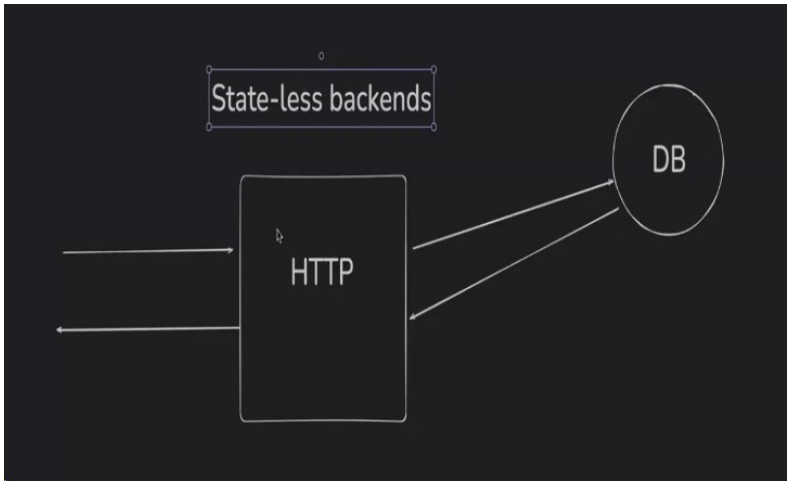
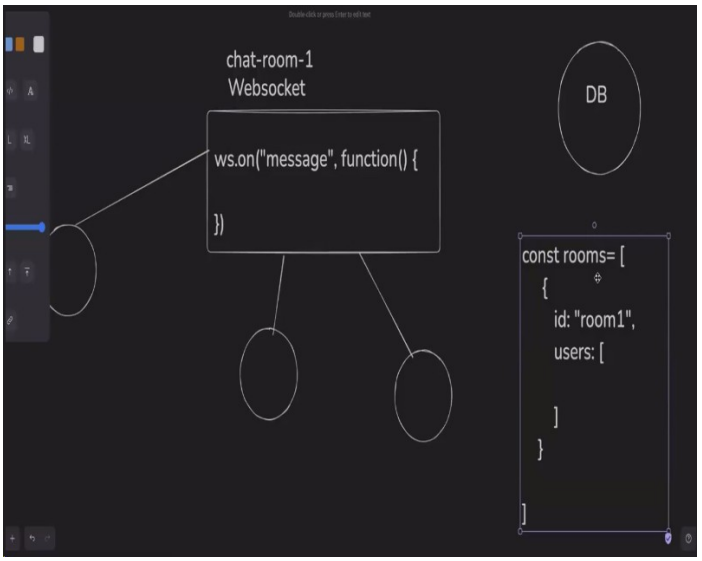
STATE MANAGEMENT
what is state less backend??
in this you does not store anything on the http not a single variable you get everything from the db and use it and give to the user



BUT FOR A WS WE NEED TO HAVE THE STATE

LIKE IN THE WS FOR EVERY USER WE HAVE TO MAINTAIN A VARIABLE WHICH HAVE SOME DATA RELATED TO THE USER AND RELATED TO THE CHAT WHICH HAS TO BE SENT TO THEM



NOW THE THINGS LIKE A USER WANT TO CONNECT TO THIS ROOM THESE THINGS SHOULD NOT BE STORED IN THE DB FOR THESE THINGS WE CAN EITHER USE LIKE REDUX, OR A FILE TO STORE ALL THIS IN A VARIABLE OR SOMETHING CALLED AS SINGLETON

REDUX CAN DO STATE MANAGEMENT IN BOTH FE AND BE(MILDLY COMPLICATED BUT STABLE)
SIMPLEST APPROACH TO STORE STATE IN A BE IS TO STORE IN A GLOBAL VARIABLE

we can only send string and binary using the ws

to test the ws we can use this
ws://localhost:8080?token=token_from_the_auth
if we use a false token it crashes the be so for that put the code in try catch

NOW THE CHAT SERVER IS MADE JUST IN 70 LINES OF CODE BUT SOME FLAWS ARE THERE
NOW DUMB APPROACH TO DIRECTLY STORE THE CHATS IN DB

LIKE IN THIS WE ARE STORING THE MSG FIRST IN DB AND THEN BROADCASTING TO THE USER FIRST PRINCIPLE APPROACH
BETTER UNDERSTANDING IN THE CHESS app
we can use queue then using pipelining we should do

HOW TO CENTRALIZE A DIV??

