Node REPL

Read
Evaluate
Print
Loop

browser JS

.help gives us commands

Process

process : This object provides information about, and control over, the current Node.js process.

process.argv : returns an array containing the command-line arguments passed when the Node.js process was launched.

```
shradhakhapra@Shradhas-MacBook-Air Backend % node script.js
[
  '/usr/local/bin/node',
  '/Users/shradhakhapra/WebClassroom/Backend/script.js'
]
shradhakhapra@Shradhas-MacBook-Air Backend % node script.js hello bye
[
  '/usr/local/bin/node',
  '/Users/shradhakhapra/WebClassroom/Backend/script.js',
  'hello',
  'bye'
]
shradhakhapra@Shradhas-MacBook-Air Backend %
```


Backend

script.js

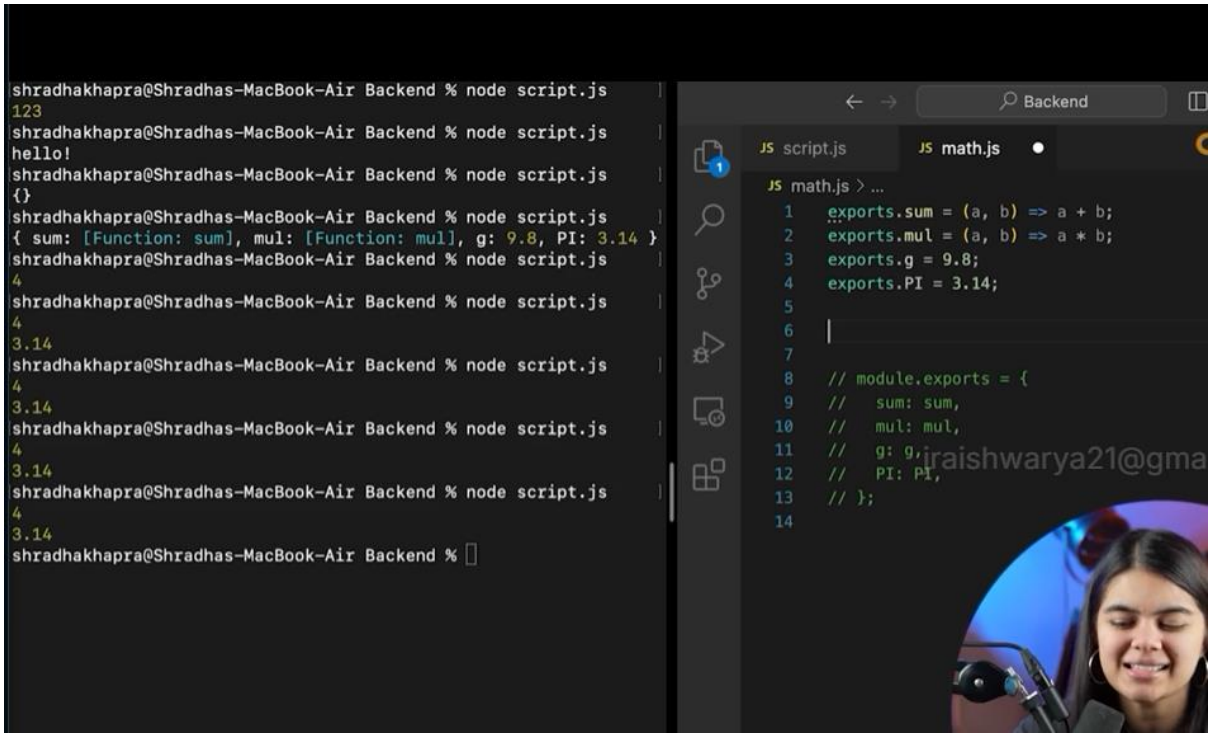
```
1   // let n = 5;
2
3   // for (let i = 0; i < n; i++) {
4   //   console.log("hello, ", i);
5   // }
6
7   // console.log("bye!");
8
9   console.log(process.argv);
10
```

module.exports

requiring files

require( ) a built-in function to include external modules that exist in separate files.

module.exports a special object



module.exports

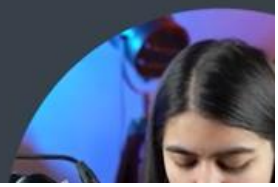
requiring directories

main()

1) require → index.js (entry point)

require( ) a built-in function to include external modules that exist in separate files.

module.exports a special object



NPM (Node Package Manager)

code

npm is the standard package manager for Node.js.

