Node.js

Article Talk

From Wikipedia, the free encyclopedia

Node.js is a cross-platform, open-source JavaScript runtime environment that can run on Windows, Linux, Unix, macOS, and more. Node.js runs on the V8 JavaScript engine, and executes JavaScript code outside a web browser.

Node.js lets developers use JavaScript to write command line tools and for server-side scripting. The ability to run JavaScript code on the server is often used to generate dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm,[6] unifying web-application development around a single programming language, as opposed to using different languages for the server- versus client-side programming.

Node.js has an event-driven architecture capable of asynchronous I/O. These design choices aim to optimize throughput and scalability in web applications with many input/output operations, as well as for real-time Web applications (e.g., real-time communication programs and browser games).[7]

The Node.js distributed development project was previously governed by the Node.js Foundation,[8] and has now merged with the JS Foundation to form the OpenJS Foundation. OpenJS Foundation is facilitated by the Linux Foundation's Collaborative Projects program.[9]

Node.js

Article Talk

From Wikipedia, the free encyclopedia

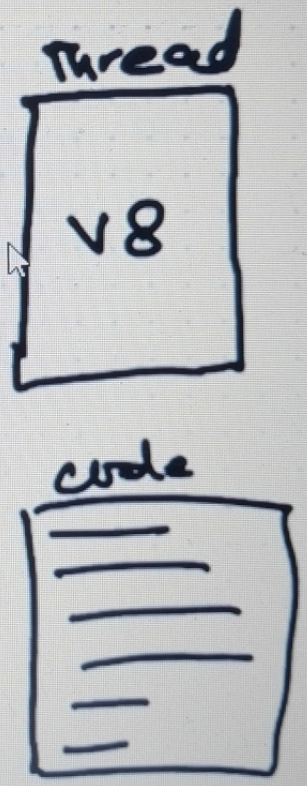
Node.js is a cross-platform, open-source JavaScript runtime environment that can run on Windows, Linux, Unix, macOS, and more. Node.js runs on the V8 JavaScript engine, and executes JavaScript code outside a web browser.

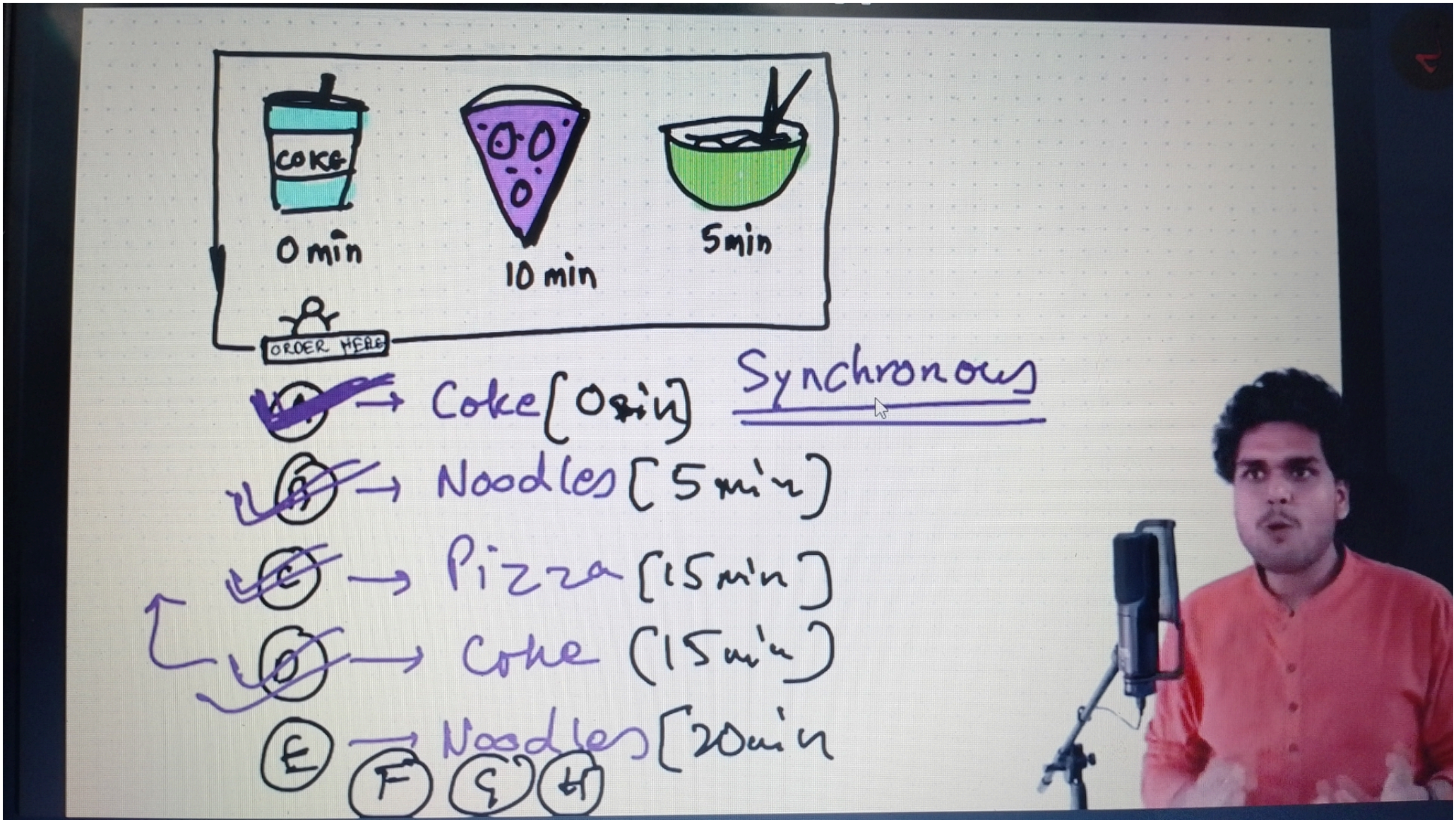
Node.js lets developers use JavaScript to write command line tools and for server-side scripting. The ability to run JavaScript code on the server is often used to generate dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm,[6] unifying web-application development around a single programming language, as opposed to using different languages for the server- versus client-side programming.

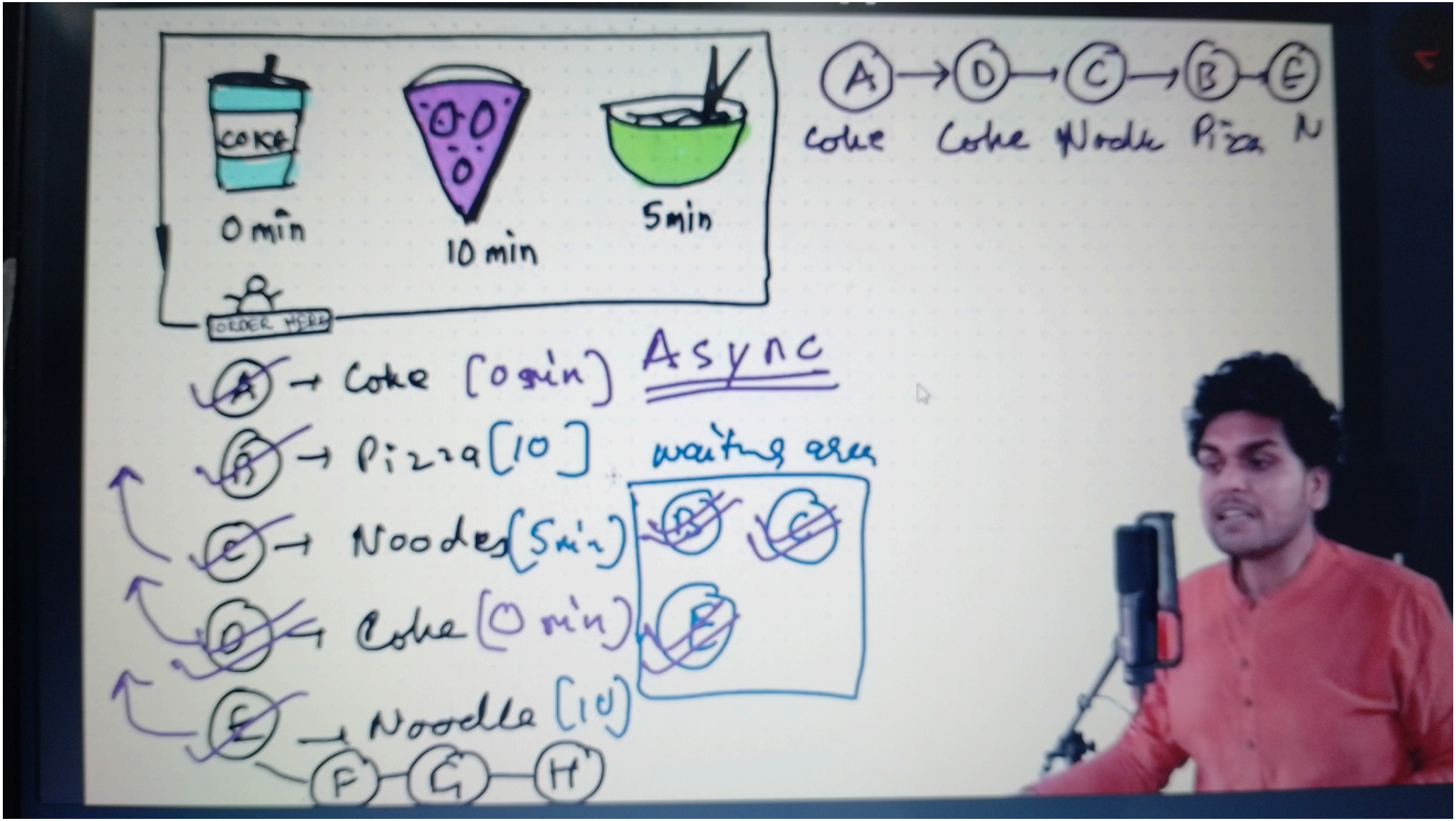
Node.js has an event-driven architecture capable of asynchronous I/O. These design choices aim to optimize throughput and scalability in web applications with many input/output operations, as well as for real-time Web applications (e.g., real-time communication programs and browser games).[7]

The Node.js distributed development project was previously governed by the Node.js Foundation,[8] and has now merged with the JS Foundation to form the OpenJS Foundation. OpenJS Foundation is facilitated by the Linux Foundation's Collaborative Projects program.[9]

JS → synchronous
single threaded




COKE
0 min
10 min
5min
ORDER HERE
Coke [0min]
Noodles [5min]
Pizza [15min]
Coke [15min]
Noodles [20min
Synchronous


COKE
0 min
10 min
5min
ORDER HERE
A → D → C → B → E
Coke Coke Noodle Pizza N
A → Coke [0 min]
Async
B → Pizza [10]
waiting area
C → Noodles [5 min]
D → Coke [0 min]
E → Noodle [10]
F → G → H

Synchronous

```
var a = 1078698;
var b = 20986;

function multiplyFn(x, y) {
  const result = a * b;
  return result;
}

var c = multiplyFn(a, b);
```

Asynchronous

```
https.get("https://api.fbi.com", (res) => {
  console.log("secret data:" + res.secret);
});

fs.readFile("./gossip.txt", "utf8", (data) => {
  console.log("File Data", data);
});

setTimeout(() => {
  console.log("Wait here for 5 seconds");
}, 5000);
```

Synchronous

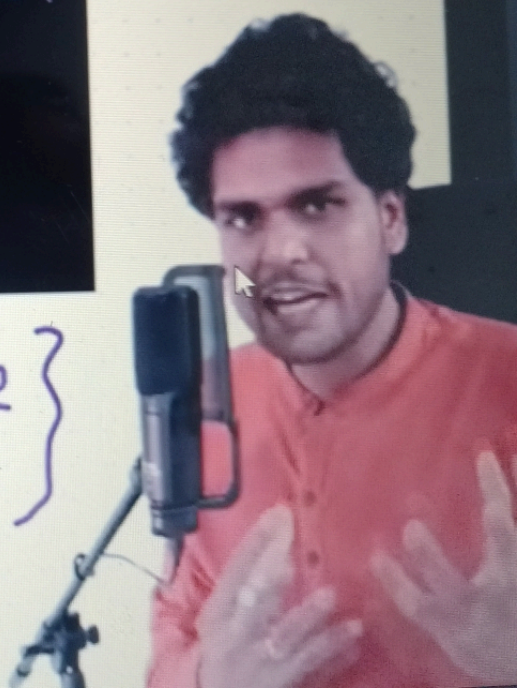
```
var a = 1078698;
var b = 20986;

function multiplyFn(x, y) {
  const result = a * b;
  return result;
}

var c = multiplyFn(a, b);
```

These tasks can execute immediately

Asynchronous

```
https.get("https://api.fbi.com", (res) => {
  console.log("secret data:" + res.secret);
});

fs.readFile("./gossip.txt", "utf8", (data) => {
  console.log("File Data", data);
});

setTimeout(() => {
  console.log("Wait here for 5 seconds");
}, 5000);
```

