Chapter 05 - Let’s get Hooked!

▼ Different ways of export

- Export by default
  - Import can have any name
  - no curly bracket on import
- Just export without default
  - import by exact name
  - need to be in curly brackets
- We can also import using *

```
//named
import { Title } from './components/Header";

//default
import Header from './components/Header";

//default and named
import Header, { Title } from './components/Header";

//or import all from file
import * as obj from './components/Header";

const title = obj.Title;
```

▼ React uses One way data-binding, Controlled/UnControlled components

- When you create an input tag with value attribute, ideally if its html - you should be able to add text in search bar. But this is JSX (which is not html) and it is not allowing to add text.

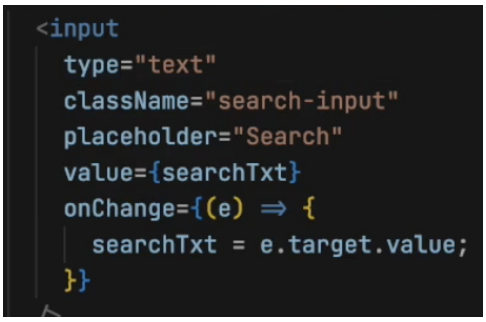
```
<div className="search-container">
  <input
    type="text"
    className="search-input"
    placeholder="Search"
    value=""
  />
```

- Even this will print ‘KFC’ in search bar and will not allow to type anything else after it

```
const Body = () => {
  let searchTxt = "KFC";

  return (
    <>
      <div className="search-container">
        <input
          type="text"
          className="search-input"
          placeholder="Search"
          value={searchTxt}
        />
```

- We can add a variable in curly braces. But we need to make the variable change when we type in input, i.e make value in searchTxt to change (two way binding). But how?
- React is One way data-binding, unlike Angular and other frameworks which is two way data binding. So when we write input , we need to do manual data binding to it
- Add onChange method, Capture the e.target.value in onChange and update variable searchTxt. Will it work? Noooooo

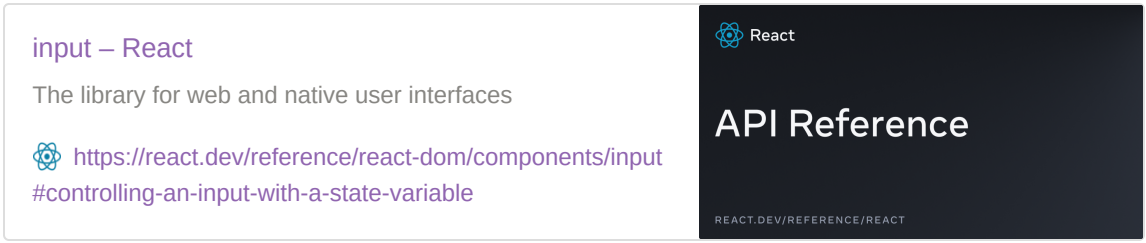


- Why it didn't work? searchTxt is declared using let. Its a local variable. So whenever we want to change the value of variable on each react rerender, this will not work. We need a React kind of variable . (i,e state)

My thoughts:

- <input /> tag used here is JSX and not html. It has some different properties.
- When we don't provide an value to <input /> in React, its Uncontrolled component & DOM api's will take care of updating DOM when we add anything in input, and it will be refelected in UI.
- But When we assign an variable to <input /> in React, its controlled component. DOM api's will not interfere, its React's responsibility to handle input now. If its not handled, we will not be able to add text in input as input element is not rerendered on each input. It will be static
- If we bind input using local variable
  - The event listener function of <input/> forms closure with local variable. So even though function context is destroyed, local variable will still be accessible
  - So we can still update the local variable inside eventlistener, but it will not be reflected in UI <input/>. Reason - this is controlled component now and we need to rerender element to update its value (or we can update element directly using DOM api like getElementById().value)
  - If we try rerendering using any way (like modifying state), the local variable will be re initialised and input remains empty.

- So we need a variable which carries it state even during rerenders. A Raect variable. That is → State
  - With this we can bind <input/> to component state variable. Update this state variable when there is change in input. This will rerender page and assign new state value to <input />
  - Refer



Why React being one-way data binding a good thing?

