HAUNTED edition

powered by infoSupport Solid Innovator

24/25 JAN 2025

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?

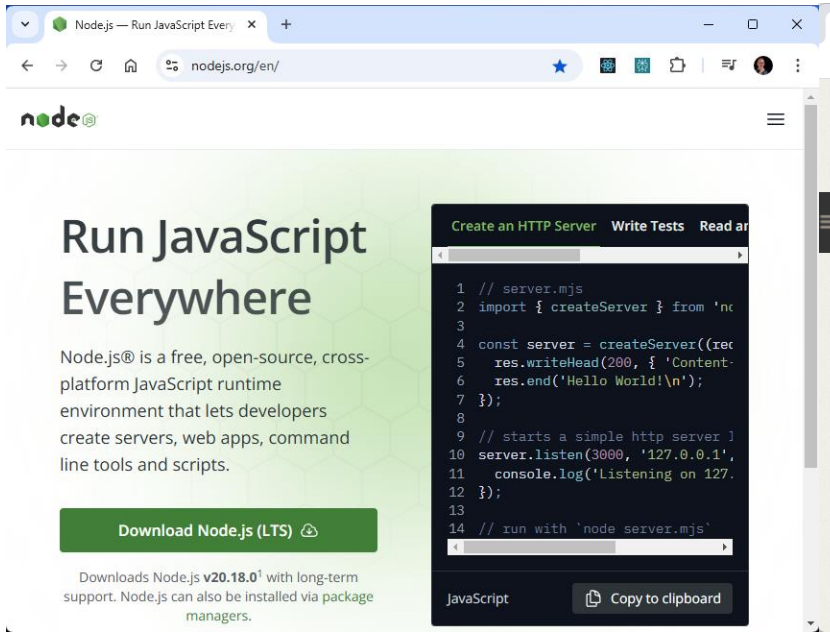
"Typing it drills it into your brain much better than simply copying and pasting it. You're forming new neuron pathways. Those pathways are going to help you in the future. Help them out now!"

