ASSERTIONS AND RETRIES

YOU WILL LEARN

- how assertions force commands to retry
- writing command chains

Question: how would you check if there are more than 2 elements on the page?

```
cy.get(todos).should('have.length.greaterThan', 2)
```

Cypress assertion

What if you cannot use `have.length.greaterThan`? We can write our own assertion logic

```
cy.get(todos).should(($el) => {
  if ($el.length <= 2) {
    throw new Error('Need at least 2 elements')
  }
})
```

```
cy.get(todos).should(($el) => {
  // synchronous logic inside should(callback)
})
```

Cypress `should(callback)` assertion function

Playwright does not have `have.length.greaterThan` so we can run a "retry" callback

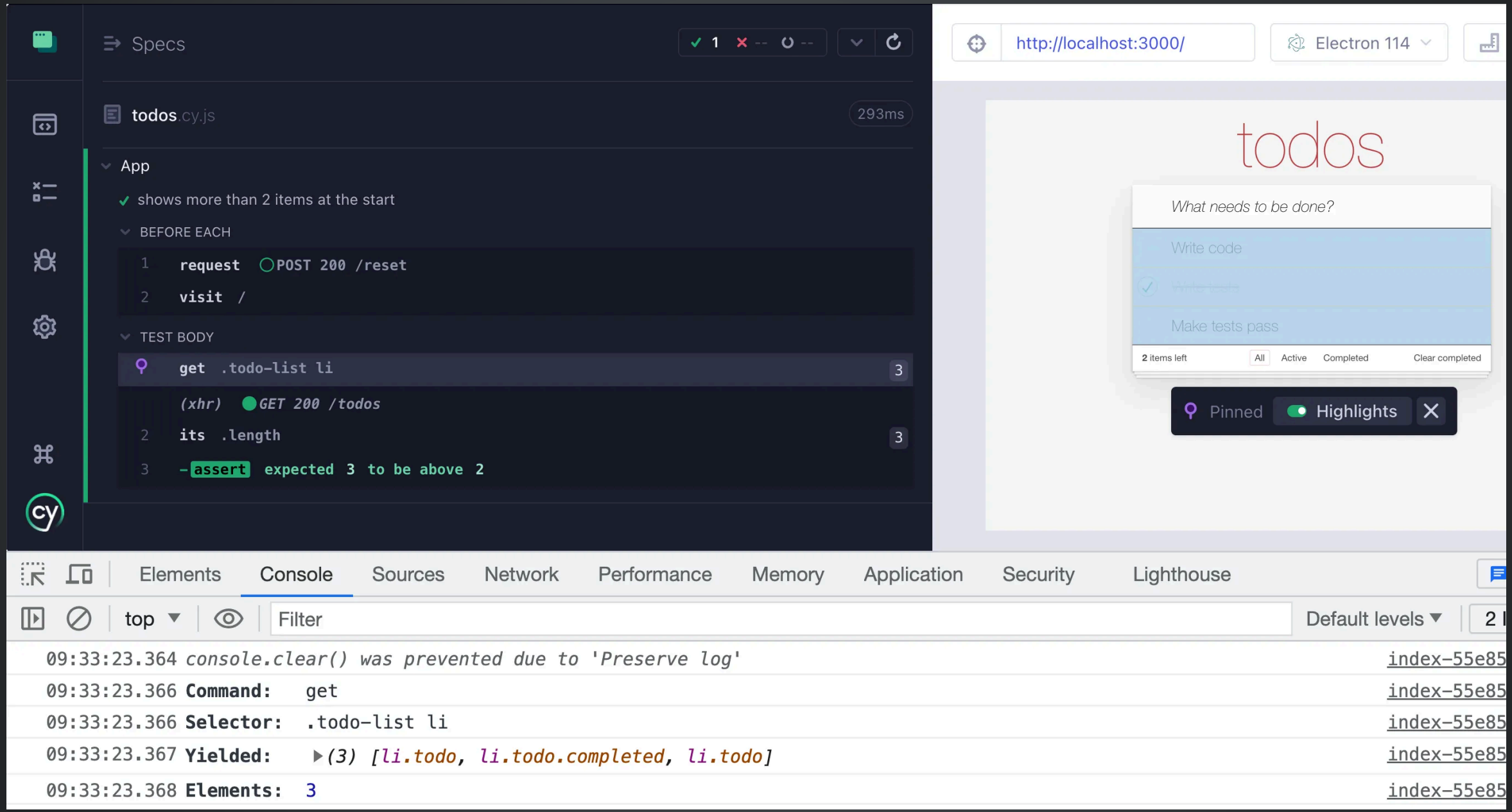
```
const todos = page.locator('.todo-list li')

await expect(async () => {
  const count = await todos.count()
  expect(count).toBeGreaterThan(2)
}).toPass()
```

