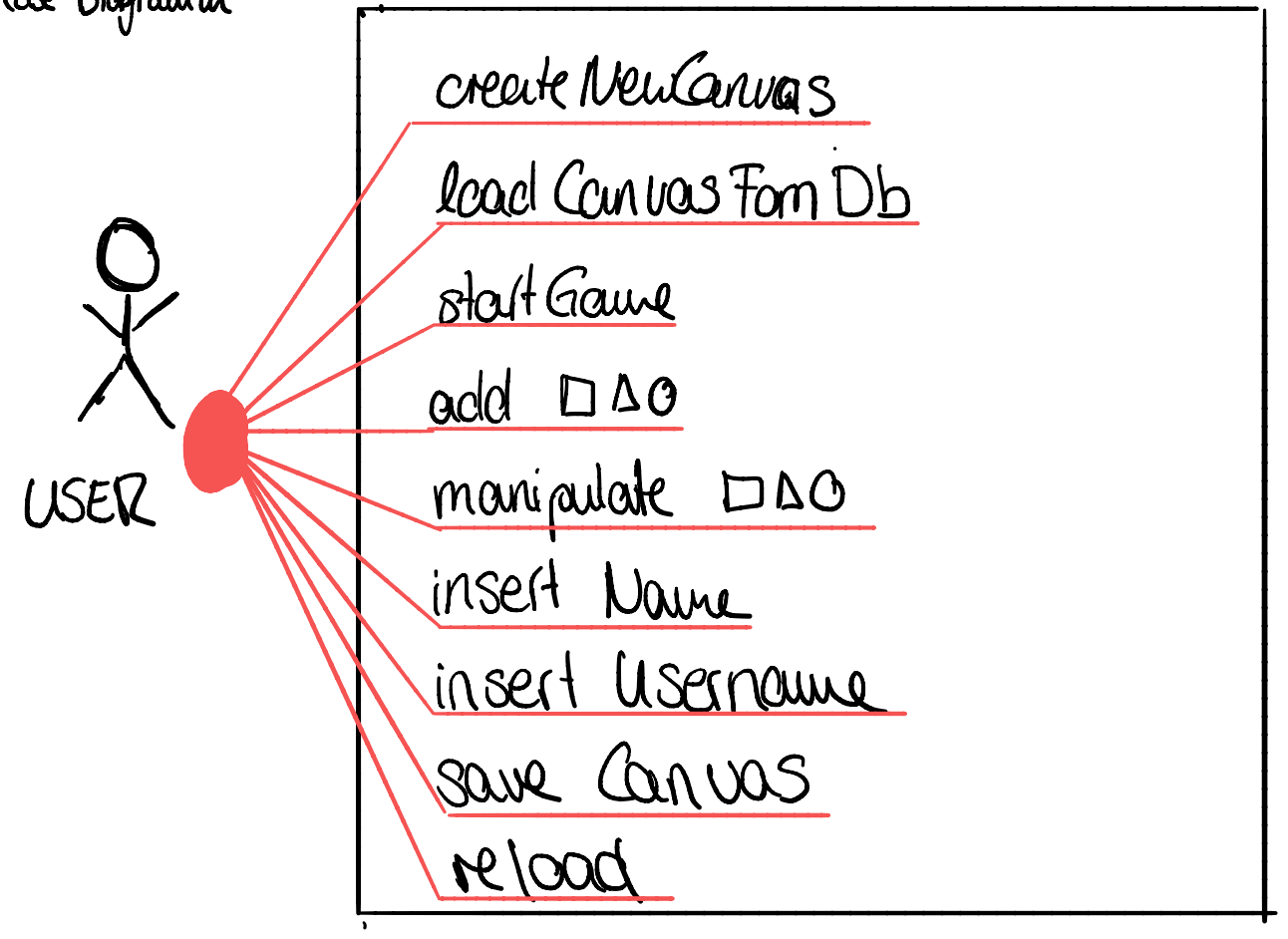
EIA2 Endabgabe

Konzept

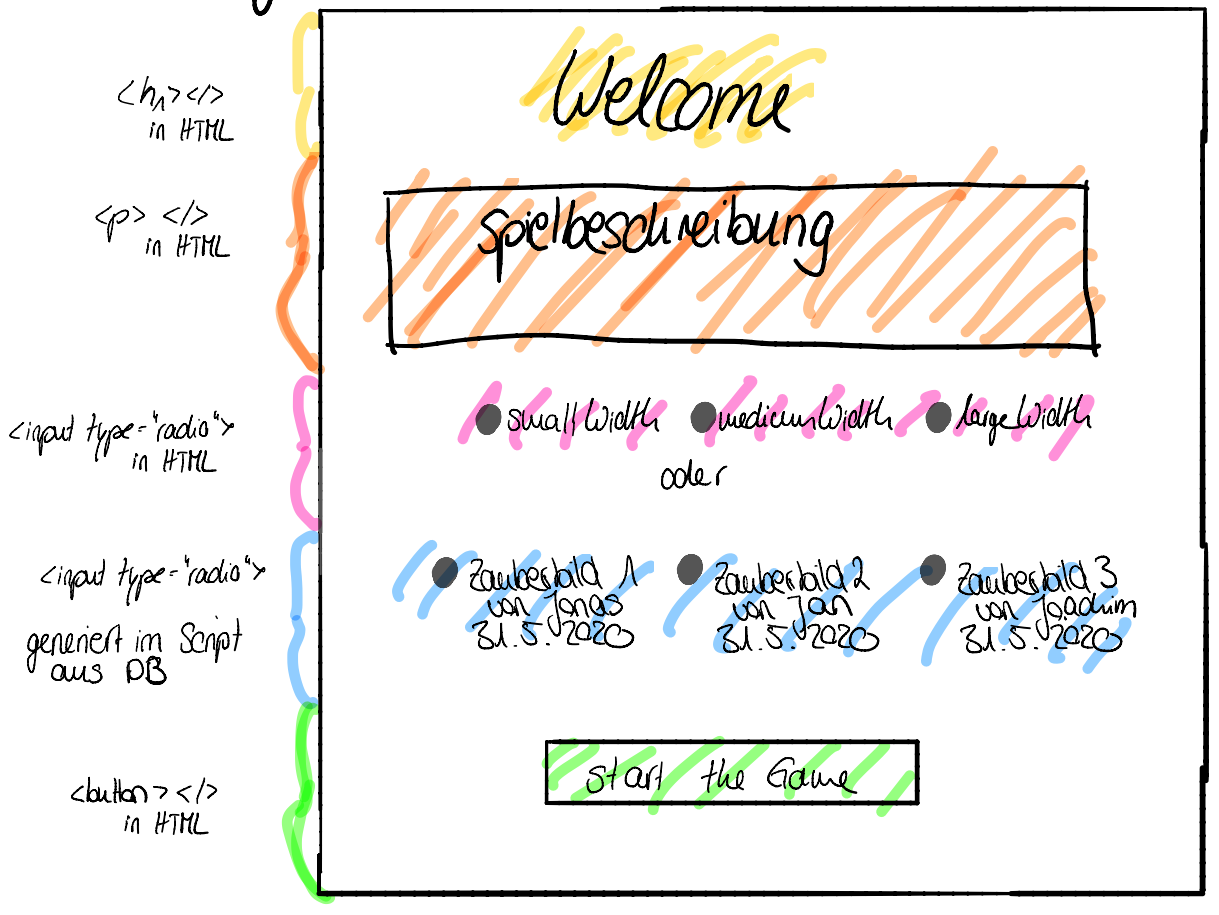
von Riccardo Piccirillo
262542

UseCase Diagramm
Skizze Interface
Client-Server-Datenbank
Klassendiagramm
Aktivitätsdiagramme

