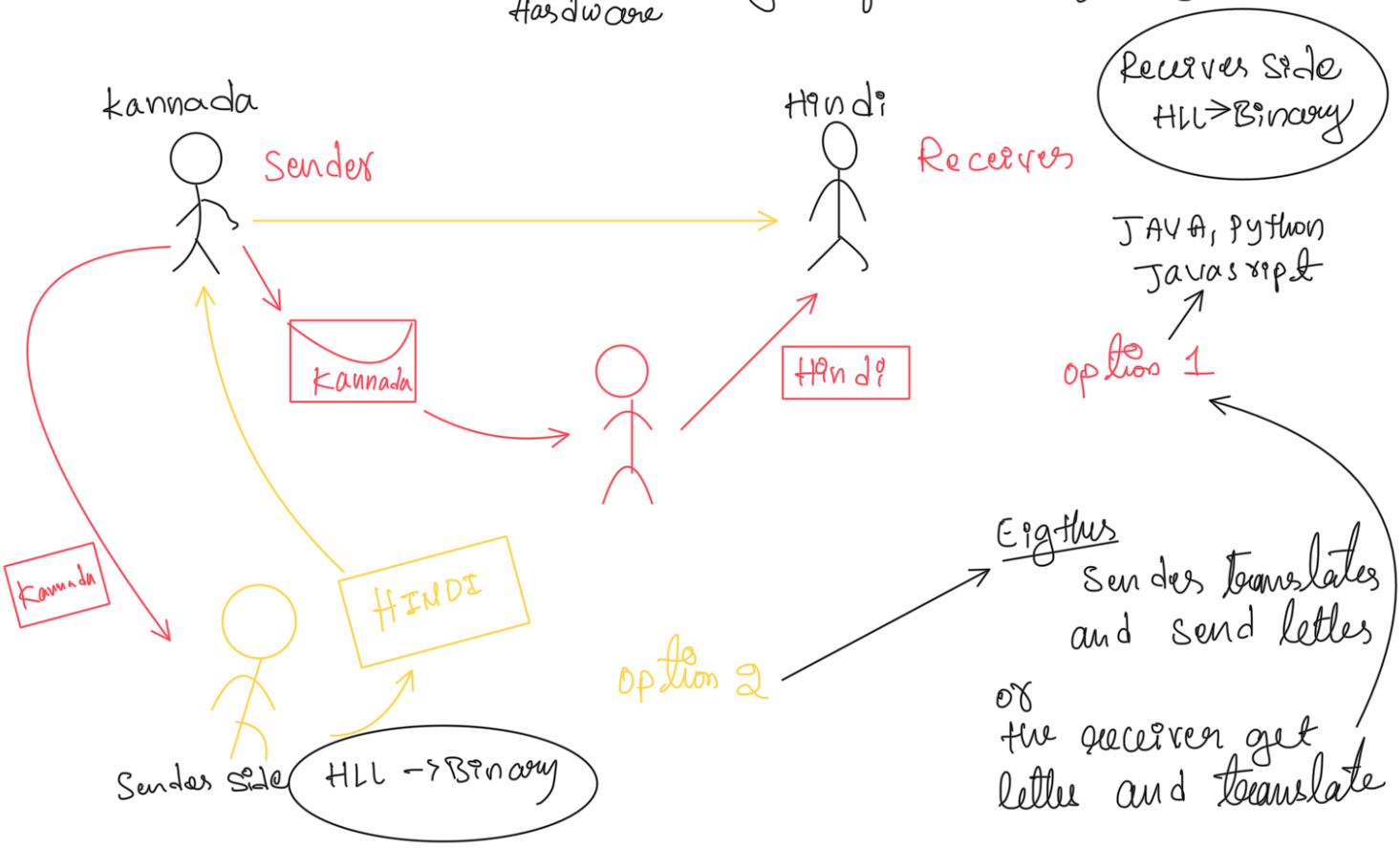
Write once & Run Anywhere

DAY 7
17/07/2025

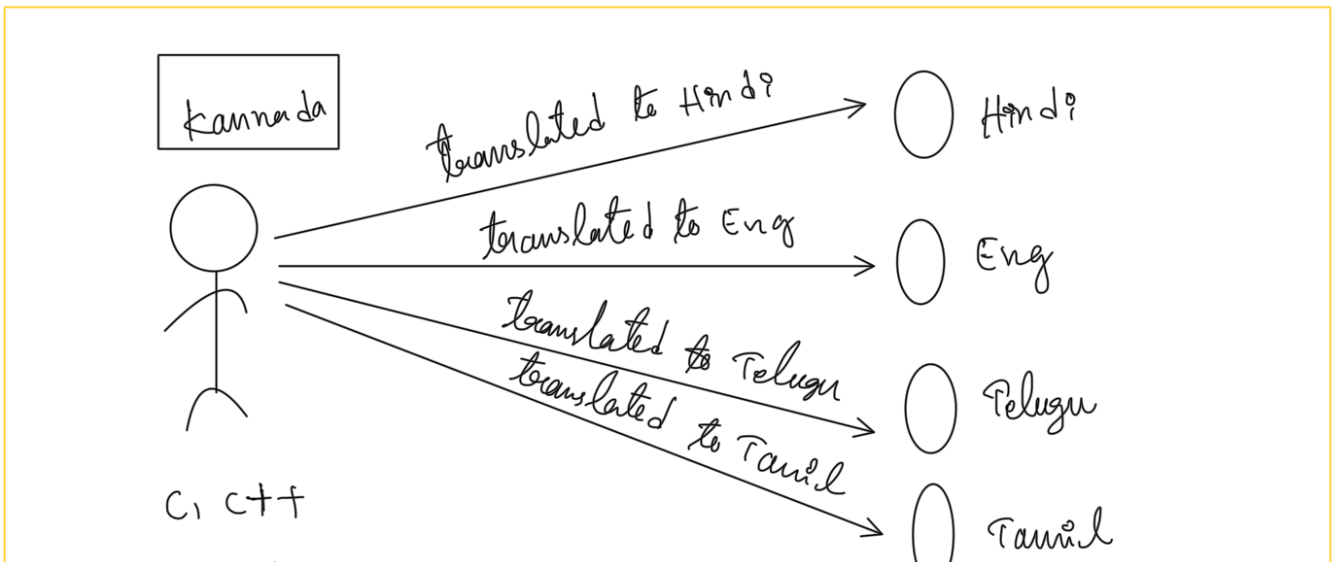
Java
Python
JavaScript

