# Tugas Assertions



Berikut selengkapnya :

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

// describe ('Browser actions', () => {

//     it('Should load correct url', () => {

//         cy.visit('http://example.com')

//         {timeout:10000}

//     });

//     it('Should check url', () => {

//         cy.url().should('include','http://example.com')

//         it('Should check correct url on the page', () => {

//             cy.get('h1').should('be.visible')

//         });

//     });

// })

describe('My First Test', () => {

    it('Clicking"type" show the right headings', () => {

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

      cy.get('h1').contains('Kitchen Sink')

      cy.contains('type').click()

      cy.url().should('include','/commands/actions')

      cy.get('.action-email')

        .type('fake@email.com')

        .should('have.value', 'fake@email.com')

        // .type() with special character sequences

        .type('{leftarrow}{rightarrow}{uparrow}{downarrow}')

        .type('{del}{selectall}{backspace}')

        // .type() with key modifiers

        .type('{alt}{option}') //these are equivalent

        .type('{ctrl}{control}') //these are equivalent

        .type('{meta}{command}{cmd}') //these are equivalent

        .type('{shift}')

        // Delay each keypress by 0.1 sec

        .type('slow.typing@email.com', { delay: 100 })

        .should('have.value', 'slow.typing@email.com')

      cy.get('.action-disabled')

        // Ignore error checking prior to type

        // like whether the input is visible or disabled

        .type('disabled error checking', { force: true })

        .should('have.value', 'disabled error checking')

cy.get('.action-focus').focus()

  .should('have.class', 'focus')

  .prev().should('have.attr', 'style', 'color: orange;')

cy.get('.action-blur').type('About to blur').blur()

  .should('have.class', 'error')

  .prev().should('have.attr', 'style', 'color: red;')

  cy.get('.action-clear').type('Clear this text')

  .should('have.value', 'Clear this text')

  .clear()

  .should('have.value', '')

  cy.get('.action-form')

  .find('[type="text"]').type('HALFOFF')

cy.get('.action-form').submit()

  .next().should('contain', 'Your form has been submitted!')

  cy.get('.action-btn').click()

// clicking in the center of the element is the default

cy.get('#action-canvas').click()

cy.get('#action-canvas').click('topLeft')

cy.get('#action-canvas').click('top')

cy.get('#action-canvas').click('topRight')

cy.get('#action-canvas').click('left')

cy.get('#action-canvas').click('right')

cy.get('#action-canvas').click('bottomLeft')

cy.get('#action-canvas').click('bottom')

cy.get('#action-canvas').click('bottomRight')

// .click() accepts a an x and y coordinate

// that controls where the click occurs :)

cy.get('#action-canvas')

  .click(80, 75)

  .click(170, 75)

  .click(80, 165)

  .click(100, 185)

  .click(125, 190)

  .click(150, 185)

  .click(170, 165)

// click multiple elements by passing multiple: true

cy.get('.action-labels>.label').click({ multiple: true })

// Ignore error checking prior to clicking

cy.get('.action-opacity>.btn').click({ force: true })

// By default, .uncheck() will uncheck all matching

// checkbox elements in succession, one after another

cy.get('.action-check [type="checkbox"]')

  .not('[disabled]')

  .uncheck().should('not.be.checked')

// .uncheck() accepts a value argument

cy.get('.action-check [type="checkbox"]')

  .check('checkbox1')

  .uncheck('checkbox1').should('not.be.checked')

// .uncheck() accepts an array of values

cy.get('.action-check [type="checkbox"]')

  .check(['checkbox1', 'checkbox3'])

  .uncheck(['checkbox1', 'checkbox3']).should('not.be.checked')

// Ignore error checking prior to unchecking

cy.get('.action-check [disabled]')

  .uncheck({ force: true }).should('not.be.checked')

  // at first, no option should be selected

cy.get('.action-select')

.should('have.value', '--Select a fruit--')

// Select option(s) with matching text content

cy.get('.action-select').select('apples')

// confirm the apples were selected

// note that each value starts with "fr-" in our HTML

cy.get('.action-select').should('have.value', 'fr-apples')

cy.get('.action-select-multiple')

.select(['apples', 'oranges', 'bananas'])

// when getting multiple values, invoke "val" method first

.invoke('val')

.should('deep.equal', ['fr-apples', 'fr-oranges', 'fr-bananas'])

// Select option(s) with matching value

cy.get('.action-select').select('fr-bananas')

// can attach an assertion right away to the element

.should('have.value', 'fr-bananas')

cy.get('.action-select-multiple')

.select(['fr-apples', 'fr-oranges', 'fr-bananas'])

.invoke('val')

.should('deep.equal', ['fr-apples', 'fr-oranges', 'fr-bananas'])

// assert the selected values include oranges

cy.get('.action-select-multiple')

.invoke('val').should('include', 'fr-oranges')

cy.get('#scroll-horizontal button')

  .should('not.be.visible')

// scroll the button into view, as if the user had scrolled

cy.get('#scroll-horizontal button').scrollIntoView()

  .should('be.visible')

cy.get('#scroll-vertical button')

  .should('not.be.visible')

// Cypress handles the scroll direction needed

cy.get('#scroll-vertical button').scrollIntoView()

  .should('be.visible')

cy.get('#scroll-both button')

  .should('not.be.visible')

// Cypress knows to scroll to the right and down

cy.get('#scroll-both button').scrollIntoView()

  .should('be.visible')

  // if you chain .scrollTo() off of cy, we will

// scroll the entire window

cy.scrollTo('bottom')

cy.get('#scrollable-horizontal').scrollTo('right')

// or you can scroll to a specific coordinate:

// (x axis, y axis) in pixels

cy.get('#scrollable-vertical').scrollTo(250, 250)

// or you can scroll to a specific percentage

// of the (width, height) of the element

cy.get('#scrollable-both').scrollTo('75%', '25%')

// control the easing of the scroll (default is 'swing')

cy.get('#scrollable-vertical').scrollTo('center', { easing: 'linear' })

// control the duration of the scroll (in ms)

cy.get('#scrollable-both').scrollTo('center', { duration: 2000 })

cy.get('.trigger-input-range')

  .invoke('val', 25)

  .trigger('change')

  .get('input[type=range]').siblings('p')

  .should('have.text', '25')

     })

    })