**React Documentation**

**What is React** : React is an open source library, it’s not a framework.It deals and focuses on UI which is rich in ecosystem.

**Why React** : Created and maintained by Facebook and has a very huge community and much demand for the skill sets.

* Seamlessly integrate react into any of your applications.
* We also have React native for mobile applications.

**Pre-requisites** :

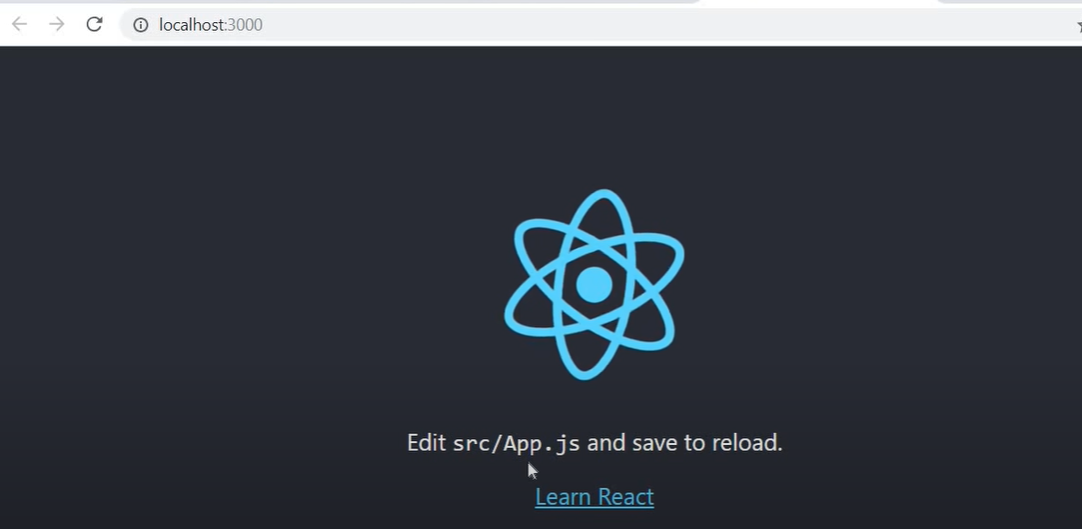
* **Required following System configuration.**
* System with at least 8GB Ram and 100 GB HDD
* Node 12.x+
* Visual studio code
* **Required following technologies knowledge.**
* HTML
* CSS
* Javascript
* ES6

**Project Setup :** Please follow the link for creating [react project](https://github.com/facebook/create-react-app)

* Open Visual Studio code and open terminal
* Open your newly created project folder
* Then execute the following commands

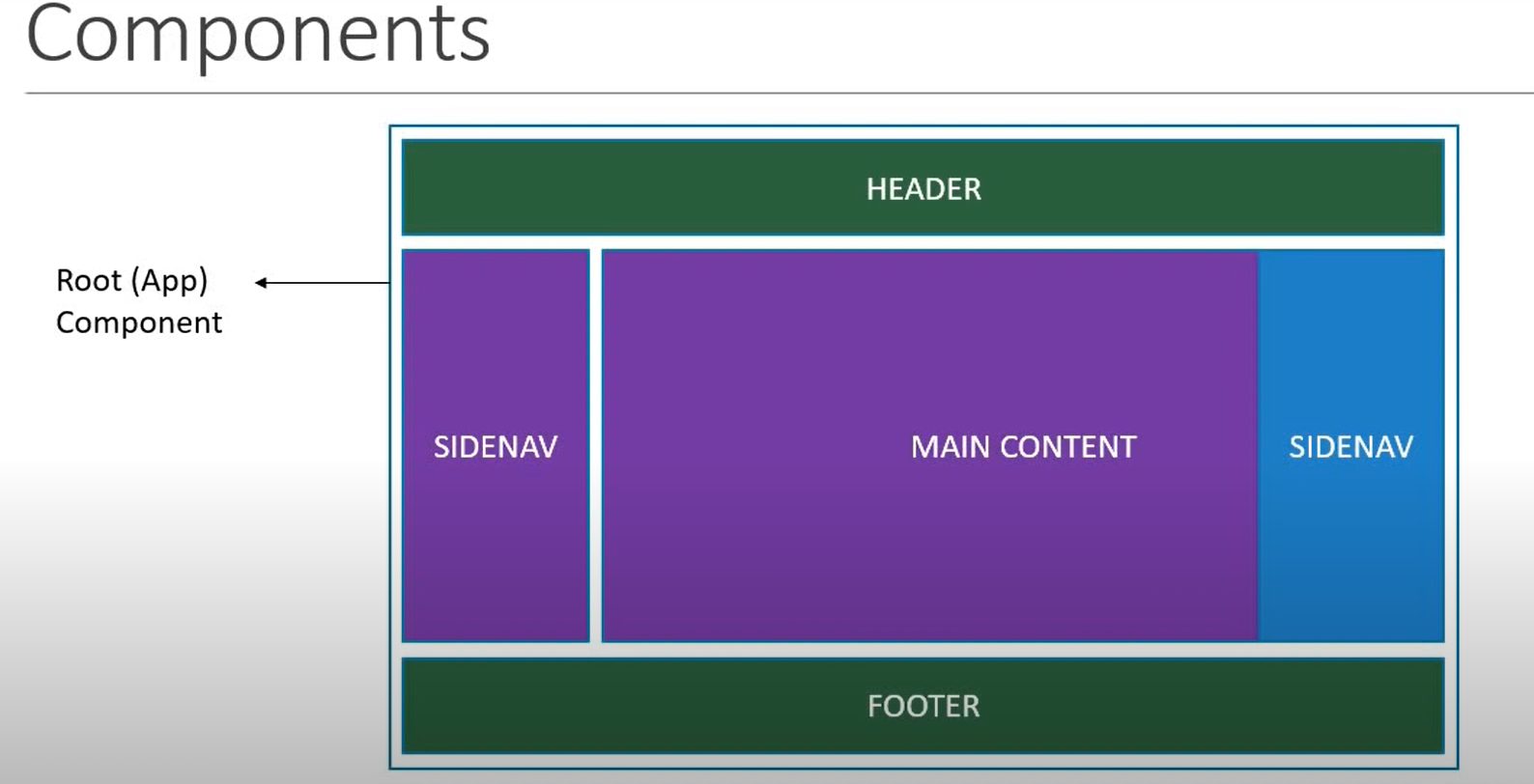
**Note** : my-app is a customized name and you can give the name whatever you want.

