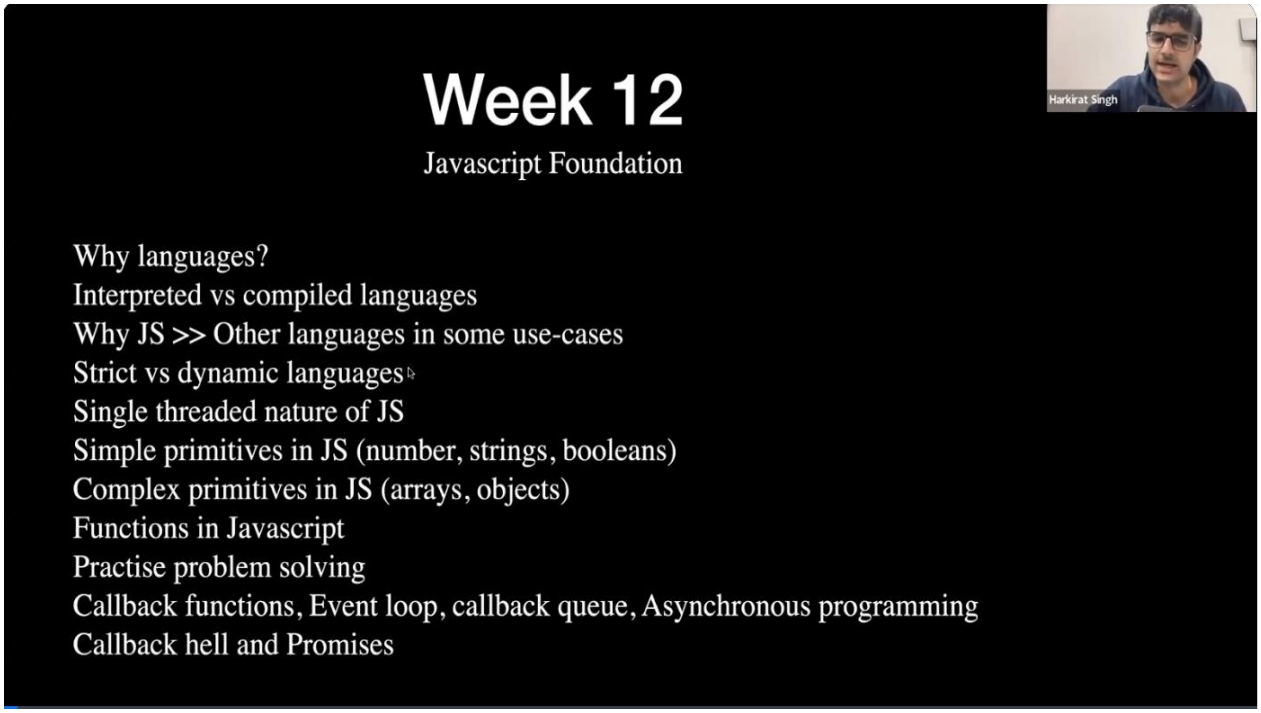
HARKIRAT DEVOPS

JS Foundations [1.2]

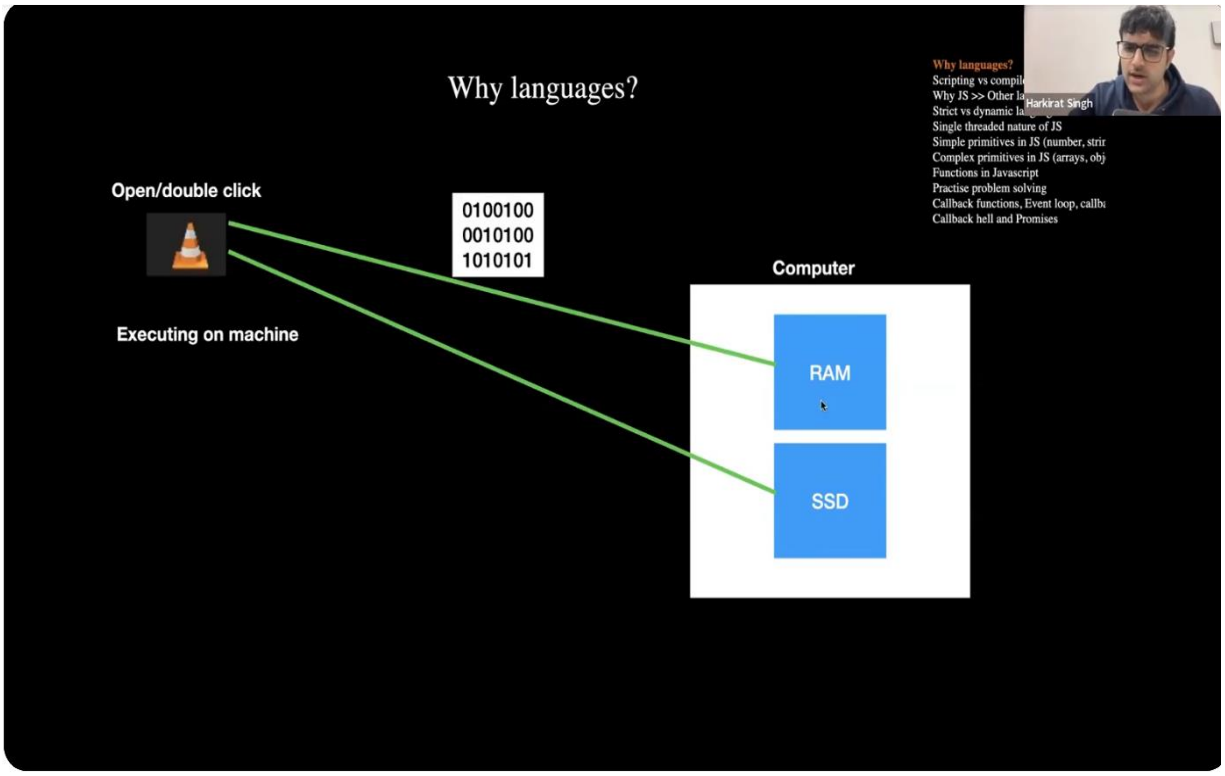


Our first question is why do we need languages?

