Testes Unitários em Angular

https://codecraft.tv/courses/angular/unit-testing/overview/

Angular foi criado com o objetivo principal de ser fácil testar.

Diferentes tipos de testes:

Testes Funcionais, ou **testes end to end (e2e)**, consistem no teste da funcionalidade completa de uma aplicação. Numa aplicação web, significa interatuar com a aplicação a correr no browser, imitando a interação de um utilizador real efetuando clicks na página.

Testes de Integração, testes que invocam diretamente o nosso código. Exemplo: teste de um serviço Angular. O teste invoca uma função com um conjunto de parâmetros e verifica se o resultado é o esperado.

Testes Unitários, ou testes isolados, semelhantes aos testes de integração exceto que obrigam a assegurar que nada é executado para além da função que se está a testar. Por exemplo ao testar um serviço que faz um pedido http a uma web api, um teste de integração inclui a invocação à api, enquanto que um teste unitário terá de usar um objeto para substituir a invocação à api para que o código executado pelo teste fique restringido apenas à função do serviço.

Jasmine é um framework JavaScript para testes que suporta a prática de desenvolvimento de software designada Behaviour Driven Development (BDD), um tipo de Test Driven Development (TDD).

Exemplo de um teste Jasmine para testar a função seguinte:

```
function helloWorld() {
    return 'Hello world!';
}
```

Especificação do teste Jasmine:

```
describe('Hello world', () => {                  // 1
    it('says hello', () => {                     // 2
        expect(helloWorld())                     // 3
            .toEqual('Hello world!');            // 4
    });
});
```

Na linha 1 `describe(string, function)`, function define um **Test Suite**, uma coleção de testes individuais designados **Test Specs**.

Na linha 2 `it(string, function)`, function define um Test Spec individual, o qual contém um ou mais **Test Expectations**.

A linha 3 `expect(expression)`, designa-se por **Expectation** e em conjunto com o Matcher descreve uma parte do comportamento esperado na aplicação.

A linha 4 `matcher(expected),` designa-se por **Matcher** e faz uma comparação booleana entre o valor esperado e o resultado do primeiro argumento da função expect.

Função it – descrição legível do que estamos a testar. É mostrado no resultado do teste e é útil para compreender o que pode falhar.

Matchers built-in:

```
expect(array).toContain(member);
expect(fn).toThrow(string);
expect(fn).toThrowError(string);
expect(instance).toBe(instance);
expect(mixed).toBeDefined();
expect(mixed).toBeFalsy();
expect(mixed).toBeNull();
expect(mixed).toBeTruthy();
expect(mixed).toBeUndefined();
expect(mixed).toEqual(mixed);
expect(mixed).toMatch(pattern);
expect(number).toBeCloseTo(number, decimalPlaces);
expect(number).toBeGreaterThan(number);
expect(number).toBeLessThan(number);
expect(number).toBeNaN();
expect(spy).toHaveBeenCalled();
expect(spy).toHaveBeenCalledTimes(number);
expect(spy).toHaveBeenCalledWith(...arguments);
```

Por vezes é necessário realizar certas atividades antes de um teste (como por exemplo criar um objeto) designadas **setup**, e outras depois do teste, designadas **teardown**.

Funções Jasmine para setup e teardown:

- beforeAll – função invocada uma vez antes de todos os specs do test suite describe correrem.
- afterAll - função invocada uma vez depois de todos os specs do test suite describe correrem.
- **beforeEach** - função invocada antes de cada test specification, função it, correr.
- afterEach - função invocada depois de cada test specification, função it, correr.

Karma é uma ferramenta que pode observar alterações nos ficheiros em desenvolvimento e correr os testes automaticamente.

Projetos Angular, criados usando Angular CLI, também criam testes unitários que usam Jasmine e Karma.

Para correr todos os testes unitários numa aplicação Angular basta executar
> ng test