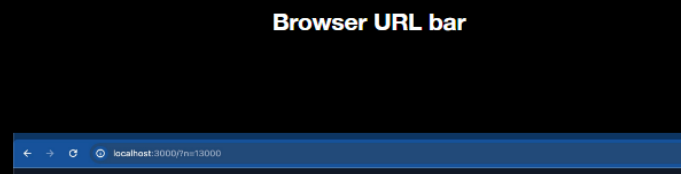
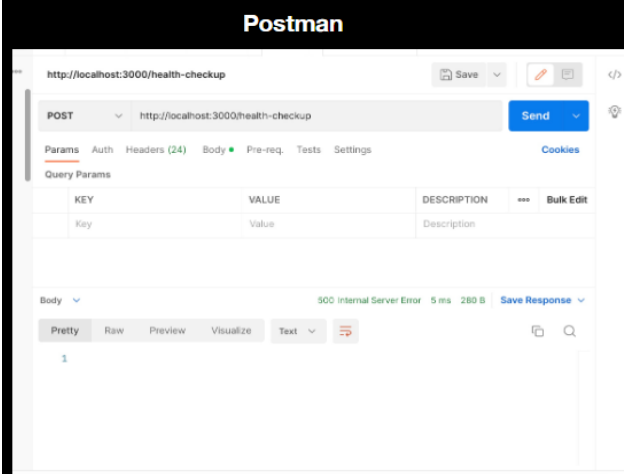


Dynamic Backend Authentication using Express, JWT and MongoDB

First, we will learn about fetch API.

Until now, we've sent requests in 2 ways



The 1st one is a POST request and the 2nd one is a GET request.

There's a third way

Lets say I ask you create an HTML page where

1. You can see the names of 10 people
2. You need to make sure you get these data from an API call



In the earlier two ways we were posting or getting the data on the server side only, but now we have to take those data from the server into the html/frontend side.

Like we are using someone else backend to get the data so for that we use (fetch) method.