Before that, we will learn what is RAM and what is SSD.

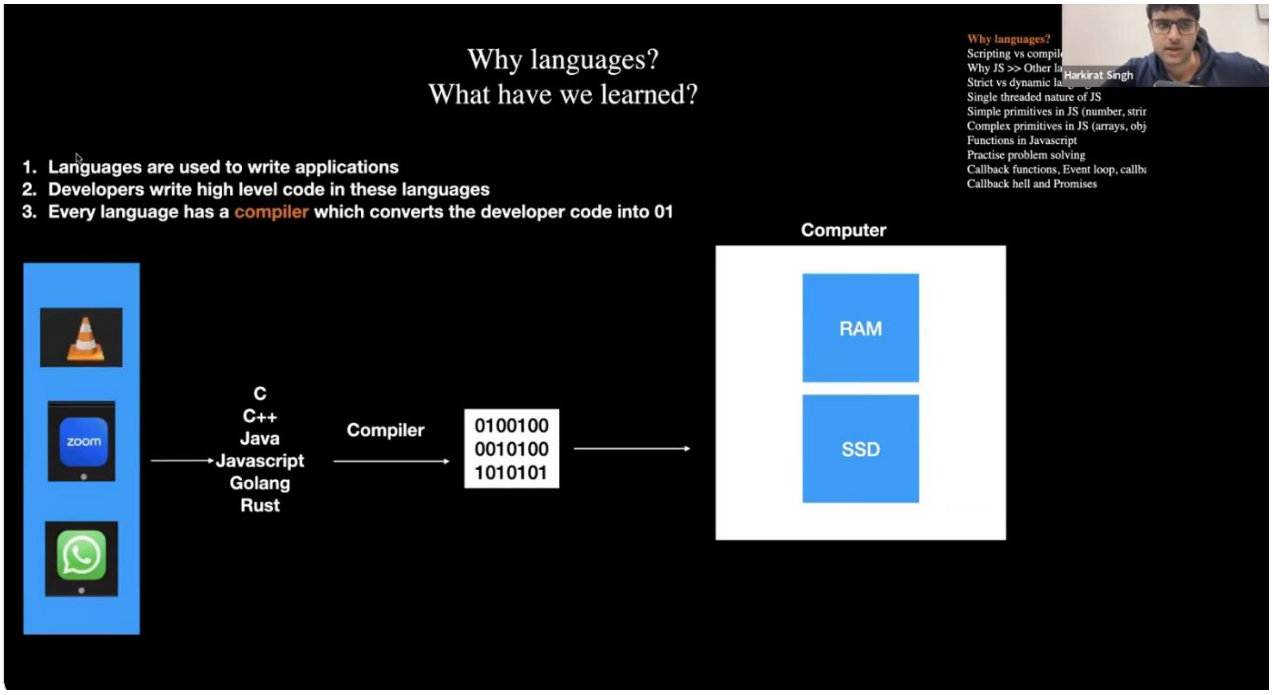
SSD is a hard drive of a PC, every movie, photo will be stored in SSD.

But when we run something, for example playing movies or having a Zoom call are resides in RAM.

