# Comparing Node.js frameworks:

Fastify [@Yitav Gil-ad](https://appleseeds-bootcamp5.slack.com/team/U02JT4AEMFX)  
Restify [@Shmuel Gabai](https://appleseeds-bootcamp5.slack.com/team/U02JT4AJSQ5)  
Nest  [@Stas Isackov](https://appleseeds-bootcamp5.slack.com/team/U02GJJH9USV)  
Koa [@Samer Siam](https://appleseeds-bootcamp5.slack.com/team/U02GJK754N5)  
Polka [@Harel](https://appleseeds-bootcamp5.slack.com/team/U02G60LAATV)

Express [@Hope](https://appleseeds-bootcamp5.slack.com/team/U02G60LAATV) Mashal

## KOA

What is KOA?

Developed by the same team that developed Express, aimed at improving Express performance by not budling middleware within its core.

|  |  |
| --- | --- |
| Pros | Cons |
| Light Weight -Minimize middleware budling within its core | Not compatible with Express-style middleware or any other Node.js middleware |
| High performance | Open-source community is very small |
| Uses ES6 Generator feature- no callback | GitHub repository has 29.9k stars, 218 contributors, and 2.8k forks. |
| Middleware cascading - customized middleware to execute one after another without any nasty callbacks. | According to [npm](https://www.npmjs.com/package/koa), 4,678 packages depend on Koa.js. |
|  |  |

## Nest.js:

|  |  |
| --- | --- |
| Pros | Cons |
| - uses express under the hood | - heavy |
| - combination with fastify adapter makes it 2x faster(see: https://github.com/nestjs/nest/runs/482105333) | - very opinionated i.e lots of syntax to learn |
| - uses decorators which is very intuitive |  |
|  |  |

## Name: Polka

|  |  |
| --- | --- |
| Pros | Cons |
| faster than Express- 33-50% faster than Express for simple applications-native HTTP server with added support for routing middleware, and sub-application | This seemed like a good initially just because of then/catch, but there are a few problems with this: |
| Middleware support, including Express middleware you already know & love-Nearly identical application API & route pattern definitions-~90 LOC for Polka, 120 including its router/ שורות קוד בסה"כ | Calling .listen() does not return the Polka instance directly, so it forces listen to always be called last. |
| -A custom, instantiated server that the Polka instance should attach its handler to. This is useful if you have initialized a server elsewhere in your application and want Polka to use it instead of creating a new http.Server. | It also prevents one-line initializers for a server.const server = polka().listen(3000);//=> server is Promise, not Polka instanceThe top-level then and catch blocks make it seem like catch is the global catch-all handler.Instead, .listen() would expose the underlying server.listen method directly, meaning that signature is the same as all other servers. |
| Polka only updates the server when polka.listen is called. At this time, Polka will create a http.Server if a server was not already provided via options.server. |  |
|  |  |
|  |  |
|  |  |

## RESTIFY

|  |  |
| --- | --- |
| Pros | Cons |
| minimalistic | Performance is a downside in restify |
| support for Dtrace |  |
| support JSON, HTTP and String, no parsing needed |  |
| Sinatra style handle chaining |  |
| semantic API |  |
| Pre-handlers  Debugger  High scale  NPM uses it |  |

## Fastify

|  |  |
| --- | --- |
| Pros | Cons |
| \*Built with the intention of being a very fast Node.js web framework | It has less community support than Express: Less plugins, less use in the industry and less community support for debugging |
| Preforming at 34,528 req/sec (Express is 8646 req/sec) |  |
| Supports async functions for use as controller code. |  |
| \*Automatically parses incoming requests into JSON if the Content-Type header suggests the body is JSON. This allows for a shorter code. |  |

## ExpressExpress is the most well-known Web Application framework based on Node.js and one of the most used Web Application framework overall. It is mainly used to develop Web Applications and REST API.

It is very fast and less opinionated. Express is also a “framework of other frameworks.” Many other frameworks are built on Express.

|  |  |
| --- | --- |
| Pros | Cons |
| The first and most important advantage of using Express.JS is that you would be able to get fast application development experience with it. | Several client request problems are faced with the middleware system offered with Express.js. |
| Several such platforms are unable to handle a higher level of requests, but with the help of Express.JS, you would be able to handle requests efficiently as it offers you the support of I/Q request handling. | There are several issues in the callbacks. |
| Express.JS has a vast, highly supportive open-source community. |  |
| You would be able to integrate several third-party applications and services with Express.JS. |  |