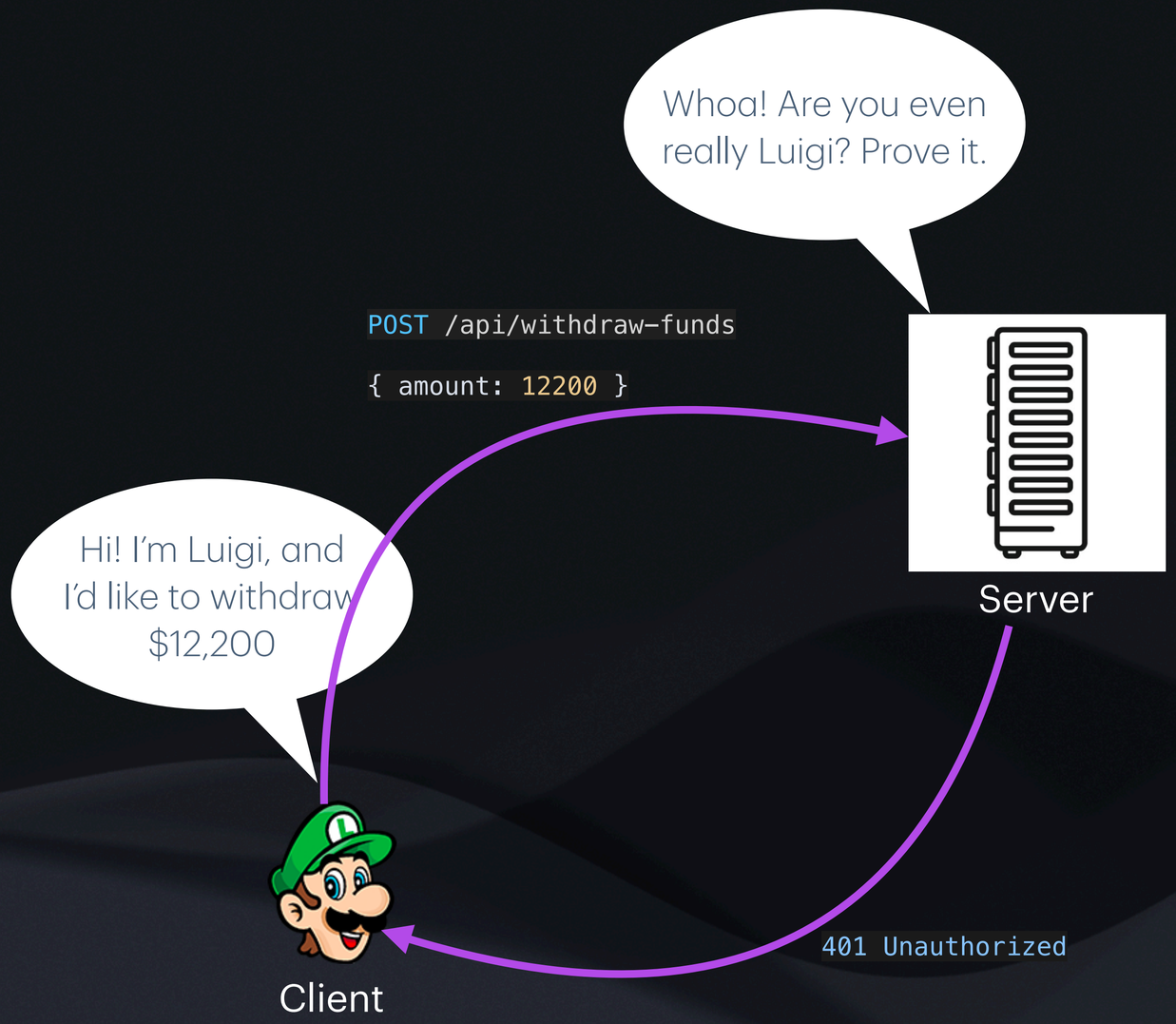
Authentication & Authorization

Cookies, JSON Web Tokens, Web Security

AAA

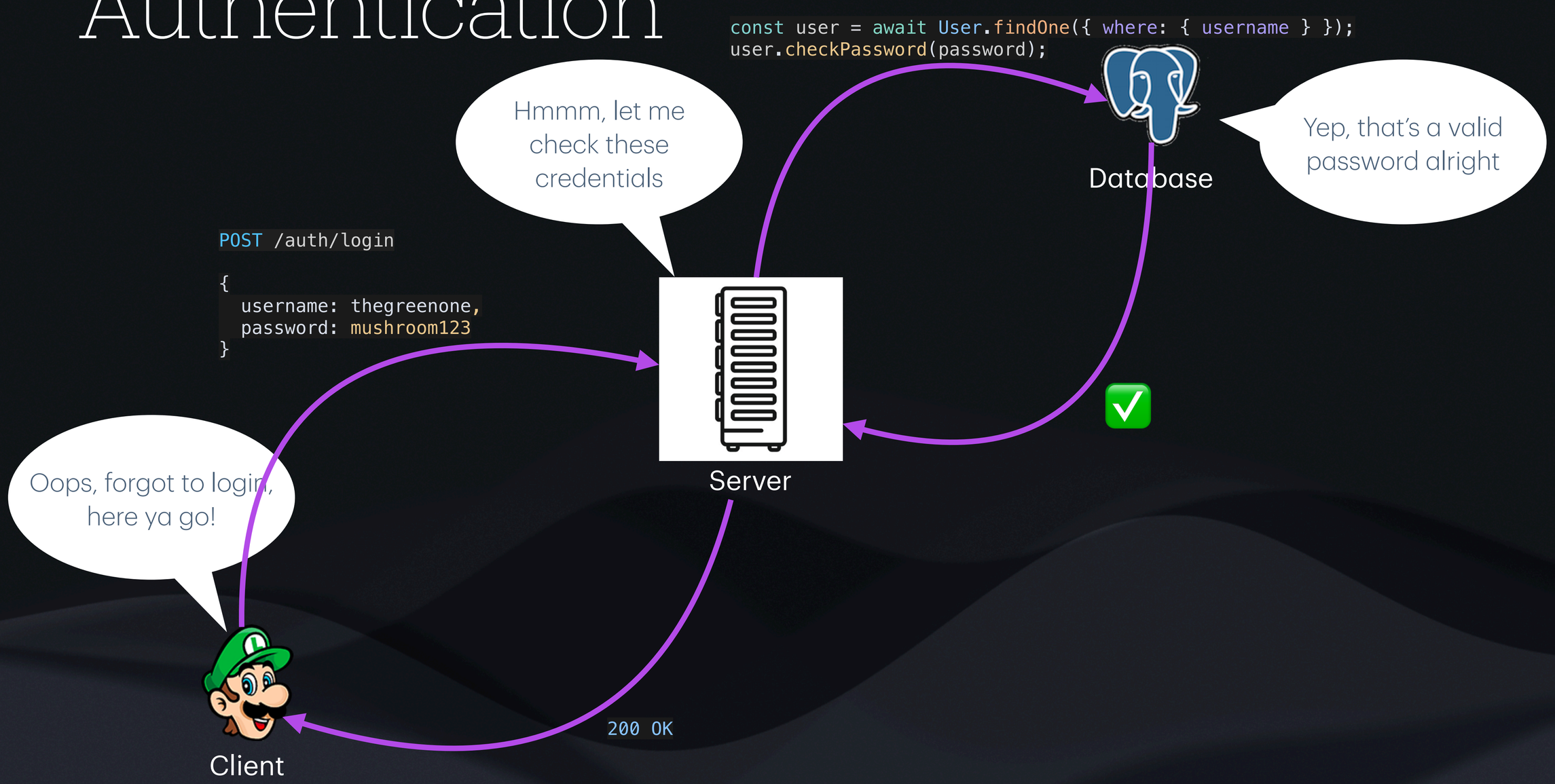
- Authentication:
  - Are you who you say you are?
- Authorization
  - Are you allowed to do that?
- Accounting
  - Are you using too many resources?

