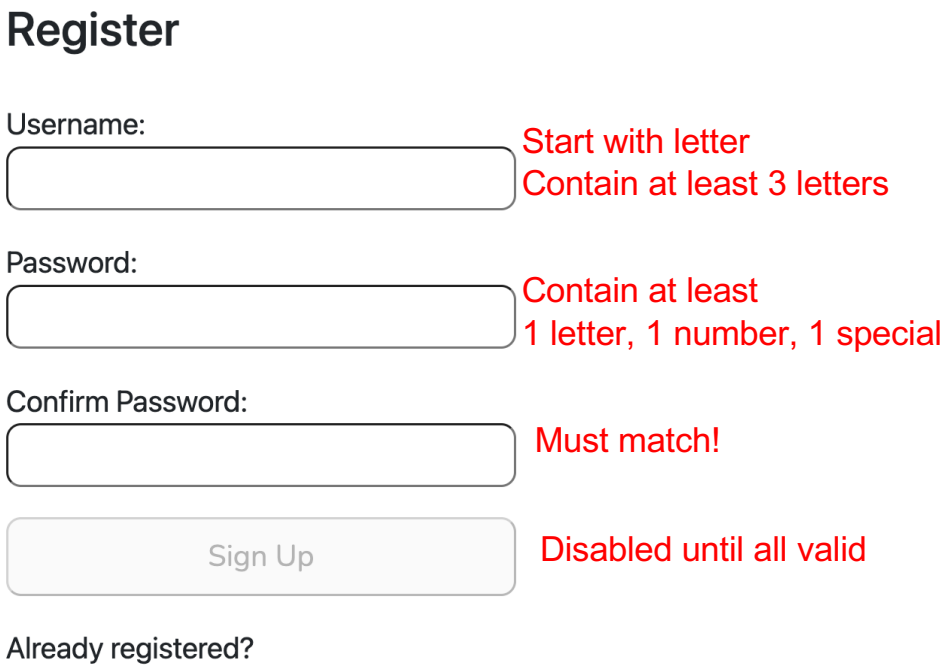
React JS

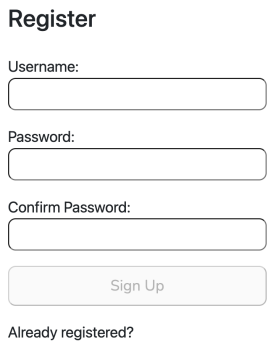
Lecture 4 (SignIn - SignUp forms)

Registration Page

- Validate username
- Validate password and matching password
- (register only when data is valid)

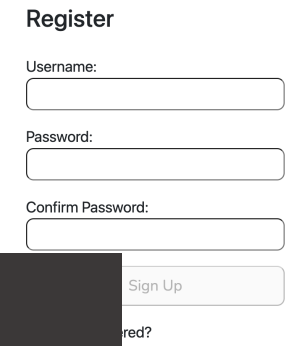


Registration Page: expressions!



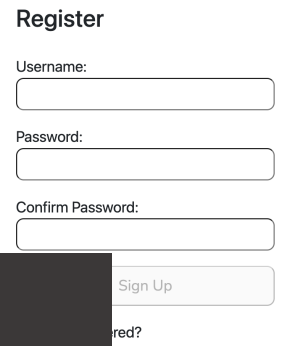

```
const user_regex= /^[a-zA-Z][a-zA-Z0-9-_]{3,23}$/;
const pwd_regex= /^(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])(?=.*[!@#$%]).{8,24}$/;
```

Registration Page: states!



