Le front-end

par Anthony Maurial

Sommaire

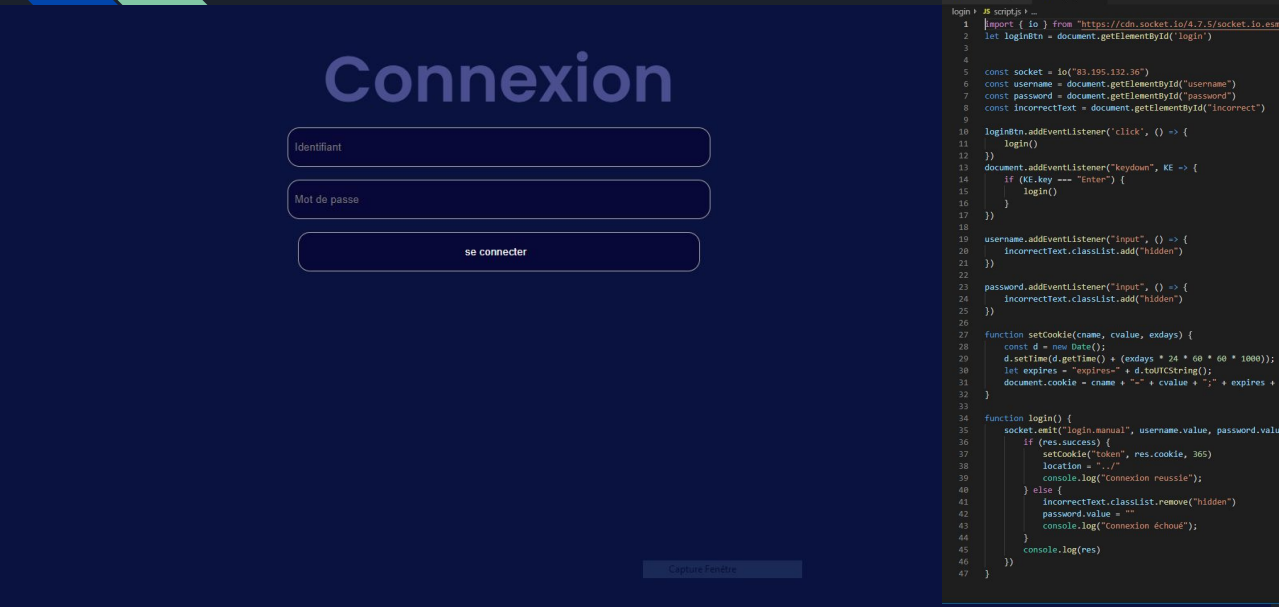
-introduction

-les détails techniques

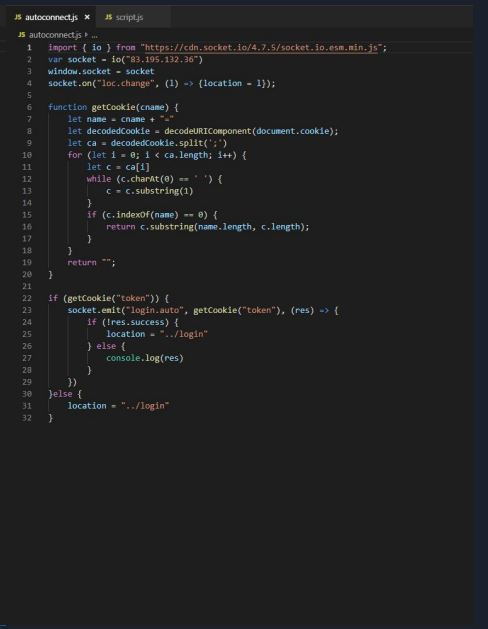
-les avancés

introduction

les détails techniques

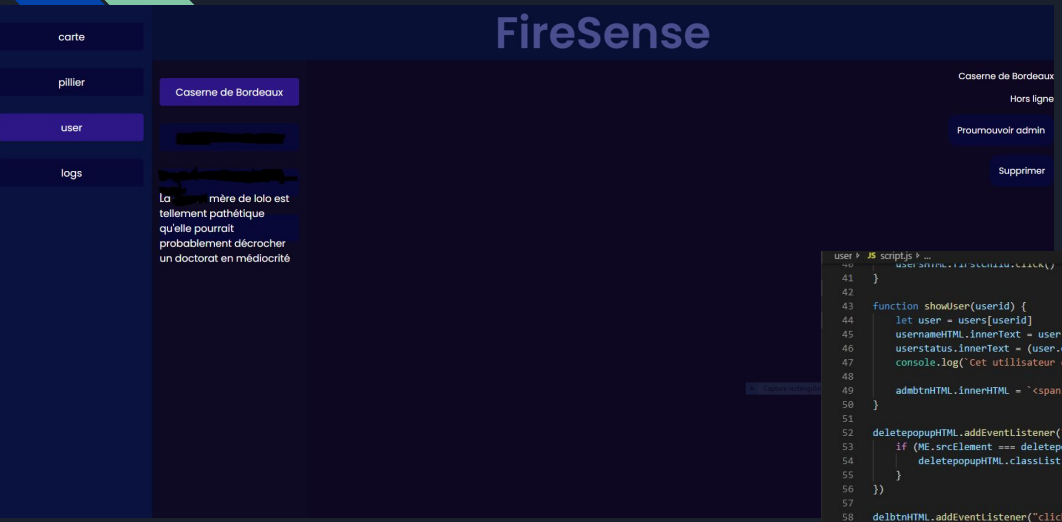



```
import { io } from "https://cdn.socket.io/4.7.5/socket.io.esm.min.js";
let loginBtn = document.getElementById('login')


const socket = io("83.195.132.36")
const username = document.getElementById("username")
const password = document.getElementById("password")
const incorrectText = document.getElementById("incorrect")

loginBtn.addEventListener('click', () => {
    login()
})
document.addEventListener("keydown", KE => {
    if (KE.key === "Enter") {
        login()
    }
})

username.addEventListener("input", () => {
    incorrectText.classList.add("hidden")
})

password.addEventListener("input", () => {
    incorrectText.classList.add("hidden")
})

function setCookie(cname, cvalue, exdays) {
    const d = new Date();
    d.setTime(d.getTime() + (exdays * 24 * 60 * 60 * 1000));
    let expires = "expires=" + d.toUTCString();
    document.cookie = cname + "=" + cvalue + ";" + expires + ";path=/";
}

function login() {
    socket.emit("login.manual", username.value, password.value, (res) => {
