Events Object and Async In Java-Script

Events Object

Definitions

Target

Event.target return the element which is targeted by user by performing event

Event.targetParentNode

This return the parent element of target

Remove()

This use to remove the element

Event.target.ParentNode.removeChild(target)

Use to remove the childElement but we have pass it

```
document.querySelector('.images').addEventListener('click', function(event){
    console.log(event.target);


    // let removeitem =event.target.parentNode // remove the parentNode


    if(event.target.tagName ==='LI'){
        let removeitem =event.target
        removeitem.remove();
    // removeitem.parentNode.removeChild(removeitem) both work same
    }
    // removeitem.parentNode.removeChild(removeitem) both work same
} ,false);
```

Async (Asynchronous) In Java-Script

Definitions

What is asynchronous in JavaScript?

In summary, asynchronous programming is an essential concept in JavaScript that allows your code to run in the background without blocking the execution of other code. Developers can create more efficient and responsive applications by using features like callbacks, async/await, and promises.

JavaScript :

- is a synchronous
- single threaded language

Execution context

- execute one line of code at a time
- each operation wait for last one to complete before executing
- contain Call stack
- Memory Heap

Note :

- alone JavaScript engine is slow as it is synchronous so every where js engine present with run time environment like browser environment , node.js environment
- program does read file for that the execution context given to carnel then carnel goes and access the file then after read the desired material then at last execution context give back to program

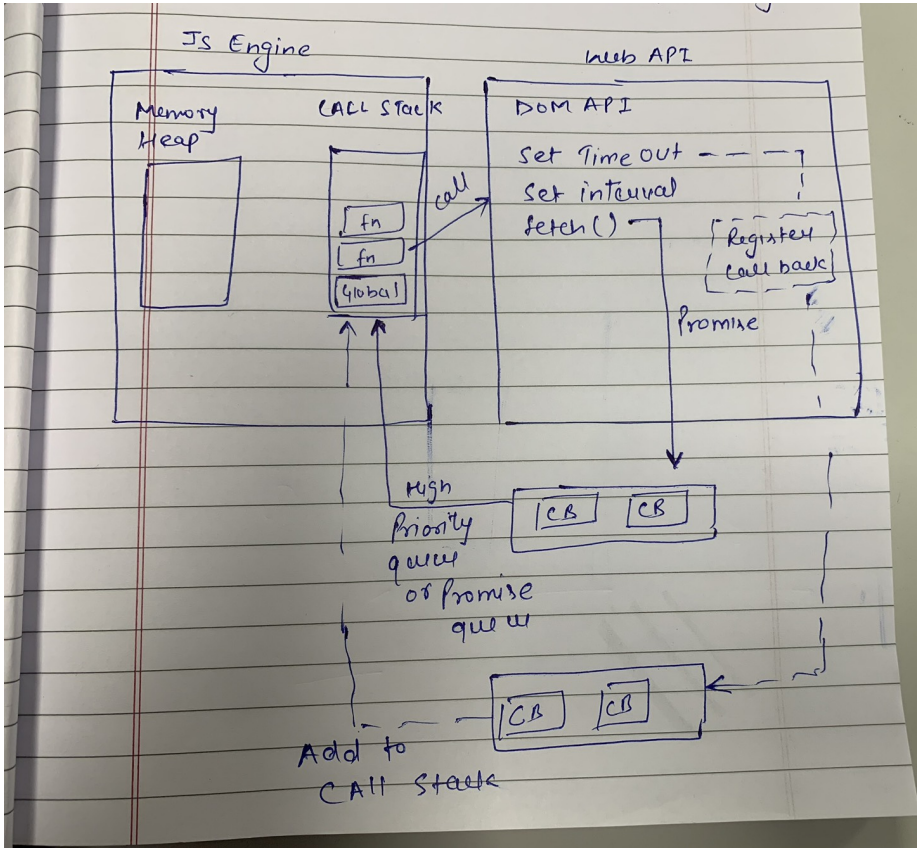
Blocking code

- block the flow of program
- Read File Synchronous

Non Blocking code

- Does not block the execution program
- Read File Asynchronous

Event Loop Diagram



explanation of diagram

- if function inside call stack want to set the timeout then it call setTimeout() from API Present
- Node API (Node environment do not have DOM API)
- register callstack : register all the event and add a callback for that event to a task Queue
- if there is two call back inside call stack and we then we setTimeOut to 0 then do exist call execute in 0 time no because setting timeout is also a call back and it register by the help of register call stack and add to task Queue this flow repeat on adding every event
- fetch() is new API and inside a High priority queue form