

Cypress Workshop



Cypress Workshop



Workshop

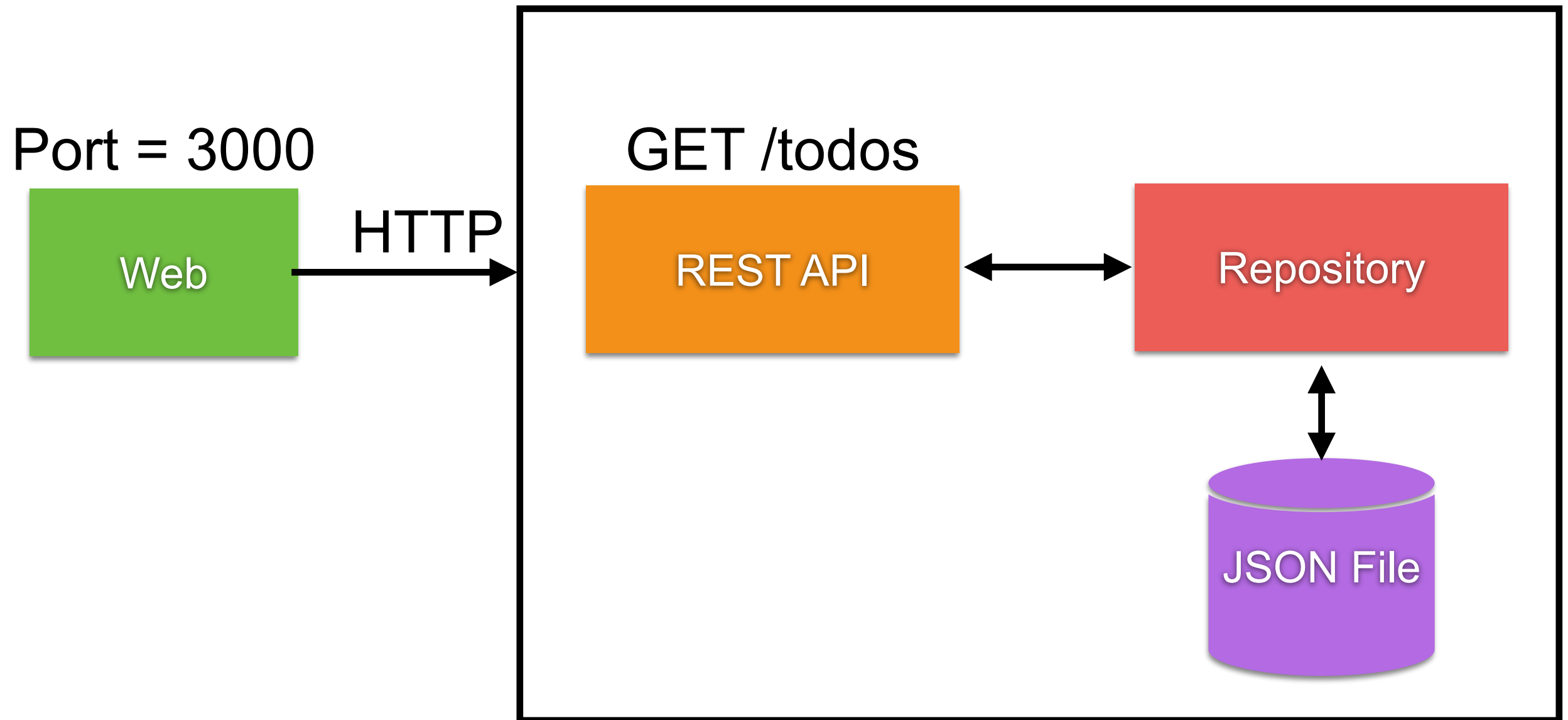
<https://github.com/up1/workshop-testing-cypress>



Frontend Testing with Cypress



Architecture



Library in workshop

TypeScript

Cypress

Chrome plugin for Vue



Cypress

[Product](#)[Docs](#)[Community](#)[Company](#)[Pricing](#)[Log In](#)[Install >](#)

Test. Automate. Accelerate.

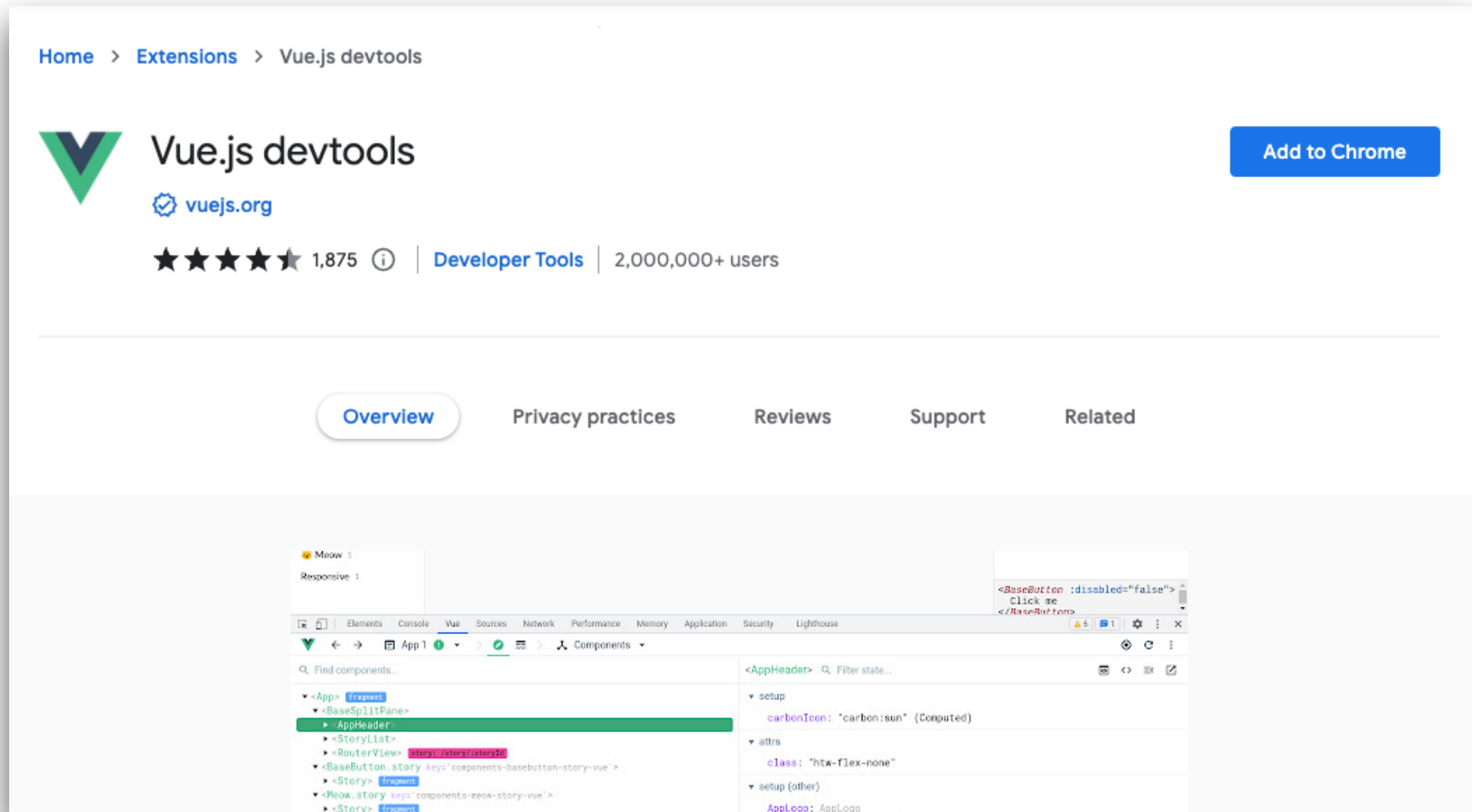
With Cypress, you can easily create tests for your modern web applications, debug them visually, and automatically run them in your continuous integration builds.

[> npm install cypress](#)[Documentation](#)

<https://www.cypress.io/>



Chrome plugin for Vue



<https://chrome.google.com/webstore/detail/vuejs-devtools/nhdogjmejiglipccpnnnanhbledajbpd>



