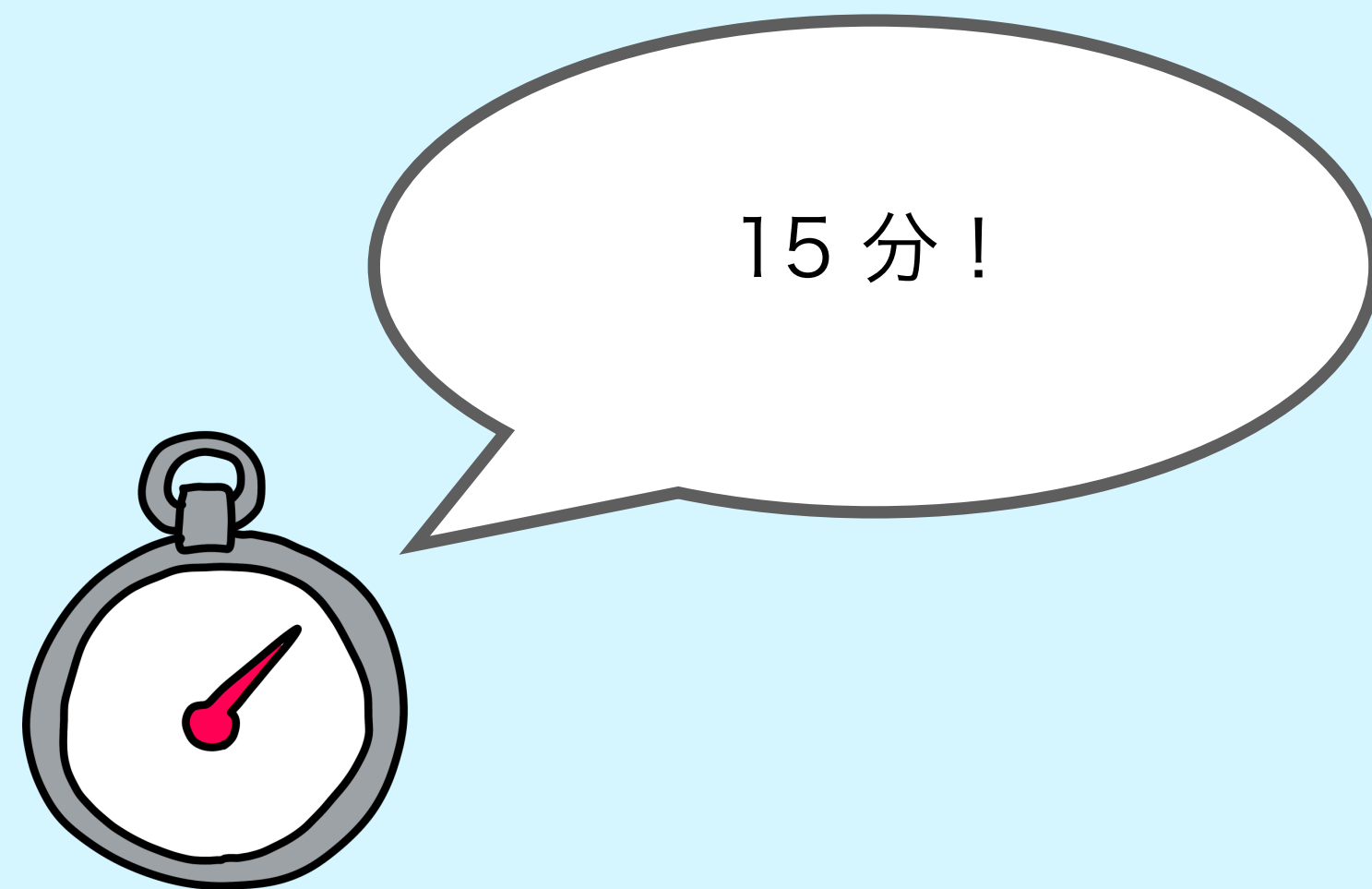
- ライブコーディングします！
- 使用想定ウィジェット
 - Text
 - Center
 - Column
 - Row
 - SizedBox
 - ColoredBox



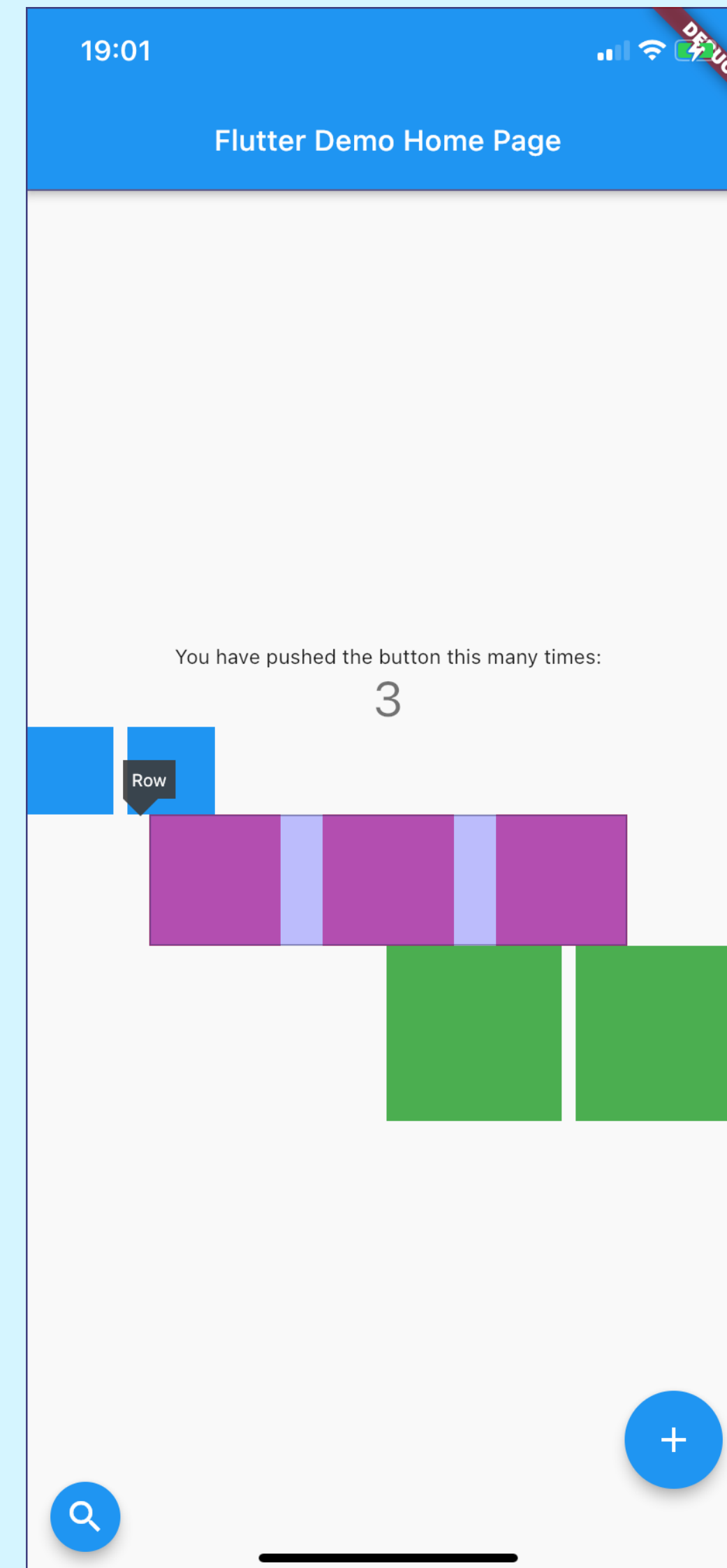
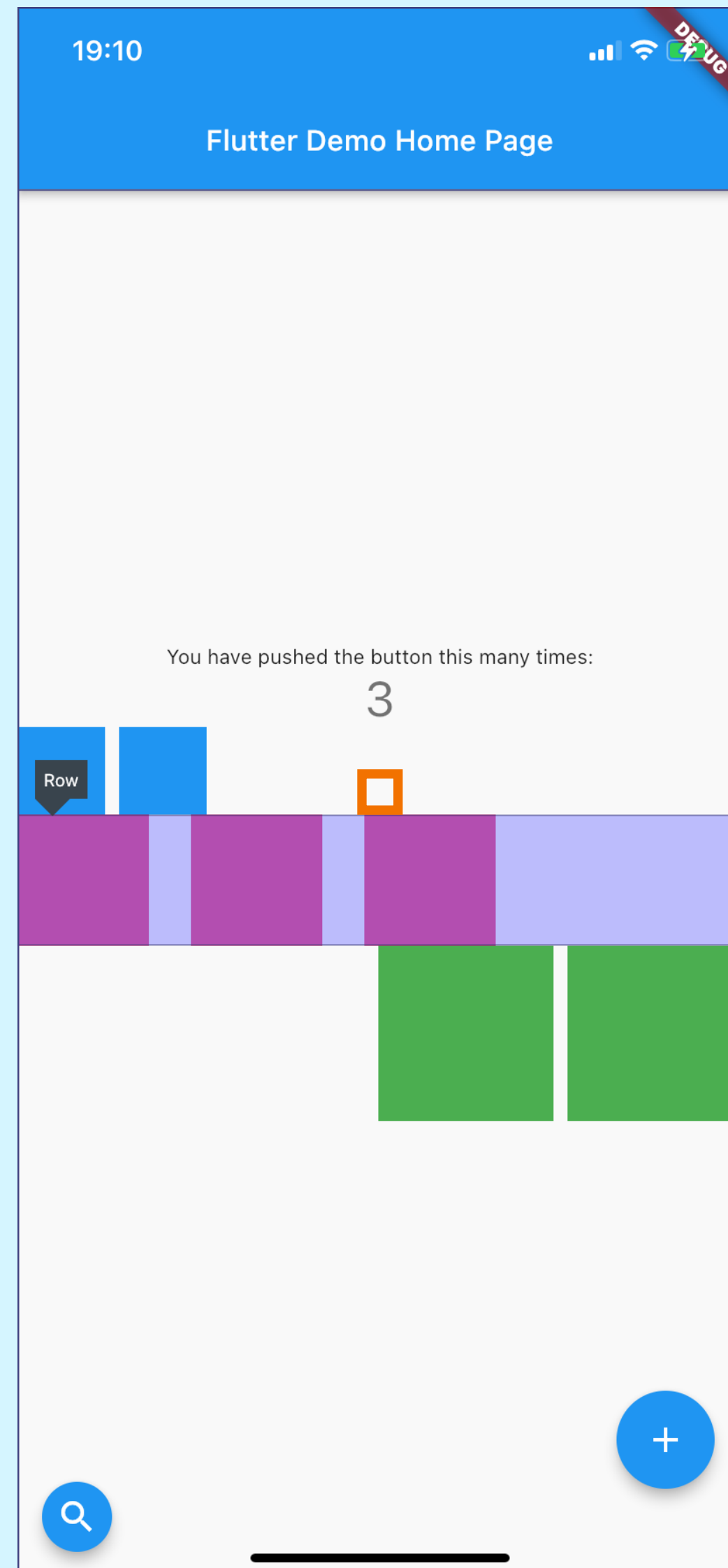
#1 Challenge! 図形を描画してみよう

<https://github.com/mixigroup/2023BeginnerTrainingFlutter/pull/18>

- Row と Column を使って右の図形を描画してみよう！
 - 大きさや色, 隙間はテキストで大丈夫です
- 質問大歓迎！！！！
 - 隙間の付け方や並べ方など

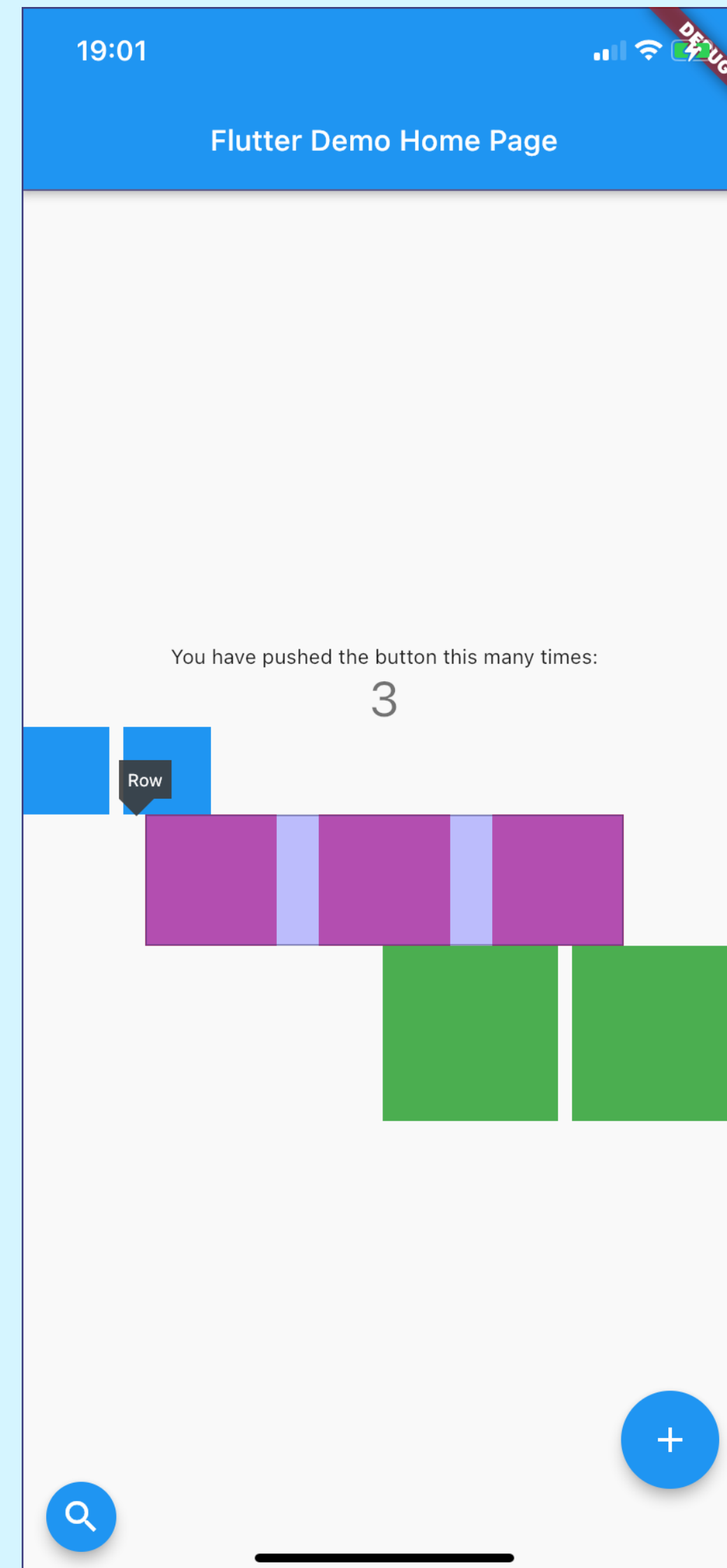
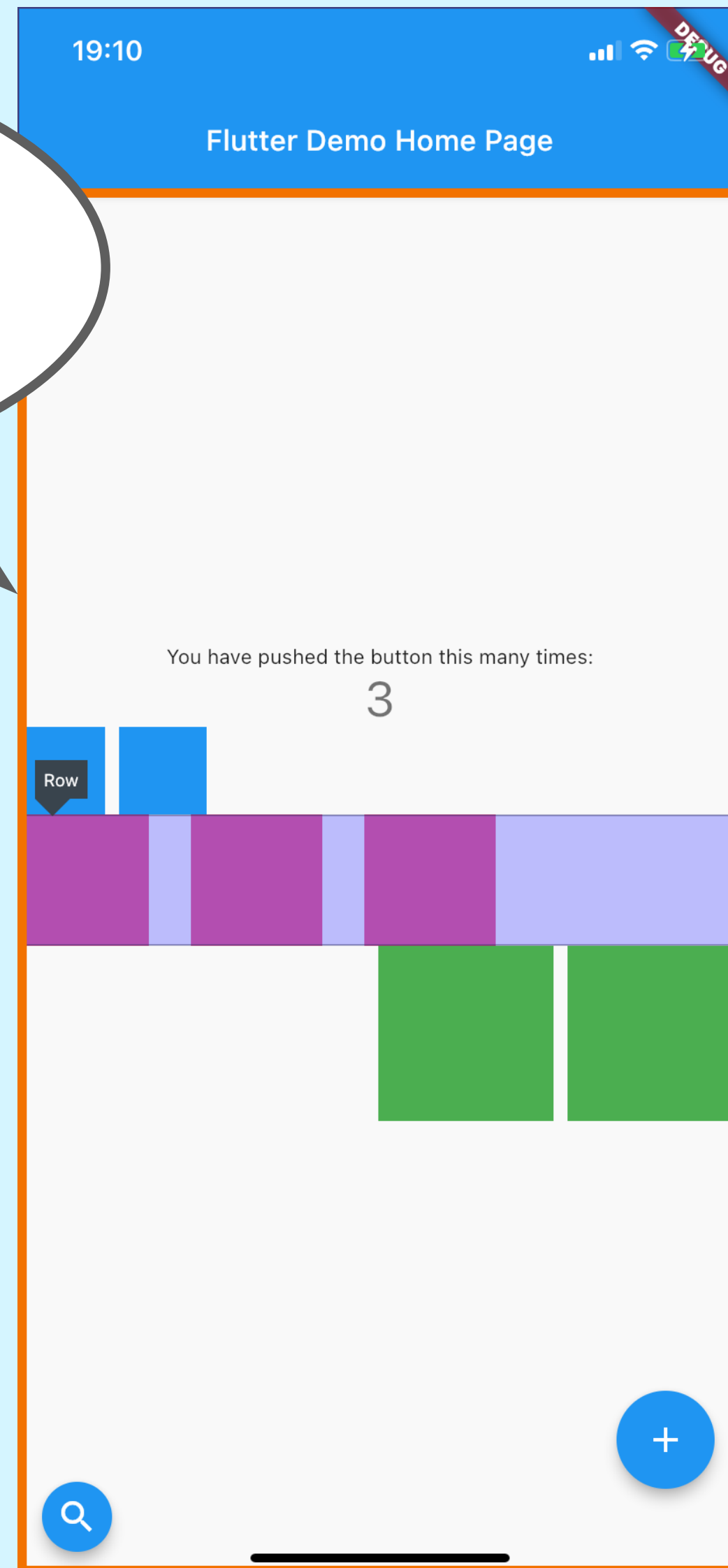