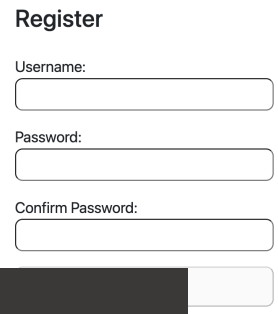

```
export default function Register() {

const [user,setUser]= useState('');
const [validName, setvalidName]= useState(false);
//is the name correct
const [nFocus, setnFocus]= useState(false);
//focus on the input field
```

Registration Page: cont.,



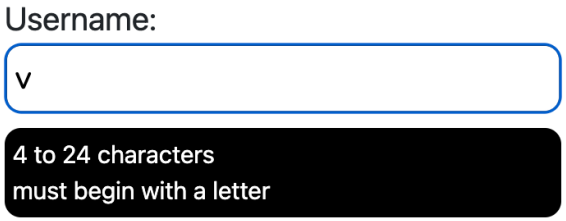

```
const [pwd,setPwd]= useState('');
const [validPass, setvPass]= useState(false);
const [passFocus, setpFocus]= useState(false);

const [mpwd,setmPwd]= useState('');
const [vmPass, setvmPass]= useState(false);
const [mpfocus, setmpFocus]= useState(false);

const [success, setSuccess]= useState(false);
```

Registration Page: validation!



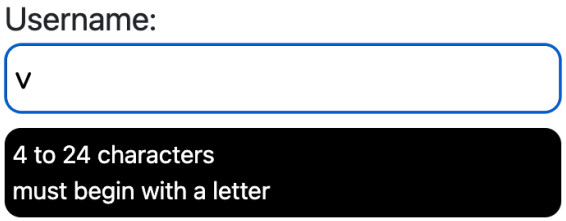

```
useEffect(()=>{
   setvalidName(user_regex.test(user))
},[user])

useEffect(()=>{
    console.log(pwd)
    console.log(pwd_regex.test(pwd))
    setvPass(pwd_regex.test(pwd))
    setvmPass(pwd===mpwd)
},[pwd,mpwd])
```

Registration Page: Form



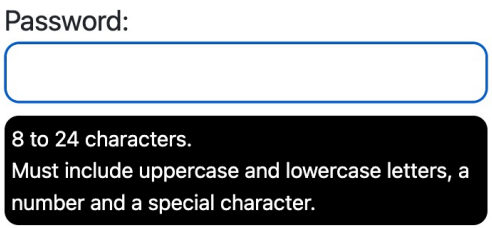

```
<label htmlFor='username'>Username: </label>
<input
    type='text'
    id='username'
    value={user}
    required
    autoComplete='off'
    onChange={(e)=>setUser(e.target.value)}
    aria-invalid={validName?'false':'true'}
    aria-describedby='usernote'
    onFocus={()=>setnFocus(true)}
    onBlur={()=>setnFocus(false)}
/>
```

Registration Page: Form



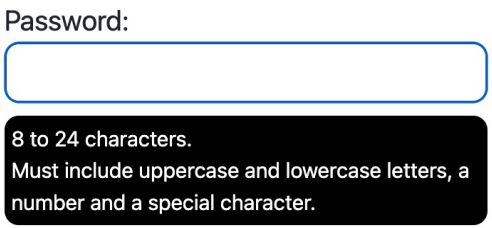

```
<label htmlFor='username'>Username: </label>
<input
    type='text'
    id='username'
    value={user}
    required
    autoComplete='off'
    onChange={(e)=>setUser(e.target.value)}
    aria-invalid={validName?'false':'true'}
    aria-describedby='usernote'
    onFocus={()=>setnFocus(true)}
    onBlur={()=>setnFocus(false)}
<p id='usernote' className={nFocus && user && !validName? "instructions":
"offscreen"}>
4 to 24 characters <br/>
must begin with a letter <br/>
</p>
```

Registration Page: Form




```
<label htmlFor="password"> Password: </label>
<input
    type="password"
    id="password"
    onChange={(e) => setPwd(e.target.value)}
    value={pwd}
    required
    aria-invalid={validPass ? "false" : "true"}
    aria-describedby="pwdnote"
    onFocus={() => setpFocus(true)}
    onBlur={() => setpFocus(false)}
/>
```

