1 : R eact State Management

مدیریت حالت به سادگی راهی برای ایجاد ارتباط و به اشتراک گذاری داده ها در بین اجزا است. این یک ساختار داده مشخص برای نشان دادن وضعیت برنامه شما ایجاد می کند که می توانید آن را بخوانید و بنویسید.

از React 16.8، هر مؤلفه React، چه تابعی و چه کلاسی، می تواند یک حالت داشته باشد.

در ساده ترین تعریف، State یک شی جاوا اسکریپت است که بخشی از یک مؤلفه را نشان می دهد که می تواند بر اساس عملکرد نتیجه کاربر تغییر کند. همچنین می توانید بگویید که حالت ها به سادگی حافظه یک جزء هستند.

هنگامی که کاربر در یک برنامه معمولی React اقدامی را انجام می‌دهد، تغییراتی در وضعیت مؤلفه رخ می‌دهد. اگرچه این بد نیست، اما اگر برنامه شروع به بزرگ شدن کند، به سرعت مشکل ساز می شود. از این رو، پیچیدگی چنین برنامه ای پیگیری همه وابستگی ها را بسیار دشوار می کند.

فرض کنید در حال ساخت یک اپلیکیشن تجارت الکترونیک هستیم. در برنامه ای مانند این، تقریباً هر عنصر می تواند یک جزء باشد - سبد خرید، دکمه ها، جلسه مشاهده سبد خرید، پرداخت، نوار ورود و غیره. در این برنامه، تنها یک اقدام کاربر برای افزودن به سبد خرید می تواند تأثیر بگذارد. بسیاری از اجزای دیگر توسط:

تغییر وضعیت خود جزء سبد خرید،

افزودن سبد خرید به تاریخچه سبد خرید کاربر،

بررسی اقلام محصول

و این فقط ذکر چند مورد از چیزهای زیاد دیگری است که می توانیم به برنامه تجارت الکترونیک اضافه کنیم. اگر مهندسان مسئول در حین توسعه برنامه، مقیاس پذیری را در نظر نگیرند، ممکن است به زودی در دراز مدت با بسیاری از اشکالات و مشکلات مواجه شوند.

اشکال زدایی و اصلاح برنامه ای مانند این در نهایت می تواند دردسرساز باشد.

سناریوهای بالا اهمیت حالت را در یک برنامه معمولی React به ما نشان می دهند.

در مدیریت وضعیت در این برنامه، می‌توانیم از هر کتابخانه‌ای که انتخاب می‌کنیم استفاده کنیم. آنها همچنان بدون توجه به کارشان انجام می دهند.

معمولاً، حالت باید به نزدیک‌ترین مؤلفه والد و حالت بعدی تا زمانی که به یک اجداد مشترک برای هر دو مؤلفه که به حالت نیاز دارند برسد و سپس منتقل شود.

شما به یک ابزار مدیریت حالت مانند Redux، Recoil نیاز دارید که حفظ این حالت ها را آسان تر می کند.

منابع: <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 : F unctional Components

مزایای functional component ها نسبت به class component

* تولید کدهای بسیار کمتر در functional component
* کدهای بسیار ساده تر و قابل فهم تر
* و اینکه functional component ها reusable تر هستند

دو نوع کامپوننت ها در React وجود داره Class Components و Functional Components.تفاوت بین این دو type خیلی واضح هست.Class components در واقع کلاس های ES6  هستند و Functional Components توابع هستند

hooks افزوده جدیدی به React 16.8 هستند. آنها به شما اجازه می دهند از حالت و سایر ویژگی های React بدون نوشتن کلاس استفاده کنید.

در مثال بالا، برای کامپوننت های تابعی، از hooks ها (useState) برای مدیریت وضعیت استفاده می کنیم. اگر یک کامپوننت تابع را بنویسید و متوجه شوید که باید حالتی را به آن اضافه کنید، قبلا باید آن را به یک componentکلاس تبدیل می کردید. اکنون می توانید از Hook در داخل کامپوننت تابع موجود برای مدیریت وضعیت استفاده کنید و نیازی به تبدیل آن به component کلاس نیست. به جای کلاس‌ها، می‌توان از Hooks در مؤلفه Functional استفاده کرد، زیرا این روش بسیار ساده‌تر برای مدیریت وضعیت است. قلاب ها را فقط می توان در functional componentsاستفاده کرد، نه در اجزای کلاس.

