用友 yonyou

前端白盒测试（单测）如何布道

朱子箭

2023-09-15

目录

CONTENT

用友 yonyou

前端现代化测试模型

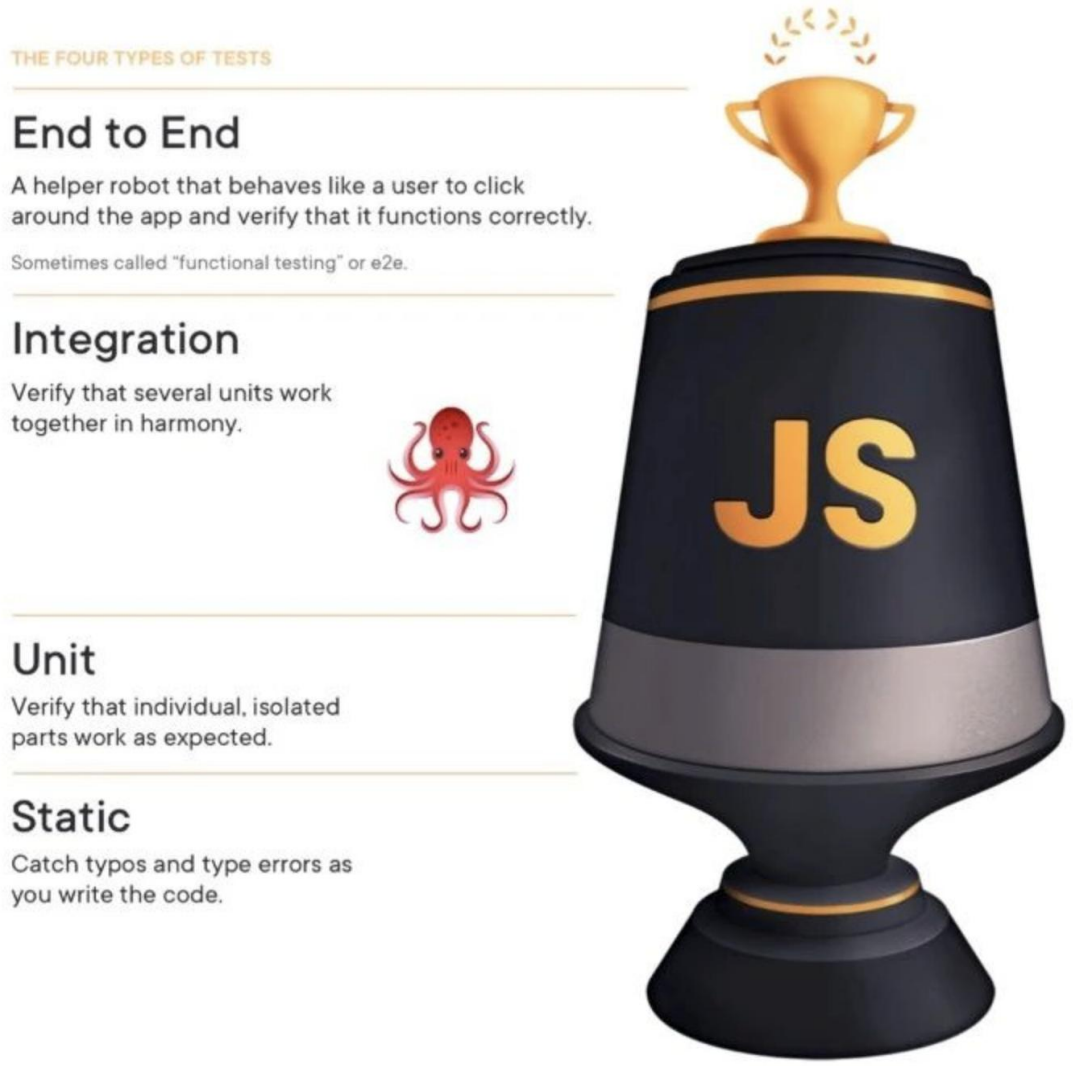
【e2e测试】
模拟用户在真实环境上操作行为(包括网络请求、获取数据库数据等)的测试。(代表库：cypress)

【集成测试】
模拟用户的行为进行测试，对网络请求、获取数据库的数据等依赖第三方环境的行为进行 mock。(代表库：jest、react—testing—library)

【单元测试】
在奖杯模型中，单元测试的职责是对一些边界情况或者特定的算法进行测试。(代表库：jest、mocha)

【静态测试】
在编写代码逻辑阶段时进行报错提示。(代表库：eslint、flow、TypeScript)

前端现代化测试模型



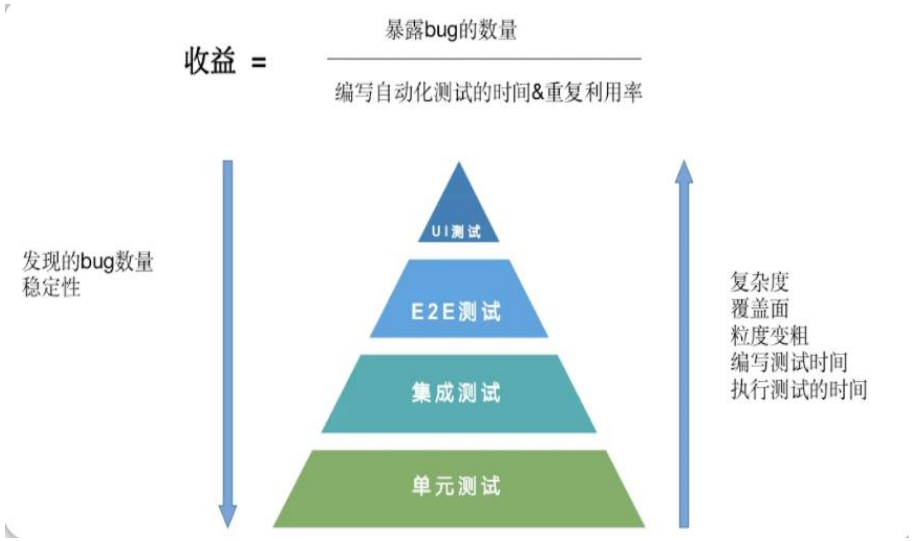
深刻认识单测的价值

一个组件如果支持场景5个，自测和单测成本对比

- 如果手动写Demo测试一遍需要 3 分钟，支持场景有5个，Demo测试编写总投入15分钟
- 如果单元测试可能每个场景需要投入30分钟，5个场景总投入150分钟

