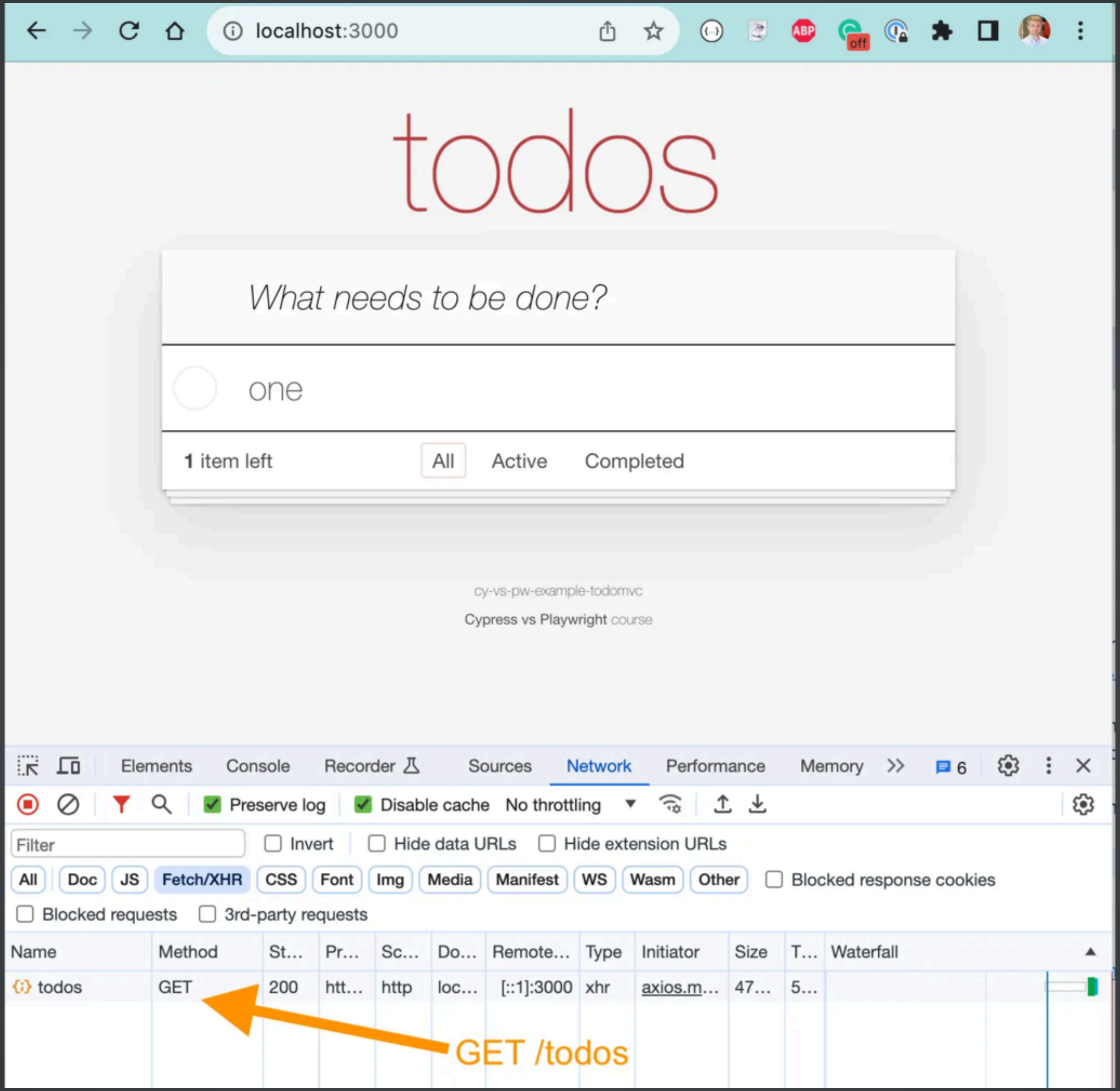
CONTROL NETWORK CALLS

📚 YOU WILL LEARN

- how to spy on / stub network calls
- how to wait for the network calls from tests
- how to use network calls in assertions
- how to delay a network call



