Cypress

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

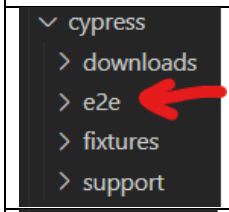
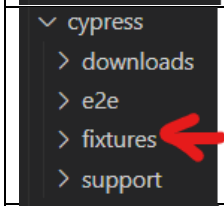
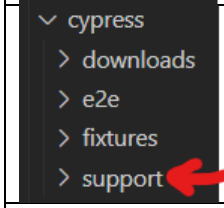
Встановлення та запуск

Встановлення	`npm install cypress --save-dev`
Для форматування додати у секцію `"extends"` `.eslintrc.json`	`{` `  "extends": [` `  . . . . . .` `  "plugin:cypress/recommended"` `  . . .  . . . . .` `  ]` `}`
Запуск	Є декілька варіантів: 1)npx cypress open 2) ----headless mode --- **npx cypress run** 3)-- head mode (with opening browser) -- **npx cypress run --browser chrome** 4) -- run some spec-- **npx cypress run --spec "шлях_до_файлу.spec.js"**

Додавання та запуск з використанням скриптів

Додаємо команду у розділ скриптів файлу `package.json`.	`{` `  "scripts": {` `  "cypress:open": "cypress open"` `  }` `}`
Зараз запуск можна робити з використанням скриптів	`npm run cypress:open`

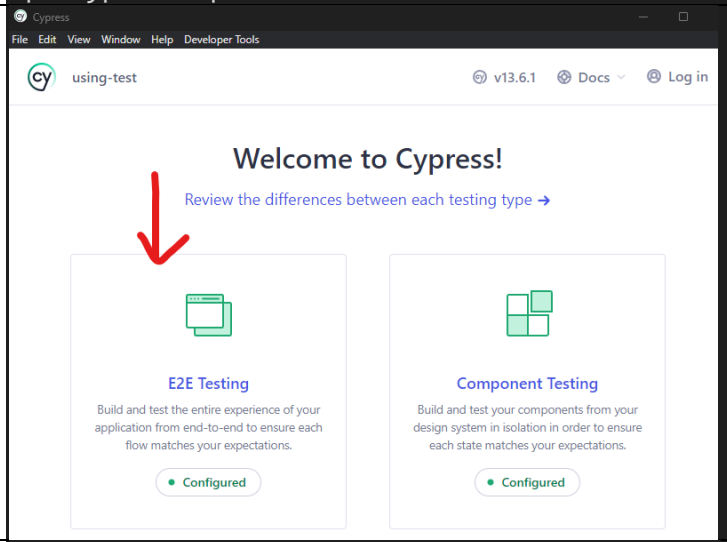
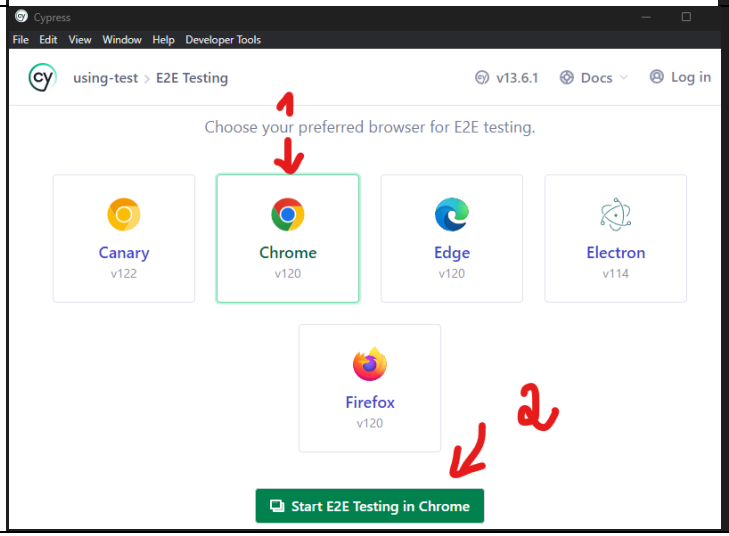
Структура папок

