# Is JavaScript Compiled or Interpreted?

Before we discuss whether JavaScript is compiled or interpreted, let's first define what each term means via an analogy. Let's say that we are stranded on an alien planet. Our spaceship needs fixing and the only way that can happen is if we hire a local mechanic. But therein, lies a problem. The instruction manual that we have is written in a language that the mechanic doesn’t understand. Logically, our next step is to hire a translator and we have a choice of either hiring an interpreter or a compiler. An interpreter would take the list of instructions from us, read the first instruction, translate it to the mechanic, wait for the mechanic to execute the step, and then keep repeating this process till the end of the document. Conversely, if we hire the compiler, it would take our list of instructions, immediately translate the entire page and give it back to us. We can then give the translated list to the mechanic who will quickly start working on the spaceship.

There are drawbacks to both. The interpreter, since it translates line by line, takes too much time between steps but it gives us a chance to quickly correct our instructions if we notice any mistakes. In the case of the compiler, the execution step will be a lot faster, as the mechanic doesn't have to wait to receive instructions; however, if there's a mistake in our instruction, we can't fix it on the fly. This is the case in programming, where the stranded pilot is the user, the interpreter is an engine (a program that executes source code) and the mechanic is the computer.

Now that we’ve got the definitions out of the way, let’s see which category JavaScript fits in. There is a general consensus that JavaScript like most scripting languages is an interpreted language. Although that was the case when JavaScript was first implemented, it has changed over time. Nowadays, it is [JIT-compiled](https://blog.bitsrc.io/the-jit-in-javascript-just-in-time-compiler-798b66e44143)(linked article describes it in detail) to native machine code in all major JavaScript implementations. Exactly when it’s compiled to machine code varies based on implementation. In the current V8 (used in Chrome and Node.js), it starts out using an interpreter since there is little reason to spend time compiling code that only runs once. However, if a function gets executed more than a couple of times, it’s immediately compiled into optimized native machine code. There is no way a JavaScript engine could ever hope to compete with other JavaScript implementations without compiling to machine code as modern JavaScript is really quite fast.

In conclusion, JavaScript, although it was initially implemented as an interpreter, is currently more closer to a compiler than an interpreter.