TẬP ĐOÀN CÔNG NGHIỆM VIỄN THÔNG QUÂN ĐỘI VIETTEL

**CÔNG TY ĐẦU TƯ CÔNG NGHỆ VIETTEL**



**BÁO CÁO THỰC TẬP**

**FRONT-END**

**Lý thuyết tuần 3**

**NGƯỜI HƯỚNG DẪN: MAI DUY HIẾU**

**THỰC TẬP SINH: ĐÀM THI LAN ANH**

**PHÒNG BAN: KỸ THUẬT CÔNG NGHỆ**

**Hà Nội- 2025**

**MỤC LỤC**

[PHẦN 1. JavaScript cơ bản 1](#_Toc201591380)

[1.1. Biến 1](#_Toc201591381)

[1.2. Kiểu dữ liệu 2](#_Toc201591382)

[1.2.1. Kiểu logic (boolen) 2](#_Toc201591383)

[1.2.2. Kiểu Null (rỗng) 2](#_Toc201591384)

[1.2.3. Kiểu Undefined - chưa định nghĩa 3](#_Toc201591385)

[1.2.4. Kiểu Number (số) 3](#_Toc201591386)

[1.2.5. Kiểu chuỗi kỹ tự javascript 3](#_Toc201591387)

[1.2.6. Kiểu Symbol 4](#_Toc201591388)

[1.3. Câu điều kiện 4](#_Toc201591389)

[1.3.1. If- else 4](#_Toc201591390)

[1.3.2. Khối lệnh else if 6](#_Toc201591391)

[1.3.3. Lệnh switch 6](#_Toc201591392)

[1.4. Vòng lặp for, while, do-while 8](#_Toc201591393)

[1.4.1. Vòng lặp for 8](#_Toc201591394)

[1.4.2. Vòng lặp while 8](#_Toc201591395)

[1.4.3. Vòng lặp do-while 9](#_Toc201591396)

[1.4.4. Lệnh continute và break 9](#_Toc201591397)

[1.5. Hàm function, arrow function 10](#_Toc201591398)

[1.5.1. Định nghĩa hàm 10](#_Toc201591399)

[1.5.2. Gọi hàm 10](#_Toc201591400)

[1.5.3. Tham số hàm 10](#_Toc201591401)

[1.5.4. Giá trị trả về của hàm 11](#_Toc201591402)

[1.5.5. Biểu thức hàm 12](#_Toc201591403)

[1.5.6. Hàm ẩn danh anonymous 12](#_Toc201591404)

[1.5.7. Biểu thức hàm chạy ngay lập tức IIFE 13](#_Toc201591405)

[1.5.8. Arrow function 13](#_Toc201591406)

[1.5.9. So sánh hàm thông thường và arrow function 13](#_Toc201591407)

[1.6. Mảng array 14](#_Toc201591408)

[1.6.1. Khai báo mảng: 14](#_Toc201591409)

[1.6.2. Thao tác với mảng 14](#_Toc201591410)

[1.6.3. Một số thuộc tính và phương thức 15](#_Toc201591411)

[1.7. Đối tượng (Object) 16](#_Toc201591412)

[1.7.1. Truy cập đến thuộc tính đối tượng 16](#_Toc201591413)

[1.7.2. Khởi tạo đối tượng 17](#_Toc201591414)

[1.7.3. Phương thức trong đối tượng 17](#_Toc201591415)

[1.7.4. Seter và Getter 19](#_Toc201591416)

# JavaScript cơ bản

#### JavaScript là ngôn ngữ lập trình mạnh mẽ cho web. Không giống như HTML và CSS JavaScript thường được sử dụng để viết logic thay vì đánh dấu.

## Biến

#### Một trong những khái niệm quan trọng nhất trong lập trình là biến. Biến trỏ đến một địa chỉ bộ nhớ cụ thể lưu trữ giá trị. Biến được đặt tên có thể được sử dụng trong toàn bộ mã của bạn để truy cập giá trị đó.

###### Biến toàn cục: biến được khai báo bên ngoài tất cả các hàm, nên nó có thể được truy cập và sử dụng ở bất kỳ đâu trong mã nguồn.