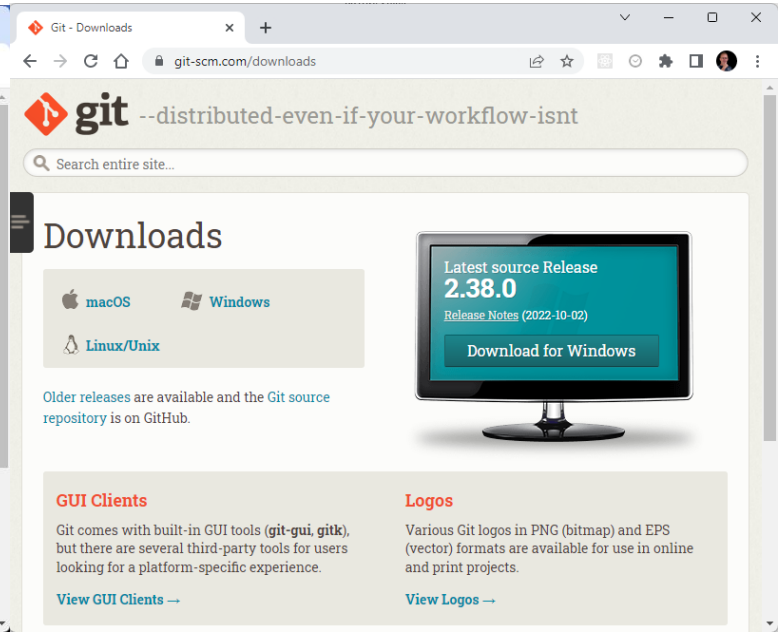
Prerequisites

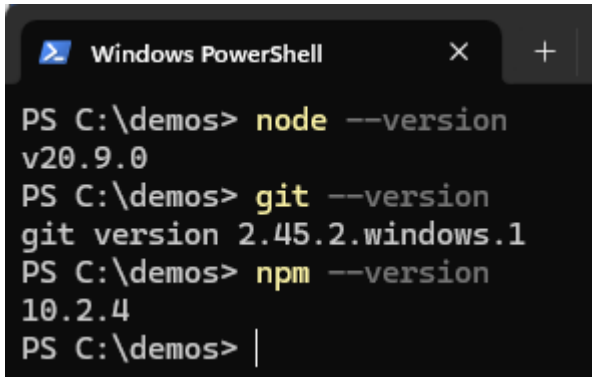
Install Node & NPM

Install the GitHub repository

Install Node.js & NPM





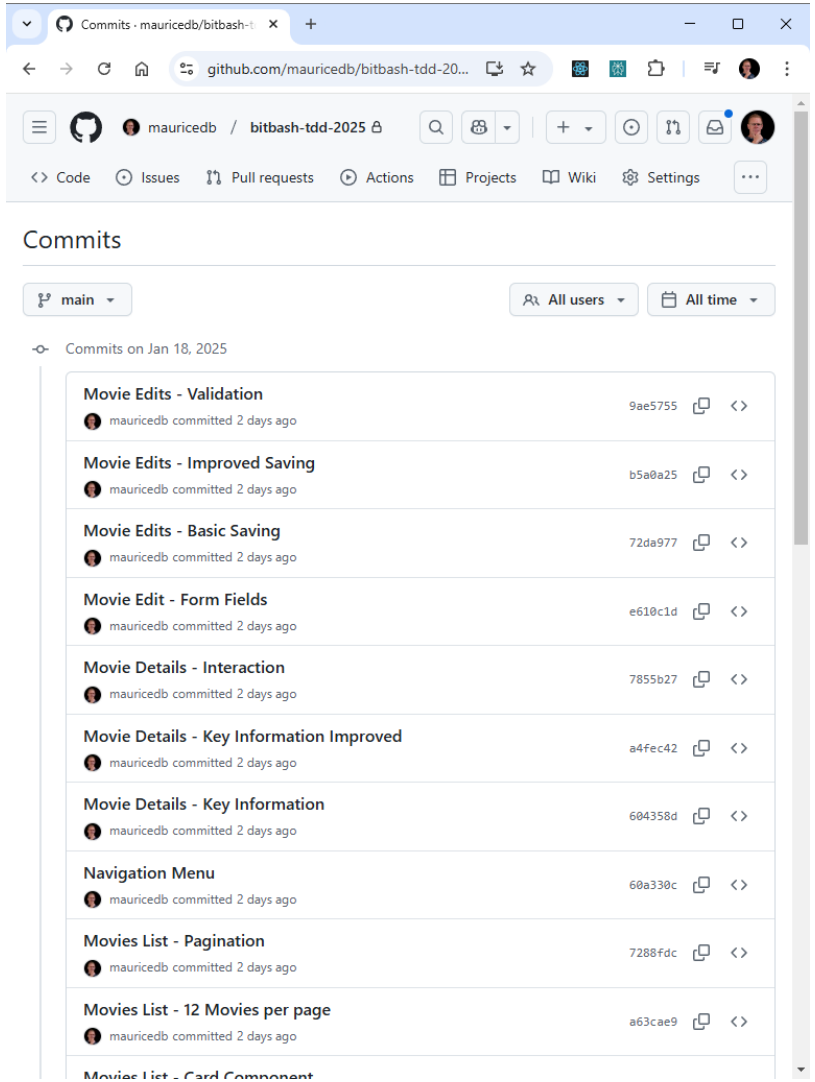


Following Along

- Repo: https://github.com/mauricedb/bitbash-tdd-2025
- Slides: https://www.theproblemsolver.dev/docs/bitbash-tdd-2025.pdf

The changes


MAKE IT SO
memegenerator.net


Commits · mauricedb/bitbash-t
github.com/mauricedb/bitbash-tdd-20...
mauricedb / bitbash-tdd-2025
Code
Issues
Pull requests
Actions
Projects
Wiki
Settings
Commits
main
All users
All time
Commits on Jan 18, 2025
Movie Edits - Validation
mauricedb committed 2 days ago
9ae5755
Movie Edits - Improved Saving
mauricedb committed 2 days ago
b5a0a25
Movie Edits - Basic Saving
mauricedb committed 2 days ago
72da977
Movie Edit - Form Fields
mauricedb committed 2 days ago
e610c1d
Movie Details - Interaction
mauricedb committed 2 days ago
7855b27
Movie Details - Key Information Improved
mauricedb committed 2 days ago
a4fec42
Movie Details - Key Information
mauricedb committed 2 days ago
604358d
Navigation Menu
mauricedb committed 2 days ago
60a330c
Movies List - Pagination
mauricedb committed 2 days ago
7288fdc
Movies List - 12 Movies per page
mauricedb committed 2 days ago
a63cae9

