Jasmine

Behavior-Driven Javascript

NHÓM 9:

- TRẦN QUANG TOÀN
- NGUYỄN THỊ NHÀN
- NGUYỄN ĐỨC THẮNG
- NGUYỄN VĂN PHÚC

Giới thiệu

- Jasmine là một framework kiểm thử cho javascript
- Jasmine được sử dụng phù hợp cho các websites, các ứng dụng nodejs và bất kỳ nơi nào mà javascript có thể chạy
- Tham khảo thêm về Jasmine tại địa chỉ: http://jasmine.github.io

Cài đặt Jasmine

- Đối với NODEJS:
  - Thêm Jasmine vào package.json:
    - `npm install –save-dev jasmine`
  - Khởi tạo Jasmine vào project của bạn:
    - `node node_modules/jasmine/bin/jasmine init`
  - Cài đặt Jasmine làm test script vào trong package.json:
    - `“script”: {“test”: “jasmine”}`
  - Chạy các kiểm thử của bạn:
    - `npm test`

Cài đặt Jasmine

- Đối với Ruby Project:
  - Tham khảo tại: https://jasmine.github.io/setup/ruby.html
- Đối với Python Project:
  - Tham khảo tại: https://jasmine.github.io/setup/python.html

Sử dụng Jasmine

- Hàm describe:
  - Được dùng để gộp nhóm các ca kiểm thử có liên quan.
  - Các tham số:
    - Tên nhóm các ca kiểm thử
    - Hàm bọc các ca kiểm thử có liên quan

```
describe(“A group tests”, function() {
    it(“A test case”, function() {
        expect(true).toBe(true);
    })
})
```

Sử dụng Jasmine

- Hàm it:
  - Được dùng để khai báo một ca kiểm thử
  - Các tham số:
    - Tên ca kiểm thử
    - Hàm bọc các đoạn mã kiểm thử

```
describe(“A group tests”, function() {
    var a;
    it(“A test case”, function() {
        a = true;
        expect(a).toBe(true);
    })
})
```

Sử dụng Jasmine

- Matchers
  - Mỗi matcher thực hiện một so sánh giữa giá trị thực (actual value) và giá trị kỳ vọng(expected value)
  - Matcher có trách nhiệm báo cho Jasmine xem phép so sánh là đúng hay sai và Jasmine sẽ gán cho ca kiểm thử đã thành công hay thất bại.
  - Tìm hiểu thêm về matcher tại: https://jasmine.github.io/api/edge/matchers.html

```
describe("Matchers", function() {

    it("matcher has positive case", function() {
        expect(true).toBe(true);
    });

    it("matcher has negative case", function() {
        expect(false).not.toBe(true);
    });
})
```

Sử dụng Jasmine

- Một số matchers thông dụng:
  - expect(x).toEqual(y): So sánh 2 đối tượng hoặc dữ liệu nguyên thủy x và y.
  - expect(x).toBe(y): So sánh x và y và pass nếu như chúng là cùng 1 object.
  - expect(x).toMatch(pattern): So sánh x và biểu thức chính quy pattern và pass nếu chúng trùng nhau.
  - expect(x).toContain(y): pass nếu mảng hoặc chuỗi x có chứa y
  - expect(x).toBeNull(): pass nếu x là null
  - expect(x).toBeLessThan(y): pass nếu x < y
  - expect(x).toBeGreaterThan(y): pass nếu x > y

Sử dụng Jasmine

- Jasmine cung cấp một vài hàm toàn cục giúp bạn tránh lặp code:
  - beforeEach: Được gọi trước khi thực hiện 1 ca kiểm thử
  - afterEach: được gọi sau khi thực hiện 1 ca kiểm thử
  - beforeAll: Được gọi một lần duy nhất trước khi tất cả các ca kiểm thử trong hàm describe được gọi
  - afterAll: Được gọi một lần duy nhất sau khi toàn bộ các ca kiểm thử được chạy.

```
describe(“DRY”, function() {
    var foo = 0;

    beforeEach(() => (foo += 1));

    afterEach(() => (foo = 0));

    beforeAll(() => (foo = 1));

    afterAll(() => (foo = 0));
})
```

Sử dụng Jasmine

- Từ khóa `this`:
  - Ta có thể dùng từ khóa `this` để chia sẻ biến giữa các hàm beforeEach, it và afterEach.
  - Mỗi nhóm ca kiểm thử (spec) sẽ có một đối tượng trống `this` và sẽ được chuyển về trống khi chuyển tới spec tiếp theo.

```
describe(“A spec”, function() {

    beforeEach(function() {
        this.foo = 0;
    });

    it(“can use the ‘this’ to share state”, function() {
        expect(this.foo).toEqual(0);
    })
})
```

Sử dụng Jasmine

- Bọc các khối describe:
  - Gọi hàm describe có thể được bọc bởi hàm describe khác
  - Điều này cho phép có thể tạo một bộ kiểm thử dưới dạng cây.
  - Trước khi một bộ kiểm thử(spec) được thực hiện, Jasmine đi xuống từng nhánh cây để thực hiện các hàm beforeEach, và sau khi spec được thực hiện, Jasmine làm điều tương tự với các hàm afterEach

```
describe(“A spec”, function() {
    var foo = 0;

    beforeEach(() => (foo = 1));

    describe(“A nested spec”, function() {
        var bar = 0;
        beforeEach(() => (bar = 1));
        it(“foo must be equal bar”, function() {
            expect(foo).toEqual(bar);
        })
    })
})
```

Sử dụng Jasmine

- Hàm gián điệp (spies):
  - Một hàm spy có thể được cài vào bất kỳ một hàm và theo dõi các lần gọi hàm và tất cả các đối số.
  - Một hàm spy chỉ tồn tại trong describe hoặc it và sẽ được xóa sau mỗi spec.
  - Một số matchers đặc biệt được dùng để tương tác với spies:
    - toHaveBeenCalled sẽ được xem là pass nếu như spy được gọi.
    - toHabeBeenCalledTimes được xem là pass nếu như spy được gọi theo một số lần nhất định.
    - toHaveBeenCalledWith sẽ trả về true nếu danh sách đối số trùng với đối số được ghi nhận khi gọi spy

Sử dụng Jasmine

- Hỗ trợ bất đồng bộ:
  - Các hàm mà ta truyền vào `beforeAll`, `afterAll`, `beforeEach`, `afterEach` và `it` có thể là bất đồng bộ.
  - Có 3 cách thông dụng để chỉ ra một hàm là bất đồng bộ là:
    - Dùng hàm callback
    - Trả về promise
    - Sử dung từ khóa async trong môi trường có hỗ trợ

```
describe("A spec", function() {
    var value;
    describe("using callback", function() {
        beforeEach(function(done) {
            setTimeout(done, 1000);
        })
    })
    describe("using promise", function() {
        afterEach(function() {
            return doAsync().then(doSomething);
        })
    })
    describe("using async", function() {
        beforeEach(async function() {
            await doAsync();
        })
    })
})
```

Tìm hiểu thêm về Jasmine

https://jasmine.github.io/pages/docs_home.html

Cảm ơn mọi người đã lắng nghe