###### Biến cục bộ: biến được khai báo bên trong một hàm hoặc block, nên chỉ có thể được truy cập trong phạm vi đó.

#### Khai báo biến có nghĩa là đặt tên cho biến. Trong JavaScript, điều này thường được thực hiện bằng từ khóa let và var. Ví dụ:





#### Ngoài ra còn có kiểu khai báo biến const dùng để khai báo hằng số, không thể gán lại giá trị mới cho biến đó. Tuy nhiên, nếu const dùng với object hoặc array, ta vẫn có thể thay đổi nội dung bên trong, chỉ không được gán lại biến.

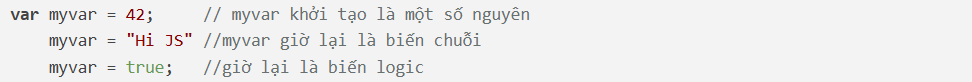


#### Việc đặt tên biến tuân theo các quy tắc cụ thể: tên có thể bao gồm các chữ cái, số, dấu đô la và dấu gạch dưới, nhưng không được chứa khoảng trắng và không được bắt đầu bằng số. Khi tên biến có nhiều hơn một từ, có những quy ước đặt tên cụ thể về cách bạn viết hoa các từ. Trong JavaScript, quy ước sử dụng là camel case.

## Kiểu dữ liệu

#### Kiểu dữ liệu ám chỉ đến kiểu của giá trị mà chương trình làm việc. Trong JavaScript có thể giữ nhiều kiểu dữ liệu như các số, chuỗi, mảng ...

#### Các biến trong Javascript khai báo một cách lỏng lẻo, có nghĩa là một biến không có định gắn với một kiểu dữ liệu nào đó (khác với C). Kiểu của biến là kiểu mà giá trị biến đó đang chứa (như một biến khai báo và gán luôn cho nó một số thì biến đó kiểu số, sau đó gán cho nó một chuỗi thì lúc đó biến có kiểu chuỗi). Hãy xem ví dụ sau:



#### Js chia các kiểu dữ liệu ra hai nhóm là kiểu dữ liệu nguyên thủy và đối tượng:

###### Kiểu dữ liệu nguyên thủy là những loại dữ liệu mà giá trị của nó nó không thể sửa đổi trong biến (tất nhiên nó có thể thay thế). Ví dụ chuỗi là một kiểu nguyên thủy, một biến chuỗi var a = "abc", giờ nếu muốn a chứa "ab" thì bạn chỉ có thể thay thế giá trị biến đó đang giữ bằng một giá trị mới "ab" chứ không thể xóa đi ký tự "c" từ chuỗi giá trị cũ để nó thành "ab". Có tất cả 6 kiểu dữ liệu nguyên thủy: boolean, null, null, undefined, number, string, symbol

###### Kiểu đối tượng, giá trị biến giữ là một đối tượng nào đó. Đối tượng đó có thể thay đổi bằng cách thay đổi các thuộc tính của nó (không cần thay thế đối tượng mới).

### Kiểu logic (boolen)

#### Biểu diễn logic đúng / sai - có 2 giá trị là true và false

|  |
| --- |
| var found = false; |

Note: Giá trị logic của 0, null, undefined, chuỗi rỗng là false. Các giá trị khác là true

### Kiểu Null (rỗng)

#### Biến có giá trị null thì nó không chứa đối tượng nào cả (rỗng)

|  |
| --- |
| var object = null; |

### Kiểu Undefined - chưa định nghĩa

#### Biến có giá trị undefined là giá trị khi biến đó được khởi tạo (chưa bị gán)

Ví dụ:



Trả về:

A dark blue background with white lines

AI-generated content may be incorrect.

### Kiểu Number (số)

#### Khác với các ngôn ngữ lập trình khác, JavaScript không định nghĩa sự khác nhau giữa các loại số ví dụ phông phân biệt số nguyên, số thực ...

