Hey what's going on guys so in this presentation we're gonna be talking about two most popular frameworks for node.js

Let's fast forward to 2009, when Ryan Dahl's two-year experiments on creating web server components succeeded and a fundamentally new technology appeared.

Node.js is an event-oriented platform used to create web applications

which provides the ability to use JavaScript on the server side.



This might look like a typical http server example on Node.js.

requestHandler: this function will be called every time a request comes to the server. If you open localhost: 3000 in your browser, two messages will appear in the console: one for /

and one for favicon.ico.

if (err): error handling: if the port is already busy or there are any other reasons why the server cannot be started, we will receive a notification about it. The http module is extremely low-level: creating a complex web application using the above code snippet is very time consuming.

And since before the advent of Node.js, the JavaScript language was used only on the client side, the appearance of such a platform was greeted by developers with

enthusiasm. This opened up new opportunities for creating applications with high performance and scalability.

It must be admitted that the expectations of the developers were justified, and at the moment Node.js remains popular and continues to move forward.

New ideas are generated regularly, and as a result, new tools and libraries appear. Such as for example Express and Кoa.

So I will first describe the main characteristics of each

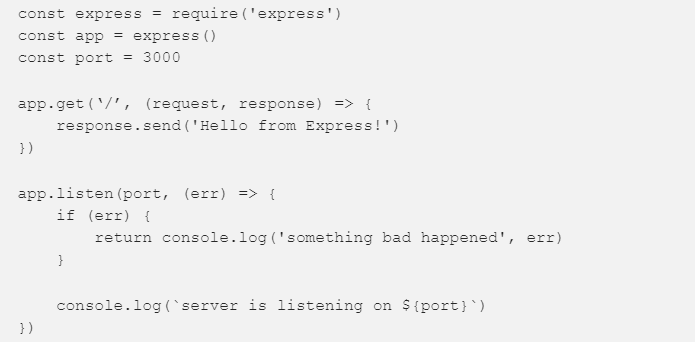
of them. And when you have a general impression, compare them to

based on several key criteria.

I will begin the description with the simplest framework used on the Node.js platform

Express has been used to develop applications for quite some time and thanks to its stability.

firmly occupies the position of one of the most popular Node.js frameworks.



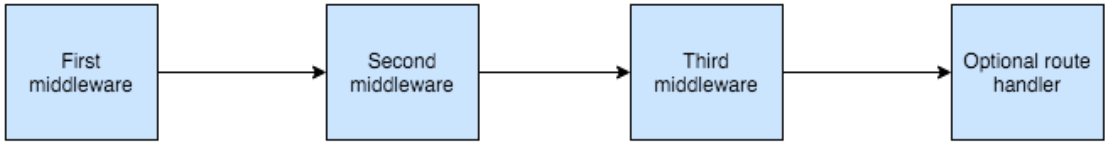
The biggest difference you can notice here is that Express gives you a router by default. You do not need to manually parse the URL to decide what to do, instead you define the application routing using app.get, app.post, app.put and so on, and they are already translated into the corresponding HTTP requests.

For example, one of the powerful concepts that Express implements is the Middleware pattern.

**Middleware**

Middleware - intermediate processor

You may think of intermediate handlers as Unix pipelines, but for HTTP requests.



On the diagram, you can see how the request goes through the conditional Express application. It passes through three intermediate handlers. Each handler can change this request, and then, based on your business logic, the third middleware will send a response, or the request will go to the handler of the corresponding route.

I will not dwell in detail, for too long. For this framework, there are a large number of detailed instructions and descriptions that are compiled by the developers who checked it.

efficiency in practice. That is why it is recommended to start working with Express if you intend to learn how to create applications on the Node.js platform.

The main feature of this framework is that Express is characterized by a small amount of basic functionality.

All other functions you need will need to get at the expense of external modules. Essentially, pure Express is a server.

and it may not have a single module.

Thanks to this minimalism, the developer initially has at his disposal a light and fast tool that he can expand.

At the same time, it is important that the choice of modules for Express is not connected with any restrictions: neither with quantitative, nor with functional.

As a result, this framework provides the developer with the ability to solve any tasks without limiting him in the choice of means.

Also pleased with the fact that the lack of ready-made universal solutions actually means that each application created will be unique.

However, the developer needs to independently select and organize modules, and this implies a large amount of work and, accordingly,

requires more time and effort from the developer.

PROS:

+ simplicity

+ flexibility

+ good scalability

+ developed community

+ detailed documentation

+ wide range of plugins

MINUSES:

+ large amount of handmade

+ uses outdated function callbacks approach

JUmp on to

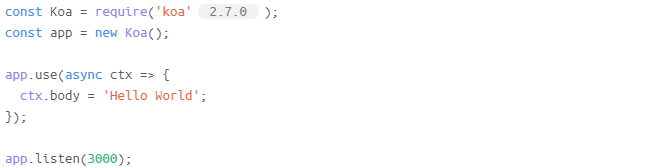
KOA.JS!

Koa was created by the development team, as a variant of the express framework of the new generation.

This improved version was developed to create web applications and APIs with enhanced performance.

Accordingly, the creators sought to take into account all the shortcomings of the predecessor and make it more modern and convenient to use.

Let's see how it was possible.



A typical example of a simple koa.js server.

Koa has almost the same functionality and surpasses Express in ease.

A characteristic feature of Koa is the use of ES6 generators.

Generators - the type of functions that can be started, stopped and resumed regardless of

at what stage of execution they are, and at the same time retain their content.

The use of ES6 generators in Koa allows you to exclude callbacks, reduces the amount of developers

work with the code and reduces the likelihood of errors.

Due to the fact that the creators of Koa have already taken into account the disadvantages identified in the process of working with Express,

This framework can boast that its use greatly simplifies the adaptation under

specific customer requests (customization). And such a characteristic can ultimately play a decisive role.

role in the selection process: today, in the context of high competition, any application

committed to using her own style.

If we talk about shortcomings, they are mainly related to the relative youth of Koa (appeared in 2013).

The framework does not enjoy the support of such an extensive community as Express, and has not yet had time to demonstrate all its capabilities.

PROS:

? light

? flexible

? fast

? ES6 generators

? better customization

MINUSES:

? insufficient community support

CONCLUSION

I will try to draw a general conclusion from what we have already told about these frameworks. Let us determine for which projects each of them is better suited:

Express.js is suitable for:

novice programmers who focus on professional growth in Node JS;

large projects involving customization;

cases where long-term application support is needed.

Koa.js is suitable for:

both small and large projects that require development in the future;

for projects with a high degree of customization;

to facilitate long-term application support.

Now that we have examined in detail the pros and cons of each of the three frameworks, let us tell you what framework we choose and why.

I would give my preference to Koa because:

Like Express, Koa does not limit the developer to use embedded modules, but allows you to choose from

the set is the one that is best suited for a particular project;

Koa has incorporated the advantages of a proven and widely used Express framework;

Koa's creators had the opportunity to analyze the difficulties faced by developers who used Express;

when creating Koa, the shortcomings of its predecessor were taken into account;

Koa is based on new standards and current trends;

Koa is suitable for developing a wide variety of applications of any size, with any degree of customization and with any support requirements;

This is my opinion. In any case, the choice is that it is up to you.

It’s all

Thanks for watch.