Anleitung zur Interaktion mit Anwendung

Mit dem „wähle Canvasgröße" (dropdown) kann zwischen den Canvasgrößen "klein, Mittel und Groß" gewählt werden

Bei dem „Wähle Hintergrund" (dropdown) kann die Farbe des Canvas gewählt werden (Farben: Blau, Gelb, Grau, Beige)

Dann kann der Nutzer eine der Formen auswählen und irgendwo auf dem canvas plazieren indem er erst die gewünschte Form anklickt und dann die gewünschte Stelle auf dem Canvas.

Wenn der Nutzer auf der Tastatur auf "D" drückt und danach auf ein Symbol auf dem Canvas wird dieses gelöscht, danach kann er weiter Symbole auf dem Canvas plazieren.

Mit dem "Save" button kann das Bild gespeichert werden

Mit dem "load" dropdown kann ein zuvor gespeichertes Bild wieder geladen werden

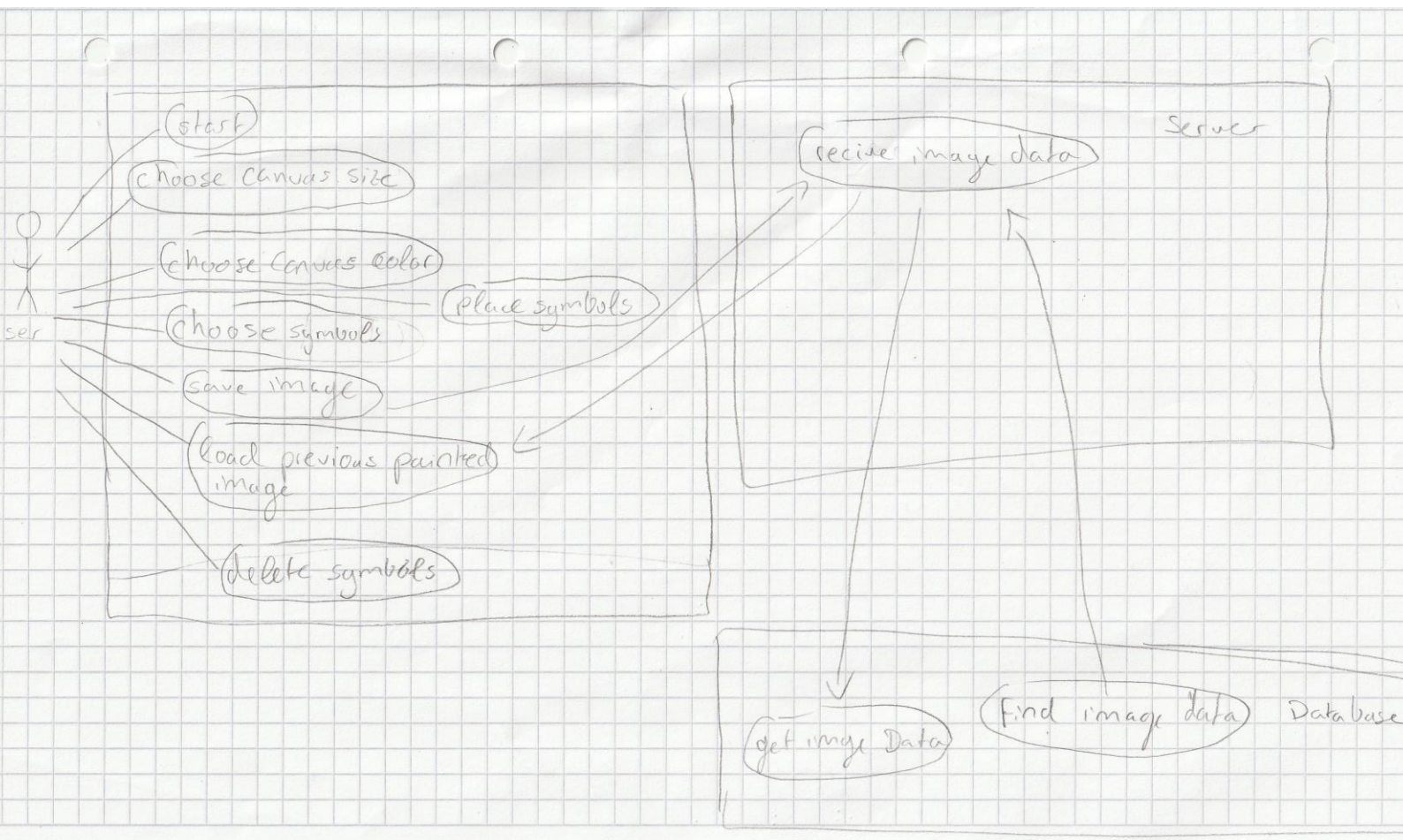
Anleitung Heroku und MongoDB

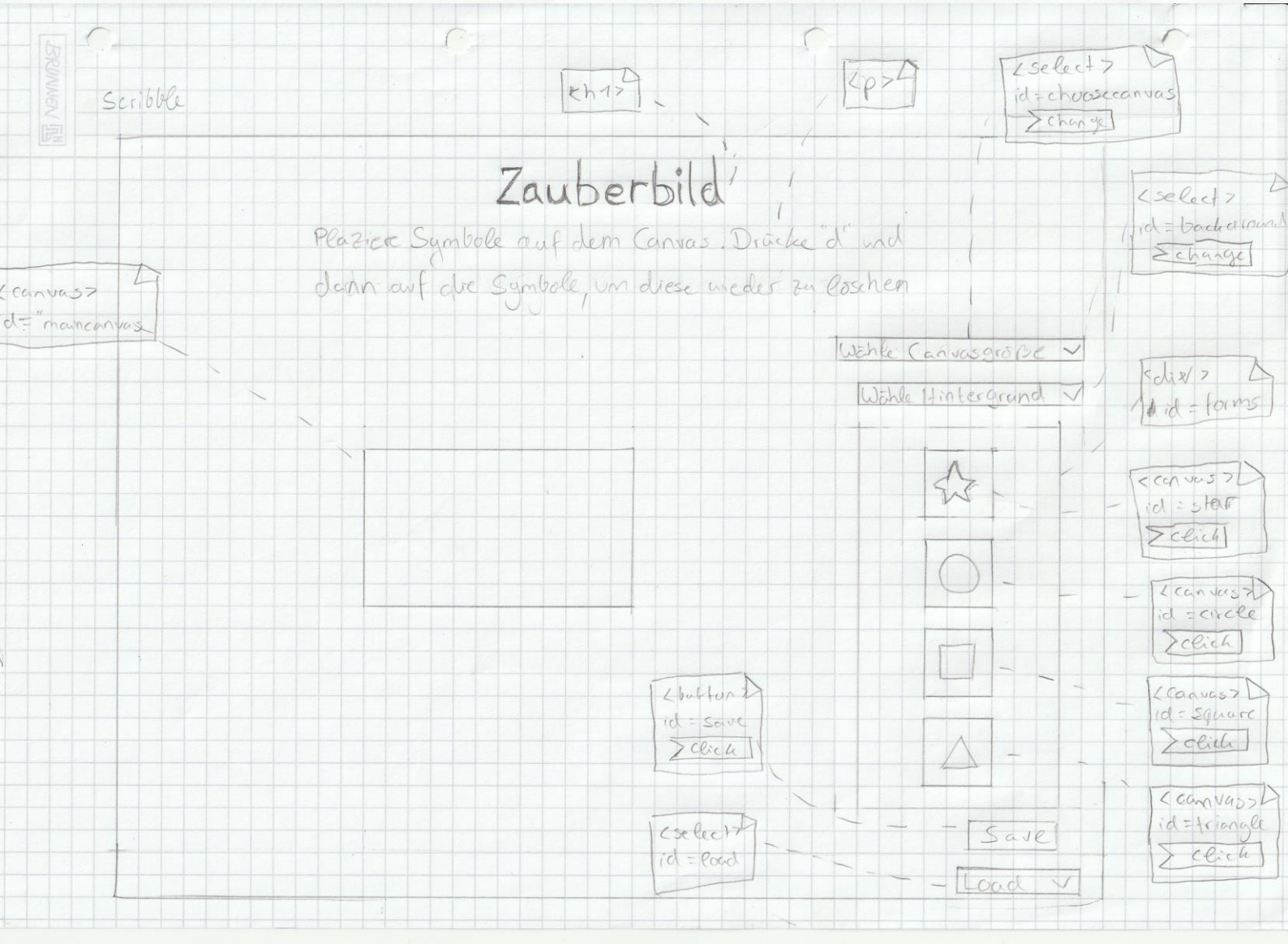
Mongo:

1. Bei Mongo anmelden
2. Einen cluster erstellen
3. Einen Testuser erstellen (Name, Passwort, Zugriff erlauben)
4. In Zeile 13 in Server 2.ts die databaseURL ~~von~~ an Testnutzer anpassen

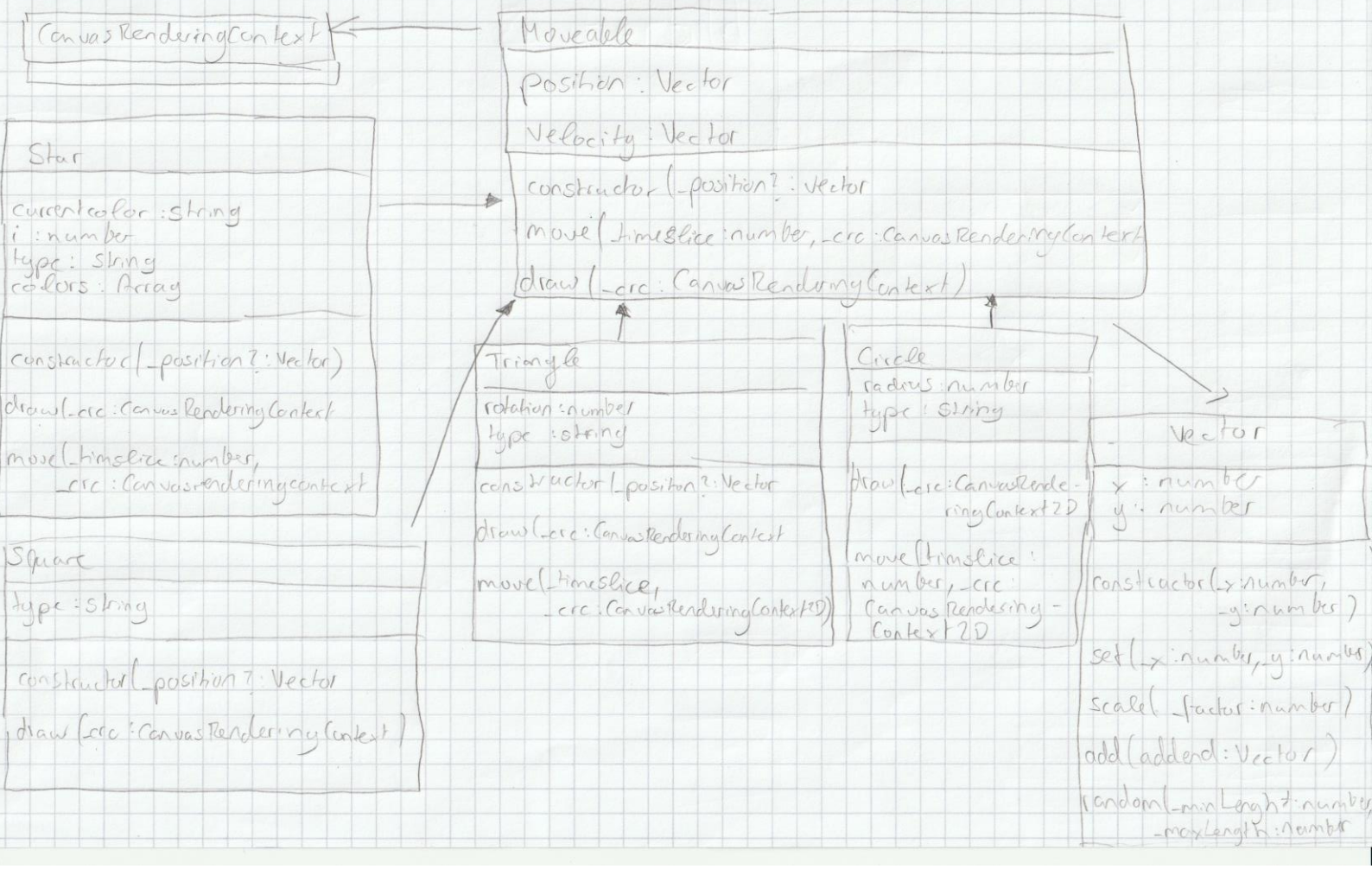
Heroku

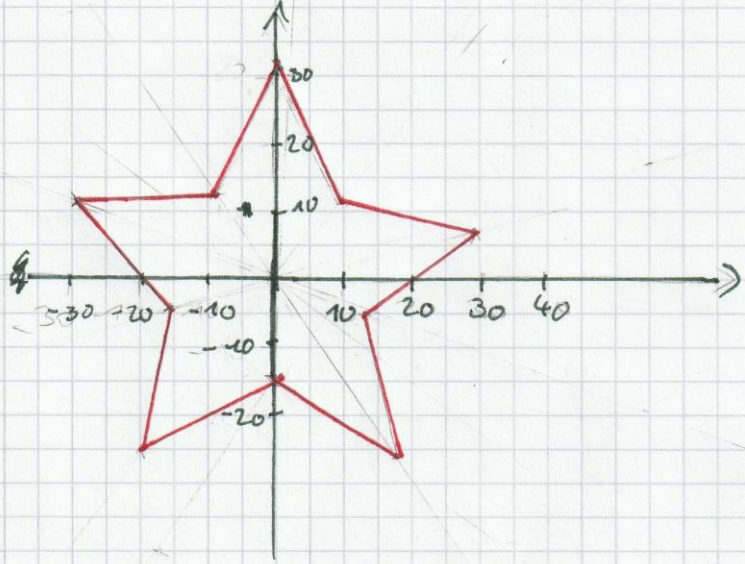
1. Nutzerkonto anlegen mit Primary language: Node.js
2. Neue app erstellen
3. Package.json Datei erstellen -> bei "start" Pfad zur Serverdatei angeben
4. Bei "Deploy" mit Github verbinden
5. Deployen :-

