1) React state management:

React developers create and maintain state. They want state management to be simple, extendible, and atomic. React has moved in this direction by introducing hooks. Problems may arise when the state should be shared among many components.

توسعه دهنده های ری اکت استیت میسازن و نگهداری میکنن برای مدیریت کردن استیت نحوه ای که هم ساده باشه قابل گسترش و اتمیک باشه ری اکت اومد و هوک هارو معرفی کرد

برای زمان هایی که استیت رو با کامپوننت ها به اشتراک میذاشتیم و در مدیریت اوها به مشکل میخوردیم .

با استفاده از کتابخانه‌هایی مثل redux یا mobX میتوان یک استیت گلوبال برای پروژه در نظر گرفت. بدین صورت که در همه ی کامپوننت ها دسترسی به آن استیت به راحتی انجام شود

State management is simply a way to engender communication and sharing of data across components. It creates a concrete data structure to represent your app's State that you can read and write. Since React 16.8, every React component, whether functional or class, can have a state

مدیریت استیت یک راه ساده برای بوجود اوردن ارتباط و اشتراک گذاری دیتا بین کامپوننت هاست .

این یک ساختار داده مشخص برای نشان دادن وضعیت استیت اپ ایجاد می کند.

Ref:

https://dev.to/workshub/state-management-battle-in-react-2021-hooks-redux-and-recoil-2am0#:~:text=State%20management%20is%20simply%20a,class%2C%20can%20have%20a%20state
.

---

2) Functional componenets :

This function is a valid React component because it accepts a single “props” (which stands for properties) object argument with data and returns a React element. We call such components “function components” because they are literally JavaScript functions.

یکی از روش های ایجاد کامپوننت ها فانکشنال کامپوننت ها هستد ولی یک سری مزیت نسبت به کلاس کامپوننت ها دارن از جمله :

۱- خوانایی فانکشنال کامپوننت ها بالاتر از کلاس کامپوننت هاست (چون همون توابع ساده جاواسکرپیتی هستند)

۲- تست کردن فانکشنال کامپوننت ها ساده تر از کلاس کامپوننت هاست.

۳- کدهای نهایی ترجمه شده (و حتی سورس کدها) با فانکشنال خیلی کوتاه تر میشه.

۴- استفاده از فانکشنال کامپوننت ها به رعایت Practice Best ها کمک میکنه. یکی از مهمترین Practice Best های ری اکت جداسازی کامپوننت های نمایشی از container هاست. استفاده از فانکشنال کامپوننت ها به رعایت این موضوع کمک زیادی میکنه (وقتی کامپوننتی بنویسیم که داخلش setState نداشته باشیم)

۵- درنهایت تیم ری اکت اعلام کرده تغییرات ورژن ۱۷ مبتنی بر استفاده از فانکشنال کامپوننت ها برای افزایش کارایی و سرعت ری اکت هست و توصیه کرده تا جایی که مجبور نشدیم از کلاس کامپوننت ها استفاده نکنیم.

Ref:

https://virgool.io/iran-react-community/%D9%81%D8%A7%D9%86%DA%A9%D8%B4%D9%86%D8%A7%D9%84-%DA%A9%D8%A7%D9%85%D9%BE%D9%88%D9%86%D9%86%D8%AA-%D8%AF%D8%B1-%D9%85%D9%82%D8%A7%D8%A8%D9%84-%DA%A9%D9%84%D8%A7%D8%B3-%DA%A9%D8%A7%D9%85%D9%BE%D9%88%D9%86%D9%86%D8%AA-b8bhzvut2zp1

---

3)HOOK in react :

هوک‌ها قابلیت جدیدی هستند که به نسخه ۱۶٫۸ اضافه شده‌اند. با استفاده از آن‌ها می‌توانید بدون نوشتن کلاس، از state و دیگر ویژگی‌های ری‌اکت استفاده کنید.

با استفاده از هوک‌ها، شما می‌توانید منطق وابسته به state را از یک کامپوننت جدا، به طور مستقل تست و چندین جا استفاده کنید .هوک‌ها به شما اجازه می‌دهند که منطق وابسته به state را بدون تغییر در طراحی سلسله مراتب کامپوننت‌ها، هربار مجددا استفاده کنید .با این ویژگی می‌توان هوک‌ها را میان کامپوننت‌ها و حتی برای جامعه توسعه‌دهندگان به اشتراک گذاشت.

ما معمولا مجبور به نگه‌داری از کامپوننت‌هایی بودیم که در ابتدا ساده بودند اما وقتی رشد کردند، منطق‌های دارای state و side effectهایی به آن‌ها اضافه شد که مدیریت آن‌ها را مشکل می‌کرد. در هر تابع lifecycle منطق‌هایی وجود دارد که با هم ارتباطی ندارند. برای مثال، یک کامپوننت ممکن است در componentDidMount وcomponentDidUpdate برای دریافت داده درخواست ارسال کند. درحالی‌که همان تابع componentDidMount می‌تواند دارای منطقی غیر مرتبط باشد که به اضافه کردنevent listener ها می‌پردازد، و در متد componentWillUnmount آن‌ها [event listener] ها [را پاکسازی (cleanup) می‌کند. کدهایی که متقابلا به هم مرتبط هستند] مثل اضافه و پاک‌سازی [event listener از یکدیگر جدا هستند، ولی کدهایی که کاملا غیر مرتبطاند] اضافه کردن event listener و دریافت داده [در یک متد نوشته می‌شوند. که این احتمال خطا و باگ و بی‌ثباتی را بیشتر می‌کند.

در بسیاری از موارد امکان این وجود ندارد که بتوانیم این کامپوننت‌ها را به اجزای کوچکتر تقسیم کنیم، زیرا منطق دارای stateدر همه جا وجود دارد. همچنین تست کردن آن‌ها دشوار است. این یکی از دلایلی است که بسیاری از افراد ترجیح می‌دهند تا ری‌اکت را با یک کتابخانه‌ی مدیریت state دیگری ترکیب کنند. اگرچه، معمولا باعث اضافه‌شدن مقدار زیادی abstractionمی‌شود و شما را مجبور می‌کند بین تعداد زیادی از فایل‌ها جابه‌جا شوید که خود باعث مشکل‌تر شدن استفاده‌ی مجدد از کامپوننت‌ها می‌گردد.

برای حل این مسیله، هوک‌ها به شما اجازه می‌دهند یک کامپونت را به توابع کوچکتری تقسیم کنید که مبنای آن ارتباط اجزایشان است) مثلا اضافه کردن یک subscription یا دریافت داده(، نه تقسیم بر اساس متدهای .lifecycle همچنین می‌توانید برای مدیریت state محلی از یک reducer کمک بگیرید تا آن را بیشتر قابل پیش‌بینی کنید.

Ref:

https://fa.reactjs.org/docs/hooks-intro.html