



Testing All The things

By Danny Hearit

BC
Code





Agenda

Goals of automated testing

Testing types

Legacy vs modern testing pyramid

Increasing confidence

E2E on local - Demo

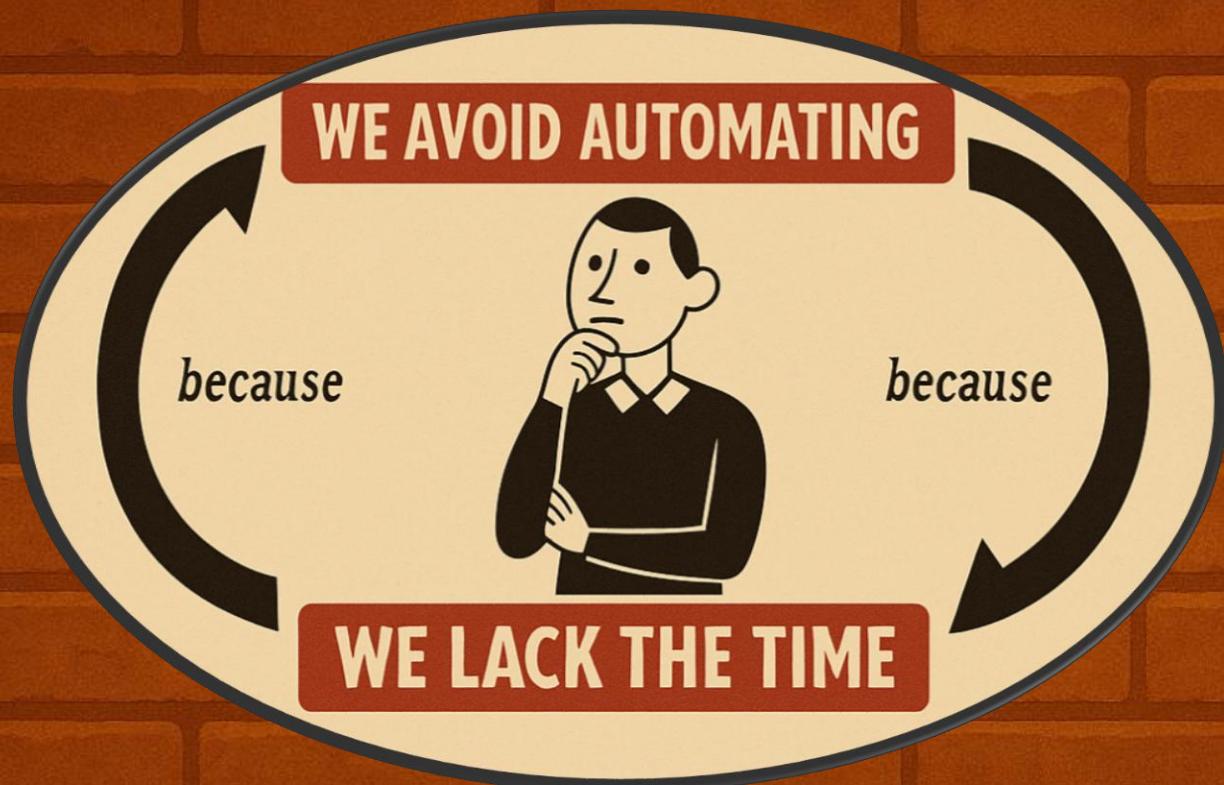
E2E on CI - Demo

Getting started on your team

Tips and best practices

Q&A

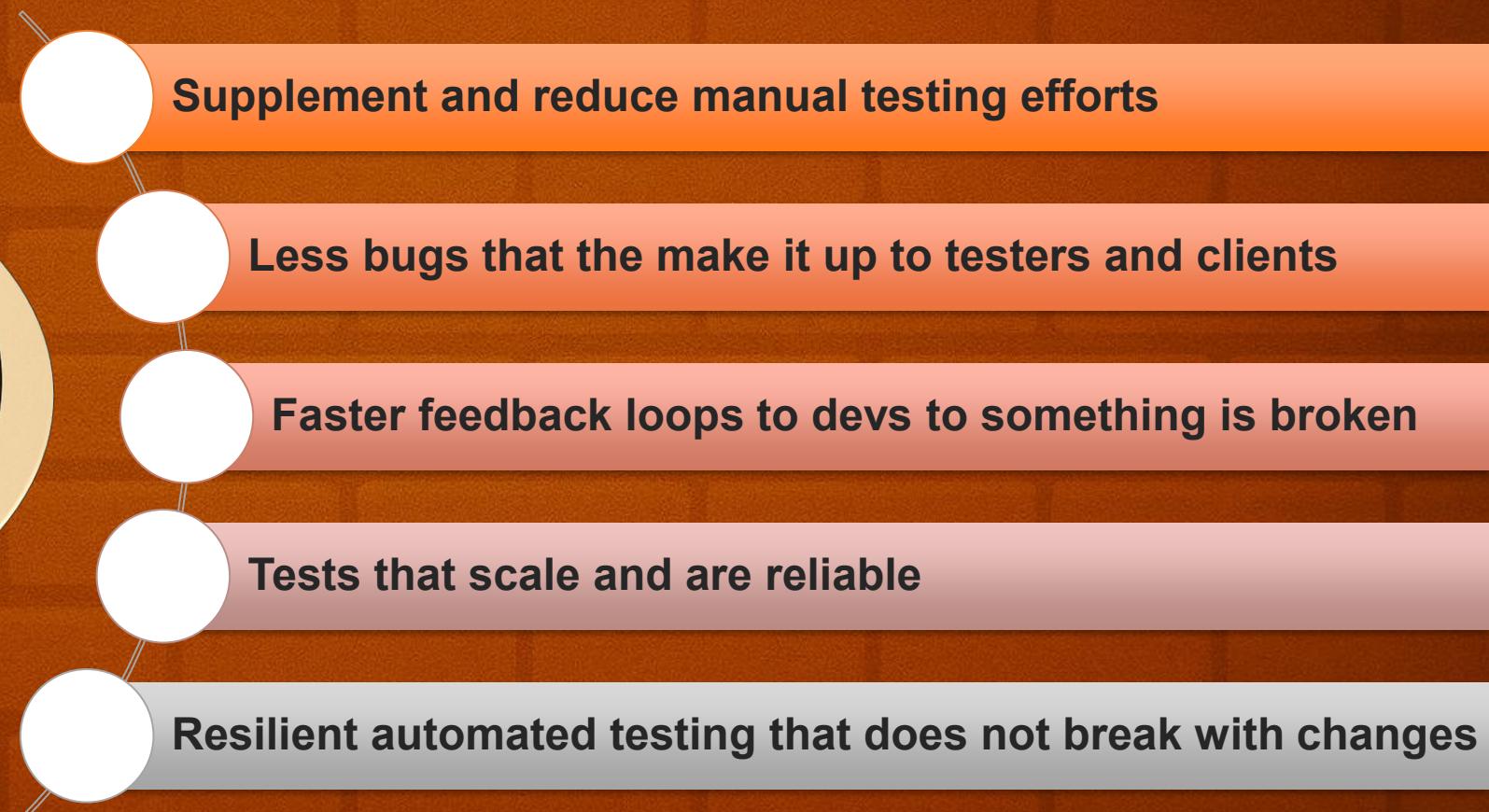
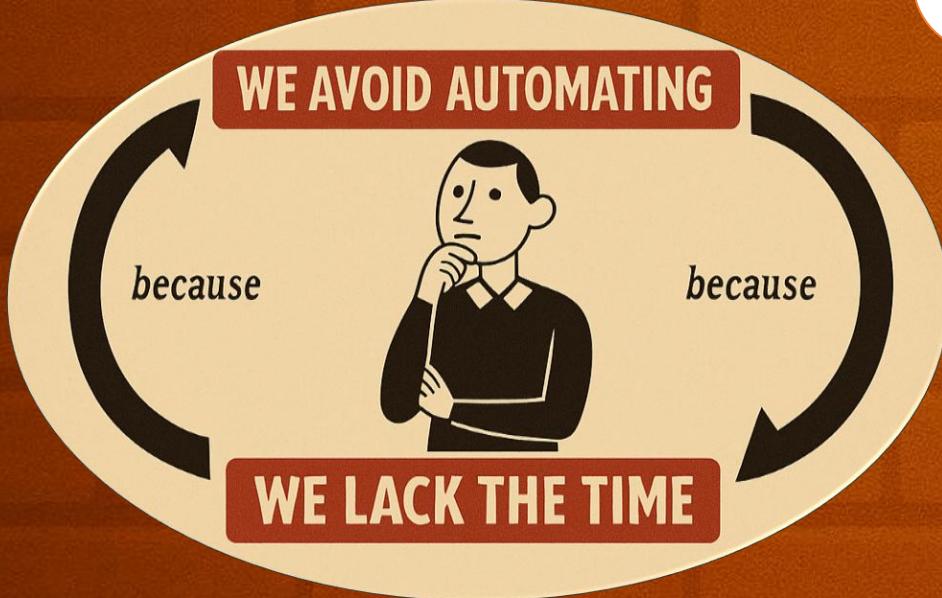
Goals of Automated Testing



