



与「十」俱进,码出未来

2018源创会年终盛典

企业级项目

Web 自动化测试工程化实践

2018.12.16 何林江，腾讯-IVWEB

个人介绍

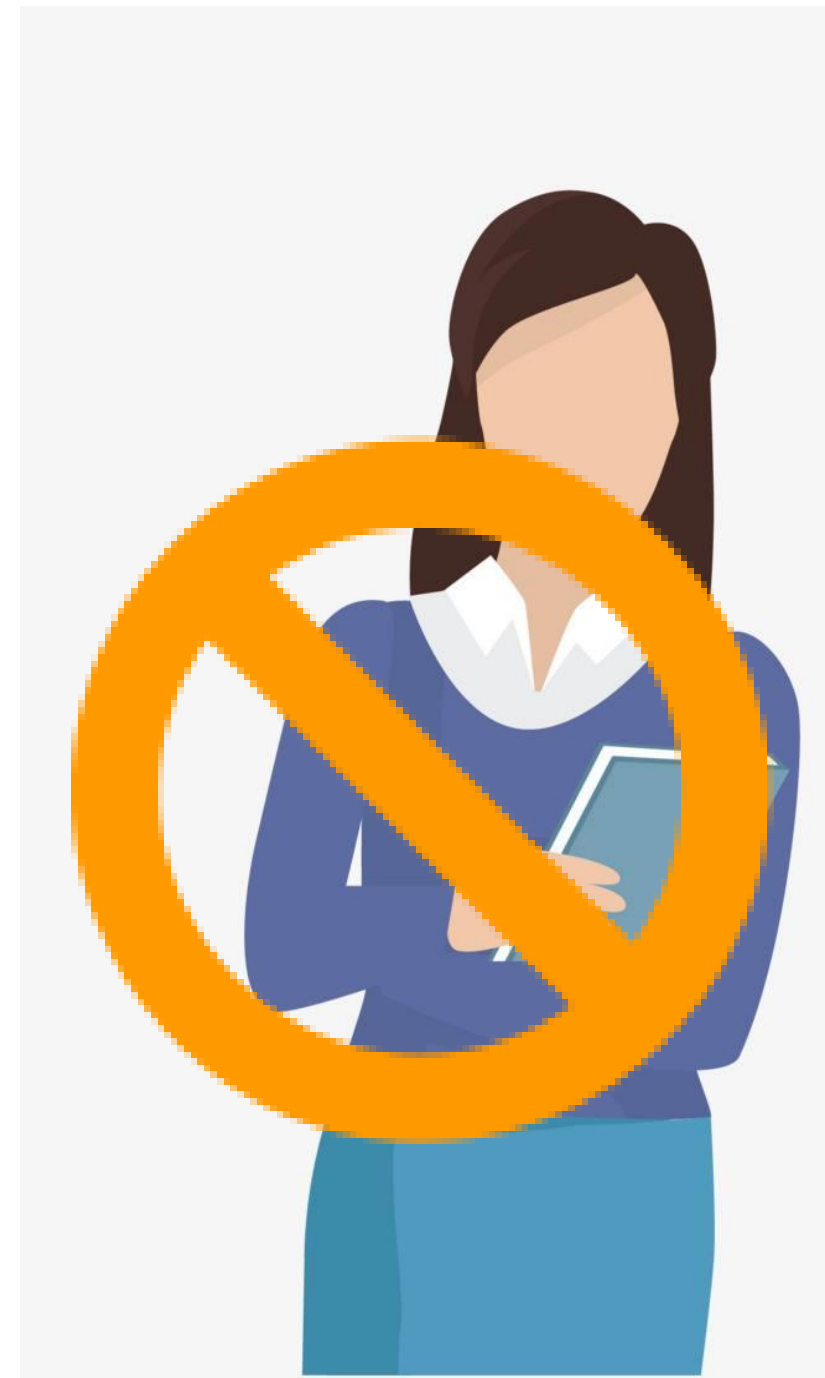
- 何林江
- 腾讯 IVWEB 团队，NOW直播
- 兴趣：
 - 前端工程化
 - 研发效率
 - **自动化测试**

目录

1. 正确认识 web 自动化测试
2. 单元测试（Unit Testing）
3. 端对端测试（End-to-End Testing）
4. 总结

1. 正确认识 web 自动化测试

1.1 什么是自动化测试？



此“测试”非彼“测试”

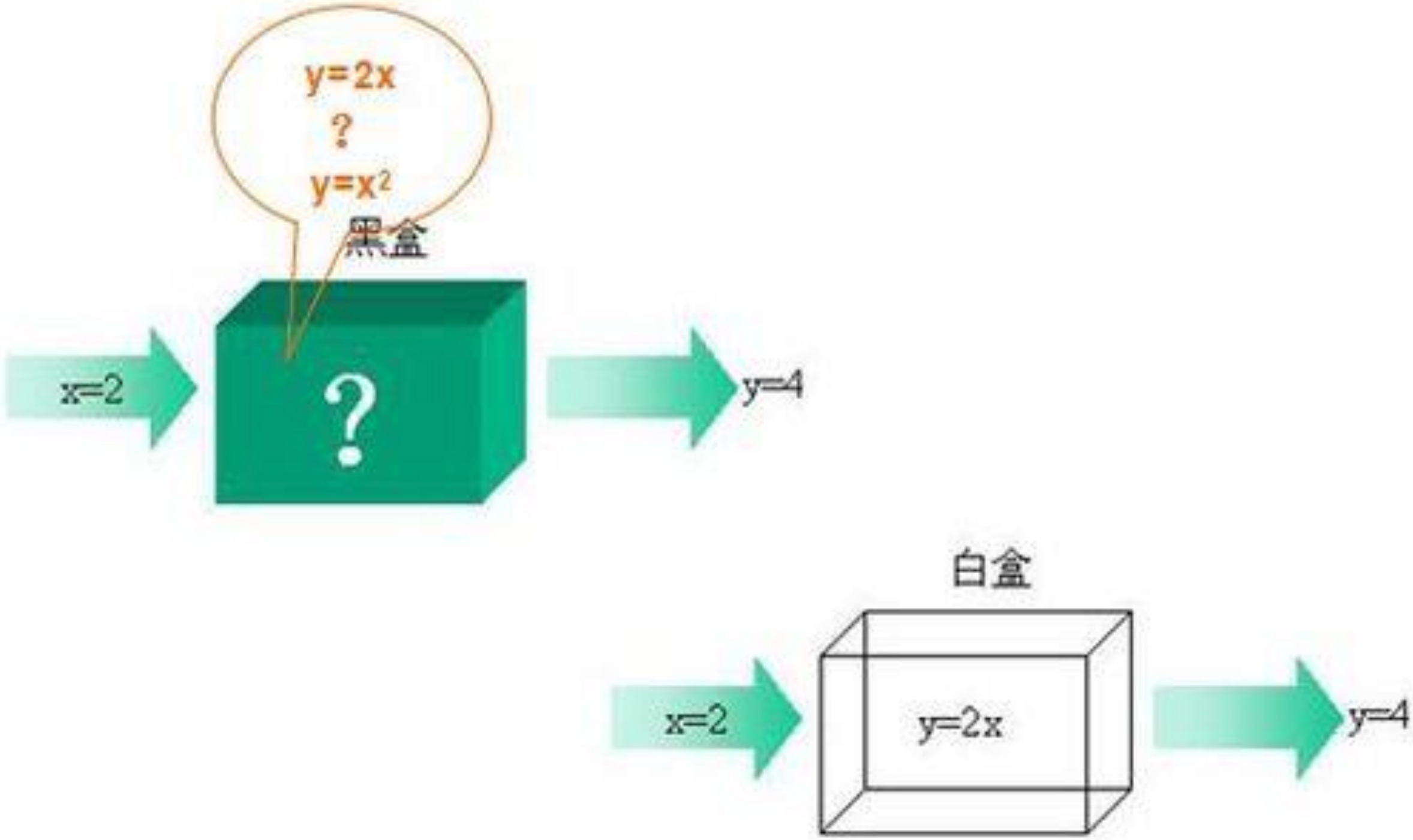
人工参与“测试”

```
function add(a, b) {  
  return a + b;  
}  
  
console.log(add(1, 2));      // 输出: 3  
console.log(add(100, 150)); // 输出: 250
```

