Transpiling on the other hand, takes the source-code from a high-level programming language and converts it into bytecode.  
Integrated development environments (IDEs) aim to integrate all such help.  
Proficient programming thus usually requires expertise in several different subjects, including knowledge of the application domain, specialized algorithms, and formal logic.  
Transpiling on the other hand, takes the source-code from a high-level programming language and converts it into bytecode.  
Trade-offs from this ideal involve finding enough programmers who know the language to build a team, the availability of compilers for that language, and the efficiency with which programs written in a given language execute.  
Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability.  
Programming involves tasks such as analysis, generating algorithms, profiling algorithms' accuracy and resource consumption, and the implementation of algorithms (usually in a particular programming language, commonly referred to as coding).  
Techniques like Code refactoring can enhance readability.  
The choice of language used is subject to many considerations, such as company policy, suitability to task, availability of third-party packages, or individual preference.  
For this purpose, algorithms are classified into orders using so-called Big O notation, which expresses resource use, such as execution time or memory consumption, in terms of the size of an input.  
However, readability is more than just programming style.  
Unreadable code often leads to bugs, inefficiencies, and duplicated code.  
 After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug.  
 It is very difficult to determine what are the most popular modern programming languages.  
To produce machine code, the source code must either be compiled or transpiled.