Welcome to the

CoGrammar

Tutorial: MERN with JWT

The session will start shortly...

Questions? Drop them in the chat. We'll have dedicated moderators answering questions.

CoGrammar

Full Stack Web Development Session Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly. **(Fundamental British Values: Mutual Respect and Tolerance)**
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Academic Sessions. You can submit these questions here: **Questions**

Full Stack Web Development Session Housekeeping cont.

- For all **non-academic questions**, please submit a query: www.hyperiondev.com/support
- Report a **safeguarding** incident: www.hyperiondev.com/safeguardreporting
- We would love your **feedback** on lectures: [Feedback on Lectures]()

Skills Bootcamp
8-Week Progression Overview

Fulfil 4 Criteria to Graduation

✅ Criterion 1: Initial Requirements

Timeframe: First 2 Weeks
Guided Learning Hours (GLH): Minimum of 15 hours
Task Completion: First four tasks

Due Date: 24 March 2024

✅ Criterion 2: Mid-Course Progress

60 Guided Learning Hours

Data Science - **13 tasks**
Software Engineering - **13 tasks**
Web Development - **13 tasks**

Due Date: 28 April 2024

CoGrammar

Skills Bootcamp Progression Overview

✅ Criterion 3: Course Progress

Completion: All mandatory tasks, including Build Your Brand and resubmissions by study period end
Interview Invitation: Within 4 weeks post-course
Guided Learning Hours: Minimum of 112 hours by support end date (10.5 hours average, each week)

✅ Criterion 4: Demonstrating Employability

Final Job or Apprenticeship Outcome: Document within 12 weeks post-graduation
Relevance: Progression to employment or related opportunity

CoGrammar

Introduction to Authentication

CoGrammar

Authentication

Definition and importance

- **Authentication** involves verifying the identity of users to access an application (or website).
- This ensures the security and integrity of online systems by allowing only authorized users to access protected resources.

CoGrammar

Authentication

Definition and importance

- Importance of Authentication:
  - **Security:** protection against unauthorized access ensures only authorized individuals can access sensitive information.
  - **User Trust and reputation:** strong authentication builds trust with customers demonstrating an organisation’s commitment to security.
  - **Compliance:** Many regulations and laws require organisations to protect sensitive information.

Authentication

Common authentication methods

- ❖ Authentication methods:
  - ➢ **Username/Password based auth:** Most traditional method where users give the username/email and password for identification.
  - ➢ **OAuth:** Use of third party applications to access a user's resources without sharing their credentials.
  - ➢ **Token Based Authentication:** Using a unique token to authenticated users to include in subsequent requests to access protected routes.
  - ➢ **Multi-factor Authentication (MFA):** Adding an extra layer of security by requiring users to provide multiple forms of verification.

Token based Authentication (JSON Web Tokens)

CoGrammar

JSON Web Tokens (JWT)

Definition and comparison to other authentication methods

- ❖ How basic authentication with tokens work:
  - ➢ The client sends the username and password to an authentication endpoint
  - ➢ The auth endpoint checks the data and if legit, generates an auth token which is relevant to the requesting user’s session
  - ➢ The client stores the token and adds it to the header of further requests
  - ➢ The server checks the token every time it receives a request and uses it to determine which user is making the request.

CoGrammar

JSON Web Tokens (JWT)

Definition and comparison to other authentication methods

❖ In basic authentication, where the username and password were passed in the headers of the url, the password becomes interceptable as it is passed as plain text when you use **(http)** instead of **(https).**

❖ The use of JWT ensures safety as it transmits information between parties securely in a JSON object.

