Difference Between Compiled and Interpreted Language

Let us check the difference between Compiled and Interpreted Language based on various parameters given in the table below.

Compiled VS Interpreted Language	
Compiled Language	**Interpreted Language**
The code of a compiled language can be directly executed by the CPU of a computer.	A program written in an interpreted language is interpreted rather than compiled.
Compile-time programs are faster than interpret-time programs.	While the program is running, interpreted programs can be modified.
From source code to execution, there are at least two steps.	From source code to execution, there is only one step.
A compiled language is a programming language with compilers rather than interpreters as implementations.	An interpreted language is a programming language whose implementations directly and freely execute instructions without first compiling a program into machine-language instructions.
Examples of compiled languages are C, C++, C#, CLEO, COBOL, etc.	Examples of Interpreted languages are JavaScript, Perl, Python, BASIC, etc.

What is Compiled Language?

A compiler is a program that generates a set of instructions that computers can execute based on a representation of the meaning of the code. A compiler translates the entire source code. Compilers required the information to enable advanced optimization and efficient code representation. During compilation, the optimization process can obtain the values in an expression.

The compilation is a set of transformations that converts the source language to the target language. As some compilers, such as Dart, can translate to JavaScript, a target language could be another programming language. Meanwhile, other compilers, such as Java, generate bytecode, which the JVM (Java Virtual Machine) interprets to generate a set of instructions that processors can execute.

What is Interpreted Language?

After knowing the difference between Compiled and Interpreted Language, let us discuss interpreted language. An interpreter generates machine code one line at a time by translating each line of code. The interpreter has fewer opportunities for optimization because it translates your program at runtime. Meanwhile, because compiled languages are notorious for their cryptic error messages, translating programs at runtime results in a dynamic type system that provides flexibility and ease of handling errors.

In a compiled language, re-compilation may necessitate restarting the entire compilation even if only minor portions of the code are changed. In some compilers, this process can take up to 30 - 40 minutes for large projects. As a side note, modern compilers have optimized for this (e.g., Dart VM Hot Reload to minimize development time and maximize productivity), but interpreted programming languages are designed for rapid prototyping and agile development.