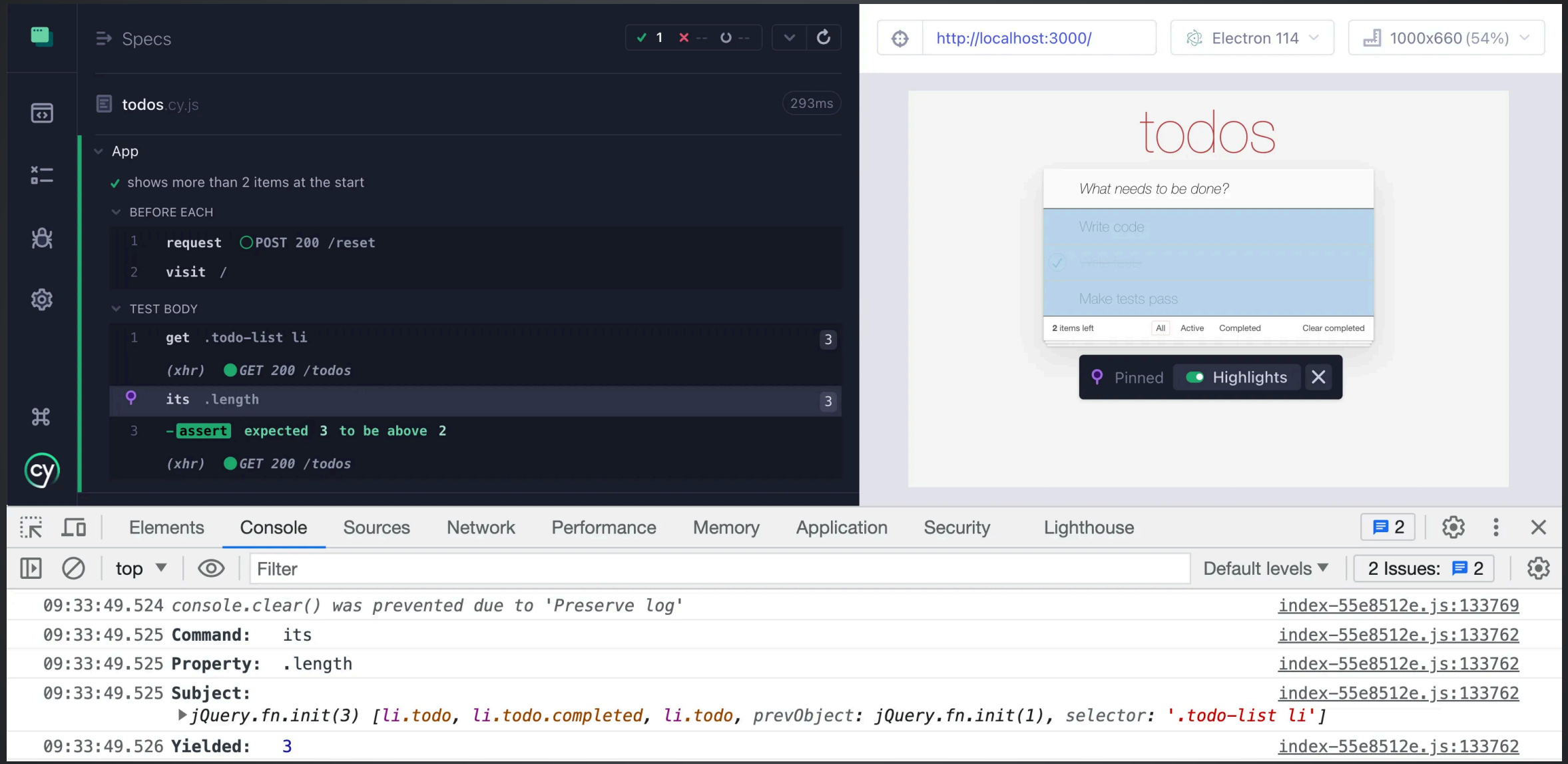
CYPRESS CHAIN

```
// rewrite the following assertion using a chain of commands
// - get the elements, which in Cypress yields a jQuery object (1)
// - get the property `length` from that jQuery object (2)
// - confirm the number is greater than 2 (3)
// - if the assertion (3) fails, go back to (1)
cy.get(todos).should(($el) => {
  expect($el.length, 'more than 2 elements').to.be.greaterThan(2)
})
```

