

```
describe('Testing Cypress.io') {
  it('Testing Cypress.io', () => {
    cy.visit('https://cypress.io')
  })
}
```

E2E TEST Your App With **cypress.io**





In this presentation:

- What are E2E tests?
- How Cypress solves the problems old frameworks face
- How to start writing Cypress today

What is e2e testing?

0

A: Test an independent unit of code

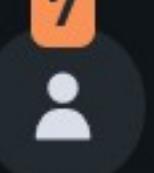
2

B: Test the interaction between units of code

4

C: Test a site's functionality like a real user would. I don't know - tell me!

1



Selenium



Injecting JavaScript

into web pages

HTML



JS

```
31 self.file = None
32 self.fingerprints = set()
33 self.logdups = True
34 self.debug = debug
35 self.logger = logging.getLogger(__name__)
36 if path:
37     self.file = open(path, 'w')
38 self.file.write('')
39 self.file.close()
40 self.fingerprints = set()
41
42 @classmethod
43 def from_settings(cls, settings):
44     debug = settings.getbool('debug')
45     return cls(job_dir(settings), debug)
46
47 def request_seen(self, request):
48     fp = self.request_fingerprint(request)
49     if fp in self.fingerprints:
50         return True
51     self.fingerprints.add(fp)
52     if self.file:
53         self.file.write(fp + os.linesep)
54
55 def request_fingerprint(self, request):
56     return request_fingerprint(request)
```