This 3 Languages code can Be Run in any type of operating systems. & Hardware

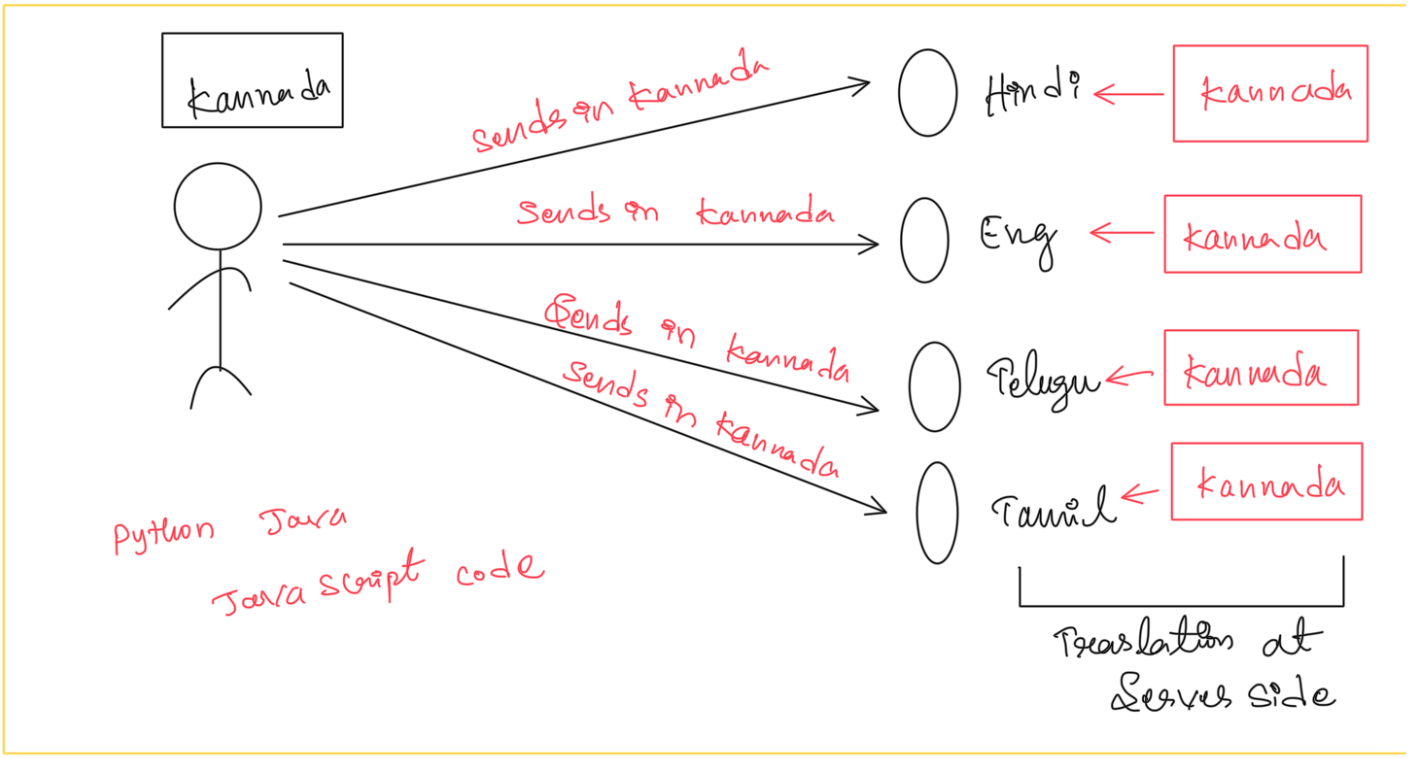


option 1 High level Language to Binary