JavaScript on the Server

This lesson explains how JavaScript is an important technology to use from the server side development perspective.

If these benefits were not enough for you, let's add some more sauce to the mix. Let's first focus on the server-side programming.

> ***Server-side*** development focuses on writing code that runs on the server of the service provider.

`Node.js` and Server-Side Development

> JavaScript is not just a client-side language anymore.

`Node.js` makes it possible to run JavaScript on the server. `Node.js` is built on the foundations of the execution engine of Google Chrome, called V8. The V8 engine is fast like a Formula One car. The V8 compiler turns JavaScript to machine code, which means that on the server, we are not dealing with *interpreted* code anymore.

> Interpreted code runs on a *virtual machine*, translating the sets of instructions written by a developer to executable commands. Interpreted languages are typically safer, but slower than *compiled* languages. Interpreted languages are also *portable*, which means that you can take the source code of a program and execute it on any system that has a virtual machine for that specific code. Languages that compile to *machine code* have the benefit of running code as close to machine code level as possible. This translates to faster execution times. Unfortunately, this machine code cannot be *ported* to different machines running different operating systems or having different hardware architecture.

> architecture.

Node.js solved the speed problem by offering a compiler to machine code. This is not a big deal yet, because most other server-side languages can do the same thing.

Why is `node.js` still a good choice for certain server-side tasks?

The reason lies in some `node.js` architectural decisions. `Node.js` comes with an **event loop** that can execute and serve parallel requests on the server without blocking. This is called *non-blocking I/O*. Opposed to many other languages, when you have to read or write a database or a file, node.js is not blocked until the I/O operation is finished. This means that `node.js` stays busy handling other tasks, possibly important for other clients.

Think about it for a moment. When is this feature beneficial?

> `Node.js` shines when there are a lot of independent parallel requests requiring a lot of I/O operations, but not many server-side computations.

For instance, 3D renderer algorithm comes with a lot of CPU operations. Even though the V8 engine produces fast machine code, there are some other languages that are faster. As those languages have a ***blocking*** nature when it comes to I/O operations, `node.js` gets the edge over them once we deal with many inexpensive operations that interact with a database.

In today's world, this use case happens to be very popular. The computers of the end users are strong enough to run a lot of logic on client-side. The application server providing an API mostly interfaces database servers and asset servers, and formats data in a way that clients can fetch and understand them using AJAX requests. The role of the server is limited to purposes that do not require a lot of CPU operations. This is an optimal environment of node.js.

This is because a lot of problems have been solved effectively by **microservices**. You can simply query a microservice to perform a CPU-heavy operation, often on another server, using a different technology. If you are

interested in microservices, check out my introduction on this topic.

Quick Quiz!

1 In node.js, everything is asynchronous.

COMPLETED 0%

1 of 2

Conclusion:

As a conclusion, once you learn client-side development, you can upgrade your skills to perform server-side development as well. Developers who are good at both client side and server side development are called *full stack developers*. Full stack development is a great career path upgrade for you in case you are looking for more responsibility in a smaller company.

Summary:

- JavaScript is compiled to fast machine code on the server, using the V8 engine of Google
- Node.js comes with non-blocking I/O, making it possible to handle many I/O operations in parallel