\*\*\*\*\*Learn about (npm install framer-motion)\*\*\*\*\*

Lifecycle: به مراحل ایجاد. تغییر و از بین رفتن یک کامپوننت در dom چرخه حیات یا lifecycle گفته میشود.به مرحله ایجاد mounting. به مرحله تغییر update و به مرحله از بین رفتن unmounting گفته میشود.

Mounting:

Constructor: در کامپوننت های کلاسی اگر بخواهیم هنگام ایجاد یک کامپوننت یک کار خاصی انجام بشود در این قسمت باید بنویسیم.مثلا

تعریف کردن state ها.

getDerivedStateFromProps: یک **static method** در کامپوننت کلاسیک React هست که برای همگام کردن state با props استفاده میشه، قبل از هر رندر (چه موقع mount چه هر update)، فراخوانی میشه و میتونه بر اساس props جدید state رو به‌روز کنه.مثلا یک prop دریافت می‌کنی که هر وقت عوض شد باید state داخلی هم باهاش هماهنگ بشه (سناریوی فرم یا ری‌ست کردن مقدار).به عنوان ورودی props و state رو میگیره و خروجی باید یک آبجکت باشه که میره به عنوان state جدید قرار میگیره.

 **static** هست، پس دسترسی به this نداره‌.

 همیشه باید یا یک object جدید (state جدید) برگردونه یا null اگه نیازی به آپدیت نباشه.

 نباید side effect انجام بدی (فقط آپدیت state بر اساس props مجازه، fetch یا دسترسی به DOM نه!).

render: بعد از متود getDerivedStateFromProps اجرا میشود و وظیفه رندر کردن کامپوننت را دارد.

مهم=> componentDidMount: این متود زمانی که کامپوننت به طور کامل رندر بشود یعنی در واقع mount بشود.

Updating: هنگامی که prop جدید ست کنیم یا از setState و forceUpdate استفاده کنیم مرحله آپدیت شروع میشود.

getDerivedStateFromProps

shouldComponentUpdate: وقتی که از setState یا prop جدید استفاده کنیم وارد این متود میشویم.forceUpdate به صورت مستقیم وارد مرحله render میشود.خروجی آن یک مقدار بولی باید باشد و اگر مقدار true باشد به مرحله render میرود.پس میتونیم اگر شرط خاصی برای rerender کردن در نظر داریم اینجا بنویسیم.به عنوان ورودی state و prop های جدید را میگیرد.وقتی یک کامپوننت والد داریم و هنگام تغییر بخوایم دوباره رندر کنیم کامپوننت های فرزند هم دوباره رندر میشوند پس میتونیم داخل این متود با شرط بررسی کنیم که اگر تغییری در کامپوننت فرزند اعمال نشده false را برگردانیم تا از رندر های اضافی که باعث کاهش performance میشود جلوگیری کنیم.

PureComponent: به جای نوشتن متود قبلی میتونیم کلاس کامپوننت را از ارث بری از کلاس pureComponent به جای component بنویسیم تا خودش همون عملکرد رو داشته باشه و از کد نویسی اضافی جلوگیری بشه.

getSnapShotBeforeUpdate: بعدا راجبش تحقیق کن

componentDidUpdate: بعد از اینکه تمام مراحل اپدیت کامل انجام بشود انجام میشود.هرچیزی که در متود قبلی برگردانده شود در این متود قابل دسترسی هستش.

Unmounting=>

componentWillUnmount: زمانی یک کامپوننت از دام واقعی حذف بشود اجرا میشود .مثلا فرض کنیم یک کامنت در حال حذف شدن است میتوانیم بعد از حذف تمام ریپلای های آن را نیز حذف کنیم.

Callback function: زمانی که به یک تابع یک ورودی دیگرکه آن هم تابع باشد بدهیم به تابع ورودی callback function میگویند.به هیچ عنوان نباید موقع پاس دادن به تابع دیگر جلوش پرانتز بزاربم چون بلافاصله اجرا میشود در صورتی که اجرا باید توسط خود تابع انجام بشود.اگر تعداد زیادی تابع داشته باشیم که هر کدام نیز ورودی callback داشته باشند به شرایطی که به ان callback hell میگویند که با استفاده از promise از ان جلوگیری میکنیم.

نکته: کد ها در جاوااسکریپت به صورت سینک اجرا میشوند ولی توابعی مانند setTimeOut باعث میشود که کد به صورت اسینک اجرا شود پس در این شرایط میتوانیم با استفاده از توابع و اجرا به صورت callback کاری کنیم کد به صورت سینک اجرا شود.

Promise

Async-await

Fetch: تابع fetch در جاوااسکریپت برای ارسال درخواست HTTP (مثل GET یا POST) به یک سرور استفاده می‌شود و داده را به صورت Promise برمی‌گرداند.

 **url**: آدرس موردنظر برای درخواست

 **options**: (اختیاری) تنظیمات مثل متد، هدرها، بادی و غیره

 **Promise**: پاسخ به صورت Promise برمی‌گردد و امکان استفاده از async/await یا then/catch را فراهم می‌کند.

 **نتیجه**: معمولاً response را با متدهایی مثل .json()، .text() یا .blob() تبدیل می‌کنند.