[Bài đọc] Căn bản về đối tượng trong JavaScript

Một đối tượng là một tập hợp có liên quan với nhau giữa dữ liệu hoặc các chức năng (việc này thường được thể hiện với các biến và hàm - chúng được gọi là các thuộc tính và phương thức khi được đưa vào trong các đối tượng.) Hãy làm việc với một ví dụ để thấy rõ được điều này.

Tạo một đối tượng thường bắt đầu bằng việc định nghĩa và khởi tạo một biến. Thử nhập dòng lệnh JavaScript sau:

let person = {};

Bạn vừa tạo ra đối tượng đầu tiên. Nhưng đó chỉ là một đối tượng rỗng, do đó chúng ta không thể làm được gì thêm với nó. Hãy cập nhật đối tượng của chúng ta như sau:

let person = {  
   name: ['Bob', 'Smith'],  
   age: 32,  
   gender: 'male',  
   interests: ['music', 'skiing'],  
   bio: function() {  
     alert(this.name[0] + ' ' + this.name[1] + ' is ' + this.age + ' years old. He likes ' + this.interests[0] + ' and ' + this.interests[1] + '.');  
   },  
   greeting: function() {  
     alert('Hi! I\'m ' + this.name[0] + '.');  
   }  
 };

Giờ bạn đã có dữ liệu và chức năng cho đối tượng của mình, chúng ta đã có thể truy xuất chúng qua một vài cú pháp khá đơn giản!

Bạn có thể gọi các thành phần của đối tượng như sau. Ví dụ:

person.bio();

Một đối tượng được tạo ra từ nhiều thành phần, mỗi thành phần đều có tên (VD name và age như ở trên), và giá trị (VD ['Bob', 'Smith'] và 32). Mỗi cặp tên/giá trị phải phân tách nhau bởi một dấu phẩy, và cặp tên/giá trị phân cách bởi dấu hai chấm (":"). Cú pháp như sau:

let objectName = {  
   member1Name: member1Value,  
   member2Name: member2Value,  
   member3Name: member3Value  
 }

Mỗi thành phần thuộc đối tượng có thể có bất kỳ giá trị gì - với đối tượng person ở trên chúng ta có giá trị của các thuộc tính (thành phần) của đối tượng này là một chuỗi, một số, hai mảng và hai hàm. Bốn thành phần đầu là các thành phần dữ liệu, và được gọi là các **thuộc tính** (property) của đối tượng. Hai thành phần sau là các hàm, nó cho phép đối tượng thao tác với các thành phần dữ liệu (thuộc tính) và được gọi là các **phương thức** (method).

Một đối tượng như vậy được gọi là Hằng Đối tượng (object literal) - chúng ta đã định nghĩa các thành phần, nội dung của đối tượng ngay khi tạo ra nó. Điều này trái ngược với việc khởi tạo ra các đối tượng từ các lớp (class), cách mà chúng ta sẽ tìm hiểu sau.

Đây là cách thông thường được sử dụng để tạo ra một hằng đối tượng khi bạn muốn chuyển giao một loạt các thành phần dữ liệu có cấu trúc và có quan hệ với nhau theo một ngữ cảnh nào đó, ví dụ như gửi một yêu cầu tới server để đưa vào trong một cơ sở dữ liệu. Việc gửi một đối tượng sẽ hiệu quả hơn nhiều so với việc gửi các thành phần độc lập, và nó cũng dễ dàng hơn việc sử dụng mảng, khi bạn muốn định nghĩa các thành phần độc lập bằng các tên gọi.

Truy xuất thuộc tính và phương thức của đối tượng:

* objectName.propertyName;
* objectName.methodName;

**Ký hiệu chấm (dấu ".")**

Ở trên, bạn đã sử dụng **dấu chấm** (".") để truy xuất các thuộc tính và phương thức của đối tượng. Tên của đối tượng (person) giống như một **namespace** (không gian tên) - phải được viết trước dấu chấm để có thể truy xuất vào bất cứ thành phần nào được **bao gói** (encapsulated) bên trong đối tượng. Sau dấu chấm là tên của thành phần bạn muốn truy xuất - đơn giản có thể là tên một thuộc tính, một phần tử của một thuộc tính dạng mảng, hoặc gọi đến một trong số các phương thức của đối tượng, VD:

person.age  
 person.interests[1]  
 person.bio()

