Speichern von Canvas Bildern

Globale Variablen werden angelegt:

```
interface Object {
type: String;
  x : String;
  y : String;
array : String;
arrayPos: String; }

interface CanvasElement {
 name: string;
Backgroundcolor: string;
CanvasWidth: string;
type: string;
  x  : string;
  y  : string;
array: string;
arrayPos: string;
```

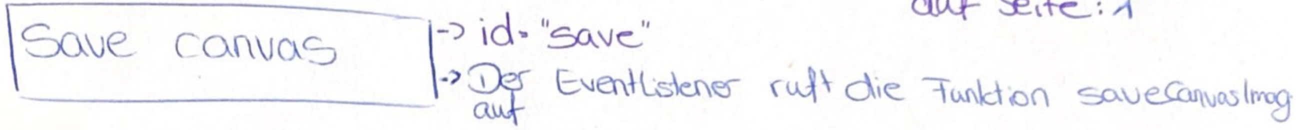
```
interface AssocStringString{
[key: string]: string; }
```

```
let rebuildArray: CanvasElement[];
let ElementNum: number = 0;
let buttonExists: boolean = false;
```

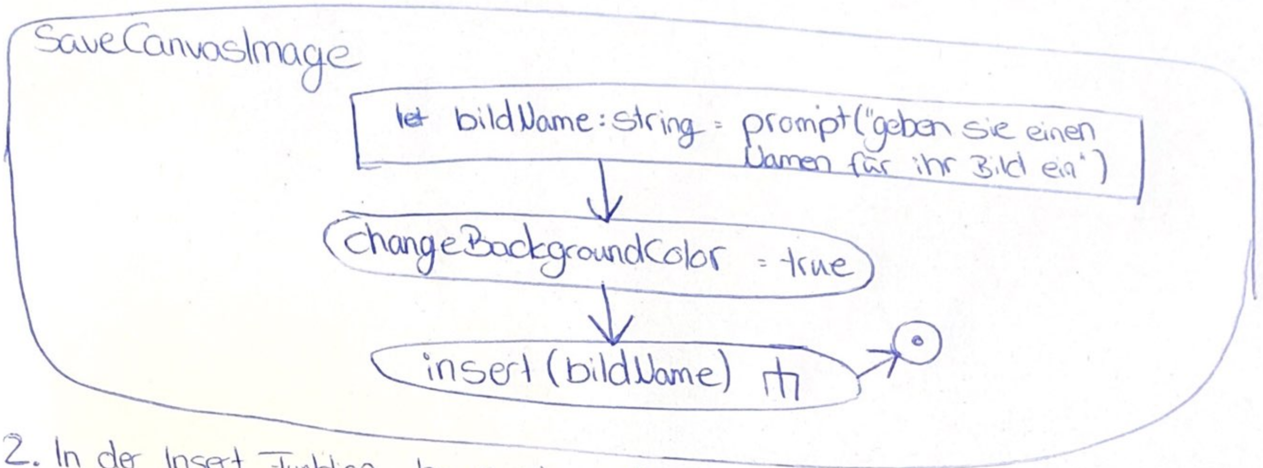
Konzept zum speichern von Canvas Bildern

1. Dem User steht ein Button zur verfügung, mit welchem er jederzeit sein Bild abspeichern kann und einen Namen geben kann.