Name	Method	St…	Pr…	Sc…	Do…	Remote…	Type	Initiator	Size	T…	Waterfall
todos	GET	200	htt…	http	loc…	[::1]:3000	xhr	axios.m…	47…	5…	

- clean up the existing code
  - `git reset --hard`
  - `git clean -d -f`
- `git checkout d1`
- `npm install`

STUB NETWORK CALLS

Let's mock the GET `/todos` network call and respond with the JSON fixture data.

IMPORTANT ⚠️

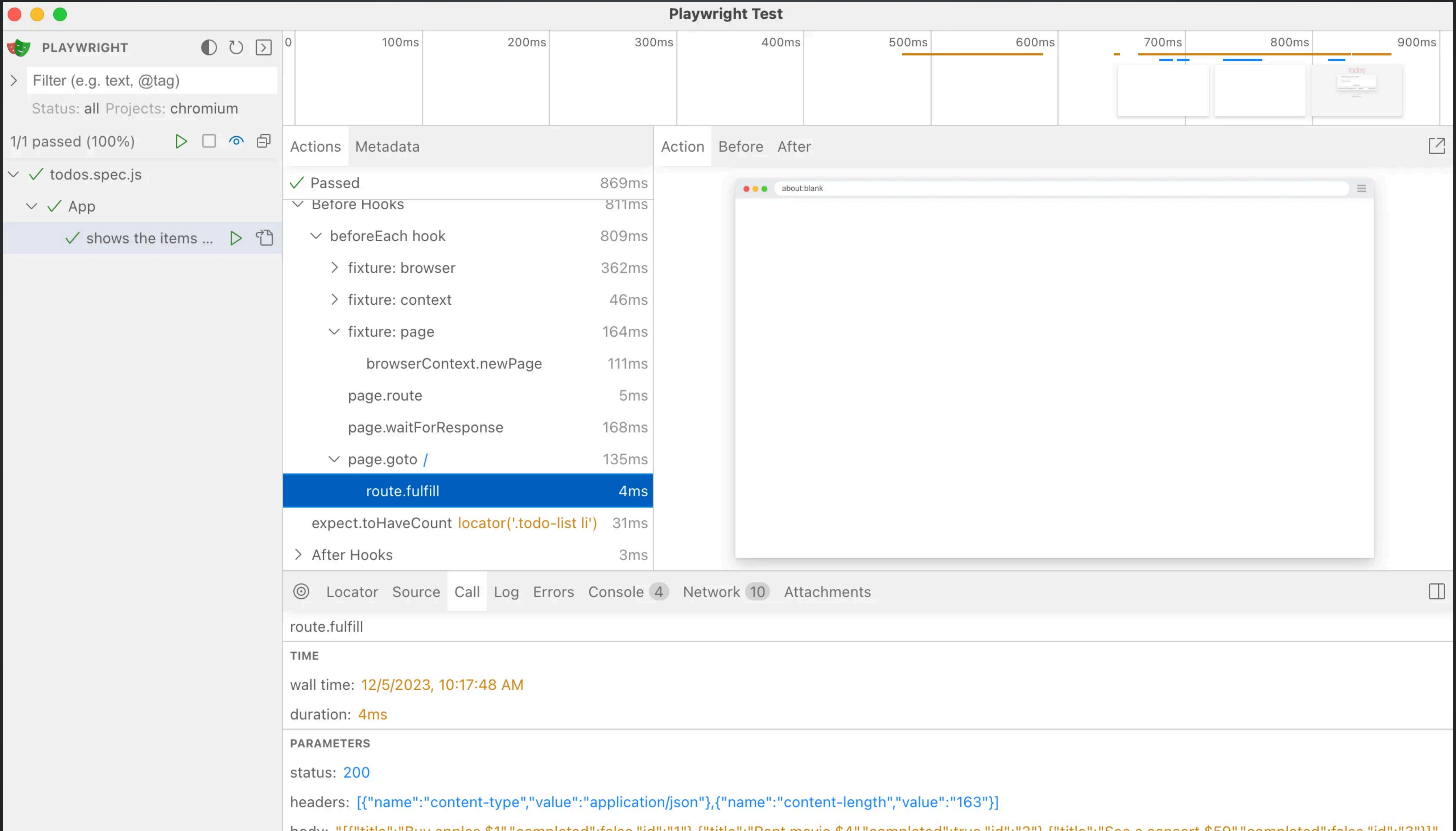
Always set up network spies / stubs **before** the action that makes the app make that network call.

