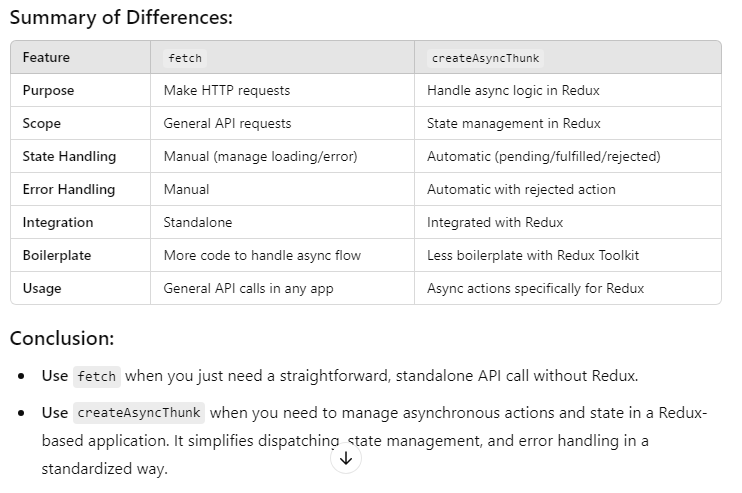
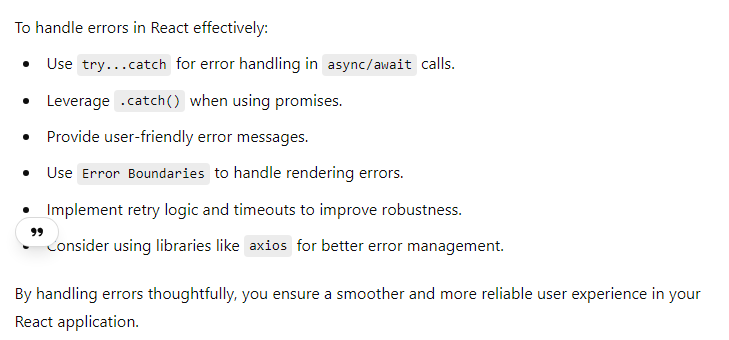
**Diff between fetch and createasyncthunk**



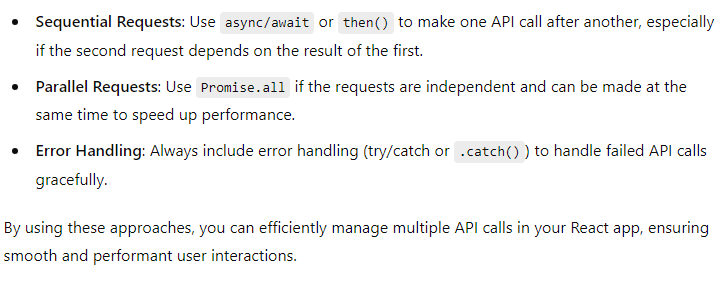
**how to protect routes and components**

To protect routes and components in React, you can implement a combination of techniques including route guards, authentication contexts, and role-based access control. These tools help ensure that only authorized users can access sensitive parts of your application, and they improve both security and user experience.

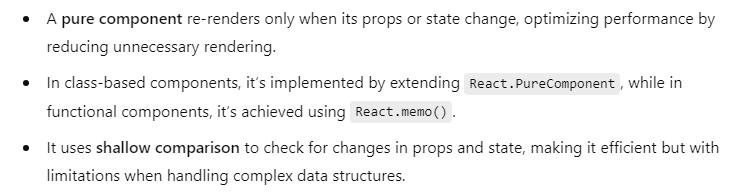
**how to handle errors**



**how can you handle multiple api one after another in reactjs**



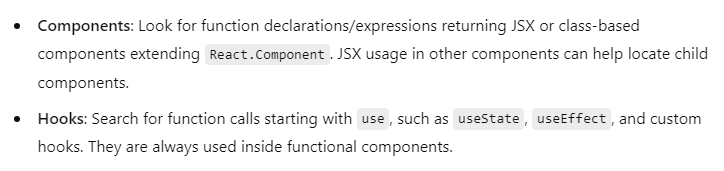
**What is pure component?**



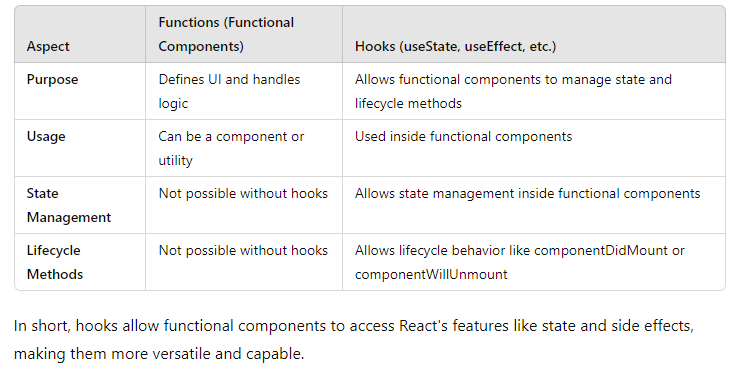
**What hook you build?**

1. **useCart**
2. **useAuth**
3. **useProducts**
4. **useCheckout**
5. **useWishlist**
6. **usePagination**
7. **useOrderHistory**