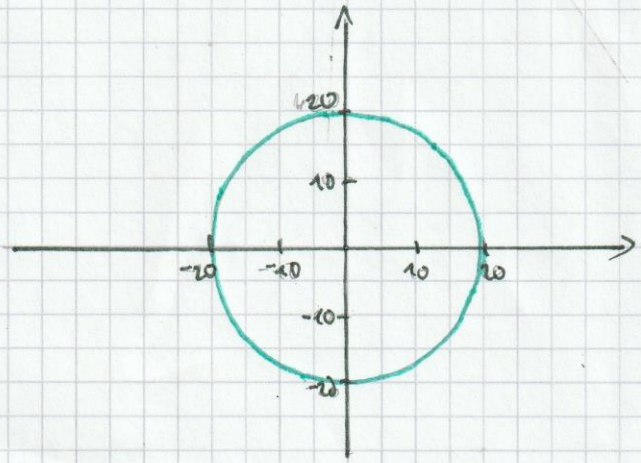

Start
Choose Canvas size
Choose Canvas color
Choose symbols
Save image
Load previous painted image
delete symbols
Place symbols
ser
Server
recive image data
get image Data
Find image data
Database

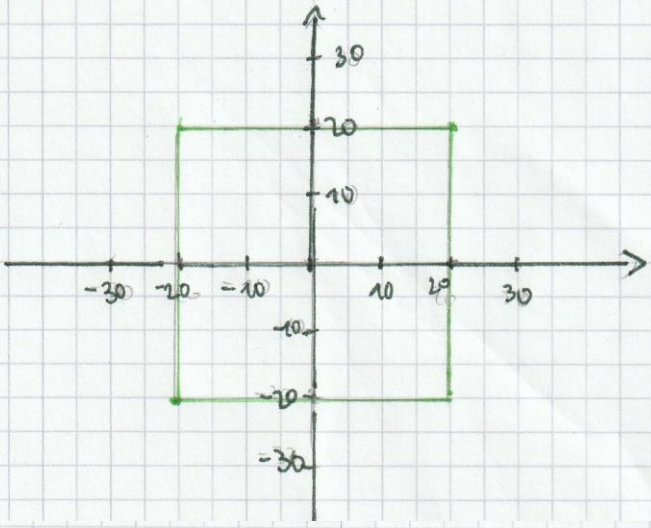

Scribble
<h1>
<p>
<select>
id = choosecanvas
change
Zauberbild
Plaziere Symbole auf dem Canvas. Drücke "d" und
dann auf die Symbole, um diese wieder zu löschen
<canvas>
d="maincanvas
Wähle Canvasgröße
Wähle Hintergrund
<select>
id = background
change
<click>
id = forms
<canvas>
id = star
click
<canvas>
id = circle
click
<canvas>
id = square
click
<canvas>
id = triangle
click
<button>
id = save
click
<select>
id = load
Save
Load

Classdiagram




30
20
10
-30
-20
-10
10
20
30
40
-10
-20


20
10
-20
-10
10
20
-10
-20


30
20
10
-30
-20
-10
10
20
30
-10
-20
-30


10
20
30
40

AD Zauberscanvas

```
let chosenform: string;
let moveables: Moveable[] = []
let backgroundimage: ImageData
let trash: boolean = false
let maincanvas: HTMLCanvasElement
let savedpicture: HTMLInputElement
```

addload listener

load → handleload

change in choosecanvas → handlechange

change in Background → selectBackground

click on triangle → chooseform

click on circle → chooseform

click on star → chooseform

click on square → chooseform

click on maincanvas → placeform

keydown → deletemode

change in load → loadPicture

click on maincanvas → deleteform

click on save → getName

handleLoad

drawforms()

findpictures()

install click-, change- and load-eventlisteners

window.setInterval(update, 20)

drawforms

```
let canvastriangle: HTMLCanvasElement = document.getElementById("triangle")
let positionsquare: Vector = new Vector(30,30);
let positioncircle: Vector = new Vector(50,50);
let positiontriangle: Vector = new Vector(70,60)
let triangle: Triangle = new Triangle(positiontriangle)
let crc2: CanvasRenderingContext2D = canvastriangle.getContext("2d")
```

triangle.draw(crc2)

```
let canvascircle: HTMLCanvasElement = document.getElementById("circle")
let crc1: CanvasRenderingContext2D = canvascircle.getContext("2d")
let circle: Circle = new Circle(positionCircle)
```

circle.draw(crc1)

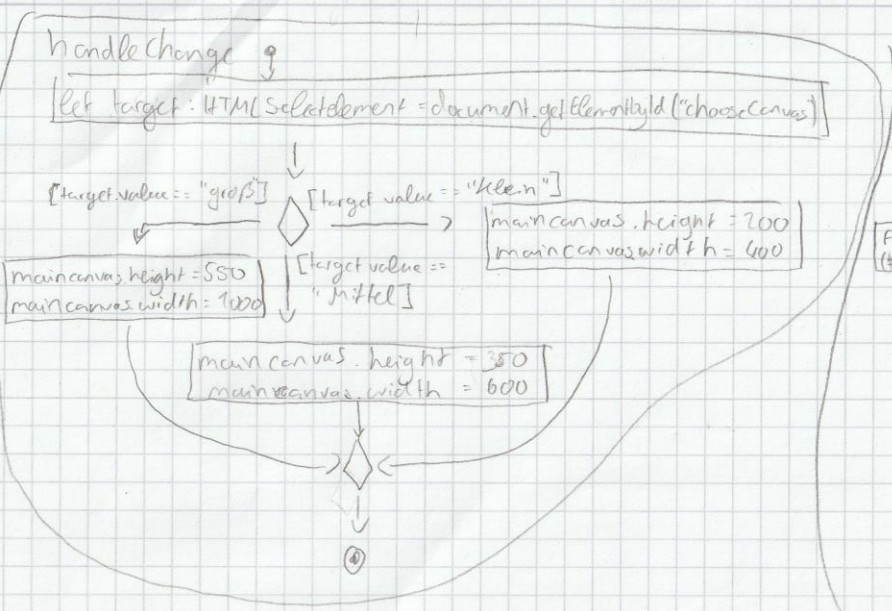
```
let canvassquare: HTMLCanvasElement = document.getElementById("square")
let crc3: CanvasRenderingContext2D = canvascircle.getContext("2d")
let square: Square = new Square(positionsquare)
```