Clone the GitHub Repository

Windows PowerShell

```
PS C:\demos> git clone https://github.com/mauricedb/bitbash-tdd-2025.git
Cloning into 'bitbash-tdd-2025'...
remote: Enumerating objects: 242, done.
remote: Counting objects: 100% (242/242), done.
remote: Compressing objects: 100% (117/117), done.
remote: Total 242 (delta 117), reused 239 (delta 114), pack-reused 0 (from 0)
Receiving objects: 100% (242/242), 370.70 KiB | 3.56 MiB/s, done.
Resolving deltas: 100% (117/117), done.
PS C:\demos>
```

Install NPM Packages

```
Windows PowerShell

PS C:\demos> cd .\bitbash-tdd-2025\
PS C:\demos\bitbash-tdd-2025> 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 go
od and tested way to coalesce async requests by a key value, which is much 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 406 packages, and audited 407 packages in 13s

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

1 moderate severity vulnerability

To address all issues, run:
  npm audit fix --force

Run `npm audit` for details.
PS C:\demos\bitbash-tdd-2025>
```

Start branch

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

Start the application

next-server (v14.2.16)

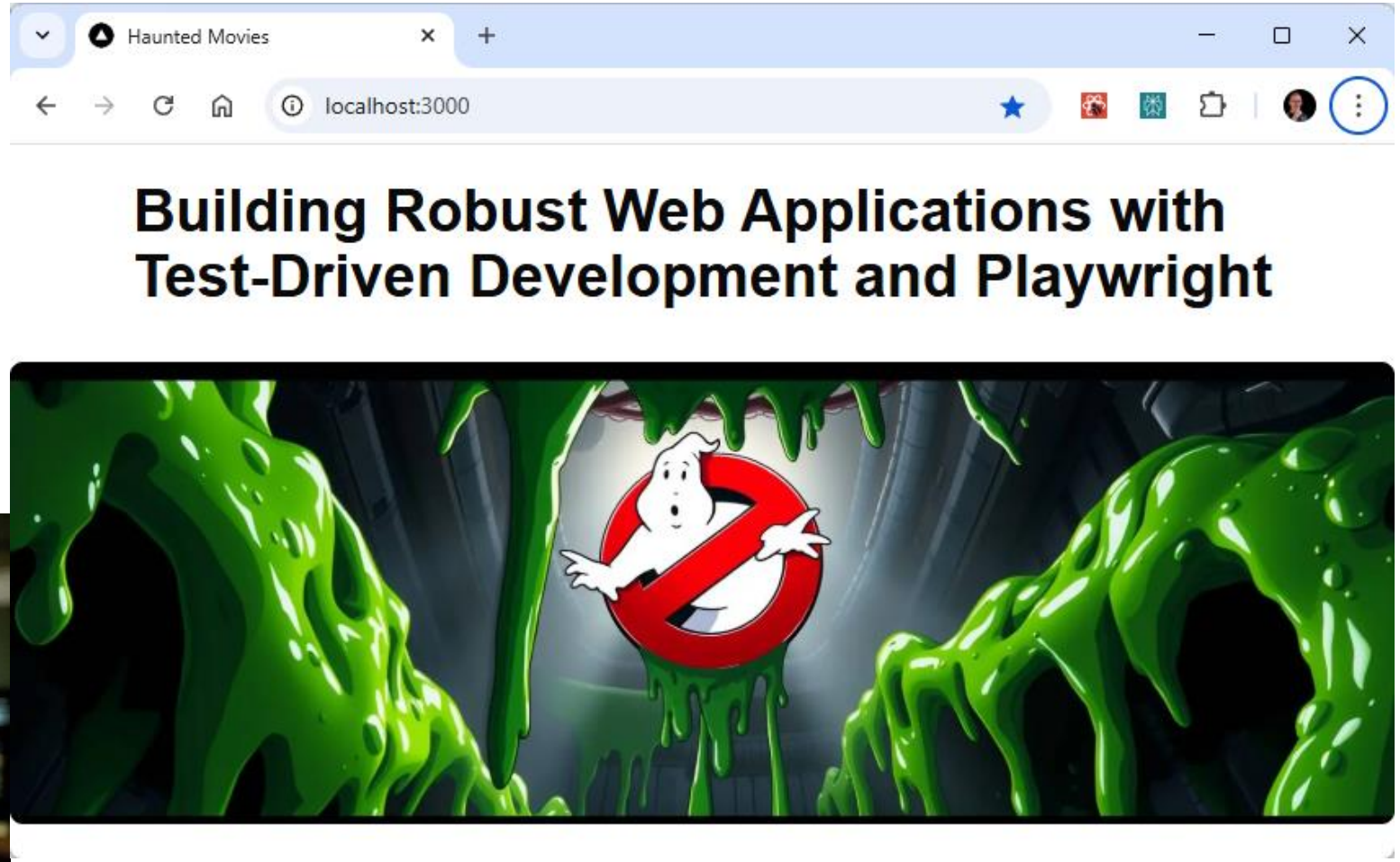
```
PS C:\demos\bitbash-tdd-2025> git checkout --track origin/00-start
branch '00-start' set up to track 'origin/00-start'.
Switched to a new branch '00-start'
PS C:\demos\bitbash-tdd-2025> npm run dev

> bitbash-tdd-2025@0.1.0 dev
> next dev

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

 ✓ Starting...
 ✓ Ready in 1631ms
```

