**Mostly About Concepts…**

**Creator’s Note**

* These aren’t notes. These are just concepts which I had problems to grasp.
* Just grab a snack and go through them once as revision.
* These are out of order as they were added when I encountered such doubts.
* This file is under construction and will update whenever I get another doubt.
* **Giriraj**

**Grab a Snack and start reading…**

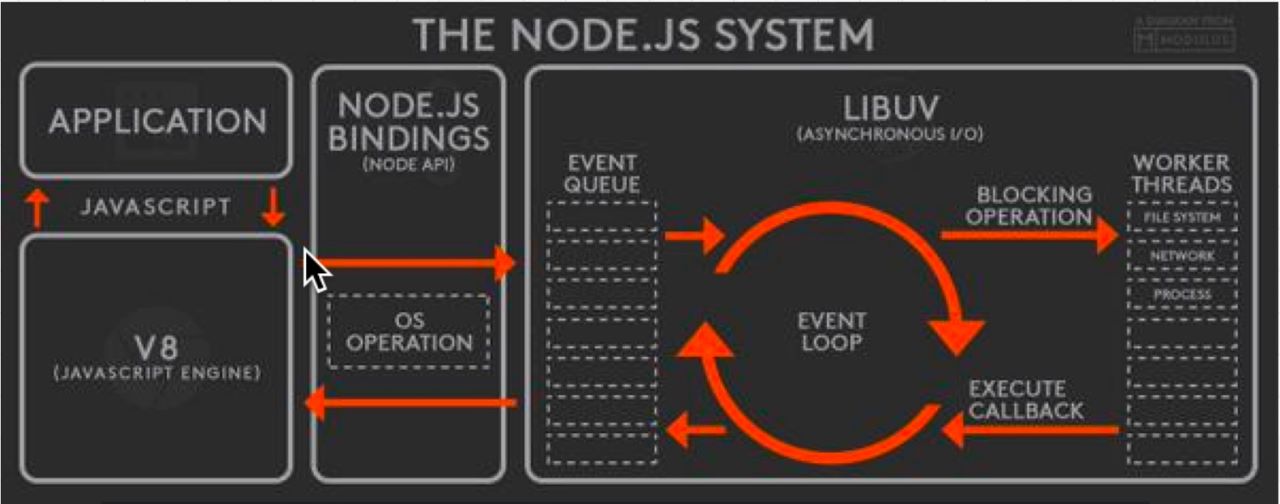
1. **Is Node a framework?**

Well, logically speaking it **isn’t** a framework. Instead, Node is a JS Runtime which means it helps in running JS code asynchronously with different tools.

**Node = V8 Engine** (runs JS Code) **+ Libuv Library** (provide other tools like io-functions, call stack, callbacks, worker threads, event loop, event queue)

Node establish connections between V8 Engine and Libuv library.

It can be said a framework but it is more accurate to call it a JS Runtime as It just establish connections but all other code, we have to do by ourselves.

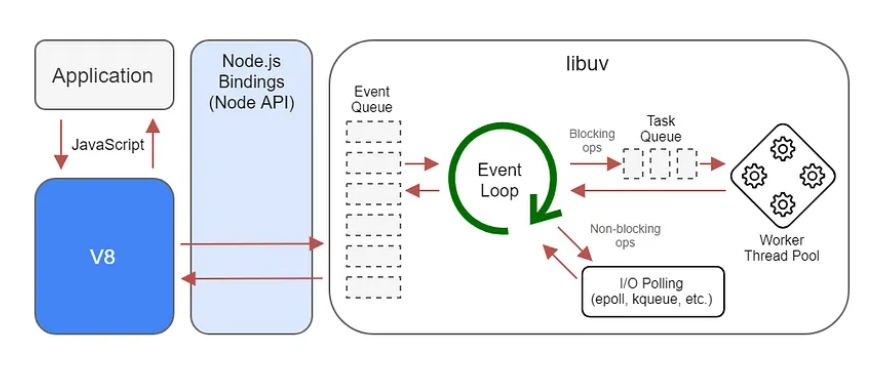


**Browsers are also JS Runtimes** as they also use some sort of JS Engine and Browser tools which do functionality like Libuv Library (not all but some) to run JS effective and asynchronously.