| npx create-react-app my-app cd my-app npm start |
| --- |



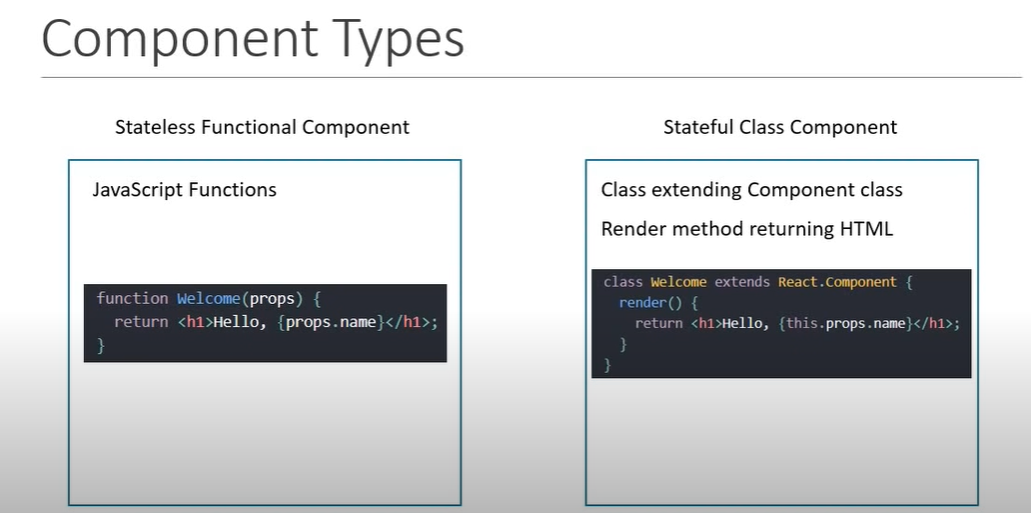
**Structure** : It's a component based architecture, which means each and individual part of UI is treated as a component.

Components are described as a part of user interface and they are reusable and can be nested inside other components.



Components are categorised into 2 types.

1. **Stateless Functional Component.**
2. **Stateful Class Component.**



**Stateless Functional Component/Functional Component** : In this method a simple javascript function is return.It almost similar to JS function.

import React from 'react';

const HelloWorldFunctional = () => {

return <h1>Hello World</h1>

}

export default HelloWorldFunctional

**Stateful Class Component/Class Component** : In this method we make use of class component to return the JSX.

import React from "react";

class HelloWorldClass extends React.Component{

render(){

return <h1> Hello world</h1>

}

}

export default HelloWorldClass

**Javascript XML(JSX)** : JSX is an extension to the Javascript language syntax.JSX tags have a tag name,attributes and children.It's similar to HTML code.

import React from "react";

const Jsx = () => {

return React.createElement(

'div',

null,

React.createElement('h1', null,'Hello Rahul')

)

}

export default Jsx

**Props** : React allows us to pass information to a component using something called props(stands for properties). Props are basically a kind of global variable or object.

* Props get passed to the component as functional parameters.
* Props are immutable.

**Props.js**

**import React from "react";**

**const Props = props => {**

**return (**

**<div>**

**<h1>Hello I am {props.name} and familiar with {props.language}</h1>**

**</div>**

**)**

**}**

**export default Props**

**App.js**

**import './App.css';**

**import Props from './Component/Props'**

**function App() {**

**return (**

**<div className="App">**

**<Props name="Rahul" language="React" />**

**<Props name="Chavan" language="Anuglar" />**

**<Props name="RahulChavan" language="Javascript" />**

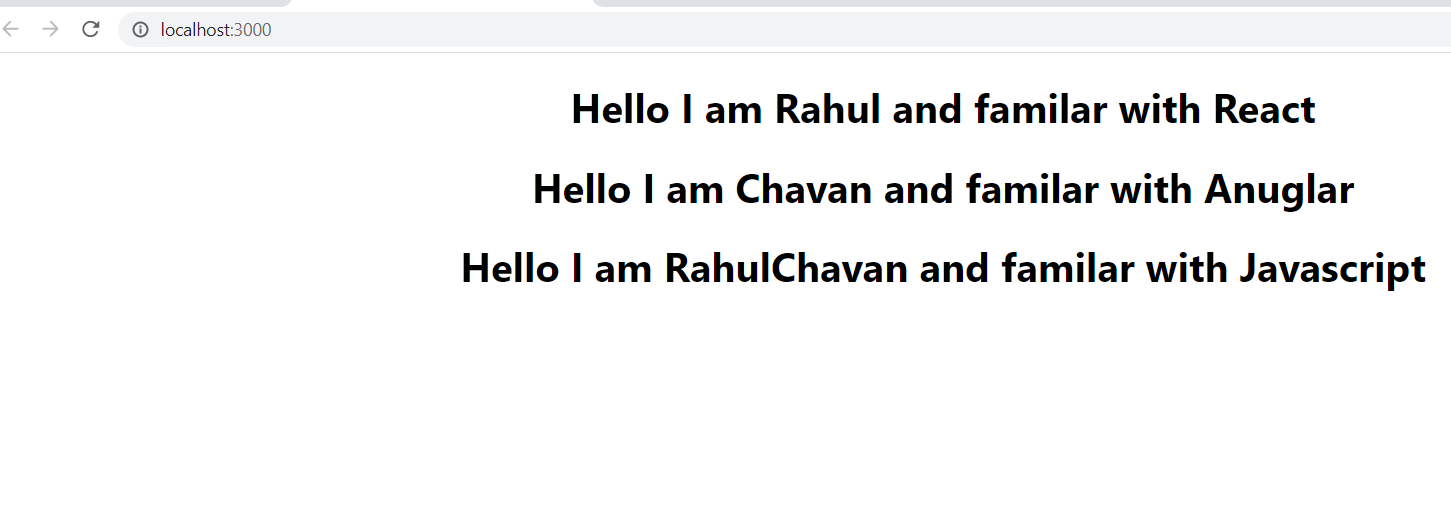
**</div>**

**);**

**}**

**export default App;**

**Output :**

****

**State :** The state is nothing but an object of a set of observable properties that control the behavior of the component.

(or)

The state of a component is an object that holds some information that may change over the lifetime of the component.

* State is managed within the component.
* Variables declared in the function body.

import React from "react";

class State extends React.Component {

constructor() {

super()

this.state = {

message : "Greeting for the day!!!"

}

}

changeMessage() {

this.setState({

message:"Thank you for visiting the page"

})

}

render() {

return (

<div>

<h1>{this.state.message}</h1>

<button onClick={() => this.changeMessage()}>Subscribe</button>

</div>

)

}

}

export default State

**setState :** setState allows you to change state in a React class Component. To change the state of a function component, you use the useState() hook.

setState() enqueues change to the component state and tells React that this component and its children need to be re-rendered with the update state.