- When we do two way binding - App becomes unpredicatble
  - Eg: We might be using a variable in different places, updating it using one input might affect other places

▼ State, Hooks, useState

- Every component in React maintains State. You can put variables in to state. Everytime you want to use local variable in React use State

```
const [searchText] = useState();
```

- What is Hooks?
  - Just a normal function, written by facebook developers
  - Every hook has a specific function for it.
  - Comes from react library

    - import using named variable

- What is useState
    - Functionality of useState hook is to create state variables
    - useState() return a array, first element of array is variableName. Second item is the set Function to update the variable
    - We destruct it

```
const[searchtxt, setSearchTxt] = useState();
```

    - This variable is a local state variable

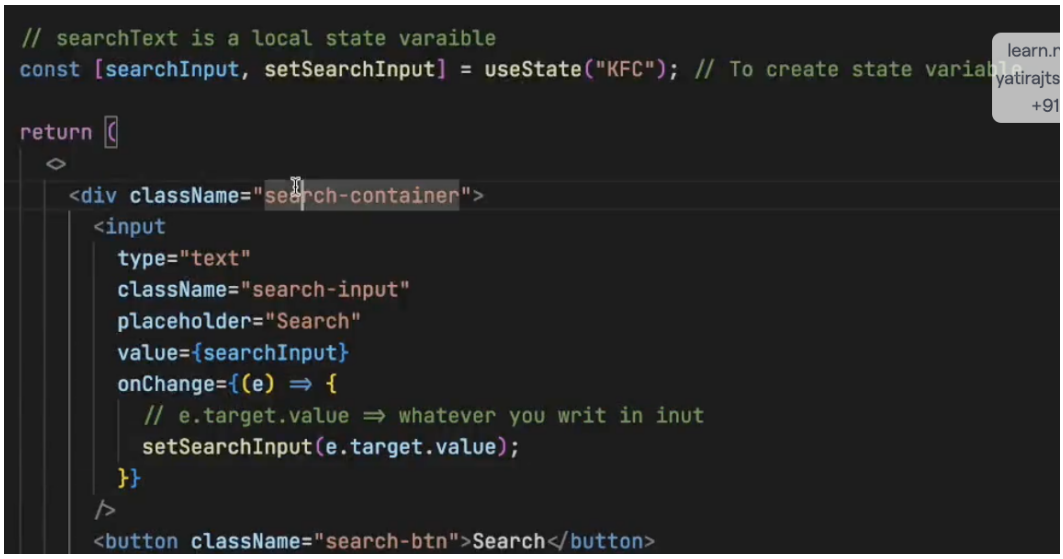
- What is State?
    - How to give default value to state

```
// searchText is a local state varaible
const [searchText] = useState("KFC"); // To create state variable
```

    - You cannot directly modify state variable. Not good practice. It Should be modified only using setter function. Its good pratice to name setter Followed by name of Variable

```
// searchText is a local state varaible
const [searchText, setSearchText] = useState("KFC"); // To create state variable
```

▼ Two way data-binding using react



- We can do the same using plain javascript, but it will mess up our head, it will also be very non performant. So always use your state variables

  Eg using plain javascript:

```
<body>
    <h1>Test</h1>
    <input
    id="myInput"
    type="text"
    value=""
    placeholder="test"

    />
</body>
<script>
function twoWayDataBinding() {
    var inputElement = document.getElementById("myInput");
    var data = '';
    myInput.addEventListener("input", function(e) {
        data = e.target.value;   // function forms closure with data & inputElemen
t
        inputElement.value = data;
    });
}
```

Interview Question:

When we have React variable, why do we need state variable?

- If someone updates the local variable, React doesnt keep track of that variable. It cannot keep track of all the variables
- So if you want to tell React to keep track of a variable, and keep it in sync with what’s hapenning in UI → use state variable.
- So React watches state variable, when its updated it will rerender UI (unlike for other local variables)

When state changes → React will not rerender whole component to which state belongs. Instead it will do reconcilliation using VDOM and Diff algorithm and sees which Element is changed - and rerenders only that element. That is why React is fast.