1. **Is Node JS Single Threaded?**

Well, on paper you can say that NodeJS is single threaded language but it isn’t completely true. Though, JS runs on single thread but for I/O and other features, Node uses LibUV Library and OS which is built-in low-level languages like C and these indeed uses multiple threads and Thread pool. Hence, Ultimately Node also uses multi threads but run JS it uses only one main single thread.

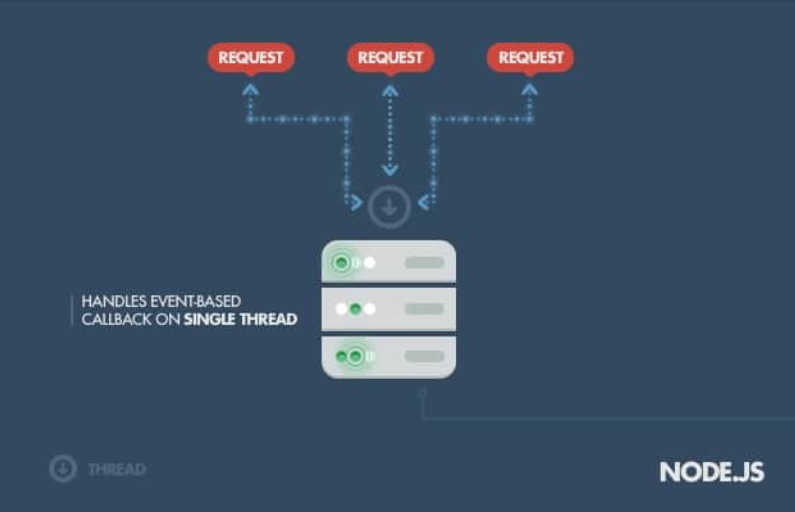
1. **Flow of NodeJS**

****

Here, you can see how event loop and LibUV’s other tools will help Node to do asynchronous tasks. As all events were sent to Event Queue from there it will be distributed accordingly whether to thread pool to perform tasks or to OS to do I/O operations etc. and after completion of task they will be placed back into queue from there they will move back to call stack and perform further execution.

1. **How Node is different from PHP and Python?**

PHP and Python are also single threaded languages, now a days, Python have inbuilt libraries of Threads. But comparing to PHP, PHP uses **Apache Server** (A Software Web Server which provides HTTP Server) and **various number of threads** and each thread will handle clients and thus, **have limited capabilities and each client makes a non-blocking call** and therefore it’s way of handling even with help of threads sometimes become inefficient. On the other side, Node handles these events in completely different way as it uses non-blocking approach and pass all the events in event queue of LibUV library where LibUV handles these events in efficient way using Thread Pool and Hence, it achieves Asynchronous behaviour.

