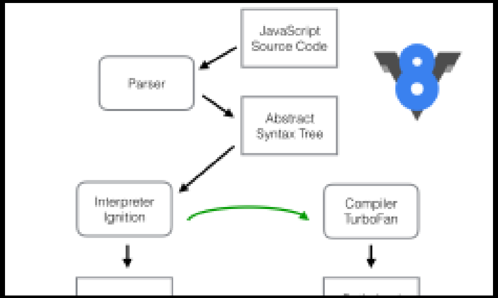
PROJECT WORK

Mini Javascript Engine (parser+interpreter)

SESSION : 2025-26

TEAM MEMBER	Student Id
ISHA JOSHI (Leader)	220222892
Anjali Lohani	220221617
Aditya Chauhan	230213709
Dhruv Gangwar	22021734



OBJECTIVE

- **Understand JavaScript Internals**
  Learn how JavaScript code is parsed, structured, and executed behind the scenes.
- **Implement Core Compiler Concepts**
  Apply key phases of compilation: lexical analysis, parsing, and interpretation.
- **Build an AST-Based Execution Model**
  Convert code into an Abstract Syntax Tree (AST) and execute it node by node.
- **Simulate Variable Scoping & Expressions**
  Handle variable declarations, arithmetic, and simple expressions using an interpreter.
- **Gain Hands-On Experience with Language Design**
  Get practical experience in building a simplified programming language engine from scratch.

TECHNOLOGY STACK

💻 Language & Runtime

•**JavaScript (ES6+)** – Core language for writing the engine

•**Node.js** – Runtime environment for executing the interpreter

⚙️ Core Tools & Libraries

•**Custom Recursive Descent Parser** – For parsing JavaScript-like syntax

•**Tokenizer / Lexer** – Built from scratch using RegEx or FSM

•**AST (Abstract Syntax Tree)** – Custom data structures to represent parsed code

•**Interpreter** – Built to walk the AST and evaluate expressions/statements

🧪 Testing & Debugging

•**Jest / Mocha** – For unit testing the parser and interpreter

•**Console Debugging** – Node.js built-in tools (like console.log, debugger)

