Scaling → scaling in database refers to the ability of a data base system to handle increasing workload

we have two types scaling

① Vertical scaling ② Horizontal scaling

Date

Vertical scaling → Vertical scaling involve increasing the capacity of single server of in a database system.

* This can be achieve by upgrading the hardware component such as increasing the ram, or increasing the capacity of CPU.

Horizontal scaling → Horizontal scaling involve adding more servers for distributing the workload to other machine.

Note relational database only supports vertical scaling and NO SQL databases supports both vertical and Horizontal scaling.

types of NOSQL databases → ① document type database (MongoDB couchDB) 2. Key value store database (REDIS, DYNAMODB) ③ column oriented database (CASSANDRA, HBase)

④ graph database (⑧) → Neo4J, GRAPHDB.

NOSQL → Any database which doesn't follow relational database or SQL is known as NOSQL database.

MongoDB → mongoDB is a cross platform, Document oriented database that provide high availbilety and scaling, mongo DB work on the concept of collection and document.

DataBase → database is a physical container for collection.

* Each database gets it's own set of collection.
* A single mongodb server as Typically multiple database

Collection → collection is a group of mongoDB document. It is an equivalent to RDBMS table.

* In mongodb we can store data in the form of JSON BSON, XML Document.
* A collection exist within a single database.

Document → A document is a set of key value pairs and document have dynamic fields.

* Dynamic schema means that document in the same collection do not follows are do not need to have the same set of field are structure.

key/value pairs

{ name : RaGHU
last Name : Ram }

JSON → (JavaScript object Notation) JSON is a lightweight format used to exchange store data.

* It is design to be easy for human and machine to understand
* JSON uses a simple structure of key value pair. JSON
* Data in JSON is enclosed within { } curly Braces and consists of key value pair.

Datatype of JSON → Number, Boolean, String, Object, Null, Array

JSON syntax → * keys and strings must be enclosed in doublequote (" ") ("key" : "value")
* Commas are used to separate key value pairs and element within arrays.
* JSON must always be enclosed within curly braces { } each key is follows a colon (:) and its corresponding value.

Spiral

Date

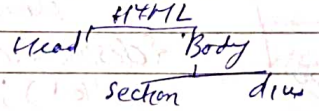
Date

React — React is developed by Jordan Walke in 2011
- Facebook — News Feed feature

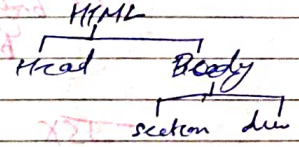
React-Dom →
maintain real time copy of virtual Dom in memory (Reconciliation) and compare V-Dom & Rdom to check update, if there is any dynamically insert the component / element.

* (Diffing algorithm) → used to compare V-DOM & R-DOM

Real DOM



Virtual DOM



React → current version V/S

Characteristics → React is fast
- React uses component based approach
- It break down UI into small-small pieces because small pieces are easy to build & maintain (components)

Virtual DOM
- It is real time copy of Real Dom.
- In Real, we perform updation on virtual Dom instead directly updation Real Dom
- Virtual Dom is faster because if update only updates done in virtual Dom to the real Dom

React — React Characteristics →
- It flows Flux-architecture

(
- We can send data from parent component to child component ~~one way~~ But we can not send data from child to forward

* Easy to learn → library.

Spiral

React component

arr=[1,2,3,4]

one way data binding

arr=[10,20,30]

HTML

```
<p class="ai">
  Hello
</p>
```

JS

```
let p = document.createElement("p");
p.setAttribute("class","one");
p.textContent = "Hello";
let body = document.body
body.appendChild(p);
```

React

```
import React from "react"
import ReactDom from "react-dom/client"
let element = React.createElement("p",
{className:one},
'Hello');
```

JSX

```
(JavaScript XML)
<p className="one">
  Hello {5+5}
</p>
```

XML → Extensible Markup language

Node JS
→ Commonjs or Vanilla JS
Older version JS → Babel

Browser
→ ECMA Script

Babel
→ It is transpiler
→ It converts ECMAScript code to Commonjs code.

