

Plattform ist der PC, da man somit eine Steuerung des Spielfisches durch wasd ermöglicht.

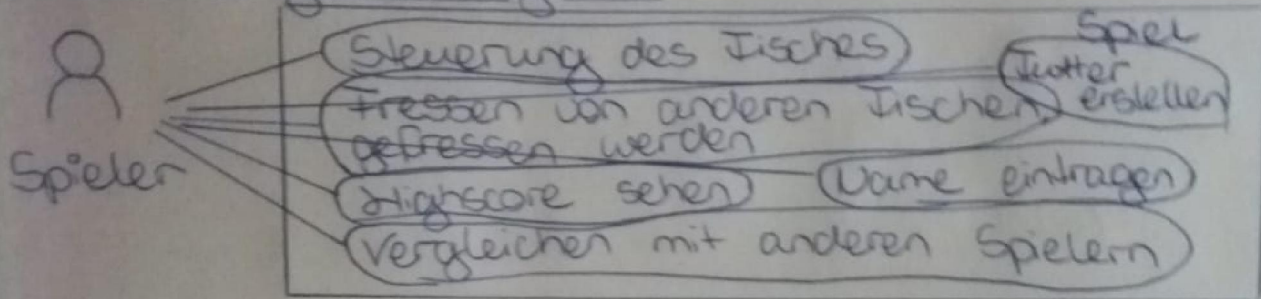
HTML Skizze

canvas
1500px • 850px
statisch

highscore
<div> mit id

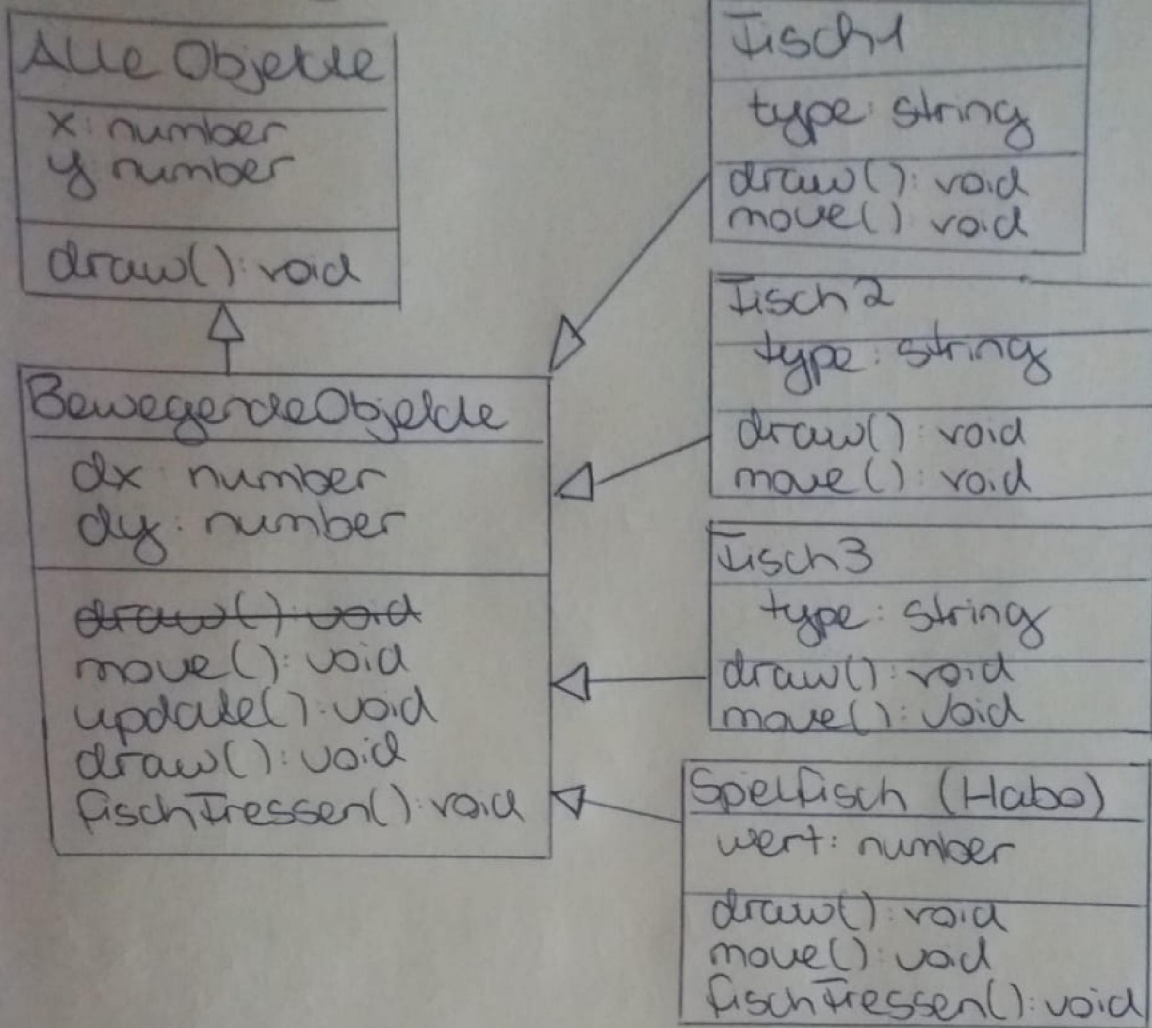
Andere Spieler
<div> mit id

Anwendungsfalldiagramm



- * Bewegung erfolgt mittels wasd
- * Hauptfisch/Spielfisch kann kleinere Tische fressen
- * Bei Spielende kann der Spieler seinen Namen eintragen
- * Spieler wird kleiner bei bestimmten Tischen (Kugelfisch) und verliert Punkte
- * Spieler stirbt bei bestimmten Tischberührungen mit Qualle
