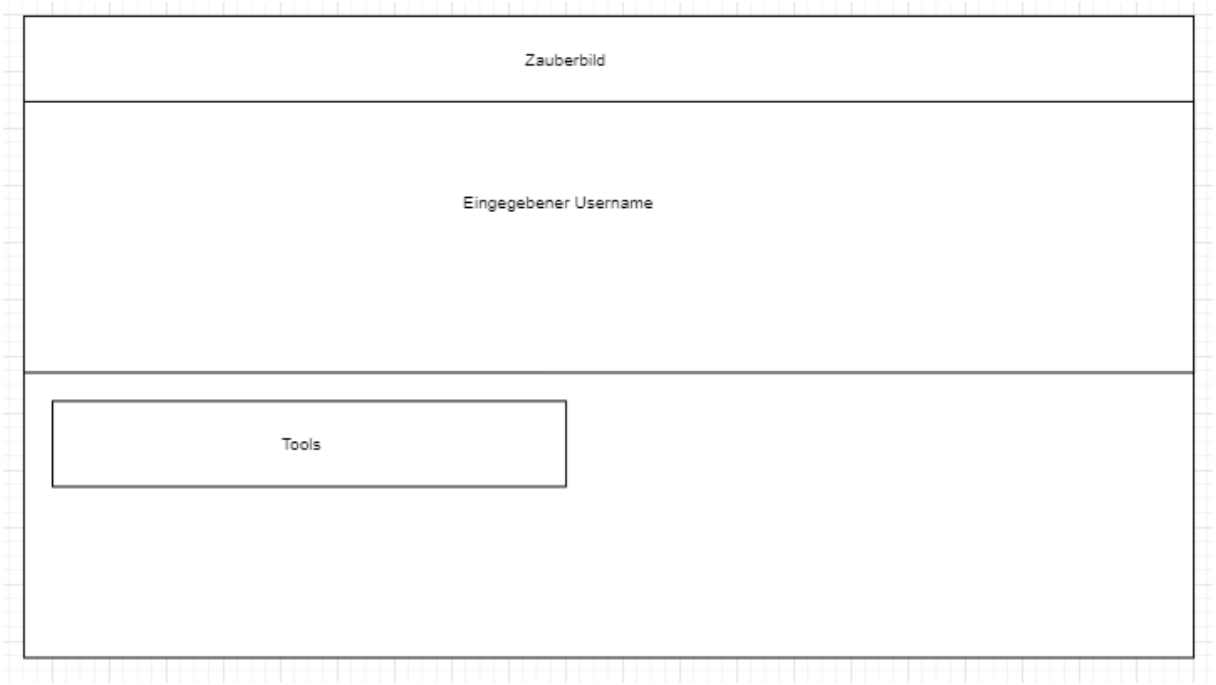
Lukas Blea

Auswahl der Plattform (PC, Mobil)

	PC	Mobil
Wo wird der Nutzer die Anwendung gebrauchen können?	Auf dem Computer muss man manchmal schnell etwas zeichnen können, um seinen Freunden im Internet visuell zu Präsentieren was sie ausdrücken möchten. Durch das schnelle und einfach zu bedienbare Animationstool kann man auch etwas spaß haben.	Es gibt unendlich viele Apps, welche dem Benutzer erlauben auf dem Handy etwas zu zeichnen. Durch den „bedingt“ kleinen Bildschirm fällt das sowieso schon etwas schwer, da der Finger nicht immer 100% erkannt wird. Der kleine Bildschirm (im Vergleich zum Monitor) war hier mein Auschlusskriterium
Usability	Am Computer hat man eine Maus und Tastatur, um mit der Anwendung zu interagieren. Auch eine stetige Internetverbindung ist gegeben, weshalb die Anwendung besser läuft, wenn man am PC sitzt	Nur seinen Finger. Keine konstante Internetverbindung

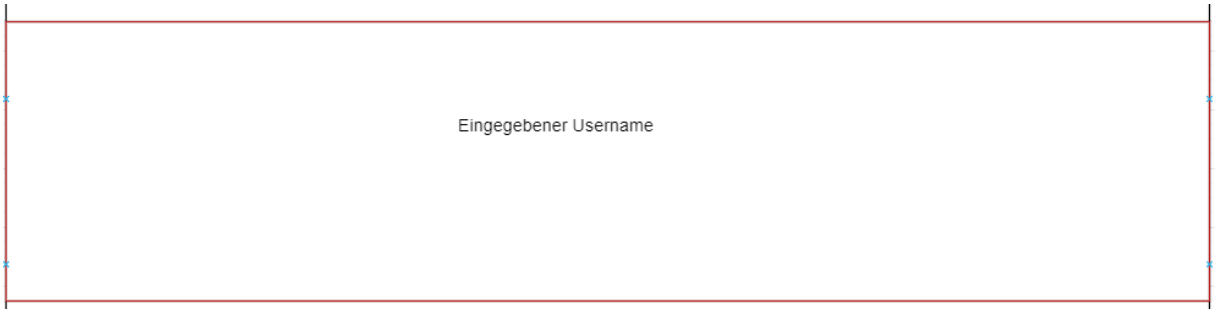
Lukas Blea

Was sieht der Nutzer, während er mit der Anwendung interagiert?



Der Nutzer soll stets seine Zeichenfläche sehen und die Tools, die ihm zur Verfügung stehen. Sein Username sollte nach der Eingabe auf der Zeichenfläche erscheinen, um ihm visuell darzustellen, wo er zeichnen/animieren kann und soll.

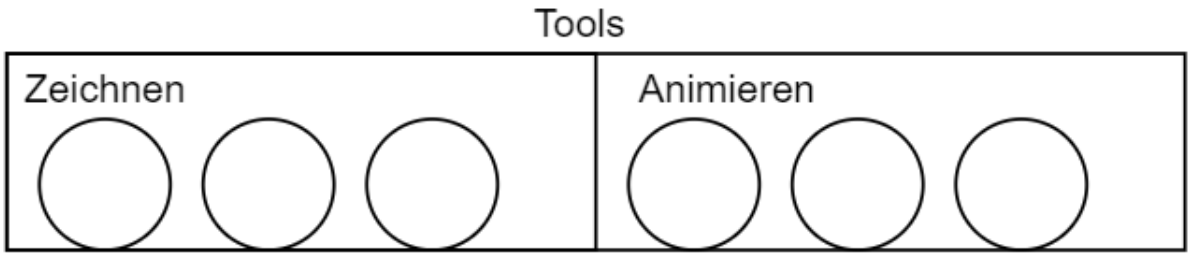
Durch einen Hover Effekt über dem Canvas, soll der Benutzer, jedes Mal, wenn er mit der Maus über den Canvas fährt visuell dargestellt bekommen, dass die Anwendung auf seine Maus reagiert und bereit zur Nutzung ist. (Im unteren Bild zu sehen)



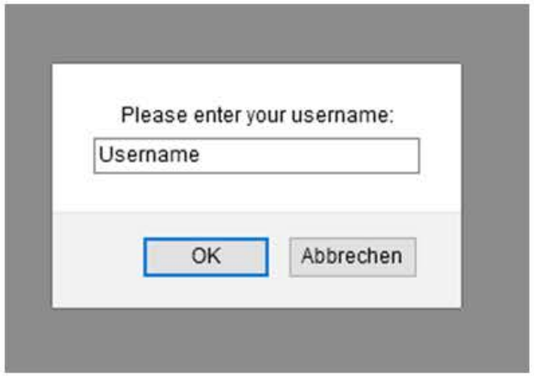
Lukas Blea

Auch die Maus soll von seinem Standard look zu einem anderen Cursor wechseln um die Anwendung möglichst ansprechend für den Nutzer zu machen. Jeder wird in der Lage sein zu verstehen was getan werden kann und soll, ohne eine ausführliche Anleitung.

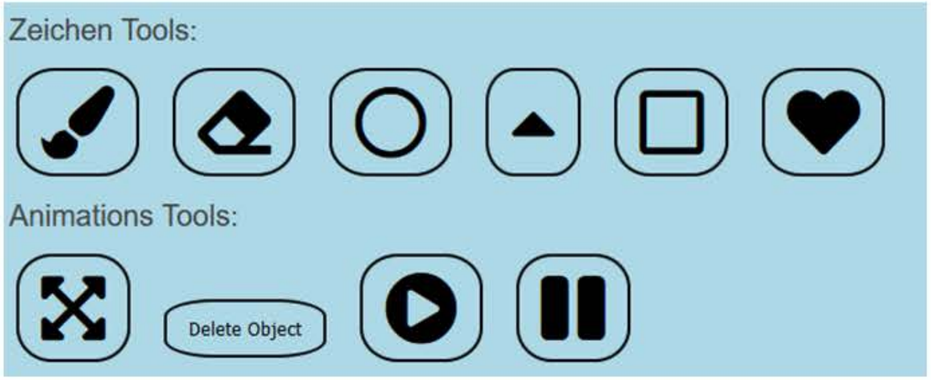
Die Tools werden geteilt in „Zeichnen“ und „Animation“. Somit hat der Nutzer die Auswahl ob er gerne etwas Zeichnen würde, oder lieber Animieren. Durch Buttons soll jeweils jedes Tool ausgewählt werden können.



1.

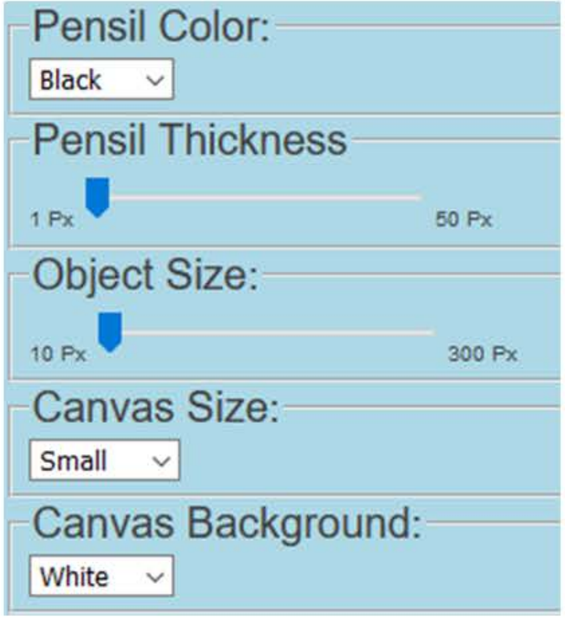

Please enter your username:
Username
OK
Abbrechen

2.


Zeichen Tools:
Animations Tools:
Delete Object

3.

4.


Pensil Color:
Black
Pensil Thickness
1 Px
50 Px
Object Size:
10 Px
300 Px
Canvas Size:
Small
Canvas Background:
White

5.

6.


Username

Lukas Blea

Anleitung zur Interaktion mit der Anwendung

1. Zuallererst soll der Nutzer seinen Usernamen festlegen, unter welchem das Bild dann auch in der Datenbank gespeichert wird. Wenn er sich dazu entscheidet nichts einzugeben wird der Default „User“ ausgegeben.

2. Im zweiten Schritt wählt der Nutzer sein Tool aus, mit welchem er auf dem Canvas zeichnen möchte. Hier hat er die Auswahl zwischen Zeichen Tools und Animation Tools. Beides gleichzeitig ist nicht möglich, jeweils nur eins von beidem. Die Formen wie Kreis, Dreieck, Viereck und Herz sind bei den Zeichentools, da sie sowohl für Zeichnen und Animation benutzt werden können. Durch das Klicken auf dem Canvas wird nun mit der Zeichenfläche interagiert. (bzw. click&drag)

3. Der Benutzer ist in der Lage seinen aktuellen Canvas zu Löschen oder zu Speichern. Durch die jeweiligen Symbole auf den Buttons wird das widergespiegelt.

4. Hier kann man seine Customisations auswählen. Man kann nicht viel falsch machen und die Beschriftung der Customisations helfen dabei zu verstehen welche Auswirkungen sie haben werden.

5. Durch „Play“ oder „Pause“ ist der Nutzer in der Lage seine Animation mit den Objekten zu starten oder auch zu pausieren. Natürlich müssen sich Objekte auf der Zeichenfläche befinden, um die Auswirkungen von „Play“ und „Pause“ zu sehen.

6. Sobald man mit der Maus über den Canvas fährt sieht der Nutzer einen Border der zeigt, dass auf die Maus reagiert wird. Alles ist bereit und es kann losgehen.

Anleitung der Installation der Anwendung, bzw. Heroku und MongoDB

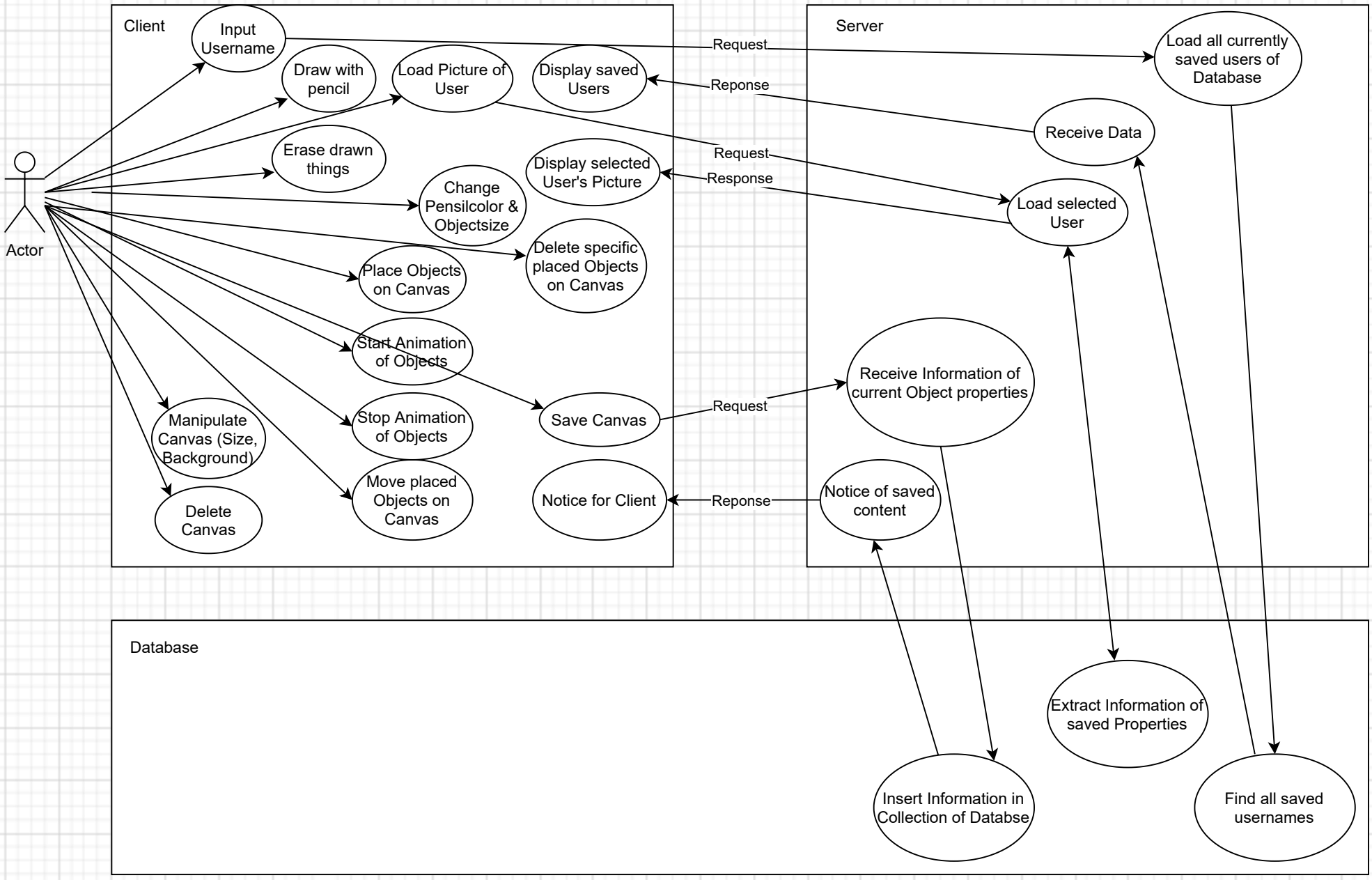
An sich braucht er zur Installation der Anwendung nichts, um zu funktionieren. Man öffnet einfach den Browser, bzw. clickt auf den Link und dann funktioniert alles. Weder Bilder noch Audiodateien sind im Projekt gegeben. Es läuft alles übers Internet.

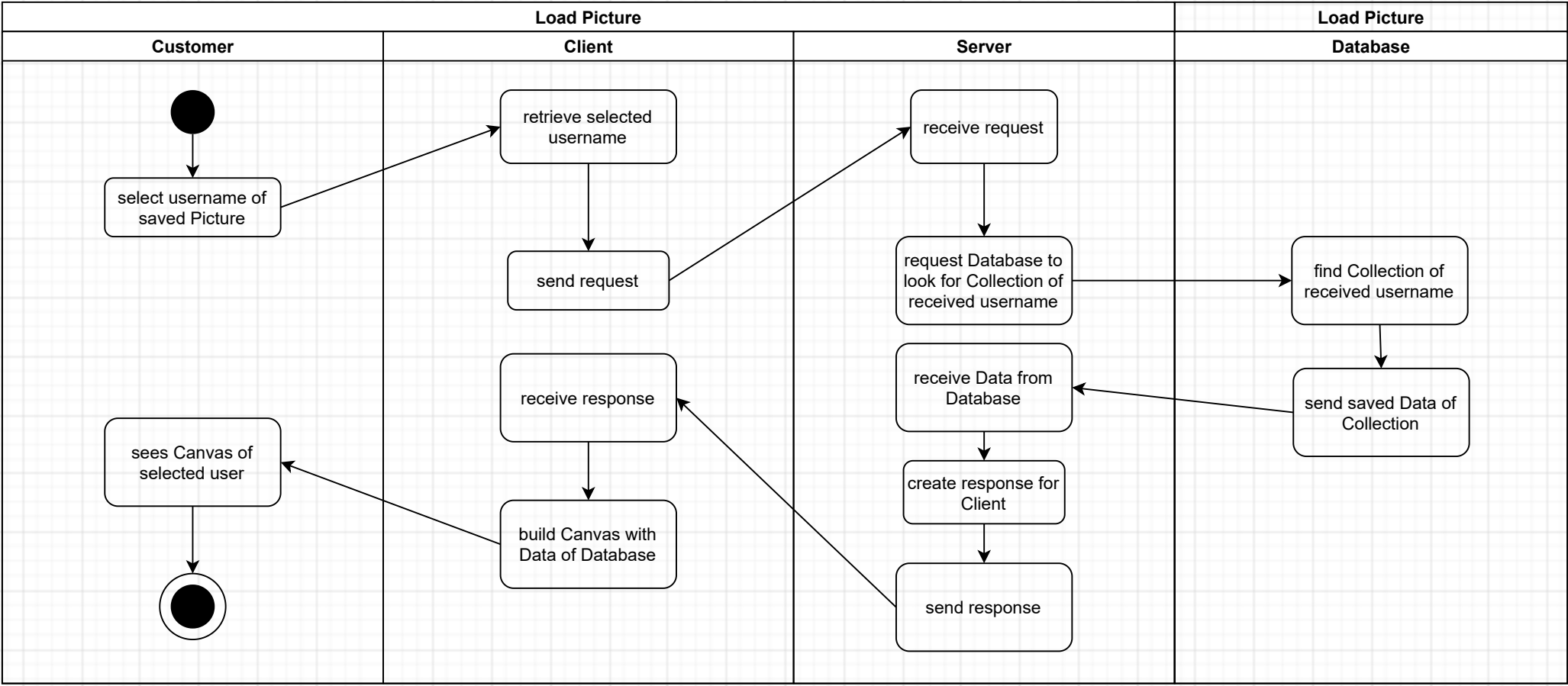
Um Mongo zu installieren erstellt man sich ein Konto, dann einen Cluster und in den Collections kann man dann seine eigenen Daten anlegen bzw. durch eine Anwendung Daten speichern lassen.