The application



Introduction to Test-Driven Development (TDD)

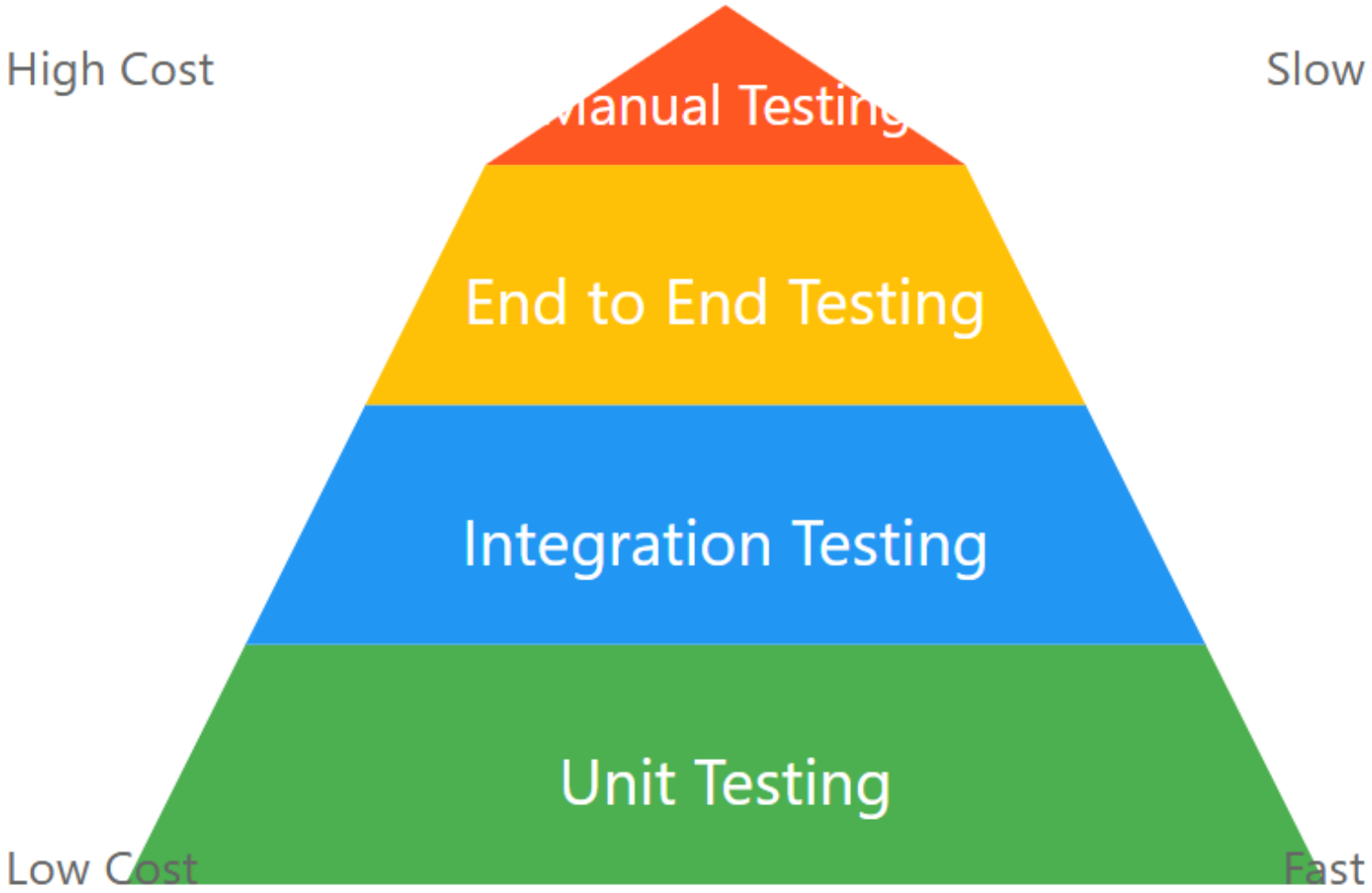
What is Test-Driven Development?