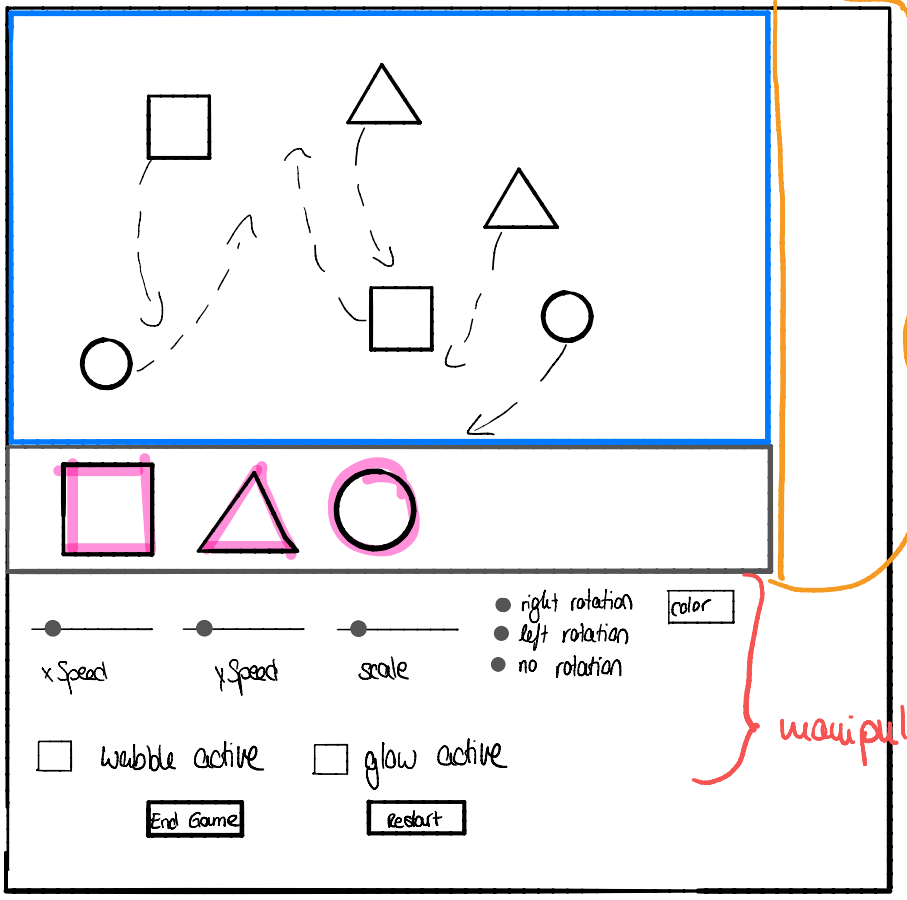
Use Case Diagramm



Skizze start Page



Skizze gamePage




Bereich, in welchen △ □ ○ gezogen werden können ⇝ dragEnd

Vorschauobjekte [MenuObjects[]] ⇝ dragStart

<canvas>

manipulationArea mit <input> type=range
type=radio
type=checkbox
type=color

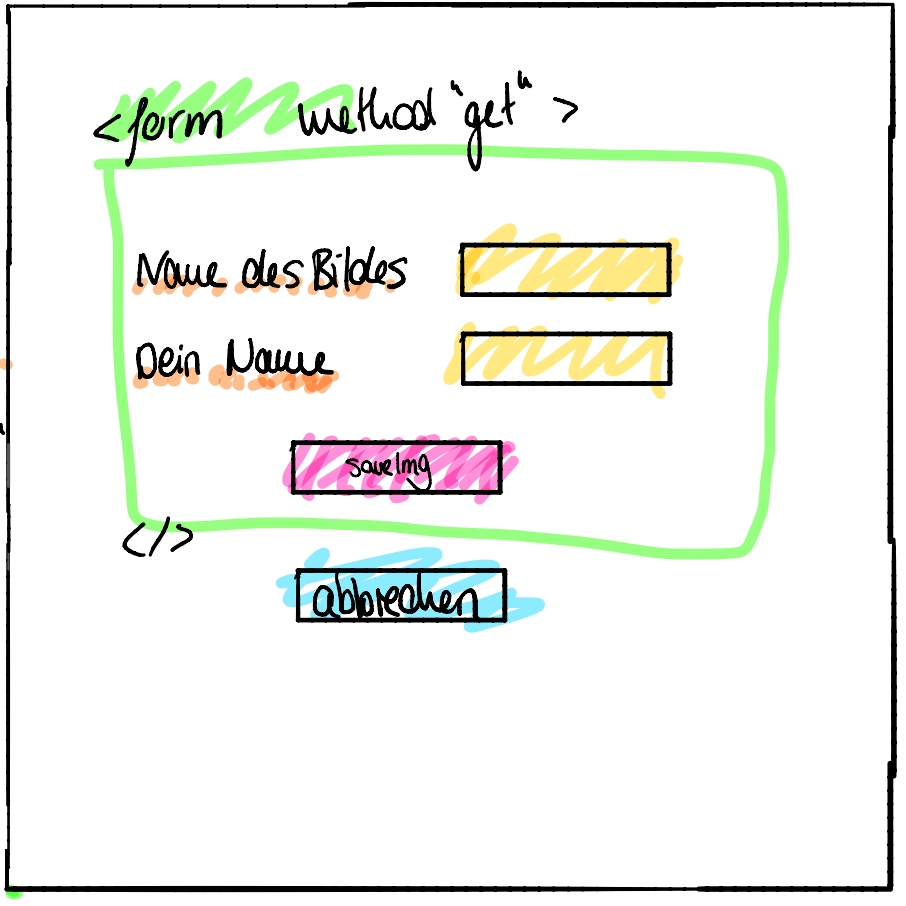
Skizze formularPage

<input type="text">
in HTML

<label for="">
in HTML

<input type="submit">
in HTML
↳ schickt Daten an Db

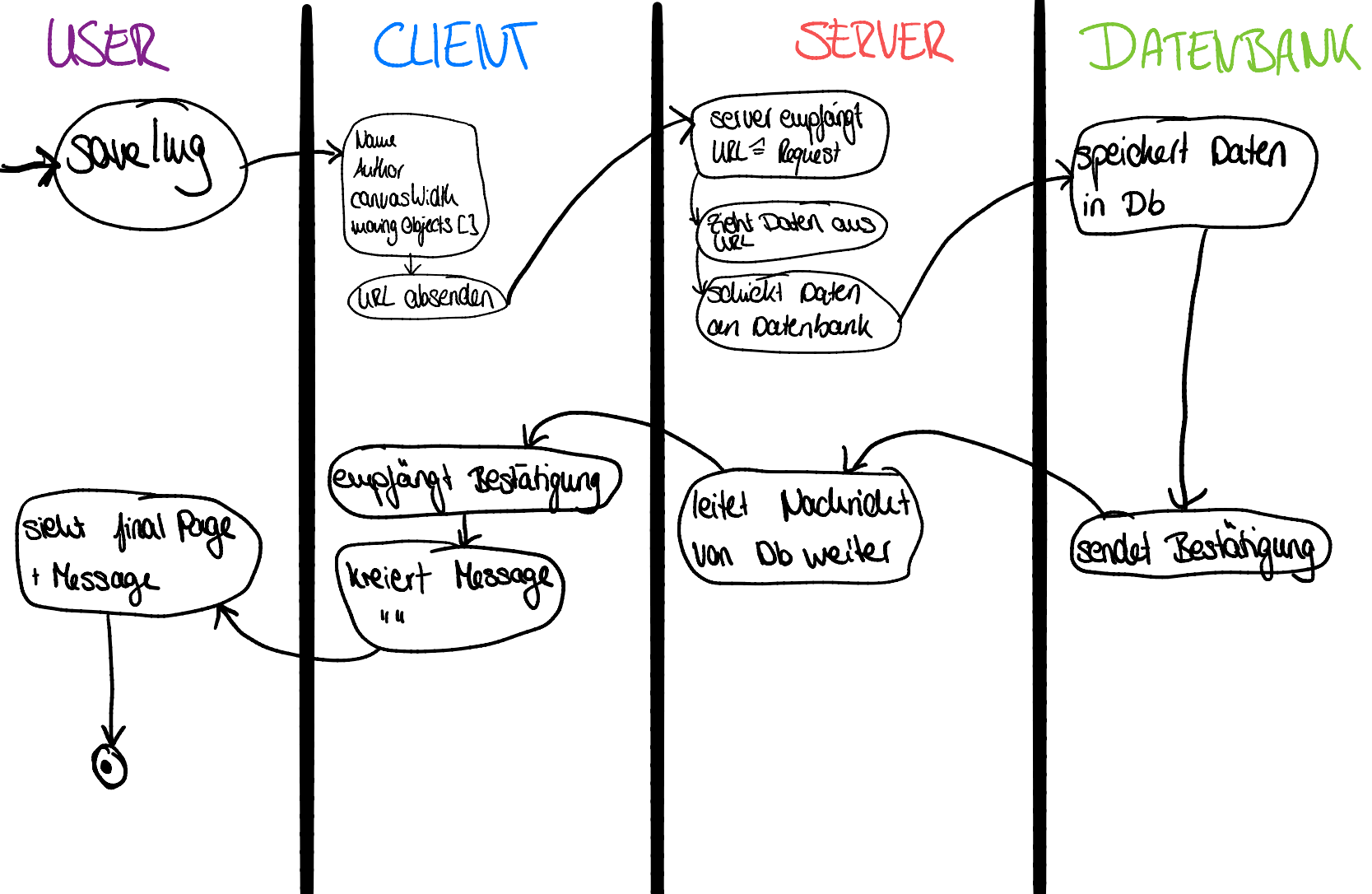
<button>
in HTML

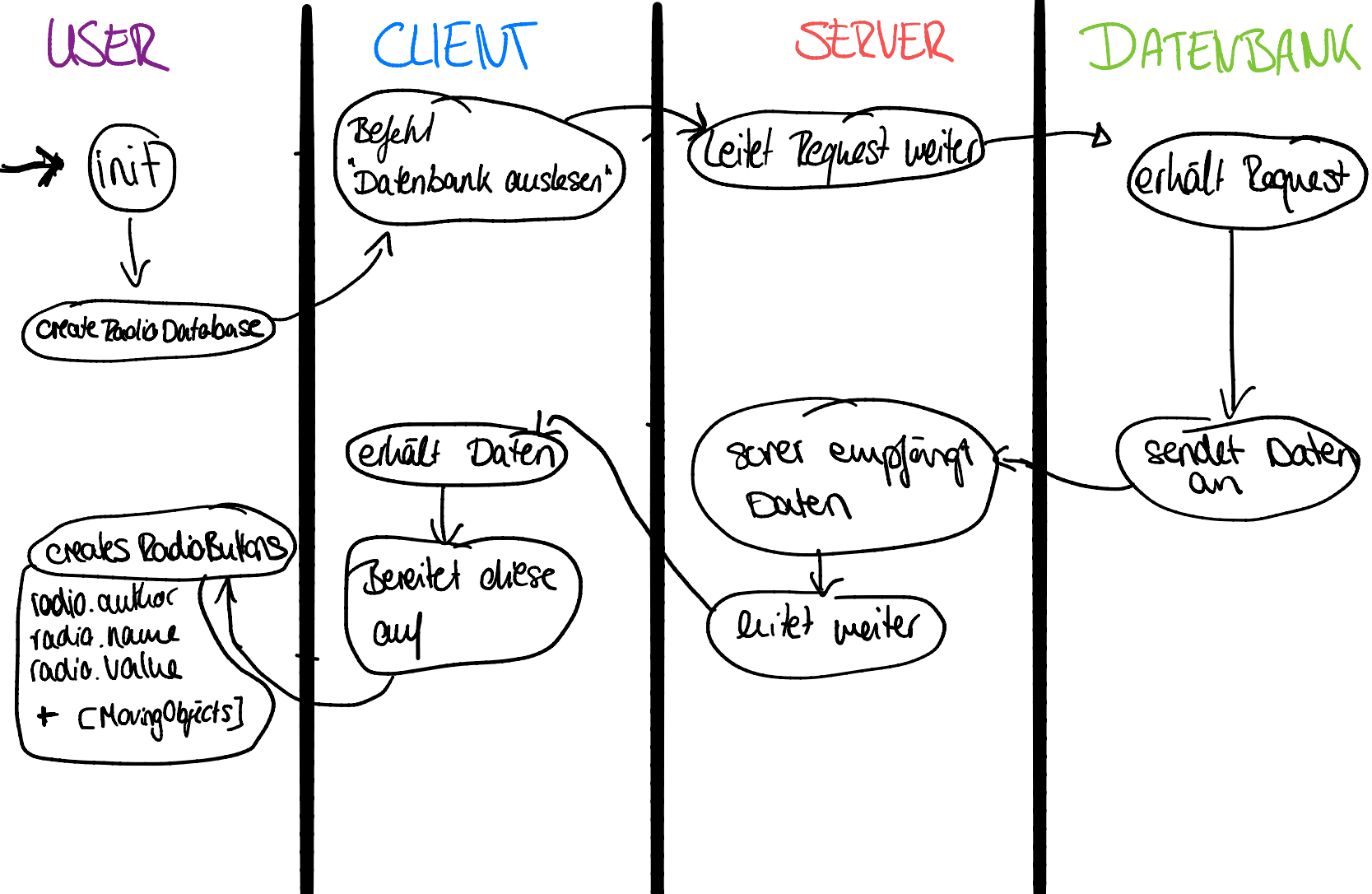


Skizze finalPage

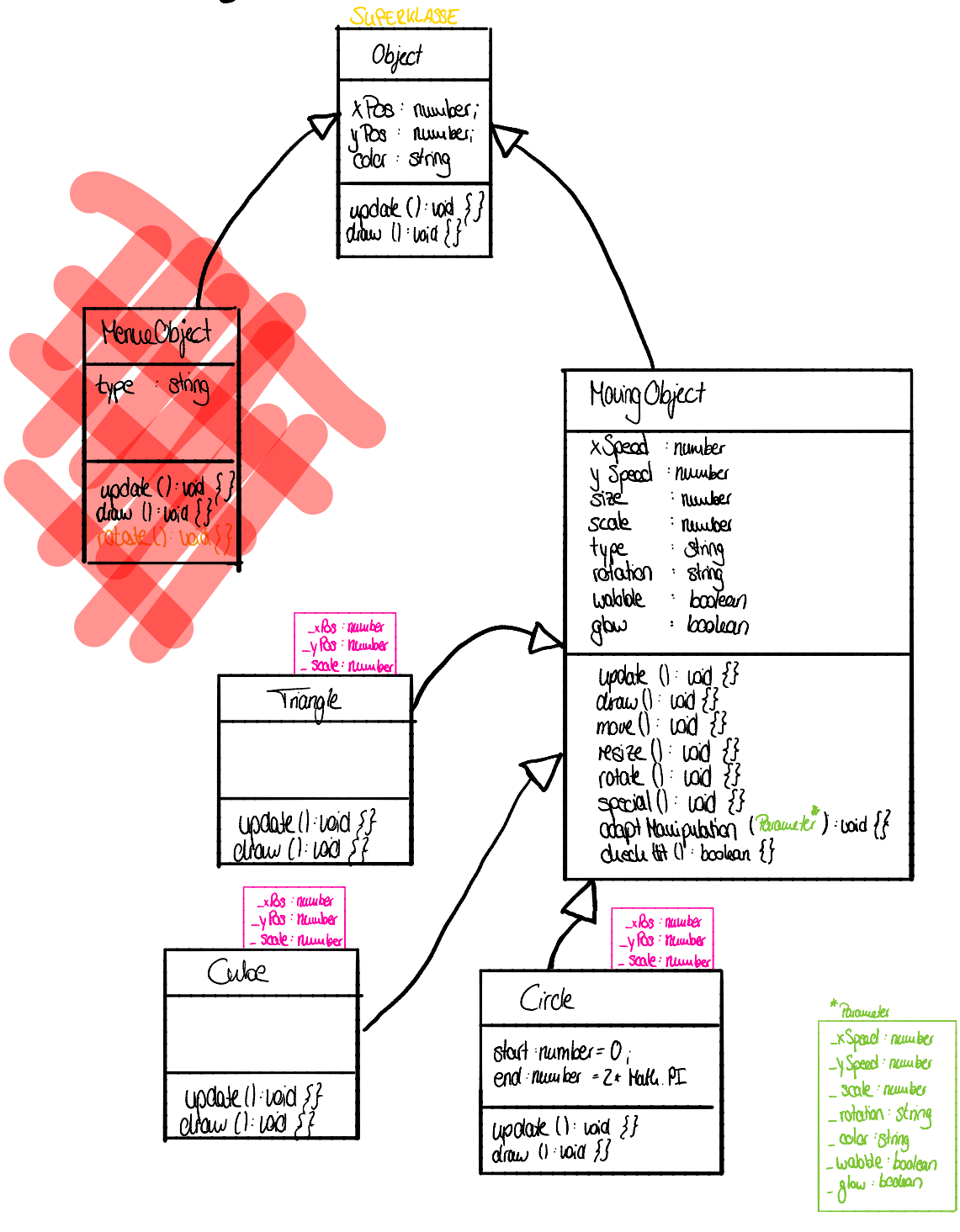
kurze Message Script

<button> in HTML


USER
CLIENT
SERVER
DATENBANK
Save Img
Name
Author
canvasWidth
moving Objects []
URL absenden
server empfängt URL ⇐ Request
Zieht Daten aus URL
Schickt Daten an Datenbank
speichert Daten in Db
sendet Bestätigung
leitet Nachricht von Db weiter
empfängt Bestätigung
kreiert Message
" "
sieht final Page + Message


USER
CLIENT
SERVER
DATENBANK
init
Befehl "Datenbank auslesen"
Leitet Request weiter
erhält Request
Create Radio Database
sendet Daten an
server empfängt Daten
erhält Daten
creates RadioButtons
radio.author
radio.name
radio.value
+ [MovingObjects]
Bereitet diese auf
leitet weiter