Registration Page: Form

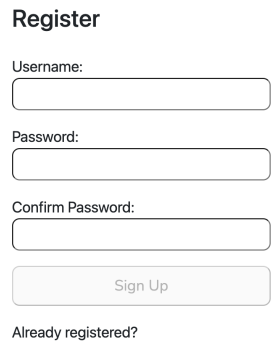



```
<label htmlFor="password"> Password: </label>
<input
      type="password"
      id="password"
      onChange={(e) => setPwd(e.target.value)}
      value={pwd}
      required
      aria-invalid={validPass ? "false" : "true"}
      aria-describedby="pwdnote"
      onFocus={() => setpFocus(true)}
      onBlur={() => setpFocus(false)}
/>

<p id="pwdnote" className={passFocus && !validPass ? "instructions" : "offscreen"}>
8 to 24 characters.<br />
Must include uppercase and lowercase letters, a number and a special character.<br />
</p>
```

Registration Page: submit validation!

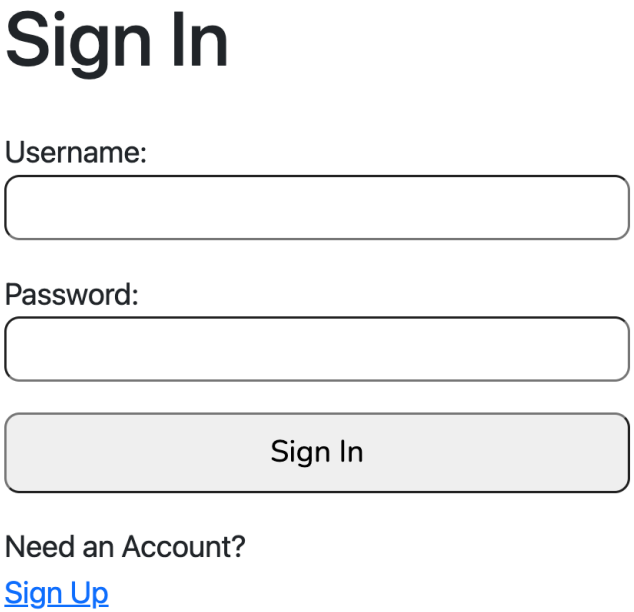
```
<button disabled={!validName || !validPass ||
   !vmPass ? true : false}>Sign Up</button>
```

```
const handleRegister = (e) =>
{
        e.preventDefault();
        //check with server if the entries are valid
        //check with server if user does not exist
        //check with server if information are saved
        setSuccess(true);
        setUser('')
        setPwd('')
        setmPwd('')
}
```



Login Page

- Validate username and password with the server
- Reroute the user to home page
- Get token + set success values

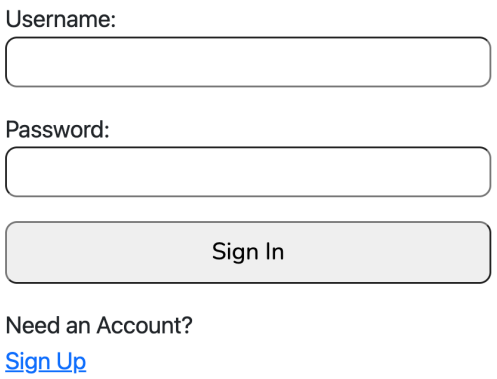


Login Page

```
const { setAuth } = useContext(AuthContext);
```

```
const [user, setUser] = useState('');
const [pwd, setPwd] = useState('');
const [success, setSuccess] = useState(false);
const [errMsg, setErrMsg]= useState('')
```

```
const handleSubmit = async (e) => {
 e.preventDefault();
 try {
 //call API
 //const accessTokn=response?.data?.accessToken;
 //const roles = response?.data?.roles;
 //setAuth({ user, pwd, roles, accessToken });

setUser('');
setPwd('');
setSuccess(true);
} catch (err) { }
}
```