*Globale Variablen auf Seite: 1

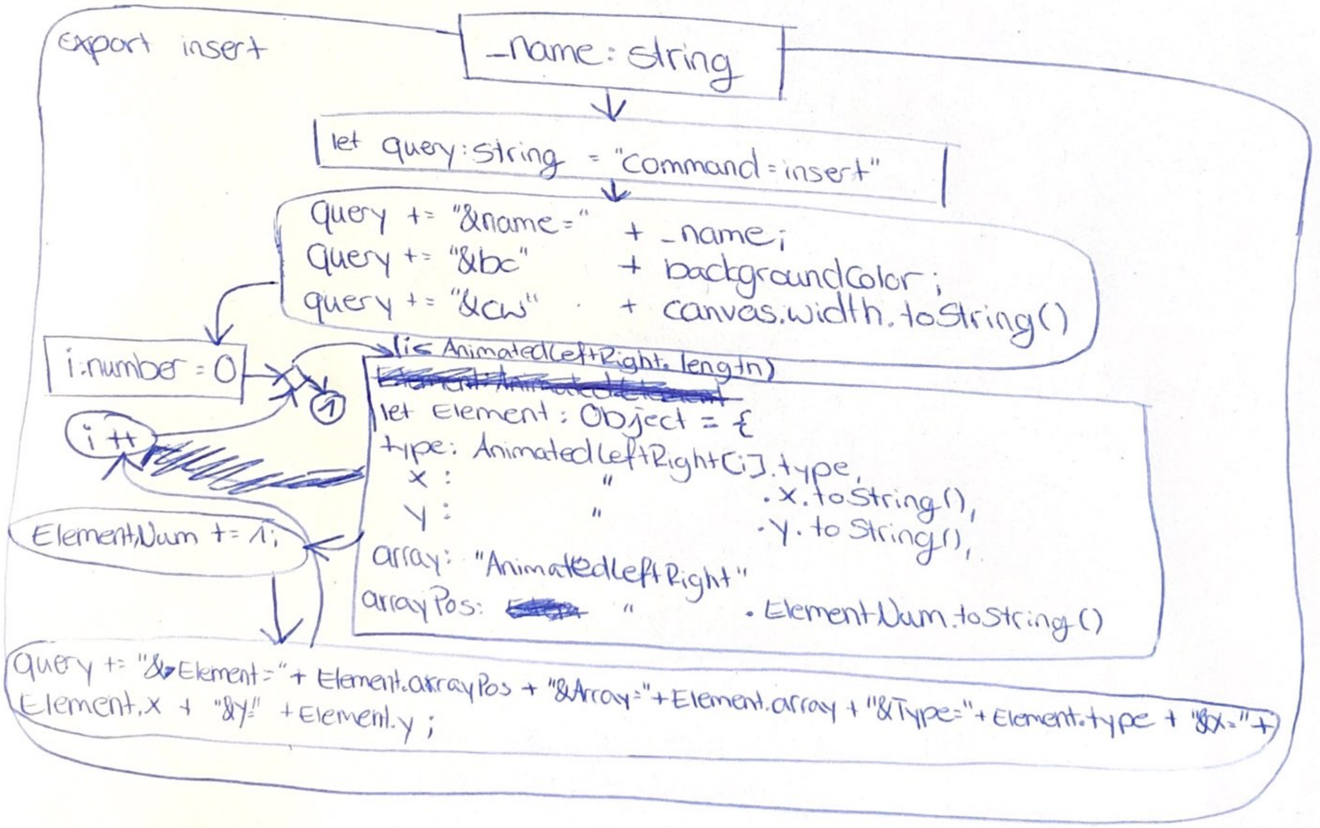


-> id: "save"
-> Der Eventlistener ruft die Funktion saveCanvasImag auf



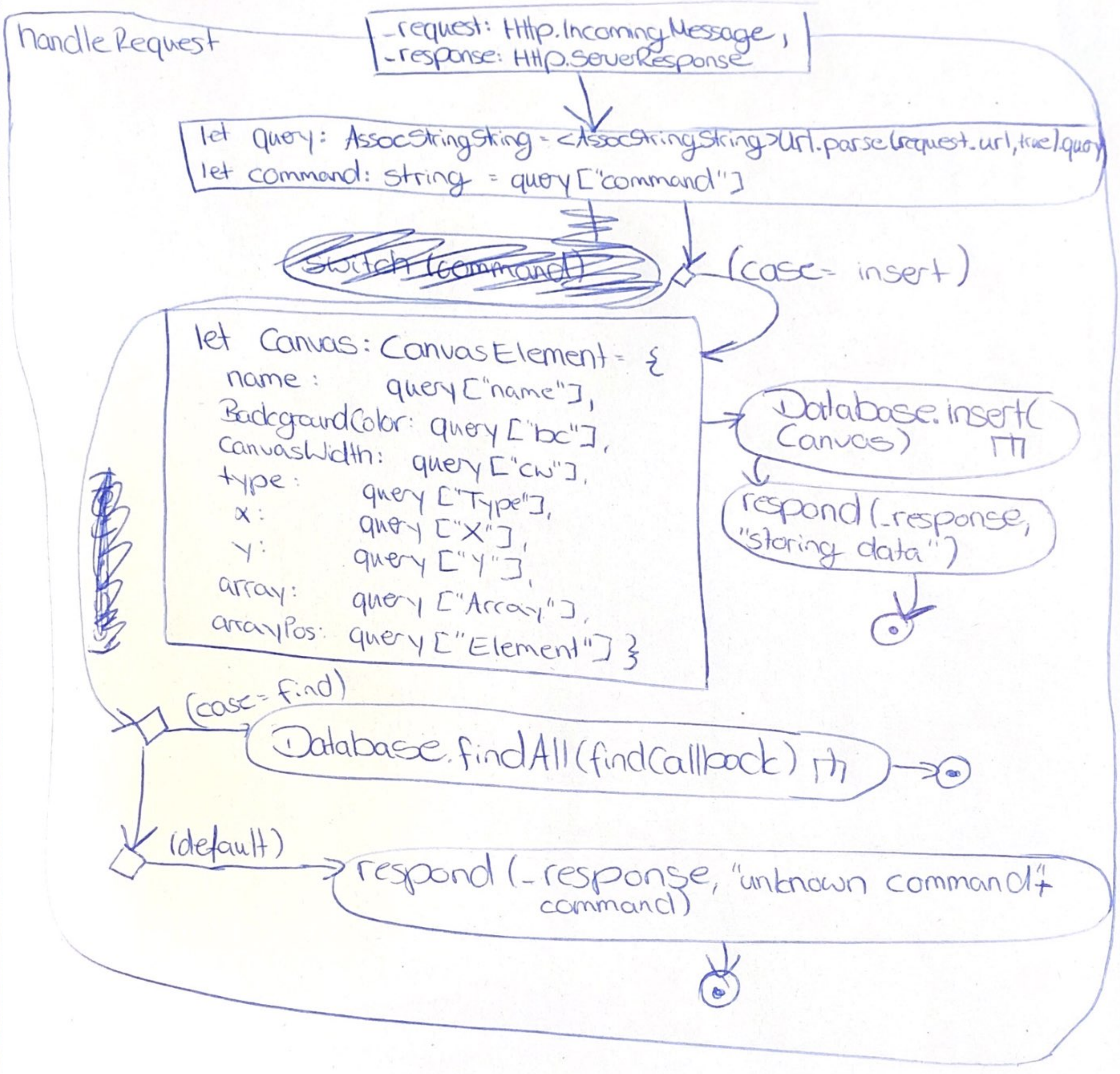
```
SaveCanvasImage
    let bildName: string = prompt("geben sie einen Namen für ihr Bild ein")
    ↓
    changeBackgroundColor = true
    ↓
    insert(bildName)
```

2. In der Insert Funktion baue ich meinen String so zusammen, dass ich alle nötigen Infos über mein Canvas Bild an den server senden kann

