Local storage and session storage

```
<script>
    if (!window.localStorage.getItem("someItem"))
        window.localStorage.setItem("someItem", 10);
    else
        console.log(window.localStorage.getItem("someItem"));
    setTimeout(() => {
        window.localStorage.removeItem("someItem");
        window.localStorage.clear();
        window.sessionStorage.clear()
    }, 2000);
</script>
```

Valjanost lozinke

```
const util = require("util");
const crypto = require("crypto");

const randomBytesAsync = util.promisify(crypto.randomBytes);
const pbkdf2Async = util.promisify(crypto.pbkdf2);


// lozinka pri registraciji
const password = "123456";

// slucajno generirano na serveru, sprema se u bazu, i tamo ostaje
// ovo je na blokirajući način
// const saltB = crypto.randomBytes(16).toString("hex");
// bolje
const salt = (await randomBytesAsync(16)).toString("hex");

// hash se sprema u bazu
// na blokirajuć način
// const hashB = crypto.pbkdf2Sync(password, salt, 1000, 64, "sha512")
//    .toString("hex");
// bolje
const hash = (
    await pbkdf2Async(password, salt, 1000, 64, "sha512")
).toString("hex");

// lozinka pri prijavi
const another_password = "123456";
const another_hash = (
    await pbkdf2Async(another_password, salt, 1000, 64, "sha512")
).toString("hex");

// check
if (hash === another_hash)
console.log("Correct password");
else console.log("Incorrect password");
```

Kreiranje tokena

1. npm install jsonwebtoken
2. `const jwt = require("jsonwebtoken");`
3. vrijednost koju zna samo onaj tko izdaje token:

```
const secret = "ratherStoreInFile";
```

4. Stvoriti:

```
const jwtSignAsync = util.promisify(jwt.sign);
const jwtVerifyAsync = util.promisify(jwt.verify);
…
app.post("/api/login", async (req, res) => {
    // pravimo se da smo ovo dobili pri loginu
    const username = "jmaltar";
    // pravimo se da je korisnik upisao dobru lozinku
    const password_correct = true;
    // pravimo se da smo iz baze podataka dohvatili korisnikov _id
    const _id = "123";

    if (password_correct) {
        // expires for 1 hour
        const exp = Math.floor(Date.now() / 1000) + 60 * 60;
        const token = await jwtSignAsync({ _id, username, exp }, secret);

        res.status(200).json({ message: "Authorized", token });
    } else res.status(401).json({ message: "Unauthorized" });
});
```

5. Demonstrirati u jQuery-ju:

```
const login = async () => {
    try {
        const res = await $.post("/api/login", { username: "jmaltar" });
        return res;
    } catch(e) {
        return null;
    }
}

$(async () => {
    const res = await login();
    if (res) {
        token = res.token;
    } else { // redirect to login page }
});
```

6. Dohvatiti token u developer konzoli preglednika

7. Demnostrirati parsiranje payload-a
   ```
   const payload = JSON.parse(window.atob(token.split(".")[1]));
   ```

8. Spremiti token u localStorage (paziti na url, može i u konzoli, ali bolje u kodu):
   ```
   window.localStorage.setItem("myAppToken", token);
   ```

9. Ugasiti preglednik, pa ga upaliti pa dohvatiti token (opet isti url u navigacijskoj traci):
   ```
   window.localStorage.getItem("myAppToken");
   ```

10. Definirati:

```
const logout = () => {
    window.localStorage.removeItem("myAppToken");
}

const save_token = (token) => {
    window.localStorage.setItem("myAppToken", token);
}

const get_token = () => {
    return window.localStorage.getItem("myAppToken");
}
```

11. Novi main:

```
$(async () => {

    logout();

    const res = await login();
    save_token(res.token);

    // logout();
    console.log(get_token());

});
```

12. Na serveru implementirati restringiranu rutu:

```
app.get("/api/restricted", async (req, res) => {
    const token = req.headers["authorization"];
    try {
        await jwtVerifyAsync(token, secret);
        res.send("Confidential information");
    } catch(e) {
        res.status(401).json({ message: "Unauthorized" });
    }
});
```

13. U main-u na klijentu:

```
    const res_ = await $.ajax({
        url: "/api/restricted",
        method: "GET",
        data: {},
        headers: {
            "authorization": get_token()
        }
    });
```

14. Middleware:

```
const jwt_protection = async (req, res, next) => {
    const token = req.headers["authorization"];
    try {
        await jwtVerifyAsync(token, secret);
        next();
    } catch(e) {
        res.status(401).json({ message: "Unauthorized" });
    }
}


app.get("/api/restricted", jwt_protection, async (req, res) => {
    res.send("Confidental information");
});
```