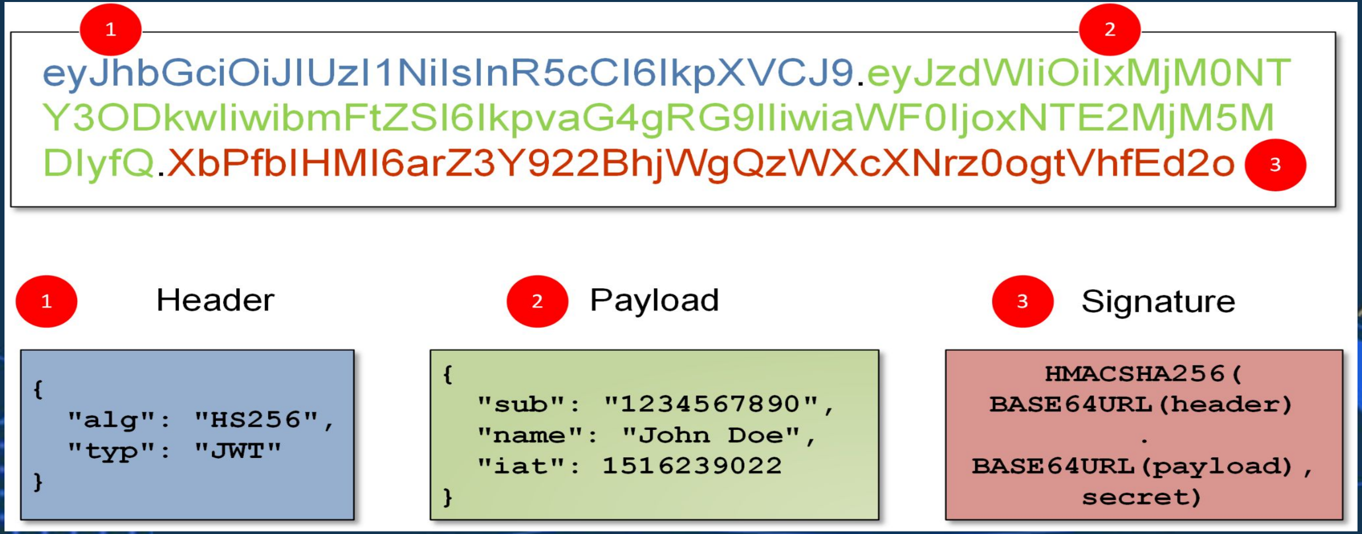
❖ JWTs are usually signed, this means you can be certain that the senders are who they say they are.

❖ Additionally, the structure of a JWT allows you to verify that the content hasn't been tampered with.

CoGrammar

Structure of a JWT

- **Header:** Contains the signing algorithm and type of token (JWT)
- **Payload:** Contains the claims or the JSON object
- **Signature:** String generated by cryptographic algorithm to verify integrity.



```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NT
Y3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5M
DIyfQ.XbPfbIHMI6arZ3Y922BhjWgQzWXcXNrz0ogtVhfEd2o
```

1 Header

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

2 Payload

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
```

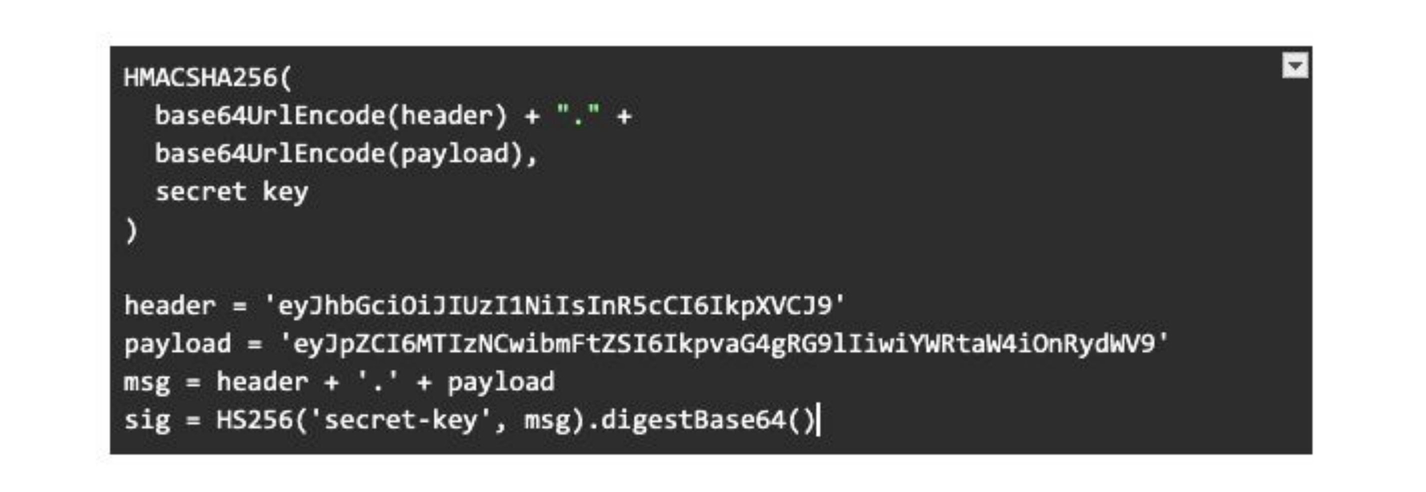
3 Signature

