LWC best practices

1. Html file: Use camel case to name your component.
2. JavaScript File: Java Script Class name should be in PascalCase

3. There are two ways to call Apex class in the Lightning web component.
    - Imperatively
    - Wire
        i. Wire as property
        ii. Wire aa function.

→As per Lightning component best practices, use @wire over imperative method invocation. @wire fits nicely in the overall Lightning Web Component reactive architecture.

4. As per LWC, the best practice is to use Lightning Data Service functions to create, Record, and delete a record over invoking Apex methods. Yes, there are some use cases where you need multiple records, then we can use Apex methods.

Give preference to user interface form type in below order.

- **lightning-record-form**: It is the fastest/most productive way to build a form.
- **lightning-record-view-form**: If you need more control over the layout, want to handle events on individual input fields, or need to execute pre-submission.
- **@wire(getRecord)**: If you need even more control over the UI or if you need to access data without a UI.

5. Use Storable Action; It will reduce calls to the Server.

@AuraEnabled(cacheable=true)

- **Caution**: A storable action might result in no call to the server. Never mark as storable an action that updates or deletes data.

6. Build reusable LWC comps.

7. Lazy loading helps you to load the data only when it is required. Infinite scrolling (**enable-infinite-loading)** enables you to load a subset of data and then load more data when users scroll to the end of the table.

- Lazy loading is an optimization technique to load the content on-demand. Instead of loading the entire data and rendering it to the user in one go as in bulk loading, the concept of lazy loading assists in loading only the required section and delays the remaining, until it is needed by the user

Steps -

- Apex class with offset (loads only said amount of data) (apex part)
- To enable infinite scrolling, specify enable-infinite-loading and provide an event handler using onload more. (HTML part)
- We create connectedCallback function to load the initial data and then we are using loadmoreData function to load more record from Apex base on offset. The onload more event handler retrieves more data when you scroll to the bottom of the table until there are no more data to load. To display a spinner while data is being loaded, set the **isLoading** property to true. (js part)

ref (lazy load) - https://www.apexhours.com/lazy-loading-in-lightning-web-component/
ref (all) - https://www.apexhours.com/lightning-web-components-lwc-part-2/

8. Use Lightning design system as much as possible. Less codes and more functionality.
9. Try to limit event listeners, it will make the comp busy. An event listener is a function in JavaScript that waits for an event to occur then responds to it.
10. In production, disable the use of debugs, as it slows down the performance of the comp.
11. If a parent has many children, then we should not call apex from all the methods. Instead, we can call apex from parent comp and pass the data to the child comp.
12. Using Custom labels to avoid hard code values/ statements.
13. Optimizing Component Communication by Using events to communicate between parent and child components efficiently.
14. Using LWC's Lifecycle Hooks Appropriately Utilize connectedCallback, renderedCallback, and other lifecycle hooks to manage component behaviour and data flow.
15. Using Static Resources to Store images, scripts to improve performance.
16. Avoiding Inline CSS and Use CSS Files to improve maintainability.