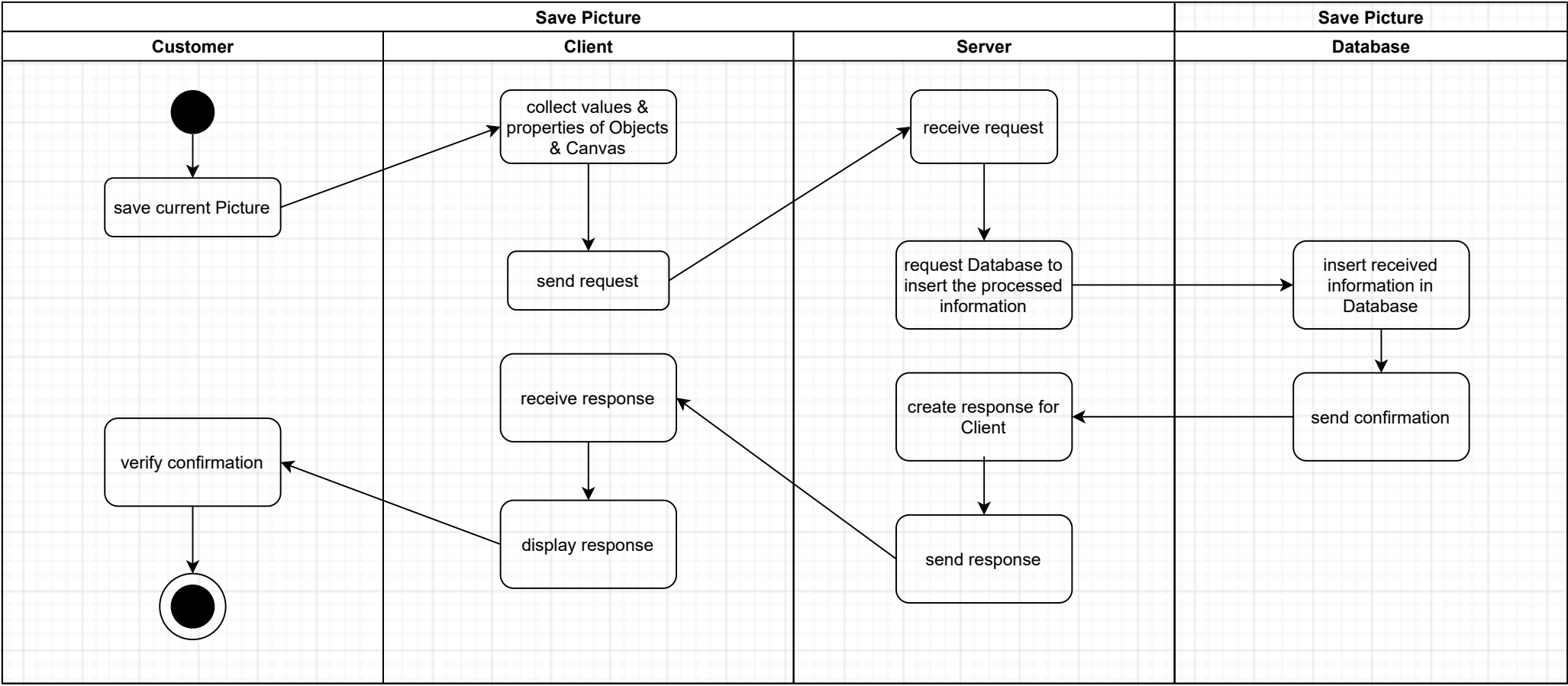
Wenn man nun auf „Connect“ drückt, muss man auf „Connect your application“ und hier den Link kopieren der gegeben wird und in seine Server.ts einsetzen. Hier ist es wichtig zu beachten, dass man einen User festlegt, und ein entsprechendes Passwort. Beides muss dann in die gegebene String eingesetzt werden. Wichtig ist es auch, einen Whitelist Zugriff mit der IP-Adresse 0.0.0.0/0 zu erlauben um Verbindungen von allen Netzwerken zuzulassen.

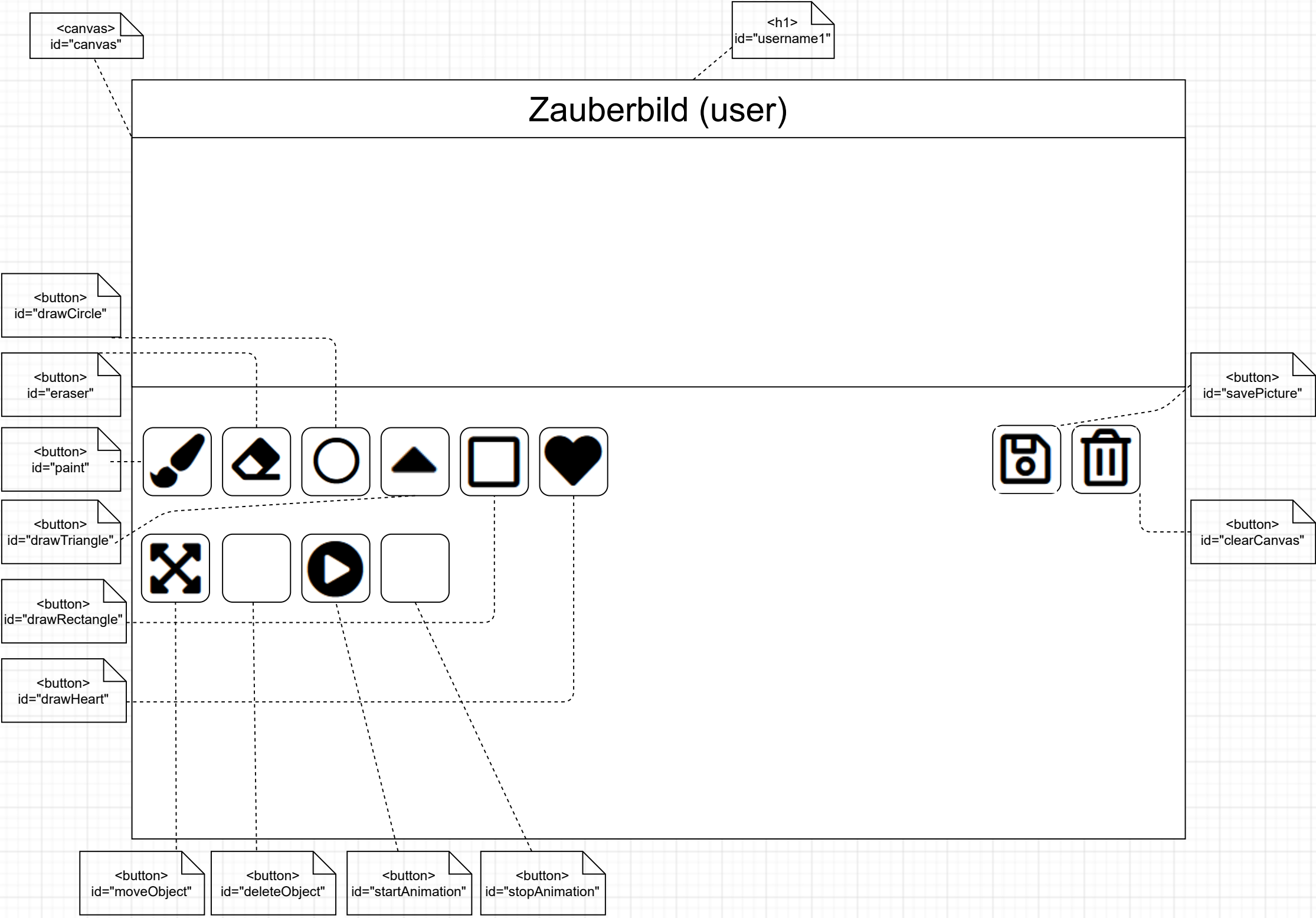
Für Heroku ist es ein bisschen anders. Hier erstellt man auch sein eigenes Profil und wählt als primary language „Node.js“ aus. Danach erstellt man eine App und verbindet seinen GitHub-Repository mit der App. Somit ist gewährleistet, dass die beiden miteinander kommunizieren können, nachdem man seine Anwendung auf GitHub hochläd.

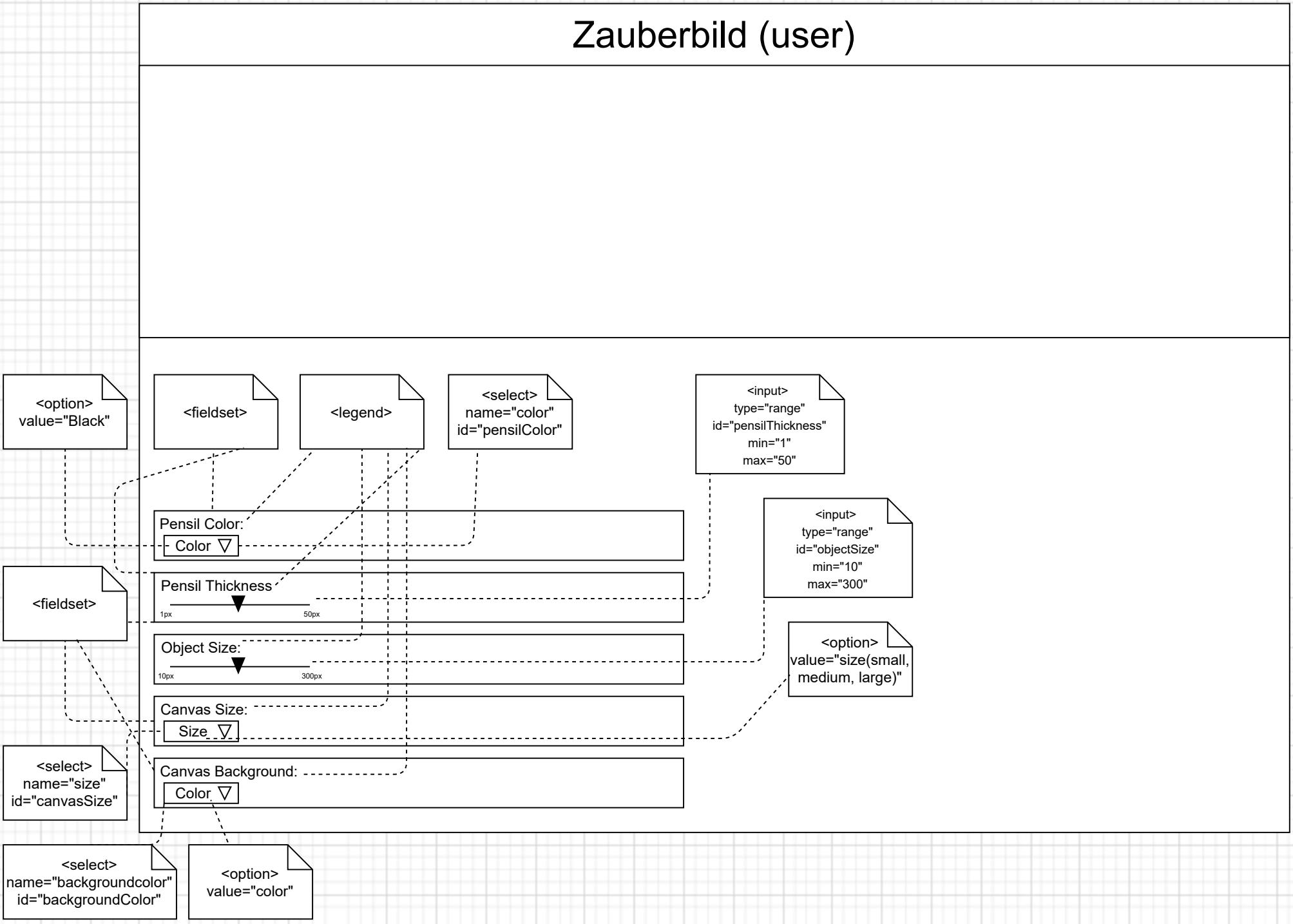
WICHTIG: Die package.json muss in der obersten Ordnerstrucktur des Repositorys liegen und mit dem richtigen Pfad auf die Server.js verweisen. Sonst klappt das ganze Spielchen nicht.

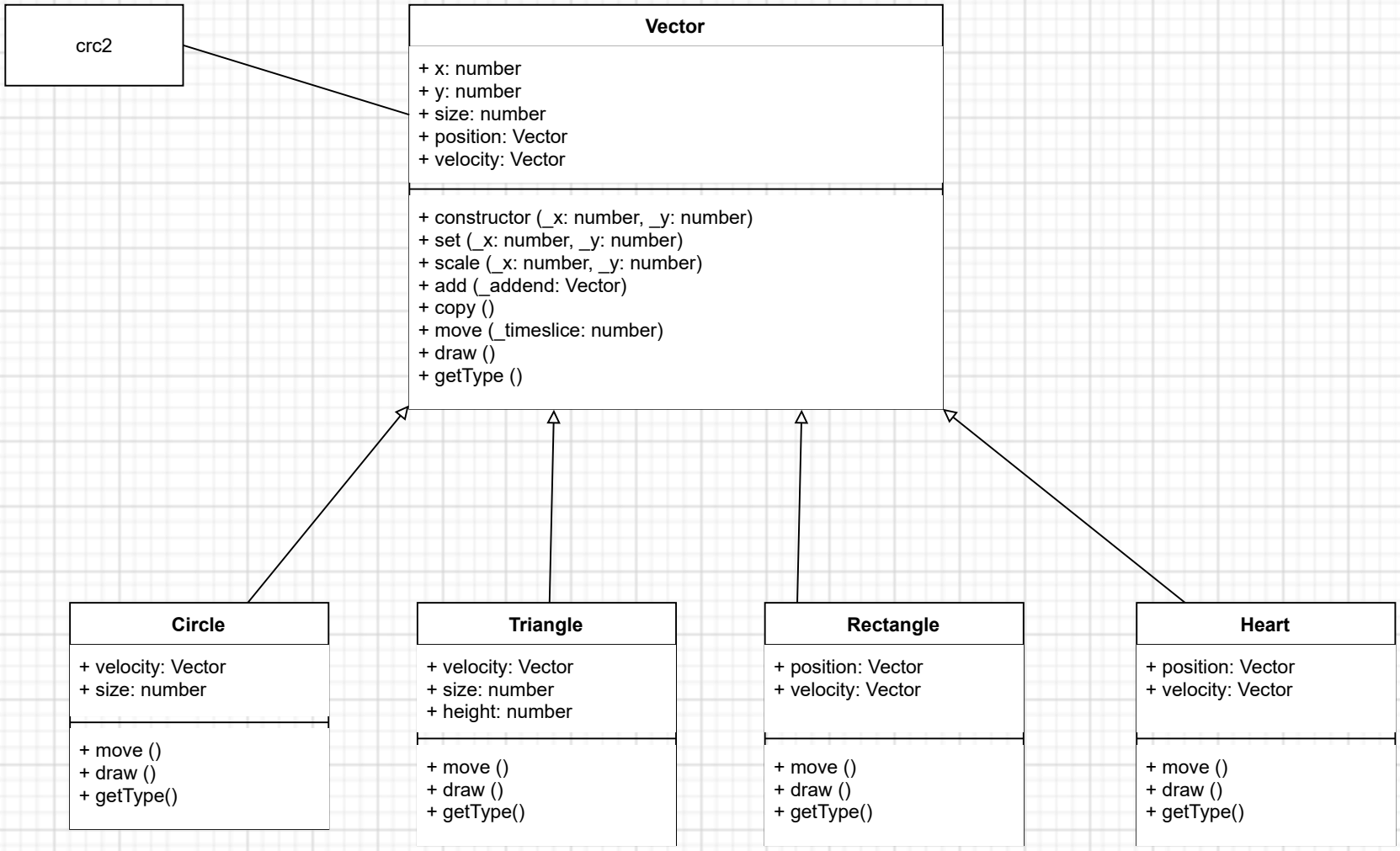

Client
Input Username
Draw with pencil
Load Picture of User
Display saved Users
Erase drawn things
Display selected User's Picture
Change Pensilcolor & Objectsize
Delete specific placed Objects on Canvas
Place Objects on Canvas
Start Animation of Objects
Stop Animation of Objects
Save Canvas
Manipulate Canvas (Size, Background)
Move placed Objects on Canvas
Delete Canvas
Notice for Client
Actor
Request
Reponse
Request
Response
Request
Reponse
Server
Load all currently saved users of Database
Receive Data
Load selected User
Receive Information of current Object properties
Notice of saved content
Database
Extract Information of saved Properties
Insert Information in Collection of Databse
Find all saved usernames


Load Picture
Load Picture
Customer
Client
Server
Database
select username of saved Picture
retrieve selected username
send request
receive request
request Database to look for Collection of received username
find Collection of received username
send saved Data of Collection
receive Data from Database
create response for Client
send response
receive response
build Canvas with Data of Database
sees Canvas of selected user

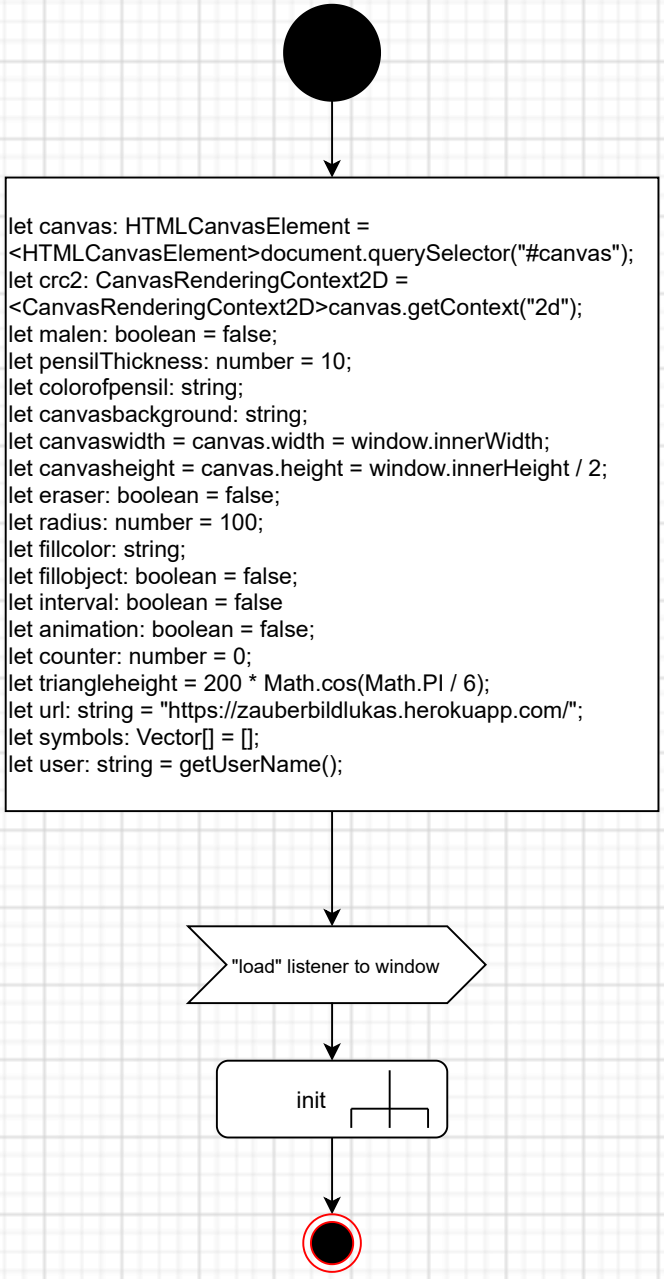

Save Picture
Save Picture
Customer
Client
Server
Database
save current Picture
collect values & properties of Objects & Canvas
send request
receive request
request Database to insert the processed information
insert received information in Database
send confirmation
create response for Client
send response
receive response
display response
verify confirmation


<canvas>
id="canvas"
<h1>
id="username1"
Zauberbild (user)
<button>
id="drawCircle"
<button>
id="eraser"
<button>
id="paint"
<button>
id="drawTriangle"
<button>
id="drawRectangle"
<button>
id="drawHeart"
<button>
id="savePicture"
<button>
id="clearCanvas"
<button>
id="moveObject"
<button>
id="deleteObject"
<button>
id="startAnimation"
<button>
id="stopAnimation"

