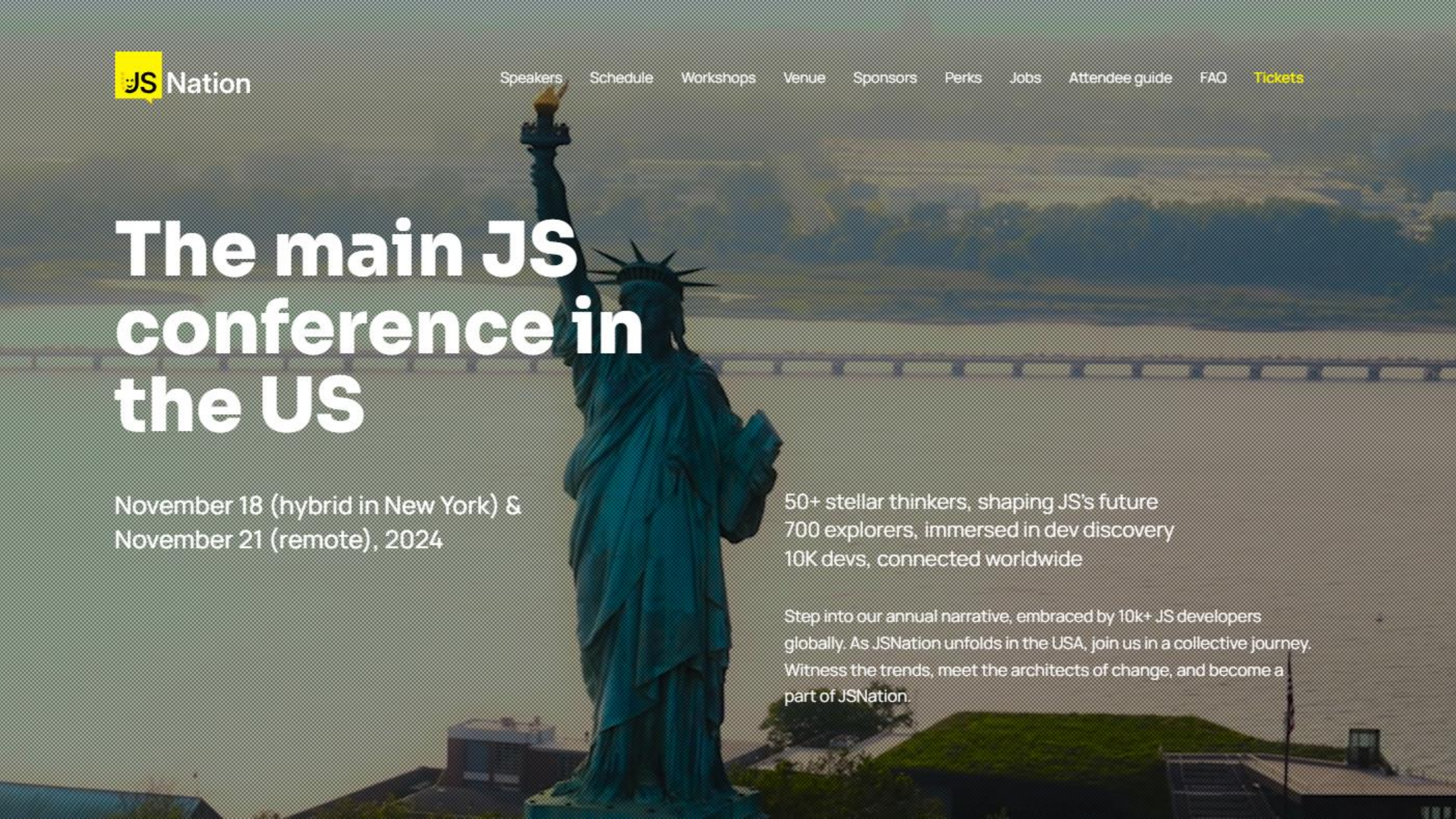


The main JS conference in the US

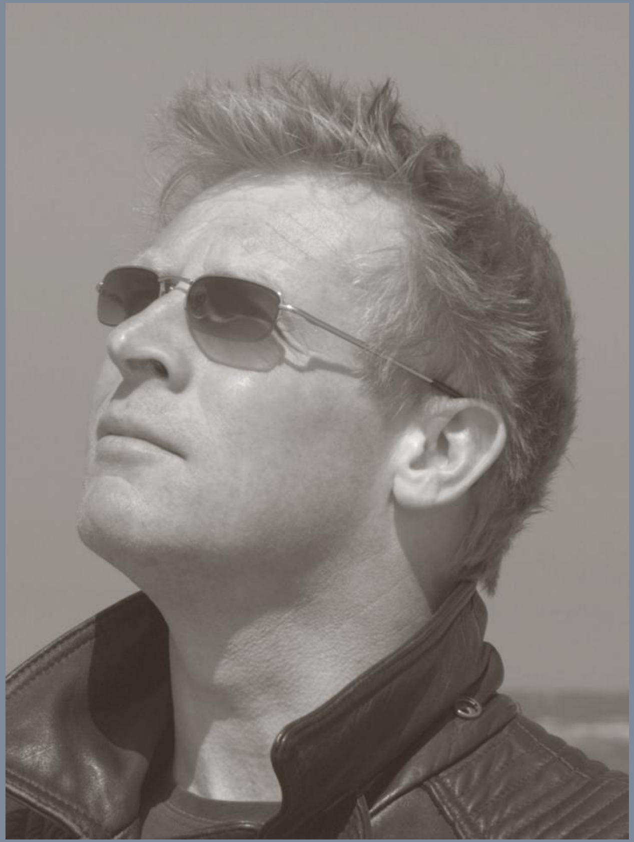
November 18 (hybrid in New York) &
November 21 (remote), 2024

50+ stellar thinkers, shaping JS's future
700 explorers, immersed in dev discovery
10K devs, connected worldwide

Step into our annual narrative, embraced by 10k+ JS developers globally. As JSNation unfolds in the USA, join us in a collective journey. Witness the trends, meet the architects of change, and become a part of JSNation.

Building Robust Web Applications with Test-Driven Development and Playwright

Maurice de Beijer
@mauricedb



- Maurice de Beijer
- The Problem Solver
- Freelance developer/instructor
- Twitter: @mauricedb
- Web: http://www.theproblemsolver.dev/
- E-mail: maurice.de.beijer@gmail.com

What We'll Build Today

- **Movie Browsing Application**
  - Landing page with navigation
  - List of top-rated movies
  - Movie details page
  - Movie editing functionality
- **Learning Objectives**
  - TDD workflow in frontend development based on user stories
  - Writing effective Playwright tests
  - Building robust web applications
  - Real-world testing scenarios

Type it out by hand?

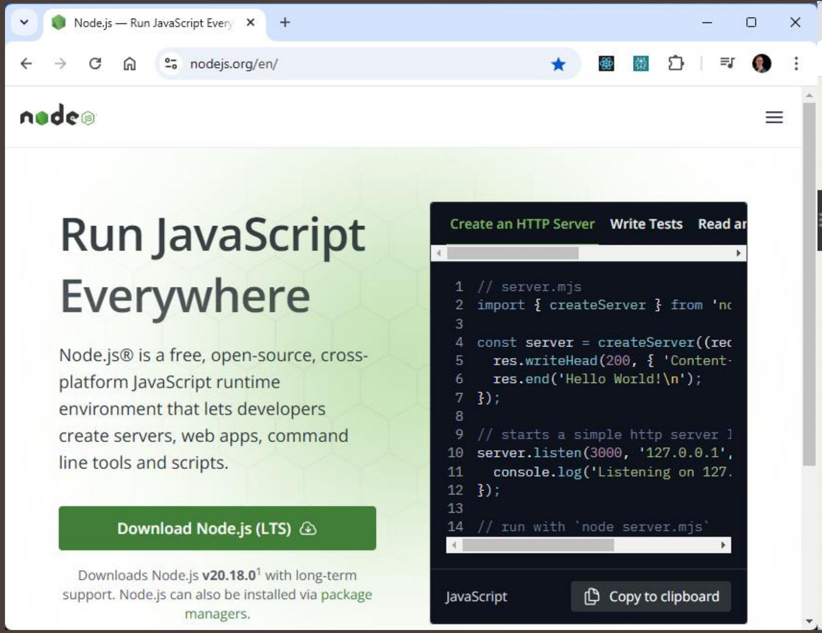
"Typing it drills it into your brain much better than simply copying and pasting it. You're forming new neuron pathways. Those pathways are going to help you in the future. Help them out now!"

Prerequisites

Install Node & NPM

Install the GitHub repository

Install Node.js & NPM


Node.js — Run JavaScript Every

nodejs.org/en/

Run JavaScript Everywhere

Node.js® is a free, open-source, cross-platform runtime environment that lets developers create servers, web apps, command line tools and scripts.

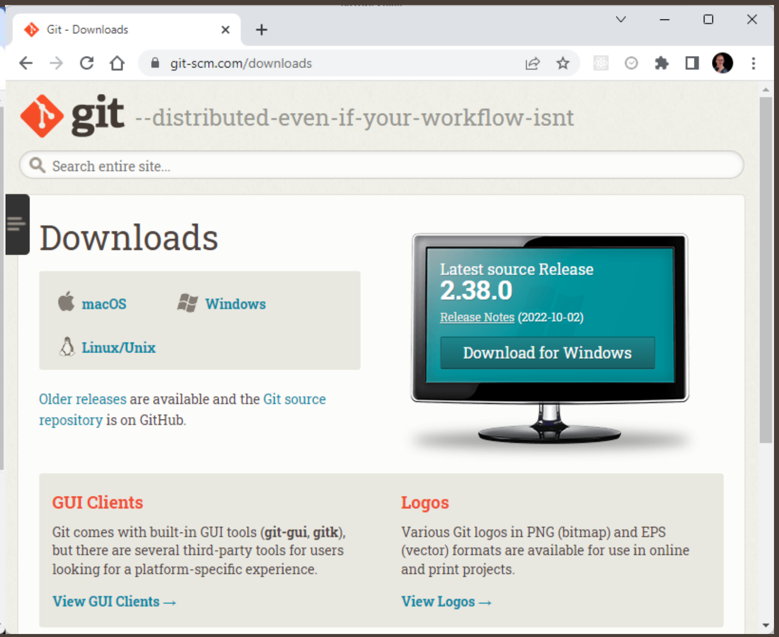
Download Node.js (LTS)

Downloads Node.js **v20.18.0**[1] with long-term support. Node.js can also be installed via package managers.

Create an HTTP Server Write Tests Read ar

```
// server.mjs
import { createServer } from 'nc

const server = createServer((rec
  res.writeHead(200, { 'Content-
  res.end('Hello World!\n');
});

// starts a simple http server l
server.listen(3000, '127.0.0.1',
  console.log('Listening on 127.
});

// run with `node server.mjs`
```

JavaScript Copy to clipboard


Git - Downloads

git-scm.com/downloads

git --distributed-even-if-your-workflow-isnt

Search entire site...

Downloads

macOS Windows

Linux/Unix

Older releases are available and the Git source repository is on GitHub.

Latest source Release
2.38.0
Release Notes (2022-10-02)
Download for Windows

GUI Clients

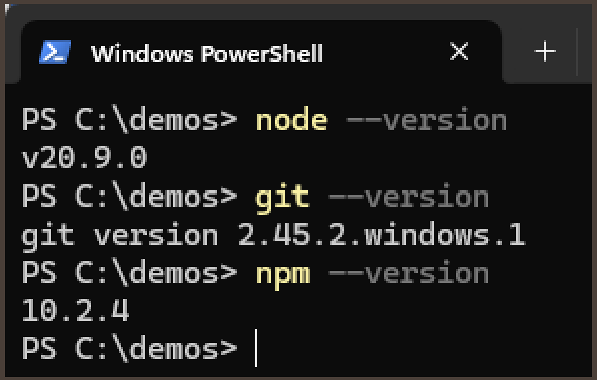
Git comes with built-in GUI tools (**git-gui**, **gitk**), but there are several third-party tools for users looking for a platform-specific experience.

View GUI Clients →

Logos

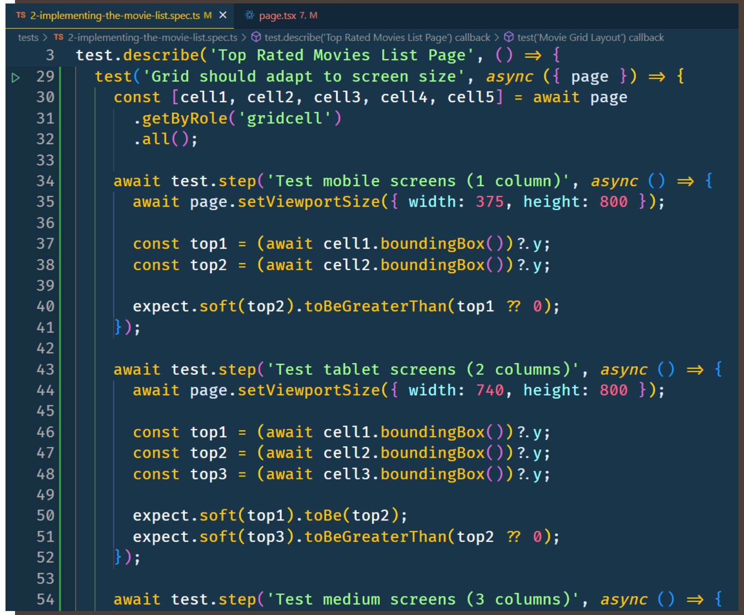
Various Git logos in PNG (bitmap) and EPS (vector) formats are available for use in online and print projects.

View Logos →


Windows PowerShell

```
PS C:\demos> node --version
v20.9.0
PS C:\demos> git --version
git version 2.45.2.windows.1
PS C:\demos> npm --version
10.2.4
PS C:\demos>
```

Following Along

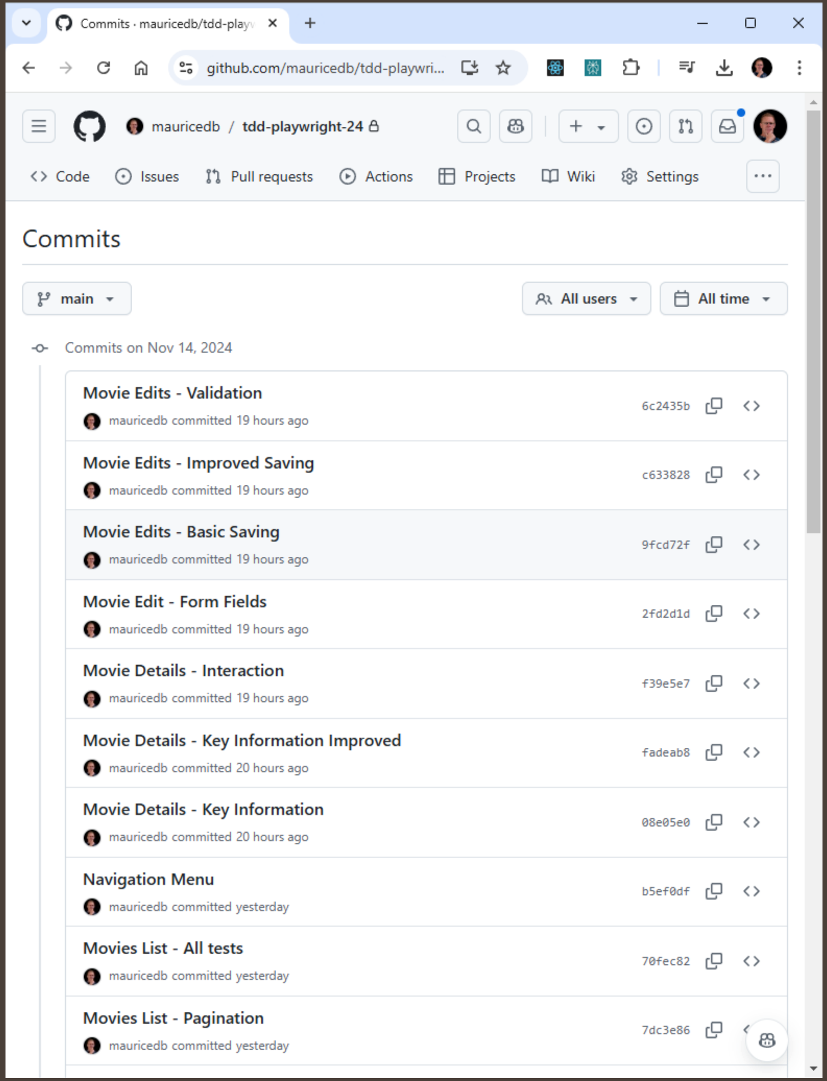

TS 2-implementing-the-movie-list.spec.ts M ✕   page.tsx 7, M

tests > TS 2-implementing-the-movie-list.spec.ts > test.describe('Top Rated Movies List Page') callback > test('Movie Grid Layout') callback

```
test.describe('Top Rated Movies List Page', () ⇒ {
  test('Grid should adapt to screen size', async ({ page }) ⇒ {
    const [cell1, cell2, cell3, cell4, cell5] = await page
      .getByRole('gridcell')
      .all();

    await test.step('Test mobile screens (1 column)', async () ⇒ {
      await page.setViewportSize({ width: 375, height: 800 });

      const top1 = (await cell1.boundingBox())?.y;
      const top2 = (await cell2.boundingBox())?.y;

      expect.soft(top2).toBeGreaterThan(top1 ?? 0);
    });

    await test.step('Test tablet screens (2 columns)', async () ⇒ {
      await page.setViewportSize({ width: 740, height: 800 });

      const top1 = (await cell1.boundingBox())?.y;
      const top2 = (await cell2.boundingBox())?.y;
      const top3 = (await cell3.boundingBox())?.y;

      expect.soft(top1).toBe(top2);
      expect.soft(top3).toBeGreaterThan(top2 ?? 0);
    });

    await test.step('Test medium screens (3 columns)', async () ⇒ {
```

