backgroundColor

id="myCanvas"

400 || 600 || 800 width

400 || 600 || 800 width

id="blue" | id=white | id=grey | id=green } button

editable

⊙ id="edit" ⊙ id="x" } radiobuttons / gleiche Radiogruppe

canvas

⊙ id="small" ⊙ id="medium" ⊙ id="large" } radiobuttons / gleiche Radiogruppe

Circle

placeCircle
id="circle"

} button → platziert Kreise auf Canvas

Color Change

⊙ id="purple" ⊙ id="pink" ⊙ id="blue" } radiobuttons / gleiche Radiogruppe

Rectangle

placeRectangle
id="rect"

} button → platziert Vierecke auf Canvas

Color Change

⊙ id="purple1" ⊙ id="pink1" ⊙ id="blue1" } radiobuttons / gleiche Radiogruppe

animation

} in dieses Div werden die buttons von der function "createAnimationButtons" geschrieben

save and load

id="save" } Button

saved pictures

} buttons

Anwendungsfalldiagramm

Anwender kann Kreise/Formen platzieren

Formen können bewegt werden (dragging)

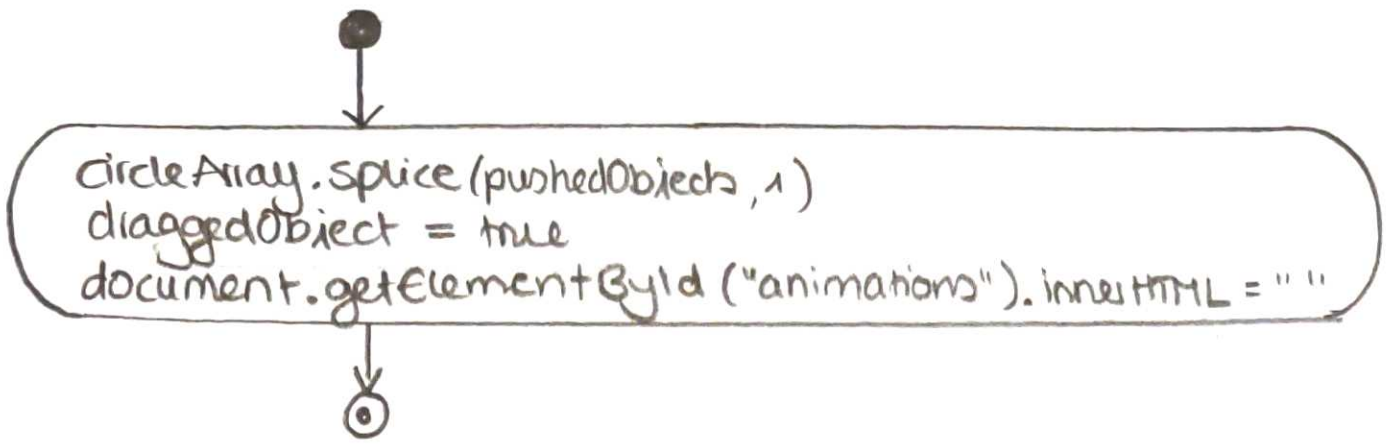
Formen können animiert werden (farbdurchlauf oder Bewegung nach rechts)

Farbe des Hintergrunds und der Kreise bzw. Rechtecke kann durch Farbauswahl angepasst werden

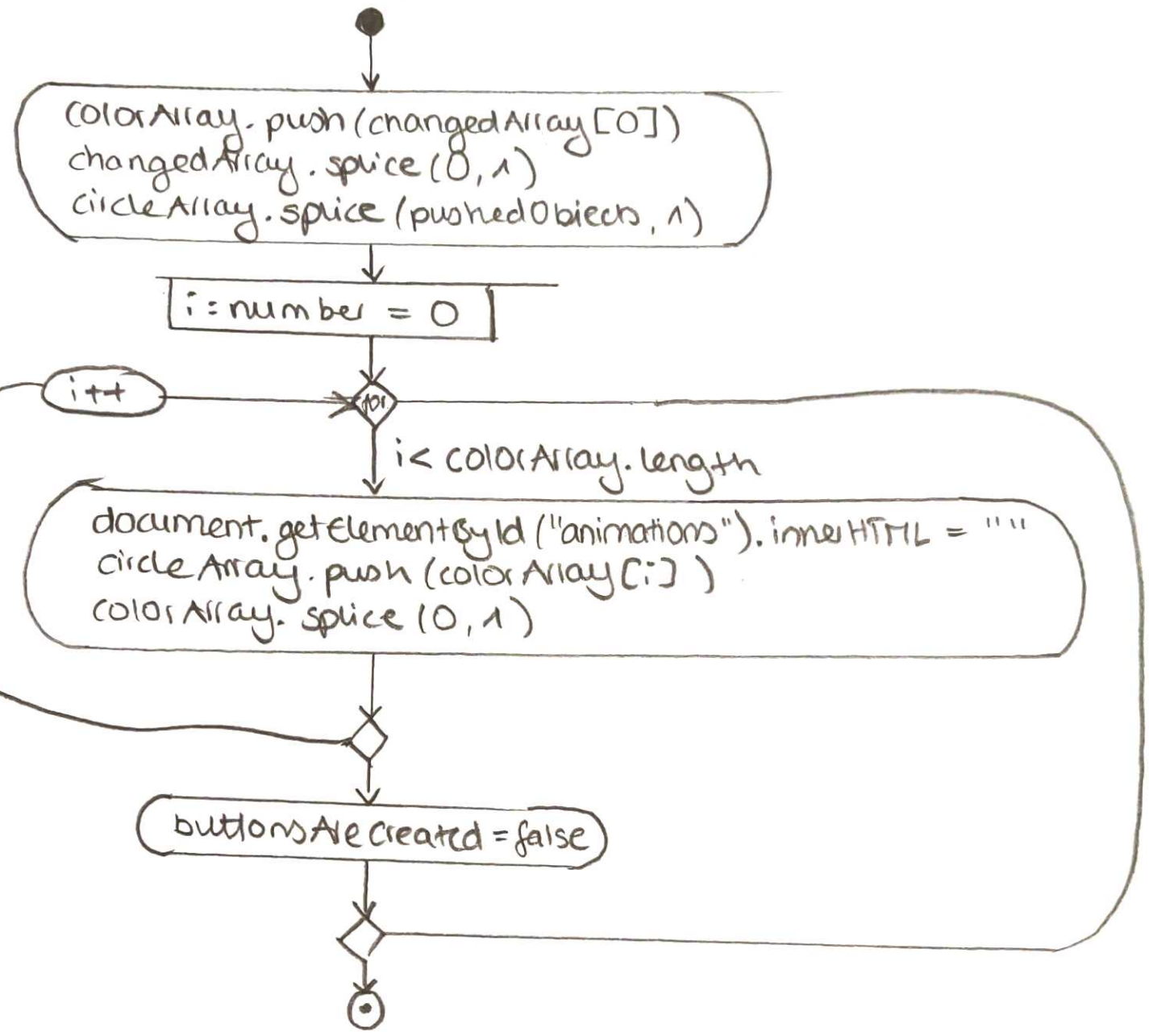
man kann die Größe der Malfläche einstellen

