CHAPTER 05

LET'S GET HOOKED

1. **What's the difference between Named Export, Default Export and * as Export?**

   Basically while writing our code, in order to make the code more readable and less complex to our fellow devs , we ought to use import and export keywords for file transfer. There're some types to export through which we can add them in our target file. There're as follows-

   Named Export

```
export let Circus = () => (
<div>
 <h1> This is a Blockbuster </h1>
</div>
);
```

   Consider the above example and we're trying to export this component to our desired file and in that file to get this, we'll write

   import {Circus} from "./files/body";

   Here files- A folder in which our body file present.(module)

   This type of export is called Named Export. Because here we're directly named export to the element which we want to export and so in importing this file we must add title of the element we want to import inside curly braces **{ } , just like above.** The name of the import should be same as the name of the export.

   Default Export

```
Const Circus = () => (
<div>
 <h1> This is a Blockbuster </h1>
</div>
);
```

   export default Circus

   Consider the above example and we're trying to export this component to our desired file and in that file to get this, we'll write

   import Circus from "./files/body";

This is the Default export. Here, we just write export followed by default keyword. This export is not written directly with the component but specified separately. Here export Default is used to export a single class, function from script file. There can only be one default export.
When you export by default, you can rename the function/variable and use. But its always better to go without renaming.

A module can contain both named exports and default export and they can be imported using **import defaultFile, {namedFile1, namedFile2…} from "./module";**

*** as export**

If you want to export everything from the file we can just do it by mentioning " * " . and write

import * as Obj from module("./files/body";)

and write in the Code **Obj.Circus…**

👉 **https://stackoverflow.com/questions/36426521/what-does-export-default-do-in-jsx**

👉 https://stackoverflow.com/questions/54810022/how-to-use-multiple-export-default-in-react

2. **What is the importance of config.js file?**

If you're having multiple components and want an IMAGE/SOME FUNCTION in 20 components. We can't just always copy the URL of image / Whole function and paste it all the time right. So, to ease the process and code, we can import it directly into the required file if we've it in one central file and for that we use **config.js** file in the project we're working in.

We can export config.js file like **export const Image = "https:……";**
We can import config.js file like **import {Image} from "../config";**

Always prefer using Named Export So, it would be more specific in multiple files case.

Not only config.js. We can use any name as we wish. It doesn't matter with the name; it can be either constant.js or any.

👉 https://www.useragentman.com/blog/config-js-%E2%80%93-a-javascript-cofiguration-library/

👉 https://stackoverflow.com/questions/30568796/how-to-store-configuration-file-and-read-it-using-react

3. **What are React Hooks?**

React uses something known as Hooks. Hooks are nothing but functions. Yes, React Hooks are simple JavaScript functions that we can use to isolate the reusable part from a functional component. Every hook has a function for it.
Hooks let us split one component into smaller functions based on what pieces are related(such as setting up a subscription or fetching data), rather than forcing a split based on lifecycle methods.
Hooks let us use more of React's features without classes. Conceptually, react components have always been closer to functions. Hooks embrace functions, but without sacrificing the practical spirit of react.
Moreover Hooks don't require us to learn complex functional or reactive programming techniques and makes easy for users.

👉 https://reactjs.org/docs/hooks-intro.html
👉 https://www.javatpoint.com/react-hooks

4. **Why do we need a useState Hook?**

useState is the first hook in React. This useState Hook is used to create Local State Variables.

Syntax to create local state variables in React

Const [searchText , setSearchText] = useState()

Syntax to create local state variables in JS **Const stext**

This Hook comes from react library. We import it using a named variable. useState() hook returns an Array. [This array has first item as our variable name , second item is setFunction : Function to update the variable] and we call this as state variable. ,useSate() at the end is a function.

👉 https://reactjs.org/docs/hooks-state.html
👉 https://dev.to/aasthapandey/why-to-use-usestate-in-react-pkf
👉 https://stackoverflow.com/questions/53165945/what-is-usestate-in-react