# Bài 5 - Hàm trong JavaScript

## Sử dụng hàm trong JavaScript

- truyền biến vào hàm function tenHam (biến) {code sử dụng biến}

- gọi hàm ra thông qua việc ghi tên hàm với các biến bên trong ()

- có thể sử dụng hàm đã khai báo nhiều lần

## Xây dựng bảng tính bằng JavaScript

- nhập tham số vào form thông qua các button

- lấy value và += để nối chuỗi

- truy cập các input thông qua name của form. Name của input

- dùng trực tiếp onClick để gán sự kiện ngay tại input

## Một số Video tham khảo thêm

- nói sâu thêm về hàm, tham số, biến toàn cục và cục bộ

## Phần lý thuyết

1. Các kiểu dữ liệu trong JavaScript

- biến trong JS – biến động có thể chứ nhiều kiểu dữ liệu như số, chuỗi, đối tượng, …

- kiểu dữ liệu là vô cùng quan trọng, nếu không xác định chương trình có thể có lỗi

- khi cộng chuỗi và số, JS sẽ chuyển số thành chuỗi để thực thi

- JS thực hiện tính toán từ trái sang phải

- chuỗi trong JS được chữa trong ngoặc đơn hoặc ngoặc kép miễn là bên trong nó không có chính loại đã chứa

- số trong JS được gom thành 1 loại, số thập phân ngăn cách bởi dấu “.”

- kiểu booleans chỉ chứa 2 giá trị true/false trong JS, thường được sử dụng để kiểm tra

- mảng trong JS được chứa trong dấu [] các giá trị được ngăn cách bởi dấu “,”

- mảng bắt đầu từ phần tử thứ 0

- đối tượng trong JS được chứa trong dấu {}

- các thuộc tính của đối tượng được viết theo cặp tên: giá trị và ngăn cách bởi dấu “,”

- để xác định kiểu của một biến ta dùng hàm typeof(biến)

- giá trị undefined đại diện cho không xác định trong JS

- giá trị chuỗi rỗng không liên quan đến kiểu không xác định, nó là chuỗi

2. Hàm trong JavaScript

- là một khối code được thực hiện cho các nhiệm vụ riêng biệt

- hàm được thực hiện khi được gọi đến tênHam(danhSachBien)

- cú pháp: function tenHam (cacBien ngăn cách bởi dấu “,”) {code thực thi}

- quy tắc đặt tên hàm giống như biến

- các tham số khi vào hàm sẽ trở thành cục bộ

- trong các chương trình khác chính là chương trình con, thủ tục

- lệnh return giúp hàm trả về một giá trị và ngừng đoạn code sau đó

- hàm giúp tránh việc lặp lại code

- nếu gọi hàm mà không có (), sẽ trả về đối tượng là chính hàm đó

- hàm có thể sử dụng như biến và tham số

- các tham số cục bộ trong hàm được tạo khi hàm bắt đầu và xóa đi khi hàm chạy hoàn tất

3. Các định nghĩa Hàm

- hàm được định nghĩa theo từ khóa khai báo nó

- có thể sử dụng hàm với tên được khai báo hoặc biến gán bằng hàm

- hàm với tên sẽ được lưu lại để chạy sau chứ không chạy ngay lập tức

\* hàm với tên không kết thúc với dấu ; bởi vì nó là một khối code chứ không phải lệnh sẽ được thực thi

- hàm được gán qua biến sẽ được lưu trữ trong biến và có thể dùng biến như hàm

- hàm mà không có tên là hàm nặc danh – anonymous

- hàm được gán qua biến cần kết thúc bởi dấu ; bởi vì nó là một câu lệnh

- hàm có thể khai báo với constructor với tham số là các biến và cuối cùng là câu lệnh thực thi, tất cả được chuyển thành chuỗi

- bởi vì constructor được khai báo với new – new Function(), mà hầu hết các trường hợp từ khóa new không được khuyến khích nên cách này rất hạn chế

- nhờ cơ chế Hoisting – nâng mà JS sẽ tự động chuyển hàm được khai báo lên đầu để thực thi dù cho nó có được khai báo sau

- hàm được gán vào biến sẽ không nhận cơ chế Hoisting này

- hàm có thể tự gọi chính nó khi nó không có từ khóa khai báo

Vd: a = function() {}(); hoặc (function() {})(); sẽ tự chạy

- hàm có thể dùng như một giá trị, hàm là một đối tượng – Objects

- dùng arguments.length để trả về số lượng biến mà hàm nhận

- .toString() chuyển hàm thành một chuỗi

- hàm được định nghĩa như một thuộc tính của một đối tượng được xem là phương thức của đối tượng đó

- hàm được thiết kế để tạo một đối tượng mới được gọi là đối tượng – object constructor

- hàm mũi tên – arrow function chỉ hoạt động với hàm không có từ kháo giúp code ngắn hơn

+ hàm mũi tên không cần {} và từ khóa return để trả về, vẫn có thể dùng đủ nếu thích

+ const x = (x, y) => x \* y;