Column や Row は親のサイズまで広がる性質を持つ



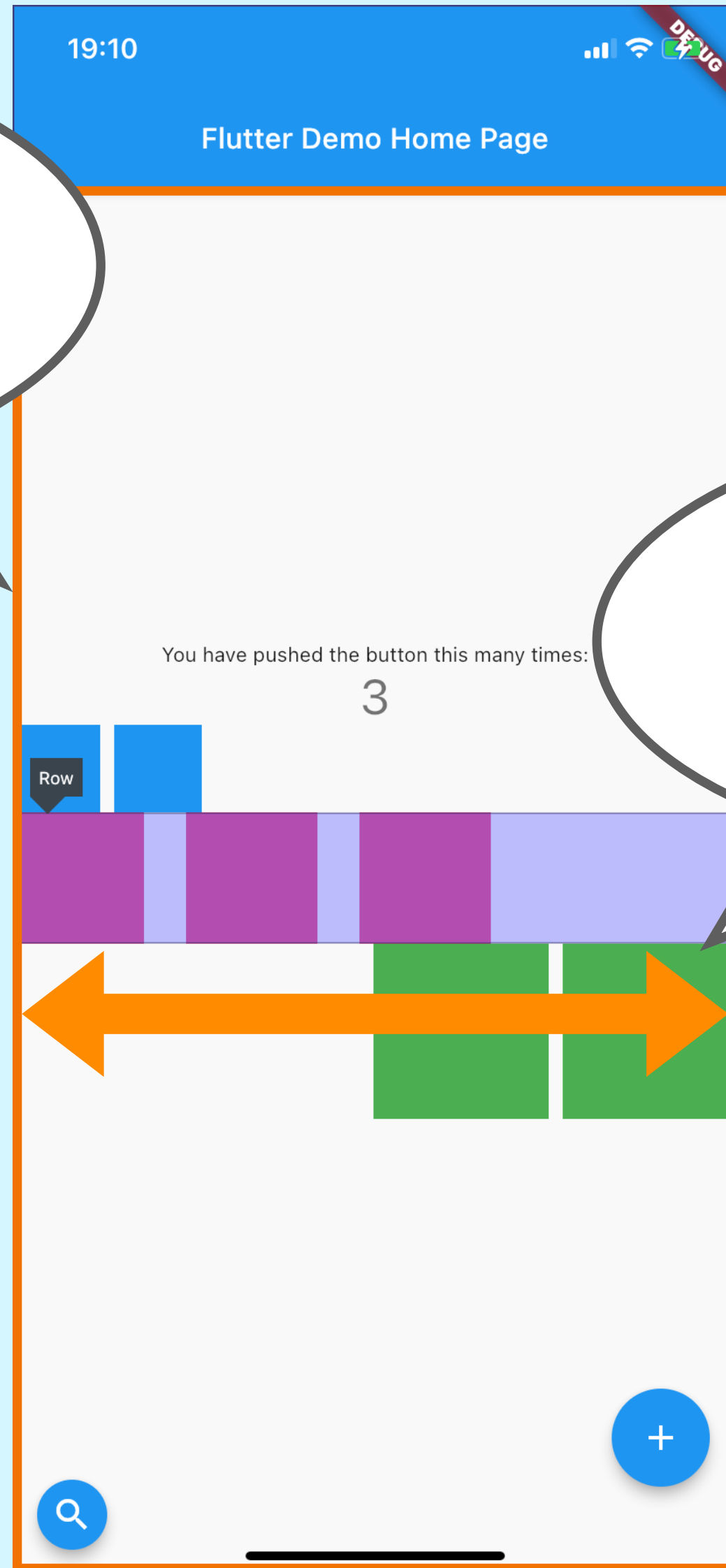
Column や Row は親のサイズまで広がる性質を持つ

Column のサイズは
オレンジの枠



Column や Row は親のサイズまで広がる性質を持つ

Column のサイズは
オレンジの枠

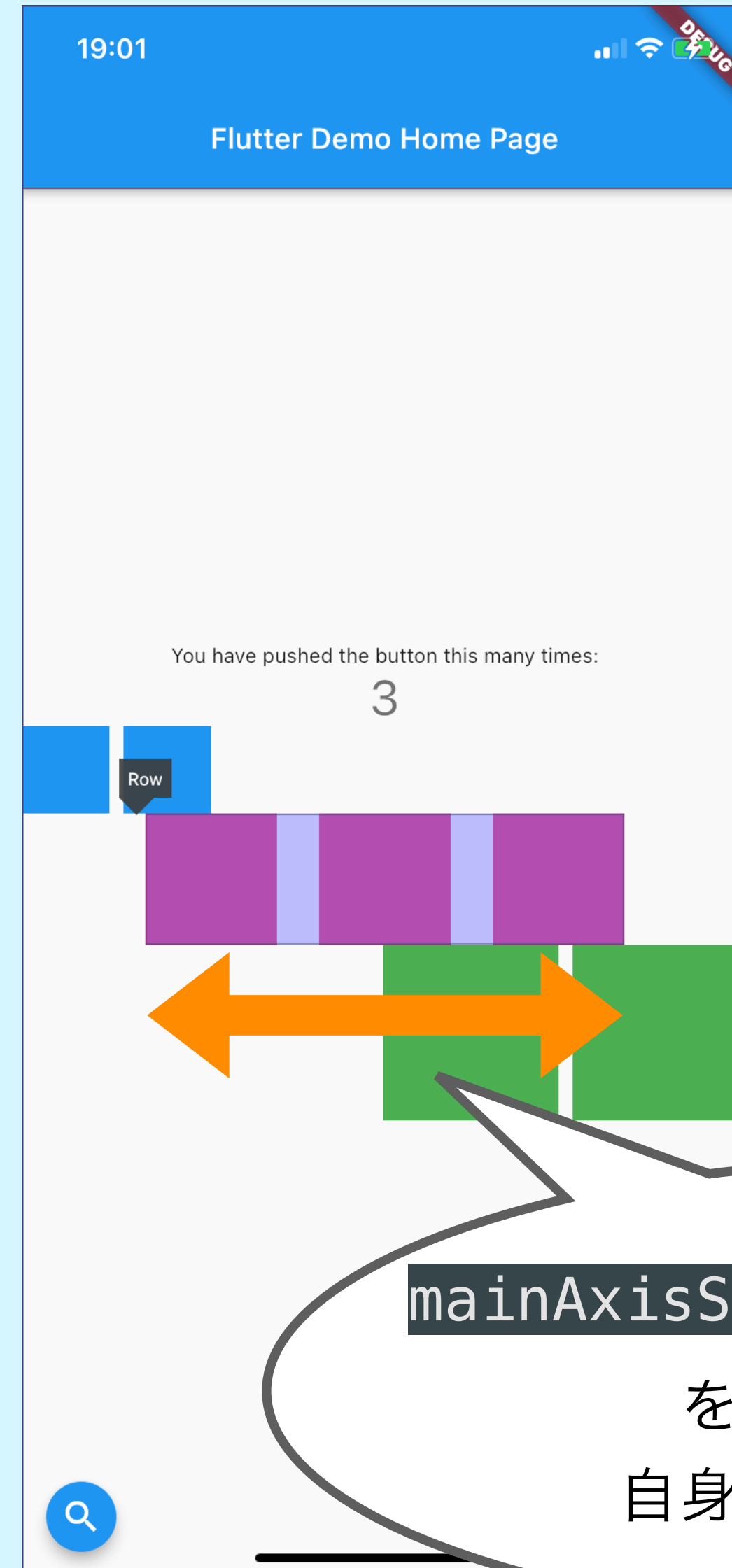
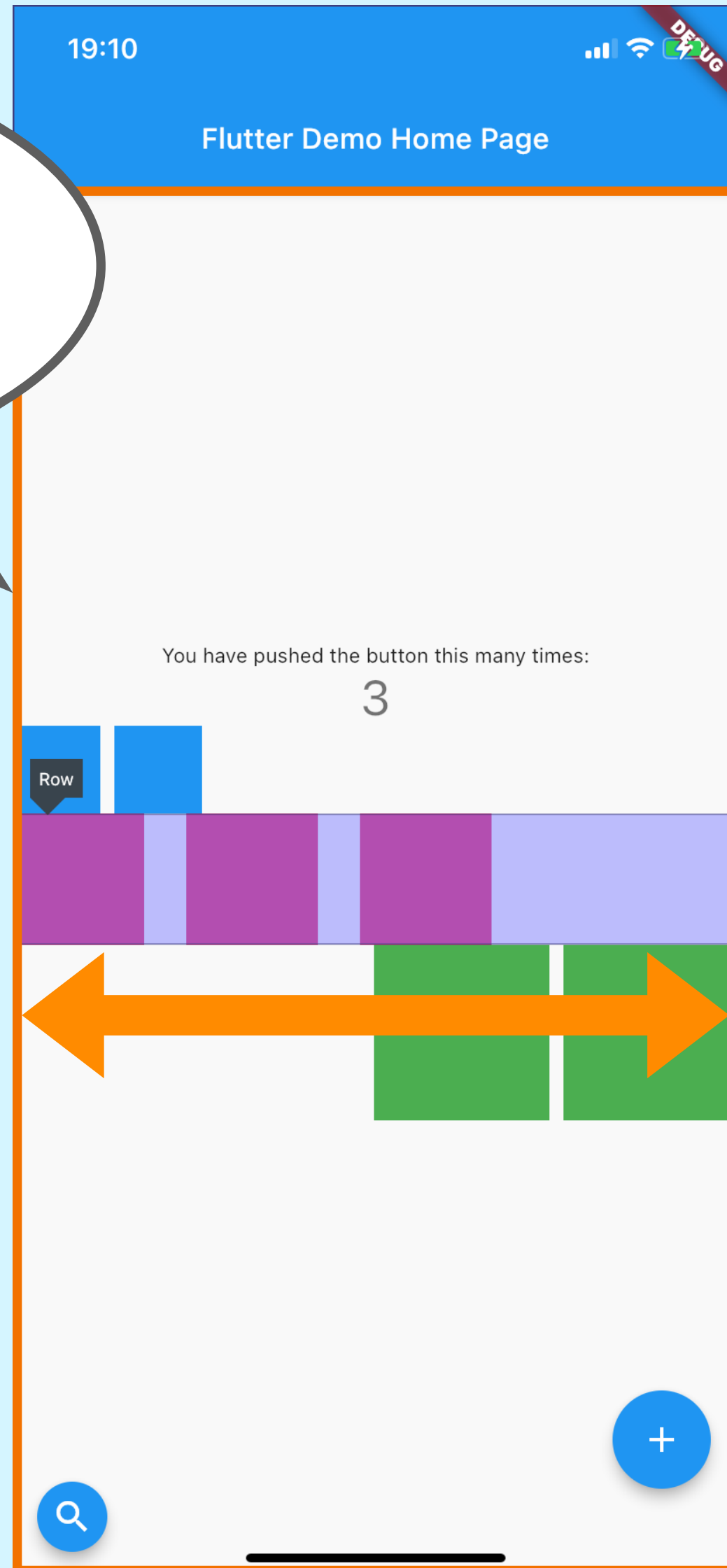


Row もめいっぱい
広がるとする



Column や Row は親のサイズまで広がる性質を持つ

Column のサイズは
オレンジの枠



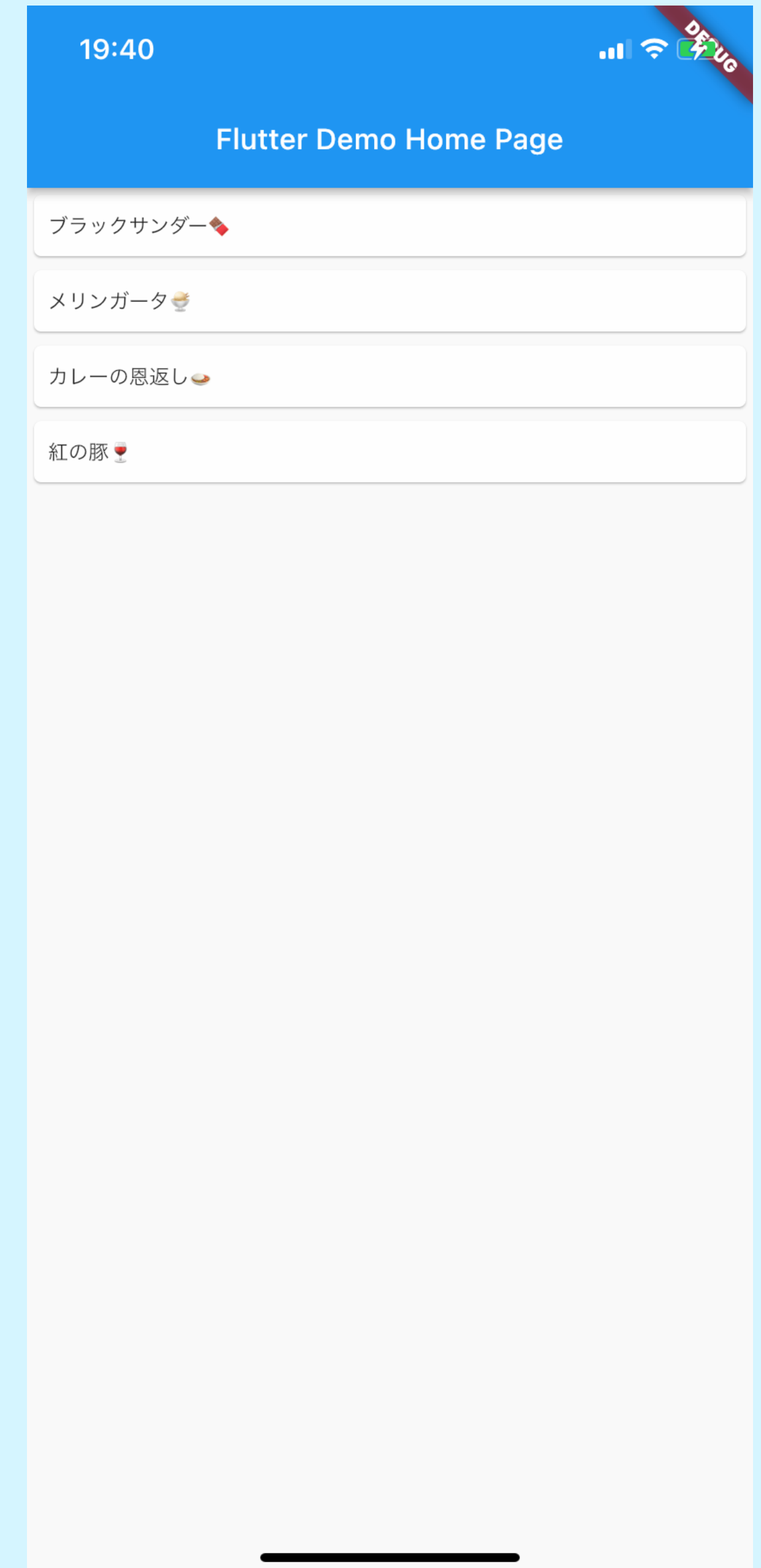
```
mainAxisSize: MainAxisSize.min,
```

を指定してあげると
自身の大きさに留まる！

#1 Try! リストを表示してみよう

<https://github.com/mixigroup/2023BeginnerTrainingFlutter/pull/2>

- ライブコーディングします！
- 使用想定ウィジェット
 - ListView
 - Card
 - Padding



#1 Challenge! リストにタイトルをつけよう

<https://github.com/mixigroup/2023BeginnerTrainingFlutter/pull/19>

- まずは思いつく方法でやってみて！
 - エラーになる人がいたらヒント出します！



Column の中に ListView を入れるときは注意！

- ListView も広がるうとしてくる性質を持つのでエラーになります

```
The following RenderObject was being processed when the exception was fired:  
RenderViewport#339c5 NEEDS-LAYOUT NEEDS-PAINT NEEDS-COMPOSITING-BITS-UPDATE:  
  needs compositing  
  creator: Viewport ← IgnorePointer-[GlobalKey#2a4db] ← Semantics ← Listener ←  
  _GestureSemantics ←  
    RawGestureDetector-[LabeledGlobalKey<RawGestureDetectorState>#c4042] ← Listener ←  
  _ScrollableScope  
    ← _ScrollSemantics-[GlobalKey#830a2] ← NotificationListener<ScrollMetricsNotification> ←  
    Scrollable ← PrimaryScrollController ← ...  
  parentData: <none> (can use size)  
  constraints: BoxConstraints(0.0<=w<=414.0, 0.0<=h<=Infinity)  
  . . .
```

Column の中に ListView を入れるときは注意！

- ListView も広がるうとしてくる性質を持つのでエラーになります

The following RenderObject was being processed when the exception was fired:
RenderViewport#339c5 NEEDS-LAYOUT NEEDS-PAINT NEEDS-COMPOSITING-BITS-UPDATE:
needs compositing
creator: Viewport ← IgnorePointer-[GlobalKey#2a4db] ← Semantics ← Listener ←
_GestureSemantics ←
RawGestureDetector-[LabeledGlobalKey<RawGestureDetectorSt ←
_ScrollableScope
← _ScrollSemantics-[GlobalKey#830a2] ← NotificationListener ←
Scrollable ← PrimaryScrollController ← ...
parentData: <none> (can use size)
constraints: BoxConstraints(0.0<=w<=414.0, **0.0<=h<=Infinity**)
...



Infinity !!