无人工参与“测试”
(自动化测试)

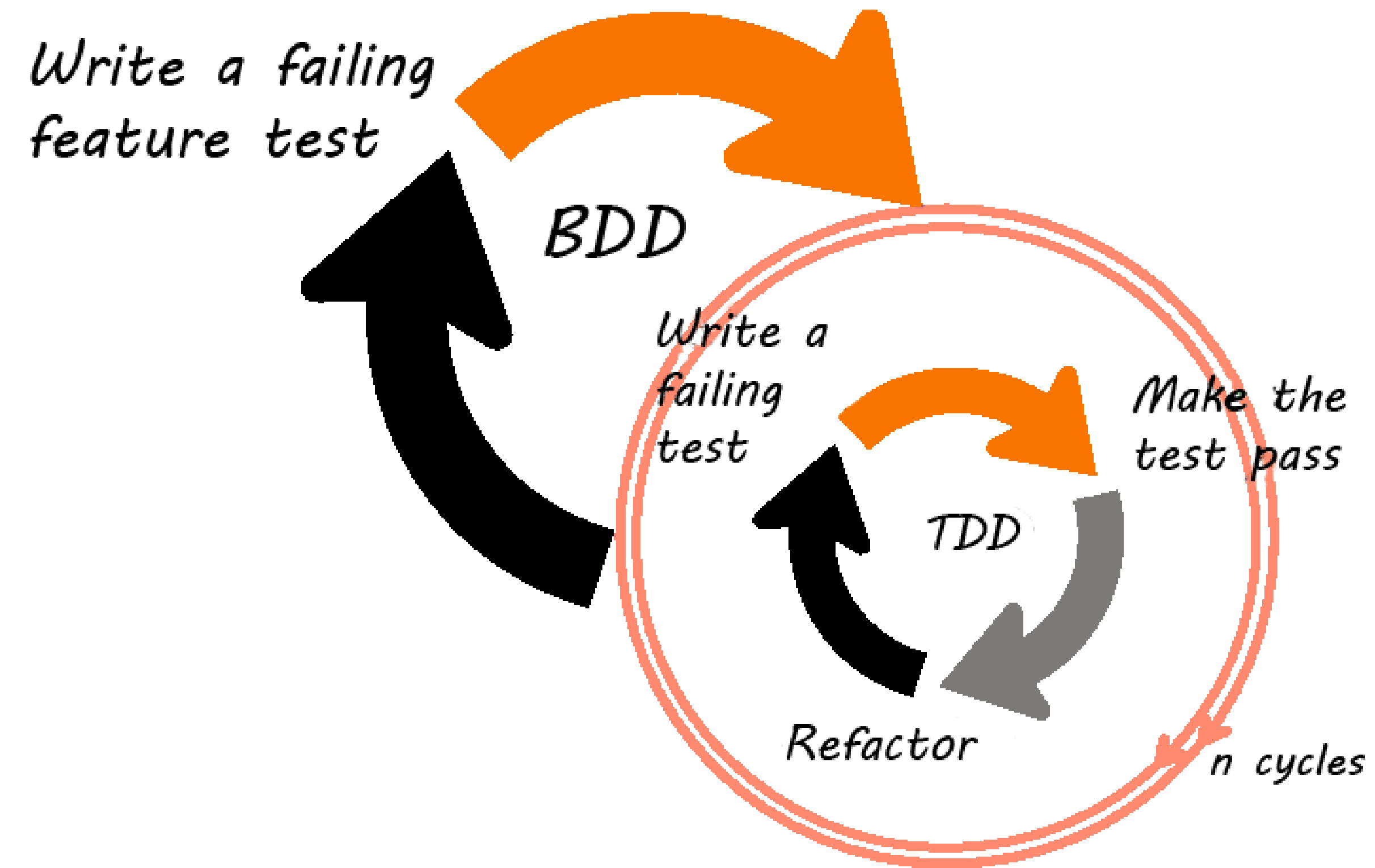
```
const expect = require('chai').expect;  
  
describe('use expect: src/add.js', () => {  
  it('add(1,2) === 3', () => {  
    expect(add(1, 2)).to.equal(3);  
  });  
});
```

1.2 测试方法：黑盒、白盒、灰盒



| 测试方法 | 使用场景 | 使用对象 |
|------|-----------|-----------|
| 黑盒测试 | 系统测试、功能测试 | 测试人员 |
| 白盒测试 | 单元测试 | 开发人员 |
| 灰盒测试 | 集成测试 | 测试人员、开发人员 |

1.3 开发模式：TDD 和 BDD



- TDD(Test Drive Development) ， 测试驱动开发
- BDD (Behavior Drive Development)， 行为驱动开发

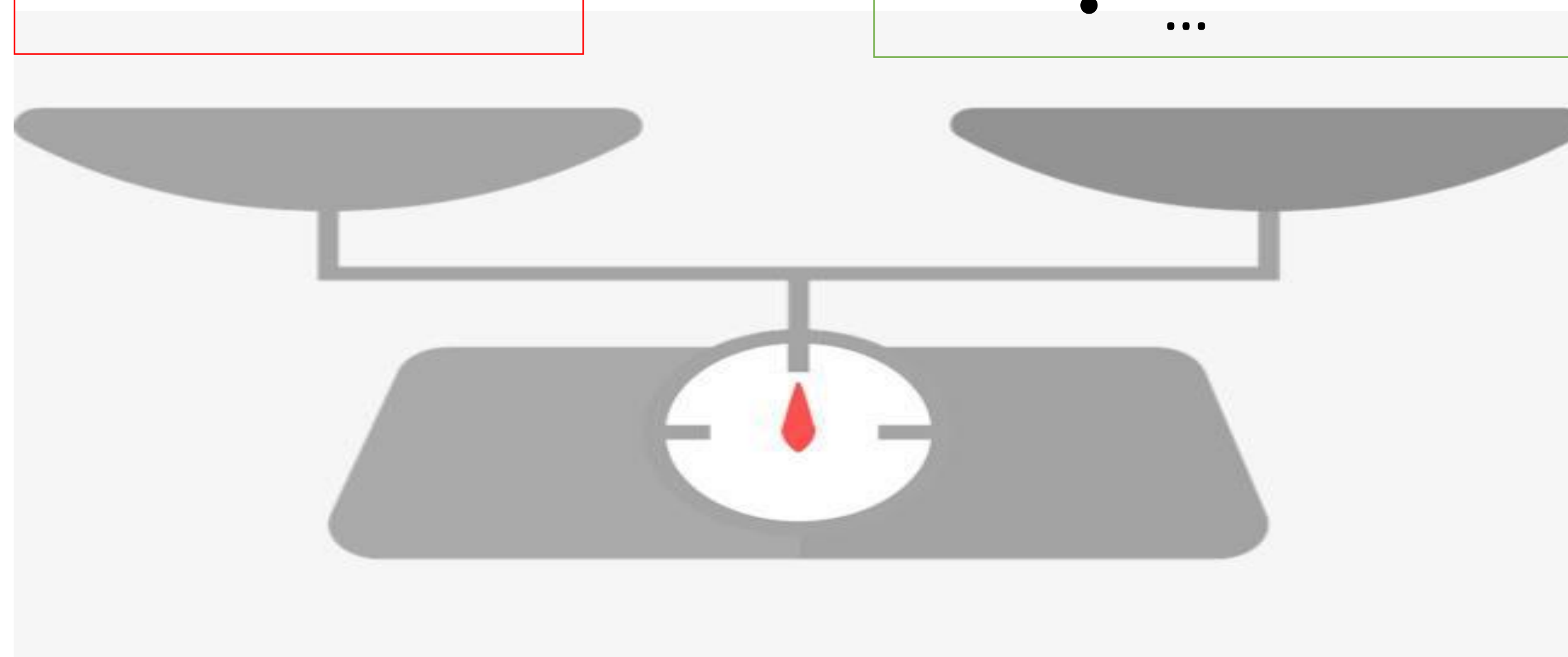
1.4 给个理由，为什么开发还要写测试用例？

代价：

- 人力成本
- 学习成本
- ...

自动化测试的好处：

- 增加程序信任度
- 让回归测试更方便
- 白盒测试
- ...



