به نام خدا

بوتکمپ مکتب شریف ۶۶

نام نام‌خانوادگی : ساتنیک شاهی جانی زرنه

- **React State Managment**

ریکت یک کتاب‌خانه‌ی مدیریت State است

وقتی یک برنامه‌ی ریکت درست می‌کنید، شما در واقع دارید یک‌سری کامپوننت را بغل هم می‌گذارید تا ساختار درختی که از <App /> شروع می‌شود و در <Input /> یا <Button /> یا ... پایان می‌یابد، بسازید. شما تمامی کامپوننت‌های لول پایین که برنامه‌تان رندر می‌کند را در یک مکان مرکزی مدیریت نمی‌کنید. شما به هر کدام از کامپوننت‌ها به طور جداگانه اجازه می‌دهید که آن را مدیریت کنند که این یک روش عالی برای ساختن UIهای فوق‌العاده است.

What is React State Management?

React components have a built-in state object. The state is encapsulated data where you store assets that are persistent between component renderings.
The state is just a fancy term for a JavaScript data structure. If a user changes state by interacting with your application, the UI may look completely different afterwards, because it's represented by this new state rather than the old state.

Why do you need React state management?

React applications are built using components and they manage their state internally and it works well for applications with few components, but when the application grows bigger, the complexity of managing states shared across components becomes difficult.

Here is a simple example of an e-commerce application, in which the status of multiple components will change when purchasing a product.

- Add that product to the shopping list
- Add product to customer history
- trigger count of purchased products

If developers do not have scalability in mind then it is really hard to find out what is happening when something goes wrong. This is why you need state management in your application.
Let’s discuss how to use react state management using react hooks and redux

What is Redux?

Redux was created to resolve this particular issue. it provides a central store that holds all states of your application. Each component can access the stored state without sending it from one component to another. Here is a simple view of how Redux works.

Actions in Redux

Actions are payloads of information that send data from your application to your store. Actions are sent using `store.dispatch()`. Actions are created via an action creator.

Reducers in Redux

Reducers specify how the application's state changes in response to actions sent to the store.

Store in Redux

The store holds the application state. You can access stored state, update the state, and register or unregister listeners via helper methods.
In other words, Redux gives you code organization and debugging superpowers. This makes it easier to build more maintainable code, and much easier to track down the root cause when something goes wrong.

What is React Hook?

These are functions that hook you into React state and features from function components. Hooks don't work inside classes and it allows you to use React features without writing a class.

Hooks are backwards-compatible, which means it doesn't keep any breaking changes. React provides some built-in
Hooks like `useState`, `UseEffect` and `useReducer` etc. You can also make custom hooks.

React Hook Rules

- Call hook at the top level only means that you need to call inside a loop, nested function, or conditions.
- React function components are called hooks only.

- https://roocket.ir/articles/application-state-management-with-react
- https://www.loginradius.com/blog/async/react-state-management/

- **Functional Components**

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

Functional Componentsکدها را برای فهمیدن ساده تر می کند

یکی از مزایای functional components اینکه کدها رو برای خواندن و فهمیدن ساده تر می کنه.اگر شما بر روی یک پروژه شخصی کار میکنید، شاید زیاد به این موضوع توجه نکنید.اگر شما بعد از یک ماه به پروژه ای که نوشتید برگردید،به احتمال زیاد درک کردن کدهایی که خودتون نوشتید،سخت خواهد بود.پیشنهاد میکنم نقل قول obert Martin رو در این کتاب در مورد Clean Code بخونید.

”نسبت زمان صرف شده برای خواندن و نوشتن 10 به 1 است.نوشتن آسان است ولی خواندن سخت “

یعنی شما شاید خیلی راحت کد می نویسید ولی زمانی که بعد مدت ها بر میگردید و کدها رو مرور می کنید باید 10 برار زمان نوشتن وقت بزارید تا کدها رو درک کنید.
این مزیت تا زمانی که شما در دو حالت کامپوننت ها رو بنویسید شاید براتون قابل درک نباشه.پس پیشنهاد میکنم حتما تست کنید و از نتایج نوشتن کامپوننت ها با function لذت ببرید.

Functional Components عملکرد بهتری دارند

Functional Componentsبیشتر reusable هستند

این مورد ممکن است یکم بحث برانگیز باشد.اگر ما در function components از state ها استفاده نکنیم،کامپوننت ها راحت تر پیاده سازی می شوند و خیلی راحت در قسمت های دیگه پروژه و پروژه های دیگه قابل استفاده هستند.