到底值不值？

假如这个组件对5个场景中任意一个场景commit改动，改N次，每次都通过手动写Demo测试 需3 分钟，那么就得投入3*5*N，如果这个组件的commit改动N>10次，就已经投入了150分钟的成本，而且还不能保证每个人每次commit都会把所有场景都测试到位。



当单测写好后，无论进过多少人之手，对其改动了多少次commit，每次直接跑单测自动化测试，无论跑成百上千次也只是之前投入的150分钟成本，并且这个测试结果还是稳定可靠。

- 承认单元测试成本是挺高，但写一次，自动测试 N 次，收益也很大
- 对于逻辑比较稳定，场景比较多样，单测必要性非常高，而不是靠每个人都自律的测全所有场景
- 单测非常适用于组件库里的组件、hooks 库里的 hooks、一些工具函数等，对发版的向下兼容性、稳定性具有很强的保障

前端测试框架技术栈选型

[jest] + [enzyme] （不推荐）

- 适应2016～2021年份的程序
- 官方已经不再活跃维护 Enzyme，且没有计划支持最新的 React 18

[jest] + [@testing-library/react] + [cypress] （三大框架推荐）

- 适应2021年份以后的程序
- testing-library 关注于集成测试的设计理念使得团队可以更轻易、高效地写出易于维护的测试代码。
- testing-library 鼓励从用户实际使用角度写测试用例，因此其 API 设计理念无形中就引入很多 UI 测试的最佳实践
- 成熟活跃的社区及轻量的实现机制保证了该测试框架可预期的长久生命力
- cypress模拟用户行为，更利于组件库的视觉回归测试（样式测试）

jest 基础用法

describe 是一套测试用例，它包含多个单元测试用例

test 是单个单元测试块，它拥有描述和划分范围的作用，即它代表我们要为该计算函数

expect 是一个断言，它表达一个计算结果，并将期望值输出。

toBe 是一个匹配器，用于检查期望值，如果不符合预期结果则应该抛出异常。

fn 是模拟一个函数计算，用于断言对函数执行有关的期望结果。

安装测试框架 npm install --save-dev jest

```
//新建mytest.spec.js，编写测试用例代码
describe("测试 map 回调函数执行情况", () => {
  test("map [1, 2, 3], 回调函数执行 3 次", () => {
    const mockFn = jest.fn((x) => x * 2);
    map([1, 2, 3], mockFn);
    expect(mockFn.mock.calls.length).toBe(3);
  });

  test("map [1, 2, 3], 回调函数返回 2, 4, 6", () => {
    const mockFn = jest.fn((x) => x * 2);
    map([1, 2, 3], mockFn);
    expect(mockFn.mock.results[0].value).toBe(2);
    expect(mockFn.mock.results[1].value).toBe(4);
    expect(mockFn.mock.results[2].value).toBe(6);
  });
});
```

执行测试用例 node jest mytest.spec.js

jest 常用断言

expect(value)：要测试一个值进行断言的时候，要使用expect对值进行包裹

---

toBe(value)：使用Object.is来进行比较，如果进行浮点数的比较，要使用toBeCloseTo
not：用来取反
toEqual(value)：用于对象的深比较
toMatch(regexpOrString)：用来检查字符串是否匹配，可以是正则表达式或者字符串
toContain(item)：用来判断item是否在一个数组中，也可以用于字符串的判断
toBeNull(value)：只匹配null
toBeUndefined(value)：只匹配undefined
toBeDefined(value)：与toBeUndefined相反
toBeTruthy(value)：匹配任何使if语句为真的值
toBeFalsy(value)：匹配任何使if语句为假的值
toBeGreaterThan(number)：大于
toBeGreaterThanOrEqual(number)：大于等于
toBeLessThan(number)：小于
toBeLessThanOrEqual(number)：小于等于
toBeInstanceOf(class)：判断是不是class的实例
anything(value)：匹配除了null和undefined以外的所有值
resolves：用来取出promise为fulfilled时包裹的值，支持链式调用
rejects：用来取出promise为rejected时包裹的值，支持链式调用
assertions(number)：验证在一个测试用例中有number个断言被调用

```
test('测试遍历过程中不能出现0值', () => {
  for (let a = 1; a < 10; a++) {
    for (let b = 1; b < 10; b++) {
      expect(a + b).not.toBe(0);
    }
  }
});
```

jest Mock函数

jest.fn(implementation)：返回一个全新没有使用过的mock function，这个function在被调用的时候会记录很多和函数调用有关的信息

```
const mockFn = jest.fn((a,b)=>a+b);
mockFn(10,100);
expect(mockFn).toHaveBeenCalled();                    // 断言这个函数被调用过
expect(mockFn).toHaveBeenTimes(1);                    // 断言这个函数被调用过1次
expect(mockFn).toHaveBeenWith(10,100);                // 断言这个函数调用的参数列表值
```

jest.spyOn(object，methodName)：返回一个mock function同jest.fn相似，但不会执行原始函数的代码，仅用于追踪object[methodName]函数调用信息

jest.fn 和 jest.spyOn 的主要区别：jest.fn 会创建一个全新的函数，而 jest.spyOn 则会替代现有的函数。

因此，如果你在测试中需要模拟一个函数的行为，那么应该使用 jest.fn；如果你需要追踪一个函数的行为，那么就使用 jest.spyOn。

@testing-library/react

render：渲染组件，返回 container 容器 dom 和其他的操作、查询对象

```
import {render} from '@testing-library/react'
const {container,baseElement,debug,rerender,unmount} = render(<Table {...props} />)
```