#### Số trong JavaScript có thể viết có hoặc không có phần thập phân

|  |
| --- |
| var num = 42; // Một số không có phần thập phân |

#### Ví dụ sau tạo ra số có phần thập phân:

|  |
| --- |
| var price = 55.55;  document.write(price); |

### Kiểu chuỗi kỹ tự javascript

#### Chuỗi ký tự trong JavaScript có thể được lưu giữ trong biến và biểu diễn trong cặp nháy đơn '' hoặc nháy kép ""

|  |
| --- |
| var name = 'Lan Anh'; //Chuỗi trong dấu nháy đơn  var text = "Tên tôi là Đàm Lan Anh"; //Chuỗi trong dấu nháy kép |

#### Để chèn các ký tự đặc biệt như nháy đơn ' nháy kép ", ký hiệu xuống dòng, tab, \ thì sử dụng ký hiệu escape \ để biểu diễn:

|  |  |
| --- | --- |
| Mã | Ký tự biểu diễn |
| \' | ' |
| \" | " |
| \\ | \ |
| \n | Xuống dòng |
| \r | Đầu dòng |
| \t | tab |

### Kiểu Symbol

#### Kiểu này mỗi giá trị là duy nhất và không thể bị sửa đổi. Giá trị Symbol được tạo ra từ hàm Symbol(), vì nó luôn tạo ra một giá trị duy nhất trên toàn hệ thống nên kiểu này thường được dùng để định danh các đối tượng cho mục đích tìm kiếm lại.

|  |
| --- |
| var sym1 = Symbol();  var sym2 = Symbol(); //sym2 luôn khác sym1 |

## Câu điều kiện

### If- else

#### Lệnh if : nếu điều kiện là đúng (true) thì thi hành các lệnh trong khối - nếu điều khiện sai false thì khối lệnh sau nó bị bỏ qua, cú pháp là:

|  |
| --- |
| **if** (điều\_kiện) {  //Các dòng lệnh trong khối  } |

Ví dụ:

|  |
| --- |
| **var** myNum1 = 7;  **var** myNum2 = 10;  **if** (myNum1 < myNum2) {  **alert**("JavaScript rất dễ học.");  } |

Thi hành đoạn mã trên sẽ hiện thị hộp thoại thông báo: JavaScript rất dễ học. vì điều kiện là true nên lệnh alert thi hành

#### Lệnh else, nếu false thì thi hành khối lệnh else

|  |
| --- |
| **if** (expression) {  // thi hành các lênh ở đây (khối if) nếu true  }  **else** {  // thi hành các lệnh (khối else) nếu điều kiện false  } |

Ví dụ:

|  |
| --- |
| **var** myNum1 = 7;  **var** myNum2 = 10;  **if** (myNum1 > myNum2) {  **alert**("Đây là khối lệnh 1");  }  **else** {  **alert**("Đây là khối lệnh 2");  } |

Chạy code trên hiện thị thông báo: Đây là khối lệnh 2 vì myNum1 > myNum2 là false

### Khối lệnh else if

#### else if sẽ tạo ra câu lệnh điều kiện if mới nếu điều kiện trước đó false

|  |
| --- |
| **var** course = 1;**if** (course == 1) {document.**write**("HTML Tutorial");} **else** **if** (course == 2) {document.**write**("CSS Tutorial");} **else** {document.**write**("JavaScript Tutorial");} |

#### Chạy code trên, trình duyệt sẽ hiện thị HTML Tutorial, nếu bạn gán biến course thì hiện thị CSS Tutorial, nếu gán course khác với 1 và 2 thì sẽ hiện thị JavaScript Tutorial

### Lệnh switch

Trong trường hợp bạn có rẽ nhánh (nhiều điều kiện) khác nhau thay vì sử dụng nhiều else if hãy dùng switch với cú pháp

|  |
| --- |
| **switch** (expression) {  **case** n1:  //.. thi hành nếu expression bằng n1  **break**;  **case** n2:  //.. thi hành nếu expression bằng n2  **break**;  default:  //.. mặc định thi hành nếu expression không bằng giá trị nào ở trên  } |