- * Tutter durch klicken erstellen

Klassendiagramm

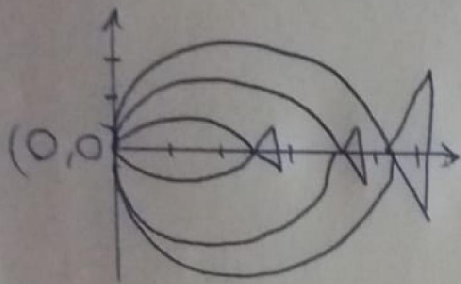
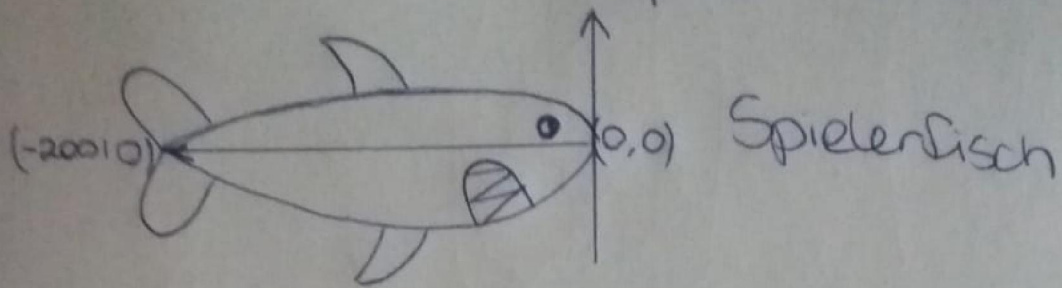


Interaktionsmöglichkeiten

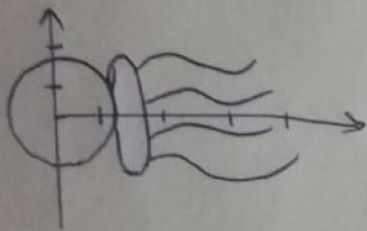
- X
 - * Spielfisch frisst kleinen Tisch
 - * Punktzahl erhöht sich
 - * kleinen Tisch wird entfernt und neuer hinzugefügt
 - * Spielfisch wächst
- X
 - * Spiel beendet sobald ich versuche einen großen Tisch zu fressen
 - * Spiel hält an, ich gebe meinen Namen ein und sehe meinen Highscore
- X
 - * Spiel endet bei Berührung mit Qualle
 - * beenden