container：这是一个DOM元素，它表示渲染组件的容器。所有的组件都会被渲染到这个容器中。在测试中，你可以通过这个容器来查询和操作组件。

baseElement：这也是一个DOM元素，它表示测试的根元素。所有的测试都应该在这个元素下进行。

如果在单测中需要操作和查询组件，使用container更合适
如果在整个页面范围内进行选择和操作，使用baseElement更合适

unmount：调用后，会卸载（删除）当前渲染的组件

debug：调用后，在控制台中输出当前渲染的组件树，常用于调试组件。

rerender：调用后，会重新渲染当前的组件，常用于组件状态更新或者props改变时。

@testing-library/react

screen：查找Dom的操作方法，建议用其代替 container 容器的 dom操作、查询对象

screen.getBy···：返回查询的匹配节点，如果没有元素匹配或找到多个匹配，则抛出描述性错误(如果期望找到多个元素，则使用getAllBy)。
screen.queryBy···：返回查询的匹配节点，如果没有元素匹配则返回null。这对于断言不存在的元素很有用。如果找到多个匹配项，则抛出错误(如果可以，请使用queryAllBy)。
screen.findBy···：返回一个Promise，该Promise在找到与给定查询匹配的元素时进行解析。如果未找到任何元素，或者在默认超时1000ms后找到多个元素，则拒绝承诺。如果需要查找多个元素，请使用findAllBy。

fireEvent：触发某个元素的事件

```
import { render, screen, fireEvent } from '@testing-library/react';
test('clicks a button', () => {
  render(<Button>click</Button>);
  const button = screen.getByRole('button');
  const button = screen.getByText(/click/i);
  expect(button).toBeInTheDocument();
  fireEvent.click(button);
});
```

@testing-library/react

renderHook：渲染一个hook函数组件，返回result和rerender对象

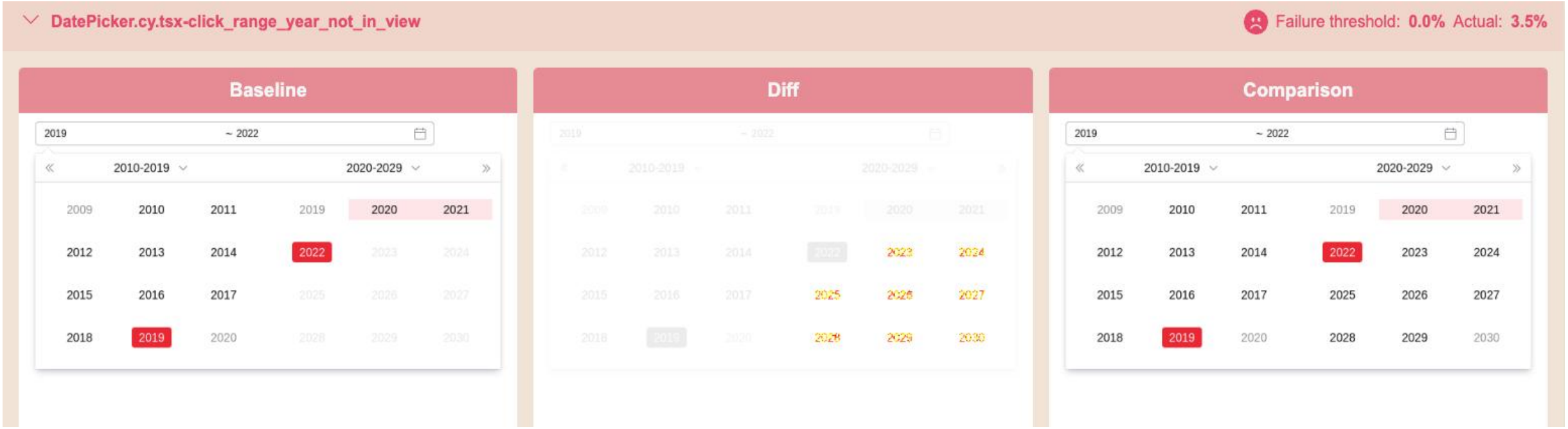
```
import {renderHook} from '@testing-library/react'

const {result,rerender} = renderHook((props) => {
   const [name, setName] = useState('')
   React.useEffect(() => {
      setName('Alice is '+(props.code || 'null'))
   }, [])
   return name
})

expect(result.current).toBe('Alice is null')

rerender({code: "ID001"})

expect(result.current).toBe('Alice is ID001')
```

result：需要通过current拿到结果值，再进行断言，类似于ref

rerender：使用新props再次渲染hook函数组件

cypress E2E测试(模拟用户真实操作)

TinperNext中使用cypress的效果



上次一次基线版本

对比出现差异区域

当前改动后的版本

适用于会影响到样式的属性
适用于用户交互后样式发生变化的快照过程记录

依赖cypress—image—diff—js包

cypress 基础用法

```
// React 18
import cy,{ mount } from 'cypress/react18'
// React 16, 17
import cy,{ mount } from 'cypress/react'
```

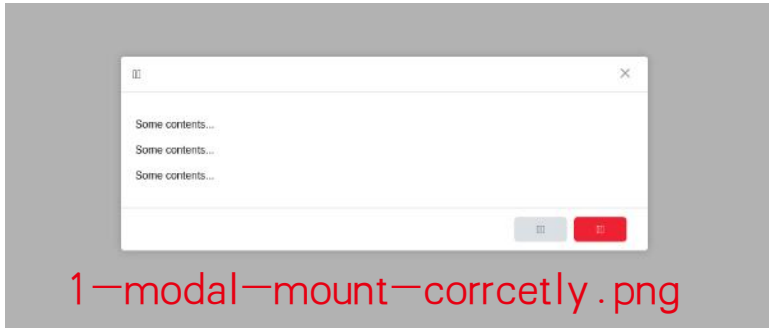
cy.mount（React组件装载）

cy.get（获取dom）

cy.click （执行动作）

cy.compareSnapshot （截屏并对比）

```
it('should mount correctly and closable', () => {
    cy.mount(<Modal visible closable>
        <p>Some contents...</p>
        <p>Some contents...</p>
        <p>Some contents...</p>
    </Modal>).then(() => {
        cy.get('.close').should('be.visible').then(() => {
            cy.compareSnapshot( email: '1-modal-mount-correctly')
            cy.get('.close').click()
            cy.compareSnapshot( email: '1-modal-closed')
        })
    })
});
```



