60分で学ぶE2Eテスト(テスト実装編)

自己紹介:末村拓也

- Test Automation Specialist @ Autify, Inc.
- JaSST Online 実行委員



今日やること

- テストツール Cypress を使った、E2Eテスト実装の流れを紹介
- 保守性が高く読みやすいコードの書き方

コードを書くところにフォーカスします

今日お話しできないこと

- 自動化の技術選定をどのように行うか
- 自動化やプログラミングに必要な基礎知識の説明
 - JavaScriptの文法
 - 。 コマンドラインの使い方
- CI/CDなど、開発サイクルの中で自動テストを活かす方法



テストに使うツール

Cypress

デベロッパーフレンドリーなE2Eテストツール

- NodeJSで動作する(=JavaScriptで記述する)
- Chrome/Firefoxに対応
- テストコードの作成やデバッグを楽にする機能がいろいろある

NodeJSのインストール

公式サイトからダウンロードしてください

https://nodejs.org/ja/

または、Macで brew コマンドが使える人はこちらでもOK

\$ brew install node

Cypressのインストール

コマンドラインで以下を実行

\$ mkdir jasst22tokyo
\$ cd jasst22tokyo
\$ npm init -y
\$ npm install cypress

初回起動時に設定ファイルとサンプルのテストコードが生成されます



\$ npx cypress open

/Users/takuyasuemura/ghq/github.com/tsuemura/jasst22-tokyo/samples		
samples	🚱 Support 🛛 🚘 Docs 🛛 🚨 Log In	
Image: Setting state of the set	🌔 Chrome 97 🗸	
Q Press Cmd + F to search	ブラウザ選択 ^{New Spec File}	
▼ INTEGRATION TESTS COLLAPSE ALL EXPAND ALL	► Run 20 integration specs	
 I-getting-started todo.spec.js 		
 > 2-advanced-examples 		
🗅 actions.spec.js	サンプルテスト	
🗅 aliasing.spec.js		
assertions.spec.js		
Connectors.spec.js		
🗅 cookies.spec.js		
🗅 cypress_api.spec.js		
🗅 files.spec.js		
Iocal_storage.spec.js		
	Version 9.2.0 Changelog	



Utilities Cypress API

Commands 🗸

GitHub

Window

Ð

cypress.io

Examples of referencing window and other properties on window in Cypress, for a full reference of commands, go to docs.cypress.io

cy.window()

To get the global window object, use the cy.window() command.

cy.window().should('have.property', 'top')

cy.document()

To get the document object, use the cy.document() command.

cy.document().should('have.property', 'charset').and('eq', 'UTF-8')

cv.title()



テストを実行すると実行結果が細かく表示されます

テストケース

1. 非会員で予約

2. 会員登録→予約→ログアウト

- 3. プレミアム会員でログイン→予約→ログアウト
- 4. 一般会員でログイン→予約→ログアウト

5. 一般会員の画面にプレミアム会員限定プランが表示されないこと

6. 非会員の画面に一般・プレミアム会員限定プランが表示されないこと

非会員で予約するシナリオの手順(1/2)

- 1. https://hotel.testplanisphere.dev/ja/ を開く
- 2. メニューから「宿泊予約」を選択
- 3. 宿泊プラン一覧から「お得な特典付きプラン」の「このプランで予約」を選 択
- 4. 宿泊日を翌月1日に設定
- 5. 宿泊数を7泊に設定
- 6. 人数を2に設定
- 7. 朝食バイキング、昼からチェックインプラン、お得な観光プランを選択8. 氏名に「テスト太郎」を入力

非会員で予約するシナリオの手順(2/2)

- 9. 確認のご連絡をメールに設定
- 10. メールアドレスにhoge@example.comを設定
- 11. ご要望・ご連絡事項に「テスト」と入力
- 12. 予約内容を確認するボタンを選択
- 13. 宿泊予約確認画面で、以下を確認
 - i. 合計金額が121,000円であること
 - ii. 期間、人数、追加プラン、お名前、確認のご連絡、ご要望・ご連絡が 入力通りになっていること
- 14. この内容で予約するボタンを選択し、以下を確認

i. 予約が完了しましたダイアログが表示されること

cypress/integration/smoke_test.js を作成

```
describe('スモークテスト', () => {
```

```
it('非会員で予約', () => {
```

```
// ここにテストコードを書く
```

```
})
```

