



Assignment 05 - Let's get Hooked



Owner



Pankaj Kumar


1: What is the difference between Named export , Default export and as export?

- **Named export** - In named export, the function is exported as:

```
export <function>;
```

and imported as:

```
import { <function> } from '..';
```

. The function is exported inside  to other modules.

- **Default export**: In default export the function is exported as:

```
export default <function>
```

and imported as:

```
import <function> from '...'
```

- **as export**: * as export is used to import the whole module as a component and access the components inside the module.

for example: We have a module ABC.js and some components inside of it and a file named XYZ.js where we want to import the components.

ABC.js:

```
export const Comp1 = () => {...}
export const Comp2 = () => {...}
export const Comp3 = () => {...}
```

in **XYZ.js** we'll import then as

```
import * as Module from 'ABC.js'
```

Now we can use them JSX as:

```
<Module.Comp1 />
<Module.Comp2 />
<Module.Comp3 />
```

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

- `config.js` file is used for the hard coded values used in our application. We can use it to import a configuration inside any component without having to copy it over and over again.

3: What are React Hooks?

- `React Hooks` are basically JavaScript functions which are provided by React. Hooks have some special capabilities that are useful for the development, like managing state, memory etc.

▼ **React provides a bunch of standard in-built hooks:**

- `useState` is a React Hook that lets you add a state variable to your component.
`const [state, setState] = useState(initialState);`
- `useEffect` is a React Hook that lets you synchronize a component with an external system.

(OR) To manage side-effects like API calls, subscriptions, timers, mutations, and more.

```
useEffect(setup, dependencies?)
```

- **useContext** is a React Hook that lets you read and subscribe to context from your component.

```
const value = useContext(SomeContext)
```

- **useReducer** is a React Hook that lets you add a reducer to your component.

(OR) A useState alternative to help with complex state management.

```
const [state, dispatch] = useReducer(reducer, initialArg, init?)
```

- **useMemo** is a React Hook that lets you cache the result of a calculation between re-renders.

(OR) It returns a memoized value that helps in performance optimizations.

```
const cachedValue = useCallback(calculateValue, dependencies)
```

- **useCallback** is a React Hook that lets you cache a function definition between re-renders.

```
const cachedFn = useCallback(fn, dependencies)
```

(OR) It returns a memorized version of a callback to help a child component not re-render unnecessarily.

- **useRef** is a React Hook that lets you reference a value that's not needed for rendering.

```
const ref = useRef(initialValue)
```

(OR) It returns a ref object with a current property. The ref object is mutable. It is mainly used to access a child component imperatively.

- **useLayoutEffect** is a version of useEffect that fires before the browser repaints the screen.

```
useLayoutEffect(setup, dependencies?)
```

(OR) It fires at the end of all DOM mutations. It's best to use useEffect as much as possible over this one as the useLayoutEffect fires synchronously.

- **useDebugValue** is a React Hook that lets you add a label to a custom Hook in React DevTools.

```
useDebugValue(value, format?)
```

- `useId` is a React Hook for generating unique IDs that can be passed to accessibility attributes.

```
const id = useId()
```

- `useTransition` is a React Hook that lets you update the state without blocking the UI.

```
const [isPending, startTransition] = useTransition()
```

4: Why do we need useState Hook?

- `useState` hook is used to maintain the state in our React application. It keeps track of the state changes. When a component is re-rendered it changes the state of our component.