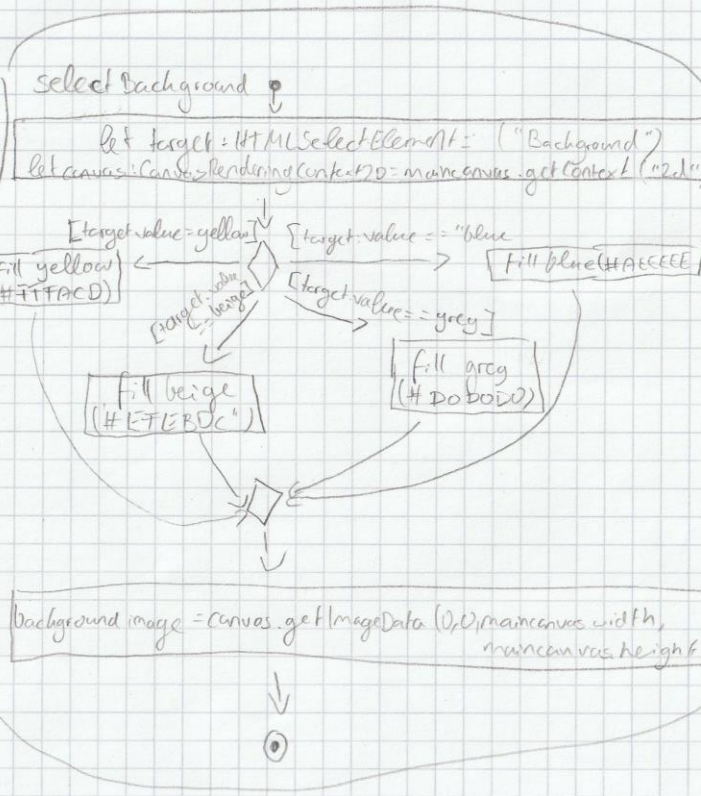
square.draw(crc3)

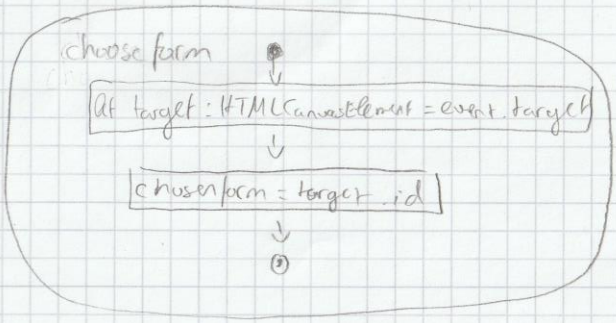
```
let canvasstar: HTMLCanvasElement = document.getElementById("star")
let crc4: CanvasRenderingContext2D = canvasstar.getContext("2d")
let star: star = new star(positionCircle
```

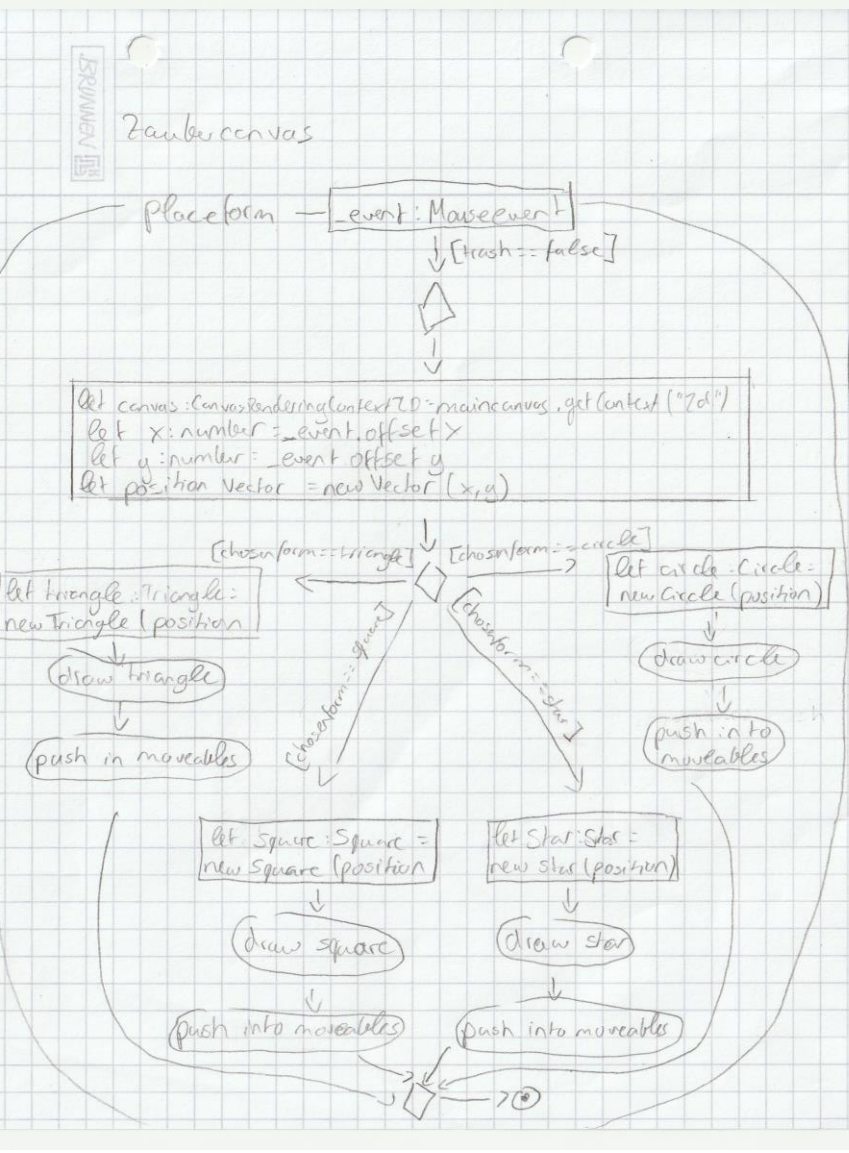
→ Star.draw(crc4)

