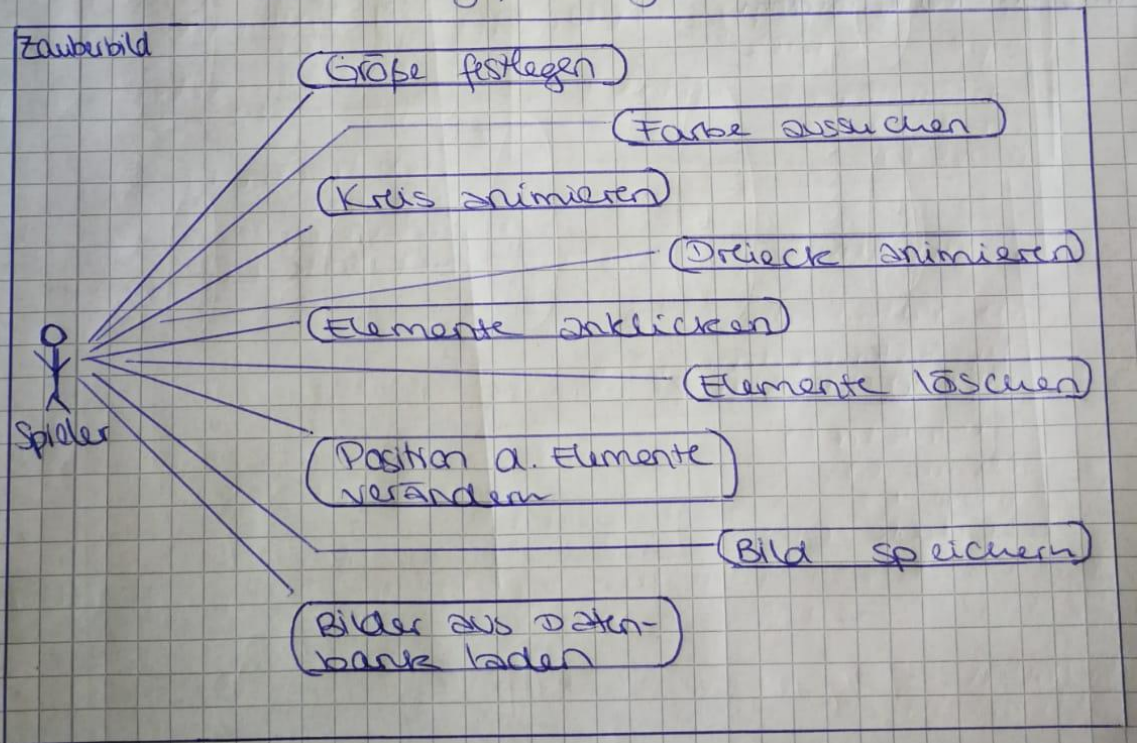
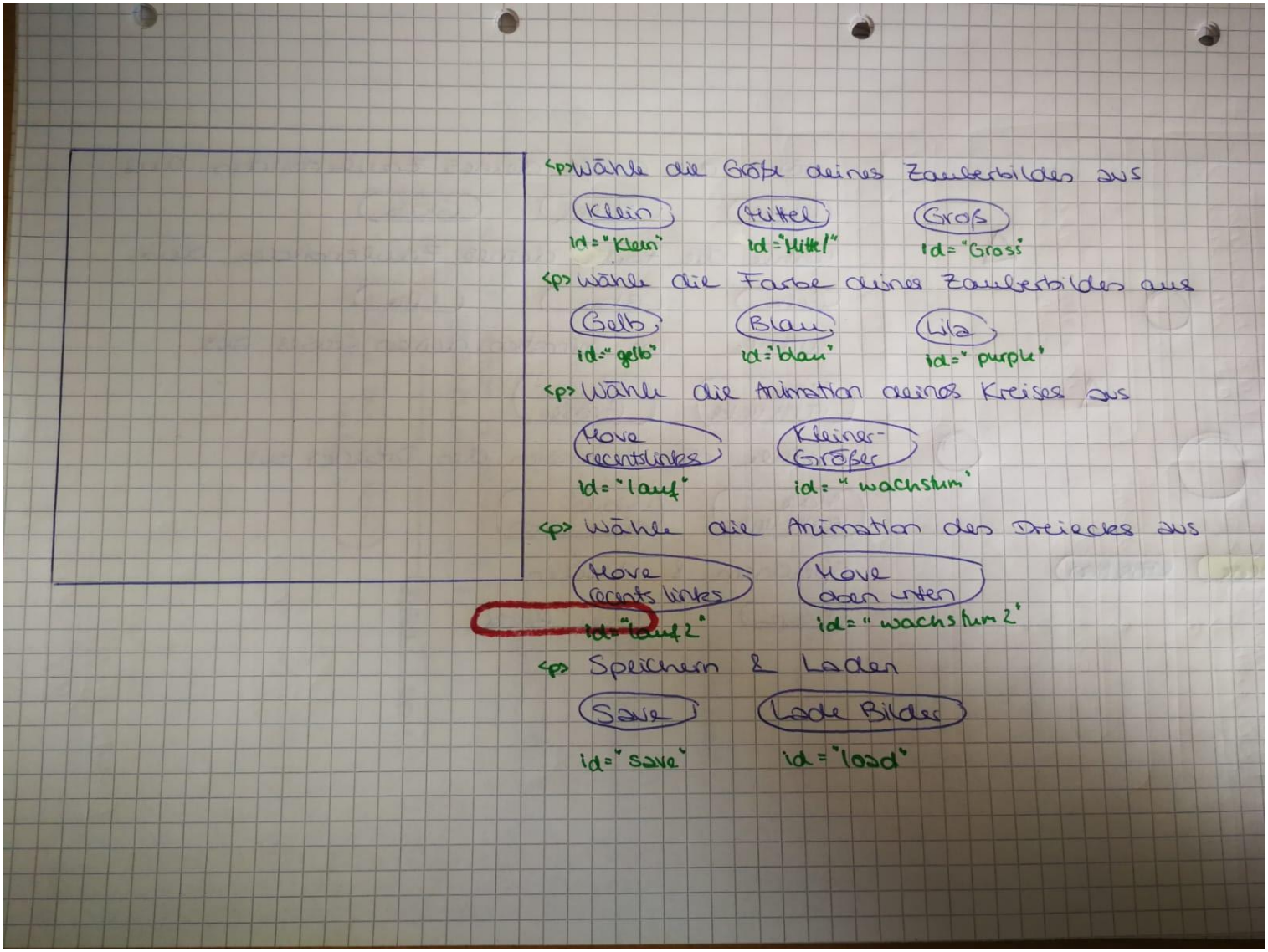
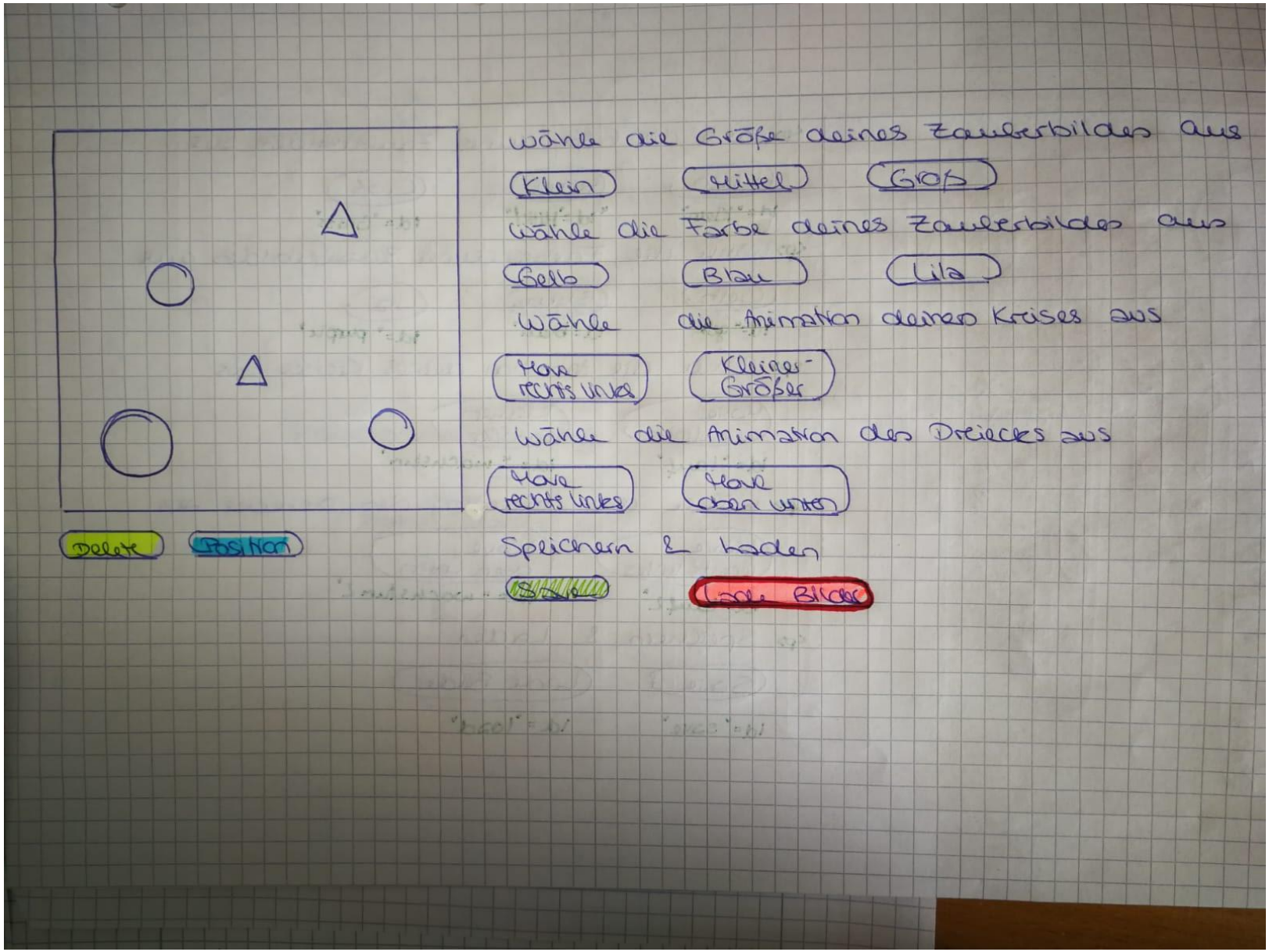
Ablaufdiagramm

