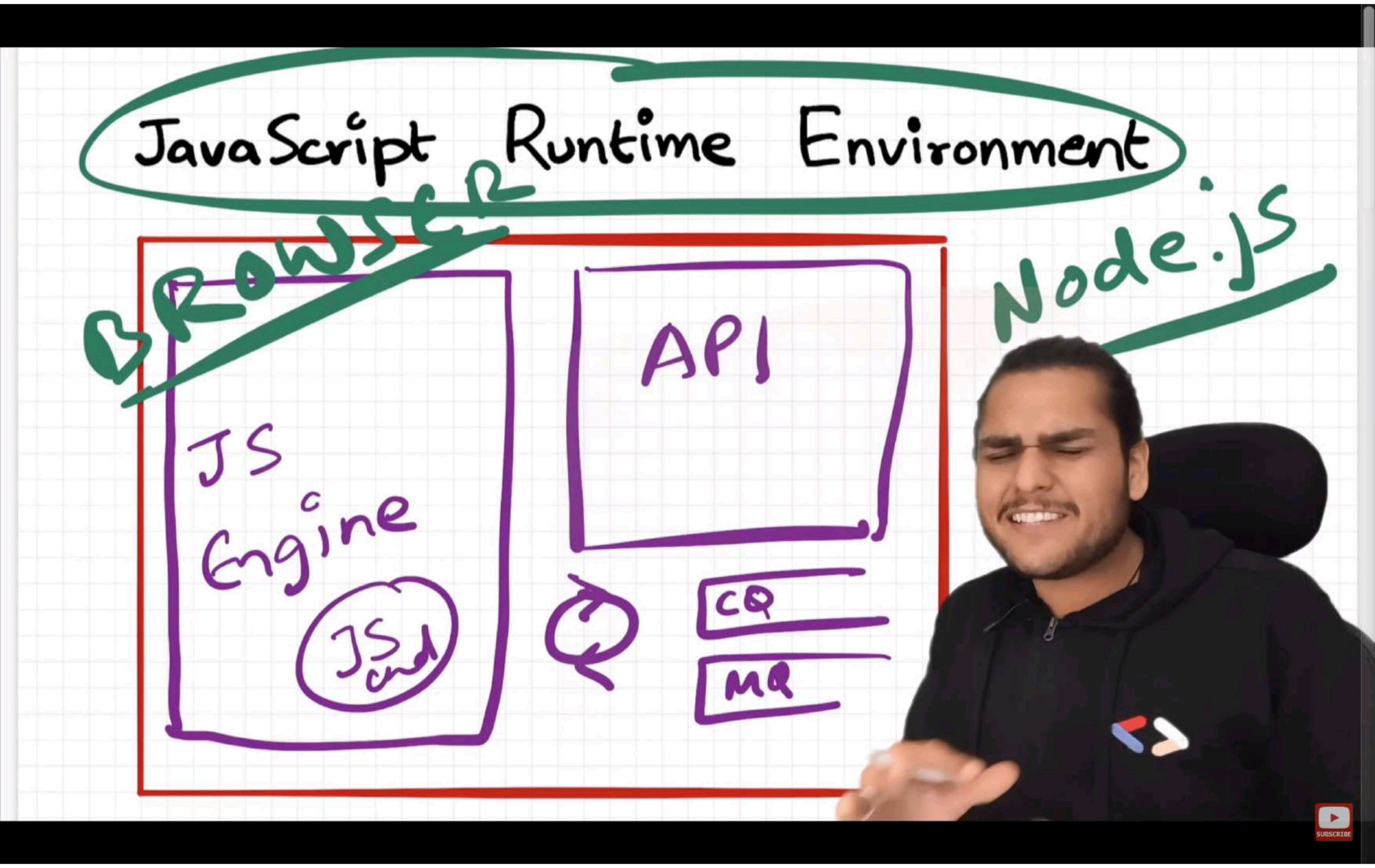
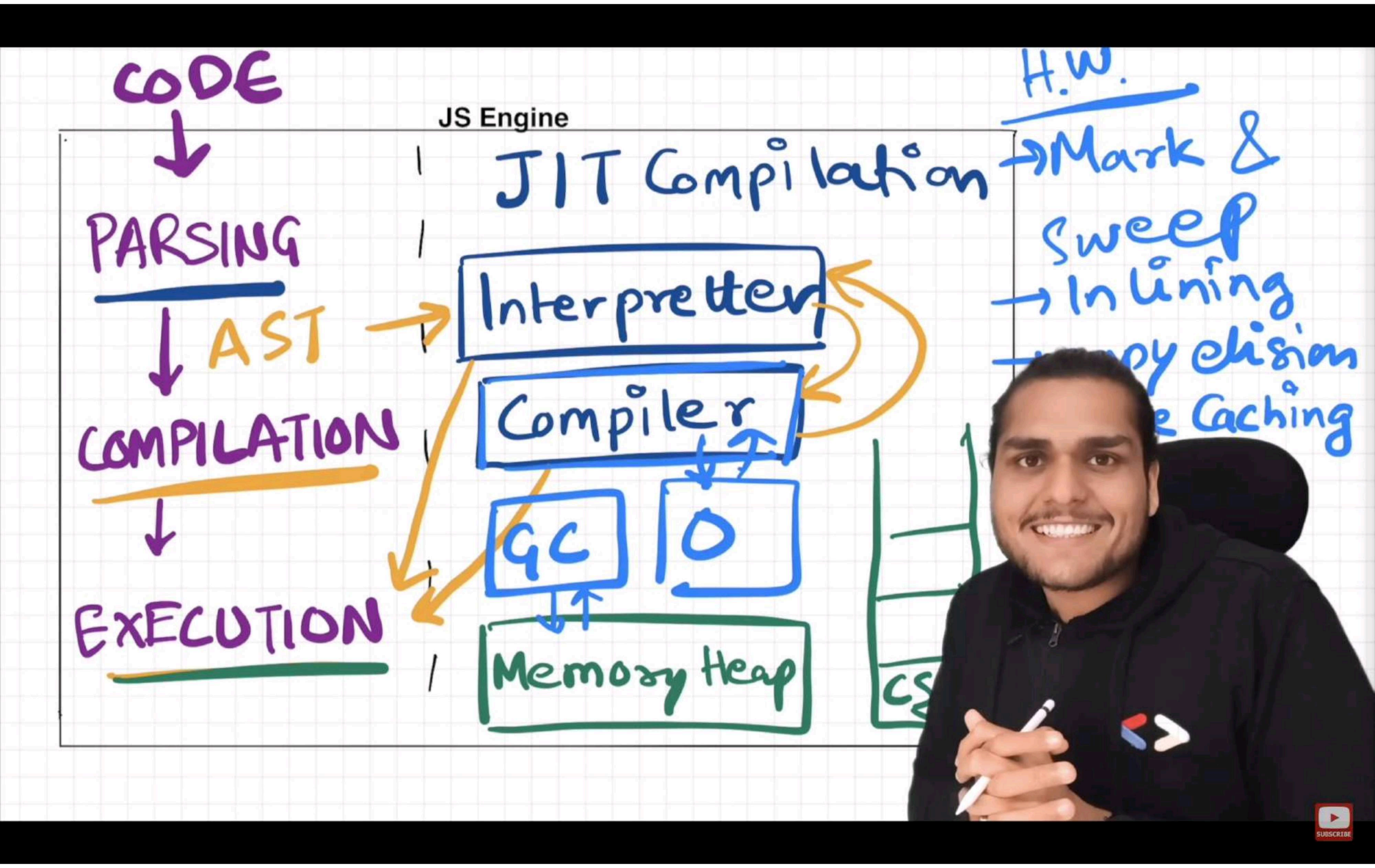
JS Engine EXPOSED 🔥 Google's V8 Architecture 🚀 | Namaste JavaScript Ep. 16