Ví dụ:

|  |
| --- |
| **var** day = 2;  **switch** (day) {  **case** 1:  document.**write**("Monday");  **break**;  **case** 2:  document.**write**("Tuesday");  **break**;  **case** 3:  document.**write**("Wednesday");  **break**;  default:  document.**write**("Another day");  }  // Outputs "Tuesday" |

#### Lưu ý từ khóa break; để điều hướng ra khỏi khối. Thường bạn cần có break; ở cuối mỗi khối của lệnh switch, nếu thiếu break thì sẽ không thoát lệnh mà sẽ thi hành khối tiếp theo. Ở ví dụ trên, nếu bỏ đi break, thì script sẽ chạy từ điểm rẽ nhánh case 2: cho đến cuối (xuất ra: Tuesday, Wednesday, Another day)

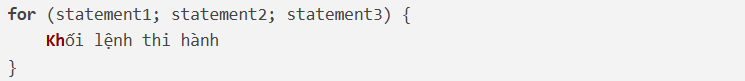
#### Từ khóa default: định nghĩa khối mặc định, khối này thi hành nếu tất cả các điều kiện rẽ nhánh không thỏa mãn.

## Vòng lặp for, while, do-while

#### Vòng lặp (loop) trong JavaScript cho phép bạn lặp lại một đoạn mã nhiều lần, giúp xử lý dữ liệu lặp một cách hiệu quả.

### Vòng lặp for

#### Vòng lặp xác định rõ số lần lặp, cú pháp:



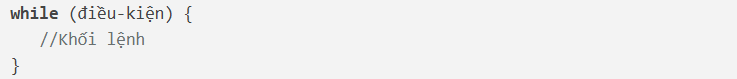
##### statement1 : lệnh thi hành trước khi vòng lặp for bắt đầu

##### statement2 : điều kiện kiểm tra trước mỗi lần thi hành khối lệnh for (true thì khối lệnh sẽ thi hành, false sẽ khối for sẽ không thi hành - thoát lặp)

##### statement3 : thi hành sau mỗi lần một vòng hoàn thành

### Vòng lặp while

#### Vòng lặp khi điều kiện còn true, cú pháp:



#### Đầu tiên nó kiểm tra điều kiện, nếu true sẽ thi hành khối lệnh. Đến cuỗi khối lại kiểm tra điều kiện, nếu điều kiện vẫn là true thì lại tiếp tục thì hành vòng mới của khối lệnh.

### Vòng lặp do-while

#### Giống với vòng lặp while nhưng khối lệnh thi hành luôn mà không kiểm tra điều kiện trước, khi khối lệnh thi hành xong mới kiểm tra điều kiện để xem có lặp lại hay không, cú pháp:

A white background with black and red lines

AI-generated content may be incorrect.

Note: Vòng lặp **do ... while** khối lệnh luôn được thi hành ít nhất một lần

### Lệnh continute và break

#### Trong vòng lặp khi gặp continue; nó sẽ bỏ qua các lệnh còn lại và khởi tạo vòng lặp mới luôn. Còn nếu gặp break; thì bỏ qua các lệnh còn lại đồng thời thoát khỏi vòng lặp.

A screenshot of a computer

AI-generated content may be incorrect.

|  |
| --- |
| //In ra các số: (bỏ qua 5)  1  2  3  4  6  7 |

#### Lệnh continue còn dùng để nhảy đến một khối lệnh có nhãn bằng cú pháp:



#### Lệnh break còn dùng để hủy thi hành khối lệnh bên ngoài có nhãn, với cú pháp:

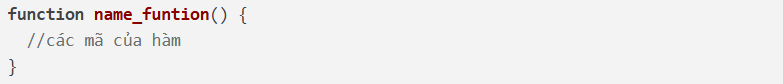


## Hàm function, arrow function

#### Hàm là một khối mã lệnh được viết nhằm một múc đích nào đó. Xây dựng hàm mang lại một số lợi ích như sử dụng lại mã đã viết, một khối mã lệnh với các tham số khác nhau mang lại các kết quả khác nhau. Một hàm thi hành khi hàm đó được gọi.

