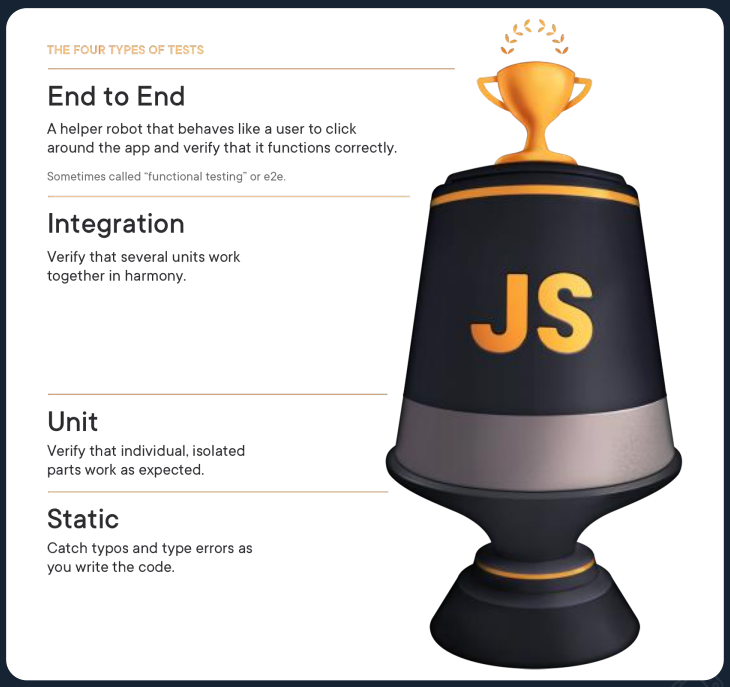
TESTY E2E Z UŻYCIEM

Dominik Nowik

RODZAJE TESTÓW

- Testy End to End
- Testy integracyjne
- Testy jednostkowe
- Analiza statyczna (eslint, prettier, Flow/Typescript)



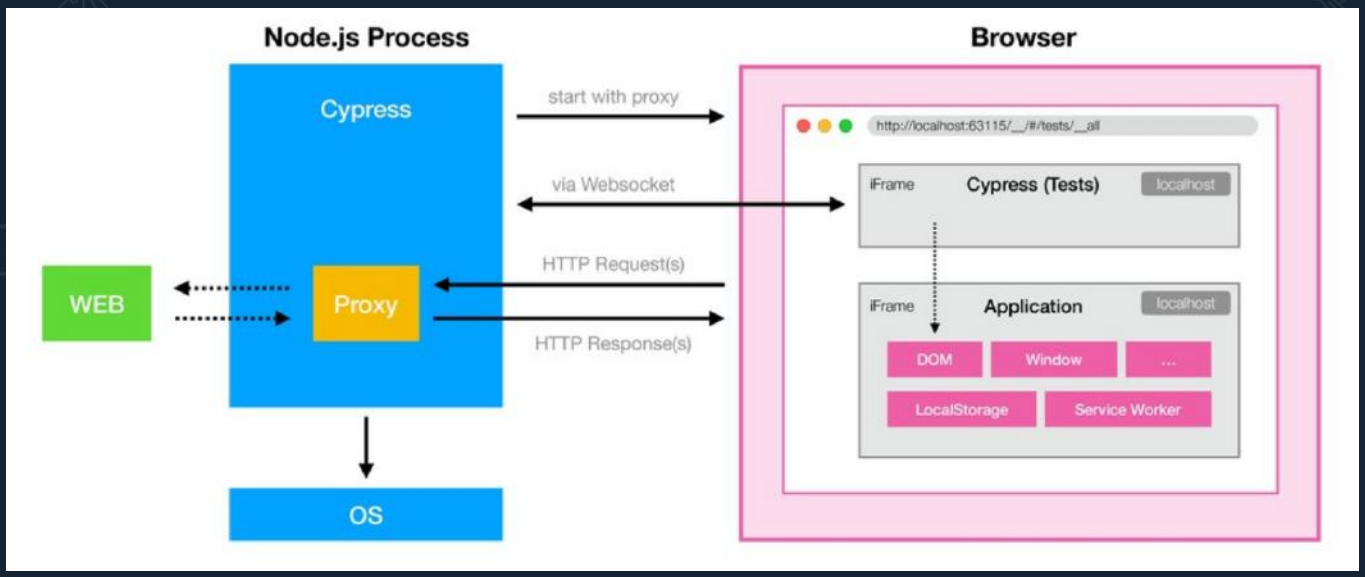
TESTY END TO END

- Automatyzujemy zachowania użytkownika na podstawie scenariuszy testowych
- Sprawdzamy aplikację 'od zewnątrz' -> na poziomie interfejsu użytkownika
- Piszemy zazwyczaj minimalną ilość testów tylko dla krytycznych miejsc aplikacji
- Długi czas wykonywania
- Wynik testu nie zawsze jest powtarzalny
- Uzyskujemy informację która funkcjonalność nie działa, sami musimy znaleźć fragment kodu
- Trudne w utrzymaniu