```
HMACSHA256(
BASE64URL(header)
.
BASE64URL(payload),
secret)
```

CoGrammar

Structure of a JWT

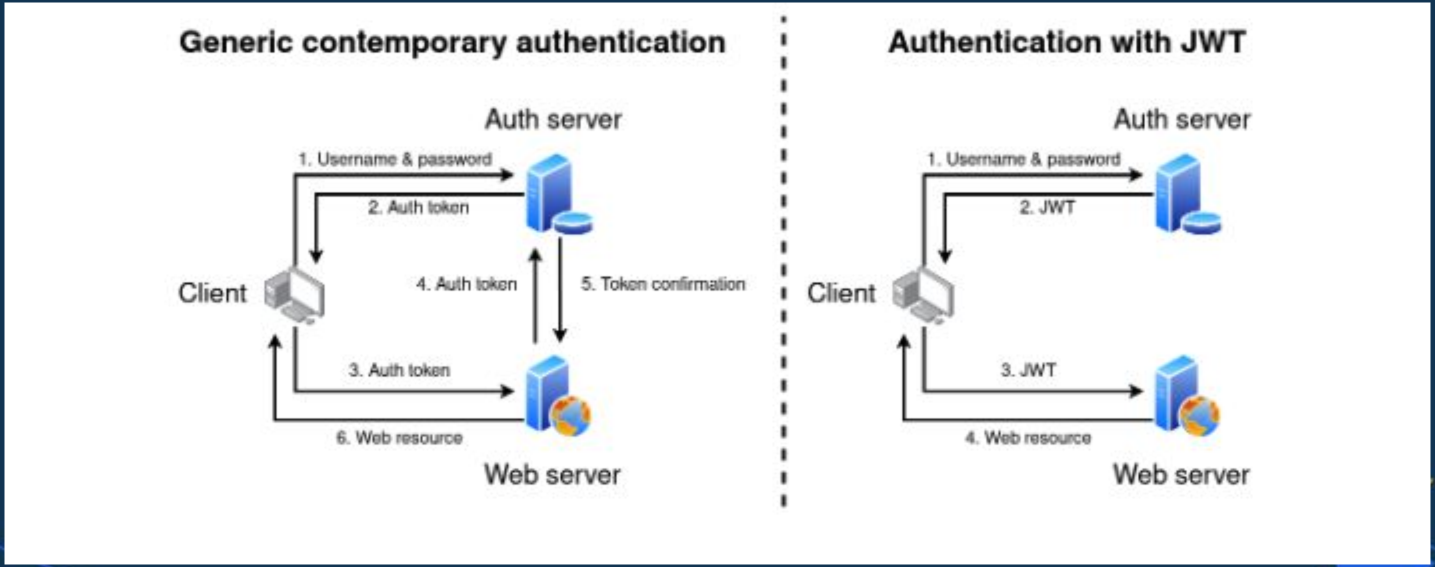
- Combining the JSON objects previously shown creates our JWT, but before combining, we first need to base64 encode the information of the header and payload and concatenate them with full stops together with the secret key. The signature will be made by the HMACSHA256() function.



```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  secret key
)