**Sub-namespaces (không gian tên phụ)**

Bạn có thể khởi tạo giá trị cho một thành phần của một đối tượng khác. Ví dụ, thử thay đổi tên thành phần của đối tượng từ

name: ['Bob', 'Smith'],

thành

name : {  
   first: 'Bob',  
   last: 'Smith'  
 },

Để làm việc này chúng ta tạo ra một **sub-namespace** (không gian tên phụ). Điều này nghe có vẻ phức tạp, nhưng thực tế không như vậy - để truy xuất các thành phần đó bạn chỉ cần bổ sung thêm một dấu chấm (".") sau đó. Ví dụ:

person.name.first  
 person.name.last

**Lưu ý:**Với việc này bạn cần thay đổi mã trong phương thức của đối tượng person, tất cả những chỗ đang viết là

name[0]  
name[1]

được thay bằng

name.first  
 name.last

Nếu không phương thức của đối tượng sẽ không hoạt động được nữa.

**Cặp ngoặc vuông ("[]")**

Để truy xuất các thuộc tính của đối tượng bạn có thể dụng cặp ngoặc vuông ("[]") thay cho dấu chấm ("."):

person.age  
 person.name.first

Bạn có thể dùng

person['age']  
person['name']['first']

Đây là cách khá giống với cách bạn dùng truy xuất các phần tử của một mảng, và căn bản nó là như vậy - thay vì sử dụng chỉ số để truy xuất phần tử giống như mảng, với cách này bạn sử dụng tên để thao tác với giá trị của mỗi thành phần. Bạn sẽ chẳng ngạc nhiên khi các đối tượng đó đôi khi được gọi là **associative array**(mảng kết hợp) - chúng ánh xạ (map) các chuỗi với các giá trị giống như cách mảng ánh xạ các con số với các giá trị.

**Thiết lập các thành phần của đối tượng**

Từ đầu tới giờ chúng ta chỉ thảo luận về cách **lấy ra** (getting) các thành phần của đối tượng - bạn cũng có thể **thiết lập**(set), cập nhật giá trị cho các thành phần đó bằng cách viết tên thành phần (thuộc tính, phương thức) bạn muốn thiết lập sau dấu chấm hoặc cặp ngoặc vuông, VD:

person.age = 45;  
 person['name']['last'] = 'Cratchit';

Thử viết các dòng mã trên và sau đó lấy ra giá trị của các thuộc tính để quan sát sự thay đổi của chúng:

person.age  
 person['name']['last']

Thiết lập các thành phần cho đối tượng không chỉ dừng lại ở việc cập nhật giá trị cho các thuộc tính, phương thức đã tồn tại mà bạn còn có thể tạo ra các thành phần mới cho đối tượng. Hãy thử các dòng lệnh sau:

person['eyes'] = 'hazel';  
 person.farewell = function() { alert("Bye everybody!"); }

Bạn có thể kiểm tra các thành phần vừa thêm mới này (một thuộc tính và một phương thức):

person['eyes']  
 person.farewell()

Một lợi ích khác của cặp ngoặc vuông là nó không chỉ được sử dụng cho việc thiết lập linh động các giá trị của thành phần trong đối tượng mà còn thiết lập linh động cho tên của các thành phần. Giả sử chúng ta muốn người dùng có thể lưu trữ các loại giá trị tùy chọn cho dữ liệu người của họ, bằng cách viết tên và giá trị của thành phần vào hai ô văn bản? Chúng ta có thể nhận được những giá trị đó như thế này:

let myDataName = nameInput.value;  
 let myDataValue = nameValue.value;

chúng ta có thể thêm thành phần mới này với tên và giá trị cho đối tượng person như sau:

person[myDataName] = myDataValue;

Để kiểm tra việc này bạn hãy bổ sung các dòng mã sau, lưu ý là phải sau (bên ngoài) cặp ngoặc nhọn dùng để định nghĩa đối tượng person:

let myDataName = 'height';  
 let myDataValue = '1.75m';  
 person[myDataName] = myDataValue;

Bây giờ hãy thử truy xuất:

person.height

Bổ sung một thuộc tính cho đối tượng sử dụng phương pháp trên không thể dùng với dấu chấm, nó chỉ chấp nhận một chuỗi tên thành phần, không sử dụng biến để chỉ định tên

