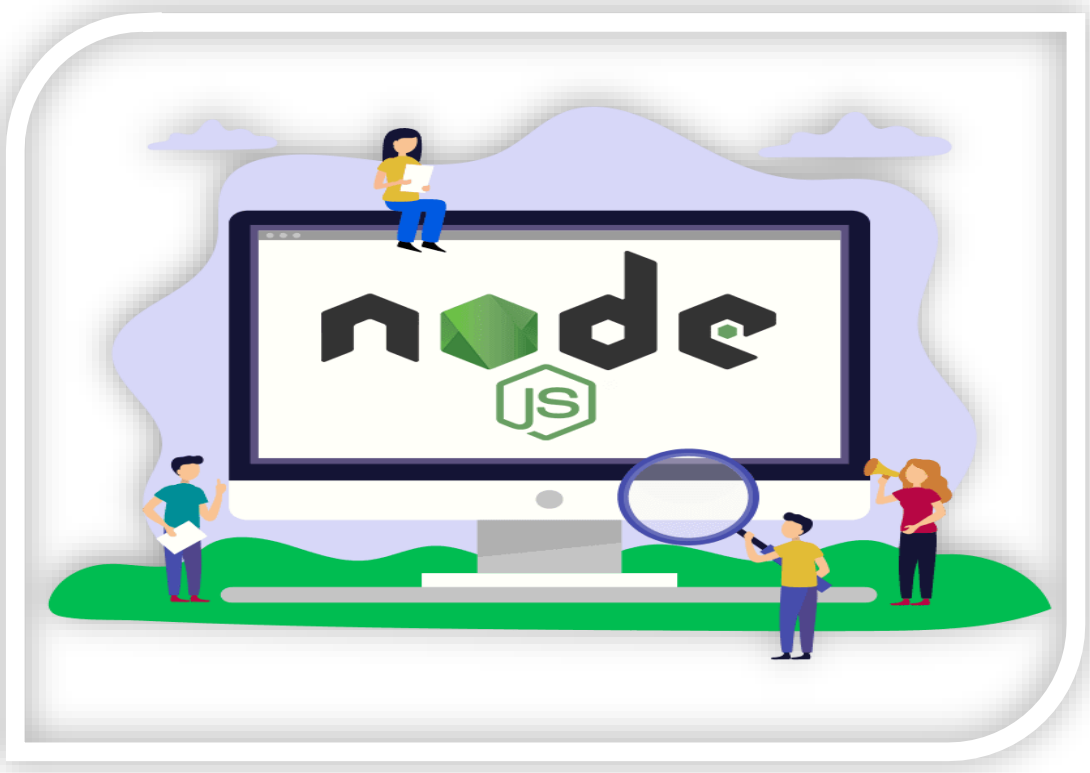
Node JS CASE STUDY



Cognitive Convergence
http://www.cognitiveconvergence.com
+1 4242530744
shahzad@cognitiveconvergence.com

Cognitive Convergence is Subject Matter Expert in Office 365, Dynamics 365, SharePoint, Project Server, Artificial Intelligence Practices, Machine Learning practices, Power Platform: Power Apps-Power BI-Power Automate-Power Virtual Agents.

Our Node Js Development, Consulting, Add-in Development, Customization, Integration services and solutions are dedicated to help companies maximize their business performance, overcoming market challenges, achieving profitability, and providing best customer service.

Contents

OBJECTIVE

Most important thing to consider is the choice of the best technology platform for the app development process. With the constant change and demand in the technology Node JS has been getting great press for being used to build real-time web applications and fast networking tools that help big web sites run and scale.

The objective of this document is to describe Node JS and its different modules. We will elaborate on the use of Node JS in various industries and how we will provide you more advance, secure web applications based on your requirement.

FOUNDATION OF NODE JS

The World Wide Web started around 30 years back. JavaScript is a programming language that was created at Netscape, Netscape Navigator, as a scripting tool to manipulate web pages inside their browser.

Part of the business model of Netscape was to sell Web Servers, which included an environment called Netscape LiveWire that could create dynamic pages using server-side JavaScript. Unfortunately, Netscape LiveWire wasn't very successful and server-side JavaScript wasn't popularized until recently, by the introduction of Node JS.

JavaScript was born about 25 years ago and about the same goes for PHP (26 years). Node JS, on the other hand, is only 11 years old. Despite its relatively short run, Node JS has done wonders for developer organizations around the world.

- In 2009 Node JS is born and the first form of “npm” is created
- In 2010 “Express” and “Socket.io” born
- In 2011 large companies start adopting Node JS increases and “hapi” is born
- In 2013 First big blogging platform start using Node JS (Ghost) and Koa is born
- In 2014 io.js
- In 2015 Node JS foundation born, IO.js merged into Node JS, private modules were introduced, and Node JS 4 introduced
- In 2016 Yarn born
- In 2017 focus on security and introduced HTTP/2
- After 2017 and till now many versions of Node JS introduced Node JS is the latest one.

An Introduction to Node JS

Node JS is a server-side platform built on Google Chrome's JavaScript Engine (V8 Engine). It was developed by Ryan Dahl in 2009 and its latest version is v0.10.36.

Definition according to official documentation of Node JS is:

"Node JS is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. Node JS uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices."

Node JS is an open source, cross-platform runtime environment for developing server-side and networking applications. Node JS applications are written in JavaScript and can be run within the Node JS runtime on OS X, Microsoft Windows, and Linux. Node JS also provides a rich library of various JavaScript modules which simplifies the development of web applications using Node JS to a great extent.

LEVERAGE THE USE OF NODE JS

Following are the areas where Node JS is proving itself as a perfect technology partner.

- I/O bound Applications
- Data Streaming Applications
- IoT applications
- Data Intensive Real-time Applications (DIRT)
- JSON APIs based Applications
- Single Page Applications

NODE JS SERVER ARCHITECTURE

Node JS uses the “Single Threaded Event Loop” architecture to handle multiple concurrent clients. Node JS Processing Model is based on the JavaScript Event-based model along with the JavaScript callback mechanism.

Requests: Depending on the actions that a user needs to perform, the requests to the server can be either blocking (complex) or non-blocking (simple).

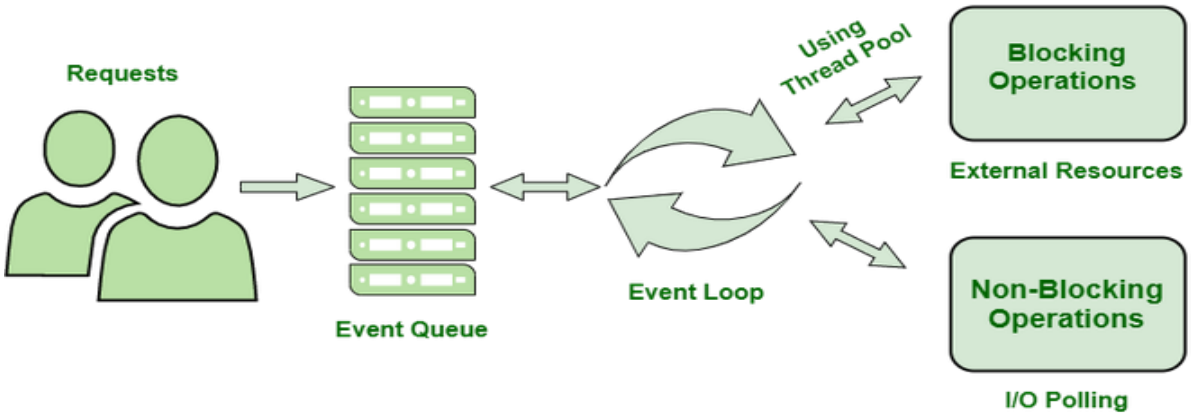
Node.js Server: The Node.js server accepts user requests, processes them, and returns results to the users.

Event Queue: The main use of Event Queue is to store the incoming client requests and pass them sequentially to the Event Loop.

Thread Pool: The Thread pool in a Node.js server contains the threads that are available for performing operations required to process requests.

