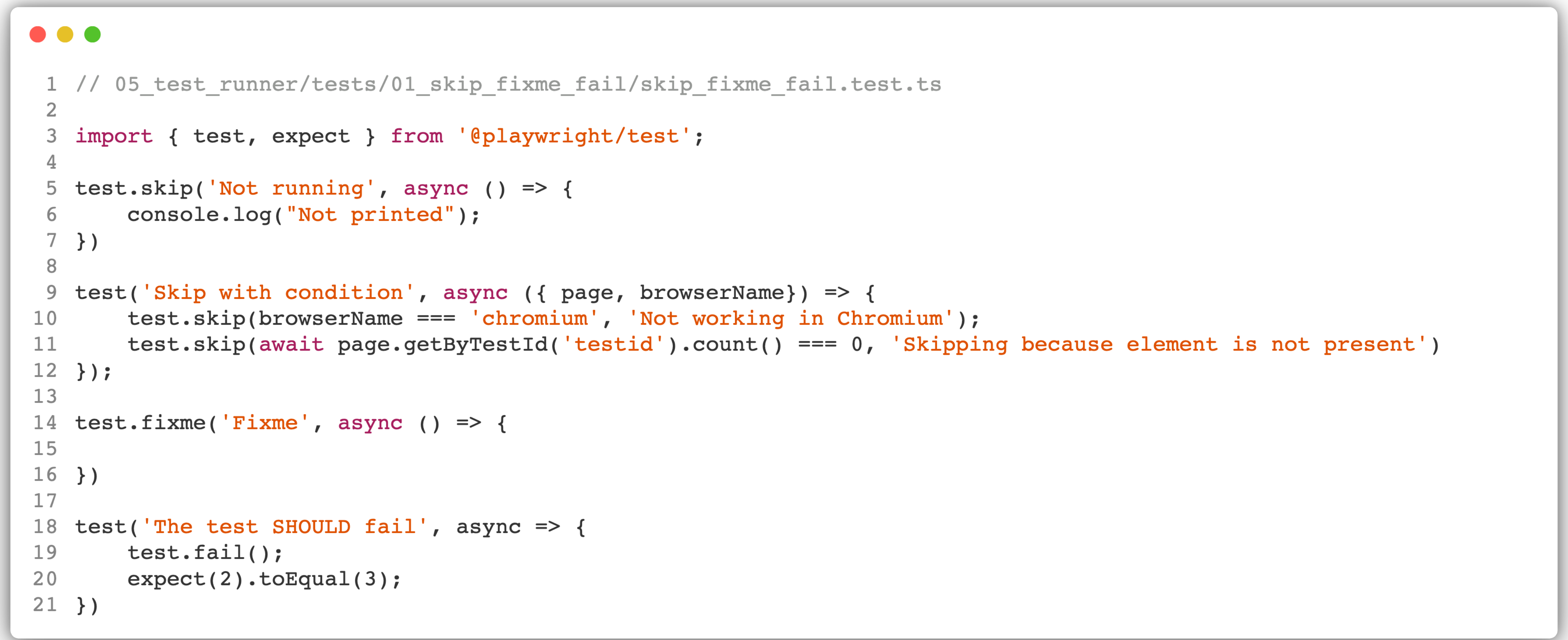
Test Runner

Konfigurieren wir unseren Test Runner

Skip, fixme and fail

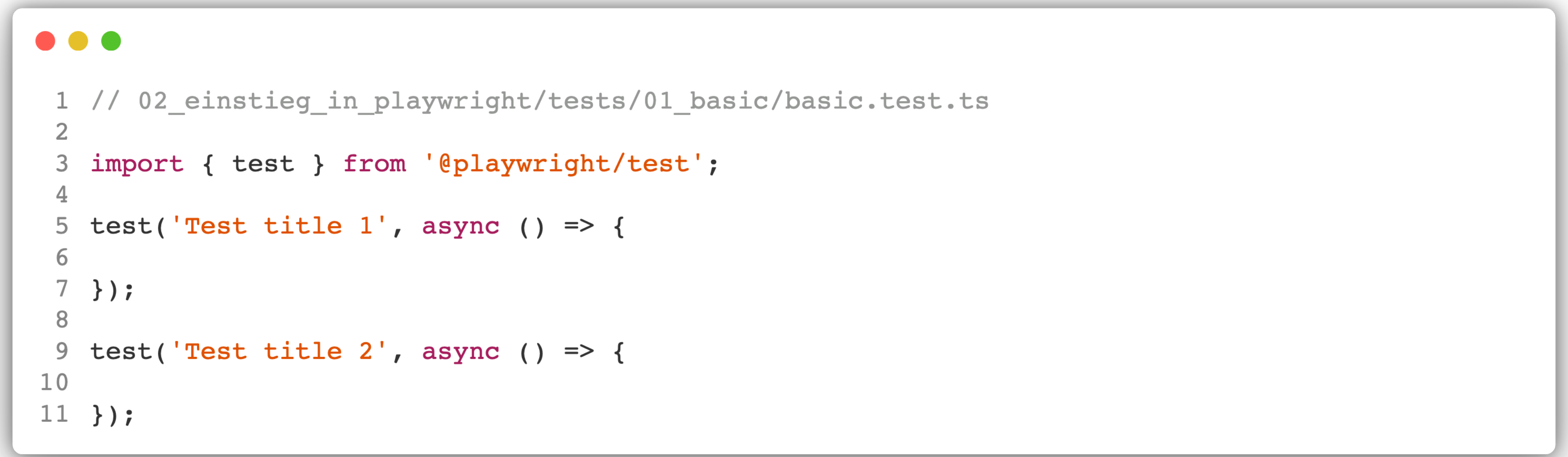


```
// 05_test_runner/tests/01_skip_fixme_fail/skip_fixme_fail.test.ts

import { test, expect } from '@playwright/test';

test.skip('Not running', async () => {
    console.log("Not printed");
})

test('Skip with condition', async ({ page, browserName}) => {