Conversion Happens at Receiver Side.

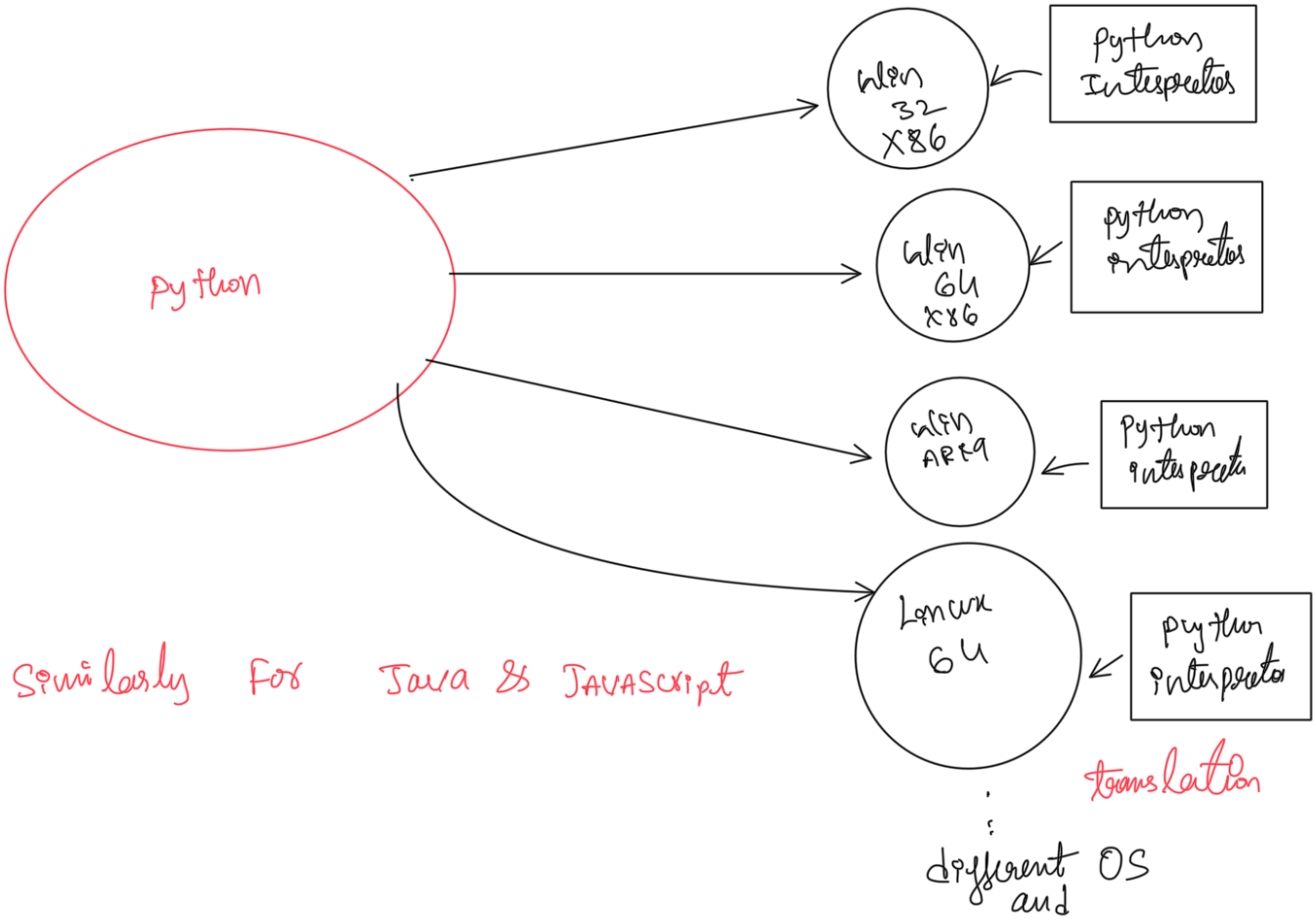
option 2 High level Language to Binary Conversion Happens at Sender Side.



