



Beyond Code Coverage: Functionality Testing with Playwright



MARLENE MHANGAMI
SENIOR Developer Advocate
Microsoft/GitHub



github.com



GITHUB OCTOVERSE 2025

More Code Is Being Created

PULL REQUESTS MERGED / MONTH

▲ 23% YoY

43.2M

pull requests merged each month in 2025



COMMITTS PUSHED IN 2025

▲ 25% YoY

~1B

commits pushed — GitHub's most active year ever



Source: [GitHub Octoverse 2025](#)



github.com



2026 GROWTH RATES

Growth Is Accelerating

A growing share of these commits are co-authored by AI agents

COMMITTS PUSHED IN 2024

▲ 25% YoY

~1B

commits pushed — GitHub's most active year ever



PROJECTED COMMITTS IN 2025

▲ 14x

~14B

at 275M commits/week x 52 weeks



Kyle Daigle - GitHub COO

Does AI Make
Developers More
Productive?



Clean code amplifies AI gains

Clean engineering environments allow AI to autonomously drive a larger share of the sprint

Clean Code Amplifies AI Gains

- A clean codebase allows AI to complete a larger share of sprint tasks

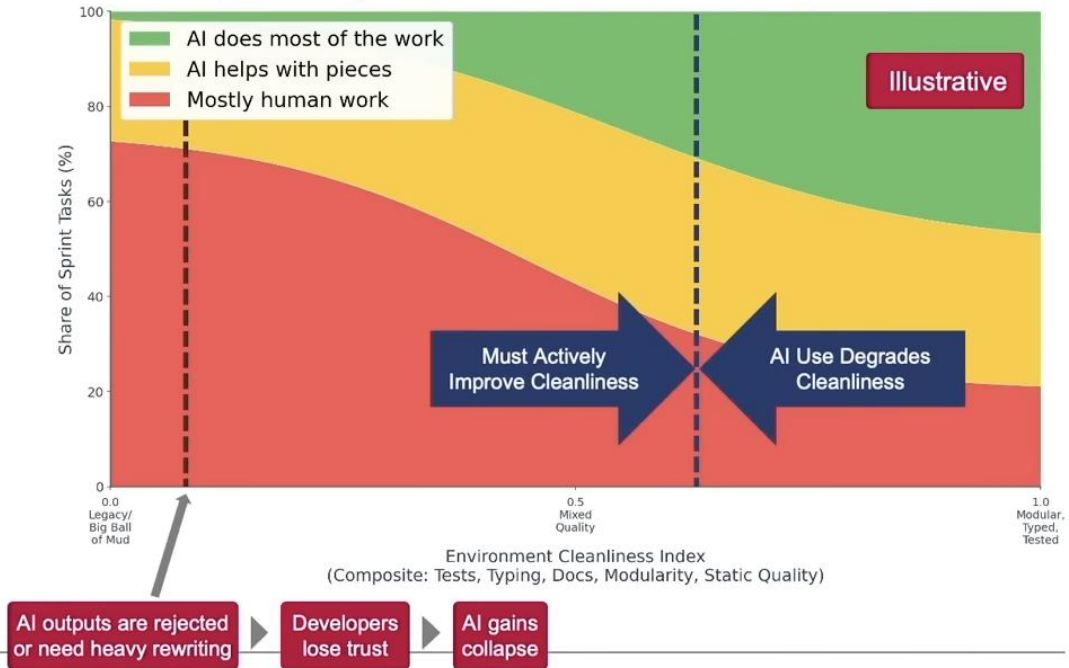
Manage Codebase Entropy

- Unchecked AI accelerates entropy (tech debt)
- High entropy degrades future AI performance