Column の中に ListView を入れるときは注意！

- ListView も広がるうとしてくる性質を持つのでエラーになります

Column



Column の中に ListView を入れるときは注意！

- ListView も広がるうとしてくる性質を持つのでエラーになります

Column



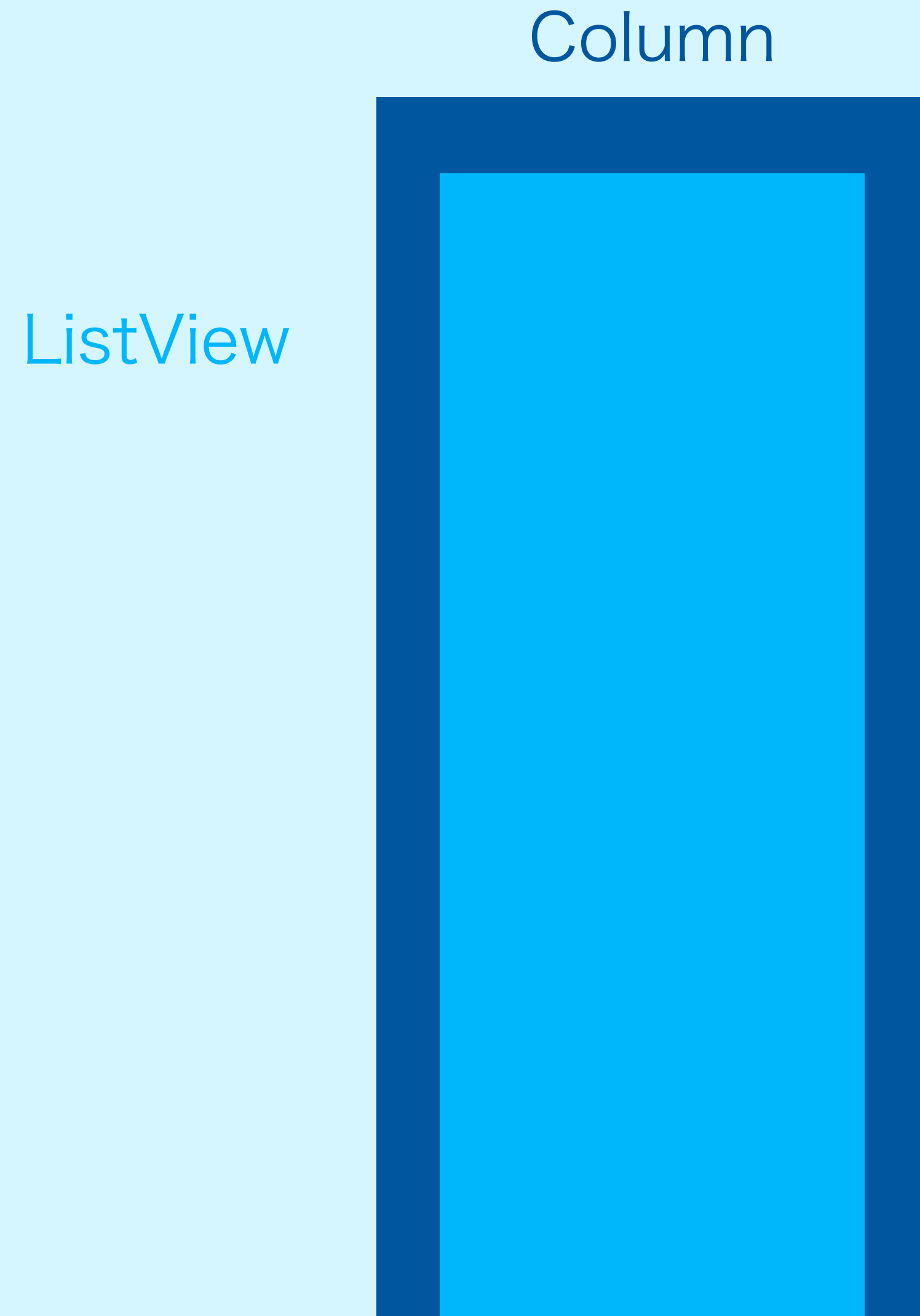
Column の中に ListView を入れるときは注意！

- ListView も広がるようになってくる性質を持つのでエラーになります



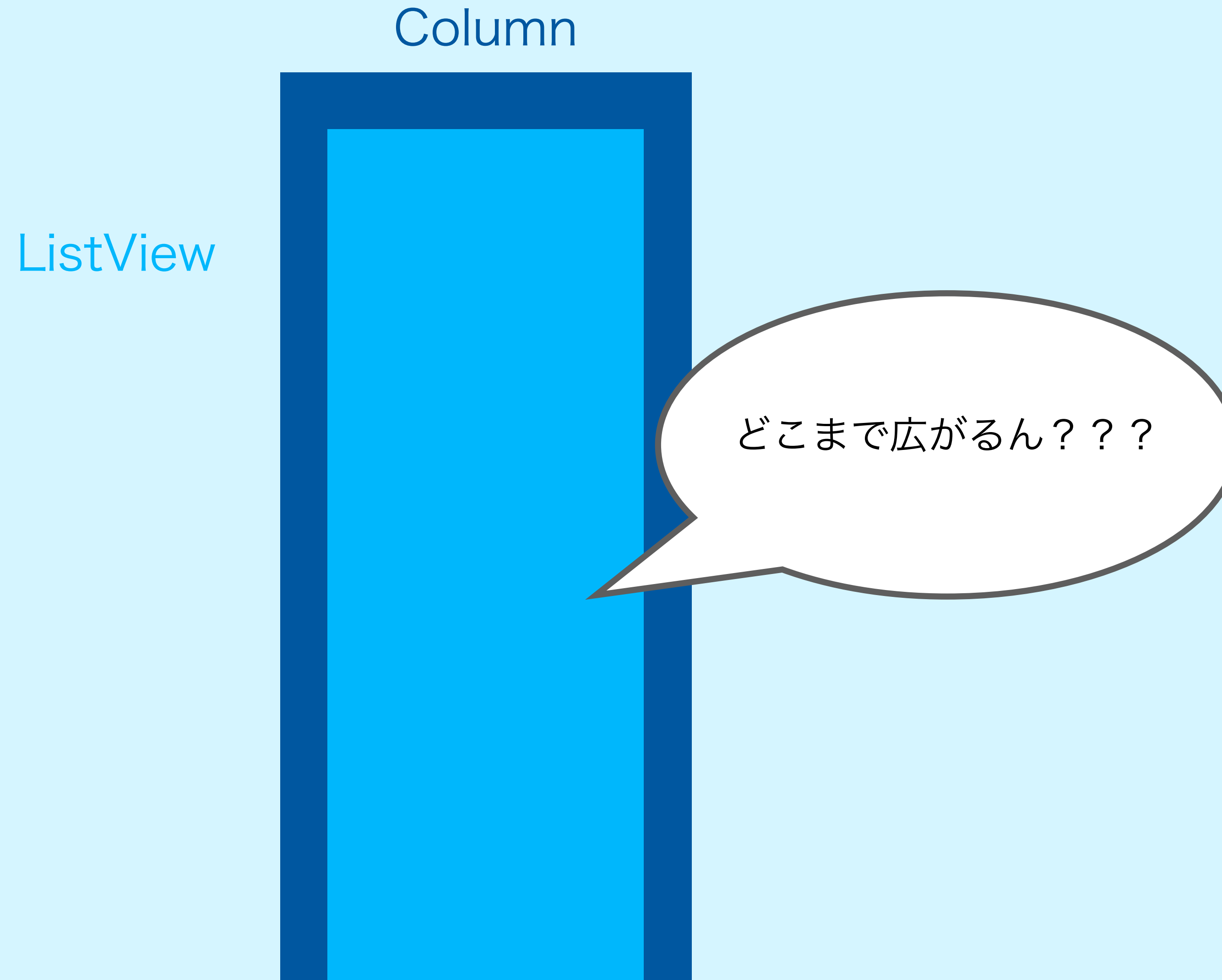
Column の中に ListView を入れるときは注意！

- ListView も広がるうとしてくる性質を持つのでエラーになります



Columnの中に ListView を入れるときは注意！

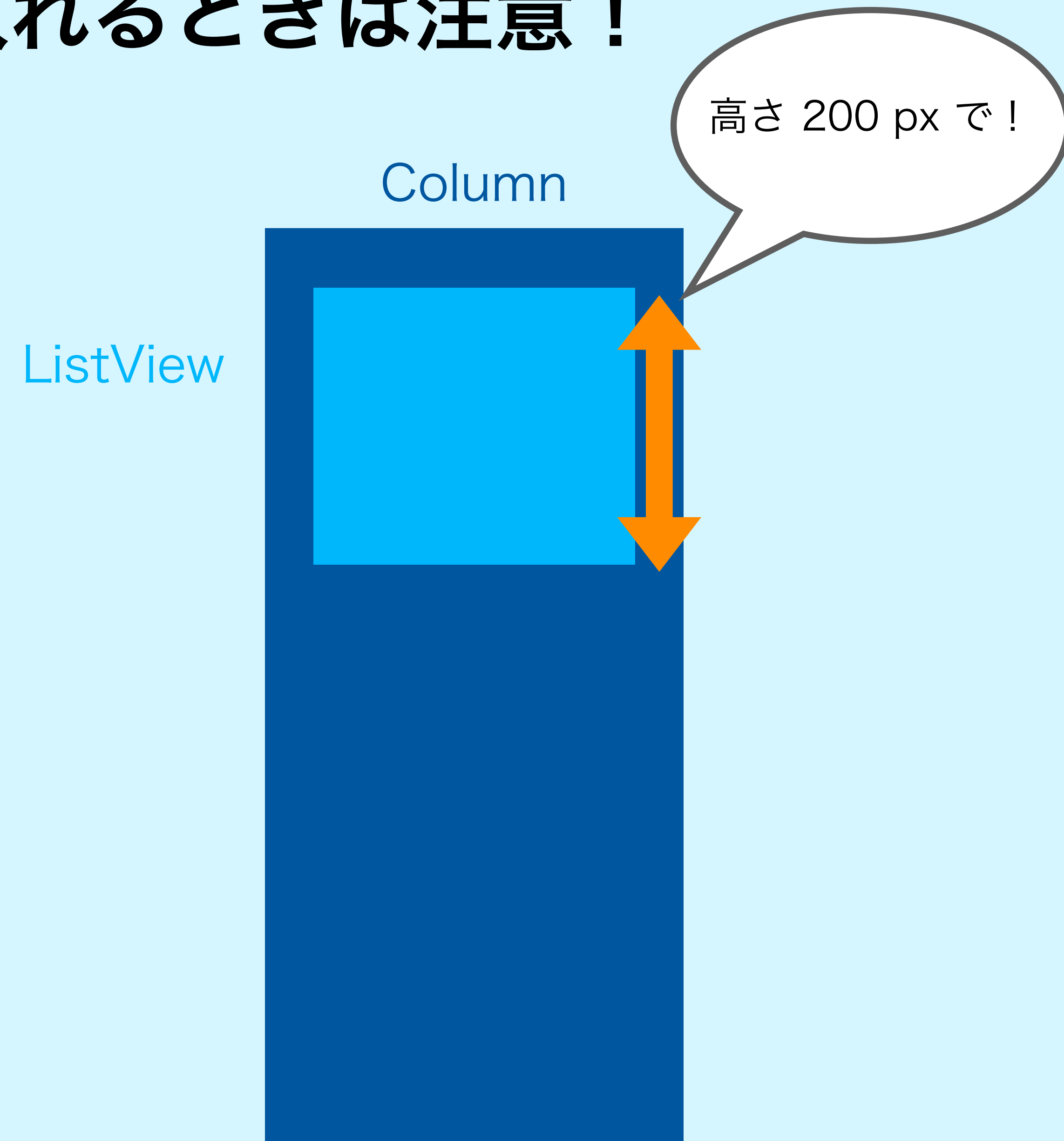
- ListView も広がるうとしてくる性質を持つのでエラーになります



Columnの中に ListView を入れるときは注意！

- 解決策

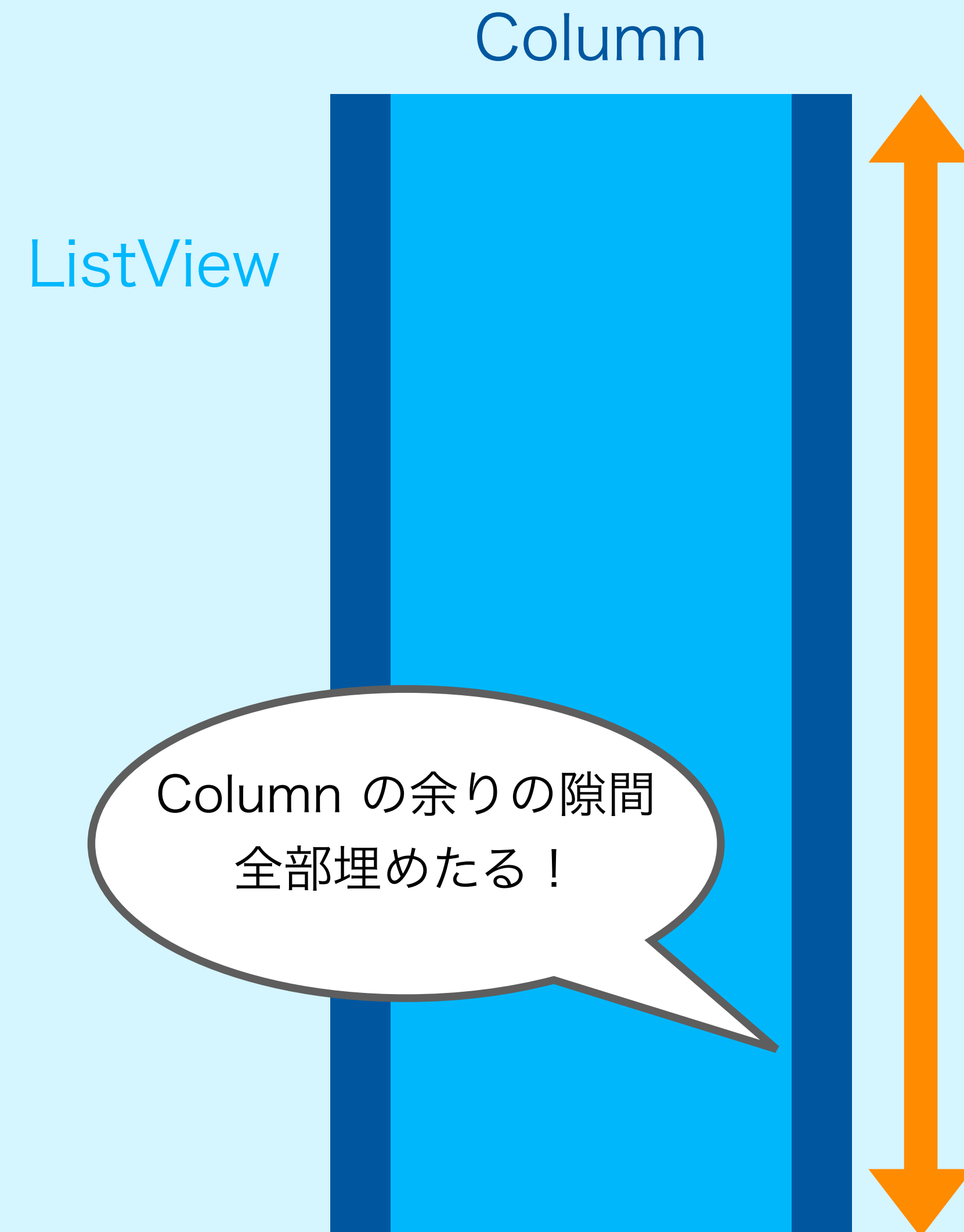
- ① ListView の高さを固定する
- ② Expanded という Widget を使う



Column の中に ListView を入れるときは注意！

- 解決策

- ① ListView の高さを固定する
- ② Expanded という Widget を使う



Column の中に ListView を入れるときは注意！

- 解決策

- ① ListView の高さを固定する
- ② Expanded という Widget を使う
- ③ リストの最初の要素にタイトルを持たせてあげる！**

ListView のみにする

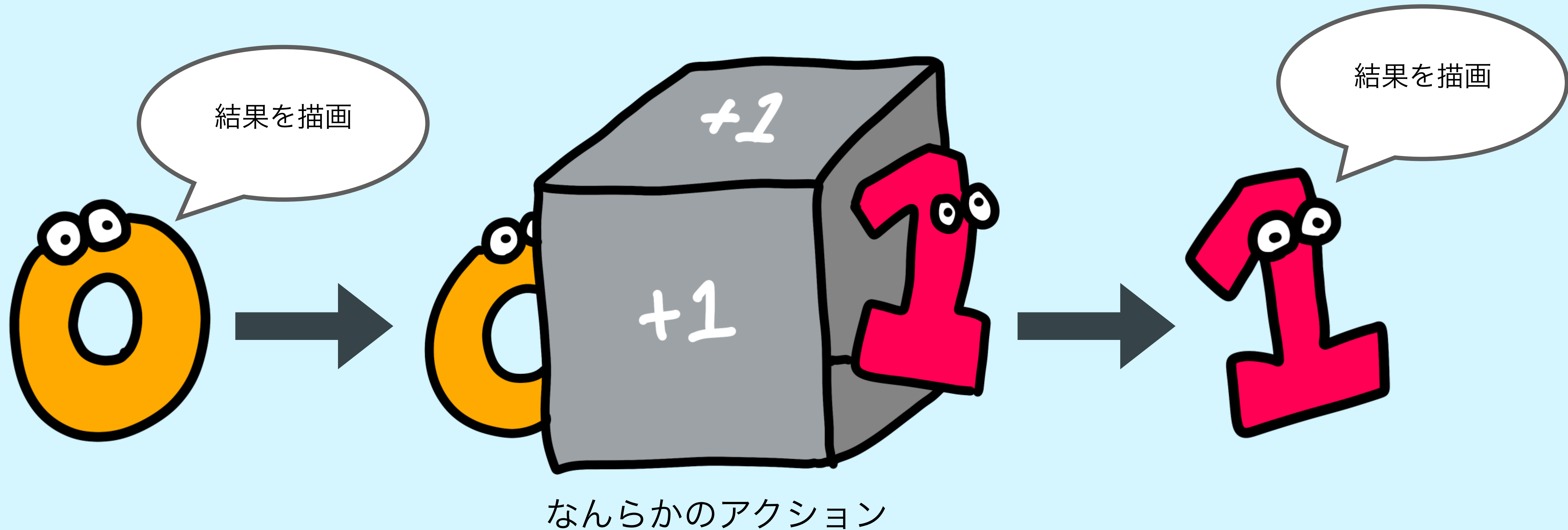


#2 State と StatelessWidget/StatefulWidget

- カウントアップして表示はどうやっているのだろうか？
 - そもそも **State** ってなあに？ (別ページで説明)
- **StatefulWidget** の紹介
- State を追加してみよう
 - 2倍していくボタン
- **State は小さく持つようにしよう！**
 - StatelessWidget の紹介

State ってなあに？

- State = 状態
 - State が変わると画面更新される（ build 内が再実行される）



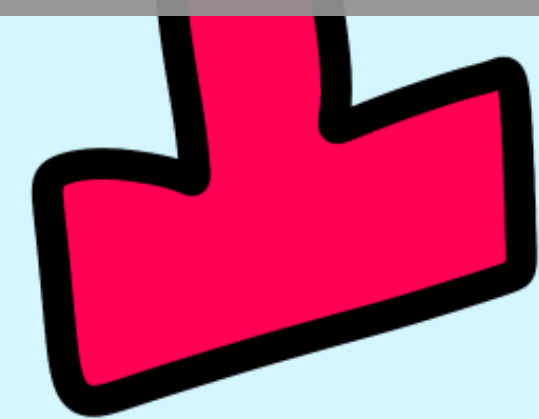
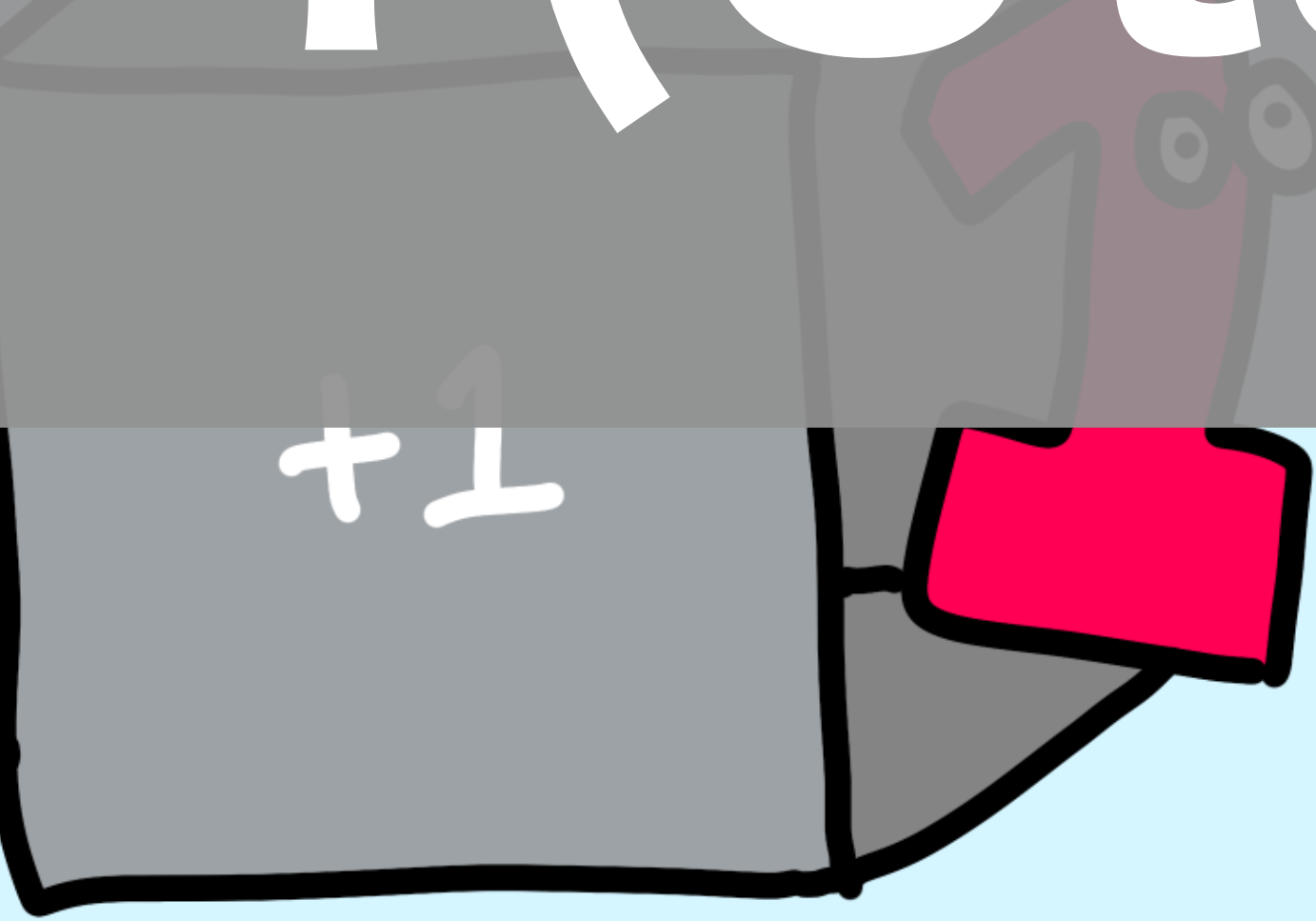
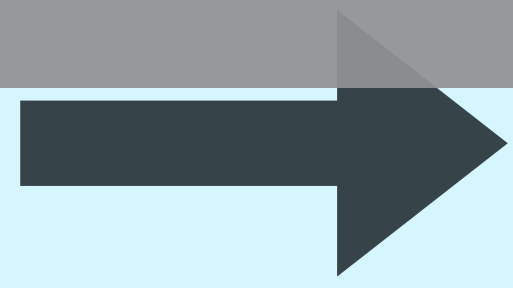
State ってなあに？

- State = 状態

- State が変わると画面更新される (build 内 が再実行される)

宣言的 UI を抽象化すると
状態を引数として UI を返す
副作用を持たない関数である

UI = f(State)

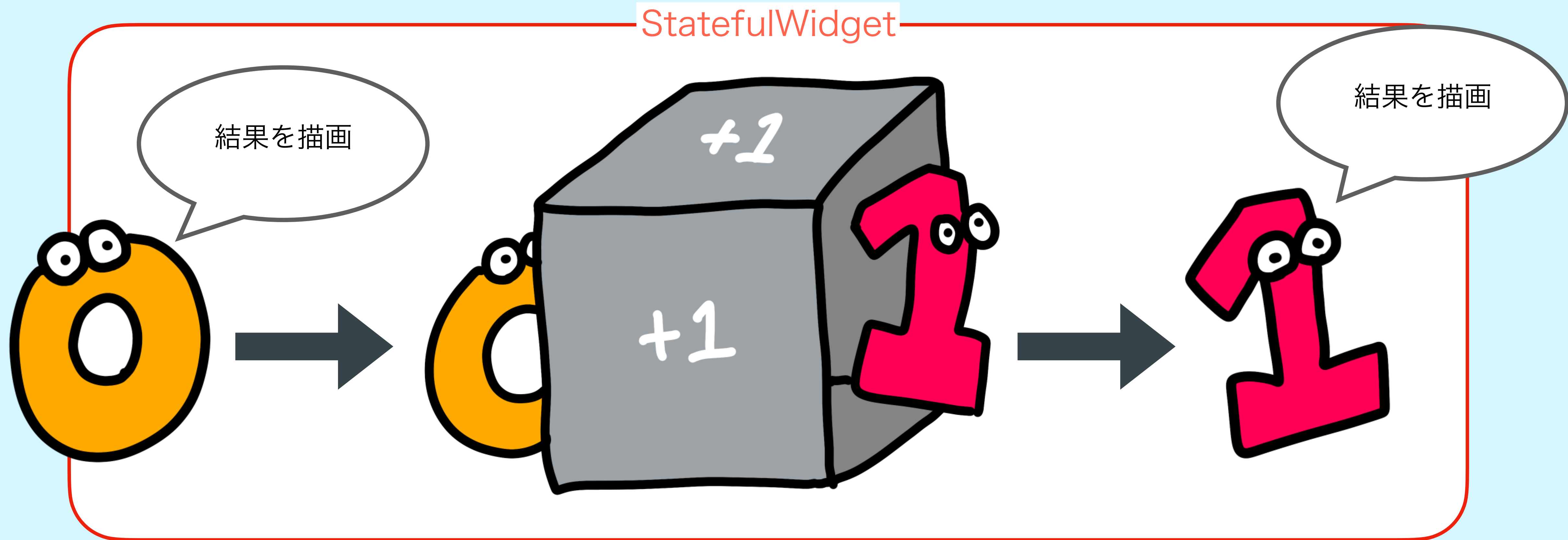


なんらかのアクション

結果を描画

State ってなあに？

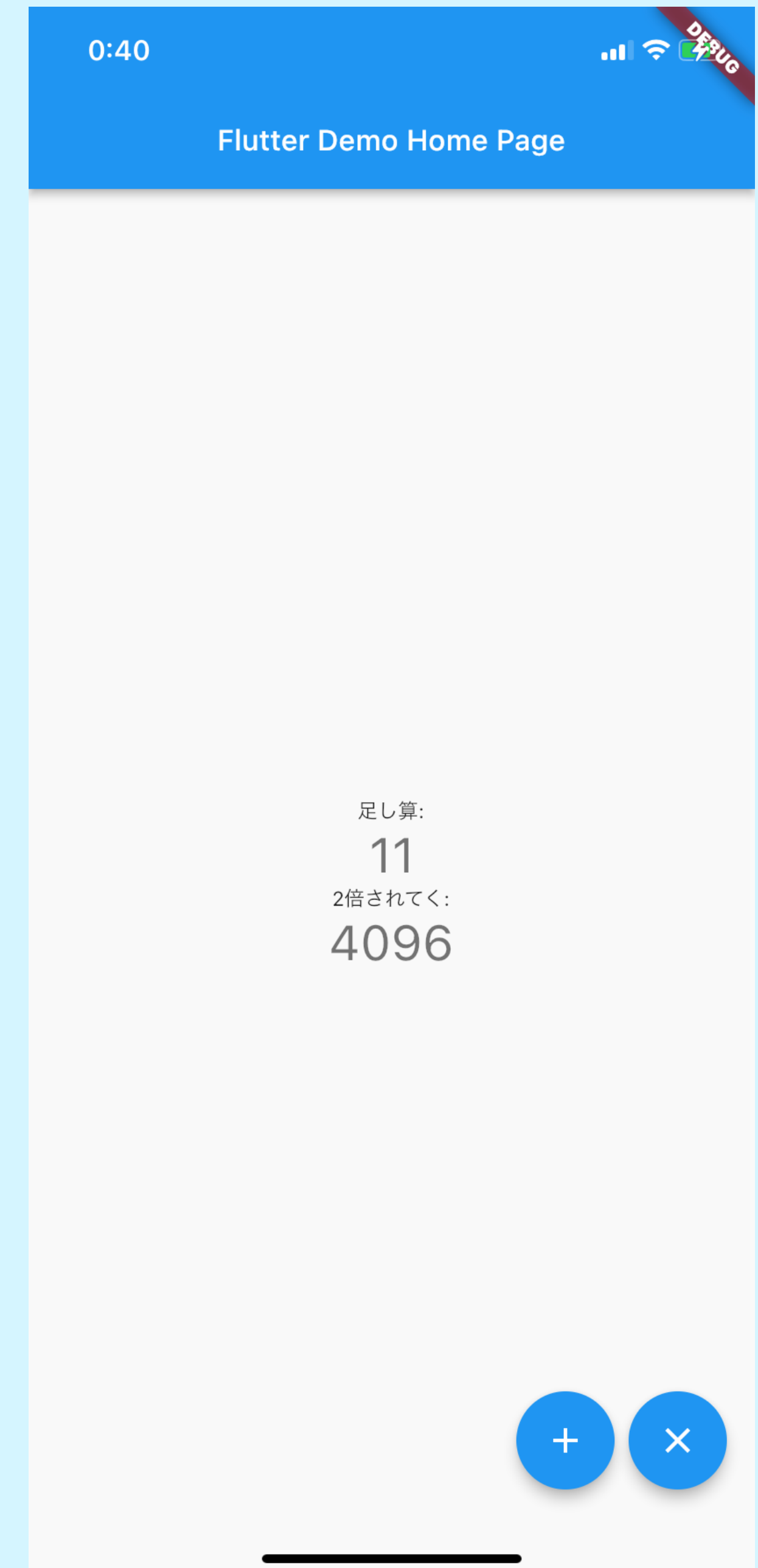
- State を使うウィジェットを **StatefulWidget** という
- 逆に使わないウィジェットは **StatelessWidget** という



