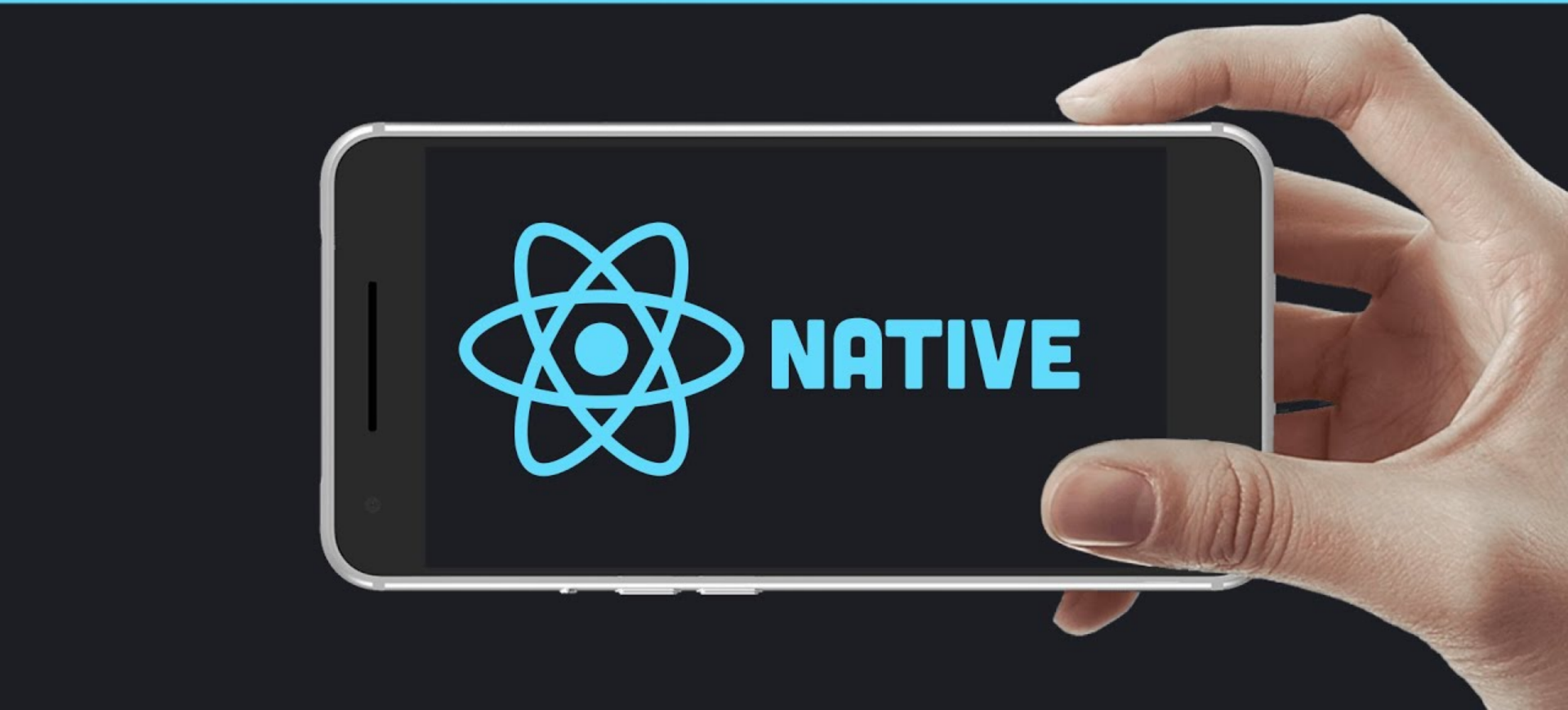
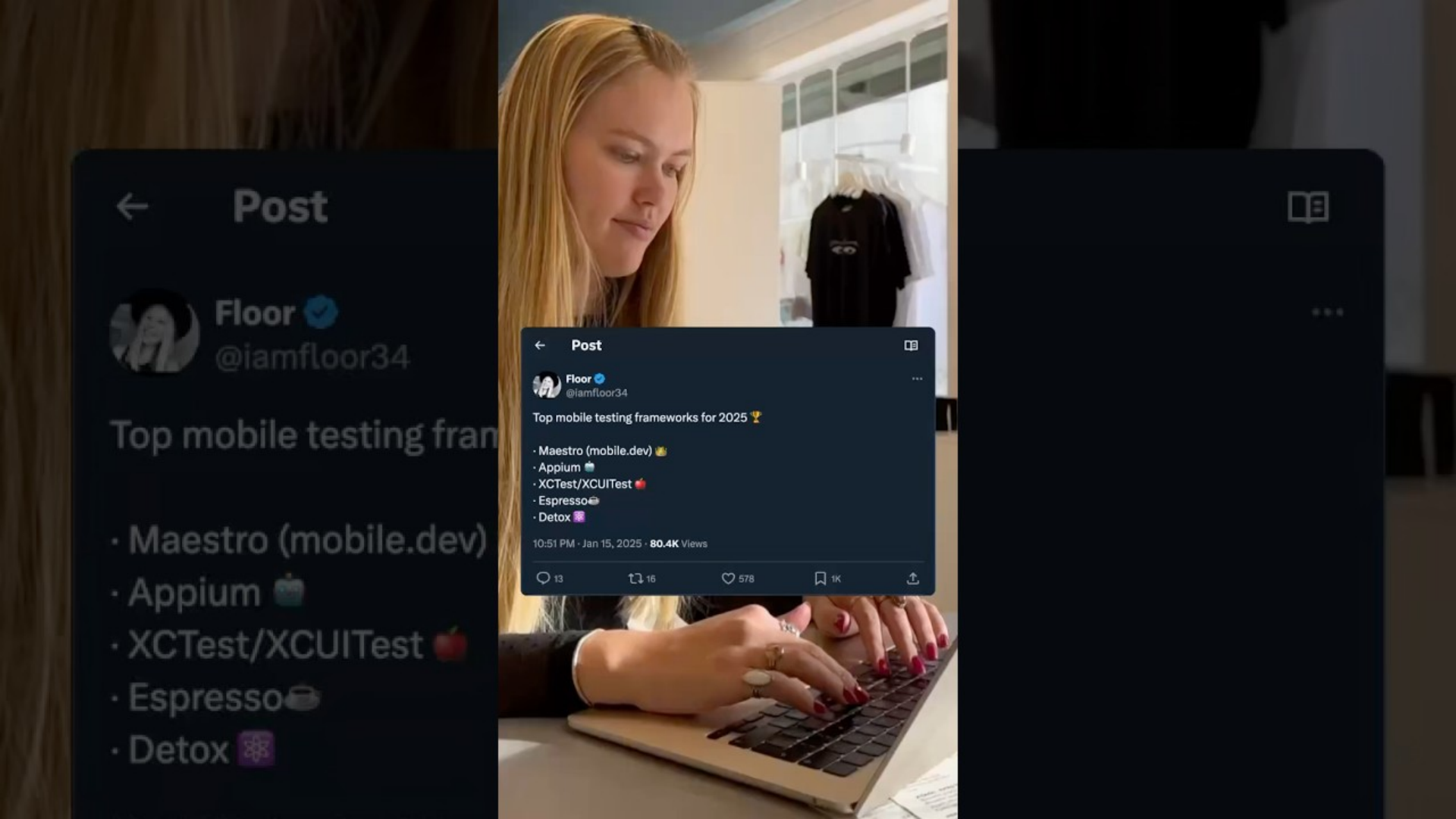
Mobile App Development

