





Hello

Maciej Kucharski

Frontend/JS Developer @SentiOne

Trainer/Coach @InfoShare Academy

Kilka słów o webinarze

Kilka słów o webinarze

1. Czym są testy automatyczne
2. Kilka słów o Cypress
3. Pierwsze uruchomienie
4. Live coding
5. Q&A


Po co testować, po co szukać 
?

Po co testować, po co szukać ?

- Aplikacje są coraz bardziej rozbudowane (zawierają wiele powiązanych ze sobą funkcjonalności) przez co mogą zawierać coraz więcej błędów

Po co testować, po co szukać ?

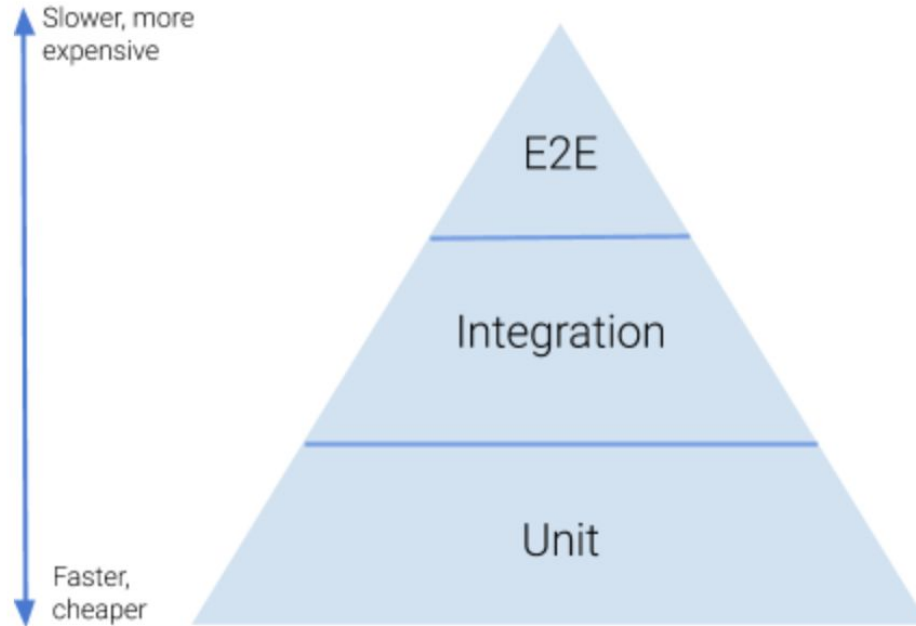
- Aplikacje są coraz bardziej rozbudowane (zawierają wiele powiązanych ze sobą funkcjonalności) przez co mogą zawierać coraz więcej błędów
- Musimy mieć pewność, że wprowadzane nowe funkcjonalności nie psują już istniejących fragmentów aplikacji

Co mogą zaoferować nam testy
automatyczne  ?

Czym są testy automatyczne

- Testy automatyczne, są to fragmenty kodu, które wykonują (np. w przeglądarce) pewne czynności dążące do sprawdzenia poprawności działania aplikacji
- Proces powstawania testów można porównać do programowania, chcemy, aby program wykonał za nas pewną czynność
- W teście musimy zaprogramować co sekwencja kodu (testu) ma wykonać (np. kliknięcie w przycisk, sprawdzenie wartości w Input)
- Praktycznie każdy test powinien zakończyć sprawdzeniem czy aktualny stan aplikacji jest tym czego oczekujemy

Testy, testy, testy



Jak może wyglądać test (w naszym języku)

1. Przejdź na stronę www.exampleapp.com
2. Kliknij w przycisk z napisem 'users'
3. Poczekaj na załadowanie strony
4. Odnajdź element typu input z id o wartości 'user-filter' i wprowadź wartość 'Jan Kowalski'
5. Kliknij przycisk z napisem 'Search'
6. Poczekaj na przefiltrowanie danych
7. Sprawdź czy pierwszy użytkownik z listy ma login o wartości 'Jan Kowalski'



cypress.io

+

e2e

=





Czym jest Cypress?