#2 Try! 2倍になる state を追加しよう

<https://github.com/mixigroup/2023BeginnerTrainingFlutter/pull/3>

- ライブコーディングします！
- State を新たに用意しよう
 - その State を 2 倍していくボタンを追加しよう



#2 Try! 2倍になる state を追加しよう

- 新しい State を定義
- 2倍するメソッドを定義

```
void _twice() {  
  setState(() {  
    _counter2 = _counter2 * 2;  
  });  
}
```

- 2倍するメソッドを呼ぶボタンを追加

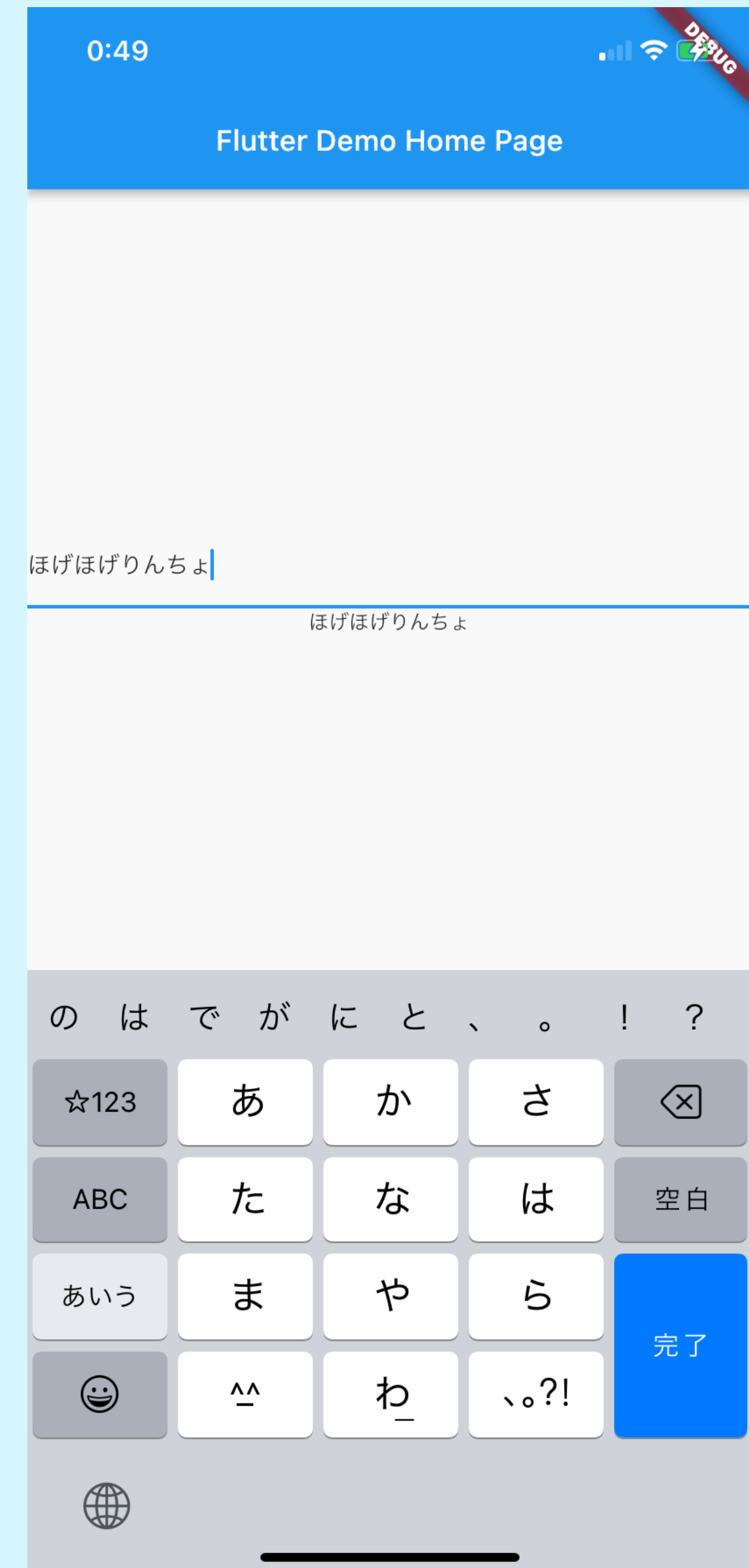
```
FloatingActionButton(  
  onPressed: () {  
    _twice();  
  },
```



#3 Try! TextFiled を使ってみよう

<https://github.com/mixigroup/2023BeginnerTrainingFlutter/pull/4>

- ライブコーディングします！
- TextFiled を使ってみよう
 - 文字打ったたびに下にテキスト表示させるには？
 - **State** にしなきゃいけないね！

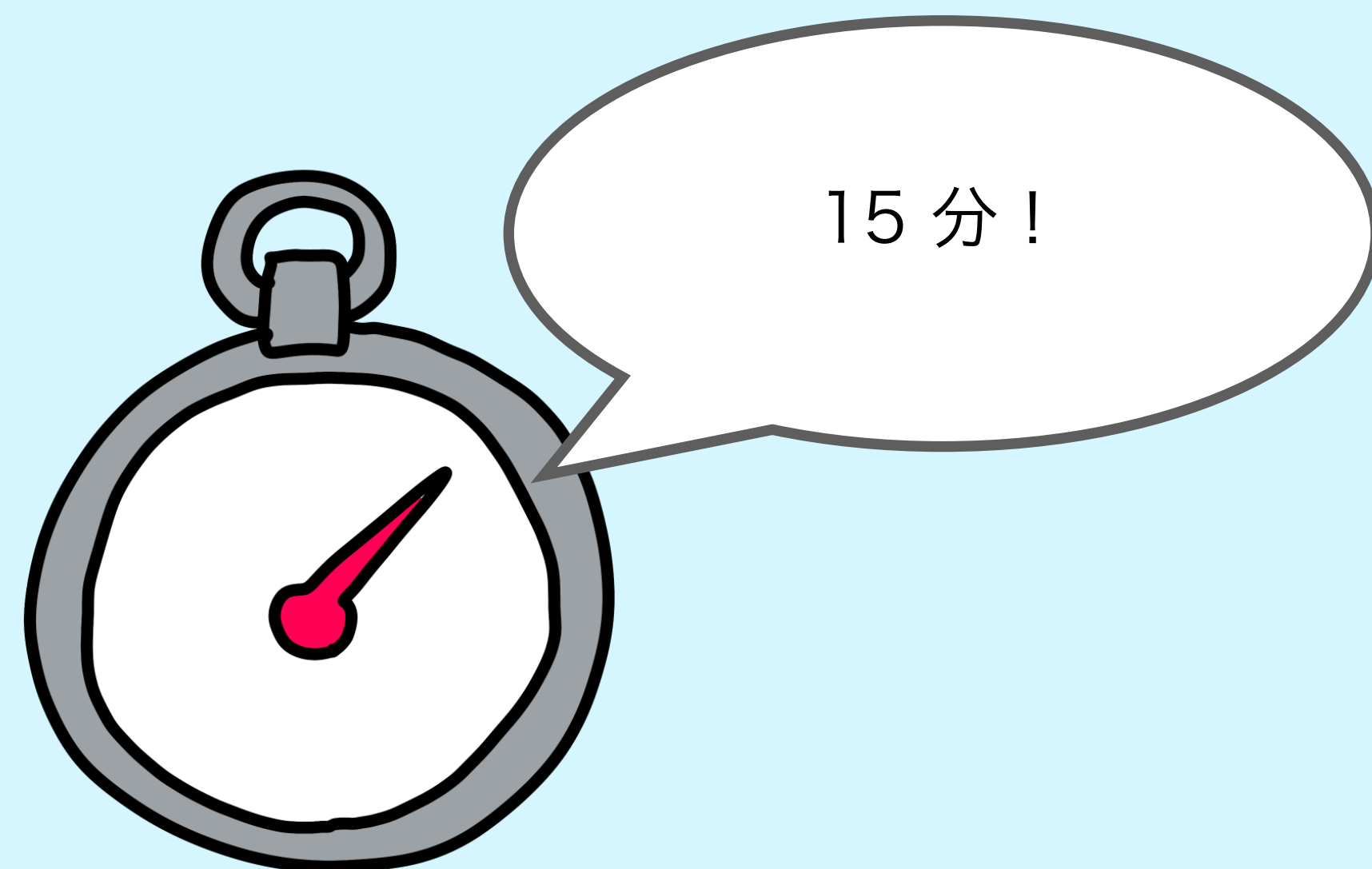


#3 Challenge! 文字表示ボタンを作ろう

- さっきは文字を打つたび表示だった
 - ボタンを押したら文字が表示されるようにしてみよう！

答え

<https://github.com/mixigroup/2023BeginnerTrainingFlutter/pull/20>



#3 Challenge! 文字表示ボタンを作るう

- `setState` のタイミングが重要
- `TextFiled` の `onChanged` では `setState` しないように
 - 再描画されてしまうため

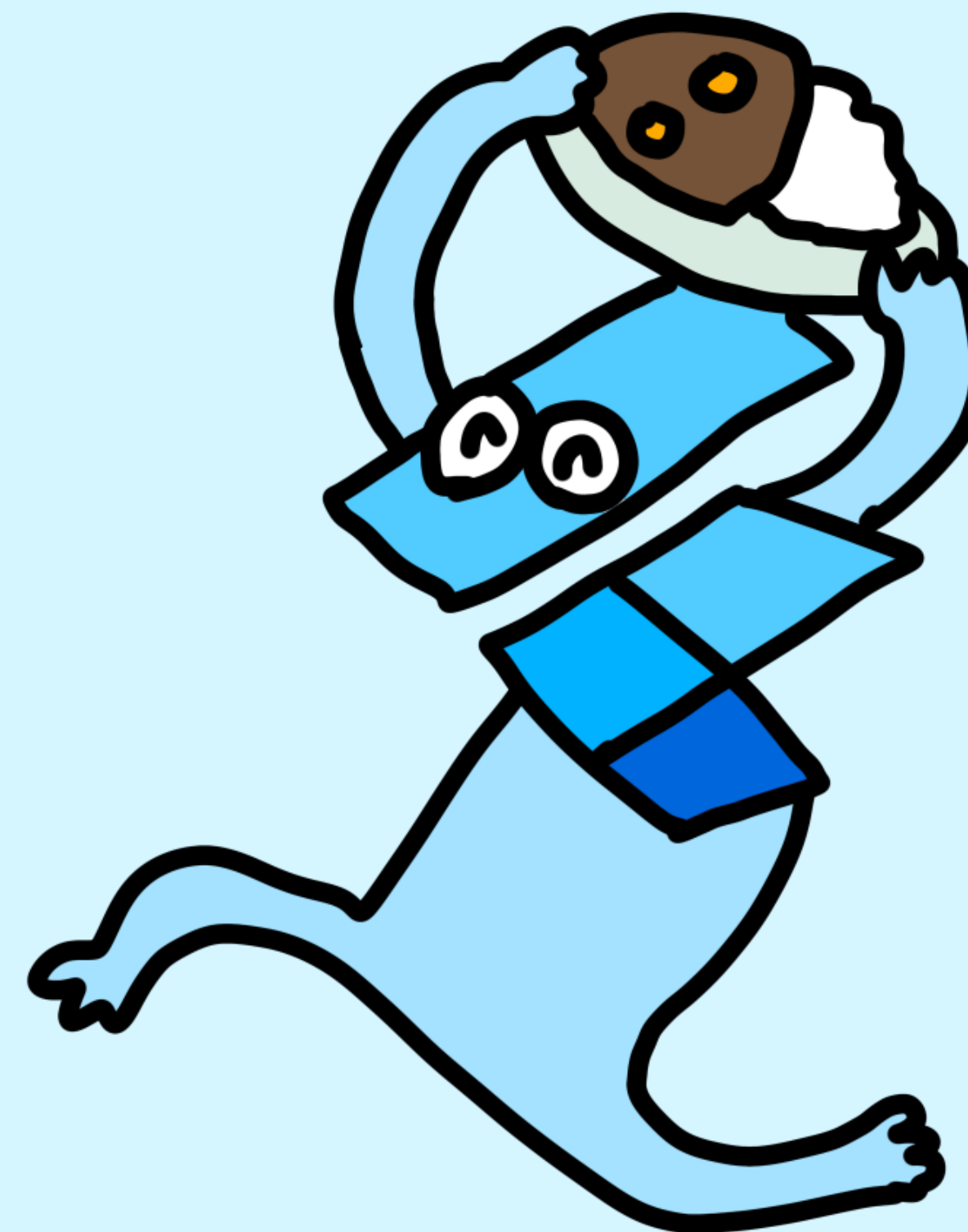
```
TextField(  
    onChanged: ((value) {  
        tmpText = value;  
    }),
```

- ボタン押した時に `setState` するように！

```
IconButton(  
    onPressed: () {  
        setState(() {  
            showText = tmpText;  
        });  
    },
```



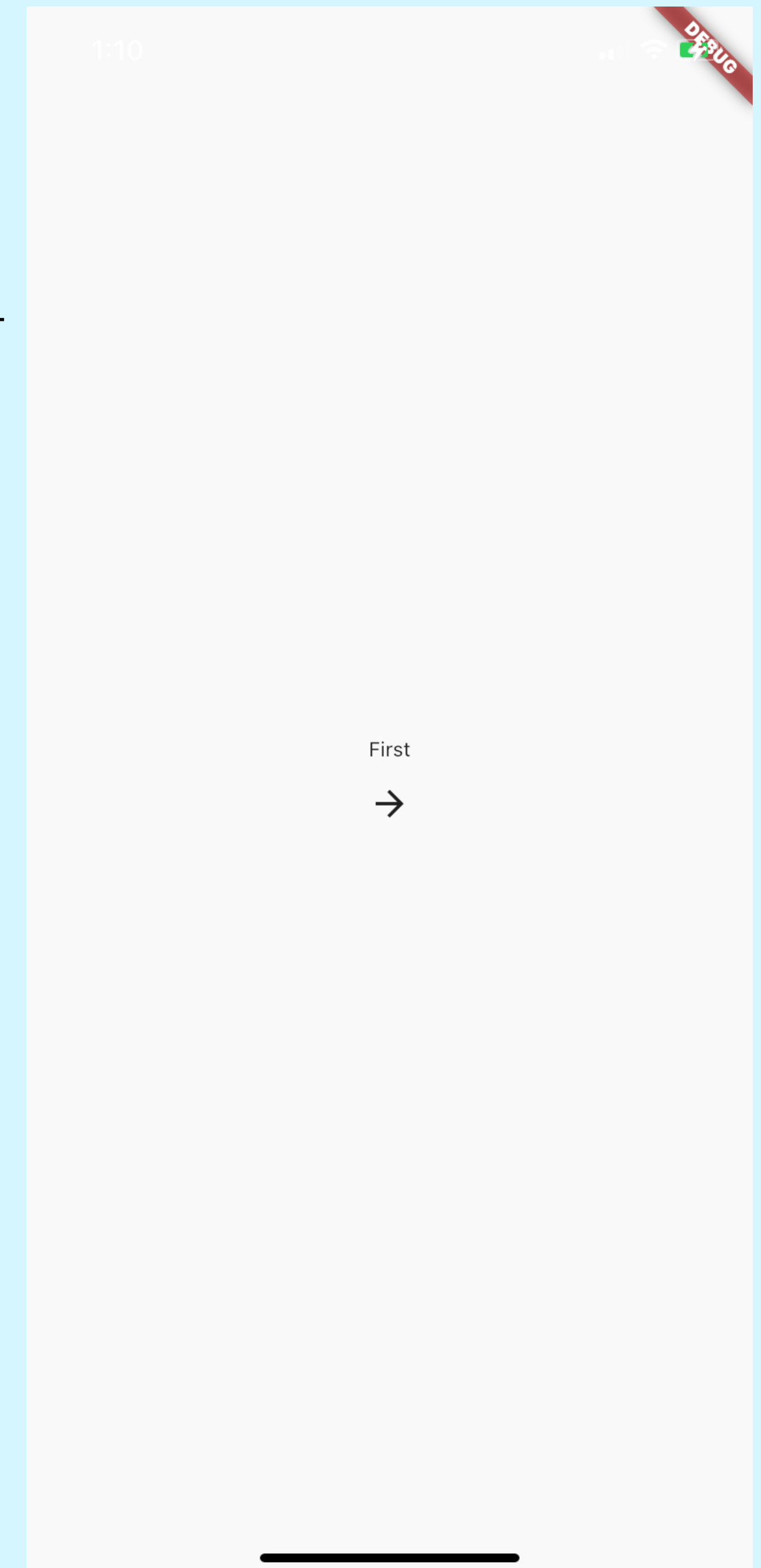
ランチ



#4 Try! 画面遷移してみよう

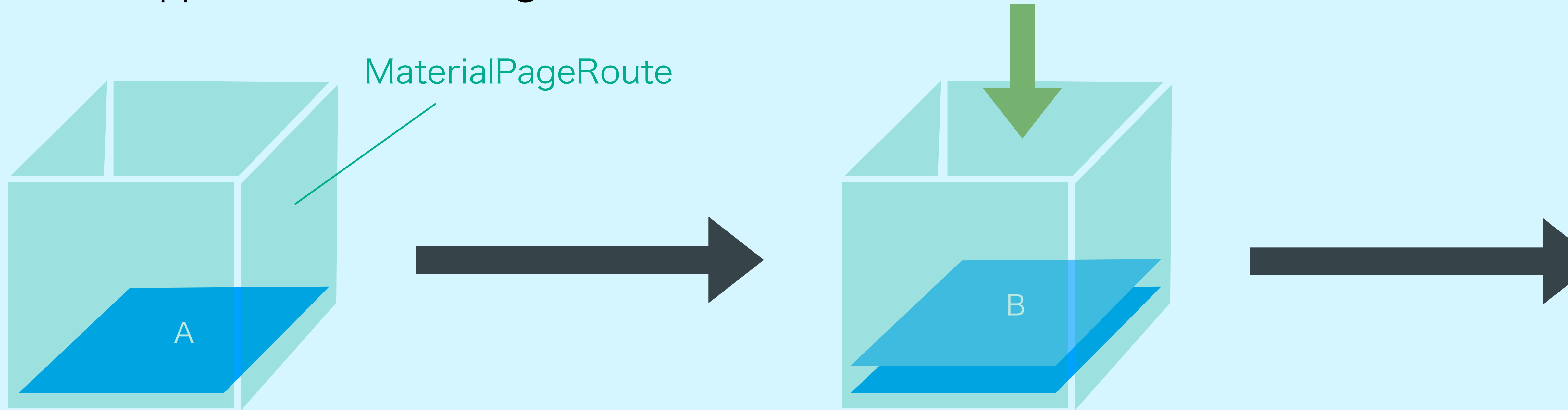
<https://github.com/mixigroup/2023BeginnerTrainingFlutter/pull/5>

- Navigator 1.0 のやり方です
- ライブコーディングします！
- FirstPage と SecondPage を作成
 - Scaffold ないと土台がないから変な表示になっちゃうよ～
- **push** の説明 (別ページで説明)
 - あらかじめ Route 登録しておくると便利だよ～
- **pop** の説明 (別ページで説明)



画面遷移

MaterialApp が持っている **Navigator** を使って画面遷移する



```
Navigator.push(  
  context,  
  MaterialPageRoute<void>(  
    builder: (BuildContext context) => B(),  
  ),  
);
```

```
Navigator.pop(context)
```


