Session 0

0. Understanding How Computers Work

Before we dive into coding, let's get one thing clear:

> A computer doesn't understand human language.
>
> It only understands **machine instructions** (1s and 0s).

So to talk to it, we need a **translator**.

0.1 Basics of a Computer

At the core, a computer can:

- Take **input** (e.g., from a keyboard)
- Give **output** (e.g., on a screen)

That's enough to build basic tools like a calculatorbut only **if** you can tell the computer **what to do**.

0.2 Analogy: Talking to a Computer

Let's say you move to China and want your neighbor to buy you milk — but they speak only Chinese.

You have two options:

👉 Learn Chinese

👉 Use a translator

Same with computers:

- Computers "speak" **machine code**
- You either learn machine code (no thanks)
- Or you use a **translator** — a programming language + tool that converts your logic into what the computer understands

---

0.3 What is Machine Code?

Machine code = 1s and 0s (binary).

It's the only language computers **natively understand**,

but it's **extremely hard** for humans to write.

So we use programming languages (like JS, Python, etc.) and let a **translator** convert our code into machine instructions.

0.4 So You Hire a Translator

That translator becomes your:

- **Interpreter** – translates your instructions **live**, line by line

  > Like Google Translate in a live conversation

- **Compiler** – translates your **whole message** at once and gives you the full translation

  > Like writing a letter, getting it fully translated, and then delivering it

0.5 Interpreter vs Compiler: Key Differences

Feature	Interpreter	Compiler
How it works	Line-by-line execution	Full translation first
Speed	Slower	Faster (after compile)
Feedback	Instant errors	Errors after compiling
Examples	JavaScript, Python	C, Go, Rust

0.6 When to Use What?

Use an **interpreter** (like JS or Python) when:

- You want to test and iterate quickly
- You're debugging code

- You’re building scripts or dynamic apps

Use a **compiler** (like C or Rust) when:

- You care about speed and performance
- You want to build optimized, secure code
- You don’t want people reading your source code (it's compiled to binary)

0.7 Why Did We Choose JavaScript?

We’re learning JavaScript because:

- ⚡ It's **fast** (uses Just-In-Time compilation)
- 🌍 It **runs everywhere** (browsers, backend, mobile)
- 💬 It has a **huge community** and great support

Whether you want to make a game, a website, or a backend API, JS can handle it.

SSSSO Vidyakshina 2025