MakeAGIF.com

more dynamic = less reliable

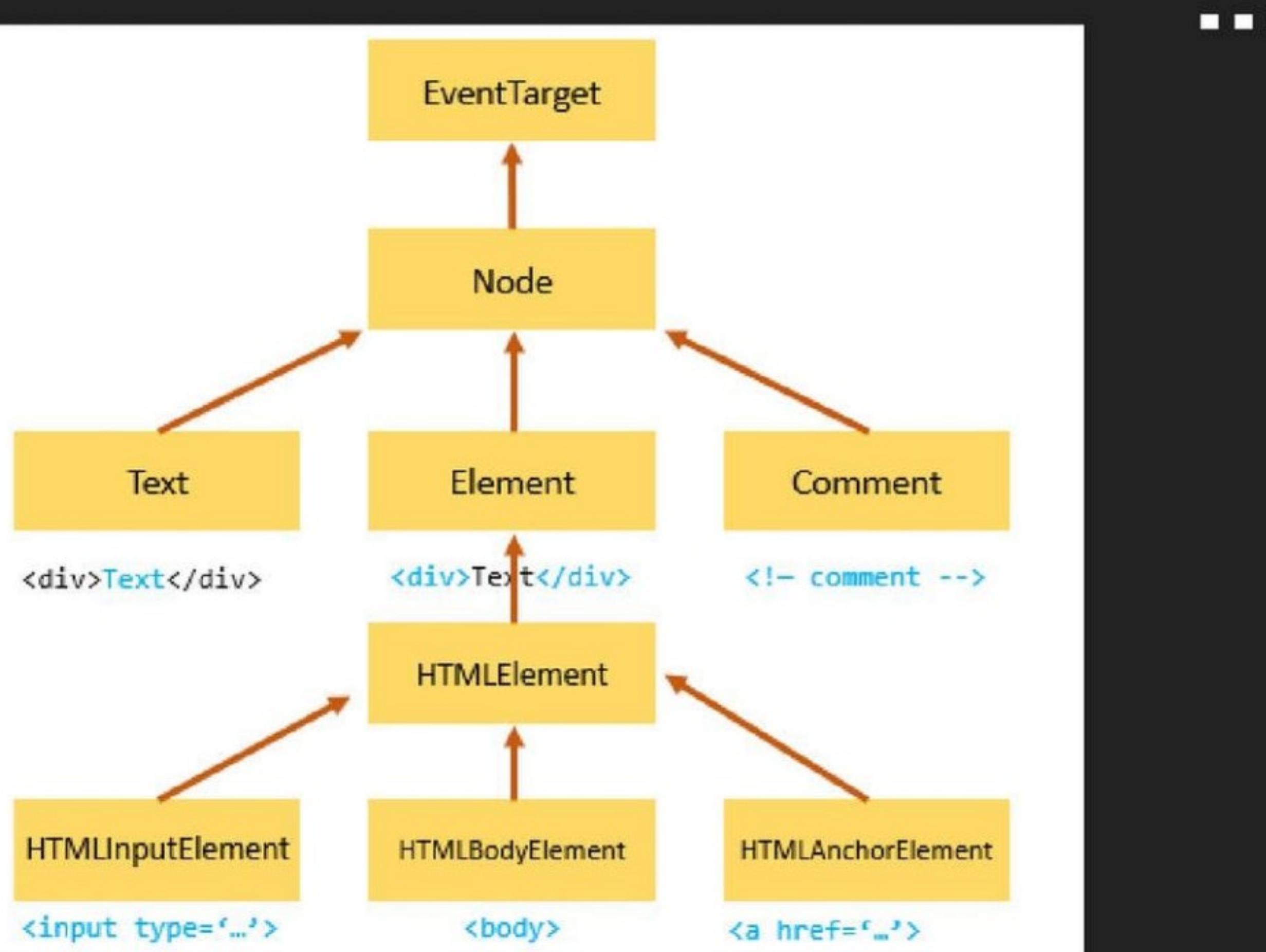
Designers



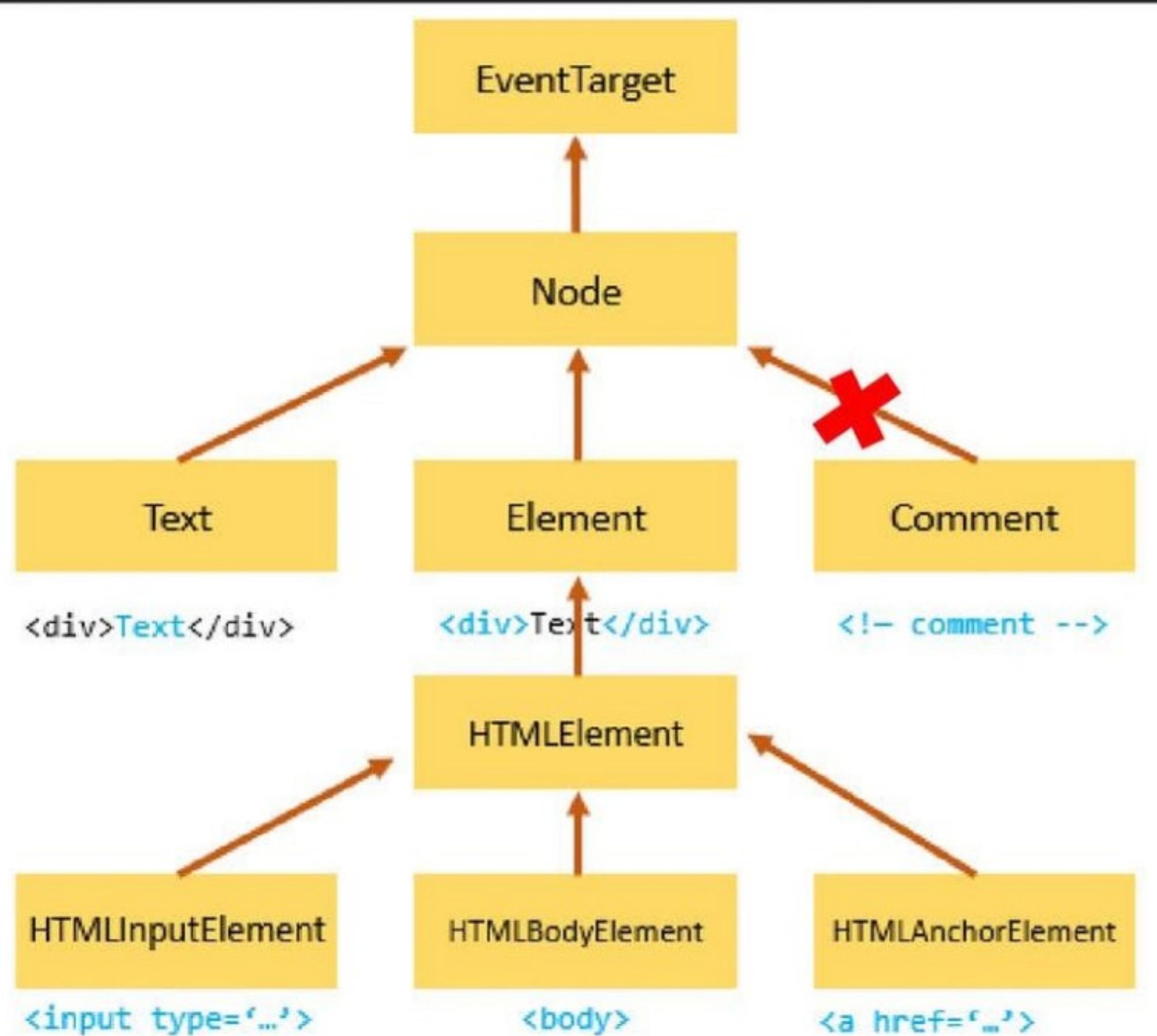
Testers



findElementById("comment")

