id = "mycanvas"

400 || 600 || 800 width

400 || 600 || 800 width

backgroundColor

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

animations

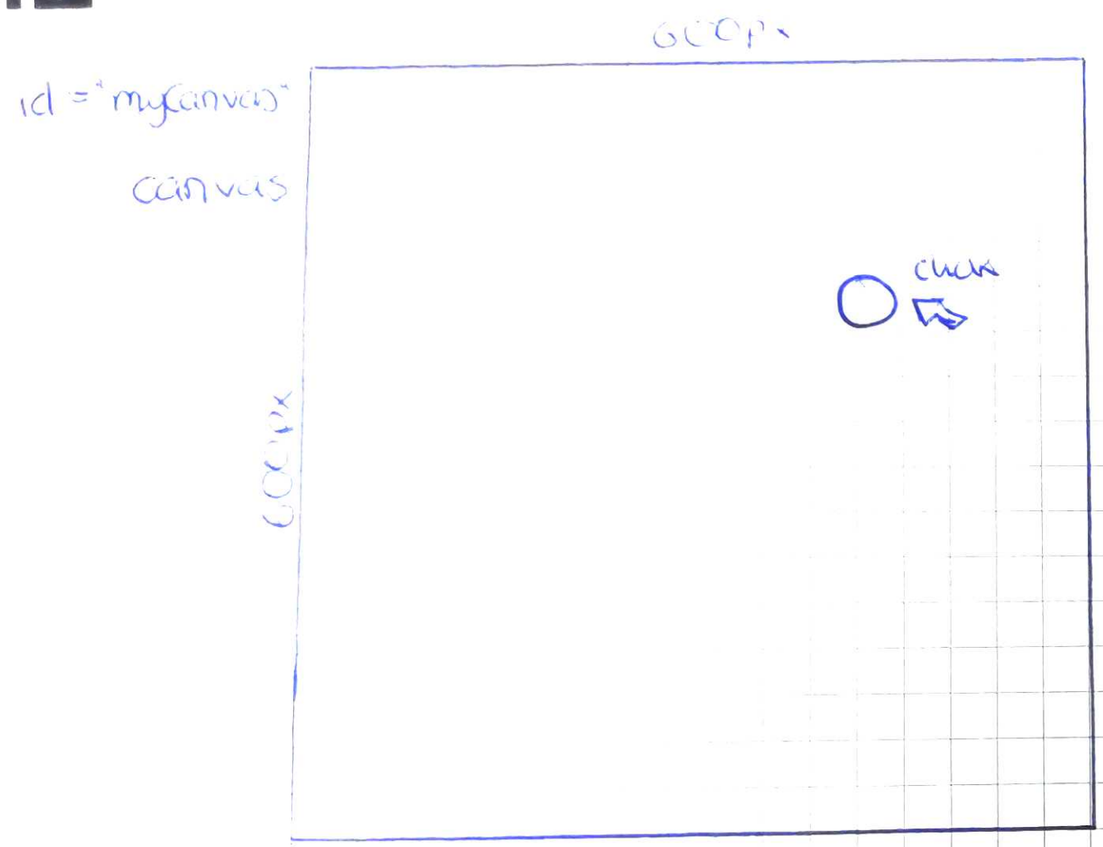
} in dieses Div werden die buttons von der function "createAnimationButtons" geschrieben

save and load

id="save" } Button

saved pictures

} buttons



background-color

blue | white | gray | green } fieldset } buttons

edit selected shape

○ edit ○ not edit } radiobutton } fieldset

size of canvas

○ small ○ medium ○ large } radiobutton } fieldset

Circle

place circle } button Form erstellen
drag | delete | rainbow } buttons "animations"
○ purple ○ pink ○ blue } radiobuttons Farbe Form
} fieldset

Rectangle

place rect } button Form erstellen
drag | delete | move } buttons "animation"
○ purple ○ pink ○ blue } radiobuttons Farbe Form
} fieldset

canvas id="myCanvas"

background-color options
blue (id=blue) | white (id=white) | grey (id=grey) | green (id=green)
} legend
} buttons
} fieldset

edit shapes
○ edit (id=edit) ○ dont edit
} legend
} radiobuttons
} fieldset

size of canvas
○ small (id=small) ○ medium (id=medium) ○ large (id=large)
} legend
} radiobuttons
} fieldset id=canvasSize

Circle
place circle (id=circleId)
color change
○ purple (id=purple) ○ pink (id=pink) ○ blue (id=blue)
} legend
} button
} legend
} radiobuttons
} fieldset id=colorOptions
} fieldset

Rectangle
place rectangle (id=rect)
color change
○ purple (id=purple1) ○ pink (id=pink1) ○ blue (id=blue1)
} legend
} button
} legend
} radiobuttons
} fieldset id=colorOptions
} fieldset

save
save picture (id=save)
} button
} legend
} fieldset

saved pictures
id=restore1 | id=restore2 | id=restore3
id=restore4 | id=restore5 | id=restore6
} legend
} buttons
} fieldset

</body>

Mainclass

x : number
y : number
dx : number
dy : number
a : number
size : number
color : number
rainbow : boolean
type : string
moving : boolean
newColor : boolean

draw()
update()
move()
constructor()

Cube

x = Math.random()*crccanvas.width.
y = Math.random()*crccanvas.height.
dx = 0
dy = 0
a = 1
size = 8
color = "SteelBlue" || "pink" || "purple"
rainbow = false
newColor = false
type = "circle"
moving = false

constructor()
draw()
move()
update()

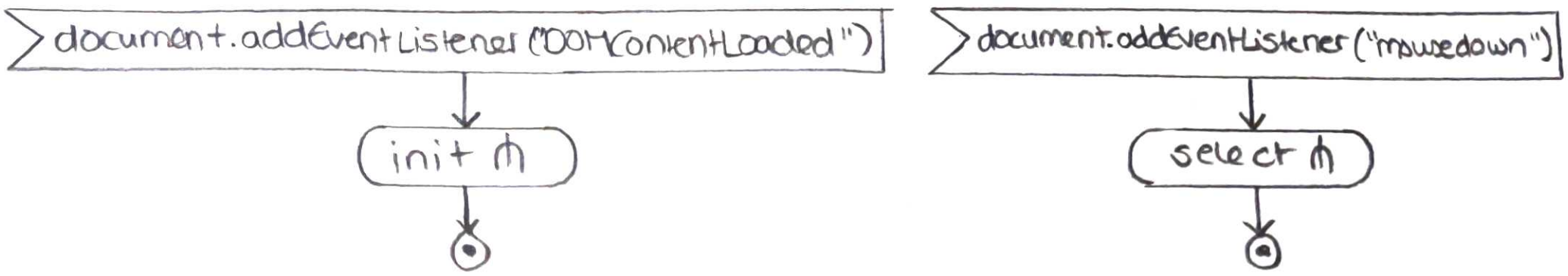
Rect

x = Math.random()*canvas.width
y = Math.random()*canvas.height
dx = 0
dy = 0
a = 1
size = 8
color = "SteelBlue" || "pink" || "purple"
rainbow = false
type = "rec"
moving = false

constructor()
draw()
move()
update()

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 Halfläche einstellen
- man kann Bilder abspeichern und sie zu einem späteren Zeitpunkt weiterbearbeiten



Globale Variabeln + Arrays

```
export let crc: CanvasRenderingContext2D;

export let circleArray : Circle [] = []
export let changedArray : Circle [] = []
export let colorArray : Circle[] = []
export let moveArray : Circle [] = []
export let deleteArray : Circle [] = []
export let backgroundColor : string = "white"
export let canvas : HTMLCanvasElement

let pushedObject : number
let fps : number = 30
let imageData : ImageData
let draggedObject : boolean = false
let boolean : boolean = false
let buttonsAreCreated : boolean = false
```

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")