Translation happens on Server Side

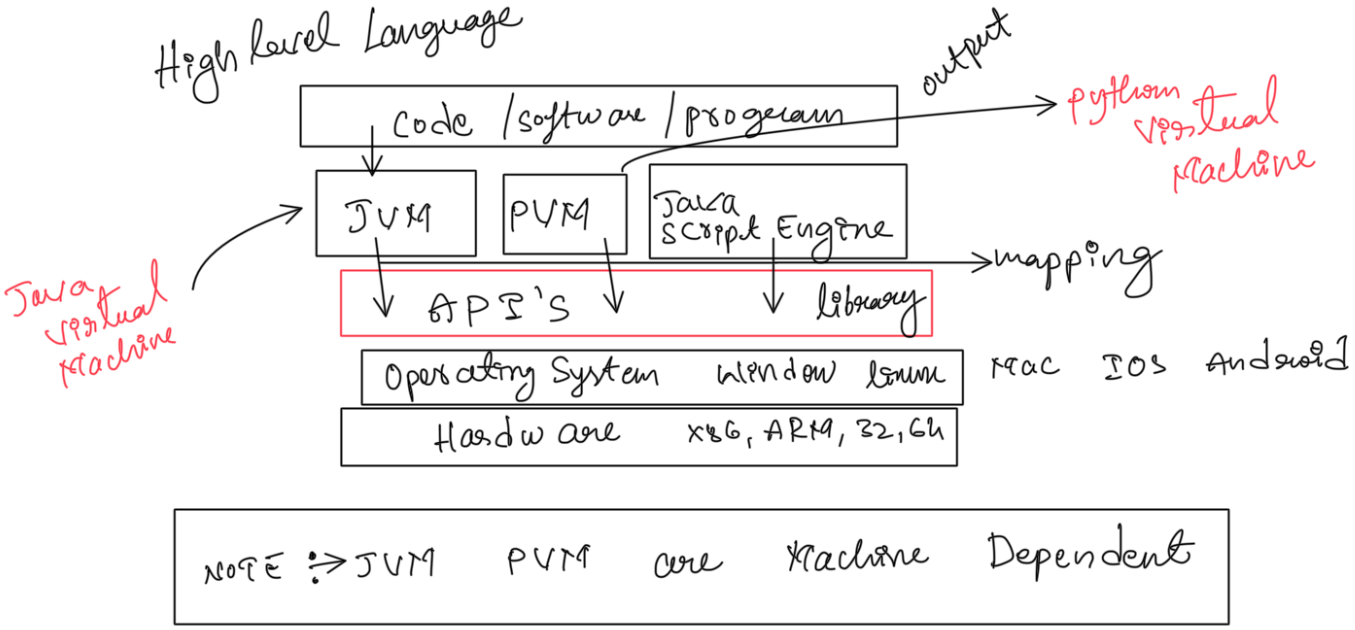


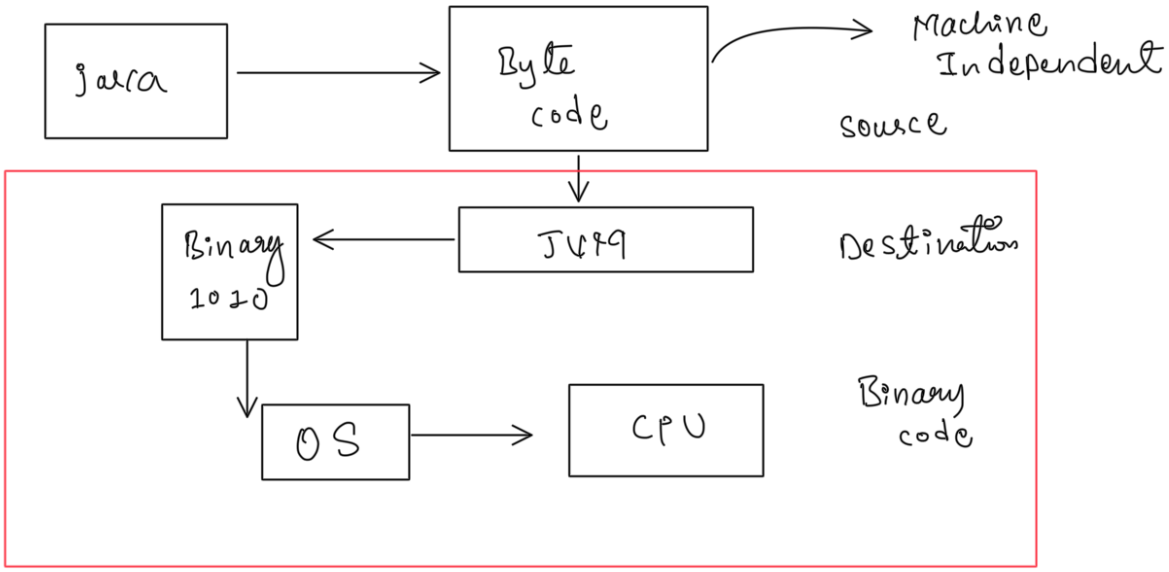
NOTE : In Both the Cases we Need translators