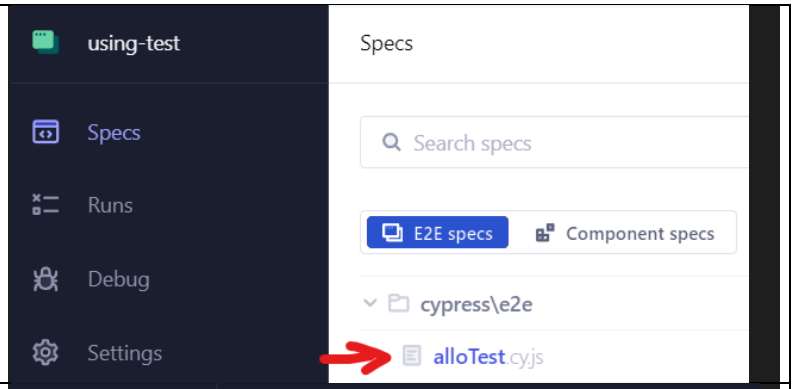
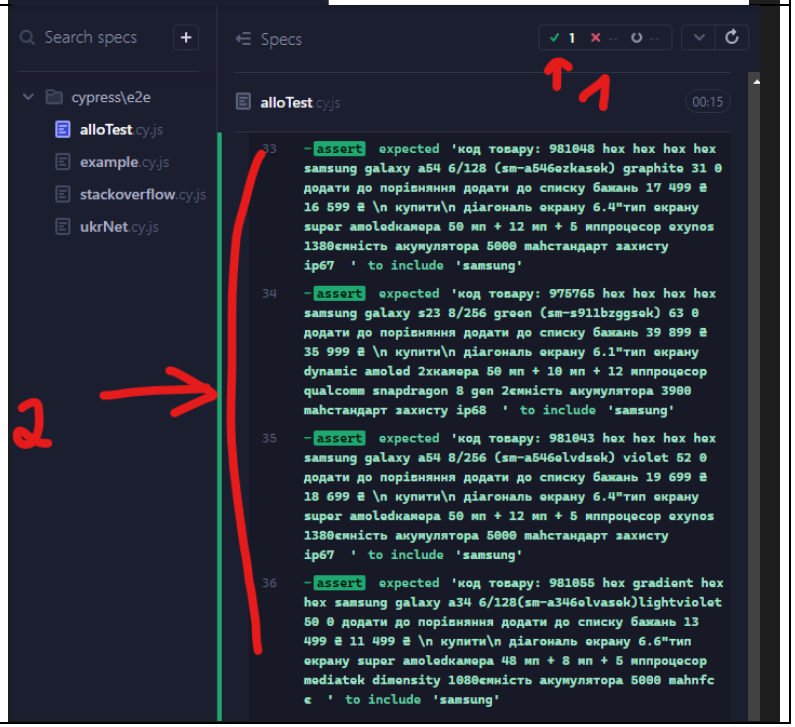
Папка	Призначення
	e2e - тести
	папка, де зберігають тестові дані для підміни, тобто зразків даних, які можуть бути використані в тестах
	тут описуємо додаткові власні команди користувача
plugins	Папка plugins використовується для встановлення розширень та налаштувань Cypress.
screenshots	Папка screenshots - місце для автоматичного збереження знімків екрану у разі невдалих тестів.
videos	Папка videos - тут можна зберігати відеозаписи тестів, якщо вони включені у налаштуваннях Cypress.

Загальна схема <u>e2e тестів</u>

Кроки	***Загальна схема можливих прикладів команд***	***Приклади команд*** ***- - - - - /cypress/e2e/alloTest.cy.js - - - - -***
		`/* eslint-disable cypress/unsafe-to-chain-command */` `describe('Allo Search Test', () => {` `  it('should filter products when searching for Samsung', () => {`
Відвідування сторінки	`cy.visit('шлях_до_сторінки')`	`    // 1. Відвідуємо головну сторінку` `    cy.visit('https://allo.ua')`
Імітація дій користувача	`cy.get('селектор').click()` `cy.get('селектор').type('текст');` `. . . . .`	`    // 2. У полі пошуку вводимо Samsung` `    cy.get('#search-form__input').type('Samsung').type('{enter}')`
Перевірка реакції	`cy.url().should('include', '/новий_шлях')` `cy.get('селектор').should('contain', 'новий_результат')` `. . . . .`	`    // Зачекайте на завантаження результатів пошуку` `    cy.get('.products-layout__container').should('be.visible')` `    // 3. Перевіряємо, чи відбулась фільтрація` `    cy.get('[data-product-id]')` `      .should('be.visible') // Перевірка видимості товарів` `      .each(($title) => {` `        expect($title.text().toLowerCase()).to.include('samsung')` `      }) // Перевірка, що кожен товар має слово Samsung у назві`
		`  })` `})`