Install Cypress

```
$npm init
```

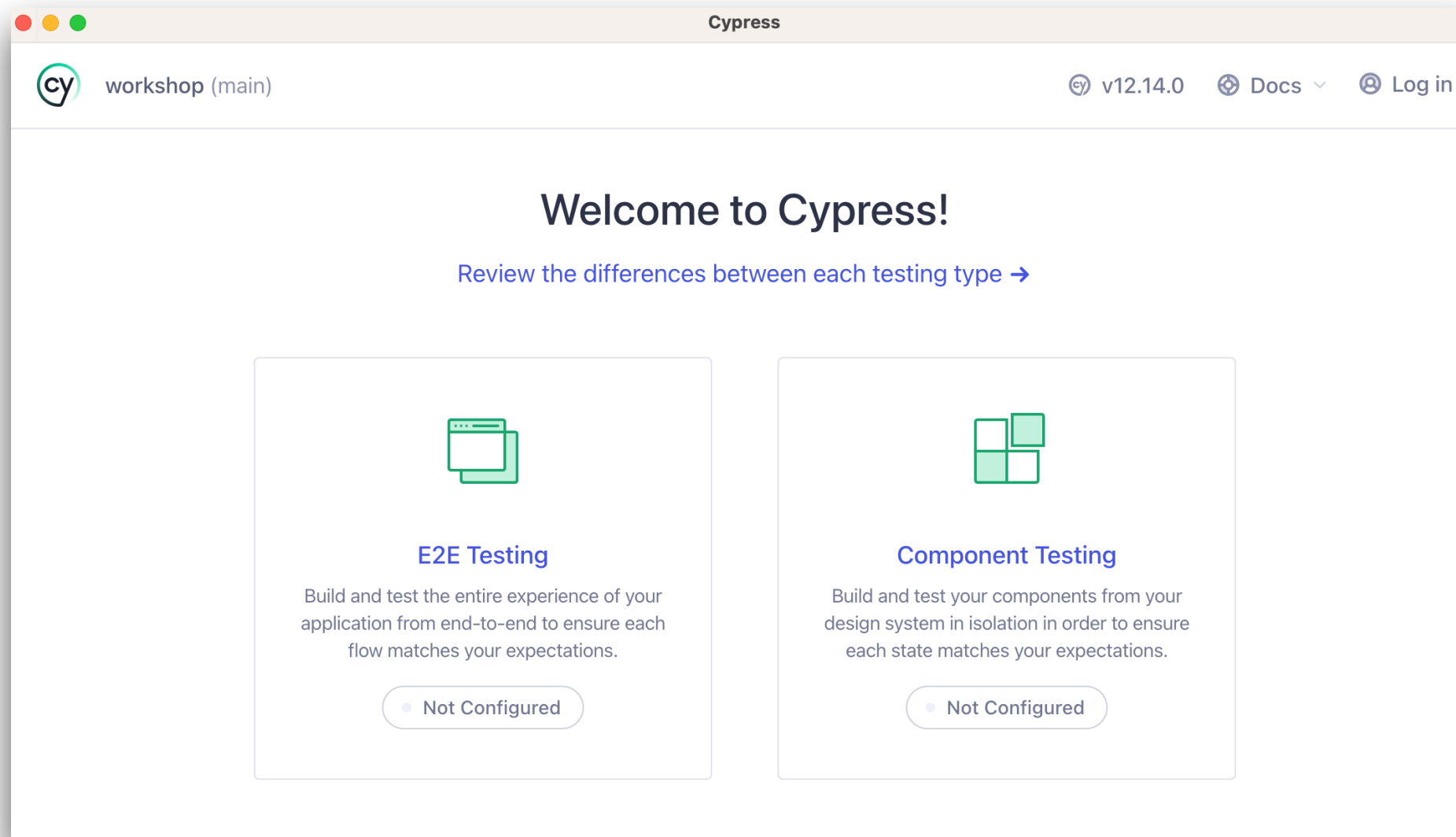
```
$npm install cypress --save-dev
```

<https://docs.cypress.io/guides/getting-started/installing-cypress>



Start Cypress UI

\$npx cypress open



<https://docs.cypress.io/guides/getting-started/opening-the-app>



Types of testing

Key differences — [Need help ?](#)



End-to-end tests:

- Visit URLs via `cy.visit()`
- Test flows and functionality across multiple pages
- Ideal for testing integrated flows in CD workflows

Code Example

```
1  it('only shows a modal on first visit', ()
2    cy.visit('http://localhost:3000/')
3    .get('[data-testid=modal]')
4    .should('be.visible')
5    .get('[aria-label=Close]')
6    .click()
7
8    // should not load a second time
9    .reload()
10   .get('[data-testid=modal]')
11   .should('not.exist')
12 })
```

Component tests:

- Import components via `cy.mount()`
- Test individual components of a design system in isolation
- Ideal for testing isolated flows and components in CI

Code Example

```
1  import BaseModal from './BaseModal'
2
3  it('closes when the X button is pressed',
4    cy.mount(<BaseModal />)
5    .get('[aria-label=Close]')
6    .click()
7    .get('[data-testid=modal]')
8    .should('not.exist')
9  })
```



Types of testing

End-to-End
Component

<https://docs.cypress.io/guides/core-concepts/testing-types>



End-to-End testing

Test app in web browser
Interact with third-party APIs
Interact like a user



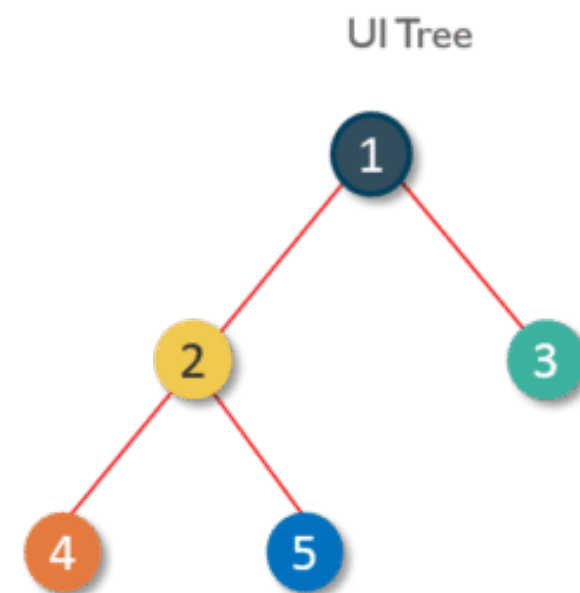
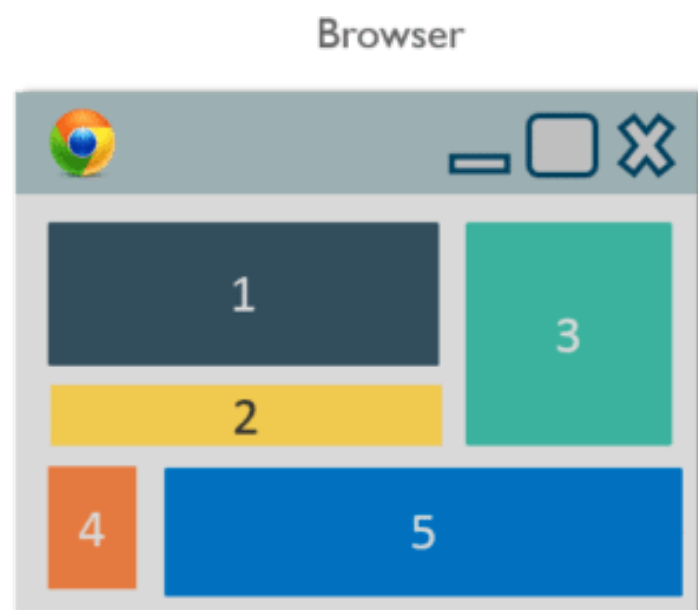
Component testing

Test in each component in app

Fast and reliable

Not call external APIs or services

Test logic and display in each component



Types of testing

	E2E	Component
What's Tested	All app layers	Individual component
Characteristics	Comprehensive, slower, more susceptible to flake	Specialized, quick, reliable
Used For	Verifying app works as a cohesive whole	Testing functionality of individual component
Written By	Developers, QA Team, SDETs	Developers, Designers
CI Infrastructure	Often requires complex setup	None needed
Initialization Command	<code>cy.visit(url)</code>	<code>cy.mount(<MyComponent />)</code>



Cypress structure

Folder name	Description
e2e	Keep specs/test files (.js, .ts)
fixtures	Keep static data that can be used in tests
supports	Config files for e2e and component test
downloads	
screenshot	
video	

<https://docs.cypress.io/guides/core-concepts/writing-and-organizing-tests>



Test structure

Mocha

Chai

TDD/BDD style

<https://docs.cypress.io/guides/core-concepts/writing-and-organizing-tests>



Mocha

JavaScript test framework on Node and browser



simple, flexible, fun

Mocha is a feature-rich JavaScript test framework running on [Node.js](https://nodejs.org/) and in the browser, making asynchronous testing *simple* and *fun*. Mocha tests run serially, allowing for flexible and accurate reporting, while mapping uncaught exceptions to the correct test cases. Hosted on [GitHub](https://github.com).

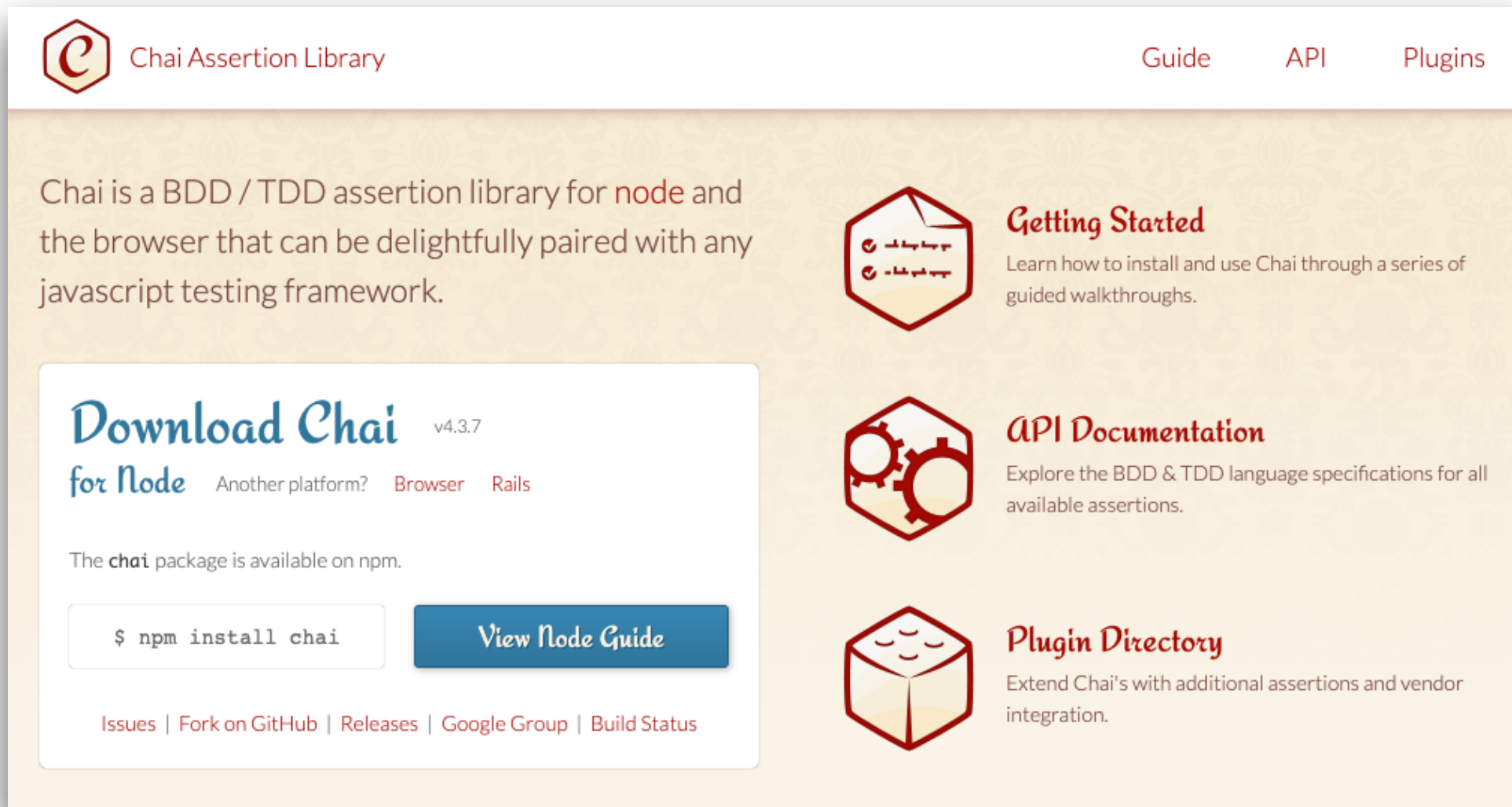
gitter [join chat](#) Sponsors 13 Backers 608

<https://mochajs.org/>



Chai

TDD/BDD assertion library for Node and browser



The screenshot shows the Chai Assertion Library website. At the top, there's a navigation bar with the Chai logo (a red 'c' in a hexagon) and the text 'Chai Assertion Library'. To the right of the logo are links for 'Guide', 'API', and 'Plugins'. Below the navigation bar, the main content area has a light beige background. On the left, there's a white box with the heading 'Download Chai' and 'v4.3.7'. Below this, it says 'for Node' and 'Another platform? Browser Rails'. A text line states 'The chai package is available on npm.' Below this are two buttons: '\$ npm install chai' and 'View Node Guide'. At the bottom of this box are links: 'Issues | Fork on GitHub | Releases | Google Group | Build Status'. To the right of the white box, there are three sections, each with a red hexagonal icon and a title. The first icon shows a document with a checkmark, titled 'Getting Started', with the text 'Learn how to install and use Chai through a series of guided walkthroughs.' The second icon shows two interlocking gears, titled 'API Documentation', with the text 'Explore the BDD & TDD language specifications for all available assertions.' The third icon shows a box with a smiley face, titled 'Plugin Directory', with the text 'Extend Chai's with additional assertions and vendor integration.'

