

Chapter 06 - Exploring the World

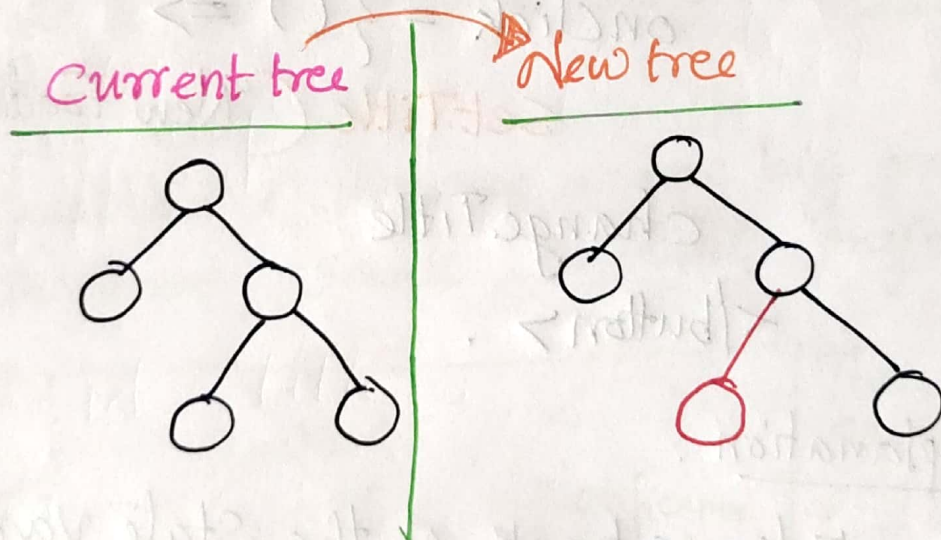
Date : 14/Jan/2023

Why React is fast ?

→ Because it has virtual DOM, Reconciliation, Diff algorithm.

→ In Diff algorithm, current tree is compared with the new tree and the difference is reflected on the DOM.

→ React Fibre is the updated reconciliation algm.



→ React is fast because of its fast DOM manipulation.

→ Diff algm 'detects' what exactly got changed in the page & it'll just change that while re-rendering the whole tree.

Use State Hook

React gives a state variable and a function that update state variable.

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

initial value.

return (

`<div>`

`<h1> { title } </h1>`

`<button`

`onClick = { () =>`

`setTitle("New Food App") } >`

`changeTitle`

`</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 and it will update the UI quickly.

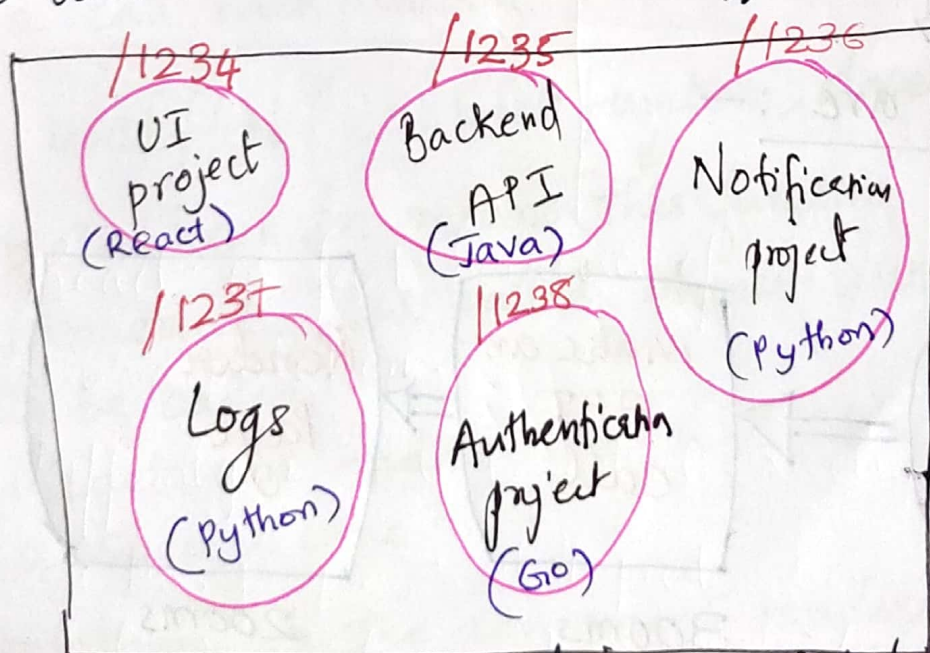
Whenever a state variable is changed, React re-freshes & or re-render the whole component. And react do this super fastly.

MICROSERVICES

In older days, there used to be a single big application. So, everything like APIs, SMS, Notification, UI, JSP pages etc used to be in same project. Suppose if we have to change one button, we used to deploy this whole project /app. 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 projects.