- 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

```
Windows PowerShell

PS C:\demos\bitbash-tdd-2025> npm init playwright@latest
Need to install the following packages:
create-playwright@1.17.135
Ok to proceed? (y) y
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)…

up to date, audited 407 packages in 3s

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

1 moderate severity vulnerability

To address all issues, run:
  npm audit fix --force

Run `npm audit` for details.
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:\demos\bitbash-tdd-2025

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:
```

package.json

{} package.json M ×

{} package.json > ...

```
5    "scripts": {
6      "dev": "next dev",
7      "build": "next build",
8      "start": "next start",
9      "lint": "next lint",
10     "e2e:test": "playwright test",
11     "e2e:test:ui": "playwright test --ui"
12   },
13   "dependencies": {
```

Playwright test console mode

```
Windows PowerShell

PS C:\demos\bitbash-tdd-2025> npm run e2e:test

> bitbash-tdd-2025@0.1.0 e2e:test
> playwright test


Running 6 tests using 4 workers
  6 passed (7.0s)

To open last HTML report run:

  npx playwright show-report

PS C:\demos\bitbash-tdd-2025>
```

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

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

Installation

Introduction

Installing Playwright

Locator Source Call Log Errors Console Network 66 Attachments Annotations

No errors

SETTINGS

Implementing the Landing Page

Implementing Landing Page

"As a haunted 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 5, M

tests > TS 1-implementing-the-landing-page.spec.ts > ...

```typescript
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('Haunted Movies');

    await expect(page.getByText('Haunted Movies').first()).toBeVisible();
    await expect(page.getByText('Home').first()).toBeVisible();
    await expect(page.getByText('About').first()).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

TS 1-implementing-the-landing-page.spec.ts M × page.tsx 2, M

tests > TS 1-implementing-the-landing-page.spec.ts > ...

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

    await expect(page).toHaveTitle('Haunted Movies');
    const navBar = page.getByRole('navigation');

    await expect(
      navBar.getByRole('link', {
        name: 'Haunted Movies Haunted Movies',
      })
    ).toBeVisible();
    await expect(navBar.getByRole('link', { name: 'Home' })).toBeVisible();
    await expect(
      navBar.getByRole('link', { name: 'Haunted Movies', exact: true })
    ).toBeVisible();
    await expect(navBar.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 | page.tsx M | TS playwright.config.ts M ×

TS playwright.config.ts > ...

```
14  export default defineConfig({
26    /* Shared settings for all the projects below. See https://playwr
27    use: {
28      /* Base URL to use in actions like `await page.goto('/')`. */
29      baseURL: 'http://127.0.0.1:3000',
30
31      /* Collect trace when retrying the failed test. See https://pla
32      trace: 'on-first-retry',
33    },
```

Landing Page - Main Content

TS 1-implementing-the-landing-page.spec.ts M × | page.tsx M | TS playwright.config.ts M

tests > TS 1-implementing-the-landing-page.spec.ts > ...

