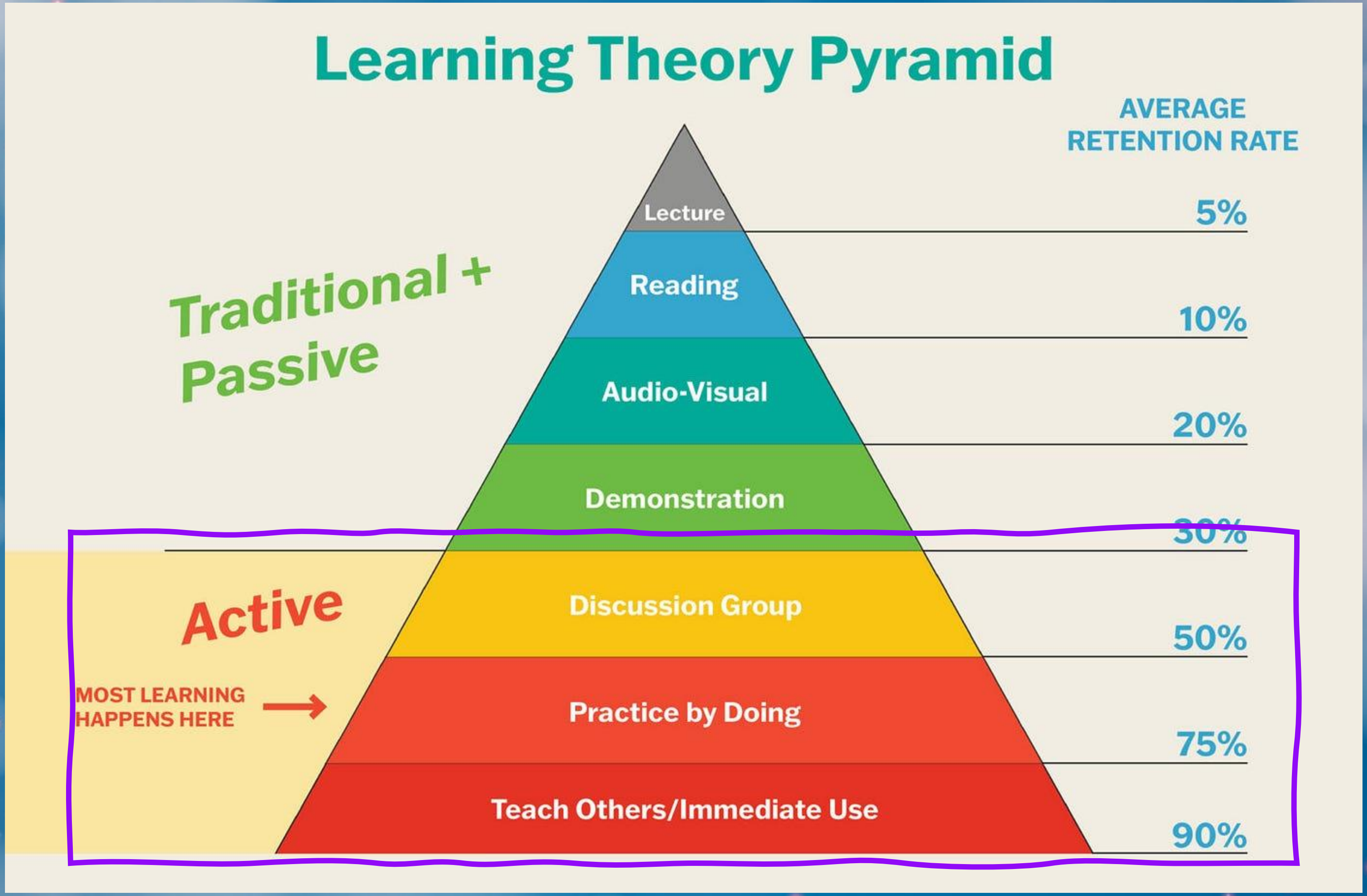
From zero to hero

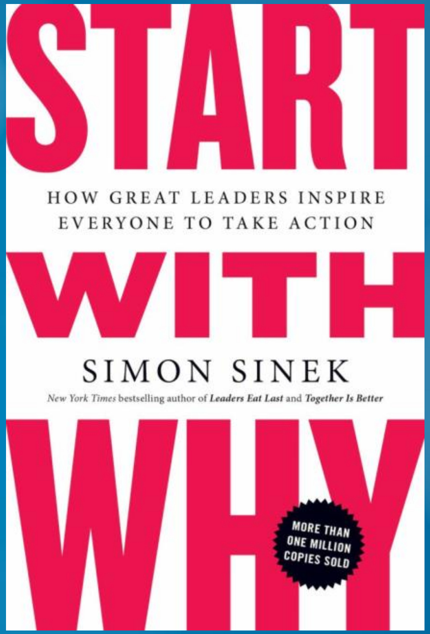
cypress

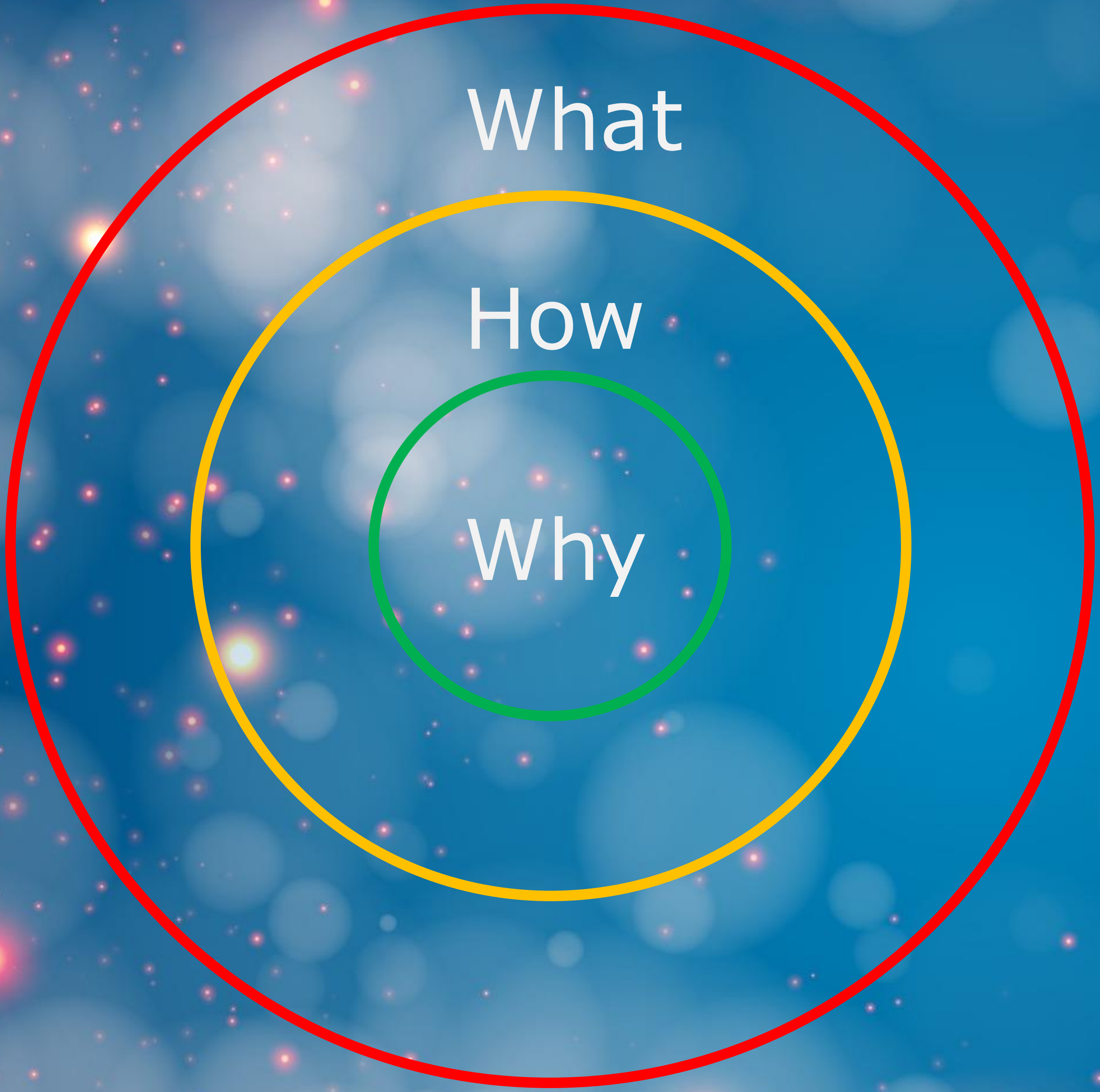
TESTCODERS

What to expect ?

(when you're expecting..)




START
HOW GREAT LEADERS INSPIRE EVERYONE TO TAKE ACTION
WITH
SIMON SINEK
New York Times bestselling author of Leaders Eat Last and Together Is Better
WHY
MORE THAN ONE MILLION COPIES SOLD


What
How
Why

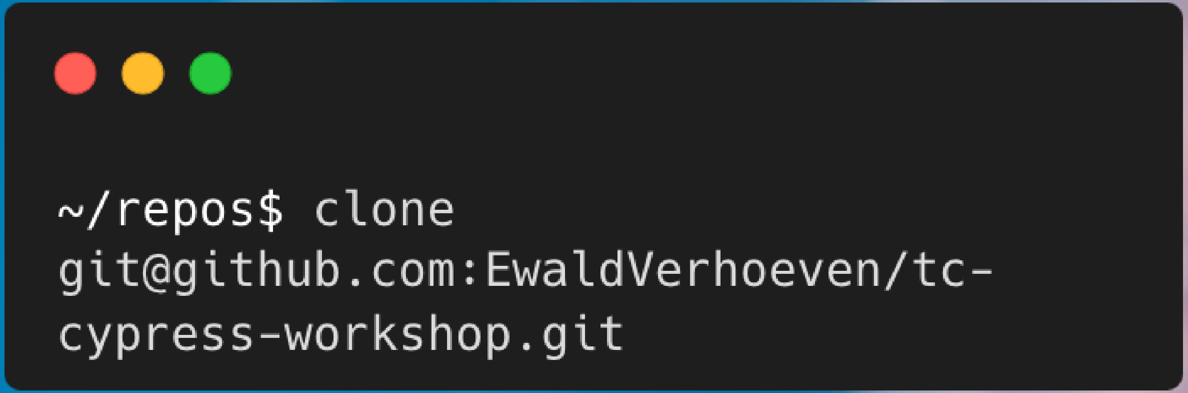
alamy

Preparations

1. Technical requirements:
   a. Node (18.X --> LTS)
   b. Git --> *optional*
2. SUT :
   a. https://github.com/marcelblijleven/testrpg (source code)
   b. https://test-rpg.vercel.app/

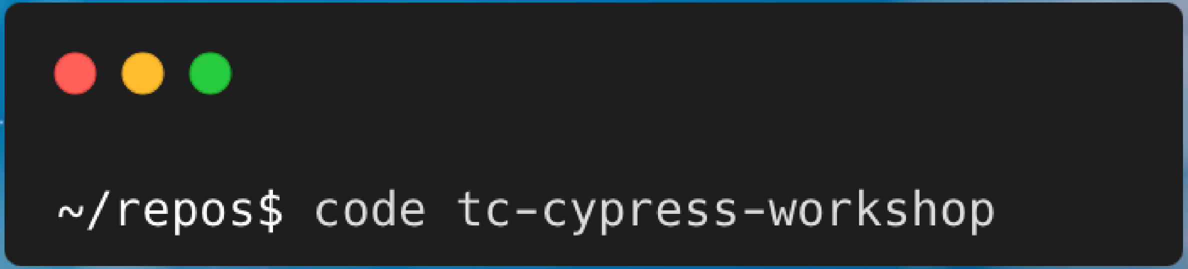
#0 Setup local cypress project (1)

1. Clone skeleton project from:
   github.com/EwaldVerhoeven/tc-cypress-workshop



```
~/repos$ clone
git@github.com:EwaldVerhoeven/tc-
cypress-workshop.git
```

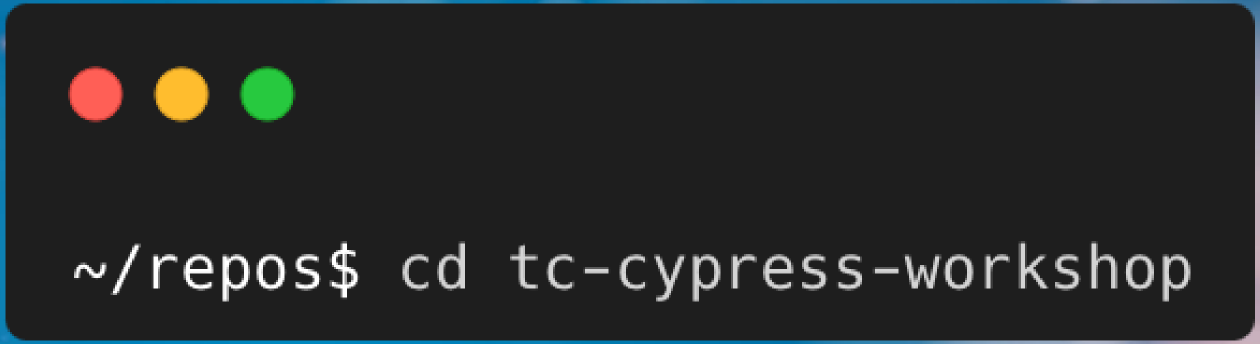
2. Open the project in VSCode (or any IDE)



```
~/repos$ code tc-cypress-workshop
```