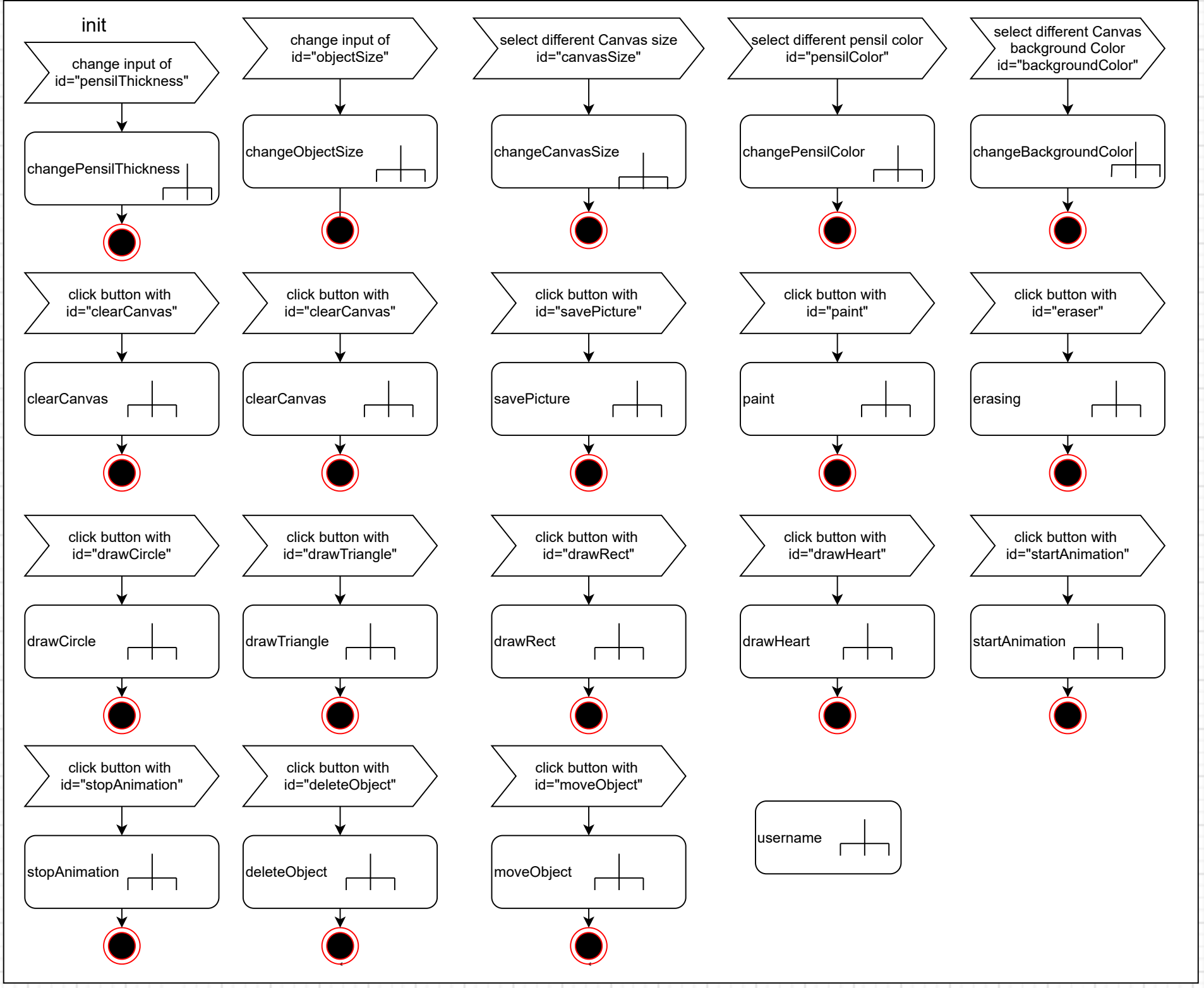

Zauberbild (user)
<option>
value="Black"
<fieldset>
<legend>
<select>
name="color"
id="pensilColor"
<input>
type="range"
id="pensilThickness"
min="1"
max="50"
Pensil Color:
Color
<input>
type="range"
id="objectSize"
min="10"
max="300"
<fieldset>
Pensil Thickness
1px
50px
Object Size:
10px
300px
<option>
value="size(small,
medium, large)"
Canvas Size:
Size
<select>
name="size"
id="canvasSize"
Canvas Background:
Color
<select>
name="backgroundcolor"
id="backgroundColor"
<option>
value="color"


crc2
Vector
+ x: number
+ y: number
+ size: number
+ position: Vector
+ velocity: Vector
+ constructor (_x: number, _y: number)
+ set (_x: number, _y: number)
+ scale (_x: number, _y: number)
+ add (_addend: Vector)
+ copy ()
+ move (_timeslice: number)
+ draw ()
+ getType ()
Circle
+ velocity: Vector
+ size: number
+ move ()
+ draw ()
+ getType()
Triangle
+ velocity: Vector
+ size: number
+ height: number
+ move ()
+ draw ()
+ getType()
Rectangle
+ position: Vector
+ velocity: Vector
+ move ()
+ draw ()
+ getType()
Heart
+ position: Vector
+ velocity: Vector
+ move ()
+ draw ()
+ getType()

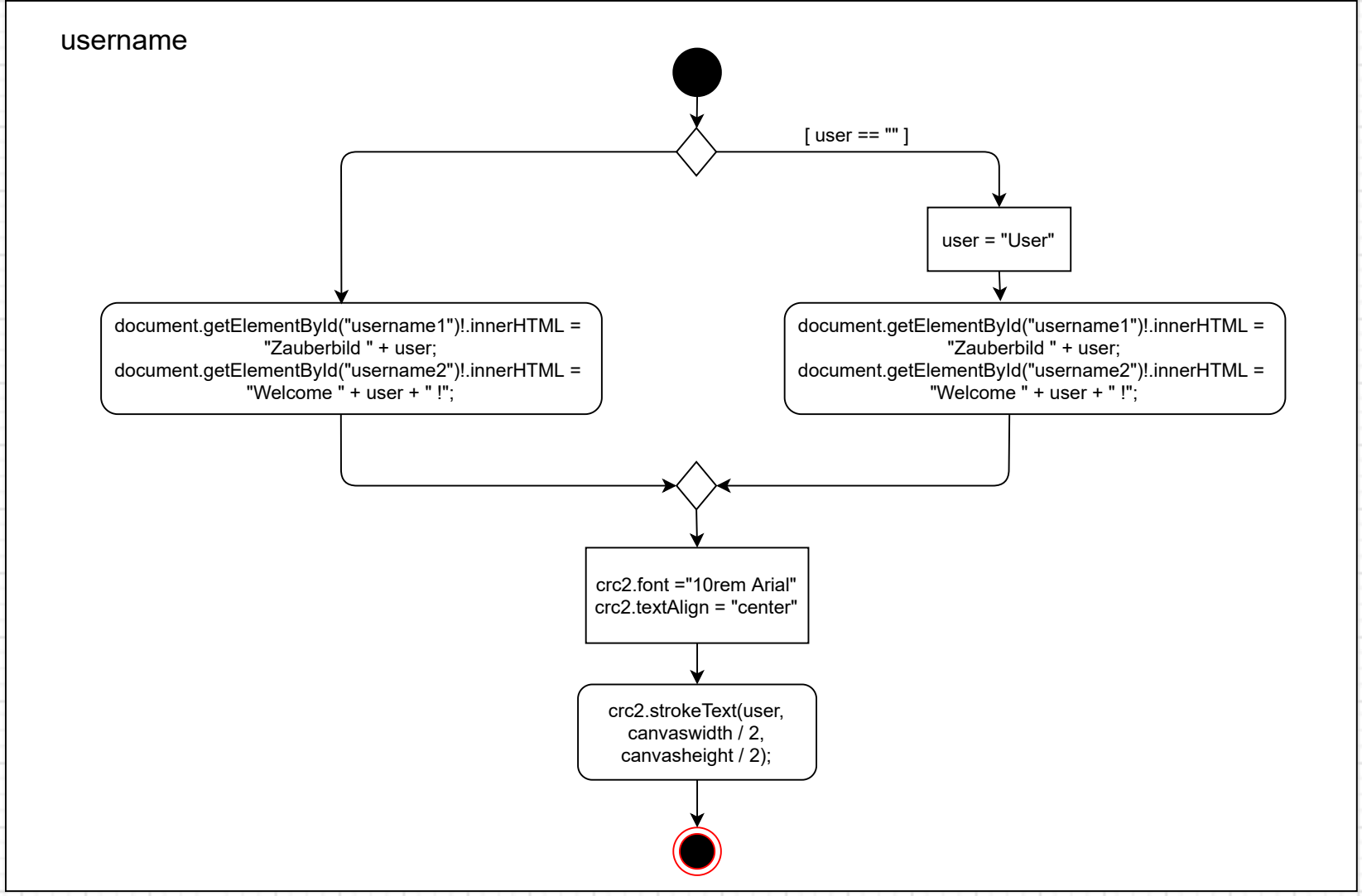
AD Canvas

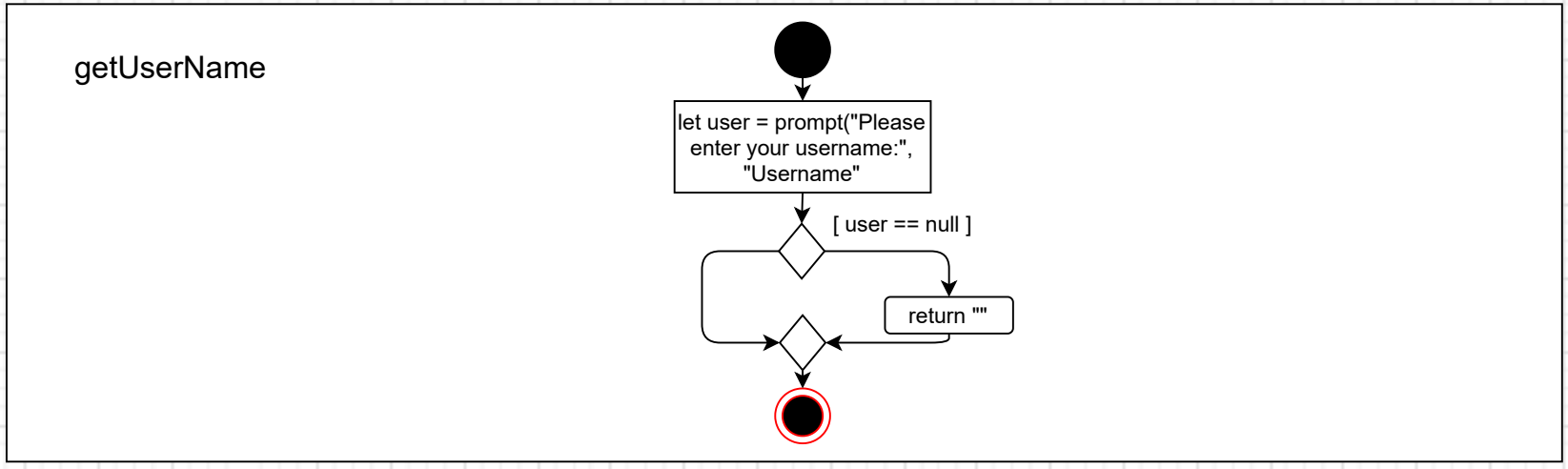

let canvas: HTMLCanvasElement =
<HTMLCanvasElement>document.querySelector("#canvas");
let crc2: CanvasRenderingContext2D =
<CanvasRenderingContext2D>canvas.getContext("2d");
let malen: boolean = false;
let pensilThickness: number = 10;
let colorofpensil: string;
let canvasbackground: string;
let canvaswidth = canvas.width = window.innerWidth;
let canvasheight = canvas.height = window.innerHeight / 2;
let eraser: boolean = false;
let radius: number = 100;
let fillcolor: string;
let fillobject: boolean = false;
let interval: boolean = false
let animation: boolean = false;
let counter: number = 0;
let triangleheight = 200 * Math.cos(Math.PI / 6);
let url: string = "https://zauberbildlukas.herokuapp.com/";
let symbols: Vector[] = [];
let user: string = getUserName();
"load" listener to window
init

AD Canvas


init
change input of id="pensilThickness"
changePensilThickness
change input of id="objectSize"
changeObjectSize
select different Canvas size id="canvasSize"
changeCanvasSize
select different pensil color id="pensilColor"
changePensilColor
select different Canvas background Color id="backgroundColor"
changeBackgroundColor
click button with id="clearCanvas"
clearCanvas
click button with id="clearCanvas"
clearCanvas
click button with id="savePicture"
savePicture
click button with id="paint"
paint
click button with id="eraser"
erasing
click button with id="drawCircle"
drawCircle
click button with id="drawTriangle"
drawTriangle
click button with id="drawRect"
drawRect
click button with id="drawHeart"
drawHeart
click button with id="startAnimation"
startAnimation
click button with id="stopAnimation"
stopAnimation
click button with id="deleteObject"
deleteObject
click button with id="moveObject"
moveObject
username

AD Canvas


username
[user == ""]
user = "User"
document.getElementById("username1")!.innerHTML = "Zauberbild " + user;
document.getElementById("username2")!.innerHTML = "Welcome " + user + " !";
document.getElementById("username1")!.innerHTML = "Zauberbild " + user;
document.getElementById("username2")!.innerHTML = "Welcome " + user + " !";
crc2.font ="10rem Arial"
crc2.textAlign = "center"
crc2.strokeText(user, canvaswidth / 2, canvasheight / 2);

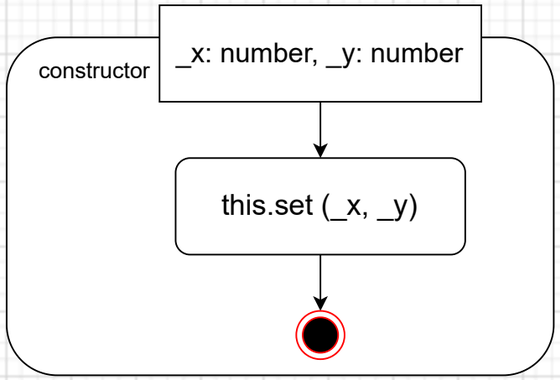

getUserName
let user = prompt("Please enter your username:", "Username"
[user == null]
return ""

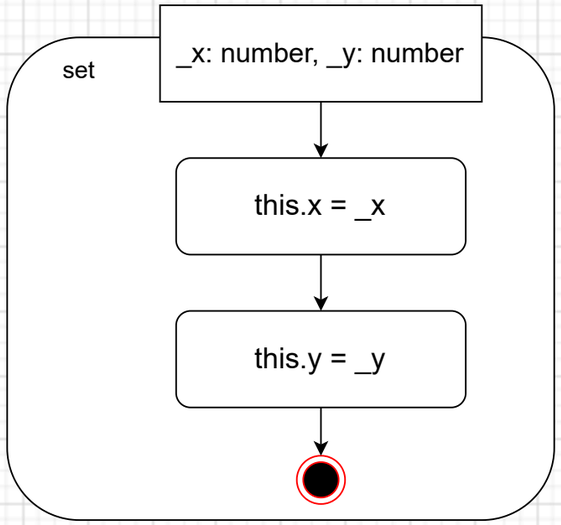
AD Vector

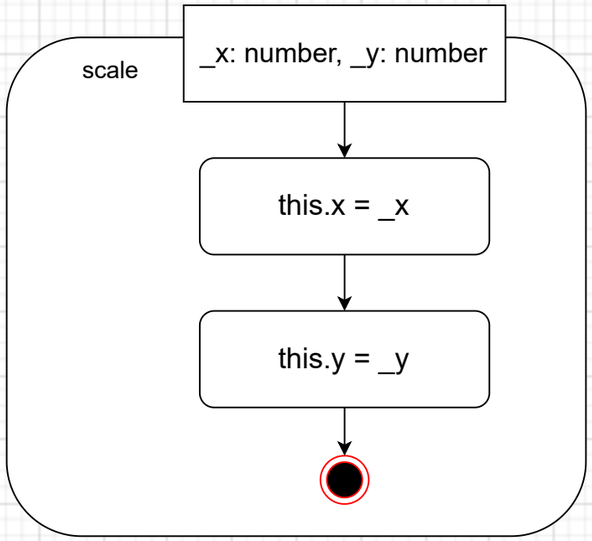
Vector

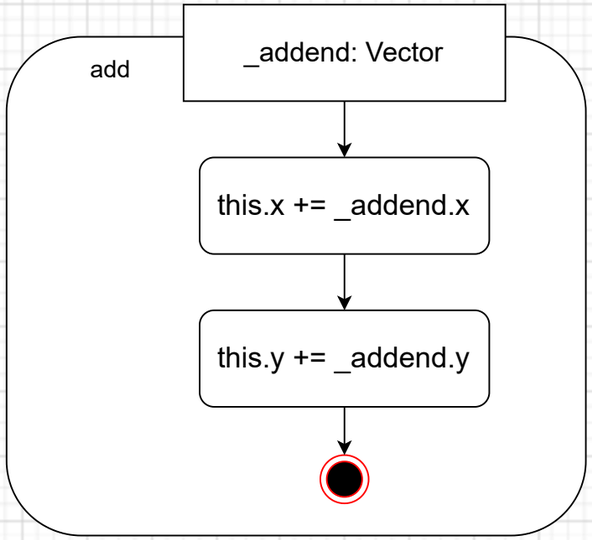
+ x: number
+ y: number
+ size: number
+ position: Vector
+ velocity: Vector

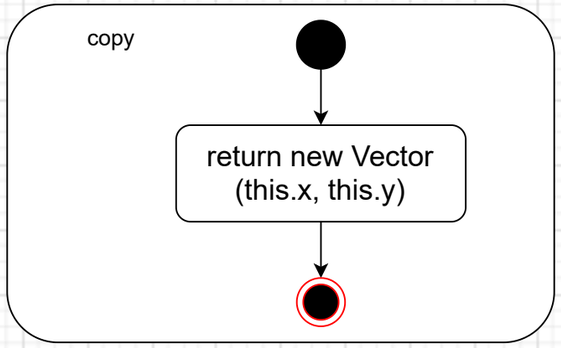
+ constructor (_x: number, _y: number)
+ set (_x: number, _y: number)
+ scale (_factor)
+ add (_addend: Vector)
+ copy ()
+ move (_timeslice)
+ draw ()
+ getType()