CYPRESS

- Kompletny framework do testowania, pozwala na pisanie przy użyciu Javascript testów każdego rodzaju
- Wiele wbudowanych narzędzi takich jak: Mocha, Chai, Sinon, **jQuery**
- Skierowany do Developerów i Inżynierów Jakości oprogramowania (QA Engineer)
- Open Sourcowy Test Runner - aplikacja pozwalająca na uruchamianie testów w przeglądarce
- Działa w oparciu o przeglądarki z rodziny Chromium: Google Chrome, Chromium, Canary, Electron
- Komercyjny Dashboard - daje wgląd w testy uruchamiane w CI(logi, nagrywane wideo, screenshoty), pozwala na równoległe uruchamianie testów, wbudowany load balancer

ARCHITEKTURA



INSTALACJA

```
yarn add --dev cypress
```

package.json:

```
"scripts": {
  "cy:run": "cypress run",
  "cy:open": "cypress open"
}
```

```
yarn cy:open
```

STRUKTURA FOLDERÓW

API - SELEKTORY

```
cy.get('[data-cy=submit]')

cy.get('.article').find('footer')

cy.get('td').closest('.filled')

cy.get('nav a').first()

cy.get('#main-content')
  .find('.article')
  .children('img[src^="/static"]')
  .first()
```

API - INTERAKCJE

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

cy.contains('Welcome').dblclick()

cy.get('input').type('Hello, World')

cy.get('[type="radio"]').first().check()

cy.get('a').trigger('mousedown')
```

Zestaw akcji, które są wykonywane przed wywołaniem eventu:

- przeskroluj do elementu
- upewnij się że element nie jest ukryty
- upewnij się że element nie jest disabled
- upewnij się że element nie animuje się
- upewnij się że element nie jest przykryty
- przescrolluj stronę by sprawdzić czy element nie jest przykryty przez inny z fixed position

API - ASERCJE

```
cy.get('button').click().should('have.class', 'active')

cy.get('.completed').should('have.css', 'text-decoration', 'line-through')

cy.get('.nav').contains('About')

cy.get('#loading').should('not.exist')

cy.get('button').should('be.visible')

cy.request('/users/1').its('body').should('deep.eq', { name: 'Jane' })

cy.get(':radio').should('be.checked')
```

- Asercje opisują porządany stan elementów, obiektów i aplikacji
- Występuje tutaj mechanizm powtarzania (auto-retry)
- Używana jest biblioteka Chai z rozszerzeniami Chai-jQuery i Sinon-Chai
- Wiele komend takich jak find(), get(), type(), click() zawiera wbudowane asercje (mają wymagania, które mogą spowodować wystąpienie błędu bez potrzeby dodawania dodatkowej asercji)

API - NETWORK REQUESTY

```
cy.server()
cy.route({
  method: 'POST',
  url: '/avatar/*',
  status: 500,
  response: {}
})

cy.server()

// we set the response to be the activites.json fixture
cy.route('GET', 'activities/*', 'fixture:activities.json').as('getActivities') // alias

// forces Cypress to wait
// until it sees a response for each request that matches
// each of these aliases
cy.wait(['@getActivities', '@getMessages'])
```

- Za pomocą server() włączamy kontrolę nad wszystkimi requestami do końca testu
- Możemy dzięki temu łatwo stubować response i kontrolować *body*, *status* i *headers*
- route() przypisuje dany response do requestu (tabela routingu)

DEMO

ZALETY

- Snapshoty
- Łatwe debugowanie
- Przewidywalne i powtarzalne wyniki testów
- Zarządzanie ruchem sieciowym
- Screenshoty i wideo
- Dobra dokumentacja

WADY

- Brak wsparcia dla otwierania kilku tabów
- Nie ma wsparcia dla SEO - indeksowanie i web crawlery
- Nie ma możliwości testowania wydajności
- Skrypty embeddowane na innych stronach (3rd party)
- Jeden test -> jedna superdomena
- Brak wsparcia dla eventów natywnych/mobilnych
- Brak wsparcia dla innych przeglądarek
- Testy są odpalane w przeglądarce -> supportowany jest tylko Javascript, komunikacja z backendem jest utrudniona

ŹRÓDŁA

- cypress.io
- Cypress: The future of E2E testing - Dominic Elm
- I see your point, but... (Part1) - Brian Mann
- I see your point, but... (Part2) - Gleb Bahmutov
- Jak pisać testy end-to-end... - Marcin Czarkowski
- Frontend Masters: Testing React Applications
- Testing Javascript - Kent C. Dodds