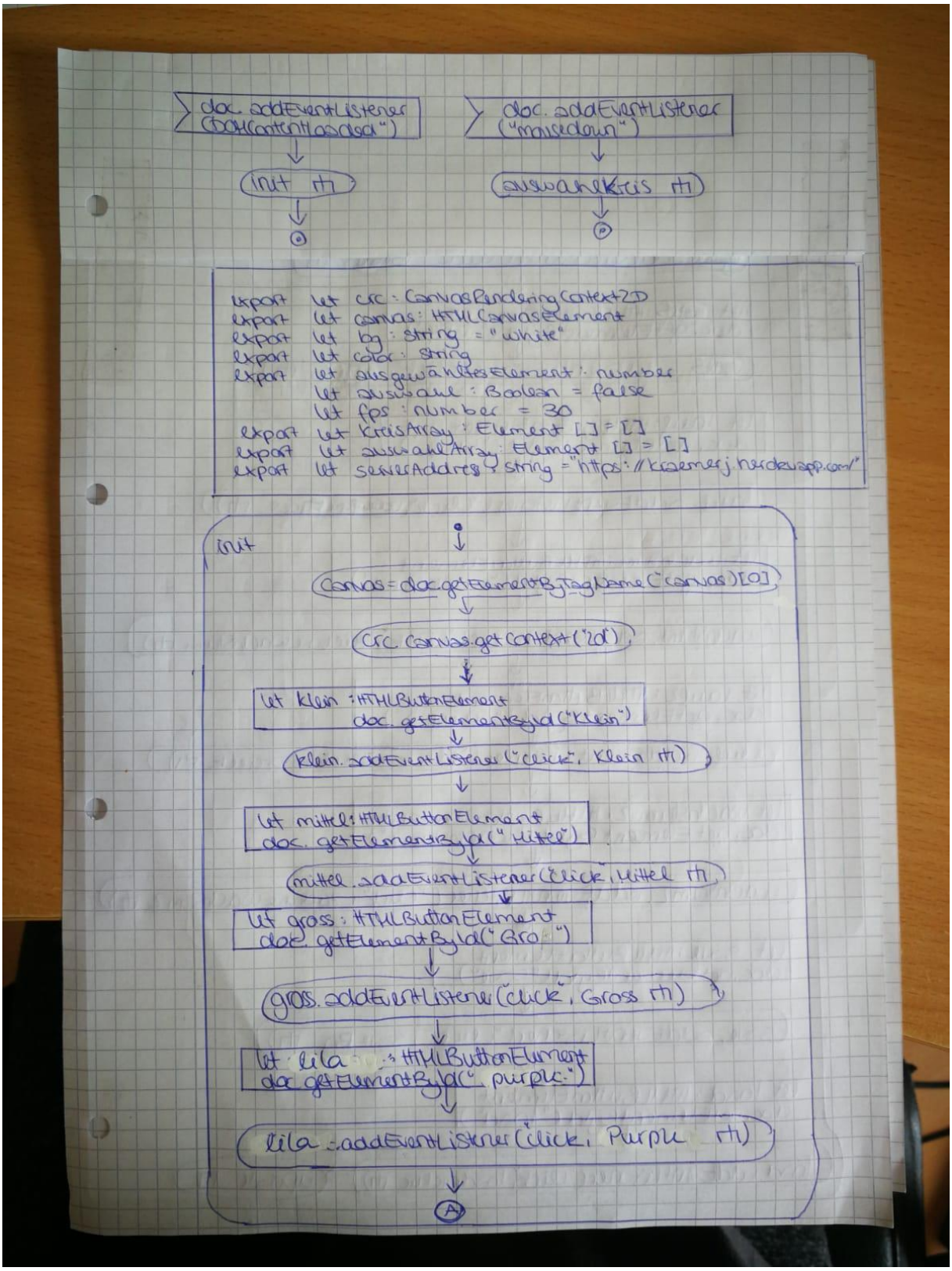
① Größe festlegen
② Farbe aussuchen
③ Animation für Kreis auswählen
④ Animation für Dreieck auswählen
⑤ Elemente anclicken:
⑥ Löschen / Position verändern
⑦ Bild speichern
⑧ gespeicherte Bilder aus Datenbank wieder herstellen und weiterbearbeiten