- Repo: https://github.com/mauricedb/tdd-playwright-24
- Slides: https://www.theproblemsolver.dev/docs/jsnation-us-2024.pdf

The changes


Commits · mauricedb/tdd-play
github.com/mauricedb/tdd-playwri...
mauricedb / tdd-playwright-24
Code
Issues
Pull requests
Actions
Projects
Wiki
Settings
main
All users
All time

Commits

Commits on Nov 14, 2024

Movie Edits - Validation
mauricedb committed 19 hours ago 6c2435b

Movie Edits - Improved Saving
mauricedb committed 19 hours ago c633828

Movie Edits - Basic Saving
mauricedb committed 19 hours ago 9fcd72f

Movie Edit - Form Fields
mauricedb committed 19 hours ago 2fd2d1d

Movie Details - Interaction
mauricedb committed 19 hours ago f39e5e7

Movie Details - Key Information Improved
mauricedb committed 20 hours ago fadeab8

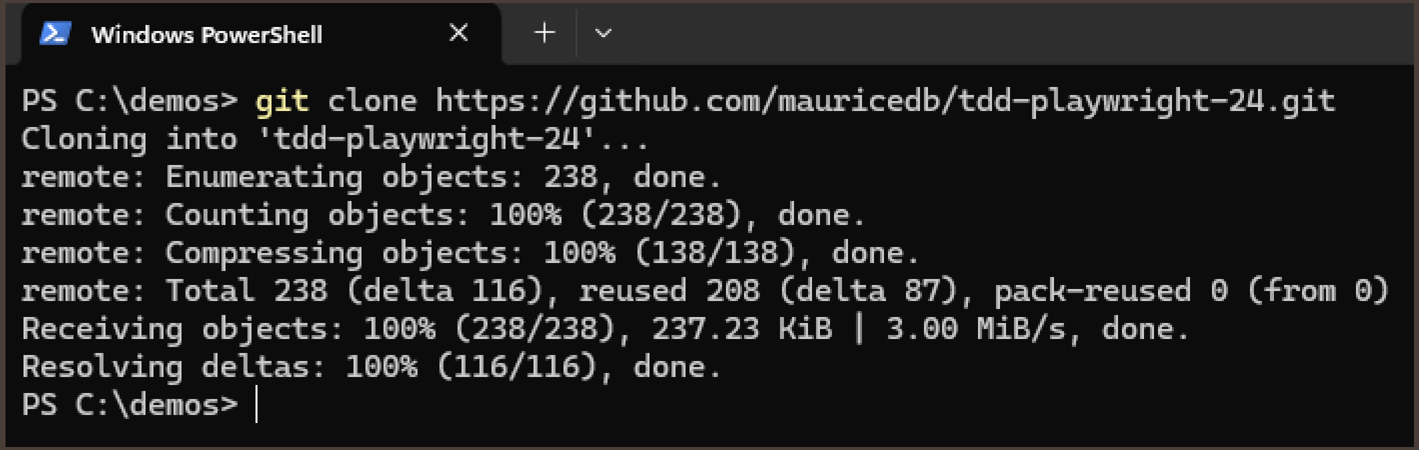
Movie Details - Key Information
mauricedb committed 20 hours ago 08e05e0

Navigation Menu
mauricedb committed yesterday b5ef0df

Movies List - All tests
mauricedb committed yesterday 70fec82

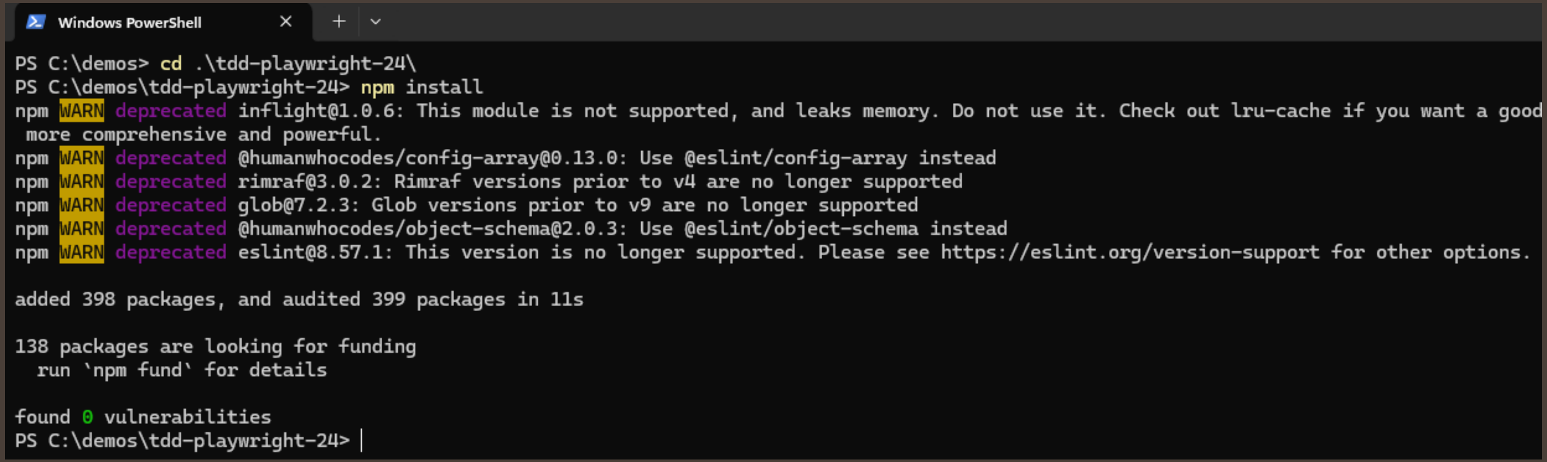
Movies List - Pagination
mauricedb committed yesterday 7dc3e86

Clone the GitHub Repository



```
PS C:\demos> git clone https://github.com/mauricedb/tdd-playwright-24.git
Cloning into 'tdd-playwright-24'...
remote: Enumerating objects: 238, done.
remote: Counting objects: 100% (238/238), done.
remote: Compressing objects: 100% (138/138), done.
remote: Total 238 (delta 116), reused 208 (delta 87), pack-reused 0 (from 0)
Receiving objects: 100% (238/238), 237.23 KiB | 3.00 MiB/s, done.
Resolving deltas: 100% (116/116), done.
PS C:\demos>
```

Install NPM Packages

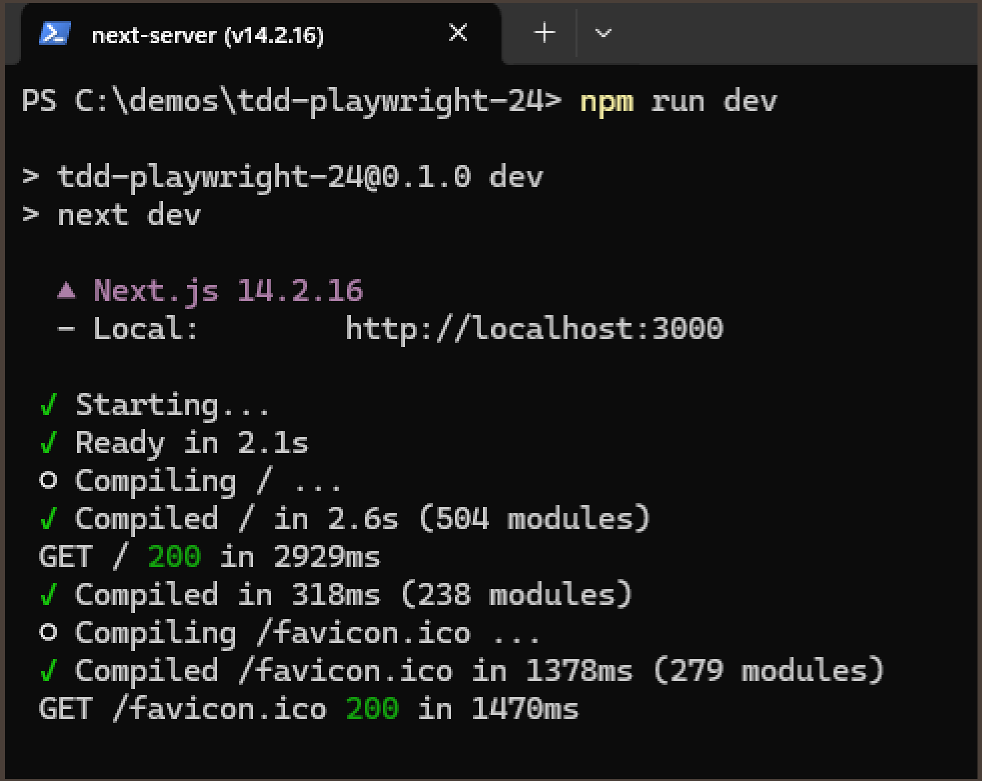


```
PS C:\demos> cd .\tdd-playwright-24\
PS C:\demos\tdd-playwright-24> npm install
npm WARN deprecated inflight@1.0.6: This module is not supported, and leaks memory. Do not use it. Check out lru-cache if you want a good
 more comprehensive and powerful.
npm WARN deprecated @humanwhocodes/config-array@0.13.0: Use @eslint/config-array instead
npm WARN deprecated rimraf@3.0.2: Rimraf versions prior to v4 are no longer supported
npm WARN deprecated glob@7.2.3: Glob versions prior to v9 are no longer supported
npm WARN deprecated @humanwhocodes/object-schema@2.0.3: Use @eslint/object-schema instead
npm WARN deprecated eslint@8.57.1: This version is no longer supported. Please see https://eslint.org/version-support for other options.

added 398 packages, and audited 399 packages in 11s

138 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\demos\tdd-playwright-24>
```

Start branch

- Start with the **00-start** branch
  - `git checkout --track origin/00-start`

Start the application



```
PS C:\demos\tdd-playwright-24> npm run dev

> tdd-playwright-24@0.1.0 dev
> next dev

  ▲ Next.js 14.2.16
  - Local:        http://localhost:3000

 ✓ Starting...
 ✓ Ready in 2.1s
 ○ Compiling / ...
 ✓ Compiled / in 2.6s (504 modules)
 GET / 200 in 2929ms
 ✓ Compiled in 318ms (238 modules)
 ○ Compiling /favicon.ico ...
 ✓ Compiled /favicon.ico in 1378ms (279 modules)
 GET /favicon.ico 200 in 1470ms
```

Introduction to Test-Driven Development (TDD)

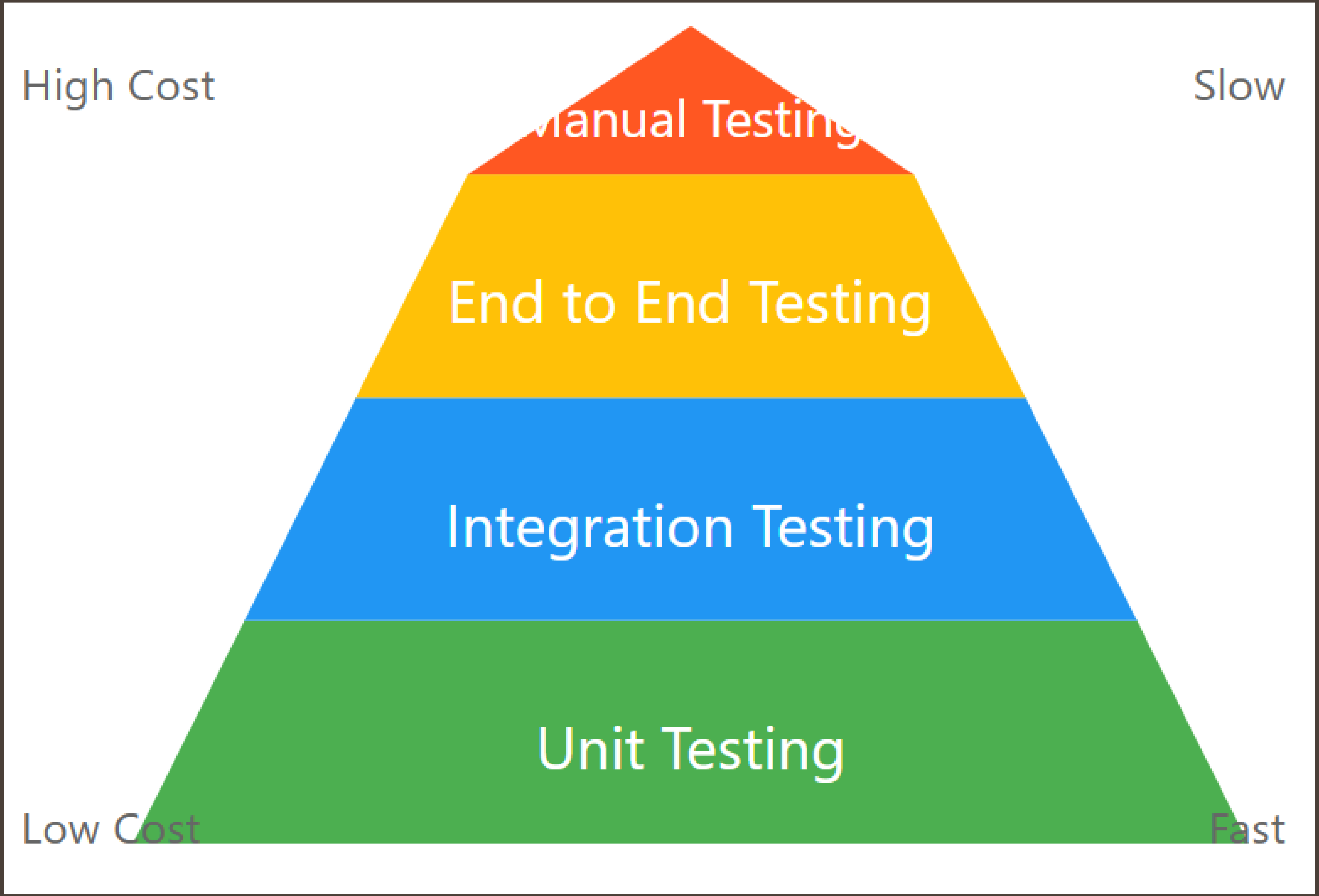
What is Test-Driven Development?

- A software **development approach** where tests are written before the actual code
- **Tests drive the design and implementation** of the code
- “**Red-Green-Refactor**” cycle

The TDD Cycle

- Write a **failing test** (Red)
- Write **minimal code to make the test pass** (Green)
- **Refactor the code** while keeping tests green
- Repeat…

Software Testing Pyramid



Benefits of TDD

- **Improved Code Quality**
    - Fewer bugs and defects
    - Better code coverage
    - Cleaner, more maintainable code
    - Built-in documentation through tests
- **Faster Development**
    - Catch bugs early in the development cycle
    - Reduce debugging time
    - More confident code changes
    - Easier refactoring
- **Better Design**
    - Forces modular design
    - Reduces code coupling
    - Promotes interface-driven development
    - Makes code more testable

Common TDD Misconceptions

- **"TDD takes too much time"**
  - Initial investment pays off in reduced debugging and maintenance
  - Faster identification of issues
  - Less time spent on manual testing
- **"I'll write tests later"**
  - Tests written after code tend to be incomplete
  - Missing edge cases
  - Code might not be designed for testability
- **"TDD is only for backend development"**
  - Frontend can benefit greatly from TDD
  - Ensures consistent UI behavior
  - Catches regression issues early

Introduction to Playwright

A powerful end-to-end testing framework for web applications

What is Playwright?

- **Modern end-to-end testing framework**
- Created and maintained as **open source** by Microsoft
- Support for **modern browsers**
- **Cross-platform** support

Key Features

- **Auto-wait** capabilities
- **Network interception**
- **Mobile device** emulation
- **Multiple browser** contexts
- Powerful **debugging tools**

Why Playwright?

