Dự án 2: JavaScript

Cài đặt

Mặc dù dự án này yêu cầu bạn chạy mã trong trình duyệt nhưng bạn cần cài đặt Node.js trên hệ thống của mình để chạy trình kiểm tra chất lượng mã. Nếu bạn chưa cài đặt Node.js và trình quản lý gói npm, hãy làm theo hướng dẫn cài đặt trên trang web Node.js.

Khi bạn đã cài đặt Node.js, hãy tạo một thư mục `project2` và trích xuất nội dung của tập tin project2.zip vào thư mục. Tệp zip chứa các tệp `test-project2.html` và `test-project2.js` hoạt động như một khung thử nghiệm cho mã bạn viết trong dự án này.

Bạn có thể tìm nạp công cụ chất lượng mã (còn được gọi là "kẻ nói dối"), ESLint, bằng cách chạy lệnh sau trong thư mục `project2` :

```
cài đặt npm
```

Điều này sẽ lấy ESLint vào thư mục con `node_modules` . Bạn sẽ có thể chạy nó trên tất cả các tệp JavaScript trong thư mục `project2` bằng cách chạy lệnh:

```
npm chạy lint
```

Mã bạn gửi phải bắt đầu bằng `"use strict";` trong mỗi tệp JavaScript và chạy ESLint bằng lệnh trên sẽ không xuất ra bất kỳ lỗi hoặc cảnh báo nào. Mọi lỗi hoặc cảnh báo sẽ được dùng để trừ điểm phong cách.

Để thực hiện nhiệm vụ, bạn có hai lựa chọn:

1. Trong trình duyệt của bạn: Mở tệp `test-project2.html` trong trình duyệt của bạn. Trang web sẽ tải mã JavaScript bạn đã viết và chạy một vài thử nghiệm.
2. Trong Node.js: Bạn cũng có thể chạy thử nghiệm mà không cần trình duyệt bằng lệnh

```
kiểm tra npm
```

Chúng tôi khuyên bạn nên sử dụng trình duyệt để phát triển vì môi trường gỡ lỗi tốt hơn nhiều.

Lưu ý: Những thử nghiệm này không bao gồm tất cả các trường hợp khó khăn. Họ ở đó để giúp hướng dẫn bạn và cho bạn biết khi nào bạn có chức năng cơ bản. Trách nhiệm của bạn là xử lý mọi thứ được nêu trong các thông số kỹ thuật sau đây chưa được kiểm tra rõ ràng trong tệp thử nghiệm mà chúng tôi cung cấp cho bạn.

Trong dự án này, chúng tôi yêu cầu bạn viết hoặc sửa đổi một số hàm JavaScript. Các vấn đề trong bài tập này có tính chất thực tế và chức năng mà bạn phát triển sẽ hữu ích trong việc hoàn thành các dự án lớp sau. Với sự sẵn có của các thư viện JavaScript để giải quyết hoặc giúp giải quyết hầu hết mọi tác vụ JavaScript mà bạn được giao, rất có thể bạn có thể giải quyết những vấn đề này chỉ bằng một vài

số dòng để gọi một số thủ tục thư viện. Vì mục tiêu của dự án là học JavaScript nên chúng tôi cấm bạn sử dụng bất kỳ thư viện JavaScript nào trong các giải pháp của mình. Các hàm tích hợp sẵn trong JavaScript, như các đối tượng Mảng và Ngày, đều được chấp nhận.

Vấn đề 1: Tạo ra MultiFilterFunction (10 điểm)

Trong thư mục project2 của bạn , hãy tạo một tệp mới có tên make-multi-filter.js. Mã cho Chức năng Đa bộ lọc của bạn sẽ có trong tệp này.

Khai báo một hàm toàn cục có tên MakeMultiFilter lấy một mảng (originArray) làm tham số và trả về một hàm có thể được sử dụng để lọc các phần tử của mảng này. Hàm được trả về (arrayFilterer) theo dõi nội bộ một khái niệm gọi là currentArray.
Ban đầu, currentArray được đặt giống với originalArray. Hàm arrayFilterer lấy hai hàm làm tham số. Họ đang:

1. filterCriteria - Hàm lấy một phần tử mảng làm tham số và trả về một boolean. Hàm này được gọi trên mọi phần tử của currentArray và currentArray được cập nhật để phản ánh kết quả của hàm filterCriteria . Nếu hàm filterCriteria trả về sai cho một phần tử thì phần tử đó sẽ bị xóa khỏi currentArray. Nếu không, nó sẽ được để lại trong currentArray. Nếu filterCriteria không phải là một hàm thì hàm được trả về (arrayFilterer) sẽ ngay lập tức trả về giá trị của currentArray mà không thực hiện lọc.

2. gọi lại - Một hàm sẽ được gọi khi quá trình lọc hoàn tất. cuộc gọi lại lấy giá trị của currentArray làm đối số. Việc truy cập cái này bên trong hàm gọi lại sẽ tham chiếu giá trị của originalArray. Nếu gọi lại không phải là một chức năng thì nên bỏ qua. gọi lại không có giá trị trả về.

Hàm arrayFilterer sẽ tự trả về trừ khi tham số filterCriteria không được chỉ định trong trường hợp đó nó sẽ trả về currentArray . Phải có nhiều hàm arrayFilterer hoạt động cùng một lúc.

