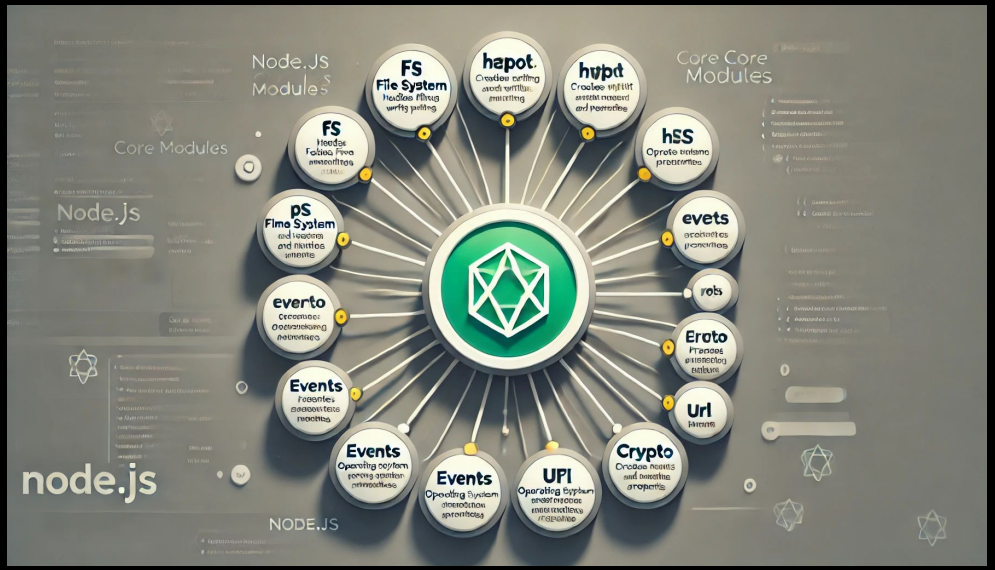
3.4 Node Core Modules



1. Built-in: Core modules are included with Node.js installation.
2. No Installation Needed: Directly available for use without npm install.
3. Performance: Highly optimized for performance.

3.4 Node Core Modules

1. fs (File System): Handles file operations like reading and writing files.
2. http: Creates HTTP servers and makes HTTP requests.
3. https: Launch a SSL Server.
4. path: Provides utilities for handling and transforming file
5. paths.os: Provides operating system-related utility methods and properties.
6. events: Handles events and event-driven programming.
7. crypto: Provides cryptographic functionalities like hashing and encryption.
8. url: Parses and formats URL strings.

3.5 Require Keyword

1. Purpose: Imports modules in Node.js.
2. Caching: Modules are cached after the first require call.
3. .js is added automatically and is not needed to at the end of module name.
4. Path Resolution: Node.js searches for modules in core, node_modules, and file paths.

Syntax:

```
const moduleName = require('module');
```

```
// Load the built-in http module
const http = require('http');

// Load the third party express module
const express = require('express');

// Load the custom myModule module
const myModule = require('./myModule');
```

3.6 Creating first Node Server

```
1  // Simple Node.js server
2  const http = require('http');
3
4  function requestListener(req, res) {
5      console.log(req);
6  }
7
8  http.createServer(requestListener);
```

3.6 Creating first Node Server

```
1  // Simple Node.js server
2  const http = require('http');
3
4  http.createServer(function (req, res) {
5    console.log(req);
6  });
```

3.6 Creating first Node Server

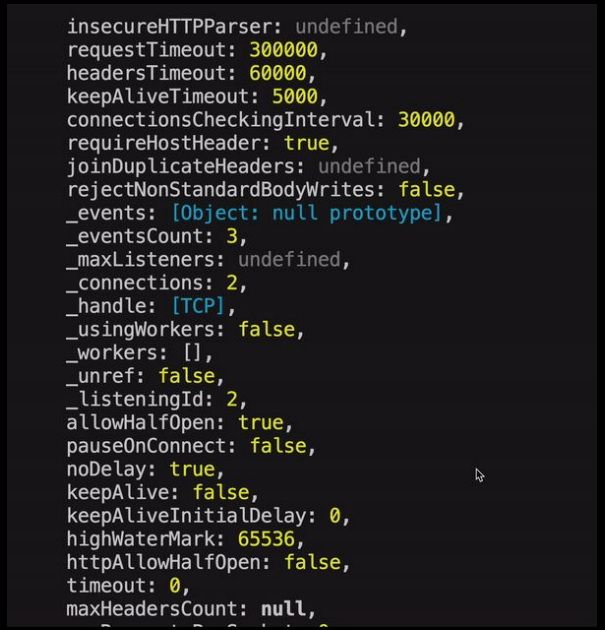
```
1  // Simple Node.js server
2  const http = require('http');
3
4  http.createServer((req, res) => {
5    console.log(req);
6  });
```

