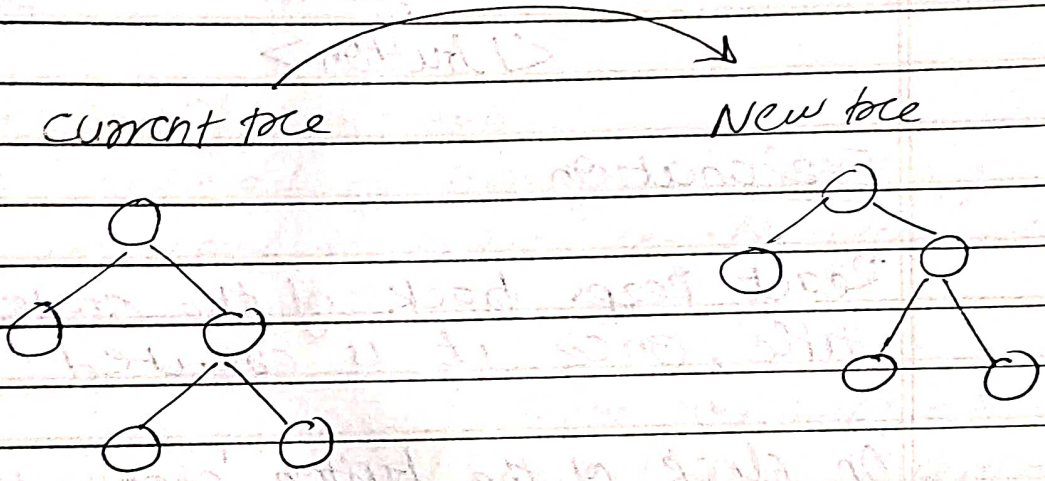
Chapter 06 Exploring the World

Page No. _____
Date

Why React is fast?

- Because it has Virtual DOM, Reconciliation, Diff algorithm
- In diff algorithm, current tree is compared with the new tree & the difference is reflected on the DOM
- React Fibre is the updated reconciliation algorithm



- React is fast because of its fast DOM manipulation
- Diff algorithm detect what exactly got changed in the page & it will just change that while re-rendering the whole tree

useState HOOK

React gives a state variable & a function that update state variable

eg :- const [title, setTitle] = useState("hi")

```
return (
  <div>
    <h1>{title}</h1>
    <button
      onClick={() =>
        setTitle("New Food App")}
    >
      Change Title
    </button>
```

Explanation

React keeps track of the state variable, title, once it is created.

On click of the button 'change title' the title get updated from 'hi' to 'New food app'. The whole UI will re-render & it will update the UI quickly.

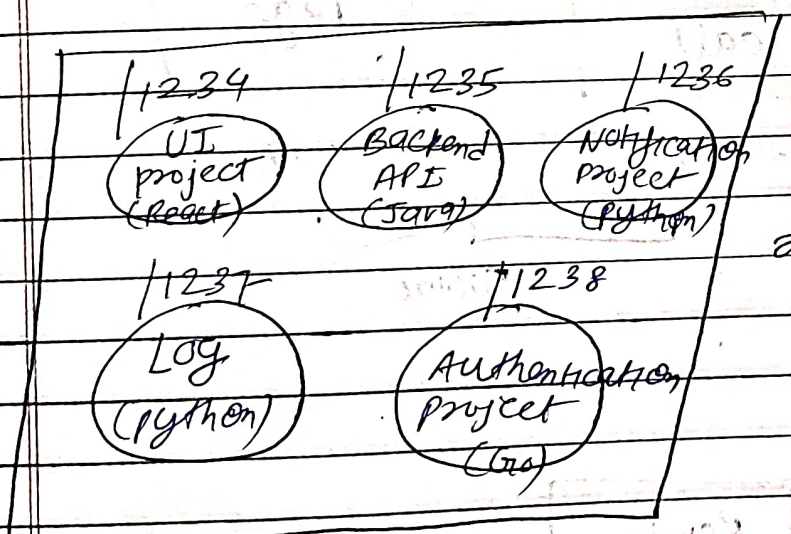
Whenever a state variable is changed, React re-fresh es or re-render the whole component.
And react do this super fastly.

Micro-Services

In older days, there used to be a single big application. So everything like API's, SMS, Notification, UI, JSP page etc used to be in same project. Suppose if we have to change one button we used to deploy this whole project / application. It was such a mess. This architecture was known as monolith.

Seperation of concern

But now instead of having a one big project, we used to have small different project:



There are seperate project here, seperation of concern or single responsibility is there.

The tools & language used in a project depends on the use case. All these project are depleyed in different ports but same domain name.