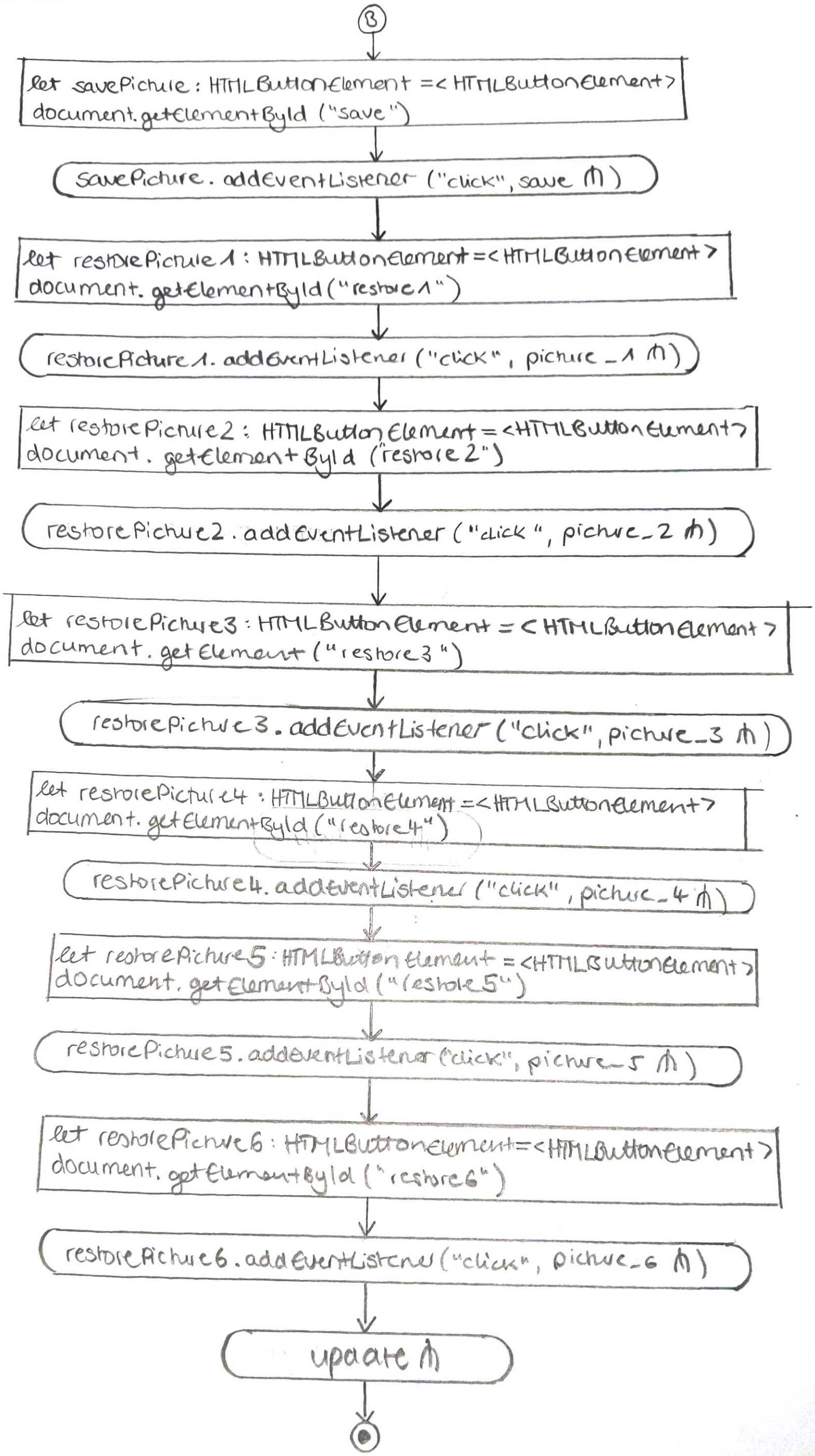
createCircle. addEventListener ("click", drawCircle)

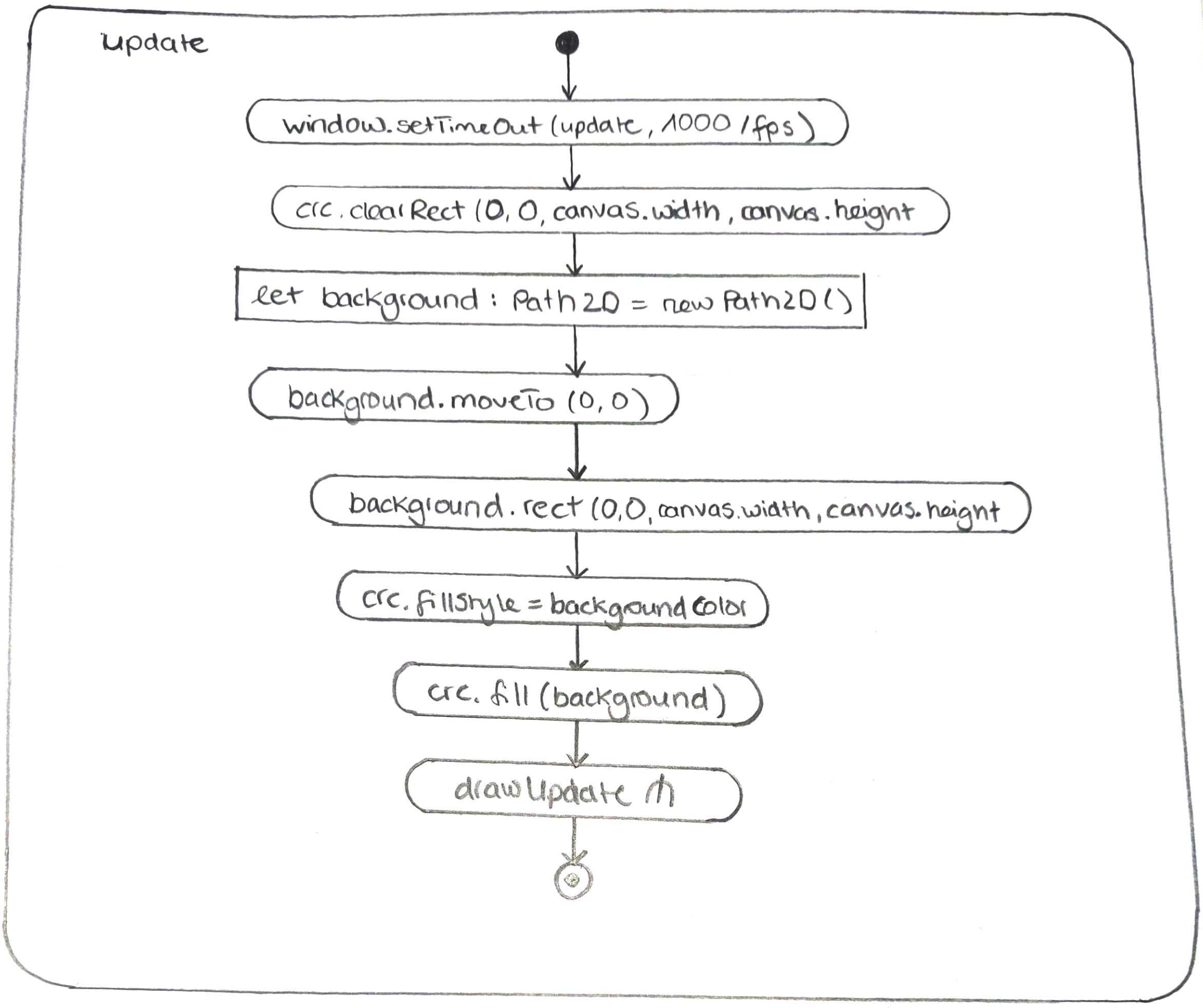
let createRect : HTMLButtonElement = <HTMLButtonElement>
document. getElementById ("rect")

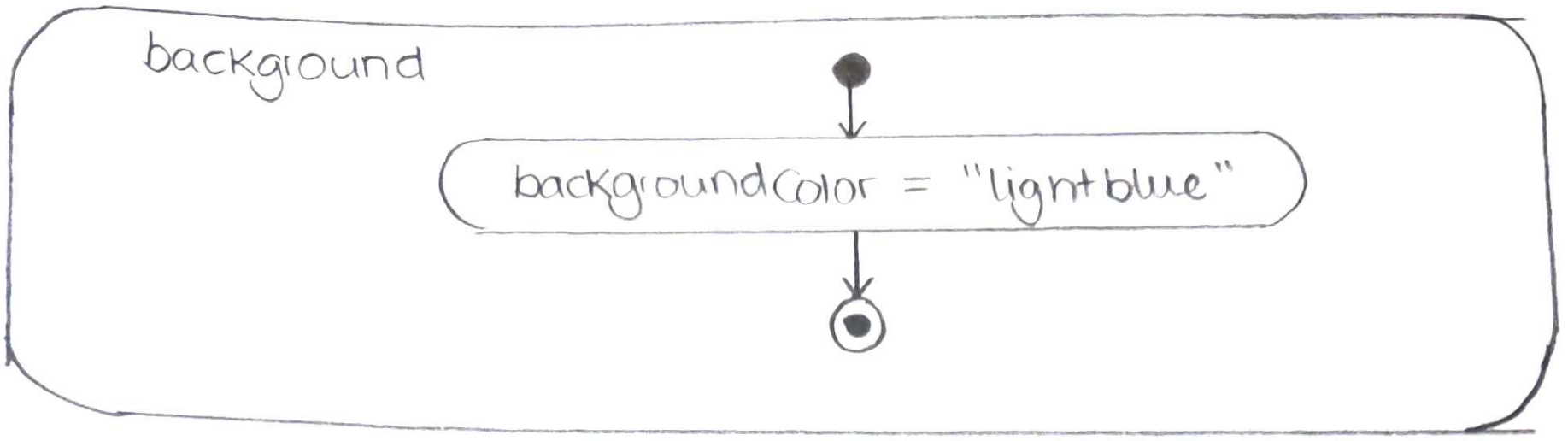
createRect. addEventListener ("click", drawRect)