man kann Bilder abspeichern und sie zu einem späteren Zeitpunkt weiterbearbeiten

dragObiects



colorObiects

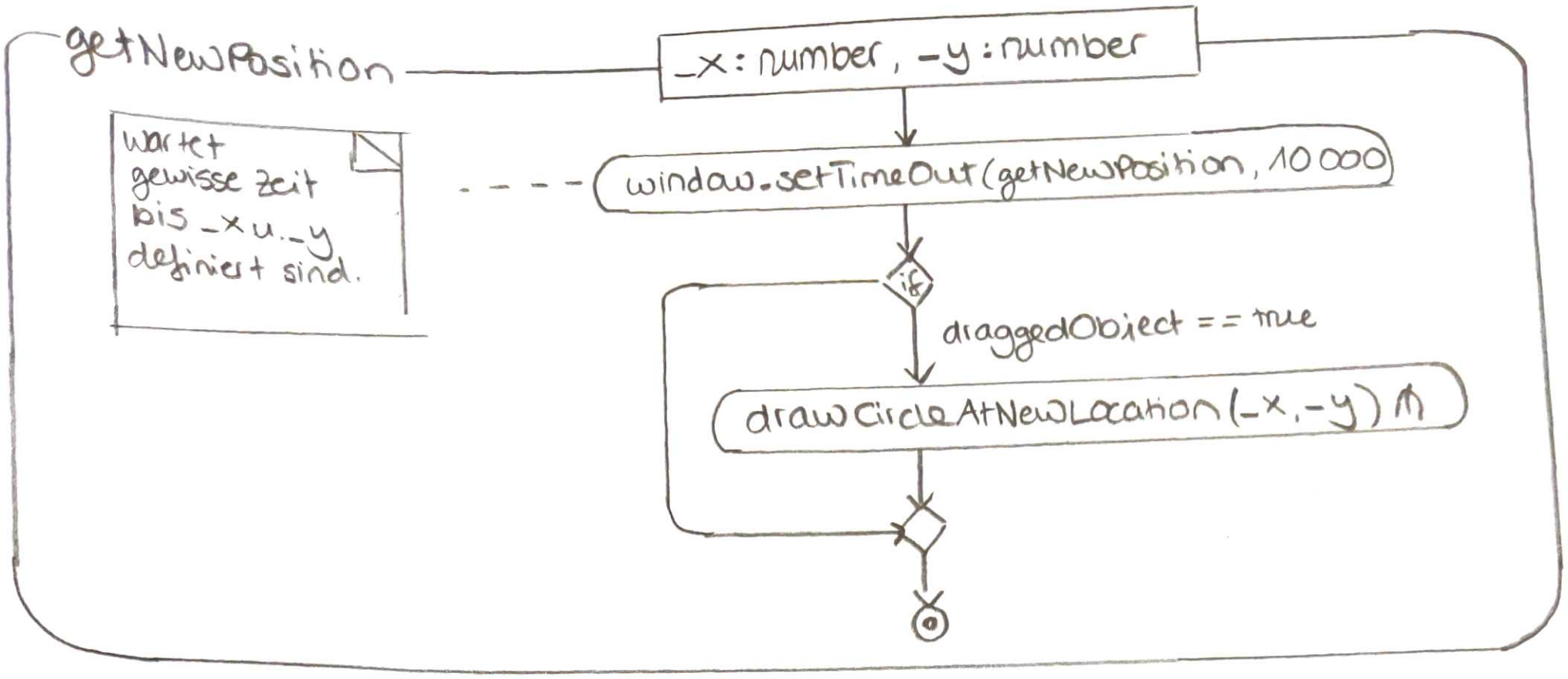


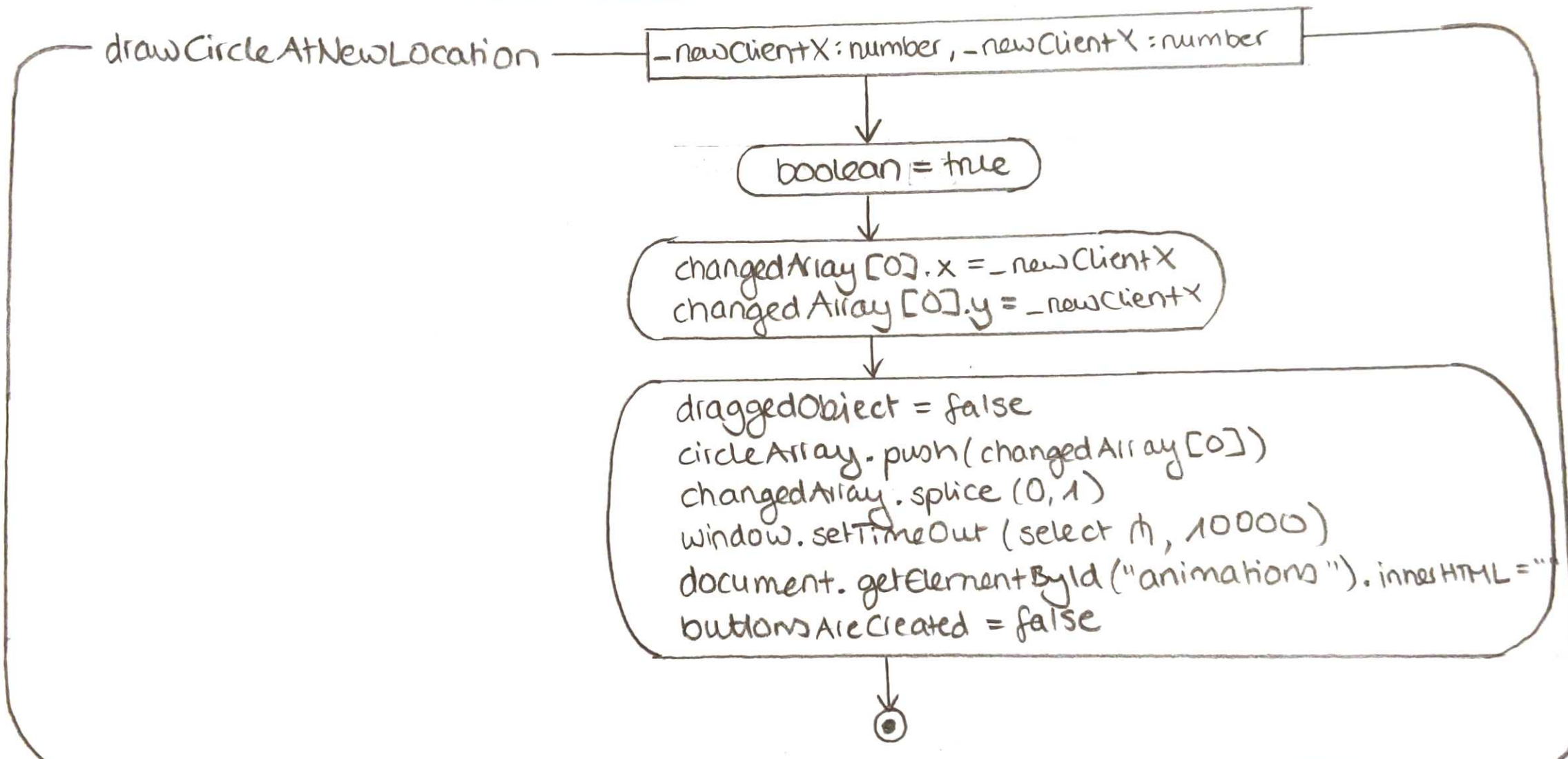
save

picture_1

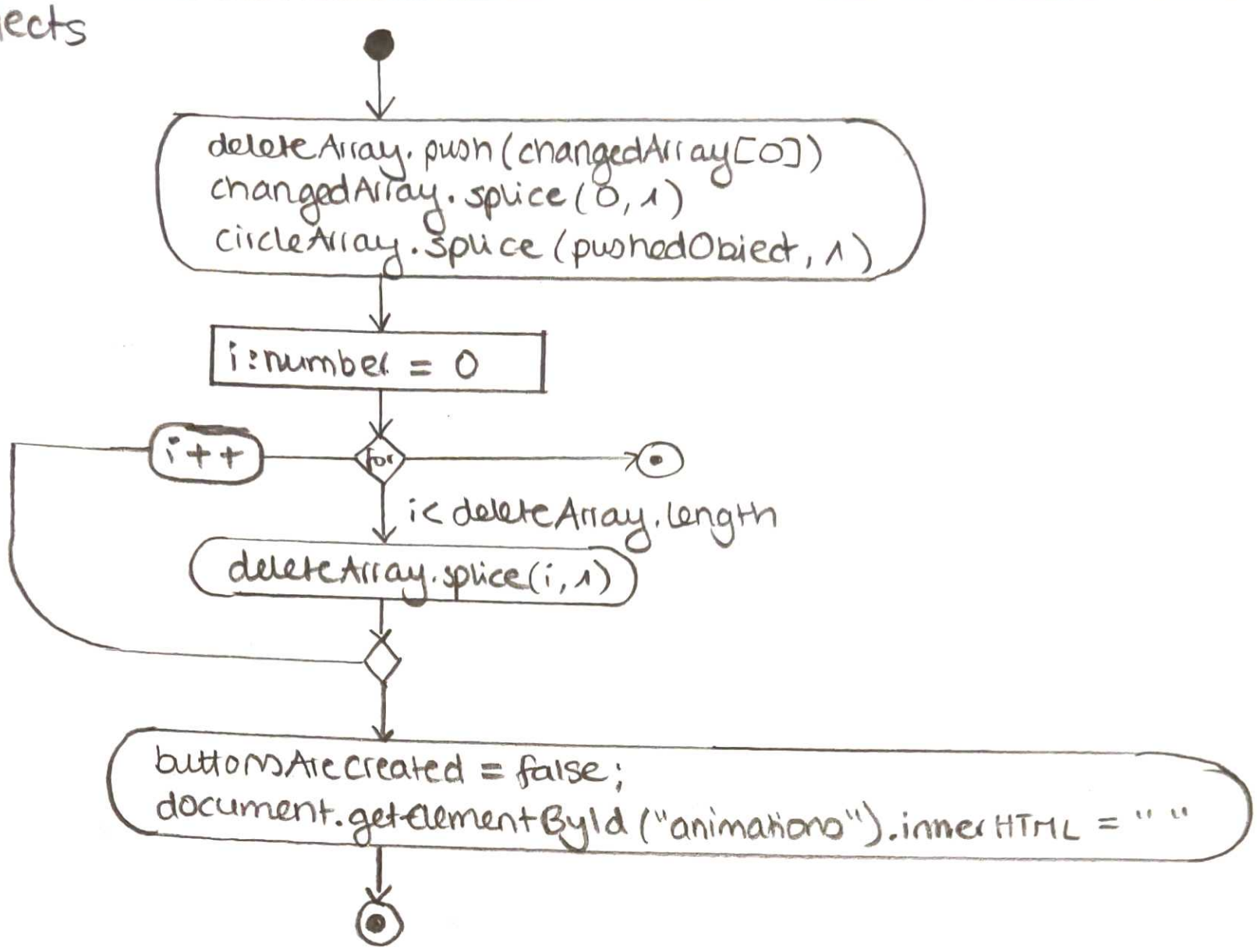
picture_2

picture_3

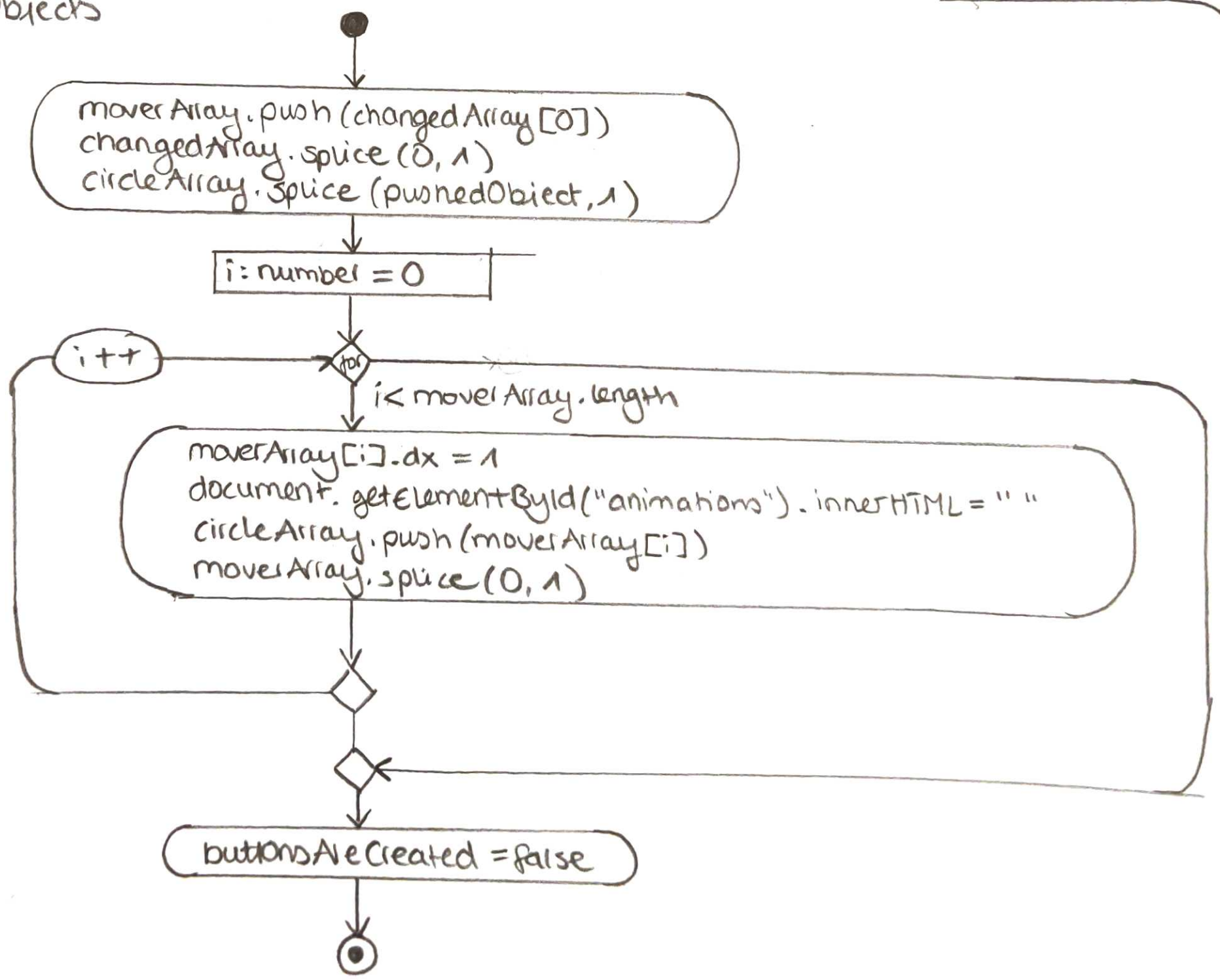

getNewPosition
_x: number, -y: number
wartet gewisse Zeit bis _x u. _y definiert sind.
window.setTimeOut(getNewPosition, 10 000
if
draggedObject == true
drawCircleAtNewLocation(_x, -y)


drawCircleAtNewLocation
_newClientX: number, _newClientY: number
boolean = true
changedArray[0].x = _newClientX
changedArray[0].y = _newClientX
draggedObject = false
circleArray.push(changedArray[0])
changedArray.splice(0,1)
window.setTimeOut(select ↑, 10000)
document.getElementById("animations").innerHTML = ""
buttonsAreCreated = false