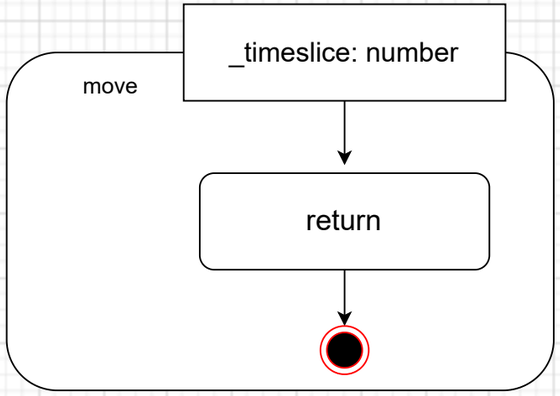


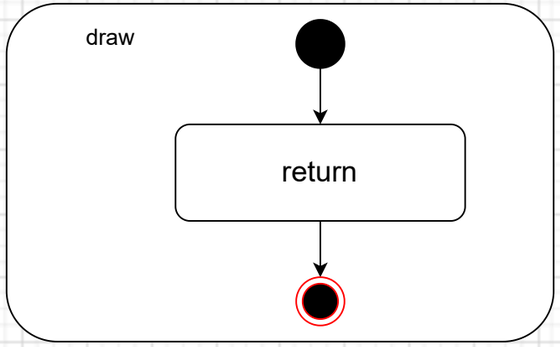


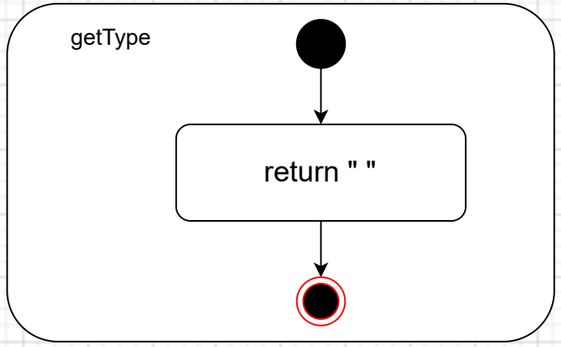






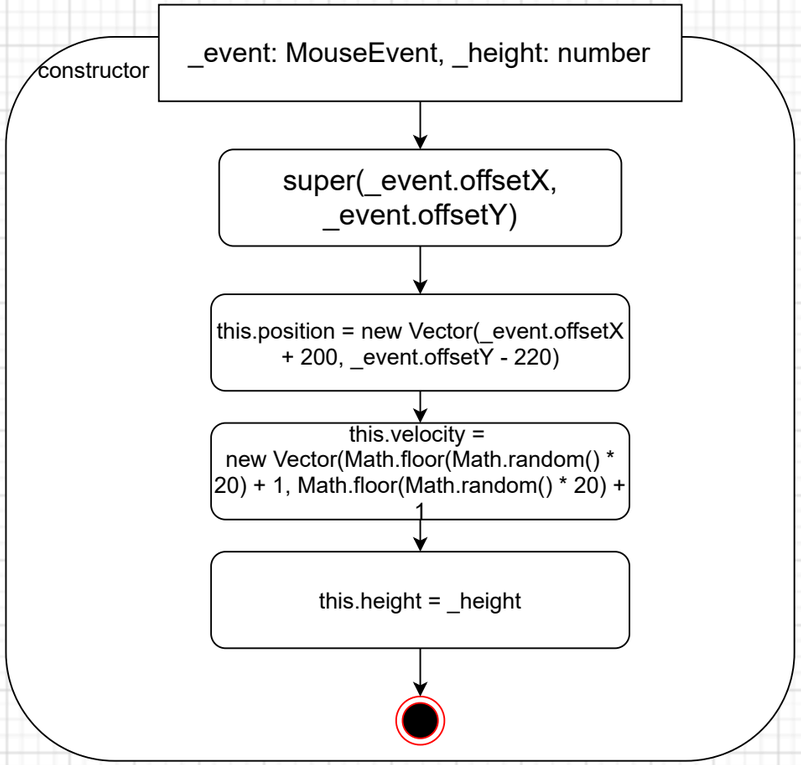


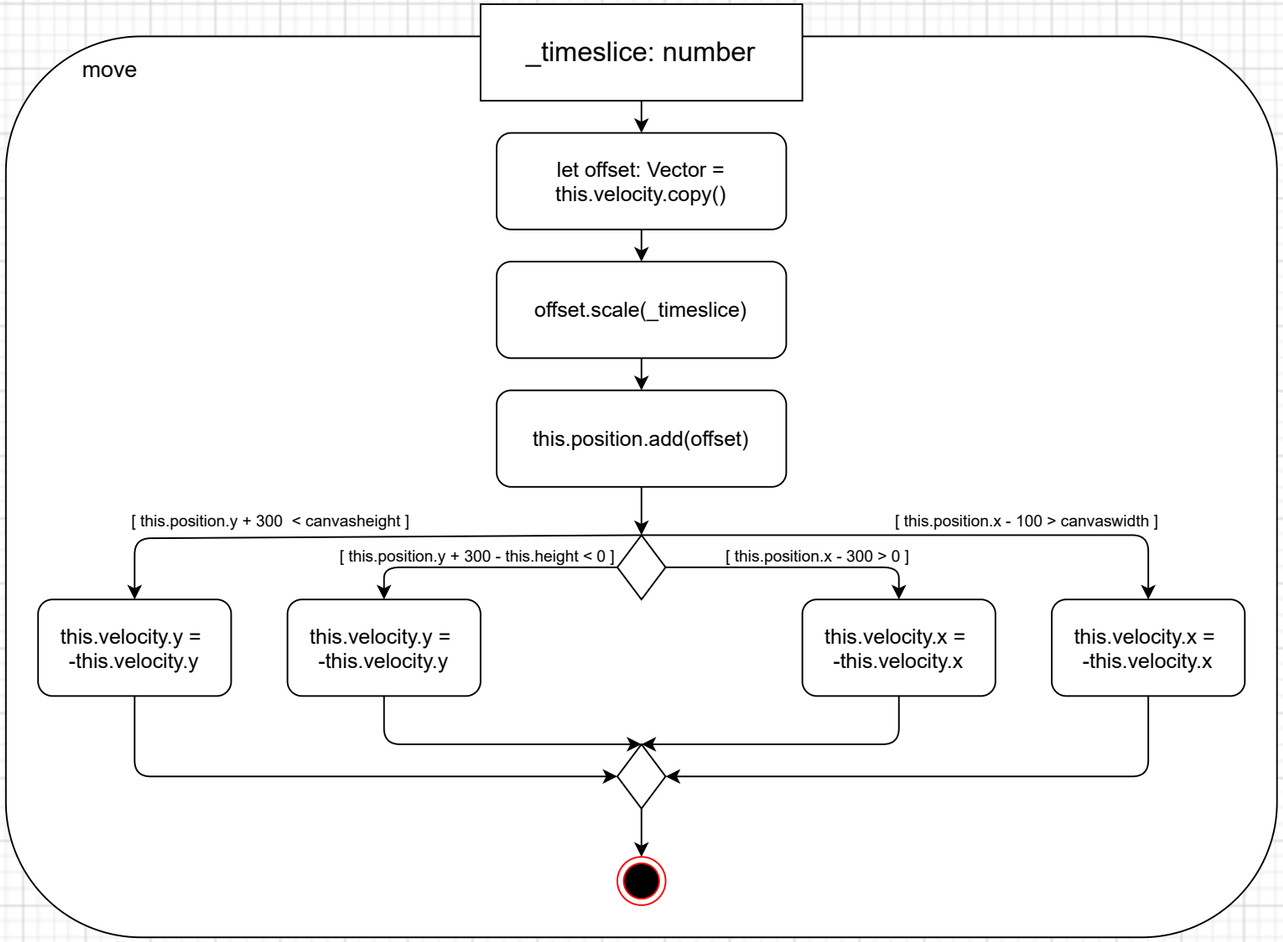




AD Triangle

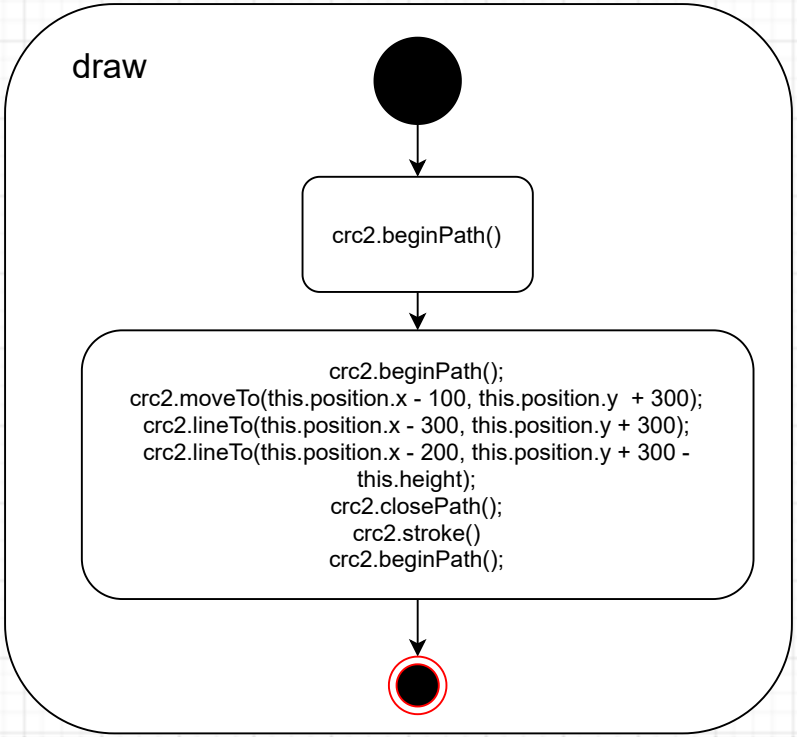
Triangle
+ velocity: Vector + size: number + height: number
+ move () + draw () + getType()

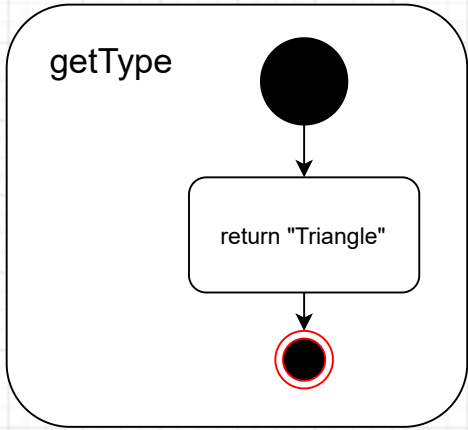




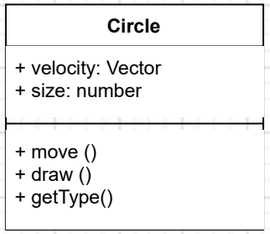
AD Triangle

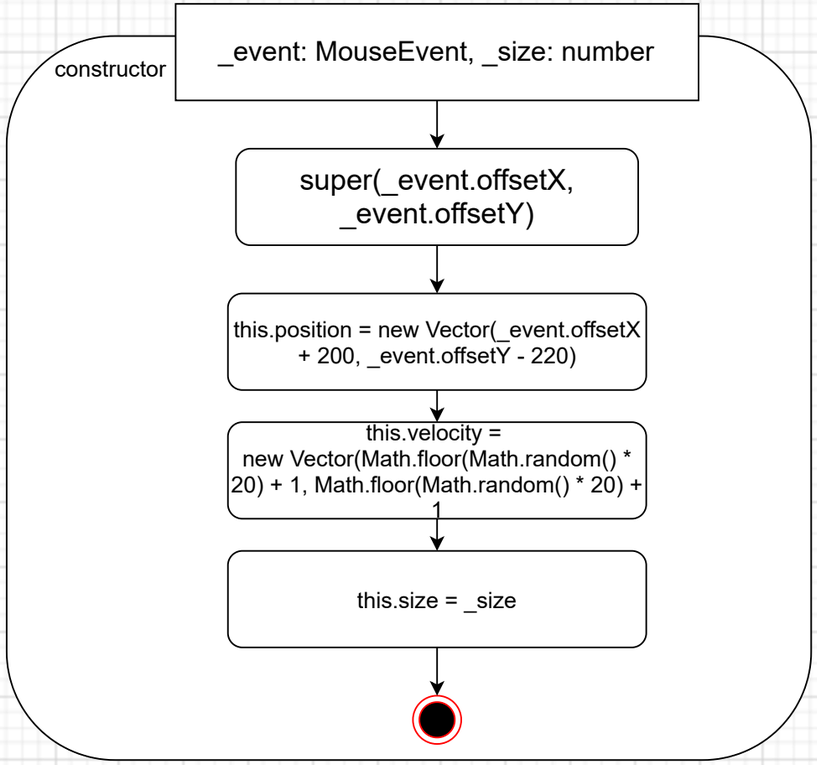
Triangle
+ velocity: Vector + size: number + height: number
+ move () + draw () + getType()

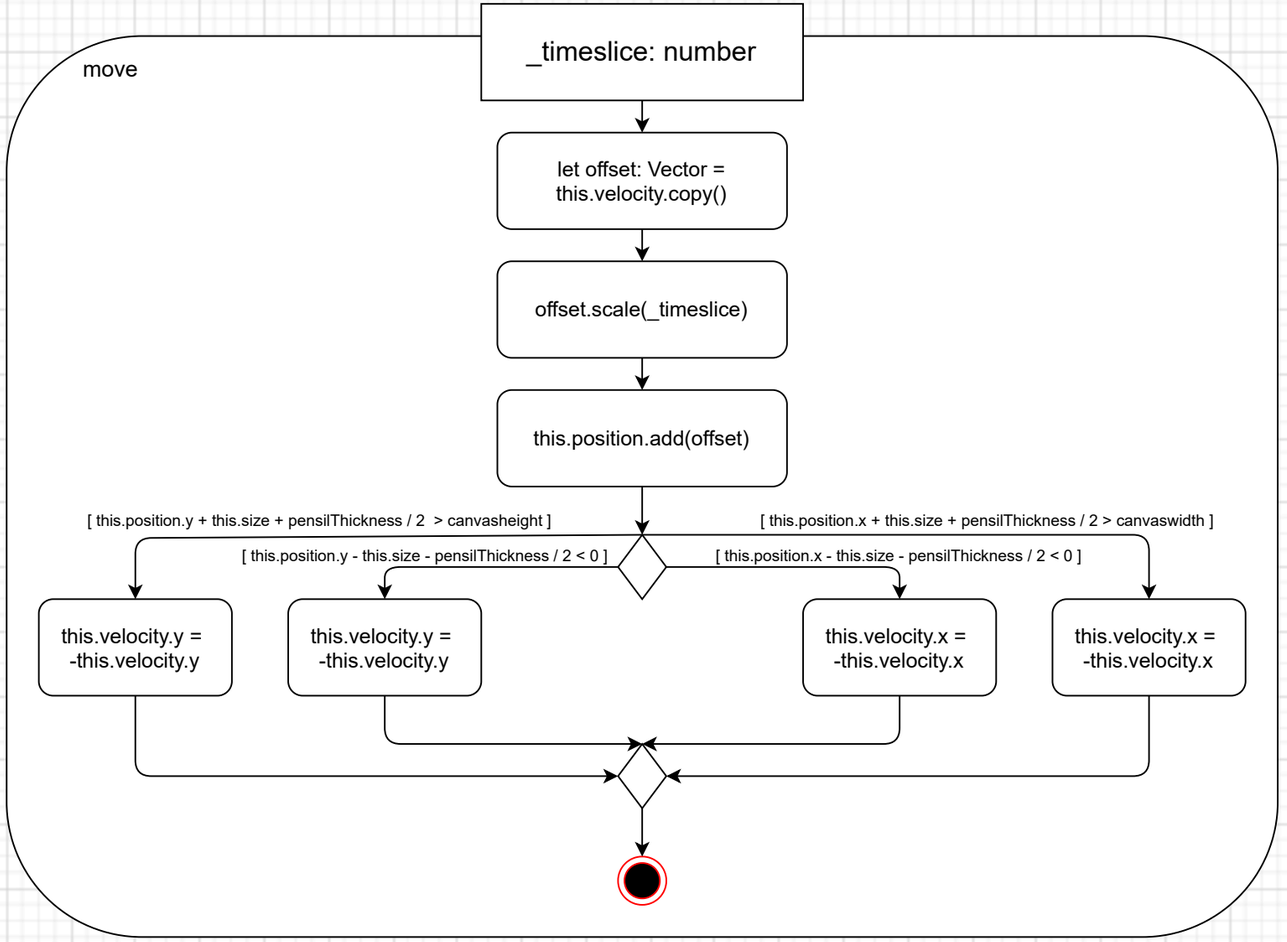


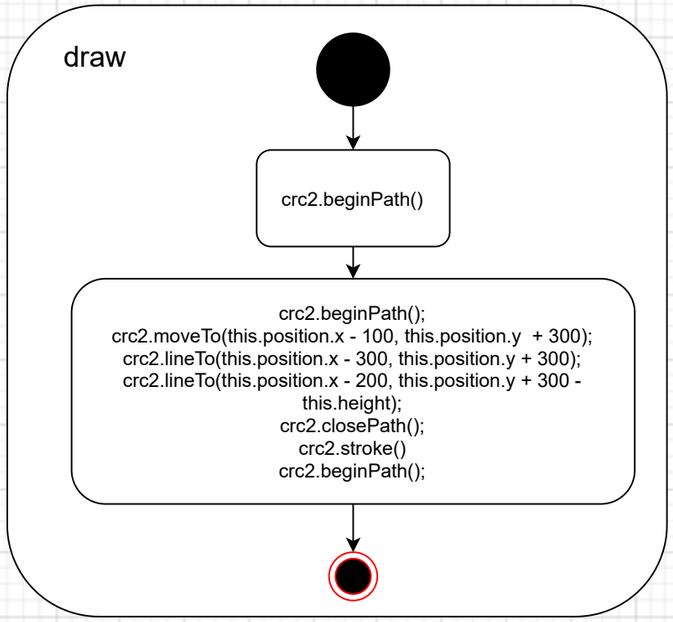


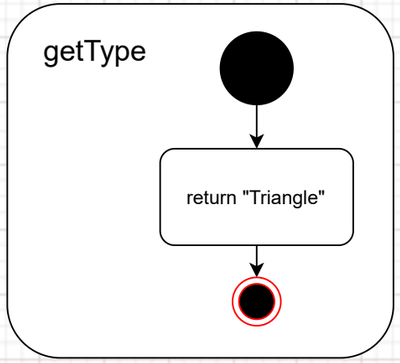
AD Circle


Circle
+ velocity: Vector
+ size: number
+ move ()
+ draw ()
+ getType()


constructor
_event: MouseEvent, _size: number
super(_event.offsetX, _event.offsetY)
this.position = new Vector(_event.offsetX + 200, _event.offsetY - 220)
this.velocity = new Vector(Math.floor(Math.random() * 20) + 1, Math.floor(Math.random() * 20) + 1
this.size = _size


move
_timeslice: number
let offset: Vector = this.velocity.copy()
offset.scale(_timeslice)
this.position.add(offset)
[this.position.y + this.size + pensilThickness / 2 > canvasheight]
[this.position.y - this.size - pensilThickness / 2 < 0]
[this.position.x + this.size + pensilThickness / 2 > canvaswidth]
[this.position.x - this.size - pensilThickness / 2 < 0]
this.velocity.y = -this.velocity.y
this.velocity.y = -this.velocity.y
this.velocity.x = -this.velocity.x
this.velocity.x = -this.velocity.x


draw
crc2.beginPath()
crc2.beginPath();
crc2.moveTo(this.position.x - 100, this.position.y + 300);
crc2.lineTo(this.position.x - 300, this.position.y + 300);
crc2.lineTo(this.position.x - 200, this.position.y + 300 - this.height);
crc2.closePath();
crc2.stroke()
crc2.beginPath();

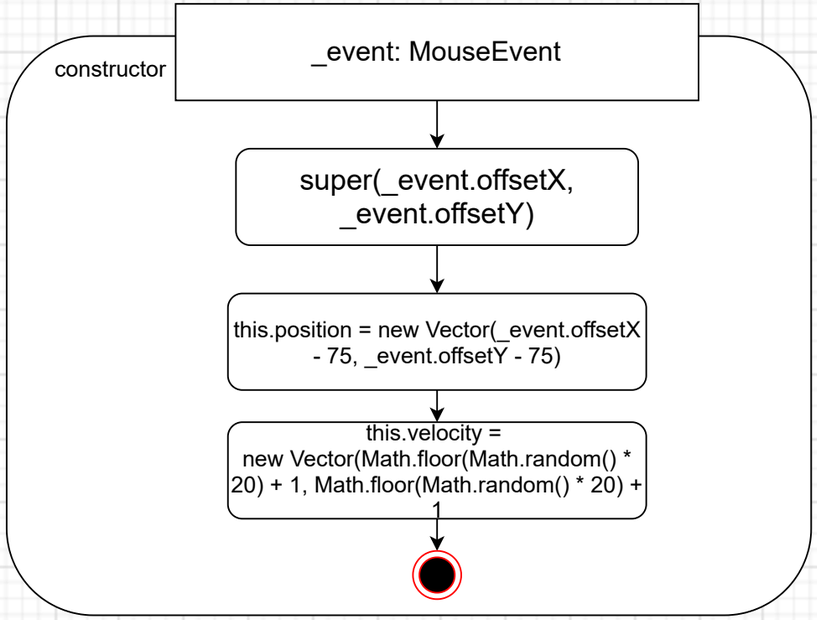

getType
return "Triangle"

