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，没有人比你自己更了解它的业务，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，你的测试结果也极容易变的不稳定，所以尽量使用自己可以信赖的数

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，也能够在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）

```
//Fails
    cy.get(".Button").click();
        // Assert
           cy.get(".nav-title").should("exist");
 cy.get(".detailslist").should("not.exist");

//Passes
        cy.get(".Button").click();
```

```
        // Assert
        cy.get(".nav-title").should("exist");
        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

```
//Fails
cy.contains('MY FAVOURITE LINKS').next().contains('Manage').as('ManageLinkBtn').should('have.attr','href','/admin/managelinks')

//Passes
cy.contains('MY FAVOURITE LINKS').next().contains('Manage')
  .as('ManageLinkBtn')
  .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

```
//Fails
    it("Input a wrong site URL", function () {


        cy.get("#search-box01").clear().type(siteURL);

        cy.get(".validate-alert").should("have.length", 1);
    });

//Passes
    it("Input a wrong URL", function () {
        // Act
        cy.get("#search-box01").clear().type(scURL);
        // Assert
        cy.get(".validate-alert").should("have.length", 1);
    });
```

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

```
//Fails
const newadminname = "Site Collection Admin"
const OriginalAdminName = "Site Collection Admin"
const newDDD = "Tom3641 test Description 25666"

//Passes
const newAdminName = "Site Collection Admin"
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

```
//Fails
let errMessage = 'The entered User ID or Password is incorrect'
var errMessage = 'The entered User ID or Password is incorrect'

//Passes
const errMessage = 'The entered User ID or Password is incorrect'
```

不要存留未引用的变量声明

```
//Fails
it("Input the time", function () {
    const time = new Date()
    cy.get("iuput").type("2020-07-12");
});

//Passes
it("Input the time", function () {
    cy.get("iuput").type("2020-07-12");
});
```

不要出现绝对地址

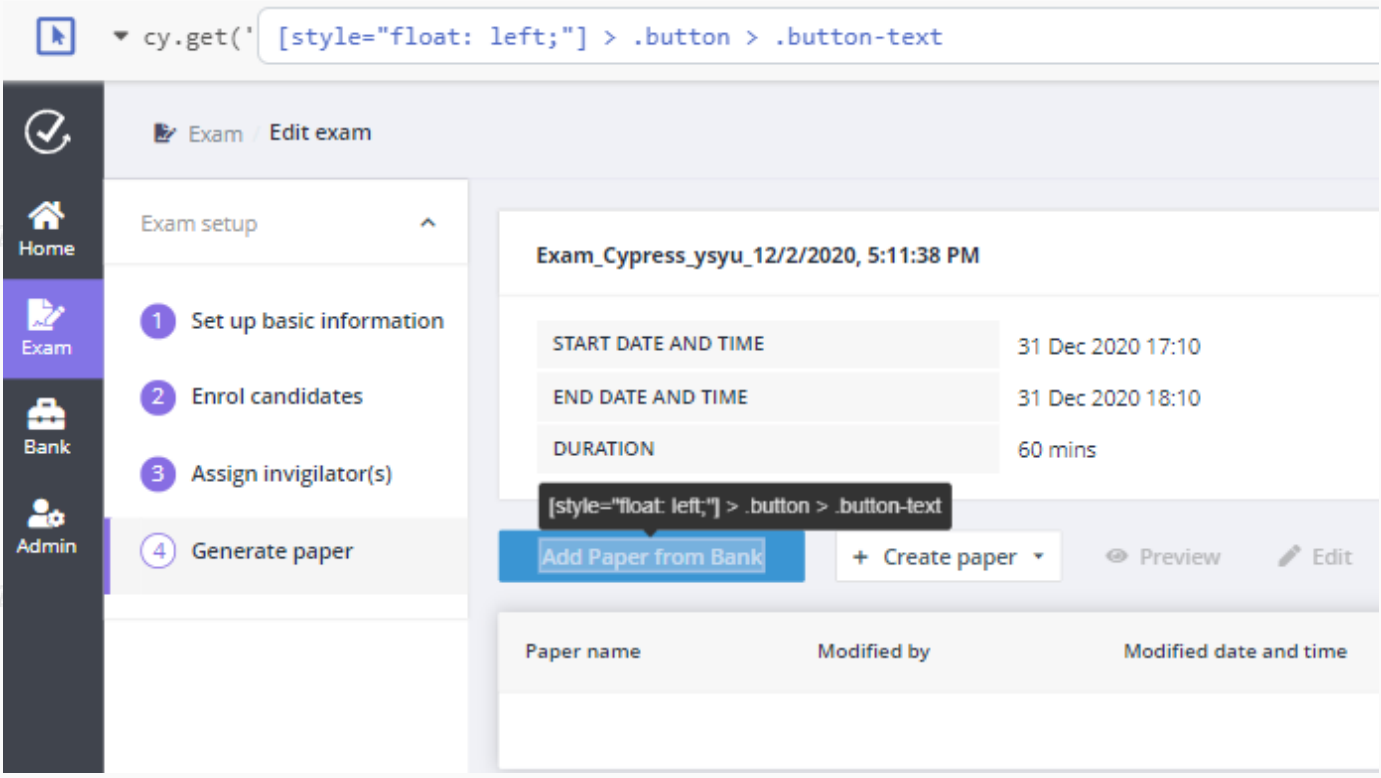
不要出现'https://test.sharepoint.com' 这样的绝对地址，不易于迁移到其他测试环境中，使用环境配置文件或者baseUrl+相对路径

