**JAVASCRIPT**

1. **Hoisting:** When the JavaScript code get compile for variable and function that get hoisted on top of the scope. Means all the declaration part moving on the current scope. Basically it is default behavior of the JavaScript.

When variable defined with let and const are hoisted to the top of the block, but not initialized.

Using a let variable before it is declared will result in a ReferenceError.

Store function can not hoisted.

1. console.log(test); // undefined with var

var test;

1. console.log(test); // undefined with reference error cannot access test before initialization with let.

let test;

1. greet()

let greet = function() {

console.log('Hi, there.'); // cannot access greet before initialization but with var greet is not defined with var

}

1. greet()

function greet(){

console.log(“Hi There”) // functions are hoisted

}

1. **Closures**

In the JavaScript, for example when we have two function like top function and nested function inside it. If we defined a variable inside of main function. Then we can call that variable inside of nested function after execution.

We can say that closure is an inner function that can call outer function’s variable but outer function can not access to the inner function variable that is called closure.

1. **Promises async await**

**Promise:** Promise is an object that allows us to perform asynchronous operations. Promise has executive function. Which execute some code. It behaves like a function it takes two arguments resolve and reject. If the function throws an error the promise will be rejected and if the function throws a fulfilled value the promise will be resolved and call to the callback function .then().

Executive function are synchronous.

BUT Resolve, reject, callback and .then() .catch() are asynchronous functions.

Suppose We did API call and that API will take some time after the API call we will do resolve then will call callback function.

**What is need of async and awaits** : await function to pause its execution and wait for a Promise to resolve before continuing.

**Async: Async** basically return a promise.

**Awaits:** This will pause the async function and wait for the code like API and other code would be execute until the response from the endpoint.

1. **Synchronous VS Asynchronous**

**Synchronous:** Synchronous means code will execute one by one like line by line.

**Asynchronous:** Asynchronous means for example which takes some times like API call because we don’t know when will be execute the API. Asynchronous is a non-blocking architecture, which means it doesn’t block further execution while one or more operations are in progress. With async programming, multiple related operations can run concurrently without waiting for other tasks to complete. We can do a multiple things at the same time.

For example, users want their apps to run fast, but fetching data from an API takes time. In these cases, asynchronous programming helps the app screen load more quickly.

1. **What is Event Loop and what is does it basically?**

What does event loop so It takes of our execution of whole JavaScript code so basically all the synchronous code they will be put one by one into the main thread and all the asynchronous code for example set Timeout, promises and callback functions those will be put into the callback queue and event loop will be checking the main thread and the callback queue once the main threads is empty right it will be start putting all this callbacks from the callback queue into the main thread that have some priorities task suppose promises is high priority and set Timeout is low priority.

Promises goes into the microtask queue and set Timeout goes into the event queue so event loop gives priority to the promises first and then set Timeout.

1. **What is prototype in JavaScript?**

Every object has private property that has to be link to the different object that is call prototype.

In JavaScript we achieve the inheritance by prototype. **Prototype property** is basically an object (also known as Prototype object), where we can attach methods and properties in a prototype object, which enables all the other objects to inherit these methods and properties.

For example create an object using function constructor.

// function constructor

  function Person(name, age, school) {

    this.name = name;

    this.age = age;

    this.school = school;

  }

  // this will show Person's prototype property.

  console.log(Person.prototype);

1. **What is Memoization in javascript?**

Memorization is a technique where the return value of a function is cached based on its input parameters.

Memoization uses the concept of closures where we can memoized a value. For example I am doing a API call

And it kinds of a get request not that things are repeating or changing. Okay so we can cached this value this get request like response I can cached and next time someone call again this get request, I can first check if it is present with us and directly use that instead of calling API. So this can achieve by memorization. Basically it boost the performance and optimization of our application.

And one more other service available to achieve this memorization that is call redis that keeps the key and value. So it’s a memory data structure.

1. **What is event bubbling?**

Event bubbling means whenever we click on element in DOM right so this time periods that onchange or onclick this will be moving up from child to the parent and then to parent and then to the parent so basically is There are two phases in event bubbling bubble phase and capture phase when call bottom to the top that is call bubble phase and we call top to the bottom that is called capture phase.

For example we have a navigation bar and we have 5 links so on each link I want to do a certain a same task

Right so there are a two methods one can be we adding all links in same function. Like onclick do this, onclick do this. 5 links we can add one by one. Or we can add directly into the parent. So what will happen this flow go to the parent only the code will be executing basically we don’t have to repeated 5 times. And suppose we have 6th link add so that would be handled automatically because everything handled by parent.