})

describe ~ it は「何をテストするのか」を書く部分

設計したテスト手順をそのままコメントとして書いちゃえ

describe('スモークテスト', () => {

it('非会員で予約', () => {

- // 1. https://hotel.testplanisphere.dev/ja/ を開く
- // 2. メニューから「宿泊予約」を選択
- // 3. 宿泊プラン一覧から「お得な特典付きプラン」の「このプランで予約」を選択
- // 4. 宿泊日を翌月1日に設定
- // 5. 宿泊数を7泊に設定
- // 6. 人数を2に設定

})

})

- // 7. 朝食バイキング、昼からチェックインプラン、お得な観光プランを選択
- //8.氏名に「テスト太郎」を入力
- // 9. 確認のご連絡をメールに設定
- // 10. メールアドレスにhoge@example.comを設定
- // 11. ご要望・ご連絡事項に「テスト」と入力
- // 12. 予約内容を確認するボタンを選択
- // 13. 宿泊予約確認画面で、以下を確認
- // 1. 合計金額が123,000円であること
- // 2. 期間、人数、追加プラン、お名前、確認のご連絡、ご要望・ご連絡が入力通りになっていること
 // 14. この内容で予約するボタンを選択し、以下を確認
- // 1. 予約が完了しましたダイアログが表示されること

テスト対象のサイトにアクセス

```
describe('スモークテスト', () => {
it('非会員で予約', () => {
```

// 1. https://hotel.testplanisphere.dev/ja/を開く cy.visit("https://hotel.testplanisphere.dev/ja/index.html");

}) })

- コマンドは(一部の例外を除き) cy から始まる
- cy.visit() は指定したURLに移動するコマンド

HOTEL PLANISPHERE

ホーム	宿泊予約	会員登録	

```
describe('スモークテスト', () => {
    it('非会員で予約', () => {
        // テスト対象のサイトにアクセス
        cy.visit("https://hotel.testplanisphere.dev/ja/index.html");
        // 2. メニューから「宿泊予約」を選択 ←イマココ
        cy.=====.click()
    })
})
```

クリックは click() でOK 宿泊予約、というリンクを どうやって指定する?

HOTEL PLANISPHERE

ホーム	宿泊予約	会員登録	ログイン

Cypressでは contains() を使って 特定の文字を含む要素を指定できる

- `宿泊予約` をクリック

 \downarrow

cy.contains('宿泊予約').click()

現在のテストコード

```
describe('スモークテスト', () => {
it('非会員で予約', () => {
```

```
// テスト対象のサイトにアクセス
```

cy.visit("https://hotel.testplanisphere.dev/ja/index.html");

```
// 2. メニューから「宿泊予約」を選択
cy.contain('宿泊予約').click()
})
```

自動化は難しくない

テスト手順をそのまま1:1対応でプログラミングすれば、それがテストコード

"https://hotel.testplanisphere.dev/ja/index.html" にアクセスする

cy.visit("https://hotel.testplanisphere.dev/ja/index.html");

"宿泊予約"をクリックする

 \downarrow

cy.contains('宿泊予約').click()

実際に動かしてみよう

コマンドラインから以下を実行する

\$ npx cypress open



Version 9.2.0 Changelog

smoke_test.jsをクリック

•	Ø	samples
---	---	---------

→ C A Not Secure | https://hotel.testplanisphere.dev/__/#/tests/integration/smoke_test.js

 \sim

Chrome is being controlled by automated test software.		×
C Tests ✓1 X () 00.28 •‡ C	https://hotel.testplanisphere.dev/ja/index.html 1000 x 660 (100%)) (
cypress/integration/smoke_test.js		
 スモークテスト 会員登録して予約してログアウト TEST BODY 1 visit https://hotel.testplanisphere.dev/ja/index 	HOTEL PLANISPHERE	
	ホーム 宿泊予約 会員登録 ログイン	
	このサイトはテスト自動化の学習用の練習サイトです。 Seleniumなどのブラウザテスト自動化を学習したい方が、実際にテストスクリプトを実行するためのテスト対象サイトとして 作成されています。 書籍やブログなどでのサンプルやデモにもお使いいただけます。ライセンスはMIT Licenseです。 自動テストの学習を目的として作成されていますが、テスト設計や技法の学習に使うことも可能です。	
	サイトの構成 ホテルの予約サイトを模した作りになっています。ログイン・会員登録・ホテルの宿泊予約のそれぞれの入力フォームを用意 しています。レスポンシブデザインに対応しているためモバイルブラウザでも表示できます。	