Engineers Must Master Task Selection

- Knowing when to use AI
- ...and when to write code manually

Task Composition by AI Involvement vs. Environment Cleanliness Index

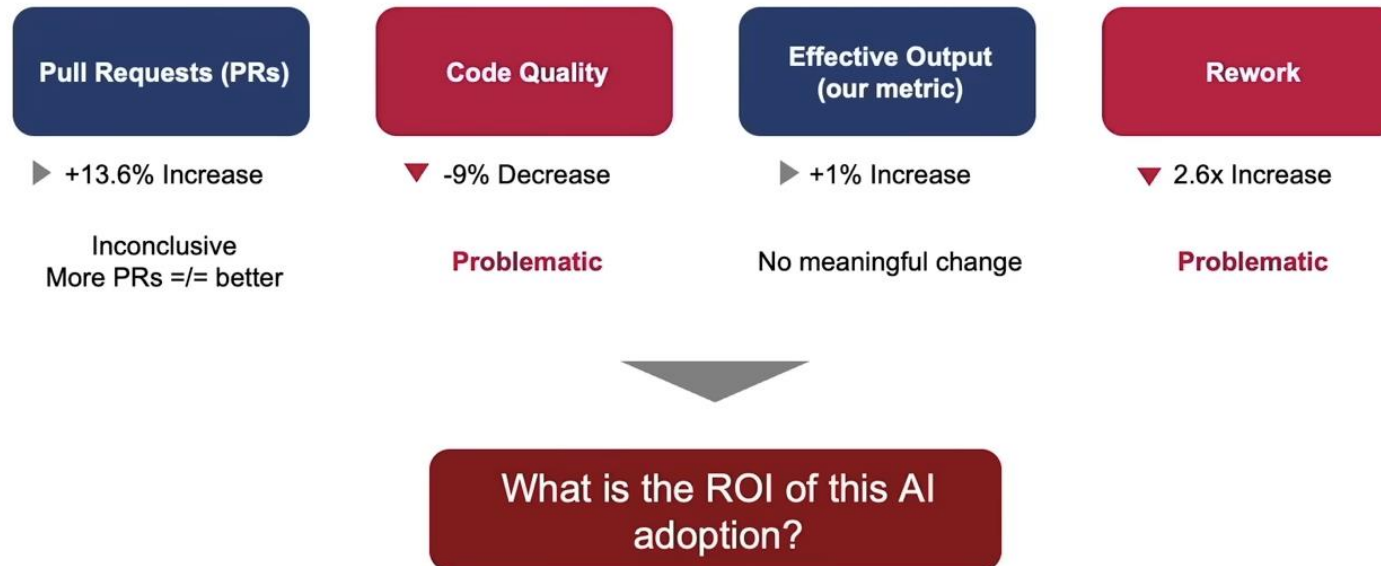


Unchecked AI amplifies entropy

Case Study

Adopting AI didn't yield positive results for this company...