Zeichnungen

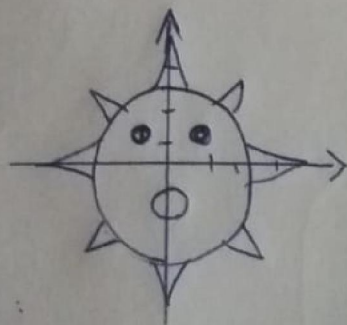
Genaue Maße werden
später entschieden



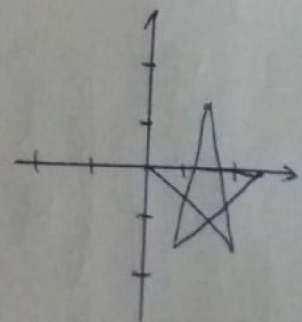
3 zu fressende Fische



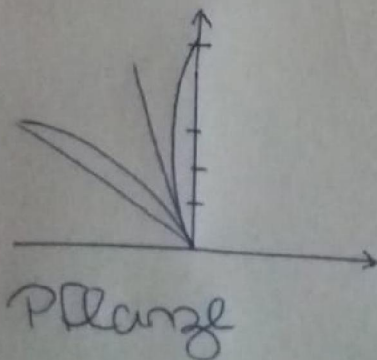
Qualle



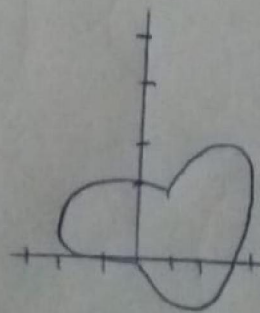
Kugelfisch



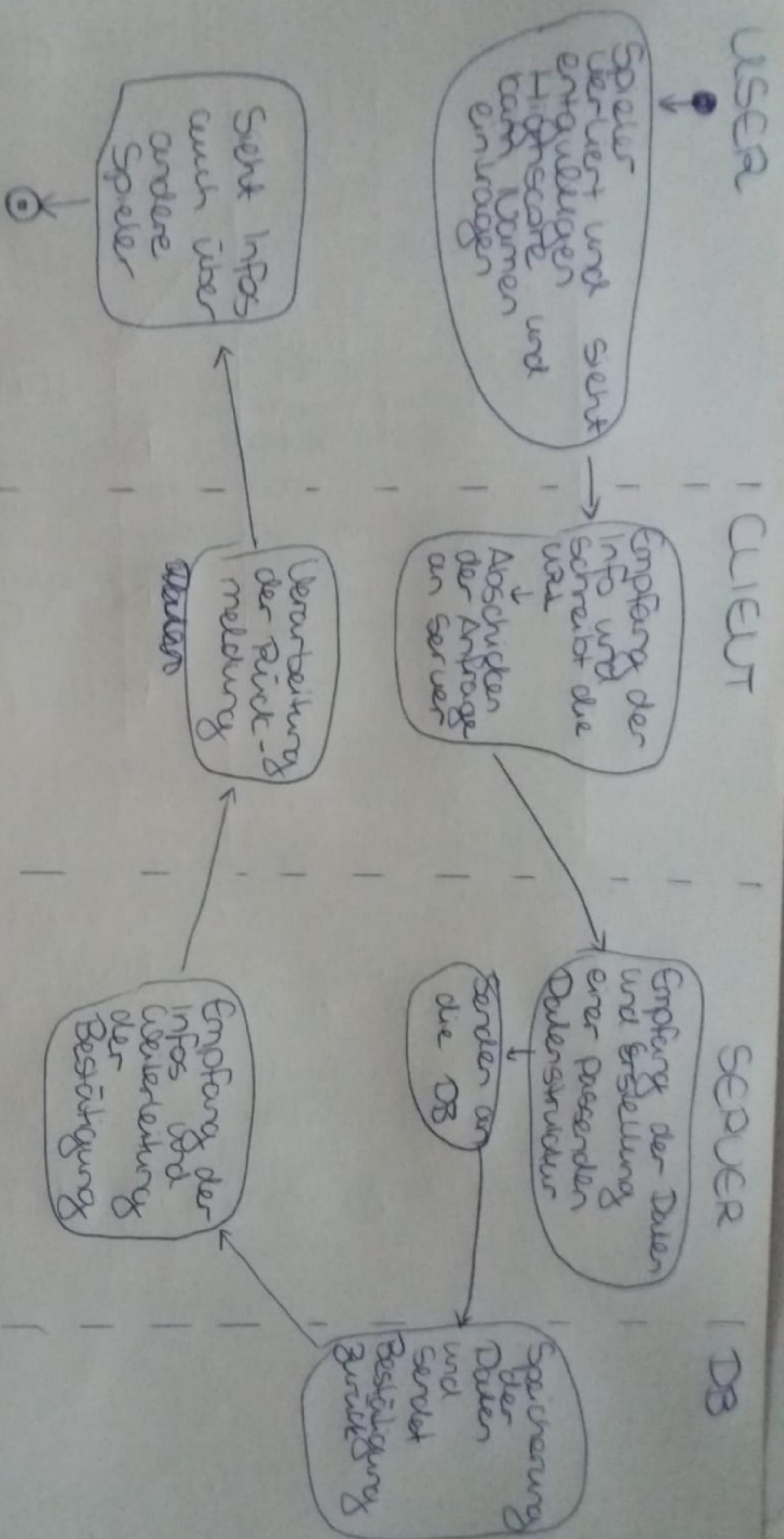
