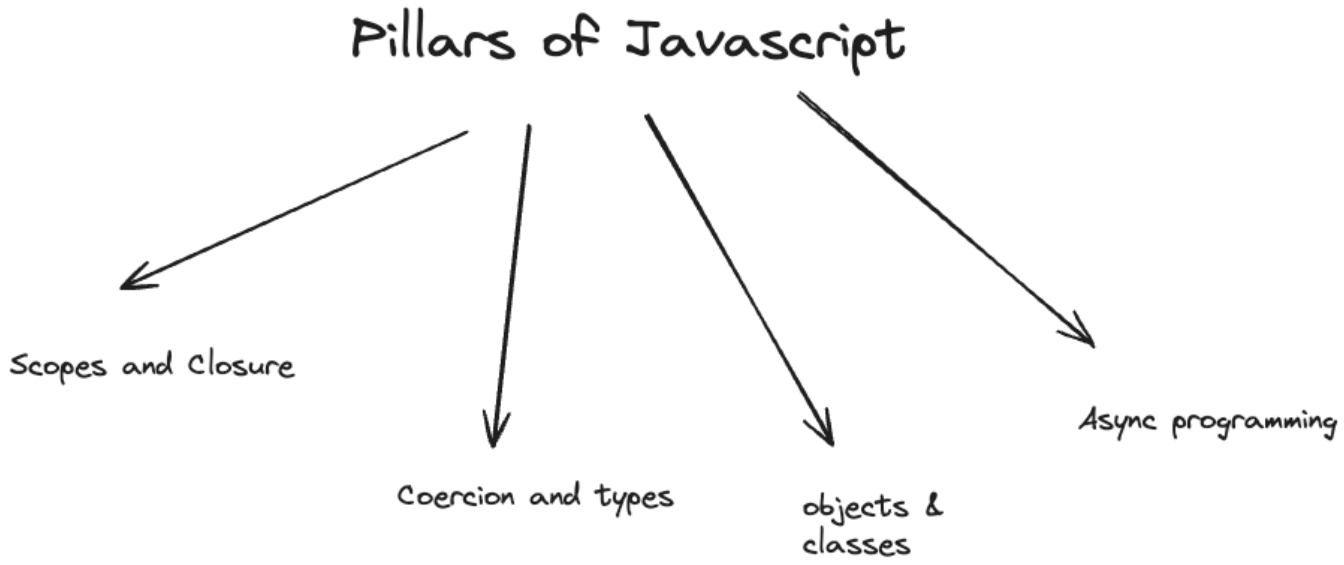
Javascript Pillars

Mainly we have 4 pillars

- Scopes and Closures
- Coercion and types
- Async Programming
- Objects and classes



Scopes

Scopes as a word is closely related to vision i.e. what particular part you can see and what you cannot see.

In Javascript, we use the concept of scopes to figure out where a variable or function is accessible / visible.

But how exactly the scoping mechanism in JS works ?

Note:

Scoping mechanism in JS is very very different than other languages (Java , C++, python etc). So don't mix the concepts of JS with other languages.

To understand the whole scoping mechanism, we need to understand the concept of compilation and interpretation in programming languages.

Most of the languages are divided into two categories:

- Compiled language
- Interpreted language

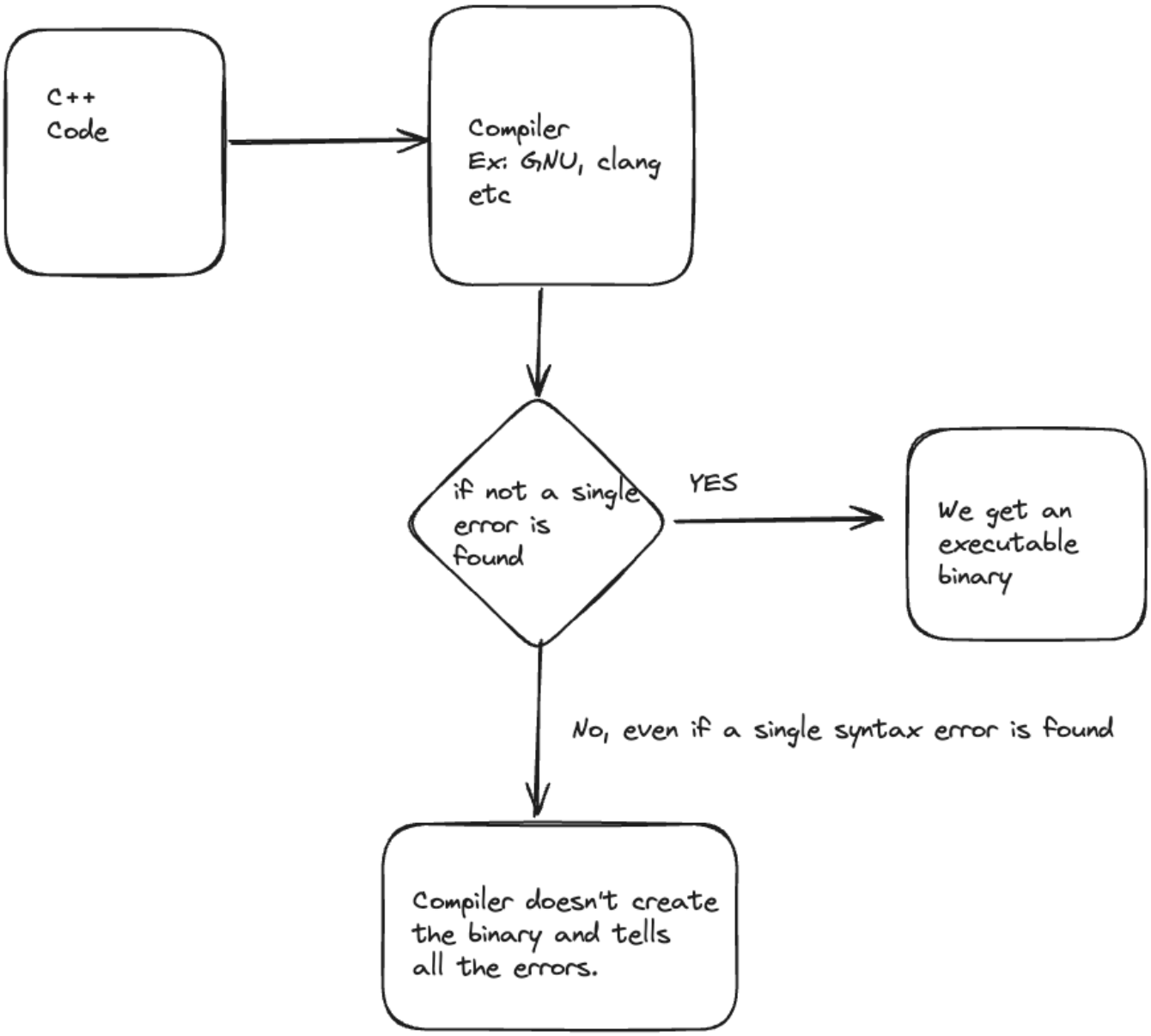
The question that should come to your mind is, whether JS is compiled or interpreted ?
To understand this let's take a deeper dive into compilation and interpretation.

Compiled Languages

Example of compiled languages are: **C** , **C++** etc.
To run the code of any compiled programming language we need to use another software called as Compiler.

Compiler takes the whole code, analyses it for errors, if there are no errors then it will give us an executable binary, but if there is any single error, then nothing will be added to the executable, in fact no executable will be made and compiler will throw the errors which are present in the code.
Interesting fact is that all the errors are told at once.
If let's say the first 100 lines of code are correct and the 101th line has an issue, still nothing will

be executed.