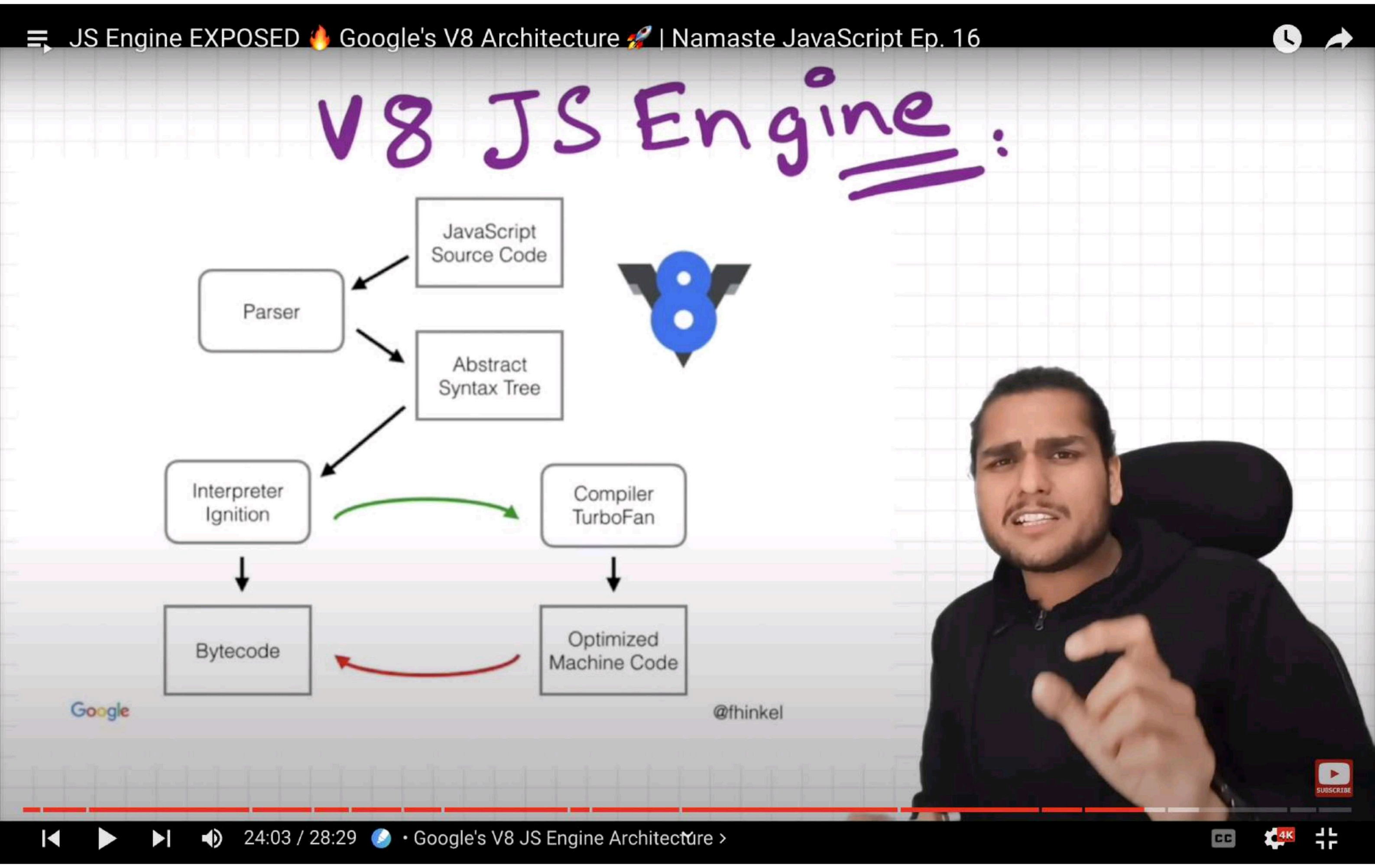
JS

is

EVERYWHERE


JavaScript Runtime Environment
BROWSER
Node.js
API
JS Engine
JS cod
CQ
MQ


CODE
PARSING
AST
COMPILATION
EXECUTION
JS Engine
JIT Compilation
Interpretter
Compiler
GC
O
Memory Heap
CS
H.W.
→ Mark & Sweep
→ Inlining
→ copy elision
e Caching