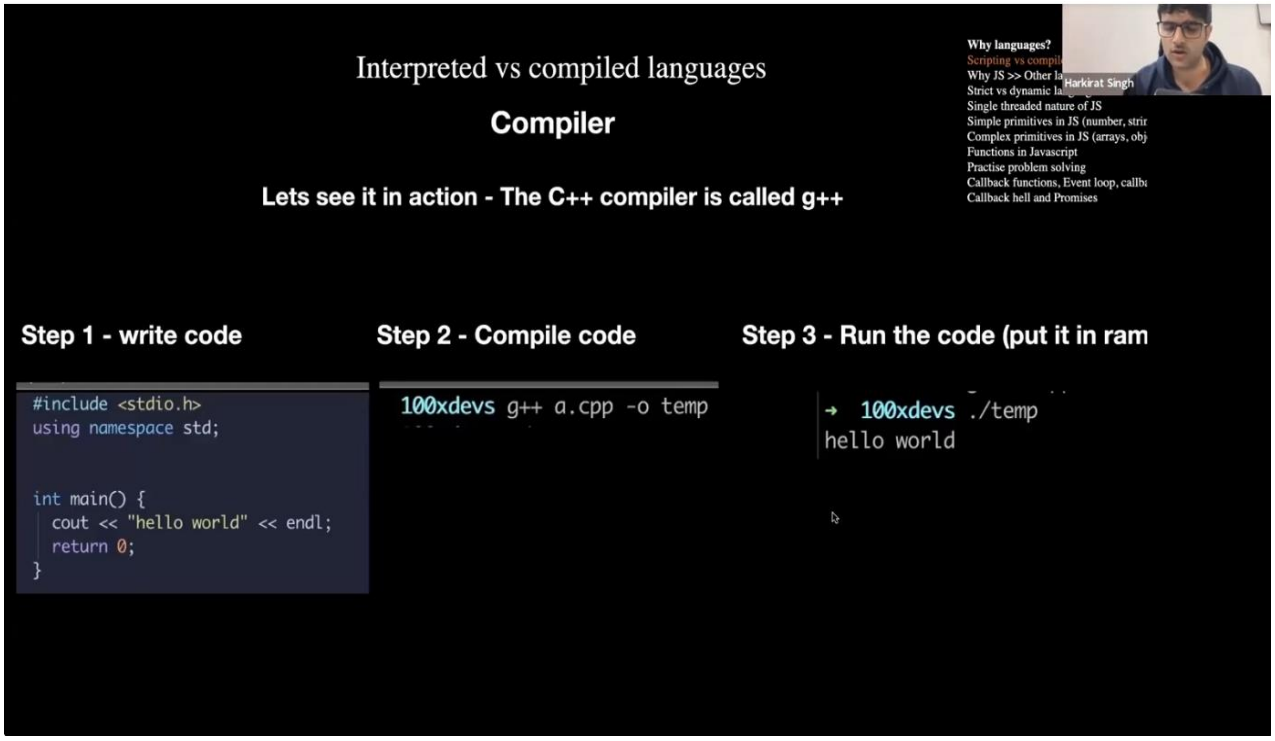


This music player was initially present in SSD, but when we double-clicked it, then it started running/playing and got stored(the thing that goes inside the RAM just the binary bits consisting of 0’s and 1’s) is inside the RAM.

Now, VLC is created by the use of Languages JS, C++, or any other language and then these languages are converted into 0’s and 1’s by the compiler so that this VLC can be understood by the machine/PC.