① library of packages
② command line tool

Installing Packages

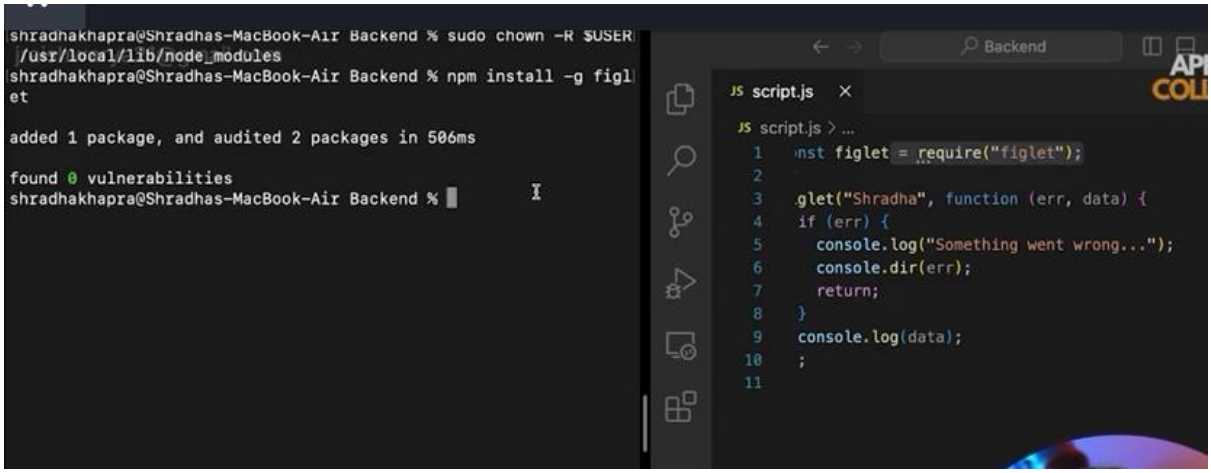
node_modules The node_modules folder contains every installed dependency for your project.

package-lock.json It records the exact version of every installed dependency, including its sub-dependencies and their versions.

package.json

The package. json file contains descriptive and functional metadata about a project, such as a name, version, and dependencies

npm init

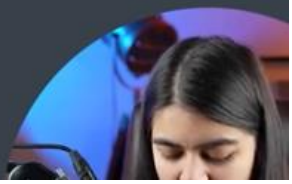


require v/s import

import { sum } from "./math.js"

We can't selectively load only the pieces we need with require but with import, we can selectively load only the pieces we need, which can save memory.

Loading is synchronous for 'require' but can be asynchronous for 'import'.



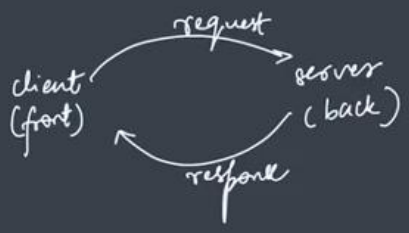
hwarya21@gmail.com

Express

A Node.js web application framework that helps us to make web applications

It is used for server side programming.

1. listen for incoming request
2. parse
3. match response with routes
4. response



Getting Started with Express

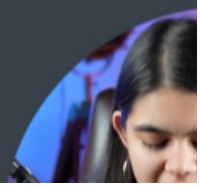
```
const express = require("express");
const app = express();

let port = 8080;

app.listen(port, () => {
  console.log(`app listening on port ${port}`);
});
```

*Ports are the logical endpoints of a network connection that is used to exchange information between a web server and a web client.

Routing

it is process of selecting a path for traffic in a network or between or across multiple networks.

```
app.get("/apple", (req, res) => {
  res.send({
    name: "apple",
    color: "red",
  });
});
```



Nodemon

To automatically restart server with code changes

Path Parameters

req.params

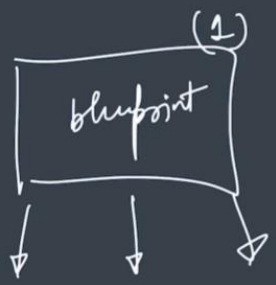
```
app.get("/ig/:username", (req, res) => {
  let { username } = req.params;
  res.send(`This account belongs to @${username}`);
});
```

Templating

model

EJS (Embedded JavaScript templates)

EJS is a simple templating language that lets you generate HTML markup with plain JavaScript.