Let me show you :-

```
<!DOCTYPE html>
<html>

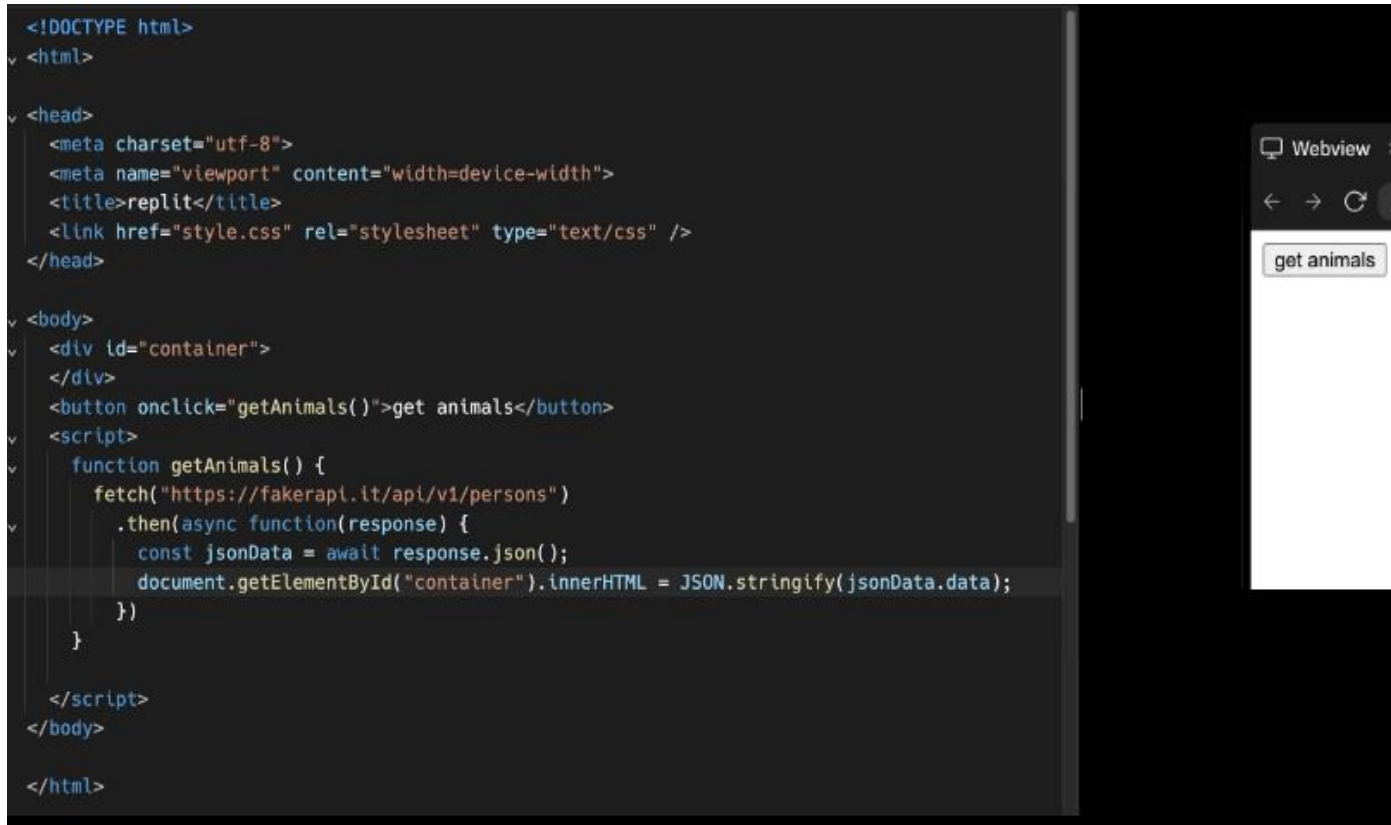
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width">
  <title>replit</title>
  <link href="style.css" rel="stylesheet" type="text/css" />
</head>

<body>
  <button onclick="getAnimalData()">Get animal data</button>
</body>

</html>
```

This function is written in an index.js file.

Now, I will show how fetch works here:-



Ye fetch ke andar jo karnama kia h , wo krne ki zaroorat nhi h , normal tarike se bhi ho jata h.

So basically we use fetch to take data from another person's server.

Now we will learn about Authentication in detail.

AUTHENTICATION:-

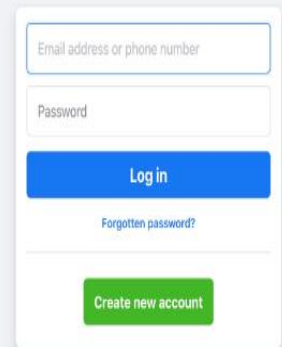
Almost all websites have auth

**There are complicated ways
(Login with google...) to do auth**

**Easiest is a username password
based auth**

facebook

Facebook helps you connect and share
with the people in your life.

A screenshot of the Facebook login interface. It features a white login box on a light blue background. The box contains two input fields: 'Email address or phone number' and 'Password'. Below these fields is a blue 'Log in' button. Under the 'Log in' button is a link that says 'Forgotten password?'. At the bottom of the box is a green 'Create new account' button.

[Create a Page](#) for a celebrity, brand or business.

**Before we get into authentication
Lets understand some cryptography jargon**

- 1. Hashing**
- 2. Encryption**
- 3. Json web tokens**
- 4. Local storage**

1.) Hashing:-

Hashing in authentication refers to the process of converting a plaintext password into a fixed-length string of characters, called a hash, using a cryptographic hash function. This hashed value is then stored in a database or system instead of the original password. When a user attempts to log in, the password they provide is hashed using the same algorithm, and the resulting hash is compared to the stored hash.

