1. => Context , HOC
2. Who is better? Fun or class?

The use of prop destructuring makes it really beneficial to see what's going on and what's coming out of the component.

There is an opinion that functional components show a greater performance compared to class components. The point is that the React functional element is a simple object with 2 properties: type(string) and props(object). To render such a component React needs to call the function and pass props – that is all.

Class components are more complex: they are instances of React.Component with the constructor in it and complicated system of methods for manipulating state and lifecycle.

2. Virtual DOM

“virtual DOM object.” A virtual DOM object is a *representation* of a DOM object, like a lightweight copy.

Once the virtual DOM has updated, then React compares the virtual DOM with a virtual DOM *snapshot* that was taken right before the update.

By comparing the new virtual DOM with a pre-update version, React figures out *exactly which virtual DOM objects have changed.* This process is called “diffing.”

Once React knows which virtual DOM objects have changed, then React updates those objects, *and only those objects,* on the real DOM. In our example from earlier, React would be smart enough to rebuild your one checked-off list-item, and leave the rest of your list alone.

3.preventDefault is called on the event when submitting the form to **prevent a browser reload/refresh**.

preventDefault() prevents the browsers default behaviour, but does not stop the event from bubbling up the DOM. The event. stopPropagation() prevents the event from bubbling up the DOM

, but does not stop the browsers default behaviour.

With stopPropagation, only the **button's click handler** is called while the **div's click handler** never fires.

Where as if you use preventDefault, only the browser's default action is stopped but the div's click handler still fires.

## **How do you prevent nested attack on GraphQL server?**Query validation · Query timeout · Query whitelisting · Query cost limiting

## **Is it possible to use inheritance with GraphQL input types?No, the spec does not allow input types to implement interfaces. And GraphQL type system in general does not define any form of inheritance**

## **How to respond with different status code in GraphQL?**Using try catch

## **How would you model recursive data structures in GraphQL?**

type Query { tags: [Tag] }

type Tag { id: ID! children: [ID] root: Boolean }

{ "tags": [ {"id": "1", "children": ["2"], "root": true}, {"id": "2", "children": [], "root": false} ] }

Using filter

Que : **How to implement a set of GraphQL mutations in single transaction?**

**1. What is DOM?**

**=> Dom represents document object model**

**=> Dom is programming interface of web document**

**=> DOM represent web document of html or xml**

**=> Web document contains object,attribute and properties**

**=> Web document object can be created,deleted and modify and manipulated data**

**=> Trigger events also in web document**

**2. What is Virtual DOM?**

**=> Virtual dom is a copy of DOM**

**=> Virtual dom update faster compare to DOM**

**=> if data update in virtual dom, virtual dom compare current state and previous state and then virtual dom updated batch send to dom to update UI**

**=> it’s don’t memory wastage**

**=> virtual dom can’t update directly HTML**

**=> Virtual dom can directly updated JSX**

**3. What makes react faster?**

**=> Virtual dom makes react faster**

**=> in virtual dom if data changes then re-render component**

**=> virtual dom don’t waste memory**

**4.types of requests**

**=>GET : is use for retrieve information**

**=>Post: is use for submit information to server**

**=>PUT: put can update the data**

**=>PATCH:patch is similar to put but in patch partially data update without modify entire data**

**=>Delete: delete use for remove data**

**5.What do you use for testing and how to write a test?**

**=> First of write test case for entire application**

**=> secondly manual test and write test case of app for user perspective**

**=> than auto testing for written test script**

**=> in test case write validation test case and other**

**5. How to clone a project using git?**

**=> in git copy the http link and paste in command line**

**Git clone \*\*http project link**

**6. How to clone using ssh?**

**=> in git copy the ssh link and paste in command line**

**Git clone \*\*ssh project link**

**7. What are the different hooks you have used and where to use them?**

**useState => useState useful for state store and update state using setstate. Store information and get info from the state**

**useEffect => useEffect mostly used for lifecycle method in useEffect and useEffect render when app re-render and state update**