Node JS
→ commonJs or Vanilla JS

Older version JS ← Babel

Browser
→ ECMA Script

Babel
→ It is transpiler
→ It converts ECMA Script code to Common JS code

React library
react-dom library
Bo

NPM →
× node package manager
× It is used to manage package and can perform install, uninstall etc

Package
react lib
react-dom
features

Registry

NPM
→ command line interface
It provides access to largest registry of package
→ npm
CLI → By using CLI, one can instruct the npm to perform installation, uninstallation etc
Registry → It is a open source registry from which npm borrows package and install them for development

any: npm install package-name] By this command package is installed globally in our system

To install package locally: npm install package-name] By this command package is installed locally

Date

= npx
* node package execute
* It install package locally without, without installing them in global spell (c)

create-react-app/vit = Tools to setup react-project

Node → Run time Enveroment
npm → To manage package in projects
└ npx → used to manage package locally
create-react-app/vit/ => Tools to setup react-project

└ npx - create-react-app First Project => In terminal
└ First project → Folder/Directory
- node-modules - Here, list of all packages is available that we are using in project.
- package-lock-json → Here detailed information of package that
→ Public → Here we store public files like index.html, logo
→ Src → Complete React code

Install
reactlibrary
react-dom lib
react-scripts
webpack

update
(HMR) HotModuleReplacement

server webpack-dev-Server

HTML/Website entry
Project
index.html

React/JS
ReactDOM.createRoot(document.getElementById("root")).render(<App />)
Src
index.js
App.js
render route Asset

Date

Node.JS
→ Common JS
Import
└ require (package-name)
Export
└ module.exports = {" }

Ecma script/
Import
Default import
import clock from "X.js"
Named import
import {D,C} from X.js

Export
└ DefaultExport
└ In a file, we use default Export only once
└ By using default export, we can export single component
Example export default A;
Named Exports
└ In a file we can use named export multiple time,
└ By using named export, we can export multiple component.
Example Export {C,D};

<HTML>
<Body>

{Root}
</Body>
</HTML>

import React from 'react'
import ReactDOM from react-dom/client import App from App.
ReactDOM.createRoot(document.getElementById("root")).render(<App />)

Date

Date

JSX

- JSX stands for Javascript and XML.
- It is syntax extension for Javascript.
- while working with react we we will JSX to creat component. because it's easier for developers as we use JS, HTML, CSS like sentence.
- React component return JSX which will be converted into react element

Rules to use JSX :—

- ReactComponent can return single JSX Element If we want to return multiple element, we have to wrap all elements inside a single parent element.
- We use Javascript inside JSX we have to use curly braces in between curly braces we can write javascript code. like variable, expression, etc.
- In JSX It is mandatory to close all tags.
- In JSX we can not use keyword like class, for because these are reserved word in javascript instead we can use className, htmlFor respectively.
- file in which we are using should be saved with '.JSX' extension.
- Component name should start with capital letter
- file in which we are using JSX it with start capital letter

Component →

- Component is Reusable piece of UI.
- They are independent piece of code
- In React, Big UI user Interface is broken down into small pieces called as component.
- We can create Component by using class or function

Date

* States and behaviours available inside component can not accessed from outer environment.
* To call a component we have to use <ComponentName> </ComponentName> convention
* We can call another component in inside other component

Class

→ Class is a function
* class is blueprint of object
* To create a class, we have to use "class" keyword
* class act as a blueprint or structure by which we can create object of type
* class is non-primitive datatype
* class has a special method, which get executed automatically when we create object method called as constructor
* To create object we have to use 'new' keyword and constructor

- Private
Protected

Constructor

* Constructor is a special method, which get called automatically when try to create object
* Constructor method is called immediatly, when when creat object.
* It is the first method which get executed
* It is used to initialize the ~~st~~ states (variables) of class
* It does not return any value
* To create constructor, we have to use constructor keyword.