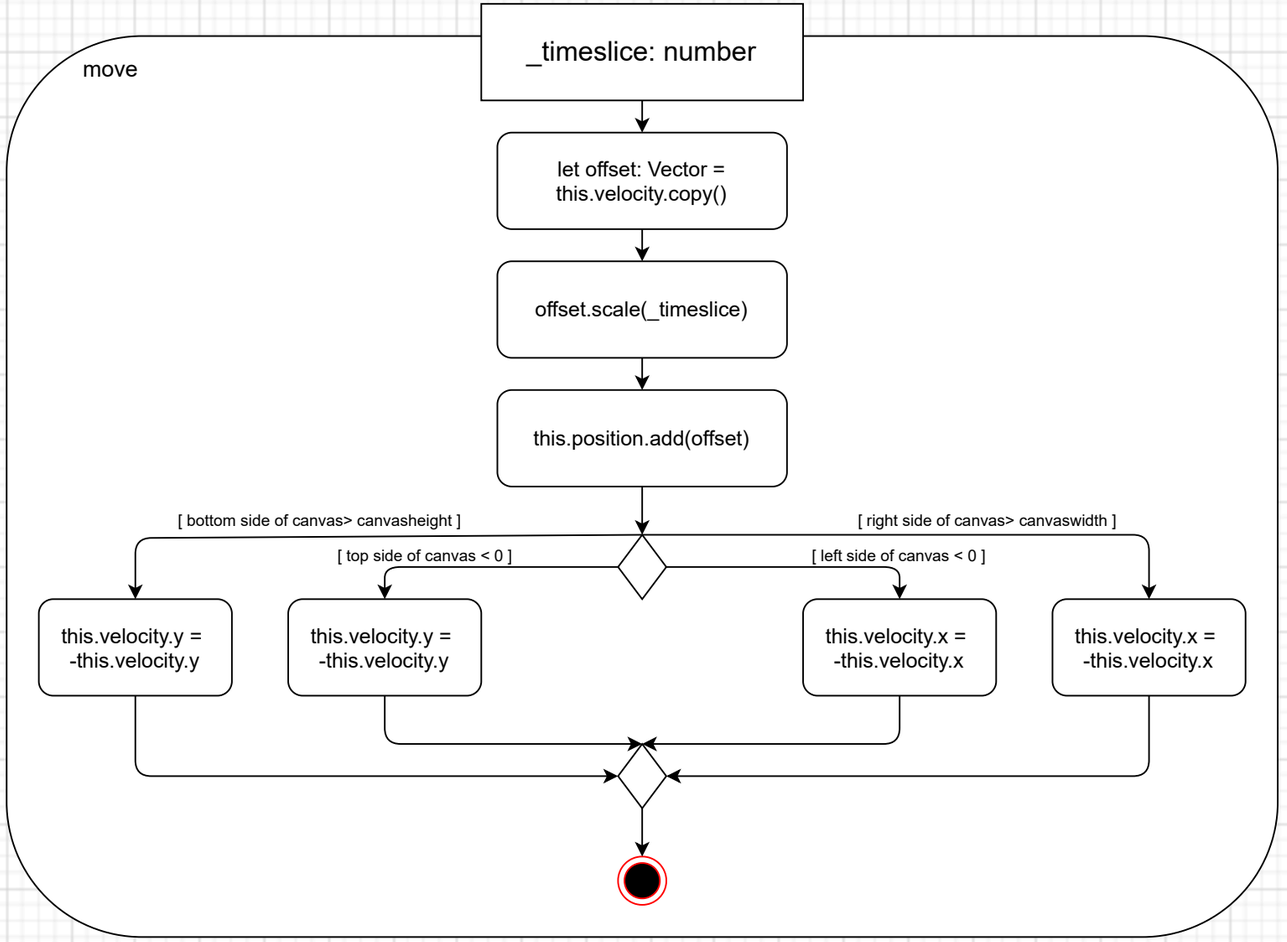
AD Heart

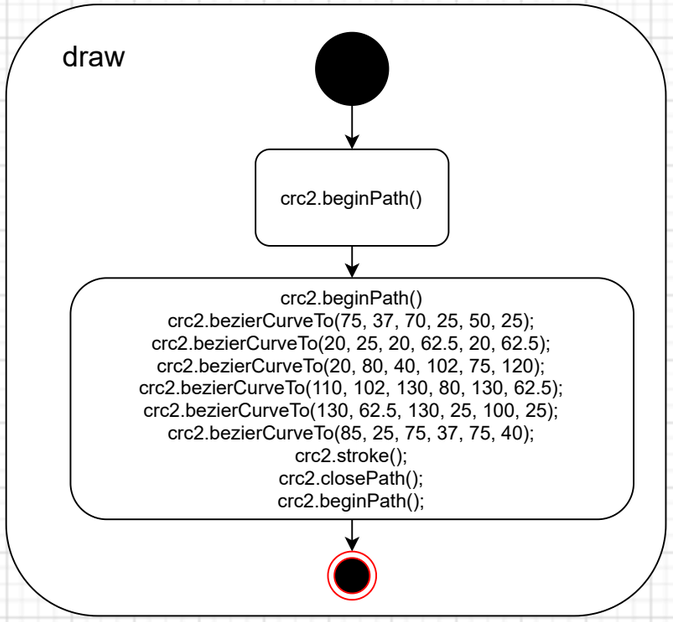
Heart

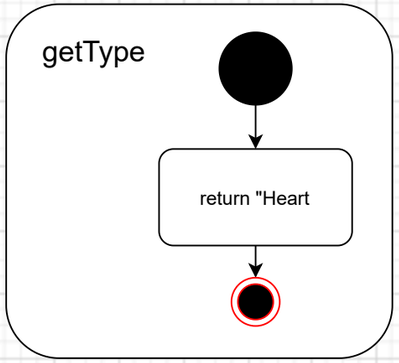
+ position: Vector
+ velocity: Vector

+ move ()
+ draw ()
+ getType()







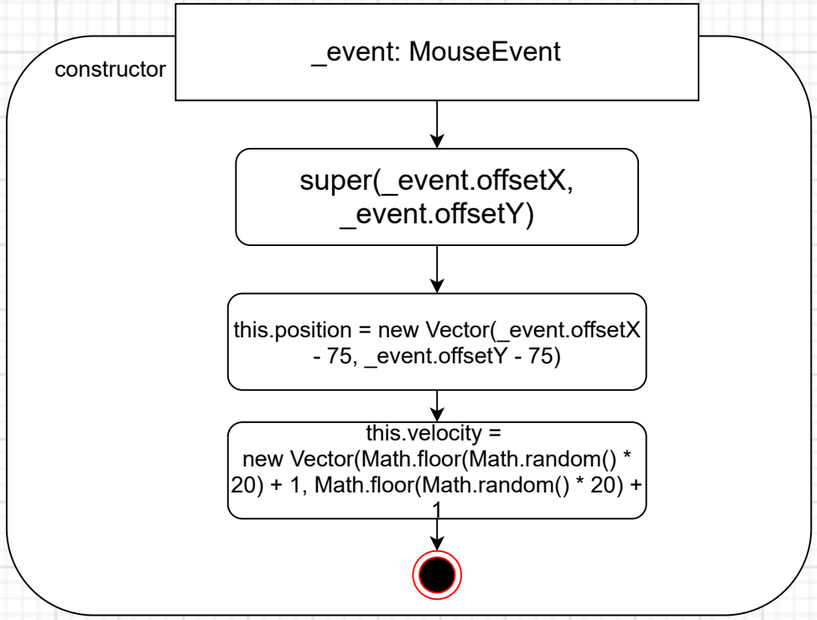


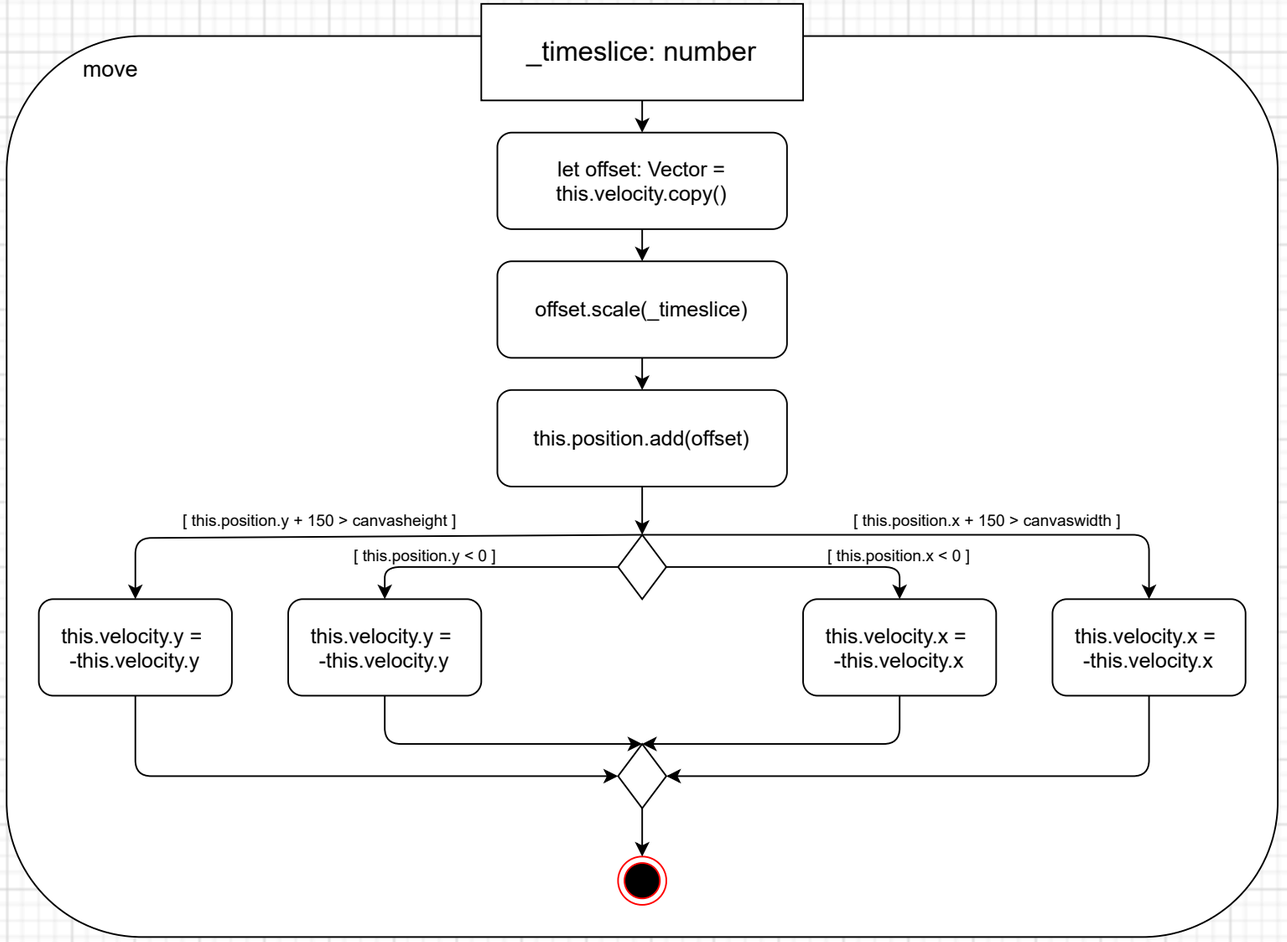
AD Rectangle

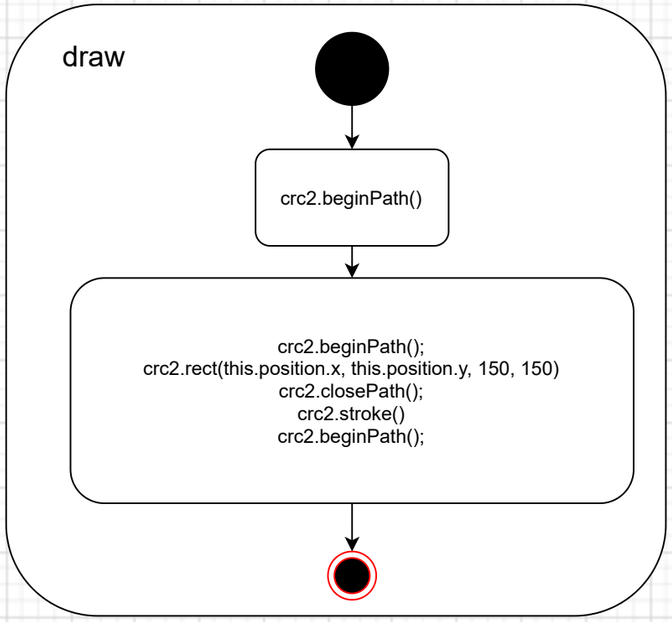
Rectangle

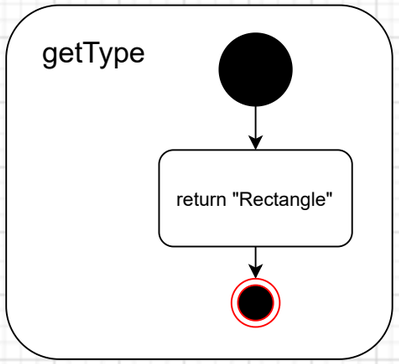
+ position: Vector
+ velocity: Vector

+ move ()
+ draw ()
+ getType()

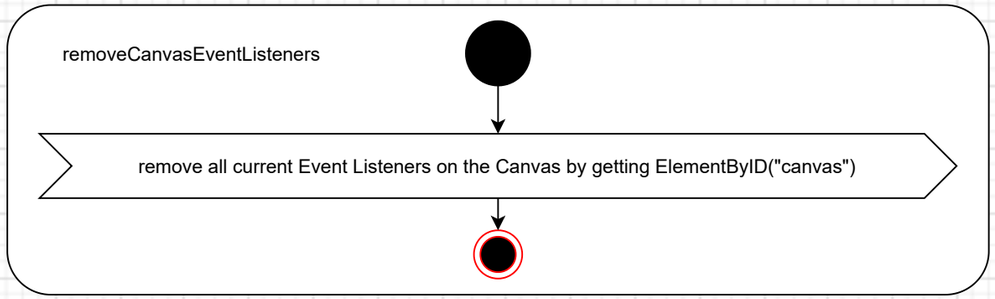


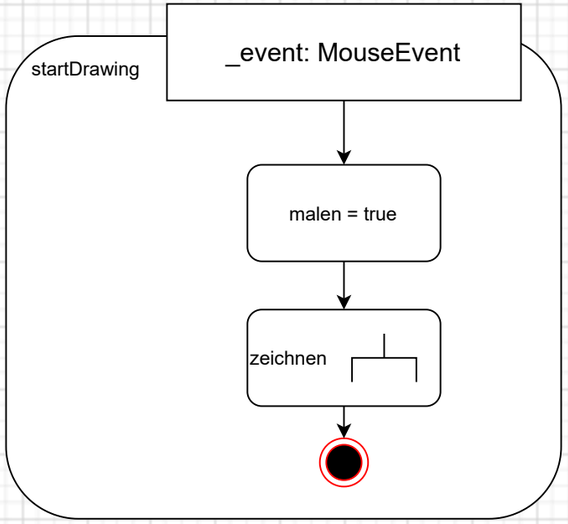


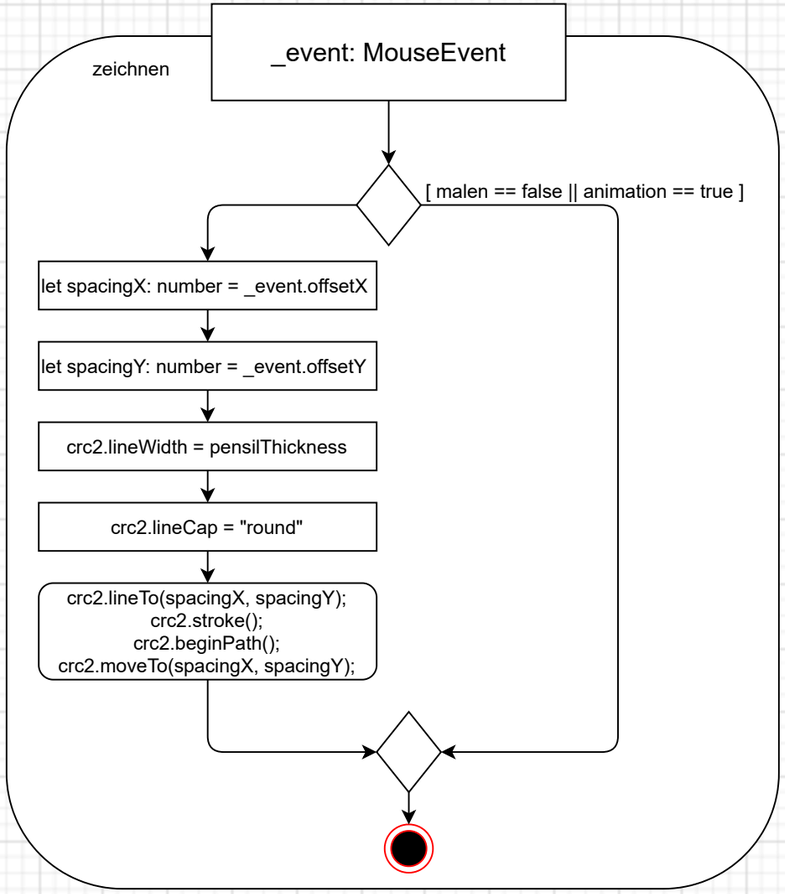


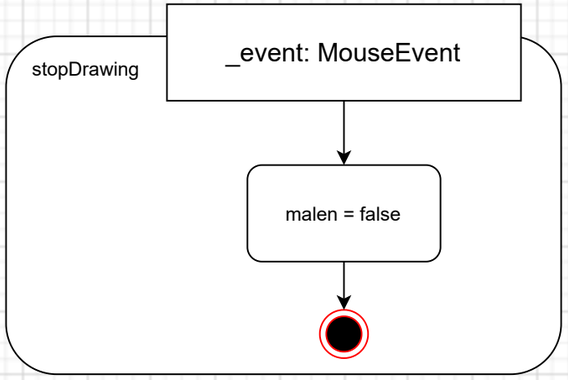


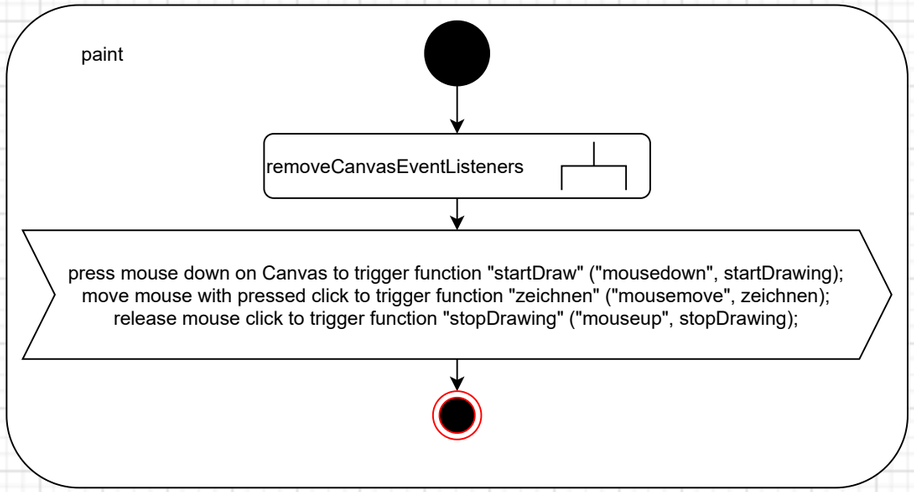
AD Tools

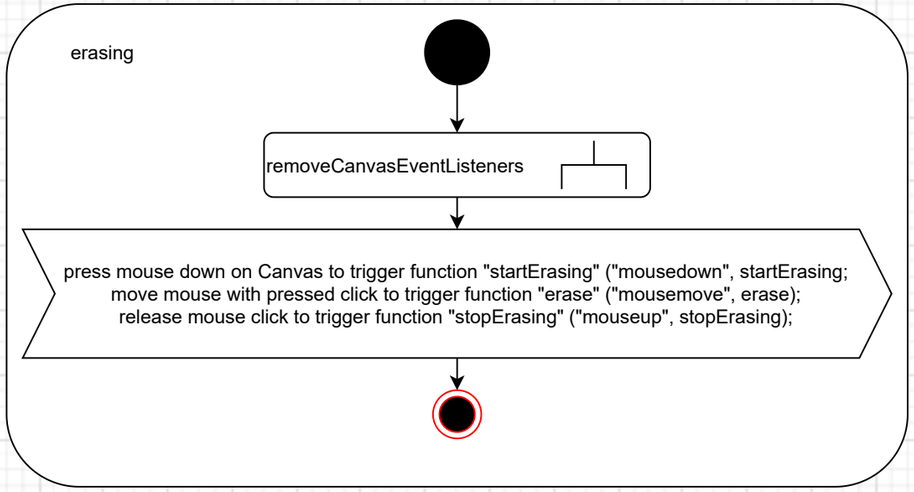

removeCanvasEventListeners
remove all current Event Listeners on the Canvas by getting ElementByID("canvas")


startDrawing
_event: MouseEvent
malen = true
zeichnen

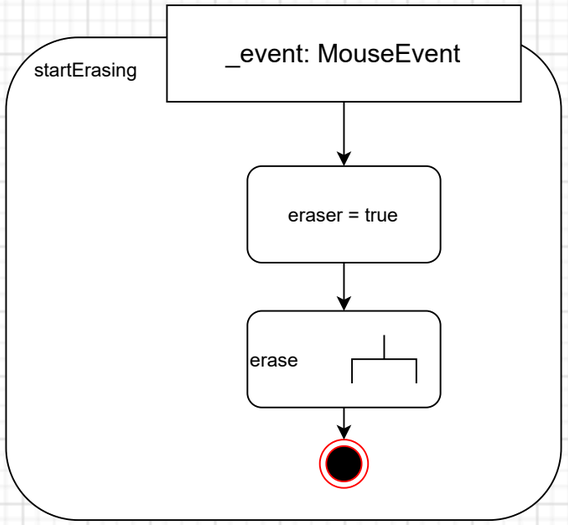

zeichnen
_event: MouseEvent
[malen == false || animation == true]
let spacingX: number = _event.offsetX
let spacingY: number = _event.offsetY
crc2.lineWidth = pensilThickness
crc2.lineCap = "round"
crc2.lineTo(spacingX, spacingY);
crc2.stroke();
crc2.beginPath();
crc2.moveTo(spacingX, spacingY);


stopDrawing
_event: MouseEvent
malen = false

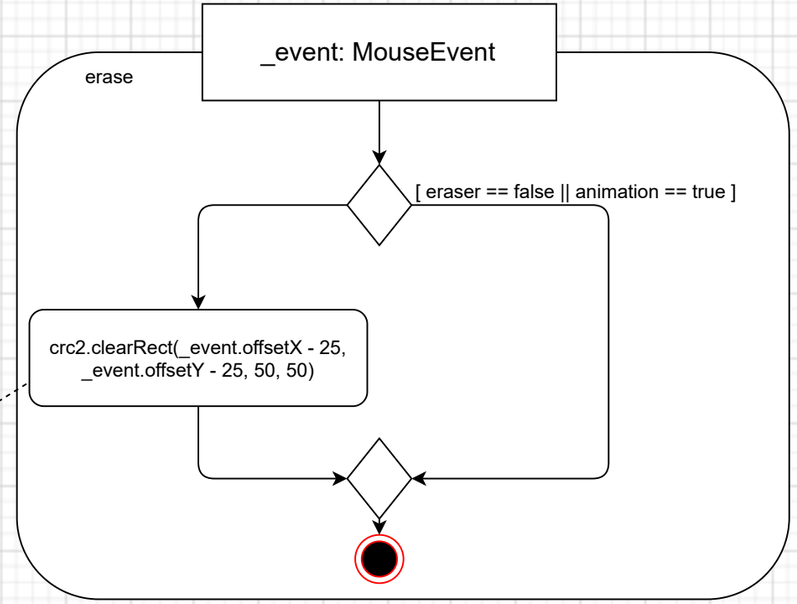

paint
removeCanvasEventListeners
press mouse down on Canvas to trigger function "startDraw" ("mousedown", startDrawing);
move mouse with pressed click to trigger function "zeichnen" ("mousemove", zeichnen);
release mouse click to trigger function "stopDrawing" ("mouseup", stopDrawing);

