

Posts

Questions

Discussions









Full time: 540 Giờ Part time: 240 Giờ Create your future with code



Published Mar 14th, 11:31 AM - 4 min read TABLE OF CONTENTS

2. Phân Biết

2.1. Biến var

2.2. Biến let

2.3. Biến const

Phân biệt kiểu biến var, let, và const trong JavaScript



JavaScript

1. Giới Thiệu



Sun* Cyber Security Team







VIBLO

báo biến.

va neu moi Browser iai co cach chay Javascript knac nnau thi cac trang web knong the hoạt động trên tất cả các trình duyệt đó được, vì vậy cần có một chuẩn chung để bắt

buộc các browser phải phát triển dựa theo chuẩn đó.

SUGGESTED ORGANIZATIONS

i → Sign In/Sign up



Một trong những tính năng nổi bật của ES6 đó là sự bổ sung let và const phục vụ cho việc khai báo biến dữ liệu. Vậy tại sao lại cần tới let và const dù trước đó đã có var để khai báo biến, thì ở bài viết này chúng ta sẽ cùng tìm hiểu lý do của từng kiểu khai

ES6 (viết tắt của ECMAScript 6) là một tập hợp các kỹ thuật nâng cao của Javascript. FCMACasine da bión bói sás mbà sán sosót mais tính Chás ás, atá sosót làm tiás absida ais

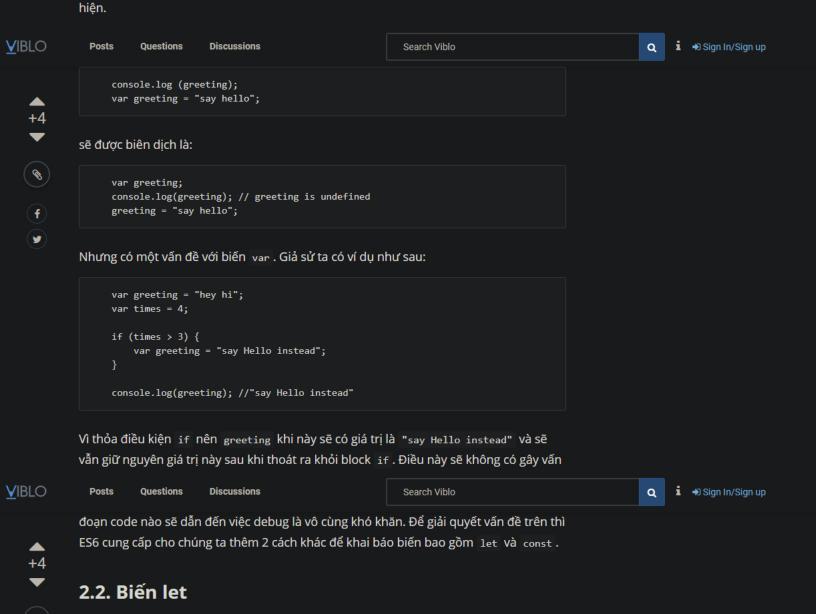
2. Phân Biệt

2.1. Biến var

các kiểu biến như number, string, boolean, etc. Trừ trường hợp được khai báo bên trong 1 function (khi đó biến var sẽ có scope là **function/locally scoped**), biến var sẽ có scope là **globally scoped**. Đặc biệt, biến var còn có thêm tính chất **hoisting**: nghĩa là dù khai báo ở đâu thì biến đều sẽ được đem lên đầu scope trước khi code được thực hiện.

Một trong những nguyên nhân khiến **let** có thể thay thế var để xử lý vấn đề nêu trên là vì biến **let** được khai báo sẽ có scope là **block scoped** chứ không phải globally hay

locally scoped.

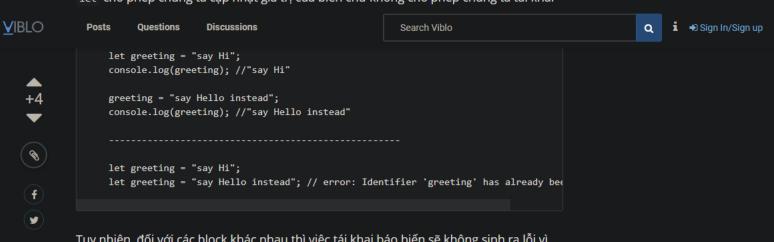


```
let greeting = "say Hi";
let times = 4;

if (times > 3) {
    let hello = "say Hello instead";
    console.log(hello); // "say Hello instead"
}
console.log(hello); // hello is not defined
```

Chúng ta có thể thấy là đối với biến có scope là block scoped nếu ra khỏi scope được khai báo thì sẽ không thể sử dụng được nữa.

let cho phép chúng ta cập nhật giá trị của biến chứ không cho phép chúng ta tái khái



■ Sign In/Sign up

Tuy nhiên, đối với các block khác nhau thì việc tái khai báo biến sẽ không sinh ra lỗi vì đối với từng scope, mỗi biến sẽ được xem xét là 1 biến riêng khác biệt.

```
let greeting = "say Hi";
if (true) {
    let greeting = "say Hello instead";
    console.log(greeting); // "say Hello instead"
}
console.log(greeting); // "say Hi"
```

Giống với var, let cũng có tính **hoisting** tuy nhiên lại khác nhau ở chỗ thay vì var được khởi tạo với giá trị là undefined thì let sẽ không có bất kỳ giá trị khởi tạo nào.

 VIBLO
 Posts
 Questions
 Discussions
 Search Viblo

y

+4 Tương tự với let cũng có scope là block scoped, và hoisting thì chúng ta có thêm 1 kiểu khai báo biến nữa là const . Trong biến const nếu trường hợp kiểu của biến là primitive (bao gồm string, number, boolean, null, và undefined) thì chúng ta sẽ không thể tái khai báo hay cập nhật giá trị mới để thay thế cho giá trị trước đó của biến. f const greeting = "say Hi"; y greeting = "say Hello instead"; // error : Assignment to constant variable. const greeting = "say Hi"; const greeting = "say Hello instead"; // error : Identifier 'greeting' has already Đối với trường hợp kiểu biến là **reference** (bao gồm object, array, và function) thì tuy không thể tái khai báo hay cập nhật giá trị của biến nhưng chúng ta vẫn có thể cập nhật giá trị cho thuộc tính của biến đó. const greeting = { message : "Hello", **VIBLO** i → Sign In/Sign up **Questions** Discussions Search Viblo greeting.message = "say Hello instead"; y console.log(greeting); // {message:"say Hello instead",number:"five"} Related Hoisting trong javascript ES6 - The Good Part (Phan 1) Giới thiệu Typescript - Sự Hoisting trong javascript - gà khác nhau giữa Typescript ... có trước hay trứng (phần 1) Nguyễn Hữu Sử Nguyễn Phúc Lương Uy Tran Pham Quang Hung 6 min read 8 min read 7 min read 2 min read **◎** 686 **§** 6 **♀** 2 **♦** 15 **◎** 1339 **§** 5 **♀** 3 **♦** 9 **●** 9392 **§** 1 **■** 0 **♦** 5 **②** 415 **§** 1 **№** 4 **♦** 3 More from Nguyen Luong Huy Giới thiêu Retrofit và một số Các cách khai báo hàm trong Interactive effects with CSS Setting up NeoVim on tính năng nổi trội clip path Ubuntu **JavaScript** Nauven Luona Huv Nauven Luona Huv Nauven Luona Huy Nauven Luona Huy