Views Directory

```
const path = require("path");

app.set("views", path.join(__dirname, "/views"));
```


Interpolation Syntax

Interpolation refers to **embedding expressions** into marked up text.

Tags

- `<%` 'Scriptlet' tag, for control-flow, no output
- `<%_` 'Whitespace Slurping' Scriptlet tag, strips all whitespace before it
- `<%=` Outputs the value into the template (HTML escaped)
- `<%-` Outputs the unescaped value into the template
- `<%#` Comment tag, no execution, no output
- `<%%` Outputs a literal '<%'
- `%>` Plain ending tag
- `-%>` Trim-mode ('newline slurp') tag, trims following newline
- `_%>` 'Whitespace Slurping' ending tag, removes all whitespace after it

Conditional Statements in EJS

Adding conditions inside EJS

```
<% if(diceVal == 6) { %>
<h2>Nice! Roll dice again.</h2>
<% } %>
```

Serving Static Files

app.use(express.static(folder_name))

```
app.use(express.static(path.join(__dirname, "public")));
```

Includes

subtemplates

```
<%- include("includes/head.ejs"); %>
```

Get & Post Requests

GET

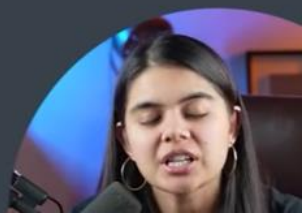
> Used to GET some response

> Data sent in query strings
(limited, string data & visible in URL)

POST

> Used to POST something (for Create/ Write/ Update)

> Data sent via request body (any type of data)



Handling Post requests

- Set up POST request route to get some response
- Parse POST request data

```
app.use(express.urlencoded({ extended: true }));
app.use(express.json());
```

Object Oriented Programming

To structure our code

- prototypes
- New Operator
- constructors
- classes
- keywords (extends, super)

Object Prototypes

Prototypes are the mechanism by which JavaScript objects inherit features from one another.

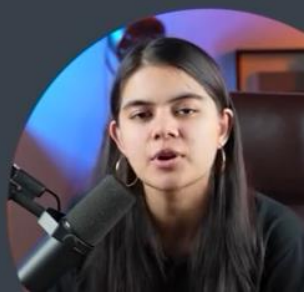
It is like a single template object that all objects inherit methods and properties from without having their own copy.

arr._proto_ (reference)

Array.prototype (actual object)

String.prototype

Every object in JavaScript has a built-in property, which is called its **prototype**. The prototype is itself an object, so the prototype will have its own prototype, making what's called a **prototype chain**. The chain ends when we reach a prototype that has `null` for its own prototype.



Object Prototypes

Prototypes are the mechanism by which JavaScript objects inherit features from one another.

It is like a single template object that all objects inherit methods and properties from without having their own copy.

arr._proto_ (reference)

Array.prototype (actual object)

String.prototype

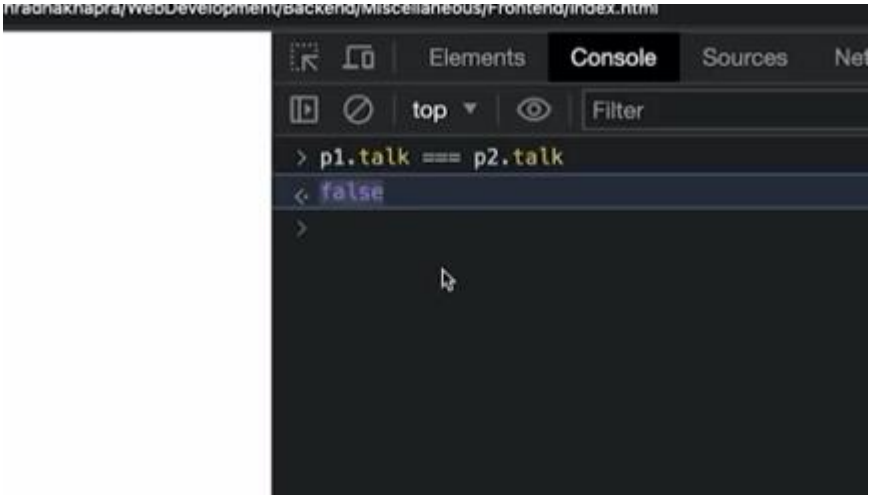
Every object in JavaScript has a built-in property, which is called its **prototype**. The prototype is itself an object, so the prototype will have its own prototype, making what's called a **prototype chain**. The chain ends when we reach a prototype that has `null` for its own prototype.