When you have to update state based on the previous state value, pass in a function as an argument instead of the regular object.

import React from "react";

class SetState extends React.Component{

constructor(props){

super(props)

this.state = {

count : 0

}

}

increment() {

this.setState(prevState => ({

count : this.state.count + 1

}))

}

render() {

return (

<div>

<div>Count-{this.state.count}</div>

<button onClick={()=> this.increment()}>Increment</button>

</div>

)

}

}

export default SetState

**Event** : What all the operations we perform using keyboard and mouse action is nothing but an event.

import React from "react";

class Event extends React.Component{

render(){

return (

<button onClick={()=> this.AnyOperation()}>TestButton</button>

)}

}

export default Event

**Binding Event/Event Handling** : Handling the event all the action which we are performing is nothing but even handling.

import React from "react";

class EventHandling extends React.Component{

constructor(props){

super(props)

this.state = {

count : 0

}

}

increment() {

this.setState(prevState => ({

count : this.state.count + 1

}))

}

render() {

return (

<div>

<div>Count-{this.state.count}</div>

<button onClick={()=> this.increment()}>Increment</button>

</div>

)

}

}

export default EventHandling

**Styling** :Styling is nothing but beautifying your application using CSS.

* CSS stylesheets
* Inline styling
* CSS modules
* CSS in JS Libraries

**Styling** :

import React from "react";

import './myStyle.css'

class Styling extends React.Component{

render(){

return <h1 className='primary'> Hello world</h1>

}

}

export default Styling

**CSS** :

.primary{

color: orange;

}

**Lifecycle Methods** : there are 4 major types of lifecycle methods as mentioned below.

* Mounting
* Updating
* Unmounting
* Error Handling

**Mounting** : This method will be called when components are mounted.

* constructor
* static getDerivedStateFromProps
* render
* componentDidMount

**Updating** : This method will be invoked when components get updated.

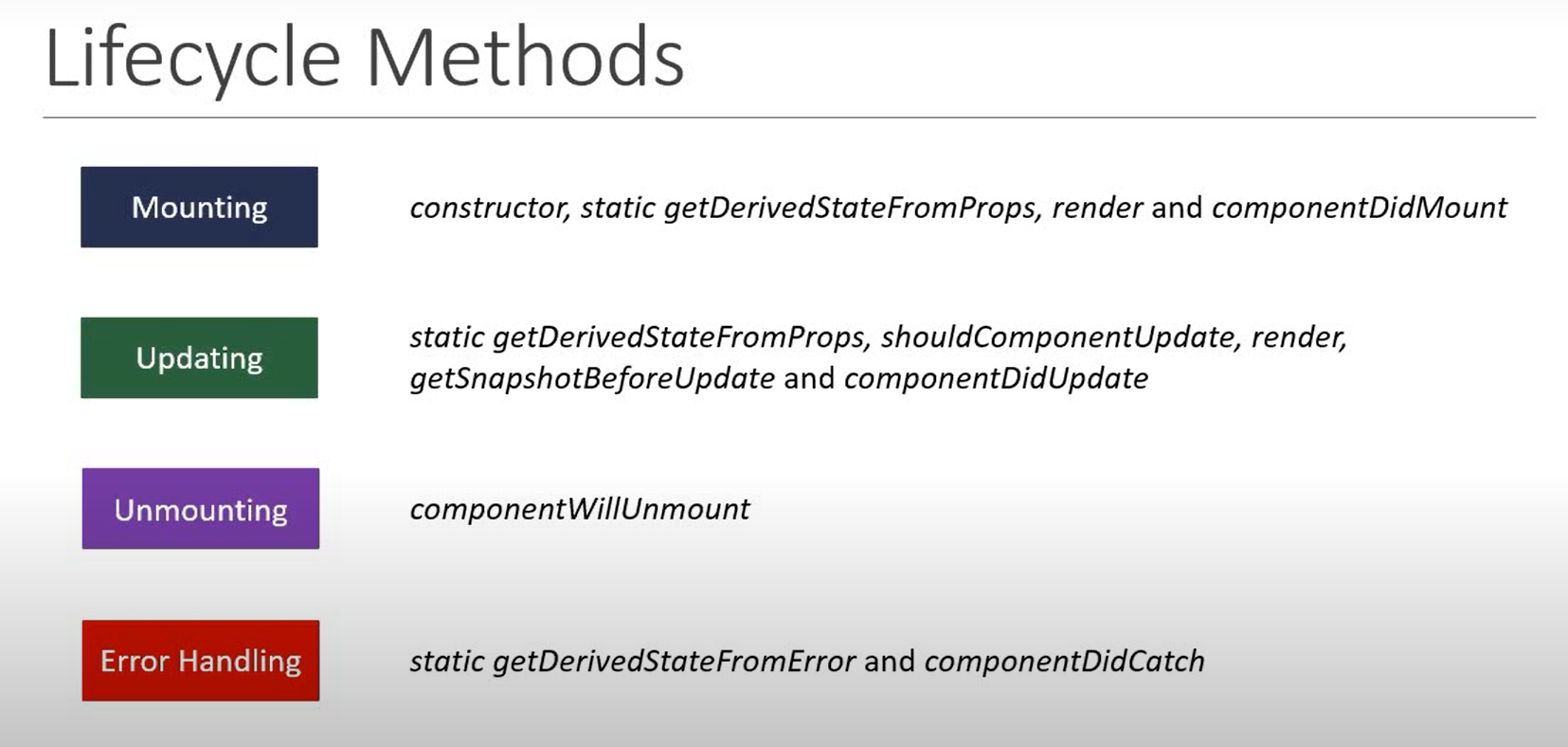
* static getDerivedStateFromProps
* shouldComponentUpdate
* render
* getSnapshotBeforeUpdate
* componentDidUpdate