**Từ khóa "this" dùng để làm gì?**

Bạn có thể chú ý đến một số thứ hơi lạ trong các phương thức của chúng ta. Hãy xem thử VD sau:

greeting: function() {  
   alert('Hi! I\'m ' + this.name.first + '.');  
 }

Bạn có thể băn khoăn từ khóa "this" dùng để làm gì. Từ khóa this dùng để tham chiếu đến đối tượng hiện hành - vì vậy trong trường hợp này từ khóa this tham chiều đến đối tượng person. Vậy tại sao không dùng person để thay thế? Bạn có thể thấy trong bài viết "[Hướng đối tượng trong JavaScript cho những người mới bắt đầu](https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/Object-oriented_JS)" khi chúng ta tạo các contructor (hàm tạo), v.v., từ khóa this là rất hữu ích - hãy luôn đảm bảo rằng các giá trị được sử dụng khi ngữ cảnh của một thành viên thay đổi (VD hai đối tượng person khác nhau có thể có tên khác nhau, nhưng muốn sử dụng tên của chính mình khi nói lời chào).

Hãy quan sát những gì được mô tả ở trên với hai đối tượng người (person) dưới đây::

let person1 = {  
   name: 'Chris',  
   greeting: function() {  
     alert('Hi! I\'m ' + this.name + '.');  
   }  
 }  
   
 let person2 = {  
   name: 'Brian',  
   greeting: function() {  
     alert('Hi! I\'m ' + this.name + '.');  
   }  
 }

Trong ví dụ trên, lệnh person1.greeting() sẽ cho kết quả "Hi! I'm Chris."; person2.greeting() thì cho kết quả là "Hi! I'm Brian.", thậm chí mã trong phương thức của chúng là giống nhau. Như đã trình bày trước đây, từ khóa this chỉ định tới đối tượng chứa mã đó - điều này không quá hữu ích khi bạn tự định nghĩa các hằng đối tượng, nhưng nó thực sự có ích khi bạn tạo các đối tượng động (ví dụ sử dụng với các constructor). Bạn sẽ thấy rõ hơn điều này ở bài viết sau.

**Sử dụng các đối tượng sẵn có**

Khi bạn theo dõi các ví dụ trên bạn có thể đang nghĩ đến việc sử dụng ký hiệu (toán tử) chấm (dấu chấm) là khá quen thuộc. Đó là bởi vì bạn đã sử dụng ký hiệu này trong suốt khóa học! Mỗi khi chúng ta làm việc với các ví dụ sử dụng API được xây dựng sẵn cho trình duyệt hoặc JavaScript, chúng ta đã sử dụng các đối tượng, vì các đặc tính này được xây dựng có cùng cấu trúc đối tượng như những gì chúng ta đã thấy ở trên, tuy vậy chúng phức tạp hơn so với những ví dụ mà chúng ta đã thực hiện.

Vì vậy khi bạn sử dụng các phươn thức để xử lý chuỗi như:

myString.split(',');

Bạn đã sử dụng một phương thức được xây dựng sẵn cho một đối tượng được tạo ra từ lớp [String](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String). Mỗi khi bạn tạo ra một chuỗi, chuỗi đó sẽ tự động được tạo ra dưới dạng một thể hiện (đối tượng) của lớp String, và do đó nó có sẵn những phương thức/thuộc tính dành cho đối tượng loại này.

Khi bạn truy xuất DOM (document object model - mô hình đối tượng tài liệu) giống như sau:

let myDiv = document.createElement('div');  
 let myVideo = document.querySelector('video');

Bạn đã sử dụng những phương thức sẵn có của các đối tượng thuộc lớp [Document](https://developer.mozilla.org/en-US/docs/Web/API/Document). Với mỗi trang web được tải về, một thể hiện (đối tượng) Document được tạo ra, có tên gọi là document, đối tượng này thể hiện toàn bộ cấu trúc, nội dung và các đặc tính khác của trang web, VD như URL của nó. Nhắc lại, điều này có nghĩa là đối tượng document có sẵn một số thuộc tính và phương thức.

Điều này là tương tự khi bạn sử dụng các đối tượng/API khác được xây dựng sẵn - VD như [Array](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array), [Math](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math), v.v..