THE VERY BASIC TESTS

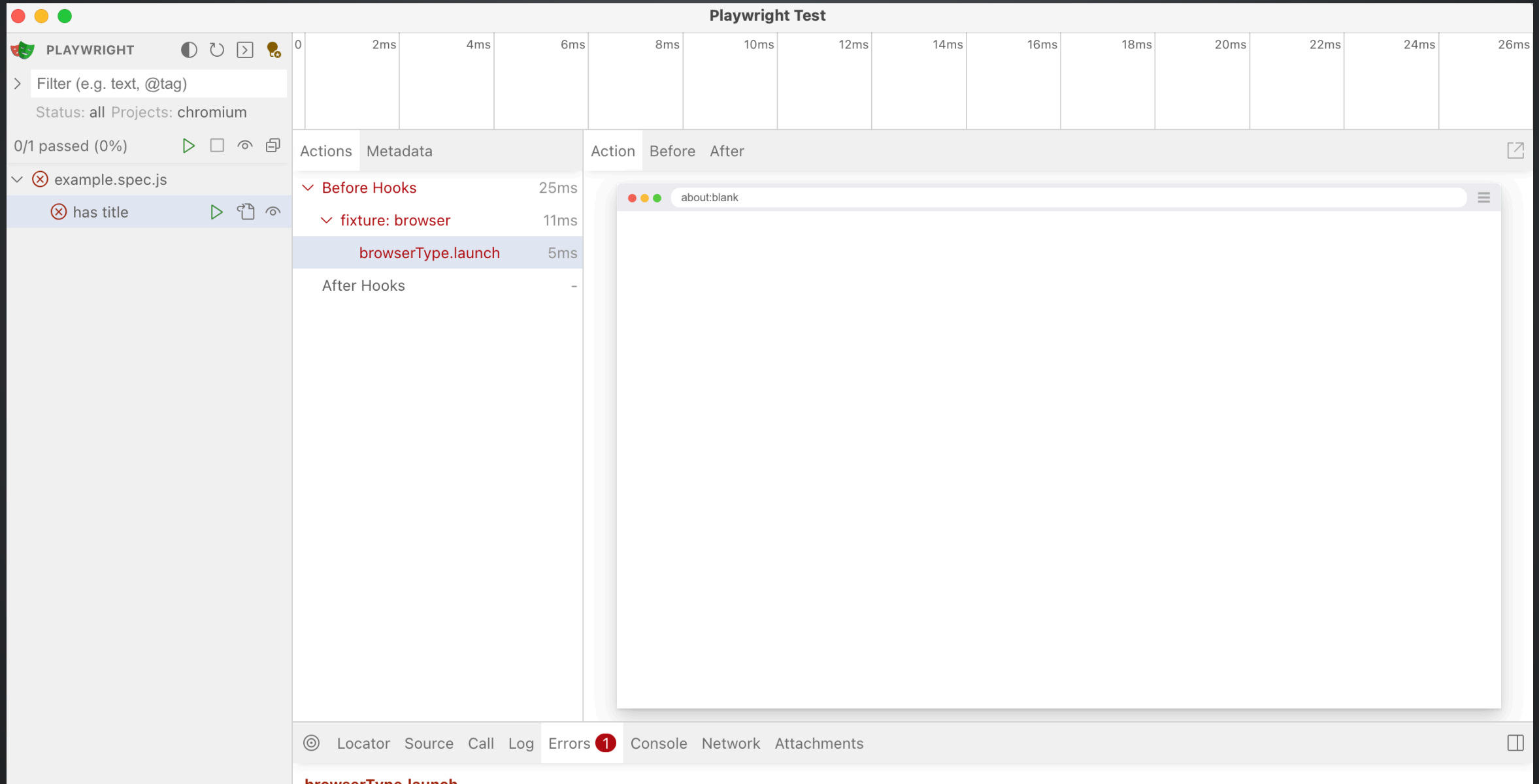
 YOU WILL LEARN

- start the application automatically
- interactive mode
- making HTTP requests
- the base URL

CLEANUP

- clean up the existing code
  - `git reset --hard`
  - `git clean -d -f`
- `git checkout a4`
- `npm install`

Tip: If necessary, run `npx playwright install` to install browsers



package.json

```
{
  "scripts": {
    "start": "json-server ...",
    "reset": "node reset-db.js",
    "test:pw": "playwright test",
    "test:cy": "cypress run"
  },
  "dependencies": {
    "json-server": "0.17.4",
    "json-server-reset": "1.6.0"
  },
  "devDependencies": {
    "cypress": "^13.4.0",
    "@playwright/test": "^1.39.0"
  }
}
```