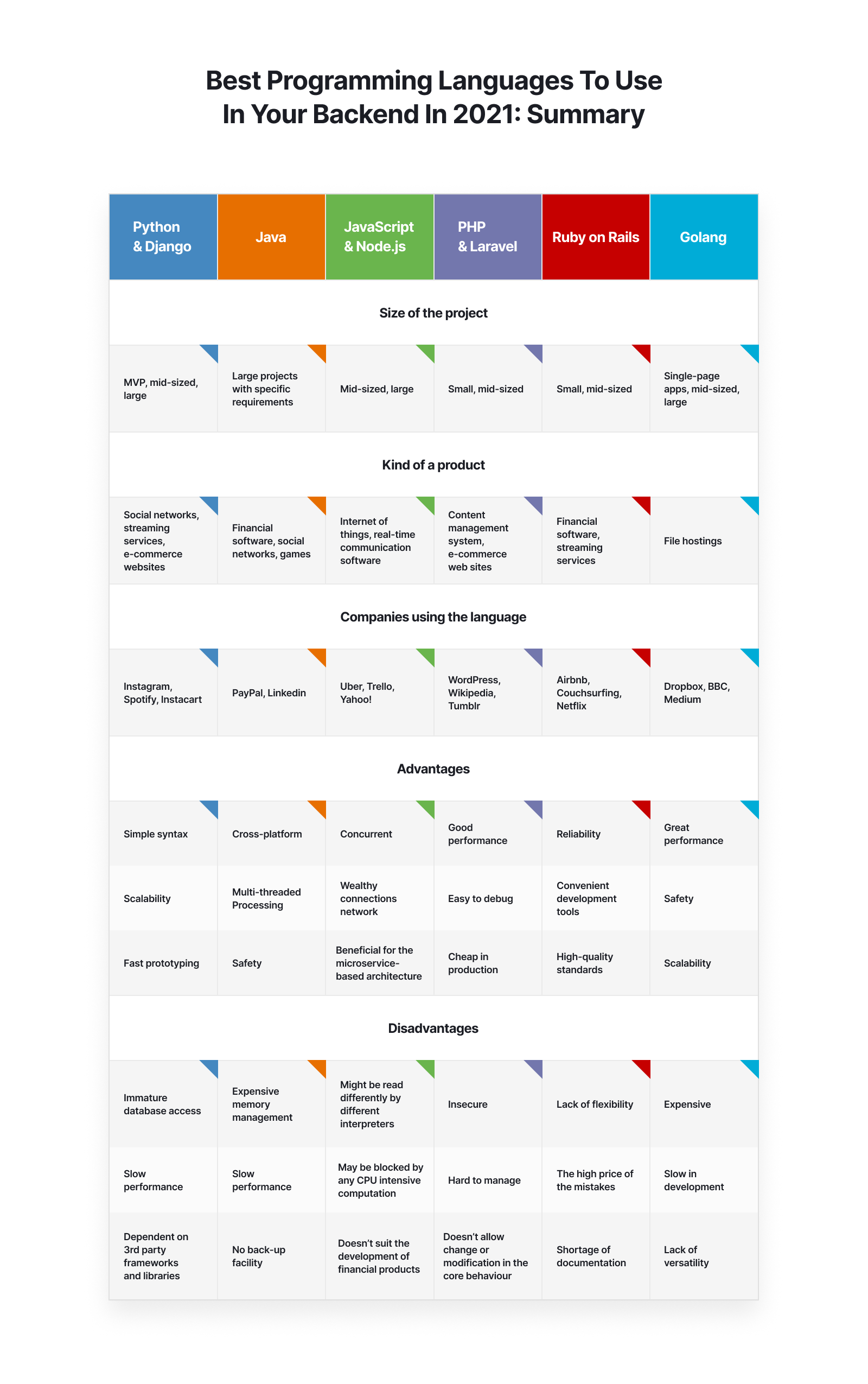
 

1. **What is Node JS best at? How can we choose when to choose Node or when any other languages?**

Node is best at distributing work among other components of hardware like Disks and other Networking Components and in meantime CPU performs another task.

* **Node & JS:** Normal Website, API Rich Website, Normal Server-Side Interactions, Real Time or Chatting websites which uses sockets or other communication methods.
* **Python & Django:** Fast & Performance Rich, High Computation, Data Intensive, AI/ML Based Sites – Python has computation rich libraries like numpy and pandas.
* **C++ & Rust:** Performance Rich and Low-Level Control.
* **C# & Java & ASP.NET:** Performance Rich, Enterprise Application, Integration with Existing Systems (C# - Best)

Below is the diagram showing Languages and their Features and Properties.



1. **.mjs extension**

By default and to be backward compatible, nodeJS follows commonJS not ECMAScript module system therefore to use module system we have to use .mjs extension.

Or we can install npm and use latest version of node to use ECMAScript module system.

1. **Use of index.JS in NodeJS**

If we have to import module, we have to give path of that module and then we can use functions exported by that module.

