TIME SERVER

HTTP JSON API

Presented By:

YIN YIN PHYO

OUTLINE

1. Introduction
2. Design
3. Implementation
4. Enhancement Ideas
5. Conclusion

INTRODUCTION

- This project focuses on building an HTTP server using Node.js that serves time-based data through APIs.
- The server provides various time formats on request, allowing clients to retrieve current time, parsed time, and UNIX epoch time.

TIME SYNCHRONIZATION

- The server can be used in applications where precise time synchronization across different clients is essential.
- Example: Distributed systems requiring consistent time references.

TIMESTAMP PARSING FOR APPLICATIONS

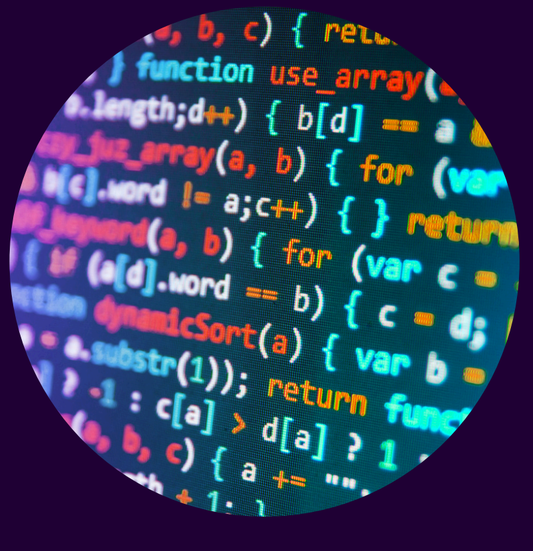
- The server helps applications convert ISO-formatted timestamps into human-readable time components such as hour, minute, and second.
- Example: Applications needing to display logs or events based on user-specific time formats.

UNIX TIME MANAGEMENT

- Provides the UNIX epoch time (milliseconds since 1970-01-01), useful for applications requiring consistent and standardized timestamp formats.
- Example: Database systems storing events as UNIX timestamps for future retrieval.

DESIGN

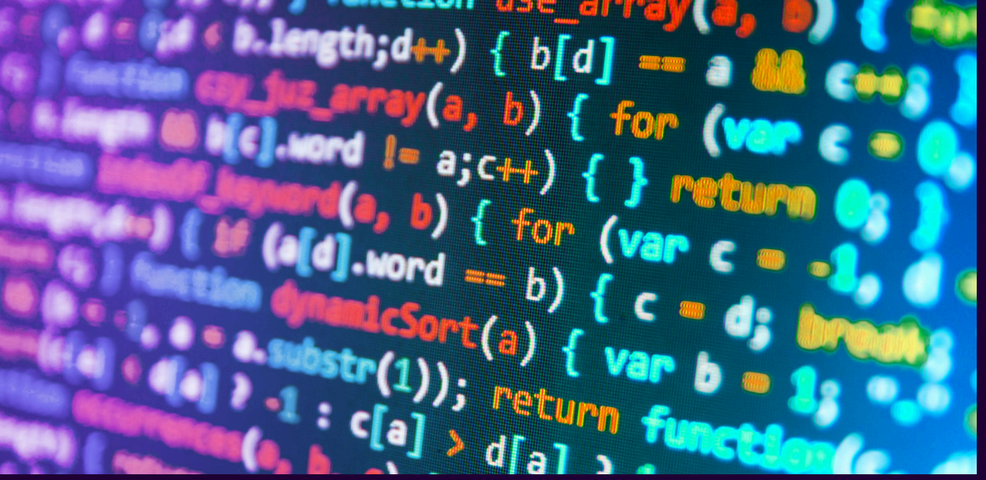
- Use of built-in http and url modules for simplicity.
- Created multiple endpoints:
  - /api/parsetime: Returns time in JSON (hour, minute, second).
  - /api/unixtime: Returns UNIX epoch time.
  - /api/currenttime: Returns the current date and time in JSON format.

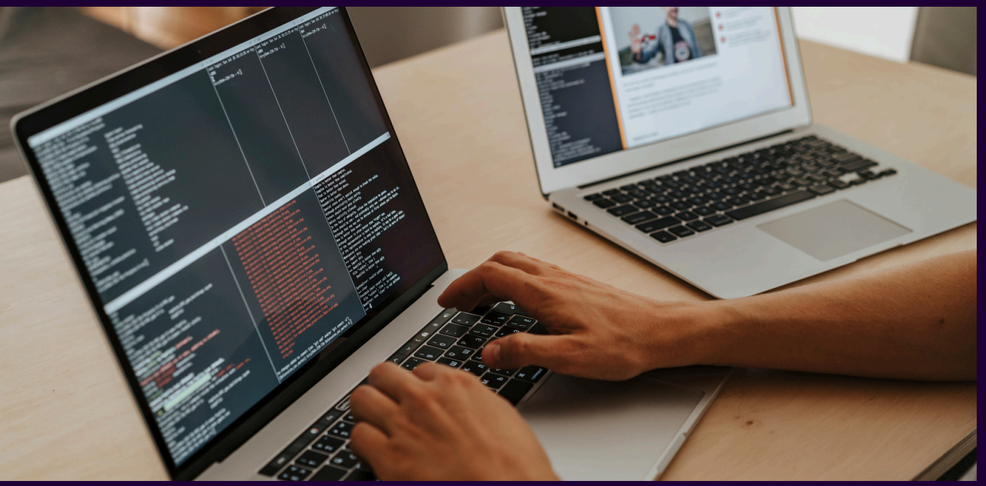


IMPLEMENTATION

TIME FORMATTING USING DATE()

- The JavaScript Date() object is used to extract and format the current time or a specific ISO timestamp.
- Functions like getHours(), getMinutes(), and getSeconds() are used to retrieve the time components.
- zeroFill() function ensures single-digit values (e.g., 9:05) are displayed in a two-digit format (e.g., 09:05).





