Cypress Browser test runner

START HERE

`npm install -D cypress`
Add to project

`npx cypress open`
Open the app

`npx cypress run`
Run headless

`npx cypress run --spec <glob>`
Run some specs

`npx cypress run --browser <name>`
Pick a browser

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

`cy.get(<selector>)`
Find elements

`cy.intercept(<method>, <url>)`
Spy or stub requests

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

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.

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

--config and --env values override the config file for that run only. Env vars named CYPRESS_* are read too.

INSTALL AND MAINTENANCE

`npm install -D cypress`
Add Cypress to the project

`npx cypress install`
Download the binary to the cache

`npx cypress install --force`
Reinstall even if cached

`npx cypress verify`
Check the binary can run

`npx cypress info`
Show browsers, paths, OS and env

`npx cypress version`
Show package, binary, Electron, Node

`npx cypress version --component <name>`
One of package, binary, electron, node

`npx cypress cache list`
List cached versions

`npx cypress cache list --size`
Add disk sizes (slow)

`npx cypress cache prune`
Remove all but the installed version

`npx cypress cache clear`
Delete every cached version

OPEN THE APP

`npx cypress open`
Open the interactive runner

`npx cypress open --e2e`
Start in end-to-end mode

OPEN THE APP (cont.)

`npx cypress open --component`
Start in component testing mode

`npx cypress open --browser <path>`
Use a custom browser

`npx cypress open --config-file <file>`
Use another config file

RUN HEADLESS

`npx cypress run`
Run all specs headless

`npx cypress run --spec <glob>`
Run matching spec files
Must also match specPattern in the config.

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

`npx cypress run --headed`
Show the browser window

`npx cypress run --e2e`
Run end-to-end tests (default)

`npx cypress run --component`
Run component tests

`npx cypress run --posix-exit-codes`
Exit 1 on failure, not the failure count

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 --config-file <file>`
Use another config file

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

`npx cypress run --reporter <name>`
Mocha reporter, e.g. junit

CYPRESS CLOUD

`npx cypress run --record --key <key>`
Record the run to Cloud

CYPRESS CLOUD (cont.)

`npx cypress run --record --parallel`
Split specs across machines

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

`cy.contains(<selector>, <text>)`
Text match limited to a selector

`cy.get(<sel>).find(<sel>)`
Search inside a subject

`cy.get(<sel>).first() / .last() / .eq(n)`
Pick one from a set

`cy.get(<sel>).within(<fn>)`
Scope commands to one element

`cy.get(<sel>).as(<name>)`
Alias an element

`cy.url()`
Yield the current URL

ACTIONS

`.click()`
Click the subject

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

`.type(<text>{enter})`
Type, then press Enter

`.clear()`
Empty an input

`.check() / .uncheck()`
Tick or untick a box

`.select(<value>)`
Choose a dropdown option

ASSERTIONS

`.should('be.visible')`
Assert visible

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

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

`.should('have.length', <n>)`
Assert element count

`.should('have.value', <v>)`
Assert input value

ASSERTIONS (cont.)

`.should('have.attr', <name>, <v>)`
Assert an attribute

`.should('not.exist')`
Assert it is gone

`cy.url().should('include', <path>)`
Assert the route

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

`cy.request(<method>, <url>, <body>)`
Send an HTTP request directly

`cy.fixture(<file>)`
Load data from the fixtures folder

`cy.intercept(<url>, { fixture: <file> })`
Stub with a fixture file

`cy.session(<id>, <setup>)`
Cache and restore a login

`cy.task(<name>, <arg>)`
Run Node code from the config

RECIPES

Run one spec in Firefox

```
# headless, one file, with a
base URL override
npx cypress run --browser
firefox --spec "cypress/e2e/
login.cy.js" --config
baseUrl=http://localhost:3000
```

Stub an API and assert

```
cy.intercept('GET', '/api/
users', { fixture:
'users.json' }).as('users')
cy.visit('/users')
cy.wait('@users')
cy.get('li').should('have.
length', 3)
```

Log in once per spec

```
cy.session('user', () => {
  cy.request('POST', '/api/
login', { user: 'a', pass:
'b' })
})
cy.visit('/dashboard')
```

Fresh cache in CI

```
npx cypress cache prune
npx cypress verify
npx cypress run --record --
parallel --group ci
```