header = 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9'
payload = 'eyJpZCI6MTIzNCwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWV9'
msg = header + '.' + payload
sig = HS256('secret-key', msg).digestBase64()
```

CoGrammar

How JWT performs over basic authentication mechanism



Source: Radix

CoGrammar

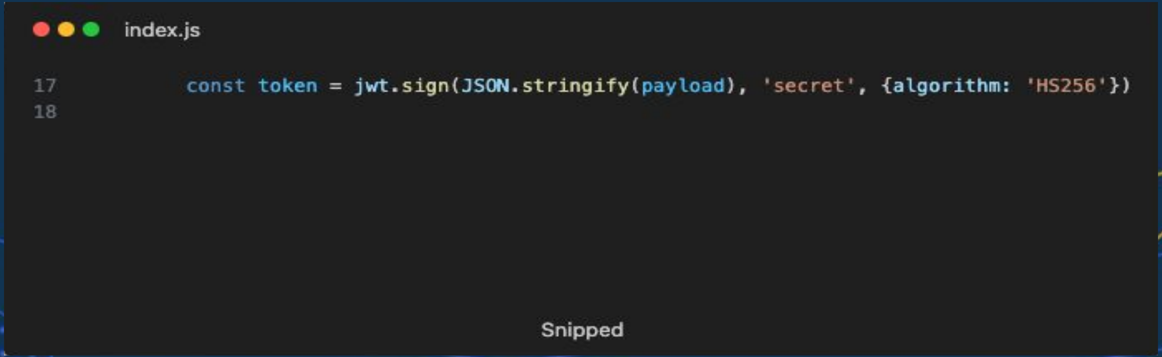
Implementing JWT

CoGrammar

JSON Web Tokens (JWT)

Implementing JWT.

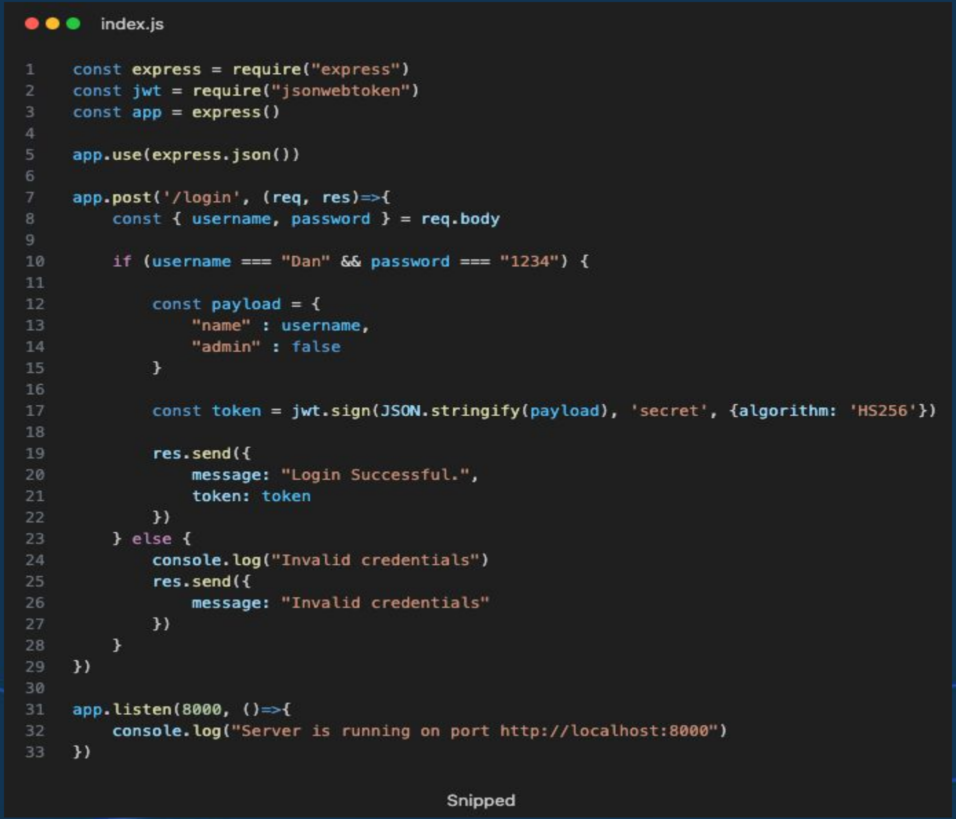
- We will use a popular library to implement JWTs in our application, it makes it easier to sign the tokens and reduces boilerplate code.
- You first need to install it in an already existing express application.
    - `npm install jsonwebtoken`
- Implementing JWT with the library becomes straightforward in this manner