1—modal—mount—corrcetly.png

cypress 穷举样式

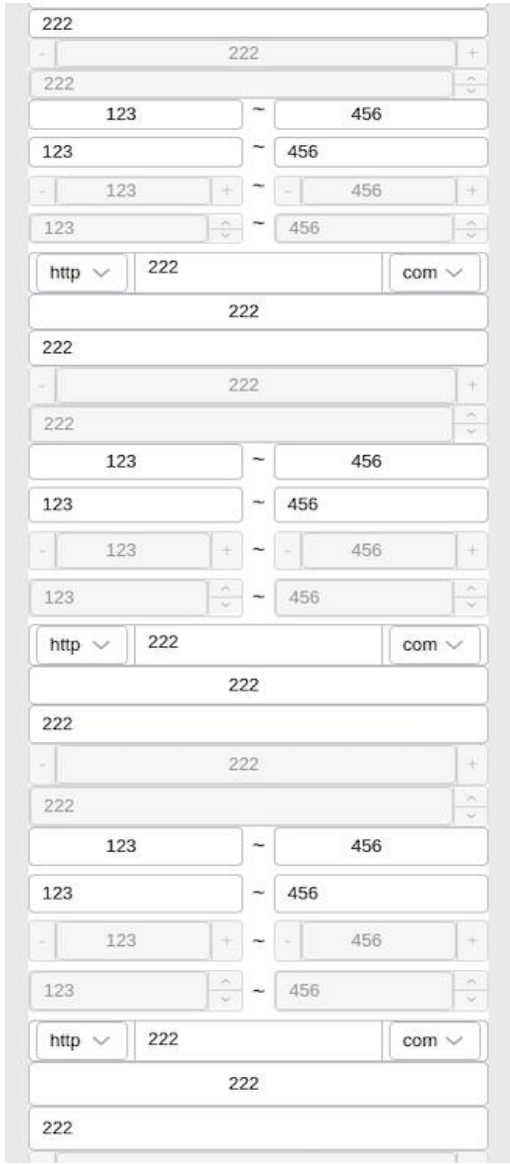
TinperNext中使用cypress的效果

```
const sizeArr: any = ['xs', 'sm', 'md', 'nm', 'lg', 'default', 'small', 'middle', 'large'];
const InputSizeDemo = (props: any) => {
    let comps: JSX.Element[] = [];
    sizeArr.forEach((size: any) => {
        comps.push(
            <>
                <Input size={size} {...props} />
                <Input size={size} {...props} type='search' />
                <Input size={size} {...props} type='password' />
                <Input size={size} {...props} disabled />
                <Input size={size} {...props} disabled type='search' />
                <Input size={size} {...props} disabled type='password' />
                <Input size={size} {...props} disabled enterButton />
                <Input size={size} {...props} disabled prefix='前缀' suffix='后缀' />

                <InputGroup size={size} {...props} style={{marginBottom: '5px', width: '50%'}}>
                    <InputGroup.Button shape='border'>
                        <Dropdown overlay={menu1} visible>
                            <Button shape='border'>
                                带有分割线的下拉 <i className='uf uf-arrow-down'> </i>
                            </Button>
                        </Dropdown>
                    </InputGroup.Button>
                    <Input type='text' size={size} {...props} />
                    <InputGroup.Button>
                        <Button icon='uf-search'></Button>
                    </InputGroup.Button>
                </InputGroup>
            </>
        );
    });
    return <>{comps}</>;
};

describe('input', () => {
    // 默认态、禁用输入框，仅输入框/有值
    it('should mount basic picker input', () => {
        cy.mount(<InputList />)
        cy.wait(200)
        cy.compareSnapshot( email: 'basic_input')
        cy.mount(<InputSizeDemo placeholder='请输入' />)
        cy.wait(200)
        cy.compareSnapshot( email: 'basic_input_size')
        cy.mount(<InputSizeDemo value='abc' allowClear />)
        cy.wait(200)
        cy.compareSnapshot( email: 'basic_input_clear_size')
        cy.mount(<InputSizeDemo bordered='bottom' />)
        cy.compareSnapshot( email: 'basic_input_border_bottom')
        cy.mount(<InputSizeDemo bordered='bottom' value='abc' allowClear/>)
        cy.compareSnapshot( email: 'basic_input_clear_border_bottom')
    })
```

可通过写一段遍历代码

穷举所有希望组合的场景

形成到一张图片中



cypress 常用操作

用友 yonyou

Dom相关操作方法

```
cy.get('.item').eq(0)

cy.get('.text').children()

cy.get('tr').filter('.selected')

cy.get('.list).find('.active')

cy.get('.list  .item').first()

cy.get('.list  .item').last()

cy.get('input').not('.required')
```

```
cy.get('.text').parent()

cy.get('.text').parents()

cy.get('tr.avtive).prev()

cy.get('tr.active').prevAll()

cy.get('.list  .item:first').next()

cy.get('.active').nextAll()

cy.get('.text').contains('Home')
```