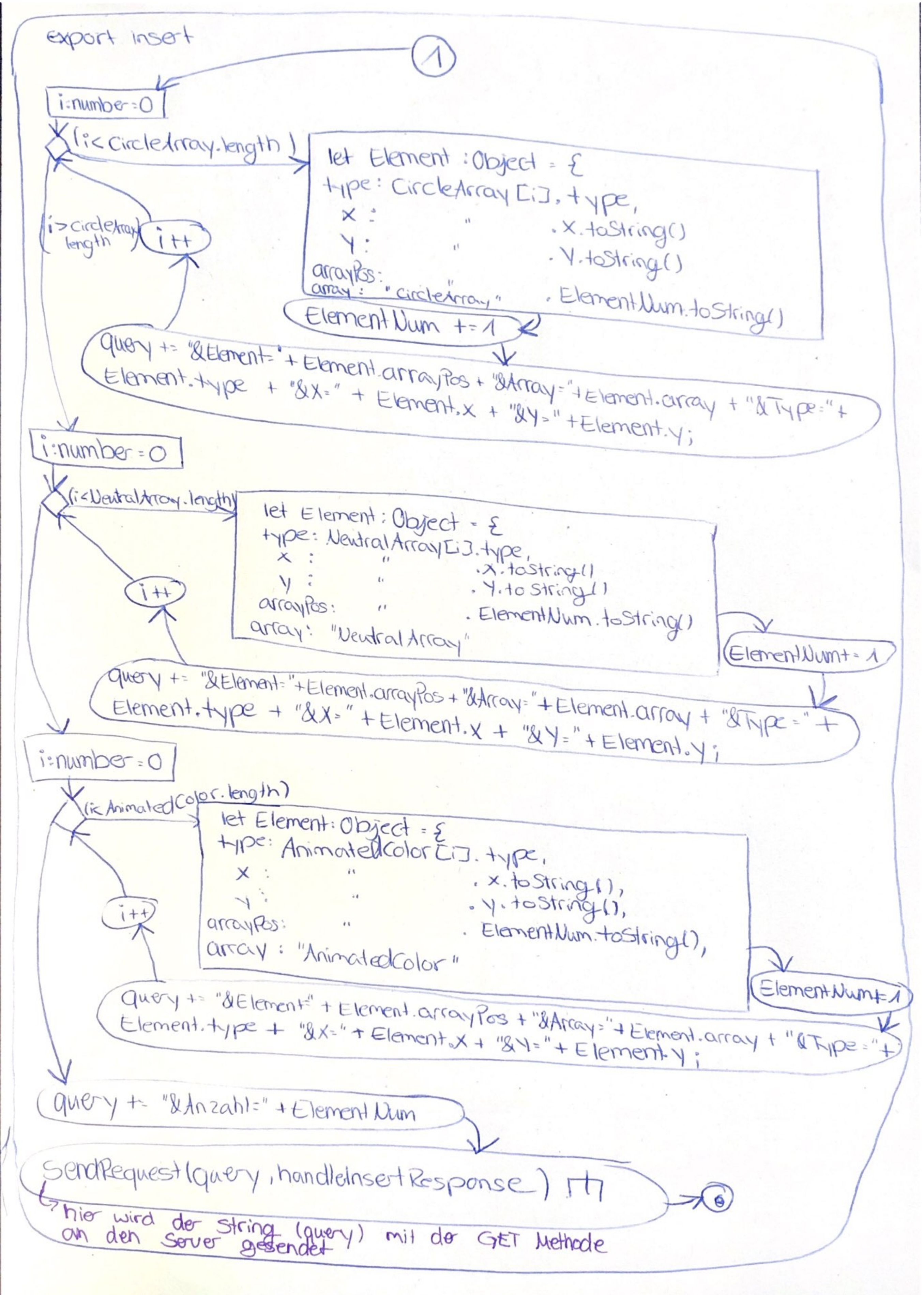


```
export insert    _name: string
    ↓
    let query: string = "command=insert"
    ↓
    query += "&name="   + _name;
    query += "&bc"      + backgroundColor;
    query += "&cw"      + canvas.width.toString()
    ↓
    i: number = 0
    (i< AnimatedLeftRight.length)
    ~~Element: AnimatedLeftRight~~
    let Element: Object = {
    type: AnimatedLeftRight[i].type,
      x:         "          .x.toString(),
      y:         "          .y.toString(),
    array: "AnimatedLeftRight"
    arrayPos: ~~Elem~~  "   .ElementNum.toString()
    ↓
    ElementNum += 1;
    ↓
    query += "&Element=" + Element.arrayPos + "&Array=" + Element.array + "&Type=" + Element.type + "&X=" +
    Element.x + "&Y=" + Element.y;
    ↓
    i ++
```

Der Server nimmt in diesem Fall in der handleRequest den command "insert" entgegen, wie es in der url steht.



```
handleRequest

_request: Http.IncomingMessage,
_response: Http.ServerResponse

let query: AssocStringString = <AssocStringString>Url.parse(request.url, true).query
let command: string = query["command"]

~~switch (command)~~

let Canvas: CanvasElement = {
  name:            query["name"],
  BackgroundColor: query["bc"],
  CanvasWidth:     query["cw"],
  type:            query["Type"],
  x:               query["X"],
  y:               query["Y"],
  array:           query["Array"],
  arrayPos:        query["Element"] }

Database.insert(Canvas)

respond(_response, "storing data")

Database.findAll(findCallback)

respond(_response, "unknown command" + command)
```



```
i:number=0

let Element :Object = {
type: CircleArray[i].type,
  x :          "          .x.toString()
  y :          "          .y.toString()
arrayPos:                  
array : "circleArray"      .ElementNum.toString()

ElementNum += 1

query += "&Element="+ Element.arrayPos + "&Array="+Element.array + "&Type="+
Element.type + "&X=" + Element.x + "&Y="+Element.y;

i:number=0

let Element : Object = {
type: NeutralArray[i].type,
  x  :       "          .x.toString()
  y  :       "          .y.toString()
arrayPos:    "          .ElementNum.toString()
array: "NeutralArray"

ElementNum+= 1

query += "&Element="+Element.arrayPos + "&Array="+Element.array + "&Type="+
Element.type + "&X=" + Element.x + "&Y="+Element.y;

i:number=0

let Element: Object = {
type: AnimatedColor[i].type,
  x :          "          .x.toString(),
  y :          "          .y.toString(),
arrayPos:      "          .ElementNum.toString(),
array : "AnimatedColor"

ElementNum+=1

query += "&Element" + Element.arrayPos + "&Array="+Element.array + "&Type="+
Element.type + "&X="+ Element.x + "&Y="+ Element.y;

query += "&Anzahl=" + ElementNum

SendRequest(query, handleInsertResponse)
```

↳ hier wird der String (query) mit der GET Methode an den Server gesendet