PLAYWRIGHT STUB THE NETWORK CALL

```
// pw/todos.spec.js

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

test.describe('App', () => {
  test.beforeEach(async ({ page }) => {
    // set up a route handler for "/todos" endpoint
    // when the route matches, fulfill it using
    // the loaded items array
    // Tip: make sure to set the content type header
    //
    // set up a promise that waits for the response
    // to the network call "/todos"
    // https://playwright.dev/docs/network
    await page.goto('/')
```

Stub the GET `/todos` network call in Playwright

```js
// pw/todos.spec.js

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

test.describe('App', () => {
  let loadSpy

  test.beforeEach(async ({ page }) => {
    // set up a route handler for "/todos" endpoint
    // when the route matches, fulfill it using
    // the loaded items array
    // Tip: make sure to set the content type header
    await page.route('/todos', (route) =>
      route.fulfill({
        headers: { 'Content-Type': 'application/json' },
```



Playwright Test

PLAYWRIGHT

Filter (e.g. text, @tag)

Status: all Projects: chromium

1/1 passed (100%)

- ✓ todos.spec.js
  - ✓ App
    - ✓ shows the items ...

Actions | Metadata

✓ Passed	869ms
Before Hooks	811ms
beforeEach hook	809ms
fixture: browser	362ms
fixture: context	46ms
fixture: page	164ms
browserContext.newPage	111ms
page.route	5ms
page.waitForResponse	168ms
page.goto /	135ms
route.fulfill	4ms
expect.toHaveCount locator('.todo-list li')	31ms
After Hooks	3ms