To stop event bubbling – e.stopPropagation().

1. Higher order function it takes the function and return the function for example map, reduce and filter because these all accept the callback.
2. **What is the use of ‘THIS’ keyword?**

* So This keyword always works like calling side. Suppose inside a function in a global space then this keyword works if we console this will refer to the window object.
* If THIS use in the object so this will refer to the owner object.
* This in a function (strict mode) this would be undefined.
* In a arrow function, so arrow function does not have any own this, so inside a arrow function THIS will point out to the parent object.

1. **What is http request?**

It is a hypertext transfer protocol, So basically HTTP protocol is based on request/response. So A client a connection with a server and sends a request to the server in the form of a request method, URL.

We have some http request methods: 1. GET, POST, PUT and DELETE as well.

1. **Why should not be use index as a key in the list of array?**

So suppose we have an array and have 3 value and we want add one more value in the end of the array so it will render only one list in our react component right. But in case if we put any value in the start of the array on that case the index number of all values will be change right. So in the second scenario the list will be loaded that’s why we should not use of index as a key so we use unique id.

1. **Can you explain the concept of memorization and its importance?**

So memorization is a technique in which the previous result or results of a function is cached so whenever we are making, we do some kind of computation or for the future purposes we can use a passed pre our evaluated data that is called memorization.

1. **Primitive and Non primitive data types?**

**Primitive**: Number, string, Boolean, undefined and null **Non-primitive:** Object, Array and RegExp (Regular Expression).

1. **\*\*\*\* Var**: Variable can be declared with the var keyword can have global scope and access anywhere in the page.

Variable declared inside a {} can not be accessed from outside the block.

**Let**: Let and const both are block scope variable. Let variables can be updated but not re-declared.

**Const**: If we declared const then we can’t change its value after declaration neither be updated.

Const a = 10;

Function f1(){

Const a = 9;

Console.log(a)//you can not re-declared

}f1()

1. **What is Scrum Master?**

*Scrum* is a framework that helps teams work together

The scrum master is the team role responsible for ensuring the team lives agile values and principles and follows the processes and practices that the team agreed they would use. The responsibilities of this role include: Clearing obstacles. ... Protecting the team from outside interruptions and distractions.

1. **GET & POST:** Both methods used to transfer data from client to server in http protocol.
2. **TRY and Catch: *Try*** is used to test a block of code for the errors and ***Catch*** used for handle the error.
3. **Set Timeout**: Runs the code/function once after the timeout.
4. **Set Interval:**  Runs/code/function repeatedly, with the length of the timeout between each repeat.
5. **What are the ES6 Features:**

Promises, Templates literals, Let and const, Rest and Spread Operator, DE structuring, Symbols and arrow function etc.

1. What is Stateless and stateful component:

**Stateful components** are keeping track of changing data, while **stateless components** print out what is given to them via props, or they always render the same thing. Stateless means doesn’t have any state.

1. **What is JSON?**

Json is format for storing and transporting data / JSON is used when data is sent from a server to a web page.

1. **MAP method?**

A map is data collection type where data stored in the form of pairs. It contains unique key. We can’t store duplicate pair in the map method. Map method creates new array by calling a provided function.

1. **REDUCE** **method?**

It reduces all the elements of the array to a single value by repeatedly applying a function. It is an alternative of using a loop and updating the result for every scanned element.

1. **FILTER Method?**

It filters the elements of the array that return false for the applied condition and returns the array which contains elements that satisfy the applied condition.

1. **What is Key in React?**

A key is a special string attribute used when creating a list of elements. So basically is used for identify which item should be updating, deleting and adding in the list.

1. **What is React Router**? React router is powerful library. React router keeps UI sync with the URL. It allows us to build a single page web application with navigation without the page refreshing.
2. **What is difference between React and React DOM?**

React is used for creating a component while React DOM is used for rendering a component in the DOM.

1. **Arrow Function**- Arrow function is used for writing a function expression and concise the syntax of code.
2. **What is a synthetic event?**  A wrapper based on the browser's native events. Including stopPropagation and preventDefault().
3. **React Lifecycle method**.

**Mounting**:  
There are four built-in lifecycle methods that are called in the following order when a component is mounted:  
**constructor( )** - This is called before anything else. We can set the initial state of the component inside this method. The constructor method is used to set the initial state and bind methods to the component.  
**getDerivedStateFromProps( )** - This is called before rendering the elements in the DOM.  
In this method, we can set the state of the component based on the props we received. This method is used very rarely.  
**render( )** - This is the only required method in the class component. This method returns the HTML elements which are going to be rendered inside the DOM.  
**componentDidMount( )** - It is called right after the component is rendered inside the DOM. All the statements which require the DOM nodes can be executed in this method. Network requests from a remote end-point can also be instantiated in this method.