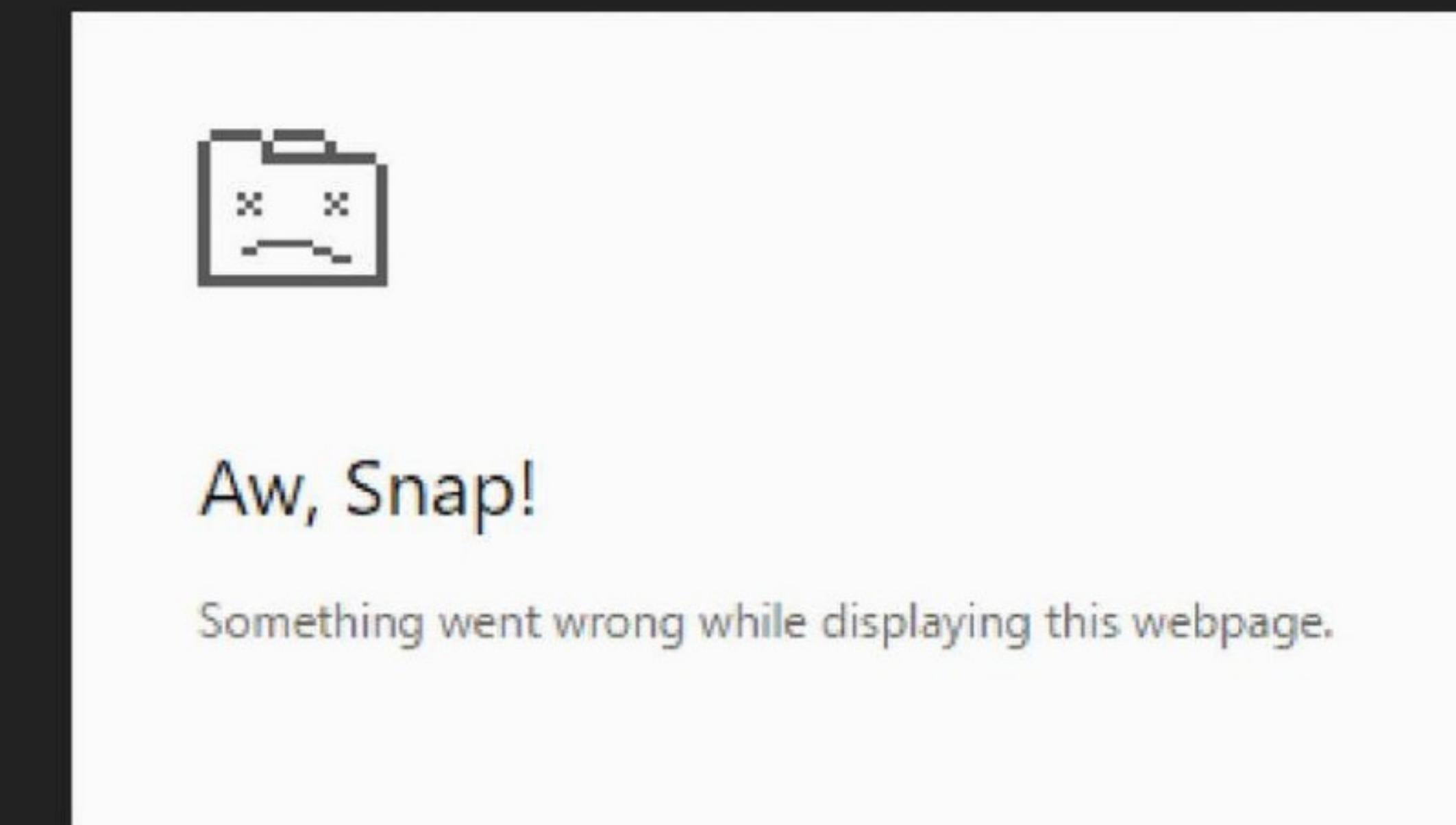


findElementById("comment")



...

element.click()



game-of-life #6 Test × bogo-devops/game-of ×

localhost:8080/me/my-views/view/All/job/game-of-life/lastCompletedBuild

Jenkins

search K Hong | log out

Jenkins ▶ K Hong ▶ My Views ▶ All ▶ game-of-life ▶ #6 ▶ Test Results [ENABLE AUTO REFRESH](#)

[Back to Project](#)

[Status](#)

[Changes](#)

[Console Output](#)

[Edit Build Information](#)

[History](#)

[Polling Log](#)

[Git Build Data](#)

[No Tags](#)

[Test Result](#)

[Previous Build](#)

