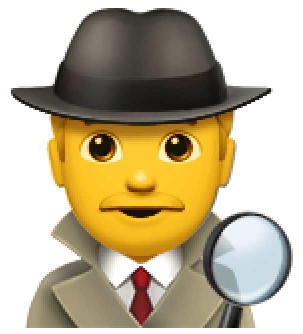
Testing Webapps 🧐

Hendrik

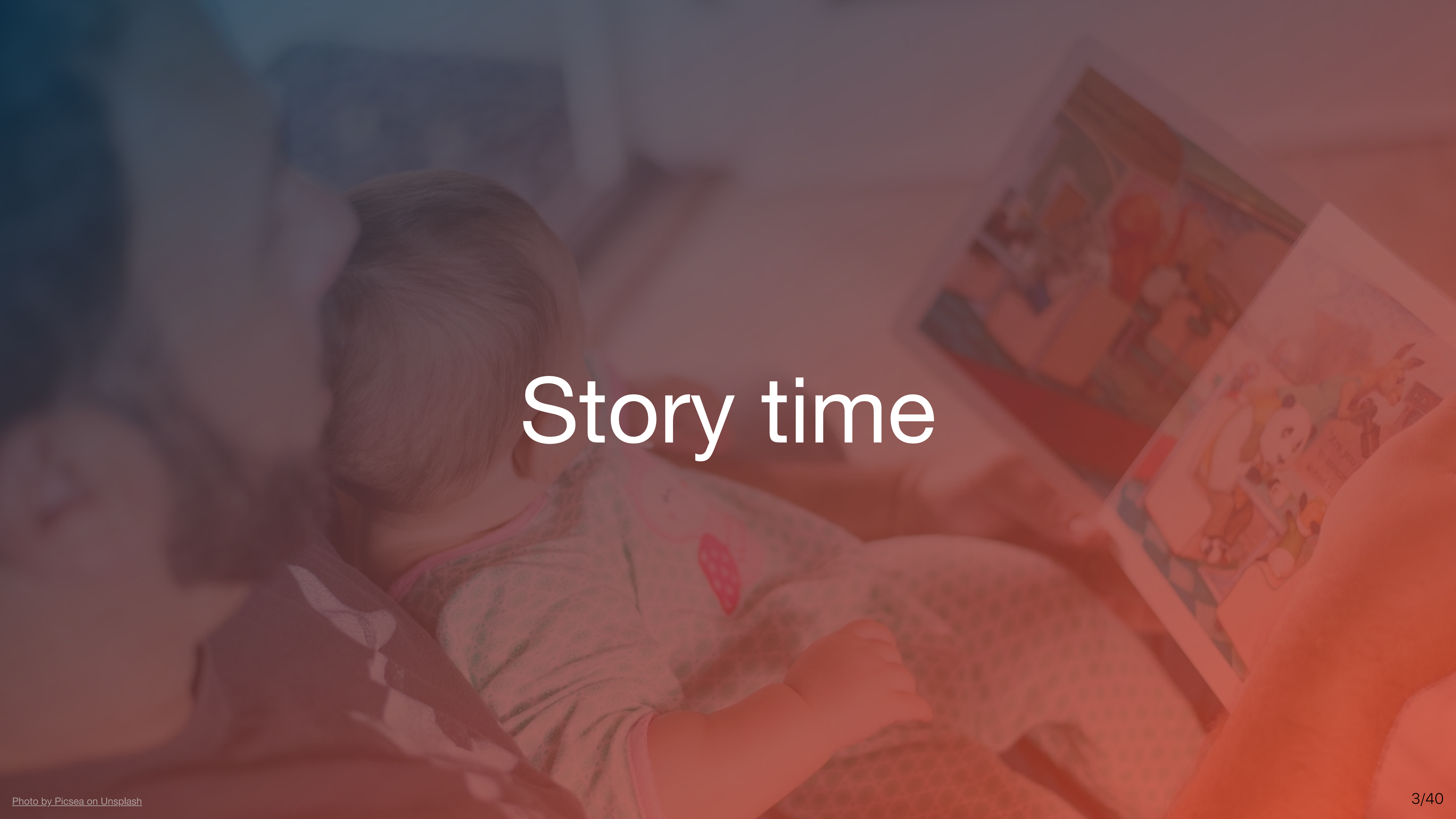
Developer for fun 🎉



Testing

- Find bugs before users do
- Ensure we fulfil specification
- Help future developers

Timeinvestment



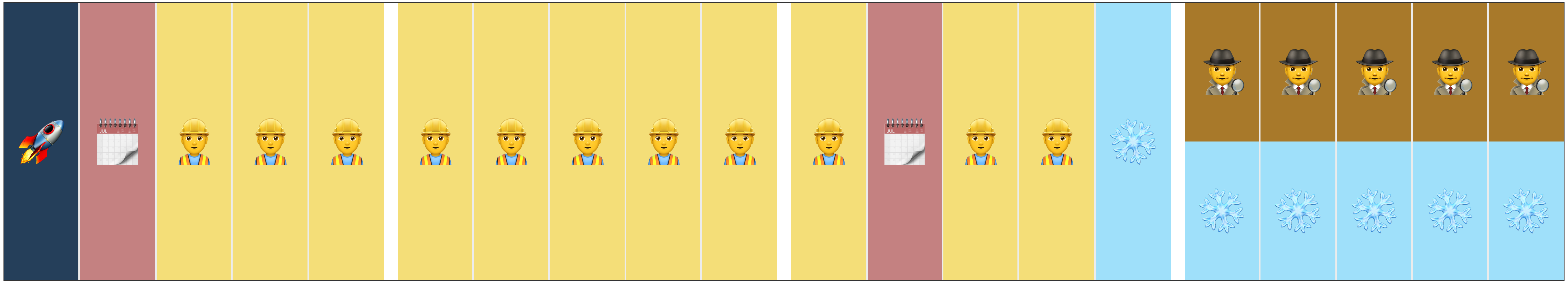
Story time

Photo by Picsea on Unsplash

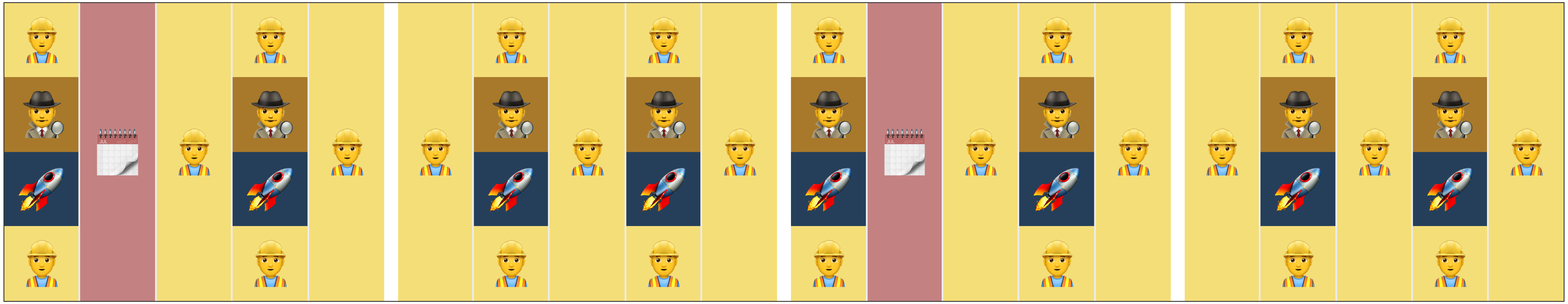
Legend

📆 Meetings	👷 Work on product
🕵️ Manual testing	🚀 Releasing software

Codefreeze



Manual Releases



Dev	Test	Meet	Releasing	Releases
11	5	2	1	1
14	2	2	2	8

Timeinvestment in days

Dev	Test	Meet	Releasing	Releases	Bugs
11	5	2	1	**1**	🐛🐛🐛🐛🐛
14	2	2	2	**8**	🐛🐛

Timeinvestment in days

"IT IS ALL ABOUT CYCLES"

