

# Sessions and Authentication

State Persistence and Application Security



**SoftUni Team**  
**Technical Trainers**



**SoftUni**



**Software University**

<https://softuni.bg>

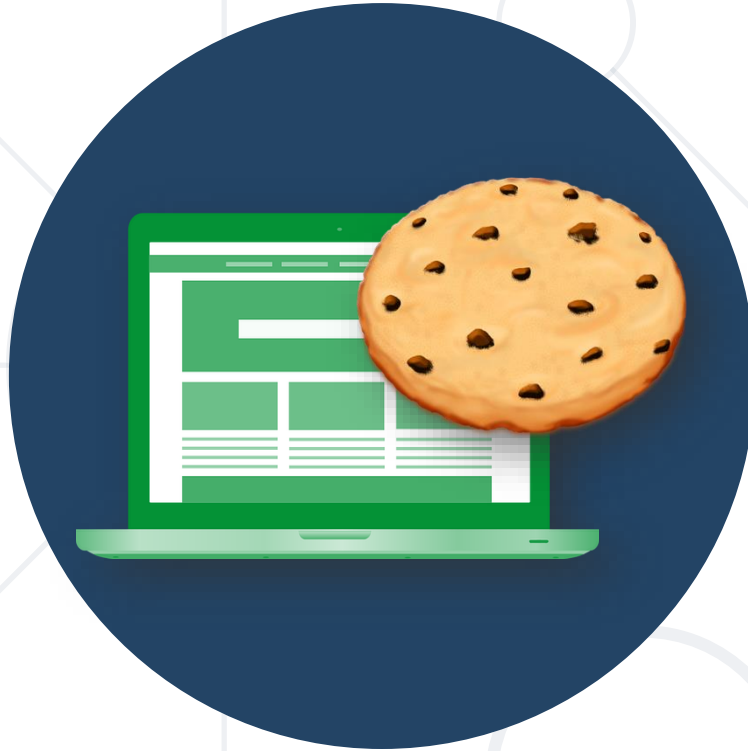
sli.do

**#js-back-end**

# Table of Contents

1. **Cookies** and **Sessions**
2. **Authentication** Concepts
3. **JSON Web Token**





# Cookies and Sessions

- HTTP is **stateless**
  - The Server and Client **don't remember** each other across requests
- To preserve state, **cookies** are stored on the **Client**
- **Session cookie** - exists on the **Server**
  - It can **store information** about a Client
  - Used to **persist state** across requests
  - Matched to a Client by their **cookie**

- **Session** is preferred when you need to store **short-term** information / values
- **Cookies** are preferred when you need to store **long-term** information / values
- Session is **safer** because is stored on the server
- Cookies are not very safe. Expiration **can be set**, and it can last for years

- You can use **cookie-parser** middleware for Express

```
npm install cookie-parser --save-exact
```

```
// use in an express app  
const cookieParser = require('cookie-parser')  
app.use(cookieParser())  
  
app.get('/setCookie', (req, res) => {  
  res.cookie("message", "hello")  
  res.end('Cookie set')  
})  
  
app.get('/readCookie', (req, res) => {  
  res.json(req.cookies)  
})
```

- You can use **express-session** middleware for Express

```
npm install express-session --save-exact
```

```
// use in an express app  
const session = require('express-session')  
app.use(session({ secret: 'my secret',  
                 resave: false,  
                 saveUnitialized: true,  
                 cookie: {secure: true } })))  
app.get('/setSession', (req, res) => {  
  req.session.message = "hello"  
  res.end('Session set')  
})  
  
app.get('/readSession', (req, res) => {  
  res.json(req.session)  
})
```





# **Authentication Concepts**

- **Authentication** is an important part of **application security**
- It serves to verify whether the **client** is who or what it declares itself to be
- It's built on several **layers of abstraction**
  - Cookies → Sessions → Security
- Authentication is **a cross-cutting concern**, best handled away from business logic
  - Request → Authentication → Business Logic → Response