触发事件方法

```
cy.get('.button).click()

cy.get('.button').dblclick()

cy.get('.menu').rightclick()

cy.get('.footer').scrolllntoView()

cy.get('.sidebar').scrollTo('bottom')

cy.get('[type="checkbox"]').check()

cy.get('[type="checkbox"]').uncheck()

cy.get('select').select('option-1')

cy.get('.button).trigger('mousedown')
```

cypress 常用断言

用友 yonyou

常用断言操作方法

```
cy.get('.err').should('be.empty')
```

```
cy.get('.err').should('be.empty').and('be.hidden')
```

```
.should('equal', 10)
expect(myVar).to.equal(10)
```

```
.should('not.equal', 'Jane')
expect(name).to.not.equal('Jane')
```

```
.should('deep.equal', { name: 'Jane' })
expect(obj).to.deep.equal({ name: 'Jane' })
```

```
.should('exist')
expect(myVar).to.exist
```

```
.should('have.all.keys', 'name', 'age')
expect(arr).to.have.all.keys('name', 'age')
```

```
.should('have.any.keys', 'age')
expect(arr).to.have.any.keys('age')
```

```
.should('include', 'c')
expect(['a','b','c']).to.include('c')
```

```
.should('to.match', /^test/)
expect('testing').to.match(/^test/)
```

```
.should('be.true')
expect(true).to.be.true
```

```
.should('be.false')
expect(false).to.be.false
```

```
.should('be.null')
expect(null).to.be.null
```

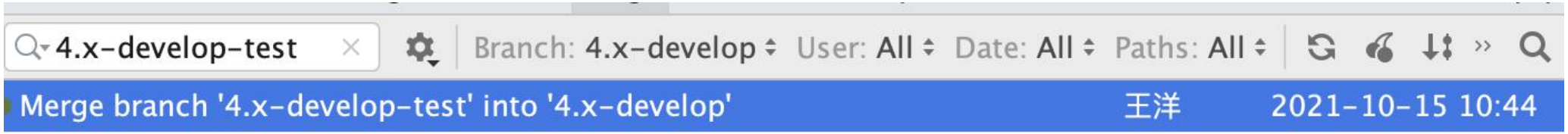
```
.should('be.undefined')
expect(undefined).to.be.undefined
```

```
.should('be.empty')
expect([]).to.be.empty
```

TinperNext单测覆盖90%历史回顾

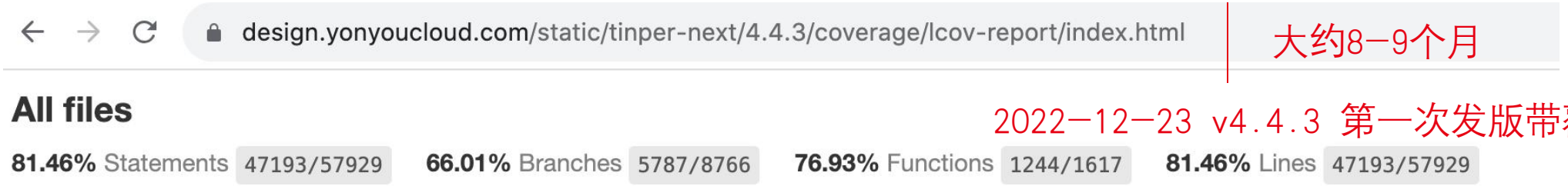
“单测覆盖率到达>90%”实际投入远不止8个月

各团队需要做持续输血/打鸡血的准备



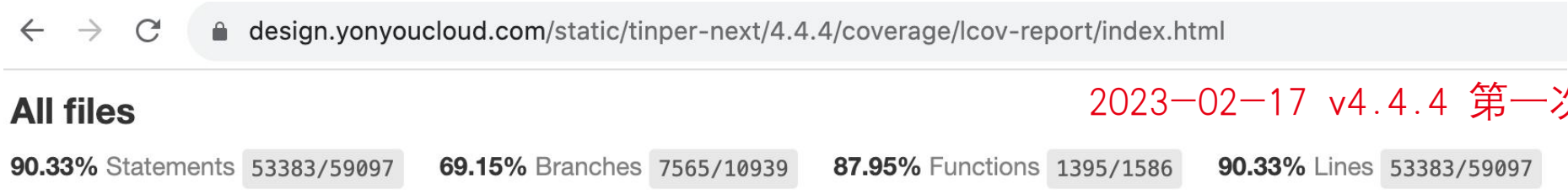
2021—10—15 第一次单测合到开发主干分支

2022—03—21 第一次单测提供测试覆盖率报告



大约8—9个月

2022—12—23 v4.4.3 第一次发版带覆盖报告超80%



2023—02—17 v4.4.4 第一次发版覆盖率超90%

TinperNext如何从enzyme快速迁移到@testing-library

```
const app = mount(<App />)
const todoList = app.find('li').length
app.find('button.delete').at(0).simulate('click')
```

- .get(index)：返回指定位置的子组件的DOM节点
- .at(index)：返回指定位置的子组件
- .first()：返回第一个子组件
- .last()：返回最后一个子组件
- .type()：返回当前组件的类型
- .text()：返回当前组件的文本内容
- .html()：返回当前组件的HTML代码形式
- .props()：返回根组件的所有属性
- .prop(key)：返回根组件的指定属性
- .state([key])：返回根组件的状态
- .setState(nextState)：设置根组件的状态
- .setProps(nextProps)：设置根组件的属性

enzyme常用API

基于testing-libary实现

```
import React from 'react';
import {render, screen, fireEvent} from "@testing-library/react";
const match = (selector, element) => {...}
const getStyleFromString = (styleString) => {...}
const eventTypeMap = {"compositionend": "compositionEnd"...}
export const mount = (component, isDebug) => {
    const {container, rerender, baseElement, unmount, debug} = render(component);
    let find
    const reactWrapper = (node1) => {...};
    find = (matcher, container2) => {

        let nodes = []
        if (container2) {...} else {
            nodes = baseElement.querySelectorAll(matcher);
        }
        return {
            // ...obj,
            ...reactWrapper(nodes[0]),
            length: nodes.length,
            some: (selector) => {...},
            exists: (selector) => {...},
            text: () => {...},
            first: () => {...},
            children: () => {...},
            at: (atIndex) => {...},
            last: () => {...},
            instance: () => {...},
            getDOMNode: () => {...},
            prop: (v) => {...},
            update: () => undefined,
            find: (matcher) => find(matcher, nodes[0]),
            hasClass: (cls) => {...},
            simulate: (eventType, ...options) => {...},
            forEach: (callBack) => {...}
        };
    }
    // mount 的返回值
    return {
        ...reactWrapper(container.children[0]),
        find,
        html: () => container.innerHTML,
        setProps: (props) => {
            const updatedComponent = React.cloneElement(component, props);
            rerender(updatedComponent);
        },
        unmount: () => unmount(),
        update: () => undefined,
        instance: () => {...},
        exists: (selector) => {...},
        debug: debug,
        props: () => {...},

    };
};
```