BC
Code



Goals of Automated Testing



BC
Code



Testing Types



Unit Tests

What is it?

- Unit == Method

What to test?

- Input/output – Not implementation
- Boundaries/good/bad combinations of arguments
- Expected errors/exceptions (and their messages)
- New features or issues

What techniques or tools commonly used?

- Frameworks – Junit, xUnit, Jasmine
- Mocking/stubbing – If needed



Integration Tests

What is it?

- Integration == Interactions between modules and outside of the system

What to test?

- Different parts working together
- External resources
- Databases, webservers, vendors

What techniques or tools commonly used?

- Scripting
- Functional unit testing
- Test accounts for vendors/external APIs





Component Tests

What is it?

- Component == Service or web components

What to test?

- Aggregation of other calls
- Higher-level operation/abstraction
- Configuration (Angular or not)

What techniques or tools commonly used?

- Angular component testing (Cypress, Angular Unit Tests)
- API/Contract testing



E2E Tests

What is it?

- E2E == User experiences from start to finish

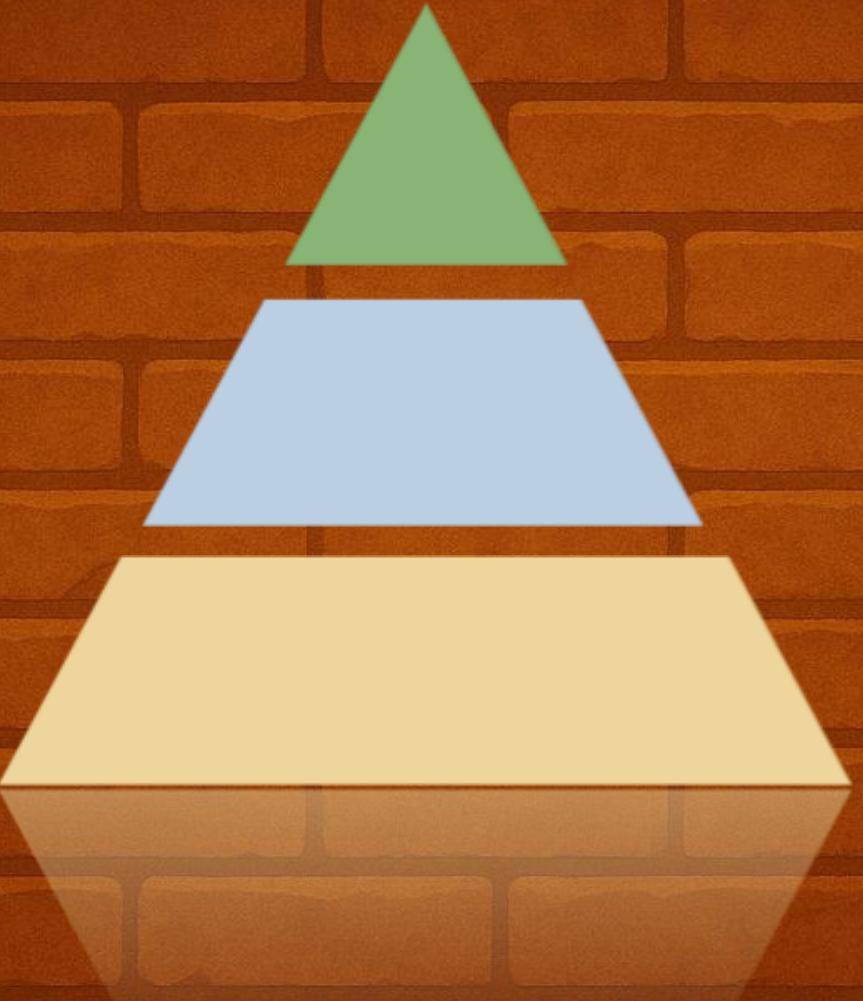
What to test?

- Workflow
- Security
- Happy path
- API responsiveness

What techniques or tools commonly used?