```
        const token = jwt.sign(JSON.stringify(payload), 'secret', {algorithm: 'HS256'})
```

JSON Web Tokens (JWT)

Implementing JWT.

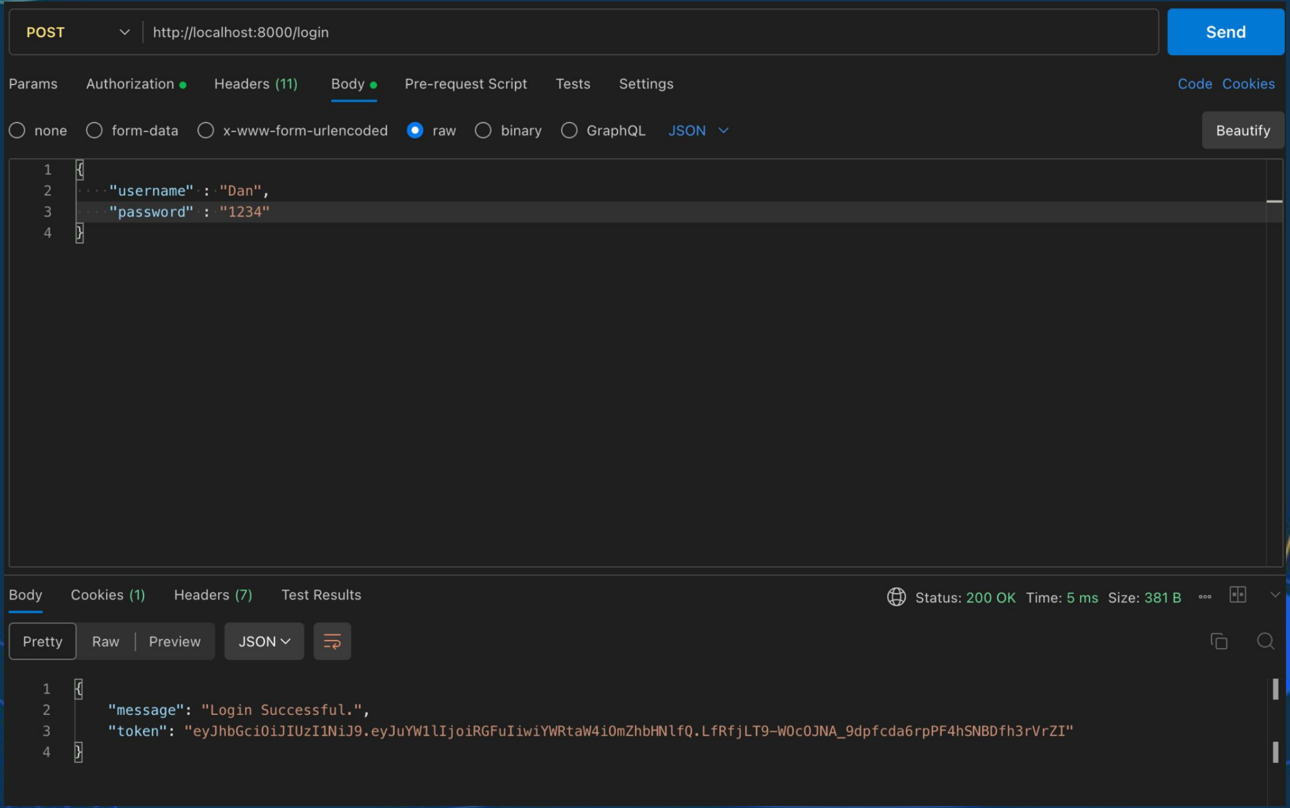


```
const express = require("express")
const jwt = require("jsonwebtoken")
const app = express()

app.use(express.json())

app.post('/login', (req, res)=>{
    const { username, password } = req.body

    if (username === "Dan" && password === "1234") {

        const payload = {
            "name" : username,
            "admin" : false
        }

        const token = jwt.sign(JSON.stringify(payload), 'secret', {algorithm: 'HS256'})

        res.send({
            message: "Login Successful.",
            token: token
        })
    } else {
        console.log("Invalid credentials")
        res.send({
            message: "Invalid credentials"
        })
    }
})