TinperNext的jest.config.js

```
const path = require("path")
const rootDir = path.resolve(__dirname, "../")
const rootConfig = require("./test/root-config")
module.exports = {
    testEnvironment: 'jsdom',
    testTimeout: 20000,
    setupFiles: ["<rootDir>/next-ui-library/test/jestsetup.js"],
    roots: rootConfig,
    moduleNameMapper: {
        '@tinper/next-ui': '<rootDir>/packages/index.tsx',
    },
    moduleFileExtensions: [...],
    moduleDirectories: [
        "<rootDir>/next-ui-library/node_modules",
    ],
    transform: {
        "\\.[jt]sx?$": "<rootDir>/next-ui-library/test/jest.preprocess.js"
    },
    snapshotSerializers: [...],
    transformIgnorePatterns: [
        "node_modules/(?!lodash-es)"
    ],
    rootDir: `${rootDir}`,
    collectCoverageFrom: [
        "**/packages/**/*.{js,jsx,tsx}",
        "!**/i*.tsx",
        "!**/test/**",
        "!**/src1/**",
        "!**/src2/**",
        "**/index.tsx",
        "!**/demo/**",
        "!**/SwipeableInkTabBar.tsx",
        "!**/SwipeableTabBarNode.tsx",
        "!**/SwipeableTabContent.tsx",
        "!**/InkTabBar.tsx",
        "!**/ScrollableTabBar.tsx",
        "!**/TabBar.tsx",
        "!**/IframeUploader.tsx",
```

```
    coverageProvider: "v8",
    coverageDirectory: "<rootDir>/next-ui-library/coverage",
    reporters: [
        'default',
        [
            '<rootDir>/next-ui-library/node_modules/jest-html-reporters',
            {
                pageTitle: '多语专项测试报告',
                publicPath: '<rootDir>/next-ui-library/coverage',
                filename: 'langReport.html',
                openReport: false,
                expand: true,
                hideIcon: true,
                includeConsoleLog: true,
                // customInfos: [{title: '未测属性', value: restApis, }]
            }
        ]
    ]
}
```

自定义场景化的单测报告

语法规则转化的配置，例如：babel相关插件

无需统计单测覆盖率的文件排除，例如：历史/废弃代码）

TinperNext按场景抽取的部分单测工具方法

testStyle：测试组件的style属性是否正常

```
testStyle({
    title: 'component: Tree, <test prop:: rootStyle>',
    Component: Tree,
    attr: {
        children: <TreeNode title="1" />,
        rootStyle: {
            color: 'red'
        }
    },
    selector: `.${prefixTree}`,
    style: {
        color: 'red'
    }
})      You, 5 months ago • [feat] (test) 补充 部分用例 …
```

1.title： 测试的标题
2.Component：被测试的组件
3.attrs：测试的属性输入(包含其他依赖属性)
4.selector 属性传入后变化的dom选择器
5.Style：断言生成的Style

TinperNext按场景抽取的部分单测工具方法

testChildren测试children是否有被生成

```
testChildren({
    title: 'component: Steps, <test prop:: children>',
    Component: Steps,
    attrs: {
        children: <Steps.Step />
    },
    selector: `.${prefixSteps}-item`
})
```

1.title： 测试的标题
2.Component： 被测试的组件
3.attrs： 测试的属性children输入(包含其他依赖属性)
4.selector children属性传入后被断言的dom选择器

TinperNext按场景抽取的部分单测工具方法

eventsTest 测试组件内各种事件是否触发及参数列表值是否正确

```
eventsTest({
    title: 'component: Tree, <test prop:: onDoubleClick>',
    Component: TreeDemo,
    propFuncName: 'onDoubleClick',
    dependentProps: {},
    selector: `.${prefixTreeNode}-content-wrapper`,
    eventName: 'doubleclick',
    eventArgs: ['0-0'],
    afterTest: (mockEvent) => {
        expect(mockEvent.mock.calls[0][1].node.props.title).toBe('parent 1')
    }      You, 7 months ago • [test] (tree) 增加 用例 …
});
```

1. title： 测试用例的标题
2. propFuncName：测试的方法名称
3. Component：用来测试的组件 可以是基本组件，也可以是封装后的组件
4. dependentProps 测试方法2 过程中需要的依赖属性
5. eventName：触发方法2 的前提触发dom的方法名称（触发了dom的此方法 代码中方法2 才被触发）
6. selector 在 5 中指的用户触发的dom
7. eventArgs 在触发了方法2 时 回调的参数（断言参数值）
8. afterTest 在触发这个方法后执行的回调（此方法实现在公共方法用例中个性化的希望测试值的定制）