- Cypress, Playwright
- Supplementing manual regression testing



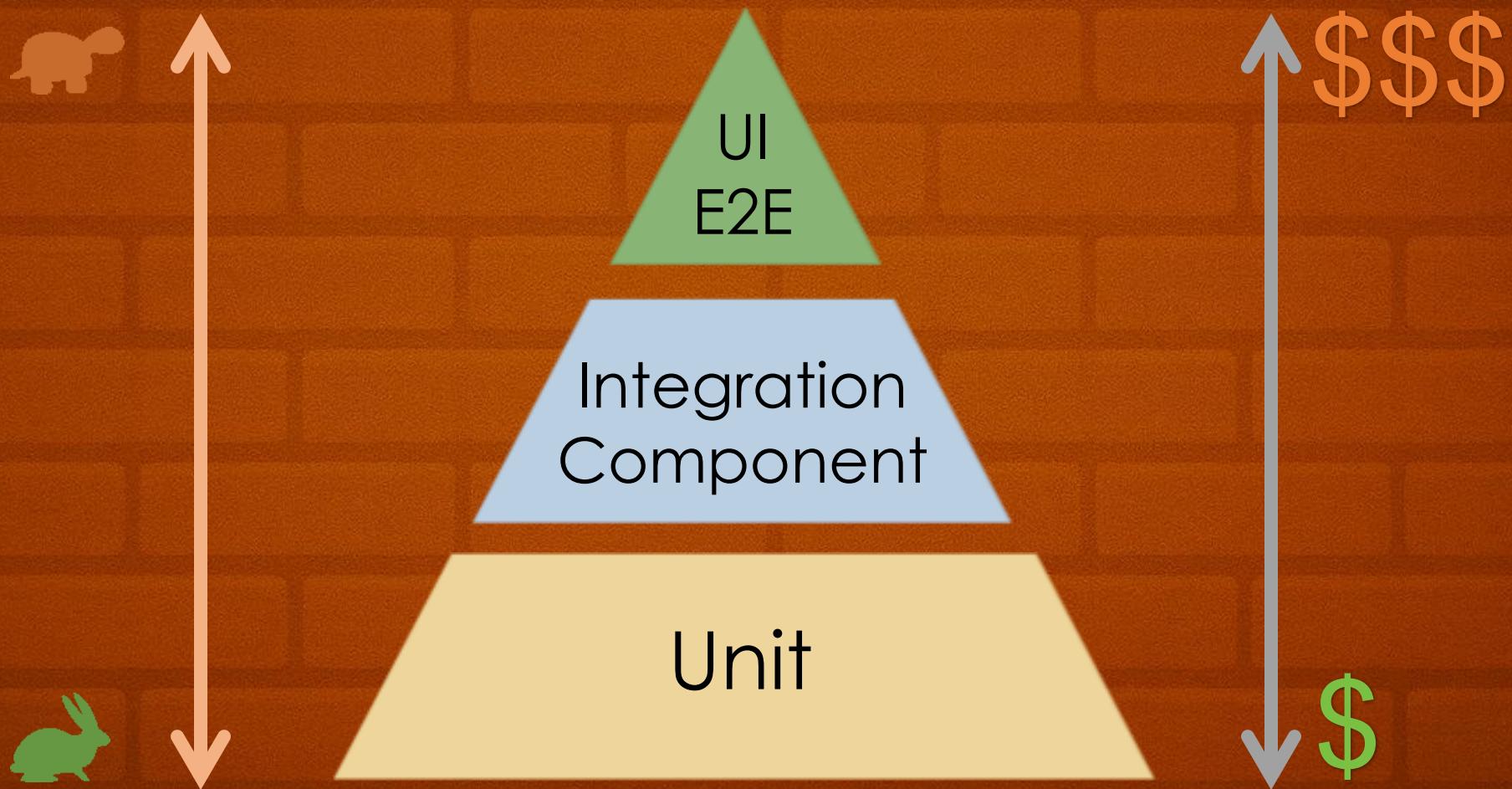


Testing Pyramid

BC
Code

Legacy

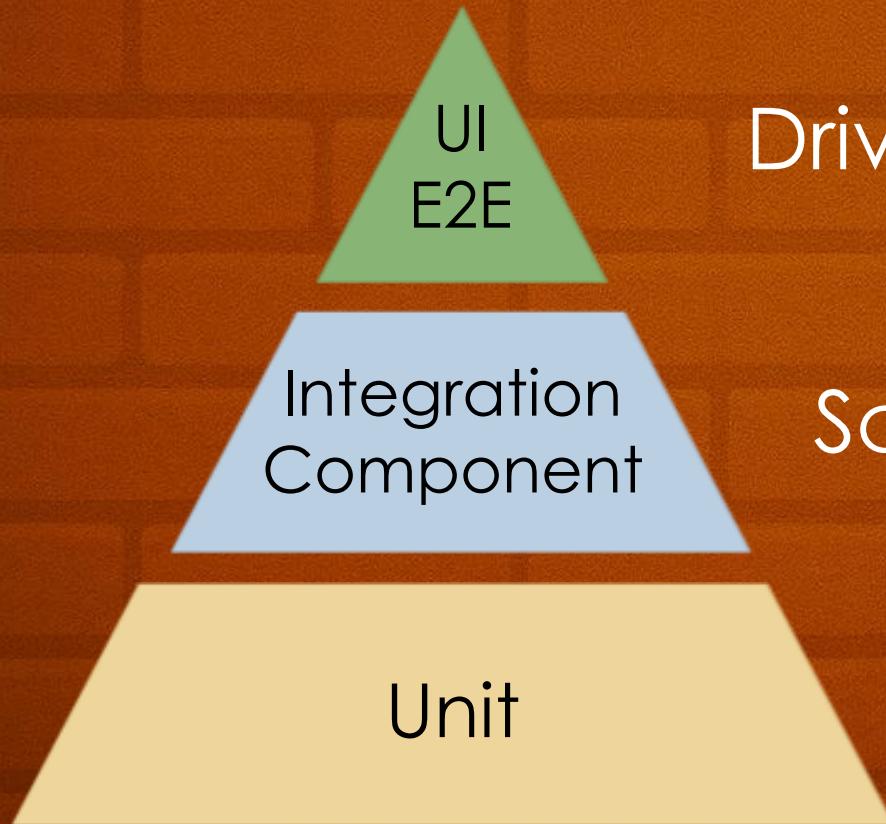
Testing Pyramid



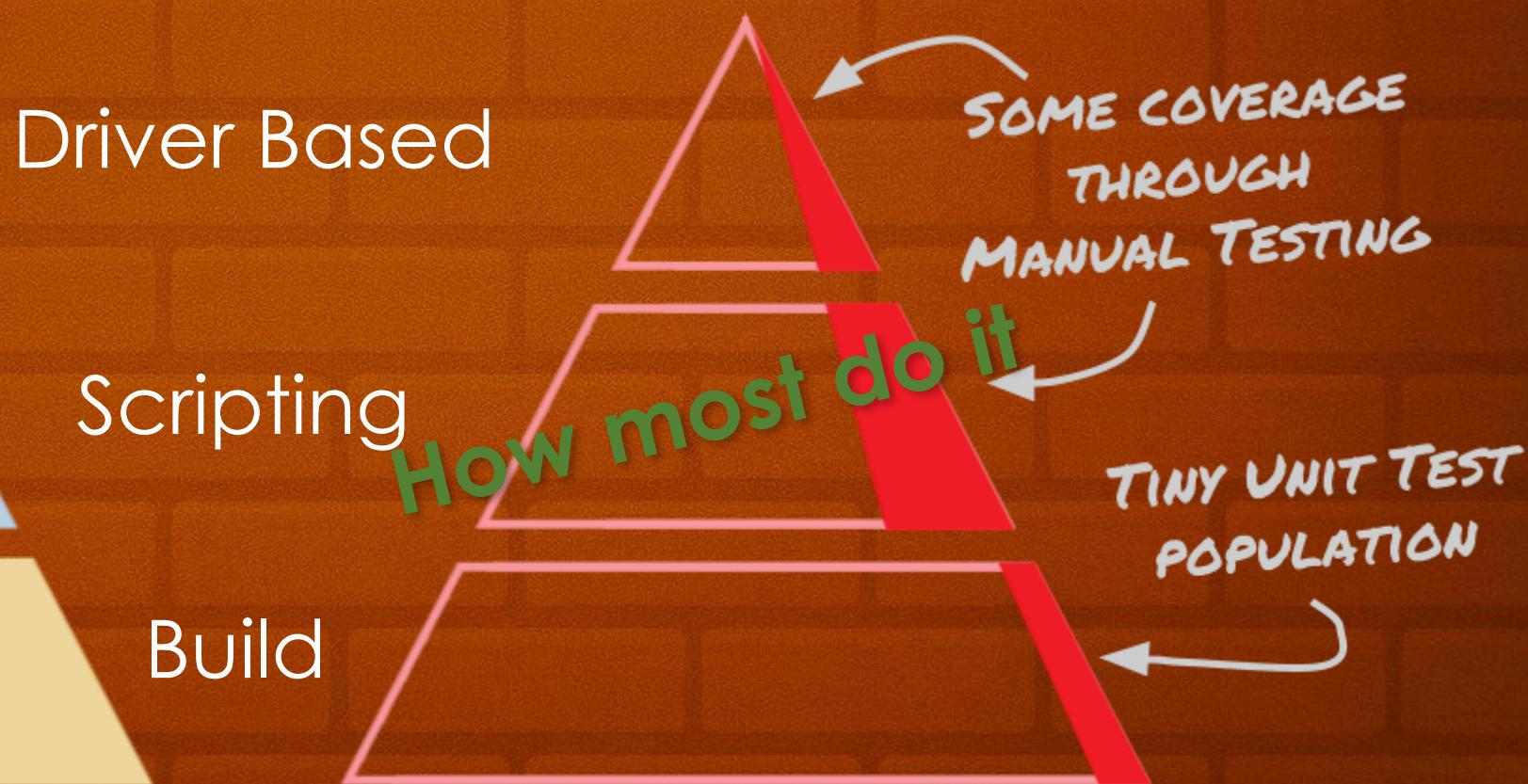
Legacy

Unhealthy Testing

Legacy Pyramid

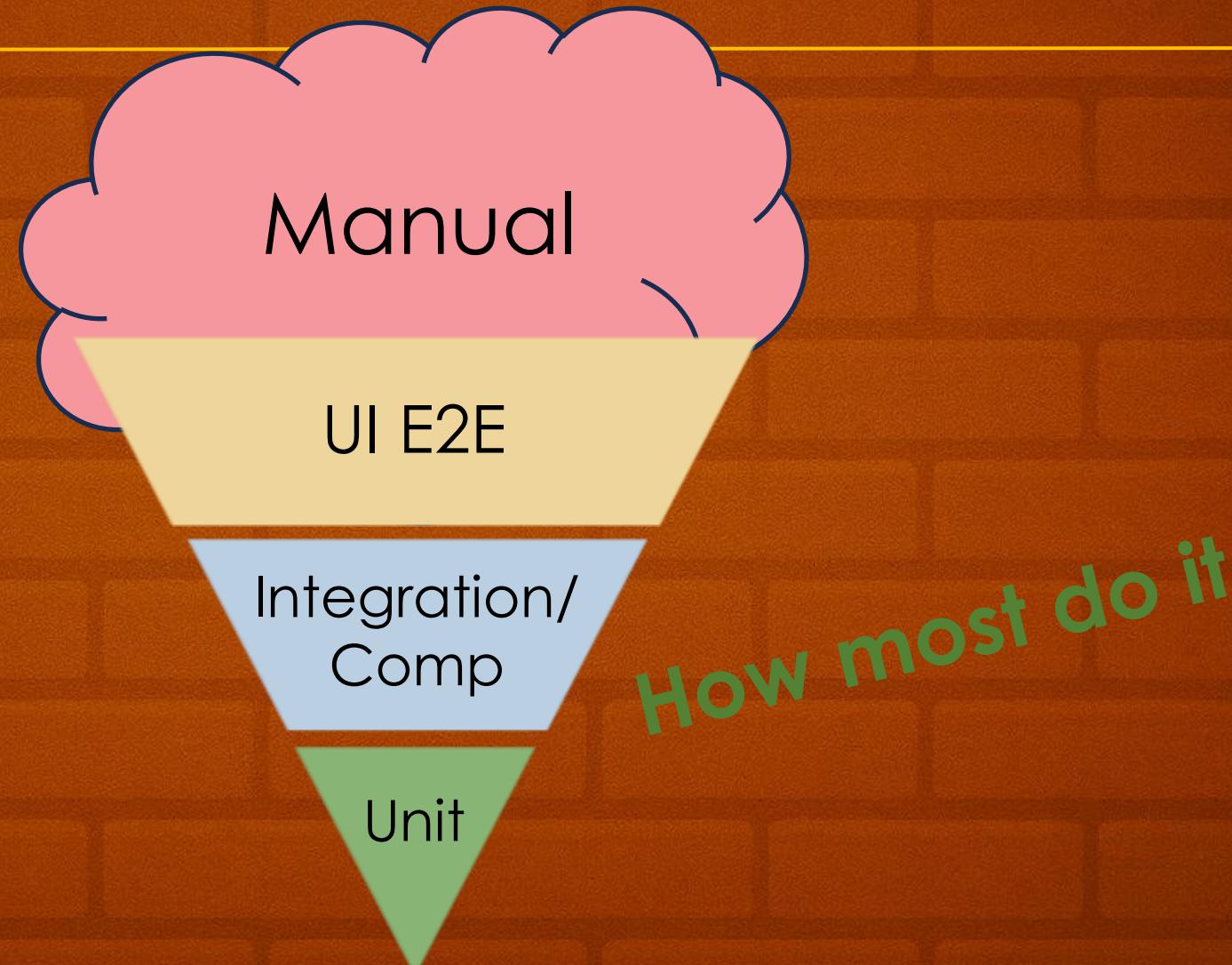


Unhealthy Testing

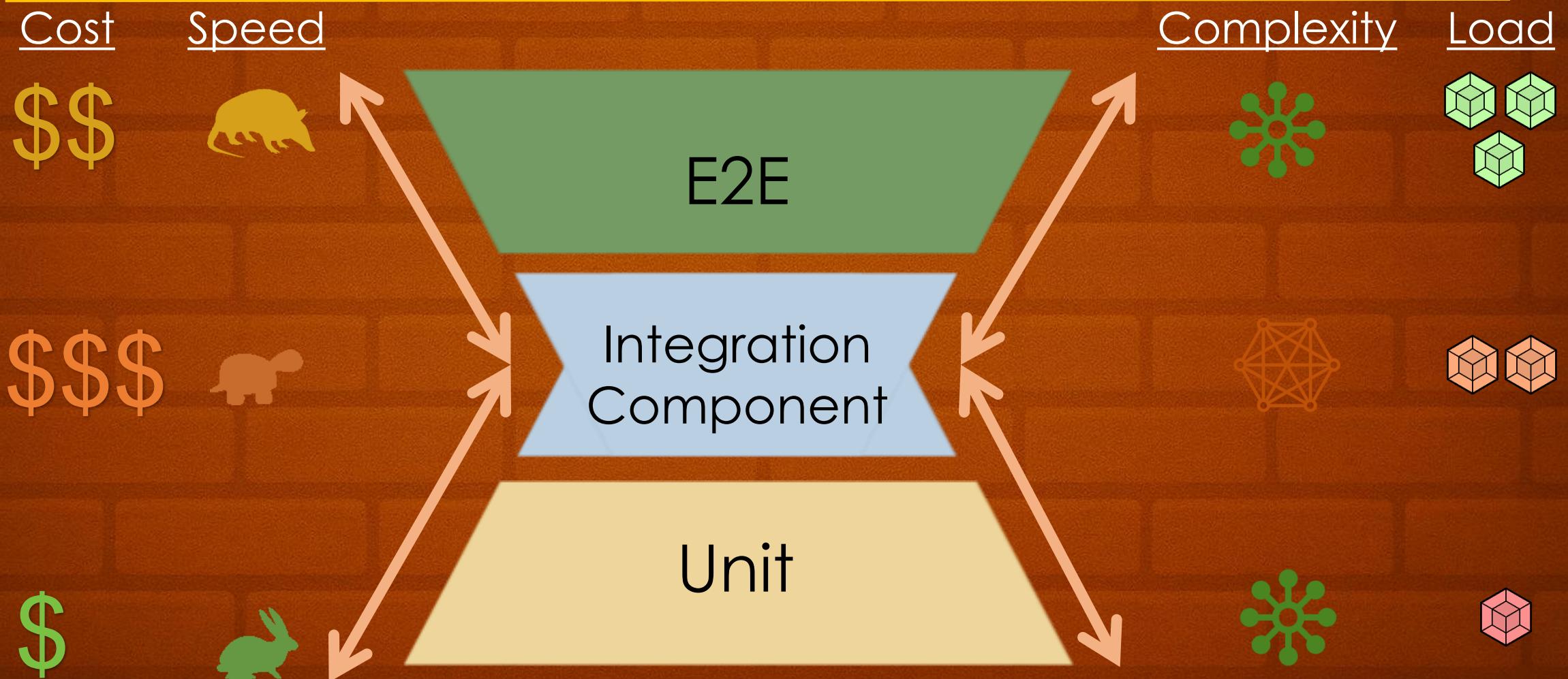


Legacy

Unhealthy Testing II



Danny's Modern Pyramid Healthy Testing

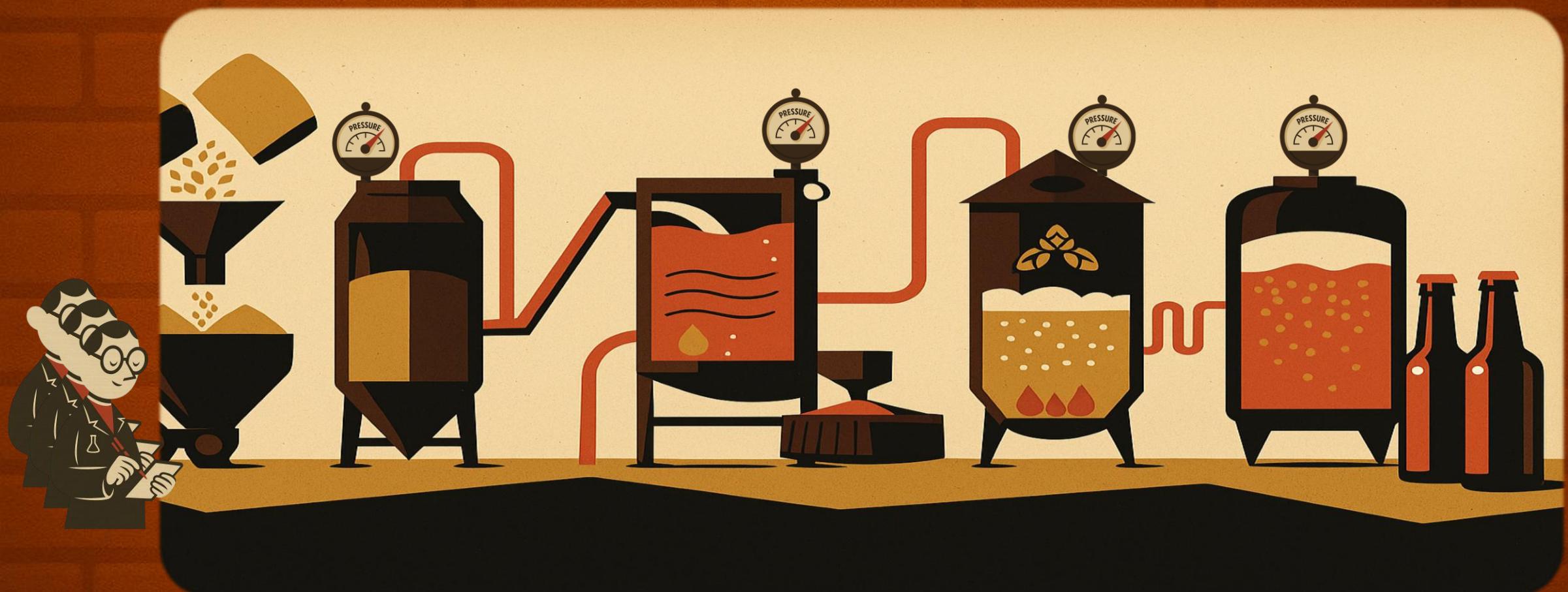




BC
Code

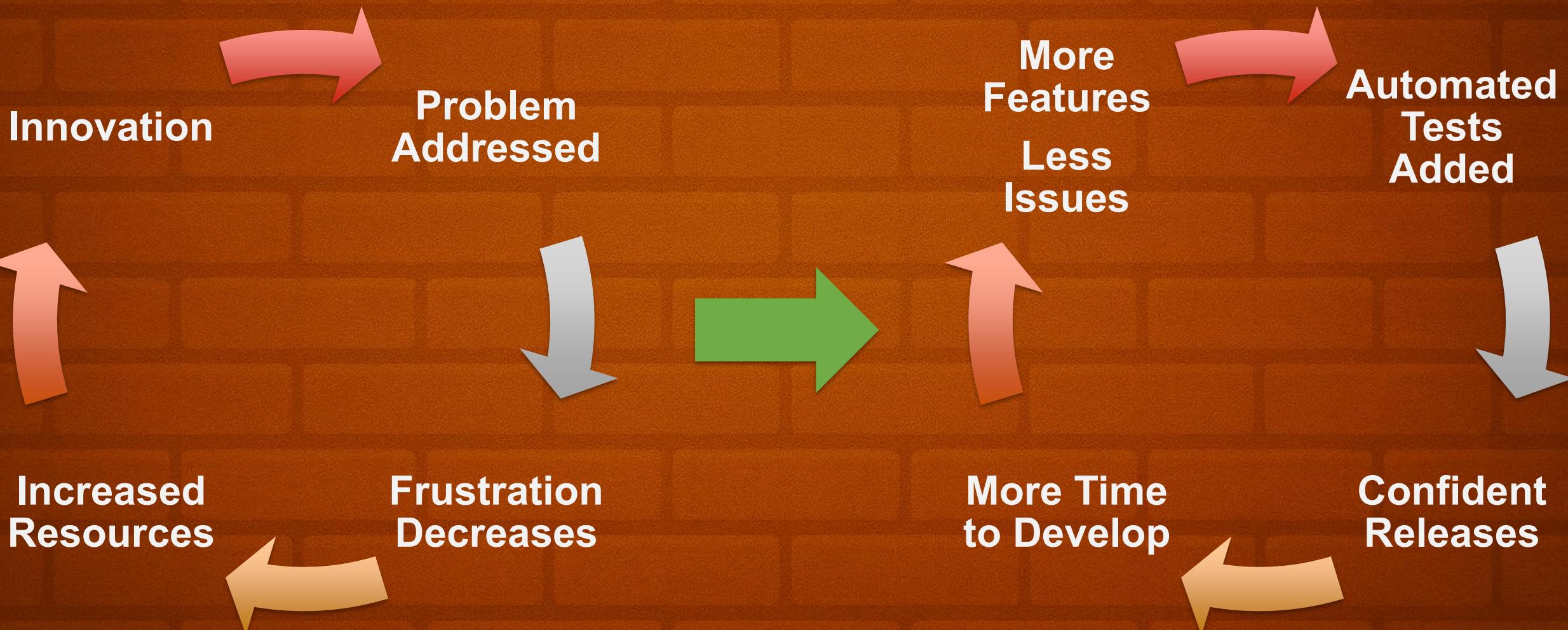
Increasing Confidence

Confident Releases





Confidence Builds On Itself



By the numbers

Case Study

- 6 Months
- 3 Devs
- 80% Code coverage
- 0 UAT bugs found
- 5 Weeks before first error
- <.1% Error rate
- >.5% Baseline error rate
- >5x Better than baseline average



Demo-ish

E2E on Local



WP WicProduct task/wicp-2599-setup-jenkins-ad-hoc-e2e can login with persona cards

Solution package.json login.spec.ts

```
36 test("can login with selector", async ({ page }: PlaywrightTestArgs & PlaywrightTestOptions) : Promise<void> => {
37   await page.getByLabel("Username").click();
38   await page.getRole("option", { name: "admin@wicvp" }).locator("span").click();
39   await page.getRole("button", { name: "Sign in" }).click();