2. 单元测试（Unit Testing）

2.1 有哪些测试框架？



Mocha



AVA



Jasmine



Jest



Tape

单纯的测试框架，需配合断言库

- chai
- should.js
- expect
- better-assert

集成框架，开箱即用

极简API

2.2 写单元测试难吗？



- Mocha + Chai
- describe, it, expect
- mocha add.test.js

add.test.js

```
const expect = require('chai').expect;

const add = require('../src/add');

describe('use expect: src/add.js', () => {
  it('add(1,2) === 3', () => {
    expect(add(1, 2)).to.equal(3);
  });
});
```


2.3 接入单元测试难吗？

1. 安装 mocha + chai

```
npm install mocha chai --save-dev
```

2. 新建 test 目录，并增加 xxx.test.js 测试文件

```
const expect = require('chai').expect;

const add = require('../src/add');

describe('use expect: src/add.js', () => {
  it('add(1,2) === 3', () => {
    expect(add(1, 2)).to.equal(3);
  });
});
```

3. 在 package.json 中的 scripts 增加 test 命令

```
"scripts": {
  "test": "node ./node_modules/mocha/bin/_mocha"
}
```

4. 执行测试命令

```
npm test
```

2.4 还要了解什么？



Karma



jsdom

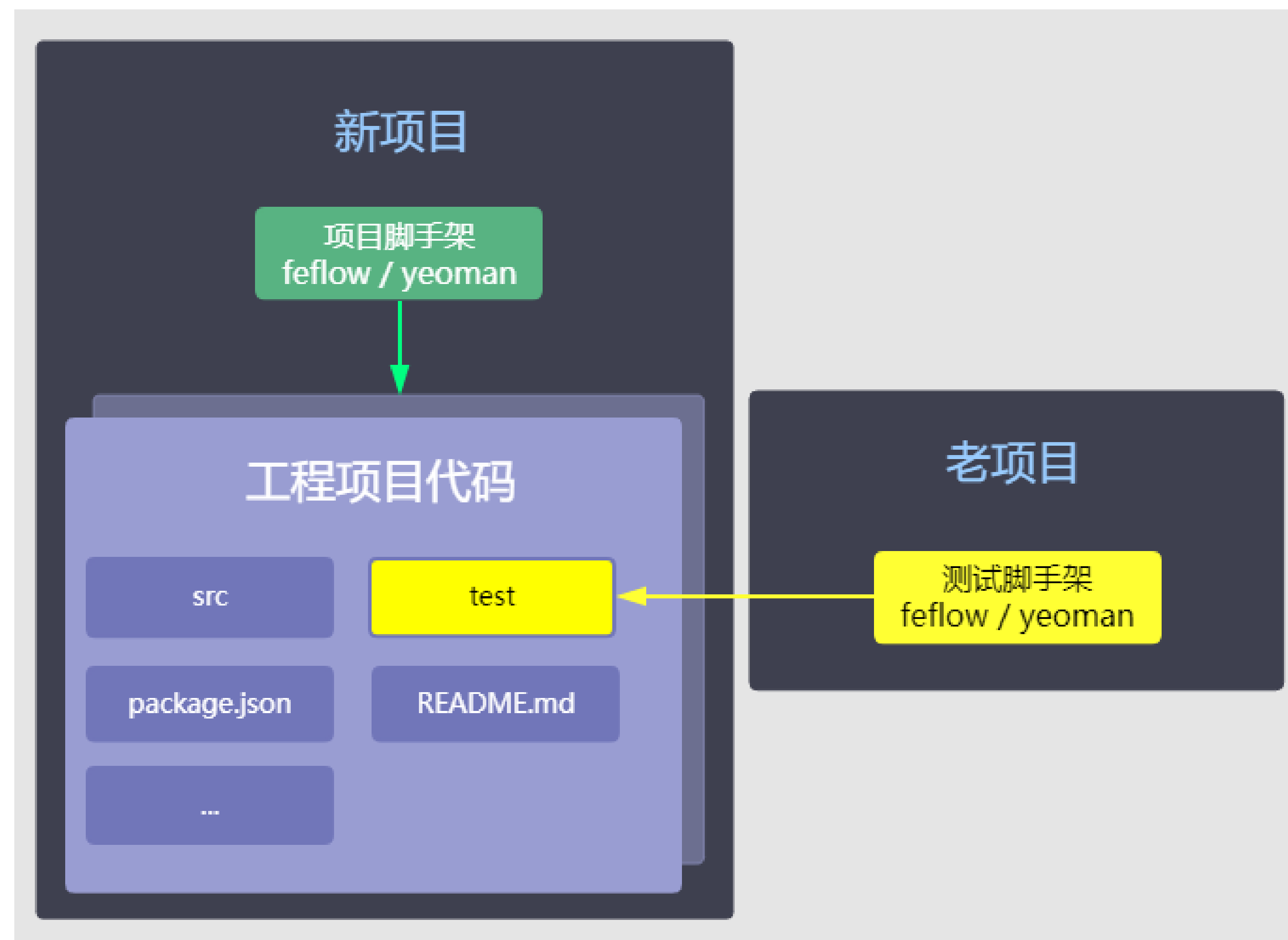


Sinon.JS

Enzyme

...

2.5.1 脚手架快速搭建环境



2.5.2 断言辅助生成和校验工具

1. 请选择测试目标的类型

字符串: "matman" 数字: 10086

对象: { "name": "matman", "age": 1, "isPublished": true }

数组: ["a", "B", 10086]

常用的断言写法

// 为字符串类型
expect(value).to.be.a('string')

a

试一试

拷贝代码

// 相等
expect(value).to.equal('matman')

equal

试一试

拷贝代码

// 不相等
expect(value).to.not.equal('matman')

equal

not

试一试

拷贝代码

// 字符串长度有多长
expect(value).to.have.lengthOf(6)

lengthOf

试一试

拷贝代码

1.生成

```
// 判断是否等于某个字符串
expect(value).to.equal('matman');

// 判断是否等于 true
expect(value).to.be.true;

// 判断是否等于 null
expect(value).to.be.null;

// 判断是否等于 undefined
expect(value).to.be.undefined;

// 判断是否为 NaN
expect(NaN).to.be.NaN;

// 判断是否等于某个对象
expect(value).to.eql({
  "name": "matman",
  "age": 1,
  "isPublished": true
});
```

2.试一试

2. 试一试

expect(4 + 5).to.equal(9)

开始测试

使用说明

- 根据要测试的值的类型进行选择，会出现常见的用法代码示例；
- 选择代码示例中的“试一试”按钮，可将代码样例复制到体验区；
- 点击体验区中的“开始测试”，则会对样例代码进行校验，并输出结果；

相关链接

- chai API : <http://www.chaijs.com/api/>
- 更多文档说明

2.5.3 扩展断言插件等

```
describe('check qq', () => {  
  let qq = 10001;  
  
  it('qq should be type of number', () => {  
    expect(qq).toBe.a('number');  
  });  
  
  it('qq should be integer', () => {  
    expect(qq % 1).toEqual(0);  
  });  
  
  it('qq should above 10001', () => {  
    expect(qq).toBe.at.least(10001);  
  });  
});
```