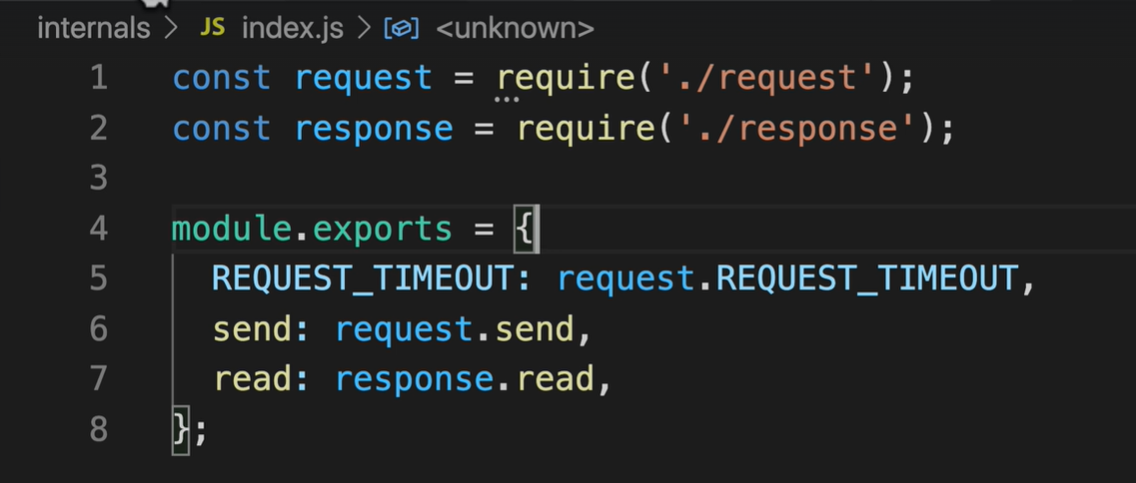
But suppose, there are many different modules in a particular folder and we have to export all or some required functions from that folder’s modules, then instead of giving path of each file.

We can give path of folder and create a index.JS file in that folder that will work as a trigger point from where all the required functions from modules will be exported via index.JS file. Thus, from a single source.

*All Modules in a Folder* ***=>*** *List all important functions exports into index.JS File* ***=>***

*Give path of folder in required file* ***=>*** *Now you can access all the functions mentioned in index.JS file.*

**Use case of index.JS**

**

**So, is it Good or Bad?**

Well, it depends adding index.JS file increases the complexity of accessing the modules/functions.

Most of the times we should avoid adding index.JS file for smaller projects or for mid-range projects if possible and we don’t want to add complexity.

But if you are working on a very large project and have too many modules and required and common export file then you can use.

**But still, it is a good practice to avoid adding index.JS file.**

1. **Some Important Files and Terms in NodeJS**

**NPM:** npm is software tool and command line program that serves as package manager for node.js. It isn’t a feature rather a utility which is installed side by side to node and helps to install, run or start script and download packages.

**NODEMON:** nodemon is another utility software that make development process easy as it automatically restarts the server whenever it detects modification in node files. Thus, eliminates the need to stop the server and then start it again.

**NODEMON and NODE are completely different. Latar is package manager and command line tool used to start or run a file or install a package, and on other side former just tracks changes in server files and restart it whenever changes detected.**

|  |  |  |
| --- | --- | --- |
| **node\_modules Folder** | **package.json** | **package-lock.json** |
| House of installed packages. All the packages required for your product will be download here. | Provides metadata about your product and its dependencies. It contains information like the project's name, version, author, description, and more. | This provides more information about the packages installed for project and the dependencies of the installed backages and so on. The specific versions of the packages and their dependencies are recorded. This helps maintain consistency across different installations or environments. |
| Generated by npm. | Manually created by commands like npm init. | Generated by npm. |
| Should not edited manually.  Should not uploaded to GitHub as it must be downloaded only by npm to maintain consistency. | Can be edited manually.  It should be uploaded as it gives the information about the packages installed, and another person use this file to download particular version of package according to this file. | Generally, not edited.  It can be uploaded or can be avoided as ultimately npm will regenerate it so it doesn’t matter that much. |

1. **Semantic Versioning – Major.Minor.Patch**
2. **~Major.Minor.Patch** – Same **Major.Minor** and greater than equal to **patch**.
3. **^Major.Minor.Patch** – Represents that All versions which are part of same **Major** and greater than equal to **Minor.Patch.**

**Exception –** In case of major version as 0. Minor is treated as Major and Patch is treated as Minor therefore **^0.Major.Minor** [ just for sake of understanding]. All versions which are part of **Major** and greater than equal to **Minor.**

There are other various other ways too. You can check out main purpose of this to show you that there are semantic versions of each package of node and we should choose according to dependencies of package build on.