### Định nghĩa hàm

#### Hàm trong JavaScript sử dụng từ khóa function tiếp theo là tên hàm và các tham số nếu có trong ngoặc (). Khối mã của hàm nằm trong khối ngoặc nhọn {}



Note: Tên hàm có thể chứa các ký tự, số, gạch dưới ... (tương tự quy tắc tên biến).

### Gọi hàm

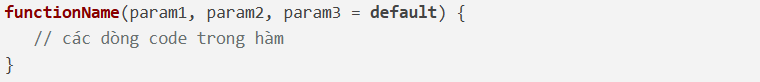
#### Để thi hành hàm bạn cần gọi nó. Để gọi hàm viết lại tên hàm và các tham số truyền vào hàm, nhớ kết thúc bằng dấu ; Một hàm có thể được gọi bao nhiêu lần là tùy bạn. Ví dụ:

A white background with black and red lines

AI-generated content may be incorrect.

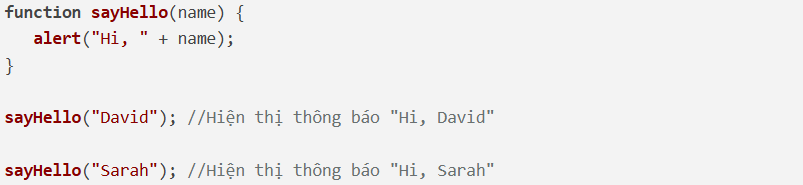
### Tham số hàm

#### Các tham số của hàm bạn cần liệt kê sau tên hàm, mỗi tham số cách nhau bởi dấu ,



#### Tham số nào muốn gán giá trị mặc định (khi gọi hàm mà không chỉ ra tham số đó thì tự động lấy mặc định) thì gán ngay ở khái báo hàm, ở trên param3 = default

###### Khi gọi hàm có tham số, bạn cần truyền tham số là giá trị thực tế để hàm thi hành



#### Hàm có nhiều tham số được sử dụng một cách tương tự, nhớ là khi gọi hàm các tham số cần truyền đúng theo tứ tự qua tên hàm

A close-up of a text

AI-generated content may be incorrect.

Note: Nếu hàm được gọi mà truyền thiếu tham số, thì tham giá trị thiếu đó của tham số được thiết lập là undefined nếu không có khai báo giá trị mặc định, còn ngược lại sẽ lấy giá trị mặc định

### Giá trị trả về của hàm

#### Hàm có thể tùy chọn có giá trị trả về hay không. Nếu có giá trị trả về hàm sử dụng lệnh return

#### Khối lệnh của hàm mà gặp đoạn lệnh return hàm sẽ dừng thi hành và trả về biểu thức của giá trị của return

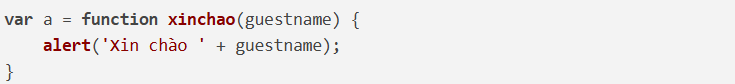
A white background with black text

AI-generated content may be incorrect.

Note: Hàm không có return một giá trị cụ thể nào, thì giá trị trả về của hàm là undefined

### Biểu thức hàm

#### Biểu thức hàm (Function Expression) là cách định nghĩa một hàm bằng cách gán nó cho một biến.



#### Có biểu thức hàm rồi (biểu thức hàm này có tên xinchao), giá trị của biểu thức này đã gán vào biến a nếu muốn gọi biểu thức ta dùng biến a tương tự như tên hàm để thực hiện biểu thức.



#### Ta để ý gọi biểu thức hàm ta dùng đến số hạng biến a chứ không thể dùng tên xinchao. Nếu ta gọi sẽ có lỗi:



#### Như vậy, tên này khai nhưng không thể dùng nó để gọi biểu thức, tên này xuất hiện trong bộ nhớ call stack giúp cho việc debug, giám sát ...Ví dụ bạn xem a là gì?



### Hàm ẩn danh anonymous

#### Chính là biểu thức hàm ở trên, nhưng trong phần khai báo bỏ tên đi (không tên). Ở ví dụ trên biểu thức hàm bỏ đi tên xinchao thì lúc này nó trở thành hàm ẩn danh, cách gọi biểu thức vẫn tương tự

A white background with black text

AI-generated content may be incorrect.