```
3   test.describe('Landing Page', () => {
4     test.beforeEach(async ({ page }) => {
5       await page.goto('/');
6     });
7
8     test('has application title and header', async ({ page }) => {…
22    });
23
24    test('has main content', async ({ page }) => {
25      const main = page.getByRole('main');
26      await expect(
27        main.getByText('Discover Your Next Favorite Haunted Movie')
28      ).toBeVisible();
29      await expect(
30        main.getByText(
31          'Browse our extensive collection of haunted movies and find your next favorite.'
32        )
33      ).toBeVisible();
34      await expect(
35        main.getByRole('link', { name: 'Browse Haunted Movies' })
36      ).toBeVisible();
37    });
38  });
```

Break time

Implementing the Movie List

Implementing the Movie List

"As a haunted movies enthusiast

I want to browse through a list of top-rated haunted 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

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

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

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

test.describe('Haunted Movies List Page', () => {
  test.beforeEach(async ({ page }) => {
    await page.goto('/haunted-movies');
  });

  test('shows haunted movies', async ({ page }) => {
    await expect(
      page.getByRole('heading', { name: 'Haunted Movies' })
    ).toBeVisible();

    await expect.soft(page.getByText('Alien')).toBeVisible();
    await expect.soft(page.getByText('Dawn of the Dead')).toBeVisible();
    await expect.soft(page.getByText('Frankenstein')).toBeVisible();
  });
});
```

Movies List - Grid Layout

"As a haunted movie enthusiast

I want to see the movies in a responsive grid"

Movies List - Grid Layout

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

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

```
3   test.describe('Haunted Movies List Page', () => {
18    test('Movie Grid Layout', async ({ page }) => {
19      await expect(page.getByRole('grid')).toBeVisible();
20      await expect(
21        page.getByRole('grid').getByRole('gridcell').getByText('Alien')
22      ).toBeVisible();
23    });
24  });
```

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 ↓A, M × | 2-implementing-the-movie-list.md | page.tsx 7, ↓A, M

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

```
3   test.describe('Haunted Movies List Page', () => {
25    test('Grid should adapt to screen size', async ({ page }) => {
26      const [cell1, cell2, cell3, cell4, cell5] = await page
27        .getByRole('gridcell')
28        .all();
29
30      await test.step('Test mobile screens (1 column)', async () => {
31        await page.setViewportSize({ width: 375, height: 800 });
32
33        const top1 = (await cell1.boundingBox())?.y;
34        const top2 = (await cell2.boundingBox())?.y;
35
36        expect.soft(top2).toBeGreaterThan(top1 ?? 0);
37      });
38
39      await test.step('Test tablet screens (2 columns)', async () => {
40        await page.setViewportSize({ width: 740, height: 800 });
41
42        const top1 = (await cell1.boundingBox())?.y;
43        const top2 = (await cell2.boundingBox())?.y;
44        const top3 = (await cell3.boundingBox())?.y;
```

Movies List - Sorted by vote

"As a haunted 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 1A, M × page.tsx 7, 1A, M

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

```
3   test.describe('Haunted Movies List Page', () => {
73    test('The movies should be sorted by `vote_average` in descending order', async ({
74      page,
75    }) => {
76      await expect
77        .soft(page.getByRole('gridcell').first().getByText('Psycho'))
78        .toBeVisible();
79
80      await expect
81        .soft(page.getByRole('gridcell').nth(1).getByText('The Shining'))
82        .toBeVisible();
83
84      await expect
85        .soft(page.getByRole('gridcell').nth(2).getByText('Alien'))
86        .toBeVisible();
87    });
88  });
```

Movies List - Card Component

"As a haunted 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

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

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

```
3   test.describe('Haunted Movies List Page', () => {
103     test('Movie Card Component', async ({ page }) => {
104       const firstMovie = page.getByRole('gridcell').first();
105
106       await expect(firstMovie.getByRole('heading')).toBeVisible();
107       await expect(firstMovie.getByRole('img')).toBeVisible();
108
109       await expect(firstMovie.getByLabel('Rating:')).toBeVisible();
110
111       await expect(
112         firstMovie.getByLabel('Rating:').getAttribute('data-value')
113       ).resolves.toBe('8.427');
114     });
115 });
```

Movies List - 12 Movies per page

"As a haunted movie enthusiast

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

Movies List - 12 Movies per page

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

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

```
3   test.describe('Haunted Movies List Page', () => {
116     test('Load 12 movies per page', async ({ page }) => {
117       await expect(page.getByRole('gridcell')).toHaveCount(12);
118     });
119 });
```