Although they are backward compatible but, not all are.

1. **A Note on API & Protocol Methods and End Points**

An API is a very broad term if we do not specify it correctly, Broadly API (Application Programming Interface) is an interface or way to interact to the server/database or application.

Take it in this way, A programmer made a server or an application. Now, he wants you to use it or you are requesting him that you want to use that server/application. Now, he can’t give you the whole code or direct access to its code as you can change and it affects the system privacy. So, he just gives you access to some functionality by giving you some functions, or tools via which you can request service from that server and not affecting the whole code.

This is an **API**; this is how system works.

Suppose, taking the web perspective

**API = HTTP/HTTPS Methods like GET, POST, PUT + End Points (which is described by programmer).**

An API uses HTTP methods to interact with its defined routes or endpoints. HTTP methods provide the means for clients (such as web browsers, applications, or other servers) to perform specific actions on these routes, accessing the functionalities provided by the API.

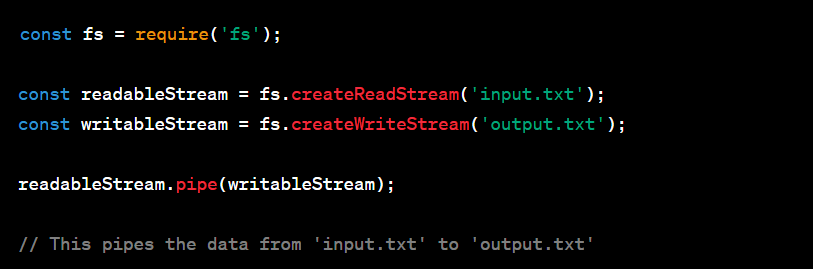
In essence, **when you make a request to an API, you're specifying an HTTP method along with the endpoint (URL) to indicate what operation you want to perform on that specific route**. The API then processes the request based on the combination of the HTTP method and the defined endpoint.

**Here, HTTP is a protocol used.**

Different protocol provides different way to handle data, HTTP/HTTPS is the protocol used over web to communicate to servers. Now, HTTP/HTTPS provides you different methods for communication and you uses those methods to communicate to each route or endpoint and request for service or functionality, and this whole system combining requesting a service from endpoint using protocol methods termed as an **API Call** and this whole system or logic to providing you these functionality is known as an **API**.

1. **Streams in Node**

Streams in Node.js are a powerful concept that allows you to efficiently work with data that can be read from or written to over time, without loading the entire dataset into memory. Streams provide an abstraction for working with data in chunks, making them especially useful when dealing with large files, network communication, or data processing.



Pipe here refers to the pipeline.

The pipe method is used to connect a readable stream to a writable stream. It takes the data that is read from the readable stream and automatically writes it to the writable stream. Essentially, it creates a data flow pipeline between the two streams.

1. **Significance of PORTS in NODE and in WEB.**

A single machine or server can be running more than one application, but a single server has only one IP then how it is possible? Well, this is possible because of port number assigned to a particular application is different.

Suppose, you have a big server (Hardware Computer) with some IP but it is also running different services/applications like INPUT/OUTPUT or PRINTER/HARDWARE or Different Functionality then to run all of them simultaneously and efficiently we can assign each service to a different port number. A particular type of request can go to a particular application by recognizing/matching port number.

Suppose, you own a company and you setup your expensive hardware computer server which have some IP. Please note that IP addresses is assigned to physical machines connected to internet or network.

You have 3 features/applications named A, B, C. Now, you want to run all these services on same server which you setup and run simultaneously, you can achieve this by assigning different ports to different applications.

Thus, 3 features running on same server simultaneously but on different ports.

Also, all the ports present in computer/server are dedicated and have purpose like 80 for http and 443 for https etc.

**Ports Number Information**

1. **Well-Known Ports – 0 to 1023 for standard devices and protocols.**
2. **Registered Ports – 1024 to 49151 for custom applications and services.**
3. **Dynamic/Private Ports – 49152 to 65535 for temporary uses and client side connections.**