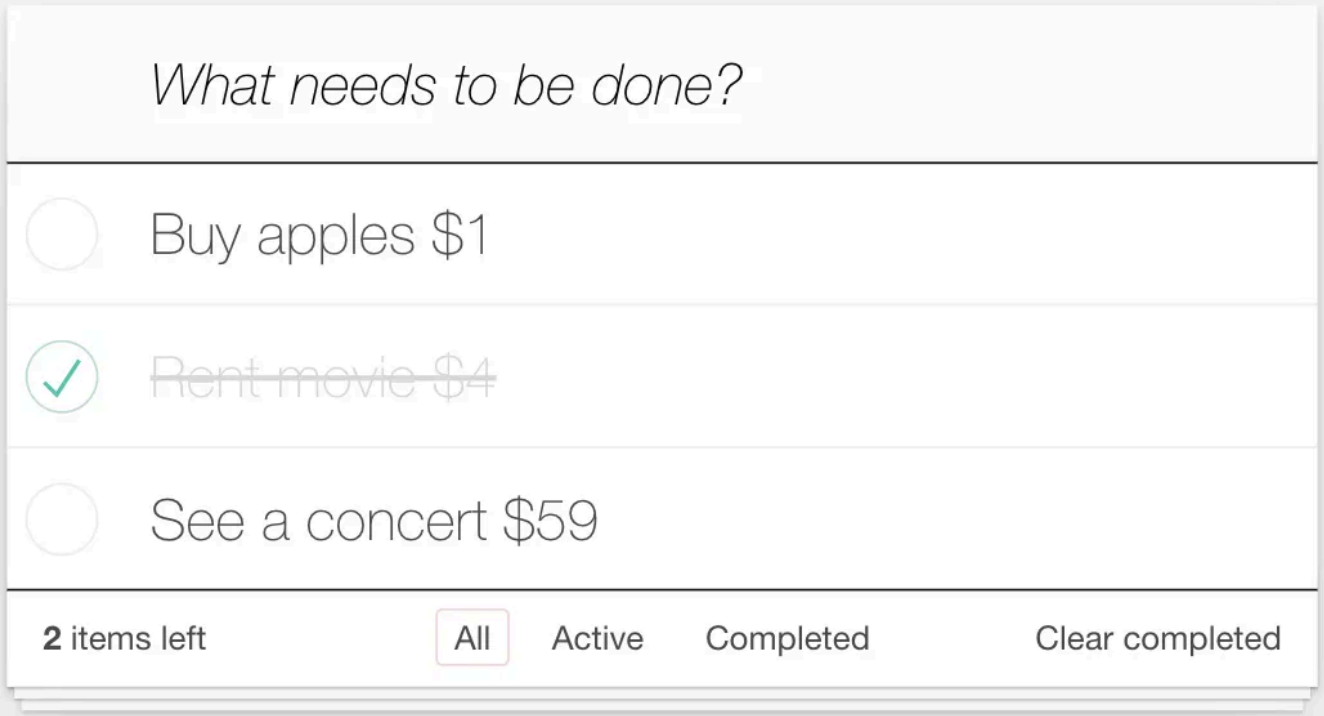
```
// rewrite the following assertion using a chain of commands
// - get the elements, which in Cypress yields a jQuery object (1)
// - get the property `length` from that jQuery object (2)
// - confirm the number is greater than 2 (3)
// - if the assertion (3) fails, go back to (1)
cy.get(todos).its('length').should('be.greaterThan', 2)
```

Each step can be debugged!


Specs
1
todos.cy.js
293ms
App
shows more than 2 items at the start
BEFORE EACH
1 request POST 200 /reset
2 visit /
TEST BODY
get .todo-list li
3
(xhr) GET 200 /todos
2 its .length
3
3 -assert expected 3 to be above 2
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
Pinned
Highlights
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Lighthouse
top
Filter
Default levels
09:33:23.364 console.clear() was prevented due to 'Preserve log'
09:33:23.366 Command: get
09:33:23.366 Selector: .todo-list li
09:33:23.367 Yielded: (3) [li.todo, li.todo.completed, li.todo]
09:33:23.368 Elements: 3
index-55e85


Specs
1
todos.cy.js
293ms
App
shows more than 2 items at the start
BEFORE EACH
1 request POST 200 /reset
2 visit /
TEST BODY
1 get .todo-list li 3
(xhr) GET 200 /todos
its .length 3
3 -assert expected 3 to be above 2
(xhr) GET 200 /todos
http://localhost:3000/
Electron 114
1000x660 (54%)
todos
What needs to be done?
Write code
Write tests
Make tests pass
2 items left
All
Active
Completed
Clear completed
Pinned
Highlights
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Lighthouse
2
top
Filter
Default levels
2 Issues: 2
09:33:49.524 console.clear() was prevented due to 'Preserve log'
index-55e8512e.js:133769
09:33:49.525 Command: its
index-55e8512e.js:133762
09:33:49.525 Property: .length
index-55e8512e.js:133762
09:33:49.525 Subject:
index-55e8512e.js:133762
jQuery.fn.init(3) [li.todo, li.todo.completed, li.todo, prevObject: jQuery.fn.init(1), selector: '.todo-list li']
09:33:49.526 Yielded: 3
index-55e8512e.js:133762

CONFIRM SORTED PRICES



How can you confirm the prices are sorted?