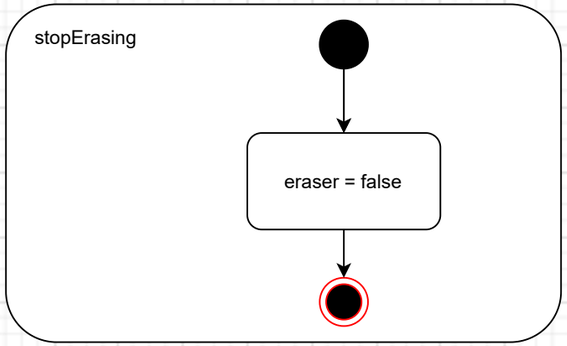

erasing
removeCanvasEventListeners
press mouse down on Canvas to trigger function "startErasing" ("mousedown", startErasing;
move mouse with pressed click to trigger function "erase" ("mousemove", erase);
release mouse click to trigger function "stopErasing" ("mouseup", stopErasing);

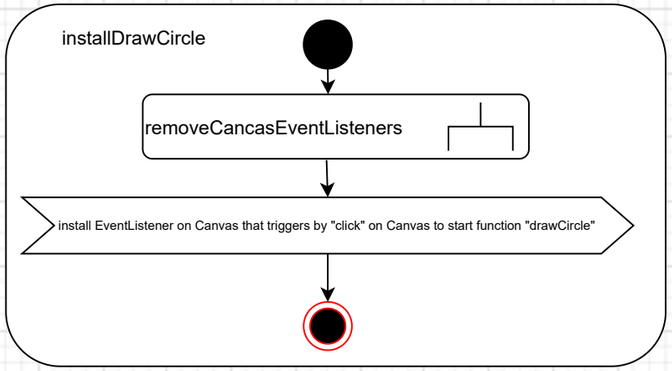
AD Tools

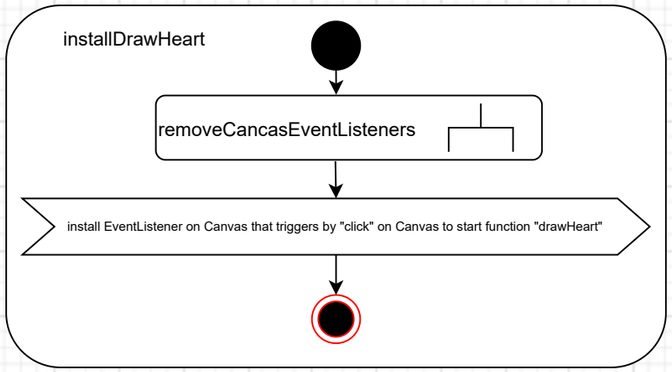


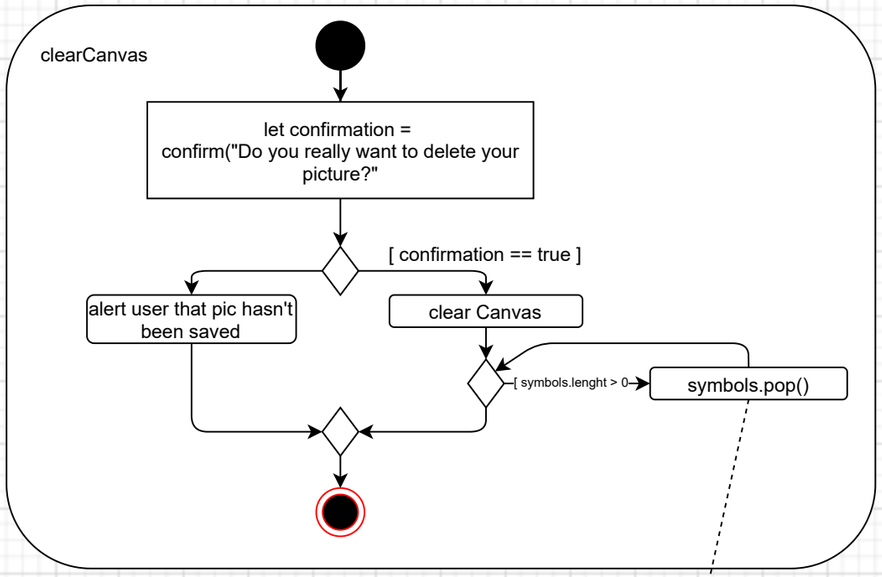
every time you move mouse with left-click active, you clear rect at your mouse position



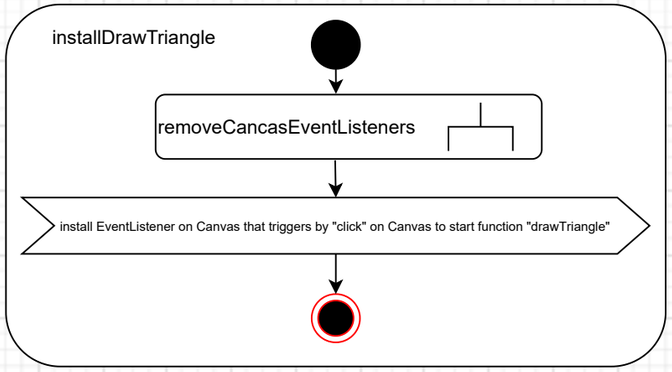


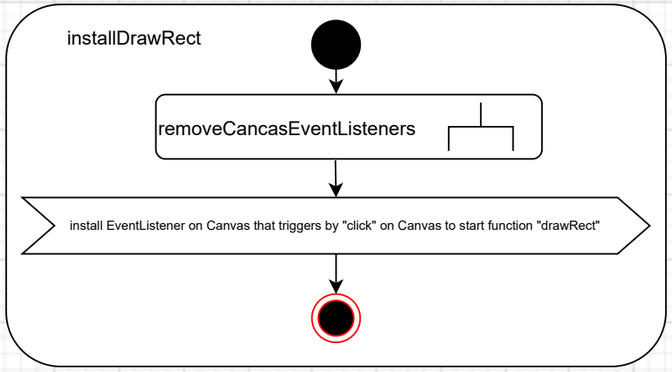




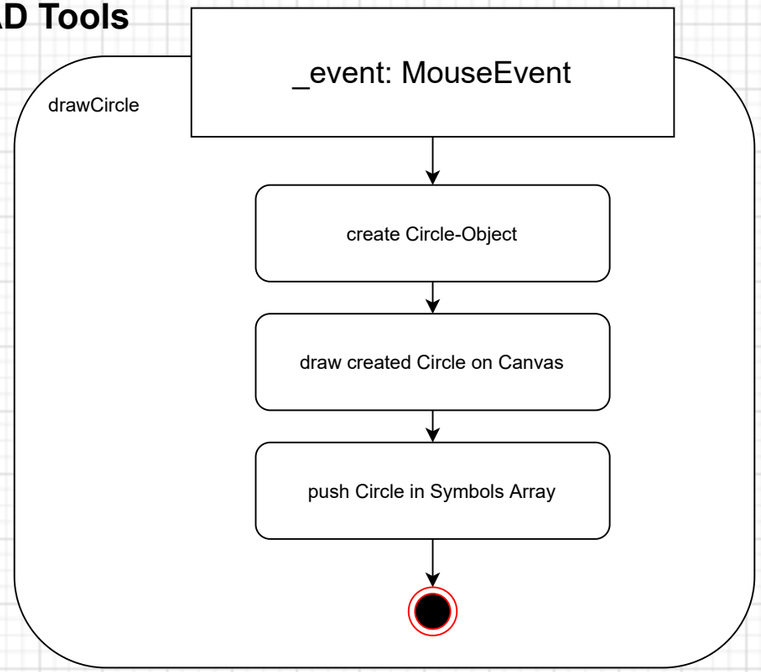


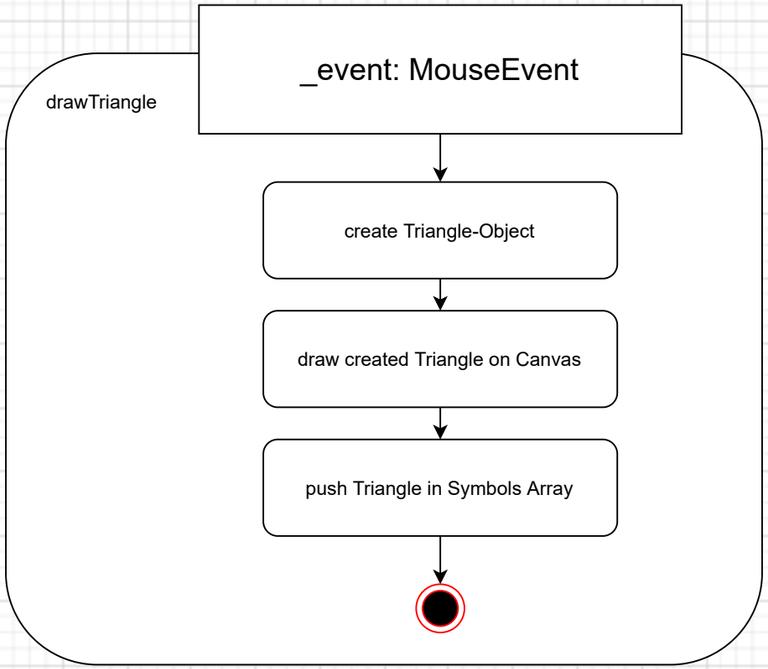
clear every Object on canvas by popping the entire array

