

ADDING ITEMS TESTS



YOU WILL LEARN

- failing tests on app error
- testing adding items

- clean up the existing code
 - git reset --hard
 - git clean -d -f
- git checkout a8
- npm install

APP CRASHES ON PURPOSE

```
// app.js

throw new Error('App crash')

// Once the above error causes your test to fail
// can you verify if the test runner catches errors
// throws asynchronously or rejected promises?

// setTimeout(() => {
//   throw new Error('Async app crash')
// }, 10)
//
// Promise.reject(new Error('rejected promise'))
```

FAIL CYPRESS TEST ON APP ERROR

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

// commands operate with respect to the "baseUrl"
// defined in the cypress.config.js file

beforEach(() => {
  cy.request('POST', '/reset', { todos: [] })
})

it('has title', () => {
  console.log('running test "%s"', Cypress.currentTest.titlePath.join('/'))
  cy.visit('/')
  cy.get('body.loaded')
  cy.get('.todo-list li').should('have.length', 0)
})
```

Can you fail the test on all 3 types of app errors?

The screenshot shows the Cypress Test Runner interface on the left and a browser window on the right.

Cypress Test Runner (Left):

- Specs:** Shows a single spec named `spec.cy.js`.
- Status Bar:** Shows 1 failing test (red), 0 pending (yellow), and 0 passed (green). The duration is 128ms.
- Test Tree:** The test tree starts with `has title`, which fails under `BEFORE EACH`. It then branches into `request POST 200 /reset` and `TEST BODY`. The `TEST BODY` section contains:
 - `visit '/'`
 - `(uncaught exception) Error: App crash`
 - Error**
- Message:** "The following error originated from your application code, not from Cypress."
- Details:** "When Cypress detects uncaught errors originating from your application it will automatically fail the current test."
- Actions:** Buttons for "View stack trace" and "Print to console".

Browser Window (Right):

- Address Bar:** `http://localhost:3000/`
- Page Content:** A todo application with the heading "todos". A modal dialog is open with the heading "What needs to be done?". Inside the modal, there is a list item `{{ todo.title }}` preceded by an empty circle icon. Below the list item, the text "Loading data ..." is displayed. At the bottom of the modal, there are buttons for "All left Active Completed" and "Clear completed".
- Page Footer:** "cy-vs-pw-example-todomvc" and "Cypress vs Playwright course".

Cypress should automatically fail the test on any thrown error.

BONUS QUESTION: HOW TO IGNORE APP ERRORS IN CYPRESS?

See <https://glebbahmutov.com/blog/sanity-test/#tests>

FAIL PLAYWRIGHT TEST ON APP ERROR

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

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

// commands operate with respect to the "baseUrl"
// defined in the playwright.config.js file

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

test('has title', async ({ page }, testInfo) => {
  console.log(`running test "%s"`, testInfo.titlePath.join('/'))

  // if the application throws an unhandled error
  // we want to fail the test. Make sure to register
```

Playwright v1.40 fixed the async errors

```
// if the application throws an unhandled error
// we want to fail the test. Make sure to register
// the error callback before visiting the page
page.on('pageerror', (exception) => {
  throw new Error('App threw an error')
})
```

You still need to register the event listener

TEST ADDING AN ITEM

- clean up the existing code
 - git reset --hard
 - git clean -d -f
- git checkout b1
- npm install

PLAYWRIGHT TEST

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

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

// start each test with zero todos
test.beforeEach(async ({ request }) => {
  await request.post('/reset', { data: { todos: [] } })
})

// Tip: read the "Actions" Guide before implementing this test
// https://playwright.dev/docs/input
// and the "Locators" guide
// https://playwright.dev/docs/locators
test('adding todos', async ({ page }) => {
  // visit the application
  // wait for the body.loaded element to be visible
```

Locating elements <https://playwright.dev/docs/locators>

```
test('adding todos', async ({ page }) => {
  // visit the application
  await page.goto('/')
  await expect(page.locator('body.loaded')).toBeVisible()
  await expect(page.locator('.todo-list li')).toHaveLength(0)
  await page.getByPlaceholder('What needs to be done?').fill('Write code')
  await page.getByPlaceholder('What needs to be done?').press('Enter')
  await expect(page.locator('.todo-list li')).toHaveLength(1)
  await expect(page.locator('.todo-list li label')).toHaveText('Write code')
})
```

CYPRESS TEST

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

// start each test with zero todos
beforeEach(() => {
  cy.request('POST', '/reset', { todos: [] })
})

// Tip: look at the following commands before writing this test
// https://on.cypress.io/visit
// https://on.cypress.io/get
// https://on.cypress.io/type
// https://glebbahmutov.com/cypress-examples/commands/assertions.html
it('adding todos', () => {
  // visit the application
  // wait for the body.loaded element to be visible
  // there should be zero todo items
```

Selecting elements: <https://on.cypress.io/best-practices#Selecting-Elements>

```
it('adding todos', () => {
  cy.visit('/')
  cy.get('body.loaded').should('be.visible')
  cy.get('.todo-list li').should('have.length', 0)
  cy.get('[placeholder="What needs to be done?"]').type('Write code{enter}')
  cy.get('.todo-list li label')
    .should('have.length', 1)
    .and('have.text', 'Write code')
})
```

REFACTOR THE TESTS

Let's remove code duplication

- clean up the existing code
 - git reset --hard
 - git clean -d -f
- git checkout b2
- npm install

REUSE PLAYWRIGHT LOCATORS

```
test('adding todos', async ({ page }) => {
  // avoid duplicator code by reusing the same locator objects
  await page.goto('/')
  await expect(page.locator('body.loaded')).toBeVisible()
  await expect(page.locator('.todo-list li')).toHaveLength(0)
  await page.getByPlaceholder('What needs to be done?').fill('Write code')
  await page.getByPlaceholder('What needs to be done?').press('Enter')
  await expect(page.locator('.todo-list li')).toHaveLength(1)
  await expect(page.locator('.todo-list li label')).toHaveText('Write code')
})
```

```
test('adding todos', async ({ page }) => {
  // avoid duplicator code by reusing the same locator objects
  const input = page.getByPlaceholder('What needs to be done?')
  const todos = page.locator('.todo-list li label')

  await page.goto('/')
  await page.locator('body.loaded').waitFor()
  await expect(todos).toHaveLength(0)
  await input.fill('Write code')
  await input.press('Enter')
  await expect(todos).toHaveText(['Write code'])
})
```

Playwright solution

REMOVE CYPRESS CODE DUPLICATION

```
it('adding todos', () => {
  // reuse the same CSS selectors
  cy.visit('/')
  cy.get('body.loaded').should('be.visible')
  cy.get('.todo-list li').should('have.length', 0)
  cy.get('[placeholder="What needs to be done?"]')
    .type('Write code{enter}')
  cy.get('.todo-list li label')
    .should('have.length', 1)
    .and('have.text', 'Write code')
})
```

```
it('adding todos', () => {
  // reuse the same CSS selectors
  const input = '[placeholder="What needs to be done?"]'
  const todos = '.todo-list li label'

  cy.visit('/')
  cy.get('body.loaded').should('be.visible')
  cy.get(todos).should('not.exist')
  cy.get(input).type('Write code{enter}')
  cy.get(todos).should('have.length', 1).and('have.text', 'Write code')
})
```

Cypress solution



ADDING ITEMS

- make your tests fail on the unexpected app errors
- select elements using best practices
- simplify the code

→ Pick the [next section](#) or jump to the [03-completing-items](#) chapter