**Unmounting** : This method will be invoked when components get unmounted.

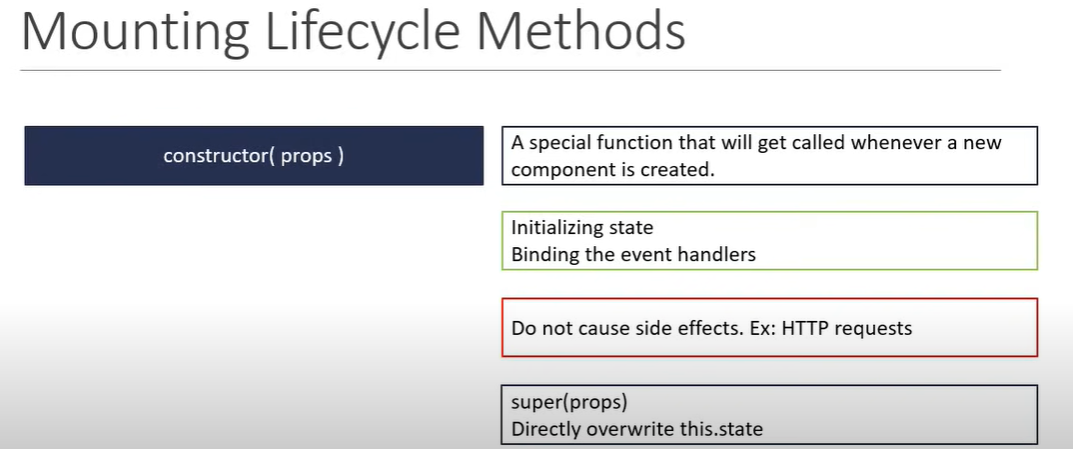
* componentWillUnmount

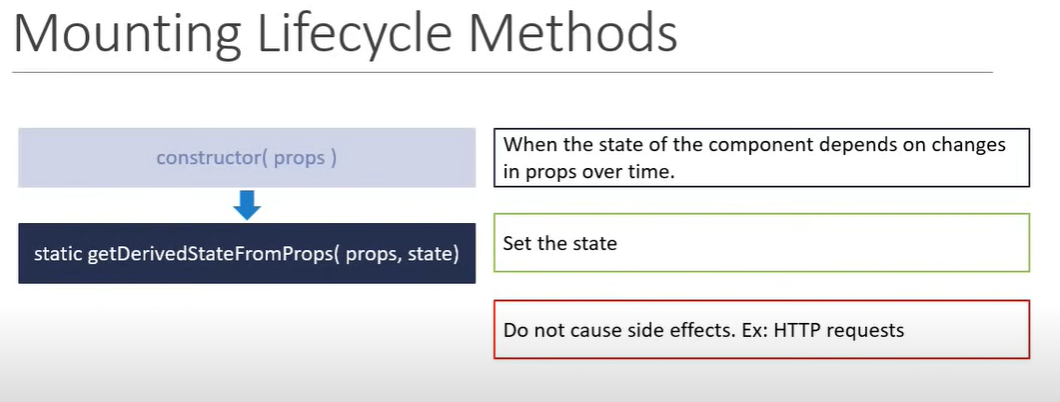
**Error Handling** : This method will be invoked when any error occurs in the program.

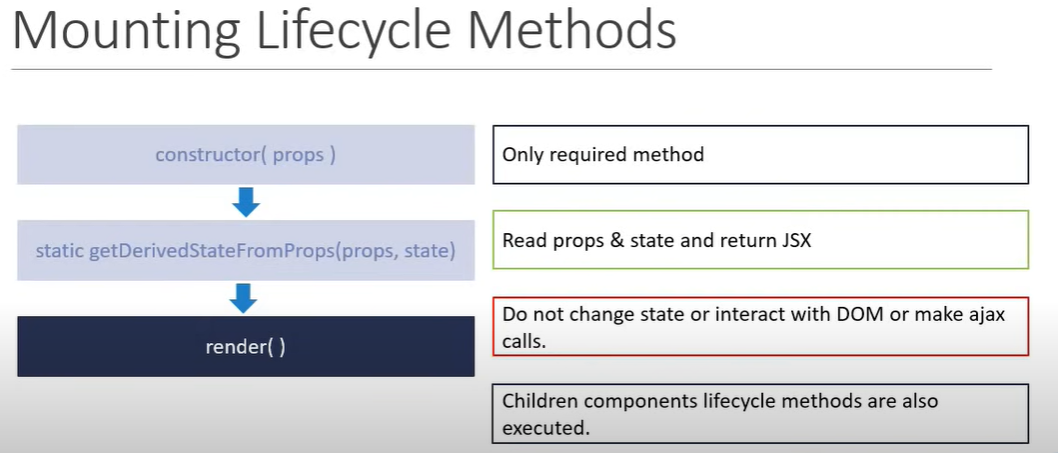
* static getDerivedStateFromError
* componentDidCatch