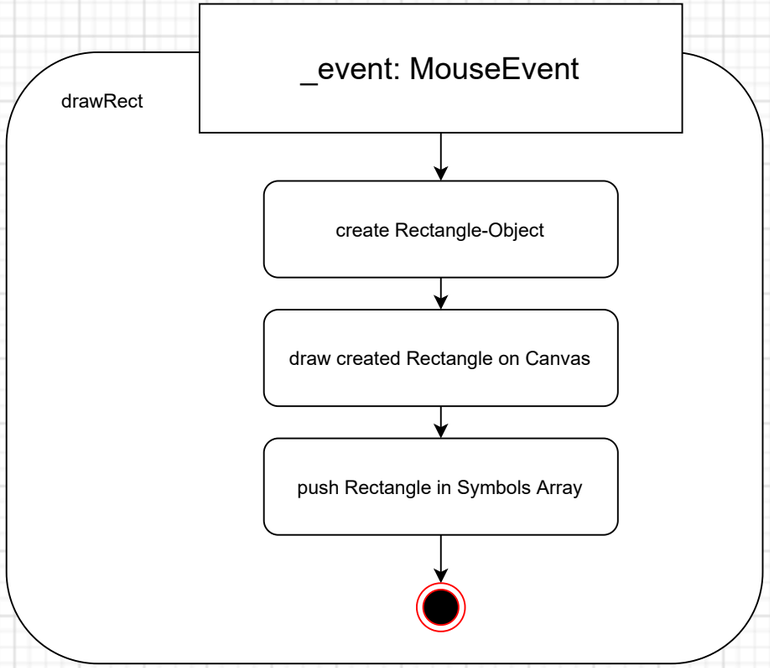


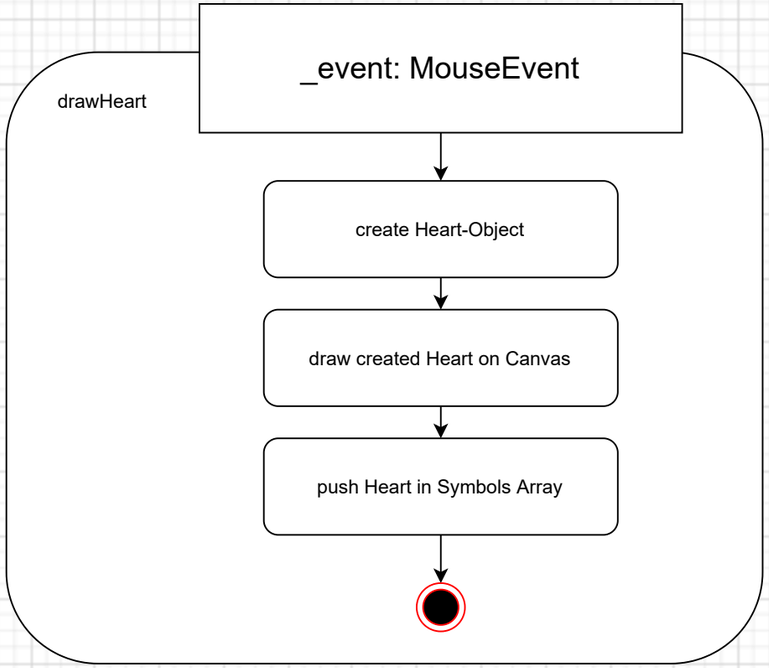


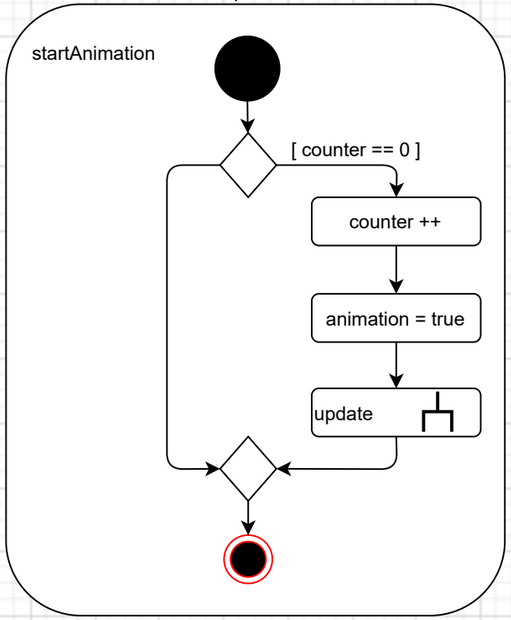
AD Tools

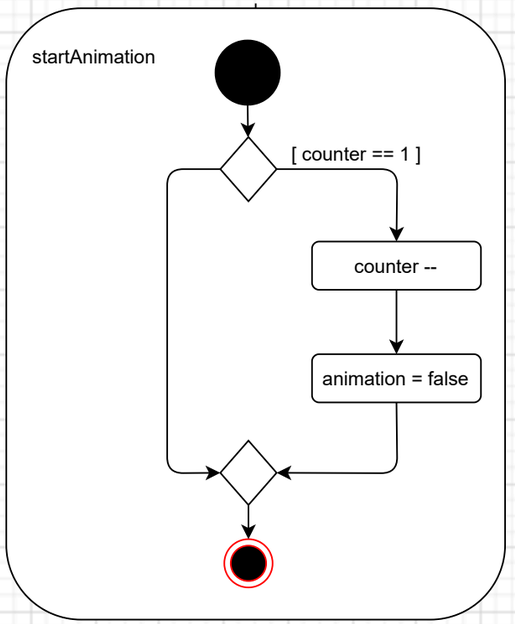






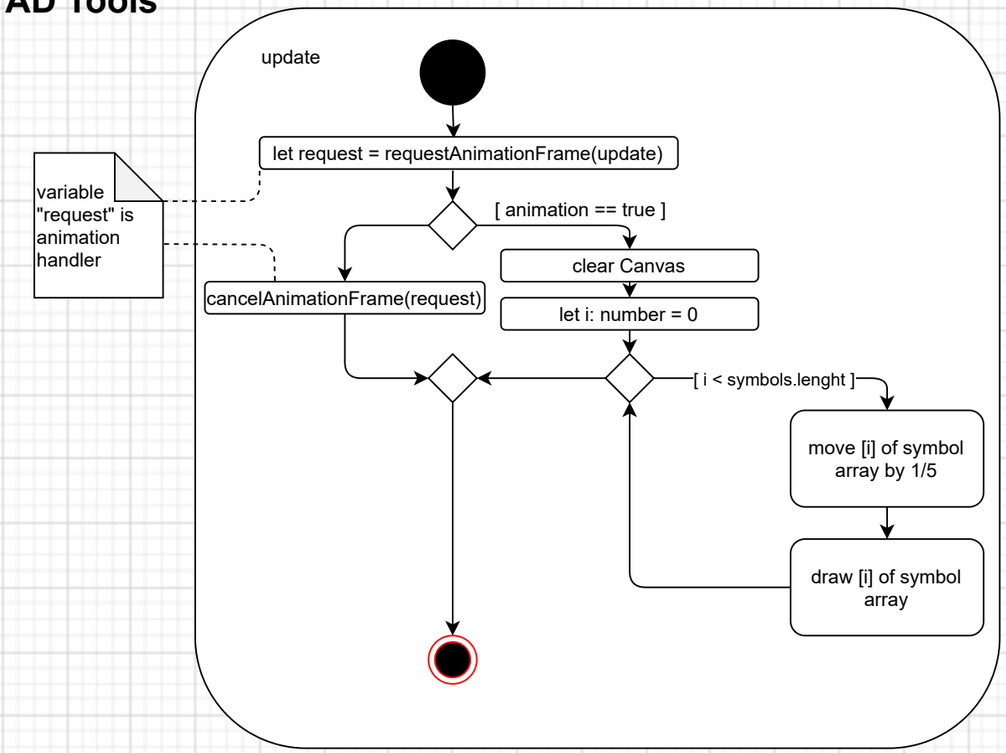


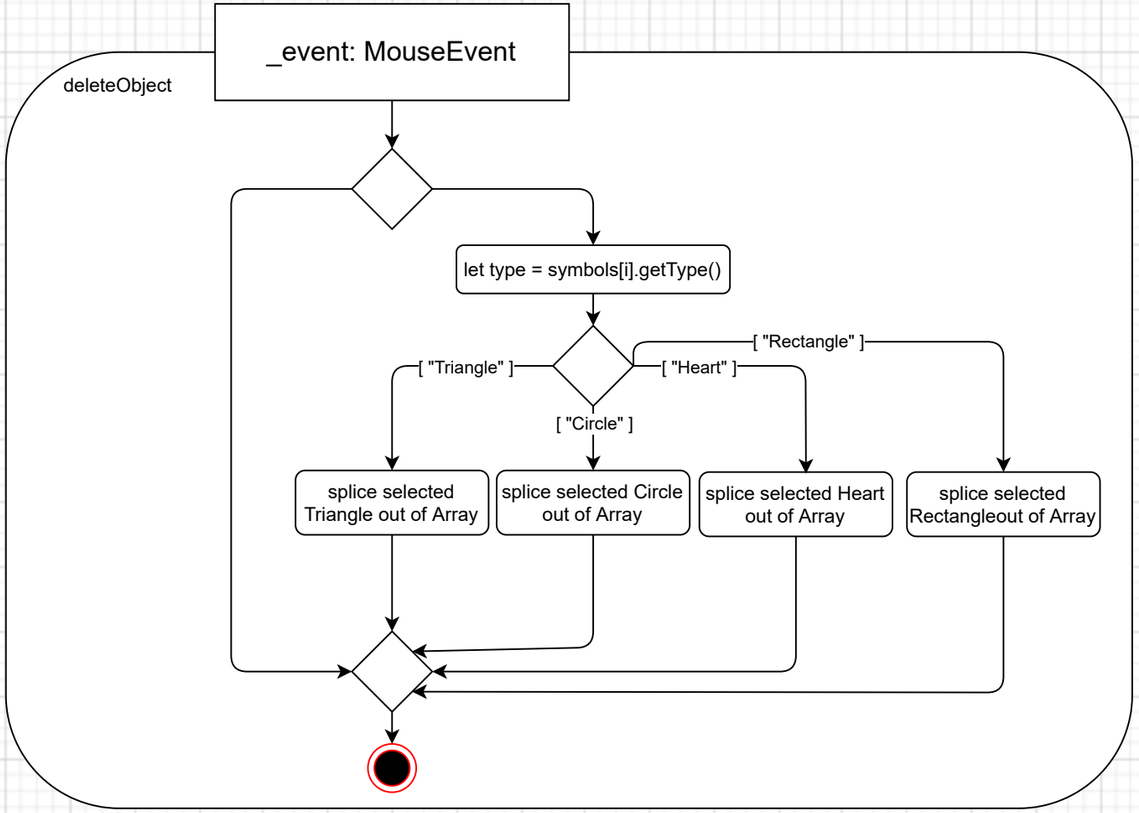


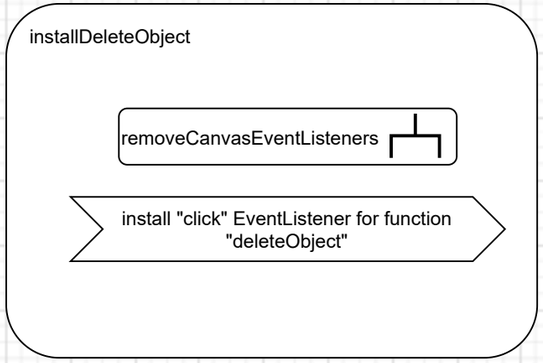


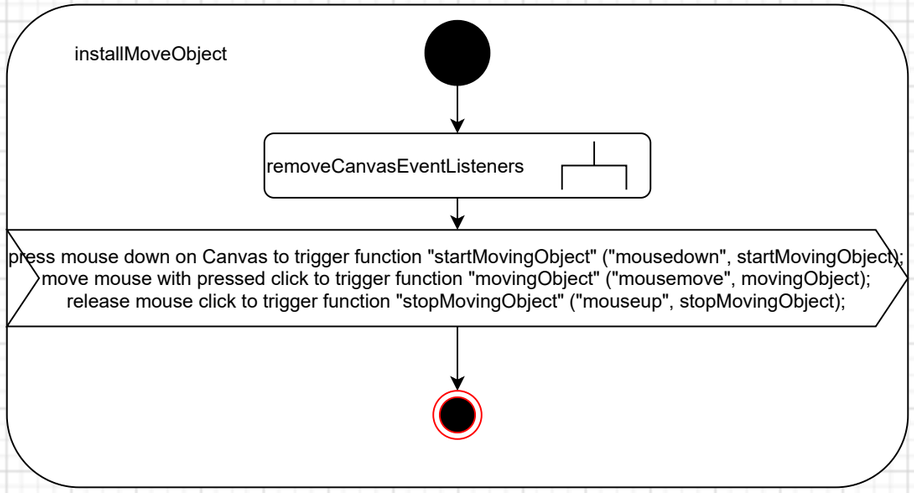
verhindert das mehrmalige Aufrufen von RAF

AD Tools

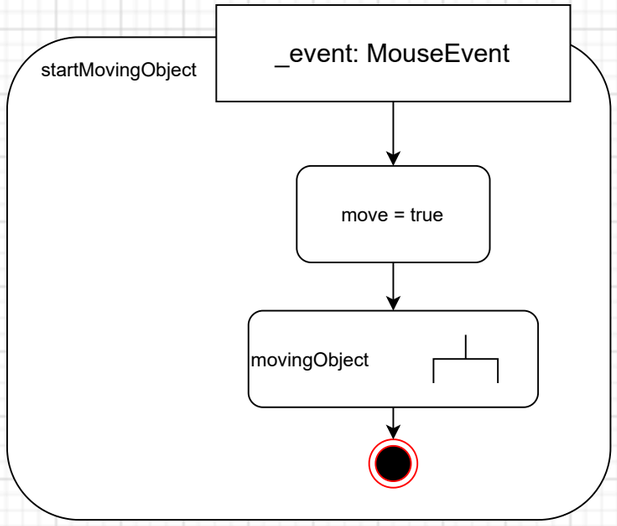

update
let request = requestAnimationFrame(update)
variable "request" is animation handler
[animation == true]
clear Canvas
cancelAnimationFrame(request)
let i: number = 0
[i < symbols.lenght]
move [i] of symbol array by 1/5
draw [i] of symbol array

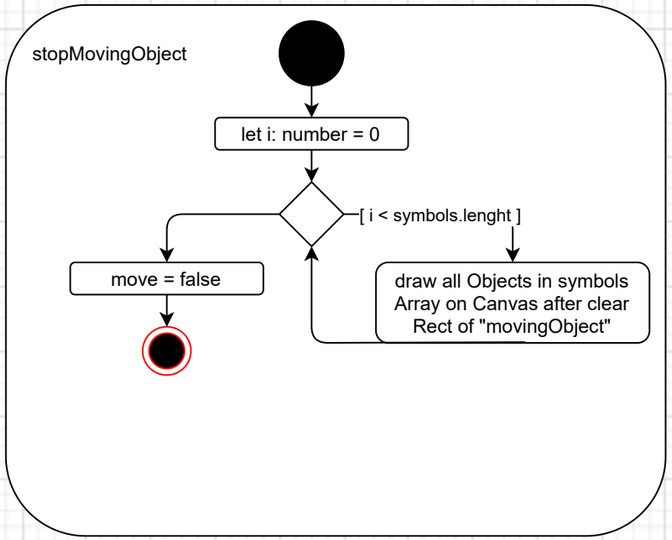

_event: MouseEvent
deleteObject
let type = symbols[i].getType()
["Triangle"]
["Heart"]
["Rectangle"]
["Circle"]
splice selected Triangle out of Array
splice selected Circle out of Array
splice selected Heart out of Array
splice selected Rectangleout of Array

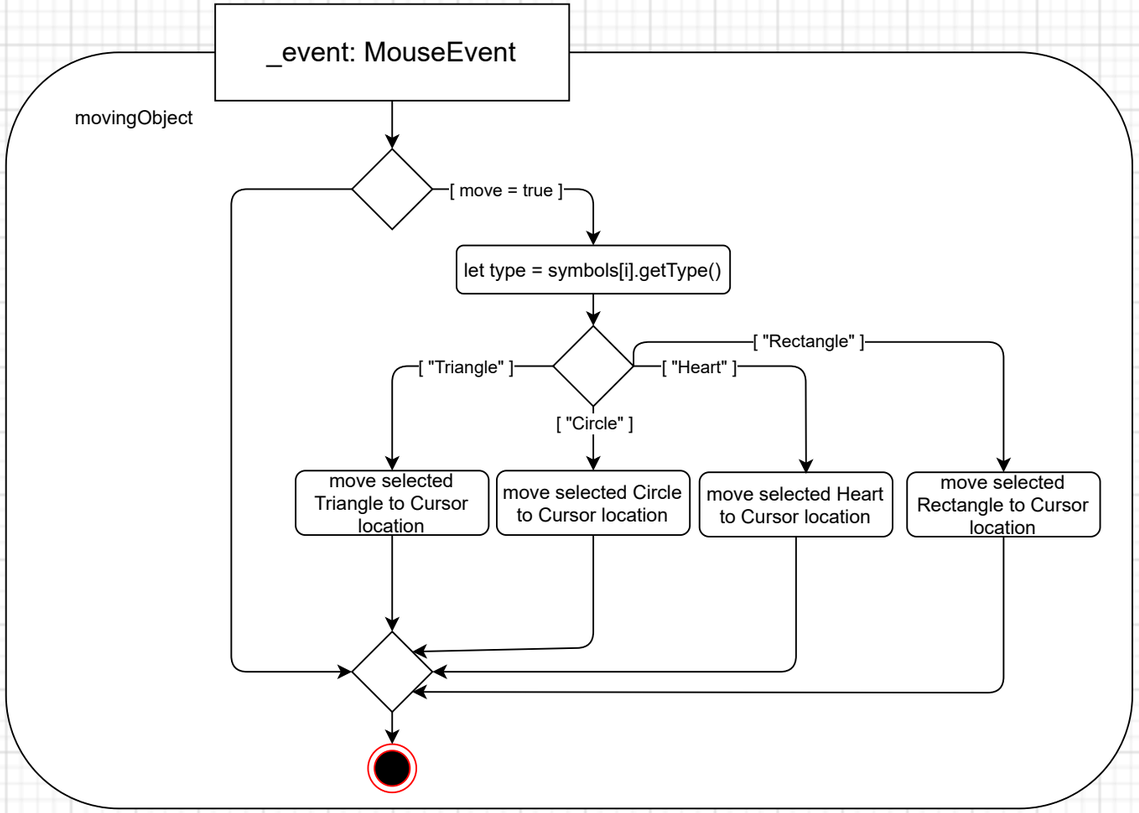

installDeleteObject
removeCanvasEventListeners
install "click" EventListener for function "deleteObject"


installMoveObject
removeCanvasEventListeners
press mouse down on Canvas to trigger function "startMovingObject" ("mousedown", startMovingObject);
move mouse with pressed click to trigger function "movingObject" ("mousemove", movingObject);
release mouse click to trigger function "stopMovingObject" ("mouseup", stopMovingObject);

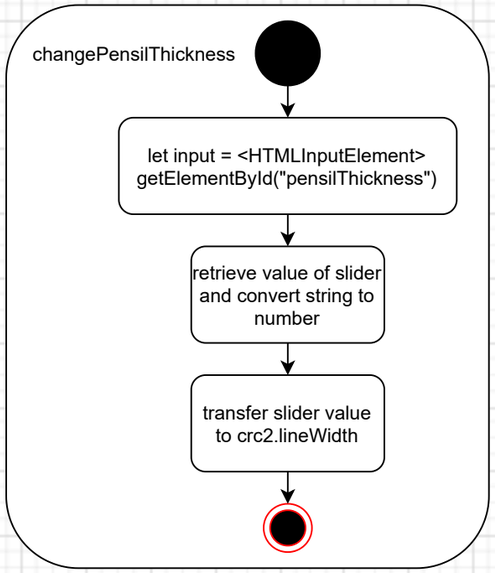
AD Tools

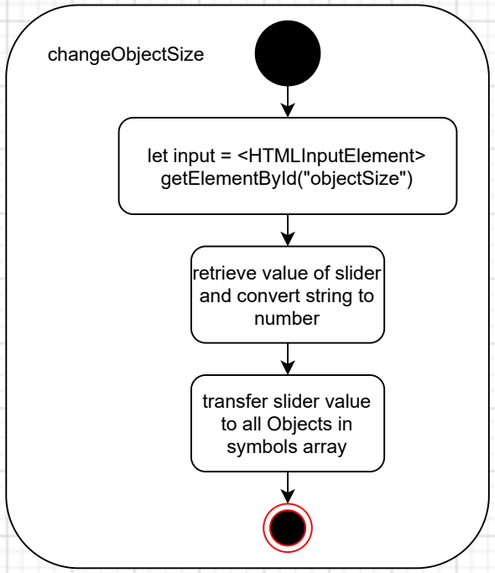

startMovingObject
_event: MouseEvent
move = true
movingObject

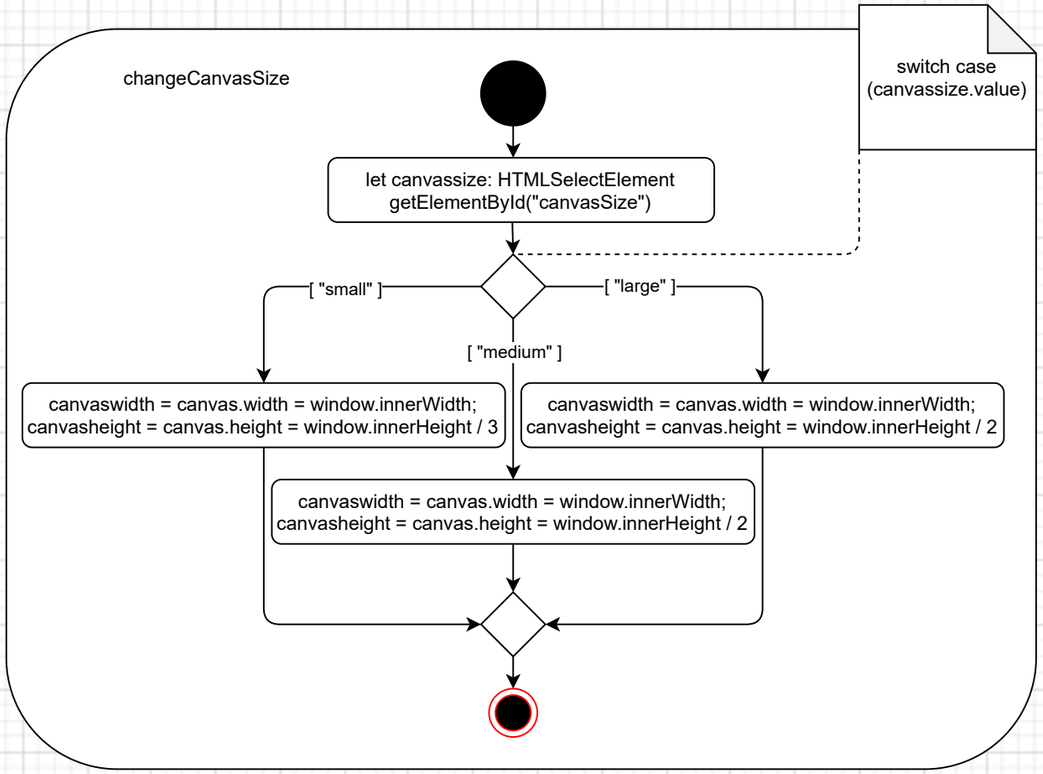

stopMovingObject
let i: number = 0
[i < symbols.lenght]
move = false
draw all Objects in symbols Array on Canvas after clear Rect of "movingObject"

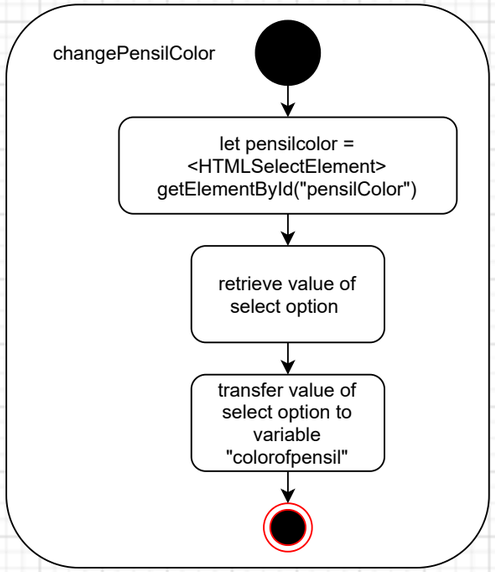

_event: MouseEvent
movingObject
[move = true]
let type = symbols[i].getType()
["Rectangle"]
["Triangle"]
["Heart"]
["Circle"]
move selected Triangle to Cursor location
move selected Circle to Cursor location
move selected Heart to Cursor location
move selected Rectangle to Cursor location