Interpreted Languages

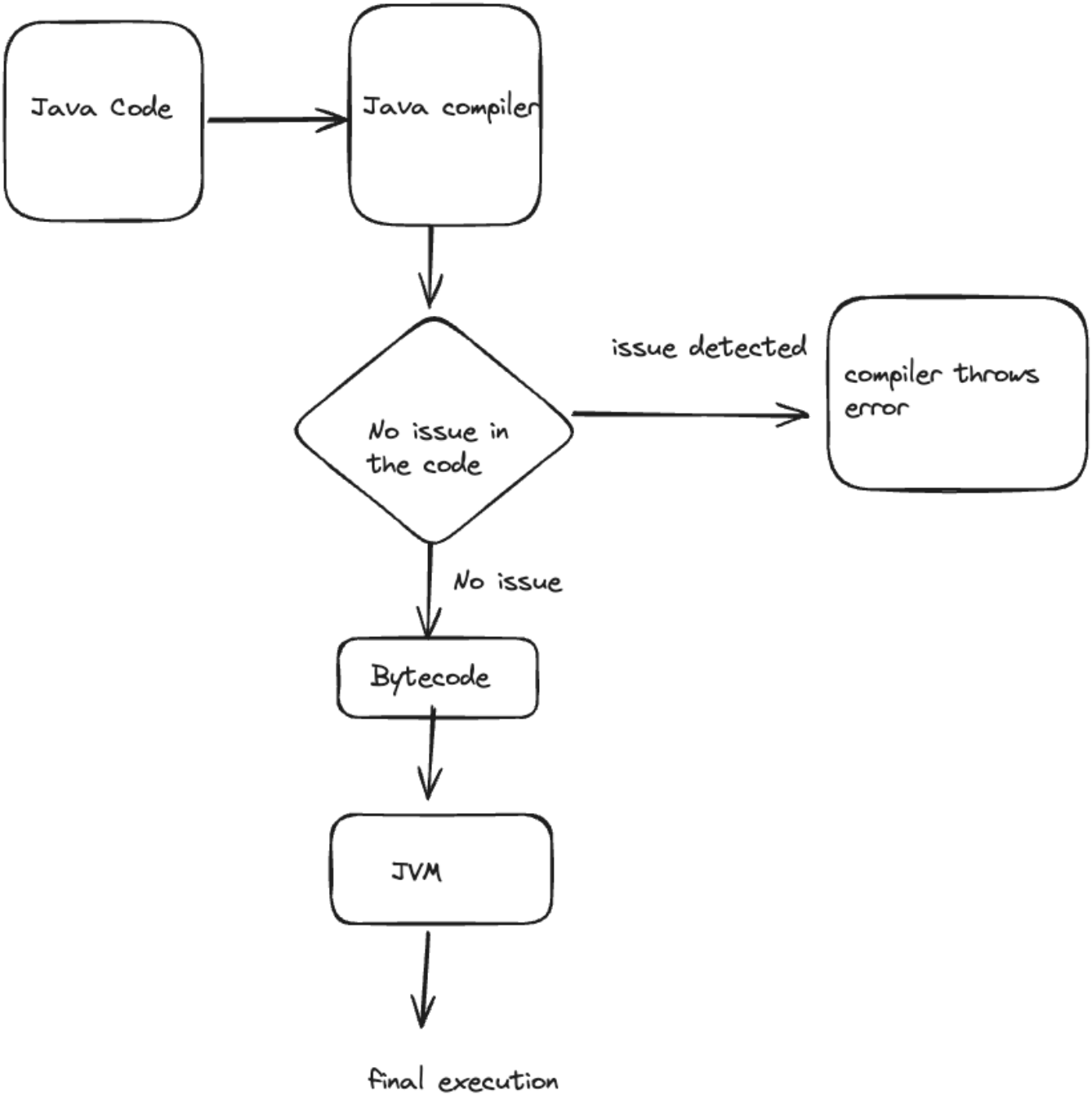
Pure interpreted languages exist for example: Bash
These are those languages, which execute our code directly, without reading / analysing the whole code prior. They execute code line by line, if any line has an error, then everything before it gets properly executed and the moment we detect the first error, execution stops.
If there is more code, after the error, nothing is executed post we encounter the error.

```
echo "hello world"
ls -l
excho 'hello world"";
echo "hello world"
```

```
[Running] bash "/Users/sanketsingh/Developer/Javascript-Intermediate/Scopes/test.sh"
hello world
total 0
-rw-r--r--  1 sanketsingh  staff   0 11 Jul 20:37 README.md
drwxr-xr-x  3 sanketsingh  staff  96 11 Jul 21:32 Scopes
/Users/sanketsingh/Developer/Javascript-Intermediate/Scopes/test.sh: line 5: unexpected EOF while
looking for matching `''
/Users/sanketsingh/Developer/Javascript-Intermediate/Scopes/test.sh: line 8: syntax error: unexpected
end of file
```

Hybrid Languages

There is kind of like a third category of languages which uses both compilation and interpretation for the final execution of code. Ex: **Java**, **JS**, **Python**, etc



Nature of JS

JS combines the process of compilation and interpretation. A lot of people think it is interpreted only but that's not true. How ?

```
console.log("wow wow");
function fun() {
    le r = uy;
}
```

The above piece of throws an error without executing `console.log` and prints nothing on the console and just says the error altogether.
This shows that JS is somehow analysing the whole prior and then only starting the execution.

Actually every JS code is executed in 2 phases:

- Phase 1 : Compilation and Scope Resolution phase
- Phase 2: Interpretation or execution phase