**1. Browser xử lý JS như nào ?**

=> thông qua JS engine, js engine nổi tiếng nhất là V8 của chrome

1.Js Chạy trên đâu? client

2.Cái gì sử lý JS. browers Xử lý js

3.tại sao?

vì trong mỗi trình duyệt web có 1 cái js Engine (nổi tiếng hiện tại y8)

bên trong y8 gồm có 3 bộ con :

**\*1** là **parser** chịu trách nhiệm đọc code

nếu đúng nó sẽ báo trong cosole và chuyển sang cho **trasfer**.

-**đồng bộ thì xử lý** : đưa vào **stack** xử lý

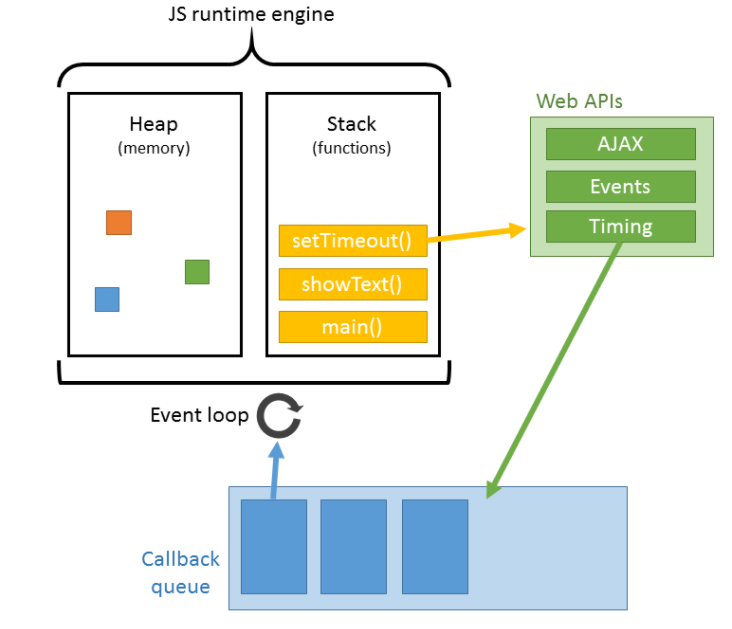
-**bất đồng bộ thì xử lý**:

-đưa vào **web APIs**(ko đưa vào stack xử lý) xử lý sau đó đưa xuống hàng chờ **callback queue** .

-event loop kiểm tra nếu stack rỗng(các đồng bộ dc xử lý xog hết) thì nó sẽ báo cho các bất đồng bộ đưa lên stack. ko thì báo lỗi và dừng .

**\*2** là **trasfer** là bộ chuyển JS sang **mã máy** sau đó đua xuống **excute** xử lý.

**\*3** là **execute**.



**2. Cơ chế hoisting là gì ?**

=> Cơ chế hoisting là cơ chế đưa các phần khai báo biến và hàm lên các dòng đầu tiên của Scope chứa nó. Tuy nhiên, chỉ di chuyển các phần khai báo, phần gán giá trị vẫn giữ nguyên vị trí cũ.

là cơ chế cho phép sử dụng biến trước khi dc khai báo. đơn giản vì parser thu thập biến,hàm trước khi đọc code nên mới có cơ chế hoisting.

**3. Callback function là gì?**

=> Callback function là một function A được truyền vào một function B dưới dạng một tham số. Tại một thời điểm nào đó tùy thuộc vào cách function B được xây dựng mà function A sẽ được function B gọi để thực thi.

*example:*

*const sayHello = (name, callback) => {*

*const msg = "Hello, " + name;*

*return callback(msg);*

*};*

*const useCallBack = sayHello("Khoa", (msg) => {*

*return msg;*

*});*

*console.log(useCallBack);*

**4. Em hiểu gì về closure?**

=> Closure là một hàm được viết lồng vào bên trong một hàm khác (hàm cha) mà nó có thể sử dụng và thay đổi các tham số của hàm cha lẫn của chính nó.