Запуск тесту

Запускаємо cypress	`npx cypress open`
Вибираємо e2e тестування	
Вибираємо браузер для тестування	

Вибираємо тест для запуску	
Аналізуємо тести: 1)скільки пройшло/не_пройшло 2)етапи проходження тесту	

Ще один приклад

Кроки	***Загальна схема можливих прикладів команд***	***Приклади команд***
Відвідування сторінки	`cy.visit('шлях_до_сторінки')`	`cy.visit('/login')`
Імітація дій користувача	`cy.get('селектор').click()` `cy.get('селектор').type('текст');`	`cy.get('#username').type('testUser')` `cy.get('#password').type('testPassword')` `cy.get('button[type="submit"]').click()`
Перевірка реакції	`cy.url().should('include', '/новий_шлях')`	`cy.url().should('include', '/dashboard')`

	cy.get('селектор').should('contain', 'новий_результат')	cy.get('.welcome-message').should('contain', 'Welcome, testUser!')

Вибірка елементів для взаємодії

Пошук за селектором	cy.get(css_селектор)	// Пошук кнопки за класом cy.get('.my-button'); // Пошук елемента за ідентифікатором cy.get('#my-element'); // Пошук за тегом cy.get('input');
Пошук за вмістом .contains(content) .contains(content, options) .contains(selector, content) .contains(selector, content, options) // ---or--- cy.contains(content) cy.contains(content, options) cy.contains(selector, content) cy.contains(selector, content, options)	cy.contains(критерій 1, внутрішній критерій 2, …)	// Пошук кнопки з текстом "Submit" cy.contains('Submit').click(); // Пошук елемента, що містить текст "Hello, World" cy.contains('div', 'Hello, World'); // yields bananas cy.contains(/^b\w+/)
пошук дочірнього елемента .find(selector) .find(selector, options)	предок .find()	// Пошук кнопки з класом в межах елемента з класом "parent" cy.get('.parent').find('.child-button'); // Пошук інпута в межах форми cy.get('form').find('input');
Пошук предка для вказаного дочірнього за допомогою parents()	дочірній. parents(. . .)	// Пошук першого предка елемента з класом "child" з класом "parent" cy.get('.child').parents('.parent:first'); // Пошук усіх предків елемента з класом "child" cy.get('.child').parents();

Введення даних

Команда	***Приклад***
Введення тексту за допомогою **.type**(). Можна використовувати спецкоманди “{enter} {backspace} {del} {esc} …” https://docs.cypress.io/api/commands/type	// Введення тексту в поле введення з ідентифікатором "username" cy.get('#username').type('john_doe'); // Введення тексту в поле введення за допомогою класу cy.get('.password-input').type('securepassword{enter}');
Виклик методу **.blur**():	// Виклик події "blur" на полі вводу з ідентифікатором "username" cy.get('#username').blur();
Виклик методу **.focus**():	// Виклик події "focus" на полі вводу з ідентифікатором "search" cy.get('#search').focus();
Очищення вмісту елемента за допомогою **.clear**():	// Очищення вмісту поля вводу перед введенням нового тексту cy.get('#message').clear().type('New message');
Встановлення чекбокса за допомогою **.check**():	// Встановлення чекбокса з ідентифікатором "agree" cy.get('#agree').check();
Зняття відмітки з чекбокса за допомогою **.uncheck**():	// Зняття відмітки з чекбокса з ідентифікатором "subscribe" cy.get('#subscribe').uncheck();
Вибір значення з випадаючого списку за допомогою **.select**():	// Вибір значення "Option 2" з випадаючого списку з ідентифікатором "dropdown" cy.get('#dropdown').select('Option 2');
Подвійний клік на елемент за допомогою **.dblclick**():	// Подвійний клік на елементі з класом "double-clickable" cy.get('.double-clickable').dblclick();
Контекстне меню за допомогою **.rightclick**():	// Відкриття контекстного меню для елемента з ідентифікатором "context-menu-item" cy.get('#context-menu-item').rightclick();
Відправлення форми за допомогою **.submit**():	// Відправлення форми з ідентифікатором "login-form" cy.get('#login-form').submit();