TinperNext按场景抽取的部分单测工具方法

testRootDomClsPrefix、testPropClassName、testPropFieldid
批量测试 组件的 clsPrefix、className，fieldid属性书否生效

```
{Menu: Menu, rootSelector: `ul.${prefix}-menu-root`, fieldidSelector: `.${prefix}-menu`, classNameSelector: `ul.${prefix}-menu-root`},
{Modal: Modal, attr:{visible: true}, classNameSelector: `.${prefix}-modal-dialog`, rootSelector: `.${prefix}-modal`, fieldidSelector: `div[role=dialogRoot]`},
{Pagination: Pagination},      You, 16 months ago • [feat] (test) 全局fieldid 外层测试代码 …
{Progress: Progress},
{Select: Select},
{Slider: Slider},
{Spin: Spin, fieldidSelector: `.${prefix}-spin-default`, rootSelector: `.${prefix}-spin-backdrop`, attr: {spinning: true}, classNameSelector: `.${prefix}-spin`},
{Steps: Steps},
{Svgicon: Svgicon, rootSelector: `.${prefix}-svgicon`, classNameSelector: `.${prefix}-svgicon`},
{Table: Table},
{Tabs: Tabs},
{Timeline: Timeline, attr: {children: [
    <Timeline.Item>Create a services site 2015-09-01</Timeline.Item>,
    <Timeline.Item>Solve initial network problems 2015-09-01</Timeline.Item>,
    <Timeline.Item>Technical testing 2015-09-01</Timeline.Item>,
    <Timeline.Item>Network problems being solved 2015-09-01</Timeline.Item>
]}},
{Timepicker: Timepicker},
{Transfer: Transfer},
{Tree: Tree},
{Treeselect: Treeselect, fieldidSelector: `.${prefix}-select-arrow-icon`},
{Upload: Upload, fieldidSelector: `div.${prefix}-upload-select`},
```

1. name： 测试标题的部分唯一内容
2. Component： 被测试的组件
3. attrs： 测试的初始化依赖的属性
4. selector 属性传入后变化的dom选择器
5. 测试方法中通过不同的断言判断属性是否生效

```
RootDomClsPrefixComponentsArray.forEach(obj => {
    const objArr = Object.entries(obj)[0];
    testRootDomClsPrefix({name: objArr[0], Component: objArr[1],selector: obj.rootSelector, attr: obj.attr});
    testPropClassName({name: objArr[0], Component: objArr[1], selector: obj.classNameSelector, attr: obj.attr})
    testPropFieldid({name: objArr[0], Component: objArr[1], selector: obj.fieldidSelector, attr: obj.attr})
})
```

TinperNext按场景抽取的部分单测工具方法

attrsTest：测试组件属性是否会在对应dom上生成对应类名

```
attrsTest({
    title: 'component: Alert, <test prop:: showIcon>',
    Component: Alert,
    attrs: {
        showIcon: true
    },
    selector: `i`,
    classnames: ["show-icon"]
});      liushld, 17 months ago • fix: eslint调整
```

1.title： 测试的标题
2.Component：被测试的组件
3.attrs：测试的属性
4.selector 属性传入后变化的dom选择器名称
5.classnames：选择器dom上是否包含此项类名

TinperNext按场景抽取的部分单测工具方法

attrsTestByLength　测试属性配置后产生的对应dom数量是否变化

```
attrsTestByLength({
    title: 'component: Drawer, <test prop:: extra>',
    Component: Drawer,
    attrs: {
        show: true,
        extra: true
    },
    selector: `.${prefixDrawer}-extra`,
    nodeCount: 1
});
```

1. title：　测试的标题
2. Component：被测试的组件
3. attrs：测试的属性
4. selector　属性传入后变化的dom选择器名称
5. nodeCount：选择器选择dom的数量

TinperNext按场景抽取的部分单测工具方法

testCustomeText 测试传入自定义文本属性 是否在对应Dom上显示

```
testCustomeText({
    title: 'Comment: Pagination, <test prop:: total>',
    Component: Pagination,
    attrs: {
        locale: 'en-us',
        pageSize: 43,
        activePage: 2,
        total: 100,
        last: false,        gong yujing, 11 months ago • [
        next: false
    },
    selector: `.${prefixPagination}-total`,
    text: 'Total100items'
});
```

1.title： 测试的标题
2.Component： 被测试的组件
3.attrs： 测试的属性输入(包含其他依赖属性)
4.selector dom选择器
5.选择的dom内文字的内容断言

TinperNext按场景抽取的部分单测工具方法

testComponentMethod 并生成用于测试组件内部方法的执行记录（执行次数和时间）

```
let resultArr = CopmArray.map(({ compName, Component, props, methodNameList, times, time, afterTest }) => testComponentMethod(
    `${compName} Performance Test`,
    compName,
    Component,
    methodNameList || ['render', 'componentDidMount'],
    props,
    times,
    time,
    afterTest
))
afterAll(() => {
    generateReport(resultArr);
});      江卫星, 5 months ago • [test] 增加组件的性能单元测试
```

1. compName： 测试的标题部分内容
2. Component： 被测试的组件
3. methodNameList： 被测试的方法列表
4. props 依赖的属性
5. times： 预期执行的次数
6. time： 预期执行方法的时间
7. afterTest 组件自定义定制断言