Run the code with:

node app.js

3.6 Creating first Node Server

```
1  // Simple Node.js server
2  const http = require('http');
3
4  const server = http.createServer((req, res) => {
5    console.log(req);
6  });
7
8  server.listen(3000);
```

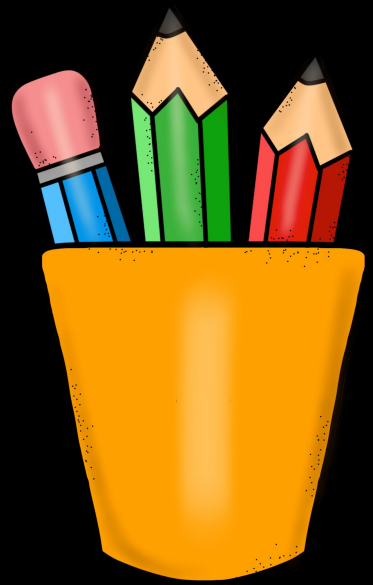


3.6 Creating first Node Server

```
1  // Simple NodeJS server
2  const http = require('http');
3
4  const server = http.createServer((req, res) => {
5    console.log(req);
6  });
7
8  const PORT = 3000;
9  server.listen(PORT, () => {
10   console.log(`Server running at http://localhost:${PORT}/`);
11 });
```

Revision

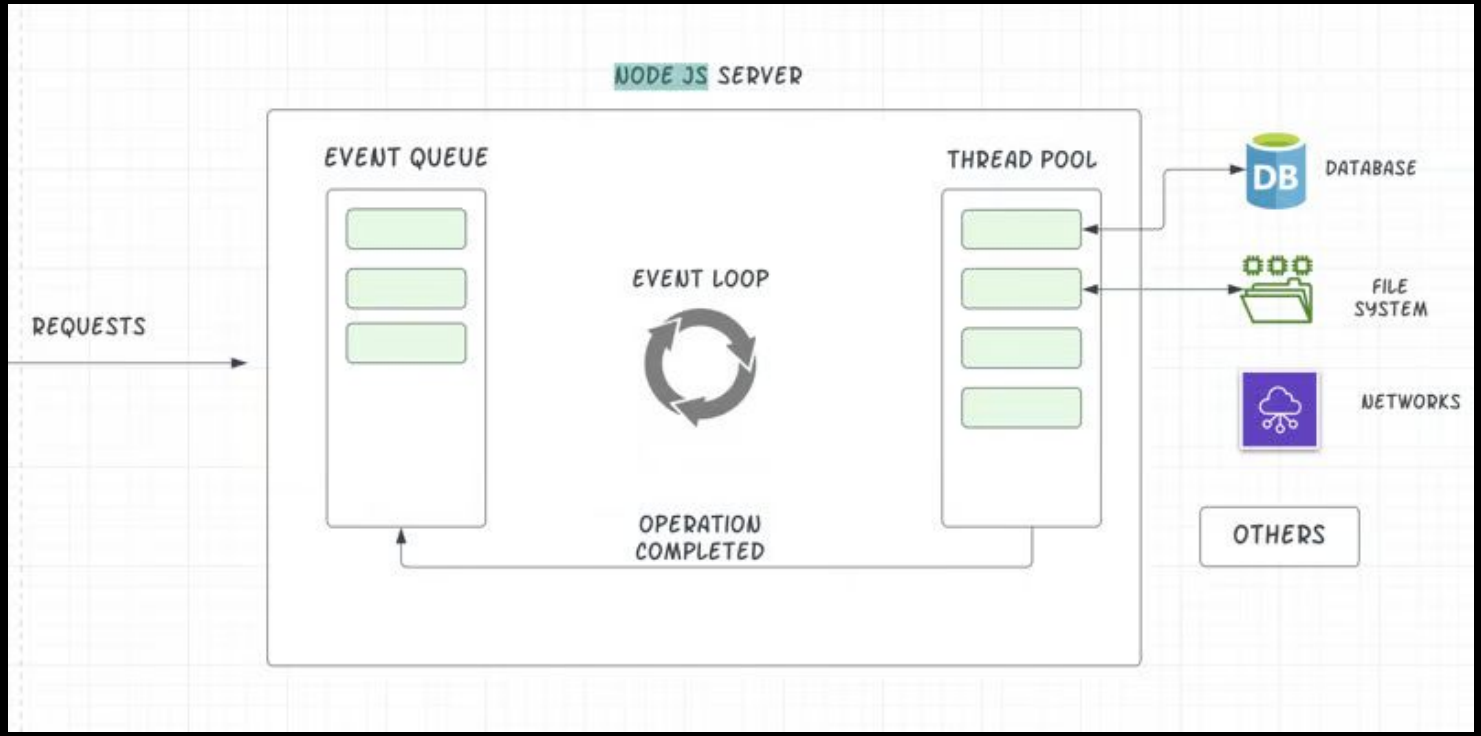
1. How DNS Works?
2. How Web Works?
3. What are Protocols?
4. Node Core Modules
5. Require Keyword
6. Creating first Node Server