        if (res.success) {
            setCookie("token", res.cookie, 365)
            location = "../"
            console.log("Connexion reussie");
        } else {
            incorrectText.classList.remove("hidden")
            password.value = ""
            console.log("Connexion échoué");
        }
        console.log(res)
    })
}
```



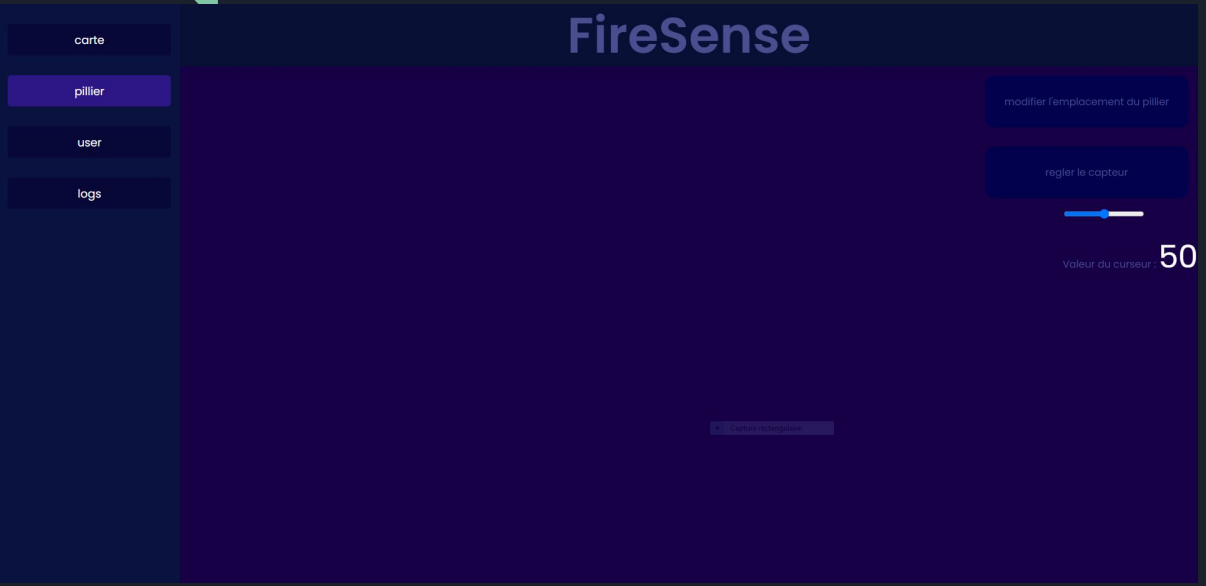
```
import { io } from "https://cdn.socket.io/4.7.5/socket.io.esm.min.js";
var socket = io("83.195.132.36")
window.socket = socket
socket.on("loc.change", (l) => {location = l});

function getCookie(cname) {
    let name = cname + "="
    let decodedCookie = decodeURIComponent(document.cookie);
    let ca = decodedCookie.split(';')
    for (let i = 0; i < ca.length; i++) {
        let c = ca[i]
        while (c.charAt(0) == ' ') {
            c = c.substring(1)
        }
        if (c.indexOf(name) == 0) {
            return c.substring(name.length, c.length);
        }
    }
    return "";
}

if (getCookie("token")) {
    socket.emit("login.auto", getCookie("token"), (res) => {
        if (!res.success) {
            location = "../login"
        } else {
            console.log(res)
        }
    })
}else {
    location = "../login"
}
```

