# JavaScript Compilation Nature

## JavaScript is an Interpreted Language

- JavaScript is an interpreted language.
- But the browser engines, like V8 in Chrome and SpiderMonkey in Firefox, use JIT compilation techniques. Instead of fully interpreting the code, they dynamically compile parts of the code into machine code just before execution. This provides significant performance improvements.

## In-depth discussion

**Interpretation:**

In a purely interpreted language, code is executed line by line, directly translating high-level instructions into machine-level instructions as it encounters them.

### JavaScript's Execution Model:

**a. Parsing:**

When a JavaScript program is run, the first step is parsing. The code is analyzed to identify its structure, syntax, and semantics. This creates an abstract syntax tree (AST), a data structure that represents the code's hierarchical structure.

**b. Compilation:**

Here's where JavaScript's execution differs from traditional interpretation. The AST is transformed into an intermediate

representation (IR) or bytecode. Some JavaScript engines might even generate optimized machine code directly from the AST. This compilation step allows engines to catch syntax errors early and optimize the code for better performance.

**c. Execution:**

The bytecode or machine code is executed by the JavaScript engine. The engine includes an interpreter that can execute the bytecode directly. This is similar to interpretation, but here, the bytecode is usually faster to execute than the original source code.

## Just-In-Time (JIT) Compilation:

Many modern JavaScript engines, like V8 in Chrome and SpiderMonkey in Firefox, use JIT compilation techniques. Instead of fully interpreting the code, they dynamically compile parts of the code into machine code just before execution. This provides significant performance improvements.

**a. Baseline Compilation:**

The engine quickly compiles the code into a basic form of machine code as it is encountered. This allows for faster execution compared to full interpretation.

**b. Optimization:**

The engine observes how the code is behaving during execution and applies various optimization techniques to the compiled code. This can involve inlining functions, eliminating unnecessary operations, and rearranging code for better cache usage.

**c. Deoptimization:**

If the assumptions made during optimization are violated (due to dynamic typing or other reasons), the engine can "deoptimize" by reverting to the baseline compilation and adjusting its assumptions.