You are in a technical interview for a full stack engineer position:

1. The first question you will have to answer is explaining JS event loop, clarifying the synchronous and asynchronous parts, queues, priority, with examples. \* write your answer as if it were for a real interview. \*

**Answer :**

JavaScript executes code in a single thread, which makes it blocking. Event loop is what allows JavaScript to perform non-blocking I/O operations. It works by continuously checking the message queue and executing any code that is waiting in the queue. The fact that JavaScript executes code in a single thread make the execution **synchronous** by default, meaning that code is executed in the order we wrote it in the code. But there are certain cases that we want to execute a code in different order which we call Asynchronous, Which means the execution occur outside the main execution stack. When we run **asynchronous** code, that code will be handled by Web APIs. So we will be able to rune more than one process running at a time.

**Queue:-**  those messages in the wait list will be stored in the message queue. Like the data structure queue, we will have first in first out order. The old message in the queue is executed first.

We have microtasks and macrotasks queue in JavaScript, microtasks are executed before macrotasks, and the priority is given to the microtasks. For example, promises are considered microtasks and setTimeout is considered macrotask.

console.log('Code Start');

const promise = new Promise((resolve) => resolve(`Promise results`));

setTimeout(() => {

console.log('setTimeout!');

}, 0);

promise.then(console.log);

console.log('Code End');

setTimeout will convert the code to Asyncronous and since promise is going to be stored in microtasks queue it will have priority and the out put will be

Code Start

Code End

Promise results

Timeout

1. The follow-up question is how may we convert a sync operation/function to become an Asynchronous?

**Answer :**  in JavaScript, you can convert a sync operation or function to become asynchronous by using the setTimeout() function, setImmediate or by using Promise and async/await.

Example

​​function f1() {

console.log('synchronous function');

}

function f2() {

setTimeout( f1, 0);

}

In the above example f1 is sync function, the code will run in chronological order. But f2 is asynchronous function since we use setTimeout.

function synchronousFunction() {

console.log('synchronous function');

}

async function asynchronousFunction() {

await new Promise(resolve => setTimeout(resolve, 0));

synchronousFunction();

}

In this example, async await make the code asynchronous.