ブラウザが開いて、URLに遷移できた

続けて書いていきましょう

宿泊プランの選択

describe('スモークテスト', () => { it('非会員で予約', () => {

// テスト対象のサイトにアクセス cy.visit("https://hotel.testplanisphere.dev/ja/index.html");

// 2. メニューから「宿泊予約」を選択 cy.contain('宿泊予約').click()

// 3. 宿泊プラン一覧から「お得な特典付きプラン」の「このプランで予約」を選択 ←イマココ

}) })

宿泊プランの選択

HOTEL PLANISPHERE

宿泊プラン-	一覧			
	ត្រូវ	すめプラン☆		
	お得な物	寺典付きプラン		
	大	(1名7,000円) 名様から		
	スタン	/ダードツイン		
	20	ノフノで予約		
		本日限り		
素泊まり	出張ビ	ジネスプラン	エステ・マッサー	ジプラン
大人1名5,500円 1名様から	5/	<1名7,500円 1名様から	大人1名9,000 1名様から	円
シングル		シングル	部屋指定なし	_
このプランで予約	20	フランで予約	このプランでき	F#9
貸し切り露天風呂プラン	カップ	ル限定プラン	テーマパーク優待	寺プラン
大人1名9,000円	×/	1名8,000円	大人1名10,000	円
14年205	プレ	2台球がら ミアムツイン	1石像から 部屋指定なし	,
	7.0	プランで予約	このプランでき	F-80

複数の宿泊プランから 「お得な特典付きプラン」を選択した い

試しに書いてみよう

お得な特典付きプラン を含む 宿泊プラン の このプランを選択 をクリックする

cy.contains('お得な特典付きプラン').contains('このプランで予約').click()

このコードで動くかな.....? 🤔

目当ての要素が見つからない

5 - contains このプランで予約 0
AssertionError
Timed out retrying after 4000ms: Expected to find content: 'このプランで予約' within the element: <h5.card- title> but never did.</h5.card-
cypress/integration/smoke_test.js:10:25
8 cy.contains('宿泊予約').click() 9
> 10 cy.contains('素泊まり').contains('このプラ:
^ 11 12 13 // // 全昌登録
和 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
View stack trace >- Print to console

cy.contains('お得な特典付きプラン') が h5 要素にマッチしてしまったのが 回田



ページの構造を見てみよう

テスト結果の画面でそのまま開発者コンソールを開けます 右クリック→Inspect

https://hotel.testplanisphere.dev/ja/plans.html	1000 x 660 (62%) Elements Console > 3
div.card-body 688 × 202 Color ■#212529 Font 16px -apple-system, "system-ui", "Seg Padding 20px ACCESSIBILITY Name Role generic Keyboard-focusable	マ <body class="bg-light"> <h1 class="text-center text-uppercase d-n
one d-lg-block display-3 my-5">Hotel Planisphere</h1> Planisphere > <nav class="navbar navbar-expand-lg navba
r-dark bg-dark py-lg-4"> /nav> flex v<div class="container bg-white py-5"> > <div class="row"></div> flex v<div class="row"></div> flex v<div class="row"> (div class="row"> v<div class="cont_lg-9 mx-auto py-5"> v<div class="cont_lg-9 mx-auto py-5"></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></nav></body>
お得な特典付きプラン 大人1名7,000円 1名様から スタンダードツイン このプランで予約 本日限り	めプラン☆
	Filter :hov .cls + [4]

探索の範囲を絞り込む

div.card-body Color Font 16px -apple-system, "sys Padding	688×202 ■#212529 tem-ui", "Seg 20px	ログイン
ACCESSIBILITY		
Name		
Role	generic	
Keyboard-focusable	\otimes	
お得な 大. スタ. この	特典付きプラン 人1名7,000円 1名様から ンダードツイン 0フランで予 約	
	本日限り	