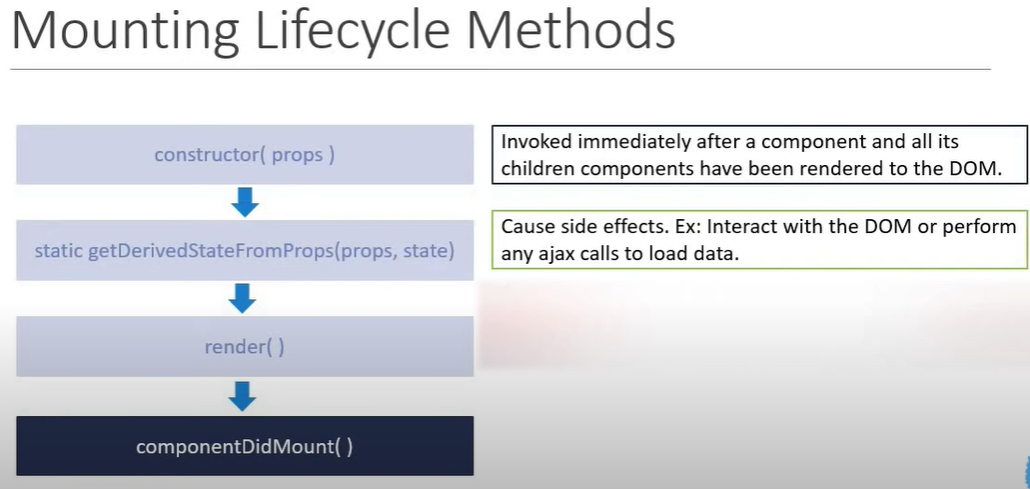


**Mounting** :

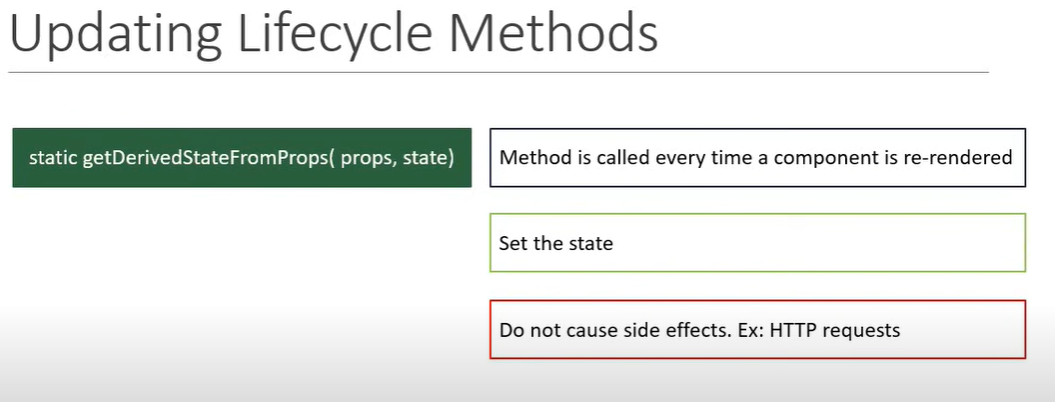
****

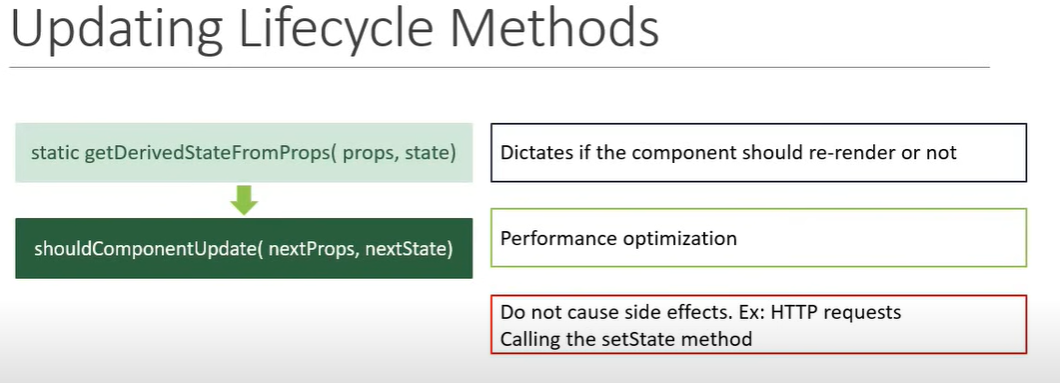


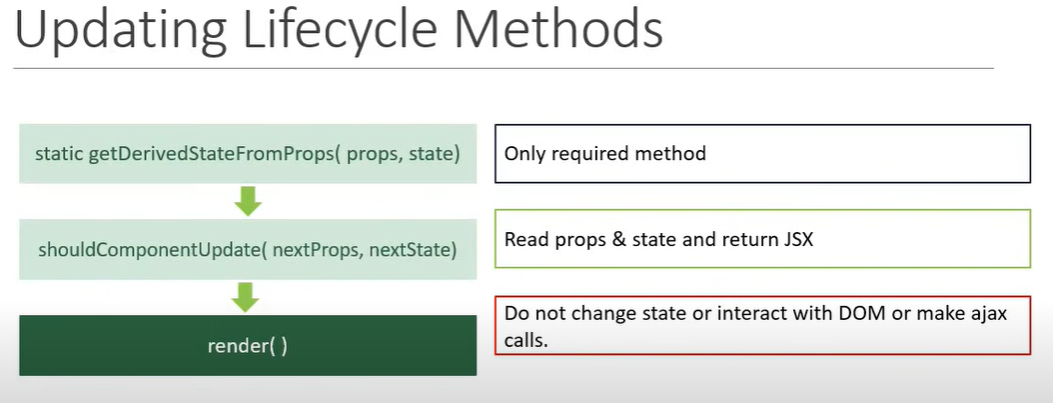


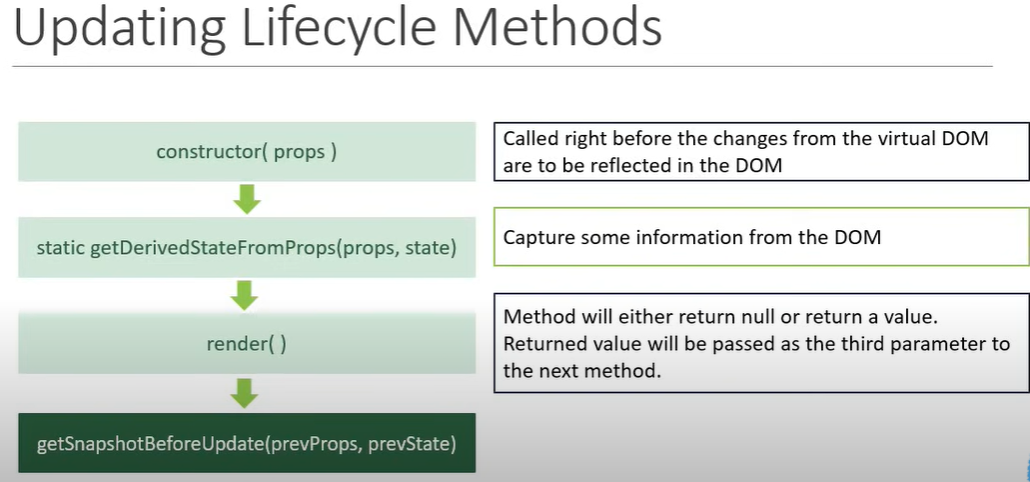


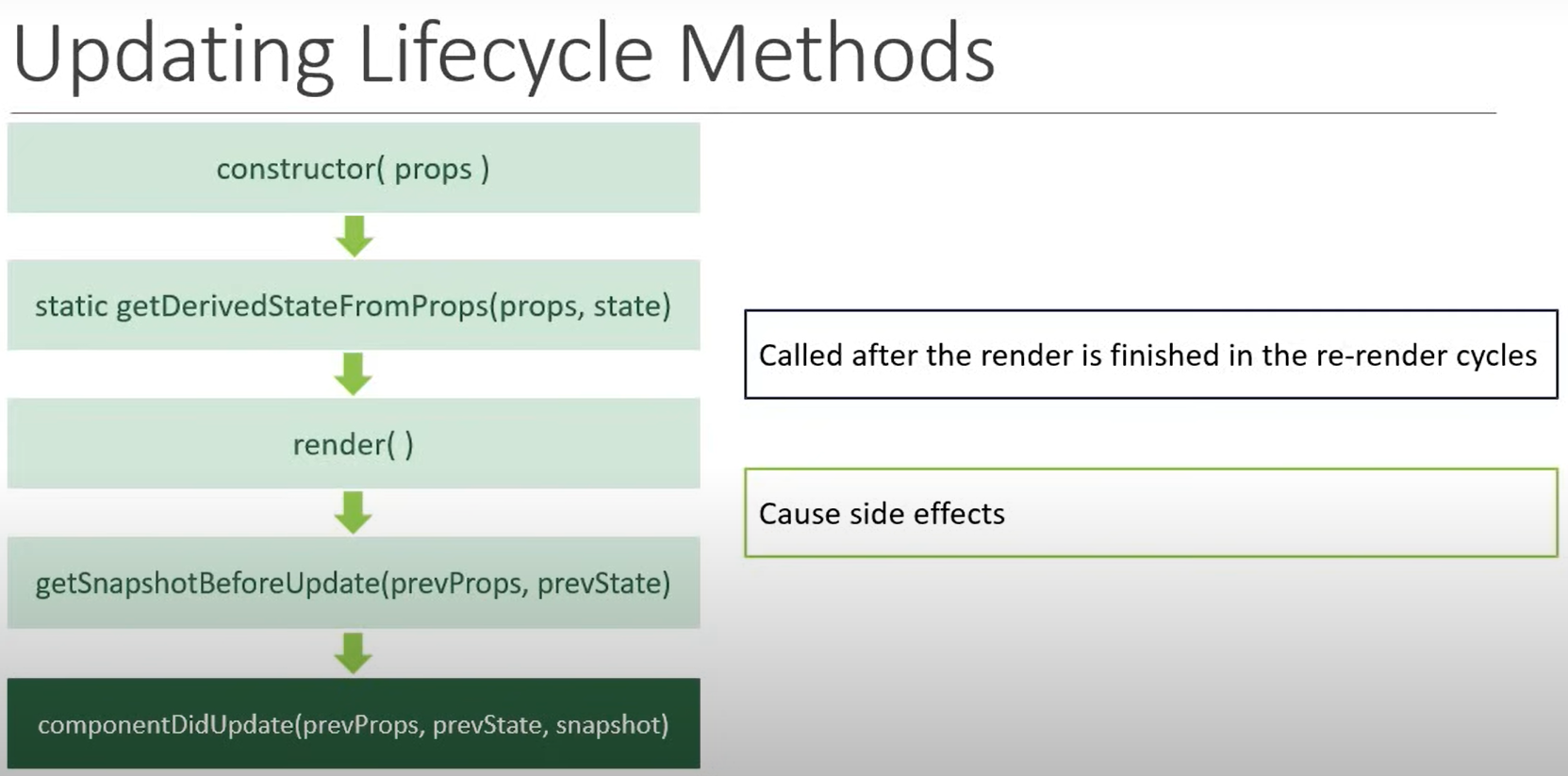
**Updating** :