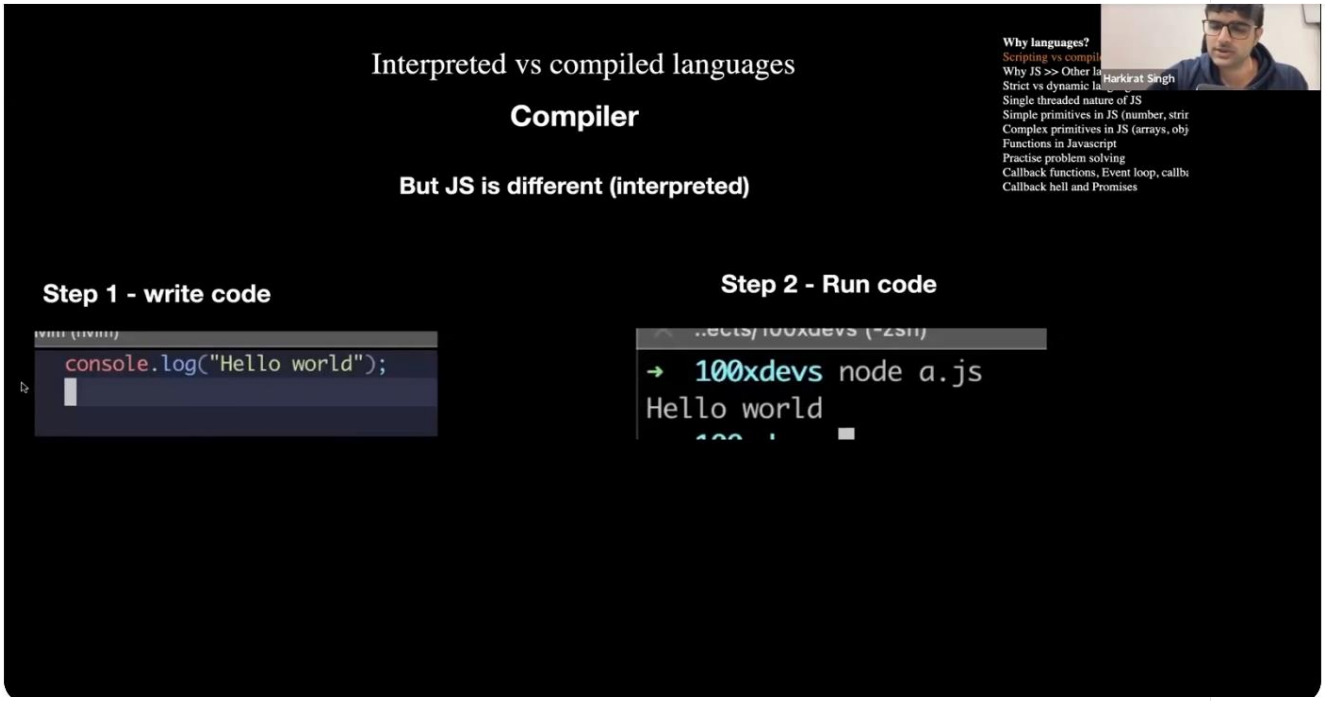


Compiled Vs Dynamic Language

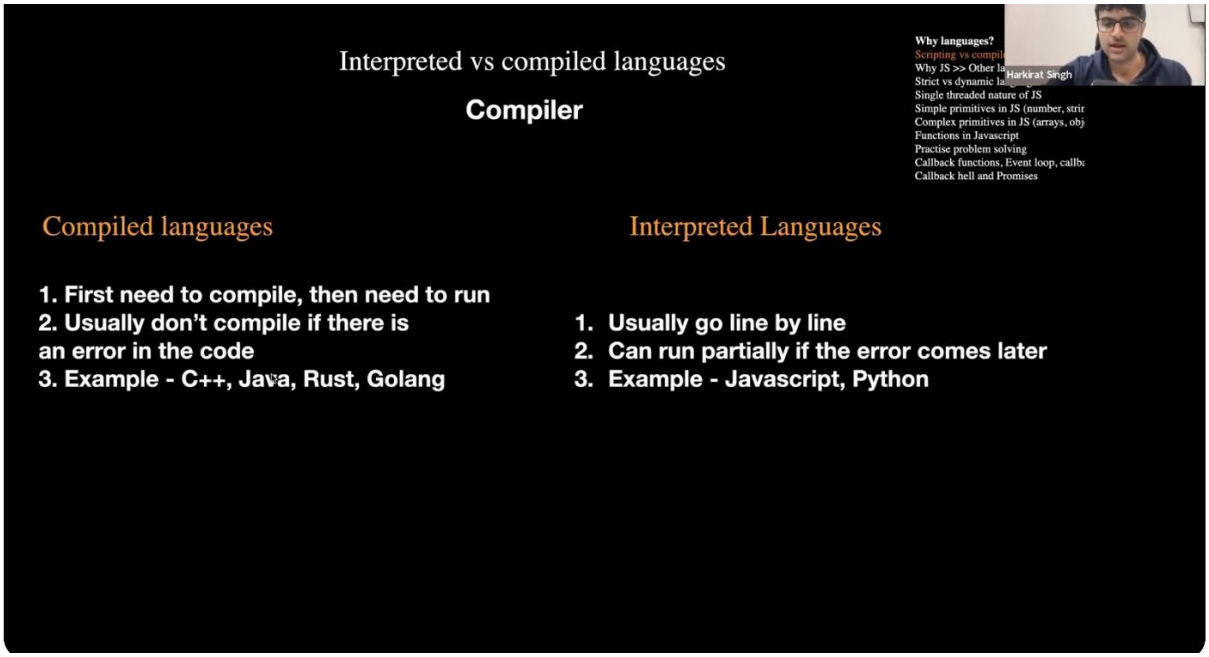


Compiled Language (C++): In compiled languages like C++, the source code is translated into machine code (binary code) before execution. This translation process is done by a compiler, which converts the entire source code into an executable file that can be directly run by the computer's processor. This executable file contains machine code specific to the target platform. Compilation typically involves several

steps such as preprocessing, compilation, assembly, and linking. Once compiled, the resulting binary code can be executed independently of the original source code.

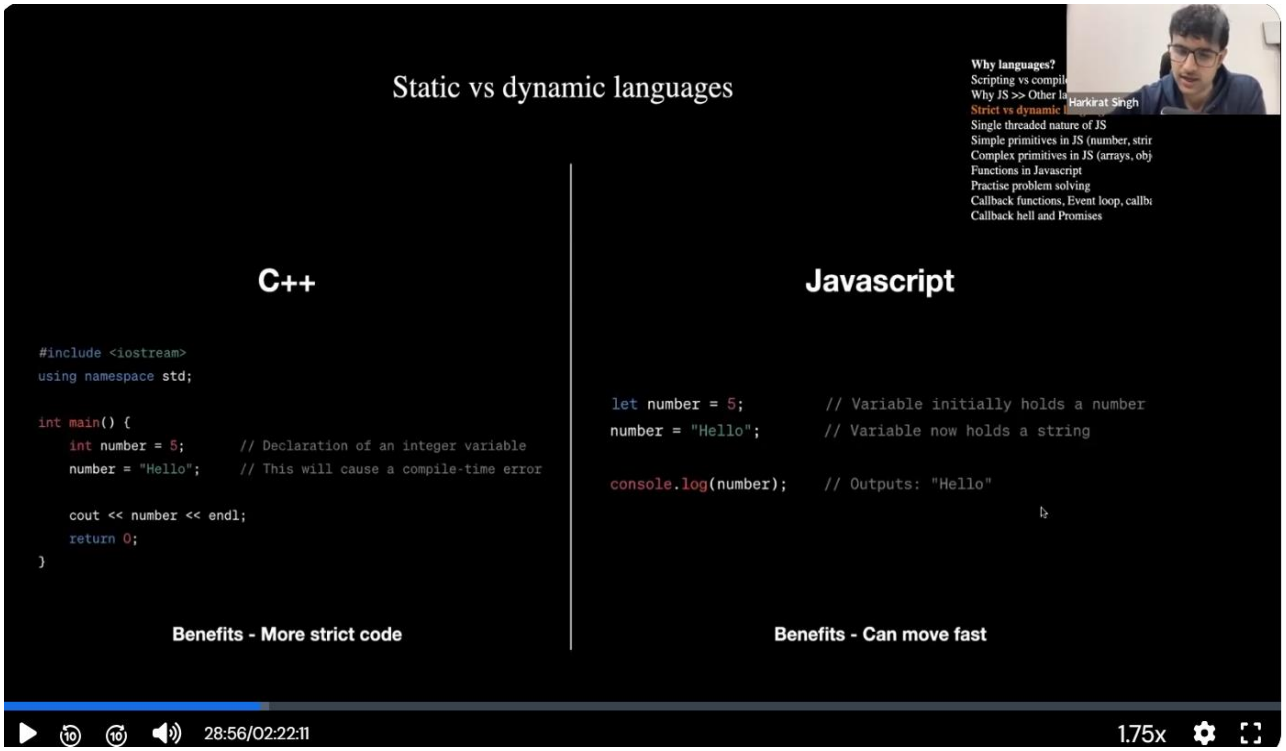


Interpreted Language (JavaScript): In interpreted languages like JavaScript, the source code is not translated into machine code before execution. Instead, the source code is directly executed by an interpreter line by line. When a JavaScript program is run, each line of code is read, parsed, and executed by the interpreter in real-time. This means that there is no separate compilation step, and the source code is translated into machine code at runtime. Interpreted languages are often platform-independent since they rely on the interpreter to execute the code, making them more flexible in terms of deployment.



Why JS is better than other Languages?