WHAT DOES CLIENT COMPONENT PAGE ERROR MEANS?
Whenever you are using any buttons or usestate we have to use use client at the top as this cannot be done in a server component

explore shadcn ui

we can use fabric js(it has all drawing logic)
d3
If we learn the canvas thing we can build the stake thingy, flappy bird also

like what we are doing is capturing the mouse movement and also like in this this is a 30 fps thing which means in this we have to first clear the canvas and then rebuild that again to make it work

```tsx
useEffect(() => {

    if (canvasRef.current) {
        const canvas = canvasRef.current;
        const ctx = canvas.getContext("2d");

        if (!ctx) {
            return
        }

        let clicked = false;
        let startX = 0;
        let startY = 0;

        canvas.addEventListener("mousedown", (e) => {
            clicked = true
            startX = e.clientX
            startY = e.clientY
        })

        canvas.addEventListener("mouseup", (e) => {
            clicked = false
            console.log(e.clientX)
            console.log(e.clientY)
        })
```

```tsx
        canvas.addEventListener("mousemove", (e) => {
            if (clicked) {
                const width = e.clientX - startX;
                const height = e.clientY - startY;
                ctx.clearRect(0, 0, canvas.width, canvas.height);
                ctx.fillStyle = "rgba(0, 0, 0)"
                ctx.fillRect(0, 0, canvas.width, canvas.height);
                ctx.strokeStyle = "rgba(255, 255, 255)"
                ctx.strokeRect(startX, startY, width, height);
            }
        })
    }

}, [canvasRef]);

return <div>
    <canvas ref={canvasRef} width={1080} height={1000}></canvas>
</div>
}
```

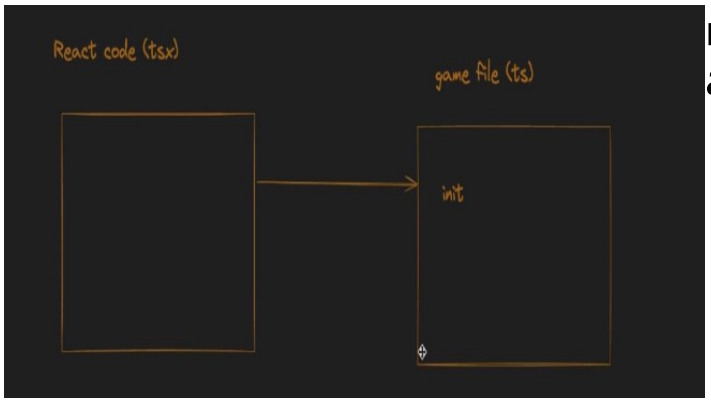
this is the easier code and in the codebase we have moved this code to draw index.ts and imported it

this is what changes when we are creating a pool table game, roullete game, fps games,

sometime to make the component a server one we have to send the data to another file and in that use the use client and then import that in the server client tsx file and use it and that page will be rendered on server side only

the clearcanvas is very similar to the react re render loop like whenever something changes it loads the thing again same happens in the clear canvas

mouse up means you are leaving the mouse not like moving the mouse up literally I was thinking that



now we have to share a variable among these

to access a variable from one page to another you can use the window object (canvas.tsx)

```tsx
const canvasRef = useRef<HTMLCanvasElement>(null);
const [selectedTool, setSelectedTool] = useState<Shape>("circle");
useEffect(() =>{
    //@ts-ignore
    window.selectedTool = selectedTool;
},[selectedTool]);

useEffect(()=>{
    if(canvasRef.current){
        initDraw(canvasRef.current,roomId,socket);
    }
},[canvasRef])
```

then used in draw.tsx

```tsx
ctx.strokeStyle = "rgba(255,255,255)"
//@ts-ignore
const selectedTool = window.selectedTool;
if(selectedTool === "rect"){
    ctx.strokeRect(startX, startY,width, height);
}else if(selectedTool === "circle"){

}
}
```

```tsx
}){

    const canvasRef = useRef<HTMLCanvasElement>(null);
    const [selectedTool, setSelectedTool] = useState<Shape>("circle");
    useEffect(() =>{
        //@ts-ignore
        window.selectedTool = selectedTool;
    },[selectedTool]);

    useEffect(()=>{
        if(canvasRef.current){
            initDraw(canvasRef.current,roomId,socket);
        }
    },[canvasRef])
    return <div style={{
        height:"100vh",
        overflow: "hidden"
    }}>
```

changed this according to the new game.tsx its just a way of structing code writing it in constructor and all

Always follow the zod schema design for the input in the postman

```
axios.post( URL ,  DATA ,  SETTINGS )
             1      2         3

- Box 2 (DATA) = what you're sending → { name: "..." }
- Box 3 (SETTINGS) = the headers (your token) → { headers: { Authorization: token } }

The token must go in Box 3, all by itself. Right now it's stuffed inside Box 2, so the server never sees it.

axios.post(
   url,
   { name: roomRef.current?.value },          ← Box 2
   { headers: { Authorization: token } }      ← Box 3
)

---
Now — to actually help, what is the issue you're still seeing? Tell me:
1. What error shows up (red text in browser console or terminal)?
2. Or does it just not create the room?

Paste the error message and I'll point to the exact cause. 🙂
```