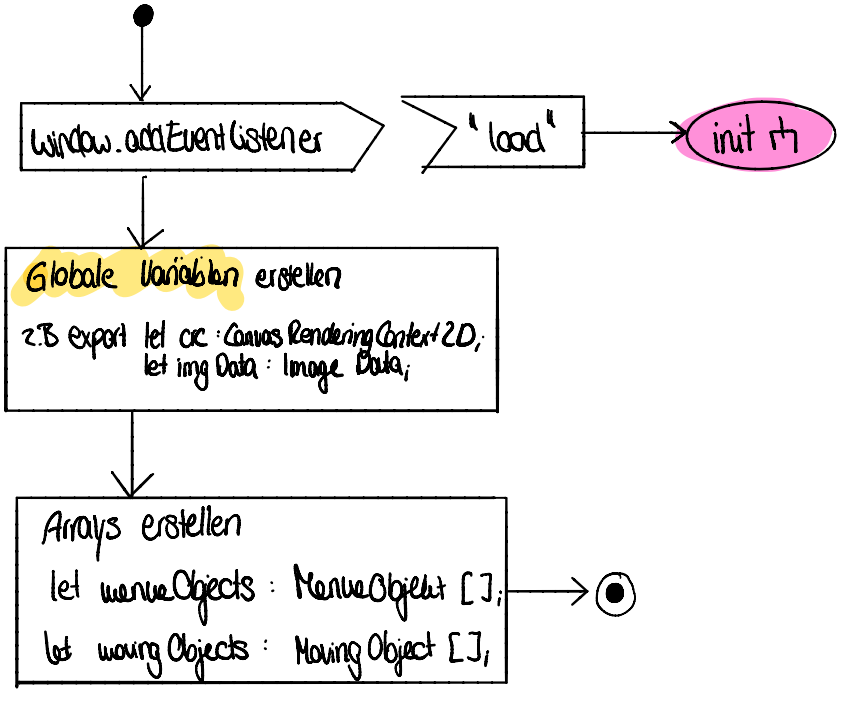
# Klassendiagramm



# Aktivitätsdiagramme

main.ts

# main.ts

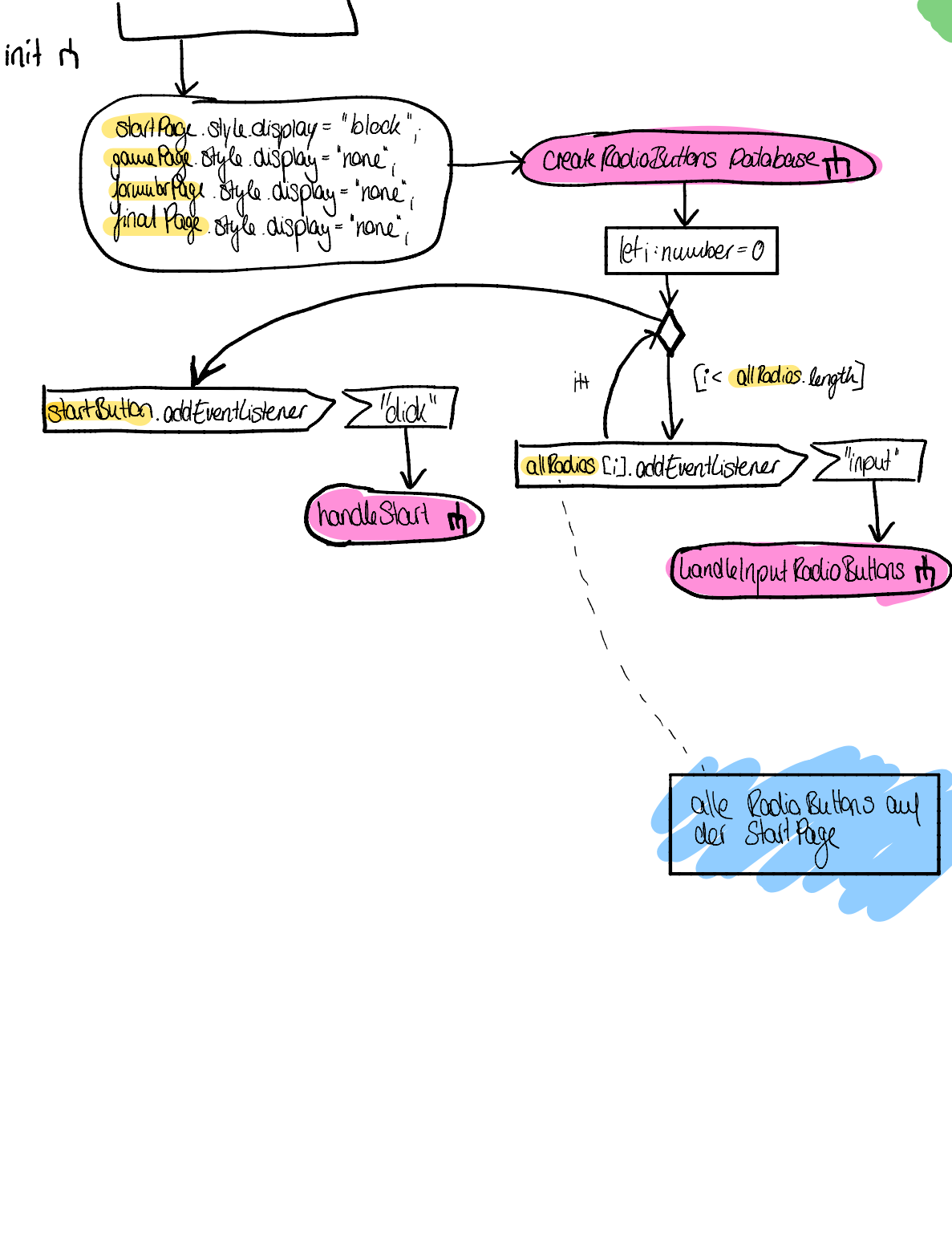


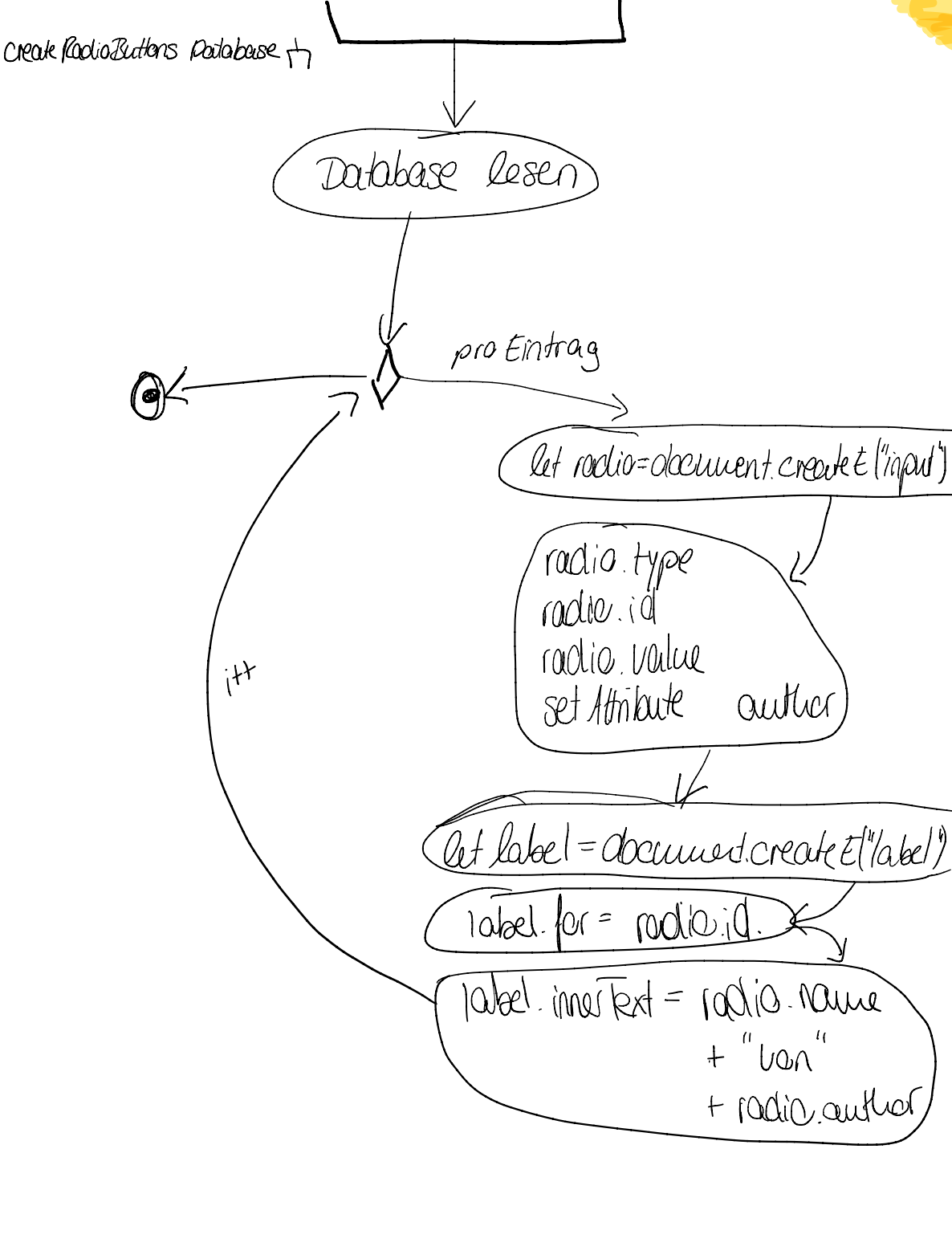

= Globale Variablen

= Funktionen main.ts

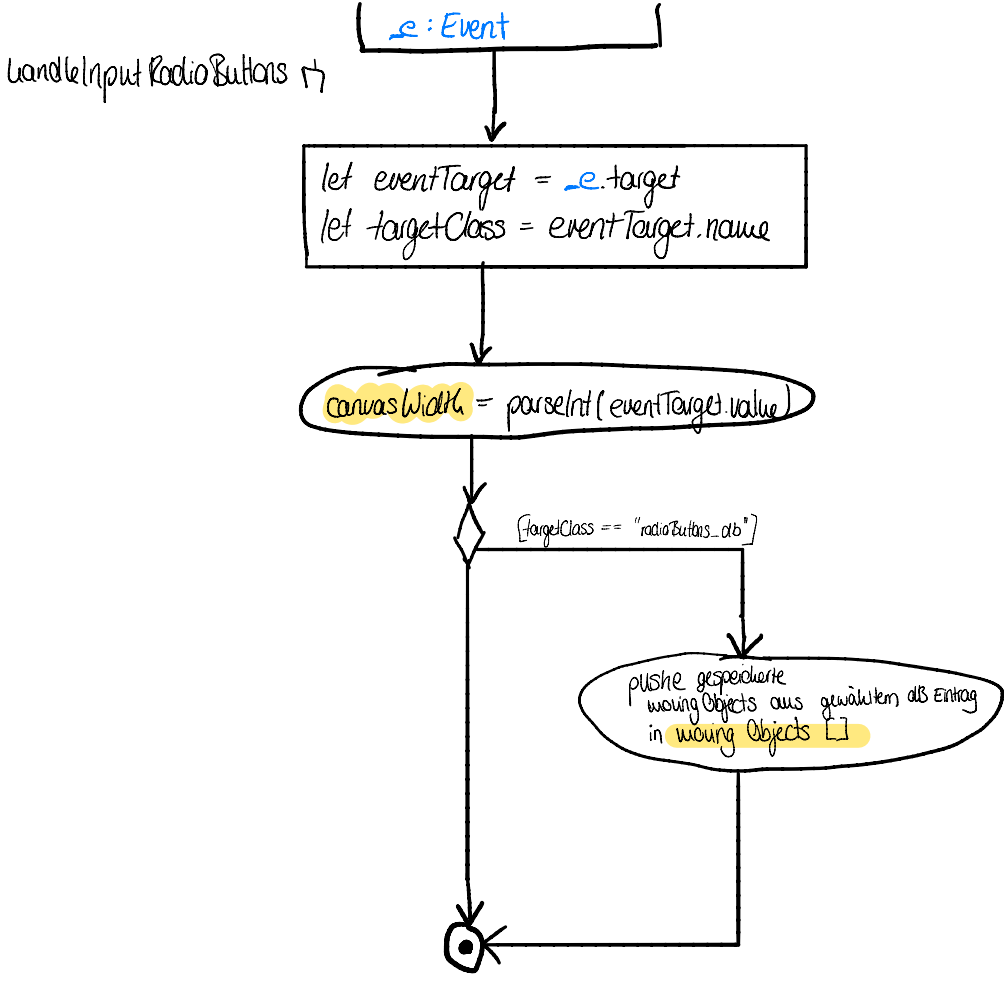
= Klassen methoden

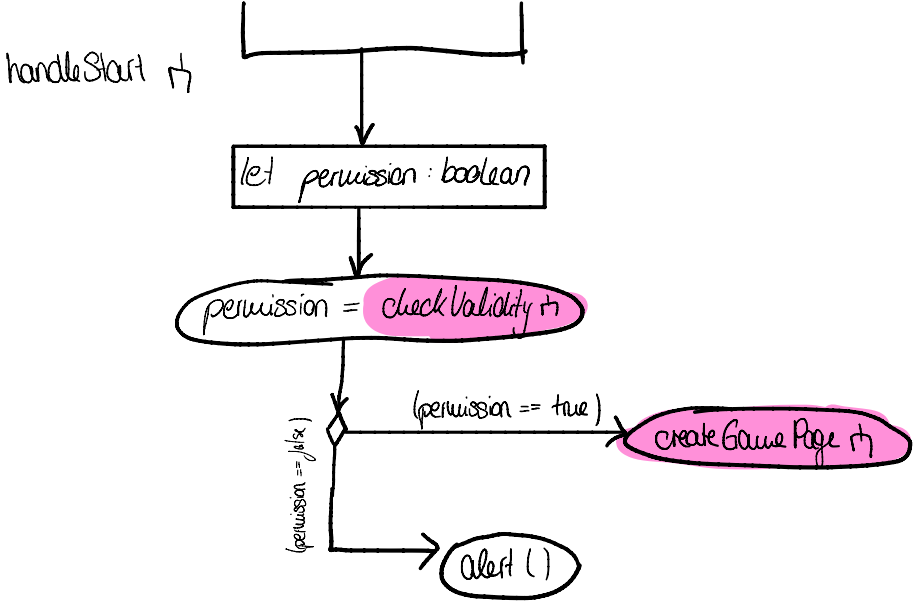
= Kommentar

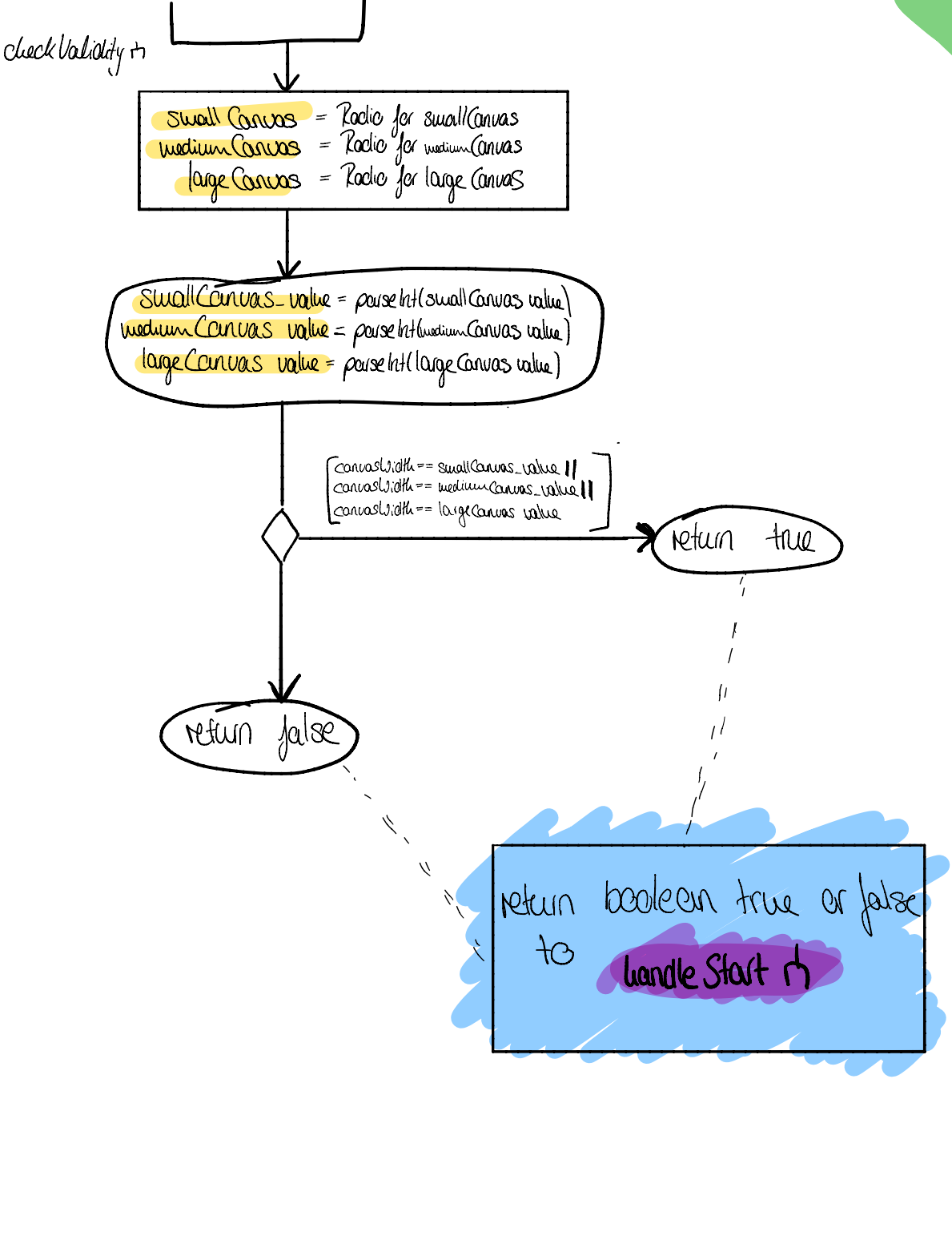

init
startPage.style.display = "block";
gamePage.style.display = "none";
formularPage.style.display = "none";
finalPage.style.display = "none";
createRadioButtonsDatabase
let i: number = 0
i++
[i < allRadios.length]
startButton.addEventListener
"click"
handleStart
allRadios[i].addEventListener
"input"
handleInputRadioButtons
alle RadioButtons auf der StartPage