Date

Extends

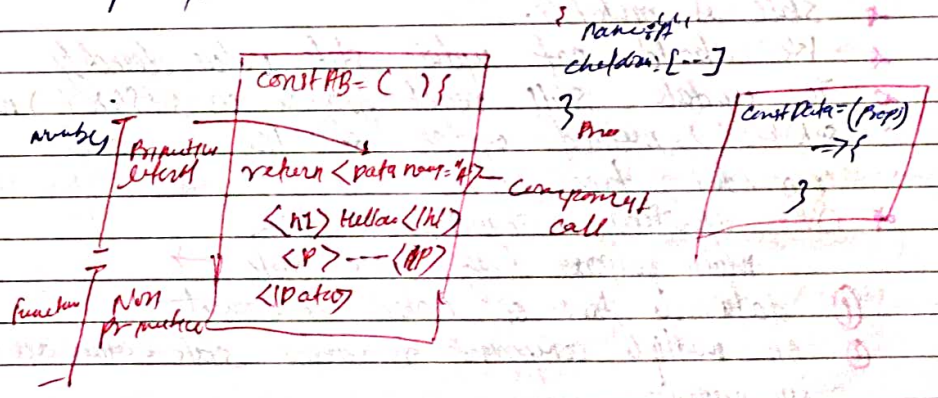
It is used to inherit states and behaviours of parent class to child class

Super

* It is used to call constructor of parent class;
* super() should be the first statement
* when we are inheriting it is mandatory to call constructor of parent class
* At the time of call, we can pass data to the parent constructor

Static

* If we use static before state or behaviour, it can't be inherited
* Because static keyword will remove that state or behaviour from prototype



Props

* Props stands for properties.
* In react props are use to parse data from one c
* Props are immutable.
At the time of component call

- breakup complete

Signal

Date

available in between that all those children will be passed to the component prop. React provide special key name as children where all children are store.

State → Object → It is a object.
- It is use to store data inside component
- We usually store object or null.
- State is an object
  ~~We usually store~~
- State object is use to store data or object with in component.

- We cannot access component state from outside.
- State is use to store data in isolate environment.
- React provides build thing state object for class base component.
- state is mutable.
- We should not update state value directly
- To update still react provide us setState() method
  setState() method update state and rerender the UI.
- State updation a synchronous task
  main purpose of have state →
  ① data is store a isolate environment
  ② on multiple component re-renders state values all reading in memory.

Note
- Where ever value of props or state changes component will re-render every time
- The component props state changes that component will re render

Date

c. The component in which props value change will re-render.

Class Component
- Know as stateful component
- It has in build state object
- It is mandatory create constructor
- Need render() method to display element
- class based component build in Life cycle
- To create class component we need to inherit from React. component class

Function Component
- Know as stateless component
- It does not have built in state object
- constructor not require
- render() method not require.
- Does not have built in life cycle method
- To implement state and lifecycle method in function component we have "Hooks"

Life Cycle of React

① Mounting phase
- method → constructor()
- static getDerivedStateFromProps()
- render()
- componentDidMount()

② Updation phase
- static getDerivedStateFromProps()
- shouldComponentUpdate()
- render()
- getSnapshotBeforeUpdate()
- componentDidUpdate()

③ unmounting phase → componentWillUnMount()

④ Error boundaries
  └ It is used handle component error without breaking complete

Mounting
constructor()
static getDerivedStateFromProps()
render()
componentDidMount()

Mounting
static getDerivedStateFromProps()
shouldComponentUpdate()
render()
getSnapshotBeforeUpdate()
componentDidUpdate()

Life cycle of React →

React provides lifecycle method to class base component these method will get executed in an sequence
React component lifecycle as four phases
Mounting phase
Updation phase
Unmounting phase
Error Boundaries

Mounting phase

In mounting phase we insert create or add element in the Dom.
In this phase following method get executed

* constructor() →
  * It used to initialize the state we
* we have to use super call statement.
  Super — It is used used to initilized the parent constructor method.
  - It help to inherit method from parent (inherit method from React.Component class)
* We should not call setstate inside the constructor
* Business logic is not allow.
* Side effect by API call are not allow.

Static getDerivedStateFromProps() :—

* It is called after a constructor in mounting phase
* It return null or new state.
* Here we can set the state based on initial props
* It is state as argument and return object with the change in state.

render() :-

* It is use to derived UI logic.
* Business logic or side effect are not allow.
* If we want to display something of UI we have to render it from render() (we return JSX elem

ComponentDidMount().

* It is called when component is render to UI
* Here we can write business logic or side effect.
* Here we can use setState method.