- **Advantages**
  - Fast and reliable tests
  - Cross-browser support out of the box
  - Modern features like web sockets
  - Rich debugging capabilities
  - Strong TypeScript support
- **Use Cases**
  - End-to-end testing
  - Component testing
  - Visual regression testing
  - Performance testing
  - Network monitoring

Playwright Core Concepts

- **Browser Contexts**
  - Isolated browser sessions
  - Independent cookie/storage states
  - Perfect for testing multi-user scenarios
- **Auto-waiting**
  - Element availability
  - Network requests
  - Animations
  - No need for explicit waits
- **Locators**
  - Reliable element selection
  - Built-in retry logic
  - Multiple selection strategies

Combining TDD and Playwright

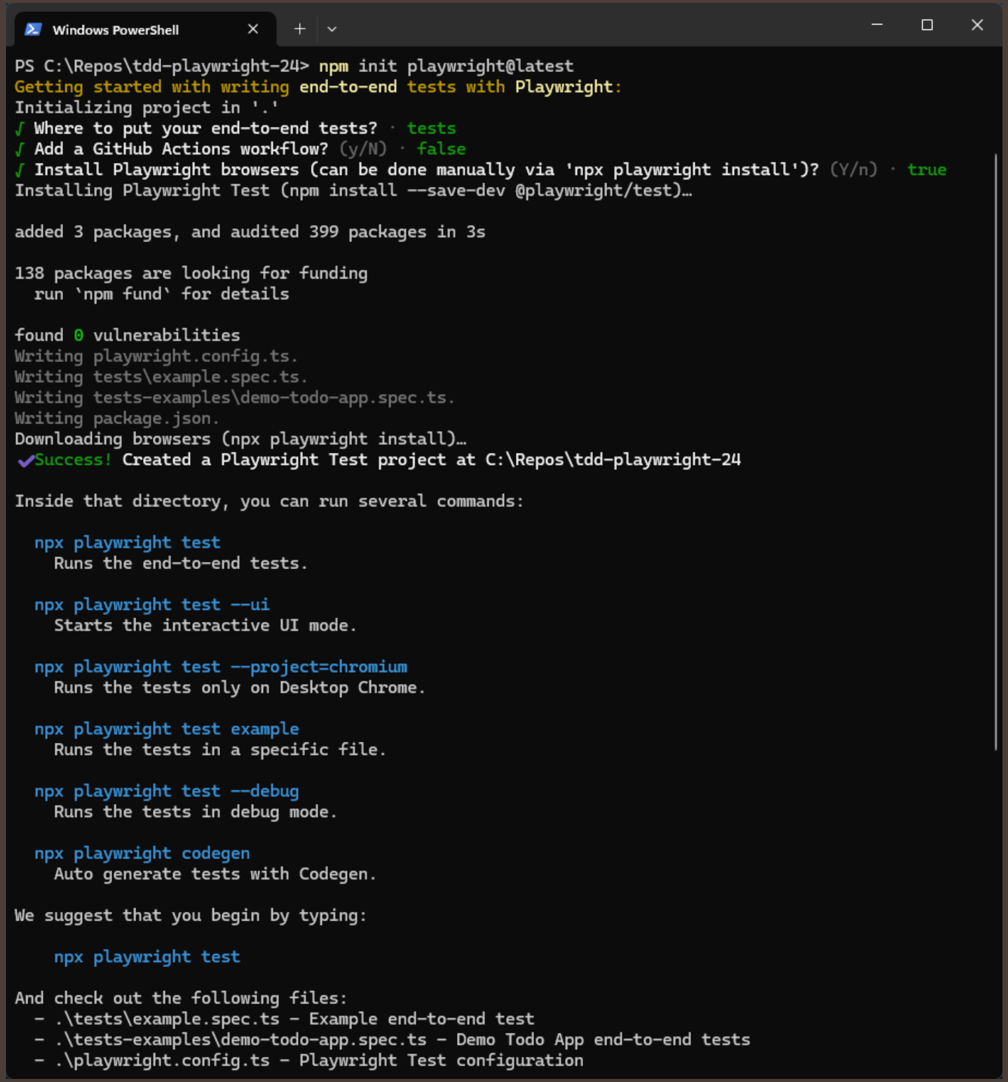
- **Workflow**
  - Write a failing Playwright test (Red)
  - Implement the feature
  - Run tests and fix issues (Green)
  - Refactor with confidence
- **Benefits**
  - Consistent UI behavior
  - Caught regression issues
  - Documented features
  - Confident deployments

Installing Playwright

Installing Playwright

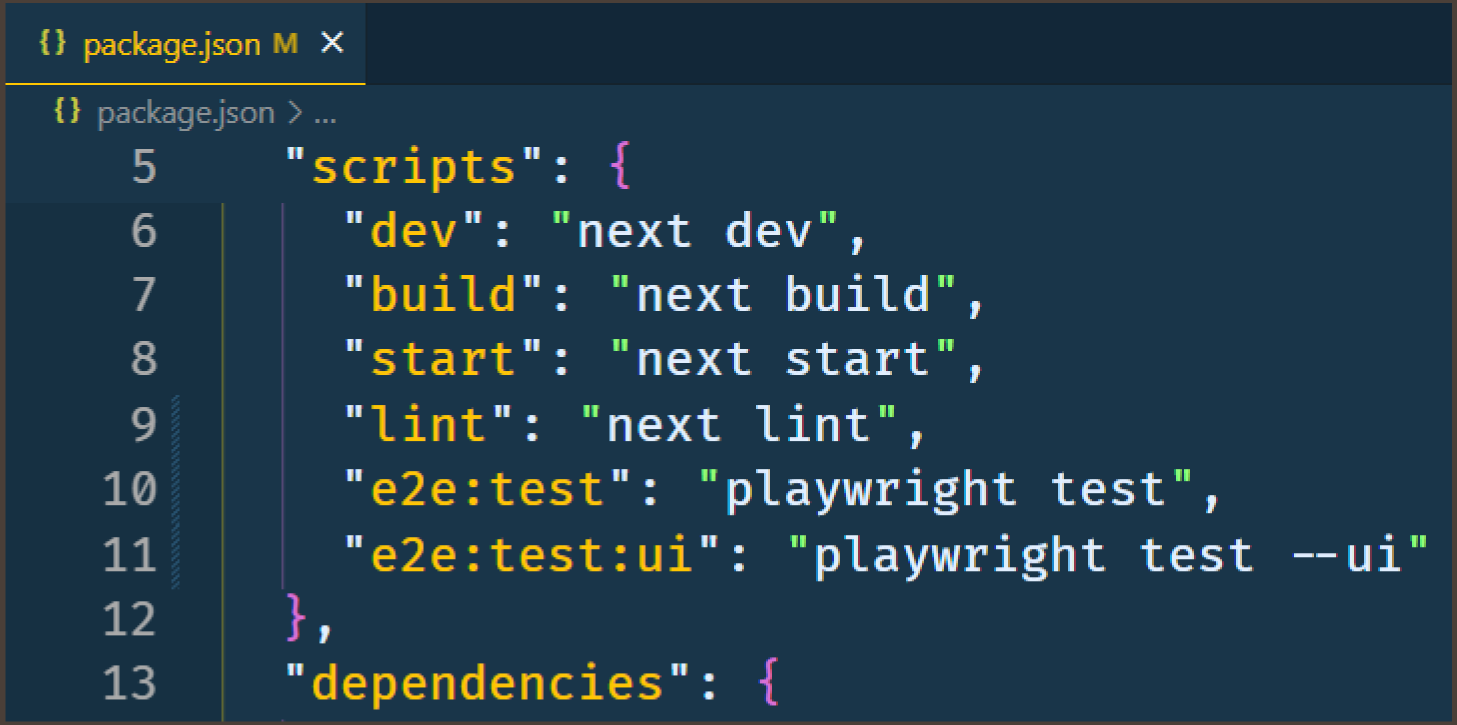
- Install Playwright from a terminal window in the root folder
  - ***`npm init playwright@latest`***
- The VS Code extension is a good alternative
  - Also allows for executing tests

npm init playwright



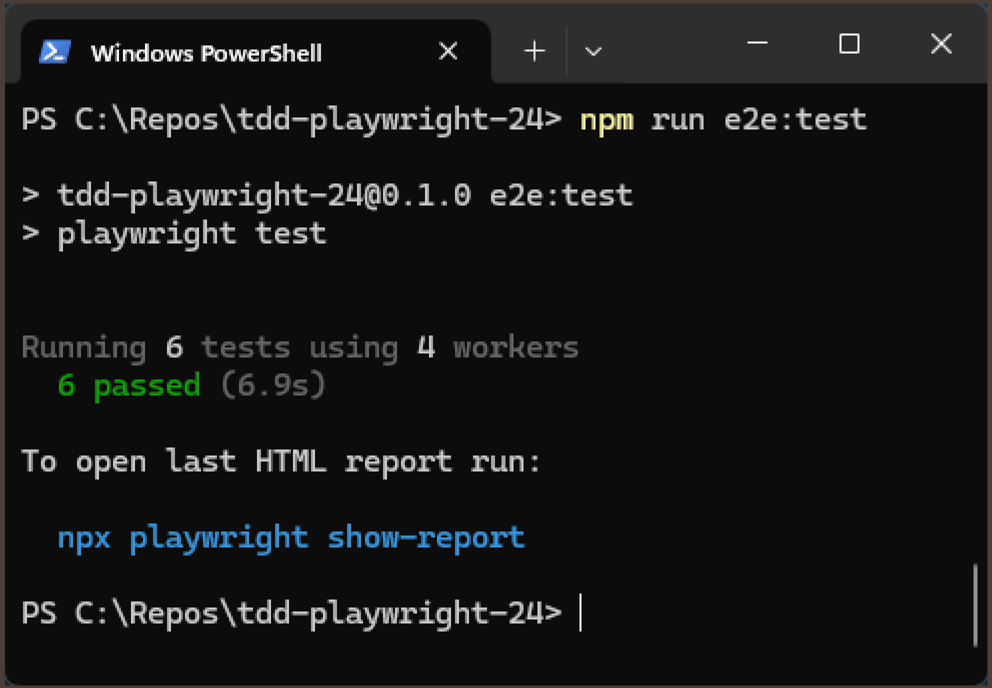
```
PS C:\Repos\tdd-playwright-24> npm init playwright@latest
Getting started with writing end-to-end tests with Playwright:
Initializing project in '.'
√ Where to put your end-to-end tests? · tests
√ Add a GitHub Actions workflow? (y/N) · false
√ Install Playwright browsers (can be done manually via 'npx playwright install')? (Y/n) · true
Installing Playwright Test (npm install --save-dev @playwright/test)…

added 3 packages, and audited 399 packages in 3s

138 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
Writing playwright.config.ts.
Writing tests\example.spec.ts.
Writing tests-examples\demo-todo-app.spec.ts.
Writing package.json.
Downloading browsers (npx playwright install)…
✔Success! Created a Playwright Test project at C:\Repos\tdd-playwright-24

Inside that directory, you can run several commands:

  npx playwright test
    Runs the end-to-end tests.

  npx playwright test --ui
    Starts the interactive UI mode.

  npx playwright test --project=chromium
    Runs the tests only on Desktop Chrome.

  npx playwright test example
    Runs the tests in a specific file.

  npx playwright test --debug
    Runs the tests in debug mode.

  npx playwright codegen
    Auto generate tests with Codegen.

We suggest that you begin by typing:

    npx playwright test

And check out the following files:
  - .\tests\example.spec.ts - Example end-to-end test
  - .\tests-examples\demo-todo-app.spec.ts - Demo Todo App end-to-end tests
  - .\playwright.config.ts - Playwright Test configuration
```

package.json



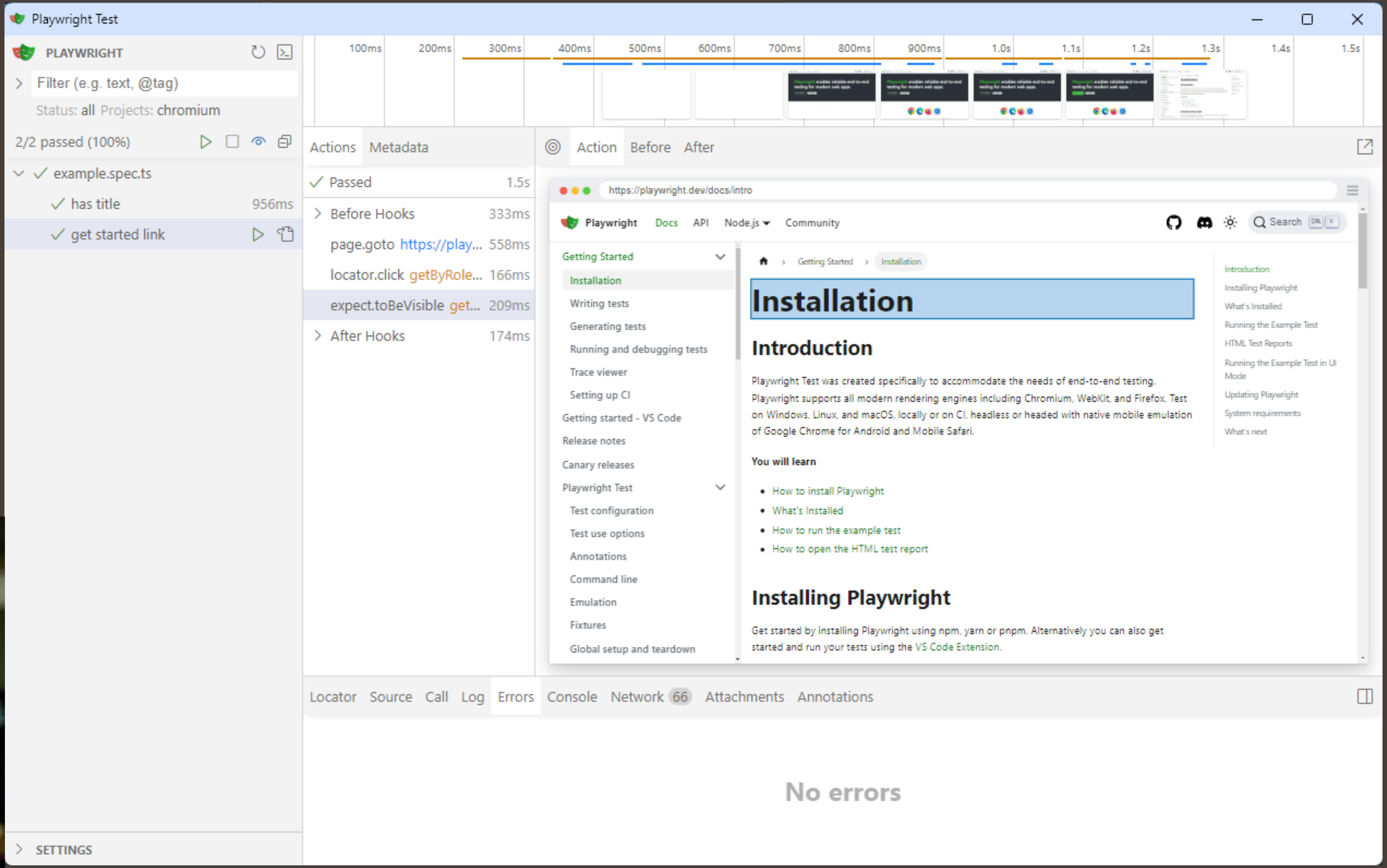
```
  "scripts": {
    "dev": "next dev",
    "build": "next build",
    "start": "next start",
    "lint": "next lint",
    "e2e:test": "playwright test",
    "e2e:test:ui": "playwright test --ui"
  },
  "dependencies": {
```

Playwright test console mode



```
PS C:\Repos\tdd-playwright-24> npm run e2e:test

> tdd-playwright-24@0.1.0 e2e:test
> playwright test


Running 6 tests using 4 workers
  6 passed (6.9s)

To open last HTML report run:

  npx playwright show-report

PS C:\Repos\tdd-playwright-24>
```

Playwright test in UI mode



Playwright Test

PLAYWRIGHT

Filter (e.g. text, @tag)

Status: all Projects: chromium

2/2 passed (100%)

- ✓ example.spec.ts
  - ✓ has title 956ms
  - ✓ get started link

SETTINGS

Actions	Metadata
✓ Passed	1.5s
Before Hooks	333ms
page.goto https://play...	558ms
locator.click getByRole...	166ms
expect.toBeVisible get...	209ms
After Hooks	174ms

Action Before After

https://playwright.dev/docs/intro

Playwright Docs API Node.js Community Search

Getting Started › Installation

Installation

Introduction

Playwright Test was created specifically to accommodate the needs of end-to-end testing. Playwright supports all modern rendering engines including Chromium, WebKit, and Firefox. Test on Windows, Linux, and macOS, locally or on CI, headless or headed with native mobile emulation of Google Chrome for Android and Mobile Safari.

You will learn

- How to install Playwright
- What's Installed
- How to run the example test
- How to open the HTML test report

Installing Playwright

Get started by installing Playwright using npm, yarn or pnpm. Alternatively you can also get started and run your tests using the VS Code Extension.

