2.2 JAVASCRIPT IN THE DESKTOP WITH NODE JS

What is Node.JS?

Node.js is an open-source, cross-platform JavaScript **runtime environment** that allows developers to run JavaScript code on the server side. Created by Ryan Dahl in 2009, Node.js has revolutionized server-side programming by offering an efficient, event-driven, and non-blocking I/O model.

It's a powerful tool used for various types of projects. Let's explore some key aspects:

- **JavaScript Runtime**: Node.js runs on the **V8 JavaScript engine**, which is also the core engine behind Google Chrome.
- **Single Process Model**: A Node.js application operates within a **single process**, avoiding the need to create a new thread for every request.
- **Asynchronous I/O**: Node.js provides a set of **asynchronous I/O primitives** in its standard library. These primitives prevent JavaScript code from blocking, making non-blocking behavior the norm.
- **Concurrency Handling**: Node.js efficiently handles **thousands of concurrent connections** using a single server. It avoids the complexities of managing thread concurrency, which can lead to bugs.
- **JavaScript Everywhere**: Frontend developers familiar with JavaScript can seamlessly transition to writing server-side code using Node.js.
- **ECMAScript Standards**: Node.js supports the latest ECMAScript standards. You can choose the version you want to use, independent of users' browser updates.

Why Node.JS?

Node.js is used to build back-end services like **APIs** like Web App, Mobile App or Web Server. A Web Server will open a file on the server and return the content to the client. It's used in production by large companies such as **Paypal**, **Uber**, **Netflix**, **Walmart**, and so on.

Reasons to Choose Node.js

- **Easy to Get Started**: Node.js is beginner-friendly and ideal for prototyping and agile development.
- **Scalability**: It scales both horizontally and vertically.
- **Real-Time Web Apps**: Node.js excels in real-time synchronization.

OBSERVE OPTIMIZE OUTSPREAD

- **Fast Suite**: It handles operations quickly (e.g., database access, network connections).
- **Unified Language**: JavaScript everywhere—frontend and backend.
- **Rich Ecosystem**: Node.js boasts a large open-source library and supports asynchronous, non-blocking programming.

PHP and ASP handling file requests:

Send Task -> Waits -> Returns -> Ready for Next Task

Node.js handling file request:

Send Task -> Returns -> Ready for Next Task

Node.js takes requests from users, processes those requests, and returns responses to the corresponding users, there is no Wait for open and read file phase in Node.js.

Basic Concepts of Node.JS

The following diagram depicts some important parts of Node.js that are useful and help us understand it better.

Node.js Example to Create Web Server

It is the basic code example to create node.js server.

```
// Importing the http module

const http = require('http');
```

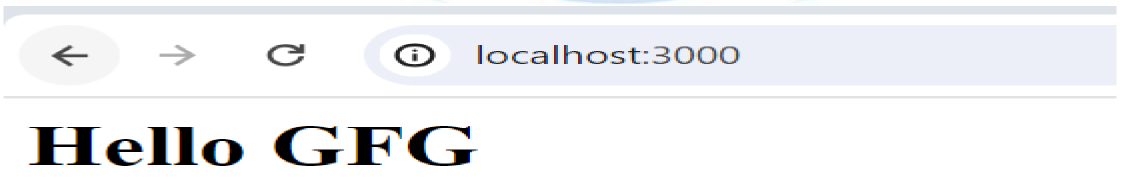
ROHINI COLLEGE OF ENGINEERING AND TECHNOLOGY
PALKULAM, KANYAKUMARI
OBSERVE OPTIMIZE OUTSPREAD

```
// Creating a server
const server = http.createServer((req, res) => {
    // Setting the content type to HTML
    res.writeHead(200, {
        'Content-Type': 'text/html'
    });

    // Sending the HTML response
    res.end('<h1>Hello GFG</h1>');
});

// Listening on port 3000
const PORT = 3000;
server.listen(PORT, () => {
    console.log(`Server running at http://localhost:${PORT}/`);
});
```

Output:



Hello GFG

Example of Node.js Server Output

Code Explaination:

- We use the **HTTP module** to create an **HTTP server.**

- The server listens on the specified **port** and **hostname**.
- When a new request arrives, the callback function handles it by setting the response status, headers, and content.

How Node.JS Works?

Node.js accepts the request from the clients and sends the response, while working with the request node.js handles them with a single thread. To operate I/O operations or requests node.js use the concept of threads. Thread is a sequence of instructions that the server needs to perform. It runs parallel on the server to provide the information to multiple clients. Node.js is an event loop single-threaded language. It can handle concurrent requests with a single thread without blocking it for one request.

Advantages of Node.JS

- **Easy Scalability:** Easily scalable the application in both horizontal and vertical directions.
- **Real-time web apps:** Node.js is much more preferable because of faster synchronization. Also, the event loop avoids HTTP overloaded for Node.js development.
- **Fast Suite:** NodeJS acts like a fast suite and all the operations can be done quickly like reading or writing in the database, network connection, or file system
- **Easy to learn and code:** NodeJS is easy to learn and code because it uses JavaScript.
- **Advantage of Caching:** It provides the caching of a single module. Whenever there is any request for the first module, it gets cached in the application memory, so you don't need to re-execute the code.

What is Node.JS file?

Node.js files contain tasks that handle file operations like **creating**, **reading**, **deleting**, etc., Node.js provides an inbuilt module called FS (File System).

Application of Node.JS

Node.js is suitable for various applications, including:

- Real-time chats
- Complex single-page applications

- Real-time collaboration tools
- Streaming apps
- JSON APIs

Common Use Cases of Node.JS

Node.js is versatile and finds applications in various domains:

1. **Web Servers**: Node.js excels at building lightweight and efficient web servers. Its non-blocking I/O model makes it ideal for handling concurrent connections.
2. **APIs and Microservices**: Many companies use Node.js to create RESTful APIs and microservices. Express.js simplifies API development.
3. **Real-Time Applications**: Node.js shines in real-time scenarios like chat applications, live notifications, and collaborative tools. Socket.io facilitates real-time communication.
4. **Single-Page Applications (SPAs)**: SPAs benefit from Node.js for server-side rendering (SSR) and handling API requests.
5. **Streaming Services**: Node.js is well-suited for streaming data, whether it's video, audio, or real-time analytics.

Node.JS Ecosystem

Node.js has a vibrant ecosystem with a plethora of libraries, frameworks, and tools. Here are some key components:

1. **npm (Node Package Manager)**: npm is the default package manager for Node.js. It allows developers to install, manage, and share reusable code packages (called modules). You can find thousands of open-source packages on the npm registry.
2. **Express.js**: Express is a popular web application framework for Node.js. It simplifies routing, middleware handling, and request/response management. Many developers choose Express for building APIs, web servers, and single-page applications.
3. **Socket.io**: For real-time communication, Socket.io is a go-to library. It enables bidirectional communication between the server and clients using WebSockets or fallback mechanisms.

4. **Mongoose**: If you’re working with MongoDB (a NoSQL database), Mongoose provides an elegant way to model your data and interact with the database. It offers schema validation, middleware, and query building.

ROHINI COLLEGE OF ENGINEERING AND TECHNOLOGY
PALKULAM, KANYAKUMARI
OBSERVE OPTIMIZE OUTSPREAD