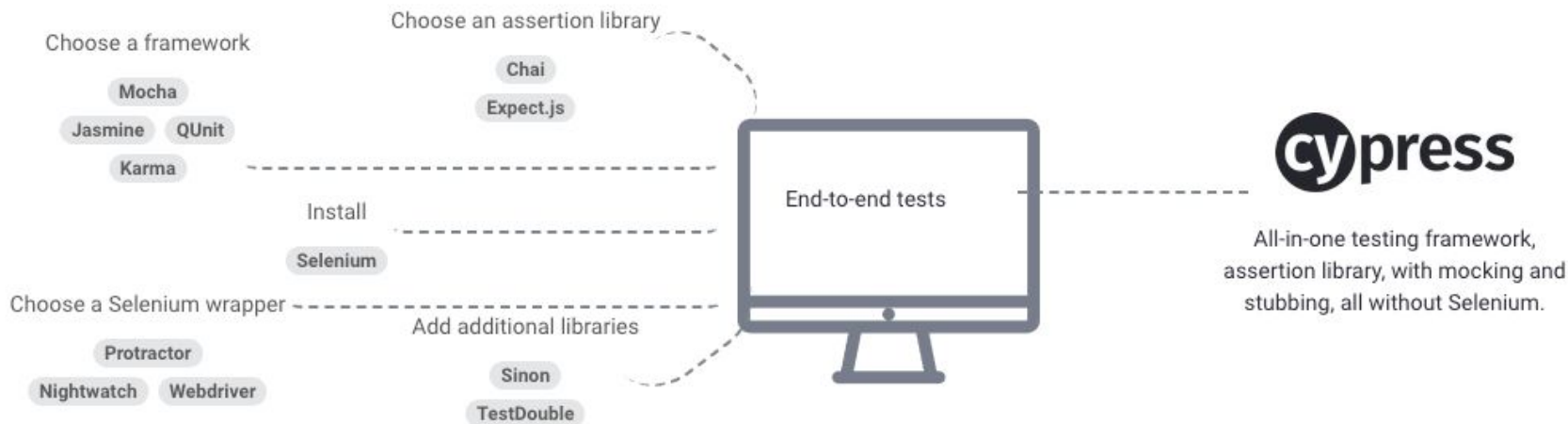
- Cypress jest to narzędzie do pisania i uruchamiania testów e2e
- Cypress nie korzysta z Selenium (jest to popularny framework do pisania testów, znany szczególnie w środowiskach Java czy C#/.net)
- Cały kod (testów i obsługi testów) piszemy w języku JavaScript
- Cypress nie wymaga praktycznie konfiguracji, praktycznie od razu po instalacji możemy zacząć pisać testy
- Wspiera wiele przeglądarek, również może działać w trybie headless (istotne dla continuous integration, np. Jenkins)

Czym jest Cypress?

Before Cypress

vs

✨ With Cypress ✨



Skoro Cypress to samo  to
sprawdźmy go w akcji!

Przygotowania do uruchomienia

Do pierwszego uruchomienia, będziemy potrzebować:




Przygotowania do uruchomienia

Dla celów prezentacyjnych mam przygotowany nowy, pusty projekt, w którym będziemy umieszczać kod naszych testów.

W przypadku produkcyjnych testów, kod może być na tym samym repo co aplikacja, którą Cypress ma testować

Przygotowania do uruchomienia

Najpierw stworzymy katalog, w którym zainicjujemy projekt npm



```
mkdir new-cypress-project
```

Następnie po przejściu do nowo utworzonego katalogu
(`cd new-cypress-project`) inicjalizujemy nowy projekt:



```
npm init -y
```

Instalacja

Cypressa można również uruchamiać jako skrypt przez npm (często preferowana opcja). Do wykonania tej czynności musimy najpierw zainstalować paczkę cypress



```
npm install cypress
```

npm jest składową Node.js i służy między innymi do instalowania zależności, uruchamiania skryptów i zarządzania zależnościami naszej aplikacji.