**useMemo => it use for optimize calculation and application performance increase**

**useRef => useRef similar to useState but in useRef data change in trigger component don’t re-render**

**useCallback => in function return another function logic uses the useCallback**

**useContext => uses the access common data for entire app in usecotext no need pass the props.**

**8. different ways to pass data from one component to another?**

**=> pass the data parents component to child through props**

**=> pass the data child to parent component using callback function in parent component**

**=> Global state store and and data retrieve in sibling component**

**9. When to use redux?**

**=> when large amount of state use in many component need to store in one store then use redux**

**=> in redux action dispatch and reducer find updated state and store in redux store and ui get retrieve info from redux store**

**10. What is webpack?**

**=> Webpack is a javascript compiler module**

**=> webpack produce the small build**

**=> build include the js file css file and assets file like image**

**11. What is babel?**

**=> Babel is javascript compiler**

**=> Babel convert new verion js to old verion js convert**

**=> Babel convert jsx to js file**

**13. what happens when you do setstate in the constructor?**

**=> it’s return error can update only mount or un mount component**

**14. what are different lifecycle methods?**

**=>componentwillMount: componentwillmount execute just before rendering and it’s server side and client side both**

**=>componentDidMount: componentDidMount execute after rendering and render client side**

**=> componentwillUnmount: componentwillUnmount execute after component unmount used clear memory**

**=> componrnWillupdate:componrnWillupdate execute just before rendering**

**=> componetDidUpdate: componetDidUpdate execute just after rendering**

**=>shouldComponentupdate: if compone update return true else false default return false**

**15. What is Redux?**

**=>Redux use in app when large state in many components**

**=> Redux store whole app data store in one store**

**Action: it’s describe what happend**

**Reducer: it’s describe what data to be changed**

**Store: store data in one store**

**View: retrieve data from store**

**16. API calls using promises**

**=> when many api call in parallel use the promise.all()**

**17. callbacks**

**=>One functio pass another function as a argument is called callback fun**

**18. prop types and ways to validate data**

**=>prop types use for validate data**

**=>whenever enter invalid variable type it display error**

**19. webpack working and use**

**=> Webpack is a javascript compiler module**

**=> webpack produce the small build**

**=> build include the js file css file and assets file like image**

**=>it’s used for single page app**

**20. What is JSX? how does JSX work on browsers?**

**=> JSX is a combination of js and html**

**=>jsx doesn’t work on browser**

**=> jsx convert into js through babel and then run in browser**

**21. how to call another API call on the success of another?**

**=> async await**

**22. what to do if there are multiple API calls? and I need to call one API on success**

**=>using promises.all()**

**24. Pure components**

**=> pure components render same output at same time and props and state are also same**

**=>ex .class component extend react**

**25. what is mapstateToProps and dispatchToProps**

**=> mapstettoprops receive state and props and allow you to extract props you pass state to the component**

**=>dispatchstatetoprops : resulting funcion get dispatched .receive state and props**

**26. what is the context?**

**=>uses the access common data for entire app in usecontext no need pass the props.**

**git stash temporarily shelves (or stashes) changes you've made to your working copy so you can work on something else, and then come back and re-apply them later on.**

**How the object loop => For in**

**diffrence between map and filter and reduce**

**=>Foreach takes a callback function and run that callback function on each element of array one by one.**

**=> The map() method is used for creating a new array from an existing one, applying a function to each one of the elements of the first array.**

**=>The filter() method takes each element in an array and it applies a conditional statement against it. If this conditional returns true, the element gets pushed to the output array. If the condition returns false, the element does not get pushed to the output array.**

**=>Also take notice filter does not update the existing array it will return a new filtered array every time.**

**=>reduce method of the array object is used to reduce the array to one single value.**

**var, let, const diffrence and the scope**

**what is suspense in react js**

**=>A component created using React.lazy() is loaded only when it needs to be rendered. Therefore, you should display some kind of placeholder content while th**

**e lazy component is being loaded , such as a loading indicator. This is exactly what React.Suspense is designed for.**

**React.Suspense is a component for wrapping lazy components. You can wrap multiple lazy components at different hierarchy levels with a single Suspense component.**

**The Suspense component takes a fallback prop that accepts the React elements you want rendered as placeholder content while all the lazy components get loaded.**

**react lazy compoent how it is work**

**Call,Apply,bind,**

**=>call : binds the this value, invokes the function, and allows you to pass a list of arguments. apply : binds the this value, invokes the function, and allows you to pass arguments as an array. bind : binds the this value, returns a new function, and allows you to pass in a list of arguments.**