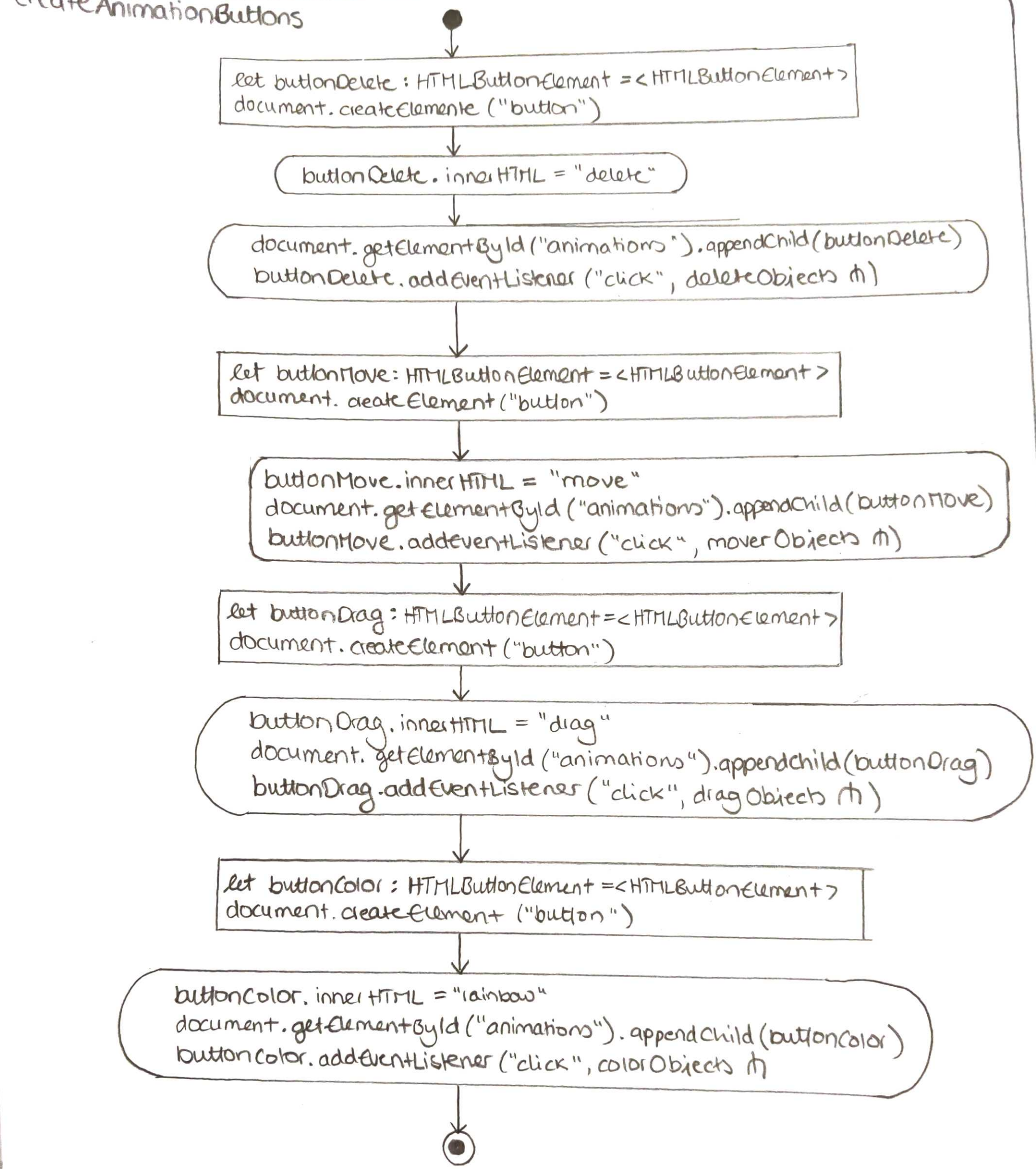
deleteObjects



moverObjects



createAnimationButtons



select

_e : MouseEvent

let clientX : number = _e.clientX
let clientY : number = _e.clientY

getnewPosition (clientX, clientY)

entweder hier edit aufrufen/definieren
oder global angelegt haben (per ID)
let edit : HTMLInputElement = <HTMLInputElement>
document.getElementById ("edit")

checkt ob radiobutton "edit" selektiert ist um zu bearbeiten

if

edit.checked = true

let i : number = 0

i++

for

i < circleArray.length

let currentX : number = circleArray[i].x
let currentY : number = circleArray[i].y

