

# -- NODE.JS --

## What is NODE.JS

- Node.js is an open source server environment and it uses JavaScript on the server.
- A Node.js application runs within a single process, without generating a new thread for each request.
- Node.js includes asynchronous I/O primitives as a part of its standard library, which prevents JavaScript code from blocking and, in general, libraries in Node.js are developed using non-blocking paradigms.
- This makes blocking behaviour the exception instead of the rule.

## 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.
1. JavaScript Runtime: Node.js runs on the V8 JavaScript engine, which is also the core engine behind Google Chrome. However, unlike the browser context, Node.js executes JavaScript code outside of the browser.
  2. Single Process Model: A Node.js application operates within a single process, avoiding the need to create a new thread for every request. This design choice contributes to Node.js' performance.
  3. 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. When performing I/O operations (e.g., reading from the network, accessing databases, or the filesystem), Node.js doesn't waste CPU cycles waiting. Instead, it resumes operations when the response arrives.
  4. 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.
  5. JavaScript Everywhere: Frontend developers familiar with JavaScript can seamlessly transition to writing server-side code using Node.js. You don't need to learn a different language.
  6. ECMAScript Standards: Node.js supports the latest ECMAScript standards. You can choose the version you want to use, independent of users' browser updates.

## Importing the http module

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

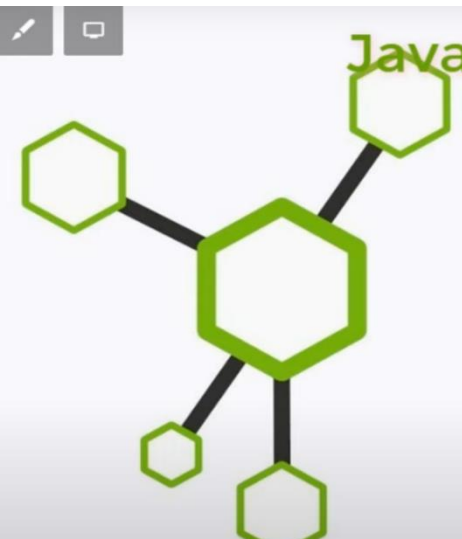
## Advantages of Node.JS

- **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.

## 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.



### JavaScript & Node are the Same?

- JavaScript and Node js Code Syntax is Same
- If you know JavaScript you can easily understand Node
- But both are not exactly the Same
- You can not connect Javascript to DB
- Node can connect with DB
- Node js run on the server side
- JavaScript is run on the **browser**

## Core Modules

Node.js includes several core modules for essential functionality. Some commonly used ones are:

1. **fs (File System)**: Read/write files and directories.
2. **http**: Create HTTP servers and clients.
3. **path**: Manipulate file paths.
4. **events**: Implement custom event handling.

## Datatypes in Node.js

Node.js contains various types of data types similar to JavaScript.

1. Boolean
2. Undefined
3. Null
4. String
5. Number

### Require http module

```
let http = require("http");
```

### Create server

```
http.createServer(function (req, res) {  
  // Send the HTTP header  
  // HTTP Status: 200 : OK  
  // Content Type: text/plain  
  res.writeHead(200, { 'Content-Type': 'text/plain' });  
  // Send the response body as "This is the example  
  // of node.js web based application"  
  res.end('This is the example of node.js web-based application \n');  
  // Console will display the message  
}).listen(5000,  
  () => console.log('Server running at http://127.0.0.1:5000/'));
```

## Common Use Cases

1. Building RESTful APIs
2. Real-time applications using WebSockets
3. Microservices architecture
4. Serverless functions with AWS Lambda

## Node.js Assert module

Assert module in Node.js provides a bunch of facilities that are useful for the assertion of the function.

The assert module provides a set of assertion functions for verifying invariants. If the condition is true it will output

nothing else an assertion error is given by the console.

```
npm install assert
```

```
const assert = require("assert");
```

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

```
let x = 4;
```

```
let y = 5;
```

```
try {
```

```
  // Checking condition
```

```
  assert(x == y);
```

```
}
```

```
catch {
```

```
  // Error output
```

```
  console.log(` ${x} is not equal to ${y}`);
```

```
}
```

---

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

```
const PORT = 3000 ;
```

```
http.createServer((req, res) => {
```

```
  res.write("Hello World!") ;
```

```
  // Signals the server that all of the response headers and body have been sent
```

```
  res.end()
```

```
}).listen(PORT, ()=> {
```

```
  console.log(`Server running at http://localhost:${PORT}/`);
```

```
});
```

---

*To make requests via the HTTP module `http.request()` method is used.*

*Syntax:*

```
http.request(options[, callback])
```

*Example :* In this example, we will see to make requests via the HTTP module `http.request()` method.

*An HTTP request is a message sent from a client to a server in a specific format. HTTP governs the structure and language of requests and responses between clients and servers. HTTP requests are divided into three sections:*

- 1. Request line: Indicates the method being used and the version of the HTTP protocol*
- 2. Method: Indicates the specific desired action to be performed on a given resource*
- 3. Response: The output sent to the client*

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

```
let options = {
```

```
  host: 'www.geeksforgeeks.org',
```

```
    path: '/courses',  
    method: 'GET'  
  };  
  http.request(options, (res) => {  
    console.log(`STATUS: ${res.statusCode}`)  
  }).end();
```

## Node.js http.ClientRequest.connection Property

*The `http.ClientRequest.connection` is an inbuilt application programming interface of class `ClientRequest` within the `HTTP` module which is used to get the reference of underlying client request socket.*

*Syntax: `const request.connection`*

Parameters: It does not accept any argument as the parameter.

Return Value: It does not return any value.

Node.js mostly used for creating APIs (bcoz JS cant)  
so that we can connect both with Web and Mobile app  
Node.js is superfast for APIs.

### REPL JS

stands for READ (read user input), Eval (evaluate user input), Print (print | output result), Loop (return and wait for new input)