    test.skip(browserName === 'chromium', 'Not working in Chromium');
    test.skip(await page.getByTestId('testid').count() === 0, 'Skipping because element is not present')
});

test.fixme('Fixme', async () => {

})

test('The test SHOULD fail', async => {
    test.fail();
    expect(2).toEqual(3);
})
```

Anatomie eines Tests

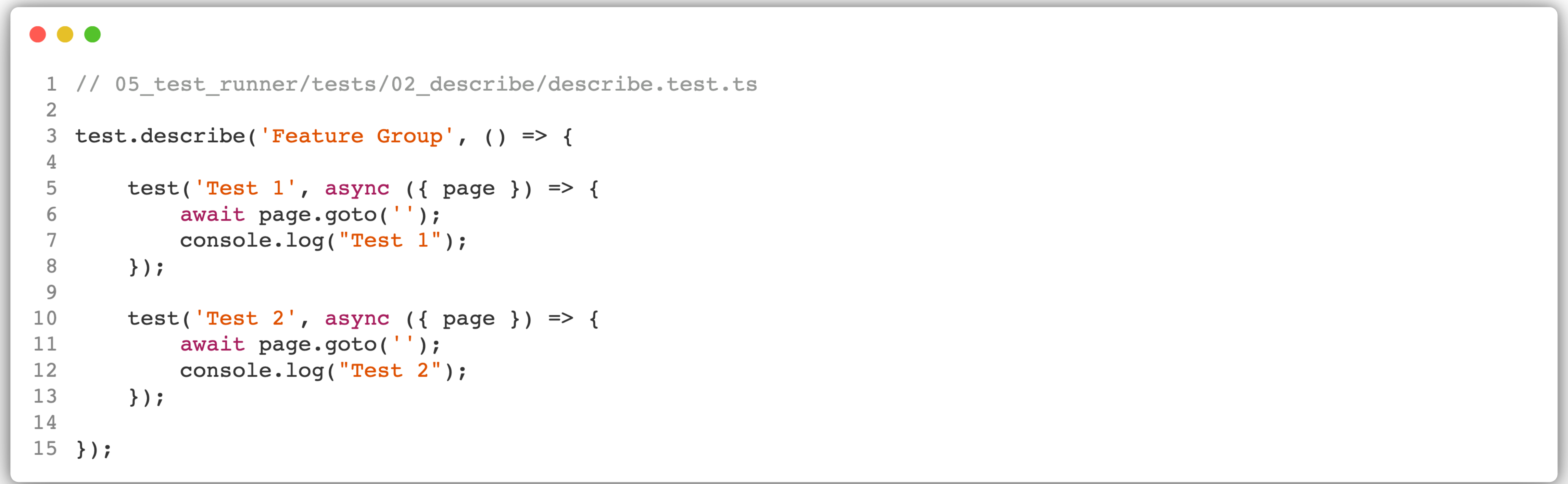
Wichtig für einen Test sind nur `zwei Dinge`:

- Im Datei-Namen muss `spec` oder `test` enthalten sein
- Der (Arrow)-Function muss `test` vorangestellt werden



```ts
// 02_einstieg_in_playwright/tests/01_basic/basic.test.ts

import { test } from '@playwright/test';

test('Test title 1', async () => {

});

test('Test title 2', async () => {

});
```

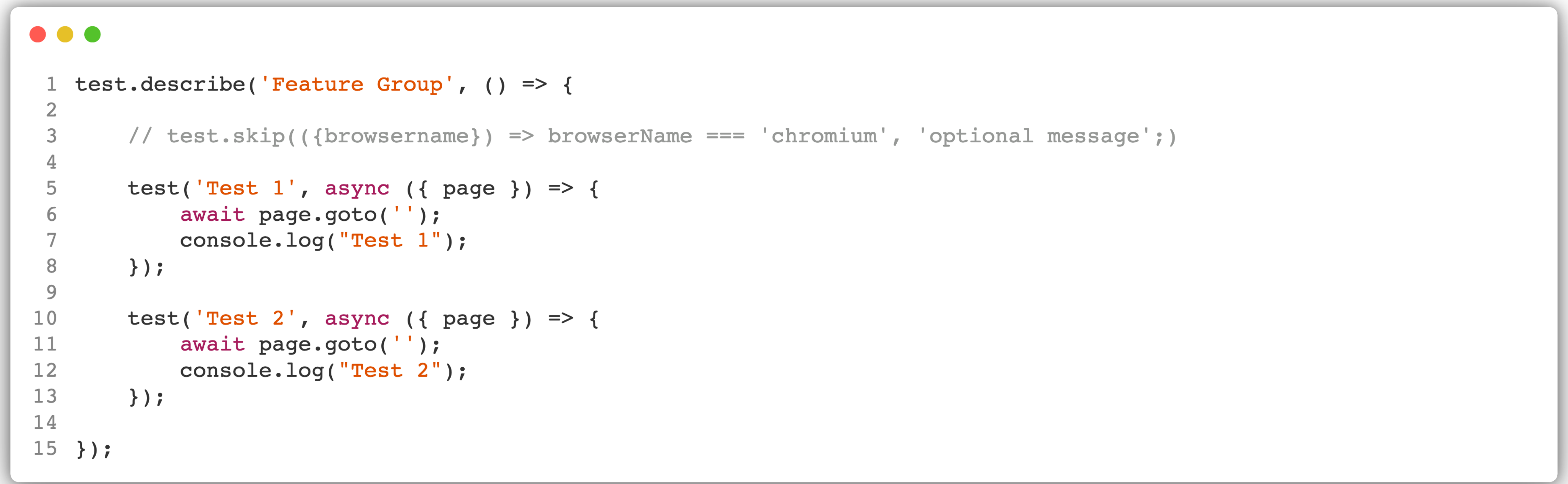
Describe



```
// 05_test_runner/tests/02_describe/describe.test.ts

test.describe('Feature Group', () => {

    test('Test 1', async ({ page }) => {
        await page.goto('');
        console.log("Test 1");
    });

    test('Test 2', async ({ page }) => {
        await page.goto('');
        console.log("Test 2");
    });

});
```

Achtung! Dadurch wird die Reihenfolge nicht beeinflusst.

