Openboard

- → HTML, CSS → JS (DOM)
- → Canvas API ~ draw
- → Files API
- → server → Express
- → Real time communication → socket.io

Pencil draw

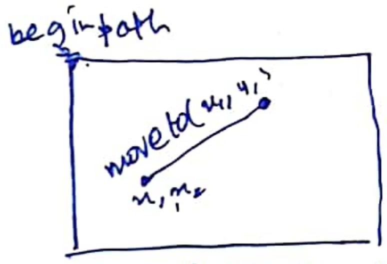
window → represents the browser page (html) → size fixed
document → represents the tab → size can vary

let ctx = canvas.getContext('2d')

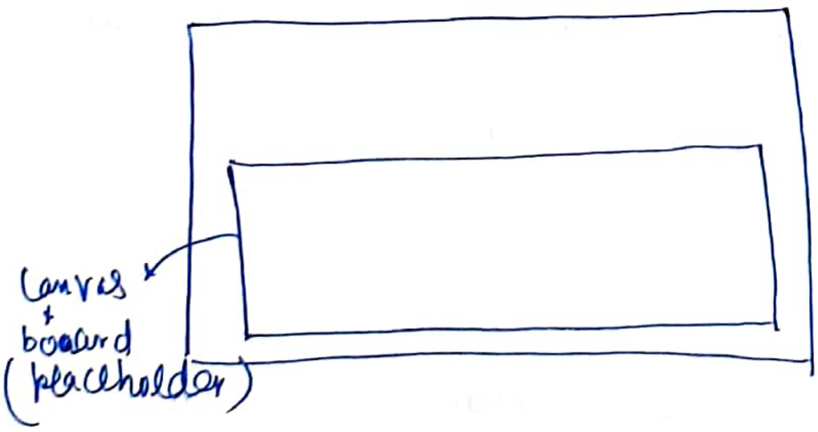
The ctx variable contains a CanvasRenderingContext2D object, and all drawing operations on the canvas will involve manipulating this object.

To draw

1. begin path → draw start (new path is initialised)
2. move to → move to a particular point without drawing anything.
3. line to → draws line between current point and previous point.



stroke → (render the line on the screen) → draw perimeter/outline
fill → render the whole area
default → fill, stroke (black color)
width → 1px



How to draw → click drag and drop.

→ (mouse down)
→ mouse move
→ mouse up
  ↳ release pointer element
→ mouseto
→ mouse leave (if goes outside the element)

] → events

~~we~~ I will be applying immediate line to calls while moving mouse

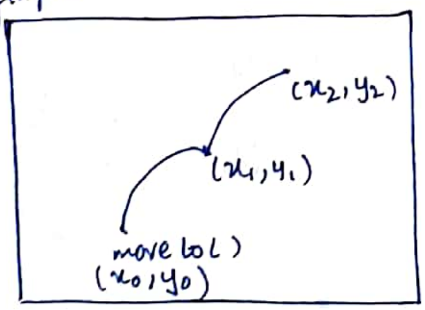
pencil → stroke style (color)
erasor → stroke style = (white)

getboard Bouldclient-
↓
give the x, y coordinate of the canvas from top left (0, 0)

handleToolChange
↳ if already active class show options
↳ if another tool is removed from select remove options

