- **React State Management**

استفاده صحیح ازstate

مستقیم state را تغییر ندهید

به‌روز رسانی state ممکن است غیرهمزمان باشد

به‌روز رسانی‌های state ادغام‌شده هستند

State management یکی از ویژگی های مهم و اجتناب ناپذیر هر برنامه پویا است .

react یک APIساده و انعطاف پذیر برای پشتیبانی از sate management در یک کامپوننت ارایه میدهد. در واقع react یک data store داینامیک برای هر کامپوننت فراهم میکند . اگر کاربر با تعامل با برنامه ، state را تغییر دهد ، ممکن است رابط کاربری بعد از آن کاملا متفاوت بنظر برسد زیرا کامپوننت با فراخوانی متد ()render به همراه state جدید خود را دوباره رندر میکند .

برنامه های react ، state خود را به صورت داخلی مدیریت میکنند و برای برنامه هایی با کامپوننت کم به خوبی کار میکنند . اما زمانی که برنامه بزرگتر میشود پیچیدگی مدیریت state های به اشتراک گذاشته شده در بین کامپوننت ها دشوار میشود و پیدا کردن اینکه وقتی مشکلی پیش می آید چه اتفاقی می افتد واقعا سخت است . به همین دلیل است که به state management نیاز داریم و برای این کار از recoil ، hooks ،redux میتوان کمک گرفت .

- https://reactjs.org/docs/state-and-lifecycle.html

- **Functional Components**

در ری‌اکت، Functional Components راهی ساده برای ساخت کامپونننت‌های بدون stateو آن‌هایی که تنها متد render دارد است. به صورت خلاصه برای ساخت کامپونننت‌های ساده و مختصر از کامپونت تابعی استفاده می‌کنیم. تنها کافی‌ست به جای ساخت کلاسی که از React.Components ارث می‌برد یک تابع بنویسیم که props را در ورودی دریافت و کد نیاز به رندر را برگرداند. نوشتن کامپوننت تابعی بسیار ساده‌تر از کلاس‌هاست و کامپونننت‌های زیادی را می‌توان به این صورت نوشت.(

Functional Components از رایج ترین کامپوننت هایی هستند که هنگام کار با ریکت با آن ها مواجه میشویم . این توابع ممکن است داده ها را به عنوان پارامتر دریافت کنند یا خیر . در Functional Components مقداری که return میکنیم کد JSX است تا به درخت DOM رندر میشود.

چندین نوع react component در گذشته رایج بود . قبل از معرفی هوک به عنوان یک کامپوننت بدون state نیز شناخته میشد . با روی کار آمدن react hooks ، اکنون این امکان وجود دارد که کل برنامه را فقط ازطریق Function ها به عنوان react component ها بنویسید . و در صورت نیاز از props به عنوان ورودی استفاده میشود . Functional Components خواندن و تست کردن و دیباگ کردن اسان تری دارند.

(

- https://reactjs.org/tutorial/tutorial.html#function-components

- **Hook in react** :

هوک ویژگی جدیدی است که در نسخه 16.8 ریکت معرفی شده است آن‌ها به شما اجازه می‌دهند که از state و دیگر قابلیت های ری‌اکت بدون نوشتن کلاس استفاده کنید.

هوک یک تابع ویژه است که به شما اجازه می‌دهد از امکانات ری‌اکت استفاده کنید. برای مثال، هوک useState به شما این امکان را می‌دهد که از state در کامپوننت‌های تابعی استفاده کنید. هوک‌های دیگری هم وجود دارند که به یادگیری آن‌ها در آموزش‌های بعدی می‌پردازیم.

اگر شما یک کامپوننت تابعی بنویسید و متوجه شوید که نیاز به استفاده از state در آن دارید، در گذشته و قبل از معرفی هوک‌ها مجبور بودید که کامپوننت خود را به کلاس کامپوننت تبدیل کنید. حالا با معرفی هوک‌ها می‌توانید هر وقت که بخواهید در کامپوننت‌های تابعی خود از state استفاده کنید. ما در ادامه این کار را انجام خواهیم داد!

(هوک ها توابعی هستند که شمارا به react state و ویژگی های function component ها متصل میکنند . هوک ها در کلاس ها کار نمیکنند و اجازه میدهند تا از ویژگی های ریکت بدون نوشتن کلاس استفاده کرد.

هوک ها backward-component هستند به این معنا که هیچ تغییر (break change) را نگه نمیدارند . ریکت چند built-in hook مثل , useReducer useEfect, useState دارد .)

- https://www.javatpoint.com/react-hooks#:~:text=Hooks%20are%20the%20new%20feature,does%20not%20work%20inside%20classes.
- https://reactjs.org/docs/hooks-overview.html
- https://www.w3schools.com/react/react_hooks.asp
- https://www.devaradise.com/react-functional-component-with-hooks
- https://www.tutorialspoint.com/reactjs/reactjs_state.htm

- https://www.loginradius.com/blog/async/react-state-management/
- https://www.dotnettricks.com/learn/react/functional-component-react