These task take time to execute



Synchronous

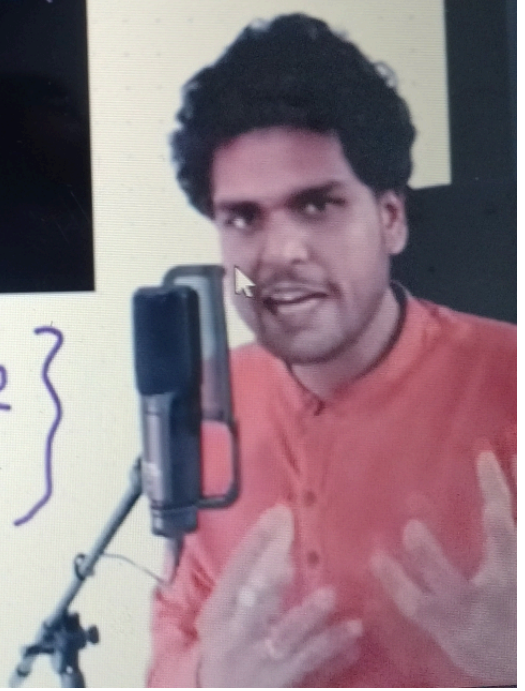
```
var a = 1078698;
var b = 20986;

function multiplyFn(x, y) {
  const result = a * b;
  return result;
}

var c = multiplyFn(a, b);
```

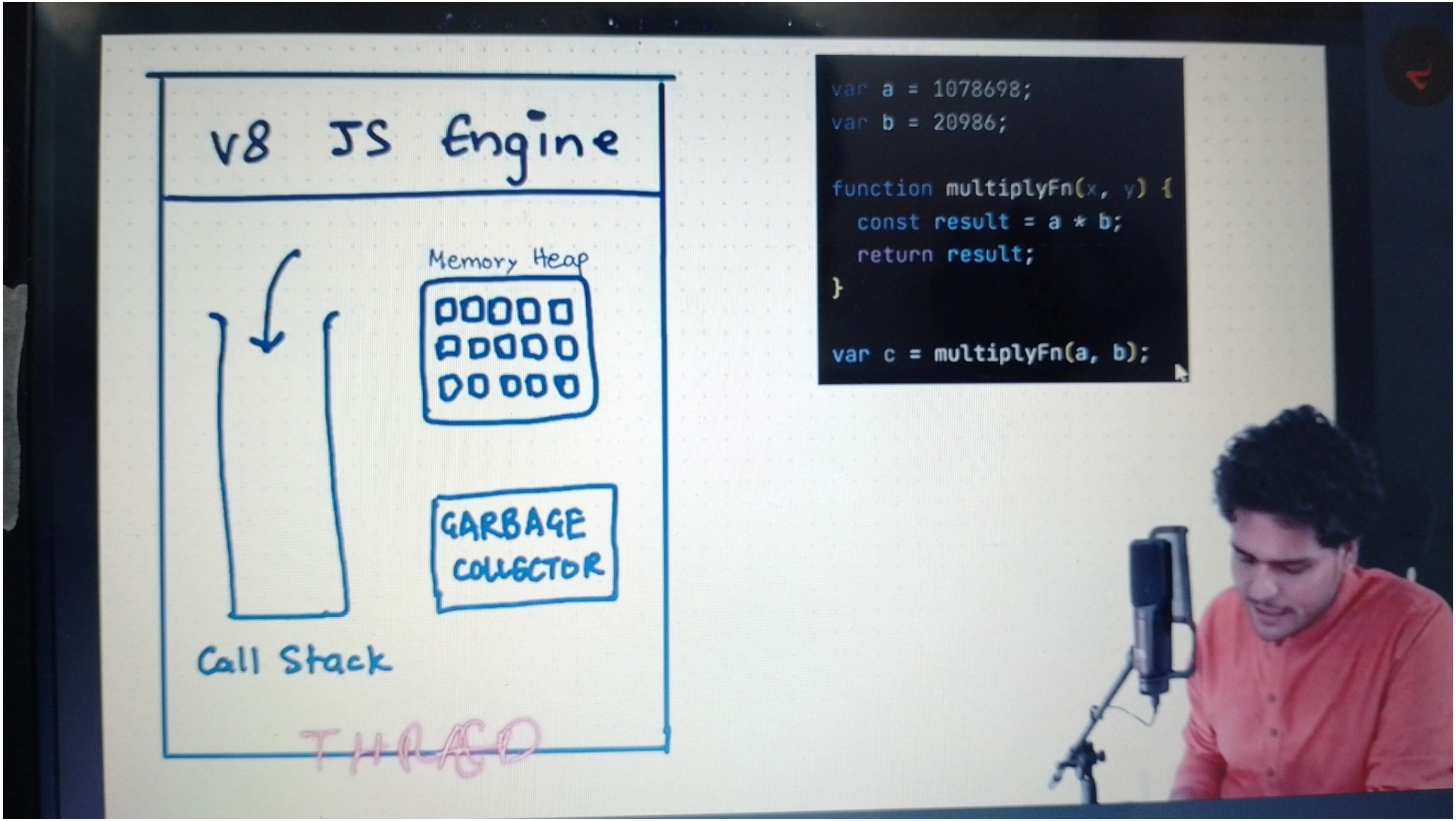
These tasks can execute immediately

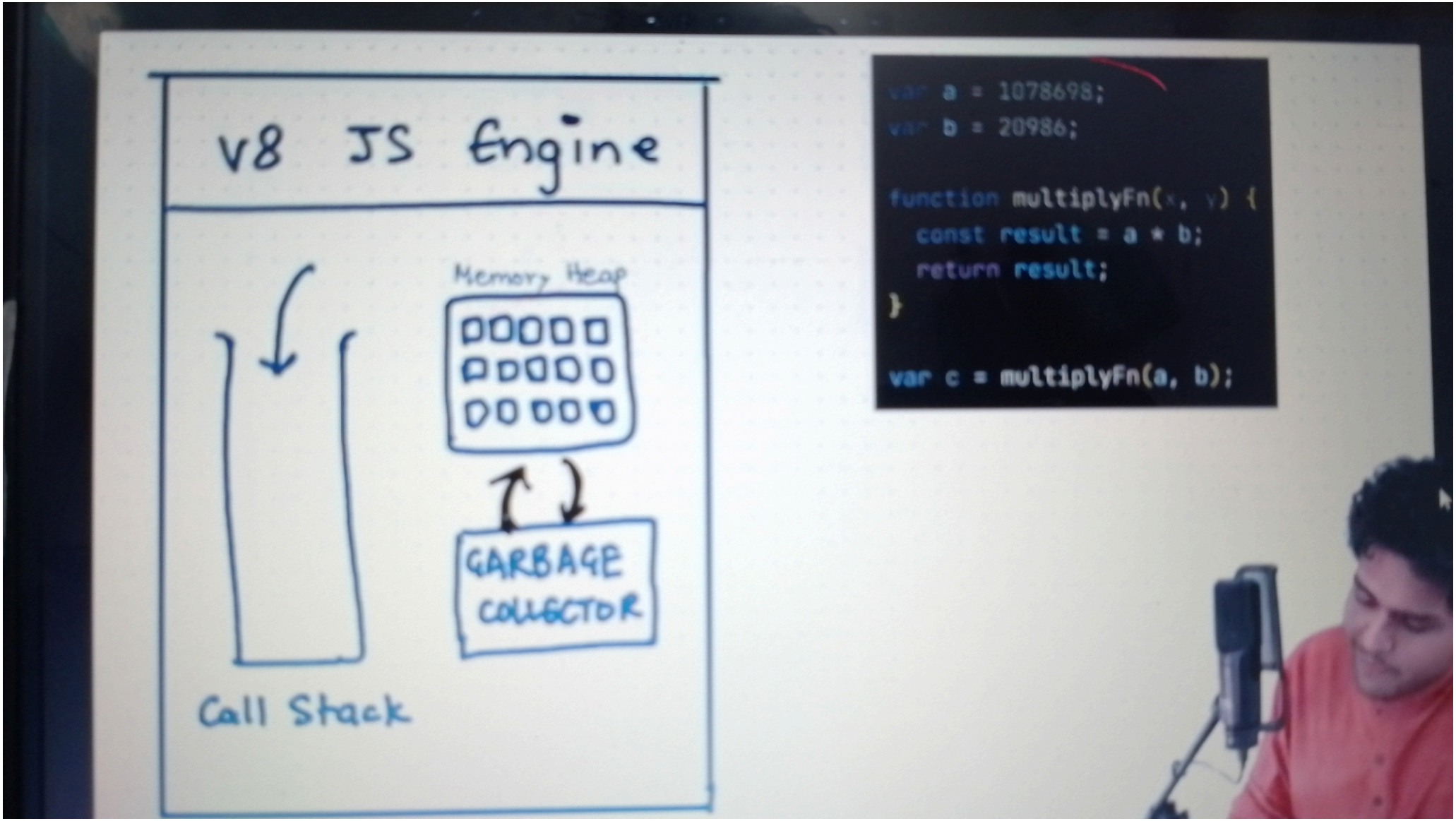
Asynchronous

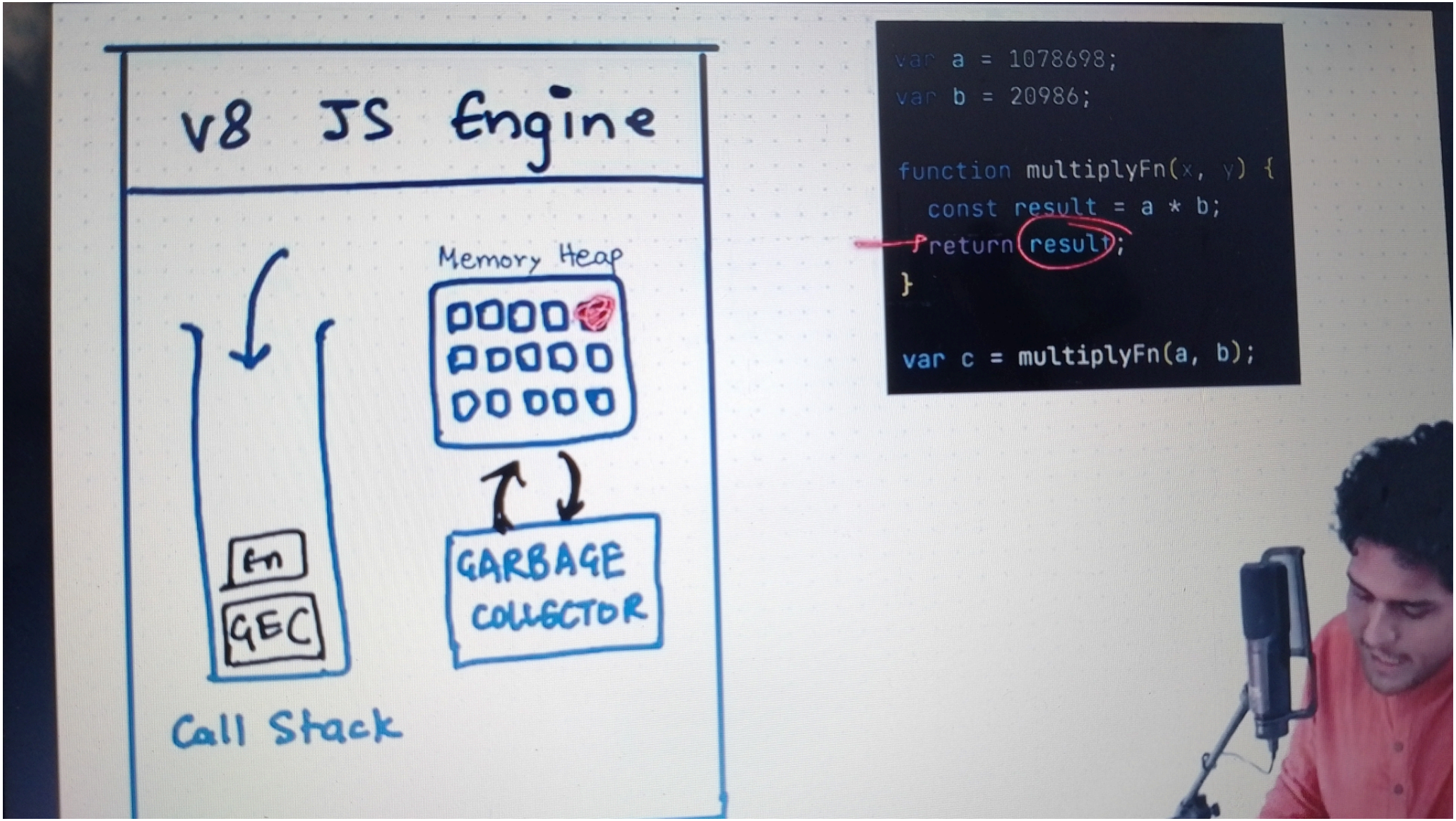
```
https.get("https://api.fbi.com", (res) => {
  console.log("secret data:" + res.secret);
});

fs.readFile("./gossip.txt", "utf8", (data) => {
  console.log("File Data", data);
});

setTimeout(() => {
  console.log("Wait here for 5 seconds");
}, 5000);
```

