42. REACT PERF & RENDERING

Agenda : (3rd of interview Level)

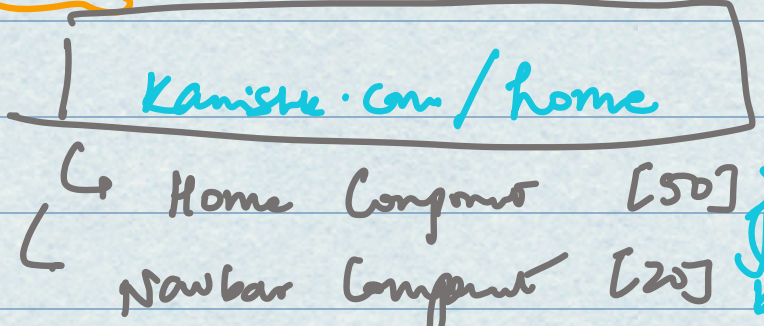
1) React Perf
   - Code Splitting
   - Dynamic Import
   - Lazy loading
     - React.Lazy
     - React Suspense

2) Memoization
   - React.Memo
   - useMemo
   - useCallback

* Code Splitting & Lazy Loading

5 Component in Portfolio website

/Home 50KB
/About 50KB
/Profile 400KB



Contact 20KB

Navbar 20KB

Blogs

Whatever your route needs, only that part of the component / file gets loaded.

Issues if NO code split

1. Long initial page load time (browser have download large JS file)
2. Poor performance (High network download ↑ speed ↓)
3. Unnecessary resource usage.
   - Unnecessary download.
   - Capacity to wasted less source.

Module bundler

Compress
Minify

Vite
Webpack
Rollup
Parcel.

Bundles together

CHUNK

DYNAMIC IMPORTS

- Allows us to load modules asynchronously.
  - On demand loading.
  - Helps in
    - ↳ Code splitting in smaller chunks
    - I
    - Chunking

fixing / Advantages.

1. Code splitting
2. On Demand Loading
3. Improved performance.
4. Better resource utilization

Webpack

automatically created

home.chunk.js
about.chunk.js
profile.chunk.js

based on route, these chunks will be downloaded.

↓

better practices

↓

React.lazy & Suspense

Given to us by React.

React.lazy ⇒ Allows us to load the component dynamically.

React.Suspense ⇒ Provides a way to handle the loading state while component is being loaded.

↓

[handle the loading state with a fallback ...]

Component

```
const Home = Lazy(() => import('./Home'));
```

REACT.lazy

- Component will only be loaded when it is needed.
- Helps in reducing initial load time.

React.Suspense

- Component that wrap the lazily-loaded component & handle loading state.
- Takes fallback prop.

```
<Suspense fallback={<Rc />}>
    <Home />
    <Profile />
</Suspense>
```

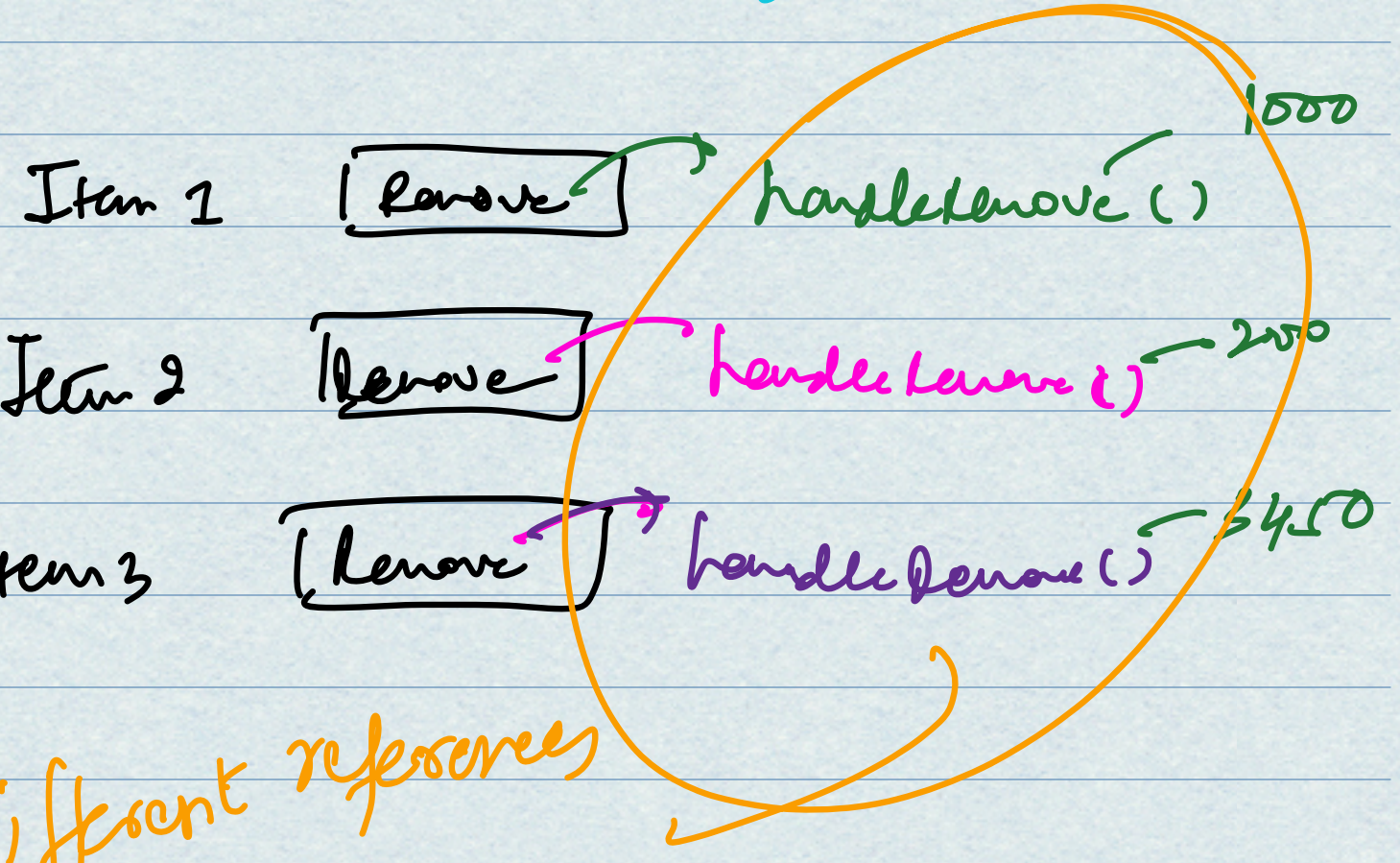
Advantages
- Simplified code
- Better UX
- Automatic code Splitting

(by Learn).

useMemo & useCallback

const sum = useMemo(() => calcSum(), [dept I])

When dep A changes, then only calcSum() be called again.



Different references

f' function

UseCallback → ◦ Look [redacted] that returns memoized version of the callback function that only changes if one of the dependencies changed.

◦ If no dependency change, the memoized will not be recreated on each renders.

◦ Same reference gets maintained.

◦ Avoids any re renders / function reference remains stable.