Describe



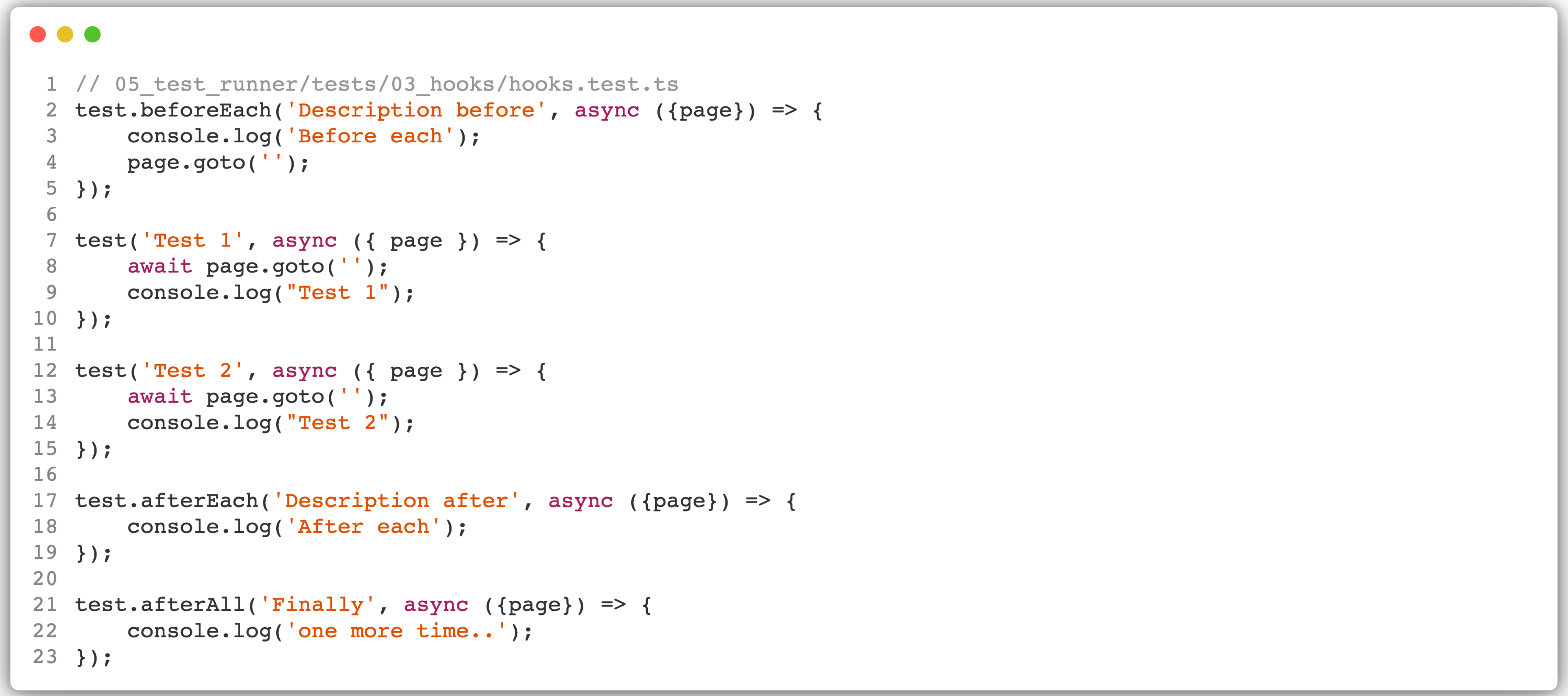
```javascript
test.describe('Feature Group', () => {

    // test.skip(({browsername}) => browserName === 'chromium', 'optional message';)

    test('Test 1', async ({ page }) => {
        await page.goto('');
        console.log("Test 1");
    });

    test('Test 2', async ({ page }) => {
        await page.goto('');
        console.log("Test 2");
    });

});
```

Ganze Gruppen können beeinflusst werden.

Hooks

```
test.beforeEach('Inject state', ...);

test('Check thing 1', async ({ page }) => {
    // ...
});

test('Check thing 2', async ({ page }) => {
    // ...
});

test.afterEach('Shut down DB', ...);
```

Hooks



```
// 05_test_runner/tests/03_hooks/hooks.test.ts
test.beforeEach('Description before', async ({page}) => {
    console.log('Before each');
    page.goto('');
});

test('Test 1', async ({ page }) => {
    await page.goto('');
    console.log("Test 1");
});

test('Test 2', async ({ page }) => {
    await page.goto('');
    console.log("Test 2");
});

test.afterEach('Description after', async ({page}) => {
    console.log('After each');
});

test.afterAll('Finally', async ({page}) => {
    console.log('one more time..');
});
```