- お得な特典付きプラン を含む
- 宿泊プラン の
- このプランを選択をクリックする

やりたいこと

お得な特典付きプラン というテキス トを含む **カード**の取得

実際

お得な特典付きプラン というテキス トを含む **見出し** が取得された

探索の範囲を絞り込む

div.card-body Color Font 16px -apple-system, "system-ui Padding	688×202 #212529 ", "Seg 20px	ログイン
ACCESSIBILITY		
Name		
Role	generic	
Keyboard-focusable	\otimes	
お得な特典付: 大人1名7,00 1名様か スタンダード このプランで	き プラン 00円 3 ツイン 9 予 約	
本日限り)	

カードを表すclassは card-body

cy.contains('div.card-body', 'お得な特典付きプラン') .contains('このプランで予約').click()

h5 ではなく

card-body というclassを持つ div 要素を取得するようになった

現在のテストコード

```
describe('スモークテスト', () => {
it('非会員で予約', () => {
```

})

// 1. https://hotel.testplanisphere.dev/ja/を開く cy.visit("https://hotel.testplanisphere.dev/ja/index.html");

```
// 2. メニューから「宿泊予約」を選択
cy.contain('宿泊予約').click()
```

```
// 3. 宿泊プラン一覧から「お得な特典付きプラン」の「このプランで予約」を選択
cy.contains('div.card-body', 'お得な特典付きプラン')
.contains('このプランで予約').click()
})
```

考えてみよう

このコードは読みやすい?

cy.contains('div.card-body', 'お得な') .contains('このプランで予約').click()

- div.card-body なんて、元のテスト設計にあったっけ?
- div.card-body がどのUIに対応してるか、後で思い出せる?
- ユーザーは div.card-body というclassを意識することがある?

😂 よくない臭いがするぞ!

テスト設計に**出てこない言葉**がテストコードに出てきたら、 テストコードからその箇所を**分離**すべきかも

カスタムコマンドを追加する

cypress/support/commands.js に以下を追加する

```
Cypress.Commands.add("getCardByText", (text) => {
    const selector = 'div.card-body'
    cy.contains(selector, text)
});
```

こう書けるようになった

```
// before
cy.contains('div.card-body', 'お得な特典付きプラン')
.contains('このプランで予約').click()
```

// after cy.getCardByText('お得な特典付きプラン').contains('このプランで予約').click()

さらに別の問題

このプランで予約 は新しいウィンドウを開くが Cypressは 複数ウィンドウのテストに対応していない

新しいウィンドウを開かないようにする

cy.getCardByText('お得な特典付きプラン') .contains('このプランで予約') .invoke('removeAttr', 'target')

リンクから「新しいウィンドウを開く」ための指定 target="_blank" を除く

参考: https://testersdock.com/cypress-new-window/

新たなカスタムコマンドを定義しよう

予約プランを開くカスタムコマンドを定義する

```
Cypress.Commands.add("openReservationPlan", (planName) => {
const buttonText = "このプランで予約"
cy
.getCardByText(planName)
.contains(buttonText)
.invoke("removeAttr", "target")
.click()
})
```

テストコードはこう書ける

// before

```
cy.getCardByText('お得な特典付きプラン').contains('このプランで予約').click()
```

// after cy.openReservationPlan('お得な特典付きプラン')

なんかめんどくさいね?

E2Eテストを書くこと自体は簡単ですが

- ツールの技術的制約の回避
- テストしづらいコンポーネントの操作

などはやっぱりめんどくさい(そしてどうしようもない)

なんでわざわざ Custom Command とか使うの?

テストスクリプトから **ユーザー操作と無関係な部分**を切り離す

- 自動化の都合でやらなければいけない処理(例:新規ウィンドウを抑制する)
- サイトの構造を表現するのに必要な記述(例: CSSセレクタ)

めんどくさい部分はどうしても出てくるので そこを上手く隠せると読みやすいコードになる

続けて書いていきましょう

- 4. 宿泊日を翌月1日に設定
- 5. 宿泊数を7泊に設定
- 6. 人数を2に設定
- 7. 朝食バイキング、昼からチェックインプラン、お得な観光プランを選択
- 8. 氏名に「テスト太郎」を入力
- 9. 確認のご連絡をメールに設定
- 10. メールアドレスにhoge@example.comを設定
- 11. ご要望・ご連絡事項に「テスト」と入力
- 12. 予約内容を確認するボタンを選択