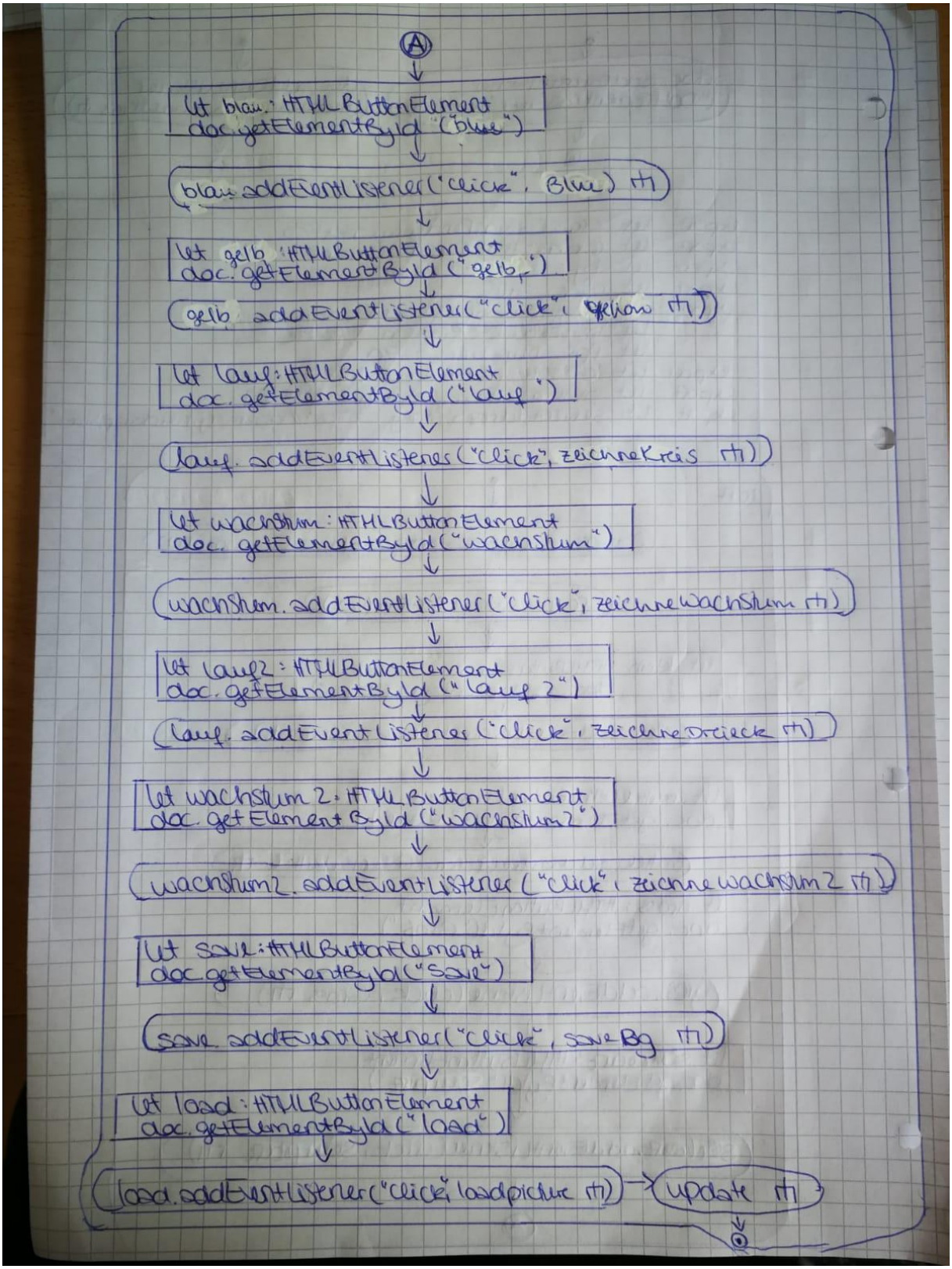
Anwendungsfalldiagramm

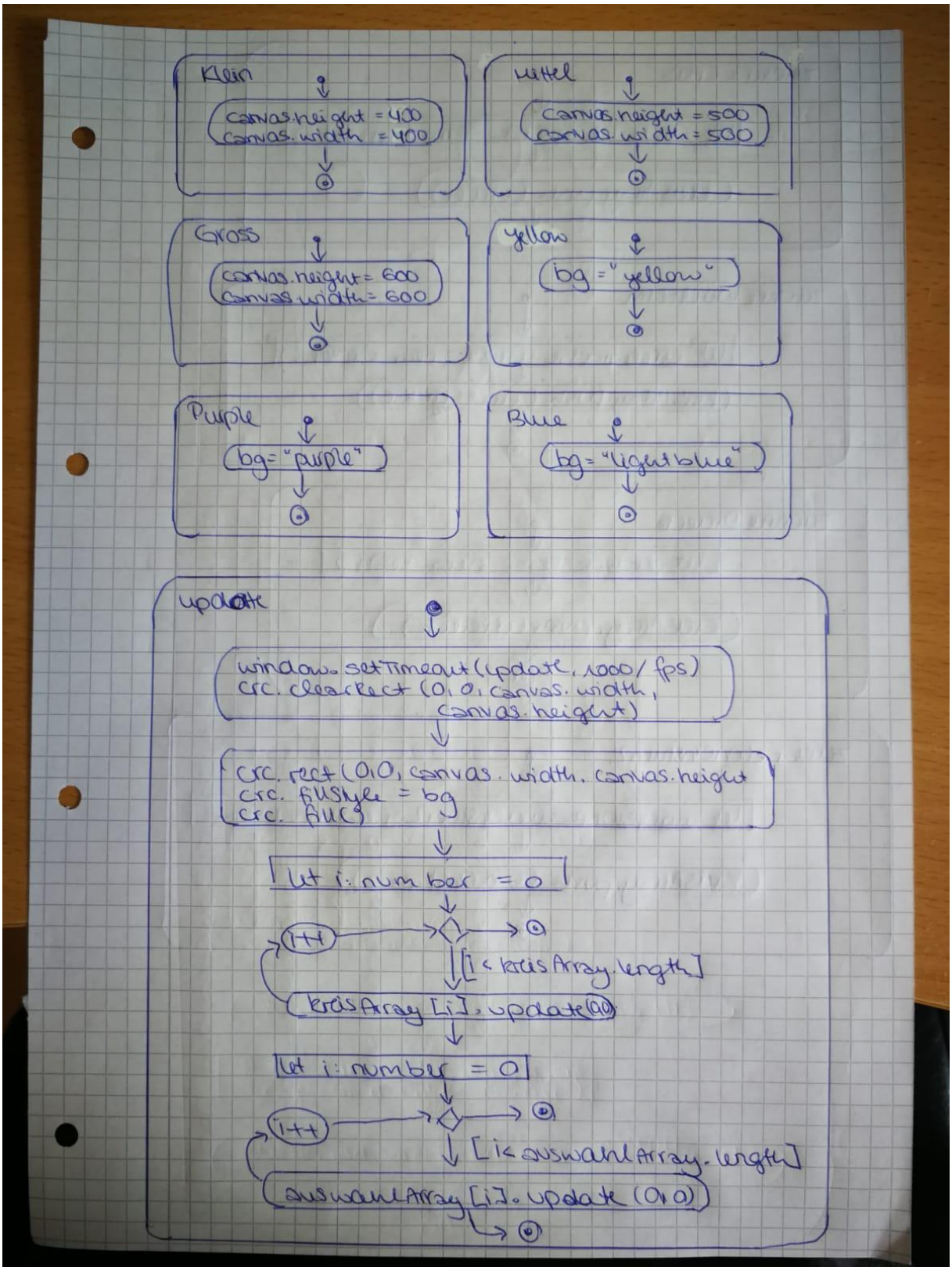


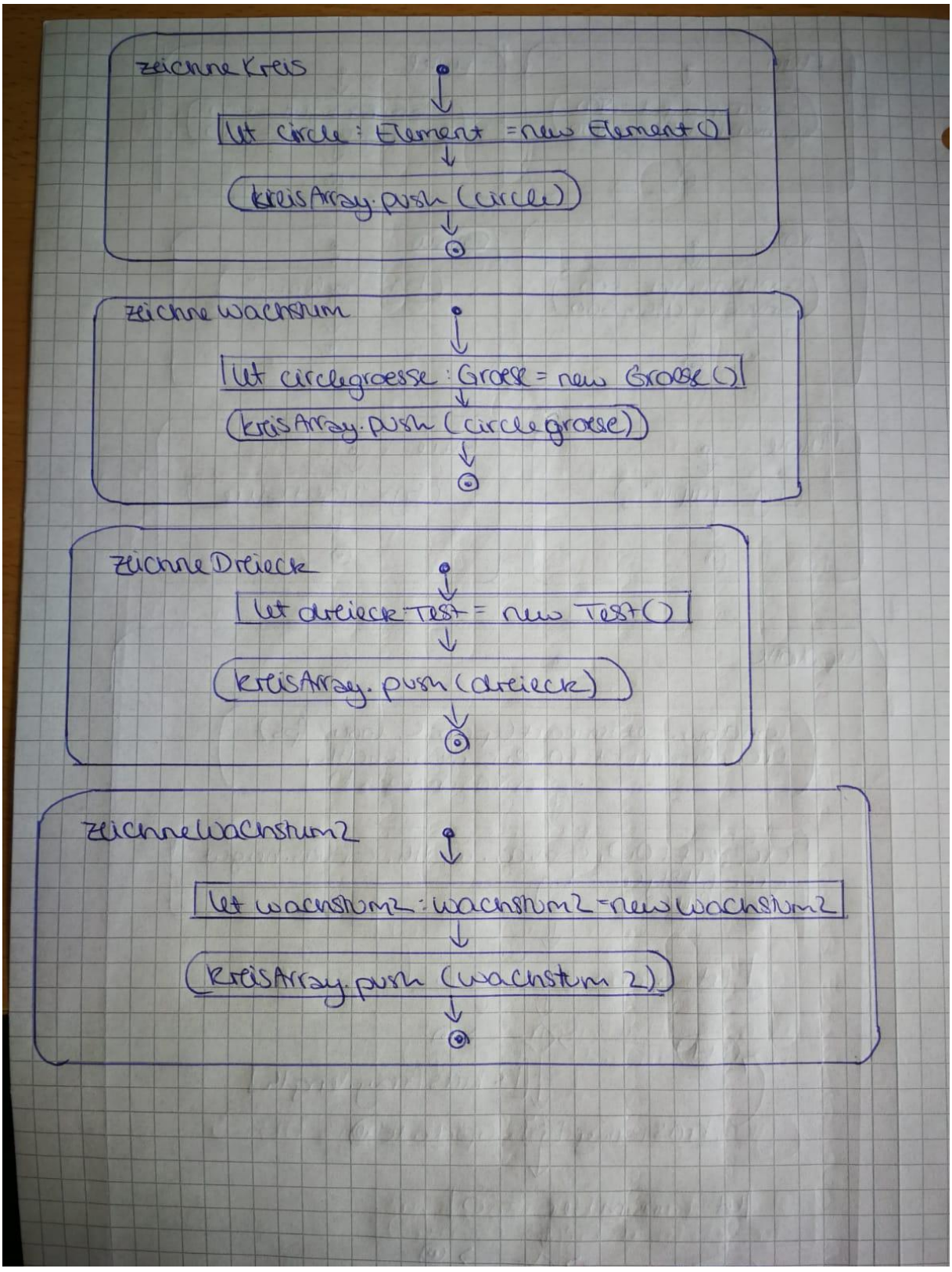

<p>Wähle die Größe deines Zauberbildes aus
Klein
id="klein"
Mittel
id="Mittel"
Groß
id="Gross"
<p> Wähle die Farbe deines Zauberbildes aus
Gelb
id="gelb"
Blau
id="blau"
Lila
id="purple"
<p> Wähle die Animation deines Kreises aus
Move
rechts links
id="lauf"
Kleiner-
Größer
id="wachstum"
<p> Wähle die Animation des Dreieckes aus
Move
rechts links
id="lauf2"
Move
oben unten
id="wachstum2"
<p> Speichern & Laden
Save
id="save"
Lade Bilder
id="load"