Photo by Jeff Fielitz on Unsplash

High performing organizations

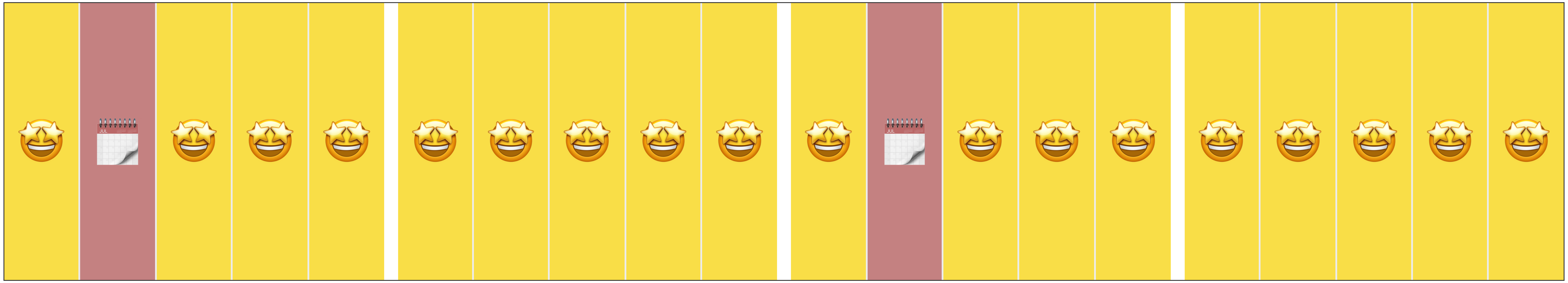
- Deploy **200** times as often
- Have a **3** times lower change failure rate
- Recover **24** times faster from failures

- DevOps Report 2016

Legend

🤩
Perfect day

My dream



Dev	Test	Meet	Releasing	Releases	Bugs
11	5	2	1	1	🐛🐛🐛🐛🐛
14	2	2	2	8	🐛🐛
18	0	2	0	200	🐛

