RUN TESTS ON CI

🎓 YOU WILL LEARN

- running Cypress E2E tests on CI
  - 📚 https://on.cypress.io/ci
- running Playwright E2E tests on CI
  - 📚 https://playwright.dev/docs/ci

CYPRESS

Q: How would you run Cypress E2E tests on GitHub Actions?

- 💡 use cypress-io/github-action
- Example repo https://github.com/bahmutov/cy-vs-pw-ci-example

```yaml
name: cypress tests
on: push
jobs:
  e2e:
    runs-on: ubuntu-22.04
    steps:
      - name: get the code
        uses: actions/checkout@v4
      - name: run tests
        uses: cypress-io/github-action@v6
        with:
          start: npm start
```

.github/workflows/ci.yml

```
66  ──────────────────────────────────────────────────────────────────────────────
67
68    Running:  routing.cy.js                                   (1 of 1)
74
75    App routing
76  resetting database
77  POST /reset 200 14.022 ms - 2
78  GET /todos 200 5.147 ms - 229
79      ✓ shows all, completed, or incomplete todos (831ms)
80    1 passing (884ms)
85    (Results)
86    ┌──────────────────────────────────────────────────────────────────────────┐
87    │ Tests:        1                                                          │
88    │ Passing:      1                                                          │
89    │ Failing:      0                                                          │
90    │ Pending:      0                                                          │
91    │ Skipped:      0                                                          │
92    │ Screenshots:  0                                                          │
93    │ Video:        true                                                       │
94    │ Duration:     0 seconds                                                  │
95    │ Spec Ran:     routing.cy.js                                              │
96    └──────────────────────────────────────────────────────────────────────────┘
97    (Video)
98    -  Video output: /home/runner/work/cy-vs-pw-ci-example/cy-vs-pw-ci-example/cypress/videos/routing.cy.js.mp4
```

```yaml
- name: run tests
  uses: cypress-io/github-action@v6
  with:
    start: npm start
```

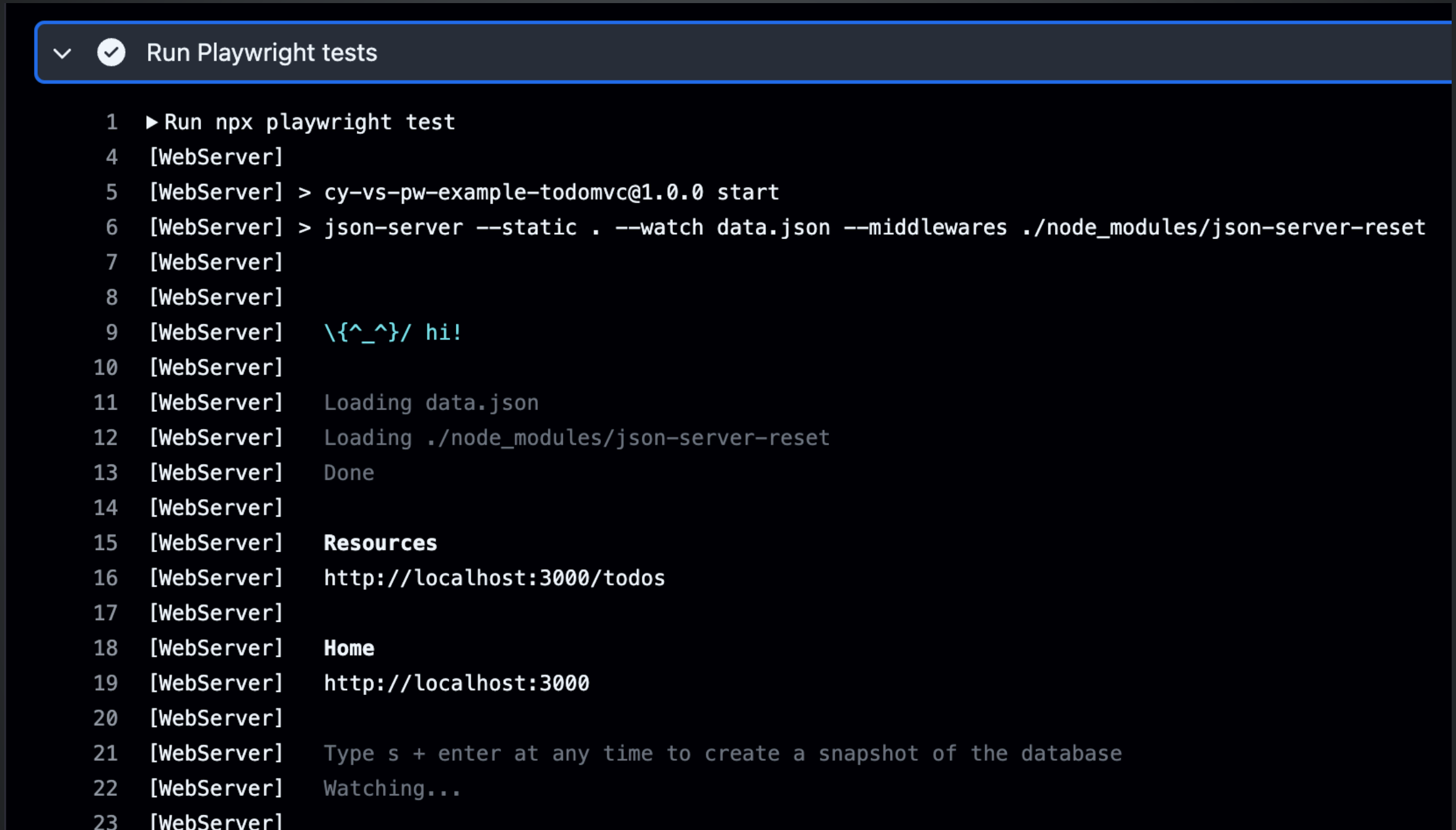
The above YML does it all: installs dependencies, caches them, starts the app, runs the tests, etc.