Test Result

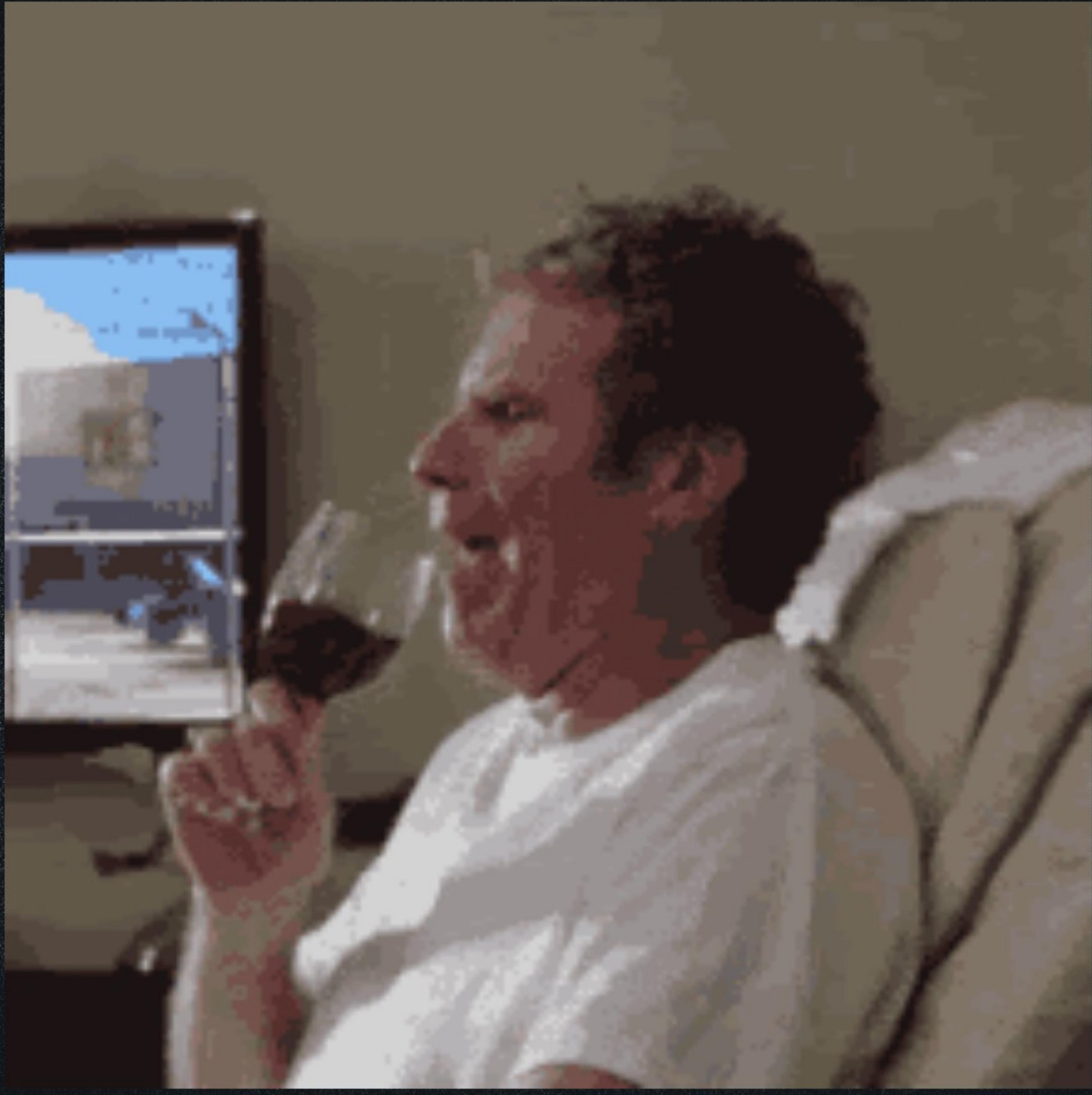
35 failures (+35) 

49 tests (-9)
Took 0.87 sec.
[add description](#)

All Failed Tests

Test Name
+ com.wakaleo.gameoflife.domain.WhenYouCreateACell.aLiveCellShouldBePrinted
+ com.wakaleo.gameoflife.domain.WhenYouCreateACell.aLiveCellSymbolShouldBe
+ com.wakaleo.gameoflife.domain.WhenYouCreateACell.aLiveCellShouldBeRepre
+ com.wakaleo.gameoflife.domain.WhenYouCreateAGrid.shouldBeAbleToCountLive
+ com.wakaleo.gameoflife.domain.WhenYouCreateAGrid.shouldBeAbleToReadThe
+ com.wakaleo.gameoflife.domain.WhenYouCreateAGrid.theGridContentsAsAnArra