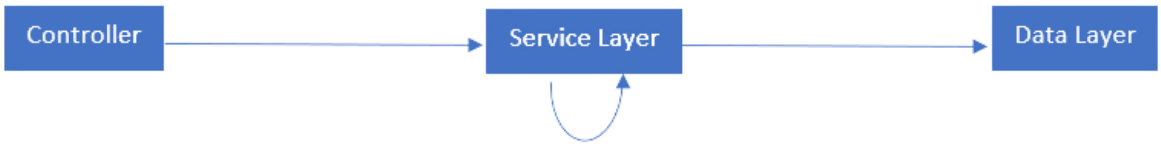
Event Loop: Event Loop receives requests from the Event Queue and sends out the responses to the clients.

External Resources: In order to handle blocking client requests, external resources are used. They can be of any type (computation, storage, etc).



Node JS work on three-layer Architectural Model which is shown in below figure

- **Controller**
  - API routes and endpoints
- **Service layer**
  - In this layer we write own all business logics
- **Data access layer**
  - In this layer we integrate/interact with a database



NPM: THE NODE PACKAGE MANAGER

When discussing Node.js, one thing that definitely should not be omitted is built-in support for package management using NPM, a tool that comes by default with every Node.js installation. The idea of NPM modules is quite similar to that of Ruby Gems: a set of publicly available, reusable components, available through easy installation via an online repository, with version and dependency management.

A full list of packaged modules can be found on the npm website, or accessed using the npm CLI tool that automatically gets installed with Node.js. The module ecosystem is open to all, and anyone can publish their own module that will be listed in the npm repository.

Some of the most useful npm modules today are:

Express

Express.js—or simply Express—a Sinatra-inspired web development framework for Node.js, and the de-facto standard for the majority of Node.js applications out there today.

Hapi

a very modular and simple to use configuration-centric framework for building web and services applications

connect

Connect is an extensible HTTP server framework for Node.js, providing a collection of high performance “plugins” known as middleware; serves as a base foundation for Express.

socket.io and sockjs

Server-side component of the two most common websockets components out there today.

pug (formerly Jade)

One of the popular templating engines, inspired by HAML, a default in Express.js.

mongodb and mongojs

MongoDB wrappers to provide the API for MongoDB object databases in Node.js.

lodash (underscore, lazy.js)

The JavaScript utility belt. Underscore initiated the game, but got overthrown by one of its two counterparts, mainly due to better performance and modular implementation.

Forever

Probably the most common utility for ensuring that a given node script runs continuously. Keeps your Node.js process up in production in the face of any unexpected failures.

Bluebird

A full featured Promises/A+ implementation with exceptionally good performance

Moment

A JavaScript date library for parsing, validating, manipulating, and formatting dates.

The list goes on. There are tons of really useful packages out t here, available to all.

BUILDING CUSTOM MIDDLEWARE

Middleware functions are functions that have access to the request object (req), response object (res), and the next middleware function in the application's request-response cycle. The next middleware function is commonly denoted by a variable named next.

Advantages of using Middleware:

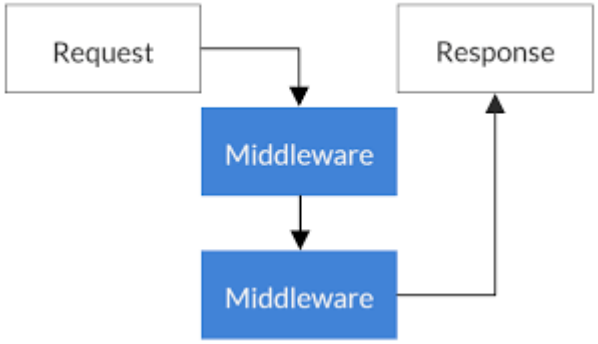
Some advantages of using middleware are:

- It can process request objects multiple times before the server works for that request.
- It can be used to add logging and authentication functionality.
- It improves client-side rendering performance.
- It is used for setting some specific HTTP headers.
- It helps for Optimization and better performance.

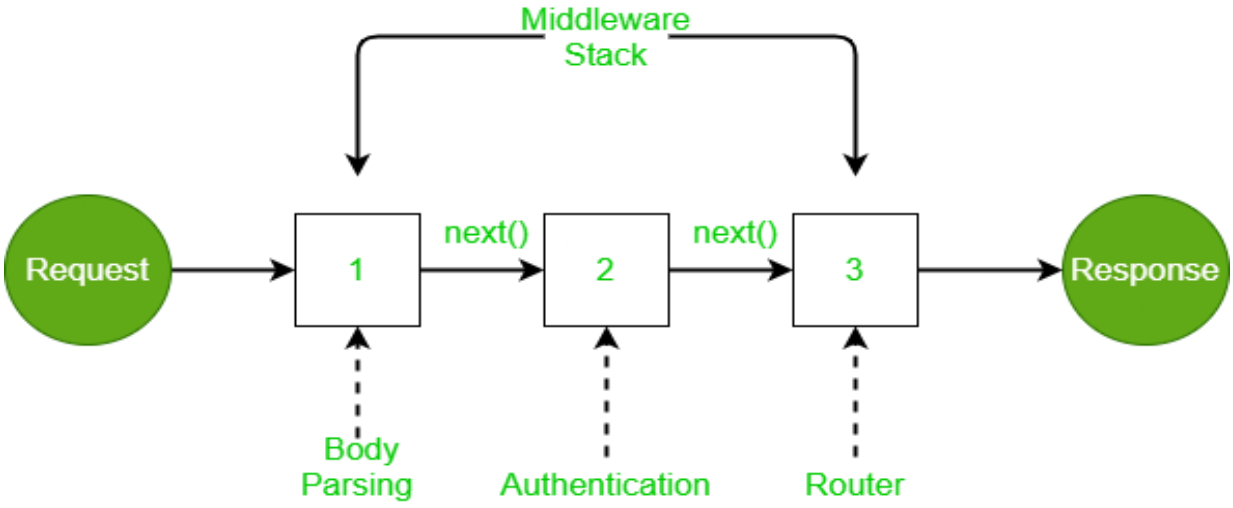
Creating Different Custom Express Middleware

Following are the different custom express middleware:

- Application-level middleware app.use
- Router level middleware router.use
- Built-in middleware express.static,express.json,express.urlencoded
- Error handling middleware app.use(err,req,res,next)



Middleware is software that is assembled into an app pipeline to handle requests and responses. Node JS provides a rich set of built-in middleware components, but in some scenarios, you might want to write a custom middleware. This topic describes how to write convention-based middleware.



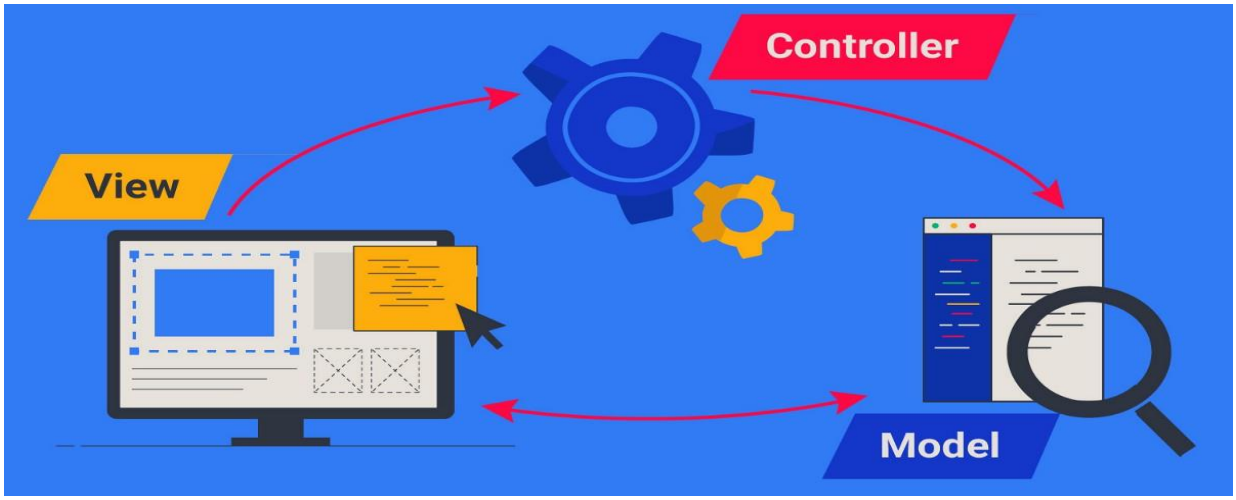
TECHNOLOGIES STACK

MVC frameworks

These frameworks offer a valuable design pattern that splits application logic into three essential parts: models, views, and controllers. Separating development concerns makes it extremely simple to maintain and scale the app. Express.js is a classic example of an MVC framework.