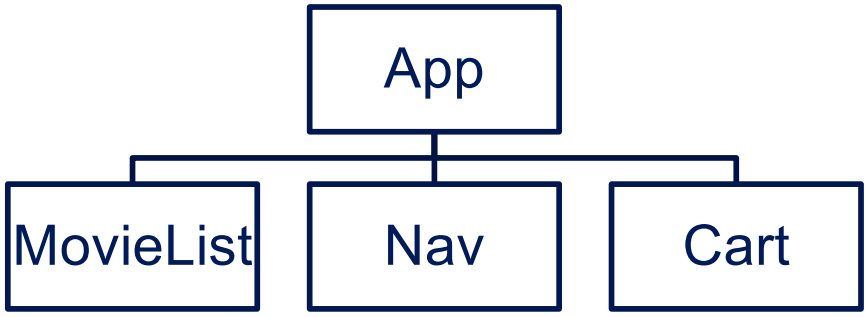


Login Page

```
<form onSubmit={handleSubmit}>
<label htmlFor="username">Username:</label>
<input
type="text"
id="username"
autoComplete="off"
onChange={(e) => setUser(e.target.value)}
value={user}
required
/>

<label htmlFor="password">Password:</label>
<input
type="password"
id="password"
onChange={(e) => setPwd(e.target.value)}
value={pwd}
required
/>
```

```
<button>Sign In</button>
</form>
<p>
Need an Account?<br />
<span className="line">
<Link to='/register'> Sign Up</Link>
</span>
</p>
```

Recap: react props and hooks



Movie Movie
Movie Movie
Movie

cartItem: {
name: movie,
price: 10KD
}

- Props drilling!
- Alternative way?

Context provides a way to pass data through the component tree without having to pass props down manually at every level.

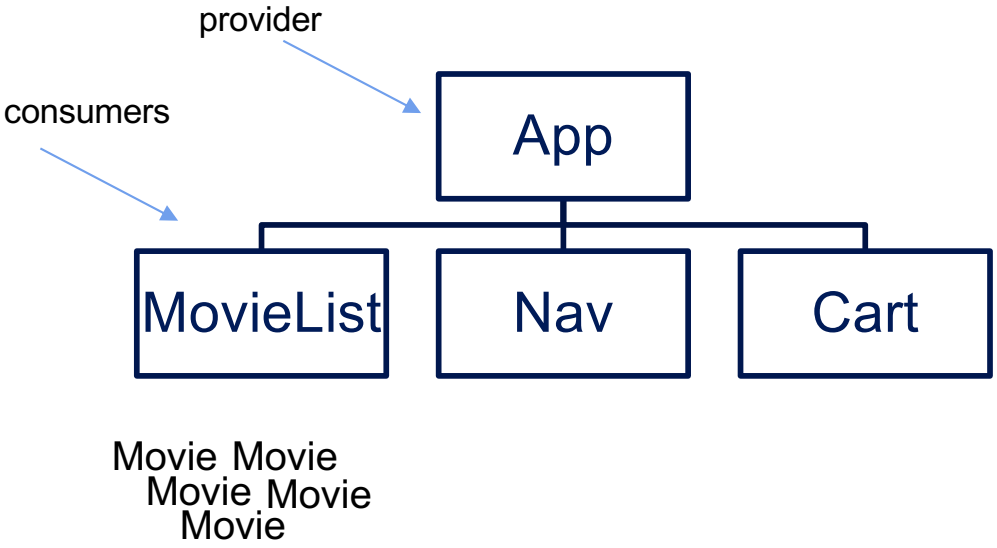
examples

Context API (for Cart)

1 Create Context

```
import { createContext, useState } from "react";

// Step 1: Create a context with a default value
const CartContext = createContext();

// Step 2: Create a provider component
function CartProvider({ children }) {
          const [cartItem, setCart] = useState([]);
          const AddCart = (movie)=>{
                    setCart((prev)=>[...prev, movie])
          }
return (
        <CartContext.Provider
          value={{cartItem, setCart, AddCart}}>

          {children}
        </CartContext.Provider>
);
}
```