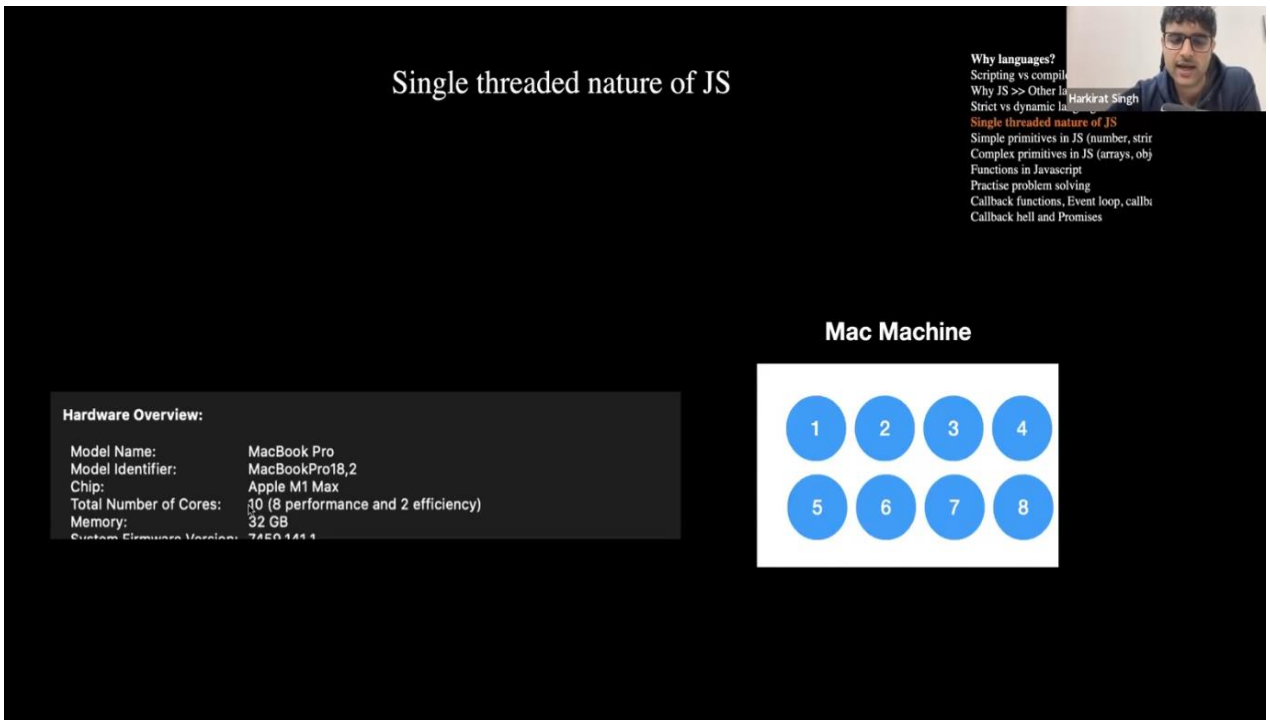
Static Vs Dynamic Language



Static Language (e.g., C++): In a static language like C++, the type of each variable is determined at compile-time and cannot change during runtime. This means that the type of every variable must be explicitly declared in the source code, and the compiler checks for type correctness before generating the executable code. Static languages often enforce strong type checking, meaning that type errors are caught at compile-time, reducing the likelihood of runtime errors related to type mismatches. Examples of static languages include C++, Java, and Rust.

Dynamic Language (e.g., Python): In a dynamic language like Python, the type of variables is determined at runtime, and variables can change their type during the execution of the program. This means that variables do not need to be explicitly typed in the source code, and their type can be inferred based on the value assigned to them. Dynamic languages typically perform type checking at runtime, allowing for more flexibility but potentially leading to runtime errors if types are incompatible. Examples of dynamic languages include Python, JavaScript, and Ruby.