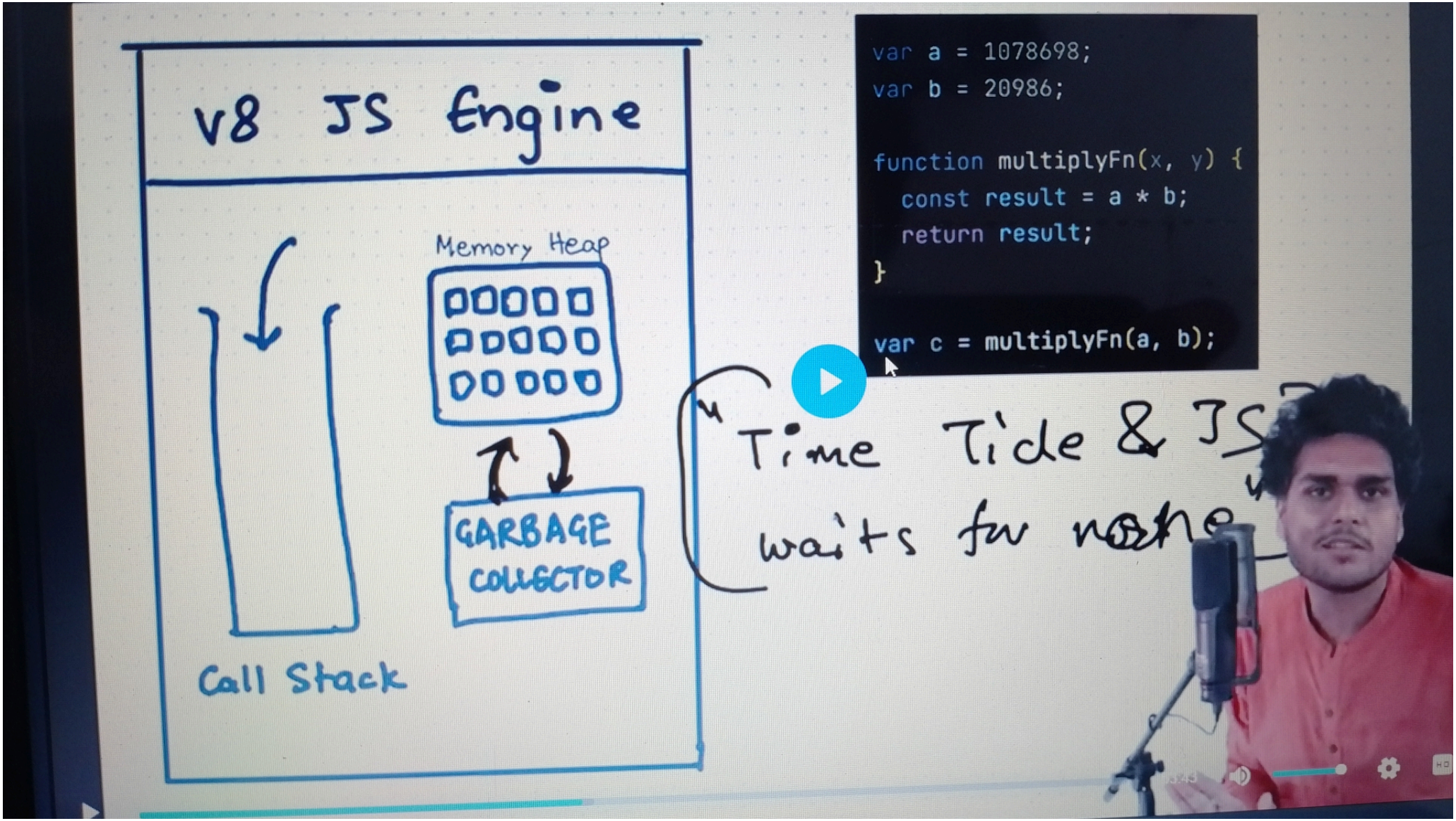
These task take time to execute

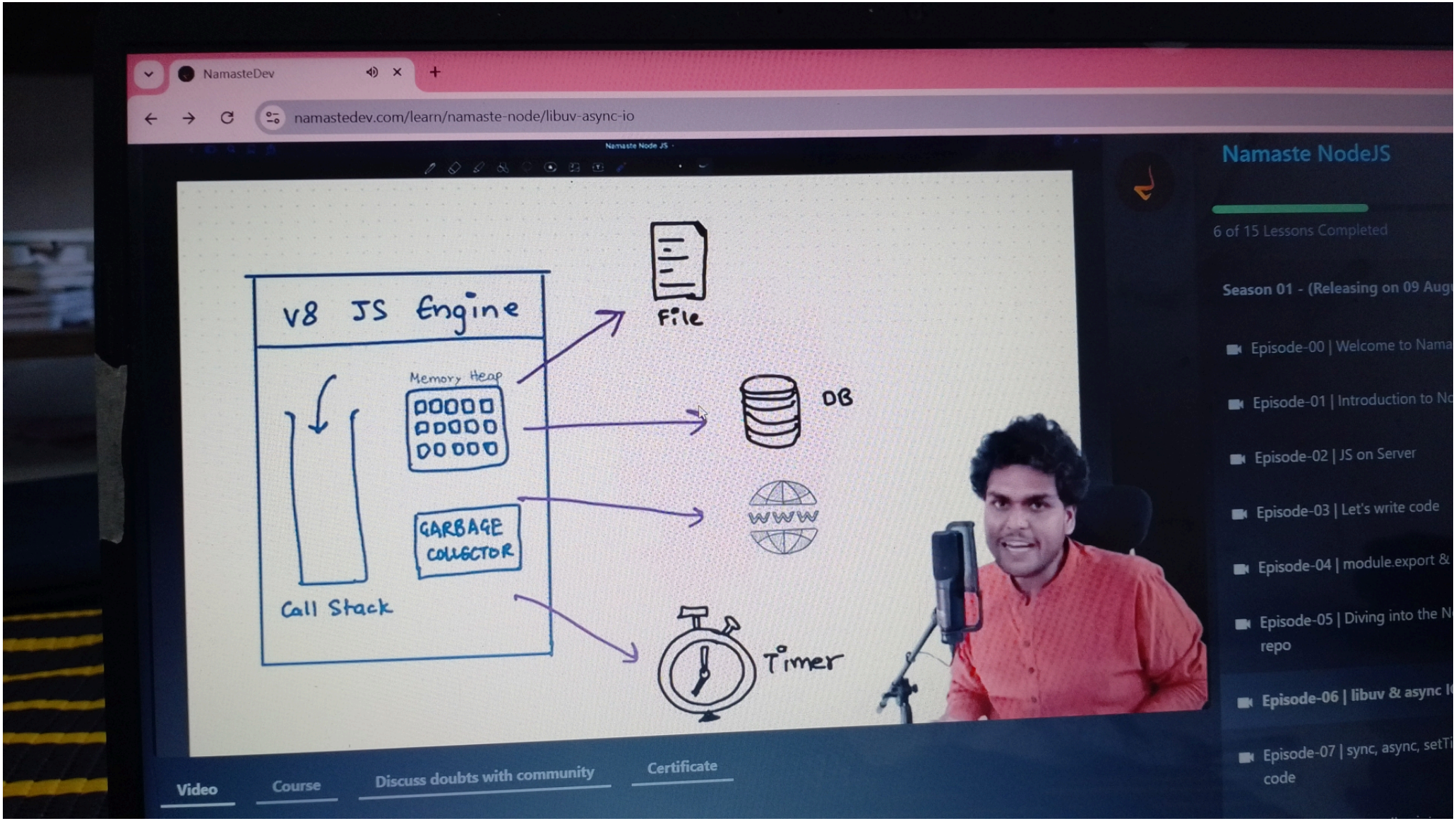


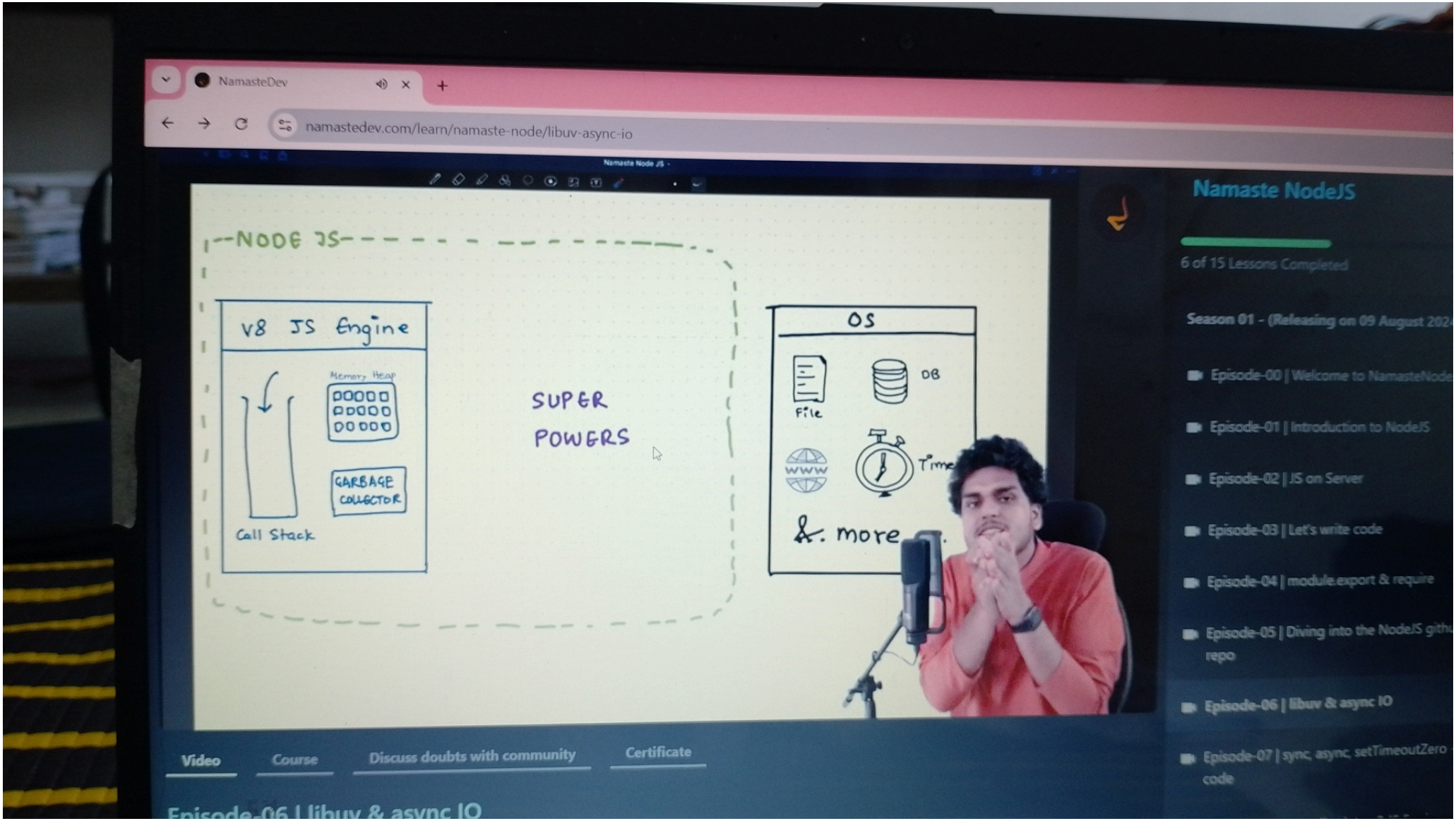

V8 JS Engine
Memory Heap
GARBAGE COLLECTOR
Call Stack
THREAD
var a = 1078698;
var b = 20986;
function multiplyFn(x, y) {
const result = a * b;
return result;
}
var c = multiplyFn(a, b);

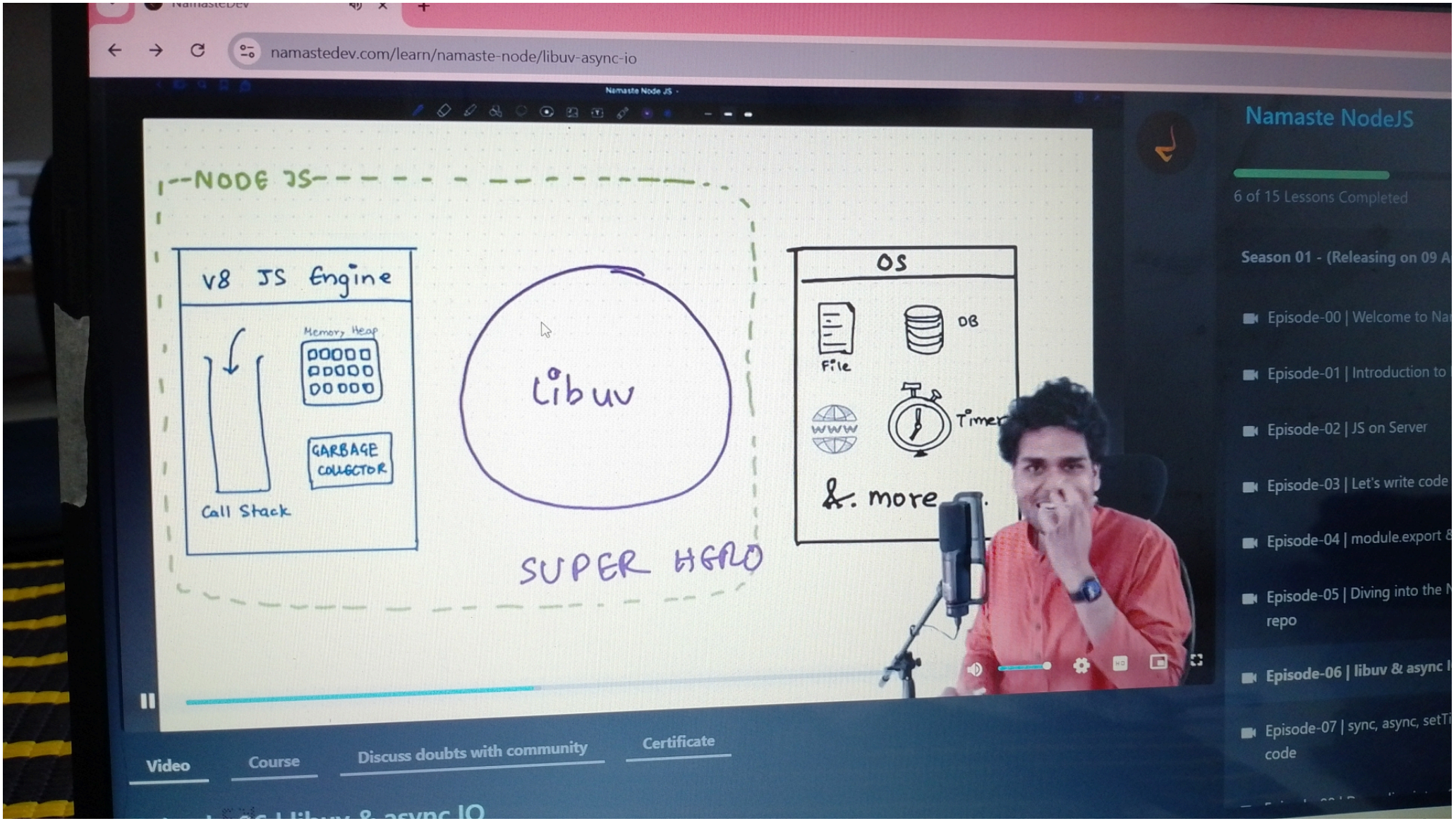

V8 JS Engine
Memory Heap
GARBAGE COLLECTOR
Call Stack
var a = 1078698;
var b = 20986;
function multiplyFn(x, y) {
const result = a * b;
return result;
}
var c = multiplyFn(a, b);