Перевірка стану (`.should()`)

https://docs.cypress.io/api/commands/should

існуванню елемента:	// Перевірка, що елемент з ідентифікатором "logo" існує на сторінці cy.get('#logo').**should('exist');**
видимості елемента:	// Перевірка, що елемент з класом "visible-element" видимий на сторінці cy.get('.visible-element').**should('be.visible');**
значенню елемента:	// Перевірка, що значення поля вводу з ідентифікатором "username" дорівнює "john_doe" cy.get('#username').**should('have.value', 'john_doe');**
стану елемента (активний/неактивний, вибраний/не вибраний):	// Перевірка, що чекбокс з ідентифікатором "subscribe" вибраний cy.get('#subscribe').**should('be.checked');**
стилі елемента (CSS):	// Перевірка, що елемент з класом "styled-element" має стиль "color" рівний "red" cy.get('.styled-element').**should('have.css', 'color', 'red');**
По довжині елементів:	// Перевірка, що кількість елементів з класом "list-item" дорівнює 3 cy.get('.list-item').**should('have.length', 3);**
По класу елемента:	// Перевірка, що елемент з ідентифікатором "my-element" має клас "active" cy.get('#my-element').**should('have.class', 'active');**
Негативні перевірки (не має класу, не видимий і т.д.):	// Перевірка, що елемент з ідентифікатором "error-message" не має класу "hidden" cy.get('#error-message').**should('not.have.class', 'hidden');** // Перевірка, що елемент з класом "hidden-element" не видимий на сторінці cy.get('.hidden-element').**should('not.be.visible');**
Використання колбек-функції в should:	// Перевірка, що текст в елементі з ідентифікатором "message" починається з "Welcome" cy.get('#message').**should(**($el) => { expect($el.text()).to.startWith('Welcome'); });
Кілька перевірок одночасно:	// Перевірка, що елемент з ідентифікатором "status" має клас "success" і видимий cy.get('#status'**).should**('have.class', 'success').**and**('be.visible');

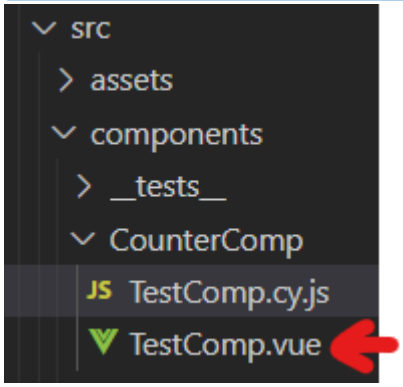
Робота з списками

Перегляд кількості елементів:	// Перевірка, що кількість елементів з класом "list-item" дорівнює 5 cy.get('.list-item').**should('have.length', 5);**
Вибір елемента за індексом у списку: `.eq( індекс )`	// Вибір третього елемента з класом "list-item" cy.get('.list-item'**).eq(2).**should('contain', 'Item 3');
Використання `.each()` для ітерації по списку:	// Перевірка кожного елемента з класом "list-item" cy.get('.list-item').**each**(($item, index) => { cy.wrap($item).should('contain', `Item ${index + 1}`); });
Фільтрація елементів за текстовим критерієм:	// Перевірка, що є елемент з класом "list-item" і текстом "Item 2" cy.get('.list-item').**contains('Item 2').**should('exist');
Використання .filter() для фільтрації елементів:	// Припустимо, що на сторінці є список елементів з класом "list-item", // і ми хочемо вибрати лише ті, які мають клас "highlighted". cy.get('.list-item').**filter**('.highlighted').should('have.length', 3); // Ще один приклад: фільтрація за вмістом тексту. // Виберемо елементи, які містять текст "Item 2". cy.get('.list-item').**filter**(':contains("Item 2")').should('have.length', 1); // Фільтрація за допомогою функції зворотного виклику. // Виберемо ті елементи, які мають клас "special" або "highlighted". cy.get('.list-item').**filter**(($el) => { return $el.hasClass('special') || $el.hasClass('highlighted'); }).should('have.length', 4); // Вибір тільки парних елементів з класом "list-item" cy.get('.list-item').**filter**(':even').should('have.length', 3);
Використання .first() та .last() для отримання першого і останнього елементів:	// Вибір першого елемента з класом "list-item" cy.get('.list-item').**first**().should('contain', 'Item 1'); // Вибір останнього елемента з класом "list-item" cy.get('.list-item').**last**().should('contain', 'Item 5');