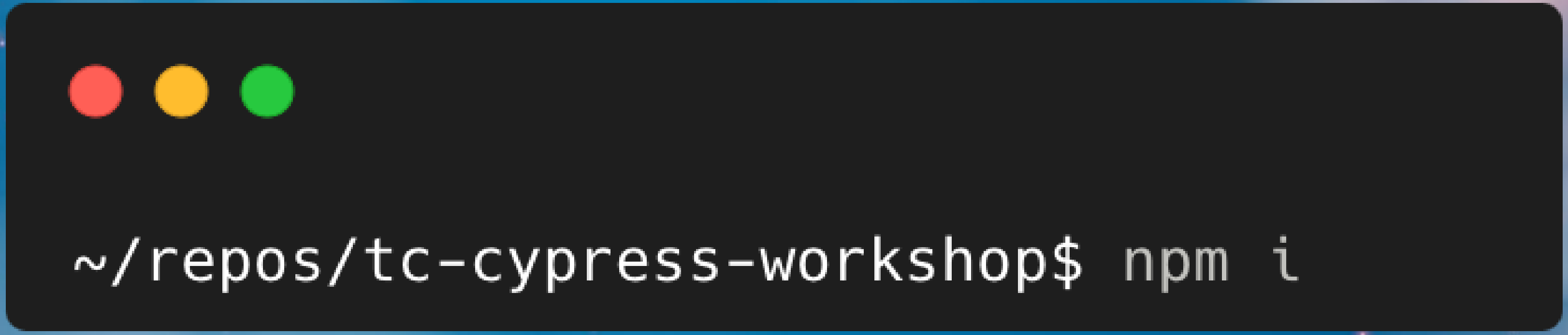
#0 Setup local cypress project (2)

1. Move into the project (root) folder

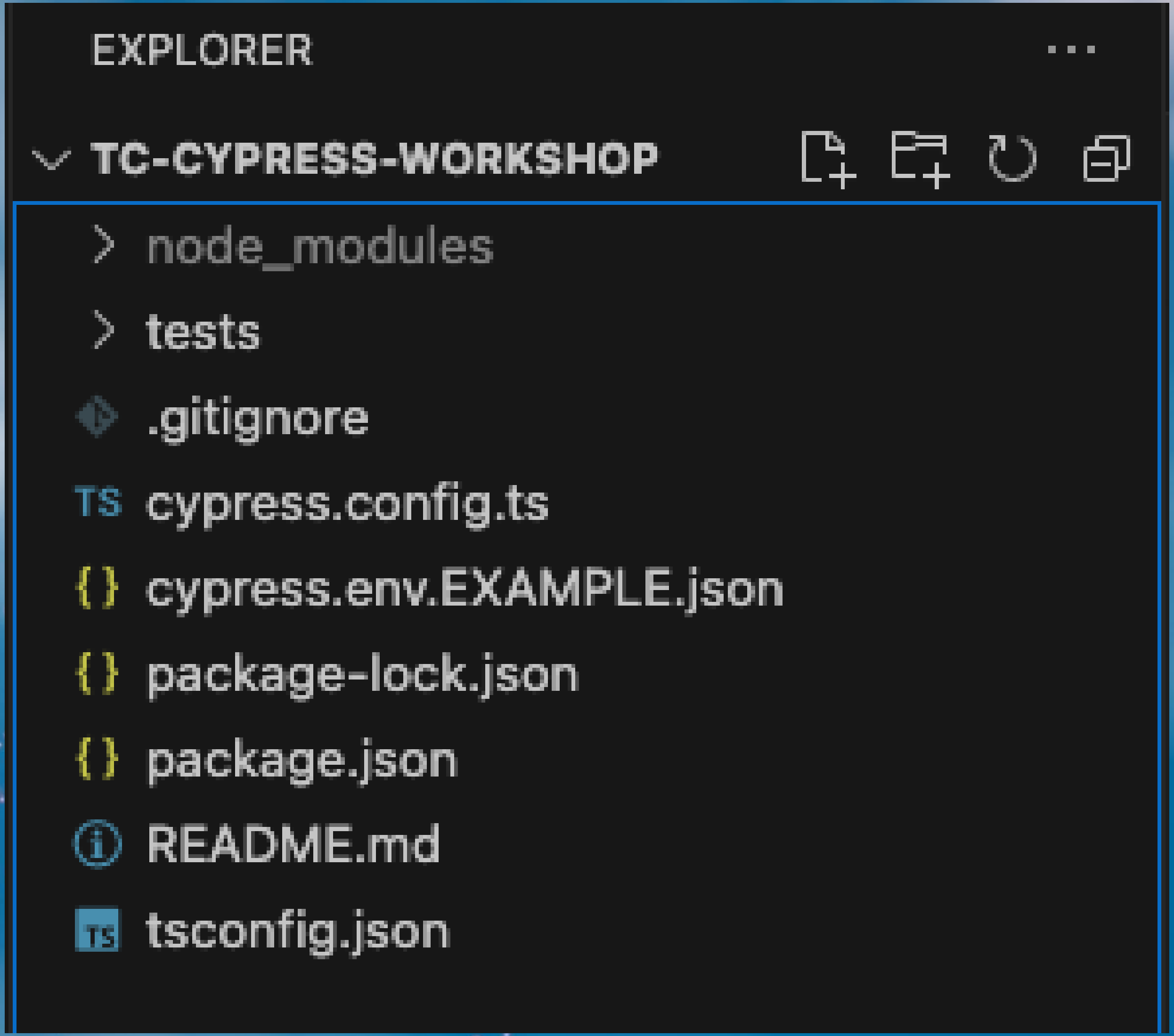


```
~/repos$ cd tc-cypress-workshop
```

2. Install dependencies



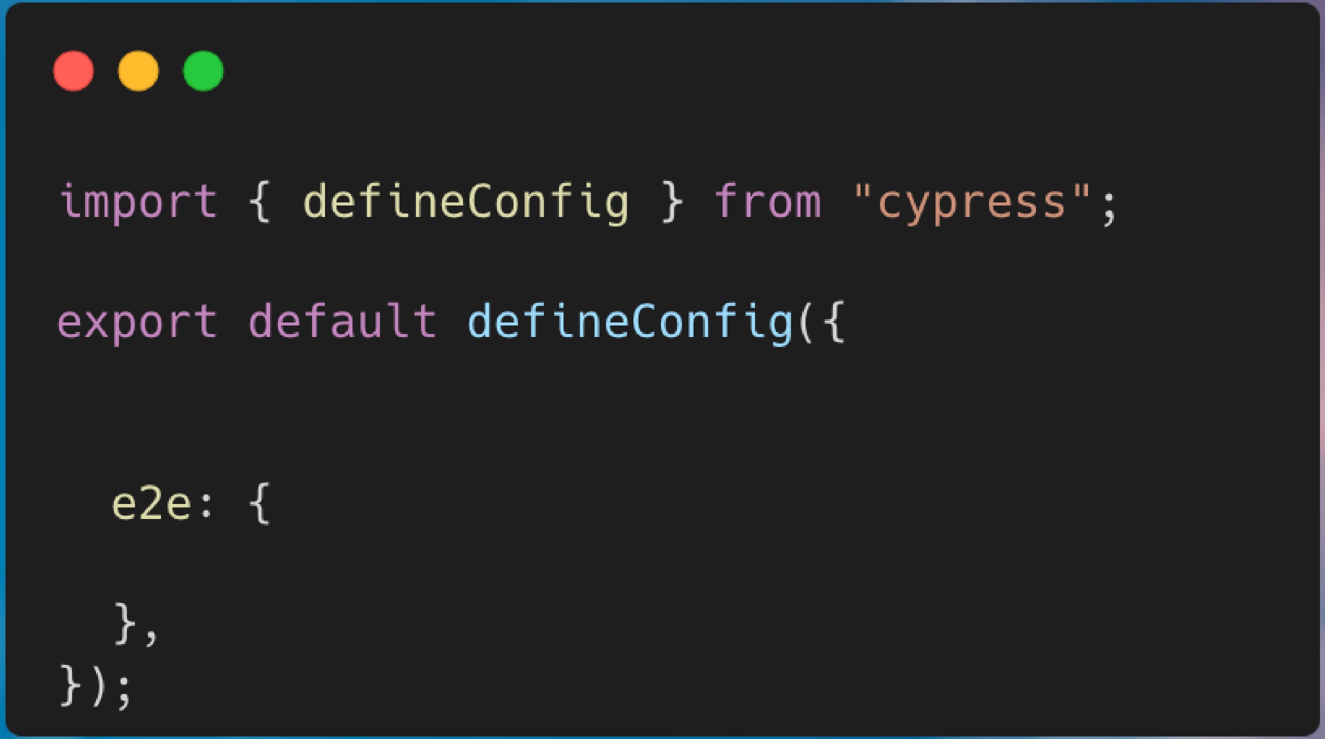
```
~/repos/tc-cypress-workshop$ npm i
```



```
> node_modules
> tests
.gitignore
cypress.config.ts
cypress.env.EXAMPLE.json
package-lock.json
package.json
README.md
tsconfig.json
```

#0 Setup local cypress project (3)

1. Add 'E2E' object to config

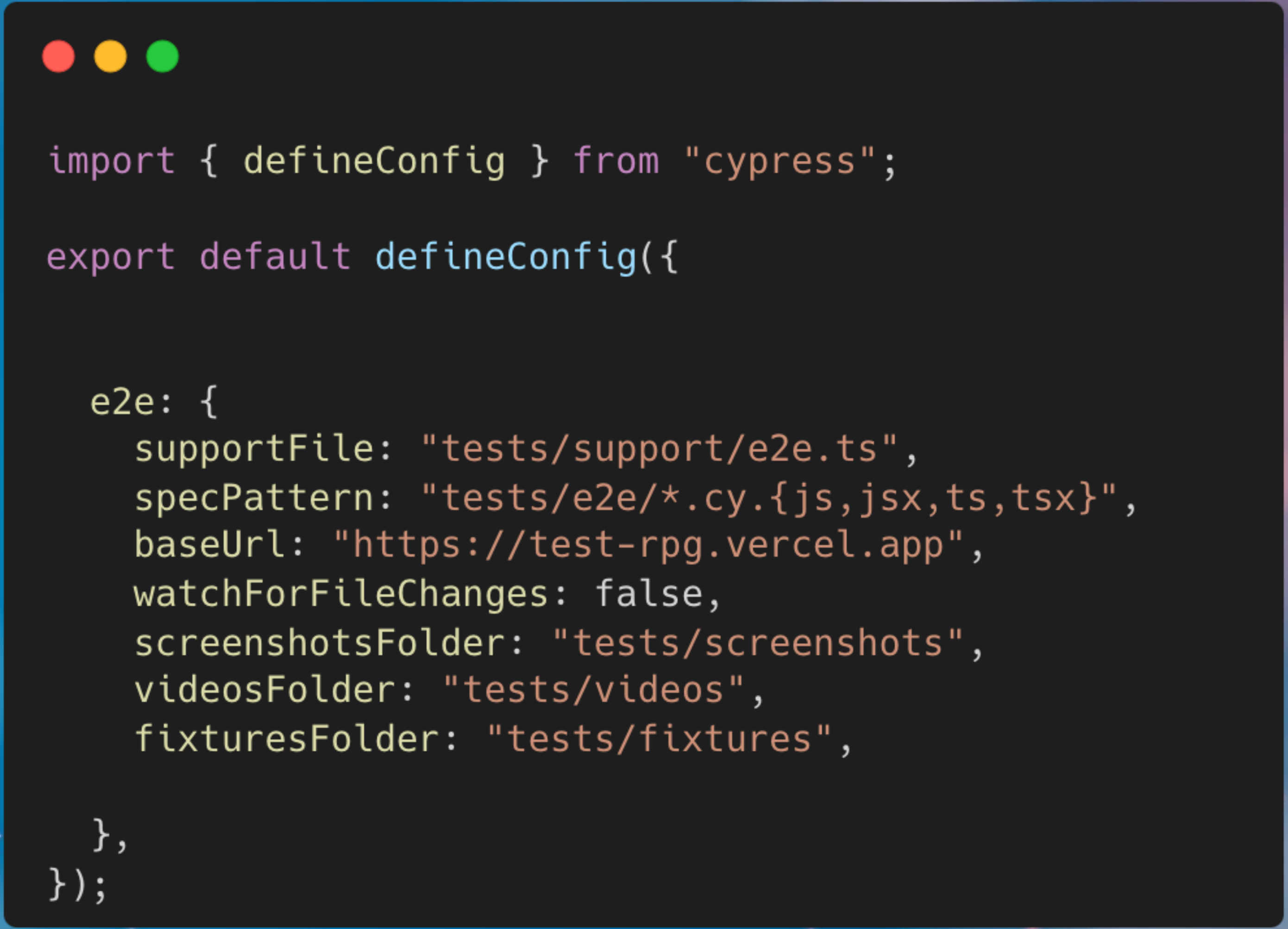


```
import { defineConfig } from "cypress";

export default defineConfig({


  e2e: {

  },
});
```