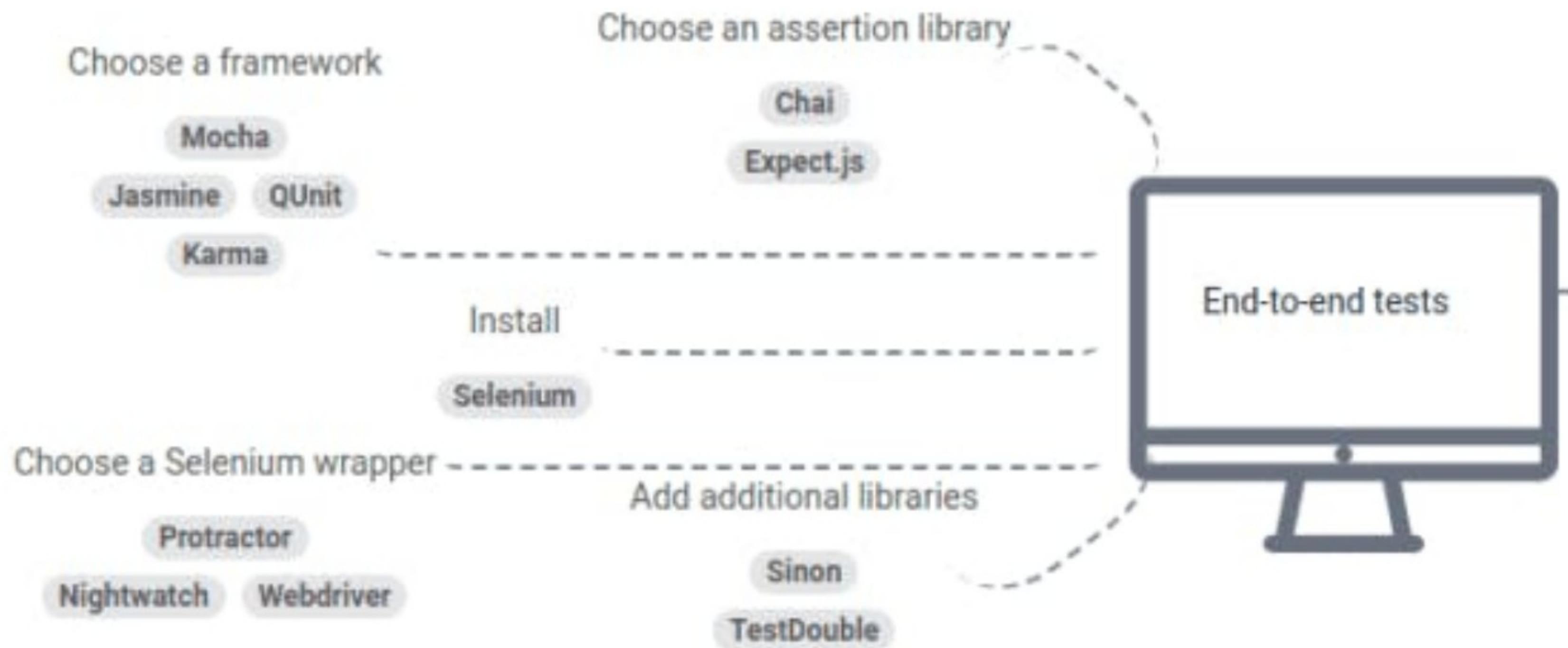
This is very common 



CYPRESS

WOHOO





All-in-one testing framework,
assertion library, with mocking and
stubbing, all without Selenium.

ILLUSTRATION FROM: [HTTPS://WWW.CYPRESS.IO/HOW-IT-WORKS/](https://www.cypress.io/how-it-works/)

All features in one framework

But ease of use is not even the best part!



