单测接入

选型

- 方案比较：测试方案调研
- 方案选定：jest + React Testing Library

接入

安装依赖

```
yarn add -W react react-dom
```

```
yarn add -W -D typescript jest @types/jest jest-environment-jsdom ts-jest @testing-library/jest-dom
@testing-library/react @testing-library/react-hooks @testing-library/user-event @babel/core
@babel/preset-env @babel/preset-react @babel/preset-typescript
```

配置

jest

```
yarn jest --init
```

```
// jest.config.js
module.exports = {
  // Automatically clear mock calls, instances, contexts and results before every test
  clearMocks: true,

  // Indicates whether the coverage information should be collected while executing the test
  collectCoverage: true,

  // The directory where Jest should output its coverage files
  coverageDirectory: 'coverage',

  // An array of regexp pattern strings used to skip coverage collection
  coveragePathIgnorePatterns: ['/node_modules/'],

  // A set of global variables that need to be available in all test environments
  globals: {
    'ts-jest': {
      tsconfig: './tsconfig.json',
      babelConfig: './babel.config.js',
    },
  },

  // A preset that is used as a base for Jest's configuration
  preset: 'ts-jest', // 新的更快的包 esbuild-jest, 待调研

  // A list of paths to modules that run some code to configure or set up the testing framework before each test
  // setupFilesAfterEnv: ['jest-extended/all'],

  // The test environment that will be used for testing
  testEnvironment: 'jsdom', // 'node'

  // A map from regular expressions to paths to transformers
  transform: {
    '^.+\\.(t|j)sx?$': 'ts-jest',
    // "^.+\\.jsx?$": "babel-jest",
  },

  // An array of regexp pattern strings that are matched against all source file paths, matched files will skip transformation
  transformIgnorePatterns: [
    // '/node_modules/',
    "/node_modules/(?!(@byted)/)", // 需要编译的依赖
  ],
};

```

tsconfig

```
tsc --init
```

```
// tsconfig.json
{
  "compilerOptions": {
    /* Language and Environment */
    "target": "es5",                                     /* Set the JavaScript language version for emitted JavaScript and include compatible library declarations. */
    "jsx": "react",                                /* Specify what JSX code is generated. */

    /* Modules */
    "module": "commonjs",                                /* Specify what module code is generated. */

    /* JavaScript Support */
    "allowJs": true,                                  /* Allow JavaScript files to be a part of your program. Use the `checkJS` option to get errors from these files. */

    /* Interop Constraints */
    "esModuleInterop": true,                             /* Emit additional JavaScript to ease support for importing CommonJS modules. This enables `allowSyntheticDefaultImports` for type compatibility. */
    "forceConsistentCasingInFileNames": true,            /* Ensure that casing is correct in imports. */

    /* Type Checking */
    "strict": true,                                      /* Enable all strict type-checking options. */

    /* Completeness */
    "skipLibCheck": true                                 /* Skip type checking all .d.ts files. */
  },
  "exclude": ["node_modules"]
}
```

babel

```
touch babel.config.js
```

```
// babel.config.js
module.exports = {
  presets: [
    ['@babel/preset-env', { targets: { node: 'current' } }],
    ['@babel/preset-react', { runtime: 'automatic' }],
    '@babel/preset-typescript',
  ],
};
```

jest 介绍

> 会寻找项目目录下匹配 `**/__tests__/**/*.[jt]s?(x), **/?(*.)+(spec|test).[tj]s?(x)` 规则的文件，作为测试用例文件。

建议各个目录下建 `__tests__` 目录，单测文件命名 xxx.test.ts(x)

- Globals：全局方法或对象，不需手动引入
  - `afterAll`、`beforeAll`：钩子函数，只执行一次。
  - `afterEach`、`beforeEach`：钩子函数，每个单测都执行。
  - `describe`：测试集。可用于限制作用域。
  - `test/it`：单测函数。
- 断言 - Expect
- 异步测试
- Mock Functions
- 调试用：describe.only、describe.skip、test.only、test.skip、

```
import { render, waitFor } from '@testing-library/react';
import React from 'react';
import userEvent from '@testing-library/user-event';
import ButtonPromise from '../src';

describe('描述测试模块，buttonPromise button', () => {
  // 具体测试用例，一个测试模块内可以有多个测试用例
  test('描述测试功能：buttonpromise loading', async () => {
    // 该函数体为具体测试内容

    // 思路：渲染组件 -> 根据测试目的模拟用户行为 -> 断言结果 -> 根据测试目的模拟用户行为
    const btnClick = jest.fn(
      () =>
        new Promise(resolve => {
          setTimeout(() => resolve(''), 2000);
        }),
    );
```

```

      // 渲染组件
     const { container } = render(
       <ButtonPromise id="test-copy-icon" onClick={btnClick}>
         按钮
       </ButtonPromise>,
     );


     // 模拟用户行为
     const btn = container.querySelector('#test-copy-icon');
     userEvent.click(btn);
     userEvent.click(btn);
     userEvent.click(btn);

     // 出现 loading 类
     // 断言测试用例结果
     await waitFor(() => expect(btnClick).toHaveBeenCalledTimes(1));

     // loading 类消失

   });
 });