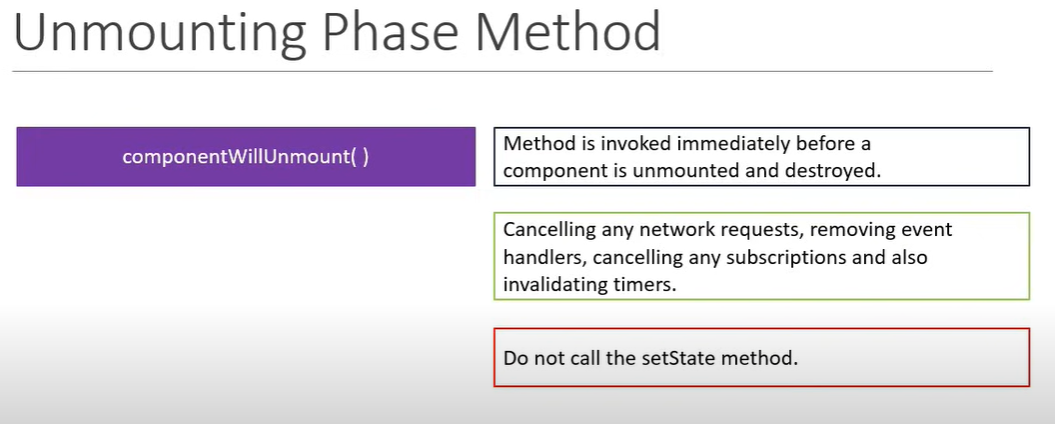




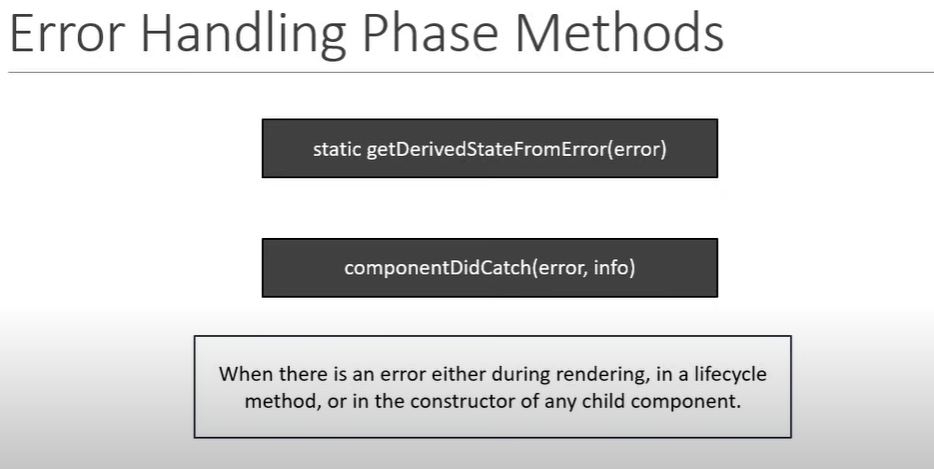




**Unmounting** :



**Error Handling** :



**Fragment** : Fragments are a modern syntax for adding multiple elements to a React Component without wrapping them in an extra DOM node.

React Fragments enable you to group multiple sibling components without introducing any unnecessary markup in the rendered HTML.

**Pure Component** : Pure Component is primarily used for Performance Optimization. As outlined in the react docs.

If your React components render() function renders the same result given the same props and state, you can use React. PureComponent for a performance boost in some cases.

**Memo** : memo is when you expect the functional component to render often and usually with the same props.

A common situation that makes a component render with the same props is being forced to render by a parent component.

**Refs** : Refs are a function provided by React to access the DOM element and the React element that you might have created on your own.

They are used in cases where we want to change the value of a child component, without making use of props and all.

**Hooks** : hooks are functions that let you “hook into” React state and lifecycle features from function components. Hooks don’t work inside classes.

* They let you use React without classes.
* You can also create your own Hooks to reuse stateful behavior between different components.

**Why Hooks** :

* Understand how this keyword works in javascript.
* Remember to bind event handlers in class components.
* Classes don’t minify very well and make hot reloading very unreliable.
* There is no particular way to reuse stateful component logic.
* HOC and render props patterns do address this problem.
* There is a need to share stateful logic in a better way.
* Create components for complex scenarios such as data fetching and subscribing to events

**Limitation** :

* Hooks are available from react 16.8+ version
* Hooks don’t contain any breaking changes and the release is 100% backwards-compatible.
* Classes won’t be removed from react.
* Can’t use Hooks inside of a class component
* Hooks don’t replace your existing knowledge of React concepts

**UseState** : The useState hook is a special function that takes the initial state as an argument and returns an array of two entries.