There are separate projects. Here, separation of concern or single responsibility is there.

The tools & languages used in a project depends on the usecase. All these projects are deployed in different ports but same domain name.

How to make an API call?

- Javascript gives us **fetch()** function
- Where do we call this api in our component?

On every ~~UI~~ ^{state} change or any time my UI is updated, this would get re-render if we called api just after or before the usestate declaration. This is not a good way.

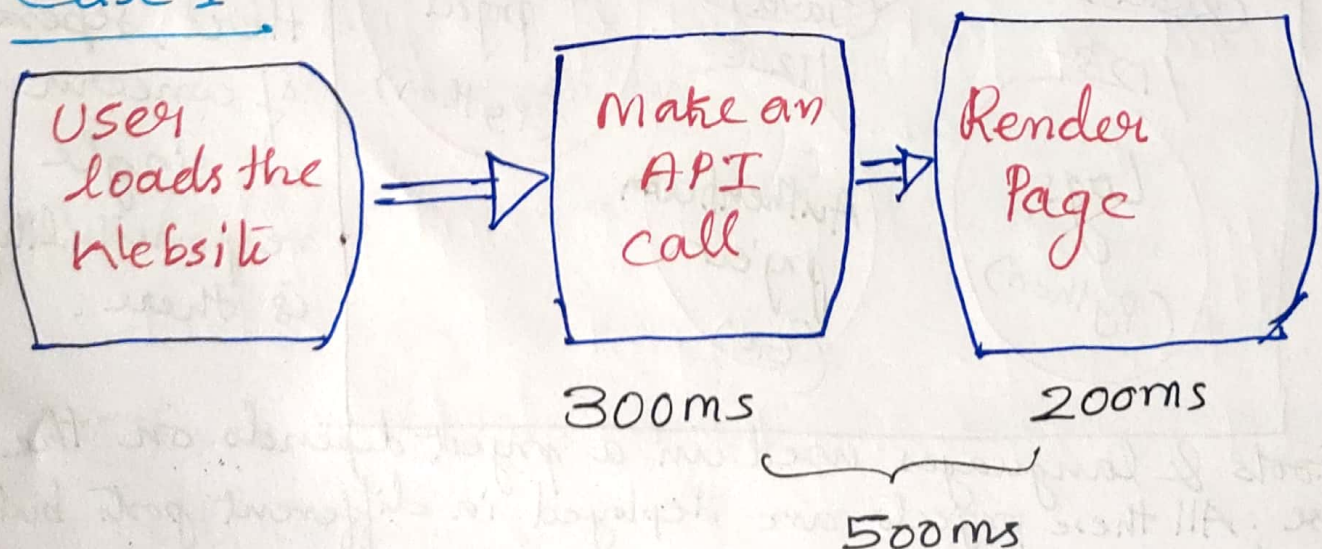
Best way to call an api?

It should be like, as in when our 'Body' component loads, it used to call an API and fill the data.

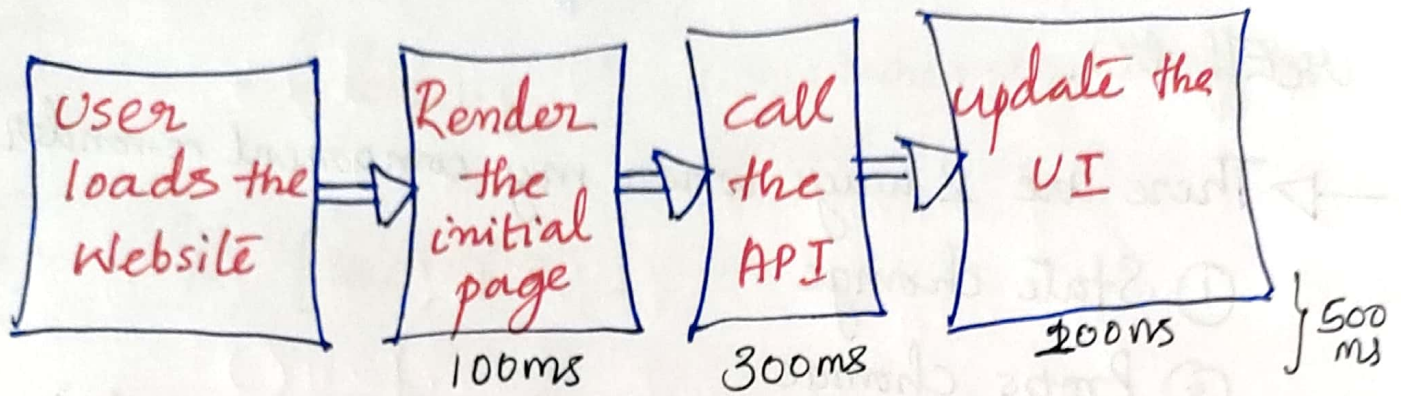
There is different ways to handle this in react : →

2 Ways are :-

Case 1



Case 2



Case 2 is the good way. All this will happens very quickly. To make this happens, react gives 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 immedietly but called whenever useEffect wanted to be called.