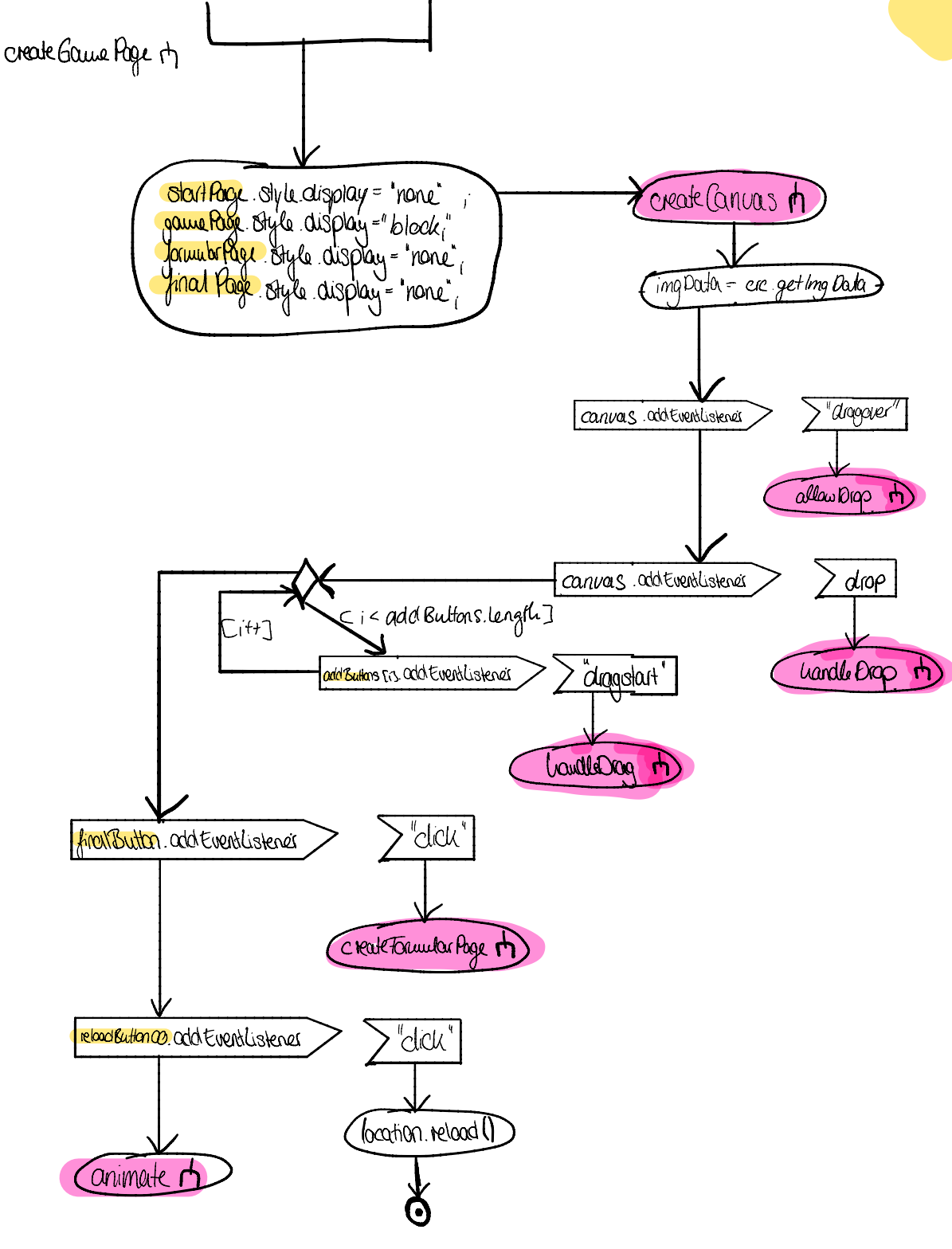

createRadioButtons Database
Database lesen
pro Eintrag
let radio=document.createE("input")
radio.type
radio.id
radio.value
setAttribute author
let label=document.createE("label")
label.for = radio.id
label.innerText = radio.name
+ "von"
+ radio.author
i++

handleInputRadioButtons

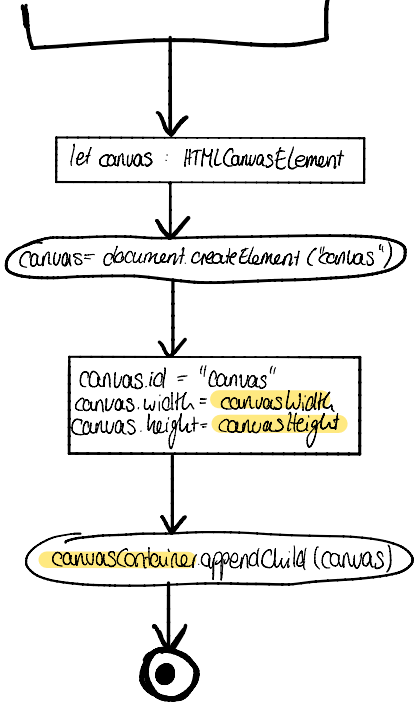


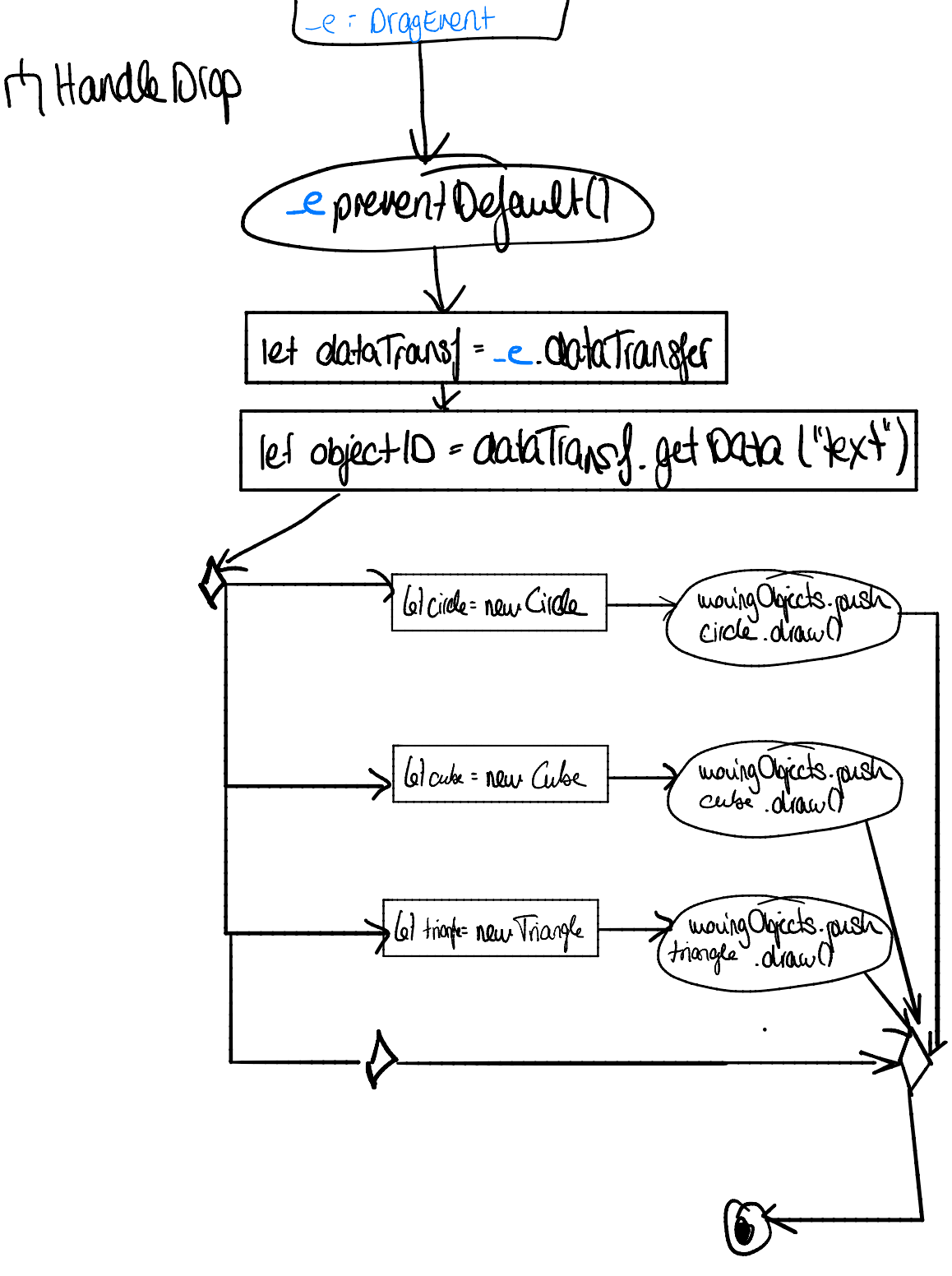

handleStart
let permission : boolean
permission = checkValidity
(permission == false)
createGamePage
(permission == false)
alert ( )

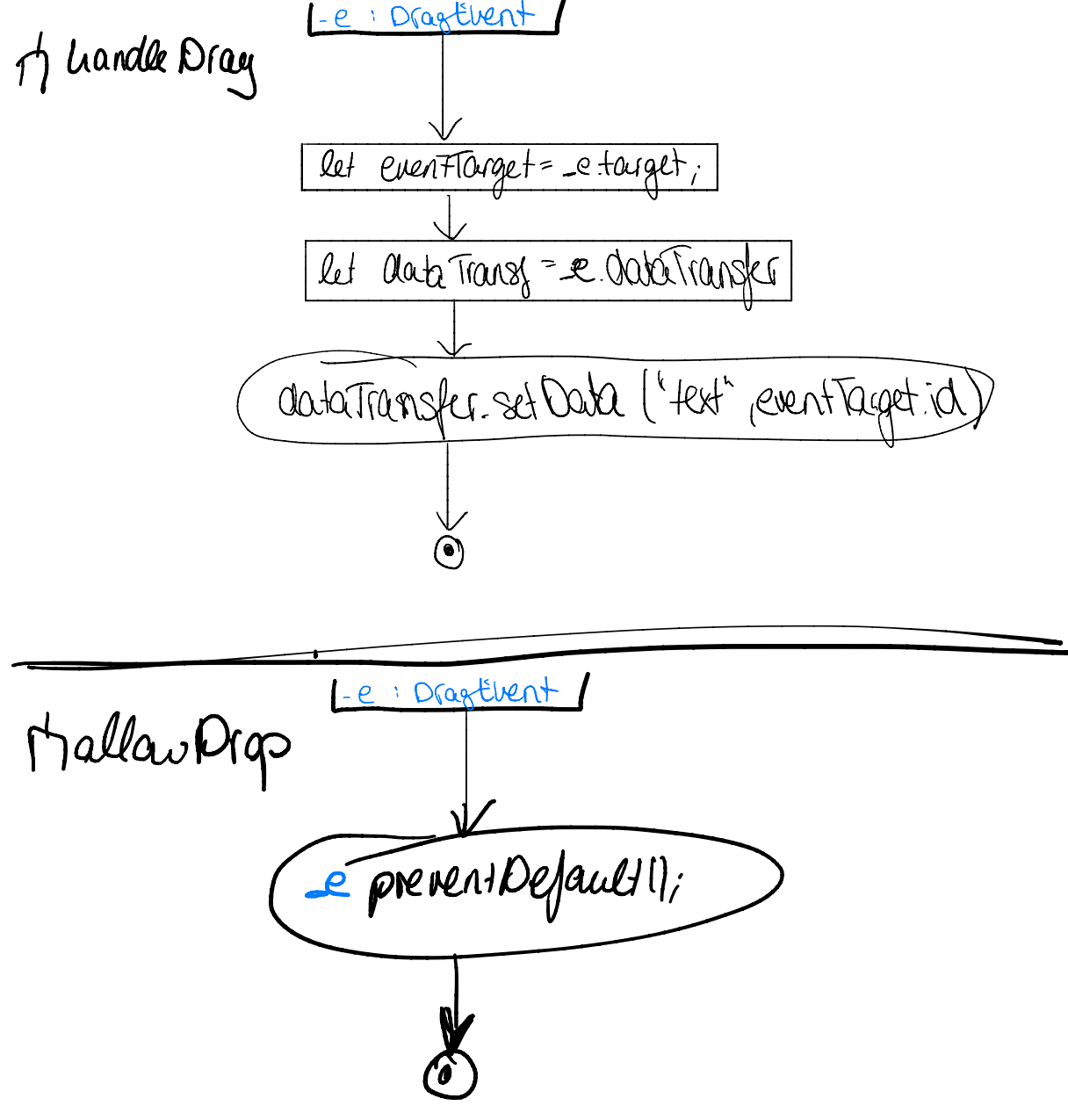

checkValidity fn
smallCanvas = Radio for smallCanvas
mediumCanvas = Radio for mediumCanvas
largeCanvas = Radio for largeCanvas
smallCanvas_value = parseInt(smallCanvas value)
mediumCanvas value = parseInt(mediumCanvas value)
largeCanvas value = parseInt(largeCanvas value)
canvasWidth == smallCanvas_value ||
canvasWidth == mediumCanvas_value ||
canvasWidth == largeCanvas value
return true
return false
return boolean true or false
to
handleStart fn