Authentication



Database

Authentication



Authentication

And then what...? 🤔

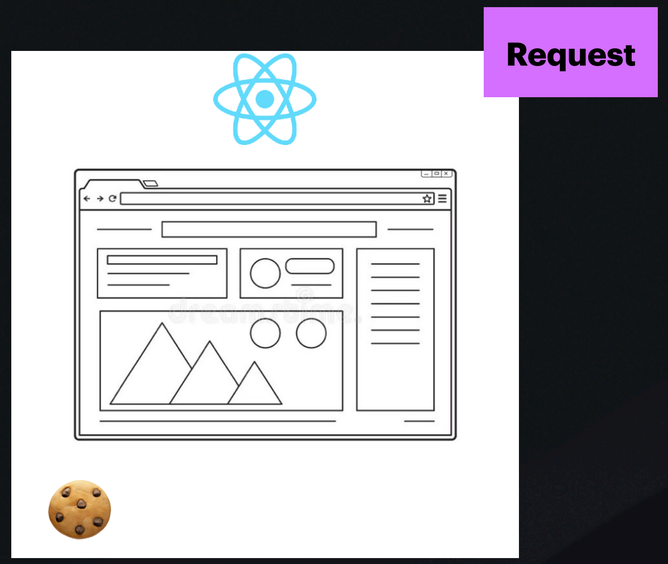
Will the client have to send their credentials on every request? 😭