Timeinvestment in days

Actors

👨‍💻

Developer

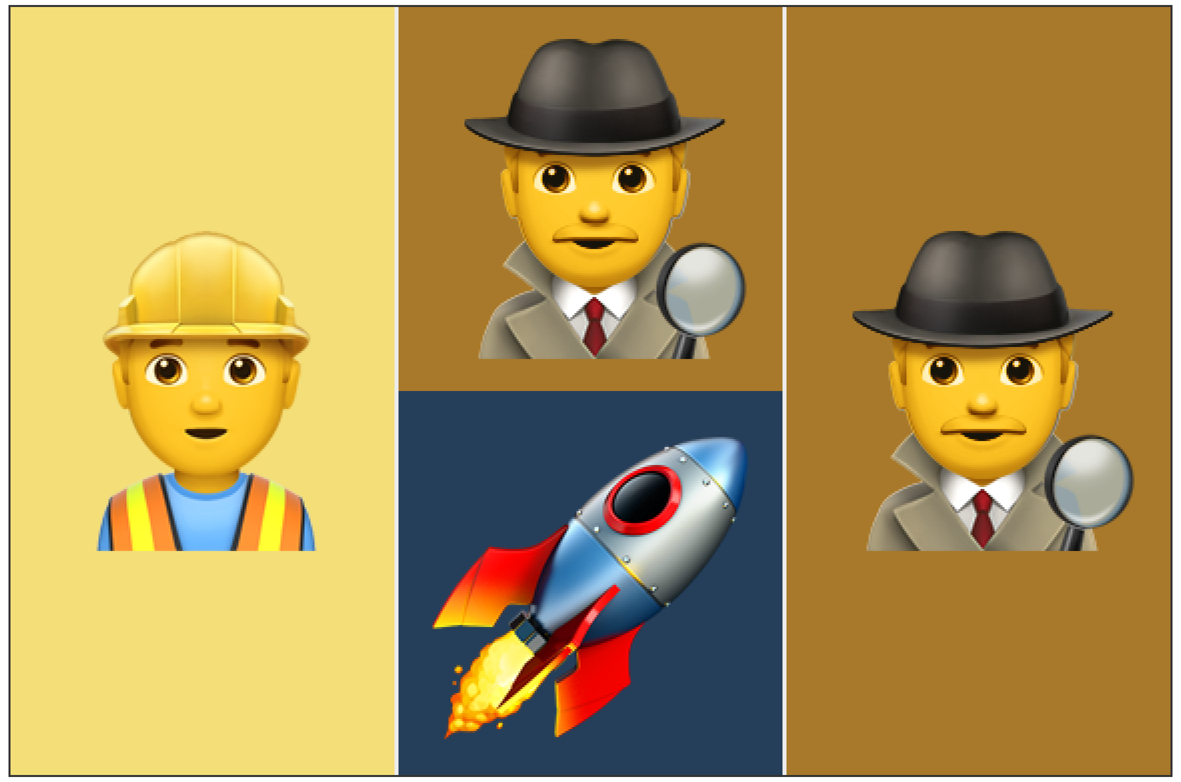
🧐

Tester / QA

🤖

Computer

👨‍💻 🤖 🧐

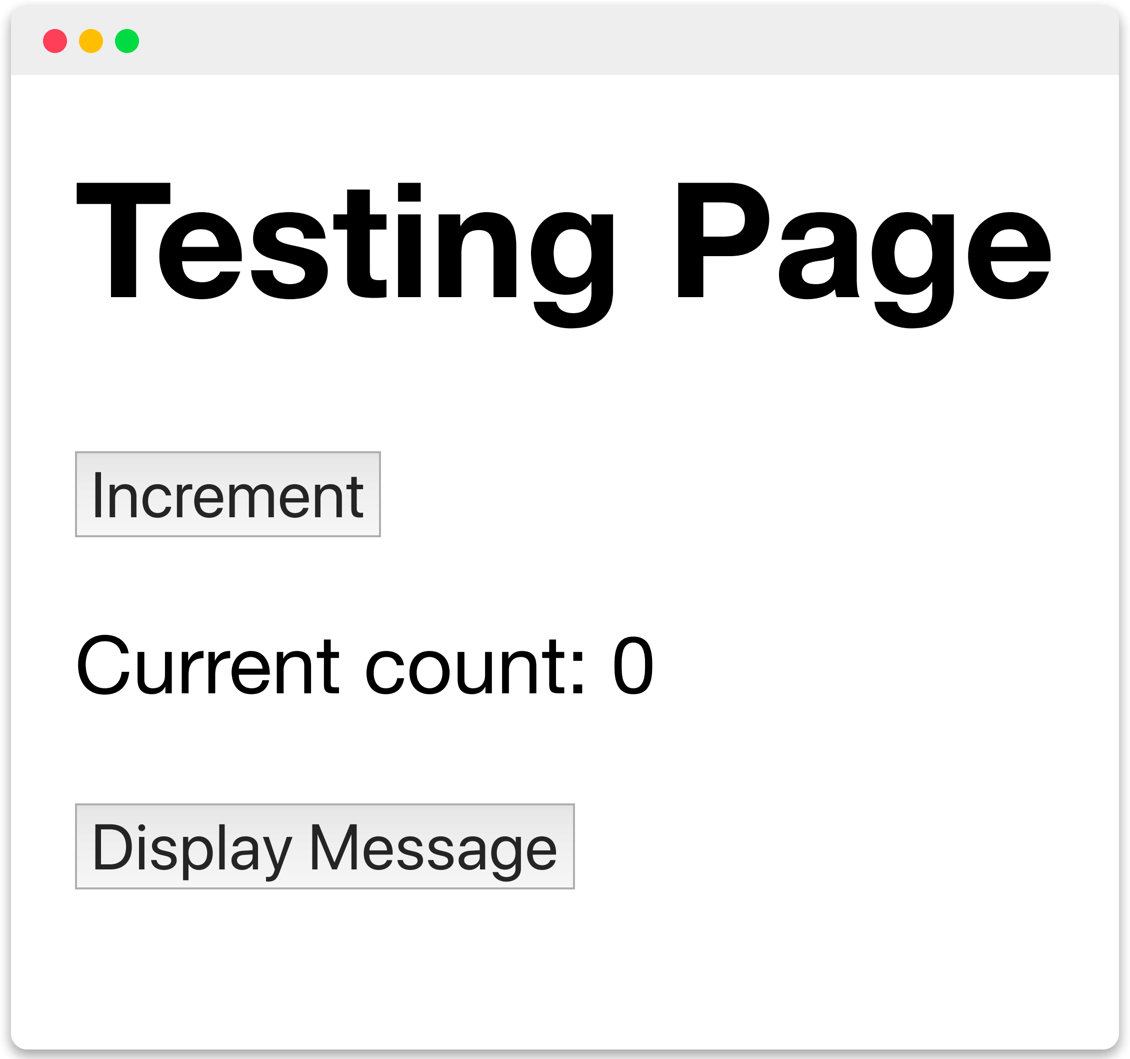


Testing 🔧

“Use the right tool for the right job„

Photo by Cesar Carlevarino Aragon on Unsplash cypress.io

e2e and integration testing



Testing Page

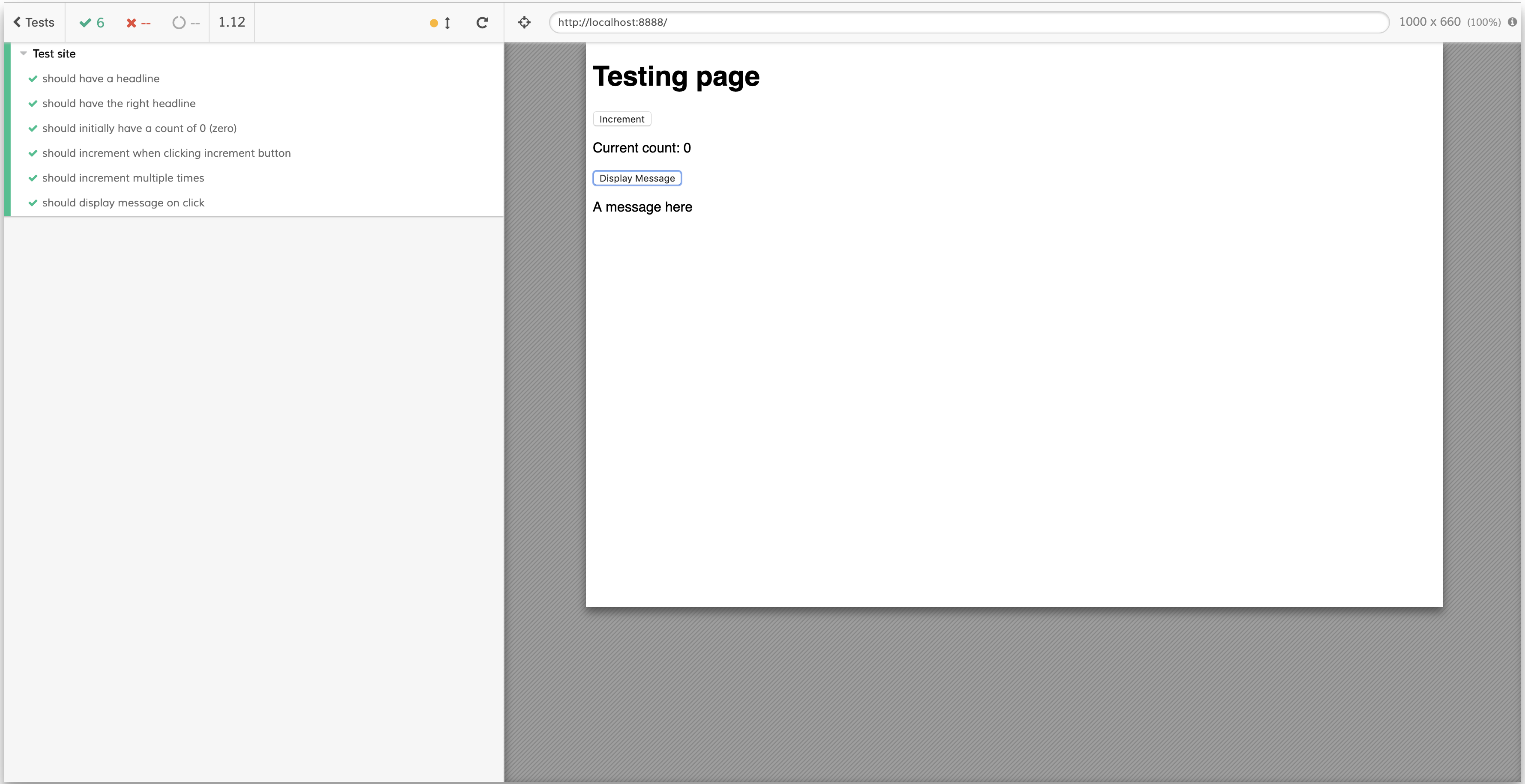
Increment

Current count: 0

Display Message

Testcases

1. Basic render 🎉
2. Counter logic 🧮
3. Messsage display 📜


< Tests ✓ 6 ✗ -- ◯ -- 1.12

http://localhost:8888/ 1000 x 660 (100%)

Test site
- ✓ should have a headline
- ✓ should have the right headline
- ✓ should initially have a count of 0 (zero)
- ✓ should increment when clicking increment button
- ✓ should increment multiple times
- ✓ should display message on click