Chai Assertion Library

Guide API Plugins

Chai is a BDD / TDD assertion library for **node** and the browser that can be delightfully paired with any javascript testing framework.

Download Chai v4.3.7
for Node Another platform? Browser Rails

The **chai** package is available on npm.

\$ npm install chai View Node Guide

Issues | Fork on GitHub | Releases | Google Group | Build Status

Getting Started
Learn how to install and use Chai through a series of guided walkthroughs.

API Documentation
Explore the BDD & TDD language specifications for all available assertions.

Plugin Directory
Extend Chai's with additional assertions and vendor integration.

<https://www.chaijs.com/>



Good Test ?

F.I.R.S.T

Fast

Independent / Isolate

Repeatable

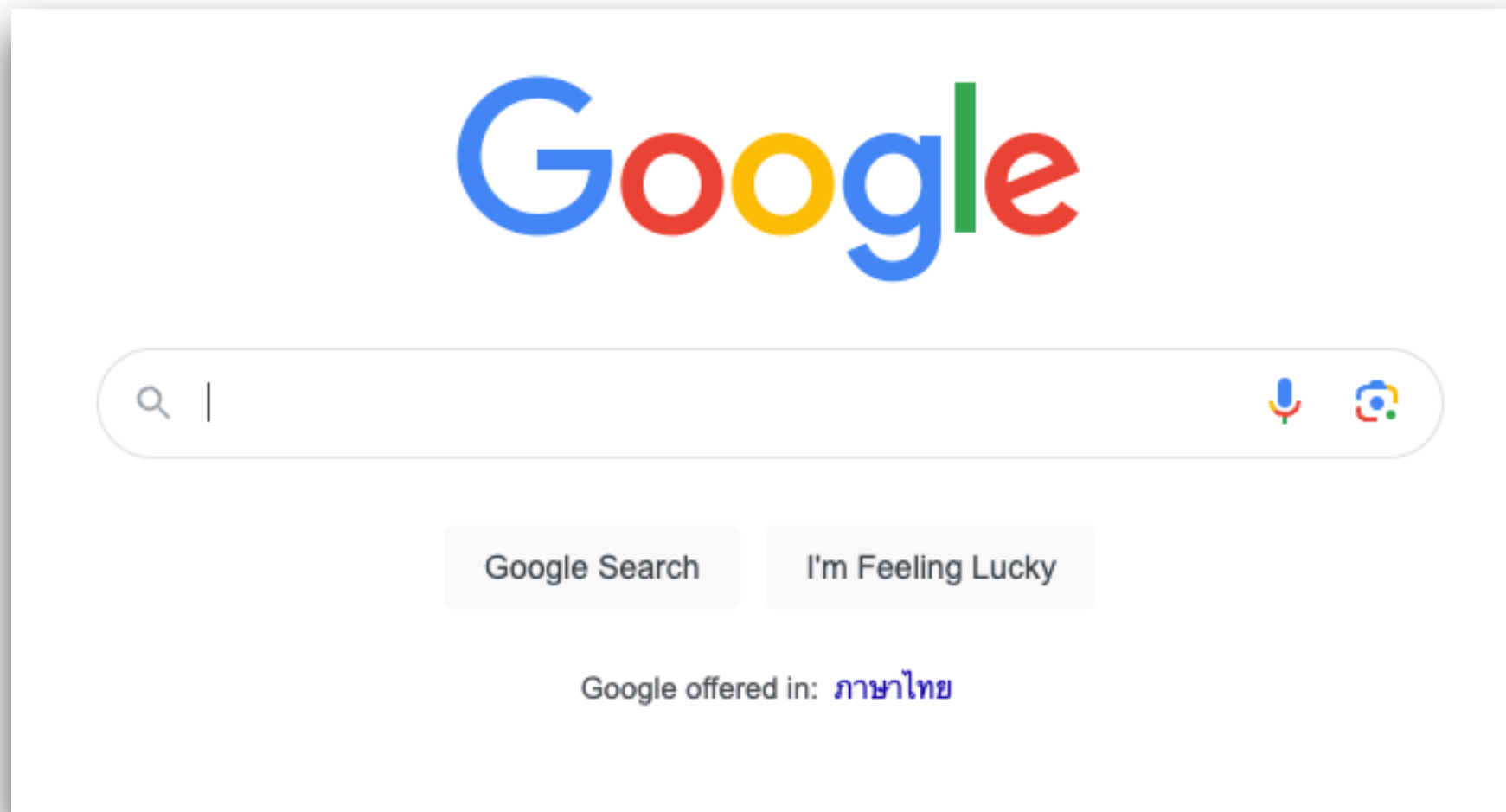
Self-verify

Timely



Hello with Google

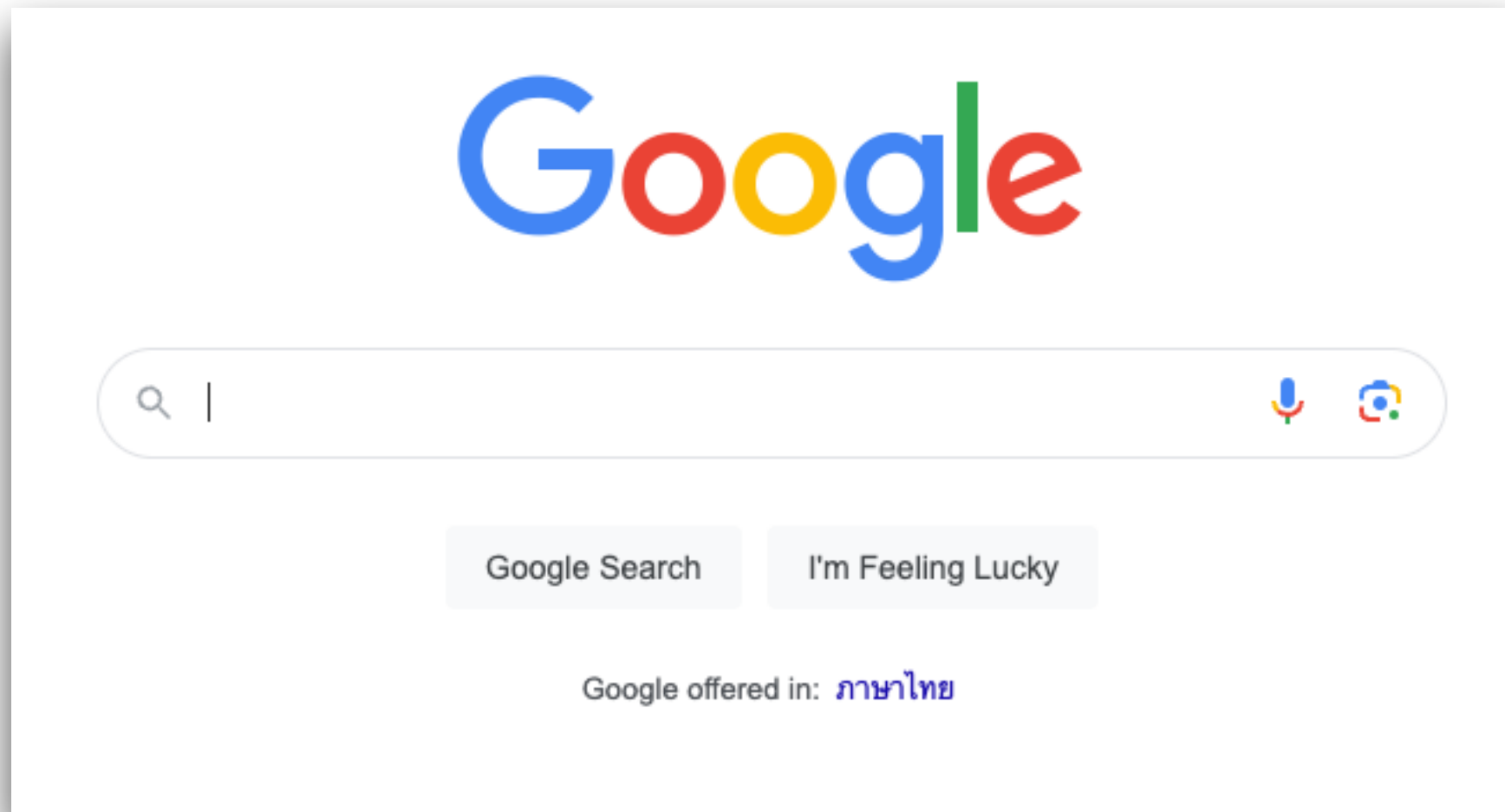




<https://www.google.com>



Test ?