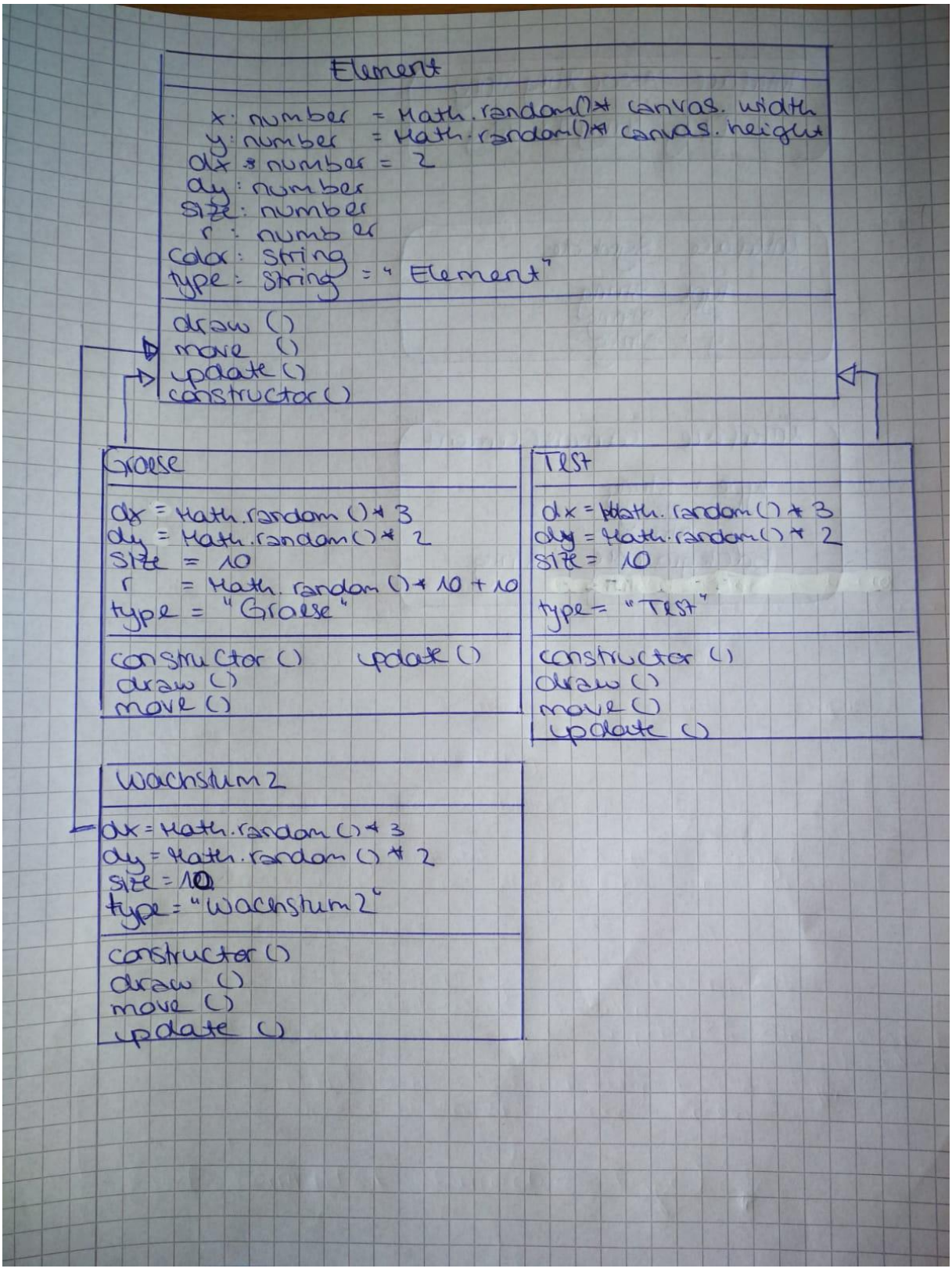

wähle die Größe deines Zauberbildes aus
Klein
Mittel
Groß
wähle die Farbe deines Zauberbildes aus
Gelb
Blau
Lila
wähle die Animation deines Kreises aus
Move rechts links
Kleiner-Größer
wähle die Animation des Dreiecks aus
Move rechts links
Move oben unten
Delete
Position
Speichern & laden
Laden

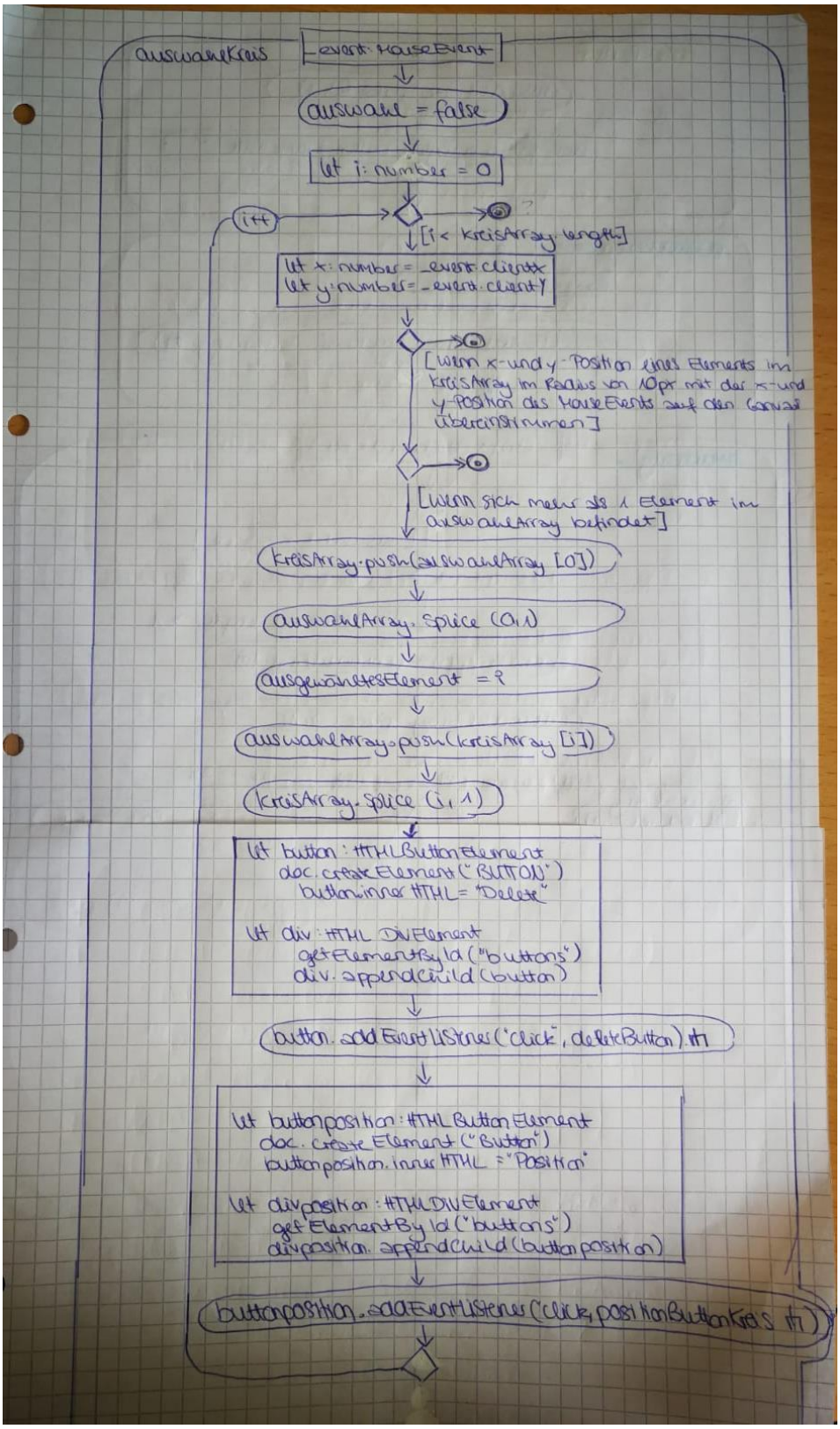

doc.addEventListener ("DOMContentLoaded")
init ()
doc.addEventListener ("mousedown")
auswahlKreis ()
export let crc: CanvasRenderingContext2D
export let canvas: HTMLCanvasElement
export let bg: string = "white"
export let color: string
export let ausgewähltesElement: number
let auswahl: Boolean = false
let fps: number = 30
export let KreisArray: Element [] = []
export let auswahlArray: Element [] = []
export let serverAddress: string = "https://kraemerj.herokuapp.com/"
init
canvas = doc.getElementsByTagName("canvas")[0]
crc = canvas.getContext("2d")
let klein: HTMLButtonElement
doc.getElementById("klein")
klein.addEventListener("click", klein ())
let mittel: HTMLButtonElement
doc.getElementById("mittel")
mittel.addEventListener("click", mittel ())
let gross: HTMLButtonElement
doc.getElementById("gross")
gross.addEventListener("click", gross ())
let lila: HTMLButtonElement
doc.getElementById("purple")
lila.addEventListener("click", purple ())
A

