Interview cheatsheet

Vue

To make a value reactive in vue 3 , use ref that wraps it inside value or reactive

Computeds are used to process a logic that needs to be shown in template and can track the changes made to the value they are returning. For example when you want to show different messages based on different values

Watchers are used for  "side effects" in reaction to state changes.

With watchers you can track changes to a value and decide what to do based on their changes

Create life cycle hook is initiated before do rendering, mounted is fired after dom rendering

Ref is used to directly access DOM elements and make changes to them

Props is used to pass values to child component, emits is used to notify parent if a change is made in child.

DefineModel is used for component v-models. It creates a two way binding between parent elemnt and local value inside component

Slots is used to give access to the template inside a child component.

Provide is used to write app level code that can be accessed every where in the app. With inject you can define where this app level code can be used

Define async component is used to lazy load a component.it has multiple options and fallbacks in case of errors,loading,etc

Composables is used to write reusable logic in the application

Plugins is like composables.

Teleport tag , teleports inner elemnts to the defined tag in it

For routing we use vue router, beforeRoute enter is an important function, and for state management we use pinia. State management means that we need to track a variable value in app level and globally

Nuxt

Nuxt uses file routing, you can define dynamic routes with [] and […all] is for not found pages.

For SPA you can use app.vue and for MPA you can use views.

Assets inside public dir is served in server and is optimized by nuxt

Assets dir is served in client.

For using app level styles you must let nuxt know which file to use with this

export default defineNuxtConfig({

css: ['~/assets/css/main.css']

})

You can write route middlewares inside middleware directory and if you append .global to the file it will be used after eachRoute. Else you must define for each page to use a middleware.

For general seo you can use nuxtConfig

app

: {

head

: {

charset

: 'utf-8',

viewport

: 'width=device-width, initial-scale=1',

}

}

For seo inside each page you can use useHead composable. Or useMeta composable.

Page transitions is defined in nuxt config like this

app

: {

pageTransition

: {

name

: 'page',

mode=

: 'out-in' }

},

For data fetching you can use either useFetch composable which uses un.js or use asyncData function that runs everything inside it, in server.

You can define app level error handling like this

nuxtApp

.

vueApp

.

config

.

errorHandler

= (

error

,

instance

,

info

) => {

// handle error, e.g. report to a service

}

// Also possible

nuxtApp

.

hook

('vue:error', (

error

,

instance

,

info

) => {

// handle error, e.g. report to a service

})

s

Javascript

Event loop: its like a train of function that uses each other values

Hoisting: javascript moves all variables to the top of the compiled code

IIFE: a function that calls it self

(**function**() {

    // IIFE code block

**var** localVar = 'This is a local variable';

    console.log(localVar); // Output: This is a local variable

})();

Closure : a function that holds and uses the value of another function inside it

function init() {

var name = "Mozilla"; // name is a local variable created by init

function displayName() {

// displayName() is the inner function, that forms the closure

console.log(name); // use variable declared in the parent function

}

displayName();

}

Promise:when you use a promise inside code, when the application runs to it, the thread will be locked and decides what to do based on promise rejection or resolve.

const myPromise = new Promise((resolve, reject) => {

setTimeout(() => {

resolve("foo");

}, 300);

});

myPromise

.then(handleFulfilledA, handleRejectedA)

Currying: is a method of coding , it is used for clean code mostly and is not necessary.

function curry(f) { // curry(f) does the currying transform

return function(a) {

return function(b) {

return f(a, b);

};

};

}

Type coercion : is the reason javascript can handle ‘1’ + 1

HOF: Higher Order Function is a function that receives a function as an argument.

s