**THUẬT TOÁN**

1. **Bobble Short Algorithm** : thuật toán sắp xếp Bubble

Ví dụ : chúng ta có một mảng array sản phẩm. Sắp xếp từ nhỏ đến lớn :

* So sánh từng cặp liên tiếp nhau với khời đầu là còn số thứ 1 🡪 đến cuối cùng chúng ta có được 1 con số lớn nhất.
* Tiếp tục làm bỏ số mới tìm được ở trước đi 🡪 có nghĩa là chúng ta giảm số cặp theo từng vòng lặp.

Kinh nghiệm : Trường hợp khó khăn nhất số nhỏ nhất nằm ở cuối cùng của array.

* Mỗi vòng lặp giảm 1 cặp xuống.
* Hoạt động tốt với array nhỏ.

1. **Insertion Sort Aligorithm :** thuật toán sắp xếp chèn

Dùng rất tốt khi mảng đã gần như sắp xếp rồi.

Ví dụ : mảng trước đó đã được sắp xếp nhưng bị thay đổi.

1. **Hight-order-function:**

* Khi chúng ta gọi hàm nhiều lần thì giá trị của hàm không thay đổi.
* Khi chúng ta gán hàm đó cho một biến rồi gọi lại thì giá trị của hàm thay đổi.
* Nhiều function, return lồng nhau : closure. Tạo biến private trong hàm.
* Hàm sẽ không dừng lại khi chúng ta return ra một hàm khác.
* Hàm con có thể truy xuất vào data của hàm cha.
* Callback : hàm truyền vào hàm khác dưới dạng tham số 🡪 giúp js sử dụng tác vụ bất đồng bộ như là bất đồng bộ.

1. **Callback: js là ngôn ngữ đồng bộ.**

* Callback : giúp code viết ngắn gọn, dễ bào trì.

Callback trong bất đồng bộ :

Nhờ vào môi trường giúp chạy bất đồng bộ.

* Môi trường browser API Timer như setTimeout, DOM hay micro( promise ), AJAX với XMLHttpRequest, request
* Dùng callback, promise để bắt output sắp xếp thứ tự kết quả theo ý đồ mong muốn.

Event loop trên browser và Nodejs :

Javascript không hỗ trợ chạy bất đồng bộ.

Even loop : Mỗi khi code được thực thi thì nó đẩy vô 🡪 callstack ( có cấu trúc như todolist LIFO )

* Nếu ta gọi hàm đồng bộ thì nó sẽ ouput ra luôn.
* Nếu ta gọi hàm bất đồng bộ thì trên môt trường Nodejs thông qua event loop đưa các tác vụ bất động bộ tương ứng với các API cụ thể. API 🡪 đến phong trờ queue 🡪 callstack trống thì 🡪 event loop sẽ đẩy các tác vụ ở trong hàm chờ theo đô ưu tiên:

1( rendering queue ) browser.

2 nextTick ( Node ), promise, ...

3 “Phòng chờ” callback của hàm chạy hàm bất đồng bộ như onClick, setTimeout, ...

Ưu tiên , tối ưu thời gian : chúng ta cho tác dụng bất đồng bộ chạy trước.

* MicroTask queue : được ưu tiên hơn macro task
* MacroTask queue :

Browse :

Sa

Nodejs :

Event loop trên Nodejs thuộc thư viện Libuv.

Event loop chạy trên main thread của javascript.

Event loop trên node có nhiều phase khác nhau.

1. **Promise**

Promise : là một parttern ( mẫu, chuẩn ).

* Dùng nó xử lý code bất đồng bộ theo kiểu bất đồng bộ.
* Dễ dàng thay đổi thứ tự của các tác vụ bất đồng bộ.

Then, catch, finally phương thức của promise.

* Then : còn tham số thứ 2 là bắt lỗi nếu không muốn dùng catch( ).
* Then không có return trong callback của then( ) :
* Then có return trong callback của then() :

Return sẽ có mặt trong then( ) tiếp theo. Return trả về một cái promise mới

1. **Regular Expression**

Khái niệm : là một dạng partten.

/[ ]/: character class: biểu thức mô tả.

. trong biểu thức là một kí tự đặng biệt nó matches với các kí tự đặng biệt trên cùng 1 hàng. Nếu muốm tìm kiếm dấu chầm dùng \.

+ thêm vào cuối để matches tất cả nhưng kĩ tự trước nó, đứng liền nhau.

Special character class :

\w lựa chọn hết chữ hoặc số và dấu gạch dưới.

\W trái ngược với \w.

\d lựa chọn tất cả các số.

\D trái ngược với \d

\s lựa chọn tất cả các khoảng trắng, tab,

\S trái ngượi với \s

[^] nghịch đảo những gì đằng sau nó.

Phương thức test trong Regex :

Method test: kiểm cha xem chuỗi có matchs không trả về true hoặc false.

Anchors :

* ^ : bắt đầu 1 string.
* $ : kết thúc 1 string.
* \b : giới

Quantifiers : toán tử

* + : gộp ít nhất 1 lần.
* ? : trước dấu hỏi có cũng được không có cũng được.
* \* : gộp nhiều lần.
* { m } : gộp m lần.
* { 2, } : gộp ít nhất 2 lần liên tục với nhau.
* ( ): toán tử gộp.

Flags

g: tất cả từ tìm kiếm.

i: không phân biệt chữ hoa chữ thường.

Method.prototype.match :

Trả về một mảng.