Stern



Pflanze



Stein

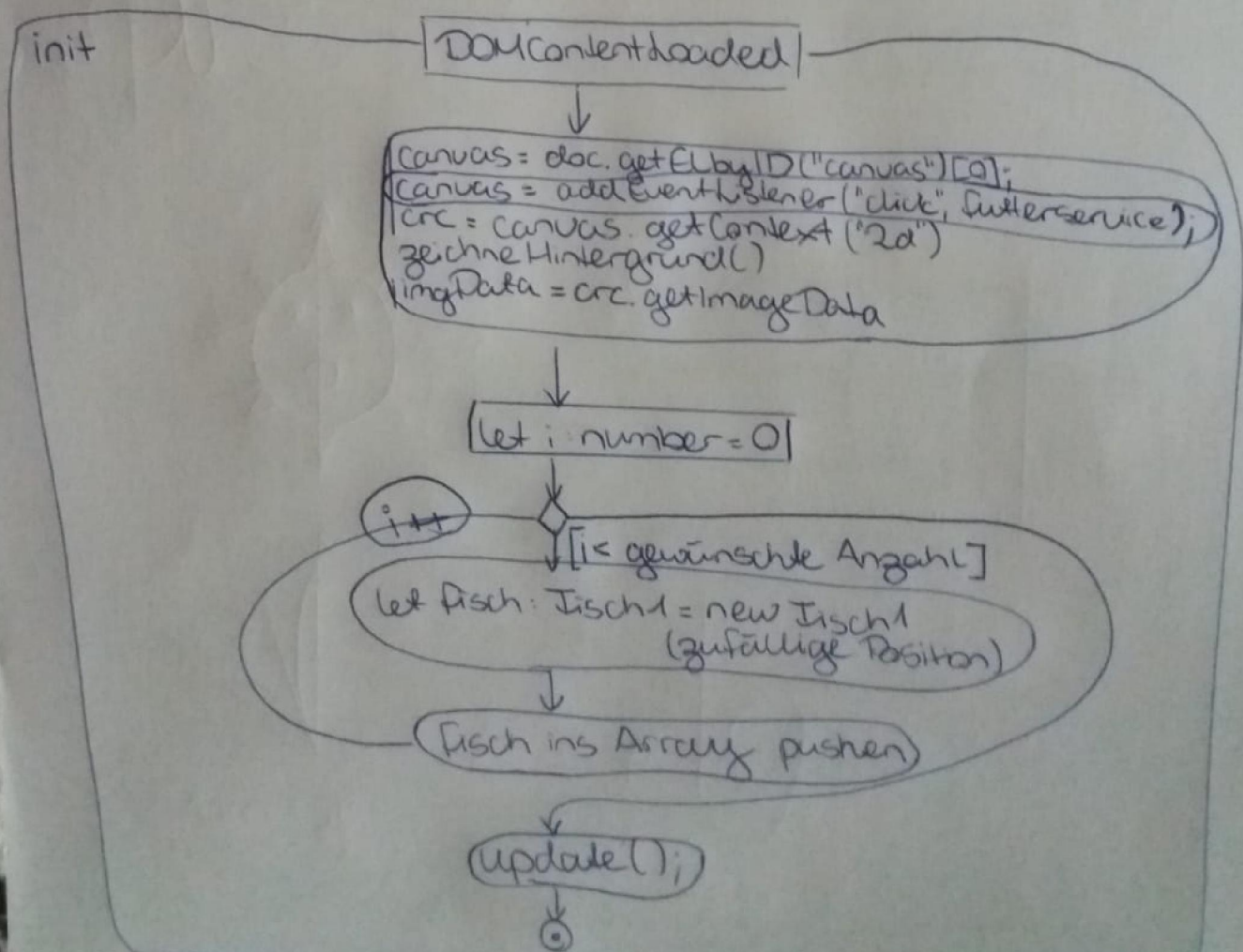


Aktivitätsdiagramm (1)

Canvas

EventListener für DOMContentLoaded
und für Keydown erstellen init

```
export let crc  
export let canvas: HTMLCanvasElem  
export let bewObjArray: BewegendeObjekte[] = []  
export let spielerName: string  
export let habo: Habo  
export let highscore: number = 0  
let fps: number = 40  
let imgData: ImageData
```



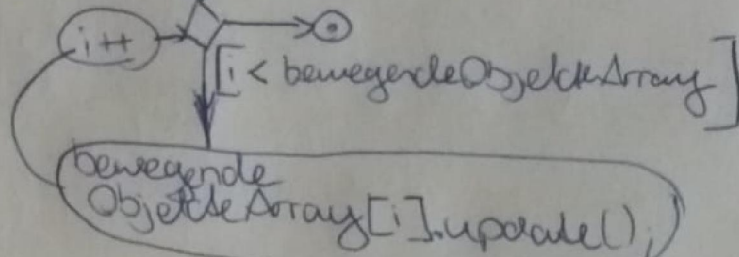
Aktivitätsdiagramm (2)

canvas

update

```
window.setTimeout(update, 1000/fps);  
ctx.clearRect(0,0,canvas.width,c.height);  
ctx.putImageData(imgData,0,0);
```