AI Impact on Productivity – Pilot Metrics Summary



Priorities for AI-assisted teams

Clean engineering environments tend to unlock more AI productivity gains

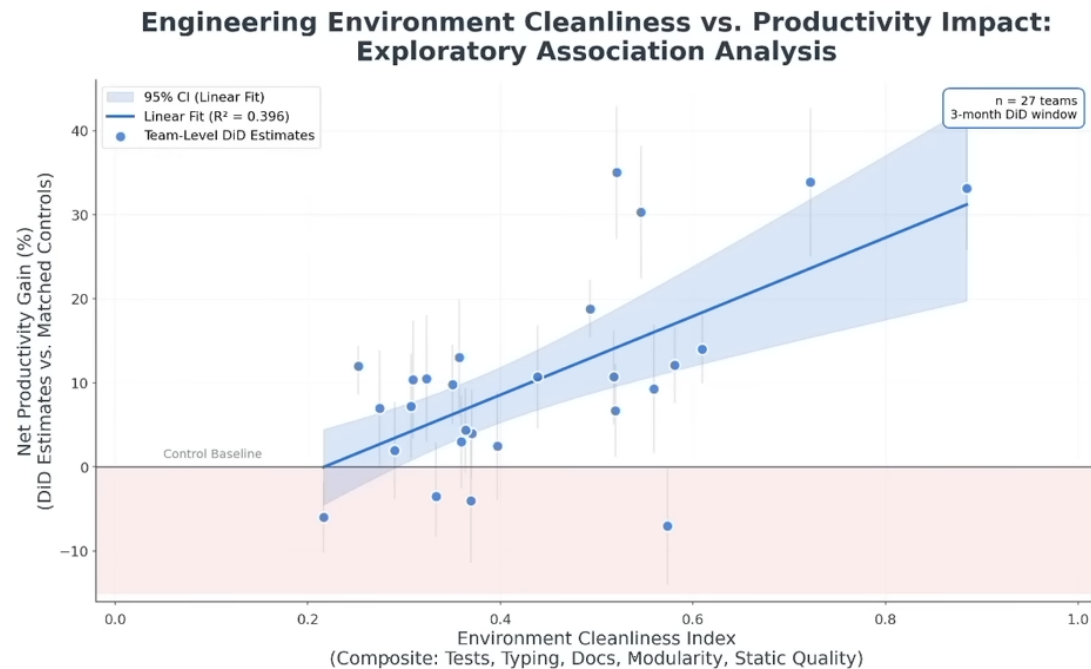
Environment Cleanliness Index

Composite score:

- Test coverage
- Type coverage
- Documentation quality & coverage
- Modularity
- Static code quality

Invest in codebase hygiene to unlock AI productivity gains

- Large, messy systems is where AI is most tempting & most dangerous
- Prioritize AI-heavy workflows in teams that already have strong engineering foundations

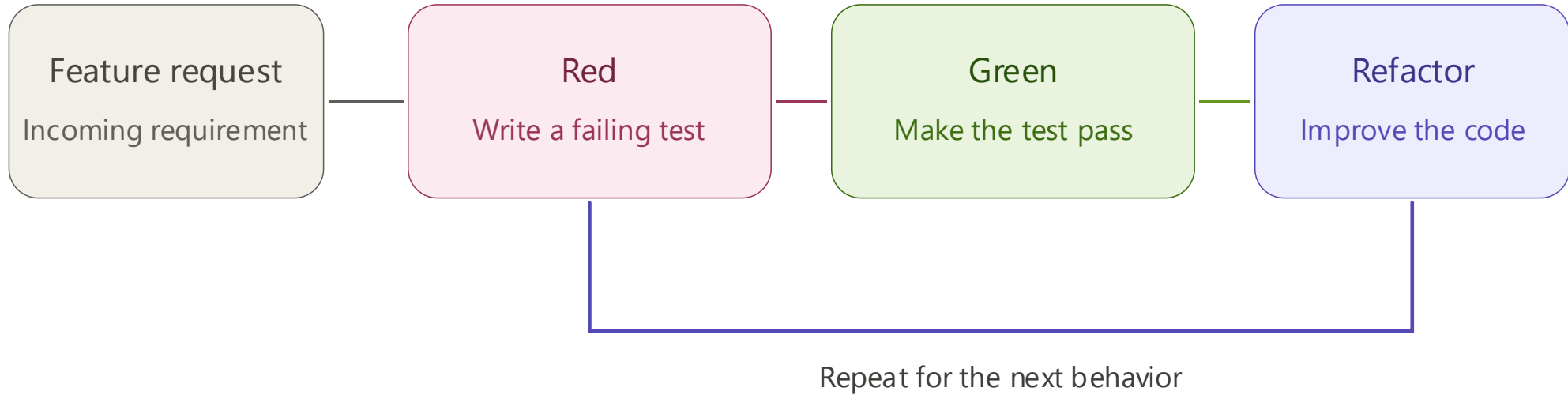



Methods: Same 27 teams and DiD effects as in previous slide. Environment Cleanliness Index is a 0 - 1 composite of test coverage, static code quality, type coverage, documentation coverage, and modularity. Plot shows OLS fit of DiD % gains on cleanliness with HC3-robust CIs; linear $R^2 \approx 0.3 - 0.4$. Exploratory, observational; cleanliness is treated as a correlate, not a causal estimate.



How Can Developers Create And Maintain Clean Code?