PLAYWRIGHT

Q: How would you run Pw E2E tests on GitHub Actions?

- 💡 check https://playwright.dev/docs/ci

```yaml
name: playwright tests
on: push
jobs:
  e2e:
    runs-on: ubuntu-22.04
    steps:
      - name: get the code
        uses: actions/checkout@v4
      - uses: actions/setup-node@v4
        with:
          node-version: 20
      - name: Install dependencies
        run: npm ci
      - name: Install Playwright Browsers
        run: npx playwright install --with-deps
      - name: Run Playwright tests
```


Run Playwright tests
1 ▶ Run npx playwright test
4 [WebServer]
5 [WebServer] > cy-vs-pw-example-todomvc@1.0.0 start
6 [WebServer] > json-server --static . --watch data.json --middlewares ./node_modules/json-server-reset
7 [WebServer]
8 [WebServer]
9 [WebServer] \{^_^}/ hi!
10 [WebServer]
11 [WebServer] Loading data.json
12 [WebServer] Loading ./node_modules/json-server-reset
13 [WebServer] Done
14 [WebServer]
15 [WebServer] Resources
16 [WebServer] http://localhost:3000/todos
17 [WebServer]
18 [WebServer] Home
19 [WebServer] http://localhost:3000
20 [WebServer]
21 [WebServer] Type s + enter at any time to create a snapshot of the database
22 [WebServer] Watching...
23 [WebServer]

CONTROL THE ENVIRONMENT

Q: How to run the tests on the same OS / browser?

