React useMemo Hook

Consider a scenario where there's a function which requires a lot of time to execute (Approximate 2~3 sec)

➡ Now, whenever there's any state change the Parent as well as the Child components will re-render

➡ Re-rendering means all the functions that are defined inside the component will be re-executed

➡ Now, if the function is really simple for example, adding 2 numbers then there's nothing to worry :))

➡ But, what if the function requires to do some heavy performance intensive tasks such as filtering & sorting the array with 1000's of elements

➡ It means that the function gets executed whenever our component gets re-rendered

➡ It also means that UI updates to some other part of the application becomes slow due to the function execution. So, how can we overcome this ?

Well, you guessed it right.. By using the useMemo Hook. But, how to use it ?

➡ useMemo hook will memoize the return value of the callback function

➡ The useMemo hook takes 2 parameters. Let's see what the parameters are in the next slide

Here's the parameters that we need to pass to the useMemo hook ⬇⬇

➡ The actual callback function

➡ The dependancy array which will tell React when to re-execute the function

How useMemo works ?

➡ So, the function gets executed when the component mounts for the 1st time

➡ And for future re-renders component will not re-execute the function instead it will use the cache value (Stored by useMemo hook)

useMemo Syntax

➡ useMemo hook only executes the function if any one of the dependancy value changes

```
const memoizedValue = useMemo ( () => {...}, [] )
```

Do you find this post helpful then please do share this post with your connections ;))