- **Bcrypt** is a password hashing function
  - Besides incorporating **salt** to protect against **rainbow table** attacks, **bcrypt** is an adaptive function
    - Over time, the iteration count can be increased to make it slower, so it remains resistant to **brute-force search attacks** even with increasing computation power



## ■ Installation

```
npm install bcrypt
```

## ■ Hash password

```
const bcrypt = require('bcrypt');  
const saltRounds = 9;  
const myPlainTextPassword = "password123";  
  
bcrypt.genSalt(saltRounds, (err, salt) => {  
  bcrypt.hash(myPlainTextPassword, salt, (err, hash) => {  
    console.log(hash);  
    // $2b$09$pdhUAoT4qE0tmku.ZkXWR0eLcJCy.LDRq.1I4IVImjrUTGuUbYQM  
  }));  
});
```

- Check password

```
const myPlainTextPassword = "password123";  
const hash = "$2b$09$pdhUAoT4qE0tmku.ZkXWROeLcJCy.LDRq.1I4IVImjrUTGuUbYQMi";  
  
bcrypt.compare(myPlainTextPassword, hash, (err, res) => {  
  console.log(res); // true  
});
```

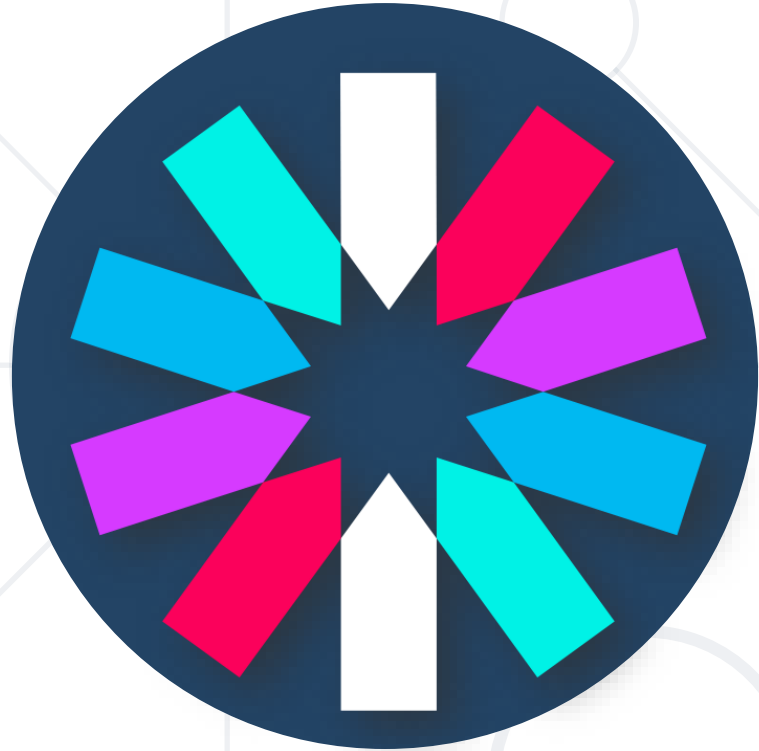
- **Async** way is recommended to **hash** and **check** password

## ■ Authentication

- The process of **verifying the identity** of a user or computer
- Questions: "**Who are you?**", "**How do you prove it?**"
- Credentials can be a password, smart card, external token, etc...

## ■ Authorization

- The process of determining what a user is **permitted** to do on a computer or network
- Questions: "**What are you allowed to do?**", "**Can you see this page?**"



**JSON Web Token**

# What is JWT?

- **JSON Web Token (JWT)** is an open standard that defines a compact and self-contained way for securely transmitting information between parties as a JSON object
- This information can be verified and trusted because it is digitally signed
- JWTs can be signed using a secret or a public/private key pair using **RSA** or **ECDSA**