Unmounting phase

* Here we update mounted element in Dom.
* Following method get executed in updation phase

static getDerivedStateFromProps() →
shouldComponentUpdate():
render():
getSnapShotBeforeUpdate():
componentDidUpdate()

static shouldComponentUpdate() →

* This method decide component will be updated or not.
* This method return boolean value. If it returns false component will not get updated.
  If it returns true then only component will be updated

Date

* By default It returns true

getSnapshotBeforeUpdate() :-

* In this method we can get the current state of or
* props and we can get the new state or new Props

ComponentDidUpdate() :-

* This method get executed when component is updated in DOM.
* Where we can use sideEffect or Business logic.

UnMounting Phase :

* In this phase we remove or delete element from the DOM.

Following method it get exec in this phase.

ComponentDidUnMount

This method get executed when component is remove from the DOM

Error Boundaries -

* Error Boundaries are features in React.
* with help of error boundaries we can catch component which error and can handle those error with out Breaking the code UI.
* It work similar try/catch.

Conditional Rendering

* Conditional Rendering is a technic in React where we render component on some condition.
* We can use following phase to render conditionally
  (i) If else
  (ii) switch
  (iii) else if
  (iv) Ternary Operator (v) Short Circuiting (&&)

Date

React Fragment => If a React component can multiple element (If we are using JSX multiple element should be wrap a single (Div)

* It means we have to wrap multiple element using an extra elements which will be added to the DOM.

React fragments let us group the list of children without adding extra node to the DOM.

React provides fragment to use it we can destructure from React

import React, {Fragment} from 'React'

① <React.Fragment> </React.Fragment>
② <Fragment> </Fragment>
③ Short syntax < > </>

When we use 1 & 2 way to use fragment we can use key are attribute

React key (Prop) -

When we are creating element dynamically at a time it is recorded to pass key prop to the each element

React key prop should be unique for each element throughout React application

React uses key prop to find element that has been updated and with the help of key prop react compares element of virtual DOM and Real DOM to perform updation

Date

React list:-

* When we use map method to generate element dynamically map method return array of react element & efficient as React list.

JS
└ HTML
└ Attach Event
└ Pass Attribut

HTML
1 <p onClick = alert(' ')>
<p onClick = alert(' ')>

React/JSX
<p onclick = {'hello'}>
<p onClick = {handle}>

React synthetic Events

handle = () => {
e.preventDefault();
alert (for submit);
}

React Synthetic events =>

* React Synthetic events are wrapper around Browser native event
* If we use browser event use directly it may or may not work in different browser because different different browser.
* Because different browsers 'api' different event names
* To handle this situation React has it own event system as a called a React Synthetic events

Date

* By this React make sure all events should work property across different browsers

We attach synthetic event as a attribute
Event name should follow camel case
Synthetic We can pass — Reference or an expression.
We can attach multiple synthetic event to a single element
ReactIcons [npm i react-icons :)]

Date

Class based component

complex
- readeath to create component

[It has in built state object
- Lifecycle Methods

Functional based Compo
→ Easy to work
missing state object
- Lifecycle method

React V16.8 | Advance React

Rule to use →

* We should not call hookes inside loop class conditions and in plain JS

Hooks

Hooks are introduce in React version 16.8

* Hooks are javascript function.
* Hooks are introduce in react so that we can implement state object and life cycle method in function based component.
* working with function base component its not complex as compare to into class based component
* with the help of hook we have implemented state in functional component with the help of use statehook and we have implemented component did Mount component Did update, and Componentwill UnMount with the help of use effect hook.

Rules to use hooks

* Hook should be called at Top level · it means behave to call hook at same level of the return keyword.
* We should not call hook inside loop, class compo

Date 14/10/2023

, condition, in plain Javascript function.

* We should not call hook inside another hook (Recomended)

useEffect =>

componentDidMount

→ It will get executed when component mounted added in Real DOM

componentDidUpdate

→ It will get execute when component updation is occur
→ Here, we can add condition accordeng to which it will get executed.
(should component update)

componentWillUnMount

→ It will get executed, when component is remove delete from Real DOM.

useEffect

→ It is hook
→ It use et we have to important & destructure it from react library.
→ It is used to implement componentDid Mount, component DidUpdate and componentwillUnmount. (ShouldComponentUpdate)
→ use Effect hook is used to write sideEffects.
  └ like apicalls, Timers etc.
→ useEffect accept 2 arguments
  ① Callback function
  ② Dependency list

Here callback function will get executed on 1st render every time.
And after 1st render, callback function will get executed only when value of dependency list changes.

