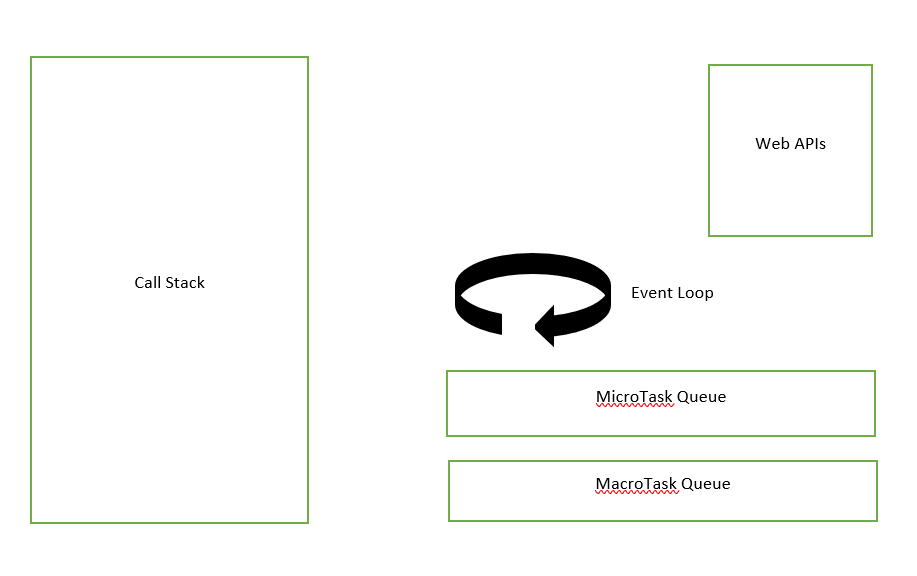
**Answer 1:**

Whenever there is a task that needs to be passed on to the Web API, it gets passed on to it and after completing the result gets queued into either the MicroTask Queue or the MacroTask Queue, depending upon the priority.

All the tasks that get queued in the MicroTask Queue are called microtasks and all the tasks that get queued in the Macrotask queue are called macrotasks.

The order of execution is all the tasks in the Call Stack are completed first, then the tasks in the MicroTask queue and at last the MacroTask queue tasks.

Example:

console.log(“Hello”);

let newPromise = new Promise((resolve,reject) => resolve());

newPromise.then(console.log(“promise resolved”));

setTimeout(() =>{

console.log(“from setTimeout”)

},1000);

Initially the first console.log gets into the call stack and then it is processed and output gets to the console.

Then the second line gets into the call stack, as seen it is a promise it gets passed on to the Web API.

Thirdly the setTimeout gets into the call stack and as this too needs to be handled by WebAPI is gets passed on to it.

After the promise is resolved in the WebAPI , promise.then gets queued in the microtask queue.

Similarly setTimeout callback too gets queued but in the macroTask queue.

Then event loop checks if the Call Stack is empty, if it is then first priority is given to the microtask queue and the call inside it gets to the call stack.

Then when everything from the microtask queue is executed then event loop takes tasks from the microtask queue and place them to the call stack.

Promises are treated specially and always placed in the microtask queue whereas other callbacks get placed in the macrotask queue.

**Answer 2:**

Private variables are those variables of a class that can be accessed only within the class.

Ex:

class Car{

#wheels = 4;

}

The wheels variable in the above class cannot be accessed outside the car class.

class Maruti extends Car{

}

const maruti = new Maruti;

console.log(maruti.#wheels);

if we try to execute the above line, it would throw an error because **#wheels** is a private member and it cannot be accessed outside the class.

Protected variables are quite similar to private variables but those can be access by inherited or sub classes.

Ex:

class Car {

    \_wheels = 4;

}

class BMW extends Car {

    wheels = this.\_wheels

    getWheels() {

        return this.wheels;

    }

}

The wheels variable declared in the parent class can be accessed by the inherited class.