23/03/23

Day 37

Controlled components -

Controlled components are a set of component that offer a declarative application programming interface or API to enable full control of the state of form elements at any point in time using React state.

Value - A special property the react added to most of the form element to determine the input content at any point in time during the render life cycle.

In order to create a controlled component we need to use a combination of local state and the value prop.

onChange callback -

The onChange callback receives an event parameter, which is an event object representing the action that just took place similar to events on DOM elements

```
handleChange(event){
    setValue(event.target.value);
}
```

onSubmit callback

```
<form onSubmit={handleSubmit} ... </form>
handleSubmit(event){
    validate(value);
    event.preventDefault();
}
```

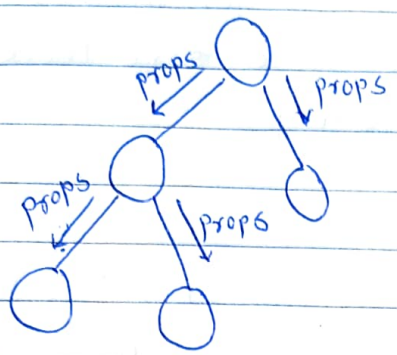
Features of controlled components

1. One-time value retrieval
2. Validating on submit
3. Instant field validation
4. Conditionally disabling a submit button
5. Enforcing a specific input format
6. Several input for one piece of data
7. Dynamic inputs.

Create a controlled form component e.g

```
const handleSubmit = (e) => { e.preventDefault();
    if (comment.length <= 10) {
    alert("please write more");
    return;
    }
}
return (
    <div className="App">
      <form onSubmit={handleSubmit}>
        <fieldset>
          <div className="field">
            <label>Commen: </label>
            <textarea value={comment} onChange
                    ={e => setComment(e.target
                        .value)} />
          </div>
          <button type="submit">submit</button>
        </fieldset>
      </form>
    </div>
```

Component tree



Context API - Alternative way to pass data
Useful for global data

e.g

```
import { createContext, useContext, useState }
        from "react";
const UserContext = createContext(undefined);

export const UserProvider = ({ children }) => {
    const [user] = useState({
        name: "ABC",
        email: "ABC@example.com",
        dob: "01/01/2001"
    });
    return <userContext.Provider value={{user}}>
            </userContext.Provide>;
};

export const UseUser = () => useContext(UserContext
```