40
41   // Wait until the login has succeeded.
42   const fakeLoginInterceptResponse : Response = await fakeLoginIntercept;
43   expect(fakeLoginInterceptResponse.status()).toBe(200);
44 });
45
46
47
48
49
50
51
52 test("can login with persona cards", async ({ page, isMobile }: PlaywrightTestArgs & PlaywrightTestOptions) : Promise<void> => {
53   // Skip tests conditionally
54   test.skip(isMobile, "This feature is not available on mobile browsers");
55
56   await page.goto("/wic-admin"); - 2 sec, 583 ms
57
58   // Sets up an interceptor to wait for a response from the URL before the request is sent. Do before button click that would send response.
59   // No await yet.
60   const fakeLoginIntercept : Promise<Response> = page.waitForResponse("../api/authentication/FakeLogin"); - 660 ms
61
62   await page.locator("@ wic-persona").filter({ hasText: "admin@wicvp" }).click(); - 485 ms
63
64
65   // Wait until the login has succeeded.
66   const fakeLoginInterceptResponse : Response = await fakeLoginIntercept;
67   expect(fakeLoginInterceptResponse.status()).toBe(200); - 1 ms
68 });
69
70
71
72
73
74
75
76
77
78 }
```

Danny Hearst

wic-product > angular > apps > wic-admin > e2e > login.spec.ts

20°F Cloudy

