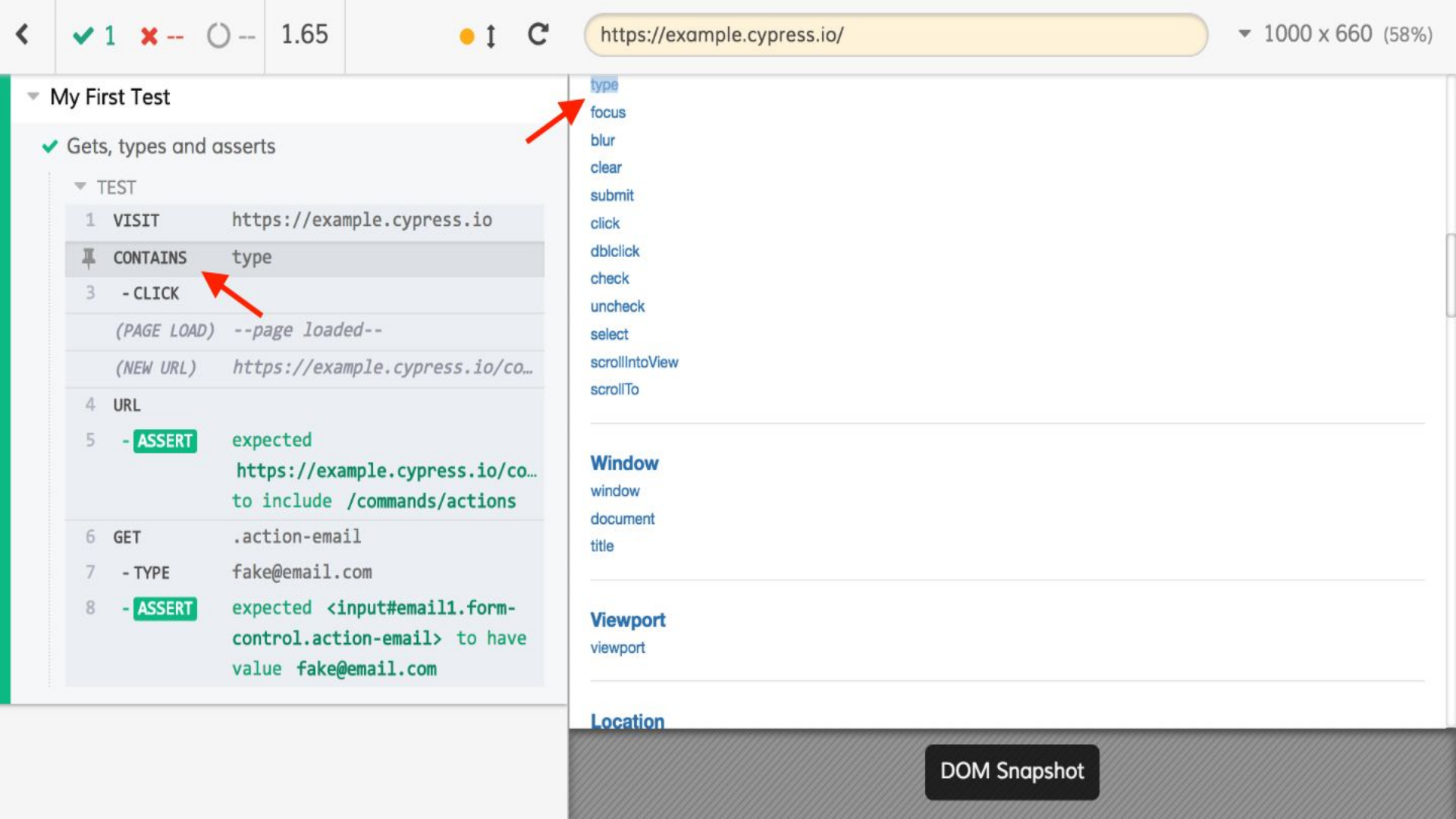
Cypress Testing 101

What is Cypress?

