**1. What is Parser ?**

A parser in JavaScript is a program or function that takes a string of code or text as input and converts it into a data structure, typically an Abstract Syntax Tree (AST), that represents the syntactic structure of the input. This process is known as parsing.

JavaScript itself has a built-in parser that is used to execute JavaScript code. Additionally, there are many third-party parsing libraries available for handling different types of data and languages.

**2. What is Abstract Syntax Tree ?**

Abstract Syntax Tree (AST) is a tree representation of the syntactic structure of source code. Each node in the tree represents a construct in the code, such as a variable declaration, an operator, or a function call. ASTs are used by compilers and interpreters to understand and process code**.**

To generate AST in JavaScript, tools like **acorn** or **babel** can be used. These tools parse the code and create the corresponding AST.

**3. What is JIT compiler in java Script?**

Just-In-Time (JIT) compilation in JavaScript is a dynamic optimization technique where the JavaScript engine compiles code into machine code during runtime, just before it's executed. This approach combines the benefits of both interpretation and ahead-of-time compilation.

Initially, the JavaScript engine interprets the code, but as the program runs, the JIT compiler identifies frequently executed sections It then compiles these hot spots into optimized machine code, which is executed directly by the processor. This process significantly improves performance compared to pure interpretation, as the compiled code runs much faster. The JIT compiler also monitors the code's execution, and if the assumptions it made during compilation become invalid it can deoptimize and recompile the code as needed.

Most modern JavaScript engines, including V8 (used in Chrome and Node.js), SpiderMonkey (Firefox), and JavaScriptCore (Safari), employ JIT compilation to enhance the execution speed of JavaScript code.

**4.** **Compiler VS Interpreter ?**

|  |  |
| --- | --- |
| **compiler** | **interpreter** |
| Translates entire source code into machine Code before execution. | Translates and executes code line-by-line during runtime. |
| Generally faster due to pre-translation | Usually slower as each line is translated on the fly |
| Displays all errors before execution | Reports errors as they occur during runtime |
| Can be more complex as errors are not immediately visible | Easier to debug as code is executed line-by-line |
| May require recompiling for different platforms | Can run the same code across different platforms without recompiling |
| Less flexibility for dynamic changes | More flexible for interactive and dynamic execution |
| Ex:- C++ compiler | Ex:- JavaScript engine in a web browser |