 Red/Green TDD Loop

Test Driven Development (TDD)

“The current fanatical TDD experience leads to a primary focus on unit tests... rebalance the testing spectrum from unit to system”

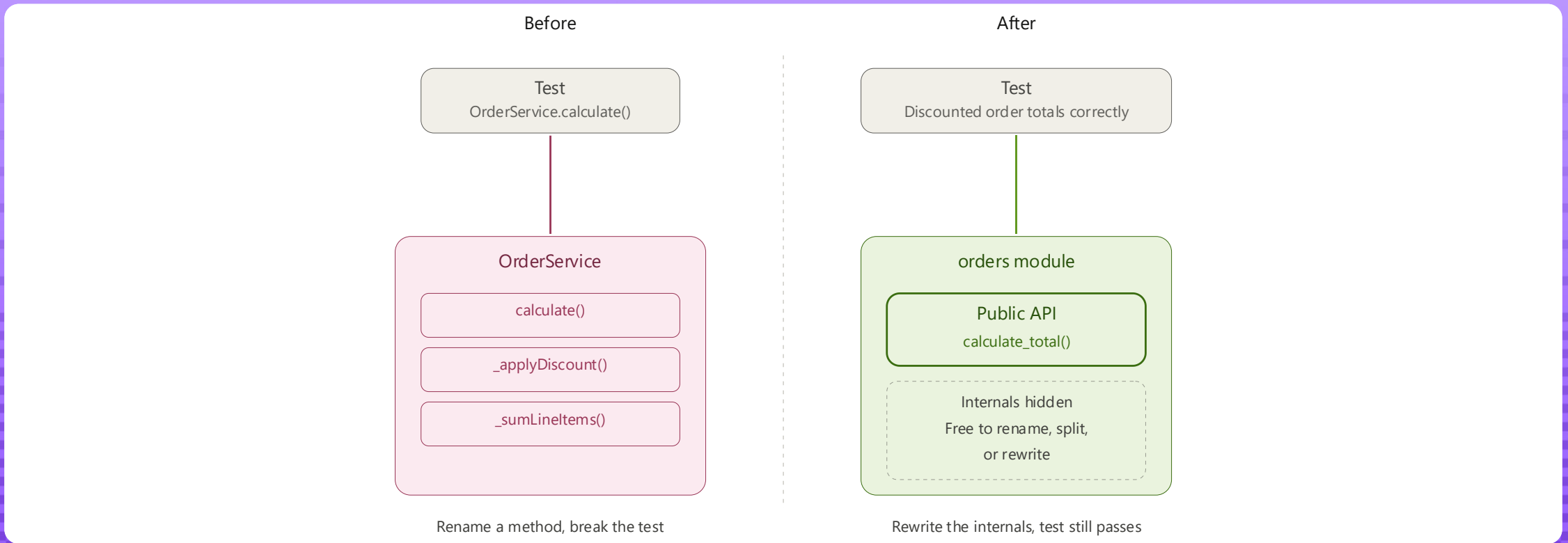
Over-indexing on code coverage

TDD is dead. Long live testing.

By [David Heinemeier Hansson](#) on April 23, 2014

Test-first fundamentalism is like abstinence-only sex ed: An unrealistic, ineffective morality campaign for self-loathing and shaming.

It didn't start out like that. When I first discovered TDD, it was like a courteous invitation to a better world of writing software. A mind hack to get you going with the practice of testing where no testing had happened before. It opened my eyes to the tranquility of a well-tested code base, and the bliss of confidence it grants those making changes to software.



‘We are not testing a method on a class
we are testing a behavior of the system’

TDD, Where Did It All Go Wrong (Ian Cooper)

You're absolutely right!

Self-affirming test

```
1 def add_tax(price):  
2     return price * 1.05 # bug: should be 1.20  
3  
4 def test_add_tax():  
5     assert add_tax(100) == add_tax(100)
```

Always passes. Tells you nothing about whether the tax is correct.

Behavioral test

```
1 def test_add_tax_applies_uk_vat():  
2     # UK VAT is 20%  
3     assert add_tax(100) == 120
```

Encodes the requirement. Catches the bug instead of affirming it.