Locator Source Call Log Errors Console Network 66 Attachments Annotations

No errors

Implementing the Landing Page

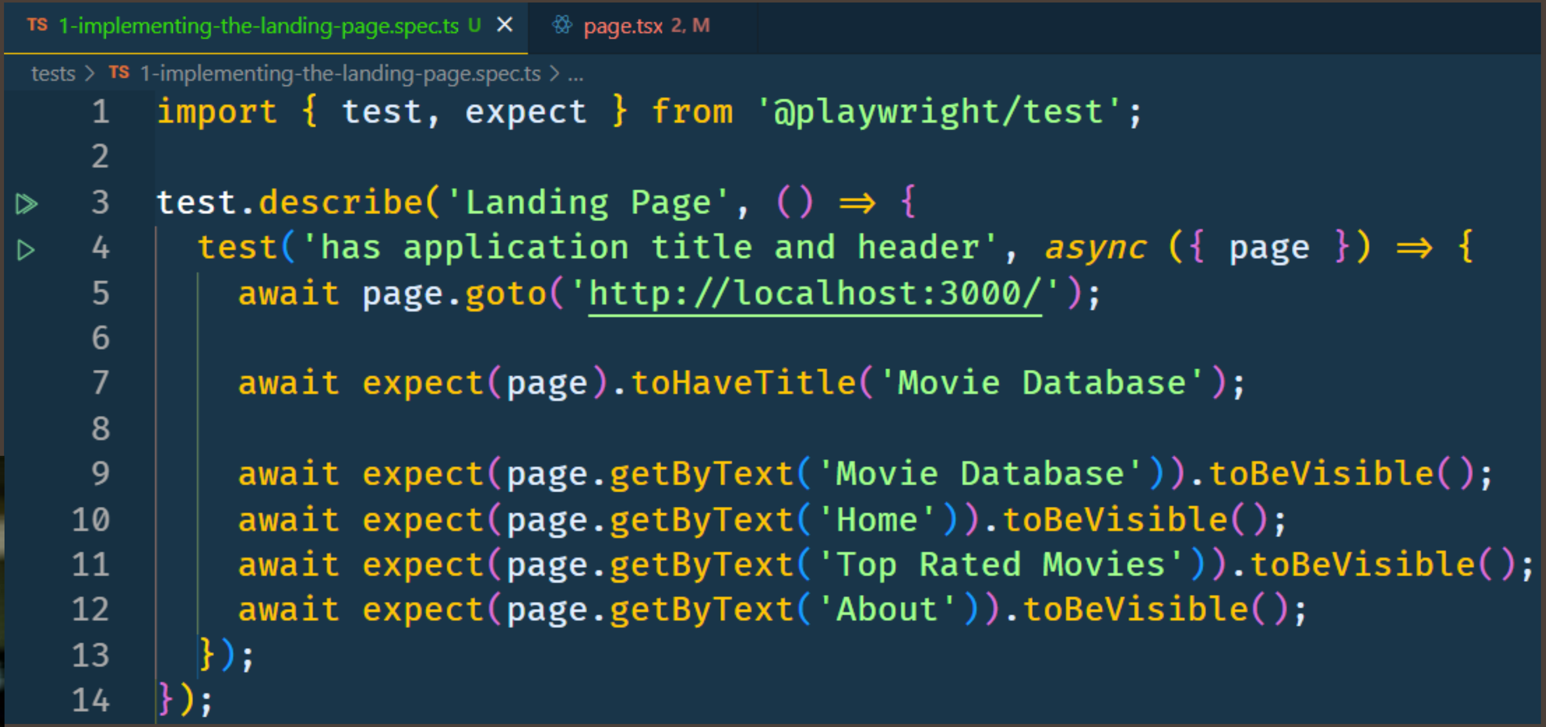
Implementing Landing Page

"As a movie enthusiast

I want to see a welcoming landing page

So that I can understand what the application offers and navigate to different sections"

Landing Page - Header Section

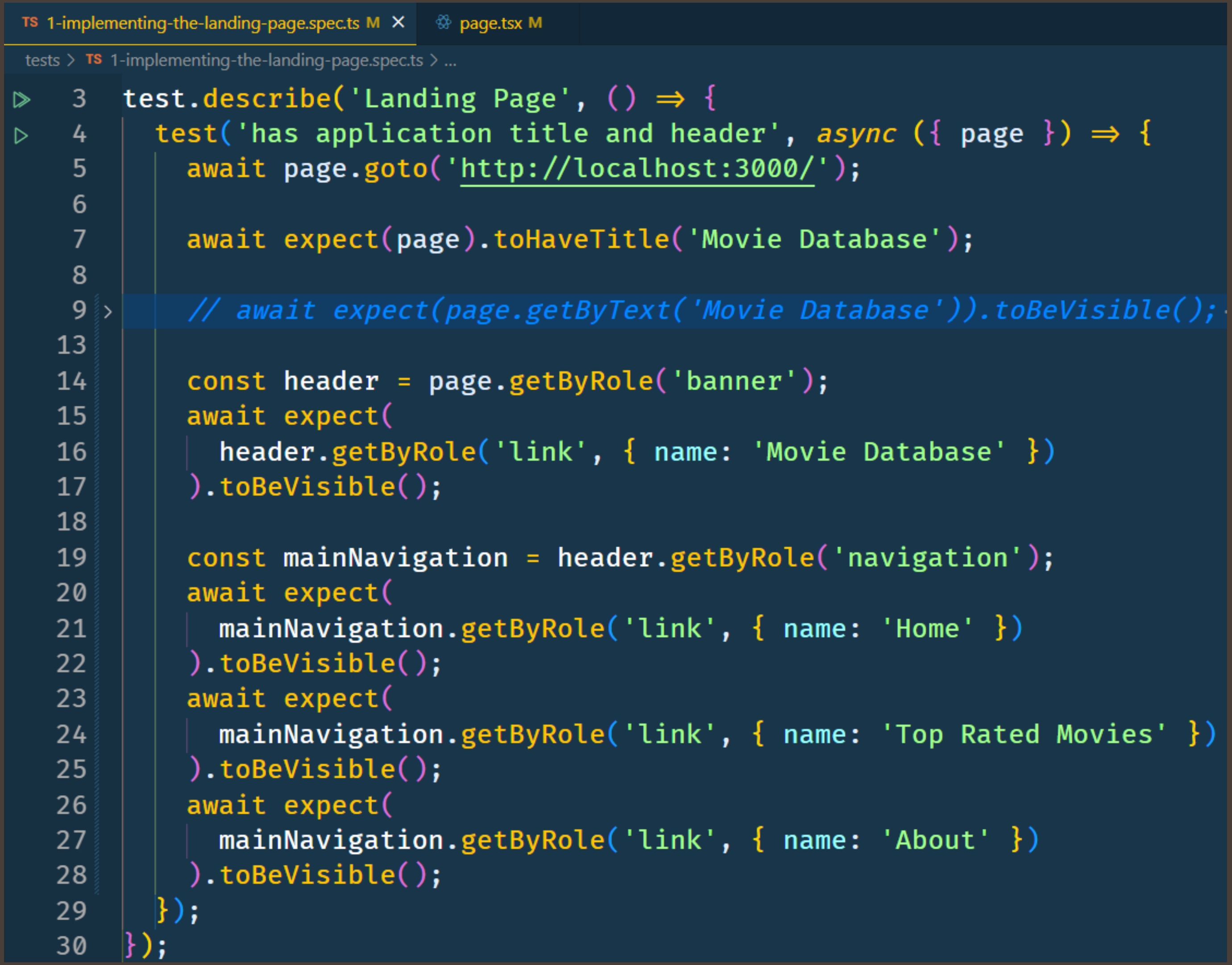

TS 1-implementing-the-landing-page.spec.ts U
page.tsx 2, M
tests > TS 1-implementing-the-landing-page.spec.ts > ...

```
import { test, expect } from '@playwright/test';

test.describe('Landing Page', () => {
  test('has application title and header', async ({ page }) => {
    await page.goto('http://localhost:3000/');

    await expect(page).toHaveTitle('Movie Database');

    await expect(page.getByText('Movie Database')).toBeVisible();
    await expect(page.getByText('Home')).toBeVisible();
    await expect(page.getByText('Top Rated Movies')).toBeVisible();
    await expect(page.getByText('About')).toBeVisible();
  });
});
```

Best Practices with Playwright

- Test **user visible** behavior
- **Prefer user-facing attributes** to XPath or CSS selectors
  - **page.getByRole()** to locate by explicit and implicit accessibility attributes.
  - **page.getByLabel()** to locate a form control by associated label's text.
  - **page.getByPlaceholder()** to locate an input by placeholder.
  - **page.getByText()** to locate by text content.
- Use **web first assertions**
  - Playwright will wait until the expected condition is met

Landing Page - Header Section with links

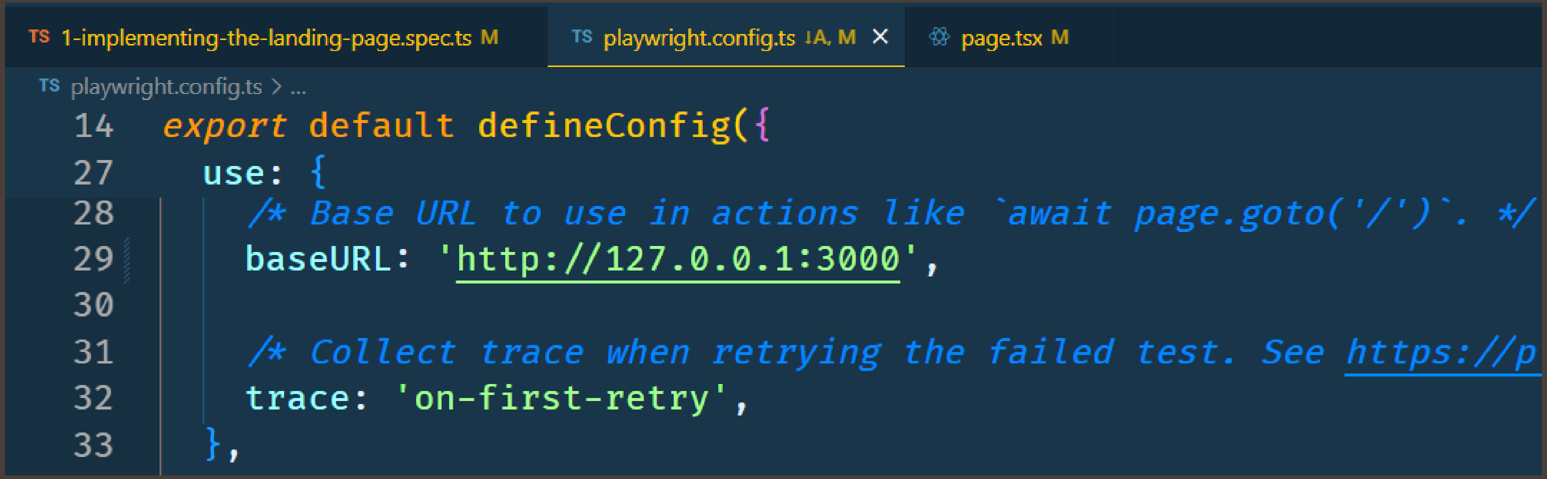

1-implementing-the-landing-page.spec.ts M
page.tsx M
tests > 1-implementing-the-landing-page.spec.ts > ...

```
test.describe('Landing Page', () => {
  test('has application title and header', async ({ page }) => {
    await page.goto('http://localhost:3000/');

    await expect(page).toHaveTitle('Movie Database');

    // await expect(page.getByText('Movie Database')).toBeVisible();

    const header = page.getByRole('banner');
    await expect(
      header.getByRole('link', { name: 'Movie Database' })
    ).toBeVisible();

    const mainNavigation = header.getByRole('navigation');
    await expect(
      mainNavigation.getByRole('link', { name: 'Home' })
    ).toBeVisible();
    await expect(
      mainNavigation.getByRole('link', { name: 'Top Rated Movies' })
    ).toBeVisible();
    await expect(
      mainNavigation.getByRole('link', { name: 'About' })
    ).toBeVisible();
  });
});
```

Playwright configuration

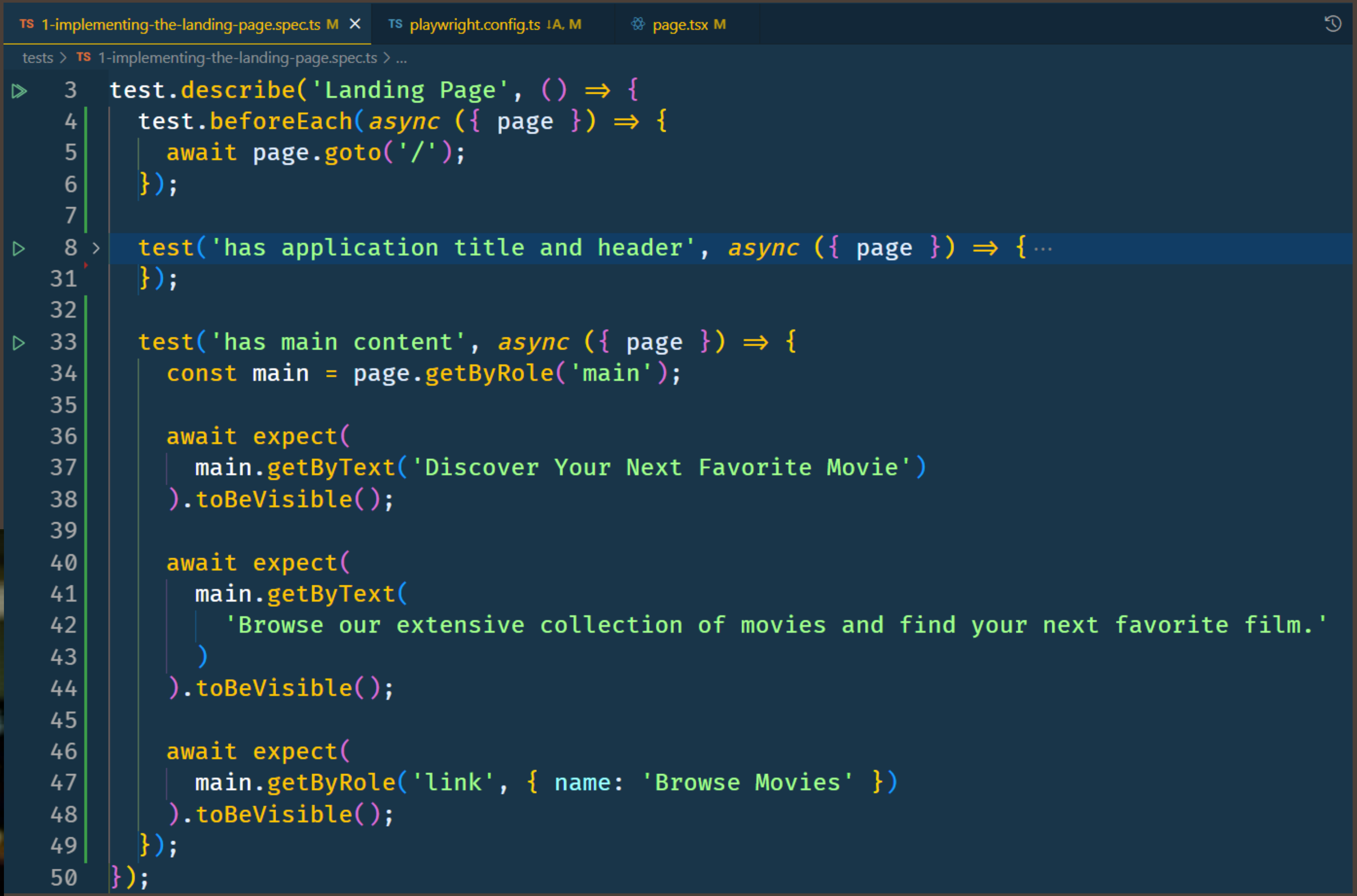
- The **Playwright configuration** can prevent some repeated code
- And make it easier to **update settings**
  - For example when running against a preview environment in the CI
  - ***`baseURL: process.env.PLAYWRIGHT_TEST_BASE_URL ?? 'http://localhost:3000’`***
- Group related tests
  - **`test.describe()`**
- Use the **test hooks** that are executed before and after tests
  - ***`test.beforeEach()`***, ***`test.afterEach()`***
  - ***`test.beforeAll()`***, ***`test.afterAll()`***

playwright.
config.ts


TS 1-implementing-the-landing-page.spec.ts M
TS playwright.config.ts 1A, M
page.tsx M
TS playwright.config.ts > ...

```
export default defineConfig({
  use: {
    /* Base URL to use in actions like `await page.goto('/')`. */
    baseURL: 'http://127.0.0.1:3000',

    /* Collect trace when retrying the failed test. See https://p
    trace: 'on-first-retry',
  },
```

Landing Page - Main Content



1-implementing-the-landing-page.spec.ts M
playwright.config.ts A, M
page.tsx M
tests > 1-implementing-the-landing-page.spec.ts > ...

```
test.describe('Landing Page', () => {
  test.beforeEach(async ({ page }) => {
    await page.goto('/');
  });

  test('has application title and header', async ({ page }) => {···
  });

  test('has main content', async ({ page }) => {
    const main = page.getByRole('main');

    await expect(
      main.getByText('Discover Your Next Favorite Movie')
    ).toBeVisible();

    await expect(
      main.getByText(
        'Browse our extensive collection of movies and find your next favorite film.'
      )
    ).toBeVisible();

    await expect(
      main.getByRole('link', { name: 'Browse Movies' })
    ).toBeVisible();
  });
});
```

Break time

Implementing the Movie List

Implementing the Movie List

"As a movie enthusiast

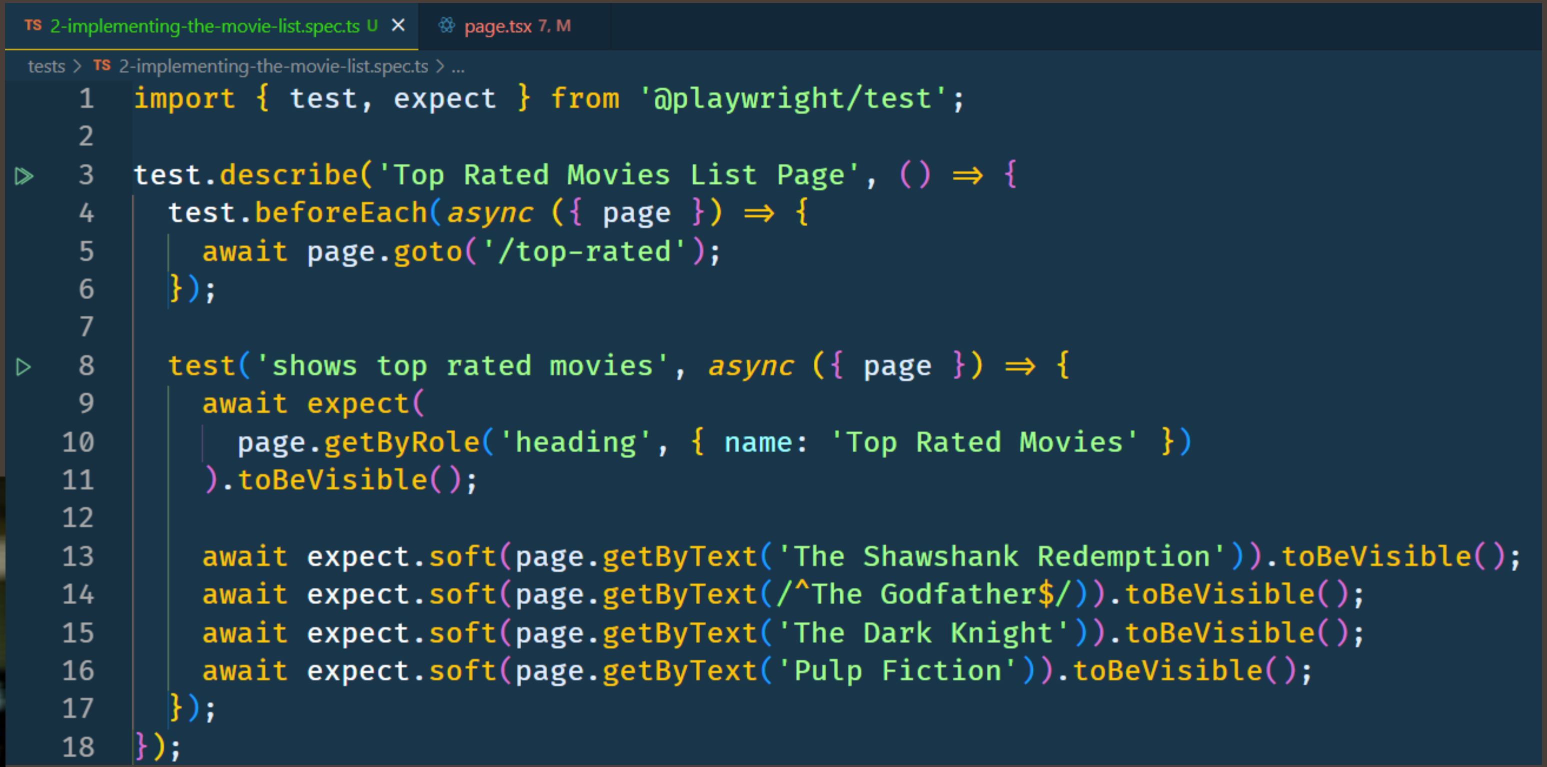
I want to browse through a list of movies

So that I can discover new films and see their ratings"

Playwright test failure

- A Playwright doesn’t need to **stop at the first failure**
  - Use ***`expect.soft()`*** to keep going after a failed expectation

Movies List - Basic Movie List


2-implementing-the-movie-list.spec.ts U
page.tsx 7, M
tests > 2-implementing-the-movie-list.spec.ts > ...