Instalacja

Następnie musimy dodać nowy skrypt do pliku package.json w sekcji script



```
"cypress:open": "cypress open"
```

Po dodaniu tego wpisu, sekcja script powinna wyglądać mniej więcej tak:

```
{  
  "scripts": {  
    "cypress:open": "cypress open"  
  }  
}
```

Uruchomienie 🥰

Teraz już możemy uruchamiać!



```
npm run cypress:open
```

To polecenie uruchomi okno programu Cypress

Cypress



new-cypress-test

Support Docs Log In

Chrome 81

Run all specs

Tests Runs Settings

Search...

INTEGRATION TESTS COLLAPSE ALL | EXPAND ALL

examples

- actions.spec.js
- aliasing.spec.js
- assertions.spec.js
- connectors.spec.js
- cookies.spec.js
- cypress_api.spec.js
- files.spec.js
- local_storage.spec.js
- location.spec.js
- misc.spec.js
- navigation.spec.js
- network_requests.spec.js
- querying.spec.js
- spies_stubs_clocks.spec.js
- traversal.spec.js

To help you get started...

We've added some folders and example tests to your project. Try running the tests in the **examples** folder or add your own test files to **cypress/integration**.

new-cypress-test

- ...
- + cypress
 - + fixtures
 - example.json
 - + integration
 - examples
 - actions.spec.js
 - aliasing.spec.js
 - ... 17 more files ...
 - + plugins
 - index.js
 - + support
 - commands.js
 - index.js

OK, got it!

DEMO

Q&A

Struktura przykładowego zestawu testów

```
describe('When page initially loaded ', () => {  
  it('title should be visible with proper name', () => {  
    // check value of the header  
  });  
  
  it("input should be with proper placeholder", () => {  
    // check value of the placeholder  
  });  
});
```

Operowanie na elemencie (komendy)

```
cy.visit('http://google.com');
```

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

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

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

```
cy.get('a').contains("Completed").click();
```

```
cy.get('a').eq(5).click();
```

Asercje (porównywanie wartości)

```
cy.get('li.selected').should('have.length', 3); // implicit assertion

cy.get('form').find('input').should('not.have.class', 'disabled'); // implicit assertion

cy.get('#header a') // implicit assertion
  .should('have.class', 'active')
  .and('have.attr', 'href', '/users');

cy.get('#loading').should('not.exist'); // implicit assertion

cy.get('tbody tr:first').should(($tr) => { // explicit assertion
  expect($tr).to.have.class('active');
  expect($tr).to.have.attr('href', '/users')
});

cy.get('input.new-todo').then(element => {
  expect(element).to.have.attr('placeholder', 'What needs to be done?');
})
```

DEMO

Hooks

```
describe('Hooks', () => {  
  before(() => {  
    // runs once before all tests in the block  
  });  
  
  after(() => {  
    // runs once after all tests in the block  
  });  
  
  beforeEach(() => {  
    // runs before each test in the block  
  });  
  
  afterEach(() => {  
    // runs after each test in the block  
  })  
});
```

DEMO

Q&A

Wykonywanie akcji na stronie

```
cy.get('input').click();
```

```
cy.get('input').type("Hello!");
```

```
cy.get('input').type("WoW!").type('{enter}');
```


DEMO

Q&A

Bonusik

Co dalej...

- Najlepiej po prostu zacząć testować swoją aplikację
- Obejrzeć filmik odnośnie “best practices”
https://www.youtube.com/watch?v=5XQOK0v_YRE
- Możliwe jest uruchamianie testów np. w Jenkinsie (lub innym CI)
- Zadbać o stabilne środowisko...
- ...i testować :)

Q&A

Dzięki!

Zapytaj o szkolenie dla Twojego zespołu



DOMINIKA KALINA

dominika.kalina@infohareacademy.com

tel. 730-830-801

szkoleniazdalne.infohareacademy.com

Każdy moment jest dobry na rozwój kompetencji!