Playwright For Functionality Testing





Playwright

Playwright is an open-source testing framework by Microsoft that automates end to end testing in the browser by simulating user interactions

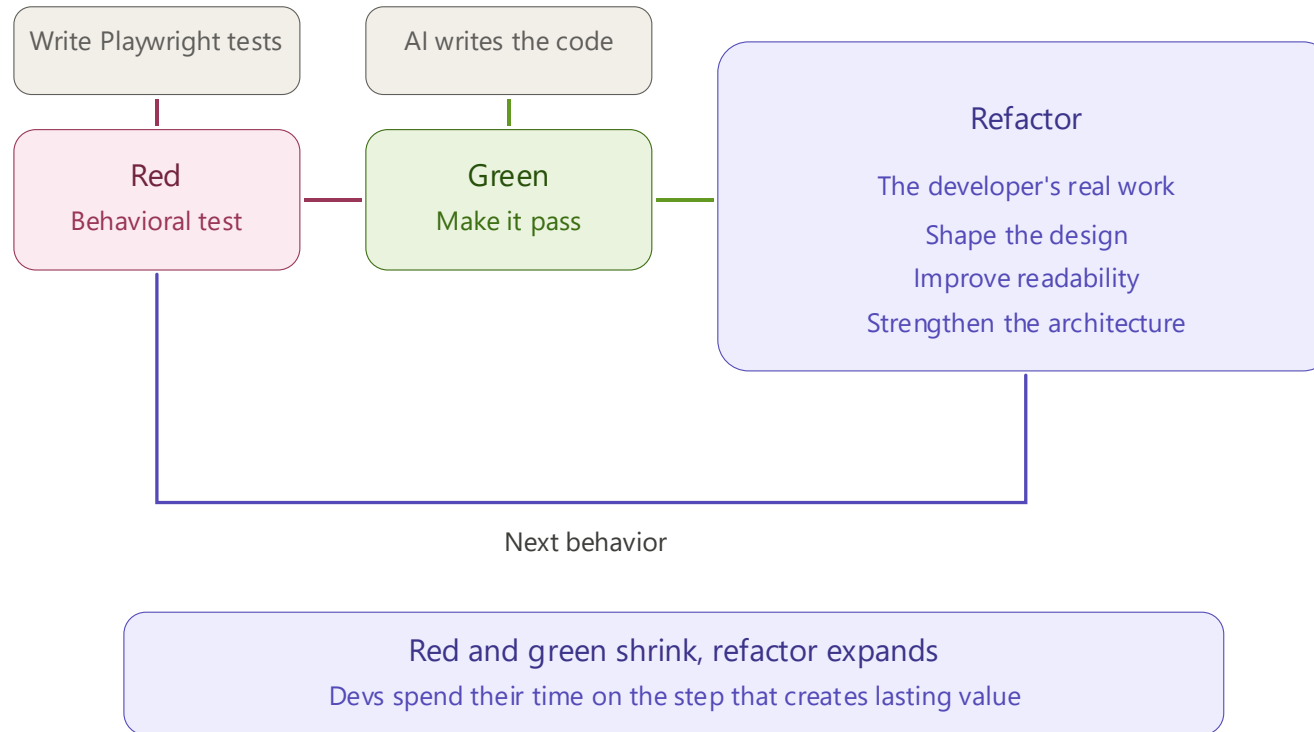
<https://playwright.dev/>

Playwright test

Python and Typescript support

```
1 from playwright.sync_api import Page, expect
2
3 def test_search_filters_toys(page: Page):
4     page.goto("/toys")
5
6     # Search by toy name
7     search_input = page.get_by_placeholder("search")
8     expect(search_input).to_be_visible()
9
10    search_input.fill("Furby")
11
12    expect(page.get_by_role("heading",
13                          name="Furby")).to_be_visible()
```





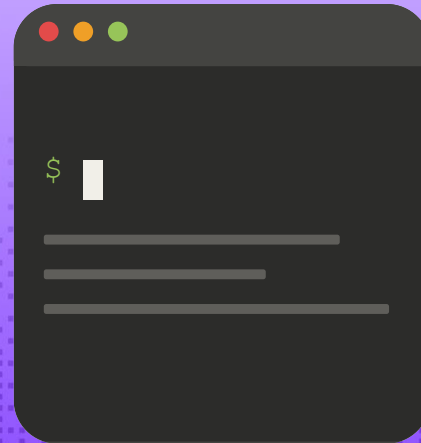
Playwright with AI for TDD

Using Playwright with coding agents



MCP Server

```
npx @playwright/mcp@latest
```



CLI

```
npm install -g @playwright/cli@latest
```



Agents

```
npx playwright init-agents --loop=vscode  
Planner, Generator and Healer (agent.md  
files)
```



