

Storybook の UI Testing Handbook を読んだ

2022-01-19 フロントエンドLT会 - vol.5 #frontendlit

自己紹介



Shingo Yamazaki

- 株式会社ログラス
- 昨年9月から University of the People で
コンピューターサイエンス勉強中



zaki-yama



zaki__yama

はじめに

フロントエンドのテストに対する悩み



- フロントエンドのテスト、何をどこまで書いたらいいものか...
- どういう観点でテストを書いたらいいのか
 - 最近はVisual Regression Testingとかよく聞く。それ以外は...?
- 比較的変更が生じやすいUIのテストをどこまで書くべきか
- どういうツール／ライブラリを使うのがいいのか
- etc.

Dominic Nguyen
@domyen

...

I've been helping out on a new book that's finally here!



UI Testing Handbook

9 chapters about how to actually test UIs w code samples, free

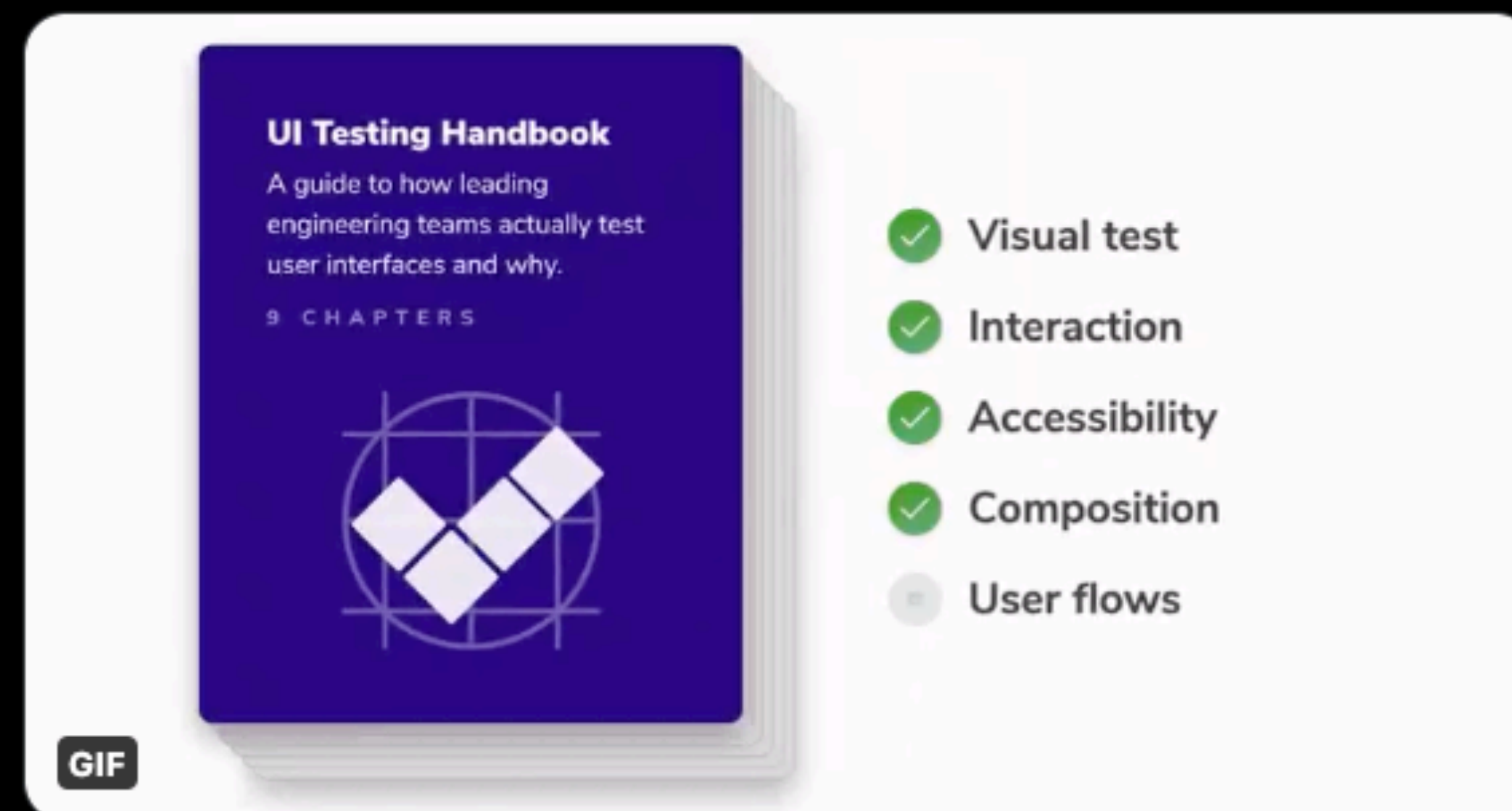
Uh, what?