#0 Setup local cypress project (4)

1. Add the following 7 settings:

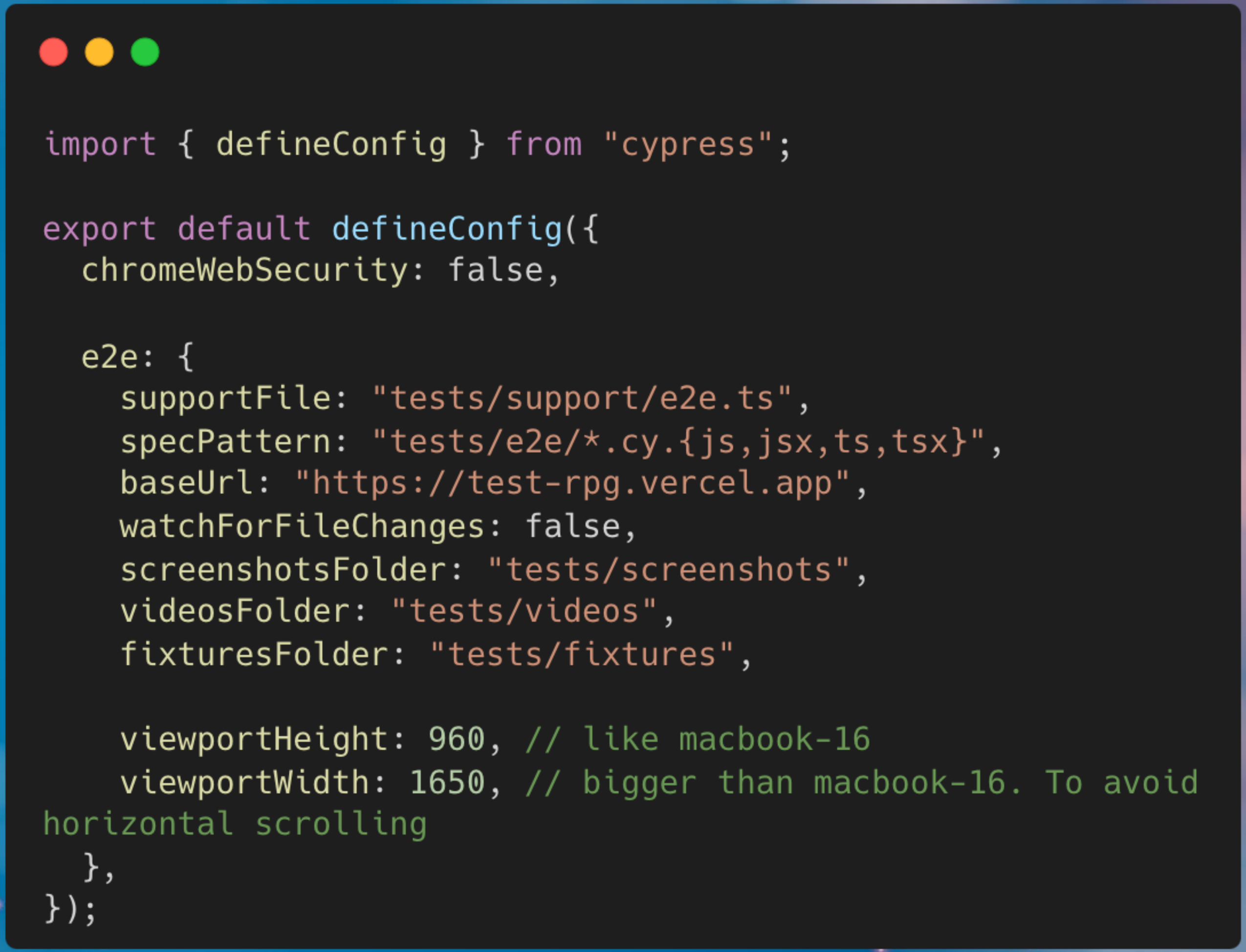
   a. "suppportFile"
   b. "specPattern"
   c. "fixturesFolder"
   d. "baseUrl"
   e. "watchForFileChanges"
   f. "screenshotsFolder"
   g. "videosFolder"



```
import { defineConfig } from "cypress";

export default defineConfig({


  e2e: {
    supportFile: "tests/support/e2e.ts",
    specPattern: "tests/e2e/*.cy.{js,jsx,ts,tsx}",
    baseUrl: "https://test-rpg.vercel.app",
    watchForFileChanges: false,
    screenshotsFolder: "tests/screenshots",
    videosFolder: "tests/videos",
    fixturesFolder: "tests/fixtures",

  },
});
```

#0 Setup local cypress project (5)

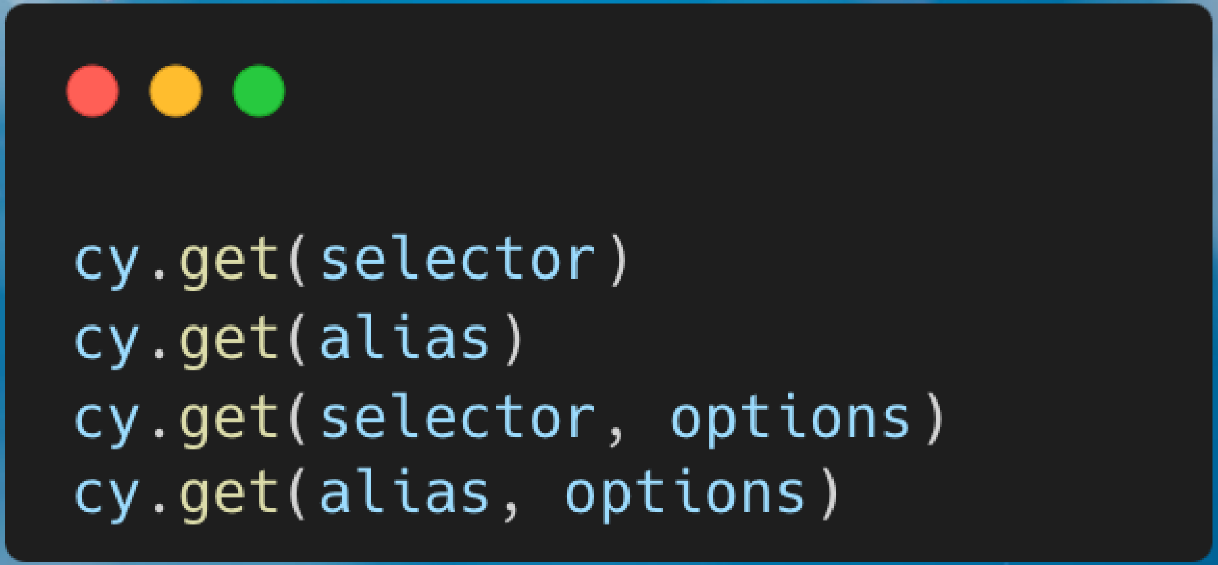
1. Add "chromeWebSecurity" and
   set value to 'false'
2. Add the
   following (e2e)settings:
   A. "viewportHeight"
   B. "viewportWidth"



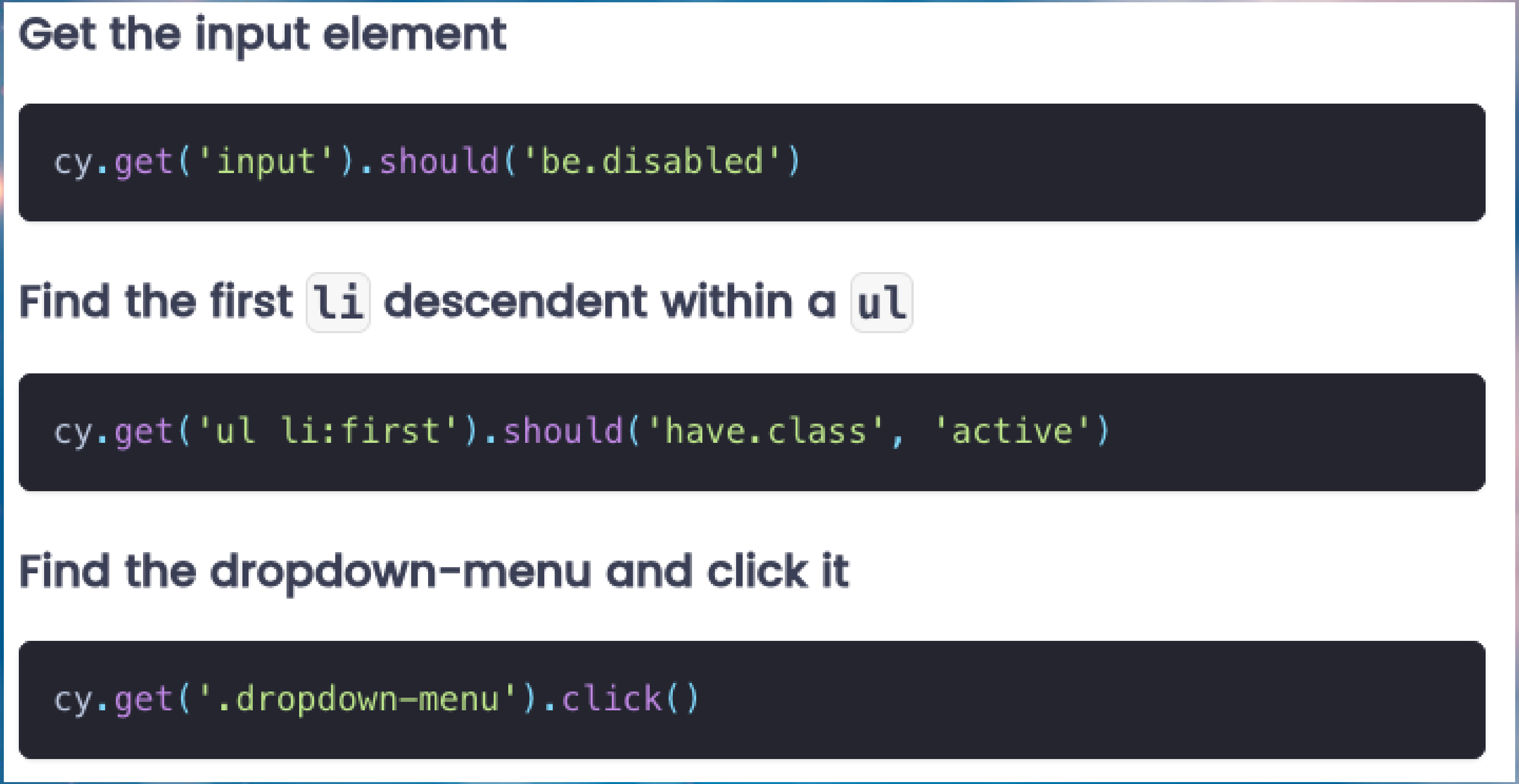
```
import { defineConfig } from "cypress";

export default defineConfig({
  chromeWebSecurity: false,

  e2e: {
    supportFile: "tests/support/e2e.ts",
    specPattern: "tests/e2e/*.cy.{js,jsx,ts,tsx}",
    baseUrl: "https://test-rpg.vercel.app",
    watchForFileChanges: false,
    screenshotsFolder: "tests/screenshots",
    videosFolder: "tests/videos",
    fixturesFolder: "tests/fixtures",

    viewportHeight: 960, // like macbook-16
    viewportWidth: 1650, // bigger than macbook-16. To avoid
horizontal scrolling
  },
});
```

Locators and selectors

- Cypress tests the DOM
- Locators in Cypres



```
cy.get(selector)
cy.get(alias)
cy.get(selector, options)
cy.get(alias, options)
```