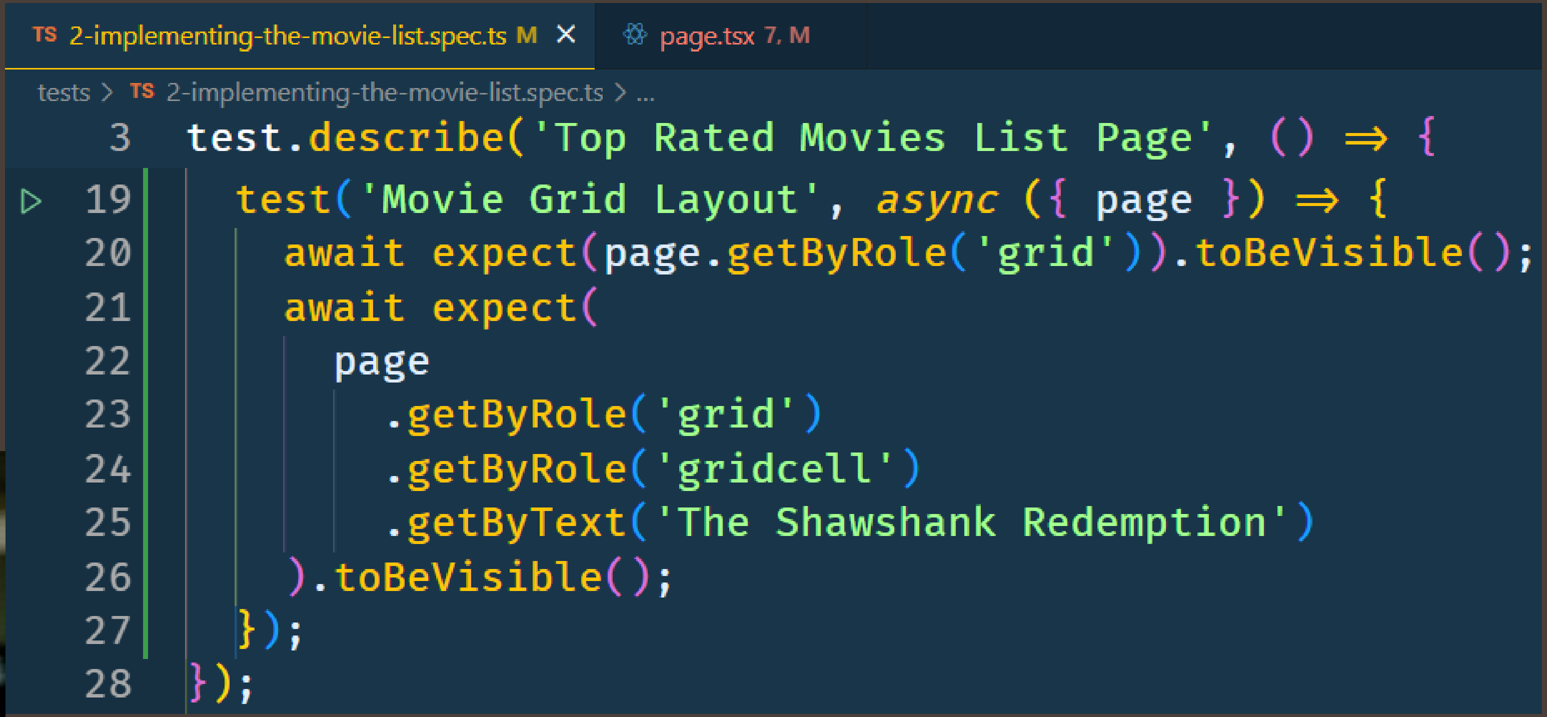
```typescript
import { test, expect } from '@playwright/test';

test.describe('Top Rated Movies List Page', () => {
  test.beforeEach(async ({ page }) => {
    await page.goto('/top-rated');
  });

  test('shows top rated movies', async ({ page }) => {
    await expect(
      page.getByRole('heading', { name: 'Top Rated Movies' })
    ).toBeVisible();

    await expect.soft(page.getByText('The Shawshank Redemption')).toBeVisible();
    await expect.soft(page.getByText(/^The Godfather$/)).toBeVisible();
    await expect.soft(page.getByText('The Dark Knight')).toBeVisible();
    await expect.soft(page.getByText('Pulp Fiction')).toBeVisible();
  });
});
```

Movies List - Grid Layout

"As a movie enthusiast

I want to see the movies in a responsive grid"

Movies List - Grid Layout

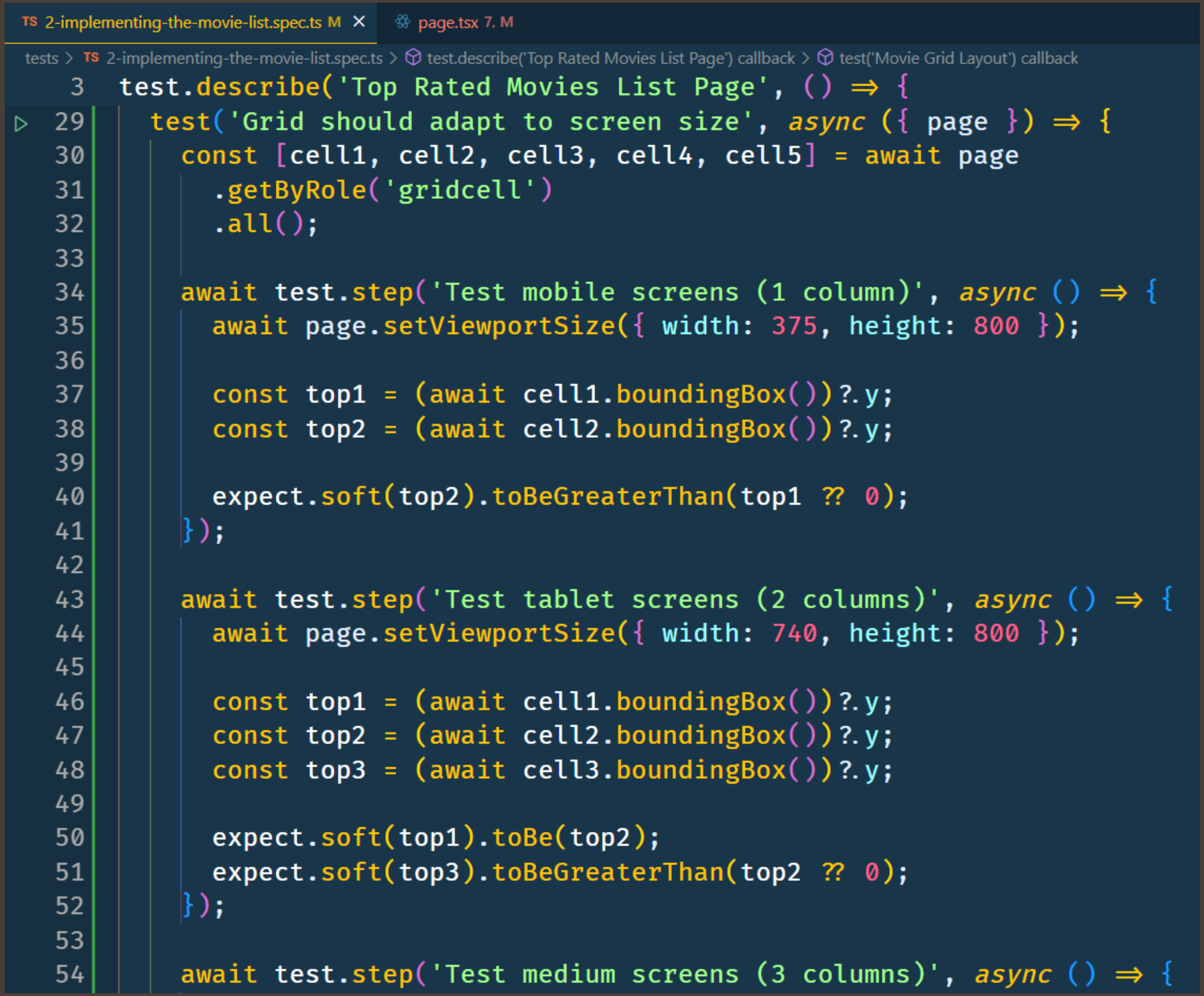

2-implementing-the-movie-list.spec.ts M
page.tsx 7, M
tests > 2-implementing-the-movie-list.spec.ts > ...

```
test.describe('Top Rated Movies List Page', () => {
  test('Movie Grid Layout', async ({ page }) => {
    await expect(page.getByRole('grid')).toBeVisible();
    await expect(
      page
        .getByRole('grid')
        .getByRole('gridcell')
        .getByText('The Shawshank Redemption')
    ).toBeVisible();
  });
});
```

Playwright test size

- **Favor a few larger tests** over many small ones
  - Break larger tests into steps with ***`test.step()`***

Movies List - Responsive Grid



TS 2-implementing-the-movie-list.spec.ts M ×   page.tsx 7, M

tests > TS 2-implementing-the-movie-list.spec.ts > test.describe('Top Rated Movies List Page') callback > test('Movie Grid Layout') callback

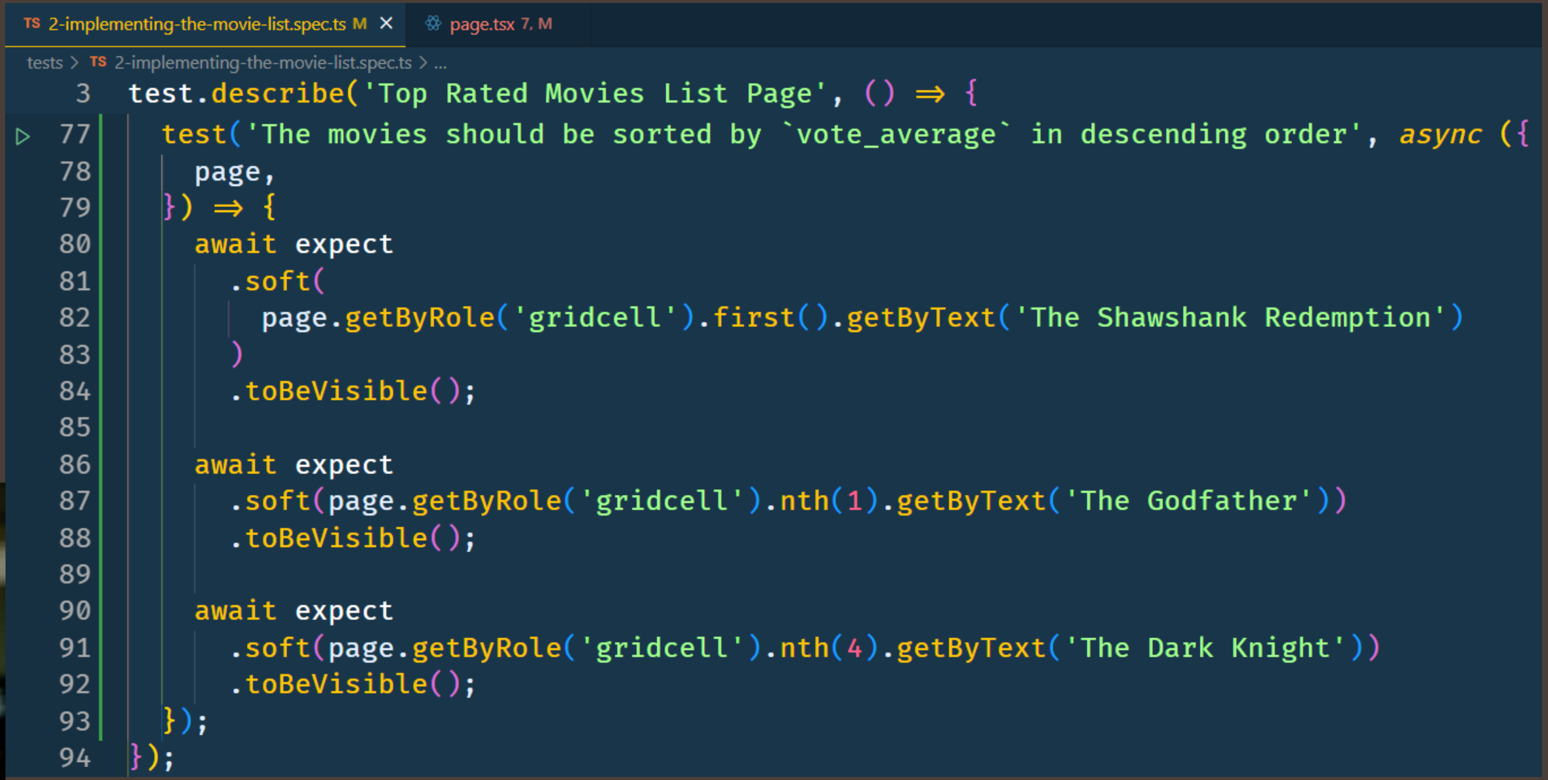
```
test.describe('Top Rated Movies List Page', () ⇒ {
  test('Grid should adapt to screen size', async ({ page }) ⇒ {
    const [cell1, cell2, cell3, cell4, cell5] = await page
      .getByRole('gridcell')
      .all();

    await test.step('Test mobile screens (1 column)', async () ⇒ {
      await page.setViewportSize({ width: 375, height: 800 });

      const top1 = (await cell1.boundingBox())?.y;
      const top2 = (await cell2.boundingBox())?.y;

      expect.soft(top2).toBeGreaterThan(top1 ?? 0);
    });

    await test.step('Test tablet screens (2 columns)', async () ⇒ {
      await page.setViewportSize({ width: 740, height: 800 });

      const top1 = (await cell1.boundingBox())?.y;
      const top2 = (await cell2.boundingBox())?.y;
      const top3 = (await cell3.boundingBox())?.y;

      expect.soft(top1).toBe(top2);
      expect.soft(top3).toBeGreaterThan(top2 ?? 0);
    });

    await test.step('Test medium screens (3 columns)', async () ⇒ {
```

Movies List - Sorted by vote

"As a movie enthusiast

I want to see the movies sorted by vote average in descending order"

Movies List - Sorted by vote



TS 2-implementing-the-movie-list.spec.ts M × | page.tsx 7, M

tests > TS 2-implementing-the-movie-list.spec.ts > ...

```ts
test.describe('Top Rated Movies List Page', () ⇒ {
  test('The movies should be sorted by `vote_average` in descending order', async ({
    page,
  }) ⇒ {
    await expect
      .soft(
        page.getByRole('gridcell').first().getByText('The Shawshank Redemption')
      )
      .toBeVisible();

    await expect
      .soft(page.getByRole('gridcell').nth(1).getByText('The Godfather'))
      .toBeVisible();

    await expect
      .soft(page.getByRole('gridcell').nth(4).getByText('The Dark Knight'))
      .toBeVisible();
  });
});
```

Movies List - Card Component

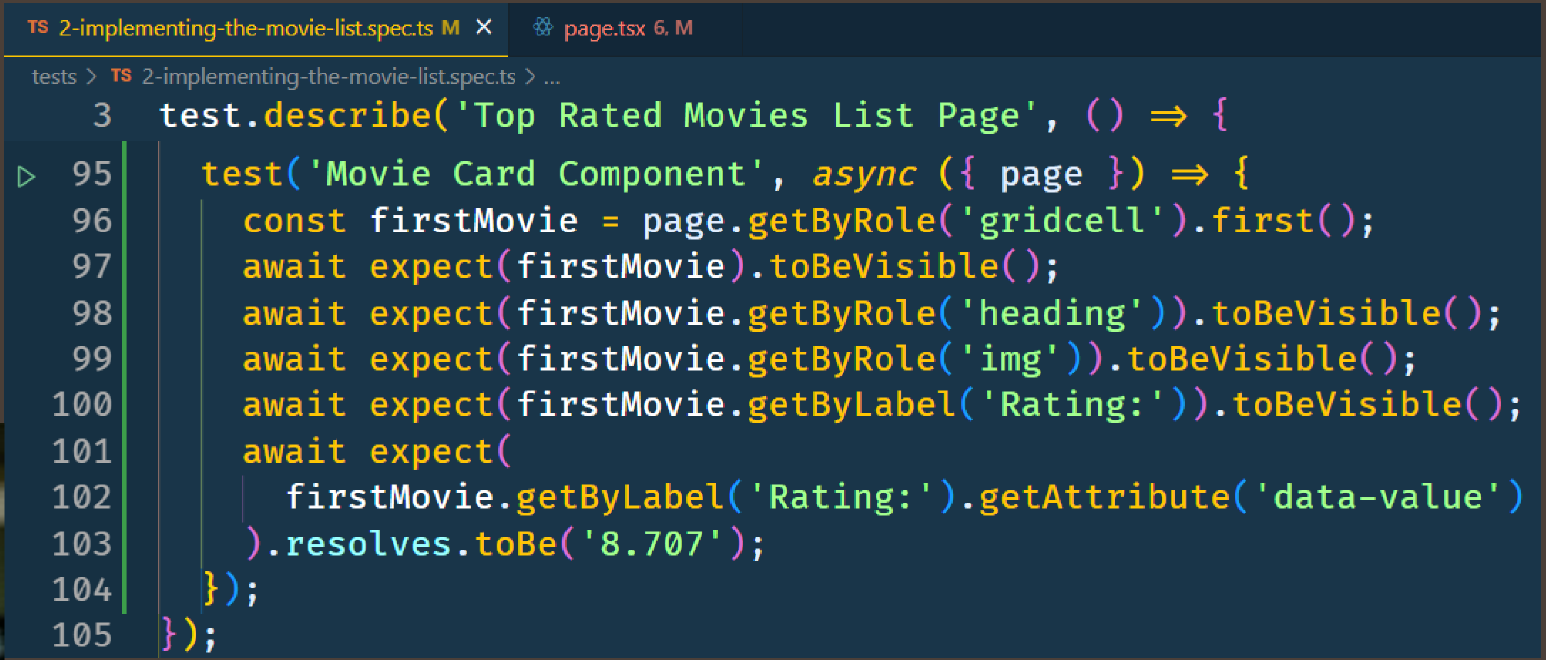
"As a movie enthusiast

I want to see each movie in a card with title, poster, rating and description"

Adding test helpers

- Use accessibility options to make elements easier to find
  - Like ***`aria-label`*** and ***`page.getByLabel()`***
  - Only use ***`id`*** or ***`data-testid`*** as a last resort
- Use ***`data-value`*** to add values in an unformatted format
  - But only if a value isn’t easy to read from the DOM

Movies List - Card Component



2-implementing-the-movie-list.spec.ts M
page.tsx 6, M
tests > 2-implementing-the-movie-list.spec.ts > ...