The screenshot shows a developer's workspace in Visual Studio Code (VS Code) with a dark theme. The top navigation bar displays the project path: WicProduct > task/wicp-2599-setup-jenkins-ad-hoc-e2e. The left sidebar includes icons for Explorer, Commit, PR, Tests, Build, NuGet, Services, Run, Terminal, Problem, and Git.

The Explorer view shows the project structure under 'Solution':

- apps
 - participant-portal
 - vendor-portal
- wic-admin
 - e2e
 - .auth
 - support
 - auth.setup.ts
 - dashboard.spec.ts
 - login.spec.ts
- src
 - .browserslistrc
 - .eslintrc.json
 - build.js
 - jest.config.ts
 - ngsw-config.json
 - playwright.config.ts
 - project.json
 - tailwind.config.js
 - tsconfig.json
 - tsconfig.app.json
 - tsconfig.editor.json
 - tsconfig.spec.json
- dist (no index)
- libs

The package.json tab is selected, showing the following script definitions:

```
"scripts": {  
    "build:participant-portal:azure-dev": "nx build --configuration=azure-dev --project=participant-portal",  
    "test": "nx affected:test --target=test --base=develop --head=HEAD",  
    "test-all": "nx run-many --all --target=test",  
    "lint": "nx run-many --all --target=lint",  
    "lint-ci": "nx run-many --all --target=lint --configuration=ci",  
    "lint-fix": "nx run-many --all --target=lint --fix",  