10 months of research from real teams distilled into 1 pragmatic guide

big ups to [@winkerVSbecks](#)

storybook.js.org/tutorials/ui-t...

ツイートを翻訳



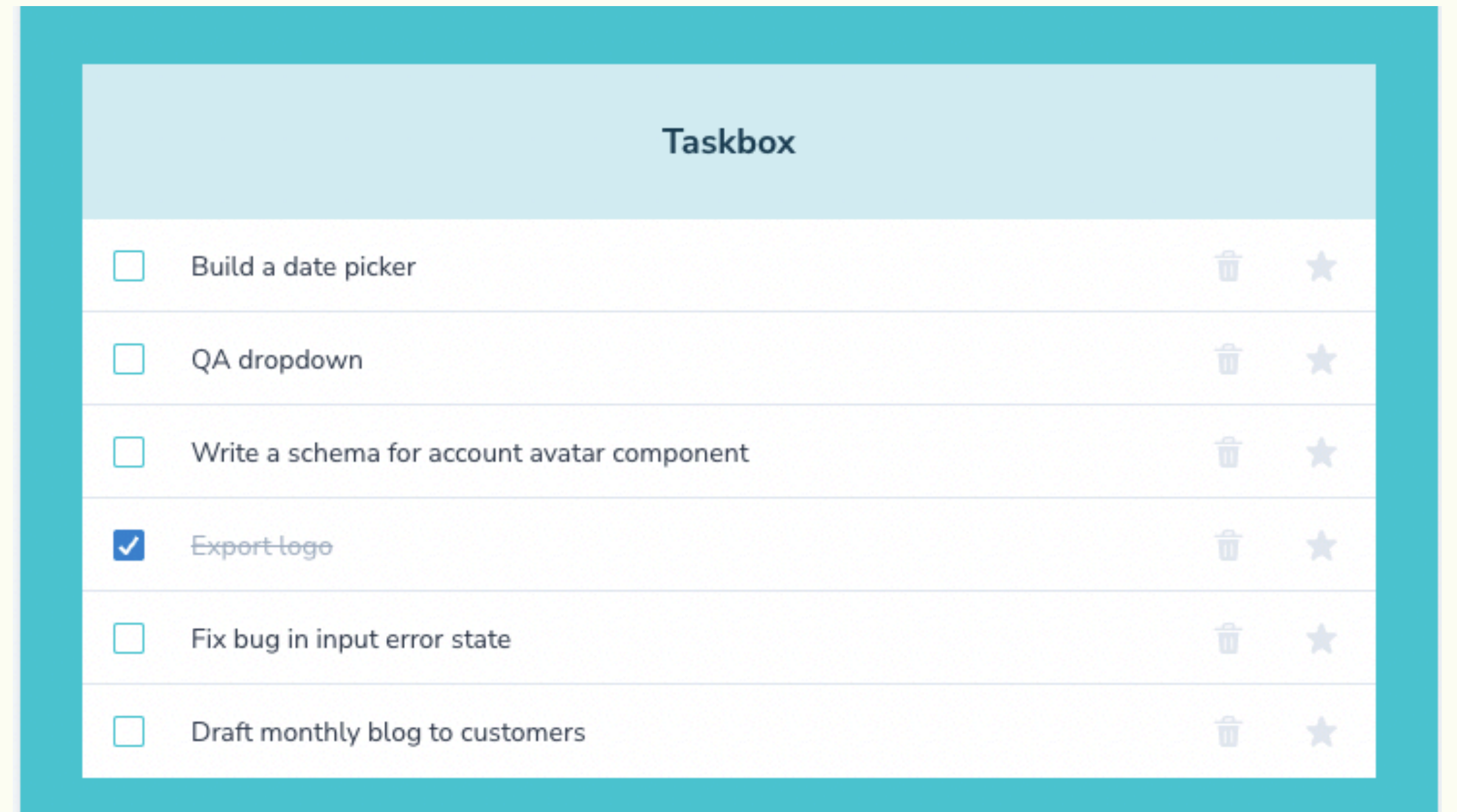
UI Testing Handbook

- Storybook が提供しているチュートリアルコンテンツのひとつ
- 昨年12月に公開された
- Twilio, Adobe, Shopify など、Storybookコミュニティの10個のチームをリサーチして得られた知見をまとめたもの（"Introduction" より）