CPU

* This Helps to Achieve Write Once Run Anywhere

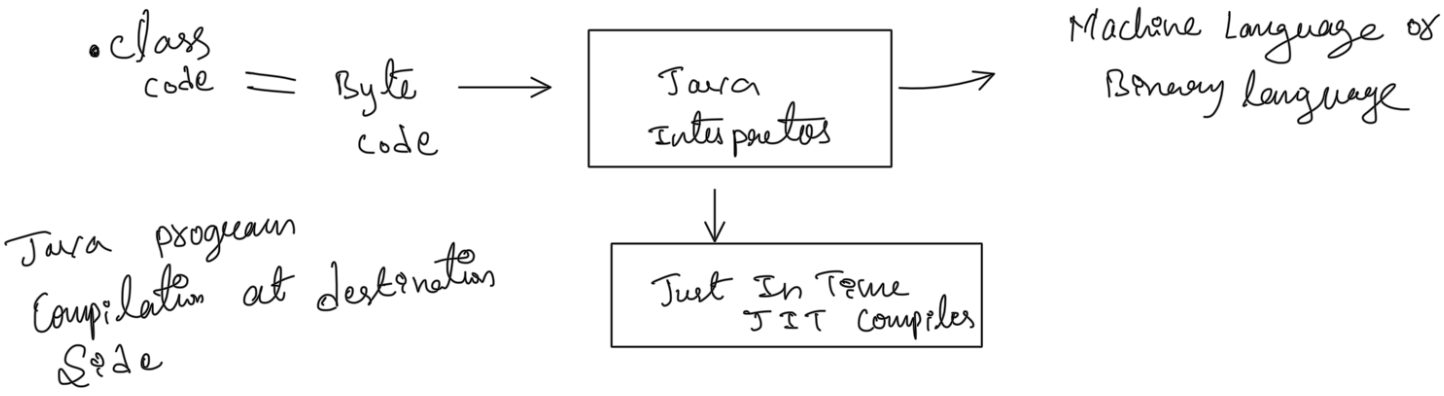


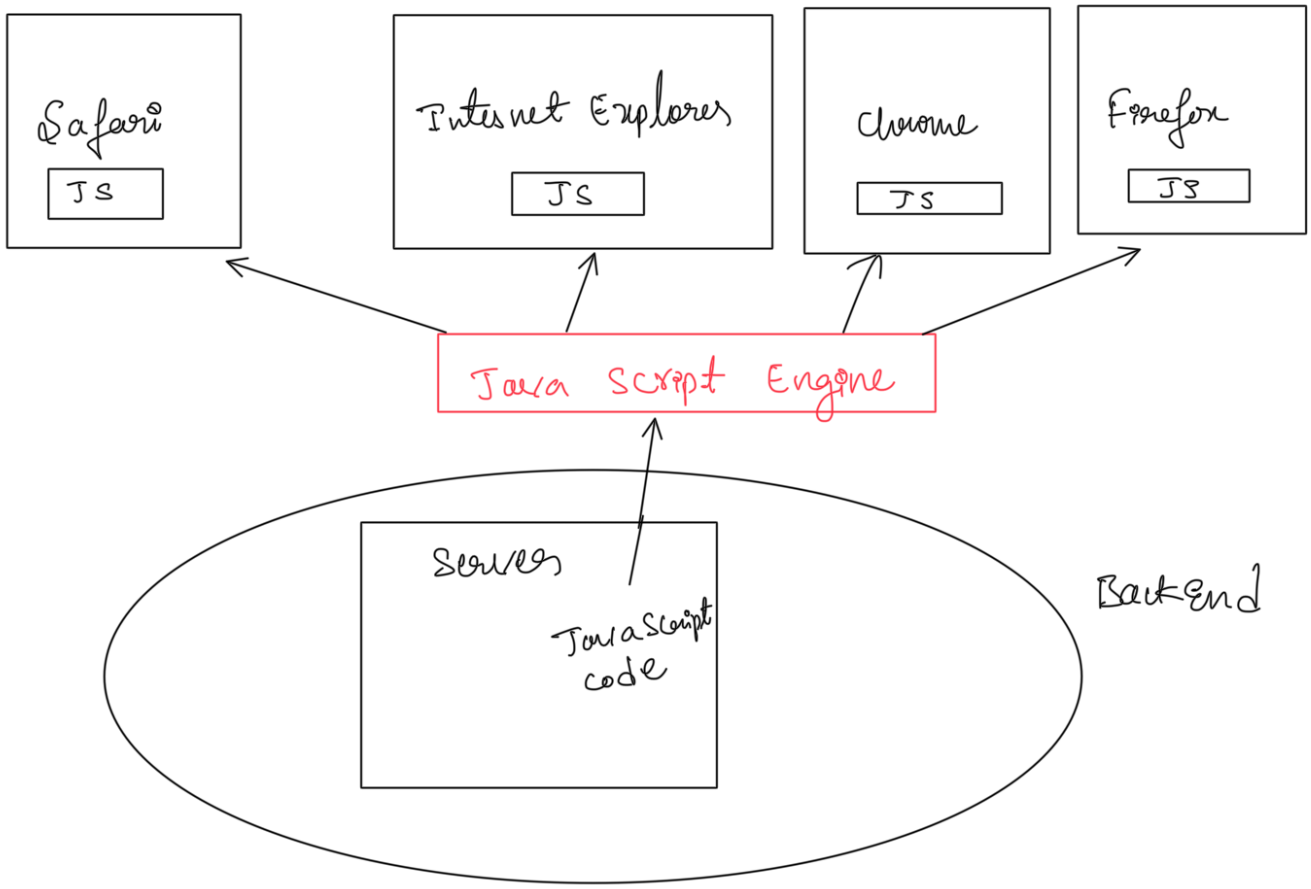


But we made it slower this code conversion at destination decrease speed of program execution

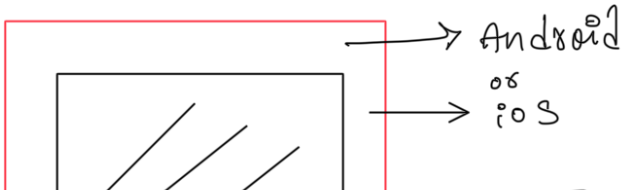
High performance softwares, device drivers, Networking these all type of softwares written in C, C++

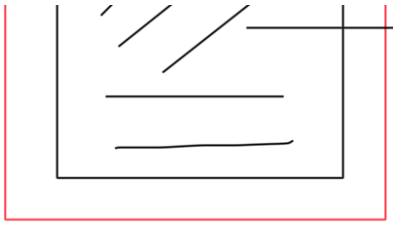
language. (Bcze these languages compiled first)





Java script Engine Created this Write once Run Anywhere Revolution first.





→ JavaScript

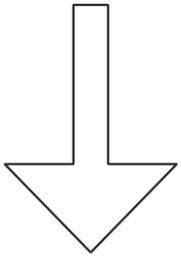
80% of code in JS

REACT framework made JavaScript very famous.