### Biểu thức hàm chạy ngay lập tức IIFE

#### Trong nhiều trường hợp, bạn khai báo biểu thức hàm rồi dùng chỉ một lần. Trong những trường hợp như vậy, bạn có thể áp dụng mô hình lập trình có tên IIFE (biểu thức hàm chạy luôn - Immediately Invokable Function Expression), để code sáng sửa và ngắn gọn hơn (nếu không thích thì cứ việc khai báo rồi gọi biểu thức cũng không sao). Ví dụ biểu thức sau sẽ chạy luôn!

A white background with black and white clouds

AI-generated content may be incorrect.

### Arrow function

#### **ECMAScript 6** đưa ra thêm cách định nghĩa biểu thức hàm một cách ngắn gọn, dùng kỹ hiệu mũi tên =>

#### Cú pháp cơ bản như sau:

A white background with black and red text

AI-generated content may be incorrect.

Ví dụ:

A screen shot of a computer

AI-generated content may be incorrect.

### So sánh hàm thông thường và arrow function

|  |  |  |
| --- | --- | --- |
| Đặc điểm | Arrow Function | Function Expression / Declaration |
| Cú pháp ngắn gọn | Yes | No (dài hơn) |
| Có this riêng | No (Kế thừa this từ bên ngoài) | Yes (Có this riêng) |
| Dùng làm phương thức đối tượng | No (Không nên dùng) | Yes (Dùng tốt) |
| Dùng tốt cho hàm callback | Yes (Rất tốt) | Yes |

## Mảng array

#### Trong lập trình, bạn thường phải làm việc với nhiều dữ liệu. Có nhiều cấu trúc dữ liệu có thể giúp bạn sắp xếp và quản lý dữ liệu của mình. Một trong những cấu trúc dữ liệu cơ bản nhất là mảng.

#### Mảng là kiểu dữ liệu không nguyên thủy có thể chứa một chuỗi giá trị. Kiểu dữ liệu không nguyên thủy khác với kiểu dữ liệu nguyên thủy ở chỗ chúng có thể chứa dữ liệu phức tạp hơn. Kiểu dữ liệu nguyên thủy như chuỗi và số chỉ có thể chứa một giá trị tại một thời điểm.

### Khai báo mảng:

#### Mảng được biểu thị bằng dấu ngoặc vuông [ ]. Dưới đây là một ví dụ về một biến với giá trị của một mảng trống:



#### Mảng được biểu thị bằng toán tử new.



#### Tạo mảng bằng cách chỉ định số lượng phần tử

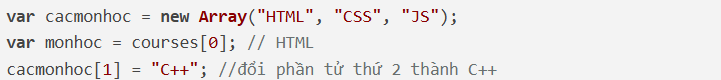
###### Ở cách tạo này, các phần tử không được gán giá trị ngay, mảng sẽ tạo ra số lượng phần tử yêu cầu sau đó giá trị từng phần tử được gán sau.

A white background with black and red lines

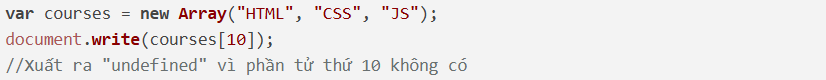
AI-generated content may be incorrect.

### Thao tác với mảng

#### Mỗi phần từ có một chỉ số trong mảng, phần tử thứ nhất chỉ số là 0. Để truy cập tới phần tử mảng dùng ký hiệu tên-mảng[chỉ-số-phần-tử]



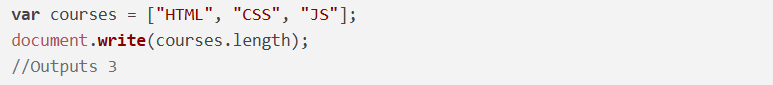
#### Khi cố gắng truy cập phần tử mảng, mà phần tử đó không tồn tại thì nó trả về giá trị bằng undefined



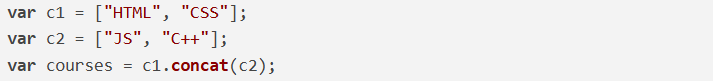
### Một số thuộc tính và phương thức

#### Một số thuộc tính và phương thức định nghĩa sẵn cho các đối tượng mảng

###### Số lượng phần tử trong mảng length



###### Kết hợp 2 mảng concat



###### Duyệt qua các phần tử mảng bằng forEach

A white screen with black text

AI-generated content may be incorrect.