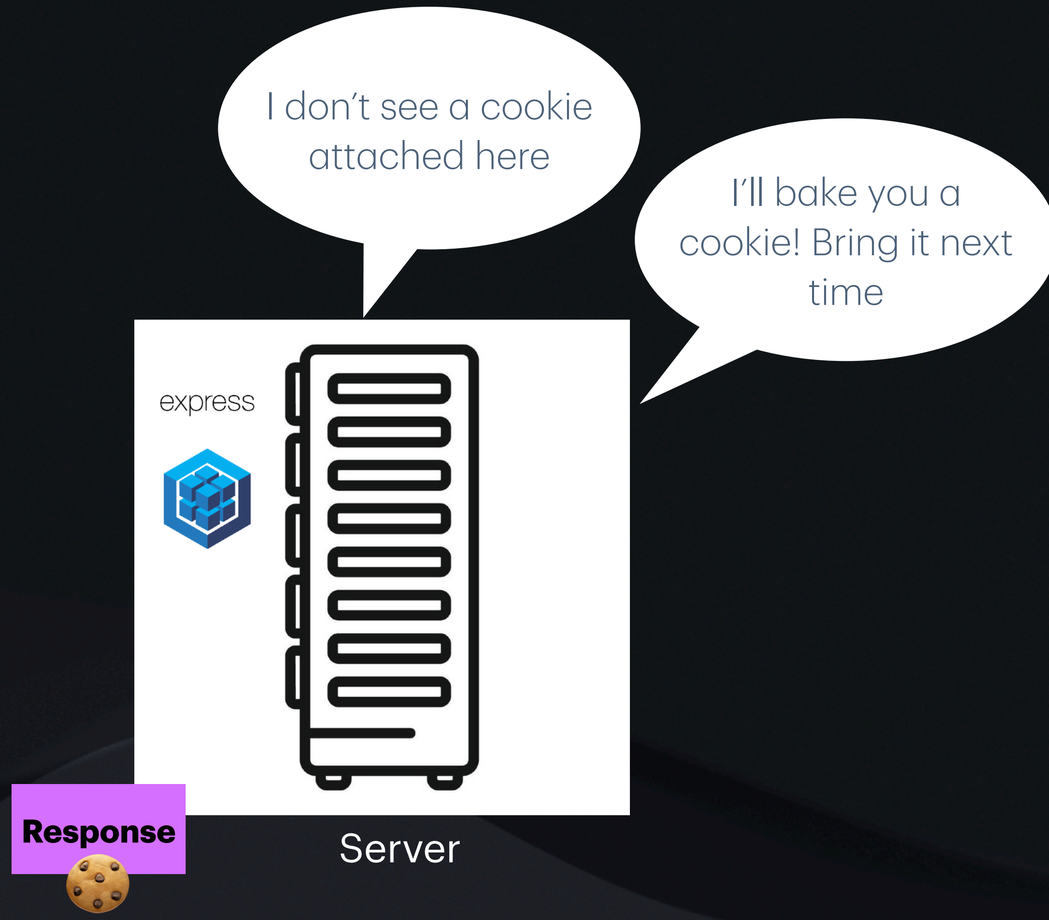
What does “login” even mean? 🤦

Cookies

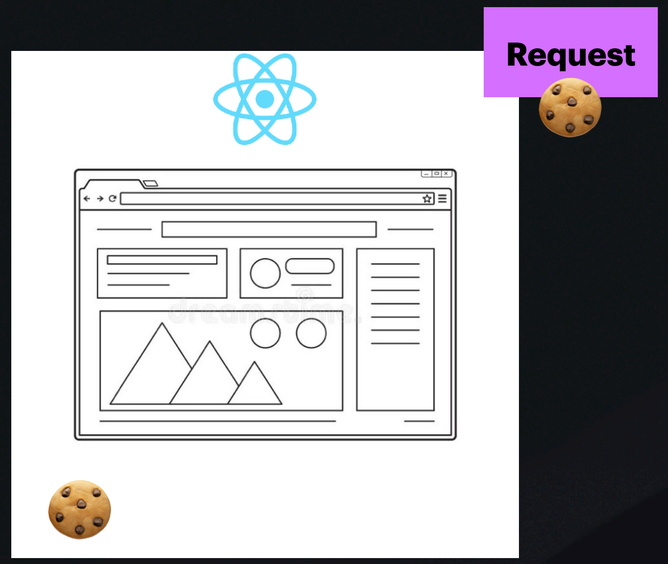
A “cookie” is just a small text file that your browser stores