REACT

The application Built using React can Support on POS Android Browsers

Java Script was Running inside the Browsers

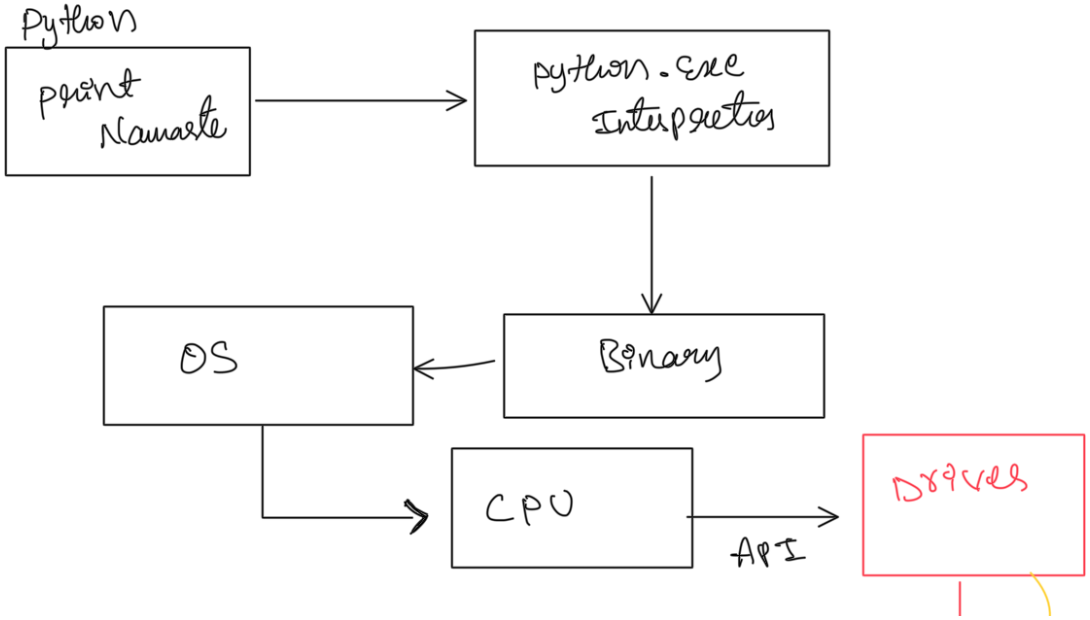


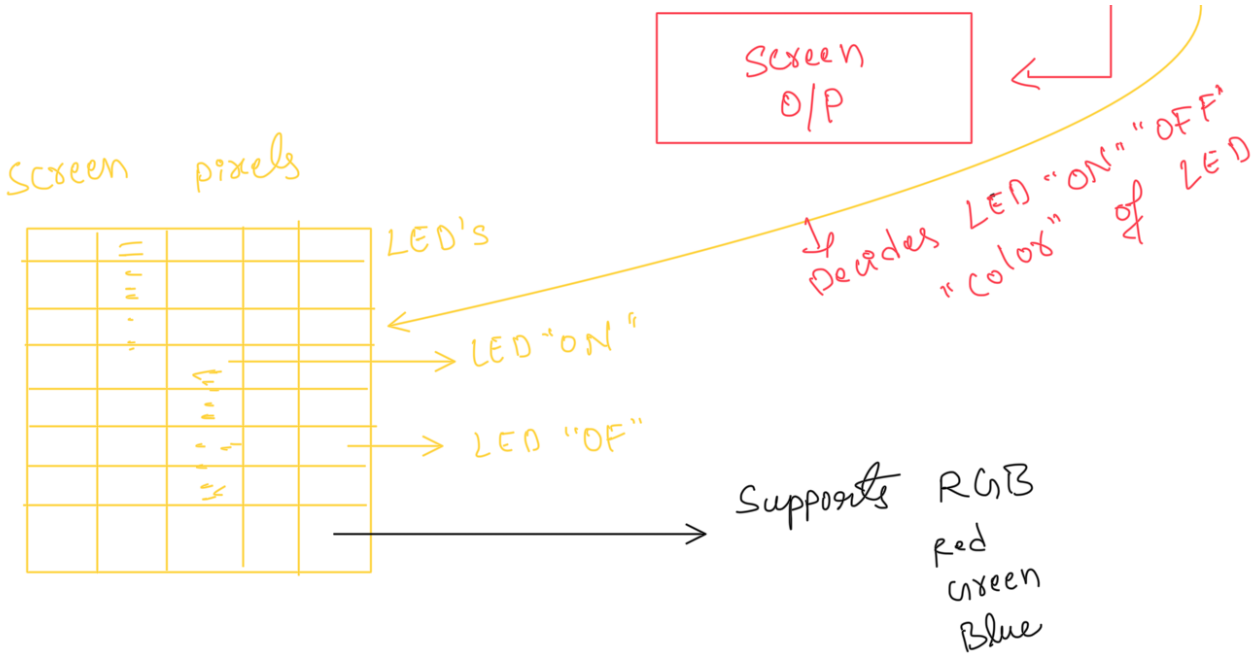
Node.js made JS to run outside Browsers (servers)

Node.js

Python, JAVASCRIPT Languages Become Very Versatile Languages Become popular.