Movies List - Pagination

"As a haunted 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 > ...

```
3   test.describe('Haunted Movies List Page', () => {
120   test('should load more movies on pagination', async ({ page }) => {
121     const firstMovieTitle = await page
122       .getByRole('gridcell')
123       .first()
124       .getByRole('heading')
125       .textContent();
126
127     await page.getByRole('link', { name: 'Next' }).click();
128
129     await expect(page).toHaveURL(/page=2/);
130
131     const secondMovieTitle = await page
132       .getByRole('gridcell')
133       .first()
134       .getByRole('heading')
135       .textContent();
136
137     // Verify we're on page 2 with a different movie
138     expect(secondMovieTitle).not.toBe(firstMovieTitle);
139   });
140 });
```

Implementing the Navigation Menu

Implementing the Navigation Menu

"As a haunted movies enthusiast using the application

I want to have a consistent navigation menu

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

Navigation Menu

3-movie-application-navigation-menu.spec.ts U × layout.tsx 1, M page.tsx 1, M

tests > 3-movie-application-navigation-menu.spec.ts > ...

```
2  test.describe('Navigation Menu', () => {
7    test('Header Section Should be visible on all pages', async ({ page }) => {
8      await expect(page).toHaveURL('/');
9      await expect(
10       page.getByRole('heading', {
11         name: 'Welcome to the Haunted Movies Database',
12       })
13     ).toBeVisible();
14     await assertNavigationMenu(page);
15
16     await page
17       .getByRole('link', { name: 'Haunted Movies', exact: true })
18       .click();
19     await expect(page).toHaveURL('/haunted-movies');
20     await expect(
21       page.getByRole('heading', { name: 'Haunted Movies' })
22     ).toBeVisible();
23     await assertNavigationMenu(page);
24
25     await page.getByRole('link', { name: 'Home' }).click();
26
27     await expect(page).toHaveURL('/');
28     await expect(
29       page.getByRole('heading', {
30         name: 'Welcome to the Haunted Movies Database',
31       })
32     ).toBeVisible();
33     await assertNavigationMenu(page);
34   });
35 });
```

Navigation Menu

TS 3-movie-application-navigation-menu.spec.ts U × layout.tsx 1, M page.tsx 1, M

tests > TS 3-movie-application-navigation-menu.spec.ts > ...

```
37  async function assertNavigationMenu(page: Page) {
38    test.step('Assert Navigation Menu', async () => {
39      const header = page.getByRole('banner');
40
41      await expect(
42        header.getByRole('link', { name: 'Haunted Movies Haunted Movies' })
43      ).toBeVisible();
44      await expect(header.getByRole('link', { name: 'Home' })).toBeVisible();
45      await expect(
46        header.getByRole('link', { name: 'Haunted Movies', exact: true })
47      ).toBeVisible();
48      await expect(header.getByRole('link', { name: 'About' })).toBeVisible();
49    });
50  }
```

Implementing the Movie Details Page

Implementing the Movie Details Page

"As a haunted movies 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.md | TS 4-implementing-the-movie-details-page.spec.ts U | page.tsx M

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

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

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

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

    await expect(page).toHaveURL('/movies/539');

    await expect(page.getByRole('img', { name: 'Psycho' })).toBeVisible();
    await expect(page.getByRole('heading', { name: 'Psycho' })).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 **Psycho** 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.md | 4-implementing-the-movie-details-page.spec.ts M | page.tsx

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

You, 34 seconds ago | 1 author (You)

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

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

  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 haunted movies 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

4-implementing-the-movie-details-page.md | TS 4-implementing-the-movie-details-page.spec.ts M | page.tsx

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

```
3   test.describe('Movie Details Page', () => {
27    test('Should navigate to the movie edit page', async ({ page }) => {
28      await page.goto('/movies/539');
29
30      await page.getByRole('link', { name: 'Edit Movie' }).click();
31
32      await expect(page).toHaveURL('/movies/539/edit');
33      // Checking if the edit form is visible would be better
34    });
35  });
```

Implementing the Movie Edit Page

Implementing the Movie Edit Page

"As an administrator of the haunted movies 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

5-implementing-the-movie-edit-page.md | 5-implementing-the-movie-edit-page.spec.ts U | genre-selector.tsx M