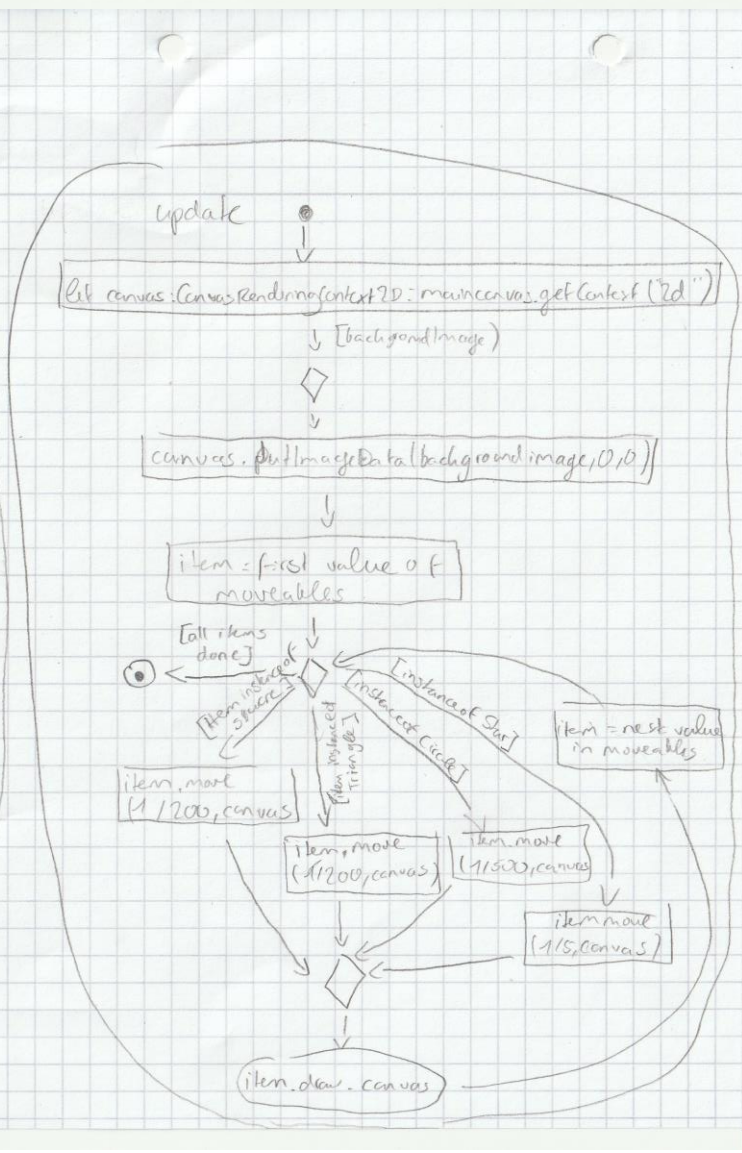
ZauberCanvas AD











BRUNNER

Zaubercanvas AD

deleteForm
_event: MouseEvent
_trash: boolean

[trash == true]

let positionX: number = _event.clientX
let positionY: number = _event.clientY
let canvasRect: ClientRect/DomRect = mainCanvas.getBoundingClientRect()
let offsetX: number = positionX - canvasRect.left
let offsetY: number = positionY - canvasRect.top

item = first value in moveables

[if item is close to clicked position]

item = next value in moveables

let index: number = moveables.indexOf(item)

moveables.splice(index, 1)

trash = false

deleteMode
_event: KeyboardEvent

[event.key == "d"]

trash = true

getName

let pictureName: string = prompt("Gib einen Namen ein!")

insertPicture(pictureName)

BRUNNER

Vector AD

Constructor
_x: number
_y: number

this.set(_x, _y)

set
+x: number
-y: number

this.x = _x
this.y = _y

scale
_factor: number

this.x *= _factor
this.y *= _factor

add
_addend: number

this.x += _addend.x
this.y += _addend.y

random
_minLength: number
_maxLength: number

length: number = minLength + Math.random() * (_maxLength - _minLength)

direction: number = Math.random() * 2 * Math.PI

this.set(Math.cos(direction), Math.sin(direction))

this.scale(length)

Movable

constructor

_position?: Vector

Set position to position or (30,30)

Set velocity to (0,0)

Set random velocity to (100,200)

move

_timeSlice: number

add velocity * _timeSlice to position

[position comp. < 0]

add canvas dimension to component

[position component > canvas dimension]

Substract canvas dimension from component

draw

Square

constructor

_position?: Vector

super(_position)

this.type = "Square"

draw

_crc: CanvasRenderingContext2D

_crc.save

_crc.translate(this.position.x, this.position.y)

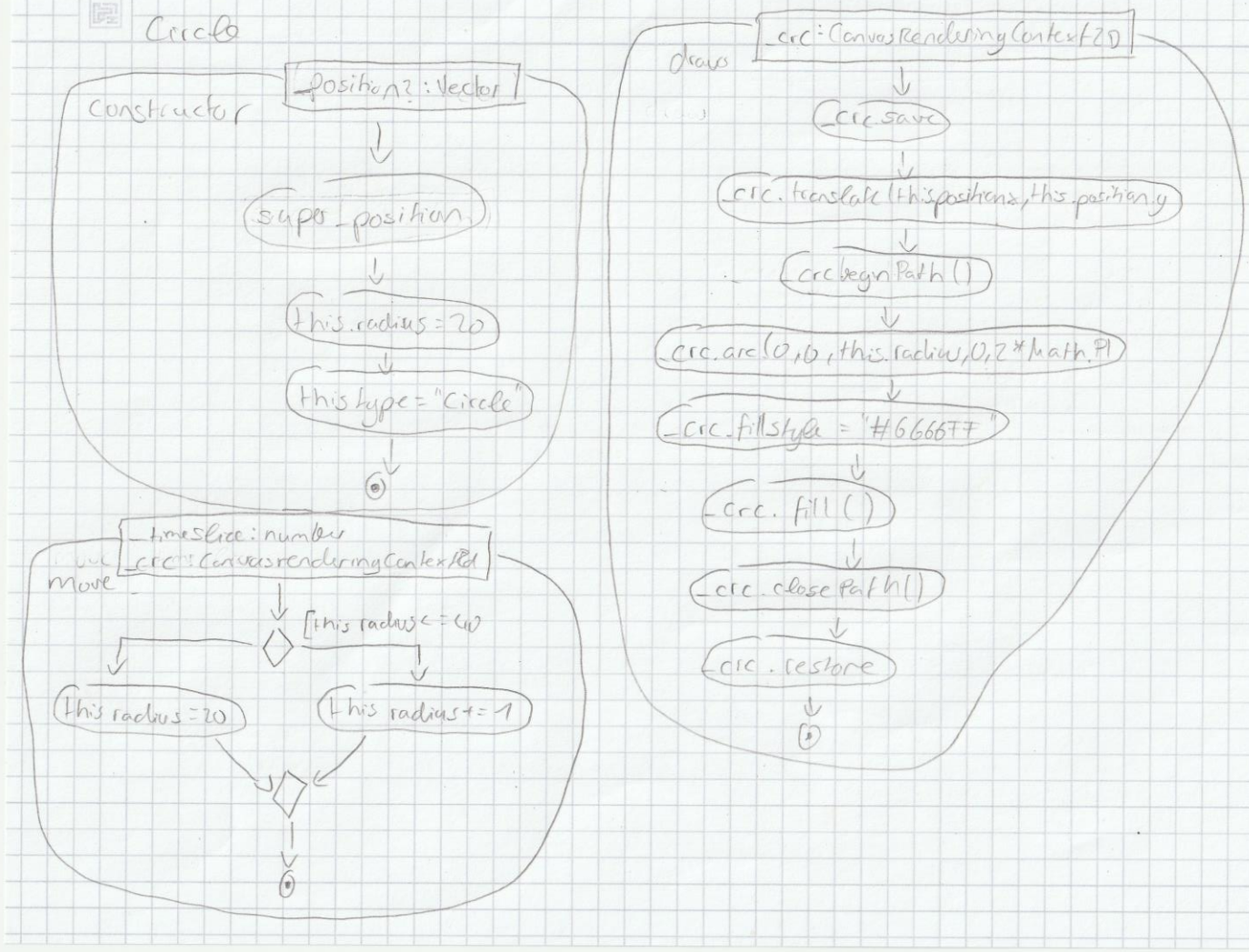
_crc.beginPath

_crc.fillStyle = "#32CD32"