Tests

100 SECONDS OF




Post
Floor
@iamfloor34
Top mobile testing frameworks for 2025 🏆
· Maestro (mobile.dev) 🐝
· Appium
· XCTest/XCUITest
· Espresso
· Detox
10:51 PM · Jan 15, 2025 · 80.4K Views
13
16
578
1K

https://www.youtube.com/watch?v=wlyHSOugGGw

Tests

O que é?

Testes são códigos que não fazem parte do seu aplicativo, e que são rodados em conjunto com o código que estamos desenvolvendo para testá-lo. A importância dos testes é de garantir o funcionamento do nosso projeto quando há mudanças, que podem ter efeitos colaterais não tão óbvios, e comprometer uma parte do nosso aplicativo

Tests

Bônus

Além disso, testes costumam nos ajudar a programar de uma maneira mais modular, quando nossos testes são muito custosos de fazer, é um sinal de que nosso código está muito acoplado

Tests

Tipos de testes

Existem 3 tipos de testes utilizados em programação, de acordo com a parcela do fluxo que está sendo testada:

- Testes unitários (Unit tests)
- Testes de integração (Integration tests)
- Testes End-to-End

Tests

Testes unitários

Testam apenas um componente, uma função, a menor unidade testável dentro do nosso código. São os mais comuns, e não é raro em empresas os testes unitários terem cobertura de 100% ou próximas a isso.

Tests

Testes de Integração

Quando envolvem mais de um componente ou função. Já estamos testando uma parte maior do nosso fluxo e como os componentes/classes/métodos conversam entre si

Tests

Testes end-to-end

Testam um fluxo inteiro do nosso sistema. Envolve a necessidade de termos o nosso backend, um sandbox, ou ao menos um serviço mockado especialmente para o teste.
Geralmente envolvem o uso de um software para simular um usuário, como selenium ou cypress.

