TEST UI

 YOU WILL LEARN

- test the user interface using fixture data
- better separate test steps

- clean up the existing code
  - `git reset --hard`
  - `git clean -d -f`
- `git checkout b5`
- `npm install`

JSON FIXTURE

```
[
  {
    "title": "Write code",
    "completed": false,
    "id": "1"
  },
  {
    "title": "Write tests",
    "completed": true,
    "id": "2"
  },
  {
    "title": "Make tests pass",
    "completed": false,
    "id": "3"
  }
```

Fixture items in `fixtures/three.json`

PLAYWRIGHT

If we reset the backend data using the fixture items

```
const items = require('../fixtures/three.json')

test.beforeEach(async ({ request }) => {
  await request.post('/reset', { data: { todos: items } })
})
```

Then we can check if the user interface shows the data correctly

```
// pw/shows-items.spec.js
const items = require('../fixtures/three.json')

test('shows items', async ({ page }) => {
  // common locators
  const todos = page.locator('.todo-list li')
  const count = page.locator('[data-cy="remaining-count"]')

  await page.goto('/')

  // shows N items

  // go through the items and confirm each is rendered correctly
  // - label text
  // - completed or not
```

```
const items = require('../fixtures/three.json')

test('shows items', async ({ page }) => {
  // common locators
  const todos = page.locator('.todo-list li')
  const count = page.locator('[data-cy="remaining-count"]')

  await page.goto('/')

  // shows N items
  await expect(todos).toHaveCount(items.length)
  // go through the items and confirm each is rendered correctly
  // - label text
  // - completed or not
  for (const [k, item] of items.entries()) {
    const itemLocator = todos.nth(k)
```

Playwright solution. **Note**: `for (... of items.entries)` loop around the `await ...` test commands.

CYPRESS

Reset the data using the fixture array

```
// cypress/e2e/shows-items.cy.js

import items from '../../fixtures/three.json'

beforeEach(() => {
  // confirm there are several items
  // and some are completed and some are not

  cy.request('POST', '/reset', { todos: items })
})
```

Confirm the data is rendered correctly

```
import items from '../../fixtures/three.json'

it('completes a todo', () => {
  // common locators
  const todos = '.todo-list li'
  const count = '[data-cy="remaining-count"]'

  cy.visit('/')

  // shows N items

  // go through the items and confirm each is rendered correctly
  // - label text
  // - completed or not

  // confirm the remaining items count is correct
```

```
it('completes a todo', () => {
  // common locators
  const todos = '.todo-list li'
  const todoLabels = todos + ' label'
  const count = '[data-cy="remaining-count"]'

  cy.visit('/')

  // shows N items
  cy.get(todos).should('have.length', items.length)

  // go through the items and confirm each is rendered correctly
  // - label text
  // - completed or not
  items.forEach((item, k) => {
    cy.get(todos).eq(k).contains('label', item.title)
```

Note: Cypress queues up its commands before executing them, thus we can use simple `Array.forEach` method.

BETTER TEST STEPS UI

- `git checkout b6`

Let's improve how the test runner shows parts of a longer test

```
// shows the expected number of items
...
// check each item
...
// shows the remaining count
```