```
//Fails
before(function () {
        cy.visit("https://Test.yourapp.com/CCQA");
    });

//Passes
 // cypress.json
 { "baseURL":"https://Test.yourapp.com}
before(function () {
        cy.visit('/CCQA');
    });
```

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



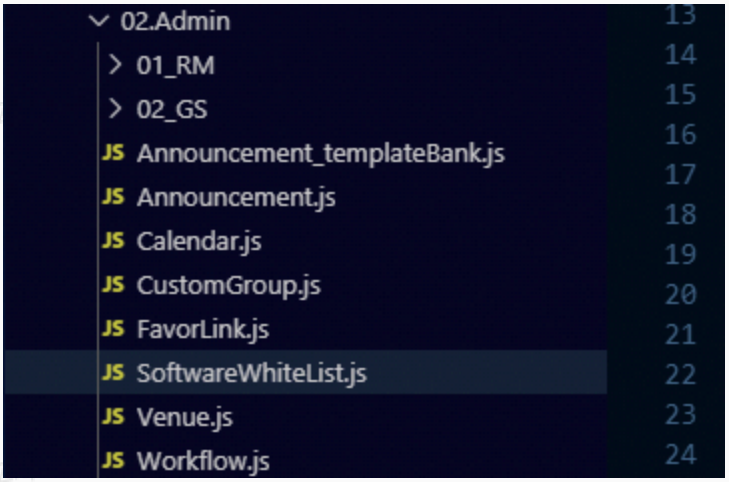
```
//Bad Example:
cy.get('[style="float: left;"] > .button > .button-text')

//Good Example:
cy.get('#right-panel [aria-label="Add Paper from Bank"]')
```

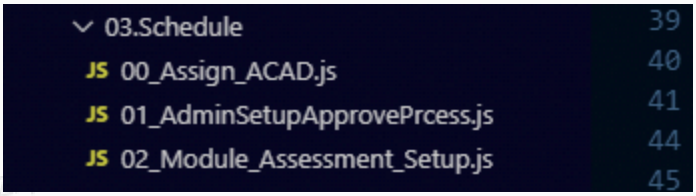
2.2. Case规范

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



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



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

```
cy.url().should('include','/abc')
//判断URL是否包含/abc

cy.contains('想要定位的字符串')
//判断页面上是否包含想要的字符串

cy.get('selector').contains('字符串')
//判断某个元素下是否包含所需字符串

cy.get('selector').should('contain','字符串')
//作用同上

cy.get('selector').should('have.class','success')
//判断元素的class是不是success

cy.get('selector').should('have.attr','href','/managelinks')
//判断元素的href属性的值，是不是/managelinks

cy.expect($headers.length).to.eq(4)
//判断一个数组的长度，等于4
```