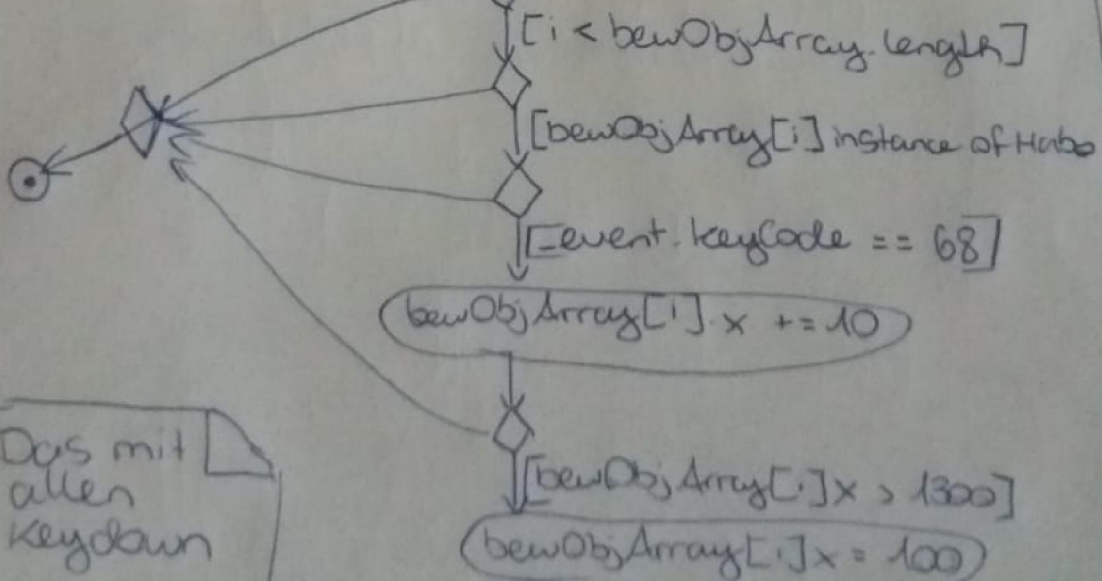
i: number



bewegungsfisches

- event: Keyboard Event

i = number = 0



Das mit
allen
keydown

Aktivitätsdiagramm (3)

Canvas

futter
Service

-event: MouseEvent

↓

```
let x, number = -event.clientX  
let y, number = -event.clientY  
let food: Futter = new Futter()
```

food.x = x
food.y = y

bewObjArray.push(food)
food.draw()

↓

export

name
Eingeben

↓

```
SpielerName = prompt("text"),  
window.location.reload(),
```

↓

export

highscore
Anzeigen

↓

```
doc.getElementById("highscore")
```

↓

```
let prodElem: HTMLDivElement
```

↓

```
prodElem.innerHTML = `<div>${highscore}</div>`
```

↓

```
doc.getElementById("highscore").appendChild(prodElem)
```

Aktivitätsdiagramm (4)

Spielefisch
(Habe)

Fischfressen

(let is number = 0)



[i < namespace.boundaryArray.length

let xdis number = namespace.boundaryArray[i].x - this.x;

let ydis number = namespace.boundaryArray[i].y - this.y;

let dis number = Wurzel aus xdis im
Quadrat + ydis im Quadrat.



(mit allen Abgelenken -> X)



distanz < 20 && namespace.boundaryArray[i].type
== "kleiner Fisch")

namespace.boundaryArray.splice(i, 1);
namespace.highscore = highscore + 1;
namespace.highscoreAnzeigen();
let fisch = new Fisch 1();
push();

AD (5)

Client

insert

let query: String = "command=insert"

query += "SpielerName=" &name
"&name"

sendRequest(query, handleInsertResponse)



handle
Find
Response

-event: Progress Event

let xhr: XMLHttpRequest = (<XMLHttpRequest>_event.target)

xhr.readyState == XMLHttpRequest.DONE

let AlleSpieler: Spieler[] = JSON.parse(xhr.response)

let i: number = 0

AlleSpieler.length > i
AlleSpieler.sort(VergleicheHighscore)

i++

i = number

i < 6

Anzahl der
angezeigten
Highscores

let prodElem: HTMLDivElement = doc.createElement("div")
prodElem.innerHTML = "<div> ... </div>"

doc.getElementById("spielstaende").appendChild(prodElem)

AD (6)

Client

Vergleich
Highscore

a: Spieler, b: Spieler

let punkte A: number = a.punktzahl
let punkte B: number = b.punktzahl

[punkte A < punkte B] [punkte A > punkte B]

return 1

return 0

return -1

Server

handle
Request

- request: Http.IncomingMessage
- response: Http.ServerResponse

let query.punktzahl = <Punktzahl> Url.parse(-request.url, true)
let commandString = query["command"]

Switch command

[case "insert"]

[case "refresh"]

let spieler: Spieler =
name: query["name"]
punktzahl: parseInt(query["punktzahl"])

Database.insert(spieler)

respond(-response, "Antworttext")

break

respond
-response
"unknown
command"

break

Database.find
All (find call
back)

break

AD (7) Database

insert

-doc: Spieler

Spieler.insertOne(-doc, handleInsert)

prepare
Answer

-e: MongoError : SpielerArray : Spieler[]

[Error]

An den Callback den
JSONString weitergeben
dafür muss man aus
dem SpielerArray mittels
Stringify ein JSON string
erstellt werden

An den _callback
den Error / Fehler
weitergeben



