





# Hello

## Maciej Kucharski

Frontend/JS Developer @SentiOne

Trainer/Coach @InfoShare Academy

# Kilka słów o webinarze

# Kilka słów o webinarze

1. O czym mówiliśmy na poprzednim webinarze?
2. Jawne asercje (Explicit assertions)
3. Dostarczanie danych do testów
4. Request / route, co jeszcze może Cypress?
5. Page Object Pattern vs App Actions

# 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?');
})
```

# 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

# Dostarczanie danych do testów

# DEMO

# Różne dane na różnych środowiskach?

# DEMO

# Zapytaj o szkolenie dla Twojego zespołu



**PRZEMYSŁAW WOŁOSZ**

[przemyslaw.wolosz@infohareacademy.com](mailto:przemyslaw.wolosz@infohareacademy.com)

tel. 600 647 729

[szkoleniazdalne.infohareacademy.com](https://szkoleniazdalne.infohareacademy.com)

**Każdy moment jest dobry na rozwój kompetencji!**

# Q&A

# Request i route



# Request i route

## Request


- Cypress posiada funkcjonalność do wykonywania zapytań HTTP,
- Możemy pisać asercje na rzeczy zwrócone z takiego zapytania,
- Możemy dostarczać danych do testów z zewnętrznych serwisów

## Route

- Służy do kontrolowania zapytań HTTP z naszej testowanej aplikacji
- Możemy 'mockować' odpowiedzi nie dotykając serwera
- Możemy pisać asercje na to jak ma wyglądać nasze zapytanie HTTP (np. czy wychodzi z dobrymi parametrami)

# DEMO

# Request i route

 Please be aware that Cypress only currently supports intercepting XMLHttpRequests. Requests using the Fetch API and other types of network requests like page loads and `<script>` tags will not be intercepted or visible in the Command Log. See [#95](#) for more details and temporary workarounds.

## Route

- Służy do kontrolowania zapytań HTTP z naszej testowanej aplikacji
- Możemy 'mockować' odpowiedzi nie dotykając serwera
- Możemy pisać asercje na to jak ma wyglądać nasze zapytanie HTTP (np. czy wychodzi z dobrymi parametrami)

# Page Object Pattern

# Page Object Pattern

- Popularny wzorzec projektowy
- Pozwala oddzielić logikę metod wchodzących w interakcje z naszą stroną od metod logiki testów
- Kod testów jest czytelniejszy
- Możliwe reużywanie funkcjonalności

# App Actions

# App Actions

- Funkcje, które modyfikują model aplikacji z pominięciem UI
- Podobnie jak przy POP jest separacja asercji od funkcjonalności przygotowującej stan aplikacji
- Programista piszący test, musi znać strukturę modelu aplikacji

# App Actions

## Plusy

- Wykonywanie akcji bez UI
- Testy mogą być znacznie szybsze
- Jest to polecane rozwiązanie przez twórców Cypressa
- Można bardziej skupić się na testowaniu funkcjonalności aplikacji

## Minusy

- Wykonywanie akcji bez UI
- Konieczna znajomość struktury aplikacji
- Aplikacja musi być przygotowana, model, lub funkcje muszą być widoczne również poza aplikacją



# Q&A

# Zapytaj o szkolenie dla Twojego zespołu



**PRZEMYSŁAW WOŁOSZ**

[przemyslaw.wolosz@infohareacademy.com](mailto:przemyslaw.wolosz@infohareacademy.com)

tel. 600 647 729

[szkoleniazdalne.infohareacademy.com](https://szkoleniazdalne.infohareacademy.com)

**Każdy moment jest dobry na rozwój kompetencji!**

# Dzięki!