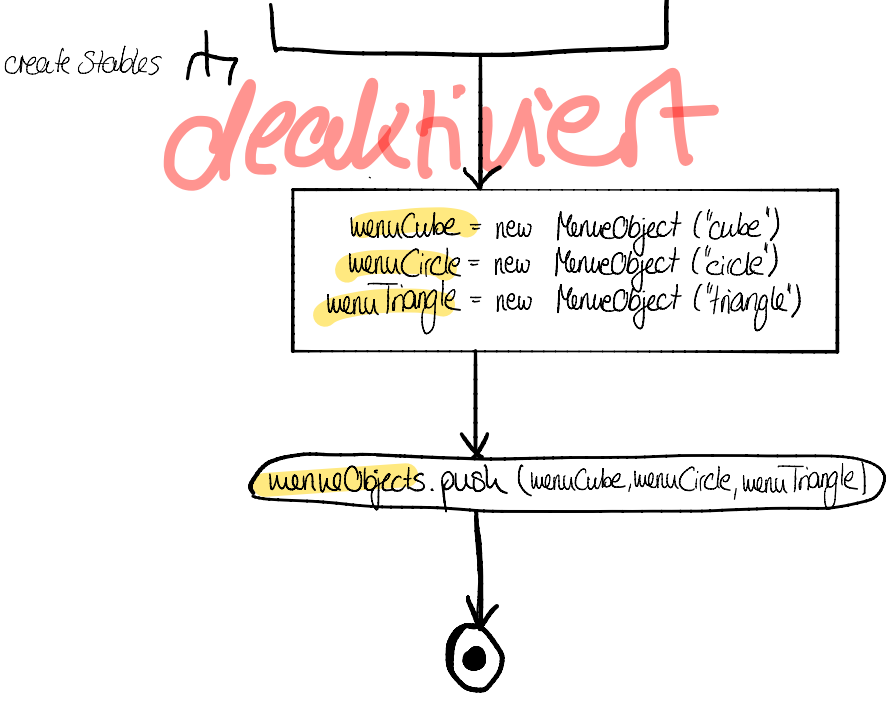

createGamePage
startPage.style.display = "none";
gamePage.style.display = "block";
formularPage.style.display = "none";
finalPage.style.display = "none";
createCanvas
imgData = ctx.getImgData
canvas.addEventListener
"dragover"
allowDrop
canvas.addEventListener
drop
handleDrop
[i++]
i < addButtons.length
addButtons[i].addEventListener
"dragstart"
handleDrag
finalButton.addEventListener
"click"
createFormularPage
reloadButton.addEventListener
"click"
location.reload()
animate

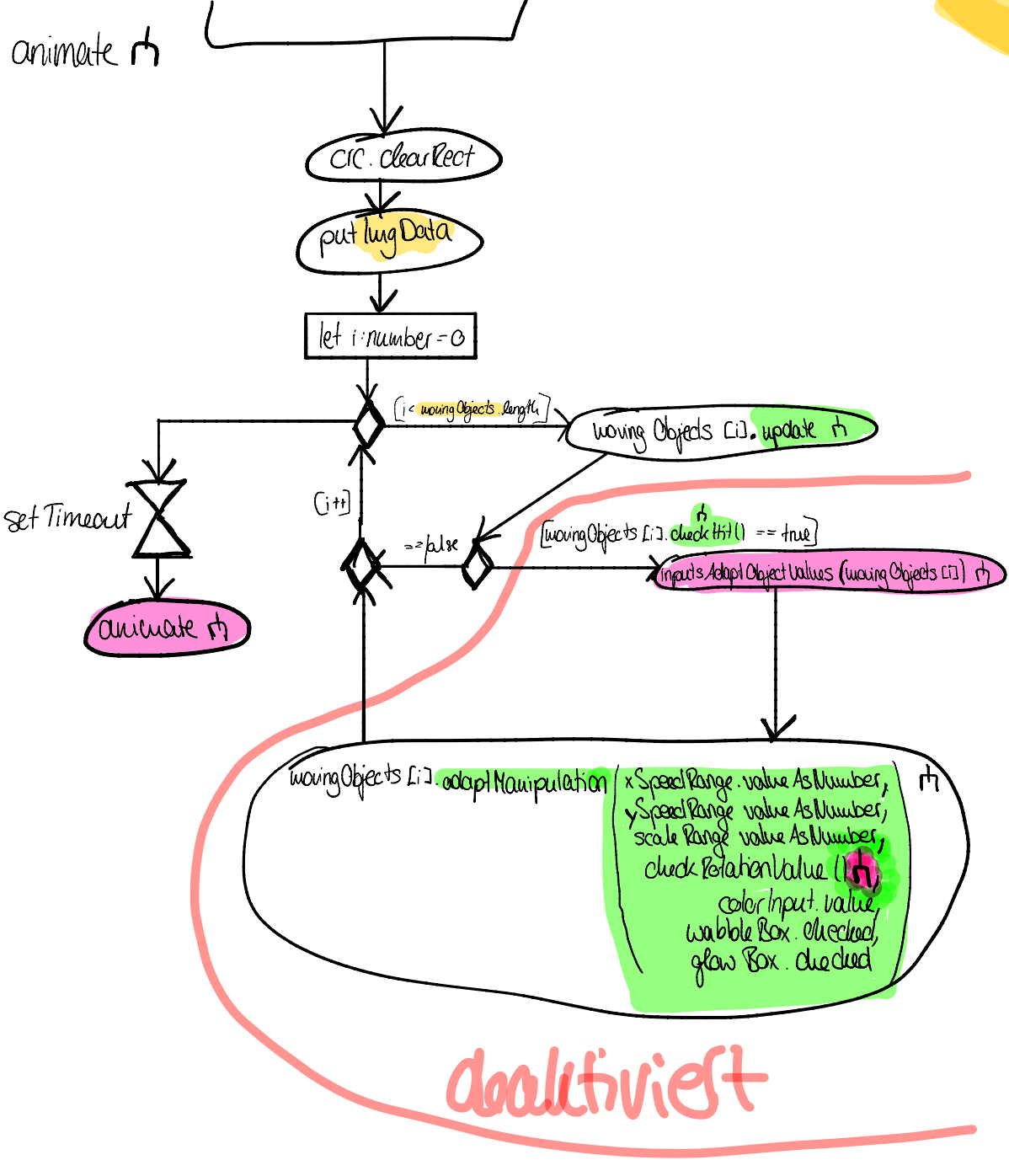
createCanvas



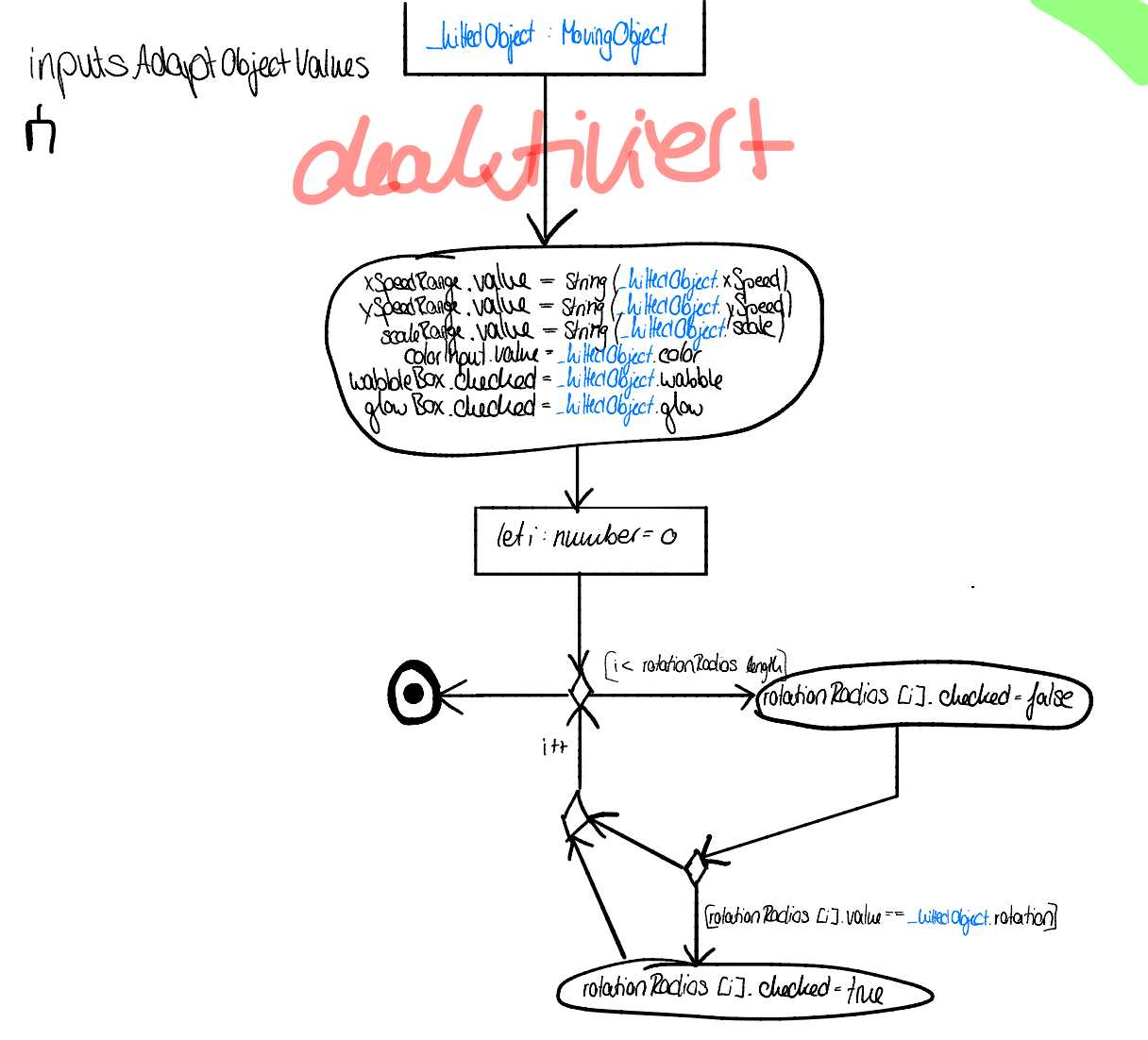

_e : DragEvent
Handle Drop
_e.preventDefault()
let dataTransf = _e.dataTransfer
let objectID = dataTransf.getData("text")
let circle = new Circle
movingObjects.push
circle.draw()
let cube = new Cube
movingObjects.push
cube.draw()
let triangle = new Triangle
movingObjects.push
triangle.draw()