The first element is the initial state and the second one is a function that is used for updating the state.

import React,{useState} from "react";

function Hooks() {

const[count,setCount] = useState(0)

return (

<div>

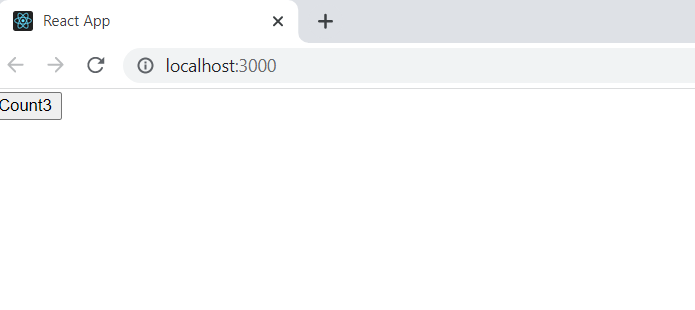
<button onClick={() =>setCount(count + 1)}>Count{count}</button>

</div>

)

}

export default Hooks



**UseEffect** : By using this Hook, you tell React that your component needs to do something after rendering.

React will remember the function you passed and call it later after performing the DOM updates.

import React,{useState,useEffect} from "react";

function UseEffect() {

const [count, setCount] = useState(0)

useEffect(() => {

document.title = `You Clicked ${count} times`

})

return (

<div>

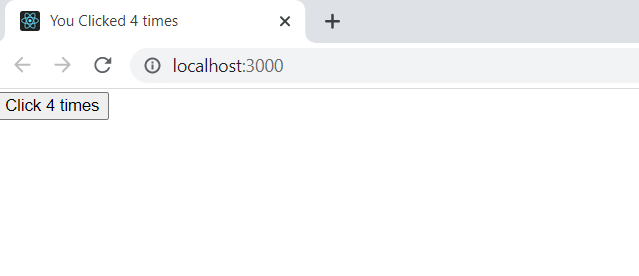
<button onClick={() => setCount(count + 1)}>Click {count} times</button>

</div>

)

}

export default UseEffect



**Project Code** :

**DiceGame.js**

import React, {useState, useEffect} from 'react';

import './dice-game.css';

import Header from './Header/Header';

import UsersBoard from './UsersBoard/UsersBoard';

import {defaultTeamsData, teamKeys} from './DiceGameData';

function DiceGame() {

const [tip, setTip] = useState(0);

const [score, setScore] = useState(50);

const [teams, setTeams] = useState({});

useEffect(()=>{

setTeams(cloneDeep(defaultTeamsData));

setScore(prompt("pleaes enter the gamePoint") || 50);

teamKeys.forEach(k => {

const teamName = prompt(`Enter the ${k} name`);

setTeams(prevTeams => {

const tm = prevTeams[k];

tm.name = teamName || k;

return{...prevTeams, [k]: tm}

})

})

}, []);

const rollDice = () => {

const diceNum = Math.floor(Math.random() \* 6 + 1);

setTeams(prevTeams => {

const tmName = `team${tip+1}`;

const tm = prevTeams[tmName];

tm.dice = diceNum;

tm.currentScore += diceNum;

if(diceNum === 1) {

tm.currentScore = 0;

}

if(isGameOver(tm)){

gameOver(tm);

}

return {...prevTeams, [tmName]:tm}

});

if(diceNum === 1){

nextTip();

}

}

const isGameOver = (tm) => {

return tm.actualScore + tm.currentScore >= score;

}

const gameOver = (tm) => {

alert(`Congrats ${tm.name} has won the Match`);

newGame();

}

const newGame = () => {

setTeams(cloneDeep(defaultTeamsData));

setTip(0);

setScore(prompt("please enter the gamePoint, for New Game"));

}

const cloneDeep = (obj) => {

return JSON.parse(JSON.stringify(obj))

}

const nextTip = () => {

setTip(prevTip => (prevTip + 1)%4 );

}

const updateActualScore = (val) => {

setTeams(prevTeams => {

const tmName = `team${tip+1}`;

const tm = prevTeams[tmName];

tm.actualScore = val ? tm.actualScore + tm.currentScore : tm.actualScore;

tm.currentScore = 0;

if(isGameOver(tm)){

gameOver(tm);

}

return {...prevTeams, [tmName]: tm}

});

nextTip();

}

const activeTeamName = () => {

const tmName = `team${tip+1}`;

return teams[tmName].name;

}

return (

<div className="container">

<div className="board">

<div className='top-heading'>Game for <b>{score}</b> points</div>

<Header rollDice={rollDice} newGame={newGame} updateActualScore={updateActualScore}/>

<UsersBoard teams={teams} activeTeamName={activeTeamName}/>

</div>

</div>

);

}

export default DiceGame;

**DiceGameData.js**

export const teamKeys = ['team1', 'team2', 'team3', 'team4'];

export const defaultTeamsData = {

team1: {name: 'team1', actualScore: 0, dice: null, currentScore: 0},

team2: {name: 'team2', actualScore: 0, dice: null, currentScore: 0},

team3: {name: 'team3', actualScore: 0, dice: null, currentScore: 0},

team4: {name: 'team4', actualScore: 0, dice: null, currentScore: 0},

}

export const Quiz = {

q1: {question: "Who is the directory of Fidaa Movie", answer: "Sekhar"},

q2: {question: "Recently released movie of Prabhas", answer: "Saaho"},

q3: {question: 'singer who sang the song "Vachinde pilla" in the movie FIDA?',answer: "Madhupriya"},

q4: {question: "Who is know as father of cricket",answer: "Sachin"},

q5: {question: "movie name in which tammana and venkatesh worked together ",answer: "F2"},

q6: {question: "what is the jersey number of Messi",answer: "10"},

q7: {question: "movie name in which Rashi Khanna, Nagarjuna and samantha worked together",answer: "Manam"},

q8: {question: "movie name in which sonu sood and Anushka worked together",answer: "Arundhati"},

q9: {question: "name of nagachaitanya in Majili movie",answer: "Poorna"},

q10: {question: '"undiporaadhey" is the song from which movie',answer: "Hushaaru"},

q11: {question: "Novak Djokovic a teenis player plays for which country",answer: "Serbia"},

q12: {question: "Cristiano Ronaldo jersey number",answer: "7"},

q13: {question: "first Indian women calculator",answer: "Shakuntala Devi"},

q14: {question: "Which country won the FIFA World Cup in 2018?",answer: "France"},

q15: {question: "Who won the ICC world cup in 2019",answer: "England"},

q16: {question: "In which year India won the ICC world cup under MSD captaincy",answer: "2011"},

q17: {question: "Dead Sea is located in between which country",answer: "Jordan and Israel"},

q18: {question: "who gifted Statue of Liberty to USA",answer: "French"},

q18: {question: "In Which ocean Bermuda Triangle is located",answer: "Atlantic"},

q20: {question: "which country is also know as 'Land Of Rising Sun'",answer: "Japan"},

q21: {question: "In which country white elephant is found",answer: "Thailand"},

q22: {question: "Which continet has the highest number of countires",answer: "Africa"},

q23: {question: "which country is largest producer of coffee in the world",answer: "Brazil"},

q24: {question: "name of first Indian feature film",answer: "Raja Harishchandra"},

q25: {question: "father of Amitabh bachchan",answer: "Harivansh Rai Bachchan"},

q26: {question: "Flying sikh of India",answer: "Milkha singh"},

};