**Handbook の内容を
ざっくり紹介します**

サンプルアプリケーション

- よくあるTodoアプリ
- アプリケーション側のコードは
ほぼいじらず、
Storybookやテストを書きながら
学んでいく構成



Introduction

- 現在の主要なJavaScriptフレームワークはいずれもコンポーネントドリブン
- Unit, Integration, E2E といった分類ではなく、UIが持つ特徴(characteristics)にフォーカス
しましょう

テストすべきUIの特徴

- Visual
 - 見た目
- Interaction
 - クリックやユーザー入力などのイベントが適切にハンドリングされるか
- Accessibility
 - アクセシビリティ
- User flow
 - 複数のコンポーネントにまたがって完結するユーザー操作が期待通り行えるか

それぞれを実現するためのツール／ライブラリ

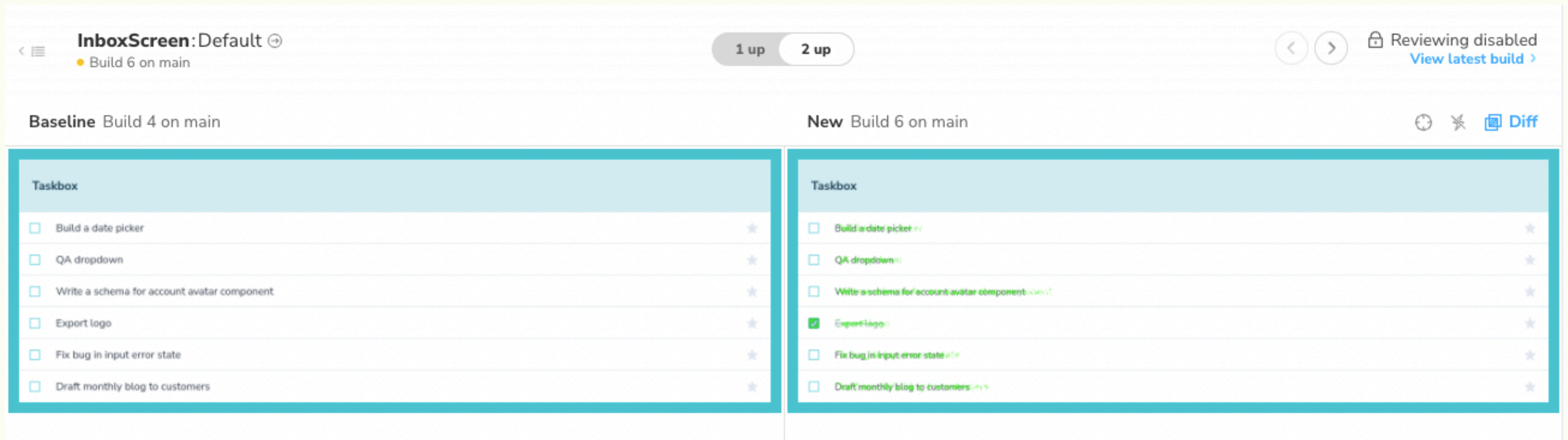
- Visual
 - 見た目 ... **Chromatic**
- Interaction ... **Jest & Testing Library**
 - クリックやユーザー入力などのイベントが適切にハンドリングされるか
- Accessibility
 - アクセシビリティ ... **Axe**
- User flow ... **Cypress (or Playwright, Selenium)**
 - 複数のコンポーネントにまたがって完結するユーザー操作が期待通り行えるか

Visual

- Chromatic を使った Visual Regression Testing の紹介
- Storybook の内容を画像でキャプチャし、コミット間で差分がないかチェックする

Visual: Chromatic

- Visual Regression Testingを簡単に導入できるサービス
 - 自前でやるとreg-suit + AWS S3などのストレージサービスで構築



Interaction

- Jest & Testing Library (@testing-library/react)
- Storybook の Story をテストケースにも再利用しましょう