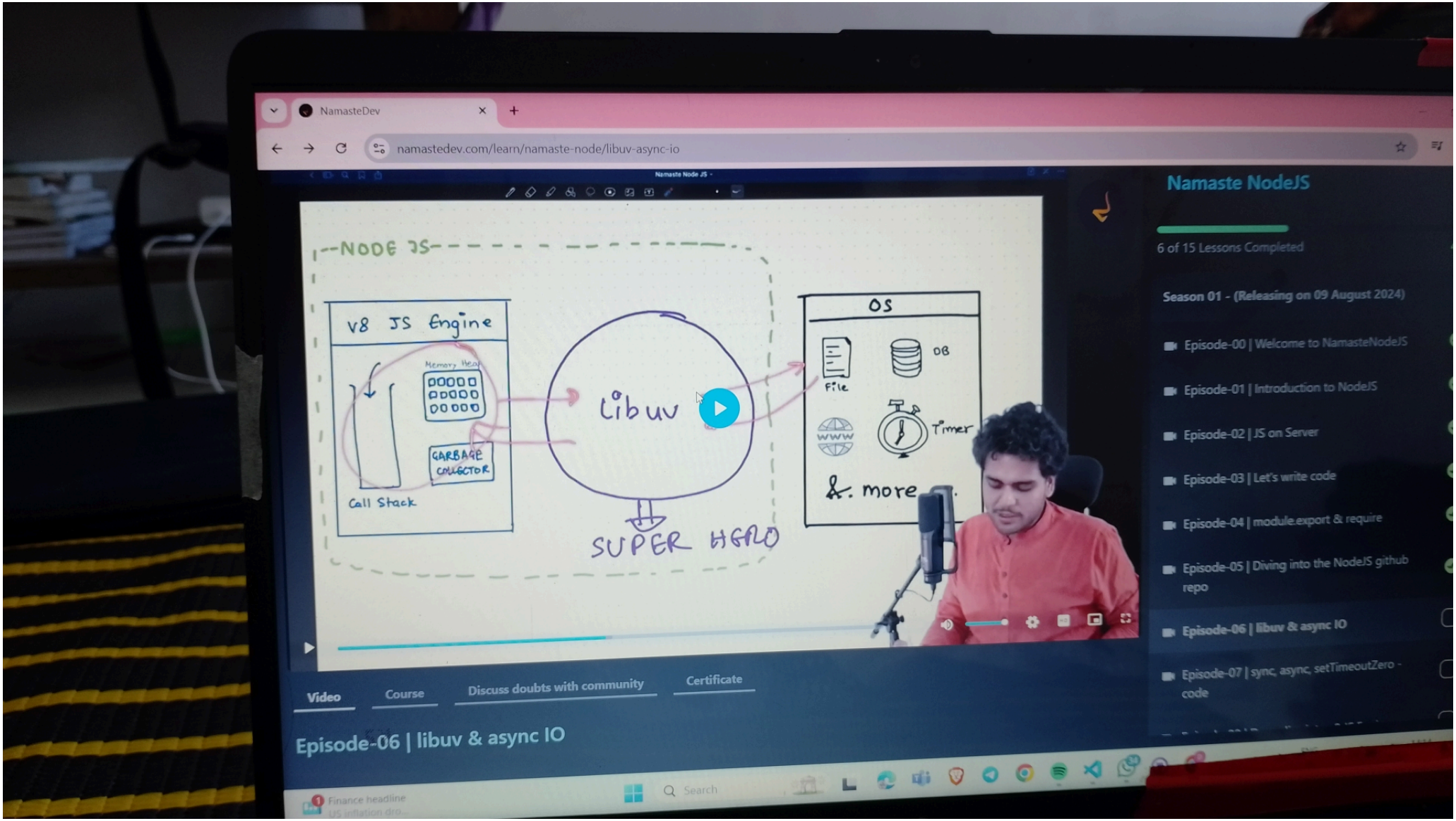

V8 JS Engine
Memory Heap
fn
GEC
Call Stack
GARBAGE COLLECTOR
var a = 1078698;
var b = 20986;
function multiplyFn(x, y) {
const result = a * b;
return result;
}
var c = multiplyFn(a, b);

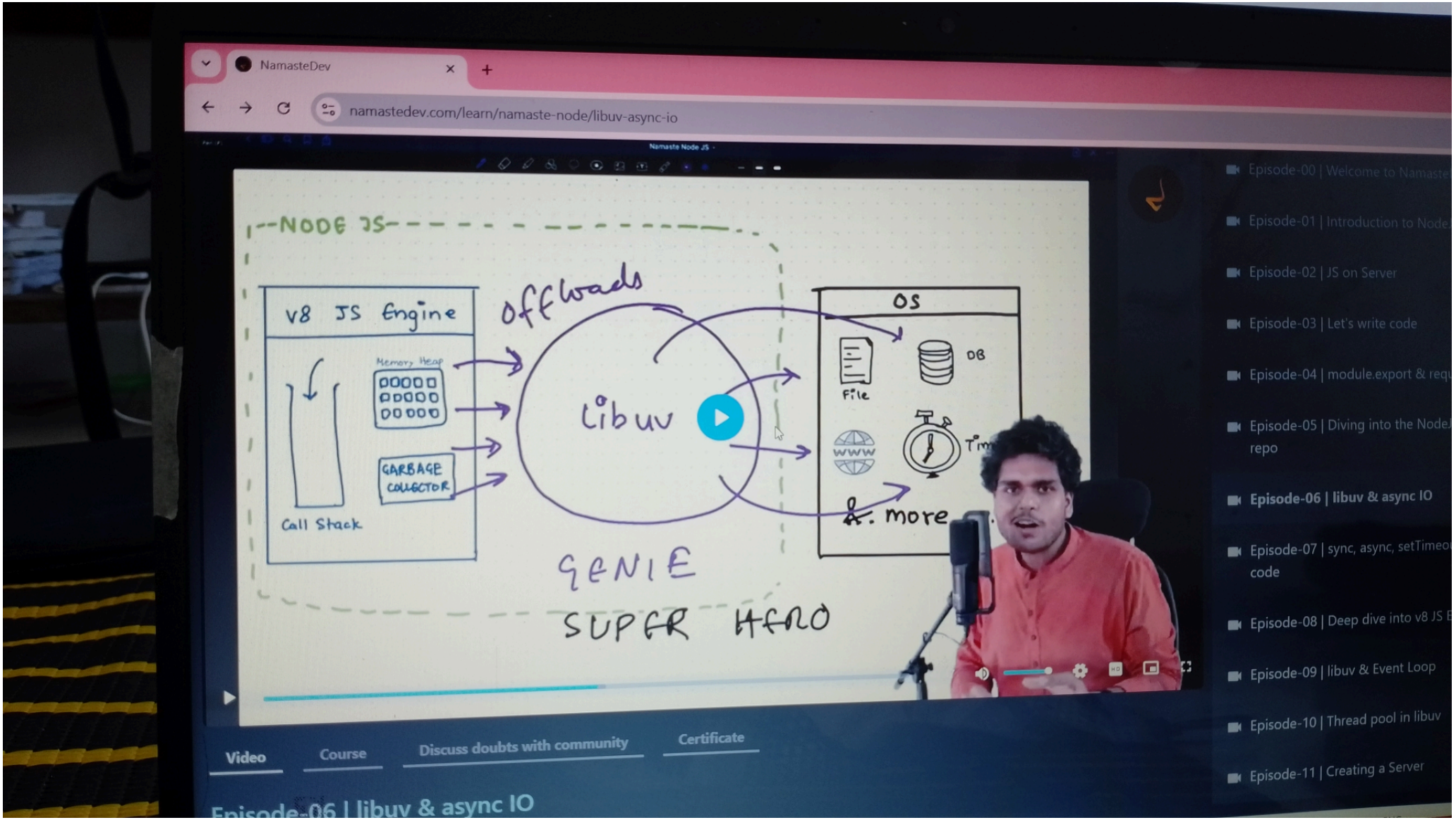

V8 JS Engine
Memory Heap
GARBAGE COLLECTOR
Call Stack
var a = 1078698;
var b = 20986;
function multiplyFn(x, y) {
const result = a * b;
return result;
}
var c = multiplyFn(a, b);
Time Tide & JS
waits for none

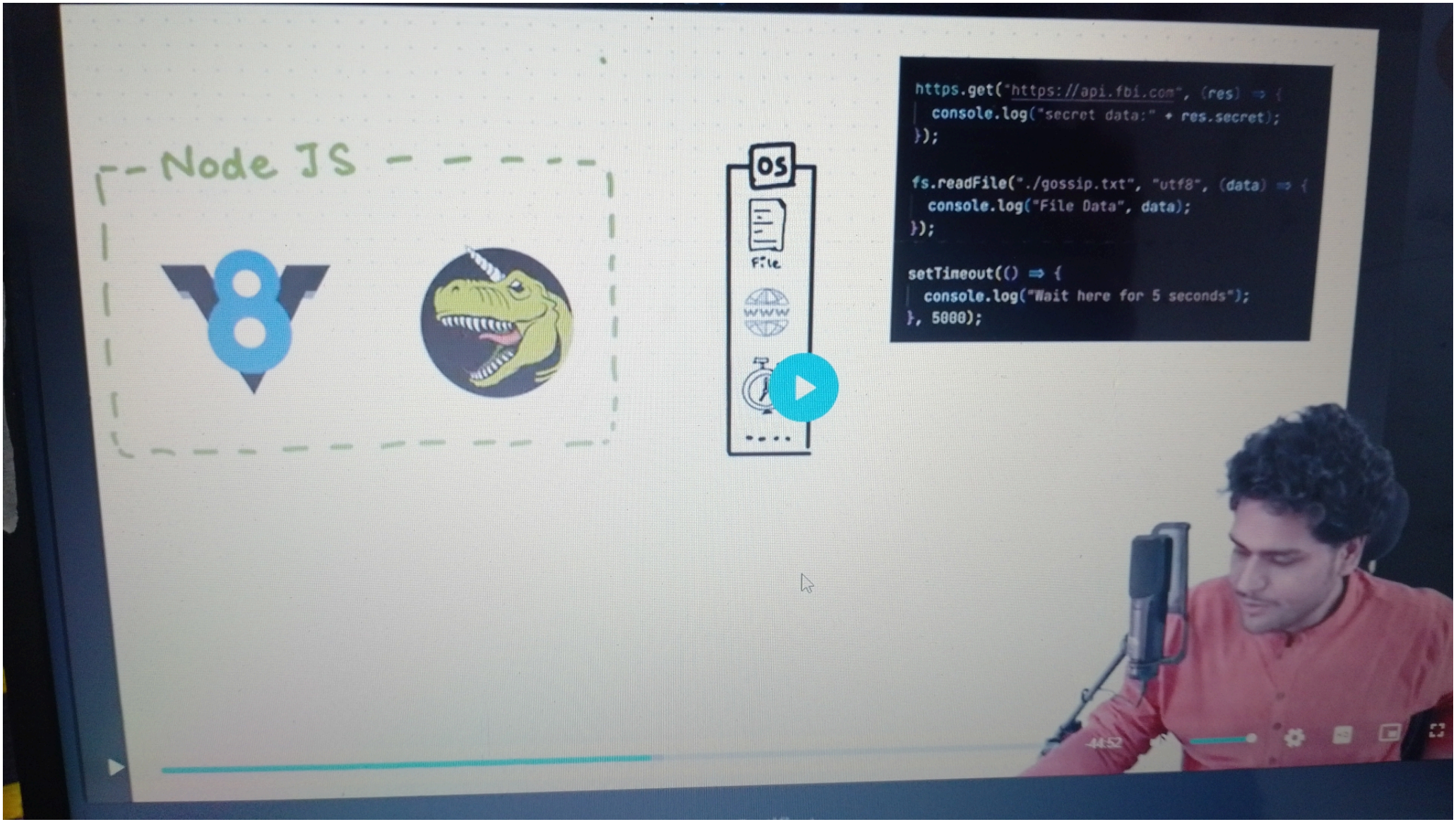

NamasteDev
namastedev.com/learn/namaste-node/libuv-async-io
Namaste Node JS
V8 JS Engine
Memory Heap
Call Stack
GARBAGE COLLECTOR
File
DB
WWW
Timer
Namaste NodeJS
6 of 15 Lessons Completed
Season 01 - (Releasing on 09 Aug
Episode-00 | Welcome to Nama
Episode-01 | Introduction to No
Episode-02 | JS on Server
Episode-03 | Let's write code
Episode-04 | module.export &
Episode-05 | Diving into the N
repo
Episode-06 | libuv & async I
Episode-07 | sync, async, setTi
code
Video
Course
Discuss doubts with community
Certificate