Dependency → It is a array of dependency (variables) on which execution of callback function depend.
Callback function will get execute any time when value of dependency changes.
→ Dependency list is optional.

② componentDidMount [Syntax useEffect(function, [d1, d2, d3])

useEffect(() => {}, [])
empty dependency list

→ Here useEffect will execute only one when component is mounted on RealDom.
→ It will not run further because there is empty dependency list.
(useEffect(function, [....]) optional.)

# useEffect
componentDidUpdate()
- updation phase (N Times)
- when ever there is change in component state or prop, it get executed
- Its execution can be controlled by →
  - shouldComponentUpdate()
    - In this, we can write logic and update acc. to which componentDidUpdate() will run.

① useEffect(() => {}) → No dependency
→ It means execution of useEffect's callback does not depends on any dependency
→ It means when ever there is change in component it will get executed.

~~useEffect(() => {})~~
useEffect(() => {setCount(count())})

② useEffect(() => {}, [d1, d2, d3 ... dn])
dependency
→ Here, callback function will get execution only when there is change in given dependencies in dependency list.

#useEffect
componentWillUnmount
- It will get executed when component is removed from realDom.
In this we write cleanup logic like closing connection removing eventlistener etc.

Date

```
useEffect(() => {
    return () => {
        clean up code
    }
})
```

* Here use effect will return a function only when component is removed from UI.
→ So If we want to perform operation after removing component from UI we can write it in function that will be returned.
→ We use for clean up.

write syntax for use syntax as component did mount component update component.

component did update

```
useEffect(() => { }) No dependency

useEffect(() => { }) [data]   dependency
```

component ~~will update~~ will mount

```
useEffect(() => {
    return () => {
        clean up code
    }
})
```

Here ~~use~~

component did mount

```
useEffect(() => { }, [])   No dependency
```

Here useEffect will execute only one time
It will not execute futher because dependency is empty

Date /

useEffect did update :-

syntax `useEffect(() => {}, [x1, x2, x3])`

* In use Effect hook if we want to use it as component did update then we have to provide dependency where we can pass multiple dependency
* dependency is not empty

~~useEffect~~

useEffect will component

```
useEffect(() => {
    return () => {
        clean up code
    }
})
```

* Here use Effect will return a function only
* when component is removed from UI.
so If we want to perform operation after removing component from UI.
* we can write it in function that will be returned
* we use for clean up.

Props drilling

* In React props drilling is a situation where we pass data through props to the nested child.
In React component tree to the last node of the tree
* If we have multiple nested child its very difficult to pass data so to handle this type situation React introduce context API which resolve props drilling situation.

Date

Date

It is by which we can store data between content directly without key to pass data through every component in component tree

Context API is a solution for props drilling situation.
In context API we maintain global state store where React applications state are available and any component can access it directly

Context API consits of →
→ Context
→ Provider Component
→ consumer component.

context Object →
To create context Object we have to use createContext() function this function return a object where data, provider component and consume component is available.
At the time of creating context we can provide default value which will be use if provider value is not accessible by the consumer.

```
const Context = createContext(0);
```

Provider Component →
Provider component will pass context data available to all its nested children component

Date

Provider object accept a value props where we can pass the data which will be consume by the consumer component of consumer.

Consumer Component

* Consumer component is use to access data provided by provider component
* Consumer component expects a function where data will be recive in parameter of that function
* Only consumer component can access the data
* If consumer component is not avail to access the data provider to provide component default value will use

Note → when we have ~~more~~ multiple context in react application its difficult to consume the data because ~~not~~ nested consumer component behave to create.

* To solve this situation react provide us ~~use~~ use Context book.