    "pre-review": "npm run build:wic-admin:prod && npm run build:vendor-portal:prod && npm run build:participant-portal:prod && npm run lint && npm run check-circles",  
    "check-circles": "npx depcruise libs --validate --exclude '^node_modules' --output-type err-long",  
    "e2e": "nx run-many --all --target=e2e",  
    "e2e:participant-portal": "nx e2e participant-portal --skip-nx-cache",  
    "e2e:participant-portal-ui": "nx e2e participant-portal --ui --skip-nx-cache",  
    "e2e:vendor-portal": "nx e2e vendor-portal --skip-nx-cache",  
    "e2e:vendor-portal-ui": "nx e2e vendor-portal --ui --skip-nx-cache",  
    "e2e:wic-admin": "nx e2e wic-admin --skip-nx-cache",  
    "e2e:wic-admin-ui": "nx e2e wic-admin --ui --skip-nx-cache"  
},  
"private": true,  
"dependencies": {  
    "@angular/animations": "18.2.0",  
    "@angular/cdk": "18.2.0",  
    "@angular/common": "18.2.0",  
    "@angular/compiler": "18.2.0",  
    "ng-circular-progress": "0.1.0"  
}
```

The Run view shows the results of running the login.spec.ts test in a browser (iPad Safari). The test passed 39 of 39 tests in 2 minutes and 37 seconds. The playwright test command is also visible.

The bottom status bar indicates the code is 4 spaces indented and in JSON format.