Cookies


Request
Client

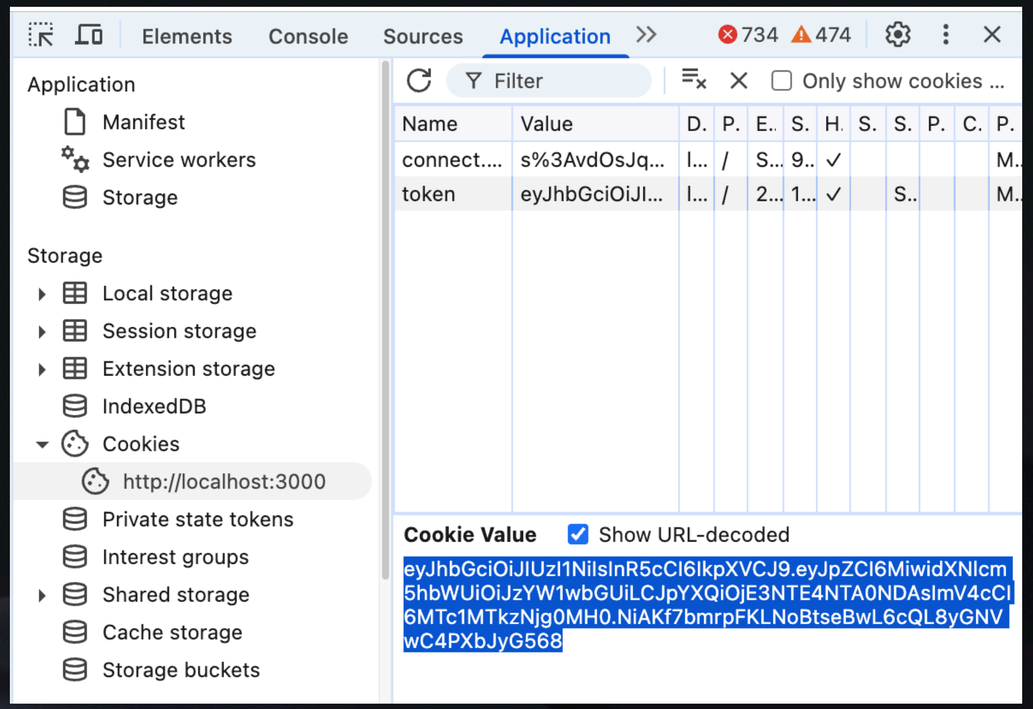

I don't see a cookie attached here
I'll bake you a cookie! Bring it next time
express
Response
Server

Cookies


Request
Client


Hey, I've seen this cookie before! Welcome back
express
Response
Server

Cookies



Cookies are created by the server,

and stored by the browser

You can view them from your browser's developer tools: Application > Cookies

They often contain sensitive information

Constrained by domain name

Let’s Look At Some Code!

Cookies

```
const cookieParser = require("cookie-parser");

// cookie parser middleware
app.use(cookieParser());
```

```
// somewhere in a route
res.cookie("token", token, {
  httpOnly: true,
  secure: true,
  sameSite: "strict",
  maxAge: 24 * 60 * 60 * 1000, // 24 hours
});
```

Express needs to be able to parse cookies that are sent in the request

`res.cookie` adds a cookie to the response

`httpOnly: true` === the cookie cannot be accessed by JS

`sameSite: “strict”` === the browser will not send the cookie to any other sites

Sessions vs JWTs

What Do We Add To The Cookie?

Option 1 – SessionID: An identifier that the server uses to keep track of the user's session in-memory

Option 2 – JSON Web Token: An object with necessary access credentials (e.g. userId, groupId, isAdmin)

JSON Web Tokens (JWT)