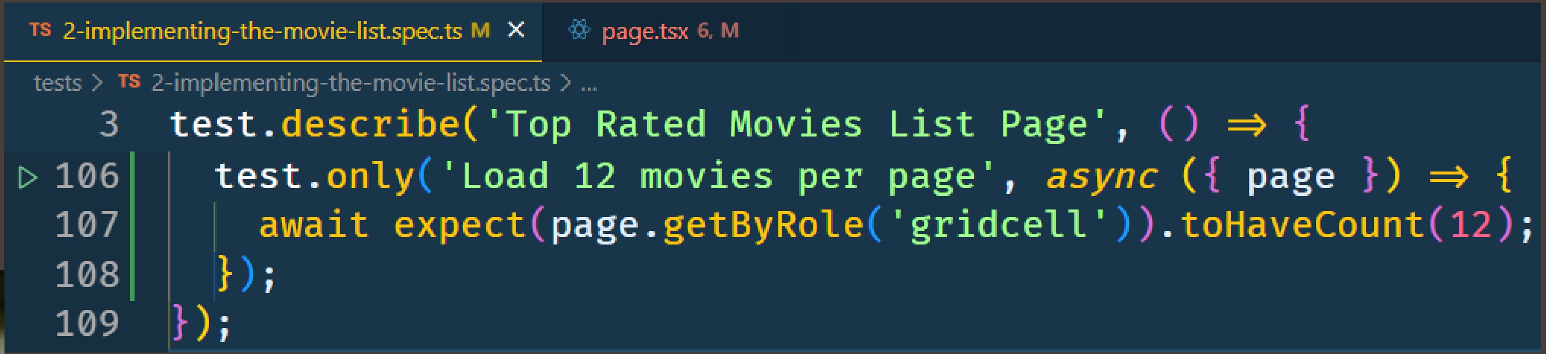
```typescript
test.describe('Top Rated Movies List Page', () => {
  test('Movie Card Component', async ({ page }) => {
    const firstMovie = page.getByRole('gridcell').first();
    await expect(firstMovie).toBeVisible();
    await expect(firstMovie.getByRole('heading')).toBeVisible();
    await expect(firstMovie.getByRole('img')).toBeVisible();
    await expect(firstMovie.getByLabel('Rating:')).toBeVisible();
    await expect(
      firstMovie.getByLabel('Rating:').getAttribute('data-value')
    ).resolves.toBe('8.707');
  });
});
```

Movies List - 12 Movies per page

"As a movie enthusiast

I want to see each 12 movie cards at the time"

Movies List - 12 Movies per pa


TS 2-implementing-the-movie-list.spec.ts M   page.tsx 6, M

tests > TS 2-implementing-the-movie-list.spec.ts > ...

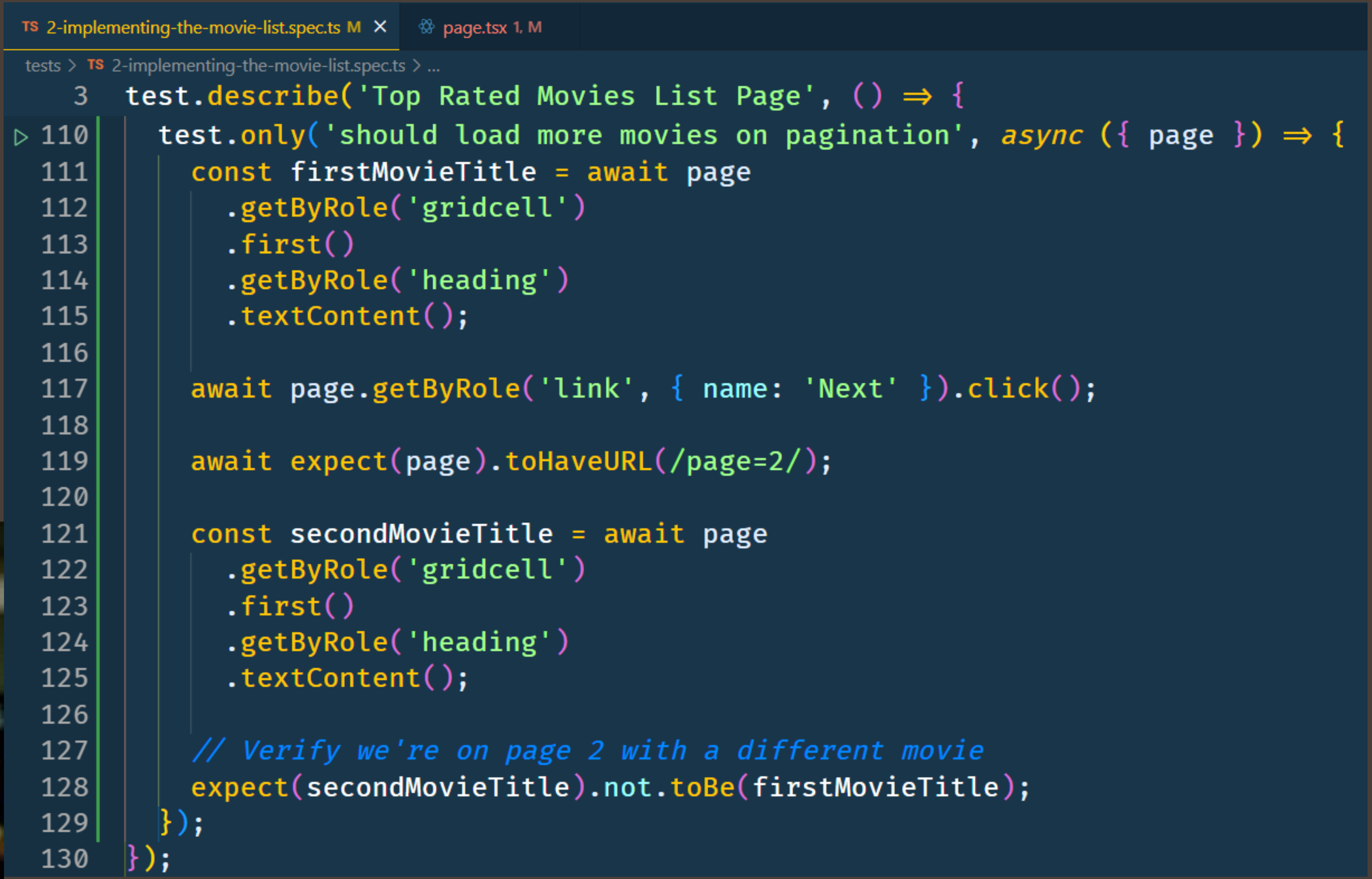
```
test.describe('Top Rated Movies List Page', () => {
  test.only('Load 12 movies per page', async ({ page }) => {
    await expect(page.getByRole('gridcell')).toHaveCount(12);
  });
});
```

Movies List - Pagination

"As a movie enthusiast

*I want to be able to click a **Next** button and see more movies"*

Movies List - Pagination



TS 2-implementing-the-movie-list.spec.ts M ×
page.tsx 1, M
tests > TS 2-implementing-the-movie-list.spec.ts > ...

```
test.describe('Top Rated Movies List Page', () => {
  test.only('should load more movies on pagination', async ({ page }) => {
    const firstMovieTitle = await page
      .getByRole('gridcell')
      .first()
      .getByRole('heading')
      .textContent();

    await page.getByRole('link', { name: 'Next' }).click();

    await expect(page).toHaveURL(/page=2/);

    const secondMovieTitle = await page
      .getByRole('gridcell')
      .first()
      .getByRole('heading')
      .textContent();

    // Verify we're on page 2 with a different movie
    expect(secondMovieTitle).not.toBe(firstMovieTitle);
  });
});
```

Implementing the Navigation Menu

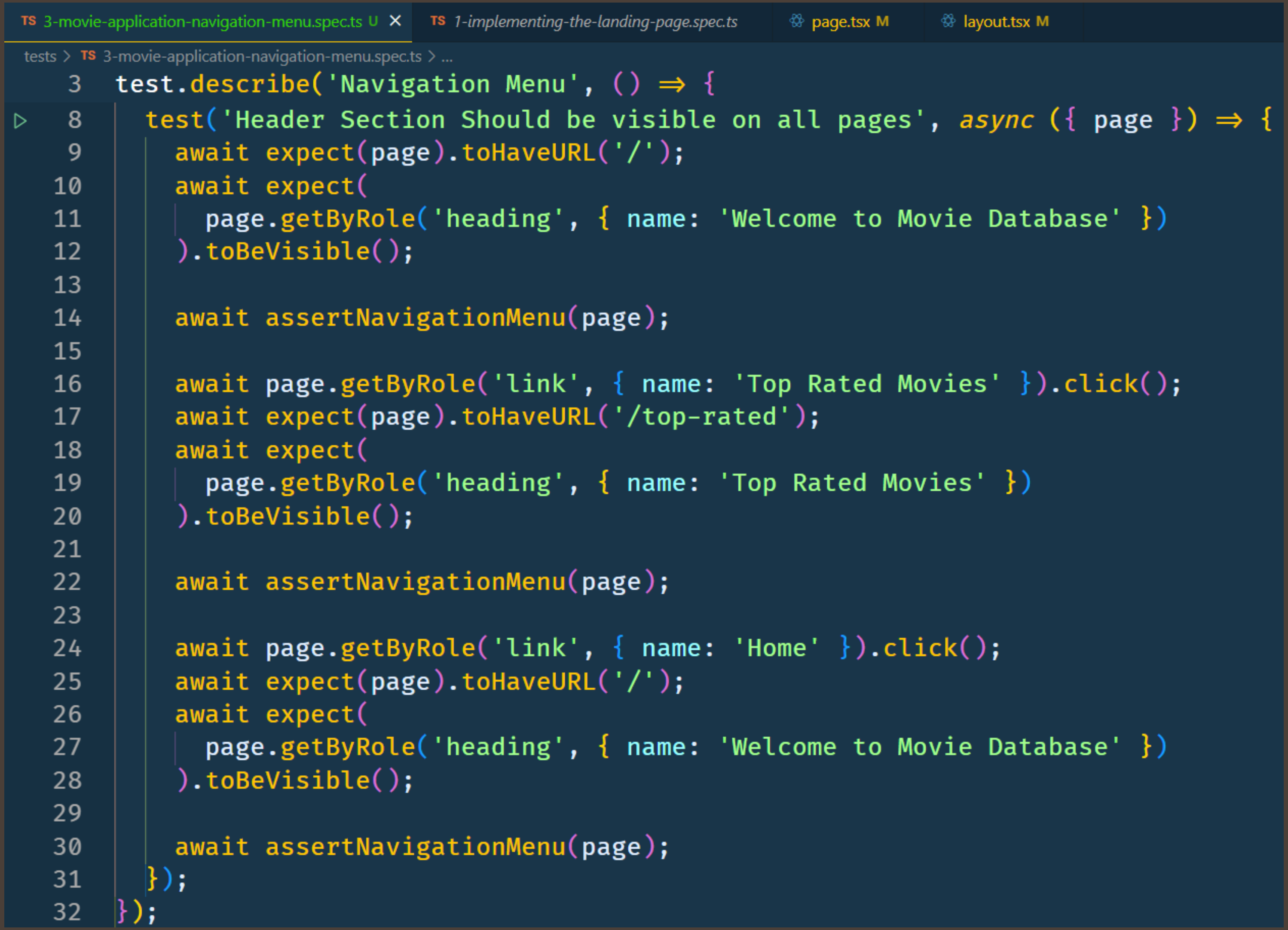
Implementing the Navigation Menu

"As a user of the movie application

I want to have a consistent navigation menu

So that I can easily access different sections of the application"

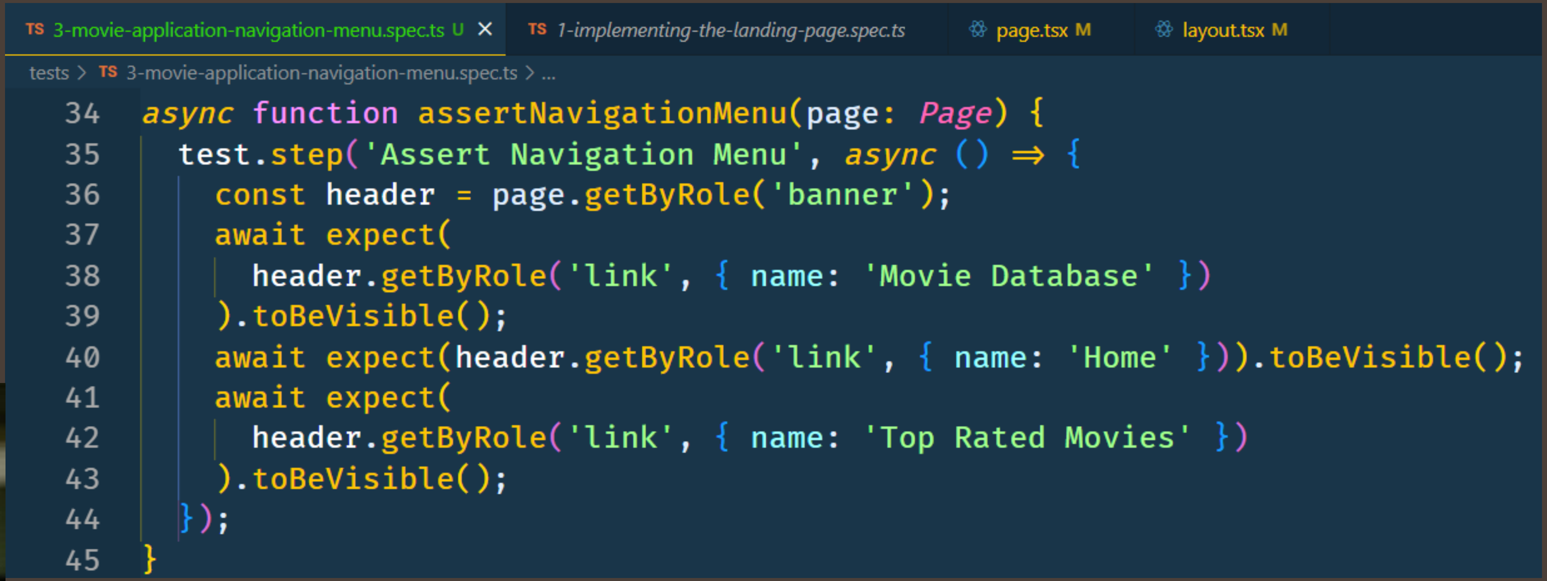
Navigation Menu



TS 3-movie-application-navigation-menu.spec.ts U ×
TS 1-implementing-the-landing-page.spec.ts
page.tsx M
layout.tsx M
tests > TS 3-movie-application-navigation-menu.spec.ts > ...

```typescript
test.describe('Navigation Menu', () => {
  test('Header Section Should be visible on all pages', async ({ page }) => {
    await expect(page).toHaveURL('/');
    await expect(
      page.getByRole('heading', { name: 'Welcome to Movie Database' })
    ).toBeVisible();

    await assertNavigationMenu(page);

    await page.getByRole('link', { name: 'Top Rated Movies' }).click();
    await expect(page).toHaveURL('/top-rated');
    await expect(
      page.getByRole('heading', { name: 'Top Rated Movies' })
    ).toBeVisible();

    await assertNavigationMenu(page);

    await page.getByRole('link', { name: 'Home' }).click();
    await expect(page).toHaveURL('/');
    await expect(
      page.getByRole('heading', { name: 'Welcome to Movie Database' })
    ).toBeVisible();

    await assertNavigationMenu(page);
  });
});
```

