**Javascript cho người mới bắt đầu – tutorial javascript cho frontend**

# **Hàm (function)**

**Function** là một trong những nền tảng cơ bản trong **JavaScript**. Một **function** trong **JavaScript** tương tự như một thủ tục — một tập hợp các câu lệnh thực hiện một tác vụ hoặc tính toán một giá trị, nhưng để một thủ tục đủ điều kiện là một hàm, nó phải nhận một số đầu vào và trả về một đầu ra trong đó có một số mối quan hệ rõ ràng giữa đầu vào và đầu ra. Để sử dụng một hàm, bạn phải xác định nó ở đâu đó trong phạm vi mà bạn muốn gọi nó.

**Các loại hàm: Gồm có 5 loại function:**

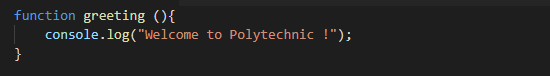
* declaretion function
* expression function
* arrow function
* anonymous function
* gennerator function

1. **Declaretion function**

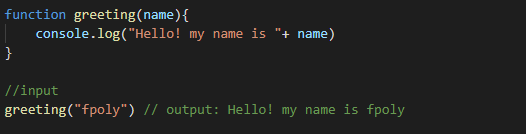
* khai báo **declaretion function** bắt buộc phải dùng từ khóa **function** ở đầu dòng lệnh
* cách khai báo:
  + Dùng từ kháo **function**
  + Tên của hàm
  + Danh sách các tham số của hàm được đặt trong dấu **ngoặc tròn** và được cách nhau bằng **dấu phẩy** (nếu có).
  + Các câu lệnh javascript xác định của hàm, được đặt trong dấu **ngoặc nhọn {…}.**
* **Declaretion function** là **function** có tính chất **hoisting**.

**Ví dụ:**

* Hàm không truyền vào tham số:



* hàm có truyền vào tham sô:

****

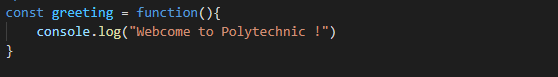
**Giải thích:** Gọi hàm **greeting** và truyền tham số **name = “fpoly”** vào hàm, khi hàm chạy thì sẽ log ra đoạn **“hello! My name is ”** và nối thêm giá trị name truyền sào.

1. **Expression function**

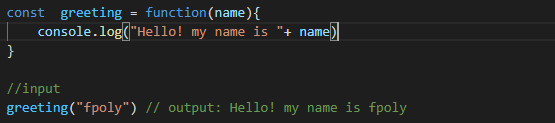
* Với **expression function** chúng ta khai báo bằng cách khai báo biến và gán giá trị **function** thay vì dùng cú pháp dịnh dạng hàm thông thường (dùng từ khóa function ở đầu dong lệnh).
* Cách khai báo:
  + Khai báo biến đển lưu **function.**
  + Dùng từ khóa **function** và **không có có tên của hàm.**
  + Danh sách cá tham số của hàm được đặt trong dấu **ngoặc tròn** và được cách bằng **dấu phẩy** (nếu có).
  + Các câu lệnh javascript xác định của hàm, được đặt trong dấu ngoặc nhọn **{…}**.
* Không giống như **declaretion function, expression function** không có tính chất **hoisting**.

**Ví dụ:**

* Hàm không truyền vào tham số:

****

* Hàm có truyền vào tham số:

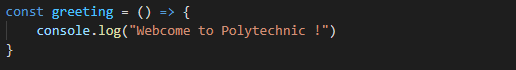
****

1. **Arrow function (cú pháp ra mắt trong bản ES6 - 2015)**

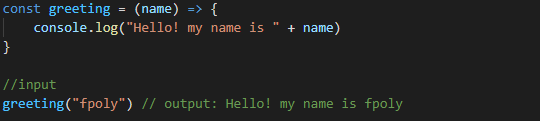
* Đây là hàm dược giới thiệu trong ES6, cũng cấp cho ta một cách chính xác hơn để viết **function** trong javascript. Nó cung cấp cho chúng ta cú pháp viết ngắn gọn và dễ đọc hơn.
* **Arrow function** thừa hưởng tất cả tính năng của **expression function**.
* Cách khia báo: cũng giống **expression function**, chũng ta khai báo bằng cách khai báo biến và gán giá trị của **function** thay vì tạo thông thường.
  + Khai báo biến để lưu **function.**
  + Danh sách các tham số của hàm được dặt trong dấu ngoặc tròn và được cách nhau bằng dấu phẩy.
  + Thay vì dùng từ khóa **function** ta dùng ký tự **“=>”** để định nghĩa **function**.
  + Câu lệnh javascript định nghĩa hàm, được đặt trong dấu ngoặc nhọn **{…}.**

**Ví dụ:**

* Hàm không truyền vào tham số:

****

* Hàm có truyền vào tham số:



1. **Anonymous function**

* Đây là hàm ẩn danh (hay hàm không tên), cái này không có gì lạ cả.
* Cách khai báo:
  + Dùng từ kháo **function** (trong trường hợp nếu **function** là tham só truyền vào một **function** khác thì chúng ta có thể không cần dùng từ khóa **function** )
  + Danh sách tham số của hàm, được đặt trong dấu **ngoặc** **tròn** và cách nhau bằng dấu **phẩy**.
  + Câu lệnh javascript định nghĩa hàm, được đặt trong dấu ngoặc nhọn **{…}.**

**Vậy đặt ra câu hỏi**:

Hàm không có tên thì ta làm cách nào để gọi hàm?

**Anonymous function** ra đời để làm gì?

* + C1: để sử dụng chúng ta có thể dùng cú pháp gán hoặc thêm dấu **ngoặc tròn**, và xem như đối tượng (hàm) này đã được hình thành và gọi ngay sau đó.
  + C2: để trả lời cho câu hàm này dùng để làm gì thì ta hãy đi vào ví dụ sau:

Ta có một hàm tính tổng 2 số ở thời điểm chạy và không dùng ở chỗ khác, thay vì định nghĩa ra một hàm riêng biệt và gọi ra thì ta dùng **anonymous function** ở đó. Vì đoan lệnh đó không được tái sử dụng ở nơi khác.

**Ví dụ:**

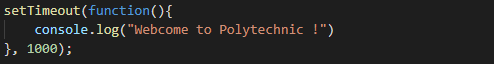
Hàm bình thường:

****

Hàm là một tham số của một hàm khác:



**Hoặc:**



**Lại nảy sinh ra câu hỏi:**

Tại sao ta không viết trực tiếp đoạn mã ra và chạy một lần mà lại dùng **anonymous function?**

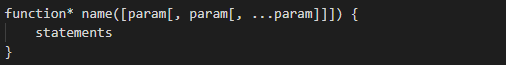
* + Trả lời cho câu hỏi trên: đúng chúng ta có thể không dùng **anonymous function** vẫn được.

Vấn đề xảy ra khi code của chúng ta nhiều hơn thì chúng ta sẽ khó kiểm soát và khắc phục nếu có vấn đề lỗi xảy ra…

* **Tóm lại** : **Anonymous function** thường được sử dụng xử lý đoạn mã chỉ cần được gọi ở một chỗ duy nhất, ta có thể viết hàm riêng rồi sau đó truyền vào hàm vào chỗ đó.

1. **Gennerator function (cú pháp ra mắt trong bản ES6 - 2015)**

* Generator function là một trong những chức năng không còn mới đối với lập trình viên Javascript. Từ phiên bản ECMAScript 2015 (ES6) nó đã được nhà phát triển đưa vào sử dụng bằng cú pháp khai báo "function\* " , và tất nhiên là trả về một Generator object.
* Cú pháp:

****

**Giải thích:** Trong đó thì name: tên hàm. param: tham số đầu vào của hàm, tối đa 255 tham số. statements: phần thân chứa nội dung của hàm.

* **Gennerator function:** đây là phần nhần quan trọng nhưng rất ít được dùng, với loại **function** này mình sẽ chia sẻ ở một series khác!

**Cảm ơn bạn đã theo dõi bài viết này, xin chào vào hẹn gặp lại bạn ở bài học tiếp theo!**