Get the input element

```
cy.get('input').should('be.disabled')
```

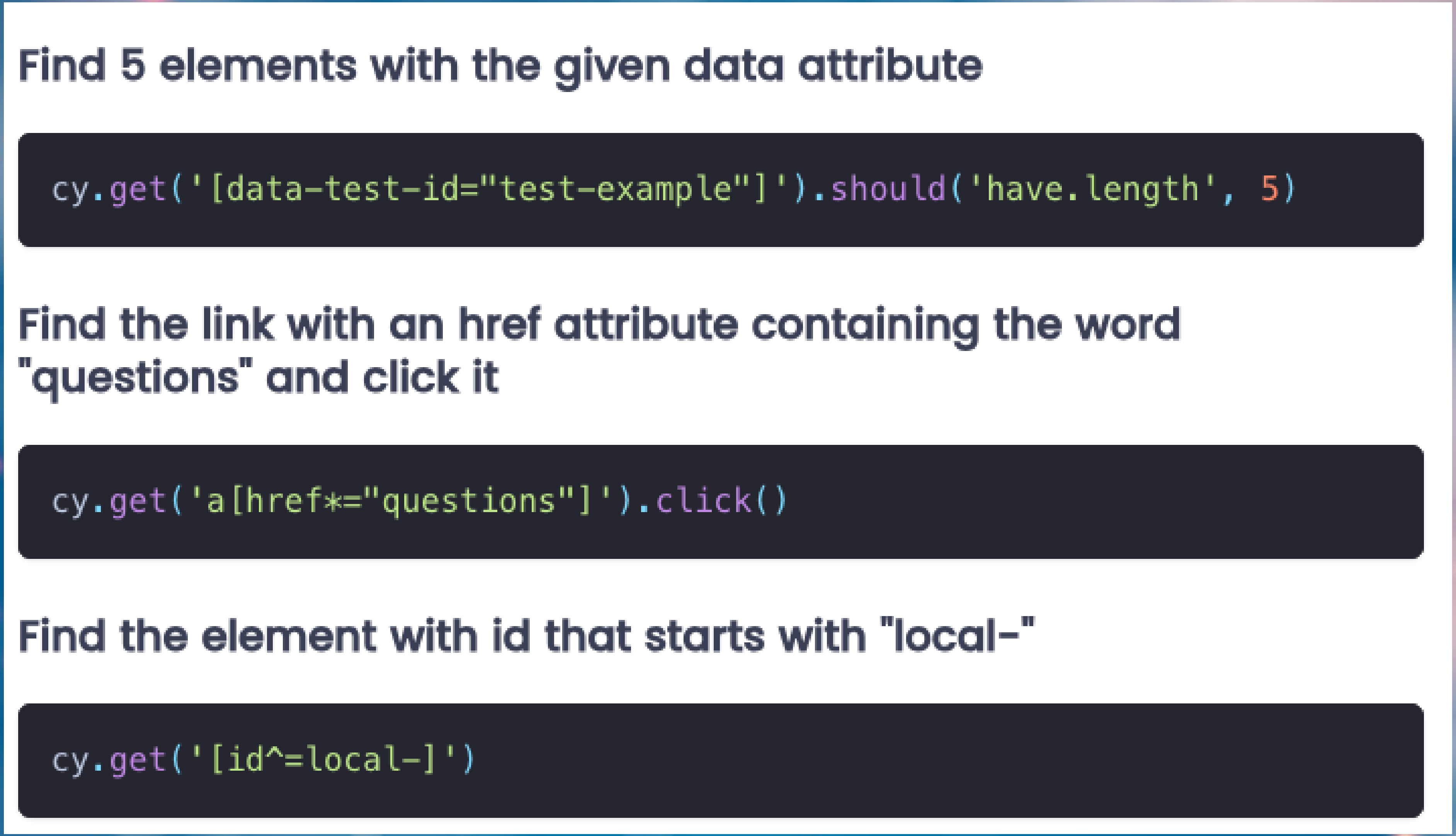
Find the first `li` descendent within a `ul`

```
cy.get('ul li:first').should('have.class', 'active')
```

Find the dropdown-menu and click it

```
cy.get('.dropdown-menu').click()
```

source: *https://docs.cypress.io/api/commands/get*


Find 5 elements with the given data attribute

```
cy.get('[data-test-id="test-example"]').should('have.length', 5)
```

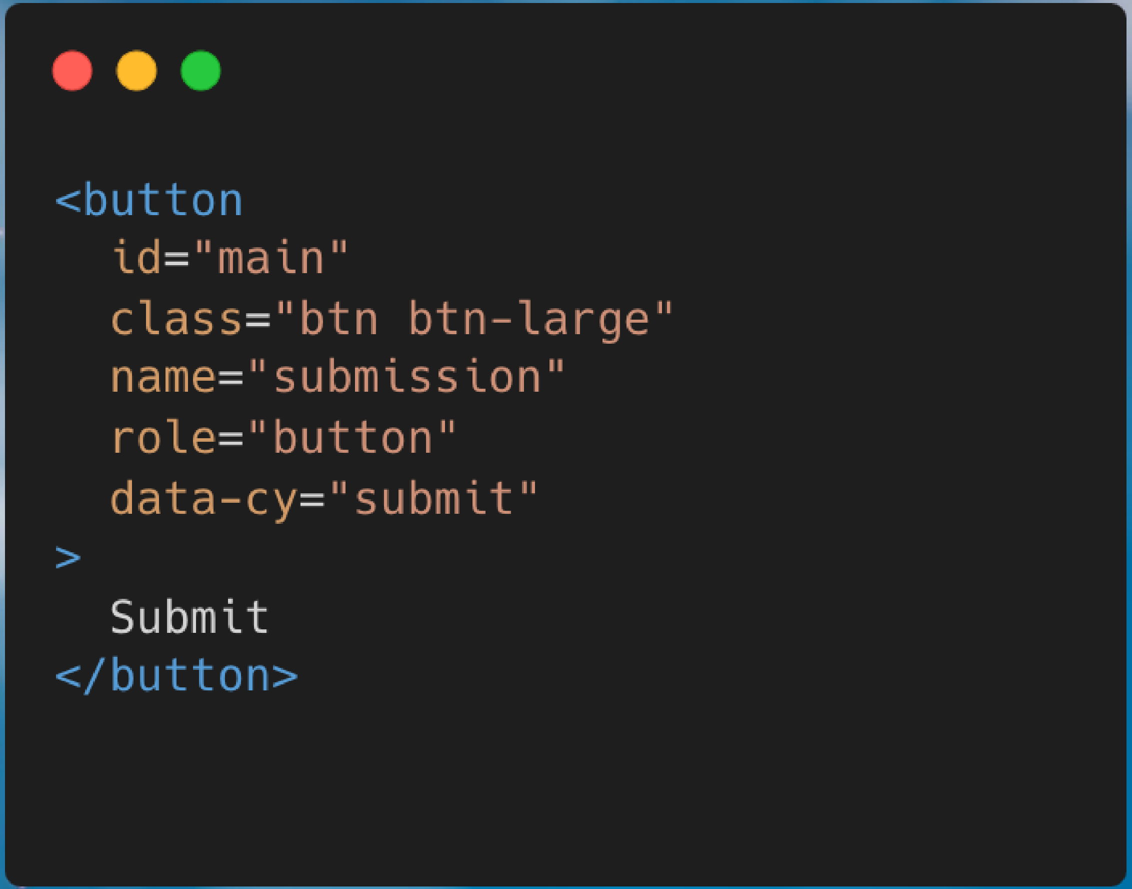
Find the link with an href attribute containing the word "questions" and click it

```
cy.get('a[href*="questions"]').click()
```

Find the element with id that starts with "local-"

```
cy.get('[id^=local-]')
```

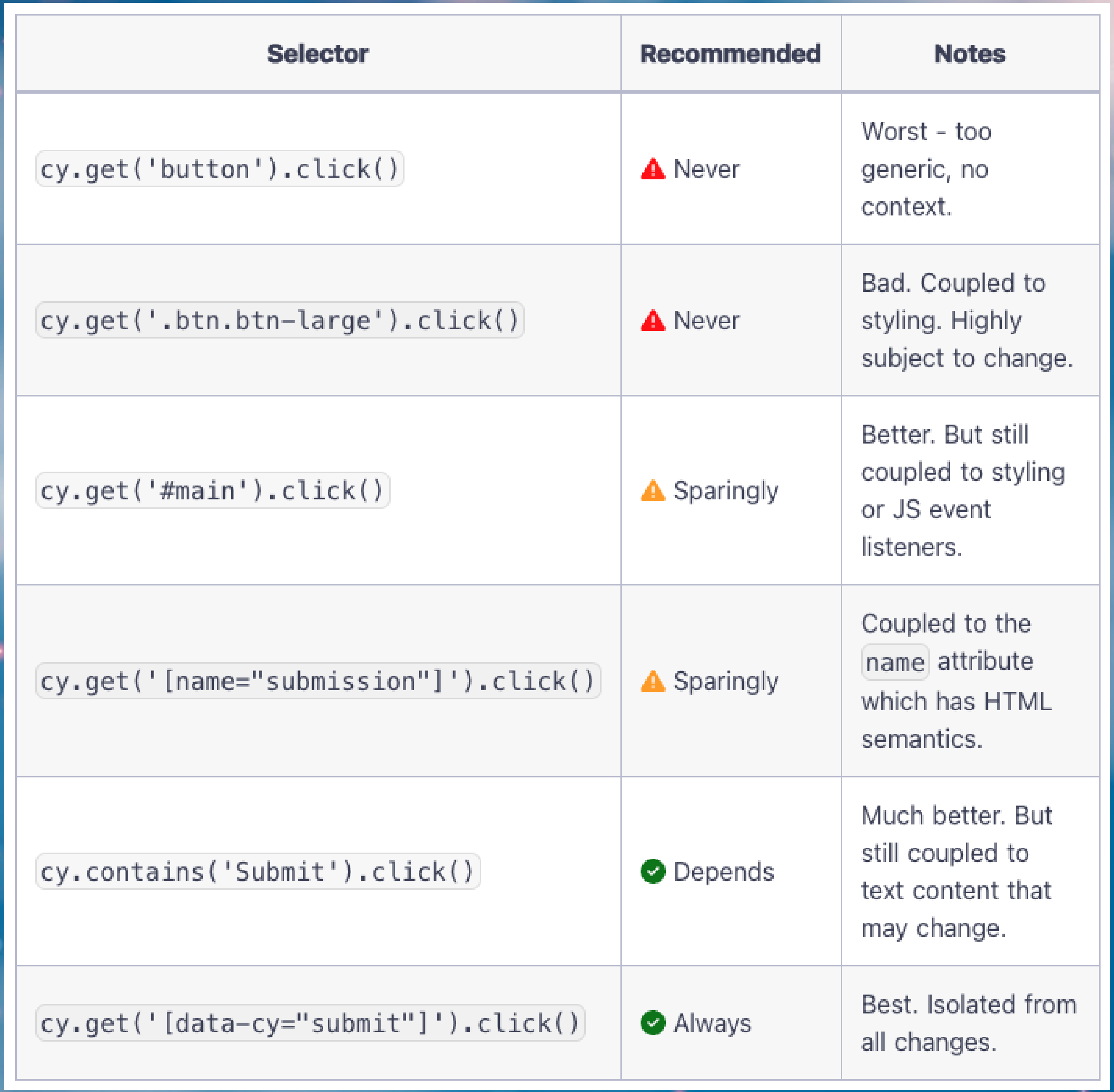
source: *https://docs.cypress.io/api/commands/get*



```
<button
  id="main"
  class="btn btn-large"
  name="submission"
  role="button"
  data-cy="submit"
>
  Submit
</button>
```

(Warmup) excersize:

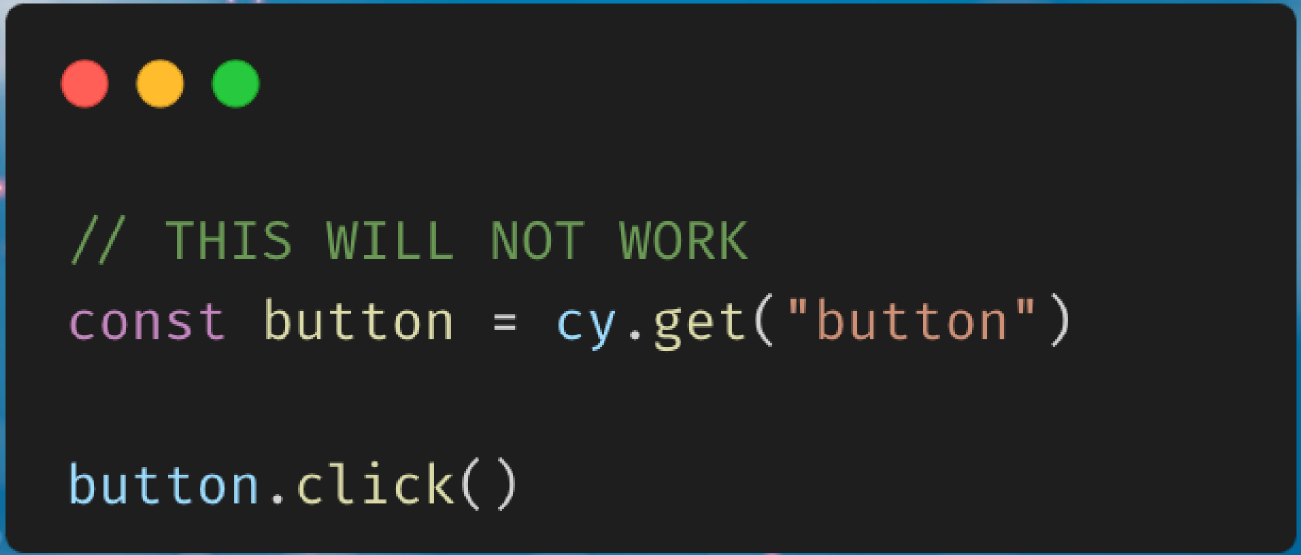
Write a selector query to yield the <button> element above



Selector	Recommended	Notes
`cy.get('button').click()`	⚠ Never	Worst - too generic, no context.
`cy.get('.btn.btn-large').click()`	⚠ Never	Bad. Coupled to styling. Highly subject to change.
`cy.get('#main').click()`	⚠ Sparingly	Better. But still coupled to styling or JS event listeners.
`cy.get('[name="submission"]').click()`	⚠ Sparingly	Coupled to the `name` attribute which has HTML semantics.
`cy.contains('Submit').click()`	✅ Depends	Much better. But still coupled to text content that may change.
`cy.get('[data-cy="submit"]').click()`	✅ Always	Best. Isolated from all changes.