---

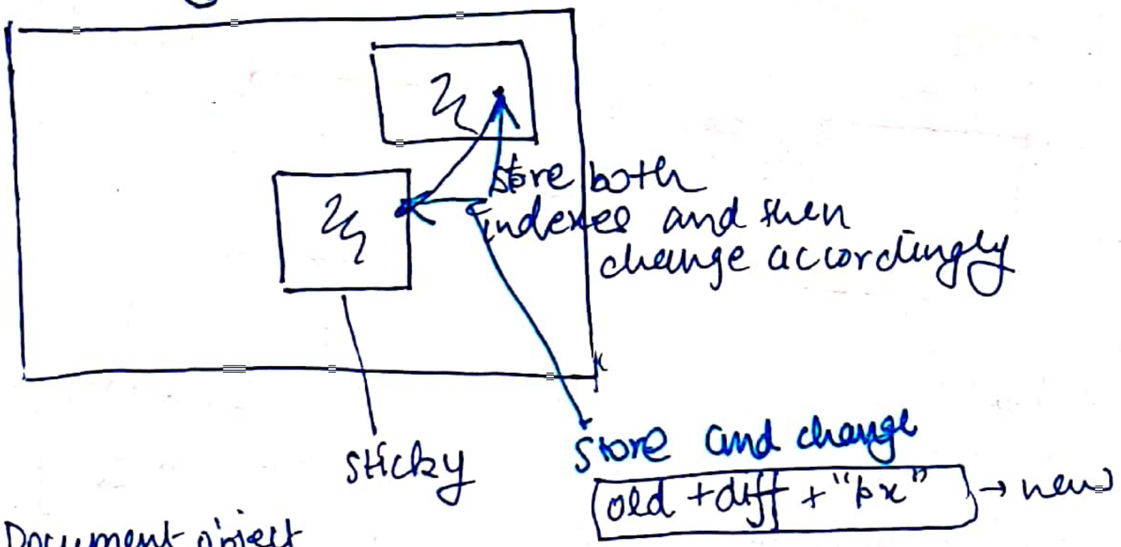
begin path ()



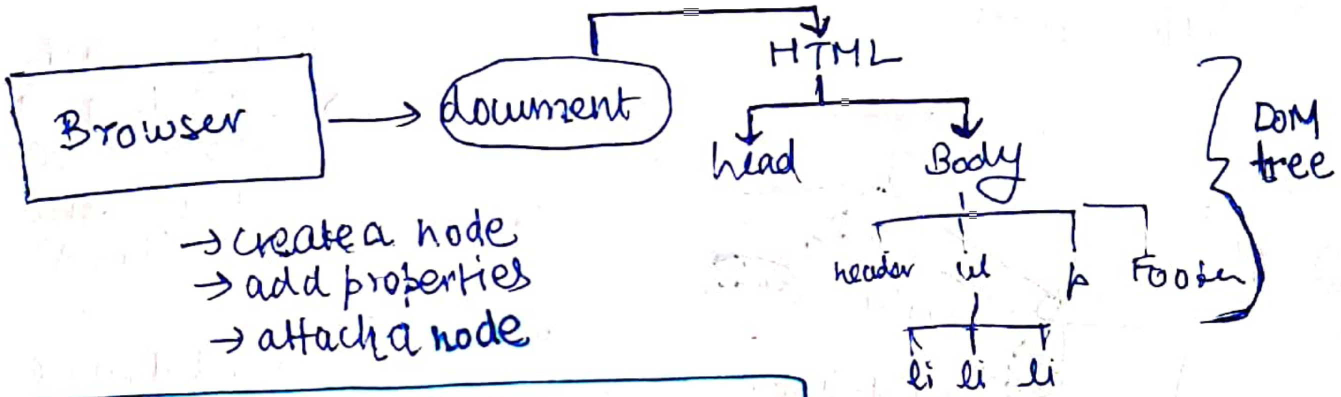
to redo undo store in an array · [x, y, type : "md", color]
(2) Pop last point
(3) clear canvas
(4) Draw for left over points.

