Fullstack javascript nanodegree

Session 6

Agenda

- DB joins
- Hashing Passwords
- JWT Authentication
- Live Activity
- Q&A

DB joins

DB joins

A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

(INNER) JOIN: Returns records that have matching values in both tables.

LEFT (OUTER) JOIN: Returns all records from the left table, and the matched records from the right table.

RIGHT (OUTER) JOIN: Returns all records from the right table, and the matched records from the left table.

FULL (OUTER) JOIN: Returns all records when there is a match in either left or right table.

DB joins

```sql
SELECT users.id, users.name, users.profile_picture, posts.content, posts.created_at
FROM users JOIN posts
ON users.id=posts.auther_id;
```

```sql
SELECT users.id, users.name, users.profile_picture, posts.content, posts.created_at
FROM users, posts
WHERE users.id=posts.auther_id;
```

```sql
SELECT u.id, u.name, u.profile_picture, p.content, p.created_at
FROM users as u, posts as p
WHERE u.id=p.auther_id;
```

DB joins

```
SELECT users.*, posts.*
FROM users, posts
WHERE users.id=posts.auther_id;
```

```
SELECT users.*, posts.*
FROM users, posts
WHERE users.id=posts.auther_id
AND posts.id=1001;
```

Hashing Passwords

Hashing passwords

- "Hashing" a password refers to taking a plain text password and putting it through a hash algorithm.

- It is essential to store passwords in a way that prevents them from being obtained by an attacker even if the application or database is compromised.

- After an attacker has acquired stored password hashes, they are always able to brute force hashes offline. As a defender, it is only possible to slow down offline attacks by selecting hash algorithms that are as resource intensive as possible.

- Hashing and encryption both provide ways to keep sensitive data safe. However, in almost all circumstances, passwords should be hashed, **NOT encrypted.**

Hashing passwords

- A salt is a random string. By hashing a plain text password plus a salt, the hash algorithm's output is no longer predictable. The same password will no longer yield the same hash.
- The salt gets automatically included with the hash, so you do not need to store it in a database.

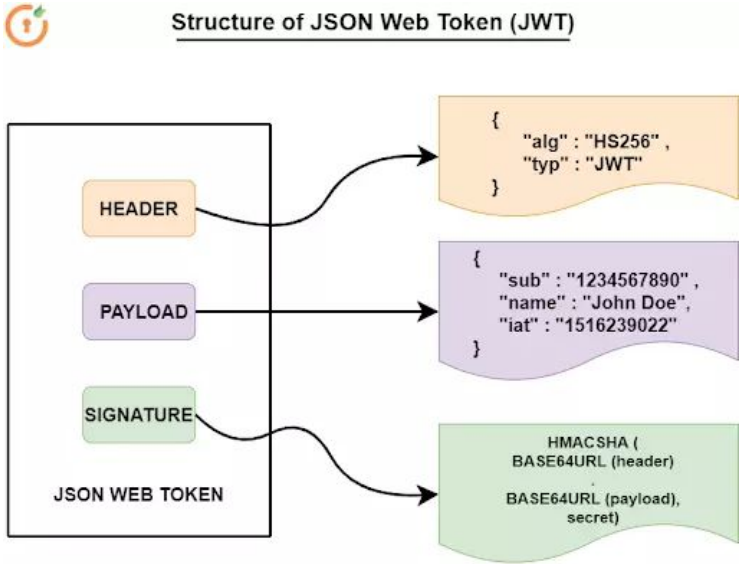
```
async toHash(password: string): Promise<string> {
  const salt = await bcrypt.genSalt(10);
  const hashedPassword = await bcrypt.hash(password, salt);

  return hashedPassword;
}

async compare(storedPassword: string, suppliedPassword: string): Promise<boolean> {
  return await bcrypt.compare(suppliedPassword, storedPassword);
}
```

JWT Authentication

JWT Authentication

- JWT, or JSON Web Token, is an open standard used to share information between two parties securely — a client and a server. In most cases.

- JWT structure is divided into three parts: header, payload, signature & is separated from each other by dot (.), and will follow the below structure:



JWT Authentication

```ts
export const currentUser = async (
  req: Request,
  res: Response,
  next: NextFunction,
) => {
  if (!req.headers.authorization) {
    return next();
  }

  const bearerToken = req.headers.authorization;  // "Bearer blablablablablabla"
  const token = bearerToken.split(' ')[1];
  const payload = jwt.verify(
    token,
    process.env.JWT_KEY!,
  ) as UserPayload;

  const user = await User.findOneById(payload.id);
  if(!user){
    return next();
  }

  req.user = user;

  next();
};
```

```ts
declare global {
  namespace Express {
    interface Request {
      user?: IUserSerialized;
    }
  }
}
```

Live Activity

Live Activity

Description:

- You will work individually.
- You should add two functionalities to the project:
    * Hashing passwords on signups
    * Require Authentication on opening users page(index & show)
- You should write testcases for it.

Setup:

- You can download the starter code from here.

Q&A

Your Feedback is Appreciated

References

- DB Joins
- Pros And Cons of ORMs
- Hashing Passwords
- What is JWT

Remember that we are here to help you

All you have to do is **ASK!**

Thanks for attending