Action | Before | After

about:blank

Locator | Source | Call | Log | Errors | Console 4 | Network 10 | Attachments

route.fulfill

TIME

wall time: 12/5/2023, 10:17:48 AM

duration: 4ms

PARAMETERS

status: 200

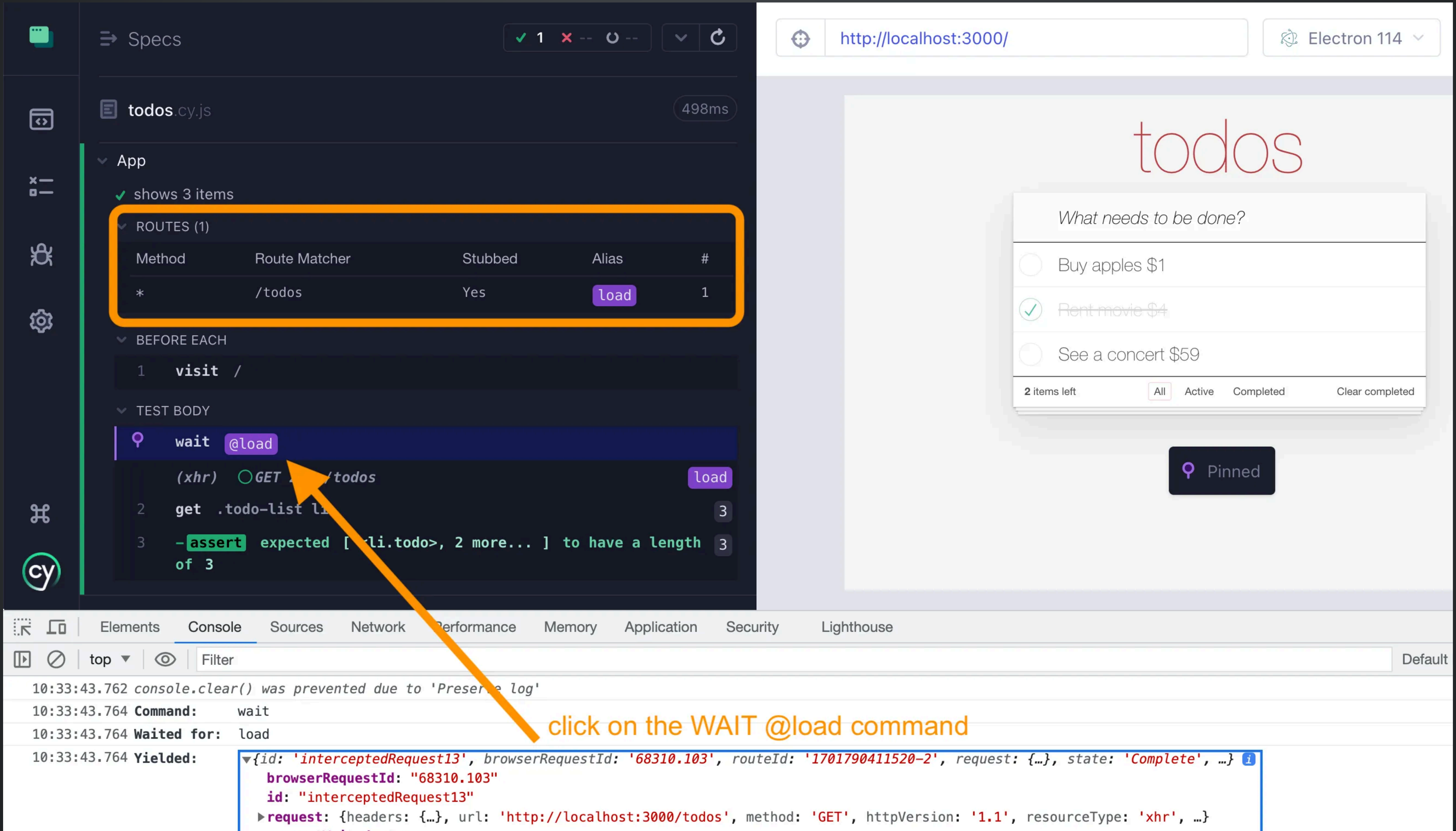
headers: `[{"name":"content-type","value":"application/json"},{"name":"content-length","value":"163"}]`

body: `"[{"title":"Buy apples $1","completed":false,"id":"1"},{"title":"Rent movie $4","completed":true,"id":"2"},{"title":"See a concert $50","completed":false,"id":"3"}]"`

CYPRESS NETWORK STUB

```
// cypress/e2e/todos.cy.js

describe('App', () => {
  beforeEach(() => {
    // stub the "GET /todos" network call the application makes
    // and return the data from the fixture file "products.json"
    // give this network stub an alias "load"
    // https://on.cypress.io/intercept
    // https://on.cypress.io/as
    cy.visit('/')
  })

  it('shows 3 items', () => {
    const todos = '.todo-list li'
    // wait for the intercepted network call "load"
// cypress/e2e/todos.cy.js

describe('App', () => {
  beforeEach(() => {
    // stub the "GET /todos" network call the application makes
    // and return the data from the fixture file "products.json"
    // give this network stub an alias "load"
    // https://on.cypress.io/intercept
    // https://on.cypress.io/as
    cy.intercept('/todos', { fixture: 'products.json' }).as('load')
    cy.visit('/')
  })

  it('shows 3 items', () => {
    const todos = '.todo-list li'
    // wait for the intercepted network call "load"
```

