Nodejs

Observables in angular

**- Design Patterns**

**1-Singleton pattern**

we don’t export the class module.exports = Singleton; but we export the instance of the class module.exports = new Singleton() instead. Node.JS will cache and reuse the same object each time it’s required

class Singleton {  
constructor() {  
this.message = 'I am an instance';  
}  
}  
module.exports = new Singleton();

2-Observer pattern

Use cases ? why u used them ? how to Implement ?

- Async parallel calls Flow control

[1,2,3,4]

Promise.all

* How to block node event loop? Ans = infinite while loop
* Wat if I have event loop blocked but still wanna keep my server responding ?   
  Use child process or clustering

<https://medium.com/tech-tajawal/clustering-in-nodejs-utilizing-multiple-processor-cores-75d78aeb0f4f> -- discuss wid rana Function setupWorkerProcesses

Event loop ? https://developer.mozilla.org/en-US/docs/Web/JavaScript/EventLoop  
wat is round robin algorithm ?

Transpile vs compile

Y use middleware in node ?  
wat is middleware ?  
- Closure

Internal function gets exposed to parent function’s   
  
-Hoisting  
  
Code Review

How express handle request ?  
  
- Express  
  
- middlewares  
-micro-services  
  
-internal working  
  
-multi-threaded

Node cluster with pm2 ????  
  
-blocking non-blocking  
  
-code-output  
  
-challenges faced

-databases  
  
- ES6/ES7 (async, spread)

Arrow vs normal function? Argument keyword is not in fat arrow function(it ll b of outer functions and its own) !

Wat r u working on these days ? node details in answer .

AWS ?? EBS ? deploymenta ? FE ? BK ? DB ?

**Events in react ?**   
All of React events are what they call Synthetic Events:

* It handles event pooling
* It handles removing event listeners for you
* It’s better optimize
* It’s more efficient on DOM resolution, and event delegation

**Postgre vs SQL**

postGre is Object-Relational Database

- micro-services

Is there diff between V8 browser and V8 server side

memory management in nodejs

**provider in angular ?**

**over riding vs overloading**

**multiple inheritance in react ?**

**abstract vs static class ?**

**garbarge collector in JS ?**

JavaScript, utilize a form of automatic memory management known as [garbage collection](https://en.wikipedia.org/wiki/Garbage_collection_(computer_science)). The purpose of a garbage collector is to monitor memory allocation and determine when a block of allocated memory is no longer needed and reclaim it.

Circular references are a common cause of memory leaks.

[Mark-and-sweep algorithm](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Memory_Management#mark-and-sweep_algorithm)

This algorithm reduces the definition of "an object is no longer needed" to "an object is unreachable".

This algorithm assumes the knowledge of a set of objects called *roots.* In JavaScript, the root is the global object. Periodically, the garbage collector will start from these roots, find all objects that are referenced from these roots, then all objects referenced from these, etc. Starting from the roots, the garbage collector will thus find all *reachable* objects and collect all non-reachable objects.

**Cycles are no longer a problem**

**preventDefault vs stoppropagation ???**

https://developer.mozilla.org/en-US/docs/Web/API/Event/preventDefault

passport module ??

classes in javascript ??

**static variable = memory allocates at time of creation   
internal class variable ?**

The **static** keyword defines a static method or property for a class. Neither static methods nor static properties can be called on instances of the class. Instead, they're called on the class itself. Static methods are often utility functions, such as functions to create or clone objects, whereas static properties are useful for caches, fixed-configuration, or any other data you don't need to be replicated across instances

class Car {  
  constructor(name) {  
    this.name = name;  
  }  
  static hello() {  
    return "Hello!!";  
  }  
}  
  
let myCar = new Car("Ford");  
  
// You can calll 'hello()' on the Car Class:  
document.getElementById("demo").innerHTML = Car.hello();  
  
// But NOT on a Car Object:  
// document.getElementById("demo").innerHTML = myCar.hello();  
// this will raise an error.

framework vs library ?

ReactJS 16.8- this introduced hooks

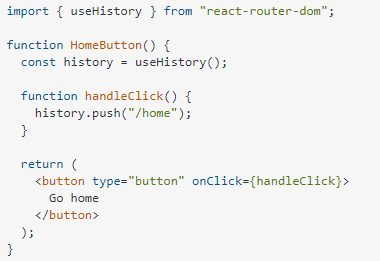
getDerivedStateFromProps vs componentWillReceiveProps?

Component life cycle methods ?

Hooks ? useState useEffect

Functional vs class based component ? use cases for both of them .

Routing in react ?



How to deploy react app ? index.html ka path use hota after build

Pure component ?

HOC ? connect . . .? use cases for HOC . examples ?

Virtual dom ? internal working ?

Controlled vs uncontrolled component ?

Babel vs webpack ?

middlewares in react ? thunk etc. . . .

ContextApi ? y use it ?

Can be used to share global data . like current user , theme etc . .

redux ? y use it ?