画面遷移

MaterialApp が持っている **Navigator** を使って画面遷移する



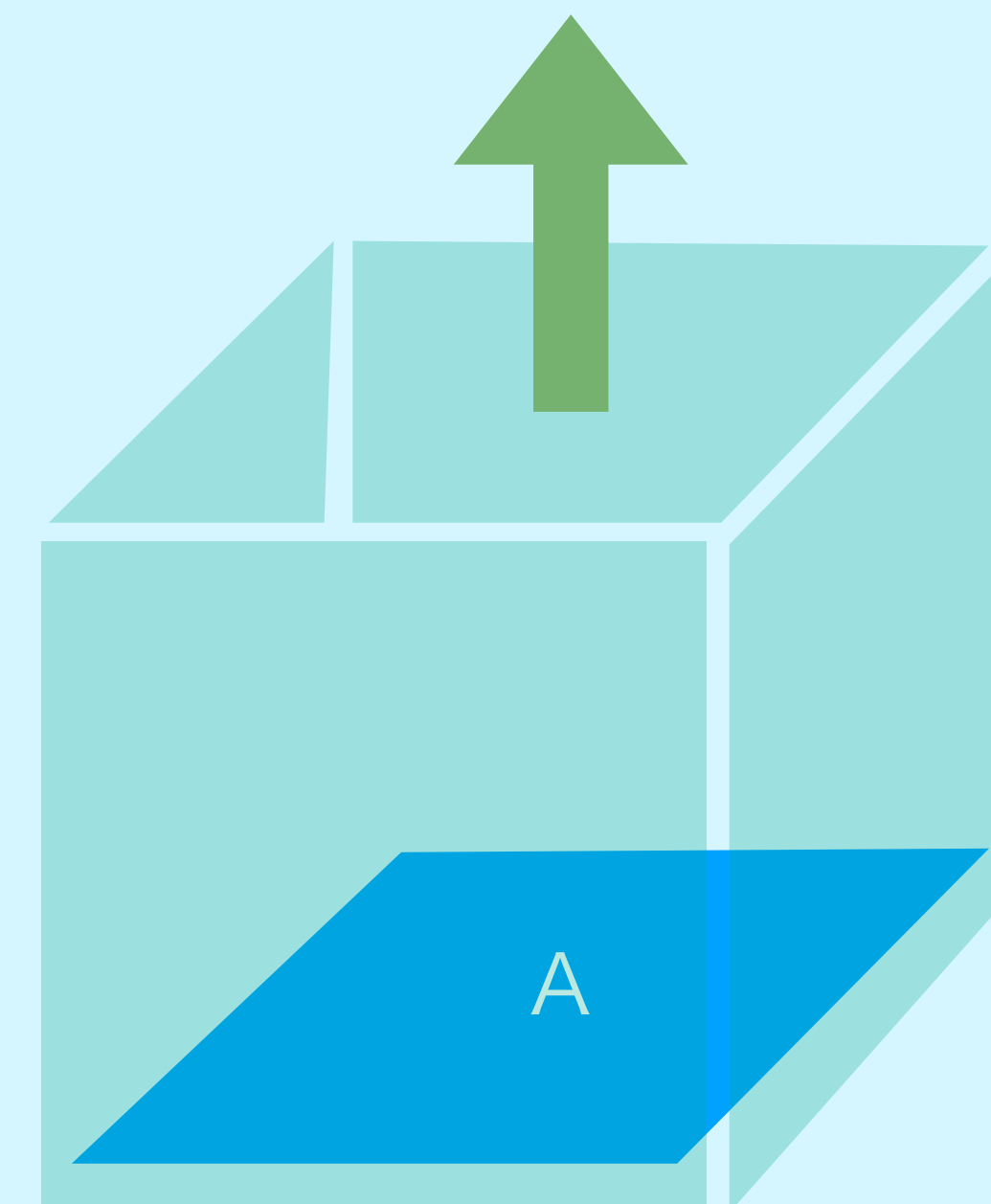
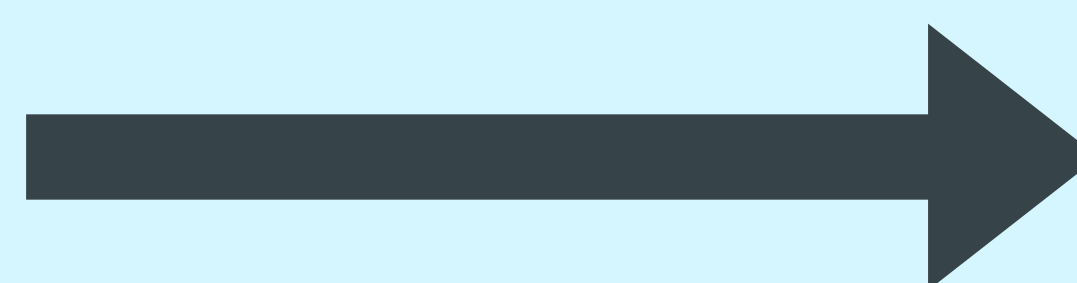
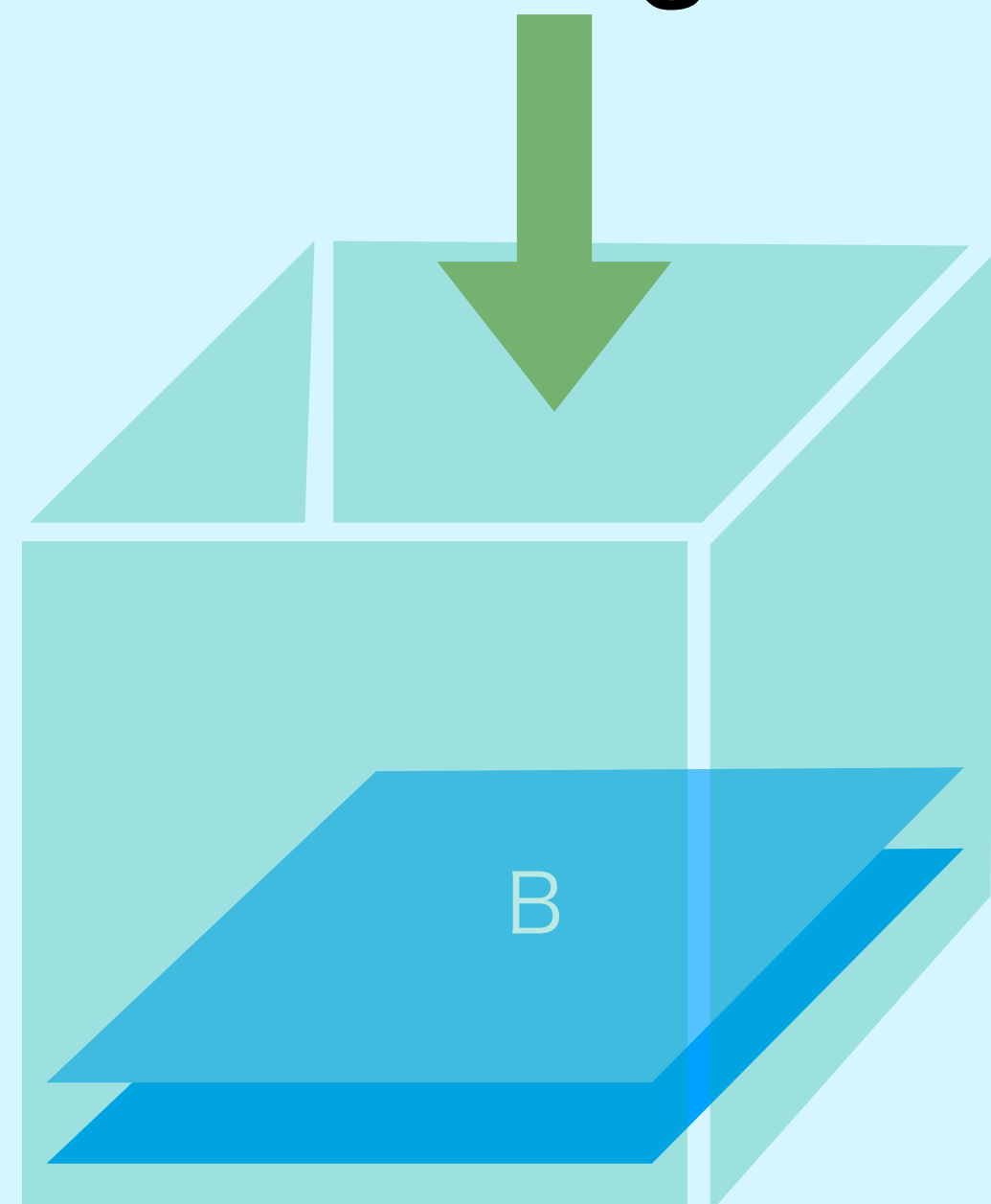
```
Navigator.push(  
  context,  
  MaterialPageRoute<void>(  
    builder: (BuildContext context) => B(),  
  ),  
);
```

```
Navigator.pop(context)
```

画面遷移

MaterialApp が持っている **Navigator** を使って画面遷移する

MaterialPageRoute

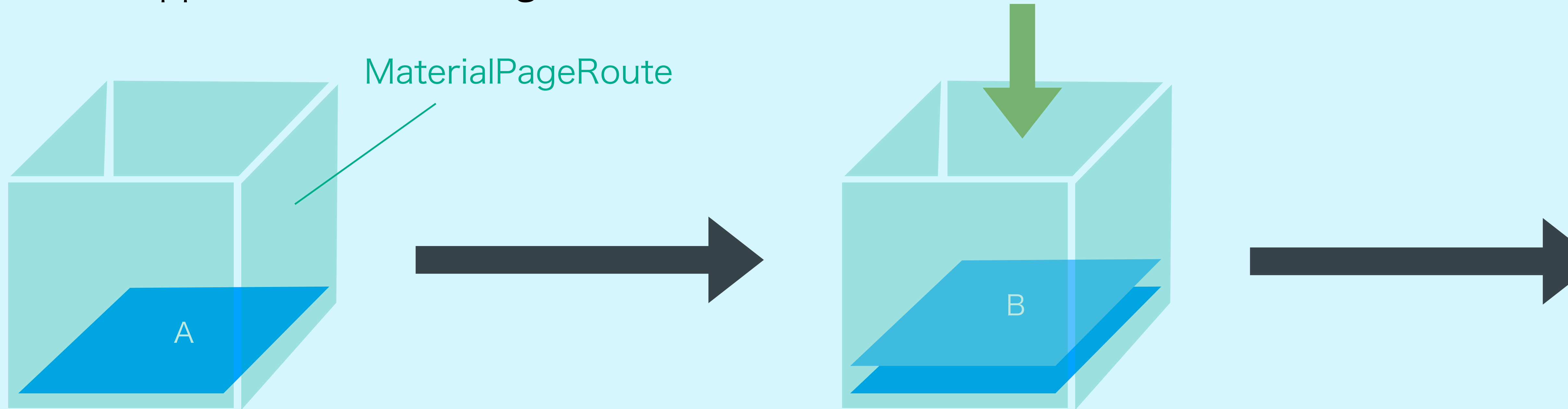


```
route<void>(
  BuildContext context) => B(),
```

```
Navigator.pop(context);
```

画面遷移

MaterialApp が持っている **Navigator** を使って画面遷移する



```
Navigator.push(  
  context,  
  MaterialPageRoute<void>(  
    builder: (BuildContext context) => B(),  
  ),  
);
```

```
Navigator.pop(context)
```

画面遷移

MaterialApp が持っている **Navigator** を使って画面遷移する



```
Navigator.push(  
  context,  
  MaterialPageRoute<void>(  
    builder: (BuildContext context) => B(),  
  ),  
);
```

```
Navigator.pop(context)
```

MaterialApp にあらかじめ Route を登録しておこう！

```
return MaterialApp(  
  title: 'Flutter Demo',  
  theme: ThemeData(  
    primarySwatch: Colors.pink,  
  ),  
  initialRoute: '/',  
  routes: <String, WidgetBuilder>{  
    '/': (BuildContext context) => const FirstPage(),  
    '/second': (BuildContext context) => const SecondPage(),  
  },  
);
```

これで呼べるようになる！

```
Navigator.pushNamed(context, "/second");
```

#5 Try! ネットワーク通信してみよう

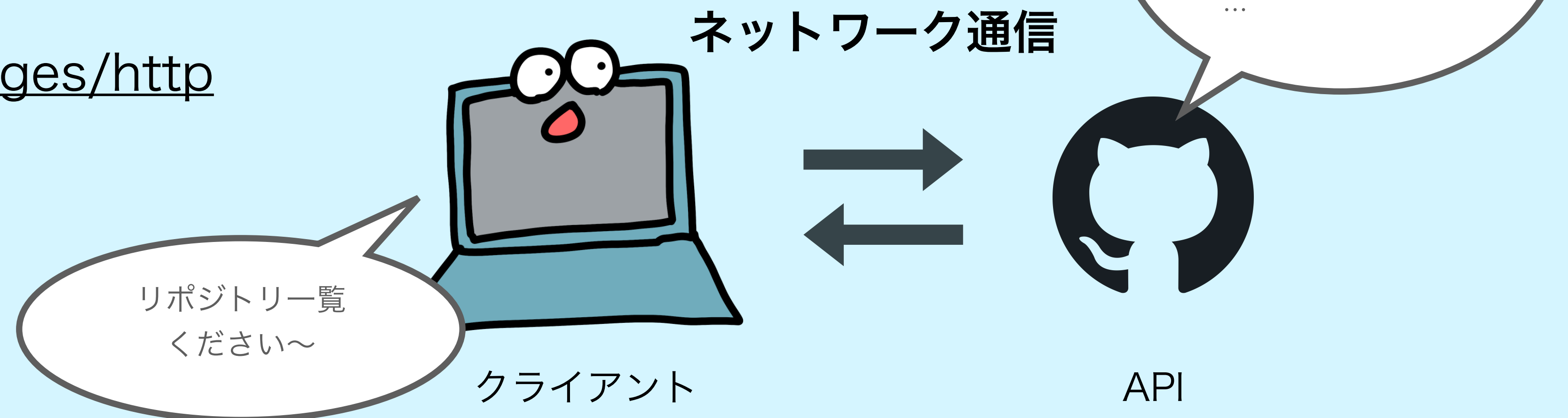
<https://github.com/mixigroup/2023BeginnerTrainingFlutter/pull/6>

- GitHub のリポジトリ一覧を取得する API を叩いてみよう！
 - <https://docs.github.com/ja/rest/repos/repos?apiVersion=2022-11-28#list-repositories-for-a-user>
- まずは curl コマンドで叩いてみよう

```
$ curl -L https://api.github.com/users/kno3a87/repos
```

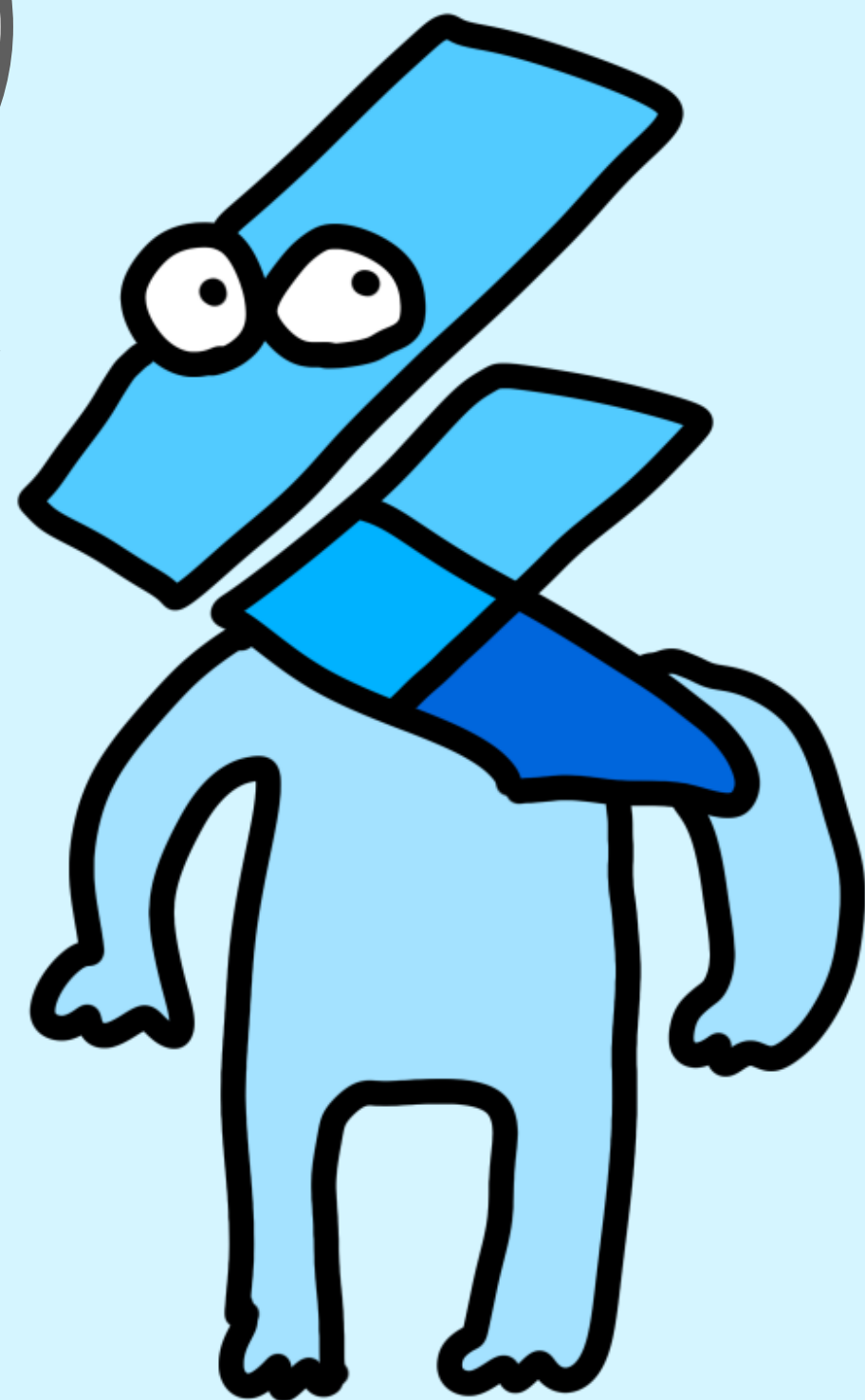
- Flutter で http 通信するために http パッケージをいれよう！

- <https://pub.dev/packages/http>



非同期処理 (async/await)

待ってあげる
(await)



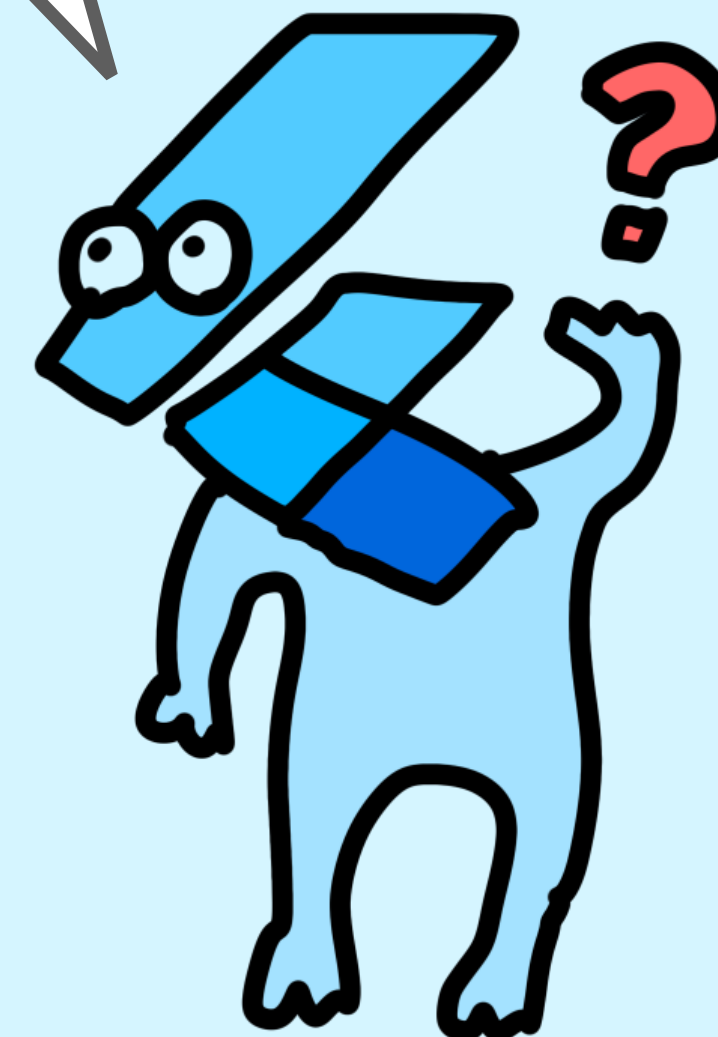
接続中！
探し中！
調べ中！
返し中！

#5 Try! json をパースしよう

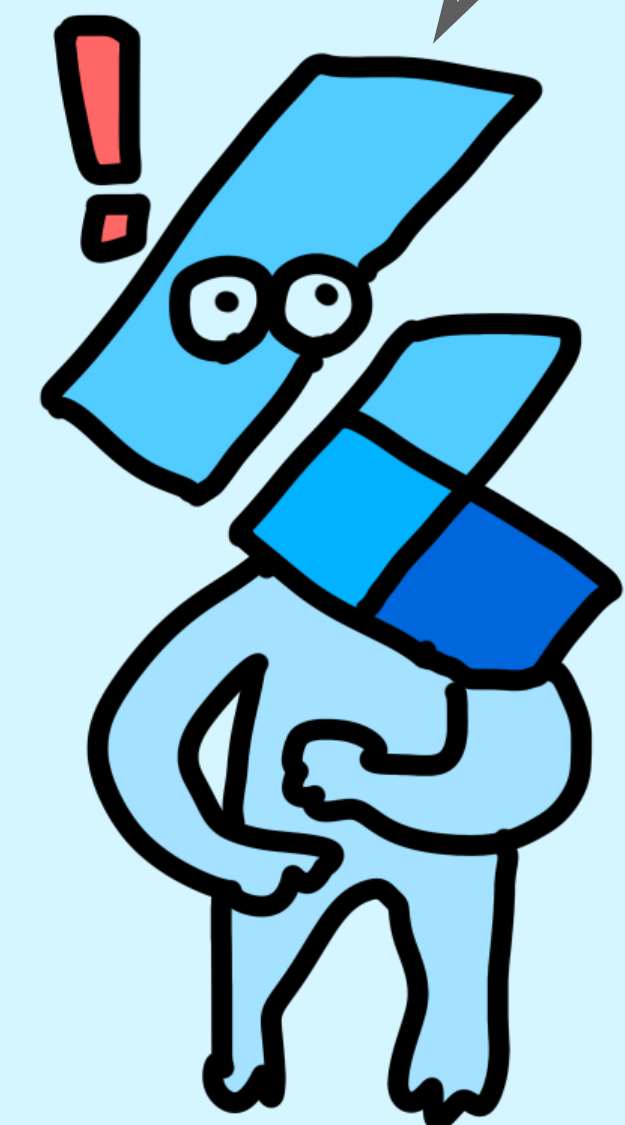
<https://github.com/mixigroup/2023BeginnerTrainingFlutter/pull/7>

- まずは json から dart で扱えるオブジェクトに変換しよう
- hoge['fuga']['piyo'] みたいな取得方法だと文字列 typo しちゃうかも…
 - モデルに変換しちゃおう！
- 使用パッケージ
 - https://pub.dev/packages/json_serializable
 - ```
$ flutter pub run build_runner build
```

```
{
 "name": "kuno"
 "age": "26"
}
```

