s|ngular

it can be done.

Testing in front with Jest and Testing Library

May 2021

s|ngular

it can be done.

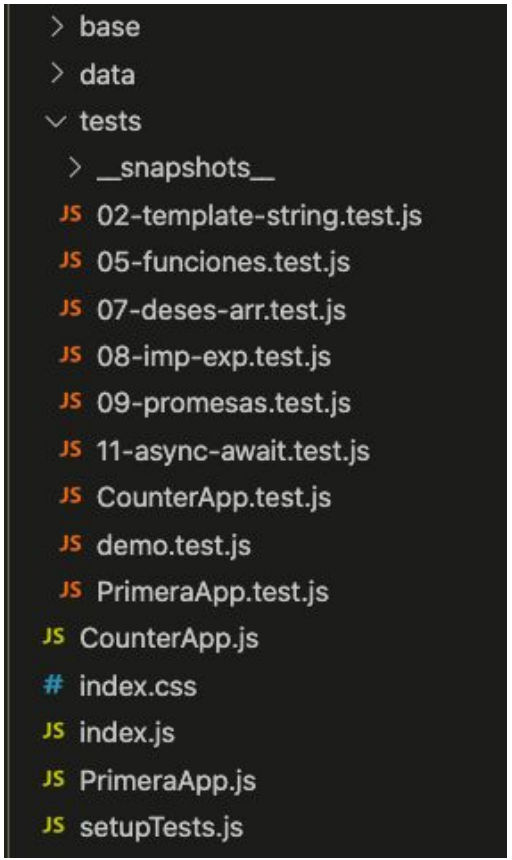
Twitter and Twitch: **Ari_Reinventada**

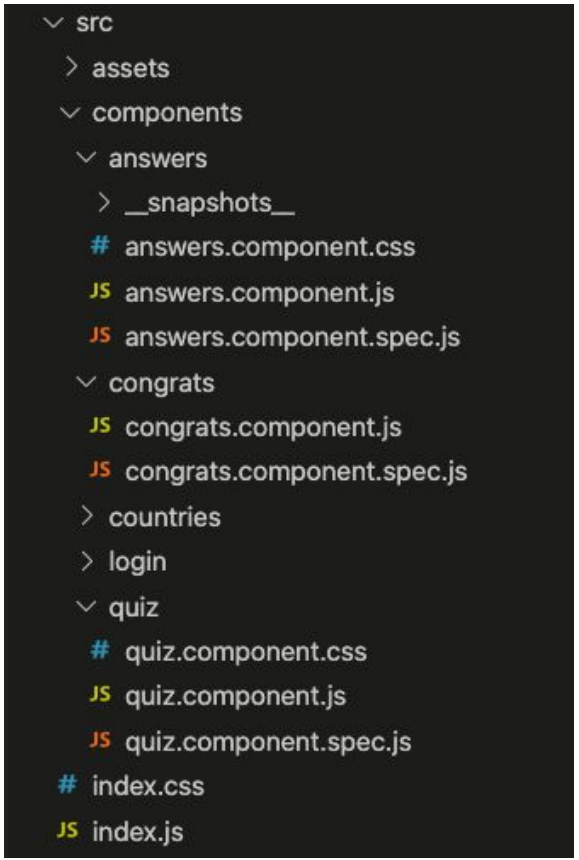
Youtube: **Ari Reinventada**

Github: **ArianeJDB**

How to start?

How to start?





How to write tests?

```
describe('Name of component', () => {

  test('should...',  () => {

    // Arrange - Given that

    // Act - When the component

    // Assert - Then the component

  });
});
```

```
describe('Name of component', () => {

  it('should...',  () => {

    // Arrange - Given that

    // Act - When the component

    // Assert - Then the component

  });
});
```

How to write tests?

```
describe('Validator', () => {
    test('should validate if comments doesnt have forbidden words', () => {
        // Arrange
        const comment = 'hola';
        const offensiveWords = defaultWords;
        //Act
        const input = validator.validator(comment, offensiveWords)
        //Assert
        expect(input).toBeFalsy();
    })
});
```

Mocking

Determinism

Spy

```
describe("Form component", () => {
  it("calls submit method", () => {
    const submit = jest.fn();
    render(<Form submit={submit} />);

    expect(submit).toHaveBeenCalled();
  });
});
```

Stub

```
describe('api test', () => {
  it('should return members when it resolves the request successfully', async () => {
    const members = [
      {
        id: 1,
        login: 'test login',
        avatar_url: 'test avatar_url',
      },
    ];

    const getStub = jest.spyOn(Axios, 'get').mockResolvedValue({
      data: members,
    });

    const result = await getMembers();

    expect(getStub).toHaveBeenCalledWith(
      'https://api.github.com/orgs/lemoncode/members'
    );
    expect(result).toEqual(members);
  })
});
```

Stub

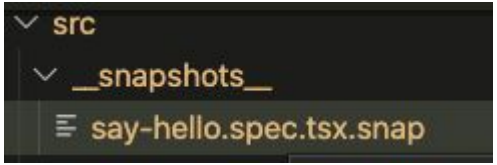
```
it('should return members when it resolves the request successfully', done => {
  const members = [
    {
      id: 1,
      login: 'test login',
      avatar_url: 'test avatar_url',
    },
  ];

  const getStub = jest.spyOn(Axios, 'get').mockResolvedValue({
    data: members,
  });


  getMembers().then(result => {
    expect(getStub).toHaveBeenCalledWith(
      'https://api.github.com/orgs/lemoncode/members'
    );
    expect(result).toEqual(members);
    done();
  });
});
```