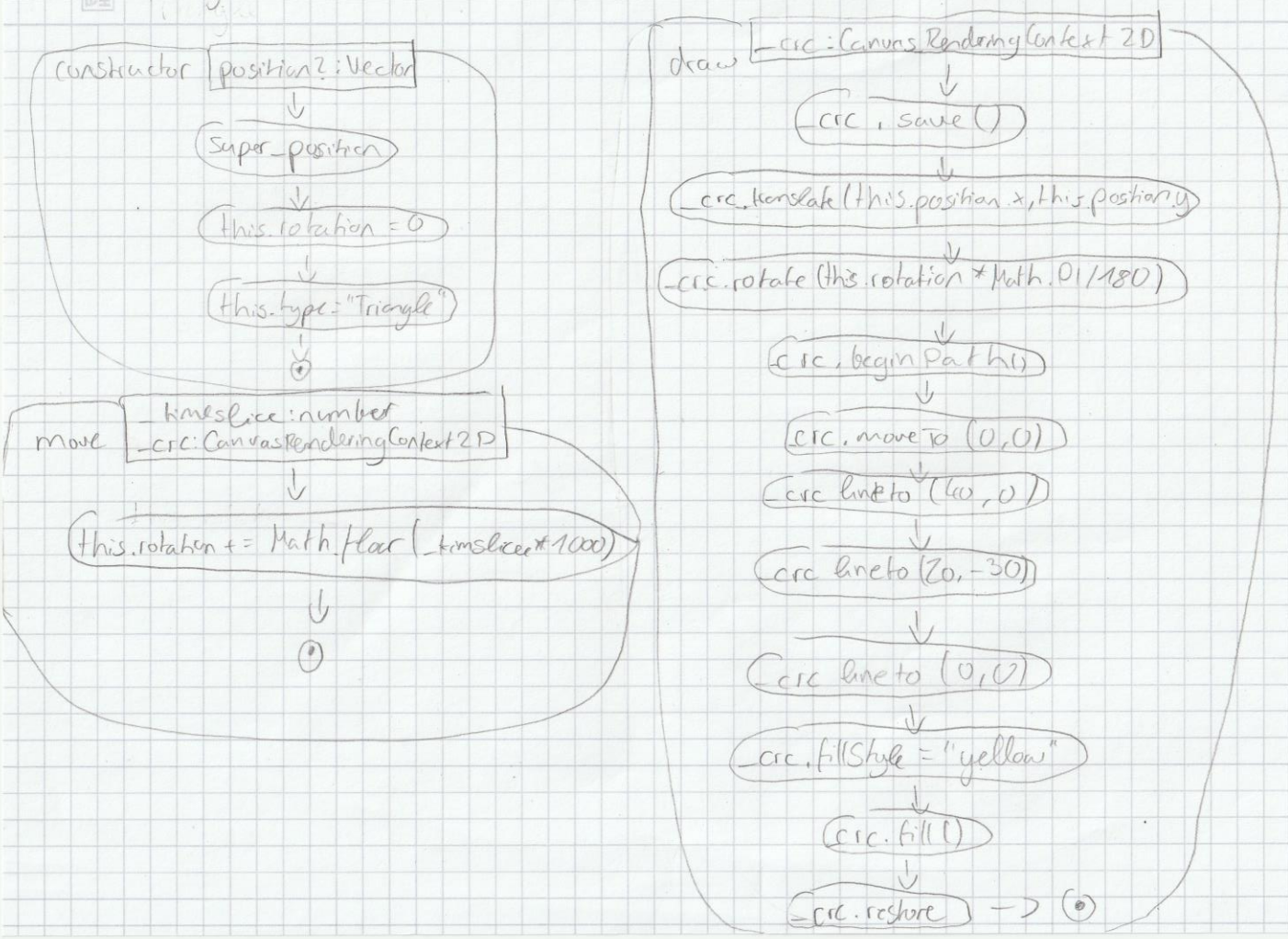
_crc.fillRect(0,0,60,60)

_crc.restore()

