Ep:10:Tailwind  
  
For styling a page we have different method:  
1>Native css i.e, External css

2>Scss and Sass  
3>Inline Css :process of inline css is heavy for js.

4>styled component:writing css in javascript, just like writing html like syntax in js i.e JSX.

5>component libraries :Material UI, chakra, ant, baseUI  
> component libraries can create consistency in UI.  
>we can use multiple libraries in single application, but we shouldn’t use it as it lead to inconsistency of UI

Cons of comp. libraries:  
>by using comp. libraries we loose control over personal customization ,as it restrict to do that.  
  
Pros of comp. libraries :  
>It makes developer work faster , easy to handle.  
  
Tailwind:  
pros:

>Reusability, easy to use, easy to debug, code size is small, faster development, bundler size is small [bcoz tailwind only include those files which is used in project, not put all classes] , flexible UI (customizable), no need to go back and forth as tailwind let you apply styling on the same under className.  
  
  
cons:  
Too much classes, high learning curve[people take to understand so less productive during learning phase] , code look ugly bcoz of too much classes.  
  
//media query:  
md:

Sm:  
  
//w-[]  
>within these [] brackets we can give dynamic values.

//Tailwind VS code extension:  
>tailwind css intellisense  
  
//Postcss  
  
>a tool for transforming css with js  
>compiler  
  
//.postcssrc  
>>consist configuration for Postcss.  
>tell parcel when bundling things up, understand tailwind with the help of postcssrc  
  
//tailwind init:  
>configuration for tailwind.  
>it gives tailwind.config.js file  
>it need list of file extension in ‘content’ key of file, that where tailwind is gonna apply.

//3 classes import in index.css

>@base

>@components

>@utilities

//Ep :11  
  
React has two layer:  
1>UI layer

2>Data layer:data layer consist of state ,props,local variable,whatever data use in application.  
  
Higher Order Component:  
It is a function which takes a component as a i/p and returns an enhance(add some extra feature on the same component) version of the same component.  
steps:  
1>create HoC

2>import HOC

3>call HOC and pass the comp.

4>use new created comp and pass the required props if any.

It’s a pure js function as it’s not modify/change existing comp,it just enhance the comp by adding some extra feature on the top of it.

>UI handle by state and props  
>props:it pass data b/w one component to other component.  
>state:it’s limited to its functional component but we can pass it through props.  
  
//Props Drilling:  
>Passing props from a parent to its child to child and so on.  
(passing data from parent to child is one way)

>React has one way data flow.

=>for passing data/props child to parent:

>use custom hook or localstorage.  
  
Note\*\*:React context is a solution of Props drilling, as we sometimes don’t need to pass data from multiple parent, in such case we keep data in central space known as “Context”, by using context we can use data in any application just by creating, importing, using or providing(using < .Provider> component it basically override/update value of context and pass under “value” key as a prop ,it use by wrapping other component where context data need to override ) or consuming(using <.Consumer> inside a ‘class’ comp it takes call back func. Inside {} nd this curly braces has data of context and this call back return jsx  
i.e, { (data)=> { <h1>{data.name}</h1>}}) .  
  
required hooks and componemts of context:  
1>createContext(create\_context\_variable),

2>Import created\_context\_variable  
  
3> const{name}= useContext(pass\_context\_variable)

4>< created\_context\_variable.Provider value={{ ,}}> //wrap other comp. with this comp.

5>< created\_context\_variable.Consumer> {(data)=>{JSX}}

React context has few hooks and component and it comes under ‘react’ library.

>substitute of Context are mobx, flux, ngrx,redux  
>A piece of data that can be accessible or update in anywhere in app/component . so no need to pass that data as props[i.e no need of props drilling]

Q// what to put inside context?  
ans. A common data which use in multiple places.

Lifting the state up:  
>Taking control of child nd give it to parent.  
>Parent control common children.  
>common child has their own state and props ,however in lifting state case ,they controlled by their parent.  
e.g, Accordian  
  
//controlled component: when parent control children component.  
//uncontrolled component: when a children or any component control themselves .  
  
//For optimizing app , check ‘profiler’ and ‘component’ React chrome extension in dev tool  
  
  
  
//Q,nested context?