PLAYWRIGHT TEST STEPS

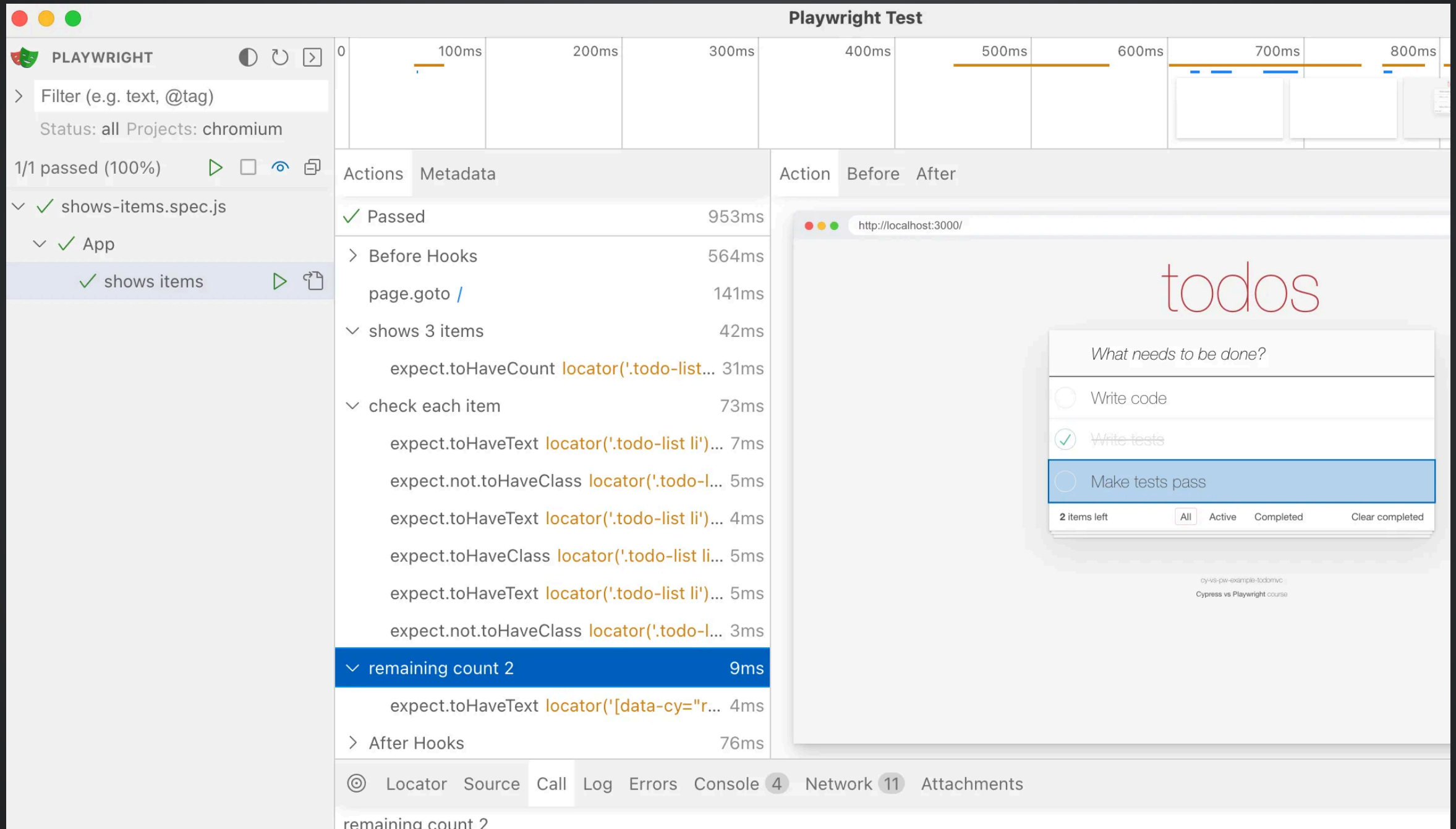
```
// pw/shows-items.spec.js

test('shows items', async ({ page }) => {
  // common locators
  const todos = page.locator('.todo-list li')
  const count = page.locator('[data-cy="remaining-count"]')

  await page.goto('/')

  // shows the expected number of items
  await expect(todos).toHaveCount(items.length)

  // check each item
  for (const [k, item] of items.entries()) {
    const itemLocator = todos.nth(k)
    await expect(itemLocator.locator('label')).toHaveText(item.title)
```