Circle

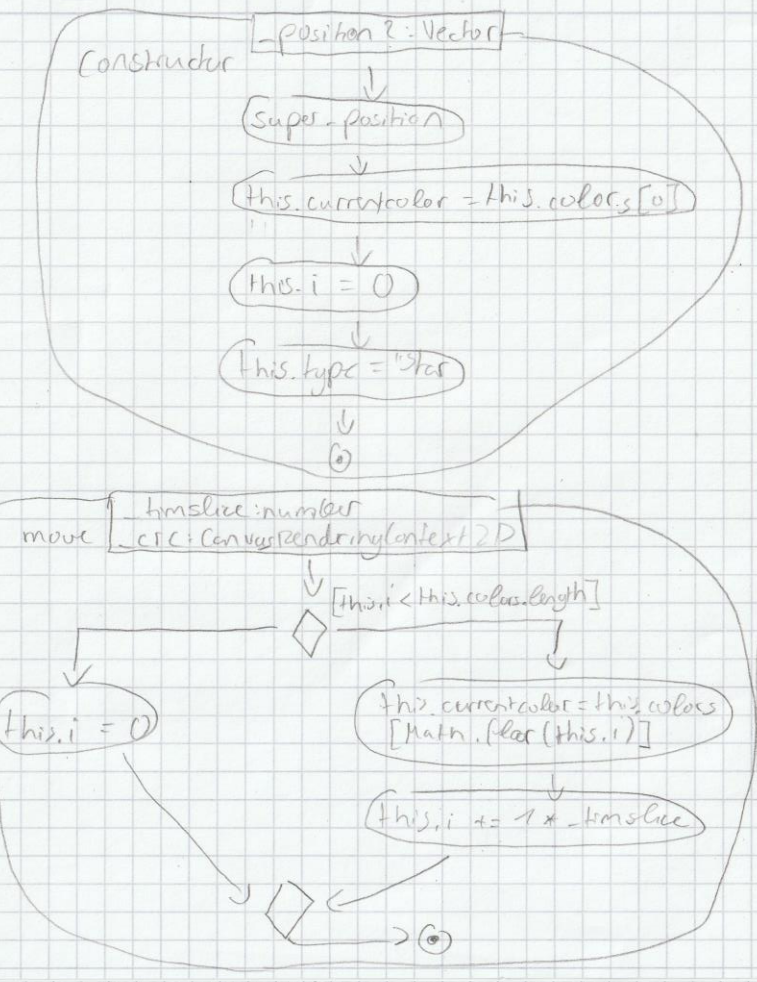


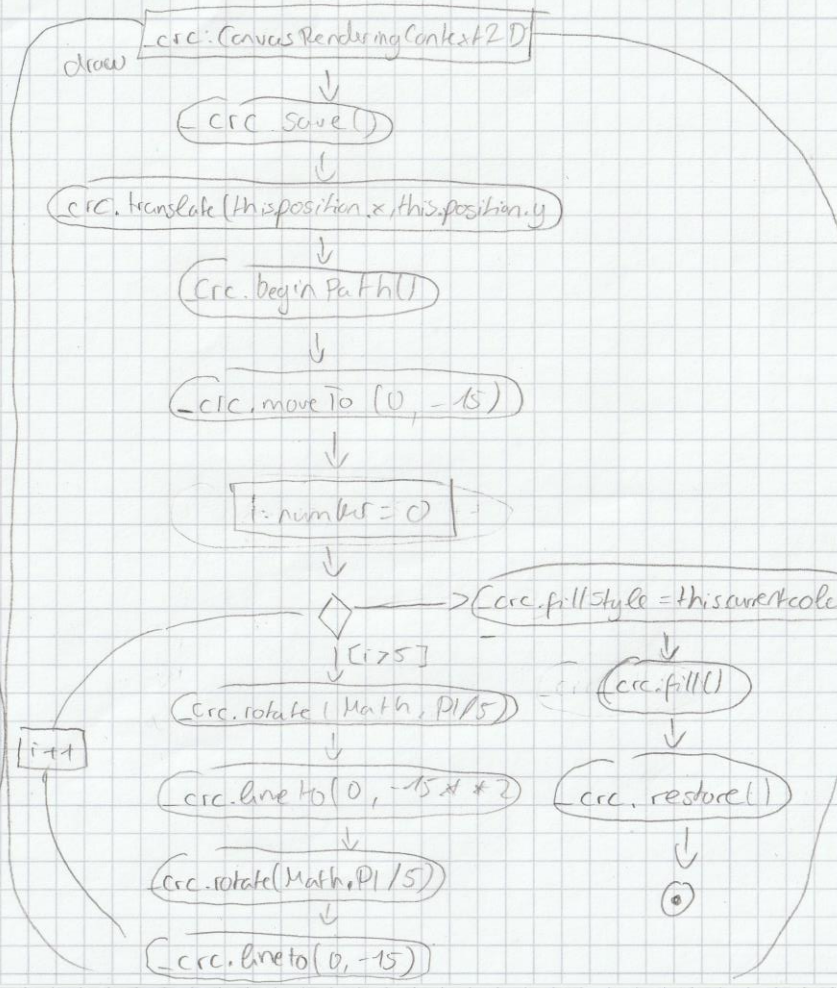
Triangle



BROWSER TU

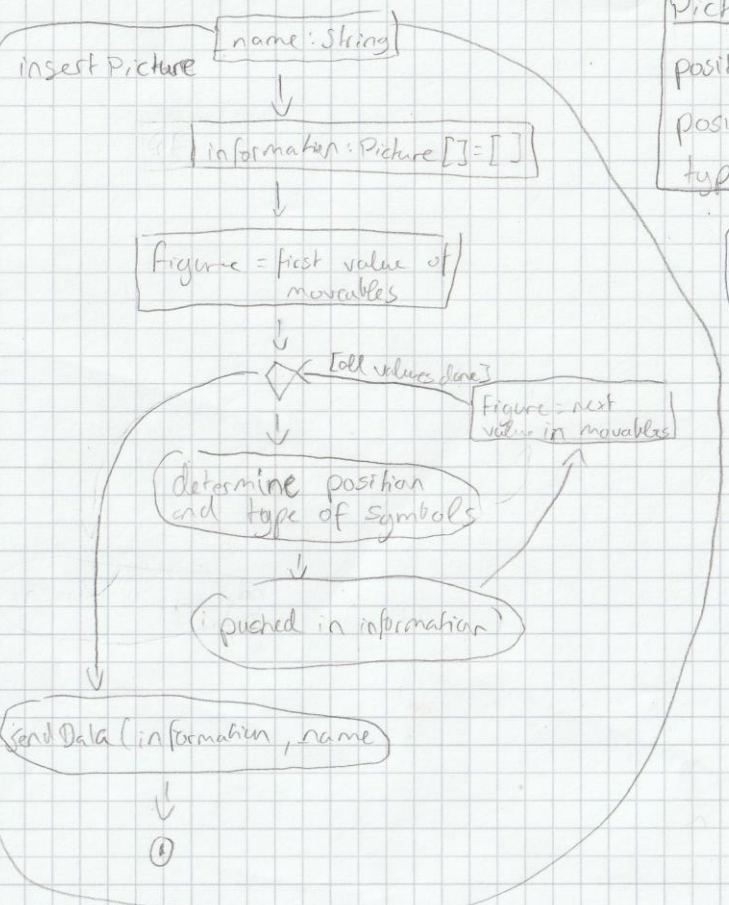
Star

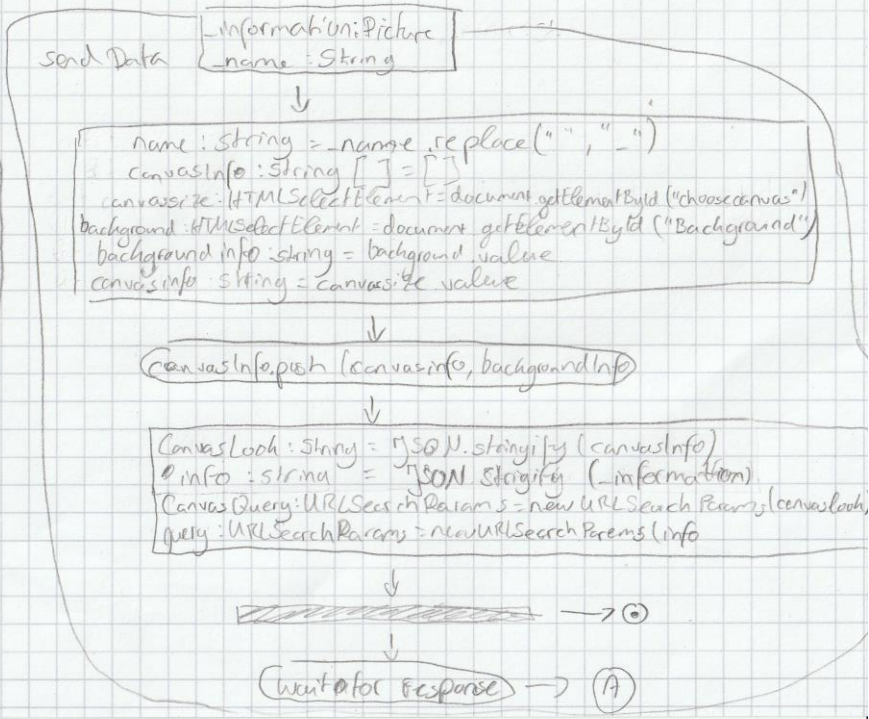




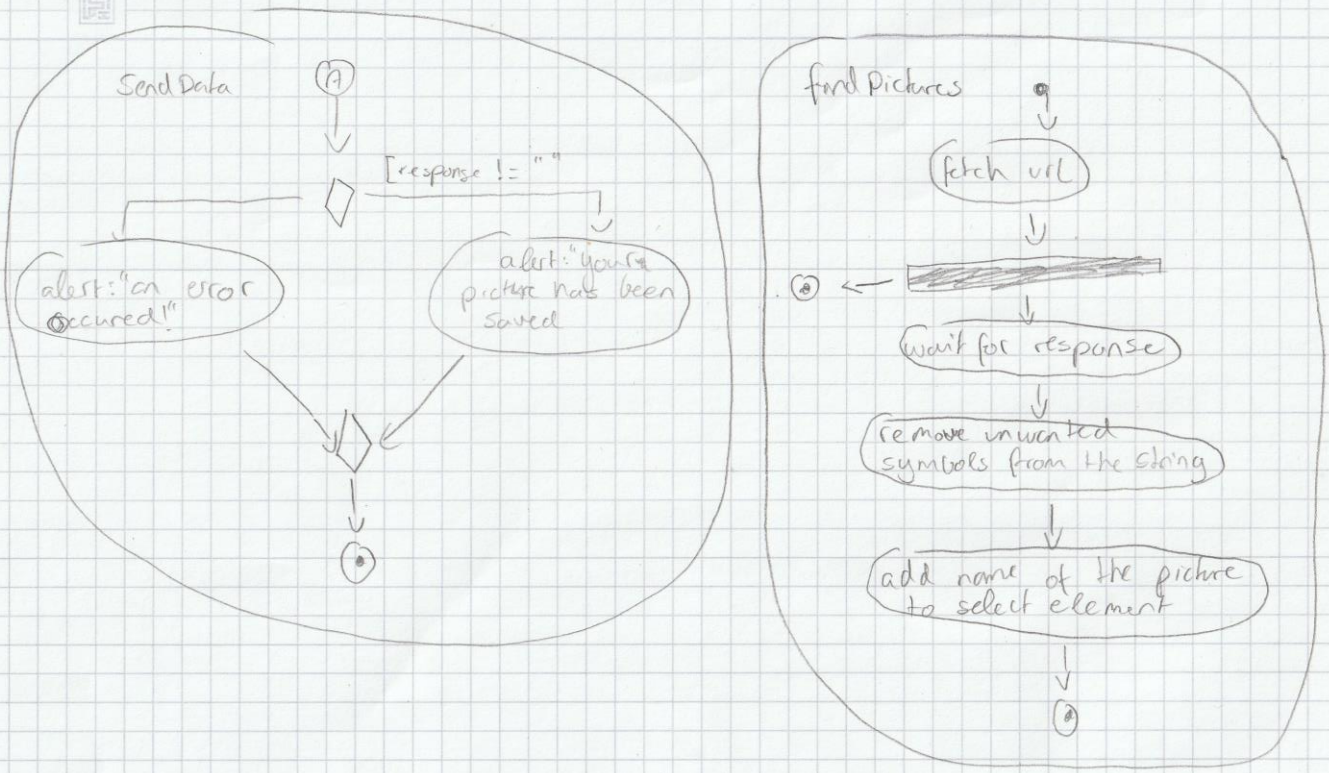
BROWSER TU

Server



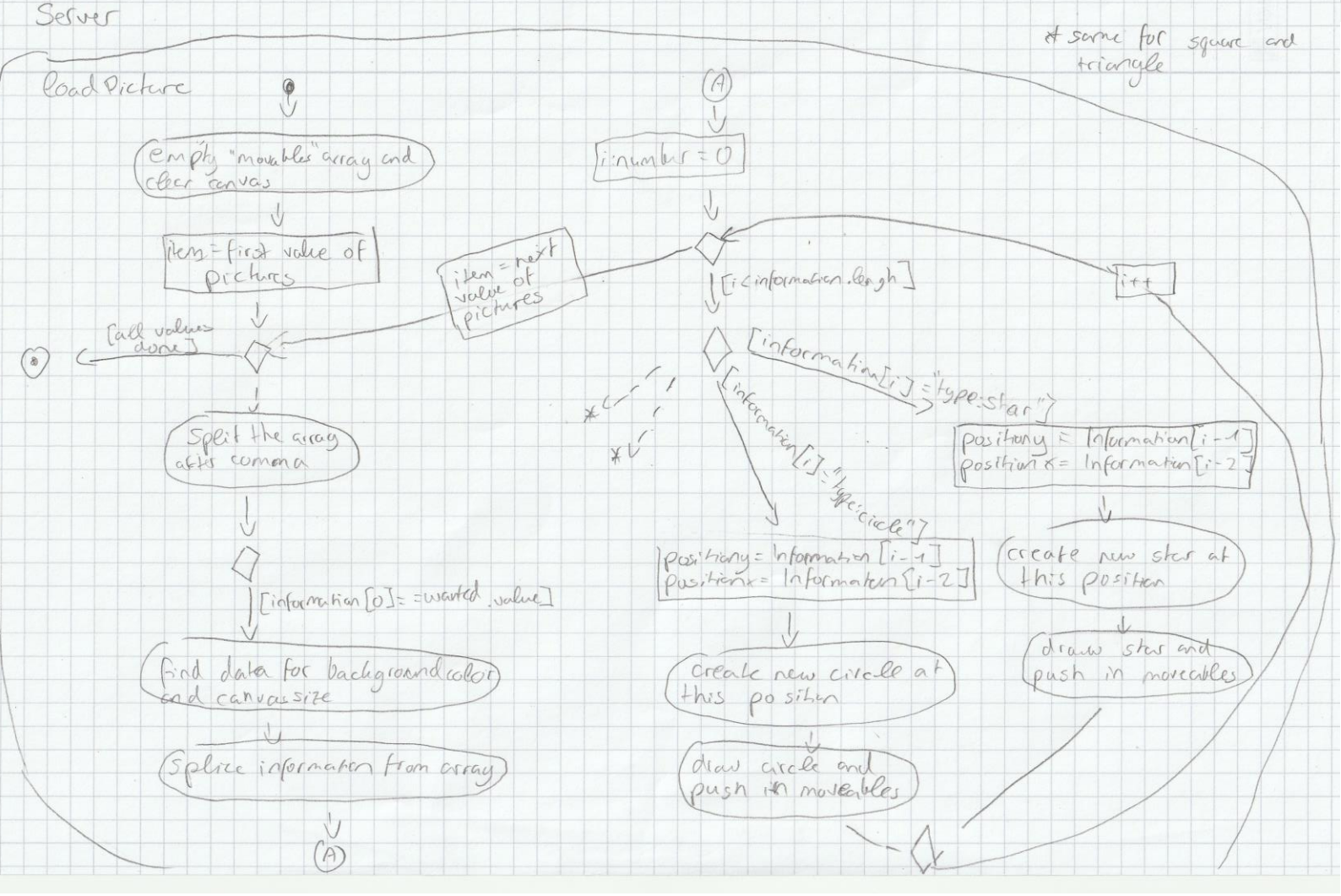


Sever


Send Data
A
[response != ""]
alert: "an error occured!"
alert: "your picture has been saved"

find Pictures
fetch url
wait for response
remove unwanted symbols from the string
add name of the picture to select element

Sever


* same for square and triangle
load Picture
empty "movables" array and clear canvas
item = first value of pictures
[all values done]
item = next value of pictures
Split the array at its comma
[information[0] == wanted value]
find data for background color and canvas size
Splice information from array
A
i:number = 0
[i < information.length]
i++
[information[i] = "type:star"]
[information[i] = "type:circle"]
positionY = Information[i-1]
positionX = Information[i-2]
create new star at this position
draw star and push in movables
create new circle at this position
draw circle and push in movables

Server 2

orders: MongoCollection
options: MongoClientOptions
MongoClient: MongoClient
allPictures: string[] = []

startServer(port to use)

connectToDatabase(database URL)

Request

handleRequest

Start Server — port: number|string

create Server

add request-listener

connect to Database — url: string

mongoClient = new MongoClient(url, options)

mongoClient.connect()

orders = mongoClient.db("Endabgabe").collection("Images")

showOrder — item: object

key = first key in item

key = next key in item

allPictures.push(key)