First page loaded

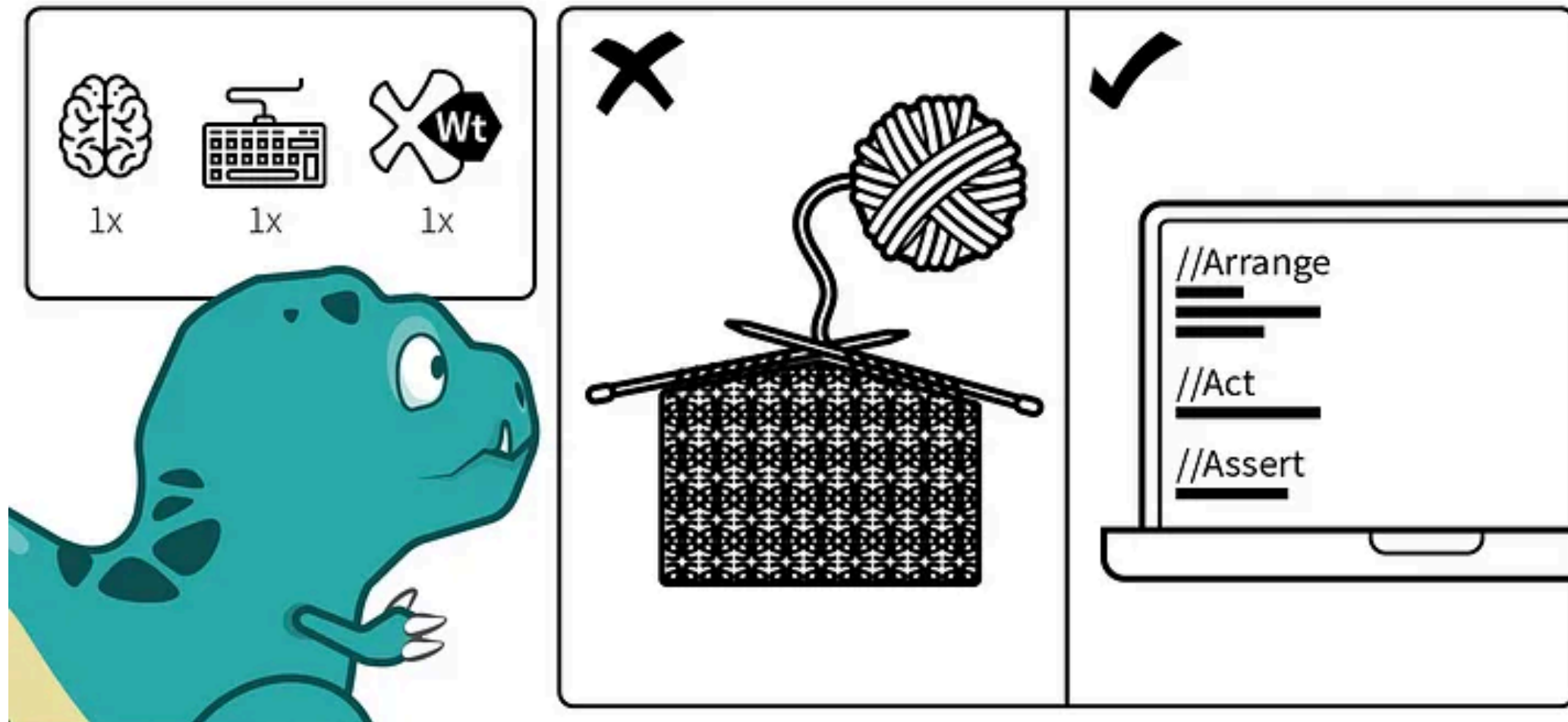
Create file /e2e/google.cy.js

```
/// <reference types="cypress" />
describe('Search with Google', () => {
  it('Loaded', () => {
    // Arrange
    cy.visit('https://www.google.com')
    // Act
    // Assert
    cy.get('.lnXdpd')

  })
})
```



AAA-Pattern



Web Element Locator (Selector)



Web Element Locator (Selector)

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



Web Element Locator (Selector)

Name	Description
id	<code>cy.get('#main').click()</code>
name	<code>cy.get('[name="submission"]').click()</code>
class	<code>cy.get('.btn.btn-large').click()</code>
xPath	Not support
data-cy	<code>cy.get('[data-cy="submit"]').click()</code>

<https://docs.cypress.io/guides/references/best-practices#Selecting-Elements>



Add XPath to Cypress

\$npm install -D @cypress/xpath

Edit file /support/e2e.js

```
require( '@cypress/xpath' );
```

Cypress no longer support !!!

<https://github.com/cypress-io/cypress/pull/26893>



```
/// <reference types="cypress" />

describe('Search with Google', () => {

  it('Loaded', () => {
    // Arrange
    cy.visit('https://www.google.com')
    // Act
    // Assert
    cy.get('.lnXdpd')

    cy.xpath('//div[4]/center/input[1]')
  })
})
```



Search result

Update file /e2e/google.cy.js

```
/// <reference types="cypress" />

describe('Search with Google', () => {

  it('Search', () => {
    // Arrange
    cy.visit('https://www.google.com')

    // Act
    cy.get('[name="q"]').type('Hello cypress')
    cy.get('[name="q"]').type('{enter}')

    // Assert
    cy.get('#result-stats')
    cy.contains('#result-stats', 'ผลการค้นหาประมาณ')
  })
})
```

<https://docs.cypress.io/api/commands/type>



Assertion

<https://docs.cypress.io/guides/references/assertions>



Cypress Hooks (Test life cycle)



Cypress hooks

Life cycle for test cases

Manage state for test

Clean up state after test

Remove duplicate logics



Cypress hooks

Name	Description
before	It runs once before starting the execution of first tests
after	It runs once after completion of all the tests
beforeEach	It runs before starting the execution of each of the tests
afterEach	It runs after finishing the execution of each of the tests

<https://docs.cypress.io/guides/core-concepts/writing-and-organizing-tests#Hooks>



```
before(() => {  
    // root-level hook  
    // runs once before all tests  
})  
  
beforeEach(() => {  
    // root-level hook  
    // runs before every test block  
})  
  
afterEach(() => {  
    // runs after each test block  
})  
  
after(() => {  
    // runs once all tests are done  
})
```



Duplication code !!

```
/// <reference types="cypress" />

describe('Search with Google', () => {

  it('Loaded', () => {
    // Arrange
    cy.visit('https://www.google.com')

    // Assert
    cy.get('.lnXdpd')
    cy.xpath('//div[3]/form/div[1]/div[1]/div[4]/center/input[1]')
  })

  it('Search', () => {
    // Arrange
    cy.visit('https://www.google.com')

    // Act
    cy.get('textarea[name="q"]')
    cy.get('[name="q"]').type('Hello cypress')
    cy.get('[name="q"]').type('{enter}')
  })
})
```



Improve code

```
describe('Search with Google', () => {
```

```
  beforeEach(() => {  
    // Arrange  
    cy.visit('https://www.google.com')  
  })
```

```
  it('Loaded', () => {  
    // Assert  
    cy.get('.lnXdpd')  
    cy.xpath('//div[3]/form/div[1]/div[1]/div[4]/center/input[1]')  
  })
```

```
  it('Search', () => {  
    // Act  
    cy.get('textarea[name="q"]')  
    cy.get('[name="q"]').type('Hello cypress')  
    cy.get('[name="q"]').type('{enter}')  
    // Assert  
    cy.get('#result-stats')  
    cy.contains('#result-stats', 'ผลการค้นหาประมาณ')  
  })
```

```
})
```



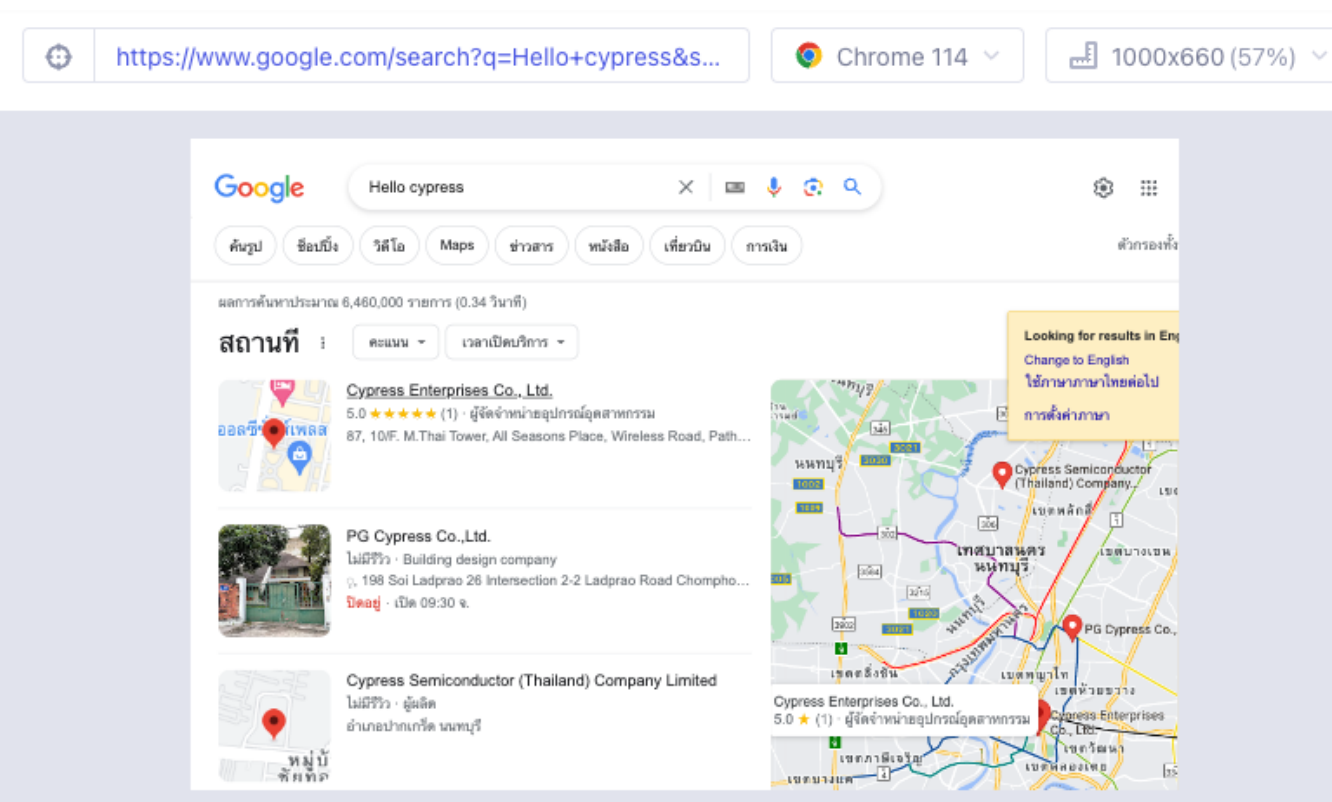
Working with XHR (XML HTTP Request)