Example:

const increase = () => {

let x = 0;

const increaseInner = () => ++x;

return increaseInner;

};

console.log(increase()()); *// output: 1*

console.log(increase()()); *// output still: 1*

const useClosure = increase();

console.log(useClosure()); *// output: 1*

console.log(useClosure()); *// output: 2 (voila !?)*

**4.1. Có mấy cách khai báo function trong js. Khác nhau là gì?**

=> Có 2 cách để khai báo function:

* Declaration function (DF):

function logSomething() {

*// do something*

}

* Expression function (EF):

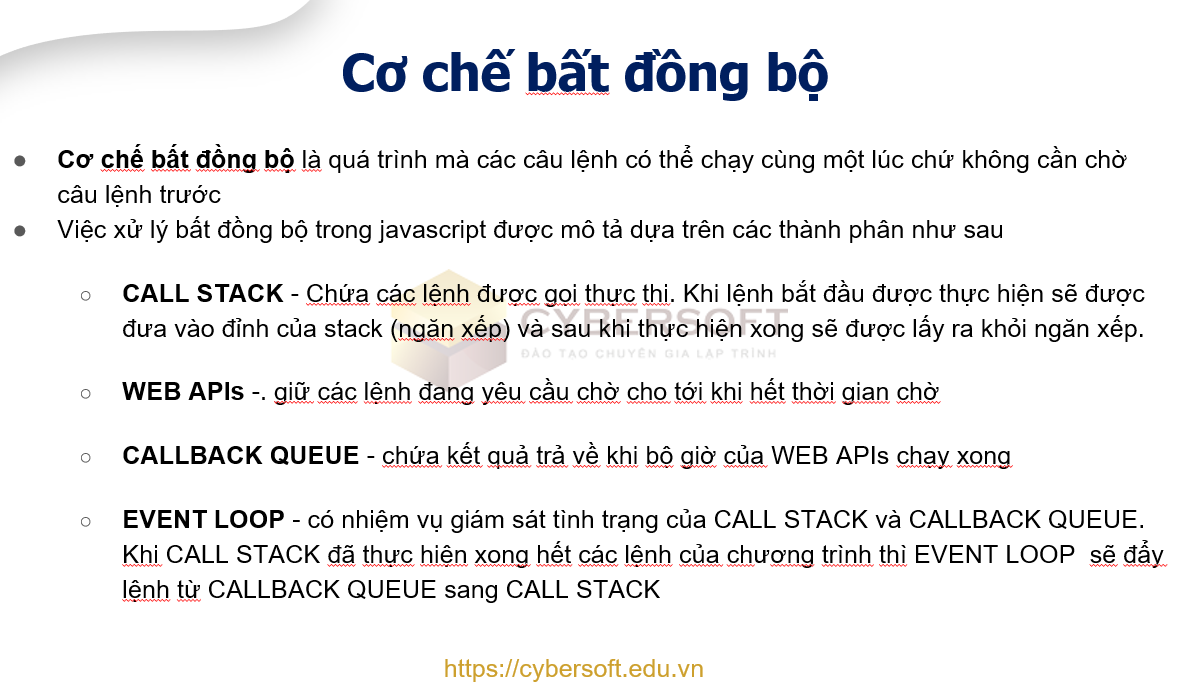
const logSomething = function () {

*// do something*

};

DF bị ảnh hưởng bởi cơ chế hoisting, còn EF thì không. Vì vậy, nếu gọi hàm viết dưới dạng EF chạy trước khi khai báo sẽ bị lỗi.

**5. Em hiểu gì về event loop**



**6.Sử dụng gì để call api, ngoài axios?**

=> - AJAX (Jquery)

- fetch() của ES6

**7.Cho câu sau, thứ tự in ra đúng là gì ?**

setTimeout(() => console.log(‘a’),200)

setTimeout(() => console.log(‘b’),0)

console.log(‘c’)

=> Explain: hàm setTimeout là hàm bất đồng bộ, sau khi đọc đến đây. JS Engine sẽ tạm thời đưa vào callback queue để chờ sau khi lệnh console.log(“c”) chạy xong, setTimeout nào có thời gian chờ ngắn hơn, sẽ được xếp trước vào callback queue. Vì vậy, kết quả lần lượt là: c

b

a

**8. Khác nhau giữa let,var và const**

=> var bị ảnh hưởng bởi cơ chế hoisting và hoạt động ở mọi nơi trong global scope

=> let không bị ảnh hưởng bởi hoisting, có thể tái gán giá trị và hoạt động trong block scope.

=> const không bị ảnh hưởng bởi hoisting, không thể tái gán giá trị và hoạt động trong block scope.