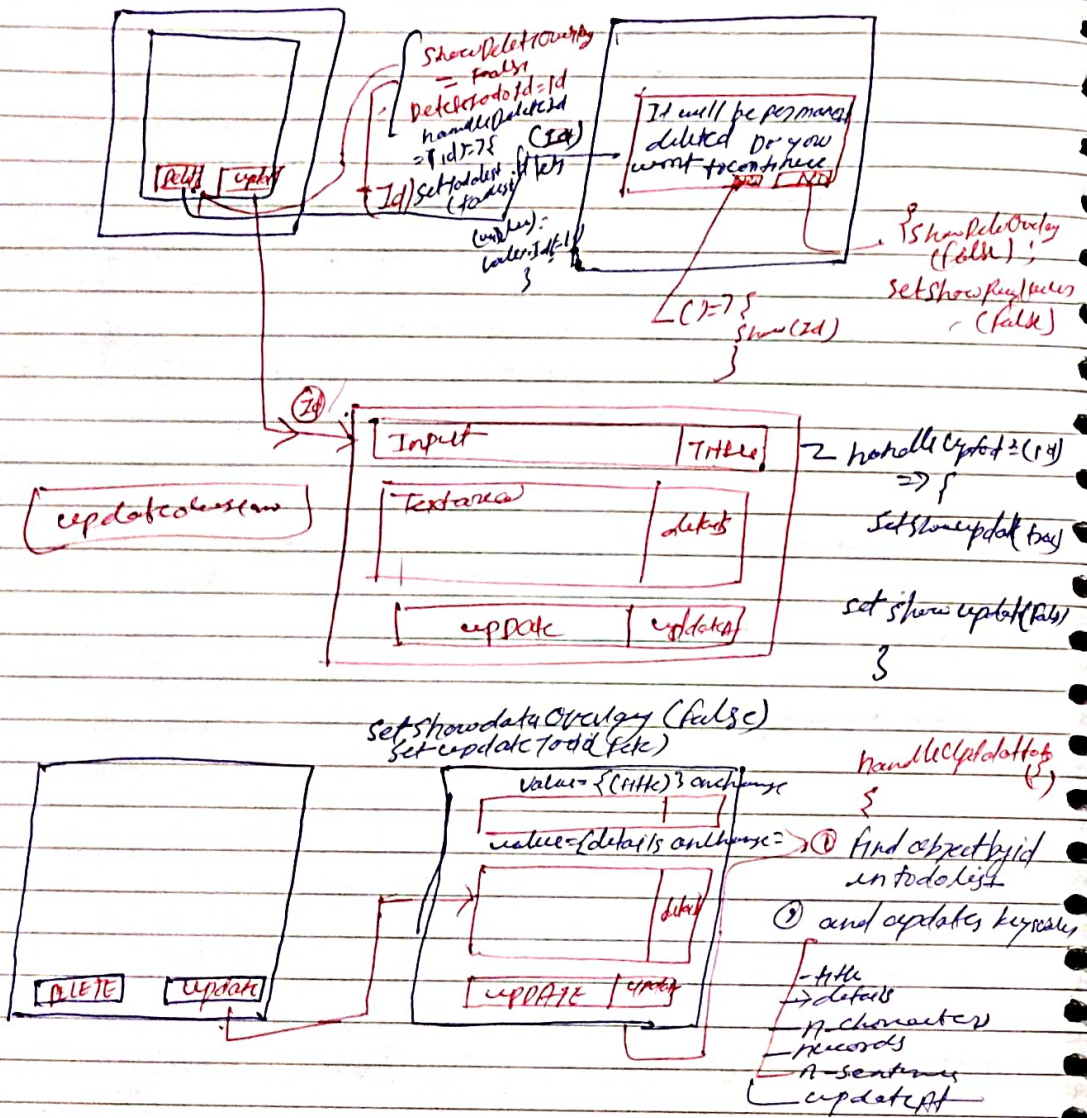
use Context Book →

* use context Book it use to access data provided by context.
* It takes context object as its argument and return current value of the context.
use Context simply the process of accessing context data.
* If we have multiple context in react application its easy to consume data of multiple context with the help of use context book.

Date

Date

Delete Design



Date

Parsed

JSX — Javascript XML

* It is template language, & its is javascript.
~~<div~~ Hellow
createElement("div",{className:"mainDiv"},"Hello")

div

JSX	How it internally works
div	

* It allows us to write html code in react js.
* It is combination of HTML and Javascript.
browser is converting JSX in browser understandable Javascript

Rules for JSX

* JSX must be closed (each an every element should be ~~each~~ close. If you dont close any element you will get error!) ~~error~~
* Even its paired and unpaired tag but they must be close.

paired
example

unpaired
<input ... />
<hr />

* ~~If~~ JSX element must be wrapped (If we are using more than one element, they should be wrapped within a parent element.)