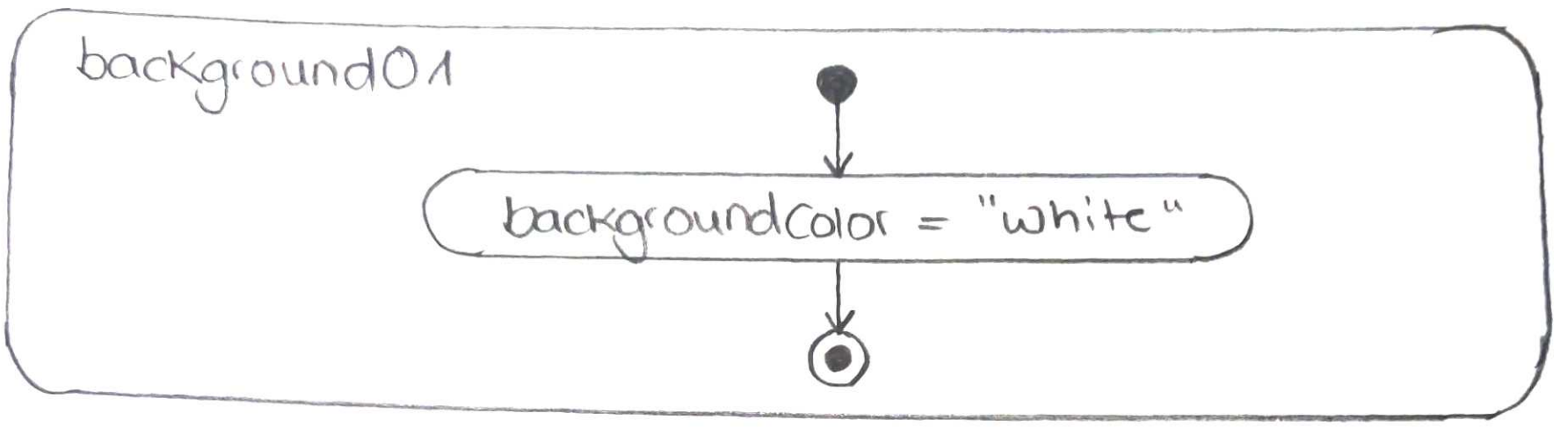
B

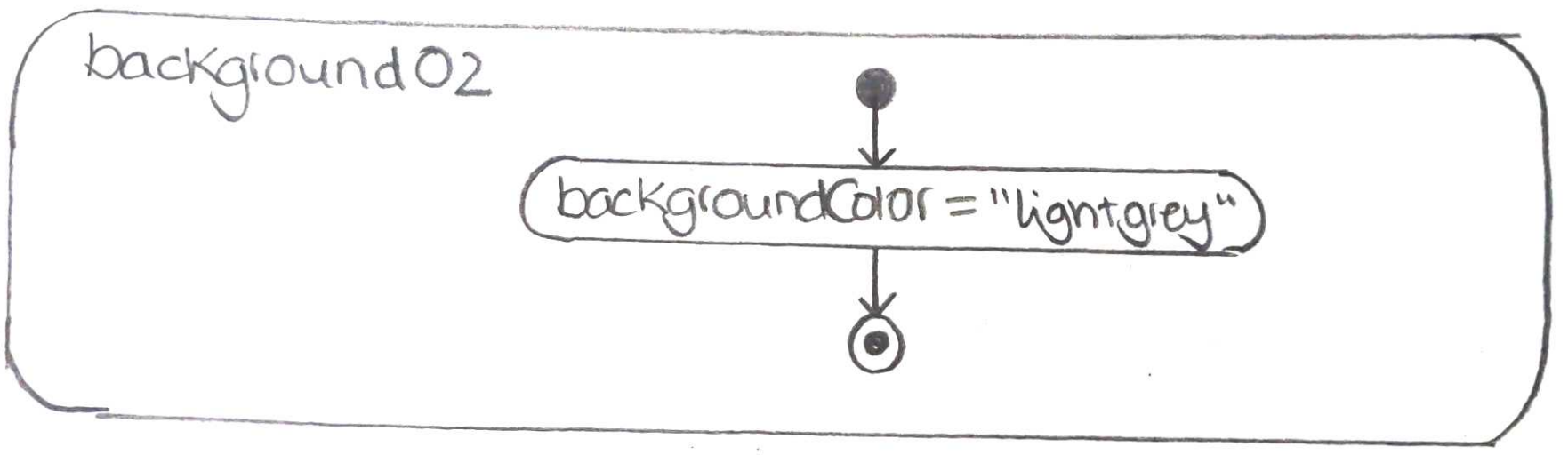
ini init

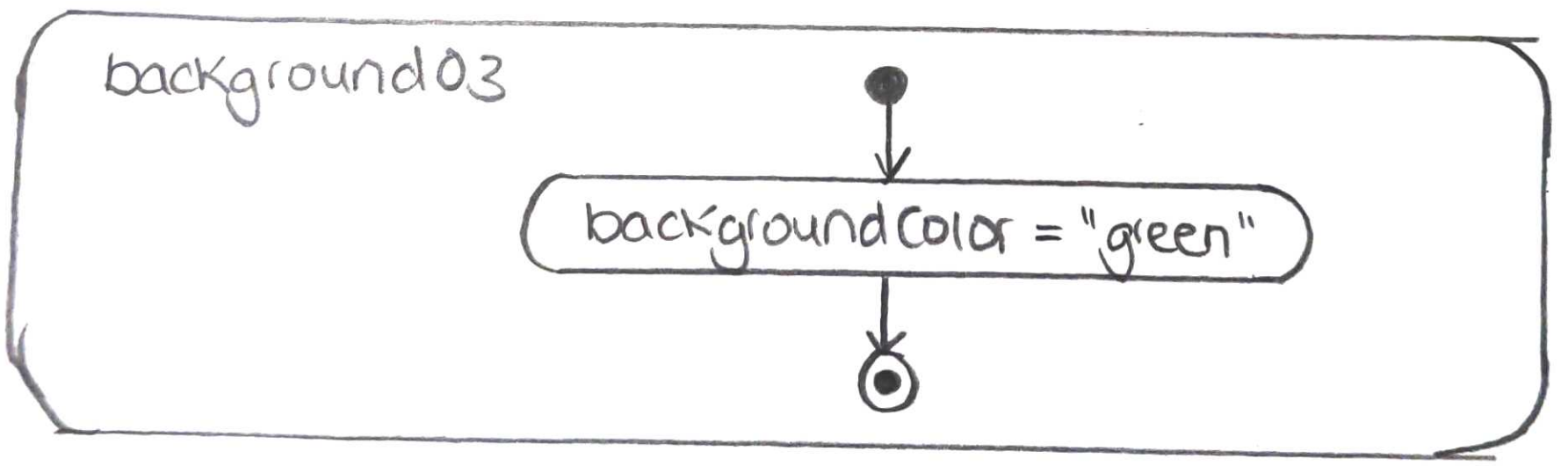

B
let savePicture : HTMLButtonElement = <HTMLButtonElement>
document.getElementById ("save")
savePicture . addEventListener ("click", save ⇑)
let restorePicture1 : HTMLButtonElement = <HTMLButtonElement>
document. getElementById ("restore1")
restorePicture1. addEventListener ("click", picture_1 ⇑)
let restorePicture2 : HTMLButtonElement = <HTMLButtonElement>
document . getElement+Byld ("restore2")
restorePicture2 . addEventListener ("click", picture_2 ⇑)
let restorePicture3 : HTMLButtonElement = <HTMLButtonElement>
document . getElement ("restore3")
restorePicture3 . addEventListener ("click", picture_3 ⇑)
let restorePicture4 : HTMLButtonElement = <HTMLButtonElement>
document. getElementById ("restore4")
restorePicture4. addEventListener ("click", picture_4 ⇑)
let restorePicture5 : HTMLButtonElement = <HTMLButtonElement>
document. getElementById ("restore5")
restorePicture5. addEventListener ("click", picture_5 ⇑)
let restorePicture6 : HTMLButtonElement = <HTMLButtonElement>
document. getElementById ("restore6")
restorePicture6. addEventListener ("click", picture_6 ⇑)
update ⇑


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()