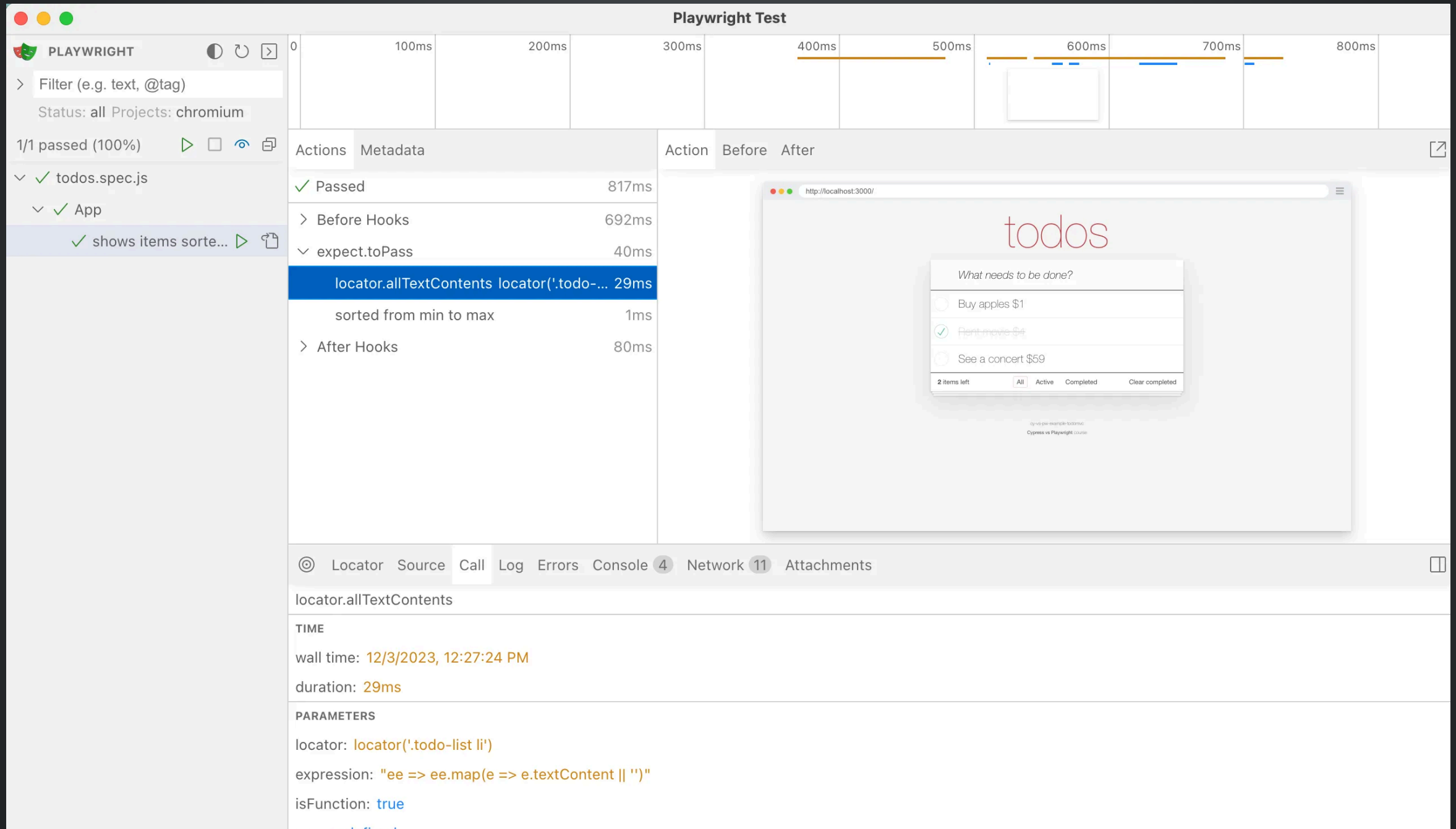
- `git checkout c5`

Todo: From each Todo item you need:

- grab its text
- use a regular expression to find the $ match
- convert the string to a number
- confirm the list of numbers is sorted