IMPLEMENTATION

API ENDPOINTS

/api/parsetime:

- This endpoint expects a query parameter with an ISO-formatted timestamp (e.g., /api/parsetime?iso=2023-10-07T12:34:56Z).
- It returns the parsed time (hour, minute, second) in JSON format.

/api/unixtime:

- Accepts the same ISO timestamp, but instead returns the UNIX epoch time (milliseconds since 1970-01-01).
- Example: { "unixtime": 1680221234000 }.

/api/currenttime:

- This endpoint returns the current date and time in JSON format (year, month, day, hour, and minute).
- Example: { "year": 2024, "month": 10, "date": 7, "hour": 14, "minute": 23 }.

IMPLEMENTATION

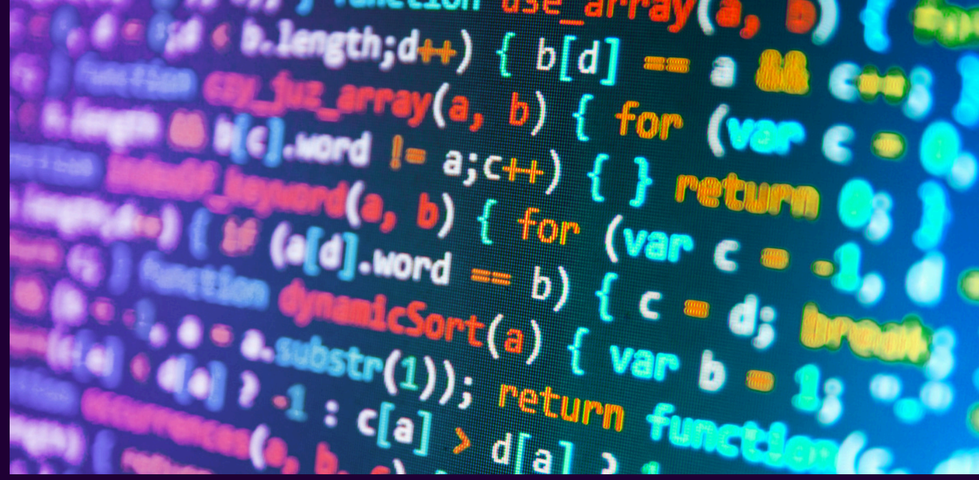
SERVER SETUP

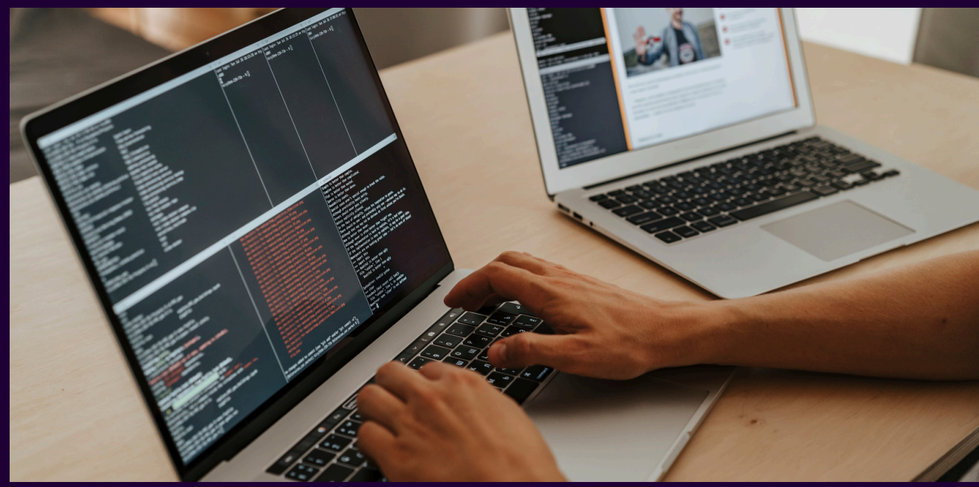
createServer():

- The http.createServer() method is used to create the server, with a callback function that handles incoming requests.
- The server listens for requests, matches the requested URL path, and returns the appropriate time format based on the endpoint.

Port setup:

- The server listens on the port specified via command-line arguments (process.argv[2]).
- This allows flexibility in specifying different ports without modifying the code.
- Example: node server.js 8000 would start the server on port 8000.





ENHANCEMENT IDEAS

Additional time formats or time zones.

Implementing error handling for invalid time inputs.

Adding more endpoints for detailed time analysis (e.g., millisecond-level precision).

CONCLUSION

Node.js offers an efficient and straightforward way to build API servers, especially for handling I/O-intensive tasks like serving time-based data.

Utilizing Node's non-blocking, event-driven architecture allows the server to handle multiple requests simultaneously, ensuring low latency and high performance.

The project demonstrates how easily APIs can be created using built-in HTTP and URL modules, without the need for additional libraries or frameworks.

Thank You!