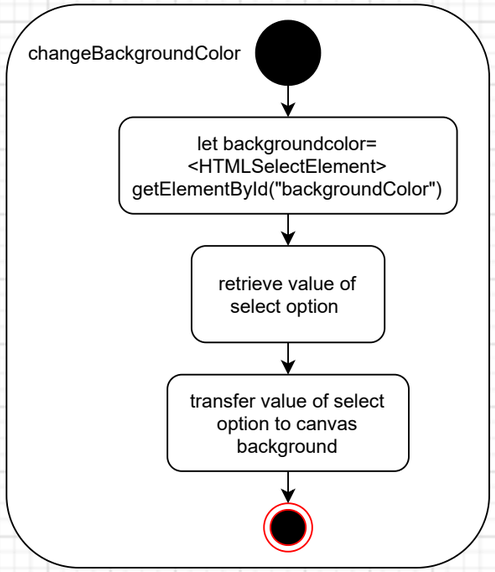
AD Customisations







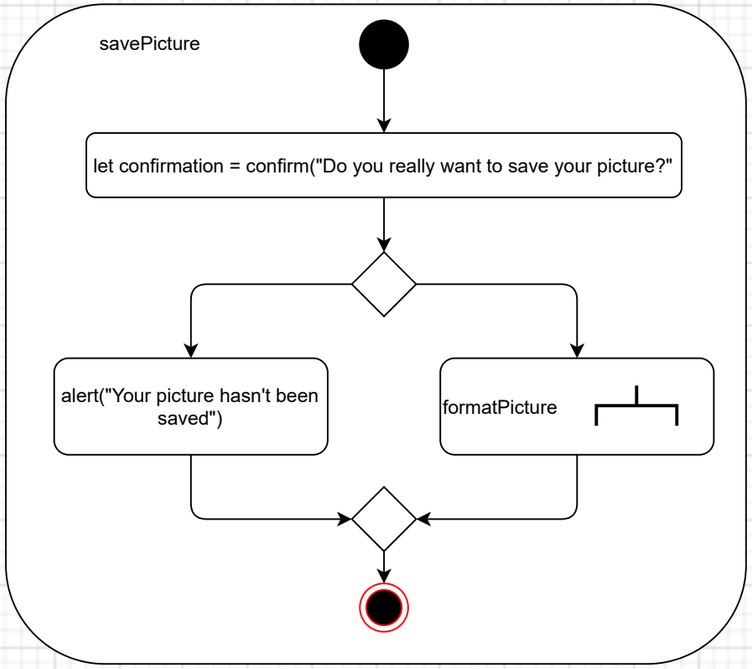


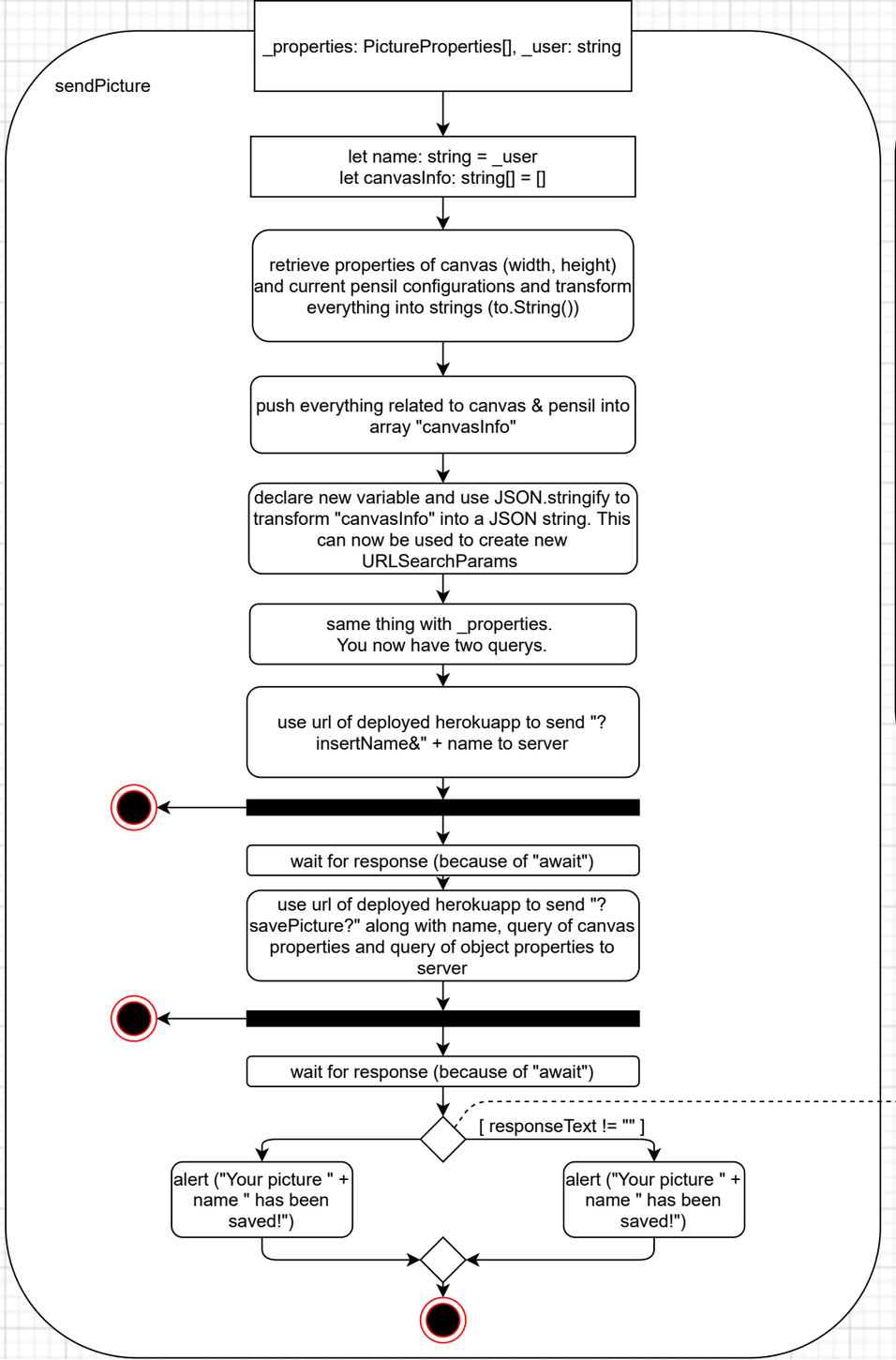


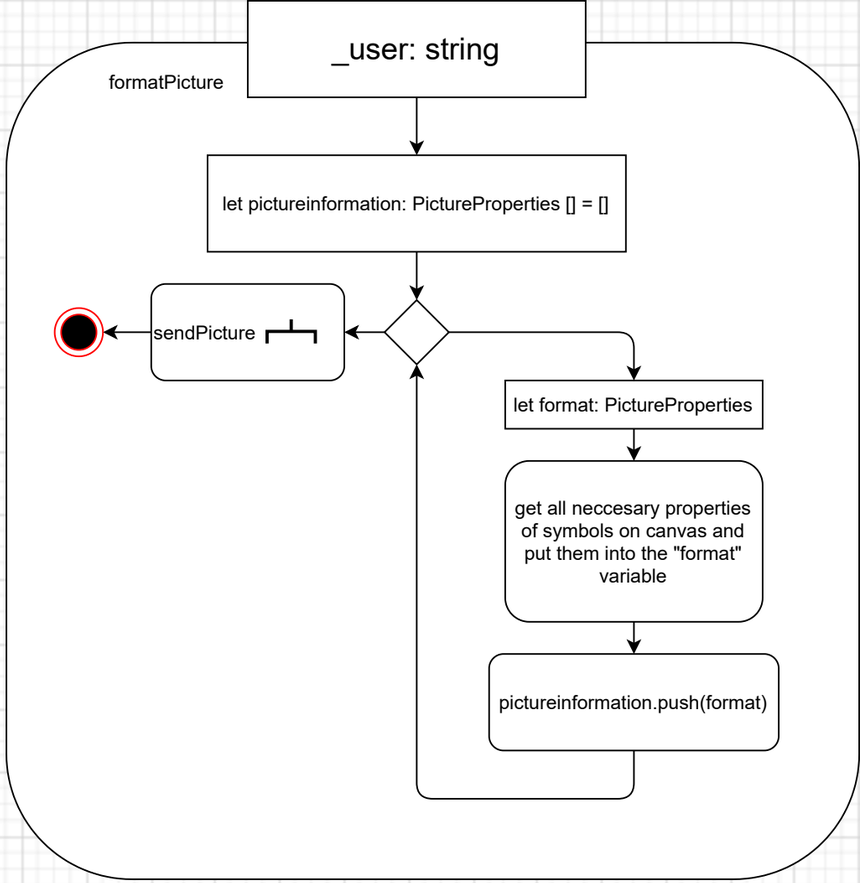
AD savePicture

«interface»
PictureProperties

type: string
positionX: number
positionY: number
velocityX: number
velocityY: number
size: number





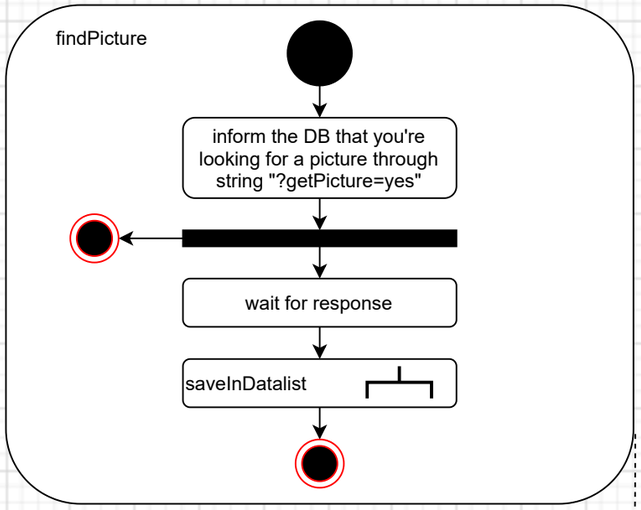


to make sure the user gets the information that the pic has been saved properly

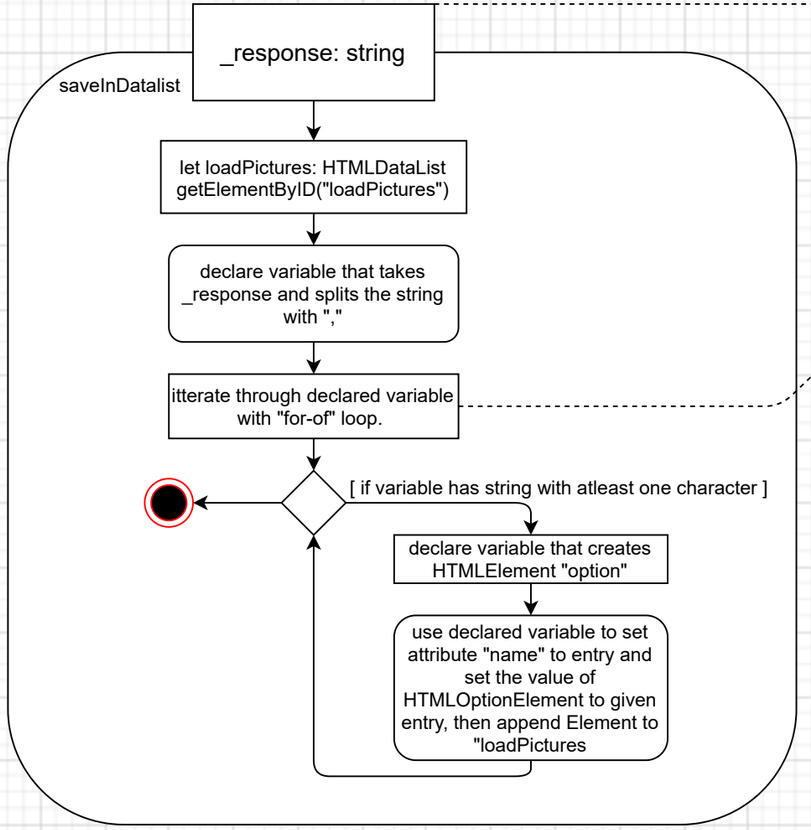
AD savePicture

«interface»
PictureProperties

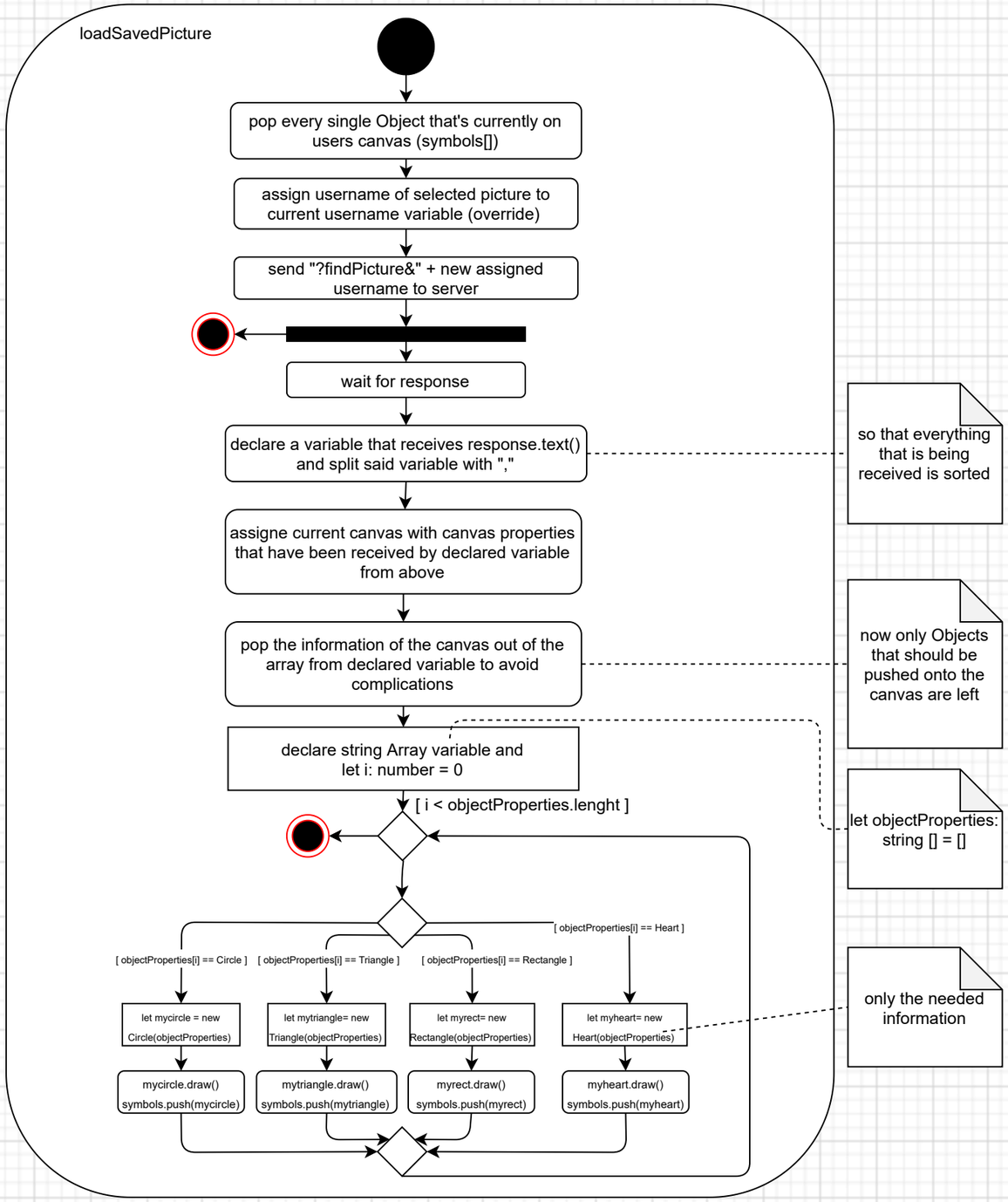
type: string
positionX: number
positionY: number
velocityX: number
velocityY: number
size: number



look for HTML Element "Datalist" and save found pictures names in seperate function through the response.text()



let entry of response



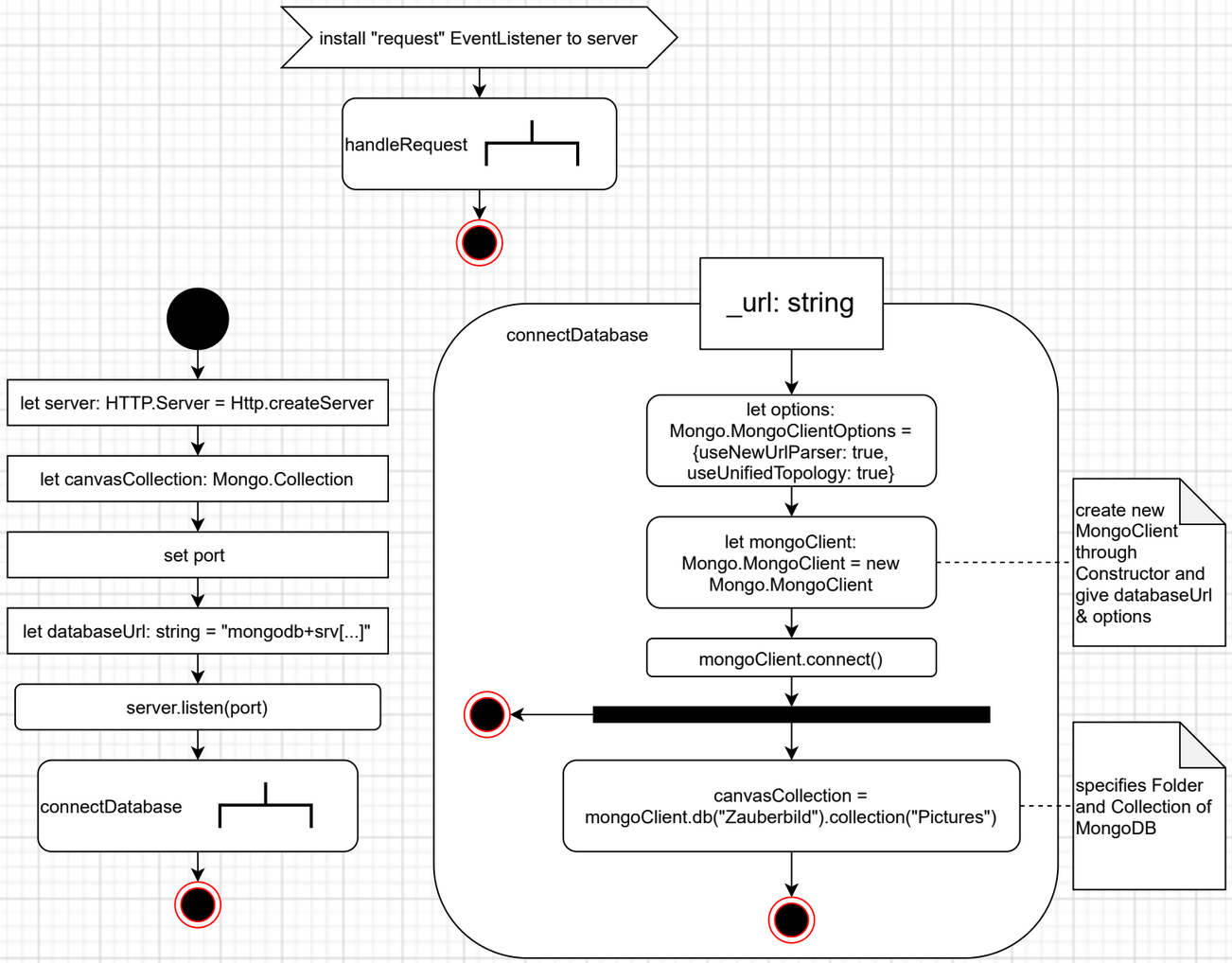
so that everything that is being received is sorted

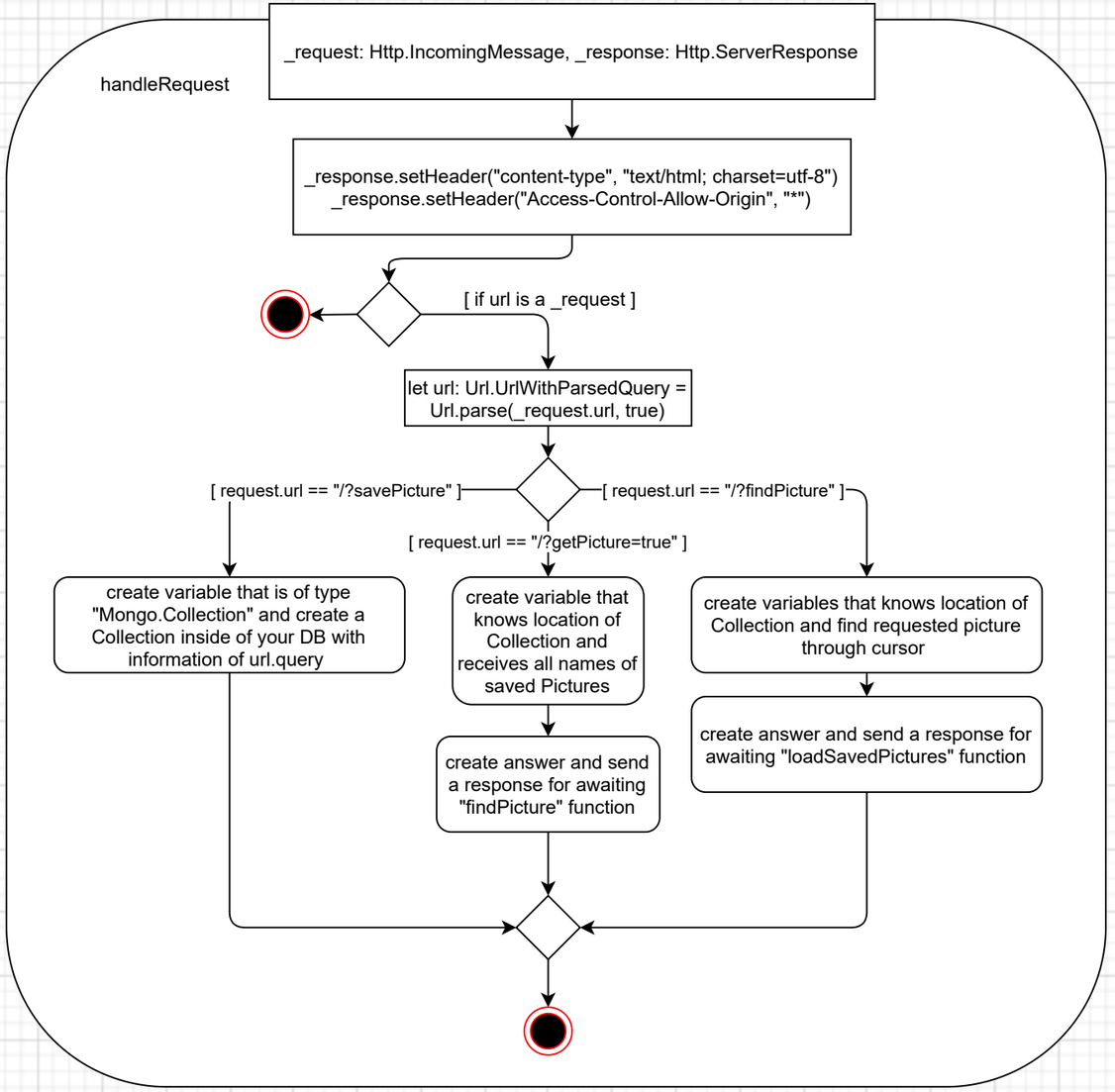
now only Objects that should be pushed onto the canvas are left

let objectProperties: string [] = []

only the needed information

AD Server


install "request" EventListener to server
handleRequest
let server: HTTP.Server = Http.createServer
let canvasCollection: Mongo.Collection
set port
let databaseUrl: string = "mongodb+srv[...]"
server.listen(port)
connectDatabase
_url: string
connectDatabase
let options: Mongo.MongoClientOptions = {useNewUrlParser: true, useUnifiedTopology: true}
let mongoClient: Mongo.MongoClient = new Mongo.MongoClient
create new MongoClient through Constructor and give databaseUrl & options
mongoClient.connect()
canvasCollection = mongoClient.db("Zauberbild").collection("Pictures")
specifies Folder and Collection of MongoDB


handleRequest
_request: Http.IncomingMessage, _response: Http.ServerResponse
_response.setHeader("content-type", "text/html; charset=utf-8")
_response.setHeader("Access-Control-Allow-Origin", "*")
[if url is a _request]
let url: Url.UrlWithParsedQuery = Url.parse(_request.url, true)
[request.url == "/?savePicture"]
[request.url == "/?findPicture"]
[request.url == "/?getPicture=true"]
create variable that is of type "Mongo.Collection" and create a Collection inside of your DB with information of url.query
create variable that knows location of Collection and receives all names of saved Pictures
create variables that knows location of Collection and find requested picture through cursor
create answer and send a response for awaiting "findPicture" function
create answer and send a response for awaiting "loadSavedPictures" function