

NodeJS: Authentication

Ashok Dudhat

Germany Startup Jobs

www.germanystartupjobs.com

Agenda

- Protected Routes
- Authentication & Authorization
- HTTP Authentication
- HTTP Authorization

Protected Routes

- Routes in our Express-App look like this:

/

/api

/customers

...

Protected Routes

- Routes in our Express-App look like this:
/
/api
/customers
...
- What if we want to protect them from certain users? I.e. users, that are unknown to us.



Protected Routes

- The guy asks for username and password!

Protected Routes

- The guy asks for username and password!
- You say your username is “hallo” and your password is “world”.

Protected Routes

- The guy asks for username and password!
- You say your username is “hallo” and your password is “world”.
- The guy looks up “hallo” and “world” in his database and finds you.

Protected Routes

- The guy asks for username and password!
- You say your username is “hallo” and your password is “world”.
- The guy looks up “hallo” and “world” in his database and finds you.
- You are now **authenticated** as known user and you get your **ticket** which **authorizes** you to drive along the road.



Authentication & Authorization

Authentication: is the process of verifying that the user is **somebody** the system knows.

Authorization: is the process of verifying that the user has access to **something** the system owns.

Ticket: A proof that the users is authorized. Mostly it is a token.

HTTP Authentication

USER

HTTP
SERVER

HTTP Authentication

USER

HTTP
SERVER

POST /login

```
{  
  username: 'hallo',  
  password: 'world'  
}
```



HTTP Authentication Flow

USER

HTTP
SERVER

POST /login

```
{  
  username: 'hallo',  
  password: 'world'  
}
```



RESPONSE /login

```
{  
  token: 'fDjbn8fnVn'  
}
```



HTTP Authentication

USER

HTTP
SERVER

POST /login

```
{  
  username: 'hallo',  
  password: 'world'  
}
```



RESPONSE /login

```
{  
  token: 'fDjbn8fnVn' <- TICKET HERE  
}
```



HTTP Authorization

USER

HTTP
SERVER

GET /customers

HEADER: authorization Bearer fDjbn8fnVn



HTTP Authorization

USER

HTTP
SERVER

GET /customers

HEADER: authorization Bearer fDjbn8fnVn

<- TICKET HERE



HTTP Authorization

USER

HTTP
SERVER

GET /customers

HEADER: authorization Bearer fDjbn8fnVn



RESPONSE /customers

<HTML>

<body>...</body>

</HTML>



Authentication Types

There are two types of Authentication you can use in any web application development.

1. **Session-based Authentication**
2. **Token-based Authentication**

NodeJS Sessions & Cookies

1. Stateless VS Stateful
2. Cookies & Sessions
3. SessionAuthentication
4. Task

1. Introduction

- Stateless



1. Introduction

- Stateless



She does not know what you ordered last week.