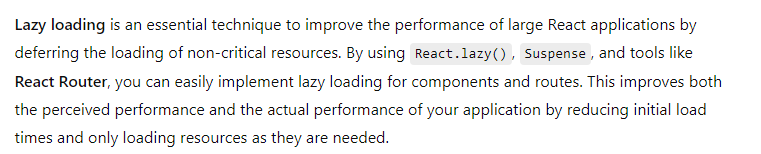
**How to find component and hook?**



**Difference between hook and functions in react**



**Lazyloading**



**Optimisation technique**

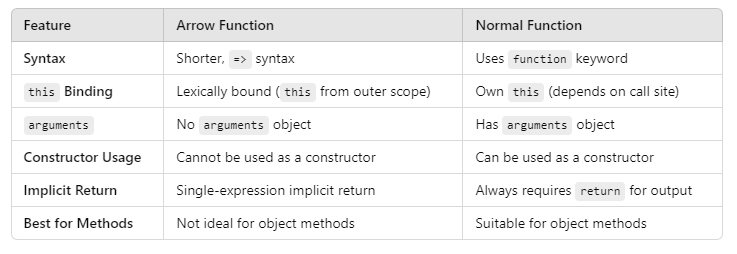
|  |
| --- |
| code splitting |
| lazyloading |
| ssr |

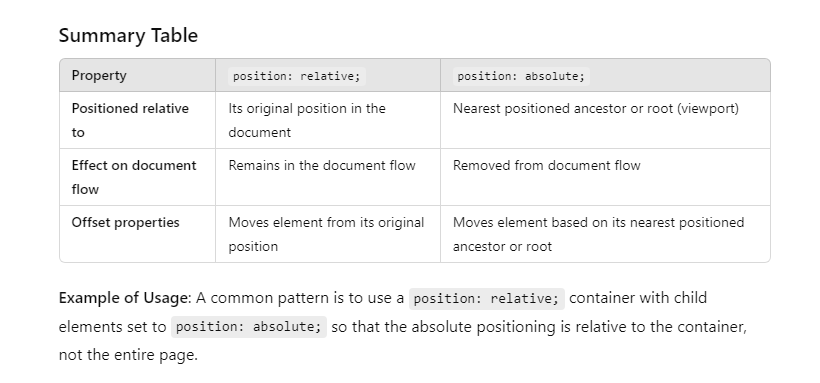
**Keys in react**

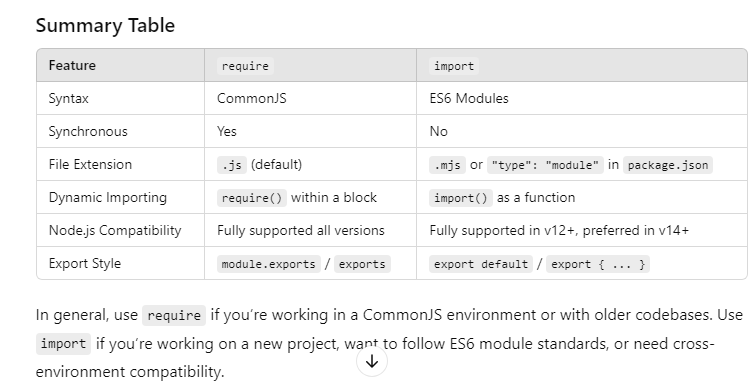
 Keys help React identify which items have changed, been added, or removed, allowing it to only re-render those elements that have changed instead of re-rendering the entire list.

 This results in improved performance, especially in large lists.

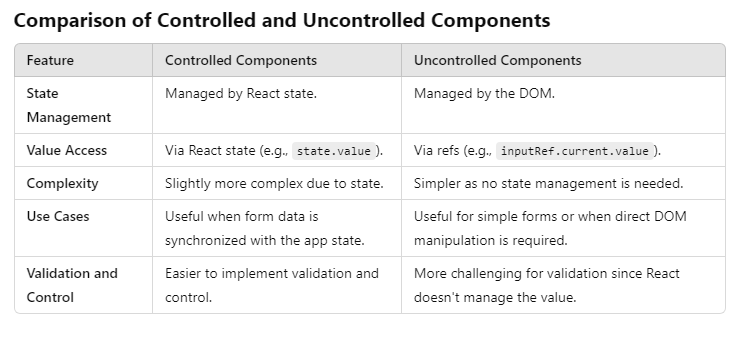
**Arrow vs Normal function**





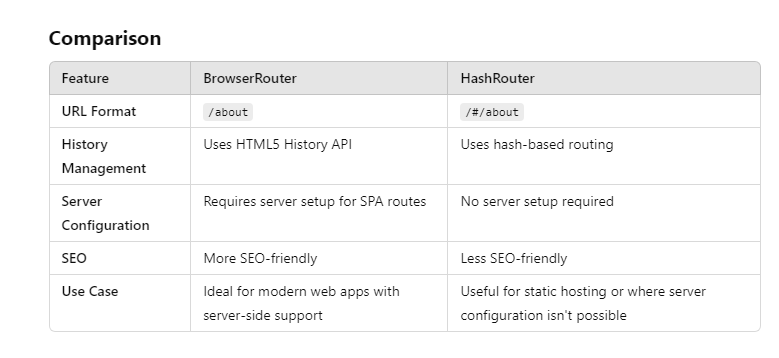


**Controlled vs uncontrolled**



**When to Use Which**

* **Controlled Components**: Preferred for most React applications as they provide better control, consistency, and synchronization with the application state.
* **Uncontrolled Components**: Suitable for quick forms, integrating with non-React libraries, or when simplicity is a priority.



**Conclusion**

* **BrowserRouter** is the preferred choice for most modern web applications where SEO and clean URLs matter.
* **HashRouter** is suitable for simpler setups, static hosting, or scenarios where server configuration is not feasible.