les détails techniques




```
user › JS script.js › ...
41  }
42
43  function showUser(userid) {
44      let user = users[userid]
45      usernameHTML.innerText = user.name
46      userstatus.innerText = (user.online ? "En ligne" : "Hors ligne")
47      console.log(`Cet utilisateur est: ${(user.admin ? "Admin" : "Rien")}`)
48
49      admbtnHTML.innerHTML = `<span>${user.admin ? "Révoquer admin" : "Proumouvoir admin"}</span>`
50  }
51
52  deletepopupHTML.addEventListener("click", (ME) => {
53      if (ME.srcElement === deletepopupHTML) {
54          deletepopupHTML.classList.add("hidden")
55      }
56  })
57
58  delbtnHTML.addEventListener("click", () => {
59      if (actual.id) {
60          deletepopupHTML.classList.remove("hidden")
61          popupText.innerText = `Voulez-vous vraiment supprimer ${users[actual.id].name} ?`
62          actualAction = "delete"
63      }
64  })
65  admbtnHTML.addEventListener("click", () => {
66      if (actual.id) {
67          deletepopupHTML.classList.remove("hidden")
68          popupText.innerText = `Voulez-vous vraiment ${users[actual.id].admin ? "révoquer les droits d'admin à" : "proumouvoir"} ${users[actual.id].name} ${users[actual.id].admin
69          actualAction = "admin.change"
70      }
71  })
72  delCancel.addEventListener("click", () => {
73      deletepopupHTML.classList.add("hidden")
74      actualAction = undefined
75  })
76
77  delConfirm.addEventListener("click", () => {
78      let user = users[actual.id]
79      socket.emit(`user.${actualAction}`, actual.id, (success) => {
80          console.log(`${actualAction} success: ${success}`);
81          deletepopupHTML.classList.add("hidden")
82      })
83  })
84
```