XHR

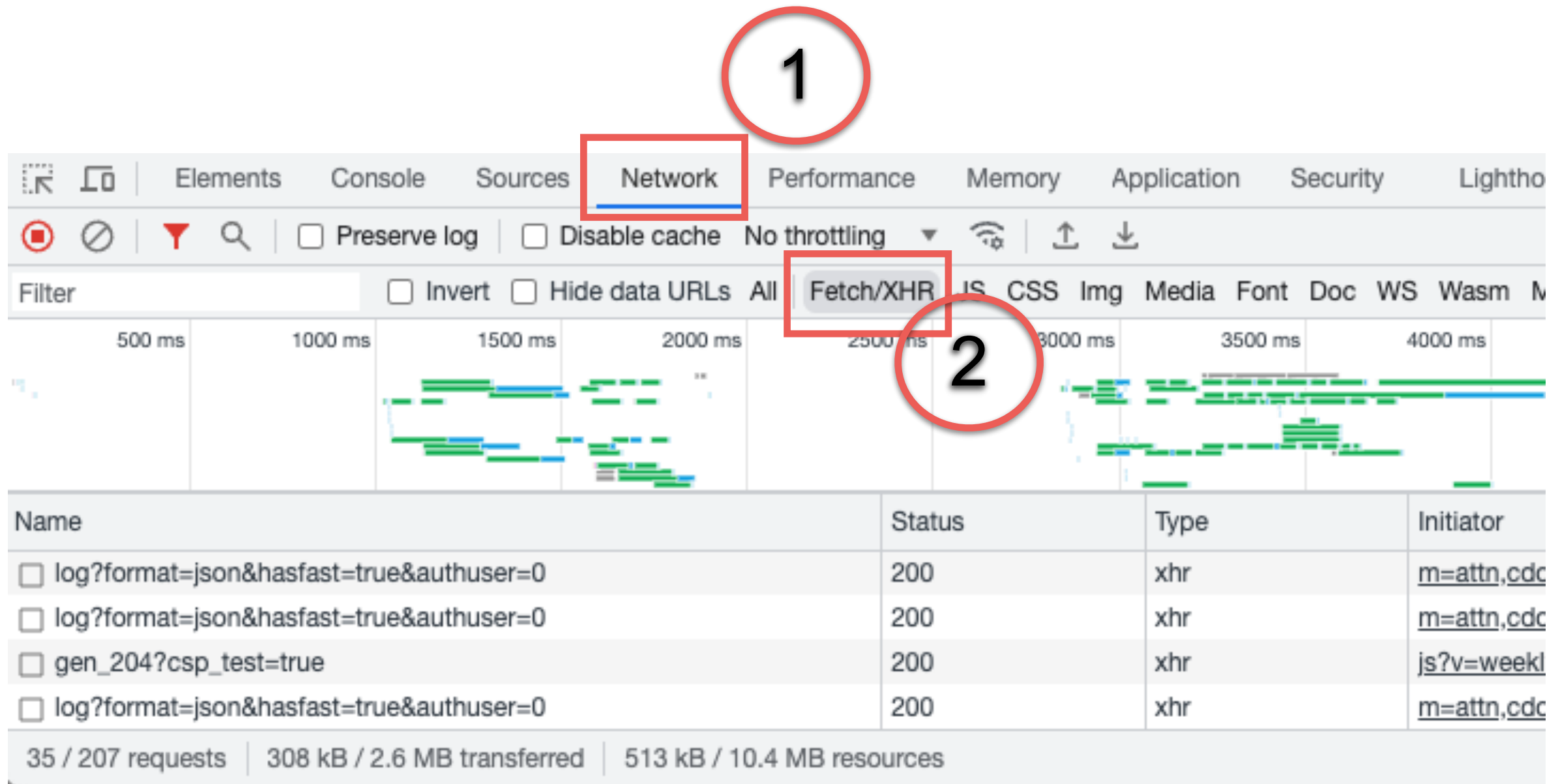


```
Specs
2
x
--
0
--
google cy.js 00:06
BEFORE EACH
1 visit https://www.google.com
(xhr) GET 200 /complete/search?
q&cp=0&client=gws-
wiz&xssi=t&gs_pcrt=2&hl=th&authuser=0&psi=w66FZP
bvC47e1sQP7piDiAE.1686482628184&dpr=1&nolsbt=1
(xhr) GET
/xjs/_/js/md=1/k=xjs.s.th.U_t0TRFb7oQ.0/am=CAAAI
AAgGoRTABtAgAAAAQACBAAAAAGAEYAAgeJQRAAAAIcFgE
MQBAwAJJQAAAAAQj9EAAAAAEDBAAAgEIAMGgIKAACAAAAI
H8ADHgBAAYTFgAAAAAEMASBIMbJEbBAAGAAAAAFTJ5MWBEA/rs=ACT90oH8enprrgo0hL3aQYwdV860zDlpJQ
(xhr) POST 200 https://play.google.com/log?
format=json&hasfast=true&authuser=0
(xhr) GET 204 /client_204?cs=1&opi=89978449
```



XHR in Google Chrome

1



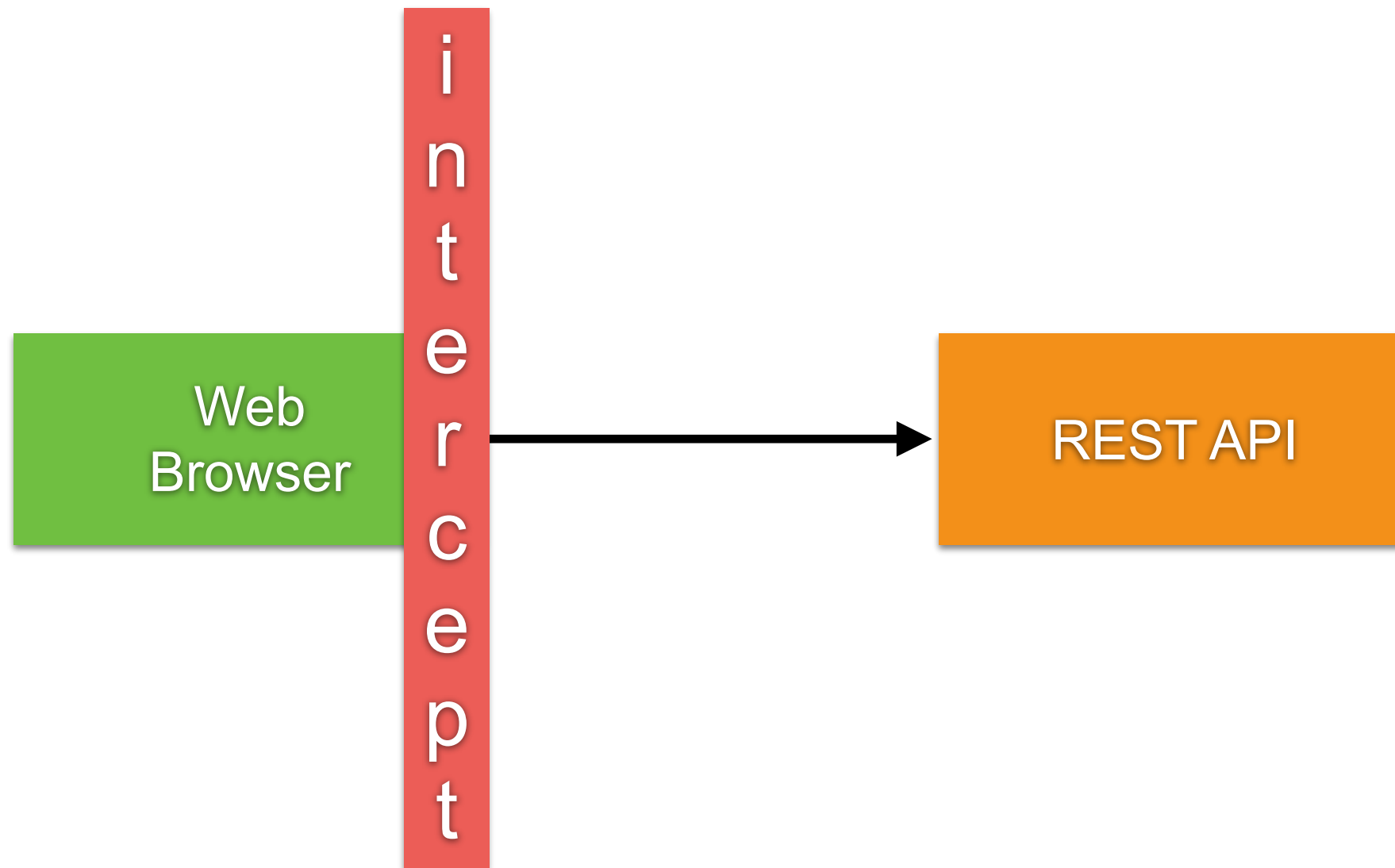
Name	Status	Type	Initiator
<input type="checkbox"/> log?format=json&hasfast=true&authuser=0	200	xhr	m=attn,cdc
<input type="checkbox"/> log?format=json&hasfast=true&authuser=0	200	xhr	m=attn,cdc
<input type="checkbox"/> gen_204?csp_test=true	200	xhr	js?v=weekl
<input type="checkbox"/> log?format=json&hasfast=true&authuser=0	200	xhr	m=attn,cdc

35 / 207 requests | 308 kB / 2.6 MB transferred | 513 kB / 10.4 MB resources



XHR with Cypress

Use network request + intercept in cypress
Stub out the backend from test



<https://docs.cypress.io/guides/guides/network-requests>



Use cases for XHR with cypress

Assert request header/body

Stub response header/body

Delay response

Waiting for response

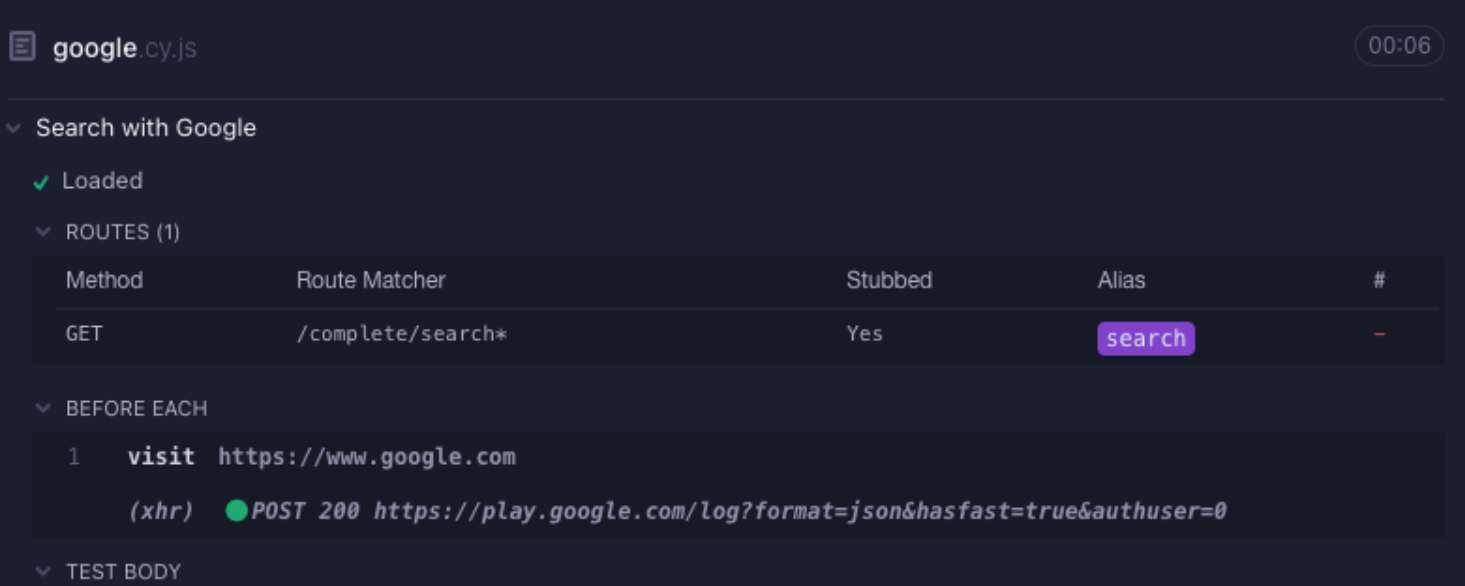
<https://docs.cypress.io/guides/guides/network-requests>