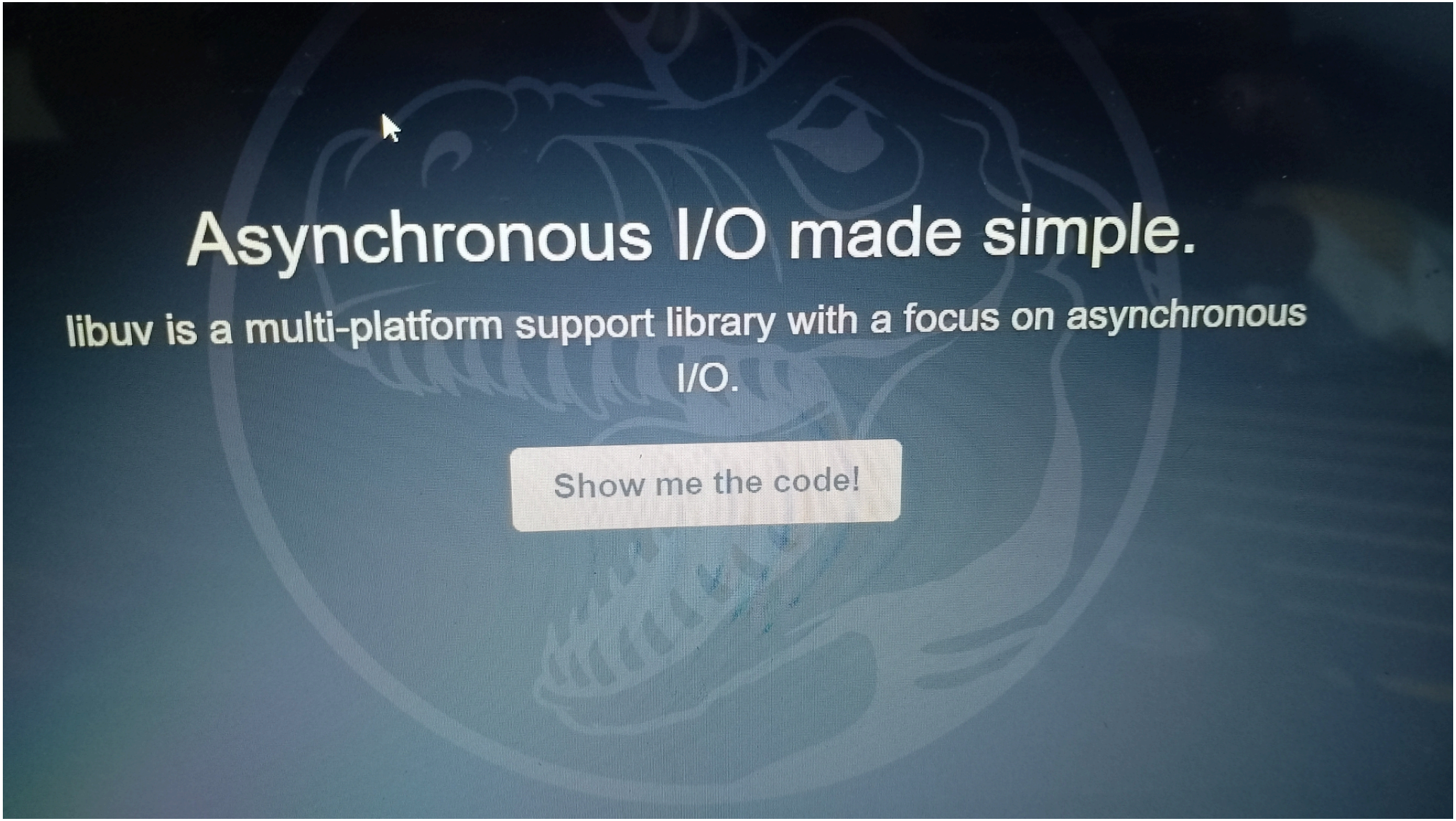

NamasteDev
namastedev.com/learn/namaste-node/libuv-async-io
NODE JS
V8 JS Engine
Memory Heap
GARBAGE COLLECTOR
Call Stack
SUPER POWERS
OS
File
DB
www
& more
Namaste NodeJS
6 of 15 Lessons Completed
Episode-00 | Welcome to NamasteNode
Episode-01 | Introduction to NodeJS
Episode-02 | JS on Server
Episode-03 | Let's write code
Episode-04 | module.export & require
Episode-05 | Diving into the NodeJS github repo
Episode-06 | libuv & async IO
Episode-07 | sync, async, setTimeoutZero code
Video
Course
Discuss doubts with community
Certificate
Episode-06 | libuv & async IO

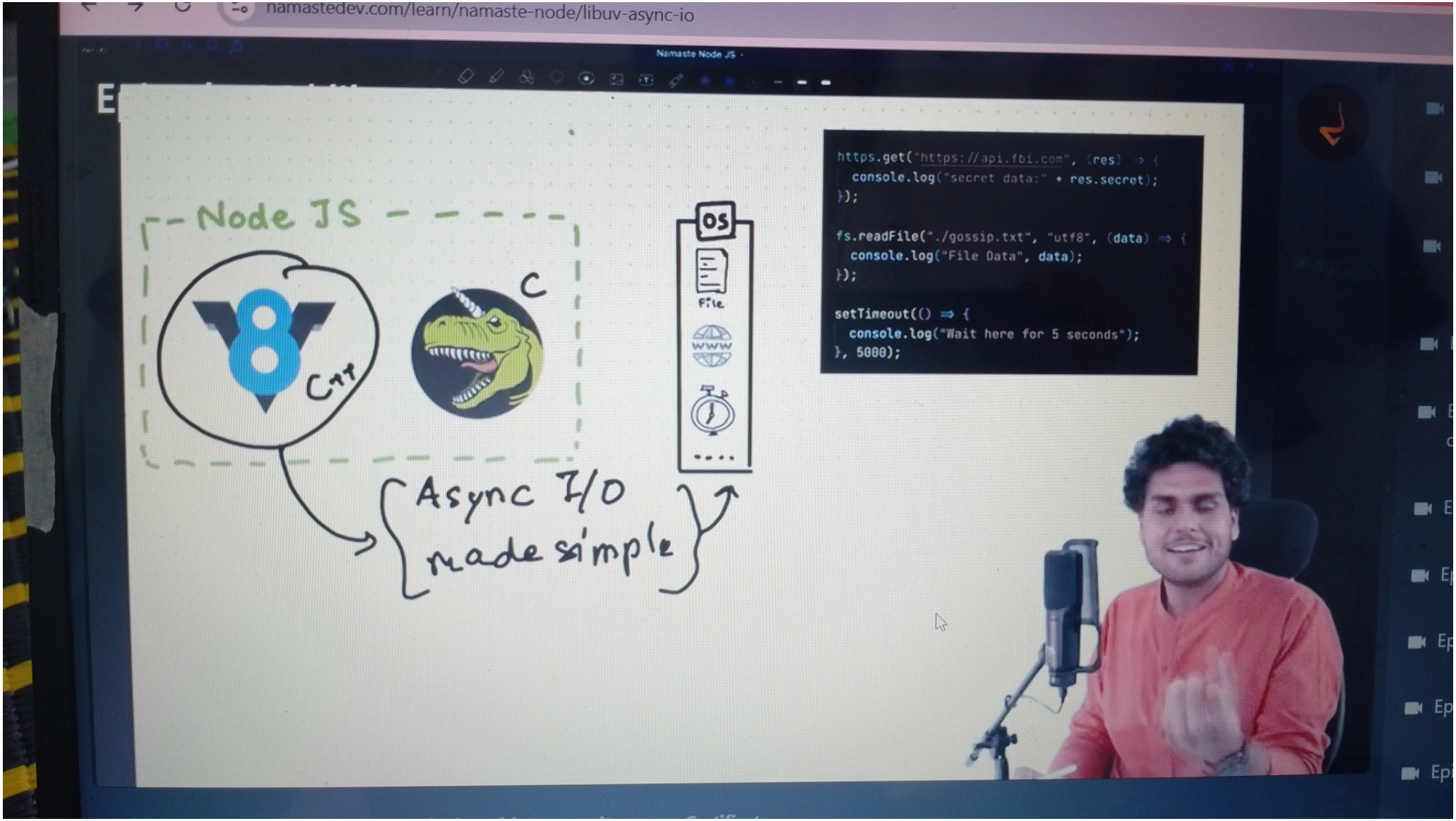

namastedev.com/learn/namaste-node/libuv-async-io
Namaste Node JS
--NODE JS--
V8 JS Engine
Memory Heap
GARBAGE COLLECTOR
Call Stack
Libuv
OS
File
DB
Timer
& more...
SUPER HERO
Namaste NodeJS
6 of 15 Lessons Completed
Season 01 - (Releasing on 09 A
Episode-00 | Welcome to Na
Episode-01 | Introduction to
Episode-02 | JS on Server
Episode-03 | Let's write code
Episode-04 | module.export
Episode-05 | Diving into the N repo
Episode-06 | libuv & async I
Episode-07 | sync, async, setTi code
Video
Course
Discuss doubts with community
Certificate

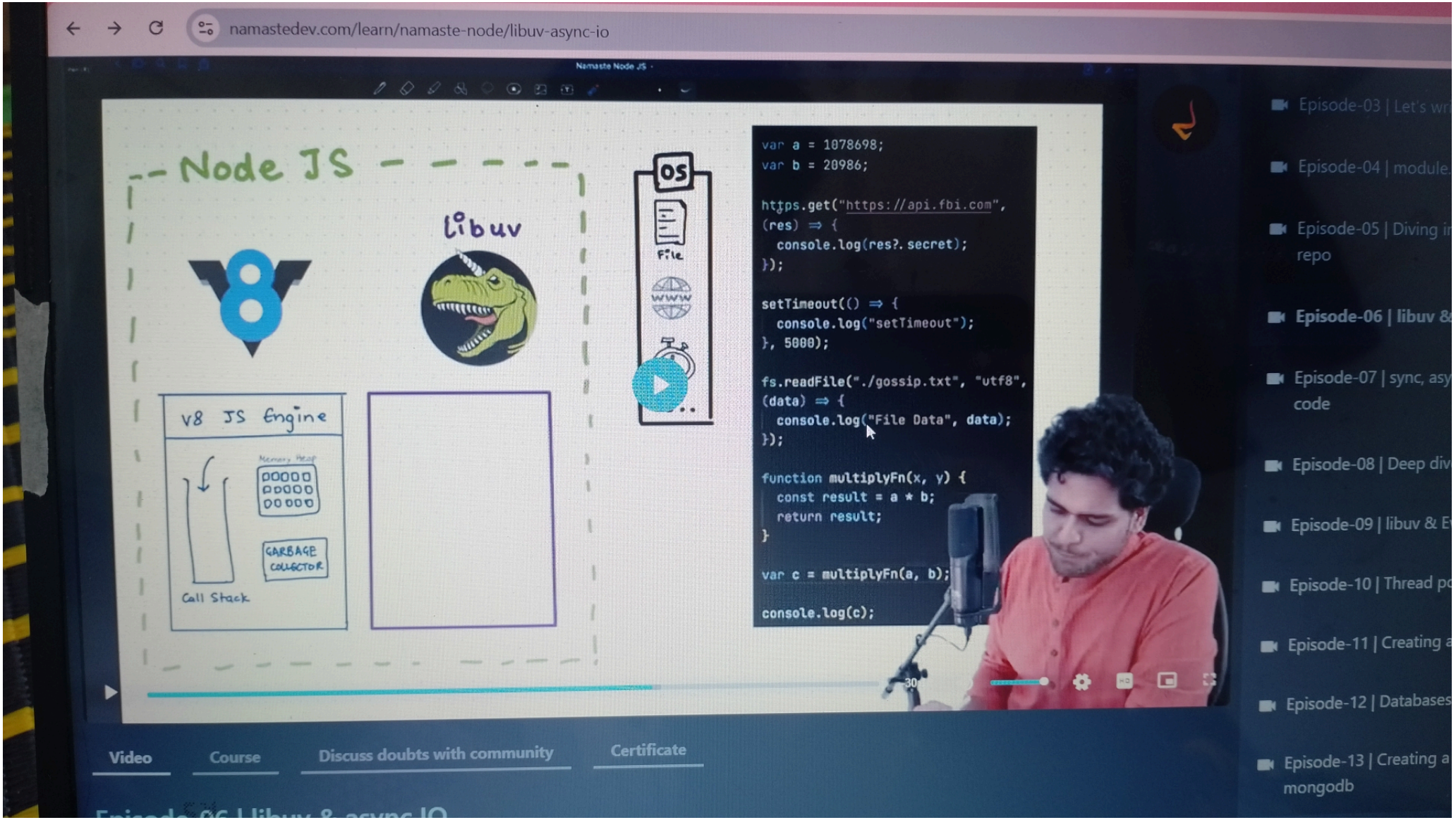

NamasteDev
namastedev.com/learn/namaste-node/libuv-async-io
--NODE JS--
V8 JS Engine
Memory Heap
GARBAGE COLLECTOR
Call Stack
Libuv
SUPER HERO
OS
File
DB
Timer
& more
Namaste NodeJS
6 of 15 Lessons Completed
Season 01 - (Releasing on 09 August 2024)
Episode-00 | Welcome to NamasteNodeJS
Episode-01 | Introduction to NodeJS
Episode-02 | JS on Server
Episode-03 | Let's write code
Episode-04 | module.export & require
Episode-05 | Diving into the NodeJS github repo
Episode-06 | libuv & async IO
Episode-07 | sync, async, setTimeoutZero - code
Video
Course
Discuss doubts with community
Certificate
Episode-06 | libuv & async IO
Search