wenn nicht übereinstimmt

if

clientX < currentX + 20

4 solcher Argumente um Radius festzulegen

boolean = true

if

boolean == false

changedArray.push (circleArray[i])
pushedObject = i

= true

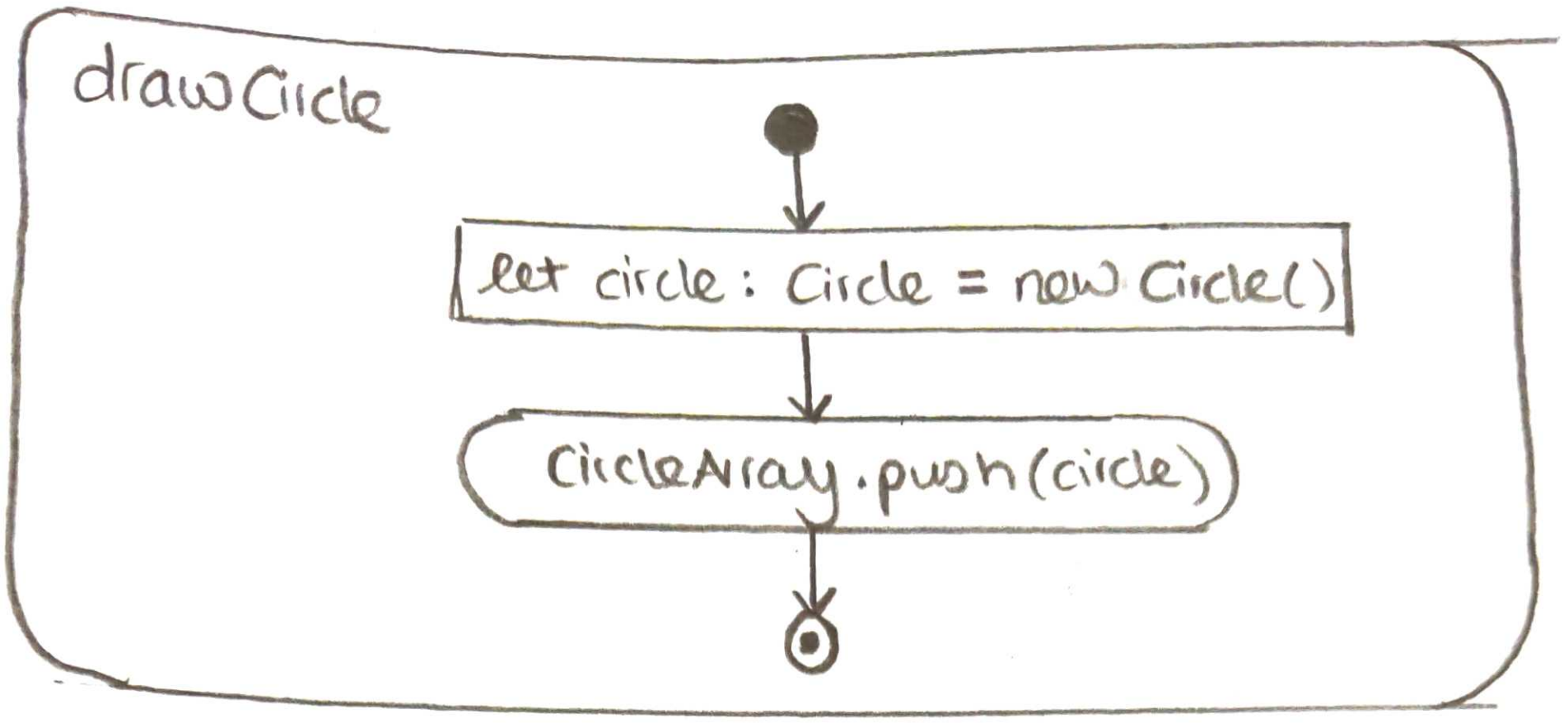
IF

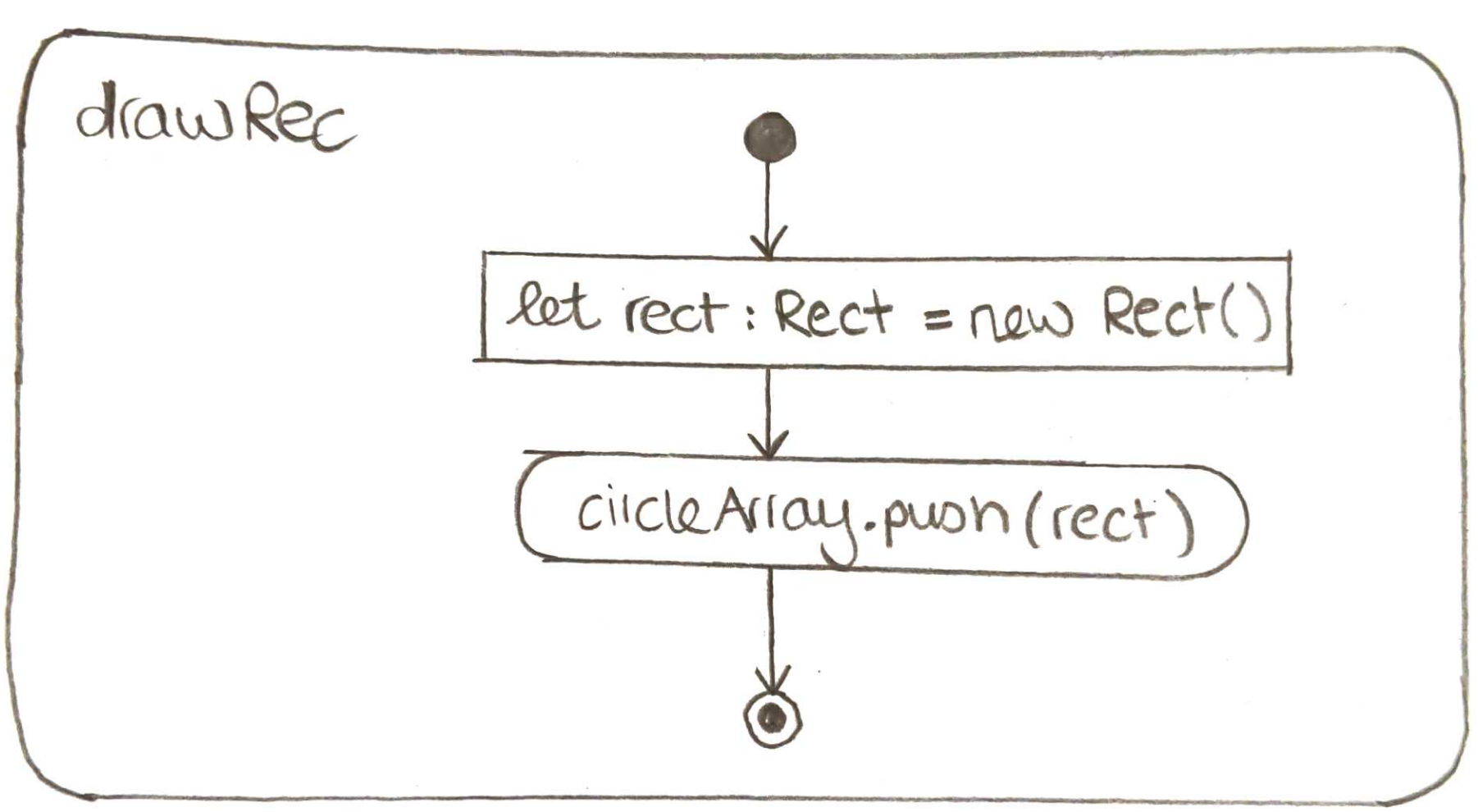
buttonsAreCreated = false

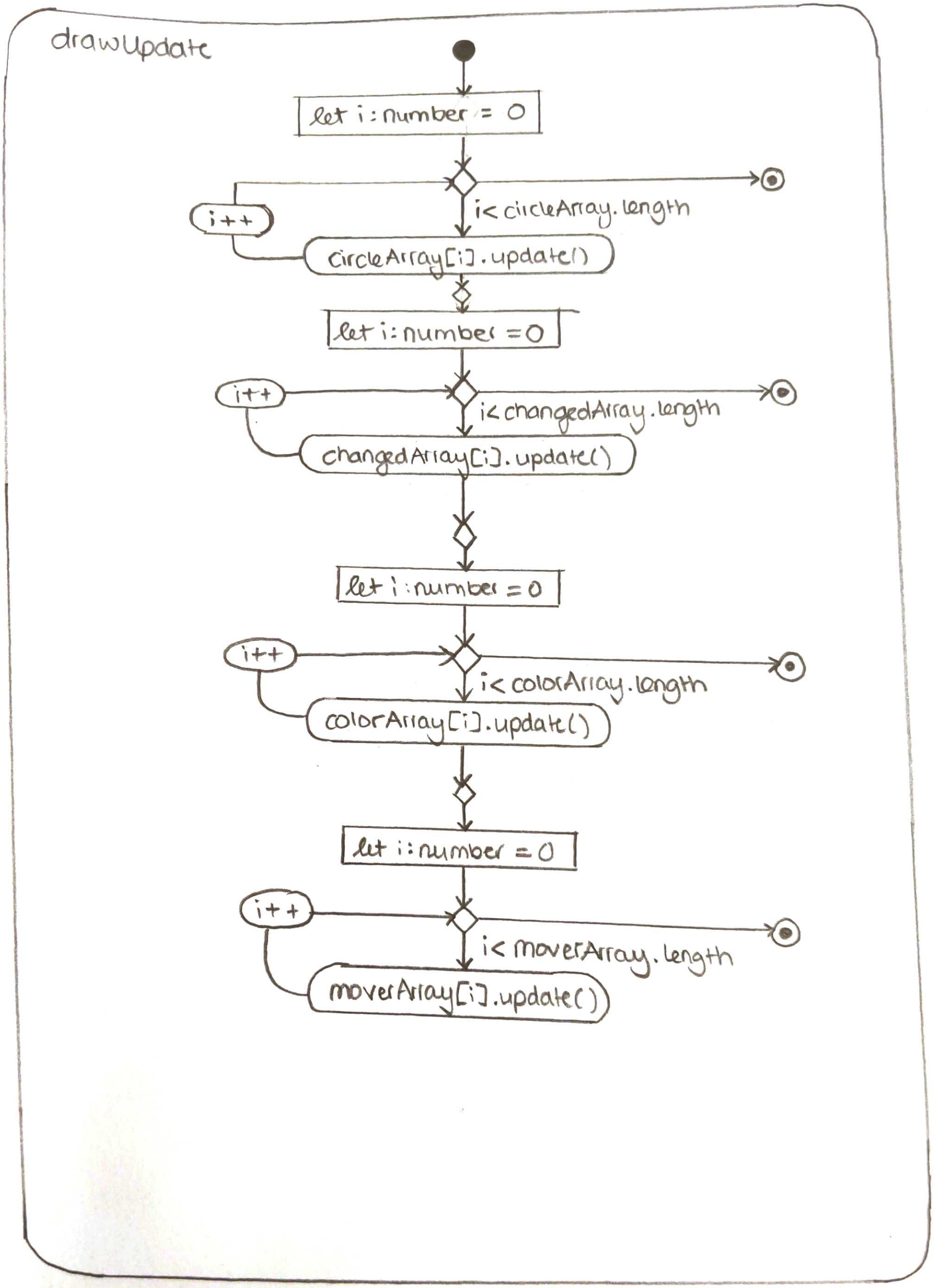
createAnimationsButtons ()