Example
<h1></h1>

<section></section>

Date

we can write element use fragment to corapped elemt
(if you are sending element more than one
you can wrapped them between fragment.
* We use fragment to avoid extra node in dom
Example < > </>
empty fragment
with fragment name → <Fragment>
<h1></h1>
<h2></h2> </Fragment>

* You can't use class or for attribute (for and class are use in both element html and javascript as reserved keyword or attribute. which can create naming comp conflict to avoid that be we'll use

class attribute → className
for attribute → htmlFor

If we are using inline css we need to write css in the from objects

<div style ={{width: "101.4", backgroundColor: "red"}}
>

Jsx Expression :—

With the help of Jsx expression we can write javascript inside html.

syntax let name = "Prasad"
 my name is {name}
jsx expression

* We can write varible, javascript opretion like arithmatic operation may thing in between

Date

Jsx operation expression.

* We can't a write looping statement like for, foreach etc in between Jsx expression. instead of that be we need to go with any of the three method that is map filter and reduce.
* We can't a write if else else if switch switch any of the conditional statement in between Jsx operation except ternary operator

Components -

These are core building block of any react application which contains reausable code and which works independently
* The aim of component is to devide the application in multiple reausable blocks of code.

* There are two types of component.
(i) class based component (ii) function base component

class	function
① These are javascript classes that extends react component and has render method that returns JSX Element	* These are javascript function that return JSX element.
* It has inbuild built in state mangment using this.state	* It uses react hooks to mange manage state. ex useState hook
* It has built in life cycle methods. render componentDidMount() componentDidUpdate()	* If doesn't have default life cycle method but we can achieve them with the help of hooks use Effect hook
* It is difficult	* It is easier and recomended

Date

State → To update the things and display them dynamically on web page we need state object.

* State are defaultly present inside class based component that why the class base are called as state full component
* In class base component to update we have to take help of set State.

State is not defaultly present in function base component, that's why they are called as state less component.

We can access state in the function base component with the help hooks. ex useState hook.

Props → Props are nothing but the properties which are used for transferring data ~~between~~ from parent component to child component

* Props follows unidirectional data flow (parent to child)
* We can't send props from child to parent

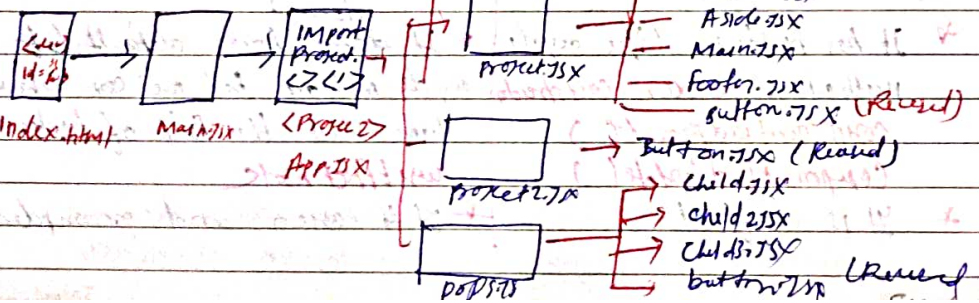
Props Parent → child
parent → child

* Props are immutable.

difference b/w

Props	State
* Props can't be modify (immutable)	* State can be modify (mutable)
* It passes data from parent to child	* It just holds the data
* We can set initial values	* We can set initial values

Folder structure



To Maintain React Project →

Date

Props.children →

* Props.children is used to access the content while declaring the child component in parent component.
* The content must be return in between opening and closing of child component while declaring.

Parent component

```
<child> <h1> Hello </h1> </child>
```

child ~~comp~~ child comp

```
const child = (Props) => {
    return (
        <div> {Props.children.Props.children} </div>
    )
}
```

default Props → When you want to pass any props from parent to child in that case if parent is not sending the props ~~that~~ and in child component we are trying to access the props. In that case instead of getting undefined or any error we can go with default props.

```
<child> </child>
const child = (Props = "Prop") => {
    return (
        <div> {Props} </div>   //Parallel
    )
}
```

Prop drilling →

App.JSX
props → Parent
props → child
props → grandchild

We want to pass some data from grandparent to grandchild

Date

* But we have to are not send able to send the data to all the component coming in between in the inheritance