Navigation Menu


TS 3-movie-application-navigation-menu.spec.ts U
TS 1-implementing-the-landing-page.spec.ts
page.tsx M
layout.tsx M
tests > TS 3-movie-application-navigation-menu.spec.ts > ...

```
async function assertNavigationMenu(page: Page) {
  test.step('Assert Navigation Menu', async () => {
    const header = page.getByRole('banner');
    await expect(
      header.getByRole('link', { name: 'Movie Database' })
    ).toBeVisible();
    await expect(header.getByRole('link', { name: 'Home' })).toBeVisible();
    await expect(
      header.getByRole('link', { name: 'Top Rated Movies' })
    ).toBeVisible();
  });
}
```

Implementing the Movie Details Page

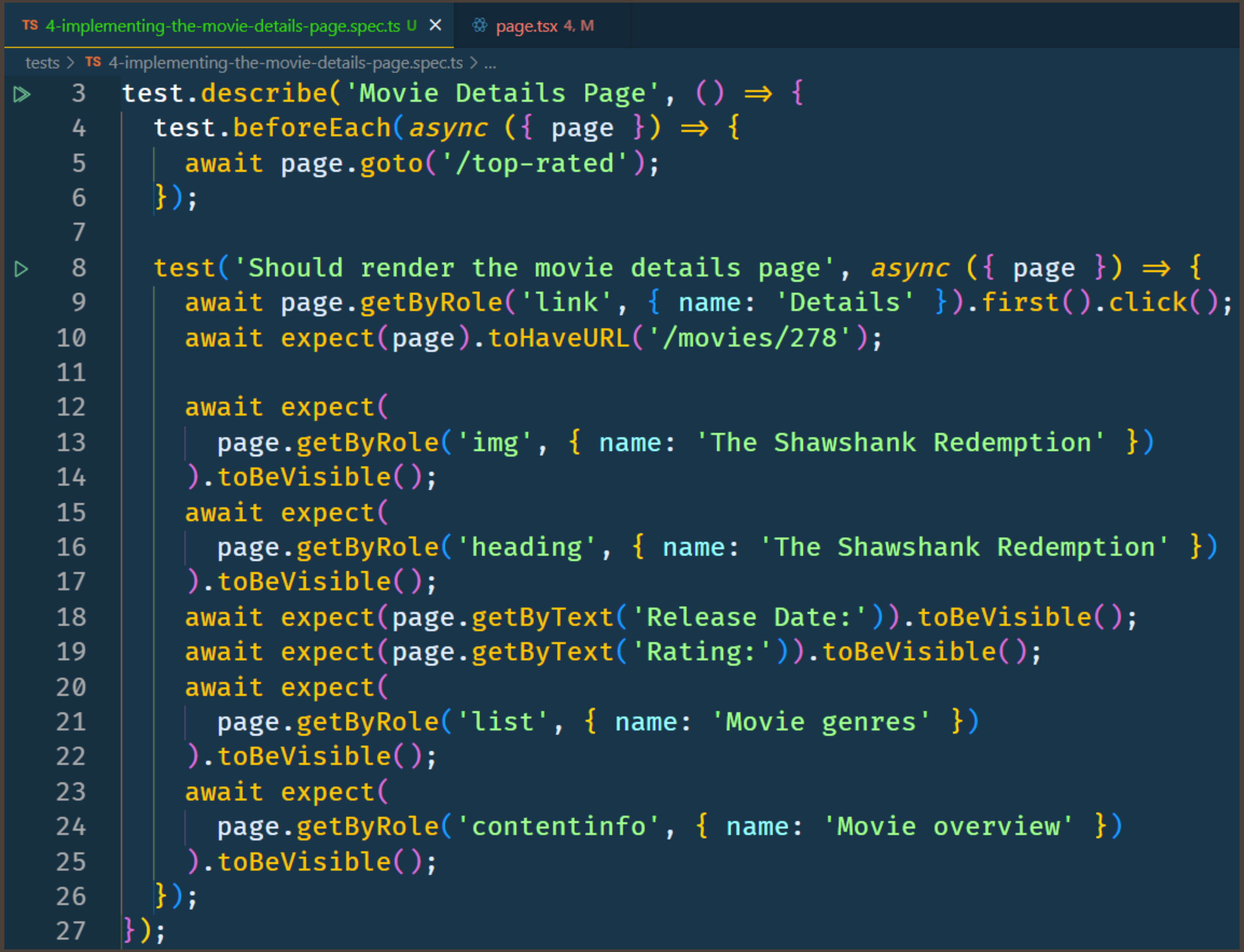
Implementing the Movie Details Page

"As a movie enthusiast using the application

I want to view comprehensive details about a specific movie

So that I can make informed decisions about watching it and learn more about the film"

Movie Details - Key Information

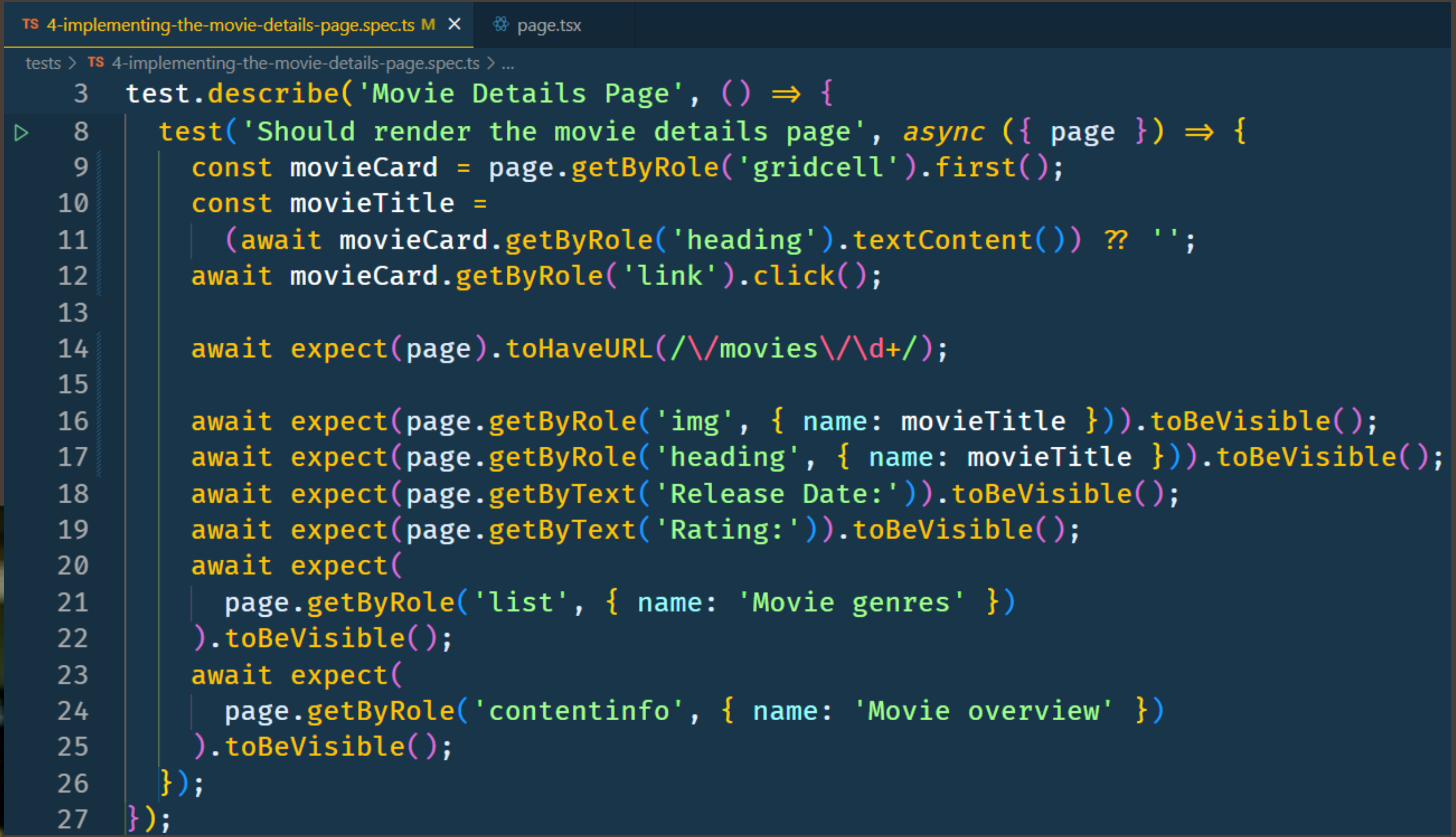

4-implementing-the-movie-details-page.spec.ts U
page.tsx 4, M
tests > 4-implementing-the-movie-details-page.spec.ts > ...

```
test.describe('Movie Details Page', () => {
  test.beforeEach(async ({ page }) => {
    await page.goto('/top-rated');
  });

  test('Should render the movie details page', async ({ page }) => {
    await page.getByRole('link', { name: 'Details' }).first().click();
    await expect(page).toHaveURL('/movies/278');

    await expect(
      page.getByRole('img', { name: 'The Shawshank Redemption' })
    ).toBeVisible();
    await expect(
      page.getByRole('heading', { name: 'The Shawshank Redemption' })
    ).toBeVisible();
    await expect(page.getByText('Release Date:')).toBeVisible();
    await expect(page.getByText('Rating:')).toBeVisible();
    await expect(
      page.getByRole('list', { name: 'Movie genres' })
    ).toBeVisible();
    await expect(
      page.getByRole('contentinfo', { name: 'Movie overview' })
    ).toBeVisible();
  });
});
```

Movie Details - Key Information Improved

- Requires the **The Shawshank Redemption** to be the first movie
    - Might no longer be true in the future
- **Adapting to the data** returned can be more reliable

Movie Details - Key Information Improved


4-implementing-the-movie-details-page.spec.ts M
page.tsx
tests > 4-implementing-the-movie-details-page.spec.ts > ...

```
test.describe('Movie Details Page', () => {
  test('Should render the movie details page', async ({ page }) => {
    const movieCard = page.getByRole('gridcell').first();
    const movieTitle =
      (await movieCard.getByRole('heading').textContent()) ?? '';
    await movieCard.getByRole('link').click();

    await expect(page).toHaveURL(/\/movies\/\d+/);

    await expect(page.getByRole('img', { name: movieTitle })).toBeVisible();
    await expect(page.getByRole('heading', { name: movieTitle })).toBeVisible();
    await expect(page.getByText('Release Date:')).toBeVisible();
    await expect(page.getByText('Rating:')).toBeVisible();
    await expect(
      page.getByRole('list', { name: 'Movie genres' })
    ).toBeVisible();
    await expect(
      page.getByRole('contentinfo', { name: 'Movie overview' })
    ).toBeVisible();
  });
});
```

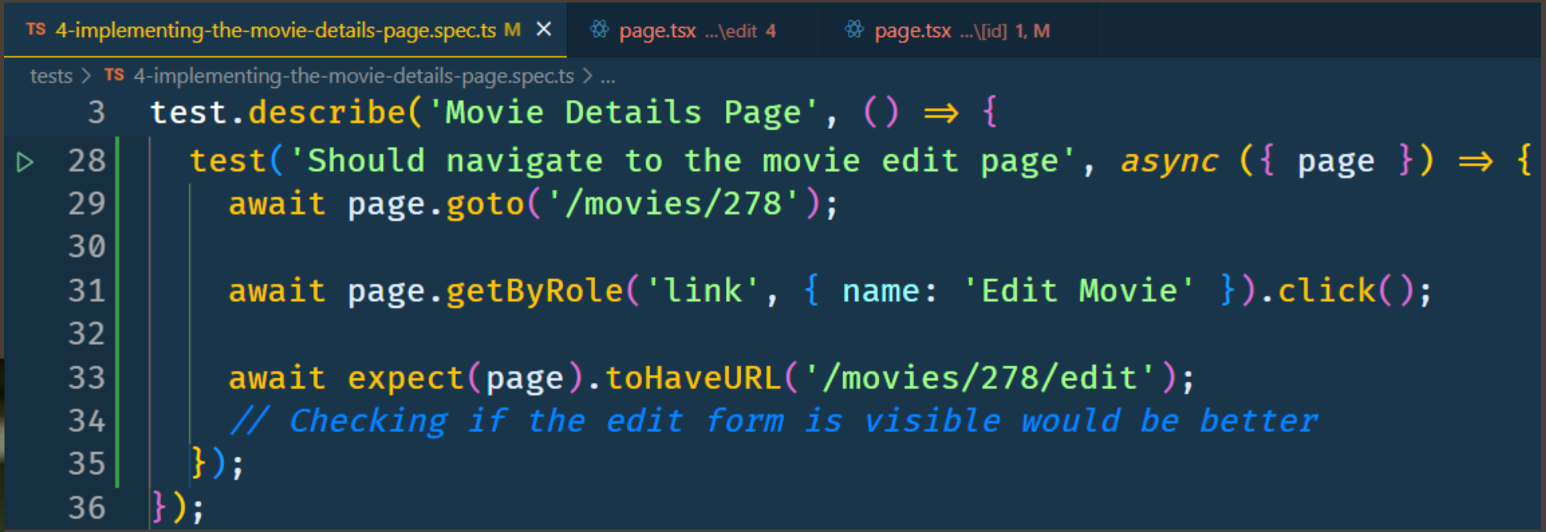
Movie Details - Interaction

"As an administrator of the movie application

I want to be able to edit a movie in the database

So that I can maintain accurate and up-to-date movie details"

Movie Details - Interaction



TS 4-implementing-the-movie-details-page.spec.ts M × | page.tsx ...\edit 4 | page.tsx ...\[id] 1, M

tests > TS 4-implementing-the-movie-details-page.spec.ts > ...

```
test.describe('Movie Details Page', () => {
  test('Should navigate to the movie edit page', async ({ page }) => {
    await page.goto('/movies/278');

    await page.getByRole('link', { name: 'Edit Movie' }).click();

    await expect(page).toHaveURL('/movies/278/edit');
    // Checking if the edit form is visible would be better
  });
});
```

Implementing the Movie Edit Page

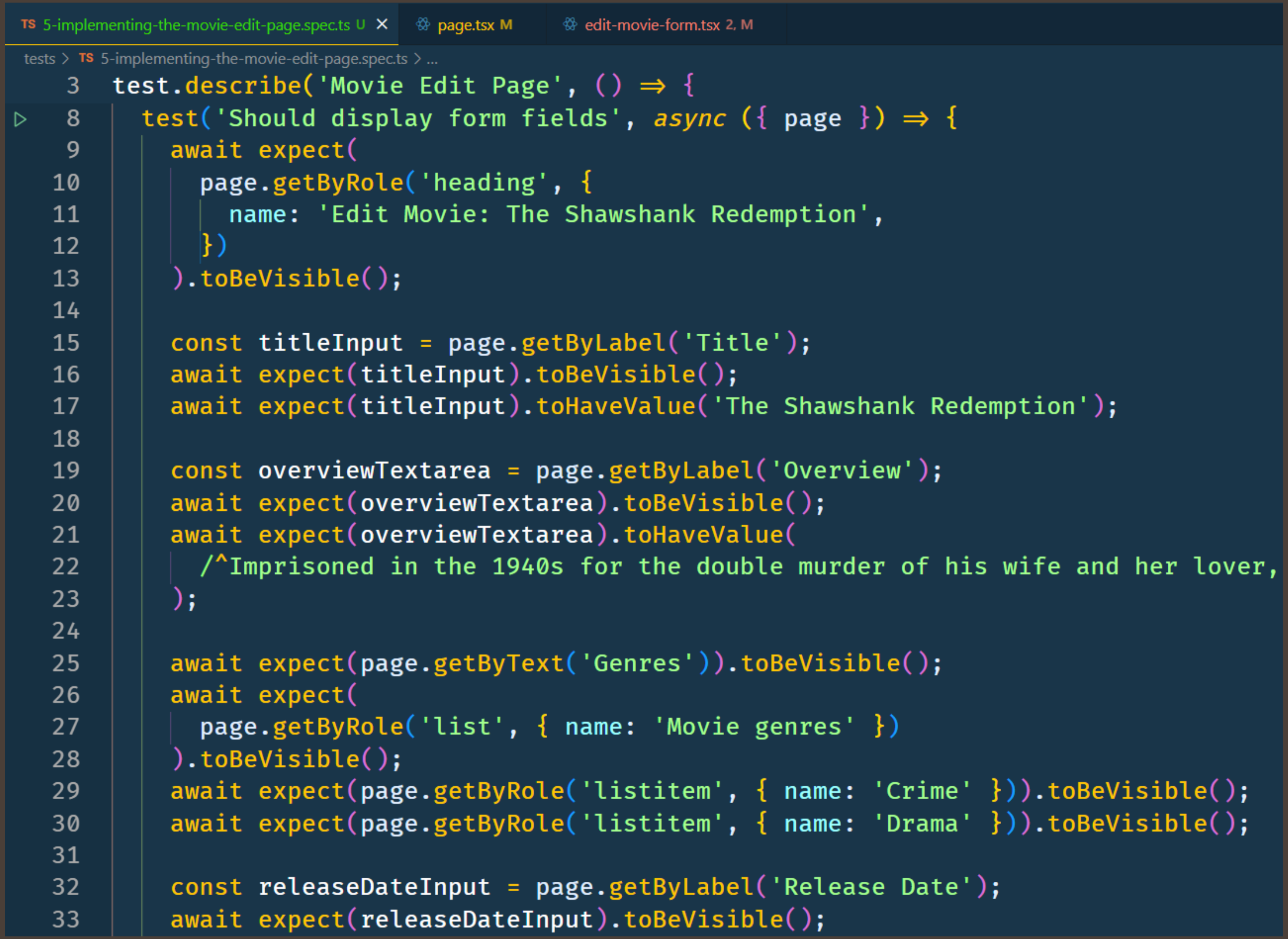
Implementing the Movie Edit Page

"As an administrator of the movie application

I want to edit existing movie information in the database

So that I can maintain accurate and up-to-date movie details"