buttonsAreCreated = true

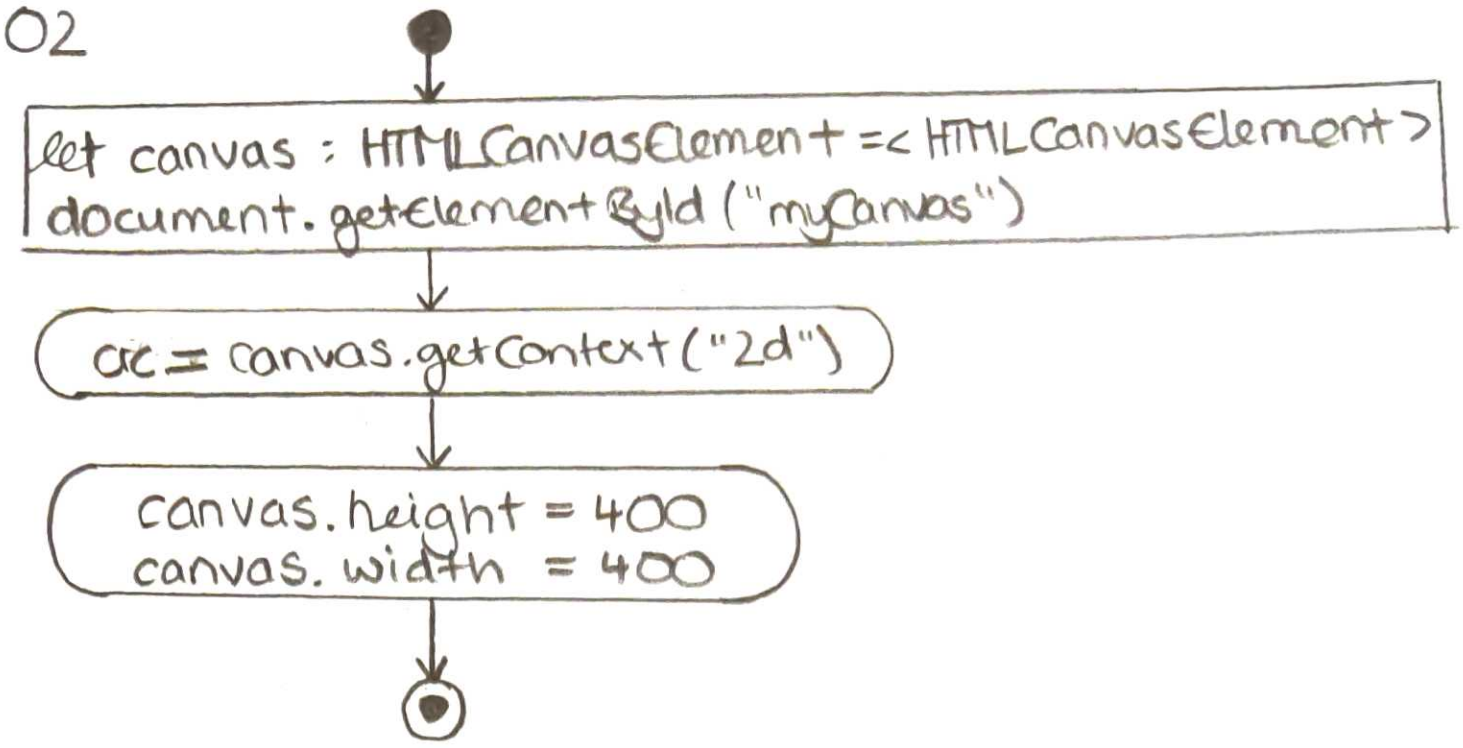
boolean = false


drawCircle
let circle : Circle = new Circle()
CircleArray.push(circle)

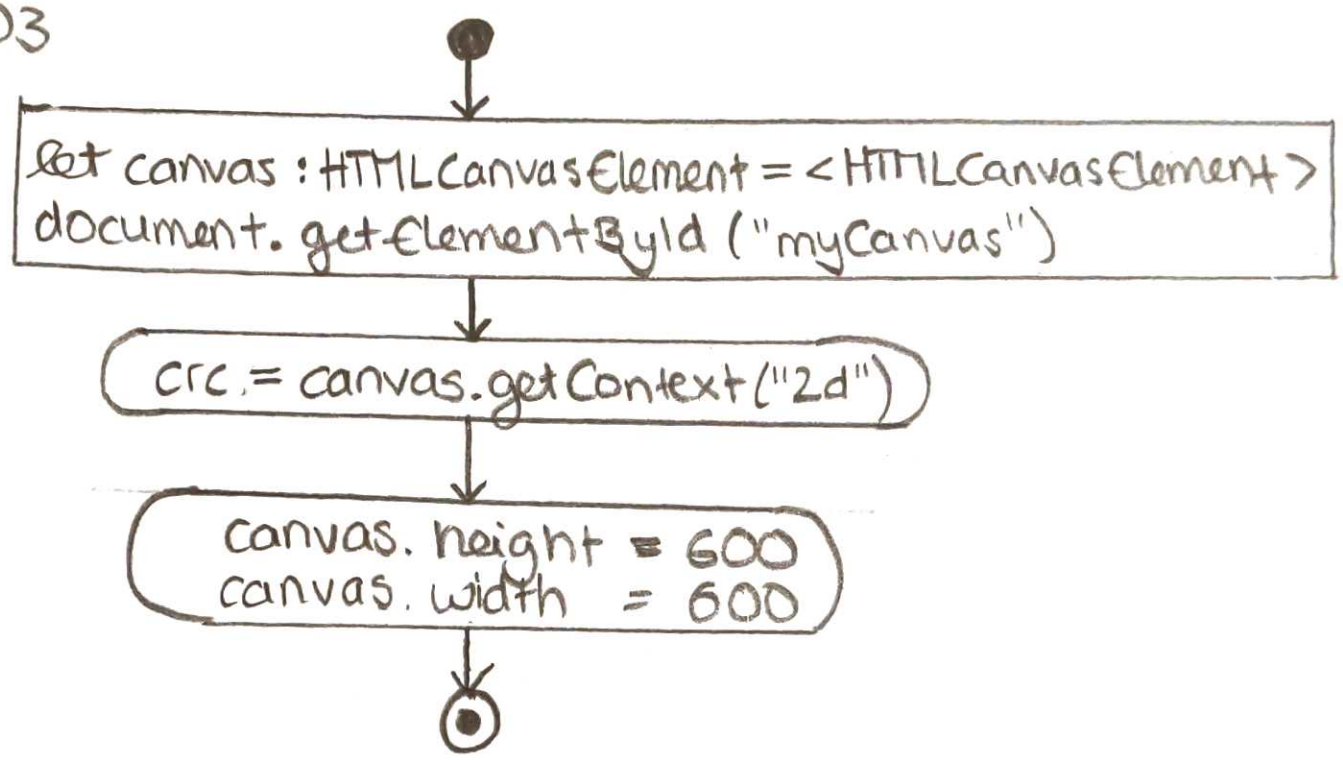

drawRec
let rect : Rect = new Rect()
circleArray.push(rect)


drawUpdate
let i : number = 0
i < circleArray.length
i++
circleArray[i].update()
let i: number =0
i++
i< changedArray.length
changedArray[i].update()
let i : number = 0
i++
i< colorArray.length
colorArray[i].update()
let i : number = 0
i++
i< moverArray.length
moverArray[i].update()