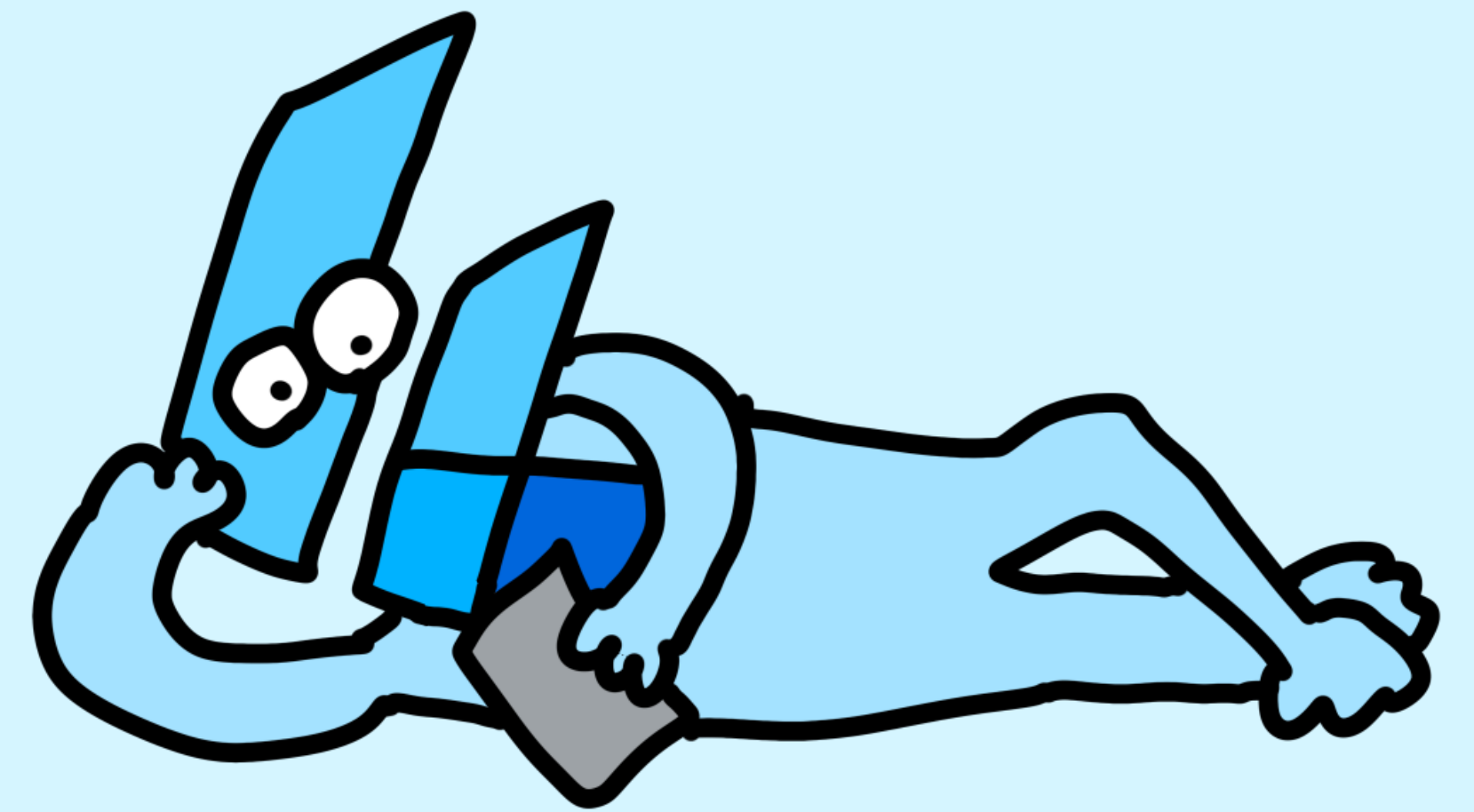


```
user.name は kuno
user.age は 26 !
```

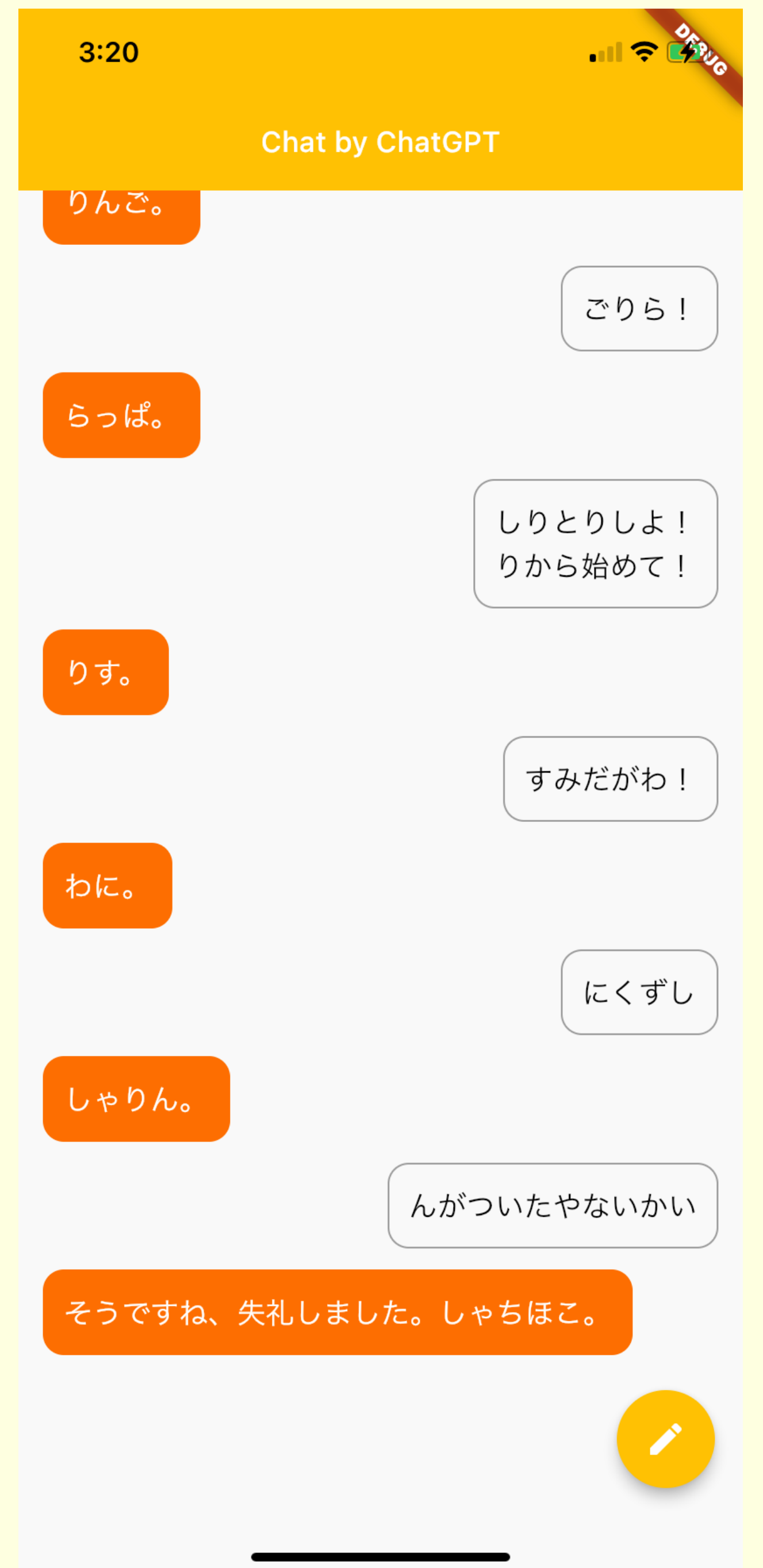
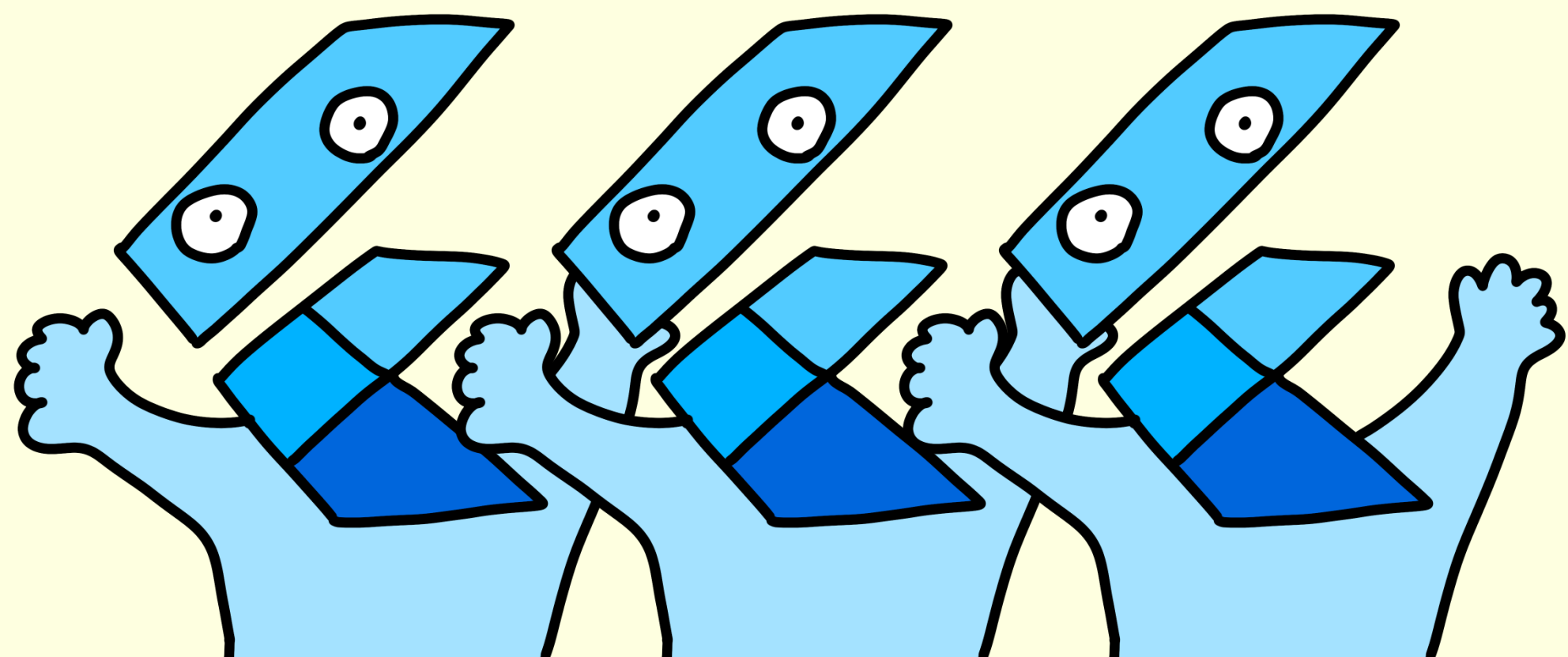


# 10 分間 休憩

次からチームチャレンジになります！



# チームでチャレンジ！ ChatGPT とのチャットアプリを 作ってみよう！



# ChatGPT とは？

- OpenAI が 2022 年 11 月に公開した**人工知能チャットボット**
- ウェブアプリだけでなく API も用意されてる
  - <https://platform.openai.com/docs/api-reference/making-requests>
- API を使う上での注意事項
  - 機密情報や第三者の知的財産権を侵害するようなインプットはしないで！！！！
  - もちろん誹謗中傷も NG
  - アウトプットは個人で利用する範囲にとどめてください



# チームでチャレンジ！

- **基本実装は各々やってください！！**
  - チームにした意図は聞きやすく・教えやすくするためなので**ワイワイ**やってください！
- 最初の問題から解いてくと順番にチャットアプリが出来上がってく想定です！
  - **最後まで辿り着けないのを想定した問題量なので焦らず！**
- 休憩も自由にとってね！**バイロン 14:30-15:30** で閉まるので**気をつけて！！**
- 定期的にチューターが見て回ります！
  - 何か質問・相談あったら即座にチューター呼んでもらって大丈夫です！





# TeamChallenge! 投稿ページの作成

- 投稿するページを作ろう！
- 閉じたら書いた文字が表示されるようにしてみよう

答え

<https://github.com/mixigroup/2023BeginnerTrainingFlutter/pull/8>



# TeamChallenge! ChatGPT と接続

- ChatGPT API のトークンは Dropbox にあります！
  - Git に push しないでね！！！！
  - .env ファイルを作成して .gitignore に書いておくと安心かも
  - ```
$ flutter run --dart-define=MY_TOKEN=<ここにトークン>
```
- さっきの投稿ページで書いた内容を API に投げてみよう
 - 結果も表示してみよう！

答え

<https://github.com/mixigroup/2023BeginnerTrainingFlutter/pull/9>

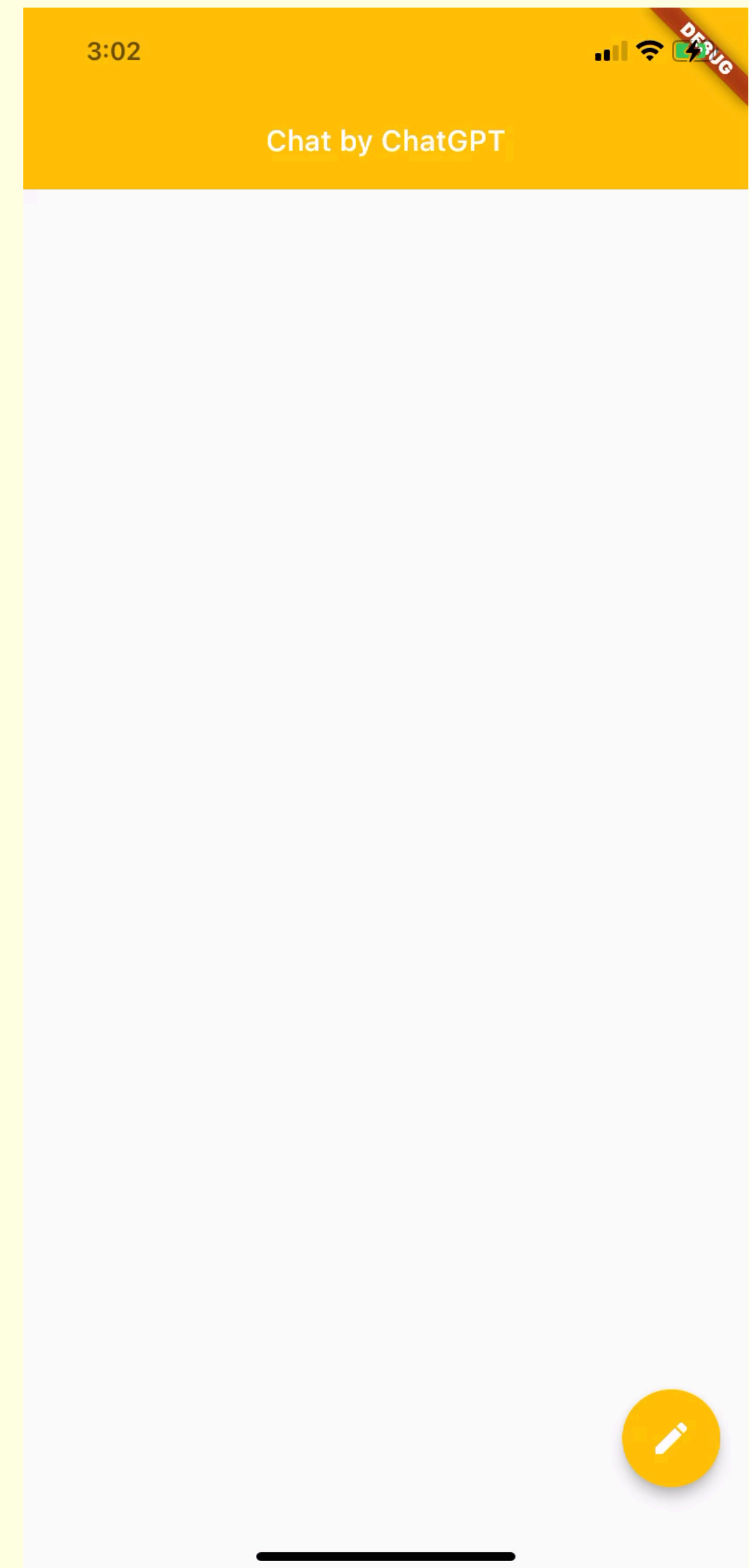


TeamChallenge! json をモデルに変換

- さっきの try! を参考にやってみよう！
- 挙動自体はさっきと変わらなくて OK
- response が入れ子になってるのはどうしたら良いかな？

答え

<https://github.com/mixigroup/2023BeginnerTrainingFlutter/pull/10>



TeamChallenge! ローディング表示しよう

- API を叩いてから結果が返ってくるまでの間ローディングを表示
- ローディングウィジェットは標準で用意されてるので探してみてね

答え

<https://github.com/mixigroup/2023BeginnerTrainingFlutter/pull/11>



TeamChallenge! ローカルに保存しよう

- 今回は Hive というパッケージを使用
 - <https://pub.dev/packages/hive>
 - 端末のファイルシステムに保存してく key-value ストア
 - read も write もめっちゃ早い！！！！
- 一旦 print デバッグなどで保存確認できれば OK です！



答え

<https://github.com/mixigroup/2023BeginnerTrainingFlutter/pull/12>

TeamChallenge! 文脈を理解させよう

- ChatGPT は文脈も理解して返答してくれます！
- 指示語を使うと文脈理解してるか確認できます
 - しりとりとかでもよさそう！

答え

<https://github.com/mixigroup/2023BeginnerTrainingFlutter/pull/13>

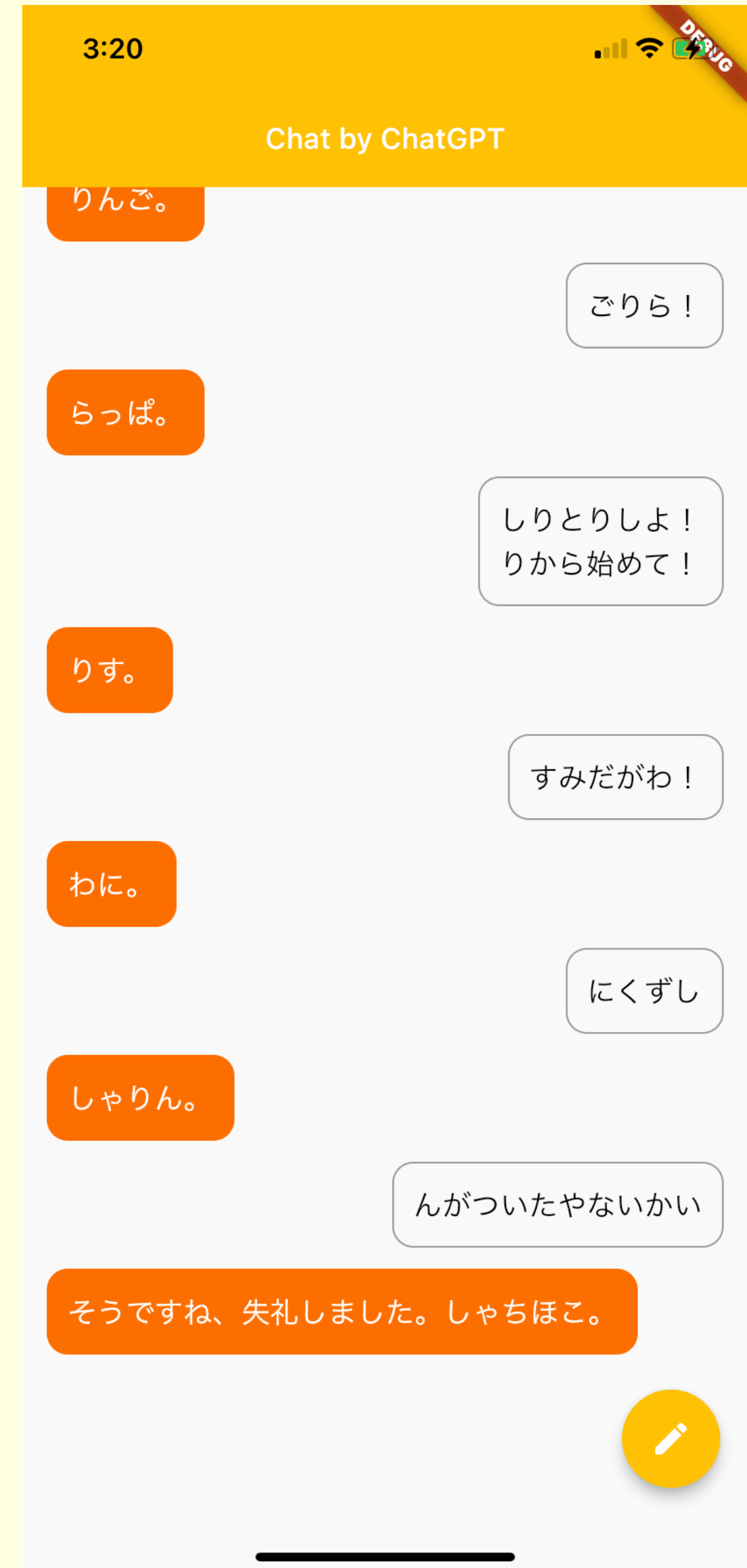


TeamChallenge! リスト表示しよう

- 保存したメッセージ内容をリスト表示にしてチャットっぽくしよう！
- デザインも自分好みに遊んでみて！

答え

<https://github.com/mixigroup/2023BeginnerTrainingFlutter/pull/16>

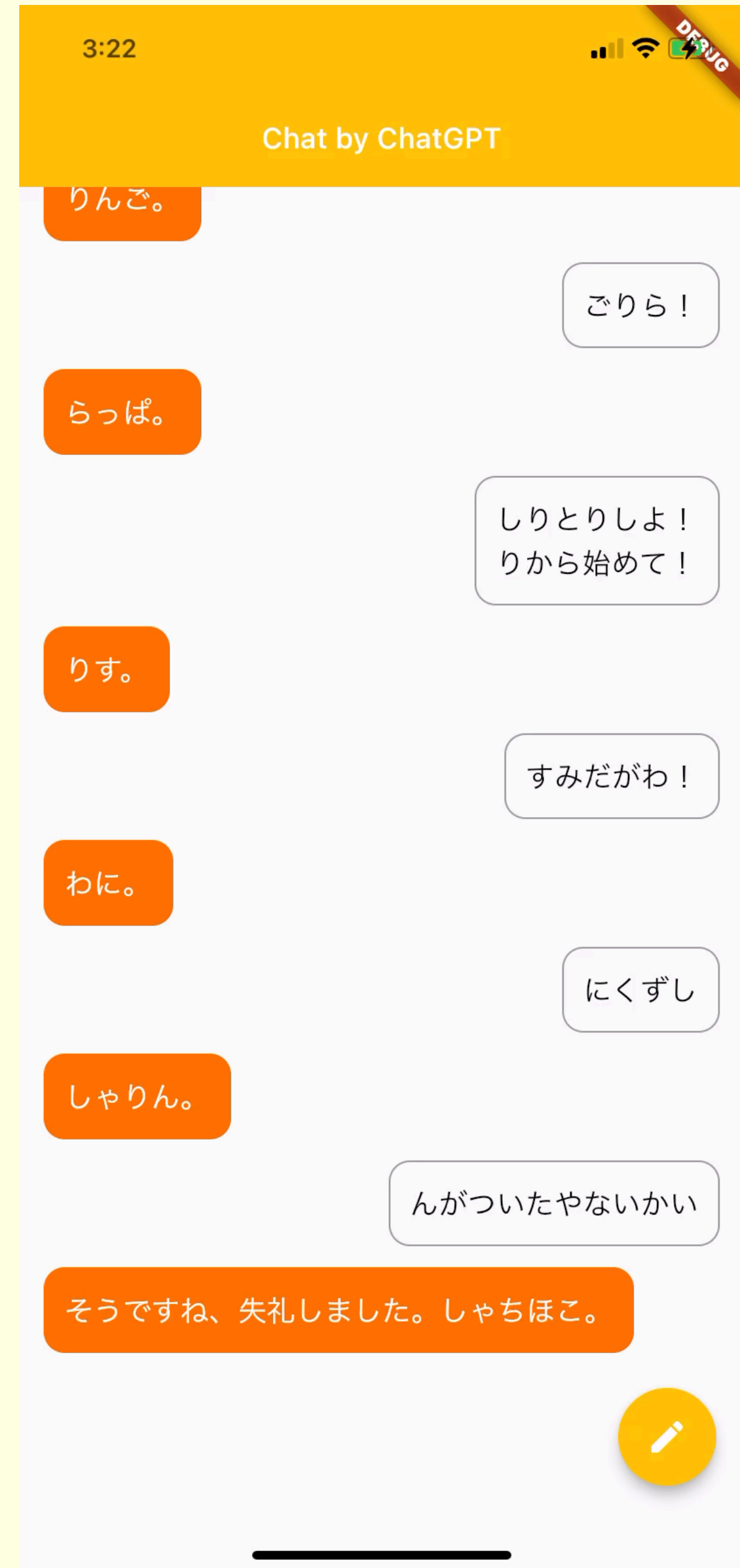


TeamChallenge! エラーを表示

- ・ タイムアウトを設けてみよう
 - ・ 3 秒経ってもレスポンスがなければエラーを返すようになど
- ・ 何かしらのエラーを受け取ったらダイアログの表示をしよう！
 - ・ ダイアログじゃなくてスナックバーでも OK

