Handwritten Notes 

"Empower your Web-app with API"

Save for Later 

Ashraya K K
@ashrayaa.

Scanned with CamScanner

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.

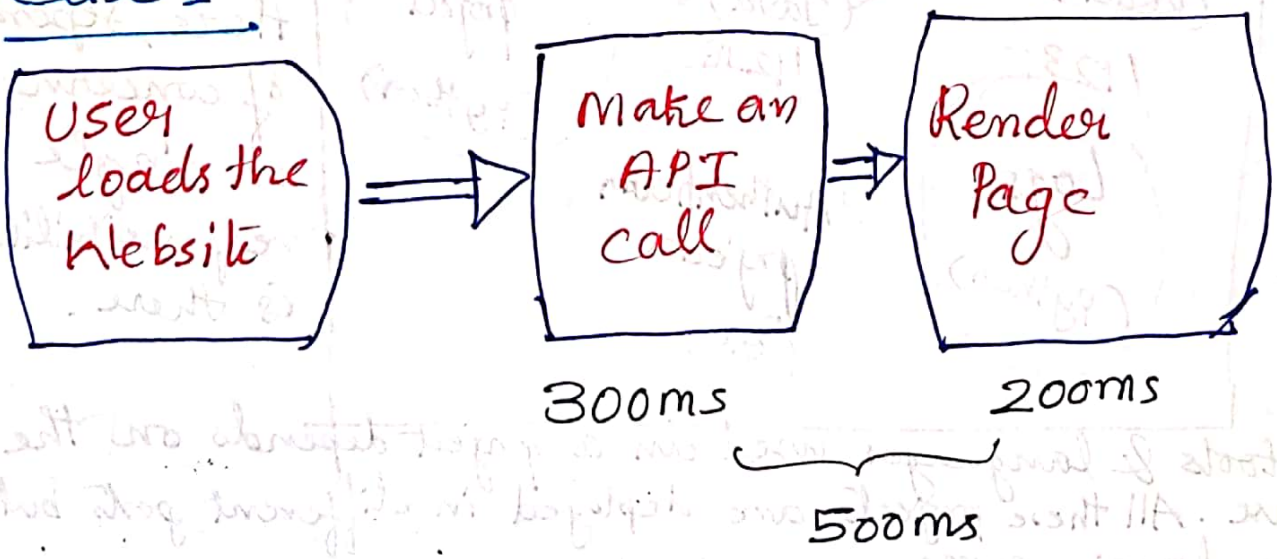
Best Way to call an api ?

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 :-

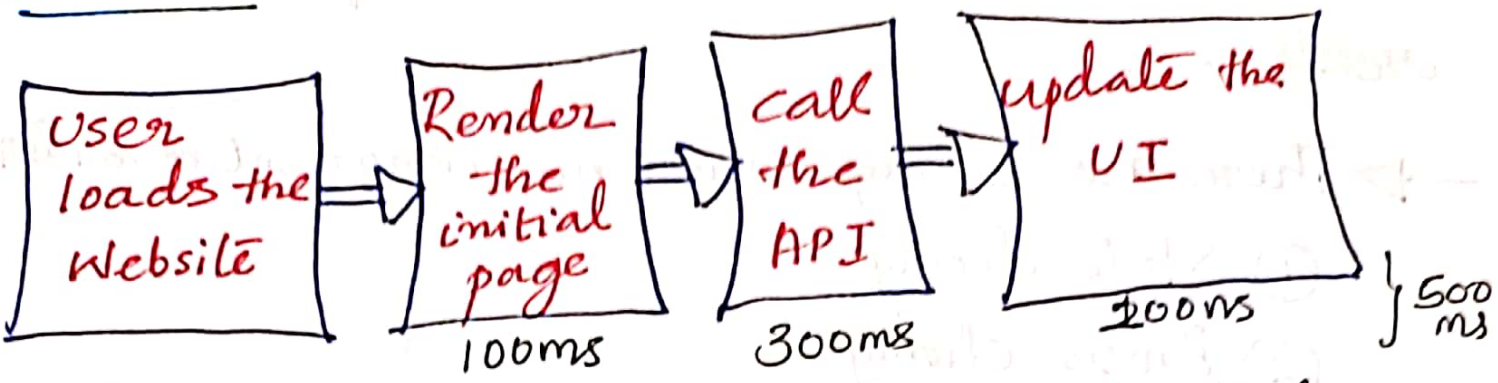
2 Ways are :-

Case 1



Scanned with CamScanner

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 immediately 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

Scanned with CamScanner

it will call the call back 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.

Scanned with CamScanner

```
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 '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. (Async-await is the most prefered way.

Scanned with CamScanner

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

* async function getRestaurants(){
const data = await fetch("https/swiggyurl");
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

Scanned with CamScanner

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 skelton.

SHIMMER UI

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

This is a bad user experience. ✗

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.

Scanned with CamScanner