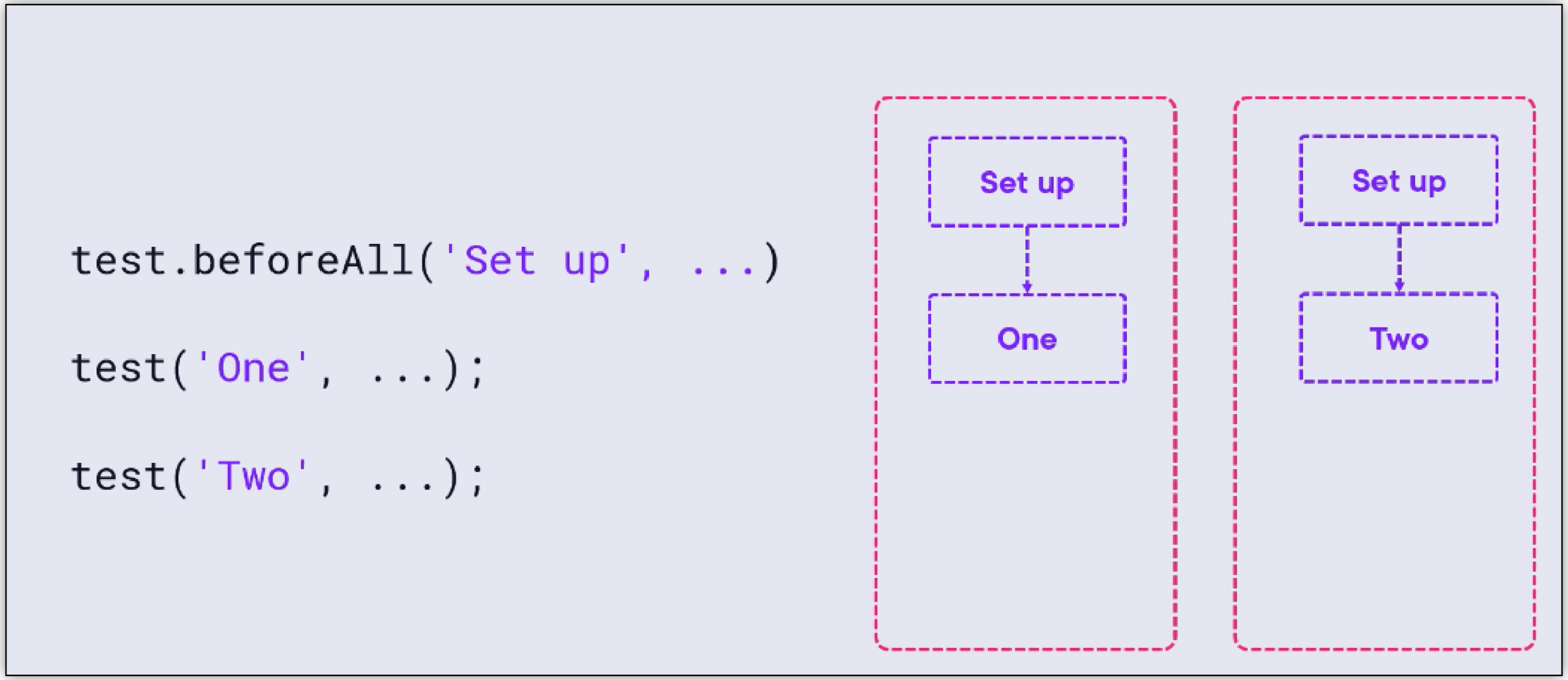
Worker 1

```
--worker 1
```

Ergibt einen sequentiellen Ablauf, default sind 2

=> beforeAll / afterAll ist für jeden Worker!