答え

<https://github.com/mixigroup/2023BeginnerTrainingFlutter/pull/14>



TeamChallenge! メッセージを全削除

- ・ 今まで保存したメッセージを全削除するボタンをつけよう！
 - ・ デバッグしやすくなります

答え

<https://github.com/mixigroup/2023BeginnerTrainingFlutter/pull/15>



TeamChallenge! Hive でもモデルを使う

- json の時みたいに文字列指定する箇所が出てきて大変！
 - モデルを使えるようにしよう！
- 使用パッケージ
 - https://pub.dev/packages/hive_generator
 - https://docs.hivedb.dev/#/custom-objects/type_adapters

答え

<https://github.com/mixigroup/2023BeginnerTrainingFlutter/pull/22>

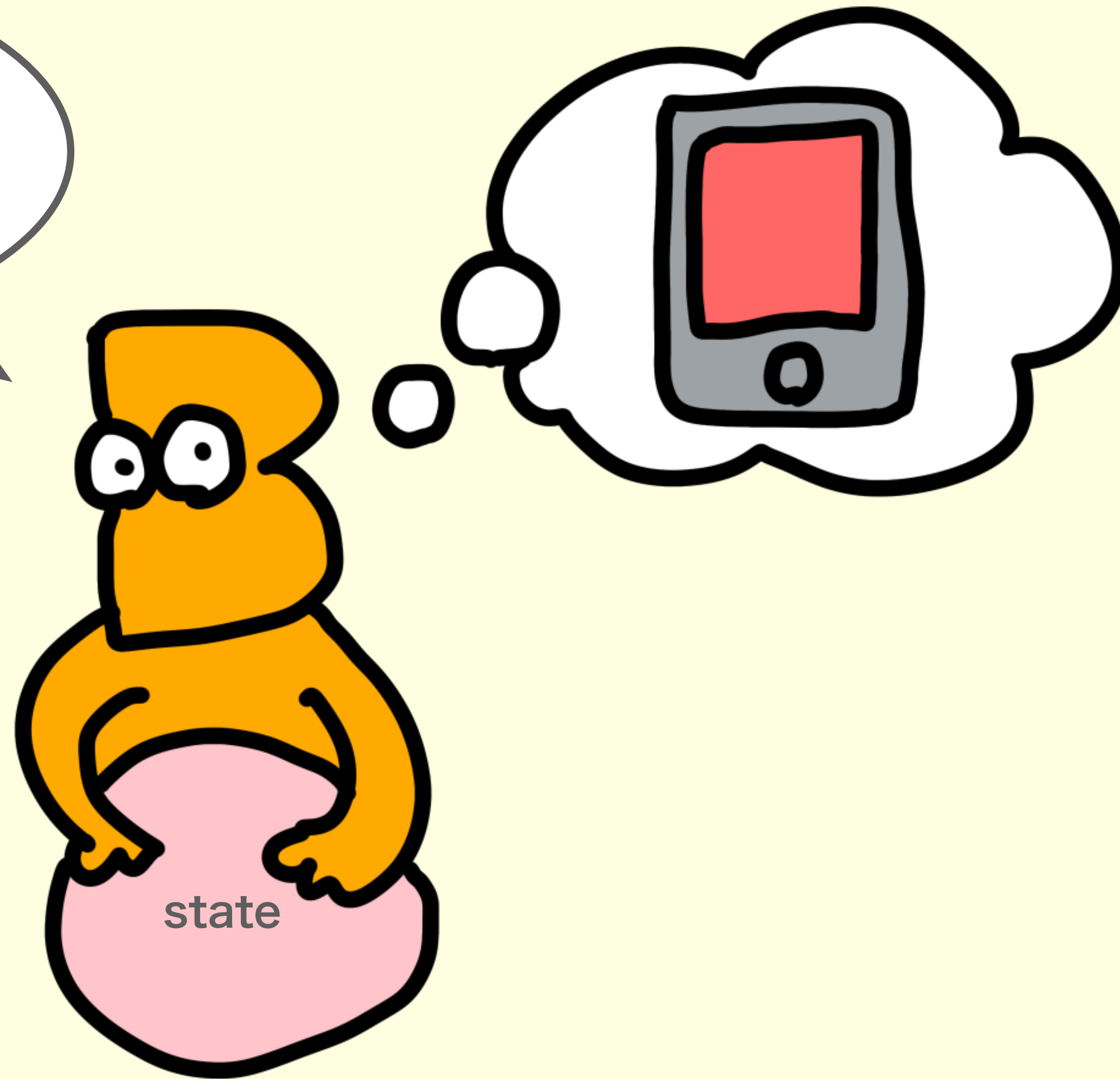


TeamChallenge! Bloc パターンに書き直してみよう

- Bloc パターンとは**状態管理に関するアーキテクチャパターン**の一つ
 - 2018 年の Google I/O で紹介された
 - ビジネスロジックと UI を分離して開発しやすくしよう！
 - <https://bloclibrary.dev/#/>
- Flutter 公式サンプル実装もある！
 - https://github.com/flutter/news_toolkit/tree/9c41161dce9991d5722c7f4e7bb4e28a7a8489a3
 - Flutter のベストプラクティスを踏襲
- 使用パッケージ
 - https://pub.dev/packages/flutter_bloc

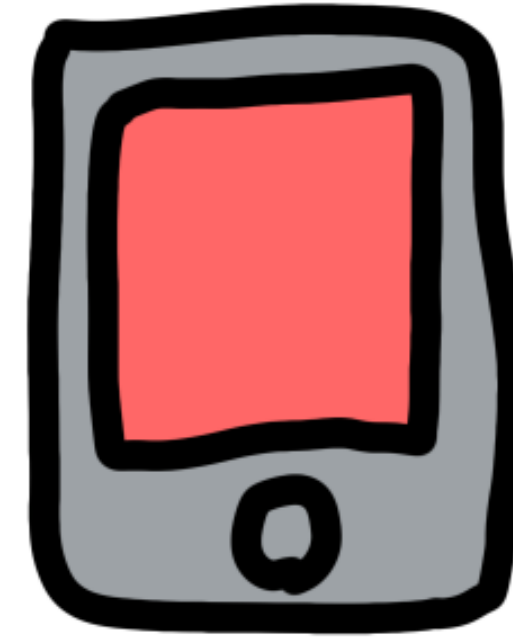
TeamChallenge! Bloc パターンに書き直してみよう

Bloc クラスは
常に state を持ってる



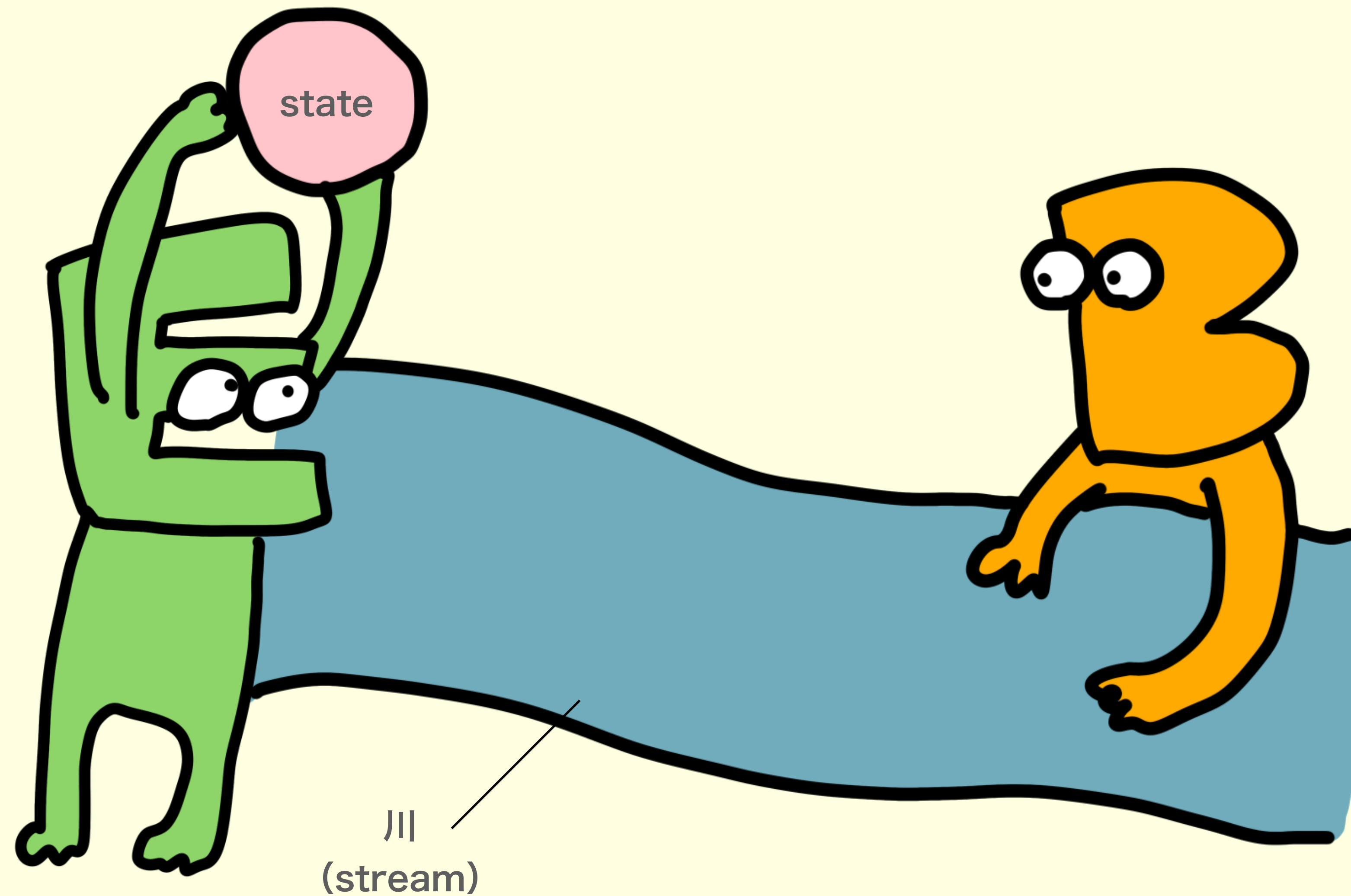
TeamChallenge! Bloc パターンに書き直してみよう

Bloc クラスは
常に state を持ってる



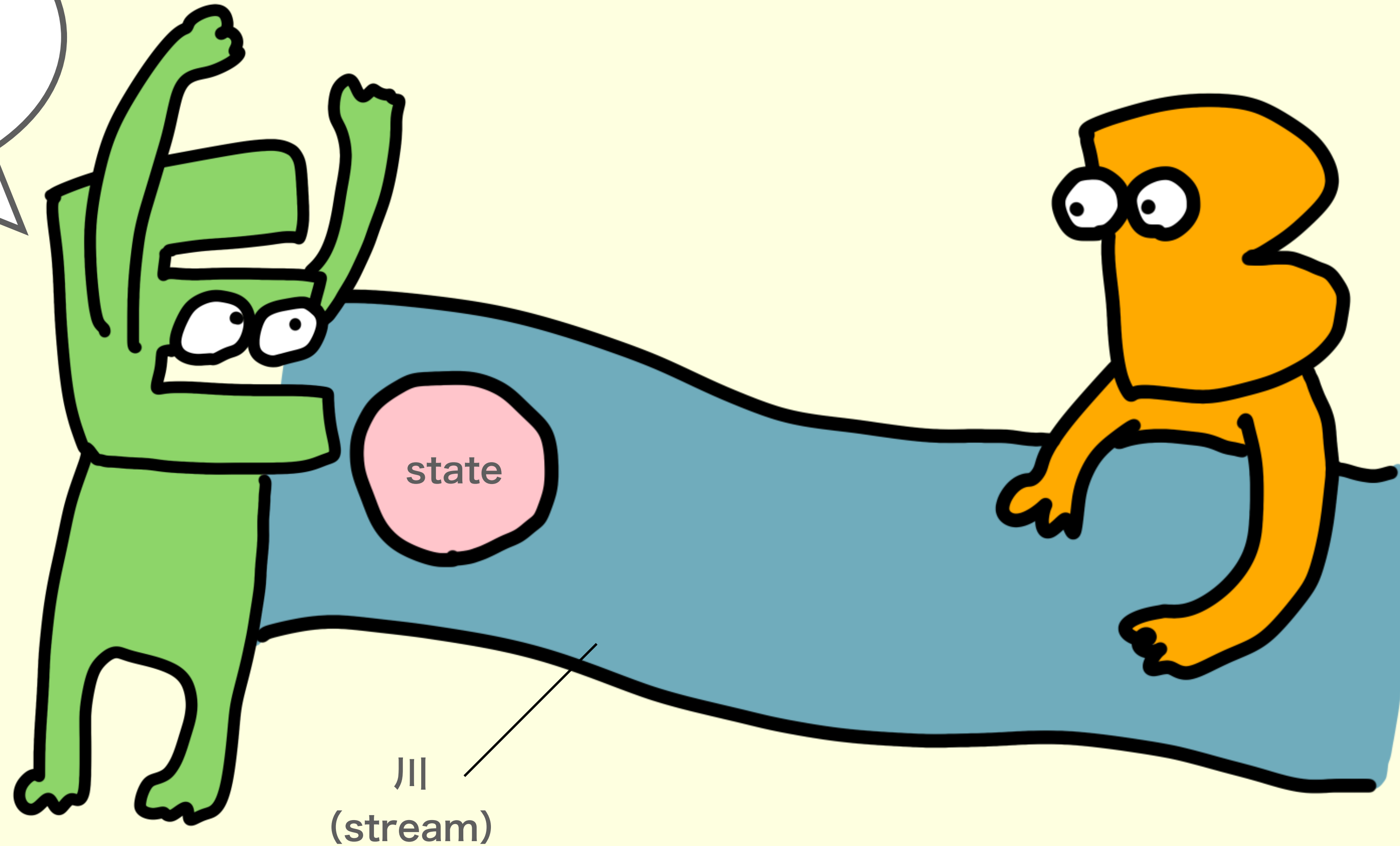
持ってる最新の state を見て
UI を構築する

TeamChallenge! Bloc パターンに書き直してみよう



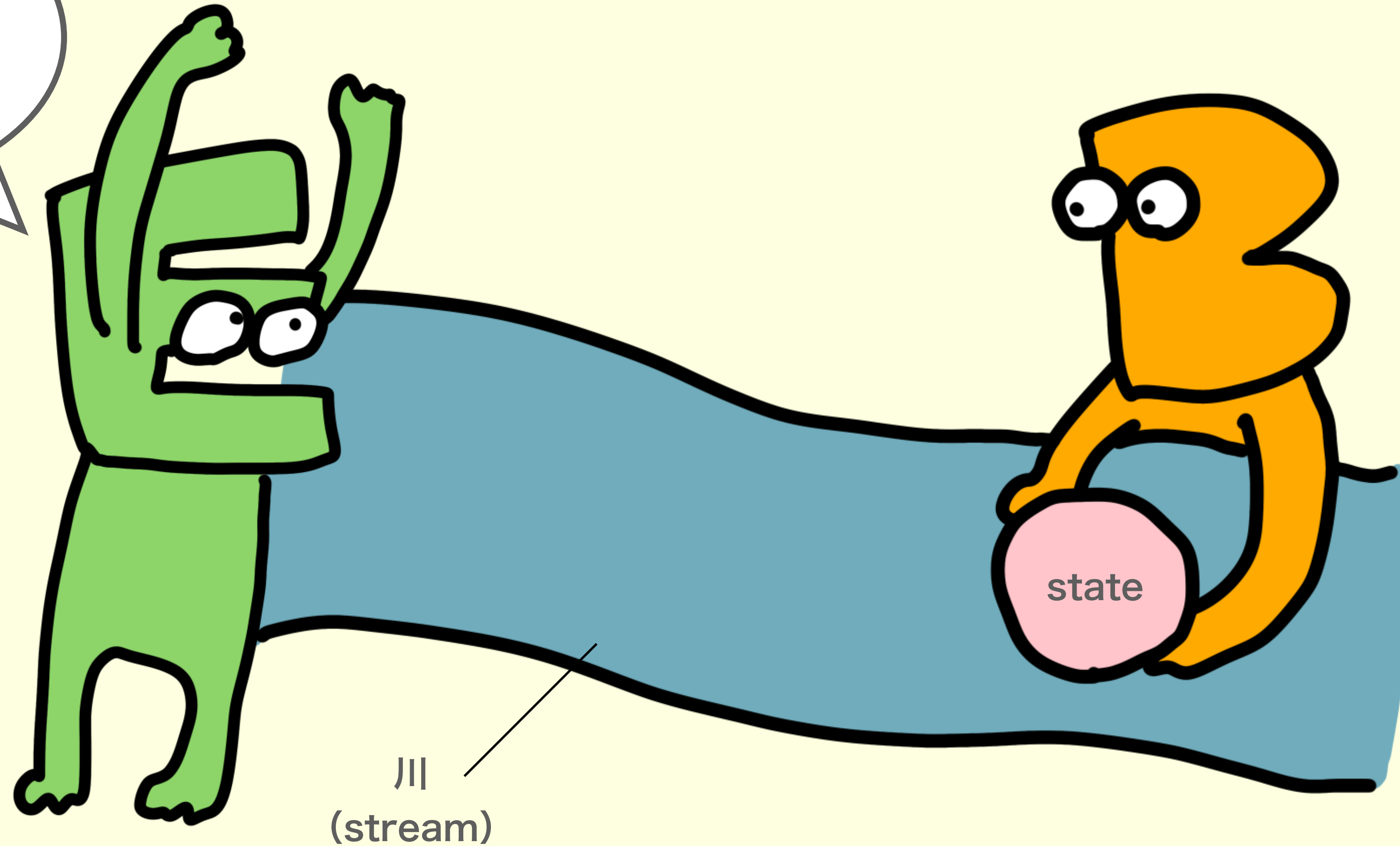
TeamChallenge! Bloc パターンに書き直してみよう

state を流す
(event)