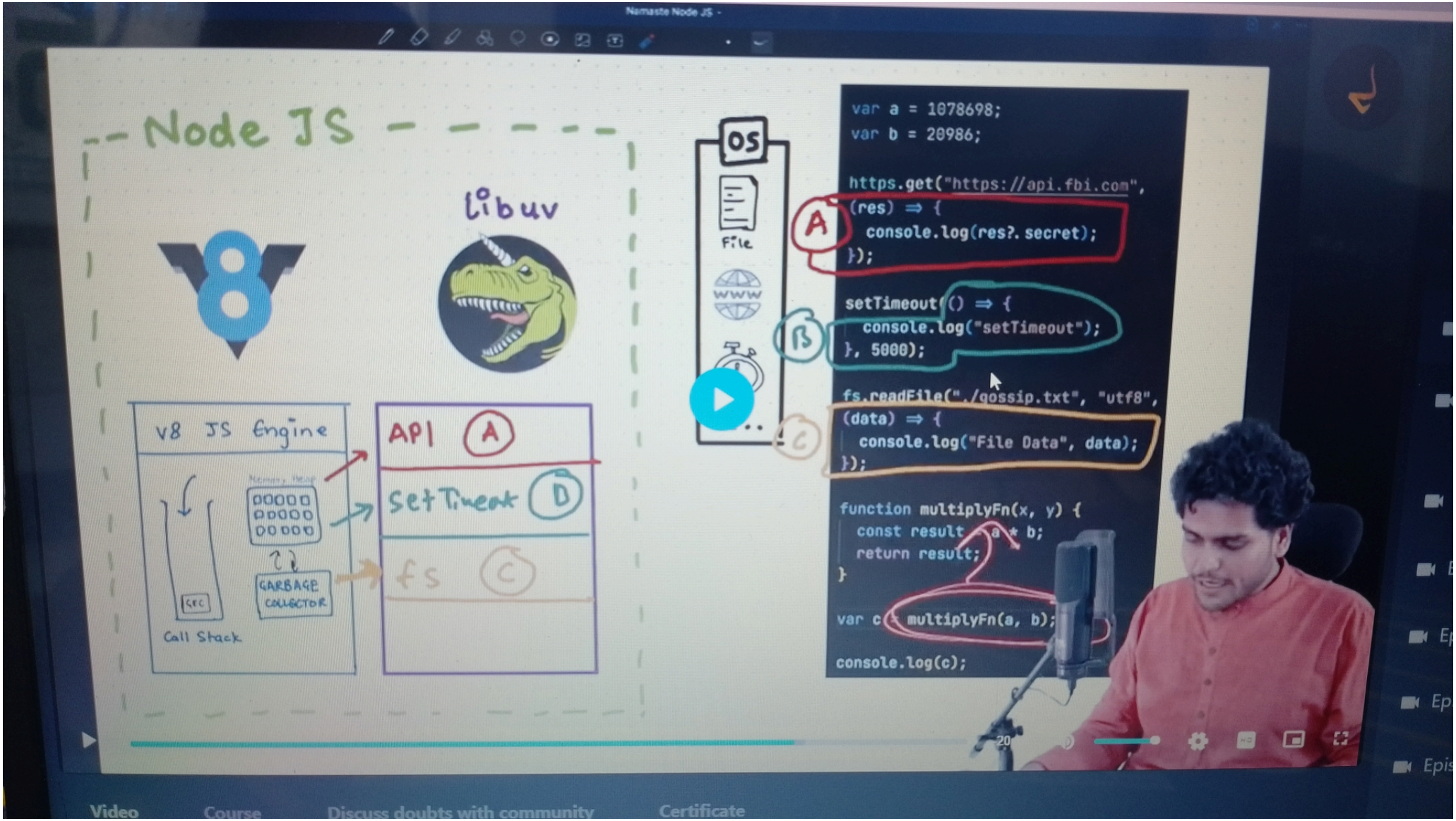

NamasteDev
namastedev.com/learn/namaste-node/libuv-async-io
Namaste Node JS
--NODE JS--
V8 JS Engine
Memory Heap
GARBAGE COLLECTOR
Call Stack
offloads
libuv
OS
File
DB
www
& more
GENIE
SUPER HERO
Episode-00 | Welcome to Namaste
Episode-01 | Introduction to Node
Episode-02 | JS on Server
Episode-03 | Let's write code
Episode-04 | module.export & requ
Episode-05 | Diving into the Node repo
Episode-06 | libuv & async IO
Episode-07 | sync, async, setTimeo code
Episode-08 | Deep dive into v8 JS E
Episode-09 | libuv & Event Loop
Episode-10 | Thread pool in libuv
Episode-11 | Creating a Server
Video
Course
Discuss doubts with community
Certificate
Episode-06 | libuv & async IO

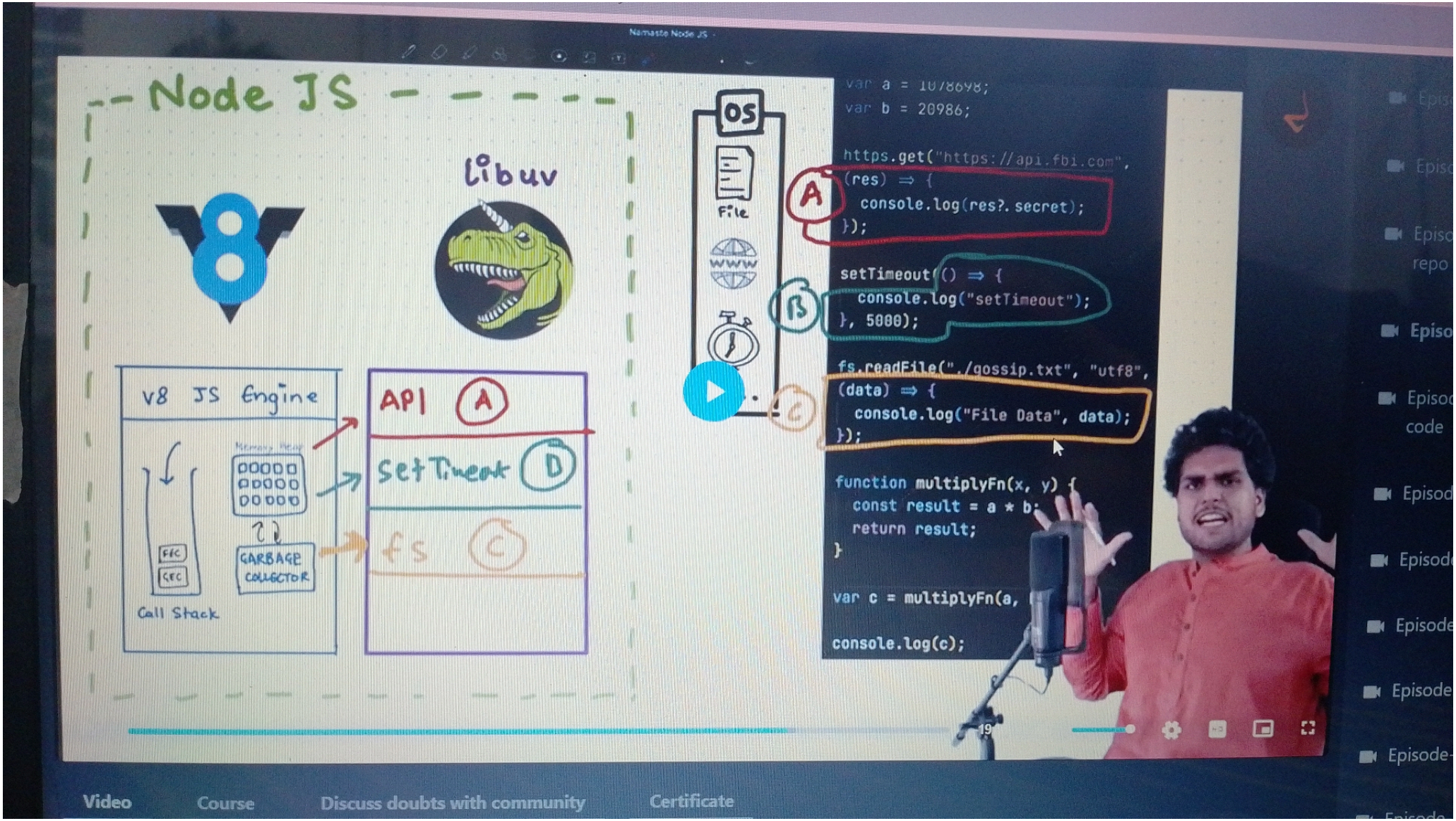

Node JS
OS
File
WWW
https.get("https://api.fbi.com", (res) => {
console.log("secret data:" + res.secret);
});
fs.readFile("./gossip.txt", "utf8", (data) => {
console.log("File Data", data);
});
setTimeout(() => {
console.log("Wait here for 5 seconds");
}, 5000);
44:52

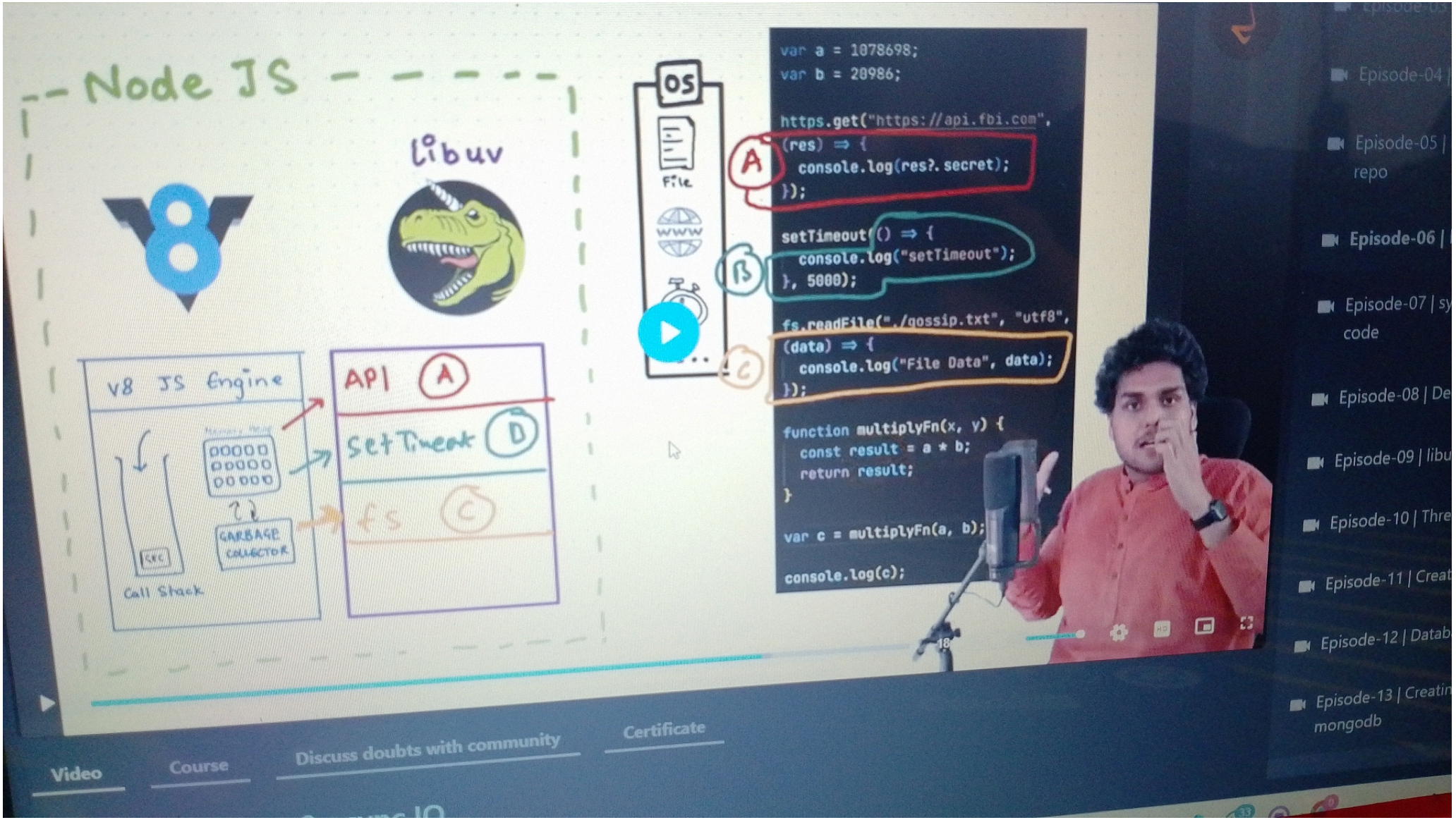

Asynchronous I/O made simple.
libuv is a multi-platform support library with a focus on asynchronous I/O.
Show me the code!