Full-stack MVC frameworks

Full-Stack MVC frameworks offer a great deal with scaffolding, libraries, template engines, and a range of other development capabilities. Additionally, they can take care of both frontend and backend development of applications.



REST API frameworks

Node.js frameworks have a firm reference to building apps faster with a ready and well-known REST API built experience. This means you don't have to worry about the architectural styles of network applications. Nearly all these frameworks provide a ready programming interface, which saves much time in development to build apps requiring an internet connection.

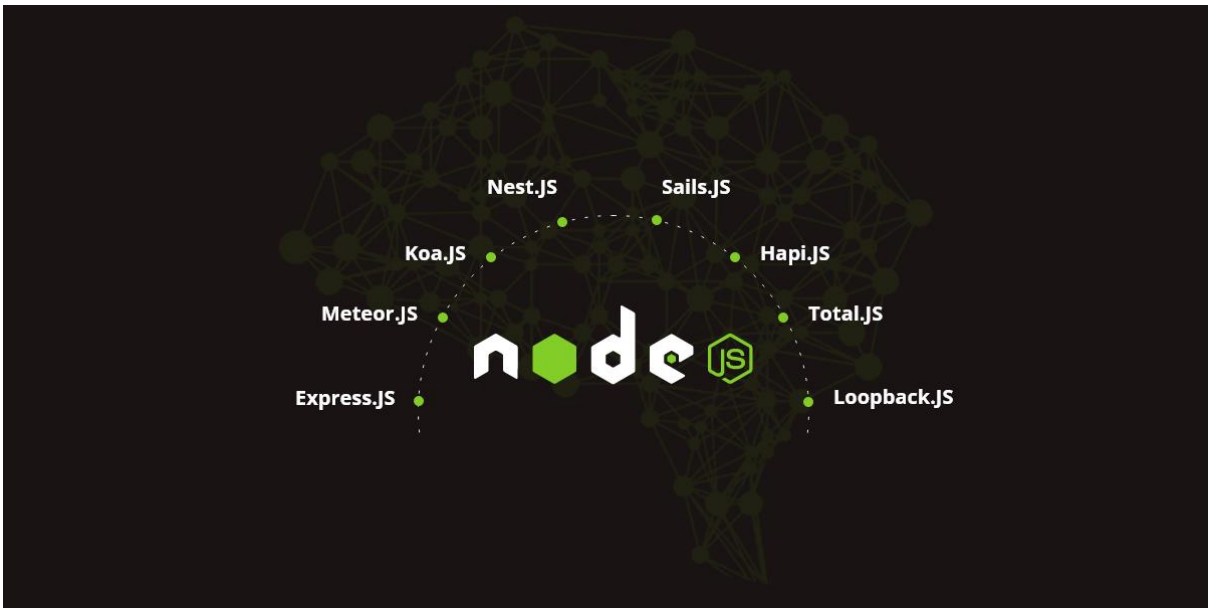
NODE JS FRAMEWORKS

Over the year many experienced JavaScript developers have built some extraordinary frameworks to get started with Node JS.

In 2022, the demand for web applications has increased with more and more businesses employing them to deliver a better user experience. Looking at this rising demand, developers are continuously testing different frameworks that offer advanced features to build web applications.

There are many frameworks of Node JS. We work on some of the famous frameworks, such as:

- Express.JS
- Socket.IO
- Nest.JS
- Koa.JS



Express.JS:

Express.js, also known as Express, is known to be one of the best Node JS frameworks. It does not necessitate a steeper learning curve; instead, a fundamental awareness of the Node JS environment and programming skills are all that is required. Its asynchronous, quick, and robust architecture works nicely with Node. Because of its high-speed I/O operations and single-threaded nature, Express is a basic requirement for apps built using the Node JS framework. A web or mobile user will have a far more delightful experience if client and server communication is improved. Apps produced with Express are used by companies like Twitter, Uber, Accenture, and IBM, among others. Because the framework has practically immediate grounds for API development, you can design web apps faster. Because of its sophisticated routing, templating, security, and error handling features, it can be used for any enterprise-grade or browser-based app.

Koa.JS

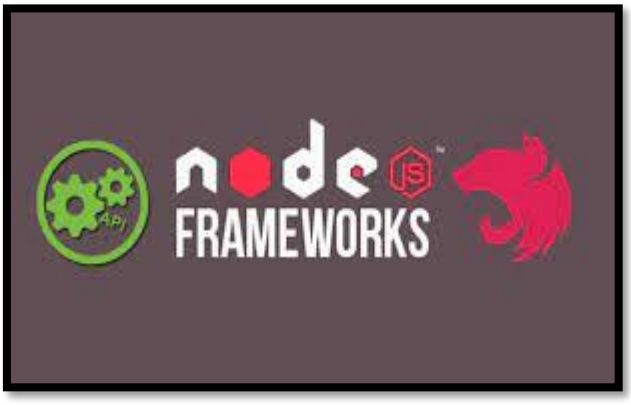
Koa is one of the most popular Node JS frameworks for constructing various online services, sometimes known as APIs. Building these APIs is a lot of fun and easy with Koa because it has a stack-like way for dealing with HTTP middleware.

It also constructively normalizes Node defects. With this framework, you can maintain several versions of material given to users with the same URL, such as translating a page, personalizing content in eCommerce websites having different formats for photos, and much more.

Another advantage of Koa is that it is very similar to Express. While writing code, you can still experience the flexibility of Express with greater freedom and fewer complexity. This reduces the impact of mistakes across the entire application stack.

Nest.JS

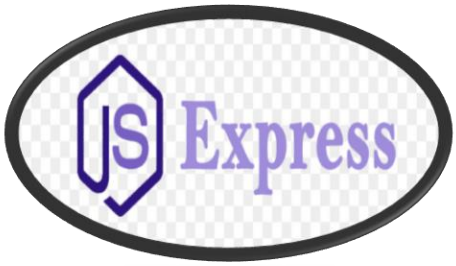
Nest.JS is a Node JS framework that allows you to create dynamic and scalable enterprise-grade apps while maintaining complete flexibility thanks to its vast libraries. It's a great example of a productivity booster on the backend. Nest.JS will not prevent you from using its libraries extensively, therefore you can utilize this framework to create multilayered enterprise applications.



The framework adheres to the clean-code architecture while preserving the code complexity of large-scale applications because it is compatible with a subset of JavaScript (i.e., generated with Typescript). Furthermore, nest.js integrates well with sister frameworks such as Fastify and Express. Nest is the best object-oriented and functional reactive programming combination (FRP).

NODE JS LIBRARIES

Express



Express is the most popular Node web framework and is the underlying library for a number of other popular Node web frameworks. It provides mechanisms to: Write handlers for requests with different HTTP verbs at different URL paths (routes). Fast, unopinionated, minimalist web framework for node.

Socket.IO

Socket.IO is an event-driven JavaScript library for real-time web applications. We require Socket.io to configure the functionality, even if all these real-time apps are connected and have the important feature of real-time updating of responses and requests.

Socket.io is a JavaScript toolkit for developing real-time programs and communicating bidirectionally between web clients and servers. You can use this library framework to create applications that require WebSocket development. Chat applications like WhatsApp, for example, run continually for live updates and refresh the background process for new messages or updates. It also provides real-time analytics with less code.

Socket.io is used by over a thousand firms, including Bepro, Barogo, and Patreon.

Body-Parser

Body-Parser is a node js library which is use as a middleware. Parse incoming request bodies in a middleware before your handlers, available under the req.body property.

CORS

CORS is a node.js package for providing a Connect/Express middleware that can be used to enable CORS with various options.

CORS stands for Cross-Origin Resource Sharing. It allows us to relax the security applied to an API. This is done by bypassing the Access-Control-Allow-Origin headers, which specify which origins can access the API. In other words, CORS is a browser security feature that restricts cross-origin HTTP requests with other servers and specifies which domains access your resources.