چرا از توابع استفاده کنیم؟

ری اکت برای تبدیل کردن کدها به یک نسخه قابل قبول در جاوا اسکریپت نیاز به یک کامپایلر دارد . این کامپایلر به نام babel شناخته شده.
اگر ما کدهای کلاس کامپوننت ها را با توابع در babel مقایسه کنیم ، متوجه میشیم که کلاس ها خروجی بسیار سنگینی نسبت به توابع دارند .

ReactJS Functional Components

Functional components are some of the more common components that will come across while working in React. These are simply JavaScript functions. We can create a functional component to React by writing a JavaScript function. These functions may or may not receive data as parameters. In the functional Components, the return value is the JSX code to render to the DOM tree.

Functional components lack a significant amount of features as compared to **class-based components**. The gap is made up with the help of a special ReactJS concept called **“hooks”**. Hooks are special functions that allow ReactJS features to be used in **functional components**.
Functional components do not have access to dedicated state variables like **class-based components**. The only **“state”** that a functional component

effectively has access to are the props passed to it from its parent component. ReactJS has access to a special hook called **useState()** that can be used for giving the illusion of working with the state in functional components.
The **useState()** is used to initialize only one state variable to initialize multiple state variables, multiple **useState()** declarations are required. The first value returned is the initial value of the state variable, while the second value returned is a reference to the function that updates it. When the state variable needs to be updated, it can be done by calling the update function and by updating the state variable directly inside it.

Functional components do not have access to lifecycle functions like class-based components do since lifecycle functions need to be defined within the boundaries of a class. If lifecycle functions need to be used with functional components, a special React hook called useEffect() needs to be used. It is worth noting that **useEffect()** isn't an exact duplicate of the lifecycle functions – it works and behaves in a slightly different manner.

Data is passed from the parent component to the child components in the form of props. ReactJS does not allow a component to modify its own props as a rule. The only way to modify the props is to change the props being passed to the child component. This is generally done by passing a reference of a function in the parent component to the child component. Props have more significance in functional components for the simple reason that functional components do not have access to a state, unlike class-based components.

- https://reactapp.ir/react-functional-components/
- https://virgool.io/iran-react-community/%DA%A9%D8%A7%D9%85%D9%BE%D9%88%D9%86%D9%86%D8%AA-%D9%87%D8%A7-%D8%AF%D8%B1-%D8%B1%DB%8C-%D8%A7%DA%A9%D8%AA-gxatvwucn1em
- https://www.geeksforgeeks.org/reactjs-functional-components/

- **Hook in React**

Hook چیست ؟

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 (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) Hookهمانند (useStateرا به صورت پیش فرض تهیه کرده است، اما شما می توانید Hook های مورد نظر خود را نوشته و استفاده کنید.

استفاده از hook در پروژه های موجود محدودیتی از نظر سازگاری کد ایجاد نمی کند و می توانید بدون آنکه کدهای موجود را تغییری بدهید از hook ها در کامپوننت های جدید استفاده کنید.

Hookها این امکان را به شما می دهند که سازماندهی side effect های مرتبط به هم را در یک بخش انجام دهید) همانند افزودن یا لغو یک (effect به جای آنکه در کلاس React آنها را در چرخه حیات های مختلف به صورت جداگانه بنویسید.

قواعد استفاده از Hook ها:

استفاده از hook ها تنها محدود به رعایت دو شرط می باشد:

- Hookها باید در بالاترین سطح فراخوانی شوند. در حلقه های تکرار، شرط ها و توابع تودرتو نباید استفاده شوند.
- Hookها تنها در کامپوننت هایی که به صورت تابع هستند قابل استفاده می باشند و در ساختار کلاس کاربرد ندارند) در داخل Hook هایی هم که خودتان ایجاد کرده اید نیز می توانید از Hook ها استفاده کنید که در ادامه توضیح داده خواهد شد.(

نکته: React از linter plugin برای چک کردن این قواعد استفاده می کند تا حتما توسط توسعه دهنده رعایت شود.

Hooks were added to React in version 16.8.

Hooks allow function components to have access to state and other React features. Because of this, class components are generally no longer needed.

What is a Hook?

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

Hook Rules

There are 3 rules for hooks:

- 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

- https://www.w3schools.com/react/react_hooks.asp
- 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://fa.reactjs.org/docs/hooks-state.html