

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

Payload

Signature

Visible after decoding

Cannot be tampered without secret.

Not encrypted by default

Created 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.JWTSECRET, {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) ExpireIn:

=> set token expiration time

```
{ expireIn: '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 as 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

Authorization

Who are you?

What can you access

login process

Role based access

Verify tokens

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 lookup.

15)

Protecting a route means allowing access only to authenticated users to ~~provide~~ 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.header('authorization');
```

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

```
  }  
  const token = authHeader.split(' ')[1];
```

```
  try {  
    const decode = jwt.verify(token, process.env.JWT_SECRET);  
    req.user = decode;  
    next();
```

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

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

```
  }  
}
```

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

```
  res.json({ msg: 'protect route access' });
```

```
});
```