

Cypress 实践标准

1. 测试原则：

2. 规范：

2.1. 代码规范：

格式：

要为代码进行格式化

避免一行代码过长

避免在方法内保留空行

变量：

变量命名要按照Camel方式命名，要有意义，给变量赋予有意义的值

考虑使用const声明

不要存留未引用的变量声明

不要出现绝对地址

不要使用自动定位，尽量避免使用无意义的选择器

2.2. Case规范

文件命名：

内容：3A模式

3. 常用断言

1. 测试原则：

对于你即将测试的系统，没有人比你更了解它的业务，你可以使用UI自动化来只测试UI交互，也可以使用UI自动化来测试一个从头到尾的故事线。

测试的内容，测试的边界将完全取决于你和你的团队的决定。这里有一些自动化测试的经验性原则，希望能给大家一些提示：

1. 避免依赖他人数据：

尽量不要依赖别人的数据，尤其不要依赖其他人使用UI自动化跑出来的数据，否则一旦创建数据的步骤出错，你的测试结果也极易变得不稳定，所以尽量使用自己可以信赖的数据。

据

2. 避免使用UI构造数据、设置状态：

很多时候测试页面的各种状态，需要先在服务器创造一些条件，之前可能会需要比如 setupDB、或者使用UI直接操作等，这些做法尽管没有什么所谓的错误，但它增加了复杂性以及不稳定性，所以一个原则就是尽量不要用UI构造数据，设置状态，除非这个设置过程是你UI测试需要覆盖的一环。Cypress提供了很多api可以供我们直接发送请求构造数据和状态，尽量使用这些api。

3. 验证自己的操作、对结果负责：

任何时候，任何位置，一旦操作了页面产生结果，一定不要忘记添加验证，这能帮助你更快的判断自己操作有没有成功。

更重要的事情是，自己要对自己产生的结果负责，比如你Case负责创建数据，一定要在自己代码里自我证明是创建成功的，而不是等待其他人的其他Case来调用、使用你创建的数据时才发现你没有创建成功。

2. 规范：

一个好的规范，可以保证测试代码在大多数情况下，遵循相同的原则，这些原则是为了使用者，将来能够更好的阅读、维护代码，也能够一定程度上避免杂乱无章的开发习惯带来的种种未知的隐患。

2.1. 代码规范：

格式：

要为代码进行格式化

要为代码进行格式化，格式化快捷键：Alt + Shift + F，或 (Alt + K + F)

```
1 //Fails
2     cy.get(".Button").click();
3         // Assert
4         cy.get(".nav-title").should("exist");
5     cy.get(".detailslist").should("not.exist");
6
7 //Passes
8     cy.get(".Button").click();
```

```

9      // Assert
10     cy.get(".nav-title").should("exist");
11     cy.get(".detailslist").should("not.exist");

```

避免一行代码过长

避免一行代码超过160个字节，如果超过160字节，可考虑在.get(),.should()等方法调用的地方换行

```

1 //Fails
2 cy.contains('MY FAVOURITE LINKS').next().contains('Manage').as('ManageLinkBtn').should('have.attr','href','/admin/managelinks')
3
4 //Passes
5 cy.contains('MY FAVOURITE LINKS').next().contains('Manage')
6   .as('ManageLinkBtn')
7   .should('have.attr','href','/admin/managelinks')

```

避免在方法内保留空行

避免方法内留有多余空行，如果要分隔，加注释分隔

```

1 //Fails
2     it("Input a wrong site URL", function () {
3
4
5         cy.get("#search-box01").clear().type(siteURL);
6
7         cy.get(".validate-alert").should("have.length", 1);
8     });
9
10 //Passes
11     it("Input a wrong URL", function () {
12         // Act
13         cy.get("#search-box01").clear().type(scURL);
14         // Assert
15         cy.get(".validate-alert").should("have.length", 1);
16     });

```

变量：

变量命名要按照Camel方式命名，要有意义，给变量赋予有意义的值

camel方式命名：第一个单词小写（个别的工具类型除外），其余单词首字母大写

```
1 //Fails
2 const newadminname = "Site Collection Admin"
3 const OriginalAdminName = "Site Collection Admin"
4 const newDDD = "Tom3641 test Description 25666"
5
6 //Passes
7 const newAdminName = "Site Collection Admin"
8 const newDescription = 'Automated UI Test Section Description'
```

考虑使用const声明

当变量不会被重新赋值时，考虑推荐用const声明代替let或var

```
1 //Fails
2 let errorMessage = 'The entered User ID or Password is incorrect'
3 var errorMessage = 'The entered User ID or Password is incorrect'
4
5 //Passes
6 const errorMessage = 'The entered User ID or Password is incorrect'
```