**Updating**:  
Updates in react are caused by changes in state or props. Update leads to re-rendering of the component. The following methods are called when a component is re-rendered:  
**getDerivedStateFromProps( )** - This method is called again when a component is being re-rendered.  
**shouldComponentUpdate( )** - This method is called before rendering the component when new props are received. It lets React know if the component’s output is affected by the newly received props or by the state change. By default, it returns true.  
**render( )** - To re-render the HTML inside the DOM, the render( ) method gets called again.  
**getSnapshotBeforeUpdate( )** - This method is called just before the newly rendered HTML gets committed to the DOM. It stores the previous state of the component so that React has an idea of what parts of the DOM needs to be updated.  
**componentDidUpdate( )** - It is called after the component gets re-rendered. This method works just like the **componentDidMount( )** method, the difference is that this method does not get called on initial render.

**Unmounting:**  
**componentWillUnmount( )** - This method is called just before the component gets destroyed. Any clean up statements should be executed inside this method. This is used for clear up the memory.

1. **Pass the data from child component to parent.**

There are few steps to pass the data from child to parent by using callback function.

* Create two components parent and child component and import child component to the parent component.
* Create a callback function in the parent component and button to trigger that function.
* And pass the callback as props to the child component.
* Send data from the child component using the callback.
* All this things handle by useState hook to manage the data. See example.

1. **How to pass data from one component to another components?**

So we can use props to pass one component to another but we have a multiple component suppose A, B, C, D and we pass the data from A to D in this case we can use the useContext and we can keep the data in global store and we can access from global store by using context.

* So there are three things in useContext to send the data

1. Create Context b) Provide the context c) consuming the context

Example: Will create the context in main component suppose we want to pass the value as name and age so we will create a context for those values and Provide the that values suppose name.provider and will that value as props and will import that A component inside of name.provider and after that we will import that create context values in the C component and will keep those value inside of useContext and then pass it.

1. **How can we handle situation in suppose want to pass data from A to anywhere? How to handle it?**

In our projects we use the context and keeping the state or whatever settled functions directly in the context and this context will be available in all the components. So we can directly pass to the child.

1. **What is Props drilling?**

Props drilling means passing the same data to all components while some components event don’t need like this but we still passing it. This can make code details un-maintainable as well because it’s difficult to track like this. What is going. In production application like there are so many components and all.

1. **What is better to use context API or redux?**

So basically both are used for state management but context API use for smaller project and redux uses for larger and complex application. For example the theme of website that doesn’t change very often right so we can use context and suppose we have cart state which change is very often and every step and changes like products adding and discount added so to handle this situation we can use redux.

1. **What are limitation of React?**

React is not a framework it’s a library it does not come hard and fast rules to do this things you can do any base like you can api call wherever you want. You don’t have any follow structure. For example angular is that is framework that brings own rules that defined the services here and call the services this way only. But react brings doesn’t like this.

So this was advantage of react. It is easy to learn.

And limitation of react

1. Lack of proper documentation b) development speed c) JSX complexity and Problem with SEO.
2. **Controlled and Uncontrolled components in the react?**

So basically **controlled component**, that is form data handled by the react. And **Uncontrolled components** that form data is handled by the DOM itself. Suppose we want to handle the form data of input so we can use onChange function to handle this but in **uncontrolled component** we can use useRef hooks and ref instead of onchange to handle this. That is called controlled and uncontrolled components.

1. **Tell me the optimize the React app performance what are ways?**

So, In the class component we had use of Pure component to boost the performance but in the function component we can use UseMemo and useCallback function to the boost the performance of the application because React.Memo and UseCallback won’t re-render the component. It will work as memorization in the application. Object and arrays we can use UseMemo and for the function we can use UseCallback . Because they will take the value from Cache.

And we can use lazy loading: it’s a technique used to reduce initial load time of react app. Lazy loading helps reduce the risk of web app performance to minimal.

1. **Error boundaries in React?**

So keeping the error boundaries in whole app, for example if the API or UI fail then it will show like its not available, something went wrong some error message like that. Change the API completely, Unmount the app and mount the error boundaries. So in the class component we can handle by **ComponentDidCatch** to handle and remove the particular error from component or DOM. If we are using a function component then we can use one NPM package which is **react-error-boundary** to catch the error from component and remove that error.

So we can use the backup and loggers like what the errors are coming.