Тестування компонентів

https://docs.cypress.io/guides/component-testing/vue/overview

Маємо створений компонент




```vue
<template>
  <div>
    <div>
      <label> Num1 <input v-model="num1" type="number" /> </label>
    </div>
    <div>
      <label> Num2 <input v-model="num2" type="number" /> </label>
    </div>
    <div>Sum = {{ sum }}</div>
    <div>prod = {{ prod }}</div>
  </div>
</template>

<script setup>
import { computed, ref } from 'vue'

const num1 = ref(0)
const num2 = ref(0)

const sum = computed(() => num1.value + num2.value)
const prod = computed(() => num1.value * num2.value)
</script>

<style lang="scss" scoped></style>
```

Створюємо файл з тестом, назву якого створюємо за шаблоном «назва_компонента».cy.js

1)Імпортуємо компонент

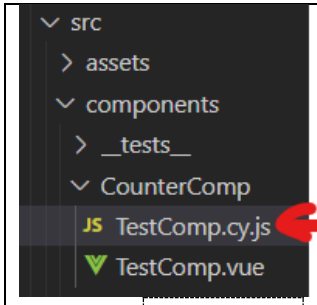
```
/* eslint-disable cypress/unsafe-to-chain-command */
import TestComp from './TestComp.vue'
```

2)Описуємо групу тестів

```
describe('<TestComp />', () => {
```

3)Монтуємо компонент

```
  beforeEach(() => {
    cy.mount(TestComp)
  })
```



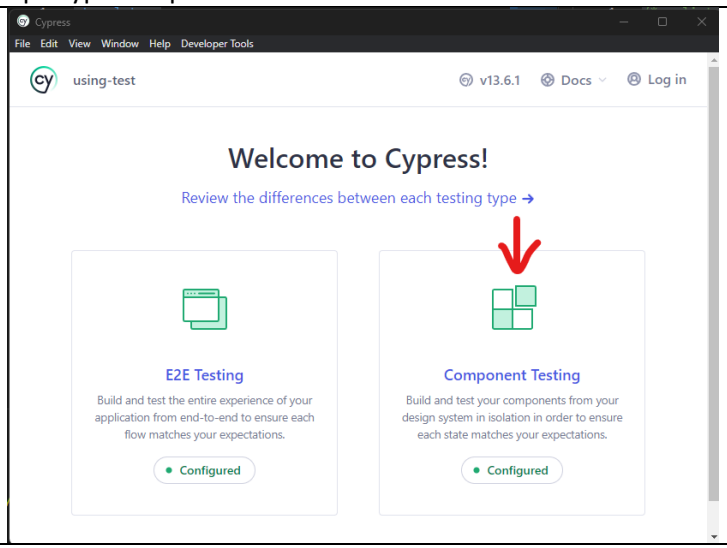

```
import компонент from 'шлях'


describe( ‘назва_групи_тетів’,
()=>
  {
     it('назва_тесту_1', () => {
      cy.mount( компонент )
       . . . перевірка елементів
...
     }
  }
)
```