🛠️ Dev Tools

•**VS Code** – Code editor

•**Git + GitHub** – Version control and collaboration

📦 Optional Enhancements

•**ESLint** – For code quality and linting

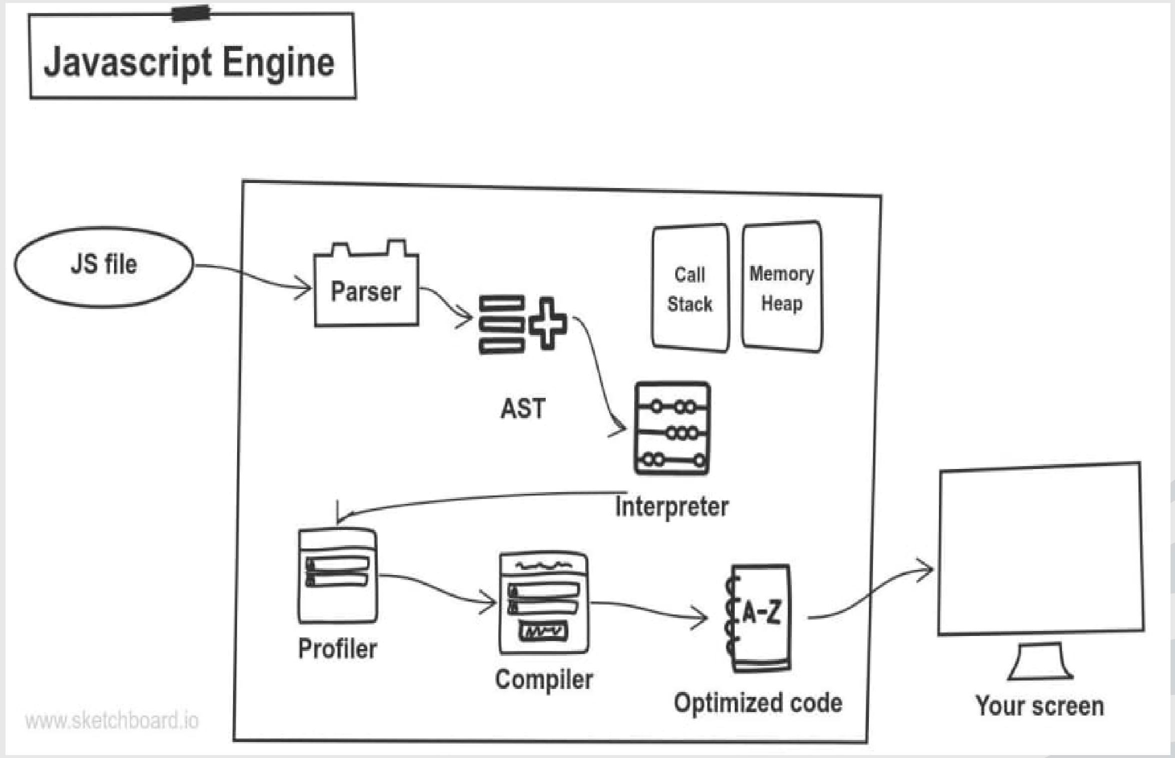
•**Prettier** – Code formatting

•**Commander.js / Inquirer.js** – For CLI interface (if needed)

Project idea/Overview

🔍 **Overview:**

•Build a simplified engine to parse and interpret JavaScript-like code
•Simulates how programming languages are processed and executed
•Covers key concepts: tokenization, parsing, AST, and interpretation



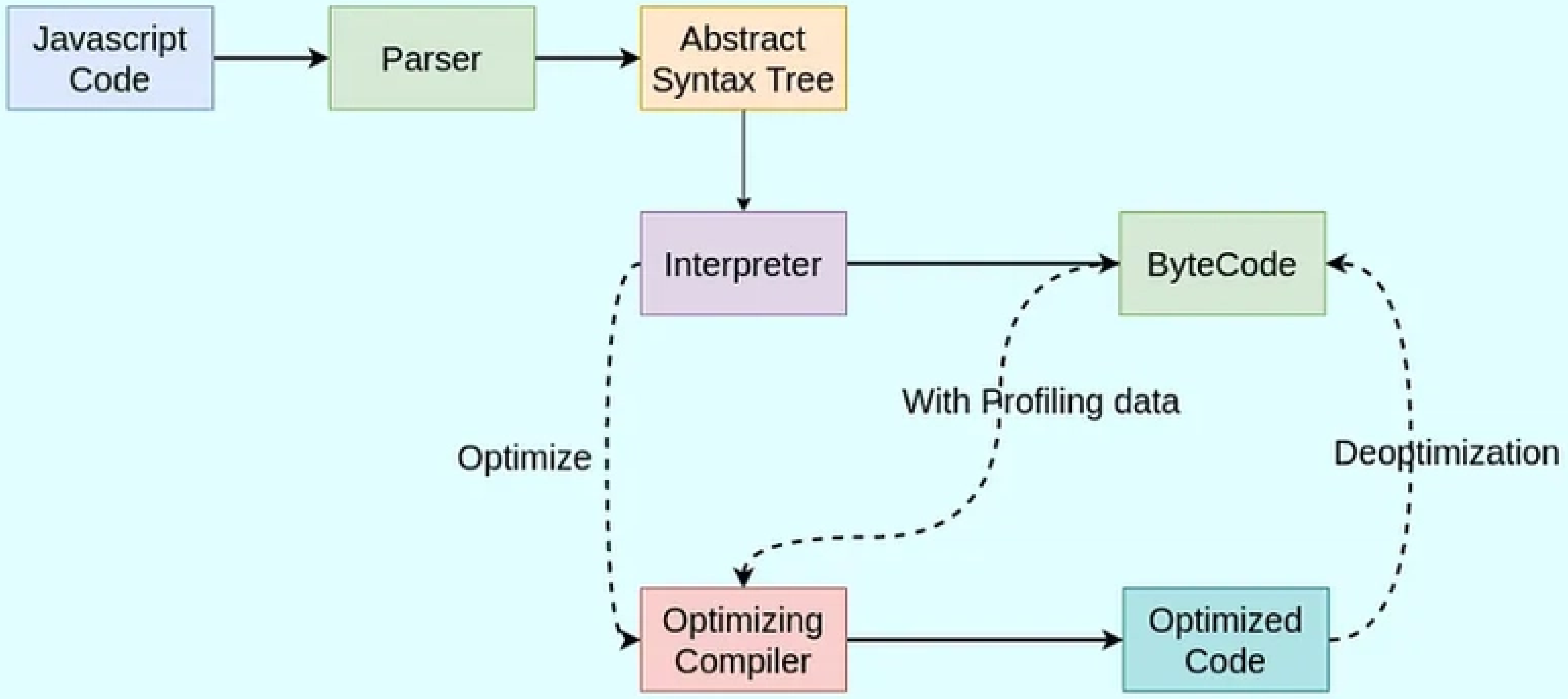
🧩 **Core Components:**

•**Tokenizer (Lexer):** Converts source code into tokens
•**Parser:** Builds an Abstract Syntax Tree (AST) from tokens
•**AST:** Tree structure representing code logic and flow
•**Interpreter:** Walks the AST and executes instructions
•**CLI/REPL (Optional):** Input JS code and get real-time output