Return vs Yield

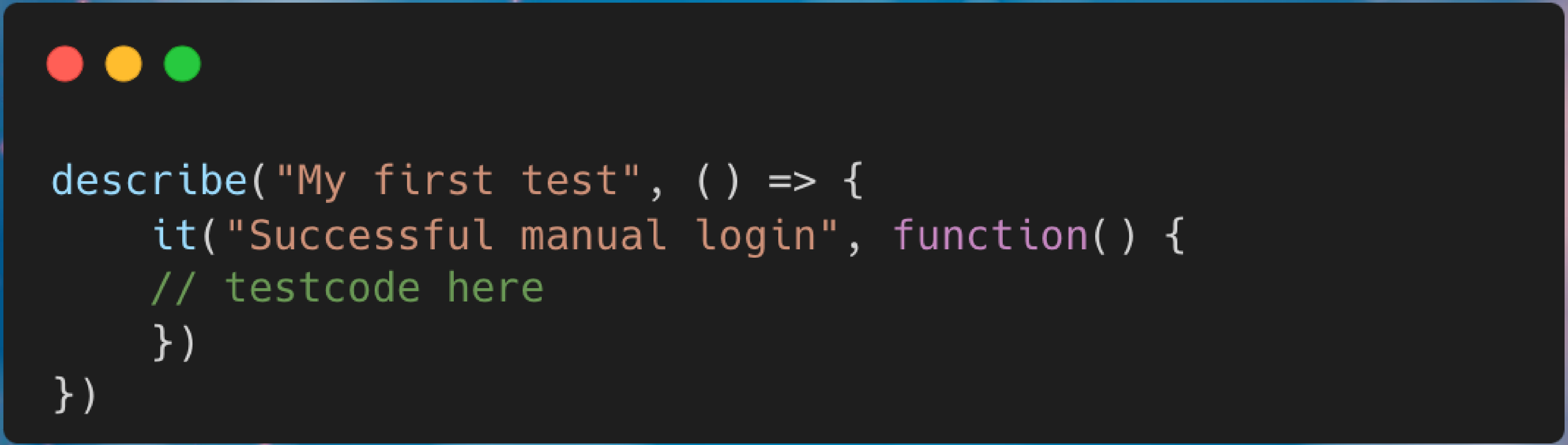
Cypress commands YIELD (not RETURN!!) subjects



```
// THIS WILL NOT WORK
const button = cy.get("button")

button.click()
```

#1 Write your first test (1)

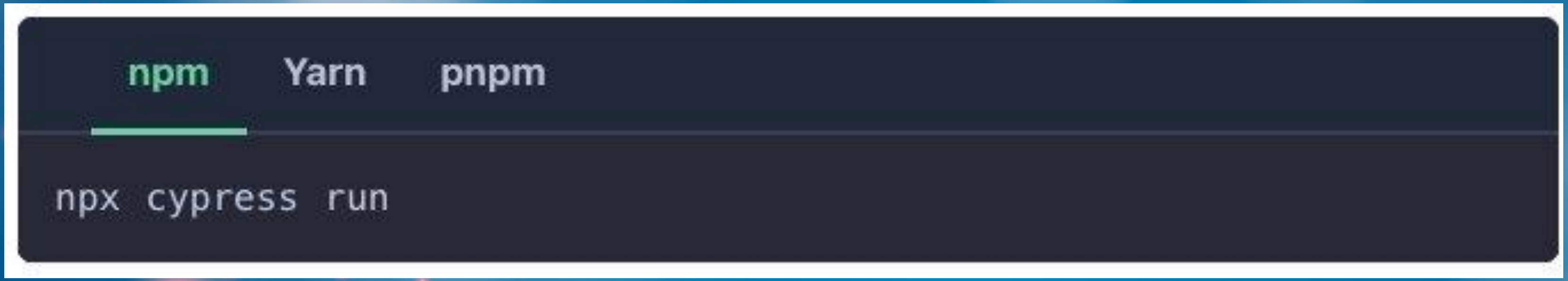
- **Add**: (test)file to your e2e folder called *rpgManualLogin.cy.ts*



```
describe("My first test", () => {
    it("Successful manual login", function() {
    // testcode here
    })
})
```

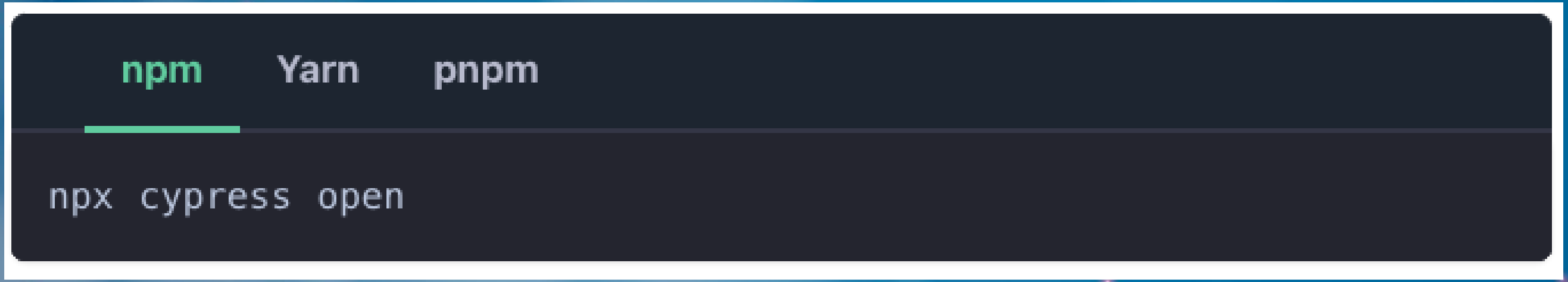
Running your tests

- *Command line (HEADLESS)*



```
npx cypress run
```

- *Cypress test runner (HEADED) --> DEMO!*



```
npx cypress open
```

#1 Write your first test (2)

- **Automate** the stepsbelow

*Hint: api's you'll need are **cy.visit()** / **cy.get('')** / **.click()** / **.type()***

Step	actions
1	Click on the 'login' button
2	Enter a valid email
3	Enter a password
4	Press 'enter' key OR Click on 'login' button in modal

#1 Write your first test (3)

- **Add**: *assertions* (according to 'results' column below) for **step 1 and 4**

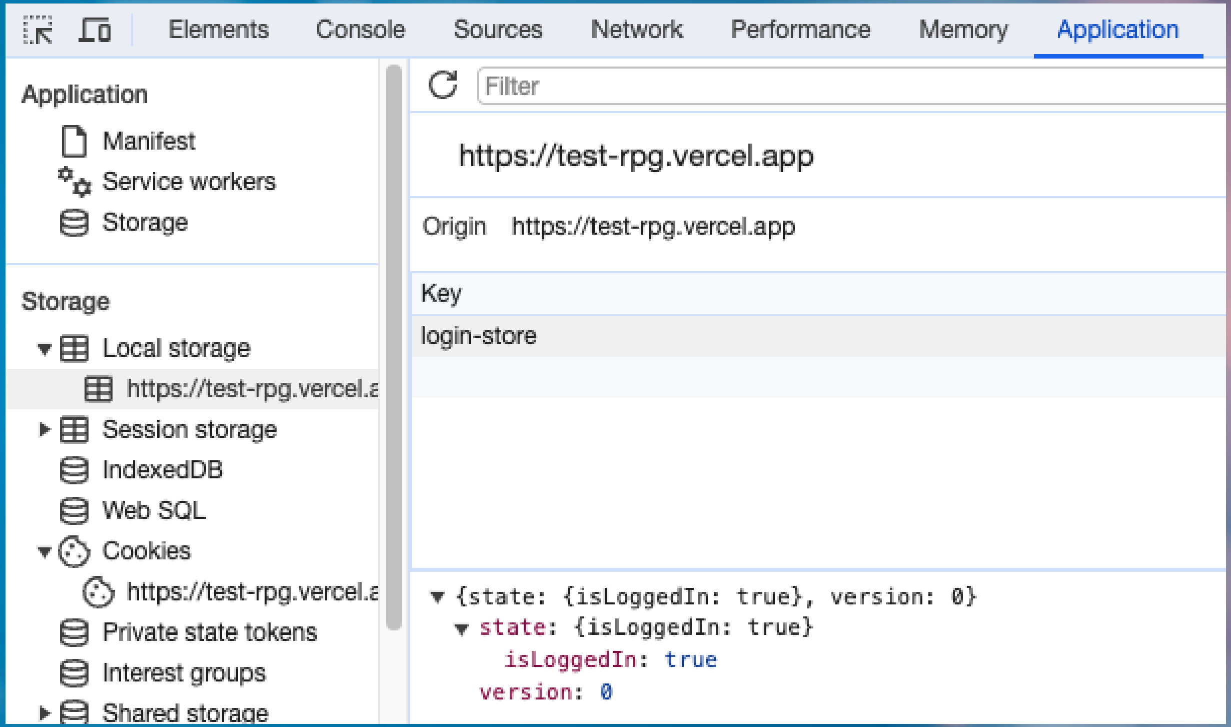
Step	actions	result
1	Click on the 'login' button	Modal with form appears
2	Enter a valid email	No validation messages
3	Enter a password	
4	Press 'enter' key OR Click on 'login' button in modal	Modal exits and button (step 1) now displays 'Logout'

Hint: api you'll need **.should()**

#1 Write your first test (4)

- **Add:** Assertions to check that you are (also technically) logged in

Hint:
What is in your browsers ***local storage?***



https://test-rpg.vercel.app

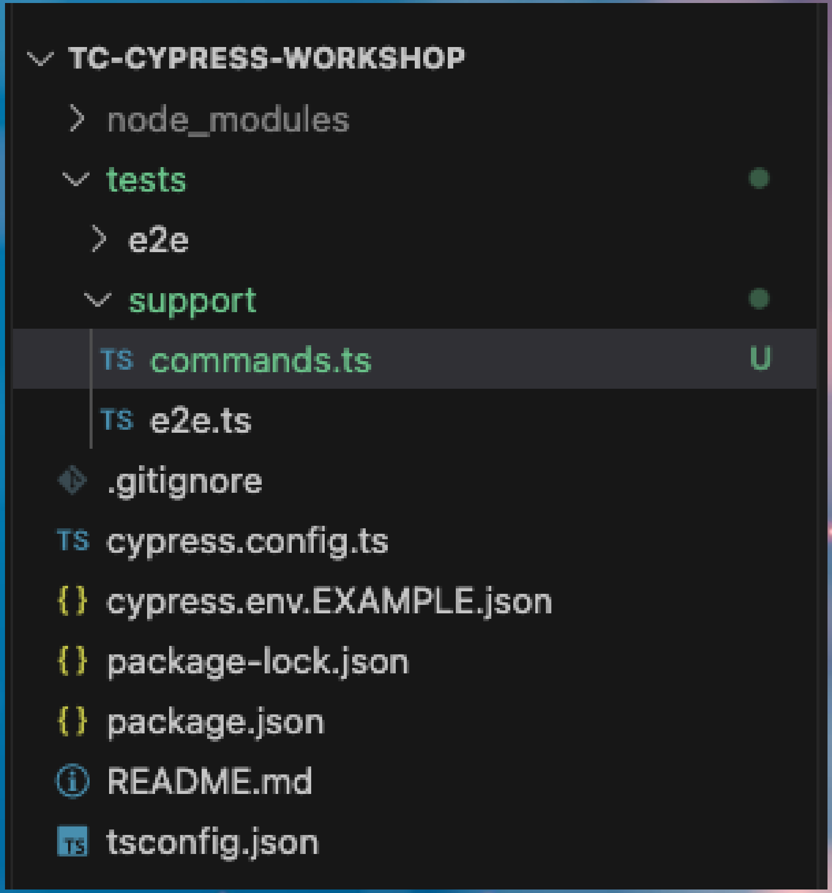
Origin https://test-rpg.vercel.app

Key
login-store

```
▼ {state: {isLoggedIn: true}, version: 0}
  ▼ state: {isLoggedIn: true}
      isLoggedIn: true
    version: 0
```

#2 Custom commands (1)