Testing page

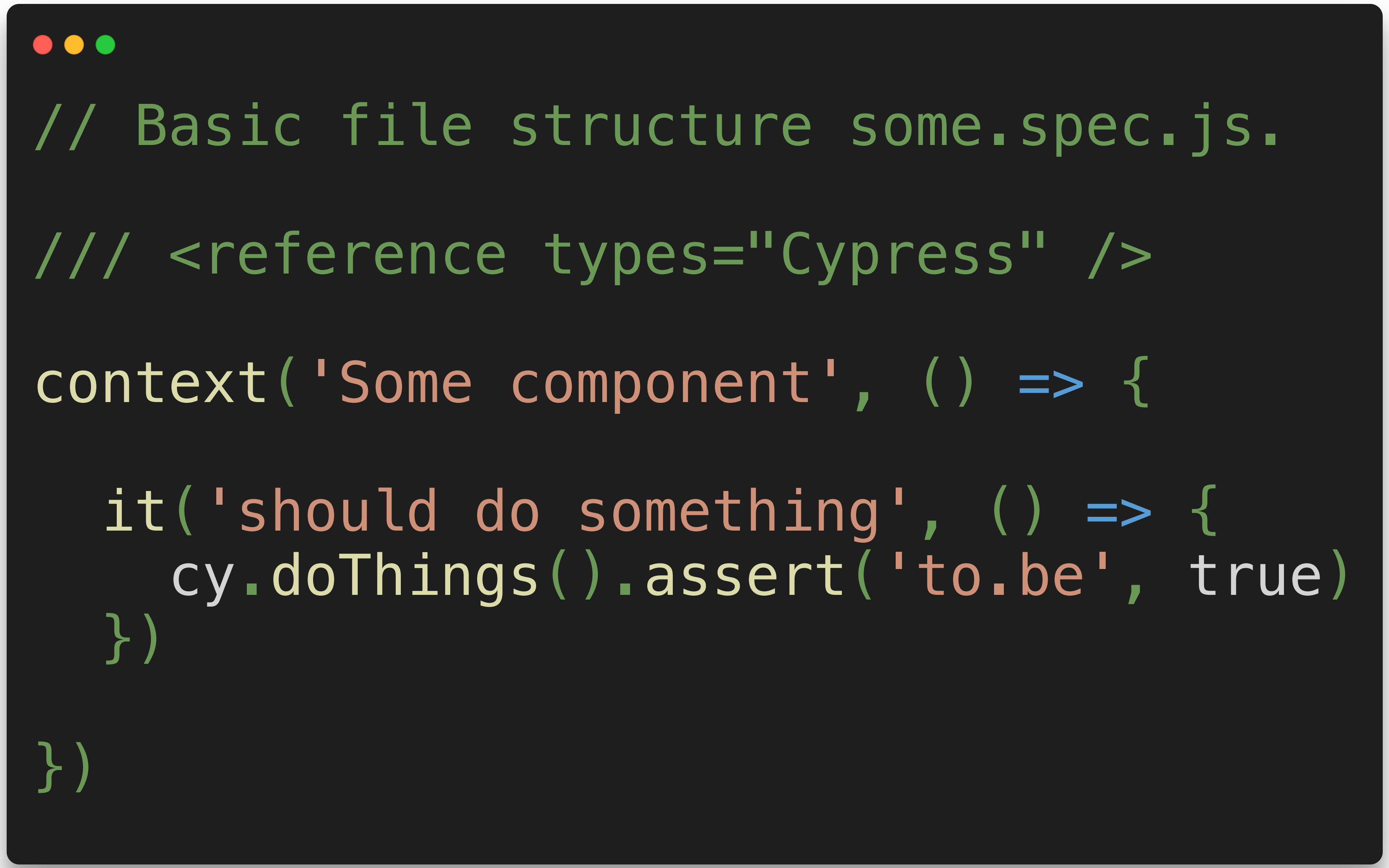
Increment

Current count: 0

Display Message

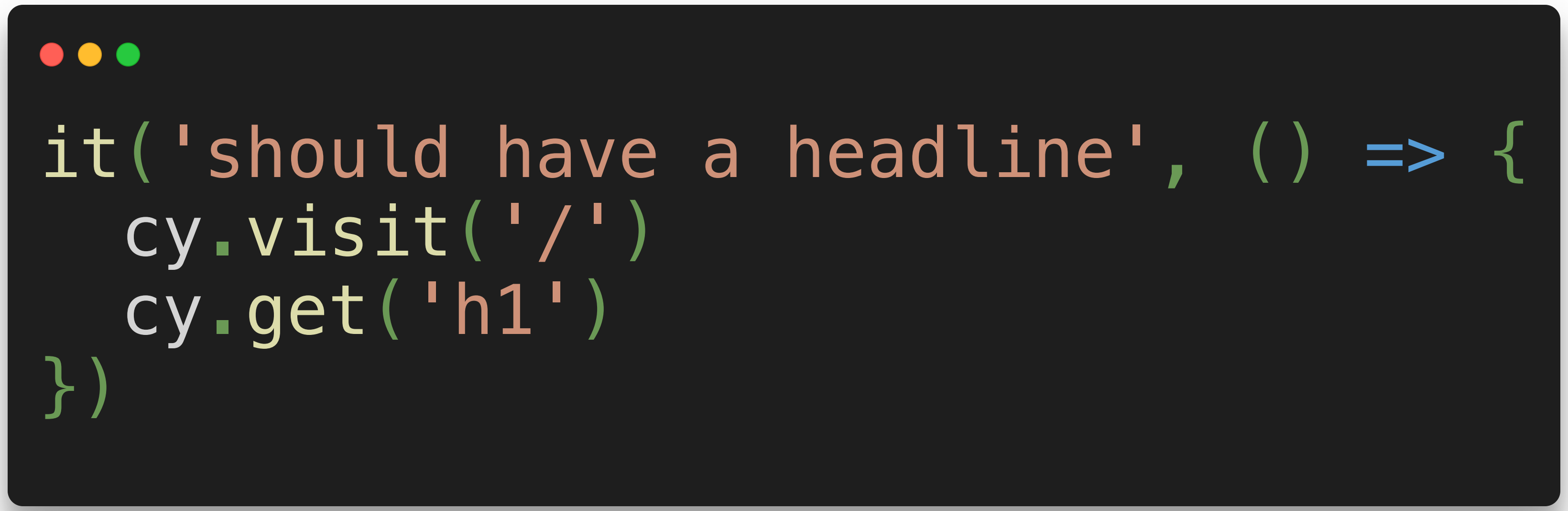
A message here

Cypress setup



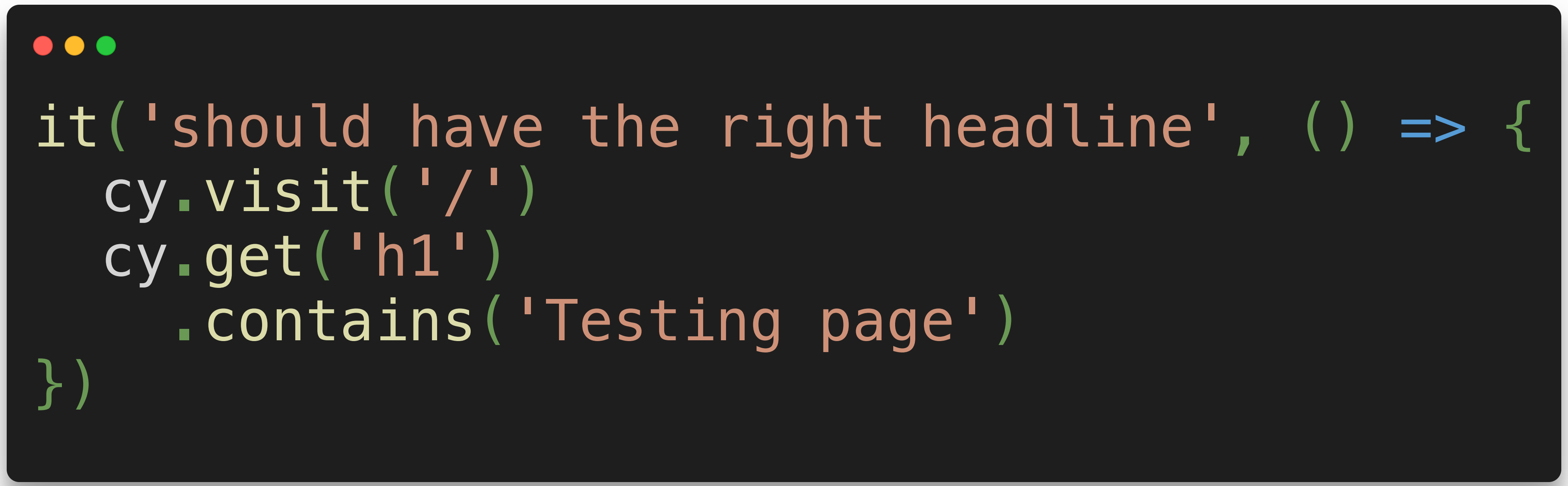
```
// Basic file structure some.spec.js.

/// <reference types="Cypress" />

context('Some component', () => {

  it('should do something', () => {
    cy.doThings().assert('to.be', true)
  })

})
```