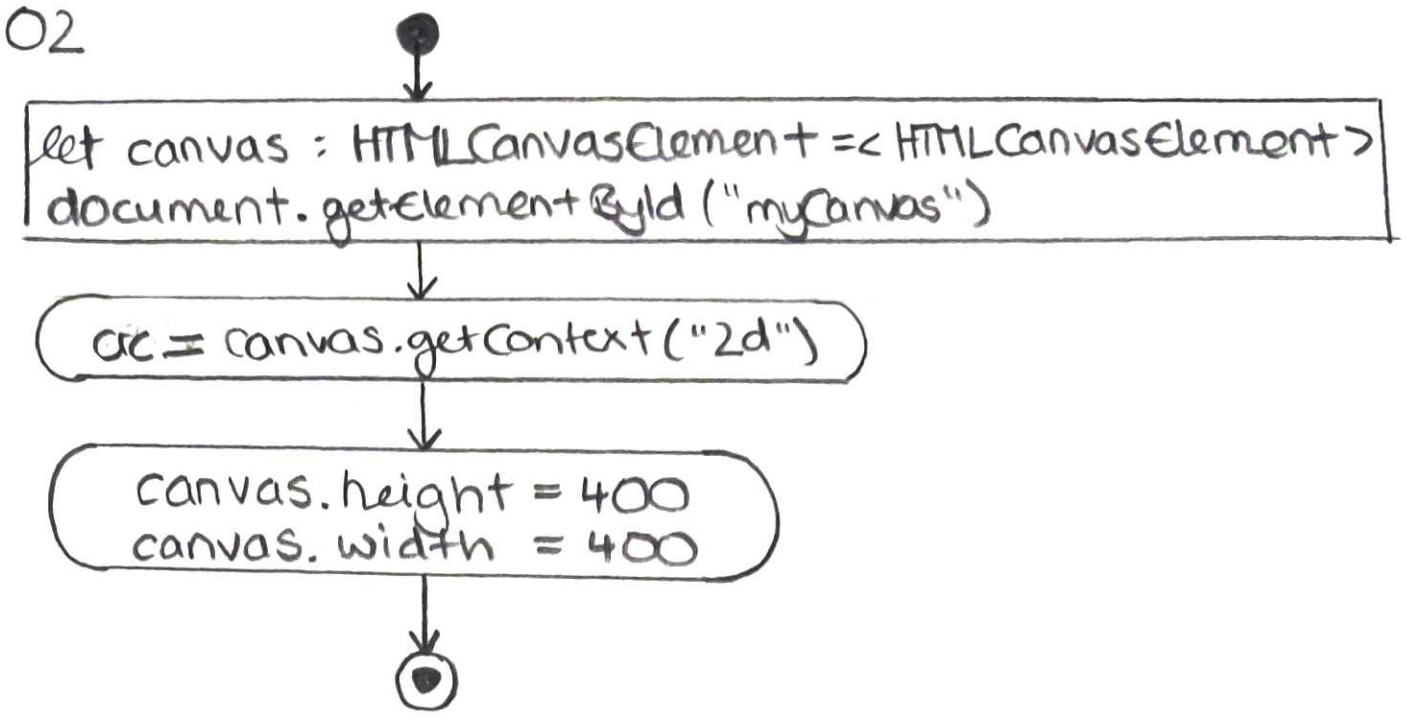

background
backgroundColor = "lightblue"


background01
backgroundColor = "white"

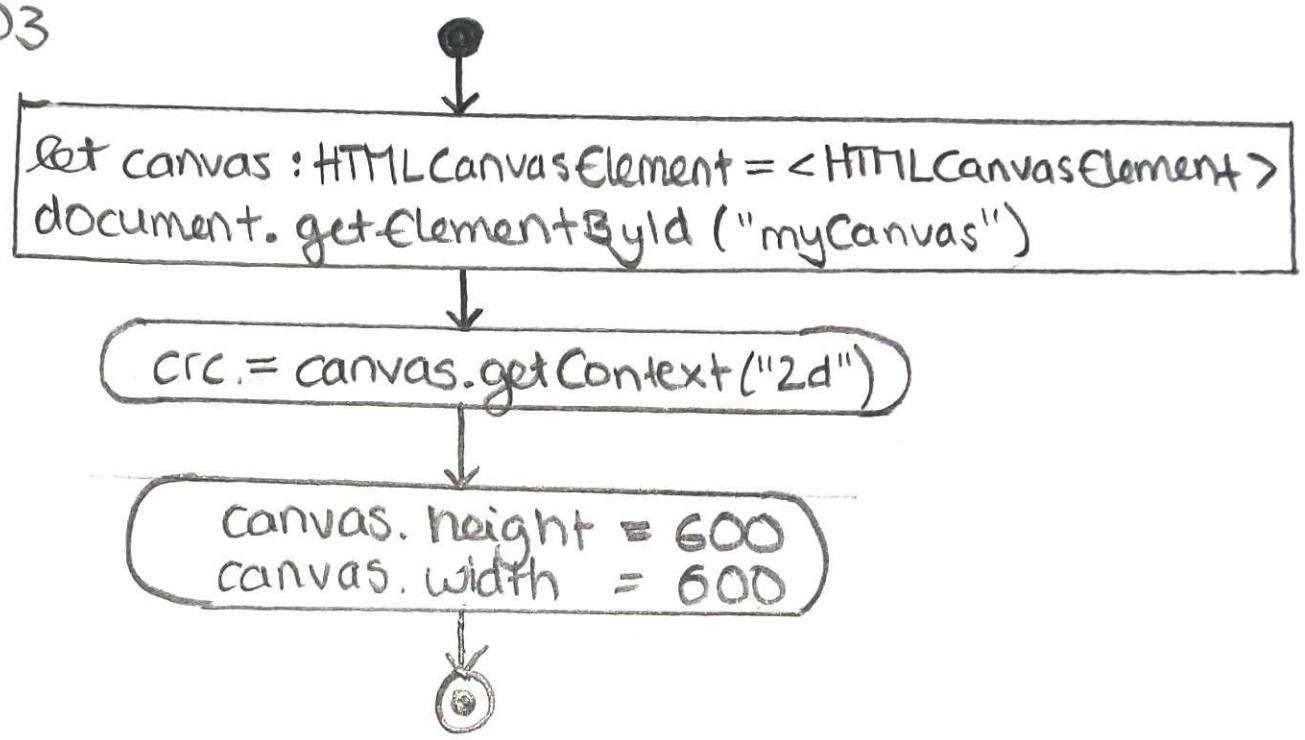

background02
backgroundColor = "lightgrey"


background03
backgroundColor = "green"

