Node.js

Node.js Introduction

Node.js is an open-source, cross-platform runtime environment that allows JavaScript to run outside the browser. It is mainly used for server-side and backend development. Node.js is based on the V8 JavaScript engine, developed by Google Chrome. It enables developers to build non-blocking, event-driven servers, making it ideal for high-performance and scalable applications.

Features of Node.js:

1. ***Asynchronous and Event-Driven***
   Node.js is asynchronous, meaning it doesn't wait for a task to complete before moving to the next one. This makes it fast and efficient.
2. ***Fast Execution***
   Thanks to the V8 engine, JavaScript code execution is extremely fast.
3. ***Single-Threaded but Highly Scalable***
   Node.js follows a single-threaded architecture but can handle multiple requests using its event loop mechanism.
4. ***Rich Package Ecosystem***
   Node.js comes with npm (Node Package Manager), which hosts thousands of libraries to simplify and speed up development.

Additional Features:

- ***Non-blocking I/O****: Node.js performs non-blocking operations by default, making it ideal for I/O-heavy applications.*
- ***Networking Support****: It supports TCP/UDP sockets, essential for building low-level network applications that browsers cannot handle.*
- ***File System Access****: It provides APIs to read and write files directly, unlike browsers which restrict file system access for security reasons.*
- ***Server-Side Capabilities****: Node.js allows JavaScript to handle HTTP requests, file operations, and server-side tasks.*
- ***Modules****: Code can be organized into reusable modules using require().*

Features Not Present in Node.js:

- ***Window Object****: Node.js does not have the browser-specific global window object.*
- ***DOM Manipulation****: Node.js lacks the Document Object Model (DOM) APIs since it's not meant for manipulating webpage content.*
- ***BOM (Browser Object Model)****: Node.js doesn't directly interact with browser-specific objects like navigator or screen.*

Note:
These features were never a part of Node.js because its primary focus is on server-side programming, not browser-related functionalities.

Why These Browser Features Are Missing in Node.js:

1. ***Window Object****:*
   In browsers, the window object provides access to browser features like alert boxes, URL location, etc. Node.js, being a server-side environment, uses a different global object suitable for server operations.

2. ***DOM Manipulation**:*
   DOM is specific to browsers for interacting with HTML and CSS. Node.js deals with backend logic, so it doesn't include DOM APIs.
3. ***BOM (Browser Object Model)**:*
   BOM features like navigator and screen are browser-specific and not needed in a server environment.

If You Need Browser-like Features in Node.js:

- *For **DOM manipulation**, libraries like **jsdom** can simulate a browser environment.*
- *For **BOM-like functionalities**, specific external libraries must be used.*

*Node.js is intentionally optimized for backend server operations, **not** for browser-based functionalities.*

Use Cases of Node.js:

- ***Backend Development**: Creating APIs like RESTful or GraphQL APIs.*
- ***Real-Time Applications**: Building chat apps, live notifications, gaming servers.*
- ***Microservices Architecture**: Developing scalable and modular applications.*
- ***Streaming Applications**: Suitable for streaming platforms like Netflix.*

Why Learn Node.js?

- *High demand in backend development.*
- *Essential for becoming a full-stack developer (JavaScript on both frontend and backend).*
- *Ideal for building real-time apps like chats, games, and collaborative tools (e.g., Google Docs).*
- *Excellent support for WebSockets and server-sent events.*

Next Topic

Node.js Installation and Basic Setup

To use Node.js, you must install it on your system. Here's the step-by-step guide:

Node.js Installation (Windows/Linux/MacOS):

1. ***Download Node.js***
    - *Visit the official website: https://nodejs.org*
    - *You'll find LTS (Long-Term Support) and Current versions.*
        - *Beginners are recommended to install the **LTS version**.*
2. ***Run the Installer***
    - *After downloading, open the installer and follow the "Next" instructions.*
    - *npm (Node Package Manager) will also be installed automatically.*
3. ***Verify Installation***
    - *Open the terminal (Command Prompt or PowerShell) and run:*
    - *node -v*

to check the Node.js version.

- *Run:*
- *npm -v*

to check the npm version.

First Node.js Application:

1. *Create a folder, e.g., **MyNodeApp**.*
2. *Inside it, create a file, e.g., **app.js**.*
3. *Write a simple code:*
4. *// app.js*
5. *console.log("Hello, Node.js!");*

6. *Run it in the terminal:*
7. *node app.js*

Output:

Hello, Node.js!