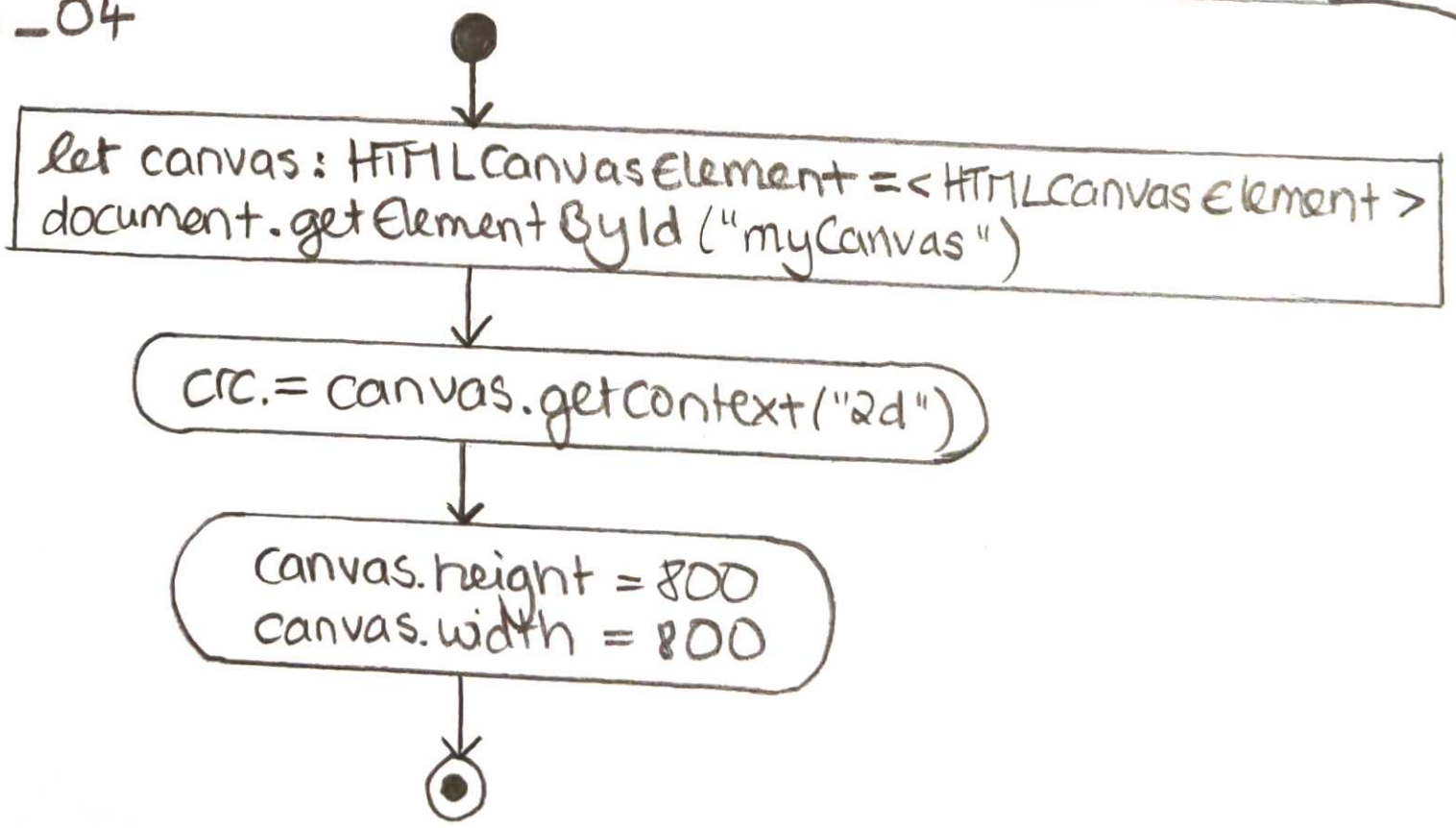
resizeCanvas_02



resizeCanvas_03



resizeCanvas_04



background

backgroundColor = "lightblue"

background01

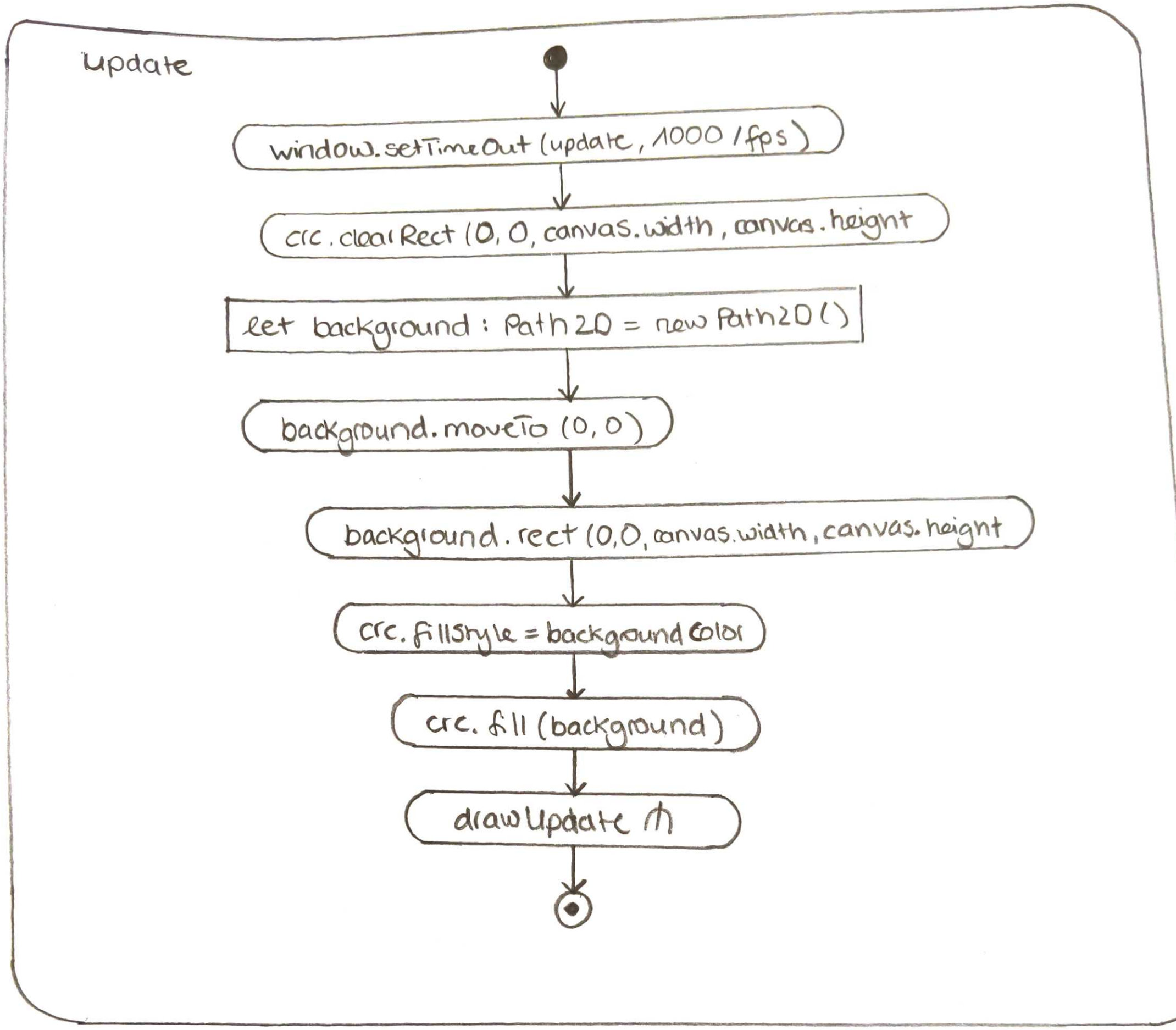
backgroundColor = "white"