XHR with Cypress

Update file /e2e/google.cy.js

```
describe('Search with Google', () => {  
  
  beforeEach(() => {  
    // Arrange  
    cy.intercept(  
      {  
        method: 'GET',  
        pathname: '/complete/search*',  
      },  
      {  
        body: []  
      }  
    ).as('search')  
  })  
})
```



The screenshot shows the Cypress test runner interface for the file `google.cy.js`. The test suite is titled "Search with Google" and is marked as "Loaded". Under the "ROUTES (1)" section, a table lists the intercepted routes:

Method	Route Matcher	Stubbed	Alias	#
GET	/complete/search*	Yes	search	-

Below the routes, the "BEFORE EACH" section shows the test steps:

- 1 visit `https://www.google.com`
(xhr) ● POST 200 `https://play.google.com/log?format=json&hasfast=true&authuser=0`

The "TEST BODY" section is currently empty.



Routing

```
describe('Search with Google', () => {
```

```
  beforeEach(() => {
```

```
    cy.intercept(  
      {  
        method: 'GET',  
        pathname: '/complete/search*',  
      },  
      {  
        body: []  
      }  
    ).as('search')
```

```
  })
```



Fixtures

Set of data located in file (/fixtures)

```
describe('Search with Google', () => {  
  beforeEach(() => {  
    cy.intercept(  
      {  
        method: 'GET',  
        pathname: '/complete/search*',  
      },  
      {  
        fixture: 'data.json',  
      }  
    ).as('search')  
  })  
})
```



Waiting

Wait for request and response

```
describe('Search with Google', () => {  
  beforeEach(() => {  
    cy.intercept(  
      {  
        method: 'GET',  
        pathname: '/complete/search*',  
      },  
      {  
        fixture: 'data.json',  
      }  
    ).as('search')  
  })  
  
  it('Loaded', () => {  
    cy.wait(['@search'])  
  })  
})
```



Delay response ?

Update file /e2e/google.cy.js

```
describe('Search with Google', () => {  
  beforeEach(() => {  
    // Arrange  
    cy.intercept(  
      {  
        method: 'GET',  
        pathname: '/complete/search*',  
        {  
          body: [],  
          delay: 2000  
        }  
      })  
    ).as('search')  
    cy.visit('https://www.google.com')  
  })  
})
```



GenericStaticResponse

```
export interface GenericStaticResponse<Fixture, Body> {  
  /**  
   * Serve a fixture as the response body.  
   */  
  fixture?: Fixture  
  /**  
   * Serve a static string/JSON object as the response body.  
   */  
  body?: Body  
  /**  
   * HTTP headers to accompany the response.  
   * @default {}  
   */  
  headers?: { [key: string]: string }  
  /**  
   * The HTTP status code to send.  
   * @default 200  
   */  
  statusCode?: number  
  /**  
   * If 'forceNetworkError' is truthy, Cypress will destroy the browser connection  
   * and send no response. Useful for simulating a server that is not reachable.  
   * Must not be set in combination with other options.  
   */  
  forceNetworkError?: boolean  
  /**  
   * Kilobytes per second to send 'body'.  
   */  
  throttleKbps?: number  
  /**  
   * Milliseconds to delay before the response is sent.  
   */  
  delay?: number  
}
```



Run in command-line



Run

`$npx cypress run`

By default, run with Electron + headless

<https://docs.cypress.io/guides/guides/launching-browsers>



Run

Browser	Run in command-line
Electron	<code>cypress run</code> <code>cypress run --headed</code>
Google Chrome	<code>cypress run --browser chrome</code> <code>cypress run --browser chrome --headed</code>
Microsoft Edge	<code>cypress run --browser edge</code> <code>cypress run --browser edge --headed</code>
Firefox	<code>cypress run --browser firefox</code> <code>cypress run --browser firefox --headed</code>

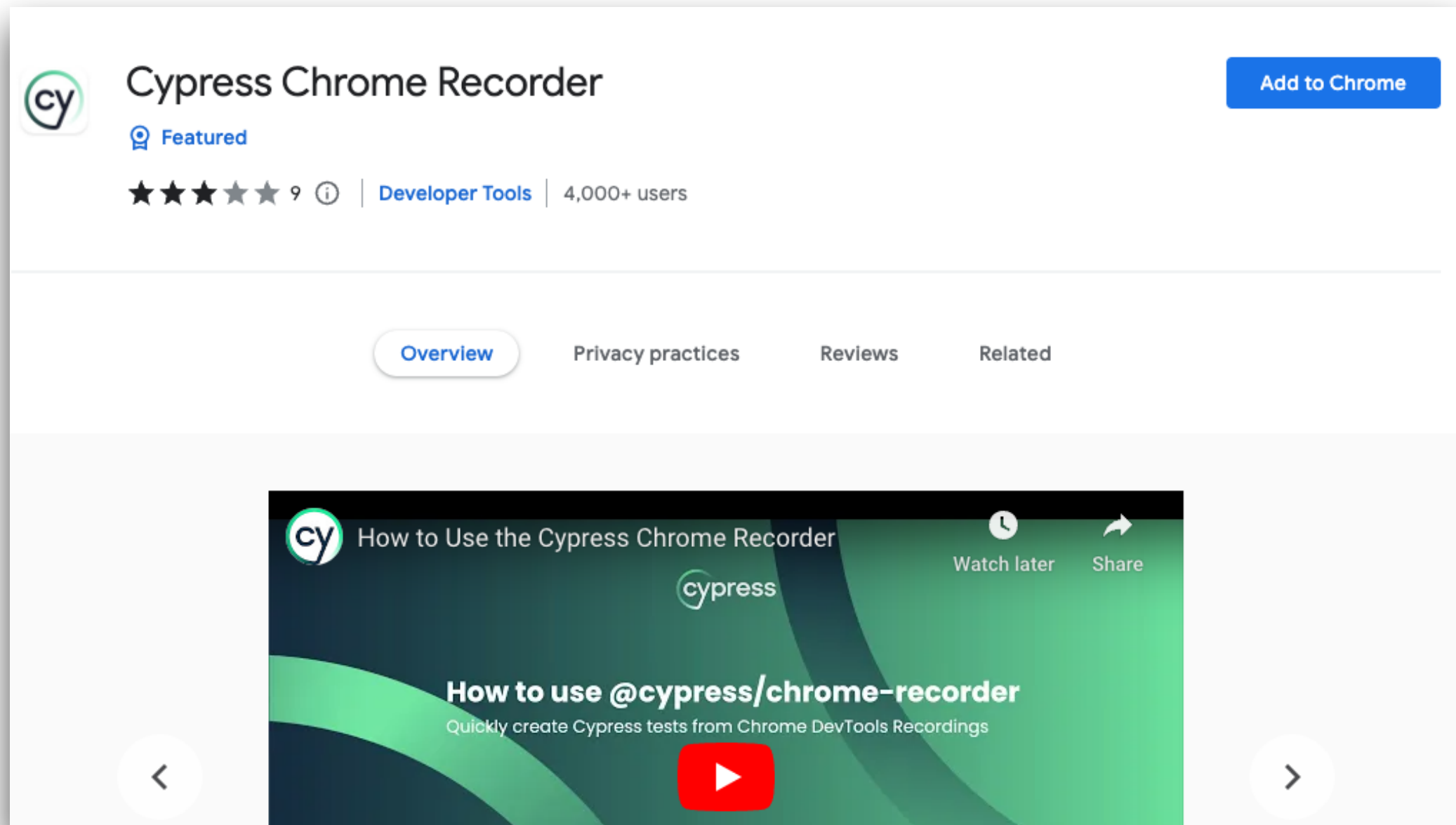
<https://github.com/cypress-io/cypress/blob/develop/packages/server/lib/browsers/chrome.ts#L36>



Export cypress test from Chrome 103+



Cypress Chrome Recorder



<https://github.com/cypress-io/cypress-chrome-recorder>



Cypress Chrome Recorder

Elements Console Sources Network Performance Memory Application Security Lighthouse Recorder × Performance insights 1 7 17

+ | Recording 6/11/2023 at 1... |

● Recording 6/11/2023 at 11:13:33 PM Performance panel Replay

Replay settings ▸ No throttling | Timeout: 5000 ms Environment Desktop | 1512×365 px