- 💡 Use Cypress official Docker image https://hub.docker.com/r/cypress/browsers/tags
- 💡 Use MS official Docker image https://playwright.dev/docs/docker

CYPRESS DOCKER EXAMPLE

```yaml
runs-on: ubuntu-22.04
# Cypress Docker image from https://hub.docker.com/r/cypress
# with browsers pre-installed
container:
  image: cypress/browsers:node-20.17.0-chrome-129.0.6668.70-1-ff-130.0.1-edge-129.0.2
steps:
  - uses: actions/checkout@v4
  - uses: cypress-io/github-action@v6
    with:
      browser: chrome
```

PLAYWRIGHT DOCKER EXAMPLE

```
jobs:
  playwright:
    name: 'Playwright Tests'
    runs-on: ubuntu-22.04
    container:
      image: mcr.microsoft.com/playwright:v1.47.2-jammy
```

STORE TEST ARTIFACTS

Screenshots (Cy), videos (Cy), test traces (Pw)

STORE CYPRESS SCREENSHOTS AND VIDEOS ON CI

```yaml
- uses: cypress-io/github-action@v6
# after the test run completes store videos and any screenshots
- uses: actions/upload-artifact@v4
  with:
    name: cypress-screenshots
    path: cypress/screenshots
    if-no-files-found: ignore
- uses: actions/upload-artifact@v4
  with:
    name: cypress-videos
    path: cypress/videos
    if-no-files-found: ignore
```

← cypress tests

save artifacts #2

Re-run all jobs ···

- Summary

Jobs

- e2e

Run details

- Usage
- Workflow file

Triggered via push 3 minutes ago	Status	Total duration	Artifacts
bahmutov pushed 7ab2c37 main	Success	36s	1

cy-e2e.yml
on: push

e2e 28s

e2e summary

Cypress Results

Result	Passed ✅	Failed ❌	Pending ✋	Skipped ↩️	Duration 🕘
Passing ✅	1	0	0	0	0.867s

