DISCORD BOT- DEMO

Discord is one of the most widely used apps in the world.
We can create our own bot in our server and deploy It in realtime.
Our bot can give realtime replies as per our coding format in Node JS.
WE HAVE to use a package for it called discord.js npm package.

Introduction to discord.js (Discord NPM Package)

1. Overview

discord.js is a powerful Node.js module that allows you to interact with the Discord API to build bots, automation tools, and integrations. It abstracts the complexities of directly interacting with the Discord API, providing developers with an easy-to-use, object-oriented interface.

2. Installation

To install discord.js, you need Node.js version 16.9.0 or newer.

Run the following command:

```
npm install discord.js
```

3. Key Features

Some of the major features of discord.js include:

- • Easy interaction with the Discord API.
- • Object-oriented design for managing servers (guilds), users, and channels.
- • Support for message handling, embeds, attachments, and reactions.
- • Voice support for music bots and audio features.
- • Rich set of events to track server activities (e.g., user joining/leaving, message creation).
- • Built-in support for slash commands and interactions.

4. Basic Usage

Here's a simple example of creating a Discord bot with discord.js:

```
const { Client, GatewayIntentBits } = require('discord.js');
const client = new Client({ intents: [GatewayIntentBits.Guilds,
```

```
GatewayIntentBits.GuildMessages, GatewayIntentBits.MessageContent] });

client.once('ready', () => {
  console.log(`Logged in as ${client.user.tag}!`);
});

client.on('messageCreate', message => {
  if (message.content === '!ping') {
    message.channel.send('Pong!');
  }
});

client.login('YOUR_BOT_TOKEN');
```

5. Use Cases

discord.js can be used to build various types of bots and automation tools, such as:

- • Moderation bots (managing users, banning, kicking, filtering messages).
- • Fun bots (games, memes, trivia).
- • Utility bots (reminders, polls, logging).
- • Music bots (playing music in voice channels).
- • Community management tools (auto-roles, announcements, welcome messages).

6. Advantages

- • Well-documented and widely supported in the community.
- • Constantly updated with Discord's API changes.
- • Strong ecosystem of tutorials, guides, and extensions.

7. Conclusion

discord.js is one of the most popular libraries for creating Discord bots. Its rich set of features, ease of use, and active community make it an excellent choice for developers who want to build interactive and feature-rich Discord bots.

PROJECT DETAILS:

Some prerequisitories: You should have a running server in discord .
Create a bot from DISCORD DEVELOPERS from a browser, and copy the token for the bot.
NOTE: the token generated is private and shouldn't be given to anyone or uploaded online ."

NOW YOU CAN INITIATE A NEW PROJECT IN NODEJS.

- First you need to acquire client and GATEWAYINTENTBITS from discord.js module using destructing object.
  //Client basically represents the bot itself
  //GatewayIntent represents the different events the bot can listen to

🎯 What are Gateway Intents?

- When your bot connects to Discord, it can "listen" to different kinds of activities happening in servers (guilds).
  But instead of giving your bot **everything**, Discord makes you **choose what you need**.
  👉 These choices are called **Intents**.

- 🔑 **Examples of Intents**
- GatewayIntentBits.Guilds → Bot can see servers (guilds) it's in.
- GatewayIntentBits.GuildMessages → Bot can see messages sent in servers.
- GatewayIntentBits.MessageContent → Bot can read the actual text of messages.
- GatewayIntentBits.GuildMembers → Bot can see members joining/leaving.
- GatewayIntentBits.GuildVoiceStates → Bot can see who joins/leaves voice channels.
- **IN SHORT :**
- GatewayIntentBits are like a **filter**. Instead of giving your bot **all events**, you choose **what events your bot can see and respond to**.

Next step is creating a client as:

```
const client = new Client({
    //Intents
});
```

Now we can add events :

```
client.on("ready", () => {
    console.log(`Logged in as ${client.user.tag}`);
});//ready is a basic event that is emitted when the bot is ready
//client.user.tag represents the tag of the user (bot) that is logged in

```

For bot to work you must login it at the end as :

```
client.login(code);//logging in with token of our bot
```

You can add other certain eventlisteners as :

```
client.on("messageCreate",(message)=>//This is an event listener that
is triggered when a new message is created by any user, even if
created by bot itself it will get triggered
{
    //message is basically a message object that is passed everytime a
    message is created in the server
    // console.log(message);
    console.log(message.content);//content is a property of message
    object that contains the actual text of the message
})
```

- Giving a reply when a message is triggered:
  Use message.reply() listener

Now, what are commands in discord and how to create custom commands for out bot:

🎯 What are Commands in discord.js?