1. Basic render 🎉



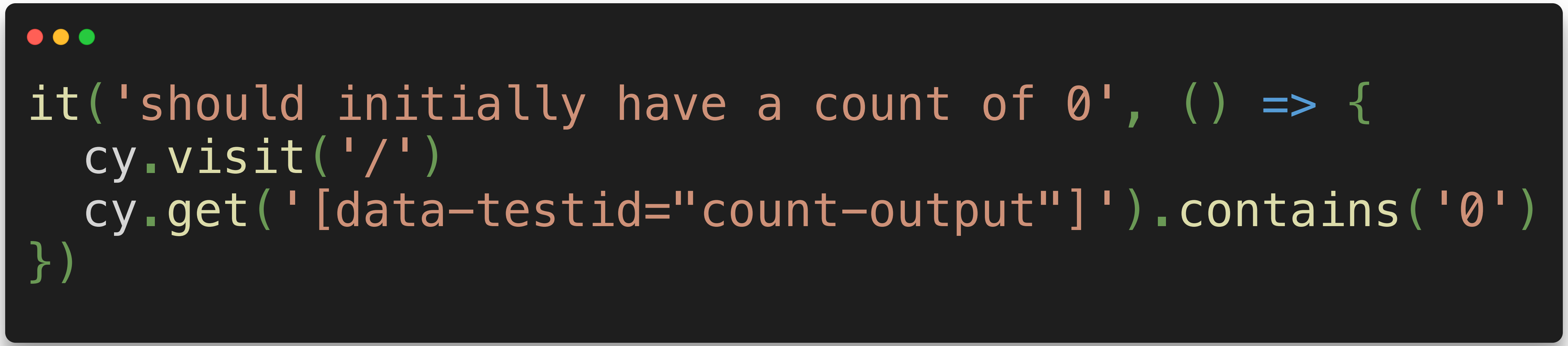
```
it('should have a headline', () => {
  cy.visit('/')
  cy.get('h1')
})
```

1. Basic render 🎉



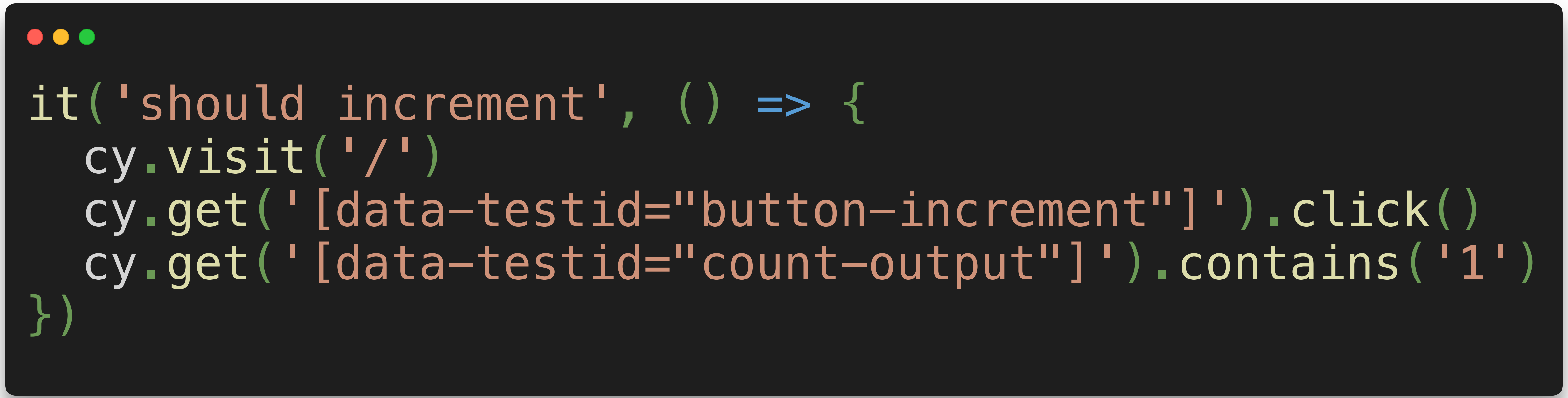
```javascript
it('should have the right headline', () => {
  cy.visit('/')
  cy.get('h1')
    .contains('Testing page')
})
```

2. Counter logic 🧮



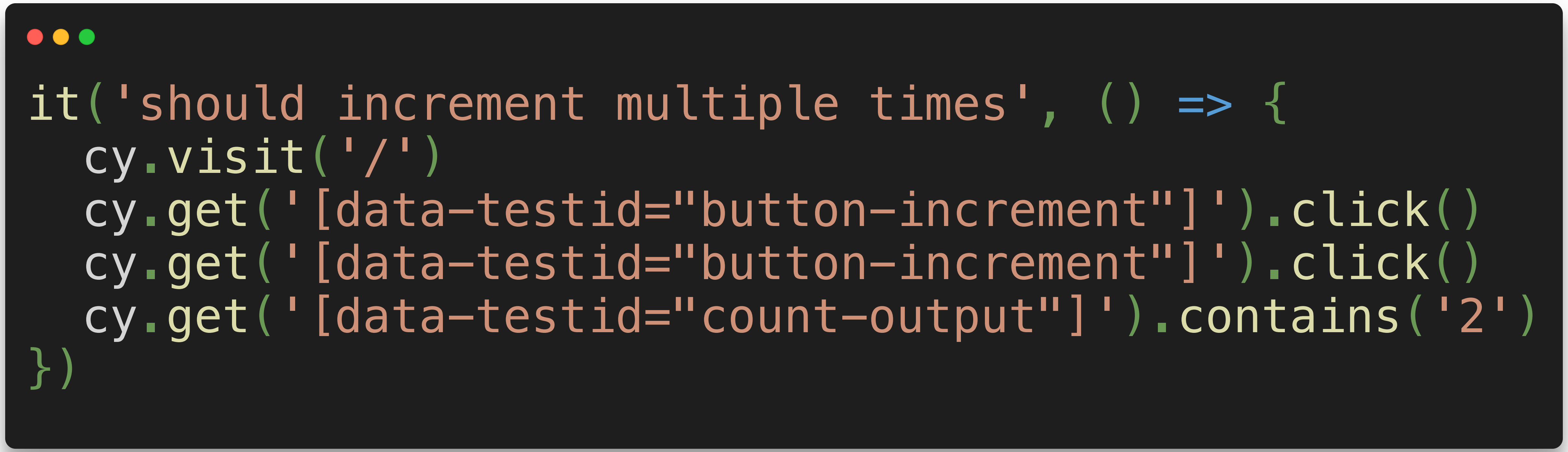
```
it('should initially have a count of 0', () => {
  cy.visit('/')
  cy.get('[data-testid="count-output"]').contains('0')
})
```