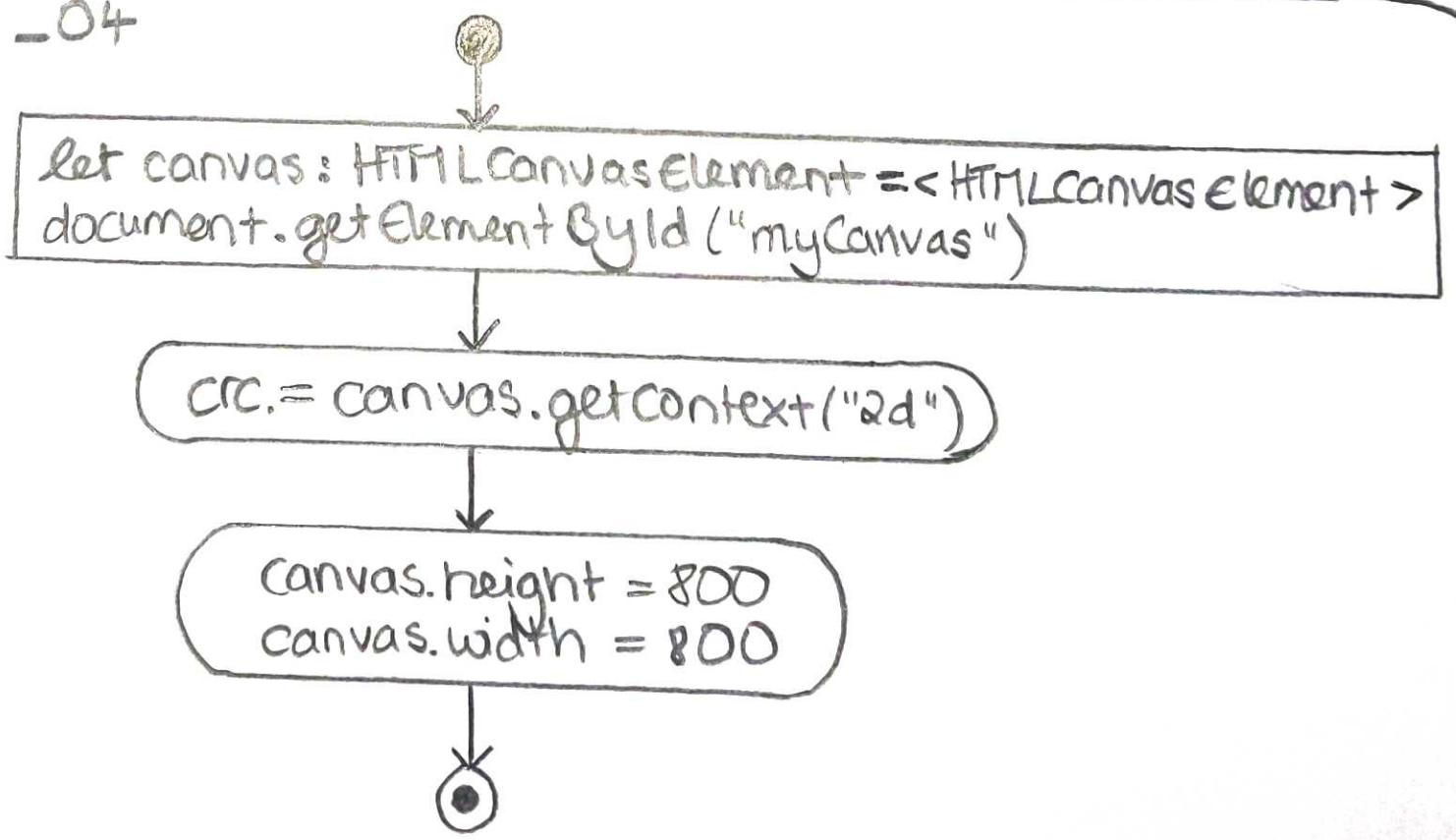
resizeCanvas_02



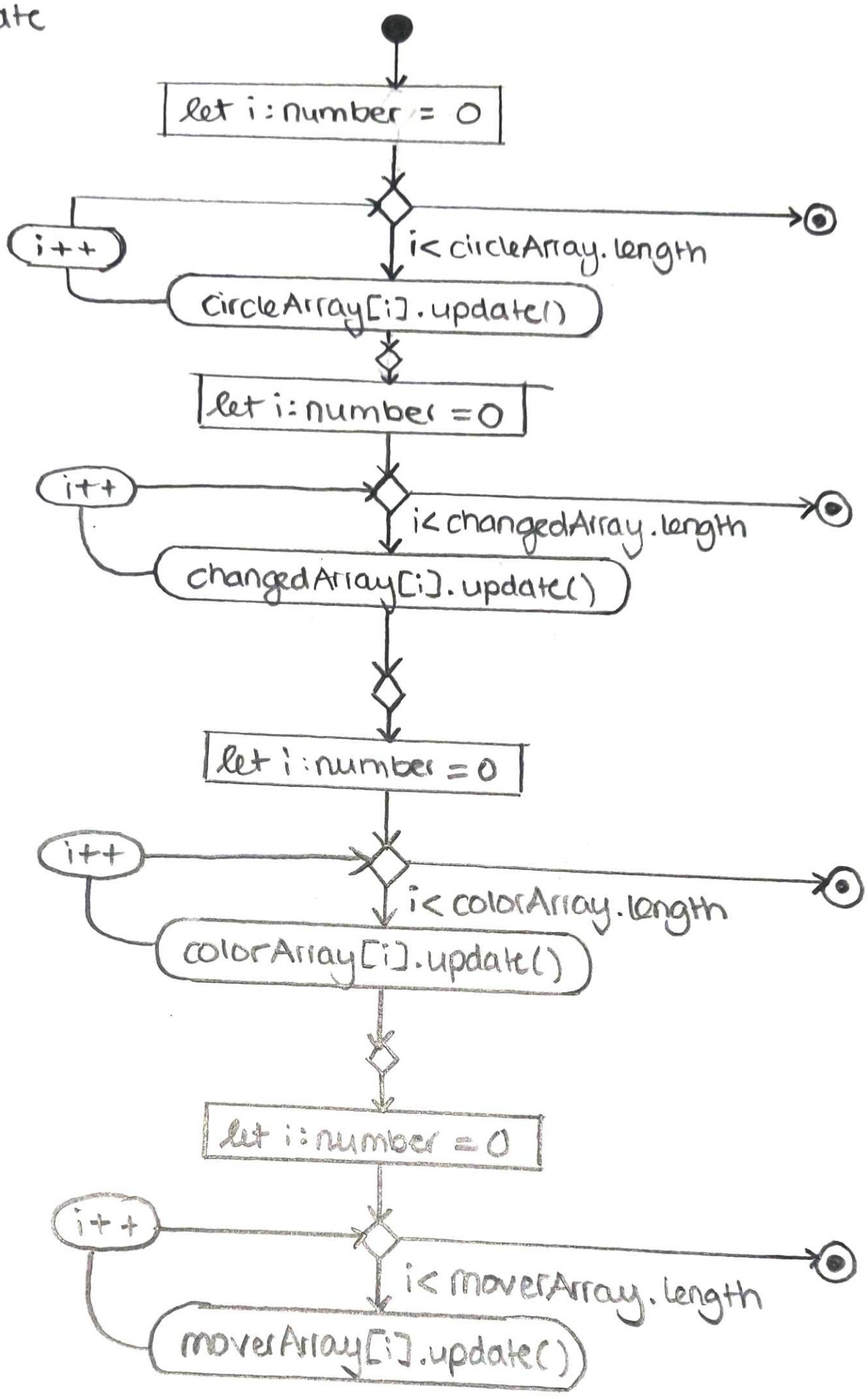
resizeCanvas_03

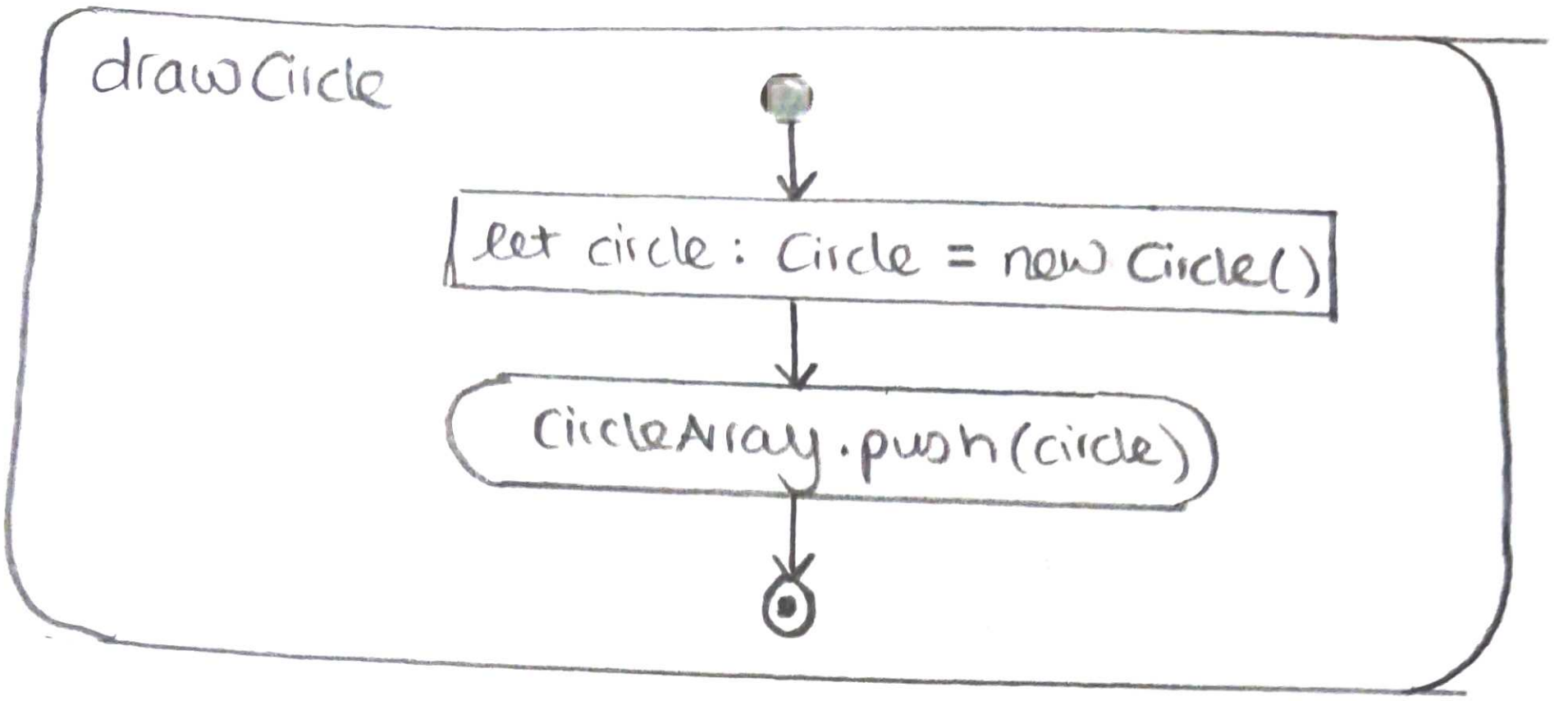


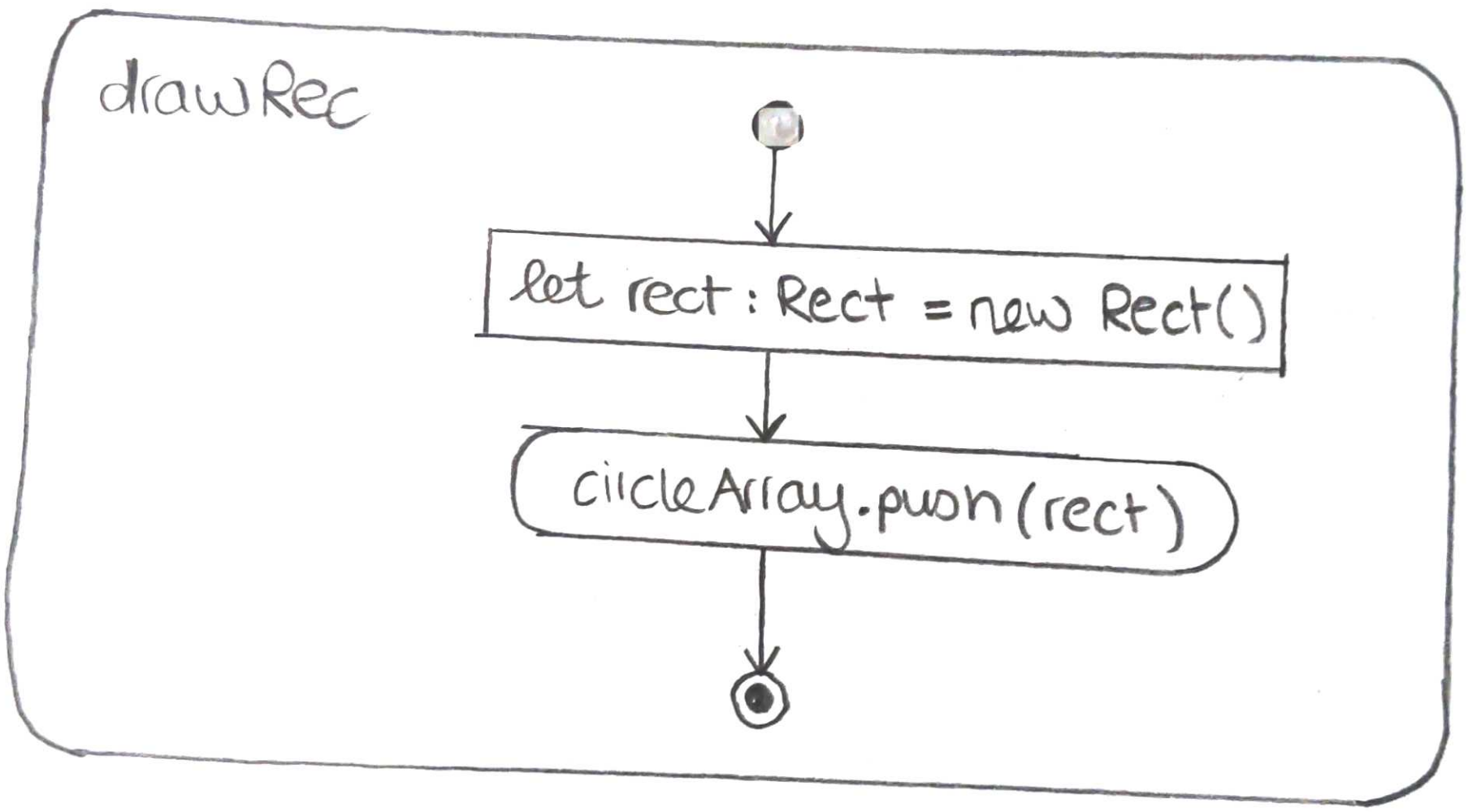
resizeCanvas_04



drawUpdate


let i:number = 0
i++
i< circleArray.length
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()


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


drawRec
let rect : Rect = new Rect()
ciicleArray.puon(rect)

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 (bei 10)
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 < current X + 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

createAnimationButtons

let buttonDelete : HTMLButtonElement = <HTMLButtonElement>