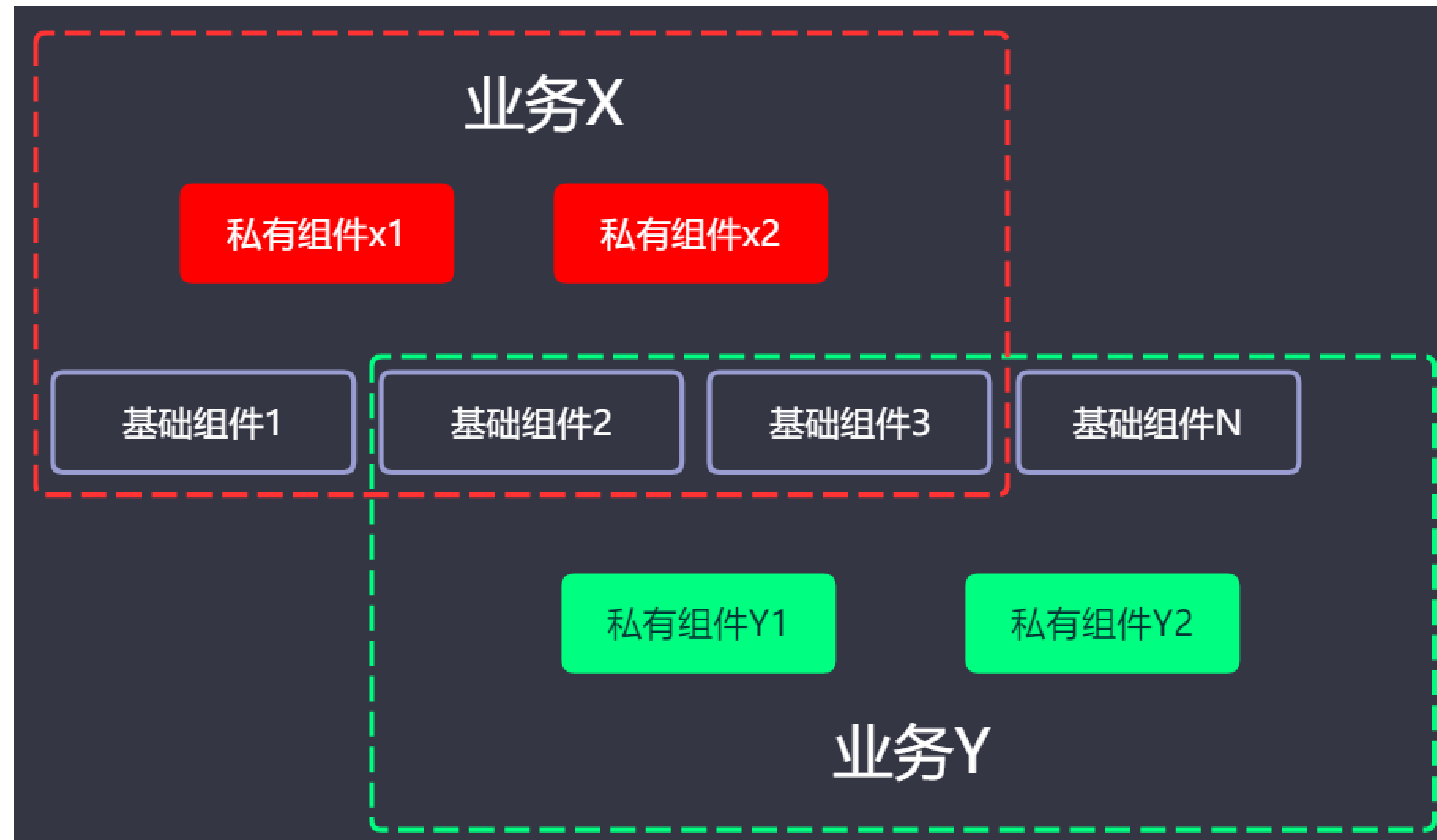
常规写法，断言数量 ≥ 3



```
describe('check qq use plugin', () => {  
  let qq = 10001;  
  
  it('qq should valid', () => {  
    expect(qq).toBe.a.validQQ;  
  });  
});
```

扩展之后，断言数量=1

2.5.4 公共组件建设



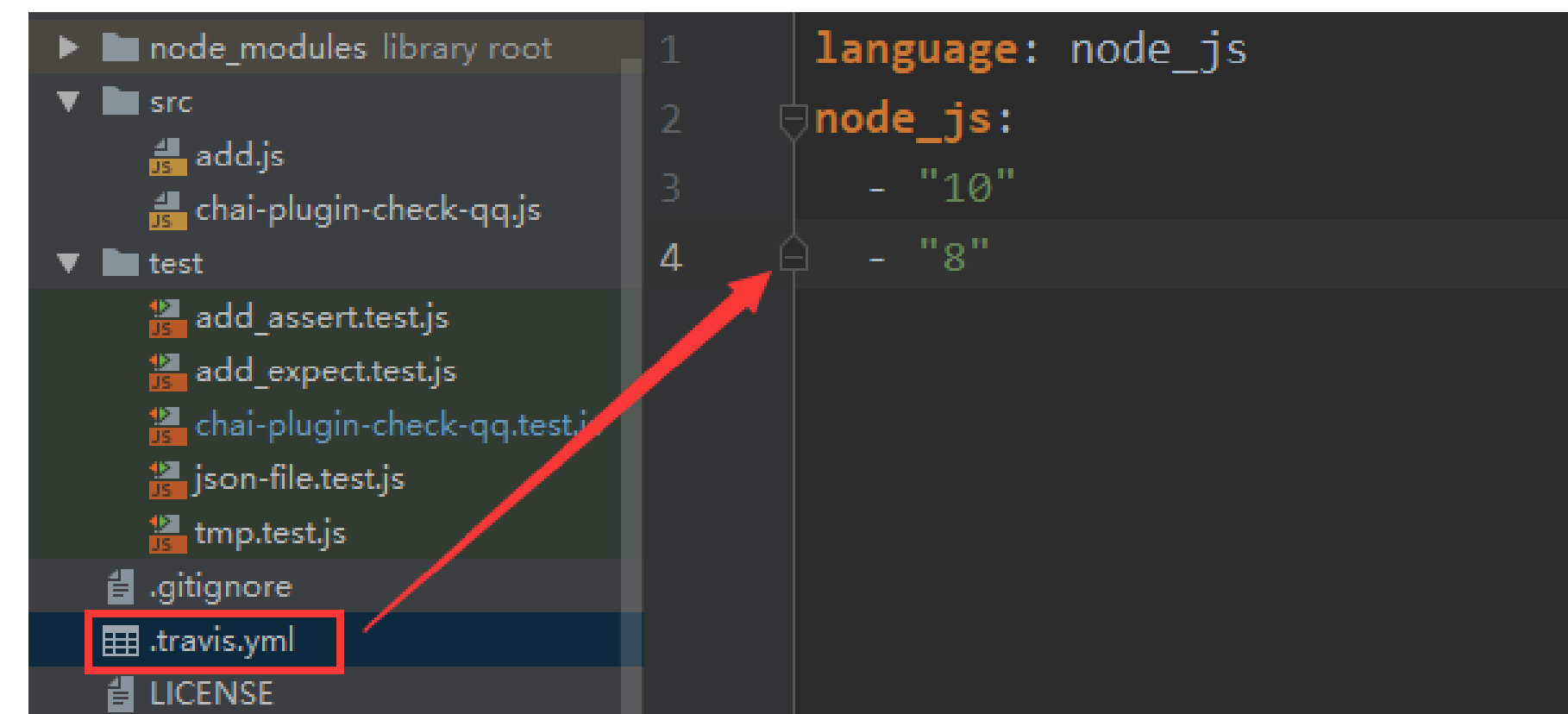
剥离前（5个） -> 剥离后（2个）

2.5.5 持续集成 CI

1. <https://travis-ci.org/> 使用 github 账号登录
2. 在 <https://travis-ci.org/account/repositories> 为项目开启