宿泊予約

お得な特典付きプラン

お一人様1泊7,000円~、土日は25%アップ。1名様~9名様、最長9泊

宿泊日後須		合計
2022/02/24		
ご予約は3ヶ月以内の日付のみ可能です。		7,000円
宿泊数必須		予約内容を確認する
1	泊	
人数 必須		スタンダードツイン ^{部屋タイプ}
1	人	ツイン
追加プラン お一人様各1,000円		定員
□ 朝食バイキング		1~2名
□ 昼からチェックインプラン		広さ 18㎡
□ お得な観光プラン		設備
氏名 必須		• ユニット式バス・トイ

フォーム入力が多い どうやって目当てのフォームに 入力するか?

HTMLのフォームの仕組みについておさらい

<label for="name">お名前</label><input id="name" type="text" />

- label と input で出来ていることが多い
- label に for 属性を付けると label と input が紐付けられる
- label をクリックすると input にフォーカスが移る

Cypressではどう扱われるか

<label for="name">お名前</label><input id="name" type="text" />

// labelが返ってくる cy.contains("お名前")

contains で取得できる要素は厳密には label 要素なので フォームに対する操作の場合、 contains では上手く動かない場合がある

- 普通の入力フォームへの入力はOK
- セレクトボックスやチェックボックスはNG

○ Clickableな要素として扱われない

ラベルのテキストからinput要素を見つける

そんなコマンドがあったらいいのにね

<label for="name">お名前</label><input id="name" type="text" />

// labelが返ってくる cy.contains("お名前")

// inputが返ってくる cy.getByLabel("お名前")

カスタムコマンド getByLabel の使用

インストール

\$ npm install cypress-get-by-label

cypress/support/commands.js に以下を追加

const { registerCommand } = require("cypress-get-by-label"); registerCommand();

宿泊予約

cy.getByLabel('宿泊日').type('2022-02-12') cy.getByLabel('宿泊数').type('7') cy.getByLabel('人数').type('1') cy.getByLabel('朝食バイキング').check() cy.getByLabel('氏名').type('ジャスト 太郎') cy.getByLabel('確認のご連絡').select('希望しない') cy.contains('予約内容を確認する').click()

上手く行かなかった



- 元々入力されているテキストに追記してしまった
- カレンダーウィジェットが表示されたまま

対処

//「宿泊日」フィールドに入っている値を一度全て消す cy.getByLabel('宿泊日').clear();

// 入力の後に ESC キーを押下してカレンダーウィジェットを消す cy.getByLabel('宿泊日').type('2022/02/12{esc}');

これもカスタムコマンドにしてしまえ

値を一度削除してから入力する fill メソッドを定義する

Cypress.Commands.add("fill", { prevSubject: 'element' }, (subject, text) => {
 subject.clear();
 subject.type(text)
})

テストコードはこうなる

cy.getByLabel('宿泊日').fill('2022/02/21{esc}')

宿泊日を翌月1日に設定

日付処理をする dayjs というライブラリを使う

\$ npm install dayjs

```
describe("スモークテスト", () => {
const dayjs = require("dayjs");
const checkInDate = dayjs().add(1, "month").startOf("month");
```

```
it("会員登録して予約してログアウト", () => {
```

```
// ...
// 4. 宿泊日を翌月1日に設定
cy.getByLabel("宿泊日").fill(`${checkInDate.format("YYYY/MM/DD")}{esc}`);
```

この日付が表す意味を表現する

```
context はテストコードに「文脈」を与える
```

```
describe("スモークテスト", () => {
```

```
context("翌月1日から7日間予約する", () => {
const dayjs = require("dayjs");
const checkInDate = dayjs().add(1, "month").startOf("month");
const checkOutDate = checkInDate.add(7, "day");
```

```
it("会員登録して予約してログアウト", () => {
```

現在のテストコード

describe("スモークテスト", () => {

context("翌月1日から7日間予約する", () => { const dayjs = require("dayjs"); const checkInDate = dayjs().add(1, "month").startOf("month"); const checkOutDate = checkInDate.add(7, "day");

it("会員登録して予約してログアウト", () => { // 1. https://hotel.testplanisphere.dev/ja/を開く cy.visit("https://hotel.testplanisphere.dev/ja/index.html");