2. Counter logic 🧮



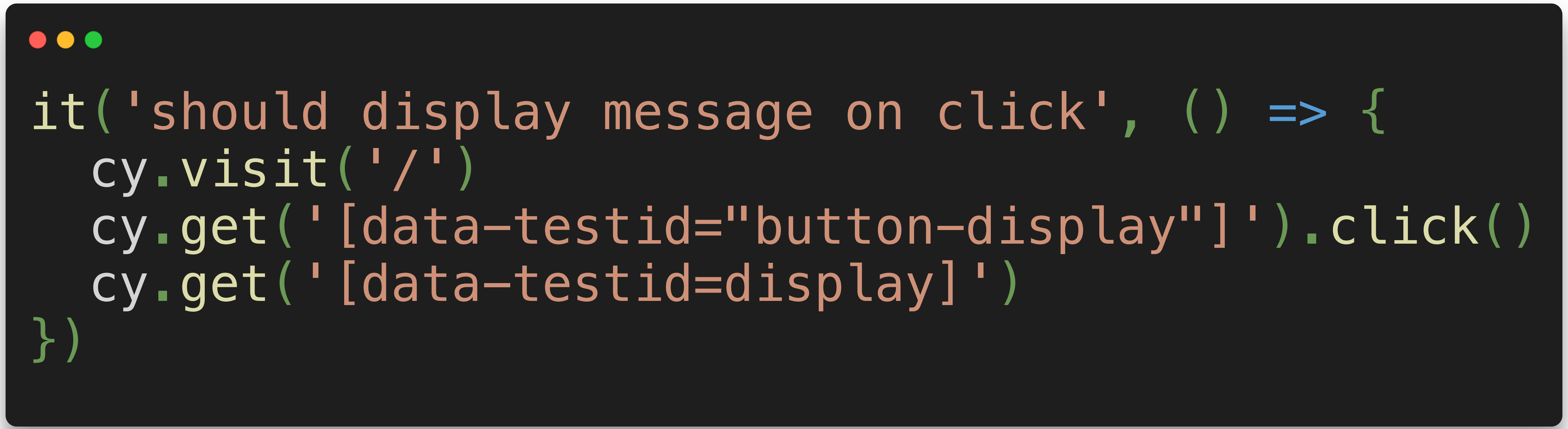
```
it('should increment', () => {
  cy.visit('/')
  cy.get('[data-testid="button-increment"]').click()
  cy.get('[data-testid="count-output"]').contains('1')
})
```

2. Counter logic 🧮



```
it('should increment multiple times', () => {
  cy.visit('/')
  cy.get('[data-testid="button-increment"]').click()
  cy.get('[data-testid="button-increment"]').click()
  cy.get('[data-testid="count-output"]').contains('2')
})
```

3. Messsage display 📜



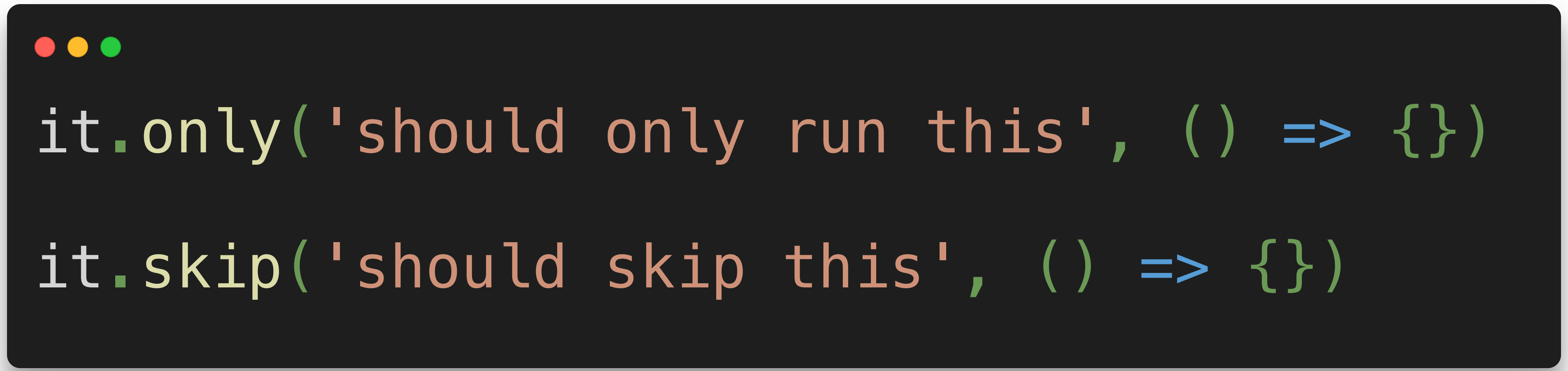
```
it('should display message on click', () => {
  cy.visit('/')
  cy.get('[data-testid="button-display"]').click()
  cy.get('[data-testid=display]')
})
```

Beyond the basics

Photo by NASA on Unsplash

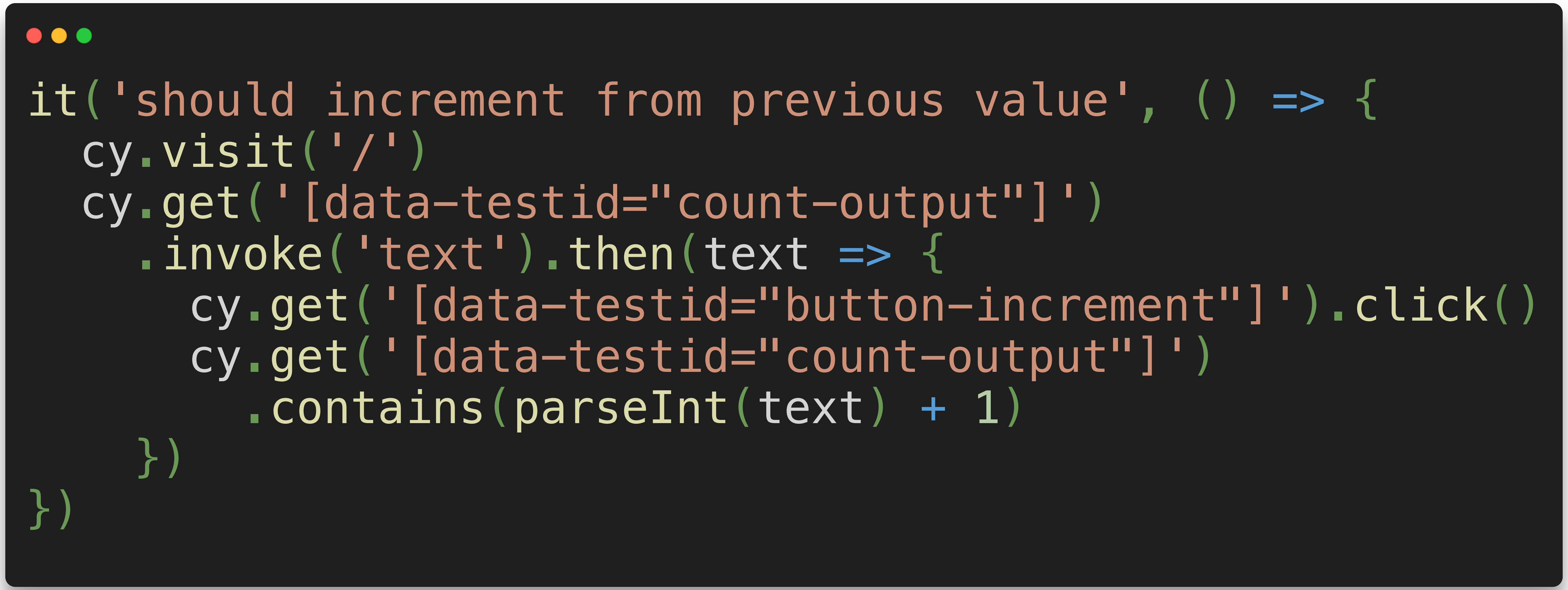
What to run

Cypress provides ways to decide, which tests to run.



```
it.only('should only run this', () => {})

it.skip('should skip this', () => {})
```