```

@test-library 工具库相关介绍

@testing-library/react

用于编写 React 组件相关的测试

- 测试一个组件，首先需要将其渲染出来，🌰

```
import {render} from '@testing-library/react'
import ButtonPromise from '../src/index.tsx';

render(<ButtonPromise />) // 渲染对应组件

```

- 渲染组件后则需要获取相关节点后再进行测试，@testing-library/react 暴露出一个 screen 实例用于通过各种方式获取节点。🌰

```
import {render, screen} from '@testing-library/react'
import ButtonPromise from '../src/index.tsx';

render(<div>test</div>);
// screen.getByText 可以通过传入 文本/正则/函数 获取元素
// screen.debug 类似与 console.log，可以用于调试打印已获取元素
screen.debug(screen.getByText('test'));

// 也可通过 render 函数返回解构的 container，调用 querySelector 方法进行 dom 原生查询
const { container } = render(<ButtonPromise id="test-btn" />)
screen.debug(container.querySelector('#test-btn'))
```

- 更多关于获取节点的方法

@testing-library/jest-dom

用于在获取到 dom 节点后编写 DOM 相关状态的测试，常用 api

toBeXXX (toBeDisabled、toBeEnable、toBeInTheDocument、toBeVisible...)，断言目标 Dom 应该处于什么状态，如此刻某个按钮应处于禁用状态

toHaveXXX (toHaveCalss、toHaveFocus、toHaveStyle、toHaveTextContent...)，断言目标此时应该具备什么形态，如 loading 时应该具备 loading 相关的类

🌰

```
import {render, screen} from '@testing-library/react'
import ButtonPromise from '../src/index.tsx';

const { container } = render(<ButtonPromise id="test-copy-icon" />);
const btn = container.querySelector('#test-copy-icon');
expect(btn).toBeEnabled(); // 此刻期望按钮是可点击的
```

@testing-library/user-event

用于编写模拟事件触发的测试

- 🌰 buttonPromise 有一个功能是当onClick事件返回promise时，将自动托管loading，可用于防止提交表单时多次点击导致触发提交。

```
import { render, waitFor } from '@testing-library/react';
import React from 'react';
import userEvent from '@testing-library/user-event';
import ButtonPromise from '../src';

test('测试 buttonPromise loading 效果', async () => {

 // 首先构造点击事件
  const submitForm = jest.fn(
    () =>
      new Promise(resolve => {
        setTimeout(() => resolve(''), 2000);
      }),
  );

  // 渲染组件
  const { container } = render(
    <ButtonPromise id="test-copy-icon" onClick={submitForm}>
      按钮
    </ButtonPromise>,
  );
  const btn = container.querySelector('#test-copy-icon');

  // 利用 useEvent 触发三次点击
  userEvent.click(btn);
  userEvent.click(btn);
  userEvent.click(btn);

  // 根据该组件功能，期望 submitForm 只被调用一次
  await waitFor(() => expect(btnClick).toHaveBeenCalledTimes(1));
});

```

- 更多用户事件触发模拟可点击查看

@testing-library/react-hooks

用于编写 React Hook 相关的测试，当 hook 很难通过组件功能进行测试或 hook 较为复杂，组件难以完整测试时使用。

官方 demo

其他工具

- Vscode 插件 - jest
- 第三方断言库 jest-extended
  - 安装依赖

```
yarn add -W -D jest-extended
```

  - 配置 transform

```
// jest.config.js
module.exports = {
    ...
    setupFilesAfterEnv: ['jest-extended/all'],
    ...
}
```

- eslint 插件
  - 安装依赖

```
yarn add -W -D eslint-plugin-testing-library
```

  - 添加 eslint 配置

```
// .eslintrc.js
module.exports = {
  ...
  overrides: [
    {
      files: ['**/*.test.[jt]sx?'],
      extends: ['plugin:testing-library/react'],
    },
  ],
  ...
};

```