Commands are the **instructions you give to your bot** through messages or slash commands.
They tell the bot **what action to perform** when a user types something.

🔑 Types of Commands

1. **Prefix Commands (old style)**
    - User types something like !ping or !help.
    - Your bot listens for messages starting with ! (or any prefix you set).
2. if (message.content === "!ping") {
3. message.reply("Pong!");
4. }
5. **Slash Commands (modern way, recommended)**
    - User types /ping in the chat.

- Discord shows the command in the UI with autocomplete.
- These are **registered with Discord** and are safer + more structured.

Example of a command interaction from index.js:

```
client.on("interactionCreate", async (interaction) => {
    // console.log(interaction);
    interaction.reply("Pong!");
});
```

NOTE : interaction create is a special type of listener THAT IS DIFFERENT FROM a normal message :

- **Types of Interactions**

1. **Chat Input Command (Slash Commands)**
    - Example: /ping
    - Most common — your bot checks interaction.commandName.
2. **User Command (Context Menu)**
    - Right-click a **user → Apps → YourCommand**
    - Interaction with a user directly.
3. **Message Command (Context Menu)**
    - Right-click a **message → Apps → YourCommand**
    - Interaction with a specific message.
4. **Component Interactions** (UI elements)
    - **Buttons** → Clicking a button under a message.
    - **Select Menus** → Choosing an option from a dropdown.
5. **Modal Submit**

- When a user fills out a **popup form (modal)** and submits it.

Above code will reply pong to all interactions. You can apply further conditions in your code.

📦 Why Commands are Useful?

- Keep your bot **organized** (each command does one job).
- Easier for users (clear and predictable).
- Slash commands come with **auto-complete, descriptions, and permissions**.

✅ In short:

Commands in discord.js are **ways for users to interact with your bot** — either by typing a **prefix command** (!play) or a **slash command** (/play).

REST AND ROUTES OBJECT FROM DISCORD.JS:

📌 REST and Routes in discord.js

🔹 REST

- A class in discord.js used to communicate directly with the **Discord API**.

- Required when you want to **register or update slash commands**.
- Works by sending HTTP requests to Discord's servers.

🔹 **Routes**

- A helper provided by discord.js to get the **correct API endpoints** easily.
- Used along with REST to tell Discord *where* to register commands (globally or in a specific server).

🔹 **Purpose Together**

- **REST + Routes** are mainly used for **managing application (slash) commands**.
- REST handles the request.
- Routes provides the correct location (endpoint).
- This combination allows your bot's slash commands to appear in Discord's UI.

✅ **In short:**

- **REST** = The messenger (sends commands to Discord).
- **Routes** = The address book (tells REST where to send commands).
- Together, they are used to register and manage **slash commands**.

REGISTERING CUSTOM COMMANDS TO THE DISCORD SERVER:

```
const token_obj = require("./code.json");

// Extracting the bot token from the JSON file
const token = token_obj.code;

// Importing REST and Routes from discord.js
// REST is used for interacting with Discord's REST API
// Routes provides API route methods (for registering commands, etc.)
const { REST, Routes } = require("discord.js");

// Defining the commands we want to register
// Each command object must contain at least a "name" and "description"
const commands = [
    {
        name: "ping",              // Command name (what user types in Discord: /ping)
        description: "Replies with Pong!" // Short description shown in Discord
    }
];

// Creating a REST client with version 10 (latest Discord API)
// Then we set our bot token so Discord knows which bot is registering commands
const rest = new REST({ version: '10' }).setToken(token);

// Async function to register slash commands to Discord
Tabnine | Edit | Test | Explain | Document
async function registerCommands() {
    try {
        console.log("Started registering application commands...");

        // Using Discord's REST API to register commands
        // Routes.applicationCommands() requires the APPLICATION_ID (client id of your bot)
        // "commands" array is sent in the body so Discord knows which commands to add
        await rest.put(
            Routes.applicationCommands(token_obj.client),
            { body: commands }
        );

        console.log("Successfully registered application commands ✅");
    } catch (err) {
        // If something goes wrong (wrong token, client id, etc.), log the error
        console.log("Error while registering commands ❌:", err);
    }
}

// Calling the function to actually run the registration
registerCommands();

// Note: This script only registers commands globally on Discord.
// It does not make the bot respond — you need separate code with Client & events for handling commands
// You can run this script once to register commands, then run your bot code separately.

//THIS REGISTERS A COMMAND FOR YOUR BOT IN THE DISCORD SERVER
```