Using previous state



```
it('should increment from previous value', () => {
  cy.visit('/')
  cy.get('[data-testid="count-output"]')
    .invoke('text').then(text => {
      cy.get('[data-testid="button-increment"]').click()
      cy.get('[data-testid="count-output"]')
        .contains(parseInt(text) + 1)
    })
})
```

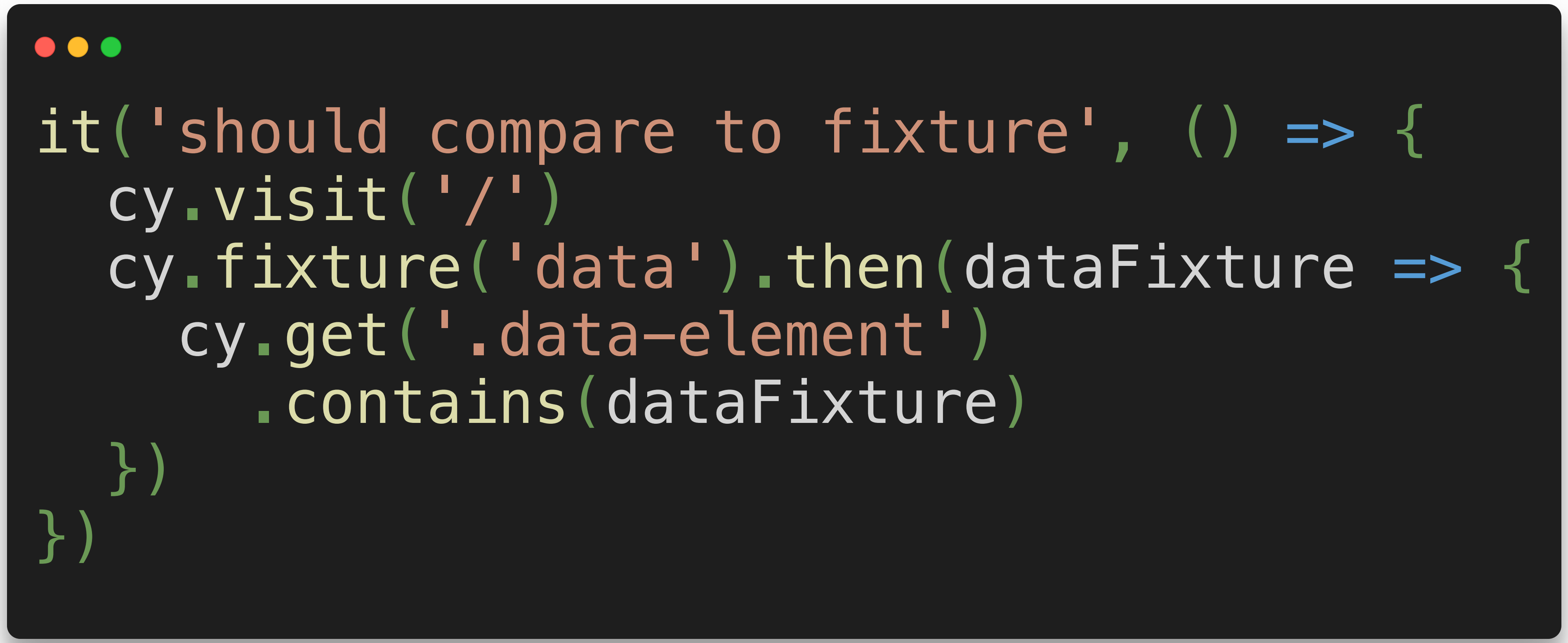
Cypress yields

Cypress yields values in a Promise like fashion.

You can **not** `await` Cypress commands

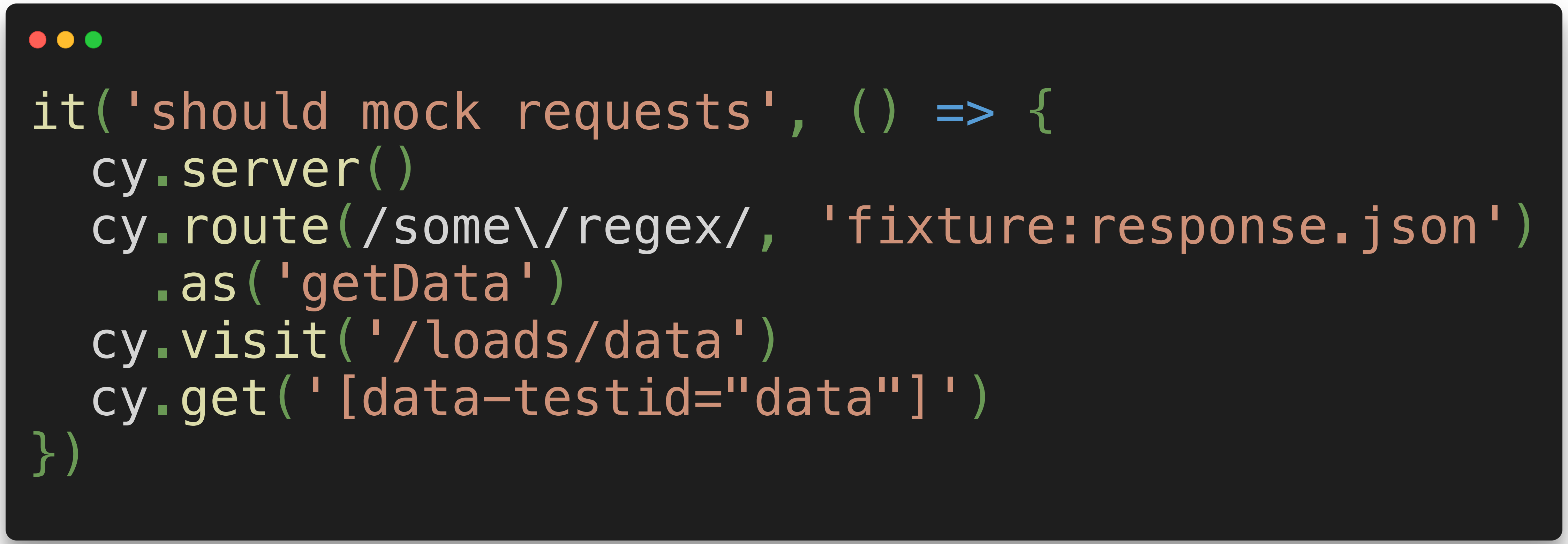
Thus we need to chain `.then()` should we want to use values from previous commands.