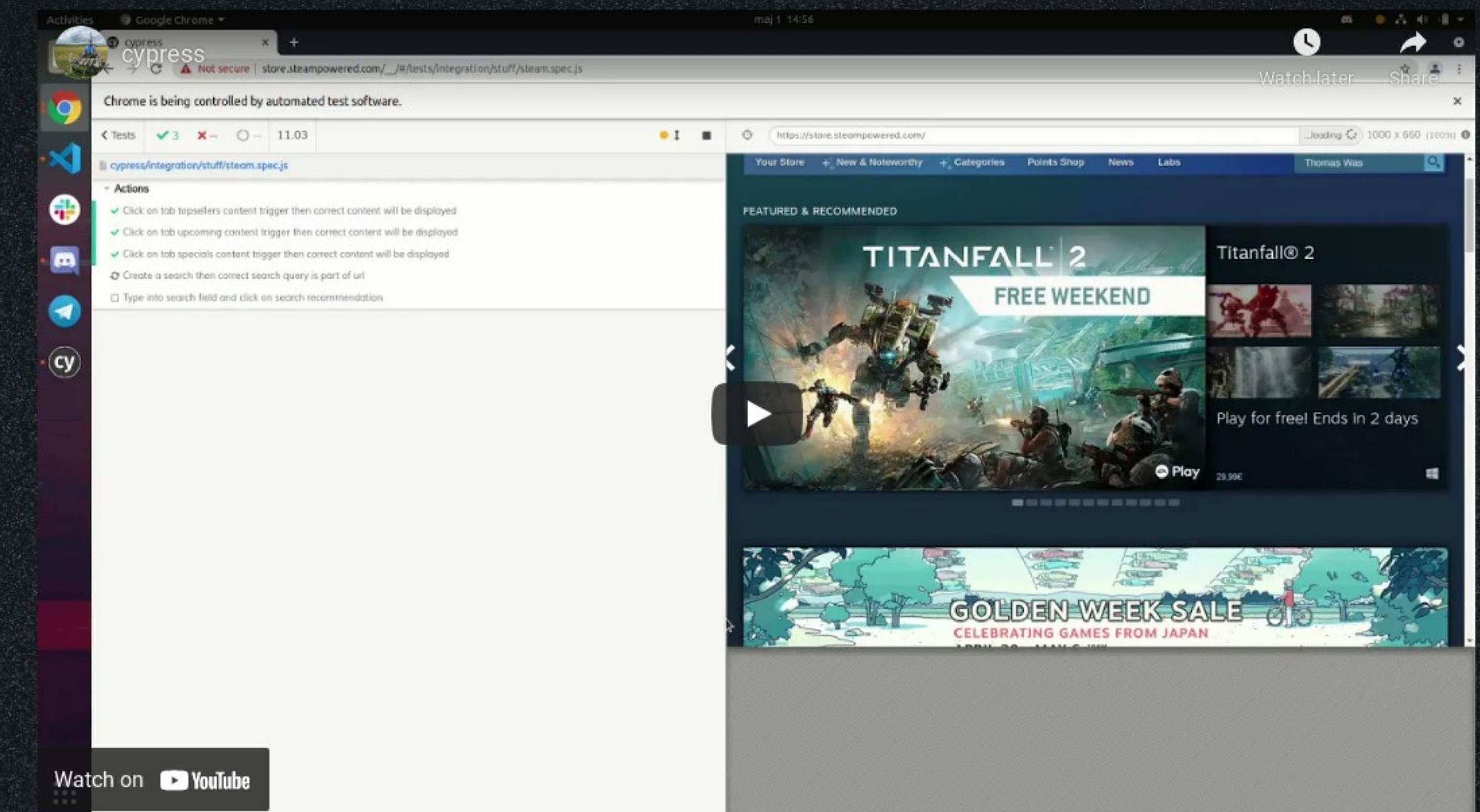
JS everything

Cypress is built using a mix of battle-tested testing libraries and novel software design. All in JavaScript.

Run fast!



Not flaky!





Cypress has a dedicated team and
a great community

Get up to speed quickly:

- yarn add cypress -d
- yarn run cypress open

```
describe('My First Test', () => {
  it('Does not do much!', () => {
    expect(true).to.equal(false)
  })
})
```

Testing the Steam store

www.store.steampowered.com/

We are testing:

- Content offering tabs
- Store search
- Store search recommendations

```
steam.spec.js ✘
steam.spec.js > context("Actions") callback > it("Click on tab content trigger then correct content will be displayed") callback
1  // 
2
3  context("Actions", () => {
4    beforeEach(() => {
5      cy.visit("https://store.steampowered.com/");
6    });
7
8    it("Click on tab content trigger then correct content will be displayed", () => {
9      cy.get("#tab_topsellers_content").should("be.hidden");
10     cy.get("#tab_topsellers_content_trigger").click();
11     cy.get("#tab_topsellers_content").should("not.be.hidden");
12   });
13
14  it("Create a search then correct search query is part of url", () => {
15  });
16})
```



The screenshot shows a video player interface with a dark theme. At the top right are 'Watch later' and 'Share' buttons. The main area displays a code editor with a file named 'steam.spec.js'. The code is written in JavaScript and uses the Cypress framework to test interactions with the Steam store. A yellow dot on the status bar indicates the current line of execution. The status bar also shows file information (steam.spec.js), encoding (UTF-8), line endings (LF), language (JavaScript), duration (1h 1m), and other settings like Chronicer and Prettier.

```
File Edit Selection View Go Run Terminal Help
steam.spec.js X
File Edit Selection View Go Run Terminal Help
steam.spec.js > context("Actions") callback > it("Type into search field and click on search recommendation") callback
1 //<reference types="cypress" />
2
3 context("Actions", () => {
4   beforeEach(() => {
5     cy.visit("https://store.steampowered.com/");
6   });
7
8   it('Click on tab content trigger then correct content will be displayed', () => {
9     cy.get('#tab_topsellers_content').should('be.hidden');
10    cy.get('#tab_topsellers_content_trigger').click();
11    cy.get('#tab_topsellers_content').should('not.be.hidden');
12  });
13
14 it("Create a search then correct search query is part of url", () => {
15   cy.get("#store_nav_search_term").type("Thomas Was Alone{enter}");
16   cy.url().should("include", "Thomas+Was+Alone");
17 });
18
19 it("Type into search field and click on search recommendation", () => {
20   cy.get("#store_nav_search_term").type("Thomas Was Alone");
21   cy.get("#searchterm_options");
22 });
23 });
24
```

Watch on YouTube

CREATING TESTS FOR THE STEAM STORE

```
describe("landing page", () => {  
  it("Search for non-existing item return nothing", () => {  
    cy.get("#login-email-input").type("user");  
    cy.get("#login-password-input").type("pass");  
    cy.get("#login-submit").click();  
  ...  
});  
  
it("Search for existing item return result", () => {  
  cy.get("#login-email-input").type("user");  
  cy.get("#login-password-input").type("pass");  
  cy.get("#login-submit").click();  
  ...  
});  
});
```

This is slow!