# When Should You Use JWT?

- JSON Web Tokens are useful for
  - **Authorization** (a most common scenario) - Once the user is logged in, each subsequent request will include JWT, allowing the user to access routes, services, and resources that are permitted with that token
  - **Information Exchange** - JSON Web Tokens are good way of securely transmitting information between parties. Because they are signed digitally

- In its compact form, JSON Web Tokens consist of three parts separated by dots ( . )
  - **Header**
  - **Payload**
  - **Signature**

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.  
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4  
gRG9lIiwiaXNTb2NpYWwiOnRydWV9.  
4pcPyMD09o1PSyXnrXCjTwXyr4Bsezdi1AVTmud2fU4
```

- Encode token

```
const jwt = require('jsonwebtoken');

const payloads = { _id, username };
const options = { expiresIn: '2d' };
const secret = 'MySuperPrivateSecret';
const token = jwt.sign(payload, secret, options);
```

```
//eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJwYXkiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.xzK8LJQz0LDkJqsng04BYxcUQzxWngyEBP
```

- Decode token

```
const token = req.cookies['token'] || sessionStorage.getItem('token');  
// Depends where you store the token..  
  
const decodedToken = jwt.verify(token, secretKey);  
  
console.log(decodedToken); // { _id: ..., username: ... }
```

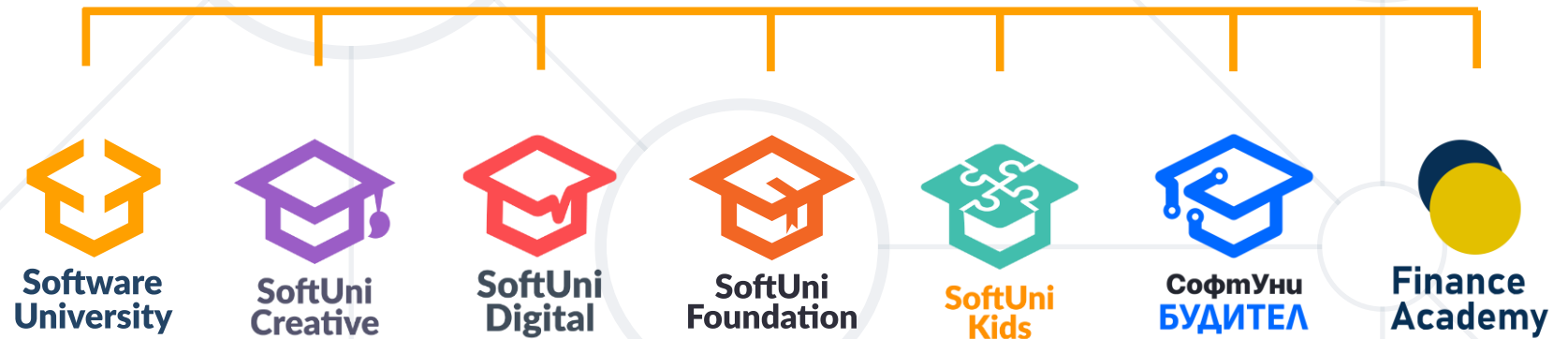
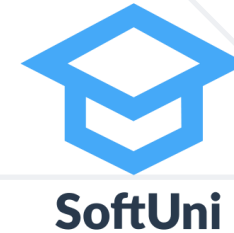
- More about JWT, you can find

- <https://jwt.io/>
- <https://www.npmjs.com/package/jsonwebtoken>

- **Cookies and Sessions**
  - Definitions and Usage
  - Cookies vs Sessions
- **Authentication Concepts**
  - Application Security with bcrypt
- **JSON Web Token**
  - What is JWT?
  - Structure and Usage



# Questions?



# SoftUni Diamond Partners



- Software University – High-Quality Education, Profession and Job for Software Developers

- [softuni.bg](https://softuni.bg)

- Software University Foundation

- [softuni.foundation](https://softuni.foundation)

- Software University @ Facebook

- [facebook.com/SoftwareUniversity](https://facebook.com/SoftwareUniversity)





- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://about.softuni.bg/>
- © Software University – <https://softuni.bg>