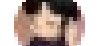



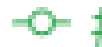













 **guide-to-unit-testing**   Settings

3. 项目根目录下新增 .travis.yml



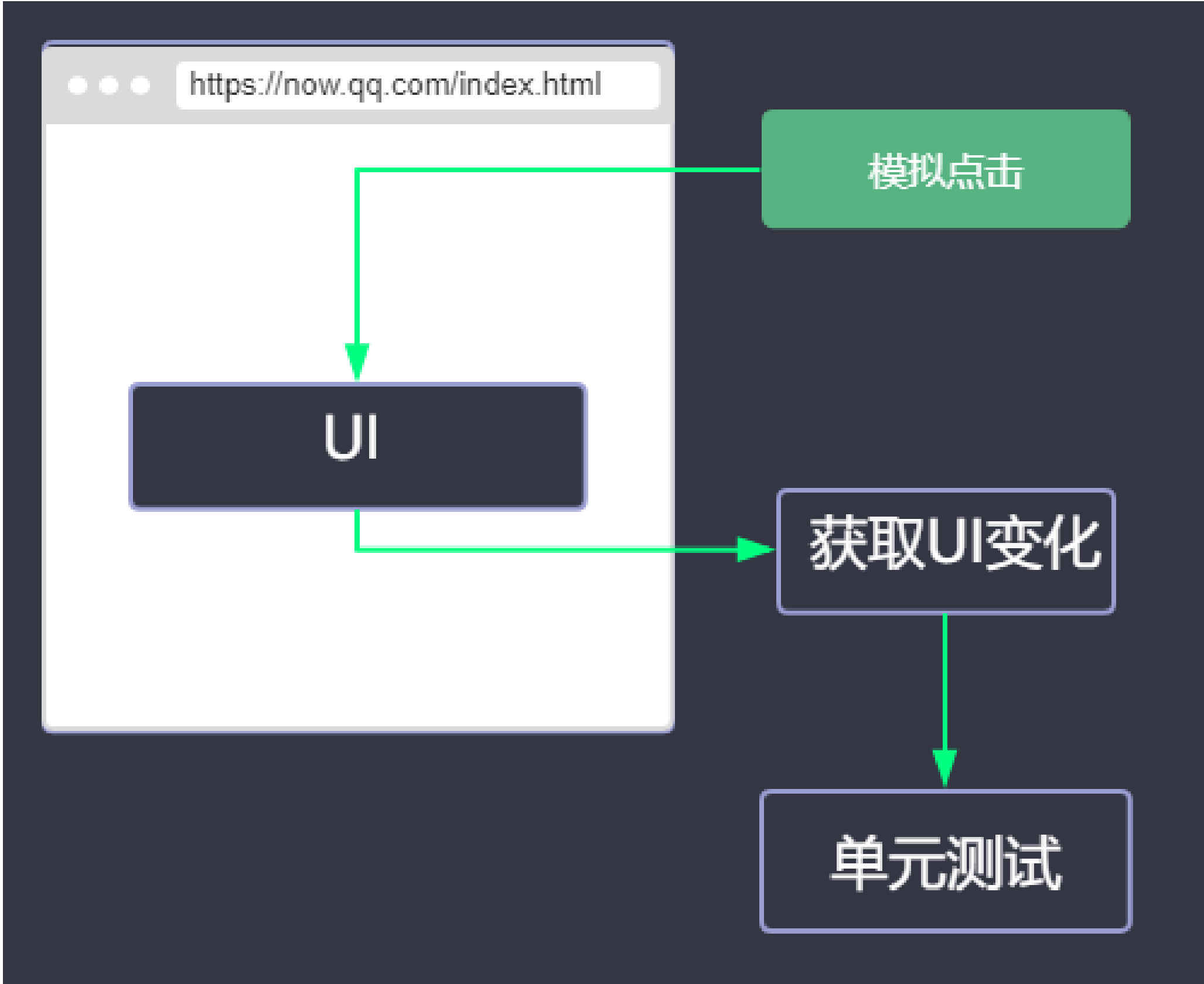
```
language: node_js
node_js:
  - "10"
  - "8"
```

4. 每次 push 之后就会自动运行测试用例

| Current | Branches | Build History | Pull Requests | More options |
|---------|----------|---|---|---------------------|
| | | | | |
| | |  feature_only feat: 临时数据 |  #4 started | 🕒 7 sec |
| | |  HeLinJiang |  b423311  | 📅 - |
| | |  feature_only feat: 增加 travis ci 配置文件 |  #3 passed | 🕒 38 sec |
| | |  HeLinJiang |  0e745a5  | 📅 about 8 hours ago |
| | |  master feat: 增加 travis ci 配置文件 |  #2 passed | 🕒 50 sec |
| | |  HeLinJiang |  0e745a5  | 📅 about 9 hours ago |
| | |  master feat: 增加 travis ci 配置文件 |  #1 errored | 🕒 1 min 10 sec |
| | |  HeLinJiang |  bf95c97  | 📅 about 9 hours ago |

3. 端对端测试（End-to-End Testing）

3.1 什么是端对端测试



端对端测试：End-to-End Testing，可简称 E2E Testing

| 测试方式 | 场景 | 特点 |
|-------|------|--------|
| 单元测试 | 组件 | 功能完善性 |
| 端对端测试 | UI交互 | 从用户的视角 |

3.2 端对端测试工具

- Nightmare -> Electron -> Chromium引擎 -> WebKit 内核
- CasperJS -> PhantomJS -> WebKit 内核
- Selenium\Protractor\Nightwatch.js\TestCafe\CodeceptJS
- ...

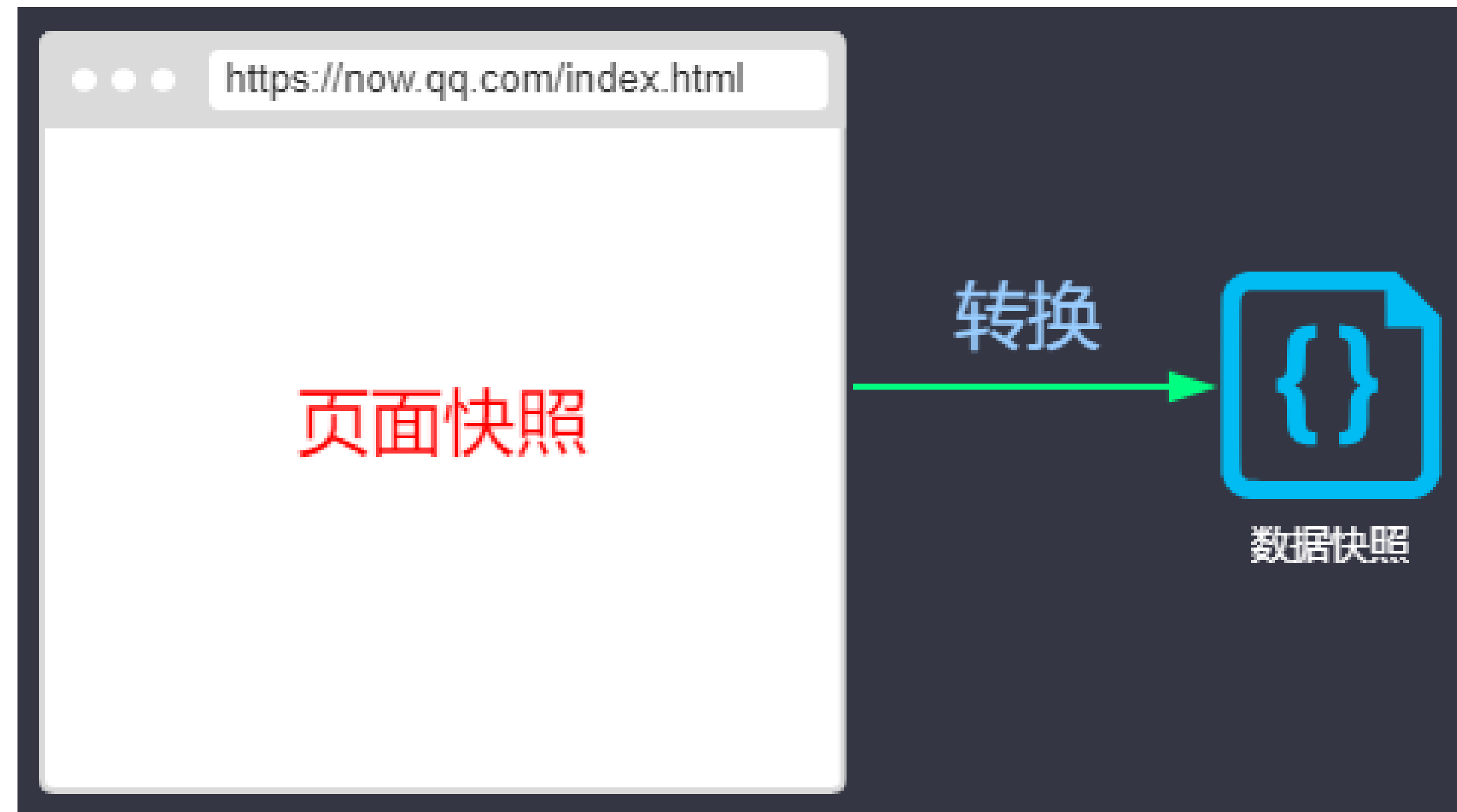
3.3 主流的端对端测试方案

```
nightmare
  .goto('http://yahoo.com')
  .type('form[action="/search"] [name=p]', 'github nightmare')
  .click('form[action="/search"] [type=submit]')
  .wait('#main')
  .evaluate(function () {
    return document.querySelector('#main .searchCenterMiddle li a').href
  })
```