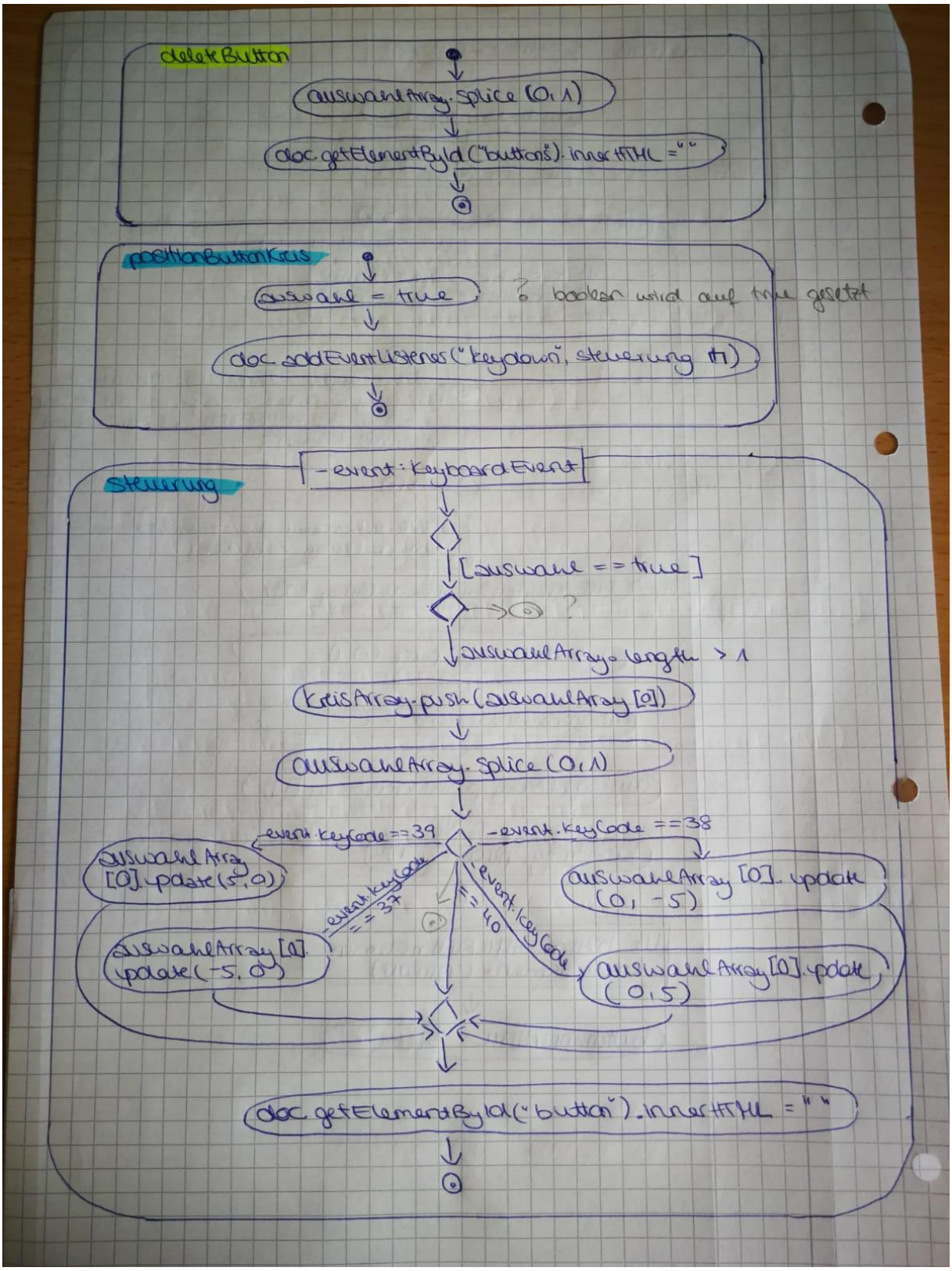

A
let blau: HTMLButtonElement
doc.getElementById("blue")
blau.addEventListener("click", Blue) ??
let gelb: HTMLButtonElement
doc.getElementById("gelb")
gelb.addEventListener("click", gelben ??)
let lauf: HTMLButtonElement
doc.getElementById("lauf")
lauf.addEventListener("click", zeichneKreis ??)
let wachstum: HTMLButtonElement
doc.getElementById("wachstum")
wachstum.addEventListener("click", zeichneWachstum ??)
let lauf2: HTMLButtonElement
doc.getElementById("lauf 2")
lauf.addEventListener("click", zeichneDreieck ??)
let wachstum2: HTMLButtonElement
doc.getElementById("wachstum2")
wachstum2.addEventListener("click", zeichneWachstum2 ??)
let save: HTMLButtonElement
doc.getElementById("save")
save.addEventListener("click", saveBg ??)
let load: HTMLButtonElement
doc.getElementById("load")
load.addEventListener("click", loadpicture ??)
update ??

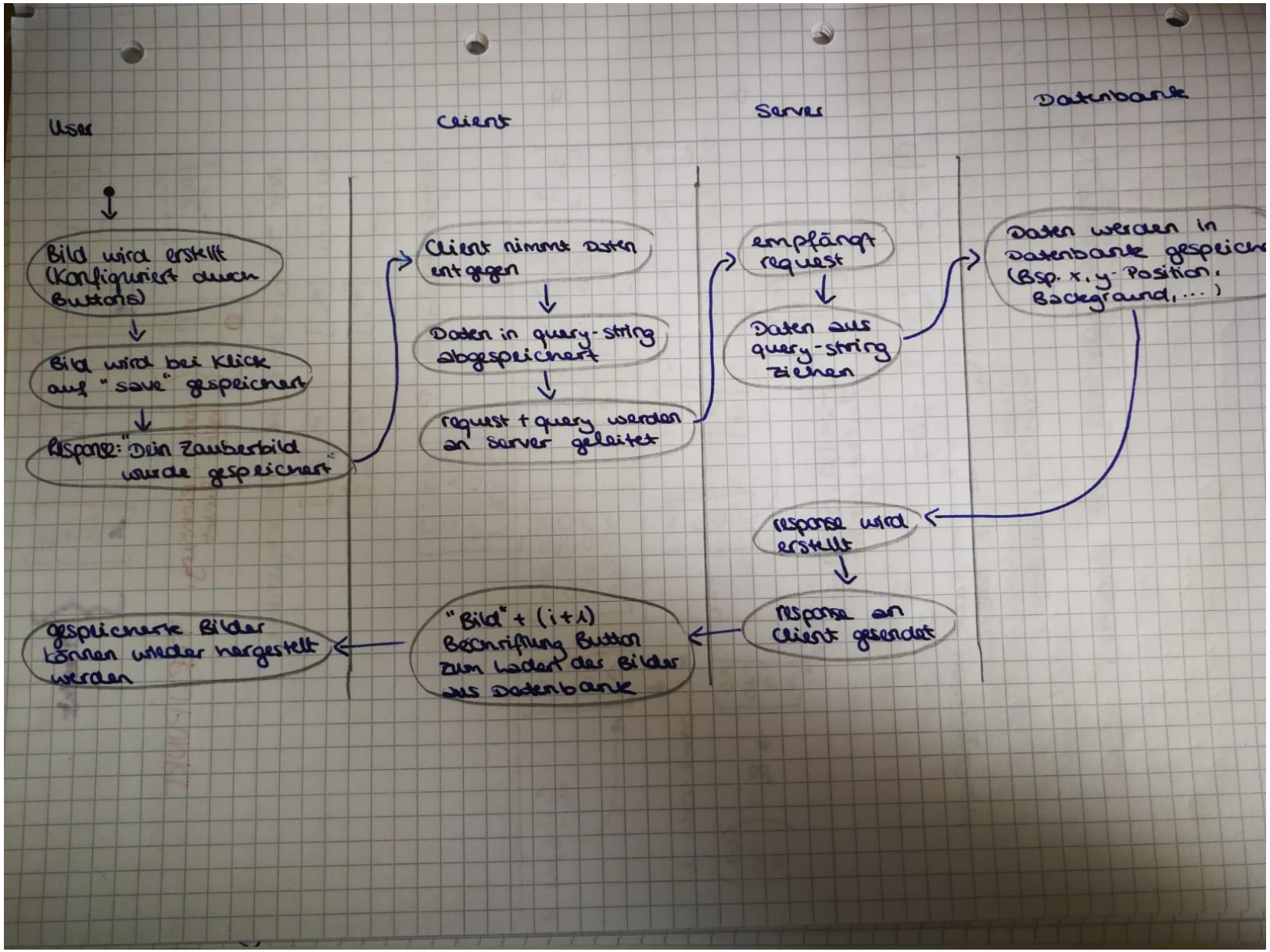

Klein
canvas.height = 400
canvas.width = 400
Mittel
canvas.height = 500
canvas.width = 500
Gross
canvas.height = 600
canvas.width = 600
yellow
bg = "yellow"
Purple
bg = "purple"
Blue
bg = "lightblue"
update
window.setTimeout(update, 1000/fps)
crc.clearRect(0,0, canvas.width, canvas.height)
crc.rect(0,0, canvas.width, canvas.height
crc.fillStyle = bg
crc.fill()
let i: number = 0
i++
[i < kreisArray.length]
kreisArray[i].update()
let i: number = 0
i++
[i < auswahlArray.length]
auswahlArray[i].update(0,0)