Hashing is a fundamental technique used in authentication for several reasons:

1. **Security:** Hashing helps protect user passwords by storing them in a non-reversible format. Even if an attacker gains access to the database storing hashed passwords, they cannot easily reverse the process to obtain the original passwords. This enhances the security of user accounts and mitigates the impact of data breaches.
2. **Data Integrity:** Hashing ensures data integrity by generating a unique hash value for each input. Even a small change in the input data will result in a significantly different hash value, making it easy to detect any tampering or unauthorized modifications.
3. **Salted Hashes:** To further enhance security, authentication systems often use salted hashes. A salt is a random value added to the password before hashing, resulting in a unique hash even for identical passwords. Salting prevents attackers from using precomputed hash tables (rainbow tables) to crack hashed passwords efficiently.
4. **Password Verification:** During authentication, the user-provided password is hashed using the same algorithm and compared to the stored hash. If the hashes match, it indicates that the user-provided password is correct, and access is granted. If they do not match, access is denied.

1. Hashing
2. Encryption
3. Json web tokens
4. Local storage

1. Hashing is one way
2. Given the output, no one can find out the input
3. Changing the input a lil bit changes the output by a lot

harkirat@gmail.com
1234561



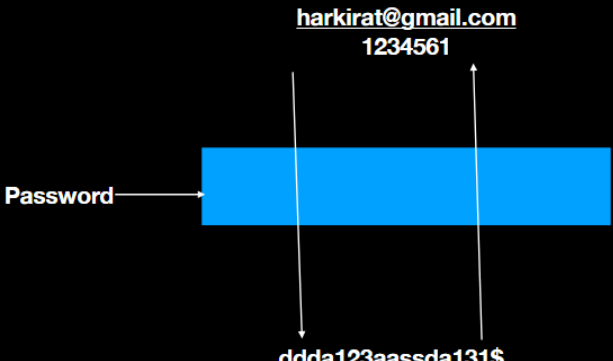
ddda123aassda131\$

It is a one-way technique, we can convert the input to hash, but we can't convert hash to input.

2.) Encryption:-

1. Hashing
2. **Encryption**
3. Json web tokens
4. Local storage

1. Encryption is two way
2. A string is encrypted using a password
3. String can be decrypted using the same password

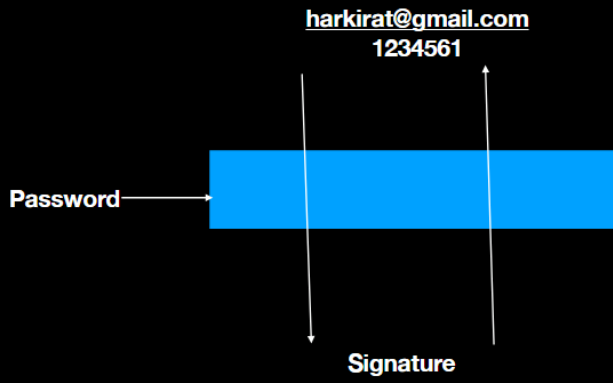


The diagram illustrates the encryption process. A blue rectangular box represents the encryption function. An arrow labeled "Password" points into the box from the left. Two arrows emerge from the box: one points down to the text "ddda123aassda131\$" and the other points up to the text "harkirat@gmail.com 1234561".

3.) JSON Web Tokens:-

1. Hashing
2. Encryption
3. **Json web tokens**
4. Local storage

1. Its neither of encryption or hashing (its technically a digital signature)
2. Anyone can see the original output given the signature
3. Signature can be verified only using the password



The diagram illustrates the JSON Web Tokens process. A blue rectangular box represents the token generation function. An arrow labeled "Password" points into the box from the left. Two arrows emerge from the box: one points down to the text "Signature" and the other points up to the text "harkirat@gmail.com 1234561".

It only works for JSON inputs.

It takes the input and generates the token.

4.) Local Storage:-

1. Hashing
2. Encryption
3. Json web tokens
4. Local storage

A place in your browser where you can store some data

Usually things that are stored include -

1. Authentication tokens
2. User language preference
3. User theme preference

