Tất cả các ngôn ngữ lập trình đều có cấu trúc dữ liệu dựng sẵn, nhưng mỗi ngôn ngữ thường có những kiểu cấu trúc dữ liệu khác nhau. Bài viết này sẽ liệt kê những kiểu dữ liệu có sẵn trong Javascript và những thuộc tính của chúng. Những kiểu dữ liệu này có thể được dùng để xây dựng những kiểu cấu trúc dữ liệu phức tạp hơn.

Kiểu động

JavaScript là một ngôn ngữ *định kiểu yếu* (hay *động*). Điều đó nghĩa là không cần phải khai báo kiểu của các biến trước khi dùng. Kiểu sẽ được xác định tự động trong khi chương trình được thực thi. Điều đó cũng có nghĩa là một biến có thể chứa giá trị của các kiểu dữ liệu khác nhau:

let foo = 42; // foo là một số (number)

foo = "bar"; // bây giờ foo là một chuỗi ký tự (string)

foo = true; // bây giờ foo là kiểu logic (boolean)

Kiểu dữ liệu

Tiêu chuẩn [ECMAScript](http://es6-features.org/) mới nhất xác định bảy kiểu dữ liệu:

* Bảy kiểu nguyên thủy (primitive):
  + number (kiểu số)
  + string (kiểu chuỗi)
  + boolean (kiểu logic)
  + null
  + undefined
  + symbol (mới trong ECMAScript 6)
  + bigint (kiểu số lớn)
* và kiểu đối tượng (object)

Giá trị nguyên thủy

Tất cả các kiểu trừ đối tượng đều được xác định giá trị bất biến (giá trị, không có khả năng thay đổi). Ví dụ, một chuỗi là bất biến. Ta gọi chúng là “giá trị sơ khai” hoặc “giá trị nguyên thủy” (primitive).

Kiểu boolean

Kiểu boolean đại diện có hai giá trị logic: true, và false.

Kiểu null

Có duy nhất một giá trị: null.

Kiểu undefined

Một biến chưa được gán giá trị có giá trị undefined. Xem [undefined](https://developer.mozilla.org/en-US/docs/Glossary/Undefined)  để biết thêm chi tiết.

Kiểu số

Theo tiêu chuẩn ECMAScript, chỉ có duy nhất một kiểu số: the double-precision 64-bit binary format IEEE 754 value (có giá trị từ -(253 -1) đến 253 -1). **Không có kiểu số nguyên**. Ngoài việc có thể chứa giá trị dấu phẩy động, kiểu số có ba giá trị biểu tượng: +Infinity, -Infinity, and NaN (not-a-number).

Để kiểm tra lớn hơn hay nhỏ hơn +/-Infinity, bạn có thể xem Number.MAX\_VALUE hoặc Number.MIN\_VALUE và bất đầu từ ECMAScript 6, bạn cũng có thể kiểm tra một số có nằm trong khoảng double-precision floating-point bằng cách dùng Number.isSafeInteger() cũng như Number.MAX\_SAFE\_INTEGER và Number.MIN\_SAFE\_INTEGER. Ngoài phạm vi này, một số trong Javascript không còn an toàn nữa.

Có một số nguyên duy nhất có hai đại diện: 0 được đại diện bởi -0 và +0. (“0” là một bí danh của +0). Trong thực tế, điều này hầu như không có tác động. Ví dụ +0 === -0 là true. Tuy nhiên, có thể nhân thấy điều này khi chia một số cho không:

> 42 / +0

Infinity

> 42 / -0

-Infinity

Mặc dù một số thường chỉ đại diện cho giá trị của nó, JavaScript cung cấp một vài toán tử nhị phân. Chúng có thể được sử dụng như một chuỗi boolean bằng cách dùng bit masking. Điều này thường được xem như là một cách tệ, tuy nhiên, JavaScript không cung cấp bất kỳ phương tiện nào khác để trình bày một tập hợp các boolean (như một mảng các boolean hay một đối tượng với các thuộc tính boolean). Bit masking cũng có xu hướng làm mã khó đọc, hiểu, và duy trì hơn. Nó có thể cấn thiết trong một môi trường rất hạn chế, giống như khi cố gắng để đối phó với hạn chế lưu trữ lưu trữ cục bộ hoặc trong trường hợp nặng khi mỗi chút so với đếm mạng. Kỹ thuật này chỉ nên được xem xét khi nó là biện pháp cuối cùng có thể được thực hiện để tối ưu hóa kích thước.

Kiểu chuỗi

Kiểu chuỗi được dùng để biểu diễn dữ liệu dạng văn bản. Nó là một dãy “các phần tử” số nguyên 16-bit. Mỗi phần tử có một vị trí trong chuỗi. Phần tử đầu tiên có chỉ số 0, tiếp theo là 1, … . Độ dài của chuỗi là số phần tử của nó.

Không giống với những ngôn ngữ như C, Chuỗi trong Javascript là bất biến. Nghĩa là một khi chuỗi được tạo thì không thể chỉnh sửa. Tuy nhiên, vẫn có thể tạo một chuỗi mới dựa vào các thao tác trên chuỗi cũ. Ví dụ:

* Tạo một chuỗi con của chuỗi ban đầu bằng cách ghép từng ký tự hoặc dùng String.substr().
* Nối hai chuỗi bằng toán tử (+) hoặc hàm String.concat().

Cẩn thận với mã “stringly-typing”!

Chuỗi có thể được dùng để biểu diễn dữ liệu với cấu trúc phức tạp. Điều này mang tới một vài lợi ích ngắn hạn:

* Rất dễ để xây dựng một chuỗi bằng phép nối.
* Dễ debug (những gì bạn thấy khi in luôn là tất cả những thứ có trong chuỗi).
* Chuỗi là điểm chung của rất nhiều API (Input, giá trị được lưu trong localStorage, phản hồi của XMLHttpRequest khi dùng responseText, …) và điều này có thể khiến lập trình viên cảm thấy thích làm việc chỉ với chuỗi.

Chuỗi có thể biểu diễn bất kì kiểu dữ liệu nào. Những đây không được xem là một ý hay. Ví dụ, đối với một separator, có thể bắt trước một chuỗi (trong khi một mảng sẽ thích hợp hơn). Thật không may, khi separator được dùng trong một “danh sách” các phần tử, danh sách bị hỏng. Một escape character có thể được chọn, ….. Tất cả những điều này yêu cầu một quy ước và tạo ra gánh nặng bảo trì không cần thiết.

Chỉ nên dùng chuỗi để lưu trữ dữ liệu văn bản. Khi biểu diễn một cấu trúc phức tạp, phân tích chuỗi thành các cấu trúc dữ liệu với mức trừu tượng cao hơn.

Kiểu Symbol

Kiểu Symbol là một kiểu mới trong Javascript tiêu chuẩn ECMAScript 6. Mỗi Symbol là một giá trị sơ khai **đơn nhất** và **bất biến** và có thể được dùng như một khóa của một Object (xem bên dưới). Trên một số ngôn ngữ lập trình, Symbol còn được gọi là “atom” (nguyên tử). Ta cũng có thể so sánh với các enumeration (enum) trong C. Xem  [Symbol](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Symbol) để biết thêm chi tiết.

Đối tượng

Trong khoa học máy tính, một đối tượng là một giá trị trong bộ nhớ được tham chiếu bởi một định danh. Một đối tượng sẽ bao gồm các thuộc tính và phương thức. Trong bài học hôm nay, chúng ta không đi sâu và kiểu Đối tượng. Đây là một chủ đề sẽ được trình bày rõ hơn trong các bài học về sau.