PLAYWRIGHT SOLUTION

```
const todos = page.locator('.todo-list li')
await expect(async () => {
  const titles = await todos.allTextContents()
  const matches = titles.map((s) => s.match(/\$(?<price>\d+)/))
  const strings = matches.map((m) => m?.groups?.price)
  // @ts-ignore
  const prices = strings.map(parseFloat)
  const sorted = structuredClone(prices).sort()
  expect(sorted, 'sorted from min to max').toEqual(prices)
}).toPass()
```


Playwright Test
PLAYWRIGHT
Filter (e.g. text, @tag)
Status: all Projects: chromium
1/1 passed (100%)
todos.spec.js
App
shows items sorte...
0
100ms
200ms
300ms
400ms
500ms
600ms
700ms
800ms
Actions
Metadata
Passed 817ms
Before Hooks 692ms
expect.toPass 40ms
locator.allTextContents locator('.todo-... 29ms
sorted from min to max 1ms
After Hooks 80ms
Action
Before
After
http://localhost:3000/
todos
What needs to be done?
Buy apples $1
See a concert $59
2 items left
All
Active
Completed
Clear completed
Locator
Source
Call
Log
Errors
Console 4
Network 11
Attachments
locator.allTextContents
TIME
wall time: 12/3/2023, 12:27:24 PM
duration: 29ms
PARAMETERS
locator: locator('.todo-list li')
expression: "ee => ee.map(e => e.textContent || '')"
isFunction: true

CYPRESS SOLUTION

```
cy.get(todos).should(($el) => {
  const titles = Cypress._.map($el, 'innerText')
  const matches = titles.map((s) => s.match(/\$(?<price>\d+)/))
  const strings = matches.map((m) => m?.groups?.price)
  const prices = strings.map(parseFloat)
  const sorted = Cypress._.sortBy(prices)
  expect(sorted).to.not.be.empty
  expect(sorted, 'sorted from min to max').to.deep.equal(prices)
})
```

We cannot simplify the Playwright code. But we can rewrite Cypress `should(callback)` into a series of separate steps.

Note: this exercise is in the branch `c6`

```
cy.get(todos)
  .then(($el) => Cypress._.map($el, 'innerText'))
  .then((titles) => titles.map((s) => s.match(/\$(?<price>\d+)/)))
  .then((matches) => matches.map((m) => m?.groups?.price))
  // @ts-ignore
  .then((strings) => strings.map(parseFloat))
  .should((prices) => {
    const sorted = Cypress._.sortBy(prices)
    expect(sorted).to.not.be.empty
    expect(sorted, 'sorted from min to max').to.deep.equal(prices)
  })
```

Replace a single `should(callback)` with a series of `cy.then(callback)...` Every step does something with the argument (the subject) and passes the result to the next step.

CYPRESS-MAP

```
.then(($el) => Cypress._.map($el, 'innerText'))
```

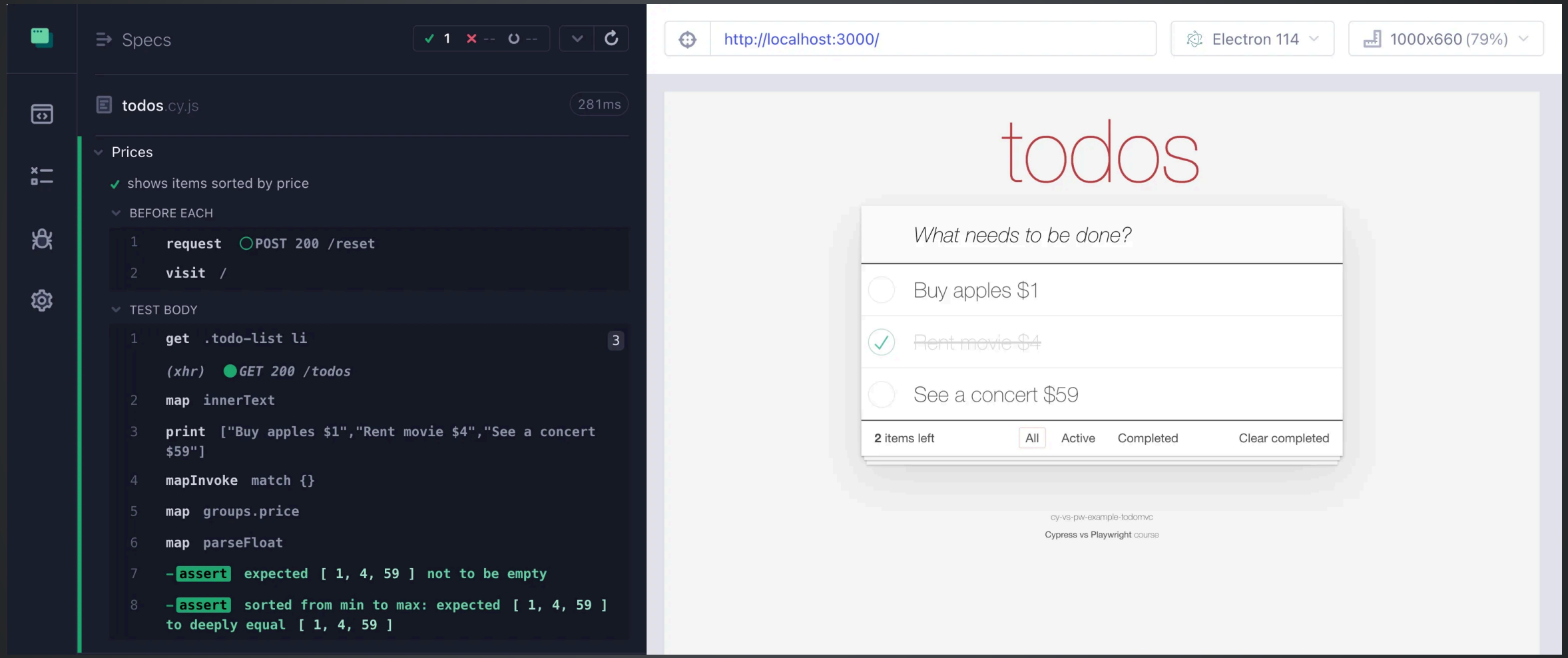
I have plugin cypress-map that can map each item to a property