JS Engine EXPOSED 🔥 Google's V8 Architecture 🚀 | Namaste JavaScript Ep. 16

V8 JS Engine :

JavaScript Source Code

Parser

Abstract Syntax Tree

Interpreter Ignition

Compiler TurboFan

Bytecode

Optimized Machine Code

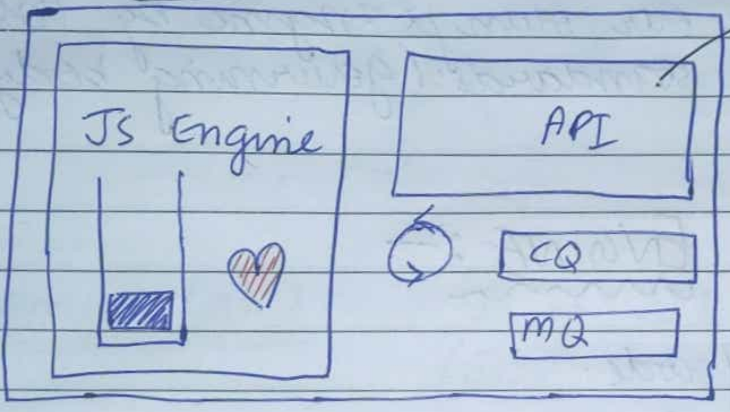
Google

@fhinkel

JS Engine & JRE

Js can run on smart watch, browser, server, light bulb, robot due to JRE
↓
Big container containg all things to run a js code.

JRE



to connect it to the outer environment.

→ JRE is not possible without JS Engine.
→ Browser can execute js code because it has JRE
→ Every browser has its own JRE.
→ Node js itself is an open JRE.

example: for running js in water coller, we need JRE in it. It can have a js engine & API such as getWaterLevel(), we can use in JS engine through a global object.

→ API can be different or similar in different JRE such as browsers JRE & node
↓ API ↓ localStorage
↓ different for it.

-> Also setTimeout & console are present in both, but they are implemented differently

lists of JS engine :

Microsoft edge - chakra

Mozilla firefox - Spider monkey -> first js engine in netscape nav.

google chrome, node js, deno - V8

☆ We can create our own js engine by following ECMA script standards. (governing body.)

JAVASCRIPT ENGINE :-

Just a piece of code.
V8 is written in C++.

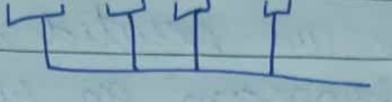
I/P (high level code) ⟶ O/P (binary machine level code)

JS Engine architecture:

code
↓
(I) PARSING (tokenization, Syntax tree form)
↓
(II) compilation (interpreter, compiler) ⇄ optimize
(both goes hand in hand)
↓
(III) Execution
(uses memory heap & call stack.)

(I) <u>Parsing</u> : 1) Broken down into tokens.

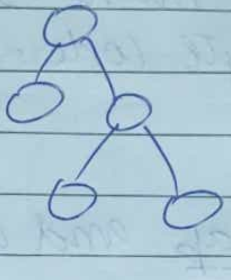
example: let a = 7



different tokens.

2) <u>Syntax parser</u> - Convert code to AST (Abstract Syntax tree)

<u>AST</u> →



<u>astexplorer.net</u>

(II) <u>Compilation</u> : JS is JIT Compilation (Just in time)

Interpreter : compile program line by line

<u>Compiler</u> : Optimize the code, regenerate, make executable & then execute. (performance improvement)

↓

<u>Efficient</u>

> <u>Code is fast</u>

(JS can behave like both, depending on JRE.)

→ Firstly JS run on interpreter, but now modern browsers & JS engines uses a compiler & an interpreter both. (JIT)

* Best of both world's.