Single Threaded Nature of JS

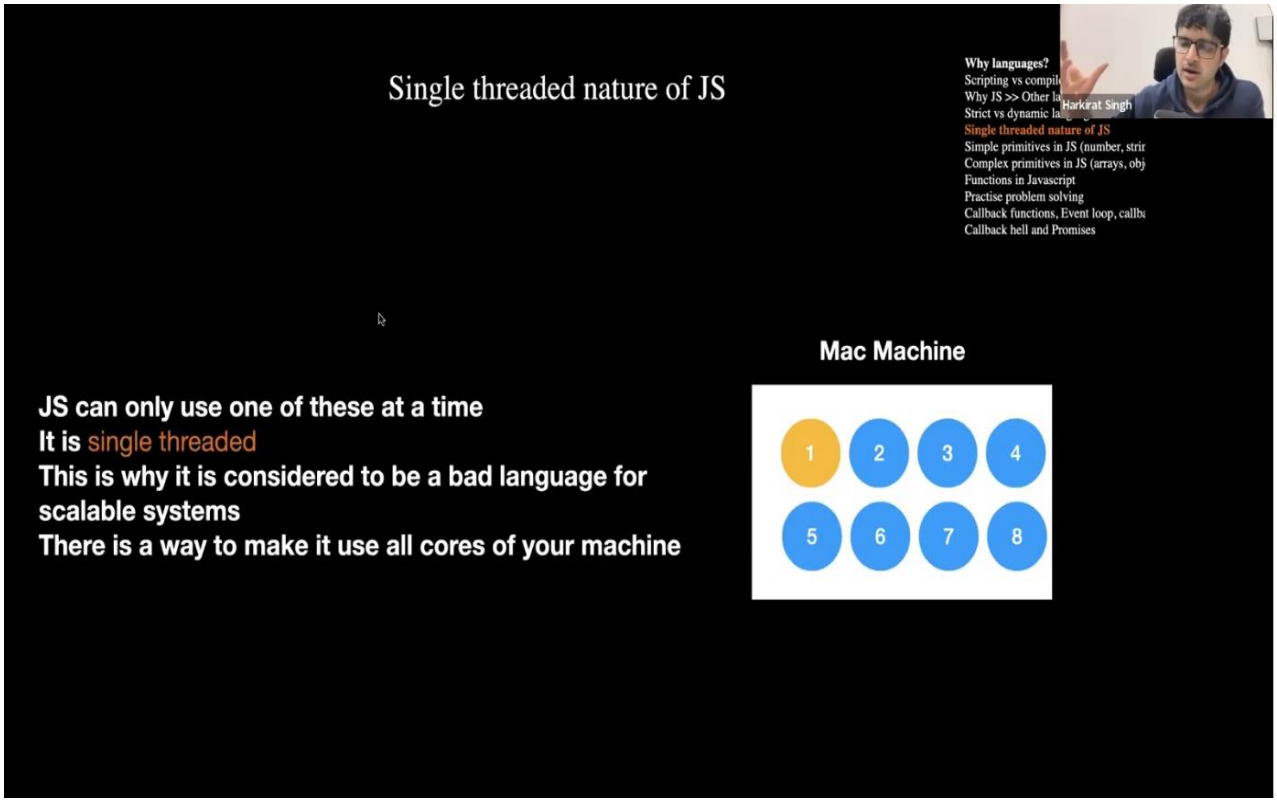


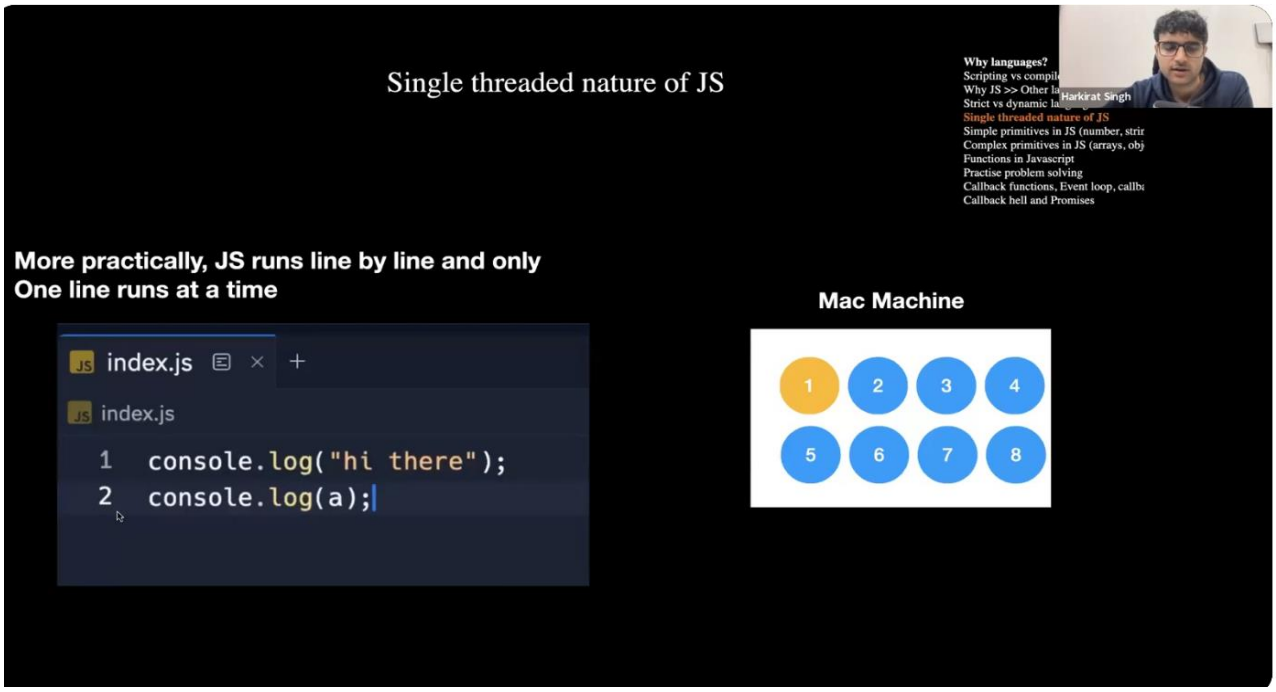
Each core will run a single process at a time, more the no. of cores we have more the no. of programs we can run at a time.

The single-threaded nature of JavaScript refers to how JavaScript code is executed within a single thread in the browser environment.