Cypress network stub solution



```
App
✓ shows 3 items
ROUTES (1)
Method   Route Matcher   Stubbed   Alias   #
*        /todos          Yes       load    1
BEFORE EACH
1   visit  /
TEST BODY
    wait  @load
    (xhr)  GET  /todos                                   load
2   get  .todo-list li                                   3
3   -assert  expected [ <li.todo>, 2 more... ] to have a length  3
    of 3
```

click on the WAIT @load command

```
10:33:43.762 console.clear() was prevented due to 'Preserve log'
10:33:43.764 Command:    wait
10:33:43.764 Waited for: load
10:33:43.764 Yielded:    ▼{id: 'interceptedRequest13', browserRequestId: '68310.103', routeId: '1701790411520-2', request: {…}, state: 'Complete', …}
                           browserRequestId: "68310.103"
                           id: "interceptedRequest13"
                          ▶request: {headers: {…}, url: 'http://localhost:3000/todos', method: 'GET', httpVersion: '1.1', resourceType: 'xhr', …}
```

SPY ON NETWORK CALL

Instead of stopping the network call, let it travel to the server. Use the response from the test.

- `get checkout d2`

In the next lessons:

- **stub** the initial `GET /todos`
- **spy** on `POST /todos` calls

PLAYWRIGHT NETWORK SPY

```js
// pw/todos.spec.js
const { test, expect } = require('@playwright/test')

test.describe('App', () => {
  test.beforeEach(async ({ page }) => {
    // intercept the route "/todos"
    // - "GET /todos" respond with an empty list
    // - otherwise let the request continue
    await page.goto('/')
  })

  test('shows the items with css class', async ({ page }) => {
    // confirm the application has finished loading
    // by checking the presence of an element with class "loaded"
    // there should be no todos
    // spy on the "POST /todos" call
```

See https://playwright.dev/docs/network

```
// pw/todos.spec.js

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

test.describe('App', () => {
  test.beforeEach(async ({ page }) => {
    // intercept the route "/todos"
    // - "GET /todos" respond with an empty list
    // - otherwise let the request continue
    await page.route('/todos', (route) => {
      if (route.request().method() === 'GET') {
        return route.fulfill({
          headers: { 'Content-Type': 'application/json' },
          body: JSON.stringify([])
        })
      } else {
```

CYPRESS NETWORK SPY

```
// cypress/e2e/todos.cy.js

describe('App', () => {
  beforeEach(() => {
    // stub the "GET /todos" network calls
    // and return an empty array
    cy.visit('/')
  })

  it('sends new todo object', () => {
    const todos = '.todo-list li'
    // confirm the application has finished loading
    // by checking the presence of an element with class "loaded"
    // and there are no items

    // spy on the "POST /todos" call
```

See https://on.cypress.io/network-requests

```js
// cypress/e2e/todos.cy.js

describe('App', () => {
  beforeEach(() => {
    // stub the "GET /todos" network calls
    // and return an empty array
    cy.intercept('GET', '/todos', { body: [] })
    cy.visit('/')
  })

  it('sends new todo object', () => {
    const todos = '.todo-list li'
    // confirm the application has finished loading
    cy.get('.loaded')
    // and there are no items
    cy.get(todos).should('have.length', 0)
```

Cypress network spy solution

USE NETWORK DATA

- `git checkout d3`

Let's spy on the network call and confirm the app shows the data correctly

PLAYWRIGHT TEST

```
// pw/todos.spec.js

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

test.describe('App', () => {
  test.beforeEach(async ({ page }) => {
    // spy on the network calls to "/todos" endpoint
    await page.goto('/')
  })

  test('shows the same number of items as sent by the server', async ({
    page
  }) => {
    // confirm the network call has happened
    // and get the response as json
    // confirm the page shows the same number of todo items
```

```js
// pw/todos.spec.js

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

test.describe('App', () => {
  let load

  test.beforeEach(async ({ page }) => {
    // spy on the network calls to "/todos" endpoint
    load = page.waitForRequest('/todos')
    await page.goto('/')
  })

  test('shows the same number of items as sent by the server', async ({
    page
  }) => {
```

