

WORKSHOP

TESTING IN JAVASCRIPT

OUTLINE

- ▶ JavaScript testing.
- ▶ Frontend testing (React).
- ▶ Backend testing (Express).

WHY TESTING

GREETING FUNCTION



```
const greeting = name => {  
  return `Hello, ${name}`  
}
```

SIMPLE WAY TO WRITE TEST



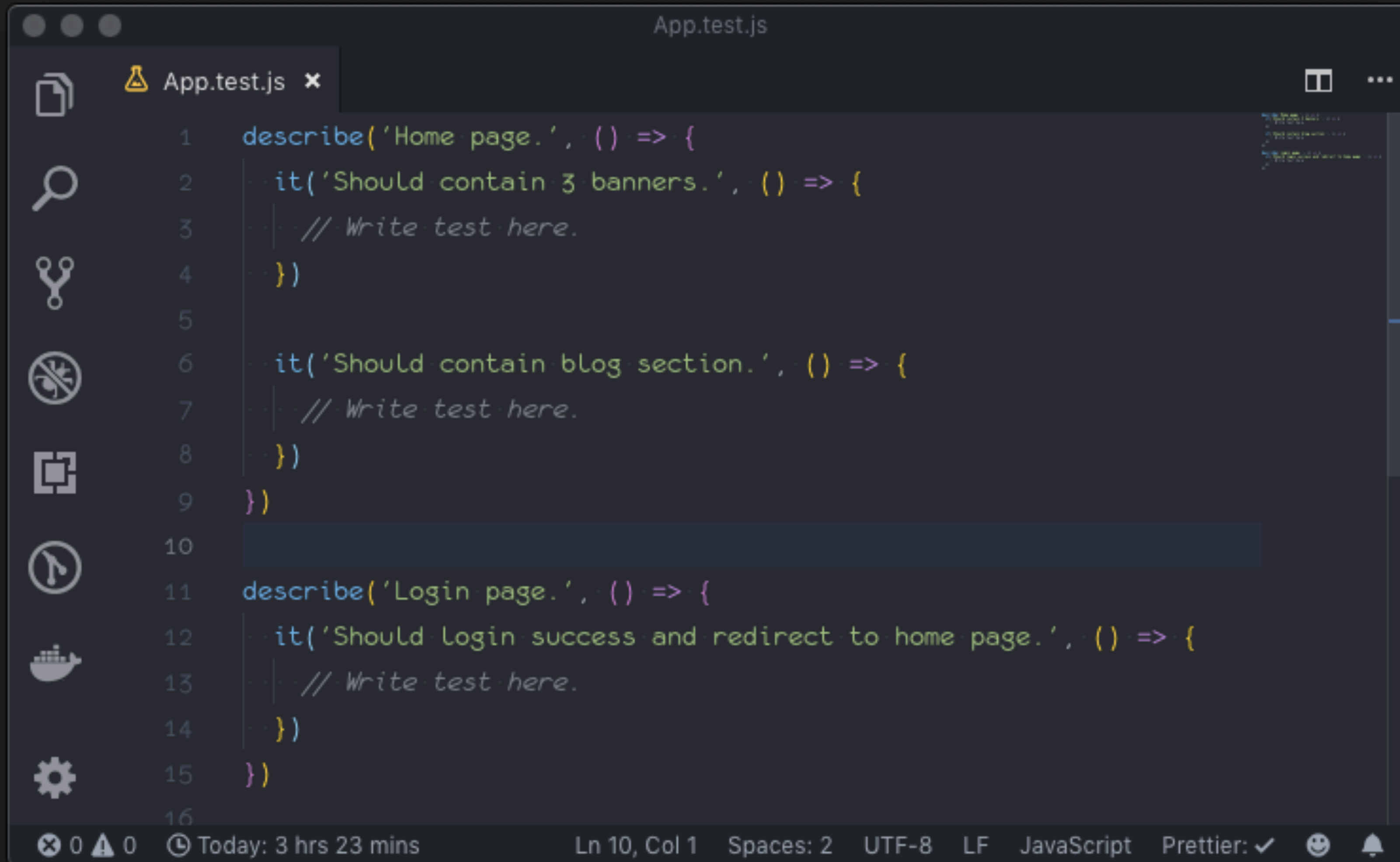
```
1 const input = 'John Doe'
2 const expect = 'Hello, John Doe'
3
4 const actual = greeting(input)
5
6 if (actual === expect) {
7   console.log('Correct!!')
8 } else {
9   console.log('Wrong!!')
10 }
11
```

TEST RUNNER

NICE REPORT

```
twitter — node ◀ node /usr/local/Cellar/yarn/1.17.0/lib...  
PASS src/App.test.js  
  Home page.  
    ✓ Should contain 3 banners. (3ms)  
    ✓ Should contain blog section.  
  Login page.  
    ✓ Should login success and redirect to home page.  
  
Test Suites: 1 passed, 1 total  
Tests:      3 passed, 3 total  
Snapshots:  0 total  
Time:       8.096s  
Ran all test suites related to changed files.
```

IT, DESCRIBE



```
App.test.js

1 describe('Home page.', () => {
2   it('Should contain 3 banners.', () => {
3     // Write test here.
4   })
5
6   it('Should contain blog section.', () => {
7     // Write test here.
8   })
9 })
10
11 describe('Login page.', () => {
12   it('Should login success and redirect to home page.', () => {
13     // Write test here.
14   })
15 })
16
```


USEFUL MATCHERS

- ▶ `expect(1+1).toEqual(2)`
- ▶ `expect(['apple', 'banana', 'orange']).toHaveLength(3)`
- ▶ `expect(onClick).toHaveBeenCalledTimes(1)`

More matcher - <https://jestjs.io/docs/en/expect>

TEST RUNNER

- ▶ Jest - <https://jestjs.io/>
- ▶ Mocha - <https://mochajs.org/>

CODING TIME

**WRITE TEST FOR
CALCULATOR MODULE**

TIME TO CODING

- ▶ Workshop repository - <https://github.com/aofleejay/testing-in-javascript>
- ▶ Write test for function `sum` inside `/calculator` directory.

WATCH MODE

- ▶ Run `yarn test --watch` or edit script in `package.json` to `yarn --watch`

TDD

TEST DRIVEN DEVELOPMENT

ABSOLUTE CALCULATOR

- ▶ New requirement “absolute input before summation”.
- ▶ Use absolute with command `Math.abs(-1)`.
- ▶ Use TDD approach!!!.

TEST COVERAGE

- ▶ Run `yarn test --coverage` to generate coverage report.
- ▶ Open `/coverage/lcov-report/index.html`

TESTING IN REACT

ENZYME

- ▶ Testing utilities for React
- ▶ Documentation - <https://airbnb.io/enzyme/>

ENZYME USEFUL FUNCTIONS

- ▶ `shallow, mount`
- ▶ `find('#id'), find('.class-name'), find('element')`
- ▶ `find('button').simulate('click')`
- ▶ `find('p').text()`

TIME TO CODING AGAIN

- ▶ Write test inside `/counter` and `/twitter` directory.

EXPRESS

BACKEND TESTING

MEDIUM API

- ▶ Get all post using - `/posts`
- ▶ Get post by id - `/post/001`

TRY DIFFERENT TOOLS

- ▶ Use [Mocha](#) instead of Jest.
- ▶ Use [Sinon](#) instead of Jest mock.
- ▶ Use [Chai](#) instead of Jest matchers.
- ▶ Use [Supertest](#) to simulate HTTP request.