Đoạn mã sau đây cho thấy cách người ta có thể sử dụng các hàm bạn xác định trong bài toán này:

```
// Gọi MakeMultiFilter() với originalArray = [1, 2, 3] trả về một

// hàm, được lưu trong biến arrayFilterer1, có thể được sử dụng để

// lọc liên tục mảng đầu vào

var arrayFilterer1 = MakeMultiFilter([1, 2, 3]);


// Gọi arrayFilterer1 (có chức năng gọi lại) để lọc ra tất cả số
rs

// không bằng 2.

mảngFilterer1(hàm (elem) {

   trả về phần tử !== 2; // kiểm tra xem phần tử có bằng 2 không

}, hàm (currentArray) {
```

```
   // 'this' trong hàm gọi lại phải tham chiếu đến originalArray là [1, 2, 3]

   console.log(cái này); // in [1, 2, 3]
   console.log(currentArray); // in [1, 3]
});

// Gọi arrayFilterer1 (không có hàm gọi lại) để lọc ra tất cả
// các phần tử không bằng 3.
mảngFilterer1(hàm (elem) {
   trả về phần tử !== 3; // kiểm tra xem phần tử có bằng 3 không
});

// Gọi arrayFilterer1 không có filterCriteria sẽ trả về currentArr ay.

var currentArray = arrayFilterer1();
console.log("currentArray", currentArray); // in ra [1] vì chúng ta đã lọc ra 2 và 3

// Vì arrayFilterer trả về chính nó nên các cuộc gọi có thể được nối tiếp
hàm filterTwos(elem) { return elem !== 2; }
hàm filterThrees(elem) { return elem !== 3; }
var arrayFilterer2 = MakeMultiFilter([1, 2, 3]);
var currentArray2 = arrayFilterer2(filterTwos)(filterThrees)();
console.log("currentArray2", currentArray2); // in [1] vì chúng tôi đã lọc
ra 2 và 3

// Nhiều bộ lọc hoạt động cùng lúc var arrayFilterer3 =
MakeMultiFilter([1, 2, 3]);
var arrayFilterer4 = MakeMultiFilter([4, 5, 6]);
console.log(arrayFilterer3(filterTwos)()); // in [1, 3]
console.log(arrayFilterer4(filterThrees)()); // in [4, 5, 6]
```

Vấn đề 2: Bộ xử lý mẫu (5 điểm)

Trong thư mục `project2` của bạn , tạo một tệp mới có tên `template-processor.js`. Mã cho Bộ xử lý mẫu của bạn sẽ có trong tệp này.

Tạo một lớp bộ xử lý mẫu (TemplateProcessor) được xây dựng bằng mẫu tham số chuỗi và có phương thức fillIn. Khi được gọi với một đối số của một đối tượng từ điển , fillIn trả về một chuỗi có mẫu chứa đầy các giá trị từ đối tượng từ điển. Bộ xử lý mẫu phải được viết bằng cách sử dụng cấu trúc nguyên mẫu và hàm tạo JavaScript tiêu chuẩn.

Phương thức fillIn trả về chuỗi mẫu với bất kỳ văn bản nào có dạng {{property}} được thay thế bằng thuộc tính tương ứng của đối tượng từ điển được truyền cho hàm.

Nếu mẫu chỉ định một thuộc tính không được xác định trong đối tượng từ điển thì thuộc tính đó sẽ được thay thế bằng một chuỗi trống. Nếu thuộc tính nằm giữa hai từ, bạn sẽ nhận thấy rằng việc thay thế thuộc tính bằng một chuỗi trống sẽ tạo ra hai khoảng trắng liên tiếp.
Ví dụ: "{{unfinedProperty}} này thật tuyệt" "Cái này thật tuyệt". Điều này ổn. Bạn không phải lo lắng về việc loại bỏ khoảng trắng thừa.

Hệ thống của bạn chỉ cần xử lý các thuộc tính được định dạng đúng. Hành vi của nó có thể không được xác định trong các trường hợp sau vì chúng tôi sẽ không kiểm tra chúng một cách rõ ràng.

- các thuộc tính lồng nhau - {{foo {{bar}}}} hoặc {{{{bar}}}} hoặc {{{bar}}}
- dấu ngoặc không cân bằng - {{bar}}}
- dấu ngoặc đơn trong bất kỳ chuỗi thuộc tính nào - day{y hoặc day}y

Đoạn mã sau đây cho thấy cách người ta có thể sử dụng các hàm bạn xác định trong bài toán này:

```
var template = "Tháng yêu thích của tôi là {{tháng}} chứ không phải ngày {{day}} hay năm {{year}}";

var dateTemplate = new TemplateProcessor(template);


từ điển var = {tháng: "Tháng 7", ngày: "1", năm: "2016"};

var str = dateTemplate.fillIn(dictionary);


khẳng định(str === "Tháng yêu thích của tôi là tháng 7 nhưng không phải ngày 1 hay năm 2016"
);


// Trường hợp: thuộc tính không tồn tại trong từ điển

var từ điển2 = {ngày: "1", năm: "2016"};

var str = dateTemplate.fillIn(dictionary2);


khẳng định(str === "Tháng yêu thích của tôi là nhưng không phải ngày 1 hay năm 2016");
```

Vấn đề 3: Sửa lỗi test-project2.js tonotpollutetheglobalnamespace(5points)

Tệp JavaScript thử nghiệm mà chúng tôi cung cấp cho bạn (test-project2.js) khai báo nhiều ký hiệu trong không gian tên JavaScript chung. Ví dụ: sau khi tập lệnh được tải, ký hiệu p1Message sẽ xuất hiện trong không gian tên chung. Sau đó, một tệp JavaScript khác có thể truy cập và thay đổi p1Message. Thay đổi test-project2.js để sử dụng mẫu mô-đun JavaScript tiêu chuẩn bằng cách sử dụng hàm ẩn danh để ẩn các ký hiệu trong không gian tên chung nhưng vẫn giữ nguyên chức năng kiểm tra.

StylePoints(5 điểm)

Những điểm này sẽ được trao nếu mã JavaScript của bạn giải quyết các vấn đề trên rõ ràng, dễ đọc và không có cảnh báo ESLint.