



Installation and Setup

Installation

Install Cypress via npm:

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

Install Cypress via yarn:

```
yarn add cypress --dev
```

Ensure Node.js and npm/yarn are installed before running the commands.

Project Structure

```
cypress/
├── fixtures/      # Static data for
tests
├── integration/   # Test files
├── plugins/       # Browser
automation and config modifications
├── support/       # Reusable custom
commands
```

Understanding the project layout helps organize and maintain tests.

Running Cypress

Open Cypress GUI:

```
npx cypress
open
```

Launches the Cypress Test Runner with a graphical interface.

Run Cypress tests in headless mode:

```
npx cypress
run
```

Executes tests from the command line without the GUI.

Writing Your First Test

Describe / It Syntax

`describe()`: Groups related tests together.
`it()`: Defines an individual test case.

```
describe('My First Test', () => {
  it('Visits the Kitchen Sink', () => {
    // test steps
  })
})
```

Visiting Pages

Visit a URL:

```
cy.visit('https://
/example.cypress.
io')
```

Navigates the browser to the specified URL.

Basic Assertions

Assert the content of an element:

```
cy.get('h1').should('contain', 'Example
Domain')
```

Verifies that the `h1` element contains the text 'Example Domain'.

Element Selectors and DOM Interaction

Element Selectors

`cy.get(selector)`: Selects elements based on a CSS selector.

`cy.contains(content)`: Selects elements containing the specified text.

`cy.find(selector)`: Searches within a previously selected element.

```
cy.get('.my-class')
cy.contains('Submit')
cy.get('#my-form').find('input')
```

DOM Interaction

`cy.click()`: Clicks an element.

`cy.type(text)`: Types text into an input field.

`cy.check()`: Checks a checkbox or radio button.

`cy.select(value)`: Selects an option from a dropdown.

```
cy.get('button').click()
cy.get('input').type('Hello, World!')
cy.get('input[type="checkbox"]').check()
cy.get('select').select('Option 1')
```

Custom Data Attributes

Using `data-*` attributes provides stable and maintainable selectors.

```
<button data-cy="submit-
button">Submit</button>
```

```
cy.get('[data-cy="submit-
button"]').click()
```

Assertions and Commands

Assertions

`should(chainer, value)`: Makes assertions about the state of the subject.

`expect(value).to.be.equal(value)`: BDD-style assertion.

`and(chainer, value)`: Alias for `should()` to improve readability.

```
cy.get('input').should('have.value',
'initial value')
expect(true).to.be.true
cy.get('input').and('have.attr',
'required')
```

Command Chaining

Cypress commands are chainable, allowing for concise and readable tests.

```
cy.get('input')
.type('my-username')
.should('have.value', 'my-username')
```

Retry-ability

Cypress commands are automatically retried until assertions pass or a timeout is reached. This makes tests more resilient to timing issues.