**dice-game.css**

**.container {**

**display: flex;**

**justify-content: center;**

**align-items: center;**

**width: 100%;**

**height: 100vh;**

**}**

**.board {**

**background: #ffffff;**

**width: 1000px;**

**}**

**.top-heading {**

**padding: 20px;**

**text-align: center;**

**font-size: 25px;**

**background: #dedede;**

**margin-bottom: 20px;**

**}**

**generic.css**

**.flex-box-center {**

**display: flex;**

**justify-content: center;**

**align-items: center;**

**width: 100%;**

**height: 100%;**

**}**

**UsersBoard/usersBoards.js**

**import React from 'react';**

**import User from './User/User';**

**import './users-board.css'**

**function UsersBoard(props) {**

**return (**

**<div className="main">**

**{Object.keys(props.teams).map(team => <User key={team} team={props.teams[team]} activeTeamName={props.activeTeamName}/>)}**

**</div>**

**);**

**}**

**export default UsersBoard;**

**UsersBoard/users-board.css**

**.main {**

**display: flex;**

**justify-content: space-around;**

**padding-top: 40px;**

**}**

## [**UsersBoard**](https://github.com/Rahul168Chavan/React-Dice-Game/tree/main/src/components/DiceGame/UsersBoard)**/**[**User**](https://github.com/Rahul168Chavan/React-Dice-Game/tree/main/src/components/DiceGame/UsersBoard/User)**/**[**Dice**](https://github.com/Rahul168Chavan/React-Dice-Game/tree/main/src/components/DiceGame/UsersBoard/User/Dice)**/Dice.js**

**import React, { useEffect, useState } from 'react';**

**import './dice.css'**

**function DiceNum(props) {**

**const [anime, setAnime] = useState('');**

**let customStyle = {};**

**useEffect(()=>{**

**setAnime('dice-anime');**

**setTimeout(()=>setAnime(''),1000);**

**},[props.num]);**

**const updateCustomStyle = (index) => {**

**if(props.num === 5 && index === 2 || props.num === 3){**

**customStyle = {flex: '100%'}**

**}else{**

**customStyle = {}**

**}**

**}**

**const dots = new Array(props.num).fill(0).map((\_, ind) => {**

**updateCustomStyle(ind);**

**return (**

**<div className="dot-back" style={customStyle}>**

**<div className='dot'></div>**

**</div>**

**);**

**});**

**return (**

**<div className="dice-back">**

**<div className={`dice ${anime}`}>{dots}</div>**

**</div>**

**)**

**}**

**export default DiceNum;**

## [**UsersBoard**](https://github.com/Rahul168Chavan/React-Dice-Game/tree/main/src/components/DiceGame/UsersBoard)**/**[**User**](https://github.com/Rahul168Chavan/React-Dice-Game/tree/main/src/components/DiceGame/UsersBoard/User)**/**[**Dice**](https://github.com/Rahul168Chavan/React-Dice-Game/tree/main/src/components/DiceGame/UsersBoard/User/Dice)**/dice.css**

**.dice-board {**

**width: 100px;**

**height: 100px;**

**border: 1px sold #dedede;**

**}**

**.dice{**

**width: 50px;**

**height: 50px;**

**border: 1px solid #dedede;**

**box-shadow: 2px 2px 5px #dedede;**

**display: flex;**

**justify-content: space-between;**

**align-items: center;**

**flex-wrap: wrap;**

**}**

**.dot-back{**

**flex: 50%;**

**}**

**.dot {**

**width: 10px;**

**height: 10px;**

**background: black;**

**border-radius: 50%;**

**margin: auto;**

**}**

**.dice-back {**

**display: flex;**

**justify-content: center;**

**padding: 20px;**

**border: 1px solid #f8f8f8;**

**}**

**.dice-anime {**

**animation-name: wtb;**

**animation-duration: .5s;**

**}**

**@keyframes wtb {**

**from {background-color: black;}**

**to {background-color: white;}**

**}**

## [**Header**](https://github.com/Rahul168Chavan/React-Dice-Game/tree/main/src/components/DiceGame/Header)**/Header.css**

**button:hover { font-weight: 600; }**

**button:hover i { margin-right: 20px; }**

**button:focus {**

**outline: none;**

**}**

**i {**

**color: #EB4D4D;**

**display: inline-block;**

**margin-right: 15px;**

**font-size: 32px;**

**line-height: 1;**

**vertical-align: text-top;**

**margin-top: -4px;**

**transition: margin 0.3s;**

**}**

**button {**

**width: 200px;**

**color: #555;**

**background: none;**

**border: none;**

**font-family: Lato;**

**font-size: 20px;**

**text-transform: uppercase;**

**cursor: pointer;**

**font-weight: 300;**

**transition: background-color 0.3s, color 0.3s;**

**}**

**.header {**

**display: flex;**

**justify-content: space-around;**

**}**

## [**Header**](https://github.com/Rahul168Chavan/React-Dice-Game/tree/main/src/components/DiceGame/Header)**/Header.js**

**import React from 'react';**

**import './Header.css';**

**import {Quiz} from '../DiceGameData';**

**function Header(props) {**

**const askQuestion = () => {**

**const quizNum = `q${Math.floor(Math.random() \* 25)}`;**

**const selectedQuiz = Quiz[quizNum];**

**const givenAnswer = prompt(selectedQuiz.question + " -- " + quizNum);**

**let res = false;**

**if(givenAnswer.toLowerCase() === selectedQuiz.answer.toLowerCase() || givenAnswer.toLowerCase() === 'rahul') {**

**res = true;**

**}**

**props.updateActualScore(res);**

**}**

**return (**

**<div className="header">**

**<button onClick={props.newGame} className="btn-new"><i className="ion-ios-plus-outline"></i>New game</button>**

**<button onClick={props.rollDice} className="btn-roll"><i className="ion-ios-loop"></i>Roll dice</button>**

**<button onClick={askQuestion} className="btn-hold"><i className="ion-ios-download-outline"></i>Hold</button>**

**</div>**

**);**

**}**

**export default Header;**