(Even if we write it in a command or in beforeEach)

You want to set application state programmatically



You can design your app for testability

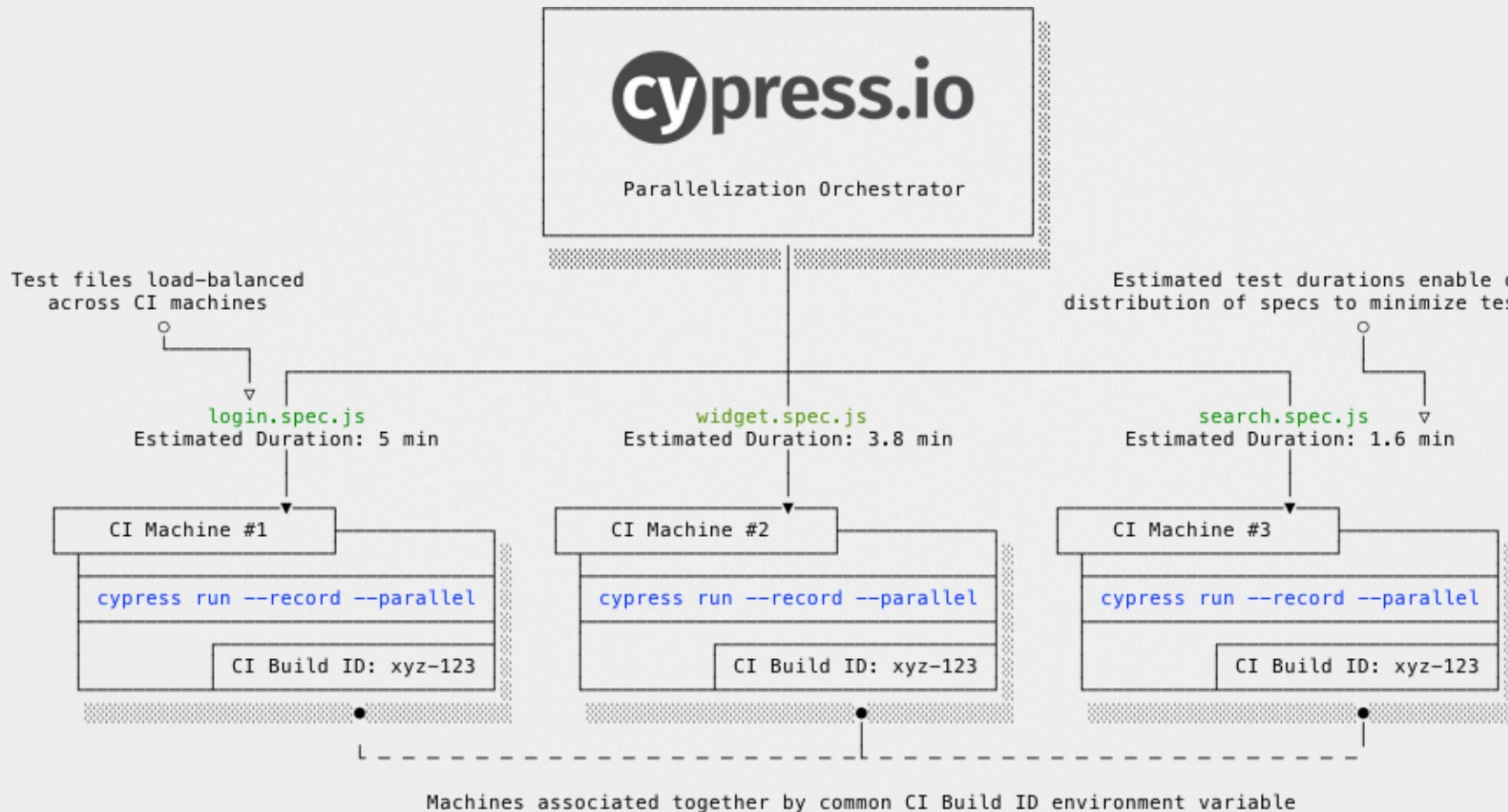
```
// index.js
if (process.env.NODE_ENV === 'test'){
  dbURI = process.env.TEST_DB_URI
}
db.connect(dbURI);
```