TODO: START THE SERVER FROM PLAYWRIGHT

- modify the `playwright.config.js` to start the server on port 3000
- verify the application starts when you run Pw tests

💡 Find docs for Playwright Web Server

SOLUTION

```
const { defineConfig, devices } = require('@playwright/test')

module.exports = defineConfig({
  ...,
  /* Run your local dev server before starting the tests */
  webServer: {
    command: 'npm run start',
    url: 'http://127.0.0.1:3000',
    reuseExistingServer: !process.env.CI,
    // if you want to see the output from the started web server
    // stdout: 'pipe',
  },
})
```

TODO: START THE SERVER FOR CYPRESS

Use bahmutov/start-server-and-test plugin

- `$ npm i -D start-server-and-test`

```
{
  "scripts": {
    "start": "json-server ...",
    "test:pw": "playwright test",
    "test:cy": "cypress run",
    "e2e": "start-test start 3000 test:cy"
  }
}
```

and execute `npm run e2e`

INTERACTIVE MODE

While working locally, we want to re-run the tests when we save the spec file

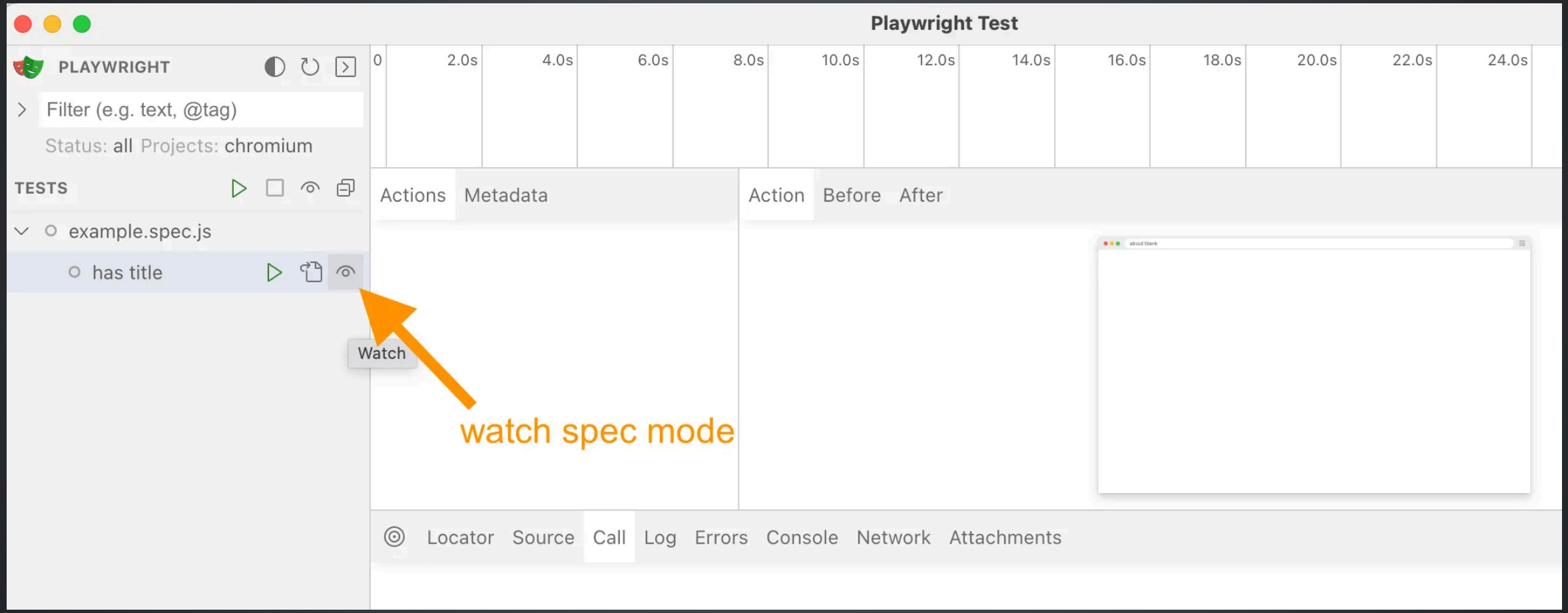
- `cypress open` = `playwright test --ui`
- `cypress run` = `playwright test`

 Playwright UI mode docs and cypress open

WORKING WITH PLAYWRIGHT TEST

- reset the code
- `git checkout a5`
- launch Pw interactive mode
- watch the spec file `example.spec.js`