React will make sure that it is called in a specific time.

→ Whenever the component renders & re-renders & re-renders, first of all, the code of the component will be called & after every render,

it will call the callback fn 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");
```

```
}, [searchText]);
```

Whenever this `searchText` is changed, call this callback function.

Suppose, if I don't want my 'call-back' fn to be dependent on anything, It will be called just once ❌.

And also It will be called after render.

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

Fetching real data

API call happens asynchronously.

Let's create a async function. (A)

Async-await is the most preferred way.

* useEffect () \Rightarrow {
getRestaurants () : } , []);

* async function getRestaurants () {
const data = await fetch ("https://swiggy.in");
const json = await data.json ();
console.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

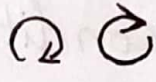
This plugin lets you bypass the CORS error

[Watch video - CORS - YT]

The old data will be rendered first for a few seconds and 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.

SHIMMER UI

There was a research done and earlier people used to saw 'Spinning' loaders  at first and then suddenly every restaurant's come up.

This is a bad user experience. X

Human brain don't like to view so many fluctuations in the UI, according to psychology. Psychologist figured out that, instead of spinners empty boxes should be shown. It is a better UI experience then for the users ✓

As suddenly changes are not happening, our eyes won't hurt

This is known as SHIMMER UI

Shimmer UI resembles actual UI, so users will understand how quickly the web or mobile app will load even before the content has shown up. Every big company is following this.

CONDITIONAL RENDERING

- Same as conditions (if operator or the conditional operator) work in Javascript.
- Your components will often need to display different things depending on different conditions.

→ In React,

you can conditionally render JSX using javascript syntax like :-

- if statement
- && operator
- ? : operator

→ Here, we need to render either a Shimmer UI or a normal (data) UI.

Pseudocode :-

1. if (restaurant) is empty \Rightarrow Load Shimmer UI
2. if (restaurant) has data \Rightarrow load actual UI

① Make a Shimmer UI component

<Shimmer/>

→ Before 'return', I can write some conditions.
ie, conditional rendering.

```
return (restaurants.length == 0) ? <Shimmer/> : (
```

```
<.>
```

```
</>
```

```
);
```


After one search, the state got updated and again when we search keywords, it won't work. This is not working because 'restaurant' gets filtered out. Everytime, we should not update this 'restaurant', so we will have to maintain 2 variables :-

Var 1 \Rightarrow List of all restaurant

Var 2 \Rightarrow List of filtered restaurant.

Whenever I'm filtering, I should filter from 'all restaurant'.

So, maintain 2 copies of the restaurant.

I show 'filtered restaurant' in my UI.

```
const [all restaurants, setAllRestaurant] = useState([]);
```

```
const [filtered restaurant, setFilteredRestaurant] = useState([]);
```

So, when I make an API call, I'll populate my all restaurants :-

```
setAllRestaurant((json?.data?.cards[2]?.data?.data?.cards));
```


→ So, for the first time, make :-

Set All Restaurants (json?.data?.cards[2].data?.data?.cards);

Set Filtered Restaurants (json?.data?.cards[2].data?.data?.cards);

At first time, make filtered restaurants as all.

→ In Search button, on clicking it, I'm passing 'restaurants' before.

Now, I have to search from my 'allRestaurants'

→ I want to show my Shimmer when my 'allRestaurants' length is 0.

Shimmer is only shown when we have no data

```
<button onClick = {() => {
```

```
  const data = filterData(searchText, allRestaurants);
```

```
  setFilteredRestaurants(data);
```

```
}}
```

```
return allRestaurants?.length === 0 ? (
```

```
  <Shimmer />) : (<? />)
```


* How to avoid Rendering component ?

→ 1 way :- use optional chaining (`allRest?.length == 0`)

→ 2nd way :- If `allRestaurant` is not there, return null, (means, It won't get rendered) .
or can return some JSX

→ EARLY RETURN

`if (!allRestaurants) return null;`

* Suppose if my `filteredRestaurants` is empty,

`if (filteredRestaurants?.length == 0) return (
 <h1>No matches </h1>;) ;`

HW :-

① Check for all Restaurants & show Shimmer

② Then show wheather `filteredRest` are not there or there .

③ I should see "No matches" only when my `filteredRest` length is zero.

Watch Recording for :-

→ Case conversion :- 02:45:00

→ Login-Logout toggle : 02:47:00

→ Read about JS Expression in Docs .