LAB 8: ASYNCHRONOUS JAVASCRIPT: PROMISES, ASYNC/AWAIT AND AJAX

MỤC TIÊU

Kết thúc bài thực hành này bạn có khả năng

- ✓ Bất đồng bộ trong Javascript (Asynchronous Javascript): Event Loop, Callback Hell, Promises, async/await, AJAX calls and APIs, Throw Errors, Error Handling,….

NỘI DUNG

LAB8.1: (SECTION 16: ASYNCHRONOUS JAVASCRIPT: PROMISES, ASYNC/AWAIT AND AJAX > CODING CHALLENGE #1)

Trong bài này bạn cần phải xây dựng một function 'whereAmI' để render ra 1 quốc gia dựa theo tọa độ GPS. Để làm được điều này bạn cần sử dụng một API thứ 2 để cho tọa độ mã hóa địa lý
Đây là nhiệm vụ của bạn:

PART 1

1. Tạo function whereAmI , whereAmI nhận 2 tham số đầu vào: vĩ độ (lat) và kinh độ (lng) (Đây là tọa độ GPS, ví dụ bên dưới)
2. Thực hiện 'reverse geocoding' của các tọa độ được cung cấp. 'Reverse geocoding' có nghĩa là nó sẽ chuyển đổi từ tọa độ thành một địa chỉ có ý nghĩa như Hà Nội,Hải Phòng,... . Sử dụng API này để 'reverse geocoding': https://geocode.xyz/api.
AJAX sẽ call API với địa chỉ URL dạng này: https://geocode.xyz/52.508,13.381?geoit=json. Sử dụng fetch API kết hợp với promises (then,catch) để lấy dữ liệu. Không được sử dụng function getJSON
3. Khi bạn đã có dữ liệu, hãy thử log nó ra để xem được tất cả các thuộc tính nhận được từ vị trí được cung cấp. Dùng data đó để log ra một dòng như: 'You are in Berlin, Germany';
4. Sử dụng method .catch() cuối mỗi promise để có thể bắt được lỗi khi thực hiện thất bại

5. API trên chỉ cho phép gửi 3 requests trong một giây. Nếu gửi quá nhanh bạn sẽ nhận được về mã lỗi 403. Chú ý có thể method .catch của fetch() sẽ không được chạy trong trường hợp này. Vì vậy hãy tự log response ra và tạo một message lỗi có ý nghĩa

PART 2

6. Sử dụng data nhận được để render ra một quốc gia. Hãy lấy thuộc tính liên quan từ result API và cắm nó vào API đang sử dụng???(có lẽ chỗ này dịch sai)
7. Render ra một quốc gia mà không có lỗi nào xảy ra

TEST COORDINATES 1: 52.508, 13.381 (Latitude, Longitude)
TEST COORDINATES 2: 19.037, 72.873
TEST COORDINATES 2: -33.933, 18.474

LAB8.2: (SECTION 16: ASYNCHRONOUS JAVASCRIPT: PROMISES, ASYNC/AWAIT AND AJAX > CODING CHALLENGE #2)

Xây dựng một chức năng loading ảnh và hiển thị ra màn hình

PART 1

1. Khởi tạo một function 'createImage' tham số đầu vào là imgPath. Function này sẽ return về một Promise với chức năng là tạo ra 1 image mới (dùng document.createElement('img')) thuộc tính src của new image sẽ là tham số imgPath được truyền vào. Khi image được tải xong hãy append image element vừa tạo vào dom và thay thế element có selector là ".image". Trong trường hợp load ảnh bị lỗi hãy reject promise đi

Nếu phần này quá khó thì chỉ cần xem phần đầu tiên của phần solution

PART 2

2. Thực hiện promise bằng .then và cũng thêm .catch để xử lý lỗi
3. Sau khi load xong image hãy tạm dừng thực thi trong 2s bằng function đã tạo trước đó
4. Sau khi chờ xong 2s, hãy ẩn image hiện tại (style = display: none) và load tiếp hình ảnh thứ 2
5. Sau khi image thứ 2 đang load. thì cũng tạm dừng thực thi trong 2s
6. Sau 2s thì sẽ ẩn hình ảnh hiện tại

(Gợi ý: Hiểu đơn giản là 1 hình ảnh sau khi được load xong sau 2s sẽ bị ẩn đi và hiện hình ảnh tiếp theo lên)

TEST DATA: Test data bằng cách đặt sai đường dẫn ảnh hoặc chỉnh tốc độ mạng xuống còn Fast 3G trong: F12 => Network => biểu tượng bên cạnh icon wifi

LAB8.3: (SECTION 16: ASYNCHRONOUS JAVASCRIPT: PROMISES, ASYNC/AWAIT AND AJAX > CODING CHALLENGE #3)

PART 1

Viết lại async function 'loadNPause' để thực hiện lại bài tập 2, sử dụng async/await (chỉ phần promise được sử dụng). So sánh ưu nhược điểm của 2 cách dùng, bạn thấy cách nào dễ dùng hơn

Nhớ để tốc độ mạng xuống 'Fast 3G'

PART 2

1. Khởi tạo 1 function bất đồng bộ 'loadAll' tham số nhận vào là một mảng các đường dẫn ('imgArr)
2. Sử dụng method .map để lặp qua lần lượt các giá trị, để tải lên tất các image bằng 'createImage' function (gọi kết quả trả về của .map là imgs)
3. Log ra 'imgs' xem nó có nhận được kết quả mong muốn không?😉
4. Sử dụng 'promise combinator' để có thể lấy ra được images element từ mảnh
5. Các imageElement nhận được sẽ thêm 1 class 'paralell' (thêm cho nó một CSS riêng để thấy sự khác biệt)

TEST DATA: ['img/img-1.jpg', 'img/img-2.jpg', 'img/img-3.jpg']. Để test, bạn cần tắt function “loadNPause”

--- Hết ---