Q1. What is the difference between Named Export, Default export and \* as export?

Answer. Named Export allows a developer to export multiple variables, functions or classes by name from a module. Each export explicitly has to be named. For example:

export const PI = 3.14;

export function add(a, b){

return a + b;

}

When importing named exports, we have to use curly braces and must use the same name as exported:

import {PI, add} from “./mathUtils.js”;

import {add as sum} from “./mathUtils.js”;

A default export is a single value (variable, function, class, etc.) that is considered the “main” export of a module. A module can have only one default export. It is declared with the default keyword:

export default function log(message){

console.log(message);

}

When importing a default export, you do not use curly braces, and you can choose any name you like:

import log from “./logger.js”;

log(‘ Hello, world! ’);

**Namespace Imports/Exports (\* as)**:

The \* as syntax is used to import all of a module’s exports into a single namespace object. It is particularly useful when you want to keep the module’s exports grouped together.

For example, if you have a module with several named exports:

export function foo() {/\* … \*/}

export function bar() {/\* … \*}

You can import all of its exports under a single namespace.

import \* as utils from ‘./utils.js’;

utils.foo();

utils.bar();

You can also re-export all exports from another module under namespace:

export \* as utils from “./utils.js”;

This type of import and export is useful for organizing imports or re-exporting from another module.

Q2. What is the importance of config.js file?

Answer: A config.js file is a text file which helps manage configuration settings for a project by making all the data centralized and organized in one file. Instead of hardcoding values like API URLs, Image URLs, database credentials, or environment specific settings throughout the codebase, we can create a config.js file where we can store all these in one place. This makes code easier to read, more manageable and easier to update.

Since config.js file contains some private information such as API Keys, credentials, secret tokens it is recommended to not push this file to remote repositories. Pushing such data can expose a project to security risk like unauthorized access, data leaks and potential hacking attempts.

It’s always better to include config.js file in .gitignore file to prevent accidental commits.

Q3. What are React Hooks?

Answer: React Hooks are like special functions that let us add features such as keeping track of data (state) and doing things when your component loads or updates, all from within function components.

Some rules:

* They are just functions, but they follow two rules
  + It can only be inside a function component
  + It is to be used at the top of the component, not inside loop or conditions.
* They let function components have their own state – With the useState hook, your components can remember things and change what they show based on that memory.
* You can share logic between components – By making your own hooks, you can take some logic about state out of your components and use it elsewhere. This is handy for things like fet