Framework AVA

1. Descrição do Framework AVA

O **AVA** é um *framework* de testes minimalista para **Node.js**, projetado para ser **rápido, simples e altamente concorrente**. Seu principal diferencial é a execução paralela dos testes, o que reduz significativamente o tempo total de execução em projetos com múltiplos casos de teste.

Além disso, o AVA adota uma sintaxe moderna baseada em **módulos ECMAScript (ESM)** e utiliza **Promises** e **async/await** nativamente, tornando-o ideal para o ecossistema JavaScript contemporâneo. A simplicidade de sua API e a ausência de configurações complexas o tornam especialmente útil em projetos que priorizam **rapidez e clareza nos testes**.

Outra característica marcante é seu **isolamento entre casos de teste**, garantindo que cada teste seja executado de forma independente, evitando interferências entre variáveis globais ou estados compartilhados. Isso contribui diretamente para a **confiabilidade e repetibilidade dos testes**.

2. Categorização do Framework AVA

i) Técnicas de Teste

O AVA se enquadra predominantemente na categoria de **testes de caixa-preta (black-box testing)**.

Isso ocorre porque, em geral, ele é utilizado para validar **o comportamento externo das funções e módulos**, sem a necessidade de inspecionar ou depender da implementação interna do código testado.

Por exemplo, ao testar uma função matemática, o desenvolvedor verifica se as entradas produzem as saídas esperadas, sem se preocupar com a lógica interna que gera o resultado.

Entretanto, ele também pode ser utilizado em **testes de caixa-branca**, especialmente em projetos menores ou em contextos de verificação de funções internas, quando o desenvolvedor deseja garantir que determinadas ramificações ou condições internas do código sejam corretamente executadas.

Essa flexibilidade torna o AVA adequado tanto para testes de **funções puras (unitárias)** quanto para módulos que envolvem **lógica interna complexa**.

ii) Níveis de Teste

O AVA é mais comumente empregado para **testes unitários**, mas pode também ser utilizado em **testes de integração**.

- **Testes Unitários:**

- São o foco principal do AVA. Ele é amplamente utilizado para verificar o comportamento de pequenas partes isoladas do código, como funções, classes ou métodos.

- Por exemplo, testar uma função que calcula juros compostos é um típico caso de uso unitário, garantindo que entradas específicas gerem o resultado esperado.

- **Testes de Integração:**

O AVA também suporta esse nível de teste, permitindo que diferentes módulos ou componentes do sistema sejam testados em conjunto, para verificar se interagem corretamente.

Isso é possível porque ele é compatível com bibliotecas de simulação (*mocks*, *stubs* e *spies*) e pode executar chamadas assíncronas para APIs, bancos de dados ou serviços externos.

Portanto, sua aplicação é flexível — indo desde a **validação isolada de funções** até a **verificação de interações entre componentes**.

iii) Tipos de Teste

Dentro da taxonomia dos tipos de teste, o AVA é mais frequentemente utilizado para:

- **Testes Funcionais:**

O framework é projetado para validar se o sistema ou função cumpre sua **função especificada**. Ele compara entradas e saídas, garantindo que o comportamento esteja conforme o esperado.

Por exemplo, verificar se uma função retorna o montante correto em um cálculo de juros compostos é um teste funcional.

- **Testes de Regressão:**

O AVA também é amplamente usado para prevenir regressões — ou seja, garantir que novas alterações de código não quebrem funcionalidades existentes. A simplicidade dos testes e a execução rápida em paralelo facilitam a manutenção de uma **suíte de regressão contínua**.

- **Testes de Bordas (Boundary Testing):**

É possível empregar o AVA para avaliar comportamentos em **limites de valores válidos e inválidos**, como entradas zero, negativas ou extremamente grandes. Essa prática aumenta a robustez e confiabilidade do sistema.

Assim, o AVA cobre de forma eficaz **testes unitários, funcionais e de regressão**, além de permitir extensões para testes de integração.

3. Forma de Instalação e Integração

A instalação e integração do AVA em um projeto Node.js é direta e rápida:

Passos para Instalação:

1. **Inicializar o projeto Node.js**:

```
npm init -y
```

2. **Instalar o AVA como dependência de desenvolvimento:**

```
npm install --save-dev ava
```

3. **Configurar o script de teste no package.json:**

```
"scripts": {
  "test": "ava"
}
```

4. **Criar arquivos de teste:**

Os arquivos de teste devem estar em uma pasta test/ ou terminarem com .test.js.

Exemplo:

```
import test from 'ava';
import { calcularMontante } from './juros.js';

test('calcula o montante corretamente', t => {
   const resultado = calcularMontante(1000, 0.05, 2);
   t.is(resultado, 1102.5);
});
```

5. **Executar os testes:**

```
npm test ou npx ava
```

O AVA também pode ser facilmente integrado a **pipelines de integração contínua (CI/CD)**, como GitHub Actions, GitLab CI e Jenkins, sendo uma ótima escolha para automação de testes em ambientes modernos de desenvolvimento ágil.

4. Conclusão

O **AVA** é um *framework* de testes **leve, rápido e moderno**, que privilegia a **simplicidade e a performance**.

Sua execução concorrente, suporte nativo a código assíncrono e integração com o ecossistema Node.js fazem dele uma excelente escolha para desenvolvedores que buscam eficiência e clareza nos testes.

Ao mesmo tempo, sua flexibilidade permite cobrir diferentes níveis e tipos de teste, desde unitários e de borda até integrações simples, tornando-o uma ferramenta poderosa para garantir **qualidade e estabilidade de software**.