document.createElemente ("button")

buttonDelete.innerHTML = "delete"

document.getElementById("animations").appendChild(buttonDelete)
buttonDelete.addEventListener ("click", deleteObjects ())

let buttonMove: HTMLButtonElement = <HTMLButtonElement>
document.createElement("button")

buttonMove.innerHTML = "move"
document.getElementById("animations").appendChild(buttonMove)
buttonMove.addeventListener("click", moverObjects ())

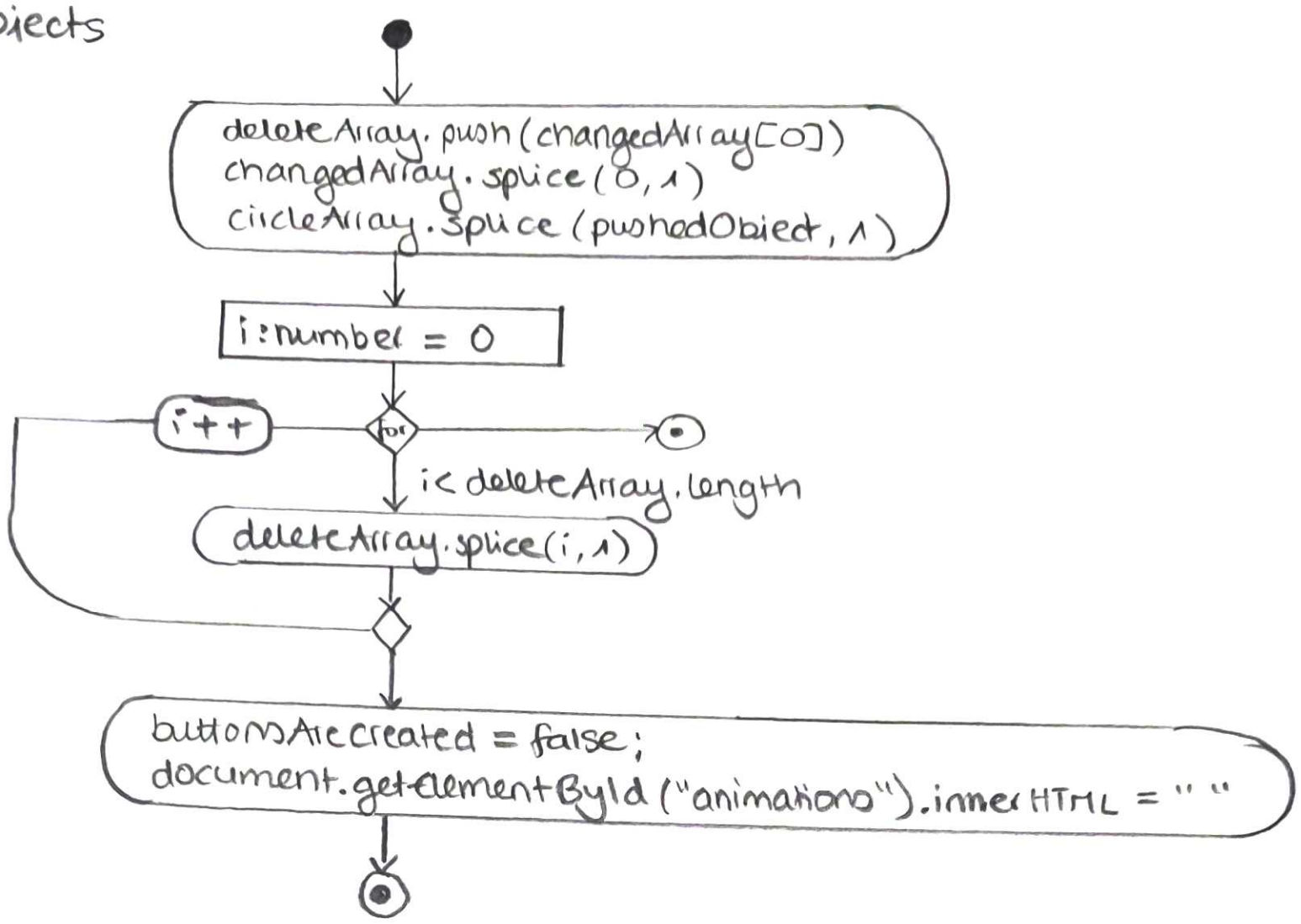
let buttonDrag : HTMLButtonElement = <HTMLButtonElement>
document.createElement("button")

buttonDrag.innerHTML = "drag"
document.getElementById("animations").appendchild(buttonDrag)
buttonDrag.addEventListener("click", dragObjects ())