不要存留未引用的变量声明

```
1 //Fails
2 it("Input the time", function () {
3     const time = new Date()
4     cy.get("input").type("2020-07-12");
5 });
6
7 //Passes
8 it("Input the time", function () {
9     cy.get("input").type("2020-07-12");
10 });
```

不要出现绝对地址

不要出现'<https://test.sharepoint.com>' 这样的绝对地址，不易于迁移到其他测试环境中，使用环境配置文件或者baseUrl+相对路径

```
1 //Fails
2 before(function () {
3     cy.visit("https://Test.yourapp.com/CCQA");
4 });
5
6 //Passes
7 // cypress.json
8 { "baseUrl":"https://Test.yourapp.com"}
9 before(function () {
10     cy.visit('/CCQA');
11 });
```

不要使用自动定位，尽量避免使用无意义的选择器

获取元素的时候，尽量使选择器的含义清晰，明了，有可读性，尽量避免使用自动获得的定位

The screenshot displays a web application interface for exam setup. The top navigation bar shows the Cypress command `cy.get` with a selector `[style="float: left;"] > .button > .button-text`. The main content area is titled "Exam_Cypress_ysyu_12/2/2020, 5:11:38 PM". It features a sidebar menu with options: Home, Exam, Bank, and Admin. The "Exam" section is active, showing a list of steps: 1. Set up basic information, 2. Enrol candidates, 3. Assign invigilator(s), and 4. Generate paper. The main panel contains fields for "START DATE AND TIME" (31 Dec 2020 17:10), "END DATE AND TIME" (31 Dec 2020 18:10), and "DURATION" (60 mins). Below these fields is a button "Add Paper from Bank" and a dropdown menu "+ Create paper". At the bottom, there is a table with columns "Paper name", "Modified by", and "Modified date and time".

```

1 //Bad Example:
2 cy.get('[style="float: left;"] > .button > .button-text')
3
4 //Good Example:
5 cy.get('#right-panel [aria-label="Add Paper from Bank"]')

```

2.2. Case规范

文件命名:

文件名尽量能概括表示所属Case的含义，并遵循pascal方式，即所有单词首字母大写，e.g: SoftwareWhiteList.js

02.Admin	13
> 01_RM	14
> 02_GS	15
JS Announcement_templateBank.js	16
JS Announcement.js	17
JS Calendar.js	18
JS CustomGroup.js	19
JS FavorLink.js	20
JS SoftwareWhiteList.js	22
JS Venue.js	23
JS Workflow.js	24

如果Case之间存在前后文关系，有一定的业务关联性，即下一条Case需要前一条Case制造条件，并且都需要测试，可以考虑为Case添加编号

03.Schedule	39
JS 00_Assign_ACAD.js	40
JS 01_AdminSetupApproveProcess.js	41
JS 02_Module_Assessment_Setup.js	44
	45

内容：3A模式

Case内容，应当尽量遵循3A模式的结构，即包含 Arrange, Action, Assert 三个部分

- Arrange 明确定义当前Case所需静态数据，变量等（非必须）
- Action 明确指出当前Case所执行的具体操作动作
- Assert 明确设置断言，为操作结果提供验收标准

```

//Arrange
let annTitle = tempTitle
const announceStu = 'CASTU0100'
Then(/^I fill Announcement form and Save$/, () => {
  // Action: Select Template
  cy.get('#create-mode-radio-1--radio').check()
  cy.get('.announcement-template-border input').eq(0).click({ force: true }).wait(500)
    .type('{downarrow}{uparrow}', { delay: 150 })
    .type('{enter}')
  //Select Participant
  cy.ParticipantSelection('#aui-folding-input-1', "Individual", announceStu)
  cy.get('.button-footer').contains('Save').click().wait(2000)
});

Then(/^I should see announcement create successful$/, () => {
  // Assert
  cy.get('[rowindex=0]').should('contain',annTitle)
  //should see two channel icon
  cy.get('[rowindex="0"] [data-columnindex="8"] span').should('have.class', 'text-green')
});

```

3.常用断言

```

1 cy.url().should('include','/abc')
2 //判断URL是否包含/abc
3
4 cy.contains('想要定位的字符串')
5 //判断页面上是否包含想要的字符串
6
7 cy.get('selector').contains('字符串')
8 //判断某个元素下是否包含所需字符串
9
10 cy.get('selector').should('contain','字符串')
11 //作用同上
12
13 cy.get('selector').should('have.class','success')
14 //判断元素的class是不是success
15
16 cy.get('selector').should('have.attr','href','/managelinks')
17 //判断元素的href属性的值，是不是/managelinks
18
19 cy.expect($headers.length).to.eq(4)
20 //判断一个数组的长度，等于4

```