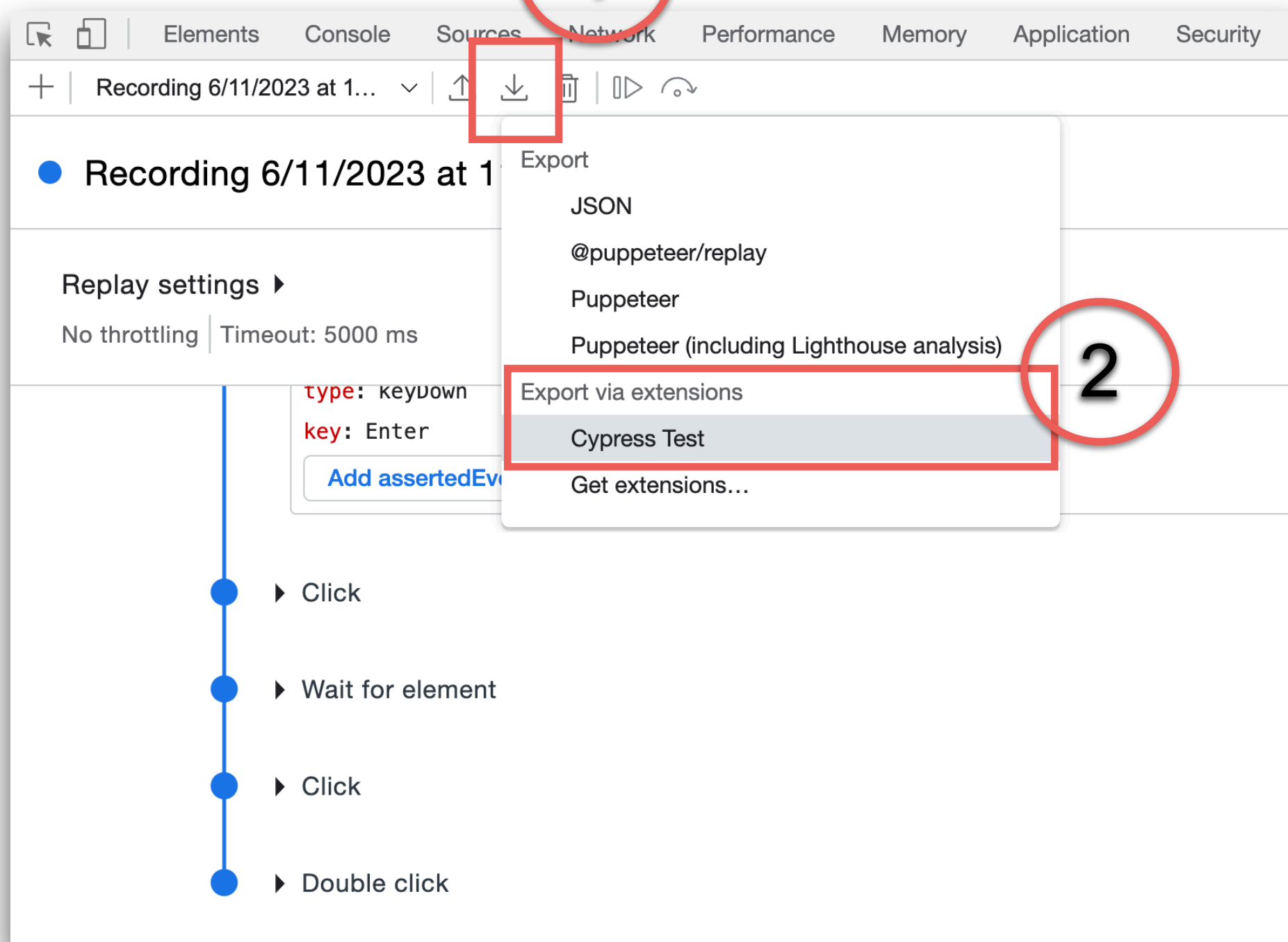
type: keyDown
key: Enter

[Add assertedEvents](#) [Add target](#) [Add timeout](#)

- ▶ Click
- ▶ Wait for element
- ▶ Click
- ▶ Double click



Cypress Chrome Recorder



<https://github.com/cypress-io/cypress-chrome-recorder>



Try by yourself

<https://demo-login-workshop.vercel.app>

1

Design test case

2

Write your test
case



Start with design test cases



Test Cases

Case name	Input		Expected Result
	Username	Password	
Login success	demo	mode	Show welcome page
Login fail case A	demo	mode2	Show error page
Login fail case B	demo2	mode	Show error page
Login fail case C	demo2	mode2	Show error page



Data Driven Testing

Use data from **fixtures**
Read data from test cases

Login-fails.json

Test case



Login success

```
describe('template spec', () => {  
  it('passes', () => {  
    // Arrange  
    cy.visit('https://demo-login-workshop.vercel.app')  
  
    // Act  
    cy.get('#username_field').should('be.visible').type('demo')  
    cy.get('#password_field').should('be.visible').type('mode')  
    cy.get('#login_button').should('be.visible').click()  
  
    // Assert  
    cy.contains('Login succeeded. Now you can logout.')  
      .should('be.visible')  
  
    cy.get('#container > h1').contains('Welcome Page').should('be.visible')  
    cy.get('#container > p').contains('Login succeeded. Now you can  
logout.').should('be.visible')  
  })  
})
```



Login fail

```
describe('Login fail spec', () => {
  it('Fail with wrong username', () => {
    // Arrange
    cy.visit('https://demo-login-workshop.vercel.app')

    // Act
    cy.get('#username_field').should('be.visible').type('demo2')
    cy.get('#password_field').should('be.visible').type('mode')
    cy.get('#login_button').should('be.visible').click()

    // Assert
    cy.get('#container > h1').contains('Error Page').should('be.visible')
    cy.get('#container > p').contains('Login failed. Invalid user name and/or password.').should('be.visible')
  })
})
```



Login-fails.json

List of test datas

```
[  
  {  
    "test_case": "Fail with wrong username",  
    "username": "demo2",  
    "password": "mode"  
  },  
  {  
    "test_case": "Fail with wrong password",  
    "username": "demo",  
    "password": "mode2"  
  }  
]
```



Test case

Read data from fixtures

```
import datas from "../fixtures/login-fails.json"
```

```
describe('Login fail spec', () => {
```

```
  datas.forEach( data => {
```

```
    it(data.test_case, () => {
```

```
      // Arrange
```

```
      cy.visit('https://demo-login-workshop.vercel.app')
```

```
      // Act
```

```
      cy.get('#username_field').should('be.visible').type(data.username)
```

```
      cy.get('#password_field').should('be.visible').type(data.password)
```

```
    })
```

```
  })
```

```
})
```



Remove duplication to command

```
import datas from '../fixtures/login-fails.json'

describe('Login fail spec', () => {

  datas.forEach( data => {

    it(data.test_case, () => {
      // Arrange
      cy.visit('https://demo-login-workshop.vercel.app')

      // Act
      cy.get('#username_field').should('be.visible').type(data.username)
      cy.get('#password_field').should('be.visible').type(data.password)
      cy.get('#login_button').should('be.visible').click()
    })

  })

})
```



Create a new command

File /support/commands.js

```
Cypress.Commands.add('login', (username, password) => {  
  cy.get('#username_field').should('be.visible').type(username)  
  cy.get('#password_field').should('be.visible').type(password)  
  cy.get('#login_button').should('be.visible').click()  
})
```

Use command from test case

```
it(data.test_case, () => {  
  // Act  
  cy.login(data.username, data.password)  
})
```



Add attribute for test with cypress ?

data-cy, **data-test**, data-testid



Add data-test attribute

```
<table>
  <tr>
    <td><label for="username_field">User Name:</label></td>
    <td><input id="username_field" data-test="username_field" size="30" type="text">
    </td>
  </tr>
  <tr>
    <td><label for="password_field">Password:</label></td>
    <td><input id="password_field" data-test="password_field" size="30" type="password">
    </td>
  </tr>
  <tr>
    <td>&nbsp;</td>
    <td><input id="login_button" data-test="login_button" type="submit" value="LOGIN">
    </td>
  </tr>
</table>
```



Create command with test-data

File /support/commands.js

```
Cypress.Commands.add('getBySel', (selector, ...args) => {  
  return cy.get(`[data-test=${selector}]`, ...args)  
})
```

```
Cypress.Commands.add('login', (username, password) => {  
  cy.getBySel('username_field').should('be.visible').type(username)  
  cy.getBySel('password_field').should('be.visible').type(password)  
  cy.getBySel('login_button').should('be.visible').click()  
})
```



Manage secret/sensitive data ?



Secret/Sensitive Data

Credential
User and password



Working with Environment Variables



Environment Variables ?

Different across developer machines
different across multiple environments
Change frequently and are highly dynamic

<https://docs.cypress.io/guides/guides/environment-variables>



Environment Variables ?

Config in cypress.config.js

Config in cypress.env.json

Use CYPRESS_*

Use --env in cypress cli

<https://docs.cypress.io/guides/guides/environment-variables>



Example

Config **baseUrl** in cypress.config.js

```
const { defineConfig } = require("cypress");

module.exports = defineConfig({
  e2e: {
    baseUrl: 'https://www.google.com',
    setupNodeEvents(on, config) {
      // implement node event listeners here
    },
  },
});
```



Use in test case

```
describe('Search with Google', () => {  
  beforeEach(() => {  
    // Arrange  
    cy.intercept(  
      {  
        method: 'GET',  
        pathname: '/complete/search*',  
      },  
      {  
        body: []  
      }  
    ).as('search')  


cy.visit('/')

  
  })  
})
```



Add more env

```
const { defineConfig } = require("cypress");  
  
module.exports = defineConfig({  
  env: {  
    data: 'hello demo',  
  },  
  
  e2e: {  
    baseUrl: 'https://www.google.com',  
    setupNodeEvents(on, config) {  
      // implement node event listeners here  
    },  
  },  
});
```



Use in test case

```
/// <reference types="cypress" />

describe('Search with Google', () => {

  it('Search', () => {
    // Act
    cy.get('textarea[name="q"]')

    cy.get('[name="q"]').type(Cypress.env('data'))

    cy.get('[name="q"]').type('{enter}')
    // Assert
    cy.get('#result-stats')
    cy.contains('#result-stats', 'ผลการค้นหาประมาณ')
  })
})
```



Run in command line

```
$CYPRESS_baseUrl= ... cypress run  
$cypress run --config baseUrl=...
```

<https://docs.cypress.io/guides/guides/environment-variables>



Working with Tags in test case



Tag in test case (only with cloud)

Grouping test cases

```
describe('My test suite', () => {  
  it('My test case', () => {  
    // tag: smoke  
    // Test code here  
  });  
});
```

\$cypress run --tag smoke

<https://docs.cypress.io/guides/guides/command-line#cypress-run-tag-lt-tag-gt>



Cypress grep

Filter tests using substring

```
$npm i -D @cypress/grep  
$cypress run --env grepTags=login
```

<https://github.com/bahmutov/cy-grep>



Config cypress grep

Support/e2e.js

```
// Import commands.js using ES2015 syntax:  
import './commands'
```

```
// Alternatively you can use CommonJS syntax:  
// require('./commands')
```

```
require('@bahmutov/cy-grep')()
```



Add tags in test case

```
describe('template spec', { tags: ['login'] }, () => {  
  it('passes', { tags: ['login'] }, () => {  
    // Arrange  
    cy.visit('https://demo-login-workshop.vercel.app')
```

