Testning med Jest och Enzyme

Vad är Jest?

Ett test-ramverk skrivet av Facebook - med stöd för parallell exekvering av tester och snapshot-testning.

Finns förinstallerat med create-react-app och installeras enkelt med npm eller yarn.

BDD -tester

Skriv tester för att testa beteende och inte kod

```
test('Ett plus ett blir två', () => {
  expect(1+1).toBe(2);
});
```

NORTH LINK

Jest körs via terminalen

```
npm install --save-dev jest

package.json
{
  "scripts": {
    "test": "jest"
  }
}

npm run test # kör test en gång
npm run test -- --watch # övervaka förändringar och kör tester
```

NORTH LINK

Matchers

Jest använder ett system de kallar matchers, som låter dig testa värden på olika sätt.

```
test('ett plus ett', () => {
    expect(1 + 1).toBe(2); # Object.is
    expect([1,1]).toEqual([1,1]);
    expect(1 + 1).toBeGreaterThan(1);
    expect(1 + 1).toBeLessThan(3);
});
```

Man kan testa alla typer av värden

```
test('Olika typer', () => {
    expect('Niklas').not.toMatch(/V/);

    const namnLista = ['niklas', 'viktor'];
    expect(namnLista).toContain('niklas');
});
```

beforeEach & afterEach

```
let val = 0;
beforeEach(() => {
val = null;
});
test('First', () => {
  expect(val).toBeNull();

  val = 1+2;
  expect(val).toBe(2)
});
test('Second', () => {
  expect(val).toBeNull();
  val = false;
  expect(val).toBeFalsy();
});
```

NORTH LINK

Asynkrona-tester

Asynkron-kod kan testas geom att använda done()-callbacken

```
test('Promise', (done) => {
  somePromiseFunction()
  .then((value) => {
    expect(value).toBe(10)
    done();
  });
});
```

Mockade funktioner

Det går att skapa mocks för att underlätta tester. En mock är en funktion som ersätter beteendet på en existerande funktion.

```
const mockCallback = jest.fn(x => 100+x);
mockCallback(100)
expect(mockCallback.mock.calls.length).toBe(101);
expect(mockCallback.mock.calls[0][0]).toBe(100);
```

Mocka moduler

Det går även att byta ut en funktion på en existerande modul.

```
import axios from 'axios';

const res = {data: 100};
jest.mock('axios');
axios.get.mockResolvedValue(resp);
```

Övning 1 - skriv tester för ett antal funktioner

installera jest och bygg tester för ett antal exporterade funktioner.

Testa att köra jest med olika cli-argument, t.ex --watch för att lyssna på förändringar.

Problemen med att testa DOMen

- Har tidigare krävt att sidan renderas i en webbläsare först - och sedan anropas via ett ramverk som phantomjs
- Svårt och frustrerande att manipulera events
- Går ofta långsamt och svårt att parallellisera

Reacts renderar utanför browsern

Kan rendera komponenter utanför browsern

Shallow vs Full-rendering

Enzyme - ett test-ramverk till React

- Gör det enkelt att traversera Komponenters DOM
- Enzymes API efterspeglar hur jQuerys API ser ut

Enzyme wrappar komponenter

```
import { shallow } from 'enzyme';

test('Check dom-text', () => {
  const wrapper = shallow(<TestComponent />);
  expect(wrapper.find(“.foo”).text()).toContain('test')
});
```

NORTH LINK

Snapshot testning

Ett snapshot test är ett test som vid sin första körning renderar en sida och sparar resultaten till en bild. Bilden kan sedan jämföras vid kommande körningar för att se om någon förändring har skett.

I kombination med React går det att generera ett snapshot för en enstaka komponent.

Ett exempel på snapshot-testning med React

```
import renderer from 'react-test-renderer';

it('renders name correctly', () => {
  const tree = renderer
    .create(<UserComponent name={"niklas"}>)
    .toJSON();
  expect(tree).toMatchSnapshot();
});
```

NORTH LINK

Övning 2 - testa React app

Skapa en ny applikation, skriv en komponent och testa att den kan skriva ut innehåll

```
create-react-app app-name
npm install --save react-test-renderer
npm install --save enzyme enzyme-adapter-react-16 react-test-renderer
```

NORTH LINK