+ hàm mũi tên không có đối tượng this, nên nó không thể dùng để khởi tạo một đối tượng

+ hàm mũi tên không được hưởng cơ chế hoisting nên bạn cần khai báo nó trước khi sử dụng

+ nên sử dụng const khi gán hàm vào biến, bởi hàm không thể thay đổi

+ hàm mũi tên không hỗ trợ các phiên bản bằng hoặc sớm hơn IE11

4. Các tham số của Hàm

- hàm trong JS không thực hiện kiểm tra về mọi mặt với các đối số được truyền vào

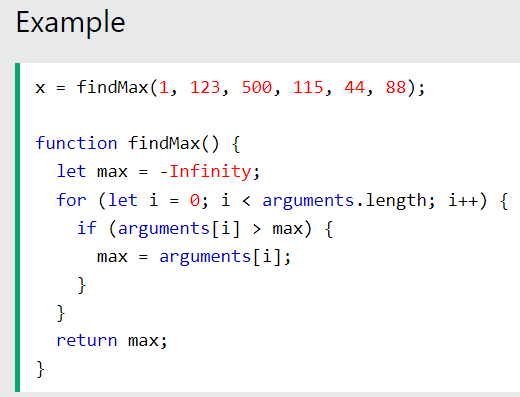
- các tham số của hàm là danh sách tên các giá trị được định nghĩa

- các đối số của hàm là các giá trị mà hàm nhận được khi thực thi

- nếu bạn gọi hàm mà thiếu đối số, các giá trị thiếu sẽ chuyển thành undefined

- bạn có thể truyền giá trị mặc định cho tham số: function myFunction(x, y = 2)

- hàm có đối tượng Argument là nơi chứa tất cả tham số truyền vào



- nếu như các tham số được truyền nhiều hơn được quy định, sử dụng Argument để truy cập chúng

- các đối số của hàm đều được chuyển thành các giá trị mới riêng biệt và không ảnh hưởng tới biến bên ngoài chứa nó

- tuy nhiên nếu là đối tượng – Object thì nó sẽ bị thay đổi do mặc định đối tượng đã là một giá trị

5. Các cách gọi Hàm

- các đoạn code bên trong hàm sẽ chạy khi hàm được gọi

- các hàm mặc định gọi ra sẽ là đối tượng toàn cục thuộc về window

- không nên khai báo các hàm một cách toàn cục có thể tạo xung đột gây lỗi

myFunction() = window.myFunction()

- từ khóa this đại diện cho đối tượng, this phụ thuộc vào cách nó được gọi và được sử dụng

+ trong phương thức khởi tạo đối tượng, this đại diện cho đối tượng

+ ở ngoài, this đại diện cho đối tượng toàn cục, trong hàm this vẫn đại diện cho đối tượng toàn cục

+ trong hàm ở chế độ nghiêm ngặt thì this là undefined

+ trong sự kiện, this đại diện cho phần tử nhận sự kiện

+ các phương thức call(), apply(), bind() giúp tham chiếu đến các đối tượng

\* this không phải là một biến, nó là một từ khóa truy cập. bạn không thể thay đổi giá trị của this

- bạn có thể dùng hàm như một phương thức của đối tượng, giúp this trở thành đối tượng chứa hàm

- từ hàm đã có bạn có thể dùng từ khóa new cùng tên hàm để khởi tạo lại và thừa kế nó bằng một biến

- this trong constructor không có giá trị, nó được khởi tạo và gán vào biến đã thừa kế hàm qua constructor

6. Đóng kín Hàm (Function closures)

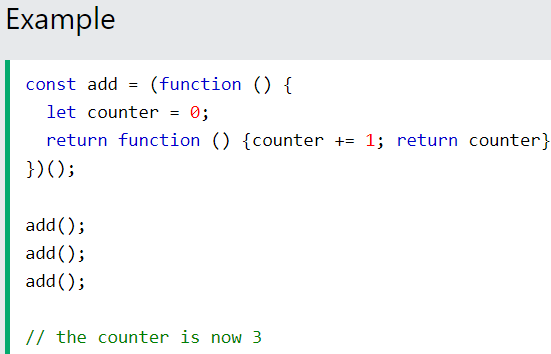
- giúp các biến trở nên cục bộ

- biến được khai báo mà không có từ khóa var, let, const sẽ luôn luôn là toàn cục

- biến toàn cục tồn tại cho đến khi đóng trang, biến cục bộ tồn tại khi khối code chứa nó được thực thi

- bởi tính cục bộ và toàn cục, làm sao để có thể tự tạo một biến cục bộ, sau đó truy cập và thay đổi nó

- lúc này ta dùng phương pháp đóng kín hàm thông qua hàm tự gọi



7. Arrow functions cho phép bạn viết các hàm với các cú pháp ngắn hơn

- được giới thiệu trong ES6

- ngắn gọn (a, b) => a \* b;

- có thể bỏ vào chữ return và {} nếu muốn

- nếu chỉ có 1 giá trị truyền vào có thể bỏ ()

- this trong Arrow functions luôn đại diện cho window toàn cục