Multer

Multer is a node. js middleware for handling multipart/form-data, which is primarily used for uploading files. It is written on top of busboy for maximum efficiency.

NOTE: Multer will not process any form which is not multipart (multipart/form-data).

Morgan

Morgan is a Node.js and Express middleware to log HTTP requests and errors and simplifies the process. Create a new Morgan logger middleware function using the given format and options. The **Format** argument may be a string of a predefined name (see below for the names), a string of a format string, or a function that will produce a log entry. The format function will be called with three arguments tokens, req, and res, where tokens is an object with all defined tokens, req is the HTTP request and res is the HTTP response. The function is expected to return a string that will be the log line, or undefined / null to skip logging.

HTTP-Errors

Create HTTP errors for Express, Koa, Connect, etc. with ease. Create a new error object with the given message msg. The error object inherits from createError.HttpError.

Error Properties

- **Expose** - can be used to signal if message should be sent to the client, defaulting to false when status >= 500
- **Headers** - can be an object of header names to values to be sent to the client, defaulting to undefined. When defined, the key names should all be lower-cased
- **Message** - the traditional error message, which should be kept short and all single line
- **Status** - the status code of the error, mirroring statusCode for general compatibility
- **StatusCode** - the status code of the error, defaulting to 500

Dotenv

Dotenv is a zero-dependency module that loads environment variables from a .env file into process.env. Storing configuration in the environment separate from code

NodeMailer

Nodemailer is a module for Node.js applications to allow easy as cake email sending.

- A single module with zero dependencies
- Heavy focus on security
- Unicode support to use any characters, including emoji
- Use HTML content, as well as plain text alternative
- Add Attachments to messages
- Embedded image attachments for HTML content
- Secure email delivery using TLS/STARTTLS
- Custom Plugin support for manipulating messages
- Sane OAuth2 authentication
- Proxies for SMTP connections

Mongoose

Mongoose acts as a front end to MongoDB, an open-source NoSQL database that uses a document-oriented data model. A "collection" of "documents" in a MongoDB database is analogous to a "table" of "rows" in a relational database. Mongoose is a MongoDB object modeling tool designed to work in an asynchronous environment. Mongoose supports both promises and callbacks.

Lodash

Lodash makes JavaScript easier by taking the hassle out of working with arrays, numbers, objects, strings, etc. Lodash's modular methods are great for:

- Iterating arrays, objects, & strings
- Manipulating & testing values
- Creating composite functions

Crypto-JS

CryptoJS is a growing collection of standard and secure cryptographic algorithms implemented in JavaScript using best practices and patterns. They are fast, and they have a consistent and simple interface.

Crypto is a module in Node JS which deals with an algorithm that performs data encryption and decryption. This is used for security purpose like user authentication storing the password in Database in the encrypted form. Crypto module provides set of classes like hash, HMAC, cipher, decipher, sign, and verify.

NODE JS SAAS FRAMEWORK

Gravityapp

Gravity is a React + Node.js SaaS application which functions as a boiler-plate template for these platforms. It includes a bunch of handy features for developers that include subscription payments to monetize their apps with a Stripe-based billing system. With the complete React library, developers can give create beautiful user interfaces. It also includes an out of the box user authentication system along with the ability to include teammates with a dedicated invitation system. With elegant pre-made email notifications, newsletters and updates can be sent out to customers within minutes. Managers have access to all users through a master dashboard that also includes relevant metrics. The form validation feature handles all associated forms to increase productivity. And with dedicated boilerplate T&Cs and privacy policy pages, users can easily manage their GDPR compliance efforts. Gravity also comes with a REST API that includes a token authentication system that is built with Express.js and MySQL.

Gravity.app having four different modules

- Gravity Server (the back-end Node application)

- Gravity Web (the front-end React client)
- Gravity Native (the front-end React Native client)
- Mission Control (the SaaS admin dashboard)

Modules

Gravity Server

Gravity Server is the backbone of both Gravity Web and Gravity Native and provides the server infrastructure required to run both client applications. All Gravity packages come with the server included.

Before working on server, we need to configure or install some things

- Install Node.js
- Create an empty database
- Register a Stripe account for payments (Gravity web only)
- Register a Mailgun account (for sending emails)
- Install Gravity

Gravity Web

The Gravity Web client is built using a beautiful, custom designed React user interface. This includes a full library of pre-built components that you can easily drop into your application, containing everything from views and tables to self-validating forms, modals, and notification banners.

Components are styled using Tailwind CSS and SCSS modules, you can use which you want to use at the component level.

Building your interface is as simple as creating a new view and adding some pre-built components. The following sections will give you an overview of how React is setup within Gravity, along with an explanation on how to use each of the components.

Gravity Native

Gravity Native is a complete mobile app template for both iOS and Android that enables you to build mobile applications faster than ever before, with no complicated native programming required.

The client is built using React Native and managed with Expo. This means you can build native mobile apps and deploy them to the app store using only JavaScript and without ever having to touch a single line of native code.

Oh, there's also live reloading as well, so you don't need to continually rebuild your app to test every change.

Mission Control

Mission Control is a centralized dashboard to manage your entire SaaS application and comes bundled with the web and power plans such as

- User Management
- Feedback
- Events
- Logs etc.

Nodewood

Nodewood is a JavaScript SaaS Starter Kit, designed to save you weeks or months of time when launching your next SaaS app. Like a general-purpose framework, Nodewood provides you with a collection of starter code that you can start customizing with your own business logic. However, the starter code that Nodewood provides is targeted towards SaaS apps specifically, and provides features like user auth/management, an app template and admin panel, subscriptions, etc. Compared to hiring someone to write all that code for you, you could save significant time and money in the race to launch your SaaS app.

First, make sure you have installed the requirements for running Nodewood,

Node.js

Node.js (opens new window) is the executable that runs JavaScript programs on your system. Since our package manager (Yarn) is a JavaScript program, we need to have Node.js installed to run it

Yarn

Yarn is an alternative package manager to NPM. For the most part, they are interchangeable, but Nodewood uses Yarn to set up workspaces (opens new window). This allows Nodewood to specify packages it uses in its own package.json file, and your app to specify packages it uses in its own package.json file.

Docker

Using Docker means you don't need to manually set up PostgreSQL and Nginx, and you don't need to have multiple terminal windows open to monitor the API and UI processes separately

Terminal

Terminals that will improve your development experience

Nodewood *Architecture*

- **Shared Code:** Since Nodewood uses JavaScript for both the server API and the client UI, this section will show you how to get the most out of sharing code between both.
- **Cascading Filesystem:** Learn how the app and wood folder interact, and how to override files in the wood folder.
- **Configuration:** Learn how to store and load configuration values, and how to override said values for testing.
- **Feature-based Development:** Nodewood breaks up code into "feature" folders to avoid having giant folders full of unrelated files. Learn how these features work.
- **Models:** Models in Nodewood aren't the ORM you're used to - in fact they have no database access at all! Why, and how to use them, is explained here.
- **Validators:** Validating your forms and API endpoints are critical ways to increase security and user experience. Nodewood has systems in place to make it much easier to do both, with a minimal amount of code.
- **Error Handling:** Learn how errors are handled in Nodewood, and how to create and throw your own so that they are caught by Nodewood's global systems correctly.

Rocketapp

Building a SaaS app from scratch takes time. You need to decide on what to build, pick a programming language, pick a web framework, pick a CSS framework. Save weeks of time implementing basic features like authentication and billing and start working on your product instantly.

- SaaS Landing page, React UI components, and API integration
- Secure Authentication and User Management
- Subscription payments with Stripe

Rocket is a production ready Node.js React boilerplate which can be customized to best suit your SaaS app.

The codebase comes with build and run scripts along with a documentation to help you launch in no time.

The code is structured in a modular fashion which makes it easy to understand and navigate. Also, adding new features and UI elements is much simpler by using the basic building blocks.

Features

- Node.js Rest API
- Multi-Tenancy with Teams
- Users & Authentication
- Social OAuth Authentication
- User Management
- Docker Support
- SSL Certificates
- NoSQL Database
- Error Handling

- Subscription Payments
- React UI
- Pre-Built Ui Components
- Responsive Templates
- Security & Permissions
- Code Quality

NODE JS ON AZURE

Introduction

Azure is a cloud platform designed to simplify the process of building modern applications. Whether you choose to host your applications entirely in Azure or extend your on-premises applications with Azure services, Azure helps you create applications that are scalable, reliable, and maintainable.