// 2. メニューから「宿泊予約」を選択 cy.contains("宿泊予約").click();

// 3. 宿泊プラン一覧から「お得な特典付きプラン」の「このプランで予約」を選択 cy.openReservationPlan("お得な特典付きプラン");

cy.wait(1000);

// 4. 宿泊日を翌月1日に設定 cy.getByLabel("宿泊日").fill(`\${checkInDate.format("YYYY/MM/DD")}{esc}`);

// 5. 宿泊数を7泊に設定 cy.getByLabel("宿泊数").fill("7");

// 6. 人数を2に設定 cy.getByLabel("人数").fill("2"); // 7. 朝食バイキング、昼からチェックインプラン、お得な観光プランを選択 cy.getByLabel("朝食バイキング").check(); cy.getByLabel("昼からチェックインプラン").check(); cy.getByLabel("お得な観光プラン").check();

// 8. 氏名に「テスト太郎」を入力 cy.getByLabel("氏名").fill("テスト 太郎");

// 9. 確認のご連絡をメールに設定 cy.getByLabel("確認のご連絡").select("メールでのご連絡");

// 10. メールアドレスにhoge@example.comを設定 cy.getByLabel("メールアドレス").fill("hoge@example.com");

```
    // 11. ご要望・ご連絡事項に「テスト」と入力
    cy.getByLabel("ご要望・ご連絡事項等ありましたらご記入ください").fill(
"テスト");
    // 12. 予約内容を確認するボタンを選択
    cy.contains("予約内容を確認する").click();
```

}); }); });

予約内容の確認

13. 宿泊予約確認画面で、以下を確認

i. 合計金額が123,000円であること

ii. 期間、人数、追加プラン、お名前、確認のご連絡、ご要望・ご連絡が 入力通りになっていること

14. この内容で予約するボタンを選択し、以下を確認

i. 予約が完了しましたダイアログが表示されること

アサーション

should の後に条件を記述する。 この例では「合計」を含む要素が「123,000円」を含むことを確認している

cy.contains("合計").should("contain", "123,000円");

https://docs.cypress.io/guides/references/assertions#Common-Assertions

テストコード

```
// 13. 宿泊予約確認画面で、以下を確認
  1. 合計金額が123,000円であること
//
// 2. 期間、人数、追加プラン、お名前、確認のご連絡、ご要望・ご連絡が入力通りになっていること
cy.contains("合計").should("contain", "123,000円");
cy.contains("お得な特典付きプラン");
cy.contains("期間")
 .next()
 .should(
  "contain".
  `${checkInDate.format("YYYY年M月D日")}~${checkOutDate.format("YYYY年M月D日")}7泊`
 );
cy.contains("人数").next().should("contain", "2名様");
cy.contains("追加プラン").next().should("contain", "朝食バイキング");
cy.contains("追加プラン").next().should("contain", "昼からチェックインプラン");
cy.contains("お名前").next().should("contain", "テスト 太郎様");
cy.contains("追加プラン").next().should("contain", "お得な観光プラン");
cy.contains("お名前").next().should("contain", "テスト 太郎様");
cy.contains("確認のご連絡")next().should("contain", "メール: hoge@example.com");
cv.contains("ご要望・ご連絡事項等").next().should("contain", "テスト");
// 14. この内容で予約するボタンを選択し、以下を確認
// 1. 予約が完了しましたダイアログが表示されること
cy.contains("この内容で予約する").click();
cv.wait(2000);
cv.contains("予約を完了しました"):
```

おわりに

Cypressについて

- Cypressは拡張性が高く、テストコードをきれいに記述するのに充分な機能を 備えています
- 反面、複数ウィンドウを利用するサイトのテストなど、対応していないサイトのテストにはコツが要ります
- まずは触ってみて、自分のプロジェクトに適用可能か確かめてみましょう

おさらい: わかりやすいテストコードを書くコツ

1. ユーザー目線の表記を心がける

サイトの内部構造を使わず、表示されたテキストで選択する

2. あいまいな部分を減らす

「xxの中のyy」というように指定して、要素探索の範囲を絞り込む

3. 「何をテストしているのか」と「どうテストするのか」を分ける

テストコードから不要な情報を出来るだけ省いて シンプルなコードを保つ

ぜひみなさんもトライしてみてください

Enjoy Testing!