TinperNext单测报告

用友 yonyou

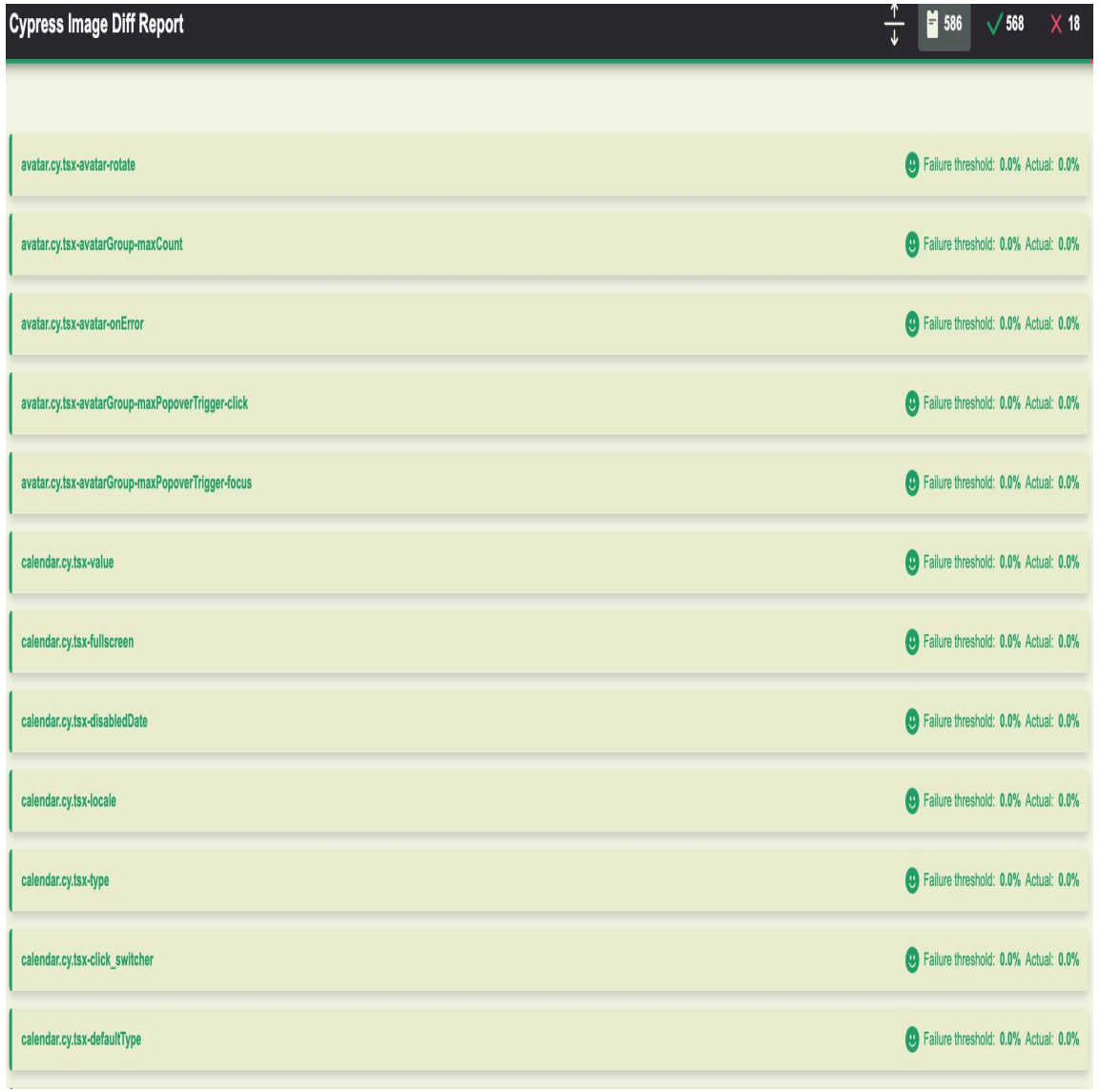
All files

90.08% Statements `56513/62731` **69.51%** Branches `9845/14162` **90.49%** Functions `1827/2019` **90.08%** Lines `56513/62731`

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

File	Statements
packages	100%
packages/wui-button-group/src	100%
packages/wui-calendar/src/i18n	100%
packages/wui-calendar/src/util	100%
packages/wui-datepicker/src/locale	100%
packages/wui-drawer/src/common	100%
packages/wui-menu/src/config-provider	100%
packages/wui-skeleton/src	100%
packages/wui-spin/src	100%
packages/wui-svgicon/src	100%
packages/wui-transition/src/util	100%
packages/wui-badge/src	99.61%
packages/wui-timepicker/src	99.56%
packages/wui-rate/src	99.4%
packages/wui-alert/src	99.24%
packages/wui-card/src	99.04%
packages/wui-list/src	99.04%
packages/wui-overlay/src	98.91%
packages/wui-empty/src	98.8%
packages/wui-input-group/src	98.66%
packages/wui-divider/src	98.46%
packages/wui-space/src	98.34%
packages/wui-popover/src	98.11%
packages/wui-pagination/src	97.34%
packages/wui-timeline/src	96.79%
packages/wui-notification/src	96.72%



TinperNext的日常化单测异常友空间提醒


```
async function send() {
    try {
        const content = `【单元测试异常提醒】：
--------------------------------
【分　支】${CI_BUILD_REF_NAME}
【Job编号】${CI_JOB_ID}
【访问地址】${href}
--------------------------------
`
        const res = await axios({
            url: 'https://yonbip.diwork.com/yonbip-ec-link/intelligent-robot/system/send?access[redacted]',
            method: 'post',
            data: {
                "timestamp": new Date().getTime(),
                "content": Base64.toBase64(content)
            }
        })
    } catch (e) {
        console.log(e)
    }

}
await send()
//
}
```

可执行的node脚本，
发友空间群消息提醒

前端单测测试各阶段落地经验总结

初期

- 投入专职开发人员学习
- 脚手架集成测试框架
- 关键组件带头实现单测用例
- Review总结、抽取公共单测用例、公共单测工具方法

中期

- 专职学习人员给内部培训学习，执行全成员参与编写
- 讲解公共单测工具方法适用场景
- 组件API属性整理形成单测开发计划
- 翻历史jira缺陷补全单测用例场景形成开发计划
- 计划执行2周开发+2周单测补充

后期

- 持续测试异常预警提醒
- 持续执行4周开发+1周单测计划，查漏补缺场景
- 持续各种测试报告输出
- 持续监管报告数据是否下降

提问&答疑

用友 yonyou

让白盒测试成为质量把关人，让自己的代码改动，真正做到让自己放心

参考资料

《jest》https://www.jestjs.cn/docs/getting-started
《cypress》https://docs.cypress.io/
《cypress-plugin》https://docs.cypress.io/plugins
《@testing-library/react》https://testing-library.com/docs/react-testing-library/intro

《TinperNext》https://git.yonyou.com/yonyou-ide/tinper-next-packages/next-ui （release分支）

feature ucf-script@3.0 将全面支持前端应用级单测