Azure supports the most popular programming languages in use today, including Python, JavaScript, Java, .NET and Go. With a comprehensive SDK library and extensive support in tools you already use like VS Code, Visual Studio, IntelliJ, and Eclipse, Azure is designed to take advantage of skills you already have and make you productive right away.

Azure services

Azure cloud-based services provide a huge variety of features. These services can be used independently or as a collection.

Top service types for JavaScript developers include:

- Hosting
- Authentication and authorization
- Containers
- VMs
- DatabasesStorage
- Search
- Cognitive services
- Metrics and logging
- DevOps

Why Node.js on Azure?

Powerful tooling

Easily deploy Node.js code to Azure from Visual Studio Code. If you prefer using command line tools (CLI), Azure has those, too. Both are built in the open, with tools, SDKs, and extensions on GitHub.

Host Node.js apps your way

Run full-stack apps directly on our managed Linux service, in serverless Functions, or in containers using Docker or Kubernetes. Host static sites directly from Azure Storage.

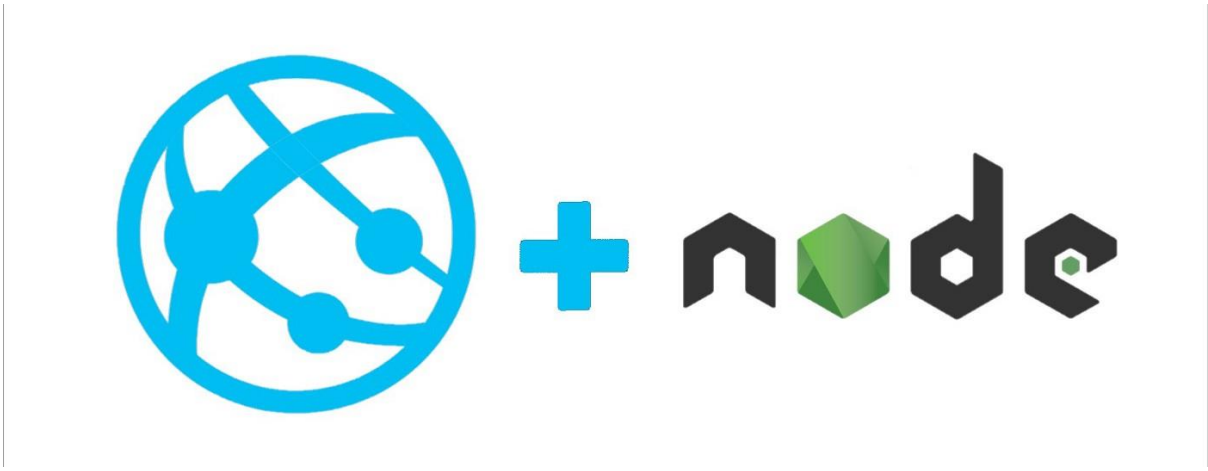
Easy to grow

Quickly add services and capabilities like MongoDB, PostgreSQL, and MySQL databases, performance monitoring, continuous integration, and secrets management as your needs expand.

Node.js web app in Azure

Setup initial environment such as

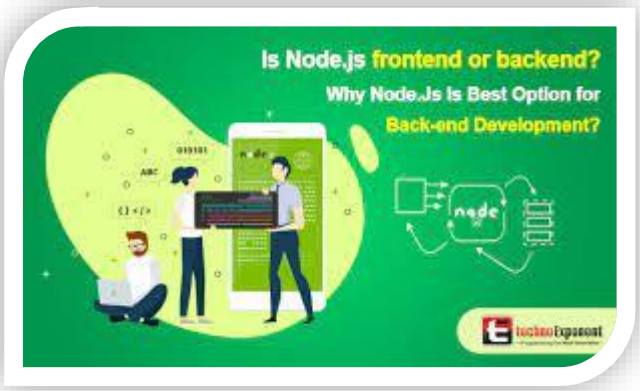
- Get an Azure account with an active subscription. Create an account for free.
- Install Node.js and npm. Run the command node --version to verify that Node.js is installed.
- Install Visual Studio Code.
- The Azure App Service extension for Visual Studio Code.



NODE JS - FRONTEND OR BACKEND?

There is a massive number of libraries built for Node.js. The idea of using Node.js for frontend is a natural extension of the various features that it provides. Let us understand why the use of Node.js enhances the developer experience.

1. JavaScript: The ability to use the same language for both frontend and backend makes it easier for developers.
2. Availability of Packages and Libraries: There are many libraries available to use in the frontend. For example, the 'moment' is a library that processes dates in the specified format.
3. The availability of the JS engine on the browser makes the integration of Node.js packages efficient. As discussed earlier, Node.js is a runtime environment, which lets you choose how to use, when to use and whether frontend or backend. Many believe that the node js backend framework or back-end scripting language only. One common language can be used for both front-end and back-end. Though the environment is entirely based on the V8 JavaScript engine, and the environment provides features of easily executing JavaScript documents that are not even executed in the web browser, so it is useful for the back-end environment as well.



One of the reasons Node.js frameworks are a popular choice for developers building a flexible and scalable backend is its event-driven, non-blocking nature. However, frontend developers will see these benefits of Node.js in their own work just as clearly.

Let's look at why Node.js works for both backend and frontend:

Reusability – JavaScript is a common language that's used to write both backend and frontend with the help of frameworks like Express.js and Meteor.js. Some popular stacks like MERN use Express.js as a backend (a Node.js framework). Multiple components can be reused between frontend and backend as well.

Productivity and developer efficiency – Thanks to a reduction in context-switching between multiple languages, a great deal of developer time can be saved. Using JavaScript for both backend and frontend results in increased efficiency, as many tools are common for both.

Huge community – A thriving online community factors into the speed of a successful development cycle. When you get stuck on a problem, there's a good chance that someone's already solved it and shared the solution on Stack Overflow. Node.js makes great use of this community, which is active and engaged when it comes to the popular runtime and its packages.

BEST FRONT-END FOR NODE JS

Why uses a framework?

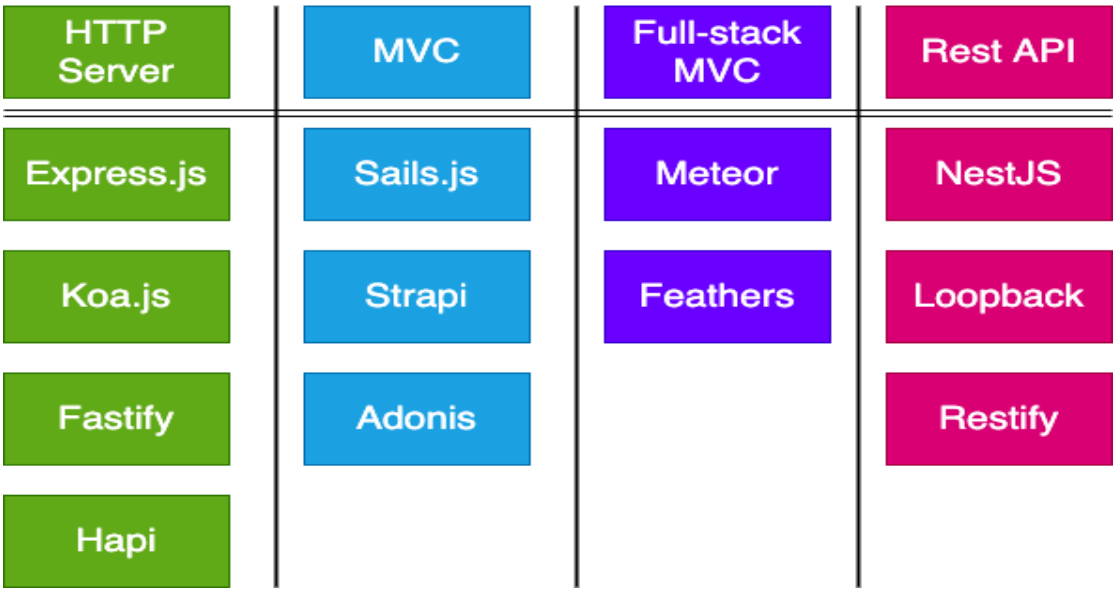
Writing server-side logic is complex and coding an entire app from scratch is time consuming. As developers, we need to focus on the business logic rather than wasting time creating an app from scratch.