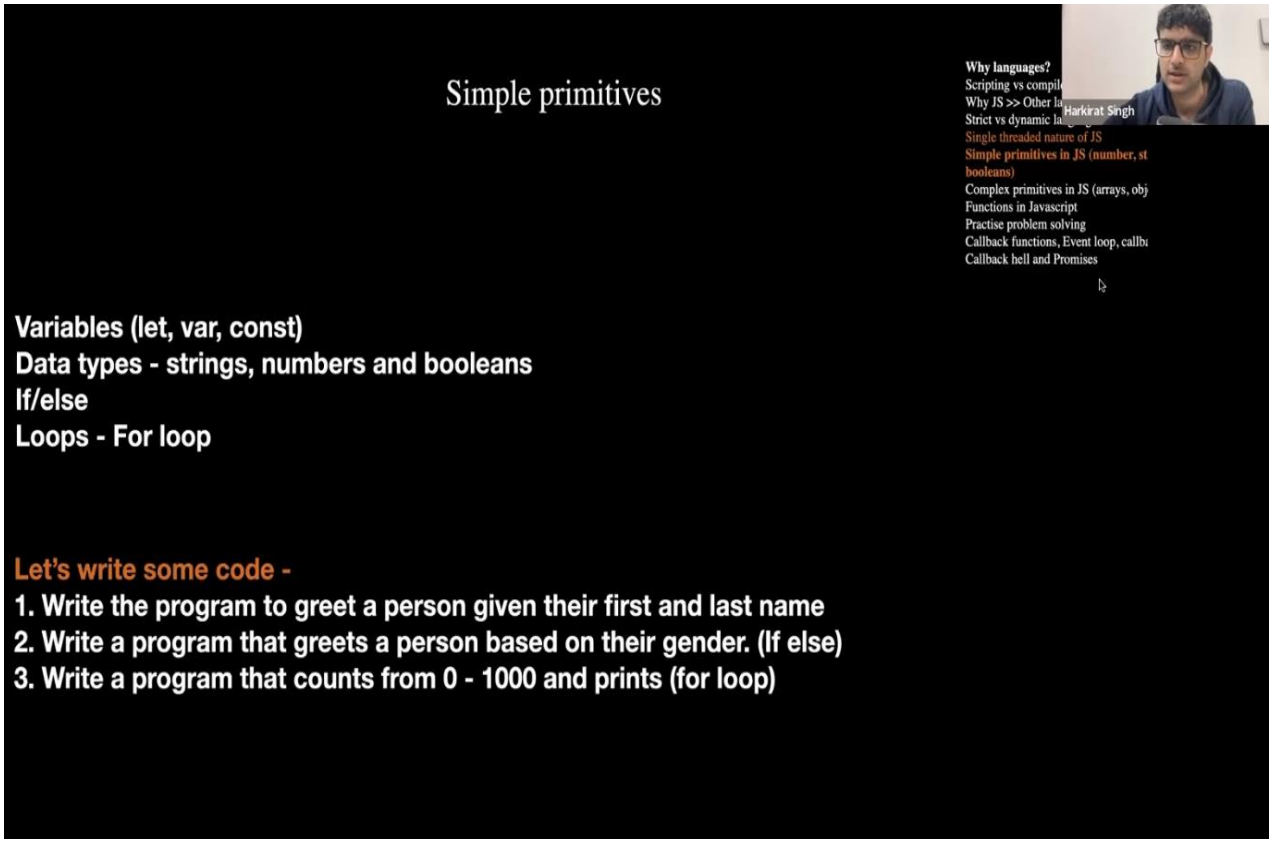
In web browsers, JavaScript is primarily used for client-side scripting, where it interacts with the Document Object Model (DOM) to manipulate web page content, respond to user actions, and perform other tasks. The JavaScript engine, which executes JavaScript code, operates within a single thread known as the main thread.

> The JavaScript engine processes tasks in a single-threaded manner. It picks up tasks from the event queue one by one and executes them sequentially on the main thread. This means that only one piece of JavaScript code is executed at a time, and other tasks must wait in the event queue until the current task is completed.

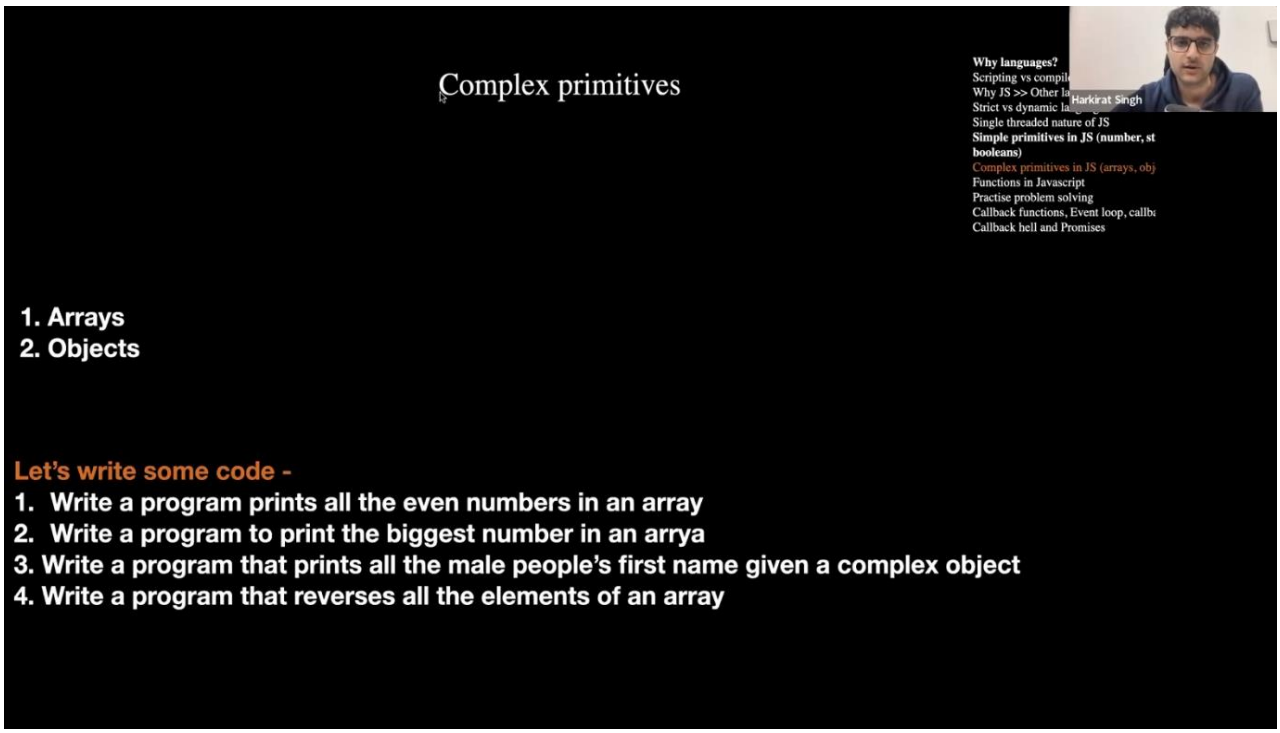




Simple Primitives



Complex Primitives



Functions

Functions

Functions let you

1. Abstract out logic in your program
2. Take arguments as an input
3. Return a value as an output
4. You can think of them as an independent program that is supposed to do something given an input
5. Functions CAN take other functions as input - this will confuse you (callbacks)

