**Giới Thiệu Về JavaScript**

# 1: Giới thiệu tổng quan về ngôn ngữ lập trình JavaScript

Ngôn ngữ lập trình Javascript được giới thiệu đầu tiên vào năm 1995. Mục đích là để đưa những chương trình vào trang web ở trình duyệt Netscape Navigator - một trình duyệt web phổ biến những năm 1990.

JavaScript được phát triển bởi [Brendan Eich](https://en.wikipedia.org/wiki/Brendan_Eich) tại Hãng truyền thông Netscape với cái tên đầu tiên là Mocha, rồi sau đó đổi tên thành LiveScript, và cuối cùng thành JavaScript. Có lẽ việc đổi tên như vậy là để giúp JavaScript được chú ý nhiều hơn. Bởi tại thời điểm này, Java đang được coi là một hiện tượng và trở nên phổ biến.

Trên thực tế, JavaScript không được phát triển dựa trên Java. Và chúng là hai ngôn ngữ hoàn toàn khác biệt. Các bạn sẽ thấy rõ hơn điều này ở phần sau của bài viết.

Javascript là một ngôn ngữ lập trình kịch bản dựa vào các đối tượng có sẵn hoặc do người dùng (programmer) tự định nghĩa, ngày nay javascript được sử dụng rất phổ biến (99% các website hiện nay) và đương nhiên là với độ phổ biến như thế thì**hầu như tất cả các trình duyệt đều hỗ trợ.**

**2: Đặc điểm chung của JavaScript**

* Là ngôn ngữ lập trình bậc cao (high-level) giống như: C/C++, Java, Python, Ruby, ... Nó rất gần với ngôn ngữ tự nhiên của con người. Trong khi ngôn ngữ lập trình bậc thấp (low-level) như: Assembly... sẽ gần với máy tính hơn.
* Là ngôn ngữ lập trình động (dynamic programming language): như Python, Ruby, Perl, ... Chúng được tối ưu hoá nhằm nâng cao hiệu suất cho lập trình viên. Trong khi ngôn ngữ lập trình tĩnh (static programming language): như C/C++, ... lại được tối ưu hoá để nâng cao hiệu suất cho phần cứng máy tính.
* Là ngôn ngữ lập trình kịch bản (scripting language): nghĩa là không cần biên dịch (compile) hay liên kết (linked) giống như ngôn ngữ lập trình biên dịch (C/C++, Java, ...) mà nó sẽ được dịch tại thời điểm chạy.
* Là ngôn ngữ [dựa trên đối tượng](https://completejavascript.com/lap-trinh-huong-doi-tuong-voi-javascript/) (object-based): tức nó gần giống như ngôn ngữ lập trình hướng đối tượng, ngoại trừ JavaScript không hỗ trợ tính kế thừa và đa hình.
* Là ngôn ngữ dựa trên nguyên mẫu (prototype-based): là một kiểu của [lập trình hướng đối tượng](https://completejavascript.com/cac-khia-canh-lap-trinh-huong-doi-tuong-trong-javascript/), trong đó các hành vi của đối tượng được sử dụng lại.

**3: Ứng Dụng Của JavaScript**

JavaScript có rất nhiều ứng dụng trên nhiều môi trường và nền tảng khác khau:

* JavaScript cùng với HTML, CSS trở thành ngôn ngữ không thể thiếu đối với website
* Ngoài ra, có rất nhiều [framework JavaScript](https://completejavascript.com/top-5-framework-javascript-moi-nhat-cho-phat-trien-web-va-app/) khác nhau phía front-end: Angular.js, Angular2, ReactJS, Vue.js, Ember.js, Meteor.js, backbone, ...
* JavaScript có thể sử dụng phía server với framework: Node.js
* Một số database sử dụng JavaScript như là kịch bản và là ngôn ngữ query: MongoDB, CouchDB
* JavaScript có thể dùng để xây dựng ứng dụng Desktop với framework: Electron. Những ứng dụng nổi tiếng có thể kể đến là: Atom, Visual Studio Code, GitKraken, Wordpress.com, ...
* Để xây dựng ứng dụng điện thoại đa nền tảng (Android, IOS) có thể dùng: React-native

**4: Ưu Điểm Và Hạn Chế Của JavaScript**

## Các ưu điểm của JavaScript

Các ưu điểm của việc sử dụng JavaScript là:

**Sự tương tác Server ít hơn**: Bạn có thể xác nhận đầu vào (input) người sử dụng trước khi gửi trang tới Server. Điều này làm tiết kiệm lưu lượng tải ở Server, nghĩa là Server của bạn tải ít hơn.