1. **What was the challenging situation that you faced in the last project in react application.**

No, I was never challenged for any such kind of position or things in my life. I expect it.

1. **Advantages and disadvantages of React js?**

**Advantages**

* React is faster and It takes less time. So react js much faster than vanilla js.
* Easy to learn. Proper documentation.
* So most important advantage of react Virtual DOM makes faster to react JS. So virtual DOM is a copy of real DOM. So whenever any changes into the component or state so that would be change from the virtual DOM and from the real dom. This is the whole process name is reconciliation. So this technique makes faster to the react.

**Disadvantages**

* So React js is not a Framework it is a library. Because we need to install the packages like react-router, state management like redux and all. Because in next JS have the routing at there.
* Other disadvantage of SEO problem.

1. **What is style component in react JS?**

Styled-components is a library that allows you to write CSS in JS while building custom components in Reacts.

1. **What is the reconciliation in the react JS?**

So whenever any change in the react component for the UI part so react create a virtual copy of that previously already have created a copy of that whenever the change happened then recreates a new copy of that with the help of most optimized approach creates a new DOM structure it only changes that part where the changes happened so it’s not re-render entire original DOM structure but the changes in specific part with the help of some optimization technique which react uses in the backend so this whole process called reconciliation.

1. **What is difference between CSR and SSR?**

**In the client side rendering** whenever user makes request server sends the bundles of JavaScript file and along with minimal html document file where the content is almost empty on the server side but once it is pushed on the client side where the whole creation of html document is with the help of JavaScript and from there whole html is created and then it is passed by the browser.

**In the server side rendering** the entire html document is created for by the server which is send to the client. But they have there off side and down side. So if you love me then I can explain it. So in this case of server side rendering the SEO will be more optimized because once we are sending to the client side the entire documents already created so that will help the SEO to scan throw the HTML and take up the content where the website is rendered. But in the case of client side we are sending the only empty HTML so that will not be optimized for the seos. So for the advantage of server side rendering.

1. **What are limitation of error boundaries?**

Error boundaries can not catch all the errors because they can’t detects the mistakes in event handlers, asynchronous programming or the server side rendering. We can use in class base component by using **componentDidCatch(error, errorInfo)** but we can’t use directly in function component so we can use third party libraries like **useErrorBoundary** by npm.

1. **Suppose do you have large amount of data like 10000 emp data if you render the data on browser then the browser will be crashed. So how can you manage it in the react?**

So there is a technique that is called virtualization so there is a library in the react called **react.window** so React window works by only rendering part of a large data set without crashing a browser.

1. **React is one way or two way data binding or unidirectional ?**

Unidirectional data flow describes a one-way data flow where the data can move in only one pathway when being transferred between different parts of the program.

React, a JavaScript library, uses unidirectional data flow. The data from the parent is known as props.

1. **Props are mutable or Immutable? And what is mutable or immutable?**

The props of a react component is aimed to store values and functions from it's parent component. It's just the pattern, props are immutable. Props are immutable.

**Immutable means:** You connote modify the props it’s read only

**Mutable means**: State is mutable so You are able to modify state using setState

1. **What is main purpose of react? Or why we use react ?**

It has huge community on Github. And it is component based architecture.

React allows developers to create large web applications that can change data, without reloading the page. The main purpose of React is **to be fast, scalable, and simple**. It works only on user interfaces in the application. Its aim is to allow developers to easily create fast user interfaces for websites and applications alike.

1. **When we updated 1000 of components then how many virtual Dom are available.**

React JS uses just **one** virtual DOM for operation. At any given time, ReactJS maintains **two** virtual DOM, one with the updated state Virtual DOM and other with the previous state Virtual DOM.

1. **What is single page application? And how its working?** A single-page app reloads only the data necessary for the user. The idea is that the website loads all the HTML/JS the first time you visit. When you then navigate, the browser will only render the content without refreshing the website.
2. **What is render and return in React?**

**In react, render is a** method that tell react what do want to display and where we can display. return in a method or function is the output of the method or function or display the output.

1. **can we change props in react?** No we can’t change or modify props inside the component as they are immutable.
2. **What is a REST API example?**

For example, a REST API would **use a GET request to retrieve a record, a POST request to create one, a PUT request to update a record, and a DELETE request to delete one**. All HTTP methods can be used in API calls. A well-designed REST API is similar to a website running in a web browser with built-in HTTP functionality.

1. **How we can handle the web api call , while so much data from api coming, like infinite data coming from the backend, how we can handle in front end side?** Using Pagination, Infinite scroller, using server side rendering.