Replace test code comments with `test.step` commands, see https://playwright.dev/docs/api/class-test#test-step

```
test('shows items', async ({ page }) => {
  // common locators
  const todos = page.locator('.todo-list li')
  const count = page.locator('[data-cy="remaining-count"]')

  await page.goto('/')

  await test.step(`shows ${items.length} items`, async () => {
    await expect(todos).toHaveCount(items.length)
  })

  await test.step('check each item', async () => {
    for (const [k, item] of items.entries()) {
      const itemLocator = todos.nth(k)
      await expect(itemLocator.locator('label')).toHaveText(item.title)
      if (item.completed) {
```

PLAYWRIGHT NESTED STEPS

```
await test.step(`shows ${items.length} items`, async () => {
  await expect(todos).toHaveCount(items.length)
})
```


Playwright Test
PLAYWRIGHT
Filter (e.g. text, @tag)
Status: all Projects: chromium
1/1 passed (100%)
shows-items.spec.js
App
shows items
0
100ms
200ms
300ms
400ms
500ms
600ms
700ms
800ms
Actions
Metadata
Passed 953ms
Before Hooks 564ms
page.goto / 141ms
shows 3 items 42ms
expect.toHaveCount locator('.todo-list... 31ms
check each item 73ms
expect.toHaveText locator('.todo-list li')... 7ms
expect.not.toHaveClass locator('.todo-l... 5ms
expect.toHaveText locator('.todo-list li')... 4ms
expect.toHaveClass locator('.todo-list li... 5ms
expect.toHaveText locator('.todo-list li')... 5ms
expect.not.toHaveClass locator('.todo-l... 3ms
remaining count 2 9ms
expect.toHaveText locator('[data-cy="r... 4ms
After Hooks 76ms
Action
Before
After
http://localhost:3000/
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
Source
Call
Log
Errors
Console 4
Network 11
Attachments
remaining count 2

CYPRESS LOG COMMAND

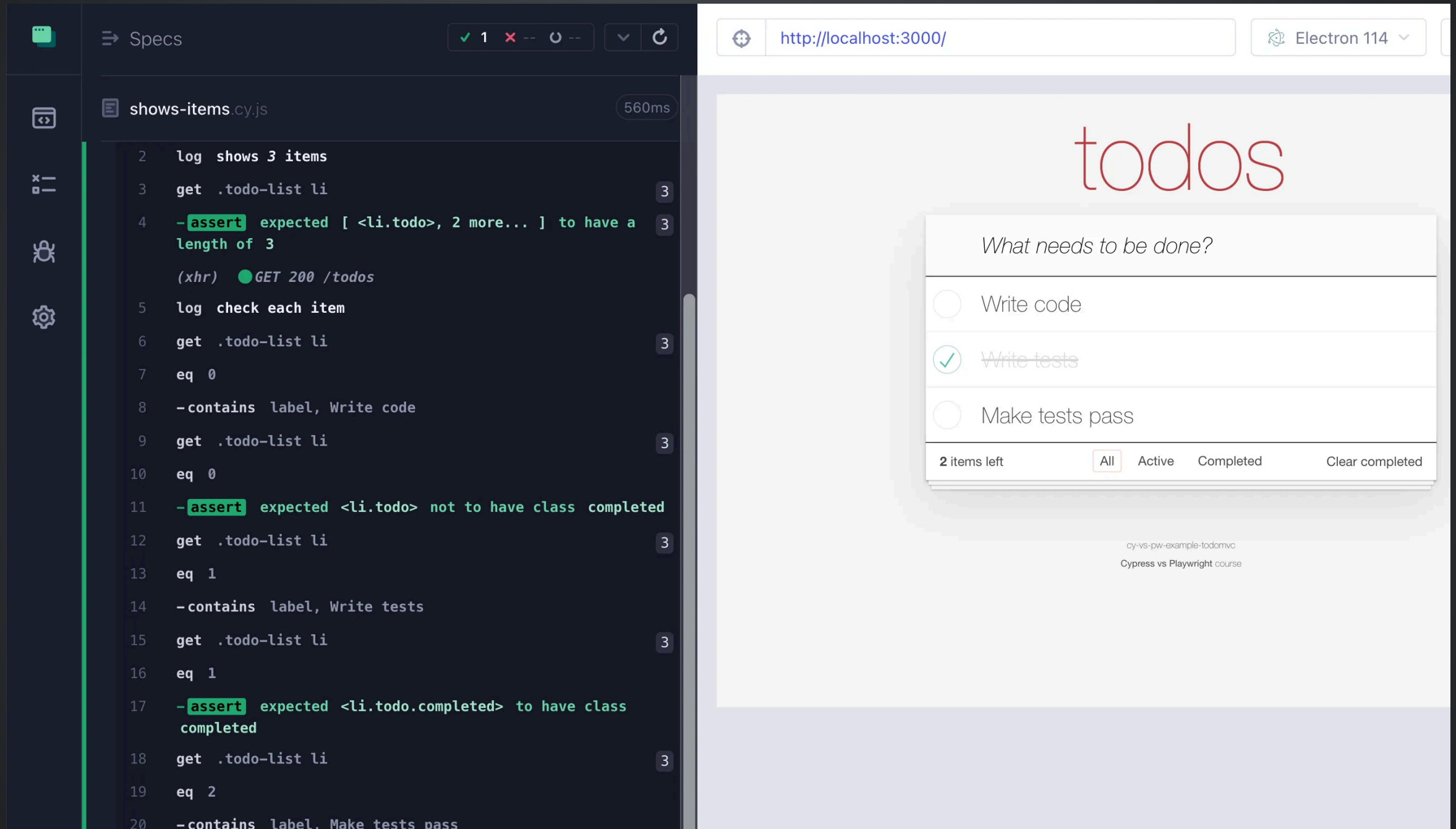
```
// cypress/e2e/shows-items.cy.js

it('completes a todo', () => {
  // common locators
  const todos = '.todo-list li'
  const count = '[data-cy="remaining-count"]'

  cy.visit('/')

  // shows the expected number of items
  cy.get(todos).should('have.length', items.length)

  // check each item
  items.forEach((item, k) => {
    cy.get(todos).eq(k).contains('label', item.title)
    if (item.completed) {
```

