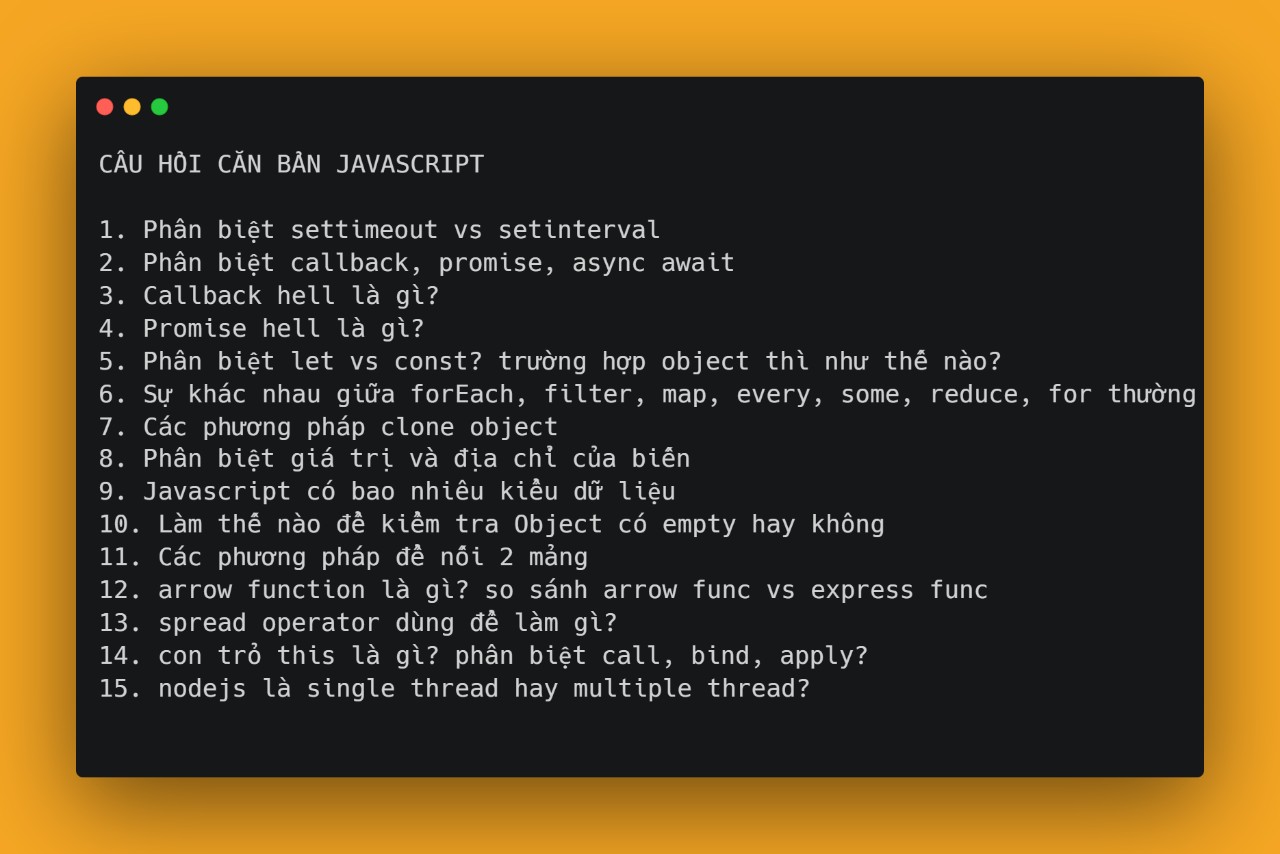
Câu hỏi căn bản JavaScript



Câu trả lời

[**Câu 1: Phân biệt settimeout và setinterval** 2](#_Toc116735695)

[**Câu 2: Phân biệt callback, promise, async await** 3](#_Toc116735696)

[**Câu 3: Callback hell là gì?** 5](#_Toc116735697)

## **Câu 1: Phân biệt settimeout và setinterval**

* Hàm setTimeout() thường được sử dụng khi mình muốn thực hiện method sau khoảng thời gian nào đó (mili giây)

setTimeout ( expression, timeout );

Như cú pháp trên thì expression sẽ được thực hiện sau timeout (mili giây)

setTimeout(() => {

    console.log("Hello")

}, 1000);

Vị dụ khi load trang thì sau 1000 mili giây sẽ in ra chữ “Hello”

* Hàm setinterval sẽ dùng khi mình muốn thực hiện method liên tục sau khoảng thời gian chờ nào đó.

setInterval ( expression, time );

Như cú pháp trên thì expression sẽ được thực hiện sau mỗi time (mili giây)

setInterval(() => {

    console.log("Bye")

}, 2000)

Ví dụ sao khi load trang 2 giây sẽ in ra chữ “Bye”, và cứ sau 2 giây lại in ra chữ “Bye”.