Job summary generated at run-time

Artifacts

Produced during runtime

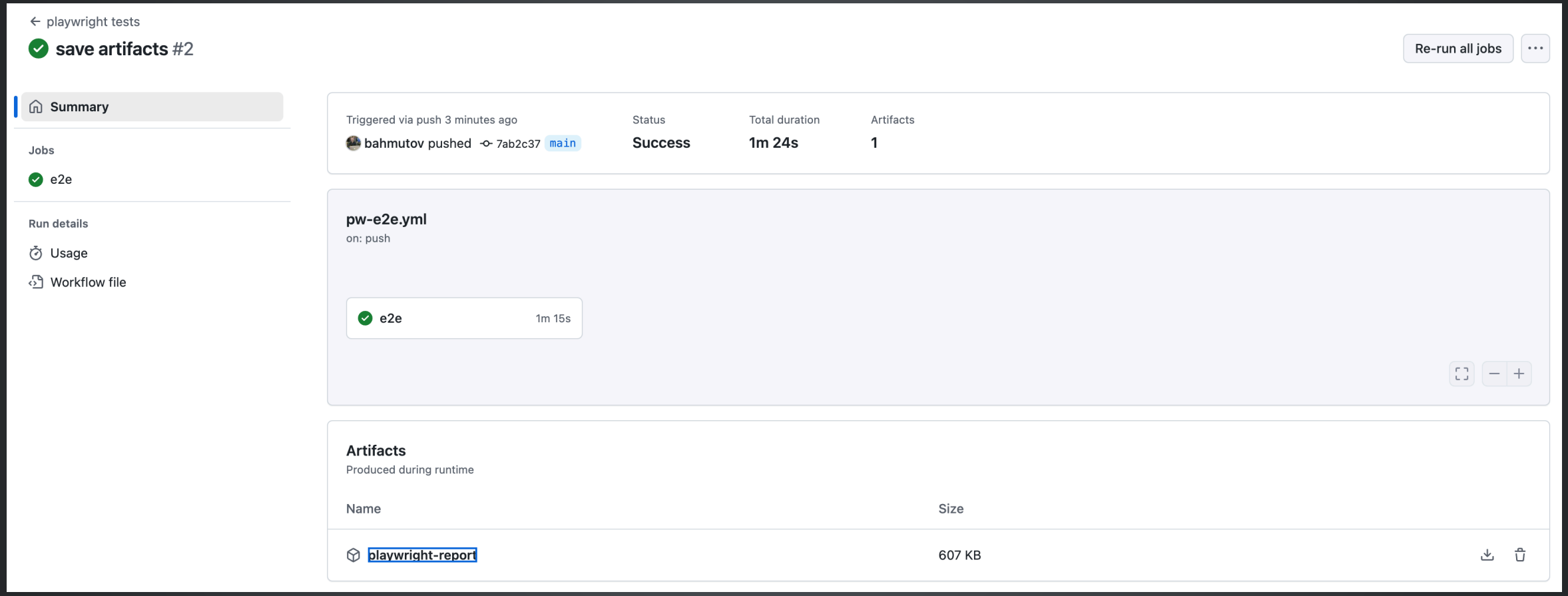
Name	Size
cypress-videos	245 KB

STORE PLAYWRIGHT TEST REPORTS AND TRACES

```yaml
- name: Run Playwright tests
  run: npx playwright test
- uses: actions/upload-artifact@v4
  if: ${{ !cancelled() }}
  with:
    name: playwright-report
    path: playwright-report/
    retention-days: 30
```

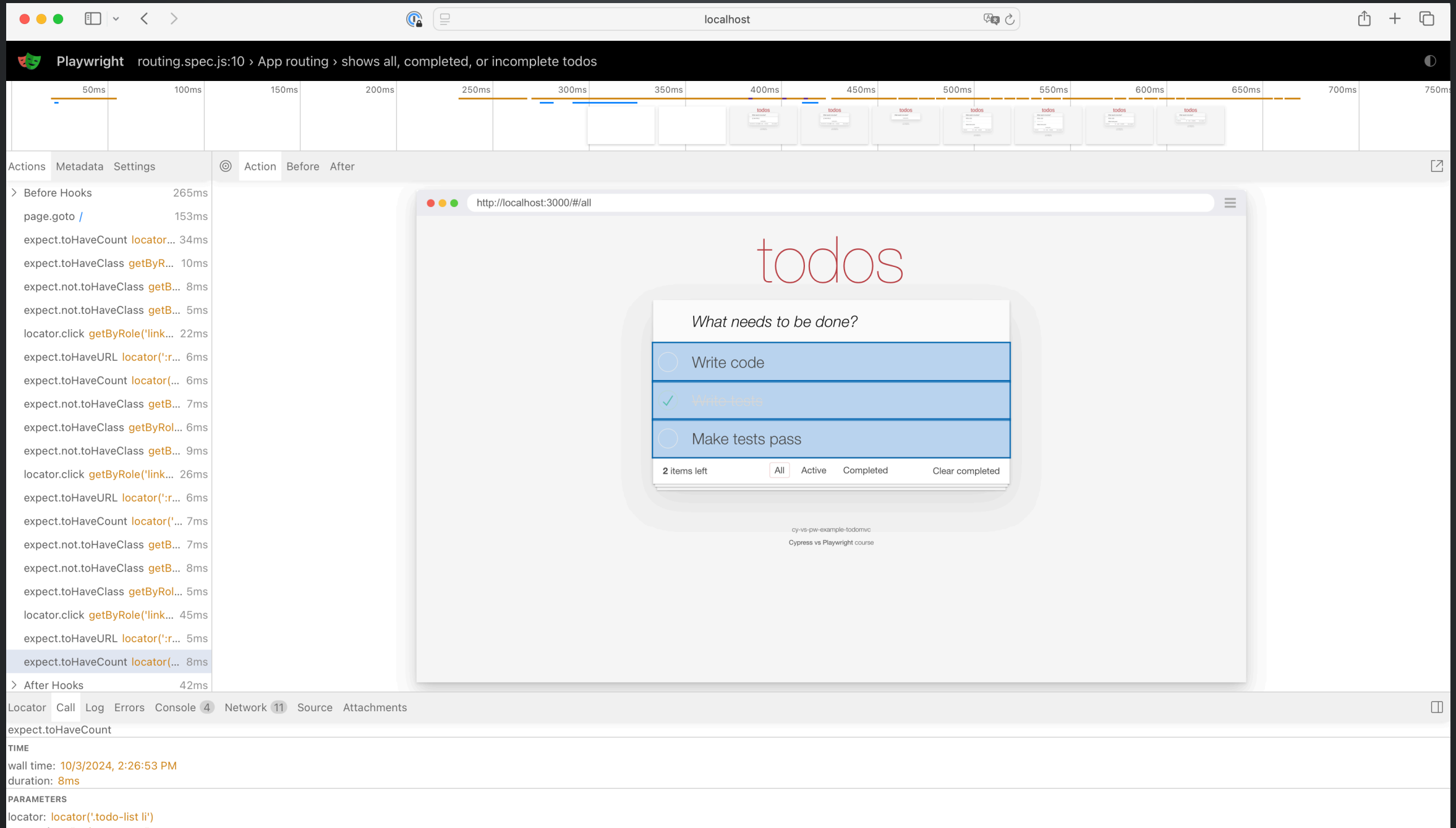
Tip: Enable traces on all CI runs

```
// playwright.config.js
/* Collect trace when retrying the failed test. See https://playwright.dev/docs/trace-
trace: Boolean(process.env.CI) ? 'on' : 'on-first-retry',
```