**Phản hồi ngay lập tức tới khách truy cập**: Họ không phải chờ cho một trang web tải lại để thấy xem nếu họ đã quên nhập cái gì đó.

**Khả năng tương tác tăng lên**: Bạn có thể tạo các giao diện mà phản ứng lại khi người sử dụng rê chuột qua chúng hoặc kích hoạt chúng thông qua bàn phím.

**Giao diện phong phú hơn**: Bạn có thể sử dụng JavaScript để bao gồm những mục như các thành phần Drag và Drop (DnD) và các con trượt (Slider) để cung cấp một Rich Interface (Giao diện giàu tính năng) tới site khách truy cập của bạn.

## Hạn chế của JavaScript

Chúng ta không thể đối xử JavaScript như là một ngôn ngữ chương trình chính thức (full-fledged). Nó thiếu các tính năng quan trọng sau:

* Client-side JavaScript không cho phép đọc và ghi các file, bởi vì lý do bảo mật.
* JavaScript không được sử dụng cho việc kết nối mạng các ứng dụng bởi vì không có những hỗ trợ có sẵn.
* JavaScript không có bất kỳ khả năng đa luồng hoặc đa xử lý.

Một lần nữa, JavaScript là một ngôn ngữ chương trình thông dịch, nhẹ mà cho phép bạn xây dựng khả năng tương tác trong các trang HTML tĩnh.

**5: Cách Sử Dụng JavaScript**

## **5.1 Viết mã lệnh trực tiếp vào trang web**

- Mã lệnh JavaScript có thể được viết trực tiếp vào trang web thông qua cặp thẻ <script></script>

**-** Các đoạn <script> có thể được đặt ở bất kì nơi nào trong trang web. Tuy nhiên nên đặt chúng ở vị trí cuối cùng trong phần tử <body>. Điều đó giúp cải thiện tốc độ tải giao diện của trang web và tránh một số trường hợp xảy ra lỗi thực thi ngoài mong đợi.

- Trong một trang web có thể sử dụng nhiều đoạn <script> và trong một đoạn <script> thì không giới hạn số lượng câu lệnh.

Ví dụ

Trang web dưới đây sử dụng một đoạn <script>, trong đoạn <script> có ba câu lệnh

<!DOCTYPE html>

<html>

<head>

<title>Lập Trình Web</title>

</head>

<body>

<h2>Tài liệu học HTML</h2>

<h2 id="demo1"></h2>

<h2>Tài liệu học JavaScript</h2>

<h2 id="demo2"></h2>

<h2 id="demo3"></h2>

<script>

document.getElementById("demo1").innerHTML = "Tài liệu học CSS";

document.getElementById("demo2").innerHTML = "Tài liệu học MySQL";

document.getElementById("demo3").innerHTML = "Tài liệu học PHP";

</script>

</body>

</html>

## **5.2 Viết mã lệnh vào tập tin JavaScript**

- Việc viết mã lệnh JavaScript trực tiếp vào trang web *(giống như cách 1)* sẽ khiến mã nguồn của trang web bị ***"rối nùi"***, làm cho người lập trình viên khó quan sát, quản lý cũng như cập nhật nâng cấp mã nguồn *(Điều này là dễ hiểu, bạn cứ thử nghĩ một trang web có khoảng vài trăm dòng lệnh HTML/CSS đã khó quan sát như thế nào rồi, mà còn nhét thêm cả JavaScript vào đó nữa thì sẽ ra sao !?)*

- Để khắc phục vấn đề trên, ta còn một cách sử dụng JavaScript khác chính là viết mã lệnh vào bên trong tập tin JavaScript, sau đó nhúng tập tin JavaScript vào trang web.

- Cách viết mã lệnh trong tập tin JavaScript cũng giống như cách viết mã lệnh trực tiếp vào trang web. Điểm khác nhau là khi viết mã lệnh trong tập tin JavaScript thì ta không cần đặt các câu lệnh bên trong cặp thẻ *<script></script>*.