Context Provider / Consumer



- **Provider** - The component that provides the value
- **Consumer** - A component that is consuming the value

CartContext

cartItem: {
name: movie,
price: 10KD
}

Context API (for Cart)

```
import {CartProvider} from './context/CartProvider’

<CartProvider>
<App />
</CartProvider>
```

Set Provider

```
import CartContext from '../context/CartProvider’;

export default function MovieList() {
..
const { cart, AddCart } = useContext(CartContext);
..
return (
<div onClick={()=>addCart(movie)} className='pr fa fa-
shopping-cart'></div>
..
)
}
```

Set Consumers

Context API (for Theme)

- Using React.createContext, we can create context and pass anything as an argument to React.createContext
- **Provider** - The component that provides the value
- **Consumer** - A component that is consuming the value

```
import React, { createContext,
useContext, useState } from 'react';
```

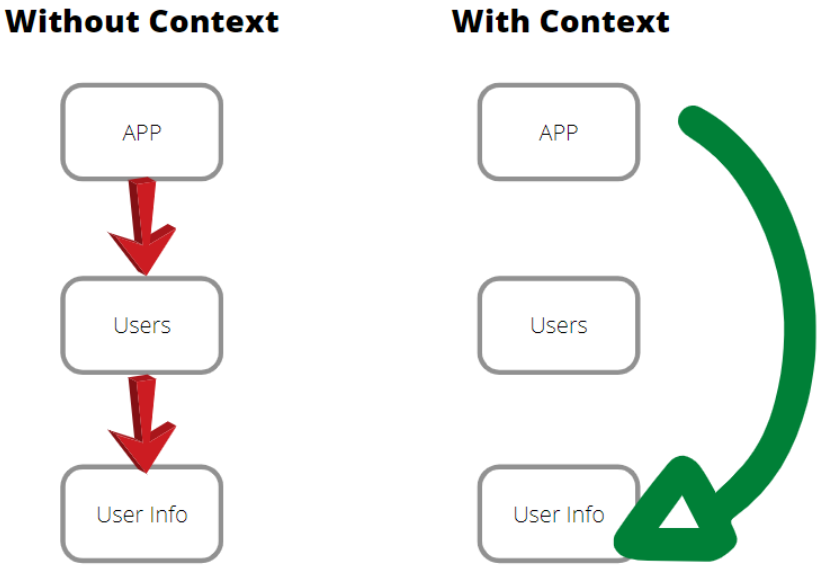
1 Create Context

```
// Step 1: Create a context with a default value
const ThemeContext = createContext();

// Step 2: Create a provider component
function ThemeProvider({ children }) {
  const[isDarkMode,setIsDarkMode]=useState(false);

  const toggleTheme =()=>{
          setIsDarkMode((prevMode) => !prevMode);
        };

return (
   <ThemeContext.Provider value={{ isDarkMode,
toggleTheme }}>
                        {children}
        </ThemeContext.Provider>
);
}
```

Context API (for Theme)

2 attach provider to the root of your App

```
function App() {
return (
  <ThemeProvider>
        <h1>Dark Mode/ Light
Mode Toggle</h1>
      <ThemeToggler />
  </ThemeProvider>
);
}

export default App;
```



Context API (for Theme)

3 Consuming the context + update

```
function ThemeToggler() {

  const { isDarkMode, toggleTheme } = useContext(ThemeContext);

return (
        <div>
          <p>Current Mode: {isDarkMode ? 'Dark' : 'Light'}</p>
          <button onClick={toggleTheme}>Toggle Theme</button>
       </div>
);
}
```

Context Page (for authentication)

```
import { createContext, useState } from "react";

const AuthContext = createContext({});

export const AuthProvider = ({ children }) => {
const [auth, setAuth] = useState({});

return (
 <AuthContext.Provider value={{ auth, setAuth }}>
     {children}
   </AuthContext.Provider>
)
}

export default AuthContext;
```