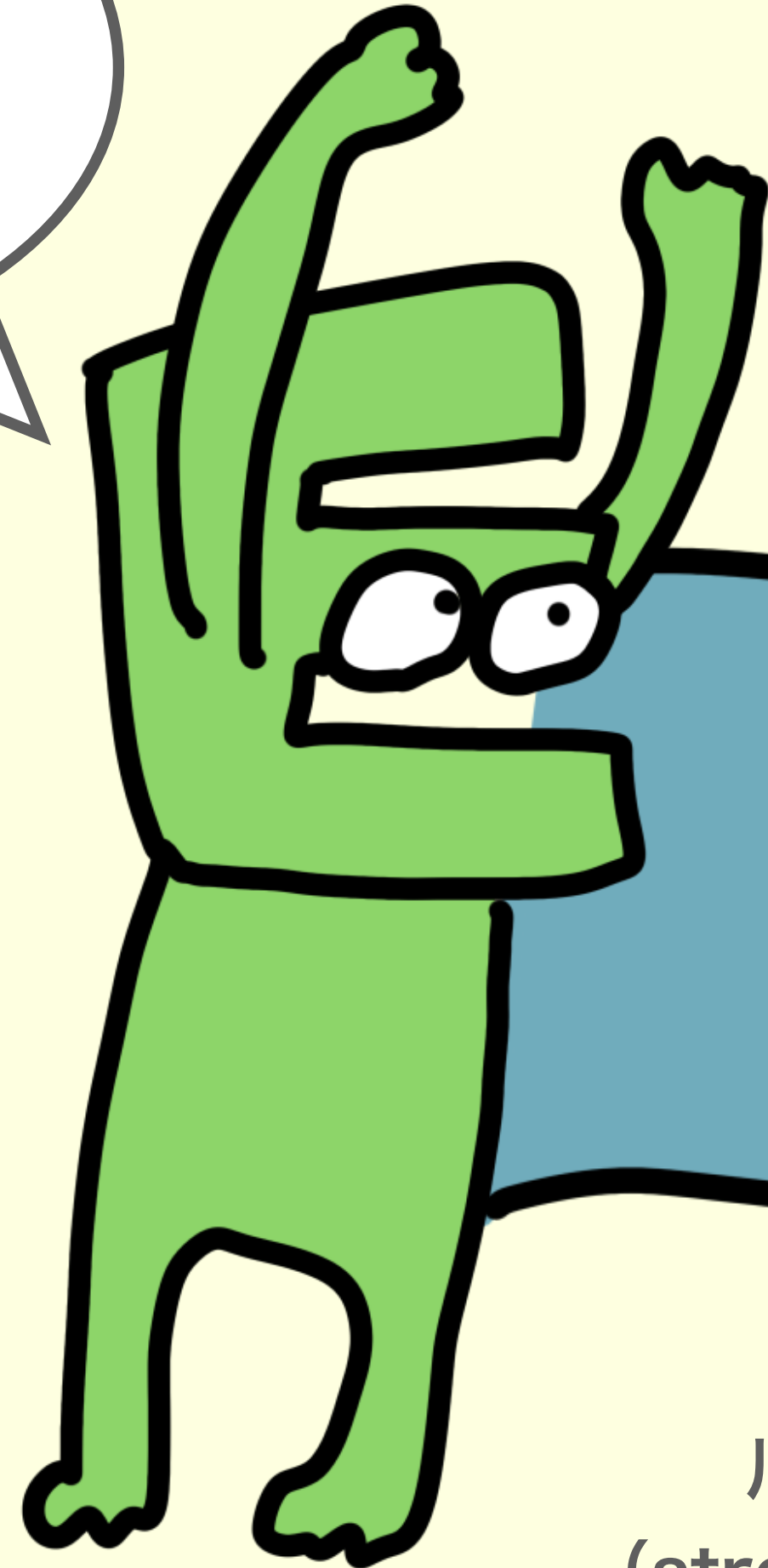
TeamChallenge! Bloc パターンに書き直してみよう

state を流す
(event)

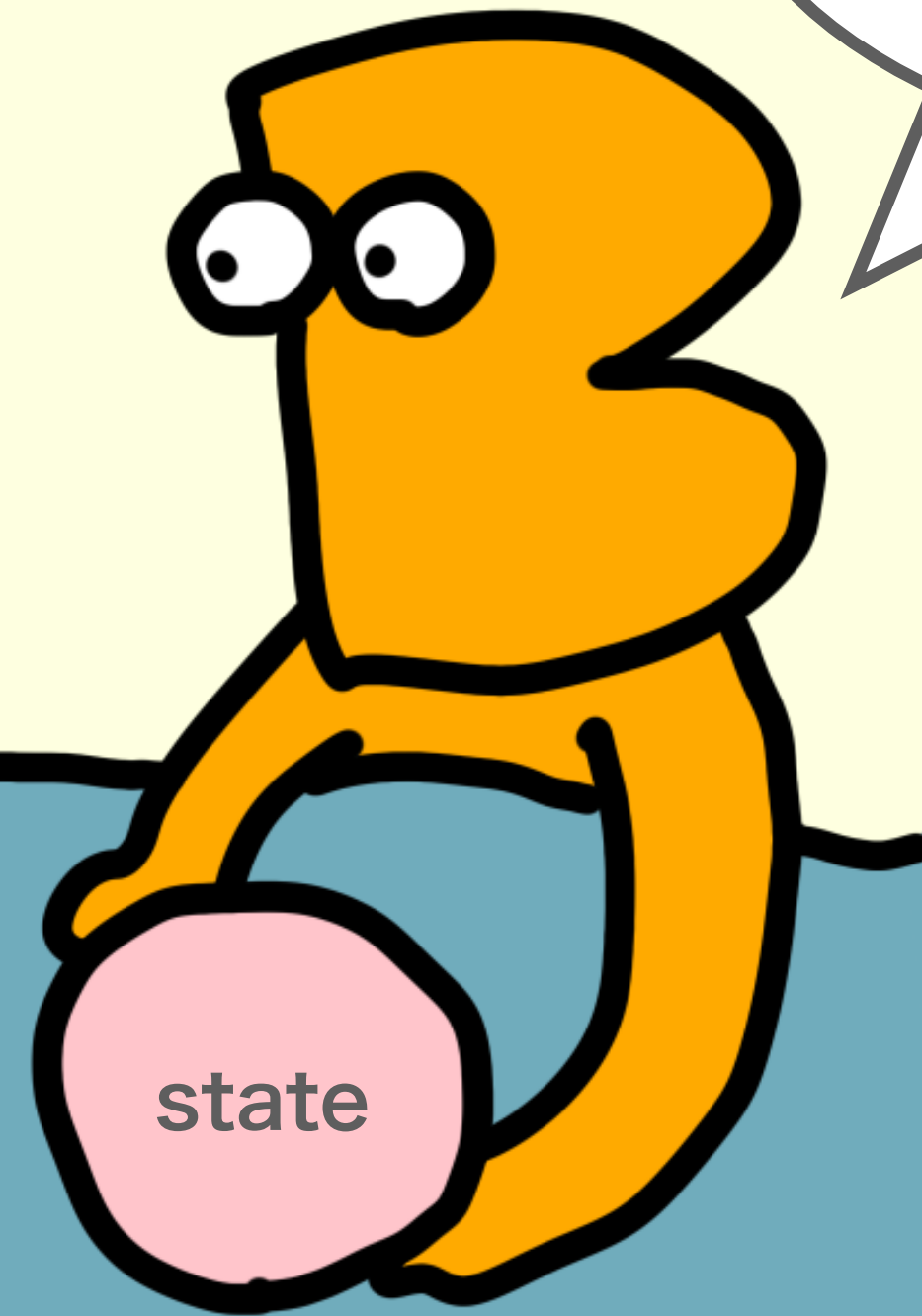


TeamChallenge! Bloc パターンに書き直してみよう

state を流す
(event)



流れてくるのを
待ち構えてる
(listen)



川
(stream)

TeamChallenge! Bloc パターンに書き直してみよう

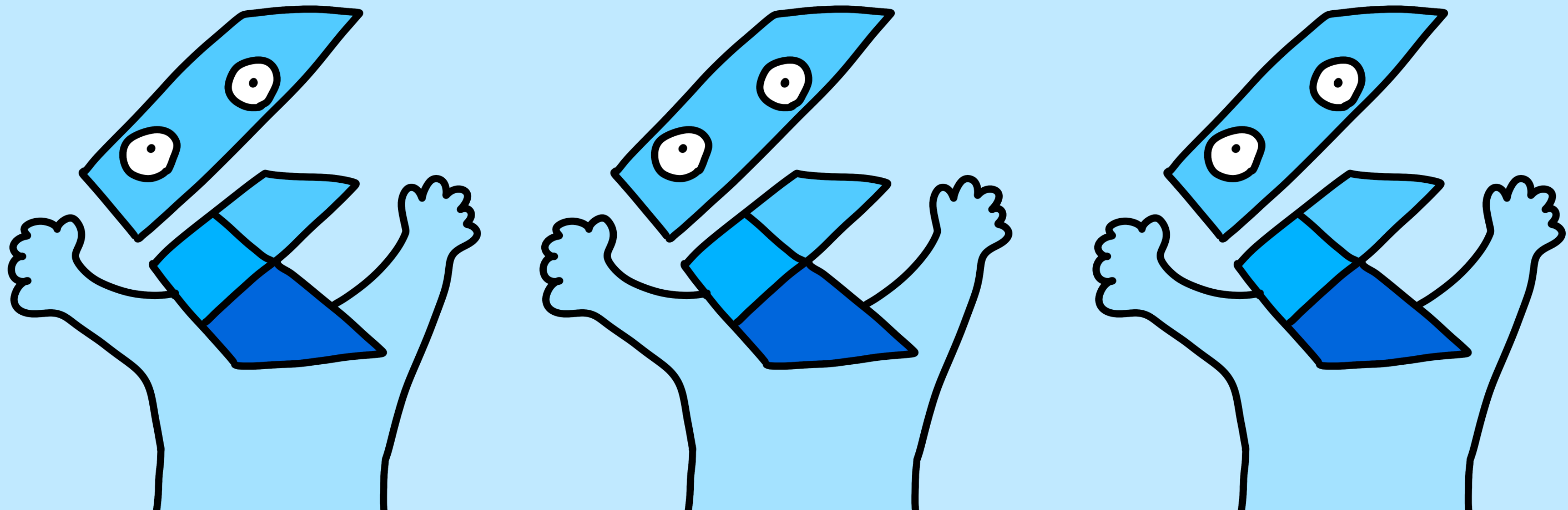
答え

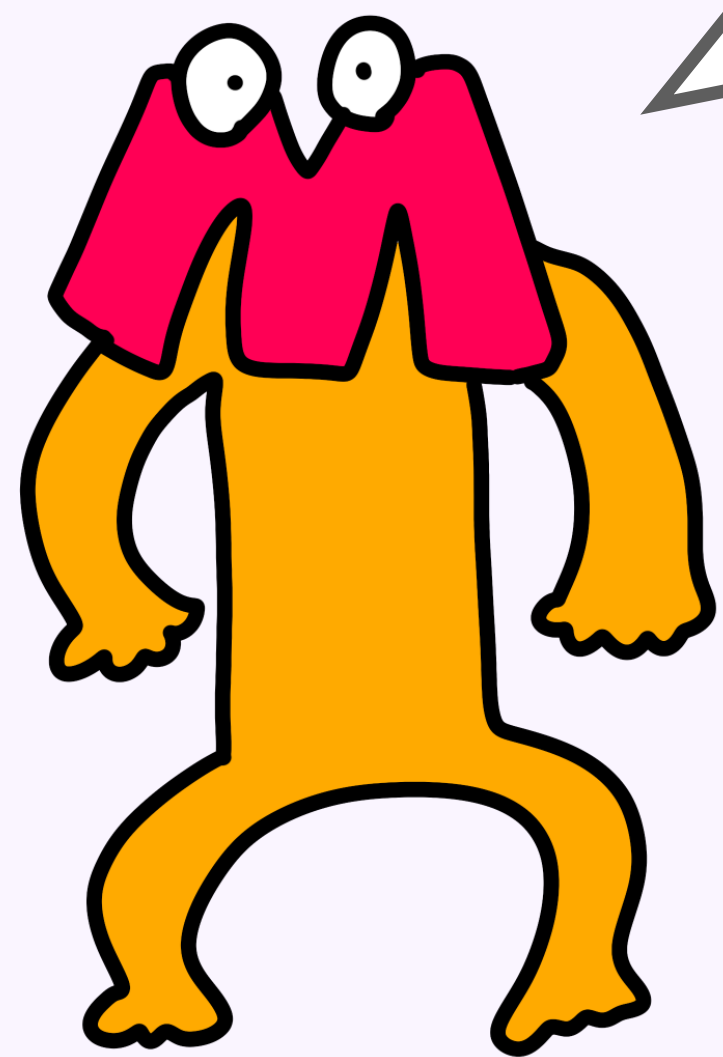
<https://github.com/mixigroup/2023BeginnerTrainingFlutter/pull/23>

クロージング

クロージング

Flutter 楽しめたかな？





以下補足資料

変数

【再代入できる】

var

【再代入できない】

final (**実行時定数**)

const (**コンパイル時定数**)

コンパイル時に定数にする方が**パフォーマンスが良い**ので const を使おう！

ウィジェットの切り出し方

```
class SampleState extends StatelessWidget {  
  final text1 = 'ほげりん';  
  final text2 = 'うなりん';  
  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      appBar: AppBar(  
        title: Text(widget.title),  
      ),  
      body: Center(  
        child: Column(  
          children: [  
            Text(text1),  
            Text(text2),  
          ],  
        ),  
      ),  
    );  
  }  
}
```

他のウィジェットでも
使いまわしたいので
切り出したい！

ウィジェットの切り出し方: helper method

```
class SampleState extends State<Sample> {  
  final text1 = 'ほげりん';  
  final text2 = 'うなりん';  
  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      appBar: AppBar(  
        title: Text(widget.title),  
      ),  
      body: Center(  
        child: Column(  
          children: [  
            Text(text1),  
            Text(text2),  
          ],  
        ),  
      ),  
    );  
  }  
}
```

簡単に切り出せる

```
class SampleState extends State<Sample> {  
  final text1 = 'ほげりん';  
  final text2 = 'うなりん';  
  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      appBar: AppBar(  
        title: Text(widget.title),  
      ),  
      body: showText(),  
    );  
  }  
  
  Widget showText() {  
    return Center(  
      child: Column(  
        children: [  
          Text(text1),  
          Text(text2),  
        ],  
      ),  
    );  
  }  
}
```


ウィジェットの切り出し方: クラス Widget

```
class SampleState
  final text1 =
  final text2 =

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(widget.title),
      ),
      body: Center(
        child: Column(
          children: [
            Text(text1),
            Text(text2),
          ],
        ),
      ),
    );
  }
}
```

クラスとして切り出すと
ちょっとめんどくさいけど
テストしやすい&
無駄な再描画を抑えられる！

```
class SampleState extends State<Sample> {
  final text1 = 'ほげりん';
  final text2 = 'うなりん';

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(widget.title),
      ),
      body: SampleText(
        text1: text1,
        text2: text2,
      ),
    );
  }
}

class SampleText extends StatelessWidget {
  final String text1;
  final String text2;

  const SampleText({
    super.key,
    required this.text1,
    required this.text2,
  });

  @override
  Widget build(BuildContext context) {
    return Center(
      child: Column(
        children: [
          Text(text1),
          Text(text2),
        ],
      ),
    );
  }
}
```

BuildContext ってなんだ？

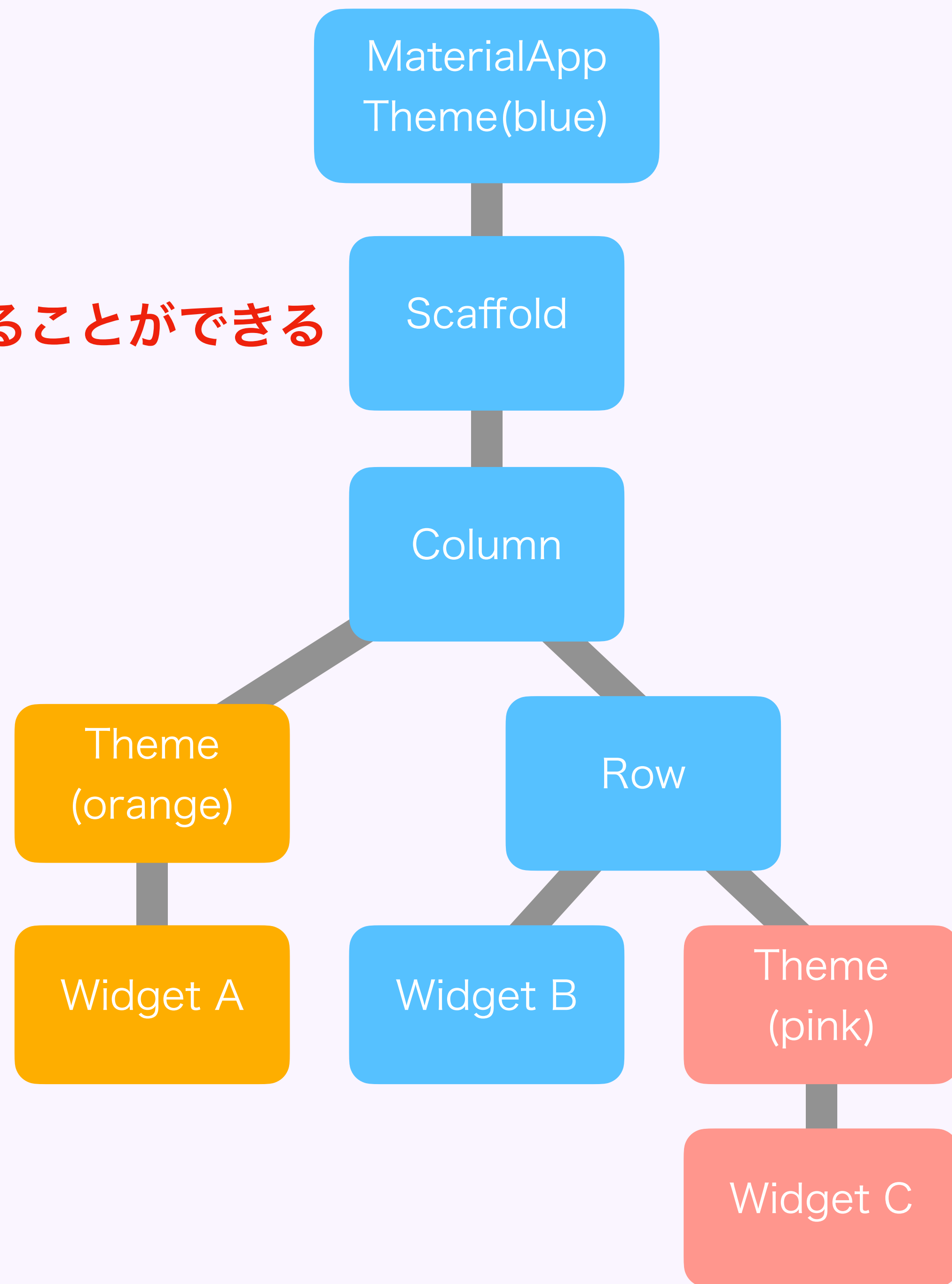
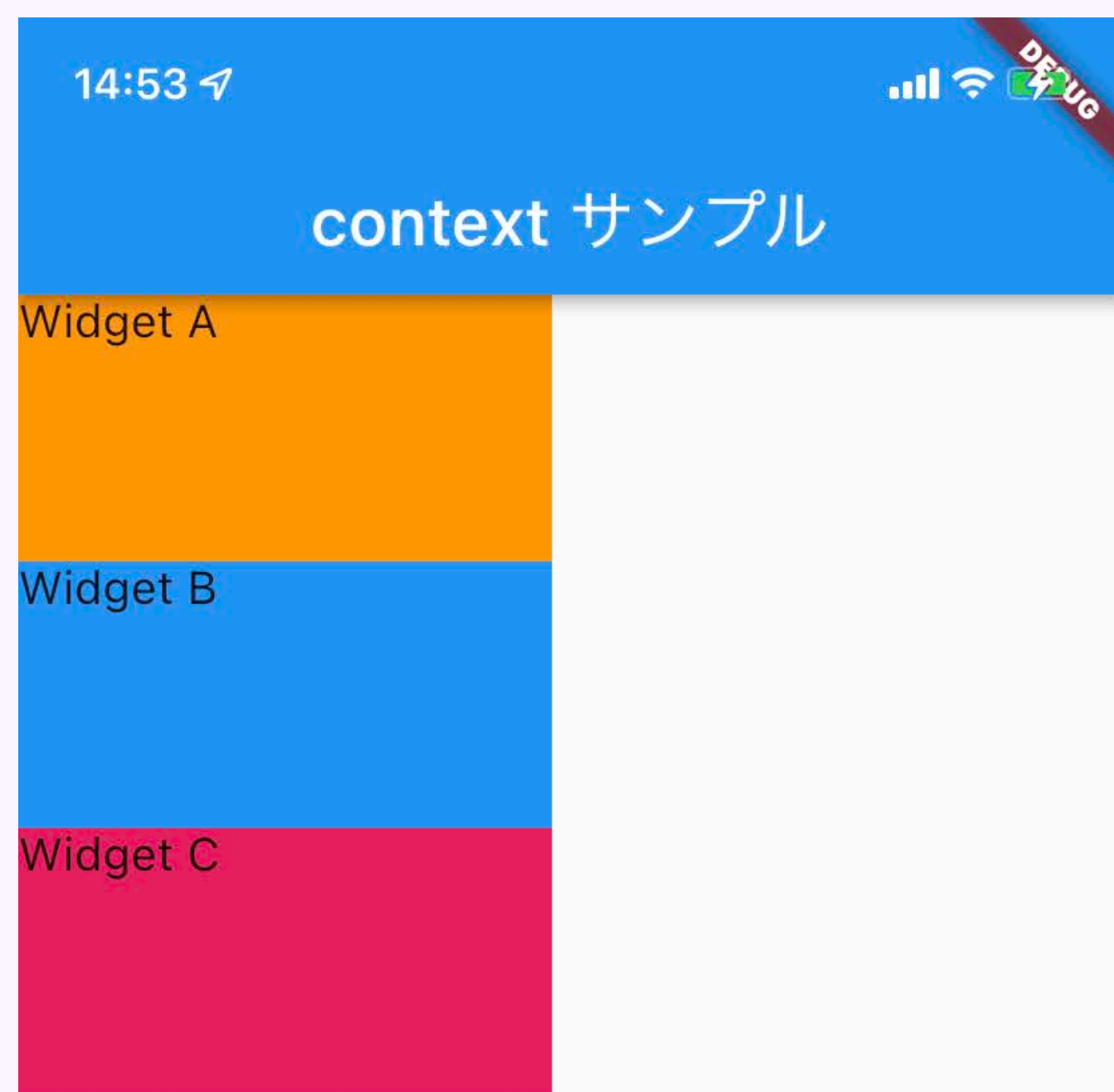
Widget は右図のようにツリー状になっている

BuildContext は血筋のことで**祖先の情報にアクセスすることができる**

※**直径の祖先**しか辿れない！

※1つの Widget インスタンスに対して

1つの BuildContext が**ペアの関係**になっている！



BuildContext ってなんだ？

Widget は右図のようにツリー状になっている

BuildContext は血筋のことで**祖先の情報にアクセスする**

※**直径の祖先**しか辿れない！

※1つの Widget インスタンスに対して

1つの BuildContext が**ペアの関係**になっている。

Widget B の血筋を辿ったら
この Theme が見つかった！



`Theme.of(context)`
は自分の血筋を辿って
祖先にある Theme を見つけてきている