app.listen(8000, ()=>{
    console.log("Server is running on port http://localhost:8000")
})
```

CoGrammar

JSON Web Tokens (JWT)

Login Request With Postman



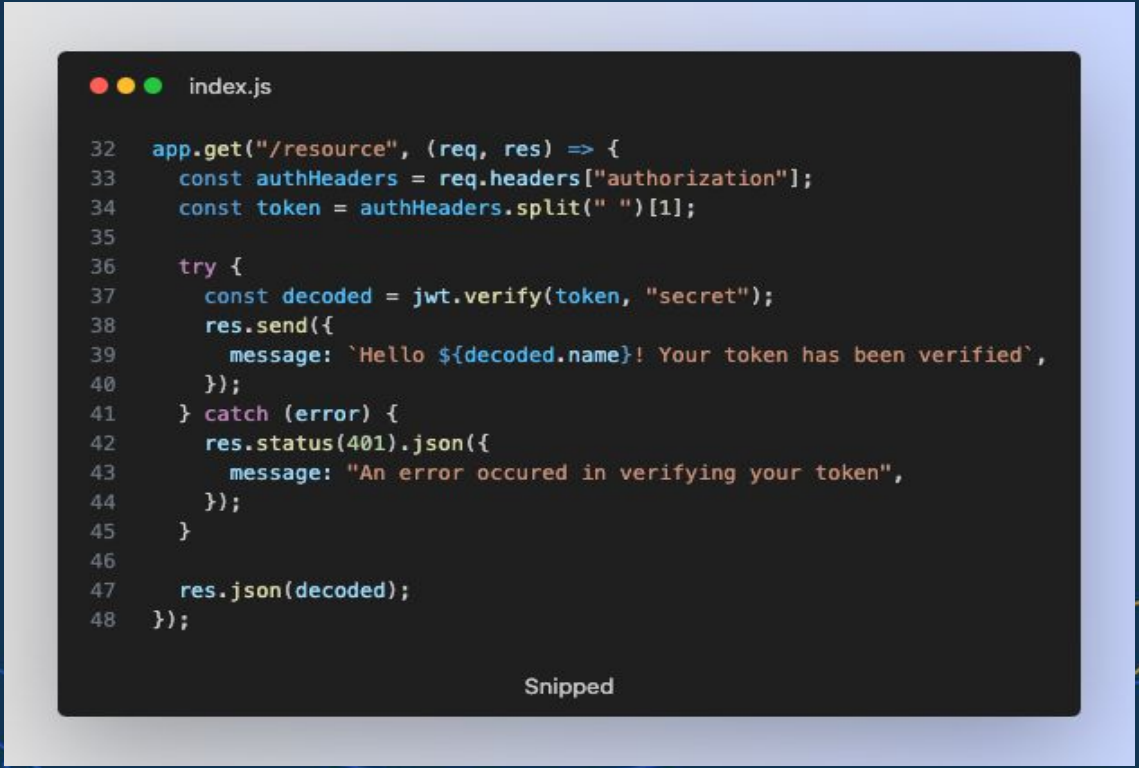
```
{
    "username" : "Dan",
    "password" : "1234"
}
```