Movie Edit - Form Fields



TS 5-implementing-the-movie-edit-page.spec.ts U × | page.tsx M | edit-movie-form.tsx 2, M

tests > TS 5-implementing-the-movie-edit-page.spec.ts > ...

```
test.describe('Movie Edit Page', () => {
  test('Should display form fields', async ({ page }) => {
    await expect(
      page.getByRole('heading', {
        name: 'Edit Movie: The Shawshank Redemption',
      })
    ).toBeVisible();

    const titleInput = page.getByLabel('Title');
    await expect(titleInput).toBeVisible();
    await expect(titleInput).toHaveValue('The Shawshank Redemption');

    const overviewTextarea = page.getByLabel('Overview');
    await expect(overviewTextarea).toBeVisible();
    await expect(overviewTextarea).toHaveValue(
      /^Imprisoned in the 1940s for the double murder of his wife and her lover,
    );

    await expect(page.getByText('Genres')).toBeVisible();
    await expect(
      page.getByRole('list', { name: 'Movie genres' })
    ).toBeVisible();
    await expect(page.getByRole('listitem', { name: 'Crime' })).toBeVisible();
    await expect(page.getByRole('listitem', { name: 'Drama' })).toBeVisible();

    const releaseDateInput = page.getByLabel('Release Date');
    await expect(releaseDateInput).toBeVisible();
```

Saving the Movie Edits

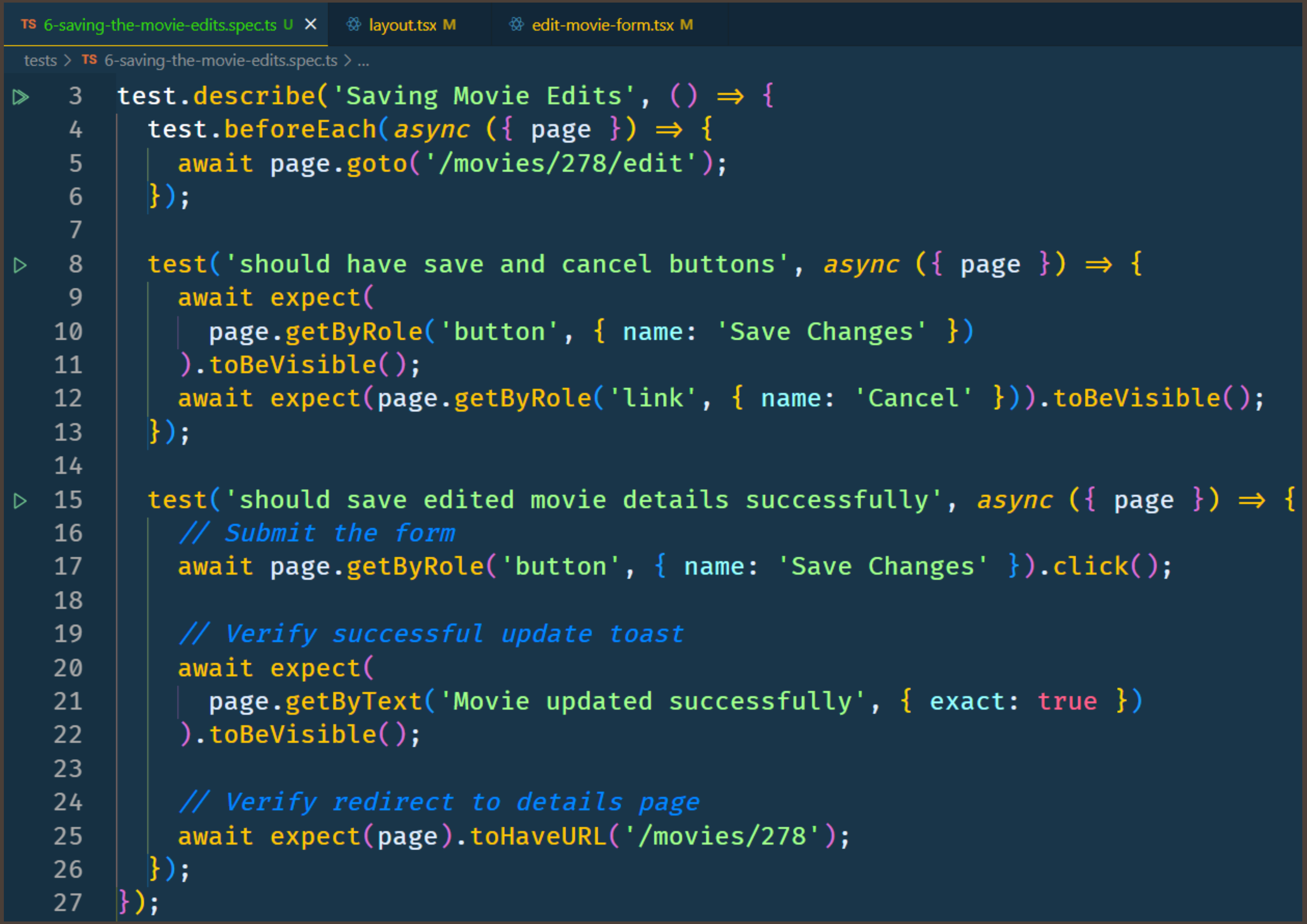
Saving the Movie Edits

"As an administrator editing movie information

I want to save my changes to the movie database

So that updated movie information is persisted and immediately available to users"

Movie Edits - Basic Saving

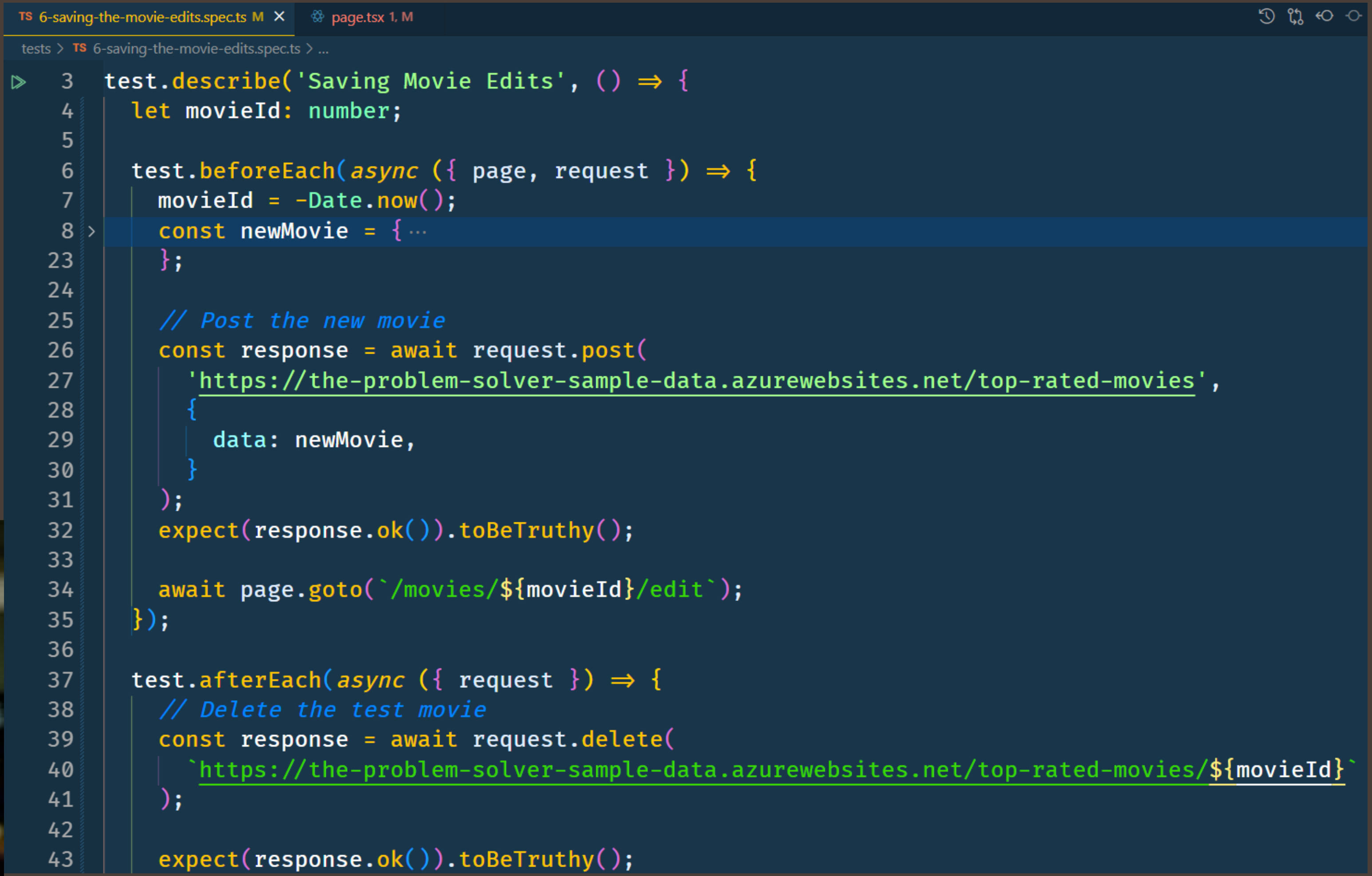

6-saving-the-movie-edits.spec.ts U
layout.tsx M
edit-movie-form.tsx M
tests > 6-saving-the-movie-edits.spec.ts > ...

```typescript
test.describe('Saving Movie Edits', () => {
  test.beforeEach(async ({ page }) => {
    await page.goto('/movies/278/edit');
  });

  test('should have save and cancel buttons', async ({ page }) => {
    await expect(
      page.getByRole('button', { name: 'Save Changes' })
    ).toBeVisible();
    await expect(page.getByRole('link', { name: 'Cancel' })).toBeVisible();
  });

  test('should save edited movie details successfully', async ({ page }) => {
    // Submit the form
    await page.getByRole('button', { name: 'Save Changes' }).click();

    // Verify successful update toast
    await expect(
      page.getByText('Movie updated successfully', { exact: true })
    ).toBeVisible();

    // Verify redirect to details page
    await expect(page).toHaveURL('/movies/278');
  });
});
```

Movie Edits - Improved Saving

- **Beware:** changing data can lead to **flaky tests**
  - Reset to the database to a known state before each test
  - Only make changes to newly added data that doesn’t show up in other tests
- Or use **Playwright network mocking**
  - Also useful to simulate and test errors like server not available

Movie Edits - Improved Saving


6-saving-the-movie-edits.spec.ts M
page.tsx 1, M
tests > 6-saving-the-movie-edits.spec.ts > ...

```
test.describe('Saving Movie Edits', () => {
  let movieId: number;

  test.beforeEach(async ({ page, request }) => {
    movieId = -Date.now();
    const newMovie = {…
    };

    // Post the new movie
    const response = await request.post(
      'https://the-problem-solver-sample-data.azurewebsites.net/top-rated-movies',
      {
        data: newMovie,
      }
    );
    expect(response.ok()).toBeTruthy();

    await page.goto(`/movies/${movieId}/edit`);
  });

  test.afterEach(async ({ request }) => {
    // Delete the test movie
    const response = await request.delete(
      `https://the-problem-solver-sample-data.azurewebsites.net/top-rated-movies/${movieId}`
    );

    expect(response.ok()).toBeTruthy();
```

Validating the Movie Edits

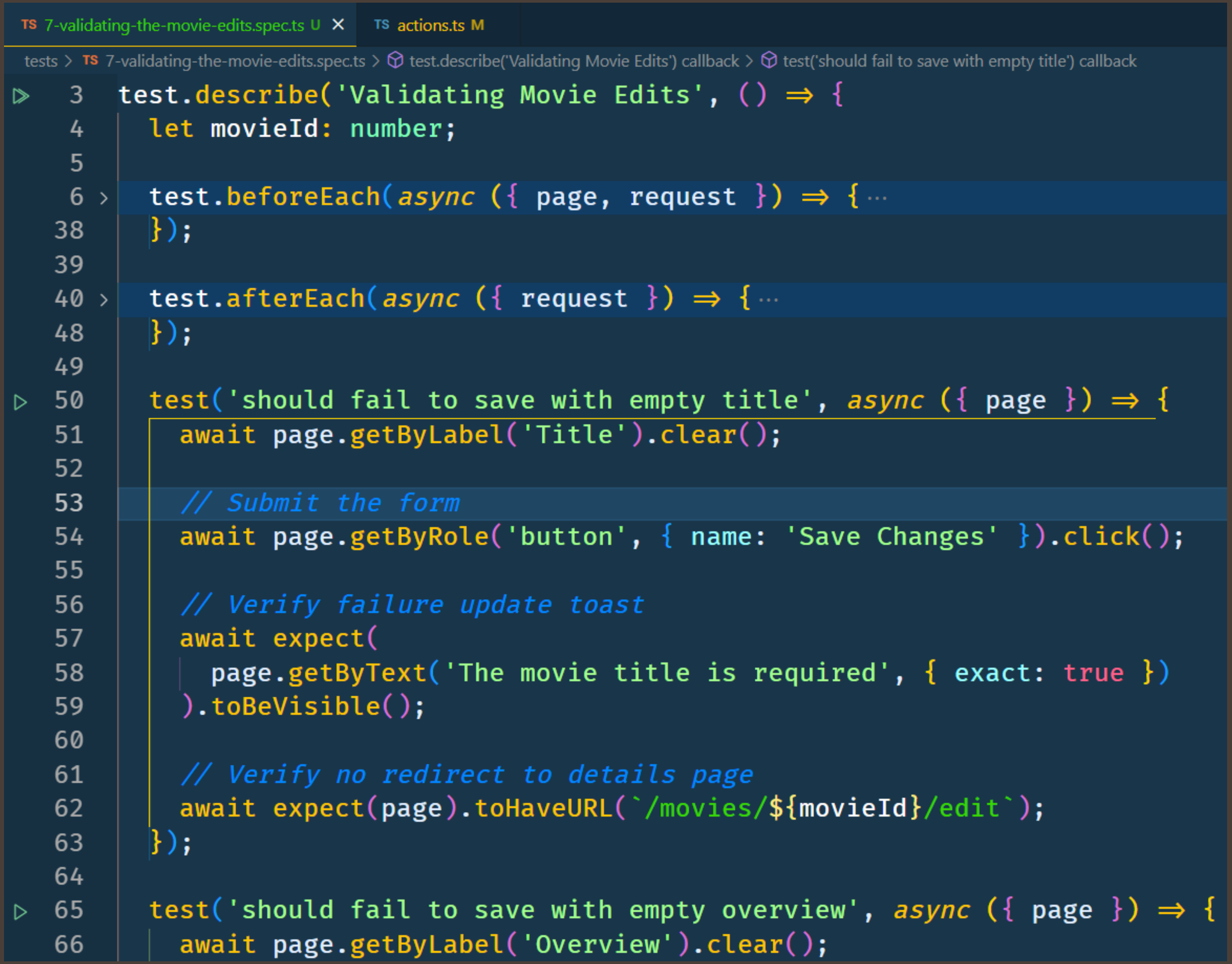
Validating the Movie Edits

"As an administrator submitting movie changes

I want feedback on the validity of my edits

So that I can correct any errors before saving to the database"

Movie Edits - Validation


7-validating-the-movie-edits.spec.ts U
actions.ts M
tests > 7-validating-the-movie-edits.spec.ts > test.describe('Validating Movie Edits') callback > test('should fail to save with empty title') callback

```
test.describe('Validating Movie Edits', () => {
  let movieId: number;

  test.beforeEach(async ({ page, request }) => {…
  });

  test.afterEach(async ({ request }) => {…
  });

  test('should fail to save with empty title', async ({ page }) => {
    await page.getByLabel('Title').clear();

    // Submit the form
    await page.getByRole('button', { name: 'Save Changes' }).click();

    // Verify failure update toast
    await expect(
      page.getByText('The movie title is required', { exact: true })
    ).toBeVisible();

    // Verify no redirect to details page
    await expect(page).toHaveURL(`/movies/${movieId}/edit`);
  });

  test('should fail to save with empty overview', async ({ page }) => {
    await page.getByLabel('Overview').clear();
```

Recommendations with Playwright

Best Practices with Playwright

- Test **Organization**
  - Group related tests
  - Use before/after hooks wisely
  - Share common setup
- Test **Reliability**
  - Use strong locators
  - Handle dynamic content
  - Consider network conditions
  - Avoid flaky tests but enable retries
- **Performance**
  - Reuse browser context when possible
  - Prefer fewer larger tests with soft asserts
  - Parallel test execution
  - Minimize unnecessary actions

Best Practices with Playwright

- Test **user visible** behavior
  - Don’t rely on things a real user doesn’t use like a class name or id
- **Prefer user-facing attributes** to XPath or CSS selectors
  - **page.getByRole()** to locate by explicit and implicit accessibility attributes.
  - **page.getByText()** to locate by text content.
  - **page.getByLabel()** to locate a form control by associated label's text.
  - **page.getByPlaceholder()** to locate an input by placeholder.
- Use **web first assertions**
  - Playwright will wait until the expected condition is met

Thank you for joining

Share your thoughts