Frameworks handle the heavy lifting and allow us to build applications more quickly. The framework provides a set of helper functions, a suite of tools, and rules. It helps us structure our application and write clean code in a short period of time.

Types of Node.js frameworks

You may feel excited to learn a new framework, but the sheer number of frameworks may overwhelm you. That's why I will start off with a diagram. Here we can begin to understand which framework might be the best choice for your project.

1. HTTP Server Framework
2. MVC Framework
3. Full Stack MVC Framework
4. REST API Framework

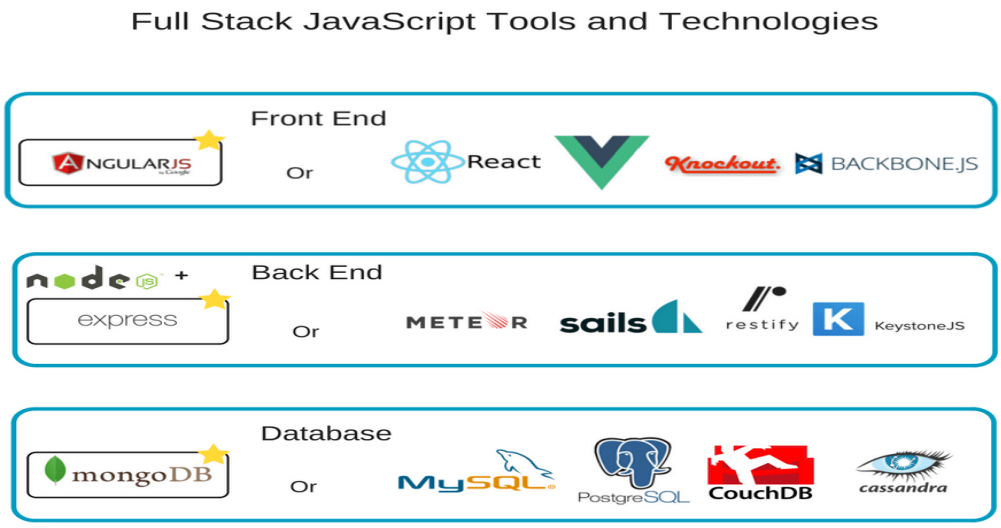


Front-end Framework working with Node.js

1. Angular: is the ideal choice for front-end integration with NodeJS as the backend.
2. React.js
3. Nest.js
4. Vue.js
5. Next.js

6. React Native etc.

Above frameworks are the best choice to use with Node JS.



SINGLE PAGE APPLICATION IN NODE JS

A single page application is a single page where a bulk of data remains the same, and only some bits get updated. At some point, it may be referred to as a collective term for a technology set to implement a Web app as a single page. Fast-response is quite an eye-catcher. See it as a switch between the screens while the page remains the same.

The development of SPAs is possible through JavaScript frameworks (React JS., Vue JS., Meteor JS., Angular JS.); Node JS backend platform, or directly with JavaScript. The choice is predetermined by the requirements that your business goal sets.

Key Features of SPA

Loading speed

Single-page applications, like any other tech innovation, have their advantages and disadvantages, which are acknowledged by both users and designers. Some of the benefits have previously been discussed, such as the loading speed, which greatly reduces the client's irritation.

Simpler development and debugging

Because you may be confined to JavaScript when designing a single page application, Node JS allows frontend and backend development to coexist seamlessly. Because the tools are already present in the frameworks, developing on JavaScript frameworks makes debugging procedures easy.

Efficiency in terms of slow connection

Users appreciate SPAs, even if they aren't always known they're using single-page apps, because they provide a far more efficient user experience by consuming less bandwidth and being able to work correctly in regions where the internet connection isn't in a rush to meet the users' needs.

Cost-effectiveness

From a cost-effective standpoint, a reduction in the number of servers with no change in traffic volume may save money. This is because no more requests to the server are required and full drawing is no longer required.

PROVIDING HIGH, MODERN WEB AUTHENTICATION AND AUTHORIZATION SECURITY

User authentication & authorization is one of the important parts of any web application. Authentication is the process of obtaining some sort of credentials from the users and using those credentials to verify the user's identity. Authorization is the process of allowing an authenticated user access to resources.

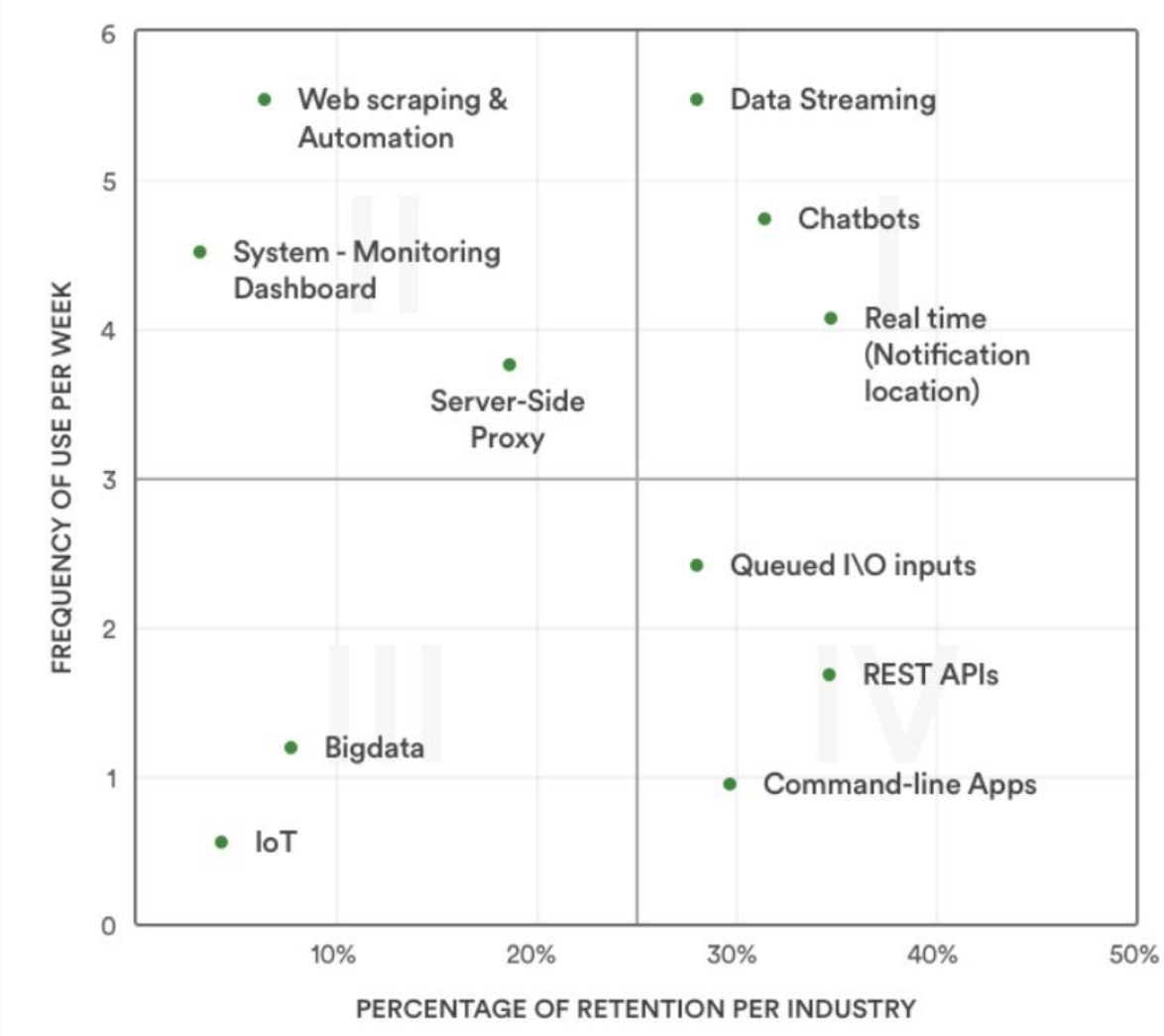
There are many algorithms available to achieve the high security of Authentication and Authorization. By combining different algorithms at the same time, we can achieve more high security. We can change or modify the security level according to your requirements.