**arrow function this vs simple function this**

**class component bind ?**

**HOC component what return**

**=> A higher-order component (HOC) is an advanced element for reusing logic in React components. Components take one or more components as arguments, and return a new upgraded component.**

**es5 and es6 diffrence what new function**

**redux value access the out side of the application bind**

**How do I access Redux outside of component?**

**=> If you need to dispatch actions from outside a React component, the same technique will work: import the store, then call store. dispatch() , passing the action you need to dispatch. It works the same as the dispatch function you get from props via react-redux's connect function**

**diffrence between spread oparator and rest oparator? and what is return**

**Spread => allows us to quickly copy all or part of an existing array**

**=>new array return**

**The spread operator makes deep copies of data if the data is not nested**

**Rest => The rest parameter syntax allows a function to accept an indefinite number of arguments as an array**

**why we use middleware in redux and how can be apply? using redux thunk=>Redux Thunk is middleware that allows you to return functions, rather than just actions, within Redux. This allows for delayed actions, including working with promises. One of the main use cases for this middleware is for handling actions that might not be synchronous, for example, using axios to send a GET request.**

**useCallback() and useMemo() diffrence and what is return in function=> The difference is that useCallback returns its function when the dependencies change while useMemo calls its function and returns the result.**

**how can we manage async and awit function error handing=>**

* **We can use try...catch for synchronous code.**
* **We can use try...catch (in combination with async functions) and the .catch() approaches to handle errors for asynchronous code.**
* **When returning a promise within a try block, make sure to await it if you want the try...catch block to catch the error.**
* **Be aware when wrapping errors and rethrowing, that you lose the stack trace with the origin of the error.**

**diffrance between export and export deafult**

**can you use jsx file without babel?**

**useReducers**

**useProviders**

**webpack**

**babel**

**context && how to access the context value?**

**javasript engine**

**JavaScript is not understandable by computer but the only browser understands JavaScript. So, we need a program to convert our JavaScript program into computer-understandable language. A JavaScript engine is a computer program that executes JavaScript code and converts it into computer understandable language.**

**\_\_scss file why we use \_\_ in scss file name**

**=>When we add “\_” in front of the SCSS file, it means that it is partial for SCSS. When the compiler gets any SCSS files starting with “\_”, it simply ignores the file. If you want this partial for SCSS to be included for styles, you must use @import. The advantage of using partials is that you can use style files to organize your code and all the files would be compiled on a single file. The purpose of partial SCSS is to keep your styles separated into logical sections.**

**In the end, we want one SCSS file to have complied whereas we can have many logical partial SCSS files.**

**What is Closure?**

**=>, a closure gives you access to an outer function's scope from an inner function**

**eslint intigrate with react?**

**react router dom**

**how can we restrict the function read only**

**const appConfig = {**

**siteName: 'Test Site',**

**key: 'ddkdkkdkd'**

**}; => Object.freeze()**

**appConfig.key = ""; // not accessible**

**output below things?**

**let office1 = {'developers': 'Web'};**

**let office2 = office1**

**office1.developers = 'Mobile';**

**console.log(office2.developers);**

**sequence of below thing?**

**setTimeout(() => {console.log('yaspal')}, 0);**

**console.log('test');**

**Promises / Async + Await examples and how can it works?**

**const users = new Promise((resolve, reject))**

**how can access this resolve and reject**

**What is Devdependencies?**

**devDependencies are modules which are only required during local development and testing, while dependencies are modules which are also required at runtime (that is during production)**

**Shadow DOM**

**Shallow Routing**

**What is generator function?**

**generators return multiple values in a sequence, one after another. They operate great with iterables and allow creating data streams with ease.**

**yield 1;**

**yield 2;**

**return 3;**

**}**

**let generator = generate();**

**let oneValue = generator.next();**

**console.log(JSON.stringify(oneValue)); // {value: 1, done: false}**

**2 objects comparison is false but how?**

**What are Local storage cons?**

**What is the Redux saga?**

**Difference between Promise & Callback & Await async**

**What is prototype**

**ci/cd deploy**

**Jsx js es5?**

**How to browser react using const arr = 7**

**How to convert es6 to older version**