Factory Functions

A function that creates objects

person { name age }

talk



New operator

The new operator lets developers create an instance of a user-defined object type or of one of the built-in object types that has a constructor function.

```
function Person(name, age) {
  this.name = name;
  this.age = age;
}

Person.prototype.talk = function () {
  console.log(`Hi, my name is ${this.name}`);
};

let p1 = new Person("adam", 25);
let p2 = new Person("eve", 25);
```

Default levels | No Issues

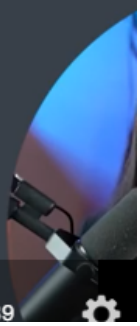
```
▶ Person {name: 'Aishu', age: 21}      app.js:20
▶ Person {name: 'Amrutha', age: 21}    app.js:20
> p1.talk === p2.talk
< true
>
```

Classes

Classes are a template for creating objects

The constructor method is a special method of a class for creating and initializing an object instance of that class.

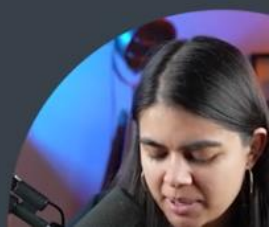
```
class Person {
  constructor(name, age) {
    this.name = name;
    this.age = age;
  }
  talk() {
    console.log(`Hi, my name is ${this.name}`);
  }
}

let p1 = new Person("adam", 25);
let p2 = new Person("eve", 25);
```