```
{
    "message": "Login Successful.",
    "token": "eyJhbGciOiJIUzI1NiJ9.eyJuYW1lIjoiRGFuIiwiYWRtaW4iOmZhbHNlfQ.LfRfjLT9-WOcOJNA_9dpfcda6rpPF4hSNBDfh3rVrZI"
}
```

CoGrammar

JSON Web Tokens (JWT)

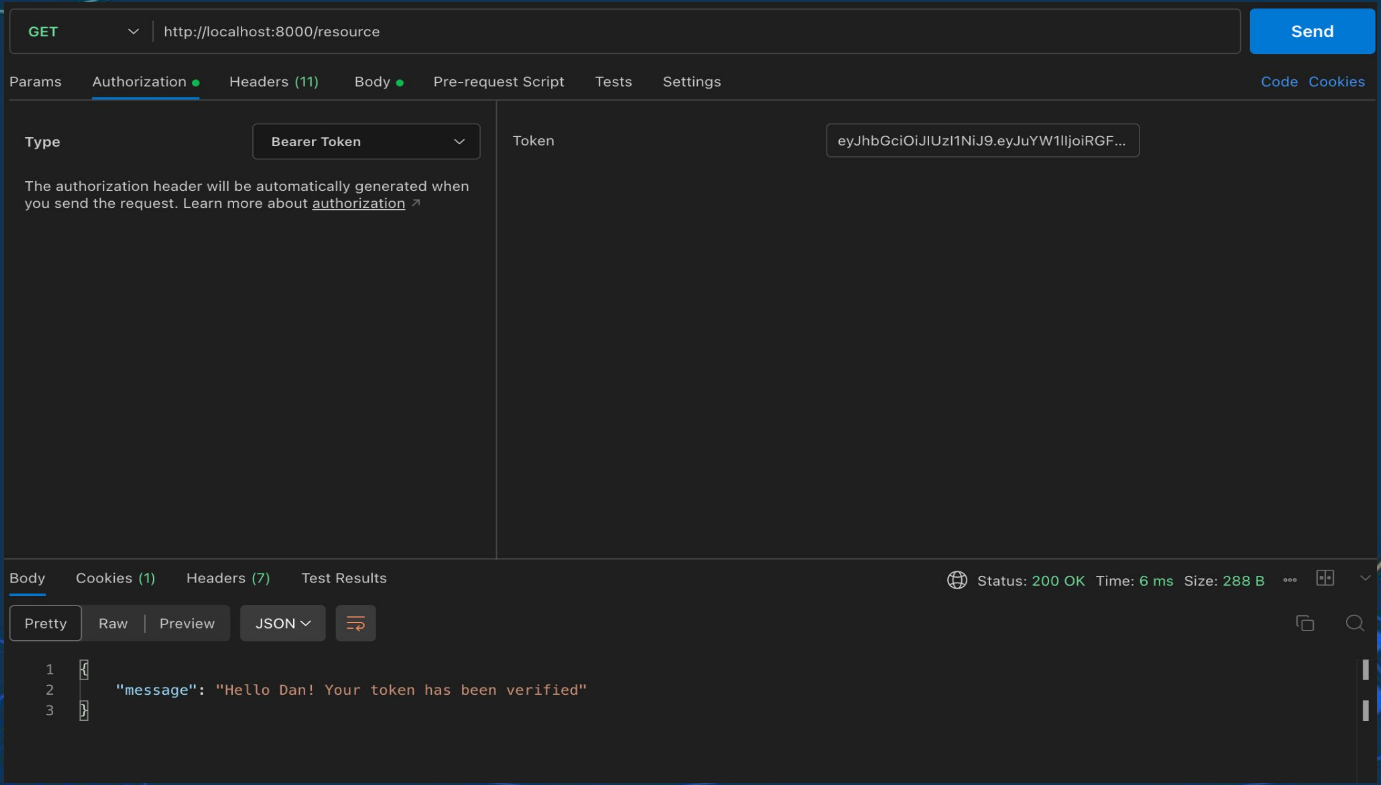
Verifying token



```
app.get("/resource", (req, res) => {
  const authHeaders = req.headers["authorization"];
  const token = authHeaders.split(" ")[1];

  try {
    const decoded = jwt.verify(token, "secret");
    res.send({
      message: `Hello ${decoded.name}! Your token has been verified`,
    });
  } catch (error) {
    res.status(401).json({
      message: "An error occured in verifying your token",
    });
  }

  res.json(decoded);
});
```

JSON Web Tokens (JWT)

Accessing and verifying request with POSTMAN



Type: Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about authorization ↗

Token: eyJhbGciOiJIUzI1NiJ9.eyJuYW1lIjoiRGF...

Status: 200 OK Time: 6 ms Size: 288 B