###### Thêm 1 phần tử vào cuối mảng push



###### Thêm một phần tử vào đầu bằng unshift



###### Xóa phần tử cuối của mảng bằng pop



###### Xóa phần tử đầu của mảng bằng shift



###### Xóa các phần tử theo chỉ số splice



###### Nhân bản một mảng với slice



## Đối tượng (Object)

#### Tạo và sử dụng các đối tượng trong JavaScript gần giống với các khái niệm trong lập trình hướng đối tượng, như thuộc tính, phương thức.

#### Các biến trong JavaScript chứa các dữ liệu, các đối tượng cũng tương tự như vậy, nhưng nó chứa được nhiều giá trị. Hãy nghĩ một đối tượng là một danh sách các giá trị được viết theo cặp nameobject:value

#### Ví dụ sau tạo ra đối tượng lưu trong biến person :

A white background with black and red text

AI-generated content may be incorrect.

Các thuộc tính của đối tượng trên là name, age, favColor, height ...

### Truy cập đến thuộc tính đối tượng

#### Có 2 cách để truy cập đến thuộc tính đối tượng để lấy giá trị thuộc tính hoặc gán giá trị vào thuộc tính, ví dụ:



A close-up of a white background

AI-generated content may be incorrect.

### Khởi tạo đối tượng

###### Cách 1: Khởi tạo cố định

#### Cách thứ nhất là tạo bằng phương pháp cố định, là cách đã thực hiện ở trên (tạo đối tường và khởi tạo luôn các thuộc tính cần có )

A white background with black and white clouds

AI-generated content may be incorrect.

###### Cách 2: Khởi tạo bằng hàm tạo

#### Với cách thứ 2 này bạn khai báo một hàm gọi là hàm tạo rồi tạo ra đối tượng bằng cú pháp new hamtao( )

A computer screen shot of a computer code

AI-generated content may be incorrect.

Note: Trong hàm tạo hoặc các hàm thuộc đối tượng, sử dụng từ khóa this để tham khảo đến đối tượng, thông qua nó truy cập các thuộc tính

### Phương thức trong đối tượng

#### Một đối tượng ngoài các thuộc tính ra nó còn chứa hàm gọi là phương thức, ví dụ truy cập một hàm



#### Như đã biết khi viết chuỗi có sử dụng đến document.write() thì write chính là phương thức của đối tượng document

#### Phương thức thuộc về một đối tượng, đối tượng này được hàm tham khảo qua từ khóa this

#### Thường các phương thức được định nghĩa qua một hàm khởi tạo đối tượng.

A computer code on a white background

AI-generated content may be incorrect.

#### Các phương thức cũng có thể định nghĩa bên ngoài hàm khởi tạo, ví dụ:

A white screen with black text

AI-generated content may be incorrect.

### Seter và Getter

#### Một thuộc tính của của đối tượng còn thiết lập nó là hàm setter hoặc getter, nếu là setter nó chỉ được gọi qua toán tử gán giá trị cho nó, nếu là getter thì chỉ được gọi khi truy cập lấy giá trị thuộc tính.

#### Hàm setter định nghĩa bằng cách cho thêm set, hàm getter định nghĩa bằng cách cho thêm get

A screenshot of a computer screen

AI-generated content may be incorrect.

###### Tạo setter và getter trong hàm tạo

#### Trong trường hợp muốn định nghĩa setter/getter trong hàm tạo đối tượng thì bạn cần định nghĩa theo nguyên tắc thêm một thuộc tính vào đối tượng đã có với lệnh Object.defineProperty

A screenshot of a computer

AI-generated content may be incorrect.

#### Ví dụ trên định nghĩa lại trong hàm tạo:

A screen shot of a computer code

AI-generated content may be incorrect.