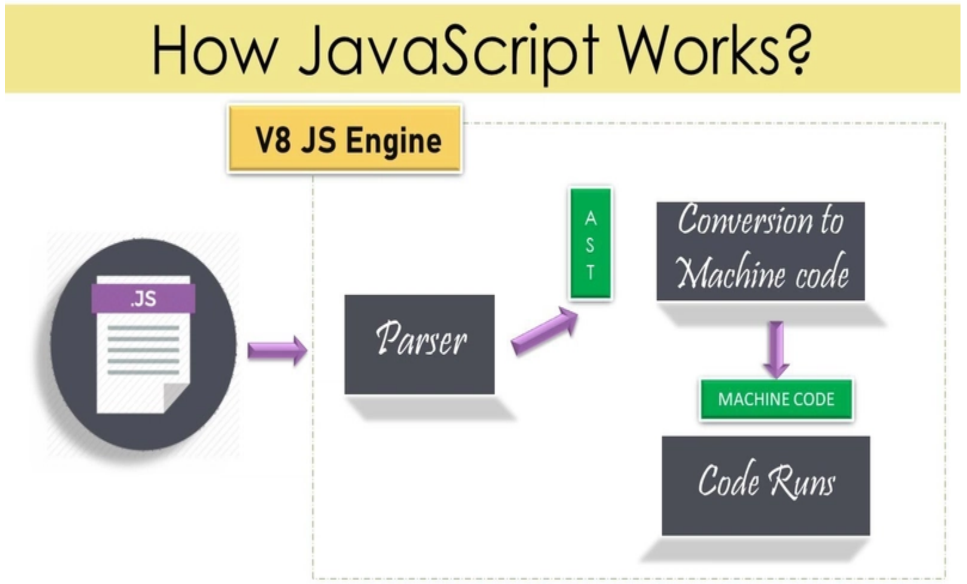
🛠 **Supported JS Features:**

•Variable declarations (let, const)
•Arithmetic & logical expressions (+, -, *, /, &&, ||)
•Conditional statements (if, else)
•Loops (while, for) *(optional)*
•Functions and block scope *(optional/advanced)*

How Javascript Engine's Work?



WORKFLOW /ARCHITECTURE



- **Workflow-arch – Mini JavaScript Engine**
- **Source Code Input**
- Raw JavaScript-like code is entered by the user (via file or CLI).
- **Tokenizer (Lexer)**
- Breaks the code into tokens (keywords, identifiers, operators, literals, etc.).
- **Parser**
- Uses grammar rules to convert tokens into an **Abstract Syntax Tree (AST)**.
- **AST Generation**
- AST is built to represent the structure and flow of the program.
- **Interpreter**
- Traverses the AST and executes code node-by-node (evaluates expressions, handles variables, etc.).
- **Environment / Scope Management**
- Tracks variable values and scopes during interpretation.
- **Output / Result**
- Final result is displayed (console output, return values, or errors).

ROLE AND RESPONSIBLITES OF GROUP MEMBERS

1) Dhruv Gangwar

a) Responsibilities:

b) Build the tokenizer (converts source code string into tokens)

c) Handle keywords, numbers, strings, punctuation, and operators

d) Write unit tests for various token types

e) Deal with edge cases (e.g., whitespace, comments)

f) Skills used: string manipulation, regex, attention to detail

2. Anjali Lohani

a) Responsibilities:

b) Take tokens and convert them into an Abstract Syntax Tree (AST)

c) Implement recursive descent parsing or similar parsing strategy

d) Design AST node types (e.g., Program, VariableDeclaration, BinaryExpression)

e) Ensure it supports correct syntax for expressions, if/while, etc.

f) Skills used: recursion, tree structures, syntax design

ROLE AND RESPONSIBLITES OF GROUP MEMBERS

3)Isha joshi
a) Traverse the AST and evaluate code

b) Implement logic for variable storage (environment/scope)

c) Handle control flow: if, while, basic operations

d) Optimize evaluation speed and error messages

e) Skills used: logic flow, evaluation strategy, JS runtime concepts

4)Aditya Chauhan
a) Combine lexer + parser + interpreter into a working pipeline

b) Build a CLI or REPL to run test code snippets

c) Write test cases covering valid and invalid inputs

d) Optional: work on advanced features (functions, return, etc.)

e) Documentation and presentation prep

f) Skills used: integration, testing, UI (CLI), optional advanced features

THANK YOU!