Snapshot

.toMatchSnapshot()

```
it('should display the person name using snapshot testing', () => {
  const person = 'Ari';

  const { asFragment } = render(<SayHello person={person} />);

  expect(asFragment()).toMatchSnapshot();
});
```

.toMatchSnapshot()



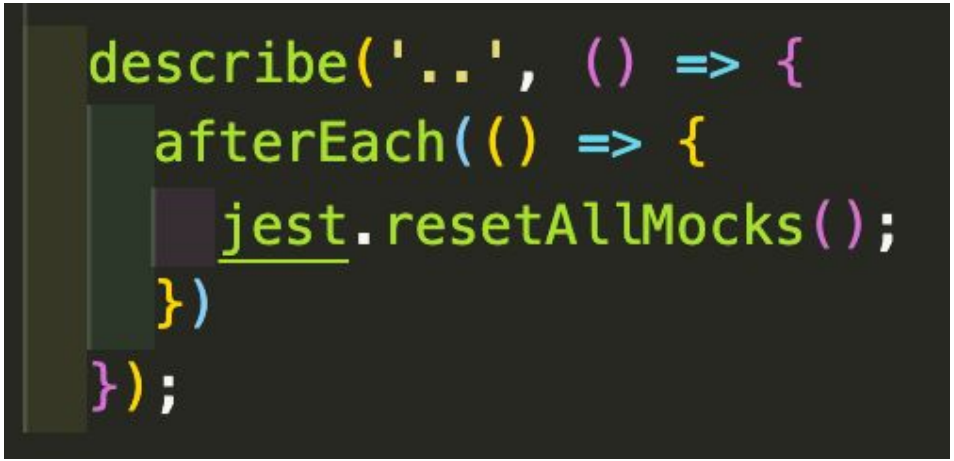

```
// Jest Snapshot v1, https://goo.gl/fbAQLP

exports[`SayHello component specs should display the person name using snapshot testing`] = `
<DocumentFragment>
  <h1>
    Hello
    <strong>
      Ari
    </strong>
  </h1>
</DocumentFragment>
`;
```

Update snapshot

```
Watch Usage
 › Press a to run all tests.
 › Press f to run only failed tests.
 › Press p to filter by a filename regex pattern.
 › Press t to filter by a test name regex pattern.
 › Press u to update failing snapshots.
 › Press i to update failing snapshots interactively.
 › Press q to quit watch mode.
 › Press Enter to trigger a test run.
```

Clean Up

jest.restoreAllMocks()



restoremocks: true

s|ngular

it can be done.

Testing Library

s|ngular

it can be done.

Testing Library… and Jest!

Jest-dom

screen.getByText()

```
describe('Card component', () => {
  it('should display a card with title when it feeds a title and body', () => {
    const props = {
      title: 'Hi, JS World!',
      onClick: jest.fn(),
    };

    render(<CardComponent {...props} />);
    const titleElement = screen.getByText(props.title);

    expect(titleElement).toBeInTheDocument();
  });
});
```

Priority

1. Queries accessible to everyone

i. getByRole

ii. getByLabelText

iii. getByPlaceholderText

iv. getByText

2. Semantic queries

i. getByAltText

ii. getByTitle

3. Test IDs

i. getByTestId

Types of queries

getBy...

```
it('should display the person name', () => {
  const person = 'Ari';

  render(<SayHello person={person} />);
  const element = screen.getByRole('heading', { name: 'Hello Ari' });

  expect(element).not.toBeNull();
  expect(element.tagName).toEqual('H1');
});
```

queryBy...

```
describe("When the user is not logged", () => {
  it("should not render the form to post new comments", async () => {
    render(Comments, {
      store
    });

    const form = screen.queryByRole("heading", "Add a comment");
    expect(form).not.toBeInTheDocument();
  });
});
```

findBy...

```
it('should show a warning for each empty input field', async () => {
  const container = document.createElement('div')
  userEvent.click(
    getByRole(container, 'button', {
      name: 'Login',
    })
  )

  expect(await findByText(container, 'Username required')).toBeVisible()
  expect(await findByText(container, 'Password required')).toBeVisible()
})
```

userEvent

userEvent

```
test("should call submit method with username and greeting", () => {
  const submit = jest.fn();
  render(<Form submit={submit} />);

  const username = screen.getByLabelText('username');
  userEvent.type(username, "Ari");

  const greeting = screen.getByLabelText('greeting');
  userEvent.type(greeting, 'Hi JS World!');

  const button = screen.getByRole('button', 'submit');
  userEvent.click(button);

  expect(submit).toHaveBeenCalledWith({
    username: 'Ari',
    greeting: 'Hi JS World!',
  });
});
});
```

throwSuggestions

throwSuggestions: true

```
import { render, configure } from '@testing-library/vue'

beforeEach(() => {
  configure({
    throwSuggestions: true,
  })
})
```

```
TestingLibraryElementError: A better query is available, try this:
findByRole('heading', { name: /plan settings \- q:scan/i })
```

Thank you

USA | Spain | Mexico | Chile | Singapore
www.sngular.com

Resources and credits

- Jest documentation
- Testing Library documentation
- Codely.tv (spanish awesome courses)
- Workshop with Jon Rojí and Fran Moreno
- Thank you Sara Lisette for this tweet