# 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 install**
Download the binary to the cache

**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

## OPEN THE APP

**npx cypress open**
Open the interactive runner

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

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

## RUN HEADLESS

**npx cypress run**
Run all specs headless

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

## RUN HEADLESS (cont.)

**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

## 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

**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

## VISIT AND QUERY (cont.)

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

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

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

## ACTIONS

**.click()**
Click the subject

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

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

## 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('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.session(***<id>***, *<setup>*)**
Cache and restore a login