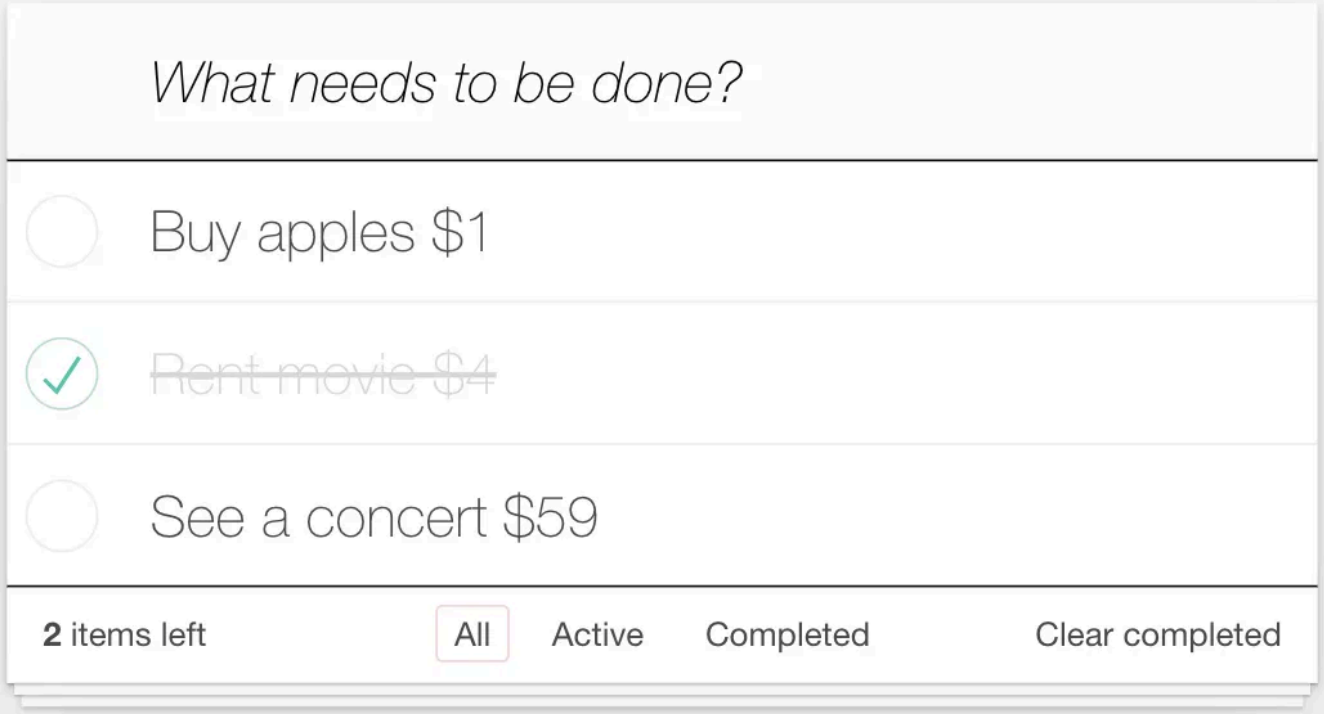
```
.map('innerText')
```

Rewrite the `cy.then(callback)...` commands using `cypress-map` queries. **Note:** this exercise is in the branch `c7`

```
cy.get(todos)
  .map('innerText')
  .mapInvoke('match', /\$(?<price>\d+)/)
  .map('groups.price')
  .map(parseFloat)
  .should((prices) => {
    const sorted = Cypress._.sortBy(prices)
    expect(sorted).to.not.be.empty
    expect(sorted, 'sorted from min to max').to.deep.equal(prices)
  })
```