→ for faster execution of programs in browser, interpreter are used to so execute program fast & also compiler are executed simeltaneously to optimize our program on runtime / JIT / ~~or~~ AOT (Ahead of time)

↓

Compiler take a code before execution & optimize it as much as it can & provide byte code.

(III.) Contain memory heap and call stack

└ { place where all the memory is present.

Garbage collector : tries to free memory, whenever possible.

whenever a function is not being used, or we clear the timeout.

uses Mark and Sweep algorithm* (popular)

Other optimization: (compiler does for code)

i) Inlining

ii) Copy elision

iii) Inline caching

→ Each company want to make their js engine as fast as possible.

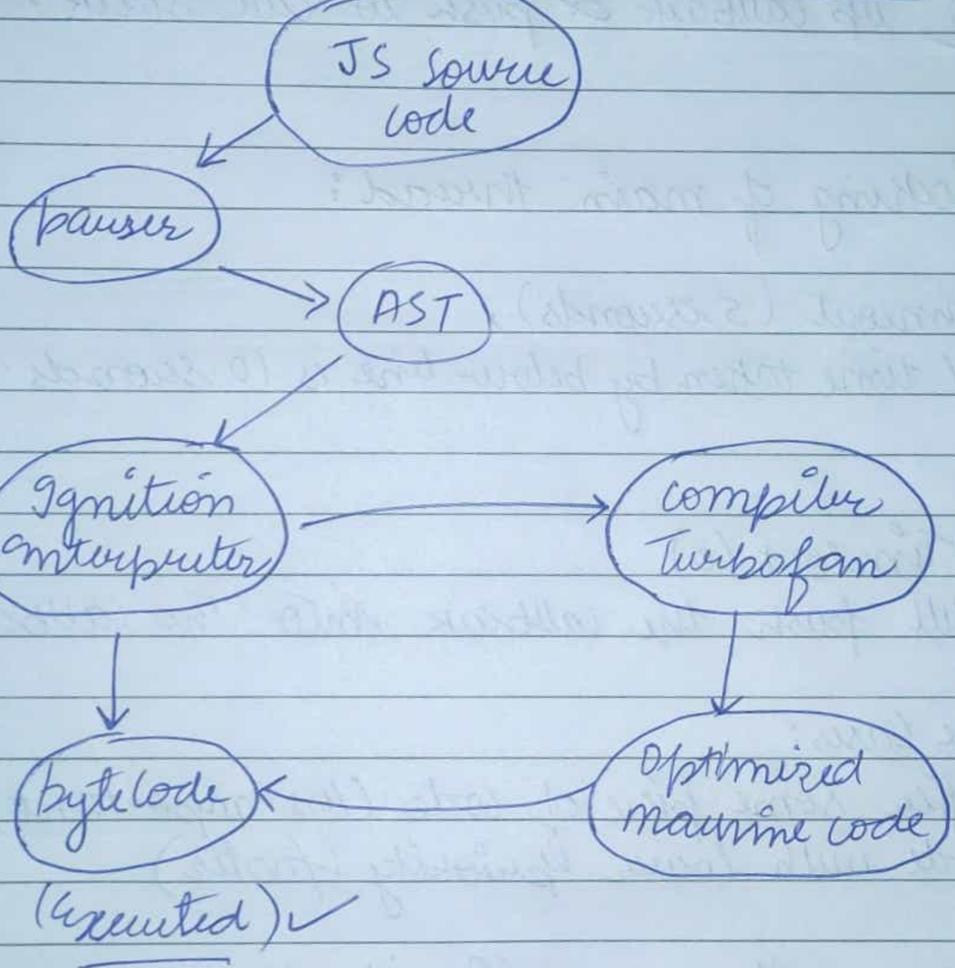
Fastest ever → V8

About V8:

Interpreter : Ignition

Optimizing Compiler : Turbofan

V8 ~~architetture~~ ~~architecutl~~ architecture :



Read from official V8 website

Orinoco : Garbage collector of V8 (uses mark & sweep algo.)

Oilpan : Garbage collector (for another purpose)