- 使用无头浏览器加载页面
- 模拟用户点击操作
- 操作完成后收集特定的UI信息
- 校验该状态值是否符合预期

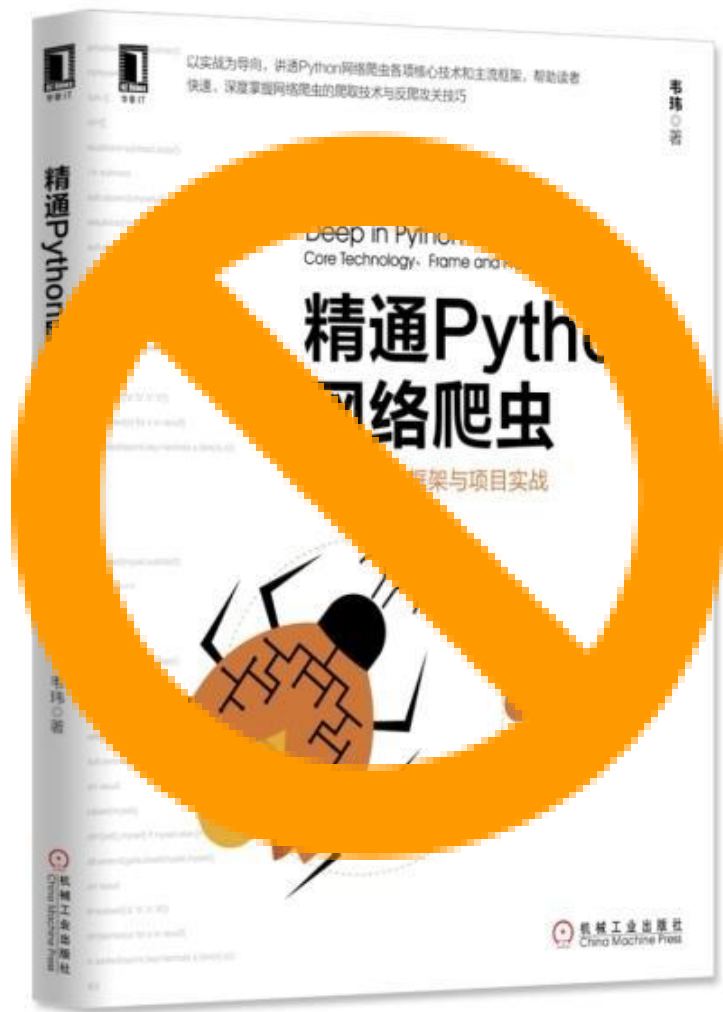
```
module.exports = {
  tags: ['git'],
  'Demo test GitHub': function (client) {
    client
      .url('http://github.com/nightwatchjs/nightwatch')
      .waitForElementVisible('body', 1000)
      .waitForElementVisible('.container h1 strong a')
      .assert.containsText('.container h1 strong a', 'nightwatch', 'Checking project title is set to nightwatch');
  },
  after(client) {
    client.end();
  }
};
```

3.4.1 我们怎么做端对端测试



- “**页面快照**”：完整的页面状态
- 用户的交互动作会产生新的“**页面快照**”
- 将“**页面快照**”转换为“**数据快照**”

3.4.2 如何将页面快照转换为数据快照？



加载jQuery写爬虫

| | | |
|---------|------------|-----------|
| 兑换商品 | 2018.10.09 | -2.19 |
| 兑换商品 | 2018.10.09 | -2.29 |
| 兑换商品 | 2018.10.09 | -2.39 |
| 兑换商品 | 2018.10.09 | -2.49 |
| 兑换商品 | 2018.10.09 | -2.59 |
| 摇一摇新手红包 | 2018.10.08 | 已过期 +3.09 |
| 摇一摇任务红包 | 2018.10.08 | 已过期 +3.19 |
| 摇一摇任务红包 | 2018.10.08 | 已过期 +3.29 |
| 摇一摇任务红包 | 2018.10.08 | 已过期 +3.39 |
| 摇一摇任务红包 | 2018.10.08 | 已过期 +3.49 |
| 摇一摇任务红包 | 2018.10.08 | 已过期 +3.59 |



28 messages

22 user me...

