Cypress notes

* example

***cy***.get('input.post-title') //2 Find the <input> with class post-title  
 .type('My First Post') //3 Type "My First Post" into it  
  
***cy***.get('input.post-body') //4 Find the <input> with class post-body  
 .type('Hello, World') //5 Type "Hello< World" into it  
  
***cy***.contains('Submit') //6 Find the element containing the text Submit  
 .click() //7 Click it  
  
***cy***.url() //8 Grab the browser URL, ensure it includes/post/my-first-post  
 .should('include','/posts/my-first-post')  
  
***cy***.get('h1') //9 Find the h1 tag, ensure it contains the text "My Firs Post"  
 .should('contain','My First Post')

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

In Cypress, when you want to interact with a DOM element directly, call [.then()](https://docs.cypress.io/api/commands/then.html) with a callback function that receives the element as its first argument. When you want to skip the retry-and-timeout functionality entirely and perform traditional synchronous work, use [Cypress.$](https://docs.cypress.io/api/utilities/$.html).

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//Find an element in the document containing the text 'New Post'  
***cy***.contains('New Post')  
  
//Find an element within '.main' containing the text 'New Post'  
***cy***.get('.main').contains('New Post')

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***Timeout***

Cypress doesn’t fail immediately the first time an element is not found. Instead, Cypress gives your app a window of time to finish whatever it may be doing! This is known as a *timeout*

// Give this element 10 seconds to appear

cy.get('.my-slow-selector', { timeout: 10000 })

cy

.get('.mobile-nav')

.should('be.visible')

.and('contain', 'Home')

Queries for the element .mobile-nav  
✨**and waits up to 4 seconds for it to exist in the DOM**✨  
✨**and waits up to 4 seconds for it to be visible**✨  
✨**and waits up to 4 seconds for it to contain the text: ‘Home’**✨

cy

.get('.mobile-nav', { timeout: 10000 })

.should('be.visible')

.and('contain', 'Home')

Gets the element .mobile-nav  
✨**and waits up to 10 seconds for it to exist in the DOM**✨  
✨**and waits up to 10 seconds for it to be visible**✨  
✨**and waits up to 10 seconds for it to contain the text: ‘Home’**✨

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Assertions*

*Assertions describe the****desired****state of your****elements****, your****objects****, and your****application****.*

Assertions let you do things like ensuring an element is visible or has a particular attribute, CSS class, or state. Assertions are commands that enable you to describe the desired state of your application.

Cypress will automatically wait until your elements reach this state, or fail the test if the assertions don’t pass. 🡪***.should***

**Default Assertions**

Many commands have a default, built-in assertion, or rather have requirements that may cause it to fail without needing an explicit assertion you’ve added.

For instance:

* [cy.visit()](https://docs.cypress.io/api/commands/visit.html) expects the page to send text/html content with a 200 status code.
* [cy.request()](https://docs.cypress.io/api/commands/request.html) expects the remote server to exist and provide a response.
* [cy.contains()](https://docs.cypress.io/api/commands/contains.html) expects the element with content to eventually exist in the DOM.
* [cy.get()](https://docs.cypress.io/api/commands/get.html) expects the element to eventually exist in the DOM.
* [.find()](https://docs.cypress.io/api/commands/find.html) also expects the element to eventually exist in the DOM.
* [.type()](https://docs.cypress.io/api/commands/type.html) expects the element to eventually be in a *typeable* state.
* [.click()](https://docs.cypress.io/api/commands/click.html) expects the element to eventually be in an *actionable* state.
* [.its()](https://docs.cypress.io/api/commands/its.html) expects to eventually find a property on the current subject.

Certain commands may have a specific requirement that causes them to immediately fail without retrying: such as [cy.request()](https://docs.cypress.io/api/commands/request.html).

cy.get(‘button.close’).click().should(‘not.exist’)

By adding [.should('not.exist')](https://docs.cypress.io/api/commands/should.html) to any DOM command, Cypress will reverse its default assertion and automatically wait until the element does not exist.

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

// Find the el with id 'some-link'

.get('#some-link')

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* You cannot race or run multiple commands at the same time (in parallel).
* You cannot ‘accidentally’ forget to return or chain a command.
* You cannot add a .catch error handler to a failed command.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

context() is identical to describe()

specify() is identical to it()

**Hooks**

These are helpful to set conditions that you want to run before a set of tests or before each test. They’re also helpful to clean up conditions after a set of tests or after each test.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**data-cy**

data-cy -> I qasemi komponentes

<itb-defect-summary> *emir I komponentes -> DoubleClick -> CTRL C -> file spec.ts Double Shift ->*

*CTRL P -> DoubleClick on html ->add data-cy*

***Clear DB***

support/index.js

const { clearDatabase } = require('../../server/db')  
on('task', {  
 'clear:db' : () => {  
 return clearDatabase;  
 }  
});

integation/spec.ts

***beforeEach***(() => {  
 ***cy***.task('clear:db')  
})

Best Practices

Use data-\* attributes to provide context to your selectors and isolate them from CSS or JS changes.

Oftentimes we see users run into problems targeting their elements because:

* Your application may use dynamic classes or ID’s that change
* Your selectors break from development changes to CSS styles or JS behavior

Luckily, it is possible to avoid both of these problems.

1. Don’t target elements based on CSS attributes such as: id, class, tag
2. Don’t target elements that may change their textContent
3. Add data-\* attributes to make it easier to target elements

Cypress enables you to write all types of tests:

* End-to-end tests
* Integration tests
* Unit tests

End to End Testing, also called E2E or UI testing is one the many testing phases covering a web application.

By writing an End to End Test it is possible to assert whether a web application **works as expected or not**. Plus, with E2E you will test the user flow of your application. Starting from the signup process.

**Users**, **articles**. All of this data are good candidates for **fixtures** aren’t they? What is a fixture in testing? A **fixture** is a piece of static data that can be used during your tests.

For my **admin user** I can create a fixture inside cypress/fixtures/users/admin.json. The fixture will look like the following:

1. {
2. **"name"**: "Valentino",
3. **"email"**: "valentino@valentinog.com",
4. **"password"**: "secret"
5. }

Be aware that it is important to have **fixtures loaded before each test**. Otherwise you won’t be able to access them. It is a good idea to load fixtures inside a [beforeEach hook](https://docs.cypress.io/guides/core-concepts/writing-and-organizing-tests.html" \l "Hooks" \t "_blank):

1. describe("Knowledge Base Application", () => {
2. beforeEach(() => {
3. cy.fixture("users/admin").**as**("admin");
4. });