Algorithms

- Sha256
- Bcryptjs
- Sha512
- Hash.digest()
- crypto.createHmac()

CATEGORIZATION OF NODE JS ACCORDING TO THE USAGE

10 industries that use Node JS in their tech stack and classified them into different categories.



Quadrant I

Quadrant I have many categories and industries are using it very often. This quadrant includes, unsurprisingly, data streaming, chatbots, and real-time data.

Quadrant II

Quadrant II contains categories only for a limited duration. Quadrant II includes web scraping, server-side proxy, and a system monitoring dashboard. These categories are heavily used by industries, they have a lower retention rate than those in Quadrant I.

Quadrant III

Quadrant III contains categories with low retention and usage frequency. However, in the next years, the use of Node JS in these categories is projected to rise like in Big Data and IoT technology

Quadrant IV

Quadrant IV contains categories with a modest frequency of use but a strong industry following. REST API, Queued I/O Inputs, and command-line programs are examples of this category.

NODE JS USE IN DIFFERENT INDUSTRIES

Now that you've gone through use case of Node JS by category, let's understand what are some industries that leverage these categories.

Media and Entertainment

More than 500k request per second send through media and entertainment industries and TBs of data store a day. To manage this kind of Web Application we required data streaming, scalable, and cutting-edge technology such as Node JS.

Application use in media and entertainment industry required high performance, speed of data, and Node JS send small chunk of data continuously so stream never stop. The data streaming would ensure that the initial request to the server will execute the first few seconds of the video and the rest will get downloaded while playing, e.g., Netflix, YouTube, etc.

Banking

Several banking businesses, like Capital One, use the JavaScript runtime environment because it speeds up development and customization. Capital One's developers used Node JS as an orchestration layer, utilizing asynchronous I/O to handle multiple server requests simultaneously. The server load of their banking software fell considerably once they switched to Node JS.

eCommerce

The eCommerce industry is well-versed in the use of Node JS! It can be used as a frontend or a backend to support significant traffic spikes during the holiday shopping season.

For example, Groupon, a daily deal website, moved their ROR software to the Node JS stack and saw significant improvements!

Education and eLearning

Because of the Internet, the education industry has been disrupted by eLearning web apps. Advanced features like course authoring, database management, forums, and webinars, reporting and analytics, and video conferencing are all available through an eLearning web app.

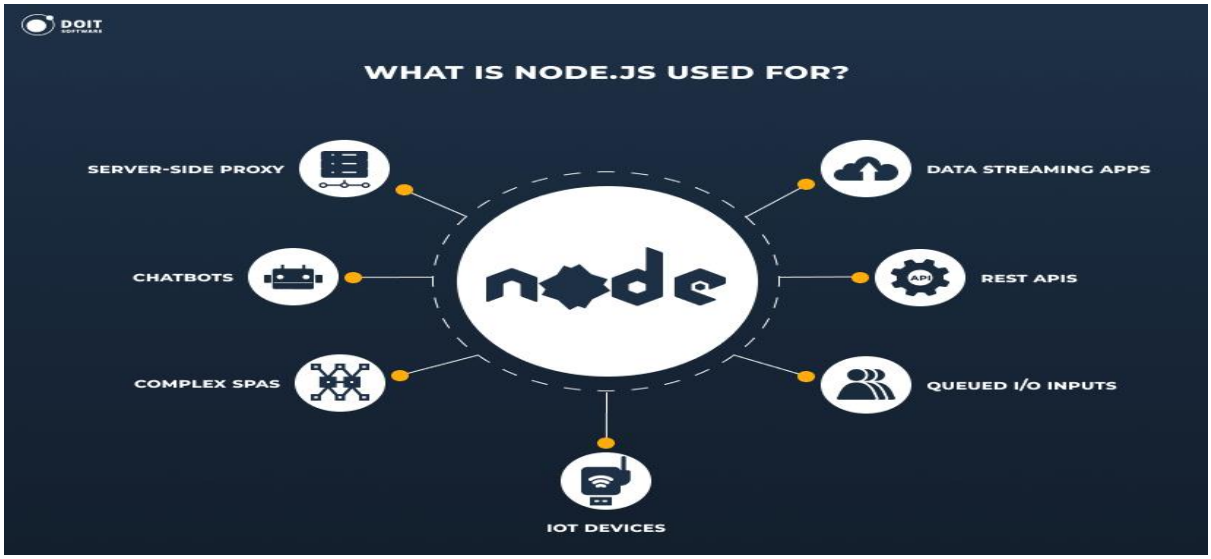
You might create a popular eLearning online software like Quizlet with Node JS, which ensures quick server-side JavaScript code execution and makes a web app lightweight and efficient.

Social Network and Communities

If you want to build an application like Facebook, Twitter, LinkedIn etc. Node JS is a best technology. By using the Node JS Stream API, we build a scale and personalize the news feeds or activity streams of your social media app.

LinkedIn, the world's largest business and employment-focused social networking service, also relied on Node JS. The organization made the decision to go from ROR to Node JS, which turned out to be a wise decision.

LinkedIn experienced 20 times faster and enhanced performance while using less memory after switching to Node JS.



BENEFITS OF NODE JS DEVELOPMENT SERVICES

Simplicity and Cost-Efficiency

One language for both client and server sides minimize the need to switch back and forth from front-end to back-end, which means you don't need to hire multiple teams for one project.

Faster Development Process

Due to the possibility of shared and reused code, JSON support various free tools and the smaller stack of files needed to finish the application.

Scalability

Scalability is usually mentioned as the №1 Node.js benefit and not without reason — the load balancing and the tool's capacity to handle multiple parallel connections, plus vertical and horizontal scaling support does wonders for businesses that aim to quickly scale and grow.

Quick Synchronization

Thanks to an event-based model that helps make real-time updates while simultaneously carrying out a series of secondary tasks in the background.

An Active Community

Not only Node.js but JavaScript as well — which means you're not alone if you encounter any problems. There's enough feedback and different internet resources to help you tackle any issue due to a lack of experience.

Higher Adaptation Rate

Due to easier and faster learning curve, especially for those familiar with its partner — JavaScript, and better for customizations due to its built-in API.

Decreased Loading Time

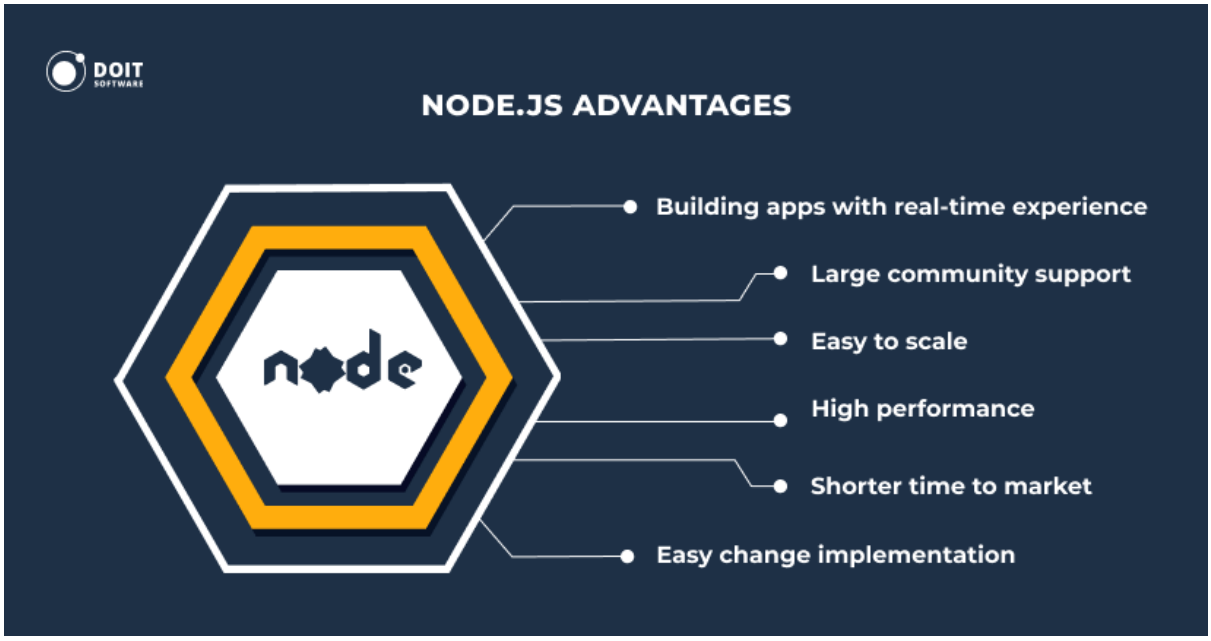
Decreased Loading Time is also among Node.js advantages thanks to a direct caching module.