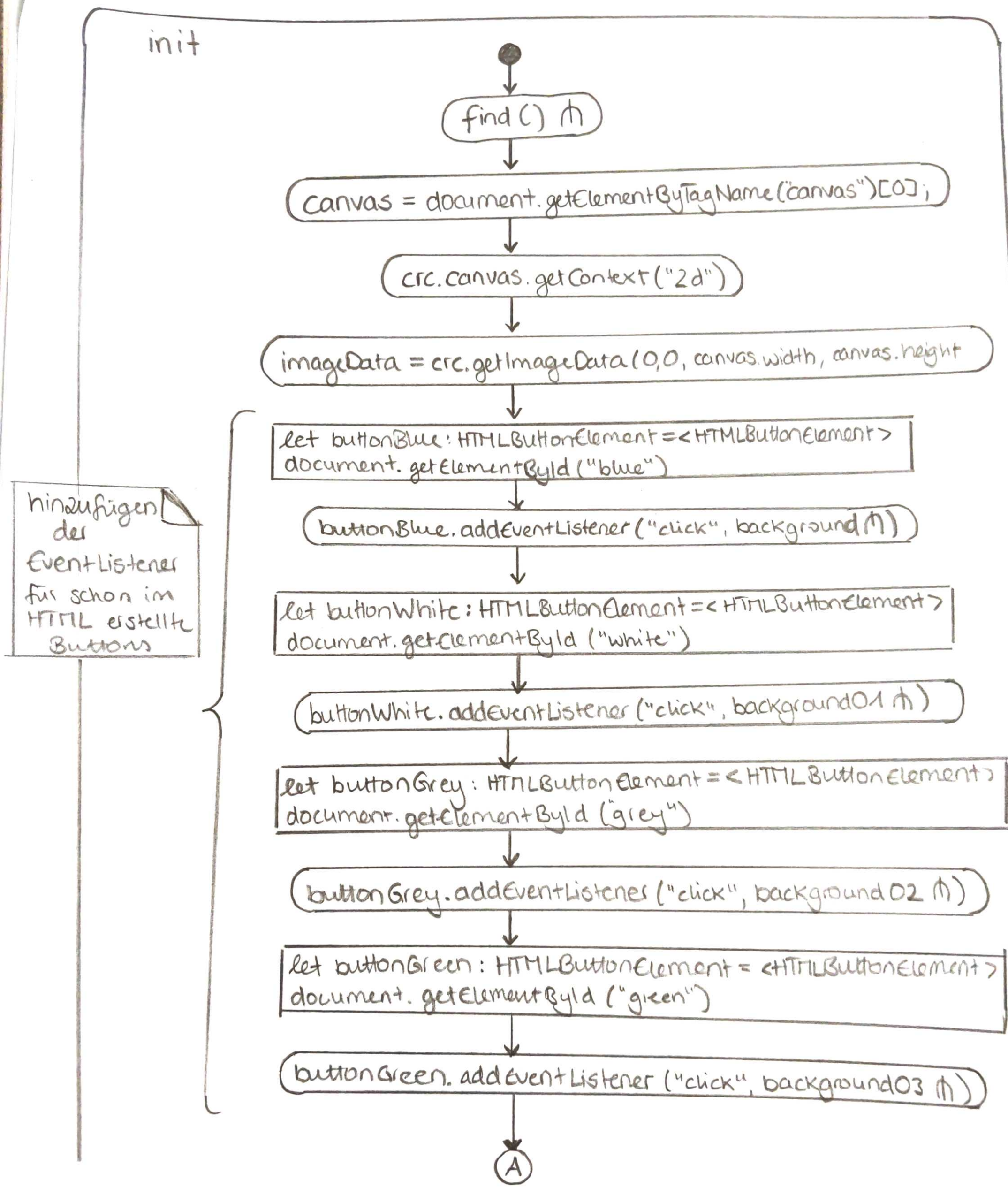
background02

backgroundColor = "lightgrey"

background03

backgroundColor = "green"


update
window.setTimeOut(update, 1000/fps)
crc.clearRect(0, 0, canvas.width, canvas.height
let background : Path2D = new Path2D()
background.moveTo(0, 0)
background.rect(0,0, canvas.width, canvas.height
crc.fillStyle = backgroundColor
crc.fill(background)
drawUpdate()


init
find () ⋔
canvas = document.getElementByTagName("canvas")[0];
crc.canvas.getContext("2d")
imageData = crc.getImageData(0,0, canvas.width, canvas.height
let buttonBlue: HTMLButtonElement=<HTMLButtonElement>
document.getElementById("blue")
buttonBlue.addEventListener("click", background ⋔)
let buttonWhite: HTMLButtonElement=<HTMLButtonElement>
document.getElementById("white")
buttonWhite.addEventListener("click", background01 ⋔)
let buttonGrey: HTMLButtonElement=<HTMLButtonElement>
document.getElementById("grey")
buttonGrey.addEventListener("click", background02 ⋔)
let buttonGreen: HTMLButtonElement=<HTMLButtonElement>
document.getElementById("green")
buttonGreen.addEventListener("click", background03 ⋔)
A
hinzufügen der EventListener für schon im HTML erstellte Buttons

A

let onchange: HTMLButtonElement = <HTMLButtonElement>
document.getElementById("small")

onchange.addEventListener("change", resizeCanvas_01)

Radiobuttons zur Festlegung der Größe des Canvas

let onchange01: HTMLButtonElement = <HTMLButtonElement>
document.getElementById("medium")

onchange01.addEventListener("change", resizeCanvas_02)

let onchange02: HTMLButtonElement = <HTMLButtonElement>
document.getElementById("large")

onchange02.addEventListener("change", resizeCanvas_03)

Radiobutton check ob man Objekte bearbeiten darf

let edit: HTMLInputElement = <HTMLInputElement>
document.getElementById("edit")

edit.addEventListener("change", select)

buttons zum erstellen von Kreisen/ Vierecken

let createCircle: HTMLButtonElement = <HTMLButtonElement>
document.getElementById("circle")

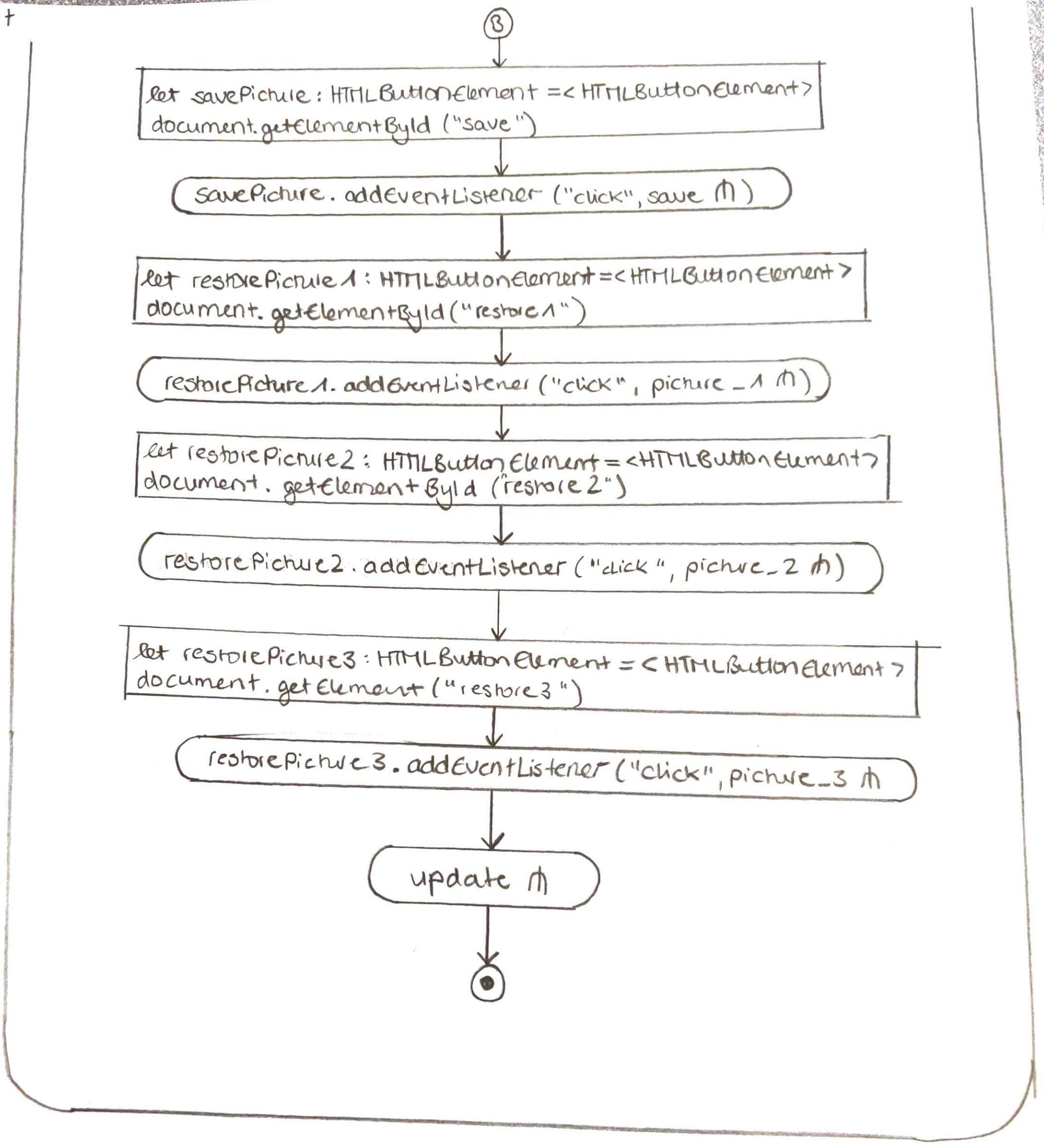
createCircle.addEventListener("click", drawCircle)

let createRect: HTMLButtonElement = <HTMLButtonElement>
document.getElementById("rect")

createRect.addEventListener("click", drawRect)

B

init


B
let savePicture : HTMLButtonElement = <HTMLButtonElement>
document.getElementById ("save")
savePicture . addEventListener ("click", save ⇑)
let restorePicture1 : HTMLButtonElement = <HTMLButtonElement>
document. getElementById ("restore1")
restorePicture1. addEventListener ("click", picture_1 ⇑)
let restorePicture2 : HTMLButtonElement = <HTMLButtonElement>
document . getElement+Byld ("restore2")
restorePicture2 . addEventListener ("click", picture_2 ⇑)
let restorePicture3 : HTMLButtonElement = <HTMLButtonElement>
document . getElement ("restore3")
restorePicture3 . addEventListener ("click", picture_3 ⇑
update ⇑