CYPRESS TEST

```
// cypress/e2e/todos.cy.js

describe('App', () => {
  beforeEach(() => {
    // spy on the network call "GET /todos"
    // give the network intercept an alias
    cy.visit('/')
  })

  it('shows the same number of items as sent by the server', () => {
    // wait for the network alias
    // get its response body and confirm
    // it is an array
    // grab its length and pass it to the cy.then callback
    // inside the callback get the number of Todo items
    // on the page, it should equal to the number of items
```

```js
// cypress/e2e/todos.cy.js

describe('App', () => {
  beforeEach(() => {
    // spy on the network call "GET /todos"
    // give the network intercept an alias
    cy.intercept('GET', '/todos').as('load')
    cy.visit('/')
  })

  it('shows the same number of items as sent by the server', () => {
    // wait for the network alias
    // get its response body and confirm
    // it is an array
    // grab its length and pass it to the cy.then callback
    // inside the callback get the number of Todo items
```

Tip: in Cypress you always pass data that you get from the app using `cy.then(callback)`

DATA CACHING

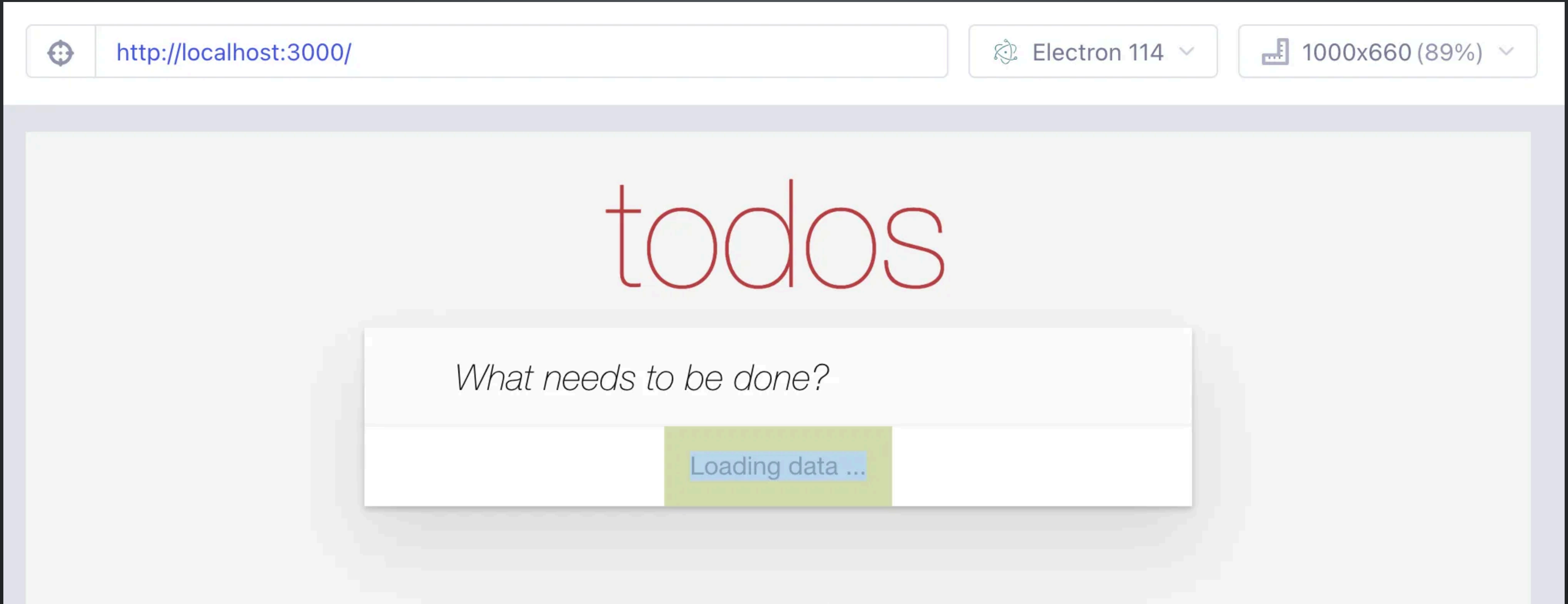
Important: our application and browser cache network data using `ETag` and `If-None-Match` headers. Playwright disables caching automatically. In Cypress you have to control it yourself, see https://glebbahmutov.com/blog/cypress-intercept-problems/