Quicker Development of a Minimum Viable Product (MVP)

Not a fully-fledged solution, but one with just enough functionality to be well presented on the market.

Utilizing Microservices

Node.js helps developers separate their projects into smaller and more manageable sections. It, in turn, helps efficiently distribute tasks among team members.



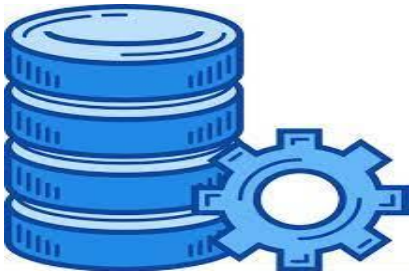
NODE JS PIPELINE

Node JS request processing pipeline consists of a sequence of middleware components that are going to be called one after the other.

- Each middleware component can perform some operations before and after invoking the next component using the next method. A middleware component can also decide not to call the next middleware component which is called short-circuiting the request pipeline.
- The middleware component in asp.net core has access to both the incoming request and the outgoing response.
- The most important point that you need to keep in mind is the order in which the middleware components are added in the Configure method of the Startup class defines the order in which these middleware components are going to be invoked on requests and the reverse order for the response. So, the order is critical for defining the security, performance, and functionality of the application.

DATABASE MODELING, AND CONFIGURATION

Database configuration



Application configuration files contain settings that are specific to a particular application. For example, in Node JS, application can have ENV file where you can store your private data variables such as private key, database string and passwords etc. which are use at different places of the application during process. You can connect your database by using the database string variable in file of the project. However, mainly connection build in Server.js file if you separate server connectivity from App.js otherwise App.js. ENV files share common elements, although the name and location of a configuration file vary depending on the application's host.

Database modeling

A data model is basically a visual representation that describes the connections between different data points and structures stored in an information system. The structure determines how the data is stored and how the system will access it. Efficient data modeling software is needed for an appropriate structure of the data. When conducting due diligence about leveraging your data to fuel business decisions, the integrity of said data is a critical prerequisite.

Each instance of a database is identical. Relationships and rules are designed and programmed into the database by a modeler. An ideal programmer for this job must grasp this and formulate a plan to carry out the task accurately and efficiently.

The Three Layers of a Data Model

Your data is going to be structured using a data warehouse modeling tool into one of the following three distinct layers, each with specific placement and function. Let’s delve into each layer separately:

Router Layer

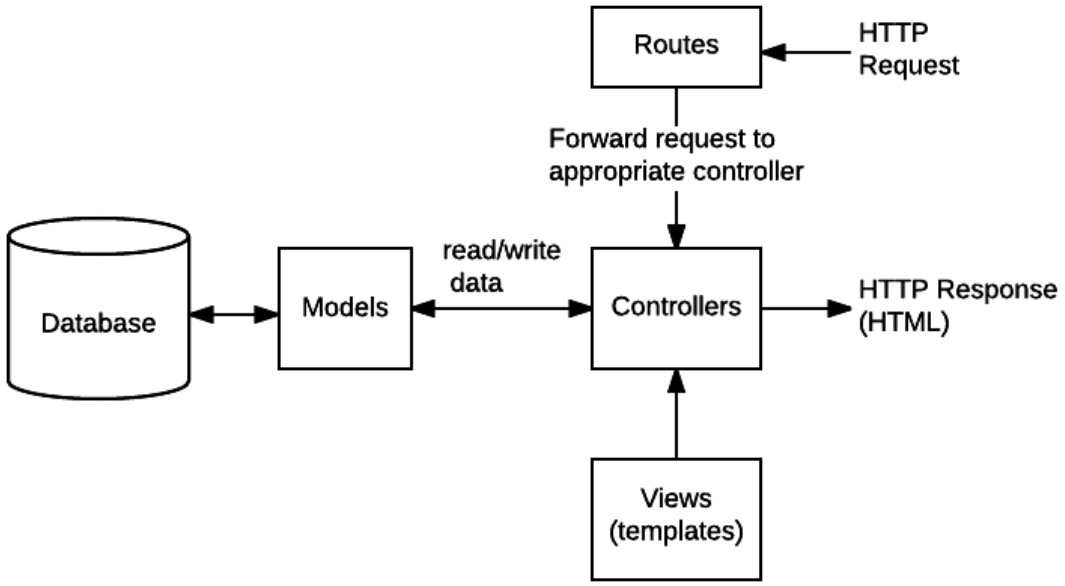
It Contains the app programming interface (API) routes of the app. Its only job is to return a response from the server.

Service Layer

It handles the business logic of the app. The data is transformed or calculated to meet the database model's requirements before being sent to the server.

Data Access Layer (DAL)

It has access to the database to create, delete, or edit data. It is where all the request and response from server logic is handled. This layer may include Hypertext Transfer Protocol or http requests to the server if there is no database connected directly into the app.



BIG COMPANIES THAT ARE CURRENTLY USING NODE JS

Based on lots of benefits from adopting Node.js into a development cycle, it's no wonder many companies have already done it. Let us present you some top companies that are ideally adopting Node JS.

Netflix

To modernize their backend development, Netflix added Node.js to their technology list. It sped up the move from the backend to the frontend and significantly decreased their loading time.

PayPal

After their regular programming tool, Java, didn't pair up well with the frontend, PayPal switched to Node.js. It upped their performance, built a new application in record time, and reduced loading time.

LinkedIn

LinkedIn has traded Ruby on Rails for Node.js and, with its help, managed to build a ten times faster application and dramatically cut down the number of servers.

Uber

Node.js helped Uber handle the growing demand and increased traffic by boosting their data processing power and connectivity while minimizing management expenses.

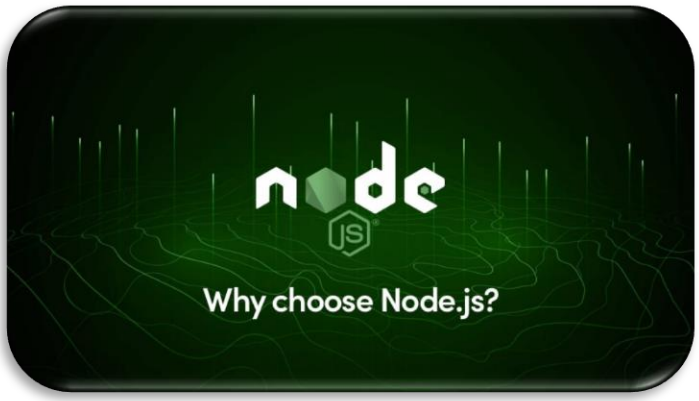
eBay

Like PayPal, eBay has also switched from Java to Node.js to upgrade their development process and synchronize any changes to client and server sides.

WHY TO CHOOSE NODE JS

Node JS is a leading technology and we've listed its main benefits:

- Unified software model
- Simplified programming model
- Package version control
- Simplified deployment
- Multi-platform
- Automatic resource management
- Type safety checking
- Unified development environment
- Ability to debug multilingual applications
- Unified error handling model
- Modern security model
- Integration of programming languages



CONCLUSION

As we have seen several advantages of Node JS for application development. Node JS is a reliable, safe, and beneficial platform using which you can build ultimate business applications. It allows **Node JS developers** to apply, expand, and scale excellent software, for both economic development and general business prosperity.

Node JS development services provide endless benefits for different problems such as security, memory management, phenomenal handling, etc. Node JS is an exciting and future dominating technology, appropriate for building robust web applications. Since most organizations search for a unique business solution to assist consumers in the best possible way, you can contact Cognitive Convergence.

Contact Us
Cognitive Convergence
http://www.cognitiveconvergence.com
+1 4242530744
info@cognitiveconvergence.com