A cartoon illustration of a bald man with a wide smile, flexing his right and left biceps. He is wearing a dark green suit jacket over a white shirt with a red bow tie. The biceps feature a white mask with a red nose and a green mask with a red nose respectively. He is standing in front of a brown brick wall with several green hop cones hanging from the top. In the bottom left corner, there is a portion of a wooden barrel.

Demo-ish

E2E on Jenkins

BC
Code

EGLE - MiHDWIS - Activity Hist X All [Wic Product] - Jenkins +

jenkins.kunzleight-hosted.com/job/Wic%20Product/ ⚡ ☆ 🔍 🌐

KL&A Housekeeping Water WIC Product 📂 | KL&A Water WIC Automated Testing Utils

Jenkins

Dashboard > Wic Product >

Status Configure + New Item Build History Add description

All +

S	W	Name ↴	Last Success	Last Failure	Last Duration	F	# Issues
✓	☀️	Create_Azure_Seed	3 mo 3 days #16	N/A	8 min 49 sec	▶	☆ -
✓	☀️	Demo-Deploy	8 days 0 hr #35	5 mo 6 days #23	26 min	▶	☆ 30
✓	☁️	Dev Deploy	1 day 4 hr #311	2 days 0 hr #310	28 min	▶	☆ 31
✗	☁️	E2E	1 mo 2 days #14	28 days #15	8 min 25 sec	▶	☆ -
✓	☁️	LinkDeploy	3 mo 10 days #8	3 mo 10 days #7	26 min	▶	☆ 32
✓	☀️	Local Azure DB Refresh	7 days 3 hr #55	N/A	7 min 21 sec	▶	☆ -
✗	☁️	Nightly-Cache-Build	1 day 13 hr #46	13 hr #47	23 min	▶	☆ -
✓	☀️	NpmCacheRefresh	13 hr #293	N/A	6 min 11 sec	▶	☆ -
✓	☁️	Publish WicProduct	5 mo 25 days #46	5 mo 25 days #45	5 min 3 sec	▶	☆ -
✓	☁️	Publish WicProduct DEV	(recording)	Stop Recording	5 min 41 sec	▶	☆ -
✓	☁️	Pull-Request-Builder	54 min #839	16 hr #836	25 min	▶	☆ 88

Build Queue: No builds in the queue.

Build Executor Status:

- Devops-Windows-Agent 0/1
- azure-devops-agent-large42c700 0/1
- azure-lps-ci-agent-2021150 0/1

20°F Cloudy 3:19 PM 2/14/2025



BC
Code

Getting Started on Your Team

Follow the Epic Journey

Setup E2E Automated Testing

- Setup Playwright framework
- Create basic tests from login page
- Create CI E2E jobs
- Develop shared patterns