Due to availability of open Source Libraries.




Screen
O/P
Screen pixels
LED's
Decides LED "ON" "OFF" "Color" of LED
LED "ON"
LED "OF"
Supports RGB
Red
Green
Blue

Write Once, Run Anywhere – Lecture Notes

Introduction

we explore the **"Write Once, Run Anywhere"** (WORA) programming paradigm — a concept that revolutionized the software industry by enabling developers to write code once and run it on multiple platforms without rewriting.

The Problem With Traditional Languages (C/C++)

Platform Dependency

- Languages like **C/C++** require you to compile code specifically for the **target operating system (OS)** and **CPU architecture** (x86, ARM, 32-bit, 64-bit).
- Each device or OS (Windows, Mac, Linux, Android, iOS) needs a separate executable.
- This makes maintenance and delivery across platforms difficult and time-consuming.

Example:

Microsoft Teams has different builds for:

- Windows
- macOS
- Linux
- Android
- iOS

Even within Windows, differences in processor architecture require different binaries.

The WORA Vision

To avoid re-compiling for every platform, the idea was born:

> ***Write the program once, and run it anywhere.***

Languages like **Java, Python, and JavaScript** make this possible.

Analogy: Kannada to Hindi Translation

Imagine sending a letter written in Kannada to five friends who all speak different languages:

- Option 1: The sender translates the letter into each target language.
- Option 2: Each receiver uses a **translator** to read the letter.

WORA follows **Option 2** — install a translator (interpreter or virtual machine) on each platform.

How It Works: Virtual Machines and Interpreters

Components:

- **Hardware**: CPU (x86, ARM, 32-bit, 64-bit)
- **Operating System**: Windows, Mac, Linux
- **Translation Layer**:
    - **Java**: JVM (Java Virtual Machine)
    - **Python**: PVM (Python Virtual Machine)
    - **JavaScript**: JS Engine (e.g., V8 in Chrome)

These virtual machines handle:

- OS-specific API mapping
- Platform-dependent execution

Architecture

```
[ Application Code (Java/Python/JS) ]
                  ↓
[ Bytecode / Source Code ]
                  ↓
[ Virtual Machine (JVM/PVM/JS Engine) ]
                  ↓
[ OS API ]
                  ↓
[ CPU Hardware ]
```

Language-Specific Details

Java

- Compile `.java` files → `.class` (bytecode)
- JVM converts bytecode → binary at runtime
- **Java Compiler**: High-level → bytecode
- **JVM**: Bytecode → machine code

```
class Hello {
  public static void main(String[] args) {
    System.out.println("Hello, World");
  }
}
```

Python

- Interpreted at runtime by PVM
- Source code sent directly (or `.pyc` compiled form)

```
print("Hello from Python")
```

JavaScript

- Sent as source code to browser
- JS engine interprets and executes it on the fly

```
console.log("Hello from JS");
```

Performance Trade-off

Drawback of WORA:

- Translation happens **during execution** → **slower** than native binaries
- C/C++ is still used in **performance-critical** systems (rockets, drivers, banking systems)

Optimization

- **JIT (Just-In-Time) Compilation**: Repeated functions are compiled into binary and cached to improve performance.
- Used in both Java and Python.

Real-World Application

JavaScript in Browsers

- Every browser (Chrome, Firefox, Safari) includes a JS Engine.
- Code sent from servers is executed in the browser using this engine.
- **Minification and Obfuscation** reduce code size and readability.

Mobile App Frameworks

- **React Native** allows JavaScript code to run on Android and iOS with minimal platform-specific code.

Summary

- Traditional languages like C/C++ are platform-specific.
- Java, Python, and JavaScript allow **WORA** through VMs and interpreters.
- Trade-off: performance is reduced but cross-platform compatibility improves.
- JIT compilers and efficient engines help bridge performance gaps.