Replace test code comments with bold log messages using the `cy.log` command https://on.cypress.io/log

```
it('completes a todo', () => {
  // common locators
  const todos = '.todo-list li'
  const count = '[data-cy="remaining-count"]'

  cy.visit('/')

  cy.log(`**shows _${items.length}_ items**`)
  cy.get(todos).should('have.length', items.length)

  cy.log('**check each item**')
  items.forEach((item, k) => {
    cy.get(todos).eq(k).contains('label', item.title)
    if (item.completed) {
      cy.get(todos).eq(k).should('have.class', 'completed')
    } else {
```

You can use simple Markdown when using `cy.log`

```
cy.log(`**shows _${items.length}_ items**`)
cy.get(todos).should('have.length', items.length)
```


Specs
1
--
--
shows-items.cy.js
560ms
2 log shows 3 items
3 get .todo-list li 3
4 -assert expected [<li.todo>, 2 more...] to have a length of 3 3
(xhr) GET 200 /todos
5 log check each item
6 get .todo-list li 3
7 eq 0
8 -contains label, Write code
9 get .todo-list li 3
10 eq 0
11 -assert expected <li.todo> not to have class completed
12 get .todo-list li 3
13 eq 1
14 -contains label, Write tests
15 get .todo-list li 3
16 eq 1
17 -assert expected <li.todo.completed> to have class completed
18 get .todo-list li 3
19 eq 2
20 -contains label, Make tests pass
http://localhost:3000/
Electron 114
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

TESTING THE UI

- If you control the application data, checking how it is displayed is relatively easy
- Playwright has good wait to indent test steps
- Cypress can format its Command Log messages
  - it can also indent commands, but only inside custom commands

➡️ Pick the next section or jump to the 05-hover chapter