

# React Review

You should understand:

- React components as functions
- React components are composable
- Components can pass props to other components
- How to set the state of a component
- State and prop changes trigger re-renders

# A Bigger Example

Consider:

```
// ...
if(userState.isLoggedIn) {
  content = <ShowStuff stuff={userState.info}/>;
} else {
  content = <Login onLogin={login}/>;
}

return (
  <div className="app">
    <Nav user={userState} onLogout={logout}/>
    {content}
  </div>
);
```

# Conclusions

Readability:

- `login/logout` are functions
  - *inside* the component
- `content` is just for cleanliness

Separation of Concerns:

- `userState` is managed in this component
- other components have it as a **read-only** prop

## **Side Note: Props are Read Only**

Props should be read-only

Common mistake: use them to initialize state

Problems when they change in the ancestor/sibling or on a re-render

Which version is true?

Avoid this: **Props are read-only**

# Considering Nav

```
import { fetchLogout } from './services';

const Nav = ({ user, onLogout }) => {
  const logout = () => {
    fetchLogout()
      .then( () => onLogout() );
  };
  return (
    <ul className='nav'>
      <li>{user.username}</li>
      { user.isLoggedIn && <li onClick={logout}>Logout</li> }
    </ul>
  );
};
```

# Nav Thoughts?

Why an `<ul>`?

- Common menu structure, unrelated to React
- React doesn't change these conventions

`user` isn't called `userState` here

- it's not state, it's a prop
- change is made through `onLogout()`

Service call is encapsulated within `Nav.jsx`

- App doesn't know HOW Nav did the logout

# Login

```
return (  
  <div className="login">  
    <p className="error">{error}</p>  
    <input onChange={ (e) => setUsername(e.target.value) }/>  
    { isLoading ?  
      <img alt="spinner" src={spinner}/> :  
      <button onClick={ performLogin }>Login</button>  
    }  
  </div>  
) ;
```

# Basic Login Thoughts

So much state **local to component**

- `isLoading`
- `error`
- `username`

Logic in JSX file, but out of JSX syntax

- `performLogin`

Spinner!



# More Login code

```
const performLogin = () => {  
  if(!username) {  
    setError(messages.USERNAME_REQUIRED);  
    return;  
  }  
  setError('');  
  setIsLoading(true);  
  
  fetchLogin(username)  
    .then( (userInfo) => onLogin(userInfo.username) )  
    .catch( (err) => {  
      setError(messages[err.code] || 'DEFAULT');  
      setIsLoading(false);  
    });  
};
```

# More Login Thoughts

Note lack of state "cleanup"

- Assumes component will be removed after login

What is `messages`? Why CAPS?

- common CONSTANTS convention
- external config file:

```
export default messages = {  
  DEFAULT: 'Oh no! Something went wrong, please try again',  
  USERNAME_REQUIRED: 'Username is required',  
  NETWORK_ERROR: 'There was a problem reaching your network, please try again',  
  LOGIN_REQUIRED: 'You must be logged in to view this content',  
  LOGIN_UNAUTHORIZED: 'You are not permitted to view this content',  
};
```

# How to tie interactions to rendering

`useEffect` hook:

- does NOT return anything
- runs when component renders
- or only when any listed vars change

# useEffect example

```
useEffect( () => {  
  fetchLoginStatus()  
  .then( userInfo => {  
    setUserState({  
      isLoggedIn: true,  
      username: userInfo.username,  
    });  
  });  
}, []);
```

# useEffect thoughts

- Here we are rendering only on **initial** render
  - the `[]` means it only runs once
  - an array of vars means rerun only if they change
  - ...no vars in array, so nothing triggers change
- If component removed and replaced, total new state
  - But that can't happen for this example

## **useEffect advice**

- Use cautiously
- Load state as needed
- Remember spinners to account for delay