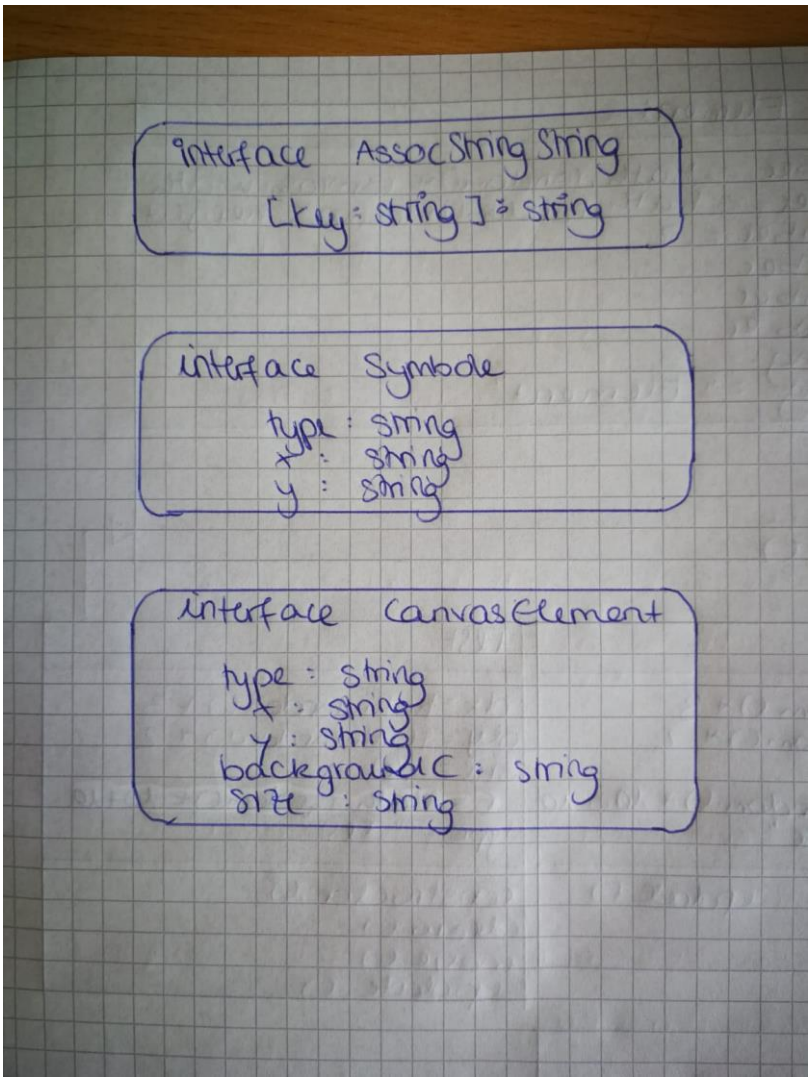

zeichne Kreis
let circle : Element = new Element()
kreisArray.push(circle)
zeichne wachstum
let circlegroesse : Groesse = new Groesse()
kreisArray.push(circlegroesse)
zeichne Dreieck
let dreieck: TEST = new TEST()
kreisArray.push(dreieck)
zeichnewachstum2
let wachstum2 : wachstum2 = new wachstum2
kreisArray.push(wachstum2)

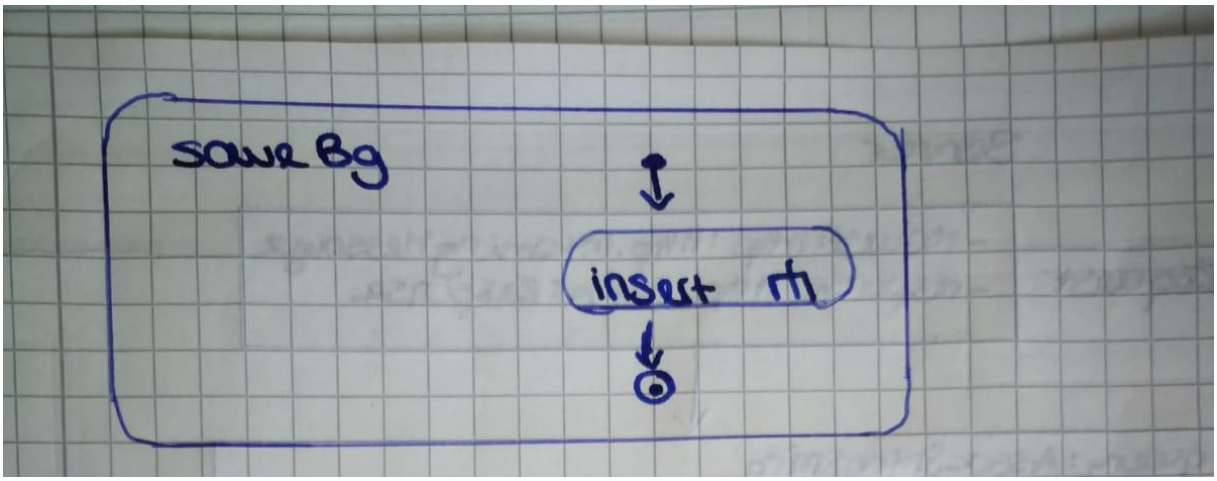

Element
x: number = Math.random() * canvas.width
y: number = Math.random() * canvas.height
dx: number = 2
dy: number
size: number
r: number
color: string
type: string = "Element"
draw ()
move ()
update ()
constructor ()
Grasse
dx = Math.random () * 3
dy = Math.random () * 2
size = 10
r = Math.random () * 10 + 10
type = "Grasse"
constructor ()
update ()
draw ()
move ()
Test
dx = Math.random () * 3
dy = Math.random () * 2
size = 10
type = "Test"
constructor ()
draw ()
move ()
update ()
Wachstum 2
dx = Math.random () * 3
dy = Math.random () * 2
size = 10
type = "Wachstum 2"
constructor ()
draw ()
move ()
update ()


auswahlKreis
event: MouseEvent
auswahl = false
let i: number = 0
i++
[i < kreisArray.length]
let x: number = _event.clientX
let y: number = _event.clientY
[wenn x- und y-Position eines Elements im kreisArray im Radius von 10px mit der x- und y-Position des MouseEvents auf dem Canvas übereinstimmen]
[wenn sich mehr als 1 Element im auswahlArray befindet]
kreisArray.push(auswahlArray[0])
auswahlArray.splice(0,1)
ausgewähltesElement = ?
auswahlArray.push(kreisArray[i])
kreisArray.splice(i, 1)
let button: HTMLButtonElement
doc.createElement("BUTTON")
button.innerHTML = "Delete"
let div: HTML DivElement
getElementById("buttons")
div.appendChild(button)
button.addEventListener("click", deleteButton)
let buttonposition: HTMLButtonElement
doc.createElement("Button")
buttonposition.innerHTML = "Position"
let divposition: HTMLDIVElement
getElementById("buttons")
divposition.appendChild(buttonposition)
buttonposition.addEventListener("click", positionButtonKreis)


deleteButton
auswahlArray.splice(0,1)
doc.getElementById("buttons").innerHTML = ""
positionButtonKreis
auswahl = true
? boolean wird auf true gesetzt
doc.addEventListener("keydown", steuerung fn)
steuerung
-event: KeyboardEvent
[auswahl == true]
?
auswahlArray.length > 1
kreisArray.push(auswahlArray[0])
auswahlArray.splice(0,1)
event.keyCode == 39
-event.keyCode == 38
auswahlArray [0].update (5,0)
auswahlArray [0].update (0, -5)
event.keyCode == 37
event.keyCode == 40
auswahlArray [0].update(-5,0)
auswahlArray [0].update (0,5)
doc.getElementById("button").innerHTML = ""


User
Client
Server
Datenbank
Bild wird erstellt (Konfiguriert durch Buttons)
Bild wird bei Klick auf "save" gespeichert
Response: "Dein Zauberbild wurde gespeichert"
Client nimmt Daten entgegen
Daten in query-string abgespeichert
request + query werden an Server geleitet
empfängt request
Daten aus query-string ziehen
Daten werden in Datenbank gespeich
(Bsp. x, y-Position, Background, ...)
response wird erstellt
response an Client gesendet
"Bild" + (i+1) Beschriftung Button zum Laden der Bilder aus Datenbank
gespeicherte Bilder können wieder hergestellt werden


interface AssocString String
[key: string]: string
interface Symbole
type: string
x: string
y: string
interface CanvasElement
type: string
x: string
y: string
backgroundC: string
size: string


saveBg
insert

Client

interface Symbole
type: string
x: string
y: string

interface CanvasElement
type: string
x: string
y: string
backgroundColor: string
size: string

let serverAddress: string = "https://kreismaj.herokuapp.com/"
export let globalArray: CanvasElement[]

insert

let query: string = "command=insert"