No errors

No warnings

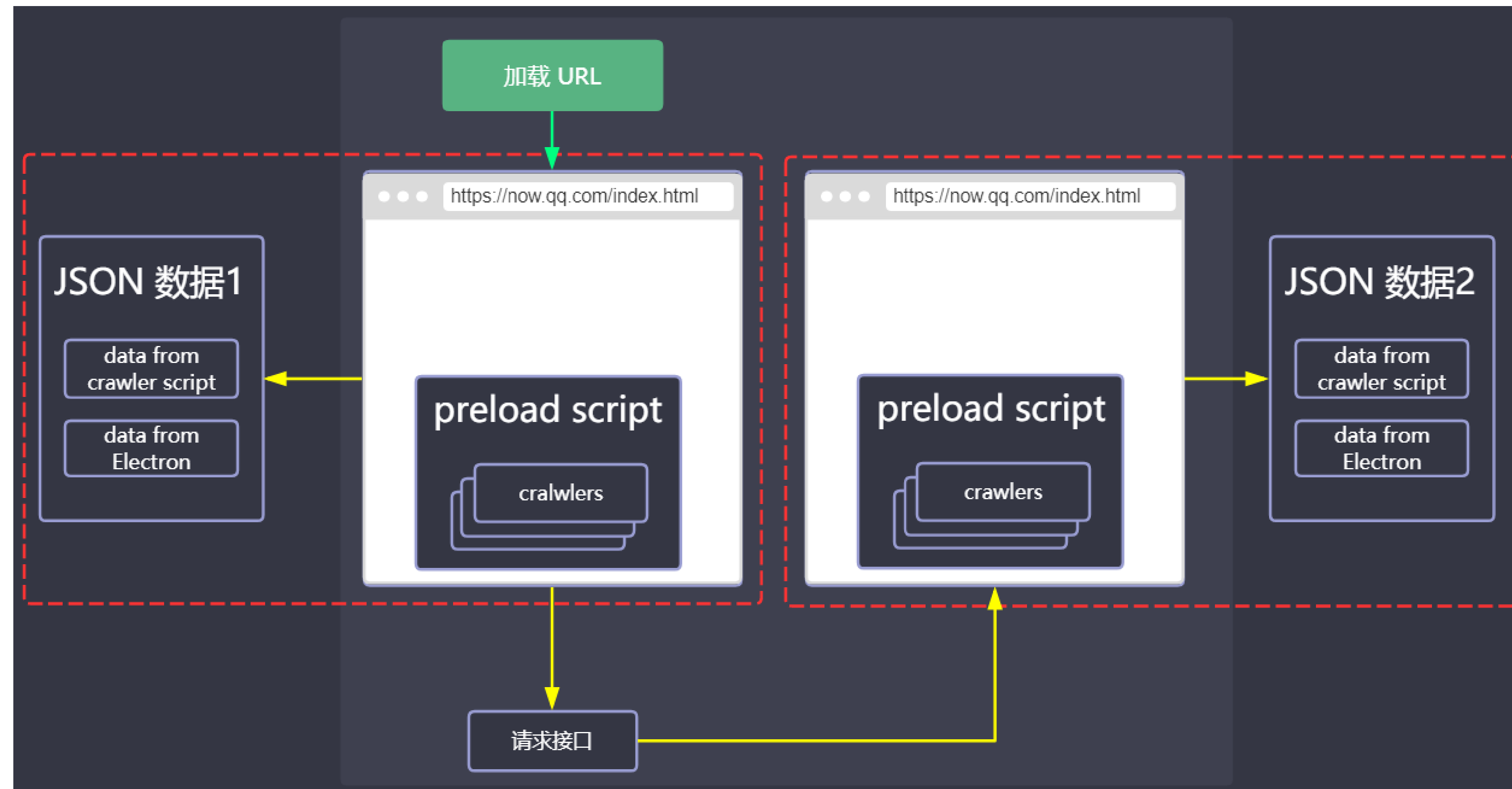
22 info

6 verbose

```
bin%ZFkTV_match%Zfsnake_redpacket%Zfrredpack%ZfgetRedpackFlow&l_3=1&l_4=0&l_5=1/2&l_6=1&l_1=now.qq.com&l_2=%ZFcg
bin%ZFkTV_match%ZFshake_redpacket%ZFredpack%ZFgetRedpackFlow&2_3=1&2_4=0&2_5=251&2_6=1&_0.07637737325152116",2
_binid=3307&now_n_http=1&no_eruda=1&no_logger=1&_ms=[{"%22_ms_name%22:%22getRedpackFlow%22,%22_ms_target%22:%22su
length":["0"],"content-type":["text/html"],"date":["Tue, 04 Dec 2018 09:31:45 GMT"],"keep-alive":["timeout=50"]
< {data: Array(2), globalInfo: {...}}
  data: Array(2)
    0: {
      listInfo: {
        emptyLogo: ""
        emptyWording: ""
        i_0_1_6: {isExist: true, isExpire: false, money: "+0.09", name: true, time: true}
        i_0_1_10001: {isExist: true, isExpire: false, money: "+0.19", name: true, time: true}
        i_0_1_10002: {isExist: true, isExpire: false, money: "+0.29", name: true, time: true}
        i_0_1_10003: {isExist: true, isExpire: false, money: "+0.39", name: true, time: true}
        i_0_1_10004: {isExist: true, isExpire: false, money: "+0.49", name: true, time: true}
        i_0_1_10005: {isExist: true, isExpire: false, money: "+0.59", name: true, time: true}
        i_0_2_6: {isExist: true, isExpire: false, money: "-1.09", name: true, time: true}
        i_0_2_10001: {isExist: true, isExpire: false, money: "-1.19", name: true, time: true}
        i_0_2_10002: {isExist: true, isExpire: false, money: "-1.29", name: true, time: true}
        i_0_2_10003: {isExist: true, isExpire: false, money: "-1.39", name: true, time: true}
        i_0_2_10004: {isExist: true, isExpire: false, money: "-1.49", name: true, time: true}
        i_0_2_10005: {isExist: true, isExpire: false, money: "-1.59", name: true, time: true}
        i_0_3_6: {isExist: true, isExpire: false, money: "-2.09", name: true, time: true}
        i_0_3_10001: {isExist: true, isExpire: false, money: "-2.19", name: true, time: true}
        i_0_3_10002: {isExist: true, isExpire: false, money: "-2.29", name: true, time: true}
        i_0_3_10003: {isExist: true, isExpire: false, money: "-2.39", name: true, time: true}
        i_0_3_10004: {isExist: true, isExpire: false, money: "-2.49", name: true, time: true}
        i_0_3_10005: {isExist: true, isExpire: false, money: "-2.59", name: true, time: true}
        i_0_4_6: {isExist: true, isExpire: true, money: "+3.09", name: true, time: true}
        i_0_4_10001: {isExist: true, isExpire: true, money: "+3.19", name: true, time: true}
        i_0_4_10002: {isExist: true, isExpire: true, money: "+3.29", name: true, time: true}
        i_0_4_10003: {isExist: true, isExpire: true, money: "+3.39", name: true, time: true}
        i_0_4_10004: {isExist: true, isExpire: true, money: "+3.49", name: true, time: true}
        i_0_4_10005: {isExist: true, isExpire: true, money: "+3.59", name: true, time: true}
        isEmpty: false
        isExist: true
        total: 24
        __proto__: Object
      }
      projectInfo: {appNow: {...}, basic: {...}, config: {...}}
      switchTabInfo: {isExist: true, isQuotaActive: false, isRedpackedActive: true, quotaText: "提额明细", redpa
      __proto__: Object
    }
    1: {listInfo: {...}, projectInfo: {...}, switchTabInfo: {...}}
      length: 2
      __proto__: Array(0)
```



3.4.3 如何运行爬虫脚本？



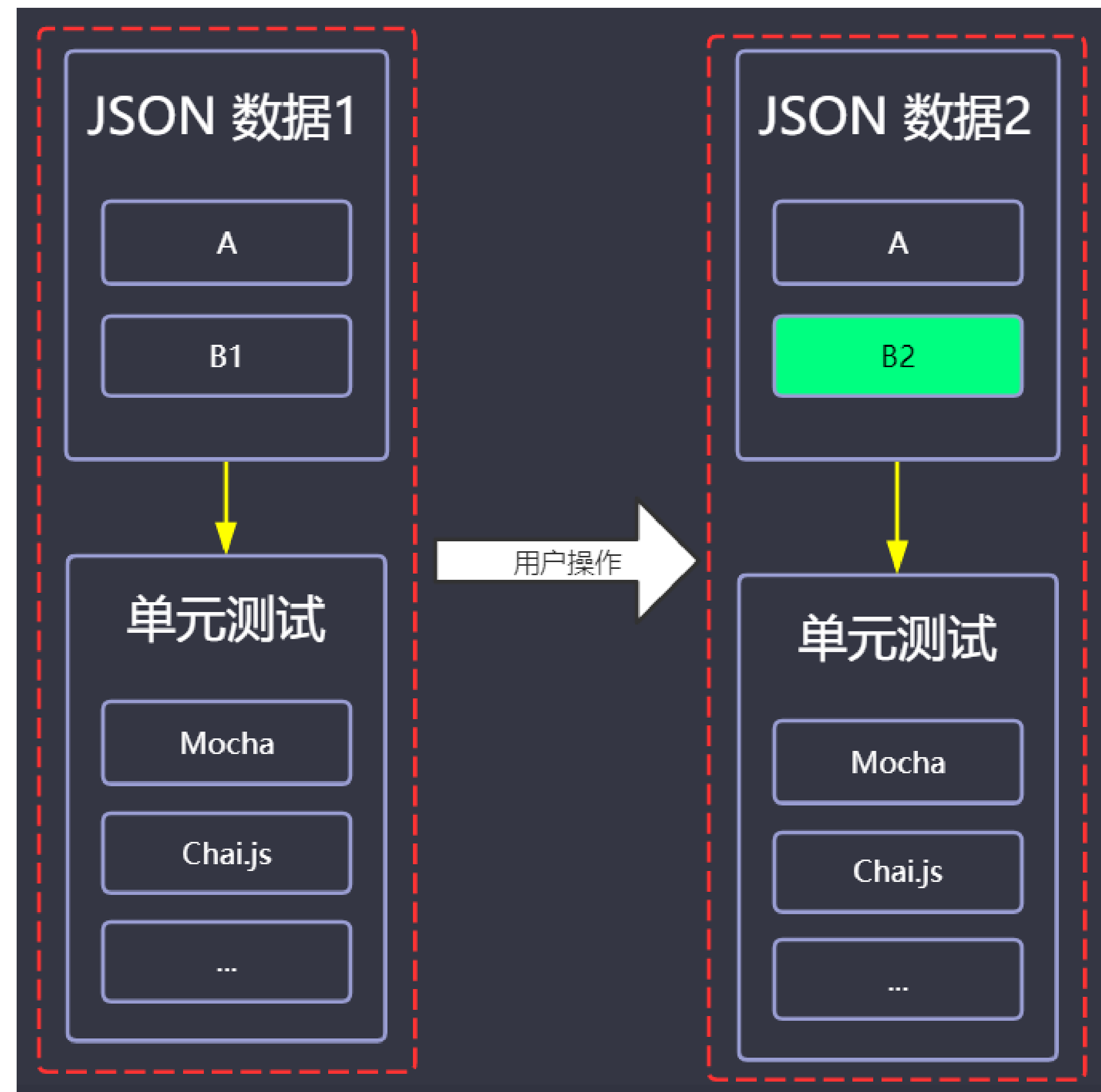
- 基于 Nightmare，支持 preload script
- 每一个页面快照产生一个数据快照