- Ví dụ, cách viết mã lệnh trong tập tin *myscript.js* dưới đây là ĐÚNG.

document.getElementById("demo1").innerHTML = "Tài liệu học CSS";

document.getElementById("demo2").innerHTML = "Tài liệu học MySQL";

document.getElementById("demo3").innerHTML = "Tài liệu học PHP";

- Ví dụ, cách viết mã lệnh trong tập tin *myscript.js* dưới đây là SAI.

<script>

document.getElementById("demo1").innerHTML = "Tài liệu học CSS";

document.getElementById("demo2").innerHTML = "Tài liệu học MySQL";

document.getElementById("demo3").innerHTML = "Tài liệu học PHP";

</script>

- Để nhúng tập tin JavaScript vào trang web thì ta chỉ cần thêm vào trang web cặp thẻ *<script>* với cú pháp như sau:

<script src="đường dẫn đến tập tin JavaScript"></script>

* **Kết luận**

- Giữa việc viết mã lệnh trong tập tin JavaScript và viết trực tiếp vào trang web thì chúng ta nên dùng cách viết mã lệnh trong tập tin JavaScript. Bởi vì đối với những chương trình lớn khoảng vài trăm hoặc vài nghìn dòng, việc viết vào tập tin JavaScript sẽ giúp ta dễ quan sát, quản lý cũng như cập nhật nâng cấp mã nguồn hơn.

- Tuy nhiên, những ví dụ trong loạt bài hướng dẫn học JavaScript này đa số khoảng từ 20 dòng lệnh đổ lại nên chúng tôi sử dụng cách viết mã lệnh JavaScript trực tiếp vào trang web để bạn dễ quan sát tổng thể hơn.

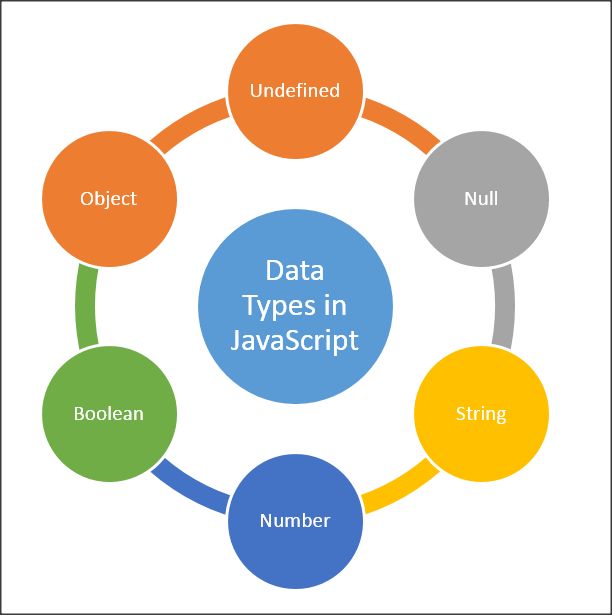
**6: Các Thành Phần Cú Pháp Chính Của JavaScript**

## **6.1 Cấu tạo của JavaScript**

Javascript được cấu tạo từ các dòng mã kết thúc bởi dấu “ ; “. Các dòng mã sẽ kết hợp với nhau theo một mối liên quan nhất định để tạo thành một khối lệnh xử lý một vấn đề nào đó.

Như các ngôn ngữ khác, Javascript cũng cần xử lý dữ liệu. Tuy nhiên các kiểu dữ liệu của Javascript là không nhiều và không được phân loại chi tiết như các ngôn ngữ lập trình khác. Javascript chỉ có 3 loại kiểu dữ liệu sau:

* **Chuỗi**: Khi một đoạn text được đặt trong dấu nháy đơn: ' ... 'hoặc dấu nháy kép " ... "thì đó được xem là một chuỗi.
* **Số**: Không phân chi tiết như các ngôn ngữ khác, Javascript đưa các số nguyên, thập phân, .. vào một kiểu số duy nhất. Các số không được đặt trong dấu nháy như chuỗi. Ví dụ như: 4, 8, 1516, 23.42.
* **Boolean**: Đây là kiểu dữ liệu chứa 2 giá trị true, hoặc false.