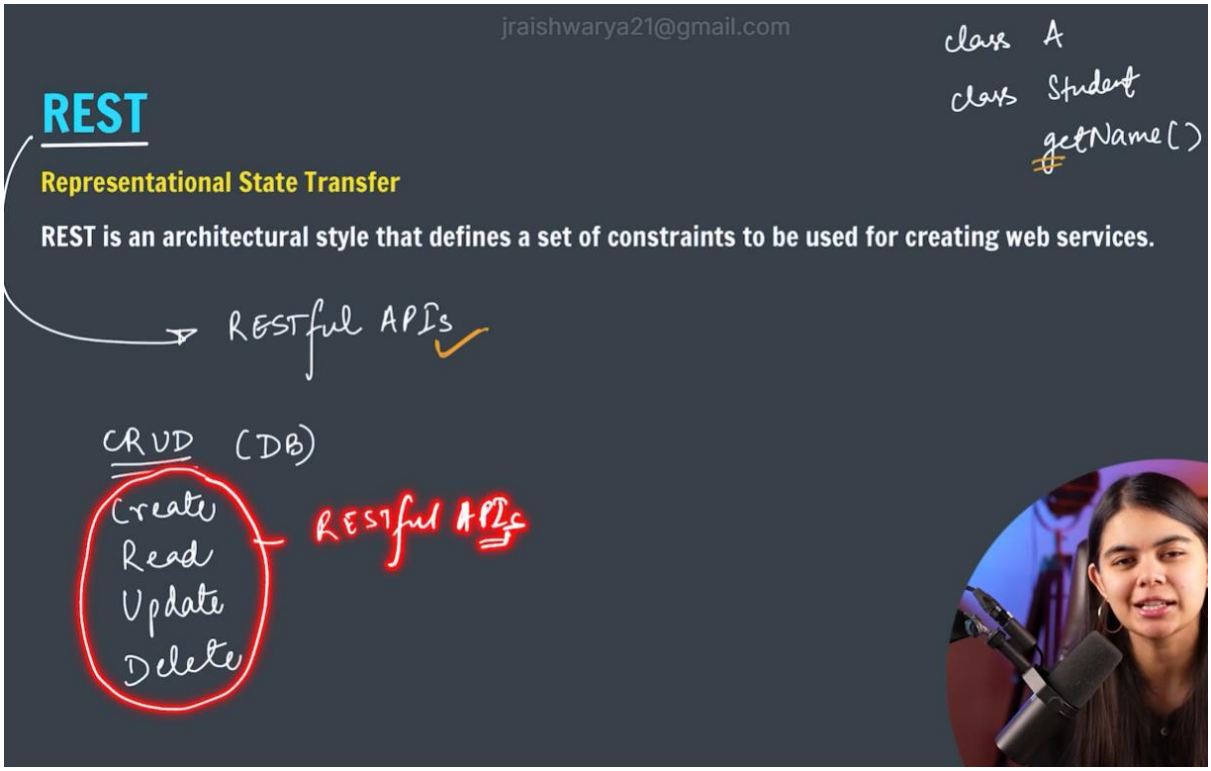



Inheritance

Inheritance is a mechanism that allows us to create new classes on the basis of already existing classes.

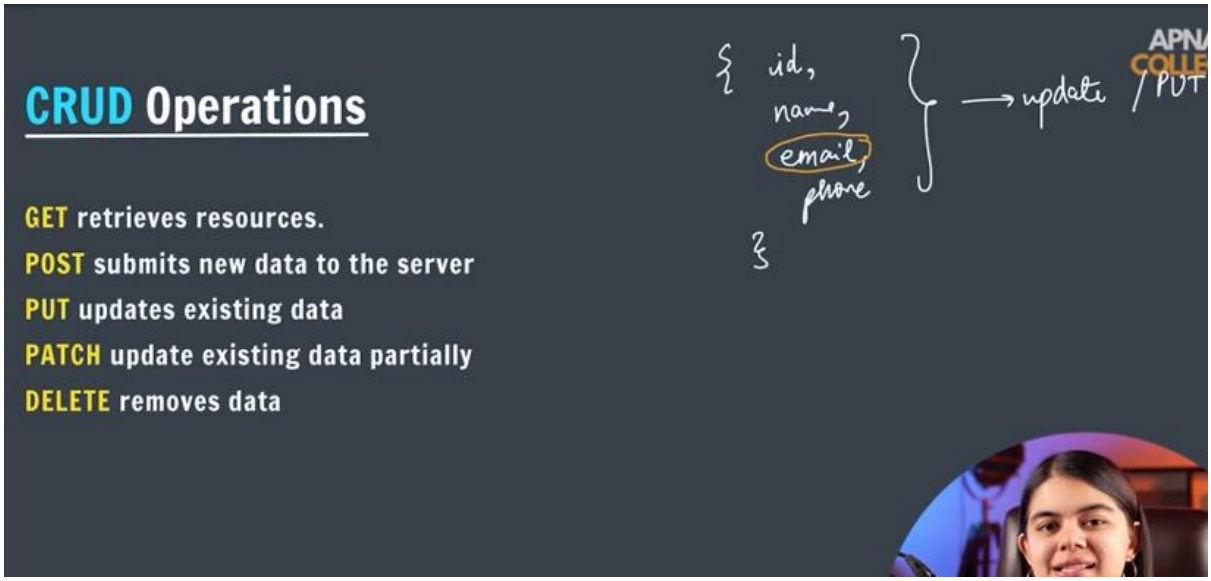
```
class Student extends Person {
  constructor(name, age, marks) {
    super(name, age);
    this.marks = marks;
  }

  greet() {
    return "hello!";
  }
}

let s1 = new Student("adam", 25, 95);
```





https://developer.x.com/en/docs/api-reference-index

https://stackoverflow.blog/2020/03/02/best-practices-for-rest-api-design/



Creating RESTful APIs

GET	/posts	to get data for all posts	INDEX (main)
POST	/posts	to add a new post	CREATE
GET	/posts/:id	to get one post (using id)	VIEW
PATCH	/posts/:id	to update specific post	UPDATE
DELETE	/posts/:id	to delete specific post	

Implement : POST /posts

Create Route

POST /posts to add a new post

2 routes

- Serve the form GET /posts/new
- Add the new post POST /posts

1) (user, content) — form

2) POST new post add to arr

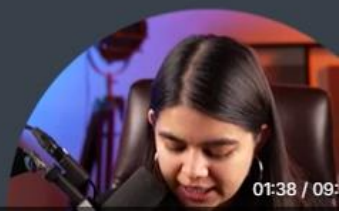
Implement : POST /posts

Create Route

POST /posts to add a new post

2 routes

- Serve the form GET /posts/new
- Add the new post POST /posts



Redirect

res.send( ) → text, html, object

res.render → ejs

res.redirect(URL)