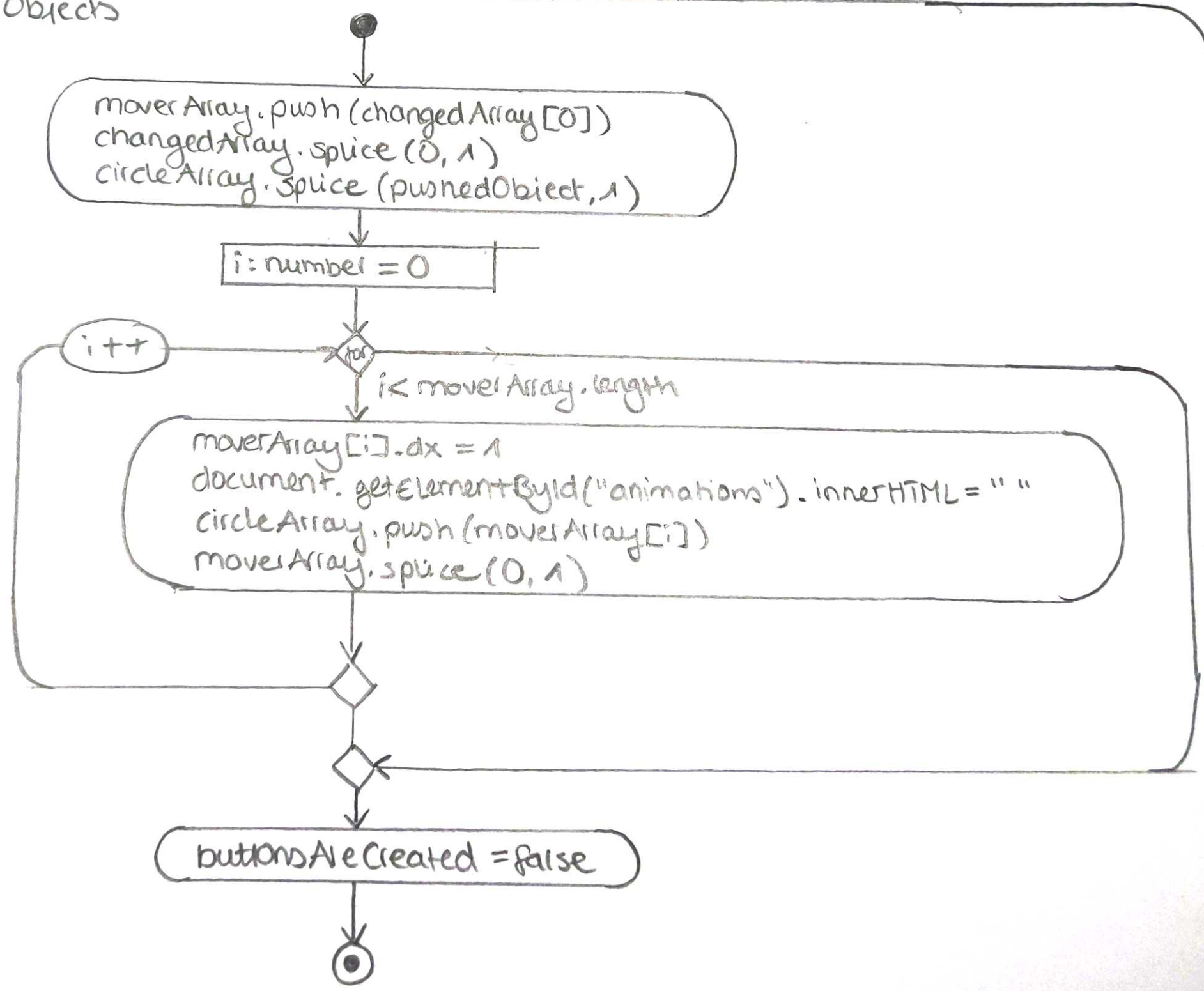
let buttonColor : HTMLButtonElement = <HTMLButtonElement>
document.createElement("button")

buttonColor.innerHTML = "rainbow"
document.getElementById("animations").appendChild(buttonColor)
buttonColor.addEventListener("click", colorObjects ()

deleteObjects



moverObjects



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 , -newClientX : 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

dragObjects

```
circleArray.splice(pushedObjects, 1)
draggedObject = true
document.getElementById("animations").innerHTML = ""
```

colorObjects

```
colorArray.push(changedArray[0])
changedArray.splice(0, 1)
circleArray.splice(pushedObjects, 1)
```

```
i: number = 0
```

for (i++ ; i < colorArray.length)

```
document.getElementById("animations").innerHTML = ""
circleArray.push(colorArray[i])
colorArray.splice(0, 1)
```

```
buttonsAreCreated = false
```

save

```
let saveName : string = prompt("name")
inner+ (saveName)
```

picture_1

```
rebuild(0)
```

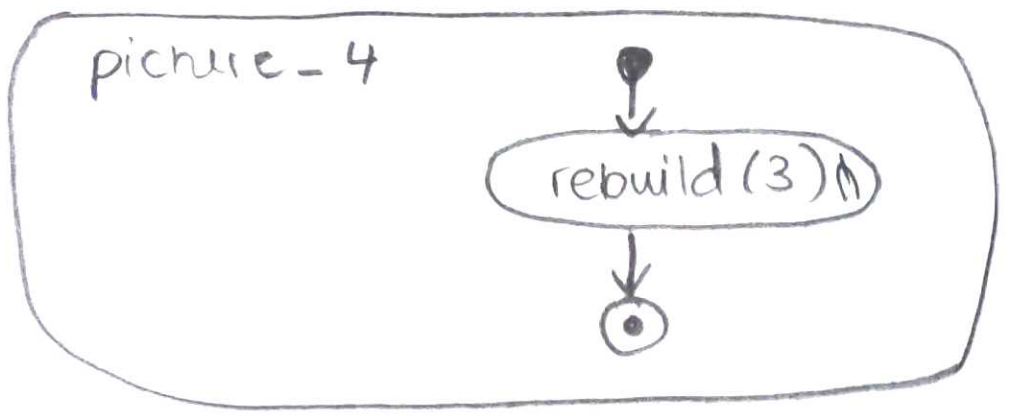
picture_2

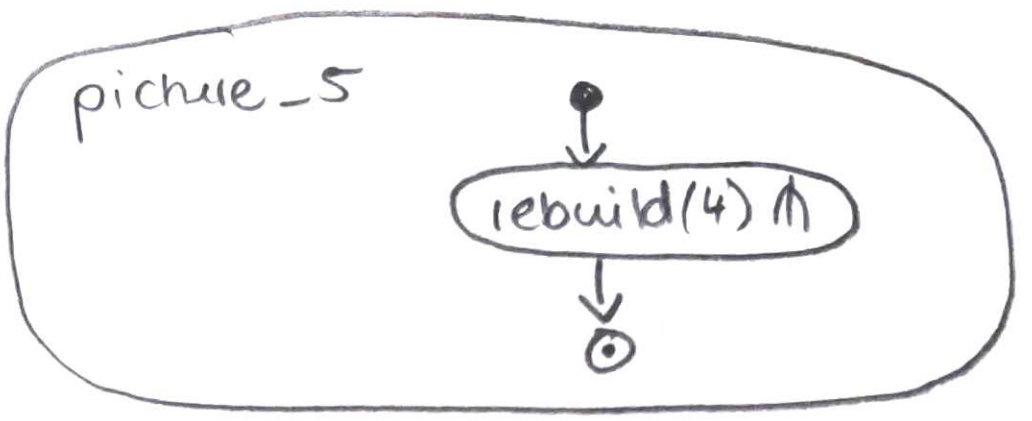
```
rebuild(1)
```

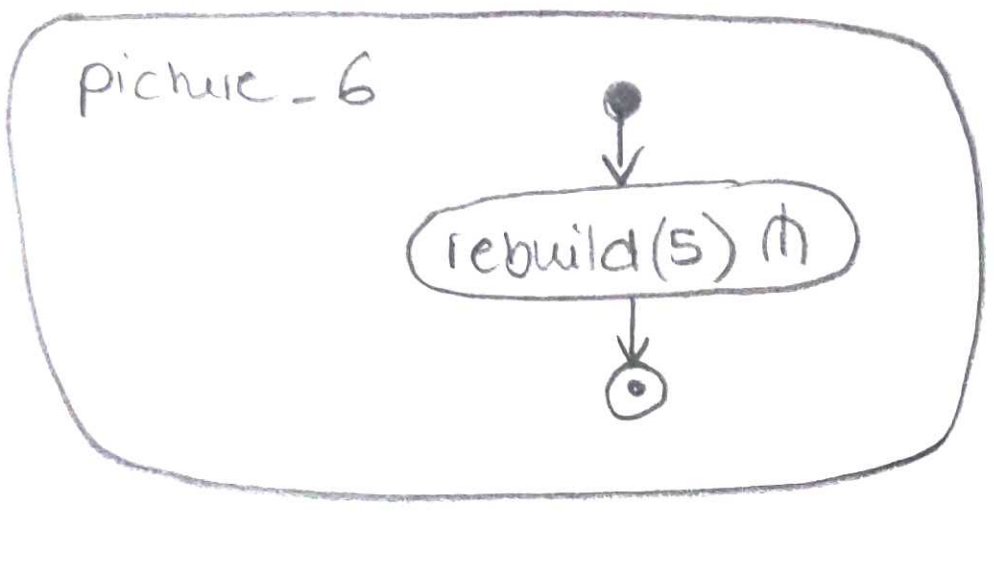
picture_3

```
rebuild(2)
```

→


picture_4
rebuild(3)


picture_5
rebuild(4)


picture_6
rebuild(5)

rebuild

-u: number

alle Objekte aus Array

let i: number = 0

for

i< circleArray.length

circleArray.splice(0, circleArray.length

i++

Werte aus Query-String als Rückgabe

let xPos : string = canvasPic[-u].x
let yPos : string = canvasPic[-u].y
let background : string = canvasPic[-u].backgroundcolor
let type : string = canvasPic[-u].type
let rainbow: string = canvasPic[-u].rainbow
let move : string = canvasPic[-u].move
let width : string = canvasPic[-u].width

backgroundColor = background

if

width == "400"

resizeCanvas_02 ()

if

width == "600"

resizeCanvas_03 ()

if

width == "800"

resizeCanvas_04 ()