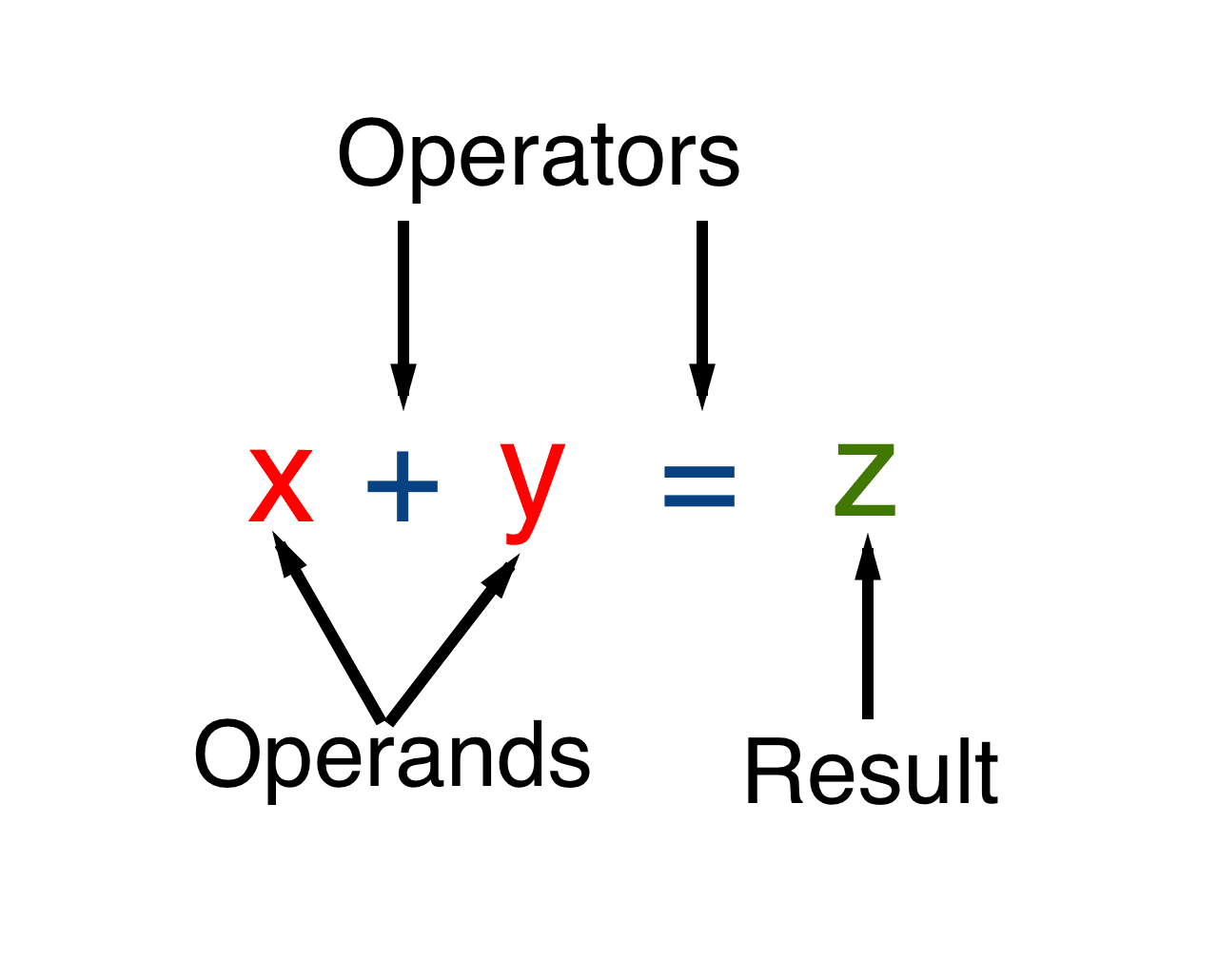


3 kiểu dữ liệu chính trong JS là chuỗi, số và boolean. Ngoài ra còn có kiểu trống (null), chưa định nghĩa (undefined) và đối tượng (Object) sẽ được nói đến trong các phần sau.

