

1) JWT & use of backend Application.

⇒ JSON WEB TOKEN IS A COMPACT,

⇒ SECURE WAY TO TRANSMIT INFORMATION BETWEEN CLIENT & SERVER AS JSON OBJECT.

Used for

Authentication (Login system)

Authorization (Access control)

Secure data exchange between client & server

⇒ Commonly used for Node.js app with Express.

2) Three parts of Jwt Token

⇒ JWT has three parts separated by dots

Header, payload, signature

⇒ Header contains algorithm & token type

⇒ Payload contains user data claims

⇒ Signature used to verify the token

3) Difference

Header → sign

Visible after decoding

Cannot be tampered without secret

Not encrypted by default

Encrypted using secret key

payload store data

Signature protects data

4) Jwt.Sign:

⇒ jwt.sign() used for create json web tokens

jwt.sign({ id: user.id, process.env.JWT\_SECRET, { expire: '1h' } })

5) Purpose of jwt.verify:

- => used to validate the token
- => check if token is valid or expired
- => Confirms signature matches secret

6) Jwt secret key stored in .env file

- => Security -> security should not be public
  - => Avoid committing secret to github
  - => Easy change in production
- Jwt - Secret = secret key

7) Jwt secret key used for verification:

- => jwt.verify will fail
- => Throw "invalid signature" error
- => user cannot access protected route

8) 'ExpiresIn':

=> set token expiration time

{ expiresIn: '1h' }

- => Token become invalid after 1 hour
- => user must login again

9) Jwt token from backend to client

Send in JSON response

```
res.json({token});
```

Set in http only cookie:

```
res.cookie("token", token, {httpOnly: true});
```

10) Client send Jwt token :

⇒ Authorization header

Authorization: Bearer <token>

Example:

headers: {

    Authorization: "Bearer \$token"

11) Returned if token is missing :

⇒ 401 unauthorized

res.status(401).json({ message: "Token missing" });

12) Token invalid or expired :

403 forbidden

401 Unauthorized

13) Authentication

Who are you?

Login process

Verify tokens

Authorization

what can you access

Role based access

checks user role

14) Stateless authentication method:

⇒ Server does not store session

⇒ token contains all user info

⇒ Each request carries the token

Server doesn't need database session lookups.

15)

Protecting a route means allowing access only to authenticated users to provide a valid JWT token.

In express, this is done using middleware middlewares checks:

- ⇒ token exist or valid
- ⇒ token is not expired.

Const protect = [req, res, next] => {

const authHeader = req.headers.authorization;

const authHeader = req.headers['Authorization']

```
if (!authHeader) { res.status(401).json({ msg: "Token missing" }); }
```

```
const tokens = authHeader.split(' ') [,];
```

```
try {
```

```
const decode = jwt.verify(tokens[0], process.env.JWTSECRET);
```

req.user = decoded;

next();

```
} catch (error) {
```

```
res.status(403).json({ msg: "Invalid or expired" });
```

```
do {
```

try {

```
app.get('/dashboard', protect, (req, res) => {
```

```
res.json({ msg: "Protect route access" });
```

});