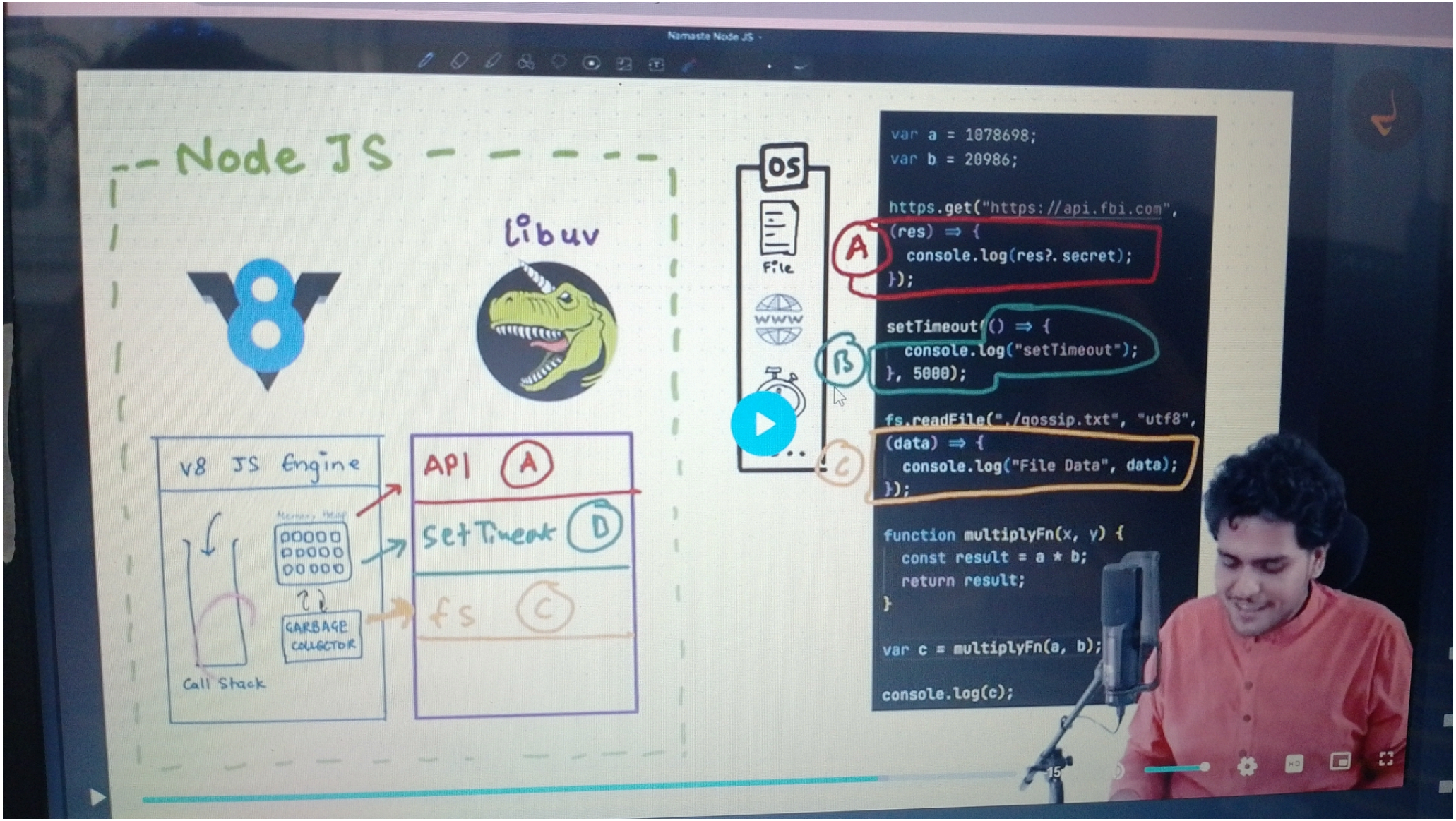

namastedev.com/learn/namaste-node/libuv-async-io
Namaste Node JS
Node JS
C++
C
OS
File
WWW
Async I/O
made simple
https.get("https://api.fbi.com", (res) => {
console.log("secret data:" + res.secret);
});
fs.readFile("./gossip.txt", "utf8", (data) => {
console.log("File Data", data);
});
setTimeout(() => {
console.log("Wait here for 5 seconds");
}, 5000);

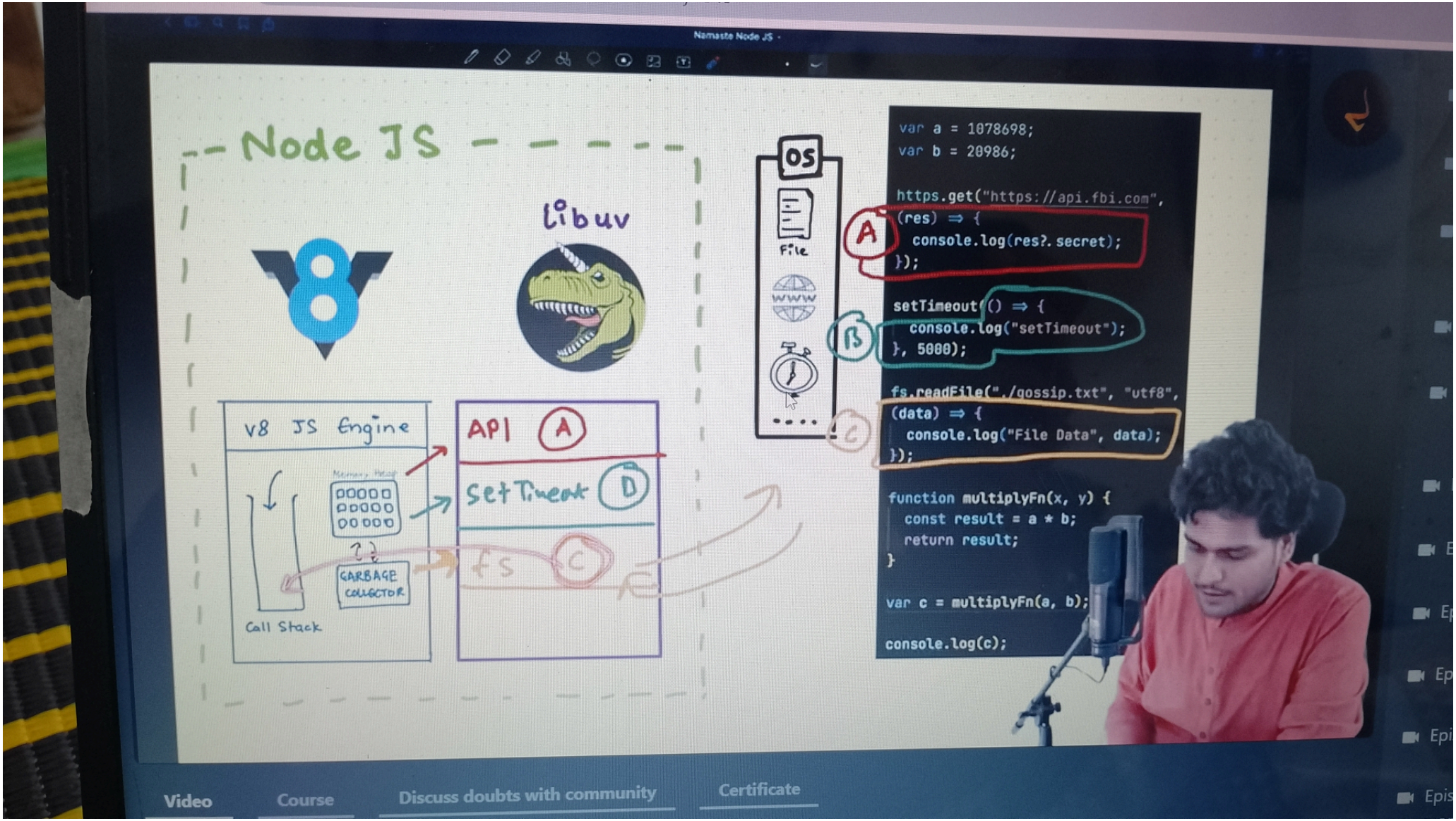

namastedev.com/learn/namaste-node/libuv-async-io
Namaste Node JS
Node JS
libuv
V8 JS Engine
Memory Heap
GARBAGE COLLECTOR
Call Stack
OS
File
WWW
var a = 1078698;
var b = 20986;
https.get("https://api.fbi.com",
(res) => {
console.log(res?.secret);
});
setTimeout(() => {
console.log("setTimeout");
}, 5000);
fs.readFile("./gossip.txt", "utf8",
(data) => {
console.log("File Data", data);
});
function multiplyFn(x, y) {
const result = a * b;
return result;
}
var c = multiplyFn(a, b);
console.log(c);
Episode-03 | Let's wr
Episode-04 | module
Episode-05 | Diving i
repo
Episode-06 | libuv &
Episode-07 | sync, asy
code
Episode-08 | Deep div
Episode-09 | libuv & E
Episode-10 | Thread po
Episode-11 | Creating a
Episode-12 | Databases
Episode-13 | Creating a
mongodb
Video
Course
Discuss doubts with community
Certificate

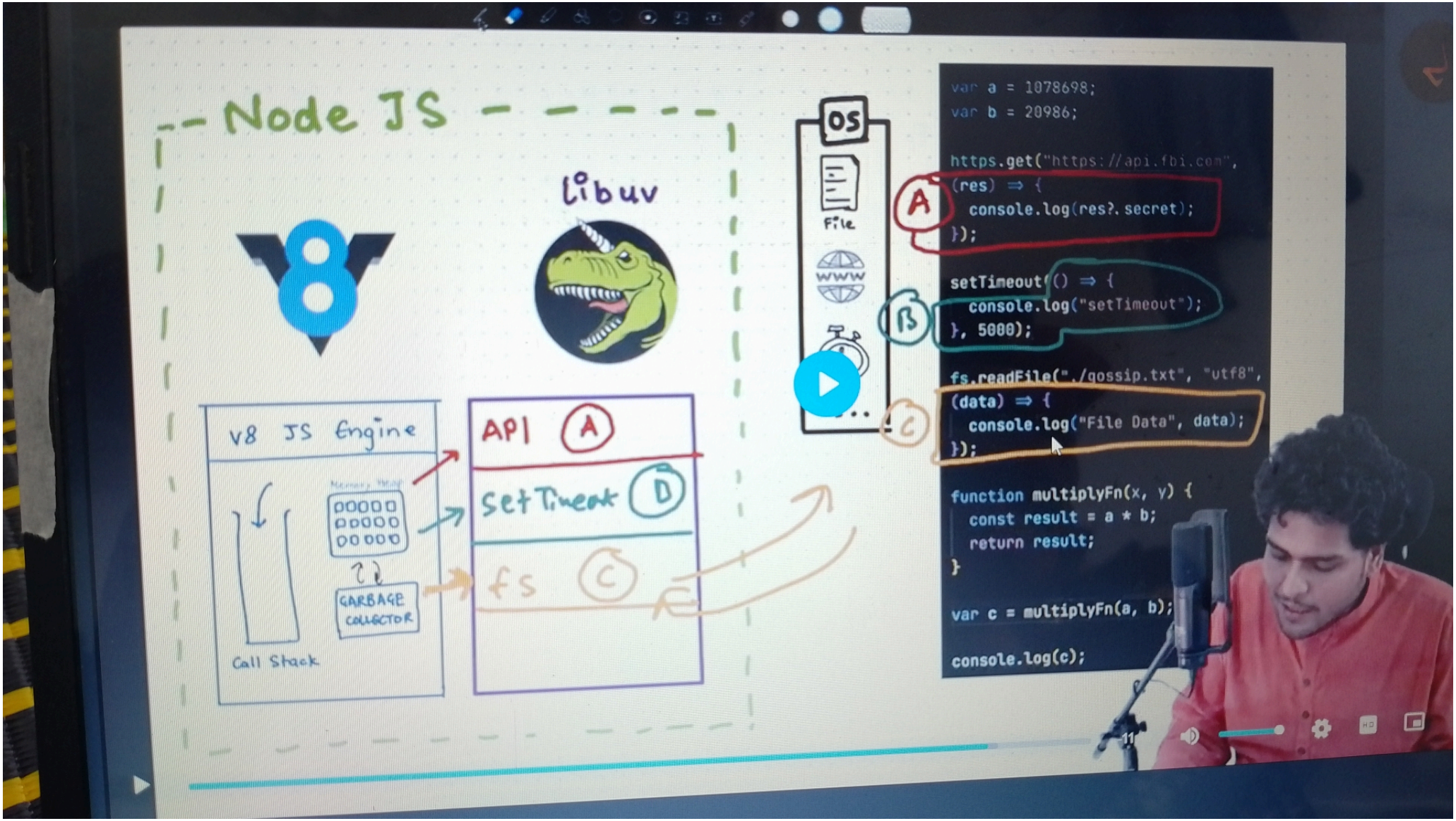

Namaste Node JS -
Node JS
libuv
V8 JS Engine
Memory Heap
GARBAGE COLLECTOR
Call Stack
API
A
setTimeout
B
fs
C
OS
File
WWW
var a = 1078698;
var b = 20986;
https.get("https://api.fbi.com",
(res) => {
console.log(res?.secret);
});
setTimeout(() => {
console.log("setTimeout");
}, 5000);
fs.readFile("./gossip.txt", "utf8",
(data) => {
console.log("File Data", data);
});
function multiplyFn(x, y) {
const result = a * b;
return result;
}
var c = multiplyFn(a, b);
console.log(c);
Video
Course
Discuss doubts with community
Certificate