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9 // header
.eyJrZXkiOiJ2YWwiLCJpYXQiOjE0MjI2MDU0NDV9 // payload
.eUiabuiKv-8PYk2AkGY4Fb5KMZeorYBLw261JPQD5lM // signature
```

A JWT always has three parts xxxxx.yyyyy.zzzzz separated by periods

A JWT typically has an expiration time

Header: Basic information about the JWT

Payload: A list of claims about the user

Signature: Proof that the JWT wasn't tampered with

JSON Web Tokens (JWT)

$$f(\text{Header}, \text{Payload}, \text{Secret Key}) = \text{Signature}$$

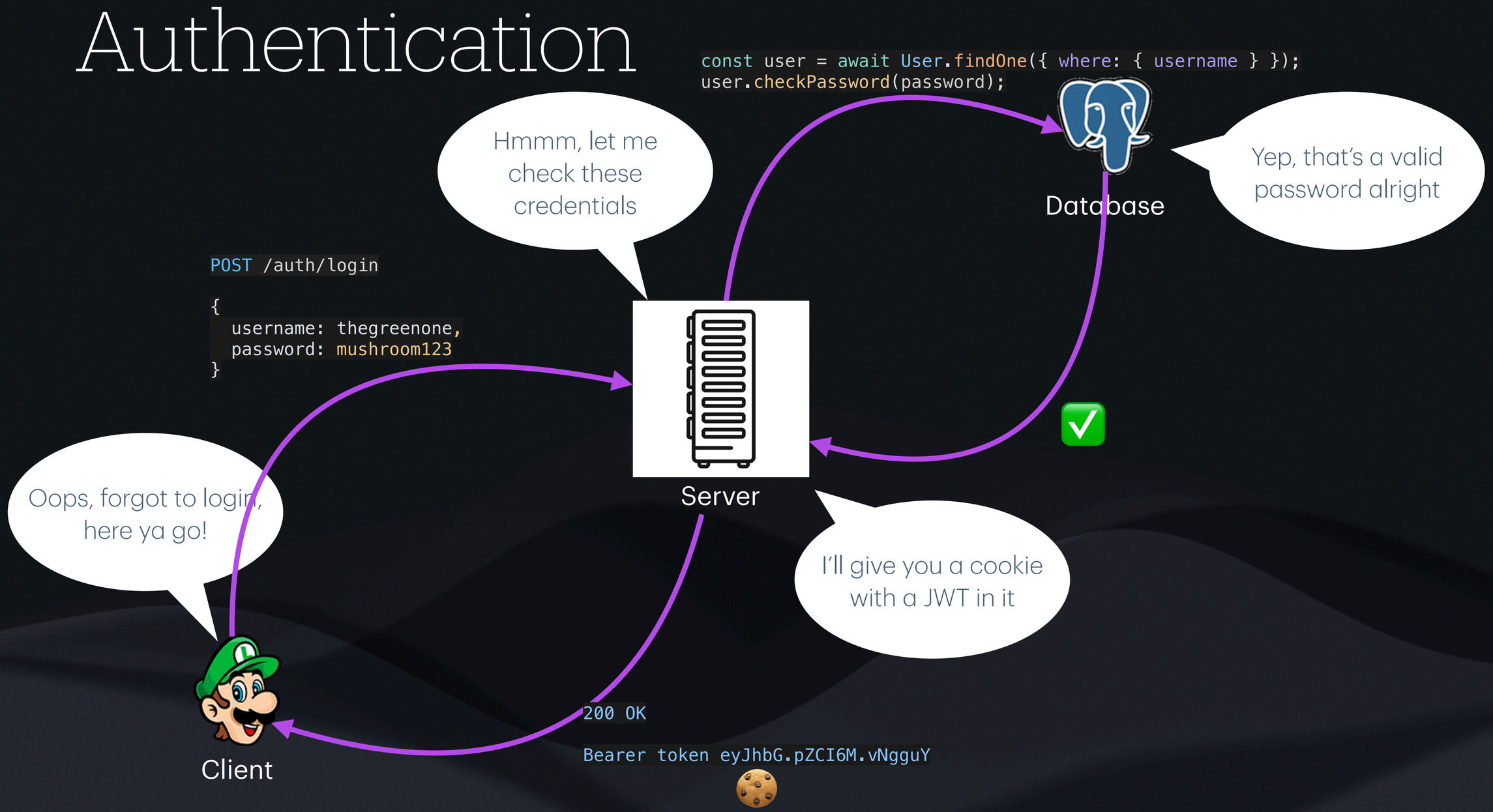
Signature

When the server creates the JWT, it “signs" it, creating the signature

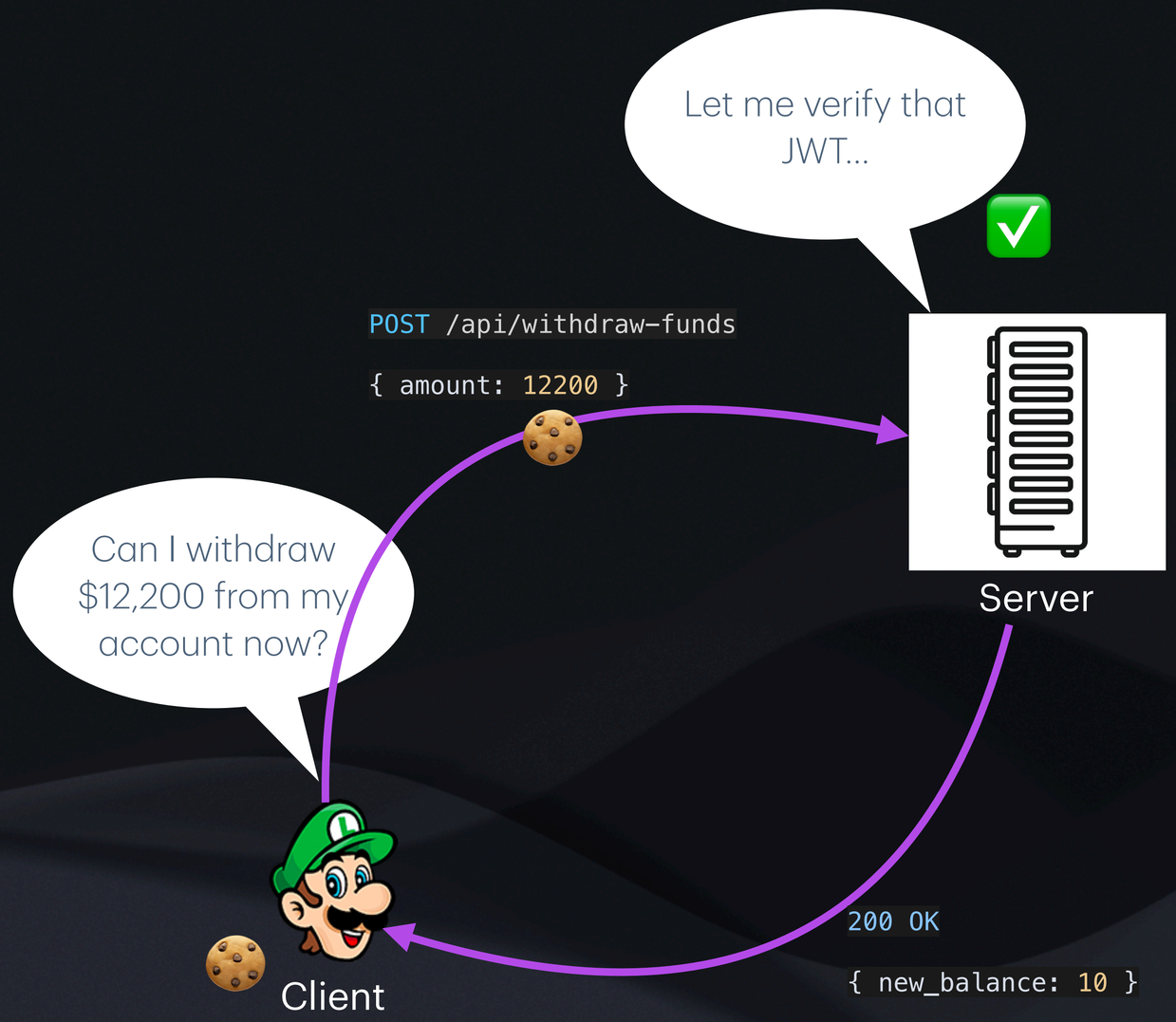
If the Header or Payload was altered, the Signature won’t match and the server will reject the JWT

How? (Let’s talk about hashing functions!)

Authentication



Authentication



Database

In general, the JWT should have all the information the server needs to *authorize* the request

No need to talk to the database again!

Authentication

Common Authentication Routes in Express:

```
app.get("/auth/me", (req, res) => {});
```

Am I logged in? If, what is my user account?

```
app.post(“/auth/signup", (req, res) => {});
```

Create a new user account – credentials should ALWAYS be in the request body

```
app.post(“/auth/login”, (req, res) => {});
```

Log me in to my existing user account, please

```
app.post(“/auth/logout”, (req, res) => {});
```

Log me out and clear the cookie

Authentication

```
app.use(morgan("dev"));
```

Express middleware modifies the request, response, and then passes them along to the next route handler

```
// POST new duck (protected route - requires authentication)
router.post("/", authenticateJWT, async (req, res) => {...}
```

Express middleware can also be applied to an individual route

Here, authenticateJWT checks the JWT and, if absent, simply responds with an error

Authentication

A Few Security Precautions:

1. Only send cookies over HTTPS (secure: true)

2. Don’t store passwords in plaintext – always hash

3. Never send sensitive information in the URL

4. JWTs should expire: shorter expiration = safer JWT

Authentication Practice