* Prop drilling is not recommended in multilevel inheritance.
* We can avoid props drilling with the help of context API

Event handling Syntheticevent in React

* React uses Synthetic Event, it's similar to the dom event But there are few syntactically changes has follows

* Event name should follow camel case. Ex onClick.
* While passing the function name we need to wrap it in {} instead of codes. Ex onClick={function}
hooks
* We don't need to use addEventlisner in react synthetic event
* While writing the function name in any event Exam: onClick we should not call the function
* We can write preventDefault() to avoid the default behaviours of any action

React Hook:

In react hook are functions that let you use the the state and other react feature.
Hook are available in function base component only.
Hooks are introduced in React 16.8 version some comonly hooks are use state, use Ref, use Effect, use Context use Callback, use Memo.

Rules to write hook:-

Always Rig write or call hook on table
you can't declare, intilize hooks inside loops conditional conditions

Hooks names start with use.
Statment all nested function.
* Don't call hooks after return statement
* Conditional hooks in event handlers
* Don't call hooks inside function pass to use memo, use Reducer use effect

Date

UseState hook:-

usestate is React hook that allows you to add state in function based component
* Stat represent data that changes over time
* state is local and private to the component you can pass state as props.
* Change made in state causes causes the component to re-render
* Hooks names start with use
You can pass any data type as a initial value
Note In class base compone you can pass only single object

Syntax

let [initialvalue, valueUpdaterfunction] = useState(initial value)
stored here

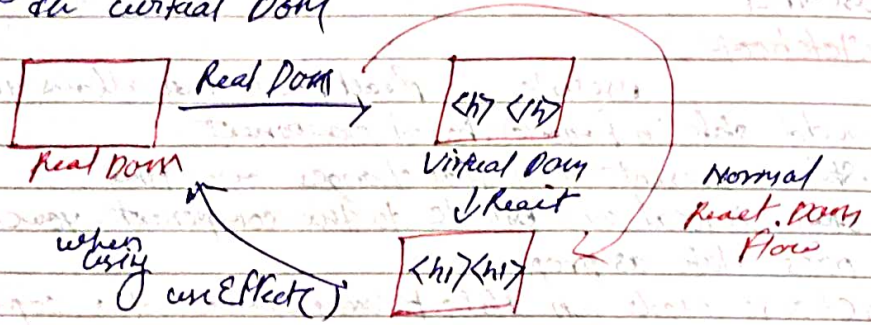
useState hook returns an array.
zero 0th index of array stored in initial value
first index of array holds the function that is use to update initial value

UseRef hook

In some scenarios we need to skip to interact of react code with virtual dom and need to directly intract with real dom.
In that case we can go with use ref hook.
for example some animation, media playback, text focus, text selection,

Date

* It is not recommended to use useRef hook because its an imperative way
* we should avoid using useRef hook because it directly interact with the real dom and skip the virtual Dom



Note

It is alternative way for document.getById() In React.

Control component and uncontrol component:

Control Component →

The component in which we use state hook to manage the state and not use useRef hook such component are called as control components.

* control component do not directly intract with real Dom.
* control component are considered as best practice and these are Faster.

Uncontrol Component →

The component in which we use useRef hook instead of use state to manage state are called uncontroll component.

Date

* Uncontrol component directly intract with the Real Dom
* uncontrol component make a application slower
* using uncontrol component is not a standard practice
* because its break the normal react flow.

useContext

It is one of the state management thing which is introduce after react 16.3 it is use to avoid props drilling

How to create context →

* To create context first we need to use create context() and store will store in one variable and export that variable.
* After that we will declare all the states and variable into that component so that we can provide them
* Now in return statement use that variable in which we have stored create context and go from this variable use provider and along with that mention value attribute which will hold the all the values are state that we need to provide.
* Here we need to mention, destructure children. After that in main.jsx we need to wrap the app.jsx into the component which is providing context.
* Now in any component of your application you can use the values provided by by context
* If they use this value we require to things → 1. The variable which is storing the context (2) use context hook
* We will store all the data in the variable return by useContext hook which is wrapping that particular context variable

global store where you can store all states and provide into entire application and in any component of your application you can access this state

Date

provide the context with the help of use context
book