Chapter 05 - lets get hooked

Q: Diff bw named, default and \* export

A: export and import of modules was introduced in ES6.

named export : can export multiple values. While import you need to wrap the name in {} .

Example:

component.js

Export const title = ‘abc’;

Main.js

Import {title} from ‘./component.js’

Default export:

One file can have only one default export. It export only one value which is upto you - can also export object of values. For import you can directly use the value exposed no need to use {}.

Example:

Component.js

Const title = ‘abc’

Default export title;

Main.js

Import title from ‘./component.js’

\* export:

This export is used when you want to import all the exported values in one file.

Example:

Component.js

Export const title = ‘abc’;

Export const name = ‘aa’

Main.js

Import \* from ‘./component.js’

Const titleValue = title;

Q:What is importance of config.js file

A: it can also be named as constant.js file. This is a place where you can store commonly used hard coded values. Its better to store hard coded values at one place assigning it to a variable so that whenever the value changes we don’t have to update in a lot of places as industry practice.

Constant.js

Export const appTitle = ‘Title of the app’

Q: What are react hooks

A: In React version 16.8, React introduced a new pattern called Hooks. React Hooks are simple JavaScript functions that we can use to isolate the reusable part from a functional component. Hooks can be stateful and can manage side-effects. Hooks allow you to reuse stateful logic without changing your component hierarchy. This makes it easy to share Hooks among many components or with the community.

### React provides a bunch of standard in-built hooks:

* useState: To manage states. Returns a stateful value and an updater function to update it.
* useEffect: To manage side-effects like API calls, subscriptions, timers, mutations, and more.
* useContext: To return the current value for a context.
* useReducer: A useState alternative to help with complex state management.
* useCallback: It returns a memorized version of a callback to help a child component not re-render unnecessarily.
* useMemo: It returns a memoized value that helps in performance optimizations.
* useRef: It returns a ref object with a current property. The ref object is mutable. It is mainly used to access a child component imperatively.
* useLayoutEffect: It fires at the end of all DOM mutations. It's best to use useEffect as much as possible over this one as the useLayoutEffect fires synchronously.
* useDebugValue: Helps to display a label in React DevTools for custom hooks.

Q: Why do we need useState hook

A: useState maintains the state of the react app i.e tracks state in functional component.

useState returns array of arguments - stateVariableName, setStateFn.

Can provide default value of stateVariable inside useState.

Const [searchTextVar, setSearchText ] = useState(‘searchDefaultText);