Cypress Browser test runner

CONCEPTS

> cypress open is interactive with a time-travel runner; cypress run is headless and meant for CI.

> Cy commands are queued and run later, not returned. Chain off them instead of using await.

Visit cy.visit
↓ query
Query cy.get, contains
↓ act
Act .click, .type
↓ assert
Assert .should, retries

> Queries and .should() retry until they pass or time out, so avoid fixed cy.wait(<ms>) pauses.

INSTALL AND MAINTENANCE

npm install -D cypress
Add Cypress to the project

npx cypress verify
Check the binary can run

OPEN THE APP

npx cypress open
Open the interactive runner

RUN HEADLESS

npx cypress run
Run all specs headless

npx cypress run --spec *<glob>*
Run matching spec files

npx cypress run --browser *<name>*
chrome, chromium, edge or firefox

npx cypress run --headed
Show the browser window

CONFIG, ENV AND REPORTERS

npx cypress run --config *<k=v, ...>*
Override config values inline

npx cypress run --config baseUrl=*<url>*
Point tests at another host

npx cypress run --env *<k=v, ...>*
Set env values for Cypress.env()

VISIT AND QUERY

cy.visit(*<url>***)**
Load a page

cy.get(*<selector>***)**
Find elements by selector or alias

cy.contains(*<text>***)**
Find an element by its text

ACTIONS

.click()
Click the subject

.type(*<text>***)**
Type into the subject

ASSERTIONS

.should('be.visible')
Assert visible

.should('have.text', *<text>***)**
Assert exact text

.should('contain', *<text>***)**
Assert text is included

NETWORK AND DATA

cy.intercept(*<method>***,** *<url>***).as(***<name>***)**
Spy on a request and alias it

cy.intercept(*<method>***,** *<url>***,** *<response>***)**
Stub a response

cy.wait('@*<name>***')**
Wait for the aliased request