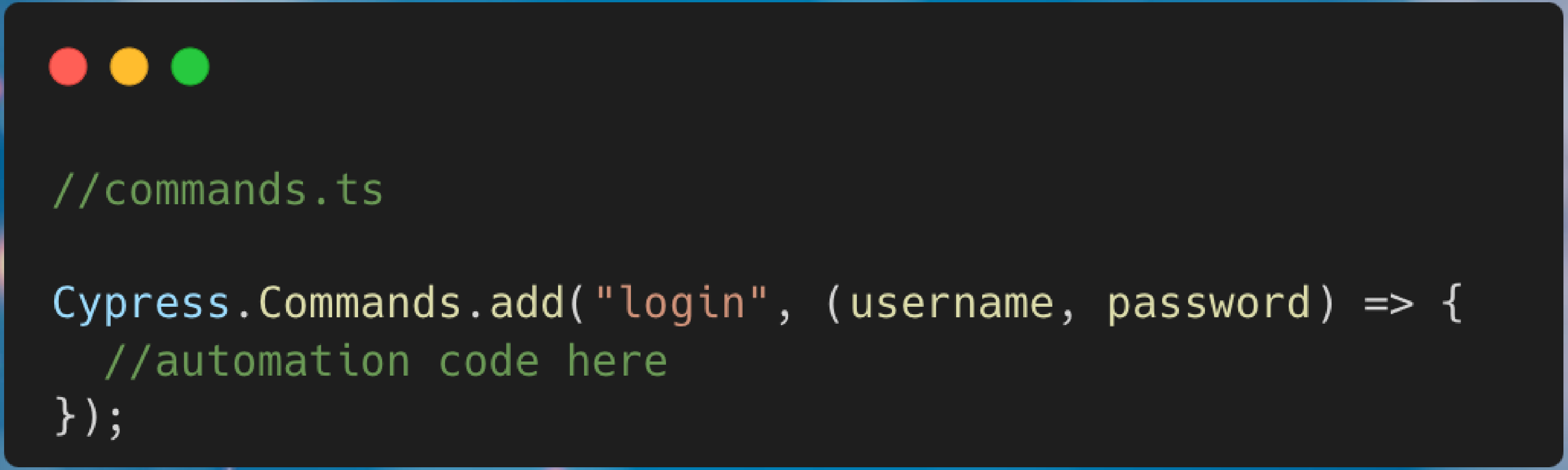
- **Add** a file called 'commands.ts' in *./tests/support*



```
TC-CYPRESS-WORKSHOP
  node_modules
  tests
    e2e
    support
      commands.ts          U
      e2e.ts
  .gitignore
  cypress.config.ts
  cypress.env.EXAMPLE.json
  package-lock.json
  package.json
  README.md
  tsconfig.json
```

#2 Custom commands (2)

- **Write** a **custom command** named '**login**' to automate/re-use the 'manual' login procedure (as done in previous exercise).\

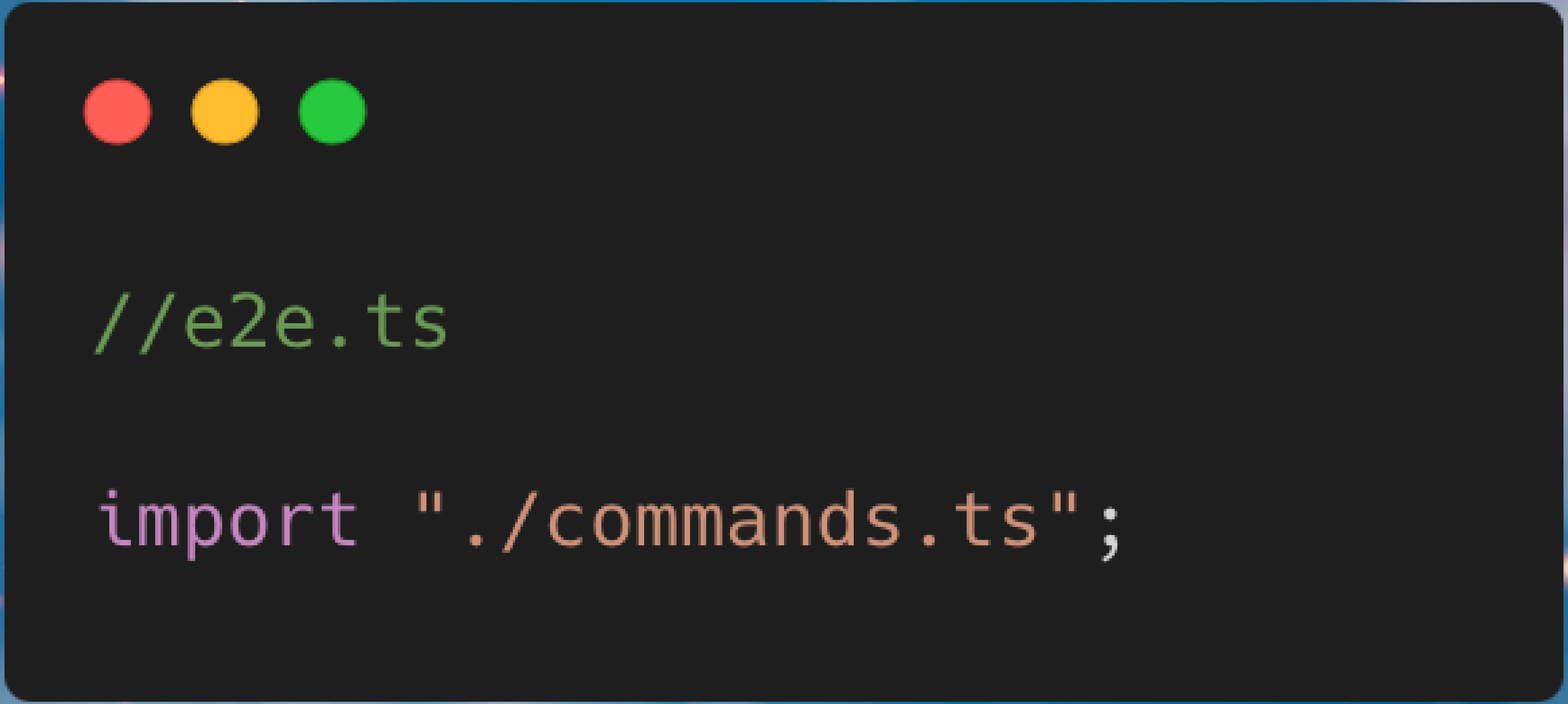


```
//commands.ts

Cypress.Commands.add("login", (username, password) => {
  //automation code here
});
```

- **Add** your custom command to the Cypress namespace and import the commands module in your support file

#2 Custom commands (3)

- **Add** your custom command to the Cypress namespace (1) and import the commands module in your support file (2)



```
//e2e.ts

import "./commands.ts";
```

#2 Custom commands (4)

- **Refactor** your test (*rpgManualLogin.cy.ts)* by using your 'login' custom command --> cy.login(usr, pwd)

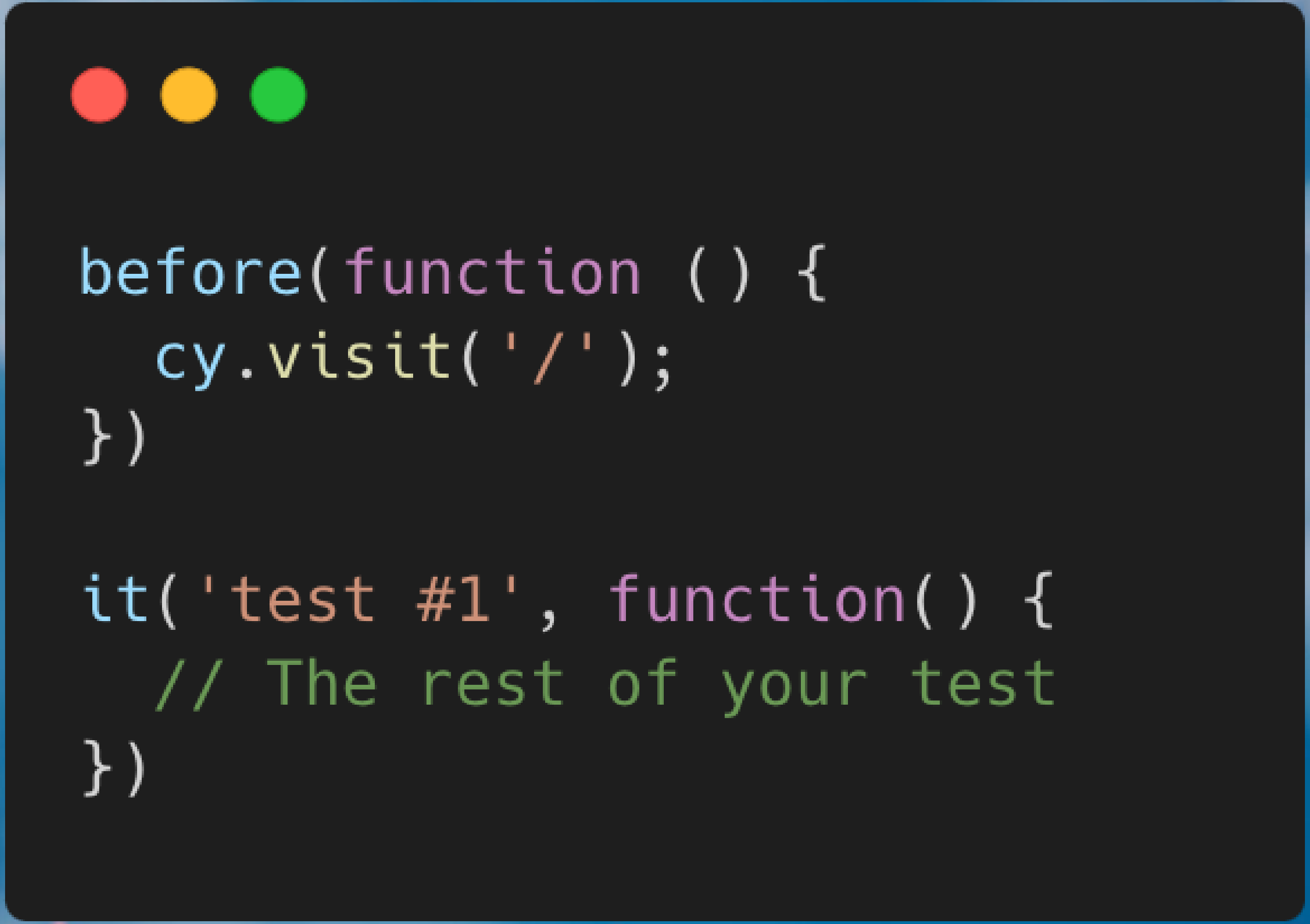
- **BONUS**: Make use of the .session() api in your 'login' command. This allows you restore session data

#3 Hooks and fixtures

1. Automate the following script in a new test file called '*playTheGame.cy.ts*'

Step	Actions	Result
0	Navigate to baseUrl & Login	Browser at baseUrl
1	Click on <button> with text: "Click here to play"	Browser at 'play'
2	Enter Character name	
3	Select option from "Build"	Desired option is selected
4	Click on <button> with text: "Start!"	
5	Click (5 times) on the <button> with tekst: " Click me … times"	Blue text appears that says 'Great job! You levelled up'
6	Upload a file by clicking 'choose file'	Blue text appears that says 'File selected, level up!'
7	Enter 'Lorem Ipsum' in the 'Type it' field	Blue text appears that says 'Dolar sit amet'
8	Slide the slider all the way to the right	Blue text appears that says 'Slid to the next level!'

Hooks syntax


```
before(function () {
  cy.visit('/');
})

it('test #1', function() {
  // The rest of your test
})
```

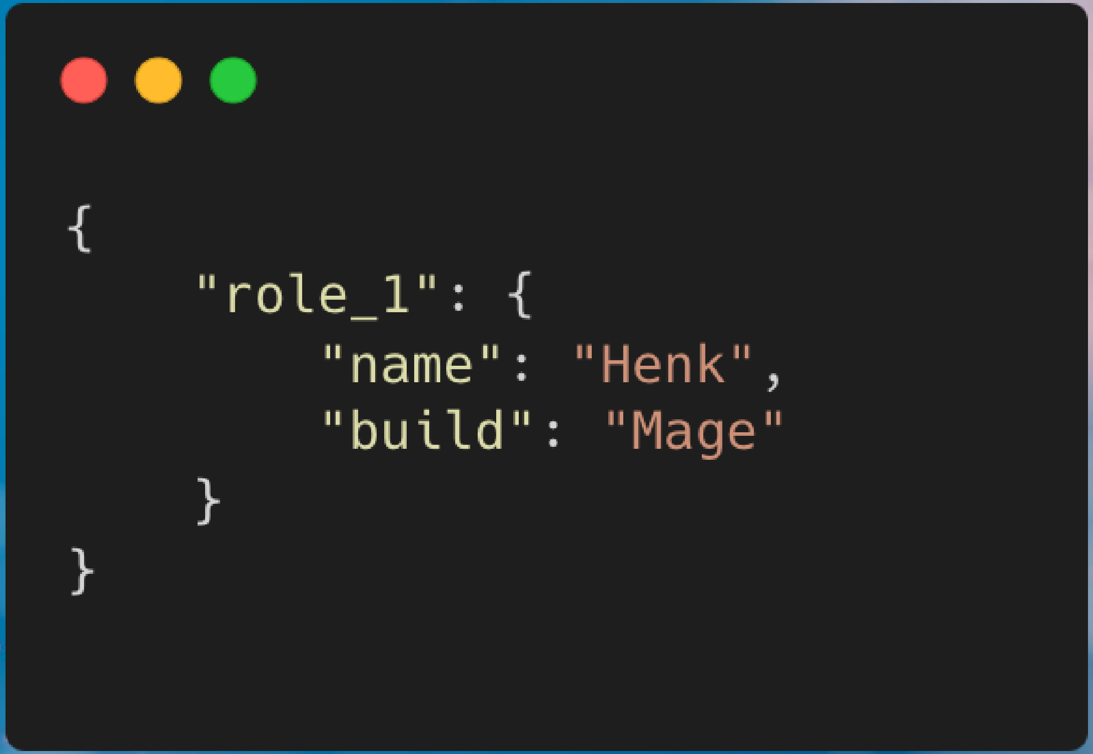
#3 Hooks and fixtures (1)

- **Refactor** the test by putting the login action in a 'hook'.

