




Welcome to the CoGrammar

Lecture: Authentication with JWT

The session will start shortly...

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



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

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

Lesson Objectives

- ❖ Define authentication and its importance in web development
- ❖ Discuss common authentication methods such as username/password, OAuth and JWT.
- ❖ Highlight the role of JWT in authentication and its benefits
- ❖ Implementation of JWT as an authentication method.

Introduction to Authentication



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.

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)



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.

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.

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.

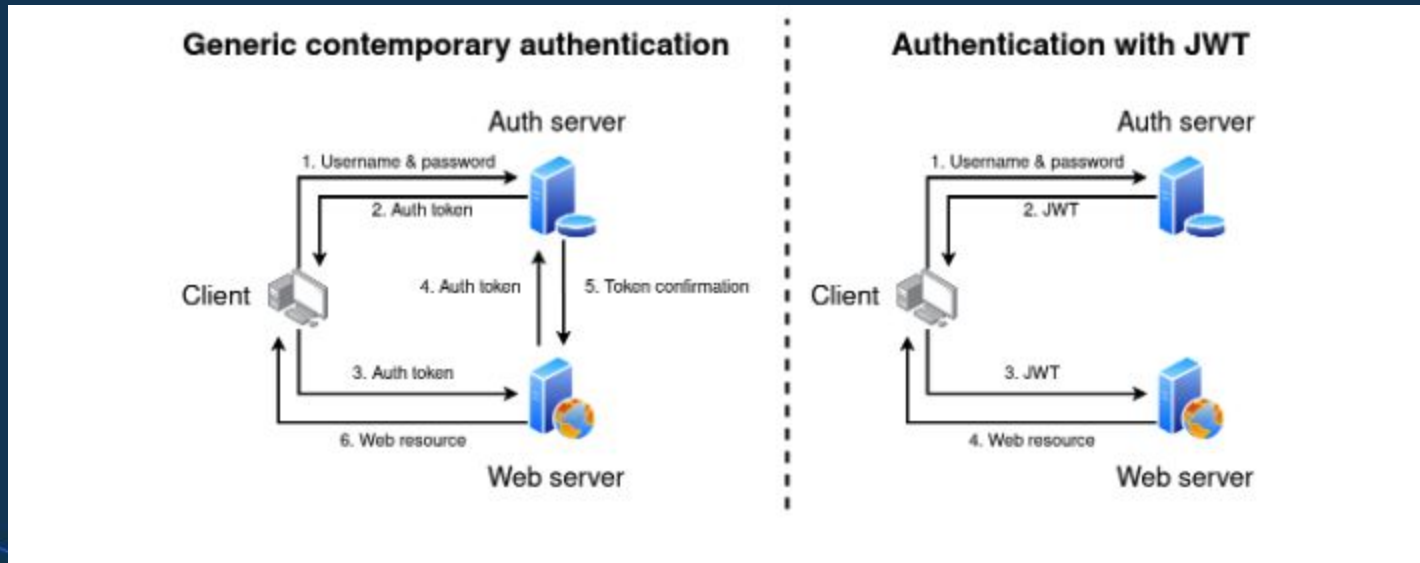


Structure of a JWT

- ❖ 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 = 'eyJpZCI6MTIzNCwibmFtZSI6IkpvaG4gRG9lIiwiaWVhY2VhY2V9'  
msg = header + '.' + payload  
sig = HS256('secret-key', msg).digestBase64()
```

How JWT performs over basic authentication mechanism



Source: [Radix](#)

Let's Breathe!

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



Implementing JWT



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

index.js

```
17     const token = jwt.sign(JSON.stringify(payload), 'secret', {algorithm: 'HS256'})  
18
```

Snipped

JSON Web Tokens (JWT)

Implementing JWT.

```
index.js

1  const express = require("express")
2  const jwt = require("jsonwebtoken")
3  const app = express()
4
5  app.use(express.json())
6
7  app.post('/login', (req, res) => {
8    const { username, password } = req.body
9
10     if (username === "Dan" && password === "1234") {
11
12       const payload = {
13         "name" : username,
14         "admin" : false
15       }
16
17       const token = jwt.sign(JSON.stringify(payload), 'secret', {algorithm: 'HS256'})
18
19       res.send({
20         message: "Login Successful.",
21         token: token
22       })
23     } else {
24       console.log("Invalid credentials")
25       res.send({
26         message: "Invalid credentials"
27       })
28     }
29   })
30
31  app.listen(8000, () => {
32    console.log("Server is running on port http://localhost:8000")
33  })
```

Snipped

Login Request With Postman



JSON Web Tokens (JWT)

Verifying token

index.js

```
32 app.get("/resource", (req, res) => {
33   const authHeaders = req.headers["authorization"];
34   const token = authHeaders.split(" ")[1];
35
36   try {
37     const decoded = jwt.verify(token, "secret");
38     res.send({
39       message: `Hello ${decoded.name}! Your token has been verified`,
40     });
41   } catch (error) {
42     res.status(401).json({
43       message: "An error occurred in verifying your token",
44     });
45   }
46
47   res.json(decoded);
48 });
```

Snipped

JSON Web Tokens (JWT)

Accessing and verifying request with POSTMAN

The screenshot shows the Postman interface for a GET request to `http://localhost:8000/resource`. The **Authorization** tab is selected, showing a **Bearer Token** type. The token value is `eyJhbGciOiJIUzI1NiJ9.eyJ1YXV1IjoIRGF...`. The **Body** tab is also visible, showing a JSON response: `{"message": "Hello Dan! Your token has been verified"}`. The status bar at the bottom indicates a **Status: 200 OK**, **Time: 6 ms**, and **Size: 288 B**.

GET `http://localhost:8000/resource` **Send**

Params **Authorization** Headers (11) Body Pre-request Script Tests Settings **Code** Cookies

Type **Bearer Token** Token `eyJhbGciOiJIUzI1NiJ9.eyJ1YXV1IjoIRGF...`

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Body Cookies (1) Headers (7) Test Results **Status: 200 OK** Time: 6 ms Size: 288 B

Pretty Raw Preview JSON `1 { 2 "message": "Hello Dan! Your token has been verified" 3 }`

Questions and Answers



Thank you for attending



Department
for Education

CoGrammar