playwright tests
save artifacts #2
Re-run all jobs
Summary
Jobs
e2e
Run details
Usage
Workflow file
Triggered via push 3 minutes ago
bahmutov pushed 7ab2c37 main
Status
Success
Total duration
1m 24s
Artifacts
1
pw-e2e.yml
on: push
e2e 1m 15s
Artifacts
Produced during runtime
Name
Size
playwright-report
607 KB

VIEW PLAYWRIGHT TRACE

- download and unzip
- `npx playwright show-report <path to the unzipped folder>`


localhost
Playwright routing.spec.js:10 › App routing › shows all, completed, or incomplete todos
50ms 100ms 150ms 200ms 250ms 300ms 350ms 400ms 450ms 500ms 550ms 600ms 650ms 700ms 750ms
Actions Metadata Settings
Action Before After
Before Hooks 265ms
page.goto / 153ms
expect.toHaveCount locator... 34ms
expect.toHaveClass getByR... 10ms
expect.not.toHaveClass getB... 8ms
expect.not.toHaveClass getB... 5ms
locator.click getByRole('link... 22ms
expect.toHaveURL locator(':r... 6ms
expect.toHaveCount locator(... 6ms
expect.not.toHaveClass getB... 7ms
expect.toHaveClass getByRol... 6ms
expect.not.toHaveClass getB... 9ms
locator.click getByRole('link... 26ms
expect.toHaveURL locator(':r... 6ms
expect.toHaveCount locator('... 7ms
expect.not.toHaveClass getB... 7ms
expect.not.toHaveClass getB... 8ms
expect.toHaveClass getByRol... 5ms
locator.click getByRole('link... 45ms
expect.toHaveURL locator(':r... 5ms
expect.toHaveCount locator(... 8ms
After Hooks 42ms
http://localhost:3000/#/all
todos
What needs to be done?
Write code
Write tests
Make tests pass
2 items left
All
Active
Completed
Clear completed
cy-vs-pw-example-todomvc
Cypress vs Playwright course
Locator Call Log Errors Console 4 Network 11 Source Attachments
expect.toHaveCount
TIME
wall time: 10/3/2024, 2:26:53 PM
duration: 8ms
PARAMETERS
locator: locator('.todo-list li')

PARALLEL CI

- a way to get N times speedup
- Cypress: use Cypress Cloud or https://github.com/bahmutov/cypress-split
- Playwright: sharding across machines https://playwright.dev/docs/test-parallel#shard-tests-between-multiple-machines

🏁 CI

- use Docker image to tightly control the container
- use Cypress official GitHub Action
- store test artifacts (both successes and failures)
- run tests in parallel 🏎️

➡️ Go to the component testing chapter