Để lưu trữ một giá trị và sử dụng nó vào các khối lệnh để xử lý bạn cần một biến lưu. Biến trong Javascript có cấu trúc như sau: var tên\_biến = giá\_trị\_biến. Giá trị của biến là kiểu dữ liệu gì thì biến đó có kiểu dữ liệu đó. Ví dụ var name = 'Tuấn'; Javascript là một ngôn ngữ chặc chẽ, nó phân biệt chữ HOA và chữ THƯỜNG nên bạn phải chú ý khi đặt tên biến và gọi biến. Đặt tên biến bạn không được phép sử dụng dấu nối như “tên-biến” mà chỉ được phép đặt liền như “TênBiến” hoặc “tên\_biến“. Tuy nhiên bạn không được sử dụng tên biến có dấu nhé.

Biến là một vùng nhớ được tạo ra để lưu trữ giá trị tạm thời. Biến có kiểu dữ liệu phụ thuộc vào kiểu dữ liệu của giá trị mà biến lưu trữ.

Các toán tử số học trong Javascript bao gồm: Cộng (+), trừ (-), nhân (\*), chia (/), chia lấy dư (%), tăng lên 1 đơn vị (++), giảm đi một đơn vị (--)… ví dụ (2 + 3) \* 4.



Javascript có đầy đủ các toán tử hỗ trợ tính toán và kết hợp

Các toán tử gán trong Javascript bao gồm: = (giá trị bên trái bằng bên phải), +=  (giá trị bên trái bằng chính nó cộng với giá trị bên phải vd: x+=y sẽ bằng x=x+y), -= (giá trị bên trái bằng chính nó trừ đi giá trị bên phải vd: x-=y sẽ bằng x=x-y), \*= (Giá trị bên trái bằng chính nó nhân cho giá trị bên phải vd x\*=y sẽ bằng x=x\*y), /= (giá trị bên trái bằng chính nó chia cho giá trị bên phải vd: x/=y sẽ bằng x=x/y), %= (giá trị bên trái bằng chính nó chia lấy dư cho giá trị bên phải vd x%=y sẽ bằng x=x%y).

Comment trong Javascript sử dụng dấu // trước comment hoặc trong cặp /\*...\*/ ví dụ như /\*..Nội dung comment..\*/

**6.2 Hàm trong JavaScript**

Trong ngôn ngữ lập trình, hàm là một trong những thành phần không thể thiếu và thường xuyên được sử dụng. Hàm được hiểu là một nhóm lệnh được gom lại để thực hiện một công việc cụ thể nào đó khi được gọi đến. Trong bài viết này mình sẽ chia sẻ chi tiết về Hàm và cách sử dụng hàm trong [Javascript](https://tuandc.com/tag/javascript).

## **Cú pháp của hàm trong Javascript**

* Một hàm được định nghĩa bằng từ khóa function, sau đó là tên hàm, tiếp theo là dấu ngoặc đơn (). Tên hàm có thể chứa chữ cái hoặc số, dấu gạch dưới,.. (tương tự quy tắt đặt tên biến). Bên trong dấu ngoặc đơn () có thể khởi tạo các tham số để nhận giá trị đầu vào. Cuối cùng là các khối lệnh sẽ được đặt bên trong dấu ngoặc nhọn {}. Cú pháp cụ thể như sau:

function tên\_hàm(Tham\_số\_1, Tham\_số\_2, Tham\_số\_3,...) {

    Khối mã lệnh Javascript

}

* Các tham số được xem là các biến cực bộ. Có thể là giá trị truyền sẵn hoặc là giá trị được truyền khi gọi hàm.

Trả về giá trị là một chức năng của hàm, khi thực hiện trả về giá trị hàm sẽ ngừng hoạt động. Để trả về giá trị trong hàm sử dụng khóa return. Ví dụ đoạn mã sau:

function fn\_return(a, b) {

    return a \* b;

}

Khi gọi hàm trên, giá trị a\*b sẽ được trả về. Để gọi hàm bạn chỉ cần cú pháp: tên\_hàm (giá\_trị\_tham\_số);. Ví dụ, để gọi hàm trên chỉ cần gọi fn\_return(4,3);. Lời gọi hàm này có thể gán cho một biến để lưu giá trị như var f = fn\_return(4,3); giá trị của biến f sẽ bằng giá trị trả về của hàm fn\_return(4,3).