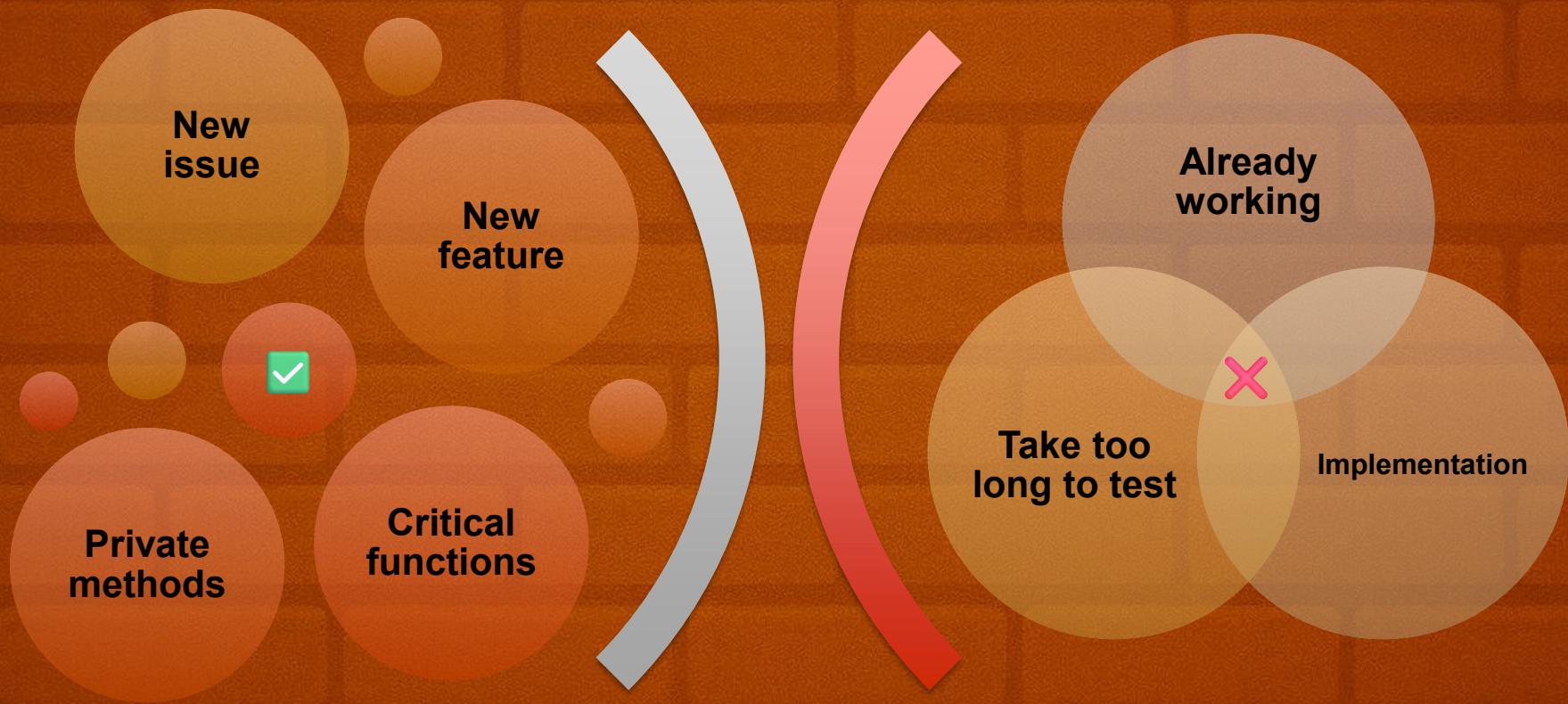




BC
Code

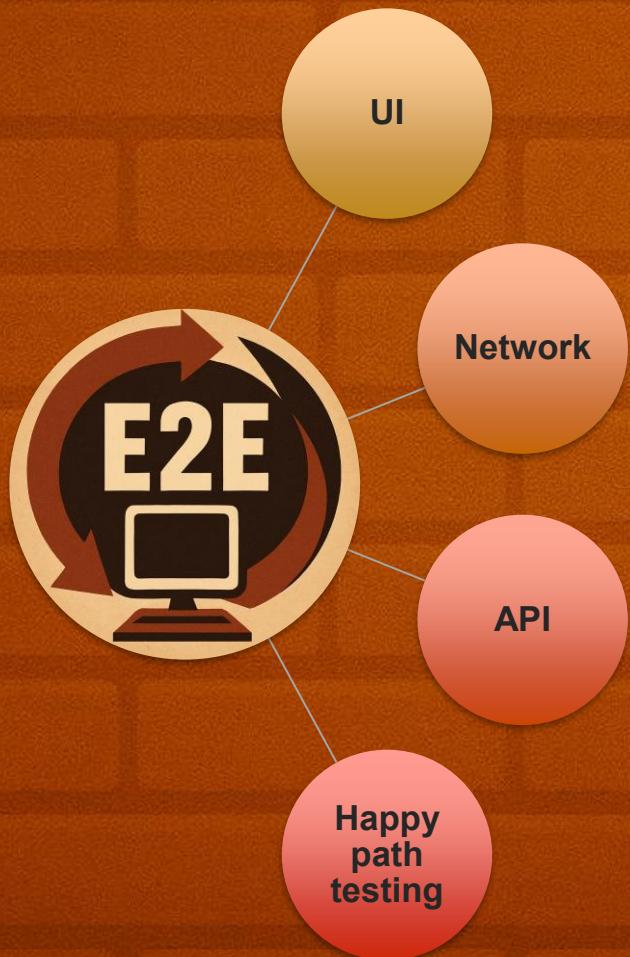
Testing Best Practices

Automated Testing Focus





E2E Testing Focus



E2E Testing Focus

Tests should be quick

5-15 minutes per test to write

More granular test or less complex

Use google

30 seconds or less per test to run

Investigate slow system or an alternative testing methods

`test.slow()`

Tests should be effective

No testing debt

Test user behavior

Fail fast

If they are no longer useful, get rid of them!

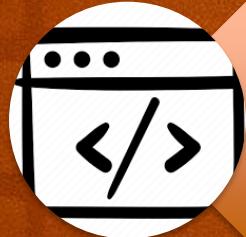
Not implementation details

Fail at the non-okay response rather than waiting for a timeout

Test Driven Development



Testable Code



Design before implementation



No skipping tests



E2E Testing With Playwright

Tests written like Unit Tests

- Typescript
- Assertions
- Async and synchronous checks
- No waiting by time

Content Interactions

- CSS Selectors - Role/Accessibility
- Session Shared Authentication
- Mature features and content

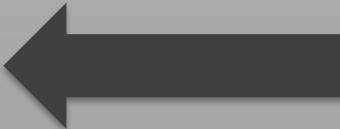


Playwright Tools



**Browser
plugin**

*Playwright
CRX*



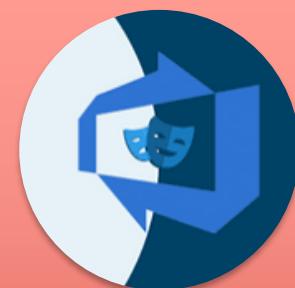
IDE plugins



Trace Viewer



UI Runner



**Playwright
MCP**





Playwright Mobile Testing

Important and easy

- So easy to miss mobile bugs
- Be the hero 🦸

Responsive design helps accessible design

- ADA & WCAG compliance teams will thank you

Recommended mobile safari

- iPad & iPhone

Recommended mobile chromium

- Pixel

```
projects: [
  // Device setup
  { name: "setup", testMatch: /\.+\.\.setup\.\.ts/ },
  {
    name: "Mobile Chromium",
    use: { ...devices["Pixel 5"] },
    dependencies: ["setup"],
  },
  {
    name: "Mobile Safari",
    use: { ...devices["iPhone 12"] },
    dependencies: ["setup"],
    timeout: 60000,
  },
  {
    name: "iPad Safari",
    use: { ...devices["iPad (gen 7)"] },
    dependencies: ["setup"],
    timeout: 60000,
  },
],
```

“



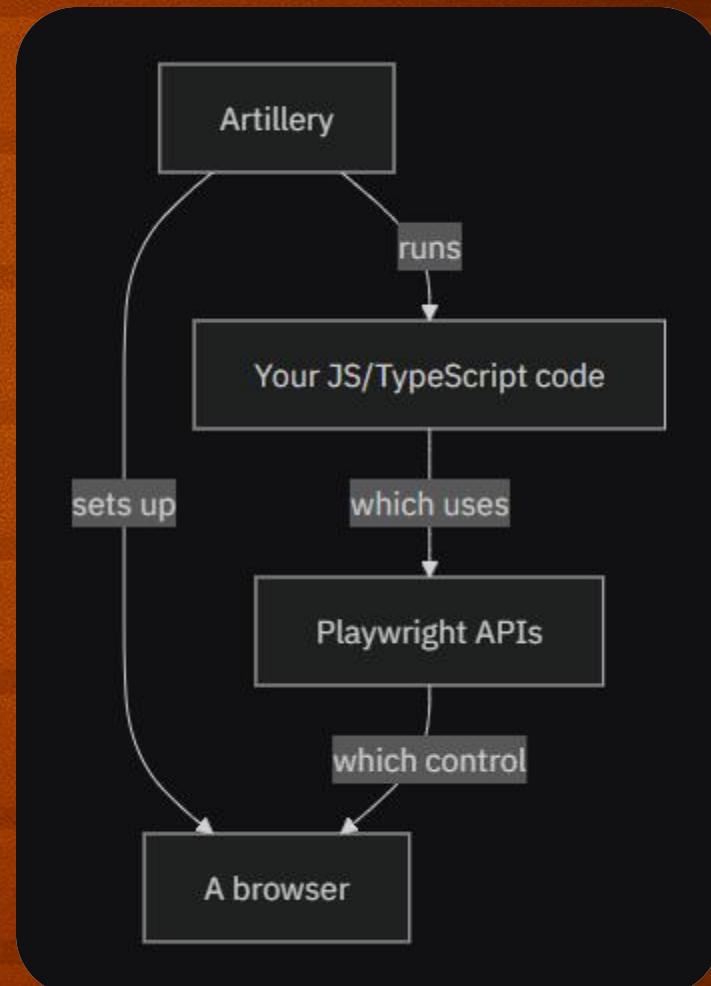
Playwright Load Testing

Artillery.io

- <https://www.artillery.io/docs/reference/engines/playwright>
- <https://www.artillery.io/docs/playwright>

With Artillery's Playwright engine you can

- Create load tests for complex web apps using Playwright's API
- Reuse existing Playwright code for load testing
- Track Core Web Vitals metrics automatically and measure how they change under load
- Create no-code tests with playwright codegen
- Scale to tens of thousands of headless Chrome instances with AWS Fargate or Azure ACI





E2E Suggested Habits

Suggested Dev Habits

- Check nightly CI runs
- Keep E2E watching while coding

Suggested BA Habits

- Suggest automated tests be written for repeatedly troubled areas

Not a replacement
for manual testing



Call to Action - Make your choice

Automated Testing

Repeatable tests

Work on new features

Automatic bug catching

Confident releases

Manual Only Testing

Bug fixes

Debugging

Prod maintenance

The difference between automated tests and not, is what you will be doing the following months of release.

Making new features?

Or prod maintenance, debugging and bug fixes?

Which would you rather do?





Questions and Resources