## **Câu 2: Phân biệt callback, promise, async await**

Điểm chung là đều dùng để sử lí bất đồng bộ

* Callback là một function được gọi ra khi một thao tác bất đồng bộ có kết quả đã sẵn sàng (kết quả này có thể là dữ liệu trả về hoặc lỗi xảy ra khi thao tác)

Callback thường được sử dụng nhiều trong Jquery:

$("#button").click(function() {

    alert("Click")

})

Thì hàm callback là:

function() {

    alert("Click")

}

* Promíe nó có cú pháp rõ ràng hơn

const myPromise = new Promise(function(resolve, reject) {

    // code here

    if (codeIsFine) {

        resolve('fine')

    } else {

        reject('error')

    }

})

myPromise

    .then(function whenOk(response) {

        console.log(response)

        return response

    })

    .catch(function notOk(err) {

        console.error(err)

})

Hãy phân tích đoạn code trên:

Một promise được khởi tạo với một function có câu lệnh resolve hoặc reject

Đặt đoạn code bất đồng bộ trong function Promise resolve khi mọi thứ xảy ra như mong muốn Nếu không thì reject

Khi resolve được chạy thì đoạn code trong .then sẽ thực thi Khi reject được chạy thì .catch sẽ được trigger

* Async sẽ thông báo rằng function sẽ xử lý bất đồng bộ, và await sẽ được dùng để báo chúng ta muốn đợi kết quả của thao tác bất đồng bộ trong một function có đánh dấu async.

## **Câu 3: Callback hell là gì?**

Callback hell kiểu như đã là mình gọi callback quá nhiều lần, sẽ làm code bị xấu. Ví dụ:

const animateAll = (animate) => {

        setTimeout(() => {

            animate(words[0]);

            setTimeout(() => {

                animate(words[1]);

                setTimeout(() => {

                    animate(words[2]);

                }, 1000)

            }, 1000)

        }, 1000)

    }

Làm thế nào để thoát khỏi callback hell?

* JavaScript cung cấp một cách dễ dàng để thoát khỏi callback hell. Điều này được thực hiện bởi hàng đợi sự kiện (event queue) và các promise.
* Một promise là một đối tượng (object) được trả về (return) từ bất kỳ hàm không đồng bộ (asynchronous function) nào, mà các hàm callback (callback function) có thể được thêm vào dựa trên kết quả của hàm trước đó.
* Promise sử dụng phương thức (method) .then() để gọi các callback không đồng bộ (async callback). Chúng ta có thể nối (chain) bao nhiêu callback tùy thích và thứ tự (order) cũng được duy trì nghiêm ngặt.
* Promise sử dụng phương thức .fetch() để tìm nạp một đối tượng từ mạng (network). Nó cũng sử dụng phương thức .catch() để bắt (catch) bất kỳ ngoại lệ (exception) nào khi bất kỳ khối (block) nào bị lỗi.
* Vì vậy, những promise này được đưa vào hàng đợi sự kiện để chúng không chặn (block) mã JS tiếp theo. Ngoài ra khi kết quả được trả về, hàng đợi sự kiện sẽ kết thúc hoạt động của nó.
* Ngoài ra còn có các từ khóa (keyword) và phương thức hữu ích khác như async, wait, settimeout() để đơn giản hóa và sử dụng tốt hơn các callback.

## **Câu 4: Promise hell là gì?**

Tuy nói Promise giúp tổ chức code, và tránh callback hell, nhưng nếu viết code không khô thoáng và sạch sẽ thì anh em vẫn rơi vào Promise Hell. Do đó lúc nào chúng ta cũng phải dùng kotexcode.

Promise Hell

funA()

    .then(function(tien) {

        return funB(tien)

            .then(function(tien\_con\_lai) {

                return funC(tien\_con\_lai)

                    .then(function() {

                        return hue\_oi();

                    })

            })

    })

Viết lại thành Kotexcode

funA()

    .then(funB)

    .then(funC)

    .catch(console.error.bind(console));

## **Câu 5: Phân biệt let và const, trường hợp Object thì làm thế nào?**