Tests

Testes unitários

Aqui focaremos nos testes unitários. Eles são os mais simples e mais utilizados, e a base para o entendimento do teste de integração (end-to-end é um caso à parte)

Tests

Testes unitários

Para os testes unitários, usaremos o Jest, a principal ferramenta de testes para javascript, e o Testing Library, que ocupou o posto de principal framework de testes para react, que antes pertencia ao Enzyme

Tests

Testes unitários

Quando estivermos preparando os nossos testes, vamos separá-los em 3 partes, utilizando o padrão Arrange, Act, Assert (AAA). Essa divisão as vezes pode até parecer verbosa para casos simples, mas vai nos ajudar a ter mais clareza de o que e como estamos testando

Tests

Arrange

Aqui é quando preparamos o que o nosso teste irá precisar. Declaramos inputs, outputs esperados, mocks necessários, criamos objetos…

Tests

Act

Aqui é a etapa que iremos, como o próprio nome diz, agir. Se for um método, por exemplo, aqui chamamos o método com o input que foi definido, e salvamos numa variável

Tests

Assert

Essa é a parte de verificação do nosso teste. Aqui nós checamos se o resultado foi o esperado

Tests

O que testar em métodos

Quando estamos testando métodos puros (independente de estado), precisamos fornecer inputs e outputs conhecidos, rodar o nosso método nos inputs, e esperar os outputs

Tests

Exemplos

Um método que faz soma, e aceita string como input:

```
const sum = (a, b) => Number(a) + Number(b);
```

Tests

```
describe('Testing sum method', () => {
  it('for 2 numbers', () => {
    // Arrange
    const inputsA = [1, 2]
    const outputA = 3

    const inputsB = [2.5, 3.33]
    const outputB = 5.83

    // Act
    const resultA = sum(...inputsA)
    const resultB = sum(...inputsB)

    // Assert
    expect(resultA).toBe(outputA);
    expect(resultB).toBe(outputB);
  });
```

```
it('for 2 strings', () => {
  // Arrange
  const inputsA = ['1', '2']
  const outputA = 3

  const inputsB = ['2.5', '3.33']
  const outputB = 5.83

  // Act
  const resultA = sum(...inputsA)
  const resultB = sum(...inputsB)

  // Assert
  expect(resultA).toBe(outputA);
  expect(resultB).toBe(outputB);
});
```

Tests

Exemplos

Um método que faz divisão, e aceita string como input:

```
const divide = (a, b) => Number(a)/Number(b);
```

Tests

```
describe('Testing divide method', () => {
  it('for 2 numbers', () => {
    // Arrange
    const inputsA = [4, 2]
    const outputA = 2

    const inputsB = [6.25, 2.5]
    const outputB = 2.5

    // Act
    const resultA = divide(...inputsA)
    const resultB = divide(...inputsB)

    // Assert
    expect(resultA).toBe(outputA);
    expect(resultB).toBe(outputB);
  });
```

```
it('for 2 strings', () => {
  // Arrange
  const inputsA = ['4', '2']
  const outputA = 2

  const inputsB = ['6.25', '2.5']
  const outputB = 2.5

  // Act
  const resultA = divide(...inputsA)
  const resultB = divide(...inputsB)

  // Assert
  expect(resultA).toBe(outputA);
  expect(resultB).toBe(outputB);
});
```

```
PS C:\Users\mateu\Desktop\mobile-dev\HMAD-2023\aula-10-storage-and-tests> npm test

> aula-10-storage-and-tests@1.0.0 test
> jest

 PASS  ./aula.test.js
  Testing sum method
    √ for 2 numbers (2 ms)
    √ for 2 strings

Test Suites: 1 passed, 1 total
Tests:       2 passed, 2 total
> aula-10-storage-and-tests@1.0.0 test
> jest

 PASS  ./aula.test.js
  Testing sum method
    √ for 2 numbers (3 ms)
    √ for 2 strings
  Testing divide method
    √ for 2 numbers
    √ for 2 strings

Test Suites: 1 passed, 1 total
Tests:       4 passed, 4 total
Snapshots:   0 total
Time:        1.409 s, estimated 3 s
Ran all test suites.
```

Tests

TUDO CERTO!! OS TESTES PASSARAM!!

Tests

O que testar em métodos

Devemos sempre nos atentar a corner cases. Seja quando os inputs não são os esperados, mas são possíveis, ou caos que são fáceis de esquecer quando programamos:

- Array vazio
- Objeto vazio
- 0, negativo, float
- String vazia

Tests

Estrutura dos testes

Como vocês observaram, os testes estão divididos em **describes** e **its**
Describes são delimitadores de escopo, enquanto os its são os testes em si.
Essa divisão passa a ser mais importante quando nossos testes se tornam mais complexos

Tests

BeforeEach, beforeAll, afterEach, afterAll

É muito comum termos um setup mais complexo em nossos testes, principalmente quando envolve mocks

app.js conf

Henry Moulton

React Native end-to-end testing with Maestro

Dúvidas, anseios, desabafos?


FIAP