What is Cypress?


Green Line (UN)
UK military bases
Controlled by the self-proclaimed Turkish Republic of Northern Cyprus, recognised solely by Ankara
30 km
Kyrenia
NICOSIA
Famagusta
Larnaca
Paphos
Limassol
Mediterranean Sea
ANKARA
TURKEY

What is Cypress?


1
--
--
1.65
https://example.cypress.io/
1000 x 660 (58%)
My First Test
Gets, types and asserts
TEST
1 VISIT https://example.cypress.io
CONTAINS type
3 - CLICK
(PAGE LOAD) --page loaded--
(NEW URL) https://example.cypress.io/co…
4 URL
5 - ASSERT expected https://example.cypress.io/co… to include /commands/actions
6 GET .action-email
7 - TYPE fake@email.com
8 - ASSERT expected <input#email1.form-control.action-email> to have value fake@email.com
type
focus
blur
clear
submit
click
dblclick
check
uncheck
select
scrollIntoView
scrollTo
Window
window
document
title
Viewport
viewport
Location
DOM Snapshot

What we’re covering:

- Running the tests locally
- Waits are **BAD**
- Some debugging tips
- Code layout / “Commands”
- Selectors
- Flakiness avoiding

That’s a lot of content

If you want to skip this:

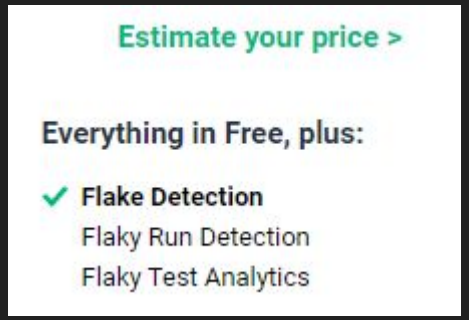
https://docs.cypress.io/guides/core-concepts/introduction-to-cypress

https://docs.cypress.io/guides/references/best-practices

First a Diversion:

End2End UI tests are hard.

They test against a running program, and UIs are generally unstable and stateful things by design



Randomness:

```
// your app code

// random amount of time
const random = Math.random() * 100

// create a <button> element
const btn = document.createElement('button')

// attach it to the body
document.body.appendChild(btn)

setTimeout(() => {
  // add the class active after an indeterminate amount of time
  btn.setAttribute('class', 'active')
}, random)
```

```
// your cypress test code
it('does something different based on the class of the button', () => {
  // RERUN THIS TEST OVER AND OVER AGAIN
  // AND IT WILL SOMETIMES BE TRUE, AND
  // SOMETIMES BE FALSE.

  cy.get('button').then(($btn) => {
    if ($btn.hasClass('active')) {
      // do something if it's active
    } else {
      // do something else
    }
  })
})
```

Selectors

One of the most common operations during an E2E cypress test is “get the thing on this page that matches X”

There are lots of different ways to do this

Selectors according to Cypress

Given a button that we want to interact with:

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

Let's investigate how we could target it:

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.

How to run locally

`npm run cy:open`

(have the backend already running)

Waiting is BAD

DO NOT WAIT

You never need to write cy.wait() before doing a DOM command

You never need to write .should('exist') after a DOM based command.

DOM commands include:
cy.get(), cy.visit(), cy.contains()
.click(), .find()

DO NOT WAIT

There **CAN** be exceptions

- As with any good rule there are exceptions.
- As with any really good rule there are almost no exceptions

DO NOT WAIT

Why are waits bad?

- Wasteful
- It’s a code smell

Debugging methods

cy.log()

cy.pause()

console.log() / log() in the application code

An example test layout

- fixtures - test data
- specFolder - the tests!

- plugins/index.ts - cypress config
- support/index.ts - cypress config

- commands.ts - reusable snippets

Avoiding flakiness

- Follow the Cypress best-practices as much as possible
- Be as specific as possible about what you assert/are testing
- Be as general as possible about how you get there