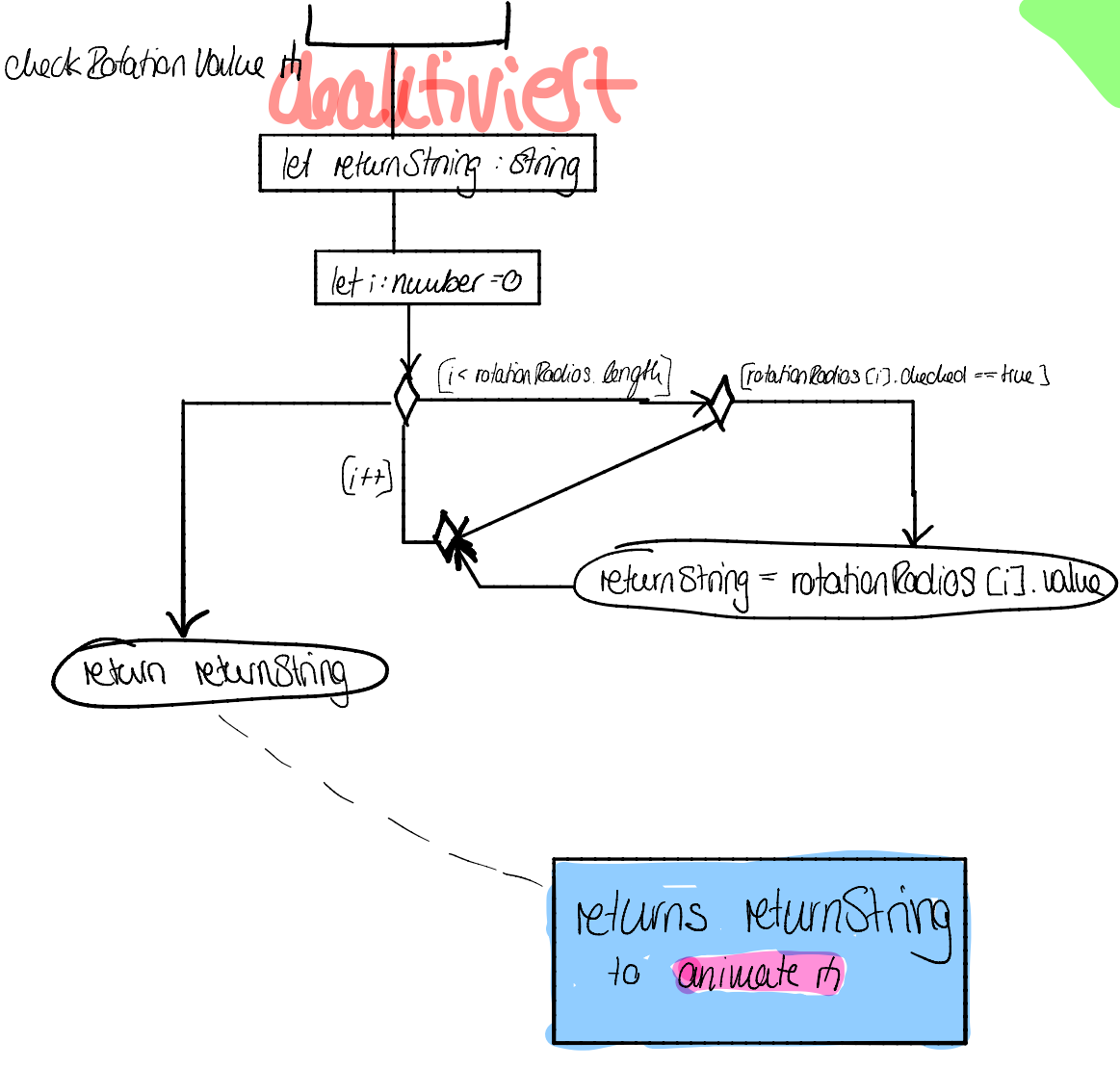

handleDrag
-e : DragEvent
let eventTarget = e.target;
let dataTransfer = e.dataTransfer
dataTransfer.setData("text", eventTarget.id)
allowDrop
-e : DragEvent
e.preventDefault();

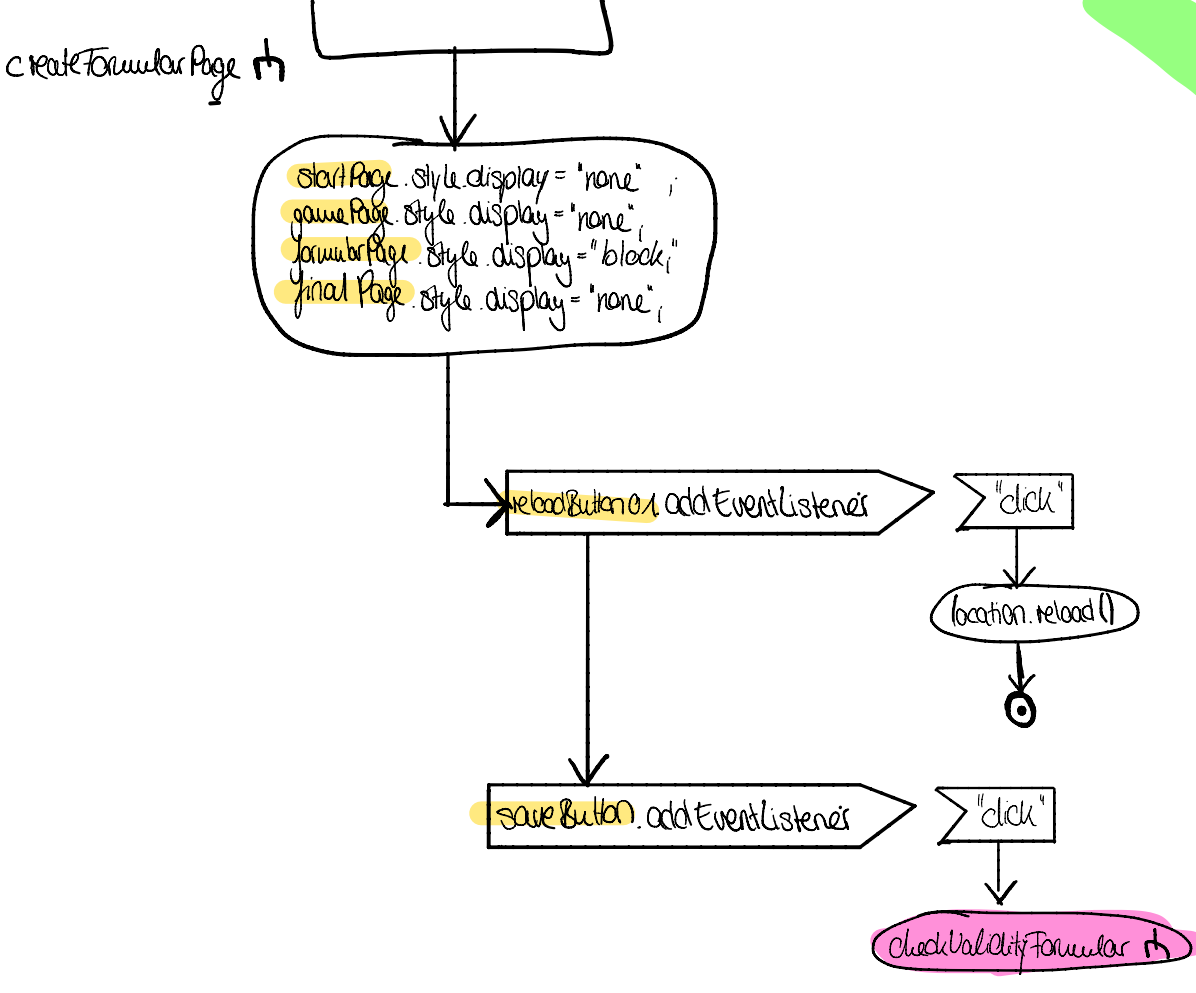

create Stables
deaktiviert
menuCube = new MenueObject ("cube")
menuCircle = new MenueObject ("circle")
menuTriangle = new MenueObject ("triangle")
menueObjects.push (menuCube, menuCircle, menuTriangle)

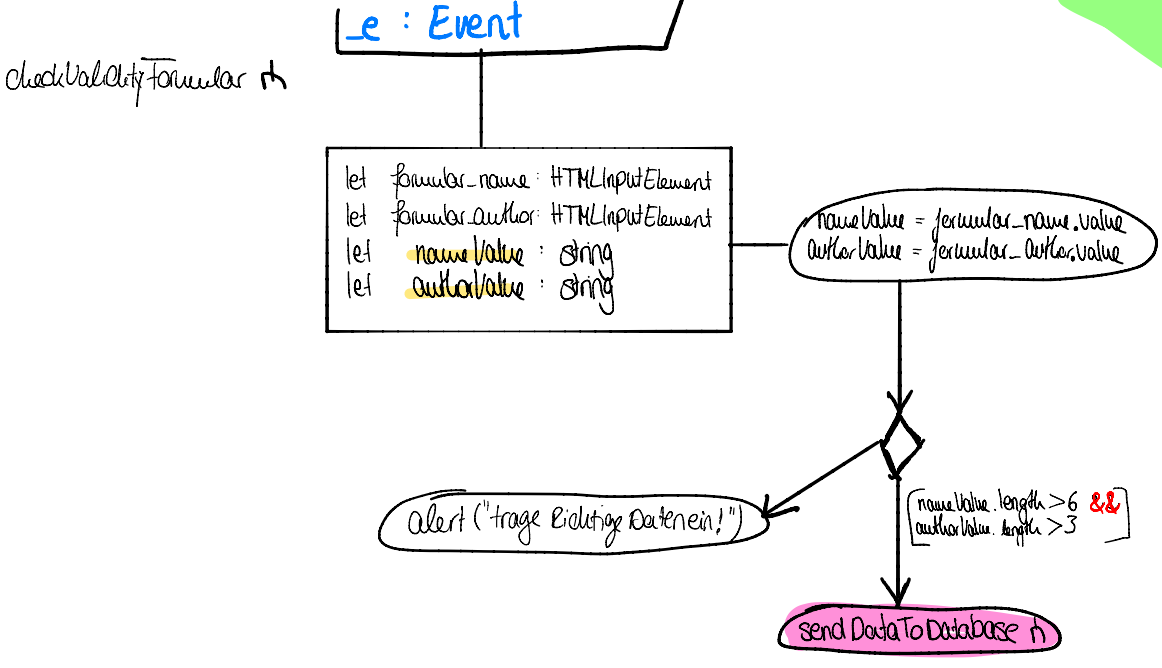

animate
ctx.clearRect
putImgData
let i:number = 0
[i < movingObjects.length]
movingObjects[i].update
setTimeout
[i++]
==pulse
[movingObjects[i].checkHit() == true]
inputsAdaptObjectValues(movingObjects[i])
animate
movingObjects[i].adaptManipulation
xSpeedRange.valueAsNumber,
ySpeedRange.valueAsNumber,
scaleRange.valueAsNumber,
checkRotationValue(),
colorInput.value,
wobbleBox.checked,
glowBox.checked
deaktiviert