**9. Có cách nào thay đổi được giá trị của const ko? Tại sao?**

=> Có thể nhưng cũng không thể, nếu trường hợp giá trị của const là kiểu tham trị thì không thể thay đổi nhưng ngược lại, nếu giá trị của const là kiểu tham chiếu (array, object) thì chúng ta có thể thêm vào hoặc bớt đi phần tử bên trong. Nguyên nhân là khi chúng ta thay đổi giá trị bên trong array hay object của một biến const, việc đó chỉ góp phần thay đổi vùng nhớ của biến chứ không thay đổi giá trị của biến, bản chất biến const ta đang đề cập vẫn là một array hay một object.

**10. Arrow function khác gì với function thường ?**

=> Arrow function không làm thay đổi ngữ cảnh của con trỏ **this** và không thể sử dụng agruments.

arguments là object giống Array khả truy cập bên trong hàm, chứa các giá trị của đối số truyền vào trong hàm đó. ... Ghi chú: “Giống Array” tức là arguments có thuộc tính length và đánh chỉ mục từ không (0), nhưng nó không có phương thức dựng sẵn của Array như forEach() và map()

**11**. **ES6 offer các tính năng mới gì cho mình sử dụng, kể tên?**

=> Arrow function, Class, promise, let & const, template string,Destructuring…

**11.1 Destructuring là gì ?**

=> Bóc tách phần tử của object(Biến tham chiếu).

**11.2 Template string là gì ?**

=> Template Literals hay còn gọi là Template Strings là một cú pháp mới để khai báo String trong Javascript được giới thiệu trong ES2015/ES6. Nó cho phép chúng ta sử dụng multi-line string, sử dụng biến, biểu thức, hàm bên trong string mà không phải thông qua phép cộng string.

Ví dụ :

Let name = ‘kha’;

Let message = ` ${name} sẽ đậu phỏng vấn ! `;

**12 Higher order function là gì :**

- Higher order functions là hàm hoạt động trên các hàm khác , bằng cách lấy chúng làm tham số hoặc trả về chúng. Nói một cách đơn giản, một Higher-Order function là hàm nhận một hàm dưới dạng đối số hoặc trả về hàm dưới dạng đầu ra.

[{},{},{}]. forEach( **( item, index ) => {}** )

**12.1 Kể tên các higher order function trong ES6**

=> reduce, map, filter, foreach

**Foreach:**

-là phương thức thực thi một hàm 1 lần cho mỗi phần tử .nếu mảng có 7 phần tử thì thực thi hàm đó 7 lần. -hàm nhận tham số đầu vào là từng phần tử của mảng và vị trí. forEach( ( item, index ) => {} ) -forEach duyệt tự động theo chiều dài của mảng và mỗi lần duyệt nó sẽ trả về 1 đối tượng . Nói nôm na nó cũng giống như filter() nhưng khác ở chỗ filter() chúng ta cần hứng 1 giá trị gì đó. -filter() thì trả về mảng mới ,còn forEach() không trả về gì cả.

**13.Tại sao lại gọi là High order function ?**

=> function chứa tham số là một callback, hoặc function trả về một function khác (closure)

**14. Spread operator dùng khi nào ?**

=> Khi cần clone, merge array/object.

Array :

Let listA =[a,b,c];

Use Spread operator : […listA,d,e,f]; (khi clone )

**15. 4 tinh chất của đối tượng?**

1 là đóng gói .

2 là tính kế thừa .

3 là tính đa hình .

4 là tính trừ tượng.

**16. Nếu có 3 promise đang chạy, nếu muốn chờ xong cả 3 mới làm thì dùng cách nào ?** => promise.all

**Promise là** một cơ chế trong **JavaScript** giúp bạn thực thi các tác vụ bất đồng bộ mà không rơi vào callback hell hay pyramid of doom, **là** tình trạng các hàm callback lồng vào nhau ở quá nhiều tầng.

17.Bubble sort, quick sort chạy như thế nào?

**Bubble sort:**

Sắp xếp nổi bọt (tiếng Anh: **bubble sort**) là một thuật toán sắp xếp đơn giản, với thao tác cơ bản là so sánh hai phần tử kề nhau, nếu chúng chưa đứng đúng thứ tự thì đổi chỗ (swap). Có thể tiến hành từ trên xuống (bên trái sang) hoặc từ dưới lên (bên phải sang).

17.Dự án trước làm về gì ? Trong dự án đó có gì e cảm nhận là khó nhất !