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

```
3  test.describe('Movie Edit Page', () => {
8    test('Should display form fields', async ({ page }) => {
9      await expect(
10       page.getByRole('heading', {
11         name: 'Edit Movie: Psycho',
12       })
13     ).toBeVisible();
14
15     const titleInput = page.getByLabel('Title');
16     await expect(titleInput).toBeVisible();
17     await expect(titleInput).toHaveValue('Psycho');
18
19     const overviewTextarea = page.getByLabel('Overview');
20     await expect(overviewTextarea).toBeVisible();
21     await expect(overviewTextarea).toHaveValue(
22       /^When larcenous real estate clerk Marion Crane goes on the lam with a wad of cash
23     );
24
25     await expect(page.getByText('Genres')).toBeVisible();
26     const genresList = page.getByRole('list', { name: 'Movie genres' });
27     await expect(
28       genresList.getByRole('listitem', { name: 'Horror' })
29     ).toBeChecked();
30     await expect(
31       genresList.getByRole('listitem', { name: 'Mystery' })
32     ).toBeChecked();
33     await expect(
34       genresList.getByRole('listitem', { name: 'Thriller' })
35     ).toBeChecked();
```

Saving the Movie Edits

Saving the Movie Edits

"As an administrator of the haunted movies application

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.md | 6-saving-the-movie-edits.spec.ts U | edit-movie-form.tsx M | layout.tsx M

tests > 6-saving-the-movie-edits.spec.ts > ...

```typescript
3  test.describe('Saving Movie Edits', () => {
4    test.beforeEach(async ({ page }) => {
5      await page.goto('/movies/539/edit');
6    });
7
8    test('should have save and cancel buttons', async ({ page }) => {
9      await expect(
10       page.getByRole('button', { name: 'Save Changes' })
11     ).toBeVisible();
12
13     await expect(page.getByRole('link', { name: 'Cancel' })).toBeVisible();
14   });
15
16   test('should save edited movie details successfully', async ({ page }) => {
17     // Submit the form
18     await page.getByRole('button', { name: 'Save Changes' }).click();
19
20     // Verify successful update toast
21     await expect(
22       page.getByText('Movie updated successfully', { exact: true })
23     ).toBeVisible();
24
25     // Verify redirect to details page
26     await expect(page).toHaveURL('/movies/539');
27   });
28 });
```

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.md | 6-saving-the-movie-edits.spec.ts M | page.tsx 1, M

tests > 6-saving-the-movie-edits.spec.ts > ...

```
3   test.describe('Saving Movie Edits', () => {
4     let movieId: number;
5
6     test.beforeEach(async ({ page, request }) => {
7       movieId = -Date.now();
8       const newMovie = {…
23      };
24
25      // Post the new movie
26      const response = await request.post(
27        `https://the-problem-solver-sample-data.azurewebsites.net/horror-movies`,
28        {
29          data: newMovie,
30        }
31      );
32      expect(response.ok()).toBeTruthy();
33
34      await page.goto(`/movies/${movieId}/edit`);
35    });
36
37    test.afterEach(async ({ request }) => {
38      // Delete the test movie
39      const response = await request.delete(
40        `https://the-problem-solver-sample-data.azurewebsites.net/horror-movies/${movieId}`
41      );
42
43      expect(response.ok()).toBeTruthy();
44    });
```

Validating the Movie Edits

Validating the Movie Edits

"As an administrator submitting haunted 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.md | 7-validating-the-movie-edits.spec.ts U | actions.ts M

tests > 7-validating-the-movie-edits.spec.ts > ...

```
3   test.describe('Validating Movie Edits', () => {
50    test('should fail to save with empty title', async ({ page }) => {
51      await page.getByLabel('Title').clear();
52
53      // Submit the form
54      await page.getByRole('button', { name: 'Save Changes' }).click();
55
56      // Verify failure update toast
57      await expect(
58        page.getByText('The movie title is required', { exact: true })
59      ).toBeVisible();
60
61      // Verify no redirect to details page
62      await expect(page).toHaveURL(`/movies/${movieId}/edit`);
63    });
64
65    test('should fail to save with empty overview', async ({ page }) => {
66      await page.getByLabel('Overview').clear();
67
68      // Submit the form
69      await page.getByRole('button', { name: 'Save Changes' }).click();
70
71      // Verify failure update toast
72      await expect(
73        page.getByText('The movie overview is required', { exact: true })
74      ).toBeVisible();
75
76      // Verify no redirect to details page
77      await expect(page).toHaveURL(`/movies/${movieId}/edit`);
78    });
79  });
```

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