DISABLE NETWORK CACHING IN CYPRESS

```
beforeEach(() => {
  // disable network caching using a Chrome Debugger Protocol command
  // by using "cy.wrap" command we ensure that the promise returned
  // by the Cypress.automation method resolves before proceeding
  // to the next Cypress command
  cy.wrap(
    Cypress.automation('remote:debugger:protocol', {
      command: 'Network.setCacheDisabled',
      params: {
        cacheDisabled: true
      }
    })
  )
  // spy on the network call "GET /todos"
  // give the network intercept an alias
  cy.intercept('GET', '/todos').as('load')
```

SLOW DOWN NETWORK REQUEST

Let's spy on the `GET /todos` network request and slow it down to test the loading element.



SLOW DOWN NETWORK CALL IN PLAYWRIGHT

- `git checkout d5`

```js
// pw/loader.spec.js

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

test.describe('App', () => {
  test('shows a loader', async ({ page }) => {
    // intercept the "/todos" call
    // and delay it by 2 seconds before
    // allowing it to continue to the server

    // spy on the "/todos" network call
    // visit the page after setting up the network spies
    await page.goto('/')
    // confirm the loading element is visible
    // confirm the loading element is hidden
    // confirm the "/todos" call has happened
```

Tip: read the https://playwright.dev/docs/api/class-route documentation.

```
// pw/loader.spec.js

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

test.describe('App', () => {
  test('shows a loader', async ({ page }) => {
    // intercept the "/todos" call
    // and delay it by 2 seconds before
    // allowing it to continue to the server
    await page.route('/todos', (route) => {
      setTimeout(() => {
        route.continue()
      }, 2000)
    })
    // spy on the "/todos" network call
    const loading = page.waitForResponse('/todos')
```

Playwright solution

DELAY NETWORK CALL IN CYPRESS

```
// cypress/e2e/loader.cy.js

describe('App', () => {
  it('shows a loader', () => {
    // intercept the "GET /todos" network call
    // let the call continue to the server
    // but delay it by 2 seconds
    // This should give the loading element plenty of time
    cy.visit('/')
    // confirm the loading element is visible
    // and then becomes hidden

    // confirm the app finishes loading really quickly
    // after the loader becomes hidden
  })
})
```

Hint: read https://on.cypress.io/intercept documentation

```js
// cypress/e2e/loader.cy.js

describe('App', () => {
  it('shows a loader', () => {
    // intercept the "GET /todos" network call
    // let the call continue to the server
    // but delay it by 2 seconds
    // This should give the loading element plenty of time
    cy.intercept('GET', '/todos', () => Cypress.Promise.delay(2000))
    cy.visit('/')
    // confirm the loading element is visible
    // and then becomes hidden
    cy.get('.loading').should('be.visible')
    cy.get('.loading').should('not.be.visible')
    // confirm the app finishes loading really quickly
    // after the loader becomes hidden
```

Cypress network request delay example

🏁 SPY AND STUB THE NETWORK FROM YOUR TESTS

- stub network calls to control the data
- spy on network calls during tests
- network caching might affect the testing
- 🎓 Cypress Network Testing Exercises course

➡️ Pick the next section or jump to the 07-clock chapter

WRITE E2E TESTS IDEAS

using Cypress and / or Playwright

```
Adding todos
   - add 1 todo
   - load all todos from a JSON fixture and add
   - add 1 todo without resetting the DB
   - clearing the input field
   - trims the input text
Editing todos
  - edit an existing item text
Deleting todos
  - delete 1 todo
  - delete multiple todos
Completing todos
  - check one todo
  - clear completed todos
Application routing
  - item filters
```