## **Các cách sử dụng hàm trong Javascript**

Hàm được viết trong tệp \*.js hoặc cặp thẻ <script></script>, Các hàm có thể được gọi lẫn nhau mà không gây ra xung đột. Ví dụ như:

function fn\_return(a, b) {

return a \* b;

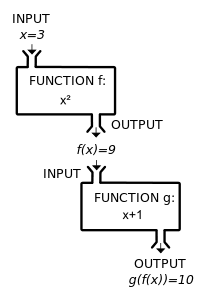
}

function fn\_return2() {

return 12 \* fn\_return(4, 3);

}

Var resuft = fn\_return2();



Hàm thường có đầu vào (input) và đầu ra (output), đầu ra của hàm này có thể được sử dụng để làm đầu vào của hàm khác.

* Khi bạn gọi hàm mà không có dấu ngoặc đơn () thì giá trị trả về sẽ là định nghĩa của hàm (cấu trúc của hàm sẽ được trả về thay vì kết quả hàm xử lý).

**6.3 Đối tượng trong JavaScript**

**Đối tượng (object)** là một biến trong ngôn ngữ lập trình, dùng để lưu trữ nhiều thuộc tính và giá trị khác nhau. Đối tượng được dùng để mô tả một thực thể trong thực tế, đôi lúc cũng được sử dụng để mô tả những thứ phi thực tế. Trong bài viết này mình sẽ chia sẻ về đối tượng và cách sử dụng những đối tượng trong [Javascript](https://tuandc.com/tag/javascript).

## **Cú pháp đối tượng trong Javascript**

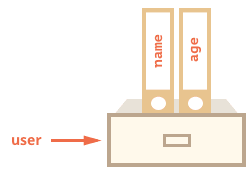
Một chiếc xe hơi được xem là một đối tượng. Nó bao gồm nhiều thuộc tính khác nhau. Mỗi thuộc tính của nó có một giá trị nhất định. Ví dụ:

Mình có một chiếc xe, Loại: sedan, Hãng: Toyota, Đời: 2016, Màu: trắng. Chiếc xe của mình là một đối tượng, và các thuộc tính như loại, hãng, đời, màu là thuộc tính của chiếc xe, và sedan, Toyota, 2016, trắng là giá trị thuộc tính của chiếc xe này. Cú pháp đối tượng của Javascript là:

**var ten\_doi\_tuong = {thuoc\_tinh1:gia\_tri1, thuoc\_tinh2:gia\_tri2,..};**

Như vậy biến đối tượng của chiếc xe của mình sẽ là:

var mycar = {loai:"sedan",hang:"Toyota",doi:"2016",mau:"trắng"};



Đối tượng là một thực thể bao gồm các thuộc tính và giá trị của nó.

## **Cách sử dụng đối tượng trong Javascript**

Khoảng cách các thuộc tính trong biến đối tượng là không quan trọng, thế nên bạn có thể viết trên 1 dòng hoặc xuống dòng nếu thích. Mình khuyến khích bạn nên sử dụng theo cách sau để nhìn code rõ ràng hơn.

var mycar = {

loai:"sedan",

hang:"Toyota",

doi:"2016",

mau:"trắng"

};

Để truy vấn các thuộc tính trong đối tượng bạn chỉ cần thực hiện cú pháp **tên\_đối\_tượng.tên\_thuộc\_tính** hoặc **tên\_đối\_tượng["tên\_thuộc\_tính"]** ví dụ mycar.loai; hoặc mycar["loai"]; kết quả trả về là giá trị của thuộc tính loai.

Một giá trị trong thuộc tính có thể là một hàm. Ví dụ:

var mycar = {

loai:"sedan",

hang:"Toyota",

doi:"2016",

mau:"trắng",

value: function(){return "$10000";}

};

Để gọi giá trị của thuộc tính value, bạn gọi như sau: mycar.value();

Khi bạn tạo biến chuỗi, số hoặc boolean, nếu bạn thêm new trước loại biến, biến của bạn sẽ trở thành một đối tượng. Ví dụ var x = new String(); Thông thường nếu bạn không xử lý phức tạp bạn không nên tạo ra các đối tượng này, nó sẽ làm chậm đi việc xử lý mã.