1. Command Prompt (CMD)

- **Definition**: The **Command Prompt** (also known as **cmd.exe**) is the default command-line interpreter on Windows OS.
- **Use in VS Code**: VS Code integrates with the system's Command Prompt to allow basic file and project management operations.
- **Common Uses**:
  - Running .exe files
  - Navigating directories (cd, dir)
  - Running batch scripts (.bat)
  - Managing Windows environment variables
- **Strengths**:
  - Simple and fast
  - Installed by default on Windows
- **Limitations**:
  - Limited scripting capabilities compared to PowerShell or Git Bash
  - No built-in support for Unix commands

---

2. Git Bash

- **Definition**: Git Bash is a terminal provided by **Git for Windows**, which provides a **Bash emulation** used to run Git commands and Unix-style command-line utilities.
- **Use in VS Code**: You can select Git Bash as your default terminal to use Linux-style commands inside VS Code on Windows.
- **Common Uses**:
  - Running Git commands (git clone, git push, etc.)
  - Using Unix commands (ls, touch, cat, grep)
  - Scripting using shell scripts (.sh)
- **Strengths**:
  - Supports most Linux/Unix commands

    - Great for Git workflows
    - Better scripting capability than CMD
- **Limitations**:
    - Might be confusing for pure Windows users
    - Slightly heavier than CMD

---

3. PowerShell

- **Definition**: PowerShell is a **task automation and configuration management** framework developed by Microsoft, featuring a powerful command-line shell and scripting language.
- **Use in VS Code**: PowerShell is integrated deeply into Windows and can be selected as the terminal in VS Code. It's also extensible and used for scripting automation tasks.
- **Common Uses**:
    - Running Windows system and admin tasks
    - Advanced scripting (.ps1 files)
    - Managing files, services, registry, and system processes
- **Strengths**:
    - Powerful scripting capabilities (object-based, unlike string-based CMD)
    - Deep Windows integration
    - Useful for DevOps tasks
- **Limitations**:
    - Slight learning curve for beginners
    - Less intuitive for Git or Linux users

---

4. JavaScript Debug Terminal

- **Definition**: A special terminal in VS Code that allows you to run Node.js scripts with **debugging enabled** automatically.
- **Use in VS Code**: Created automatically or manually via the Command Palette → Create JavaScript Debug Terminal. Used for running and debugging JavaScript apps.

- **Common Uses**:
  - Debugging Node.js applications
  - Setting breakpoints, viewing variables, and call stacks
  - Stepping through JavaScript code
- **Strengths**:
  - Built-in integration with VS Code debugger
  - Useful for developers writing backend in Node.js
  - Helps in real-time code debugging
- **Limitations**:
  - Only for JavaScript/Node.js projects
  - Not suitable for general terminal usage