Implement : GET /posts/:id

Show Route

id → 1, 2, 3, 4

1a, 2b, 3c

"abcdef" → "ghij"

GET	/posts/:id	to get one post (using id)

1 2 3 4, 5, 6, 7, 8

counter, index

Create id for Posts

UUID Package

Universally unique identifier

npm install uuid

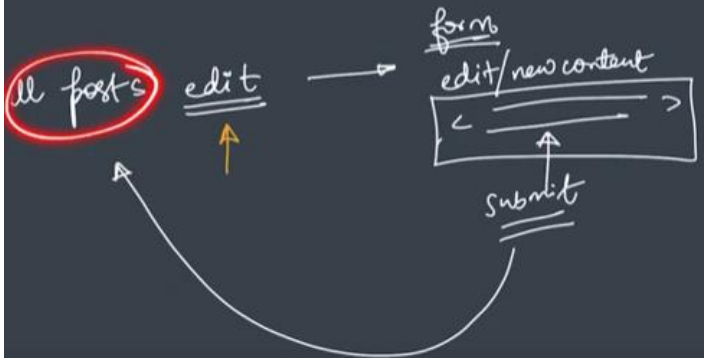
id, username, content

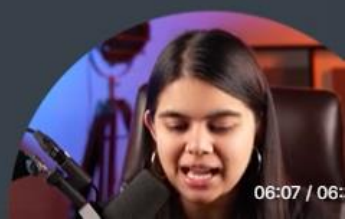
edit

Implement : PATCH /posts/:id

Update Route

PATCH /posts/:id to update specific post





html — get / post

Create Form for Update

Edit Route

Serve the edit form **GET** /posts/:id/edit

Implement : /posts/:id

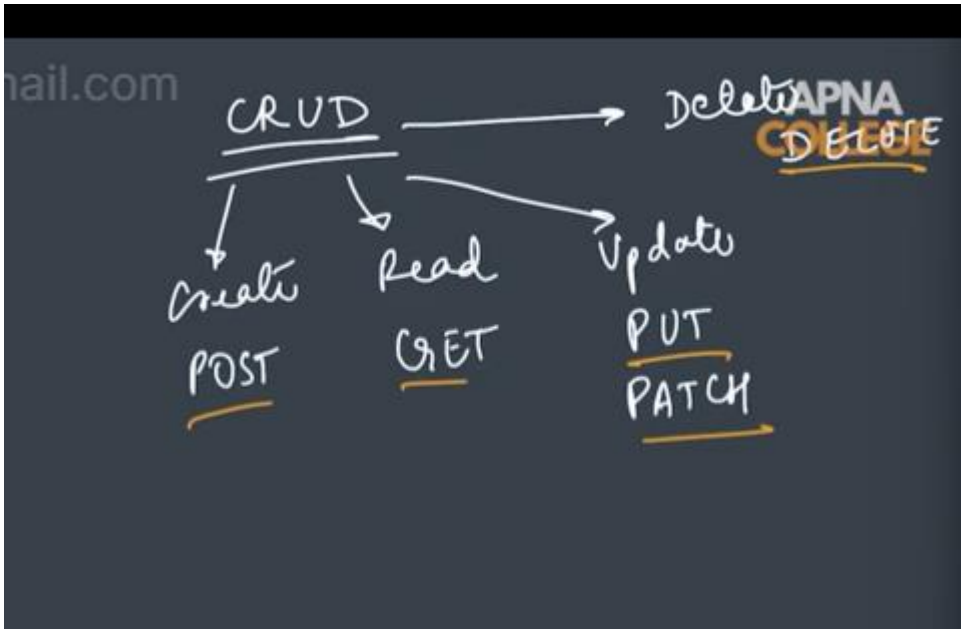
Destroy Route

DELETE /posts/:id to delete specific post

Implement : /posts/:id

Destroy Route

DELETE /posts/:id to delete specific post


CRUD
Delete
DELETE
Create
POST
Read
GET
Update
PUT
PATCH