4. Request & Response

1. Node Lifecycle & Event Loop
2. How to exit Event Loop
3. Understand Request Object
4. Sending Response
5. Routing Requests
6. Taking User Input
7. Redirecting Requests

4.1 Node Lifecycle & Event Loop



4.2 How to exit Event Loop

```
1  // Simple Node.js server
2  const http = require('http');
3
4  const server = http.createServer((req, res) => {
5    console.log(req);
6    process.exit(); // Stops event loop
7  });
8
9  const PORT = 3000;
10 server.listen(PORT, () => {
11   console.log(`Server running at http://localhost:${PORT}/`);
12 });
```

4.3 Understand Request Object

```
  [Symbol(kHeaders)]: {
    host: 'localhost:3000',
    connection: 'keep-alive',
    'cache-control': 'max-age=0',
    'sec-ch-ua': '"Chromium";v="128", "Not;A=Brand";v="24", "Google Chrome";v="128"',
    'sec-ch-ua-mobile': '?0',
    'sec-ch-ua-platform': '"macOS"',
    'upgrade-insecure-requests': '1',
    'user-agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (
36',
    accept: 'text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/web
nge;v=b3;q=0.7',
    'sec-fetch-site': 'none',
    'sec-fetch-mode': 'navigate',
    'sec-fetch-user': '?1',
    'sec-fetch-dest': 'document',
    'accept-encoding': 'gzip, deflate, br, zstd',
    'accept-language': 'en-US,en-IN;q=0.9,en;q=0.8,hi-IN;q=0.7,hi;q=0.6',
    cookie: 'token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6ImM3M2Y0MzNjLTFlYzYtNDU
qvUSrMLqbP1uy5bTjnJEQHXc1c'
  },
  [Symbol(kHeadersCount)]: 32,
  [Symbol(kTrailers)]: null,
  [Symbol(kTrailersCount)]: 0
}
```

https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers

4.3 Understand Request Object

```
// Simple NodeJS server
const http = require('http');

const server = http.createServer((req, res) => {
  console.log(req.url, req.method, req.headers);
});

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

Use the browser to access:

http://localhost:3000/

http://localhost:3000/products

4.4 Sending Response

```
1  // Simple NodeJS server
2  const http = require('http');
3
4  const server = http.createServer((req, res) => {
5    //res.setHeader('Content-Type', 'json');
6    res.setHeader('Content-Type', 'text/html');
7    res.write('<html>');
8    res.write('<head><title>Complete Coding</title></head>');
9    res.write('<body><h1>Like / Share / Subscribe</h1></body>');
10   res.write('</html>');
11   res.end();
12 });
13
14 const PORT = 3000;
15 server.listen(PORT, () => {
16   console.log(`Server running at http://localhost:${PORT}/`);
17 });
```