COMPLETING ITEMS

📚 YOU WILL LEARN

- Completing items test
- using JSON fixtures

- clean up the existing code
  - `git reset --hard`
  - `git clean -d -f`
- `git checkout b3`
- `npm install`

PLAYWRIGHT: COMPLETE AN ITEM TEST

```
// pw/complete.spec.js

const { test, expect } = require('@playwright/test')

test.describe('Complete todos', () => {
  test.beforeEach(async ({ request }) => {
    await request.post('/reset', { data: { todos: [] } })
  })

  test('completes a todo', async ({ page }) => {
    // common locators
    ...
    // enter three todos
    // "Write code", "Write tests", "Make tests pass"
    // confirm the item labels
    // confirm the "3" todos remaining is shown
// enter three todos
// "Write code", "Write tests", "Make tests pass"
await input.fill('Write code')
await input.press('Enter')
await input.fill('Write tests')
await input.press('Enter')
await input.fill('Make tests pass')
await input.press('Enter')

// confirm the item labels
await expect(todoLabels).toHaveText([
  'Write code',
  'Write tests',
  'Make tests pass'
])
// confirm the "3" todos remaining is shown
```

I really ❤️ Playwright's `toHaveText` assertion

```
await expect(todoLabels).toHaveText([
  'Write code',
  'Write tests',
  'Make tests pass'
])
```

In Cypress I would use cypress-map to do it

```
cy.get(todoLabels).should('read', [
  'Write code',
  'Write tests',
  'Make tests pass'
])
```

CYPRESS: COMPLETE AN ITEM TEST

```
cy.visit('/')
cy.get('body.loaded').should('be.visible')

// enter three todos
// "Write code", "Write tests", "Make tests pass"
// confirm the item labels
// confirm the "3" todos remaining is shown
// complete the first item by clicking its toggle element
// the first item should have class "completed"
// confirm the 2nd and the 3rd items do not have class "completed"
// Bonus: can you confirm classes for all 3 elements at once
// confirm there are 2 remaining items
// and that we can clear the completed items button appears
// enter three todos
// "Write code", "Write tests", "Make tests pass"
cy.get(input)
  .type('Write code{enter}')
  .type('Write tests{enter}')
  .type('Make tests pass{enter}')

// confirm the item labels
cy.get(todoLabels)
  .should('have.length', 3)
  .then(($el) => Cypress._.map($el, 'innerText'))
  .should('deep.equal', ['Write code', 'Write tests', 'Make tests pass'])
// confirm the "3" todos remaining is shown
cy.contains(count, 3)

// complete the first item by clicking its toggle element
```

USE JSON FIXTURES

- `git checkout b4`

Adds `fixtures/three.json` with 3 todo items

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

USE JSON FIXTURE IN PLAYWRIGHT

Update the `pw/complete.spec.js` file to reset the initial data using `fixtures/three.json` and remove the hard-coded values

```
// complete.spec.js
test.describe('Complete todos', () => {
  test.beforeEach(async ({ request }) => {
    request.post('/reset', { data: { todos: [] } })
  })

  test('completes a todo', async ({ page }) => {
    // common locators
    const input = page.getByPlaceholder('What needs to be done?')
    const todos = page.locator('.todo-list li')
    const todoLabels = todos.locator('label')
    const count = page.locator('[data-cy="remaining-count"]')

    await page.goto('/')
    await page.locator('body.loaded').waitFor()
```

PLAYWRIGHT SCRIPT RUNS IN NODE

```
const { test, expect } = require('@playwright/test')
const items = require('../fixtures/three.json')

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

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

    await page.goto('/')
```

JSON FIXTURES IN CYPRESS

Update the `cypress/e2e/complete.cy.js` spec to use the `fixtures/three.json` data

💡 You can directly require or import JSON files or use the cy.fixture command

```js
// cypress/e2e/complete.cy.js

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

describe('Complete todos', () => {
  beforeEach(() => {
    // immediately create 3 todos
    // using the JSON file "fixtures/three.json"
    cy.request('POST', '/reset', { todos: items })
  })

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

Cypress JSON fixture solution

🏁 LESSONS LEARNED

- use JSON fixtures to quickly set the initial state
- avoids UI test duplication

➡️ Pick the next section or jump to the 04-test-ui chapter