Complete Explanation: Express TODO API for Beginners

This document explains how to create a basic TODO API using Node.js and the Express framework. It includes a POST route to add tasks and a GET route to fetch all tasks. Each part of the code is explained clearly for beginners.

Code:

```
const express = require('express')
const app = express()

// Middleware to parse JSON data from requests
app.use(express.json())

const taskList = []

// POST route to add a new task
app.post('/todo', (req, res) => {
  const { title, description } = req.body

  const newTask = {
    id: taskList.length + 1,
    title,
    description
  }

  taskList.push(newTask)

  res.status(201)
  console.log(taskList)
  res.json(taskList)
})

// GET route to retrieve all tasks
app.get('/todo', (req, res) => {
  res.json(taskList)
})

const port = 3000

// Start the server
```

```
app.listen(port, () => {
  console.log(`The server is live on port ${port}`)
})
```

Code Explanation

const express = require('express') – Imports the Express module. Express helps simplify web server creation in Node.js.

const app = express() – Creates an instance of the Express app which will handle requests and responses.

app.use(express.json()) – Enables middleware to parse JSON data sent in request bodies. Required to access req.body.

const taskList = [] – Creates an empty array to store tasks. Each task will be an object with an id, title, and description.

app.post('/todo', (req, res) => {...}) – Defines a POST route at /todo. This route accepts a JSON body and adds a new task to the taskList.

const { title, description } = req.body – Destructures the title and description properties from the incoming request body.

id: taskList.length + 1 – Generates a unique ID by adding 1 to the current number of tasks.

res.status(201) – Sets the response status code to 201, meaning 'Created'.

res.json(taskList) – Sends the updated taskList back to the client in JSON format.

app.get('/todo', (req, res) => {...}) – Defines a GET route at /todo to retrieve all tasks in the taskList.

app.listen(port, () => {...}) – Starts the server and makes it listen for requests on port 3000.

How Requests Work

There are different HTTP methods used to interact with servers:

• GET – Used to retrieve data

• POST – Used to send data to the server

When you visit http://localhost:3000/todo in a browser, it sends a GET request. Therefore, the app.get('/todo') route handles it and returns all tasks.

To add tasks, use Postman or any API client to send a POST request to http://localhost:3000/todo with JSON data like:

{ "title": "Buy milk", "description": "From the local store" }

Conclusion

You've now built a simple TODO API with two main features:

1. POST /todo – Adds a new task

2. GET /todo – Retrieves the full list of tasks

This is a foundational step in building a full-stack app.