DEMO



Helpful Commands

Installation:

- Python: `pytest tests/search_spec.py`
- Typescript: `npx playwright test`

Run tests:

- Python: `pytest tests/search_spec.py`
- Typescript: `npx playwright test`

<https://playwright.dev/>

AI GENERATES CODE

Green: Generate code quickly to make tests pass

Prompt: Generate the code required for the tests to pass, try and match the implementation style of the codebase. Do not edit the tests.

```
1 'use client';
2
3 import { useState } from 'react';
4 import { Toy } from '@prisma/client';
5 import { ProductCard } from '@components/toys/product-card';
6 import { SearchBar } from '@components/toys/search-bar';
7
8 interface ToysCatalogueProps {
9   toys: Toy[];
10 }
11
12 /** Displays the toys collection in a grid layout */
13 /** Displays the toys collection with client-side search filter */
14 export function ToysCatalogue({ toys }: ToysCatalogueProps) {
15   const [search, setSearch] = useState('');
16
17   const filtered = toys.filter((toy) => {
18     const q = search.toLowerCase();
19     return (
20       toy.name.toLowerCase().includes(q) ||
21       toy.category.toLowerCase().includes(q) ||
22       toy.shortTagline.toLowerCase().includes(q)
23     );
24   });
25 }
```



Refactor: Review and fix any issues with the implementation

```
1 import { Toy } from '@prisma/client';
2 import { ProductCard } from '@components/toys/product-card';
3
4 interface ToysCatalogueProps {
5   toys: Toy[];
6 }
7
8 /** Displays the toys collection in a grid layout */
9 export function ToysCatalogue({ toys }: ToysCatalogueProps) {
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
return (
```

```
1+ 'use client';
2+
3+ import { useState } from 'react';
4 import { Toy } from '@prisma/client';
5 import { ProductCard } from '@components/toys/product-card';
6+ import { SearchBar } from '@components/toys/search-bar';
7
8 interface ToysCatalogueProps {
9   toys: Toy[];
10 }
11
12+ /** Displays the toys collection with client-side search filtering */
13 export function ToysCatalogue({ toys }: ToysCatalogueProps) {
14+   const [search, setSearch] = useState('');
15+
16+   const filtered = toys.filter((toy) => {
17+     const q = search.toLowerCase();
18+     return (
19+       toy.name.toLowerCase().includes(q) ||
20+       toy.category.toLowerCase().includes(q) ||
21+       toy.shortTagline.toLowerCase().includes(q)
22+     );
23+   });
24+
25 return (
```



Best Practices

- Add screenshots to PRs
- Use headless mode for multi-tasking
- Commit before running Healer
- Generate tests one feature at a time

The screenshot shows a GitHub pull request titled "Add search and filter capabilities to toys catalogue #1". The PR description includes a code snippet for filter logic, a list of UI behavior changes, and a screenshot of the application's initial state with filters sidebar.

Filter logic converts database prices (paise) to pounds for comparison:

```
const toyPriceInPounds = toy.price / 100;
const matchesMinPrice = filters.minPrice === 0 || toyPriceInPounds >= filters.minPrice;
```

UI Behavior

- Results counter shows "X of Y toys"
- Clear button appears only when filters active
- Sidebar uses sticky positioning
- Available filter options derived from actual toy data

Screenshots

Initial state with filters sidebar:

The screenshot shows the "All Toys" page on the "Tailspin Toys" website. It features a search bar, a filters sidebar, and a grid of three toys: "Speak & Spell" (Electronic, £19.99), "Water Ring Toss" (Classic, £7.99), and "Mr Frosty" (Activity, £22.99). Each toy card includes an "Add to Basket" button.

<https://playwright.dev/>



Playwright

Resources

Slides: aka.ms/playwright-ai-engineer

Documentation: playwright.dev

Github repo:

Connect:

LinkedIn: [marlenemhangami](https://www.linkedin.com/in/marlenemhangami)

X: [@marlene_zw](https://twitter.com/marlene_zw)

Bluesky: [marlene@bluesky.social](https://bsky.app/profile/marlene@bluesky.social)

Website: marlene.ai