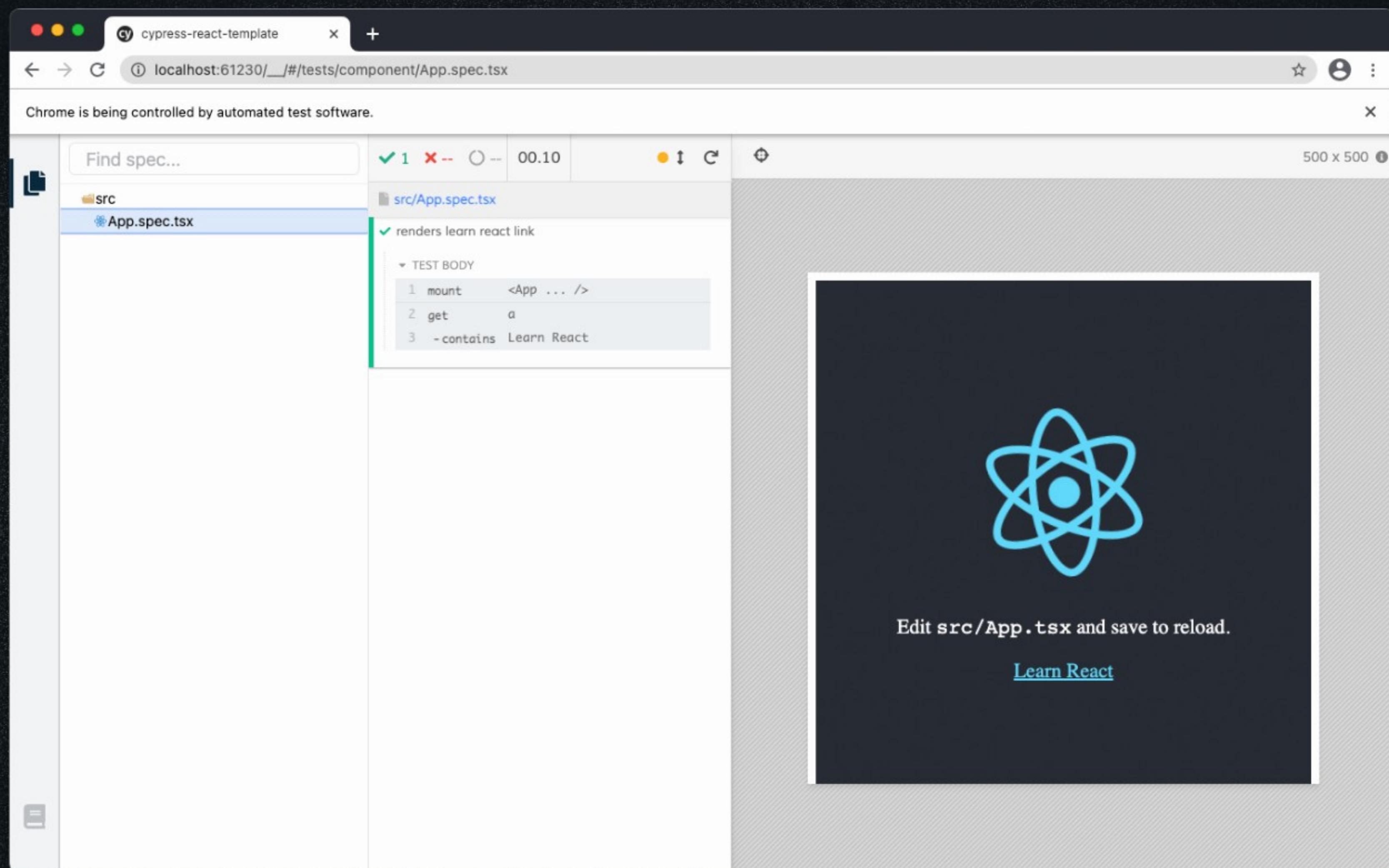
```
// cypress/plugins/index.js
module.exports = async (on, config) => {
  const db = await DB.connect();
  on('task', {
    async createUser(data) return db.insert(data);
  })
}
```

```
// docker-compose.test.yml
app:
  build: Dockerfile.test
  environment:
    - NODE_ENV=test
    - TEST_DB_URI=${DB_USER}@db:5432
  depends_on:
    - db
db:
  environment:
    - DB_USER
  ports:
    - 5432:5432
```

Directly in your application, test, and environment configuration







Cypress testing a React component directly



More things plugins help you do:

1. Create snapshot tests
2. Get the code-coverage of your app
3. Integrate Cypress with Slack and Jest
4. More. New plugins are released all the time.



CYPRESS: 3 to Take Home

1. e2e tests are **important** to be able to **assure the business logic and user flow** of any application
2. Selenium is outdated. **Cypress is easy!**
3. Just do a **npm install cypress --save-dev** and get up to speed with Cypress today!

Thank you

& Happy testing!