let i: number = 0

for

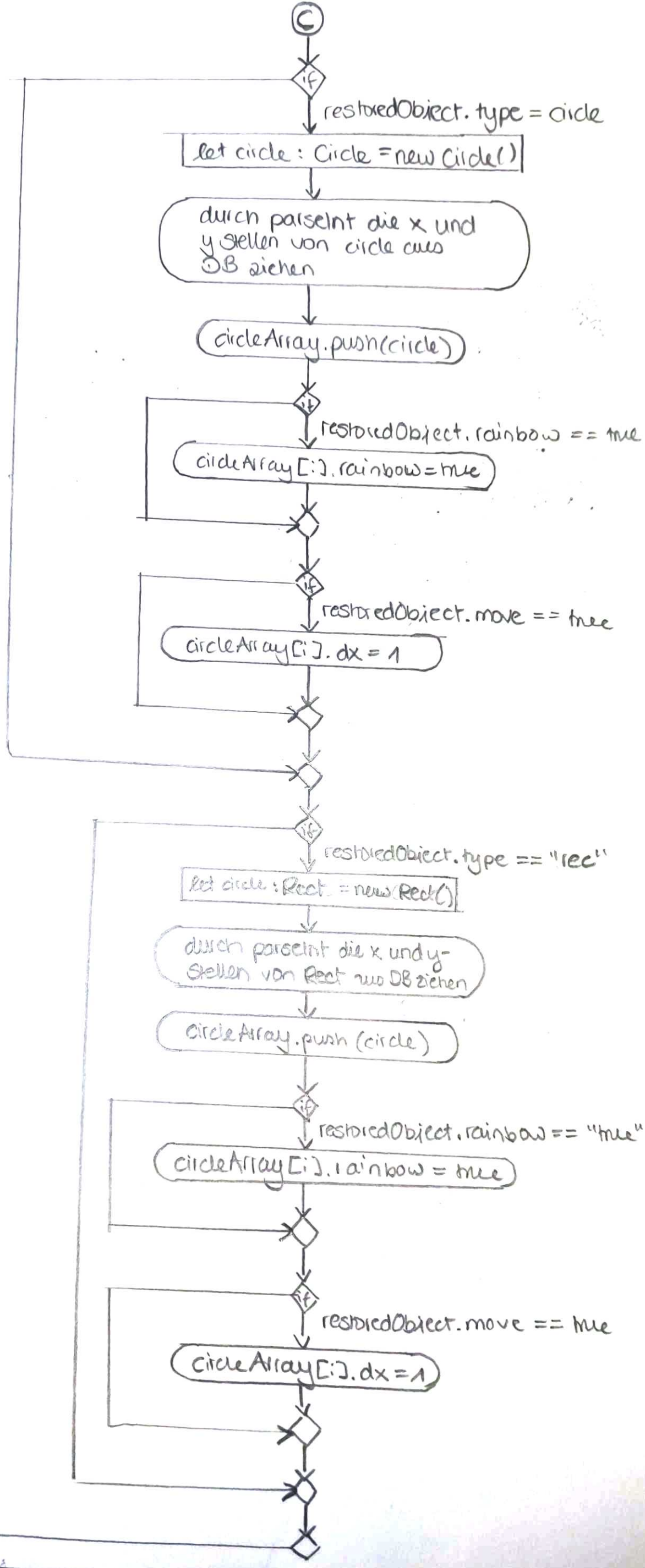
i< yPos.length

let restoredObject : Object=

i++

x: xpos[i]
y: ypos[i]
type: type[i]
rainbow: rainbow[i]
move: move[i]

C


restoredObject.type = circle
let circle : Circle = new Circle()
durch parseInt die x und y stellen von circle aus OB ziehen
circleArray.push(circle)
if
restoredObject.rainbow == true
circleArray[i].rainbow = true
if
restoredObject.move == true
circleArray[i].dx = 1
if
restoredObject.type == "rec"
let circle : Rect = new Rect()
durch parseInt die x und y-Stellen von Rect wo OB ziehen
circleArray.push(circle)
if
restoredObject.rainbow == "true"
circleArray[i].rainbow = true
if
restoredObject.move == true
circleArray[i].dx = 1

DBClient - veränderte Funktionen

```
export interface Object {
    x: string;
    y: string;
    type: string;
    rainbow: string;
    move: string;   }
```

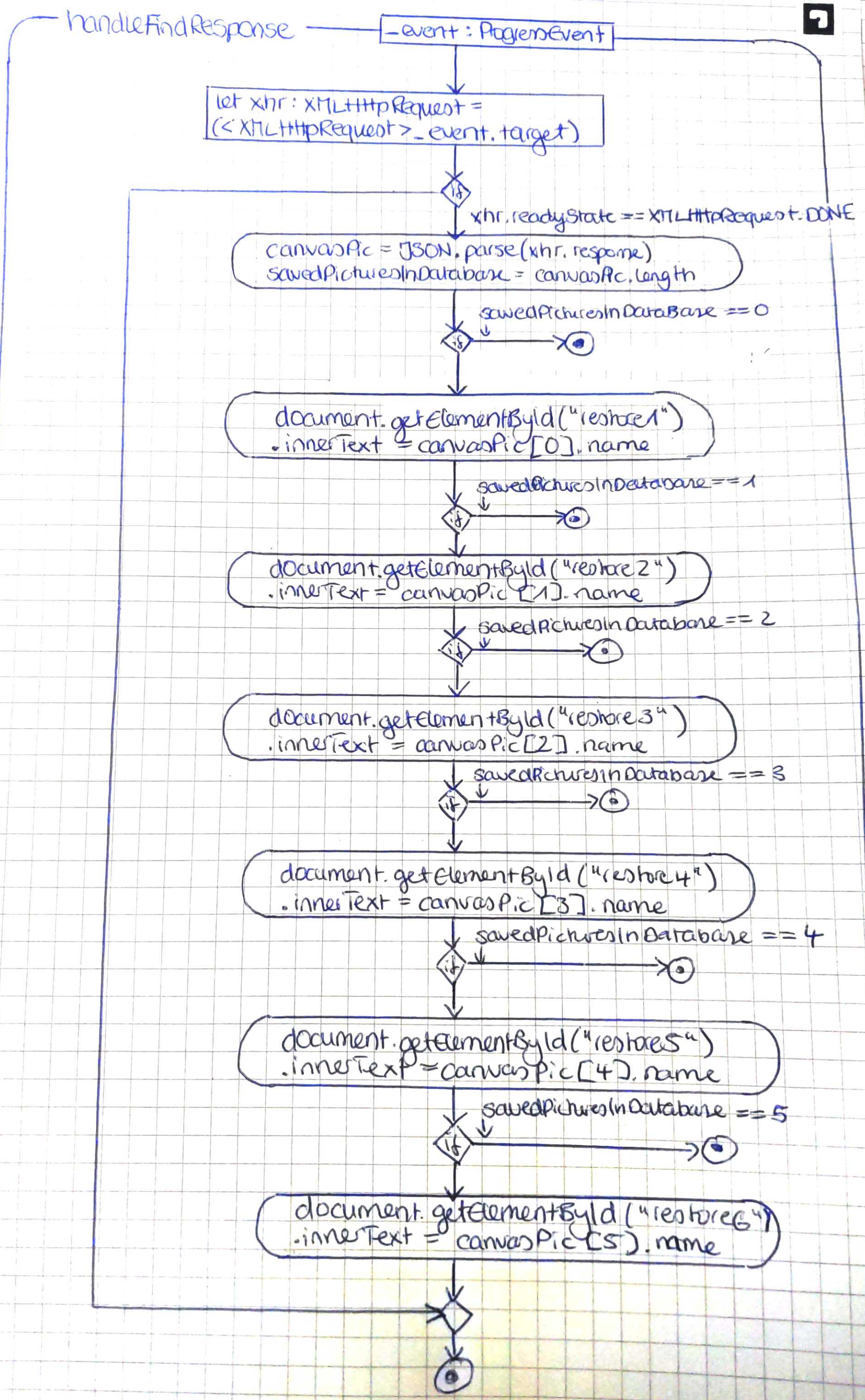
Globale Variabeln:

```
export let canvasPic: CanvasElement[];
let serverAdress: string = "https://koelliefe.herokuapp.com/";
let savedPicturesInDatabase: number;
```

export — insert(_name: string)

- let query: string = "command=insert"
- query += "&name=" + _name
- query += "&backgroundColor=" + backgroundColor
- query += "&canvasWidth=" + canvas.width
- i: number = 0
- for (i < circleArray.length) → (i++)
  - wandle alle numbers oder booleans des Interfaces in strings mit .toString() um. → Daten definiert
  - query += "&X=" + circle.x + "&Y=" + circle.y + "&type=" + circle.type + "&rainbow=" + circle.rainbow + "&move=" + circle.move
- sendRequest(query, handleInsertResponse ↑)

handleFindResponse



```
interface AssoStringString {
    [key: string]: string }
```

Interface gesamte Canvas

```
interface CanvasElement {
  name: string;
  backgroundColor: string;
  x: string;
  y: string;
  move: string;
  width: string;
  type: string;
  rainbow: string;          }
```

Domänenübergreifendes Aktivitätsdiagramm