Episode 9 - Optimizing Our App

Why should we build hooks or use Single Responsibility Principal ?

→ Modularity
→ Reusability
→ Readability
→ Maintainability
→ Testable
→ Easy to Debug

→ Single Responsibility : Every function or class in React should have single responsibility. It is easy to test bugs

Components are functions & we can wrap up a small logic inside a function & use it anywhere. To keep reusable functions we can create a folder utils & import it from there

Modularity → we have broken down code into meaningful pieces to make it reusable & testable

How to create a Custom Hook ?

Use 'use' before the Hook name to identify it easily as Hooks.

→ Using Custom Hooks

```
const restaurant = useRestaurant (resId);
```

Hook is a utility function or helper functions

→ Create Custom Hooks

```
const useRestaurant = (resId) => {
        const [restaurant, setRestaurant] = useState(null)

    // Get data from API
        useEffect (() => {
            getRestaurant ();
        }, [])

    ... Rest of the code

    return restaurant;
```

A functional component is a function that returns JSX

Custom Hooks has its own reconciliation going on then it will update the DOM automatically

A normal function can't have reconciliation, it can't create a state variable that's why we use a Custom Hook

→ Creating a Custom Hook "Are you online or not"

```
const useOnline = () => {
    Const [isOnline, setIsOnline] = useState (true);

    useEffect (() => {
        window.addEventListener ("online", () => {
            setIsOnline (true);
        });

        window.addEventListener ("offline", () => {
            setIsOnline (false);
        })
    }, []);

    return isOnline;
}

export default useOnline;
```

→ Using a Custom Hook

```
const isOnline = useOffline ();

if (isOnline) {
    return <h1> Offline, please check your network </h1>
}
```

How to fake offline in browser ?
Chrome Dev Tools < Network Tab < Offline

You need to clean up event listener after calling when we go out of our event listener otherwise it will create a new eventlistener (Unmouting phase)

```
return () => {
    window. removeEventListener ()
}
```

Code

```
useEffect(() => {
    const handleOnline = () => {
        setIsOnline(true);
    }

    window.addEventListener("online", handleOnline);

    const handleOffline = () => {
        setIsOnline(false)
    }

    window.addEventListener("offline", handleOffline)

    return () => {
        window.removeEventListener("online", handleOnline);

        window.removeEventListener("offline", handleOffline);
    }
})
```

Parcel bundles up all the code and store it in 1 JavaScript file. In development, size of this file will be large but in production bundle the size will be small.

But large scale production apps cannot have all the code in 1 JS file as it will make the app slow.

What is Chunking / Code Splitting / Dynamic Bundling / Lazy Loading / On demand loading / Dynamic Import ?

Even if we have bundled our whole code in one file we can't load all of them at once as it will make the app slow so we have to split the code into smaller chunks. As only that part will load in the app while is currently being visited

For Lazy Loading a component:

```
import {lazy} from 'react';
const Instamart = lazy(() => import("./components/Instamart"));
```

It's a promise

This will not include the code of this file in the bundle. That file will load only on visiting that component

→ When load the component "on Demand" React tries to suspend it. For the first time, on visiting that component. It will suspend as it is not loaded yet due to lazy loading & react tries to render it.

To handle it we use Suspense

```
<Suspense></Suspense>
```

There will be a slight delay when the code split component is fetched, so it is important to display a loading state otherwise it will not render the page & show error

In routing,

```
{
  path: "/instamart",
  element: (
          <suspense>
              < Instamart/ >
          </suspense>
      )
}
```

→ Fallback
It is an attribute of suspense tag which will load a component until the lazy loads completes

```
<Suspense fallback = { <shimmer/> } >
      < Instamart/ >
</suspense >
```

Is chunking necessary for smaller apps?
No, it is important for big apps

Never ever lazy load inside a component because it will be lazy loaded after every render