1. \*\*cy.visit(url)\*\* — переход на указанную страницу.

cy.visit('https://example.com');

2. \*\*cy.get(selector)\*\* — получение элемента по селектору.

cy.get('#username');

3. \*\*cy.contains(text)\*\* — получение элемента, который содержит указанный текст.

cy.contains('Login');

4. \*\*cy.click()\*\* — клик по элементу.

cy.get('button.submit').click();

5. \*\*cy.type(text)\*\* — ввод текста в элемент (например, в текстовое поле).

cy.get('input[name="password"]').type('mypassword');

6. \*\*cy.clear()\*\* — очистка содержимого элемента.

cy.get('input[name="username"]').clear();

7. \*\*cy.submit()\*\* — отправка формы.

cy.get('form').submit();

8. \*\*cy.should(condition)\*\* — проверка условия (например, видимость элемента).

cy.get('.success-message').should('be.visible');

9. \*\*cy.wait(time)\*\* — ожидание указанного времени.

cy.wait(2000); // ожидание 2 секунды

10. \*\*cy.intercept()\*\* — перехват сетевых запросов для их тестирования.

cy.intercept('GET', '/api/users', { fixture: 'users.json' }).as('getUsers');

11. \*\*cy.fixture()\*\* — загрузка фикстуры для использования в тестах.

cy.fixture('users.json').then((users) => {

// использование данных из фикстуры

});

12. \*\*cy.reload()\*\* — перезагрузка текущей страницы.

13. \*\*cy.screenshot()\*\* — создание скриншота текущего состояния.

cy.screenshot('homepage');

14. \*\*cy.viewport(width, height)\*\* — изменение размера окна браузера.

cy.viewport(1280, 720); // установка размера окна

15. \*\*cy.request(options)\*\* — выполнение HTTP-запроса.

cy.request('POST', '/api/login', { username: 'user', password: 'pass' });

16. \*\*cy.wrap(subject)\*\* — обертка вокруг объекта для выполнения команд Cypress.

cy.wrap(user).should('have.property', 'name', 'John');

17. \*\*cy.then(callback)\*\* — выполнение функции после завершения предыдущей команды.

cy.get('.item').then(($item) => {

// работа с элементом

const text = $item.text();

expect(text).to.equal('Expected Text');

});

18. \*\*cy.each(callback)\*\* — перебор элементов и выполнение функции для каждого из них.

cy.get('.list-item').each(($el) => {

cy.wrap($el).should('be.visible');

});

19. \*\*cy.invoke(method, args)\*\* — вызов метода на элементе.

cy.get('input[name="username"]').invoke('val').should('equal', 'expectedValue');

20. \*\*cy.spy()\*\* — шпион для отслеживания вызовов функции.

const myFunction = () => {};

cy.spy(window, 'myFunction').as('myFunctionSpy');

cy.get('button').click();

cy.get('@myFunctionSpy').should('have.been.called');

Метод should() принимает один или несколько аргументов, которые определяют, что именно нужно проверить. Вот основные моменты о его использовании:

Синтаксис

**cy.get('selector').should('assertion')**

1. Проверка видимости элемента

cy.get('.my-element').should('be.visible');

Проверяет, что элемент с классом .my-element видим на странице.

2. Проверка наличия элемента

cy.get('.my-element').should('exist');

Проверяет, что элемент с классом .my-element существует в DOM.

3. Проверка текста элемента

cy.get('.my-element').should('have.text', 'Expected Text');

Проверяет, что текст внутри элемента соответствует ожидаемому значению.

4. Проверка значения атрибута

cy.get('input[name="username"]').should('have.attr', 'type', 'text');

Проверяет, что элемент input с именем username имеет атрибут type со значением text.

5. Проверка наличия класса

cy.get('.my-element').should('have.class', 'active');

Проверяет, что элемент имеет класс active.

6. Проверка состояния элемента

cy.get('input[type="checkbox"]').should('be.checked');

Проверяет, что чекбокс отмечен.

7. Проверка стиля элемента

cy.get('.my-element').should('have.css', 'color', 'rgb(255, 0, 0)');

Проверяет, что цвет текста элемента соответствует красному.

8. Проверка количества элементов

cy.get('.my-element').should('have.length', 3);

Проверяет, что на странице есть три элемента с классом .my-element.