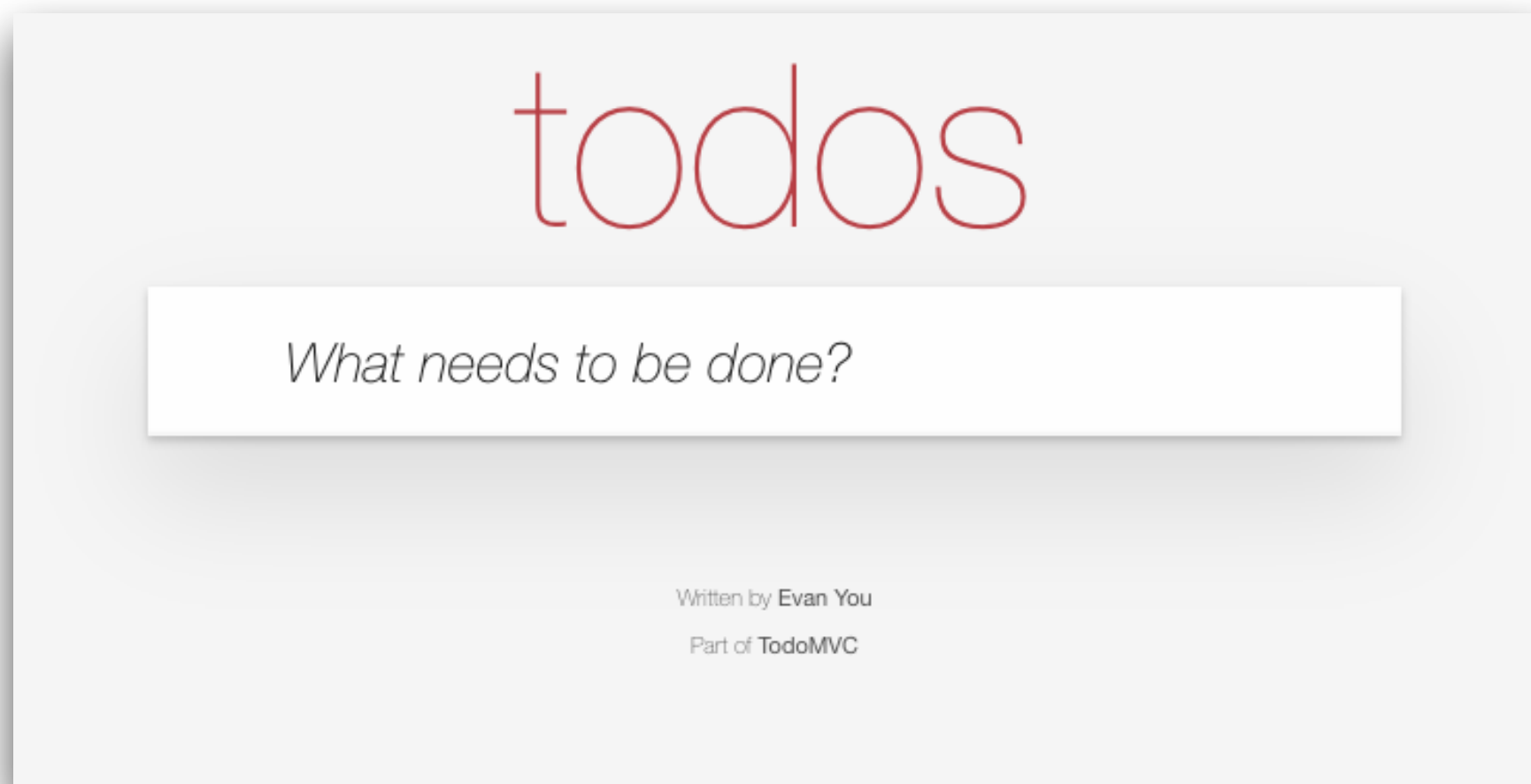


Write a test case with Cypress TODOs app



Start TODO app

```
$npm install  
$npm start
```



<http://localhost:3000/>



First test case

Create file /e2e/demo.cy.js

```
/// <reference types="cypress" />

describe('Example to-do app', () => {

  it('first page', () => {
    // Arrange
    cy.visit('http://localhost:3000')

    // Act

    // Assert
    cy.get('.new-todo').get('footer')
    cy.contains('h1', 'todos')
    cy.contains('h1', /^todos$/)
    cy.contains('[data-cy=app-title]', 'todos')
  })
})
```



Better to use web element

Use data-* attributes for web element

```
/// <reference types="cypress" />
```

```
describe('Example to-do app', () => {
```

```
  it('first page', () => {
```

```
    // Arrange
```

```
    cy.visit('http://localhost:3000')
```

```
    // Act
```

```
    // Assert
```

```
    cy.contains('[data-cy=app-title]', 'todos')
```

```
  })
```

<https://docs.cypress.io/guides/references/best-practices#Selecting-Elements>



Run test

```
$npmx cypress open  
$npmx cypress run
```

<https://docs.cypress.io/guides/guides/command-line>



Cypress Test Dashboard

The screenshot displays the Cypress Test Dashboard interface. On the left, the 'Specs' panel shows a test file 'hello.cy.js' with a duration of 85ms. The test suite is 'Example to-do app', and the current test is 'first page'. The test body contains the following commands:

```
1 visit http://localhost:3000
2 get .new-todo
3 get footer
4 -contains h1, todos
5 -contains h1, /^todos$/
6 -contains [data-cy=app-title], todos
```

Below the commands, the network log shows three successful GET requests to /todos:

```
(xhr) GET 200 /todos
(xhr) GET 200 /todos
(xhr) GET 200 /todos
```

On the right, the browser window shows the 'todos' application. The title is 'todos' in red. The input field contains the text 'What needs to be done?'. There are three radio buttons, each followed by the text 'test'. At the bottom, it says '3 items left' and has three tabs: 'All', 'Active', and 'Completed'. The footer of the application says 'Written by Evan You' and 'Part of TodoMVC'.



Test case ?

todos

What needs to be done?

Written by Evan You

Part of TodoMVC



Design Test Cases ?

Add a task
Add multiple tasks ?
Update status



Let's workshop !!



Cypress best practices

<https://github.com/cypress-io/cypress-example-recipes>



Cypress best practices

Organizing tests

Control stage

Select web elements

Use `after` or `afterEach` hooks to cleanup state

Unnecessary waiting



Cypress test patterns

Page object pattern

Custom commands

App actions

<https://www.cypress.io/blog/2019/01/03/stop-using-page-objects-and-start-using-app-actions/>



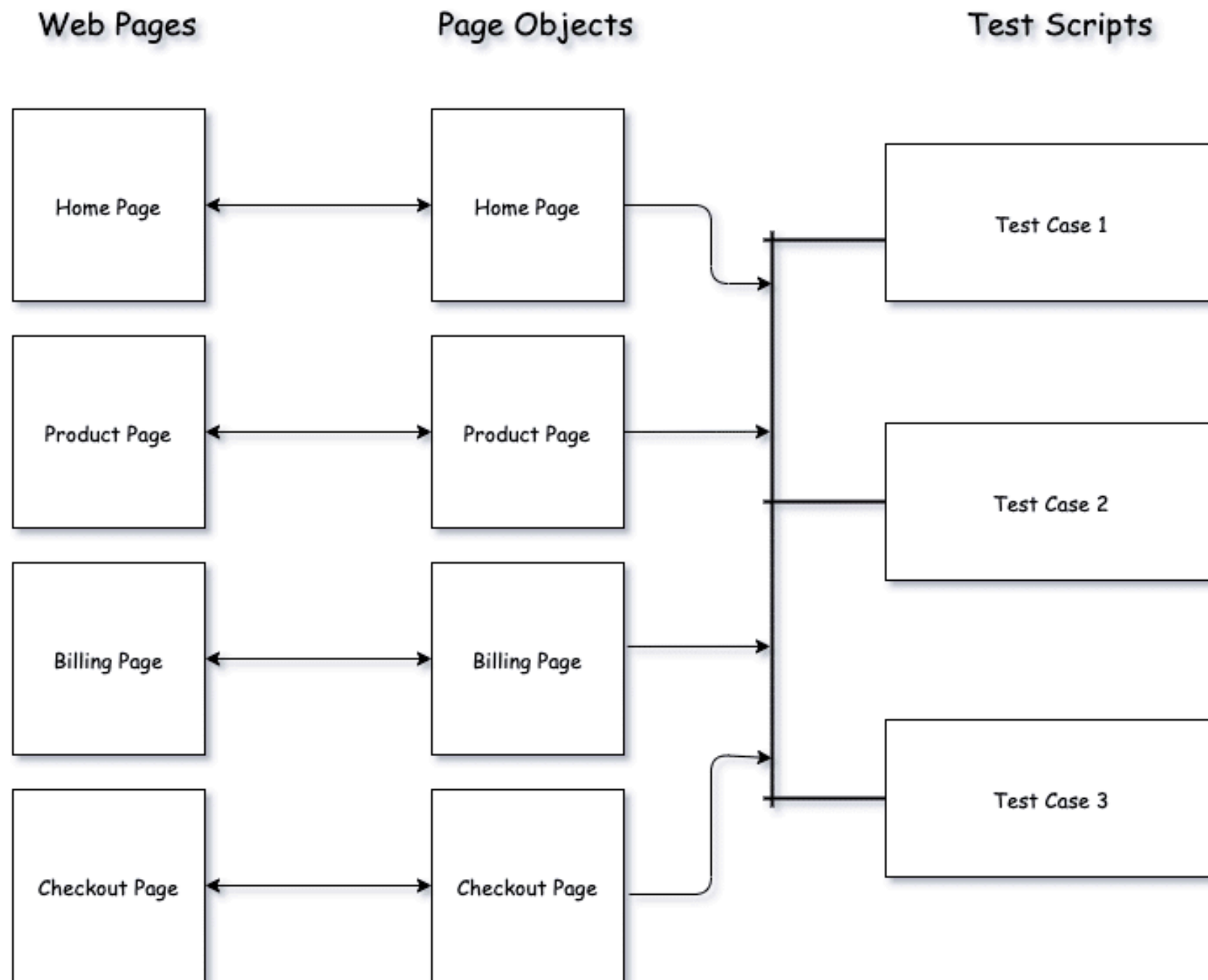
Page object pattern

Page Object Model is a ***design pattern*** in the automation world which has been famous for its test maintenance approach and avoiding code duplication

<https://www.toolsqa.com/cypress/page-object-pattern-in-cypress/>



Page object pattern



Custom commands

Reuse logic into command in cypress
Not state

```
Cypress.Commands.add('login', (username, password) => {  
  cy.get('#login-username').type(username)  
  cy.get('#login-password').type(password)  
  cy.get('#login').submit()  
})
```

```
it('logs in', () => {  
  cy.visit('/login')  
  cy.login('username', 'password')  
})
```

<https://docs.cypress.io/api/cypress-api/custom-commands>



Cypress App actions

Cypress allows interacting with the application under the test directly **using the application logic**. This makes the application testing **faster** and more **stable** as it bypasses the UI element interaction.

<https://www.browserstack.com/guide/how-to-use-cypress-app-actions>



Cypress Testing Library

Cypress Testing Library



Simple and complete custom Cypress commands and utilities that encourage good testing practices.

[Read the docs](#) | [Edit the docs](#)

<https://testing-library.com/docs/cypress-testing-library/intro/>



Workshop