Each step is now shown in the Command Log and can be debugged





Prices should be $1, $4, and $59

DIFFERENT SOLUTIONS = DIFFERENT RETRIES

Note: this exercise is in the branch c8

```
it('shows items sorted by price - C', () => {
  const todos = '.todo-list li'

  cy.get(todos).should(($el) => {
    const titles = Cypress._.map($el, 'innerText')
    console.log(titles)
    const matches = titles.map((s) => s.match(/\$(?<price>\d+)/))
    const strings = matches.map((m) => m?.groups?.price)
    // @ts-ignore
    const prices = strings.map(parseFloat)
    expect(prices).to.deep.equal([1, 4, 59])
  })
})
```

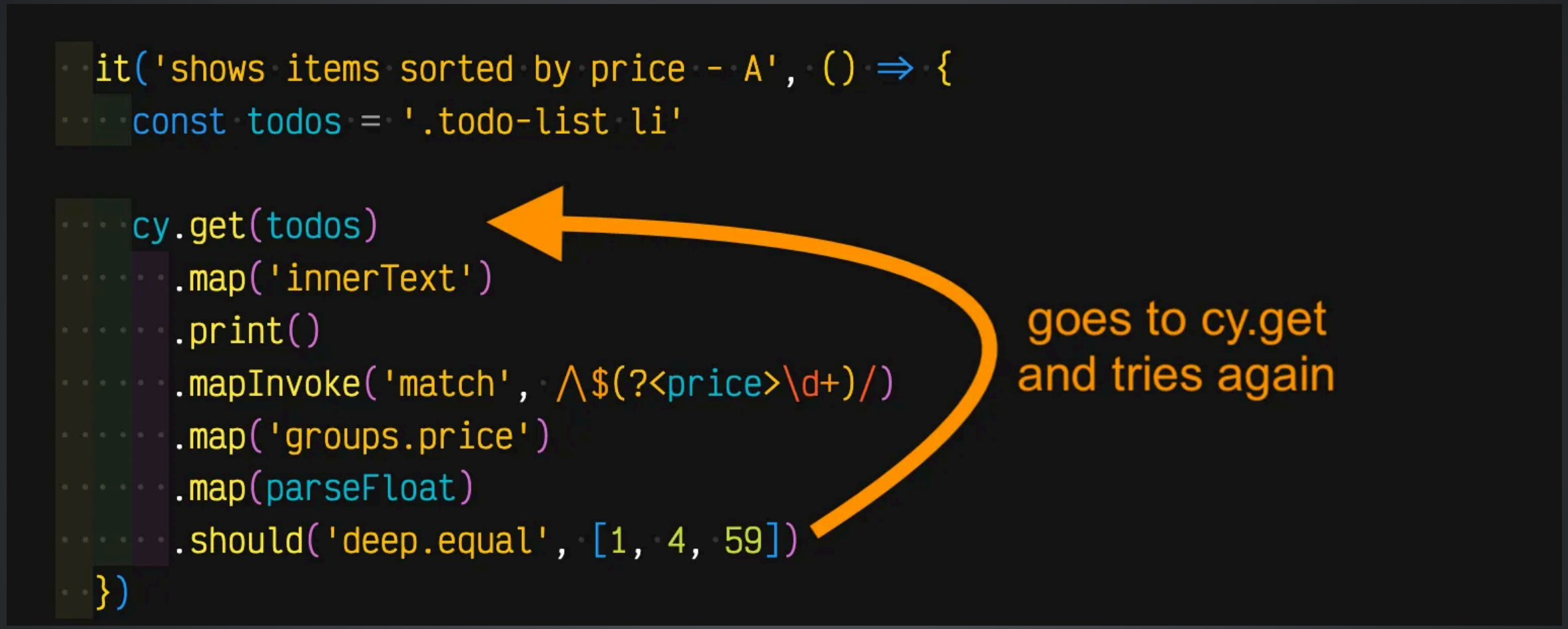
Error in should(callback) goes back to the cy.get query command