Fixtures



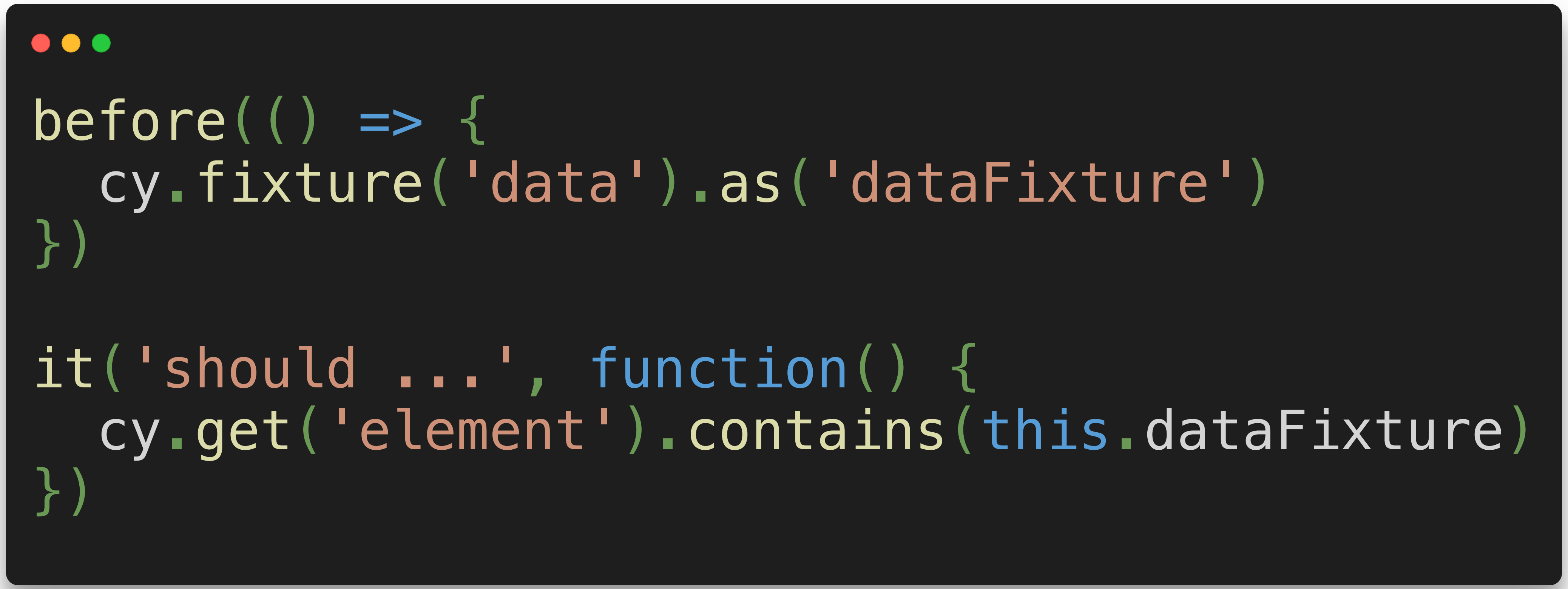
```
it('should compare to fixture', () => {
  cy.visit('/')
  cy.fixture('data').then(dataFixture => {
    cy.get('.data-element')
      .contains(dataFixture)
  })
})
```

Mocks and fixtures



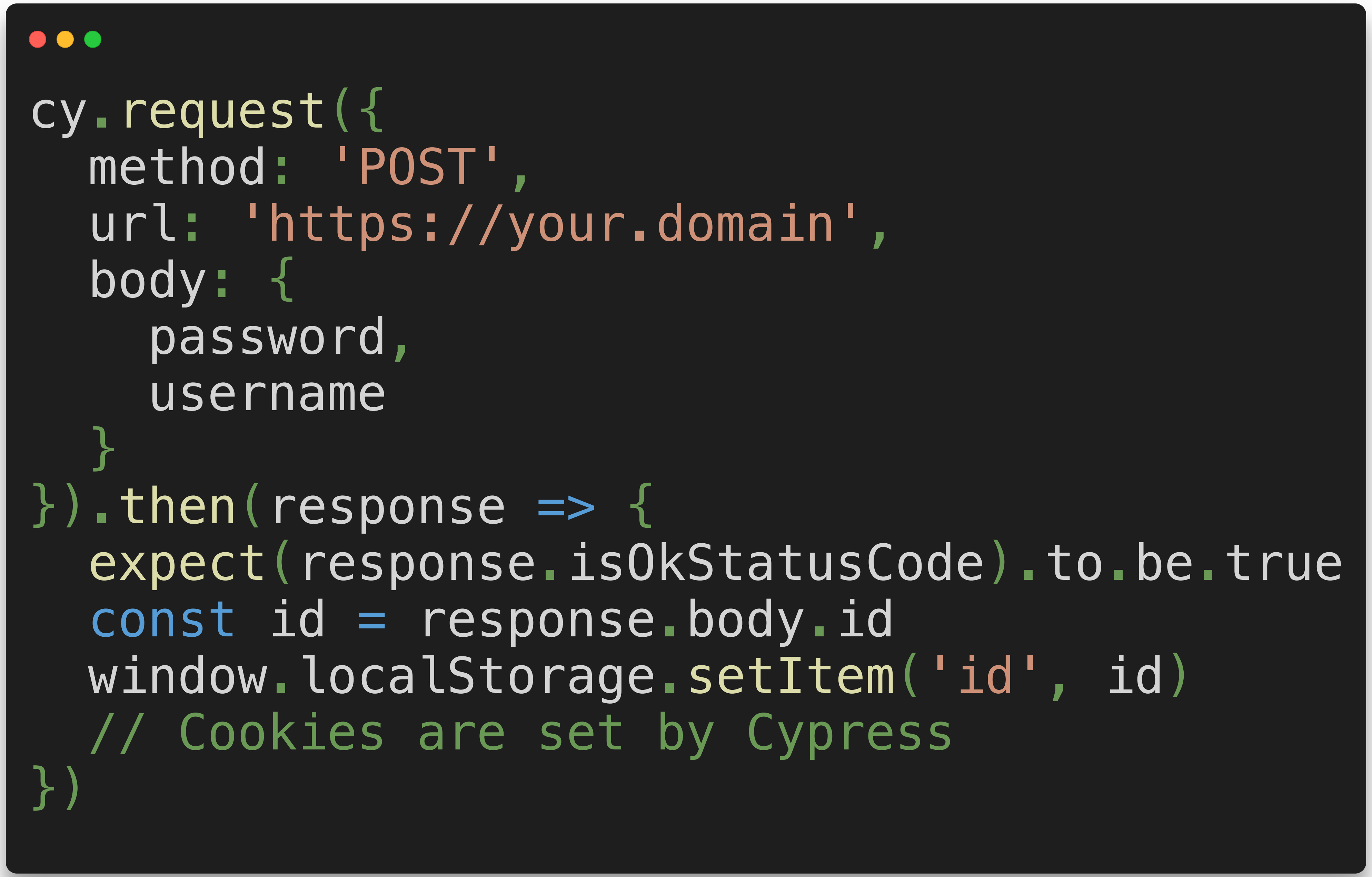
```
it('should mock requests', () => {
  cy.server()
  cy.route(/some\/regex/, 'fixture:response.json')
    .as('getData')
  cy.visit('/loads/data')
  cy.get('[data-testid="data"]')
})
```

Aliases



```
before(() => {
  cy.fixture('data').as('dataFixture')
})

it('should ...', function() {
  cy.get('element').contains(this.dataFixture)
})
```

Do it programmatically


```
cy.request({
  method: 'POST',
  url: 'https://your.domain',
  body: {
    password,
    username
  }
}).then(response => {
  expect(response.isOkStatusCode).to.be.true
  const id = response.body.id
  window.localStorage.setItem('id', id)
  // Cookies are set by Cypress
})
```

Custom Commands

```
Cypress.Commands.add('login', (username, password) => {
  // Login programmatically
})

// some.spec.js
beforeEach(() => {
  cy.login('username', 'password')
})
```

Best Practices

Consider what to e2e-test.

Expose APIs from your application.

`.get()` does not need positive assertions.

Speed up your tests by logging in programmatically.

More best practices

Resources

- Cypress Guides
- Cypress API
- Examples
- Repo for this presentation

🏁 THE END 🏁