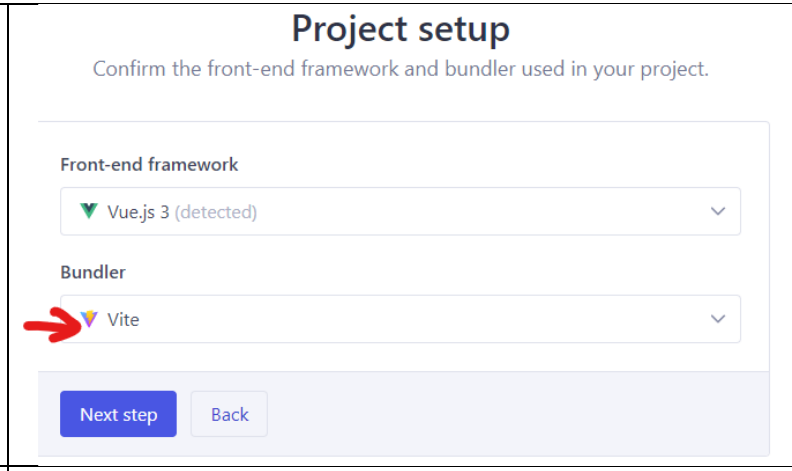
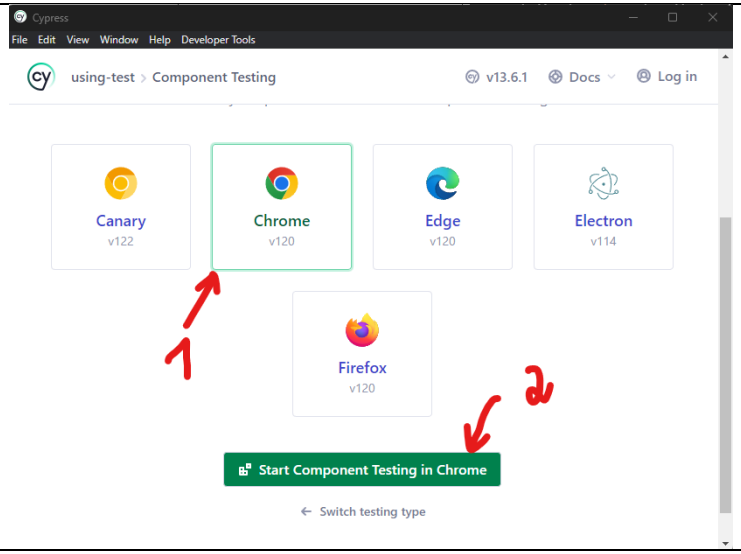
4)Додаємо тести

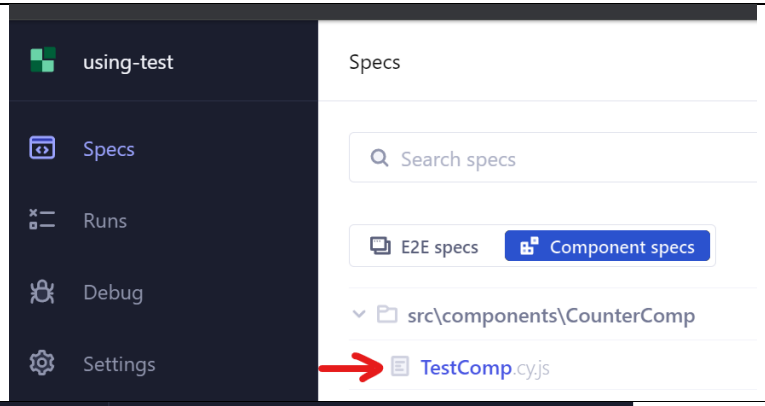
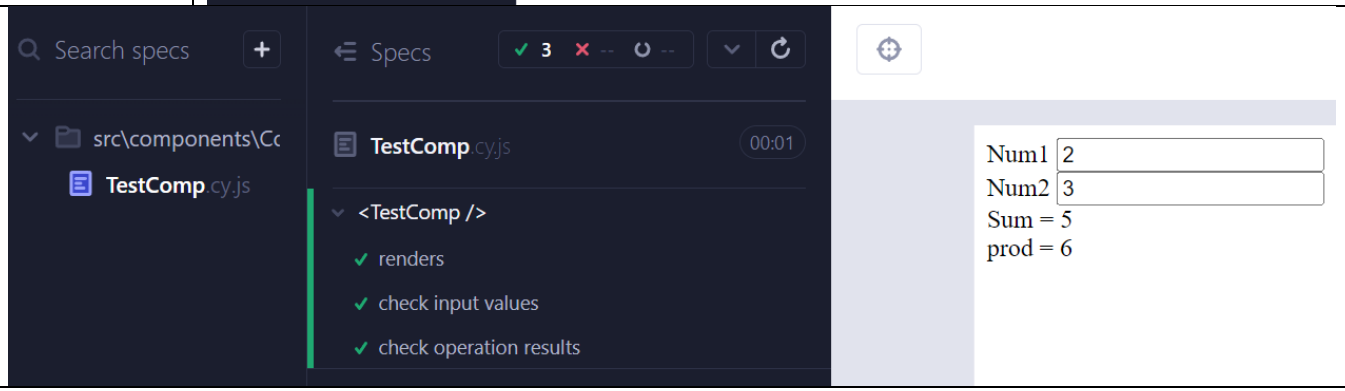
```
  it('renders', () => {
    cy.get('label').contains('Num1').should('exist')
  })
```

```
it('check input values', () => {
    cy.get('input').eq(0).clear()
    cy.get('input').eq(0).type('2asdasd{enter}')
    cy.get('input').eq(0).should('have.value', '2')
    cy.get('input').eq(1).type('3{enter}')
  })
  it('check operation results', () => {
    cy.get('input').eq(0).clear().type('2{enter}')
    cy.get('input').eq(1).clear().type('3{enter}')
    cy.get('div').contains('5').should('exist')
    cy.get('div').contains('6').should('exist')
  })
})
```