WATCHING THE SPEC FILE



FINISH THE TEST

```
// pw/example.spec.js

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

test('has title', async ({ page }) => {
  await page.goto('http://localhost:3000/')

  // Expect a title "to contain" a substring.
  await expect(page).toHaveTitle('cy-vs-pw-example-todomvc')

  // confirm there are 3 todo items on the page
  // use the CSS selector ".todo-list li"
  // https://playwright.dev/docs/locators
  // and the count assertion
  // https://playwright.dev/docs/api/class-locatorassertions
})
```

💡 Read Page locators and Playwright assertions

Playwright Test

PLAYWRIGHT

Filter (e.g. text, @tag)

Status: all Projects: chromium

1/1 passed (100%)

example.spec.js

has title

0 50ms 100ms 150ms 200ms 250ms 300ms 350ms 400ms 450ms 500ms 550ms 600ms

Actions Metadata

Before Hooks 472ms

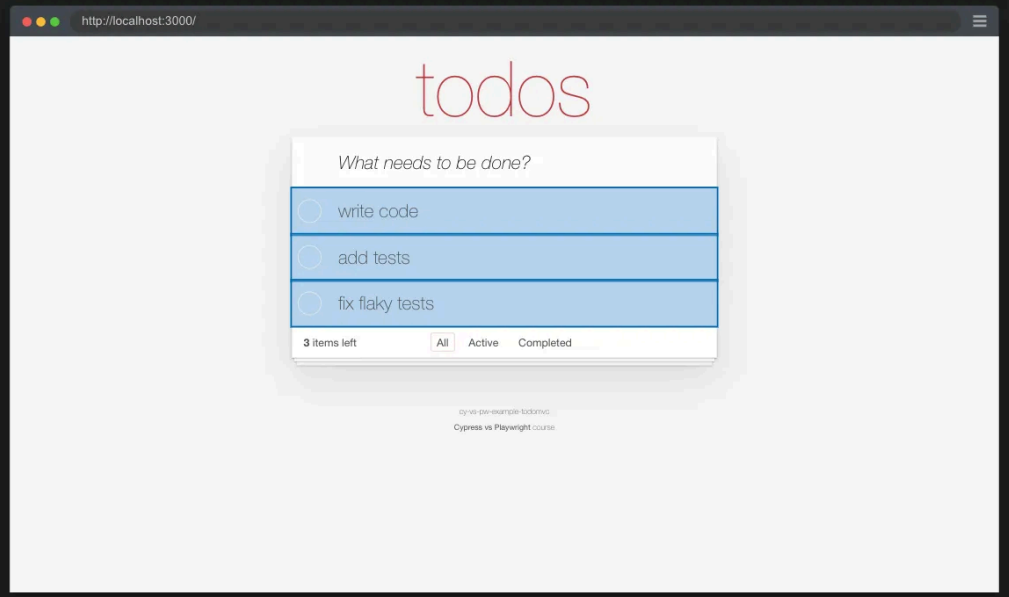
page.goto http://localhost:... 145ms

expect.toHaveTitle locator('... 30ms

expect.toHaveCount locator('... 5ms

After Hooks 4ms

Action Before After



Locator Source Call Log Errors Console 4 Network 10 Attachments

expect.toHaveCount

TIME

wall time: 11/8/2023, 10:58:07 AM

duration: 5ms

PARAMETERS

locator: locator('.todo-list li')

expression: "to.have.count"

expectedNumber: 3

expectedValue: undefined

PW SOLUTION

```
// confirm there are 3 todo items on the page
// use the CSS selector ".todo-list li"
// https://playwright.dev/docs/locators
// and the count assertion
// https://playwright.dev/docs/api/class-locatorassertions
await expect(page.locator('.todo-list li')).toHaveCount(3, {
  timeout: 1000
})
```

TODO: CYPRESS TEST

```
// cypress/e2e/spec.cy.js
cy.visit('http://localhost:3000/')

// the page title should have text "cy-vs-pw-example-todomvc"
// https://on.cypress.io/title
cy.title().should('equal', 'cy-vs-pw-example-todomvc')

// confirm there are 3 todo items on the page
// use the CSS selector ".todo-list li"
// https://on.cypress.io/get
// and "should have length" assertion
// https://on.cypress.io/should
```

```
// confirm there are 3 todo items on the page
// use the CSS selector ".todo-list li"
// https://on.cypress.io/get
// and "should have length" assertion
// https://on.cypress.io/should
cy.get('.todo-list li').should('have.length', 3)
```

Question: how do you change the command's timeout?

Specs

✓ 1 ✗ -- ○ --

spec.cy.js 992ms

✓ has title

TEST BODY

```
1  visit  http://localhost:3000/
2  title
3  -assert  expected  cy-vs-pw-example-todomvc  to equal  **cy-vs-pw-example-todomvc**
4  get  .todo-list li                                                    3
5  -assert  expected  [ <li.todo>, 2 more... ]  to have a length of  3   3
   (xhr)  GET 200 /todos
```

http://localhost:3000/

Electron 114

1000x660 (35%)

Question: what do you see at the end of the test in Playwright? In Cypress?