Page No.
Date

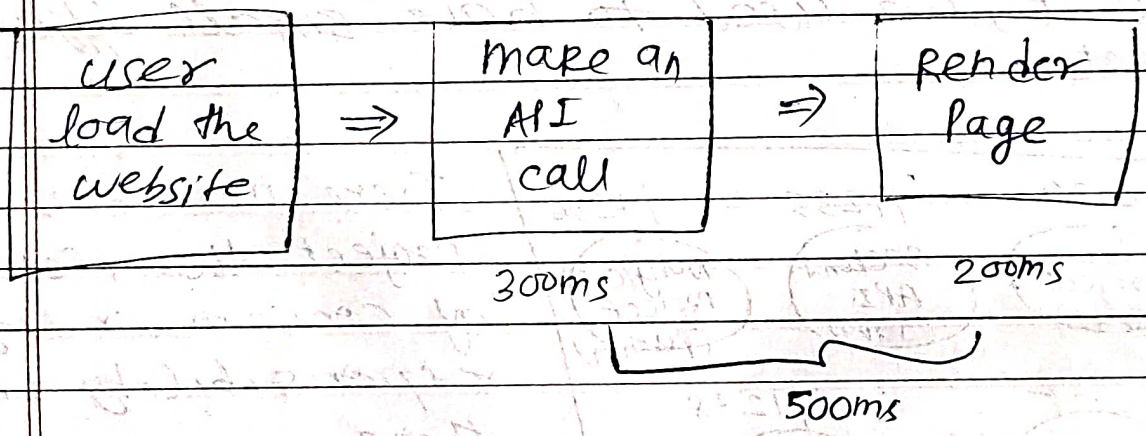
Best way to call an api

It should be like, as in when our body component load, it used to call on API and fill the data.

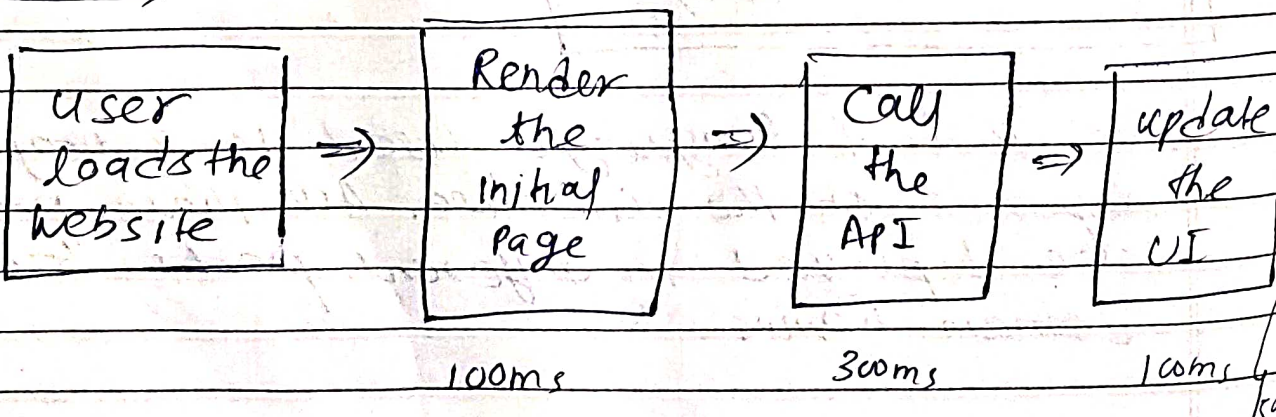
There is different ways to handle this in react :-

2 ways are :-

Case 1



Case 2



Page No.
Date

Case 2

is the good way. All this will happens very quickly to make this happen, react give us a hook : useEffect.

useEffect Hook

- We get this from react library.
- useEffect is a function. We call this function by passing another function to it which is a call back function

```
useEffect (callback function, dependency array)
```

- call back fn means 'this function' is not called immediatly but called whenever useEffect wanted to be called.

React will make sure that it is called in specific time

- Whenever the component renders & re-renders & re-render, first of all, the code of the component will be called & after every render it will call the call back function that pass inside useEffect()

→ There are 2 ways when my component re-render

(1) State change
(2) Props change

→ useEffect will be called on every re-render which is a bad way. If we don't want to call it after every re-render, pass in a dependency array into it

```
useEffect(() => {}, []);
```

[] → dependency array
() => {} → call back function

Role of Dependency Array

Eg: const [searchTxt, setSearchTxt] = useState("");

→
```
useEffect(() => {
  console.log("render");
}, [searchTxt]);
```

Suppose, I want to call this useEffect only when the "searchTxt" changes then I have to pass that "searchTxt" in the array

useEffect(() => {
  console.log("call this when dependency is changed");
}, [searchTxt]);

Whenever this searchTxt is changed, call this callback function.

Suppose, if I don't want my "callback fn" to be dependent on anything. It will be called just once (X)

And also it will be called after render

So, useEffect will be called just once & after initial render if there is empty dependency array.

Fetching real data

Api call happens asynchronously.

Let's create a async function

Async - await is the most preferred way.

Page No.
Date

* useEffect (() => {
getRestaurants(); }, []);

* async function getRestaurants() {
const data = await fetch("https/swiggy");
const json = await data.json();
consale.log(json);

setRestaurant(JSON.data.cards[2].data.data
.cards)

}

There are security parameters & browser will block us from calling that swiggy api.

Browser won't let us to call swiggy from our localhost.

To modify this, there is a plugin - CORS.

[Allow CORS: Access control allow origin]
Add this plugin to chrome.

This plugin lets you bypass the cors errors.

[watch - video - CORS - YT]

The old data will be rendered first for a few second & then new data comes.

If we removed that old data, page shows an ugly UI. So, initial screen to get rendered should be a loader/ Shimmer UI. That is a basic skeleton.