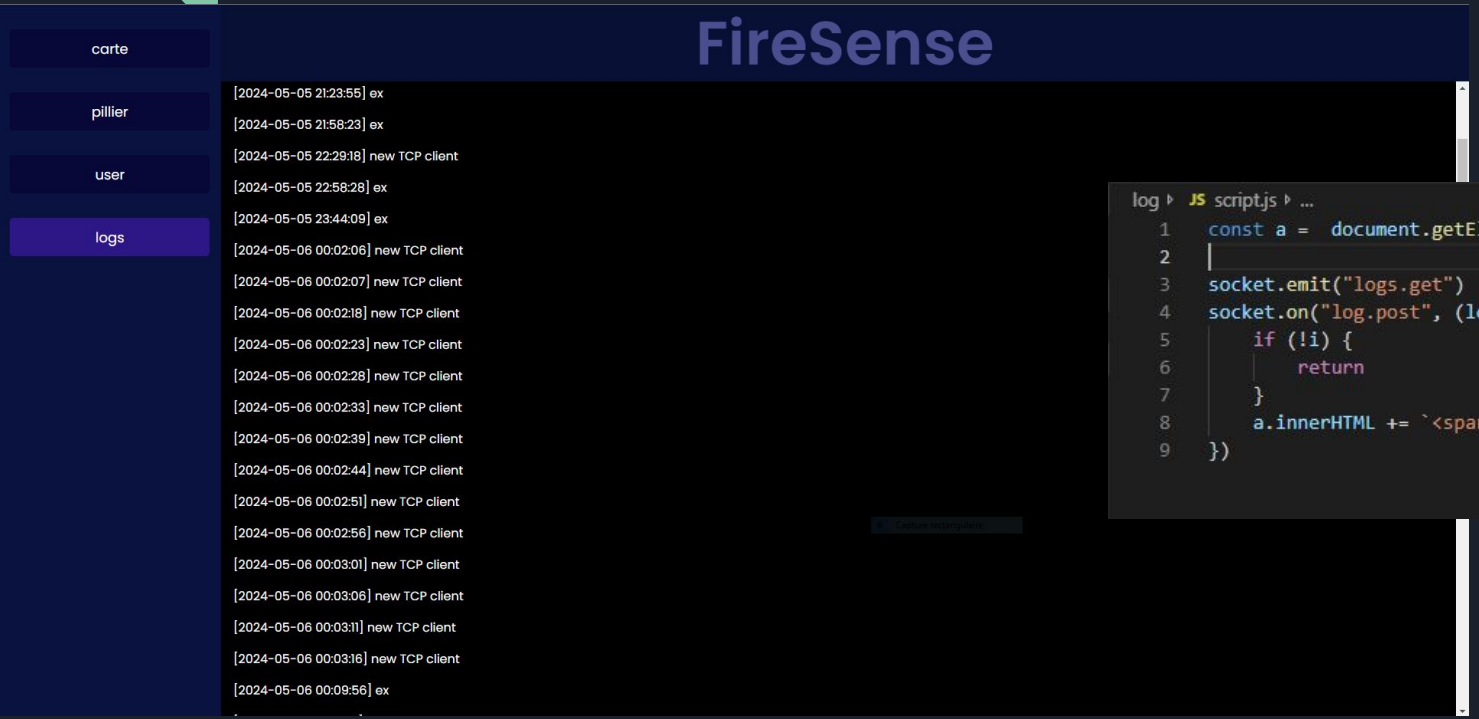
les détails techniques



pillier ▸ JS script.js ▸ socket.emit("pylons.get") callback

```
19  })
20
21  popupReload.addEventListener("click", () => {
22      location.reload()
23  })
24
25  let bg = new Image()
26  bg.src = "../pillier.png"
27
28  let ptx_selected = {
29      x: 0,
30      y: 0,
31      show: false,
32      ptx: null,
33      mousedown: false,
34  }
35
36  socket.emit("pylons.get", (pt) => {
37      ptx = pt
38      console.log(pt);
39  })
40
41  locationBTN.addEventListener("click", () => {
42      canvas.classList.toggle("hidden")
43  })
44
45  let ptx = {}
46
47  function draw() {
48      requestAnimationFrame(draw)
49      ctx.drawImage(bg, 0, 0)
50
51      if (ptx_selected.show) {
52          drawPoint(ptx_selected.x, ptx_selected.y, 11, "white")
53      }
54      for (const id in ptx) {
55          let pt = ptx[id]
56          drawPoint(pt.pos.x * 10, pt.pos.y * 10, 10)
57      }
58  }
59
60  function drawPoint(x, y, pointSize, color = "red") {
61      ctx.fillStyle = color
62      ctx.beginPath()
63      ctx.arc(x, y, pointSize, 0, Math.PI * 2)
64      ctx.fill()
65  }
66  draw()
67
```

les détails techniques



log ▸ JS script.js ▸ ...

```
1  const a =  document.getElementById("root")
2
3  socket.emit("logs.get")
4  socket.on("log.post", (log, i) => {
5      if (!i) {
6          return
7      }
8      a.innerHTML += `<span class="log">${log}<span>`
9  })
```

les avancés