Cypress: `cy.get` is retried

```
it('shows items sorted by price - B', () => {
  const todos = '.todo-list li'

  cy.get(todos)
    .then(($el) => Cypress._.map($el, 'innerText'))
    .then((titles) => titles.map((s) => s.match(/\$(?<price>\d+)/)))
    .then((matches) => matches.map((m) => m?.groups?.price))
    // @ts-ignore
    .then((strings) => strings.map(parseFloat))
    .should('deep.equal', [1, 4, 59])
})
```

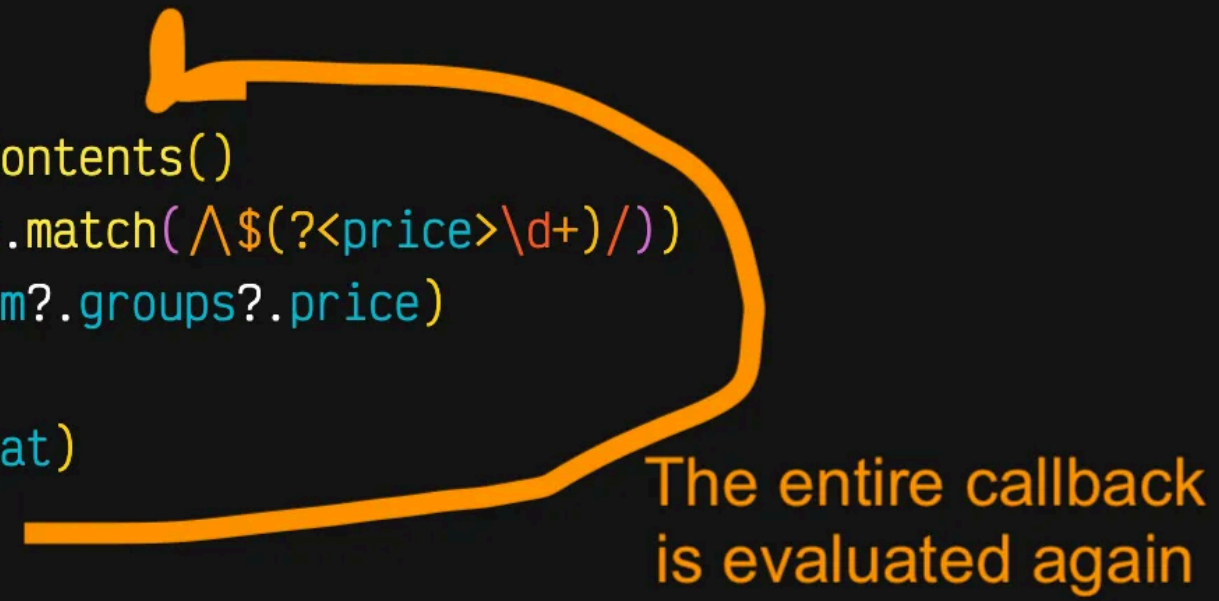
Cannot retry
Previous command
is NOT a query

Cypress: `cy.then` stops the retries, the test probably fails if the prices are not shown right away



Cypress: The entire chain of queries (safe) is retried, including `cy.get`.
The test is solid.

```
test('shows items sorted by price - D', async ({ page }) => {
  const todos = page.locator('.todo-list li')

  await expect(async () => {
    const titles = await todos.allTextContents()
    const matches = titles.map((s) => s.match(/\$(?<price>\d+)/))
    const strings = matches.map((m) => m?.groups?.price)
    // @ts-ignore
    const prices = strings.map(parseFloat)
    expect(prices).toEqual([1, 4, 59])
  }).toPass()
})
```



Playwright: elements are queried and the entire callback is retried

```
test('shows items sorted by price - E', async ({ page }) => {
  const todos = page.locator('.todo-list li')

  const titles = await todos.allTextContents()
  await expect(async () => {
    const matches = titles.map((s) => s.match(/\$(?<price>\d+)/))
    const strings = matches.map((m) => m?.groups?.price)
    // @ts-ignore
    const prices = strings.map(parseFloat)
    expect(prices).toEqual([1, 4, 59])
  }).toPass({ timeout: 5000 })
})
```

The callback is evaluated which does NOT include getting the elements

Playwright: Getting the elements' text is *outside* the callback, so elements are *not* queried again. The test might be flaky or fail.

Cypress: chain of queries is retried. Are shown nicely in the Command Log.

Playwright: commands inside the callback `expect(...).toPass()` are retried, not really shown in the UI

Tip: use cypress-recurse for something similar to `expect(...).toPass()`

🏁 ASSERTIONS AND RETRIES

- commands vs assertions
- query again elements to make stable assertions

➡️ Pick the next section or jump to the 09-app-access chapter