Event Loop :

Event loop is non blocking asynchronous execution mechanism that effeciently manages

I/O operations and callbacks, allowing for concurrent handling of tasks.

When node js is start it initializes the event loop, processes the provided input script

which may make async api calls , schedule timers, or call process.nextTick(), then

begins processing the event loop.

Phases >

* timers
* pending callbacks
* idle, prepare
* poll
* check
* close callbacks

Each phase has its own FIFO queue.

Event Emitter :

Eventemitter basically is an event by using this we can create custom event.

It exposes two methods emit and on

emit() = basically it use to trigger the event

on() = basically it is use to register the event

Example :-

const EventEmitter = require('events');

const eventEmitter = new EventEmitter();

eventEmitter.on('start',() => {

console.log("Event started");

})

eventEmitter.emit('start');

EventEmitter in node js is a built-in modules, it enables you to work with custom events.

EventEmitter basically it exposes several methods by using that we can handle custom events

1) on,addListener -> is used to register listeners( this both are working same)

3) once -> it is listen only once and after any emit is called it will ignored

2) emit -> is used to trigger the specific event

4) removeListener(eventname, functionName) -> it accept two parameter event name and function name

5) removeAllListeners() -> it will remove all listeners of node js program

Difference between spring boot and node js

node js : If we have to handle multiple I/O then in that case we choose node js, it require less memory.

spring boot : If we have to create standalone application with full of security then in that case i should go for spring boot, it require high CPU intensive usage.

What is let,const and var ?

A variable declared with the var keyword has global scope, and its value can be changed throughout the program.

A variable declared with the let keyword has block scope, meaning it is limited to the block (usually within curly braces {}) in which it is declared.

const is similar to let, but with one key difference: the value assigned to a const variable cannot be changed after it is initially assigned.

Scope and Closures ?

Scope in javascript defines what variable you have access to. two types of scope

1) global -> A variable which is declared outside functions or curly bracs({}) , it is said global scope.

2) local > A variable which is declared inside functions or curly bracs({}), it is said local scope.

Closures -> When you create function within another function, it means you have created a closure.The inner function is the closure. This closure usually returned so you can use the outer function variable.

Closure Example :

function outerFunction() {

let outer = 2;

return function innerFunction(value) {

console.log(outer, value);

};

}

ES6 Features :

let keyword = it is used to create variable and it has a scope inside curly bracs({}).

const keyword = it is same as let, but one key difference ( after initialization we cannot change the value of it).

arrow functions = it is a short syntax fo writing function expressions. You dont need the function keyword, the return keyword, and the curly brackets.

spread operator = it is use to combine multiple array value into one array.

For of =

Map Objects = in this value stored in the form of key value pair, it maintains insertion order

Set Objects = it is not allowing duplicate value

Classes = it is a blueprint of object, it contains method,variables

Promises = Promise is an object representing the eventual completion or failure of an asynchronous operation.

String.includes() = include function will take string as a paramter and check inside string.

String.startsWith() = startsWith() function will take string as a parameter and check whether string is start with given parameter

String.endsWith() = endsWith() function will take string as a parameter and check whether string is ends with given parameter

Array.from() = this will create string into array

Array.keys() = It will return array iterator object with the keys of the array

Array.find() = It will return the first value of array if it pass the condition

Array.findIndex() = It will return the index value of first element of array if it pass the condition

Object entries() = It will return the object of given array

const fruits = ["Banana", "Orange", "Apple", "Mango"];

const f = fruits.entries();

for (let x of f) {

  console.log(x);

}

IIFE => Immediately invoked function express

Difference between normal vs arrow function

1. this binding > Arrow function does not have there own this , where normal function have there own this keyword.
2. Arguments > arrow function does not have arguments object as paramter( it is like an array), where normal function have there own arguments object.

Callbacks, Promises and async/await

Callbacks > Callback is a function it is pass as an argument to another function. It will execute only when function execution is complete. The fundamental concept of callback is use to handle the asynchronous operation , such as reading file , handling the network request.

Promises > It is use to handle the asynchronous operation.

Async/await > It is same as promises. We can use await multiple time inside an async function, but one thing we have to keep in mind is that function should be async.

Error handling

Describe the event delegation pattern in JavaScript.

Event delegation in javascript is a pattern that efficiently

handles events.Events can be added to a parent element

instead of adding every single element.

It is useful because the event can be listened to on multiple

elments by using just one event handler. It also uses less memory

and give better performance.

The 3 steps to using event delegation

> Determine the parent of elments to watch for events

> Attach the event listener to the parent element

> Use event.target to select the target element

Binds, Calls and apply method

Bind() >

bind() method is used to create a new function and sets the this keyword to the specified object.

Call () >

The call method sets the this inside the function and immediately executes that function.

The difference between call() and bind() is that the call() sets the this keyword and executes the function immediately and it does not create a new copy of the function, while the bind() creates a copy of that function and sets the this keyword.

Apply() >

The apply method is same as call method only the difference is that it accept the array of argument instead of comma separated.

The call, bind and apply methods can be used to set the this keyword independent of how a function is called.