query += "&bg=" + bg
query += "&canvaswidth=" + canvaswidth

let i: number = 0

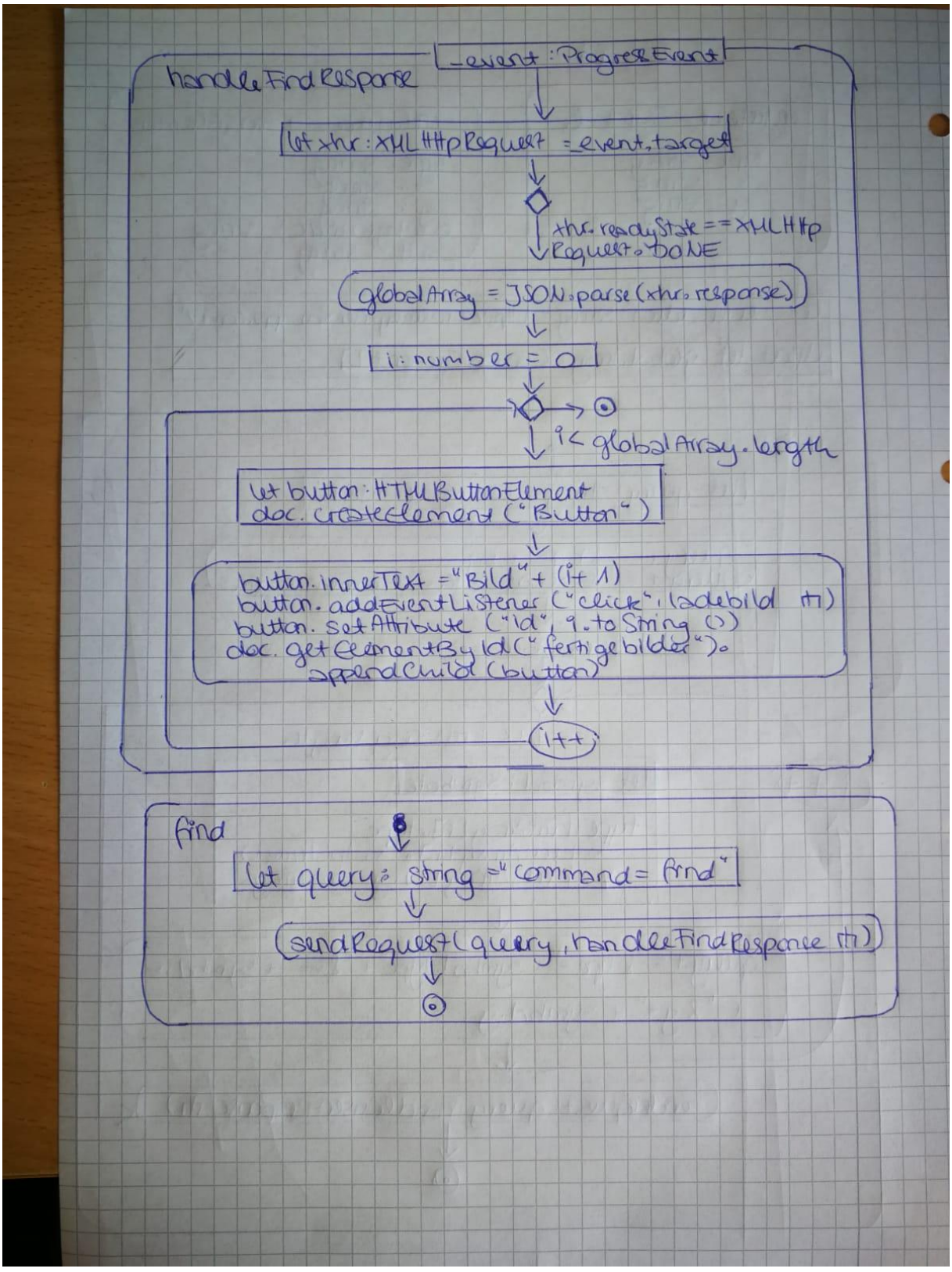
i < kreisArray.length

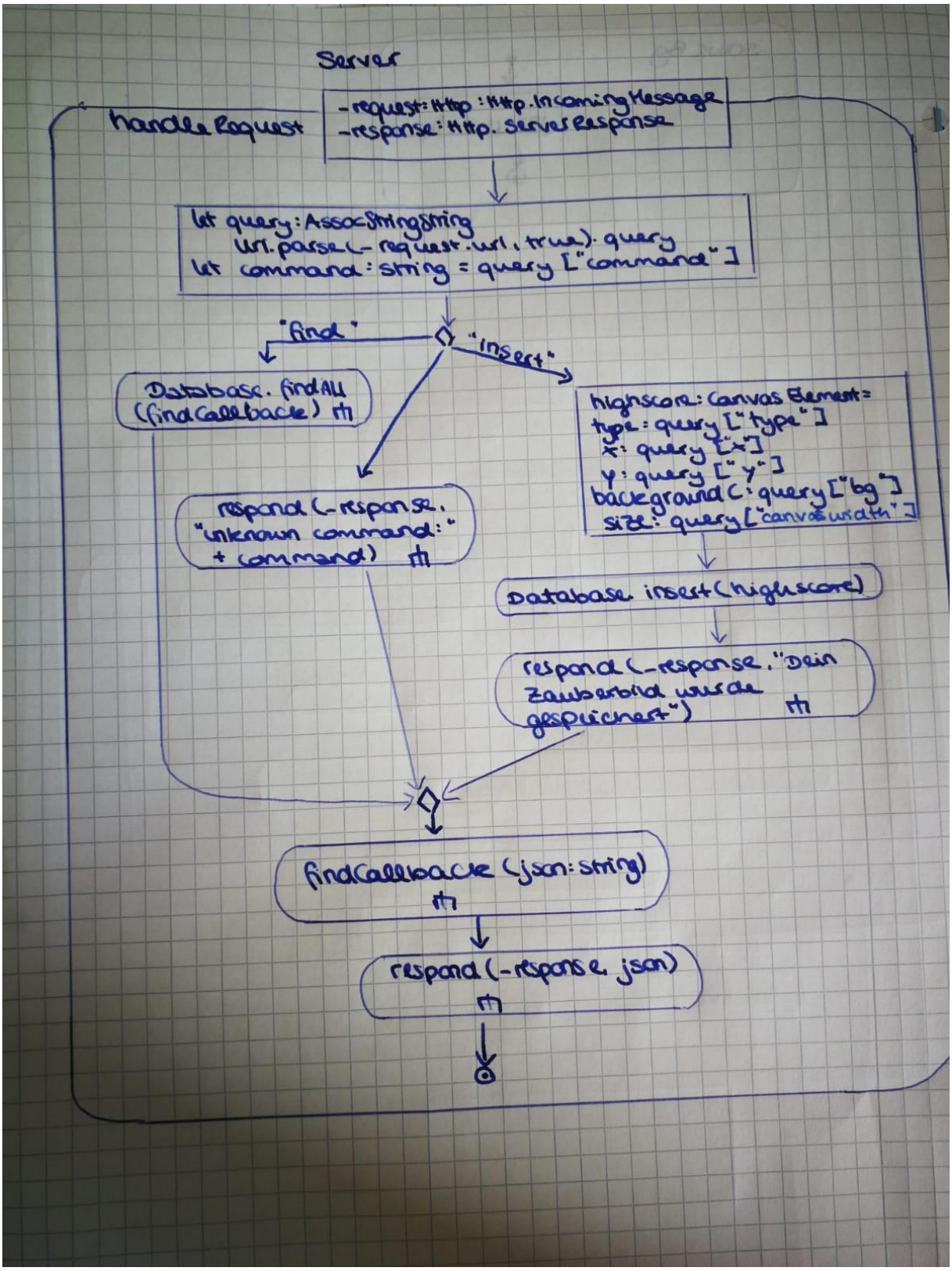
let symbol: Symbole
type: kreisArray[i].type
x: kreisArray[i].x.toString()
y: kreisArray[i].y.toString()