Let's write some code -

1. Write a function that finds the sum of two numbers
2. Write another function that displays this result in a pretty format
3. Write another function that takes this sum and prints it in passive tense

Functions

Functions let you

1. Abstract out logic in your program
2. Take arguments as an input
3. Return a value as an output
4. You can think of them as an independent program that is supposed to do something given an input
5. Functions CAN take other functions as input - this will confuse you (callbacks)

https://gist.github.com/hkirat/898ac1da32b6b347a8c0c3e73e1c0666

index.js > ...

```javascript
function sum(num1, num2) {
    let result = num1 + num2;
    return result;
}

function displayResult(data) {
    console.log("Result of the sum is : " + data);
}

function displayResultPassive(data) {
    console.log("Sum's result is : " + data);
}

// You are only allowed to call one function after this
// How will you displayResult of a sum
```

Synchronous and Asynchronous functions

Callback functions, event loops, callback queue

Synchronous vs Asynchronous functions

Synchronous

All the code we've written until now
All code running line by line (hence sync)

Asynchronous

Asynchronous functions in programming are those that allow a program to start a potentially long-running operation and continue executing other tasks without waiting for that operation to complete. This is particularly important in environments like web browsers or Node.js, where waiting for an operation to finish (like fetching data from a server or reading a large file) could make the application unresponsive.

Callback functions, event loops, callback queue

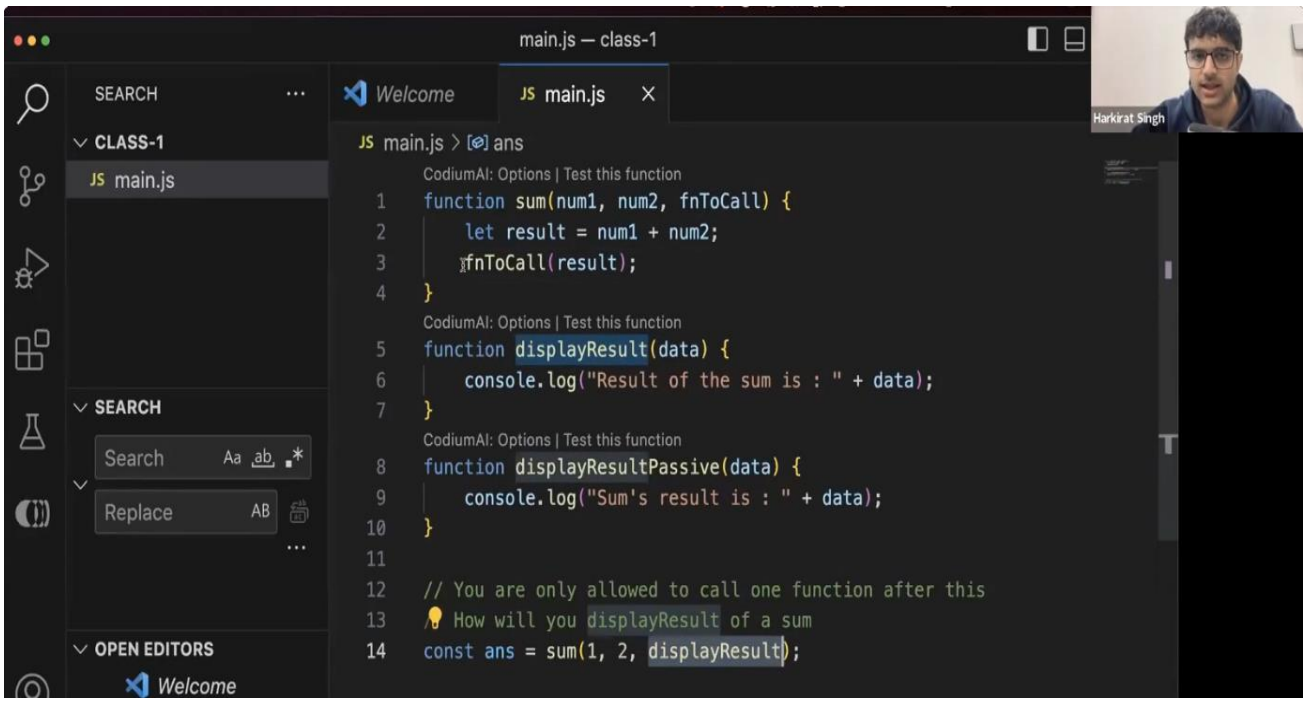
Synchronous vs Asynchronous functions

Why languages?
Scripting vs compiled languages
Why JS >> Other languages in some
Strict vs dynamic languages
Single threaded nature of JS
Simple primitives in JS (number, st
booleans)
Complex primitives in JS (arrays, obj
Functions in Javascript
Practise problem solving
Callback functions, Event loop, callb
Callback hell and Promises

Synchronous

```
function sum() {
  let ans = 0;
  for (let i = 0; i<1000; i++) {
    ans = ans + i;
  }
  return ans;
}
```

Callback Functions



Ek functions argument ki tarah pass kia h bus , aur usko use krlia h , bus itna hi h callback function.