React State Management

State یک شی JavaScript است که داده های پویای یک کامپوننت را ذخیره می کند و رفتار آن را تعیین می کند. از آنجا که stateپویاست ، این پویایی یک کامپوننت را قادر می سازد تا اطلاعات را بین رندرها ردیابی کند و پویا و تعاملی باشد.

بزرگترین چالش در اپلیکیشن ریکت برای فرانت developer مدیریت state است. اپلیکیشن های بزرگ ریکت به تنهایی برای مدیریت پیچدگی ها کافی نیست و این دلیل هست که بیشتر developer ها از ریکت هوک و دیگر لایبرری های مدیریت state مانند redux استفاده می کنند.

React state management :

کامپوننت های ریکت یک آبجکت stateداخلی دارند. این state داده های کپسوله شده و پایدار در بین رندرینگ های کامپوننت ها است.

State یک واژه انتزاعی در جاوااسکریپت است اگر به کاربر با تعامل با اپلیکیشن شما state را تغییر دهد .واسط کاربری شما(ui) ممکن است بعد از تغییر کاملا متفاوت به نظر برسد. زیرا با اینstate جدید نشان می شود .

یکی از دلایلی که ما به مدیریت state در ریکت نیاز داریم آن است که رریکت از کامپوننت ها استفاده می کند آنها state را درداخل خودشان مدیریت می کنند و این برای کامپوننت ها تعداد کم به خوبی کار می کند اما زمانی که اپلیکیشن بزرگتر واپلیکیشن پیچیده تر می شود مدیریت state و شر کردن state در بین کامپوننت ها دشوار می شود.

state فقط در یک کامپوننت کلاس قابل استفاده است. اگر پیش بینی می کنید که یک کامپوننت به مدیریت state نیاز دارد ، باید به عنوان یک کامپوننت کلاس ایجاد شود نه یک کامپوننت فانکشنال. برای تغییر دادن مقادیر state می توانیم ازتابع setState استفاده کنیم.

<https://www.loginradius.com/blog/async/react-state-management/>

<https://amanjacademy.com/the-concept-and-application-of-state-in-react-js-for-beginners/>

Functional Components

یکی از روش های ساخت کامپوننت در ریکت فانکشنال کامپوننت ها هستند و اولین و بهترین نوع ساخت کامپوننت در ریکت ، فانکشنال کامپوننت ها هستند.

function Welcome(props) {

return <h1>Hello, {props.name}</h1>;

}

بنا به جاوااسکریپت ES6 این فانکشنن معتر است چون یک ریکت المنت(JSX) بر می گرداند. و همچنین ما می توانیم آن را به صورت arrow فانکشن نیز بنویسیم .

const Welcome = (props) => {

return <h1>Hello, {props.name}</h1>;

}

Prop یک آرگومان آبجکتی است که یه المنت ریکت بر می گرداند.

همچنین یکی از مزیت های کلاس تا قبل از ورژن 16.8 استفاده از **state**  بود که در رکت 16.8 با استفاده از هوک و useState این ویژگی به فانشکال کامپوننت ها هم اضافه شد  
استفاده از **Lifecycle** **Hooks**  (useEffect) که در ورژن 16.9 اضافه شد و برنامه نویسان توانستند life cycle در فانکشنال کامپوننت ها استفاده نمایند.

فانکشنال کامپوننت ها به دلیل خوانایی بالاتر نسبت به کلاس کامپوننت ها (چون همان توابع ساده جاوااسکریپت هستند) دارند. و تست کردن فانکشنال کامپوننت ها ساده تر از کلاس کامپوننت ها هستند. و کدهای نهایی ترجمه شده با فانکشنال خیلی کوتاه تر از کلاس کامپوننت ها است. و در کل فانکشنال کامپوننت ها به best practice بودن کد ها کمک می کنند

یکی از مهمترین **Best** **Practice** 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 )

<https://virgool.io/iran-react-community>

<https://www.freecodecamp.org/news/react-components-jsx-props-for-beginners>

Hook in react

همانطور که قبلا هم گفتیم مزیت اصلی یکی از مزیت های کلاس تا قبل از ورژن 16.8 استفاده از **state**  بود و life cycle ها بودند که فانکشنال کامپوننت ها این امکان فراهم نبود اما هوک ها با اضافه شدن به فانکشنال ها کامپوننت ها این امکان را برای ما فراهم می کنند هوک ها از نسخه 16.8 اضافه شده اند و به ما اجازه می دهند که بدون از state و دیگر فیچر های ریکت بدون نوشتن کلاس استفاده کنیم .

هوک ها از ریکت ایمپورت می شوند و در کلاش کامپوننت ها قابل استفاده نیستند

ازقوانین هوک ها می توان گفت که 1- هوک ها فقط می توانند در داخل اجزای تایع ریکت فرخوانی شوند 2- هوک ها فقط می توان در بالای کامپوننت فراخوانی کرد 3- هوک ها نمی توانند conditional باشند.

اولین فیچر هوک همان useState هست که به ما اجازه می دهد. که یک state در فانکشنال کامپوننت پیاده سازی کنیم

useEffect هوک به ما اجازه می دهد که ساید افکت ها روی کامپوننت ما اجرا بشوند. ساید افکت ها مانند آپدیت مستقیم dom و رfetching دیتا و... ، useEffect ها دو آرگومان می گیرند که دومی اختیاری است .

useEffect(<function>, <dependency>)

useMemo : یکی از امکانات هوک هست که به ما اجازه می دهد که مقادیر را بین رندرها حفظ کند می توان از آن برای ذخیره یک مقدار قابل توجه استفاده کرد که هنگام به روز رسانی باعث رندر کجدد نمی شود.

usecallback ، useContextو useRef دیگر توابع تعریف شده در هوک می باشند که امکاناتی برای بهبود عملکرد فانکشنال کامپوننت ها فراهم کرده اند.

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

<https://www.w3schools.com/react/react_hooks.asp>

<https://www.w3schools.com/react/react_usestate.asp>

<https://www.w3schools.com/react/react_useeffect.asp>