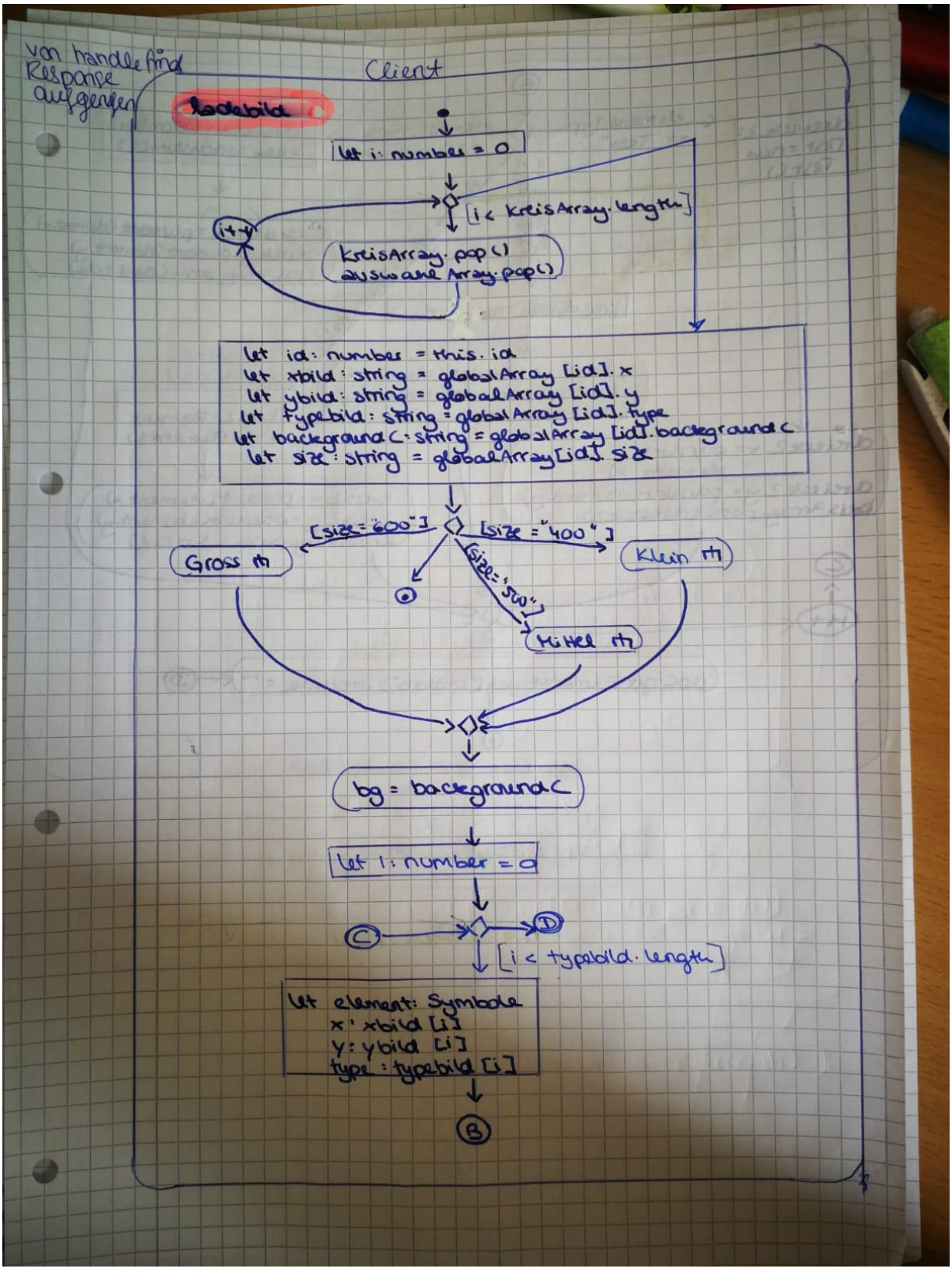
query += "&type=" + symbol.type + "&x=" + symbol.x + "&y=" + symbol.y

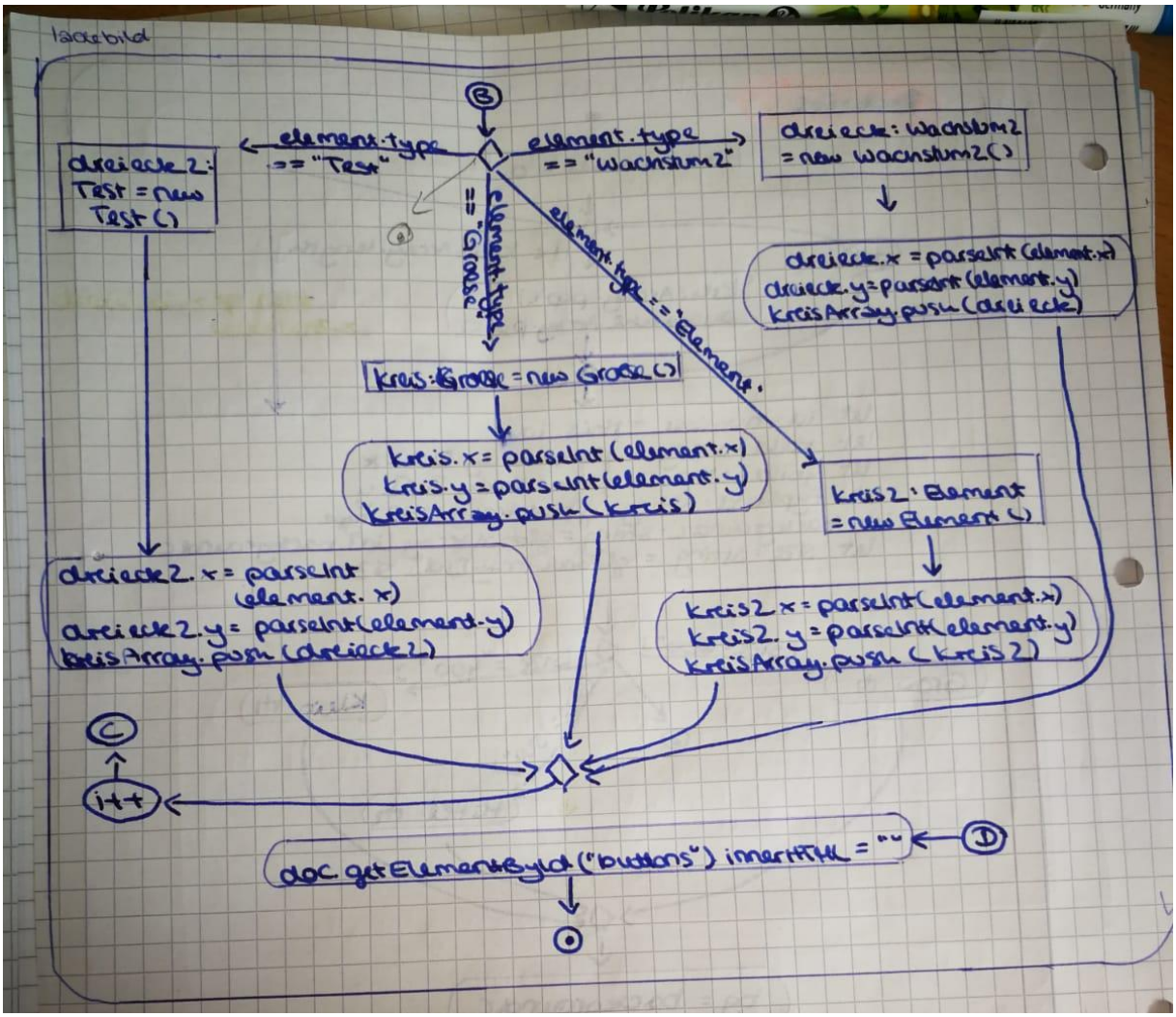
i++

sendRequest(query, handleInsertResponse)


handleFindResponse
_event : ProgressEvent
let xhr : XMLHttpRequest = _event.target
xhr.readyState == XMLHttpRequest.DONE
globalArray = JSON.parse(xhr.response)
i: number = 0
i < globalArray.length
let button: HTMLButtonElement
doc.createElement("Button")
button.innerText = "Bild" + (i+1)
button.addEventListener("click", ladebild th)
button.setAttribute("id", i.toString())
doc.getElementById("fertigebilder").appendChild(button)
i++
find
let query: string = "command=find"
sendRequest(query, handleFindResponse th)


Server
handleRequest
-request: Http: Http.IncomingMessage
-response: Http. ServerResponse
let query: AssocStringString
Url.parse(-request.url, true).query
let command: string = query ["command"]
"find"
"insert"
Database. findAll
(findCallback)
respond(-response,
"unknown command:"
+ command)
highscore: CanvasElement=
type: query ["type"]
x: query ["x"]
y: query ["y"]
background: query ["bg"]
size: query ["canvasWidth"]
Database. insert(highscore)
respond(-response, "Dein
Zauberbild wurde
gespeichert")
findCallback(json: string)
respond(-response, json)


von handle find Response aufgerufen
Client
ladebild
let i: number = 0
i++
[i < kreisArray.length]
kreisArray.pop()
allSwahlArray.pop()
let id: number = this.id
let xbild: string = globalArray [id].x
let ybild: string = globalArray [id].y
let typebild: string = globalArray [id].type
let backgroundC: string = globalArray [id].backgroundC
let size: string = globalArray[id].size
[size="600"]
[size="400"]
[size="500"]
Gross m
Klein m
Mittel m
bg = backgroundC
let i: number = 0
[i < typebild.length]
let element: Symbole
x: xbild [i]
y: ybild [i]
type: typebild [i]
C
D
B


Tafelbild
B
element.type == "Test"
element.type == "Wachstum2"
element.type == "Gross"
element.type == "Element"
dreieck2: Test = new Test()
dreieck: Wachstum2 = new Wachstum2()
dreieck.x = parseInt(element.x)
dreieck.y = parseInt(element.y)
kreisArray.push(dreieck)
kreis: Gross = new Gross()
kreis.x = parseInt(element.x)
kreis.y = parseInt(element.y)
kreisArray.push(kreis)
kreis2: Element = new Element()
dreieck2.x = parseInt (element.x)
dreieck2.y = parseInt(element.y)
kreisArray.push(dreieck2)
kreis2.x = parseInt(element.x)
kreis2.y = parseInt(element.y)
kreisArray.push(kreis2)
C
i++
doc.getElementById("buttons").innerHTML = ""
D