```
{
    "message": "Hello Dan! Your token has been verified"
}
```

CoGrammar

Let’s Breathe!

Let’s take a small break before moving on to the next topic.

CoGrammar

User Permissions

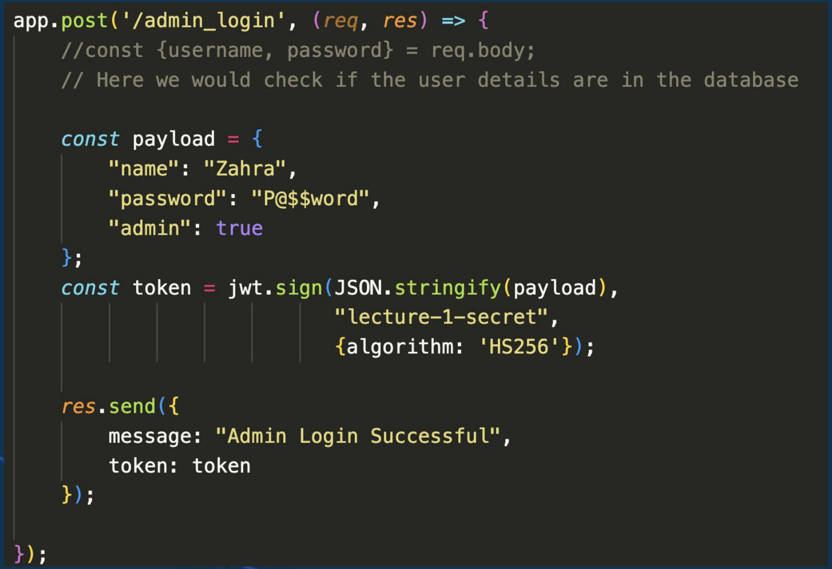
CoGrammar

User Permissions

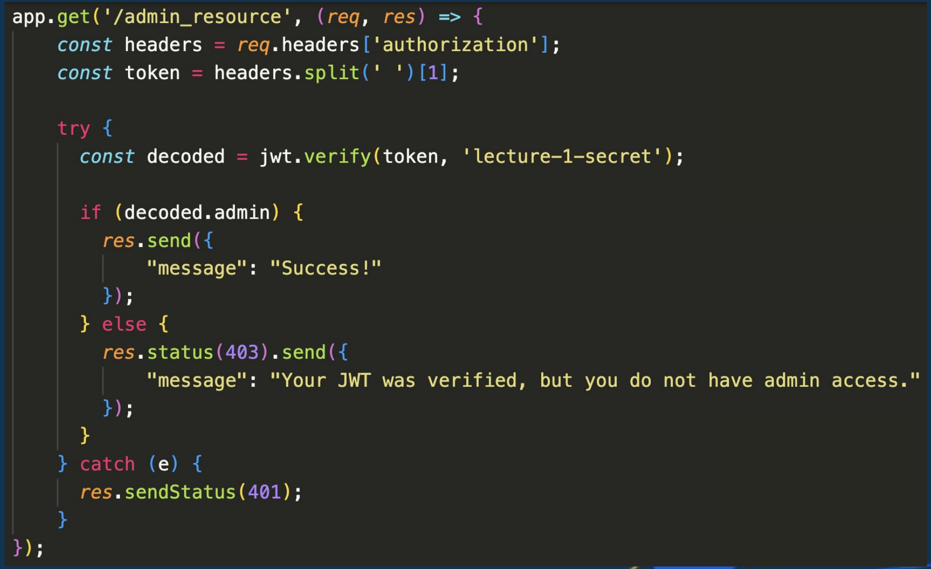
- ❖ By adding an **admin** attribute to the **payload of the auth endpoint**, we can implement user permissions i.e. features or resources only accessible to users with certain privileges.
- ❖ The admin attribute can **only** be added at the endpoint,

CoGrammar

User Permissions



```javascript
app.post('/admin_login', (req, res) => {
    //const {username, password} = req.body;
    // Here we would check if the user details are in the database

    const payload = {
        "name": "Zahra",
        "password": "P@$$word",
        "admin": true
    };
    const token = jwt.sign(JSON.stringify(payload),
                           "lecture-1-secret",
                           {algorithm: 'HS256'});

    res.send({
        message: "Admin Login Successful",
        token: token
    });

});
```



```javascript
app.get('/admin_resource', (req, res) => {
    const headers = req.headers['authorization'];
    const token = headers.split(' ')[1];

    try {
      const decoded = jwt.verify(token, 'lecture-1-secret');

      if (decoded.admin) {
        res.send({
            "message": "Success!"
        });
      } else {
        res.status(403).send({
            "message": "Your JWT was verified, but you do not have admin access."
        });
      }
    } catch (e) {
      res.sendStatus(401);
    }
});
```

Full Stack App with JWT

CoGrammar

Full Stack App

1. Set-up your Express.js server
2. Set-up your React.js server
3. Set-up your MongoDB connection
4. Configure your Mongoose model
5. Create your Mongoose queries
6. Create your HTTP routes in your Express.js server
7. Create your front-end which pulls information from your Express.js server

Questions and Answers

Thank you for attending

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

CoGrammar