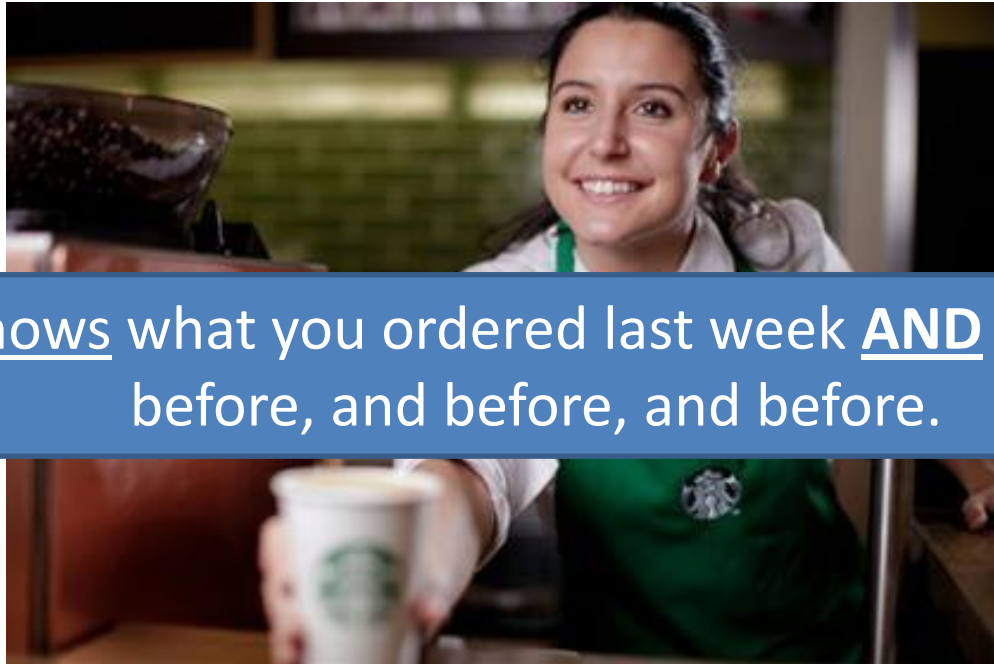
1. Introduction

- Stateful



1. Introduction

- Stateful



She knows what you ordered last week AND the week before, and before, and before.

1. Introduction

- **Request:** HTTP packet sent from client to server („One coffee please“)

1. Introduction

- **Request:** HTTP packet sent from client to server („One coffee please“)
- **Response:** HTTP packet sent back from server to client after it received a Request („There you go – here it is“)

1. Introduction

- **Request:** HTTP packet sent from client to server („One coffee please“)
- **Response:** HTTP packet sent back from server to client after it received a Request („There you go – here it is“)
- **Transaction:** One pair of Request and Response („One coffee please“ – „There you go – here it is“)

1. Introduction

- **Request:** HTTP packet sent from client to server („One coffee please“)
- **Response:** HTTP packet sent back from server to client after it received a Request („There you go – here it is“)
- **Transaction:** One pair of Requests and Responses („One coffee please“ – „There you go – here it is“)
- **Session:** A set of transactions
 - „One coffee please“ – „There you go – here it is“)
 - „One latte please“ – „Okay – here it is“)
 - „One espresso please“ – „Okay.“)

2. Cookies & Sessions

- **Cookies:** is an ID that is sent in the HTTP-Header and identifies a session (saved mostly client-side).
- **Session:** A set of transactions, saved server-side mostly in the memory, as a file or on the database.
- No JavaScript necessary. When a browser receives a reponse, it automatically saves the cookie in all future requests.
- Getting rid of cookies: delete them in browser configs.

3. Session Authentication (1/4)

USER

HTTP
SERVER

POST /login

```
{  
  username: 'hallo',  
  password: 'world'  
}
```



3. Session Authentication (2/4)

USER

HTTP
SERVER

POST /login

```
{  
  username: 'hallo',  
  password: 'world'  
}
```



RESPONSE /login

Header: Cookie-ID: 123

```
{  
  
}
```



username + password
exists. create a
session object and send
back cookie.

3. Session Authentication (3/4)

USER

HTTP
SERVER

GET /content

Header: Cookie-ID: 123

{

}



3. Session Authentication (4/4)

USER

HTTP
SERVER

GET /content

Header: Cookie-ID: 123

{

}



RESPONSE /content

Header: Cookie-ID: 123

{

<div>secret content</div>

}



4. Task



4. Task

Create a small Too-late-comer website with Webpack, JQuery, Bootstrap and MongoDB. The user can keep track of students who come too late.

- 1) It has three navigation points.
 - 1) Logout
 - 2) New Later-Comer (a form where a new too-late comer can be saved)
 - 3) History (shows a history of people coming too late)

By default, 3) is the landing page

4. Task

1) New Too Later Comer GUI

The image shows a graphical user interface (GUI) window titled "Too-Late-Comer". The window has a light gray background and a darker gray title bar. Inside the window, there are three input fields arranged vertically in the center. The first field is labeled "Username", the second is labeled "Password", and the third is a button labeled "Login". The "Login" button is shaded gray, while the other two fields are white with black borders.

Too-Late-Comer

Username

Password

Login

4. Task

1) New Too Later Comer GUI

[New Too Later Comer](#) | [History](#) | [Logout](#)

Max Mustermann

12.02.2018

9:35

Add

4. Task

1) History

New Too Later Comer <u>History</u> Logout				
Name	Date	Time	Minutes too late	Action
Max Mustermann	11.02.2018	09:30	15	Delete
Julia Müller	11.02.2018	09:45	30	Delete

4. Task

1) Login/Logout

Too-Late-Comer

You are logged out now.

Login