یک functional componentفقط یک تابع جاوا اسکریپت ساده است که props را به عنوان آرگومان می پذیرد و یک عنصر React را برمی گرداند. وهیچ روش رندری در functional componentsاستفاده نمی شود.وهمچنین به عنوان مؤلفه های Stateless شناخته می شود زیرا آنها به سادگی داده ها را می پذیرند و آنها را به شکلی نمایش می دهند که عمدتاً مسئول رندر UI هستند

یک class component از شما می خواهد که از React گسترش دهید. کامپوننت کنید و یک تابع رندر ایجاد کنید که عنصر React را برمی گرداند. وباید متد render() داشته باشد که HTML را برمی گرداند وهمچنین به عنوان مؤلفه های Stateful شناخته می شود زیرا منطق و حالت را پیاده سازی می کنند.

<https://www.geeksforgeeks.org/differences-between-functional-components-and-class-components-in-react/>

3: Hook in react

Hooks allow us to "hook" into React features such as state and lifecycle methods.

Hook Rules

* Hooks can only be called inside React function components.
* Hooks can only be called at the top level of a component.
* Hooks cannot be conditional
* Hook ها توابعی ساده هستند که این امکان را به ما می دهند از قابلیت های state و lifecycle در react بدون استفاده از ساختار class در کامپوننت هایی که به صورت تابع هستند استفاده کنیم. Hook ها در داخل class ها غیر قابل استفاده می باشند (توصیه نمی شود که کامپوننت های موجود را دوباره بازنویسی کنید اما می توانید در کامپوننت های جدید از Hook ها استفاده کنید).
* React تعدادی Hook (همانند useState) را به صورت پیش فرض تهیه کرده است،
* useState یک Hook می باشد. این تابع را داخل یک تابع کامپوننت فراخوانی می کنیم تا یک state محلی را به آن اضافه کنیم. React این state را بین فراخوانی های متعدد کامپوننت حفظ می کند. useState یک آرایه با دو عنصر بر می گرداند که عنصر اول مقدار فعلی state و عنصر دوم یک تابع برای بروزرسانی مقدار state می باشد. این تابع شبیه this.setState می باشد با این تفاوت که state قبلی و جدید را با هم ادغام نمی کند.
* useState تنها یک پارامتر به عنوان مقدار اولیه state می گیرد. در مثال بالا مقدار اولیه 0 می باشد و سبب می شود شمارشگر از 0 آغاز می شود، توجه داشته باشید که بر خلاف this.state که مقدار باید حتما object باشد در اینجا مقدار state می تواند object یا هر نوع داده دیگری باشد. مقدار اولیه state تنها در فراخوانی اول کامپوننت استفاده می شود.

### **Effect Hook چیست ؟**

به احتمال زیاد اعمالی مانند دریافت/ارسال داده، یا کار با DOM را در یک کامپوننت React انجام داده اید. این اعمال اصطلاحا "Side Effects" نامیده می شوند زیرا بر سایر کامپوننت ها نیز اثراتی دارند که در حین اجرای اولیه کامپوننت قابل انجام نیست.

با استفاده از useEffect امکان استفاده از اعمال side effect در کامپوننت تابعی ممکن می باشد. در واقع هدف معادلسازی استفاده از componentDidMount، componentDidUpdate و componentWillUnmount در کلاس React می باشد اما با این تفاوت که از یک متد استفاده شود.

<https://virgool.io/@afshin.talebi/%D8%A2%D8%B4%D9%86%D8%A7%DB%8C%DB%8C-%D8%A8%D8%A7-hook-%D8%AF%D8%B1-react-exd5l6akzufi>

https://www.w3schools.com/react/react\_hooks.asp