Interaction

```
import { render, waitFor, cleanup, within, fireEvent, } from "@testing-library/react";
import { composeStories } from "@storybook/testing-react";
import * as stories from "../InboxScreen.stories";

describe("InboxScreen", () => {

  const { Default } = composeStories(stories);

  it("should pin a task", async () => {
    const { queryByText, getByRole } = render(<Default />);

    await waitFor(() => {
      expect(queryByText("You have no tasks")).not.toBeInTheDocument();
    });

    const getTask = () => getByRole("listitem", { name: "Export logo" });

    const pinButton = within(getTask()).getByRole("button", { name: "pin" });

    fireEvent.click(pinButton);

    const unpinButton = within(getTask()).getByRole("button", {
      name: "unpin",
    });

    expect(unpinButton).toBeInTheDocument();
  });
});
```

Interaction

Storyをインポートし、
そのままテストケースとして利用する

```
import { render, waitFor, cleanup, within, fireEvent, } from "@testing-library/react";
import { composeStories } from "@storybook/testing-react";
import * as stories from "../InboxScreen.stories";

describe("InboxScreen", () => {

  const { Default } = composeStories(stories);

  it("should pin a task", async () => {
    const { queryByText, getByRole } = render(<Default />);

    await waitFor(() => {
      expect(queryByText("You have no tasks")).not.toBeInTheDocument();
    });

    const getTask = () => getByRole("listitem", { name: "Export logo" });

    const pinButton = within(getTask()).getByRole("button", { name: "pin" });

    fireEvent.click(pinButton);

    const unpinButton = within(getTask()).getByRole("button", {
      name: "unpin",
    });

    expect(unpinButton).toBeInTheDocument();
  });
});
```

Accessibility

- Axe というライブラリを使うとアクセシビリティをある程度機械的にチェックできる
- さらにStorybook用のアドオンやJestでAxeを使ったチェックを走らせるための jest-axe というライブラリがある

Accessibility: storybook-addon-a11y

The screenshot shows the Storybook interface with a 'Taskbox' component. The task list includes:

- ☐ Build a date picker
- ☐ QA dropdown
- ☐ Write a schema for account avatar component
- ☒ Export logo (highlighted with a red dashed box)
- ☐ Fix bug in input error state
- ☐ Draft monthly blog to customers

At the bottom, the 'Accessibility' tab is active, showing the following results:

Controls (1)	Actions	Accessibility
1 Violations	18 Passes	0 Incomplete
Highlight results ☒		
> Elements must have sufficient color contrast ☒		

A red arrow points from the text box on the right to the 'Accessibility' tab results.

アクセシビリティ違反箇所が
Storybook上で確認できる
(☒入れると該当箇所がハイライト)

Accessibility: jest-axe

```
import { axe, toHaveNoViolations } from "jest-axe";
import { composeStories } from "@storybook/testing-react";
import * as stories from "../InboxScreen.stories";

expect.extend(toHaveNoViolations);

describe("InboxScreen", () => {
  ...
  const { Default } = composeStories(stories);

  // Run axe
  it("should have no accessibility violations", async () => {
    const { container, queryByText } = render(<Default />);

    await waitFor(() => {
      expect(queryByText("You have no tasks")).not.toBeInTheDocument();
    });

    const results = await axe(container);
    expect(results).toHaveNoViolations();
  });
});
```

toHaveNoViolations() を呼び出すだけ

User flow

- いわゆるE2Eテスト
- 2つの選択肢が考えられる
 1. バックエンドまで含めた完全なテスト環境を使う（O'Reillyチームはこっち）
 2. バックエンドはモックし、フロントエンドのみ（Twilioチームはこっち）
- チュートリアルでは後者。Cypress を使う
 - “ログイン画面で正しいユーザー名&パスワードを入力したら、タスク一覧が開ける” というテストケース
- ここでも Story をテストケースとして再利用

まとめ

こういったことが学べてよかった😊

- 各種テスト観点とそれを実現するための具体的なライブラリ
 - Chromatic, Axe, Cypress, etc.
- Storybookを中心に置いた実用的なワークフロー
 - StoryをJestやCypressのテストにも再利用する、という考え方
 - GitHub ActionsによるCIについても紹介されている（“Automate” の章）
- 実際にコード書いて動くものを見ながら学べたのも個人的にはよかった

引き続きわからん😭

- それぞれのテストをどういう基準で組み合わせて、アプリケーション全体としての品質を担保するか
- 特に Interaction と User flow (E2E) の棲み分け
- “Given this trade-off, most teams use a hybrid approach to balance effort and value. E2E tests are limited to only critical user flows and interaction tests are used to verify all other behavior.”
- バックエンドをモックしたE2Eテストはなんとなく中途半端なのでは...という感じもする

ご清聴ありがとうございました

写経したコードは

<https://github.com/zaki-yama-labs/ui-testing-handbook>
にあります