DOM 中的信息

JS 中的信息

Network 加载等额外信息

3.4.4 如何测试？

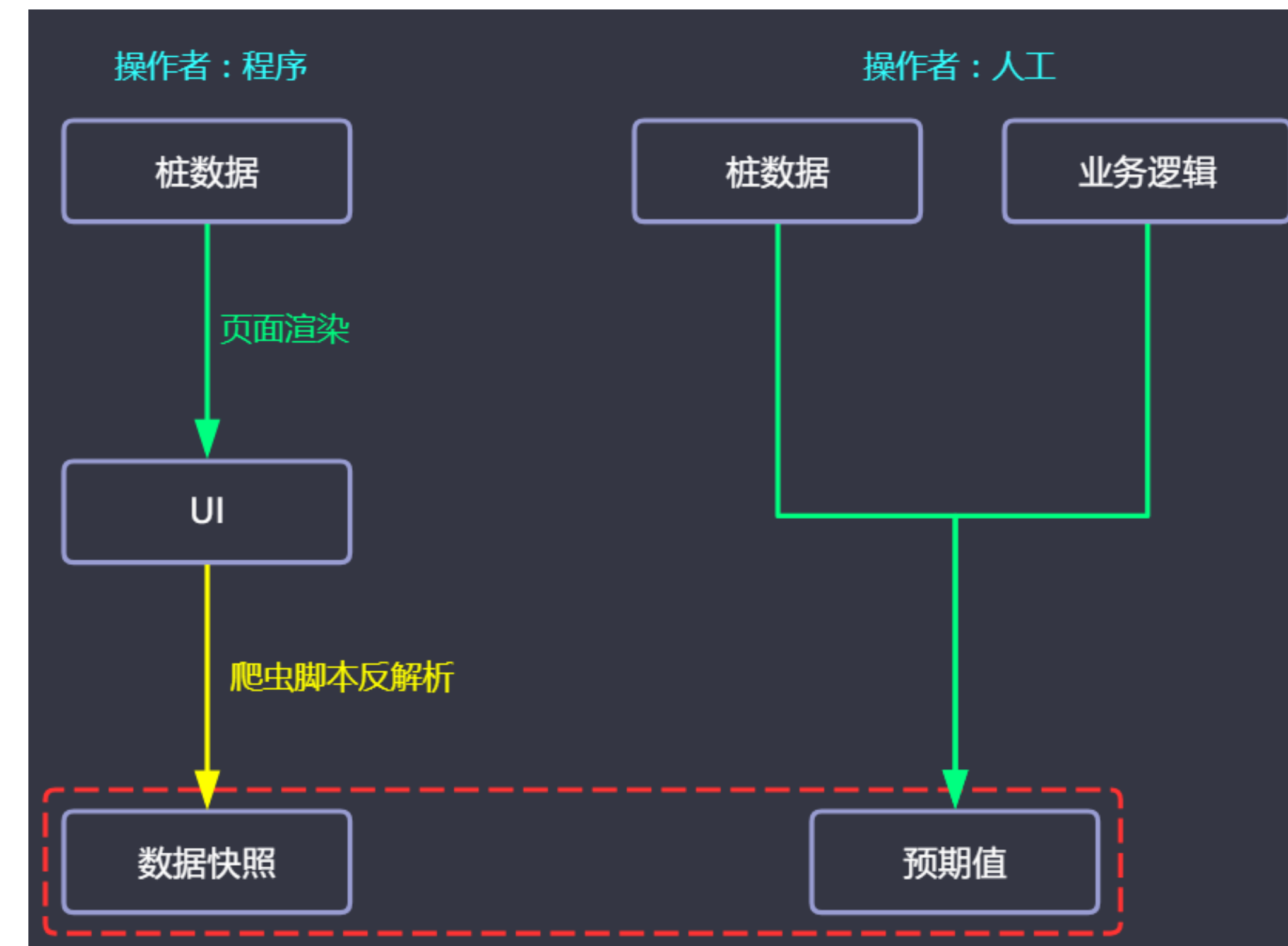
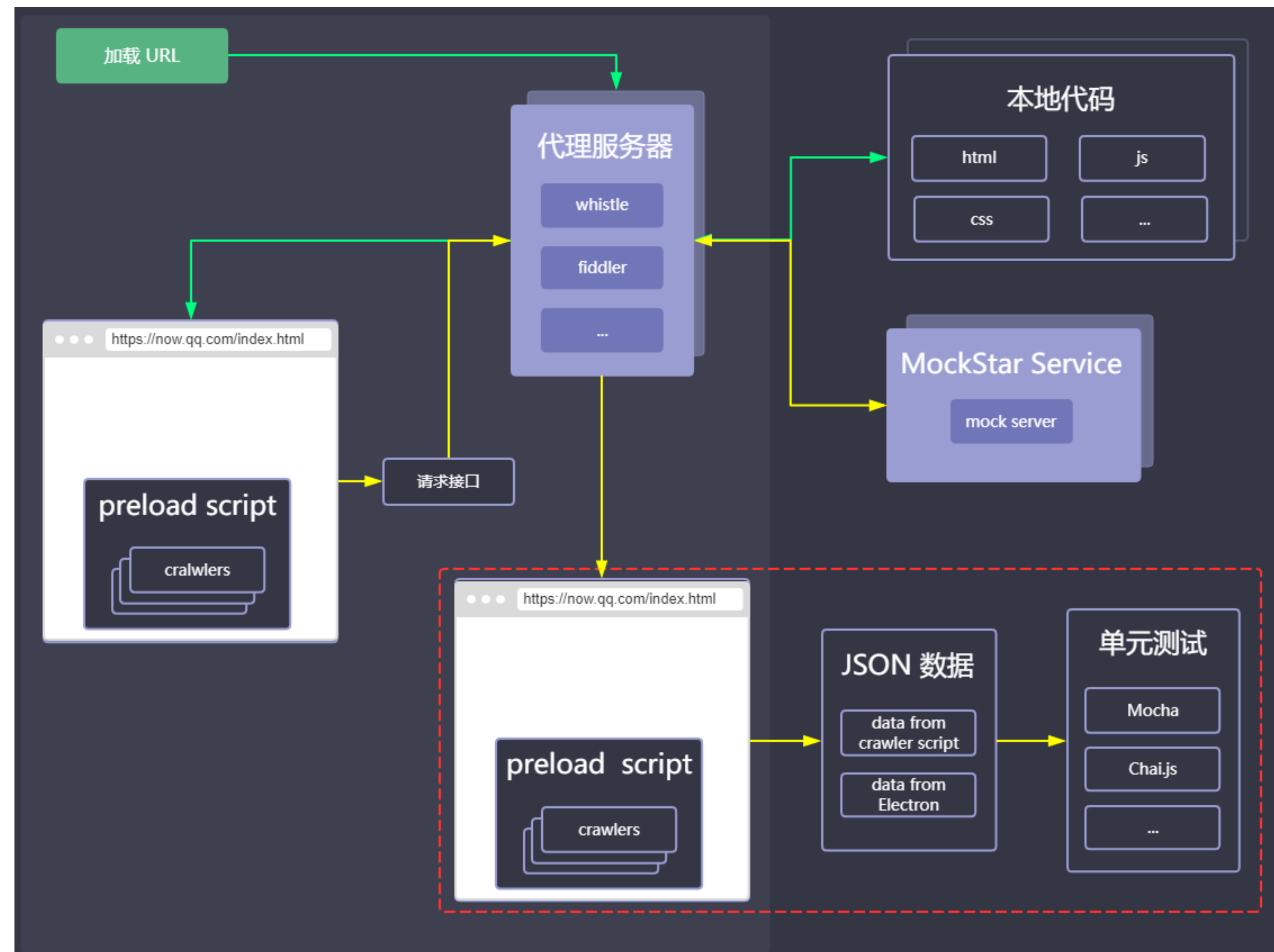


- 校验数据快照本身是否符合预期
- 校验数据快照之间的变化是否符合预期

3.4.5 黑盒测试 VS 白盒测试



3.4.6 端对端白盒测试方案架构图



- <https://github.com/mockstarjs/mockstar>
- <https://github.com/matmanjs/matman>

4. 总结

总结

单元测试

- 业界成熟方案多，接入成本不高
- 可通过工具建设等方式减低难度
- 更适合公共组件

端对端测试

- 接入成本较高
- 白盒测试方式容易入手
- 更适合UI测试

END

感谢大家，欢迎一起交流！

- <https://www.ivweb.io/>
- <https://github.com/helinjiang>
- <https://github.com/feeflow/feeflow>
- <https://github.com/mockstarjs/mockstar>
- <https://github.com/matmanjs/matman>

