State management(1 به سادگی راهی برای ایجاد ارتباط و به اشتراک گذاری داده ها در بین componentها است. 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 تابع و چه کلاس، 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.

کتابخانه های بسیاری برای مدیریت state ایجاد شده اند از جمله : Recoil، Jotai، Redux، Rematch، Hookstate و...

(2 کامپوننت های تابعی (function components)  در واقع فقط همان توابع JavaScript هستند .آنچه این تابع را به یک component ری اکت تبدیل می کند این است که “props” را به عنوان آرگومان با داده می پذیرد و سپس یک عنصر React را برمی گرداند.

در یک فانکشن بر خلاف کلاس ها خبری از کانستراکتور و ارث بری نیست. برای تعریف یک کامپوننت فانکشنال ابتدا یک فانکشن با اسم مورد نظر و دلخواه تعریف می کنیم و سپس داخل آن JSX مورد نیاز را پیاده سازی کرده و سپس آن را برای استفاده در کامپوننت های دیگر export می کنیم.

از مزایای استفاده از function componentها:

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. (‌همونطور که تو مثال این مقاله دیدید)

۴- استفاده از فانکشنال کامپوننت ها به رعایت **Best** **Practice** ها کمک میکنه. 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 )

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.

3) Hook قابلیت جدیدی است که از نسخه 16.8 به React اضافه شده است و امکان استفاده از state و دیگر قابلیت های React را بدون استفاده از ساختار class ممکن می سازد.

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.

React تعدادی Hook همانند useState را به صورت پیش فرض تهیه کرده است، اما ما می توانیم Hook های مورد نظر خود را نوشته و استفاده کنیم.

اعمالی مانند دریافت/ارسال داده، یا کار با DOM در یک کامپوننت React 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.