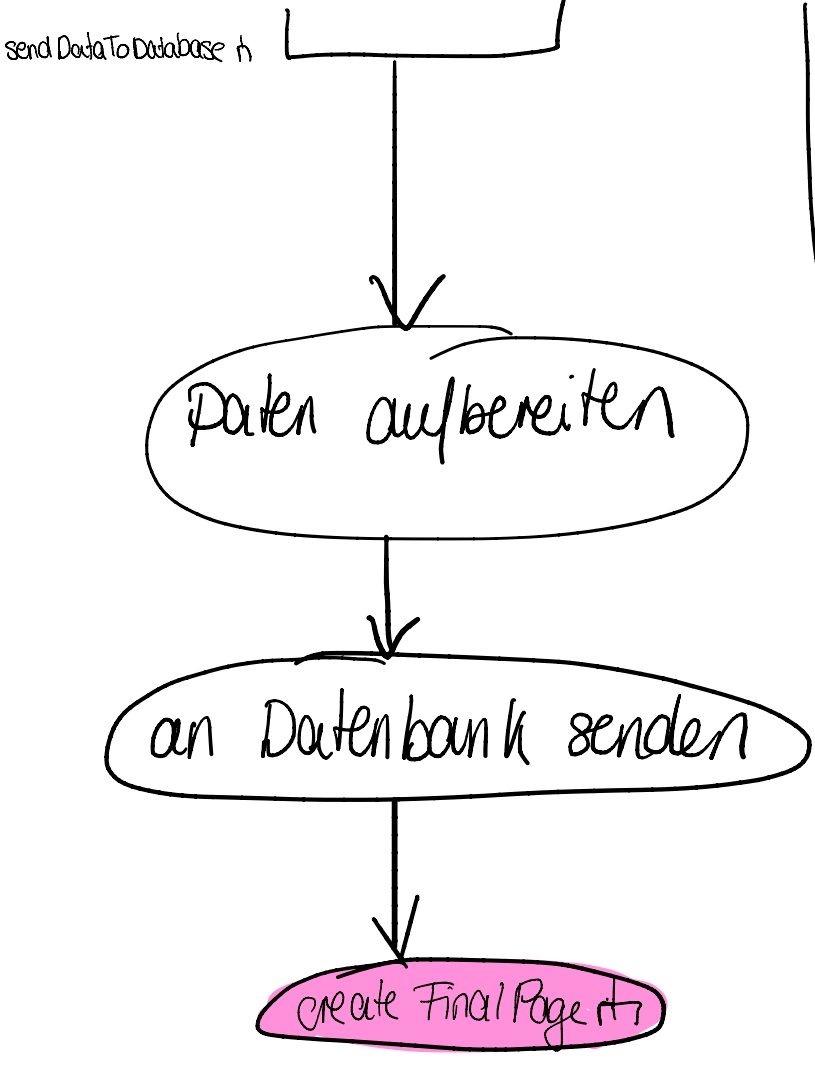
inputs Adapt Object Values




checkRotationValue()
deactivate
let returnString : string
let i : number = 0
(i < rotationRadios.length)
[rotationRadios[i].checked == true]
(i++)
returnString = rotationRadios[i].value
return returnString
returns returnString
to animate()

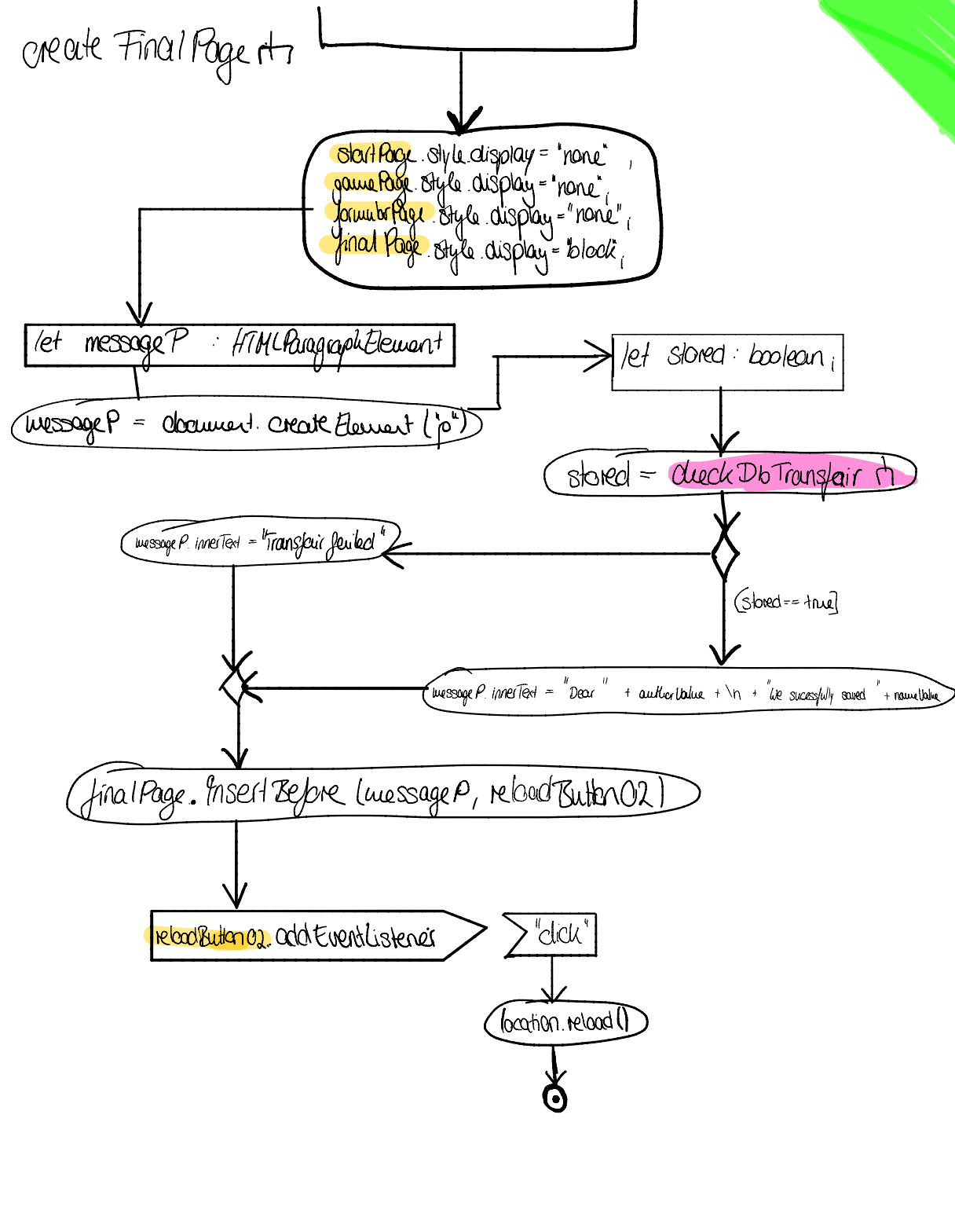

createFormularPage
startPage.style.display = "none";
gamePage.style.display = "none";
formularPage.style.display = "block";
finalPage.style.display = "none";
reloadButton.addEventListener
"click"
location.reload()
saveButton.addEventListener
"click"
checkValidityFormular

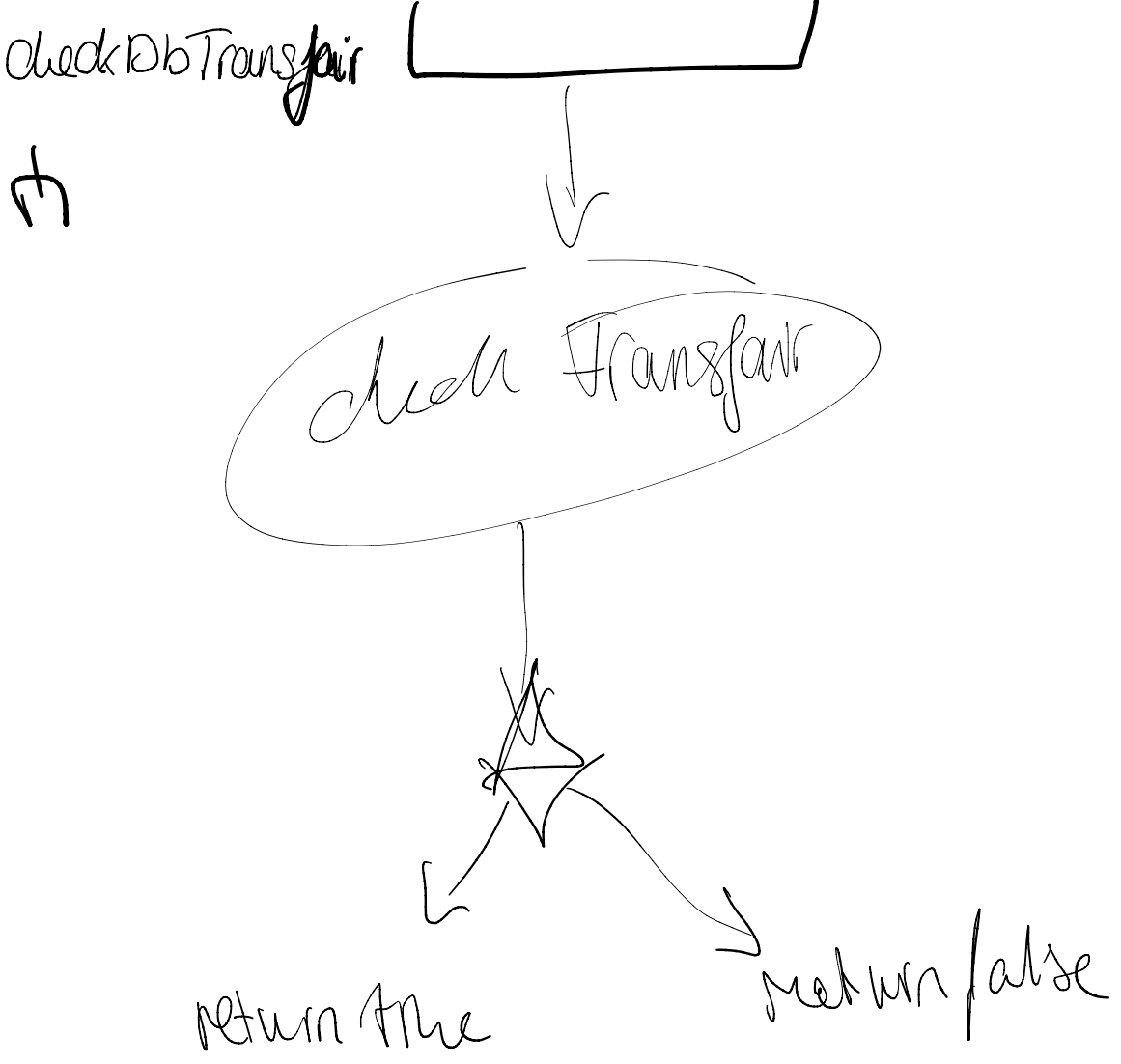

checkValidityFormular
_e : Event
let formular_name : HTMLInputElement
let formular_author : HTMLInputElement
let nameValue : String
let authorValue : String
nameValue = formular_name.value
authorValue = formular_author.value
alert("trage richtige Daten ein!")
nameValue.length > 6 &&
authorValue.length > 3
sendDataToDatabase




**Muss** gespeichert werden

- movingObjects []
- canvasWidth
- name
- author
- date


createFinalPage()
startPage.style.display = "none";
gamePage.style.display = "none";
formularPage.style.display = "none";
finalPage.style.display = "block";
let messageP : HTMLParagraphElement
let stored : boolean;
messageP = document.createElement("p")
stored = checkDbTransfair()
messageP.innerText = "Transfair fehlerd"
(stored == true)
messageP.innerText = "Dear " + authorValue + \n + "We sucessfully saved " + nameValue
finalPage.insertBefore(messageP, reloadButton02)
reloadButton02.addEventListener
"click"
location.reload()


checkDbTransfair
check Transfair
return true
return false