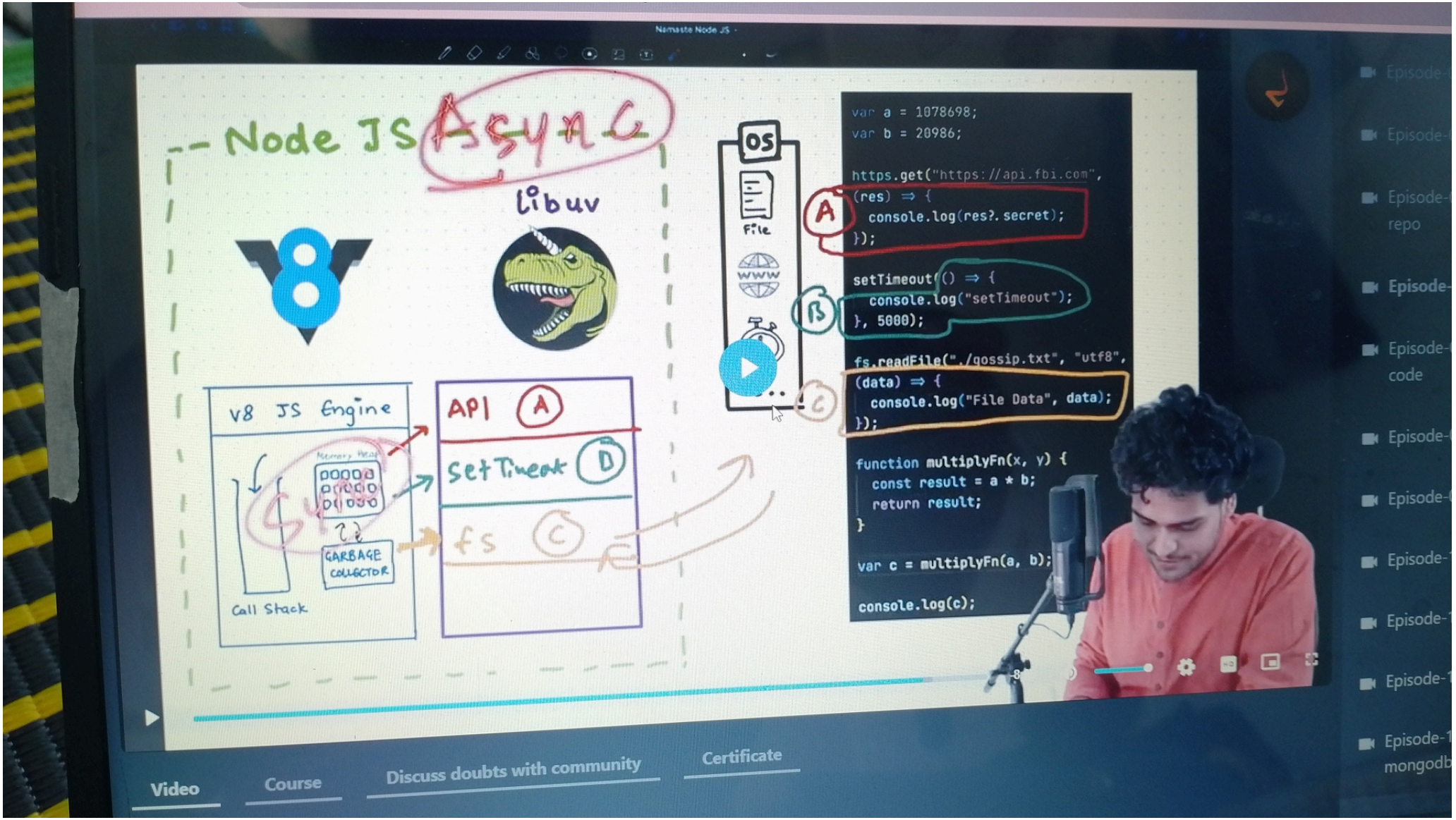

Namaste Node JS
Node JS
libuv
V8 JS Engine
Memory Heap
FFC
CFC
GARBAGE COLLECTOR
Call Stack
API
A
SetTimeout
B
fs
C
OS
File
WWW
var a = 1078698;
var b = 20986;
https.get("https://api.fbi.com",
(res) => {
console.log(res?.secret);
});
setTimeout(() => {
console.log("setTimeout");
}, 5000);
fs.readFile("./gossip.txt", "utf8",
(data) => {
console.log("File Data", data);
});
function multiplyFn(x, y) {
const result = a * b;
return result;
}
var c = multiplyFn(a,
console.log(c);
Episode
repo
code
Video
Course
Discuss doubts with community
Certificate

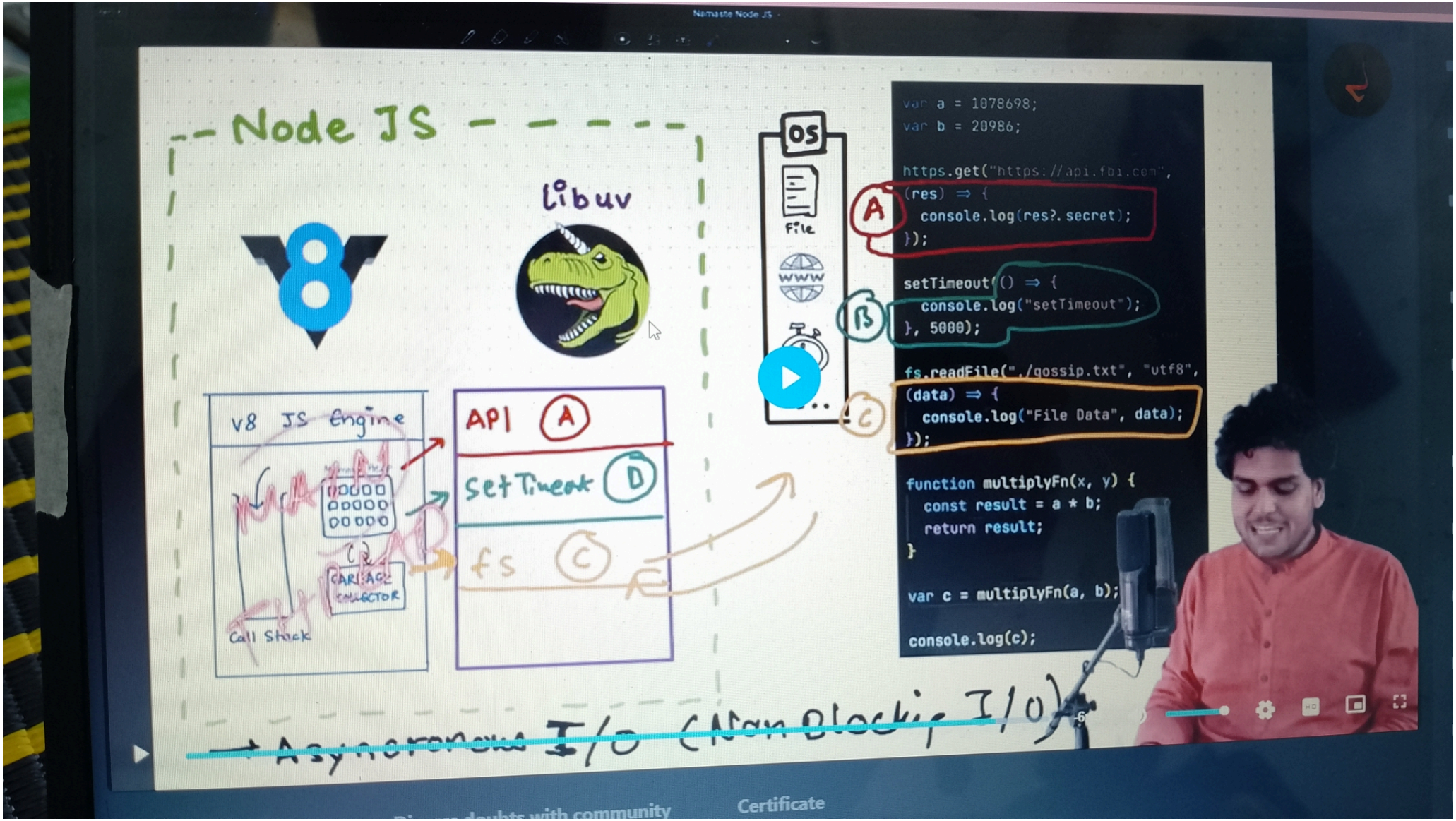

Node JS
libuv
V8 JS Engine
Call Stack
GARBAGE COLLECTOR
API
A
SetTimeout
B
fs
C
OS
File
WWW
var a = 1078698;
var b = 20986;
https.get("https://api.fbi.com",
(res) => {
console.log(res?.secret);
});
setTimeout(() => {
console.log("setTimeout");
}, 5000);
fs.readFile("./gossip.txt", "utf8",
(data) => {
console.log("File Data", data);
});
function multiplyFn(x, y) {
const result = a * b;
return result;
}
var c = multiplyFn(a, b);
console.log(c);
Episode-04
Episode-05 | repo
Episode-06 |
Episode-07 | sy code
Episode-08 | De
Episode-09 | libu
Episode-10 | Thre
Episode-11 | Creat
Episode-12 | Datab
Episode-13 | Creatin mongodb
Video
Course
Discuss doubts with community
Certificate


Namaste Node JS
Node JS
libuv
V8 JS Engine
Memory Heap
GARBAGE COLLECTOR
Call Stack
API
A
SetTimeout
D
fs
C
OS
File
WWW
var a = 1078698;
var b = 20986;
https.get("https://api.fbi.com",
(res) => {
console.log(res?.secret);
});
setTimeout(() => {
console.log("setTimeout");
}, 5000);
fs.readFile("./gossip.txt", "utf8",
(data) => {
console.log("File Data", data);
});
function multiplyFn(x, y) {
const result = a * b;
return result;
}
var c = multiplyFn(a, b);
console.log(c);
A
B
C
15


Namaste Node JS
Node JS
libuv
V8 JS Engine
Memory Heap
GARBAGE COLLECTOR
Call Stack
API
A
setTimeout
B
fs
C
OS
File
WWW
D
var a = 1078698;
var b = 20986;
https.get("https://api.fbi.com",
(res) => {
console.log(res?.secret);
});
setTimeout(() => {
console.log("setTimeout");
}, 5000);
fs.readFile("./gossip.txt", "utf8",
(data) => {
console.log("File Data", data);
});
function multiplyFn(x, y) {
const result = a * b;
return result;
}
var c = multiplyFn(a, b);
console.log(c);
Video
Course
Discuss doubts with community
Certificate


Node JS
libuv
V8 JS Engine
Memory Heap
GARBAGE COLLECTOR
Call Stack
API
A
SetTimeout
B
fs
C
OS
File
WWW
A
B
C
var a = 1078698;
var b = 20986;
https.get("https://api.fbi.com",
(res) => {
console.log(res?.secret);
});
setTimeout(() => {
console.log("setTimeout");
}, 5000);
fs.readFile("./gossip.txt", "utf8",
(data) => {
console.log("File Data", data);
});
function multiplyFn(x, y) {
const result = a * b;
return result;
}
var c = multiplyFn(a, b);
console.log(c);


Namaste Node JS
Node JS Async
libuv
v8 JS Engine
Memory Heap
Call Stack
GARBAGE COLLECTOR
API
A
setTimeout
B
fs
C
OS
File
WWW
var a = 1078698;
var b = 20986;
https.get("https://api.fbi.com",
(res) => {
console.log(res?.secret);
});
setTimeout(() => {
console.log("setTimeout");
}, 5000);
fs.readFile("./gossip.txt", "utf8",
(data) => {
console.log("File Data", data);
});
function multiplyFn(x, y) {
const result = a * b;
return result;
}
var c = multiplyFn(a, b);
console.log(c);
Video
Course
Discuss doubts with community
Certificate
Episode-
repo
code
mongodb


Namaste Node JS
Node JS
libuv
V8 JS Engine
Call Stack
API
A
setTimeout
B
fs
C
OS
File
WWW
D
var a = 1078698;
var b = 20986;
https.get("https://api.fbi.com",
(res) => {
console.log(res?.secret);
});
setTimeout(() => {
console.log("setTimeout");
}, 5000);
fs.readFile("./gossip.txt", "utf8",
(data) => {
console.log("File Data", data);
});
function multiplyFn(x, y) {
const result = a * b;
return result;
}
var c = multiplyFn(a, b);
console.log(c);
Asynchronous I/O (Non Blocking I/O)
Certificate