set interval → repeats work after a certain interval

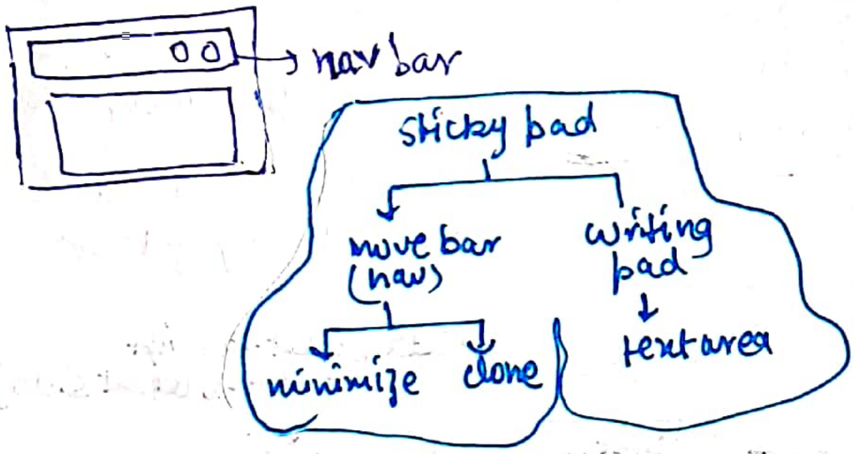
To move sticky pad



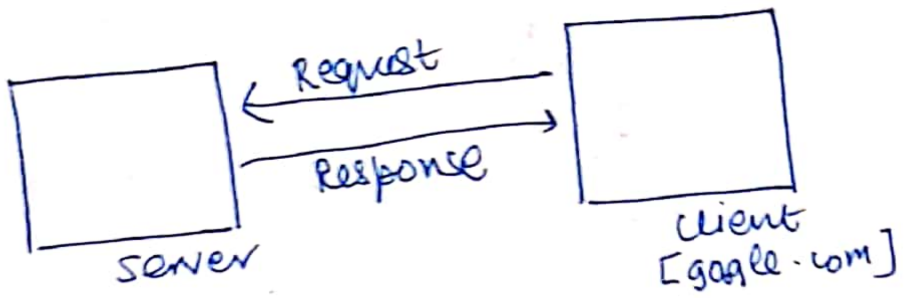
DOM - Document object model



→ create a node
→ add properties
→ attach a node



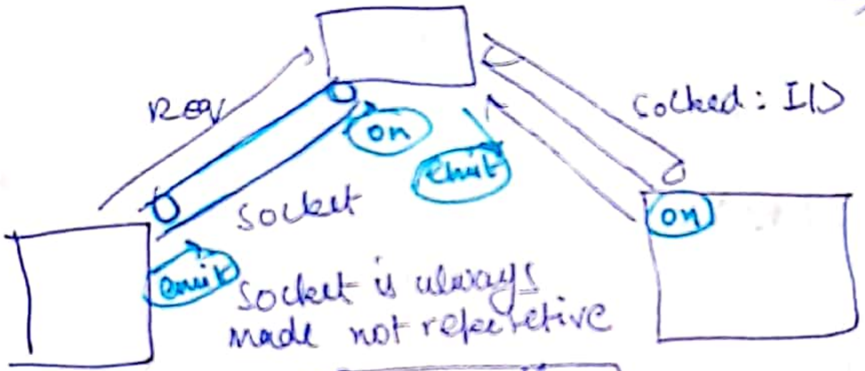
To upload use
sam div / querys
used using
DOM queries.



IP ┌→ DNS (domain name server)
└ Has IP of google.com

https → stateless
(means repetedly goes to DNS to recieve IP)
better for communication
use sockets

→ Request sent
connection closed
response recieved
is a seperate thing



socket is always made not repetetive

size → small
are not stateless

→ Sockets once connection is made not closed
(pathway is created)

Don't have to make connection / get IP

web sockets → event based
→ server should be socket enabled
→ client side (socket.io)

→ add socket.io script to html (client side)

To send msg · socket.emit() emit()
receiving socket.listen() on()

socket.io
→ ws
→ long Polling

To make realtime
→ write logic for send
→ write for recieve

PORT: process.env.PORT
git init -y
★ touch .gitignore

socket.io cheetsheet
→ send data globally to clients

package.json and .gitignore should be outside while deploying
add node modules to gitignore
→ npm i express socket.io

If image is zoomed by 120%
How to bring to 100%

20 is $1/6$th of 120%.

So will scale down 0.15

$1 - 0.15 = \boxed{0.85}$