Key Concepts in Node.js:

1. ***REPL (Read-Eval-Print Loop)***
   Node.js has an interactive terminal where you can directly execute JavaScript commands. Just type node in the terminal to start.
2. ***npm (Node Package Manager)***
   Used to install third-party libraries and modules. Example:
3. *npm install express*
4. ***Creating a Server***

```
5. const http = require("http");
6.
7. const server = http.createServer((req, res) => {
8.   console.log(req.url, req.headersSent, req.method);
9.
10.  res.setHeader("Content-Type", "text/html");
11.  res.write(`
12.    <!DOCTYPE html>
13.    <html>
14.    <head>
15.      <title>Form</title>
16.    </head>
17.    <body>
18.      <form action="/submit" method="post">
19.        <label for="name">Name:</label>
20.        <input type="text" id="name" name="name"><br><br>
21.        <label for="gender">Gender:</label>
22.        <select id="gender" name="gender">
23.          <option value="male">Male</option>
24.          <option value="female">Female</option>
25.          <option value="other">Other</option>
26.        </select><br><br>
27.        <input type="submit" value="Submit">
28.      </form>
29.    </body>
30.    </html>
```

```
  `);
  res.end();
});

server.listen(3000, () => {
  console.log("Server running at http://localhost:3000/");
});
```

Run the file:

node app.js

Open in browser:
http://localhost:3000

How to Stop a Running Node.js Server?

- ***1. Close from Terminal (CTRL + C)***
  - *Go to the terminal where the server is running.*
  - *Press **CTRL + C**.*
  - *The server will stop.*

Example:

node server.js

(Press CTRL + C to stop)

How DNS (Domain Name System) Works?

DNS is like the internet's phonebook, translating domain names (e.g., google.com) into IP addresses (e.g., 142.250.182.206). Without DNS, you would have to remember the numeric IP address of every website.

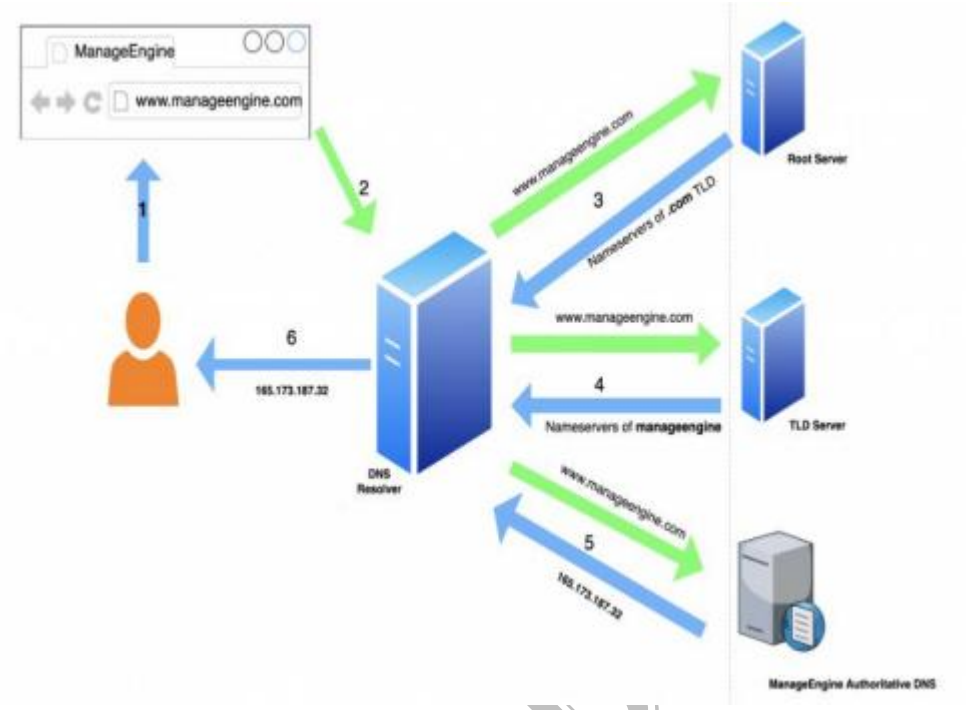
DNS Workflow:

1. ***User Request***
   - *When you type a URL like www.example.com, the browser needs to find out which server it should connect to.*
   - *It sends a request to the DNS system to get the IP address.*

2. ***DNS Resolver***
    - *The browser first contacts a DNS resolver, usually provided by your ISP.*
    - *The resolver's job is to find the correct IP address.*
3. ***Root DNS Server***
    - *If the resolver doesn't know, it asks the Root DNS server.*
    - *Root servers guide the resolver to the right TLD server.*
4. ***TLD (Top-Level Domain) Server***
    - *The resolver contacts the TLD server (e.g., .com for example.com) for further directions.*
5. ***Authoritative DNS Server***
    - *Finally, the resolver contacts the authoritative DNS server, which knows the exact IP address.*
6. ***IP Address Returned***
    - *The authoritative server sends the IP address back to the DNS resolver, which sends it to the browser.*
7. ***Connection Established***
    - *The browser uses the IP address to connect to the server and load the requested webpage.*

Step-by-Step Example:

1. *You type **www.google.com**.*
2. *Browser asks the DNS resolver for the IP address.*
3. *The resolver:*
    - *Queries the Root server → TLD server (.com) → Authoritative server.*
4. *The resolver receives the IP address (e.g., 142.250.182.206).*
5. *The browser connects to that IP address and loads the webpage.*


ManageEngine
www.manageengine.com
1
2
www.manageengine.com
3
Nameservers of .com TLD
Root Server
6
165.173.187.32
www.manageengine.com
4
Nameservers of manageengine
TLD Server
DNS Resolver
www.manageengine.com
5
165.173.187.32
ManageEngine Authoritative DNS