- *Brain teaser:*
  What hook is the best '*type of hook*' to perform teardown and cleanup stuff?

#3 Hooks and fixtures (2)

- **Add** a 'fixtures' directory (under ./tests)
- Add a file (in the fixtures directory) called '*testdata.json*' and add an object with the following key/value pairs



```
{
    "role_1": {
        "name": "Henk",
        "build": "Mage"
    }
}
```

#3 Hooks and fixtures (3)

- **Add / load** the fixture (*using cy.fixture()*) in the appropriate hook
- **Refactor** your test to use the fixture instead of 'hardcoded' data.

#4 Network requests

```
cy.request()
```

```
cy.intercept()
```

#4 Network requests (1)

- **Add** a POST request to the available endpoint (using *.request()*). Put it in a after/before hook.
  (checkout: https://test-rpg.vercel.app/api for endpoints)
- ***BONUS:***
  use the .then() method to yield the response object and assert the expected status code (200) and/or (console.)log the response object.

#4 Network requests (2)

- Spy on a GET request (sent by the front-end) to the '/api/builds' endpoint and assert the expected status code (200).

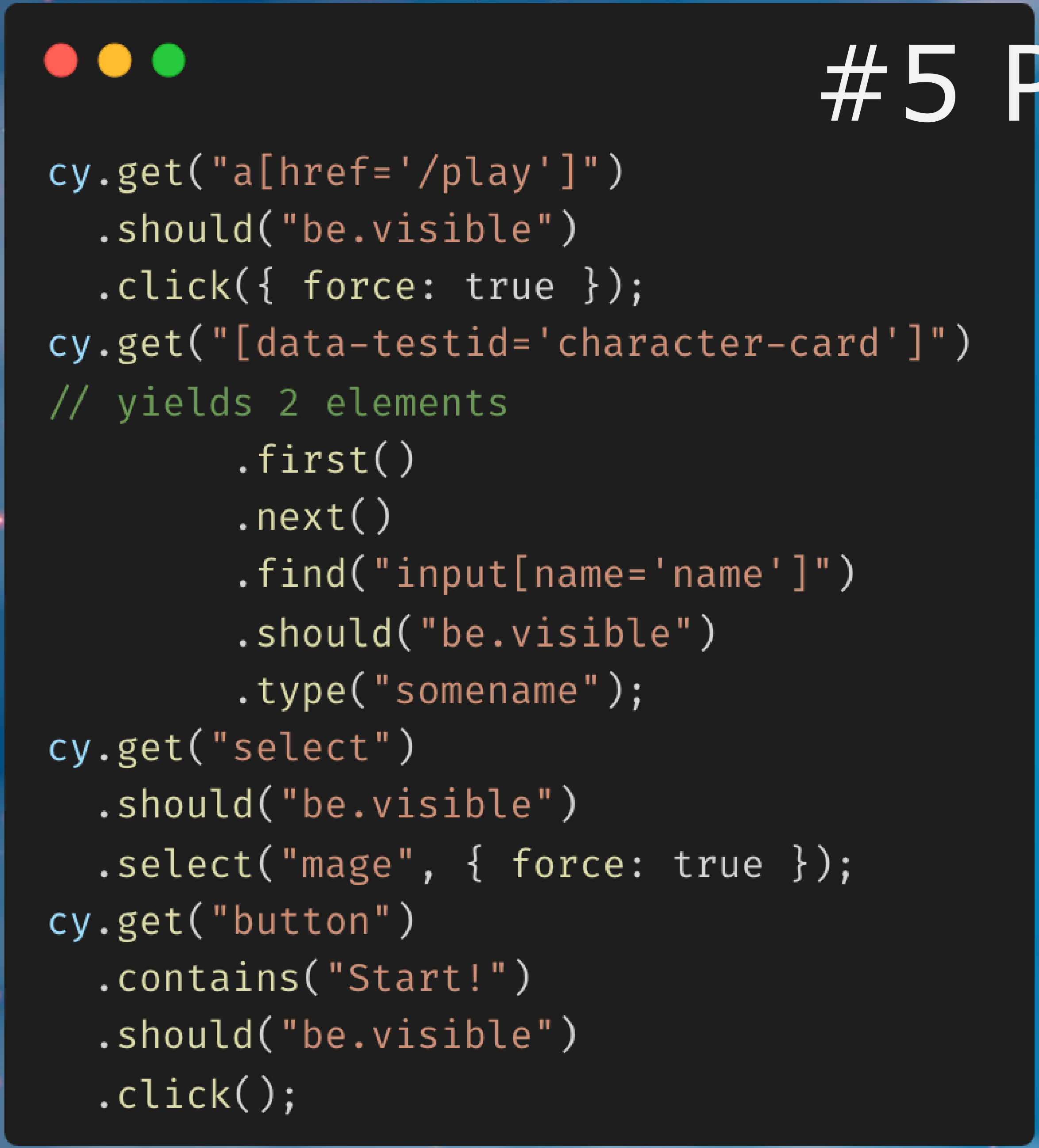
- *Hint: you'll need (at least) 4 different methods -> .intercept() being one of them...*

#4 Network requests (3)

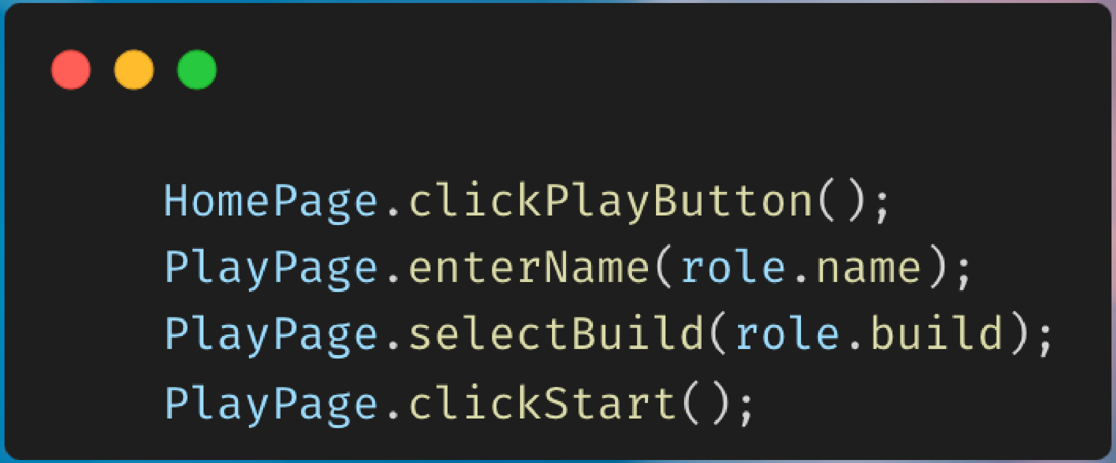
- Alter the response body (i.e. 'Mock') of the same call to the '/api/builds' to set all 'skills levels' to there maximum value.

- See the effect of your mocked data in the app

#5 Page object model (0)



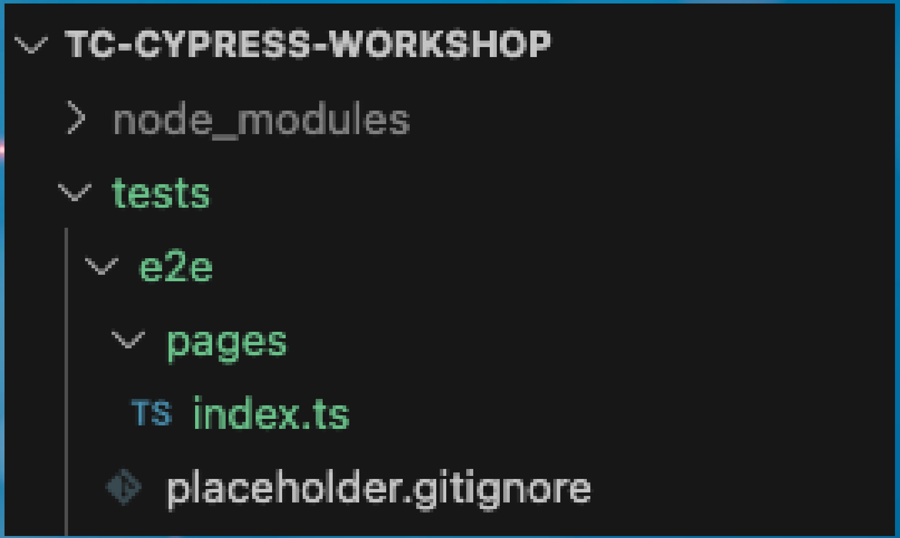
```
cy.get("a[href='/play']")
  .should("be.visible")
  .click({ force: true });
cy.get("[data-testid='character-card']")
// yields 2 elements
        .first()
        .next()
        .find("input[name='name']")
        .should("be.visible")
        .type("somename");
cy.get("select")
  .should("be.visible")
  .select("mage", { force: true });
cy.get("button")
  .contains("Start!")
  .should("be.visible")
  .click();
```



```
    HomePage.clickPlayButton();
    PlayPage.enterName(role.name);
    PlayPage.selectBuild(role.build);
    PlayPage.clickStart();
```

#5 Page object model (1)

1. **Add** a subfoler 'pages' at ./tests/e2e and **add** and *index.ts* file



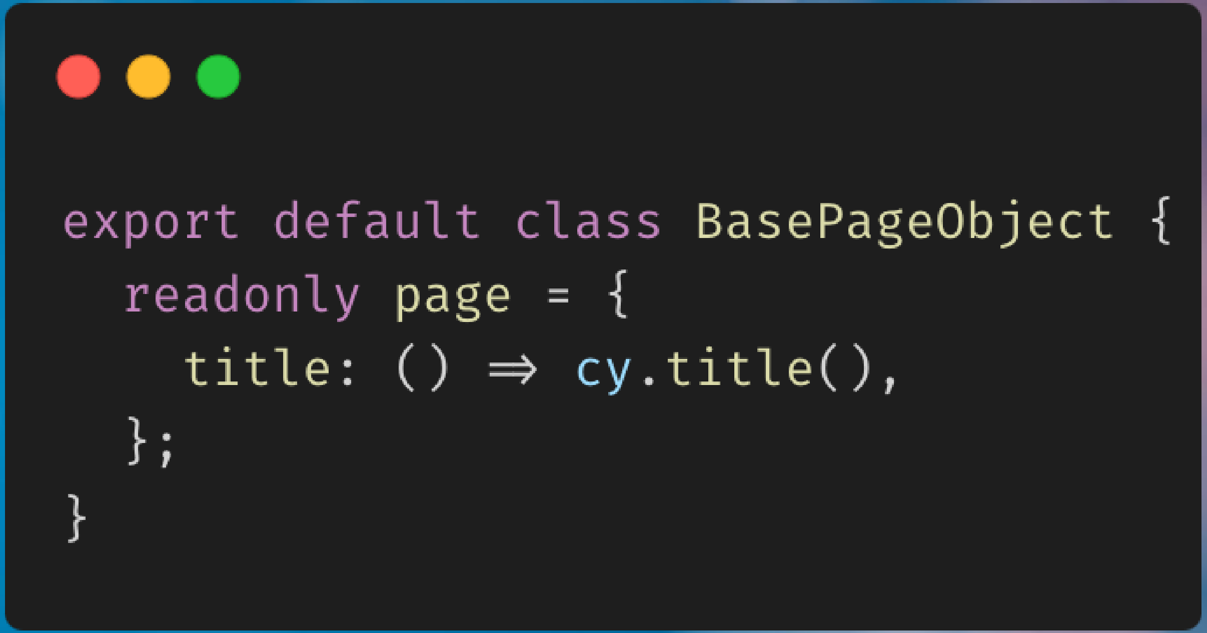
```
TC-CYPRESS-WORKSHOP
  > node_modules
  v tests
    v e2e
      v pages
        TS index.ts
      placeholder.gitignore
```

#5 Page object model (2)

- **<u>Add</u>** your first pageObject file and name it '*homePage.po.ts*'

BONUS:

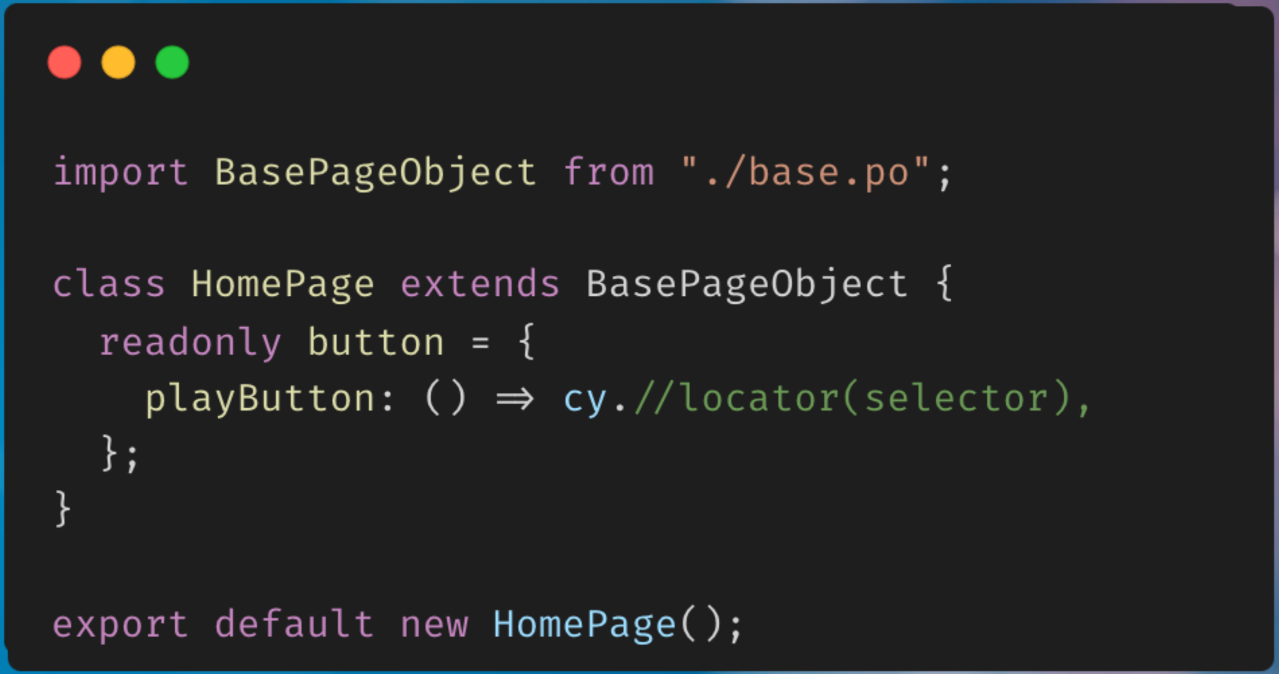
Add and define a base class in '*base.po.ts*' for other page objects (classes) to enherit from



```
export default class BasePageObject {
  readonly page = {
    title: () ⇒ cy.title(),
  };
}
```

#5 Page object model (3)

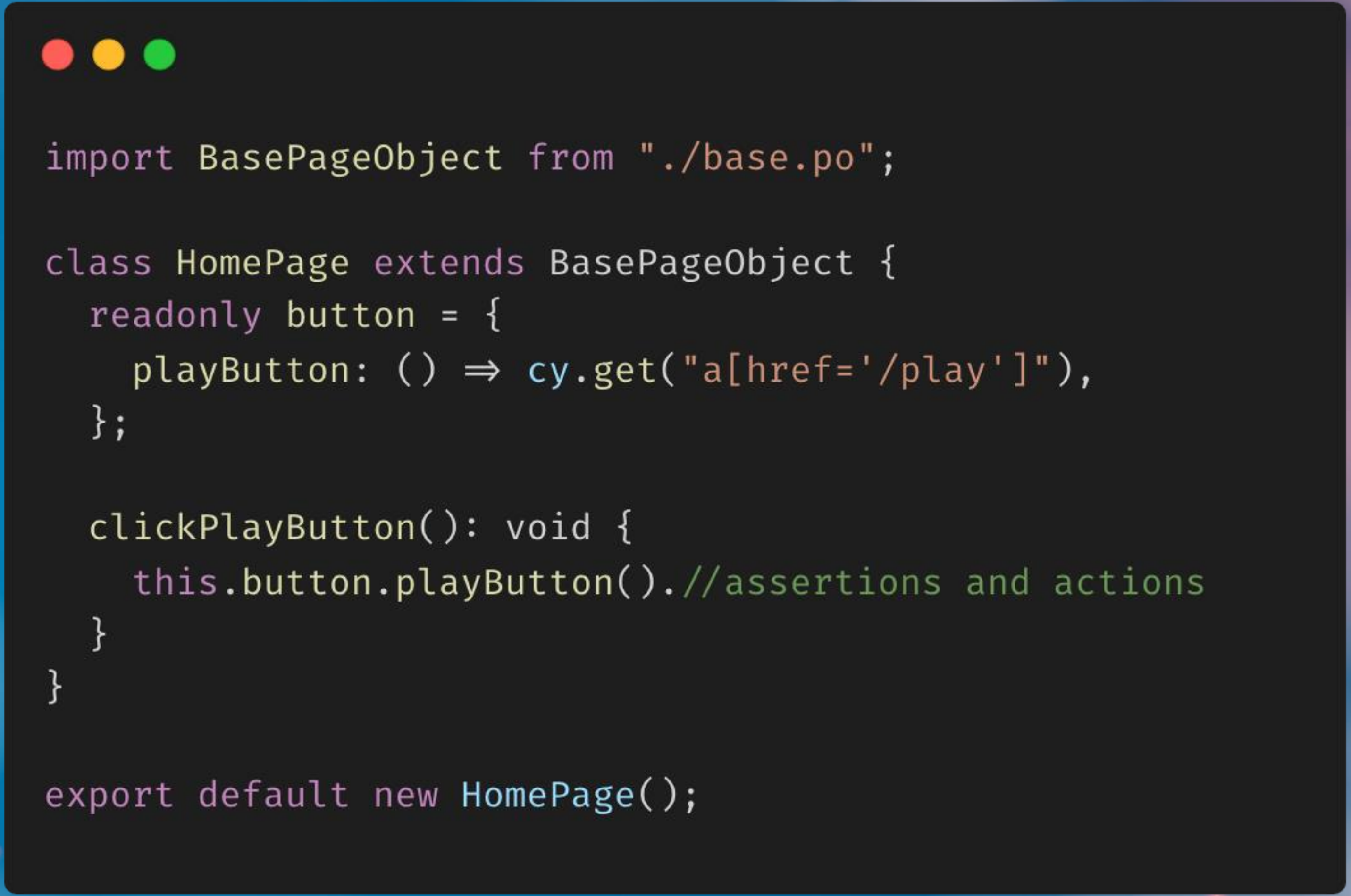
- **Define** the page object (HomePage) and an attribute for all the buttons on the page.
- Add the 'playButton' plus locator/selector code *(hint: use arrow function)*



```
import BasePageObject from "./base.po";

class HomePage extends BasePageObject {
  readonly button = {
    playButton: () ⇒ cy.//locator(selector),
  };
}

export default new HomePage();
```

#5 Page object model (3)

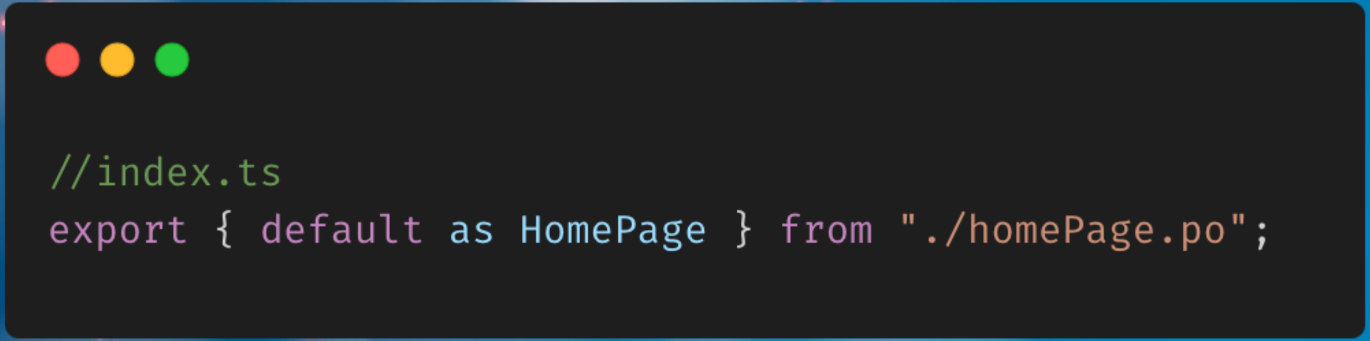
- **<u>Add</u>** methods to the class that represents 'Actions' on the html elements. <u>PAY ATTENTION TO READABLE NAMING!!!</u>



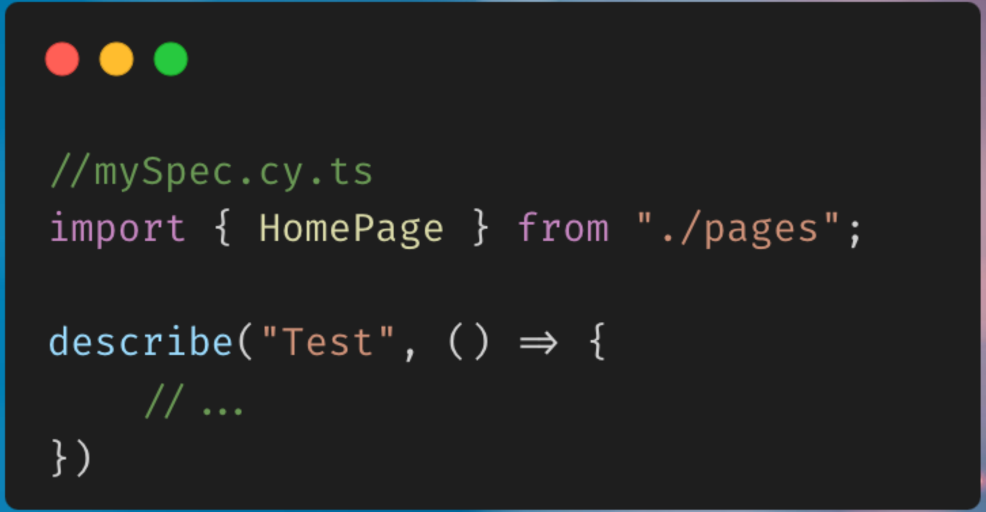
```
import BasePageObject from "./base.po";

class HomePage extends BasePageObject {
  readonly button = {
    playButton: () ⇒ cy.get("a[href='/play']"),
  };

  clickPlayButton(): void {
    this.button.playButton().//assertions and actions
  }
}

export default new HomePage();
```

#5 Page object model (3)

- Add your (exported) pageobject to the index.ts file and then import the pageobject(s) in your test files (*cy.ts).



```
//index.ts
export { default as HomePage } from "./homePage.po";
```



```
//mySpec.cy.ts
import { HomePage } from "./pages";

describe("Test", () => {
    // ...
})
```

- **Refactor** the test(s) to use the pageobject methods in your test (*.cy.ts) cypress

Yield vs Return

Syntax

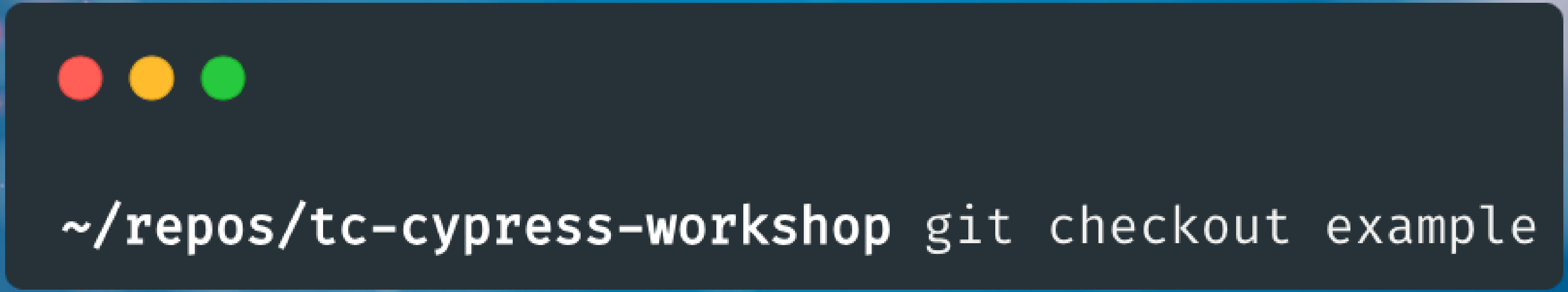
Browser access

ewald@testcoders.nl

ewaldverhoeven.nl

https://github.com/EwaldVerhoeven

TESTCODERS



```
~/repos/tc-cypress-workshop git checkout example
```