

Features

- ✓ Vite's config, transformers, resolvers, and plugins.
- ✓ Use the same setup from your app to run the tests!
- ✓ Smart & instant watch mode, like HMR for tests!
- ✓ Component testing for Vue, React, Svelte, Lit and more
- ✓ Out-of-the-box TypeScript / JSX support
- ✓ ESM first, top level await
- ✓ Workers multi-threading via **Tinypool**
- ✓ Benchmarking support with **Tinybench**
- ✓ Filtering, timeouts, concurrent for suite and tests
- ✓ Jest-compatible Snapshot
- ✓ Chai built-in for assertions + Jest expect compatible APIs
- ✓ Tinyspy built-in for mocking
- ✓ happy-dom or jsdom for DOM mocking
- ✓ Code coverage via c8 or istanbul
- ✓ Rust-like in-source testing

In-Source Test

O Vitest também fornece uma maneira de executar testes em seu código-fonte junto com a implementação, semelhante aos testes de módulo do Rust.

Isso faz com que os testes compartilhem o mesmo fechamento que as implementações e sejam capazes de testar contra estados privados sem exportar. Enquanto isso, também traz um ciclo de feedback mais próximo para o desenvolvimento.

```
apps > my-app > src > App.tsx > ...
1 import { useState } from 'react';
2 import './App.css';
3 import { Button } from 'ziegler-ui';
4 import { userService } from './services';
5
6 export function add(...args: number[]) {
7   return args.reduce((a, b) => a + b, 0);
8 }
9
10 function App() {
11   const [user, setUser] = useState<any>(null);
12
13   const handleClick = async () => {
14     const { data } = await userService.getUserProfile(1);
15
16     setUser(data);
17   };
18
19   return (
20     <div className="App">
21       <Button label="Fetch User" onClick={handleOnClick} />
22       {user && <p>{user.name}</p>}
23     </div>
24   );
25 }
26
27 export default App;
28
29 if (import.meta.vitest) {
30   const { it, expect, test, describe } = import.meta.vitest;
31   describe('App', () => {
32     test('Button be defined', async () => {
33       it('add', () => {
34         expect(add()).toBe(0);
35         expect(add(1)).toBe(1);
36         expect(add(1, 2, 3)).toBe(6);
37       });
38     });
39   });
40 }
41
```