Worker 1

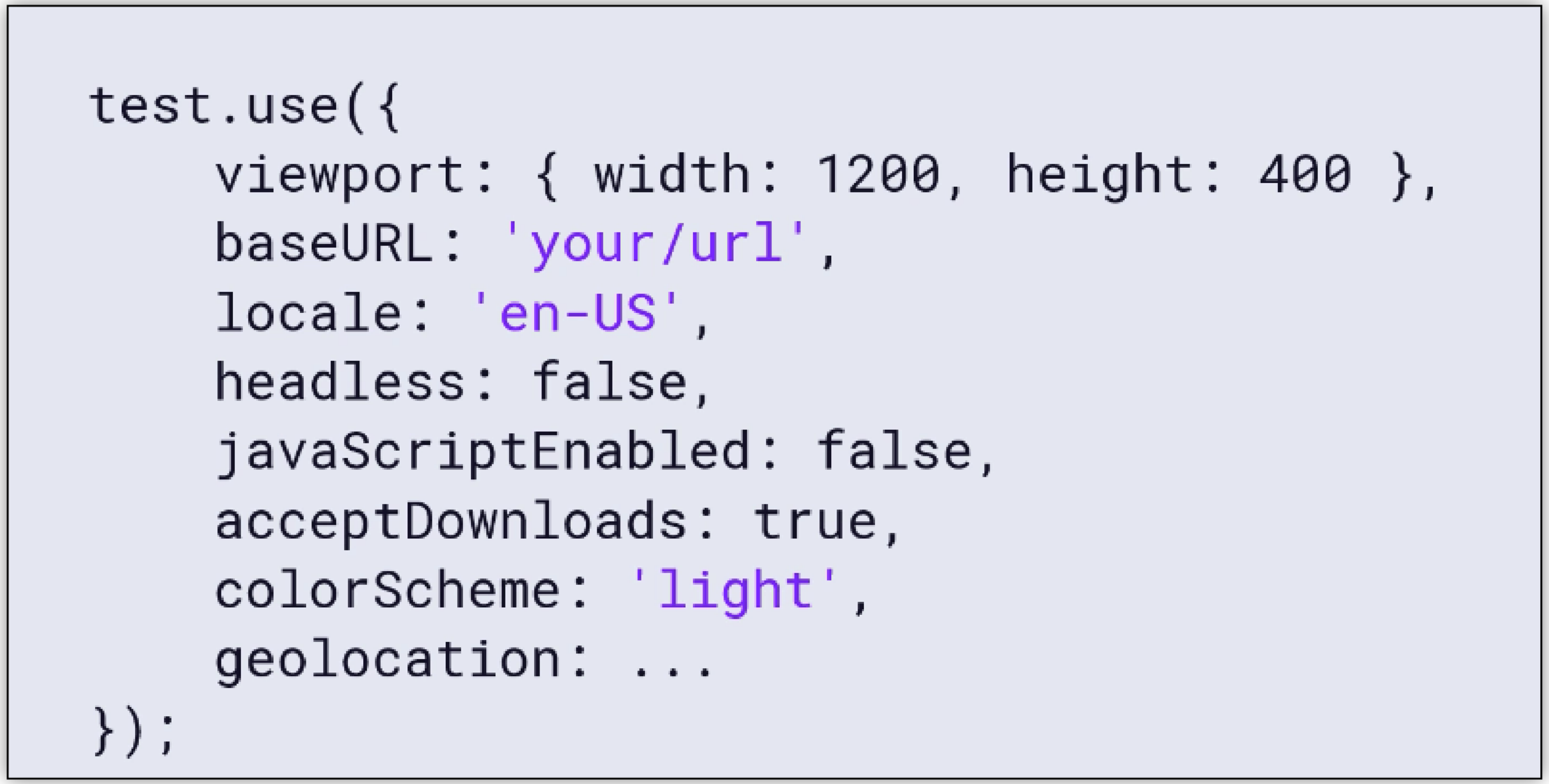


```
test.beforeAll('Set up', ...)

test('One', ...);

test('Two', ...);
```

Parametrize tests

```javascript
[
  { name: 'Alice', expected: 'Hello, Alice!' },
  { name: 'Bob', expected: 'Hello, Bob!' },
  { name: 'Charlie', expected: 'Hello, Charlie!' },

].forEach(({ name, expected }) => {
  // You can also do it with test.describe() or with multiple tests as long the test name is unique.
  test(`testing with ${name}`, async ({ page }) => {
    await page.goto(`https://example.com/greet?name=${name}`);
    await expect(page.getByRole('heading')).toHaveText(expected);
  });
});
```

use



```
test.use({
    viewport: { width: 1200, height: 400 },
    baseURL: 'your/url',
    locale: 'en-US',
    headless: false,
    javaScriptEnabled: false,
    acceptDownloads: true,
    colorScheme: 'light',
    geolocation: ...
});
```

Abschluss

Ende

Das war alles für dieses Kapitel