Запуск тесту компонента

Запуск		npx cypress open
Вибираємо “Component testing”		

		
Вибираємо браузер, у якому буде здійснено тестування		

Вибираємо файл тесту для виконання		
Аналізуємо результати		

Встановлення параметрів компонента (props)

```
describe( ‘назва_групи_тетів’,
()=>
  {
     it('назва_тесту_1', () => {
      cy.mount( компонент ,
     {
      props: {
        власт.1: знач.1,
        власт.2: знач.2,
        . . . . . .
      }
     }
)
      . . . перевірка елементів
...
     }
  }
)
```

```
/* eslint-disable cypress/unsafe-to-
chain-command */
import TestComp from './TestComp.vue'

describe('<TestComp />', () => {
  it('sending props', () => {
    cy.mount(TestComp, {
      props: {
        initNum1: 11,
        initNum2: 22
      }
    })

    cy.get('input').eq(0).should('have.
value', '11')
    cy.get('input').eq(1).should('have.
value', '22')
  })
})
```

```
<template>
  <div>
    <div>
      <label> Num1 <input v-model="num1"
type="number" /> </label>
    </div>
    <div>
      <label> Num2 <input v-model="num2"
type="number" /> </label>
    </div>
    <div>Sum = {{ sum }}</div>
    <div>prod = {{ prod }}</div>
  </div>
</template>

<script setup>
import { computed, ref } from 'vue'

const props = defineProps({
  initNum1: {
    type: Number,
    default: 0
  },
  initNum2: {
    type: Number,
    default: 0
  }
})

const num1 = ref(props.initNum1)
const num2 = ref(props.initNum2)

const sum = computed(() => num1.value +
num2.value)
const prod = computed(() => num1.value *
num2.value)
```

		</script> <style lang="scss" scoped></style>

Page Objects у Cypress

Page Object — *це шаблон проектування в тестуванні, який дозволяє вам відокремити логіку тестів від деталей реалізації інтерфейсу користувача. У Cypress використання Page Objects допомагає зробити ваш код тестів більш організованим та легше читати.*

Структура Page Object:

1. Класи Page Object: Визначте клас для кожної сторінки чи фрагмента сторінки вашого додатка.
2. Локатори елементів: Визначте всі локатори елементів, які ви використовуєте на сторінці. Це може бути окремий об'єкт чи змінна з локаторами елементів.
3. Методи для взаємодії з елементами: Створіть методи для різних дій на сторінці. Наприклад, метод для введення тексту, кліку на кнопку, перевірки стану елемента тощо.

Приклад

Клас, що містить необхідні локатори та методи для сторінки	Створення тесту з використанням класу
---------- pages/LoginPage.js -------------------- class LoginPage { **//- - - - - Локатори елементів - - - -** get usernameInput() { return cy.get('#username'); } get passwordInput() { return cy.get('#password'); } get loginButton() { return cy.get('button[type="submit"]'); } **//- - - - - Методи для взаємодії з елементами - - - -** typeUsername(username) { this.usernameInput.type(username); } typePassword(password) { this.passwordInput.type(password); }	// integration/ui/login_spec.js import LoginPage from '../../pages/LoginPage'; describe('Login functionality', () => { it('should login with valid credentials', () => { // Використання методів з Page Object LoginPage.typeUsername('myUsername'); LoginPage.typePassword('myPassword'); LoginPage.clickLoginButton(); // Додаткові перевірки // ... }); });

```
  clickLoginButton() {
    this.loginButton.click();
  }

  // Інші методи для перевірок, які вам потрібні
}

export default new LoginPage();
```