 Read https://glebbahmutov.com/blog/cy-vs-pw-browser/ for more.

MAKE HTTP REQUEST

- clean up the existing code
  - `git reset --hard`
  - `git clean -d -f`
- `git checkout a6`
- `npm install`

Our app loads the data from `data.json` file. We want to clean up the data before each test. We can do it by making a HTTP call:

```
# using HTTPie client https://httpie.io/
$ http :3000/reset todos:=[]
HTTP/1.1 200 OK

# using curl
$ curl --header "Content-Type: application/json" \
  -d '{"todos":[]}' \
  http://localhost:3000/reset
```

The file `data.json` should have an empty "todo" list.

APP LOADS

```
// app.js

SET_LOADING(state, flag) {
  state.loading = flag
  if (flag === false) {
    // an easy way for the application to signal
    // that it is done loading
    document.body.classList.add('loaded')
  }
}
```

TODO: PLAYWRIGHT SPEC

```
// pw/example.spec.js

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

// before each test clear all todos
// by making a POST request to "/reset" endpoint
// and pass an object { todos: [] }
// which should clear all existing todos
// see:
// https://playwright.dev/docs/api/class-test
// https://playwright.dev/docs/api-testing

test('has title', async ({ page }) => {
  await page.goto('http://localhost:3000/')

  // confirm the page has finished loading todos
```

💡 Read https://playwright.dev/docs/api-testing

```
// pw/example.spec.js

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

test.beforeEach(async ({ request }) => {
  await request.post('http://localhost:3000/reset', { data: { todos: [] } })
})

test('has title', async ({ page }) => {
  await page.goto('http://localhost:3000/')
  await expect(page.locator('body')).toHaveClass('loaded')
  // ^^^ alternative:
  await page.locator('body.loaded').waitFor()
  await expect(page.locator('.todo-list li')).toHaveCount(0)
})
```

One possible Playwright solution.

TIP: `waitFor` METHOD

```
await <locator>.waitFor({
  // defaults to 'visible'
  state: 'attached' | 'detached' | 'visible' | 'hidden',
  timeout: 0 // ms
})
```

Documentation https://playwright.dev/docs/api/class-locator#locator-wait-for

PW .WAITFOR VS CYPRESS ASSERTIONS

```
await <locator>.waitFor()
// same as
cy.get(<locator>).should('be.visible')

await <locator>.waitFor({ state: 'attached' })
// same as
cy.get(<locator>)
```

TODO: CYPRESS SPEC

```
// cypress/e2e/spec.cy.js

// before each test clear all todos
// by making a POST request to "/reset" endpoint
// and pass an object { todos: [] }
// which should clear all existing todos
// see:
// https://on.cypress.io/writing-and-organizing-tests#Hooks
// https://on.cypress.io/request

it('has title', () => {
  // visit the page "localhost:3000"
  // https://on.cypress.io/visit
  cy.visit('http://localhost:3000/')

  // confirm the page has finished loading todos
```

```
beforeEach(() => {
  cy.request('POST', 'http://localhost:3000/reset', { todos: [] })
})

it('has title', () => {
  cy.visit('http://localhost:3000/')
  cy.get('body').should('have.class', 'loaded')
  // ^^^ alternative:
  cy.get('body.loaded')
  cy.get('.todo-list li').should('have.length', 0)
})
```

SET THE BASE URL

- clean up the existing code
  - `git reset --hard`
  - `git clean -d -f`
- `git checkout a6`
- `npm install`

PLAYWRIGHT BASE URL

Modify the `pw/example.spec.js` and `playwright.config.js` to remove the hard-coded `localhost:3000` from the spec

```
// playwright.config.js

const { defineConfig, devices } = require('@playwright/test')
module.exports = defineConfig({
  ...
  use: {
    /* Base URL to use in actions like `await page.goto('/')`. */
    baseURL: 'http://localhost:3000',
  },
})
```

CYPRESS BASE URL

Modify the `cypress/e2e/spec.cy.js` and `cypress.config.js` to remove the hard-coded `localhost:3000` from the spec

```
// cypress.config.js

const { defineConfig } = require('cypress')

module.exports = defineConfig({
  e2e: {
    baseUrl: 'http://localhost:3000'
  }
})
```

🏁 CONCLUSIONS

- PW and CY have interactive mode
- you can make API requests to prepare the data before the test
- do not hard-code the base URL

➡️ Pick the next section or jump to the 02-adding-items chapter